TP N°1_Part2
Prise en main de Quartus II V13
Introduction à la synthèse logique
Programmation d'un FPGA Cyclone II
Durée: 2h30 max

Ce TP est la suite du TP N°1_Part1.

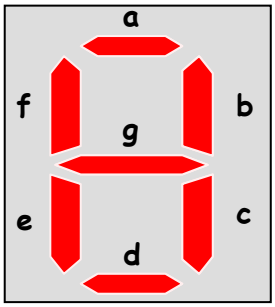
A NE PAS OUBLIER !
Un compte rendu de TP est à déposer dans l'ENT (dépôt d'un devoir) (TP1_Part2_QII_VotreNom.pdf) ainsi que le projet complet au format zip.

Cette seconde partie à **réaliser en autonomie** doit vous permettre de mettre à profit la première. Bien sûr la difficulté quelque peu accrue mais surmontable!

Objectifs à atteindre.
1) Écrire une description « comportementale » d'un :
A) Un décodeur BCD-7seg,
B) D'un compteur BCD 4bits piloté par une horloge de 1s
2) Mettre en œuvre est un chronomètre sur 90s.
Vous utiliserez donc deux afficheurs 7 segments disponibles sur la carte DE2, HEX0 et HEX1 pour visualiser le temps. Au terme des 90s le comptage s'arrête et une LED s'allume pour préciser la fin du chronométrage.

II. A) Afficheur 7 segments.

On souhaite réaliser le circuit de décodage d'un afficheur 7 segments avec un FPGA Cyclone II EP2C35F672C6. Le projet sera nommé ***Dec7seg.***
Les segments (LEDs rouges ici) d'un afficheur « 7 segments » sont repérés de **a** à **g** selon le dessin ci-dessous :



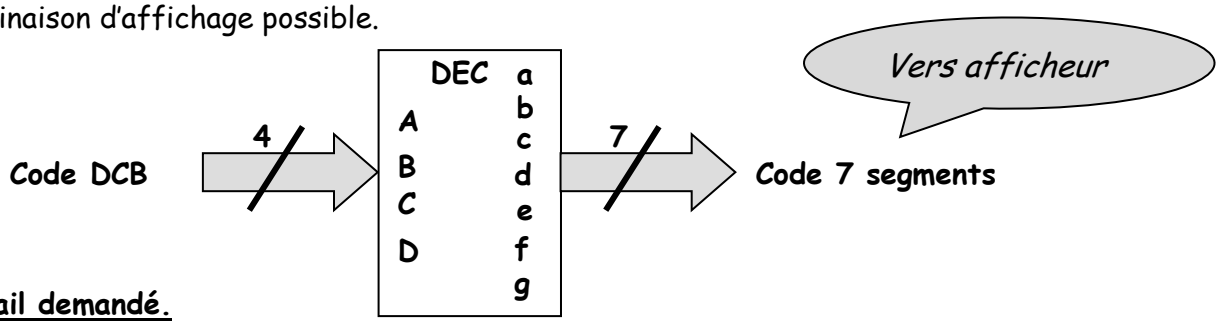
Selon les modèles (Afficheur à anode ou cathode commune) un NLB ('0') ou NLH ('1') allume un segment.
Dans notre cas l'application d'un **NLB ('0')** sur un segment l'**allumera**.
Le vérifier en consultant la documentation de la carte DE2 (Ressources_DE2).
→ ***Faire valider par votre enseignant.***

Décodeur ou Transcodeur.

Les entrées du décodeur, au nombre de 4, sont nommées en général **A, B, C, D**. Elles représentent la valeur **D**écimal **C**odé en **B**inaire (Code **DCB** ou BCD en anglais) du chiffre à afficher (par exemple 12 sera codé 0001 0010 en DCB).

Note : *On admettra que toute combinaison d'entrée strictement supérieure à 9 affiche la lettre **E** pour **Erreur**. Il faudra donc coder cette lettre !*

Le circuit décodeur produit sur ces 7sorties les informations qui définissent une combinaison d'affichage possible.



Travail demandé.

a) **Créer** un répertoire **TP1_Part2A** dans votre répertoire de travail **TP_QuartusII_V13**.

b) **Créer** sous **Quartus II** le projet dans ce répertoire de travail.

c) **Donne**r dans l'éditeur de texte de **QuartusII** une description comportementale du circuit décodeur en utilisant les instructions séquentielles ***case…when… end case***. Un ***process*** est donc utilisé… (*Revoir le cours si nécessaire*)

Attention le fichier vhdl doit avoir le même nom que l'entité correspondante.

Voici le début du fichier VHDL que vous renseignerez par vos noms, date et groupe:

```
library ieee ;
use ieee.std_logic_1164.all ;
-- Description comportementale d'un décodeur BCD 7segments
-- N : valeur à afficher,
-- SEG : 7 segments de l'afficheur.
-- Carte DE2 : EP2C35F672C6
-- Auteurs :
-- Date :
-- CERI Avignon Master1 SICOM
entity Dec7Seg IS
      port (
            N     : in integer range 0 to 15;
            SEG   : out std_logic_vector (6 downto 0)
            );
end Dec7Seg;

architecture Comp_DEC7S of Dec7Seg is
begin
    end process;
end Comp_DEC7S;
```

A compléter

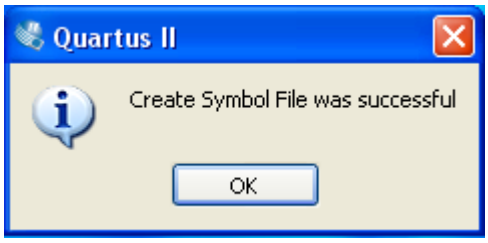
A compléter

→ **Faire valider par votre enseignant.**

Contrairement à ce qui a été fait dans la prise en main vous allez créer un « module » (composant) décodeur BCD 7 segments. Pour cela :

- **Aller** dans ***Project Navigator*** et **effectuer** un « click droit » sur le fichier **vhd**,
- **Sélectionner** ***Create/update*** puis ***Create Symbol File For Current File***

La fenêtre ci-dessous doit apparaître. Elle précise qu'un fichier d'extension **bsf** (**b**lock **s**ymbol **f**ile) a été créé avec succès !



Pour placer le « composant » Dec7Seg dans un schéma procéder comme suit :

- **Aller** dans le menu File puis **choisir** ***New***
- **Sélectionner** ***Block Diagram/Schematic File***

→ L'éditeur de schéma s'ouvre.

Attention il faut Enregistrer le fichier d'extension **bdf** sous un nom différent de celui du fichier vhdl. Sinon vous aurez une erreur de compilation.

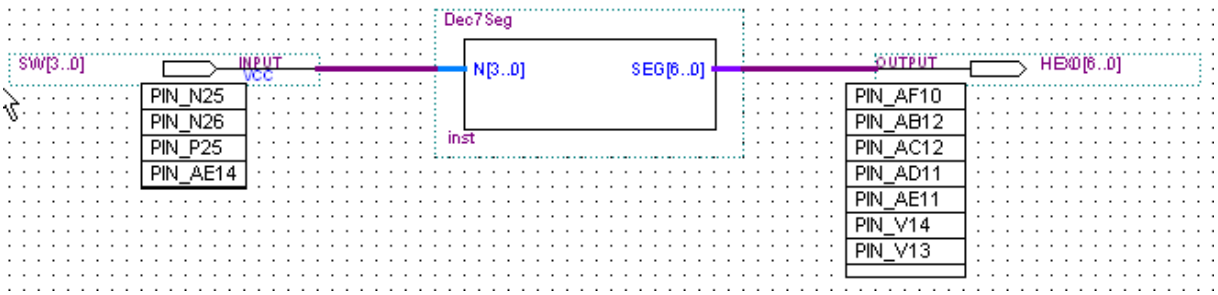
→ **Nommer-le « Deco7Seg »** !

→ Le fichier **bdf** s'ajoute alors à la liste des fichiers du projet.

Pour placer le composant sur le schéma :

- **Cliquer** 2 fois sur dans l'éditeur de schéma,
- **Sélectionner** dans ***Project*** le fichier **Dec7seg.bsf**,
- **Placer**-le sur le schéma ;
- **Consulter** la documentation de la carte **DE2** (User manual Version1.6) pour réaliser l'assignement de **N** (*input*) et **SEG** (*output*) respectivement sur **SW0** à **SW3** et **HEX0**. (*Reprendre la première partie sinon*).

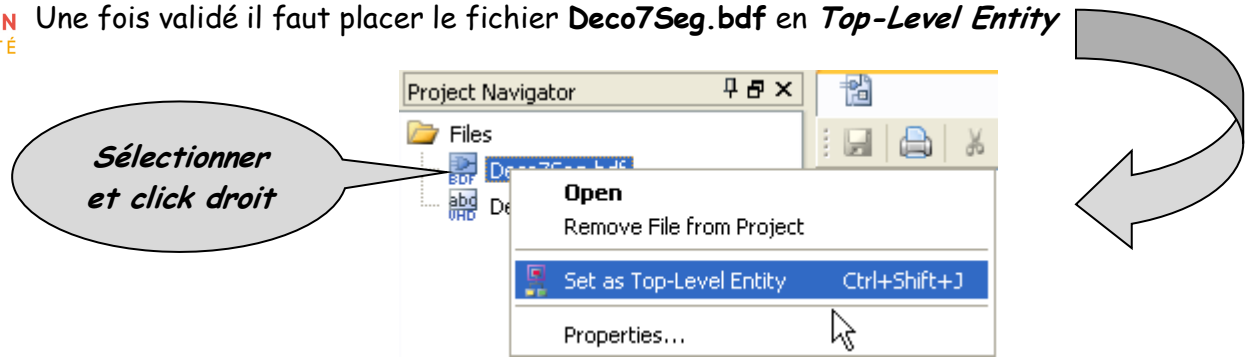
Si votre travail est correctement fait vous devez obtenir le résultat suivant :



Attention à l'ordre des segments dans votre description…

→ **Faire valider par votre enseignant.**

Une fois validé il faut placer le fichier **Deco7Seg.bdf** en ***Top-Level Entity***



d) **Lancer** la compilation du projet afin d'obtenir le fichier **sof**.
Ne pas tenir compte des nombreux warnings !

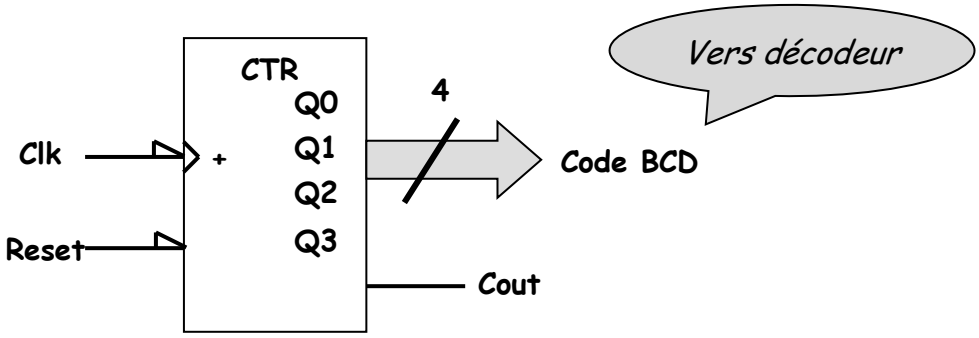
- **Vérifier** dans le menu ***Assignments*** > ***Pin Planner*** que les signaux de « connexion » avec la carte **DE2** se sont bien réalisés :

Node Name	Direction	Location
HEX0[6]	Output	PIN_V13
HEX0[5]	Output	PIN_V14
HEX0[4]	Output	PIN_AE11
HEX0[3]	Output	PIN_AD11
HEX0[2]	Output	PIN_AC12
HEX0[1]	Output	PIN_AB12
HEX0[0]	Output	PIN_AF10
SW[3]	Input	PIN_AE14
SW[2]	Input	PIN_P25
SW[1]	Input	PIN_N26
SW[0]	Input	PIN_N25

→ **Faire valider par votre enseignant.**

e) **Programmer** le FPGA sur la carte **DE2**.
f) **Effectuer** les tests de bon fonctionnement.
→ ***Faire valider par votre enseignant.***
→ ***Fermer le projet Dec7Seg.qpf.*** Menu ***File*** puis ***Close Project.***

II.B) <u>Synthèse-Simulation et programmation d'un compteur BCD 4 bits.</u>
Le compteur possède :
- Une RàZ <u>asynchrone</u> active sur **NLB** (signal **Reset**),
- Incrémentation du compteur sur **front descendant** (signal **Clk**),
- Une sortie de retenue (signal **Cout**).



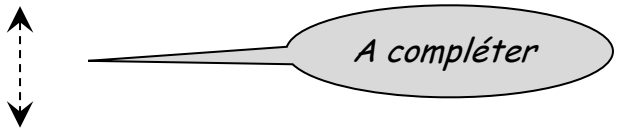
Travail demandé.

a) **Créer** un répertoire **TP1_Part2B** dans votre répertoire de travail principal. Créer sous **QuartusII** le projet ***CTR4Bits_BCD*** dans ce répertoire de travail.

b) **Donner** dans l'éditeur de texte de **QuartusII** une description comportementale du circuit compteur en utilisant obligatoirement un ***process*** .

(*Revoir le cours si nécessaire*).

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

-- Description comportementale du compteur BCD 4 bits
-- Clk          : Horloge active sur Front descendant,
-- Reset        : Remise a zéro Asynchrone,
-- Cout         : Sortie retenue,
-- Q            : mot de sortie du compteur
-- Carte DE2    : EP2C35F672C6
-- Fichier vhdl :CTR4Bits_BCD.vhd
-- Auteur       :
-- Date         :
-- CERI Avignon Master1 SICOM

entity CTR4Bits_BCD is
     port( Clk            : in std_logic;
           Reset          : in std_logic;
           Cout           : out std_logic;
           Q              : out std_logic_vector(3 downto 0));
end CTR4Bits_BCD;

architecture Comp_CTR4Bits_BCD of CTR4Bits_BCD is
signal ValCnt : std_logic_vector(3 downto 0);
begin
```




```
end Comp_CTR;
```

c) **Simulation fonctionnelle du compteur BCD.**

Nous allons utiliser pour cela un outil de simulation assez simple et pratique:

University Programm VWF.

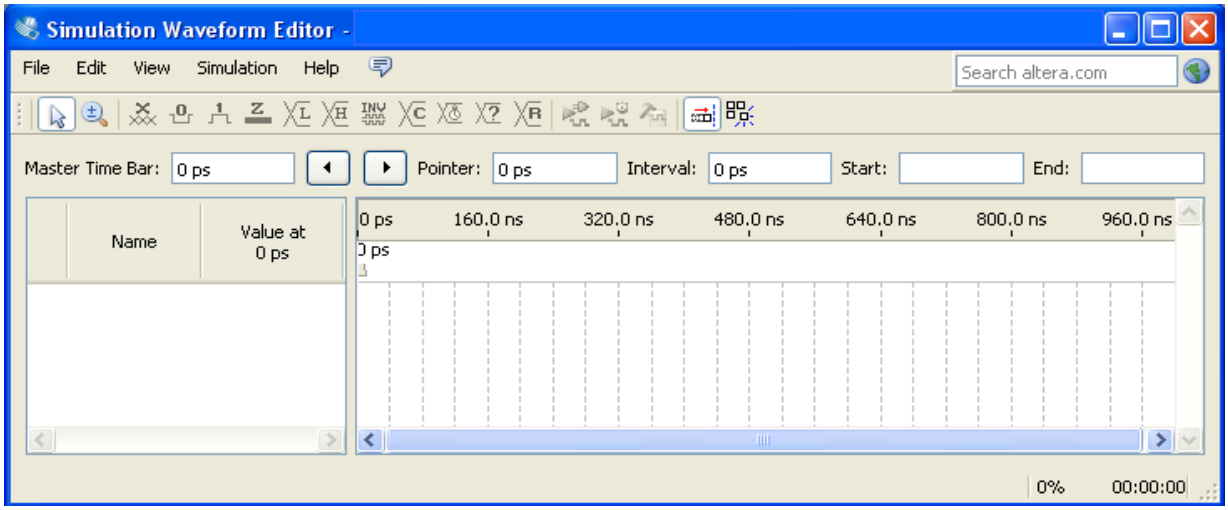
N'oublier pas de placer le fichier *CTR4Bits_BCD.vhd* en Top Level entity !

Etape N°1 : Préparer les signaux stimuli.

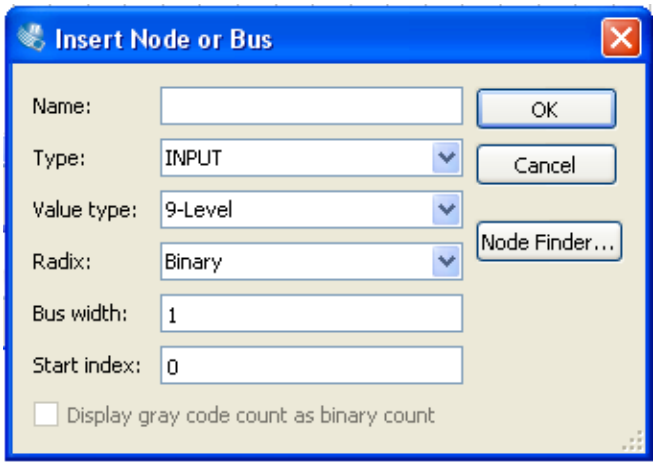
- **Compiler** le fichier **CTR4Bits_BCD.vhd**,
- **Dérouler** le menu ***File*** de QuartusII choisir ***New***,
- **Sélectionner dans *Verification/Debugging Files*** l'item ***University Program VWF***.

Puis valider par ***OK***.

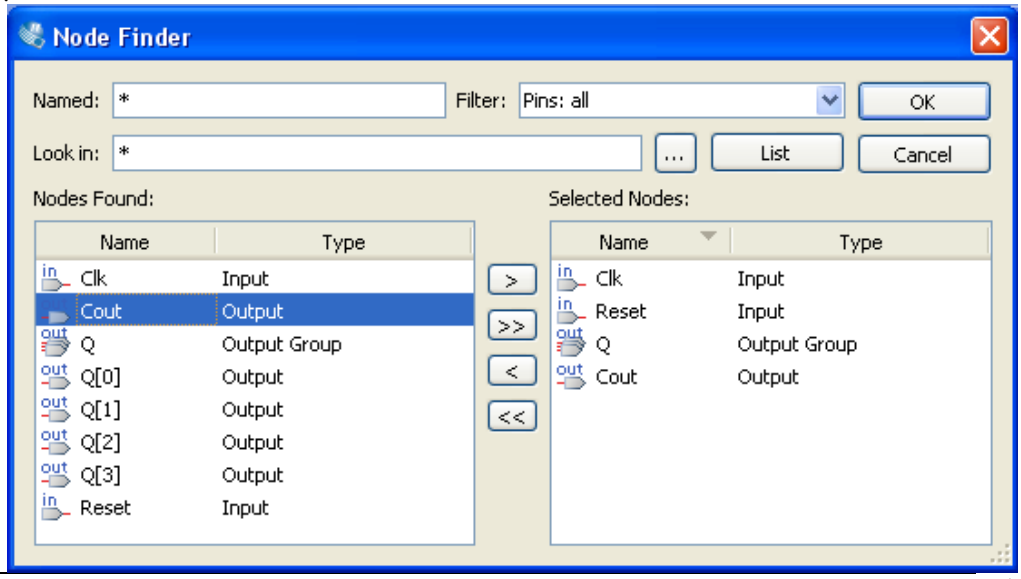
L'éditeur de simulation s'ouvre :



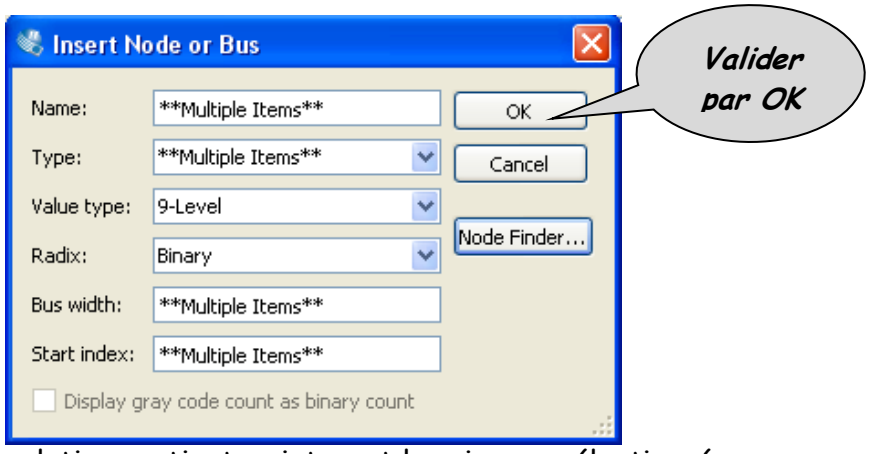
- Dans le menu ***Edit*** **choisir** ***Inser Node or Bus…***
- **Cliquer** sur ***Node Finder…***



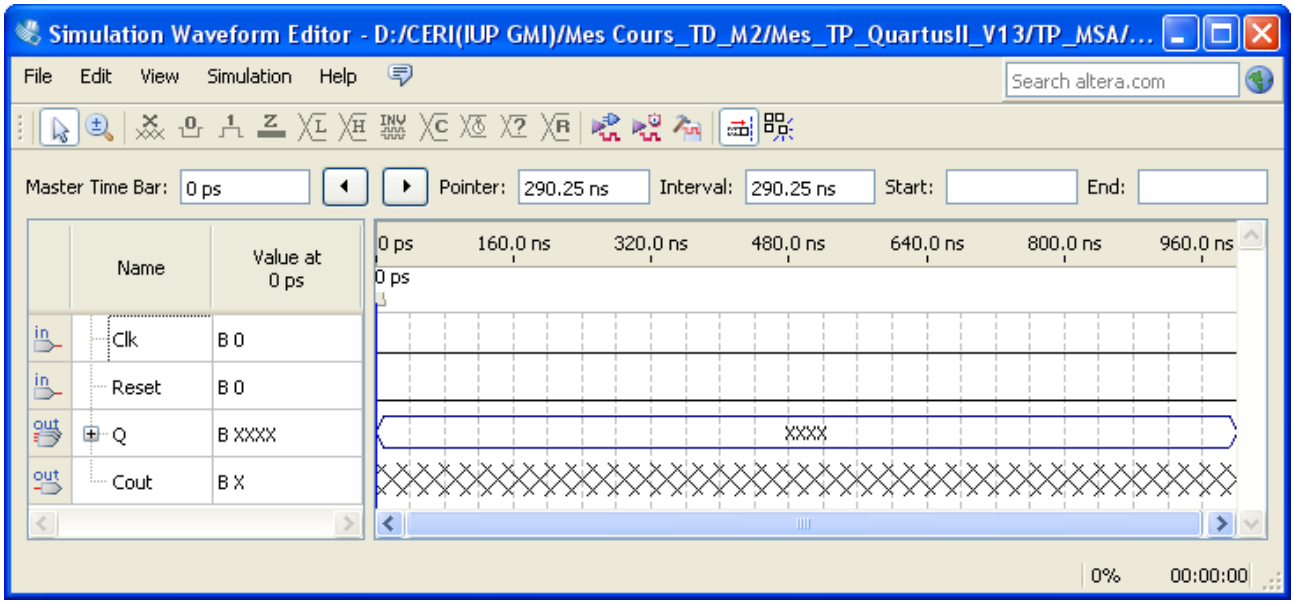
- **Cliquer** sur ***List***,
- **Sélectionner** les signaux comme représentés sur la capture d'écran ***Node Finder***
- **Valider** par **OK**.



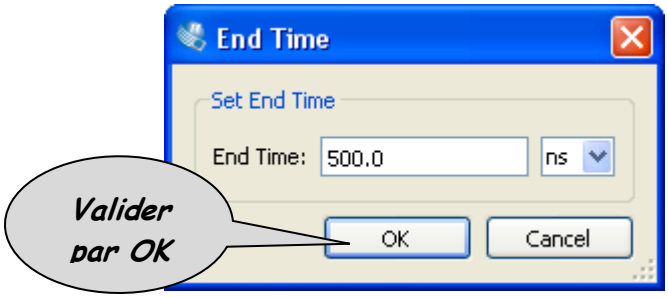
Vous retrouvez la fenêtre ***Insert Node or Bus*** complétée comme ci-dessous :



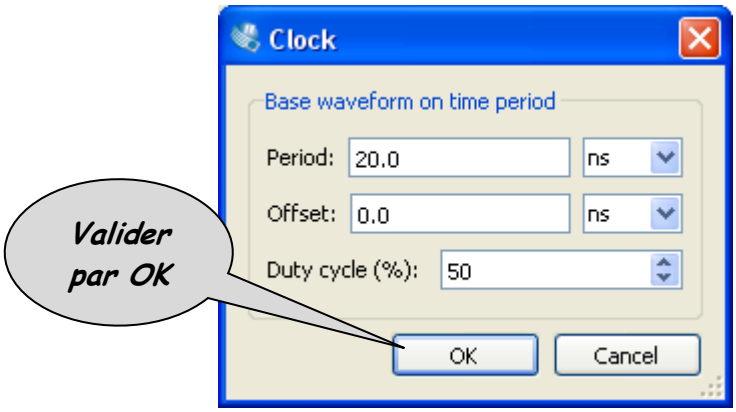
Votre éditeur de simulation contient maintenant les signaux sélectionnés :



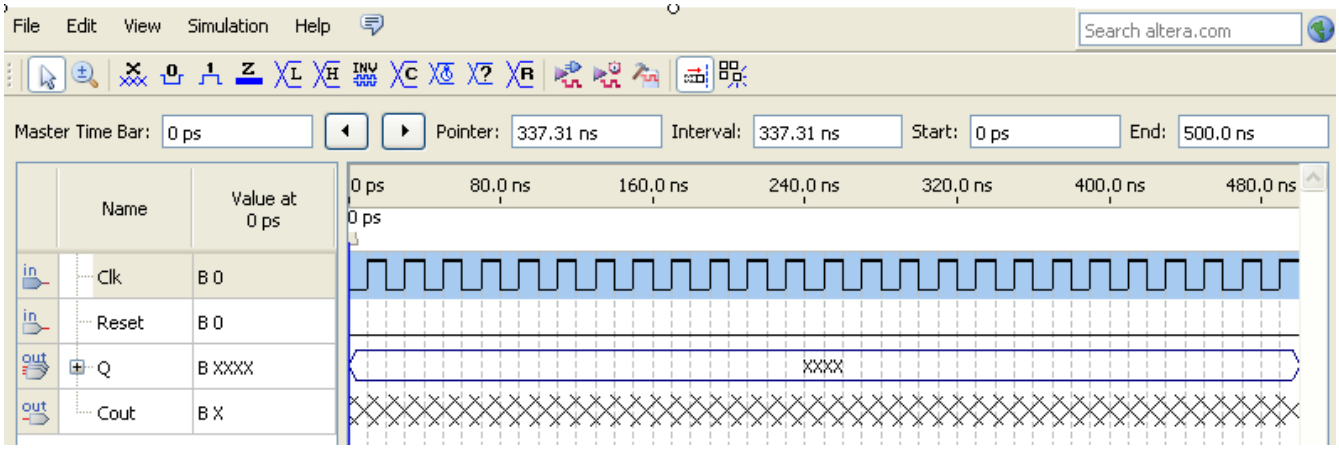
- Dans le menu ***Edit*** choisir ***Grid Size***. **Vérifier** que ***Period*** est de **5ns**.
- Dans le menu ***Edit*** choisir ***Set End Time***. **Donner** une valeur de **500ns**



- **Sélectionner** le signal ***Clk en cliquant dessus***. Une zone bleue claire apparaît. Un « clic droit » **sur cette zone**, ouvre un menu contextuel. **Choisir *Value*** puis ***Overwrite Clock*** et compléter comme indiqué page suivante :



Votre signal **Clk** est dessiné :

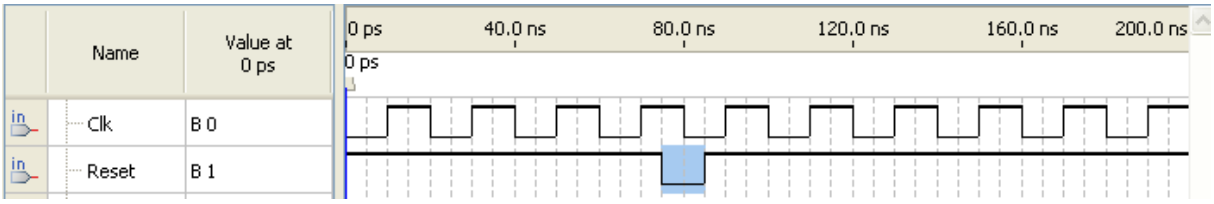


Le signal Reset est actif sur **NLB**. Pour mettre en évidence le bon fonctionnement asynchrone de celui-ci on va admettre qu'à 75ns un NLH d'une durée de 10ns se produit.

- **Sélectionner** le signal **Reset**. **Cliquer** sur le bouton 

- **Créer** une *zone bleue* entre **75ns** et **85ns** puis cliquer sur le bouton 
→ Utiliser l'outil **loupe** pour obtenir plus de précision.

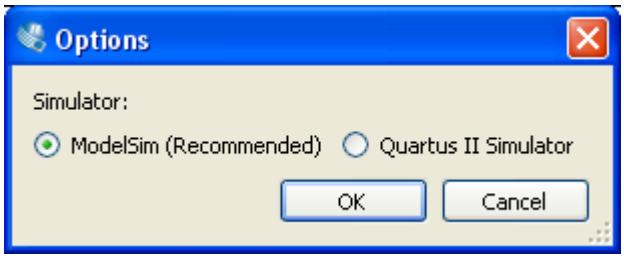
Vous devez obtenir ceci :
→ **Start** et **End** doivent indiquer 75.0ns et 85.0ns respectivement.



→ Faire valider cette première étape par votre enseignant.

Etape N°2 : Lancer une simulation fonctionnelle.

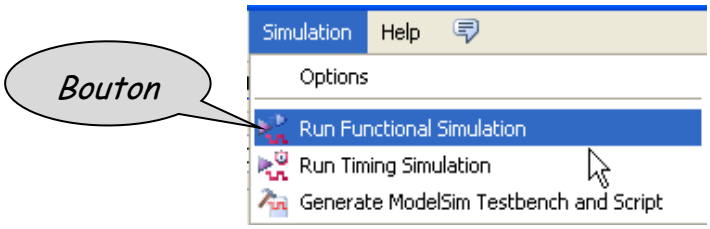
- **Dérouler** le menu ***Simulation choisir Options. Opter*** pour ModelSim(Recommended).



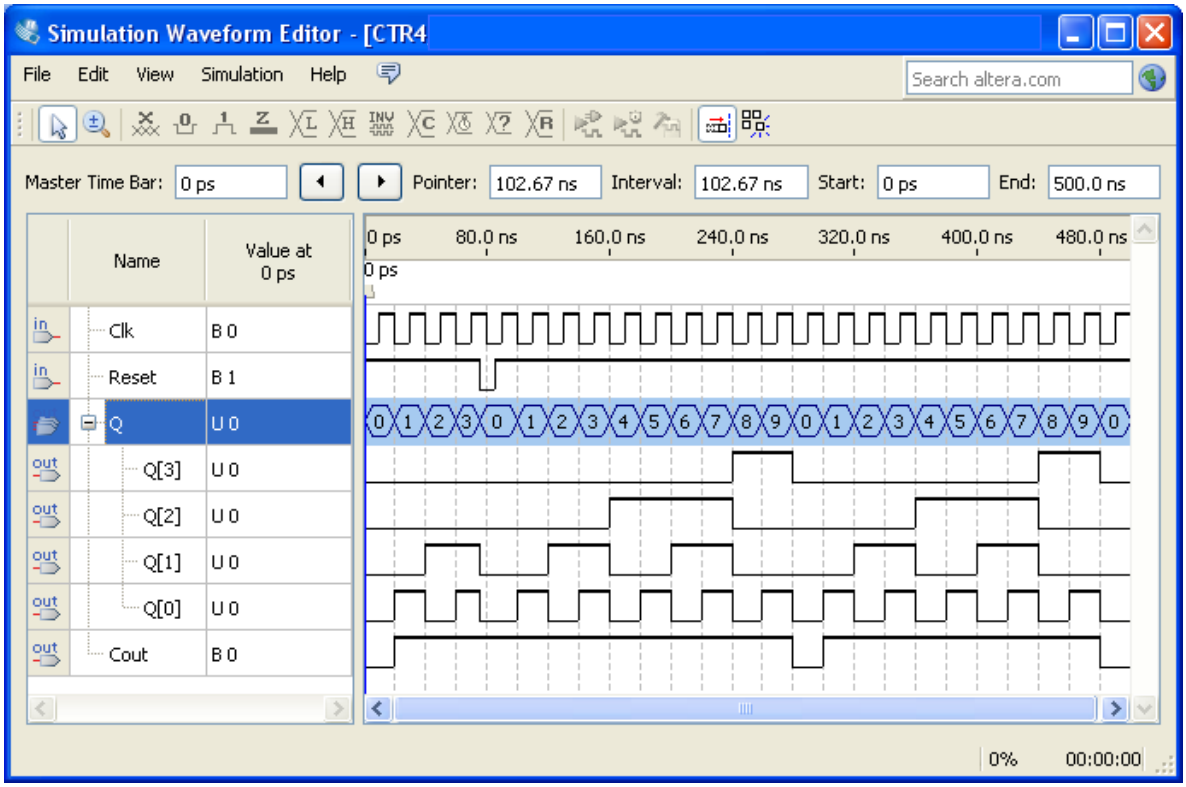
- **Sauvegarder** votre travail sous le nom **Sim_CTR4Bits_BCD** l'extension est ***vwf*** dans le répertoire \simulation\modelsim.

Important : Retourner provisoirement sur QuartusII et relancer une compilation du fichier CTR4Bits_BCD.vhd.

- **Lancer** la simulation. Dans le même menu **choisir** l'option ***Run Functionnal Simulation*** *ou* ***appuyer*** *sur le bouton correspondant dans la barre des boutons.*



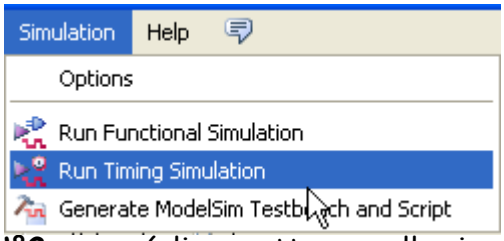
Une fenêtre s'ouvre montrant la progression de la simulation…
Si tout s'est bien passé vous obtenez le résultat suivant.



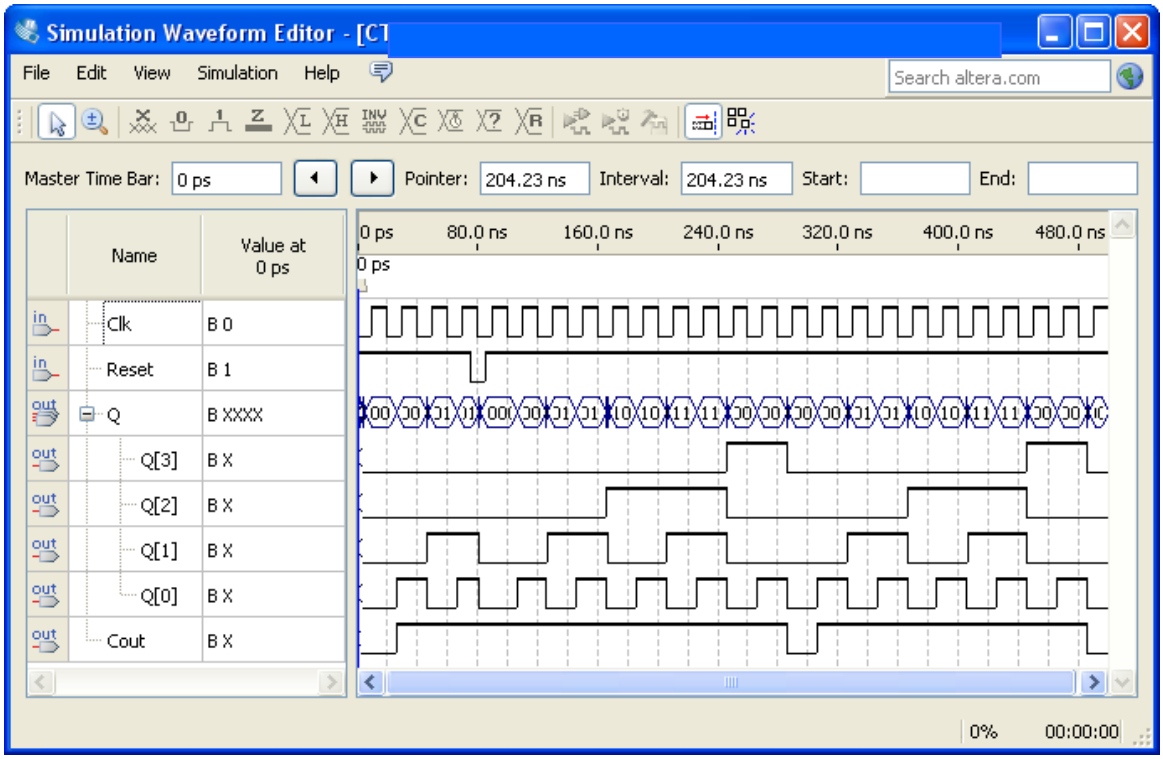
- **Analyser** le résultat obtenu puis (ne pas fermer la fenêtre obtenue) :

→ Faire valider cette seconde étape par votre enseignant.

Etape N°3 : Lancer une simulation temporelle.



- **Reprendre l'étape N°2** pour réaliser cette nouvelle simulation.
- **Développer** le chronogramme de **Q**



- **Analyser** le résultat obtenu puis (ne pas fermer la fenêtre obtenue)
  Quelle différence constatez-vous ?

→ ***Faire valider cette seconde étape par votre enseignant.***

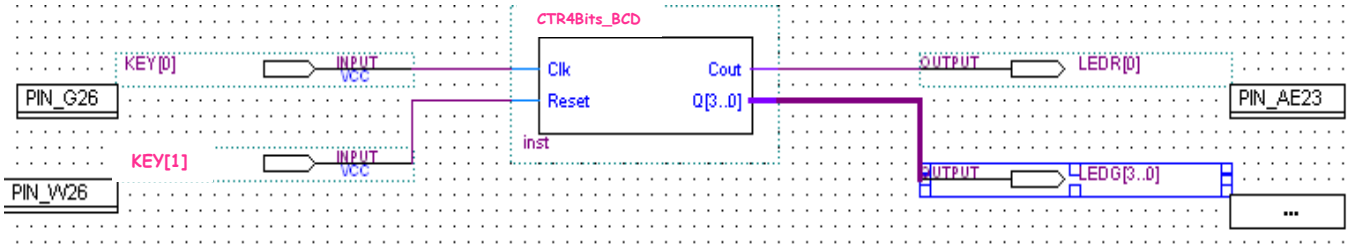
Fermer le simulateur et revenir sur QuartusII.

d) **Créer** le schéma du compteur : ***CTR4BitsBCD.bdf***. Après sa création ne pas oublier de le placer en ***Top-Level Entity.***

e) Pour **valider** votre travail sur la carte DE2 vous aller assigner :

- Le Bouton KEY0 à **Clk**,
- Le bouton KEY1 à **Reset**,
- La LEDR0 à **Cout**,
- Les LEDG0 à LEDG3 aux sorties **Q0** à **Q3** respectivement.

Si votre travail est correctement fait vous devez obtenir le résultat suivant :



f) **Lancer** la compilation du projet afin d'obtenir le fichier **sof**.

Ne pas tenir compte des warnings !

- **Vérifier** dans le menu ***Assignments > Pin Planner*** que les signaux de « connexion » avec la carte **DE2** se sont bien réalisés. **Attention soyez vigilant** !

→ Faire valider par votre enseignant.

ATTENTION avant de programmer le FPGA vérifier IMPERATIVEMENT que les « unused Pins » sont configurées « As input tri-stated » … !!!

g) **Programmer** le FPGA sur la carte **DE2**.

h) **Effectuer** les tests de bon fonctionnement.

→ Faire valider par votre enseignant.

→ Fermer le projet CTR4Bits_BCD.qpf. **File** puis **Close Project**

II.C) Fusion **de compteur BCD et Décodeur BCD 7 segments.**

Dans cette troisième approche vous allez créer un projet qui utilisera 2 décodeurs BCD 7 segments (II.A) et 2 compteurs BCD (II.B).

A chaque appui sur le bouton KEY0 (**Clk**) l'affichage est incrémenté de 0 à 99 d'une unité. La RàZ (KEY3) restera Asynchrone.

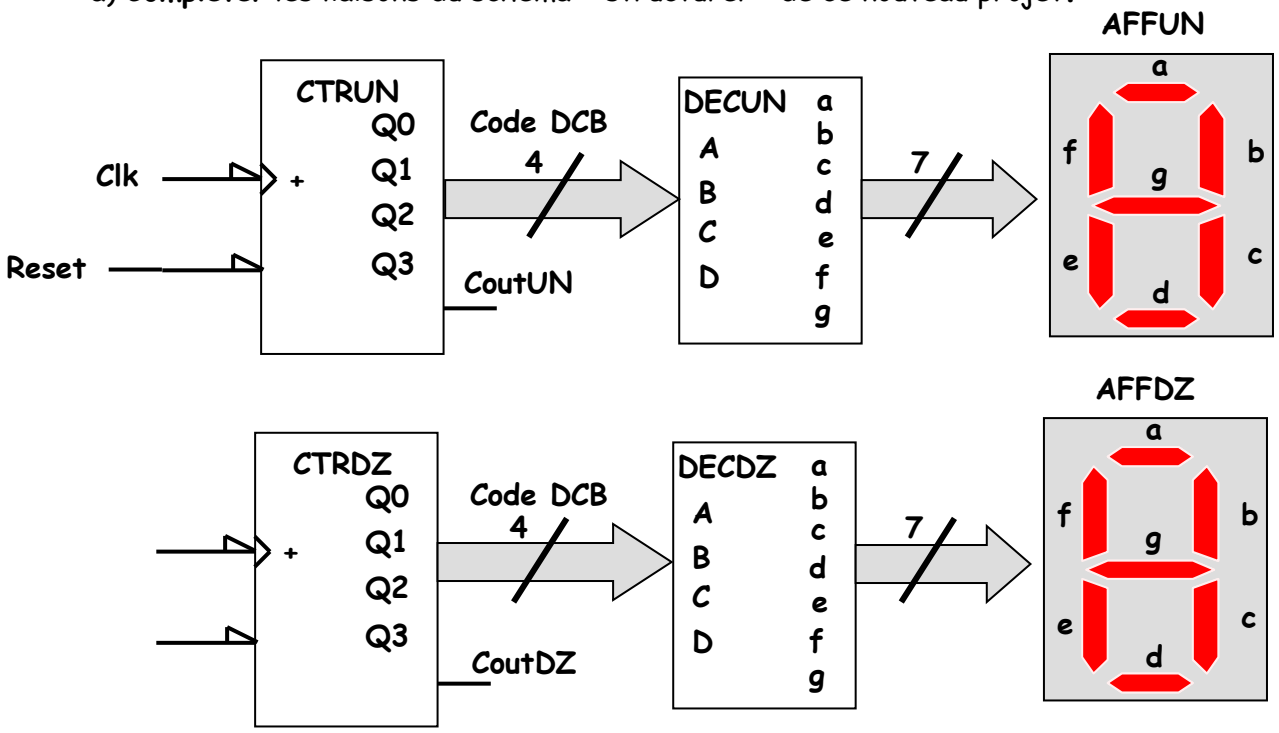
Le projet portera le nom de **AFF_V1** et se trouvera dans le dossier **TP1_Part2C**

- Dans ce projet **importer** les fichiers **vhdl CTR4Bits_BCD** et **Dec7Seg** des parties **II.A** et **II.B**.
- Pour chacun d'entre eux **créer** les fichiers **bsf** correspondant,

A partir de cela vous pouvez commencer le travail demandé page suivante.

Travail demandé.

a) **Compléter** les liaisons du schéma « structurel » de ce nouveau projet.



→ ***Faire valider par votre enseignant.***

b) **Placer** les instances des « composants » dans l'éditeur de schéma de QuartusII. **Dessiner** le schéma **AFFV1.bdf**. Ne pas oublier d'assigner les E/S !

→ ***Faire valider par votre enseignant.***

c) Une fois le schéma validé **effectuer** sa compilation et la programmation du **FPGA** sur la carte DE2.

→ ***Faire valider par votre enseignant.***

d) Selon la façon dont vous avez codé le **Cout** du compteur BCD un défaut peut apparaître dans le fonctionnement attendu. Le corriger jusqu'à obtenir le résultat demandé.

→ ***Faire valider par votre enseignant.***

e) **Créer** le fichier ***AFFV1.vhd***.

Quel type de description propose QuartusII ?

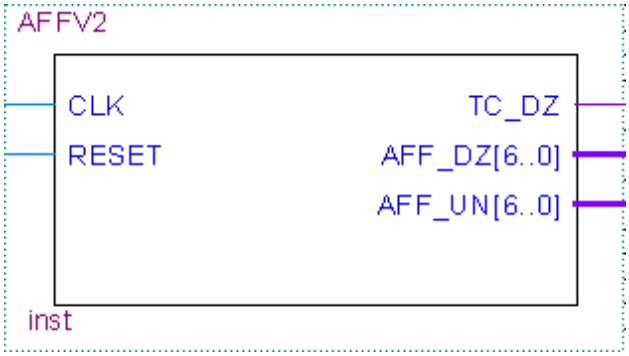
II.D) <u>Réalisation finale.</u>

Notre objectif est de réaliser un chronomètre de 90s sur 2 digits, nous aurons besoin d'une horloge de 1s, qui servira donc de base de temps.

La carte DE2 possède un oscillateur à quartz très stable de 50MHz (CLOCK_50).

Dans cette partie le répertoire du projet sera **TP1_Part2D** et vous nommerez le projet **Chrono**.

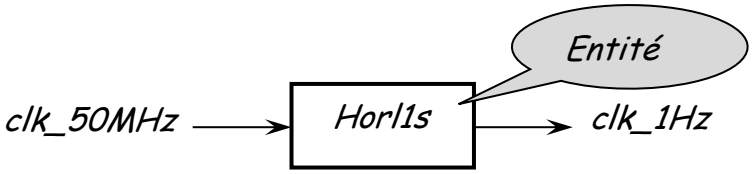
- **Réaliser à partir de AFFV1.vhd,** un module unique de comptage/d'affichage nommé AFFV2.bsf (instance) ressemblant au modèle ci-dessous :



Tester rapidement sur la carte DE2 votre module en l'incluant dans un schéma sauvé sous le nom **Chrono.bdf**

ATTENTION !
Ne pas oublier de la placer en Top Level Entity et de mettre les broches inutilisées en haute impédance !
→ Faire valider le fonctionnement par votre enseignant

II.D.1) Produire une horloge de 1s précise.
Il existe plusieurs possibilités pour réaliser notre base de temps. Je vous propose ici de la créer sous la forme d'une description comportementale VHDL.
Ci-dessous la « boite noire » à décrire :



On constate « facilement » que la « fonction de transfert » de cette « boite noire » est une **division de fréquence** dont le principe à décrire est assez simple.
Dans notre cas on se pose la première question suivante :
Quel est le rapport de division entre 50MHz et 1Hz ?
Compléter les blancs
→ La réponse est naturellement _____.
S'agissant de la réalisation et de l'utilisation de signaux d'horloge ces derniers sont en général actifs sur fronts. Nous admettrons ici qu'ils le sont sur fronts montants.
Admettons également que **clk_1Hz** est au **NLB** au départ.
La sortie **clk_1Hz** doit rester dans cet état _____ de fronts de **clk_50MHz** et en changer (passage au NLH) pour y rester _____ de fronts de **clk_50MHz** et ainsi de suite. (50Mhz→ T=20ns → _____x20ns=_____s.
Nous avons donc ici besoin d'un diviseur constant de _____.
Le langage VHDL permet de déclarer dans un ***process*** des variables (mot clef **VARIABLE**) et des constantes (mot clef **CONSTANT**).

Question1 : *Proposer un algorigramme traduisant le **process** à décrire.*

Question2 : ***Compléter** la description ci-dessous (**Horl1s.vhd**) et tester directement sur DE2, votre base de temps avec AFFV2. N'oubliez pas de créer un **fichier bsf**…*

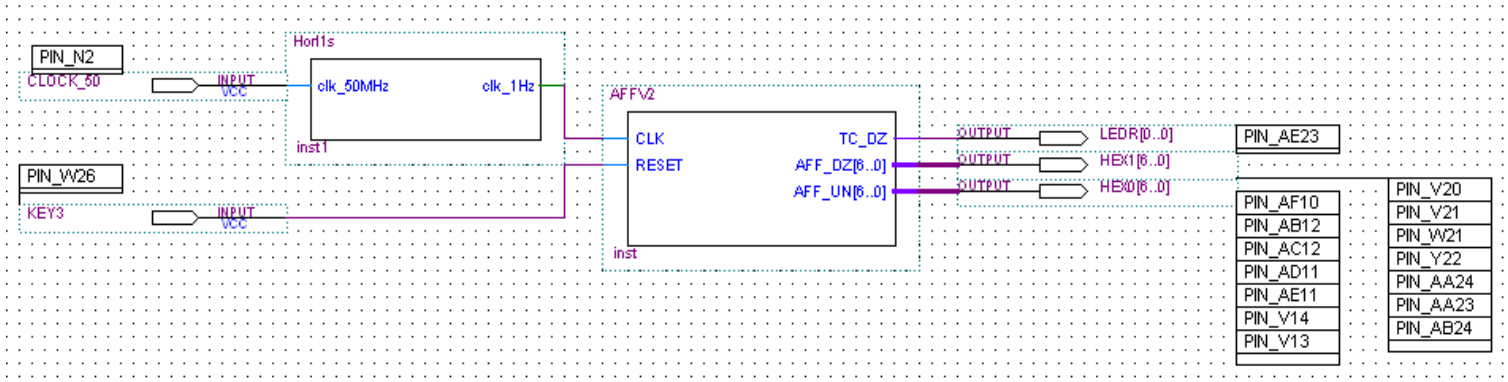
→ Faire valider par votre enseignant.

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

ENTITY Horl1s IS
 PORT
      ( clk_50MHz : IN std_logic;
        clk_1Hz: INOUT std_logic); -- pour l'horloge de periode 1s );
END Horl1s;

ARCHITECTURE behavior OF Horl1s  IS
BEGIN

      PROCESS (________)
      VARIABLE cnt      : integer range 0 to __________
      CONSTANT diviseur : integer :=____________

      BEGIN

      END PROCESS;

END behavior;
```

A compléter

A compléter

Résultat attendu :



II.D.2) Arrêter le compteur à 90s.

En l'état notre chronomètre évolue de 0 à 99.

Question3 :

***Proposer** une modification de **Horl1s.vhd** et donc de **Horl1s.bsf** permettant de **stopper automatiquement** le comptage à 90s. **Sauver** le nouveau fichier VHDL en **Horl90s.vhd**. **Tester** sur carte DE2,*

→ Faire valider par votre enseignant.

Fin du TP1.