MULTIPLICADORES BINÁRIOS EM MICROARQUITETURA RISC-V

José Augusto dos Santos Lemos (IC), Gustavo Della Colletta (PQ)[1]
[1]*Universidade Federal de Itajubá.*

Palavras-chave: Arquitetura e Organização de Computadores. Computação de Alto Desempenho. RISC-V.

Introdução

O conjunto de instruções do RISC-V é um conjunto de instruções gratuito e aberto, desenvolvido na Universidade da Califórnia em Berkeley, designado ao desenvolvimento do hardware *open-source* no contexto da arquitetura e organização de computadores. O RISC-V possui um conjunto base de instruções "I" e extensões opcionais que adicionam outras instruções mais complexas, para processadores de 32, 64 e 128 bits, em que a modularidade do ISA permite uma grande variedade nas tecnologias desenvolvidas na área, a depender da aplicação específica em que é essencial a utilização de circuitos microprocessadores (PATTERSON & WATERMAN, 2019). O objetivo deste trabalho, é analisar a implementação das instruções de multiplicação para um processador RISC-V de 32 bits, que estão contidas na extensão "M", na qual os diferentes algoritmos de multiplicação resultam em diferentes implementações dentro dos circuitos digitais e por consequência, diferentes resultados em termos de desempenho e área do processador. É também necessário ressaltar, que não foram implementadas e analisadas as instruções privilegiadas "Zicsr", em que a arquitetura do microprocessador é não privilegiada, sem o tratamento de exceções ou mudanças no modo de operação.

Metodologia

O processador RISC-V com o conjunto base RV32I e o processador com as instruções de multiplicação RV32IM assim como os circuitos aritméticos de multiplicação binária, foram desenvolvidos em Verilog HDL e sintetizados utilizando o Vivado Design Suite 24.1 da Xilinx para a FPGA Artix-7 XCA750T-3CSG324C para o maior *speed grade* permitido. Foram analisados a máxima frequência de *clock*, a latência, e o consumo de elementos lógicos para os algoritmos de multiplicação e para ambos os processadores, em que os algoritmos de multiplicação escolhidos foram o multiplicador de Vedic apresentado na figura 1, que se baseia em operações cruzadas e verticais de multiplicação, resultando em um circuito digital puramente combinacional, e o multiplicador *Shift-Add* que se baseia em somas e deslocamentos sucessivos, resultado em um circuito digital sequencial como mostra a figura 2. A partir dos testes realizados, o multiplicador de Vedic apresentou um melhor desempenho se comparado ao multiplicador *Shift-Add* e portanto, foi o multiplicador escolhido para a implementação das instruções de multiplicação no processador RV32IM.

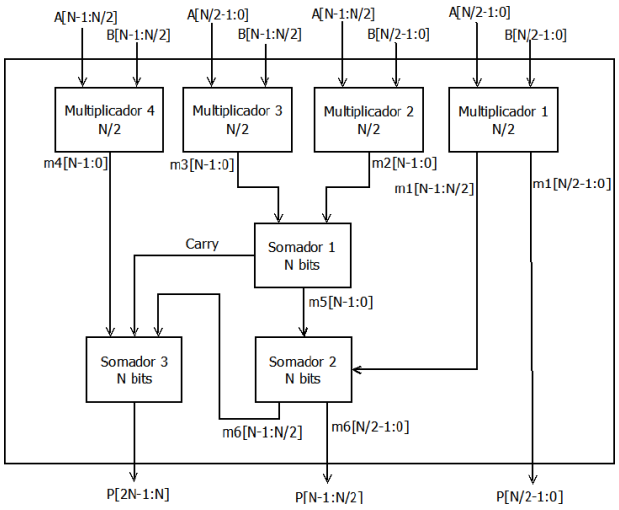

Figura 1: Multiplicador de Vedic de N bits (OLIVEIRA, MORENO, DE OLIVEIRA DUTRA, & PIMENTA, 2017)

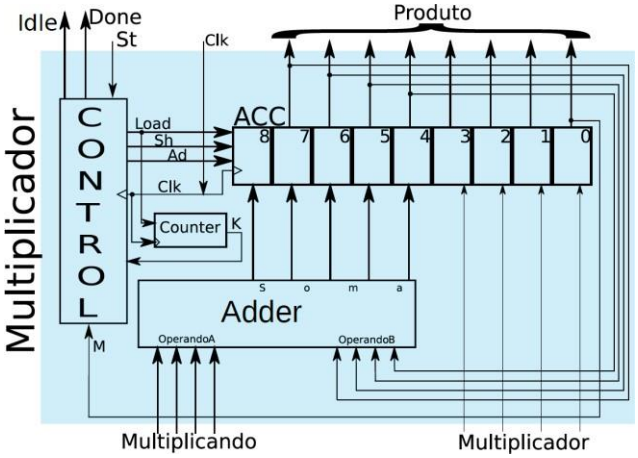

Figura 2: Diagrama de Blocos do Multiplicador Shift-Add (ITAJUBÁ, 2024)

Resultados e discussão

Os resultados obtidos para os circuitos multiplicadores utilizando o algoritmo *Shift-Add* e o algoritmo Vedic, estão presentes na tabela 1, em que o multiplicador de Vedic apresenta uma latência consideravelmente menor apesar da menor frequência de *clock*, porém em contrapartida apresenta um número consideravelmente maior de elementos lógicos.

Tabela 1: Resultados dos circuitos multiplicadores

Algoritmo	Frequência	Latência	Elementos lógicos
Vedic	50,86 MHz	39,31 ns	1983
Shift-Add	201,90 MHz	326,89 ns	93

Os resultados obtidos para os processadores RISC-V com ambos conjuntos de instruções estão presentes na tabela 2, em que o algoritmo escolhido para a implementação das instruções de divisão foi o algoritmo de Vedic devido a sua menor latência, apesar do grande consumo de elementos lógicos.

Tabela 2: Resultados para ambos processadores

Conjunto de instruções	Frequência	Elementos lógicos
RV32I	67,31 MHz	1219
RV32IM	41,89 MHz	3180

Os testes de verificação das instruções do conjunto base e de multiplicação foram realizados por meio da ferramenta de simulação do Vivado Design Suite 24.1, em que para o processador RV32I o código de teste é o código presente na figura 3 e para o RV32IM é o código presente na figura 4.

```
#       RISC-V Assembly         Description
main:   addi x2, x0, 5          # x2 = 5
        addi x3, x0, 12         # x3 = 12
        addi x7, x3, -9         # x7 = (12 - 9) = 3
        or   x4, x7, x2         # x4 = (3 OR 5) = 7
        and  x5, x3, x4         # x5 = (12 AND 7) = 4
        add  x5, x5, x4         # x5 = 4 + 7 = 11
        beq  x5, x7, end        # shouldn't be taken
        slt  x4, x3, x4         # x4 = (12 < 7) = 0
        beq  x4, x0, around     # should be taken
        addi x5, x0, 0          # shouldn't execute
around: slt  x4, x7, x2         # x4 = (3 < 5) = 1
        add  x7, x4, x5         # x7 = (1 + 11) = 12
        sub  x7, x7, x2         # x7 = (12 - 5) = 7
        sw   x7, 84(x3)         # [96] = 7
        lw   x2, 96(x0)         # x2 = [96] = 7
        add  x9, x2, x5         # x9 = (7 + 11) = 18
        jal  x3, end            # jump to end, x3 = 0x44
        addi x2, x0, 1          # shouldn't execute
end:    add  x2, x2, x9         # x2 = (7 + 18) = 25
        sw   x2, 0x20(x3)       # [100] = 25
done:   beq  x2, x2, done       # infinite loop
```

Figura 3: Código de testes para o procesador RV32I (HARRIS & HARRIS, 2021)

```
_boot:

    addi x1, x0, 243    // places 243 on x1
    addi x2, x0, -243   // places -243 on x2
    // unsigned x unsigned
    mul x3, x1, x1
    mulhu x4, x1, x1
    // signed x unsigned
    mul x5, x1, x2
    mulhsu x6, x1, x2
    // signed x signed
    mul x7, x2, x2
    mulh x8, x2, x2
```

Figura 4: Código de testes para o processador RV32IM

A arquitetura do processador RISC-V RV32IM no *pipeline de 5 estágios* é mostrada na figura 3, e os resultados obtidos por meio da ferramenta de simulação do Vivado Design Suite 24.1 para o processador RV32I com as instruções do conjunto base e para o processador RV32IM com as instruções de multiplicação, estão presentes nas figuras 5, 6,7 e 8 respectivamente. Percebe-se pelas simulações que ambos os processadores executaram corretamente as instruções presentes nos códigos de teste, em que os códigos foram convertidos para código de máquina em binário de acordo com as especificações do ISA do RISC-V, e descarregados na memória de programa para que a devida execução. Os sinais ALUResult, Result, PC, WriteData e MemWrite, significam respectivamente o resultado calculado pelos circuitos aritméticos, contador de programa, barramento de escrita que será enviado à memória de dados e habilitação da escrita na memória de dados.

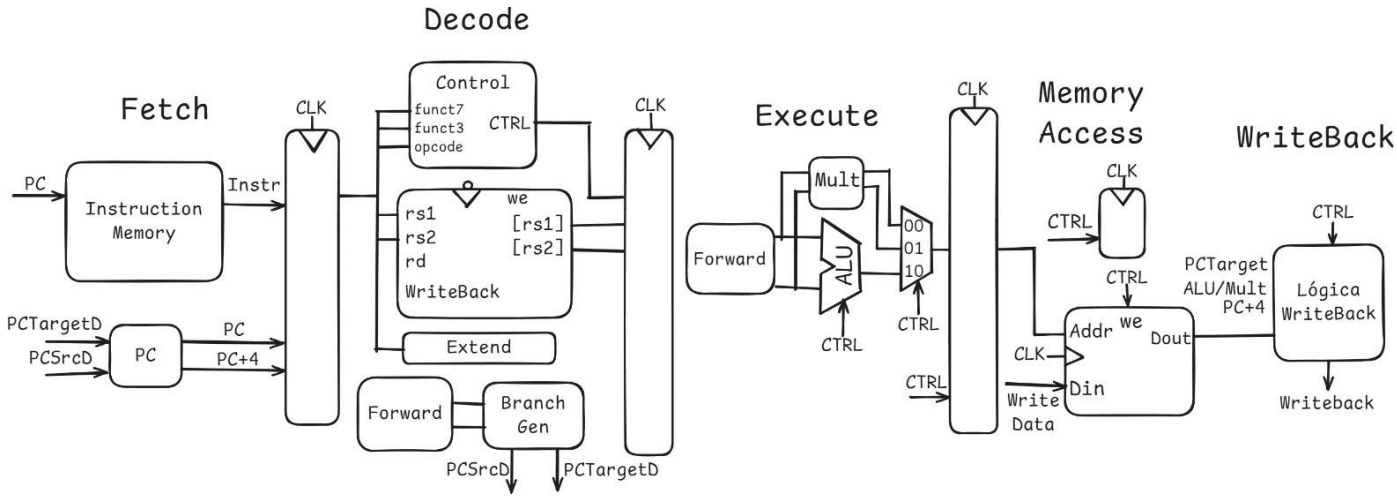

Figura 5: Arquitetura do processador RISC-V RV32IM

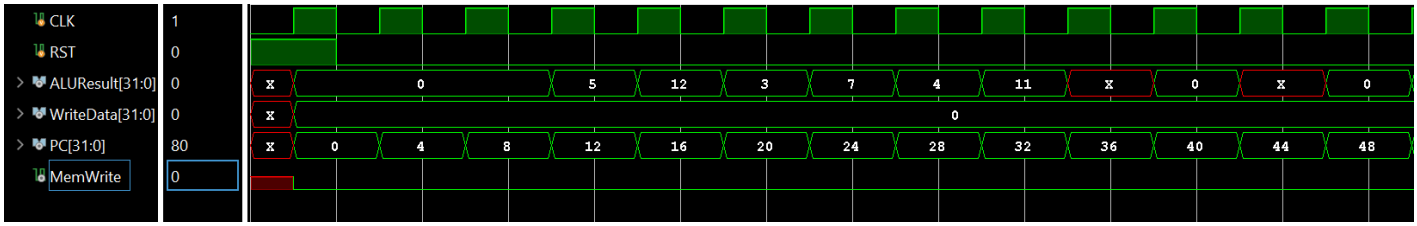

Figura 6: Teste das instruções do RISC-V RV32I parte 1

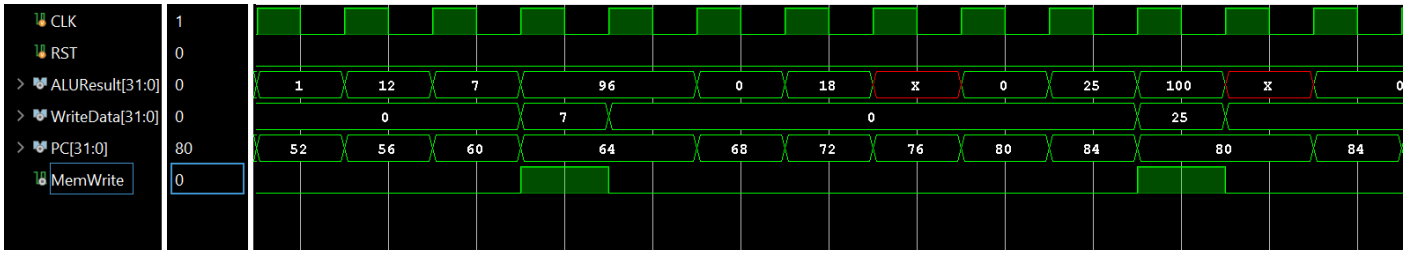

Figura 7: Teste das instruções do RISC-V RV32I parte

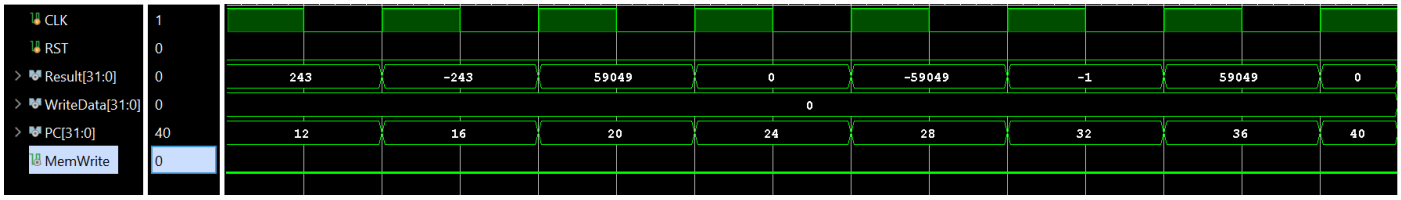

Figura 8: Teste das instruções de multiplicação do RISC-V RV32IM

Conclusões

Os resultados indicam que a inserção das instruções de multiplicação produz efeitos indesejados no desempenho e área dos processadores, devido a maior complexidade das operações. O uso de algoritmos sequenciais, como o multiplicador Shift-Add, resulta em uma menor utilização de elementos lógicos, mas provoca uma significativa redução na frequência de clock do processador. Em contrapartida, os multiplicadores combinacionais como o multiplicador Vedic, apresentam latência inferior, oferecendo melhor desempenho em comparação aos multiplicadores sequenciais. No entanto, eles acarretam um aumento considerável no consumo de elementos lógicos e, consequentemente, na área total do circuito. Trabalhos futuros podem investigar técnicas avançadas de pipelining para integrar instruções que requerem múltiplos ciclos de clock, visando otimizar o desempenho do processador.

Agradecimentos

Os autores agradecem a Unifei e ao Programa Institucional Voluntário de Iniciação Científica PIVIC, pela oportunidade fornecida de aprofundamento dos conhecimentos. Os autores agradecem também ao professor Maurílio Pereira Coutinho, pelas discussões e orientações que contribuíram para o desenvolvimento deste trabalho.

Referências

1. WATERMAN, A.; ASANOVIC, K. *The RISC-V Instruction Set Manual Volume I: Unprivileged.sty*. [S.l.], 2019. Disponível em: <https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf>.

2. OLIVEIRA, J. G. M.; MORENO, R. L.; DE OLIVEIRA DUTRA, O.; PIMENTA, T. C. "Implementation of a reconfigurable neural network in FPGA." In: International Caribbean Conference on Devices, Circuits and Systems (ICCDCS), 2017, Cozumel, Mexico. Anais... New York: IEEE, 2017. p. 41-44. DOI: 10.1109/ICCDCS.2017.7959699.

3. HENNESSY, John L.; PATTERSON, David A. **Computer Architecture: A Hardware/Software Interface - RISC-V Edition**. 1. ed. Cambridge, MA: Morgan Kaufmann, 2020.

4. HENNESSY, John. Arquitetura de Computadores - Uma Abordagem Quantitativa. Rio de Janeiro: Grupo GEN, 2019. E-book. ISBN 9788595150669. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595150669/. Acesso em: 29 ago. 2024.

5. UNIVERSIDADE FEDERAL DE ITAJUBÁ. ELTD05 – Sistemas Digitais. 2024. Disponível em: https://odutra.unifei.edu.br/eltd05/. Acesso em: 6 set. 2024.