Desenvolvimento de um Algoritmo IP CORDIC Universal nas Arquiteturas Serial e Paralela

Igor Azevedo Cintra; Guilherme Pires Piedade; Luiz Ricardo Pires; Natanael Irland de Souza;
Rafael da Silva Macêdo; Marcelo Pelagio Pontes Morais; Cícero Luiz Alves Cunha; Letícia Carneiro de Souza

***Resumo*—Neste artigo, é apresentado um estudo teórico dos fundamentos do algoritmo CORDIC (COrdinate Rotation DIgital Computer) que foram necessários e suficientes para o desenvolvimento de um algoritmo CORDIC universal. Este algoritmo foi implementado nas arquiteturas serial e paralela. Os recursos utilizados foram a linguagem de descrição de hardware Verilog e o Python. A análise de simulação foi realizada em Python, e a implementação e verificação foram realizadas em Verilog. O algoritmo desenvolvido contribuirá fortemente em diversas aplicações futuras, tais como, sistemas de telecomunicações, robótica e energia que exigem requisitos de operação em tempo real e precisão.**

Palavras-Chave—CORDIC, pseudorrotações, Arquitetura serial, Arquitetura paralela

I. INTRODUÇÃO

As funções trigonométricas e hiperbólicas são amplamente utilizadas em computação científica, energia, robótica, biomédica, telecomunicações, processamento de imagens e redes neurais, entre outras [1]. Com o rápido desenvolvimento dessas áreas, requisitos de velocidade e precisão foram impostos para o cálculo dessas funções matemáticas, e, ao mesmo tempo, o consumo de recursos lógicos de hardware precisa ser controlado. No entanto, sua desvantagem é a velocidade de convergência: geralmente, uma rodada de iteração pode aumentar apenas um dígito efetivo, e mais ciclos de clock são necessários para que o resultado atinja a precisão exigida. O algoritmo básico do CORDIC (COrdinate Rotation DIgital Computer) é uma técnica iterativa eficiente para computar rotações vetoriais e funções matemáticas elementares (soma, multiplicação, divisão, raiz quadrada, funções trigonométricas, logaritmos e hiperbólicas) [2][3]. Os cálculos deste algoritmo podem ser realizados por uma sequência de somas e deslocamentos de bit. Portanto, o CORDIC é bastante útil em projetos de sistemas embarcados que precisam de resposta em tempo real e uma precisão satisfatória.

Com base nisso, neste artigo propõe-se realizar um estudo teórico do algoritmo CORDIC genérico e analisar alguns resultados obtidos durante o desenvolvimento de um algoritmo CORDIC universal, implementado nas arquiteturas serial e paralela. A organização deste artigo é descrita a seguir. Na seção II, são apresentados os fundamentos do algoritmo CORDIC, tipos de operação e modelamento das funções e suas limitações. Na seção III, é realizada uma breve descrição do desenvolvimento do algoritmo CORDIC universal, tais como, quantidade de iterações, tipo de dados de entrada e saída, ordem de grandeza do erro e equações generalizadas utilizadas. Na seção IV, são analisados alguns resultados da implementação do algoritmo em Verilog. Na seção V, são apresentadas algumas conclusões e sugerido um trabalho futuro.

II. FUNDAMENTOS DO ALGORITMO CORDIC

A base do algoritmo CORDIC é a teoria das transformações lineares. O algoritmo consiste na realização de sucessivas pseudorrotações trigonométricas circulares, hiperbólicas e lineares.

A. Pseudorrotações Trigonométricas

A Fig. 1 mostra o processo de rotação de um vetor v_0 de um ângulo θ a fim de obter-se o vetor final v_R. Da Fig. 1, obtêm-se:

$$\begin{pmatrix} x_R \\ y_R \end{pmatrix} = cos(\theta) \begin{pmatrix} 1 & -tg(\theta) \\ tg(\theta) & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{1}$$

e, usando a identidade trigonométrica

$$cos(\theta) = \frac{1}{\sqrt{1+tg^2(\theta)}} \tag{2}$$

em (1), resulta que

$$\begin{pmatrix} x_R \\ y_R \end{pmatrix} = \frac{1}{\sqrt{1+tg^2(\theta)}} \begin{pmatrix} 1 & -tg(\theta) \\ tg(\theta) & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{3}$$

Donde a matriz de rotação $\mathrm{ROT}(\theta)$ é definida por

$$\mathrm{ROT}(\theta) = \frac{1}{\sqrt{1+tg^2(\theta)}} \begin{pmatrix} 1 & -tg(\theta) \\ tg(\theta) & 1 \end{pmatrix} \tag{4}$$

Contudo, a rotação de um ângulo θ pode ser realizada aplicando-se sucessivas pseudorrotações parciais. Ou seja,

$$\theta = \sum_{i=0}^{\infty} \theta_i = \sum_{i=0}^{\infty} d_i \, tg^{-1}(2^{-i}) \tag{5}$$

Onde $d_i = 1$ se a pseudorrotação for à esquerda e $d_i = -1$ se for à direita. Consequentemente, a matriz de rotação é reescrita como,

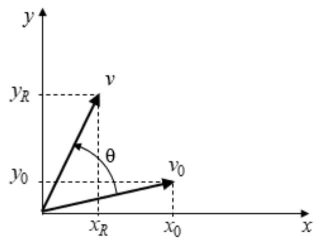


Fig. 1. Rotação vetorial por um ângulo θ

$$\mathrm{ROT}(\theta)=\prod_{i=0}^{\infty}\frac{1}{\sqrt{1+2^{-2i}}}\begin{pmatrix}1 & d_i 2^{-i}\\ d_i 2^{-i} & 1\end{pmatrix}=\prod_{i=0}^{\infty}\mathrm{ROT}(\theta_i) \qquad (6)$$

B. Rotação Maxima e Resíduo Máximo

O ângulo de rotação, dado por (5), possui um valor máximo θ_{max} para aplicação das pseudorrotações. Esse valor máximo pode ser estimado considerando-se apenas pseudorrotações à esquerda, $d_i = 1$. Assim, reescrevendo-se (5) na forma de dois somatórios tem-se:

$$\theta_{\max} = \sum_{i=0}^{4} tg^{-1}(2^{-i}) + \sum_{i=5}^{\infty} 2^{-i} \qquad (7)$$

E como o segundo somatório, em (7), é a soma de todos os elementos de uma progressão geométrica, o ângulo de rotação máximo será

$$\theta_{\max} = tg^{-1}(1) + tg^{-1}(0,5) + tg^{-1}(0,25) + tg^{-1}(0,125) +$$

$$+ tg^{-1}(0,0625) + 2^{-4} = 1,7433(rd) \qquad (8)$$

Ou, em graus, $\theta_{\max} = 99,88(°)$. Agora, no caso de (5) ser truncada em $i = (n-1)$,

$$\theta_{n-1} = \sum_{i=0}^{n-1} d_i\, tg^{-1}(2^{-i}) + \lim_{j\to\infty}\frac{2^{-n}(2^{-j}-1)}{(2^{-1}-1)} \qquad (9)$$

e o resíduo máximo R_{max} será

$$R_{\max} = 2^{-n+1} \qquad (10)$$

C. Modos de Operação e Modelamento do CORDIC

A Fig. 2(a) mostra o modo de operação por rotação e na Fig. 2(b) tem-se o modo de operação por vetorização. O CORDIC é modelado com três caminhos de dados (*x*, *y* e *z*). O caminho *x* é utilizado para denotar a componente real de um valor complexo, *y* para a componente imaginária, e *z* para o ângulo de rotação. Já no modo de rotação, Fig. 2(a) o objetivo é zerar a componente *z* após um número de rotações, garantindo que o vetor de entrada v_0 seja rotacionado em um dado ângulo θ, já no modo de vetorização, Fig. 2(b), a abordagem busca minimizar o valor de *y* para zero, resultando no vetor final $v_R = (x_R, 0)$.

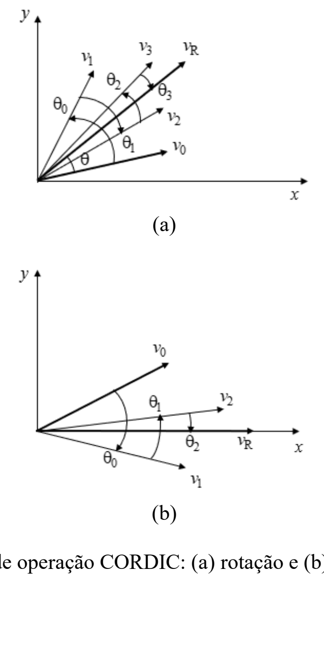


Fig. 2. Modos de operação CORDIC: (a) rotação e (b) vetorização

O modelamento é realizado conforme a trajetória executada durante a rotação. Na Fig. 3 tem-se as ilustrações dos três modos típicos (a) linear, (b) circular e (c) hiperbólico.

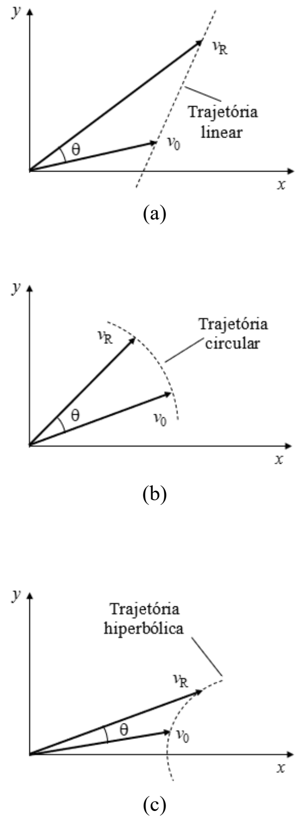


Fig. 3. Modos de modelamento: (a) linear, (b) circula e (c) hiperbólico

D. Fator de Escala

No caso de *modelamento circular* o *fator de escala* é definido como sendo o produtório

$$K = \prod_{i=0}^{n}\sqrt{1+2^{-2i}} \qquad (11)$$

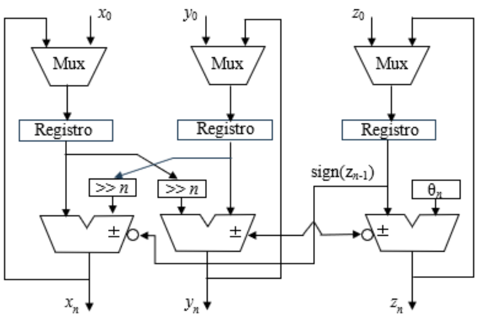
Onde a quantidade de iterações é $(n+1)$. Enquanto que no *modelamento hiperbólico*, esse *fator de escala* é definido pelo produtório

$$K' = \prod_{i=1}^{n}\sqrt{1-2^{-2i}} \qquad (12)$$

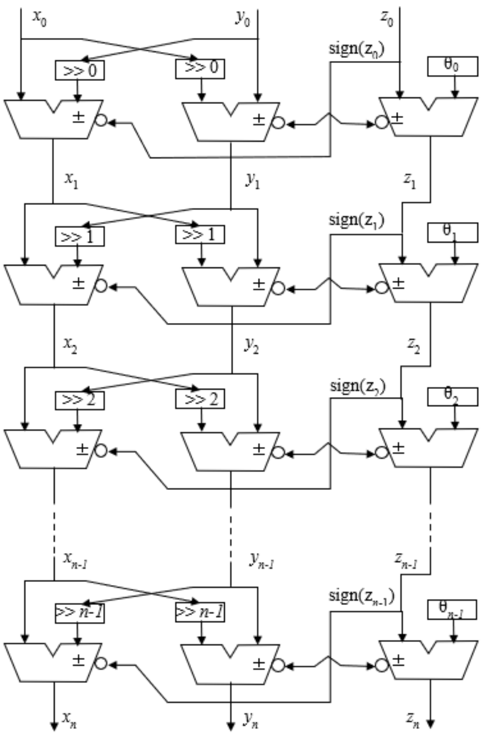
Onde *n* é a quantidade de iterações.

III. ALGORITMO CORDIC UNIVERSAL

O algoritmo CORDIC desenvolvido é universal pois possibilita operações nos modos rotação e vetorização, usando modelamento circular, linear e hiperbólico, implementados nas arquiteturas serial, Fig. 4(a), e paralela, Fig. 4(b). Portanto, ele oferece recursos para implementação de várias funções matemáticas. O algoritmo realiza uma sequência de pseudorrotações parciais utilizando apenas operações de soma e deslocamentos de bit, combinadas com operações de consulta, ou sejam, pesquisas em tabelas LUT (do inglês, Look-Up Table). Neste algoritmo foram realizadas as correções de *ganho* e vários testes a fim de validar o funcionamento correto do algoritmo. O algoritmo foi implementado em Verilog e seus resultados foram validados pelo software Python. No algoritmo foram definidos uma quantidade de iterações de 16 e o padrão de dados Q16.16 sinalizados para a entrada e saída. Também foi verificado e adotado um *erro* da ordem de 10E-3 [4].



(a)



(b)

Fig. 4. Arquiteturas implementadas: (a) série e (b) paralela

A. Equações Generalizadas

As equações básicas do CORDIC foram reformuladas de maneira generalizada e unificada, adequando-as para o cálculo das pseudorrotações nos sistemas de coordenada circular, hiperbólico e linear. São introduzidas três novas variáveis m, $\lambda(n)$ e $W_{\lambda(n)}$. As equações generalizadas são as seguintes:

$$x_{n+1} = x_n - m d_n y_n 2^{-\lambda(n)} \quad (13)$$

$$y_{n+1} = y_n + d_n x_n 2^{-\lambda(n)} \quad (14)$$

$$z_{n+1} = z_n - d_n W_{\lambda(n)} \quad (15)$$

Onde os valores das variáveis m e $\lambda(n)$ com os possíveis modos são descritos na Tabela I e as funções computáveis pelo CORDIC para cada modo de operação e para cada sistema de coordenadas, são resumidos na Tabela II.

TABELA I. VALORES POSSÍVEIS DAS VARIÁVEIS m e $\lambda(n)$

m	tipo	$\lambda(n)$
1	circular	n
-1	hiperbólico	$n-j$, j é o maior valor tal que, $3^{j+1}+2j-1\leq 2n$
0	linear	n

TABELA II. FUNÇÕES COMPUTÁVEIS PELO CORDIC

Tipo	m	W_n	Rotação $d_n = sign(z_n)$	Vetorização $d_n = sign(-y_n)$
C	1	$tg^{-1}(2^{-n})$	$x_n \to K(x_0 cos z_0 - y_0 sen z_0)$ $y_n \to K(x_0 sen z_0 + y_0 cos z_0)$ $z_n \to 0$	$x_n \to K\sqrt{x_0^2+y_0^2}$ $y_n \to 0$ $z_n \to z_0 + tg^{-1}(y_0/x_0)$
H	-1	$tgh^{-1}(2^{-n})$	$x_n \to K'(x_1 cosh z_1 + y_1 senh z_1)$ $y_n \to K'(x_1 senh z_1 + y_1 cosh z_1)$ $z_n \to 0$	$x_n \to K'\sqrt{x_1^2-y_1^2}$ $y_n \to 0$ $z_n \to z_1 + tgh^{-1}(y_1/x_1)$
L	0	2^{-n}	$x_n \to x_0$ $y_n \to y_0 + x_0 z_0$ $z_n \to 0$	$x_n \to x_0$ $y_n \to 0$ $z_n \to z_0 + (y_0/x_0)$

C – Circular; H – Hiperbólico; L - Linear

B. Operações Matemáticas Implementadas

A Tabela III mostra as operações realizadas e as entradas (x_{in}, y_{in} e z_{in}) e saídas (x_{out}, y_{out} e z_{out}) do algoritmo CORDIC universal. As operações matemáticas computadas são identificadas na tabela por: (a) seno e cosseno; (b) arco tangente e módulo; (c) multiplicação; (d) divisão; (e) seno e cosseno hiperbólicos; (f) arco tangente e módulo hiperbólicos.

TABELA III. OPERAÇÕES MATEMÁTICAS IMPLEMENTADAS

Ítem	x_{in}	y_{in}	z_{in}	x_{out}	y_{out} ou z_{out}
(a)	$1/K$	0	θ	$cos(\theta)$	$y_{out} \to sen(\theta)$
(b)	x_{in}	y_{in}	0	$K\sqrt{x_{in}^2+y_{in}^2}$	$z_{out} \to tg^{-1}(y_{in}/x_{in})$
(c)	x_{in}	0	θ	x_{in}	$y_{out} \to x_{in} \cdot z_{in}$
(d)	x_{in}	y_{in}	0	x_{in}	$z_{out} \to y_{in}/z_{in}$
(e)	$1/K'$	0	θ	$cosh(\theta)$	$y_{out} \to senh(\theta)$
(f)	x_{in}	y_{in}	0	$K'\sqrt{x_{in}^2-y_{in}^2}$	$z_{out} \to tgh^{-1}(y_{in}/x_{in})$

IV. Resultados e Discussões

As Fig. 4-7 mostram os resultados obtidos das operações de *multiplicação*, *seno*, *seno* e *cosseno hiperbólicos*. As simulações foram realizadas em Verilog. A operação de multiplicação, Fig. 5, mostra que o algoritmo CORDIC serial apresenta erros inferiores a 10E-2. Enquanto que nos cálculos operação seno, Fig. 6, o CORDIC paralelo apresentou erros menores, ou seja, inferiores a 10E-3.

A Fig. 7 e Fig. 8 mostram os resultados obtidos para o seno e cosseno hiperbólicos, para um ângulo de 40 graus. Nota-se que o CORDIC paralelo é 23 vezes mais rápido que o CORDIC serial.

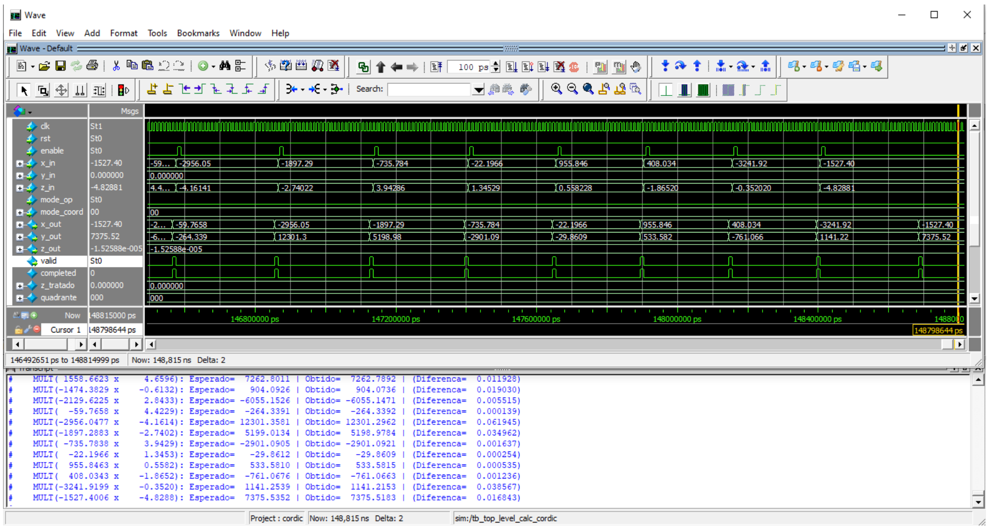


Fig. 5. Implementação em Verilog da operação de *multiplicação* realizada pelo algoritmo CORDIC na arquitetura serial

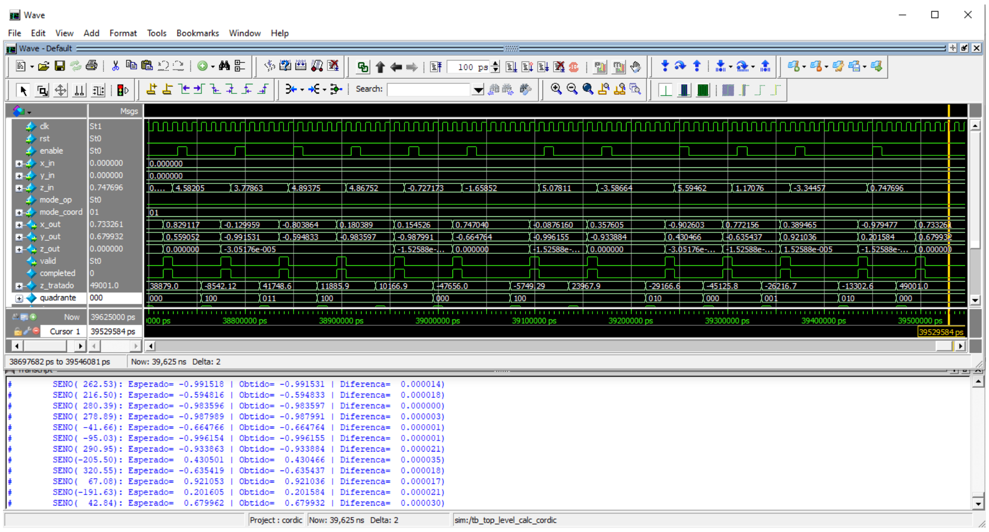


Fig. 6. Implementação em Verilog da operação *seno* realizada pelo algoritmo CORDIC na arquitetura paralela

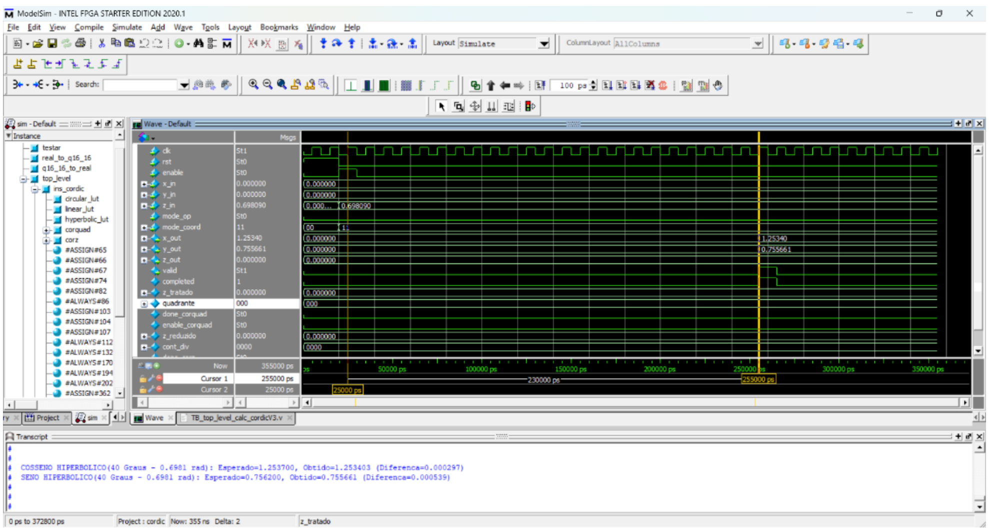


Fig. 7. Implementação em Verilog da operação de *senh*(40°) e *cosh*(40°) realizada pelo algoritmo CORDIC na arquitetura serial

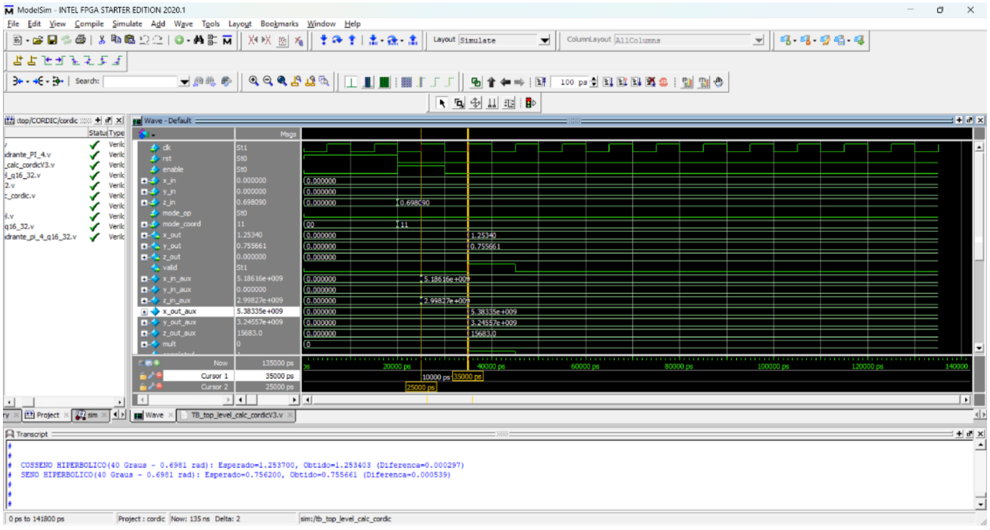


Fig. 8. Implementação em Verilog da operação de $senh(40°)$ e $cosh(40°)$ realizada pelo algoritmo CORDIC na arquitetura paralela

V. Conclusões

O algoritmo CORDIC universal possibilitou operações nos modos rotação e vetorização, usando modelamento circular, linear e hiperbólico. O CORDIC serial e o paralelo apresentaram praticamente mesmos resultados nos cálculos, porém, o serial apresenta uma latência bem maior. A boa precisão dos cálculos mostrou a eficiência do algoritmo. Com relação a trabalhos futuros, o CORDIC universal tem potencial para: "desenvolvimento de uma ALU para coprocessador trigonométrico".

Agradecimentos

Os autores são gratos à SOFTEX pelo suporte financeiro e a todos os servidores do INATEL que contribuíram de forma direta ou indireta para o desenvolvimento deste trabalho.

Referências

[1] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 ears of cordic: Algorithms, architectures, and applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, pp. 1893–1907, 2009.

[2] J.E. Volder, "The CORDIC Trigonometric Computing Technique." IRE Trans. Comput., Vol. EC-8, 1959, pp. 330-334.

[3] J.S. Walther, "A Unified Algorithm for Elementary Functions." AFIPS Spring Joing Comput. Conf., 1971, pp. 379-385.

[4] L. R. Pires, (2025) "CORDIC" (1.0) https://github.com/luizrpires/CORDIC