Arquitetura Computacional Pipeline de 32 bits em VHDL

HENRIQUE COLODETTI ESCANFERLA

Bacharelado em Ciência da Computação - Departamento de Informática

Universidade Federal do Paraná

hencolesc@gmail.com

Resumo. *Este meta-artigo tem como objetivo apresentar e descrever sobre todo um projeto, desenvolvimento e implementação de uma arquitetura de computador em blocos de 32 bits descrito na linguagem VHDL. Neste projeto, foi pedido instruções adicionais além das já implementadas, suporte para detecção de hazards e inserção de bolhas (instruções nulas), forwards para acelerar o desempenho do conjunto de instruções em execução e previsão em dois níveis de desvios com suporte aos hazards envolvidos.*

1. O Projeto

De início, foi dado uma implementação com intruções basicas de aritmética, manipulação de memória e saltos de instruções. Algumas modificações interessantes foram feitas antes de qualquer ação sobre os objetivos do projeto.

O cálculo do endereço de desvios, que estava no estágio EX foi transferido para o estágio ID e, para que isto fosse feito, também foi alterado o estágio do extensor e shifter do imediato da instrução do EX para o ID pois ele é usado no calculo do endereço de desvio. Isto permite o conhecimento dele um ciclo de relógio antes do que sem essa mudança. O sinal que indica desvio foi transferido do estágio MEM para o estágio EX e isto é de extrema importancia para diminuir hazards, e com isso, a quantidade de bolhas necessárias para contornar tais hazards.

2. Instruções ADDM e LUI

O desenvolvimento das tarefas foi iniciado pelas 2 intruções adicionais: addm e lui.

Começando pela lui, pois ela lhe permite inserir dados em registradores e, por meio das instruções LW e SW, em posições de memória, a implementação foi feita adicionando uma operação shift de 16 posições na ULA (Unidade Lógica Aritmética). Após shiftar o imediato da instrução, ele é guardado no registrador de destino no estágio WB.

A implementação da instrução addm foi bem simples. Um somador de 32 bits no estágio WB que soma o conteúdo do registrador RT e a saida da memória indexada por RT e, assim, é gravado o resultado no registrador RD de destino.

3. Fowards/Hazards

Existem vários casos os quais os forwards conseguem resolver completamente os hazards existentes, entretanto, existem aqueles que necessitam de forward e bolha para que a execução das instruções esteja coerente com o esperado.

Primeiramente, foram adicionados multiplexadores na entrada de dados da ULA, assim, qualquer resultado existente nos estágios MEM ou WB pode ser facilmente redirecionado para a ULA sem que precisemos esperar pela escrita no WB.

Um multiplexador foi colocado na entrada de dados da memória para resolver um hazard especifico e muito comum: cópia de memoria. Imagine um LW seguido de um SW, ambos interagindo com o mesmo registrador RT, os dados lidos no estágio MEM do LW estão, agora, no estágio WB enquanto que o que queremos gravar com o SW está ja na MEM com um valor antigo do registrador RT. Com o multiplexador de forward, isto é resolvido sem bolha alguma, de forma muito simples. Os dados do registrador RT são redirecionados do estágio WB para o estágio MEM e, assim, está feita a cópia de memória.

Mesmo com a ajuda dos forwards, ainda existem casos quando bolhas são necessárias. Foi implementado o congelamento do estágio ID quando é detectado hazard entre a instrução no estágio ID e aquela do estágio EX, e além disso, o congelamento do IF foi implementado somente impedindo o avanço de PC. Deste modo, a instrução do estágio EX segue em frente e as instruções nos estágios ID e IF esperam até que o sinal de bolha fique desativado, quando então, a execução volta ao normal.

3. Tempo gasto no Projeto

Aproximadamente, foi gasto 2 semanas na implementação do projeto. Uma boa parte dele, em torno de 70%, em testes e correção de erros/bugs. A linguagem VHDL demonstra ser de tipagem muito rígida e a apresentação dos erros, tanto de compilação quanto o diagrama de sinais gerado, dificulta o processo de execução mental e algorítmica do projeto em foco. Isto deixa o processo de debug muito mais trabalhoso do que o de costume comparado a outras linguagens.

3. Resultados

As instruções ADDM e LUI funcionam perfeitamente. Todos os forwards e hazards são devidamente tratados exceto pelo desvio de branchs. Neste caso, o PC se perde e a linha de execução perde a lógica.

A máquina de estados que preve desvios em dois níveis não apresentou o resultado esperado. Praticamente todos os sinais do diagrama ficaram zerados, indefinidos ou considerados lixo de memória. O pc caminha de forma erronea e nenhuma instrução sequer consegue chegar no estágio ID. O pc alterna entre posições válidas e inválidas em um padrão em loop fixo.

References

Patterson, D.A. e Hennessy, J.L. Organização e projeto de computadores - A interface hardware/software. 3a ed. ISBN 9788535215212 ou 4a ed. ISBN 9788535235852, Campus Elsevier

Ashenden, P. The Designer's Guide to VHDL, Third Edition. Morgan Kaufmann; 3 edition (May 29, 2008). ISBN 978-0120887859