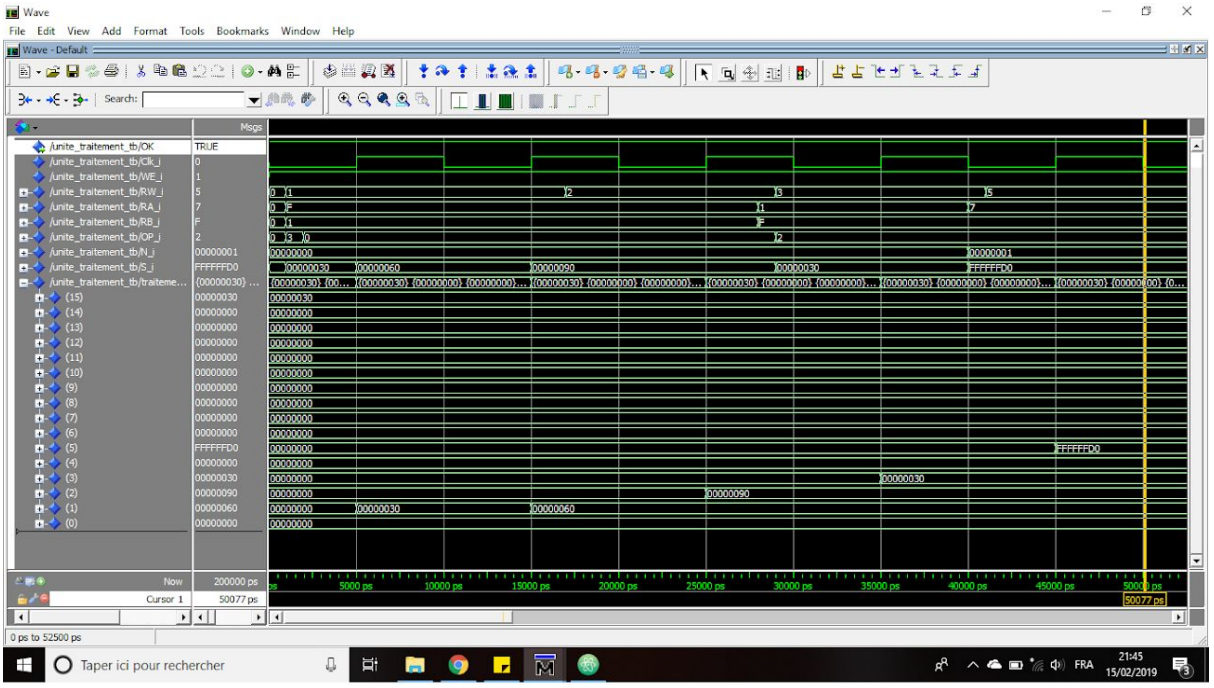
Ludovic Duplessis
Thomas Bouix

POLYTECH SORBONNE

Rapport de simulation du Processeur Monocycle ARM7 TDMI

Test bench de l'Unité de Traitement Simple

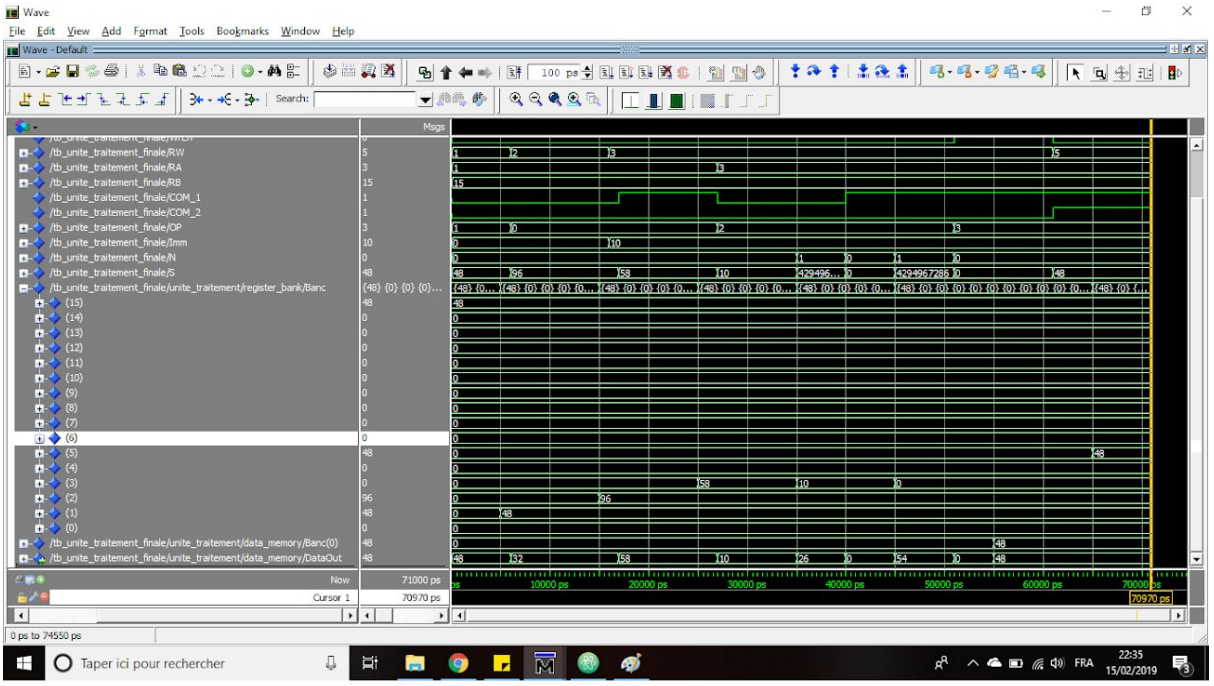


Le registre 15 vaut 0x30 au démarrage. On commence par initialiser tous les signaux en entrée à '0' puis on réalise les opérations suivantes :

- 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)

Le registre R(1) passe effectivement de 0 à 0x30 au premier coup de clock, puis prend la valeur 0x60. Le registre R(2) prend la valeur 0x90 et le registre R(3) la valeur 0x30. Lors de la dernière opération, on a R(5) = 0 - 0x30, ce qui vaut -48 en décimal. Cependant on affiche le résultat en hexadécimal d'un nombre négatif encodé avec le complément à 2. On a logiquement un nombre très grand, mais qui représente néanmoins bien la valeur -48.

Test bench de l'Unité de Traitement Finale

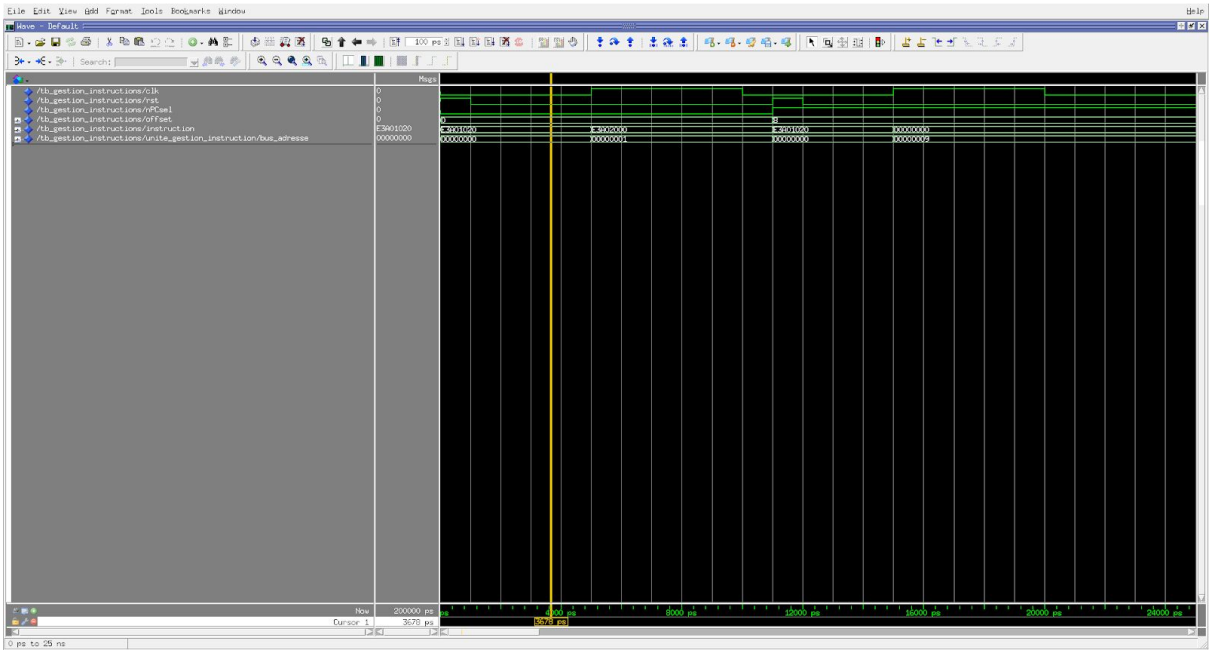


On commence par initialiser à '0' tous les signaux en entrée sauf WrEn qu'on laisse à '1'.
R(15) vaut 48 au départ.
On réalise alors les opérations suivantes:

- R(1) = R(15) -- copie de valeur registre / registre
- R(2) = R(1) + R(15) -- addition de deux registres
- R(3) = R(15) + 10 -- addition registre / immédiat
- R(3) = R(3) - R(15) -- soustraction registre - registre
- M(0) = R(15) -- écriture d'un registre en mémoire
- R(5) = M(0) -- lecture mémoire dans un registre

La copie de de R(15) dans R(1) se passe bien: au premier coup de clock, R(1) vaut bien 48. L'addition de deux registre est également un succès, au deuxième coup de clock, R(2) vaut 96. L'addition entre un registre et un immédiat fonctionne: on charge un immédiat qui vaut 10 en décimal et on le somme avec R(15). Le résultat est bien stocké dans R(3) et vaut 58.
On réalise la soustraction R(3) - R(15) et on stocke cette valeur dans R(3). R(3) change bien et vaut 10. On écrit ensuite la valeur de R(15) en M(0): l'écriture marche et M(0) prend la valeur 48. On termine par lire la valeur de M(0) dans R(5): l'opération fonctionne et on peut voir R(5) prendre la valeur 48.

Test Bench de l'unité de gestion des instructions



Entrées:

- clk (horloge)
- rst (commande de reset asynchrone du registre pc)
- nPCsel (commande du multiplexeur)
- offset (valeur d'incrémentation de l'adresse)

Sorties:

- instruction (l'instruction en mémoire sur 32 bits)

Le comportement de l'entité est le suivant :

instruction <= Instruction Memory(PC),

avec PC un compteur synchrone tel que:

si nPCsel = 0 alors PC = PC + 1
si nPCsel = 1 alors PC = PC + 1 + offset
si rst = 0 alors PC = 0

Observons maintenant le wave du test-bench, au départ on a rst = 1, le bus d'adresse est bien à 0, et l'instruction correspond bien à Instruction Memory(0).
Au premier front montant, on a rst = 0, nPCsel = 0, PC s'incrémente de 1, l'instruction change devenant Instruction Memory(1).
A 10 ns, on a rst = 1, PC retourne à 0, l'instruction redevient Instruction Memory(0).
Au deuxième front montant: rst = 0, nPCsel = 1, offset = 8, PC prend la valeur PC + 1 + offset (0 + 1 + 8 = 9), l'instruction est égale à 0 car il n'y a que les 8 premières lignes de remplies dans Memory Instruction, les autres sont toutes initialisées à 0.

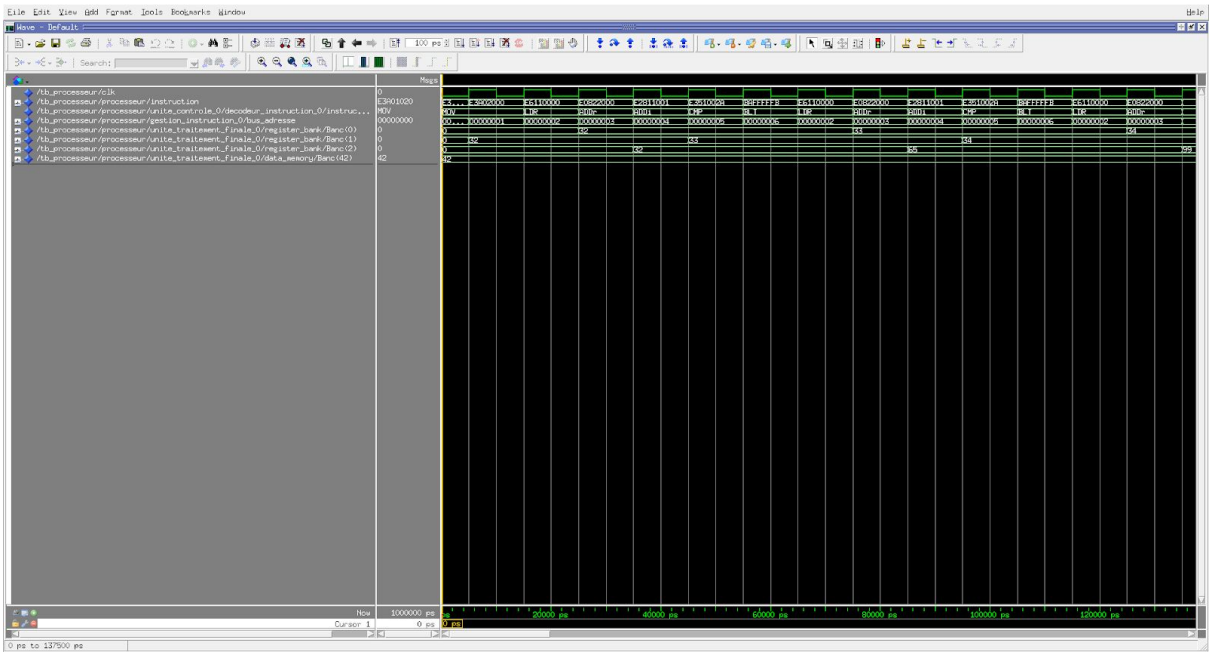
Test Bench du Processeur Monocycle

Le code dans instruction est une boucle qui répète les instructions suivantes :

1. mettre 32 dans R1
2. mettre 0 dans R2
3. charger dans R0, 0(R1)
4. R2 = R2 + R0
5. R1 = R1 + 1
6. si R1 < 42 revenir à 3.
7. sinon enregistrer R2 dans 0(R1)

En chargeant les registres de manière judicieuse, c'est-à-dire en mettant dans le registre la valeur de son adresse, le code additionne simplement les entiers de 20 à 41 avant de stocker le tout dans le registre 42. R1 sert de compteur, R2 contient la somme en cours et R0 l'entier qui est ajouté à la somme.

Observons maintenant les fenêtres wave :



À chaque front montant, le processeur exécute l'instruction précédente avant de se mettre à jour.
Au premier front montant :

le registre 1 reçoit 32, l'adresse est incrémentée correctement

Au deuxième front montant :

le registre 2 reçoit 0, l'adresse est incrémentée correctement

Au troisième front montant :

le registre 0 reçoit 0(R1), soit 0(32), c'est-à-dire 32

Au quatrième front montant :

le registre 2 reçoit R2 + R0 (0 + 32 = 32)

Au 5e front montant :

le registre 1 reçoit R1 + 1 (32 + 1 = 33)

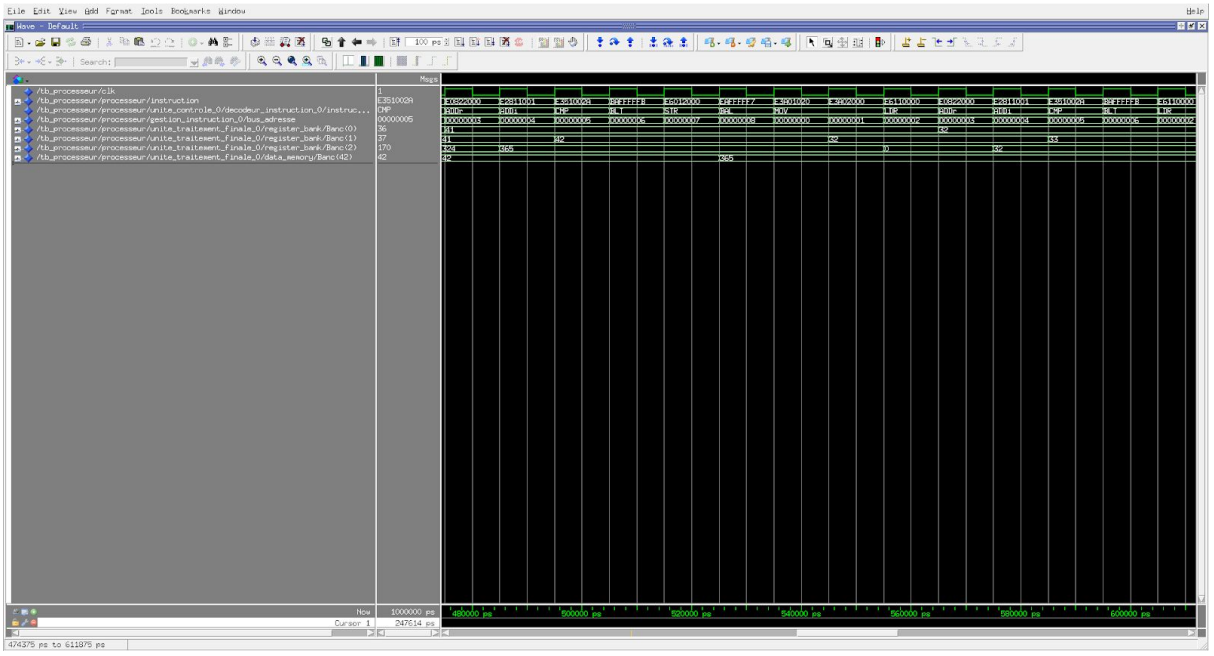
Au 6e front montant :

le registre R1 est comparé avec 42

Au 7e front montant :

comme R1 < 42 on reboucle à l'instruction n°2, le bus adresse passe donc de 6 à 2

Cette boucle se poursuit jusqu'à ce que R1 = 42, à ce moment la condition n'est plus vérifiée, on ne saute plus à l'instruction 2 mais on continue à la 7.



À 525 000 ps :

l'instruction 7 est exécutée, le registre 0(R1) (avec R1 = 42) reçoit le le contenue de R2 soit 365.

Au front montant suivant :

on recommence le programme depuis le début, l'adresse de l'instruction passe donc à 0.