BURNOT Jean-Christophe 3 ETI – Groupe C GUZELIAN Raphaël

KLEITZ Antoine

COMPTE-RENDU PROJET CHRONOSCORE

**Sommaire**

[I) Introduction: 5](#__RefHeading___Toc2289_1921084482)

[II) Objectifs: 5](#__RefHeading___Toc2291_1921084482)

[Le système à développer devra afficher: 5](#__RefHeading___Toc2850_1921084482)

[Doit avoir comme fonctionnalités: 5](#__RefHeading___Toc2852_1921084482)

[L'affichage se fera sur: 5](#__RefHeading___Toc2856_1921084482)

[Les boutons et interrupteurs seront: 5](#__RefHeading___Toc2854_1921084482)

[III) Présentation de la carte: 6](#__RefHeading___Toc2293_1921084482)

[A)La carte: 6](#__RefHeading___Toc2328_1921084482)

[B)Bus de configuration / Alimentation: 6](#__RefHeading___Toc2330_1921084482)

[C)Boutons utilisés dans le projet: 6](#__RefHeading___Toc2332_1921084482)

[IV) Architecture de Chronoscore: 7](#__RefHeading___Toc2295_1921084482)

[A)Fonctions spécifiques et Schéma général (modules instanciés dans Chronoscore): 7](#__RefHeading___Toc2297_1921084482)

[Entrées et Sorties du bloc Chronoscore: 7](#__RefHeading___Toc2334_1921084482)

[Schéma général du bloc chronoscore: 9](#__RefHeading___Toc2336_1921084482)

[Fonctions spécifiques et rôle des modules: 9](#__RefHeading___Toc2338_1921084482)

[Fonctionnement général du bloc: 10](#__RefHeading___Toc2340_1921084482)

[B)Le bloc Display: 11](#__RefHeading___Toc2299_1921084482)

[Entrées et Sorties du bloc display: 11](#__RefHeading___Toc2342_1921084482)

[Schéma général du bloc display: 12](#__RefHeading___Toc2344_1921084482)

[Fonctions spécifiques et rôle des modules: 12](#__RefHeading___Toc2346_1921084482)

[Fonctionnement général du bloc: 13](#__RefHeading___Toc2348_1921084482)

[Chronogramme (Simulation behavioral): 13](#__RefHeading___Toc2350_1921084482)

[Fonctions intéressantes à détailler: transcoder\_7segs 14](#__RefHeading___Toc2942_1921084482)

[C)Le bloc Score: 15](#__RefHeading___Toc2301_1921084482)

[Entrées et Sorties du bloc Score: 15](#__RefHeading___Toc2352_1921084482)

[Schéma général du bloc Score: 16](#__RefHeading___Toc2354_1921084482)

[Fonctions spécifiques et rôle des modules: 16](#__RefHeading___Toc2356_1921084482)

[Fonctionnement général du bloc: 17](#__RefHeading___Toc2358_1921084482)

[Chronogramme (Simulation behavioral): 17](#__RefHeading___Toc2360_1921084482)

[D)Le Bloc Chronometer: 18](#__RefHeading___Toc1458_2458708573)

[Entrées et Sorties du bloc Chronometer: 19](#__RefHeading___Toc2362_1921084482)

[Schéma général du bloc Chronometer: 20](#__RefHeading___Toc2364_1921084482)

[Fonctions spécifiques et rôle des Chronometer: 20](#__RefHeading___Toc2366_1921084482)

[Fonctionnement général du chronometer: 21](#__RefHeading___Toc2368_1921084482)

[Chronogramme (Simulation behavioral): 21](#__RefHeading___Toc2370_1921084482)

[Fonctions intéressantes à détailler: eq45min 21](#__RefHeading___Toc2938_19210844821)

[Fonctions intéressantes à détailler: Register\_1b\_R 22](#__RefHeading___Toc2938_192108448211)

[Fonctions intéressantes à détailler: counterDec\_4b\_RE 22](#__RefHeading___Toc2938_192108448212)

[V)Méthode d'implémentation / Test de Chronoscore: 22](#__RefHeading___Toc2305_1921084482)

[VI)Conclusion: 23](#__RefHeading___Toc2307_1921084482)

[A)Retour d'expérience 23](#__RefHeading___Toc2309_1921084482)

[B)Opinion sur le projet 23](#__RefHeading___Toc2311_1921084482)

[C)Discussion sur de potentiels problèmes rencontrés / solutionnés ou pas 23](#__RefHeading___Toc2313_1921084482)

[D)Proposition d'ajout ou modification au projet pour améliorer chronoscore 24](#__RefHeading___Toc2315_1921084482)

[VII)Annexes: 26](#__RefHeading___Toc2317_1921084482)

[A)Chronogrammes 26](#__RefHeading___Toc2319_1921084482)

[Phase1: transcoder\_3v8: 26](#__RefHeading___Toc1488_1874340578)

[Phase1: mux\_8x1x4b: 26](#__RefHeading___Toc1490_1874340578)

[Phase1: transcoder\_7segs: 27](#__RefHeading___Toc1492_1874340578)

[Phase1: mux\_8x1x1b: 27](#__RefHeading___Toc1494_1874340578)

[Phase1: register\_8b: 28](#__RefHeading___Toc1496_1874340578)

[Phase1: Tregister\_1b: 28](#__RefHeading___Toc1498_1874340578)

[Phase1: counter\_3b\_E: 29](#__RefHeading___Toc1500_1874340578)

[Phase2: register\_1b\_R: 29](#__RefHeading___Toc1502_1874340578)

[Phase2: equ45min: 29](#__RefHeading___Toc1504_1874340578)

[Phase2: counterDec\_4b\_RE: 30](#__RefHeading___Toc1506_1874340578)

[Phase2: counterSen\_4b\_RE: 30](#__RefHeading___Toc1508_1874340578)

[Phase2: register\_1b\_E: 30](#__RefHeading___Toc1510_1874340578)

[Phase2: counterDec\_4b\_RE: 31](#__RefHeading___Toc1512_1874340578)

[Phase2: register\_1b: 31](#__RefHeading___Toc1514_1874340578)

[B)Vue RTL 32](#__RefHeading___Toc2321_19210844821)

[Phase1: transcoder\_3v8: 32](#__RefHeading___Toc1488_18743405781)

[Phase1: mux\_8x1x1b: 32](#__RefHeading___Toc1494_18743405781)

[Phase1: Tregister\_1b: 33](#__RefHeading___Toc1498_18743405781)

[Phase1: counter\_3b\_E: 33](#__RefHeading___Toc1500_18743405781)

[Phase2: equ45min: 34](#__RefHeading___Toc1504_18743405781)

[C)Codes VHDL 34](#__RefHeading___Toc2321_1921084482)

[D)Afficheur 7 segments 35](#__RefHeading___Toc2484_1921084482)

# I) Introduction:

Le projet chronoscore vise à développer, un circuit intégré numérique fournissant à un système d’affichage pour des match de foot. Ce circuit permettra d'afficher le score et le temps pendant la partie. Cet affichage se fera au moyen d'afficheur 7 segments et d'une sortie VGA.

# II) Objectifs:

## Le système à développer devra afficher:

* Le score de l'équipe locale et de l'équipe visiteur
* Le temps écoulé

## Doit avoir comme fonctionnalités:

* Possibilité d'incrémenter le score des différentes équipes (quand le chronomètre est en fonction)
* Blocage du chronomètre à 45min (mi-temps)
* Pause du chronomètre sur demande
* Remise à 0 du score et remise à 0 du chronomètre

## L'affichage se fera sur:

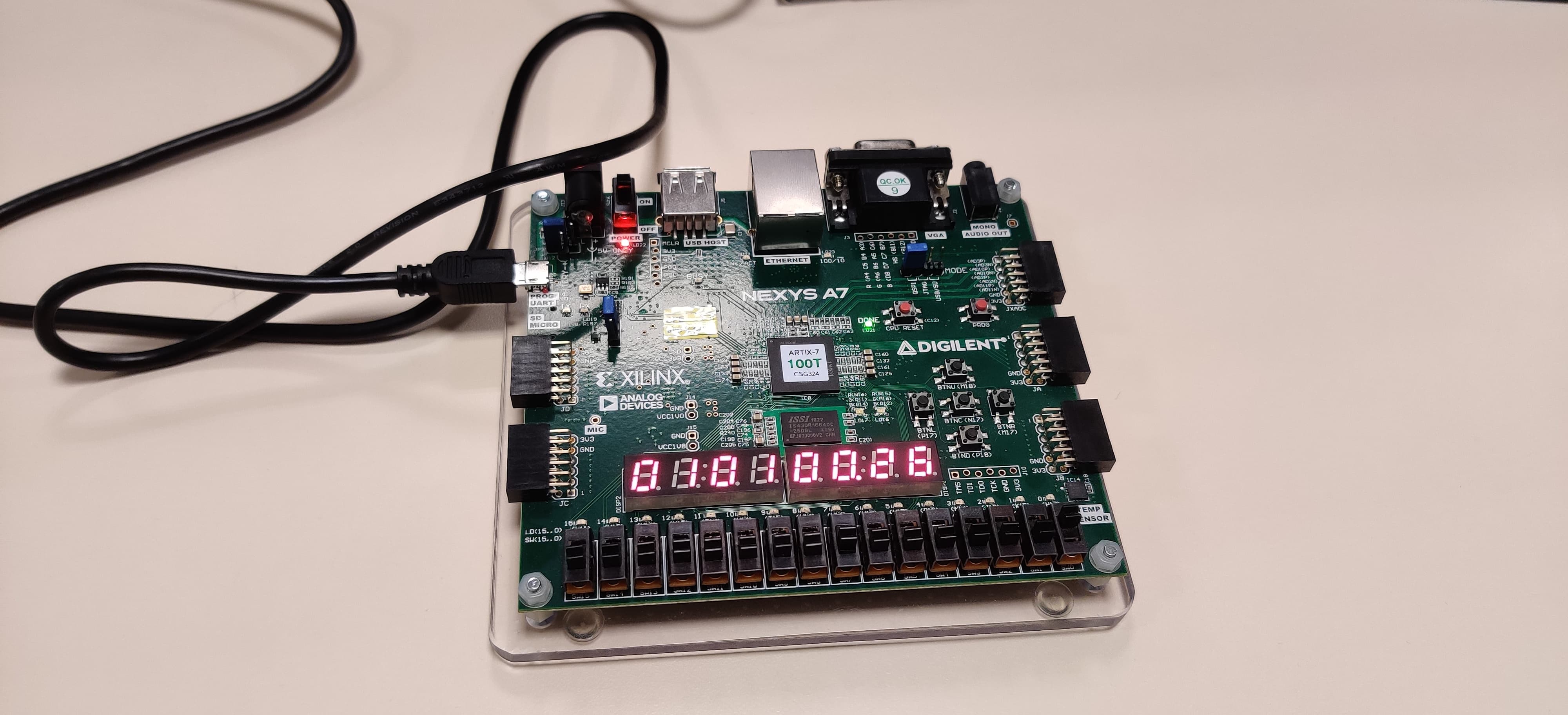
* Un ensemble de 8 afficheur 7 segments
* Un écran compatible avec le protocole VGA

## Les boutons et interrupteurs seront:

* Interrupteur START,
* Interrupteur WAIT\_t,
* Bouton poussoir RESET (mettre le chronomètre à 0),
* Bpreset (remise à 0 du score),
* VGAONOFF (pour activer ou désactiver l'affichage VGA)
* TESTVGA (Pour tester l'affichage VGA)

# III) Présentation de la carte:

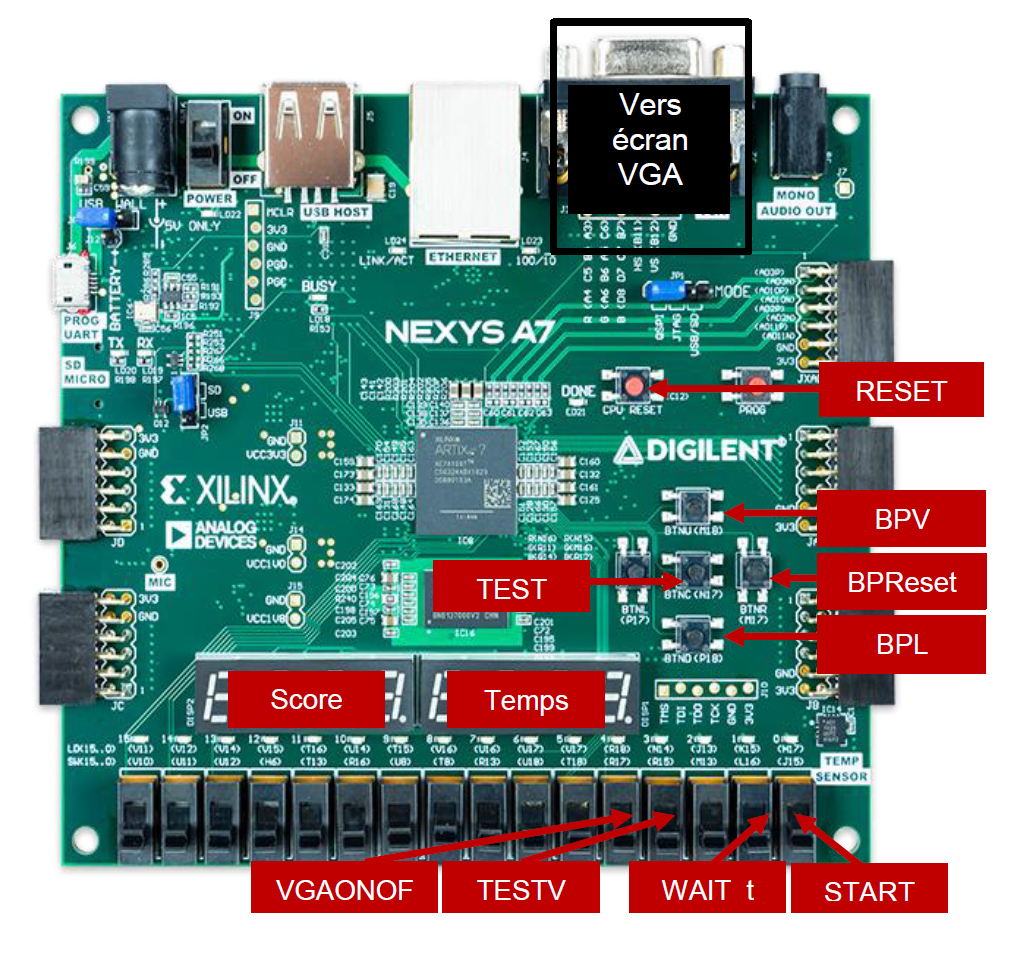
## A)La carte:

La carte utilisée est une NEXUS A7 de la marque DIGILENT

## B)Bus de configuration / Alimentation:

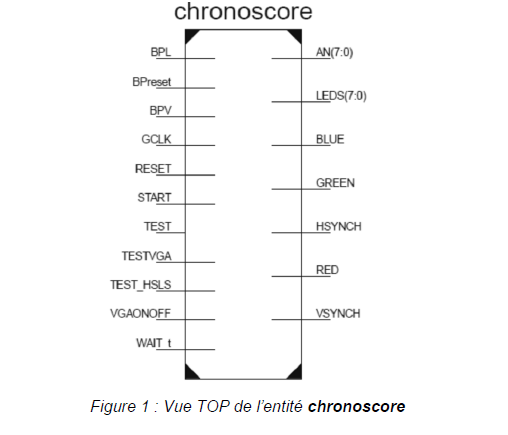
L'alimentation de la carte et l'injection du code programmé en VHDL se fait via le port micro USB de la carte

## C)Boutons utilisés dans le projet:



# IV) Architecture de Chronoscore:

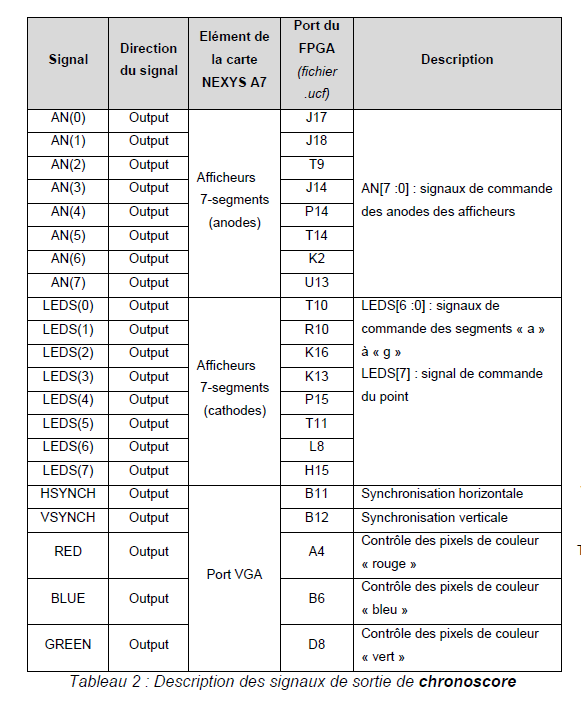
## A)Fonctions spécifiques et Schéma général (modules instanciés dans Chronoscore):

Nous pouvons représenter le système avec ses entrées et ses sorties grâce à la vue TOP:

### Entrées et Sorties du bloc Chronoscore:

Les entrées sont récapitulées dans le tableau suivant:

Nous avons comme sorties:



### Schéma général du bloc chronoscore:

### Fonctions spécifiques et rôle des modules:

Les sous-blocs composants chronoscore sont:

* **chronometer**: Il fournit aux sous-blocs displayet vgaDisplayles données temporelles (minutes et secondes) à afficher. (source: sujet TP)
* **display**: Il permet de gérer les données à afficher sur les 8 afficheurs 7-segments de la carte NEXYS A7. (source: sujet TP)
* **score**: Il fournit aux sous-blocs multiplexDataet vgaDisplayles scores des équipes (unités et dizaines) à afficher. (source: sujet TP)
* **vgaDisplay**: Il génère deux signaux de synchronisation HSYNCHet VSYNCHainsi que l'affichage d'images auto-générées. (source: sujet TP)
* **timeGenerator**: Il représente l’horloge de notre carte NEXYS A7.

Nous avons également des portes logiques:

* **inv**: porte logique NON à une entrée.
* **and2b1**: porte logique ET à deux entrés.

### Fonctionnement général du bloc:

Le bloc chronoscore est le bloc de plus haut niveau de notre projet. Il gère les deux fonctionnalités de notre chronoscore.

* Affichage du temps de jeu avec arrêt à 45min sur un afficheur 7 segments.
* Affichage et gestion du score sur un afficheur 7 segments.
* L'affichage sur un écran LCD compatible avec la norme VGA.

**Pour la partie temps de jeu:** Cette fonctionnalité repose sur l'horloge de la carte. Cette horloge est gérée par le bloc timeGenerator qui depuis cette horloge mère génère 3 horloges filles; CLK, CE\_1ms de période 1ms et CE\_1s de période 1s. L'horloges CE-1s est celle qui va permettre de compter le temps de jeu, l'autre servira pour de l'antirebond dans le bloc score. Ensuite le bloc chronometer va à l'aide des l'horloges CLK et CE\_1s compter le temps de jeu en secondes et en minutes. Le bloc chronometer est également le bloc qui gère le départ du chronomètre (signal START), la pause (signal WAIT\_t), la remise à zéro (signal RESET).

Les valeurs de temps ressortent de chronometer sous 4 variables de 4 bits. Chacune de ces variable correspond à un digit de l'affichage:

* sec\_unit correspond au digit des unités pour le compteur des secondes
* sec\_dec correspond au digit des dizaines pour le compteur des secondes
* min\_unit correspond au digit des unités pour le compteur des minutes
* min\_dec correspond au digit des dizaines pour le compteur des minutes

Ces 4 signaux vont ensuite commander le bloc display qui va en fonction de leurs valeurs allumer en temps réel les segments adéquats en jouant sur la persistance rétinienne pour obtenir quelque chose de lisible et afficher le temps de jeu.

**Pour la partie gestion du scores:** Cette fonctionnalité est gérée par le bloc score qui est le bloc qui s'occupe de l'incrémentation du score en fonction des bouton sur lesquels on appui.

* BPL incrémente le score de l'équipe locale
* BPV incrémente le score de l'équipe visiteur

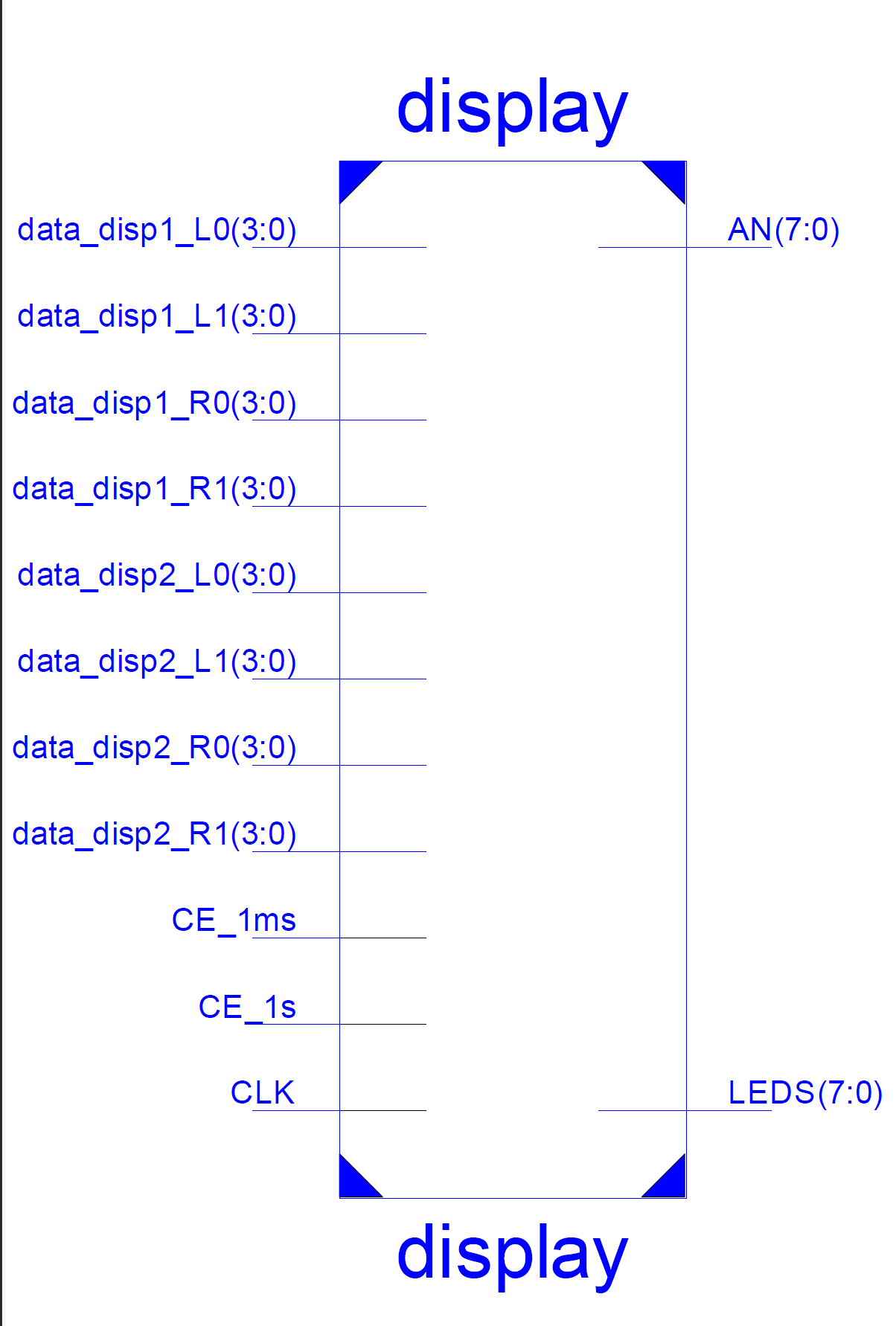
Les valeurs des scores ressortent du bloc score sous 4 variables de 4 bits. Chacune de ces variable correspond à un digit de l'affichage:

* loc\_unit correspond au digit des unités pour le score de l'équipe locale
* loc\_dec correspond au digit des dizaines pour le score de l'équipe locale
* vis\_unit correspond au digit des unités pour le score de l'équipe visiteur
* vis\_dec correspond au digit des dizaines pour le score de l'équipe visiteur

**Pour la partie gestion du scores:** On donne au bloc vgaDisplay les informations du chronomètre et des scores et il va s'occuper de l'affichage.

## B)Le bloc Display:

Nous pouvons représenter le système avec ses entrées et ses sorties grâce à la vue TOP:



### Entrées et Sorties du bloc display:

Les entrées du bloc display sont:

* **CE\_1ms**: horloge de période 100ns
* **CLK**: horloge de période 10 ns
* **data\_disp**: vecteurs de dimension 4 permettant de transmettre l’information sur le caractère hexadécimal à afficher
* **CE\_1s**: horloge de période 100 ns

Les sorties du bloc display sont:

* **AN**: vecteur de dimension 8 permettant de passer d'un afficheur 7 segment à un autre de manière rapide (assez rapide pour que l’œil ne distingue pas).
* **LEDS**: vecteur de dimension 8 permettant d'allumer les segments adéquats de l'afficheur 7 segments.

### Schéma général du bloc display:

### Fonctions spécifiques et rôle des modules:

* **counter\_3b\_E:** compteur 3 bits actif sur front montantdu signal d'horloge. Il dispose d’une entrée de validation CE (Clock Enable) synchrone et active à l'état HAUT.
* **transcoder\_3v8:** 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.
* **register\_8b:** registre synchrone constitué de 8 bascules D actives sur front montant du signal d'horloge.
* **mux\_8x1x4b:** multiplexeur de 8 signaux d’entrée de 4 bits vers 1 signal de sortie de 4 bits. Il dispose de trois signaux de sélection.
* **transcoder\_7segs:** transcodeur qui associe à un code binaire sur 4 bits, le code 7-segments correspondant à la valeur hexadécimale de l’entrée.
* **Tregister\_1b:** 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.
* **mux\_8x1x1b:** multiplexeur de 8 signaux d’entrée de 1 bit vers 1 signal de sortie de 1 bit. Il dispose de trois signaux de sélection.

### Fonctionnement général du bloc:

Le sous-bloc displaygère les données à afficher sur les 8 afficheurs 7-segments de la carte NEXYS A7.

Les quatre afficheurs de droite (afficheurs 0 à 3) indiquent le temps en minutes et secondes.

Les quatre afficheurs de gauche (afficheurs 4 à 7) indiquent le score des deux équipes.

(source: sujet TP)

Les Informations concernant le temps et le score arrivent en temps réel sur le bloc mux\_8x1x4b.

Ce sont les 8 valeurs de data. Le mux est commandé par le signal AN\_sel sortie du compteur.

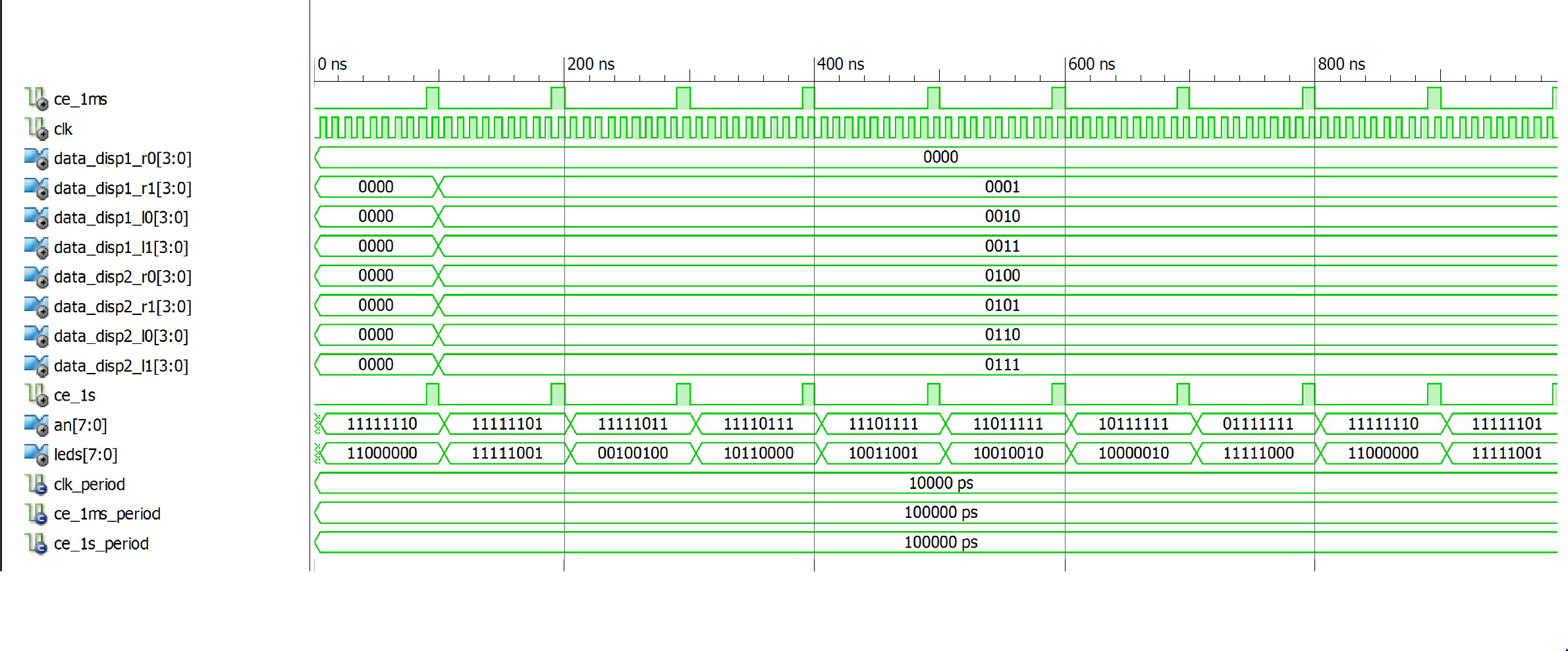
Le signal AN\_sel est le signal sortie d'un compteur 3 bits. Cette sortie va compter de 0 à7 et faire sortir du mux l'information sur l'afficheur 7 segments voulu. Le signal AN\_sel est aussi le signal qui après passage dans un transcodeur et stockage dans un bascule va donner AN (le signal qui indique le 7 segments).

Le fait que ce soit le signal AN-sel qui commande le mux fait que pour chaque valeur de AN (qui représente un afficheur 7 segment) on aura l'information correcte à a afficher (sortie du mux (segs\_data)). Cette sortie du mux passe ensuite par un transcodeur pour passer d'un code binaire sur 4 bits au code de l'afficheur 7 segments.

Les bascules en sortie du montage permettent de maintenir la valeur de l'affichage jusqu'au moment où les entrées se mettent à changer.

Le Tregister permet d'avoir le clignotement du point et le mux de concaténer l'état du point avec l'état des autres 7 segments

### Chronogramme (Simulation behavioral):

****

Ce chronogramme illustre nos propos. On se place dans la situation où la data ne varie pas au cours du temps. On voit que AN\_sec varie de 0 à 7 et que pour chaque changement réalisé la valeur leds nous donne les segments allumés (actif à l'état bas). Vous trouverez en annexe le tableau pour la correspondance entre le bus binaire et le numéros affiché.

### Fonctions intéressantes à détailler: transcoder\_7segs

C'est la fonction qui traduit un valeur d'un digit (de 0 à 15) codé sur 4 bits en un code qui permet d'allumer les bon segments de l'afficheur (voir annexe).

Les segments sont actifs à l'état bas.

Voici le tableau de vérité de cette fonction:

|  |  |  |
| --- | --- | --- |
| A[3:0] | Caractère hexadécimal à afficher | O[6:0] |
| 0000 | 0 | 1000000 |
| 0001 | 1 | 1111001 |
| 0010 | 2 | 0100100 |
| 0011 | 3 | 0110000 |
| 0100 | 4 | 0011001 |
| 0101 | 5 | 0010010 |
| 0110 | 6 | 0000010 |
| 0111 | 7 | 1111000 |
| 1000 | 8 | 0000000 |
| 1001 | 9 | 0010000 |
| 1010 | A | 0001000 |
| 1011 | b | 0000011 |
| 1100 | C | 1000110 |
| 1101 | d | 0100001 |
| 1110 | E | 0000110 |
| 1111 | F | 0001110 |

## C)Le bloc Score:

Nous pouvons représenter le système avec ses entrées et ses sorties grâce à la vue TOP:

### 

### Entrées et Sorties du bloc Score:

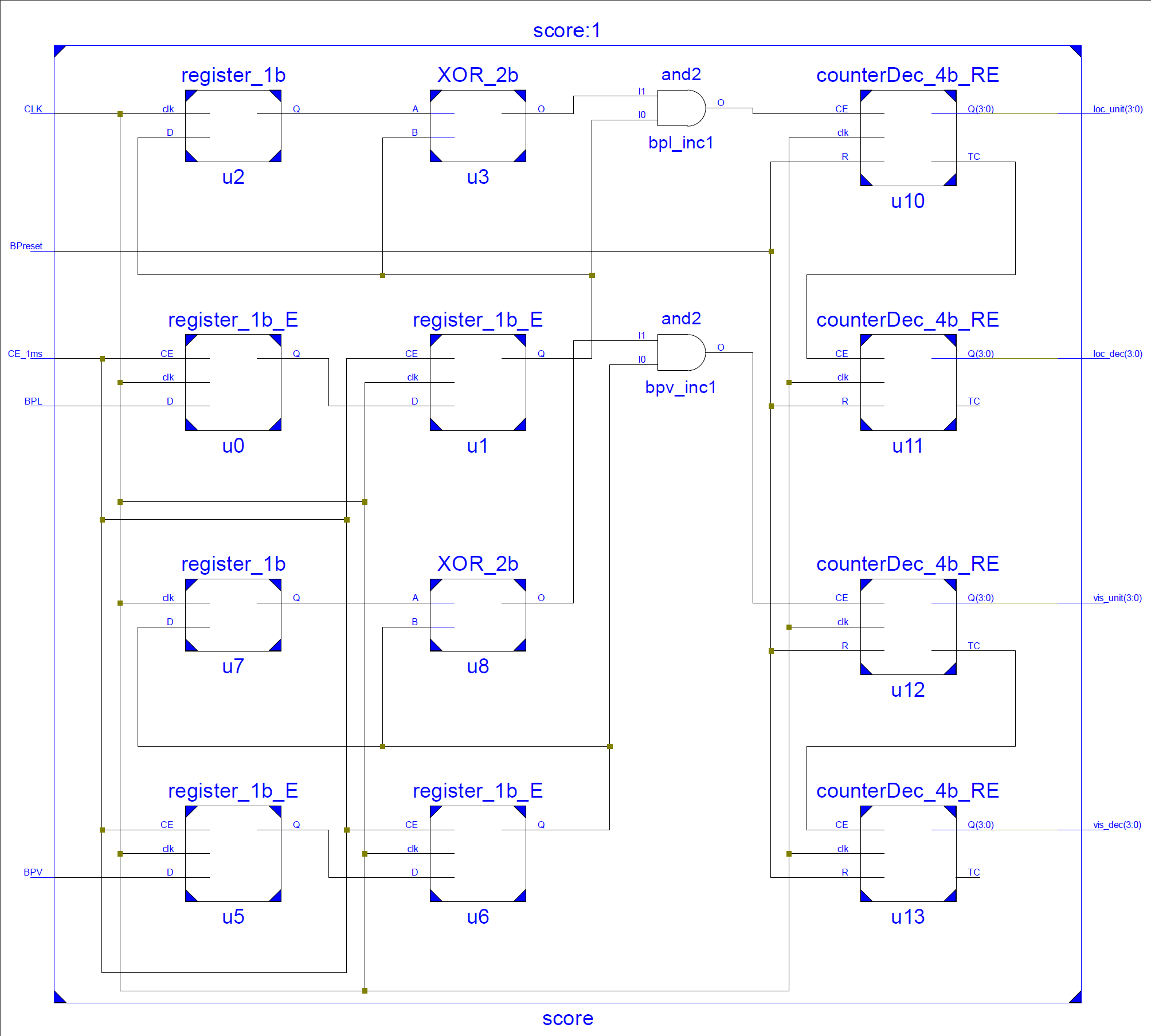
Les entrées du bloc score sont:

* **BPL:** Incrémentation du score pour l’équipe locale.
* **Bpreset:** Remise du score.
* **BPV:** Incrémentation du score pour l’équipe visiteurs.
* **CE\_1ms:** horloge de période 100 ns, durée à l’état haut 10 ns.
* **CLK:** horloge de période 10 ns, durée à l’état HAUT 10 ns.

Les sorties du bloc score sont:

* **loc\_unit:** bus de 4 bits qui représente l’unité du score de l’équipe locale, quand il passe à l’équivalent décimal de 9, le RESET le refait passer à 0 et loc\_dec s’incrémente de 1.
* **loc\_dec:** bus de 4 bits qui représente la dizaine du score de l’équipe locale. Dans notre étude nous n’avons pas étudié le cas où l’on dépasse 99 buts.
* **vis\_unit:** bus de 4 bits qui représente l’unité du score de l’équipe visiteur, quand il passe à l’équivalent décimal de 9, le RESET le refait passer à 0 et vis\_dec s’incrémente de 1.
* **vis\_dec:** bus de 4 bits qui représente la dizaine du score de l’équipe visiteur. Dans notre étude nous n’avons pas étudié le cas où l’on dépasse 99 buts.

### Schéma général du bloc Score:



### Fonctions spécifiques et rôle des modules:

* **register\_1b\_E:** registre 1 bit à 3 entrées. 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.
* **counterDec\_4b\_RE:** compteur 4 bits qui compte de 0 à 9 et qui est actif sur front montant du signal d'horloge clk.
* **register\_1b:** registre synchrone constitué d’une bascule D active sur front montant du signal d'horloge clk.
* **And2:** porte ET qui prend comme entrée XOR2 et register\_1b\_E et donne en sortie CE, le signal d’activation du compteur counterDec\_4b\_RE
* **XOR2:** porte OU-EXCLUSIF qui prend comme entrée register\_1b\_E et register\_1b

### Fonctionnement général du bloc:

Le bloc score a trois fonctions:

* Détecter les fronts des signaux BPL et BPV correspondants aux appuis de l'utilisateur sur les boutons servant à incrémenter le score de l'équipe locale pour BPL et de l'équipe visiteur pour BPV.
* Gérer l'incrémentation du score (nombre d'appui sur BPL et BPV).
* Avoir un mécanisme d'anti-rebond pour éviter une mauvaise incrémentation du score.

Pour rajouter un but (incrémenter le score) l'utilisateur appui sur les bouton BPL et BPV. Cet appui génère un front montant. Le front montant est capté par les différents registres (register\_1b\_E) en entrée du bloc (détection).

Ces registres vont envoyer le front au registres suivant qui vont incrémenter le compteur.

Le front montant passe donc par 2 registres avant d'incrémenter le compteur (notion importante pour l'antirebond).

il faut un mécanisme qui évite que pour chaque appui de bouton on incrémente notre score du nombre de front montant que le système voit quand le bouton est à 1. Cette sécurité est assuré par la logique combinatoire (and2) qui dépend également des lignes 1 et 3 (register\_1b suivi de XOR\_2b).

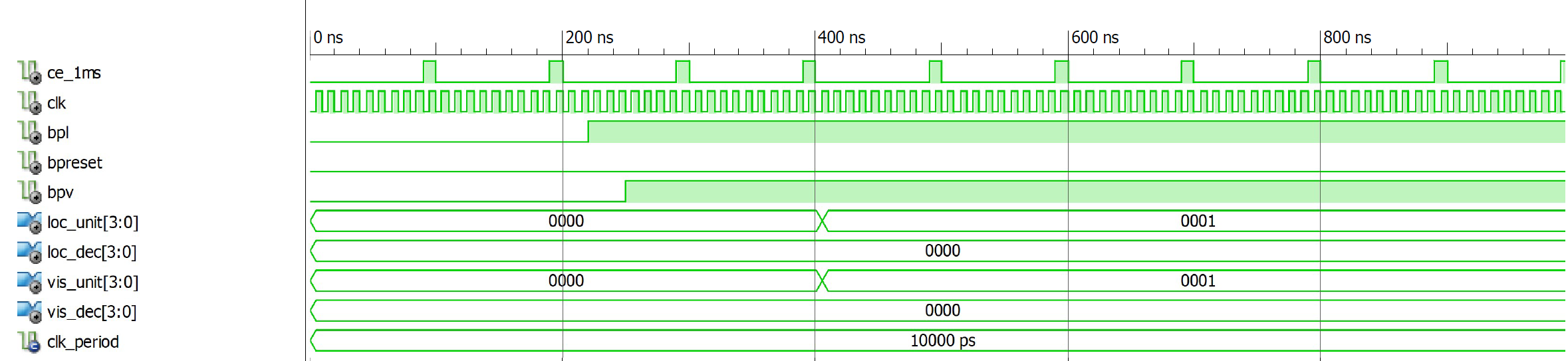
On a alors un montage qui est capable d'associer à un appui de bouton une incrémentation de 1 au niveau du score. Il reste encore le soucis de l’anti-rebond à régler.

Pour l'antirebond, cette protection se fait par un clock enable cadencé à 1ms sur deux étages de registres (register\_1b\_E).

Sachant que le rebond est un phénomène très court qui intervient juste après un appui sur un bouton, ce montage permet bien de l'éviter. En effet en mettant un clock enable de 1ms les appui humain bien supérieurs à 1ms seront forcément captés par le système alors que les rebonds auront des chances de tomber à un moment où la clock est désactivée. On répète ce montage deux fois et on obtient une protection contre les rebonds performante.

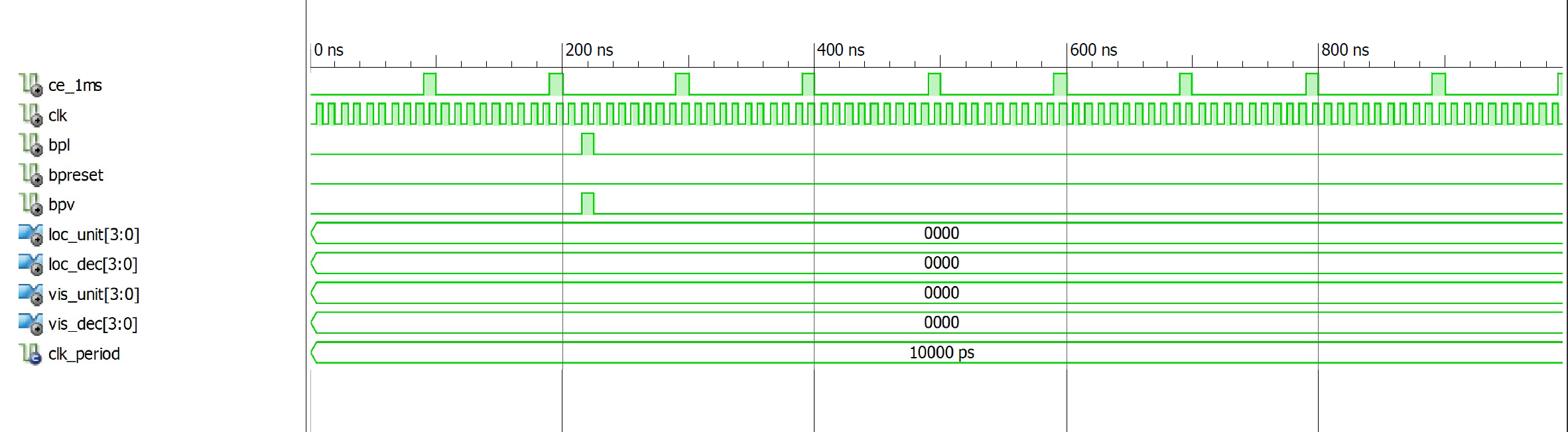
Le seul soucis que cette correction est le fait qu'un appui de l'utilisateur inférieur à 1ms ne sera pas détecté. Dans la pratiques les appuis de l'utilisateur sont toujours plus longs que 1ms.

### Chronogramme (Simulation behavioral):



Ce test bench nous montre que lorsqu'on laisse le bouton appuyé, le compteur ne s'incrémente pas du nombre de coup d'horloge.

Ce test bench nous montre que le système d’anti rebond est fonctionnel

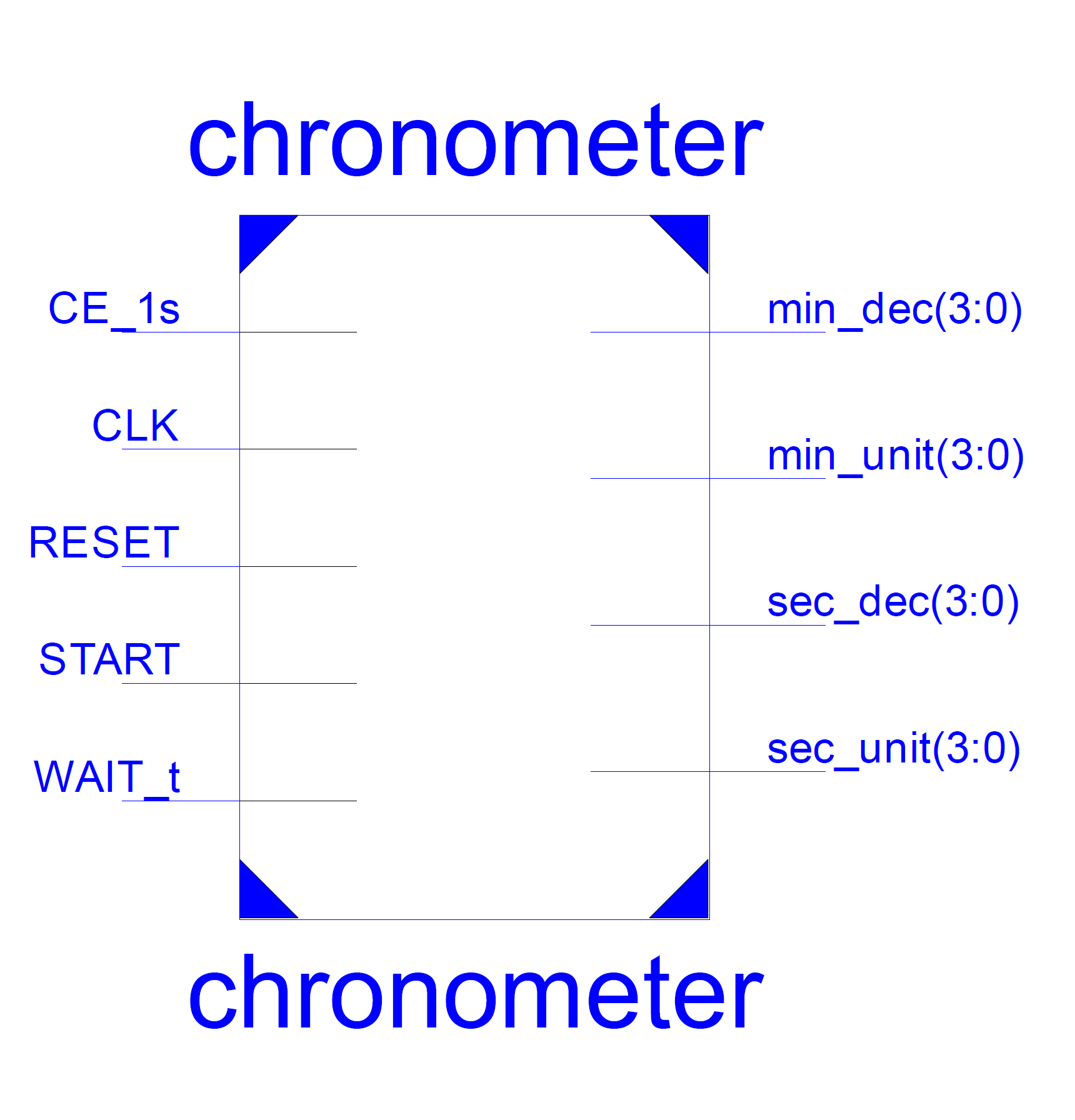


En effet on programme une impulsion très rapide sur les bouton bpv et bpl et on voit que le score ne s'incrémente pas.

### 

### D)Le Bloc Chronometer:

Nous pouvons représenter le système avec ses entrées et ses sorties grâce à la vue TOP:

****

### Entrées et Sorties du bloc Chronometer:

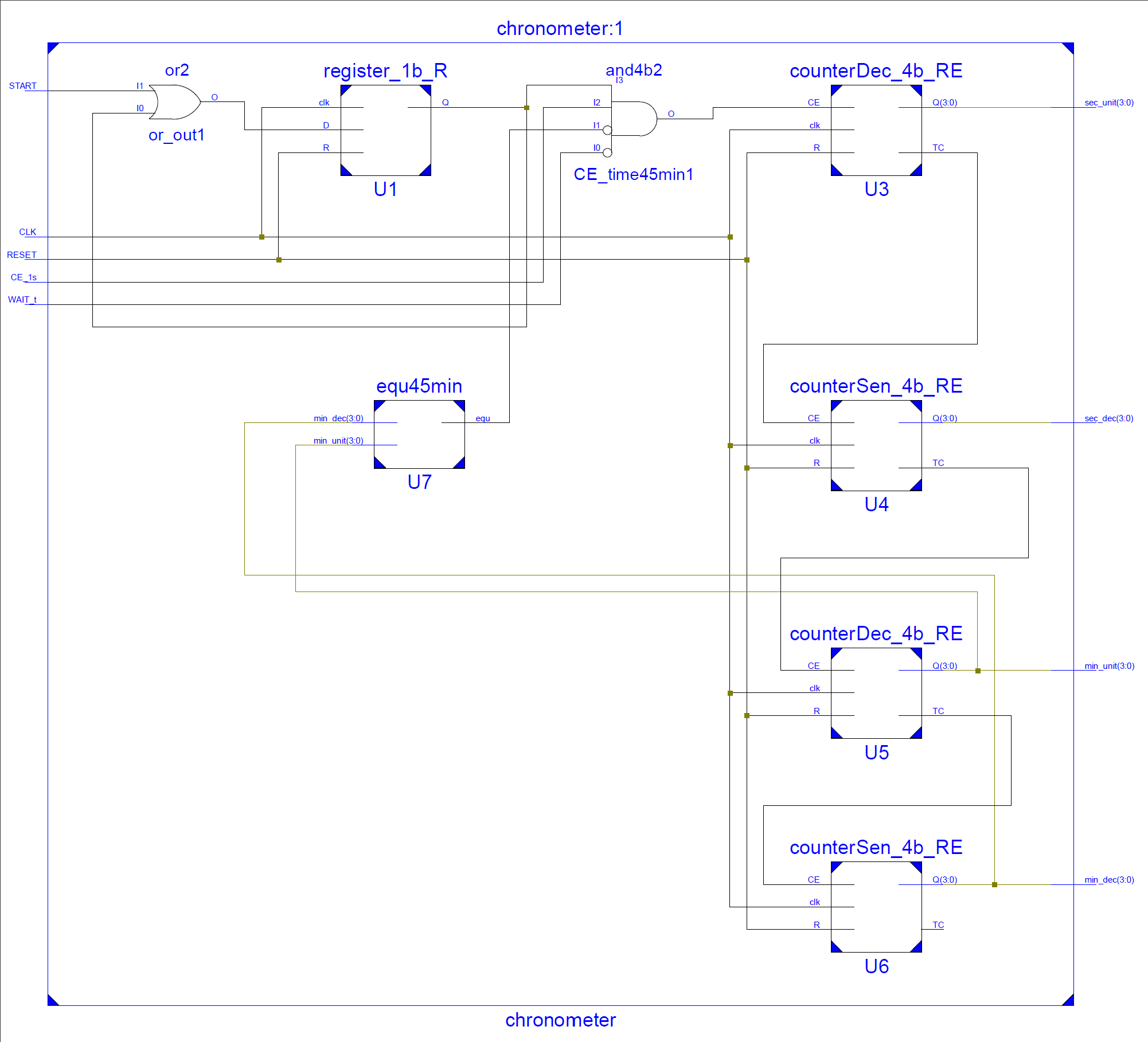
Les entrées du bloc chronometer sont:

* **CE\_1S:** horloge de période 100 ns, durée à l’état HAUT 10 ns.
* **CLK:** horloge de période 10 ns, durée à l’état HAUT 10 ns.
* **RESET:** Remise à zéro du chronomètre.
* **START:** Démarrage du chronomètre.
* **WAIT\_t:** Mise en pause du chronomètre.

Les sorties du bloc chronometer sont:

* **sec\_unit:** bus de 4 bits qui représente l’unité des secondes, quand il passe à l’équivalent décimal de 9, le RESET le refait passer à 0 et sec\_dec s’incrémente de 1.
* **sec\_dec:** bus de 4 bits qui représente la dizaine des secondes, quand il passe à l’équivalent décimal de 6, le RESET le refait passer à 0 et min\_unit s’incrémente de 1.
* **min\_unit:** bus de 4 bits qui représente l’unité des minutes, quand il passe à l’équivalent décimal de 9, le RESET le refait passer à 0 et min\_dec s’incrémente de 1.
* **min\_dec:** bus de 4 bits qui représente la dizaine des minutes, quand il passe à l’équivalent décimal de 4 et que min\_unit vaut 5, le WAIT\_t met en pause le chronomètre.

### Schéma général du bloc Chronometer:



### Fonctions spécifiques et rôle des Chronometer:

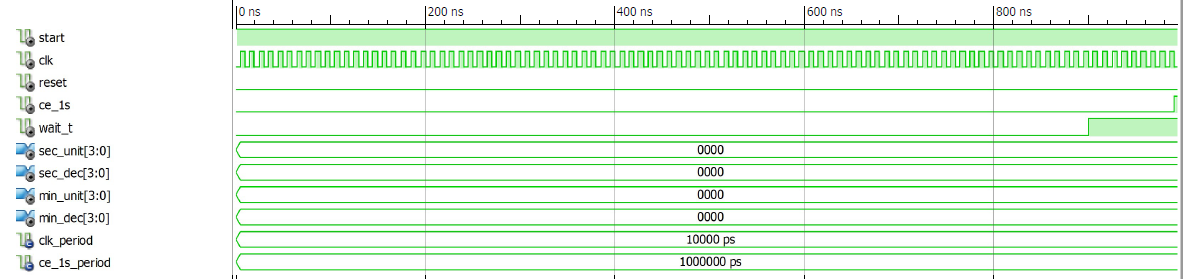
* **register\_1b\_R:** registre synchrone constitué d’une bascule D active sur front montant du signal d'horloge clk. Il dispose d’une entrée de remise à zéro R (Reset) synchrone active à l'état HAUT
* **counterDec\_4b\_RE:** compteur 4 bits qui compte de 0 à 9 et qui est actif sur front montant du signal d'horloge clk.
* **equ45min:** comparateur logique si l'entrée des unités vaut « 5 », et que celle des dizaines vaut « 4 », la sortie vaut « 1 », sinon elle vaut « 0 ».
* **counterSen\_4b\_RE:** 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. **Or2:** porte OU qui prend 2 entrées: START et start\_reg.
* **and4b2:** porte ET qui prend 4 entrées: start\_reg, CE\_1s, NON equ45, NON WAIT.

### Fonctionnement général du chronometer:

Le bloc chronometer est un bloc qui gère le chronomètre. Pour se faire Il utilise une horloge cadencé à 1s et 4 compteurs qui s'incrémentent les un les autres. Pour le blocage à 45min une incrémentation classique ne suffit pas on alors rajouté le bloc equ45min pour stopper le comptage à 45min en activant de la logique combinatoire.

L’anti-rebond sur l'interrupteur START est géré par or2 et register\_1b\_R. Il y a une comparaison entre la valeur actuelle précédente qui empêche les éventuels dysfonctionnement.

### Chronogramme (Simulation behavioral):

Ce test bench ne permet de conclure car la durée d’acquisition est courte. Il faudra augmenter la durée d’acquisition. La fonction est néanmoins fonctionnelle (vérification lors de l’implantation).

### Fonctions intéressantes à détailler: eq45min

|  |  |  |
| --- | --- | --- |
| min\_dec\_int\_open | min\_unit\_int | equ45 |
| 0100 (4) | 0101 (5) | 1 |
| Tous les autres cas |  | 0 |

Cette fonction permet de bloquer le compteur des minutes à 45 le tableau logique donne l'équation: 𝑒𝑞𝑢 <= ′1′ 𝑤ℎ𝑒𝑛 (𝑚𝑖𝑛\_𝑑𝑒𝑐 = "0100" 𝐴𝑁𝐷 𝑚𝑖𝑛\_𝑢𝑛𝑖𝑡 = "0101") 𝐸𝐿𝑆𝐸 ′0′

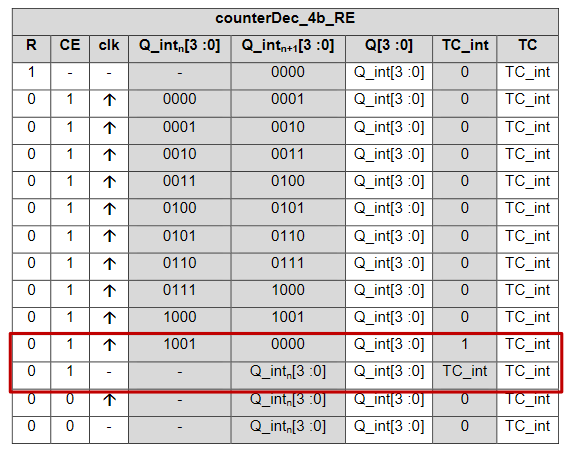
### Fonctions intéressantes à détailler: Register\_1b\_R

C'est un registre un bit à registre synchrone

|  |  |  |  |
| --- | --- | --- | --- |
| D | R | clk | Qn |
| X | 1 | rising-edge | 0 |
| 0 | 0 | rising-edge | 0 |
| 1 | 0 | rising-edge | 1 |
| X | X | - | Qn-1 |

### Fonctions intéressantes à détailler: counterDec\_4b\_RE

Il s’agit d’un compteur 4 bits avec une entrée d’activation CE synchrone (CLK) et un reset asynchrone. La table de vérité de cette fonction est :



# V)Méthode d'implémentation / Test de Chronoscore:

Pour vérifier que tous nos modules fonctionnaient normalement sur la carte, nous avons implémenté chaque module les uns après les autres. Les fichiers .ngc et .ucf de chaque module sont fournis sur eCampus. Avec ces fichiers .ngc et .ucf nous avons pu compléter le projet, de sorte à ce que chaque erreur soit facilement repérable dans chaque module.

Pour que cela soit plus clair, prenons l’exemple du sous bloc score:

D’abord nous ajoutons les fichiers timeGenerator.ngc, vgaDisplay.ngc, display.ngc et chronometer.ngc, au projet dans lequel se trouve notre module score. Puis nous générons le fichier à implémenter dans le FPGA.

Ensuite, nous implantons le projet dans le FPGA puis nous testons le système à l’aide de la carte. En cas de mauvais fonctionnement, nous aurions alors directement su que l’erreur venait du module score. En cas de bon fonctionnement, nous savions aussitôt que celui-ci était valide.

Une fois le fonctionnement global du système validé, nous avons remplacé le fichier display.ngc par les modules vhdl codés lors de la première séance. Enfin, nous avons reconstitué le bloc chronoscore avec les fichiers développés par nous-même lors de la première séance. Après validation, nous avons directement testé le système sur la carte et tout fonctionnait.

# VI)Conclusion:

## A)Retour d'expérience

Pour conclure, nous pouvons dire que ce projet nous a permis de mettre en applications les notions de cours, de manipuler des fichiers VHDL ou encore de découvrir comment programmer un système avec les ‘process’.

La majorité des problèmes rencontrés venaient de la syntaxe, dans nos fichiers VHDL même si le mapping a aussi représenté une difficulté.

## B)Opinion sur le projet

Le projet a été pour nous quelque chose d'enrichissant et si nous devons parler de notre expérience personnelle: C'est même le TP qui m'as permis de bien mieux comprendre le langage VHDL. Ce fût un plaisir.

## C)Discussion sur de potentiels problèmes rencontrés / solutionnés ou pas

* Nous avons rencontré des soucis avec counter\_3b\_E. En effet nous avions des fonctions en retard, donc un membre a pris l'initiative de continuer chez lui. Ce bloc a donc été codé chez lui. Arrivé en TP il sort son fichier (Display) et se rend compte qu'un câble n'est pas branché, il revérifie les connexions mais tout semble correct. Après des tentatives de nettoyage du projet effectué par Mme.STEFFEN. Notre binôme se rend compte que le bohavioral du bloc est fonctionnel mais pas le post-route. Finalement on se rend compte que même si la logique était fonctionnel ISE était incapable de trouver un schéma de bloc correct et envoyait notre bloc à la masse. Nous avons du recoder la fonction d'un autre manière.
* Il y a eu différentes difficultés dues au changements d'environnements
* Jean-Christophe a eu des difficulté d'installation de l'ISE sous Linux et a été contraint d'utiliser Windows.
* La machine produite semble fiable et les différents systèmes d’anti-rebond fonctionnels. Une phase plus approfondis de tests et de Benchmarks nous en dira plus sur les éventuels problèmes.

## D)Proposition d'ajout ou modification au projet pour améliorer chronoscore

Pour ce chronoscore on peut imaginer différentes améliorations. En effet, nous n'utilisons qu'une petite partie des ressources de la carte.

* On pourrait imaginer l'affectation d'un bouton qui permette d'enlever du score en cas d'erreur. Pour faire cela, il faudrait implémenter un fonction supplémentaire qui permettrait de passer nos compteurs en mode décompteur (Théorie vue en ELN2).
* On peut également imaginer d'autres modes de comptage pour d'autres sports
* On peut penser à une amélioration du module VGA aujourd'hui vieillissant, soit en ajoutant un adaptateur pour avoir un autre type de sortie. Si cela est possible, il serait même plus propre de dessouder le module VGA pour y installer un module plus récent (HDMI, mini HDMI ou DisplayPort). Mais, pas sûr que cela plaise à Digilent.
* Un contrôle à distance via un module supplémentaire pour connecter le système en réseau. Il faudra réfléchir à traduction éventuelle des paquets TCP englobés dans de l'IP (ou passer sur un microcontrôleur)
* Mais, selon nous l'amélioration la plus importante serait le passage sur un microcontrôleur (même si cela implique de faire passer le code à un plus haut niveau (langage de programmation plutôt que description matérielle). Il faudra donc l'ajout d'une mémoire morte ROM pour pouvoir exécuter et stocker le code proprement. Cette dernière permettrai au chronoscore de pouvoir s'arrêter sans perdre la configuration. Il faudra donc réfléchir à la manière dont le système démarrera et exécutera le code, au système de fichier utilisé, à l'usage éventuels d'un bios voir d'un micro noyau type UNIX/LINUX

Toutes ces interrogations nous mènerons vers la fin de la phase de création/test sur un FPGA, et donc à la mise en service de premiers prototypes avec la création d'un ASIC et donc d'un PCB sur mesure pour notre chronoscore.

Mais ça c'est une autre histoire !

# VII)Annexes:

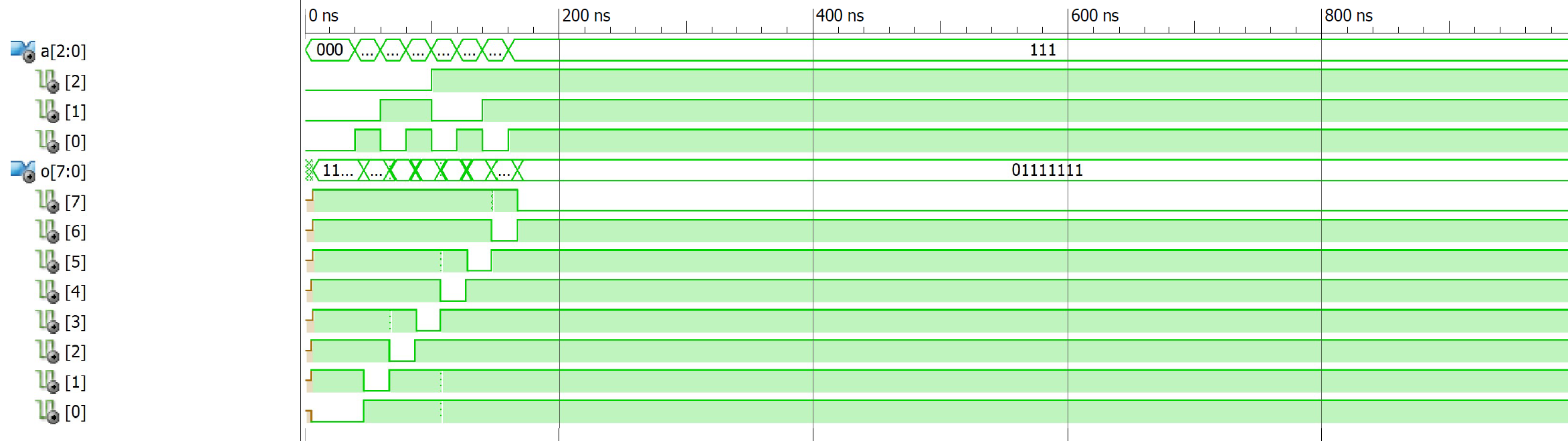
## A)Chronogrammes

### Phase1: transcoder\_3v8:

Behavioral:

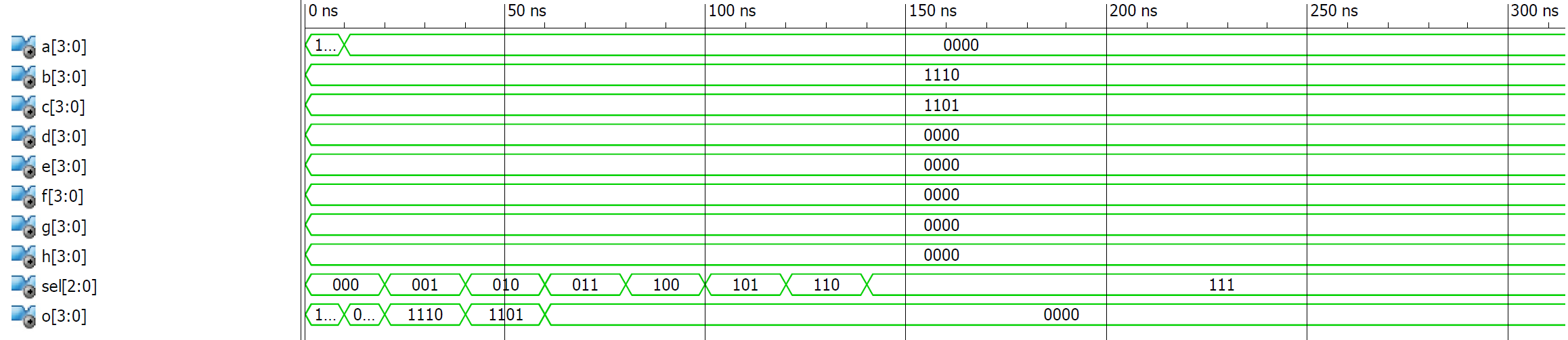


Post-Route:



### Phase1: mux\_8x1x4b:

Behavioral:



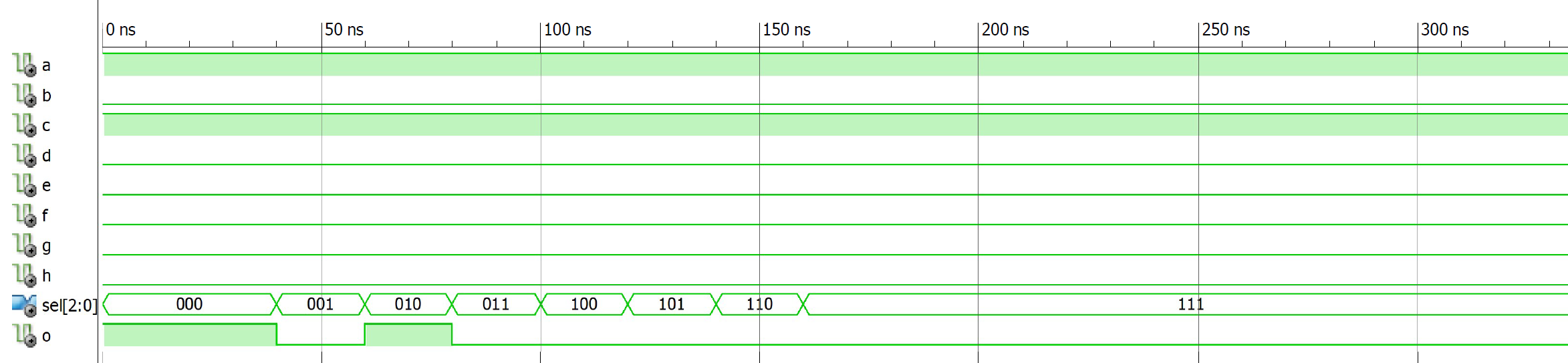
### Phase1: transcoder\_7segs:

Behavioral:

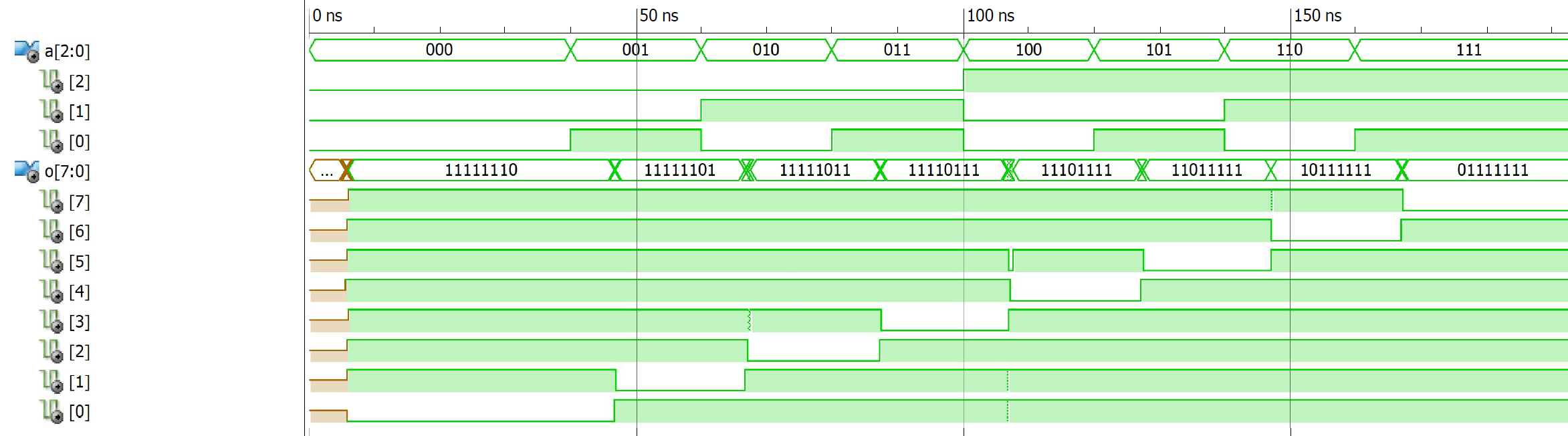


### Phase1: mux\_8x1x1b:

Behavioral:

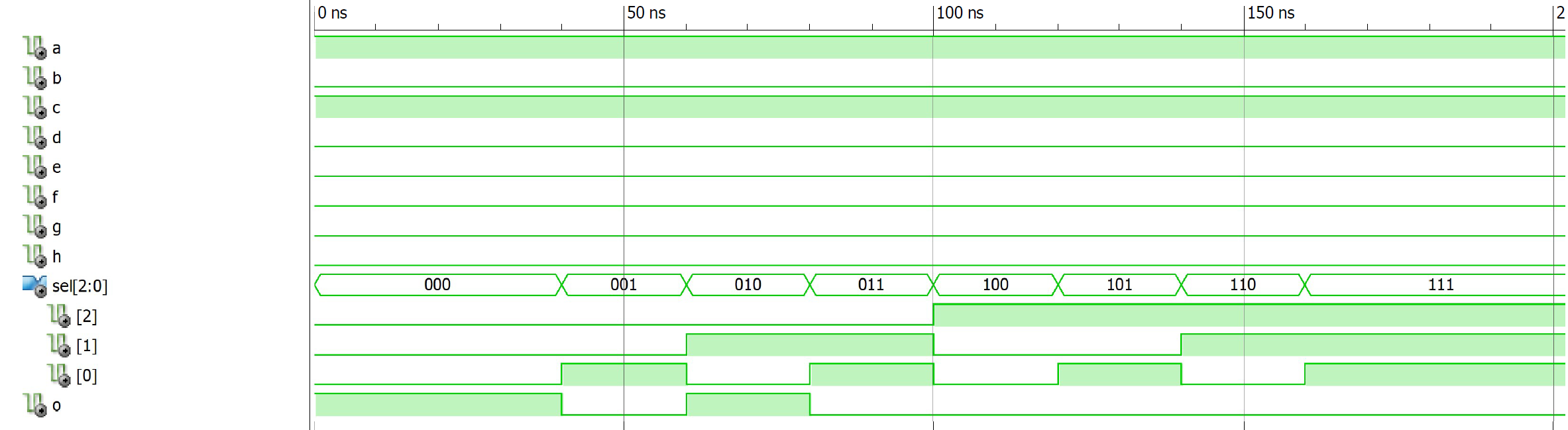


Post-Route:

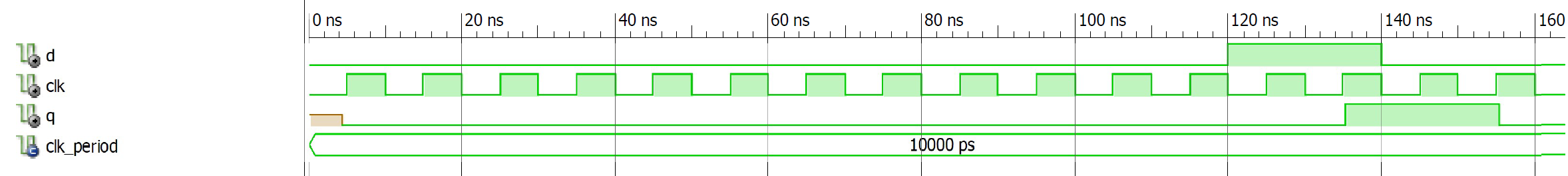


### Phase1: register\_8b:

Behavioral:



Post-Route:



### Phase1: Tregister\_1b:

Behavioral:

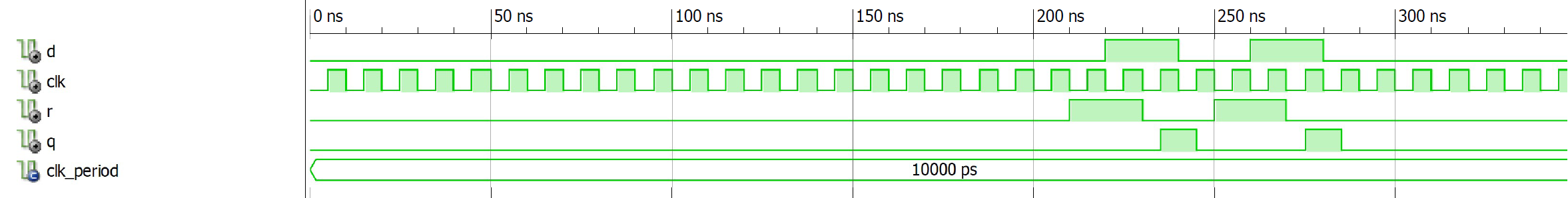
### 

### Phase1: counter\_3b\_E:

Behavioral:

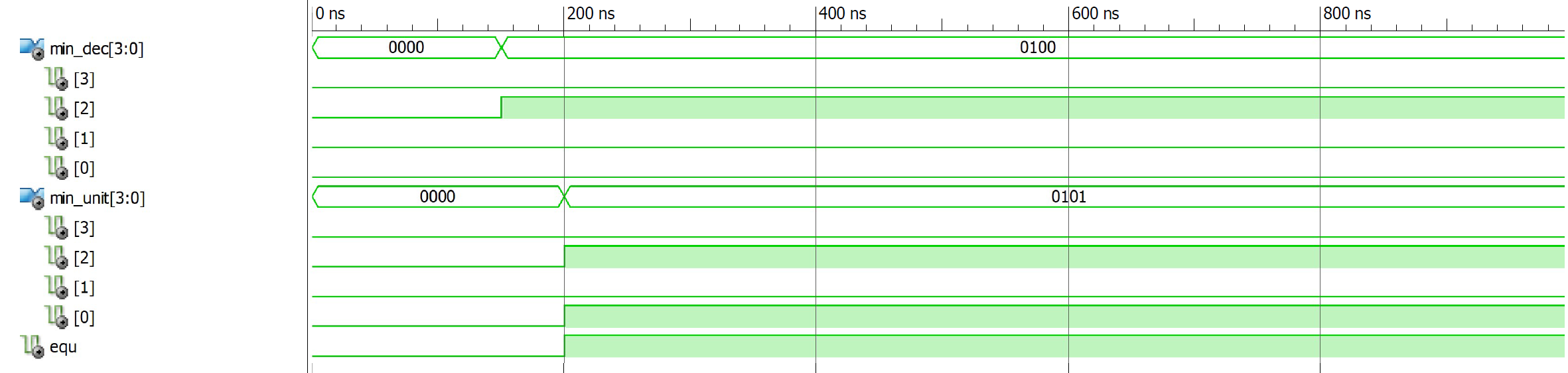
### Phase2: register\_1b\_R:

Behavioral:



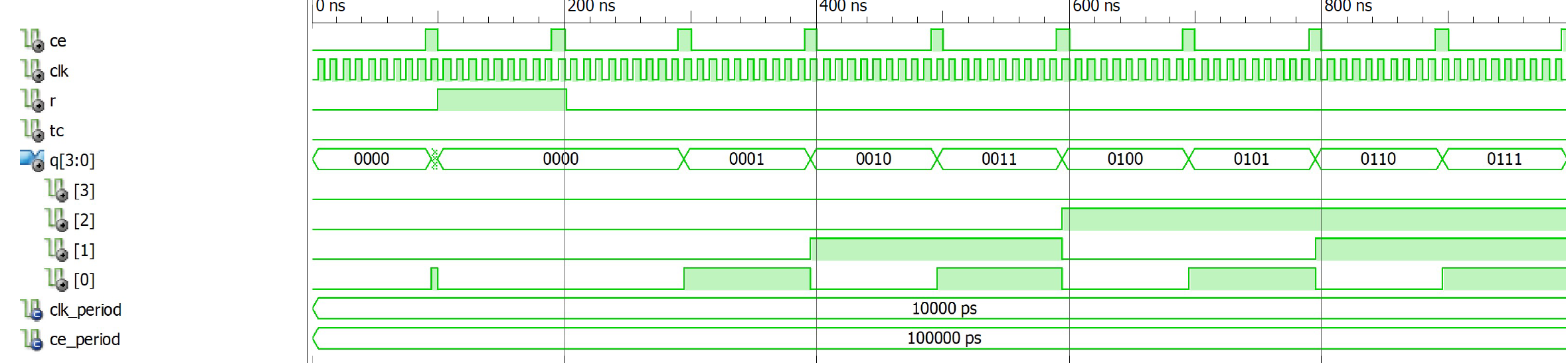
### Phase2: equ45min:

Behavioral:



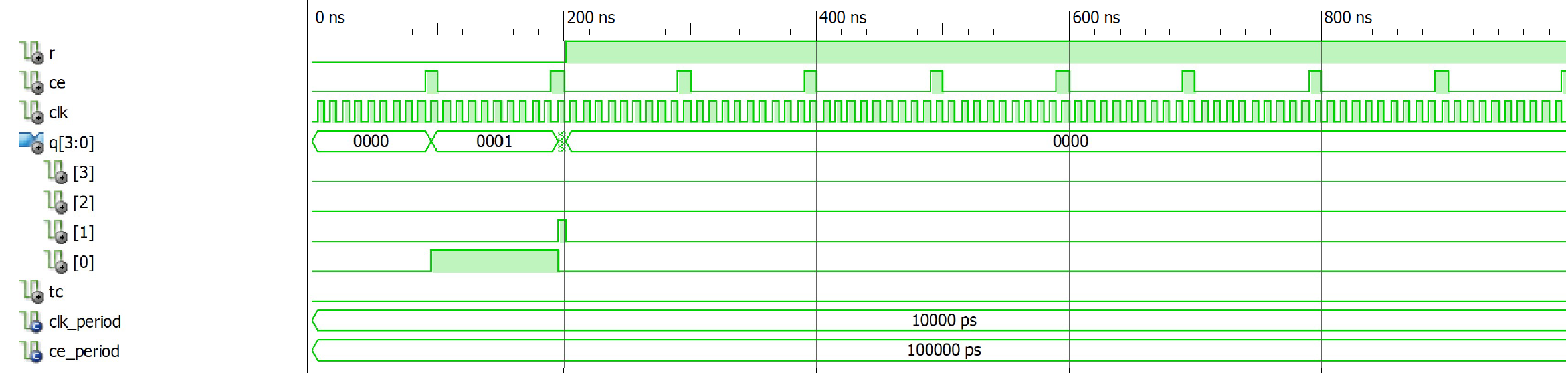
### Phase2: counterDec\_4b\_RE:

Behavioral:



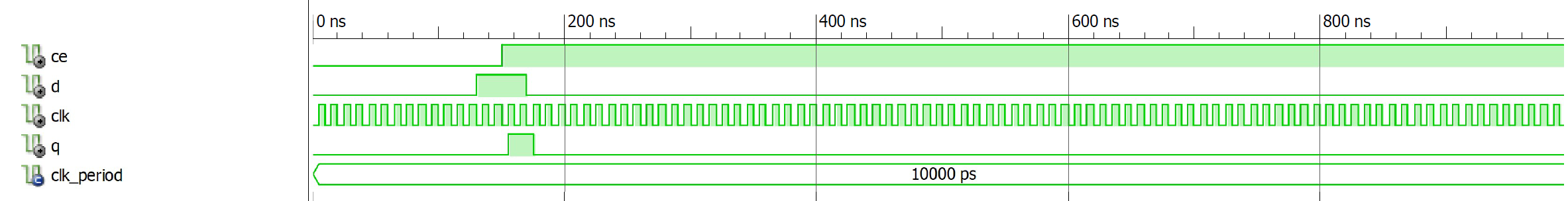
### Phase2: counterSen\_4b\_RE:

Behavioral:



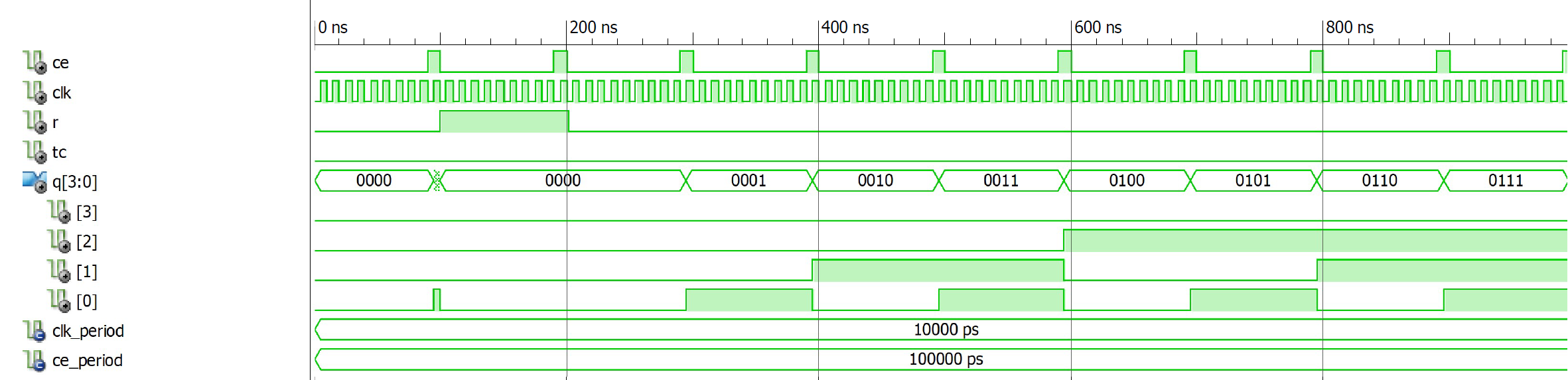
### Phase2: register\_1b\_E:

Behavioral:



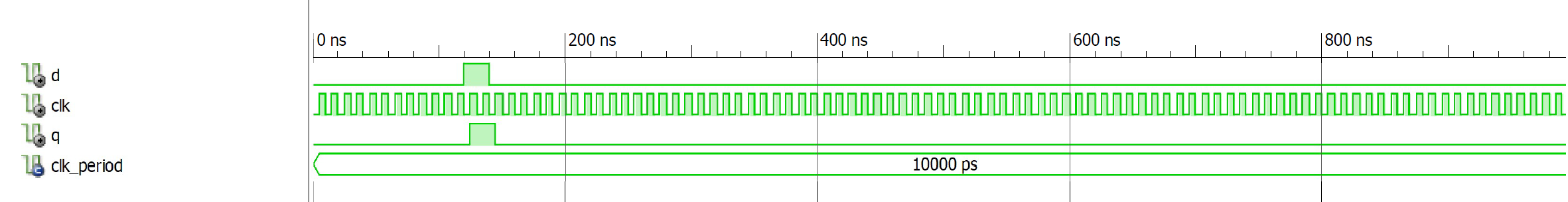
### Phase2: counterDec\_4b\_RE:

Behavioral:

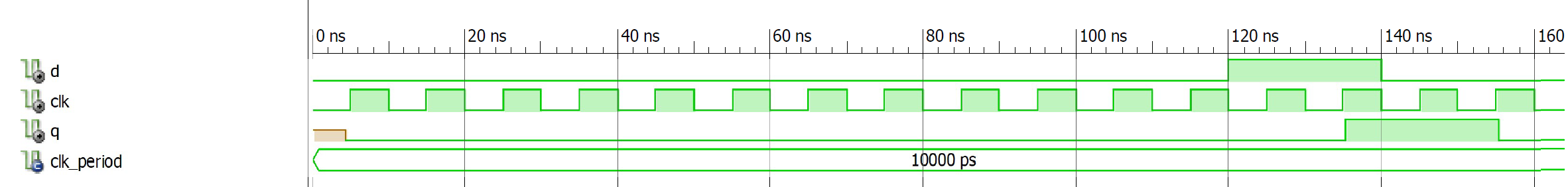


### Phase2: register\_1b:

Behavioral:

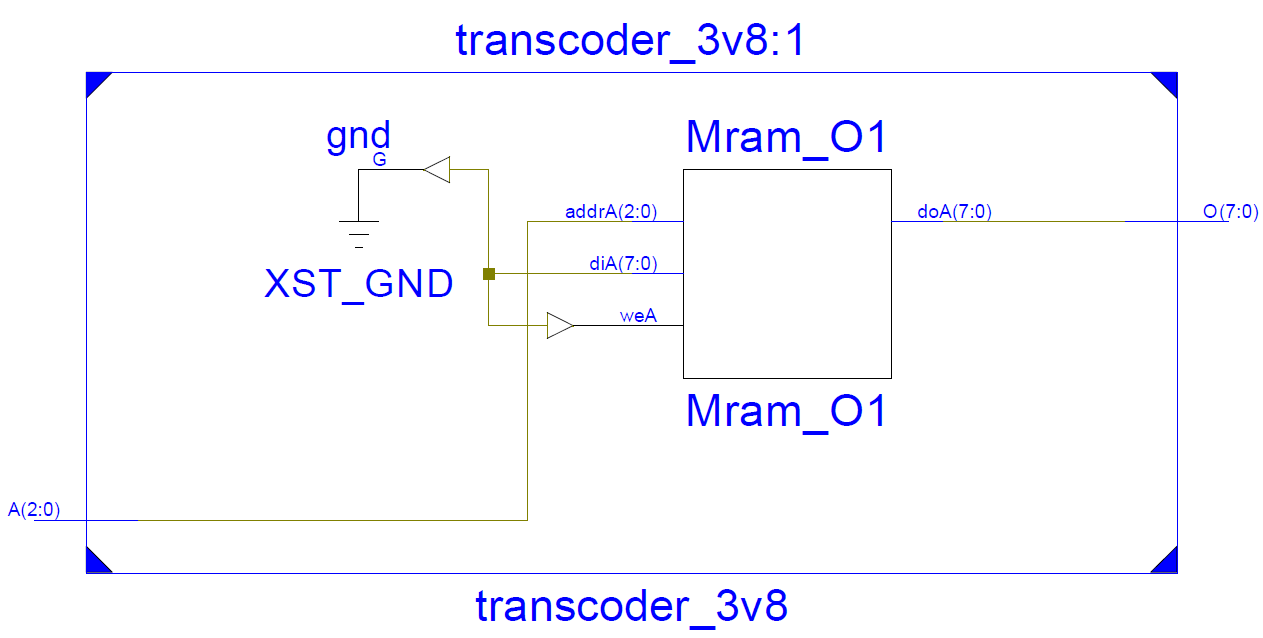


Poste-Route:

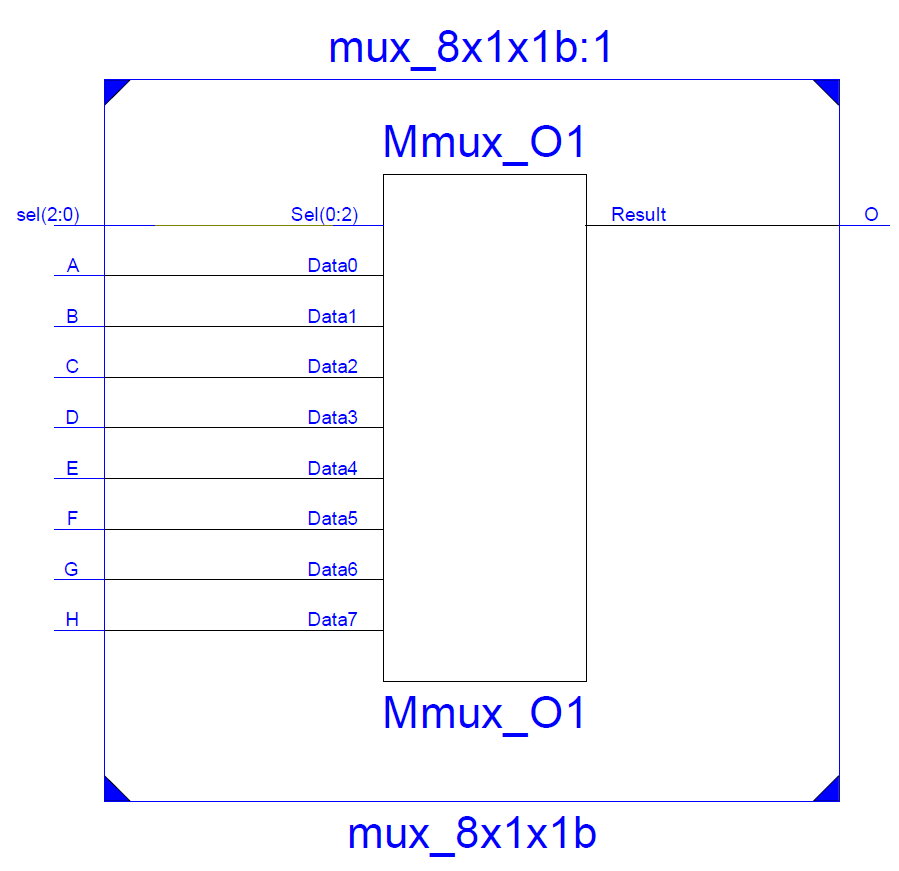


## **B)Vue RTL**

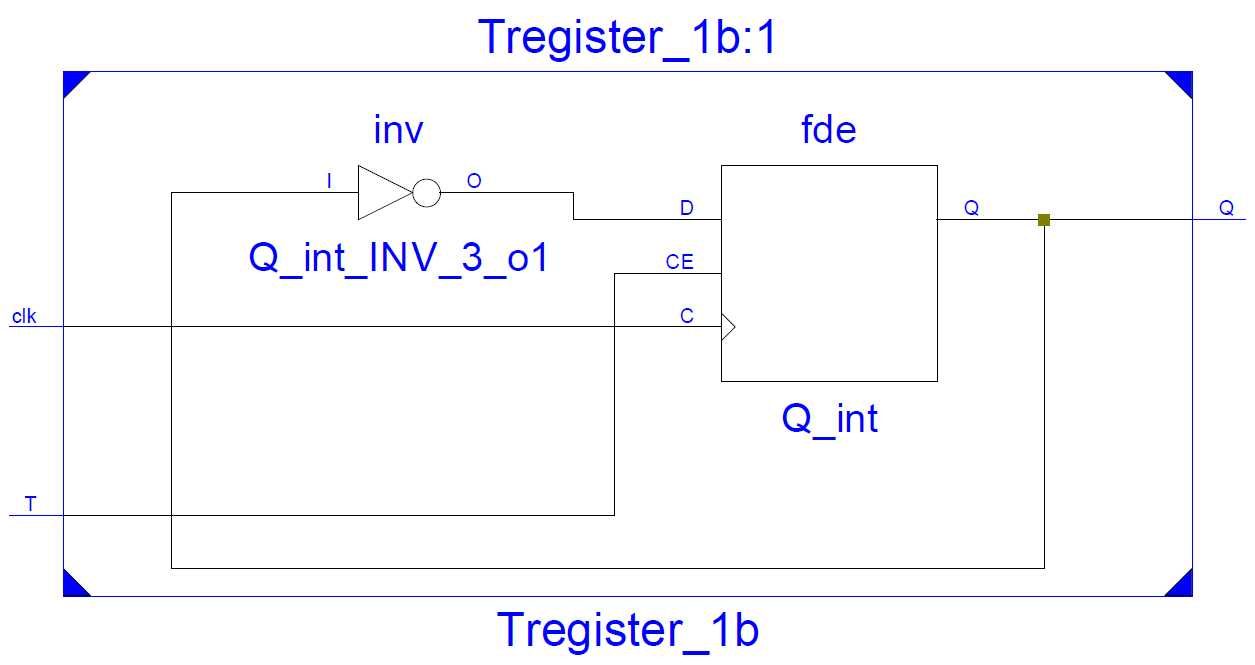
### Phase1: transcoder\_3v8:



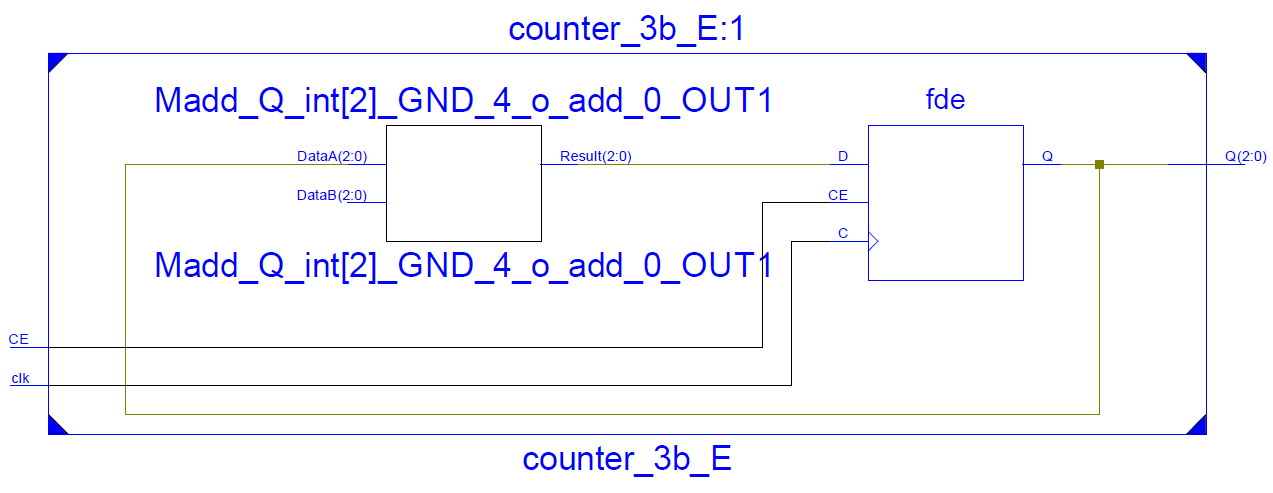
### Phase1: mux\_8x1x1b:



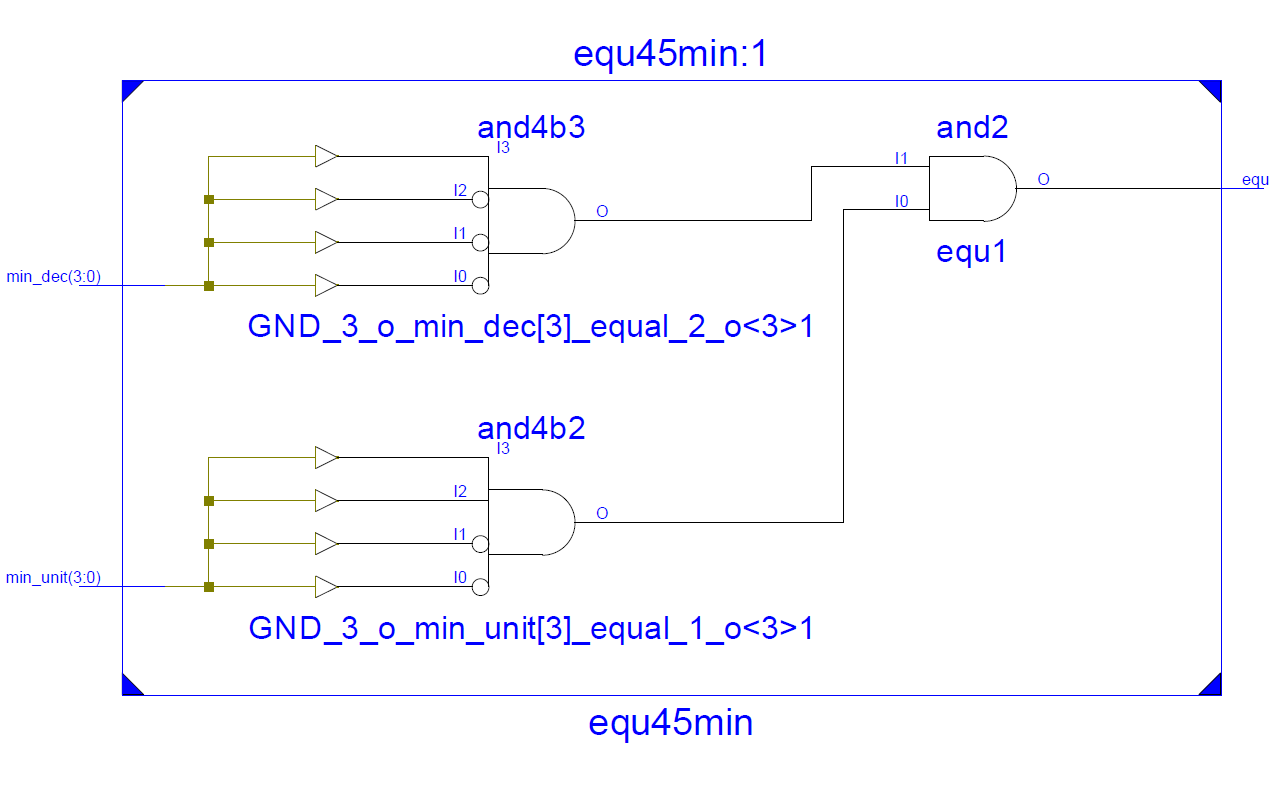
### Phase1: Tregister\_1b:



### Phase1: counter\_3b\_E:



### Phase2: equ45min:



## C)Codes VHDL

Vous pouvez retrouver l'ensemble des codes VHDL ainsi que les testbench sur le GitHub:

https://github.com/Jean-Christophe-BURNOT/Scoring-Project



## D)Afficheur 7 segments

