RVSH - Um processador RISC-V para fins didáticos

Eduardo Michel Deves de Souza
Universidade do Vale do Itajaí, Brasil
eduardomichel@edu.univali.br

Nathalia Adriana de Oliveira
Universidade do Vale do Itajaí, Brasil
oliveiranathalia@edu.univali.br

Douglas Almeida dos Santos
Universidade do Vale do Itajaí, Brasil
Université de Montpellier, França
douglasas@edu.univali.br

Douglas Rossi de Melo
Universidade do Vale do Itajaí, Brasil
drm@univali.br

ABSTRACT

Embedded systems constitute the class of computers that present the most significant volume and are increasingly present in everyday life. The main element of these systems is the processor, which can be found in discrete form, represented by a physical component, or cores, as used in programmable logic devices. Processors of the same architecture share the same instruction set but may differ in the organization's implementation. RISC (Reduced Instruction Set Computer) is the class of architectures that favors a simple, reduced instruction set. RISC-V is an example of such architecture, which consists of an initiative by academia and industry to be open and free, aiming for easy and optimized implementations. However, due to the recent disclosure of its features and specifications, RISC-V needs more reference material for digital and embedded system designs. This work proposes the RVSH, a simple RISC-V processor for teaching and research activities. The implementation aims to allow the adoption of this architecture in topics such as digital systems, computer architecture, microcontrollers, and embedded systems design.

KEYWORDS

Sistemas Embarcados, Arquitetura de Computadores, RISC-V

1 INTRODUÇÃO

Sistemas embarcados compõem a maior classe de sistemas computacionais e são projetados de modo a cumprir uma função determinada. Todo sistema embarcado é composto por uma unidade de processamento, que desempenha o monitoramento e controle das informações provenientes dos periféricos [1]. Os processadores de propósito geral são geralmente utilizados em sistemas embarcados. Isso é dado principalmente, pelo seu bom desempenho aliado ao baixo custo unitário [2].

Desde a divulgação da primeira arquitetura de sua classe [3], os processadores RISC (do inglês, Reduced Instruction Set Computer) têm aumentado sua presença de mercado, sendo hoje a linha de processadores dominante no segmento de sistemas embarcados. O RISC-V é uma arquitetura de processadores aberta e gratuita, que teve sua concepção iniciada na última década [4]. Seu projeto foi coordenado por David Patterson [5], um dos idealizadores dos processadores RISC e criador da arquitetura MIPS. Atualmente, a fundação RISC-V conta com mais de 200 empresas associadas [6].

Tendo em vista o crescimento da adoção da arquitetura RISC-V, alguns trabalhos apresentam simuladores RISC-V voltados para o ensino [7–11]. Alguns desses trabalhos apresentam simuladores em nível arquitetural, em que o objetivo é o ensino da linguagem de montagem, enquanto outros abordam o nível microarquitetural do processador. No entanto, os trabalhos que ensinam o nível microarquitetural utilizam abordagens complexas que utilizam pipeline e não apresentam um modelo sintetizável do processador.

Nesse contexto, este trabalho apresenta o RVSH, um processador RISC-V simples com arquitetura monociclo voltado para uso no ensino e como suporte em pesquisas relacionadas. Esse processador foi implementado utilizando a linguagem de descrição de hardware sem a utilização de módulos proprietários. Como premissa, foi buscado manter as entidades com descrições simples e de fácil entendimento. No contexto do ensino, esse processador apresenta um modelo simples para que alunos possam avaliar a sua implementação para dispositivos programáveis utilizando de linguagem de descrição de hardware.

2 FUNDAMENTAÇÃO TEÓRICA

O RISC-V é uma arquitetura para processadores de propósito geral, que especifica o padrão da linguagem de máquina entendida pelo processador. Essa arquitetura tem como princípio se manter adequada para qualquer dispositivo computacional, podendo ser utilizada tanto em dispositivos de alto desempenho como microcontroladores. Atualmente, tem-se 4 conjuntos de instruções-base [12], as quais são:

A) RV32I, um conjunto de 40 instruções completo o suficiente para para satisfazer os requisitos básicos de sistemas operacionais modernos;
B) RV64I, similar com o conjunto RV32I, difere apenas na largura dos registradores de inteiros e no contador de programa (PC - Program Counter);
C) RV32E, se assemelha ao conjunto RV32I, porém foi projetado para utilizar apenas 15 registradores; e
D) RV128I, é similar ao RV32I e RV64I, difere apenas na largura dos registradores de inteiros e no contador de programa (PC - Program Counter).

Essa arquitetura foi projetada para simplificar a implementação: a codificação das instruções é extremamente regular, o modelo da memória é direto e não possui instruções complexas para acesso à memória. Um dos benefícios do RISC-V é que núcleos mínimos são muito menores do que similares, como ARM e x86, apesar da diferença não ser significativa em núcleos de maior capacidade [13].

3 TRABALHOS RELACIONADOS

Atualmente, existem aplicações que auxiliam na aprendizagem da arquitetura RISC-V no meio acadêmico. Os principais exemplos encontrados são simuladores e plataformas.

3.1 Simuladores

O Jupiter é um montador e simulador de código aberto para a arquitetura RISC-V, voltado para educação. Tem suporte ao conjunto base de inteiros, com extensões de multiplicação, divisão de inteiros e ponto flutuante de precisão simples [7].

O RARS é um montador e simulador de código aberto para a arquitetura RISC-V, baseado no MARS. O RARS suporta o conjunto de instruções base de inteiros, as extensões de multiplicação e divisão de inteiros, ponto flutuante de precisão simples e dupla e também a interrupções a nível de usuário [8].

O Ripes é um simulador e editor de código construído para a arquitetura RISC-V. É voltado ao ensino da linguagem de montagem e é executado em várias microarquiteturas [9].

O Venus é um simulador RISC-V para uso no ensino. Oferece suporte a todas as pseudo-instruções do RISC-V padrão, contanto que se traduzam em uma série de instruções compatíveis [10].

O WebRISC-V é um ambiente de simulação online, com caminho de dados em pipeline gráfico, construído para a arquitetura RISC-V. É adequado para ensinar como o código à nível de montagem é executado na arquitetura pipeline RISC-V e para ilustrar os elementos arquiteturais [11].

3.2 Plataformas

O Pulpino é uma plataforma de código aberto, baseada na arquitetura RISC-V, desenvolvida no Instituto Federal Suiço de Tecnologia. Pode ser configurada para os núcleos RISCY ou zero-riscy [14].

O Rocket Chip é um gerador SoC (System-on-Chip) de núcleos de processadores RISC-V, desenvolvido na Universidade da Califórnia em Berkeley. É composto de núcleos de execução ordenada (Rocket) e fora de ordem (BOOM) [15].

O NeoRV32 é um processador compatível com a arquitetura RISC-V que possui um sistema altamente configurável. Tal sistema é projetado de forma a facilitar a integração em projetos SoC [16].

4 SOLUÇÃO PROPOSTA

Este trabalho foi motivado pelo projeto de processador tolerante a falhas previamente descrito [17]. Este projeto tem como principal objetivo a simplicidade e fácil compreensão, visto que, é destinado para fins didáticos.

A visão geral do RVSH é constituída por três blocos: a Memória de Instruções (`inst_mem`), a Unidade Central de Processamento (`rv_cpu`) e a Memória de Dados (`data_mem`). A Figura 1 corresponde à representação do bloco Unidade Central de Processamento, detalhando as estruturas internas e suas funcionalidades.

A Busca de Instrução (`inst_fetch`) seleciona a instrução a ser executada. O Banco de Registradores (`reg_file`) é onde ficam alocados os registradores do processador. A Unidade Lógica Aritmética (`alu`) é responsável por realizar o processo de execução de operações lógicas e aritméticas do processador. O Seletor de

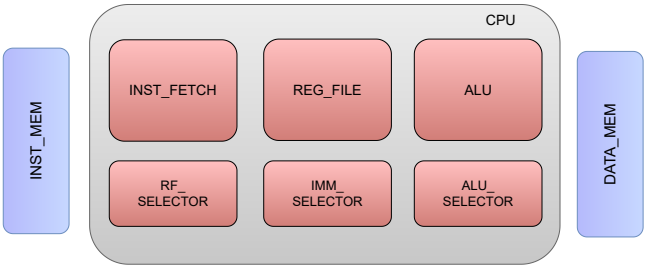

Figura 1: Visão geral do RVSH

Escrita no Registrador (`rf_selector`) seleciona o que será escrito no Banco de Registradores. O Seletor de Imediato (`imm_selector`) seleciona o caminho que o valor imediato irá percorrer. O Seletor de Dado da ULA (`alu_selector`) seleciona o dado que irá para a Unidade Lógica Aritmética. Por fim, todos os blocos são administrados por um Controlador (`ctrl`), abstraído na imagem.

Para a descrição de hardware foi utilizada a linguagem VHDL e o software Intel Quartus Prime 20.1, com a ferramenta ModelSim Intel FPGA 2020.1 para realizar as simulações necessárias para a verificação do processador.

5 RESULTADOS

Nesta seção são apresentados os resultados de síntese e simulação, visando obter o custo e verificar o correto funcionamento do processador.

5.1 Síntese

Para obter o custo da implementação dos componentes, utilizou-se a ferramenta Intel Quartus Prime para a síntese com as informações de custo e também obtenção de modelo em nível de transferência de registradores.

Na Tabela 1 são apresentados os recursos utilizados pelo processador isolado e também acrescido das memórias. Considerando apenas o processador, a quantidade de Flip-Flops (FFs) utilizados é de 1024 e o uso de Look-Up Tables (LUTs) para lógica combinacional é de 1504, com frequência máxima do processador de 50,08 MHz. Em um sistema com processador e memórias, observa-se um aumento de FFs em 103% e de LUTs em 22%, com degradação da frequência máxima em 17%.

Tabela 1: Recursos utilizados pelo processador

Componente	FFs	LUTs	Fmax (MHz)
CPU	1024	1504	50,08
CPU+MEM	2080	1831	41,32

5.2 Verificação

A ferramenta ModelSim Intel FPGA foi utilizada para verificar o funcionamento do processador, executando um modelo que possui instruções de todas as classes disponíveis no processador desenvolvido.

As instruções utilizadas para verificação foram: carregamento de endereço (`lui`), soma aritmética com o dado imediato (`addi`), substração aritmética (`sub`), transferência de dados da memória para o registrador (`lw`) e a operação inversa (`sw`), salto condicional (`beq`) e desvio incondicional (`jal`).

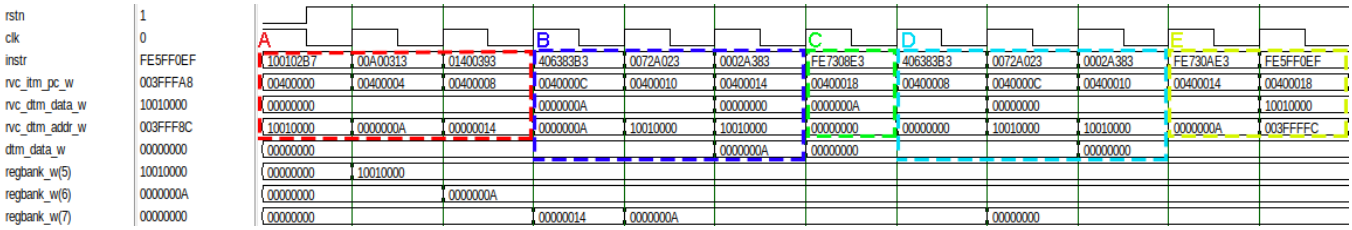

Figura 2: Diagrama de formas de onda do funcionamento do processador

Foi utilizado o código do Quadro 1, na linguagem de montagem da arquitetura RISC-V, para verificar o funcionamento do processador, demonstrando o uso das instruções anteriormente mencionadas.

Quadro 1: Código para verificação do processador

```
main:
   lui   x5, 0x010010      # carrega o endereço no x5
   addi  x6, x0, 10        # adiciona 10 no x6
   addi  x7, x0, 20        # adiciona 20 no x7
loop:
   sub   x7, x7, x6        # subtrai x7 com x6
   sw    x7, 0(x5)         # armazena x7 na posição x5 da memória
   lw    x7, 0(x5)         # carrega da posição x5 da memória no x7
   beq   x6, x7, loop      # verifica igualdade ente x6 e x7
   jal   main              # salta para o início
```

A Figura 2 apresenta o diagrama de formas de onda da simulação da execução do processador. As operações estão separadas em 5 campos de seleção de A a E.

- **A)** As operações descritas na área de seleção A correspondem às inicializações dos valores dos registradores, respectivamente `x5`, `x6` e `x7`;
- **B)** Na área de seleção B ocorrem as operações executadas no laço de repetição, realizando a subtração dos registradores `x7` e `x6` e armazenando o resultado para uso posterior;
- **C)** Na área de seleção C é verificada a igualdade dos registradores `x6` e `x7`, realizando o retorno para o rótulo `loop`;
- **D)** Na área de seleção D é efetuada novamente a subtração entre `x7` e `x6`, o que resulta em conteúdos diferentes que implicam na não realização do salto condicional;
- **E)** A área de seleção E apresenta o salto incondicional para o início do programa (`main`).

6 CONSIDERAÇÕES FINAIS

Neste trabalho é apresentado o RVSH, um processador de propósito geral baseado na arquitetura RISC-V, voltado para fins didáticos e descrito em VHDL. O processador desenvolvido visa facilitar a aprendizagem de conceitos relacionados a arquitetura de computadores e sistemas digitais.

O processador foi caracterizado em termos de custo em elementos lógicos (FFs e LUTs) e desempenho (Fmax), na sua forma isolada e também com as memórias de instruções e de dados. O processador também foi verificado por meio de simulação, que comprovou seu correto funcionamento ao executar instruções das diferentes classes presentes na especificação da arquitetura.

Como trabalhos futuros, pretende-se desenvolver uma série de roteiros e práticas de laboratório utilizando o processador desenvolvido como referência. Também, pretende-se utilizar o processador como base para construção de sistemas mais complexos e adições de novas extensões ao processador, como unidade de multiplicação e unidade de ponto flutuante. O código fonte do processador está disponível em [18].

AGRADECIMENTOS

Este trabalho foi financiado em parte pelo Programa de Bolsas Universitárias de Santa Catarina (UNIEDU), pela Fundação de Amparo à Pesquisa e Inovação de Santa Catarina (FAPESC-2021TR001907) e pela Region d'Occitanie e École Doctorale I2C da Universidade de Montpellier (20007368/ALDOCT-000932).

REFERÊNCIAS

[1] Jonathan W Valvano. *Introduction to embedded microcomputer systems: Motorola 6811 and 6812 simulation*. Thomson-Brooks/Cole, 2003.
[2] Frank Vahid and Tony D Givargis. *Embedded System Design: A Unified Hardware/Software Introduction*. John Wiley & Sons, Inc., 1st edition, 2001.
[3] Carlo H Séquin and David A Patterson. *Design and Implementation of RISC I*. University of California at Berkeley, 1982.
[4] Andrew Waterman, Yunsup Lee, David A Patterson, and Krste Asanovic. The RISC-V instruction set manual, volume I: Base user-level ISA. *EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62*, 116, 2011.
[5] David A Patterson. Reduced instruction set computers. *Communications of the ACM*, 28(1):8–21, 1985.
[6] RISC-V Foundation. RISC-V: The free and open RISC instruction set architecture, 2019. URL https://riscv.org/.
[7] Andrés Castellanos. Jupiter. URL https://github.com/andrescv/Jupiter.
[8] Benjamin Landers. RARS. URL https://github.com/TheThirdOne/rars.
[9] Morten Borup Petersen. Ripes. URL https://github.com/mortbopet/Ripes.
[10] Keyhan Vakil. Venus. URL https://github.com/kvakil/venus.
[11] Roberto Giorgi and Gianfranco Mariotti. WebRISC-V: a web-based education-oriented RISC-V pipeline simulation environment. In *ACM Workshop on Computer Architecture Education (WCAE-19)*, pages 1–6, jun 2019.
[12] Andrew Waterman and Krste Asanovi'c. *The RISC-V Instruction Set Manual*. 2019. URL https://riscv.org/technical/specifications/.
[13] David Kanter. RISC-V offers simple, modular ISA. *Microprocessor Report*, 2016.
[14] Francesco Conti. Pulpino. URL https://github.com/pulp-platform/pulpino.
[15] Krste Asanović, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Ben Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David A. Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. The rocket chip generator. URL https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html.
[16] Stephan Nolting. NeoRV32. URL https://github.com/stnolting/neorv32.
[17] Douglas Almeida Santos, Lucas Matana Luza, Cesar Albenes Zeferino, Luigi Dilillo, and Douglas Rossi Melo. A low-cost fault-tolerant RISC-V processor for space systems. In *2020 15th Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–5, 2020.
[18] RVSH. Um processador RISC-V para fins didáticos. URL http://xarc.org/rvsh.