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

Remerciement

Mes remerciements vont évidemment tout d'abord à mon ami et soutien le plus fidèle, Ayoub LADJICI, qui, malgré les rumeurs, ragots et ses propres difficultés, a su m'épauler tout au long de cette année, m'encourager et me donner tout l'espoir qu'il a pu.

Un très grand merci aussi à Monsieur MARTIN qui m'a beaucoup appris pendant nos séances encadrées depuis le cinquième semestre. Avec son collègue Monsieur PINNA, qui par le cours d'électronique numérique 1, m'ont fait découvrir une nouvelle passion pour ce langage étant l'amour parfait entre l'informatique et l'électronique.

Mes remerciements vont aussi à Monsieur DOUZE qui a toujours fait preuve d'une grande pertinence dans ses réponses même lorsque mes questions étaient des plus basiques. Merci à lui et à notre très bien aimé Thibault (alias Papa des Ei) qui ont beaucoup participé à mon épanouissement personnel et qui m'ont appris à bien structurer mes codes.

Pour finir, j'aimerais remercier toutes ces personnes extraordinaires que j'ai rencontrées cette année dans la formation de mes rêves depuis de nombreuses années, et qui, par leur simple présence, me rendaient heureux : les copains du TPA, Inessa, Nicolas, Maxime, Ayman, Victor, Quentin, Papa, Ilias, Nadir, et ceux du TPF, nos incroyables alternants avec une mention toute particulière pour ces perles qui m'ont accompagné tout au long de l'année dans tous nos différents travaux en traitement du signal : Liza, Jihen, Amel, Salma, Farah et Ioana, les alternants de mon groupe, Asdjad, Fatou, Enes, Hakan, Vedat, Abde, Richard, Amine, les deux Tarek, Michel, Emmanuel, Théo, et les alternants qu'on ne pouvait voir que trop rarement grâce aux restructurations mais qui ont tout de même su rayonner : Arthur, Léa, Gloire A Dieu, Gabriel, Rémy, Ali, Elsa, Karlitou, Sékouba, Tom et pour finir Mon Pitit Chou.

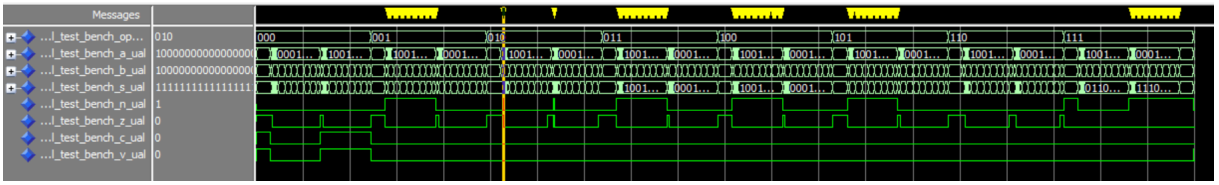
Introduction

Dans le cadre de notre parcours universitaire en 3ème année d'école, nous sommes amenés, en Electronique Numérique 3, à réaliser et simuler un cœur de processeur mono-cycle. Ce projet implique la conception et la simulation de différents blocs principaux tels que l'unité de traitement, l'unité de gestion des instructions et l'unité de contrôle. L'objectif est de diviser le projet en petits composants tels que des multiplexeurs, des registres, des bancs de mémoire, ainsi qu'une Unité Arithmétique et Logique (UAL). Cela nous permettra de simuler, d'assembler et de tester notre modèle sur une carte FPGA.

Partie 1 : Unite de traitement

1.1 Unité Arithmétique et Logique

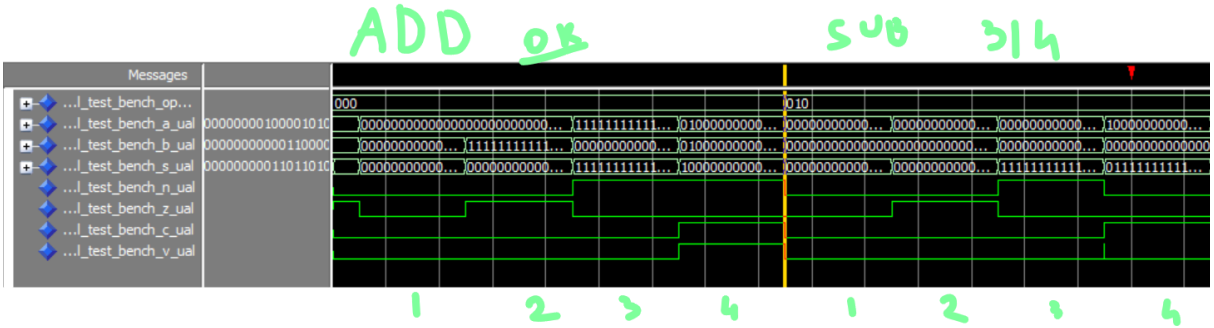
Figure 1.1.1 : Chronogramme de l'UAL avec 2 boucles for testant toutes les opérations de l'UAL.



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.



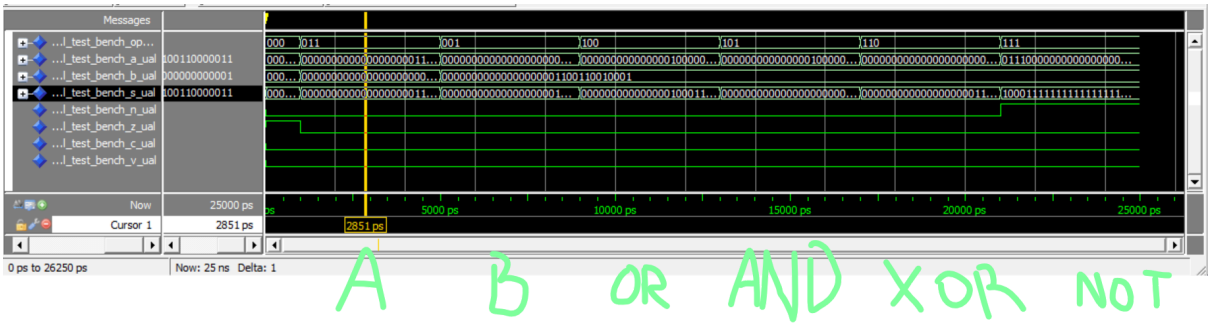
Dans cette sous-partie, nous avons testé en détail les opérations ADD et SUB dans un deuxième banc de test. Tout d'abord avec un test d'opération simple en zone 1, puis un test de nullité cherchant à lever seulement le drapeau Z en zone 2, ensuite un test de négativité cherchant à lever seulement le 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;
```

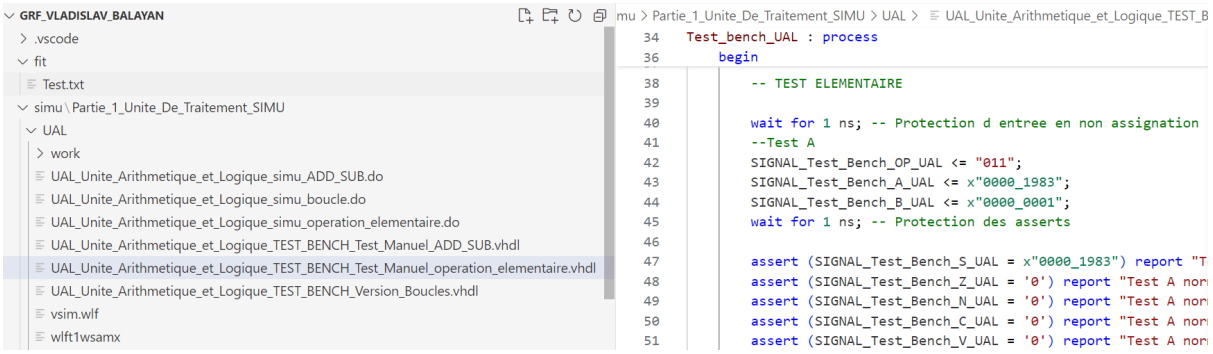
Malheureusement, après plusieurs essais et vérifications, nous avons effectué de multiples corrections et modifications, retrouvables dans l'annexe 1.1 Debogage UAL. Cependant, nous avons rencontré un problème persistant avec la retenue négative que nous n'avons pas pu résoudre malgré un long acharnement, 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.

1.2 Banc de Registre

Tout d'abord, nous avons pensé à ajouter un nouveau module s'occupant exclusivement de l'horloge pour une meilleure compartimentation des fonctions des modules. Celui-ci a été directement testé en condition d'utilisation normale, ici dans un banc de test. À terme, il servira à toutes les fonctions et modules synchrones.

Figure 1.2.0 : Chronogramme de l'horloge générée depuis un module extérieur

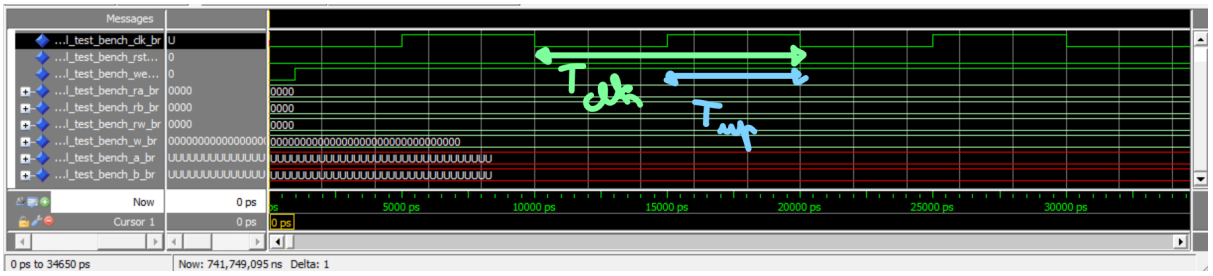


Figure 1.2.1 : Chronogramme du Banc de Registre indiquant la mise à jour des registres en fonction de l'adresse donnée et demandée en lecture ou écriture.

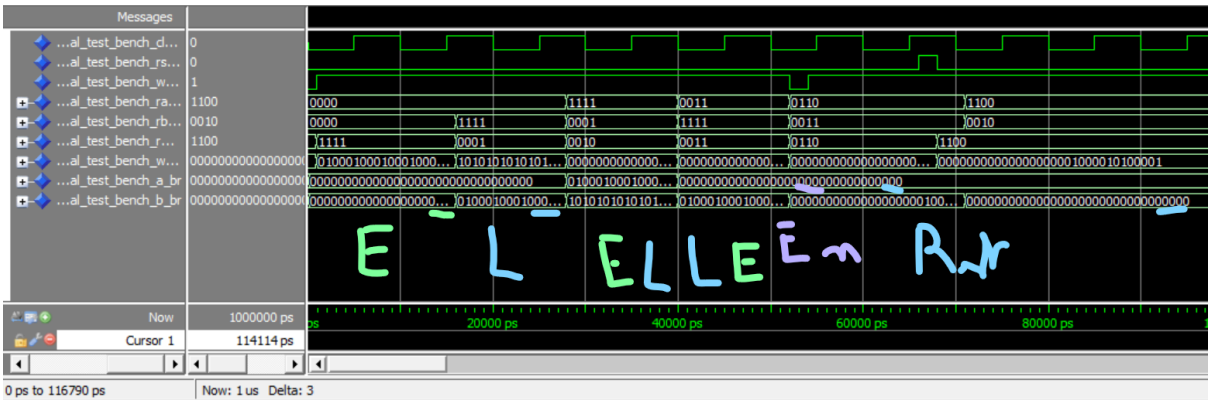
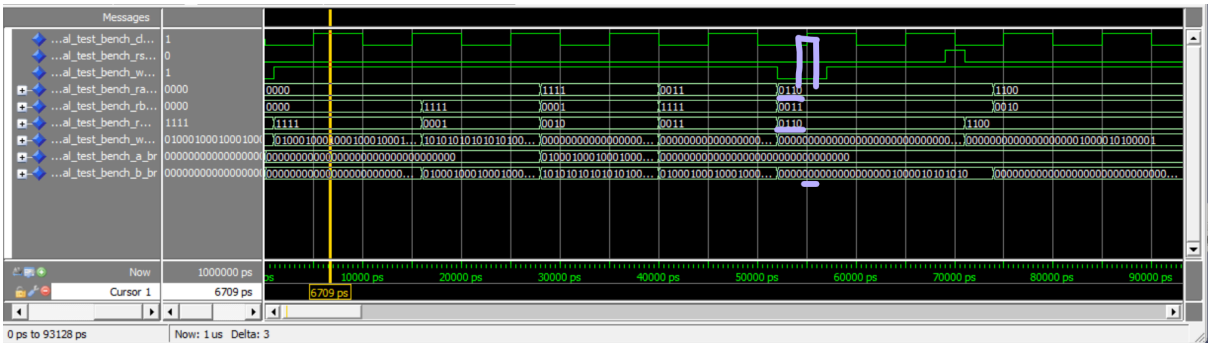


Figure 1.2.2 : Chronogramme du Banc de Registre indiquant le fonctionnement de l'enable



Nous avons testé l'écriture simple, la lecture simple, l'écriture simultanée, l'écriture avant lecture, l'écriture après lecture, le reset et l'enable. Tous les tests avec assert ont été réussis. Nous constatons une remise à zéro après le reset et que les sorties A et B sont correctement assignées en fonction des adresses de registres demandées. Enfin, nous constatons de manière plus claire sur la Figure 1.2.2 que lorsque l'écriture est désactivée, bien que les registres soient identiques entre celui en écriture et en lecture, au coup d'horloge les valeurs A et B ne sont pas mises à jour.

Conclusion

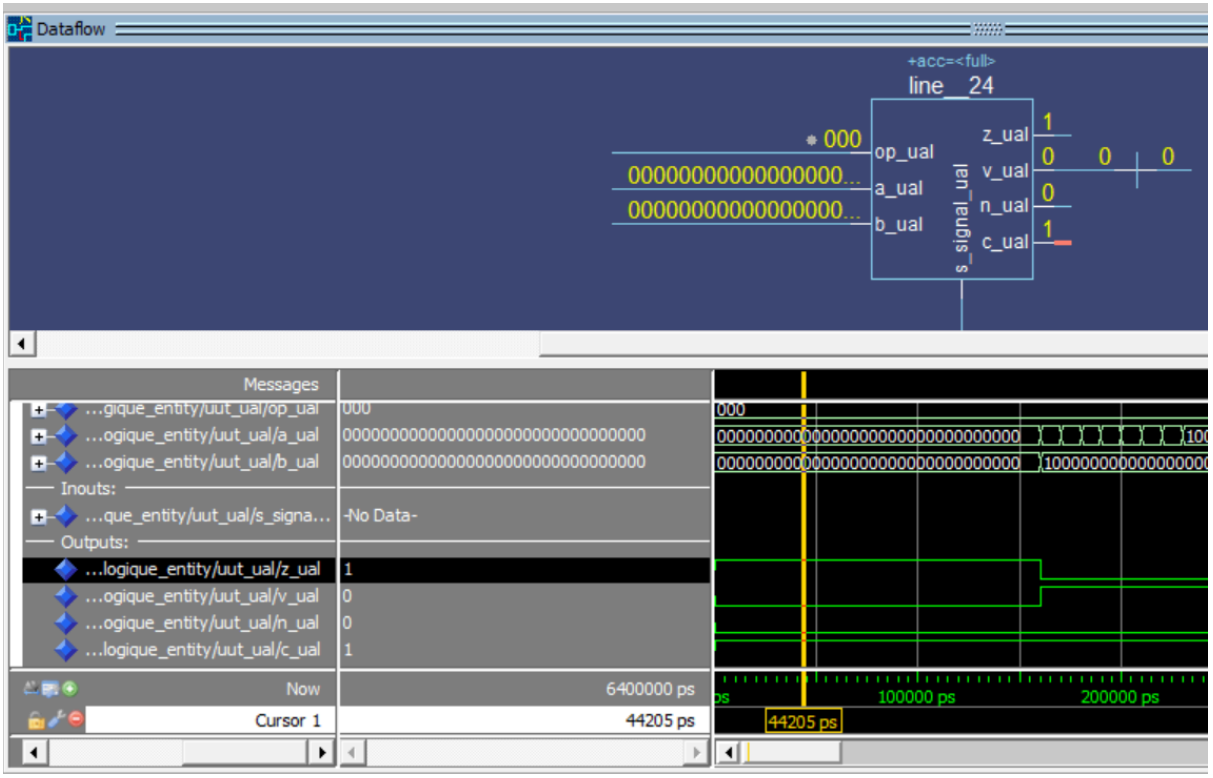
Bien que le projet n'ait pas été achevé et ait connu un démarrage sur les chapeaux de roues, avec un très long blocage au niveau de l'UAL et des problèmes personnels, celui-ci m'aura beaucoup apporté. J'ai finalement trouvé une forme de libération et surtout une satisfaction personnelle qui me donne encore plus de motivation pour la suite de mon parcours universitaire et humain. Malgré le fait que certains n'aient pas cru en moi pendant longtemps, je reste convaincu que cela était principalement dû à un manque de communication. J'ai tenté de remédier à cela trop tardivement, je peux l'admettre, notamment en exposant mes difficultés par e-mail en raison de l'absence au niveau du stage, mais si vous souhaitez en savoir davantage, ce matin-là, j'avais un rendez-vous en matinée à l'annexe 12. Douze, ce qui explique pourquoi je n'étais pas présent. Par ailleurs, j'avais besoin d'une bonne préparation nonobstant mon innocence avérée ou non, ce qui, comme nous l'avons vu, s'est révélé insuffisant en raison du peu de temps accordée à celle ci.

Pour conclure ce rapport, j'espère sincèrement, Monsieur DOUZE, que nos relations seront réellement apaisées à l'avenir, malgré vos récentes et très graves accusations vis-à-vis des PC, accusations qui m'ont profondément touché. J'espère également que notre belle filière Ei, EiSE, Ei2I, EiLI, EiST continuera de prospérer comme elle l'a toujours fait, grâce à Thibault HILAIRE, Dimitri GALAYKO, Yann DOUZE, Benoît FABRE, Annick ALEXANDRE, et enfin Michel REDON, et que toutes les futures promotions s'y sentiront aussi épanouies que j'aurais pu l'être, tout comme l'a été notre grand frère Monsieur VIATEUR.

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.



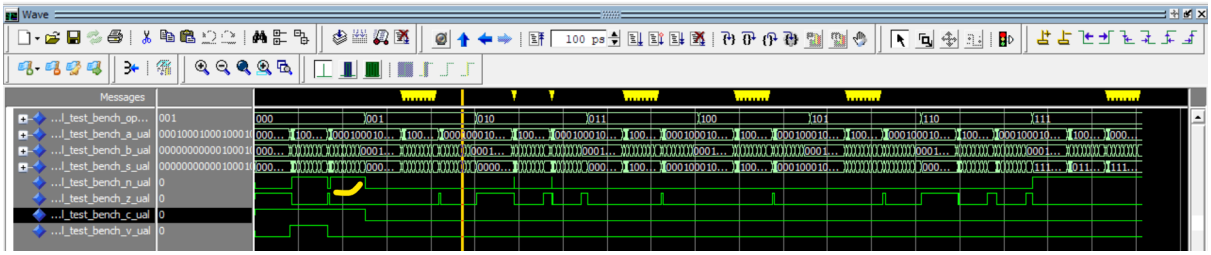

```
60    -- Retenu
61        if (((OP_UAL = "000") and (A_UAL(31) = B_UAL(31))) or ((OP_UAL = "010") and (A_UAL(31) /= B_UAL(31))))  then
62            if ((std_logic_vector (signed(A_UAL)+signed(B_UAL)) > x"7FFF_FFFF") or (std_logic_vector (signed(A_UAL)+signed(B_UAL)) < x"8000_000
63                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;
```



12. Douze