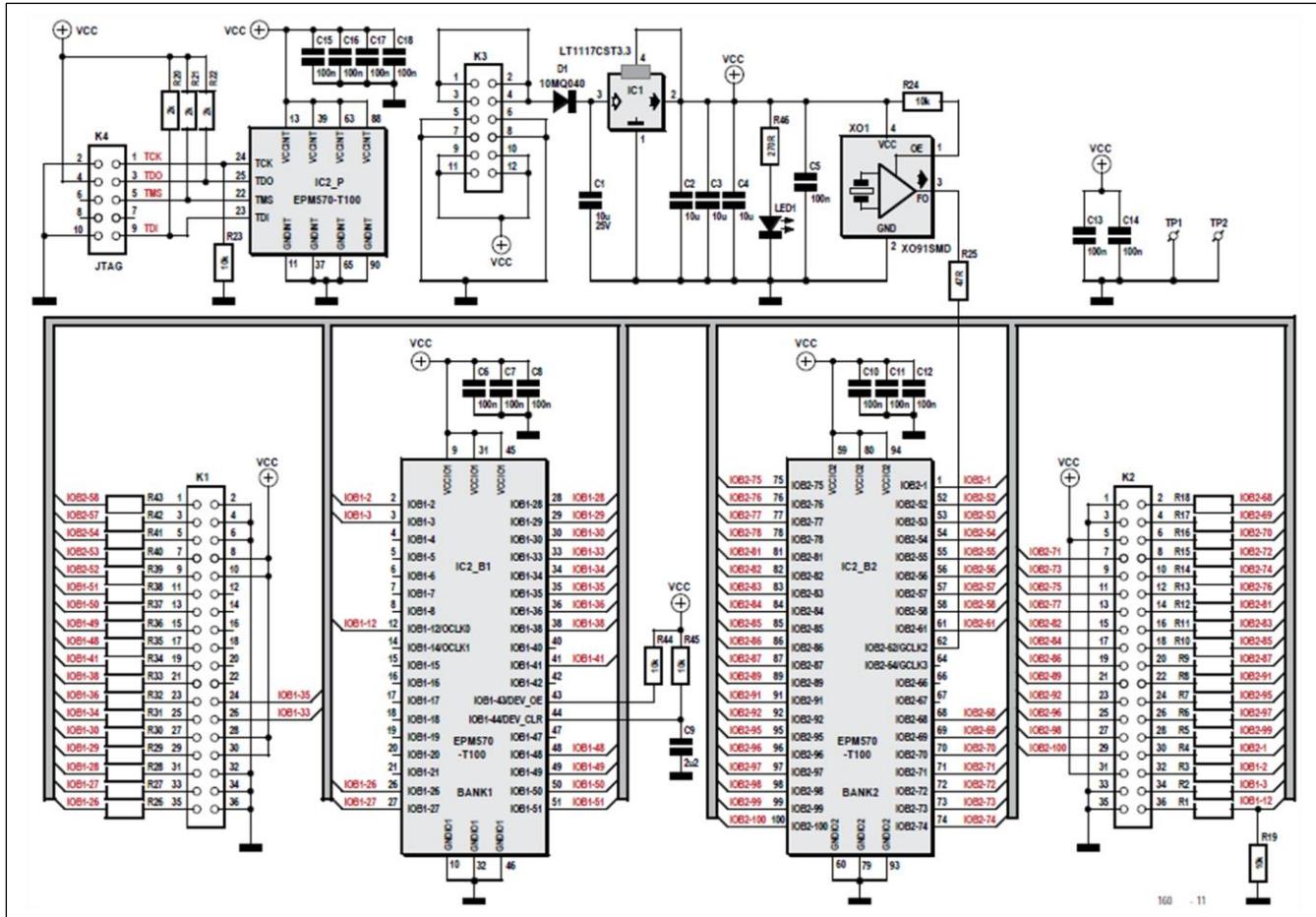


INTERFACE CPLD ELEKTOR.

Carte CPLD ELEKTOR – Schéma structurel.



INTERFACE CARTE CPLD.

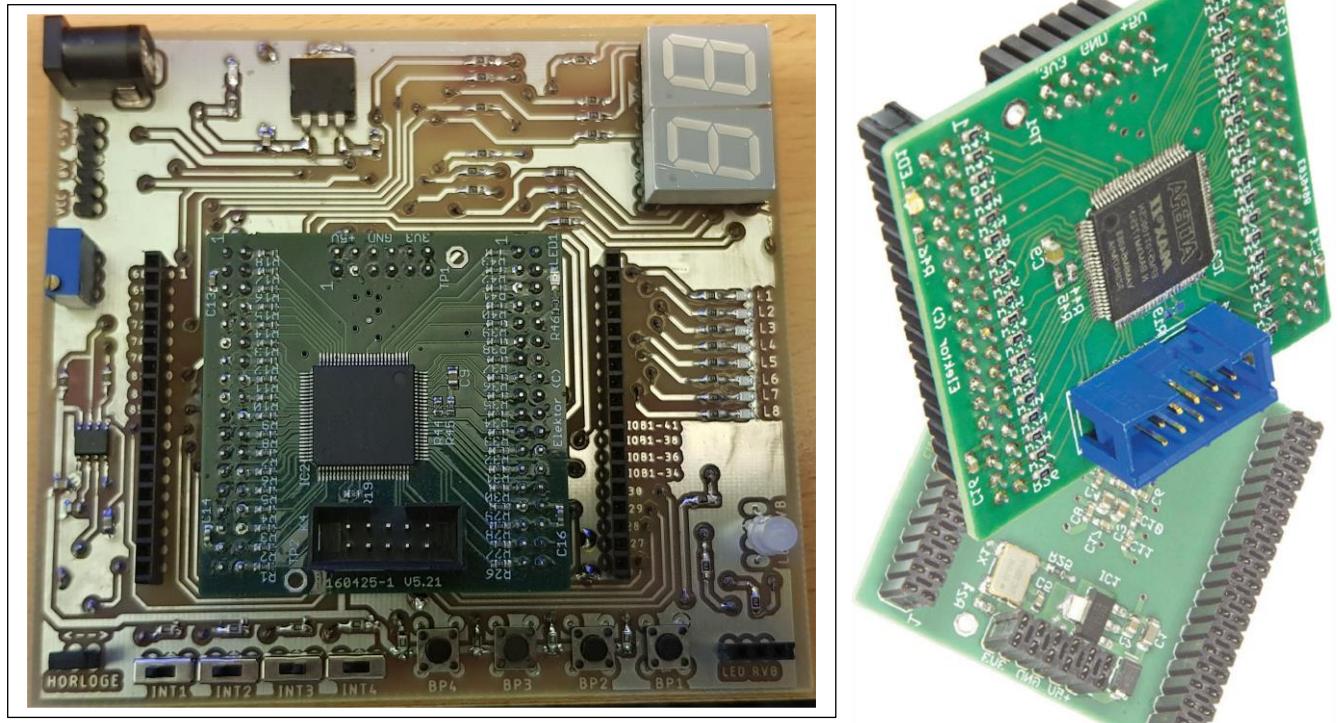
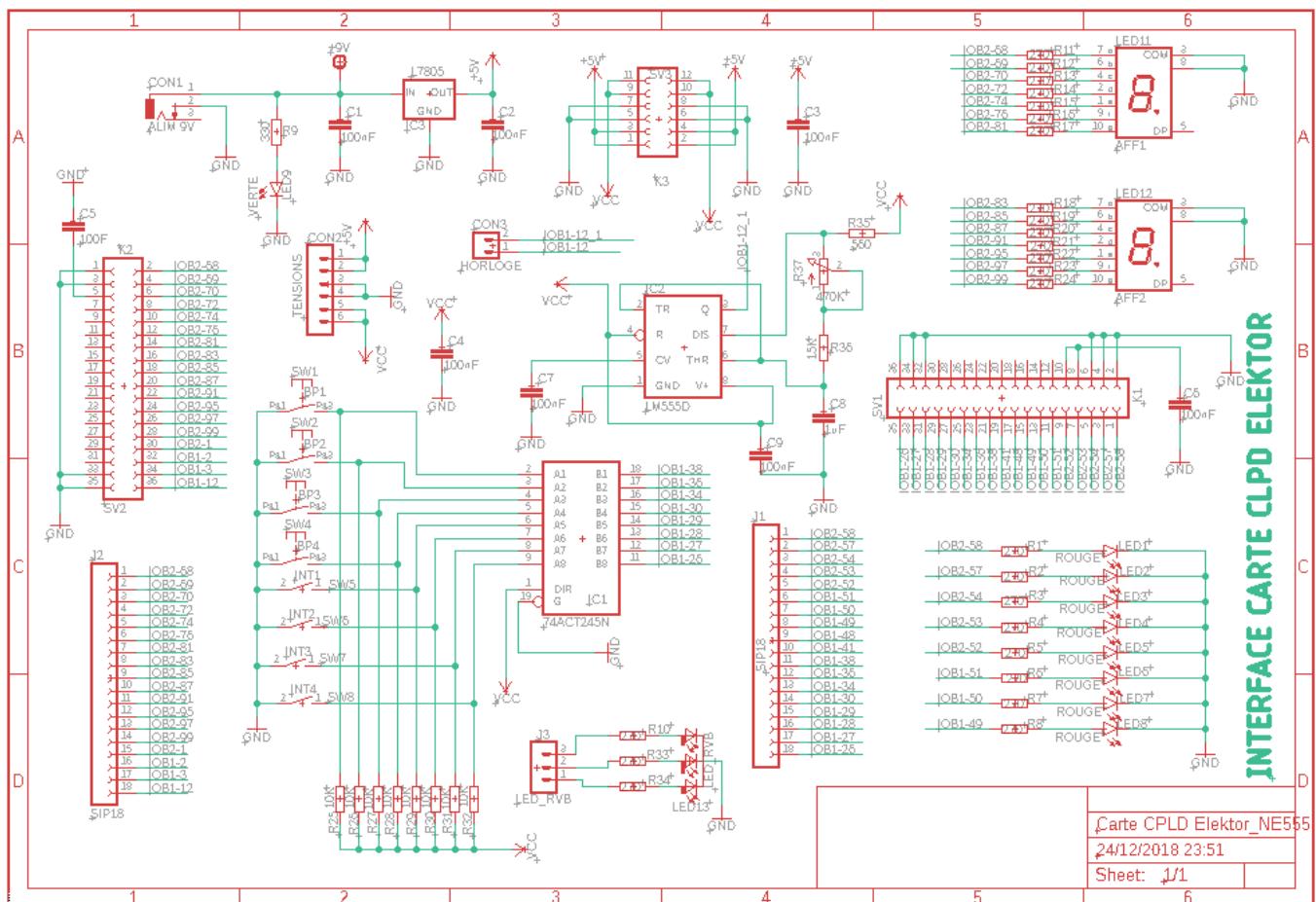


Schéma structurel.



Coté composants.

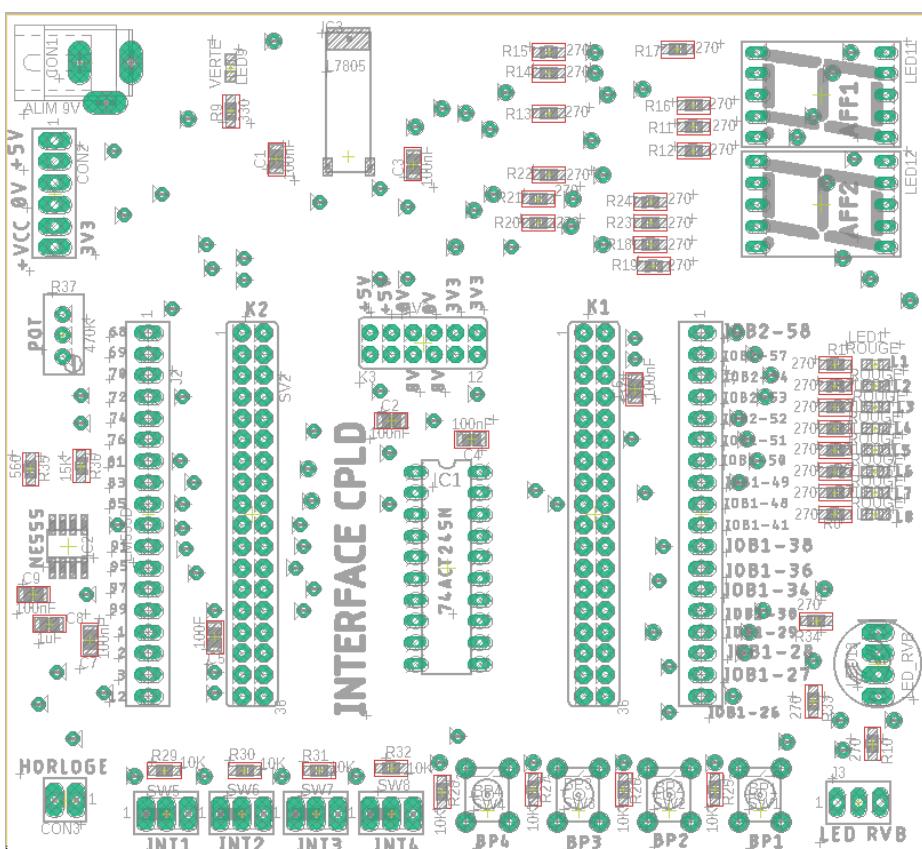
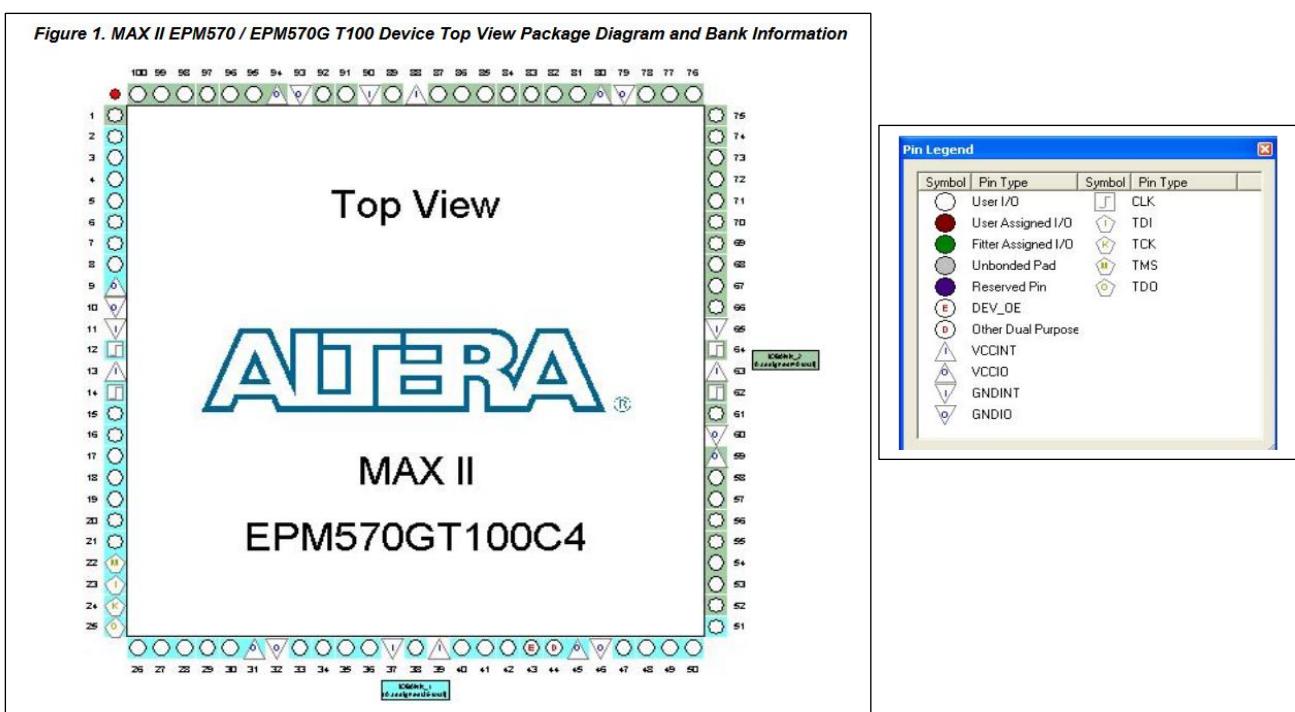


Figure 1. MAX II EPM570 / EPM570G T100 Device Top View Package Diagram and Bank Information



Affectation des broches.

N° Broche	Nom	Affectation	Remarque
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	

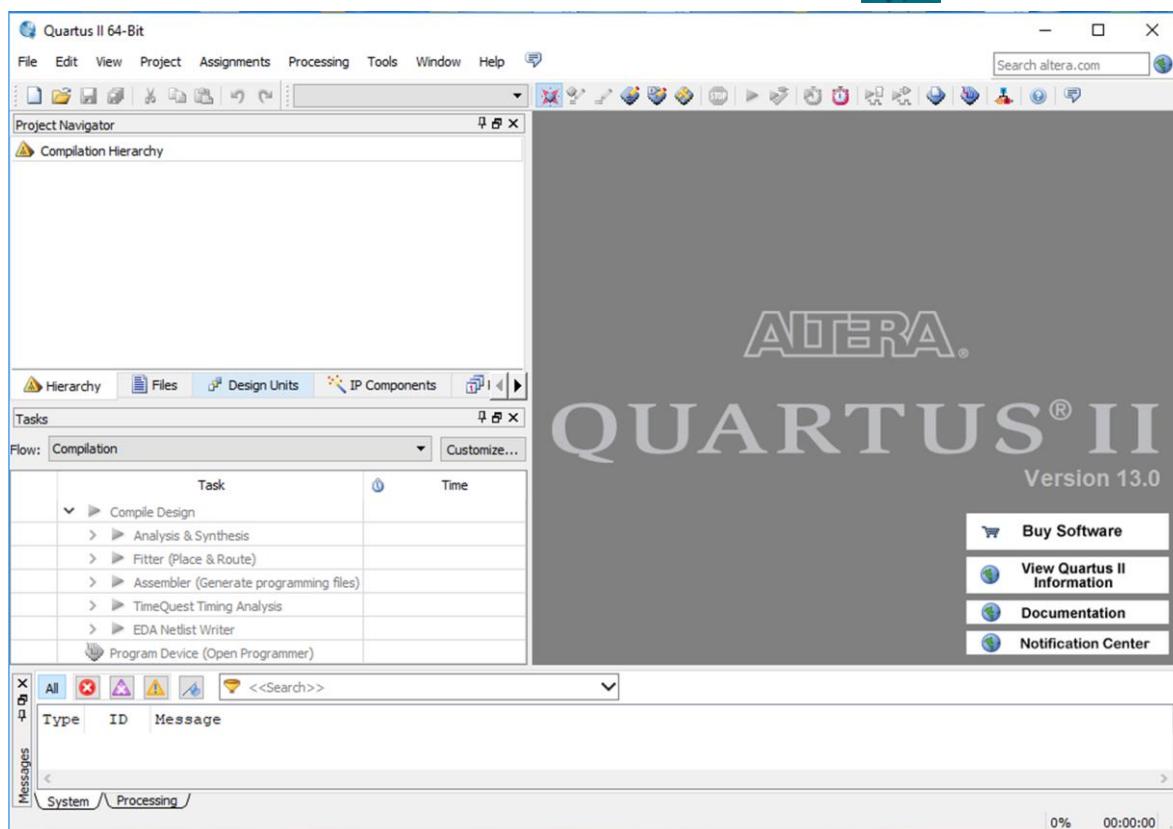
Les interrupteurs INT1, INT2, INT3 et INT4.			
29	IOB1-29	Lecture état de l'interrupteur INT1	
28	IOB1-28	Lecture état de l'interrupteur INT2	
27	IOB1-27	Lecture état de l'interrupteur INT3	
26	IOB1-26	Lecture état de l'interrupteur INT4	

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		

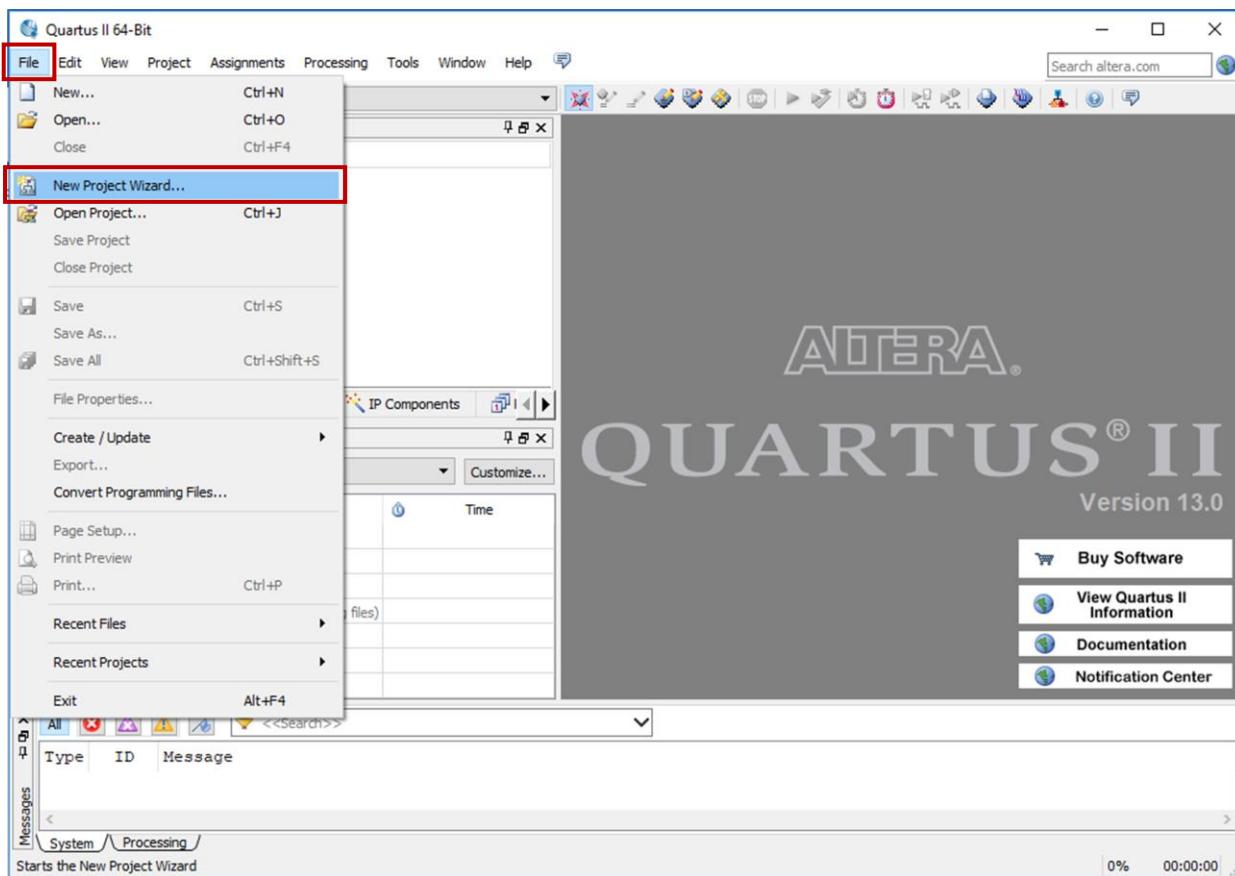
Les signaux d'horloge			
62	IOB2-62/GCLK2	Horloge CPLD 40 MHz	Oscillateur à quartz XO1
12	IOB1-12	Disponible pour une horloge externe	

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

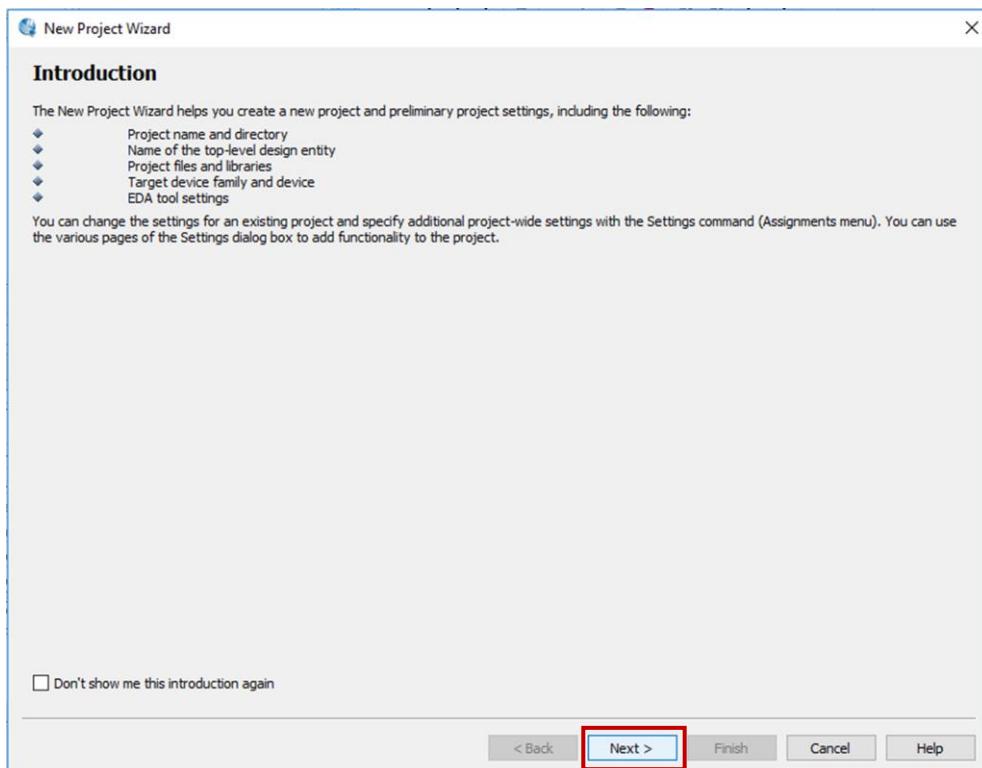
☞ Lancer le logiciel en cliquant sur l'icône "QUARTUS II"

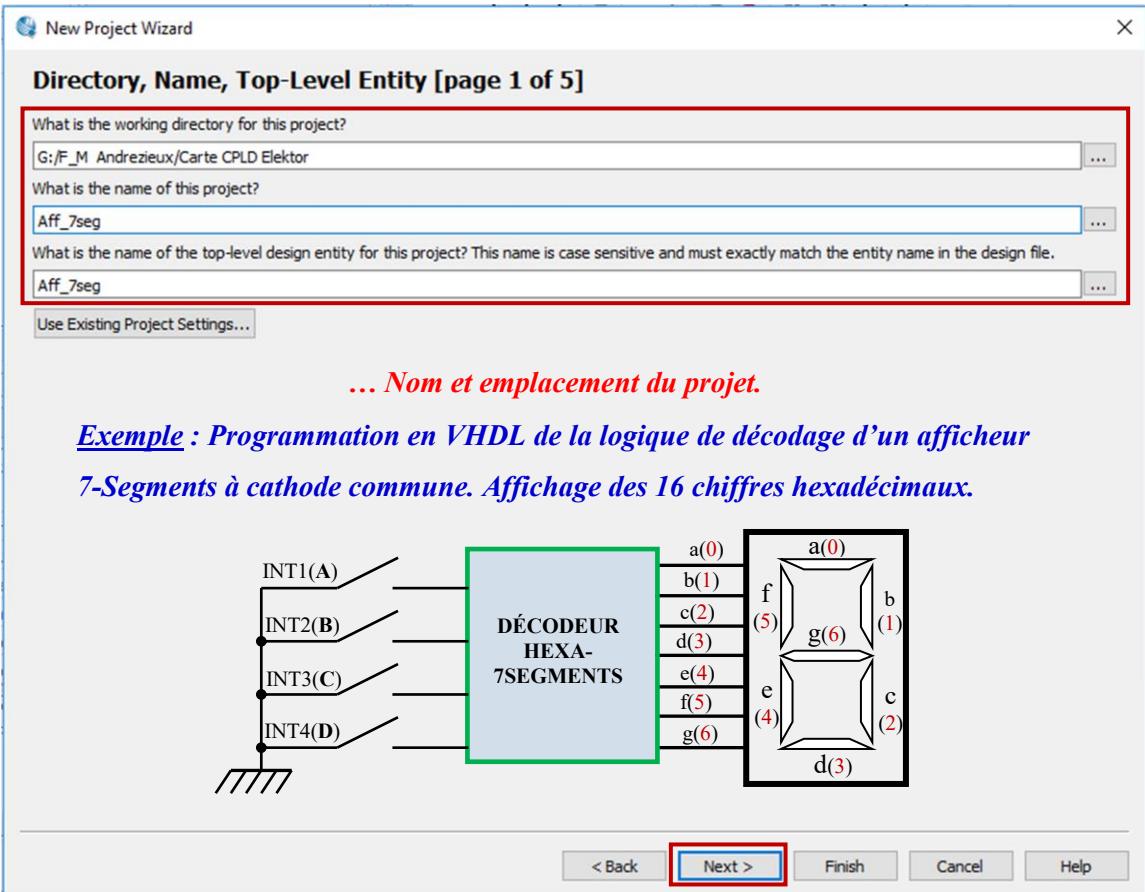


... Choisir ensuite l'assistant pour commencer un nouveau programme.

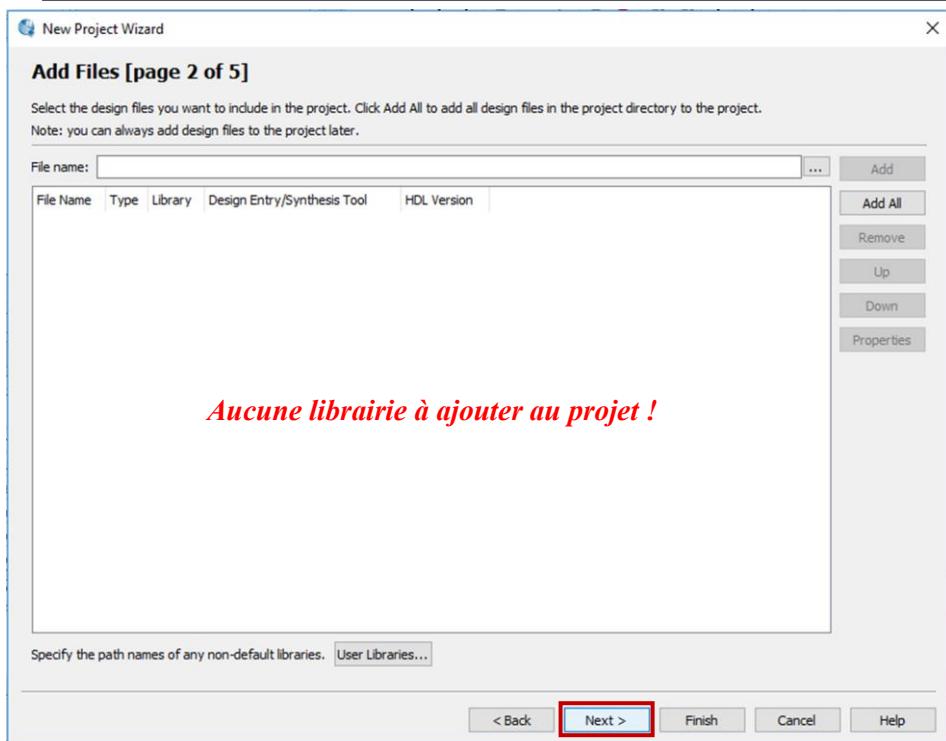
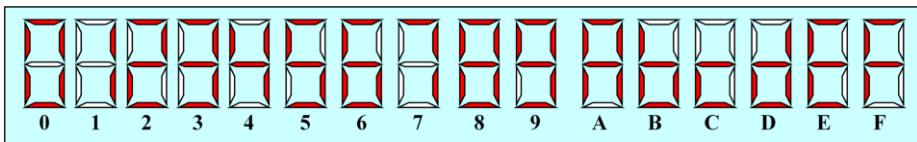


☞ Suivre les cinq étapes proposées par l'assistant.

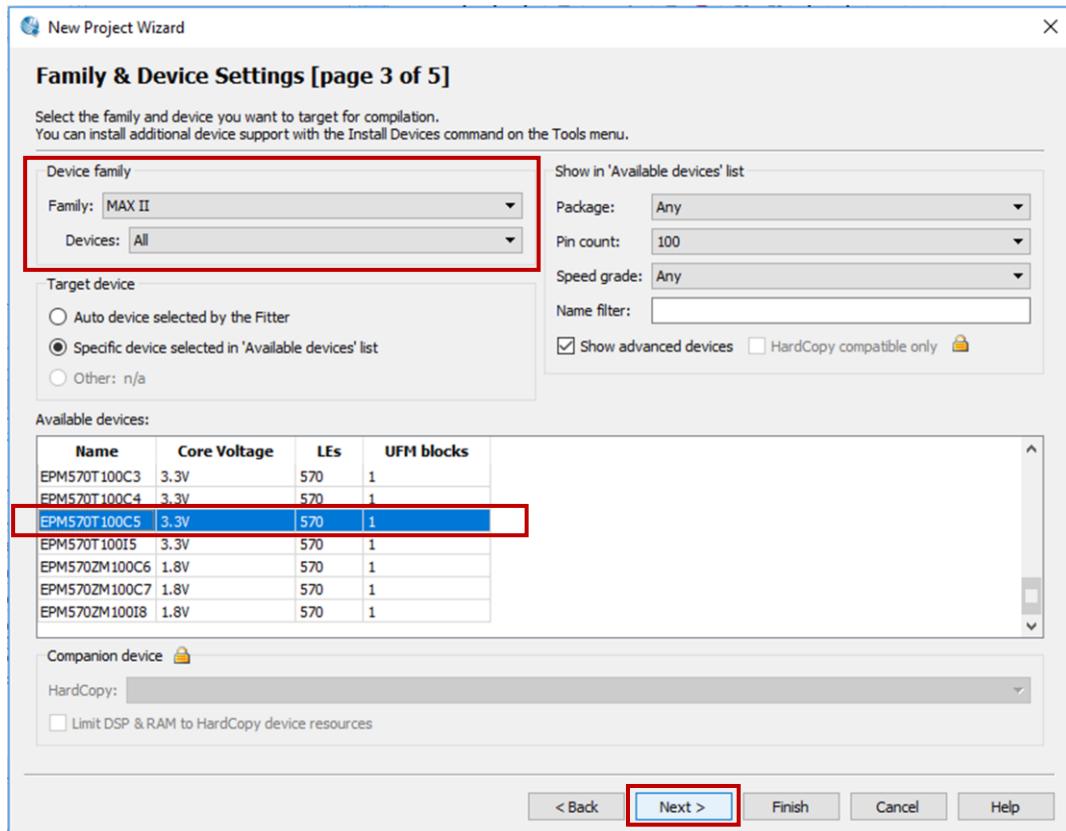




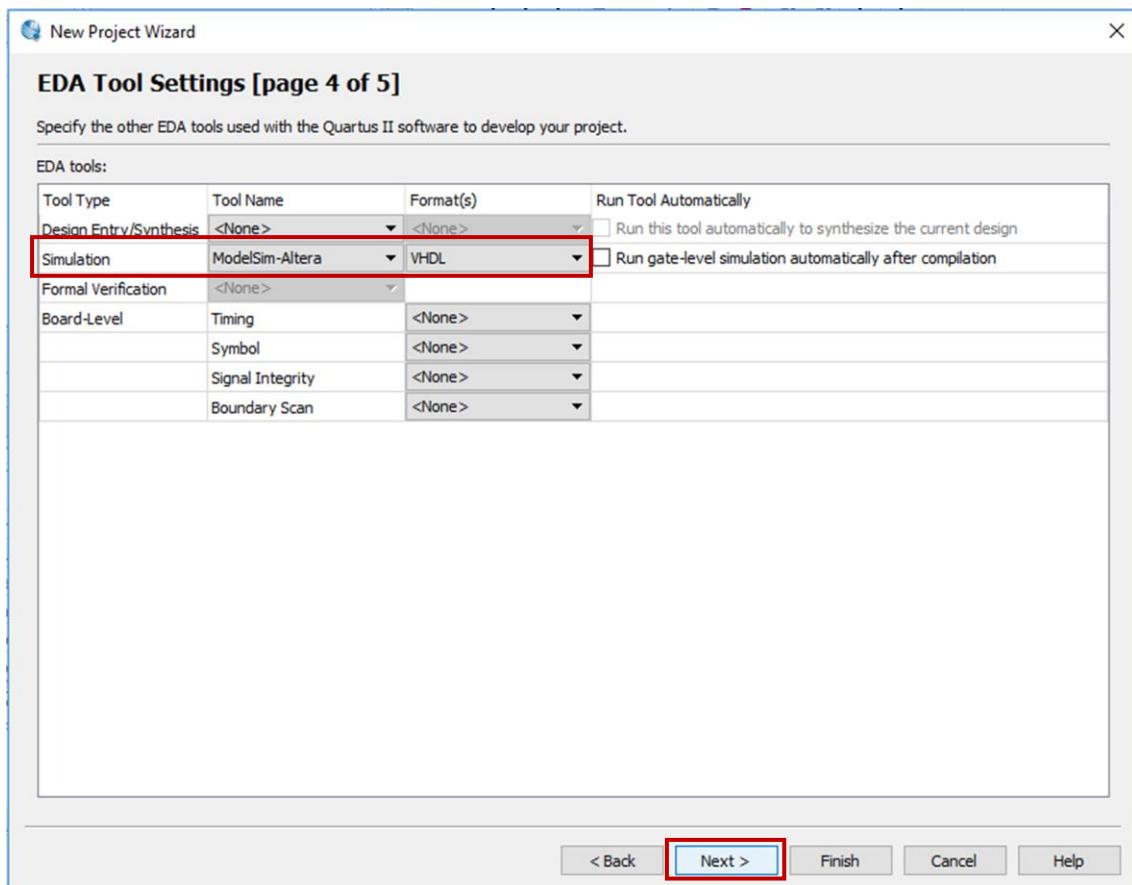
Affichage des chiffres décimaux et hexadécimaux.

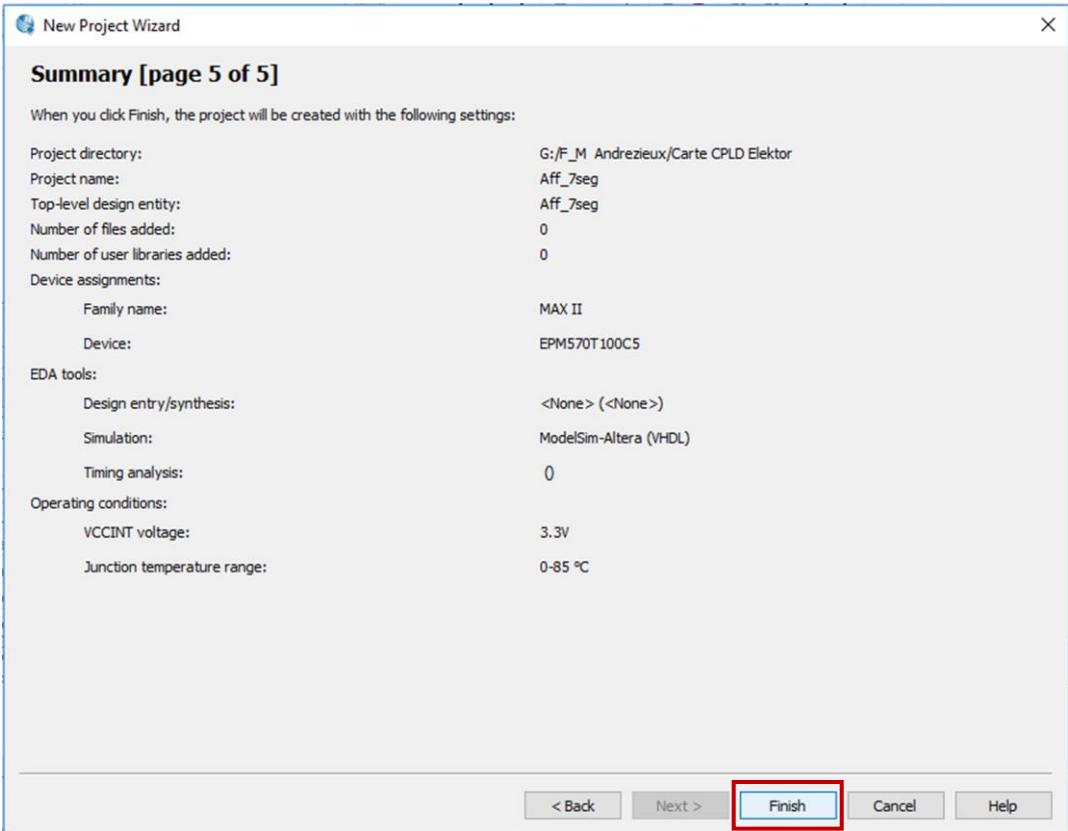


☞ Choisir le circuit logique à programmer : **EPM570T100C5**.



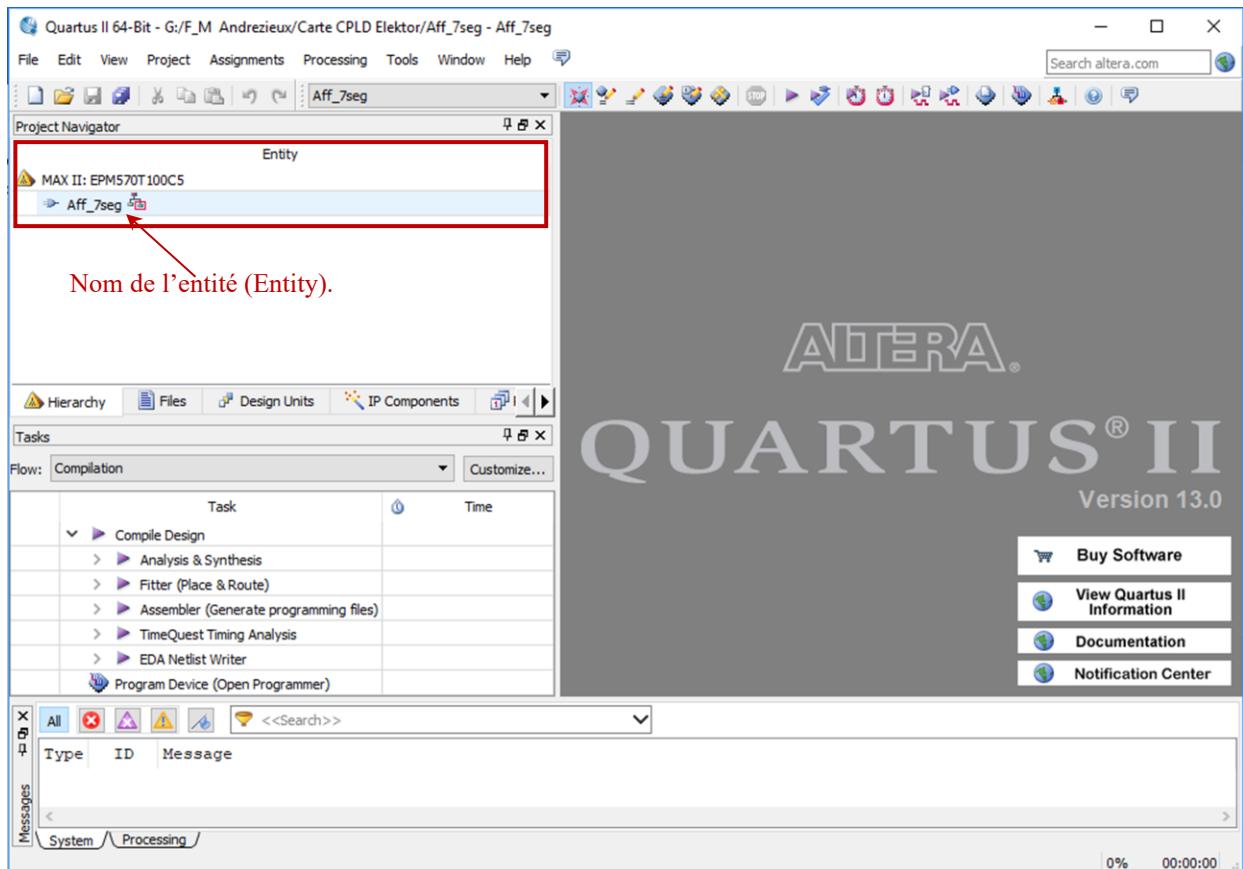
☞ Choisir le module de simulation : **ModelSim-Altera VHDL**.



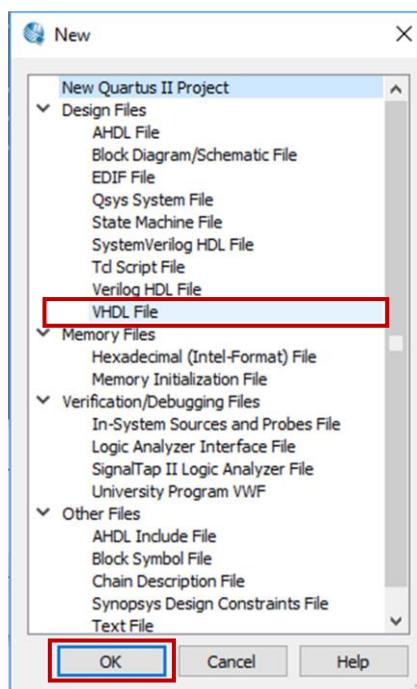
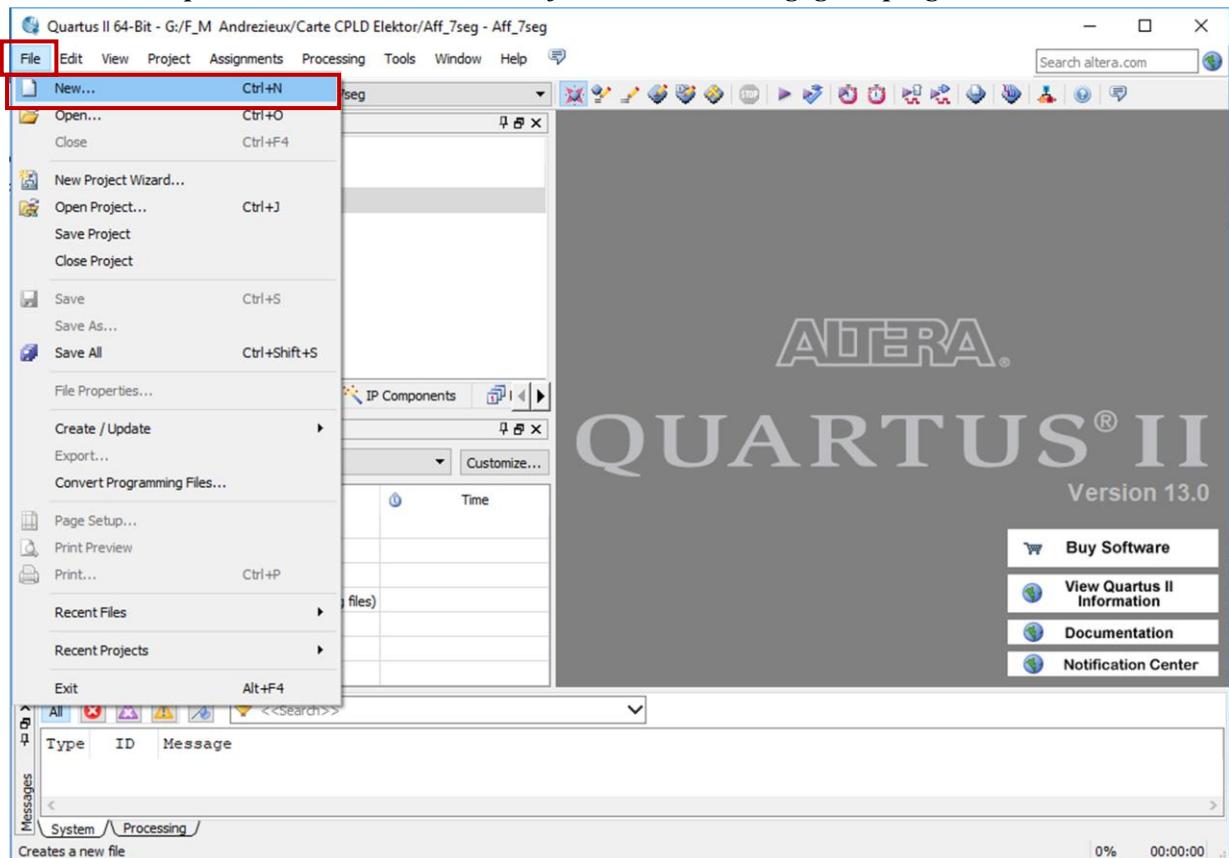


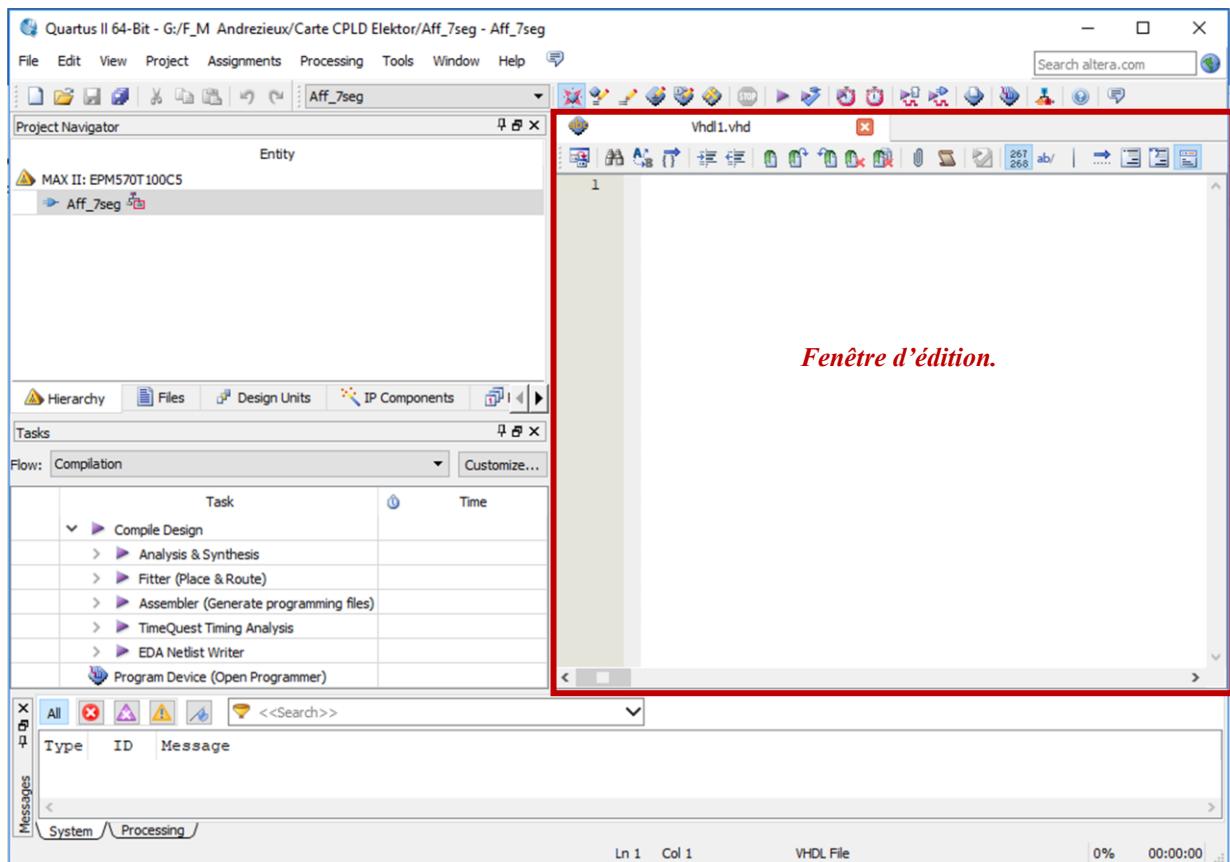
Cliquer sur "Finish" pour fermer l'assistant.

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;

```

```

1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3
4 Entity Aff_7seg is
5 PORT(A, B, C, D : IN STD_LOGIC;
6      Sa, Sb, Sc, Sd, Se, Sf, Sg : OUT STD_LOGIC);
7 END Aff_7seg;
8
9 ARCHITECTURE Afficheur OF Aff_7seg IS
10 BEGIN
11   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);
12   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));
13   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));
14   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
15   | (NOT(A) AND B AND C) OR (A AND NOT(B) AND C);
16   Se <= (NOT(A) AND B) OR (C AND D) OR (NOT(A OR C)) OR (B AND D);
17   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));
18   Sg <= D OR (B AND NOT(C)) OR (NOT(A) AND C) OR (NOT(B) AND C);
19 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.

Flow Status	Successful - Sat Dec 08 20:28:57 2018
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	Aff_7seg
Top-level Entity Name	Aff_7seg
Family	MAX II
Device	EPM570T100C5
Timing Models	Final
Total logic elements	7 / 570 (1 %)
Total pins	11 / 76 (14 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

☒ Affectation des broches.

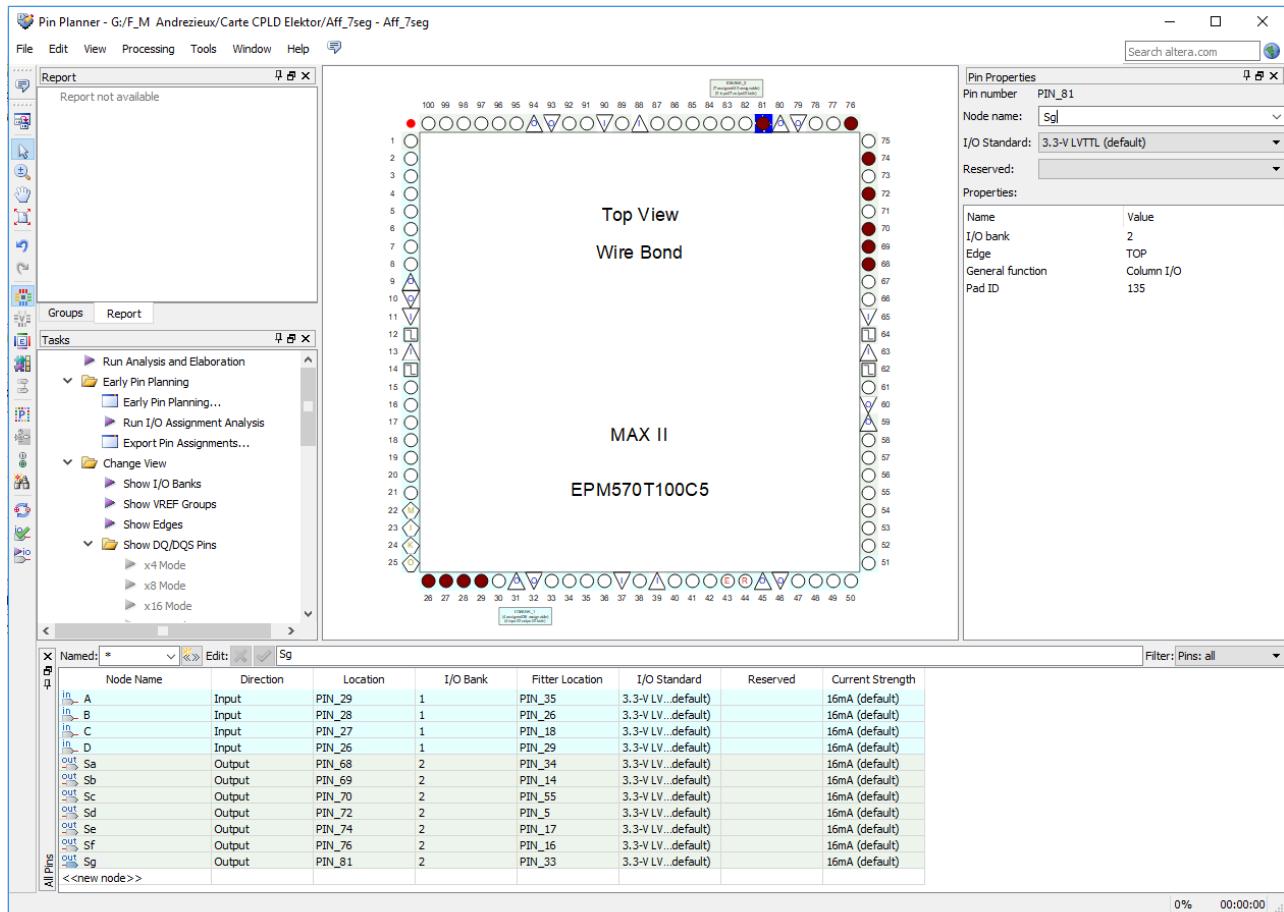
Les interrupteurs INT1, INT2, INT3 et INT4.		
29	IOB1-29	Lecture état de l'interrupteur INT1
28	IOB1-28	Lecture état de l'interrupteur INT2
27	IOB1-27	Lecture état de l'interrupteur INT3
26	IOB1-26	Lecture état de l'interrupteur INT4

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.

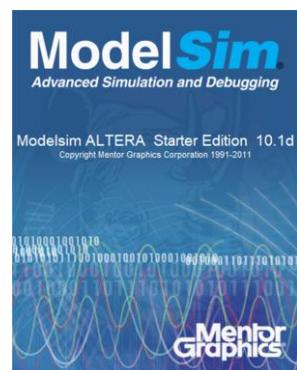
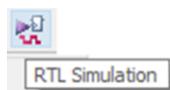


- Une fois l'affectation des broches réalisée, recompiler le projet en cliquant sur l'icône "Start Compilation".

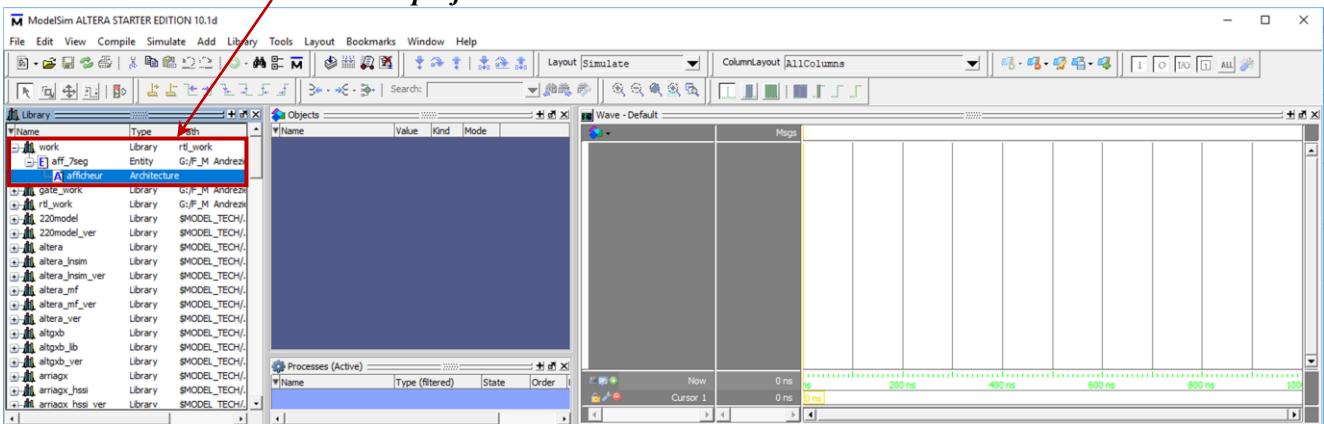


SIMULATION.

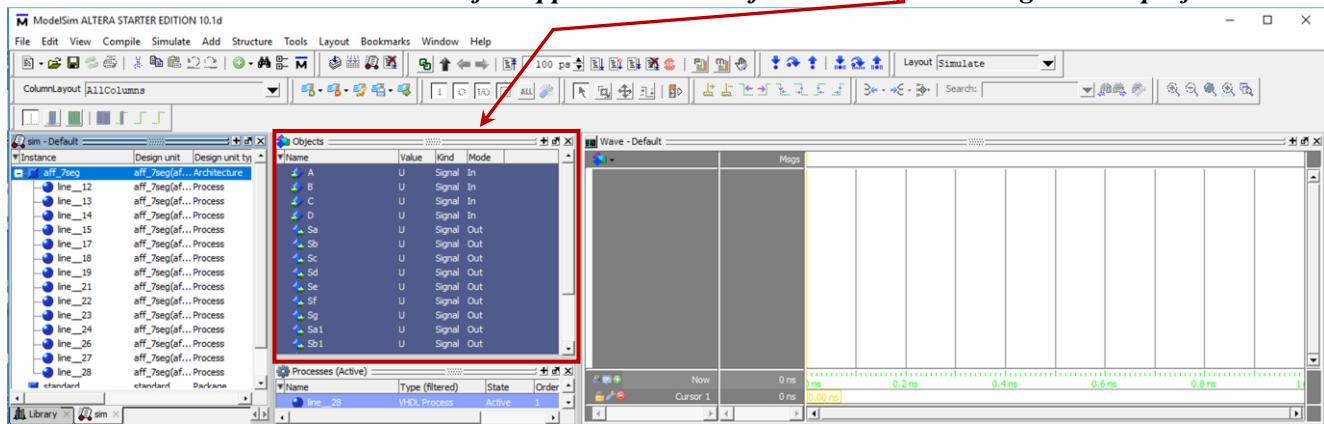
- ➊ Lancer le module de simulation en cliquant sur l'icône "RTL Simulation" dans la barre d'outils QUARTUS.



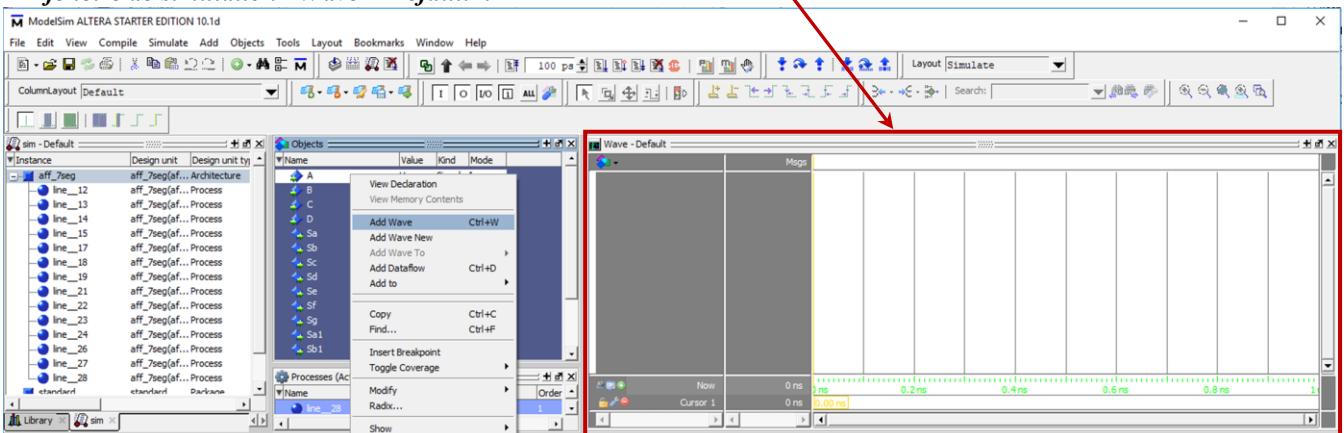
➋ Choisir l'architecture du projet dans le dossier "work".



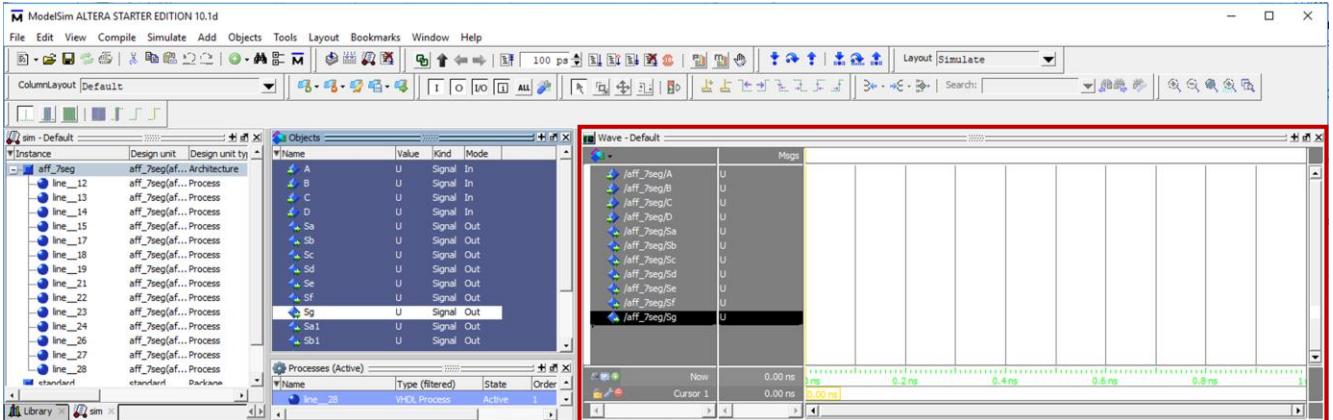
... Un double clic sur "Architecture" fait apparaître dans la fenêtre "Objets" les signaux du projet.



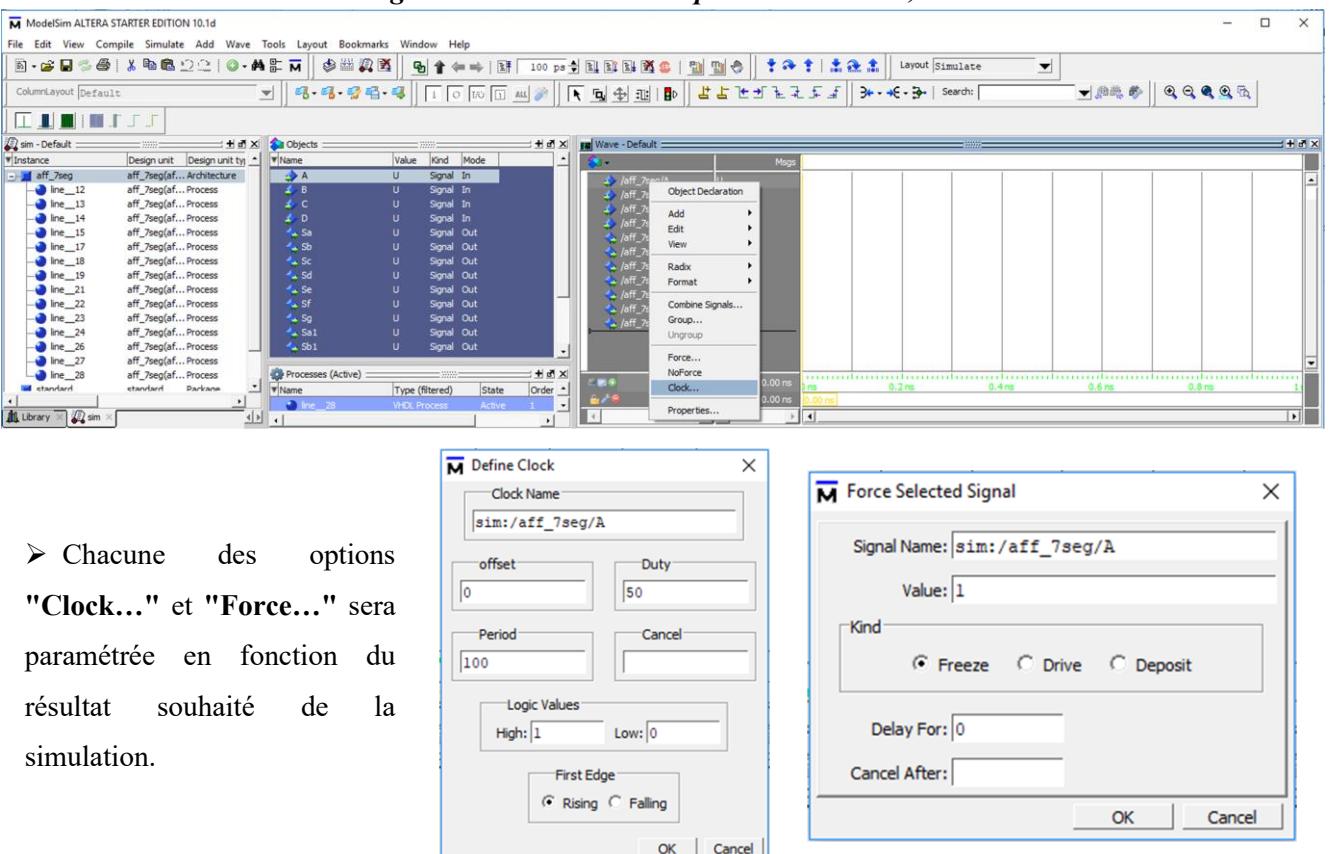
... Faire un clic droit sur chacun des signaux et choisir "Add Wave". Les signaux sont ainsi transférés dans la fenêtre de simulation "Wave - Default".



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

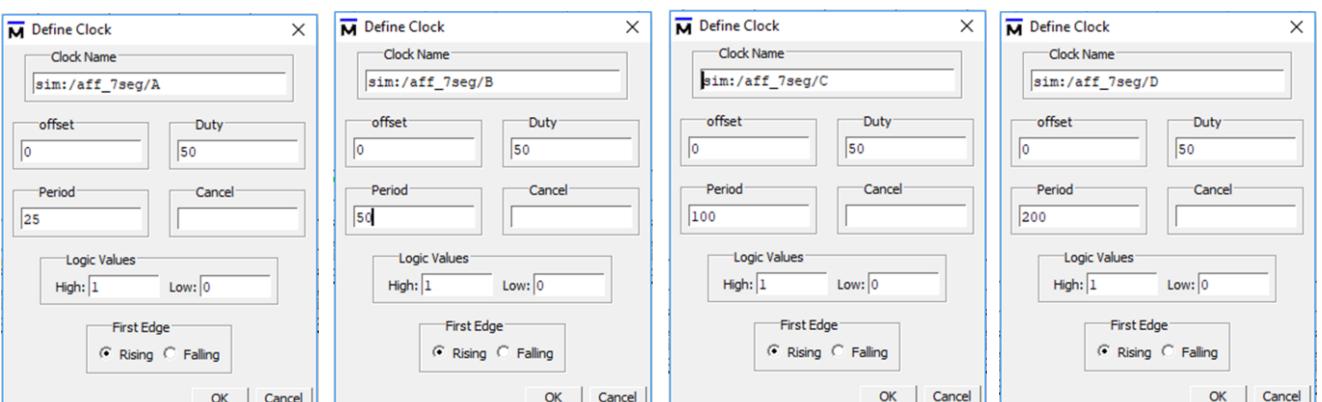


➤ Faire un clic droit sur un signal et choisir l'une des options "Force...", "NoForce" ou "Clock...".

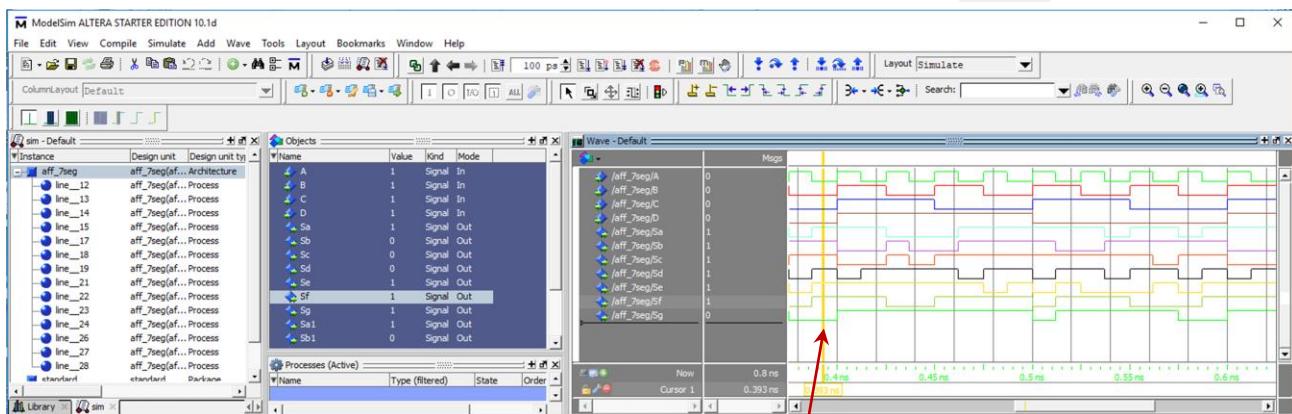
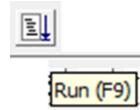


➤ Chacune des options "Clock..." et "Force..." sera paramétrée en fonction du résultat souhaité de la simulation.

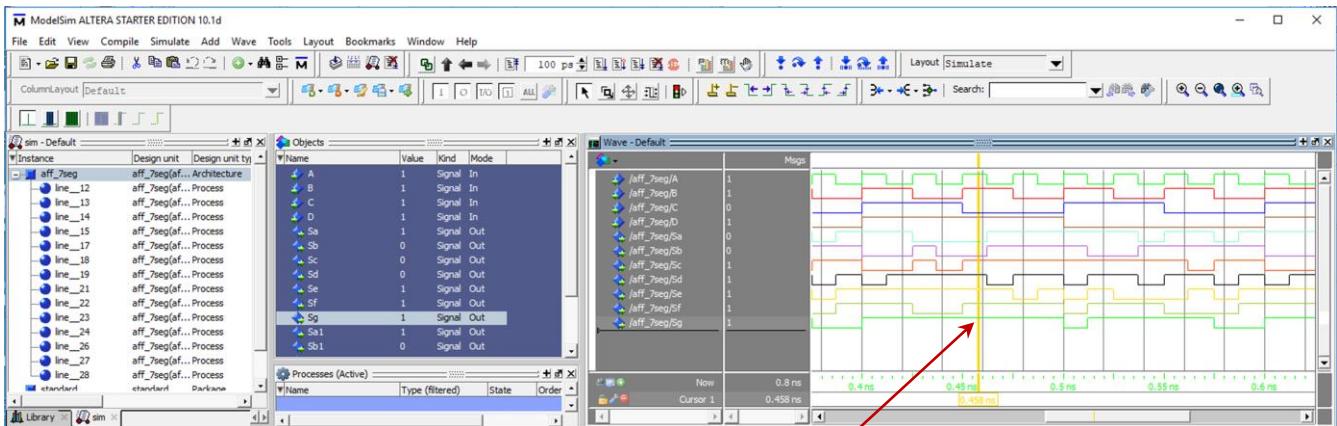
➤ Pour les signaux A, B, C et D du décodeur BCD-7Segments, on choisit l'option "Clock" avec le paramétrage suivant :



☞ 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).

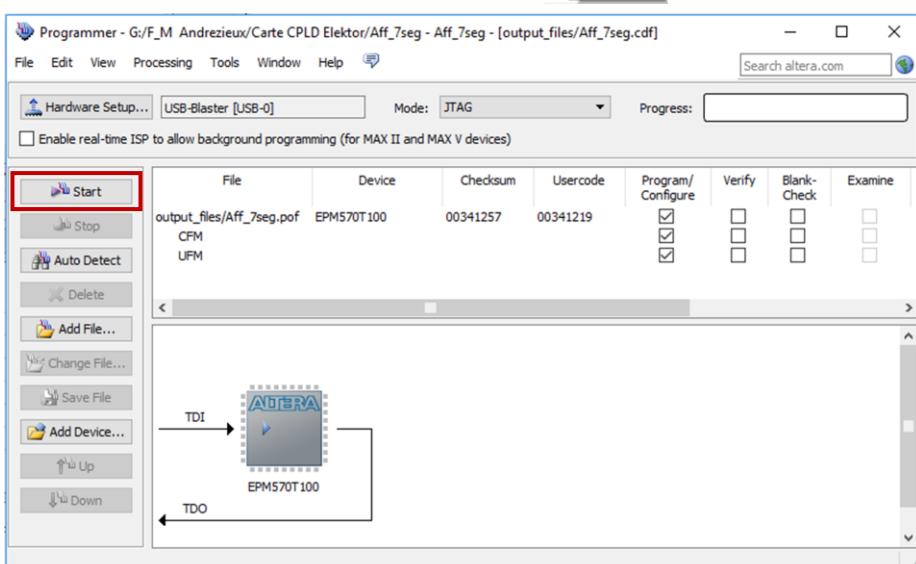
Tous les segments sont allumés sauf le "a" et le "b".

➤ PROGRAMMATION DE LA PUCE CPLD : EPM570T100C5.

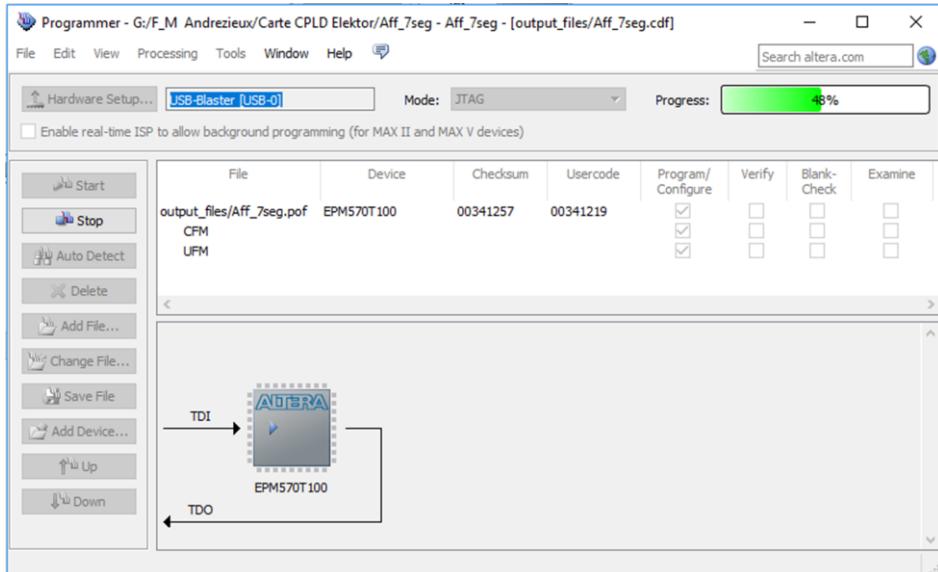
➤ Cliquer sur l'icône "Programmer" de la barre d'outils QUARTUS.



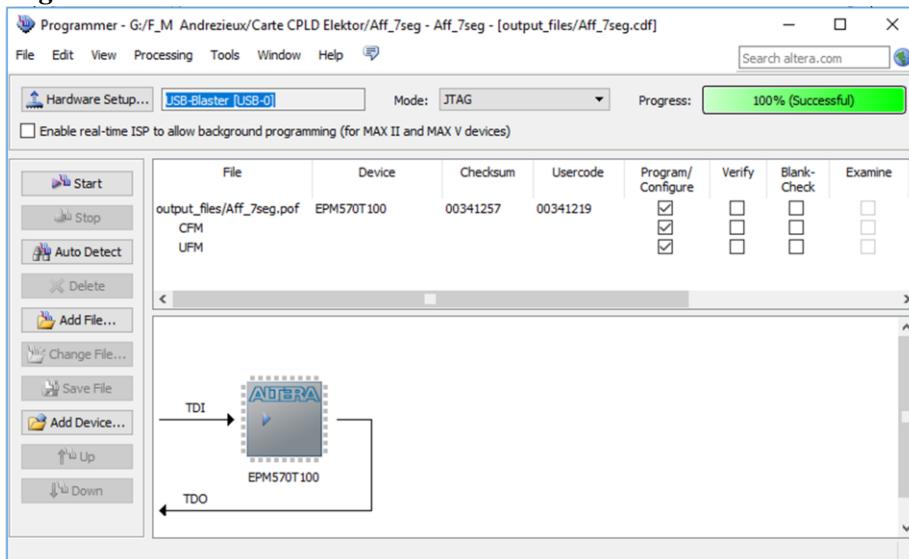
Cliquer ensuite sur "Start" pour lancer la programmation de la puce.



... 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 :

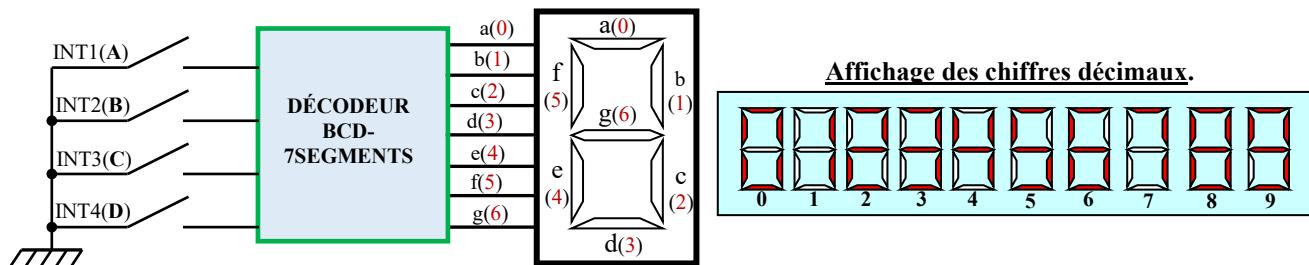
```
cible <= source1 when condition_booléenne1 else
    source2 when condition_booléenne2 else
    source3 when condition_booléenne3 else
    .....
    .....
    sourceN;
```

```

Quartus II 64-Bit - AffichageHexa
File Edit View Project Assignments Processing Tools Window Help
Project Navigator Entity MAX II: EPM570T100C5 AffichageHexa
File Design Units IP Components Tasks Compilation Report - AffichageHexa
Search altera.com
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 <= "111111" 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   "1111011" 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 <= "111111" 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   "1111011" 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.



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	1	0	0	0	0	0	0	0
	1	1	1	1	0	0	0	0	0	0	0

Équations simplifiées des segments :

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

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

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

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

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

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

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

Programme VHDL.

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Décodeur BCD 7-Segments
ENTITY AffichageDec3 IS
    PORT(A, B, C, D : IN STD_LOGIC; -- Les entrées du décodeur
          Sa, Sb, Sc, Sd, Se, Sf, Sg : OUT STD_LOGIC; -- Les segments de l'afficheur 1
          Sal, Sbl, Scl, Sdl, Sel, Sfl, Sgl : OUT STD_LOGIC); -- Les segments de l'afficheur 3
END AffichageDec3;

ARCHITECTURE Afficheur OF AffichageDec3 IS
BEGIN
    -- Equations des 7-segments de l'afficheur : Premier digit.
    Sa <= ((NOT(B OR C)) AND (NOT(A OR D))) OR ((B AND (NOT(C OR D))) AND (C AND NOT(D)) AND (A OR B));
    Sb <= (NOT(C OR D)) OR (NOT(B OR C)) OR (NOT(D) AND NOT(A XOR B));
    Sc <= (NOT(B OR C)) OR ((NOT(D)) AND (C OR (A AND B)));
    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)));
    Se <= (NOT(A)) AND ((NOT(C OR (B AND D))) OR (B AND (NOT(D))));
    Sf <= (NOT(A OR B OR C)) OR ((NOT(B OR C)) AND D) OR ((NOT(A AND B)) AND (C AND (NOT(D))));
    Sg <= (B AND (NOT(C OR D))) OR ((NOT(B)) AND (C XOR D)) OR (C AND (NOT(A OR D)));
    -- Equations des 7-segments de l'afficheur : Deuxième digit.
    Sal <= ((NOT(B OR C)) AND (NOT(A OR D))) OR ((C AND NOT(D)) AND (A OR B));
    Sbl <= (NOT(C OR D)) OR (NOT(B OR C)) OR (NOT(D) AND NOT(A XOR B));
    Scl <= (NOT(B OR C)) OR ((NOT(D)) AND (C OR (A AND B)));
    Sdl <= (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)));
    Sel <= (NOT(A)) AND ((NOT(C OR (B AND D))) OR (B AND (NOT(D))));
    Sfl <= (NOT(A OR B OR C)) OR ((NOT(B OR C)) AND D) OR ((NOT(A AND B)) AND (C AND (NOT(D))));
    Sgl <= (B AND (NOT(C OR D))) OR ((NOT(B)) AND (C XOR D)) OR (C AND (NOT(A OR D)));
END Afficheur;

```

Named:	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_Sa	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 Signal...e output		16mA (default)
out_Sb	Output	PIN_69	2	PIN_69	3.3-V LV..default			16mA (default)
out_Sb1	Output	PIN_85	2	PIN_85	3.3-V LV..default			16mA (default)
out_Sc	Output	PIN_70	2	PIN_70	3.3-V LV..default			16mA (default)
out_Sc1	Output	PIN_87	2	PIN_87	3.3-V LV..default			16mA (default)
out_Sd	Output	PIN_72	2	PIN_72	3.3-V LV..default			16mA (default)
out_Sd1	Output	PIN_91	2	PIN_91	3.3-V LV..default			16mA (default)
out_Se	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_Sf	Output	PIN_76	2	PIN_76	3.3-V LV..default			16mA (default)
out_Sf1	Output	PIN_97	2	PIN_97	3.3-V LV..default			16mA (default)
out_Sg	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"

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Décodeur BCD 7-Segments
ENTITY AffichageDec2 IS
PORT( A : IN STD_LOGIC_VECTOR(3 DOWNTO 0); -- Les entrées du décodeur
      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 AffichageDec2;

ARCHITECTURE Afficheur OF AffichageDec2 IS
BEGIN
  -- Codage l'afficheur 1.
  AF1 <=
    "0111111" WHEN A = "0000" ELSE --Affichage du 0
    "0000110" WHEN A = "0001" ELSE --Affichage du 1
    "1011011" WHEN A = "0010" ELSE --Affichage du 2
    "1001111" WHEN A = "0011" ELSE --Affichage du 3
    "1100110" WHEN A = "0100" ELSE --Affichage du 4
    "1101101" WHEN A = "0101" ELSE --Affichage du 5
    "1111101" WHEN A = "0110" ELSE --Affichage du 6
    "0000111" WHEN A = "0111" ELSE --Affichage du 7
    "1111111" WHEN A = "1000" ELSE --Affichage du 8
    "1100111" WHEN A = "1001" ELSE --Affichage du 9
    "0000000"; -- Afficheur éteint.

  -- Codage de l'afficheur 2.
  AF2 <=
    "0111111" WHEN A = "0000" ELSE --Affichage du 0
    "0000110" WHEN A = "0001" ELSE --Affichage du 1
    "1011011" WHEN A = "0010" ELSE --Affichage du 2
    "1001111" WHEN A = "0011" ELSE --Affichage du 3
    "1100110" WHEN A = "0100" ELSE --Affichage du 4
    "1101101" WHEN A = "0101" ELSE --Affichage du 5
    "1111101" WHEN A = "0110" ELSE --Affichage du 6
    "0000111" WHEN A = "0111" ELSE --Affichage du 7
    "1111111" WHEN A = "1000" ELSE --Affichage du 8
    "1100111" WHEN A = "1001" ELSE --Affichage du 9
    "0000000"; -- Afficheur éteint.
END Afficheur;

```

The screenshot shows the Quartus II interface with the following components:

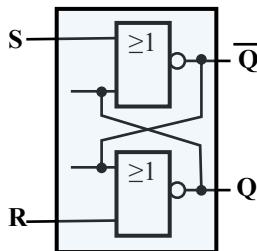
- Project Navigator:** Shows the MAX II: EPM570T100C5 device and the Entity AffichageDec2.
- Tasks:** A list of compilation tasks including Compile Design, Analysis & Synthesis, Filter (Place & Route), Assembler (Generate programming files), TimeQuest Timing Analysis, and EDA Netlist Writer.
- Code Editor:** Displays the VHDL code for the 7-segment decoder.
- Pin Assignment Table:** A table showing the mapping between node names (A[3] to AF2[0]) and physical pins (PIN_26 to PIN_83). The table includes columns for Node Name, Direction, Location, I/O Bank, Filter Location, I/O Standard, Reserved, and Current Strength.

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

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

LES BASCULES – LES REGISTRES – LES COMPTEURS : Exemples.

1- LA BASCULE R-S : Bascule asynchrone.

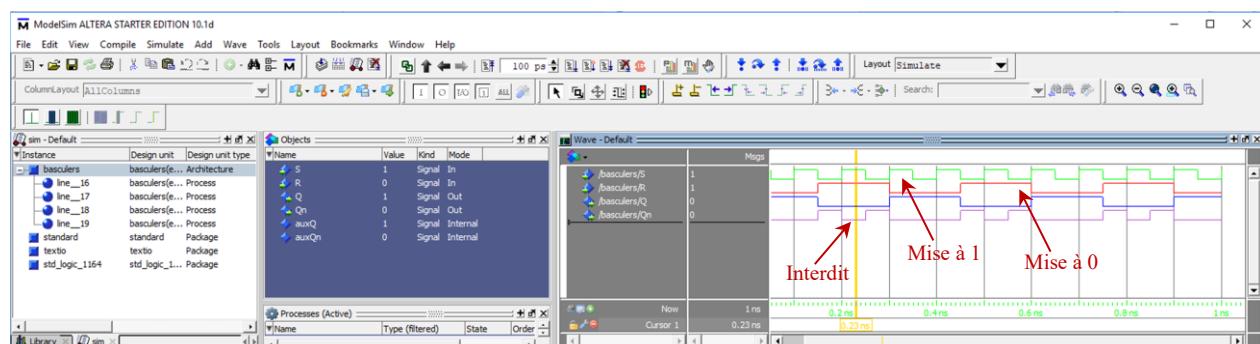
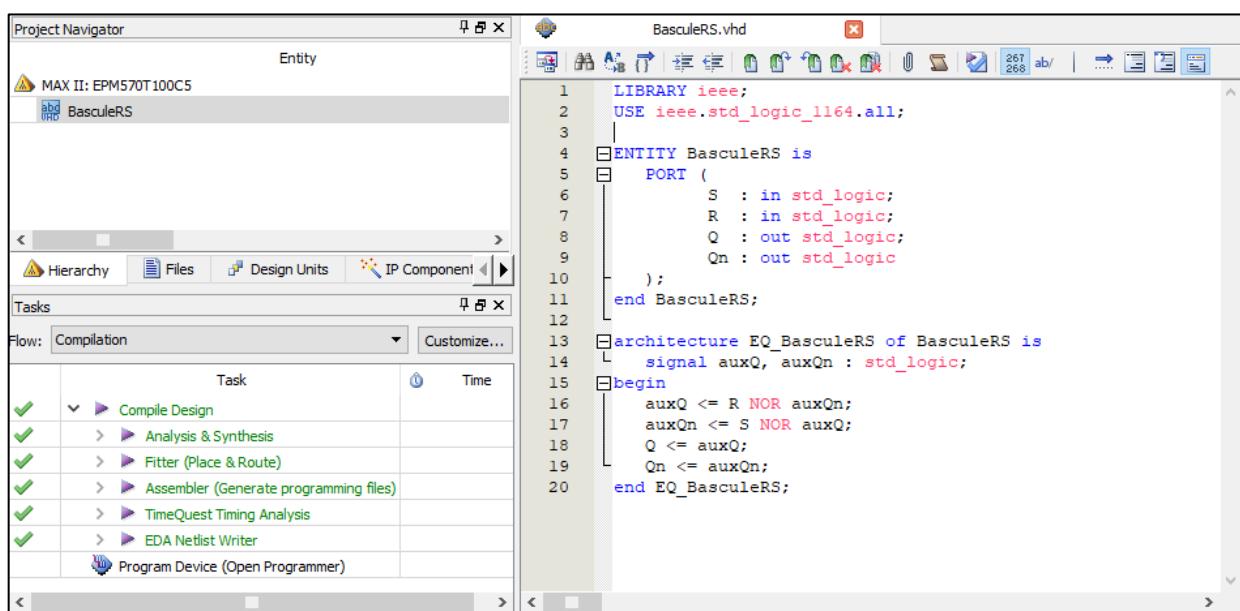


➤ Équations des sorties Q et \bar{Q} en fonction des entrées S et R.

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

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



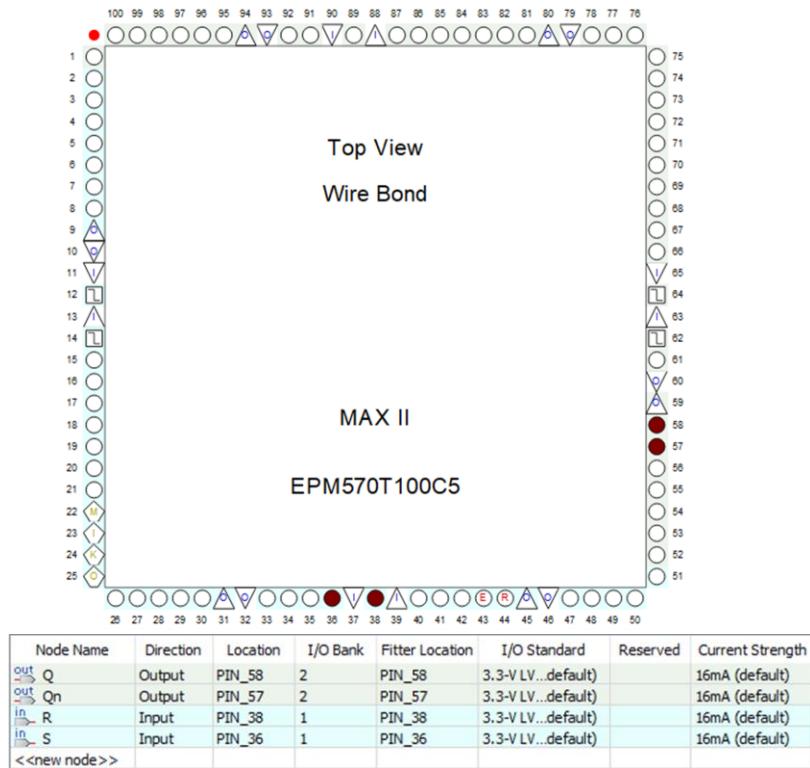
Affectation des broches.

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

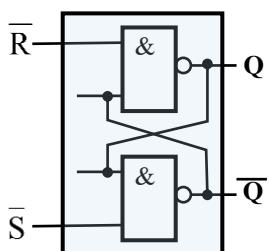
Broche N°38	IOB1-38	Lecture état du bouton poussoir BP1	R (Reset)
Broche N°36	IOB1-36	Lecture état du bouton poussoir BP2	S (Set)

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



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



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

- $Q = \overline{\bar{R} \cdot \bar{Q}}$ (1)
- $\bar{Q} = \overline{S \cdot Q}$ (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
abd BasculeNRNS

Tasks
Flow: Compilation
Task Time
Compile Design 00:00:11
Analysis & Synthesis 00:00:03
Filter (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)

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY BasculeNRNS IS
PORT (
    Sn : IN std_logic;
    Rn : IN std_logic;
    Q : OUT std_logic;
    Qn : OUT std_logic
);
END BasculeNRNS;

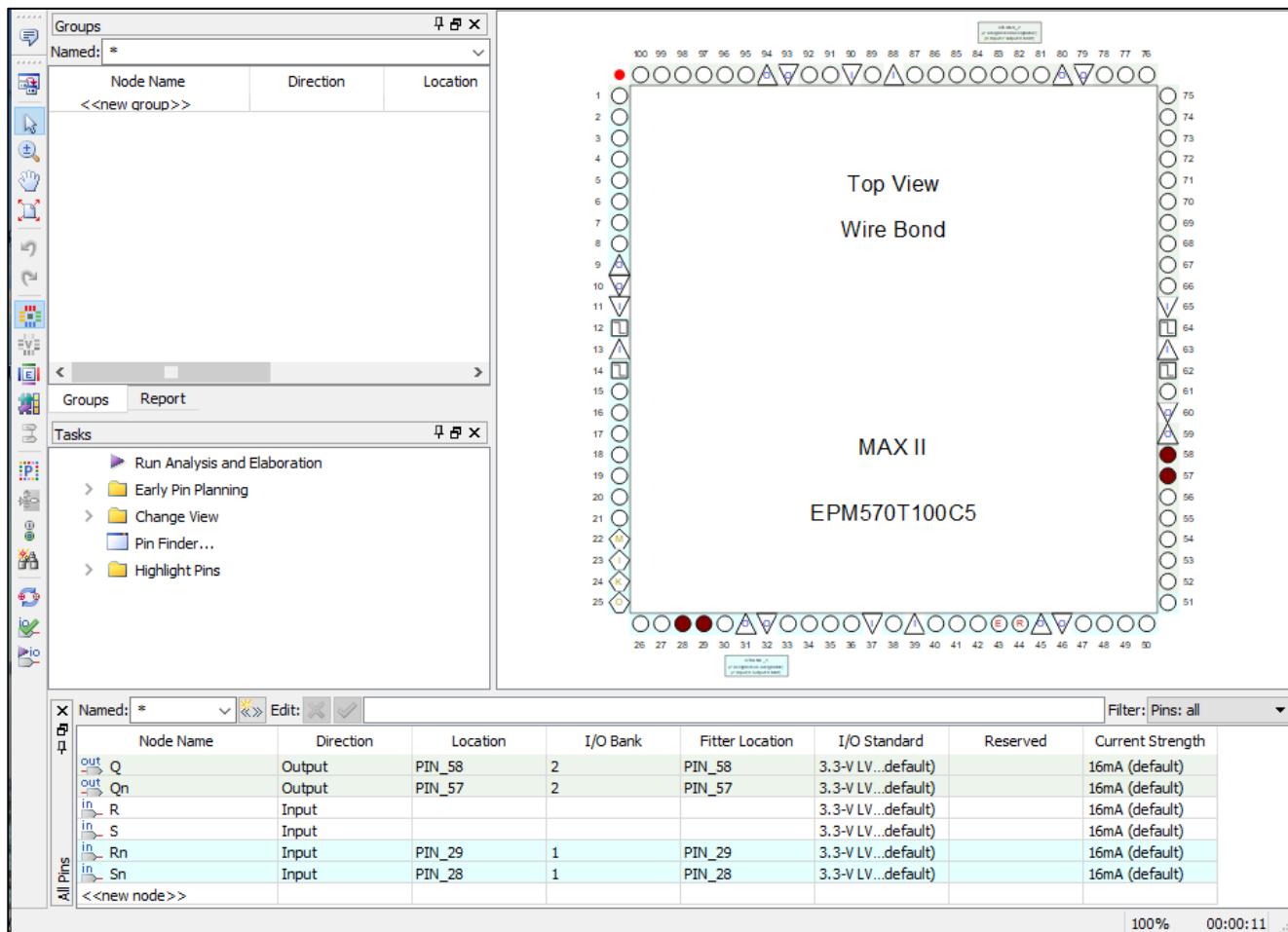
ARCHITECTURE EQ_BasculeNRNS OF BasculeNRNS IS
SIGNAL auxQ, auxQn : STD_LOGIC;
BEGIN
    auxQ <= Rn NAND auxQn;
    auxQn <= Sn NAND auxQ;
    Q <= auxQ;
    Qn <= auxQn;
END EQ_BasculeNRNS;

```

Affectation des broches.

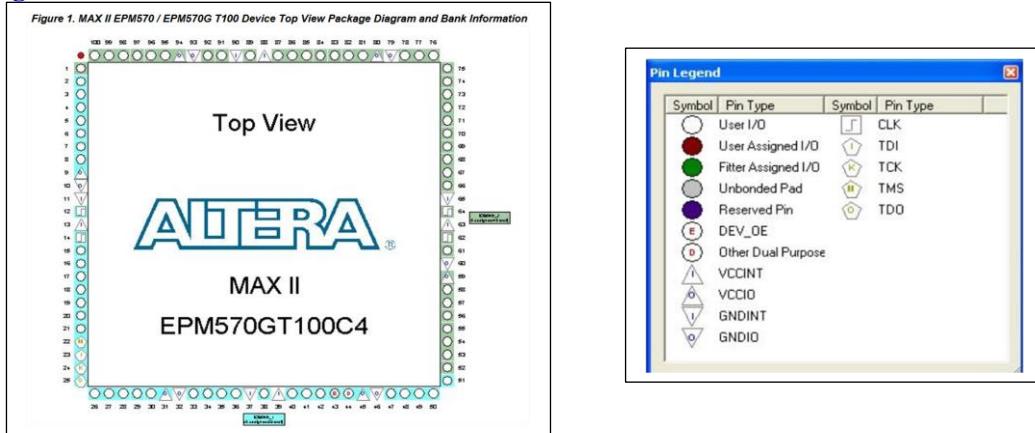
Les entrées R (reset) et S (set) de la bascule.			
Boche N°29	IOB1-29	Lecture état de l'interrupteur INT1	Rn (Reset)
Broche N°28	IOB1-28	Lecture état de l'interrupteur INT2	Sn (Set)

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 programme 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.

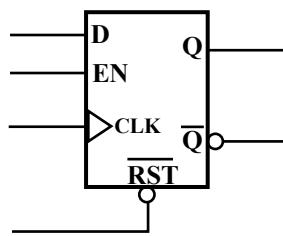


Table de vérité					
CLK	EN	RST	D	Q	\bar{Q}
X	0	0	X	0	1
X	1	0	X	0	1
X	0	1	X	Q_n	\bar{Q}_n
↑	1	1	0	0	1
↑	1	1	1	1	0

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

➤ Programme VHDL.

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY BasculeD IS
PORT (
    CLK : IN STD_LOGIC;
    EN : IN STD_LOGIC;
    RSTn : IN STD_LOGIC;
    D : IN STD_LOGIC;
    Q : OUT STD_LOGIC;
    Qn : OUT STD_LOGIC
);
END BasculeD;

ARCHITECTURE EQ_BasculeD OF BasculeD IS
BEGIN
    Bascule : PROCESS (CLK, EN, RSTn, D) IS
    BEGIN
        IF (RSTn = '0') THEN
            Q <= '0';
            Qn <= '1';
        ELSEIF rising_edge(CLK) THEN
            IF (EN = '1') THEN
                Q <= D;
                Qn <= NOT(D);
            END IF;
        END IF;
    END PROCESS Bascule;
END EQ_BasculeD;

```

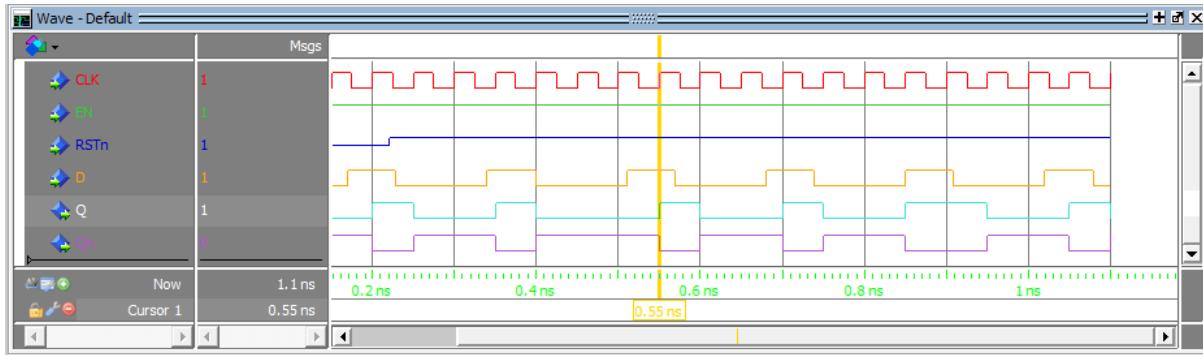
➤ Affectation des broches.

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 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

Pin assignment table								
	Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
All Pins	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>>							

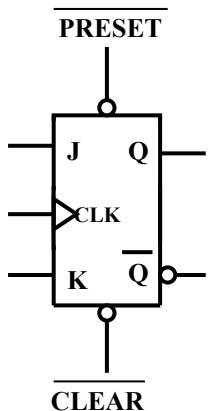
Résultat de la simulation.



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.

➤ Symbol et table de vérité.



CLK	J	K	<u>PRESET</u>	CLEAR	Q	<u>\bar{Q}</u>	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	\bar{Q}	Mémoire
L	X	X	1	1	Q	\bar{Q}	Mémoire
↑	0	0	1	1	Q	\bar{Q}	Mémoire
↑	1	0	1	1	1	0	Mise à 1
↑	0	1	1	1	0	1	Mise à 0
↑	1	1	1	1	\bar{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 PRESET active à l'état bas pour un forçage à "1" et une entrée CLEAR active à l'état bas pour un forçage à "0".

☺ Le programme VHDL.

The screenshot shows the Quartus II interface with the project "BasculeJK" open. The left pane displays the Project Navigator with the entity "BasculeJK" selected. The right pane shows the VHDL source code for the entity. The code defines a JK flip-flop with inputs CLK, PRE, CLR, J, K and outputs Q and Qn. It includes an architecture EQ_BasculeJK with a process that updates the outputs based on the input signals and the state of auxiliary signal auxQ.

```

LIBRARY IEEE;
USE ieee.std_logic_1164.all;

ENTITY BasculeJK IS
PORT ( CLK : IN std_logic;
        PRE : IN std_logic;
        CLR : IN std_logic;
        J   : IN std_logic;
        K   : IN std_logic;
        Q   : OUT std_logic;
        Qn  : OUT std_logic
      );
END BasculeJK;

ARCHITECTURE EQ_BasculeJK OF BasculeJK IS
signal auxQ : std_logic;
begin
    LesSignaux : process (CLK, PRE, CLR, J, K) is
    begin
        if ((PRE = '1') and (CLR = '0')) then
            Q <= '1';
            auxQ <= '1';
            Qn <= '0';
        elsif ((PRE = '0') and (CLR = '1')) then
            Q <= '0';
            auxQ <= '0';
            Qn <= '1';
        elsif ((PRE = '1') and (CLR = '1')) then
            if (rising_edge (CLK)) then
                if ((J = '1') and (K = '0')) then
                    Q <= '1';
                    auxQ <= '1';
                    Qn <= '0';
                elsif ((J = '0') and (K = '1')) then
                    Q <= '0';
                    auxQ <= '0';
                    Qn <= '1';
                elsif ((J = '1') and (K = '1')) then
                    Q <= not auxQ;
                    Qn <= auxQ;
                end if;
            end if;
        end process;
    end EQ_BasculeJK;

```

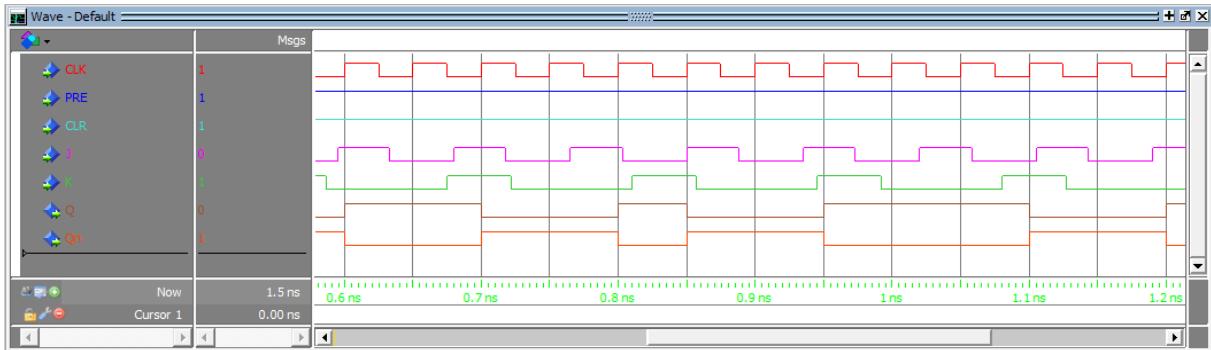
☛ Affectation des broches.

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 PRE de la bascule
Broche N°28	IOB1-28	Lecture état de l'interrupteur INT2	Entrée 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

All Pins	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.



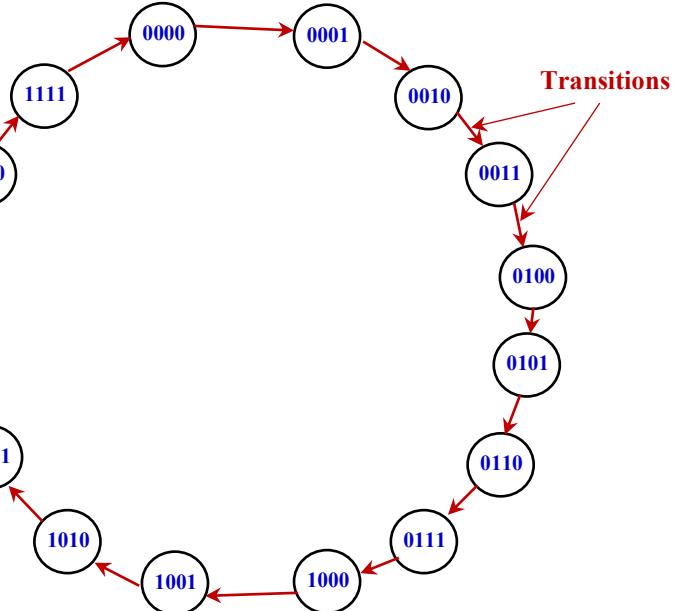
LES COMPTEURS.

1- Compteur synchrone à cycle complet Modulo-16.

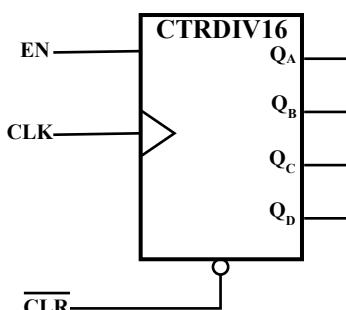
Il s'agit d'un compteur à $16 = 2^4$ états distincts. Il utilise 4 bascules synchrones A, B, C et D câblés en diviseur de fréquence par 2. Les sorties des 4 bascules sont Q_A , Q_B , Q_C et Q_D .

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.

The screenshot shows the Quartus II interface with the project "CompteurMod16". The left pane displays the Project Navigator with the entity "CompteurMod16" selected. The right pane shows the VHDL code for the entity:

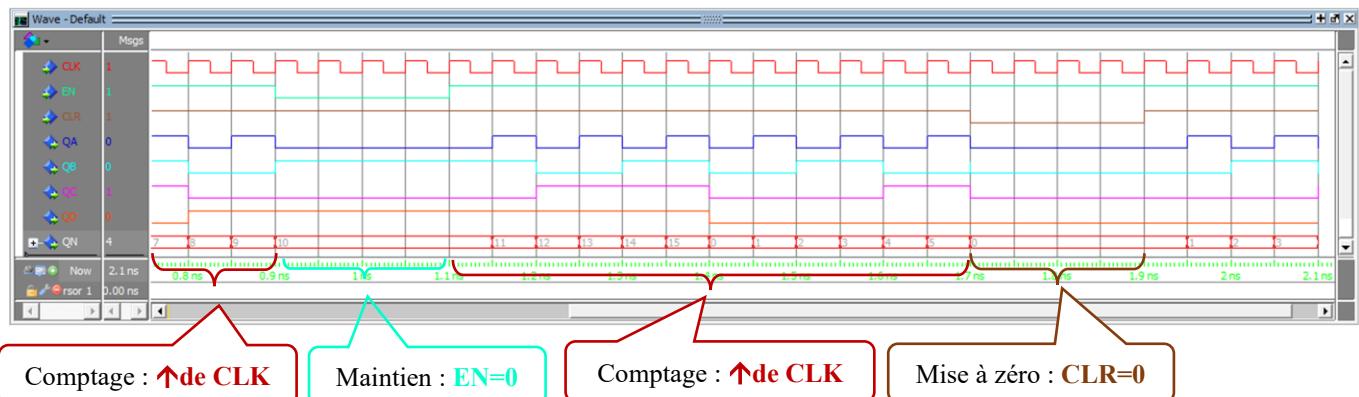
```

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 (
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 CompteurMod16;
14
15 ARCHITECTURE Comptage of CompteurMod16 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; -- Incrémentation du compteur
26 end if;
27 end process;
28 QN <= aux;
29 end Comptage;
30 
```

☞ Affectation des broches.

Node Name	Direction	Location	I/O Bank	Filter 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.

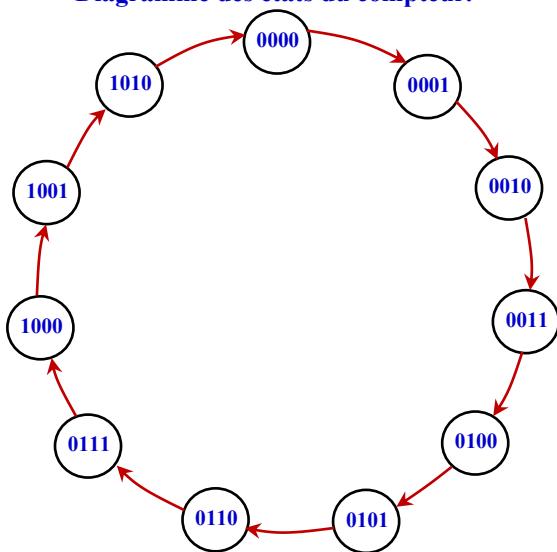


2- Compteur synchrone à cycle incomplet : compteur Modulo 11.

☞ Compteur à 11 états distincts.

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.



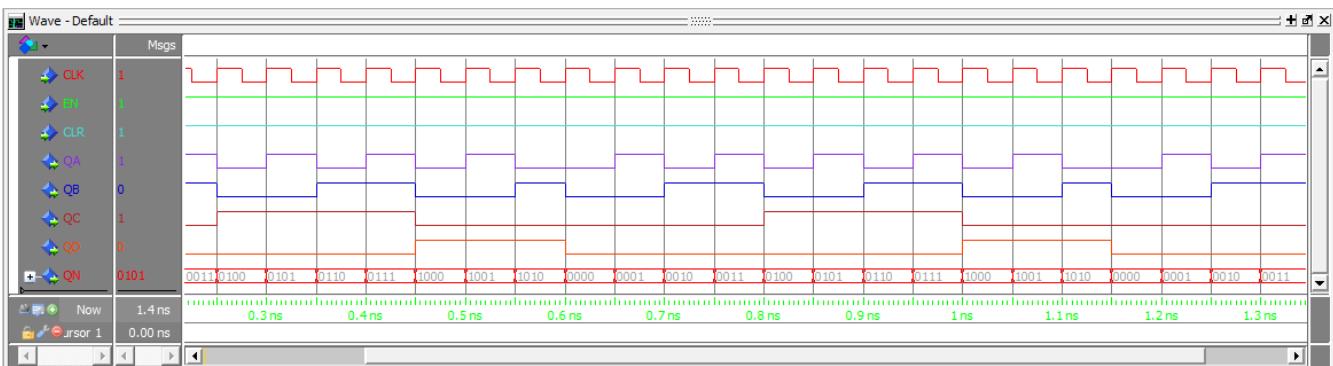
☞ Programme VHDL.

```

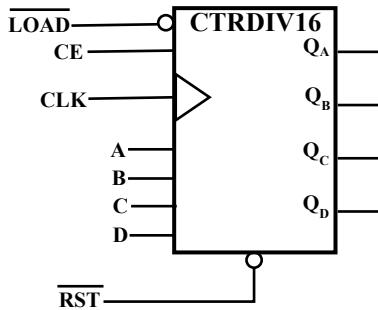
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 USE ieee.std_logic_arith.all;
4 USE ieee.std_logic_unsigned.all;
5
6 ENTITY CompteurMod11 IS
7 PORT (
8   CLK : IN std_logic; -- Signal d'horloge
9   EN : IN std_logic; -- Validation du comptage
10  CLR : IN std_logic; -- Mise à zéro du compteur
11  QN : 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; -- Incrémentation du compteur
26 if (aux = "1010") then -- Si valeur 10 atteinte
27 aux <= (others => '0'); -- alors mise à zéro
28 end if;
29 end process;
30 QN <= aux;
31 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

The screenshot shows the Quartus II interface with the project "CompteurChangement". The VHDL code is as follows:

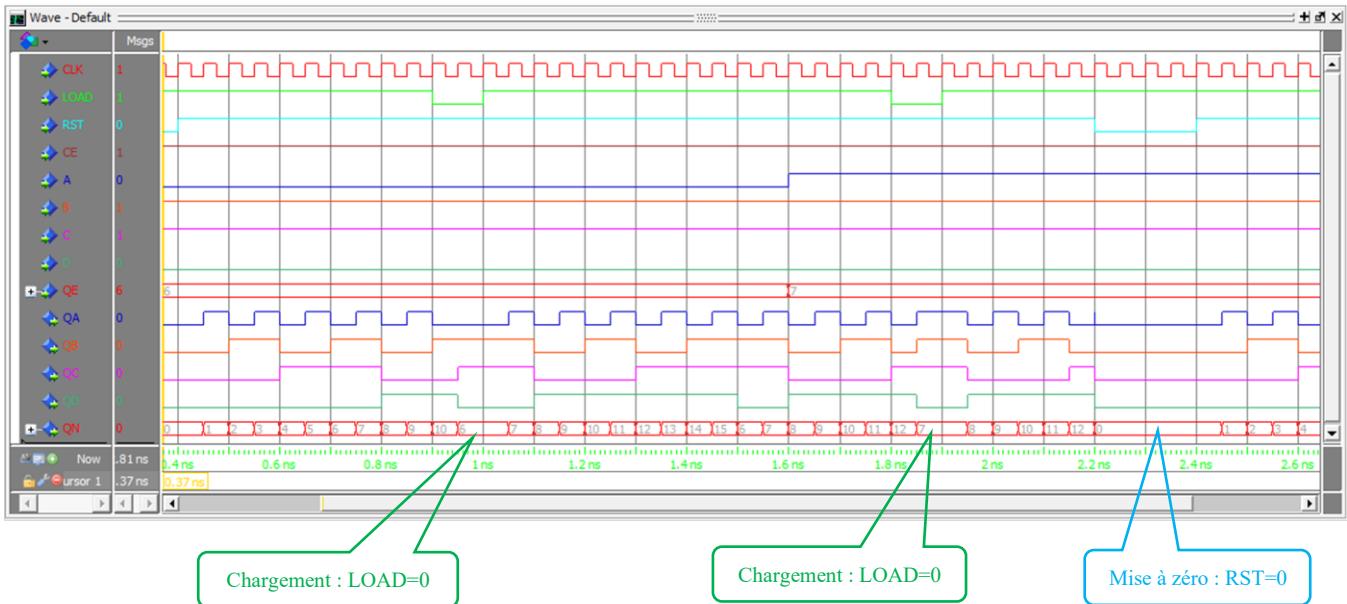
```

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 CompteurChangement 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    QE : IN std_logic_vector(3 DOWNTO 0); -- Données de chargement
13    QN : OUT std_logic_vector(3 DOWNTO 0) -- Sorties du compteur
14 );
15
16 ARCHITECTURE Comptage OF CompteurChangement 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                 -- Si aux = "1111" alors aux <= (OTHERS => '0');
28                 IF (aux = "1111") THEN
29                     aux <= QE; -- Fin du cycle on recharge
30                 ELSE aux <= aux + 1; -- Si pas fin de cycle on incrémente
31                 END IF;
32             END IF;
33         END IF;
34     END PROCESS;
35     QN <= aux;
36 END Comptage;
    
```

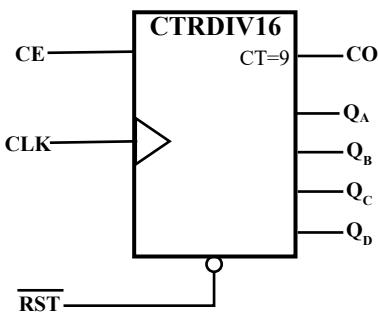
Affectation des broches.

Named:	*	Edit:	Filter: Pins: all				
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in CE	Input	PIN_36	1	PIN_36	3.3-V LV...default)		16mA (default)
in CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
in LOAD	Input	PIN_38	1	PIN_38	3.3-V LV...default)		16mA (default)
in QE[3]	Input	PIN_26	1	PIN_26	3.3-V LV...default)		16mA (default)
in QE[2]	Input	PIN_27	1	PIN_27	3.3-V LV...default)		16mA (default)
in QE[1]	Input	PIN_28	1	PIN_28	3.3-V LV...default)		16mA (default)
in QE[0]	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)
in RST	Input	PIN_34	1	PIN_34	3.3-V LV...default)		16mA (default)

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 $Q_A = Q_B = Q_C = Q_D = 0$.
- **RST** = 0 : Mise à 0 du compteur $Q_A = Q_B = Q_C = Q_D = 0$.

Programme VHDL.

```

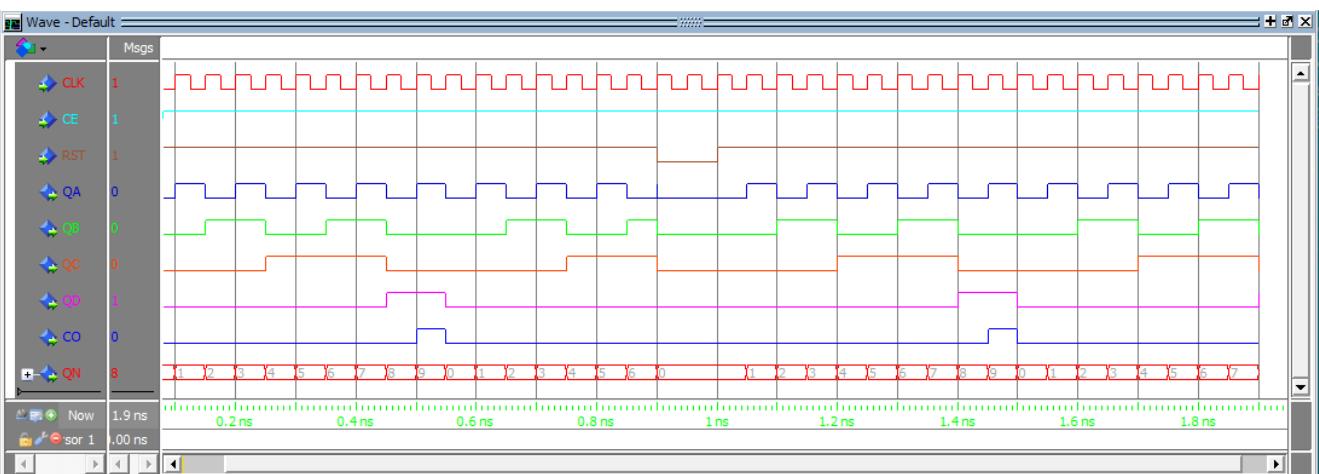
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 USE ieee.std_logic_arith.all;
4 USE ieee.std_logic_unsigned.all;
5
6 ENTITY CompteurMod10RCO IS
7 PORT ( CLK : IN std_logic; --Signal d'horloge
8        CE : IN std_logic; --Entrée de validation du compteur
9        RST : IN std_logic; -- Entrée de mise 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 if;
32   end process; -- Fin cycle de comptage
33   QN <= aux;
34   process (aux, CE) -- Ce processus génère le signal de fin de cycle: Carry Out (CO)
35   begin
36     if (aux = "1001") then -- Si Q0=0; Q1=0; Q2=0; Q3=1
37       CO <= '1'; -- CO = 1
38     else
39       CO <= '0'; -- Sinon
40     end if;
41   end process;
42 end Comptage;

```

Affectation des broches.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength
in CE	Input	PIN_38	1	PIN_38	3.3-V LV...default)		16mA (default)
in CLK	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
out CO	Output	PIN_49	1	PIN_49	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_36	1	PIN_36	3.3-V LV...default)		16mA (default)

Résultat de la simulation.

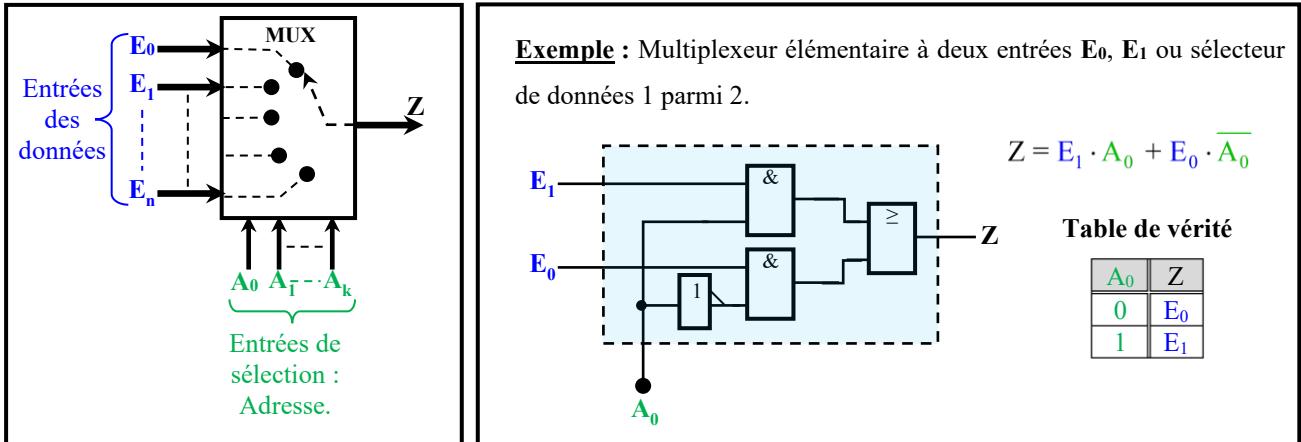


LES MULTIPLEXEURS – Exemples.

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**.



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 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.

$$Z = E_0 \cdot \bar{A}_0 \cdot \bar{A}_1 + E_1 \cdot A_0 \cdot \bar{A}_1 + E_2 \cdot \bar{A}_0 \cdot A_1 + E_3 \cdot A_0 \cdot A_1$$

Programme VHDL : Multiplexeur à 2 entrées.

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Mux1Parmi2 IS
PORT ( E0 : IN std_logic; --Entrée de donnée E0
       E1 : IN std_logic; --Entrée de donnée E1
       A0 : IN std_logic; --Sélection A0
       Z : OUT std_logic --Sortie du multiplexeur
      );
END Mux1Parmi2;

ARCHITECTURE Multiplexeur OF Mux1Parmi2 IS
Begin
  Z <= (E0 and (not(A0))) or (E1 and A0);
end Multiplexeur;
  
```

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

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Mux1Parmi2When IS
PORT ( E0 : IN std_logic; -- Entrée de données E0
       E1 : IN std_logic; -- Entrée de données E1
       A0 : IN std_logic; -- Sélection A0
       Z : OUT std_logic -- Sortie du multiplexeur
);
END Mux1Parmi2When;

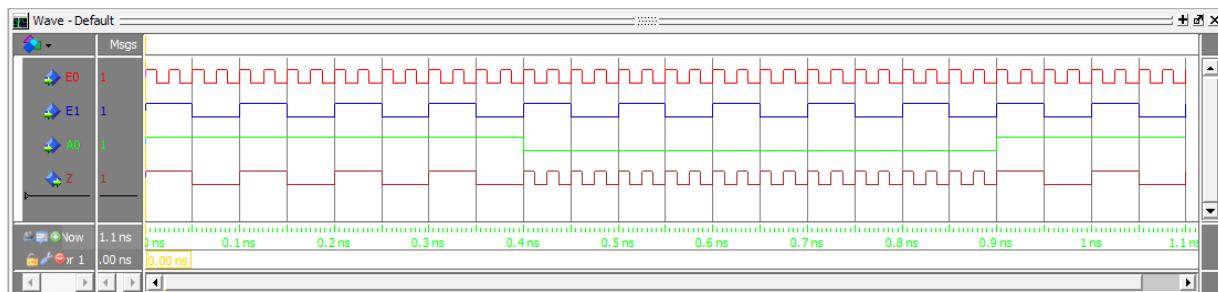
ARCHITECTURE Multiplexeur OF Mux1Parmi2When IS
begin -- Equation du multiplexeur |
Z <= E0 when A0 = '0' else E1;
end Multiplexeur;

```

Affectation des broches.

All Pins	Named: *	Edit:	Filter: Pins: all					
	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	E0	Input	PIN_38	1	PIN_38	3.3-V LV...default)		16mA (default)
in	E1	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
out	Z	Output	PIN_58	2	PIN_58	3.3-V LV...default)		16mA (default)

Résultat de la simulation.



☒ Programme VHDL : Multiplexeur à 4 entrées de données.

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Mux1Parmi4 IS
PORT ( E0 : IN std_logic; -- Entrée de donnée E0
       E1 : IN std_logic; -- Entrée de donnée E1
       E2 : IN std_logic; -- Entrée de donnée E2
       E3 : IN std_logic; -- Entrée de donnée E3
       A0 : IN std_logic; -- Sélection: Adresse A0
       A1 : IN std_logic; -- Sélection: Adresse A1
       Z : OUT std_logic
);
END Mux1Parmi4;

ARCHITECTURE Multiplexeur OF Mux1Parmi4 IS
begin -- Equation de la sortie Z du multiplexeur
Z <= (E0 and (not A0) and (not A1)) or (E1 and A0 and (not A1)) or
      (E2 and (not A0) and A1) or (E3 and A0 and A1);
end Multiplexeur;

```

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

```

Quartus II 64-Bit - G:/F_M Andrezieux/Carte CPLD Elektor/Mux1Parmi4When/Mux1Parmi4When - Mux1Parmi4When

File Edit View Project Assignments Processing Tools Window Help
Search altera.com

Project Navigator Entity Mux1Parmi4When.vhd Compilation Report - Mux1Parmi4When.x
MAX II: EPM7070T100C5 Mux1Parmi4When

Tasks Flow: Compilation
Task 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 00:00:02
EDA Netlist Writer 00:00:02
Program Device (Open Programmer)

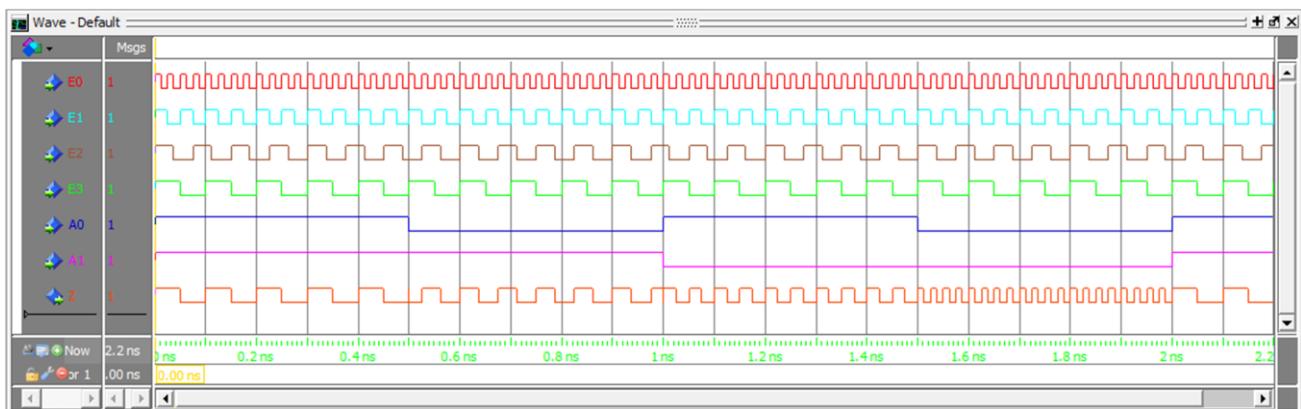
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY Mux1Parmi4When IS
PORT ( E0 : IN std_logic; --Entrée de donnée E0
       E1 : IN std_logic; --Entrée de donnée E1
       E2 : IN std_logic; --Entrée de donnée E2
       E3 : IN std_logic; --Entrée de donnée E3
       A : IN std_logic_vector(1 DOWNTO 0);--Les 2 bits d'adresse
       Z : OUT std_logic --Sortie du multiplexeur
      );
END Mux1Parmi4When;
ARCHITECTURE Multiplexeur OF Mux1Parmi4When IS
BEGIN
  Z <= E0 WHEN A = "00" ELSE
    E1 WHEN A = "01" ELSE
    E2 WHEN A = "10" ELSE
    E3;
END Multiplexeur;

```

Affectation des broches.

Named:	*	Edit:	Filter:	Pins: all			
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 E0	Input	PIN_38	1	PIN_38	3.3-V LV...default)		16mA (default)
in E1	Input	PIN_12	1	PIN_12	3.3-V LV...default)		16mA (default)
in E2	Input	PIN_62	2	PIN_62	3.3-V LV...default)		16mA (default)
in E3	Input	PIN_36	1	PIN_36	3.3-V LV...default)		16mA (default)
out Z	Output	PIN_58	2	PIN_58	3.3-V LV...default)		16mA (default)

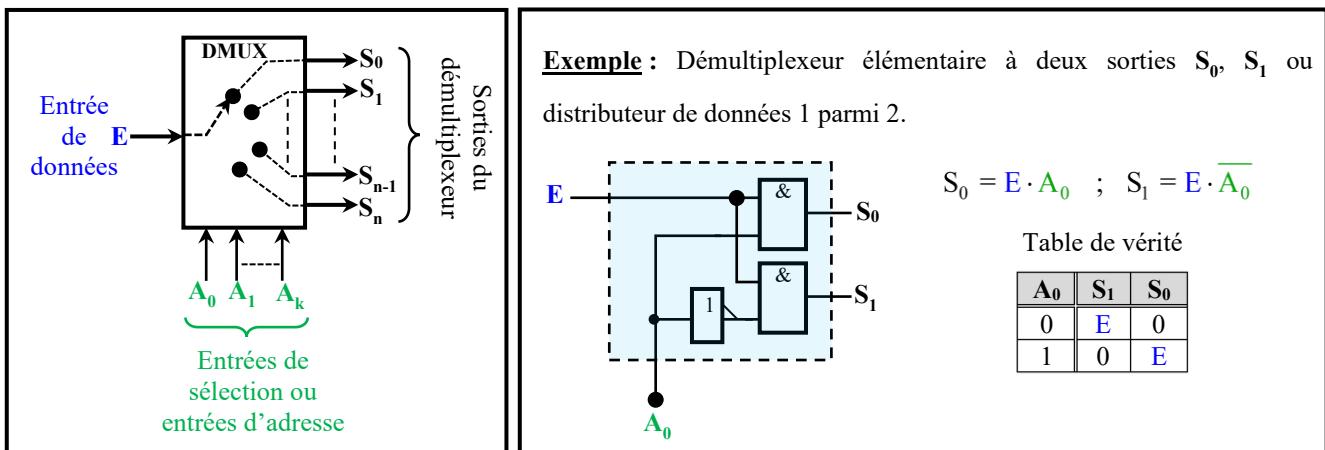
Résultat de la simulation.



LES DÉMULTIPLEXEURS – Exemples.

Un démultiplexeur (**DMUX**) ou distributeur de données est un circuit n'ayant qu'une entrée de donnée **E** et plusieurs sorties **S₀, S₁, S₂, S₃ ...**

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 **A₀, A₁, A₂, A₃ ...**



Autre exemple : Démultiplexeur à 4 sorties **S₀, S₁, S₂, S₃** ou distributeur de données 1 parmi 4.

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

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.

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

Programme VHDL : Démultiplexeur à 2 sorties.

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY DMux1Parmi2 IS
PORT ( E : IN std_logic; --Entrée de données
        A0 : IN std_logic; --Entrée de sélection: Adresse
        S0 : OUT std_logic; --Sortie S0 du démultiplexeur
        S1 : OUT std_logic --Sortie S1 du démultiplexeur
);
END DMux1Parmi2;

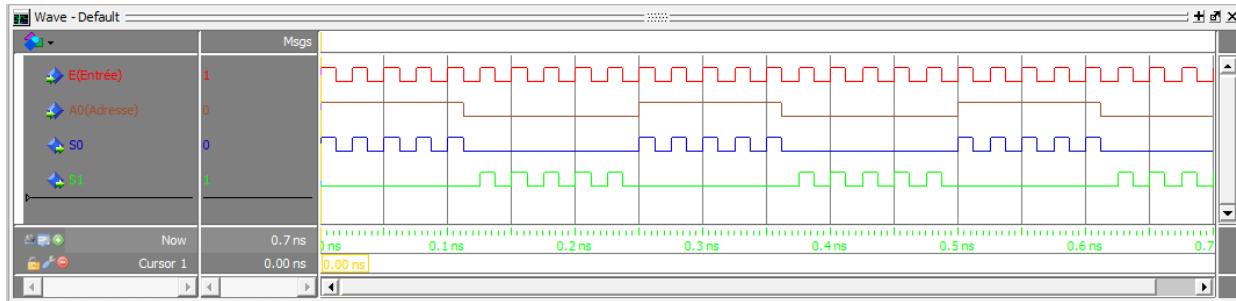
ARCHITECTURE Demultiplexeur OF DMux1Parmi2 IS
begin -- Les équations du démultiplexeur
    S0 <= E and A0;
    S1 <= E and (not A0);
end Demultiplexeur;

```

Affectation des broches.

All Pins	Named: *	Edit: <input type="button" value="X"/> <input type="button" value="Y"/> <input type="button" value="Z"/>	Filter: Pins: all					
	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.

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY DMux1Parmi4 IS
PORT (
    E : IN std_logic; -- Entrée de données
    A0 : IN std_logic; -- Entrée d'adresse: poids faible A0
    A1 : IN std_logic; -- Entrée d'adresse: poids fort A1
    S0 : OUT std_logic; -- Première sortie du démultiplexeur
    S1 : OUT std_logic; -- Deuxième sortie du démultiplexeur
    S2 : OUT std_logic; -- Troisième sortie du démultiplexeur
    S3 : OUT std_logic; -- Dernière sortie du démultiplexeur
);
END DMux1Parmi4;

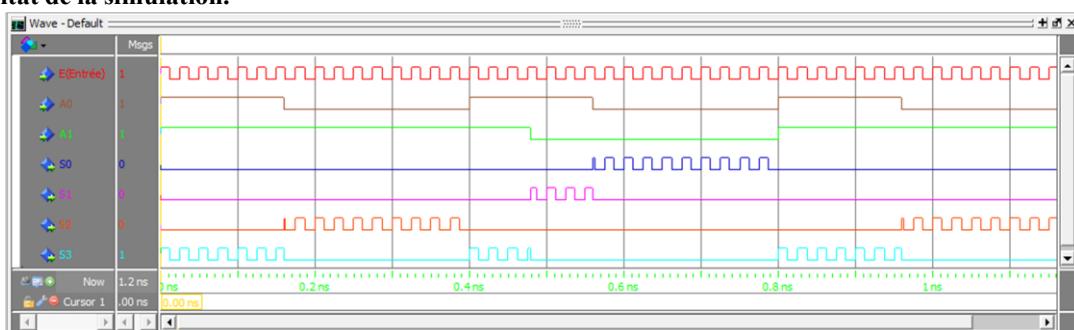
ARCHITECTURE Demultiplexeur OF DMux1Parmi4 IS
begin -- Les équations des sorties du démultiplexeur
    S0 <= (E and (not A0) and (not A1));
    S1 <= (E and A0 and (not A1));
    S2 <= (E and (not A0) and A1);
    S3 <= (E and A0 and A1);
end Demultiplexeur;

```

Affectation des broches.

All Pins	Named: *	Edit: <input type="button" value="X"/> <input type="button" value="Y"/> <input type="button" value="Z"/>	Filter: Pins: all					
	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;
```

```

Quartus II 64-Bit - DMux1Parmi4When

File Edit View Project Assignments Processing Tools Window Help
Search altera.com

Project Navigator Entity DMux1Parmi4When.vhd Compilation Report - DMux1Parmi4When
MAX II: EPM570T100C5
DMux1Parmi4When

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY DMux1Parmi4When IS
PORT (
    E : IN std_logic; --Entrée de données
    A : IN std_logic_vector(1 DOWNTO 0); --Les 2 bits d'adresse
    S0 : OUT std_logic; --Première sortie du démultiplexeur
    S1 : OUT std_logic; --Deuxième sortie du démultiplexeur
    S2 : OUT std_logic; --Troisième sortie du démultiplexeur
    S3 : OUT std_logic --Dernière sortie du démultiplexeur
);
END DMux1Parmi4When;

ARCHITECTURE Demultiplexeur OF DMux1Parmi4When IS
begin
    process(A) --Affectation des sorties
    begin
        if (A = "00") then
            S0 <= E;
            S1 <= '0';
            S2 <= '0';
            S3 <= '0';
        elsif (A = "01") then
            S0 <= '0';
            S1 <= E;
            S2 <= '0';
            S3 <= '0';
        elsif (A = "10") then
            S0 <= '0';
            S1 <= '0';
            S2 <= E;
            S3 <= '0';
        else
            S0 <= '0';
            S1 <= '0';
            S2 <= '0';
            S3 <= E;
        end if;
    end process;
end Demultiplexeur;

```

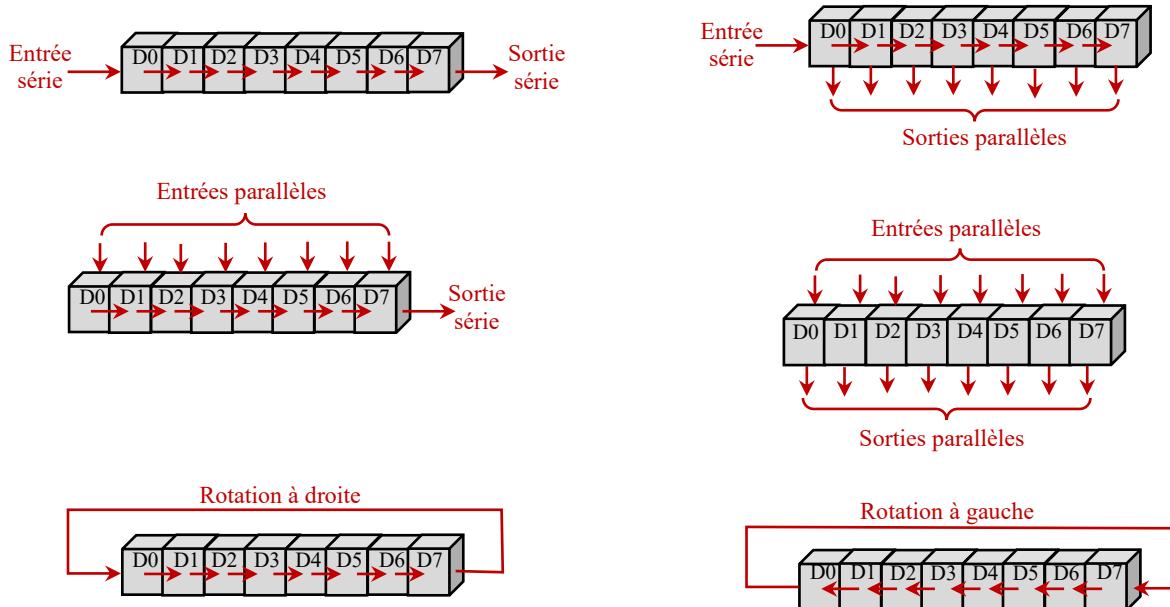
LE REGISTRE A DÉCALAGE.

Un registre à décalage est une association 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.

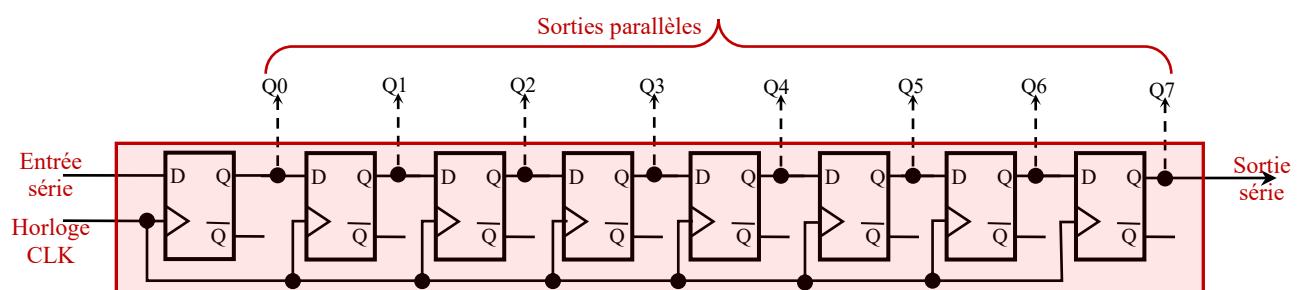
La capacité de stockage d'un registre est le nombre total de bits de données qu'il peut emmagasiner. Chaque étage (ou bascule) représente une capacité de stockage de 1 bit. Un registre à décalage 8 bits est donc une association série de 8 bascules.

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.

```

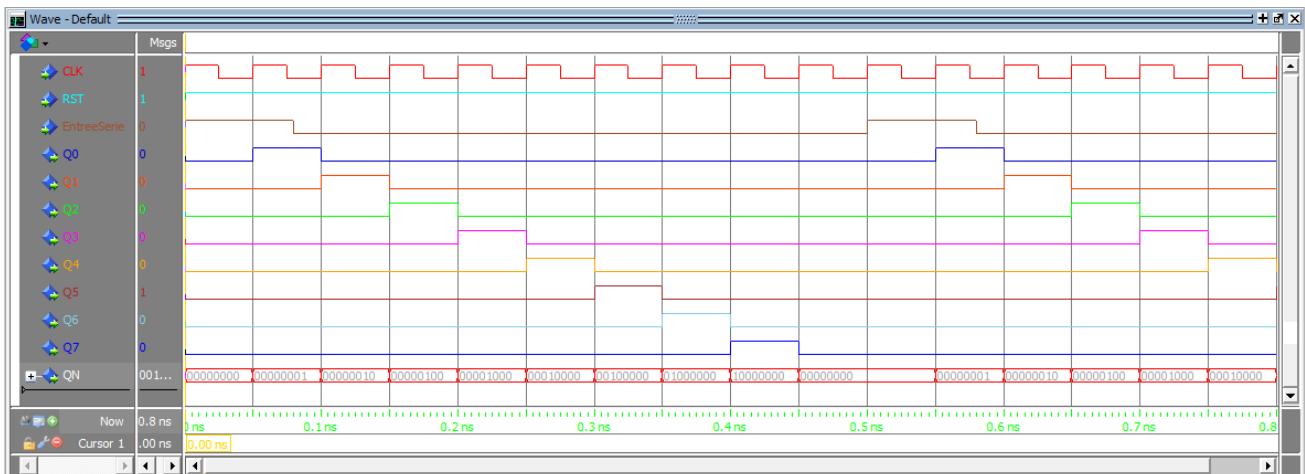
Quartus II 64-Bit - RegistreDecalage8bits
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
Project Navigator Entity MAX II: EPM570T100C5
RegistreDecalage8bits
RegistreDecalage8bits.vhd Compilation Report - RegistreDecalage8bits
Tasks Flow: Compilation
Task Time
Compile 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 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 process;
25 Charge : process(aux)
26 begin
27 QN <= aux;
28 end process;
29 end Registre;
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 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.

The screenshot shows the Quartus II interface with the project "RegistreDecalageSerSer8bits" open. The left pane displays the Project Navigator and Task list, both showing successful compilation. The right pane shows the VHDL source code for the entity and architecture of the register.

```

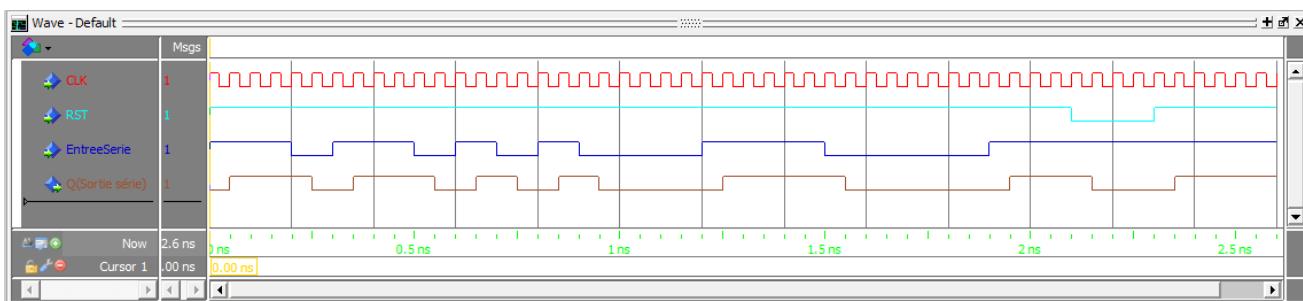
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 USE ieee.numeric_std.all;
4
5 ENTITY RegistreDecalageSerSer8bits IS
6 PORT ( CLK : IN std_logic; -- Signal d'horloge
7        EntreeSerie : IN std_logic; -- Entrée série
8        RST : IN std_logic; -- Mise à zéro du registre
9        Q : OUT std_logic -- 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 process;
25 process(aux)
26 begin
27   Q <= aux;
28 end process;
29 end Registre;
30

```

☞ Affectation des broches.

Named:	*	Edit:	Filter:	Pins: all			
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 :

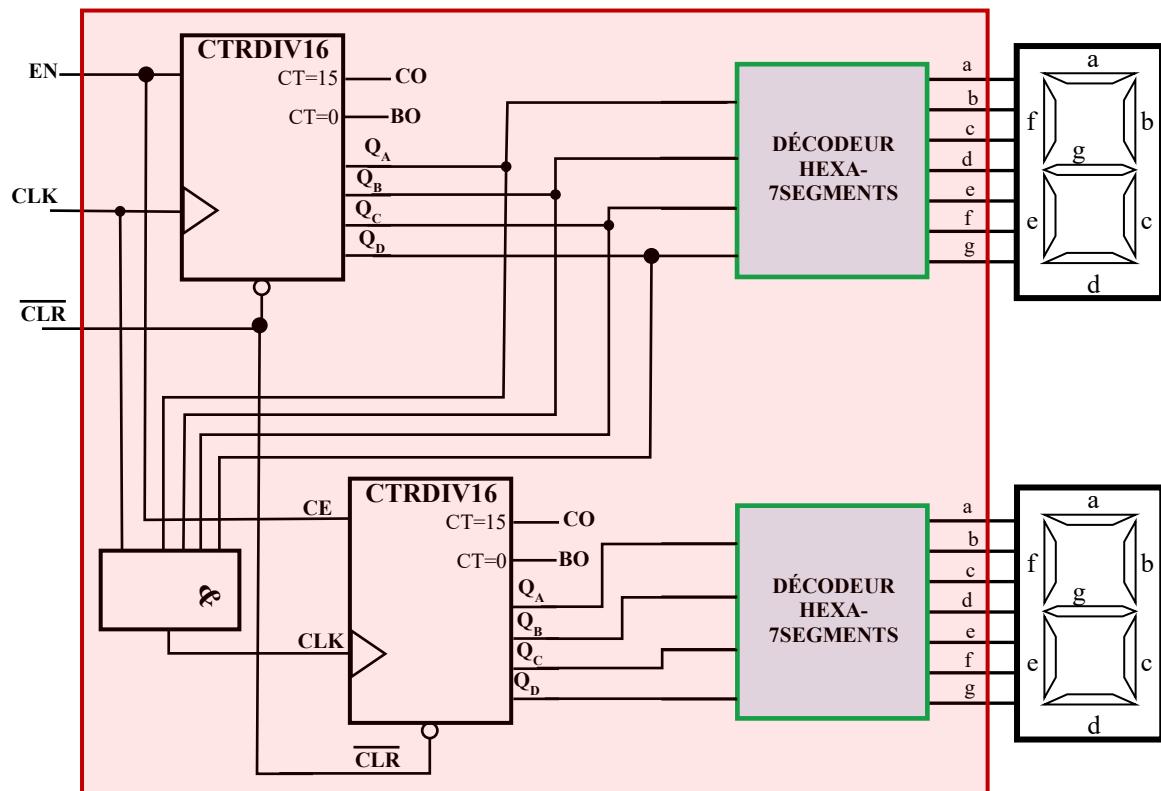
https://fr.wikibooks.org/wiki/TD3_VHDL_Compteurs_et_registres

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

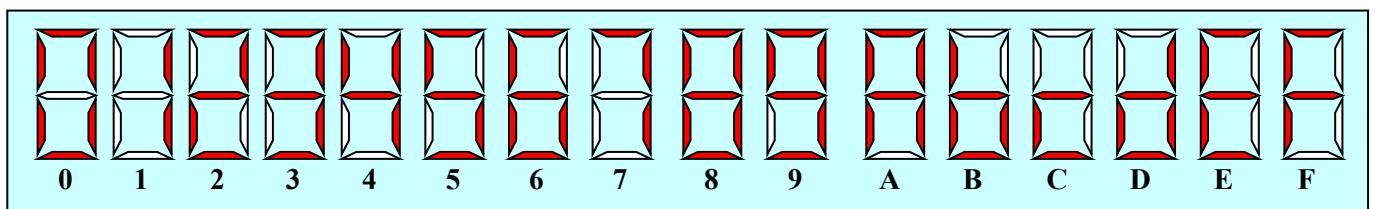
https://www.enib.fr/~kerhoas/vhdl_cours_3.html

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

- CLK : Signal d'horloge.
- EN : Entrée de validation du comptage.
- CLR : Entrée de mise à zéro du comptage.



Affichage des chiffres décimaux et hexadécimaux.



Programme VHDL : "CompteurAffichageHexa.vhd".

The screenshot shows the Quartus II 64-Bit interface with the project "CompteurAffichageHexa" open. The main window displays the VHDL source code for the entity "CompteurAffichageHexa". The code defines a process "Comptage" that increments two auxiliary variables, "aux" and "aux2", based on the clock edge and enable signal. It also handles carry operations for segments CO, BO, CO2, and BO2. The process "Retenue" is used to map these auxiliary values to the seven segments of the display. The code is annotated with comments explaining the logic for each segment.

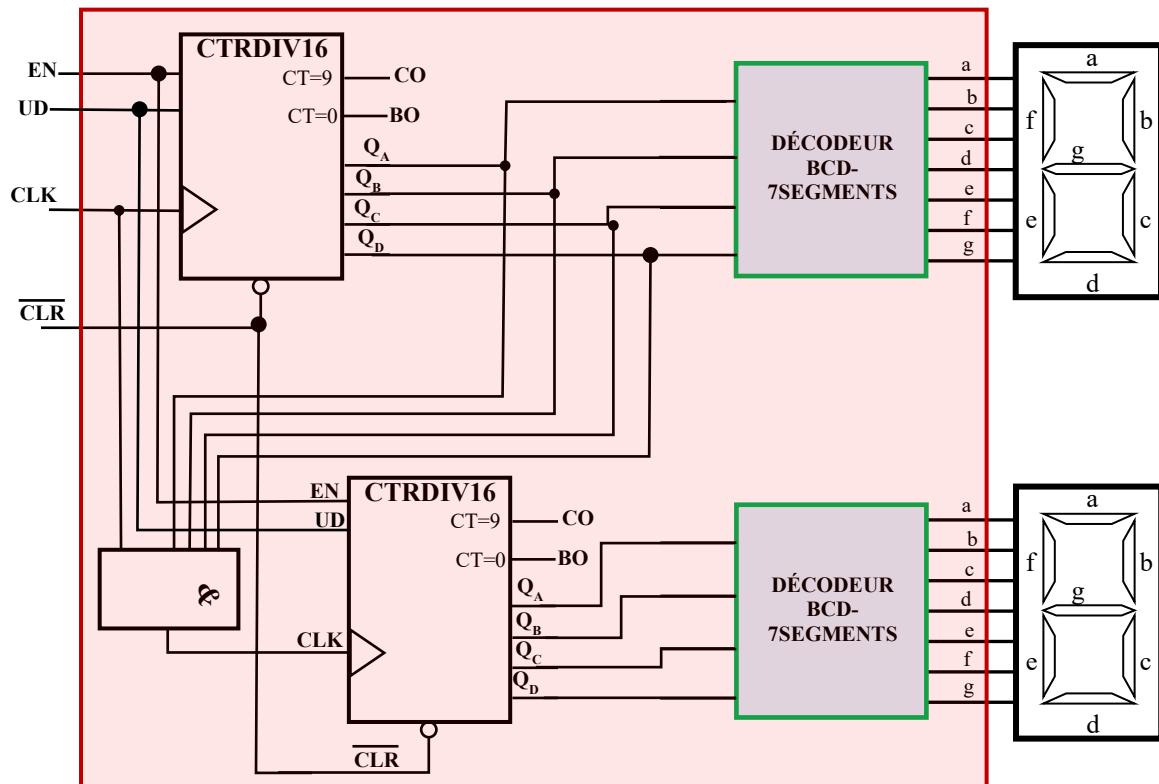
```

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); -- Les segments de l'afficheur 1
16    AF1 : OUT std_logic_vector(6 downto 0); -- Les segments de l'afficheur 2
17 );
18 END CompteurAffichageHexa;
19
20 ARCHITECTURE LaStructure OF CompteurAffichageHexa IS
21 signal aux : std_logic_vector(3 downto 0); -- Signaux intermédiaires pour le 1er digit de l'afficheur
22 signal aux2 : std_logic_vector(3 downto 0); -- Signaux intermédiaires pour le 2e digit de l'afficheur
23 begin
24 Comptage : process (CLK, CLR, EN)
25 begin
26 if((CLR = '0') and (EN = '1')) then
27 aux <= (others => '0');
28 aux2 <= (others => '0');
29 elsif ((CLR = '1') and (EN = '0')) then
30 aux <= aux;
31 elsif rising_edge (CLK) then -- Front montant du signal d'horloge
32 if (aux = "1111") then -- Si fin de cycle compteur 1er digit
33 aux <= "0000";
34 if (aux2 = "1111") then -- Si fin de cycle compteur 2e digit
35 aux2 <= "0000";
36 else
37 aux2 <= aux2 + 1; -- Incrémentation du compteur 2e digit
38 end if;
39 end if;
40 end if;
41 end process;
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(3)) 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) AND NOT (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 AF2(4) <= (NOT (aux(0) AND aux(1))) OR (aux(2) AND aux(3)) OR (NOT (aux(0) OR aux(2)));
78 OR (aux(1) AND aux(3));
79 AF2(5) <= (NOT (aux(0) OR aux(1) OR aux(2))) OR (NOT (aux(2)) AND aux(3)) OR (aux(1) AND aux(3));
80 OR (NOT (aux(0)) AND aux(2) AND NOT (aux(3))) OR (NOT (aux(1)) AND aux(2) AND NOT (aux(3)));
81 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));
82
83 -- Equations des 7-segments de l'afficheur : Deuxième digit.
84 AF1(0) <= (NOT (aux2(0) OR aux2(2))) OR (aux2(1) AND aux2(2)) OR (aux2(1) AND (NOT aux2(3)));
85 OR (aux2(0) AND aux2(2) AND (NOT aux2(3))) OR ((NOT aux2(1)) and (NOT aux2(2)) AND aux2(3));
86 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));
87 OR (aux2(0) AND aux2(1) AND (NOT aux(3))) OR (NOT (aux2(0) OR aux2(1) OR aux2(3)));
88 AF1(2) <= ((NOT aux2(2)) AND aux2(3)) OR (aux2(2) AND (NOT aux2(3))) OR ( aux2(0) AND (NOT aux2(1)));
89 OR (aux2(0) AND (NOT aux2(2))) OR (NOT (aux2(1) OR aux2(2)));
90 AF1(3) <= (NOT (aux2(0) OR aux2(1) OR aux2(2))) OR (NOT (aux2(0)) AND NOT (aux2(1)) AND aux2(3));
91 OR (aux2(1) AND NOT (aux2(2)) AND NOT (aux2(3))) OR (aux2(0) AND NOT (aux2(2)) AND aux2(3));
92 AF1(4) <= (NOT (aux2(0) AND aux2(1))) OR (aux2(2) AND aux2(3)) OR (NOT (aux2(0) OR aux2(2)));
93 OR (aux2(1) AND aux2(3));
94 AF1(5) <= (NOT (aux2(0) OR aux2(1) OR aux2(2))) OR (NOT (aux2(2)) AND aux2(3)) OR (aux2(1) AND aux2(3));
95 OR (NOT (aux2(0)) AND aux2(2) AND NOT (aux2(3))) OR (NOT (aux2(1)) AND aux2(2) AND NOT (aux2(3)));
96 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));
97
98 end process;
99 end LaStructure;
100

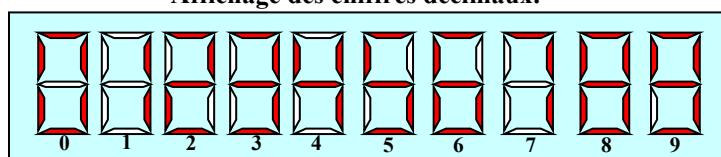
```

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

- **CLK** : Signal d'horloge.
- **EN** : Entrée de validation du comptage.
- **CLR** : Entrée de mise à zéro du comptage.
- **UD** : Comptage-DécomptageLIBRARYe. **UD = 1** → Comptage ; **UD = 0** → Décomptage.



Affichage des chiffres décimaux.



Programme VHDL : "CompteurAffichageDec2.vhd".

Quartus II 64-Bit - CompteurAffichageDec2

File Edit View Project Assignments Processing Tools Window Help

Project Navigator Entity MAX II: EPM570T100CS CompteurAffichageDec2

Hierarchy Files Design Units IP Components 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.std_logic_arith.all;
4 USE ieee.std_logic_unsigned.all;
5
6 ENTITY CompteurAffichageDec2 IS
7 PORT ( CLK : IN std_logic; --Signal d'horloge
8 CE : IN std_logic; --Entrée de validation du compteur
9 UD : IN std_logic; --UD=1 Comptage ; UD=0 Décomptage
10 RST : IN std_logic; --Mise à zéro du compteur
11 CO, CO2 : OUT std_logic; --Carry Out. Sortie fin de cycle
12 BO, BO2 : OUT std_logic; --Borrow Out. Sortie début de cycle
13 Sa, Sb, Sc, Sd, Se, Sf, Sg : OUT STD_LOGIC; --Segments de l'afficheur 1
14 Sal, Sbl, Scl, Sdl, Sel, Sfl, Sgl : OUT STD_LOGIC; --Segments de l'afficheur 2
15 QM : OUT std_logic_vector(3 downto 0); --Les sorties du compteur 2
16 QN : OUT std_logic_vector(3 downto 0); --Les sorties du compteur 1
17 );
18 END CompteurAffichageDec2;
19
20 ARCHITECTURE Comptage OF CompteurAffichageDec2 IS
21 signal aux : std_logic_vector(3 downto 0); --Signaux intermédiaires
22 signal aux2 : std_logic_vector(3 downto 0); --Signaux intermédiaires
23 begin
24 process (CLK, CE, RST) --Processus de comptage-décomptage
25 begin
26 if (RST = '0') then
27 aux <= "0000";
28 aux2 <= "0000";
29 elsif (CE = '1') then
30 if (rising_edge (CLK)) then
31 if (UD = '1') then --Cycle de comptage
32 if (aux = "0000") then
33 aux <= "0000";
34 if (aux2 = "1001") then
35 aux2 <= "0000";
36 else
37 aux2 <= aux2 + 1;
38 end if;
39 else
40 aux <= aux + 1;
41 end if;
42 else -- Cycle de décomptage
43 if (aux = "0000") then
44 aux <= "1001";
45 if (aux2 = "0000") then
46 aux2 <= "1001";
47 else
48 aux2 <= aux2 - 1;
49 end if;
50 else
51 aux <= aux - 1;
52 end if;
53 end if;
54 end if;
55 end if;
56 QN <= aux;
57 QM <= aux2;
58 end process; -- Fin du processus de comptage-décomptage
59
60 process (aux, aux2)
61 begin -- Processus qui gère l'affichage et qui génère les signaux de sortie CO et BO
62 if (aux = "0000") then
63 BO <= '1';
64 CO <= '0';
65 elsif (aux = "1001") then
66 CO <= '1';
67 BO <= '0';
68 else
69 BO <= '0';
70 CO <= '0';
71 end if;
72
73 if (aux2 = "0000") then
74 BO2 <= '1';
75 CO2 <= '0';
76 elsif (aux2 = "1001") then
77 CO2 <= '1';
78 BO2 <= '0';
79 else
80 BO2 <= '0';
81 CO2 <= '0';
82 end if;
83
84 -- Equations des 7-segments de l'afficheur : Premier digit.
85 Sa <= aux(1) OR aux(3) OR (aux(0) AND aux(2)) OR (NOT(aux(0) OR aux(2)));
86 Sb <= NOT(aux(2)) OR (aux(0) AND aux(1)) OR (NOT(aux(0) OR aux(1)));
87 Sc <= aux(0) OR NOT(aux(1)) OR aux(2);
88 Sd <= NOT(aux(0) OR aux(2)) OR (aux(1) AND NOT(aux(2))) OR (NOT(aux(0)) AND aux(1)) OR (aux(0) AND NOT(aux(1)) AND aux(2));
89 Sel <= NOT(aux(0) OR aux(2)) OR (NOT(aux(0)) AND aux(1));
90 Sf1 <= NOT(aux(0) OR aux(1)) OR aux(3) OR (NOT(aux(0)) AND aux(2)) OR (NOT(aux(1)) AND aux(2));
91 Sg1 <= aux(3) OR (NOT(aux(0)) AND aux(2)) OR (aux(1) XOR aux(2));
92
93 -- Equations des 7-segments de l'afficheur : Deuxième digit.
94 Sa <= aux2(1) OR aux2(3) OR (aux2(0) AND aux2(2)) OR (NOT(aux2(0) OR aux2(2)));
95 Sb <= NOT(aux2(2)) OR (aux2(0) AND aux2(1)) OR (NOT(aux2(0) OR aux2(1)));
96 Sc <= aux2(0) OR NOT(aux2(1)) OR aux2(2);
97 Sd <= NOT(aux2(0) OR aux2(2)) OR (aux2(1) AND NOT(aux2(2))) OR (NOT(aux2(0)) AND aux2(1)) OR (aux2(0) AND NOT(aux2(1)) AND aux2(2));
98 Se <= NOT(aux2(0) OR aux2(2)) OR (NOT(aux2(0)) AND aux2(1));
99 Sf <= NOT(aux2(0) OR aux2(1)) OR aux2(3) OR (NOT(aux2(0)) AND aux2(2)) OR (NOT(aux2(1)) AND aux2(2));
100 Sg <= aux2(3) OR (NOT(aux2(0)) AND aux2(2)) OR (aux2(1) XOR aux2(2));
101
102 end process;
103 end Comptage;
104

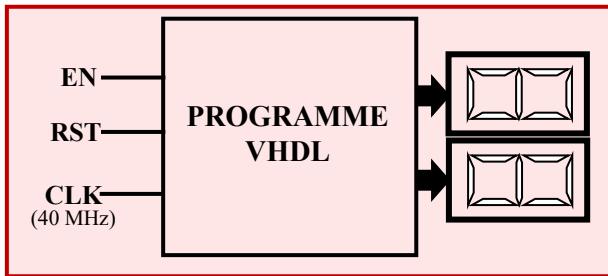
```

☞ 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_83	2	PIN_83	3.3-V LV..(default)		16mA (default)
Sa1	Output	PIN_83	2	PIN_83	3.3-V LV..(default)		16mA (default)
Sb	Output	PIN_85	2	PIN_85	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)
Sr1	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. $N = \frac{\ln(40000000)}{\ln(2)} = 25,25$. 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 hexadécimal **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"

The screenshot shows the Quartus II 64-Bit interface with the project "CompteurAffichageDec60s2". The main window displays the VHDL source code for the entity "CompteurAffichageDec60s2". The code implements a 60-second timer with a 1Hz output and 7-segment display outputs for two digits. The architecture uses a process to increment six 4-bit counters (aux1 to aux6) and one 2-bit counter (aux7). It also handles the reset (RST), clock (CLK), and enable (CE) inputs. The display logic is implemented using equations for each segment (Sa to Sg) based on the state of the QM and QN counters.

```

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

ENTITY CompteurAffichageDec60s2 IS
PORT (
    CLK : IN std_logic; --Signal d'horloge de fréquence F = 40MHz
    CE : IN std_logic; --Entrée de validation du compteur
    RST : IN std_logic; --Mise à zéro du compteur
    Q : OUT std_logic; --Sortie 1Hz
    Sa, Sb, Sc, Sd, Se, Sf, Sg : OUT STD_LOGIC; --Segments de l'afficheur 1
    Sal, Sbl, Scl, Sdl, Sel, Sfl, Sgl : OUT STD_LOGIC --Segments de l'afficheur 2
);
END CompteurAffichageDec60s2;

Architecture Comptage of CompteurAffichageDec60s2 is
--On a besoin de 26 bascules pour un diviseur modulo 40000000
signal aux1 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 1
signal aux2 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 2
signal aux3 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 3
signal aux4 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 4
signal aux5 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 5
signal aux6 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 6
signal aux7 : std_logic_vector(3 downto 0); --Signaux intermédiaires pour les sorties du compteur 7
signal auxQN : std_logic_vector(3 downto 0); --Signaux intermédiaires pour le digit 2 de l'afficheur
signal auxQM : std_logic_vector(3 downto 0); --Signaux intermédiaires pour le digit 1 de l'afficheur
begin
process (CLK, CE, RST, aux1, aux2, aux3, aux4, aux5, aux6, aux7, auxQN, auxQM) --Processus de comptage
begin
    if (RST = '0') then
        aux1 <= "0000";
        aux2 <= "0000";
        aux3 <= "0000";
        aux4 <= "0000";
        aux5 <= "0000";
        aux6 <= "0000";
        aux7 <= "00";
        auxQN <= "0000";
        auxQM <= "0000";
    elsif (CE = '1') then
        if (rising_edge (CLK)) then --Incrémentation des 6 compteurs 4 bits et du 7e à 2 bits
            if (aux1 = "1111") then
                aux1 <= "0000";
                if (aux2 = "1111") then
                    aux2 <= "0000";
                    if (aux3 = "1111") then
                        aux3 <= "0000";
                        if (aux4 = "1111") then
                            aux4 <= "0000";
                            if (aux5 = "1111") then
                                aux5 <= "0000";
                                if (aux6 = "1111") then
                                    aux6 <= "0000";
                                    if (aux7 = "11") then
                                        aux7 <= "00";
                                    else
                                        aux7 <= aux7 + 1; --Incrémentation compteur 7
                                        if (aux7(0) = '1') then --Sortie du compteur 7: F = 1Hz
                                            if (auxQN = "1001") then
                                                auxQN <= "0000";
                                                if (auxQM = "1001") then
                                                    auxQM <= "0000";
                                                    else
                                                        auxQM <= auxQM + 1; --Incrémentation digit 1 afficheur
                                                    end if;
                                                else
                                                    auxQN <= auxQN + 1; --Incrémentation digit 2 afficheur
                                                end if;
                                            end if;
                                        else
                                            aux6 <= aux6 + 1; --Incrémentation compteur 6
                                        end if;
                                    else
                                        aux5 <= aux5 + 1; --Incrémentation compteur 5
                                    end if;
                                else
                                    aux4 <= aux4 + 1; --Incrémentation compteur 4
                                end if;
                            else
                                aux3 <= aux3 + 1; --Incrémentation compteur 3
                            end if;
                        else
                            aux2 <= aux2 + 1; --Incrémentation compteur 2
                        end if;
                    else
                        aux1 <= aux1 + 1; --Incrémentation compteur 1
                    end if;
                end if;
            end if;
        end if;
        Q <= aux7(0);
    -- Mise à zéro du compteur : 10 0110 0010 0101 1010 0000 0000 -> 0x2625A00 -> 40000000
    if ((aux1="0000") and (aux2="0000") and (aux3="1010") and (aux4="0101") and (aux5="0010") and (aux6="0110") and (aux7="10")) then
        aux1 <= "0000";
        aux2 <= "0000";
        aux3 <= "0000";
        aux4 <= "0000";
        aux5 <= "0000";
        aux6 <= "0000";
        aux7 <= "00";
    end if;
end process; -- Fin du processus de comptage
Affichage: process (aux7, auxQN, auxQM)
begin
    -- Equations des 7-segments de l'afficheur : Premier digit.
    Sa <- auxQM(1) OR auxQM(3) OR (auxQM(0) AND auxQM(2)) OR (NOT(auxQM(0) OR auxQM(2)));
    Sb <- NOT(auxQM(2)) OR (auxQM(0) AND auxQM(1)) OR (NOT(auxQM(0) OR auxQM(1)));
    Sc <- auxQM(0) OR NOT(auxQM(1)) OR auxQM(2);
    Sd <- NOT(auxQM(0) OR auxQM(2)) OR (NOT(auxQM(1) AND NOT(auxQM(2))) OR (NOT(auxQM(0) AND auxQM(1)) OR (auxQM(0) AND NOT(auxQM(1)))));
    Se <- NOT(auxQM(0) OR auxQM(2)) OR (NOT(auxQM(0) AND auxQM(1)));
    Sf <- NOT(auxQM(0) OR auxQM(1)) OR auxQM(3) OR (NOT(auxQM(0)) AND auxQM(2)) OR (NOT(auxQM(1)) AND auxQM(2));
    Sg <- auxQM(3) OR (NOT(auxQM(0)) AND auxQM(2)) OR (auxQM(1) XOR auxQM(2));
    -- Equations des 7-segments de l'afficheur : Deuxième digit.
    Sal <- auxQN(1) OR auxQN(3) OR (auxQN(0) AND auxQN(2)) OR (NOT(auxQN(0) OR auxQN(2)));
    Sbl <- NOT(auxQN(2)) OR (auxQN(0) AND auxQN(1)) OR (NOT(auxQN(0) OR auxQN(1)));
    Scl <- NOT(auxQN(0) OR auxQN(3)) OR (auxQN(1) AND NOT(auxQN(2))) OR (NOT(auxQN(0) AND auxQN(1)) OR (auxQN(0) AND NOT(auxQN(1))) AND auxQN(2));
    Sel <- NOT(auxQN(0) OR auxQN(2)) OR (NOT(auxQN(0)) AND auxQN(1));
    Sfl <- NOT(auxQN(0) OR auxQN(1)) OR auxQN(3) OR (NOT(auxQN(0)) AND auxQN(2)) OR (NOT(auxQN(1)) AND auxQN(2));
    Sgl <- auxQN(3) OR (NOT(auxQN(0)) AND auxQN(2)) OR (auxQN(1) XOR auxQN(2));
end process Affichage;
end Comptage;

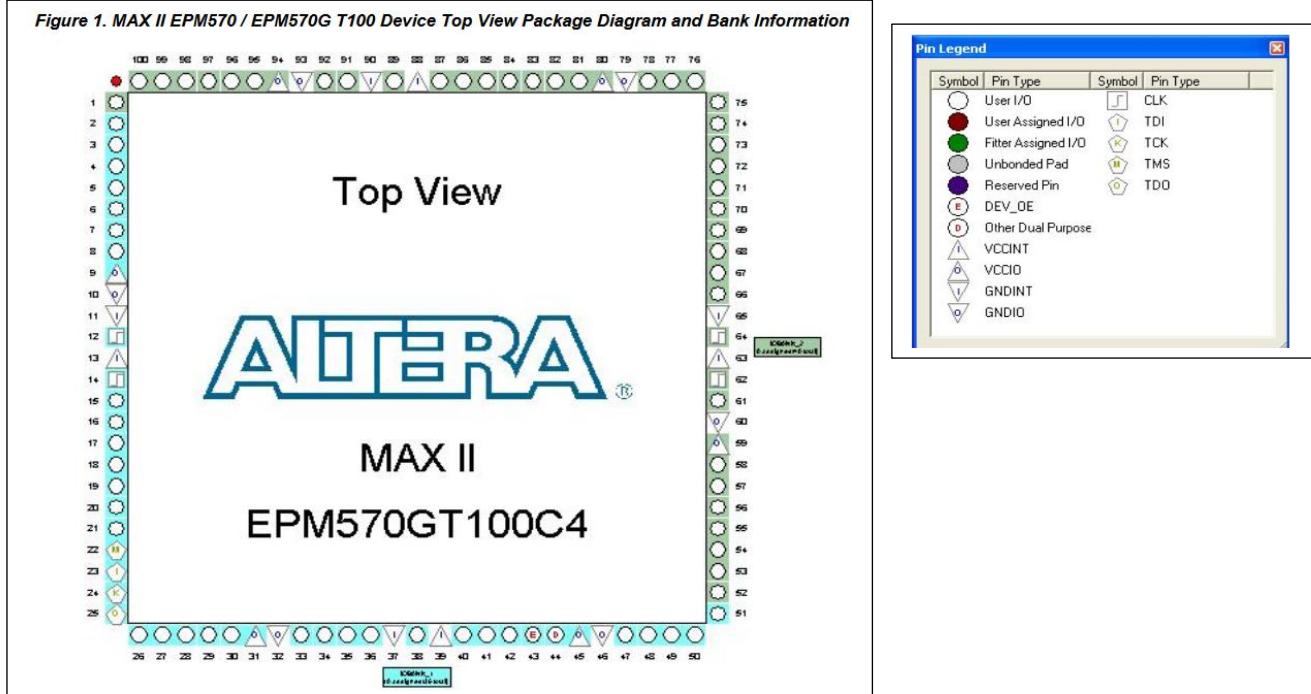
```

Affectation des broches.

All Pins	Node Name	Direction	Location	I/O Bank	Filter Location	I/O Standard	Reserved	Current Strength
	in_CE	Input	PIN_29	1	PIN_29	3.3-V LV...default)		16mA (default)
	in_CLK	Input	PIN_62	2	PIN_62	3.3-V LV...default)		16mA (default)
	in_RST	Input	PIN_38	1	PIN_38	3.3-V LV...default)		16mA (default)
	out_Sa	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)		16mA (default)
	out_Sb	Output	PIN_69	2	PIN_69	3.3-V LV...default)		16mA (default)
	out_Sb1	Output	PIN_85	2	PIN_85	3.3-V LV...default)		16mA (default)
	out_Sc	Output	PIN_70	2	PIN_70	3.3-V LV...default)		16mA (default)
	out_Sc1	Output	PIN_87	2	PIN_87	3.3-V LV...default)		16mA (default)
	out_Sd	Output	PIN_72	2	PIN_72	3.3-V LV...default)		16mA (default)
	out_Sd1	Output	PIN_91	2	PIN_91	3.3-V LV...default)		16mA (default)
	out_Se	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_SF	Output	PIN_76	2	PIN_76	3.3-V LV...default)		16mA (default)
	out_SF1	Output	PIN_97	2	PIN_97	3.3-V LV...default)		16mA (default)
	out_Sg	Output	PIN_81	2	PIN_81	3.3-V LV...default)		16mA (default)
	out_Sq1	Output	PIN_99	2	PIN_99	3.3-V LV...default)		16mA (default)
	out_Q	Output	PIN_58	2	PIN_58	3.3-V LV...default)		16mA (default)
	<<new node>>							

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.