Module d'approfondissement MSE
Reconfigurable Computing

M. Starkier - E.Messerli

Labo Decode

5/10/2009 – V1.1

Informations générales

Vous devez rendre à l'issue de ce labo :

1. La description VHDL complète commentée
2. Les codes assembleur de vos tests
3. Un bref rapport expliquant et justifiant vos choix de conception en précisant les tests que vous avez effectués pour valider votre design.

Objectif du laboratoire

L'objectif est de concevoir et de réaliser le décodage des instructions de traitement. Ce travail s'effectue en trois étape :

1. Gestion d'instructions de traitement arithmétique et logique sans shift.
2. Réalisation du barrel shifter
3. Ecriture des flags de status dans le CPSR. Gestion d'instructions de test et du report de carry

Outils

Qestasim compilation simulation VHDL et system Verilog

WinARM toolchain (assembleur / linker) ARM

ARM9_registers_viewer (affichage des registres du processeur)

Fichiers et documents fournis

1- **Description VHDL de l'ARM9-REDS** :

- Bloc Fetch simplifié
- Bloc Decode à compléter
- Bloc Execute simplifié
- Bloc Memory simplifié
- Banque de registre
- Top du modèle

2- Fichiers de compilation et test

- Test bench System Verilog
- Scripts TCL

Travail à effectuer

Informations labo Decode

1- Vous n'avez pas à gérer les instructions conditionnelles pour ce labo. Toutes les instructions implémentées doivent s'exécuter sans condition.

2- Le forwarding ne doit pas être géré dans le cadre de ce labo. Pour exécuter correctement deux instructions successives dépendantes, vous devrez ajouter des instructions NOP au code assembleur

3- A chaque étape, vous devez écrire le code assembleur de test (uniquement les instructions implémentées), puis assembler et générer le fichier exécutable. Vous pouvez tester ensuite en pas à pas avec QuestaSim, le test bench fourni et un outil de visualisation des registres ARM (ARM9_registers_viewer)

Note importante: Le test-bench fourni est complet : Le model supporte toutes les instructions (à quelques exceptions près). Vous devez être attentif à n'utiliser que les instructions supportées par votre description en cours de développement. Dans le cas contraire, le test-bench vous renverra une erreur.

Etape 1 – Instructions arithmétiques et logiques:

Vous devez écrire votre description VHDL dans le bloc decode afin que le processeur ARM9-REDS traite correctement les 8 instructions suivantes :

MOV, MVN, ADD, SUB, AND, EOR, ORR, NOP

L'opérande **Oprnd2** sera uniquement de type **Immediate value** ou **Register**.

Vous devez également fournir le code de commande de l'alu. Une table est fournie dans le paquetage.

Note :

1 - Pour l'étape 1, vous n'avez pas à gérer les shifts ou la mise à jour des flags de status

2 - Pour cette étape, la valeur immédiate doit être comprise entre 0 et 0xFF (car le shift n'est pas implémenté).

Etape 2 – Barrel Shifter:

Vous devez écrire votre description VHDL dans le sous-bloc shifter du bloc Execute, afin que le processeur ARM9-REDS traite correctement les shifts de l'opérande **Oprnd2 :**

Logical shift left immediate

Logical shift right immediate

Arithmetic shift right immediate

Rotate right immediate

Vous devez également tester des instructions de l'étape 1 avec des **valeurs immédiates plus grande que 0xFF**, telles que 0x104,0xFF0,0xFF00,0xFF000,0xFF000000,0xF000000F (utilisation implicite d'une rotation à droite).

Note : Pour l'étape 2, vous n'avez pas à gérer la mise à jour des flags de status

Etape 3 : Mise à jour des flags de status, instructions de test et report de carry

Vous devez écrire ou modifier la description VHDL afin que le processeur ARM9-REDS traite correctement les 6 instructions suivantes :

ADC, SBC report de carry

TST,TEQ, CMP, CMN test et comparaison

Les instructions implémentées à l'étape 1 devront mettre à jour les 4 flags de status du registre CPSR en fonction de **S**.

Annexe : Test et simulation du projet

1. WinARM tool-chain

a. installation

Chargez l'exécutable WinARM-20060606.exe sur \\eint20\Reds\Labo\ReCo. Installez le dans C:\WinARM.Rajoutez dans la variable d'environnement "PATH" les chemins C:\WinARM\bin et C:\WinARM\utils\bin.

b. Utilisation

L'interface utilisateur de WinARM s'appelle le Programmers Notepad2 qui vous permet d'éditer le code assembleur et d'assembler :
Menu Démarrer -> Shells -> Programmers_Notepad.exe .

Complétez le nom du fichier source (sans extension) dans le makefile fourni avec le projet pour assembler et linker.

Pour compiler, utiliser le menu Tools --> Make All. Le compilateur va créer plusieurs fichiers dans le répertoire du projet. Ceux qui nous intéressent sont :

- *source*.bin, fichier contenant les instructions sous format binaire (même nom que le fichier source avec extension .bin)
- *source*.lst, fichier contenant l'adresse de chaque instruction, ainsi que l'instruction en format hexadécimal.

c. Utilisation avec QuestaSim

Copiez le fichier *source*.bin dans le répertoire de travail de QuestaSim. Le test bench accède à ce fichier pour émuler la mémoire d'instruction. Pour cela, le nom du fichier .bin doit correspondre à celui écrit dans le fichier du test bench **memory.sv.** *(modifiez memory.sv si nécessaire).*

2. Installation du projet

Décompressez le projet fourni dans un répertoire D:\ReCo\..... et changez de répertoire dans QuestaSim (Change Directory).

3. Compilation et simulation

Pour compiler, lancer le script fourni **Compile.do**.
Pour simuler, lancer le script fourni **Simule.do**.
Pour charger les signaux, lancer le script **Load_Signals**.do

4. Test

Lancer le viewer **ARM9_registers_viewer** (Tools --> Tcl --> Execute Macro...) Le viewer vous permet de visualiser les registres del'ARM, de les comparer avec les résultats attendus (générés par le modèle ARM du test bench) et de faire du pas à pas (bouton RUN).