ELN2 : PROJET SCORING 2.0

Affichage de la durée et du score d'un match de football à l'aide d'un FPGA

Phase 1 : Gestion des afficheurs 7-segments

La **phase 1** du projet **Scoring 2.0** est consacrée au développement sous-bloc **display** dédié à la gestion des afficheurs 7-segments de la carte **NEXYS A7** de la société **Digilent**. Cette gestion est constituée de **logique combinatoire et de logique séquentielle**. Ce sous-bloc permettra de valider rapidement et visuellement le bon fonctionnement de chaque nouveau sous-bloc réalisé pour le système **chronoscore**, son développement est donc essentiel pour débuter la réalisation du projet.

1 Rappels sur l'architecture de chronoscore

1.1 TOP Module

La vue de niveau « TOP » du système **chronoscore** est donnée à la figure 1.

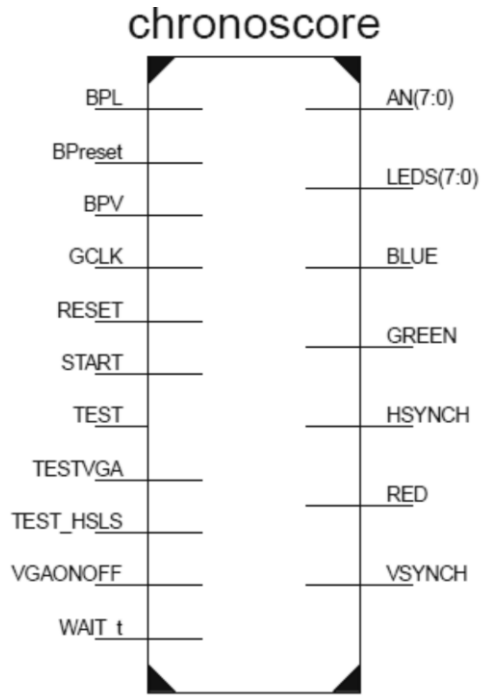


*Figure 1 : Vue TOP de l'entité **chronoscore***

1.2 Entrées et sorties du système

Le tableau 1, précise les signaux d'entrée de **chronoscore**, leur direction, leur localisation sur la carte **NEXYS A7** (voir « Présentation générale du projet » figure 2) et le port d'entrée sur le **FPGA**.

Signal	**Direction du signal**	**Elément de la carte NEXYS A7**	**Port du FPGA** *(fichier .ucf)*	**Description**
GCLK	Input	Quartz	E3	Oscillateur **Epson SG-8002JF** générant une fréquence de **100 MHz**
START	Input	SW0	J15	Démarrage du chronomètre
WAIT_t	Input	SW1	L16	Mise en pause du chronomètre
RESET	Input	BTN CPU RESET	C12	Remise à 0 du chronomètre
BPL	Input	BTNU	P18	Incrémentation du score pour l'équipe locale
BPV	Input	BTND	M18	Incrémentation du score pour l'équipe des « visiteurs »
BPreset	Input	BTNR	M17	Remise à zéro du score
VGA ONOFF	Input	SW4	R17	Activation de l'écran VGA
TEST VGA	Input	SW3	R15	Activation des images de test de l'écran VGA
TEST_ HSLS	Input	SW2	M13	Réservé
TEST	Input	BTNC	N17	Réservé

*Tableau 1 : Description des signaux d'entrée de **chronoscore***

Le tableau 2, précise les signaux de sortie de **chronoscore**, leur direction, l'élément concerné sur la carte **NEXYS A7** (voir « Présentation générale du projet » figure 2) et le port d'entrée sur le **FPGA**.

Signal	Direction du signal	Elément de la carte NEXYS A7	Port du FPGA *(fichier .ucf)*	Description
AN(0)	Output	Afficheurs 7-segments (anodes)	J17	AN[7 :0] : signaux de commande des anodes des afficheurs
AN(1)	Output		J18	
AN(2)	Output		T9	
AN(3)	Output		J14	
AN(4)	Output		P14	
AN(5)	Output		T14	
AN(6)	Output		K2	
AN(7)	Output		U13	
LEDS(0)	Output	Afficheurs 7-segments (cathodes)	T10	LEDS[6 :0] : signaux de commande des segments « a » à « g » LEDS[7] : signal de commande du point
LEDS(1)	Output		R10	
LEDS(2)	Output		K16	
LEDS(3)	Output		K13	
LEDS(4)	Output		P15	
LEDS(5)	Output		T11	
LEDS(6)	Output		L8	
LEDS(7)	Output		H15	
HSYNCH	Output	Port VGA	B11	Synchronisation horizontale
VSYNCH	Output		B12	Synchronisation verticale
RED	Output		A4	Contrôle des pixels de couleur « rouge »
BLUE	Output		B6	Contrôle des pixels de couleur « bleu »
GREEN	Output		D8	Contrôle des pixels de couleur « vert »

*Tableau 2 : Description des signaux de sortie de **chronoscore***

1.3 Décomposition de **chronoscore** en sous-blocs

Le système global **chronoscore** a été décomposé en cinq sous-blocs représentés figure 2.

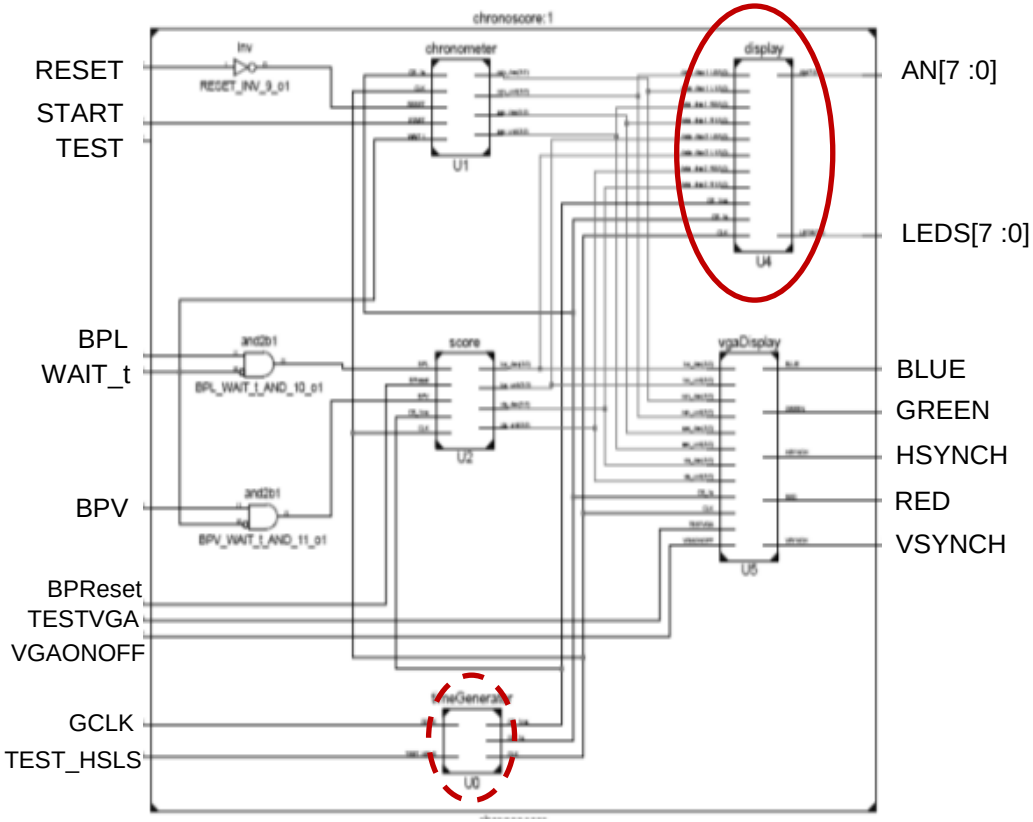


*Figure 2 : Architecture interne de l'entité **chronoscore***

La phase 1 du projet est restreinte à la mise en œuvre des sous-blocs **timeGenerator** et **display** mis en évidence sur la figure ci-dessus.

1.4 Sous-blocs **timeGenerator**

Le synoptique détaillé de **timeGenerator** est rappelé figure 3 (pour la description des signaux CE_1ms et CE_1s, se reporter au document « Présentation générale du projet »).

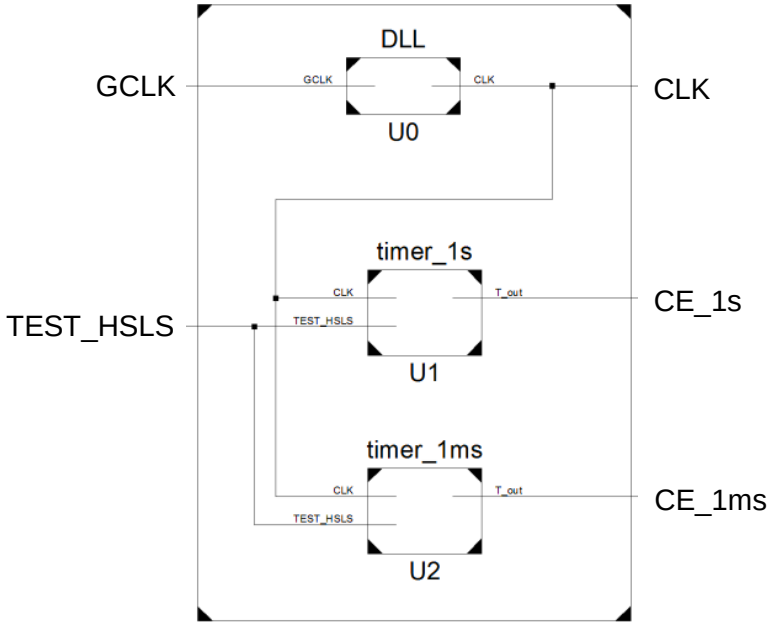


*Figure 3 : Architecture du sous-bloc **timeGenerator***

Pour simplifier la mise en œuvre du projet, une version exécutable de **timeGenerator** (**timeGenerator.ngc**) est **fournie** sur **CPe-Campus**.

1.5 Sous-bloc **display**

Le sous-bloc **display** gère les données à afficher sur les 8 afficheurs 7-segments de la carte **NEXYX A7**.

Les quatre afficheurs de droite (disp1 : afficheurs 0 à 3) doivent indiquer le temps écoulé en minutes et secondes.

Les quatre afficheurs de gauche (disp2 : afficheurs 4 à 7) doivent indiquer le score de l'équipe locale et celui de l'équipe des visiteurs.

Le synoptique complet du sous-bloc **display** est donné figure 4 (une version pleine page est donnée en annexe 1).

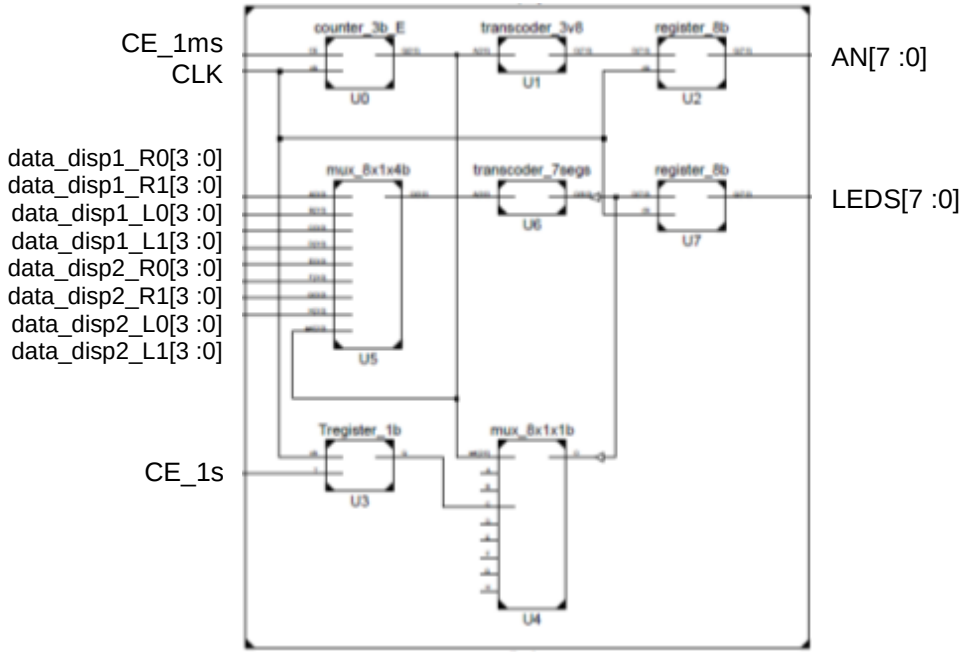


*Figure 4 : Architecture du sous-bloc **display***

2 Phase 1 du projet – Gestion des afficheurs 7-segments

Les entités concernées par la première partie du projet, nommée **chronoscore_phase1,** sont précisées dans l'arborescence figure 5 :

chronoscore_phase1
- ├ **timeGenerator** *(fichier timeGenerator.ngc disponible sur CPe-Campus)*
- └ **display**
  - ├ counter_3b_E
  - ├ mux_8x1x1b
  - ├ mux_8x1x4b
  - ├ register_8b
  - ├ Tregister_1b
  - ├ transcoder_3v8
  - └ transcoder_7segs

*Figure 5 : Entités de **chronoscore_phase1***

Le sous-bloc **display** comporte sept fonctions :

- counter_3b_E
- mux_8x1x4b
- mux_8x1x1b
- register_8b
- Tregister_1b
- transcoder_3v8
- transcoder_7segs

Il y a 4 fonctions de type **combinatoire** et 3 fonctions de type **séquentiel.**

Ces 7 fonctions sont décrites ci-dessous dans l'ordre croissant de complexité.

2.1 Spécifications de la fonction **transcoder_3v8**

La fonction **transcoder_3v8** est en réalité un **décodeur** qui met à l'état **BAS** le numéro du signal de sortie, correspondant à l'équivalent décimal du code binaire sur trois bits appliqué sur ses entrées.

L'entrée de l'entité **VHDL** sera définie comme un vecteur A de dimension 3 : A[2 :0].
La sortie de l'entité **VHDL** sera définie comme un vecteur O de dimension 8 : O[7 :0].

2.2 Spécifications de la fonction **transcoder_7segs**

La fonction **transcoder_7segs** est un **transcodeur** qui associe à un code binaire sur 4 bits, le code 7-segments correspondant à la valeur **hexadécimale** de l'entrée.
La figure 6 ci-dessous précise le nom des segments en fonction de leur répartition sur l'afficheur ainsi que la répartition des segments éclairés pour les chiffres de 1 à 9.

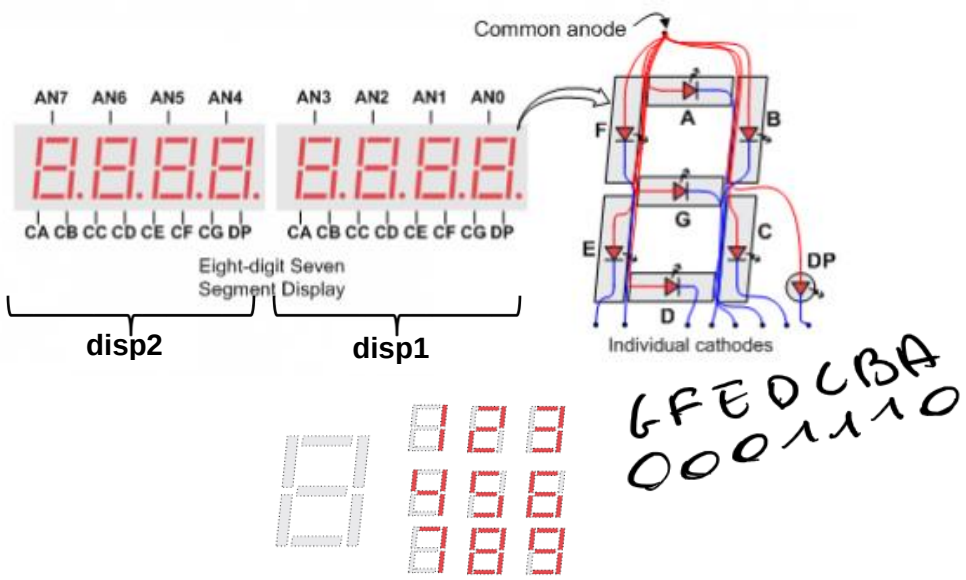


Figure 6 : Répartition des segments d'un afficheur

Les segments **éclairés** seront pilotés par un signal à l'état **BAS**, les segments **éteints** par un signal à l'état **HAUT**.

L'entrée de l'entité **VHDL** sera définie comme un vecteur A de dimension 4 : A[3 :0].

La sortie de l'entité **VHDL** sera définie comme un vecteur O de dimension 7 : O[6 :0].
Le segment « a » correspondra à O[0] et le segment « g » à O[6].

La table de vérité de l'entité **transcoder_7segs** est :

transcoder_7segs		
A[3 :0]	**Caractère hexadécimal à afficher**	**O[6 :0]**
0000	0	0000001
0001	1	1001111
0010	2	0010010
0011	3	0000110
0100	4	1001100
0101	5	0100100
0110	6	0100000
0111	7	0001111
1000	8	0000000
1001	9	0000100
1010	A	0001000
1011	**b**	1100000
1100	C	0110001
1101	**d**	1000010
1110	E	0110000
1111	F	0111000

Attention ! Pour le code « 1011 », le caractère hexadécimal à afficher est « **b** » et pour le code « 1101 » le caractère est « **d** ».

2.3 Spécifications de la fonction **mux_8x1x1b**

La fonction **mux_8x1x1b** est un **multiplexeur** de 8 signaux d'entrée de 1 bit vers 1 signal de sortie de 1 bit. Il dispose de trois signaux de sélection.

Les entrées de l'entité **VHDL** seront définies comme des signaux A, B, C, D, E, F, G, H et comme un vecteur sel de dimension 3 : sel[2 :0].
La sortie de l'entité **VHDL** sera définie comme un signal : O.

La table de vérité de l'entité **mux_8x1x1b** est :

mux_8x1x1b									
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**sel[2 :0]**	**O**
A	-	-	-	-	-	-	-	000	A
-	B	-	-	-	-	-	-	001	B
-	-	C	-	-	-	-	-	010	C
-	-	-	D	-	-	-	-	011	D
-	-	-	-	E	-	-	-	100	E
-	-	-	-	-	F-	-	-	101	F
-	-	-	-	-	-	G	-	110	G
-	-	-	-	-	-	-	H	111	H

Remarque : *Le symbole « - » signifie « quelle que soit la valeur du signal ».*

*Ce symbole n'existant pas en **VHDL**, la description de la fonction sera centrée sur les cas où il y a changement.*

2.4 Spécifications de la fonction **mux_8x1x4b**

La fonction **mux_8x1x4b** est un **multiplexeur** de 8 signaux d'entrée de 4 bits vers 1 signal de sortie de 4 bits. Il dispose de trois signaux de sélection.

Les entrées de l'entité **VHDL** seront définies comme des vecteurs A, B, C, D, E, F, G, H de dimension 4 et comme un vecteur sel de dimension 3 : A[3 :0], B[3 :0], C[3 :0], D[3 :0], D[3 :0], E[3 :0], F[3 :0], G[3 :0], H[3 :0], sel[2 :0].
La sortie de l'entité **VHDL** sera définie comme un vecteur O de dimension 4 : O[3 :0].

La table de vérité de l'entité **mux_8x1x4b** est :

mux_8x1x4b									
A[3 :0]	**B[3 :0]**	**C[3 :0]**	**D[3 :0]**	**E[3 :0]**	**F[3 :0]**	**G[3 :0]**	**H[3 :0]**	**sel[2 :0]**	**O[3 :0]**
A[3 :0]	-	-	-	-	-	-	-	000	A[3 :0]
-	B[3 :0]	-	-	-	-	-	-	001	B[3 :0]
-	-	C[3 :0]	-	-	-	-	-	010	C[3 :0]
-	-	-	D[3 :0]	-	-	-	-	011	D[3 :0]
-	-	-	-	E[3 :0]	-	-	-	100	E[3 :0]
-	-	-	-	-	F[3 :0]	-	-	101	F[3 :0]
-	-	-	-	-	-	G[3 :0]	-	110	G[3 :0]
-	-	-	-	-	-	-	H[3 :0]	111	H[3 :0]

Remarque : *Le symbole « - » signifie « quelle que soit la valeur du signal ».*

*Ce symbole n'existant pas en **VHDL**, la description de la fonction sera centrée sur les cas où il y a changement.*

2.5 Spécifications de la fonction **register_8b**

La fonction **register_8b** est un **registre synchrone** constitué de 8 bascules D **actives sur front montant** du signal d'horloge.

Les entrées de l'entité **VHDL** seront définies comme un signal clk et comme un vecteur D de dimension 8 : clk, D[7 :0].
La sortie de l'entité **VHDL** sera définie comme un vecteur Q de dimension 8 : Q[7 :0].

La table de vérité de l'entité **register_8b** est donnée ci-dessous. Cette table fait apparaître l'état du signal interne Q_int[7 :0] nécessaire pour la description **VHDL** de l'état du registre.

register_8b			
D[7 :0]	**clk**	**Q_int[7 :0]**	**Q[7 :0]**
D[7 :0]	↑	D[7 :0]	Q_int[7 :0]
-	-	Q_int[7 :0]	Q_int[7 :0]

Remarque : *Le symbole « - » signifie « quelle que soit la valeur du signal ».*
Pour le signal d'horloge, ce symbole signifie « quelle que soit la valeur du signal (HAUT, BAS, front descendant) ***autre que*** *la transition front montant ».*
Le symbole « - » n'existant pas en ***VHDL****, la description de la fonction sera centrée sur les cas où il y a changement.*

2.7 Spécifications de la fonction **Tregister_1b**

La fonction **Tregister_1b** est une **bascule T synchrone active sur front montant** du signal d'horloge. L'état de sortie de la bascule est inversé lorsque l'entrée T est à l'état HAUT.

Les entrées de l'entité **VHDL** seront définies comme un signal clk et comme un signal T.
La sortie de l'entité **VHDL** sera définie comme un signal Q.

La table de vérité de l'entité **Tregister_1b** est donnée ci-après. Cette table fait apparaître l'état du signal interne Q_int nécessaire pour la description **VHDL** de l'état du registre.

Tregister_1b			
T	**clk**	**Q_int**	**Q**
0	↑	Q_int	Q_int
1	↑	**NOT** Q_int	Q_int
-	-	Q_int	Q_int

Remarque : *Le symbole « - » signifie « quelle que soit la valeur du signal ».*
Pour le signal d'horloge, ce symbole signifie « quelle que soit la valeur du signal (HAUT, BAS, front descendant) ***autre que*** *la transition front montant ».*
Le symbole « - » n'existant pas vraiment en ***VHDL****, la description de la fonction sera plutôt centrée sur les cas où il y a changement.*

2.8 Spécifications de la fonction **counter_3b_E**

La fonction **counter_3b_E** est un **compteur 3 bits actif sur front montant** du signal d'horloge. Il dispose d'une entrée de validation CE (Clock Enable) **synchrone** et **active à l'état HAUT**.

Les entrées de l'entité **VHDL** seront définies comme un signal clk et comme un signal CE.
Les sorties de l'entité **VHDL** seront définies comme un vecteur Q de dimension 3 : Q[2 :0].

La table de vérité de l'entité **counter_3b_E** est donnée ci-dessous. Cette table fait apparaître l'état du signal interne Q_int[2 :0] nécessaire pour la description **VHDL** de l'état du compteur.

counter_3b_E				
CE	**clk**	**Q_int_n[2 :0]**	**Q_int_{n+1}[2 :0]**	**Q[2 :0]**
1	↑	000	001	Q_int[2 :0]
1	↑	001	010	Q_int[2 :0]
1	↑	010	011	Q_int[2 :0]
1	↑	011	100	Q_int[2 :0]
1	↑	100	101	Q_int[2 :0]
1	↑	101	110	Q_int[2 :0]
1	↑	110	111	Q_int[2 :0]
1	↑	111	000	Q_int[2 :0]
0	-	-	Q_int_n[2 :0]	Q_int[2 :0]]

Remarque : *Le symbole « - » signifie « quelle que soit la valeur du signal ».*
Pour le signal d'horloge, ce symbole signifie « quelle que soit la valeur du signal (HAUT, BAS, front descendant) ***autre que*** *la transition front montant ».*
Le symbole « - » n'existant pas vraiment en ***VHDL****, la description de la fonction sera plutôt centrée sur les cas où il y a changement.*

Un compteur peut être défini de **manière simple** comme un **additionneur**.
Rappel : Pour pouvoir faire des additions, il faut ajouter à la description de l'entité, la bibliothèque **IEEE.NUMERIC_STD.ALL.**

3 Préparation

Etudier le document intitulé « Présentation du projet ».

Apporter une attention toute particulière à la partie 3.3 : « Principe de fonctionnement des afficheurs 7-segments ».

3.1 Préparation pour les **fonctions de type combinatoire** du sous-bloc **display**

1. Ecrire la table de vérité de la fonction **transcoder_3v8**.
   Ecrire les expressions algébriques sous la 2ème forme canonique (produit de sommes) des sorties de la fonction.
2. Compléter la table de vérité de la fonction **transcoder_7segs** (selon modèle figure 6 pour les chiffres).
   Pour chaque ligne convertir en hexadécimal le code sur 7 bits obtenu en considérant que le bit de poids fort manquant est égal à « 0 ».
3. Ecrire l'expression algébrique sous la 1ère forme canonique (somme de produits) de la sortie de **mux_8x1x1b**.
4. Ecrire, à partir de syntaxes **d'équations concurrentes**, le code **VHDL** de l'**architecture** de chaque fonction à réaliser.
5. Définir le chronogramme des vecteurs de test de chacune de ces fonctions.

3.2 Préparation pour les **fonctions de type séquentiel** du sous-bloc **display**

1. Ecrire, à partir de syntaxes **de « process »**, le code **VHDL** de l'**architecture** de chaque fonction à réaliser.
2. Définir le chronogramme des vecteurs de test de chacune de ces fonctions.
3. Exercice :
   Un compteur 3 bits synchrone est réalisé à partir de bascules D. Sachant que les entrées D des bascules sont le futur des sorties Q, on peut écrire la table de vérité suivante :

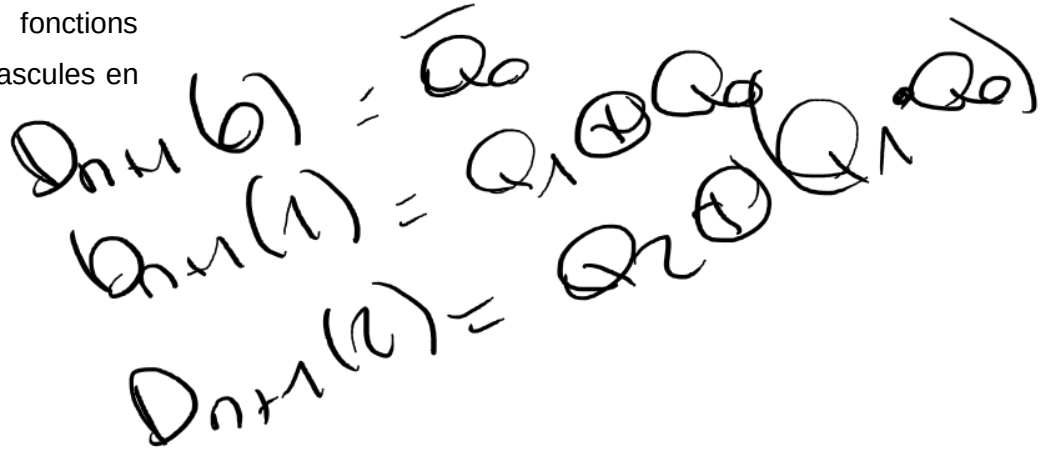
counter_3b		
clk	**Q_n[2 :0]**	**D_{n+1}[2 :0]**
↑	000	001
↑	001	010
↑	010	011
↑	011	100
↑	100	101
↑	101	110
↑	110	111
↑	111	000

A partir de cette table, écrire les équations des fonctions combinatoires à appliquer sur les entrées D_{n+1} des bascules en fonction des sorties Q_n :

$D_{n+1}(0) = f(Q_n(0), Q_n(1), Q_n(2))$

$D_{n+1}(1) = f(Q_n(0), Q_n(1), Q_n(2))$

$D_{n+1}(2) = f(Q_n(0), Q_n(1), Q_n(2))$



4. Réalisation du projet chronoscore_phase1

Le projet **chronoscore_phase1** met en œuvre une version de **chronoscore** restreinte aux sous-blocs **timeGenerator** et **display**.

Le développement du sous-bloc **display** s'effectue en équipe de 4 ou 5 selon l'organisation ci-dessous :

Binôme (trinôme) impair : entité **transcoder_3v8**
entité **mux_8x1x4b**
entité **register_8b**
entité **Tregister_1b**
entité **counter_3b_E**

Binôme pair : entité **transcoder_7segs**
entité **mux_8x1x1b**
entité **register_8b**
entité **Tregister_1b**
entité **counter_3b_E**

Equipe : intégration du sous-bloc **display**
intégration de **chronoscore_phase1**

Chaque binôme (trinôme) doit réaliser **deux fonctions combinatoires** et les **trois fonctions séquentielles**.

4.1 Création du projet

Démarrer **ISE version 14.6** (64_bit) via le menu **Démarrer** de Windows et créer le projet **chronoscore_phase1** ayant les caractéristiques suivantes :

Name	**chronoscore_phase1**
Location	**(…)\GR_X\ELN2\SCORING\EQUIPE_N**
Working Directory	**(…)\GR_X\ELN1\SCORING\EQUIPE_N**
Top_Level source type	**HDL**

(…) : **CPE_USERS\TPELEC_3ETI** (Répertoire sous Windows).

GR_X : X vaut de A à D en fonction du groupe de TP
EQUIPE_N : N vaut de 1 à 8 en fonction du numéro d'équipe

Les différents répertoires devront être créés au préalable.

Rappel ! Le système crée automatiquement le répertoire associé au nom du projet. Il travaille uniquement dans ce répertoire. **Il est important que tous les fichiers sources y soient placés.**

Les caractéristiques du **FPGA** cible sont :

Evaluation Development Board	None Specified
Product Category	All
Family	**Artix7**
Device	**XC7A100T**
Package	**CSG324**
Speed	**-1**
Top-Level Source Type	**HDL**
Synthesis Tool	**XST (VHDL/Verilog)**
Simulator	**ISim (VHDL/Verilog)**
Preferred Language	**VHDL**
Property Specification in Project File	Store non-default values only
Manual Compile Orders	☐
VHDL Source Analysis Standard	**VHDL-200X**
Enable Massage Filtering	☐

4.2 Réalisation des fonctions du sous-bloc **display**

Pour chacune des-**fonctions** du sous-bloc **display :**

- Créer l'entité* **VHDL** correspondante (Projet → New Source).
- Définir ses signaux d'entrées* et sorties*.
  (*) : Respecter les noms imposés dans les spécifications.
- Ecrire le code **VHDL** de son architecture.
  Préciser dans la zone d'entête de chaque fichier source les noms des auteurs (par ordre alphabétique) :
  -- Engineer: NOM1_NOM2 (ajouter **NOM3** si trinôme).
- Faire la synthèse.

Pour l'entité **transcoder_3v8** (binôme **impair**) :

A partir de la vue **Technologique**, expliquer comment sont réalisées les équations obtenues en préparation.

Effectuer la simulation **Behavioral** et la simulation **Post-Route**.

Exploiter les résultats.

Pour l'entité **mux_8x1x4b** (binôme **impair**) :

Effectuer la simulation **Behavioral**.

Exploiter le résultat.

Pour l'entité **transcoder_7segs** (binôme **pair**) :

Effectuer la simulation **Behavioral**.

Exploiter le résultat.

Pour l'entité **mux_8x1x1b** (binôme **pair**) :
A partir de la vue **Technologique**, expliquer comment sont réalisées les équations obtenues en préparation.
Effectuer la simulation **Behavioral** et la simulation **Post-Route**.
Exploiter les résultats.

Pour l'entité **register_8b** (binômes **impair** et **pair**) :
Effectuer la simulation **Behavioral** et la simulation **Post-Route**.
Exploiter les résultats des simulations.
Expliquer notamment pourquoi sur la simulation **Post-Route,** il faut attendre **100 ns** avant d'obtenir de véritables résultats.

Pour l'entité **Tregister_1b** (binômes **impair** et **pair**) :
A partir des résultats obtenus pour la vue **Technologique**, expliquer comment réaliser simplement une bascule T.
Effectuer la simulation **Behavioral**.
Exploiter les résultats des simulations.
Expliquer notamment le rôle de l'entrée T.

Pour l'entité **counter_3b_E** (binômes **impair** et **pair**) :
Comparer les résultats obtenus pour la vue **Technologique**, avec les équations obtenues dans l'exercice de préparation.
Effectuer la simulation **Behavioral**.
Exploiter les résultats des simulations.
Expliquer notamment le rôle de l'entrée CE.

Pour ces simulations, les signaux **clk**, **CE_1ms** et **CE_1s** seront définis de la manière suivante :

- **CLK** : horloge de période 10 ns, durée à l'état HAUT 5 ns,
- **CE_1ms** : horloge de période 100 ns, durée à l'état HAUT 10 ns,
- **CE_1s** : horloge de période 100 ns, durée à l'état HAUT 10 ns.

La description **VHDL** de ces signaux est fournie en annexe 2.

Pour chaque entité **jugée valide** (résultats corrects après synthèse et simulations) :
- **imprimer et valoriser** (c'est-à-dire mettre en valeur les éléments intéressants) les documents suivants :
  le code source (fichier .vhd),
  la vue **Technologique**,
  les chronogrammes des simulations **Behavioral** et/ou **Post-Route**.
- **compléter la ligne correspondante du tableau de caractérisation** du sous-bloc **display** (fourni(1) sur CPe-Campus) en précisant les informations suivantes :
  le nombre de SLICES,
  le nombre de SLICES LUTS,
  le nombre de SLICES REGISTERS ou IOB FLIP-FLOP,
  le nombre d'IOBS (entrées / sorties).

(1) Préciser l'équipe et les noms des étudiants.

Pour chacune des fonctions **transcoder_3v8**, **mux_8x1x1b**, **register_8b** **rassembler** dans le tableau les temps de propagation(2) les informations(3) suivantes :

le temps du chemin le plus long (à rechercher sur la durée totale de la simulation),

le temps dans le buffer d'entrée,

le temps dans le buffer de sortie.

N.B : Le temps dans la fonction (logique + fils) sera calculé automatiquement.

(2) Les temps de propagation à rassembler dans le tableau doivent apparaitre **clairement** sur la simulation **Post-Route**.

(3) Les informations à rassembler dans le tableau sont disponibles dans le document **Place and Route Report** une fois la phase d'implémentation réussie. Si la page **HTML** ne se met pas à jour, on peut les trouver dans le fichier d'extension « **.par »** créé par l'outil ISE. Ce fichier est disponible dans le répertoire du projet.

4.3 Réalisation du sous-bloc **display**

1. Créer l'entité* VHDL correspondante (Projet → New Source).
   Définir ses signaux d'entrées* et sorties*.
   (*) : Respecter les noms imposés dans les spécifications.
2. Ecrire le code VHDL de son architecture à partir de ses fonctions de base.
   Préciser dans la zone d'entête du fichier source les noms des auteurs (par ordre alphabétique) :
   -- Engineer: **NOM1_NOM2** (ajouter **NOM3** si trinôme).
   *L'entité **display** comporte des signaux internes suivants (voir le synoptique fourni en annexe 1) :*
   ***AN_sel**, **AN_T3v8**, **T1s**, **DP**, **segs_data**, **sseg_7**, **sseg**.*
   *Le signal **sseg** est obtenu par concaténation de **DP** et **sseg_7**.*
   *Les entrées non utilisées de l'entité **mux_8x1x1b** (**A**, **B**, **D, E, F, G, H**) sont positionnées à **'1'**.*
3. Faire la synthèse.
   Vérifier que le synoptique obtenu est conforme à celui attendu.
4. Effectuer plusieurs simulations **Behavioral** permettant de mettre en évidence le fonctionnement du sous-bloc :
   - la gestion des anodes : visualisation sur une durée de **1ms** des signaux d'entrée **CLK** et **CE_1ms**,
     du signal de sortie **AN[7:0],**
     des signaux de sortie des entités **counter_3b_E**, **transcoder_3v8**),

- la gestion des cathodes : visualisation sur une durée de **1ms**

  des signaux d'entrée **CLK**, **CE_1ms**, **data_disp1_R0**, **data_disp1_R1**, **data_disp1_L0**, **data_disp1_L1**, **data_disp2_R0**, **data_disp2_R1**, **data_disp2_L0**, **data_disp2_L1**,

  du signal de sortie **LEDS[7:0],**

  des signaux de sortie des entités **mux_8x1x4b** et **transcoder_7segs**,
- la gestion du clignotement du point : visualisation sur une durée de **2.5 ms**

  des signaux d'entrée **CLK**, **CE_1ms**, **CE_1s**, **data_disp1_R0**, **data_disp1_R1**, **data_disp1_L0**, **data_disp1_L1**, **data_disp2_R0**, **data_disp2_R1**, **data_disp2_L0**, **data_disp2_L1**,

  du signal de sortie **LEDS[7:0]**,

  des signaux de sortie des entités **counter_3b_E**, **Tregister_1b**, **mux_8x1x1b**).

Pour ces simulations, les signaux **CLK**, **CE_1ms** et **CE_1s** seront définis de la manière suivante :

- **CLK** : horloge de période 10 ns, durée à l'état HAUT 5 ns,
- **CE_1ms** : horloge de période 100 ns, durée à l'état HAUT 10 ns,
- **CE_1s** : horloge de période **800 ns**, durée à l'état HAUT 10 ns.

La description **VHDL** de ces signaux est fournie en annexe 2.

Les signaux d'entrée **data_disp1_R0** à **data_disp2_L1** seront positionnés de manière à pouvoir être clairement identifiés.

Une fois l'entité **jugée valide** (résultats corrects après synthèse et simulations) :

- **imprimer et valoriser** les documents suivants :

  le code source (fichier .vhd),

  la vue **RTL** développée,

  les chronogrammes des simulations **Behavorial**.
- **compléter la ligne correspondante du tableau de caractérisation** du sous-bloc **display** (fourni(1) sur CPe-Campus) en précisant les informations suivantes :

  le nombre de SLICES,

  le nombre de SLICES LUTS,

  le nombre de SLICES REGISTERS ou IOB FLIP-FLOP

  le nombre d'IOBS (entrées / sorties).

(1) Préciser l'équipe et les noms des étudiants.

4.4 Insertion du sous-bloc **timeGenerator**

1. A partir du cours ELN2 sur **CPe-Campus**, télécharger le fichier **timeGenerator.ngc**
2. Placer ce fichier dans le répertoire du projet.
   Ajouter le fichier au projet **chronoscore_phase1** (Projet → Add Source).

4.5 Finalisation du système **chronoscore_phase1**

1. A partir du cours ELN2 sur **CPe-Campus**, télécharger les fichiers :
   - **chronoscore_phase1.vhd**,
   - **chronoscore_phase1.ucf**.
2. Placer ces fichiers dans le répertoire du projet.
   Ajouter les fichiers au projet **chronoscore_phase1** (Projet → Add Source).
   Rappel : Pour **chronoscore_phase1.ucf** penser à mettre **chronoscore_phase1.vhd** en « **top-module** » avant.
3. Par défaut, **chronoscore_phase1** affiche l'année universitaire en cours.
   Modifier le fichier **chronoscore_phase1.vhd** de manière à pouvoir afficher la date sous la forme :
   - Sur disp2 : Année sur 4 digits
   - Sur sisp1 : 0X (X Groupe de TP) puis EN (N Numéro d'équipe)
4. Exécuter l'ensemble du flot de conception pour générer le fichier binaire qui servira à configurer les **LUTs**, bascules et interconnexions du **FPGA**.
5. Implanter le système **chronoscore_phase1** dans le **FPGA** à l'aide de l'outil **iMPACT**.
6. Tester la réalisation à l'aide de la carte de développement et **faire valider** par un assistant.

5 En fin de séance

Dépôt de fichiers :

- Faire un répertoire **« .zip »** des fichiers sources créés et du tableau Excel complété.
  *Les répertoires au format **tar** ou **7z** ne seront **pas acceptés**.*
  Le nom du répertoire devra être formaté de la manière suivante ;
  ELN2-X-NOM1-NOM2-… :
  X, groupe de TP
  NOM1, nom étudiant(e) 1
  NO2, nom étudiant(e) 2
  etc …

  } Mettre les noms des membres de l'équipe par **ordre alphabétique**

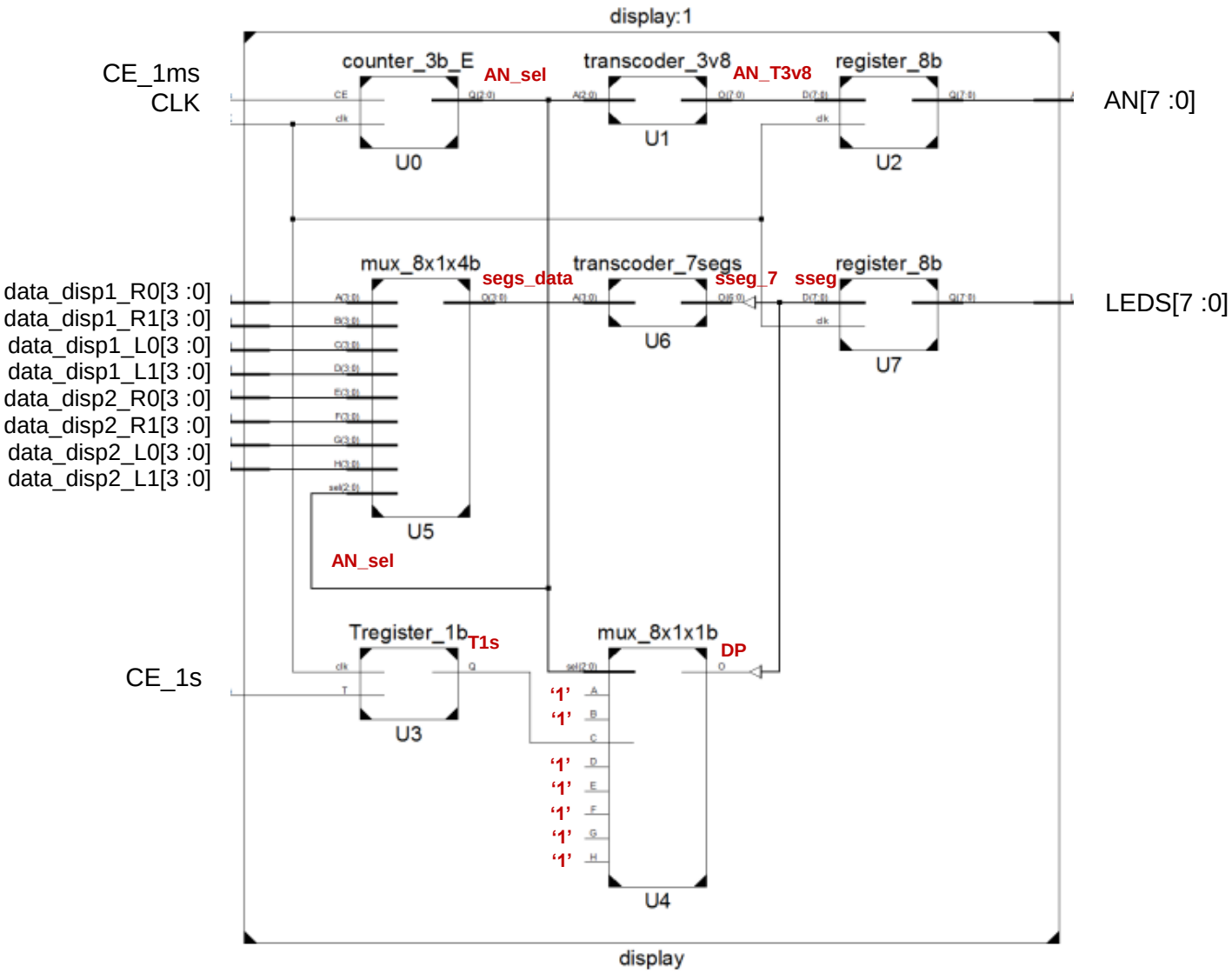
- Le dépôt du répertoire devra être effectué sur CPe-Campus par l'étudiant(e) n°1 dans l'ordre alphabétique.

Documents papier à rendre :

- Pour chaque entité créée, rendre :
  Impression du code source,
  Impression valorisée de la vue **RTL** ou **Technologique** selon le cas,
  Impression valorisée de la simulation **Behavorial**,
  Impression valorisée de la simulation **Post-Route** selon le cas.
- Tableau de caractérisation du sous-bloc **display**.

Annexe 1

Architecture du sous-bloc **display**



L'entité **display** comporte des signaux internes suivants :

AN_sel, **AN_T3v8**, **T1s**, **DP**, **segs_data**, **sseg_7**, **sseg**.

Le signal **sseg** est obtenu par concaténation de **DP** et **sseg_7** :

sseg(7 downto 0) <= DP & sseg_7(6 downto 0);

Les entrées non utilisées de l'entité mux_8x1x1b (**A**, **B**, **D, E, F, G, H**) sont positionnées à **'1'**.

Annexe 2

Code **VHDL** des signaux **CLK**, **CE_1ms** et **CE_1s** à utiliser pour les simulations

-- Clock period definitions *(à placer dans la zone de déclarations du fichier _tb.vhd)*

```
constant clk_period : time := 10 ns;
constant CLK_period : time := 10 ns;
constant CE_1ms_period : time := 100 ns;
constant CE_1s_period : time := 100 ns;     -- ou constant CE_1s_period : time := 800 ns;
```

-- Clock process definitions *(à placer entre BEGIN et END)*

```
clk_process :process                        – ou CLK_process : process
begin
    clk <= '0';                             -- ou CLK <= ‘0’;
    wait for clk_period/2;
    clk<= '1';                              -- ou CLK <= ‘1’;
    wait for clk_period/2;
end process;

CE_1ms_process :process
begin
    CE_1ms <= '0';
    wait for 90 ns;
    CE_1ms <= '1';
    wait for 10 ns;
end process;

CE_1s_process :process
begin
    CE_1s <= '0';
    wait for 90 ns;                         -- ou wait for 790 ns;
    CE_1s <= '1';
    wait for 10 ns;
end process;
```