Processador Didático de 8 bits em FPGA: Uma Ferramenta para o Ensino Prático em Arquitetura de Computadores

Pedro M. Botelho, Roberto. C. Rabêlo, Thiago W. Bandeira

[1]Universidade Federal do Ceará (UFC) - Av. José de Freitas Queiroz, 5003, 63902-580
Cedro – Quixadá – Ceará - Brasil

Abstract. *In this paper, we present the design and VHDL implementation of an 8-bit didactic processor. The processor is based on the RISC philosophy and is designed to be simple, facilitating its use as a study tool by students and teachers, in Computer Architecture courses, as well as in other related courses.*

Resumo. *Neste trabalho, apresentamos o projeto e a implementação em VHDL de um processador didático de 8 bits. O processador é baseado na filosofia RISC e foi projetado para ser simples, facilitando seu uso como ferramenta de estudo por alunos e professores, em cursos de Arquitetura de Computadores, assim como em outros cursos relacionados.*

1. Introdução

Desde o início da era digital, o principal objetivo tem sido construir máquinas mais poderosas e otimizadas. Para poder entender o funcionamento de computadores mais avançados (ou até mesmo projetá-los) é necessário primeiro compreender modelos mais simples. Assim, surge a necessidade de um modelo básico que possibilite o ensino eficiente de computadores, capacitando os alunos a compreender não apenas a teoria, mas também a prática.

O ponto-chave de um modelo didático é sua simplicidade, e a linguagem VHDL facilita bastante a implementação. A implementação em VHDL do modelo Neander [de Abreu 2014], e o projeto e implementação de um modelo de 4-bits em VHDL para a implantação em um curso de Microprocessadores [Santa 2017] fornecem marcos interessantes no desenvolvimento de ferramentas didáticas.

A construção de um modelo específico para o ensino oferece maior flexibilidade, permitindo a personalização de sua estrutura de acordo com as necessidades de alunos e professores. Dito isso, este artigo aborda o projeto e a implementação, em VHDL, de um processador de 8 bits denominado CRISC, desenvolvido para ser suficientemente simples e, ao mesmo tempo, útil como ferramenta de ensino nas disciplinas de Arquitetura de Computadores e áreas correlatas. São detalhados os passos seguidos e as decisões tomadas durante o projeto, visando criar um processador que seja simultaneamente simples e flexível.

2. Trabalhos Relacionados

A necessidade de criar um modelo educacional suficientemente simples para o ensino de processadores é antiga, existindo vários trabalhos voltados para isso que trazem a tona diferentes características para impulsionar o aprendizado.

Em [de Abreu 2014] é realizada a implementação, em VHDL, do Neander, um modelo de processador de 8-bits criado na UFRGS especialmente para o ensino. Esse modelo não tinha uma implementação real até então, tendo apenas 11 instruções simples, uma arquitetura baseada em acumulador e instruções simples onde o único operando é um endereço da memória.

Em [Santa 2017] é abordado o projeto e implantação de um processador didático de 4-bits, usando VHDL. Motivado pela complexidade dos processadores geralmente utilizados no ensino, sendo muito completos, impedindo que os alunos modifiquem sua estrutura e funcionamento, e pela alta taxa de reprovação na disciplina de microprocessadores, foi criado um processador minimalista e modular.

Em [Nascimento 2006] é discutida a necessidade de um ensino prático de hardware, que normalmente exige componentes caros e complexos. Para superar essas dificuldades, o projeto utiliza a tecnologia de FPGA para desenvolver e implementar um processador em hardware real. Isso possibilita que estudantes aprendam de maneira prática, utilizando uma arquitetura de processador semelhante à MIPS, uma referência conhecida na área, facilitando a compreensão de processadores comerciais.

3. Fundamentação Teórica

O trabalho apresentado é uma dissertação a respeito do projeto de um processador simples. Processadores, por mais complexos que sejam, sempre terão uma estrutura minimamente semelhante.

Um **computador** é a união de vários componentes (processador(es), memória(s) e periférico(s) de entrada e saída) para o processamento de dados. O coração do computador é o **processador**, ou unidade central de processamento (CPU), que interpreta e executa **instruções**, que representam as operações aritméticas, lógicas, de controle e de E/S que o processador realiza, podendo processar vários bits de uma vez (palavra de dados). O processador é composto por duas subunidades: *datapath* e unidade de controle.

É no **datapath** que os dados são processados, sendo literalmente o caminho dos dados que os dados percorrem no decorrer do processamento. Contém a **unidade lógica e aritmética** (ULA), responsável pelo processamento dos dados por meio de operações lógicas e aritméticas. Os dados processados geralmente ficam armazenados em **registradores**, os quais são espaços de armazenamento internos ao processador.

Já a **unidade de controle** gerencia as operações no sistema por meio de sinais de controle, tanto internos (ex.: qual registrador será lido) quanto externos (ex.: escrever ou ler na memória). Ainda, decodifica as instruções para que os sinais corretos possam ser enviados para os diversos componentes do sistema, de forma a executá-la no *datapath*.

As **memórias** são componentes essenciais em um sistema computacional, responsáveis pelo armazenamento de dados e instruções que o processador utiliza, que podem estar no mesmo dispositivo de memória (arquitetura de Von Neumann)ou em dispositivos separados (arquitetura de Harvard).

Existem várias metodologias de projeto para processadores simplificados, sendo a mais conhecida a filosofia **RISC** (computador com conjunto de instruções reduzido), que possui algumas instruções simples executadas em um único ciclo de ***clock***, sendo o sinal de sincronização do sistema, simplificando o projeto.

Para implementar o processador podem ser usados componentes digitais discretos, ou pode ser usado uma linguagem de descrição de hardware (HDL), como o **VHDL**, que permite a descrição de todo o projeto do processador de maneira programática. Dessa forma é possível simular o projeto, ou sintetizá-lo e gravá-lo em uma **FPGA**, sendo um hardware reconfigurável via HDL.

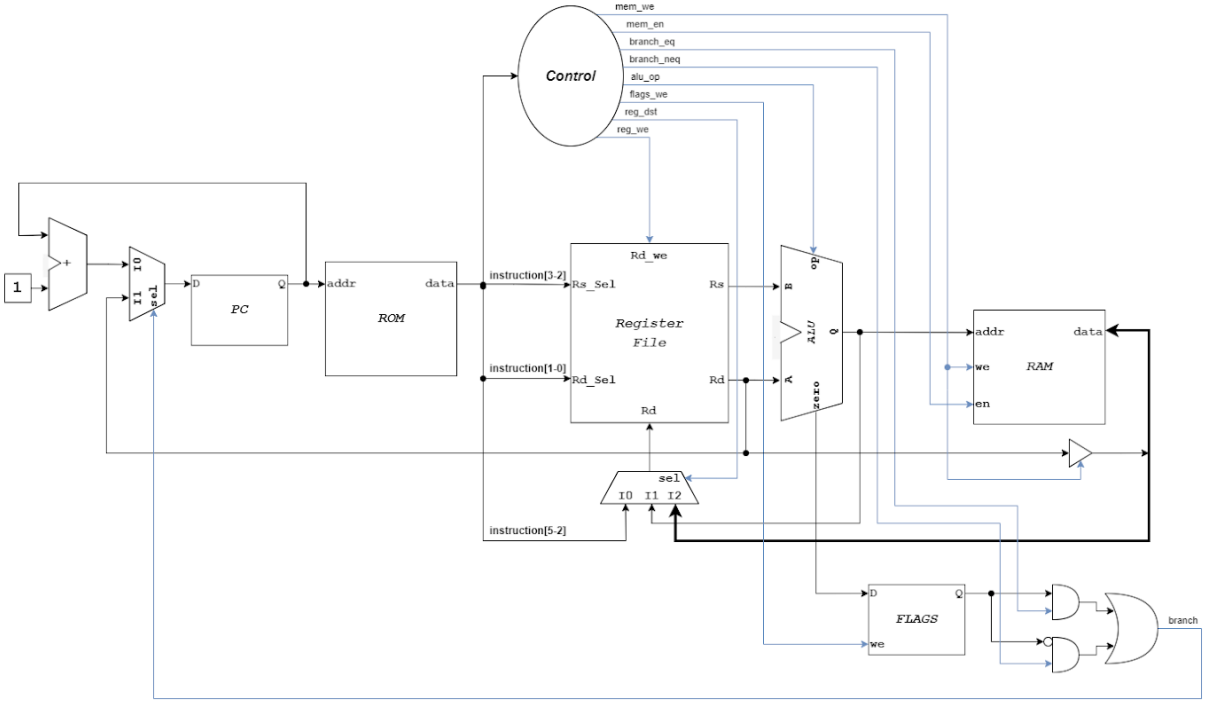
4. Procedimentos Metodológicos

Como dito anteriormente, o objetivo desse trabalho é explicitar os passos do projeto de um processador para o ensino de arquitetura de computadores. Para esse fim, esse artigo foi separado em quatro métricas, como mostrado a seguir:

4.1. Especificações do Projeto

O projeto de um processador deve cumprir uma série de requisitos específicos para alcançar algum fim específico. De forma a simplificar, foram usadas características da filosofia RISC que visam a facilidade de projeto, entendimento e uso. Com base nesses requisitos, o modelo CRISC (*Compact* RISC) foi projetado conforme mostra a Figura 1.

Figura 1. Diagrama do Processador CRISC



Estabeleceu-se o tamanho da palavra de dados, sendo de 8 bits (para dados e instruções). Para permitir um modelo monociclo (uma instrução por ciclo de *clock*) foram usadas instruções simples e Arquitetura de Harvard. Ainda, quatro registradores são acessíveis e as instruções possuem no máximo dois operandos (destino e origem).

4.2. Instruções do Processador

O processador precisa de instruções para realizar operações. Como visto na Seção 4.1 essas instruções devem ser simples e devem ter tamanho de 8 bits. Com base nessas

restrições, foram projetadas 16 instruções básicas para o processador, como mostra a Tabela 1.

Tabela 1. Instruções do Processador

Instrução	Descrição	7	6	5	4	3	2	1	0
HALT	Para o processador	0	0	0	0	0	0	0	0
MOV Rd, Rs	Rd <= Rs	0	0	0	1	Rs_1	Rs_0	Rd_1	Rd_0
ADD Rd, Rs	Rd <= Rd + Rs	0	0	1	0	Rs_1	Rs_0	Rd_1	Rd_0
SUB Rd, Rs	Rd <= Rd + Rs	0	0	1	1	Rs_1	Rs_0	Rd_1	Rd_0
AND Rd, Rs	Rd <= Rd AND Rs	0	1	0	0	Rs_1	Rs_0	Rd_1	Rd_0
OR Rd, Rs	Rd <= Rd OR Rs	0	1	0	1	Rs_1	Rs_0	Rd_1	Rd_0
SHL Rd, Rs	Rd <= Rd <<Rs	0	1	1	0	Rs_1	Rs_0	Rd_1	Rd_0
SHR Rd, Rs	Rd <= Rd >>Rs	0	1	1	1	Rs_1	Rs_0	Rd_1	Rd_0
NOT Rd	Rd <= ¬Rd	1	0	0	0	0	0	Rd_1	Rd_0
B Rd	PC <= Rd	1	0	0	0	0	1	Rd_1	Rd_0
BEQ Rd	Se Z, PC <= Rd	1	0	0	0	1	0	Rd_1	Rd_0
BNE Rd	Se Z, PC <= Rd	1	0	0	0	1	1	Rd_1	Rd_0
LDR Rd, Rs	Rd <= mem(Rs)	1	0	0	1	Rs_1	Rs_0	Rd_1	Rd_0
STR Rd, Rs	mem(Rs) <= Rd	1	0	1	0	Rs_1	Rs_0	Rd_1	Rd_0
CMP Rd, Rs	Z <= Rd - Rs	1	0	1	1	Rs_1	Rs_0	Rd_1	Rd_0
LI Rd, X	Rd <= X	1	1	X_3	X_2	X_1	X_0	Rd_1	Rd_0

Apenas as instruções mais simplistas foram incluídas, de forma a facilitar a implementação e o uso. Os dois endereços iniciais da memória de dados foram mapeados para escrever em LEDs (saída) e ler de botões (entrada) respectivamente, portanto não foram necessárias instruções extras.

4.3. Implementação em VHDL

O modelo foi implementado em VHDL usando a ferramenta Vivado, que permite a edição, simulação e gravação do projeto. O código foi implementado tendo em mente a leitura e possível modificações pelos estudantes e professores. Portanto, o projeto é modularizado, podendo ser encontrado em um repositório do Github [Botelho 2024b].

Os módulos foram feitos usando arquiteturas híbridas, isso é, alguns módulos desenvolvidos de forma estrutural (o módulo é composto por outros módulos) e outros de forma comportamental (o comportamento do módulo depende de seus comandos), e outros usando ambos. Isso permite uma rápida modificação e leitura.

4.4. Gravação na FPGA

O projeto em VHDL pode ser simulado usando a ferramenta Vivado (permitindo uma rápida e flexível verificação de seu funcionamento), mas sua maior vantagem é a gravação em uma FPGA física.

O código foi projetado de forma a permitir a gravação em uma FPGA (foi usada a placa Zybo), possibilitando a verificação do andamento do programa usando LEDs e botões da placa mapeados na memória do processador. Dessa forma, estudantes e professores podem usar uma implementação real do processador em seus estudos.

5. Resultados

A implementação do processador foi polida de forma a não apresentar falhas, tendo sido submetida a vários testes virtuais e físicos, de forma a verificar seu funcionamento.

Na simulação foi visualizado que os sinais gerados pela unidade de controle conforme a instrução sendo executada realiza o controle esperado no *datapath*, dessa forma chegando aos resultados corretos. O principal objetivo dos testes virtuais foi ter certeza que todas as instruções são decodificadas e executadas de maneira correta.

Ao executar sobre a FPGA foi verificado que o programa consegue ler os botões, processar os dados e escrever nos LEDs de acordo. O principal objetivo dos testes físicos foi assegurar que o projeto VHDL é completamente sintetizável (consegue descrever um *hardware* funcional), de forma que as instruções executadas funcionem de maneira adequada e que os periféricos possam ser acessados.

Ao final, o completo funcionamento do modelo foi assegurado, podendo ser usado por alunos e professores em disciplinas relacionadas a arquitetura de computadores em conjunto com o *Assembler* CASM [Botelho 2024a], de forma a facilitar a programação.

6. Conclusão

Com o estudo da computação cada vez mais essencial atualmente, compreender o funcionamento dos computadores é uma competência básica para o profissional de TI, sendo esta sua principal ferramenta. Dessa forma, a utilização de um modelo simplificado para o ensino de computadores revela-se crucial para um bom entendimento da teoria e prática.

Não só na arquitetura de computadores, mas em várias disciplinas é necessário entender o funcionamento de processadores. Assim, essa ferramenta possui ampla aplicabilidade, auxiliando no ensino por meio de sua estrutura e instruções simples.

Sendo um trabalho em andamento, já tendo sido projetado o processador e o montador, o próximo passo é avaliar o uso dessa ferramenta nas disciplinas de Arquitetura de Computadores e relacionadas, como Microcontroladores e Sistemas Digitais, para analisar o impacto do uso dessa ferramenta.

No geral, este trabalho serve como base para outros projetos de maior complexidade, como, por exemplo, modelos de 32 bits com várias instruções, periféricos mais complexos (e.g. temporizadores), interrupções e até mesmo processadores com pipeline, podendo ser então um marco importante no estudo de computadores.

Referências

Botelho, P. (2024a). Repositório do montador. https://github.com/botelhocpp/crisc_assembler.

Botelho, P. (2024b). Repositório do processador. https://github.com/botelhocpp/crisc_vhdl.

de Abreu, M. A. L. (2014). Implementação de processadores didáticos utilizando linguagem de descrição de hardware.

Nascimento, V. L. (2006). Projeto e implementação de um processador em fpga para ensino de arquitetura de computadores.

Santa, F. M. (2017). Minimalist 4-bit processor focused on processors theory teaching.