Rapport

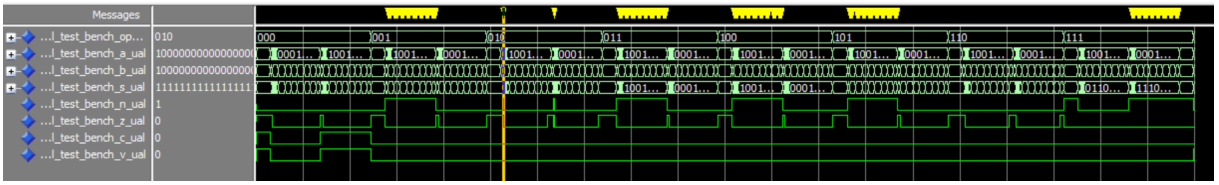
Projet d'Électronique Numérique - VHDL Processeur Monocycle

Membre : Vladislav Levovitch BALAYAN
Soutien et aide : Ayoub LADJICI
Aide à la rédaction et aide au débogage : ChatGPT + Monsieur DOUZE Yann

Partie 1 : Unite de traitement

1.1 Unité Arithmétique et Logique

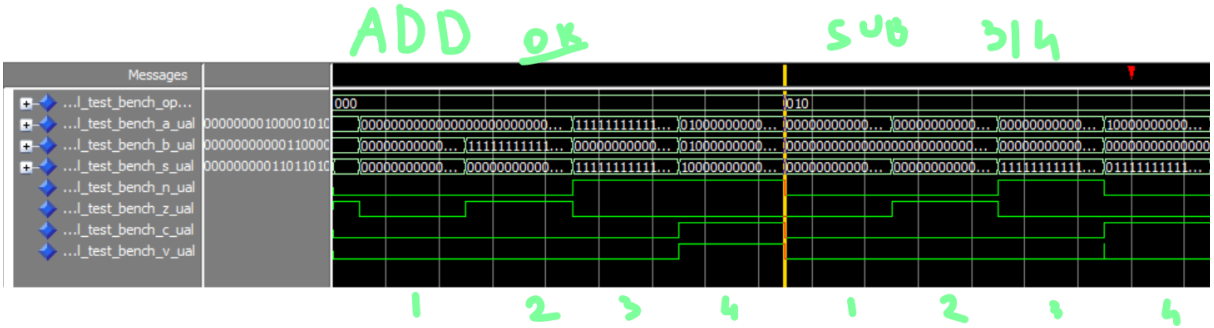
Figure 1.1.1 : Chronogramme de l'UAL avec 2 boucles for testant toutes les opérations de l'UAL.



Nous remarquons bien que les drapeaux nuls et négatifs visibles tout au long du test bench sont cohérents avec les parties où les résultats sont négatifs ou nuls. Par ailleurs, on remarque que les seuls cas de débordement et de retenue se produisent au niveau de la zone d'addition.

Dans un second temps, le test ne vérifiant pas la zone de soustraction ici, nous allons la tester pour vérifier sa validité plus en détail.

Figure 1.1.2 : Chronogramme de l'UAL avec des valeurs précises pour les opérations ADD et SUB.



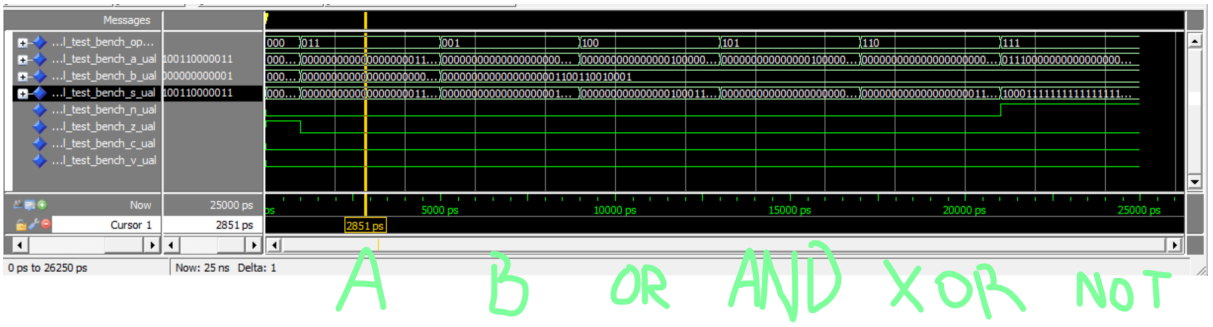
Dans cette sous-partie, nous avons testé en détail les opérations ADD et SUB dans un deuxième banc de test. Tout d'abord avec un test d'opération simple en zone 1, puis un test de nullité cherchant à lever seulement le drapeau Z en zone 2, ensuite un test de négativité cherchant à lever seulement le drapeau N en zone 3, et enfin un test en zone 4 pour vérifier les drapeaux de débordement et de retenue. Tous nos tests ont validé nos assertions de résultats et de drapeaux, hormis le test de retenue en zone 4 pour l'opération de soustraction.

Figure 1.1.3 : Extrait de code montrant la partie posant problème au niveau de la retenue.

```
129     -- Test Sub soustraction avec Debordement_Retenue
130     SIGNAL_Test_Bench_A_UAL <= x"8000_0000";
131     SIGNAL_Test_Bench_B_UAL <= x"0000_0001";
132     wait for 1 ns;
133
134     assert (SIGNAL_Test_Bench_S_UAL = x"7FFF_FFFF") report "Test SUB Debordement : Erreur resutat" severity error;
135     assert (SIGNAL_Test_Bench_Z_UAL = '0') report "Test SUB Debordement : Z_UAL incorrect" severity error;
136     assert (SIGNAL_Test_Bench_N_UAL = '0') report "Test SUB Debordement : N_UAL incorrect" severity error;
137     assert (SIGNAL_Test_Bench_C_UAL = '1') report "Test SUB Debordement : C_UAL incorrect" severity error;
138     assert (SIGNAL_Test_Bench_V_UAL = '1') report "Test SUB Debordement : V_UAL incorrect" severity error;
139     wait for 3 ns;
```

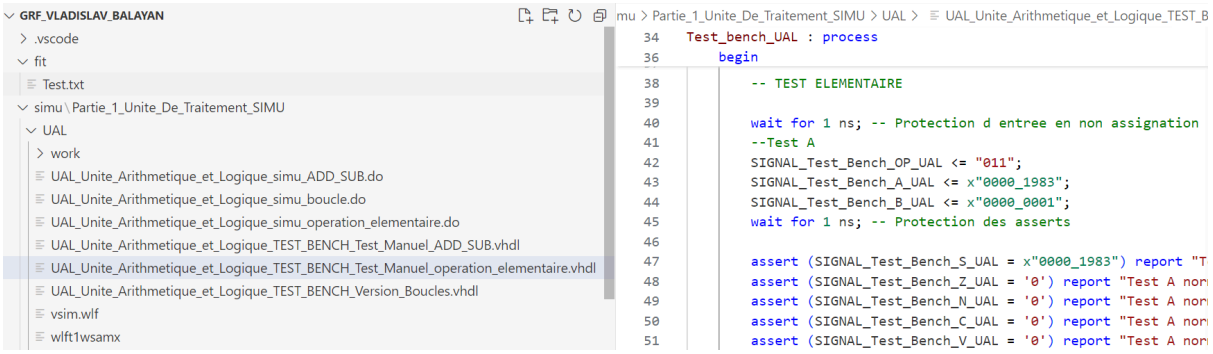
Malheureusement, après plusieurs essais et vérifications, nous avons effectué de multiples corrections et modifications, retrouvables dans l'annexe 1.1 Debogage UAL. Cependant, nous avons rencontré un problème persistant avec la retenue négative que nous n'avons pas pu résoudre malgré un long acharnement, hormis des résultats pour la soustraction tout de même corrects. Nous avons cherché aide et conseil, mais résoudre cela aurait nécessité de reprendre entièrement la structure d'un collègue, ce qui aurait perdu de son intérêt. Pour cette raison, nous allons continuer le projet en sachant que la retenue négative reste ambiguë et non résolue.

Figure 1.1.4 : Chronogramme de l'UAL avec des valeurs précises pour les opérations A, B, OR, AND, XOR et NOT.



Nous avons réitéré les tests dans un troisième banc de test pour vérifier les opérateurs A, B, OR, AND, XOR et NOT. Ceux-ci n'ont levé aucun rapport d'erreur et se sont donc avérés concluants.

Figure 1.1.5 : Extrait de code montrant en particularité la partie A.



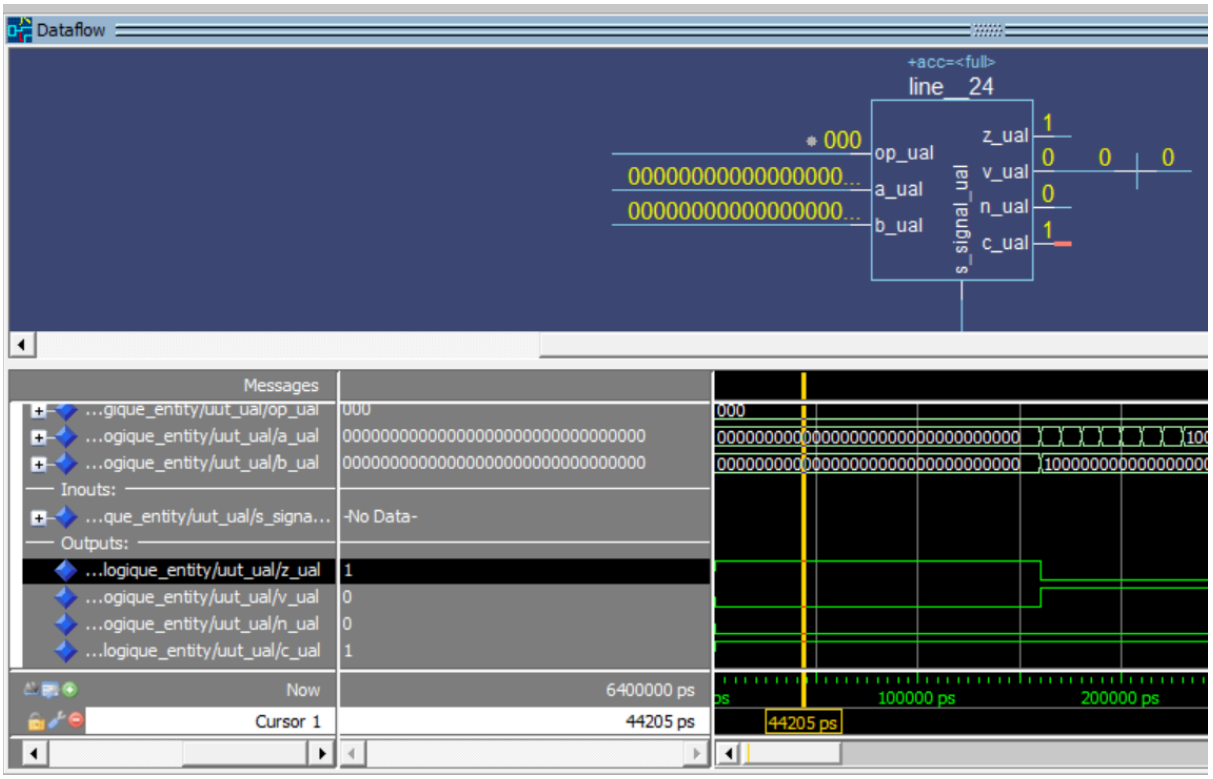
Voici un extrait de code pour la Figure 1.1.4 afin de vérifier en détail que nos valeurs de sortie sont correctes de manières plus lisible que directement sur le chronogramme.

Ici, nous obtenons en sortie 1983 au lieu de 0001 ; aucun des drapeaux ne doit être levé et c'est bien le cas. La vérification est bonne.

Annexe de débogage

1.1 Debogage UAL

Nous venons de constater par la simulation que pour l'opération ADD, 0 + 0 a bien une comme drapeaux 1 pour zéro mais a aussi 1 pour la retenenu ce qui n'est pas correct. En effet on remarque que la retenue est activée presque par défaut en permanence.




```
60  -- Retenu
61      if (((OP_UAL = "000") and (A_UAL(31) = B_UAL(31))) or ((OP_UAL = "010") and (A_UAL(31) /= B_UAL(31))))  then
62          if ((std_logic_vector (signed(A_UAL)+signed(B_UAL)) > x"7FFF_FFFF") or (std_logic_vector (signed(A_UAL)+signed(B_UAL)) < x"8000_000
63              C_UAL <= '1';
64          end if;
65      end if;
66
67      if (((OP_UAL = "010")) and (A_UAL(31) /= B_UAL(31))) and (A_UAL(31) = B_UAL(31)) then
68          if ((std_logic_vector (signed(A_UAL)+signed(B_UAL)) > x"7FFF_FFFF") or (std_logic_vector (signed(A_UAL)+signed(B_UAL)) < x"8000_000
69              C_UAL <= '1';
70          end if;
71      end if;
```

Entre temps nous avons essayé une autre approche du test d'où le nouveau paterne mais nous avons aussi remarqué que le drapeaux negatif est nous avons eu plusieurs version de la retenu :

```
-- Retenu Bis
-- if (((OP_UAL = "000") and (A_UAL(31) = B_UAL(31))) or ((OP_UAL =
"010") and (A_UAL(31) /= B_UAL(31))))  then
```

```
--        if (((std_logic_vector (signed(A_UAL)+signed(B_UAL)) >
x"7FFF_FFFF") and (B_UAL(31) = 0)) or (std_logic_vector
(signed(A_UAL)+signed(B_UAL)) < x"8000_0000")) then -- On verifie si la
valeur entiere de A + B depasse le max ou min
--            C_UAL <= '1';
--        end if;
--    end if;
--
--    if (((OP_UAL = "010")) and (A_UAL(31) /= B_UAL(31))) and
(A_UAL(31) = B_UAL(31)) then
--        if ((std_logic_vector (signed(A_UAL)+signed(B_UAL)) >
x"7FFF_FFFF") or (std_logic_vector (signed(A_UAL)+signed(B_UAL)) <
x"8000_0000")) then -- On verifie si la valeur entiere de A - B depasse
le max ou min
--            C_UAL <= '1';
--        end if;
--    end if;

-- Retenu Tris

--    if ((OP_UAL = "000"

--   if ((OP_UAL = "000" or OP_UAL = "010") and ((A_UAL(30) =
B_UAL(30)) ) -- On verfie le bit de plus gros poids

--    C_UAL <= '1';

--   end if;

-- Retenu Tetra essaie
    if (((OP_UAL = "000") and (A_UAL(31) = B_UAL(31) and
B_UAL(31)/=B_UAL(30) and B_UAL(30) = A_UAL(30)))) then -- or ((OP_UAL =
"010") and (A_UAL(31) /= B_UAL(31) and A_UAL(31)/=A_UAL(30) and
B_UAL(30) /= A_UAL(30))))  then
        C_UAL <= '1';
    end if;

-- Retenu Pinta essaie
--    if (OP_UAL = "010") and ((signed(A_UAL) - signed(B_UAL)) <=
to_signed(2**31-1,32) and (signed(A_UAL) - signed(B_UAL)) >=
to_signed(-2**31, 32))
```

```
--         C_UAL <= '1';
--     end if;

-- Retenu soustraction
    if ((OP_UAL = "010") and (A_UAL(31) /= B_UAL(31)) and
(S_SIGNAL_UAL(31) /= A_UAL(31))) then
        V_UAL <= '1';
    end if;
```

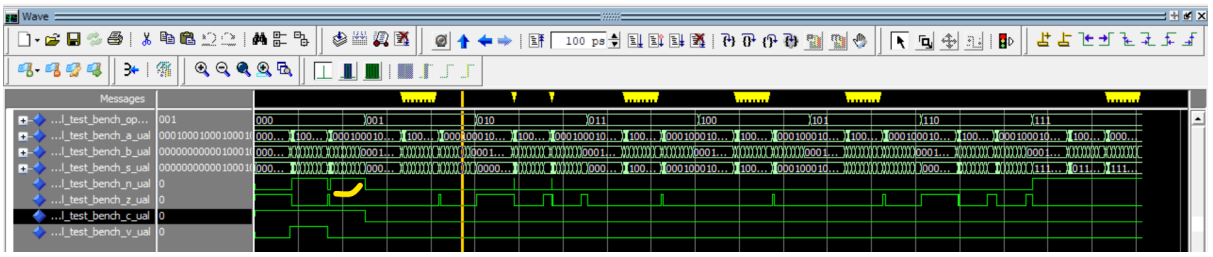


Figure simplement illustrative