TRABALHO DE GRADUAÇÃO

RISC-V SiMPLE

Arthur de Matos Beggs

Brasília, Dezembro de 2019

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

RISC-V SiMPLE

Arthur de Matos Beggs

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Marcus Vinicius Lamar, CIC/UnB _____
Orientador

Profa. Carla M. C. C. Koike, CIC/UnB _____
Examinador Interno

Prof. Marcelo Grandi Mandelli, CIC/UnB _____
Examinador Interno

Brasília, Dezembro de 2019

FICHA CATALOGRÁFICA

ARTHUR, DE MATOS BEGGS
RISC-V SiMPLE,

[Distrito Federal] 2019.

???, ???p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2019). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1. RISC-V
2. Verilog
3. FPGA

I. Mecatrônica/FT/UnB
II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

BEGGS, ARTHUR DE MATOS, (2019). RISC-V SiMPLE. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-n°???, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, ???p.

CESSÃO DE DIREITOS

AUTOR: Arthur de Matos Beggs

TÍTULO DO TRABALHO DE GRADUAÇÃO: RISC-V SiMPLE.

GRAU: Engenheiro
ANO: 2019

Arthur de Matos Beggs

SHCGN 703 Bl G Nº 120, Asa Norte

70730-707 Brasília – DF – Brasil.

Dedicatória

Arthur de Matos Beggs

Agradecimentos

Arthur de Matos Beggs

RESUMO

Desenvolvimento e documentação de uma plataforma de ensino de arquitetura de computadores em *Verilog* sintetizável em *FPGA*, com foco em um processador com arquitetura do conjunto de instruções *RISC-V* implementado em três microarquiteturas para ser utilizado como recurso de laboratório na disciplina de Organização e Arquitetura de Computadores da Universidade de Brasília. A plataforma funciona em dois modelos de *FPGA* disponíveis no laboratório da Universidade, possui periféricos de depuração como *display* dos registradores do processador na saída de vídeo, além de outros periféricos como *drivers* de áudio e vídeo para uma experiência mais completa de desenvolvimento, e permite que o processador seja substituído por implementações de diversas arquiteturas de *32 bits* com certa facilidade.

Palavras Chave: RISC-V, Verilog, FPGA

ABSTRACT

Keywords: RISC-V, Verilog, FPGA

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

ASIC	Circuito Integrado de Aplicação Específica — *Application Specific Integrated Circuit*
BSD	Distribuição de Software de Berkeley — *Berkeley Software Distribution*
CISC	Computador com Conjunto de Instruções Complexo — *Complex Instruction Set Computer*
CSR	Registradores de Controle e Estado — *Control and Status Registers*
DSP	Processamento Digital de Sinais — *Digital Signal Processing*
FPGA	Arranjo de Portas Programáveis em Campo — *Field Programmable Gate Array*
hart	*hardware thread*
ISA	Arquitetura do Conjunto de Instruções — *Instruction Set Architecture*
MIPS	Microprocessador sem Estágios Intertravados de *Pipeline* — *Microprocessor without Interlocked Pipeline Stages*
OAC	Organização e Arquitetura de Computadores
PC	Contador de Programa — *Program Counter*
PLL	Malha de Captura de Fase — *Phase-Locked Loop*
RISC	Computador com Conjunto de Instruções Reduzido — *Reduced Instruction Set Computer*
SDK	Conjunto de Programas de Desenvolvimento — *Software Development Kit*
SoC	Sistema em um Chip — *System on Chip*
SiMPLE	Ambiente de Aprendizado Uniciclo, Multiciclo e *Pipeline* — *Single-cycle Multicycle Pipeline Learning Environment*
RAS	Pilha de Endereços de Retorno — *Return Address Stack*

Capítulo 1

Introdução

1.1 Motivação

O mercado de trabalho está a cada dia mais exigente, sempre buscando profissionais que conheçam as melhores e mais recentes ferramentas disponíveis. Além disso, muitos universitários se sentem desestimulados ao estudarem assuntos desatualizados e com baixa possibilidade de aproveitamento do conteúdo no mercado de trabalho. Isso alimenta o desinteresse pelos temas abordados e, em muitos casos, leva à evasão escolar. Assim, é importante renovar as matérias com novas tecnologias e tendências de mercado sempre que possível, a fim de instigar o interesse dos discentes e formar profissionais mais capacitados e preparados para as demandas da atualidade.

Hoje, a disciplina de Organização e Arquitetura de Computadores da Universidade de Brasília é ministrada utilizando a arquitetura *MIPS32*. Apesar da arquitetura *MIPS32* ainda ter grande força no meio acadêmico (em boa parte devido a sua simplicidade e extensa bibliografia), sua aplicação na indústria tem diminuído consideravelmente na última década.

Embora a curva de aprendizagem de linguagens *Assembly* de alguns processadores *RISC* seja relativamente baixa para quem já conhece o *Assembly MIPS32*, aprender uma arquitetura atual traz o benefício de conhecer o *estado da arte* da organização e arquitetura de computadores.

Para a proposta de modernização da disciplina, foi escolhida a *ISA RISC-V* desenvolvida na Divisão de Ciência da Computação da Universidade da Califórnia, Berkeley como substituta à *ISA MIPS32*.

1.2 Por que RISC-V?

A *ISA RISC-V* (lê-se *"risk-five"*) é uma arquitetura *open source* com licença *BSD*, o que permite o seu livre uso para quaisquer fins, sem distinção de se o trabalho possui código-fonte aberto ou proprietário. Tal característica possibilita que grandes fabricantes utilizem a arquitetura para criar seus produtos, mantendo a proteção de propriedade intelectual sobre seus métodos de implementação e quaisquer subconjuntos de instruções não-*standard* que as empresas venham a

produzir, o que estimula investimentos em pesquisa e desenvolvimento.

Empresas como Google, IBM, AMD, Nvidia, Hewlett Packard, Microsoft, Qualcomm e Western Digital são algumas das fundadoras e investidoras da *RISC-V Foundation*, órgão responsável pela governança da arquitetura. Isso demonstra o interesse das gigantes do mercado no sucesso e disseminação da arquitetura.

A licença também permite que qualquer indivíduo produza, distribua e até mesmo comercialize sua própria implementação da arquitetura sem ter que arcar com *royalties*, sendo ideal para pesquisas acadêmicas, *startups* e até mesmo *hobbyistas*.

O conjunto de instruções foi desenvolvido tendo em mente seu uso em diversas escalas: sistemas embarcados, *smartphones*, computadores pessoais, servidores e supercomputadores, o que permitirá maior reuso de *software* e maior integração de *hardware*.

Outro fator que estimula o uso do *RISC-V* é a modernização dos livros didáticos. A nova versão do livro utilizado em OAC, Organização e Projeto de Computadores, de David Patterson e John Hennessy, utiliza a *ISA RISC-V*.

Além disso, com a promessa de se tornar uma das arquiteturas mais utilizadas nos próximos anos, utilizar o *RISC-V* como arquitetura da disciplina de OAC se mostra a escolha ideal no momento.

1.3 O Projeto RISC-V SiMPLE

O projeto *RISC-V SiMPLE (Single-cycle Multicycle Pipeline Learning Environment)* consiste no desenvolvimento de um processador com conjunto de instruções *RISC-V*, sintetizável em *FPGA* e com *hardware* descrito em *Verilog*. A microarquitetura implementada nesse trabalho é uniciclo, escalar, em ordem, com um único *hart* e com caminho de dados de 64 bits. Trabalhos futuros poderão utilizar a estrutura altamente configurável e modularizada do projeto para desenvolver as versões em microarquiteturas multiciclo e *pipeline*.

O processador contém o conjunto de instruções I (para operações com inteiros, sendo o único módulo com implementação mandatória pela arquitetura) e as extensões *standard* M (para multiplicação e divisão de inteiros) e F (para ponto flutuante com precisão simples conforme o padrão IEEE 754 com revisão de 2008). O projeto não implementa as extensões D (ponto flutuante de precisão dupla) e A (operações atômicas de sincronização), e com isso o *soft core* desenvolvido não pode ser definido como de propósito geral, G (que deve conter os módulos I, M, A, F e D). Assim, pela nomenclatura da arquitetura, o processador desenvolvido é um *RV64IMF*.

O projeto contempla *traps*, interrupções, exceções, *CSRs*, chamadas de sistema e outras funcionalidades de nível privilegiado da arquitetura.

O *soft core* possui barramento Avalon para se comunicar com os periféricos das plataformas de desenvolvimento. O projeto foi desenvolvido utilizando a placa DE2–115 com *FPGA Altera Cyclone* e permite a fácil adaptação para outras placas da Altera.

Capítulo 2

Revisão Teórica

2.1 Arquitetura de Computadores

Para nos comunicarmos, necessitamos de uma linguagem, e no caso dos brasileiros, essa linguagem é o português. Como toda linguagem, o português possui sua gramática e dicionário que lhe dão estrutura e sentido. Línguas humanas como o português, inglês e espanhol são chamadas de linguagens naturais, e evoluíram naturalmente a partir do uso e repetição. [1]

Por causa da excelente capacidade de interpretação e adaptação da mente humana, somos capazes de criar e entender novos dialetos que não seguem as regras formais das linguagens naturais que conhecemos. Porém, fora da comunicação casual é importante e às vezes obrigatório que nos expressemos sem ambiguidade. Línguas artificiais como a notação matemática e linguagens de programação possuem semântica e sintaxe mais rígidas para garantir que a mensagem transmitida seja interpretada da maneira correta. Sem essa rigidez, os computadores de hoje não seriam capazes de entender nossos comandos.

Para a comunicação com o processador de um computador, utilizamos mensagens chamadas de instruções, e o conjunto dessas instruções é chamado de Arquitetura do Conjunto de Instruções (*ISA*). Um processador só é capaz de entender as mensagens que obedecem as regras semânticas e sintáticas de sua *ISA*, e qualquer instrução que fuja das suas regras causará um erro de execução ou realizará uma tarefa diferente da pretendida. A linguagem de máquina é considerada de baixo nível pois apresenta pouca ou nenhuma abstração em relação à arquitetura.

As instruções são passadas para o processador na forma de código de máquina, sequências de dígitos binários que correspondem aos níveis lógicos do circuito. Para melhorar o entendimento do código e facilitar o desenvolvimento, uma outra representação é utilizada, o *assembly*. Um código *assembly* é transformado em código de máquina por um programa montador, ou *assembler*, e o processo inverso é realizado por um *disassembler*. As linguagens *assembly*, dependendo do *assembler* utilizado, permitem o uso de macros de substituição e pseudo instruções (determinadas instruções que não existem na *ISA* são expandidas em instruções válidas pelo montador) e são totalmente dependentes da arquitetura do processador, o que impede que o mesmo código seja executado em arquiteturas diferentes.

A Figura 2.1 é uma representação simplificada de um processador. A unidade de controle lê uma instrução da memória e a decodifica; o circuito de lógica combinacional lê os dados dos registradores, entrada e memória conforme necessário, executa a instrução decodificada e escreve no banco de registradores, na memória de dados ou na saída se for preciso; a unidade de controle lê uma nova instrução e o ciclo se repete até o fim do programa. A posição de memória da instrução que está sendo executada fica armazenada em um registrador especial chamado de Contador de Programa, ou *PC*. Algumas instruções modificam o *PC* condicionalmente ou diretamente, criando a estrutura para saltos, laços e chamada e retorno de funções.

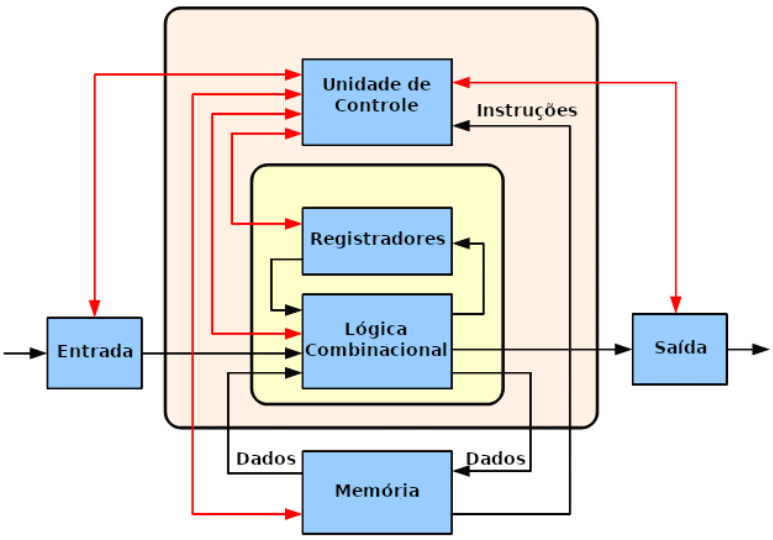

Figura 2.1: Abstração da arquitetura de um computador. [2]

Historicamente, as arquiteturas são divididas em *ISAs RISC* e *CISC*. Na atualidade, a diferença entre elas é que as *ISAs RISC* acessam a memória por instruções de *load/store*, enquanto as *CISC* podem acessar a memória diretamente em uma instrução de operação lógica ou aritmética.

Algumas arquiteturas *RISC* notáveis são a *RISC-V*, a *ARM* e a *MIPS*. Quanto às *CISC*, a *x86* e sua extensão de 64 *bits*, a *AMD64*, são as mais conhecidas.

2.1.1 Arquitetura RISC-V

2.1.2 Arquitetura MIPS

2.1.3 Arquitetura ARM

2.1.4 Arquitetura x86

2.1.5 Arquitetura AMD64

2.2 Microarquiteturas

2.2.1 Uniciclo

2.2.2 Multiciclo

2.2.3 Pipeline

2.3 Representação de Hardware

2.3.1 Verilog HDL

2.4 Síntese Lógica

2.4.1 Análise e Síntese

2.4.2 Fitting

2.4.3 Timing Analyzer

2.5 Simulação

2.6 Field Programmable Gate Arrays

Field Programmable Gate Arrays—ou *FPGAs*—são circuitos integrados que permitem o desenvolvimento de circuitos lógicos reconfiguráveis. Por serem reprogramáveis, as *FPGAs* geram uma grande economia em tempo de desenvolvimento e em custos como os de prototipagem, validação e manufatura do projeto em relação aos circuitos de aplicações específicas, os *ASICs*. As *FPGAs* podem ser tanto o passo intermediário no projeto de um *ASIC* quanto o meio final do projeto quando a reconfigurabilidade e os preços muito mais acessíveis forem fatores importantes.

Cada fabricante de *FPGAs* possui seus *softwares* de desenvolvimento, ou *SDKs*. A indústria de *hardware* é extremamente protecionista com sua propriedade intelectual, sendo a maioria dessas

ferramentas de código proprietário. Para a Intel Altera®, essa plataforma é o Quartus Prime®.

FPGAs mais modernas possuem, além do arranjo de portas lógicas, blocos de memória, *PLLs*, *DSPs* e *SoCs*. Os blocos de memória internos funcionam como a memória *cache* de um microprocessador, armazenando os dados próximo ao seu local de processamento para diminuir a latência. Os *PLLs* permitem criar sinais de *clock* com diversas frequências a partir de um relógio de referência, e podem ser reconfigurados a tempo de execução. *DSPs* são responsáveis pelo processamento de sinais analógicos discretizados, e podem ser utilizados como multiplicadores de baixa latência. Já os *SoCs* são microprocessadores como os ARM® presentes em celulares, e são capazes de executar sistemas operacionais como o Linux.

Além de disponíveis na forma de *chips* para a integração com placas de circuito impresso customizadas, as *FPGAs* possuem *kits* de desenvolvimento com diversos periféricos para auxiliar no processo de criação de soluções. Esses *kits* são a principal ferramenta de aprendizagem no universo dos circuitos reconfiguráveis. No Laboratório de Informática da UnB, as placas *terasIC DE1-SoC* com a *FPGA Intel® Cyclone V SoC* estão disponíveis para os alunos de OAC desenvolverem seus projetos.

2.6.1 Arquitetura Generalizada de uma FPGA

De forma genérica, uma *FPGA* possui blocos lógicos, chaves de interconexão, blocos de conexão direta e portas de entrada e saída, conforme apresentado na Figura 2.2.

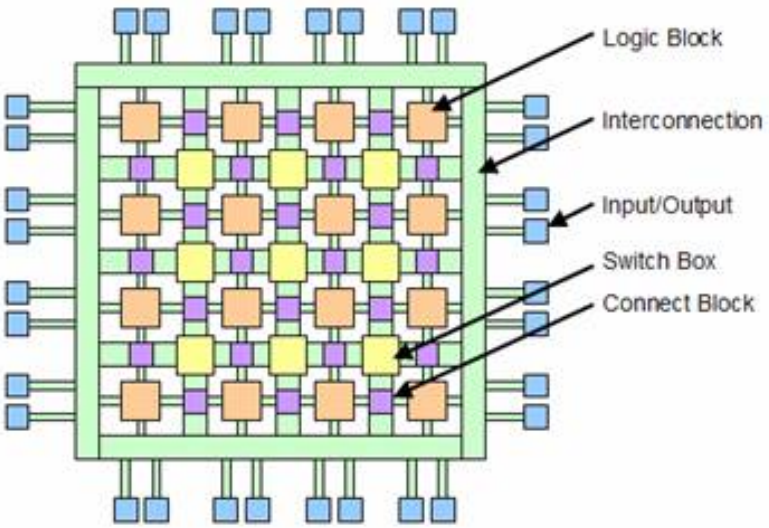

Figura 2.2: Abstração da arquitetura de uma FPGA Fonte: Olin College of Engineering

Os blocos lógicos possuem *lookup tables*, registradores, somadores e multiplexadores. É neles que a lógica reconfigurável é implementada.

Já as chaves de interconexão são responsáveis por conectar os diversos blocos da *FPGA*. A Figura 2.3 exemplifica como é feito o roteamento da malha de interconexão. Os blocos de conexão direta são um tipo especial de chave de interconexão, e sua função é ligar blocos lógicos adjacentes.

Por fim, as portas de entrada e saída conectam a *FPGA* ao "mundo externo" e.g. *drivers* de áudio e vídeo.

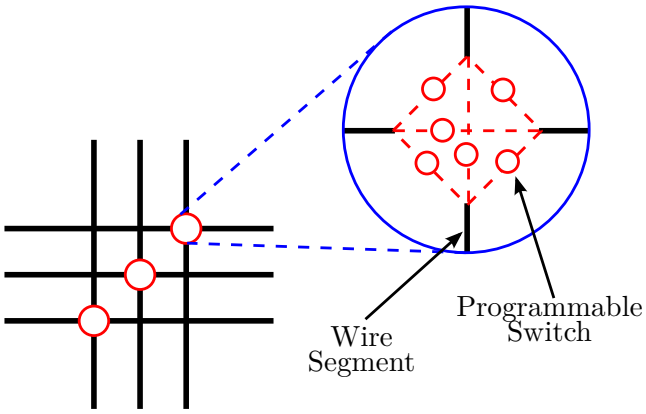

Figura 2.3: Funcionamento da chave de interconexão Fonte: Wikimedia

Capítulo 3

Sistema Proposto

Capítulo 4

Resultados

Capítulo 5

Conclusões

5.1 Perspectivas Futuras

REFERÊNCIAS BIBLIOGRÁFICAS

[1] LYONS, J. *Natural language and universal grammar*. Cambridge England New York: Cambridge University Press, 1991. ISBN 9780521246965.

[2] LAMBTRON. *Block diagram of a basic computer with uniprocessor CPU. (CC BY-SA 4.0)*. 2015. Disponível em: <https://en.wikipedia.org/wiki/File:ABasicComputer.gif>.

ANEXOS

I. DESCRIÇÃO DO CONTEÚDO DO CD

Descrever CD.

II. PROGRAMAS UTILIZADOS

Quais programas foram utilizados?