Rapport

Processeur mono-cycle

Introduction au Logic Design / VHDL

Maxime Goeller
Eve Gresse

Introduction :

Dans ce rapport nous allons discuter de la méthode que nous avons choisie afin de résoudre certain problèmes, ainsi que l'organisation de notre code (Partie Exercice) et de nos bancs de test (Partie Banc de test).

Partie 1 : Unité de traitement.

Exercice:

1. *ALU (UAL.vhd):*
   Pour cette partie, nous avons simplement créé un process avec pour signaux d'entrée a, b, op ainsi que y, le signal que nous avons gardé comme signal temporaire pour la sortie S.
   Dans ce processus nous avons simplement mis dans y l'issue attendue par l'ALU en fonction de l'opérateur, avec une sécurité qui met tous les bit à 0.
   Enfin, nous avons établi les bit de flag de la façon suivante :
   - N : y en signé est inférieur à 0 => 1 sinon 0
   - Z : y en non-signé est égal à 0 => 1 sinon 0
   - V : (le bit de poid fort de a est égal à celui de b mais différent de celui de y) et l'op est une addition ou soustraction => 1 sinon 0
   - C : (y est inférieur ou égal à a ou b) et (a et b en signé sont différents de 0) et l'op est une addition => 1 sinon 0

2. *Banc de registre (RegisterBank.vhd):*
   Après avoir initialisé le Banc (15 places de 32 bits), nous avons donc connecté dans un processus la Clock et le Reset à la logique suivante :
   - Lorsque le Reset est à 1, le registre se réinitialise.
   - Lors du front montant de la Clock, nous écrivons dans le registre RW le message W, si tant est que l'écriture soit demandée.

   Suite à cela, nous passons les signaux de sortie A est B à l'issue demandée par RA et RB.

3. *Assemblage de L'ALU et de l'unité de traitement (ProcessingUnit.vhd):*
   Ce processus assemble L'ALU et le banc de registre comme sur le schéma de la partie 1.1.2.

4. *Extension de signe (ExtensionSigne.vhd)*:
   Dans cette partie nous étendons le vecteur en entrée qui est de 8 bits en un vecteur de 32 bits tout en conservant son signe.

5. *Mémoire de Donnée (MemoireDonnees.vhd):*
   Ce processus stocke le DataIn en entrée dans une mémoire si le bit WrEn est à A. Il le stocke à la place Addr dans la mémoire. La mémoire contient 64 places de 32 bits.

6. *Multiplexeur (Multiplexeur.vhd):*
   Ce processus est tout simplement un multiplexeur 2 vers 1 qui renvoie A ou B en fonction de COM.

7. *Assemblage de la partie 1 (UniteDeTraitement.vhd):*
   Cette partie assemble tous les processus de la partie de la manière qui est indiquée sur le schéma de la partie 1.3.

Banc de test:

Cette partie a deux banc de tests:

- Pour tester la partie 1.1.2 il faut lancer le simuPro.do
- Pour tester la partie 1.3 il faut lancer le simuTraitement.do

Les tests bench comportent les tests demandés dans le sujet.

Partie 2: Unité de gestion des instructions.

Exercice:

1. *Extension PC (ExtensionPC.vhd):*
   Reprend exactement le même code que ExtensionSigne.vhd à l'exception qu'il prend 24 bits en entrée à la place 8 pour ExtensionSigne.vhd.

2. *Instructions (instruction_memory.vhd):*
   Reprend le code donnée dans l'annexe: Les instructions en assembleur à donner au processeur.

3. *Multiplexeur (Multiplexeur.vhd):*
   Reprend le même schéma que le multiplexeur de l'exercice 1 mais ne fait pas exactement la même chose, au lieu de juste renvoyer A ou B, il renvoie A+1 ou A+B+1.

4. *Registre PC (RegistrePC.vhd):*
   Renvoie DataIn si reset=0 sinon renvoie un vecteur de 32 bits de 0.

5. *Assemblage de la partie 2 (UniteInstructions.vhd):*
   Assemble la partie 2 en suivant le schéma de celle-ci

Banc de test:

Pour lancer la test bench il suffit d'exécuter le simuInstructions.do.
Le banc de test de cette partie est assez sommaire mais permet de voir le bon fonctionnement de cette partie, on voit bien qu'après 50 ms le code effectue 5 instructions sachant que la période donnée a la clock est de 10ms. On peut augmenter ou diminuer le wait pour voir le comportement du processus.

Partie 3: Unité de Contrôle

Instruction	nPCSel	RegWr	ALUSrc	ALUCtr	PSREn	MemWr	WrSrc	RegSel	RegAff
ADDi	0	1	1	000	1	0	0	1	0
ADDr	0	1	0	000	1	0	0	0	0
BAL	1	0	X	X	0	0	0	1	0
BLT	0 / 1	0	X	X	1	0	0	1	0
CMP	0	0	0 / 1	010	1	0	0	1	0
LDR	0	1	0	011	0	0	1	0 / 1	0
MOV	0	1	0 / 1	001	0	0	0	1	0
STR	0	0	0	001	0	1	0	0 / 1	1

Légende:
[0-1]+ : Valeur fixe.
0 / 1 : Selon la situation, multiples valeurs possibles.
X : Valeur n'impacte pas la commande.

Exercice:

1. *Decodeur (DecodeurInstruction.vhd):*
   Le décodeur instruction prend une instruction et les flags de la précédente instruction en entrée et renvoie toutes les commandes nécessaires à l'exercice 1 et 2.
   Pour cela il y a deux processus: un qui récupère l'instruction et l'autre qui configure les commandes.

2. *Registre 32 bits Afficheur (Registre32bits.vhd):*
   Suit le même schéma que Registre PC, seulement écrit uniquement si WE est à 1.

3. *Registre 32 bits PSR (Registre32PSR.vhd):*
   Suit le même schéma que le registrePC, seulement si Reset est 0 et que la clock monte on vérifie si WE est à 1, si c'est le cas on écrit les flags N, C, V et Z de l'instruction.

Banc de test:

Pour cette partie nous avons préféré reporter nos banc de test dans la partie 4, comme elle ne rajoute que la liaison entre les trois premières parties.

Partie 4: Assemblage et validation du processeur.

Exercice:

- *Assemblage des partie 1, 2 et 3 (UniteProcesseur.vhd):*
  Assemblage des 3 premières parties en suivant le schéma indiqué.

Banc de test:

Pour lancer le banc de test il faut lancer le simuProcesseur.do.
La mémoire interne a été mise à 12 et est décrémenté de 1 à chaque passage. Donc dans les adresse de 0x20 à 0x2A il y a une suite de 1 à 12.
Le résultat attendu est donc de 0…01000001 en binaire, 0041 en hexadécimal et 65 en décimal. Nous avons donc fait qu'un seul assert qui est le résultat final, tout en nous appuyant sur l'ensemble des signaux pour debugger.

Partie 5: Test du processeur sur carte FPGA

Exercice (cet exercice est rangé dans le dossier de l'exercice 4):

1. *Top Level pour FPGA (top_level.vhd):*
   Assemble les afficheurs et le UniteProcesseur.vhd.

2. *Affichage (Affichage.vhd):*
   Renvoie les valeurs de HEX en fonction de DATA et du POL.

Banc de test:

Pour lancer le banc de test il faut lancer le simutop.do.

Pour ce qui est du test sur carte FPGA malgré un banc de test concluant (HEX 0-3 aux bonnes valeurs : 0041) et tous les signaux au vert sur ModelSim, nous n'avons pas réussi à faire fonctionner le code sur le fpga, ce dernier affichant 0000 lors de notre dernière tentative. Nous avons beaucoup cherché mais nous n'avons pas trouvé la cause de ce problème.