

Computação Aproximada no Processador RISC-V NEORV32

Jailton Nunes Filho¹, Pedro e Silva¹, Hytalo Bassi¹, Eduardo Paes¹,
Jhullya Martins¹, Daniela Catelan¹

¹ Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (FACOM - UFMS)
Av. Costa e Silva, s/n – Campo Grande – MS – Brasil

{jailton_nunes_filho, hytalo_bassi, pedro.candia}@ufms.br

{eduardo_paes, jhullya.e.s.martins, daniela.catelan}@ufms.br

Abstract. *This article describes the methodology used to apply Approximate Computing in the RISC-V NEORV32 processor, making use of the CFS present in the softcore architecture to insert the approximation and utilize it in tests with the algorithm: Loop Perforation. The Quartus tools exposed that the current consumption in the processor core had reductions of 0.29% and 0.49%, a decrease of 240 logic elements, at a cost of 2.51% in the maximum frequency.*

Resumo. *Este artigo descreve a metodologia utilizada para aplicar a Computação Aproximada no processador RISC-V NEORV32, tomando uso do CFS presente na arquitetura do softcore para inserir a aproximação e aproveitá-la em testes com o algoritmo: Loop Perforation. As ferramentas do Quartus expuseram que o consumo da corrente no núcleo do processador teve reduções de 0,29% e 0,49%, uma diminuição de 240 elementos lógicos, a um custo de 2,51% na frequência máxima.*

1. Introdução

A miniaturização dos transistores trouxe consigo desafios inéditos, como o aumento exponencial da corrente de fuga. Esse fenômeno impôs a limitação conhecida como dark silicon. Diante desse cenário, tornou-se urgente buscar abordagens que conciliem desempenho e eficiência energética.

A Computação Aproximada (CA) [Leon et al. 2023] surge como uma resposta a esse desafio ao admitir que nem todas as aplicações exigem precisão absoluta, torna-se possível introduzir erros pequenos e controlados, reduzindo o consumo de energia, área e tempo de execução.

As instruções aproximadas [Catelan 2025] surgem como uma estratégia promissora ao permitir a execução de operações com precisão reduzida de forma controlada. As instruções aritméticas aproximadas já demonstraram resultados promissores em termos de eficiência, especialmente em aplicações tolerantes a erro.

Nesse contexto, a arquitetura RISC-V [RISC-V 2022], por ser aberta, modular e extensível, oferece um ambiente ideal para a experimentação com instruções aproximadas. Em especial, o processador NEORV32 [Nolting 2026] destaca-se como uma plataforma flexível e de fácil customização, permitindo investigar diferentes níveis de aproximação e analisar seus impactos sobre desempenho, consumo de recursos e precisão. Assim, este trabalho propõe a experimentação da CA no processador RISC-V NEORV32, com ênfase na introdução e avaliação de instruções aritméticas aproximadas.

2. Metodologia

Para introduzir a aproximação na arquitetura do NEORV32 sem afetar cálculos importantes como os de endereços de memória realizados pela Unidade Lógica Aritmética (ULA), foi utilizado o *Custom Functions Subsystem* (CFS) presente no arquivo VHDL, `neorv32_cfs.vhd`. Este subsistema oferece 64 kb de espaço de endereçamento mapeado em memória que podem ser utilizados nas memórias customizadas e na interface de registradores, sendo a alternativa encontrada para que a lógica aproximada não afete todas as instruções dependentes da soma ou da subtração da ULA.

Dois arquivos externos à arquitetura tradicional do *softcore* foram adicionados. Esses são pacotes que implementam a lógica da Soma Aproximada (ADDX) e Subtração Aproximada (SUBX) [Catelan 2025]. O trecho de Código 1 exibe como os registradores do CFS foram integrados para executarem a matemática exata e aproximada no `neorv32_cfs.vhd`.

```
1 opa_add_aprox <= cfs_reg_wr(0);
2 opb_add_aprox <= cfs_reg_wr(1);
3 opa_sub_aprox <= '0' & cfs_reg_wr(0);
4 opb_sub_aprox <= '0' & cfs_reg_wr(1); -- 2. Execucao da aproximacao
5 res_add_aprox <= Func_Add_InXA1(opa_add_aprox, opb_add_aprox);
6 res_sub_aprox <= Func_Sub_APSC4(opa_sub_aprox, opb_sub_aprox);
7 -- 3. Definicao das saidas para leitura
8 cfs_reg_rd(4) <= res_add_aprox;
9 cfs_reg_rd(5) <= res_sub_aprox(31 downto 0);
10 cfs_reg_rd(6) <= res_add_exato;
11 cfs_reg_rd(7) <= res_sub_exato;
```

Código 1. Integração dos registradores e execução da aproximação.

A técnica de *Loop Perforation* (LP) [Catelan 2023] afeta diretamente o iterador de estruturas de repetição como o `for` e o `while`. Ao aplicarmos a aproximação nas iterações teremos um efeito parecido ao de incrementar o iterador na computação exata. O trecho de Código 2 mostra como foi implementada a aproximação em um *Script* em C por meio do CFS.

```
1 gp = 1; // grau perfuracao potencia 2 (0, 1, 2, 4, 8)
2 s = (uint32_t)pow(2, gp);
3 for(i = 0; i < n; i = ADDX){
4     soma = soma + 1;
5     NEORV32_CFS->REG[3] = soma;
6     NEORV32_CFS->REG[0] = i;
7     NEORV32_CFS->REG[1] = s;
8     ADDX = NEORV32_CFS->REG[2];
9     if (ADDX <= i){ ADDX = i + s;}} //Skip_factor
```

Código 2. Script em C do Loop Perforation no CFS.

3. Resultados e Discussões

Com o auxílio do Quartus, um *software* de *Electronic Design Automation* (EDA), foi possível obter informações de área, desempenho e consumo do NEORV32 processando a lógica do LP de maneira exata e aproximada, sendo elas expostas na Tabela 1.

Tabela 1. Comparação de área, desempenho e consumo: Exato vs. Aproximado.

Parâmetro / Componente	Exato	Aproximado	Variação (%)
Síntese e Desempenho			
Elementos Lógicos	3646	3406	-6,58
Registradores	1521	1521	0,00
Bits de Memória	99328	99328	0,00
Frequência Máxima (MHz)	104,36	101,74	-2,51
Dissipação Térmica (mW)			
Núcleo Total – Dinâmica	17,00	16,92	-0,47
Núcleo Total – Estática	77,93	77,93	0,00
Bloco Personalizado (CFS)	0,64	0,73	+14,06
Corrente Consumida (mA)			
VCCINT (Núcleo) – Dinâmica	20,90	20,84	-0,29
VCCINT (Núcleo) – Estática	14,19	14,12	-0,49
VCCIO (I/O) – Dinâmica	0,03	0,03	0,00
VCCIO (I/O) – Estática	7,89	7,89	0,00

A aplicação da Computação Aproximada se mostrou eficiente no consumo de corrente do núcleo do processador ao executar a técnica de *Loop Perforation*, pois houve reduções de 0,29% e 0,49% na corrente dinâmica e estática de VCCINT, respectivamente, além de uma diminuição de 240 elementos lógicos (-6,58%), a um custo de uma frequência máxima 2,51% menor.

4. Conclusão

Este trabalho buscou aplicar e comparar o funcionamento da CA no NEORV32, sendo uma das principais conquistas as reduções de 0,29% e 0,49% na corrente consumida. Trabalhos futuros incluem introduzir algoritmos e técnicas além do LP para testar a eficiência da aproximação em diferentes cenários e encontrar novas formas de alterar o processador para receber os pacotes MULX e DIVX.

Agradecimentos

O presente trabalho foi realizado com apoio da Universidade Federal de Mato Grosso do Sul – UFMS/MEC - Brasil

Referências

- Catelan, D. e. a. (2023). Instruction-level loop perforation. In *Anais do XXIV WSCAD*.
- Catelan, D. L. (2025). Approximate computing: Contributions to the design of arithmetic circuits and instruction-set architectures. *Repositório Institucional da UFMS*.
- Leon, V., Hanif, M., Armeniakos, G., Jiao, X., Shafique, M., Pekmestzi, K., and Sou-drís, D. (2023). Approximate computing survey, part i: Terminology and software e hardware approximation techniques.
- Nolting, S. (2026). The neorv32 risc-v processor. <https://github.com/stnolting/neorv32> v1.13.3. Accessed: Aug/2026.
- RISC-V (2022). Risc-v toolchain. Available at: <https://github.com/riscv-collab/riscv-gnu-toolchain>. Accessed: June/2025.