المدرسة الوطنية للعلوم التطبيقية
ⵜⵜⵍⵎ ⵜⴰⵏⴰⵎⵓⵔⵜ ⵏ ⵜⵎⴰⵡⴰⵍⵜ ⵜⵓⵏⵖⵓⵜ
Ecole Nationale des Sciences Appliquées de Fès

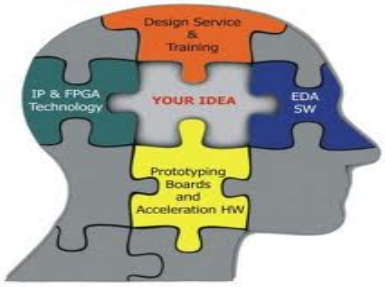
جامعة سيدي محمد بن عبد الله
ⵜⴰⵡⴰⵏⴰⵙⵜ ⵓⵙⵎⵎⵓⵔ ⵙⵉⴷⵉ ⵎⵓⵃⵎⴰⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

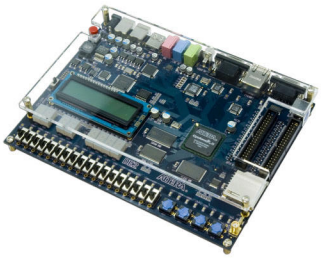
1ère Année GSEII

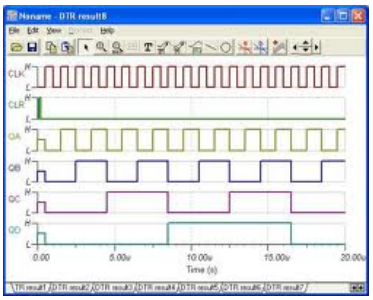
# Circuits Reprogrammables & Conception des Circuits Intégrés Numériques En VHDL

## TRAVAUX PRATIQUES

**TP-4 :** Simulation, Synthèse et Implémentation FPGA





**Pr : Anass MANSOURI**

# *SOMMAIRE*

# 1 - Présentation du TP

Ce TP consiste à spécifier, simuler puis synthétiser et finalement implémenter sur un circuit FPGA un compteur BCD (logique synchrone) et un décodeur 7-segments en VHDL (logique combinatoire).

➢ Vérifier le compteur par simulation logique (ModelSim).

➢ Synthétiser le projet complet sur Quartus.

➢ Implémenter le projet sur la carte DE2-70 d'ALTERA.

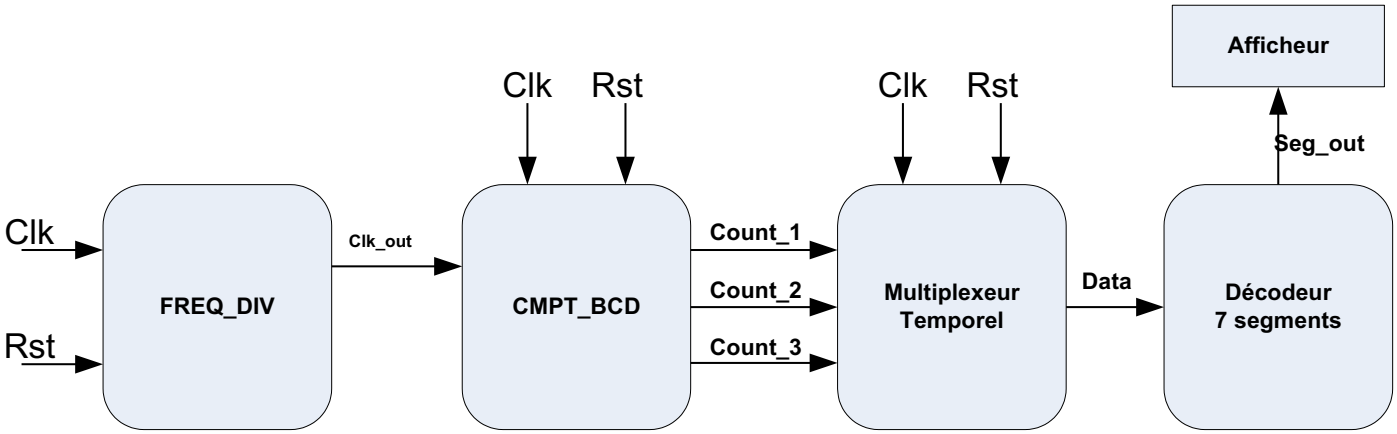
Ce TP doit afficher un compteur de 0 à 999 sur les afficheurs 7-segments de la carte de démonstration DE2-70.

Malgré sa simplicité, ce TP permet de mettre en œuvre l'essentiel de la méthodologie et des outils de conception.

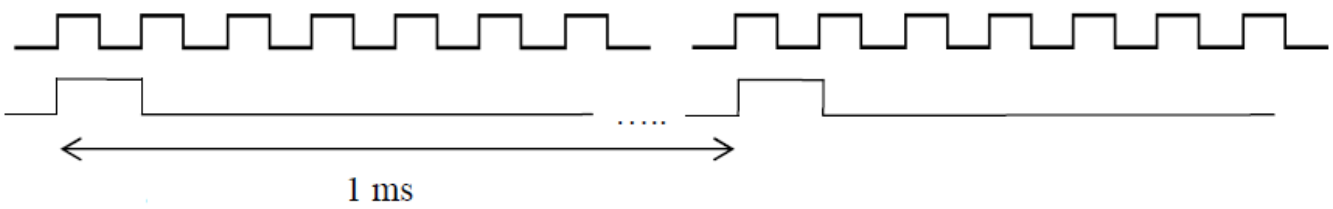
# 2 - Spécification du TP

L'horloge d'entrée du FPGA est une horloge de 27 MHz ou bien de 50 MHz. On désire compter de 000 à 999 au rythme d'une incrémentation toutes les « N » millisecondes, N étant paramétrable à l'instanciation par un paramètre « generic » : N. Le compteur doit repasser à 000 lorsque l'on se trouve à 999.

Voici le schéma descriptif de la fonctionnalité demandée :



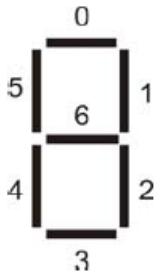

Le bloc **FREQ_DIV** fournit un signal **Clk_out** actif (niveau logique '1') toutes les 1 millisecondes durant un cycle d'horloge à 27 MHz ou à 50 MHz.



Le signal **Clk_out** arrive sur un port d'entrée du bloc **CMPT_BCD** et doit être utilisé pour réaliser l'incrémentation du compteur toutes les per_ms.
Il est hors de question d'utiliser **Clk_out** comme horloge (l'horloge est toujours Clk, l'horloge principale !!!) Il vous faudra réaliser un test sur l'état de **Clk_out**.
Le compteur BCD a 3 ports de sortie : un pour les unités, un pour les dizaines et un pour les centaines. Chacun de ces ports est bien sûr codé sur 4 bits, afin de pouvoir compter de 0 à 9 inclus ... (et pas au-delà).
Ces ports sont ensuite encodés par le bloc **SEVEN_SEG** pour pouvoir être affichés sur les afficheurs 7 segments de la carte DE2-70 Board. On notera que l'afficheur est multiplexé temporellement : on alterne rapidement les trois valeurs en conjonction avec un vecteur de sélection qui active un afficheur à la fois. Grâce à la persistance rétinienne, on a l'impression d'un affichage simultané des quatre chiffres.

## 2-1 Décodeur 7-segments

Voici le fonctionnement d'un afficheur 7-segments :
Les segments sont traditionnellement identifiés par les valeurs de 0 à 6. Dans notre module, la sortie est un vecteur croissant de 1 à 7 correspondant aux segments 0 à 6.




De plus, nous avons une entrée de sélection de polarité « Pol » :
Pol = 1 signifie des afficheurs actifs niveau 1
Pol = 0 signifie des afficheurs actifs niveau 0
Ainsi, pour afficher le chiffre 1, on devra établir SegOut = 0110000 si Pol=1 et 1001111 si Pol=0.

***Question*** :

- Créer les quarts répertoires (src, Test_bench, Modelsim, Quartus).
- Faire la description VHDL du module. Simuler le bloc en Modelsim.
- Ecrire un Script de compilation et de simulation.
- Lancer une synthèse Quartus de votre code, interpréter les résultats.
- Implémenter et valider le bloc **SEVEN_SEG sur FPGA.**

## 2- 2 Le compteur BCD en VHDL

Codez un compteur BCD qui compte toutes les N x 1 ms de 000 à 999. N'oubliez pas que le compteur doit repasser à 000 lorsque l'on est à la valeur 999...

***Question*** :

- Créer les quarts répertoires (src, Test_bench, Modelsim, Quartus).
- Faire la description VHDL du module. Simuler le bloc en Modelsim.
- Ecrire un Script de compilation et de simulation.
- Lancer une synthèse Quartus de votre code, interpréter les résultats.
- Pourquoi on ne peut pas valider le boc sur FPGA.

## 2- 3 Diviseur de fréquence

Le bloc **FREQ_DIV** permet de diviser la fréquence de l'horloge de l'entrée et fournit un signal **Clk_out** actif (niveau logique '1') toutes les 1 millisecondes. Ce bloc permet de rendre l'affichage des résultats de comptage visible sur FPGA, donc sa mission principale seulement de faciliter le processus de validation sur FPGA.

***Question*** :

- Créer les quarts répertoires (src, Test_bench, Modelsim, Quartus).
- Faire la description VHDL du module. Simuler le bloc en Modelsim.
- Ecrire un Script de compilation et de simulation.
- Lancer une synthèse Quartus de votre code, interpréter les résultats.
- Valider le bloc sur FPGA, la sortie Clk_out doit être connecté vers une LED de la carte DE2-70 board.

## 2- 4 Multiplexeur Temporel et afficheur

Ce bloc permet la multiplication temporelle de la sélection des afficheurs 7-segments, c-a-d on alterne rapidement les trois valeurs en conjonction avec un vecteur de sélection qui active un afficheur à la fois. Grâce à la persistance rétinienne, on a l'impression d'un affichage simultané des quatre chiffres. Ce multiplexage temporel est assuré par le bloc **Temp_MUX**

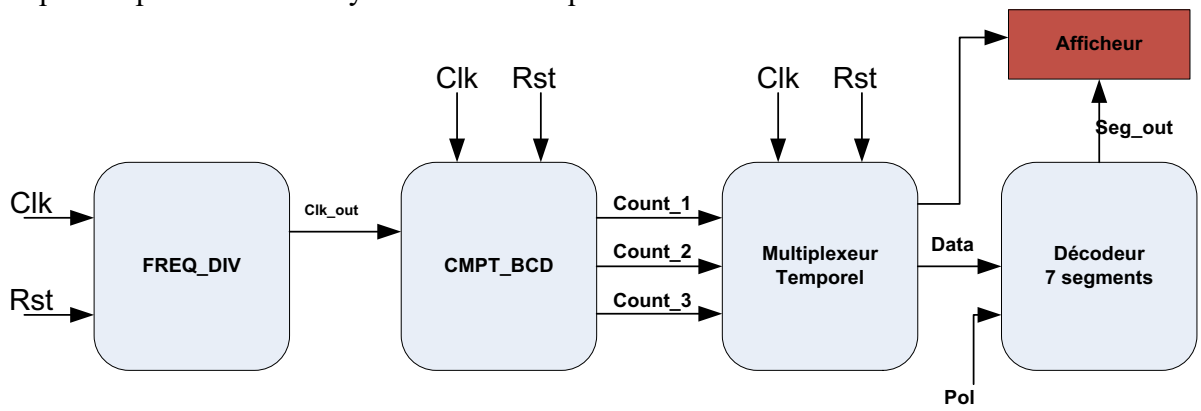
Le bloc Afficheur permet la gestion d'accès vers les afficheurs 7-segments da la carte DE2-70 board.

***Question*** :

- Créer les quarts répertoires (src, Test_bench, Modelsim, Quartus).
- Faire la description VHDL du module. Simuler le bloc en Modelsim.
- Ecrire un Script de compilation et de simulation.
- Lancer une synthèse Quartus de votre code, interpréter les résultats.
- Implémenter et valider le bloc **SEVEN_SEG sur FPGA.**

## 2- 5 Architectures Top

Assemblez le décodeur, le compteur BCD, le diviseur de fréquence et le multiplexeur temporelle pour réaliser le système électronique sur FPGA.




***Question*** :

- Créer les quarts répertoires (src, Test_bench, Modelsim, Quartus).
- Faire la description VHDL du module. Simuler le bloc en Modelsim.
- Ecrire un Script de compilation et de simulation.
- Lancer une synthèse Quartus de votre code, interpréter les résultats.
- Implémenter et valider le système **sur FPGA.**

## 3- Annexe

### 1- Squelette d'écriture d'un programme en VHDL

```
--************************************************************--
--                          ENSA FES                          --
--                        Filière :                           --
--************************************************************--
--Title         : Le nom du programme
--TP            : Le nom du TP
--Block         : Le nom du bloc
--************************************************************--
--File          : Le nom du fichier
--Authors       : Le nom du binôme
--Created       : La date de création
--************************************************************--
--Description :
--
--************************************************************--
--************************************************************
--                        Used Libraries
--************************************************************
--************************************************************--
--                        ENTITY Declaration
--************************************************************
--************************************************************--
                        RTL Description
--************************************************************
```