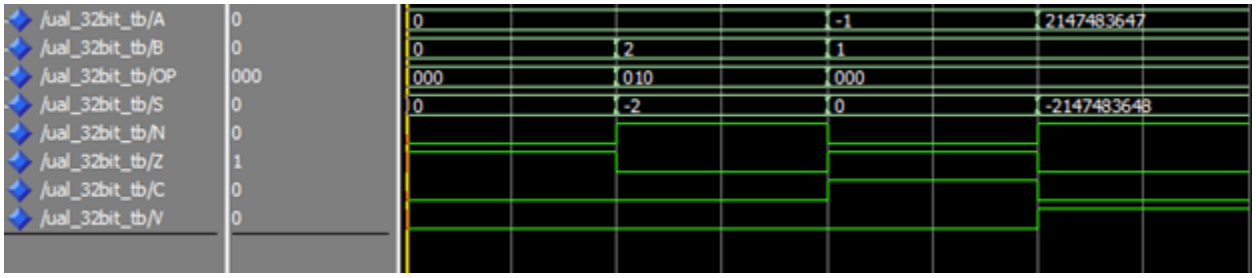
Processeur Mono-Cycle

Soukarieh Ali – 21113343

Table de Matiere

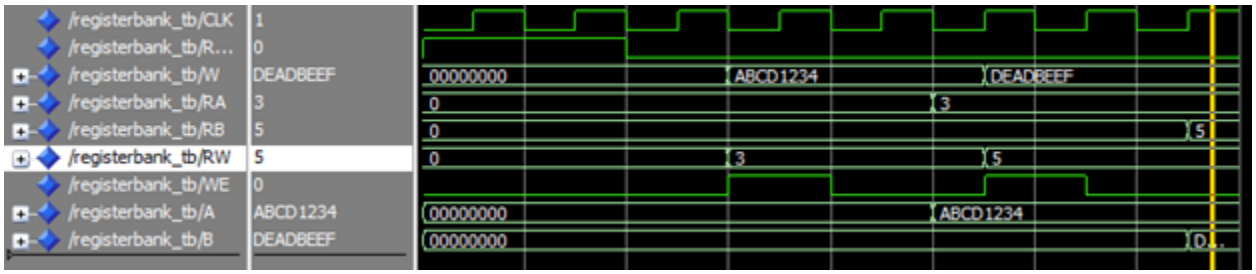
PARTIE 1 – UNITE DE TRAITEMENT

Unité Arithmétique et Logique



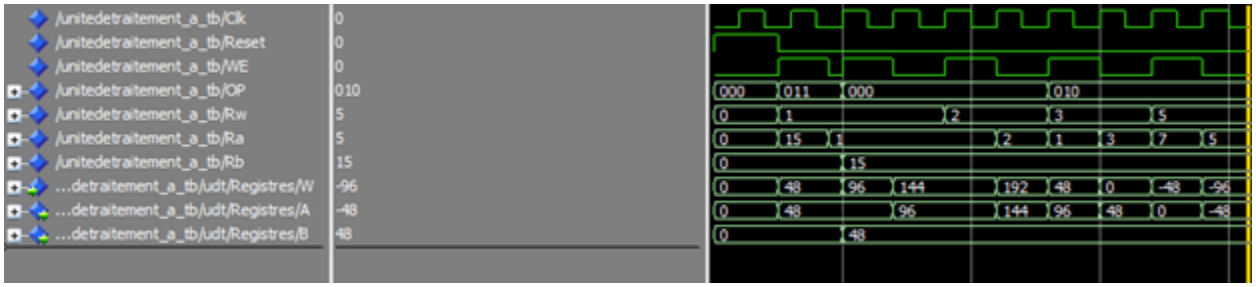
- 0 + 0 = 0 => déclenche le drapeau Z puisque la somme est 0.
- 0 – 2 = -2 => déclenche le drapeau N puisque le résultat est négatif.
- 1111 1111 1111 1111 1111 1111 1111 1111 + 1 => déclenche le drapeau C puisque l'ajout de 1 à la valeur maximale d'un nombre binaire non signé entraîne un débordement, revenant à 0.
- 0111 1111 1111 1111 1111 1111 1111 1111 + 1 => déclenche le drapeau V puisque l'ajout de 1 à la valeur positive maximale d'un nombre binaire signé le transforme en nombre le plus négatif, provoquant un débordement du complément à deux.

Banc de Registres



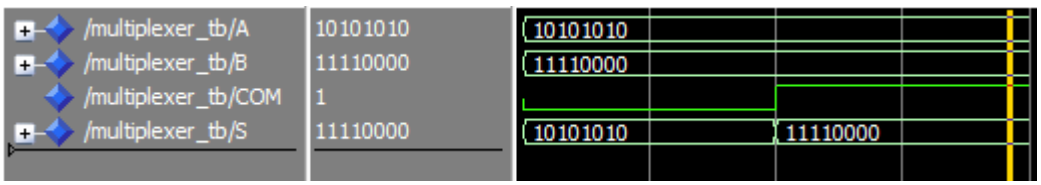
On a configuré RW, qui spécifie le registre dans lequel on écrit lorsque WE est égal à 1, sur la valeur 3 et on a entré la donnée ABCD1234. On a réglé RA et RB sur les valeurs 3 et 5 respectivement pour lire ces registres en sortie. Lorsque WE est à 0, on est en mode lecture, c'est pourquoi on peut voir ABCD1234 dans le registre 3.

Unité De Traitement – 1.1.2



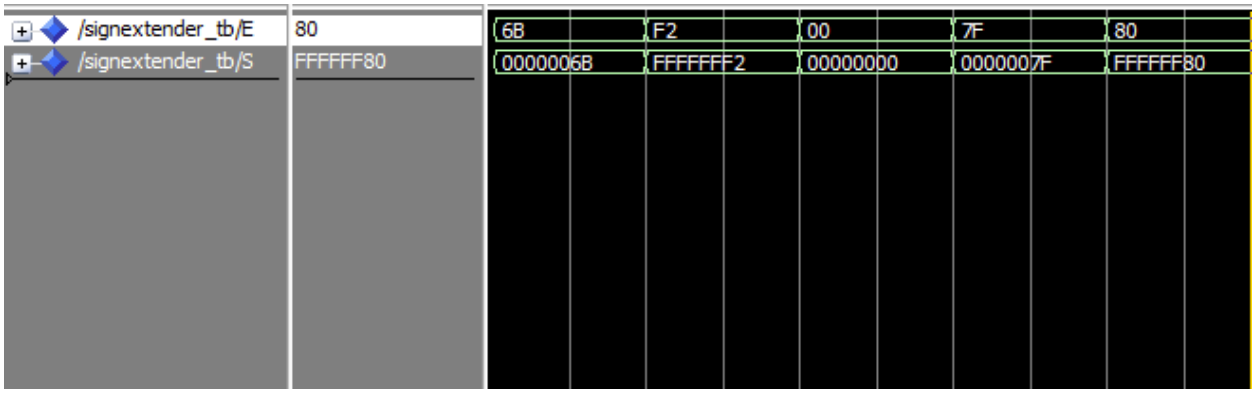
Exemple de bon fonctionnement: Au quatrième coup d'horloge, on observe que le registre A prend la valeur 96, résultant de l'addition (OP = 000) de sa valeur précédente avec celle du registre B, soit 48 + 48 = 96.

Multiplexeur 2 vers 1



S prend la valeur de A lorsque COM est à 0 et celle de B lorsque COM est à 1, ce qui correspond au comportement attendu du multiplexeur.

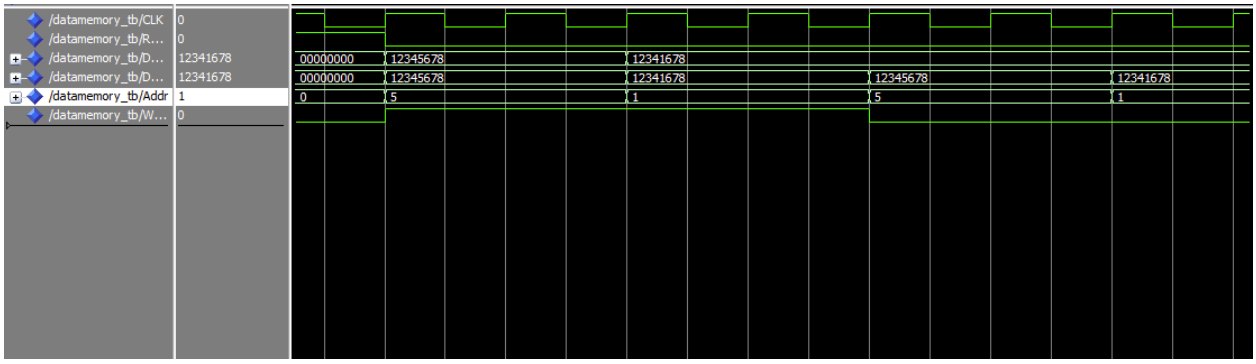
Extension de signe



Exemples de bon fonctionnement:

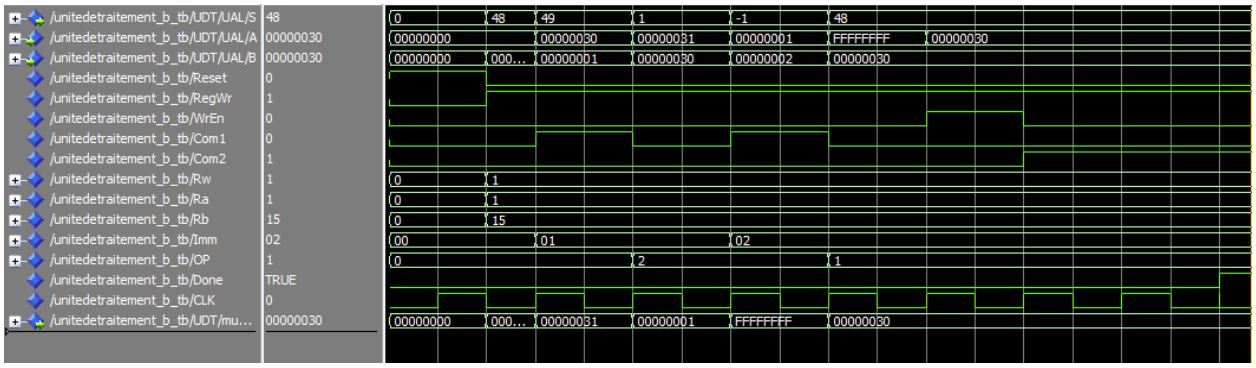
- Quand l'entrée est égale à 107 en décimal (6B en hexadécimal), la sortie est égale à 107 en décimal (0000006B en hexadécimal), ce qui signifie que l'extension de signe a réussi.
- Quand l'entrée est égale à -128 en décimal (80 en hexadécimal), la sortie est égale à -128 en décimal (FFFFFF80 en hexadécimal), ce qui signifie que l'extension de signe a réussi.

Mémoire de données



Les valeurs 12345678 et 12341678 sont écrites dans les registres 5 et 1, respectivement. Lorsque WE passe à 0 (mode lecture), ces mêmes valeurs se retrouvent dans les registres correspondants.

Unité De Traitement – 1.3.1



Exemples de bon fonctionnement :

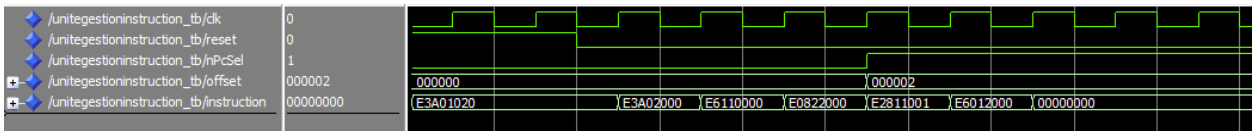
- Au deuxième coup de l'horloge dans la simulation VHDL, le signal `Imm` reste défini à "00000001", utilisé pour une opération d'addition avec le contenu du registre pointé par `Ra`. Le signal `Com1` est activé ('1'), indiquant l'utilisation de la valeur immédiate plutôt qu'une valeur de registre

secondaire. L'opération d'addition est spécifiée par le code `OP` réglé sur "000". Simultanément, `RegWr` est également activé ('1'), autorisant l'écriture du résultat de cette addition dans le registre désigné par `Rw`. Ainsi, au cours de ce cycle, la valeur immédiate est additionnée au contenu du registre `Ra`, et le résultat est enregistré dans `Rw`, conformément à la programmation prévue.

- Au sixième coup d’horloge le signal WrEn est activé ('1'), indiquant une opération d'écriture en mémoire. Pendant ce cycle, les données stockées dans le registre B sont écrites dans la mémoire à l'adresse 2. Ensuite, au septième coup d'horloge, WrEn passe à 0 et Com2 est activé ('1'), permettant ainsi de lire les données stockées à l'adresse 2 de la mémoire. Nous retrouvons la valeur '0000030', qui est exactement celle attendue.

PARTIE 2 - UNITE DE GESTION DES INSTRUCTIONS

Unité de Gestion des Instructions




```
result (0):=x"E3A01020";-- 0x0 _main -- MOV R1,#0x20 -- R1 = 0x20
result (1):=x"E3A02000";-- 0x1       -- MOV R2,#0x00 -- R2 = 0
result (2):=x"E6110000";-- 0x2 _loop -- LDR R0,0(R1) -- R0 = DATAMEM[R1]
result (3):=x"E0822000";-- 0x3       -- ADD R2,R2,R0 -- R2 = R2 + R0
result (4):=x"E2811001";-- 0x4       -- ADD R1,R1,#1 -- R1 = R1 + 1
result (5):=x"E351002A";-- 0x5     -- CMP R1,0x2A  -- Flag = R1-0x2A,si R1 <= 0x2A
result (6):=x"BAFFFFFB";-- 0x6     -- BLT loop -- PC =PC+1+(-5) si N = 1
result (7):=x"E6012000";-- 0x7     -- STR R2,0(R1) -- DATAMEM[R1] = R2
result (8):=x"EAFFFFF7";-- 0x8     -- BAL main     -- PC=PC+1+(-9)
```

Initialement, on lit les instructions séquentiellement sans aucun décalage : E3A01020, puis E3A02000, etc. Cependant, une fois que offset est réglé à 2 et nPcSel à 1, on commence à observer des sauts de décalage + 1 dans la séquence. Par exemple, la cinquième instruction lue devient E2811001, et la huitième instruction lue est E6012000.

PARTIE 3 – UNITÉ DE CONTROL

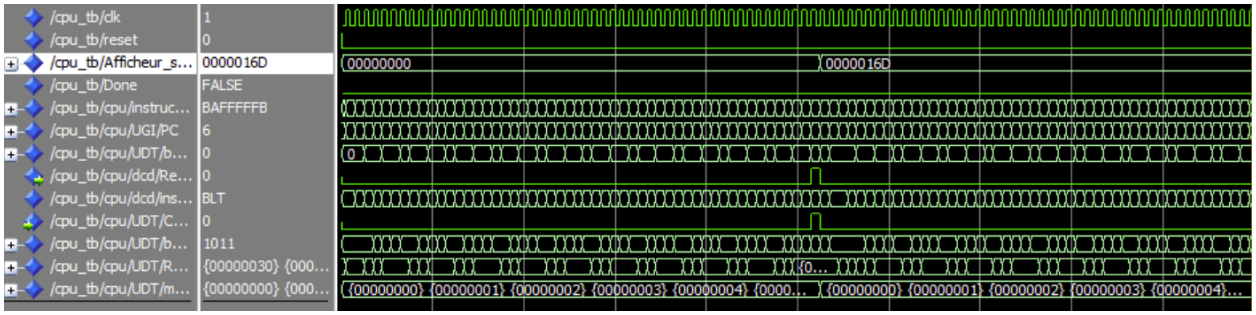
Registre 32 Bits Avec Bit de Commande de Chargement

On observe l'entrée F0F0F0F0 en sortie un coup d'horloge plus tard uniquement lorsque WE est mis à 1, ce qui correspond au comportement attendu d'un registre.

PARTIE 4 – ASSEMBLAGE ET VALIDATION

Pour l'assemblage final, j'ai créé une nouvelle version de l'unité de traitement (unite_de_traitement.vhd) en ajoutant un registre pour l'affichage et en ajoutant les drapeaux en sortie.

Pour l'initialisation de la mémoire, on affecte à chaque emplacement, de 0 à 63, sa propre adresse transformée en un vecteur logique de 32 bits.




```
0x0 _main :  MOV R1,#0x20        ; --R1 <= 0x20
0x1          MOV R2,#0           ; --R2 <= 0
0x2 _loop :  LDR R0,0(R1)        ; --R0 <= DATA_MEM[R1]
0x3          ADD R2,R2,R0        ; --R2 <= R2 + R0
0x4          ADD R1,R1,#1        ; --R1 <= R1 + 1
0x5          CMP R1,0x2A         ; -- R1 - 0x2A, mise à jour de N
0x6          BLT loop            ; --branchement à _loop si R1 inferieur a 0x1A
0x7  _end  : STR R2,0(R1)        ; --DATA_MEM[R1] <= R2
0x8          BAL main            ; --branchement à _main
```

Dans ce code, on entre dans une boucle où la valeur de R2 est mise à jour à chaque cycle d'horloge en ajoutant la valeur de R1. Cette opération est répétée 10 fois. Ensuite, lorsque R1 atteint la valeur 0x2A (ce qui correspond à 42 en décimal, soit le registre numéro 42 de la mémoire), on passe à l'instruction STR. Cette

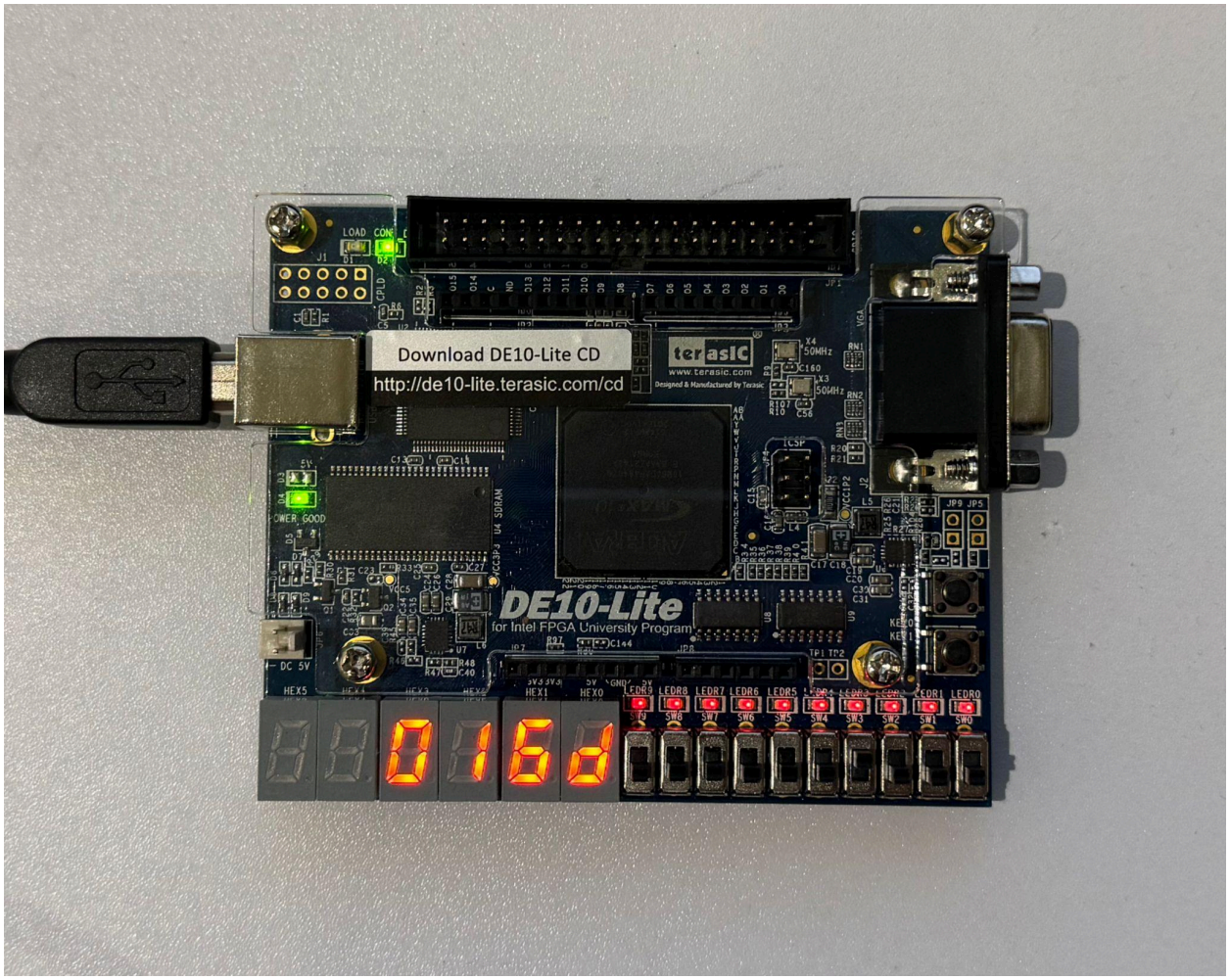
instruction transfère le résultat accumulé dans R2 vers le registre 42 et met RegAff a 1 qui permet d'afficher cette valeur. Le calcul de la récursion pour déterminer le contenu final de R2 avant le transfert utilise la formule suivante :

$$r2(0) = 0$$

$$r(n) = r(n-1) + 32 + n$$

À la neuvième itération, la valeur de R2 est donc 333 (représentée en hexadécimal par 16D). Ce qui est la valeur affichée.

PARTIE 5 – TEST DU PROCESSEUR SUR CARTE FPGA



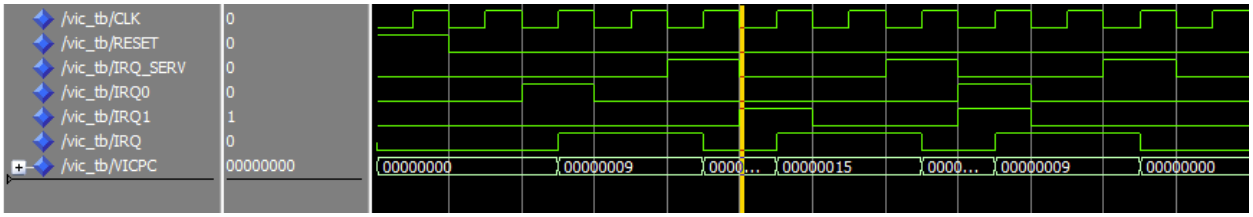
Total logic elements 3,006 / 49,760 (6 %)

On observe bien la valeur attendue, conforme à celle obtenue lors des simulations.

Le fichier bitstream (cpu.sof) se trouve dans un dossier appelé fichiers_bitstream.

PARTIE 6 – Gestion des Interruptions Externes

VIC : Contrôleur d'interruption vectorisé



Lorsque IRQ0 est à 1, VICPC prend la valeur 0x9 au prochain coup d'horloge. Lorsque IRQ1 est à 1, VICPC prend la valeur 0x15 au prochain coup d'horloge. Lorsque IRQ0 et IRQ1 sont tous deux à 1, VICPC prend la valeur 0x9, montrant la priorité de IRQ0 sur IRQ1. Tant qu'une interruption est en cours, IRQ reste à 0.

Modification du décodeur d’instruction

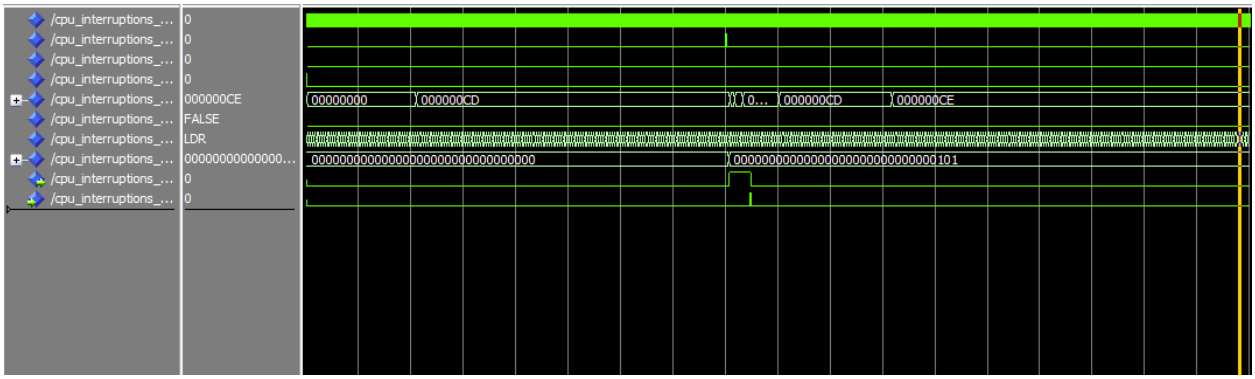
J'ai ajouté IRQ_END en tant que sortie pour mon décodeur d'instructions, et inclus un cas qui gère lorsque l'instruction est BX. Si l'instruction est BX, IRQ_END prend la valeur 1.

Modification sur l’unité de gestion des instructions

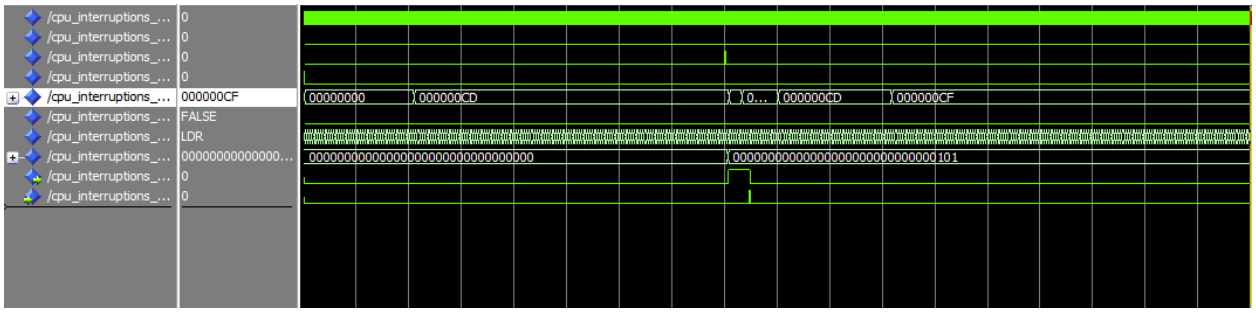
Dans la conception de "UniteGestionInstructionsInterruptions", j'ai ajouté les signaux d'entrée `IRQ`, `VICPC`, `IRQ_END`, et les signaux

de sortie IRQ_SERV. J'ai modifié le processus pour gérer efficacement les interruptions. Lorsqu'une interruption est détectée (IRQ = 1), le programme copie la valeur de VICPC dans PC tout en sauvegardant simultanément la valeur actuelle de PC dans un registre nommé LR (Link Register). Si l'interruption est traitée, le signal IRQ_SERV est mis à ‘1’; il reste à ‘0’ dans tous les autres cas. Lorsque IRQ_END est actif, PC est réinitialisé à la valeur de LR plus un, permettant ainsi au programme principal de reprendre son exécution.

Simulation



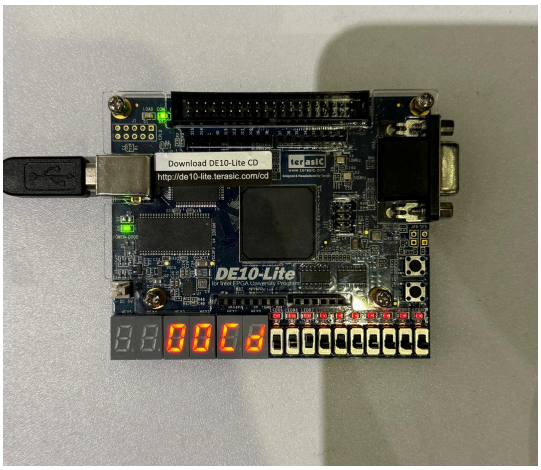
Pendant la simulation, placer IRQ0 sur 1 entraîne une augmentation de 1 à la sortie, ce qui est cohérent avec le code gérant l'interruption déclenchée par IRQ0.



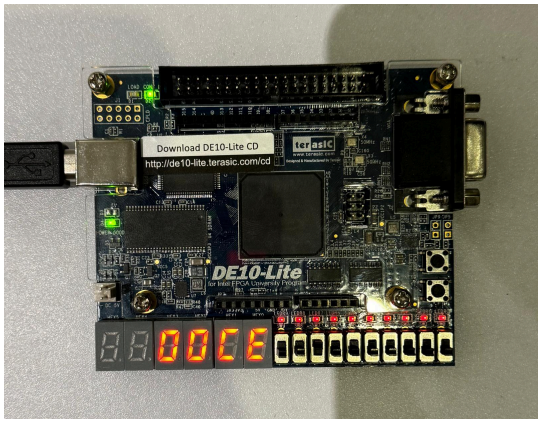
Pendant la simulation, placer IRQ1 sur 1 entraîne une augmentation de 2 à la sortie, ce qui est cohérent avec le code gérant l'interruption déclenchée par IRQ1.

Test du VIC sur FPGA

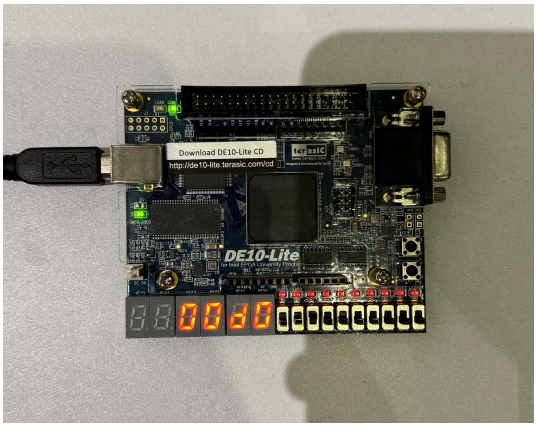
Total logic elements 3,848 / 49,760 (8 %)
Etat Initial:



Apres IRQ0:



Apres IRQ1:



On peut voir que lorsqu'on appuie sur le bouton IRQ0, 1 est ajouté à l'état initial, puis 2 supplémentaires sont ajoutés à cette dernière somme lorsque l'on clique sur le bouton IRQ1. C'est le comportement attendu.

Le fichier bitstream (cpu_interruptions.sof) se trouve dans un dossier appelé.

fichiers_bitstream.