ELN2 : PROJET SCORING 2.0

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

Phase 2 : Gestion du chronomètre et du score

(Préparation)

La **phase 2** du projet **Scoring 2.0** est consacrée à la gestion du chronomètre et du score. Elle met en œuvre les sous-blocs **chronometer** et **score.** Ces sous-blocs ayant des composants très similaires, il est possible de les concevoir conjointement.

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.

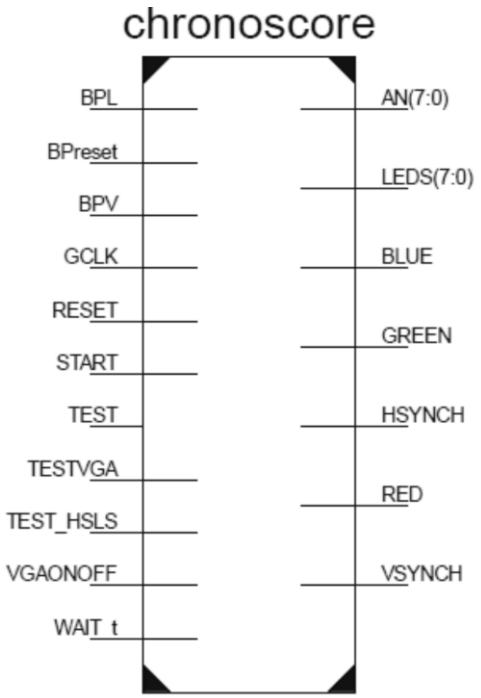


*Figure 1 : Vue TOP de l'entité **chronoscore***

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.

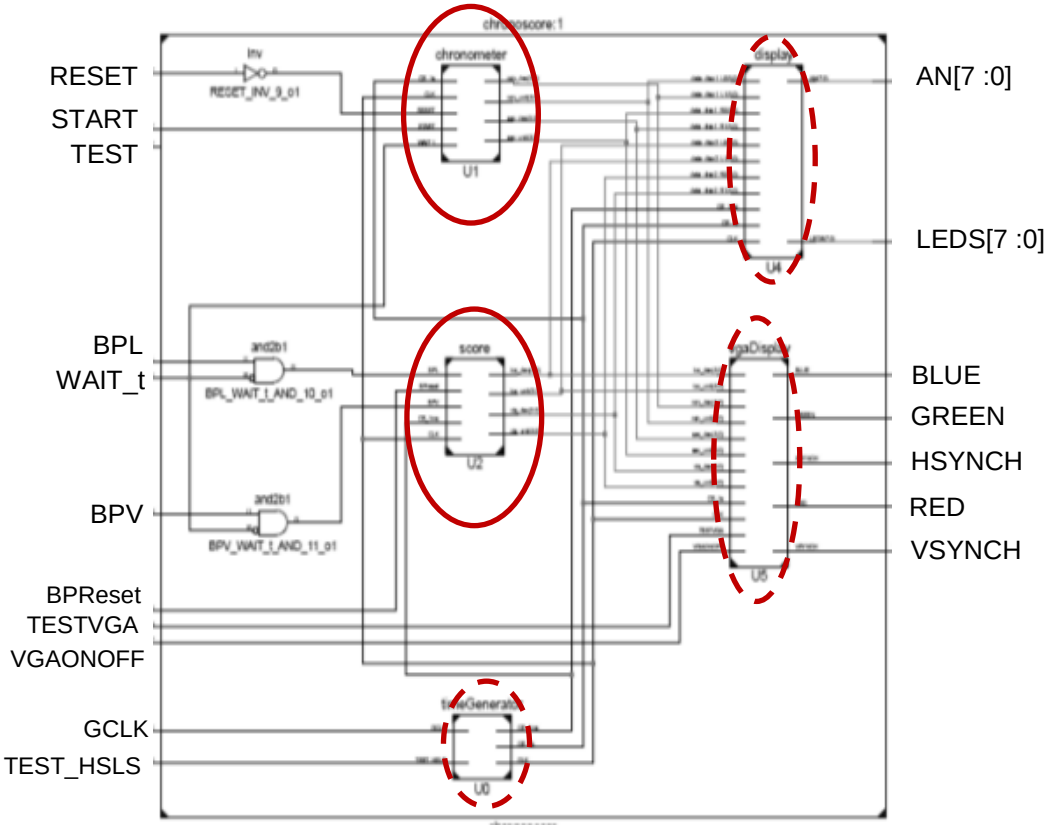


*Figure 2 : Architecture interne de l'entité **chronoscore***

La phase 2 du projet met en œuvre les sous-blocs **timeGenerator** (déjà utilisé), **display** (déjà étudié), **chronometer** et **score.**

1.4 Sous-blocs **timeGenerator** (rappel)

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 »).

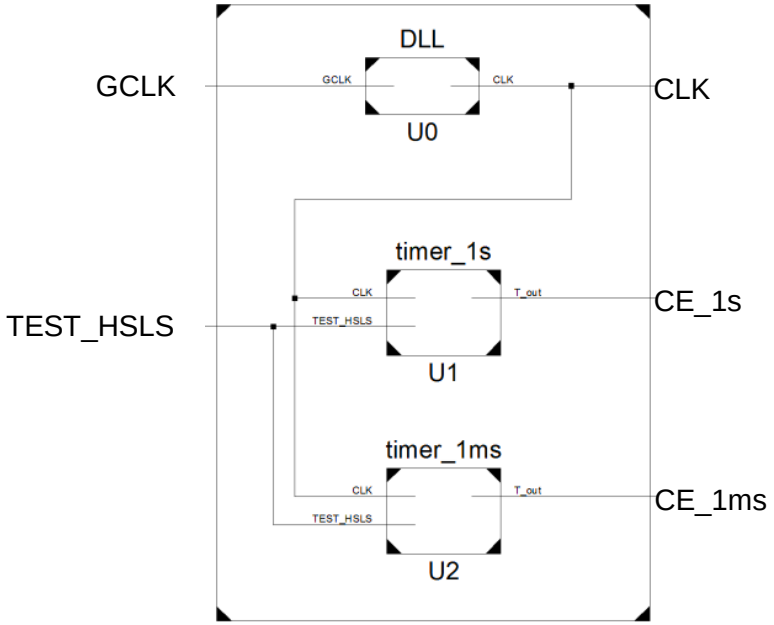


*Figure 3 : Architecture du sous-bloc **timeGenerator***

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** (rappel)

Le synoptique détaillé de **display** est rappelé figure 4 (une version pleine page est donnée en annexe 1)**.**

Les quatre afficheurs de droite de la carte NEXYS A7 (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.

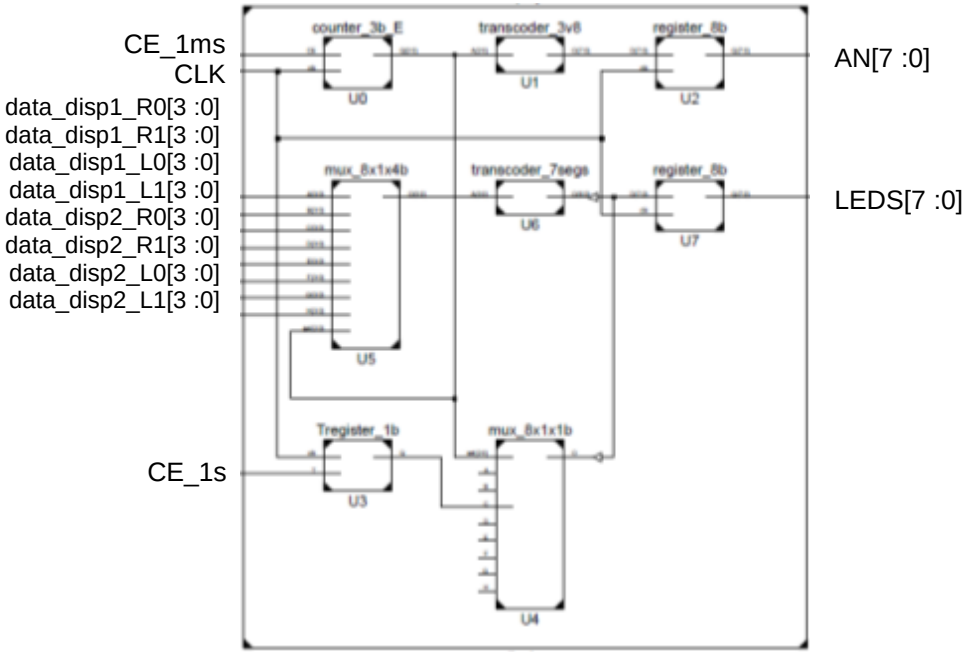


*Figure 4 : Architecture du sous-bloc **display***

Le sous-bloc **display** est issu de la phase 1. Si son développement n'a pas été terminé, il est possible de tester la phase 2 en utilisant une version exécutable de l'entité **display** (display.ngc) fournie sur **CPe-Campus**.

1.7 Sous-bloc **chronometer**

Le sous-bloc **chronometer** fournit aux sous-blocs **display** et **vgaDisplay** les données temporelles (minutes et secondes) à afficher.

Le synoptique détaillé de **chronometer** est donné figure 5 (une version pleine page est donnée en annexe 2).

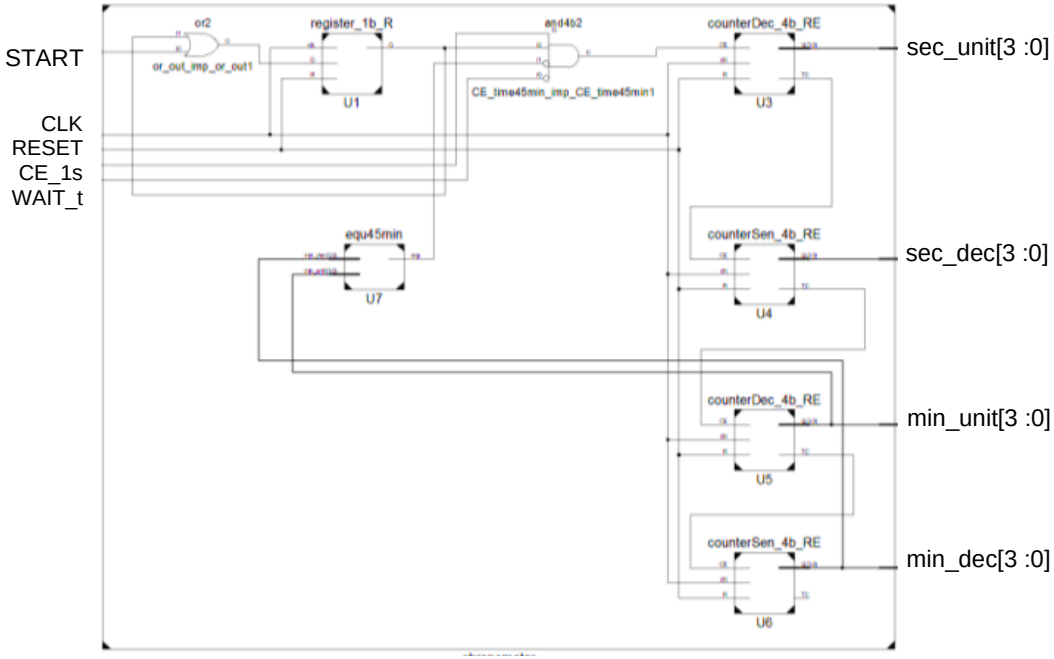


*Figure 5 : Architecture de l'entité **chronometer***

1.8 Sous-bloc **score**

Le sous-bloc **score** fournit aux sous-blocs **multiplexData** et **vgaDisplay** les scores des équipes (unités et dizaines) à afficher.

Le synoptique détaillé de **score** est donné figure 6 (une version pleine page est donnée en annexe 3).

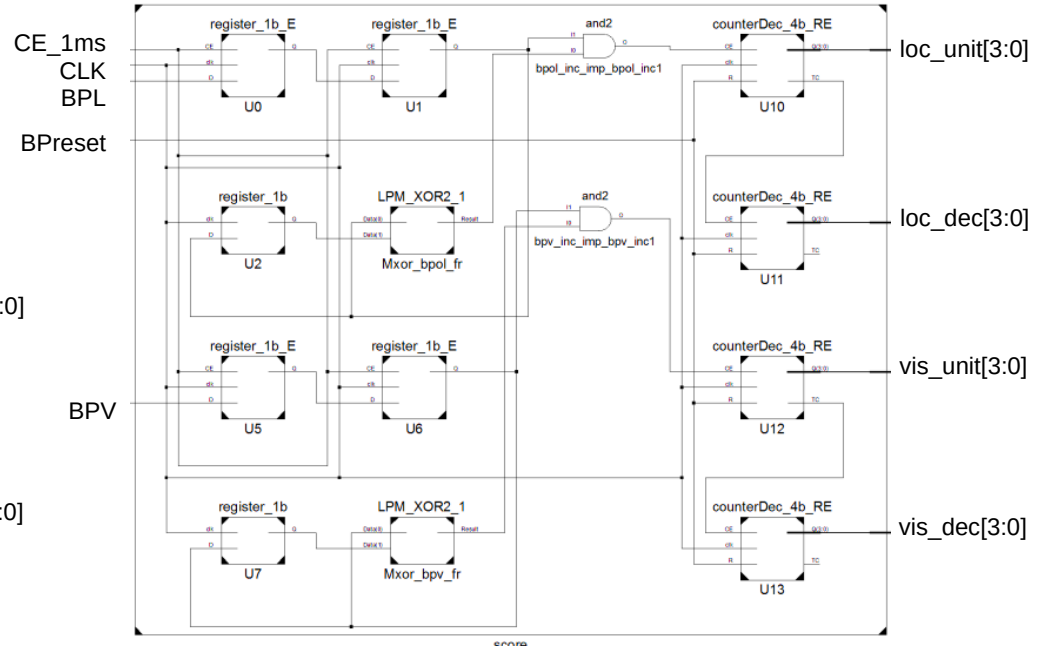


*Figure 6 : Architecture de l'entité **score***

2 Phase 2 du projet – Gestion du chronomètre et du score

Les entités concernées par la seconde partie du projet, nommée **chronoscore_phase2**, sont indiquées dans l'arbre figure 7 :

chronoscore
- **timeGenerator** *(fichier timeGenerator.ngc disponible sur CPe-Campus)*
- **chronometer**
  - counterDec_4b_RE
  - counterSen_4b_RE
  - equ45min
  - register_1b_R
  - AND4n2 *(à décrire directement dans l'entité chronometer.vhd)*
  - OR *(à décrire directement dans l'entité chronometer.vhd)*
- **score**
  - counterDec_4b_RE
  - register_1b
  - register_1b_E
  - AND *(à décrire directement dans l'entité chronometer.vhd)*
  - XOR2
- **display** *(fichier display.ngc disponible sur CPe-Campus)*
- **vgaDisplay** *(fichier vgaDisplay.ngc disponible sur CPe-Campus)*

*Figure 7 : Entités de **chronoscore_phase2***

Le sous-bloc **chronometer** comporte six fonctions listées dans un ordre croissant de complexité :

- OR *(à décrire directement dans l'entité chronometer.vhd)*
- AND4n2 *(à décrire directement dans l'entité chronometer.vhd)*
- equ45min
- register_1b_R
- counterSen_4b_RE
- counterDec_4b_RE

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

Le sous-bloc **score** comporte 5 fonctions listées dans un ordre croissant de complexité :

- AND *(à décrire directement dans l'entité chronometer.vhd)*
- XOR_2b
- register_1b
- register_1b_E
- counterDec_4b_RE

Il y a 2 fonctions de type combinatoire et 3 fonctions de type séquentiel dont une est également présente dans **chronometer (**counterDec_4b_RE).

2.1 Spécifications des **fonctions de type combinatoire**

Les fonctions sont décrites par ordre de complexité croissante.

2.1.1 Spécifications de la fonction **XOR_2b (score)**

La fonction **XOR_2b** est la fonction combinatoire OU-EXCLUSIF à deux entrées connue.

Les entrées de l'entité **VHDL** seront définies comme des signaux : A, B.
La sortie de l'entité **VHDL** sera définie comme un signal : O.

2.1.2 Spécifications de la fonction **equ45min (chronometer)**

La fonction **equ45min** est un comparateur logique. Lorsque l'entrée des unités vaut « 5 », et que celle des dizaines vaut « 4 », la sortie vaut « 1 ». Sinon, elle vaut « 0 ».

Les entrées de l'entité **VHDL** seront définies comme des vecteurs de dimension 4 : min_dec[3 :0], min_unit[3 :0].
La sortie de l'entité **VHDL** sera définie comme un signal : equ.

2.2 Spécifications des **fonctions de type séquentiel**

2.2.1 Spécifications de la fonction **register_1b (score)**

La fonction **register_1b** est un registre **synchrone** constitué d'une bascule D **active sur front montant** du signal d'horloge **clk**.

Les entrées de l'entité **VHDL** seront définies comme des signaux : D, clk.
La sortie de l'entité **VHDL** sera définie comme un signal : Q.
Un signal interne Q_int sera nécessaire pour décrire l'état du registre.

2.2.2 Spécifications de la fonction **register_1b_R (chronometer)**

La fonction **register_1b_R** est un registre **synchrone** constitué d'une bascule D **active sur front montant** du signal d'horloge **clk**. Il dispose d'une entrée de remise à zéro **R** (Reset) **synchrone active à l'état HAUT**.

Les entrées de l'entité **VHDL** seront définies comme des signaux : R, D, clk.
La sortie de l'entité **VHDL** sera définie comme un signal : Q.
Un signal interne Q_int sera nécessaire pour décrire l'état du registre.

2.2.3 Spécifications de la fonction **register_1b_E (score)**

La fonction **register_1b_E** est un registre **synchrone** constitué d'une bascule D **active sur front montant** du signal d'horloge **clk**. Il dispose d'une entrée de validation **CE** (Clock Enable) **synchrone active à l'état HAUT**.

Les entrées de l'entité **VHDL** seront définies comme des signaux : CE, D, clk.
La sortie de l'entité **VHDL** sera définie comme un signal : Q.
Un signal interne Q_int sera nécessaire pour décrire l'état du registre.

2.2.5 Spécifications de la fonction **counterSen_4b_RE (chronometer)**

La fonction **counterSen_4b_RE** est un compteur **4 bits** qui compte de **0 à 5** et qui est **actif sur front montant** du signal d'horloge **clk**.
Il dispose d'une entrée de remise à zéro **R** (Reset) **asynchrone active à l'état HAUT** et d'une entrée de validation **CE** (Clock Enable) **synchrone active à l'état HAUT**.

Il dispose également d'un signal de sortie **TC** (Terminal Count) qui passe à **l'état HAUT** pendant une période d'horloge en fin de comptage.

Les entrées de l'entité **VHDL** seront définies comme des signaux : R, CE, clk.
Les sorties de l'entité **VHDL** seront définies comme un vecteur de dimension 4 et un signal : Q[3 :0], TC.
Un signal interne Q_int[3 :0] sera nécessaire pour connaître l'état du compteur et agir sur le signal de sortie TC via le signal interne TC_int.

La table de vérité de l'entité **counterSen_4b_RE** est donnée ci-dessous. Cette table fait apparaître les états des signaux internes Q_int[3 :0] et TC_int.

counterSen_4b_RE							
R	**CE**	**clk**	**Q_int_n[3 :0]**	**Q_int_{n+1}[3 :0]**	**Q[3 :0]**	**TC_int**	**TC**
1	-	-	-	0000	Q_int[3 :0]	0	TC_int
0	1	↑	0000	0001	Q_int[3 :0]	0	TC_int
0	1	↑	0001	0010	Q_int[3 :0]	0	TC_int
0	1	↑	0010	0011	Q_int[3 :0]	0	TC_int
0	1	↑	0011	0100	Q_int[3 :0]	0	TC_int
0	1	↑	0100	0101	Q_int[3 :0]	0	TC_int
0	1	↑	0101	0000	Q_int[3 :0]	1	TC_int
0	1	-	-	Q_int_n[3 :0]	Q_int[3 :0]	TC_int	TC_int
0	0	↑	-	Q_int_n[3 :0]	Q_int[3 :0]	0	TC_int
0	0	-	-	Q_int_n[3 :0	Q_int[3 :0]	0	TC_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.*
Rappel : Un compteur peut être défini de manière simple comme un **additionneur**. Pour pouvoir faire des additions, il faut ajouter à la l'entité la bibliothèque **IEEE.NUMERIC_STD.ALL.**

La figure 8 donne le chronogramme des signaux de l'entité **counterSen_4b_RE**. Remarquer notamment que le signal **TC** est à

l'état **HAUT** pendant une période d'horloge **clk** quand la condition sur Q_int[3 :0] est remplie **et que** le signal **CE** est à l'état **HAUT**. Sinon, il est à l'état **BAS**.

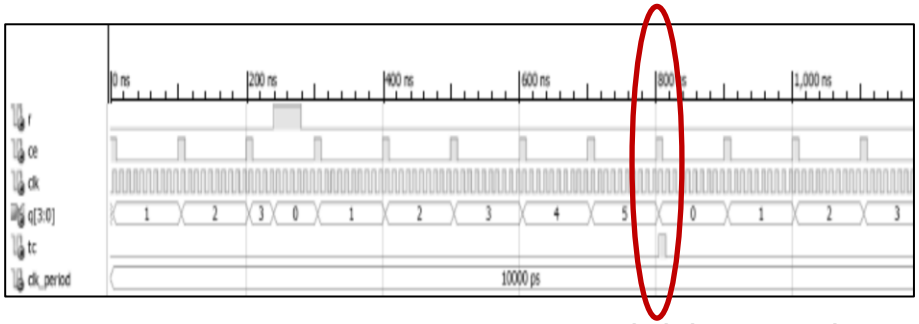


*Figure 8 : Chronogramme du signal TC généré par l'entité **counterSen_4b_RE**.*

2.2.6 Spécifications de la fonction **counterDec_4b_RE (chronometer et score)**

La fonction **counterDec_4b_RE** est un compteur **4 bits** qui compte de **0 à 9** et qui est **actif sur front montant** du signal d'horloge **clk**.

Il dispose d'une entrée de remise à zéro **R** (Reset) **asynchrone active à l'état HAUT** et d'une entrée de validation **CE** (Clock Enable) **synchrone active à l'état HAUT**.

Il dispose également d'un signal de sortie **TC** (Terminal Count) qui passe à **l'état HAUT** pendant une période d'horloge en fin de comptage.

Les entrées de l'entité **VHDL** seront définies comme des signaux : R, CE, clk.

Les sorties de l'entité VHDL seront définies comme un vecteur de dimension 4 et un signal : Q[3 :0], TC.

Un signal interne Q_int[3 :0] sera nécessaire pour connaître l'état du compteur et agir sur le signal de sortie TC via le signal interne TC_int.

La table de vérité de l'entité **counterDec_4b_RE** est donnée ci-après. Cette table fait apparaître les états des signaux internes Q_int[3 :0] et TC_int.

counterDec_4b_RE							
R	**CE**	**clk**	**Q_int_n[3 :0]**	**Q_int_{n+1}[3 :0]**	**Q[3 :0]**	**TC_int**	**TC**
1	-	-	-	0000	Q_int[3 :0]	0	TC_int
0	1	↑	0000	0001	Q_int[3 :0]	0	TC_int
0	1	↑	0001	0010	Q_int[3 :0]	0	TC_int
0	1	↑	0010	0011	Q_int[3 :0]	0	TC_int
0	1	↑	0011	0100	Q_int[3 :0]	0	TC_int
0	1	↑	0100	0101	Q_int[3 :0]	0	TC_int
0	1	↑	0101	0110	Q_int[3 :0]	0	TC_int
0	1	↑	0110	0111	Q_int[3 :0]	0	TC_int
0	1	↑	0111	1000	Q_int[3 :0]	0	TC_int
0	1	↑	1000	1001	Q_int[3 :0]	0	TC_int
0	1	↑	1001	0000	Q_int[3 :0]	1	TC_int
0	1	-	-	Q_int_n[3 :0]	Q_int[3 :0]	TC_int	TC_int
0	0	↑	-	Q_int_n[3 :0]	Q_int[3 :0]	0	TC_int
0	0	-	-	Q_int_n[3 :0]	Q_int[3 :0]	0	TC_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.*
Rappel : Un compteur peut être défini de manière simple comme un **additionneur**. Pour pouvoir faire des additions, il faut ajouter à la description la bibliothèque **IEEE.NUMERIC_STD.ALL.**

La figure 9 donne le chronogramme des signaux de l'entité counterDec_4b_RE. Remarquer notamment que le signal **TC** est à l'état **HAUT** pendant une période d'horloge **clk** quand la condition sur Q_int[3 :0] est remplie et que le signal **CE** est à l'état **HAUT**. Il est à l'état **BAS** sinon.

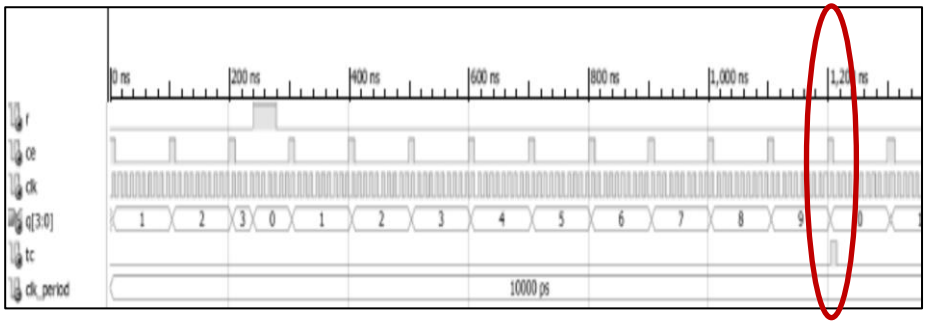


Figure 9 : Chronogramme du signal TC généré par l'entité ***counterDec_4b_RE****.*

3 Préparation

3.1 Préparation pour les fonctions de type combinatoire

1. Fonction **AND4n2** (**chronometer**):

   La fonction **AND4n2** est une fonction ET de 4 entrées A, B, C, D pour laquelle les entrées C et D sont inversées (voir figure 5 architecture de l'entité **chronometer**) :

   - écrire la table de de la fonction,
   - écrire, à partir de syntaxes d'équations concurrentes, le code **VHDL** de la fonction,

2. Fonction **equ45min** (**chronometer**) :
   - écrire la table de vérité simplifiée de la fonction,
   - écrire, à partir de syntaxes d'équations concurrentes, le code **VHDL** de la fonction,
   - définir un vecteur de test.

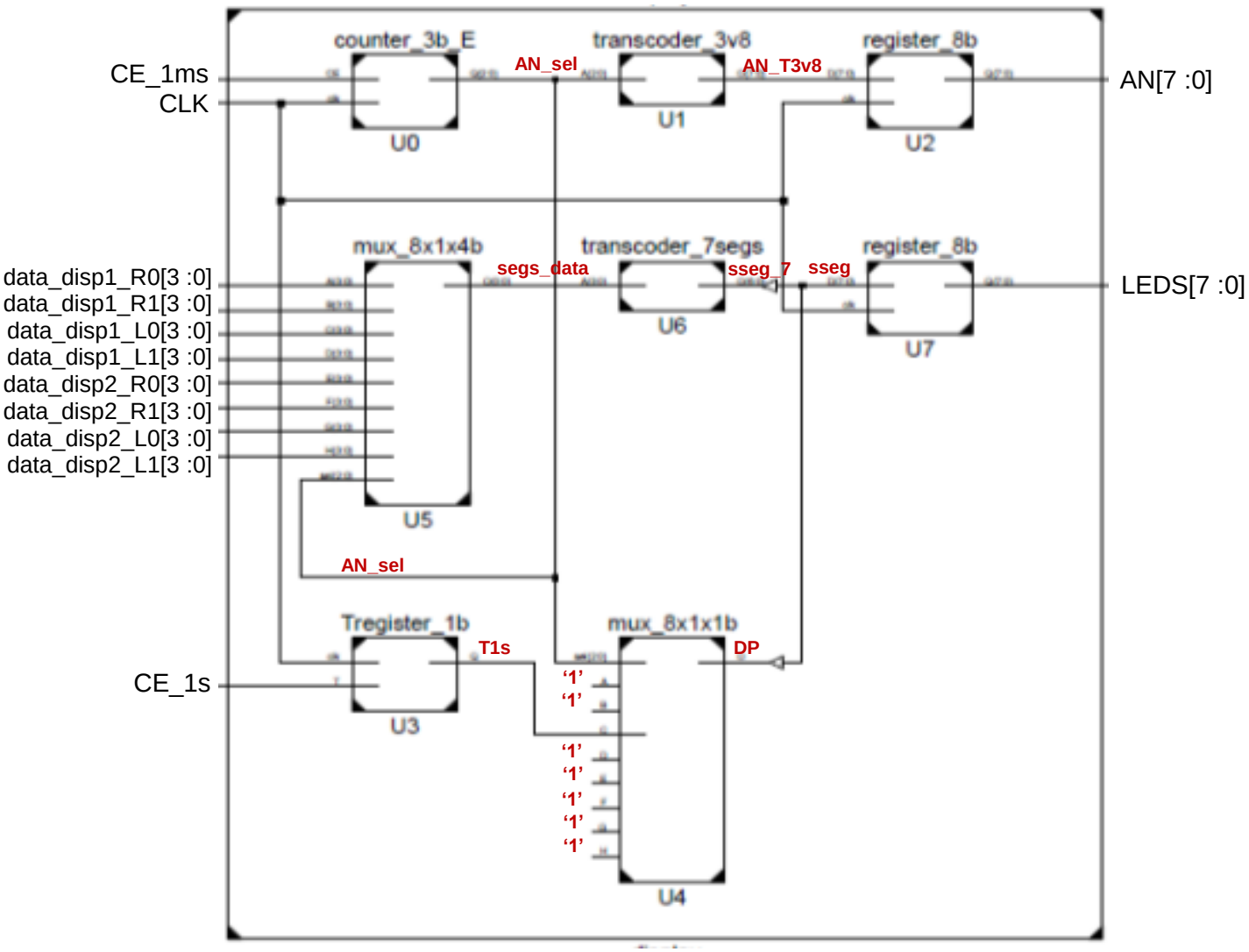
3.2 Préparation pour les fonctions de type séquentiel

1. Fonction **register_1b** (**score**) :
   - écrire la table de vérité de la fonction,
   - écrire, à partir de syntaxes de « process », le code **VHDL** de la fonction,
   - définir un vecteur de test.
2. Fonction **register_1b_R** (**chronometer**) :
   - écrire la table de vérité de la fonction,
   - écrire, à partir de syntaxes de « process », le code **VHDL** de la fonction,
   - définir un vecteur de test mettant en évidence le mode de fonctionnement « normal »,
   - définir un vecteur de test mettant en évidence le rôle de l'entrée **R**.
3. Fonction **register_1b_E** (**score**) :
   - écrire la table de vérité de la fonction,
   - écrire, à partir de syntaxes de « process », le code **VHDL** de la fonction,
   - définir un vecteur de test mettant en évidence le mode de fonctionnement « normal »,
   - définir un vecteur de test mettant en évidence le rôle de l'entrée **CE**.
4. Fonction **counterSen_4b_RE** (**chronometer**) :
   - écrire, à partir de syntaxes de « process », le code **VHDL** de la fonction,
   - définir un vecteur de test mettant en évidence le mode de fonctionnement « normal »,
   - définir un vecteur de test mettant en évidence le rôle de l'entrée **CE**.
   - définir un vecteur de test mettant en évidence le rôle de l'entrée **R**.
5. Fonction **counterDec_4b_RE** (**chronometer et score**) :
   - écrire, à partir de syntaxes de « process », le code **VHDL** de la fonction,

- définir un vecteur de test mettant en évidence le mode de fonctionnement « normal »,
- définir un vecteur de test mettant en évidence le rôle de l'entrée **CE**.
- définir un vecteur de test mettant en évidence le rôle de l'entrée **R**.

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**.

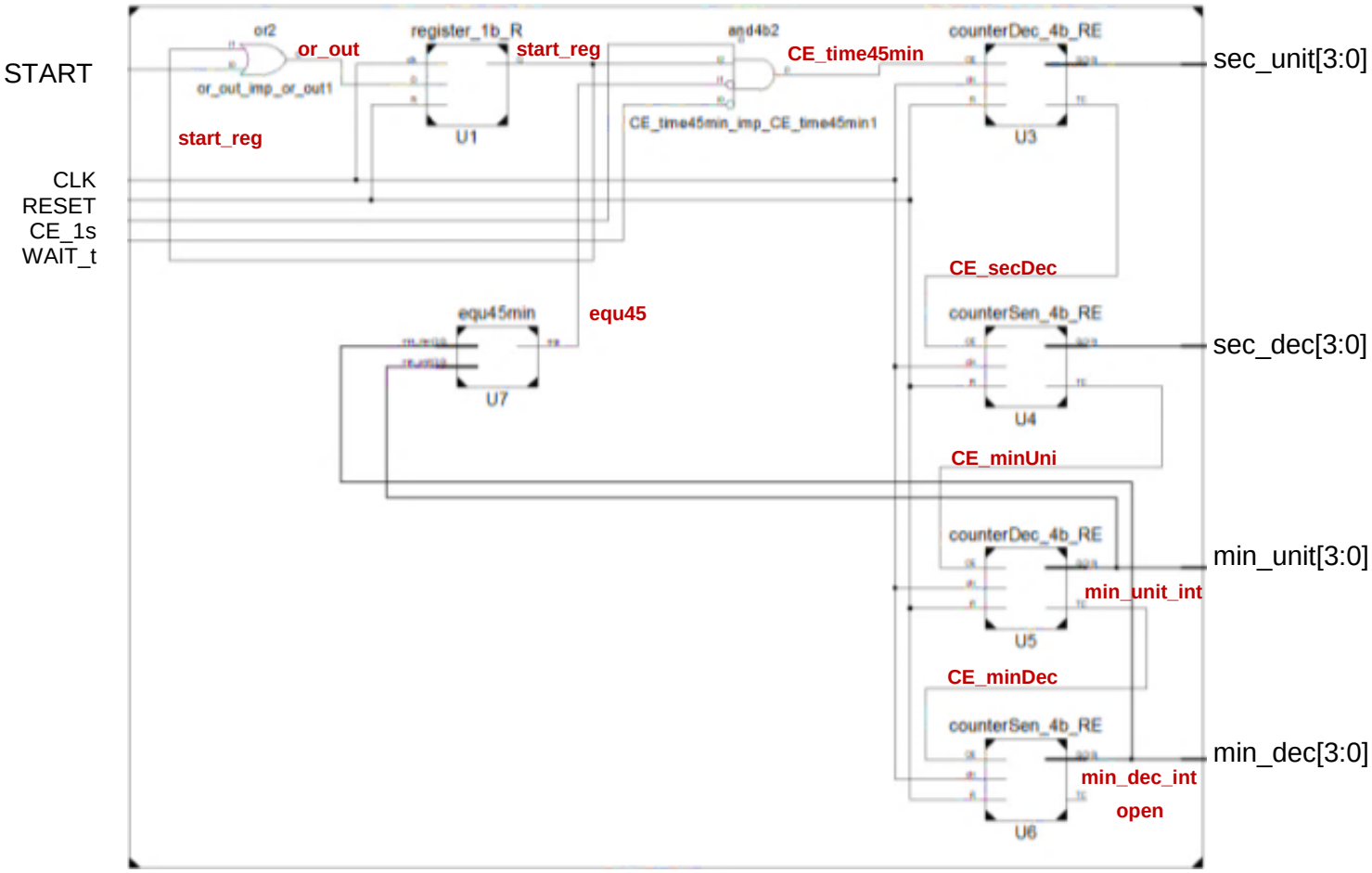
Le signal **sseg** est obtenu par concaténation de **DP** et **sseg_7** :

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

Architecture du sous-bloc **chronometer**



L'entité **chronometer** comporte les signaux internes suivants :

start_reg, **or_out**, **equ45**, **CE_time45min**,

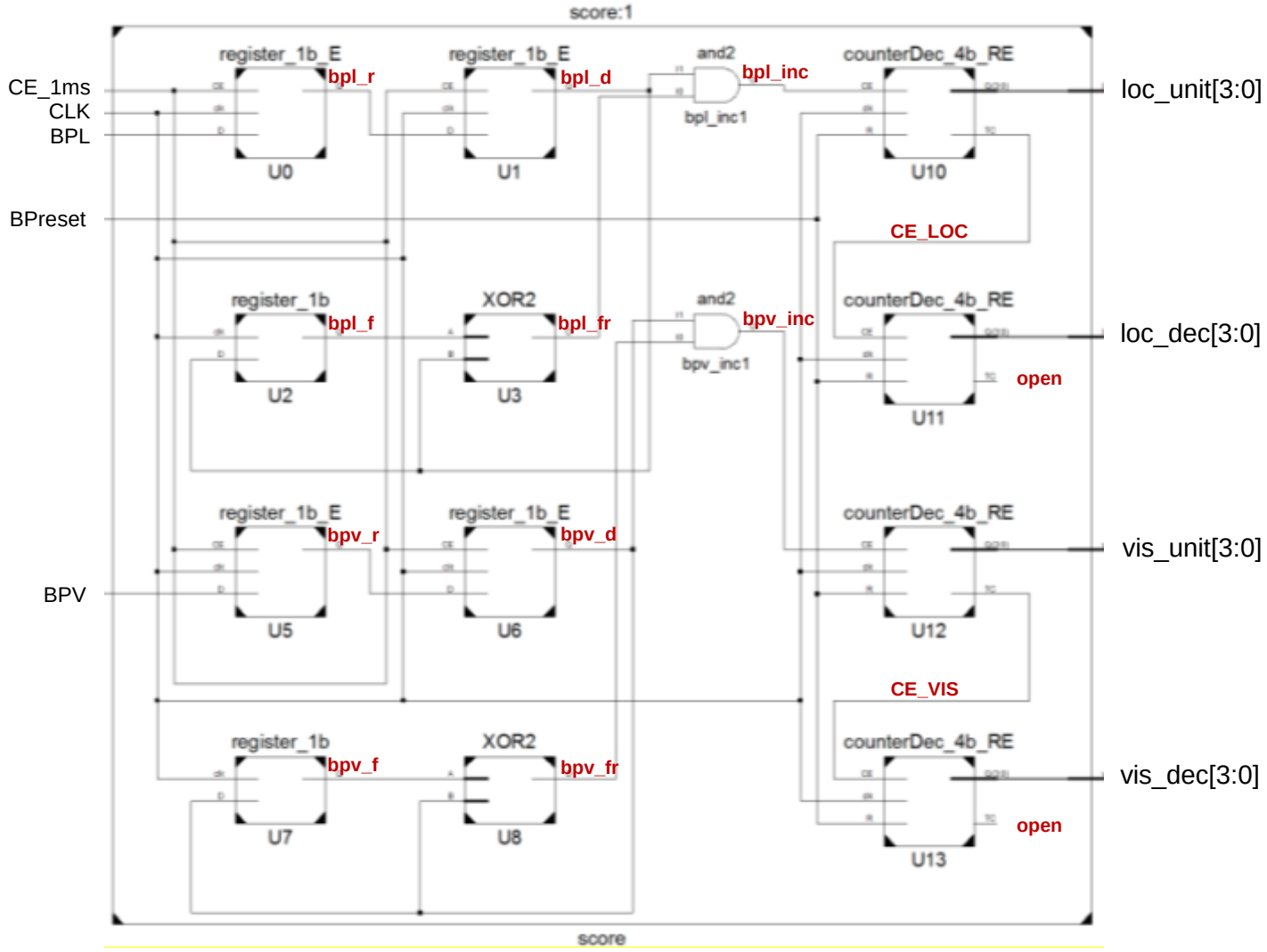
CE_secDec, **CE_minUni**, **CE_minDec**,

min_unit_int, **min_dec_int**.

La sortie **TC** non utilisée de l'entité **counterSen_4b_RE** des minutes (U6) est positionnée à **'open'**.

Annexe 3

Architecture du sous-bloc **score**



L'entité **score** comporte les signaux internes suivants :

bpl_r, **bpl_d**, **bpl_f**, **bpl_fr**, **bpl_inc**, **CE_LOC**,

bpv_r, **bpv_d**, **bpv_f**, **bpv_fr**, **bpv_inc**, **CE_VIS**.

Les sorties non utilisées des entités **counterDec_4b_RE** (U11 et U13) sont positionnées à **'open'**.