VIII Simpósio de Iniciação Científica 2025

"Do conhecimento acadêmico à transformação sustentável: inovação com validação científica"

ESTUDO DE VIABILIDADE DE IMPLEMENTAÇÃO DE UM SOFTCORE RISC-V MULTI-CICLO EM FPGA

Maria Clara Rodrigues Ribeiro[1] (IC), Gustavo Della Colletta (PQ)[1]
[1]*Universidade Federal de Itajubá.*

Palavras-chave: Análise de desempenho. Arquitetura de computadores. RISC-V. Verilog HDL.

Introdução

O RISC-V é uma *Instruction Set Architecture* (ISA) gratuita e aberta, desenvolvida para ser de simples implementação. Devido à sua natureza aberta, permite que qualquer pessoa ou instituição o utilize ou modifique sem custos. Por ser modular, o conjunto de instruções RISC-V possibilita a adição de extensões opcionais ao conjunto base de instruções (RV32I) de acordo com as necessidades da aplicação, garantindo alta flexibilidade e adaptabilidade. (PATTERSON; WATERMAN, 2019). O presente trabalho utiliza o conjunto base RV32I para implementação e análise do projeto.

A arquitetura *multi-cycle* executa instruções em múltiplos ciclos de clock, reutilizando unidades funcionais em diferentes etapas, controladas através de uma máquina de estados finita (FSM). Essa abordagem reduz a quantidade de *hardware* necessário em comparação com arquiteturas *single-cycle*, que executam qualquer instrução em um único ciclo de clock, mas aumenta o número de ciclos necessários por instrução.

O uso crescente do RISC-V, elevado por seu caráter aberto e gratuito e pela comunidade ativa que o mantém e atualiza, reforça a relevância de estudar a implementação desse conjunto em diferentes plataformas, sendo uma escolha estratégica graças à sua flexibilidade. No contexto acadêmico, o *softcore multi-cycle* permite uma compreensão aprofundada do fluxo de dados e do impacto da escolha da arquitetura sobre o desempenho e uso de recursos de FPGA.

Dessa forma, este trabalho tem como objetivo geral avaliar o potencial de implementação de uma arquitetura RISC-V *multi-cycle* em FPGA, e como objetivos específicos: implementar a arquitetura e realizar simulações com instruções do conjunto RIV32I, a fim de analisar o desempenho obtido. Para tal, foi utilizada a linguagem Verilog HDL e o software Quartus para desenvolver a arquitetura e realizar simulações, juntamente com o software Vivado.

Metodologia

O presente trabalho iniciou-se com um período dedicado ao estudo da ISA RISC-V, com enfâse no formato de cada instrução do conjunto base RV33I (R, I, S, B, U, J). Esse estudo englobou também a compreensão do funcionamento das microarquiteturas *single-cycle* e *multi-cycle* e suas difenças com relação ao re-uso de estruturas funcionais, como a ULA, e da forma de controle, sendo a primeira controlada por lógica combinacional a partir da instrução e a segunda por uma máquina de estados finita para controlar os vários ciclos de execução de uma instrução.

A primeira implementação foi da arquitetura *single-cycle*, desenvolvida no software Quartus Prime Lite Edition 18.1, em linguagem Verilog HDL. A arquitetura desenvolvida foi simulada utilizando o conjunto de instruções mostrado na Figura 1, para a FPGA MAX 10M50DAF484C7G. O código de testes envolve uma sequência de intruções de *load word* (carregar dados da memória), *store word*(salvar dados na memória),*branch* (desvios no código), etc. A penúltima instrução (sw x2, 0x20(x3)), armazena o valor de x2 no endereço x3 + 0x20. Se as instruções anteriores forem executadas corretamente, x3 terá 0x44 e x2, 25. Logo, verifica-se sucesso de execução se o valor 25 foi armazenado na posição 0x64 (0x44+0x20) da memória

A partir dessa simulação, foram obtidos dados de frequência máxima de clock e consumo de elementos lógicos no circuito.

Após validada a simulação, iniciou-se o desenvolvimento da arquitetura *multi-cycle*, também em Verilog HDL. Para esta arquitetura, utilizou-se novamente o conjunto de instruções apresentado na Figura 1 e o mesmo modelo de FPGA. Extraiu-se os mesmos parâmetros de desempenho obtidos para o *single-cycle*.

Para fins de análise de desempenho em diferentes plataformas, ambas as arquiteturas foram sintetizadas e simuladas na ferramenta Vivado Design Suite 24.1, para FPGA's da fabricante Xilinx. Para isso, foi escolhida a FPGA Artix-7 XC7A50T-3CSG324C, a qual apresenta o

speed grade -3, o mais rápido. Manteve-se o mesmo conjunto de instruções de teste e mesmos parâmetros captados, permitindo comparação dos resultados.

Figura 1- Código de teste (HARRIS & HARRIS, 2021).

```
#       RISC-V Assembly         Description             Address   Machine Code
main:   addi x2, x0, 5          # x2 = 5                0         00500113
        addi x3, x0, 12         # x3 = 12               4         00C00193
        addi x7, x3, -9         # x7 = (12 - 9) = 3     8         FF718393
        or   x4, x7, x2         # x4 = (3 OR 5) = 7     C         0023E233
        and  x5, x3, x4         # x5 = (12 AND 7) = 4   10        0041F2B3
        add  x5, x5, x4         # x5 = 4 + 7 = 11       14        004282B3
        beq  x5, x7, end        # shouldn't be taken    18        02728863
        slt  x4, x3, x4         # x4 = (12 < 7) = 0     1C        0041A233
        beq  x4, x0, around     # should be taken       20        00020463
        addi x5, x0, 0          # shouldn't execute     24        00000293
around: slt  x4, x7, x2         # x4 = (3 < 5) = 1      28        0023A233
        add  x7, x4, x5         # x7 = (1 + 11) = 12    2C        005203B3
        sub  x7, x7, x2         # x7 = (12 - 5) = 7     30        402383B3
        sw   x7, 84(x3)         # [96] = 7              34        0471AA23
        lw   x2, 96(x0)         # x2 = [96] = 7         38        06002103
        add  x9, x2, x5         # x9 = (7 + 11) = 18    3C        005104B3
        jal  x3, end            # jump to end, x3 = 0x44 40       008001EF
        addi x2, x0, 1          # shouldn't execute     44        00100113
end:    add  x2, x2, x9         # x2 = (7 + 18) = 25    48        00910133
        sw   x2, 0x20(x3)       # [100] = 25            4C        0221A023
done:   beq  x2, x2, done       # infinite loop         50        00210063
```

Os valores obtidos, bem como suas análises e comparações, são apresentados na seção a seguir.

Resultados e discussão

A arquitetura *single-cycle* desenvolvida é ilustrada pelo diagrama RTL apresentado na Figura 2. As formas de onda obtidas na simulação para estão indicadas pelas Figuras 3 e 4.

Figura 2 – Diagrama RTL para arquitetura *single-cycle*

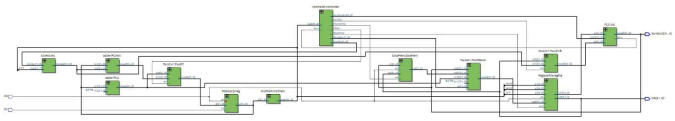


Figura 3 – Formas de onda do teste do *single-cycle* pt.1

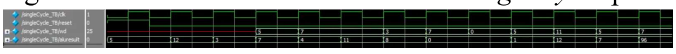


Figura 4 – Formas de onda do teste do *single-cycle* pt.2

O diagrama RTL obtido para a arquitetura *multi-cycle* é indicado na Figura 5, e o diagrama de estados da FSM que o controla na Figura 6. As formas de onda obtidas com a simulação estão nas Figuras 7 e 8.

A FSM do multicycle (Figura 6) apresenta inicialmente os estados s0 e s1, referentes, respectivamente, ao *fetch* e *decode* da instrução, comuns a todos os tipos. São neles que a instrução é carregada da memória e decodificada, identificando o tipo de instrução à ser executada. Os próximos estados são percorridos de acordo com a instrução definida em s1: s2 calcula o endereço de memória para *store word* e *load word*; s3 realiza a leitura desse endereço para *load word*; s4 grava o valor lido em s3 em registro; s5 armazena no endereço calculado em s2 o valor de um registro(*store word*); s6 realiza operações da ULA entre registros; s7 grava no registro o valor obtido em s6, s8 ou s9; s8 opera a ULA para instruções com valor imediato; s9 calcula e atualiza o endereço de *jump*; e s10 verifica a condição de *branch*.

Figura 5 – Diagrama RTL para arquitetura *multi-cycle*

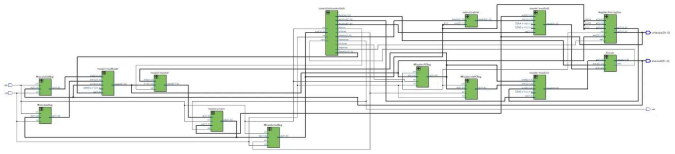


Figura 6 – Diagrama de estados da FSM do *multi-cycle*

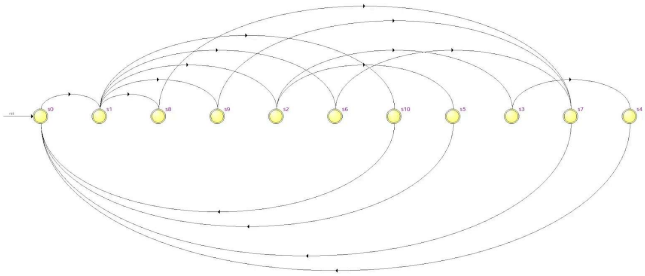


Figura 7 – Resultado do teste do *multi-cycle* pt.1

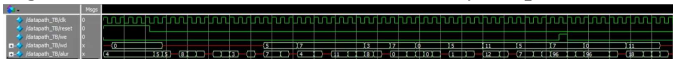
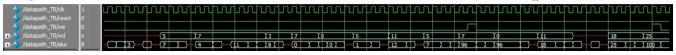


Figura 8 – Resultado do teste do *multi-cycle* pt.2



Verifica-se nas Figuras 2 e 5 que o *single-cycle* possui elementos duplicados, como duas memórias e dois somadores, além de uma ULA e três multiplexadores, utilizando apenas um registrador intermediário (flip flop). Já o *multi-cycle*, apesar de reduzir os módulos duplicados a uma ULA e uma memória, requer cinco registradores intermediários e quatro multiplexadores, para controle e armazenamento dos valores durante a execução da instrução.

Como descrito na seção anterior, a validação do programa de testes é confirmada pelo armazenamento do valor 25 no endereço 100 (0x64) da memória. Para essa verificação, monitorou-se os sinais *aluresult* (endereço calculado para a penúltima instrução), e *wd*

(dado a ser escrito neste endereço). Ao fim da simulação em ambas as arquiteturas, indicada nas Figuras 3, 4, 7 e 8, wd tem valor 25 e aluresult, 100, comprovando funcionamento correto das arquiteturas, permitindo sua análise.

Os parâmetros de frequência máxima de clock (Fmáx) e consumo de recursos lógicos na FPGA Intel MAX 10M50DAF484C7G para ambas as arquiteturas estão dispostos na Tabela 1:

Tabela 1 – Fmáx e uso de recursos na FPGA MAX 10

Arquitetura	Fmáx	Consumo	Uso
single-cycle	49.14 MHz	4.208	8%
multi-cycle	70.58 MHz	3.194	6%

Os dados apresentados na Tabela 1 evidenciam que a arquitetura *multi-cycle* atinge frequência máxima de 70,58 MHz, 1,44 vezes maior que a *single-cycle* (49,14 MHz), para a mesma FPGA. Isso ocorre devido ao tempo do caminho crítico: a instrução mais lenta (*load word*) é executada em um único ciclo no *single-cycle*, que concentra diversas operações — ler e decodificar a instrução, acessar registradores e memória, operar na ULA e escrever em registrador — resultando em um atraso de 20,35 ns. No *multi-cycle*, as etapas são feitas em diferentes ciclos, diminuindo o tempo crítico em cada etapa para 14,17 ns, o que permite maior frequência de operação.

Em relação ao consumo, o *multi-cycle* usa 6% dos recursos lógicos (3.194) frente aos 8% consumidos pelo *single-cycle* (4.219). Isso se explica pela reutilização da ULA e da memória unificada para dados e instruções. O *single-cycle*, para executar todos os processos em um único ciclo, requer *hardware* adicional, como um somador dedicado para calcular o próximo endereço de instrução e um para a lógica de *branch*, o que aumenta o consumo total de recursos.

O consumo total de recursos apresentado no Quartus indica a quantidade de LEs(*logic elements*) utilizados. Um LE é a menor unidade lógica da arquitetura da MAX 10, composto de um LUT(*look-up table*) e um registrador programável, que pode ser mapeado para flip-flop (INTEL CORPORATION, 2025). A soma total desse consumo é composta por: LEs usados apenas como lógica combinacional, somente como registradores ou com uso simultâneo de LUT e registrador. Os valores indicados para cada um desses usos nas duas arquiteturas estão indicados na Tabela 2.

Tabela 2 – Distribuição do uso de LEs para a MAX 10

Arquitetura	Lógica	Registradores	Ambos
single-cycle	1.115	1401	1.703
multi-ciclo	749	1.029	1.406

A diferença entre os LEs utilizados para esses três fins confirma o maior consumo do *single-cycle*, principalmente devido à presença dos somadores dedicados necessários nessa arquitetura.

Para a simulação das duas arquiteturas na FPGA Artix-7 XC7A50T-3CSG324C, os dados obtidos da síntese e simulação no Vivado estão indicados na Tabela 3.

Tabela 3 – Fmáx e uso de recursos na FPGA Artix-7

Arquitetura	Fmáx	Consumo	Uso
single-cycle	100 MHz	450	1.38%
multi-cycle	123.76 MHz	928	2.85%

Na Artix-7, o consumo de recursos refere-se diretamente aos LUTs utilizados, sem contabilizar registradores, o que diferencia a metodologia de contagem das FPGAs usadas, dificultando comparação direta. A fim de aproximar a análise, desconsidera-se os LEs utilizados apenas como registro na MAX 10. Com isso, tem-se que o total de recursos consumidos em MAX 10 para o *multi-cycle* é de 4.33% (2.155), porcentagem próxima à obtida para a Artix, demonstrando desempenho semelhante nos dois dispositivos.

Para a arquitetura *single-cycle*, entretanto, mesmo considerando apenas os LEs usados com lógica combinacional (1.115+1.703), a porcentagem de consumo na MAX 10 (5,33%) é consideravelmente maior que na Artix-7 (1.38%). Além disso, o consumo de recursos apresentado para a arquitetura *single-cycle* foi menor que o *multi-cycle*, mesmo com a necessidade de *hardware* a mais, comportamento contrário ao demonstrado para a MAX 10.

Isso decorre da arquitetura interna da própria Artix-7. Seu CLB (*configurable logic block*) apresenta muxes internos e cadeias de carry dedicadas (XLINX, 2019), que são priorizados na síntese de arquiteturas que envolvem, por exemplo, somadores e muxes com grandes entradas. O uso destes elementos dedicados não é contabilizado no consumo total de recursos apresentado após a síntese.

Logo, ao sintetizar o *single-cycle*, os somadores adicionais que ele possui são mapeados para essas estruturas e, com isso, não são contabilizados no consumo total. No *multi-cycle*, entretanto, o acréscimo de LUTs devido à implementação da FSM de controle não se beneficia desses recursos internos, tornando essa arquitetura relativamente mais custosa em termos de consumo lógico quando sintetizada na Artix-7. Ou seja, a diferença do consumo lógico causada pela FSM se evidencia na síntese para Artix, visto que o *hardware* necessário a mais para a arquitetura *single-cycle* utiliza os elementos internos do CLB da Artix, e não são totalmente contabilizados para consumo de recursos.

Além disso, a arquitetura *multi-cycle* nas diferentes placas apresentou variações de frequência entre as duas FPGAs. A frequência máxima de operação obtida para a Artix-7 foi 123,76 MHz, equivalente a um atraso de caminho mínimo de 8,08 ns, 1,75 vezes mais rápida que a atingida pela MAX 10 (70,58 MHz, 14,17 ns de atraso). Isso se explica pela diferença na capacidade de cada FPGA, em que a Artix alcança uma frequência máxima de buffer global(BUFG) de até 628 MHz enquanto a MAX suporta até 416MHz, indicando maior potencial da Artix-7 para operar frequências mais altas.

Conclusões

Os resultados confirmam a viabilidade de implementação do *softcore* RISC-V nos dois dispositivos analisados. Em ambas as FPGAs, o consumo de recursos manteve-se baixo (menor que 9% na MAX10 e menir que 3% de LUTs na Artix-7), indicando possibilidade de expansão da arquitetura para adicionar periféricos e ampliar o conjunto de instruções passível de ser executado.

A arquitetura provou-se portável e flexível: foi descrita uma única vez em Verilog HDL e sintetizada no Quartus (Intel MAX 10 10M50DAF484C7G) e no Vivado (Xilinx Artix-7 XC7A50T-3CSG324C), comprovando que o projeto é enxuto e aplicável a diferentes famílias.

Na MAX 10, o *multi-cycle* apresentou frequência máxima maior (70,58 MHz vs 49,14 MHz) e menor uso de LEs (6% vs 8%) que o *single-cycle*. Isso se explica pelo menor caminho crítico ao particionar a execução em etapas e pelo reuso de operadores (ALU/somadores), reduzindo a adição de *hardware* do *single-cycle*.

Na Artix-7, o *multi-cycle* usou mais LUTs que o *single-cycle* (2,85% vs 1,38%). Isso é coerente com a arquitetura do CLB da Xilinx, que possui cadeias de carry e multiplexadores internos: somadores e parte da seleção são mapeados para esses recursos dedicados (sem "custar" LUTs extras), de modo que mais *hardware* no *single-cycle* pesa pouco em LUTs, enquanto o *multi-cycle* aumenta rede de controle para viabilizar o reuso, consumindo LUTs. Além disso, as ferramentas contam recursos de forma diferente (LE = LUT+FF no Quartus; Vivado contabiliza LUTs e FFs separadamente). Equiparando a MAX 10 para LEs combinacionais (4,33%), o consumo se aproxima da Artix-7 (2,85%).

Quanto ao desempenho geral, embora o *multi-cycle* atinja frequência maior, ele executa o conjunto de testes em 69 ciclos, enquanto o *single-cycle* o faz em 17. Assim, o tempo total do teste foi menor no *single-cycle* (na MAX 10: 0,35 µs no *single-cycle* vs ~0,98 µs no *multi-cycle*), evidenciando as vantagens e desempenho de ambas arquiteturas em diferentes métricas.

Em suma, o projeto é viável, felxível e escalável. Com a folga observada no consumo, há espaço para evoluir a microarquitetura mantendo custos baixos de implementação nas duas famílias.

Agradecimentos

Agradeço à Universidade Federal de Itajubá (UNIFEI) pela oportunidade de realização deste trabalho de iniciação científica, que possibilitou o aprofundamento dos conhecimentos em microarquitetura e eletrônica digital, enriquecendo minha formação acadêmica e profissional.

Referências

PATTERSON, David; WATERMAN, Andrew. *Guia prático RISC-V: atlas de uma arquitetura aberta*. 1. ed. [S.l.]: RISC-V Foundation, 2019. Disponível em: https://biblioteca.univali.br/pergamumweb/vinculos/pdf/Guia%20prático%20RISC-V%20-%20atlas%20de%20uma%20arquitetura%20aberta.pdf. Acesso em: 18 ago. 2025.

HARRIS, Sarah L.; HARRIS, David Money. *Digital Design and Computer Architecture: RISC-V Edition*. 2. ed. Cambridge: Morgan Kaufmann, 2021.

INTEL CORPORATION. *MAX 10 FPGA Device Overview*. San Jose, CA: Intel, 2025. Disponível em: https://www.intel.com/programmable/technical-pdfs/max10-handbook.pdf. Acesso em: 18 ago. 2025.

XILINX. *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics*. DS181 (v1.19). San Jose, CA: Xilinx, 2019. Disponível em: https://www.digikey.com.br/htmldatasheets/production/2053661/0/0/1/xc7a100t-2fgg676ces9937.pdf. Acesso em: 18 ago. 2025.