Desempenho do Algoritmo Paralelo CORDIC em Implementação em FPGA

Diego Barragán G., Karlo G. Lenzi, Luís G. P. Meloni

***Resumo*— O presente trabalho apresenta um projeto em hardware digital para cálculo de funções transcendentes (magnitude e fase de números complexos) por meio das coordenadas de um vetor usando o algoritmo CORDIC no modo vetorial. O código foi sintetizado para FPGA com ferramentas de desenvolvimento da Xilinx para a família Spartan-3E xc3s500e. É feita a análise de desempenho mudando-se os parâmetros e o número de iterações do algoritmo. Os resultados estão no estado da arte na prática corrente com 16 bits de largura de palavra com uso de 16 iterações. Tais resultados demostram a eficiência da arquitetura proposta para a implementação do método CORDIC.**

***Palavras-Chave*— CORDIC, VHDL, FPGA, Módulo, Fase.**

***Abstract*— This article presents a design in digital hardware for transcendental function computation (magnitude and phase of a complex number) using vector coordinates in the CORDIC algorithm in vectoring mode. The code was synthesized for FPGA using Xilinx development tools for the Spartan-3E xc3s500e family. Besides, performance analysis is realized changing algorithm parameters and the number of iterations. The results are at state of art for present practice with 16 word width requiring 16 interactions. These results validate the proposed architecture for the CORDIC algorithm.**

***Keywords*— CORDIC, digital IC design, FPGA, Module, Phase.**

I. Introdução

O desenvolvimento de novas técnicas que permitem que um mesmo hardware suporte diferentes padrões de comunicações, deu origem à técnica de Radio Definido por Software (SDR), a qual propõe um projeto de sistemas sem fio configurados e controlados por software sem mudar o hardware e com capacidade de operar em um amplo espectro de frequências possibilitando uma interoperabilidade entre sistemas de diferentes arquiteturas [8] [6].

As plataformas para desenvolver os sistemas de comunicação baseados em SDR possuem processadores de propósito geral, processadores digitais de sinais (DSP), arranjos de portas programáveis em campo (FPGA) ou circuitos integrados de aplicação específica (ASIC). Por motivo de desempenho, paralelismo e flexibilidade, a tecnologia de FPGA é uma das melhores alternativas de implementação para a infraestrutura de sistemas de comunicações. Além disso, a redução do tempo de desenvolvimento na prototipagem (se comparado ao projeto de ASICs) empregando-se ferramentas dos fornecedores (Xilinx, Altera) faz ainda mais atrativo o uso das FPGA.

Diego Barragán G., Luís G. P. Meloni¸Departamento de Comunicações DECOM, RT-DSP, FEEC - UNICAMP, Campinas, SP, Brasil, E-mail: {diego,meloni}@decom.fee.unicamp.br. Karlo G. Lenzi, Centro de Pesquisa e Desenvolvimento em Telecomunicações CPqD, Campinas, SP, Brasil, E-mail: klenzi@cpqd.com.br

Muitos algoritmos de comunicações digitais requerem da avaliação de funções elementares tais como operações trigonométricas. Geralmente, o método de look-up table (LUT) pode ser utilizado para o cômputo exato das funções trigonométricas, porém se requer uma grande memória da FPGA. Uma alternativa ótima para o cálculo de tais funções é o algoritmo CORDIC (*COordinate Rotation DIgital Computer*) [13].

Este artigo apresenta o projeto de uma arquitetura eficiente em FPGA para o algoritmo CORDIC para o cálculo do modulo e fase de um vetor em formato retangular. A implementação descrita possui diversas aplicações: ela pode ser empregada, por exemplo, em aplicações de comunicações digitais na detecção de fase ou para a sincronização de portadoras [12] [7] [5].

O projeto foi realizado usando uma descrição estrutural e sintetizado no circuito FPGA Spartan-3E xc3s500e usando o ambiente de Xilinx ISE 14.1.

O trabalho está organizado da seguinte forma: a seção 2 apresenta os conceitos matemáticos do cálculo da tangente inversa e do modulo de um vetor. A seção 3 apresenta os componentes do projeto e suas conexões no circuito. A seção 4 apresenta os resultados da simulação e finalmente a seção 5 conclui o trabalho.

II. Conceitos Matemáticos

O algoritmo CORDIC pode ser empregado em dois modos de operação [2], ilustrados na Figura 1:

Modo rotacional: a entrada é o vetor $\vec{r} = (x, y)$ e o ângulo θ a girar, e a saída é o vetor girado com novas coordenadas $\vec{r}' = (x', y')$. Desta forma, o vetor de entrada gira um ângulo especifico que é introduzido como parâmetro.

Modo vetorial: a entrada é o vetor $\vec{r} = (x, y)$ e a saída é a magnitude R e o ângulo θ. Neste caso, o algoritmo gira o vetor para alinhá-lo ao eixo X (acumulando o ângulo necessário para fazer essa rotação, cujo acumulador é iniciado em zero) até minimizar a magnitude do eixo Y a 0.

As equações que permitem implementar o algoritmo CORDIC são chamadas equações CORDIC, as quais são:

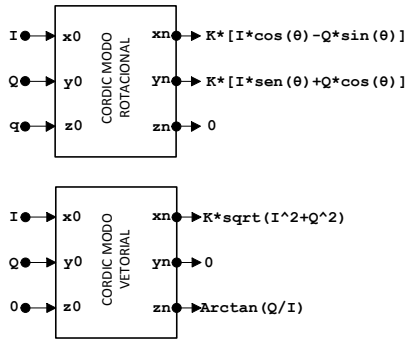

Fig. 1. CORDIC: modo rotacional e modo vetorial.

$$x_{i+1} = x_i - y_i\, d_i\, 2^{-i} \tag{1}$$

$$y_{i+1} = y_i + x_i\, d_i\, 2^{-i} \tag{2}$$

$$z_{i+1} = z_i - d_i \arctan\left(2^{-i}\right) \tag{3}$$

Usando as equações CORDIC, obtém-se rotações para ângulos menores que 90^o [4]. Para atingir rotações maiores que 90^o, deve-se realizar primeiro uma rotação de $\pm\, 90^o$ (para garantir que o vetor fique apenas no I ou IV quadrantes) com o emprego das seguintes equações:

$$x_{i+1} = -\, y_i\, d_i \tag{4}$$

$$y_{i+1} = x_i\, d_i \tag{5}$$

$$z_{i+1} = d_i\, .90^o \tag{6}$$

onde $d_i = -1$ para se girar de 90^o e $d_i = +1$ para se girar de -90^o.

Para calcular a tangente inversa e módulo dum vetor, emprega-se o algoritmo CORDIC circular em modo vetorial (o vetor da entrada é girado para ser alinhado com o eixo Y). Neste caso, as coordenadas x_i e y_i do vetor $\vec{r}$ fazem a iniciação do algoritmo, e ao final das iterações, obtém-se na variável z_{i+1}, o valor do ângulo θ do vetor $\vec{r}$, e na variável x_{i+1} obtém-se o valor da magnitude do vetor $\vec{r}$ multiplicada pelo fator A_n. As equações para esse modo são (com $z_0 = 0$ e n igual ao número de iterações):

$$x_n = A_n \sqrt{(x_0)^2 + (y_0)^2} \tag{7}$$

$$y_n = 0 \tag{8}$$

$$z_n = z_0 - \arctan\left(\frac{y_0}{x_0}\right) \tag{9}$$

$$A_n = \prod_{i=0}^{n-1} \sqrt{1 + 2^{-2i}} \tag{10}$$

Portanto, o algoritmo tem duas etapas [2]: iniciação juntamente com rotação por $\pm\, 90^o$ e pseudo-rotações iterativas que buscam levar a zero a variável y_i.

Para realizar a etapa de rotação de $\pm 90^o$, utiliza-se o procedimento na Figura 2.

1) Inicialização: $x_i = x$; $y_i = y$; valores do vetor $\vec{r} = (x, y)$, $i = 0$.
2) Se $y_i \geq 0$, então $d_i = -1$, caso contrario, $d_i = 1$.
3) Rotação por $\pm\, 90^o$:

$$x_{i+1} = -\, y_i\, d_i$$
$$y_{i+1} = x_i\, d_i$$
$$z_{i+1} = d_i\, .90^o$$

Fig. 2. Procedimento da etapa de inicialização e rotação para garantir que o vetor fique apenas no I ou IV quadrantes.

Para realizar a etapa de N pseudo-rotações iterativas, utiliza-se o procedimento na Figura 3.

1) Se $y_i \geq 0$, então $d_i = -1$, caso contrario, $d_i = 1$.
2) Pseudo-rotações:

$$x_{i+1} = x_i - y_i\, d_i\, 2^{-i}$$
$$y_{i+1} = y_i + x_i\, d_i\, 2^{-i}$$
$$z_{i+1} = z_i - d_i \arctan\left(2^{-i}\right)$$

3) Incrementar i.
4) Se i é menor a n vai para o passo 1.
5) Sair

Fig. 3. Procedimento para as etapas das pseudo-rotações.

III. Projeto e Implementação Dos Componentes Da FPGA

Os diversos módulos de implementação na FPGA são apresentados nesta seção.

A. Considerações do cálculo de magnitude e fase

Para implementar o algoritmo CORDIC em hardware (FPGA ou ASIC) temos dois tipos de representação numérica em ponto-fixo [14]:

Formato fracionário ponto-fixo em complemento a 2 para a magnitude: para representar os valores de x_i e y_i (faixa de -1 a 1) e da magnitude final de x_n (que é $A_n = 1,6467$, o ganho de processamento com n=16). Nesta representação tem-se 1 bit de sinal, 1 bit de magnitude e 14 bits da parte fracionária: $A(1, 14)$. Portanto, possui uma faixa de -2 a 1,999.

Formato fracionário ponto-fixo em complemento a 2 para a fase: para representar os valores da fase z_n (faixa de 0 a $-\pi$ e de 0 a π), tem-se 1 bit de sinal, 2 bits de magnitude e 13 bits da parte fracionária: $A(2, 13)$. Portanto, a faixa é de -4 a 3,999.

B. Projeto da unidade de rotação de ± 90 graus

Com a finalidade de aumentar a extensão de convergência do algoritmo CORDIC (de $-\pi/2$ a $\pi/2$ para $-\pi/2$ a $3\pi/2$) é necessária uma rotação do vetor para colocá-lo nos quadrantes I e IV. O diagrama funcional da unidade de rotação é apresentado na Figura 4. Neste caso, é preciso três somadores algébricos e um negador. Neste projeto, o somador da fase inicial de $+90^o$ é trocado pelos sinais de -90^o e $+90^o$ já armazenados, e a saída depende do bit mais significativo (bit de sinal) do componente y_i. Para o tratamento dos sinais x_i e y_i, o projeto possui uma unidade para calcular o complemento a2 e gerar os sinais de x_{i+1} e y_{i+1}.

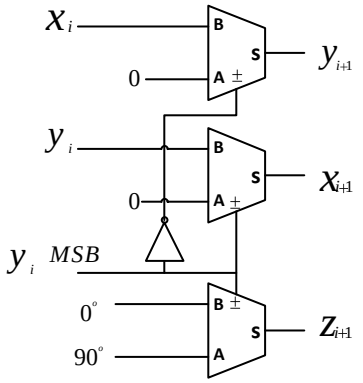

Fig. 4. Diagrama funcional da unidade de rotação de $\pm 90^o$.

C. Unidade de Soma

Com uma combinação de somadores completos (*full-adder*) é realizada a soma de dois números binários de uma largura determinada. Há várias arquiteturas para um somador: *ripple carry adder*, *ripple carry bypass adder* e *carry look-ahead adder* [9] [10]. Para o projeto, a escolha foi o somador *ripple carry*, devido a fato de que é o somador mais simples em arquitetura, porém tem o desempenho mais fraco em tempo de execução.

Acrescentando uma etapa de inversão a cada bit do vetor binário e estabelecendo o sinal de carry da entrada ao valor de 1, tem-se um somador algébrico. O carry é somado ao valor invertido do vetor, tendo, portanto, a representação dos vetores binários em complemento a2. O diagrama funcional da unidade de soma se apresenta na Figura 5.

D. Unidade de Deslocamento Conectada por Trilhas

O deslocamento de um vetor binário à direita produz a divisão por dois. Para cada iteração é preciso uma unidade de deslocamento distinta, portanto tem-se uma unidade de deslocamento interconectado por trilhas. A extensão do bit de sinal é realizada atribuindo o bit mais significativo aos bits mais significativos do resultado conforme o número de deslocamentos. A Figura 6 ilustra a unidade de deslocamento para o caso de um e três deslocamentos sob um vetor de 8 bits.

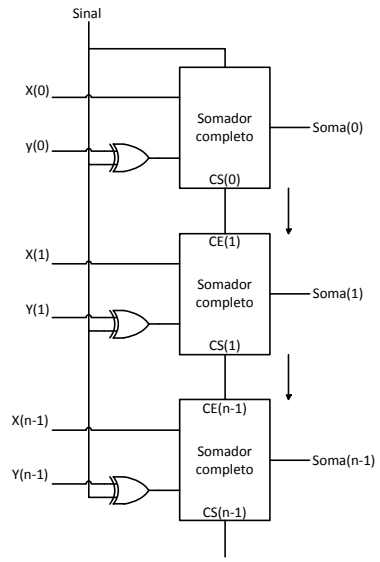

Fig. 5. Arquitetura interna de um somador composto de somadores completos.

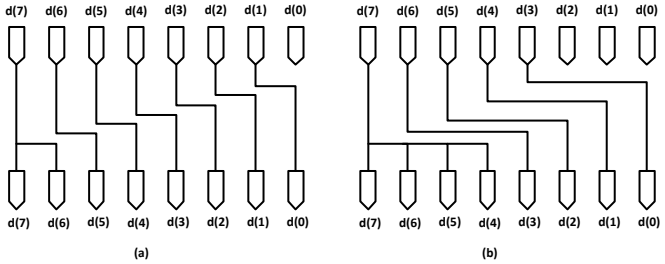

Fig. 6. Ilustração da unidade de deslocamento: (a) para 1 deslocamento, (b) para 3 deslocamentos sobre um operador de 8 bits.

E. Projeto da unidade de pseudo- rotação

A unidade de pseudo-rotação (também chamada de micro rotação) tem a finalidade de girar o vetor em cada uma das iterações. Esta unidade é composta dos seguintes blocos: duas unidades de deslocamento, três somadores algébricos e um negador. O diagrama funcional dessa unidade é apresentado na Figura 7. Para determinar a operação de soma ou subtração considera-se o bit mais significativo do sinal y (o bit de sinal). Os valores da Tabela I são os ângulos $\alpha_i = \arctan\left(2^{-i}\right)$ que são pré-armazenados na FPGA [1].

F. Arquitetura Paralela Desenvolvida

O algoritmo CORDIC desenvolvido separa em etapas cada iteração. Cada etapa é composta dos mesmos componentes: duas unidades de deslocamento, um negador e três somadores algébricos. Portanto, a saída de uma etapa corresponde à entrada da etapa seguinte.

O algoritmo CORDIC desenvolvido introduz duas vantagens importantes. As unidades de deslocamento e as constantes correspondentes a cada iteração (valores de

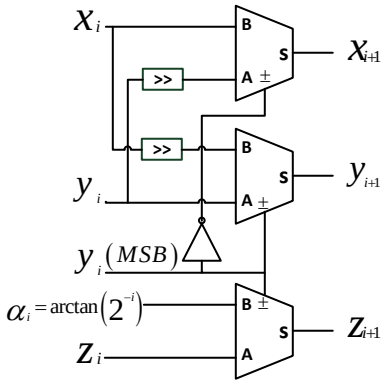

Fig. 7. Diagrama funcional da unidade de pseudo-rotação.

$\alpha_i = \arctan\left(2^{-i}\right)$) podem ser interconectadas por meio de trilhas da FPGA. A segunda vantagem é que o circuito é puramente combinatório e não precisa de registros, portanto o projeto torna-se simples. A desvantagem óbvia é a grande quantidade de área que precisa para sua implementação (o que implica em um maior consumo de potência [2]). O diagrama funcional da arquitetura paralela desenvolvida é apresentado na Figura 8.

TABELA I
VALORES DE $\alpha_i = \arctan\left(2^{-i}\right)$, E SUA REPRESENTAÇÃO EM FORMATO BINÁRIO (16 BITS).

Iteração	Valor decimal (rad)	Ponto fixo
0	0.785398163397448	1100100100001111
1	0.463647609000806	0111011010110001
2	0.244978663126864	0011111010110110
3	0.124354994546761	0001111111010101
4	0.062418809995957	0000111111111010
5	0.031239833430268	0000011111111111
6	0.015623728620477	0000001111111111
8	0.007812341060101	0000000111111111
9	0.003906230131967	0000000011111111
10	0.001953122516479	0000000001111111
11	0.000976562189559	0000000000111111
12	0.000488281211195	0000000000011111
13	0.000244140620149	0000000000001111
14	0.000122070311894	0000000000000111
15	0.000061035156174	0000000000000011

A maioria das vezes, em especial para FPGAs, uma arquitetura puramente combinacional não é vantajosa. O algoritmo desenvolvido é facilmente convertido em uma estrutura *pipelined* inserindo registros entre os somadores algébricos. No caso da maioria das arquiteturas da FPGA, elas já têm registros em cada célula lógica, portanto a adição de registros pipeline não tem custo adicional de hardware [3].

IV. RESULTADOS

Esta seção apresenta os resultados da implementação em FPGA e análise de desempenho.

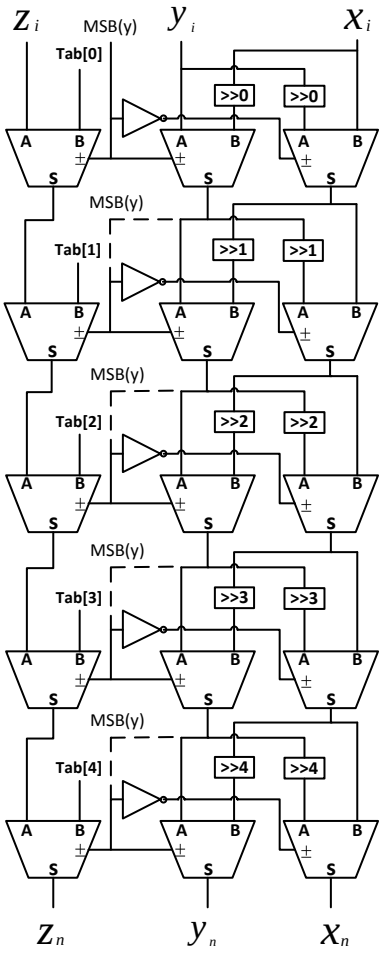

Fig. 8. Arquitetura bit-paralelo desenvolvida.

A. Erro Absoluto

Para a avaliação da arquitetura proposta, as entradas foram dez vetores com ângulos igualmente espaçados em 30 graus (colocando vetores nos quatro quadrantes). O código foi testado variando-se o valor das iterações desde 4 a 16. Os valores resultantes da simulação foram comparados com os valores obtidos em MATLAB [11]. Para o cálculo do erro absoluto empregou-se a seguinte equação:

$$Erro = |Cordic_{Matlab} - Cordic_{Vhdl}| \qquad (11)$$

O resultado do erro absoluto é apresentado na Figura 9.

B. Implementação em FPGA

O algoritmo CORDIC em VHDL foi sintetizado em uma FPGA Spartan-3E xc3s500e com as características da Tabela II.

O resultado do *testbench* para o caso de 16 iterações é apresentado na Figura 10.

O consumo de recursos na FPGA é apresentado na tabela III.

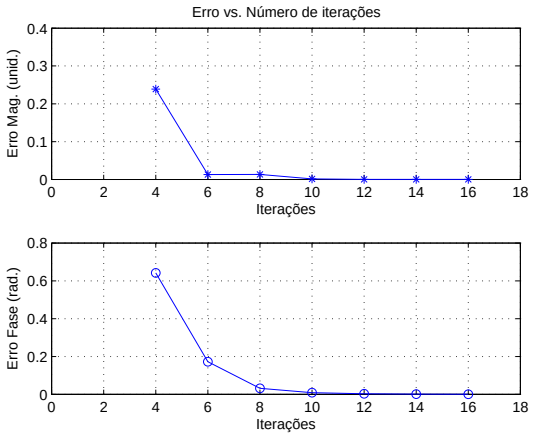

Fig. 9. Gráficos de erro absoluto.

TABELA II
CARACTERÍSTICAS DA FPGA EMPREGADA NO PROJETO.

Property Name	Value
Top-Level Source Type	HDL
Family	Spartan3E
Device	xc3s500e
Package	FG320
Speed	-4
Synthesis Tool	XST (VHDL/Verilog)
Simulator	Isim (VHDL/Verilog)
Preferred Language	VHDL
VHDL Source Analysis Standard	VHDL-93

V. CONCLUSÕES

Pelo gráfico apresentado na Figura 9, é fácil deduzir que há uma dependência direta entre o número de iterações com respeito à precisão do calculo da tangente inversa e do módulo de um vetor. Conforme se diminui o número de iterações, o erro cresce rapidamente divergindo dos valores corretos. Baseados nos resultados da tabela III, o tempo de processamento e o consumo de recursos da FPGA diminui proporcionalmente ao número de iterações. A escolha do número de iterações vem dependendo de onde vai ser empregado o CORDIC, por exemplo, em aplicações da arquitetura CORDIC para estimar o desvio de frequência em sistema OFDM, emprega-se um número maior o igual que 10 [7], mantendo um bom desempenho do circuito.

Também, é evidente que a área da FPGA será afetada pelo uso da arquitetura paralela desenvolvida, haja vista que está se fazendo uma cópia da iteração principal por um determinado número de iterações, as quais são cópias da iteração principal. Portanto, a arquitetura apresentada tem vantagens pelo o fato de ser simples de se implementar, já que não precisa de uma unidade de controle, como é o caso da arquitetura iterativa [10]. Por outro lado, a arquitetura paralela atinge velocidades de processamento maiores comparadas com a arquitetura iterativa, a qual requer um alto *fan-in*, o que reduz a velocidade máxima de operação do dispositivo [2].

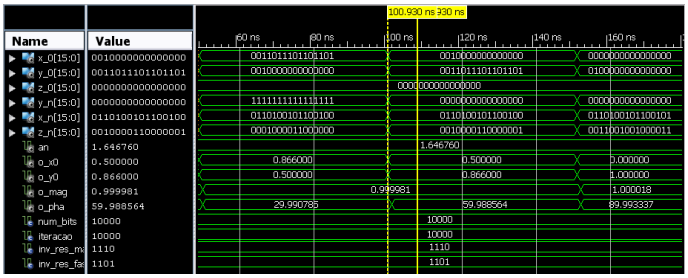

Fig. 10. Resultado do *test bench* com 16 iterações.

TABELA III
RESUMO DA SIMULAÇÃO EM CONSIDERAÇÃO À ÁREA E VELOCIDADE

Utilização	Iterações						
	16	14	12	10	8	6	4
Slices	901	683	512	356	214	122	49
4 input LUTs	1590	1207	901	630	376	214	87
IOBs	80	70	60	50	40	30	20
Delay (μs)	0.39	0.30	0.23	0.16	0.10	0.06	0.03

VI. AGRADECIMENTOS

A equipe do Laboratório RT-DSP agradece o suporte recebido da FINEP, através do Convênio "Rede H.264 SBTVD", contrato nº 01.08.0287.00. Este trabalho recebeu o apoio financeiro parcial da CAPES e CNPq.

REFERÊNCIAS

[1] Ferney Amaya and Jaime Velasco. Diseño de la tangente inversa usando el algoritmo cordic. *Taller XII Iberchip*, 1:183 – 186, 2006. 10.1023/A:1020202417934.

[2] Ray Andraka. A survey of cordic algorithms for fpga based computers. pages 191–200, 1998.

[3] R. Bhakthavatchalu, M.S. Sinith, P. Nair, and K. Jismi. A comparison of pipelined parallel and iterative cordic design on fpga. In *Industrial and Information Systems (ICIIS), 2010 International Conference on*, pages 239 –243, 29 2010-aug. 1 2010.

[4] X. Hu, R.G. Harber, and S.C. Bass. Expanding the range of convergence of the cordic algorithm. *Computers, IEEE Transactions on*, 40(1):13 – 21, jan 1991.

[5] Xu Li and Wang Qin. Cordic based algorithm for frequency offset estimation. In *Communication Technology (ICCT), 2010 12th IEEE International Conference on*, pages 817 –820, nov. 2010.

[6] A. Luiz Garcia Reis, A.F. Barros, K. Gusso Lenzi, L.G. Pedroso Meloni, and S.E. Barbin. Introduction to the software-defined radio approach. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 10(1):1156 –1161, jan. 2012.

[7] J. Mar, Chi-Cheng Kuo, and Shih-Hao Chou. Sdr structure based cfo estimation and compensation circuit for ofdm systems using reconfigurable cordic fpga modules. In *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*, pages 352 –355, dec. 2010.

[8] Joseph Mitola. *Systems-Level Architecture Analysis*, pages 112–170. John Wiley & Sons, Inc., 2002.

[9] Behrooz Parhami. *Computer arithmetic: algorithms and hardware designs*. Oxford University Press, Oxford, UK, 2000.

[10] Robert Joachim Schweers. Descripción en vhdl de arquitecturas para implementar el algoritmo cordic. http://biblioteca.universia.net/, Julho 2009.

[11] Sencha. Simple phase estimator. http://luminouslogic.com, Julho 2012.

[12] J. Valls, T. Sansaloni, A. Perez-Pascual, V. Torres, and V. Almenar. The use of cordic in software defined radios: a tutorial. *Communications Magazine, IEEE*, 44(9):46 –50, sept. 2006.

[13] Jack E. Volder. The cordic trigonometric computing technique. *Electronic Computers, IRE Transactions on*, EC-8(3):330 –334, sept. 1959.

[14] Randy Yates. Fixed-point arithmetic: An introduction. http://www.digitalsignallabs.com/fp.pdf, Julho 2009.