Proposta e Implementação do Módulo de Tratamento de Hazard em Hardware para a Arquitetura MIPS Pipeline

Lucas Rocha[1]; Vinícius Santos[2] ; Nicholas Dutra[2];Rafael Soares[3]

[1]*Universidade Federal de Pelotas – {lrdrocha, vrdsantos, ndlihtnov, rafael.soares}@inf.ufpel.edu.br*

1. INTRODUÇÃO

Atualmente é cada vez maior o uso de sistemas computacionais presentes em dispositivos eletrônicos de consumo tais como smartphones, computadores desktops, tablets, GPS entre outros. Arquiteturas de hardware são compostas por processadores de propósito geral, memórias, dispositivos de entrada e saída de dados, além de módulos de hardware dedicados a funções específicas e barramentos de comunicação. Estas arquiteturas são programadas por meio de um conjunto de instruções e com isso oferecem uma estrutura flexível que pode ser usada para diferentes aplicações como por exemplo edição de texto, navegação na internet, execução de jogos entre outras.

O processador MIPS é muito utilizado pela indústria e também em pesquisas no ambiente acadêmico, principalmente por sua característica RISC e sua simplicidade de organização tornando-se muito didática tal como a versão monociclo capaz de realizar operações lógicas e aritméticas. Já a versão pipeline do MIPS retrata as operações do monociclo com o benefício de utilizar a técnica do Pipeline. Pipeline é uma técnica de implementação de processadores que permite um nível de paralelismo de execução de instruções por meio da sobreposição temporal das diversas fases da execução de instruções (PATTERSON; HENESSY,2000).

A técnica de pipeline aumenta o desempenho de execução de instruções, porém sofre com questões de dependências entra instruções também conhecidas na literatura por *hazards* ou conflitos. Estes conflitos surgem de situações em que a instrução seguinte não pode ser executada no próximo ciclo de relógio para não alterar o resultado final da execução. De acordo com (PATTERSON;HENESSY,2000) são três os tipos de hazards: estrutural, controle e de dados.

Neste trabalho dois tipos de hazard são abordados, o hazard de dados e o hazard de controle. O hazard de dados acontece quando a execução de uma instrução depende do resultado de outra que ainda está no pipeline (PATTERSON;HENESSY,2000). Hazard de dados é um conflito que ocorre devido a dependência de dados entre duas operações. O conflito decorre no momento em que o dado necessário para uma operação da ULA ainda está sendo propagado pelos estágios do Pipeline, estando indisponível naquele momento. A solução mais explorada nestes casos consiste na interconexão dos estágios, fazendo com que um resultado possa ser utilizado imediatamente na próxima operação da ULA.

Já o hazard de controle é um tipo de conflito conhecido como conflito de controle, originário da necessidade de se tomar um decisão com base nos resultados de uma instrução, enquanto outras estão sendo executadas (PATTERSON;HENESSY,2000). Hazard de Controle ocorre quando é necessário executar uma instrução BEQ (Branch on equal), porém é preciso esperar um ou mais ciclos de processamento até que a decisão possa ser tomada de fato. As soluções exploradas para o problema se baseiam na predição de salto, momento

que o processador MIPS tenta prever a decisão de instruções BEQ, com objetivo de realizar o maior número de saltos possível sem permanecer ocioso.

Este trabalho tem como objetivo propor e implementar em hardware uma solução para o hazard de controle e hazard de dados, usando para isso a linguagem de descrição de hardware VHDL.

2. METODOLOGIA

A solução proposta para o problema do hazard de dados, é o adiantamento, que conforme foi mencionado previamente no texto, envolve o redirecionamento de estágios anteriores de forma temporal, os dados ainda não salvos efetivamente, que são dados mais recentes no contexto de registradores.

$2 = 0
memory(0)<= Addi $2 $2 2
memory(1)<= *Add $3 $2 $2*
(Exemplo 1, caso de Hazard de dados)

Observando o Exemplo 1, as instruções consecutivas dentro do pipeline, que utilizam os registradores destino como operadores, como a instrução Add, possuem um hazard de dados, pois como efetivamente o conteúdo mais recente da instrução do registrador $2, ainda está em transito dentro dos estágios de pipeline, e não foi salvo de forma efetivo, portanto é preciso detectar duas instruções antes da instrução atual, se um dos registradores operandos, foi usado como registrador destino. Para isto foi criado um chamado detector de conflitos, que basicamente salva registradores destino de instruções passadas, para que seja comparado com registradores de instruções atuais, caso seja encontrado, é informado isto ao hardware de controle, que gera sinais, para dois multiplexadores, antes da ULA (unidade lógica aritmética), redirecionando os dados mais recentes. A solução visível no caminho de dados é mostrada na Figura 1 com a coloração vermelha, o componente de detecção de conflitos foi adicionado no controle, bem como o gerador de sinais para os multiplexadores.

A solução proposta para o problema do hazard de controle, é a chamada predição dinâmica, como foi mencionado o problema do hazard de controle, se concentra da temporalidade do teste de branch on equal,que revela se o salto dentro da memória será realizado através de uma comparação entre dois registradores operandos, que ocorre no terceiro estágio, e a deliberação para o componente PC(program counter), só ocorre no quarto estágio, onde caso o programador seja descuidado, ou o compilador ignore este fato, pode ocorrer uma quebra temporal grave dentro do programa executado.

memory(0)<=Beq $0 $0 10
memory(1)<=addi $1 $0 1
memory(2)<=addi $2 $0 2
(Exemplo 2, caso de hazard de controle)

Observando o Exemplo 2 acima de teste, é possível observar que a comparação entre o mesmo registrador, sempre resultará em um pulo, ou seja há duas instruções que serão computadas antes, de ocorrer a atualização do PC, sendo que se for utilizado os registradores $1 ou $2, o programa resultará em um resultado indesejado. É necessário salientar que para programas onde se deseja grande desempenho, não é desejável utilizar a instrução NOP (No operation), que basicamente faz com que o processador fique parado por um ciclo,

Então a solução de predição dinâmica une duas soluções, a solução de predição estática onde sempre ocorrerão os saltos e a solução onde nunca ocorrerão os saltos. Unindo ambas a solução é possível realizar um processador adaptativo a o tipo de programa, caso haja vários saltos bem-sucedidos, será ativado no caminho de dados a solução onde sempre ocorrerá o salto, caso haja em sua maioria saltos onde não são bem-sucedidos haverá será ativada no caminho de dados a solução de nunca ocorrerá o salto. Estás opções são controladas através de um componente que colhe resultados de instruções BEQ, desta forma, ele ativará no caminho de dados a solução que melhor se encaixa na necessidade.

3. RESULTADOS E DISCUSSÃO

A jornada de pesquisa é experienciar e validar, que o tratamento de controle opere para todos os casos possíveis e implementar algoritmos que explorem os hazards de dados e de controle de uma forma conjunta.

É possível observar na Figura 1, a solução de hazard de controle, por predição dinâmica, na cor azul, não é possível observar o componente que controla a solução pois o mesmo está dentro do componente de controle do Mips.

Foi utilizado a ferramenta Quartus 16.0.0 Build 211 lite Edition para a construção em hardware a fim de gerar a solução proposta de hazards de dados e controle.

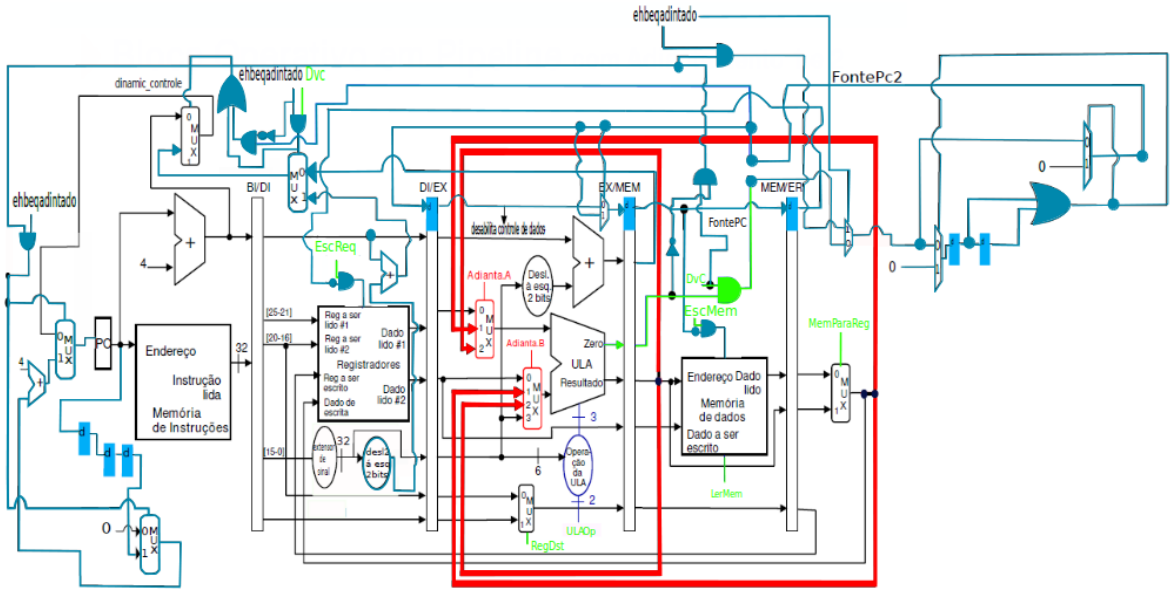

Figura 1: Estrutura do processador MIPS remodelada para a resolução das hazards.
modificado de (PATTERSON;HENESSY,2000)

O testbench usado para a verificação da solução é apresentado como segue:

– Addi $1 $1 3
– Add $1 $1 $1 (8 vezes executada).

As formas de onda da execução são mostradas no ModelSim na Figura 2.

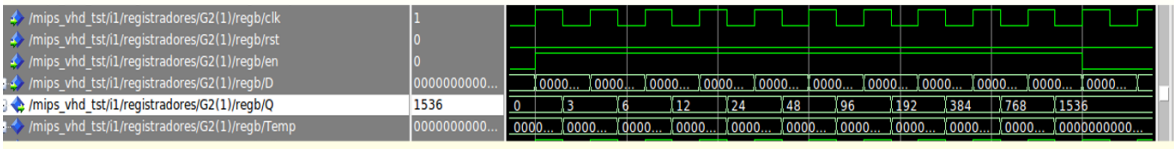

Figura2: Formas de onda da geração de um testbench.

Fica claro que houve em somente um ciclo de relógio o salvamento integral da soma, que depende dos dados gerados pela execução da instrução anterior. Esta execução específica não é capaz de validar a arquitetura para todos os casos, entretanto, apresenta-se esta execução como prova parcial do funcionamento da arquitetura.

Esta solução proposta em hardware, trará um tratamento de hazards de dados e de controle, que abranja 100% em todos os modelos passiveis de tratamento.

Os resultados parciais são detalhados na Tabela 1:

	Mips Pipeline Original	Mips Pipeline Proposto
Frequência	91.79MHz	73.86MHz
Número de elementos lógicos	1.508/6.272(43%)	2.708/6.272(43%)
Funções combinadas totais	1.433/6.272(36%)	2.242/6.272(36%)
Registradores lógicos dedicados	499/6.272(20%)	1.280/6.272(20%)
Total de registradores	1280	1280
Total de pinos	34/92(37%)	34/92(37%)
Pinos virtuais totais	0	0
Total de bits de memória	33/276.480(<1%)	33/276.480(<1%)
Multiplicador incorporado	0/30(0%)	0/30(0%)
Total PLLS	0/2(0%)	0/2(0%)

Tabela1: Resultados parciais obtidos.

4. CONCLUSÕES

“Existem menos nisto do que aquilo que os olhos percebem”, esta frase dita por Tallulah Bankhead, numa alusão a Alexandre Wollcott, em 1922 é uma forma de antagonizar o que foi proposto nesta pesquisa, estes dois problemas dos quais foi investigado um desfecho final para os mesmos, acarretam muitos problemas dentro da comunidade computacional, por isso foi apresentado uma solução em hardware para um dos problemas da arquitetura mips pipeline, que são os hazards de dados e controle.

Como a presente pesquisa não se encontra finalizada, ainda se torna incapaz de validar a arquitetura para uma sucessão de instruções, no entanto a pesquisa apresenta estimativas positivas dentro dos tópicos esperados.

Desta forma, espera-se futuramente comprovar testes em multiplos cenários, validando o modelo sobre a geração de área, frequência, consumo de energia e dissipação de potência.

5. REFERÊNCIAS BIBLIOGRÁFICAS

PATTERSON, D. A.; HENESSY, J. L. **Organização e Projeto de Computadores: a interface hardware/software**.2a.ed. Rio de Janeiro: Livros Técnicos e científicos Editora S.A. , 2000. 551p.

BUNCHHOLZ, W; **Planning a Computer System : Project Stretch.** 1a,ed.:Mcgraw-Hill Book Company,Inc,1962.322p.