

Nom & Prénom :

Groupe :

TP1

Prise en main du VHDL « Serrure électronique »

Présentation

L'objectif de ce TP consiste à prendre en main les concepts de la conception de fonctions logiques en VHDL. Pour cela, nous allons concevoir une serrure électronique dédiée à l'ouverture d'une porte, comme cas d'étude. L'utilisateur programme un code au moyen de boutons (dip-switch). Si le code correspond à la séquence que l'utilisateur rentre, la serrure s'ouvre, sinon elle reste fermée. Le schéma fonctionnel de la serrure est donné ci-dessous. L'architecture électronique de la serrure est composée d'un clavier matriciel à 16 touches au format hexadécimal (touches de 0 à F), d'un circuit de décodage des touches, d'un circuit de décision pour valider le code rentrée, et, un circuit d'ouverture de la serrure (simulée au moyen de deux diodes, rouge pour fermée et verte pour signaler que la serrure est ouverte).

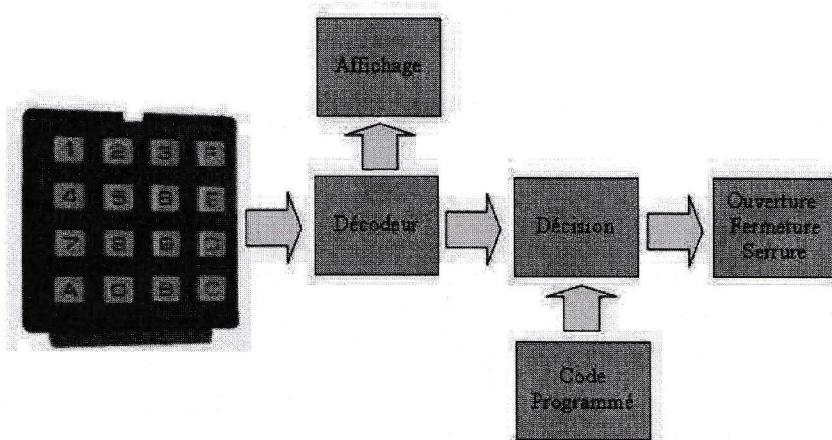


Figure 1 : Serrure électronique

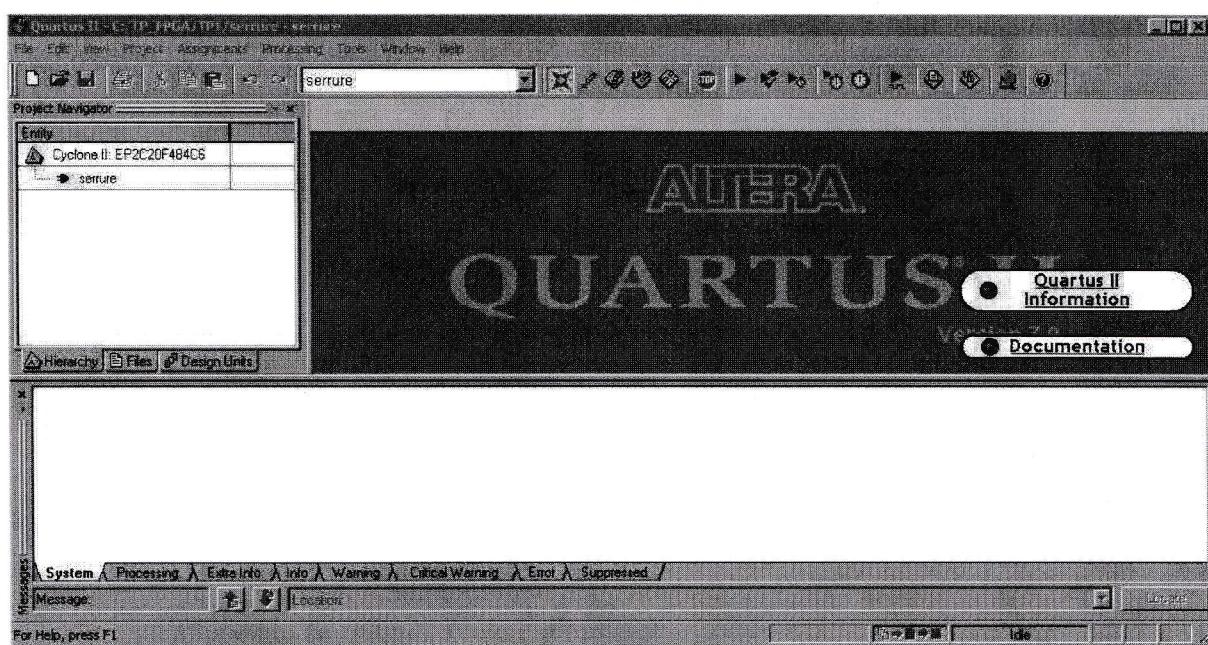
Nom & Prénom :

Groupe :

Préliminaires

Pour ce TP, nous allons créer un nouveau projet.

1. Ouvrir le programme Quartus
2. Créer sur votre compte, un répertoire TP_FPGA
3. Créer un sous-répertoire TP1
4. Sous Quartus, créer un nouveau projet. Dans le menu file, sélectionner new projet wizard.
5. Remplissez les champs suivants
 - a. What is the working directories for this project ?
./TP_FPGA/TP1
 - b. What is the name of this project ?
serrure
 - c. What is the name of the top-level design entity for this project ...?
serrure
6. Page 3 du wizard
 - a. Dans la fenêtre Family, sélectionnez **Cyclone II**
 - b. Dans la liste des circuits disponibles, sélectionnez : **EP2C20F484C6**
7. Appuyer sur finish. Vous venez de créer votre premier projet Quartus.



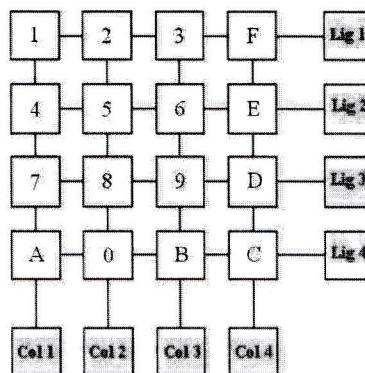
Nom & Prénom :

Groupe :

Partie I. Décodage des touches du clavier

Un clavier à 16 touches présente une réorganisation matricielle comme le montre la figure ci-dessous. Lorsqu'une touche est enfoncée, la connexion est alors faite entre la ligne et la colonne correspondant à la touche enfoncée. Pour déterminer quelle touche est enfoncée, il suffit d'alimenter l'une des quatre colonnes et de scruter les lignes. En envoyant le message binaire 1000 sur les colonnes (col1 à 1 et toutes les autres à 0), si la touche 4 est enfoncée, on récupère sur les lignes le message 0100 (lig2 à 1 et toutes les autres à 0).

Dans un premier temps, le clavier sera émulé au moyen des périphériques de la carte. Les SWITCH 0 à 3 permettront de modéliser l'état des colonnes et les SWITCH 4 à 8 l'état des lignes.



On désire réaliser un décodage des touches enfoncées. Lorsqu'une touche est enfoncée, elle correspond au croisement d'une ligne et d'une colonne dont les états sont hauts. En connaissant les états des lignes et des colonnes, il est alors possible de déterminer la touche qui a été enfoncée. Une seule touche peut être enfoncée à la fois.

La table de vérité correspondant à la touche enfoncée en fonction des états des lignes et des colonnes, est donnée ci-dessous.

Col4	Col3	Col2	Col1	Lig4	Lig3	Lig2	Lig1	Touche
0	0	0	1	0	0	0	1	1
0	0	0	1	0	0	1	0	4
0	0	0	1	0	1	0	0	7
0	0	0	1	1	0	0	0	A
0	0	1	0	0	0	0	1	2
0	0	1	0	0	0	1	0	5
0	0	1	0	0	1	0	0	8
0	0	1	0	1	0	0	0	0
0	1	0	0	0	0	0	1	
0	1	0	0	0	0	1	0	
0	1	0	0	0	1	0	0	

Nom & Prénom :

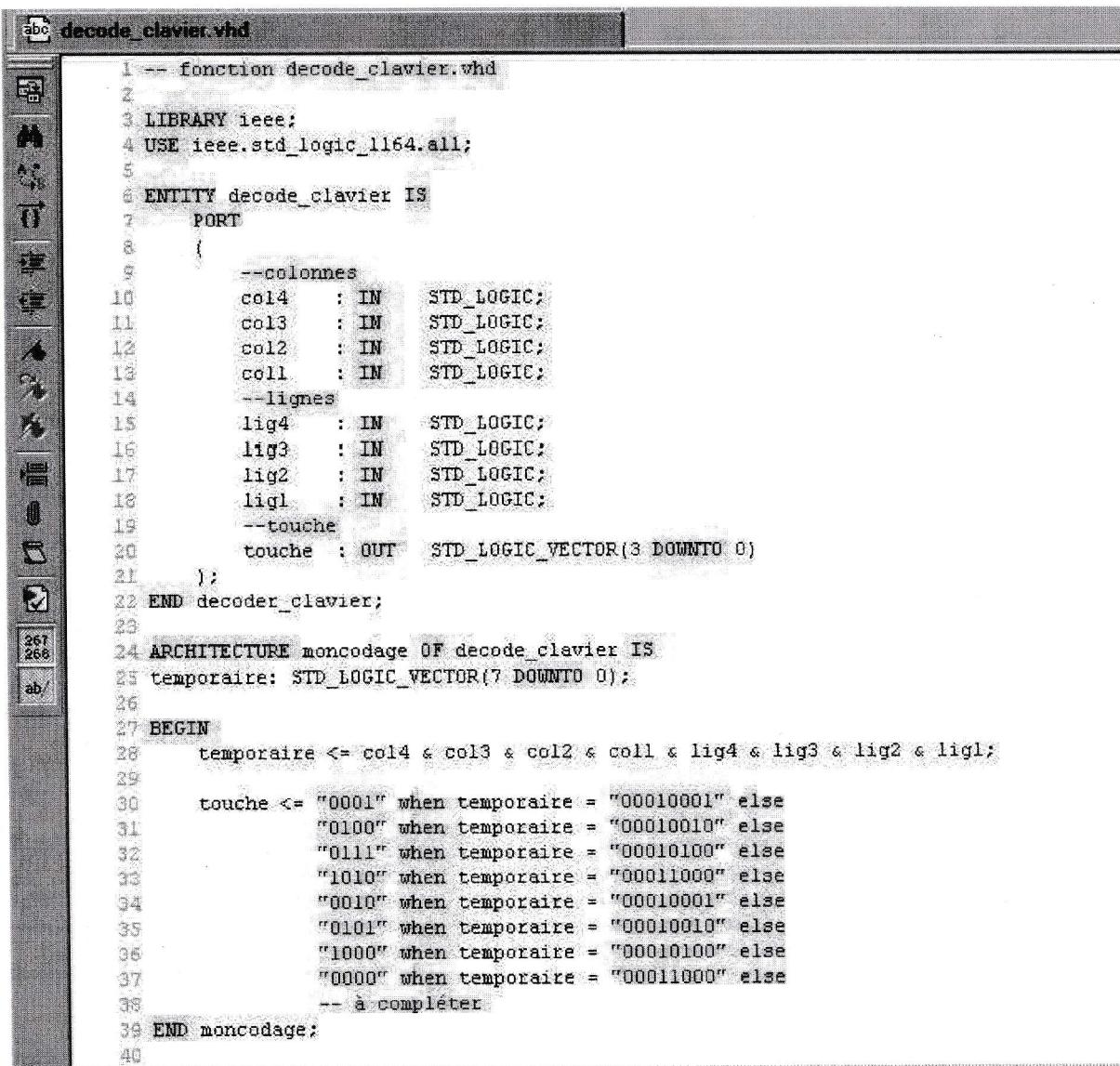
Groupe :

0	1	0	0	1	0	0	0	
1	0	0	0	0	0	0	1	
1	0	0	0	0	0	1	0	
1	0	0	0	0	1	0	0	
1	0	0	0	1	0	0	0	
Autres cas possibles								0

Question 1. Compléter la table de vérité pour les colonnes 3 et 4 sélectionnées

Le décodage des touches correspondant à la table de vérité ci-dessus est une fonction combinatoire de la forme **touche=f°(col4,col3,col2,col1,lig4,lig3,lig2,lig1)**

Pour coder cette fonction en VHDL, nous allons utiliser une instruction concurrente conditionnelle de type when-else.



```
abc decode_clavier.vhd
1 -- fonction decode_clavier.vhd
2
3 LIBRARY ieee;
4 USE ieee.std_logic_1164.all;
5
6 ENTITY decode_clavier IS
7 PORT
8 (
9     --colonnes
10    col4 : IN STD_LOGIC;
11    col3 : IN STD_LOGIC;
12    col2 : IN STD_LOGIC;
13    col1 : IN STD_LOGIC;
14    --lignes
15    lig4 : IN STD_LOGIC;
16    lig3 : IN STD_LOGIC;
17    lig2 : IN STD_LOGIC;
18    lig1 : IN STD_LOGIC;
19    --touche
20    touche : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
21 );
22 END decoder_clavier;
23
24 ARCHITECTURE moncodage OF decode_clavier IS
25 temporaire: STD_LOGIC_VECTOR(7 DOWNTO 0);
26
27 BEGIN
28     temporaire <= col4 & col3 & col2 & col1 & lig4 & lig3 & lig2 & lig1;
29
30     touche <= "0001" when temporaire = "00010001" else
31         "0100" when temporaire = "00010010" else
32         "0111" when temporaire = "00010100" else
33         "1010" when temporaire = "00011000" else
34         "0010" when temporaire = "00010001" else
35         "0101" when temporaire = "00010010" else
36         "1000" when temporaire = "00010100" else
37         "0000" when temporaire = "00011000" else
38         -- à compléter
39 END moncodage;
40
```

Nom & Prénom :

Groupe :

- Menu file / New, sélectionnez un fichier de type VHDL et sauvez le sous le nom **decode_clavier.vhd**
- Recopier le code ci-dessus.

Question 2. Dans l'entité, quel est le type de la sortie touche ?

Type de la sortie touche	
--------------------------	--

Question 3. Représentez le circuit decode_clavier en faisant apparaître les entrées / sorties.

Dessin du circuit decode_clavier	
-------------------------------------	--

Question 4. Donnez la signification du signal temporaire dans l'architecture

Signal temporaire ?	
---------------------	--

Question 5. Complétez dans l'architecture l'instruction when-else afin que le circuit soit conforme à la table de vérité

Code VHDL	
-----------	--

Nom & Prénom :

Groupe :

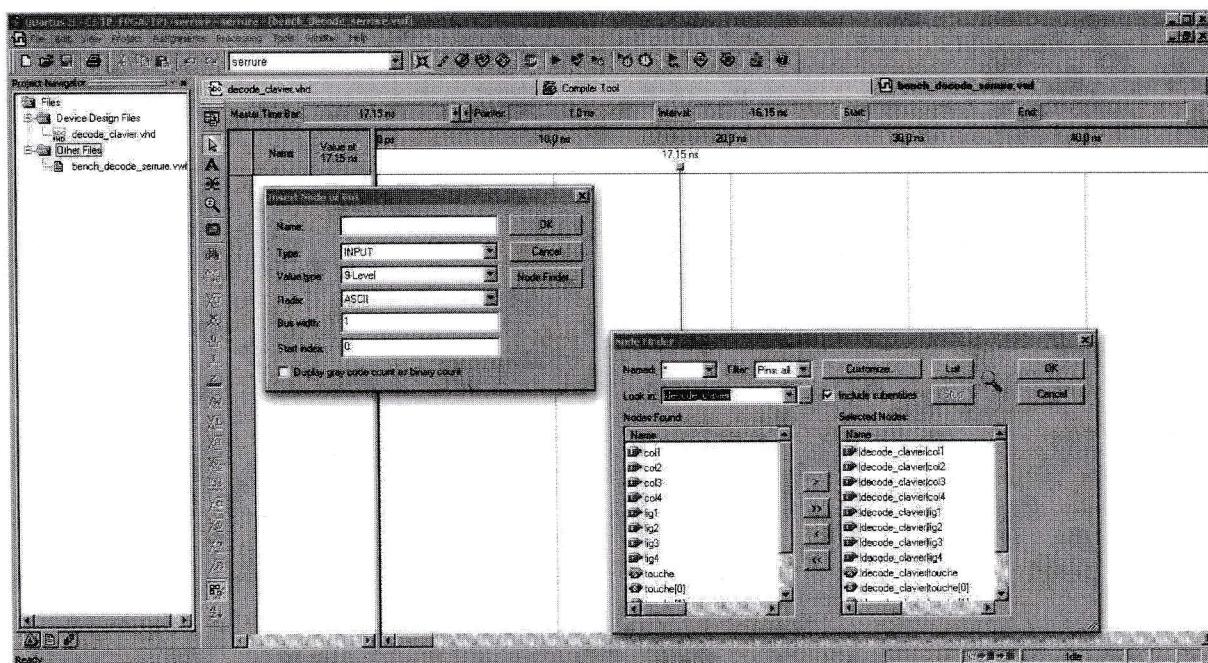
- Compiler le fichier decode_serrure.vhd
 - Pour cela, sélectionner le fichier dans l'onglet « Files » (fenêtre à gauche) et déclarez le comme **Set as Top-Level Entity**
 - Dans le menu Processing, sélectionnez **Processing tool**
 - Lancez la compilation : **Start Compilation**

Question 6. Deux erreurs apparaissent, lesquelles ? Comment les corriger ?

Erreurs	Solutions

Validation 1. Faite valider par l'enseignant

- Simuler le fichier decode_serrure.vhd
 - Menu File/New sélectionnez dans l'onglet « Others Files » un fichier de type **Vector Waveform Files**
 - Sauvegardez le fichier sous le nom **bench_decode_serrure**
 - Dans la fenêtre de gauche, double cliquez et sélectionnez **node finder**
 - Dans le node finder, sélectionnez dans Filter Pins : all et appuyer sur le bouton List. Toutes les entrées/sorties apparaissent. Sélectionnez toutes et faites les basculer dans la partie de droite avec le bouton >. Appuyer sur Ok.
 - Vous venez de sélectionnez les entrées/sorties qui vont être simulées.



Nom & Prénom :

Groupe :

Question 7. Dessinez les stimuli temporels en entrées qui vont permettre de tester toute la table de vérité.

Mem-Block	Name	Val. 0 ps	0 ps	100 ps	200 ps	300 ps	400 ps	500 ps	600 ps	700 ps	800 ps	900 ps	1000 ps	1200 ps	1300 ps	1400 ps	1500 ps	1600 ps
col[0]	col1	A[0]																
col[1]	col2	A[0]																
col[2]	col3	A[0]																
col[3]	col4	A[0]																
lig[1]	lig1	A[0]																
lig[2]	lig2	A[0]																
lig[3]	lig3	A[0]																
lig[4]	lig4	A[0]																
touche	touche	A[0]												[0]				

- Pour la simulation du circuit decode_serrure.vhd au moyen du fichier de test bench
 - Dans le menu Processing, sélectionnez : **Simulation tool**
 - Dans le mode de simulation, sélectionnez : **Functional**
 - Générez la netlist de simulation fonctionnelle : bouton Generate Functional ...
 - Lancez la simulation : bouton **Start**

Validation 2. Faites valider par l'enseignant

Question 8. Dessinez les chronogramme de la sortie touche pour une simulation fonctionnelle.

Question 9. Dessinez les chronogramme de la sortie touche pour une simulation timing. Pour cela recommencez la simulation en mode timing.

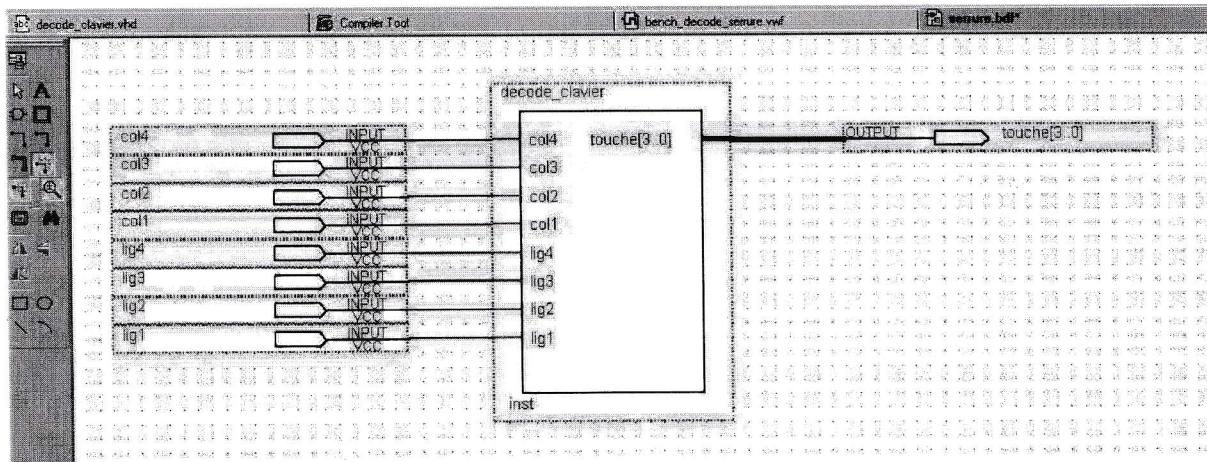
Nom & Prénom :

Groupe :

Question 10. Conclusion

- Assignez les entrées / sorties du circuit aux broches du FPGA. Pour cela :
 - Créer un nouveau fichier de type BDF (Block Diagram File – fichier de description schématique).
 - Sauver le sous le nom : serrure.bdf
 - Sélectionner le fichier decode_serrure.vhd et dans le menu File, Sélectionner Create/Update, create a symbol files for current file.
 - Dans le fichier serrure.bdf, insérer le composant decode_serrure
 - Mettre en entrée/sortie des broches (input, output)
 - Via l'assignement editor, affecter les entrées du circuit decode_clavier aux broches du FPGA

Validation 3. Faites valider par l'enseignant



Swith(0)	Col4	PIN_L22
Swith(1)	Col3	PIN_L21
Swith(2)	Col2	PIN_M22
Swith(3)	Col1	PIN_V12
Swith(4)	Lig4	PIN_W12
Swith(5)	Lig3	PIN_U12
Swith(6)	Lig2	PIN_U11
Swith(7)	Lig1	PIN_M2

Nom & Prénom :

Groupe :

Partie II. Affichage du code sélectionné

On désire afficher sur un afficheur 7 segments la valeur de la touche enfoncée. Un afficheur 7 segments, comme son nom l'indique est composée de 7 leds qui portent généralement les noms a, b, c, d, e, f et g.

Allumer un des 7 segment consiste à positionner un **0 logique** sur la sortie du FPGA qui est reliée au segment. La liste des broches est donnée ci-dessous avec la correspondance des segments.

La fonction combinatoire va être développée en VHDL.

Afficheur 7 segments Hex0	a	PIN_J2
	b	PIN_J1
	c	PIN_H2
	d	PIN_H1
	e	PIN_F2
	f	PIN_F1
	g	PIN_E2

- Menu file / New, sélectionnez un fichier de type VHDL et sauvez le sous le nom **segments.vhd**
- Recopier le code ci-dessous dans le fichier segments.vhd

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3
4 ENTITY segments IS
5   PORT
6   (
7     --ENTREES
8     touche : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
9     --SORTIES
10    a : OUT STD_LOGIC;
11    b : OUT STD_LOGIC;
12    c : OUT STD_LOGIC;
13    d : OUT STD_LOGIC;
14    e : OUT STD_LOGIC;
15    f : OUT STD_LOGIC;
16    g : OUT STD_LOGIC
17  );
18 END segments;
19
20 ARCHITECTURE moncodage OF segments IS
21
22 BEGIN
23   a <= ---a completer;
24   b <= ---a completer;
25   c <= ---a completer;
26   d <= ---a completer;
```

Nom & Prénom :

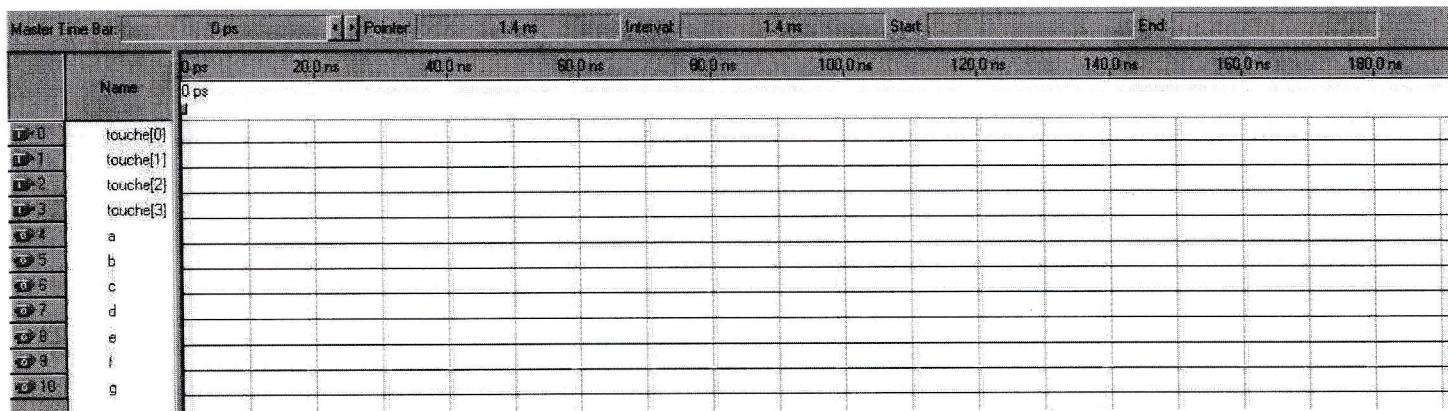
Groupe :

Question 11. Complétez l'architecture et donnez le code VHDL

Code VHDL	
-----------	--

Validation 4. Faites valider par l'enseignant

Question 12. Dessinez ci-dessous le résultat de la simulation de ce circuit.



- Créer un symbol graphique pour ce circuit
- Insérer le symbol dans le fichier serrure.dbf
- Relier les sorties du circuit decode_serrure aux entrées du circuit segments
- Assigner les sorties du circuit segments aux broches du FPGA. Utiliser la table de correspondance.

Nom & Prénom :

Groupe :

Partie III. Comparaison au code programmé

La serrure électronique ne doit ouvrir la porte que si la touche rentrée correspond à la valeur pré-enregistrée. La valeur sera programmée localement en VHDL et ne sera connue que de l'étudiant.

La valeur rentrée au clavier (touche) doit alors être comparée à la valeur programmée pour commander la serrure. La led rouge LEDR0 sera allumée si les codes sont différents (clavier et programmé) et la led vertes LEDG0 sera allumée si les codes sont les mêmes.

Exemple : La valeur programmée est CODE= "0101". Si Col2 et Lig2 sont sélectionnées, c'est à dire la touche 5 enfoncée, la led verte s'allume sinon c'est la rouge.

Cette fonction combinatoire de comparaison sera programmée en VHDL. La table de vérité est donnée ci-dessous :

Touche(4)	Touche(3)	Touche(2)	Touche(1)	Sortie	Led
x	x	x	x	0	Rouge
CODE(4)	CODE(3)	CODE(2)	CODE(1)	1	Verte

Question 13. Donner l'équation logique de Sortie en fonction de Touche(4), Touche(3), Touche(2), Touche(1), CODE(4), CODE(3), CODE(2) et CODE(1)

Equation logique	
------------------	--

- Menu file / New, sélectionnez un fichier de type VHDL et sauvez le sous le nom **comparateur.vhd**
- Recopier le code ci-dessous dans le fichier comparateur.vhd

Nom & Prénom :

Groupe :

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3
4 ENTITY comparateur IS
5   PORT
6   (
7     --ENTREES
8     touche : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
9     --SORTIES
10    Sortie : OUT STD_LOGIC;
11    LedR  : OUT STD_LOGIC;
12    LedV  : OUT STD_LOGIC
13  );
14 END comparateur;
15
16 ARCHITECTURE moncodage OF comparateur IS
17 SIGNAL CODE : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0101";
18 BEGIN
19   Sortie <= -- à compléter
20   LedR <= -- à compléter
21   LedV <= -- à compléter
22
23 END moncodage;
```

Validation 5. Faite valider par l'enseignant

Question 14. Donner le code VHDL de l'architecture

Code VHDL	
-----------	--

- Créer un symbol graphique pour ce circuit comparateur
- Insérer le symbol dans le fichier serrure.dbf
- Relier les sorties du circuit decode_serrure aux entrées du circuit comparateur
- Relier les sorties du circuit comparateur aux leds rouge et verte

LEDR0	PIN_R20
LEDG0	PIN_U22

Validation 6. Faites valider par l'enseignant dans la salle.

Nom & Prénom :

Groupe :

TP2 et TP3

Mini-Projet FPGA

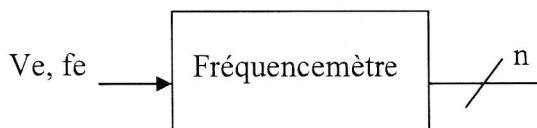
Nom & Prénom :

Groupe :

Conception d'un fréquencemètre numérique à calibrage automatique

I. Principe du fréquencemètre

Un fréquencemètre numérique est un appareil qui permet de mesurer la fréquence d'un signal d'entrée. Ici, notre signal d'entrée V_e à analyser sera un signal numérique de rapport cyclique 50% (rapport entre le temps au niveau haut du signal / à la période).



Le principe de fonctionnement du fréquencemètre que vous allez réaliser repose sur l'utilisation d'une fenêtre de comptage (figure 1) correspondant au niveau haut du signal d'entrée. En comptant le nombre de périodes d'une fréquence référence f_h (de fréquence plus grande que le signal d'entrée), dans cette fenêtre, nous pouvons alors en déduire la fréquence du signal d'entrée. Une demi-période du signal à analyser comporte $N \cdot T_h$. Connaissant T_h et en mesurant N , on peut déterminer la fréquence du signal d'entrée par la formule suivante :

$$T_e = 2 \cdot N \cdot T_h$$

$$f_e = \frac{f_h}{2 \cdot N}$$

Exemple 1: La fréquence de référence est de 50MHz ($T_h=20\text{ns}$). On mesure 4323 périodes de référence. La fréquence d'entrée est $5 \cdot 10^7 / (2 \cdot 4323) = 5783\text{Hz}$.

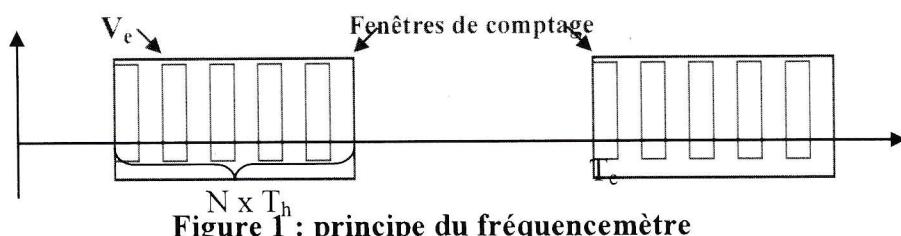
Exemple 2: La fréquence de référence est de 5MHz ($T_h=200\text{ns}$). On mesure 4323 périodes de référence. La fréquence d'entrée est $5 \cdot 10^6 / (2 \cdot 4323) = 578\text{Hz}$.

Exemple 3: La fréquence de référence est de 500kHz ($T_h=2\text{ms}$). On mesure 4323 périodes de référence. La fréquence d'entrée est $5 \cdot 10^5 / (2 \cdot 4323) = 57\text{Hz}$.

Exemple 4: La fréquence de référence est de 50kHz ($T_h=20\text{ms}$). On mesure 4323 périodes de référence. La fréquence d'entrée est $5 \cdot 10^4 / (2 \cdot 4323) = 5\text{Hz}$.

A partir des exemples, on comprend que la fréquence de référence va fixer le calibre de mesure et la précision de celle-ci.

La fréquence de référence f_h est obtenue par sélection d'une des quatre horloges de référence (f_1, f_2, f_3 ou f_4). Pour une fréquence du signal d'entrée, la fréquence de référence sera celle qui permettra d'obtenir la meilleure résolution concernant N .



Nom & Prénom :

Groupe :

Le fréquencemètre possède ici deux modes. Un mode manuel où l'utilisateur pourra sélectionner lui-même l'horloge de référence, et un mode automatique où en fonction de la valeur de N, un organe de décision déterminera le meilleur calibre (la meilleure fréquence parmi les quatre disponibles) pour N.

On se limitera ici à un fréquencemètre à sélection manuel du calibre. La partie automatique est considérée comme optionnelle et pourra être abordée par l'étudiant lorsque le fréquencemètre en mode manuel sera opérationnel.

Nom & Prénom :

Groupe :

Le schéma de principe d'un tel montage (figure 2) est donné ci-dessous et comporte plusieurs blocs que nous nous proposons de dimensionner par rapport au cahier des charges donné.

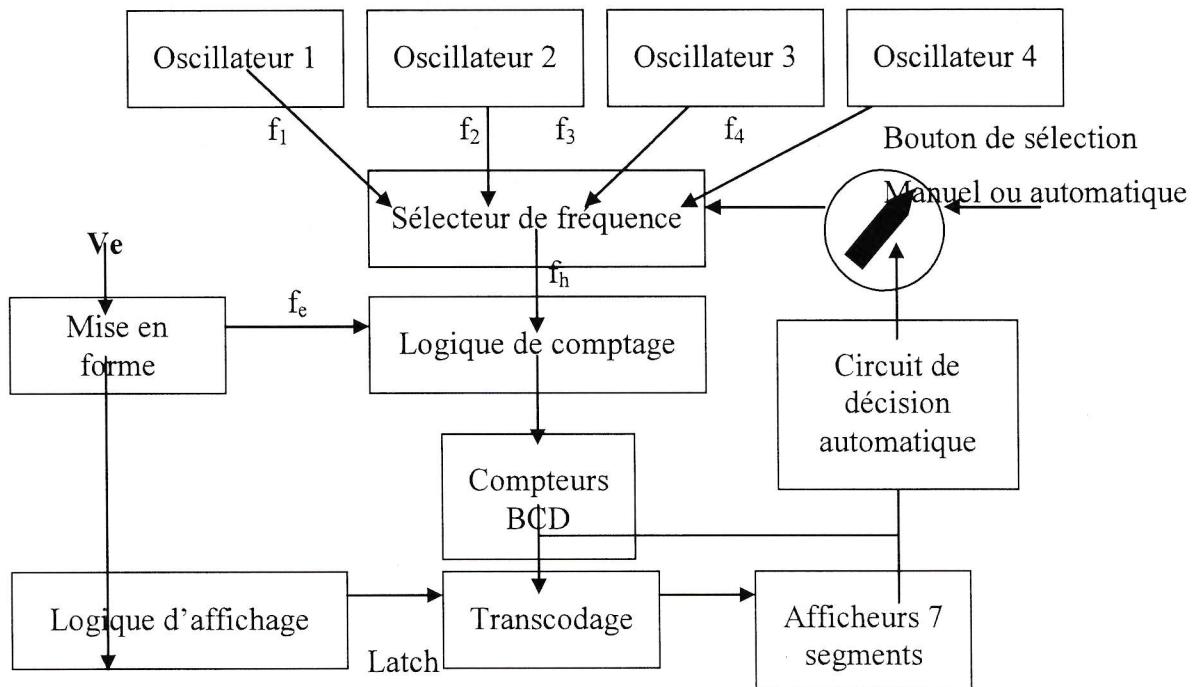


Figure 2 : principe du montage

Nom & Prénom :

Groupe :

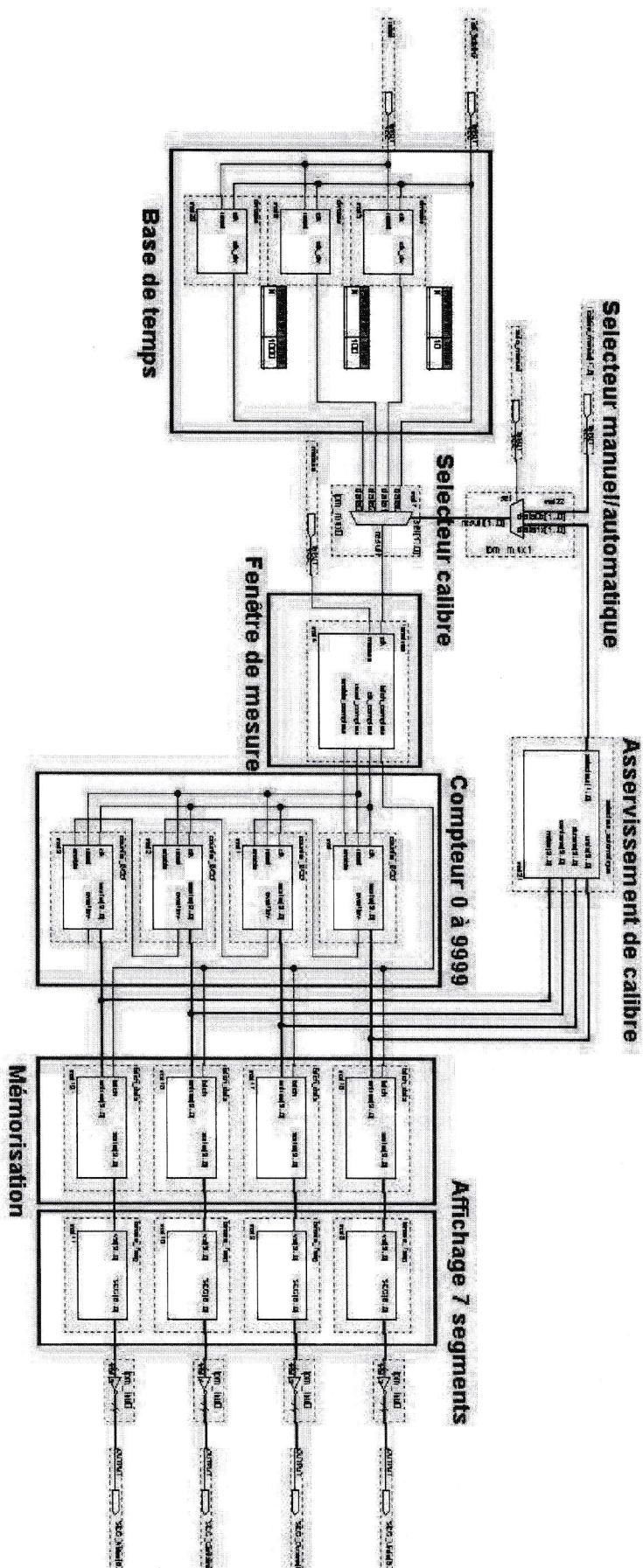


Figure 3: Implémentation sous Quartus

Nom & Prénom :

Groupe :

II. Spécifications

On désire réaliser un fréquencemètre numérique répondant au cahier des charges suivant :

Amplitude du signal d'entrée : $V_e \in [0 ; 5V]$: TTL

Fréquences de références :

- F1 : 50 MHz
- F2 : 5 MHz
- F3 : 500kHz
- F4 : 50 kHz

Plage de mesure de N : 0 à 9999

Affichage de N : 4 digits, 4 afficheurs 7 segments

III. Questions théoriques préliminaires

Question 1. En fonction des quatre fréquences de références disponibles, déterminer quelles seront les plages de mesures concernant la fréquence d'entrée. Pour cela compléter le tableau ci-dessous.

Fréquence de référence	Min	Max
$F_h=F_1=50MHz$		
$F_h=F_2=5MHz$		
$F_h=F_3=500kHz$		
$F_h=F_4=50kHz$		

Question 2. En déduire la plage de mesure de ce fréquencemètre

Fréquence Min	Fréquence Max

Question 3. Pour chaque calibre (fréquence f1, f2, f3 et f4), quelle sera la précision théorique de l'appareil. Cette estimation sera faite sur la base ou la plus petite valeur mesurable sur un calibre correspond à N=1.

Fréquence de référence	Précision théorique
$F_h=F_1=50MHz$	
$F_h=F_2=5MHz$	
$F_h=F_3=500kHz$	
$F_h=F_4=50kHz$	

Préliminaires

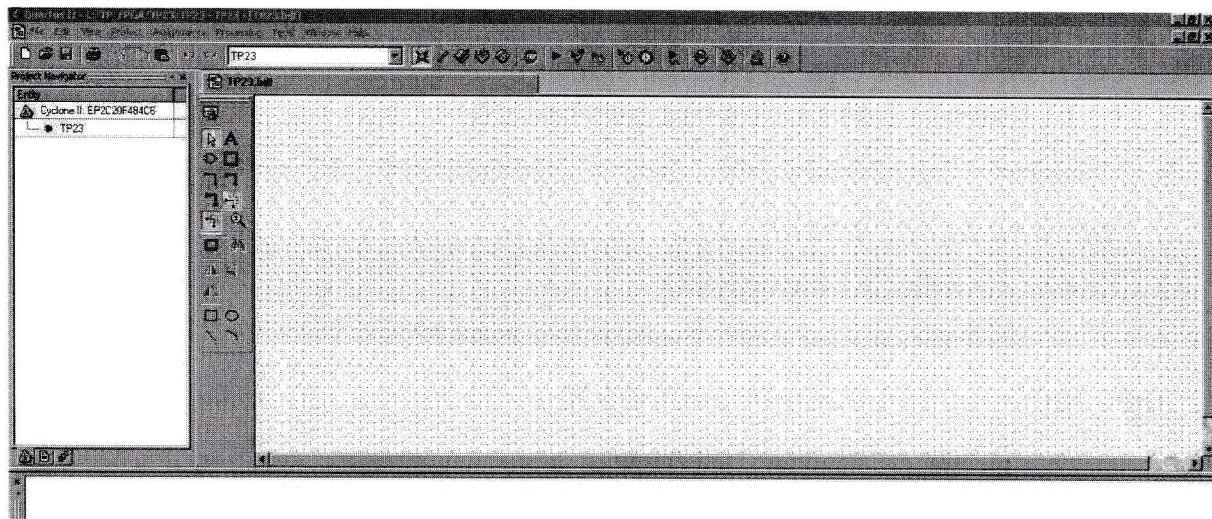
Pour ce TP, nous allons créer un nouveau projet.

8. Ouvrir le programme Quartus
9. Créer sur votre compte, un répertoire TP_FPGA
10. Créer un sous-répertoire TP23
11. Sous Quartus, créer un nouveau projet. Dans le menu file, sélectionner new projet wizard.
12. Remplissez les champs suivants

Nom & Prénom :

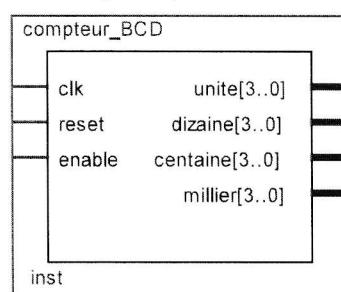
Groupe :

- a. What is the working directories for this project ?
..../TP_FPGA/TP23
 - b. What is the name of this project ?
TP23
 - c. What is the name of the top-level design entity for this project ...?
TP23
13. Page 3 du wizard
- a. Dans la fenêtre Family, sélectionnez **Cyclone II**
 - b. Dans la liste des circuits disponibles, sélectionnez : **EP2C20F484C6**
14. Appuyer sur finish. Vous venez de créer votre premier projet Quartus.



IV. Compteur N

Le compteur N est un compteur décimal (BCD) dont les sorties évoluent entre 0 et 9999.



Son entité comprend trois entrées ; une horloge, un signal de remise à zéro du compteur et des sorties et un signal enable qui autorise le compteur de compter. Les quatre sorties sur 4 bits évoluent de 0 à 9 (compteur BCD) et donnent respectivement la valeur de N sur 4 digits

Nom & Prénom :

Groupe :

(millier, centaine, dizaine et unité). Les évolutions des sorties seront conditionnées aux fronts montants de l'horloge.

Pour réaliser ce comptage entre 0 et 9999, 4 compteurs sont nécessaires. Un compteur unité, un compteur dizaine, un compteur centaine et un compteur millier. Le compteur dizaine est incrémenté de 1 lorsque le compteur unité passe de 9 à 0. Idem pour le compteur centaine qui s'incrémentera lorsque les compteurs unité et dizaine valent 9. Le compteur millier est incrémenté lorsque les compteurs unités, dizaines et centaines sont tous à 9.

Le code VHDL ci-dessous donne l'entité et l'architecture à compléter de ce compteur N.

- Définir un nouveau fichier VHDL
 - Menu File/New et sélectionner le type VHDL
 - Sauver le fichier sous le nom `compteurN.vhd`
- Compléter l'architecture afin que le comportement du circuit respecte les spécifications fonctionnelles.
- Simuler et vérifier le comportement.
 - On testera que les sorties évoluent bien entre 0000 et 9999.

Question 4. Consigner dans votre rapport :

- a. Le code VHDL de l'architecture
- b. Les résultats de simulation.

Validation 1. Faites valider par l'enseignant dans la salle.

Nom & Prénom :

Groupe :

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 USE ieee.numeric_std.all;
4
5 ENTITY compteurN IS
6 PORT
7 (
8     --ENTREES
9     clk      : IN STD_LOGIC;
10    reset   : IN STD_LOGIC;
11    enable  : IN STD_LOGIC;
12    --SORTIES
13    unite   : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
14    dizaine : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
15    centaine: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
16    millier : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
17 );
18 END;
19
20 ARCHITECTURE myarchitecture OF compteurN IS
21 SIGNAL count_u : UNSIGNED(3 DOWNTO 0);
22 SIGNAL count_d : UNSIGNED(3 DOWNTO 0);
23 SIGNAL count_c : UNSIGNED(3 DOWNTO 0);
24 SIGNAL count_m : UNSIGNED(3 DOWNTO 0);
25
26 BEGIN
27    unite <= count_u;
28    dizaine <= count_d;
29    centaine <= count_c;
30    millier <= count_m;
31
32    PROCESS (clk, reset)
33    BEGIN
34        IF reset = '1' THEN
35            count_u <= (OTHERS => '0');
36            count_d <= (OTHERS => '0');
37            count_c <= (OTHERS => '0');
38            count_m <= (OTHERS => '0');
39        ELSIF (clk'EVENT AND clk = '1') THEN
40            count_u <= count_u + 1;
41
42    A COMPLETER
43
```

V. Affichage et mémorisation

Cette partie concerne la mémorisation et l'affichage des valeurs en sorties du compteur N. La figure ci-dessous montre les connexions des blocs de mémorisation, d'affichage ainsi que la définition des broches du FPGA connectées aux afficheurs 7 segments.

Nom & Prénom :

Groupe :

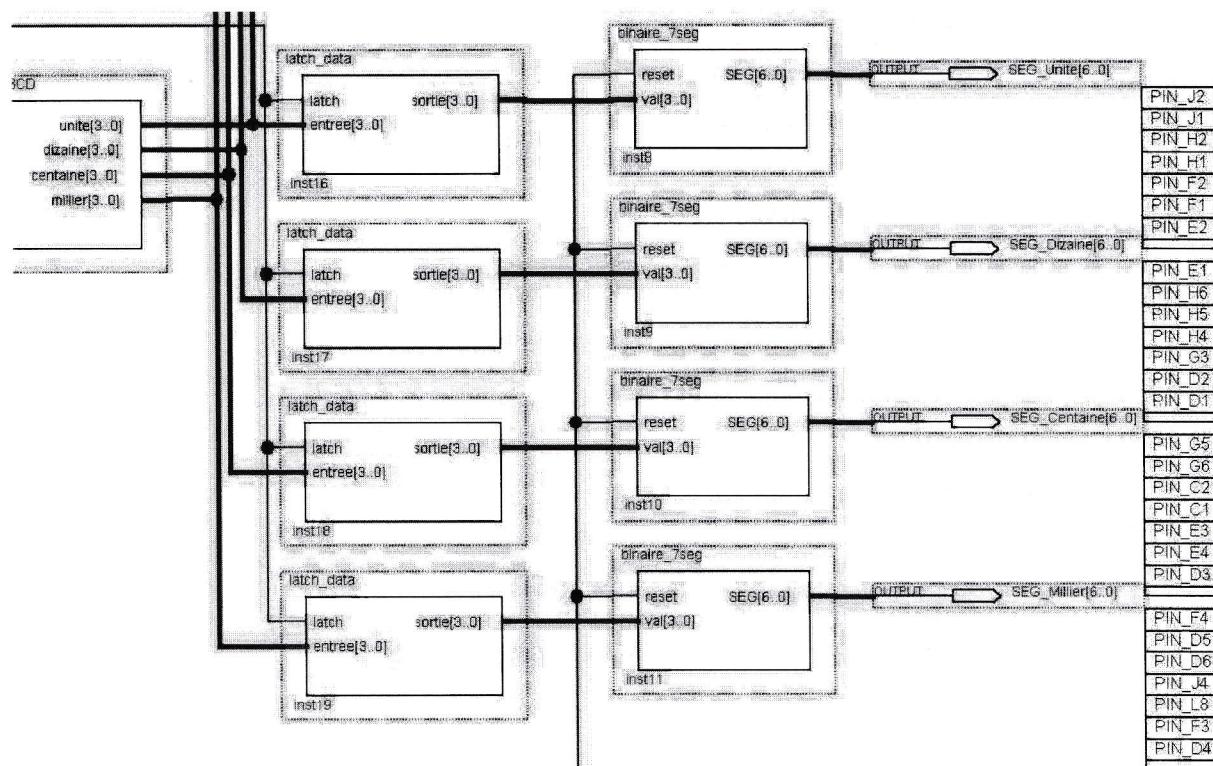


Figure 4

L'entité `latch_data` est un circuit latch 4 bits qui recopie la valeur de l'entrée définie sur 4 bits sur une sortie 4 bits lorsque le signal de commande `latch` est valide. La table de vérité de ce circuit est ci-dessous :

<code>latch</code>	<code>sortie[3..0] à n+1</code>	<code>Etats</code>
↑	<code>entrée[3..0]</code>	Recopie
0	<code>sortie[3..0] à n</code>	Mémoire
1	<code>sortie[3..0] à n</code>	

- Créer un fichier VHDL (File/New)
- Enregistrer le sous le nom `latch_data.vhd`
- Ecrire en VHDL, l'entité et l'architecture du circuit `latch_data`
- Simuler et vérifier le comportement.

Question 5. Consigner dans votre rapport

- Les codes VHDL
- Les résultats de simulation.

Validation 2. Faites valider par l'enseignant

L'entité `binaire_7seg` est un circuit qui transcode la valeur de l'entrée 4 bits sur une sortie 7 bits pour afficher le résultat d'un des digits du compteur sur un afficheur 7 segments.

- Créer un fichier VHDL (File/New)
- Enregistrer le sous le nom `binaire_7seg.vhd`

Nom & Prénom :

Groupe :

- Ecrire en VHDL, l'entité et l'architecture du circuit latch_data
 - Simuler et vérifier le comportement.

Question 6. Consigner dans votre rapport

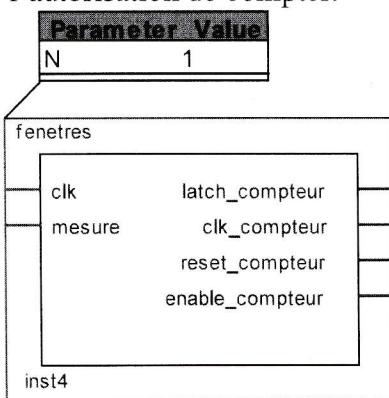
- c. Les codes VHDL
 - d. Les résultats de simulation.

Validation 3. Faite valider par l'enseignant

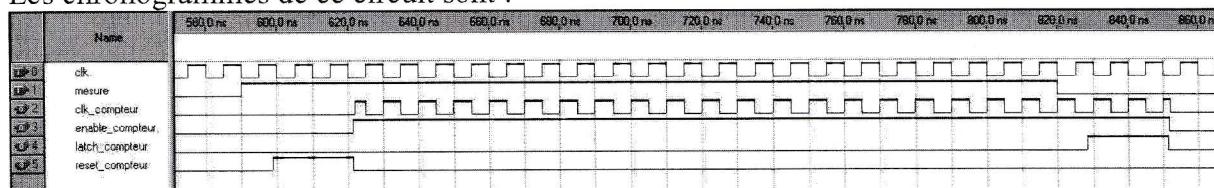
Afin de tester cette première partie, connecter le compteur, le circuit de mémorisation et l'affichage. On utilisera un bouton de type switch button comme signal de commande enable du circuit latch.

VI. Logique de comptage et fenêtre de mesure

On désire mesurer uniquement le nombre N sur la fenêtre de comptage. Pour cela, un bloc de fenêtrage doit être développé. L'entité fenêtre prend en entrée, le signal de mesure à analyser et l'horloge de référence (une des quatre ; f1, f2, f3 ou f4). Les sorties du circuit, sont la commande de mémorisation du circuit de mémorisation, l'horloge du compteur, la commande de remise à zéro du compteur et l'autorisation de compter.



Les chronogrammes de ce circuit sont :



Lorsque la fenêtre de comptage passe de 0 à 1, on commence par remettre à zéro le compteur, puis on autorise le compteur à compter. Avant que la fenêtre de mesure repasse à l'état bas, on mémorise la valeur du compteur en activant le signal `latch_compteur`. Le paramètre generic `N`, permet de définir la largeur temporelle des signaux `enable_compteur` et `latch_compteur`, en nombre de périodes d'horloge.

Question 7. Analyser le code VHDL donné

Question 8. Faite un organigramme correspondant au fonctionnement du circuit

Nom & Prénom :

Groupe :

- Simuler et vérifier le comportement.

Question 9. Consigner dans votre rapport, les développements et les résultats de simulation.

Validation 4. Faite valider par l'enseignant

Afin de tester cette première partie, nous allons connecter les circuits fenêtre, compteur, le circuit de mémorisation et l'affichage. Pour effectuer cette description schématique :

- Créer un fichier BDF (File/New)
- Enregistrer le sous le nom TP23.bdf
- Créer les symboles des composants, compteurN, latch_data et binaire_7seg
- Relier les composants entre eux conformément à la figure 4
- Synthétiser le fichier BDF
- Simuler et vérifier le comportement.

Question 10. Consigner dans votre rapport les résultats de simulation.

On vérifiera pratiquement le bon fonctionnement de ces quatre circuits.

Validation 5. Faite valider par l'enseignant

VII. Base de temps

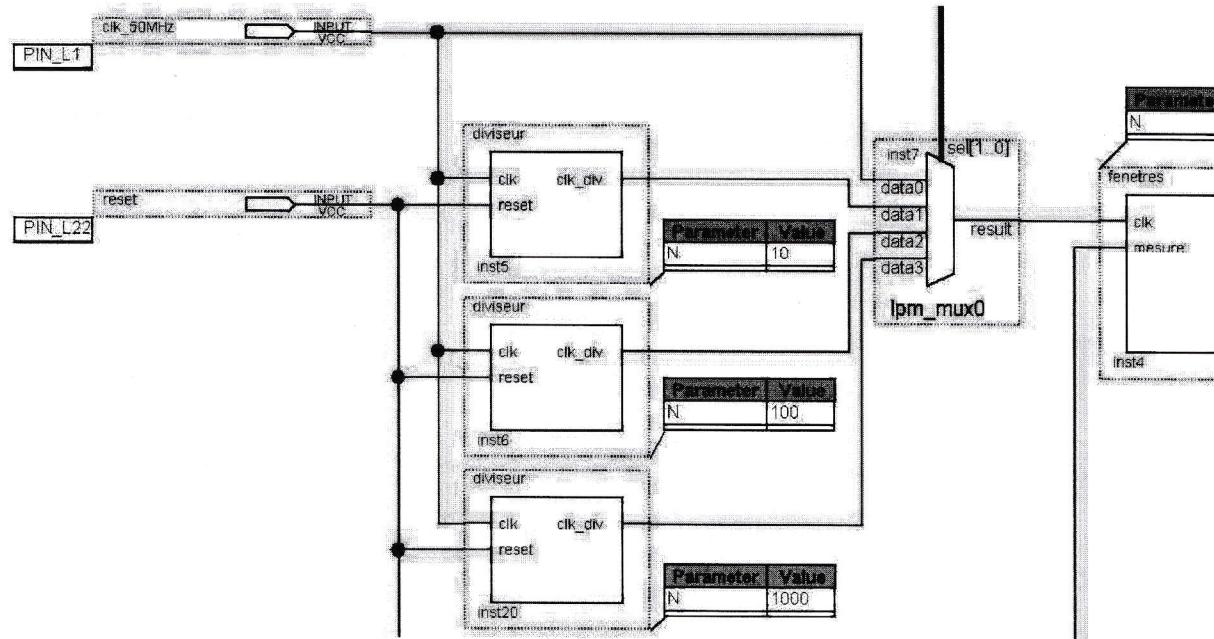
Le fréquencemètre intègre quatre calibres définis au moyen de quatre horloges. Ces horloges (50MHz, 5MHz, 500kHz et 50kHz) sont produites au moyen de divisions fréquentielles d'une horloge de référence à 50MHz. La figure ci-dessous illustre ce principe. La sélection d'une des quatre fréquences se fait au moyen d'un multiplexeur de 4->1. Ce dernier sera conçu à partir du composant lpm_mux disponible dans la librairie du constructeur.

Dans le fichier TP23.bdf

- a. Insérer un composant multiplexeur via la librairie lpm_mux
- b. Définir les paramètres du multiplexeur (4 entrées sur 1 bit)
- c. Relier le composant mux au reste du circuit conformément à la figure ci-dessous

Nom & Prénom :

Groupe :



L’entité diviseur est un compteur (diviseur de fréquence) générique. Il permettra de diviser le signal d’horloge par un facteur N programmable. Le rapport cyclique du signal de sortie clk_div est de 50%. Les fréquences f_1 , f_2 , f_3 et f_4 seront obtenues par division du quartz à 50MHz de la carte FPGA. Les rapports 10, 100 et 1000 permettront d’obtenir des fréquences de 5MHz, 500kHz et 50kHz de références.

- Créer un fichier VHDL (File/New)
- Enregistrer le sous le nom diviseur.vhd
- Ecrire en VHDL, l’entité et l’architecture du circuit diviseur
- Simuler et vérifier le comportement.

Question 11. Consigner dans votre rapport

- a. Les codes VHDL
- b. Les résultats de simulation.

Validation 6. Faite valider par l’enseignant

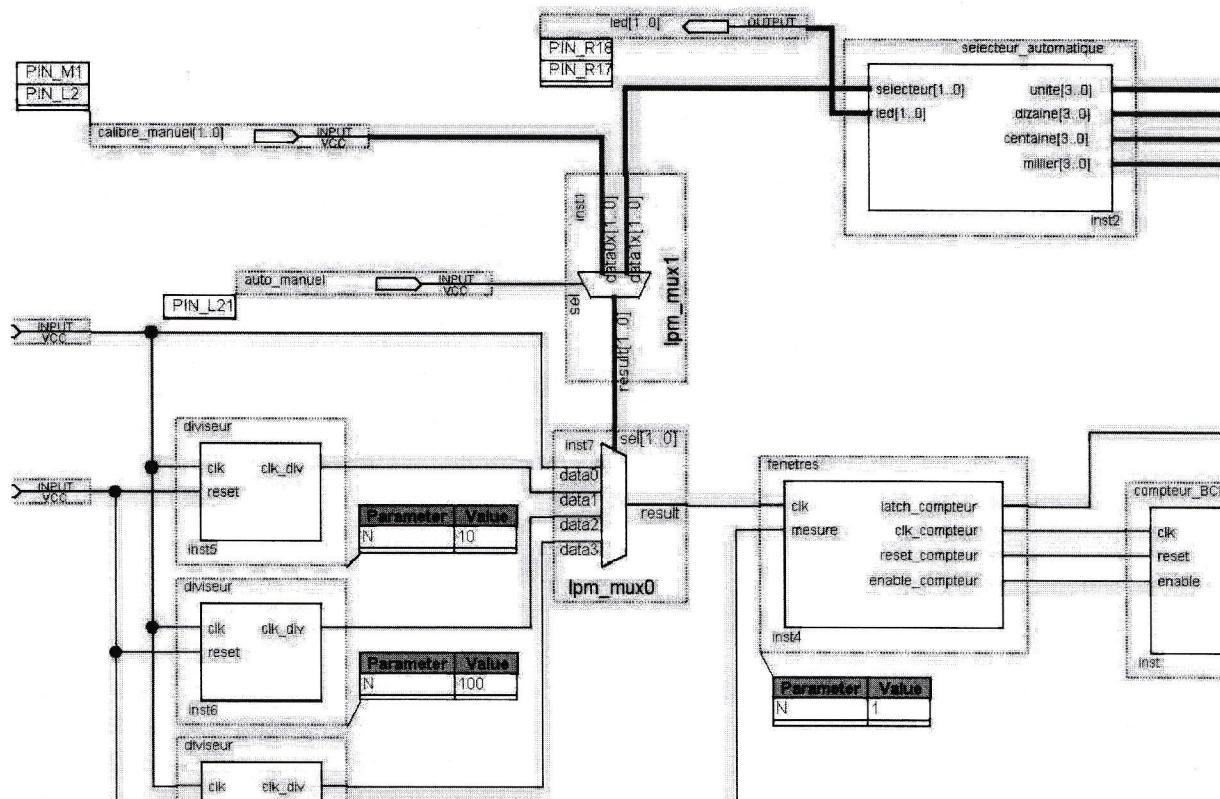
Au moyen de cette entité et des composants disponibles dans la librairie constructeur, réaliser l’architecture de la figure ci-dessus. Connecter cette partie aux précédents développements.

VIII. Mesures expérimentales

De manière à sélectionner le calibre disponible pour déterminer la fréquence du signal à mesurer, on désire que le fréquencemètre possède deux modes. Le premier manuel laisse la liberté à l’utilisateur de choisir l’une des quatre fréquences de références disponibles, tandis que le second mode automatique, utilise les sorties du compteur pour déterminer si N est représenté avec suffisamment de digits.

Nom & Prénom :

Groupe :



Dans un premier temps, on ne s'occupera que de la partie manuelle. On utilisera pour cela les switch SW8 et SW9 pour commander le multiplexeur lpm_mux0 (sélecteur de la fréquence).

Après avoir câbler votre montage, blocs par blocs, on se propose de mesurer les performances de ce montage vis à vis du cahier des charges initial. Pour cela, il est nécessaire d'injecter un signal dont on va mesurer la fréquence. Une carte fille comprenant un connecteur BNC est connectée sur la carte FPGA. Le signal à analyser est disponible sur la broche **PIN_A13**

- Modifier votre fichier BDF pour injecter le signal disponible sur la carte fille
- Faite valider par l'enseignant
- Synthétiser et programmer la carte FPGA
- Injecter via le GBF **un signal TTL**

Validation 7. Faite valider par l'enseignant

Répondre aux questions suivantes :

Question 12. Comment peut on étalonner ce montage ?

Question 13. Quelle est la plage de fréquence mesurable précisément ?

Question 14. Quelles sont les incertitudes de mesure ?

Question 15. Quels sont les inconvénients de ce montage ?

Question 16. Remplir le tableau de mesure ci-dessous

Fréquences	N théorique	N pratique
------------	-------------	------------

Nom & Prénom :

Groupe :

5Hz		
50Hz		
100Hz		
500Hz		
1kHz		
5kHz		
10kHz		
50kHz		
100kHz		
500kHz		
1MHz		
5MHz		

Question 17. Tracer sur un même graphique, les courbes $N_{\text{théorique}}=f^{\circ}(\text{fréquence})$

et $N_{\text{expérimental}}=g^{\circ}(\text{fréquence})$.

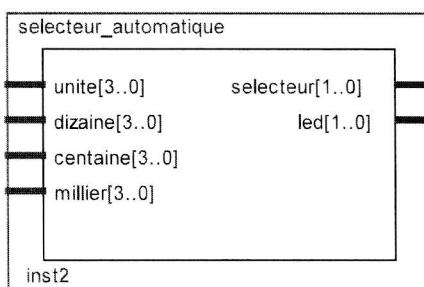
Question 18. Conclusion

Partie optionnelle

L'architecture du fréquencemètre proposée, permet de mesurer la fréquence à un facteur d'échelle 2 près. Si on veut une mesure directe, quelles sont les modifications à apporter ?

- Proposer les modifications à apporter
- Test et valider votre solution si le temps vous le permet.

Dans un second temps, on désire réaliser la partie automatique du sélecteur de calibre. On utilisera pour cela un second multiplexeur lpm_mux1 qui prendra en entrée, soit les commandes manuelles de sélection du calibre (les switch SW8 et SW9), soit les sorties du circuit selecteur_automatique. La commande du multiplexeur lpm_mux1 sera le switch SW1. pour commander le multiplexeur lpm_mux0 (sélecteur de la fréquence).



Le circuit selecteur_automatique calcul le meilleur calibre possible en fonction du N en sortie du compteur BCD. Les sorties sont, la valeur du sélecteur de calibre et l'affichage sur 2 leds de la valeur du sélecteur.

Question 19. Proposer un algorithme

Question 20. Proposer une architecture VHDL de cette entité

Question 21. Simuler et vérifier le comportement.

Question 22. Consigner dans votre rapport, les développements et les résultats de simulation.

Nom & Prénom :

Groupe :

Question 23. Insérer et tester pratiquement les performances du fréquencemètre à calibre automatique