Chen Pascal

Moutou Killian

Architecture des processeurs

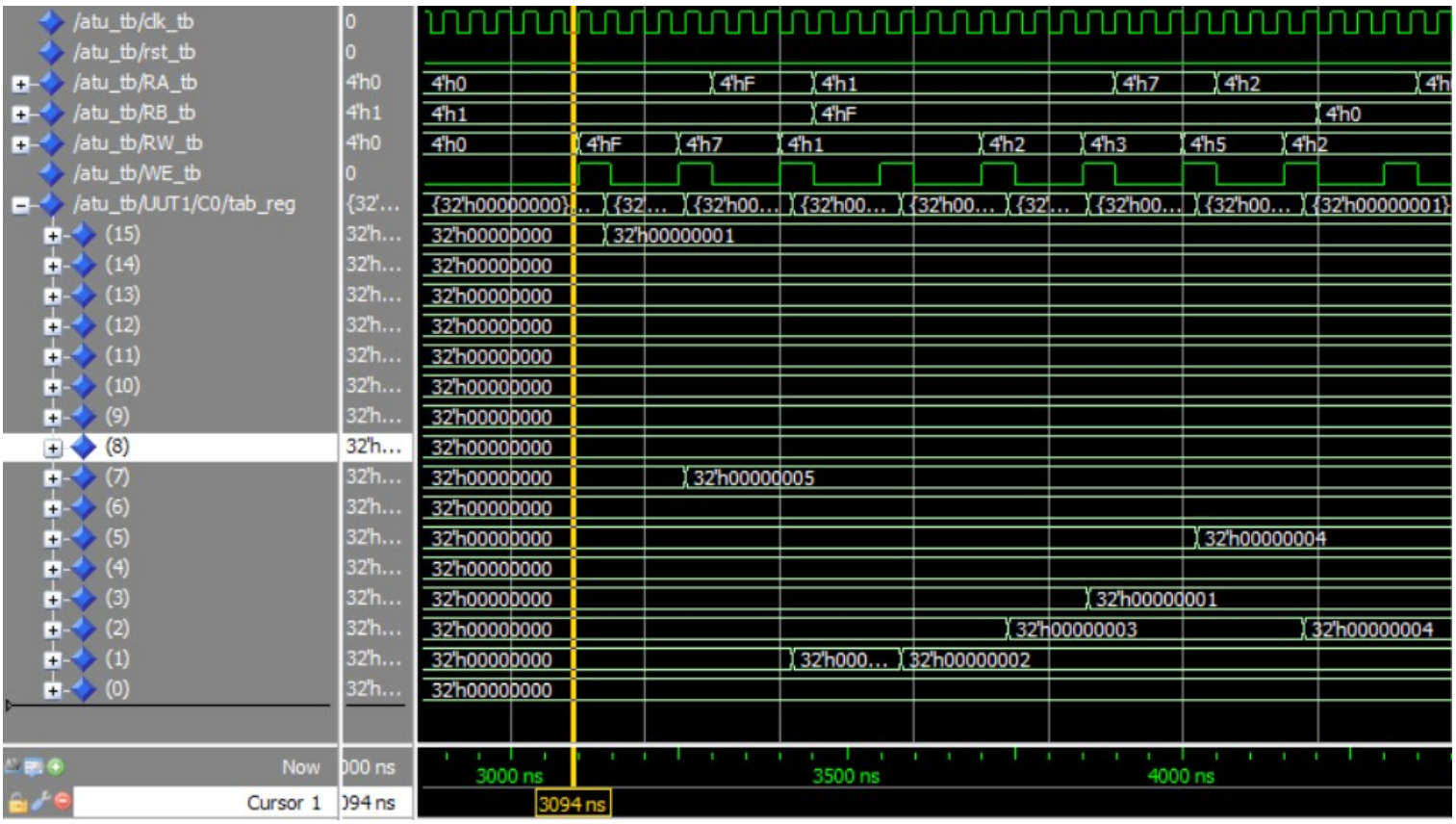
Processeur Monocycle

Simulation en VHDL

Test de L'Unité de Logique (ALU)

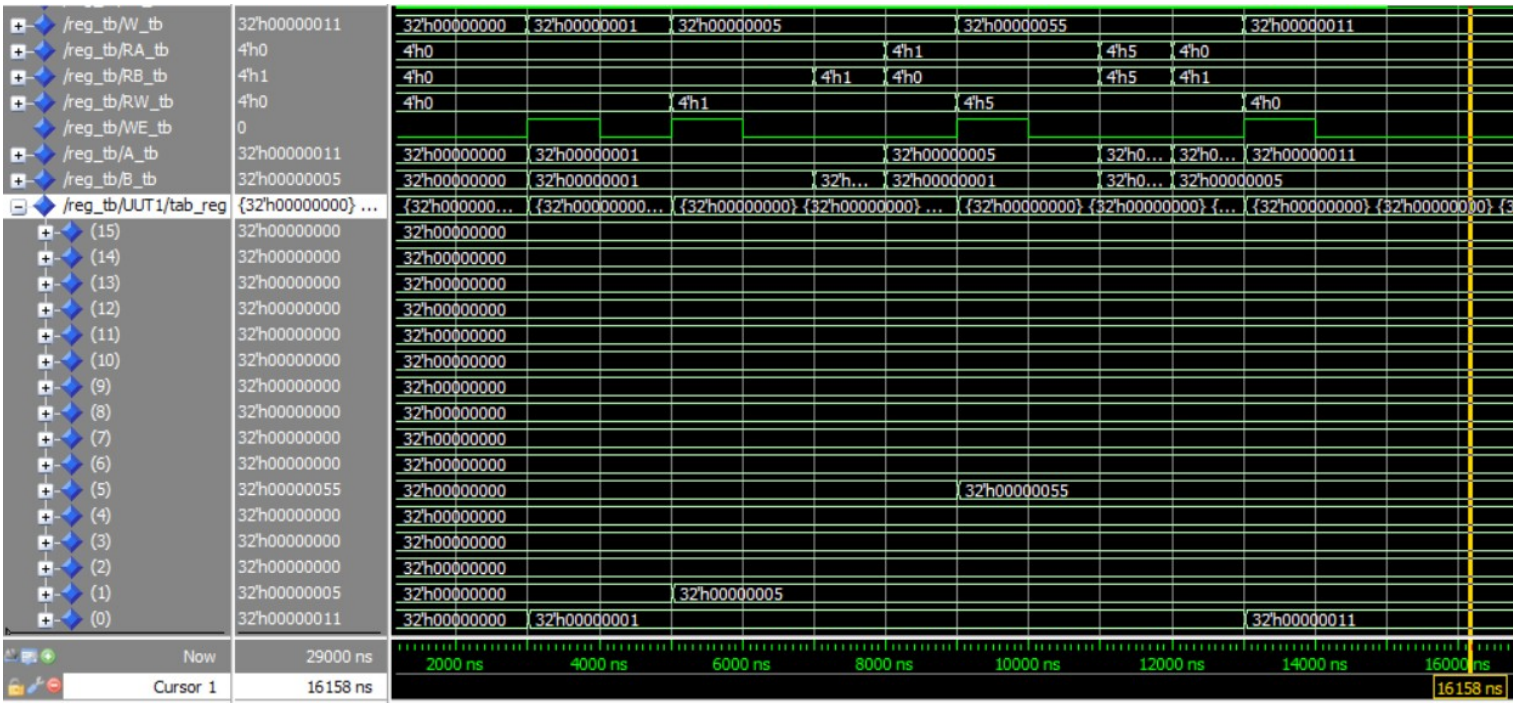
Nous exécutons les commandes suivantes en ayant préalablement initialisé les registres R(1) et R(7) aux valeurs 1 et 5 respectivement :

R(1) = R(15)
R(1) = R(1) + R(15)
R(2) = R(1) + R(15)
R(3) = R(1) – R(15)
R(5) = R(7) – R(15)

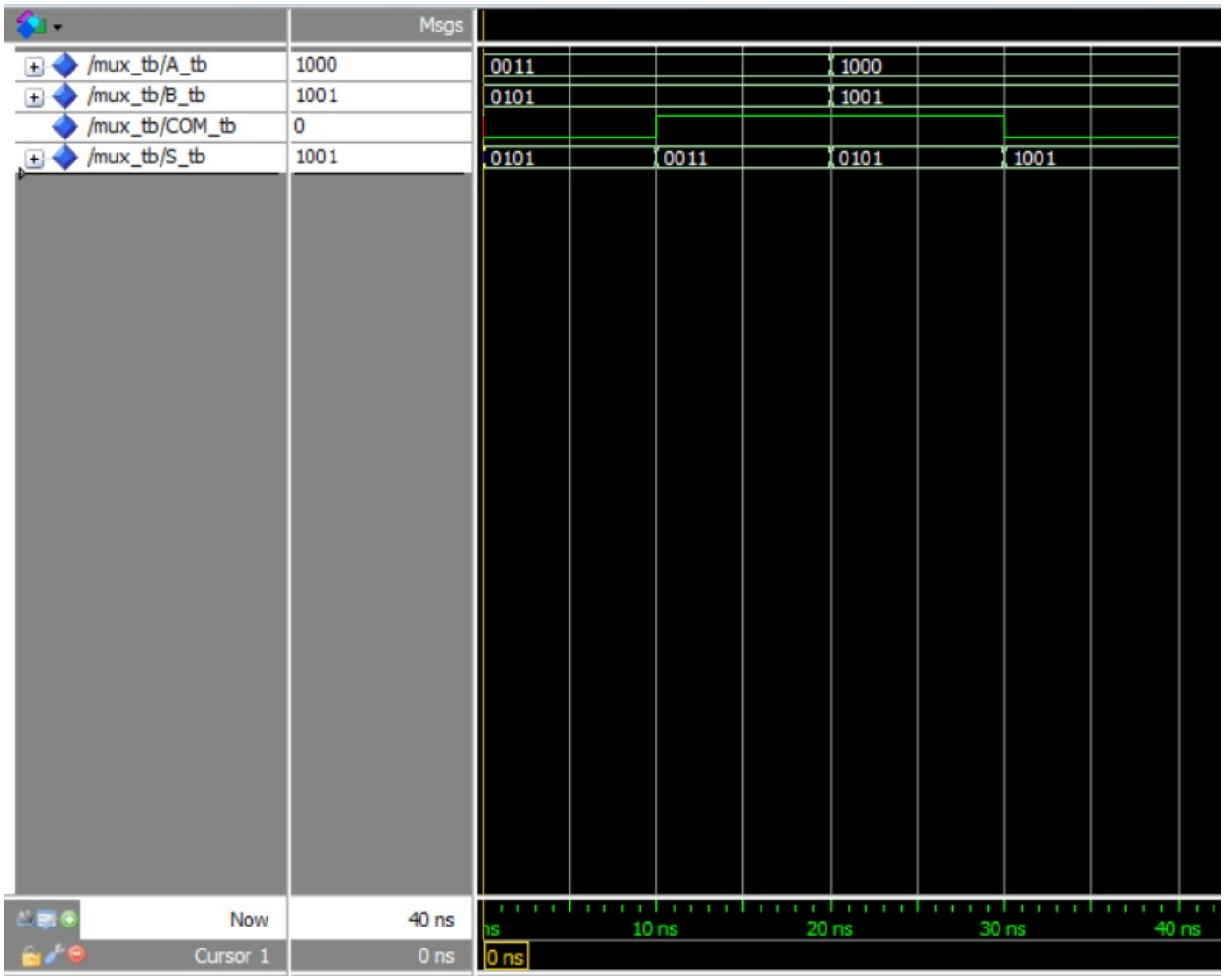


Test de du banc de Registre (REG)

Afin de tester le banc de registre, on vient simplement écrire dans les registre R(0), R(1), et R(5) les valeurs respectives 1, 5, et 55. Puis on remplace la valeur du registre R(1) par 11.

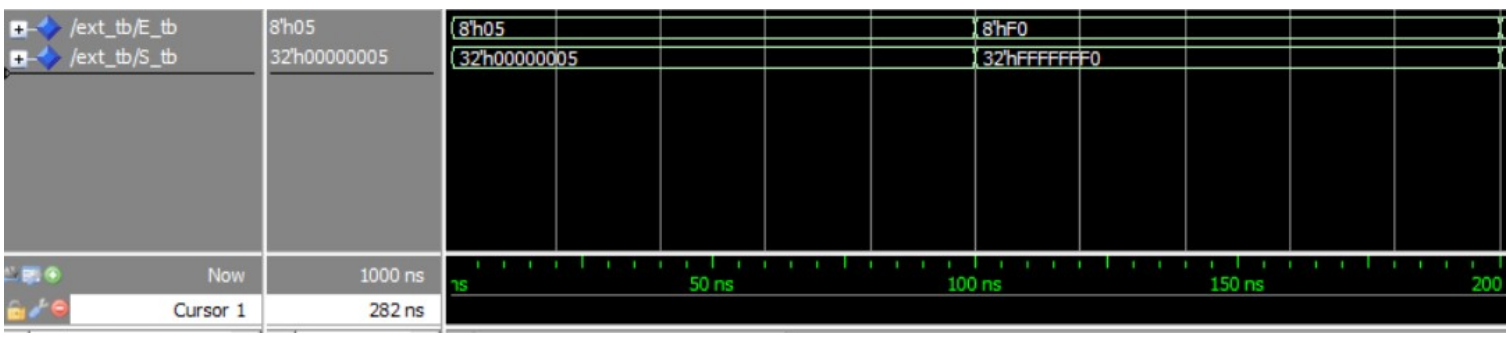


Test du Multiplexeur (MUX)



Test de l’Extension de signe (EXT)

La valeur en entrée sur 8 bits est bien transposée sur 32 bits en sortie.



Test de l'Unité de Traitement (ATU)

Pour tester l'assemblage de l'unité des traitement, on procède au test successif de plusieurs opérations comme décrit ci-dessous en ayant préalablement initialisé les registres R(0) et R(1) avec les valeurs respectives 1 et 3 :

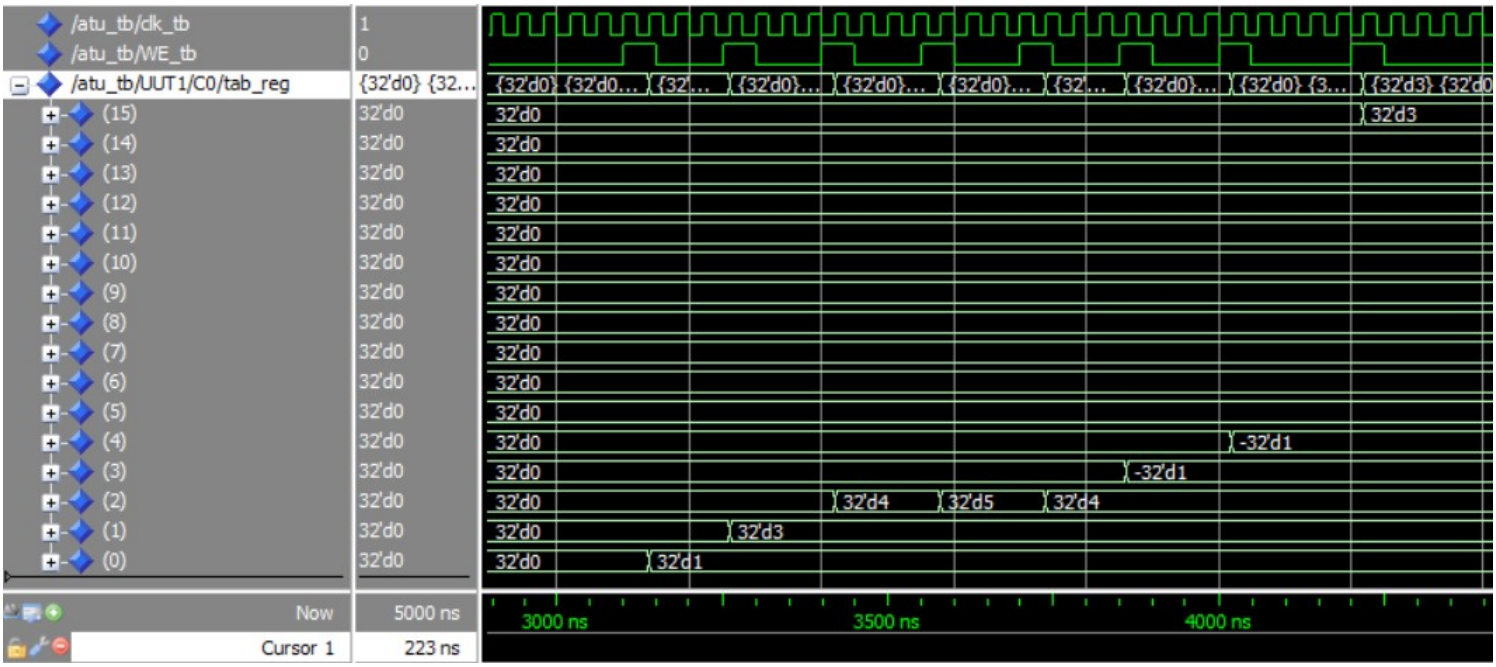
R(2) = R(0) + R(1) = 4

R(2) = R(2) + 1 = 5

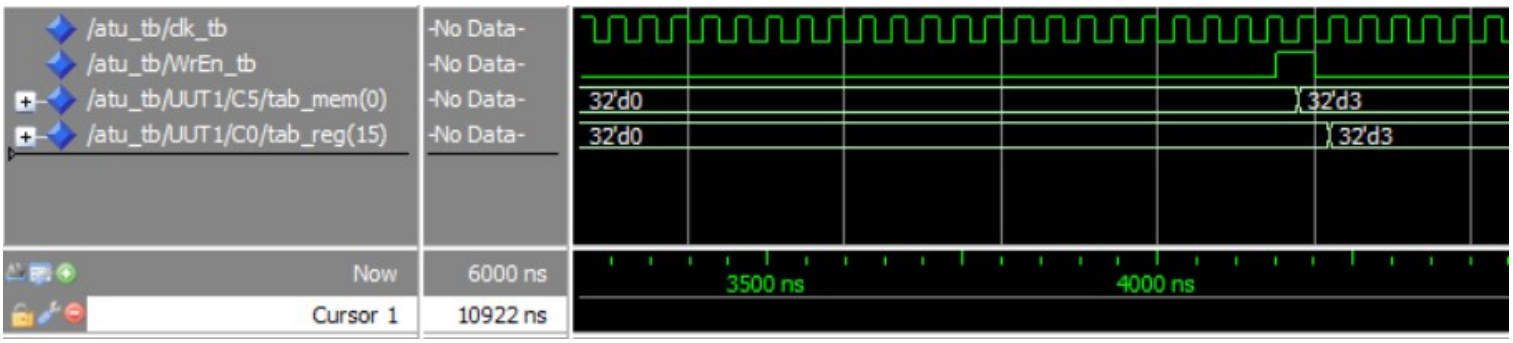
R(2) = R(2) – R(0) = 4

R(3) = R(0) – 2 = -1

R(4) = R(3) = -1

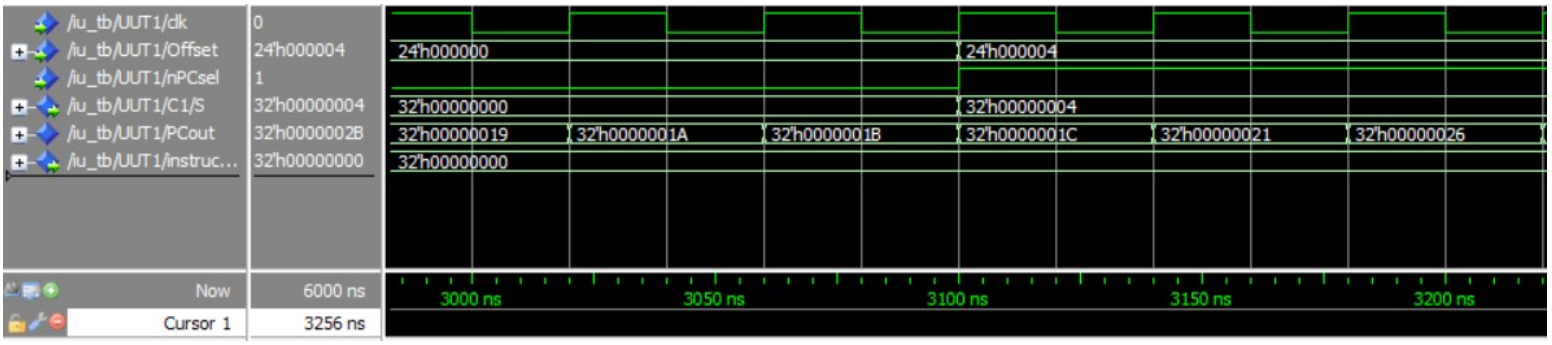


On effectue ensuite une écriture dans la mémoire de données de R(1) = 3 à l'adresse 0. Puis on lit cette donnée qui vient d'être écrite et on la stocke dans R(15).



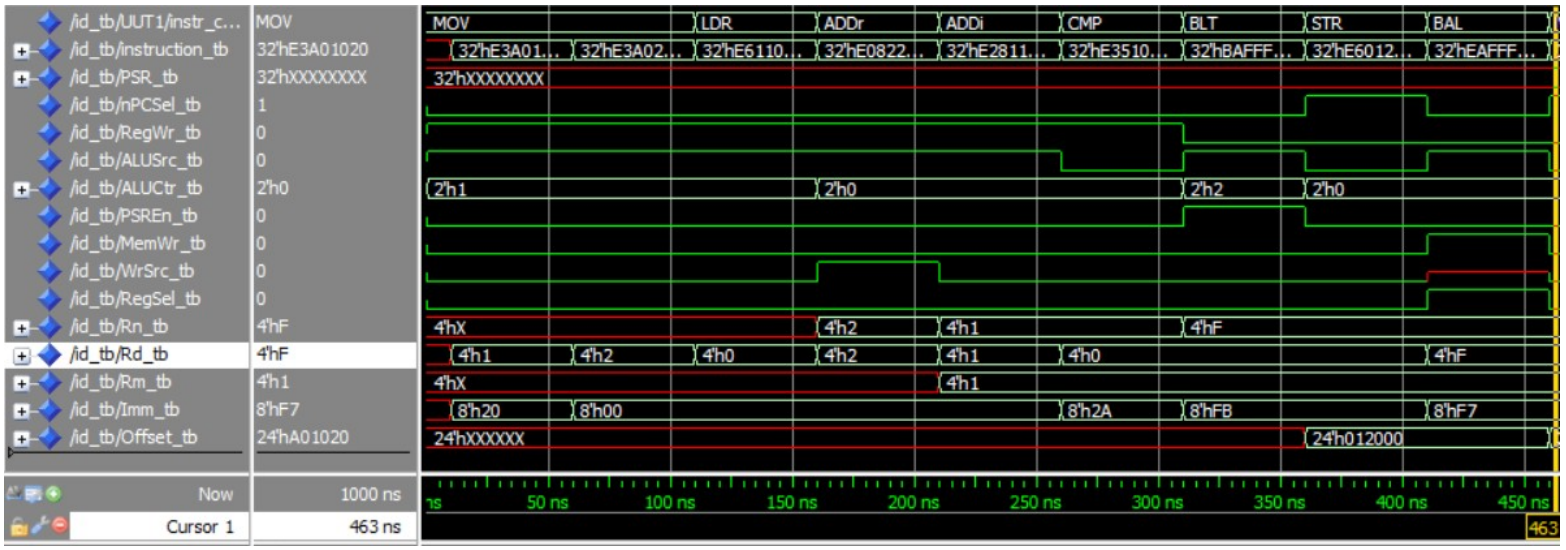
Test de l’Unité de Gestion des Instruction (IU)

Ici on vient vérifier que le pointeur d’instruction en entrée de la mémoire (PCout) d’instruction s’incrémente bien à chaque coup d’horloge. Puis on applique un offset de 4 en entrée :



Test du Décodeur d’Instruction (ID)

On vérifie bien que le décodeur reconnaît la trame de chaque instruction grâce au premier signal qui affiche le nom de l’instruction se trouvant actuellement en entrée (instruction_tb):



Les instructions testées sont les instructions du programme test du main.

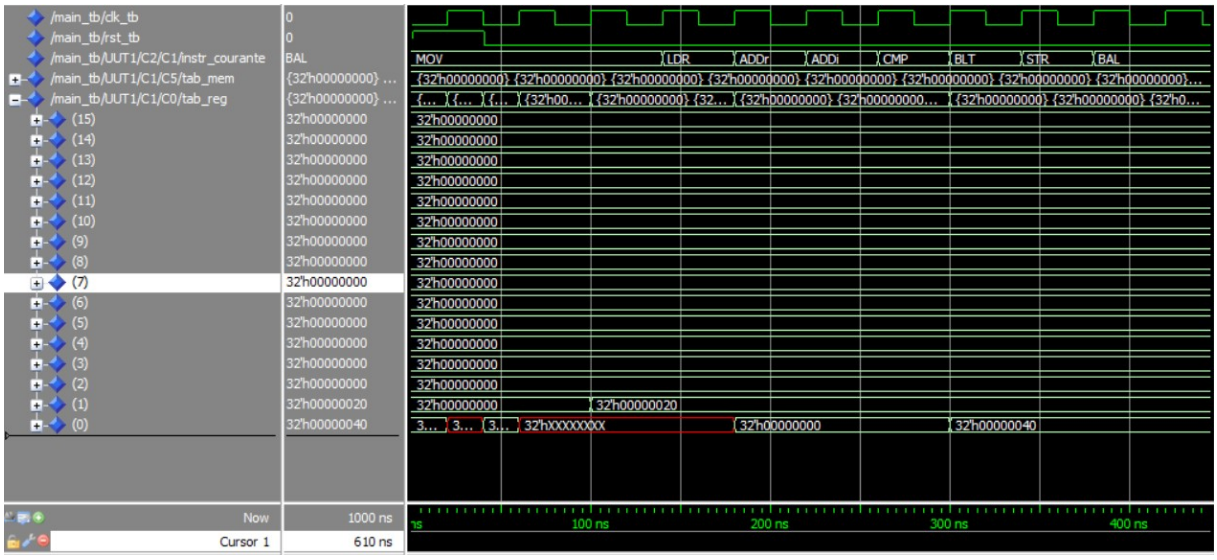
Simulation du Processeur

On a maintenant modifié la mémoire d'instruction pour qu'elle contienne le code binaire des instructions suivantes :

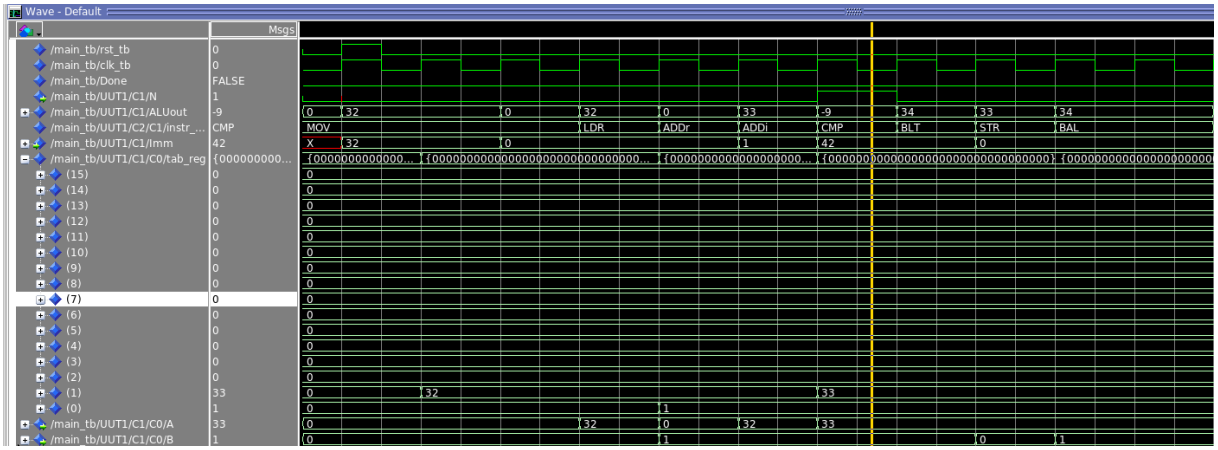
```
0x0   _main :    MOV R1,#0x20
0x1              MOV R2,#0
0x2   _loop :    LDR R0,0(R1)
0x3              ADD R2,R2,R0
0x4              ADD R1,R1,#1
0x5              CMP R1,0x2A
0x6              BLT loop
R1 inferieur a 0x2A
      _end  :
0x7              STR R2,0(R1)
0x8              BAL main
```

On a aussi initialisé la mémoire de données entre les adresses 0x20 et 0x2A comme suit :

```
result (32):=x"00000001"; -- 0x20
result (33):=x"00000002";
result (34):=x"00000003";
result (35):=x"00000004";
result (36):=x"00000005";
result (37):=x"00000006";
result (38):=x"00000007";
result (39):=x"00000008";
result (40):=x"00000009";
result (41):=x"0000000A";
result (42):=x"0000000B"; -- 0x2A
```



On constate que les instructions sont bien lues mais que la boucle ne s'effectue pas au niveau de l'instruction BLT. Le problème vient d'une erreur dans l'entité Décodeur (ID) ou du fait que l'exécution des instructions est en retard par rapport à leur lecture.



On voit que maintenant après modification du code, cela fonctionne, mem[R1] = 0, donc l'addition n'a aucun effet, l'addition en immédiat a été réalisé avec succès et comme R1 < 42 (0x2A), alors on quitte la boucle. Et on a plus ce décalage, donc tout ce qui a été fait précédemment permet de valider.