Rapport de projet de sysnum

Ryan Lahfa, Constantin Gierczak-Galle, Julien Marquet, Gabriel Doriath Döhler

Contents

0.1 Pipeline

Ce processeur implémente la pipeline standard en 5 étages : * Récupération de l'instruction * Décodage * Exécution * Accès mémoire * Écriture retour

Nous avons choisi d'optimiser la pipeline en mettant en place un système de forwarding entre les étages. Grâce à l'architecture RISC-V, il suffisait d'implémenter le forwarding entre l'étage EXE et les étages MEM et WB : chaque étage déclare l'éventuel registre dans lequel il écrit et les autres obtiennent (grâce à l'unité de forwarding) une vue sur l'état des registres qui correspond à ce que sera l'état des registres *après* que les étages suivant auront écrit leurs données.

> On pourrait éviter de relier l'étage WB à l'unité de forwarding, mais des contraintes techniques liées au timing sur Verilog nous ont poussés à choisir cette stratégie, même si elle implique d'utiliser un peu plus de circuits.

Nous avons implémenté un début de système de prédiction de branche – mais sans aller jusqu'à faire des statistiques à la volée sur l'exéction des programmes. Nous avons décidé de toujours prédire que les branches ne seront pas prises (ce qui était la solution la plus simple). En prévision des cas où nous nous trompons sur ces prédictions, nous avons ajouté dans le processeur un signal `KILL` qui sert à vider toute la pipeline.

L'ISA RISC-V est conçue pour permettre d'impélenter raisonnablement facilement cette pipeline, nous ne nous sommes donc pas heurtés à de trop gros problèmes (ce qui ne nous a cependant pas empêchés de passer quelques moments à nous battre contre Verilog pour comprendre comment faire exécuter les opérations logiques dans l'ordre prévu).

0.2 Synchronisation avec la mémoire

Comme nous voulions avoir un système de mémoire évolué, nous avons dû prévoir les cas où les appels à la mémoire prendraient un temps arbitrairement long.

Pour cela, nous avons un signal `STALL` qui bloque les étages `IF`, `ID`, `EXE` et `MEM` dans leur état courant en attendant un signal de type `ACK` de la part de la mémoire.

Ceci a aussi demandé d'implémenter une petite machine à états dans l'étage MEM : * Soit on suit une exécution normale * Soit on est en train d'attendre des données de la mémoire