IMPLEMENTAÇÃO DO MIPS PIPELINE

GUILHERME DAVESAC GOEBEL[1]; HUBERTO KAISER FILHO[2]; BRUNO ZATT[3]

[1]*Universidade Federal de Pelotas – gdgoebel@inf.ufpel.edu.br*
[2]*Universidade Federal de Pelotas – hkaiser@inf.ufpel.edu.br*
[3]*Universidade Federal de Pelotas – zatt@inf.ufpel.edu.br*

1. INTRODUÇÃO

A implementação do processador MIPS Pipeline Multi-ciclo a ser discutida neste artigo, como qualquer projeto em hardware está dividida em bloco operativo e bloco de controle. O principal objetivo do trabalho desenvolvido é desenvolver um hardware com suas especificações bem definidas e com sua codificação de fácil entendimento, para que possa ser utilizado em futuros trabalhos a serem desenvolvidos em grupos de pesquisa da comunidade acadêmica. Suas unidades são interconectadas através de um arquivo principal, o “MIPS.vhd”, que ligara cada componente do circuito e realiza o mapeamento dentro de cada estágio do MIPS Pipeline Multi-ciclo. Cada estágio do MIPS executa uma tarefa em específico, seja cálculo do endereço ou execução. A instrução deve passar por todos os estágios para que seja processada corretamente, com exceções de algumas instruções as quais não necessitam de alguns estágios como por exemplo Jump e Branchs.

Apresentamos a seguir como construímos cada estágio e bloco do projeto. Para a síntese desta implementação utilizaremos a FPGA da marca Altera família Cyclone II, dispositivo EP2C35F672C6. Como este dispositivo possui um número limitado de bits de memória, limitamos a nossa ´ implementação para utilizar memorias de 2^{10} posições (pois tradicionalmente o MIPS possui memória de 2^{32} posições).

A ideia principal do MIPS Pipeline Multi-ciclo em relação a versões anteriores como a Monociclo ou até mesmo o Neander, é otimizar o máximo do hardware disponível, paralelizando as operações dentro de um mesmo ciclo de clock. Tomando como exemplo a operação Load Word, esta versão permite que enquanto esteja sendo feito o acesso à memória para a captura da informação, os demais estágios (que estariam ociosos numa implementação monociclo) sejam utilizados para a execução de outras instruções subsequentes.

2. METODOLOGIA

Para este projeto o MIPS Pipeline Multi-ciclo divide-se em 5 estágios como pode ser visto na Figura 1, busca de instrução, decodificação da instrução, execução e leitura e escrita na memória de dados e por último a escrita no banco de registradores. Como a implementação a ser abordada utiliza Pipeline para o desenvolvimento, ao invés de utilizar vários registradores, preferimos o uso de um único registrador, do tipo flip-flop, de tamanho equivalente a soma de todos os bits de transição do estágio anterior. Esta abordagem deixa claro que o registrador flip-flop em questão é um componente do pipeline, dificultando assim confundir com outros registradores.

A busca de instruções ocorre no primeiro estágio, é responsável pelo cálculo do próximo endereço (PC + 4) que será o ponto de partida para sabermos qual instrução o MIPS deve executar. Por padrão, o PC é iniciado com o endereço zero, e como nosso tipo de implementação é endereçada a byte, o PC fará a

soma de PC + 4. Então, este mesmo endereço será utilizado pela memória de instruções que saberá qual instrução retornar para disponibilizar para os próximos ciclos e assim processar tal instrução. A divisão dos bits de endereço (exceto o "opcode") irá variar de acordo com o tipo da instrução requerida, seja ela do tipo R, I ou J.

O segundo passo é a decodificação da instrução, a saída deste estagio será dividida em vários vetores como visto a seguir: (i) Opcode; (ii) Registradores a serem lidos (dois); (iii) Imediato para instruções do tipo I ou Função para instruções do tipo R; (iv) Registrador de destino da instrução. A unidade de controle como pode ser visto na Figura 1 do MIPS funciona em conjunto com cada um destes estágios e é definida no segundo Estágio. Ela recebe 6 bits de chamados de Opcode, este componente tem como função gerar sinais de controle para cada estágio. Para isso o componente utiliza o Opcode para gerar sinais específicos para cada tipo de instrução, seja ela R, I ou J. O banco de registradores como pode ser visto na Figura 1, possui um mapeamento semelhante à da memória. Possui 32 registradores disponíveis para leitura e escrita, ele recebe endereços os quais serão lidos pelo bando e retornados os seus respectivos valores. Ele também possui a capacidade de escrever em algum registrador presente no banco, para isso é necessário que habilite o sinal de escrita e ao mesmo tempo enviar o endereço do registrador a ser escrito e o seu respectivo dado. Vale lembrar que este banco de registradores tem a capacidade de ler e escrever ao mesmo tempo, ou seja, no mesmo ciclo de clock, possibilitando assim que um dado que esteja sendo escrito possa ser lido no mesmo instante. Neste mesmo estágio ainda possuímos um componente o qual serve como extensão de sinal para o imediato, pois o MIPS trabalha sempre com instruções de 32 bits e ela possui apenas 16 bits (Figura 1).

O terceiro estágio do pipeline é voltado para a execução da instrução lida anteriormente. A ULA presente neste estágio, é responsável por efetuar qualquer tipo de processamento definido pelo opcode. Para isso ela recebe um sinal proveniente de um componente chamado Operação da ULA, ele recebe dados de controle e juntamente com a Função define qual operação deve ser realizada sobre aqueles dados. Como sinal de saída, ela possui do dado resultante da operação e um sinal de controle chamado de zero, o qual e utilizado para definição dos Branchs. Ainda nesse ciclo e tratado as instruções de Branchs e Jumps. Tais instruções são referentes ao PC + 4 da instrução, logo, o imediato o qual possui quantas posições devem ser puladas e deslocado duas vezes para esquerda então é somado com o PC + 4 através de um somador para que seja definido qual posição deve executar no próximo ciclo. Também neste estágio, através de outro multiplexador é feito a escolha de qual registrador de escrita será utilizado, esta escolha se dá com base no tipo de instrução podendo ser assim proveniente da instrução R a qual possui um espaço reservado para esta finalidade ou do tipo I que utiliza o espaço de um dos registradores de leitura.

Esta quarta etapa tem como objetivo o armazenamento do dado na memória de dados. Esta etapa não e necessária para todas as instruções, caso seja necessário o dado proveniente da ULA será o endereço de escrita e o dado em si e passado por um outro sinal. A memória descrita neste estágio como pode ser observado na Figura 1 possui 2_{10} posições e é endereçada a byte.

O quinto e último estágio e responsável pela escrita dos registradores. Através de um multiplexador o controle do MIPS escolhe qual dado deve ser enviado para o registrador, podendo ser escolhido entre um dado proveniente da memória ou um dado previamente declarado na instrução ou registrador de destino.

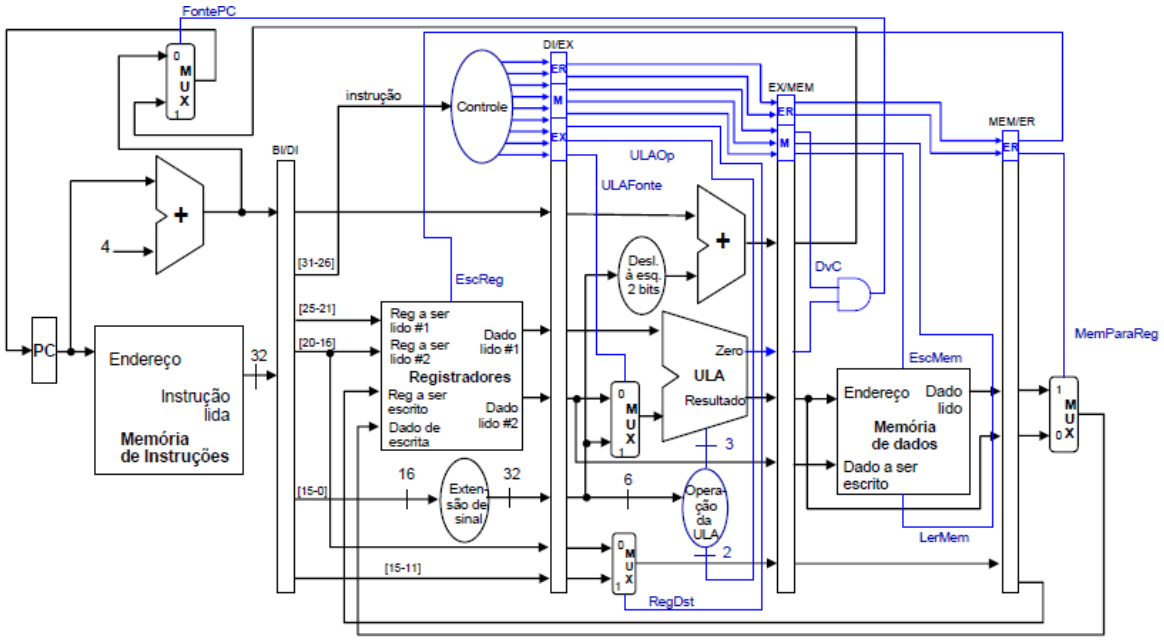

Figura 1: Arquitetura implementada.

3. RESULTADOS E DISCUSSÃO

Para esta implementação, foram escolhidas algumas instruções como Beq, addi, lw, sw, j, sll, srl, sra, add, sub, and, or, xor, nor slt e sltu. Vale lembrar que para instruções do tipo R, o Opcode é padrão ”000000” então o que define a instrução é o sinal Função. As demais instruções as quais não foram especificadas neste projeto, poderão ser implementadas como uma proposta de trabalho futuro. A arquitetura descrita neste projeto foi projetada pensando em trabalhos futuros possibilitando assim, que sejam adicionadas novas funcionalidades sem que seja necessárias grandes mudanças na estrutura da arquitetura, tornando assim a inclusão de novas instruções fácil, limitando a complexidade apenas para instrução em si. A implementação do MIPS Pipeline no modo Multi-Ciclo consiste em um grande ganho para o desempenho de processadores. A estabilidade das operações permite reduzir em mais da metade do tempo uma sequência de instruções a serem executadas.

Segue abaixo a tabela com os valores dos elementos utilizados e sua porcentagem total da FPGA que foi usada para simulação.

Descrição	Valor	Utilização da FPGA
Total de elementos lógicos	1576	5%
Total Funções Combinacionais	1448	4%
Registradores Dedicados	499	2%
Pinos Utilizados	34	7%
Total de Bits de memória	32768	7%
Frequência máxima obtida	77.45 MHz	7.74%

4. CONCLUSÕES

Com os resultados obtidos sendo os esperados, concluímos que o projeto implementado está de acordo com o planejado e completamente funcional, deixando assim este trabalho disponível para professores e alunos usufruírem.

Como sugestão, é possível usar este projeto como base e implementar as demais instruções que compõe o MIPS. Além disso, a leitura para o registrador de instruções no estágio 1 é feita a partir de dados inicializados na própria implementação VHDL, no entanto criar um arquivo de texto externo ao código no qual fossem feitas leituras, estaria muito mais próximo de aplicações reais e de como o MIPS funciona na pratica. Um problema recorrente quando se trabalha com implementações Pipeline, é o conhecido “hazard” ou conflito que pode ser de dados ou paradas ou então no bloco de controle (de desvios condicionais). Tais conflitos seriam resolvidos em uma implementação que ao detectar o conflito inserisse um adiantamento, em outros casos uma “bolha” que executaria uma instrução NOP ou ainda no controle fazer o descarte de instruções (flushed).

5. REFERÊNCIAS BIBLIOGRÁFICAS

Livro

PATTERSON, D. A. and Hennessy, J. L. **Organização e projeto de computadores: a interface hardware/ software,** Rio de Janeiro: CAMPUS, 2005.

Documentos eletrônicos

Cheriton School of Computer Science, **MIPS Encoding Reference**, 2015 acessado 1 de Junho, 2015, disponível em: <https://www.student.cs.uwaterloo.ca/~isg/res/mips/opcodes>