IMPLEMENTAÇÃO DE FUNÇÕES NÃO-LINEARES EM PROCESSADOR SOFT-CORE BASEADO EM FPGA

Tiago de Castro Falcão Guimarães[1] - tiagofalcaoguimaraes@gmail.com
Luciano Manhães de Andrade Filho[1] - luciano.andrade@ufjf.br
Augusto Santiago Cerqueira[1] - augusto.santiago@ufjf.br
Guilherme Barroso Morett[2] - guilherme.morett@iprj.uerj.br
Gustavo Barbosa Libotte[2] - gustavolibotte@iprj.uerj.br
[1]Universidade Federal de Juiz de Fora – Juiz de Fora – MG
[2]Universidade do Estado do Rio de Janeiro, Instituto Politécnico - Nova Friburgo - RJ

Abstract. *Este trabalho descreve a implementação e a comparação de diferentes métodos de aproximação para funções não lineares no processador soft-core de arquitetura reduzida SAPHO. Foram estudadas as funções seno, cosseno, tangente, arco-tangente, exponencial, tangente hiperbólica e sigmoide, empregando três abordagens distintas: séries de Maclaurin, tabelamento e o algoritmo CORDIC. As soluções foram implementadas em duas larguras de dados, 23 e 32 bits e os resultados avaliados segundo o erro médio e a quantidade de ciclos de clock necessários à execução. Os achados indicam que as séries de Maclaurin tendem a fornecer maior precisão, sobretudo na configuração de 32 bits, porém com custo de maior tempo de execução. O tabelamento mostrou-se como uma alternativa que apresenta relação satisfatória entre acurácia e desempenho. Já o CORDIC obteve desempenho inferior no SAPHO, em razão da exigência de elevada precisão e dos erros nos circuitos responsáveis por cálculos em ponto flutuante. As conclusões são sustentadas por gráficos, tabelas e trechos de código que corroboram a análise apresentada.*

Keywords: *FPGA, Aproximações Numéricas, Funções Não Lineares, Processador Embarcado.*

1. INTRODUÇÃO

O *Scalable Architecture Processor for Hardware Optimization* (*SAPHO*) é um *processador soft-core* desenvolvido no Núcleo de Instrumentação e Processamento de Sinais da Universidade Federal de Juiz de Fora. Diferente de outras arquiteturas *PSC*, o *SAPHO* realiza alocação automática de recursos de hardware na etapa de compilação, gerando circuitos otimizados conforme as instruções do código do usuário, e, por ser implementado em *FPGAs*, permite o processamento rápido e eficiente de grande volume de dados com baixo uso de recursos computacionais (21).

Para ampliar o alcance de aplicações e a capacidade de processamento do processador, torna-se necessário ampliar sua biblioteca em C com aproximações para funções não lineares, buscando um equilíbrio entre precisão e velocidade. As rotinas padrão da biblioteca *math.h* não

são adequadas para muitos sistemas embarcados: dependem de suporte de hardware específico (como *floating-point unit* padronizada), envolvem tabelas e lógica complexa para tratar casos especiais e não oferecem controle explícito sobre ciclos de execução ou critérios de término das estimativas. Desenvolver versões nativas dessas funções possibilita implementações portáveis, com precisão ajustável, ciclos determinísticos e redução substancial do tamanho do código, características essenciais em ambientes embarcados (4; 11).

Este trabalho propõe, implementa e compara três estratégias de aproximação, séries de Maclaurin, tabelamento com interpolação e o algoritmo *CORDIC*, aplicadas às funções seno, cosseno, tangente, arco tangente, exponencial, tangente hiperbólica e sigmoide. Cada método foi desenvolvido em C, adaptado para a linguagem do *SAPHO* e avaliado em dois conjuntos de parâmetros do processador, considerando precisão (erro médio) e custo em ciclos de clock.

A organização do artigo é a seguinte: a Seção 2. discute o *SAPHO* e suas particularidades; os Capítulos 3., 4. e 5. apresentam, respectivamente, a fundamentação e as escolhas de implementação para Maclaurin, tabelamento e CORDIC; a Seção 6. reúne a comparação entre os métodos e configurações; e a Seção 7. sintetiza as conclusões e recomendações derivadas dos resultados.

2. O Processador SAPHO

O processador SAPHO, cujo nome significa Scalable Architecture Processor for Hardware Optimization, foi projetado para prover alto desempenho e flexibilidade em aplicações de processamento intensivo (KAPISCH; 21). O projeto surgiu da necessidade de soluções que otimizem a eficiência na execução de tarefas computacionais complexas em hardwares distintos, com ênfase na modularidade e na reconfigurabilidade (21).

A arquitetura do SAPHO é modular e orientada para escalabilidade. A estrutura integra blocos de controle e de dados que permitem a composição de unidades funcionais especializadas e a alocação automática de circuitos conforme as instruções do programa.

A ULA do SAPHO é fortemente parametrizável pelo compilador e por parâmetros de projeto. O compilador sinaliza as operações necessárias e a ferramenta gera automaticamente circuitos específicos dentro da ULA para cada instrução detectada. Essa parametrização permite ajustar a largura de palavra e o formato numérico para equilibrar precisão, uso de recursos e latência (17).

O fluxo de geração de hardware e de inicialização parte de uma IDE que traduz código em um subconjunto de C em Assembly. O funcionamento interno do SAPHO integra a decodificação de instruções, a alocação dinâmica dos recursos necessários e o roteamento dos dados entre as unidades funcionais de modo a minimizar latência. Mecanismos de predição e de verificação de dependências evitam conflitos e reduzem tempos de espera entre estágios de execução, o que melhora o rendimento em cenários com elevada concorrência de tarefas (21).

Em suma, a combinação de arquitetura modular parametrizável geração automática de hardware e mecanismos de controle avançados torna o SAPHO adequado a aplicações como processamento de sinais computação de alto desempenho e sistemas embarcados onde o equilíbrio entre precisão área e consumo é determinante (KAPISCH; 21).

3. Aproximação por séries de Maclaurin

Nesta seção apresenta-se a fundamentação e as estratégias implementacionais baseadas em séries de Maclaurin para aproximar funções não lineares no contexto do processador *SAPHO*. As funções consideradas foram: seno, cosseno, tangente, arco-tangente, exponencial, tangente hiperbólica e sigmoide.

A Série de Taylor é a base teórica para aproximações polinomiais de funções analíticas; a série de Maclaurin corresponde ao caso especial centrado em 0 e é usada aqui para reduzir o custo computacional das aproximações. A sua forma truncada é dada pela equação 1 (3; 19).

$$f(x) \approx \sum_{n=0}^{N} \frac{f^{(n)}(0)}{n!} x^n \tag{1}$$

Para funções trigonométricas, aproveita-se a periodicidade e realiza-se normalização do argumento, por meio da redução do ângulo, para manter $|x|$ pequeno, melhorando a convergência da série. As séries usuais empregadas são apresentadas na equação 2 (6; 20).

$$\sin(x) \approx \sum_{n=0}^{N} (-1)^n \frac{x^{2n+1}}{(2n+1)!}, \qquad \cos(x) \approx \sum_{n=0}^{N} (-1)^n \frac{x^{2n}}{(2n)!}. \tag{2}$$

A iteração eficiente entre termos é feita utilizando a razão entre termos consecutivos não nulos mostrada na equação 3, o que reduz multiplicações e evita o cálculo explícito de fatoriais.

$$\frac{T_{m+2}}{T_m} = -\frac{x^2}{(m+1)(m+2)} \tag{3}$$

A tangente foi obtida, pragmaticamente, pela razão apresentada na equação 4 e quando aplicável evitando divisões quando $\cos(x)$ é muito pequena, embora também exista uma expansão direta em potências envolvendo números de Bernoulli que pode ser utilizada quando conveniente. A expansão por Bernoulli converge lentamente perto dos limites de seu domínio e requer cálculo/armazenamento dos B_{2n} para eficiência subsequente (18; 6).

$$\tan(x) = \frac{\sin(x)}{\cos(x)} \tag{4}$$

Para a arco-tangente, parte-se da derivada mostrada na equação 5 e da expansão geométrica para obter a série da equação 6, válida para $|x| < 1$. Identidades como $\arctan(x) = \frac{\pi}{2} - \arctan(1/x)$ são aplicadas para reduzir argumentos fora do intervalo de convergência e, quando necessário, transformar o argumento para subintervalos como $(-0{,}5, 0{,}5)$ a fim de acelerar a convergência (19; 6).

$$\frac{d}{dx} \arctan(x) = \frac{1}{1+x^2} \tag{5}$$

$$\arctan(x) \approx \sum_{n=0}^{N} \frac{(-1)^n}{2n+1} x^{2n+1}, \qquad |x| < 1 \tag{6}$$

A forma iterativa empregada utiliza a razão entre termos dada na equação 7.

$$\frac{T_{m+2}}{T_m} = -\frac{m}{m+2}x^2 \tag{7}$$

A série da exponencial e^x possui convergência global, conforme a equação 8, o que a torna robusta para aproximações numéricas; a relação recursiva mostrada na equação 9 permite cálculo iterativo com custo controlado (1; 3).

$$e^x \approx \sum_{n=0}^{N} \frac{x^n}{n!} \tag{8}$$

$$\frac{T_{m+1}}{T_m} = \frac{x}{m+1} \tag{9}$$

A tangente hiperbólica e a sigmoide foram tratadas por relações com a exponencial. As fórmulas utilizadas aparecem nas equações 10 e 11, de modo que a expansão de $e^{\pm x}$ fornece aproximações para ambas as funções, garantindo convergência global e comportamento numérico previsível ao ajustar n (11; 10).

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1, \tag{10}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{11}$$

Do ponto de vista implementacional, as escolhas principais foram:

1. normalização do argumento para restringir o domínio de aproximação;
2. critério de parada baseado em tolerância sobre o módulo do próximo termo;
3. aplicação de identidades matemáticas para acelerar a convergência quando necessário.

Essas decisões permitem controlar explicitamente a troca entre precisão (erro médio) e custo (ciclos de clock), facilitando a adaptação dos códigos em C ao ambiente embarcado do *SAPHO* e à avaliação comparativa frente a outras técnicas estudadas no trabalho (16; 4).

4. Tabelamento

Nesta seção apresenta-se a técnica de tabelamento e suas aplicações práticas para aproximação de funções não lineares no contexto do processador SAPHO. A ideia central consiste em pré-computar e armazenar valores discretos de uma função em pontos selecionados, reduzindo a necessidade de cálculo em tempo de execução, o que é vantajoso em sistemas com recursos limitados (15).

O tabelamento armazena valores da função em pontos x_i, com $i = 0, 1, \ldots, N$, e recorre à interpolação quando o argumento solicitado não coincide com um ponto da tabela. A interpolação linear utilizada neste trabalho é referida na equação 12 (5).

A equação 12 expressa a estimativa linear entre os pontos x_i e x_{i+1}, e foi adotada por apresentar baixo custo computacional e por exigir poucas operações aritméticas, característica importante em aplicações embarcadas.

$$f(x) \approx f(x_i) + \left(\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \right) (x - x_i), \tag{12}$$

Para as funções trigonométricas, o tabelamento explora simetrias e periodicidade para reduzir o domínio de armazenamento, por exemplo armazenando valores apenas no intervalo de 0 a $\pi/2$ e recuperando os demais por transformações angulares (6). A interpolação linear então estima valores intermediários com baixo custo de memória.

A função tangente foi obtida a partir da razão entre as aproximações de seno e cosseno, evitando tabelamento direto nas vizinhanças de descontinuidade, onde seria necessário um número elevado de entradas para manter precisão ou o uso de intervalos distintos e de espaçamento irregular para pontos de valor elevado em módulo. Antes da consulta, realiza-se normalização angular para o intervalo de 0 a 2π e, em seguida, projeção para a região de referência utilizada na tabela, o que garante monotonicidade e uso correto da interpolação (20; 8).

A função arco tangente foi tabulada no intervalo de 0 a 1, sendo tratados valores fora desse intervalo por meio da identidade mostrada na equação 13, e valores negativos por paridade, reduzindo assim o espaço de armazenamento necessário (6).

$$\arctan(x) = \frac{\pi}{2} - \arctan\left(\frac{1}{x}\right), \quad x > 1 \tag{13}$$

A função exponencial não apresenta resultados satisfatórios através do tabelamento, devido ao seu crescimento não cíclico, o que leva à necessidade de intervalos artificiais limitados ou ao consumo excessivo de memória; por isso recomenda-se empregar métodos alternativos para e^x no contexto deste trabalho (18).

As funções tangente hiperbólica e sigmoide beneficiam-se de saturação em seus extremos, permitindo tabelamento em intervalos limitados, por exemplo de 0 a 2 para a tangente hiperbólica e de 0 a 6 para a sigmoide, com reconstrução de valores negativos por simetrias apropriadas (10).

As vantagens do tabelamento incluem latência reduzida e simplicidade de implementação, enquanto as limitações envolvem o consumo de memória e perda de precisão fora do refinamento da grade tabulada. A escolha do espaçamento entre pontos e do método de interpolação determina o compromisso entre memória e erro de aproximação, aspecto avaliado em comparação com as demais técnicas estudadas neste trabalho (15).

5. Algoritmo CORDIC

O *COordinate Rotation DIgital Computer* (CORDIC), proposto por J. E. Volder em 1959, é um algoritmo iterativo para aproximação de funções trigonométricas, hiperbólicas e lineares mediante rotações discretas e operações de deslocamento em aritmética fixa (2; 13). A seguir apresenta-se, de forma concisa e referenciada, a dedução e as equações essenciais no modo circular, que foram empregadas na implementação avaliadas neste trabalho. A relação linear que descreve a rotação de um ponto (x_0, y_0) pelo ângulo ϕ é dada pela Equação 14 (14; 2).

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \tag{14}$$

Fatorando o termo $\cos(\phi)$ e adotando rotações elementares com $\tan(\phi_n) = 2^{-n}$, obtém-se a forma conveniente para implementação em hardware, em que o multiplicador por 2^{-n} reduz-se a deslocamentos de bit, mostrada na Equação 15 onde $\phi_n = \arctan(2^{-n})$ e o fator de proporcionalidade consiste no produto cumulativo dos $\cos(\phi_n)$ sobre as iterações (22; 7).

$$\begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix} = \cos(\phi) \begin{bmatrix} 1 & \tan(\phi) \\ -\tan(\phi) & 1 \end{bmatrix}, \tag{15}$$

$$\begin{bmatrix} \cos(\phi_n) & \sin(\phi_n) \\ -\sin(\phi_n) & \cos(\phi_n) \end{bmatrix} \propto \begin{bmatrix} 1 & 2^{-n} \\ -2^{-n} & 1 \end{bmatrix}, \tag{16}$$

O ganho acumulado K_m originado pela multiplicação sucessiva pelas componentes $\cos(\phi_n)$ converge rapidamente. Assim, a normalização inicial por $1/K_\infty$ é adotada na prática para eliminar a necessidade de correção ao fim da aproximação. O limite é dado na Equação 17.

$$\frac{1}{K_\infty} = \lim_{n\to\infty} \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}} \approx 1.6467605. \tag{17}$$

Desenvolvendo a multiplicação matricial elemento a elemento para um passo qualquer e utilizando 2^{-n} como razão entre termos, obtém-se as atualizações iterativas elementares, com sentido horário ou anti-horário tratado pelo sinal apropriado, mostrado na Equação 18. Para uma iteração, $d_n = +1$ se $z_n > 0$ e $d_n = -1$ se $z_n < 0$, e $z_0 = \phi$. O termo $\arctan(2^{-n})$ é obtido a partir de uma tabela pré-computada com N entradas (13; 22).

$$\begin{cases} x_{n+1} = x_n - d_n\, 2^{-n}\, y_n, \\ y_{n+1} = y_n + d_n\, 2^{-n}\, x_n, \\ z_{n+1} = z_n - d_n\, \arctan(2^{-n}), \end{cases} \tag{18}$$

Ao término de N iterações, tendo sido aplicada a normalização inicial por $1/K_\infty$ quando necessário, os componentes do vetor aproximam as funções trigonométricas desejadas, com $x_N \approx \cos(\phi)$ e $y_N \approx \sin(\phi)$ (7; 22).

Os modos vetorial e hiperbólico do CORDIC são conceitualmente análogos ao modo circular: no modo vetorial executa-se rotações de modo a reduzir a componente y a zero com acumulação angular em z, procedimento usado para $\arctan$, enquanto o modo hiperbólico emprega rotações hiperbólicas e tabelas de $\operatorname{artanh}(2^{-n})$ para obter $\sinh$, $\cosh$, $\tanh$, exponenciais e logaritmos. (7; 12)

6. Implementação em Código e Comparação dos Resultados

Para avaliar as aproximações implementadas, isto é, séries de Maclaurin, tabelamento e o algoritmo CORDIC, foram realizadas implementações em C e adaptações para a sublinguagem do processador soft-core SAPHO. As funções trigonométricas, seno, cosseno e tangente, foram avaliadas em radianos no intervalo $[0, 6{,}3]$, para que um período completo fosse incluso nos

testes, enquanto as funções não periódicas, arco tangente, exponencial, tangente hiperbólica e sigmoide, foram avaliadas em $[-2, 2]$. Todas as funções foram testadas com um passo de $0{,}01$. Em cada ponto testado calculou-se o erro absoluto da aproximação em relação ao valor de referência e o número médio de ciclos de clock necessários à convergência.

Os testes consideraram dois conjuntos de parâmetros do SAPHO, que aparecem discriminados nos gráficos e nas tabelas: 32 bits na ULA (1 bit sinal, 8 bits expoente, 23 bits mantissa) 23 bits na ULA (1 bit sinal, 6 bits expoente, 16 bits mantissa). A Figura 1 mostra um desses gráficos, que relaciona os valores de entrada, partindo de $x = 0$, então $x = 0,01$, depois $x = 0,02$, com incrementos $\Delta x = 0,01$ até chegar em $x = 6.3$, em relação ao erro obtido comparando os resultados obtidos com as aproximações deste trabalho com os resultados obtidos utilizando a biblioteca *numpy* da linguagem de programação *Python*.

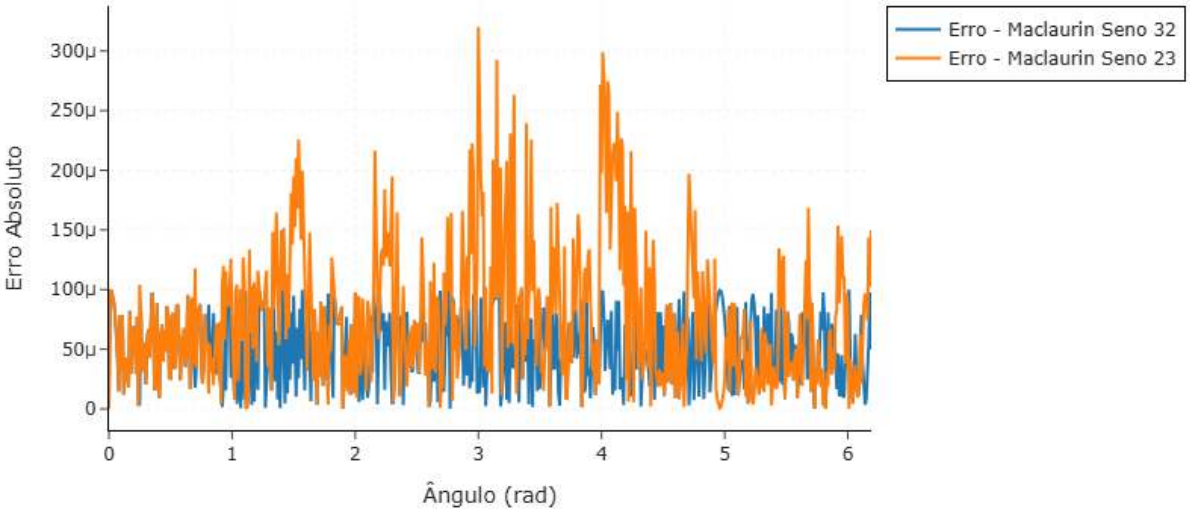

Figure 1: Erro da função seno através de séries de Maclaurin.

Em vez de se apresentar as 40 figuras individuais geradas, foram criados três gráficos agregados que condensam todos os resultados por método, sendo esses os Gráficos 2, 3 e 4. Nelas, estão contidos também os valores para o número médio de ciclos de clock necessário para a convergência, que serve como uma estimativa de quão rapidamente os resultados são obtidos no processador. Esses valores são mostrados nos topos das barras que representam o erro médio de cada método. Assim, na horizontal dos gráficos tem-se: o método utilizado e o número de bits na ULA. No eixo vertical, ou seja, a amplitude das barras, o erro médio calculado. E no topo das barras, o número médio de ciclos de clock para a convergência.

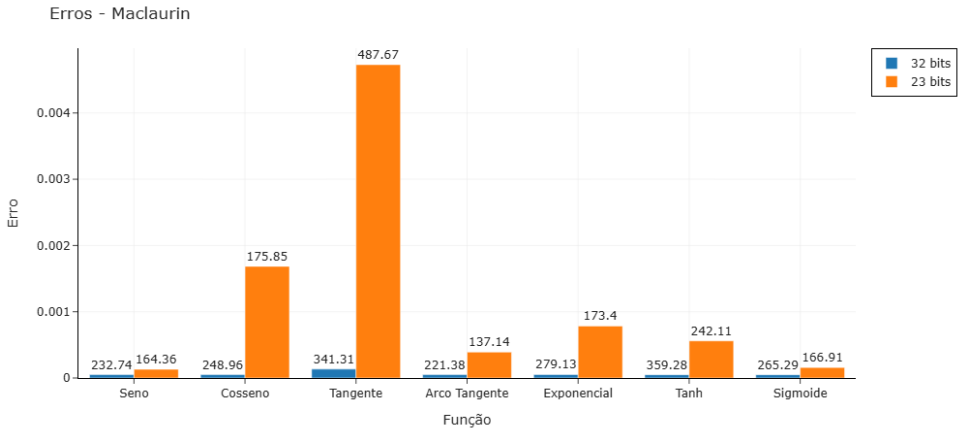

Figure 2: Erros — Séries de Maclaurin com 32 e 23 bits.

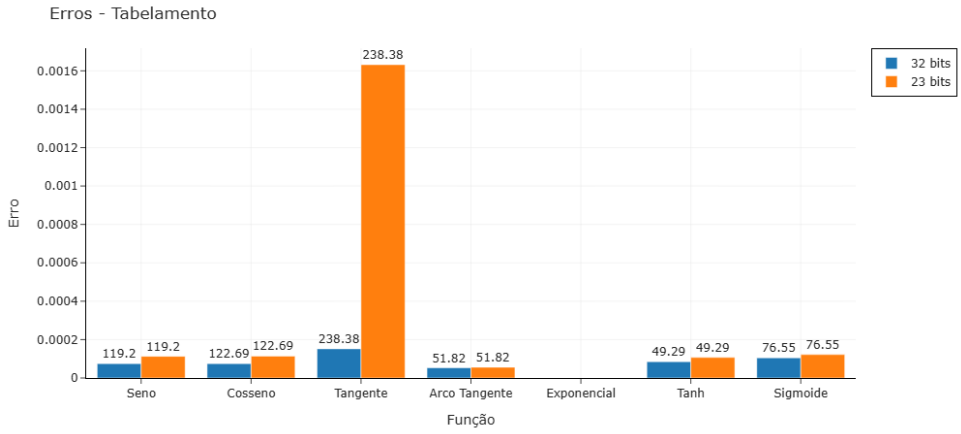

Figure 3: Erros — Tabelamento com 32 e 23 bits.

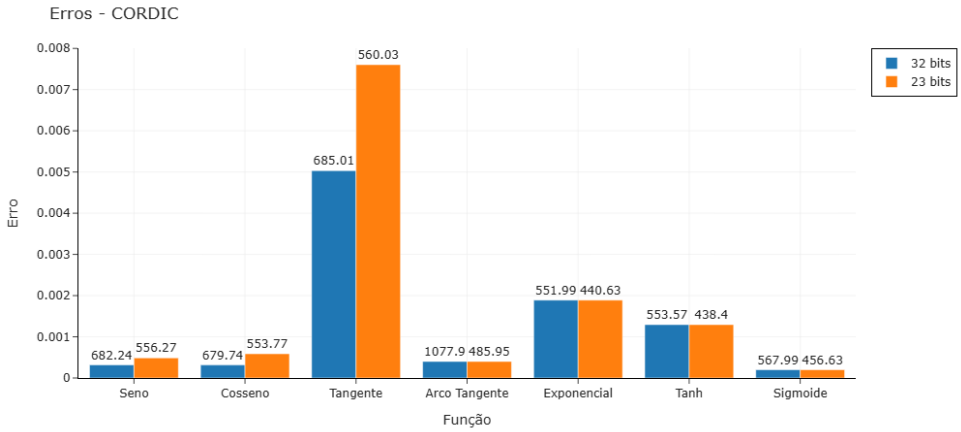

Figure 4: Erros — CORDIC com 32 e 23 bits.

7. Conclusão

A análise dos resultados indica que o algoritmo CORDIC não foi a alternativa mais satisfatória no ambiente experimental considerado: embora, teoricamente, o CORDIC necessite de aproximadamente 16 iterações para alcançar uma incerteza da ordem de $\pm 5 \times 10^{-5}$, devido à adição sucessiva dos termos $\pm 2^{-n}$ em cada iteração, na prática o erro numérico introduzido pela aritmética em ponto flutuante de 23 e 32 bits supera essa melhora teórica. Em particular, somas e subtrações entre parcelas muito pequenas tornam o sinal d_n sensível a erros de arredondamento, ocasionando escolhas de rotação incorretas e picos de erro observáveis nos resultados. Além disso, aumentar o número de iterações além de certos valores ótimos, aproximadamente $N = 14$ para 32 bits e $N = 11$ para 23 bits, não reduz o erro médio por conta do erro acumulado vindo da forma de representação numérica, enquanto eleva o número necessário de ciclos de clock da aproximação. Uma alternativa com potencial seria a implementação de CORDIC inteiro quantizado, que pode mitigar os efeitos de arredondamento de ponto flutuante e melhorar a precisão sem aumento proibitivo do número de ciclos, o que merece investigação em trabalhos futuros (13; 12).

Ao comparar tabelamento e séries de Maclaurin observa-se uma dualidade clara: as séries de Maclaurin, na configuração de 32 bits, proporcionaram, em média, a maior precisão, porém a um custo de maior número médio de ciclos de clock; o tabelamento, por sua vez, mostrou-se significativamente mais rápido e, com 23 bits na ULA, frequentemente alcançou melhor precisão e menor custo em ciclos do que a implementação por Maclaurin com 23 bits, evidenciando que não há uma alternativa que apresenta resultados objetivamente melhores para um caso geral do que os demais. Deve-se ainda notar que parâmetros experimentais como a tolerância de parada nas séries de Maclaurin e o número de entradas da tabela de valores, isto é, 48 valores armazenados na memória para cada aproximação, influenciam fortemente os resultados, de modo que ajustes dirigidos pelos requisitos da aplicação podem alterar o balanço entre precisão e desempenho.

Em suma, com os parâmetros usados neste trabalho, ou seja, larguras de ULA, número de entradas em tabelas e tolerância de parada, as séries de Maclaurin em 32 bits são a melhor opção quando a prioridade é precisão; o tabelamento é preferível quando a prioridade é latência e economia de lógica, especialmente em representações de 23 bits; e o CORDIC em ponto flutuante mostrou-se menos atrativo devido à sensibilidade ao arredondamento.

REFERENCES

[1] Agarwal, R.P, P. K. P. S. (2011). *An Introduction to Complex Analysis*. Springer.
[2] AGUIAR, R.G., N. V. M. F. H. H. (2020). Implmentação do algoritmo cordic para cálculo de seno e cosseno em fpga. *For Science*.
[3] Boyce, W.E., D. R. M. D. (2017). *Elementary differential equations and boundary value problems.* John Wiley & Sons, 2017.
[4] Briggs, I. Lad, Y. P. P. (2023). Implementation and synthesis of math library functions. *Cornell University*.
[5] Driscoll, T.A., B. R. (2018). *Fundamentals of numerical computation*. Society for Industrial and Applied Mathematics.
[6] Gelfand, I.M., S. M. (2001). *Trigonometry*. Birkhäuser Verlag.
[7] ITEM, M., L. J. G. Y. O. G. S. M. M. O. (2023). Transpimlib: A library for efficient transcendental functions on processing-in-memory systems. *ETH Zurich*.
[8] Kantabutra, V. (1996). On hardware for computing exponential and trigonometric functions. *IEEE Transactions on Computers*.

[KAPISCH] KAPISCH, E.B., A. M. F. L. S. L. Sapho - scalable-architecture processor for hardware optimization: an fpga customizable implementation approach.
[10] Kyurkchiev, N., M. S. (2015). Sigmoid functions some approximation and modelling aspects. *ResearchGate*.
[11] LI, P., J. H. X. W. X. C. Y. H. H. K. (2023). A reconfigurable hardware architecture for miscellaneous floating-point transcendental functions. *Advances of Electronics Research from Zhejiang University*.
[12] Moroz, L., S. V. (2018). The cordic method of calculating the exponential function. *Technical Transactions*.
[13] Parris, R. (2010). Elementary functions and calculators. *Cacocntery Iundtbhns enf Deaduaethrs*.
[14] PELLEGRINI, J. (2015). *Álgebra Linear*. Aleph0.
[15] PRADHAN, C., L. M. T. J. (2022). Efficient table-based function approximation on fpgas using interval splitting and bram instantiation. *Cornell University*.
[16] Press, W.H., T. S. V. W. F. B. (2002). *Numerical Recipes in C*. Press Syndicate of the University of Cambridge.
[17] SANTOS, V.A.M., K. E. S. L. F. L. (2019). Implementação de circuitos aritméticos em ponto flutuante, utilizando formato com número de bits configurável. *Sociedade Brasileira de Automática*.
[18] Sebah, P., G. X. (2002). Introduction on bernoulli's numbers. *http://numbers.computation.free.fr/*.
[19] Stein, E.M., S. R. (2003). *Signals and systems*. Princeton University Press.
[20] Stewart, J.D., C. D. W. S. (2021). *Cálculo: Volume 1*. Cengage Learning.
[21] Viccini, L. (2023). Otimizações para processador soft-core embarcado em fpga visando implementação de métodos de reconstrução online de energia em aceleradores de partículas. Master's thesis, UFJF.
[22] Zhou, Z.L, V. D. (2023). Cordic algorithm and its applications. *Università degli Studi di Padova*.

IMPLEMENTATION OF NONLINEAR FUNCTIONS IN AN FPGA-BASED SOFT-CORE PROCESSOR

Abstract. *This paper describes the implementation and comparison of different methods for approximating nonlinear functions on the SAPHO reduced-architecture soft-core processor. The study considers the sine, cosine, tangent, arctangent, exponential, hyperbolic tangent and sigmoid functions, and evaluates three approaches: Maclaurin series, lookup tables with interpolation, and the CORDIC algorithm. Implementations were performed for two data-width configurations (23 and 32 bits) and assessed in terms of mean error and number of clock cycles required for execution. Results indicate that Maclaurin series tend to provide higher accuracy—especially in the 32-bit configuration—at the expense of longer execution time. Lookup tables present a favorable trade-off between accuracy and performance, while CORDIC performed worse on the SAPHO due to its demand for higher precision and the propagation of errors in the floating-point computation circuitry. Conclusions are supported by graphs, tables and code excerpts that substantiate the analysis.*

Keywords: *FPGA, Numerical Approximations, Nonlinear Functions, Embedded Processor*