Relatório - Trabalho Prático 2: Mips I4

Eugênio Pacceli Jonatas Cavalcante Lucas Augusto
Samuel Oliveira Victor Pires Diniz

15 de Novembro de 2015

Organização de Computadores 2 - 2º Semestre de 2015

1 Introdução

O segundo trabalho prático do semestre envolve a transformação do processador MIPS com *pipeline* implementado no trabalho anterior em um processador superescalar **I4**. Esse breve relatório pretende discutir os novos módulos implementados e as mudanças realizadas, entrando em detalhes sobre o funcionamento do processador e sobre as dificuldades encontradas no desenvolvimento.

2 Mudanças realizadas

A transformação do MIPS *pipeline* no I4 consiste em, basicamente:

- Dividir o estágio de execução em suas unidades funcionais
- Criar o novo estágio de *Issue*, que encaminha as instruções para as unidades funcionais corretas.
- Implementar uma *Scoreboard*, utilizada para manter controle sobre quais registradores estão sendo escritos e prevenir *hazards* no processador.
- Implementar a unidade de detecção de hazard em si, também utilizada dentro do *Issue*.

As mudanças acima implicam, também, em diversas outras mudanças no funcionamento interno do processador para garantir que a incorporação das novidades ocorra como esperado.

Além disso, foi modificado o módulo externo que lida com a interação entre o processador e as entradas e saídas da *FPGA*, para o momento da síntese do circuito na placa e teste prático.

2.1 Scoreboard

O scoreboard foi implementado e funciona como esperado, com duas interfaces assíncronas de leitura (para interação com o detector de hazard) e uma interface síncrona de escrita. Para testar seu funcionamento, o testbench `scoreboard_tb0.v` foi feito, e sua execução gera resultados corretos.

2.2 Detector de Hazard

O detector de hazard é um módulo assíncrono que recebe sinais do scoreboard e determina se deve ocorrer um stall no processador, caso a instrução nova provoque um hazard de dados. Sua funcionalidade é testada no testbench `hazarddetector_tb0.v`, que opera apropriadamente.

2.3 Divisão do estágio de execução

A divisão do estágio de execução foi, como proposto pela especificação, feita dividindo o módulo em três partes: multiplicação, operações em memória e outras operações (principalmente as operações lógico-aritméticas). Para isso, foram implementados os módulos `Mult.v`, `Mem.v` e `AluMisc.v`, respectivamente.

2.3.1 Mult.v

Essa unidade funcional lida com a operação nova de multiplicação, dividida, também, em quatro estágios, como proposto na especificação: *Mult_0.v*, *Mult_1.v*, *Mult_2.v* e *Mult_3.v*. Para testá-la, foi feito o testbench `mult_tb0.v`, que gera o resultado esperado.

2.3.2 Mem.v

Por sua vez, a unidade funcional *Mem.v* trata as instruções de leitura e escrita na memória, dividida em dois estágios reais (*Mem_0.v* e *Mem_1.v*) e dois estágios de encaminhamento para o ciclo seguinte, para garantir a execução em ordem.

2.3.3 AluMisc.v

Finalmente, *AluMisc.v* lida com as operações da ALU, do *shifter* e outras operações miscelâneas. Apenas um dos seus sub-estágios é realmente funcional, e, portanto, todo o código foi mantido dentro de apenas um arquivo, com os outros três estágios "falsos" dentro dele, também. O testbench `alumisc_tb0.v` testa brevemente sua funcionalidade e opera dentro do esperado para com sua saída.

2.4 Issue

O novo estágio de *Issue* recebe como entrada os sinais obtidos no *Decode* e instancia internamente o detector de *hazards* e o scoreboard. Ele é responsável por determinar qual unidade funcional corresponde a qual instrução e por repassar os sinais apropriados para cada unidade. Além disso, ele deve reagir apropriadamente aos sinais do detector de hazard, enviando o sinal de stall para os estágios anteriores, e também deve escrever no scoreboard quando a instrução atual realiza uma escrita de registrador.

2.5 Outras mudanças

Como comentado previamente, foi necessário alterar diversos sinais nos módulos antigos do processador para garantir o bom funcionamento das novidades. Em particular, o módulo de *Decode* e *Writeback* tiveram seus sinais de entrada e saída modificados, para enviar para o novo *Issue* e receber dados das unidades funcionais. No que cabe ao *Writeback*, foi necessário também permitir que ele recebesse de apenas uma das três unidades funcionais a cada ciclo, de forma mutuamente exclusiva.

3 Problemas que persistem

O processador ainda não funciona como um todo. Há problemas na interação entre o *Decode*, o *Issue* e o banco de registradores. No momento, há suspeitas de que isso ocorra devido à interação entre interfaces síncronas e assíncronas, mas não foi possível encontrar o bug ainda. Por esse motivo, não foi possível garantir ainda, também, o funcionamento correto do *Issue* e do *Decode*. Também por esse motivo não há certeza quanto à correção das modificações feitas no `mips_synth.v`, que lida com a síntese na FPGA.

4 Conclusão

Nesse relatório, foram detalhadas as mudanças realizadas no processador MIPS para transformá-lo num processador superescalar em ordem. O trabalho não foi concluído ainda com êxito, mas os problemas citados na seção anterior estão sendo resolvidos no momento, e a versão funcional será enviada assim que estiver pronta. Pedimos desculpas pelo atraso no envio dessa versão incompleta, assim como pelas dificuldades encontradas durante o processo de desenvolvimento e teste.