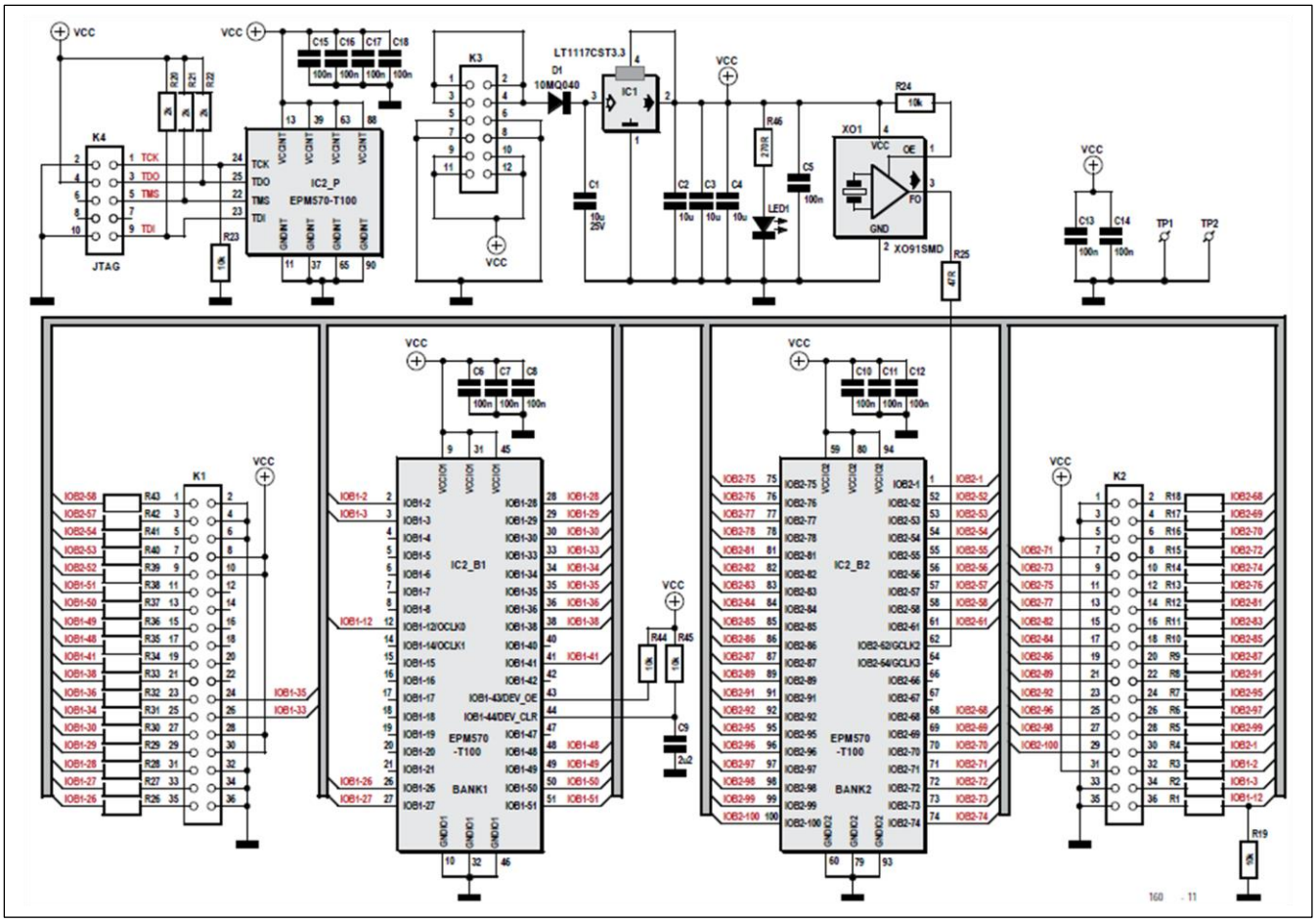
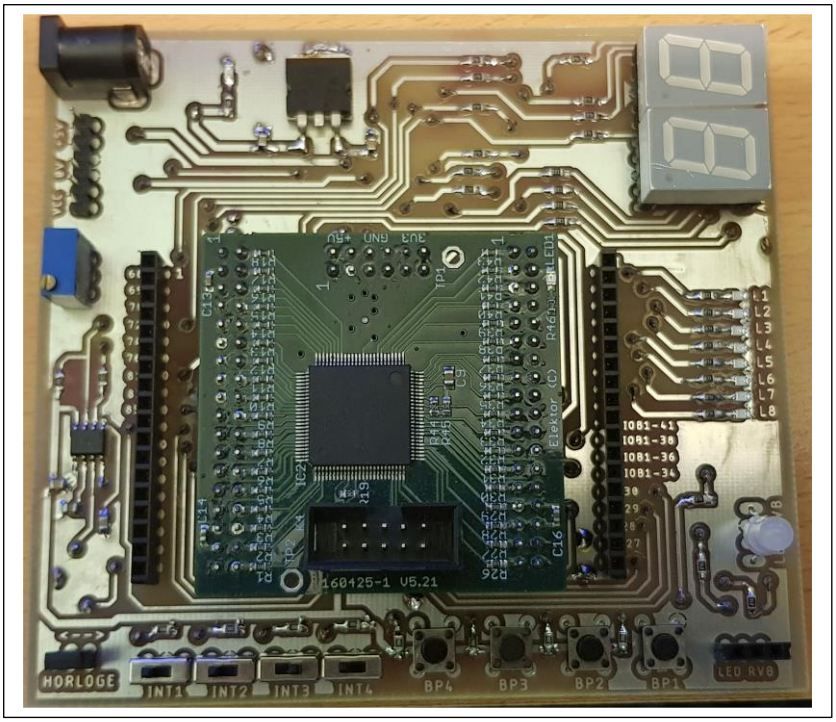
Carte CPLD ELEKTOR – Schéma structurel.



INTERFACE CARTE CPLD.



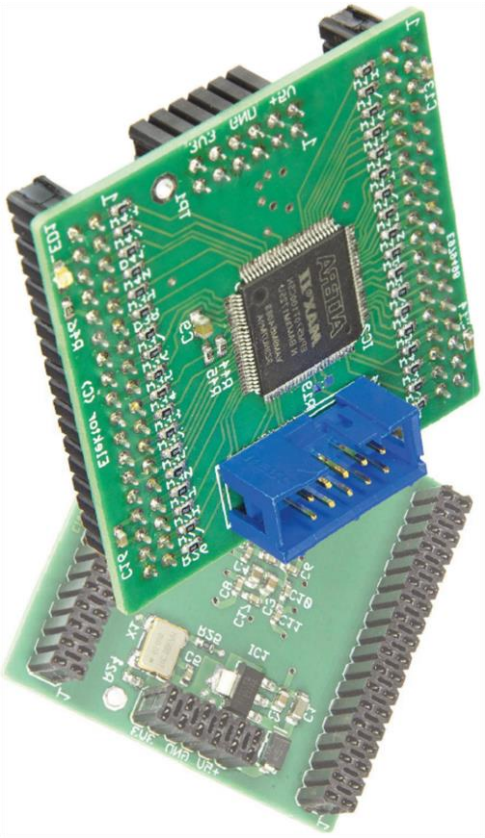
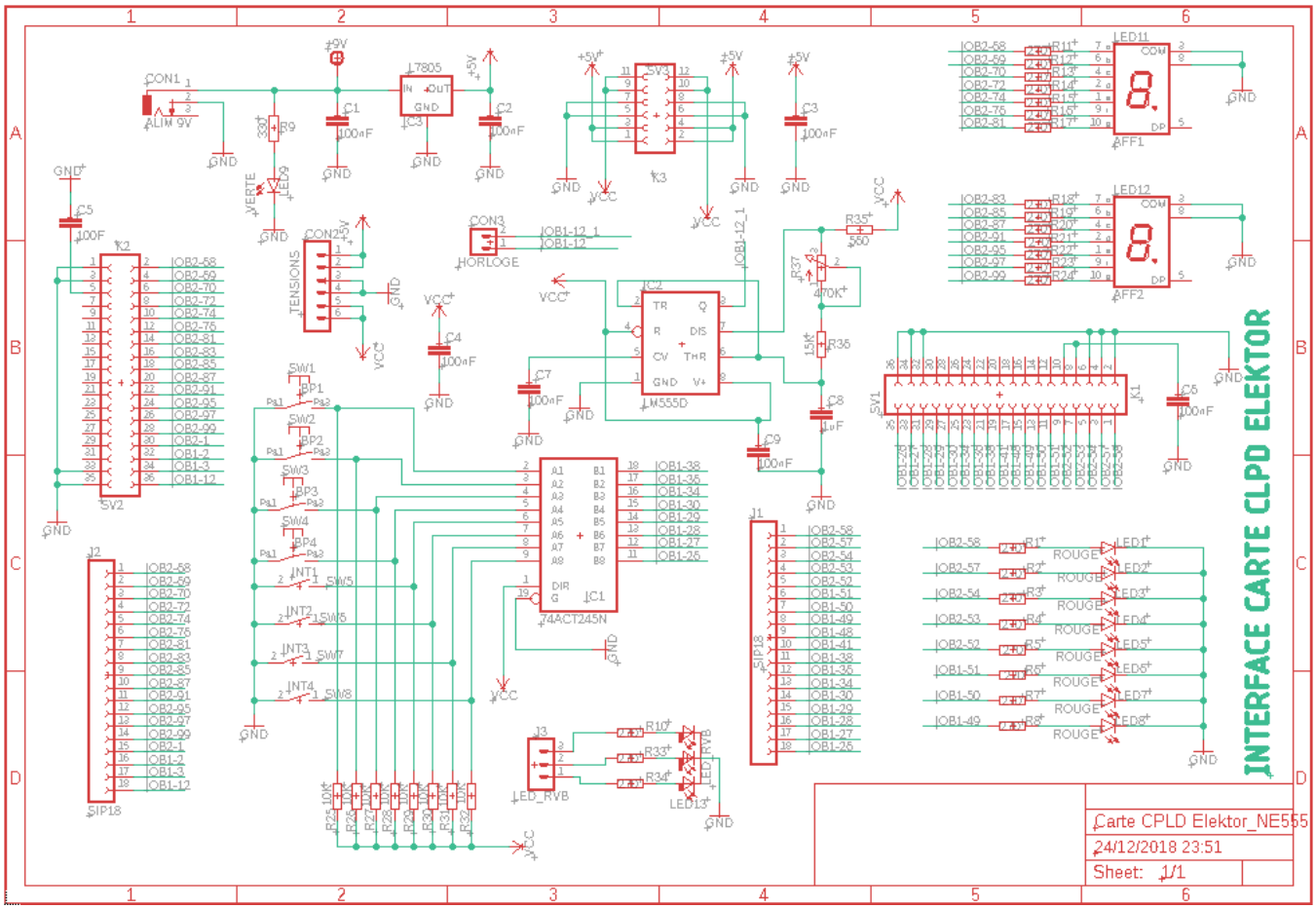


Schéma structurel.



Coté composants.

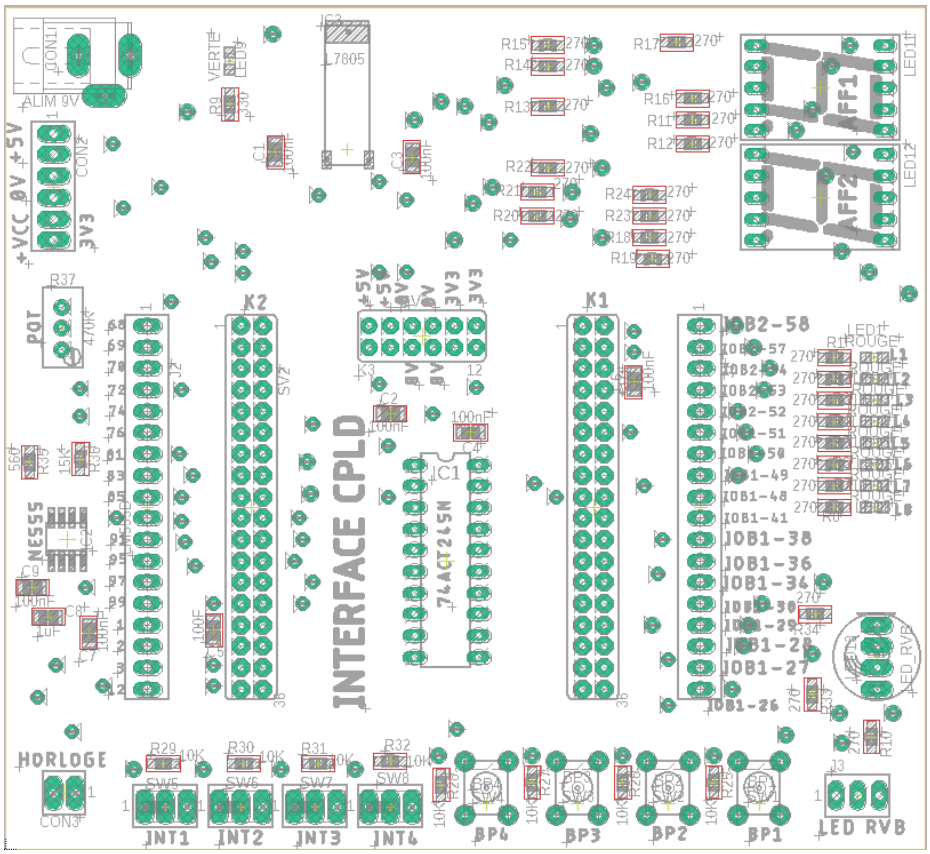
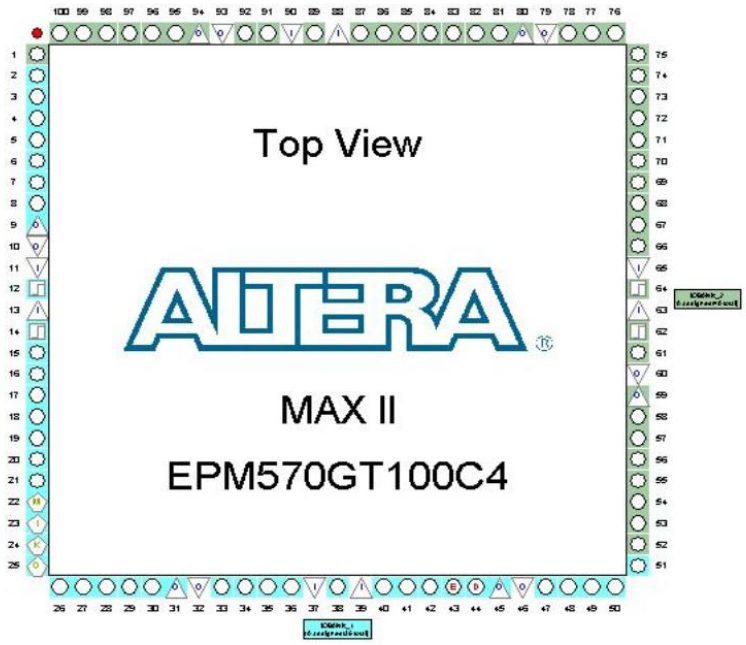
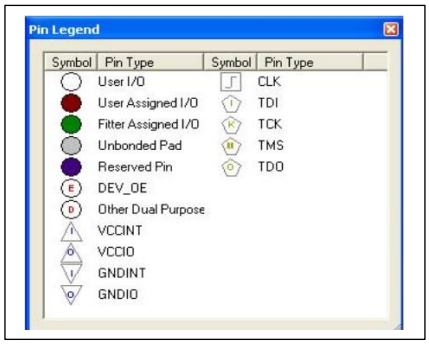


Figure 1. MAX II EPM570 / EPM570G T100 Device Top View Package Diagram and Bank Information





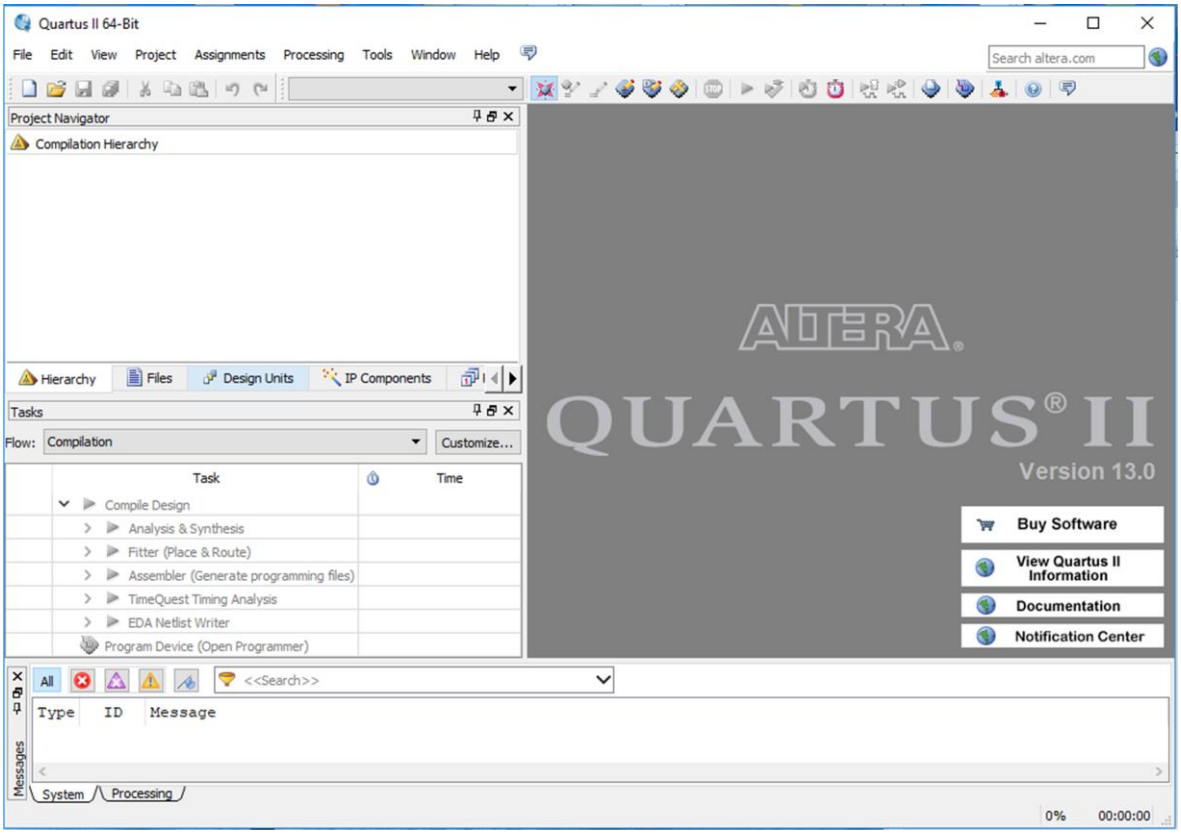
Affectation des broches.

N° Broche	Nom	Affectation	Remarque
Afficheurs 7-Segments : **AFF1** et **AFF2**			
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"**	
Les Leds **L1**, **L2**, **L3**, **L4**, **L5**, **L6**, **L7** et **L8**.			
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**	

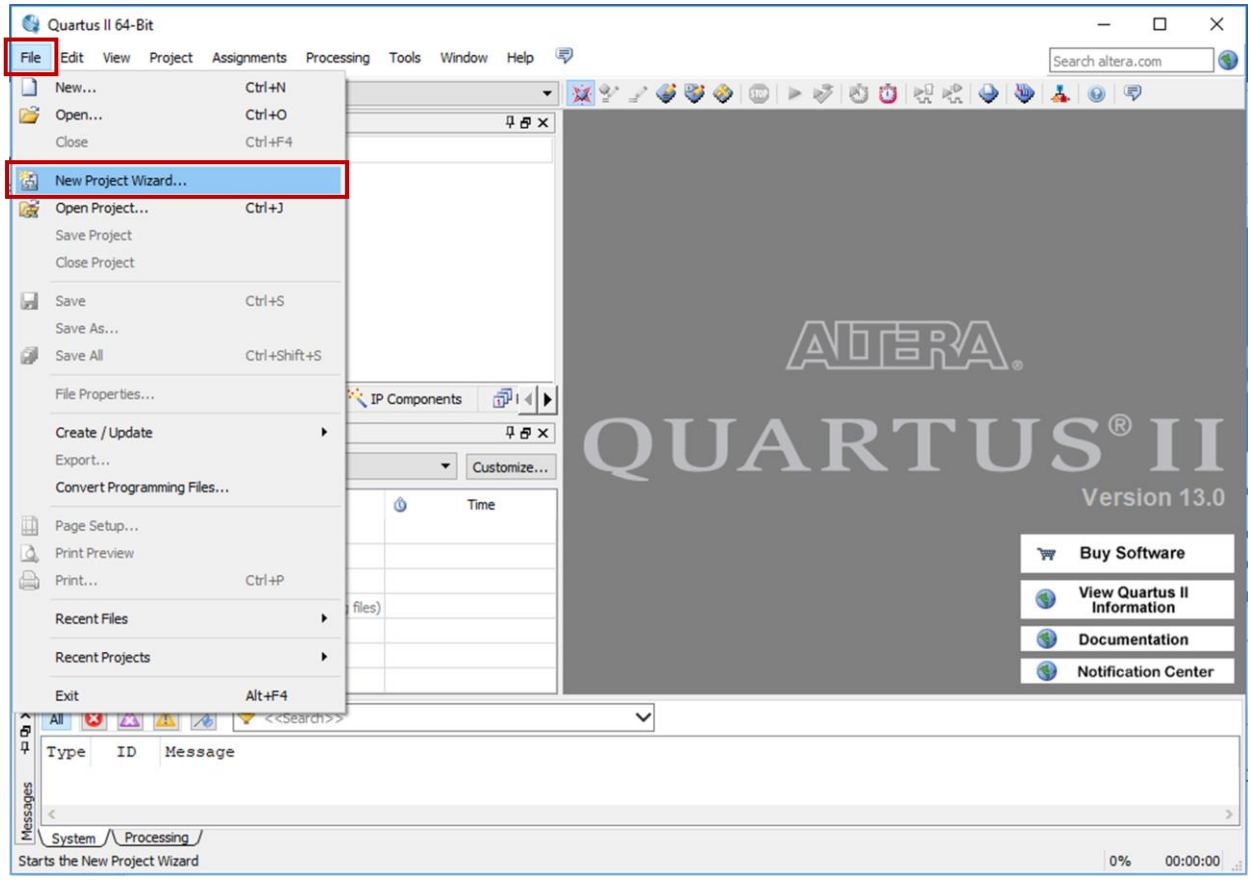
Les boutons poussoirs **BP1**, **BP2**, **BP3** et **BP4**.			
38	IOB1-38	Lecture état du bouton poussoir **BP1**	
36	IOB1-36	Lecture état du bouton poussoir **BP2**	
34	IOB1-34	Lecture état du bouton poussoir **BP3**	
30	IOB1-30	Lecture état du bouton poussoir **BP4**	
Les interrupteurs **INT1**, **INT2**, **INT3** et **INT4**.			
29	IOB1-29	Lecture état de l'interrupteur **INT1**	
28	IOB1-28	Lecture état de l'interrupteur **INT2**	
27	IOB1-27	Lecture état de l'interrupteur **INT3**	
26	IOB1-26	Lecture état de l'interrupteur **INT4**	
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	

<u>QUARTUS</u> : Outil de développement et de téléversement de programmes pour les puces Altera.

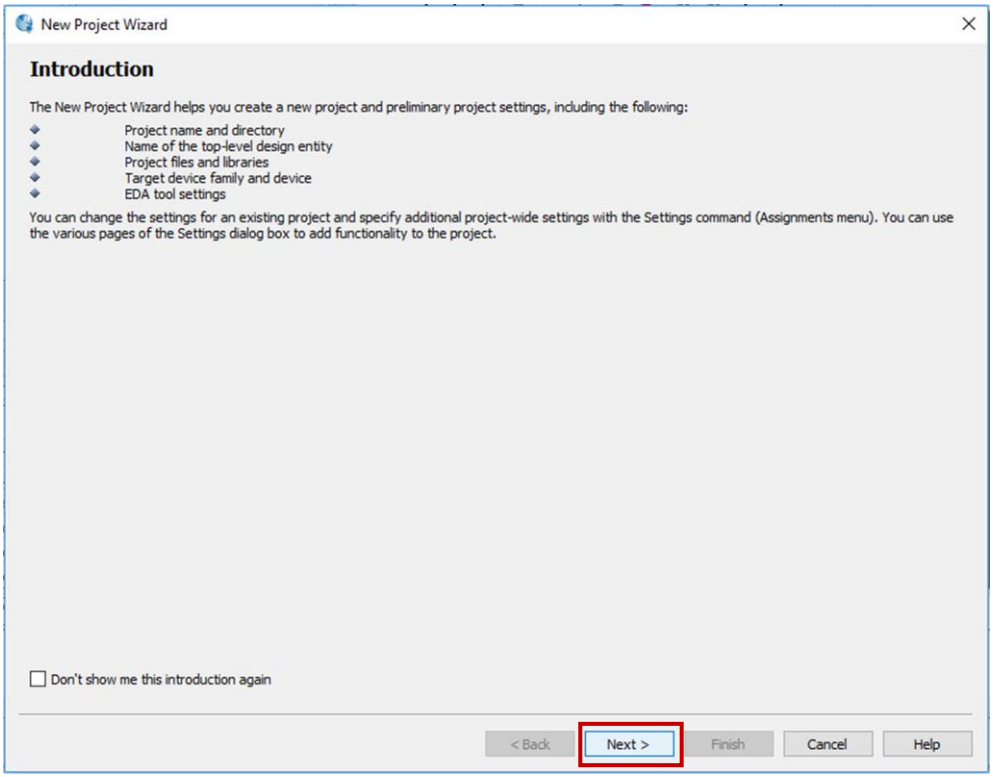
☞ Lancer le logiciel en cliquant sur l'icône **"QUARTUS II"** 

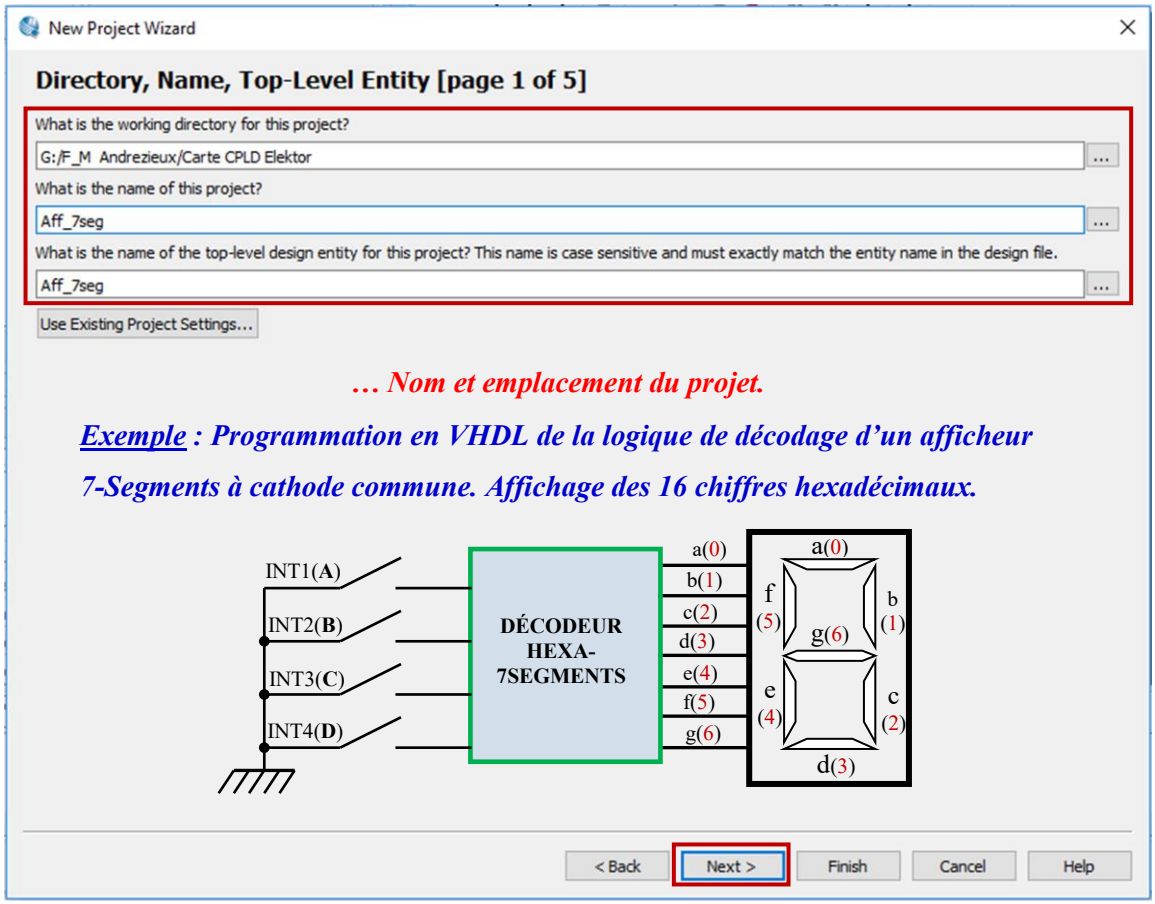


… Choisir ensuite l'assistant pour commencer un nouveau programme.



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

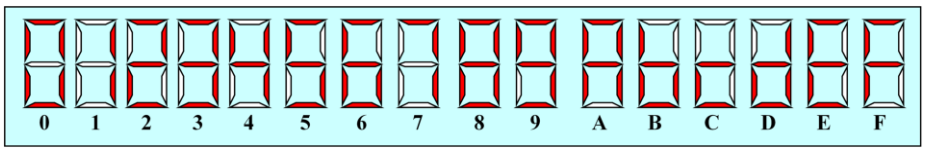


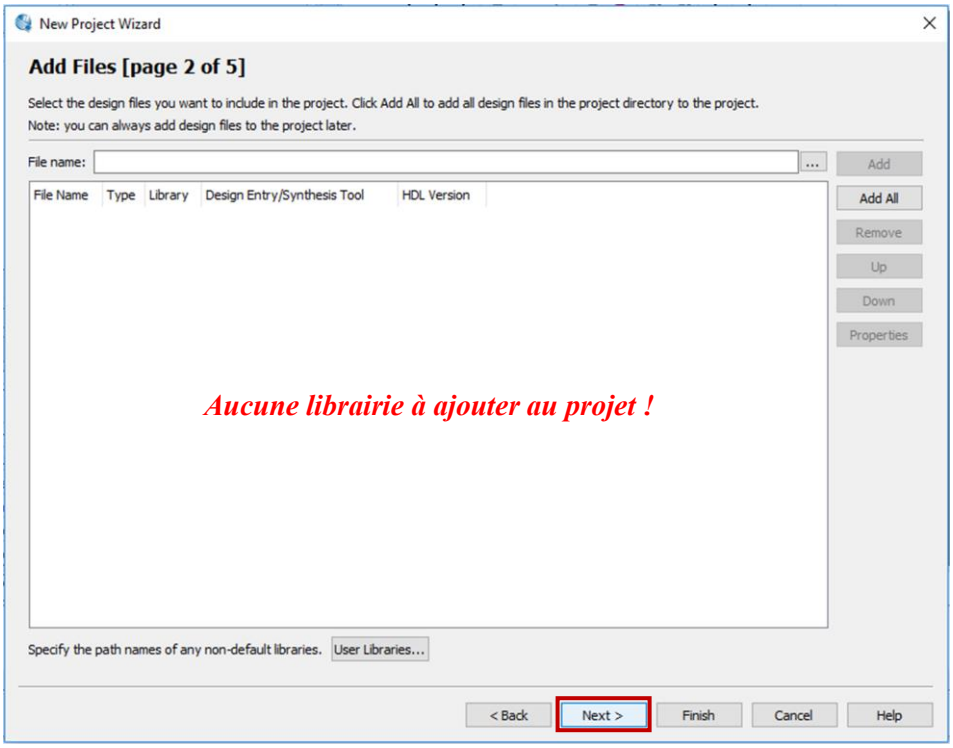


… Nom et emplacement du projet.

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

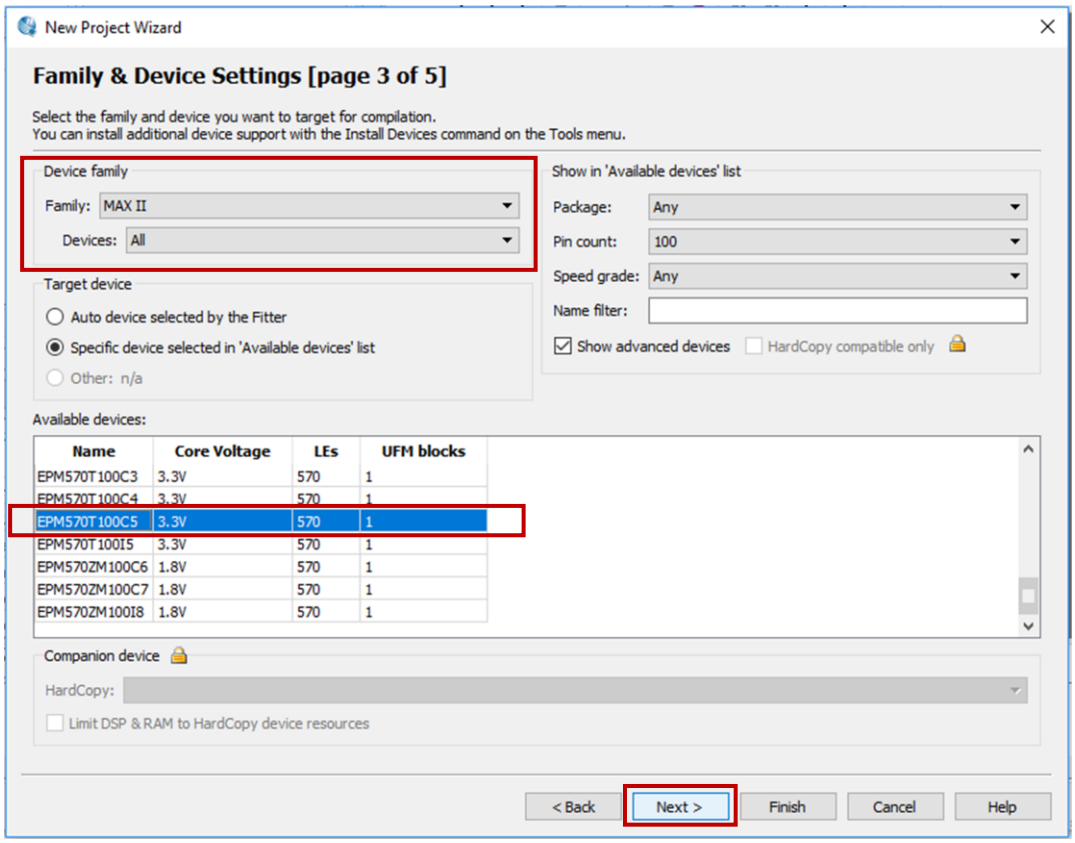
Affichage des chiffres décimaux et hexadécimaux.



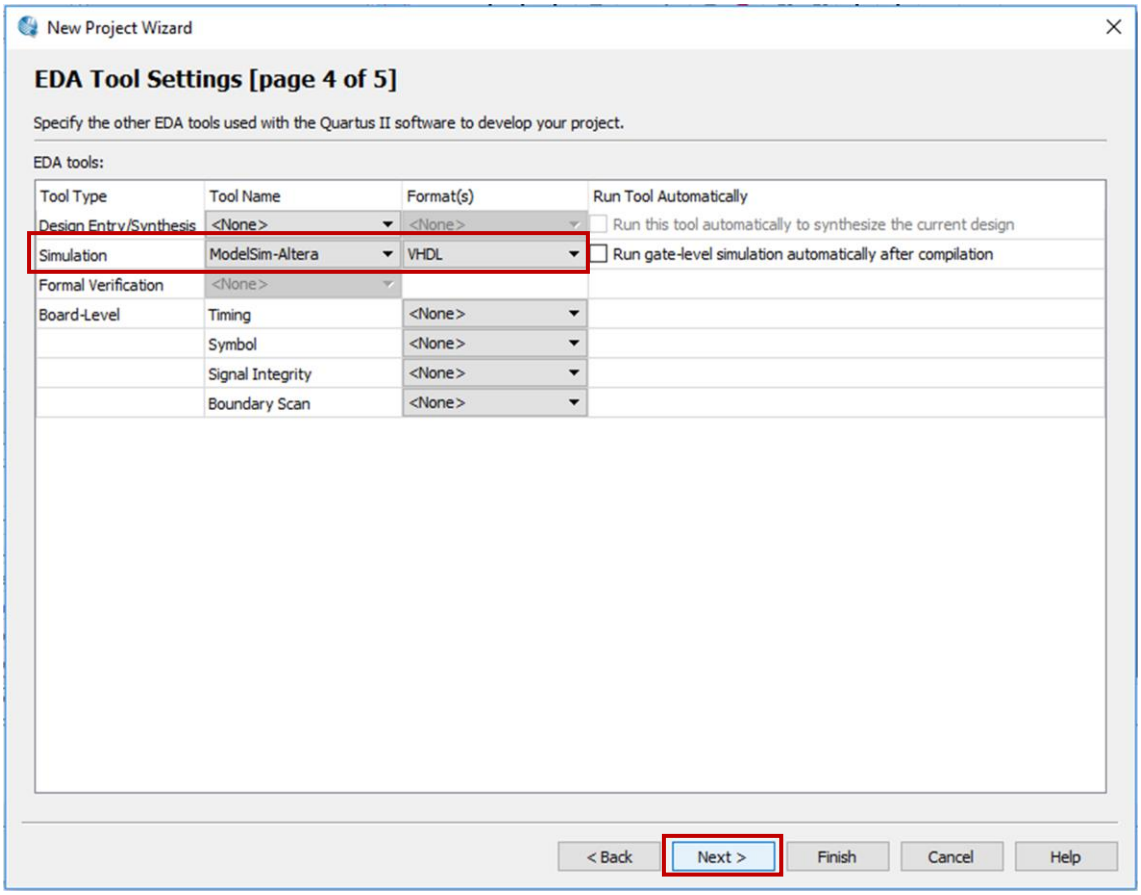


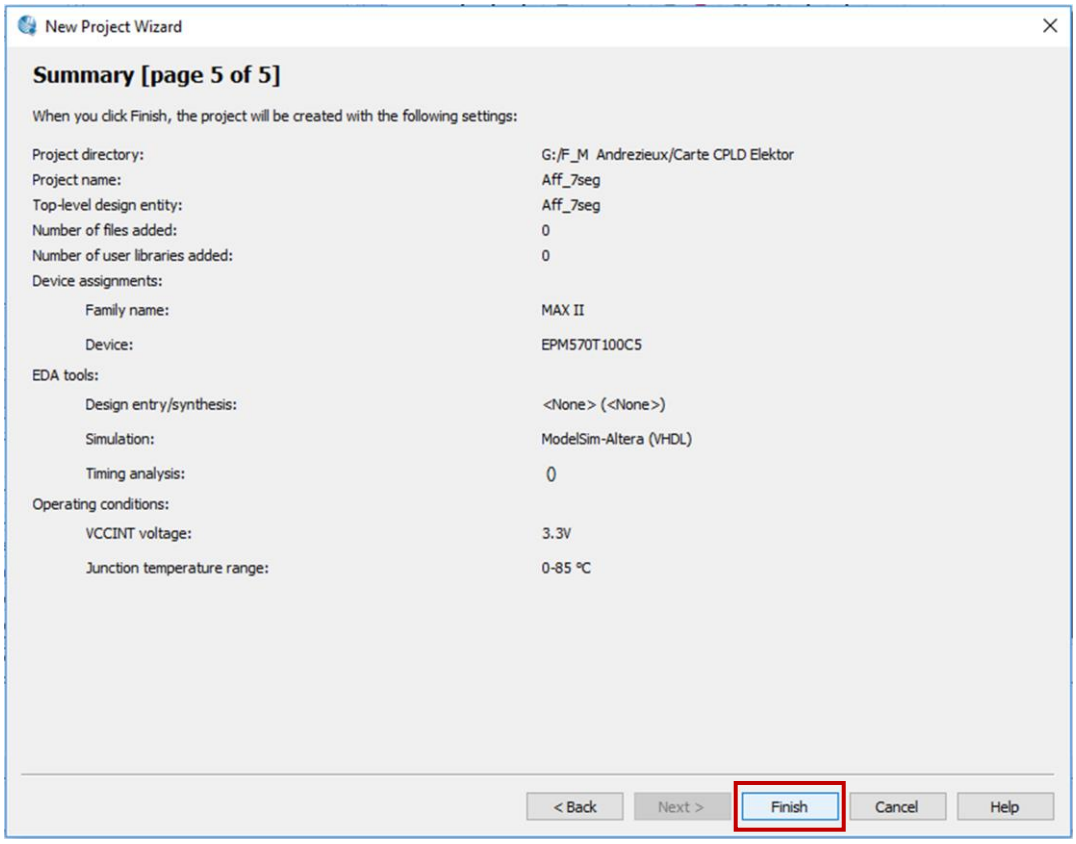
Aucune librairie à ajouter au projet !

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



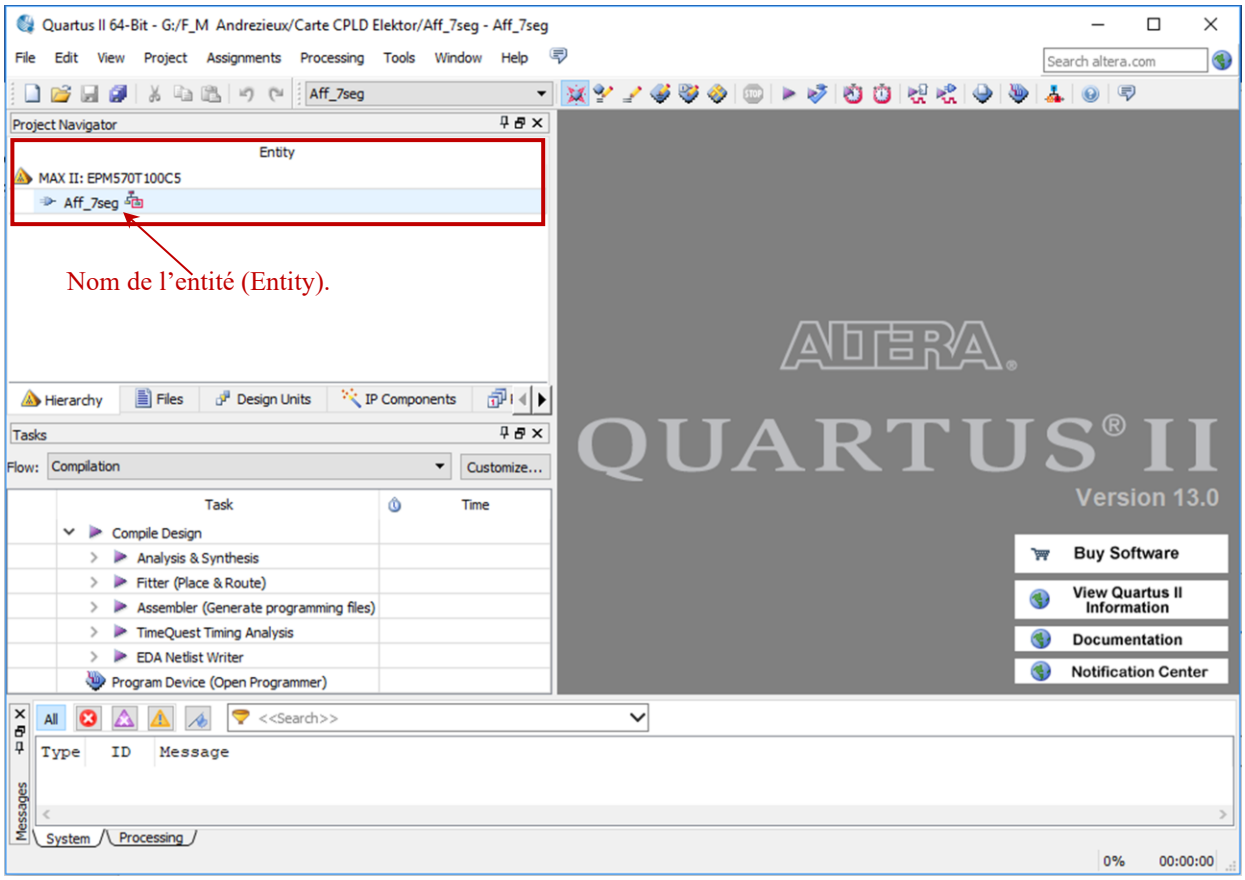
☞ ***Choisir le module de simulation : ModelSim-Altera VHDL.***



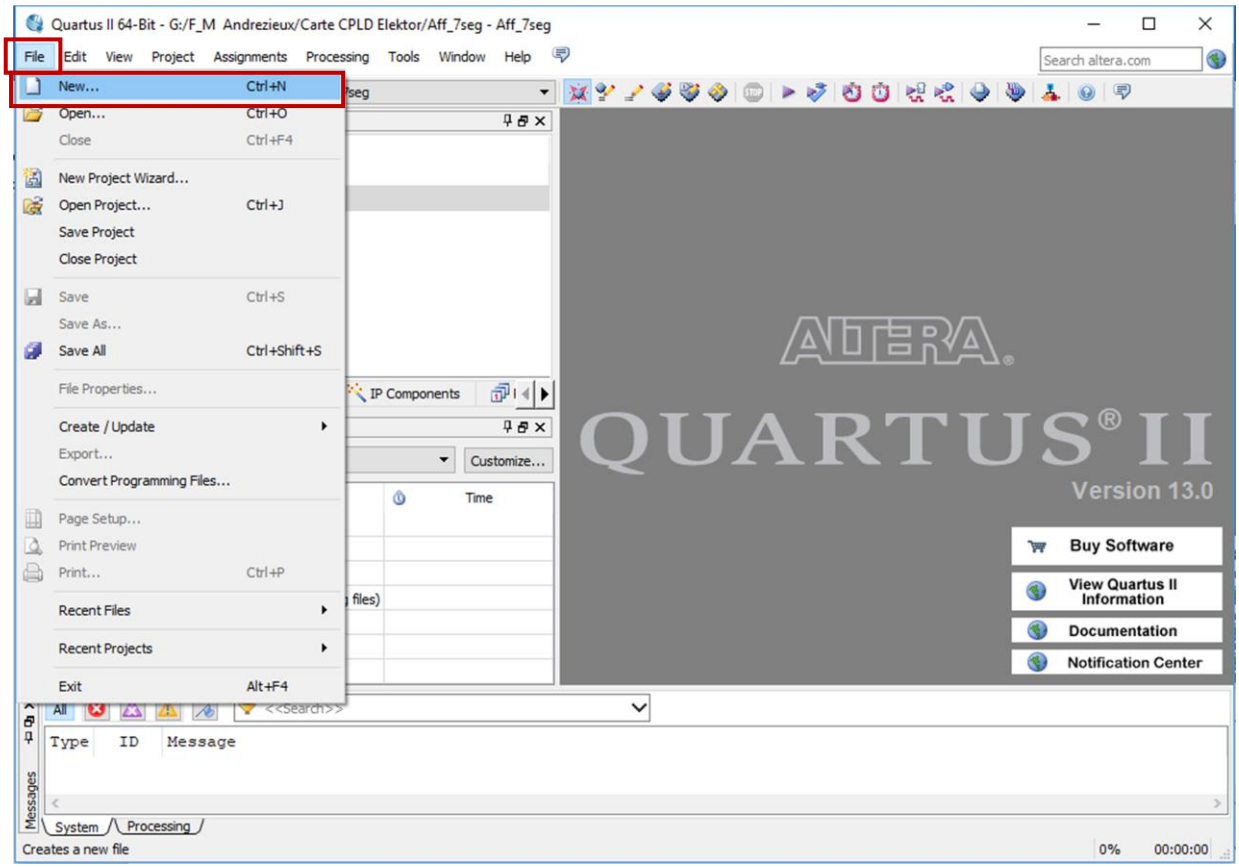


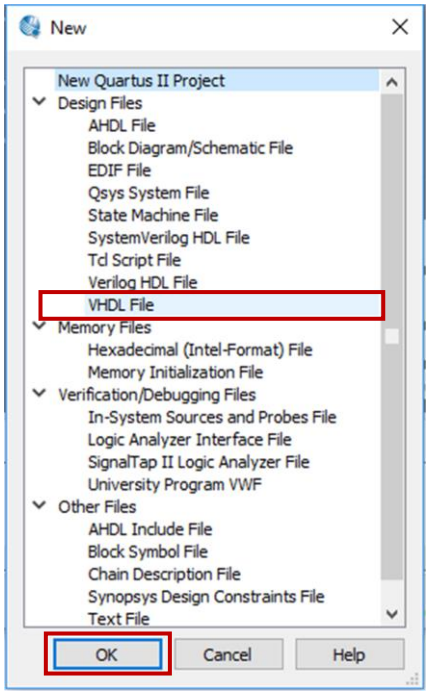
☞ ***Cliquer sur "Finish" pour fermer l'assistant.***

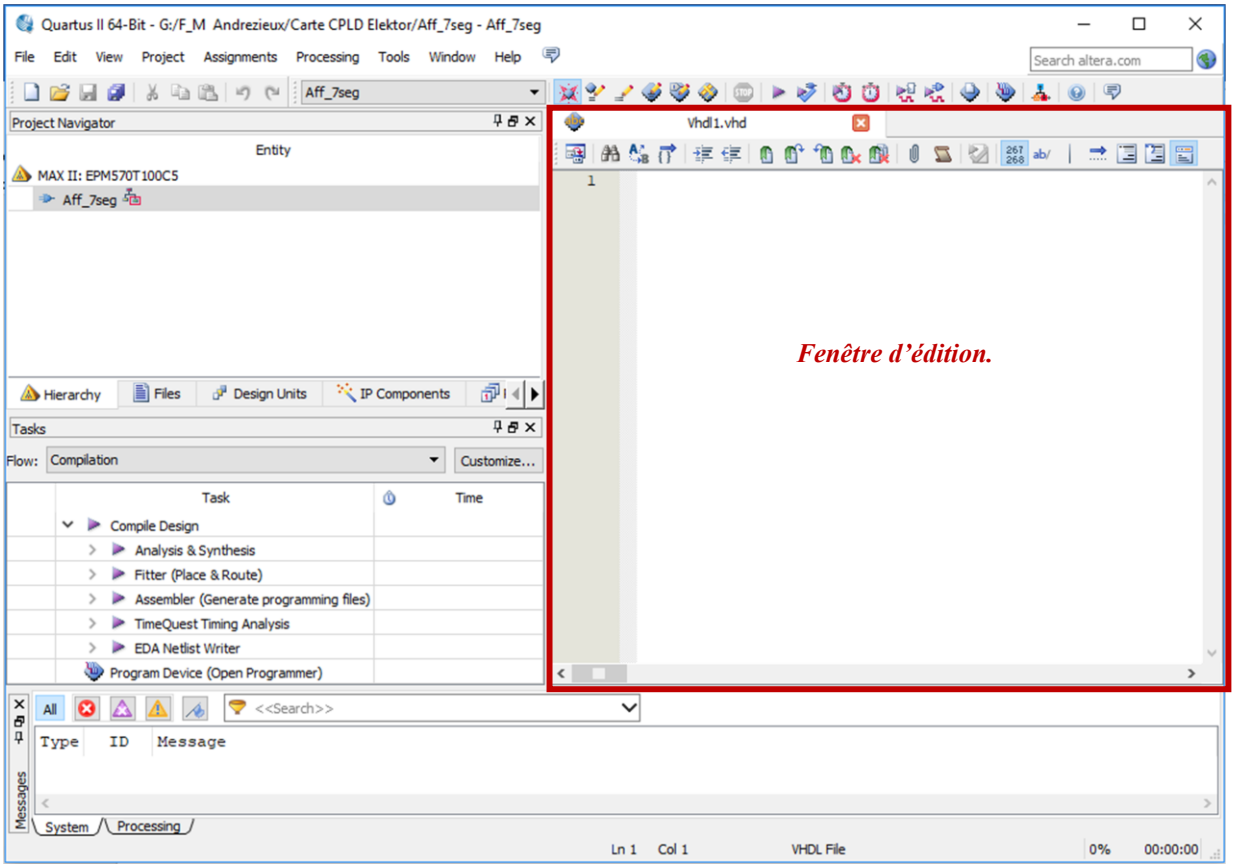
✍ Environnement de développement "QUARTUS".



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

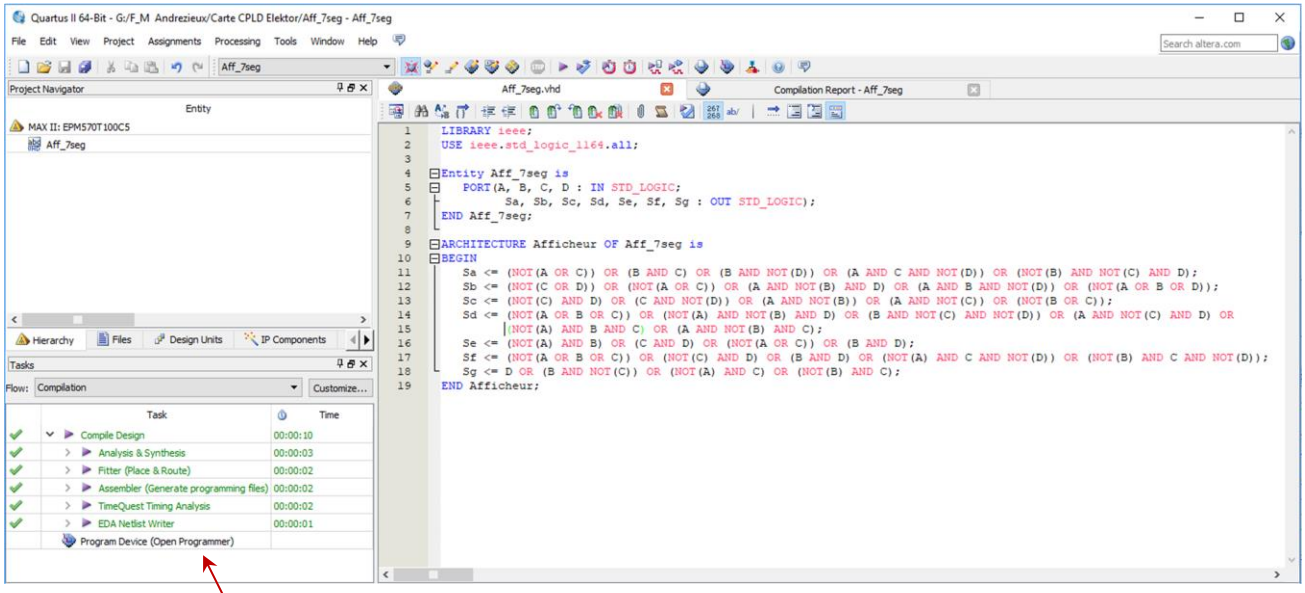






☺ **Programme du décodeur BCD-7Segments.** **Les équations simplifiées des segments sont données !**

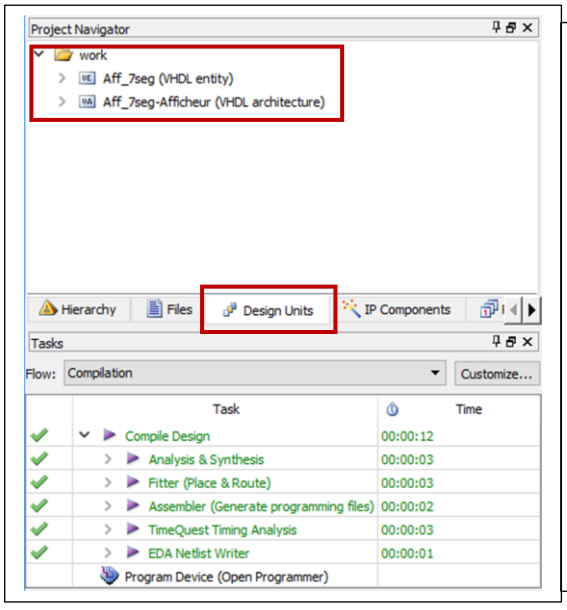
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

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;
```

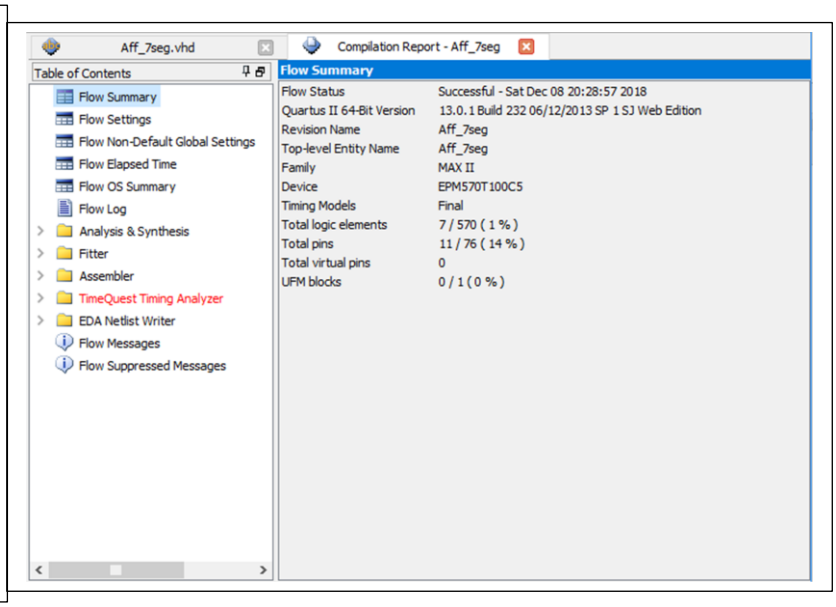


Résultat de la compilation.

☺ A la fin de la compilation, un dossier **"work"** est crée …

☺ La fenêtre **"Flow Summary"** résume les éléments du projet et l'utilisation de la puce CPLD.





Affectation des broches.

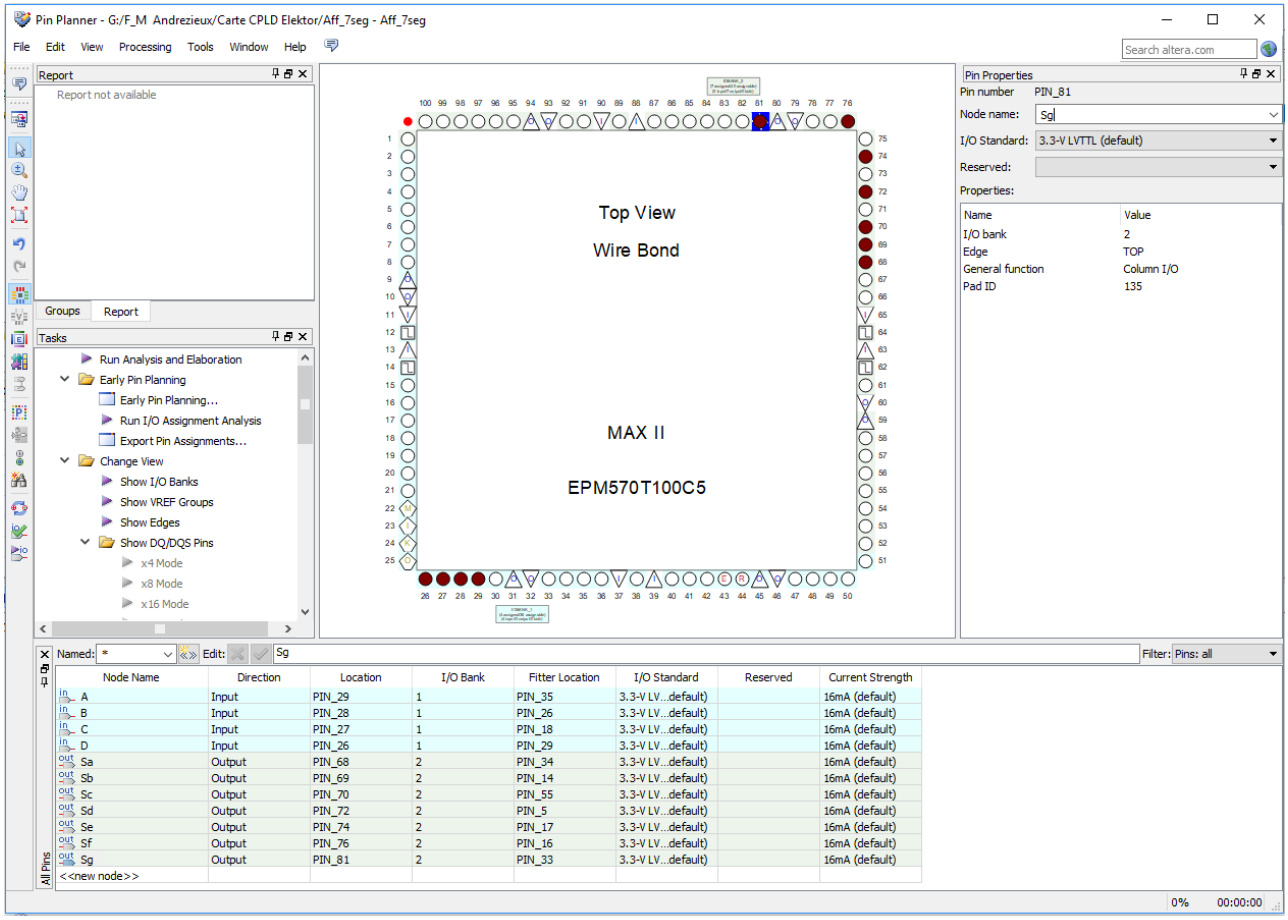
Les interrupteurs **INT1**, **INT2**, **INT3** et **INT4**.			
29	IOB1-29	Lecture état de l'interrupteur **INT1**	
28	IOB1-28	Lecture état de l'interrupteur **INT2**	
27	IOB1-27	Lecture état de l'interrupteur **INT3**	
26	IOB1-26	Lecture état de l'interrupteur **INT4**	

Afficheurs 7-Segments : **AFF1** et **AFF2**			
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"**	

Broches utilisées.

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

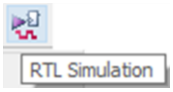
☞ Un double clic sur la broche choisie ouvre une fenêtre décrivant ses propriétés. Les affectations des broches de la puce EPM570T100C5 aux broches de l'afficheur et aux interrupteurs se fera conformément au tableau donné ci-dessus.

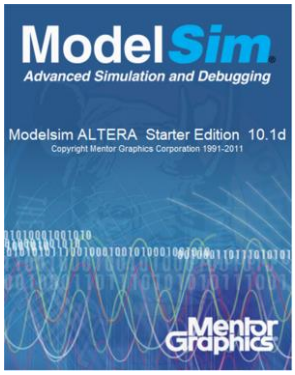


☺ Une fois l'affectation des broches réalisée, recompiler le projet en cliquant sur l'icône **"Start Compilation".**

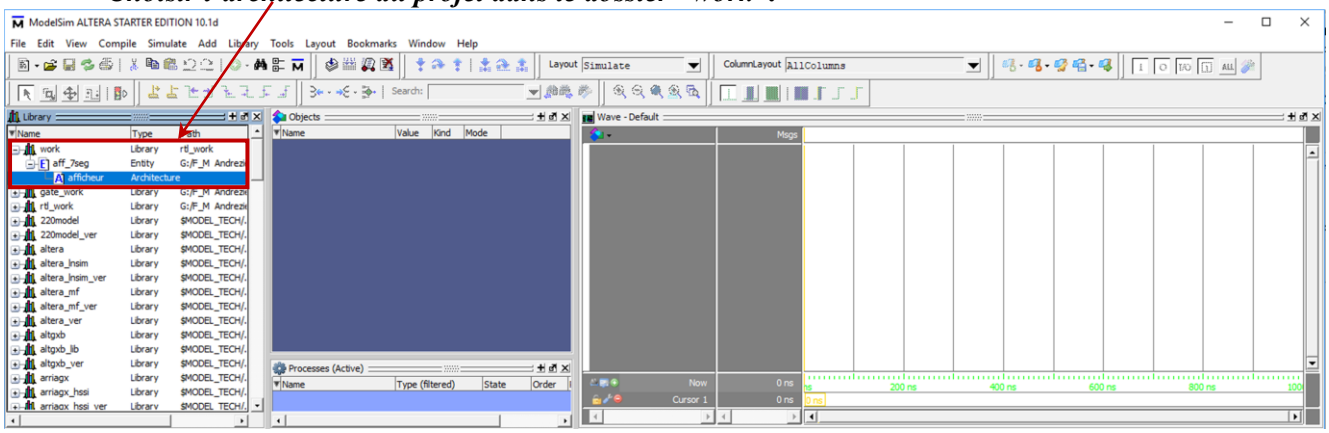
SIMULATION.

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

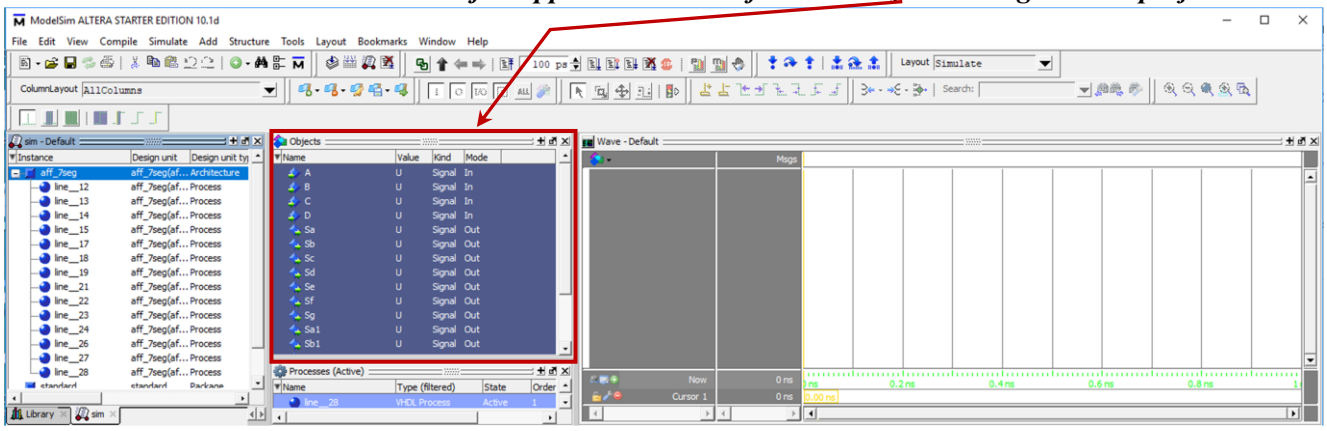




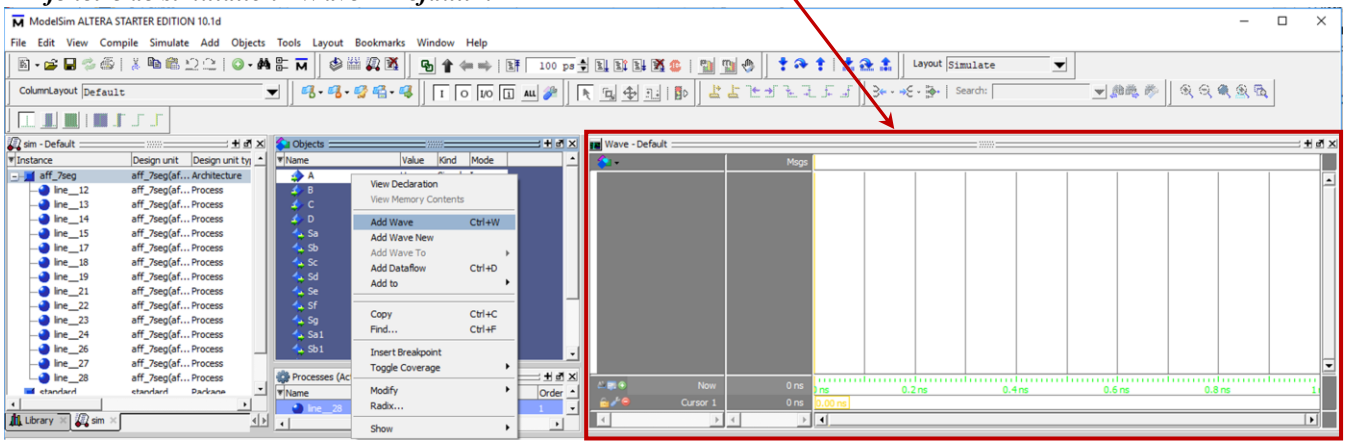
☞ ***Choisir l'architecture du projet dans le dossier "work".***



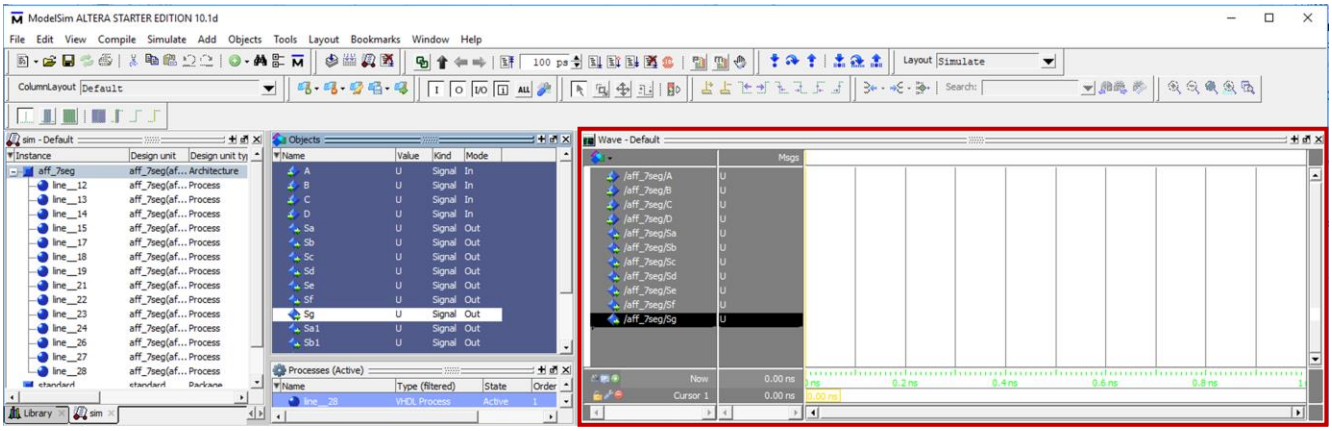
… Un double clic sur "Architecture" fait apparaître dans la fenêtre "Objets" les signaux du projet.



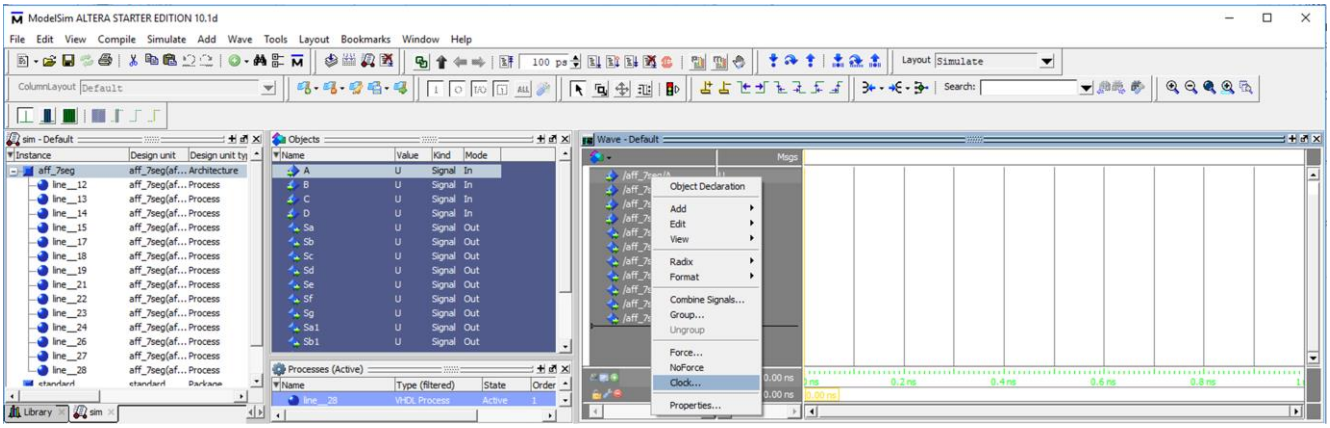
… Faire un clic droit sur chacun des signaux et choisir "Add Wave". Les signaux sont ainsi transférés dans la fenêtre de simulation "Wave – Default".



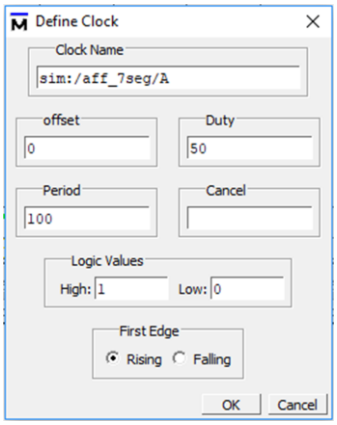
➢ **Paramétrage des signaux "A", "B", "C" et "D".**

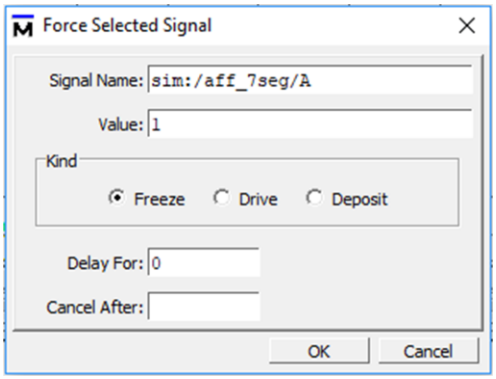


☞ ***Faire un clic droit sur un signal et choisir l'une des options "Force…", "NoForce" ou "Clock…".***

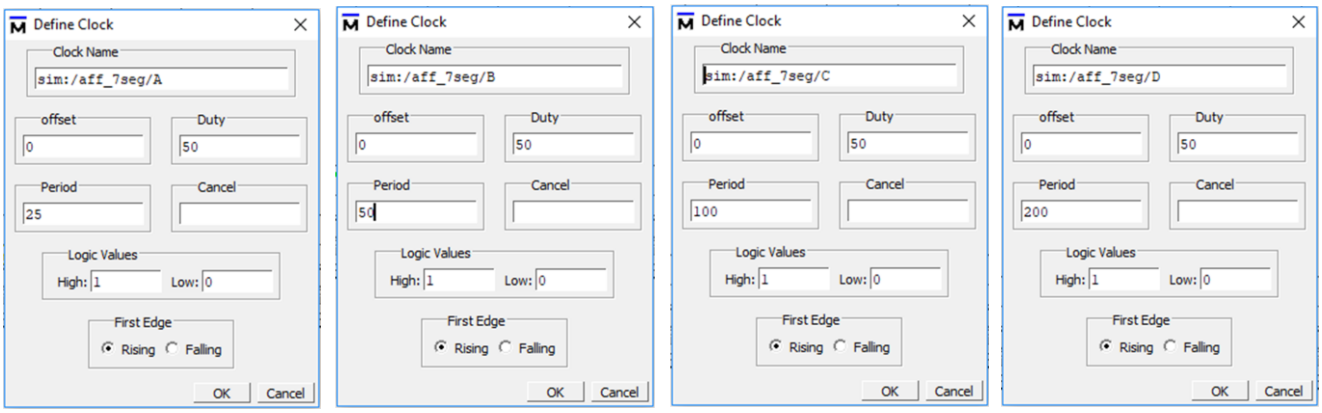


➢ Chacune des options **"Clock…"** et **"Force…"** sera paramétrée en fonction du résultat souhaité de la simulation.

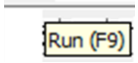


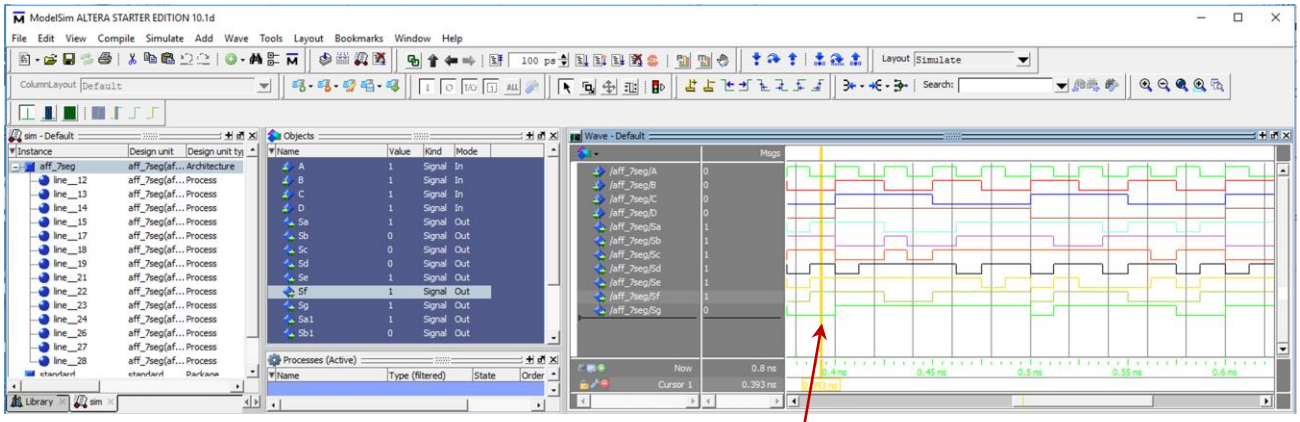


☞ Pour les signaux **A**, **B**, **C** et **D** du décodeur BCD-7Segments, on choisit l'option **"Clock"** avec le paramétrage suivant :

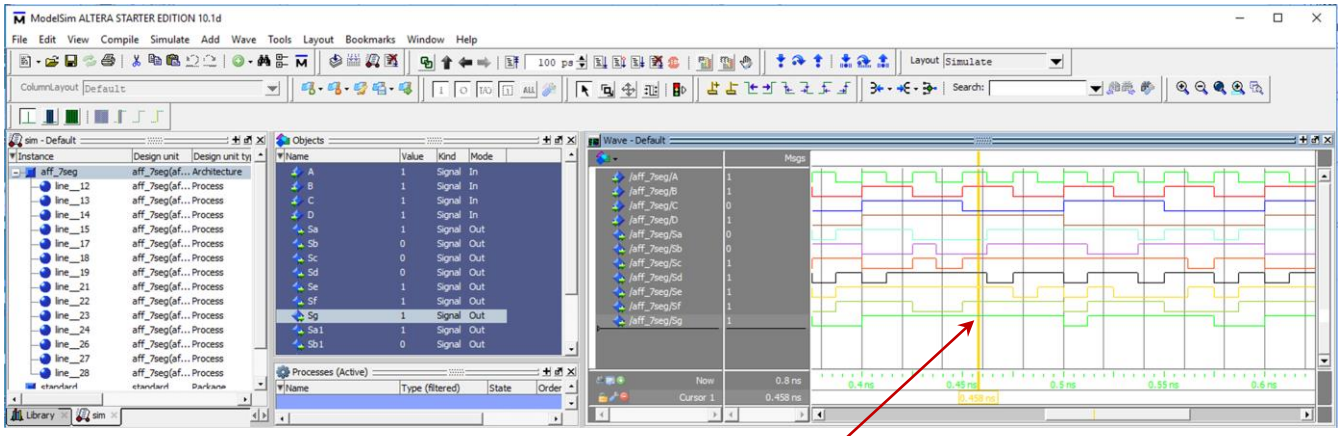


Lancer la simulation en cliquant successivement sur l'icône "Run". 





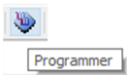
Affichage du "0" : A = 0 ; B = 0 ; C = 0 et D = 0. Tous les segments sont allumés sauf le "g".



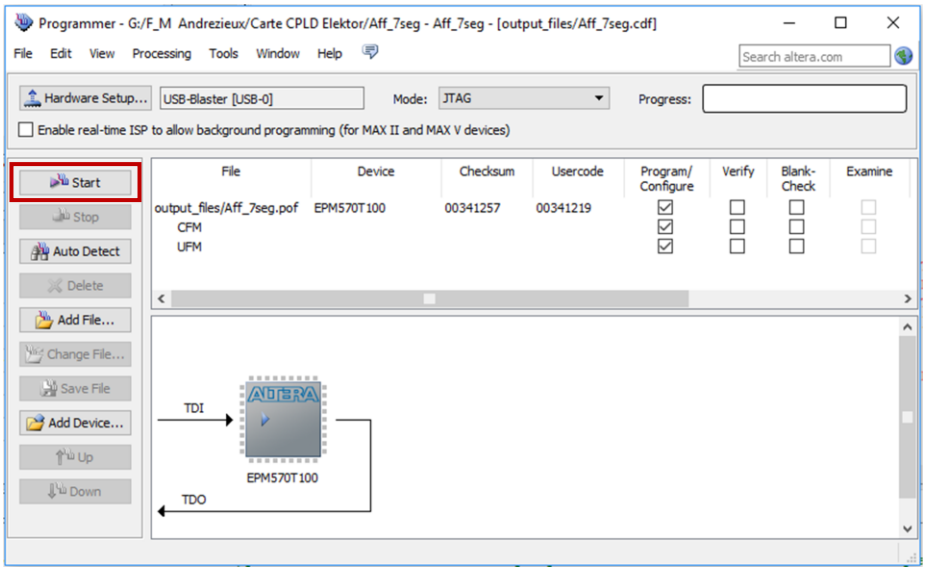
Affichage du "b" : A = 1 ; B = 1 ; C = 0 et D = 1 (0xb soit 11 en valeur décimale).

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

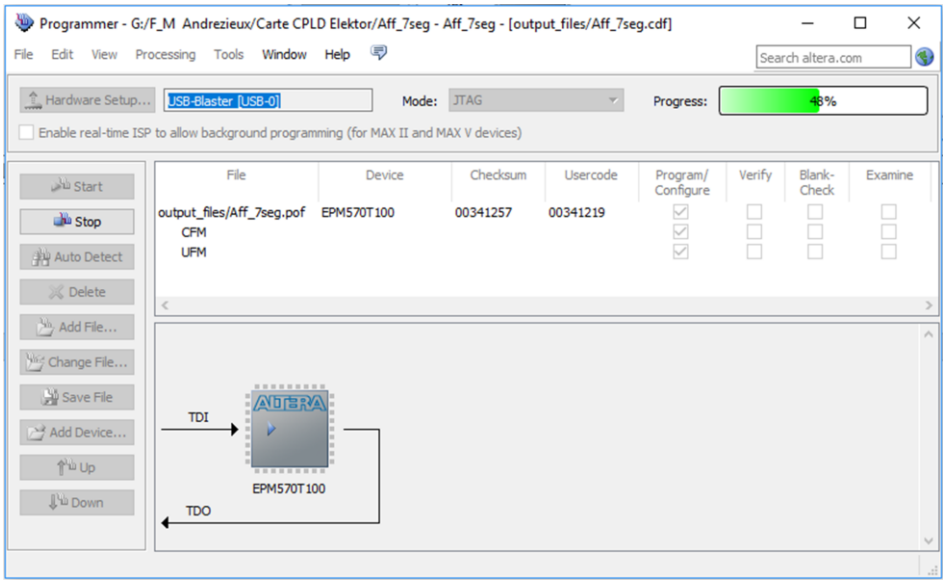
PROGRAMMATION DE LA PUCE CPLD : EPM570T100C5.

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

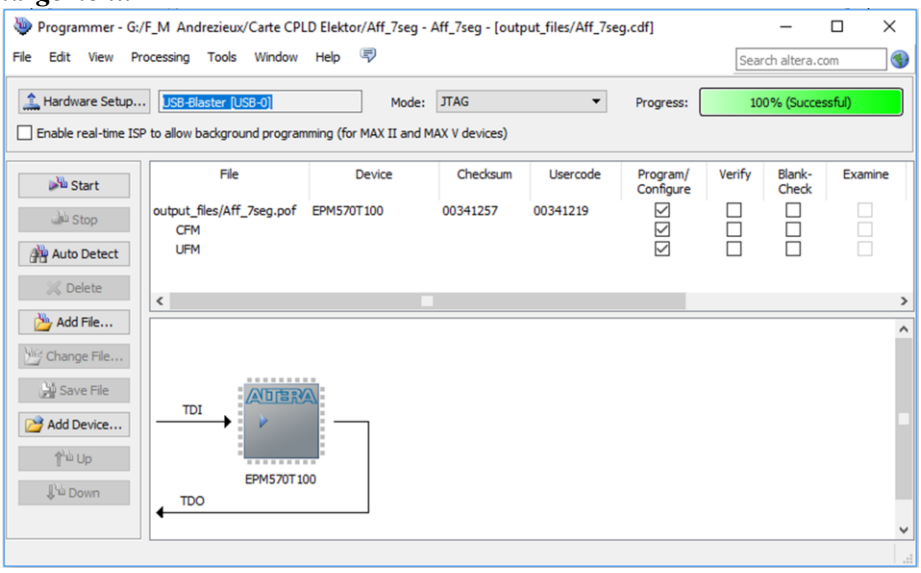
Cliquer ensuite sur **"Start"** pour lancer la programmation de la puce.



… Téléchargement



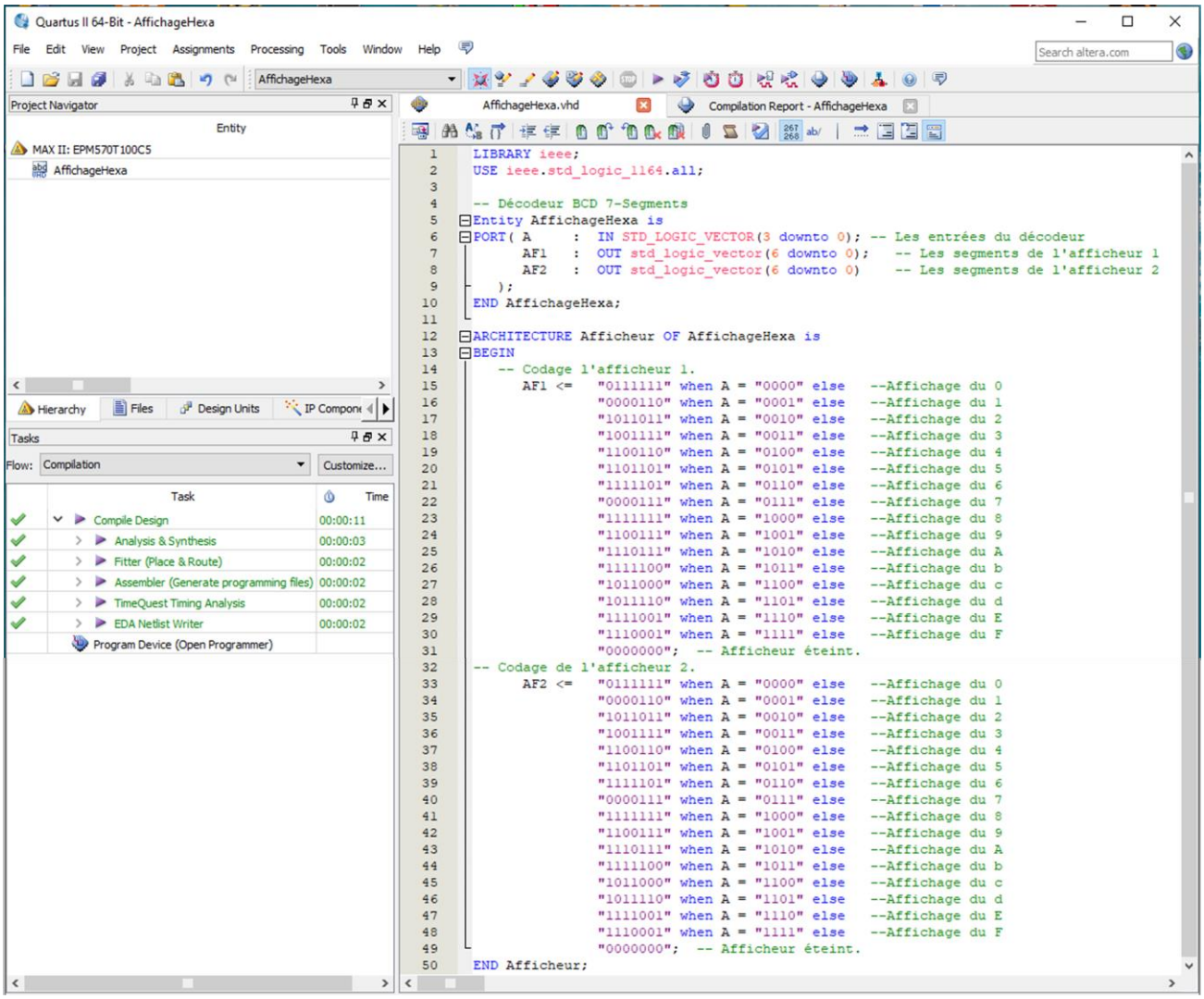
Fin du téléchargement.



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

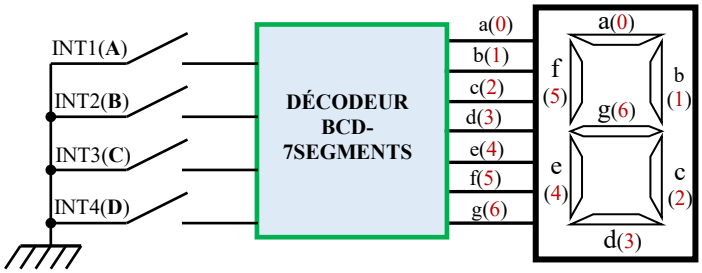
  La syntaxe de l'instruction **"when…else"** est la suivante :

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

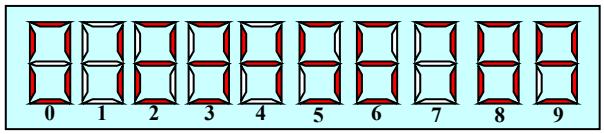



```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  -- Décodeur BCD 7-Segments
5  Entity AffichageHexa is
6  PORT( A     :  IN STD_LOGIC_VECTOR(3 downto 0); -- Les entrées du décodeur
7        AF1   :  OUT std_logic_vector(6 downto 0);   -- Les segments de l'afficheur 1
8        AF2   :  OUT std_logic_vector(6 downto 0)    -- Les segments de l'afficheur 2
9    );
10 END AffichageHexa;
11
12 ARCHITECTURE Afficheur OF AffichageHexa is
13 BEGIN
14    -- Codage l'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                "1110111" when A = "1010" else   --Affichage du A
26                "1111100" when A = "1011" else   --Affichage du b
27                "1011000" when A = "1100" else   --Affichage du c
28                "1011110" when A = "1101" else   --Affichage du d
29                "1111001" when A = "1110" else   --Affichage du E
30                "1110001" when A = "1111" else   --Affichage du F
31                "0000000";  -- Afficheur éteint.
32 -- Codage de l'afficheur 2.
33       AF2 <=   "0111111" when A = "0000" else   --Affichage du 0
34                "0000110" when A = "0001" else   --Affichage du 1
35                "1011011" when A = "0010" else   --Affichage du 2
36                "1001111" when A = "0011" else   --Affichage du 3
37                "1100110" when A = "0100" else   --Affichage du 4
38                "1101101" when A = "0101" else   --Affichage du 5
39                "1111101" when A = "0110" else   --Affichage du 6
40                "0000111" when A = "0111" else   --Affichage du 7
41                "1111111" when A = "1000" else   --Affichage du 8
42                "1100111" when A = "1001" else   --Affichage du 9
43                "1110111" when A = "1010" else   --Affichage du A
44                "1111100" when A = "1011" else   --Affichage du b
45                "1011000" when A = "1100" else   --Affichage du c
46                "1011110" when A = "1101" else   --Affichage du d
47                "1111001" when A = "1110" else   --Affichage du E
48                "1110001" when A = "1111" else   --Affichage du F
49                "0000000";  -- Afficheur éteint.
50 END Afficheur;
```

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



Affichage des 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 :

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

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

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

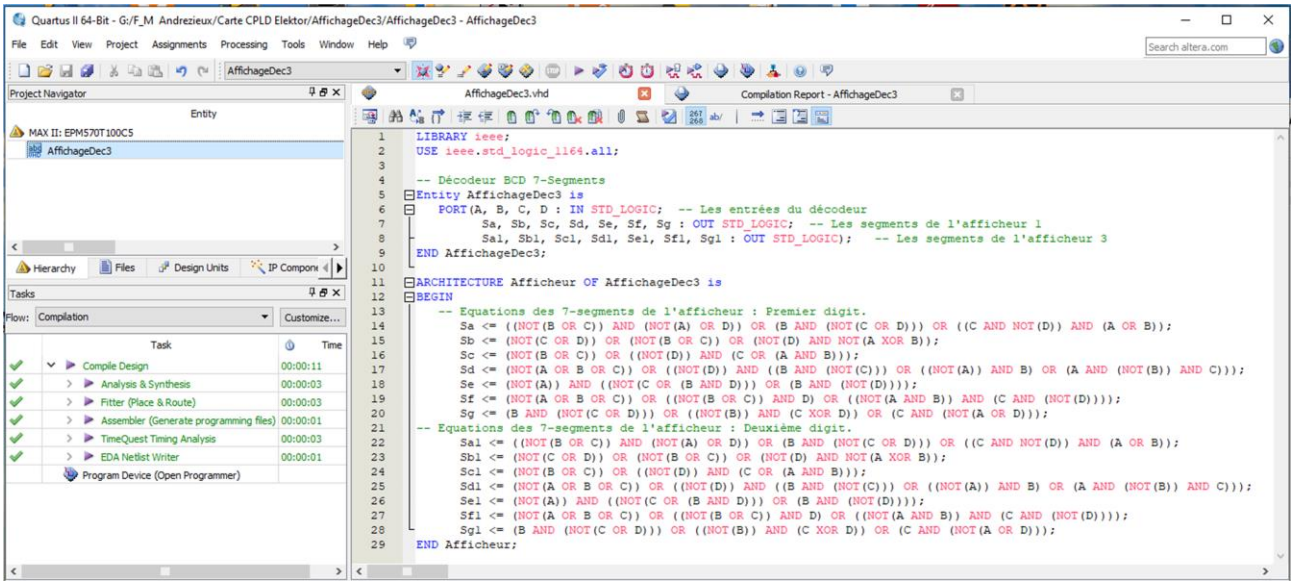
$$Sd = \overline{(A+B+C)} + \overline{D} \cdot (B \cdot \overline{C} + \overline{A} \cdot B + A \cdot \overline{B} \cdot C)$$

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

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

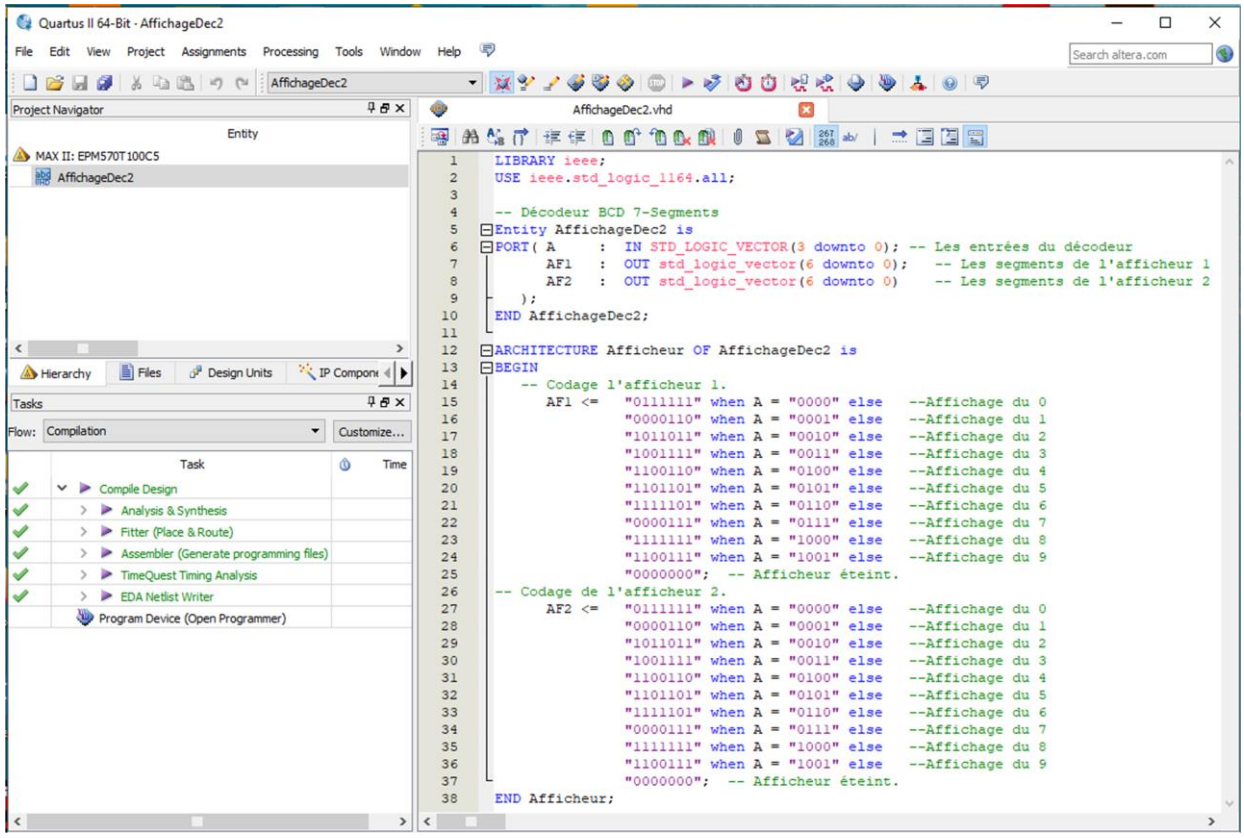
$$Sg = B \cdot \overline{(C+D)} + \overline{B} \cdot (C \oplus D) + C \cdot \overline{(A+D)}$$

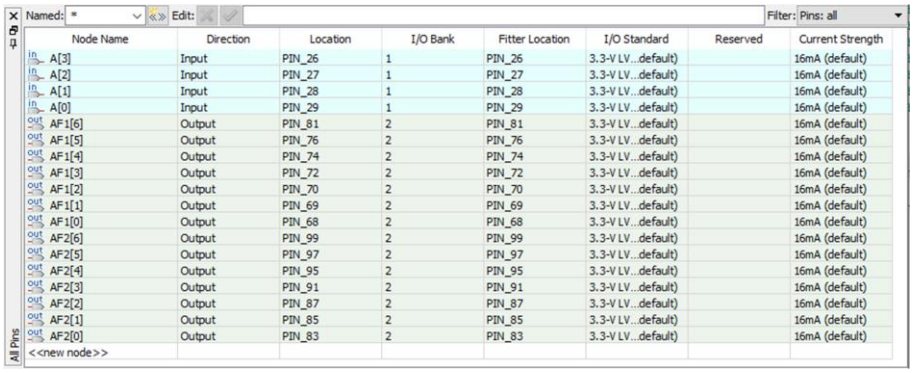
Programme VHDL.



Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
A	Input	PIN_29	1	PIN_29	3.3-V LV...default)		16mA (default)
B	Input	PIN_28	1	PIN_28	3.3-V LV...default)		16mA (default)
C	Input	PIN_27	1	PIN_27	3.3-V LV...default)		16mA (default)
D	Input	PIN_26	1	PIN_26	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)	As Signa...e output	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)
<<new node>>							

Programme VHDL du décodeur utilisant l'instruction "when…else"



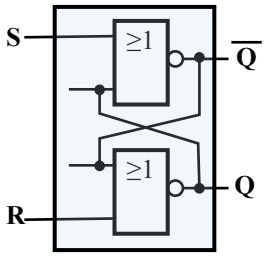


1- Après avoir programmer le circuit **FPGA** et vérifier le bon fonctionnement du décodeur BCD 7-Segments, **faire valider le fonctionnement par le professeur.**

2- Rédiger un compte rendu du TP précisant la programmation d'un circuit "Décodeur BCD-7Segments".

3- Conclure.

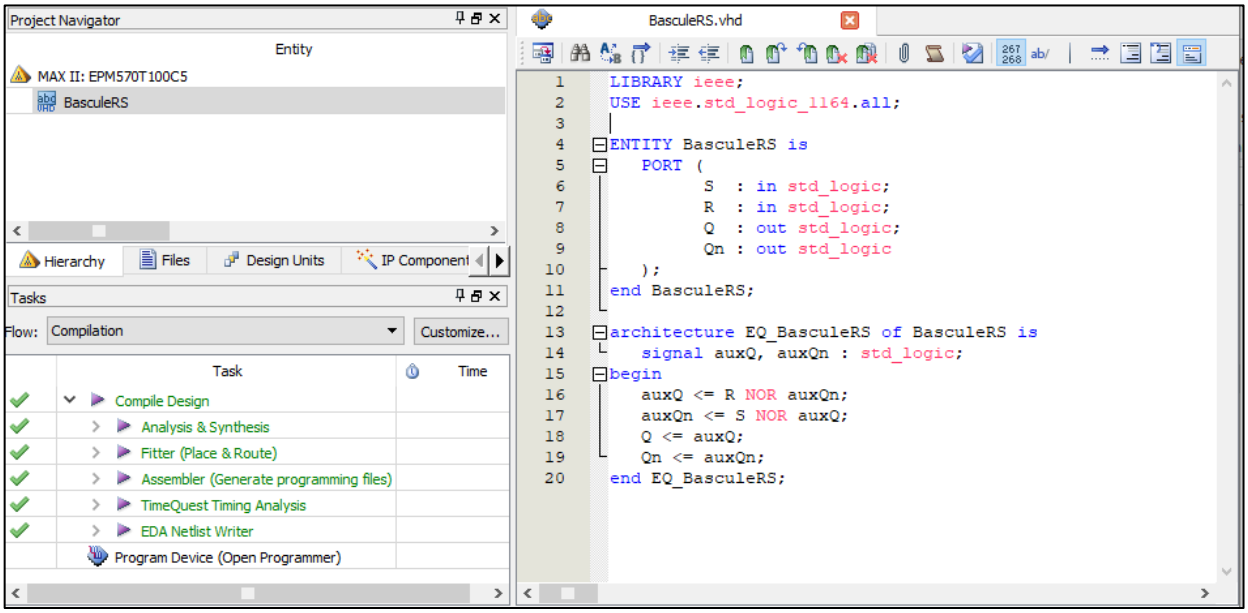
1- LA BASCULE R-S : Bascule asynchrone.

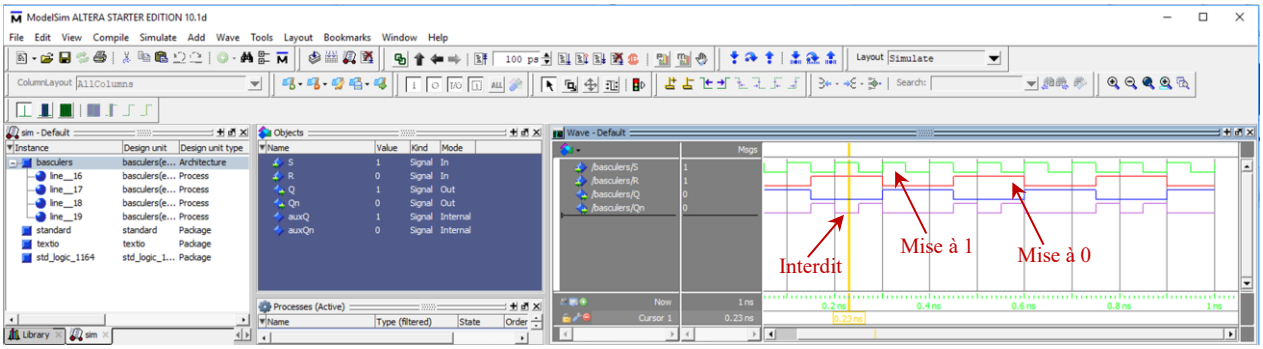


- Équations des sorties Q et $\overline{Q}$ en fonction des entrées S et R.
  - $Q = \overline{R + \overline{Q}}$ (1)
  - $\overline{Q} = \overline{S + Q}$ (2)
- (2) dans (1) : $Q = \overline{R + \overline{S + Q}} = \overline{R} \cdot (S + Q)$
- (1) dans (2) : $\overline{Q} = \overline{S + \overline{R + \overline{Q}}} = \overline{S} \cdot (R + \overline{Q})$

Table de vérité.

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

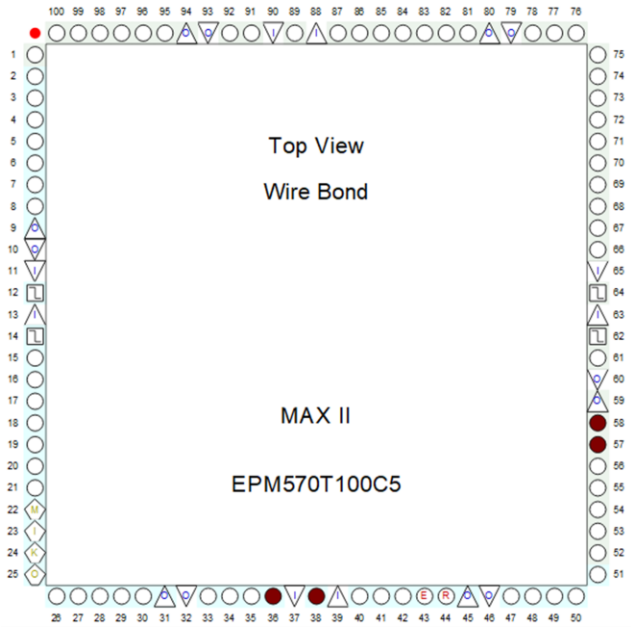




Affectation des broches.

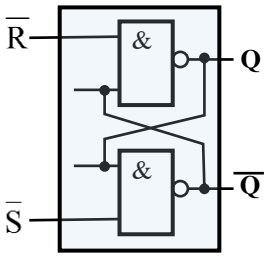
Les entrées R (reset) et S (set) de la bascule.			
Broche N°38	IOB1-38	Lecture état du bouton poussoir **BP1**	R (Reset)
Broche N°36	IOB1-36	Lecture état du bouton poussoir **BP2**	S (Set)

Les sorties Q et Qn de la bascule.			
Broche N°58	IOB2-58	Commande diode led **L1**	Sortie Q
Broche N°57	IOB2-57	Commande diode led **L2**	Sortie inversée Qn



Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
Q	Output	PIN_58	2	PIN_58	3.3-V LV...default)		16mA (default)
Qn	Output	PIN_57	2	PIN_57	3.3-V LV...default)		16mA (default)
R	Input	PIN_38	1	PIN_38	3.3-V LV...default)		16mA (default)
S	Input	PIN_36	1	PIN_36	3.3-V LV...default)		16mA (default)
<<new node>>							

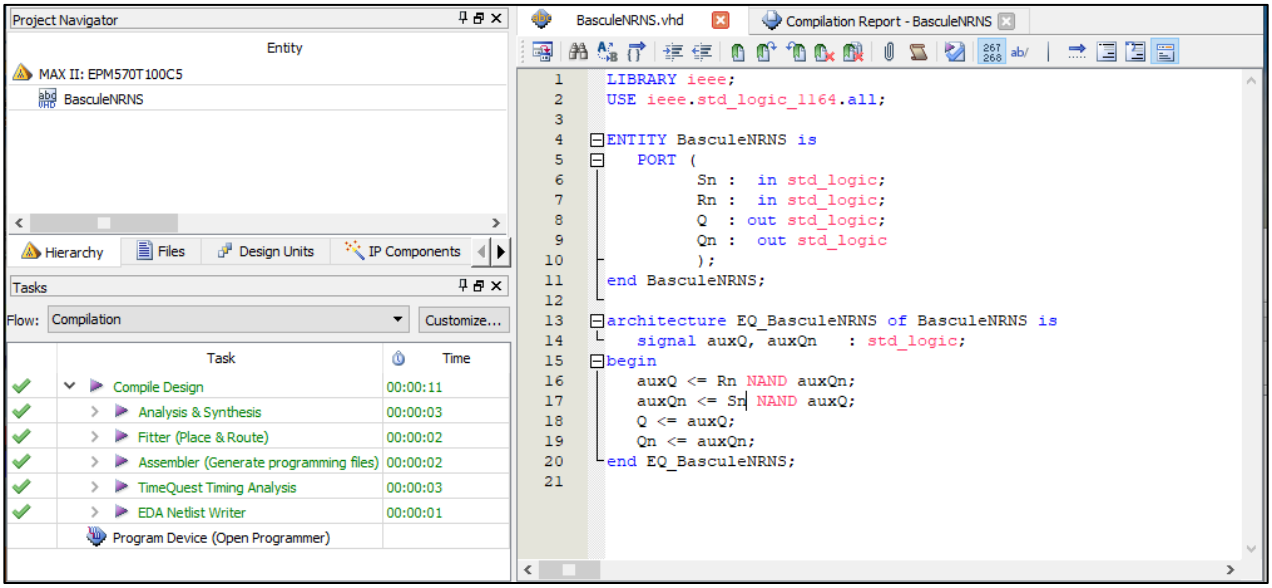
2- LA BASCULE $\overline{R}-\overline{S}$: Bascule asynchrone.



- Équations des sorties Q et $\overline{Q}$ en fonction des entrées S et R.
  - $Q = \overline{\overline{R} \cdot \overline{Q}}$ (1)
  - $\overline{Q} = \overline{\overline{S} \cdot Q}$ (2)

Table de vérité.

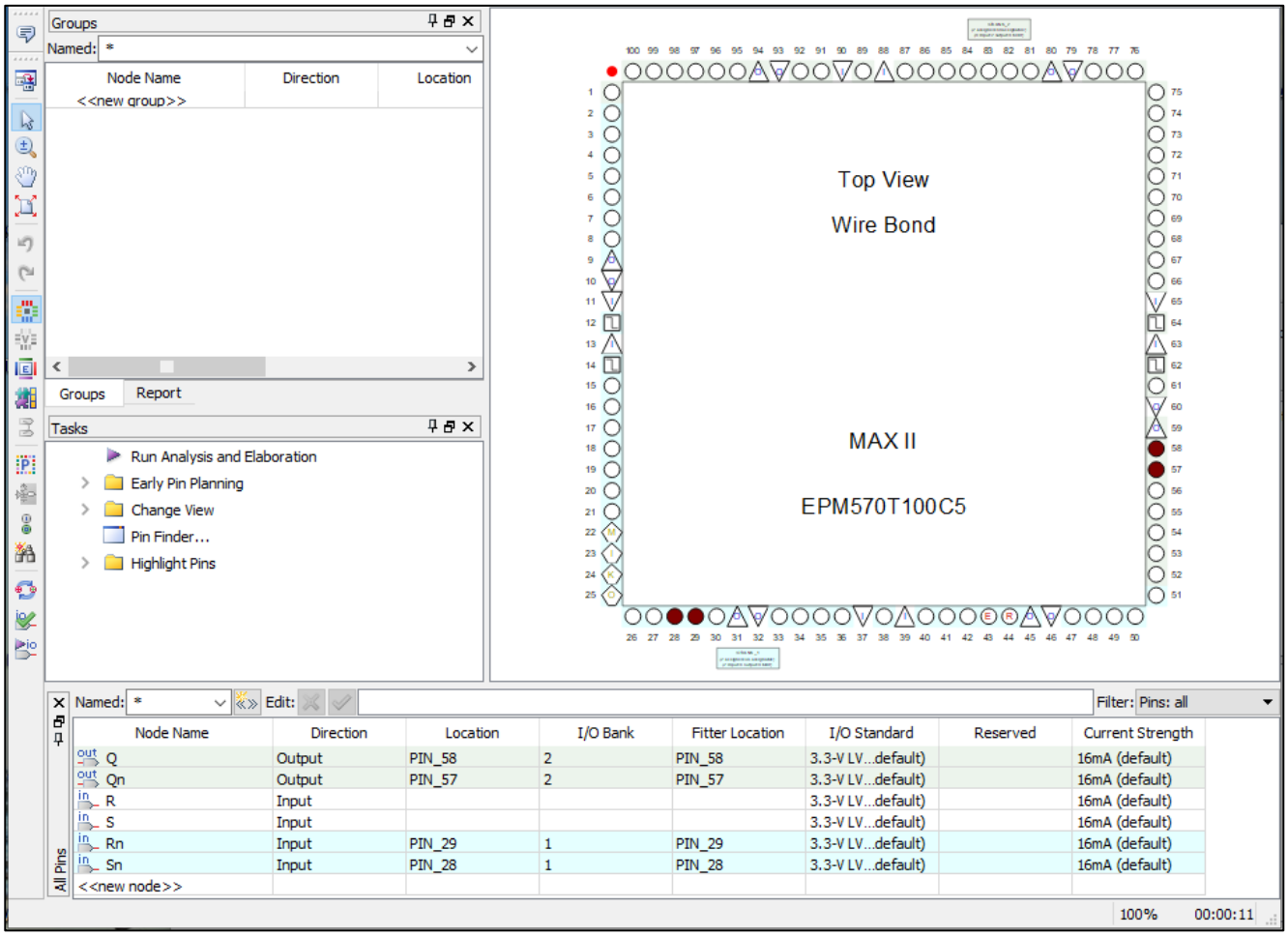
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	Fonction
0	0	1	1	Interdit
0	1	0	1	Mise à 1
1	0	1	0	Mise à 0
1	1	Q	$\overline{Q}$	Mémoire



Affectation des broches.

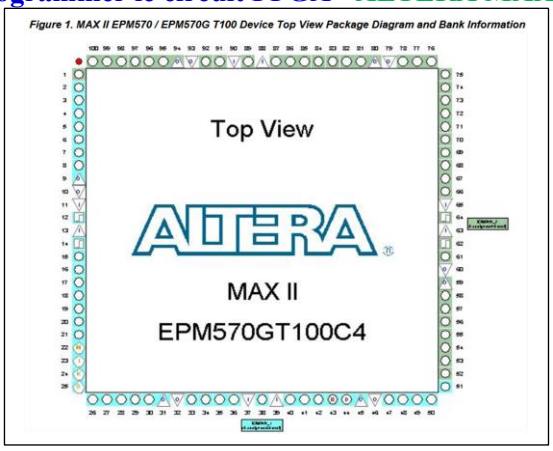
Les entrées R (reset) et S (set) de la bascule.			
Boche N°29	IOB1-29	Lecture état de l'interrupteur **INT1**	**Rn** (Reset)
Broche N°28	IOB1-28	Lecture état de l'interrupteur **INT2**	**Sn** (Set)

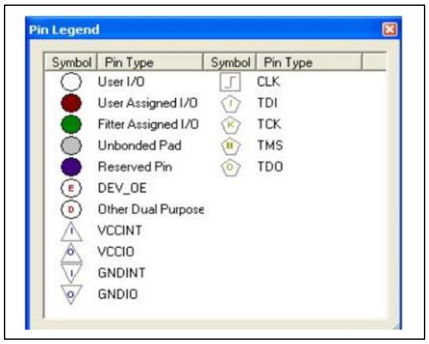
Les sorties Q et Qn de la bascule.			
Broche N°58	IOB2-58	Commande diode led **L1**	Sortie **Q**
Broche N°57	IOB2-57	Commande diode led **L2**	Sortie inversée **Qn**



- **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 échéancier 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.

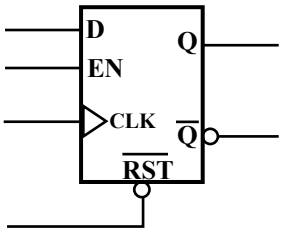
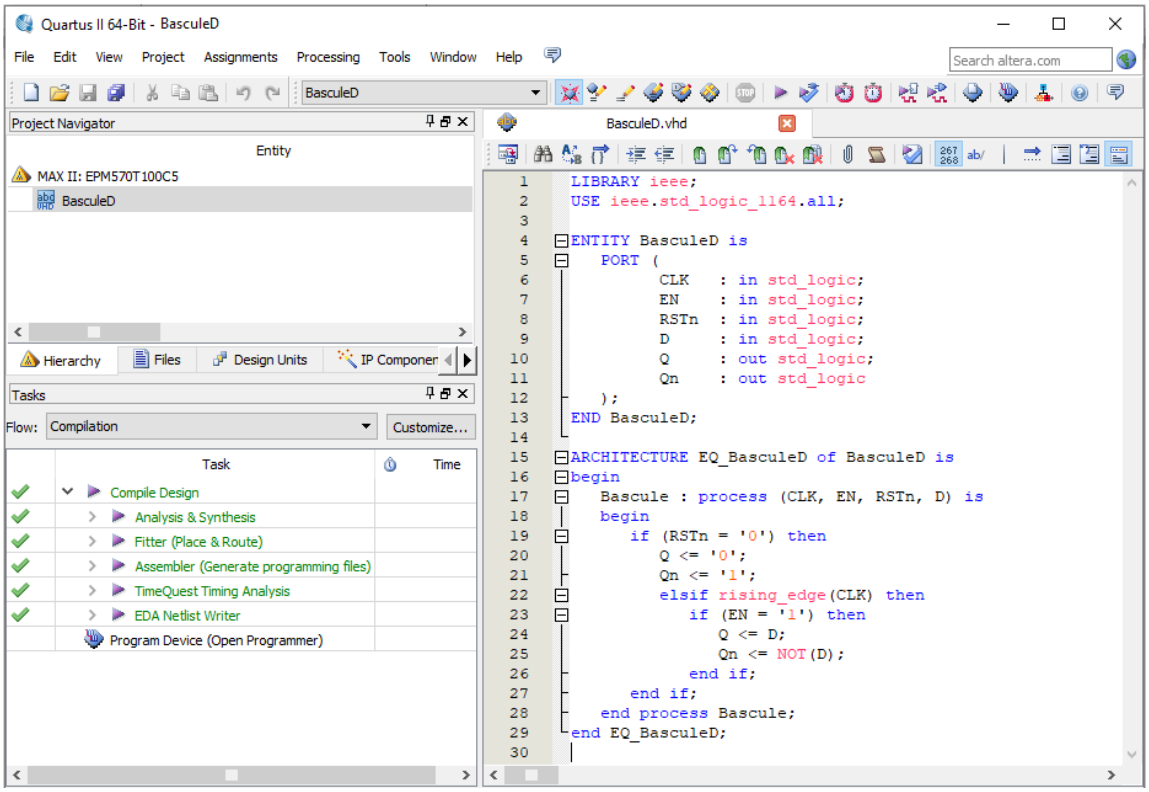


Table de vérité					
CLK	EN	$\overline{RST}$	D	Q	$\overline{Q}$
X	0	0	X	0	1
X	1	0	X	0	1
X	0	1	X	Q_n	$\overline{Q_n}$
↑	1	1	0	0	1
↑	1	1	1	1	0

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

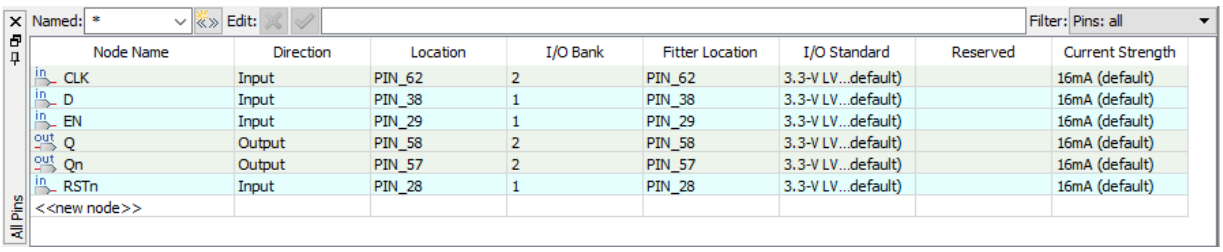
✍ Programme VHDL.



➢ Affectation des broches.

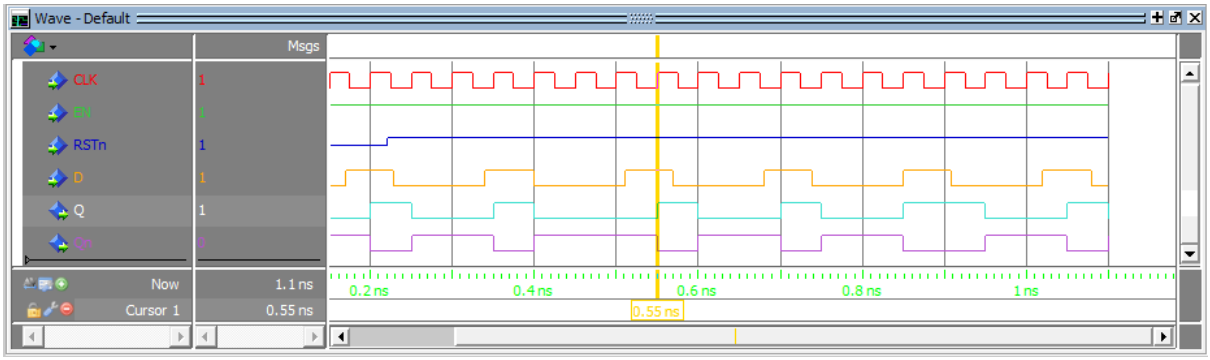
Les sorties de la bascule D			
Broche N°58	IOB2-58	Commande diode led **L1**	Sortie Q
Broche N°57	IOB2-57	Commande diode led **L2**	Sortie inverse Qn

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



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

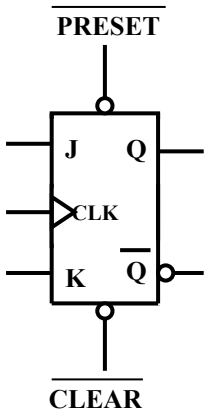
Résultat de la simulation.



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.

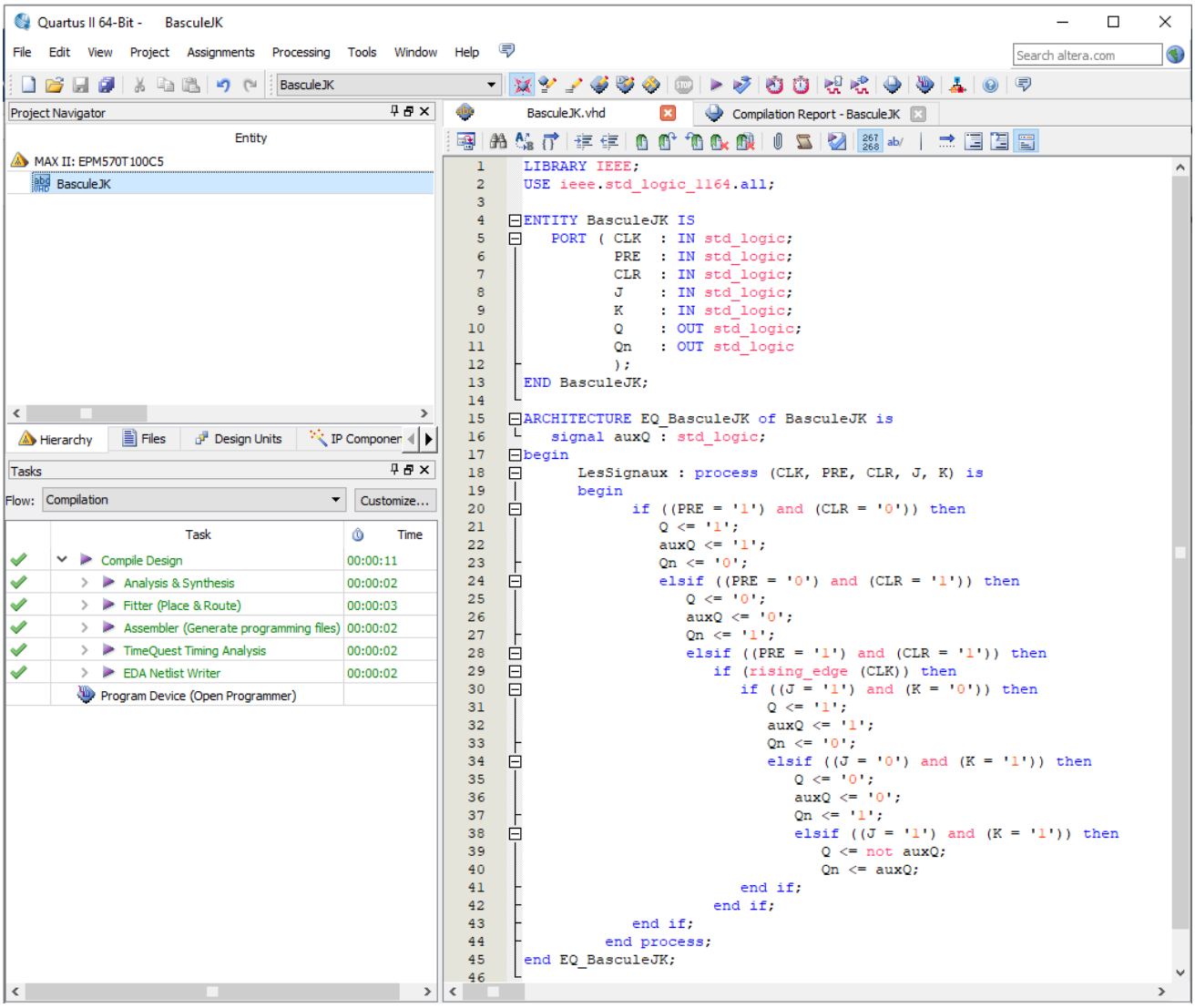
➢ Symbole et table de vérité.



CLK	J	K	$\overline{PRESET}$	$\overline{CLEAR}$	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 $\overline{PRESET}$ active à l'état bas pour un forçage à "1" et une entrée $\overline{CLEAR}$ active à l'état bas pour un forçage à "0".

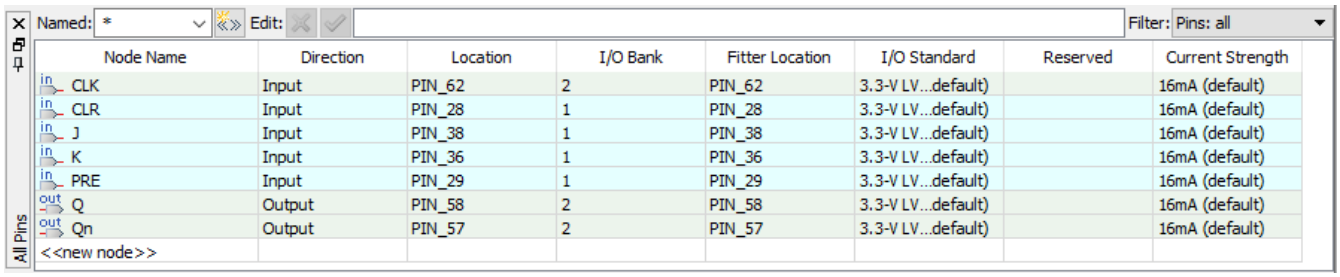
☺ **Le programme VHDL.**



☞ **Affectation des broches.**

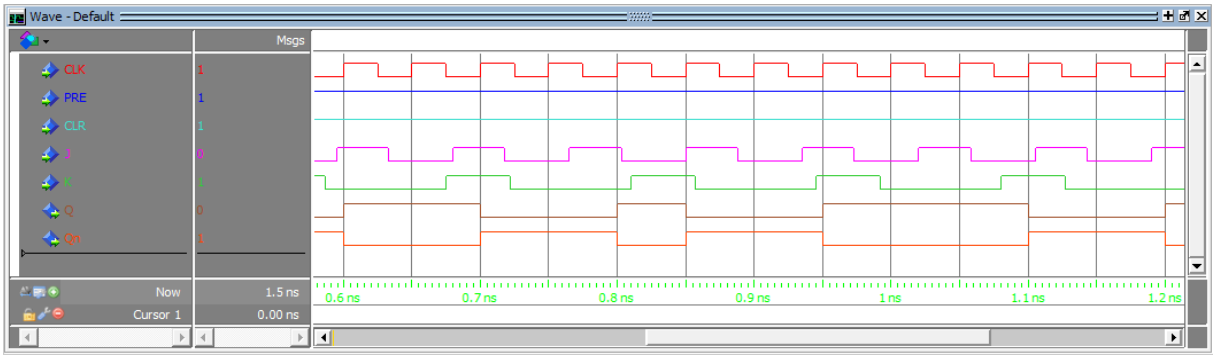
Les sorties de la bascule J-K			
Broche N°58	IOB2-58	Commande diode led **L1**	Sortie Q
Broche N°57	IOB2-57	Commande diode led **L2**	Sortie inverse Qn

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



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

Résultat de la simulation.



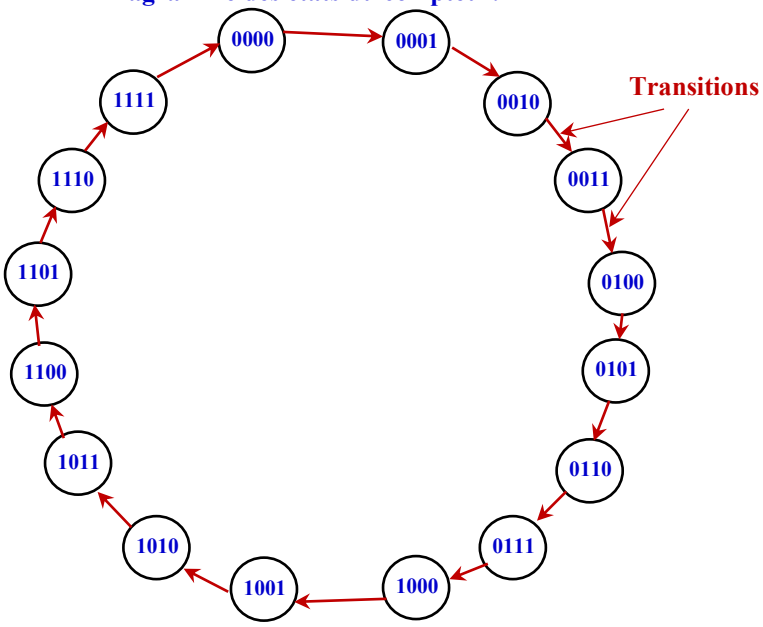
LES COMPTEURS.

1- Compteur synchrone à cycle complet Modulo-16.

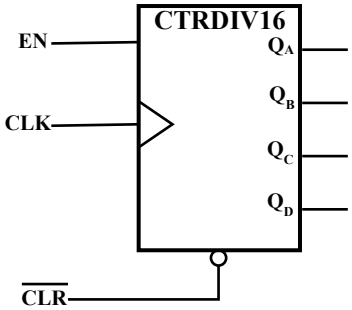
Il s'agit d'un compteur à 16 = 2^4 états distincts. Il utilise 4 bascules synchrones **A**, **B**, **C** et **D** câblés en diviseur de fréquence par 2. Les sorties des 4 bascules sont Q_A, Q_B, Q_C et Q_D.

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.

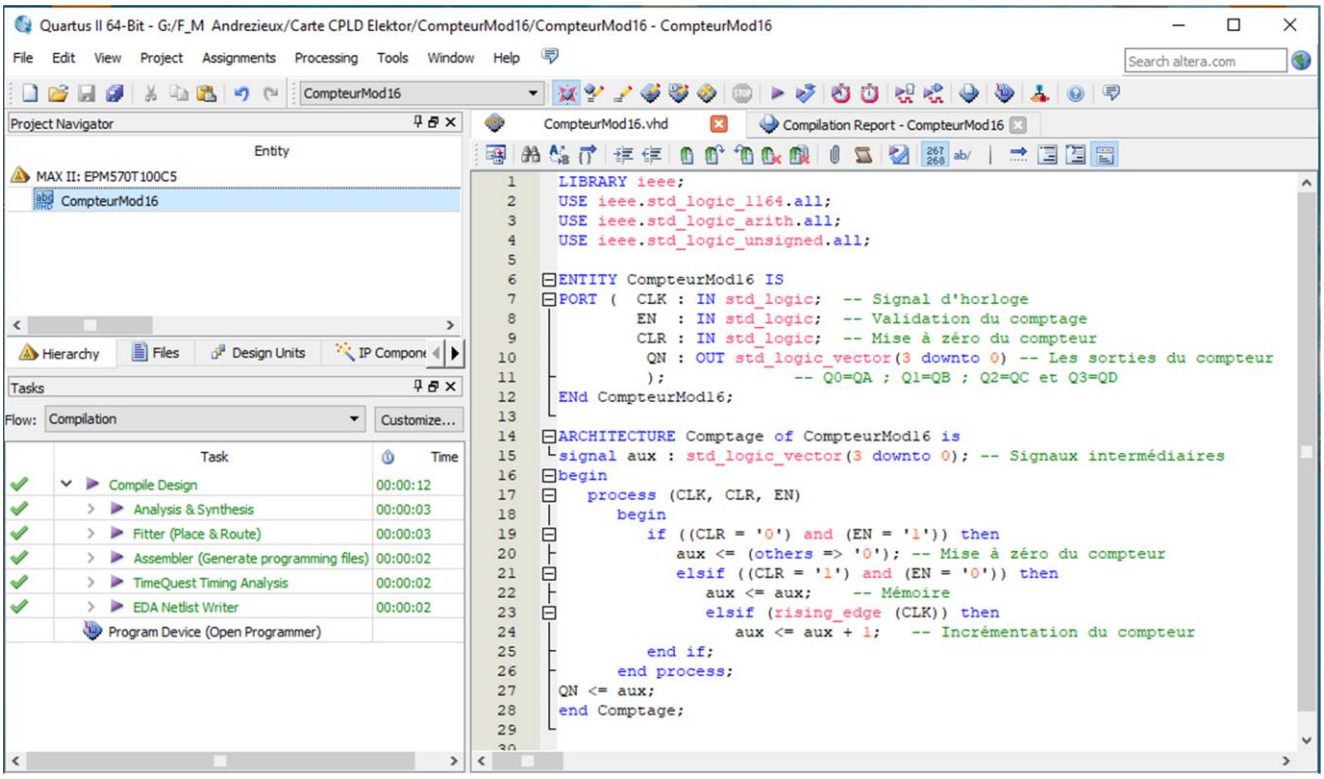


☞ Le passage d'un état au suivant, **appelé transition**, se fait sur une **transition valide** (front montant ou descendant) du signal d'horloge.



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

☺ **Programme VHDL.**

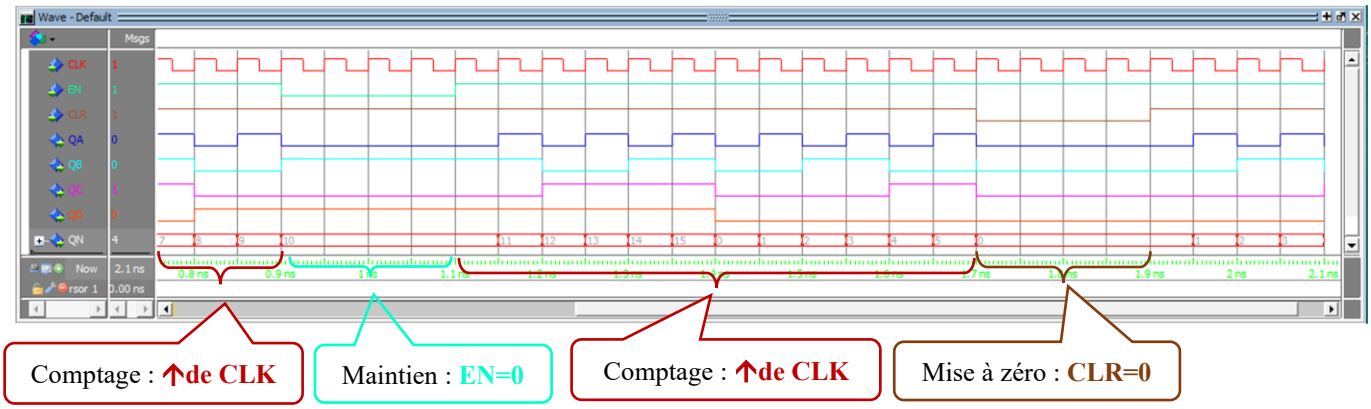


☞ **Affectation des broches.**

Named: * Edit: Filter: Pins: assigned

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
CLR	Input	PIN_28	1	PIN_28	3.3-V LV...default)		16mA (default)
EN	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)
<<new node>>							

<u>Résultat de la simulation</u>.

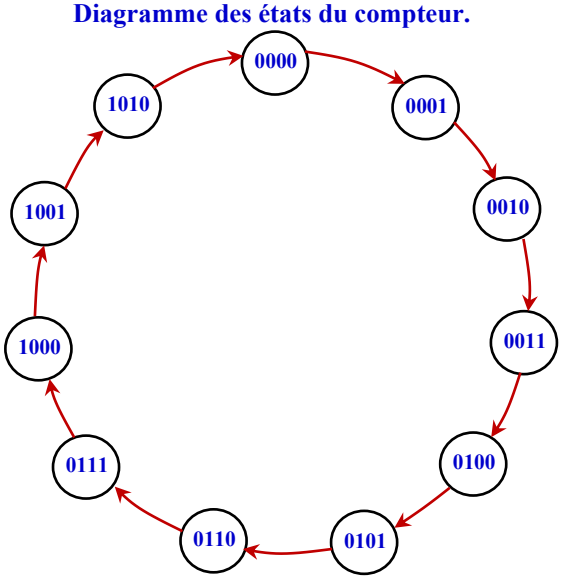


2- Compteur synchrone à cycle incomplet : compteur Modulo 11.

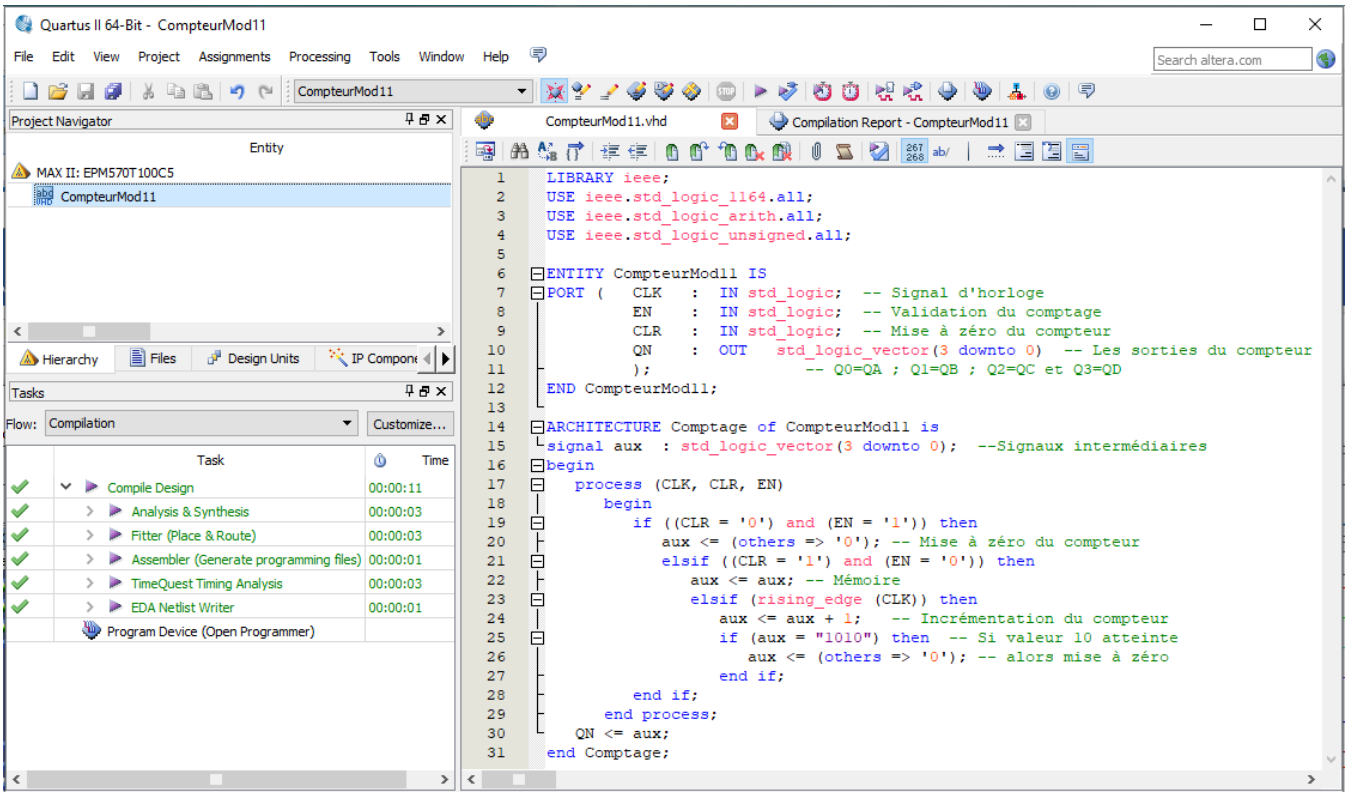
☞ Compteur à 11 états distincts.

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

Diagramme des états du compteur.

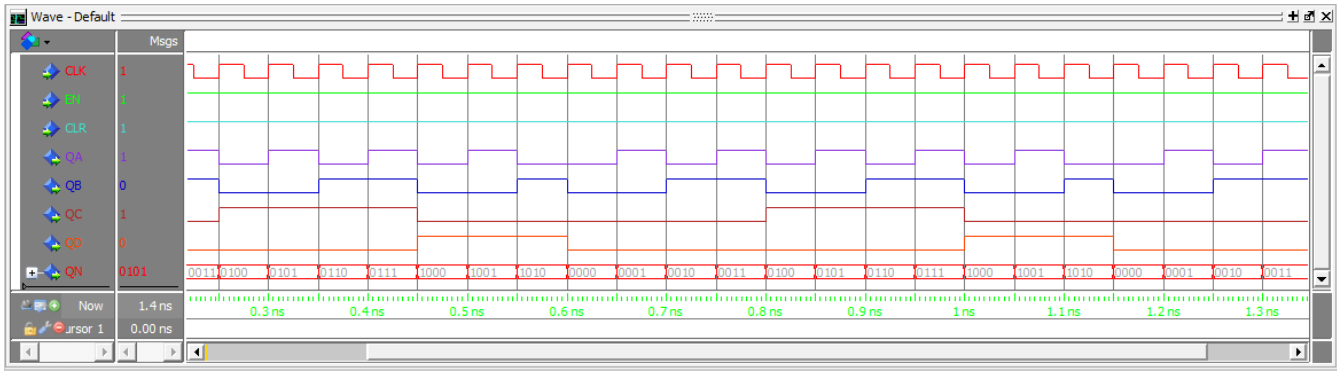


✍ **Programme VHDL.**

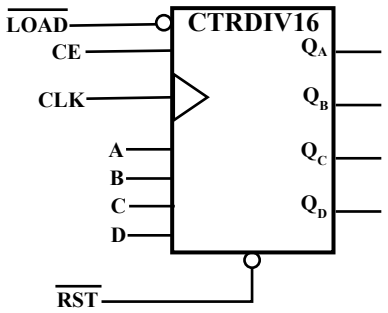


```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

ENTITY CompteurMod11 IS
PORT (   CLK    :  IN std_logic;  -- Signal d'horloge
         EN     :  IN std_logic;  -- Validation du comptage
         CLR    :  IN std_logic;  -- Mise à zéro du compteur
         QN     :  OUT   std_logic_vector(3 downto 0)  -- Les sorties du compteur
         );                      -- Q0=QA ; Q1=QB ; Q2=QC et Q3=QD
END CompteurMod11;

ARCHITECTURE Comptage of CompteurMod11 is
signal aux  : std_logic_vector(3 downto 0);  --Signaux intermédiaires
begin
   process (CLK, CLR, EN)
      begin
         if ((CLR = '0') and (EN = '1')) then
            aux <= (others => '0'); -- Mise à zéro du compteur
         elsif ((CLR = '1') and (EN = '0')) then
            aux <= aux; -- Mémoire
            elsif (rising_edge (CLK)) then
               aux <= aux + 1;   -- Incrémentation du compteur
               if (aux = "1010") then  -- Si valeur 10 atteinte
                  aux <= (others => '0'); -- alors mise à zéro
               end if;
         end if;
      end process;
   QN <= aux;
end Comptage;
```

Résultat de la simulation.

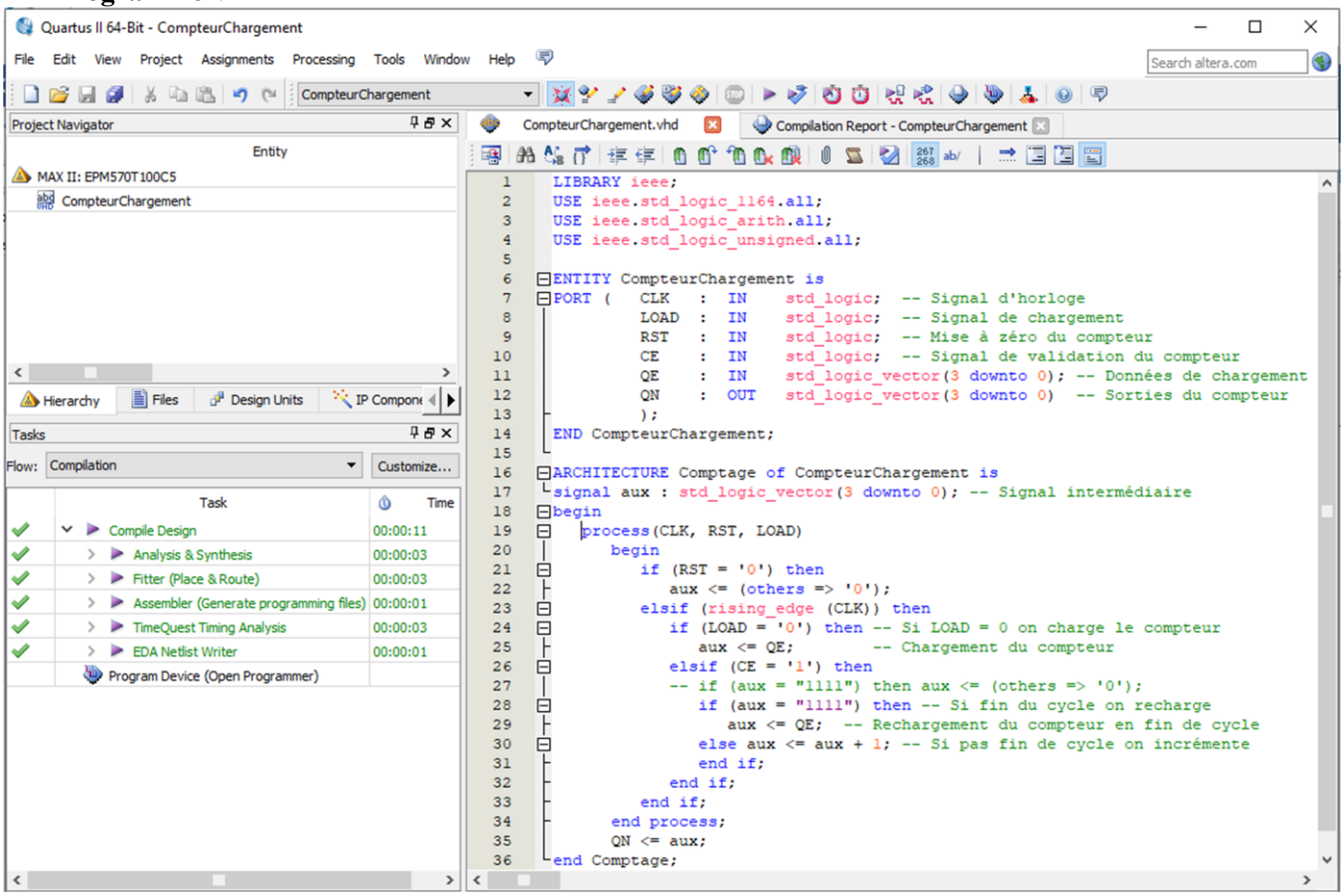


3- Compteur synchrone Modulo-16 avec chargement parallèle.



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

Programme VHDL



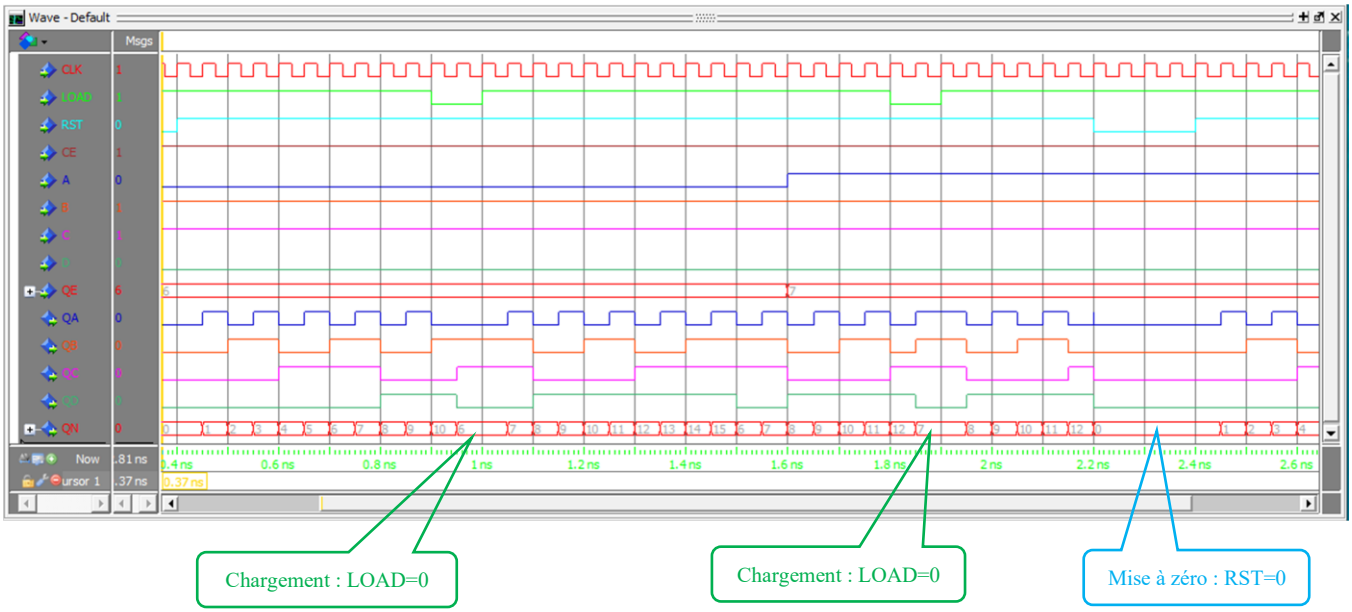
Affectation des broches.

Named: * Edit: Filter: Pins: all

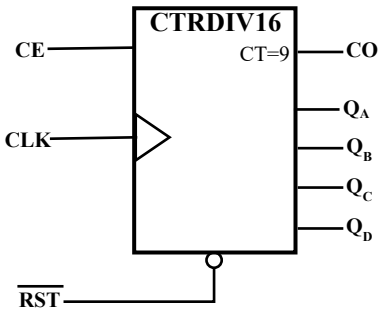
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
CE	Input	PIN_36	1	PIN_36	3.3-V LV...default)		16mA (default)
CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
LOAD	Input	PIN_38	1	PIN_38	3.3-V LV...default)		16mA (default)
QE[3]	Input	PIN_26	1	PIN_26	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>>							

All Pins

Résultat de la simulation.

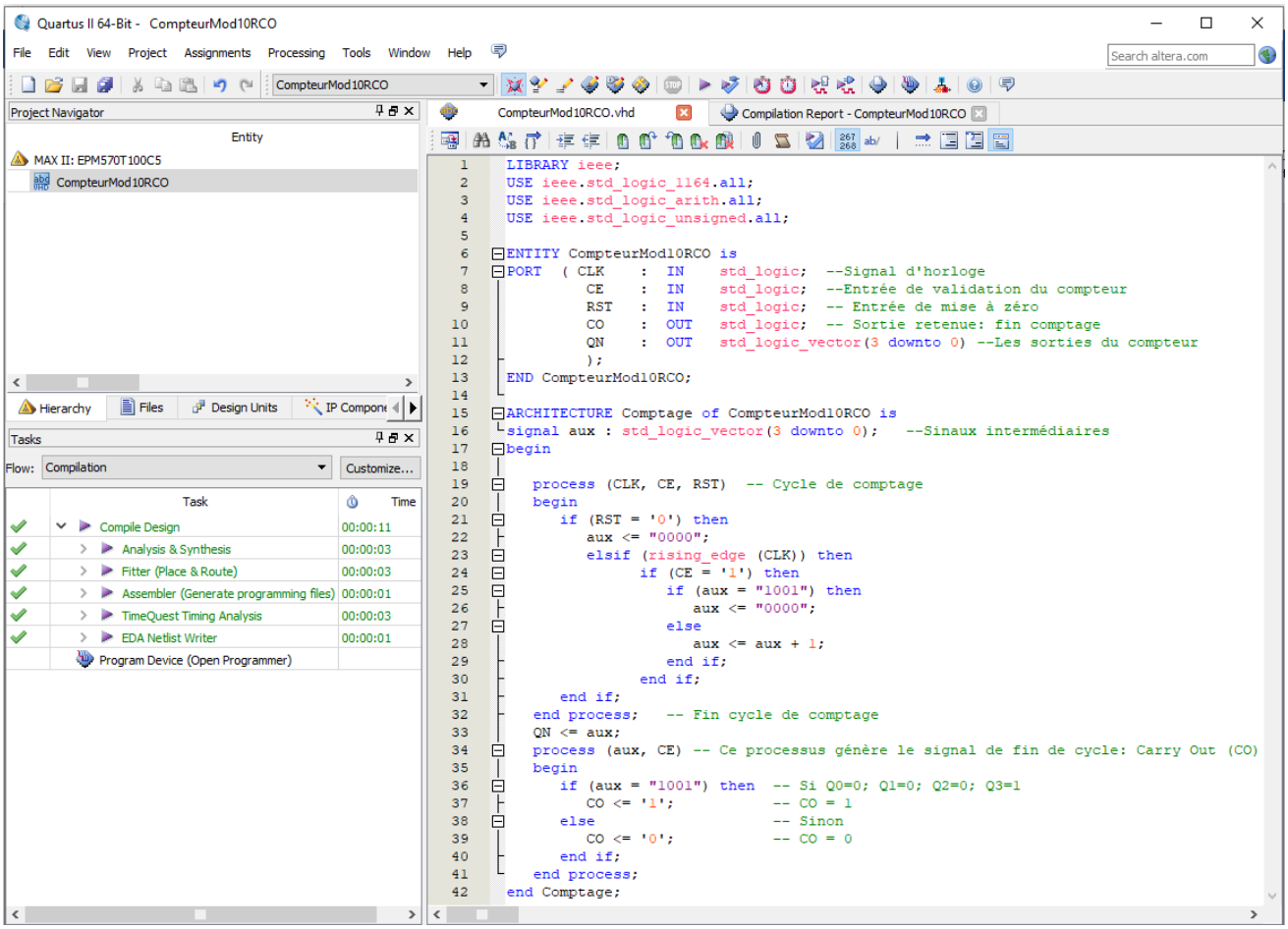


4- COMPTEUR SYNCHRONE À DÉCADE (Modulo-10) : Compteur Binaire Codé Décimal BCD.

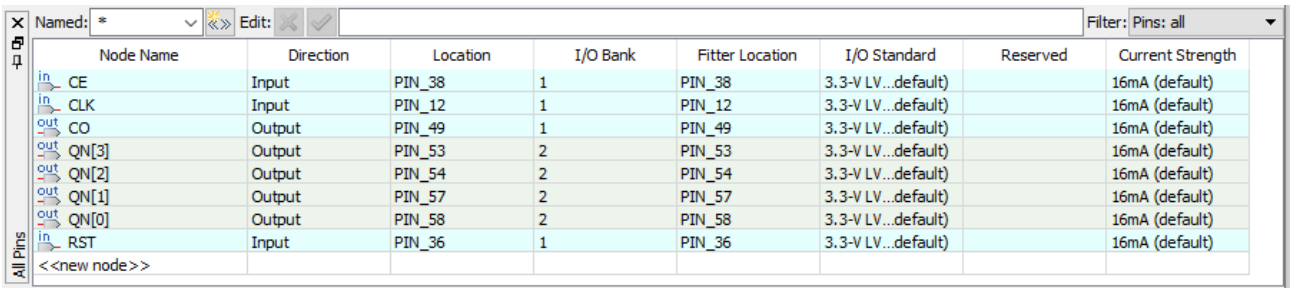


- **CLK :** Signal d'horloge
- **CO** : Sortie retenue = 1 si le compteur est en fin de cycle. C'est-à-dire si $Q_A = 1$; $Q_B = 0$; $Q_C = 0$; $Q_D = 1$.
- **CO** : Sortie retenue = 0 si $Q_A = Q_B = Q_C = Q_D = 0$.
- $\overline{\textbf{RST}} = 0$: Mise à 0 du compteur $Q_A = Q_B = Q_C = Q_D = 0$.

Programme VHDL.

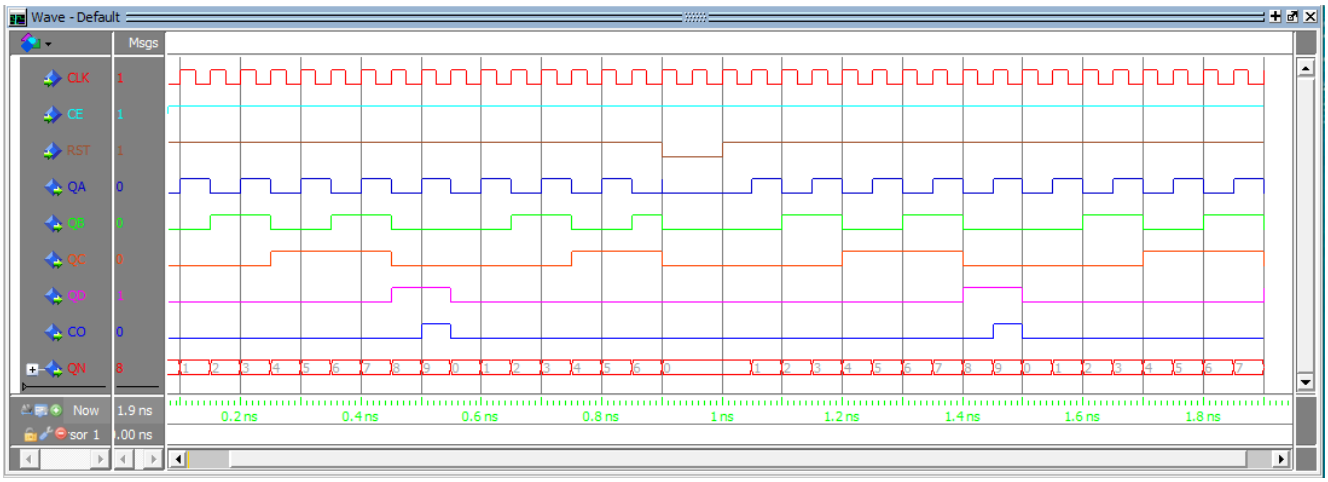


Affectation des broches.



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.

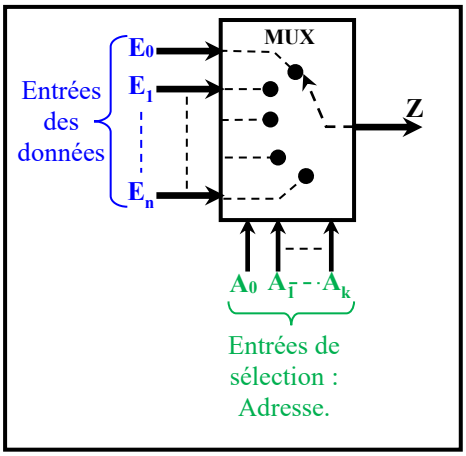


LES MULTIPLEXEURS – Exemples.

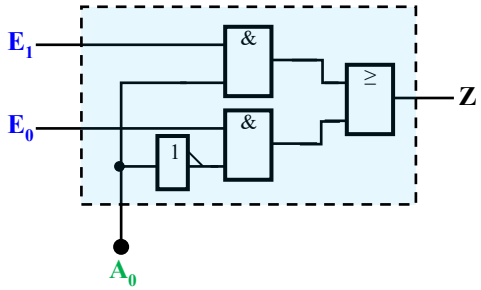
Un multiplexeur (**MUX**) ou sélecteur de donnés est un circuit logique ayant :

- 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 E_0, E_1 ou sélecteur de données 1 parmi 2.



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

Table de vérité

A_0	Z
0	E_0
1	E_1

Autre exemple : Multiplexeur à 4 entrées E_0, E_1, E_2, E_3 ou sélecteur de données 1 parmi 4.

Pour sélectionner une entrée parmi 4, on a besoin de 2 entrées d'adresse A_0 et A_1.

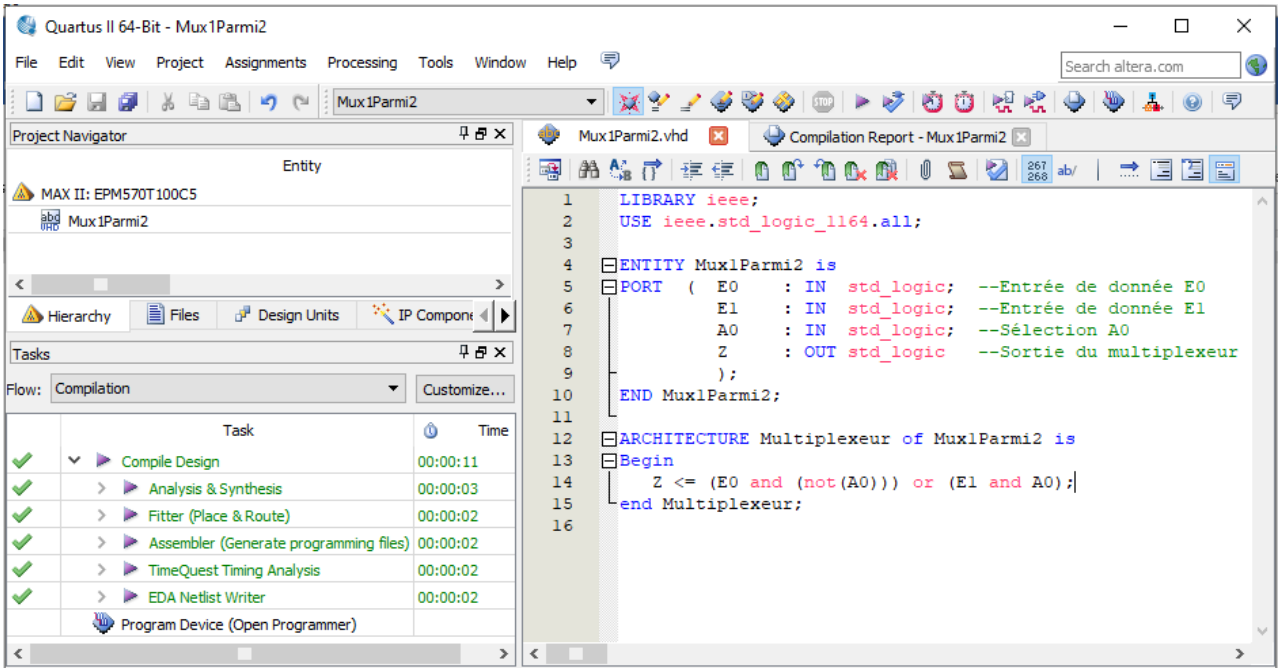
Table de vérité

A_1	A_0	Z
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

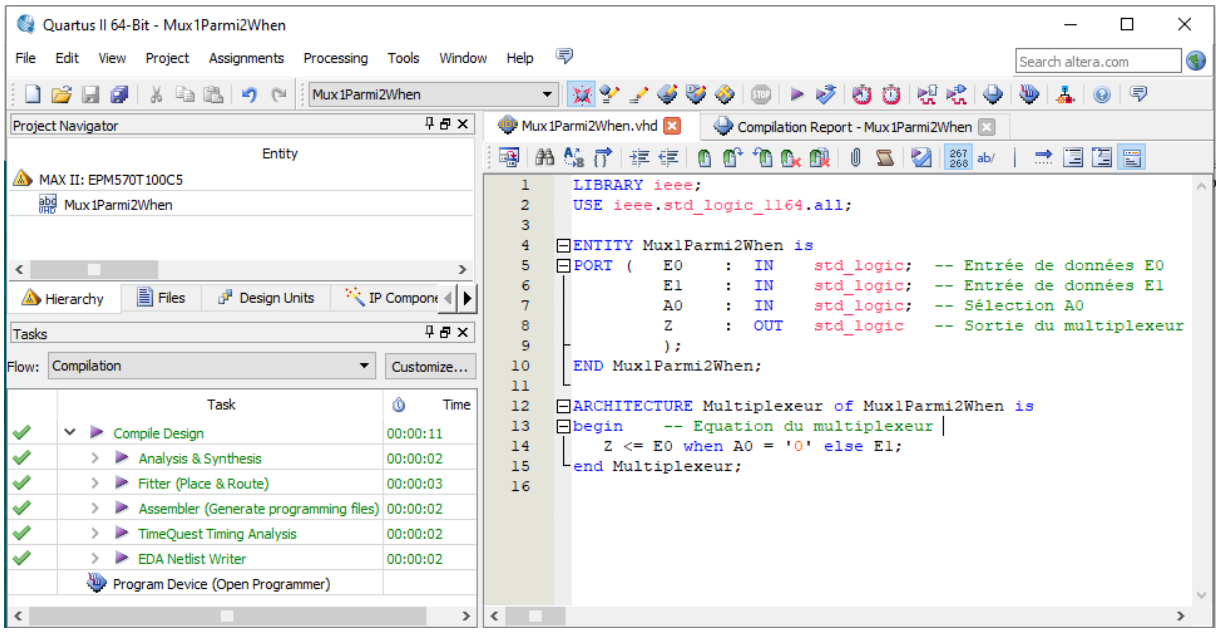
Équation de la sortie Z du multiplexeur.

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

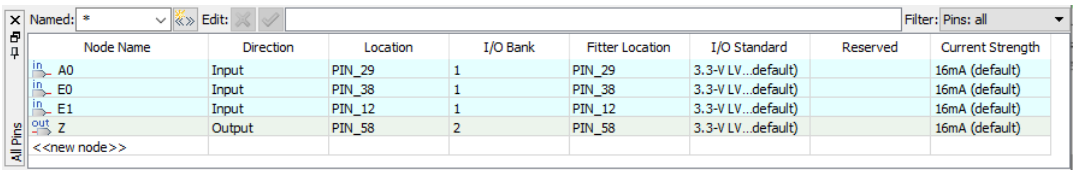
Programme VHDL : Multiplexeur à 2 entrées.



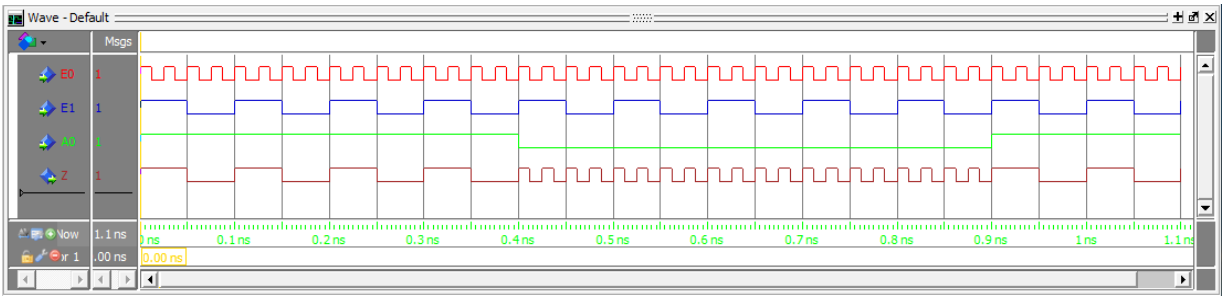
☺ **Autre version utilisant l'instruction "when … else".**



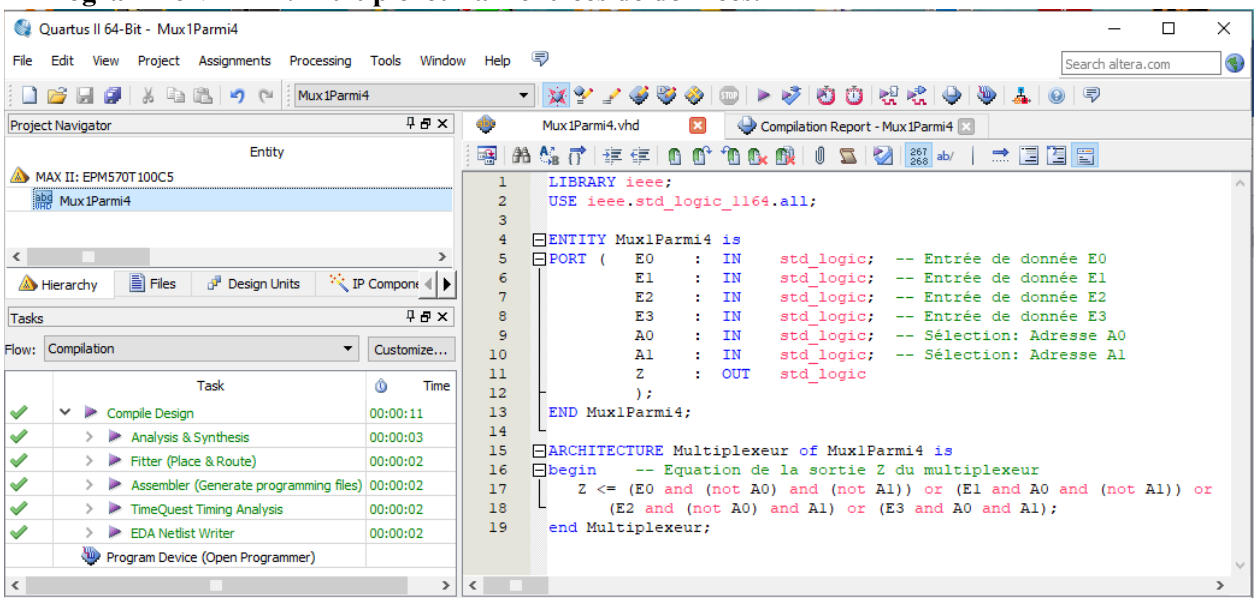
Affectation des broches.



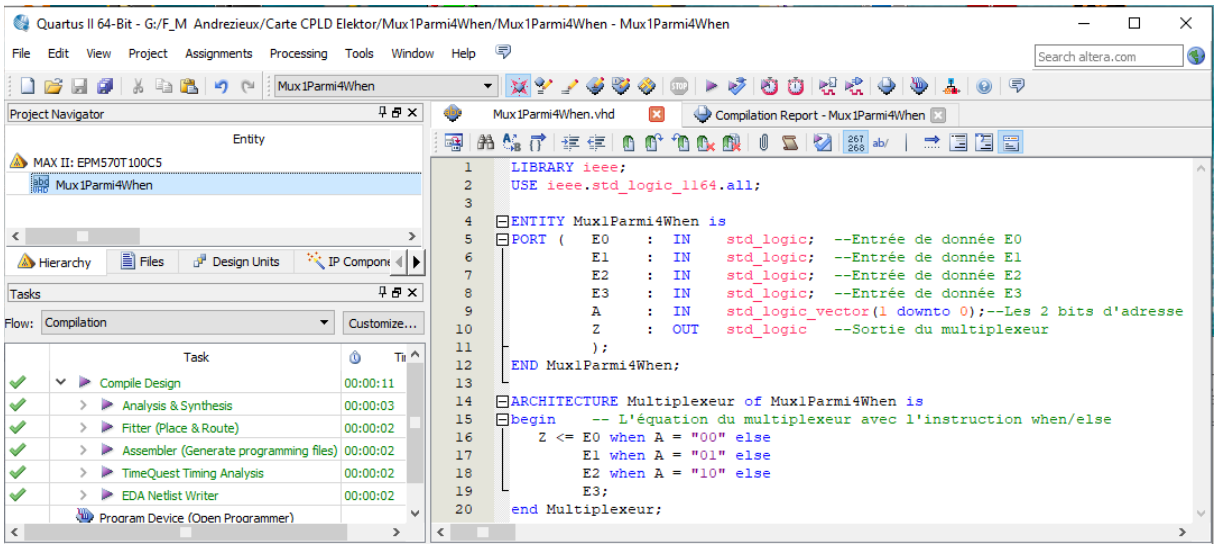
Résultat de la simulation.



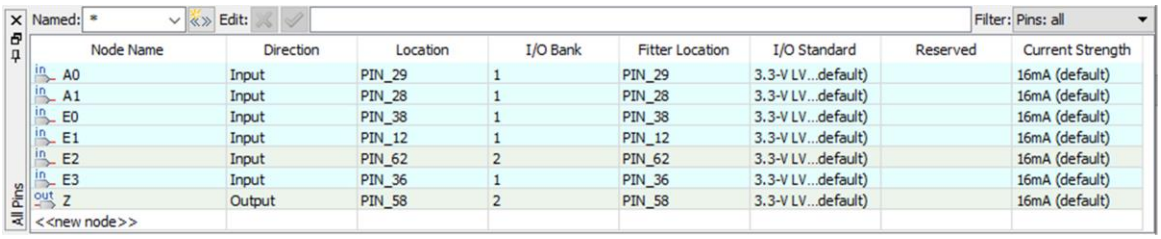
✍ **Programme VHDL : Multiplexeur à 4 entrées de données.**



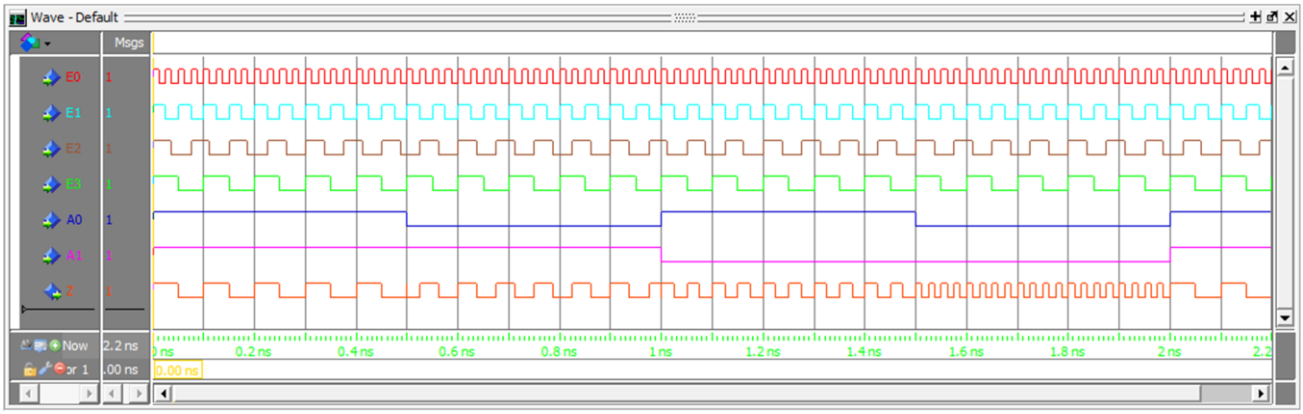
☺ **Autre version utilisant l'instruction "when … else".**



Affectation des broches.



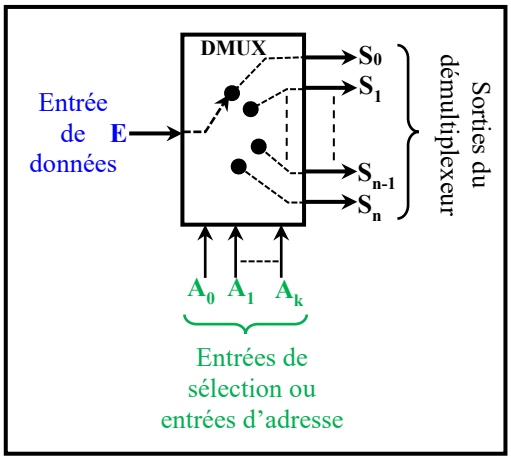
Résultat de la simulation.



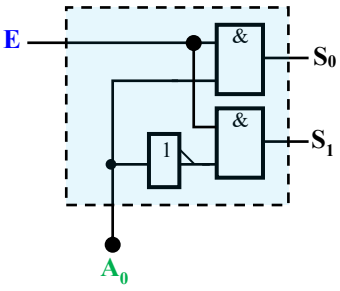
LES DÉMULTIPLEXEURS – Exemples.

Un démultiplexeur (**DMUX**) ou distributeur de données est un circuit n'ayant qu'une entrée de donnée **E** et plusieurs sorties S_0, S_1, S_2, S_3 ….

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_0, A_1, A_2, A_3 ….



Exemple : Démultiplexeur élémentaire à deux sorties S_0, S_1 ou distributeur de données 1 parmi 2.



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

Table de vérité

A_0	S_1	S_0
0	E	0
1	0	E

Autre exemple : Démultiplexeur à 4 sorties S_0, S_1, S_2, S_3 ou distributeur de données 1 parmi 4.

Pour sélectionner une sortie parmi 4, on a besoin de 2 entrées d'adresse A_0 et A_1.

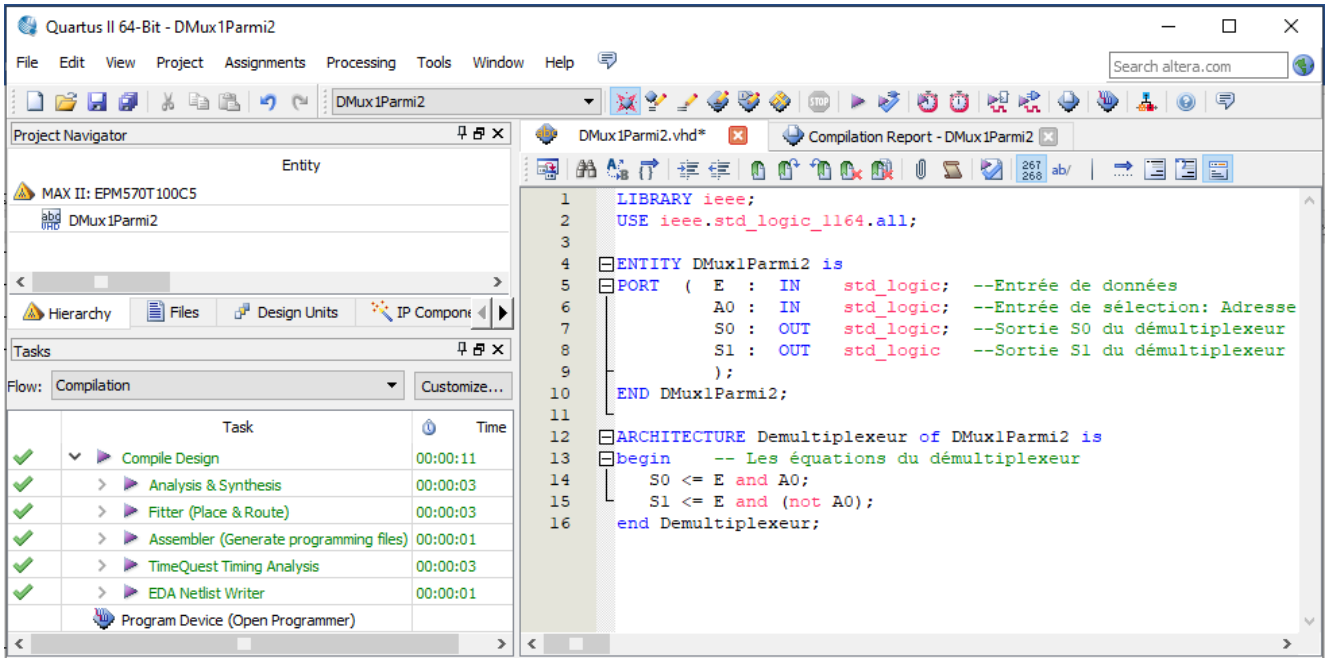
Table de vérité

A_1	A_0	S_3	S_2	S_1	S_0
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.

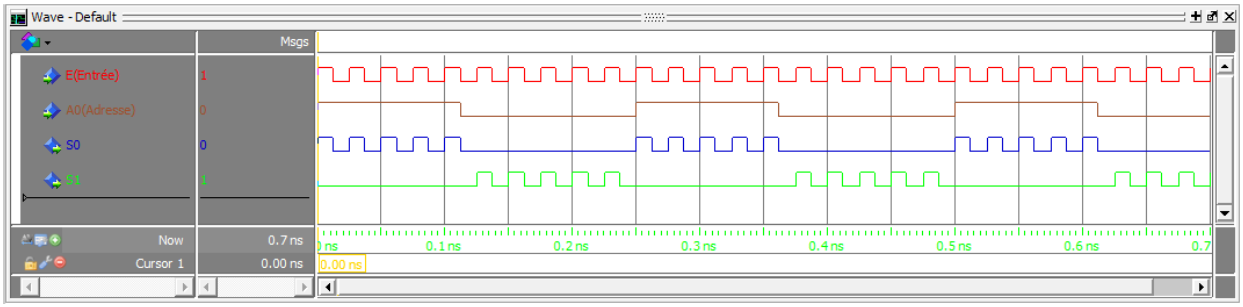


Affectation des broches.

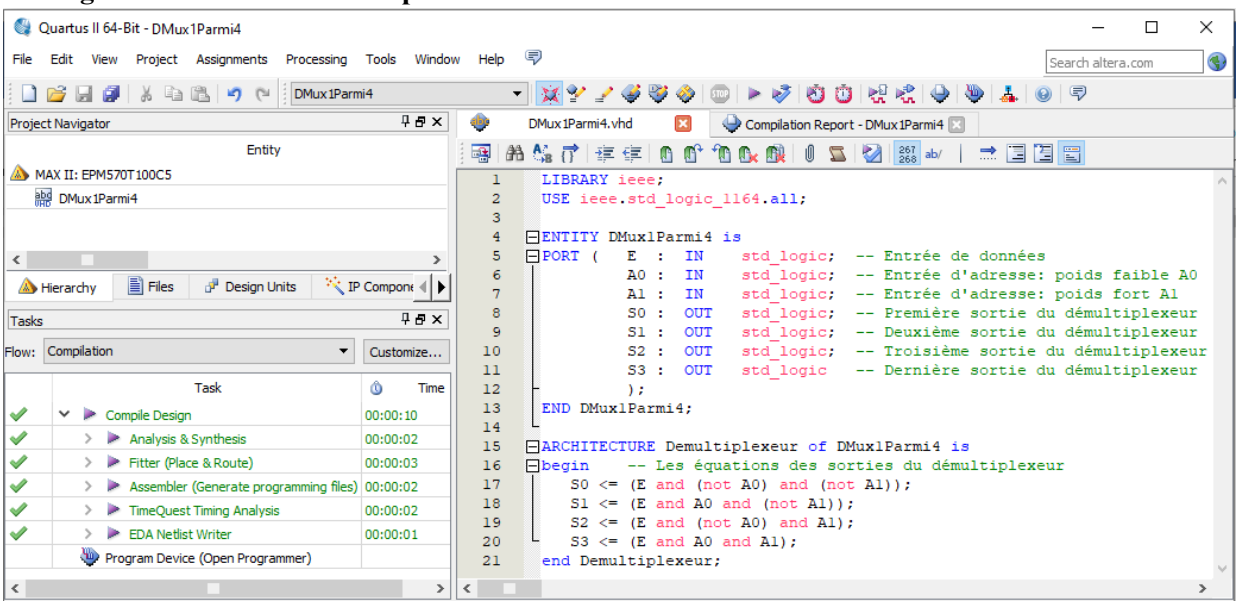
Named: * Edit: Filter: Pins: all

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)
E	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
S0	Output	PIN_58	2	PIN_58	3.3-V LV...default)		16mA (default)
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.

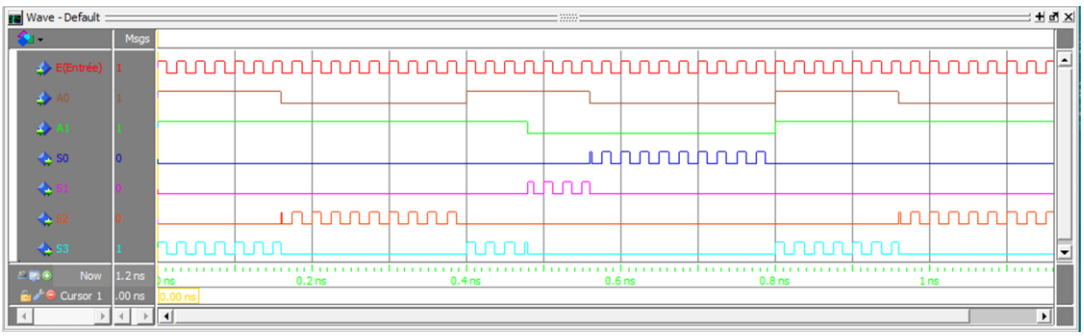


Affectation des broches.

Named: * Edit: Filter: Pins: all

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)
A1	Input	PIN_28	1	PIN_28	3.3-V LV...default)		16mA (default)
E	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
S0	Output	PIN_58	2	PIN_58	3.3-V LV...default)		16mA (default)
S1	Output	PIN_57	2	PIN_57	3.3-V LV...default)		16mA (default)
S2	Output	PIN_54	2	PIN_54	3.3-V LV...default)		16mA (default)
S3	Output	PIN_53	2	PIN_53	3.3-V LV...default)		16mA (default)
<<new node>>							

Résultat de la simulation.

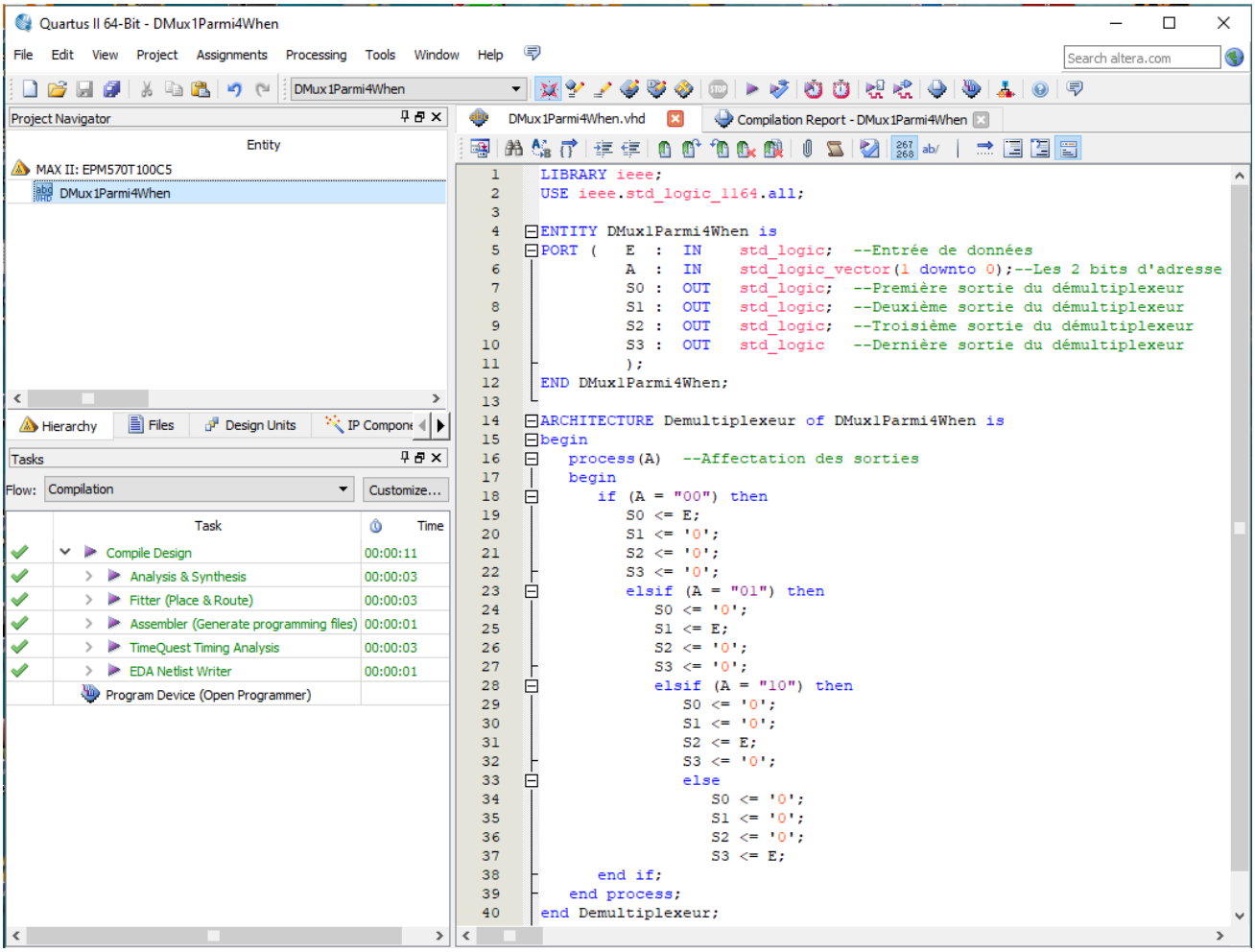


☺ **Autre version utilisant l'instruction "if … then … else … end if".**

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;



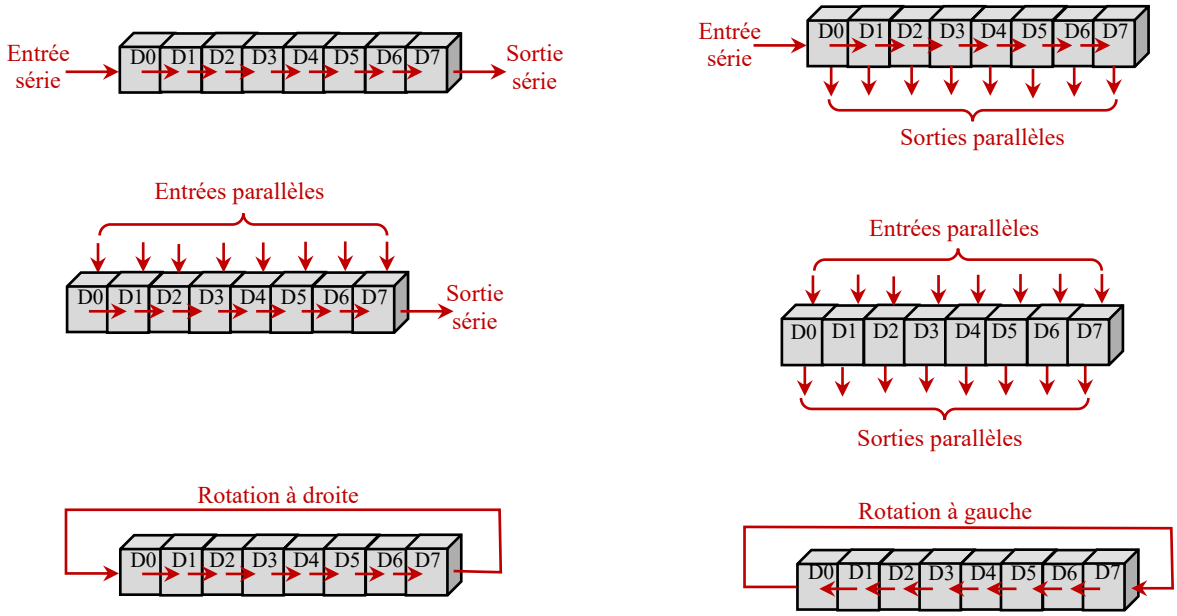
LE REGISTRE A DÉCALAGE.

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.

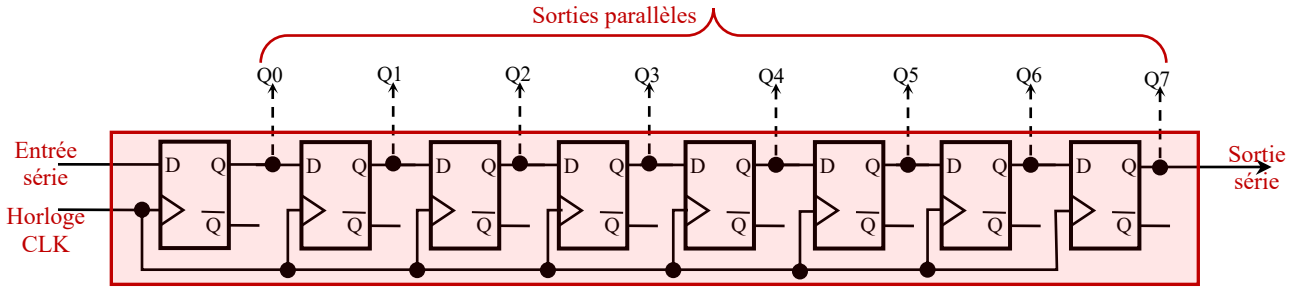
La capacité de stockage d'un registre est le nombre total de bits de données qu'il peut emmagasiner. Chaque étage (ou bascule) représente une capacité de stockage de 1 bit. Un registre à décalage 8 bits est donc une association série de 8 bascules.

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

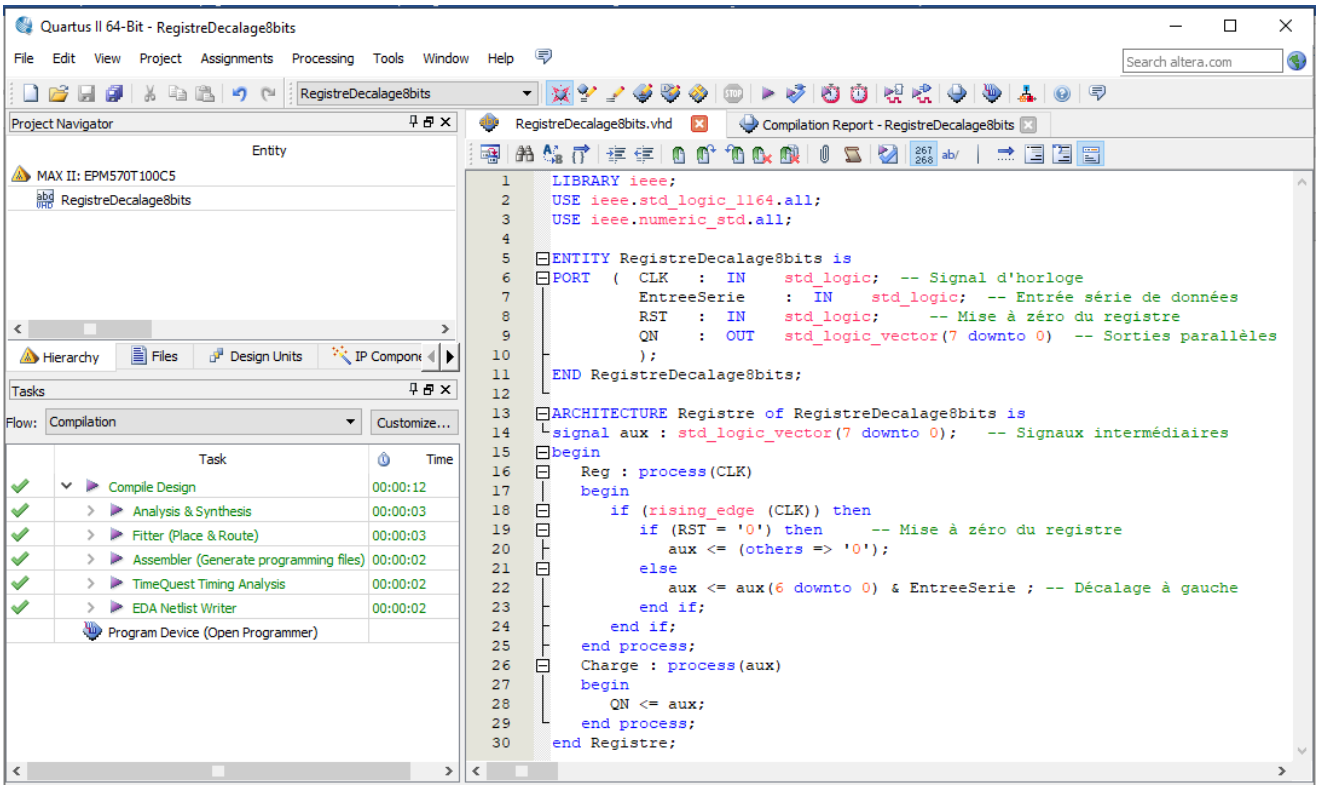
Déplacement des données dans les registres à décalage.



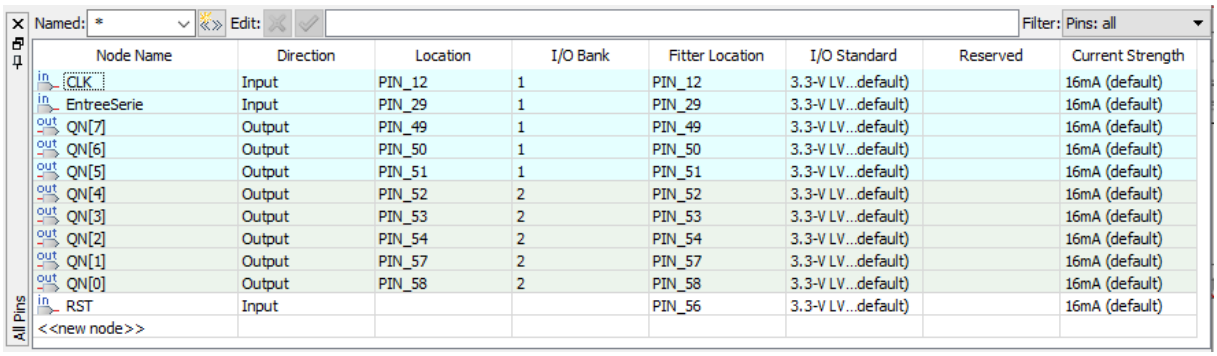
Exemple : Registre à décalage 8 bits – Entrée série et Sortie série ou parallèle.



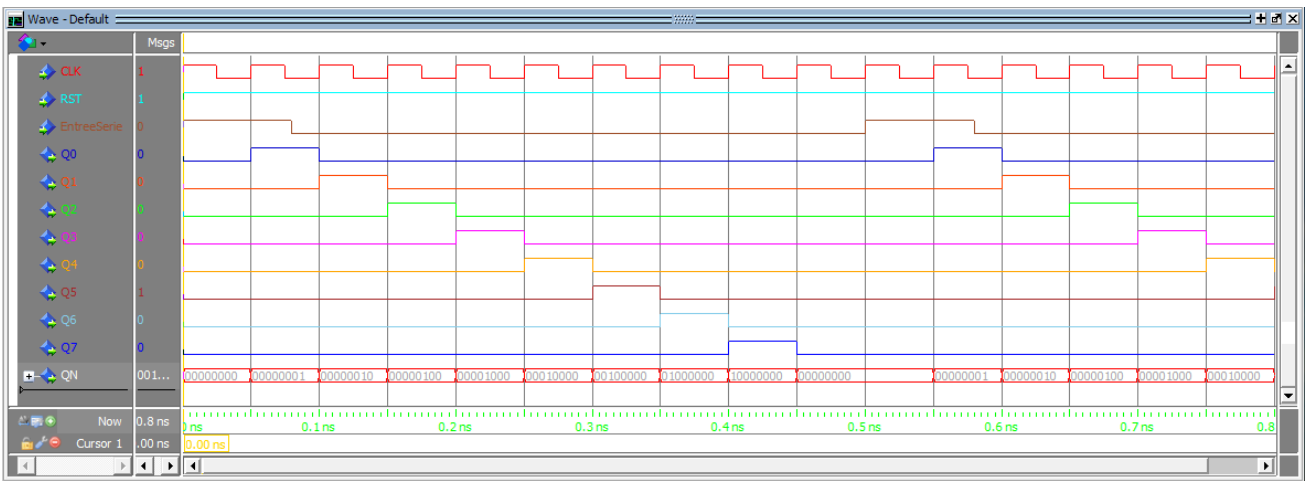
☺ **Programme VHDL.** Entrée série – Sorties parallèles.



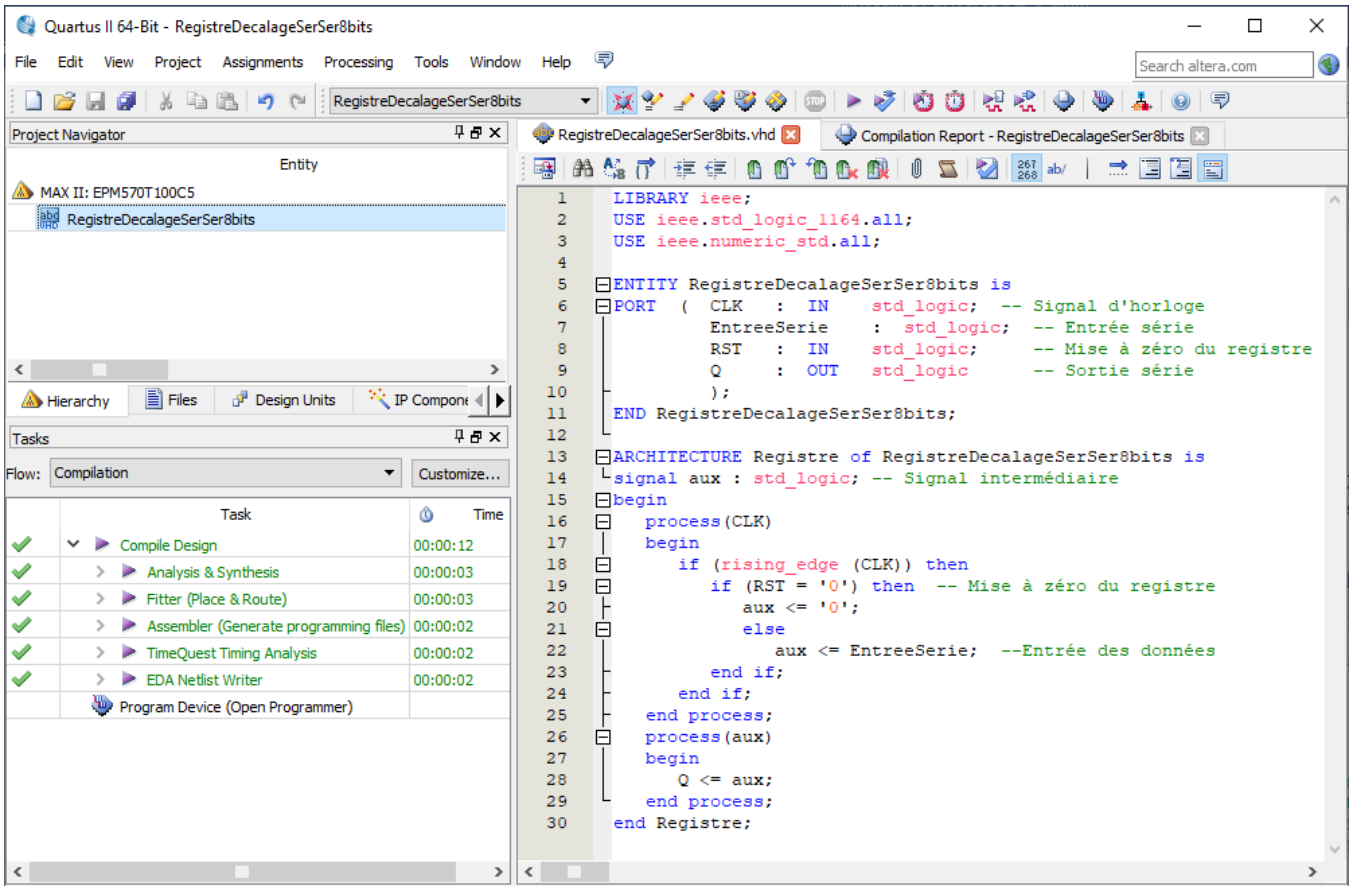
☞ **Affectation des broches.**



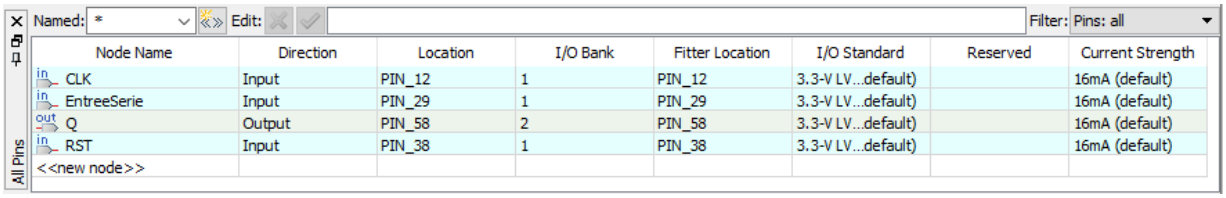
Résultat de la simulation.



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

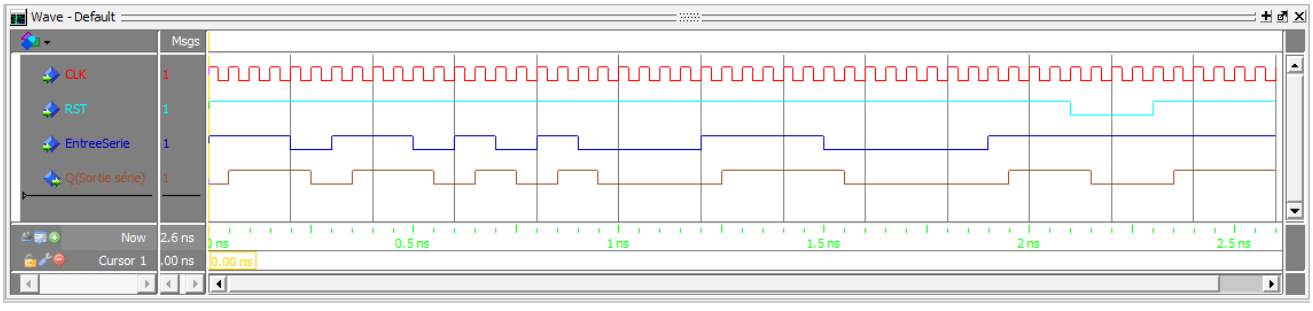


☞ **Affectation des broches.**



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

Résultat de la simulation.



Sources :

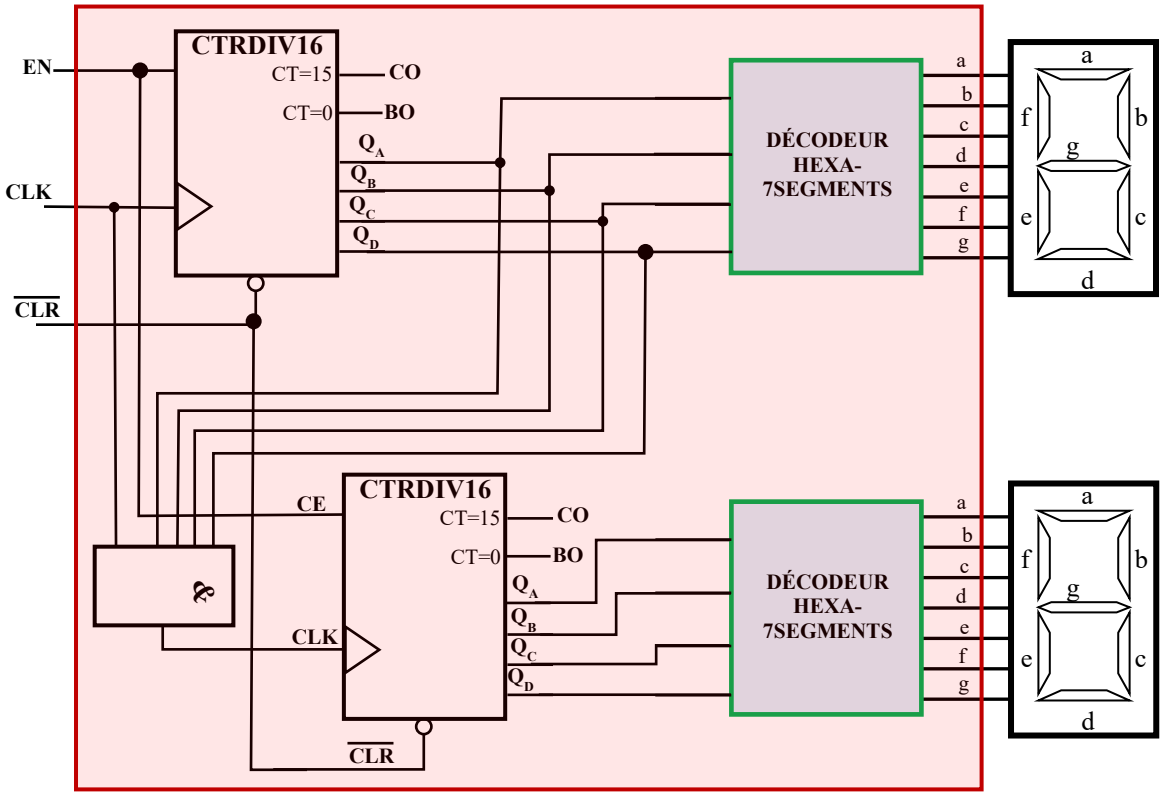
https://fr.wikibooks.org/wiki/TD3_VHDL_Compteurs_et_registres

https://hdl.telecom-paristech.fr/vhdl_comportemental.html

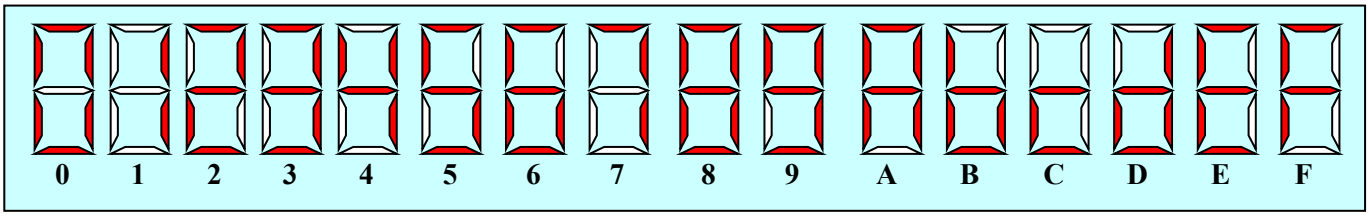
https://www.enib.fr/~kerhoas/vhdl_cours_3.html

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

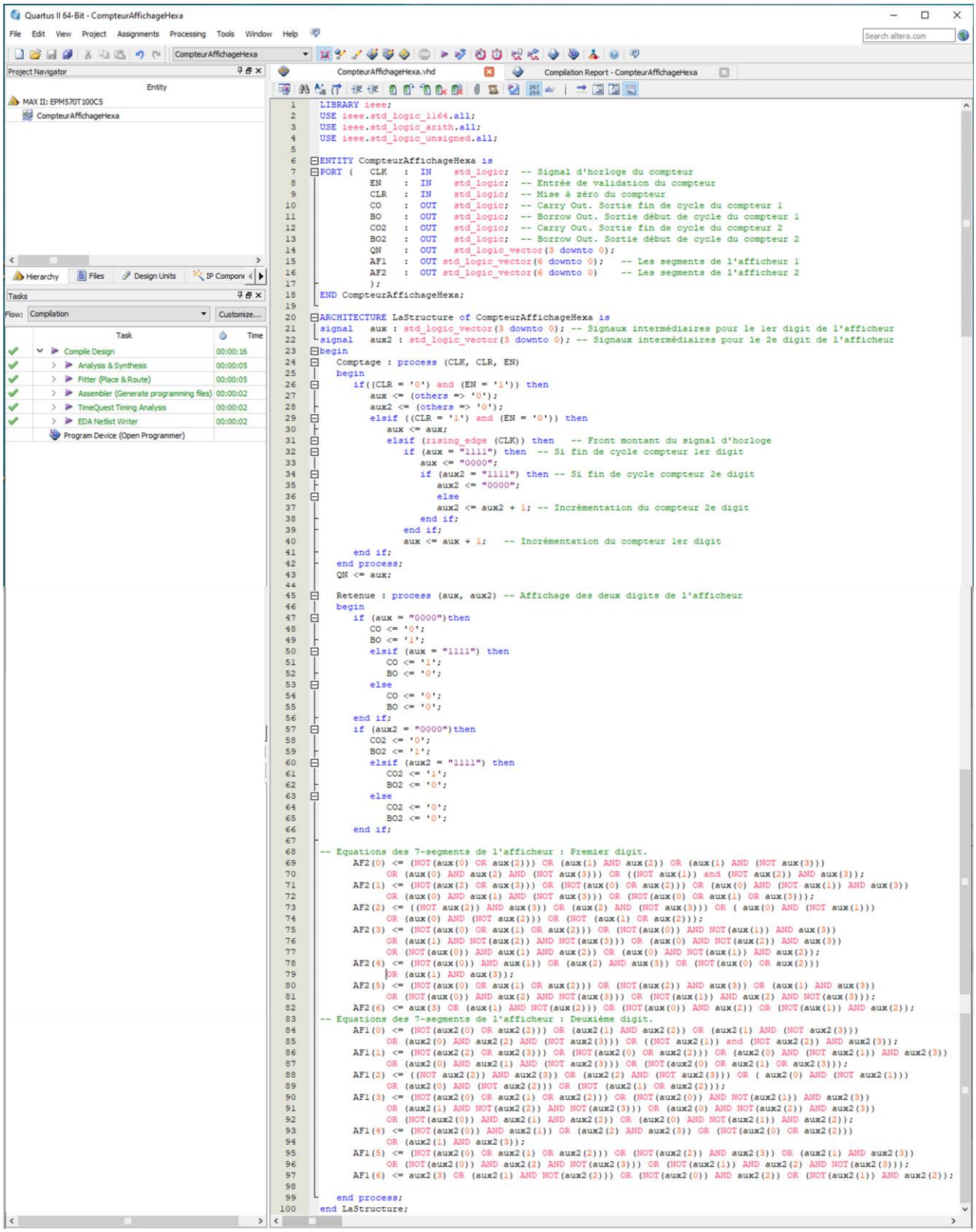
- **CLK :** Signal d'horloge.
- **EN :** Entrée de validation du comptage.
- $\overline{CLR}$: Entrée de mise à zéro du comptage.



Affichage des chiffres décimaux et hexadécimaux.



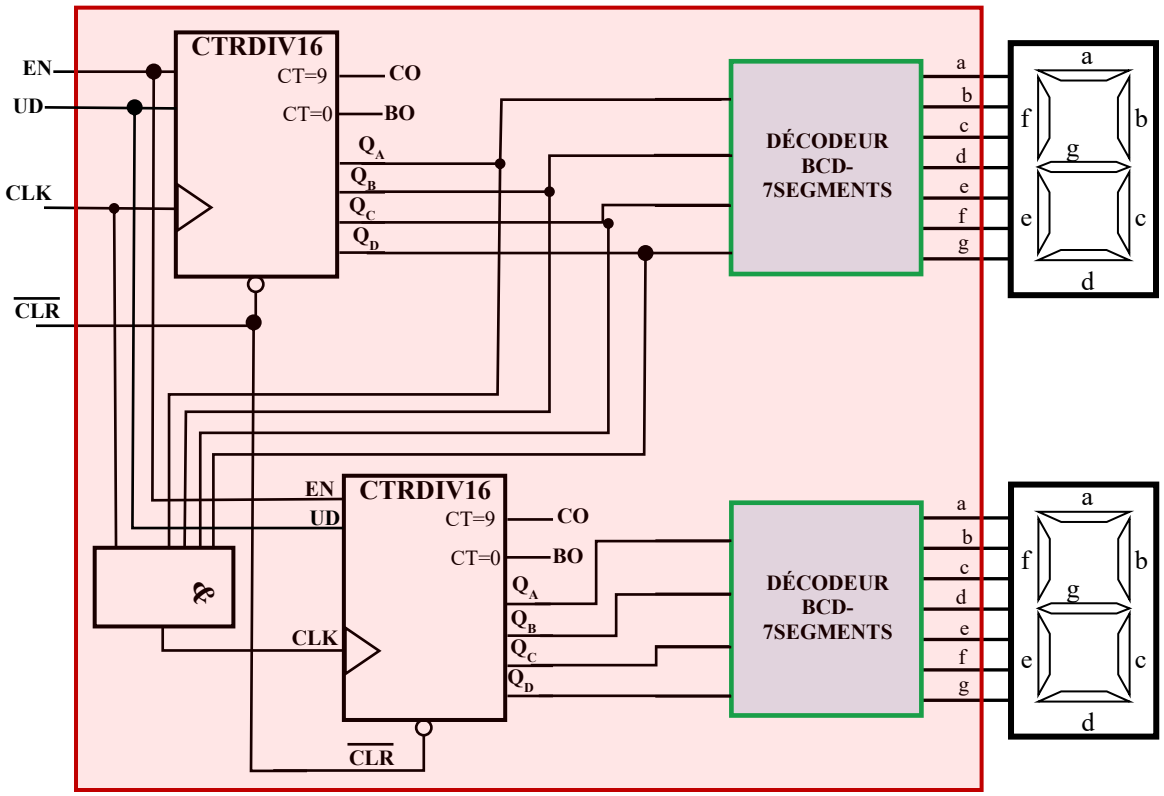
Programme VHDL : "CompteurAffichageHexa.vhd".



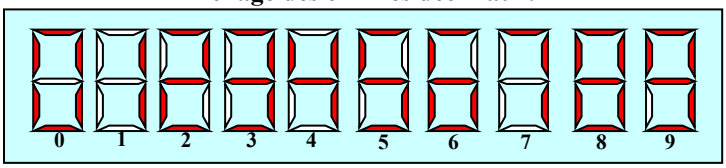

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

ENTITY CompteurAffichageHexa is
PORT (   CLK   :  IN    std_logic;  -- Signal d'horloge du compteur
         EN    :  IN    std_logic;  -- Entrée de validation du compteur
         CLR   :  IN    std_logic;  -- Mise à zéro du compteur
         CO    :  OUT   std_logic;  -- Carry Out. Sortie fin de cycle du compteur 1
         BO    :  OUT   std_logic;  -- Borrow Out. Sortie début de cycle du compteur 1
         CO2   :  OUT   std_logic;  -- Carry Out. Sortie fin de cycle du compteur 2
         BO2   :  OUT   std_logic;  -- Borrow Out. Sortie début de cycle du compteur 2
         QN    :  OUT   std_logic_vector(3 downto 0);
         AF1   :  OUT std_logic_vector(6 downto 0);   -- Les segments de l'afficheur 1
         AF2   :  OUT std_logic_vector(6 downto 0)    -- Les segments de l'afficheur 2
         );
END CompteurAffichageHexa;

ARCHITECTURE LaStructure of CompteurAffichageHexa is
signal   aux : std_logic_vector(3 downto 0); -- Signaux intermédiaires pour le 1er digit de l'afficheur
signal   aux2 : std_logic_vector(3 downto 0); -- Signaux intermédiaires pour le 2e digit de l'afficheur
begin
   Comptage : process (CLK, CLR, EN)
   begin
      if((CLR = '0') and (EN = '1')) then
         aux <= (others => '0');
         aux2 <= (others => '0');
      elsif ((CLR = '1') and (EN = '0')) then
         aux <= aux;
         elsif (rising_edge (CLK)) then   -- Front montant du signal d'horloge
            if (aux = "1111") then  -- Si fin de cycle compteur 1er digit
               aux <= "0000";
               if (aux2 = "1111") then -- Si fin de cycle compteur 2e digit
                  aux2 <= "0000";
                  else
                  aux2 <= aux2 + 1; -- Incrémentation du compteur 2e digit
               end if;
            end if;
            aux <= aux + 1;   -- Incrémentation du compteur 1er digit
      end if;
   end process;
   QN <= aux;

   Retenue : process (aux, aux2) -- Affichage des deux digits de l'afficheur
   begin
      if (aux = "0000")then
         CO <= '0';
         BO <= '1';
         elsif (aux = "1111") then
            CO <= '1';
            BO <= '0';
         else
            CO <= '0';
            BO <= '0';
      end if;
      if (aux2 = "0000")then
         CO2 <= '0';
         BO2 <= '1';
         elsif (aux2 = "1111") then
            CO2 <= '1';
            BO2 <= '0';
         else
            CO2 <= '0';
            BO2 <= '0';
      end if;

   -- Equations des 7-segments de l'afficheur : Premier digit.
      AF2(0) <= (NOT(aux(0) OR aux(2))) OR (aux(1) AND aux(2)) OR (aux(1) AND (NOT aux(3)))
            OR (aux(0) AND aux(2) AND (NOT aux(3))) OR ((NOT aux(1)) and (NOT aux(2)) AND aux(3));
      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))
            OR (aux(0) AND aux(1) AND (NOT aux(3))) OR (NOT(aux(0) OR aux(1) OR aux(3)));
      AF2(2) <= ((NOT aux(2)) AND aux(3)) OR (aux(2) AND (NOT aux(3))) OR ( aux(0) AND (NOT aux(1)))
            OR (aux(0) AND (NOT aux(2))) OR (NOT (aux(1) OR aux(2)));
      AF2(3) <= (NOT(aux(0) OR aux(1) OR aux(2))) OR (NOT(aux(0)) AND NOT(aux(1)) AND aux(3))
            OR (aux(1) AND NOT(aux(2)) AND NOT(aux(3))) OR (aux(0) AND NOT(aux(2)) AND aux(3))
            OR (NOT(aux(0)) AND aux(1) AND aux(2)) OR (aux(0) AND NOT(aux(1)) AND aux(2));
      AF2(4) <= (NOT(aux(0)) AND aux(1)) OR (aux(2) AND aux(3)) OR (NOT(aux(0) OR aux(2)))
            OR (aux(1) AND aux(3));
      AF2(5) <= (NOT(aux(0) OR aux(1) OR aux(2))) OR (NOT(aux(2)) AND aux(3)) OR (aux(1) AND aux(3))
            OR (NOT(aux(0)) AND aux(2) AND NOT(aux(3))) OR (NOT(aux(1)) AND aux(2) AND NOT(aux(3)));
      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));
   -- Equations des 7-segments de l'afficheur : Deuxième digit.
      AF1(0) <= (NOT(aux2(0) OR aux2(2))) OR (aux2(1) AND aux2(2)) OR (aux2(1) AND (NOT aux2(3)))
            OR (aux2(0) AND aux2(2) AND (NOT aux2(3))) OR ((NOT aux2(1)) and (NOT aux2(2)) AND aux2(3));
      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))
            OR (aux2(0) AND aux2(1) AND (NOT aux2(3))) OR (NOT(aux2(0) OR aux2(1) OR aux2(3)));
      AF1(2) <= ((NOT aux2(2)) AND aux2(3)) OR (aux2(2) AND (NOT aux2(3))) OR ( aux2(0) AND (NOT aux2(1)))
            OR (aux2(0) AND (NOT aux2(2))) OR (NOT (aux2(1) OR aux2(2)));
      AF1(3) <= (NOT(aux2(0) OR aux2(1) OR aux2(2))) OR (NOT(aux2(0)) AND NOT(aux2(1)) AND aux2(3))
            OR (aux2(1) AND NOT(aux2(2)) AND NOT(aux2(3))) OR (aux2(0) AND NOT(aux2(2)) AND aux2(3))
            OR (NOT(aux2(0)) AND aux2(1) AND aux2(2)) OR (aux2(0) AND NOT(aux2(1)) AND aux2(2));
      AF1(4) <= (NOT(aux2(0)) AND aux2(1)) OR (aux2(2) AND aux2(3)) OR (NOT(aux2(0) OR aux2(2)))
            OR (aux2(1) AND aux2(3));
      AF1(5) <= (NOT(aux2(0) OR aux2(1) OR aux2(2))) OR (NOT(aux2(2)) AND aux2(3)) OR (aux2(1) AND aux2(3))
            OR (NOT(aux2(0)) AND aux2(2) AND NOT(aux2(3))) OR (NOT(aux2(1)) AND aux2(2) AND NOT(aux2(3)));
      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));

   end process;
end LaStructure;
```

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

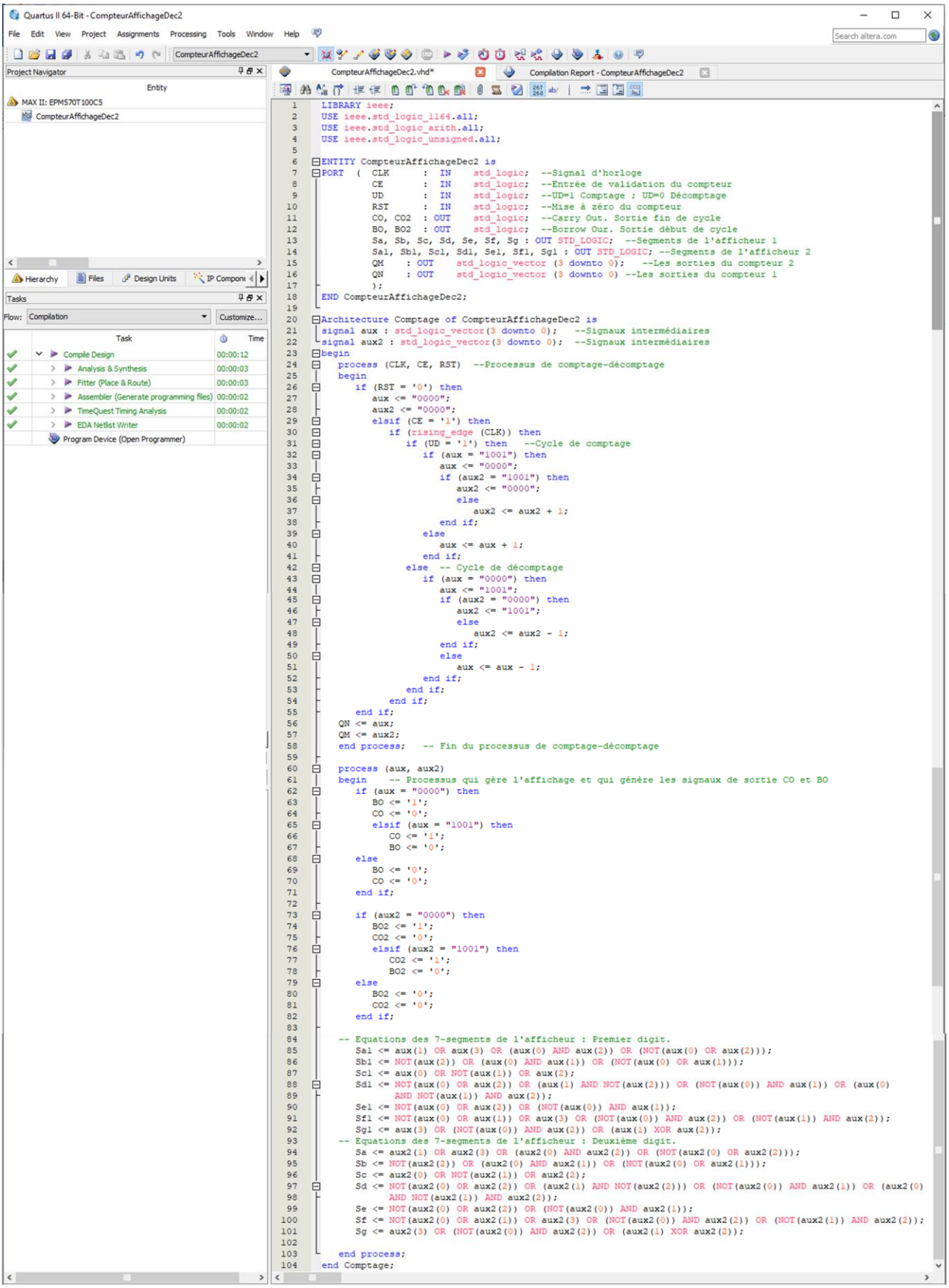
- **CLK :** Signal d'horloge.
- **EN :** Entrée de validation du comptage.
- $\overline{CLR}$: Entrée de mise à zéro du comptage.
- **UD :** Comptage-DécomptagLIBRARYe. **UD = 1** → Comptage ; **UD = 0** → Décomptage.



Affichage des chiffres décimaux.



Programme VHDL : "ComptageAffichageDec2.vhd".



☞ **Affectation des broches.**

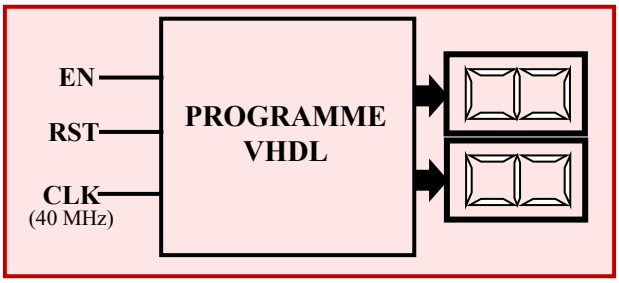
Named: * Edit: Filter: Pins: assigned

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)
<<new node>>							

All Pins

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. $N = \frac{Ln(40000000)}{Ln(2)} = 25{,}25$. 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**.

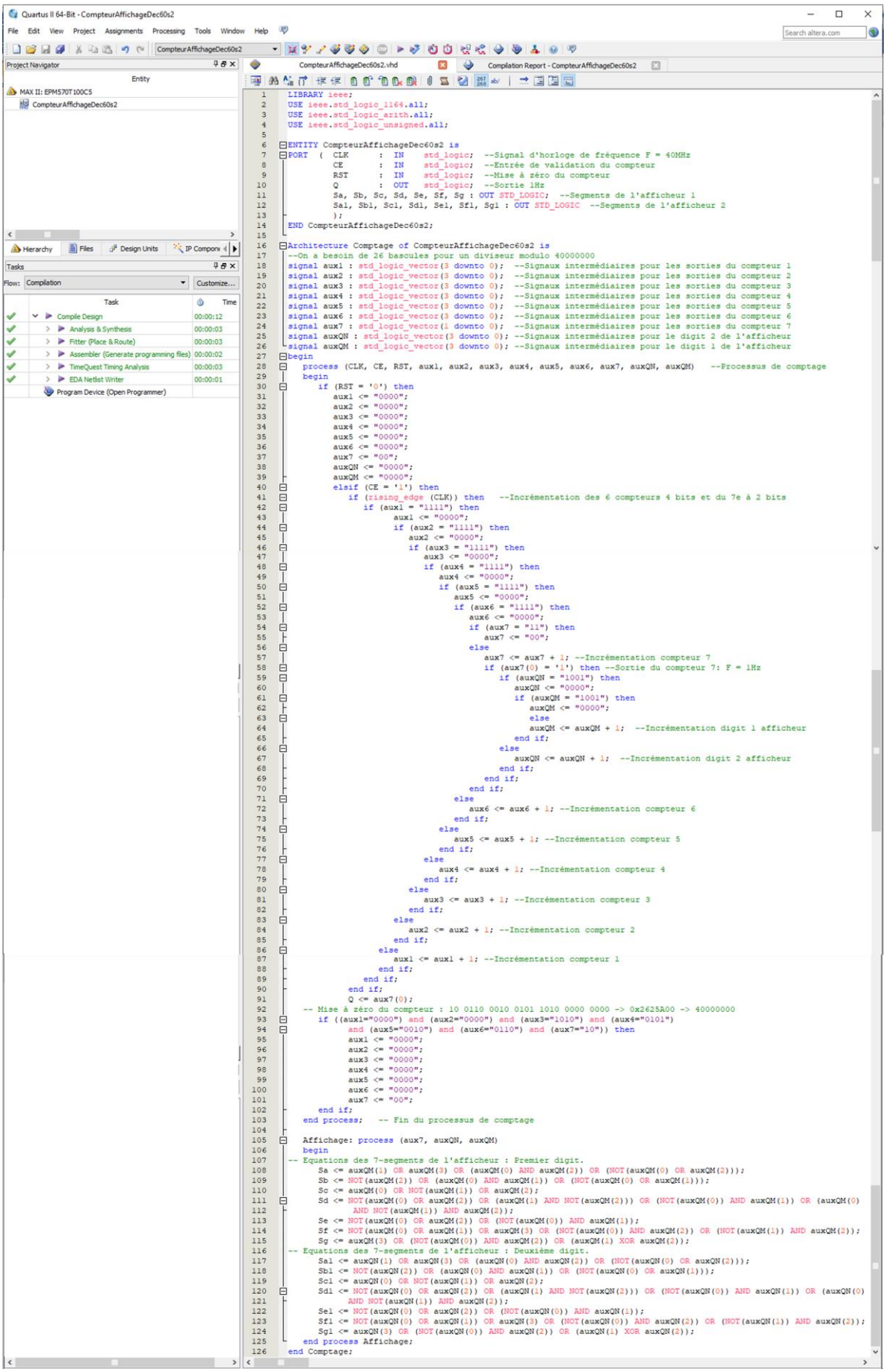
aux1 (aux1(3), aux1(2), aux1(1) et aux1(0)) ; **aux2** (aux2(3), aux2(2), aux2(1) et aux2(0)) :

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"



Affectation des broches.

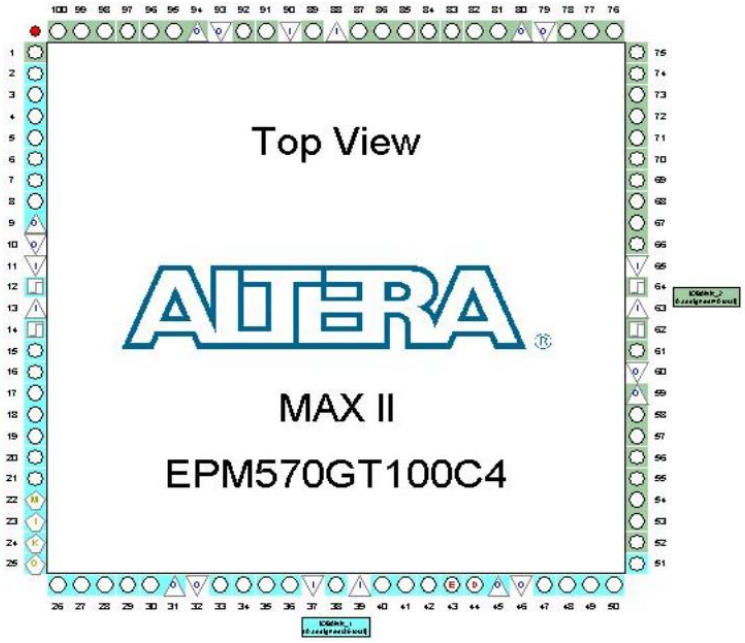
Named: * Edit: Filter: Pins: assigned

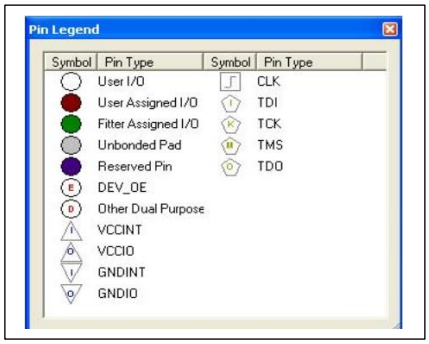
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
CE	Input	PIN_29	1	PIN_29	3.3-V LV...default)		16mA (default)
CLK	Input	PIN_62	2	PIN_62	3.3-V LV...default)		16mA (default)
RST	Input	PIN_38	1	PIN_38	3.3-V LV...default)		16mA (default)
Sa	Output	PIN_68	2	PIN_68	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)
<<new node>>							

All Pins

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

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





- **Vérifier le fonctionnement du compteur : faire valider le fonctionnement par le professeur.**