TRABALHO DE GRADUAÇÃO

RISC-V SiMPLE

Arthur de Matos Beggs

Brasília, Julho 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

Prof. Ricardo Pezzuol Jacobi, CIC/UnB _____
Co-Orientador

Brasília, Julho de 2019

FICHA CATALOGRÁFICA

ARTHUR, DE MATOS BEGGS RISC-V SiMPLE, [Distrito Federal] 2019. $n°$???, ???p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2019). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia. 1. RISC-V 2. ??? 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

Dedico ao pato de borracha especialista em TI que sempre me ajuda a depurar meus códigos.

Arthur de Matos Beggs

Agradecimentos

Agradecimentos!

Arthur de Matos Beggs

RESUMO

Desenvolvimento de um processador com arquitetura do conjunto de instruções RISC-V em Verilog sintetizável em FPGA para ser utilizado como recurso de laboratório para a disciplina de Organização e Arquitetura de Computadores da Universidade de Brasília. O processador contará com perifericos e.g. saída de vídeo para proporcionar um ambiente completo de desenvolvimento de hardware e software.

Palavras Chave: RISC-V

ABSTRACT

Abstract.

Keywords: RISC-V

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*
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
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

A Arquitetura RISC-V

2.1 Visão Geral da Arquitetura

A *ISA RISC-V* é uma arquitetura modular, sendo o módulo base de operações com inteiros mandatório em qualquer implementação. Os demais módulos são extensões de uso opcional. A arquitetura não suporta *branch delay slots* e aceita instruções de tamanho variável. A codificação das instruções de tamanho variável é mostrada na Figura 2.1. As instruções presentes no módulo base correspondem ao mínimo necessário para emular por *software* as demais extensões (com exceção das operações atômicas).

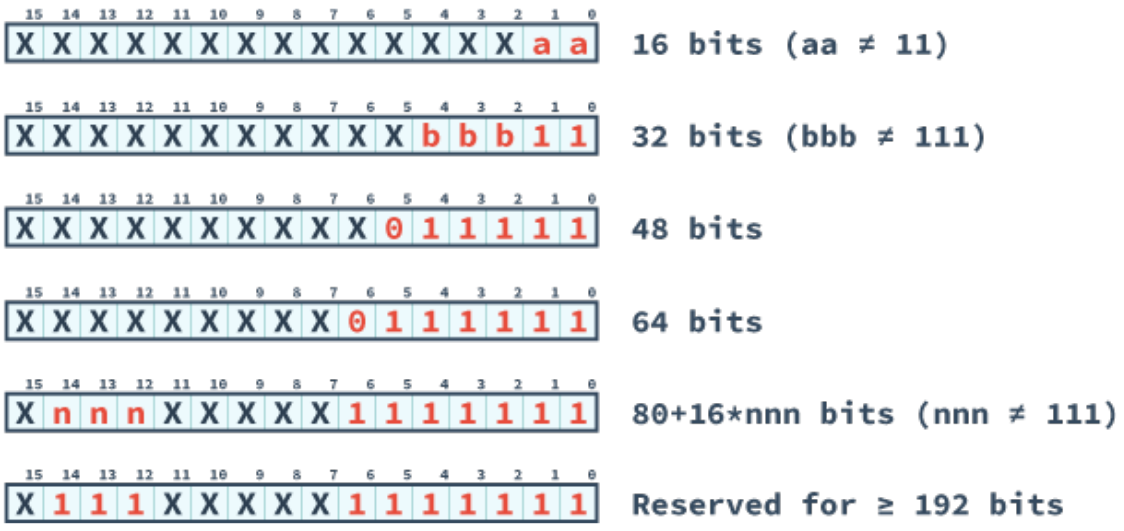

Figura 2.1: Codificação de instruções de tamanho variável da arquitetura *RISC-V*

A nomenclatura do conjunto de instruções implementado segue a seguinte estrutura:

- As letras “RV”;
- A largura dos registradores do módulo Inteiro;
- A letra “I” representando a base Inteira. Caso o subconjunto Embarcado (*Embedded*) seja implementado, substitui-se pela letra “E”;
- Demais letras identificadoras de módulos opcionais.

Assim, uma implementação com registradores de 64 bits somente com o módulo base de Inteiros é denominado “RV64I”.

2.2 Módulo Inteiro

2.3 Extensões

2.3.1 Extensão M

2.3.2 Extensão A

2.3.3 Extensão F

2.3.4 Extensão D

2.3.5 Outras Extensões

2.4 Arquitetura Privilegiada

2.5 Formatos de Instruções

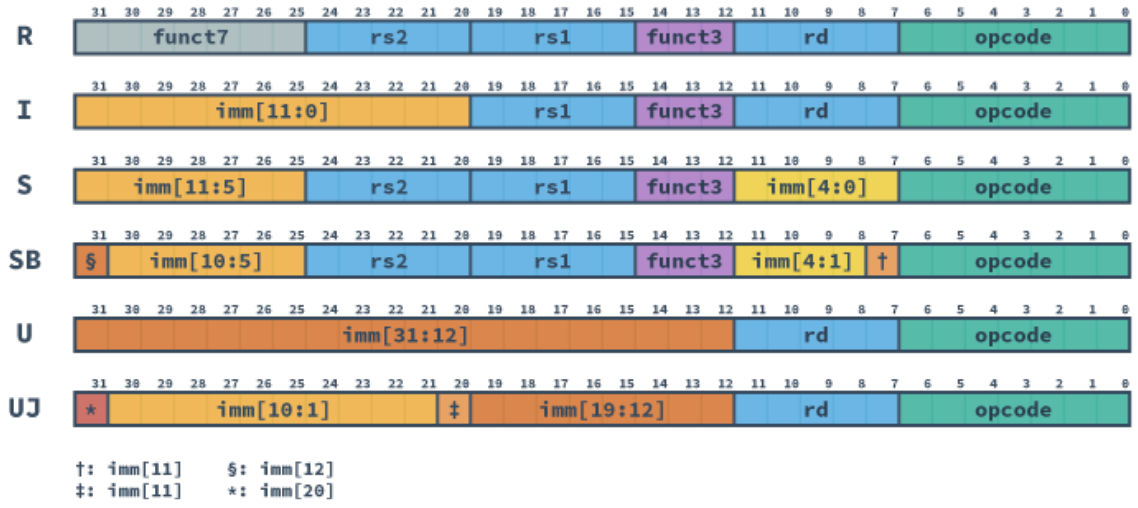

Figura 2.2: Formatos de Instruções da*ISA RISC-V*

2.6 Formatos de Imediatos

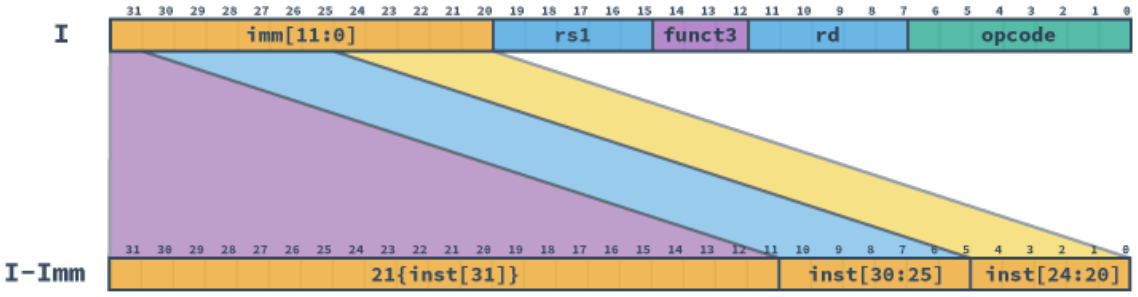

Figura 2.3: Formação do Imediato de tipo I

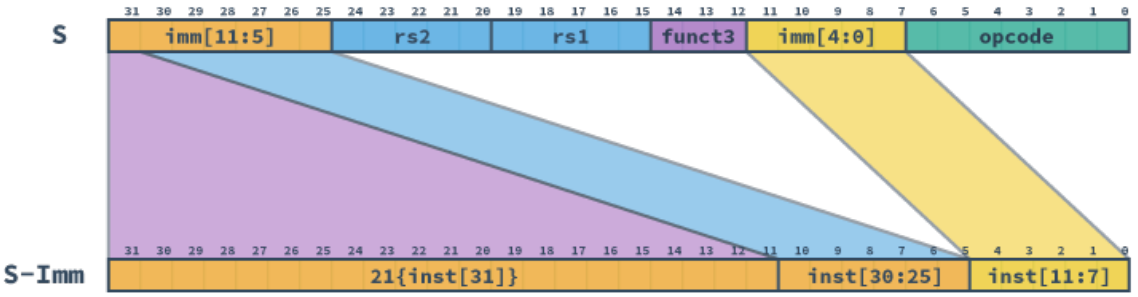

Figura 2.4: Formação do Imediato de tipo S

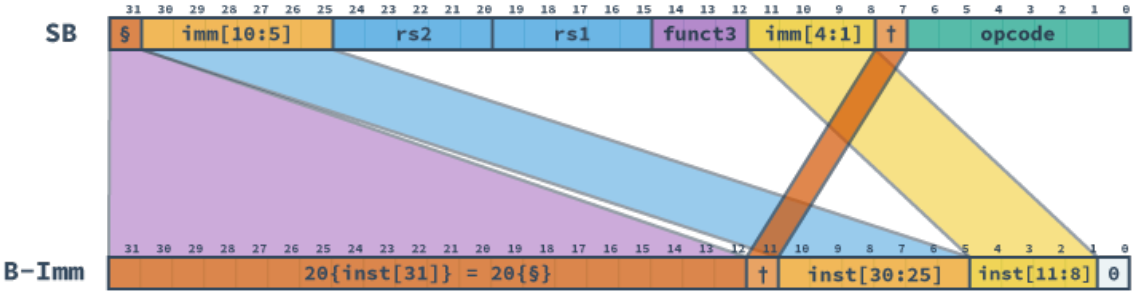

Figura 2.5: Formação do Imediato de tipo B

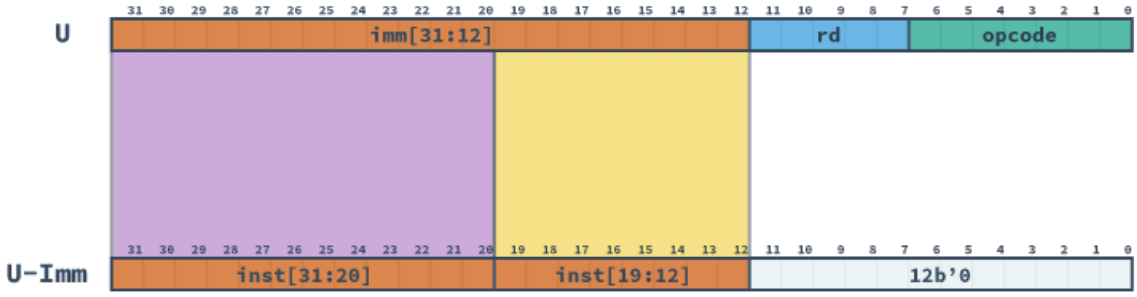

Figura 2.6: Formação do Imediato de tipo U

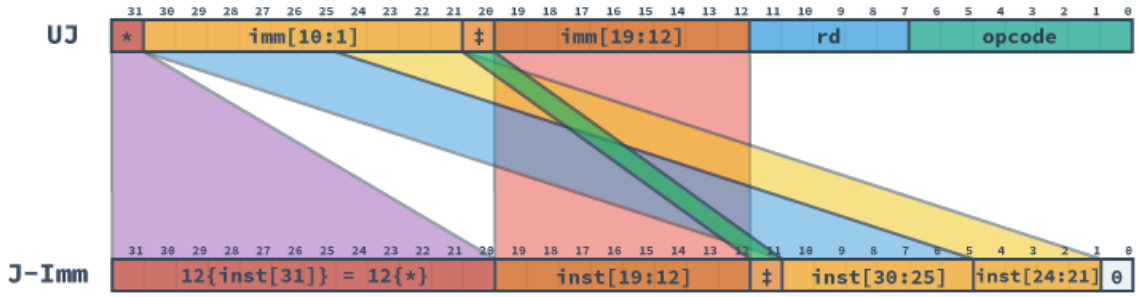

Figura 2.7: Formação do Imediato de tipo J

Capítulo 3

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.

3.1 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 3.1.

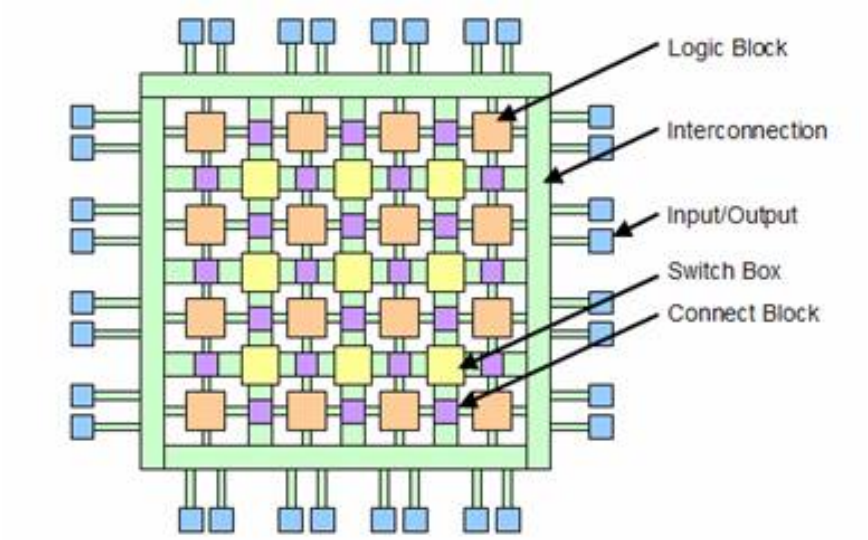

Figura 3.1: Abstração da arquitetura de uma FPGA Fonte: Olin College of Engineering

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 3.2 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.

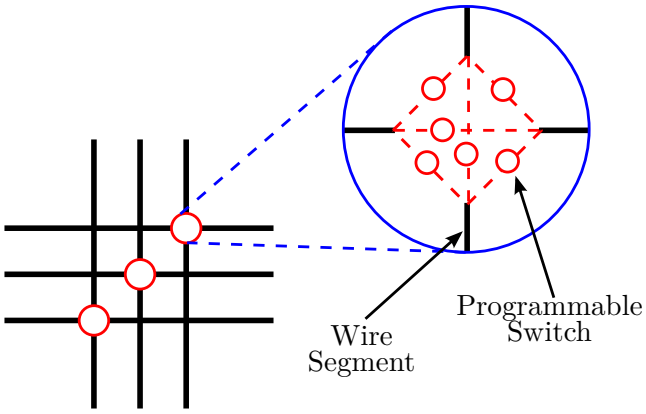

Figura 3.2: Funcionamento da chave de interconexão Fonte: Wikimedia

3.2 Arquitetura da FPGA Cyclone V SoC

A Figura 3.3 apresenta a arquitetura da *FPGA Cyclone V SoC*.

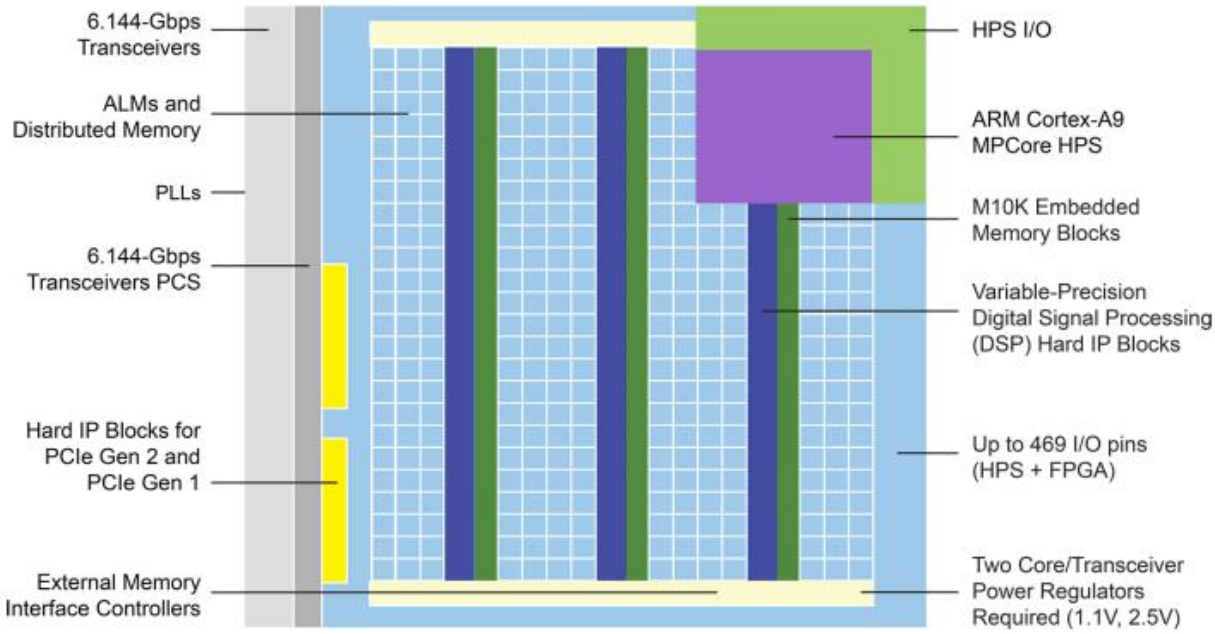

Figura 3.3: Arquitetura da *FPGA Altera Cyclone V SoC* Fonte: Intel

3.2.1 Adaptative Logic Modules

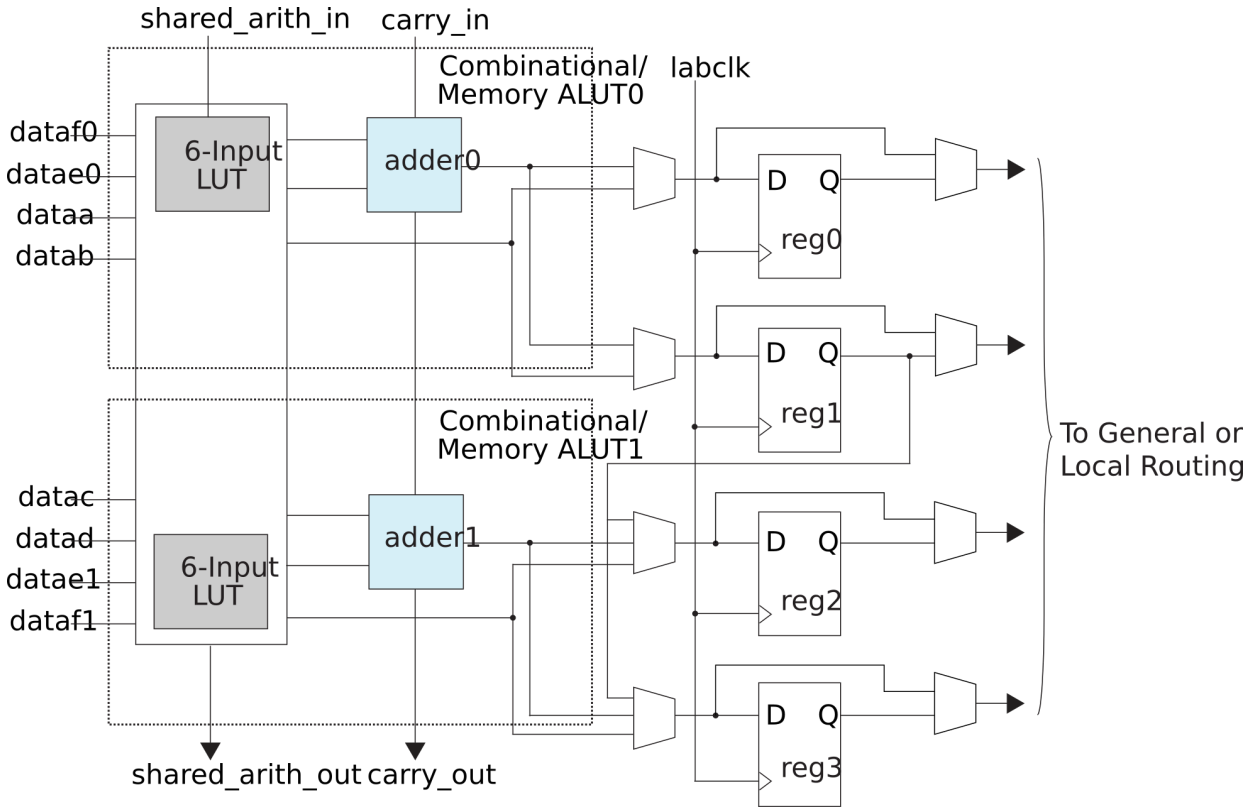

Figura 3.4: Diagrama de blocos de um ALM Fonte: Intel

3.2.2 Embedded Memory Blocks

3.2.3 Hard Processor System

3.2.4 Phase-Locked Loops

3.3 A Placa de Desenvolvimento DE1-SoC

Capítulo 4

Verilog

Capítulo 5

Síntese em Hardware

5.1 Análise e Síntese

5.2 Fitting

5.3 Timing Analyzer

5.4 Simulação

Capítulo 6

Implementação

6.1 Caminho de Dados

O caminho de dados projetado para a implementação da microarquitetura uniciclo é apresentado na Figura 6.1.

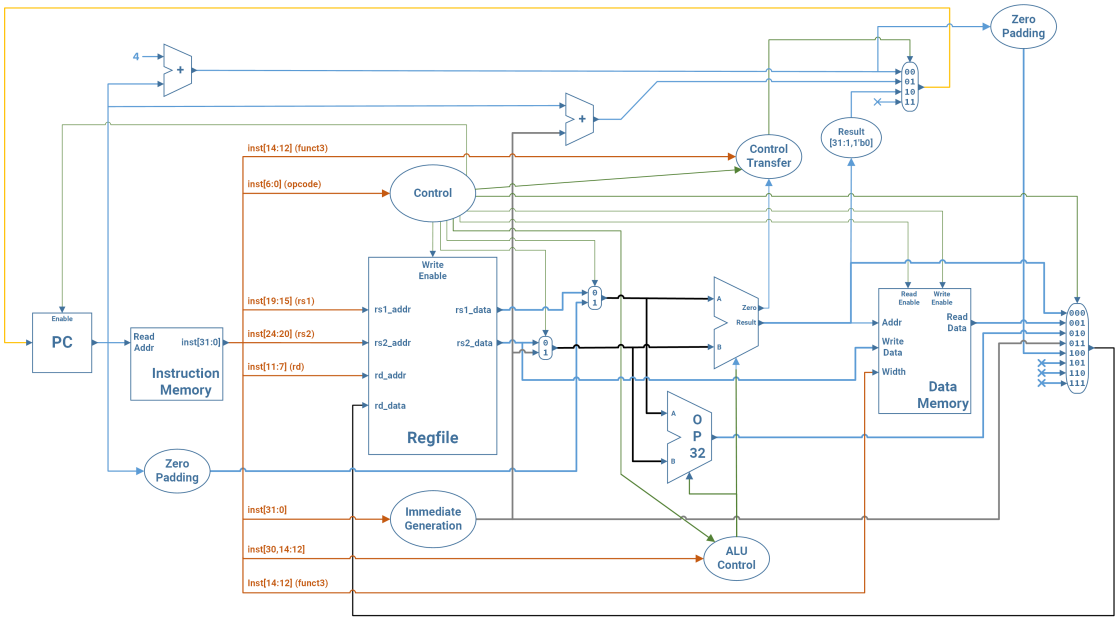

Figura 6.1: Caminho de Dados implementado para o módulo I

O *datapath* possui um banco de 32 registradores de uso geral de 64 bits cada. A memória possui arquitetura Harvard, sendo a memória de instruções (*text*) *read-only* e a memória de dados (*data*) *read-write*. São implementadas 49 instruções, sendo elas:

- LUI: Load Upper Intermediate;

- AUIPC: Add Upper Intermediate to Program Counter;
- JAL: Jump And Link;
- JALR: Jump And Link Register;
- BEQ: Branch if EQual;
- BNE: Branch if Not Equal;
- BLT: Branch if Less Than;
- BGE: Branch if Greater or Equal;
- BLTU: Branch if Less Than Unsigned;
- BGEU: Branch if Greater or Equal Unsigned;
- LB: Load Byte;
- LH: Load Halfword;
- LW: Load Word;
- LBU: Load Byte Unsigned;
- LHU: Load Halfword Unsigned;
- SB: Store Byte;
- SH: Store Halfword;
- SW: Store Word;
- ADDI: ADD Immediate;
- SLTI: Set on Less Than;
- SLTIU: Set on Less Than Unsigned;
- XORI: XOR Immediate;
- ORI: OR Immediate;
- ANDI: AND Immediate;
- SLLI: Shift Left Logical Immedate;
- SRLI: Shift Right Logical Immediate;
- SRAI: Shift Right Arithmetic Immediate;
- ADD: ADD;
- SUB: SUB;

- SLL: Shift Left Logical;
- SLT: Set on Less Than;
- SLTU: Set on Less Than Unsigned;
- XOR: XOR;
- SRL: Shift Right Logical;
- SRA: Shift Right Arithmetic;
- OR: OR;
- AND: AND;
- LWU: Load Word Unsigned;
- LD: Load Double;
- SD: Store Double;
- ADDIW: ADD Immediate Word-size;
- SLLIW: Shift Left Logical Immedate Word-size;
- SRLIW: Shift Right Logical Immediate Word-size;
- SRAIW: Shift Right Arithmetic Immediate Word-size;
- ADDW: ADD Word-size;
- SUBW: SUB Word-size;
- SLLW: Shift Left Logical Word-size;
- SRLW: Shift Right Logical Word-size;
- SRAW: Shift Right Arithmetic Word-size;

Para que o processador seja completamente compatível com a especificação da *ISA*, falta implementar tratamentos de exceções, interrupções e *traps*, Registradores *CSR*, instruções de chamada ao ambiente (ECALL/EBREAK), instruções de *fencing* de memória, suporte ao acesso desalinhado à memória de dados e pilha de endereço de retorno (RAS).

Capítulo 7

Resultados

Capítulo 8

Conclusões

8.1 Perspectivas Futuras

ANEXOS

I. DESCRIÇÃO DO CONTEÚDO DO CD

Descrever CD.

II. PROGRAMAS UTILIZADOS

Quais programas foram utilizados?