PAULO ROBERTO BUENO DE CARVALHO

Projeto de circuito Oscilador Controlado Numericamente implementado em CMOS com otimização de área

São Paulo

2016

PAULO ROBERTO BUENO DE CARVALHO

Projeto de circuito Oscilador Controlado Numericamente implementado em CMOS com otimização de área

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências no Programa de Engenharia Elétrica

São Paulo

2016

PAULO ROBERTO BUENO DE CARVALHO

Projeto de circuito Oscilador Controlado Numericamente implementado em CMOS com otimização de área

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências no Programa de Engenharia Elétrica

Área de Concentração: Microeletrônica

Orientador: Prof. Dr. Wilhelmus A. M. Van Noije

São Paulo

2016

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Carvalho, Paulo Roberto Bueno de
Projeto de circuito Oscilador Controlado Numericamente implementado em CMOS com otimização de área / P. R. B. Carvalho -- versão corr. -- São Paulo, 2016.
97 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Energia e Automação Elétricas.

1.ENGENHARIA ELÉTRICA 2.CIRCUITOS DIGITAIS 3.CIRCUITOS INTEGRADOS VLSI 4.ESPECTROSCOPIA 5.ARITMÉTICA I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Energia e Automação Elétricas II.t.

À minha família, que me mostrou o valor do trabalho árduo, da perseverança, me deu apoio e sempre me motivou durante minha jornada.

Aos meus amigos da área da Engenharia Elétrica e de outras áreas do conhecimento, que me ajudaram em diferentes atividades, estiveram presentes nos poucos momentos de lazer e me incentivaram, e que também servem de inspiração para o meu trabalho.

À minha noiva, por acreditar em minha capacidade, por seu apoio, paciência, amor e compreensão.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer ao meu orientador, Prof. Dr. Wilhelmus Van Noije, pela oportunidade, apoio, paciência, amizade, pelo incentivo e por ter me motivado durante o desenvolvimento do projeto e desta dissertação de mestrado.

Aos integrantes do grupo de pesquisa da DMPSV (Divisão de Metodologias de Projeto de Sistemas VLSI) do Laboratório de Sistemas Integráveis, pelo fornecimento da infraestrutura e do suporte necessários à base de dados utilizada para o projeto do circuito integrado. Em especial, ao José Amaya, pelo auxílio na implementação do sistema e pela submissão do trabalho no Programa "*2015 Free IC Fabrication mini@sic*" do IMEC; ao Hugo Daniel Hernandez, pela integração do layout do circuito implementado no topo do chip fabricado; e ao Lucas Severo, pela ajuda na confecção da placa de circuito impresso e na execução dos testes do chip fabricado.

Ao IMEC pela oportunidade dada através do programa "*2015 Free IC Fabrication mini@sic*", oferecido as universidades brasileiras, a partir do qual foi possível solicitar a fabricação do chip; em especial, ao Prof. Jacobus Swart, incentivador do programa no Brasil, pela parceria, junto à equipe do IMEC, no processo de encapsulamento dos chips utilizados para os testes.

Aos meus superiores da Design House do LSI-TEC (Laboratório de Sistemas Integráveis Tecnológico), por viabilizarem e incentivarem a realização dos meus estudos; aos meus pares de trabalho – e também amigos – do departamento, pelo companheirismo, pelos conselhos e por terem me proporcionado experiências na área de microeletrônica que embasaram minha pesquisa.

“If I have seen further it is by standing on

the shoulders of Giants.”

Sir Isaac Newton

RESUMO

Este trabalho consiste no projeto e implementação em CMOS de um circuito integrado digital para geração de sinais, denominado Oscilador Controlado Numericamente. O circuito será aplicado em um sistema de Espectroscopia por Bioimpedância Elétrica, utilizado como método para detecção precoce de câncer do colo do útero. Durante o trabalho, realizou-se o estudo dos requisitos do sistema de espectroscopia e as especificações dos tipos de sinais a serem gerados. Levantou-se, na bibliografia, algumas técnicas de codificação em linguagem de hardware para otimização do projeto nos quesitos área, potência dissipada e frequência máxima de funcionamento. Para implementar o circuito, também se pesquisou o fluxo de projeto de circuitos digitais, focando as etapas de codificação em linguagem de descrição de hardware *Verilog* e os resultados de síntese lógica e de *layout*. Foram avaliadas duas arquiteturas, empregando-se algumas das técnicas de codificação levantadas durante o estudo bibliográfico. Estas arquiteturas foram implementadas, verificadas em plataforma programável, sintetizadas e mapeadas em portas lógicas no processo TSMC 180 nm, onde foram comparados os resultados de área e dissipação de potência. Observou-se, nos resultados de síntese lógica, redução de área de 78% e redução de 83% na dissipação de potência total no circuito em que se aplicou uma das técnicas de otimização em comparação com o circuito implementado sem otimização, utilizando uma arquitetura CORDIC do tipo *unrolled*. A arquitetura com menor área utilizada - 0,017 mm^2 - foi escolhida para fabricação em processo mapeado. Após fabricação e encapsulamento do circuito, o chip foi montado em uma placa de testes desenvolvida para avaliar os resultados qualitativos. Os resultados dos testes foram analisados e comparados aos obtidos em simulação, comprovando-se o funcionamento do circuito. Observou-se uma variação máxima de 0,00623% entre o valor da frequência do sinal de saída obtido nas simulações e o do circuito fabricado.

Palavras-chave: Projetos de Circuitos Integrados Digitais. Espectroscopia por Bioimpedância. Oscilador Controlado Numericamente. Linguagem de Descrição de Hardware. RTL. CORDIC. Técnicas de codificação. Otimização. CMOS.

ABSTRACT

The aim of this work is the design of a digital integrated circuit for signal generation called Numerically Controlled Oscillator, designed in 180 nm CMOS technology. The application target is for Electrical Bioimpedance Spectroscopy system, and can be used as a method for early detection of cervical cancer. Throughout the work, the spectroscopy system requirements and specifications of the types of signals to be generated were studied. Furthermore, the research of some coding techniques in hardware language for design optimization in terms of area, power consumption and frequency operation was conducted looking into the bibliography. The digital design flow was studied focusing on the Verilog hardware description language and the results of logic synthesis and layout, in order to implement the circuit. Reviews of two architectures have been made, using some of the encoding techniques that have been raised during the bibliographical study. These architectures have been implemented, verified on programmable platform, synthesized and mapped to standard cells in TSMC 180 nm process, which compared the area and total power consumption of results. Based on the results of logic synthesis, a 78% area reduction and 83% power consumption reduction were obtained on the implemented circuit with encoding techniques for optimization in comparison with the another circuit using a CORDIC unrolled architecture. The architecture with smaller area – 0.017 mm^2 - was chosen for implementation in the mapped process. After the circuit fabrication and packaging, the chip was mounted on an evaluation board designed to evaluate the functionality. The test results were analyzed and compared with the simulation results, showing that the circuit works as expected. The output signals were compared between theoretical and experimental results, showing a maximum deviation of 0.00623%.

Keywords: Digital Integrated Circuits Design. Electrical Bioimpedance Spectroscopy. Numerically Controlled Oscillator. Hardware Language Description. RTL. CORDIC. Encoding Techniques. Optimization. CMOS.

LISTA DE FIGURAS

Figura 1 - Diagrama de blocos do sistema EBE 23

Figura 2 - Sistema DDS com entrada de controle de sintonia de sinal 24

Figura 3 - Diagrama de blocos do NCO utilizando LUT 25

Figura 4 - Comportamento do CORDIC em execução 27

Figura 5 - Arquitetura do circuito CORDIC 31

Figura 6 - Arquitetura CORDIC Bit-Serial (KUMAR e SAPPAL, 2011) 32

Figura 7 - Fluxo de Projeto de CIs 34

Figura 8 - Metodologia Top-down (PALNITKAR, 2003) 39

Figura 9 - Metodologia Bottom-up (PALNITKAR, 2003) 39

Figura 10 - Estimativas de área levantadas por (ZHANG, 2001) 44

Figura 11 - Descrição de potência estática de uma célula (WEST e HARRIS, 2005) 46

Figura 12 - Representação dos parâmetros da potência de chaveamento 47

Figura 13 - Técnica de inversão de barramento - número de transições (ARORA, 2012) (STAN e BURLESON, 1995) 49

Figura 14 - Representação Numérica 49

Figura 15 - Controle de dados da lógica – habilitação do caminho de dados 50

Figura 16 - Controle de habilitação de clock 50

Figura 17 - Sinais com várias transições misturados a lógica normal 52

Figura 18 - Sinal com muitas transições separado do resto da lógica 52

Figura 19 - Codificação binária versus codificação Grey (ARORA, 2012) 53

Figura 20 - Codificação binária da linha de seleção do MUX (ARORA, 2012) 54

Figura 21 - Codificação "one-hot" da linha de seleção do MUX (ARORA, 2012) 54

Figura 22 - Transições redundantes que consomem potência 55

Figura 23 - Transições redundantes contidas para redução de potência 56

Figura 24 - Transições redundantes durante barramento ponto para multi-ponto 56

Figura 25 - Transações redundantes contidas em barramento ponto para multi-ponto 57
Figura 26 - Memória circular 58
Figura 27 - Exemplo de extração de um MUX desnecessário 59
Figura 28 - Exemplos de equivalências utilizando-se primitivas lógicas para a codificação 60
Figura 29 - Logica sem compartilhamento de recursos 61
Figura 30 - Lógica com compartilhamento de recursos 61
Figura 31 - Diagrama de blocos do circuito NCO 65
Figura 32 - Diagrama de blocos do circuito gerador de ângulos 66
Figura 33 - Representação da fase do ângulo 68
Figura 34 - Diagrama de tempo do CORDIC 71
Figura 35 - Arquitetura da Unidade Aritmética do CORDIC 71
Figura 36 - Layout do NCO implementado 75
Figura 37 - Imagem do GDS (esquerda) e foto do chip fabricado (direita) 76
Figura 38 - Foto da placa de avaliação para testes do chip fabricado 77
Figura 39 - Análise comparativa entre os valores de frequência calculados e medidos 83
Figura 40 - Medida de saída do NCO de testes para frequência de saída 76 Hz. 84
Figura 41 - Medida da saída do NCO de testes para frequência de 156,11 kHz 84
Figura 42 - Medida da saída do NCO de testes para frequência de 1 MHz 85
Figura 43 - Medida SFDR da saída do circuito 85

LISTA DE TABELAS

Tabela 1 - Interfaces do módulo CORDIC 65

Tabela 2 - Representação binária dos quadrantes 69

Tabela 3 - Valores de $\tan^{-1}$ calculados e convertidos para implementação da tabela de arco tangente em RTL 72

Tabela 4 - Resultados de área e dissipação de potência totais obtidos entre duas arquiteturas do NCO extraídos em nível de portas lógicas (pós síntese física) 78

Tabela 5 – Comparação de resultados de dissipação de potência real versus extraído em nível de portas lógicas 79

Tabela 6 - Análise comparativa de recursos lógicos obtida na implementação em FPGA das duas arquiteturas 80

Tabela 7 - Comparação dos resultados de área e consumo do circuito fabricado com trabalhos de referência 80

Tabela 8 – Frequência do sinal de saída do NCO 82

LISTA DE ABREVIATURAS E SIGLAS

ASIC	*Application Specific Integrated Circuits*
CCU	Câncer do Colo do Útero
CI	Circuito Integrado
CMOS	*Complementary Metal Oxide Semiconductor*
CORDIC	*COordinate Rotation Digital Computer*
CT	Centros de Treinamento
DDS	*Direct Digital Synthesizer*
DAQ	*Data Acquisition*
DRC	*Design Rule Check*
EBE	Espectroscopia de Bioimpedância Elétrica
EDA	*Electronic Design Automation*
EIS	*Electrical Impedance Spectroscopy*
ERC	*Electrical Rule Check*
FPGA	*Field-Programmable Gate Array*
FSM	*Finite State Machine*
GDSII	Formato de base de dados utilizado pela indústria de CIs.
HDL	*Hardware Description Language*
IEEE	Institute of Electrical and Electronic Engineers
INCA	Instituto Nacional do Câncer
LC	*Logic Cells*
LSB	*Least Significant Bit*
LUT	*Look Up Table*
LVS	*Layout Versus Schematic*
MCTI	Ministério da Ciência, Tecnologia e Inovação
MPW	*Multi Project Wafer*

MUX	Multiplexador
NCO	*Numerically Controlled Oscillator*
PLI	*Programming Language Interface*
PLL	*Phase-Locked Loop*
QoR	*Quality of Results*
QoS	*Quality of Source*
ROM	*Read Only Memory*
RTL	*Register Transfer Level*
SFDR	*Spurious Free Dynamic Range*
TSMC	Taiwan Semiconductor Manufacturing Company Limited
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*
VLSI	*Very Large Scale Integration*

SUMÁRIO

1. INTRODUÇÃO 17
1.1. Considerações Gerais 17
1.2. Justificativa 19
1.3. Objetivos 19
1.4. Organização do Documento 20
2. REVISÃO SOBRE CIRCUITOS NCO E TÉCNICAS DE CODIFICAÇÃO EM NÍVEL RTL 22
2.1. Sistema de Espectroscopia por Bioimpedância Elétrica – EBE 22
2.2. Arquiteturas de gerador de sinais - Oscilador Controlado Numericamente 24
2.2.1 Look Up Table (LUT) 25
2.2.2 O Algoritmo CORDIC 26
2.3. Fluxo de projetos para circuitos digitais 33
2.4. Linguagem de Descrição de Hardware 37
2.5. Linguagem Verilog 38
2.6. Metodologia de projeto 39
2.7. Síntese Lógica 41
2.8. Simulação 42
2.9. Técnicas de otimização 42
2.9.1 Técnicas de codificação para redução de potência dissipada 45
2.9.2 Técnicas de codificação para otimização de área 59
2.9.3 Considerações a respeito das técnicas de codificação 62
3. IMPLEMENTAÇÃO DO CIRCUITO 64
3.1. Arquitetura implementada do NCO 64
3.1.1 Gerador de ângulos - Angle Generator 65

3.1.2 Pré-rotação (Pre-rotator) 68
3.1.3 Módulo CORDIC (CORDIC Arithmetic Unit) 70
3.1.4 Tabela de Arco tangente - Arctangent Table 72
3.1.5 Iteration Controller 72
3.1.6 Multiplexador de saída 72
3.2. Implementação do circuito NCO 74
3.3. Resultados Experimentais 77
4. CONCLUSÕES 87
4.1. Considerações Futuras 88
REFERÊNCIAS BIBLIOGRÁFICAS 90
APÊNDICE A – Código Verilog do Oscilador Controlado Numericamente e submódulos 95

1. INTRODUÇÃO

1.1. Considerações Gerais

O câncer no colo do útero - CCU - é o segundo tipo de neoplasia prevalecente nas mulheres, depois do câncer de mama. Com um alto número de casos e mortes por ano no mundo - de acordo com (DAS, DAS e CHATTERJEE, 2015), (DURAISAMY, JAGANATHAN e BOSE, 2011) e (WHO., 2007.), 500 mil casos e mais de 280 mil mortes - representa um grave problema na saúde pública. No Brasil, o Instituto Nacional do Câncer (INCA), órgão do Ministério da Saúde responsável pelo combate ao câncer, estima que, em 2016, devam ocorrer 16 mil novos casos da doença enquanto, em 2013, ocorreram 5.430 mortes em decorrência desse tipo de Câncer (INCA, 2016).

Segundo (BROWN, MILNES, *et al.*, 2005), (DURAISAMY, JAGANATHAN e BOSE, 2011), nos países em via de desenvolvimento (os quais apresentam 80% dos casos em nível mundial), programas eficientes de rastreamento são difíceis de implementar, principalmente devido a razões logísticas e econômicas. Portanto, faz-se necessário desenvolver técnicas alternativas para a detecção precoce do câncer no colo do útero.

Dentre as técnicas para detecção do CCU, destaca-se o teste de Papanicolau. De baixo custo, mas de resultado demorado (MIRANDA, BARRERO e ECHEVERRI, 2007), apresenta uma porcentagem alta de resultados falsos negativos, da ordem de 30 a 40%, uma vez que a sensibilidade de detecção desta técnica é menor que 60%.

Diante disso, outros autores como (DURAISAMY, JAGANATHAN e BOSE, 2011), (MIRANDA, 2005), (BROWN, MILNES, *et al.*, 2005), (MIRANDA, BARRERO e ECHEVERRI, 2007) e (KITCHENER, CASTLE e COX, 2006), reportam diferentes estudos para fazer a detecção antecipada de câncer no colo do útero; estudos esses relacionados, principalmente, aos conceitos da Espectroscopia por Bioimpedância Elétrica (EBE) e às propriedades óticas dos tecidos.

A EBE tem sido utilizada como método diagnóstico de diferentes doenças (ABDUL, BROWN, *et al.*, 2006), (BROWN, MILNES, *et al.*, 2005) e (BARROW e WU, 2007).

Destaca-se, de acordo com os resultados obtidos, o alto potencial da EBE para ajudar no rastreamento de doenças, incluindo o CCU.

As medições da bioimpedância elétrica para detecção do câncer são relativamente de baixo custo, visto que requerem, unicamente, um dispositivo para geração de corrente e medição de tensão. A necessidade de uma técnica não invasiva e de baixo custo permite que a medição da bioimpedância elétrica seja uma opção atrativa como dispositivo de diagnóstico.

A EBE baseia-se na diferença do comportamento elétrico dos tecidos normais em relação àqueles que têm algum nível de neoplasia. Para sua aplicação, estabelece-se uma configuração de quatro eletrodos (dois para injeção de corrente e os restantes para medição de tensão). Dessa maneira, aplica-se um sinal de corrente ao tecido sob teste com um valor de frequência variável (desde 100 Hz até 1 MHz), ao mesmo tempo em que se mede a tensão. Com os valores adquiridos de tensão e corrente, calcula-se o valor da impedância elétrica e analisa-se o gráfico em relação à frequência (Diagrama de Bode). Da análise do espectro da impedância elétrica pode-se estabelecer o nível de neoplasia que apresenta o tecido.

Para que este dispositivo seja de baixo custo, faz-se necessário o projeto de circuitos integrados de baixo consumo, fabricados em tecnologias cada vez menores e com maiores velocidades de processamento de informações de dados, a fim de manter desempenho. Procura-se, também, ao mesmo tempo, diminuir consumo de potência e reduzir área, aumentando a autonomia e a portabilidade destes dispositivos.

Diversas técnicas de otimização de projetos de circuitos integrados são empregadas, tanto para circuitos analógicos como para circuitos digitais. Para o projeto de circuitos digitais, especificamente, muitas destas técnicas e recursos são aplicados por meios que automatizam esse processo, as chamadas ferramentas de Automatização de Projeto Eletrônico (*Electronic Design Automation* (EDA)). No entanto, estas ferramentas não são suficientes para que o circuito digital atinja o consumo de potência esperado ou, muitas vezes, acabam otimizando a dissipação de potência, mas exigindo o aumento de área, utilizando células digitais maiores ou circuitos mais complexos.

Através dos Centros de Treinamento (CT) mantidos pelo Programa CI Brasil, do Ministério da Ciência, Tecnologia e Inovação (MCTI), houve um aumento na formação de projetistas de circuitos integrados no cenário brasileiro de microeletrônica. Muitos destes profissionais de circuitos digitais são oriundos de áreas relacionadas à Engenharia da Computação, e possuem uma base de programação que utiliza linguagens de alto nível de abstração. Ao efetuarem o projeto por meio de linguagens de descrição de hardware, como Verilog ou VHDL, estes projetistas acabam implementando algoritmos de alto nível de abstração. Tais descrições, do ponto de vista sistêmico, são eficientes, mas quando sintetizados pelas ferramentas de EDA, geram circuitos digitais muitas vezes ineficientes, que ocupam muita área e consomem muita potência. Isso ocorre devido ao distanciamento entre a forma como o algoritmo foi descrito e o resultado da interpretação do sistema implementado pela ferramenta de síntese, tornando-se muitas vezes uma estrutura de código não sintetizável, de forma que, não é possível gerar um hardware que represente o algoritmo descrito.

1.2. Justificativa

Este trabalho está justificado na necessidade de identificar as principais técnicas e boas práticas de codificação de um circuito digital, que podem ser aplicadas no início do fluxo de projeto e na etapa de descrição de hardware, por meio de linguagem própria para este propósito, buscando a otimização de circuitos através da diminuição da área e da redução da dissipação de potência. A partir deste levantamento, deseja-se desenvolver um circuito digital importante dentro do sistema EBE para geração de sinais – o Oscilador Controlado Numericamente – *Numerically Controlled Oscillator* (NCO), aplicando-se as melhores técnicas e práticas identificadas de forma a contribuir com a otimização do sistema de EBE, para que o circuito se torne, como um todo, de baixo custo.

1.3. Objetivos

- Estudar as técnicas de codificação em linguagem de descrição de hardware Verilog, que visam à otimização do processo de síntese lógica, medindo o

impacto que estas técnicas causam no resultado da síntese, tanto para o consumo de potência como para utilização de área;

- Desenvolver o projeto do circuito NCO: especificação do circuito, implementação em linguagem Verilog utilizando-se das técnicas pesquisadas e validação do circuito através de simulações funcionais;
- Análise de síntese e comparação entre o bloco digital implementado em linguagem Verilog com a síntese lógica do mesmo, utilizando um conjunto de bibliotecas de células da tecnologia definida;
- Realização de layout utilizando ferramenta de posicionamento e roteamento automática e análise de desempenho e de potência;
- Implementação do bloco em um CI de aplicação específica (ASIC) em processo CMOS definido e fabricação em corrida *Multi Project Wafer* (MPW);
- Elaboração de placa de testes e montagem do CI fabricado;
- Testes do chip e avaliação dos resultados.

1.4. Organização do Documento

O presente documento está dividido da seguinte forma:

Inicialmente, no Capítulo 2, apresenta-se revisão bibliográfica sobre o sistema de EBE e descreve-se a arquitetura do circuito de geração de sinais, apresentando-se duas propostas de sistemas para sua geração. A partir daí, é feita uma análise sobre o fluxo de projeto de circuitos digitais, considerando-se um projeto automatizado normalmente utilizado pelas empresas de projetos. Em seguida, faz-se uma abordagem sobre a linguagem de descrição de hardware Verilog e sobre as técnicas de codificação para otimização de área e de dissipação de potência.

O Capítulo 3 descreve os procedimentos realizados em cada etapa da implementação e dos testes do circuito NCO, sobre o qual foi realizada a aplicação das técnicas de otimização. São apresentados os resultados dos testes do circuito integrado fabricado, analisando-se, em comparação com outras arquiteturas

levantadas no estudo bibliográfico, os resultados obtidos de área e dissipação de potência.

No Capítulo 4 são apontadas as principais conclusões obtidas através dos resultados observados neste trabalho.

Finalmente apresentam-se as referências bibliográficas utilizadas nesta pesquisa.

Os códigos de descrição de hardware do circuito implementado são apresentados na seção de Apêndices.

2. REVISÃO SOBRE CIRCUITOS NCO E TÉCNICAS DE CODIFICAÇÃO EM NÍVEL RTL

Neste capítulo, são apresentados alguns conceitos básicos envolvidos na aplicação do sistema EBE. Na sequência, descrevem-se as características do bloco responsável pela geração dos sinais no sistema de medição da EBE, para definir as principais especificações que deve atingir para o adequado desempenho do sistema. Posteriormente, descreve-se a teoria de algoritmo e arquitetura de circuitos utilizados para realizar a geração de sinais. Também são discutidos alguns conceitos básicos no projeto de circuitos integrados digitais, apresentando-se uma revisão sobre o fluxo de projeto para CIs de aplicação específica (ASIC) baseado em células básicas. Em seguida, é abordada a linguagem HDL Verilog e, então, são descritas algumas técnicas de otimização que podem ser aplicadas durante a etapa de descrição do circuito em RTL.

2.1. Sistema de Espectroscopia por Bioimpedância Elétrica – EBE

Esse método consiste, inicialmente, na técnica da Espectroscopia, uma forma de mensurar uma quantidade como uma função da frequência e do comprimento de onda (NEWS MEDICAL, 2015). E a bioimpedância é o comportamento dos tecidos quando submetidos a estímulos elétricos de forma alternada ou constante. O desenvolvimento de um equipamento de medição da EBE para a detecção precoce do CCU possibilita obtenção de resultados com baixo custo, tornando os exames mais acessíveis e a obtenção de resultados mais rápida e segura, podendo, assim, atingir um maior número de mulheres (ABDUL, BROWN, *et al.*, 2006). A EBE, portanto, analisa os diferentes comportamentos dos tecidos a estímulos elétricos em diferentes frequências, de 100 Hz até 1 MHz, no caso da detecção do CCU. Para isso, normalmente, essa medida é feita através de quatro eletrodos, dois para a injeção e dois para a extração do sinal, como mostrado na Figura 1. Com os valores de corrente e tensão em mãos, calcula-se a impedância elétrica e analisa-se o gráfico em relação à frequência (diagrama de Bode). A partir da comparação do diagrama obtido com o de tecidos em diferentes estágios do CCU, pode-se determinar o nível da neoplasia presente.

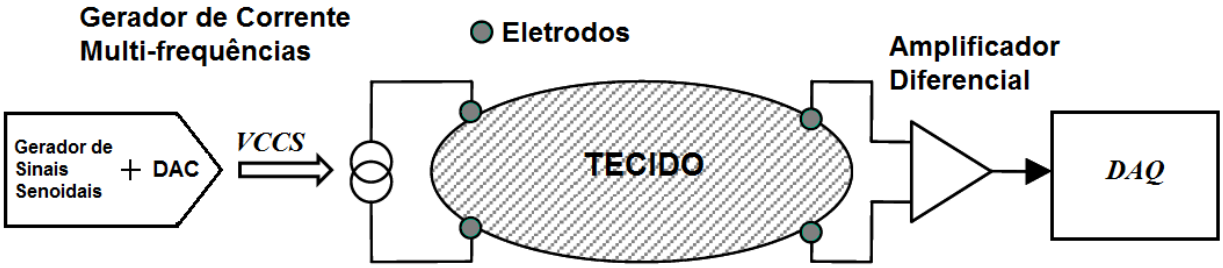

Figura 1 - Diagrama de blocos do sistema EBE

Os blocos principais do sistema EBE são: Gerador de Corrente Multi-frequências que, por sua vez, é composto por um Gerador de Sinais Senoidais, conversor Digital Analógico (*Digital Analog Converter* (DAC)) e fonte de corrente controlada por tensão (VCCS); eletrodos, que fornecem uma interface entre a condução elétrica no sistema e a condução iônica no tecido, sendo dois eletrodos para injetar o sinal elétrico no tecido e dois para extração do sinal; um amplificador diferencial de instrumentação, para amplificar o sinal extraído do tecido, devido sua baixa amplitude; e um módulo de Aquisição de Dados (*Data Acquisition* (DAQ)), para reconstrução e tratamento dos sinais extraídos e formatação para leitura dos dados (PALACIO e VAN NOIJE, 2015), (WALKER, 2001).

Existem diversas maneiras de se implementar o gerador de sinais que vão desde instrumentos de bancada, passando por circuitos analógicos do tipo *Phase-Locked Loop* (PLL), algoritmos em software para geração de sinais, tabelas implementadas em memórias, até chegar em circuitos digitais específicos. Uma forma adequada de gerar um sinal senoidal com valor de frequência controlado digitalmente é utilizando um Oscilador Controlado Numericamente (*Numerically Controlled Oscillator* (NCO)), posto que, por meio desse bloco, deve-se garantir a geração de um sinal de tensão de frequência variável com alta estabilidade. O sinal digital gerado pelo NCO deve ser convertido em um sinal analógico por meio de um conversor Digital – Analógico e, posteriormente, filtrado com o fim de eliminar os componentes de alta frequência do sinal senoidal requerido pela Fonte de Corrente Controlada por Tensão (*Voltage Controlled Current Supply* (VCCS)), a qual, finalmente, recebe o sinal de tensão e deve fornecer o sinal de corrente que será aplicado no tecido.

2.2. Arquiteturas de gerador de sinais - Oscilador Controlado Numericamente

O oscilador controlado numericamente, do inglês *Numerically Controlled Oscillator* (NCO), é um circuito digital gerador de sinais que fornece um sinal discreto no tempo, com frequência e fase precisamente controladas e referenciadas a uma frequência de *clock* do sistema. Quando utilizado em conjunto com um conversor Digital/Analógico (DAC) ligado à saída do NCO, este sistema é denominado Sintetizador Direto Digital de Frequências, *Direct Digital Synthesizer* (DDS).

Estes circuitos possuem vantagens em relação a outros tipos de osciladores baseados em circuitos analógicos, do tipo PLL, como, por exemplo: resoluções de micro Hertz na frequência de saída e capacidade de sintonia com resolução precisa, através de controle digital; mudança rápida entre duas frequências distintas de saída, sem distorções de tempo de acomodação típica de implementações analógicas; interface de controle digital que facilita o controle remoto de sistemas por meio de um processador; possibilidade de utilização como um sintetizador com quadratura, apresentando precisão na diferença de fase entre os dois sinais gerados. Outra grande vantagem é que este circuito fornece uma varredura de sinais com frequências diferentes, enquanto que, com o uso de outros tipos de osciladores, seria necessário o uso de vários componentes discretos e osciladores para cada faixa de frequência a ser escolhida. (FERREIRA, 2016), (IN-GI e WHAN-WOO, 2004), (KADAM, SASIDARAN, *et al.*, 2002).

A arquitetura básica desse circuito é demonstrada na Figura 2.

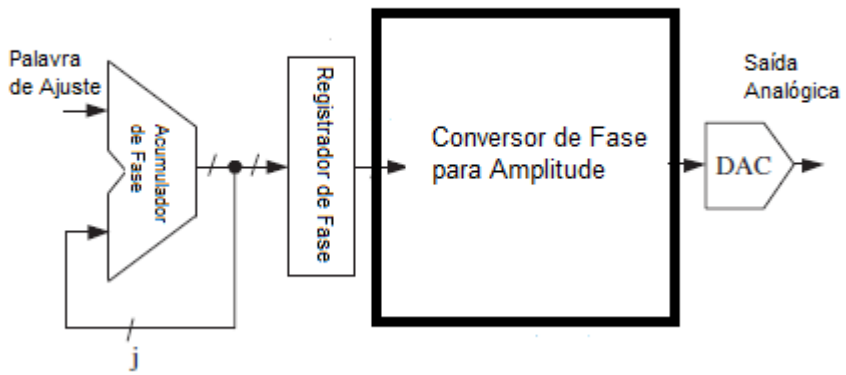

Figura 2 - Sistema DDS com entrada de controle de sintonia de sinal

O NCO é o núcleo do sistema DDS, sendo composto por um acumulador de fase formado por um contador de módulo de *N* bits e um registrador de fase e por um circuito Conversor de Fase para Amplitude, responsável por gerar um sinal senoidal de acordo com o valor da fase de entrada.

Durante a pesquisa, foram encontradas duas arquiteturas de circuito para conversão de fase para amplitude:

- Look Up Table (LUT) (FERREIRA, 2016);

- Algoritmo CORDIC (*COordinate Rotation Digital Computer*) (CARVALHO, PALACIO e VAN NOIJE, 2016).

2.2.1 Look Up Table (LUT)

A implementação de um NCO pode ser feita na forma mais simples, utilizando uma memória somente de leitura (*Read Only Memory* (ROM)) que armazena a informação da amplitude digital. A Figura 3 mostra o diagrama do NCO utilizando a ROM.

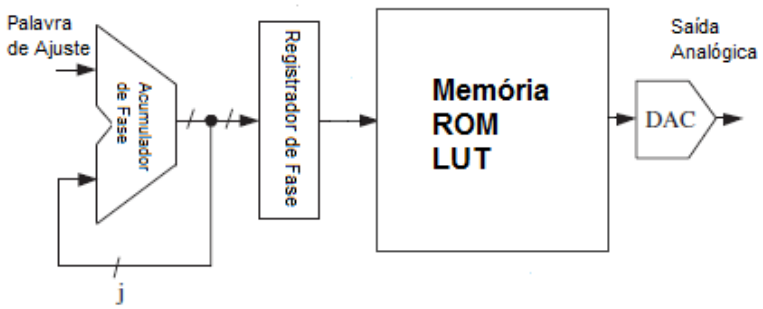

Figura 3 - Diagrama de blocos do NCO utilizando LUT

Essa memória possui uma tabela (LUT) de valores predefinidos que corresponde a um ciclo completo de senóide. Nessa abordagem, a LUT é projetada como o conversor de fase para amplitude, e o Acumulador de Fase calcula os endereços de

acesso a cada uma das posições da memória. Desta forma, o conteúdo da memória (valores de amplitude equivalente a uma senóide) são as entradas do conversor D/A. Um dos requisitos para essa abordagem é que o DAC deve ser de alta velocidade. A frequência de saída desta implementação é dependente da frequência do *clock* de referência e do tamanho do passo da amplitude senoidal programado na ROM. A vantagem dessa implementação é que a qualidade, a linearidade e o desempenho em Corrente Alternada (CA) são muito boas, conforme descrito por (RAIRIGH, LIU, *et al.*, 2009). A desvantagem desta abordagem é que a área ocupada em silício é relativamente maior que em outras abordagens (os valores encontrados nas referências são comparados com os resultados de área obtidos do circuito implementado neste trabalho, na Seção 3), além de não possuir uma flexibilidade na sintonia. Ou seja, a frequência de saída só pode mudar alterando-se a frequência do *clock* de referência ou pela reprogramação da memória, caso ela seja do tipo reprogramável.

2.2.2 O Algoritmo CORDIC

O CORDIC (*COordinate Rotation Digital Computer*) é um algoritmo simples, desenvolvido por Jack Volder em 1959, utilizado para calcular funções trigonométricas e hiperbólicas (VOLDER, 1959). Normalmente, emprega-se esse algoritmo em sistemas que não dispõem de estruturas de multiplicação ou que requerem menor área e consumo de potência. Devido às características do algoritmo, este sistema pode ser facilmente implementado em FPGAs para aplicações biomédicas (VACHHANI, SRIDHARAN e MEHER, 2009).

Para a aplicação proposta, dentre os modos de operação e as funções disponíveis, serão apresentados os algoritmos para os cálculos de seno e cosseno.

Os algoritmos são baseados no principio de que todas as funções trigonométricas podem ser calculadas através da rotação de vetores, através de um método iterativo aplicado por ângulos arbitrários, utilizando-se apenas somas e deslocamentos (*shifts e adders*). O algoritmo é derivado das transformadas de rotação dadas por (1a) e (1b):

$$x' = x\cos\varphi - y\sin\varphi \qquad (1a)$$

$$y' = y\cos\varphi + x\sin\varphi \qquad (1b)$$

As eq. (1a) e (1b) mostram a rotação de um vetor no plano Cartesiano pelo ângulo φ como mostra a Figura 4, onde para cada rotação do vetor (1, φ) nos ângulos de φ_1 a φ_6, obtém-se o valor final de x', que representa o cosseno, e y', que representa a função seno.

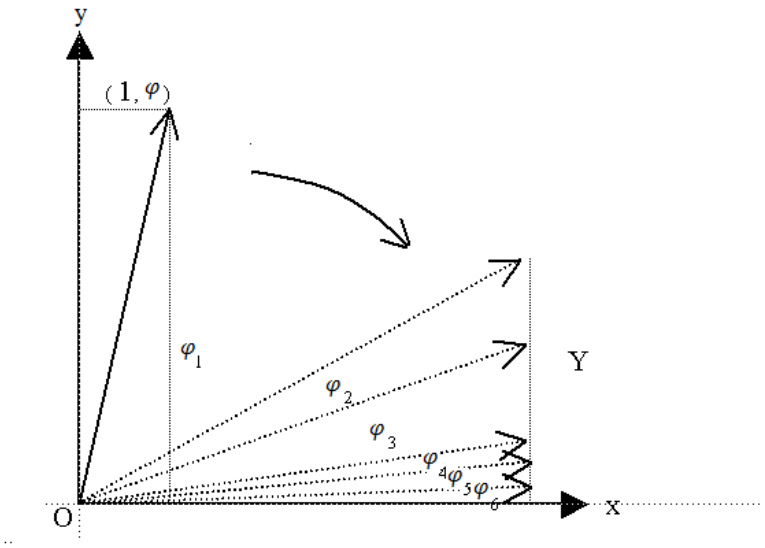

Figura 4 - Comportamento do CORDIC em execução

Rearranjando estas equações, obtêm-se (2a) e (2b):

$$x' = \cos\varphi\ [x - y\tan\varphi] \qquad (2a)$$

$$y' = \cos\varphi\ [y - x\tan\varphi] \qquad (2b)$$

Se o ângulo de rotação é restrito de tal forma que $\tan\varphi = \pm 2^{-i}$, então multiplicar por $\tan\varphi$ é equivalente a uma operação de deslocamento. Esta operação é mostrada nas eq. (3a) e (3b). O índice *i* representa o número de iterações do algoritmo CORDIC.

$$x' = \cos\left(\arctan\left(2^{-i}\right)\right)\left[x - y2^{-i}\right] \qquad (3a)$$

$$y' = \cos\left(\arctan\left(2^{-i}\right)\right)\left[y + x2^{-i}\right] \qquad (3b)$$

Na primeira iteração, quando *i=0*, o vetor de entrada é rotacionado em:

$$\tan^{-1}(2^{-0}) = 45°$$

Na segunda iteração rotacional, o vetor em: $tan^{-1}(2^{-1}) = 26.56°$, e assim por diante.

Em cada iteração *i*, uma decisão é tomada para rotacionar em direção ao ângulo final desejado. O ângulo de rotação vai se tornando sucessivamente pequeno a cada iteração até que o vetor converge para o ângulo desejado.

A transformada rotacional iterativa é mostrada nas equações (4a - e).

$$x_{i+1} = k_i\left[x_i - y_i * d_i * 2^{-i}\right] \qquad (4a)$$

$$y_{i+1} = k_i\left[y_i + x_i * d_i * 2^{-i}\right] \qquad (4b)$$

$$k_i = \cos\left(\arctan\left(2^{-i}\right)\right) \qquad (4c)$$

$$k_i = \frac{1}{\sqrt{1 + 2^{-2i}}} \qquad (4d)$$

$$d_i = \pm 1 \qquad (4e)$$

O termo *d* é sempre igual a +1 ou -1 dependendo da direção de rotação para a respectiva iteração. Quando *d=+1* o vetor é rotacionado no sentido anti-horário, enquanto que, quando é igual a *-1*, o vetor é rotacionado no sentido horário.

O termo

$$k_i = \cos\left(\arctan\left(2^{-i}\right)\right)$$

da eq. (4) torna-se uma constante para cada iteração. Cada termo k_i é independente do sentido de rotação, porque a função cosseno é uma função par, $\cos\varphi = \cos -\varphi$. Se o número total de iterações é fixo, os termos k_i podem ser fatorados. O produto de todos os termos k_i é representado como K_n, onde *n* representa o número total de

iterações, a partir da iteração 0. Eq. (5) mostra como calcular K_n pela multiplicação de todos os termos k_i de 0 a *n*. Para um número suficientemente grande de iterações (por exemplo, 10), K_n converge para um valor constante de aproximadamente 0,607253. O algoritmo rotacional calculado sem os termos k_i terão um ganho total de $1/K_n$ ou A_n representada pela eq. (6).

$$K_n = \prod_{i=0}^{n} \frac{1}{\sqrt{1+2^{-2i}}} \qquad (5)$$

$$A_n = \frac{1}{K_n} \qquad (6)$$

Uma terceira equação é adicionada para controlar o ângulo de rotação composto, sendo mostrada na eq. (7).

$$z_{i+1} = z_i - d_i \arctan\left(2^{-i}\right) \qquad (7)$$

O algoritmo iterativo é normalmente utilizado em um dos seguintes modos: Rotacional ou Vetorizado. No modo Vetorizado, a entrada é dada por um vetor $\underset{r}{\rightarrow} = (x, y)$ 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. No modo Rotacional, o vetor de entrada é rotacionado para um dado ângulo de entrada Z_0. O sinal do valor de z_i determina a direção de rotação para cada iteração, de forma que um valor absoluto é reduzido após cada iteração, obtido através das eqs. (8a), (8b) e (8c). As equações para o modo de operação Rotacional são apresentadas pelas eqs. (9a), (9b) e (9c) onde x_n e y_n representam o valor final de x e y.

$$x_{i+1} = x_i - y_i * d_i * 2^{-i} \qquad (8a)$$

$$y_{i+1} = y_i + x_i * d_i * 2^{-i} \qquad (8b)$$

$$z_{i+1} = z_i - d_i * \arctan\left(2^{-i}\right) \qquad (8c)$$

$d_i = -1$ se $z_i < 0$,caso contrário $d_i = +1$

$$A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}} \qquad (9a)$$

$$x_n = A_n[x_0 \cos(z_0) - y_0 \sin(z_0)] \qquad (9b)$$

$$y_n = A_n[x_0 \cos(z_0) + y_0 \sin(z_0)] \qquad (9c)$$

A função para calcular seno e cosseno utiliza apenas o modo Rotacional, através da imposição de condições iniciais a $y_0 = 0$ e x_0 igual ao valor inverso de A_n. Com estas condições iniciais, os valores finais não precisam ser reescalados.

O algoritmo demonstrado do sistema CORDIC pode ser implementado de forma direta através da estrutura demonstrada na Figura 5.

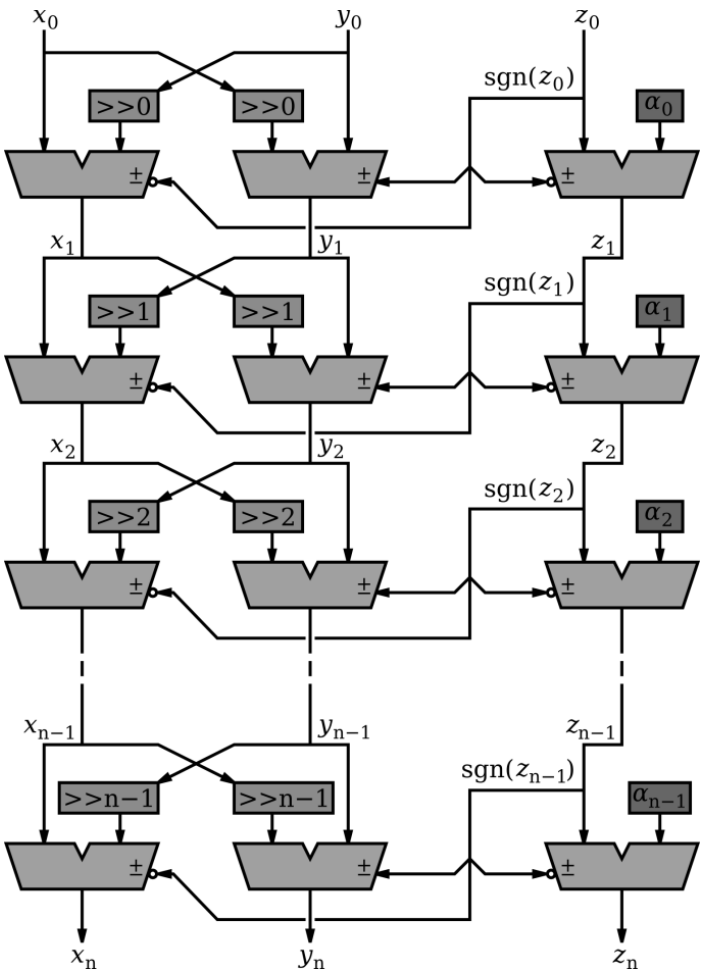

Figura 5 - Arquitetura do circuito CORDIC

Essa arquitetura é composta por vários estágios de iteração e é conhecida como Arquitetura Paralela Desenvolvida (*parallel unrolled architecture*) (SARAVANAN e RAMASANY, 2013). Nesta proposta, cada estágio composto de somadores e registros de deslocamento, utilizados para efetuar a operação de deslocamento do vetor, é responsável por calcular cada uma das iterações apresentadas nas eqs. (8a), (8b) e (8c). No último estágio, são obtidos os valores finais de x_n e y_n.

Outra arquitetura de CORDIC existente é a arquitetura Bit-Serial, onde é processado um bit por vez, reduzindo o número de interconexões entre os operadores. Essa arquitetura é apresentada na Figura 6 e é normalmente empregada para reduzir a área do circuito CORDIC, uma vez que os operadores de soma e subtração irão tratar apenas 1 bit por iteração. Porém, a desvantagem dessa arquitetura é que para obter o resultado final, é necessário o uso de um *clock* mais rápido do que o utilizado

na arquitetura paralela. Os registradores de deslocamento (*shift reg*) são carregados com os valores de inicialização um bit por vez. Uma vez carregados os valores, os dados são deslocados da esquerda para a direita, antes de chegarem aos somadores/subtratores seriais. Cada iteração é executada em *m* ciclos de relógio, onde *m* é a resolução do CORDIC em número de bits. Deslocadores seriais são implementados para capturar e deslocar os bits dos *shift registers*. Existe um circuito de controle que não é apresentado na figura, que fornece o bit de sinal do dado deslocado serialmente. Os resultados dos somadores seriais voltam para os *shift registers*, de forma que em *m* ciclos de relógio, os resultados da outra iteração são armazenados nos *shift registers*. Uma conversão completa demora $n \times m + 2$ ciclos de relógio, onde *n* corresponde ao número de iterações.

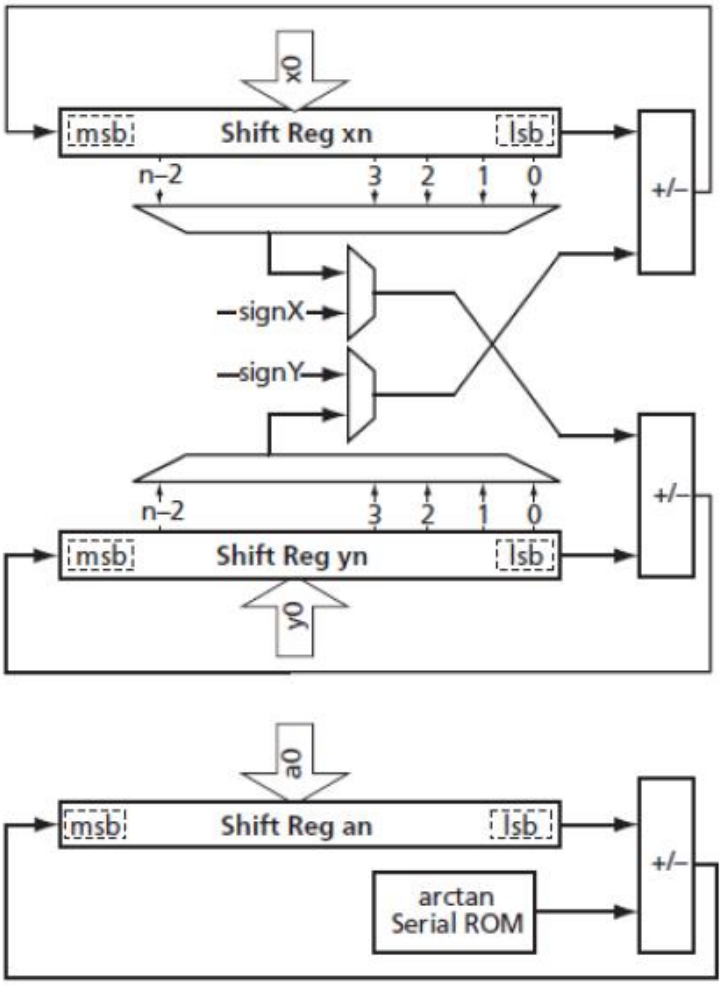

Figura 6 - Arquitetura CORDIC Bit-Serial (KUMAR e SAPPAL, 2011)

2.3. Fluxo de projetos para circuitos digitais

Um fluxo de projeto é definido como uma sequência de procedimentos que permite que os projetistas consigam avançar desde a especificação do projeto até a implementação física do circuito (WEST e HARRIS, 2005).

Existe uma gama de dispositivos que podem ser usados para construir um circuito digital customizado. Dentre as opções disponíveis, será discutido o fluxo de projeto que pode ser aplicado para *Standard-cell ASIC* (ASIC baseado em células base) e *Field Programmable Gate-Array* (FPGA).

No projeto de ASIC utilizando células base, um circuito é construído utilizando-se um conjunto predefinido de componentes lógicos, conhecido como *standard cells.* São células pré-projetadas e seus leiautes são previamente validados e testados. O uso desta tecnologia permite ao projetista trabalhar em nível de portas lógicas, que é o nível de abstração acima do nível de transistores, de forma a simplificar o projeto. Estas células são fornecidas pelo fabricante através de um conjunto de bibliotecas que possuem portas lógicas, componentes combinacionais simples, como multiplexadores, somadores de 1 bit, elementos de memória básicos como flip-flops tipo D, etc. A interconexão entre estas células e a escolha de quais tipos de células serão utilizadas dependem do tipo de circuito que será implementado durante o projeto e da ferramenta.

Um fluxo de projeto padrão de um ASIC é mostrado na Figura 7.

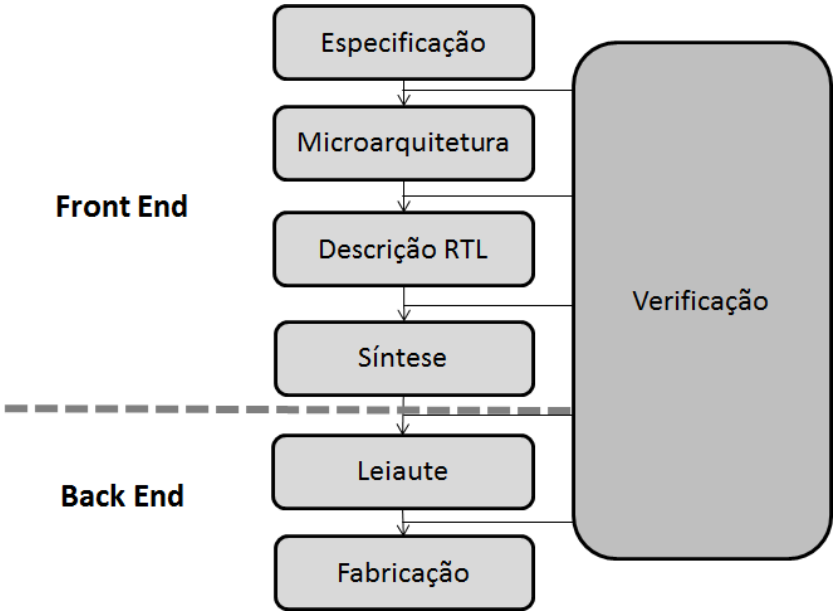

Figura 7 - Fluxo de Projeto de CIs

No fluxograma da Figura 7, o projeto é particionado em duas etapas principais: *Front end*, onde são englobadas as etapas de descrição comportamental até a síntese lógica; e *Back end*, que engloba as etapas estruturais e de implementação física.

O projeto inicia-se através da primeira camada de abstração do projeto, no nível comportamental, e segue para a próxima etapa estrutural em nível de chaves e registros ou *Register Transfer Level* (RTL). Esta etapa é chamada de síntese comportamental, porque é onde ocorre a captura do projeto descrito em nível HDL (elementos de memória e lógica) em RTL.

Essa descrição é transformada em uma descrição física apropriada para fabricação do chip.

Baseando-se numa dada especificação, o projetista forma uma ideia geral com relação à solução para o problema. Neste estágio, um diagrama de blocos é gerado para representar o projeto. Geralmente, ferramentas de EDA não são necessárias nesta fase.

Este estágio costuma ser definido com o conceito do projeto. Esta etapa é de vital importância para todo o projeto, pois consome muito tempo e exige importantes decisões, técnicas e gerenciais.

Tendo-se a ideia geral definida, os próximos passos para a continuação do projeto são:

• Projeto Funcional - Visa obter a descrição comportamental abstrata de um sistema que funcione de acordo com as especificações. Esta etapa pode considerar alguns aspectos estruturais ou físicos que influenciem o comportamento, principalmente através de estimativas, mas deve, preferencialmente, preocupar-se com a funcionalidade. Normalmente a especificação do sistema em linguagem humana será traduzida para descrições comportamentais de hardware como VHDL, Verilog ou SystemC, e diagramas de tempo (BHATNAGAR, 2002).

• Projeto Arquitetural - Tendo-se o comportamento de todo o sistema, escolhe-se as opções arquiteturais para sua subdivisão, e também as de implementação interna de cada subdivisão. Aqui se consideram estruturas como processadores, memórias, circuitos dedicados, e suas microarquiteturas internas como tipos de cache, *pipeline*, etc. Um dos modelos mais gerais de arquitetura de um sistema é como um conjunto de máquinas que se comunicam. Cada uma é formada por uma parte operativa (Data Path) e uma parte de controle (PC), sendo a PC uma máquina de estados finitos (FSM) que gera sinais de controle para fluxo e processamento dos dados na parte operativa (BHATNAGAR, 2002).

• Projeto Lógico - Nesta etapa refina-se o sistema estruturalmente. Módulos definidos na arquitetura são detalhados, e funções no domínio comportamental são traduzidas para o nível lógico estrutural. Usam-se portas lógicas, registradores, multiplexadores, pequenos macro módulos de biblioteca (parametrizáveis ou não), barramentos, etc. Estes elementos devem ser selecionados conforme sua disponibilidade na tecnologia de implementação escolhida, pois não serão mais alterados. Atualmente a transformação do projeto arquitetural para o lógico é realizada através de ferramentas EDA específicas, como o Design Compiler da Synopsys e o RTL Compiler da empresa Cadence (BHATNAGAR, 2002).

• Projeto de Leiaute - Também denominado de projeto físico, é o conjunto de passos necessários para, a partir da descrição estrutural do circuito

(composto por blocos, portas lógicas ou transistores), sintetizar a descrição geométrica final das máscaras, incluindo o leiaute daqueles que já o tem definido. Nesta etapa é praticamente obrigatório o uso de programas para realização automática das tarefas desejadas, em consequência do elevado número de elementos (BHATNAGAR, 2002), (PALNITKAR, 2003).

• Verificação do Leiaute versus Esquemático - Depois de se completar a descrição do leiaute, uma verificação leiaute versus esquemático (*Layout Versus Schematic* (LVS)) é realizada. Isto assegura que o leiaute está em conformidade com o esquemático. Nesta etapa do projeto, se necessário, são feitas diversas vezes a verificação LVS e após a verificação DRC (Design Rule Check), até que haja total conformidade com o esquemático (BHATNAGAR, 2002), (PALNITKAR, 2003).

• Checagem de Regras de Projeto (Design Rule Check - DRC) - Este estágio é dependente da tecnologia de processo que se utiliza para fabricar o chip. A verificação das regras de projeto assegura que as regras utilizadas no processo de fabricação não são violadas. O leiaute deve ser desenhado baseado nas limitações da regra de projeto. Como já foi dito anteriormente, nesta etapa o projeto comuta entre leiaute, LVS e DRC, até que haja conformidade (BHATNAGAR, 2002), (PALNITKAR, 2003).

• Preparação da base de dados, geração do CIF e GDSII - (formatos de arquivos contendo informações para a fabricação) - Este é o ultimo estágio antes do processo de fabricação (WEST e HARRIS, 2005).A maioria das *foundries*, fábricas de processos de fabricação de circuitos integrados, aceita submissões nestes formatos. Estas são em forma de arquivos que descrevem o layout do circuito integrado. Incorporando todos os detalhes necessários para a fabricação do chip (WEST e HARRIS, 2005).

• Fabricação - O circuito é fabricado por longos processos físicos e químicos a partir do seu leiaute completo, ou usando uma forma simplificada de implementação, dentre as descritas adiante. A fabricação de protótipos, através de processos feitos em lâminas multiprojetos, conhecidos como Multi Project Wafer (MPW) oferecidos pelas foundries, ou sua emulação, é importante para que o componente possa ser testado fisicamente e em

conjunto com seu ambiente de operação antes da produção em massa (BHATNAGAR, 2002), (WEST e HARRIS, 2005).

• Encapsulamento e Teste - No passado o que era uma simples escolha de um encapsulamento físico para o substrato de silício, hoje engloba opções e tarefas bastante complexas, como o uso de MCMs (Multi-Chip Modules), ou algoritmos para interconexão de pinos de um novo encapsulamento. Finalmente, o teste para garantir o funcionamento de cada componente após sua fabricação é complexo em termos de hipóteses e cobertura de falhas, geração e aplicação de vetores de testes. Já é um padrão a existência de circuitos de autoteste dentro dos próprios componentes (BIST); ao final desta fase, tem-se o circuito pronto para o uso (BHATNAGAR, 2002), (WEST e HARRIS, 2005).

2.4. Linguagem de Descrição de Hardware

As linguagens de descrição de hardware (*Hardware Description Language,* HDL), permitem aos projetistas modelarem concorrência de execução de processos em elementos de hardware e descreverem os circuitos digitais em nível RTL ou comportamental, etc. (WEST e HARRIS, 2005)

Costuma-se dividir as linguagens HDL em níveis de abstração:

- Comportamental: Este é o maior nível de abstração disponibilizado pelas linguagens. Um módulo pode ser descrito em termos de seu algoritmo sem possuir detalhes de implementação de hardware;
- Fluxo de Dados (*Register Transfer Level* – RTL): Neste nível o módulo é projetado especificando-se o fluxo de dados. O projetista informa como os dados fluem entre os registradores de hardware e como os dados são processados no projeto;
- Descrição em nível de portas lógicas (*netlist ou gate-level*): Estes módulos são projetados em termos de portas lógicas e interconexões entre suas portas. Projetar neste nível é similar a descrever um projeto em termos de um diagrama lógico de portas.

As HDLs apresentam como vantagens:

- Permitem que o circuito digital possa ser descrito em um nível abstrato sem vínculo com um processo de fabricação especifico. As ferramentas de síntese lógica automaticamente convertem a descrição RTL em uma descrição em nível de portas lógicas, não necessitando o reprojeto do circuito para diferentes tecnologias;
- Permitem que a verificação funcional possa ser feita nas primeiras etapas do fluxo de projeto, permitindo com que o projeto possa ser otimizado e corrigido nessa etapa;
- Possui similaridade com as linguagens de programação de software, facilitando a descrição do circuito de forma concisa e que permite facilitar a identificação de problemas comparada a uma descrição em nível de portas lógicas.

2.5. Linguagem Verilog

Verilog HDL (CHURIWALA e GARG, 2011), ou simplesmente Verilog, foi introduzida em 1983 pela Gateway Design Automation, proprietária desta linguagem de descrição de hardware e simulação.

Em 1988 foi apresentada pela empresa Synopsys uma ferramenta Verilog baseada não mais apenas na simulação, mas que também realizava síntese lógica desta linguagem. Em 1989 com a aquisição da Gateway pela Cadence Design System a popularidade da linguagem Verilog aumentou muito, passando a ser umas das mais utilizadas no projeto de circuitos integrados. A linguagem tem sua raiz sintática na linguagem C e permite que diferentes níveis de abstração sejam misturados num mesmo modelo.

Deste modo, o projetista pode definir um modelo de hardware em termos de chaves (*switches*), portas lógicas (*gates*), RTL ou código comportamental, necessitando dominar apenas uma linguagem para estímulos e para gerar projetos hierárquicos. As mais populares ferramentas de síntese suportam Verilog, tornando-a uma linguagem muito escolhida pelos projetistas. Todos os fabricantes disponibilizam bibliotecas em Verilog para simulação e síntese lógica. O uso de PLI (Programming Language Interface) é uma poderosa característica que permite ao usuário escrever

em um código C comum e interagir com a estrutura de dados internos do Verilog. Projetistas podem personalizar seu simulador Verilog de acordo com suas necessidades com o uso de PLI (CHURIWALA e GARG, 2011), (PALNITKAR, 2003).

2.6. Metodologia de projeto

Há dois tipos básicos de metodologia de projeto: a *top-down* apresentada na Figura 8 e a *bottom-up* apresentada na Figura 9. Na metodologia de projeto *top-down* define-se o bloco principal e depois se identificam os sub-blocos necessários para compor o bloco principal. A divisão dos blocos e sub-blocos é feita até o nível das células básicas. Na metodologia de projeto *bottom-up* é realizado o caminho inverso. Identificam-se as células básicas que estão disponíveis, que são organizadas formando um bloco maior. Este procedimento é realizado até chegar a um bloco principal.

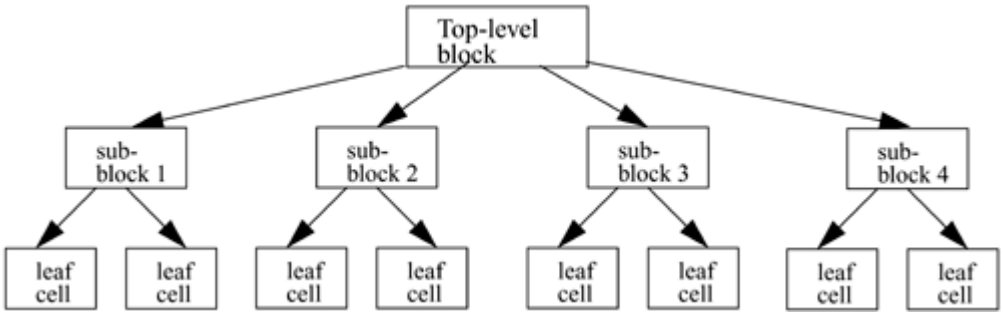

Figura 8 - Metodologia *Top-down* (PALNITKAR, 2003)

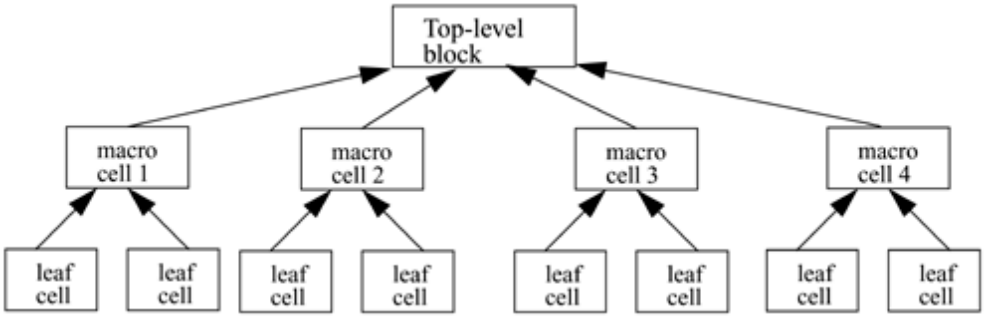

Figura 9 - Metodologia *Bottom-up* (PALNITKAR, 2003)

Existe uma terceira abordagem de metodologia de projeto que é tipicamente adotada, conhecida como *middle-out.* Essa metodologia é uma combinação das metodologias *top-down* e *bottom-up*, onde um grupo de projetistas desenvolve uma biblioteca de células básicas e circuitos primitivos otimizados, enquanto outro grupo define como o projeto deve ser estruturado, quebrando a funcionalidade em blocos e sub-blocos, construindo macro células utilizando as células básicas desenvolvidas pelo primeiro grupo de projetistas.

Em Verilog existe o conceito de módulo, que é a construção de um bloco básico. Um módulo pode ser apenas um elemento ou uma coleção de elementos em menor nível com uma funcionalidade comum que pode ser usado em vários lugares no projeto. O módulo supre a necessidade funcional de um bloco de maior nível através de suas portas de interface (entradas e saídas), mas esconde sua implementação interna. Isto permite aos projetistas modificar módulos internos sem afetar o resto do projeto. Em Verilog um módulo é declarado por uma palavra-chave module. Uma palavra-chave correspondente *endmodule* deve aparecer no final da definição do módulo. Cada módulo deve ter um *module_name* que é identificado por um módulo e um *module_terminal_list* que descreve os terminais de entrada e saída do módulo. O trecho de código abaixo mostra a estrutura de um módulo tipico descrito em linguagem de hardware Verilog (PALNITKAR, 2003).

```
module <module_name> (<module_terminal_list>);
...
<module internals>
...
...
endmodule
```

O Verilog permite ao projetista combinar todos os quatro níveis de abstração em um projeto. Normalmente, o nível mais alto de abstração é o mais flexível, e o projeto torna-se independente da tecnologia. É possível realizar uma inserção de módulos já criados, denominados instâncias. Quando o módulo é invocado, a linguagem Verilog cria um *template* em que cada objeto tem seu próprio nome, variáveis, parâmetros e interfaces de entrada e saída.

A funcionalidade do projeto pode ser testada pela aplicação de estímulos e verificação de resultados. O bloco que realiza esta tarefa é chamado bloco de estímulos, e pode ser escrito em Verilog. Dois estilos de aplicação de estímulos são possíveis. No primeiro estilo, o bloco de estímulo instancia o bloco do projeto e controla os sinais diretamente no bloco do projeto. O segundo estilo é inserir o bloco de estímulos e de projeto em um módulo principal. Assim o bloco de estímulos interage com o bloco do projeto só através da interface. O módulo de estímulo controla os sinais de entrada do bloco do projeto e verifica o sinal de saída.

2.7. Síntese Lógica

Como visto nas seções anteriores, a linguagem Verilog suporta vários níveis de abstração e estilos de descrição, incluindo a descrição estrutural, fluxo de dados e descrição comportamental. Permitem misturar vários níveis de entrada do projeto, incorporando todos os níveis de abstração e minimizando a lógica necessária, resultando em um *netlist* final na tecnologia de fabricação escolhida. Todas as construções da linguagem são simuláveis, mas nem sempre sintetizáveis, porque existem muitas construções na linguagem que não têm representação válida em circuitos digitais devido à sua abstração.

Um projeto descrito em Verilog, depois de simulado pode ser sintetizado e otimizado, sendo possível ainda ao usuário misturar descrições HDL com esquemáticos. A hierarquia do projeto e o modo com que este é particionado podem influenciar no resultado da síntese. O particionamento do projeto pode ser baseado na funcionalidade, mas em alguns casos pode ser feito apenas com propósitos de síntese. Os objetivos principais de particionamento para síntese são produzir o melhor resultado, melhorar a velocidade e aperfeiçoar o tempo e simplificar o processo de síntese. Para tanto, há uma série de medidas que podem ser tomadas no sentido de otimização da síntese, por exemplo, limitar o tamanho dos blocos pela quantidade de portas lógicas. Os tempos de execução dos algoritmos de síntese não são lineares, portanto, se um bloco a ser sintetizado é grande demais, sua execução pode demorar um longo tempo.

Separar subprojetos com objetivos diferentes em módulos e registrar todas as entradas e saídas de um bloco são uma boa prática para simplificar o processo de

síntese; lógicas com características similares devem ser agrupadas para melhorar a qualidade da síntese.

2.8. Simulação

O propósito principal da simulação funcional é o de ajudar na verificação da exatidão de um projeto ASIC. Por definição, simulação digital é um processo auxiliado por computador de exercitar o modelo teórico de um projeto como uma função do tempo, por alguns estímulos na entrada. O modelo teórico consiste na descrição de como os componentes no projeto são conectados, chamado de conectividade, e de modelos de simulação que descrevem a funcionalidade de cada um dos componentes. A sequência de entrada é referida como estímulos ou vetores. A conectividade é obtida como resultado da captura de esquemáticos ou do processo de síntese. A simulação lógica é usada para verificar se o projeto se encontra dentro das especificações pretendidas. A habilidade do projetista em usar o simulador determina o grau de confiança no projeto, aumentando o nível de qualidade na produção de ASIC.

2.9. Técnicas de otimização

Muitas são as técnicas e os processos utilizados nas várias etapas do projeto e fabricação de um circuito integrado para uso específico (ASIC). Todas estas técnicas buscam diminuir o custo de produção diminuindo a área de silício ocupada pelo circuito, ou aumentar o desempenho, incrementando a freqüência de trabalho e diminuindo o consumo de energia. Tenta-se otimizar ao máximo os circuitos projetados para atender aos requisitos cada vez mais restritos de consumo, desempenho e durabilidade, impostos pelo concorrente mercado de semicondutores (CADENCE, 2015).

Como mencionado anteriormente neste trabalho, pesquisou-se algumas técnicas de codificação em linguagem de descrição de hardware Verilog, que propõe a otimização do circuito a ser implementado. Técnicas estas que são empregadas na fase de codificação do projeto e visam melhorar os resultados da síntese Lógica, onde a linguagem HDL é transformada em portas lógicas.

As ferramentas de síntese automática como Encounter RTL Compiler da fabricante Cadence Inc., já realizam diversas otimizações no momento do processo de síntese. Estas otimizações ocorrem na sua maioria nos trechos de código HDL que implementam lógica combinacional, por ser mais fácil implementar algoritmos de otimização para este tipo de lógica (CADENCE, 2015).

A otimização algorítmica empregada pelas ferramentas de síntese automática nem sempre pode otimizar ao máximo um projeto. Baseando-se nisso, algumas destas técnicas de codificação que serão estudadas tentam aumentar o controle sobre a lógica gerada pelas ferramentas de síntese automática, fazendo com que as mesmas gerem estruturas já estudadas e simplificadas, com isso proporcionando um circuito mais otimizado.

Um ponto importante é a questão das ferramentas de síntese automática permitir elevar o nível de abstração através de descrições de alto nível do hardware, não se necessitando realizar a descrição individual de cada porta lógica ou até mesmo transistor. Algumas das técnicas apresentadas neste trabalho vão de encontro a esta filosofia das ferramentas, onde se utiliza um nível mais baixo de abstração na linguagem de descrição de hardware para conseguir um processo de síntese mais controlado e se obter o resultado desejado.

Na literatura, encontrou-se um estudo comparando dois níveis de abstração da codificação de uma linguagem HDL com o resultado de área da síntese lógica utilizando-se ferramentas da fabricante Synopsys. Weijun Zhang (ZHANG, 2001) comparou várias codificações em nível comportamental com suas equivalentes codificadas em nível RTL, utilizando a linguagem de descrição de hardware VHDL (IEEE 2004).

O estudo de Zhang obteve resultados positivos quando da codificação em um nível menos abstrato como é mostrado no gráfico da Figura 10. Na coluna, é apresentado, por exemplo, Total, a área (*gates*) consumida pelos dois níveis de codificação, comportamental e RTL. Pode-se notar uma redução muito grande no número total de portas lógicas do nível comportamental para o nível RTL. O gráfico também apresenta o número de portas lógicas utilizadas por blocos sequenciais e blocos combinacionais. Nota-se uma inversão nos valores, para a codificação comportamental, onde o número de portas lógicas utilizadas em blocos sequenciais

é maior. Já para a codificação RTL, o número de portas lógicas utilizadas em codificação combinacional é maior. Isso demonstra que a síntese automática de descrições comportamentais infere na sua maioria circuitos sequenciais do tipo máquinas de estado. Já na codificação RTL onde os blocos gerados pela síntese automática possuem um controle mais rígido, pode-se forçar a ferramenta a utilizar circuitos específicos e melhorar os resultados.

Ainda no gráfico, observamos nos quatro pares de colunas mais a direita, os números de portas de entrada e saída, conexões, células padrões e referências de potencial foram utilizadas para as duas implementações.

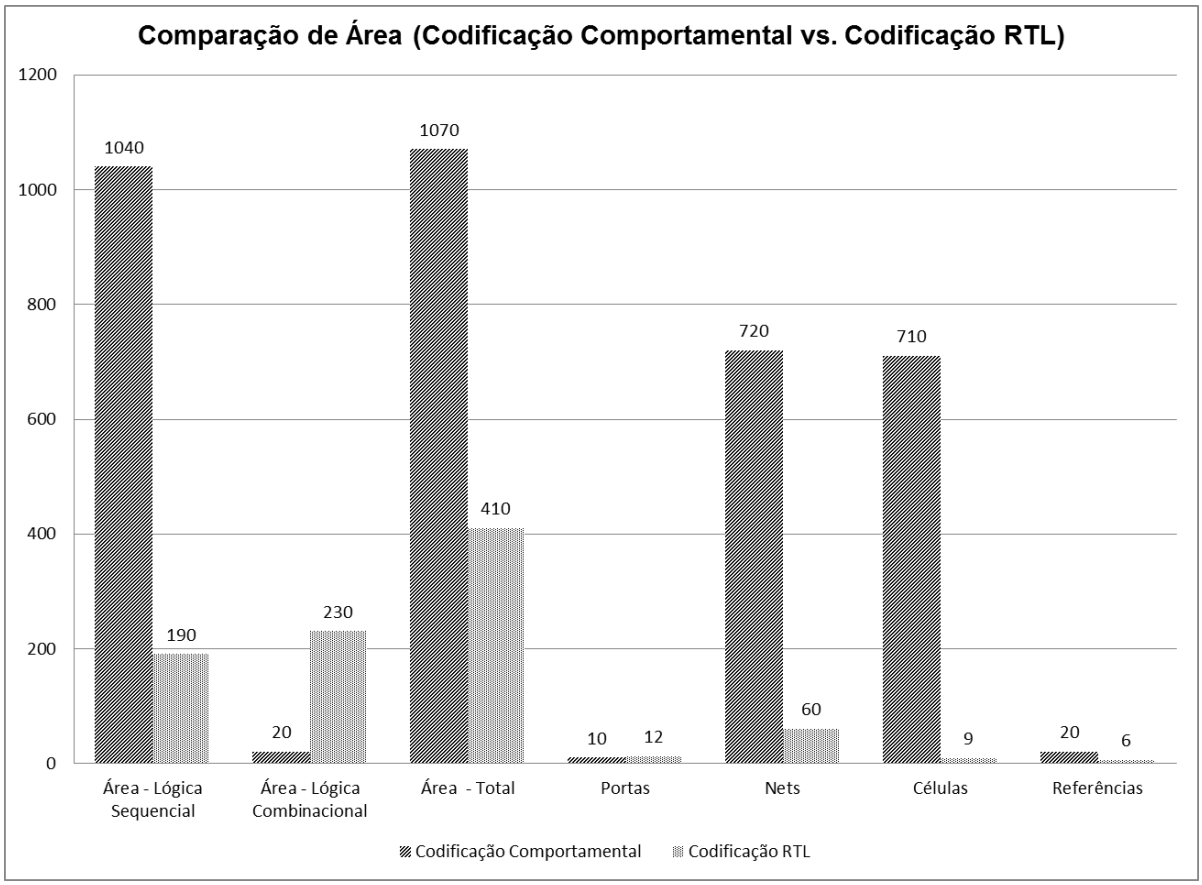

Figura 10 - Estimativas de área levantadas por (ZHANG, 2001)

Com base nesse estudo, a existência de um ponto de equilíbrio ótimo entre o nível de abstração da descrição de hardware, otimização da síntese e custo da implementação pode ser conseguido.

Para tanto será necessário o estudo e a análise destas técnicas propostas, discutindo-se a situação especifica onde as mesmas devem ser aplicadas.

A seguir serão descritas algumas das principais técnicas de otimização pesquisadas na literatura e que são utilizadas no projeto de um ASIC. Tais técnicas se restringem em nível de descrição de hardware através de linguagem HDL, mais especificamente a linguagem Verilog. O estudo destas técnicas e o impacto de sua implementação através da linguagem Verilog na síntese lógica são um dos principais objetivos deste trabalho.

Após levantamento em literatura das técnicas, classificaremos as mesmas em dois grandes grupos: técnicas de otimização de potência e área.

As técnicas de potência são aquelas que buscam diminuir a potência total dissipada, sendo aqui empregadas técnicas que reduzem a potência dinâmica. As técnicas de redução de área ocupada em silício buscam nesta etapa de codificação HDL, a redução de portas lógicas.

2.9.1 Técnicas de codificação para redução de potência dissipada

É conhecido na literatura (WEST e HARRIS, 2005) que três componentes influenciam na potência total dissipada em um circuito integrado. A primeira é a corrente de fuga existente na junção dos transistores da estrutura CMOS, uma característica específica dos materiais utilizados na fabricação. A segunda é a potência dissipada estaticamente, resultante do processo de polarização dos transistores que também é bastante influenciada pelo material utilizado, e por variáveis ambientais como temperatura, campo magnético entre outras. A terceira componente é a potência dissipada dinamicamente, resultante do processo de chaveamento ou troca de estado dos transistores.

O cálculo da potência total dissipada P_{TOTAL} é dado pela eq.(8):

$$P_{TOTAL} = P_{ESTÁTICA} + P_{DINÂMICA} \quad (8)$$

Onde $P_{ESTÁTICA}$ é a potência estática, dada pela eq.(9):

$$P_{ESTÁTICA} = I_{LEAKAGE} * V_{DD} \quad (9)$$

onde $I_{LEAKAGE}$ é a corrente total de fuga (tendo como potenciais fontes: corrente sublimiar, corrente de fuga de porta do transistor e corrente de junção polarizada reversamente) e V_{DD} é a tensão de alimentação.

A potência estática é considerada como “nula” para processos CMOS antigos. Já em processos abaixo de 65 nm, a potência estática se torna um componente importante a ser considerado no projeto. Os dados de potência estática são fornecidos pela *foundry* dentro dos arquivos da biblioteca de células, conforme mostrado na Figura 11.

```
leakage_power ()  {
  value : 0.000000E+00;
  when : "A&!B";
  power_level : "VDDNW";
}

leakage_power ()  {
  value : 4.430000E+02;
  when : "A&!B";
  power_level : "VDD";
}
```

Figura 11 - Descrição de potência estática de uma célula (WEST e HARRIS, 2005)

A Potência dinâmica $P_{DINÂMICA}$ é composta de dois elementos: Potência de chaveamento e Potência Interna.

A potência de chaveamento é dada pela eq. (10):

$$P_{CHAVEAMENTO} = \alpha * C_L * V_{DD}^2 * f \quad (10)$$

Onde α é o fator de atividade de chaveamento do sinal elétrico na trilha (transições lógicas entre ‘0’ e ‘1’), C_L é dada pela capacitância da trilha de interconexão entre os dispositivos, conhecida como C_{WIRE}, somada as capacitâncias internas das células-padrão, conhecida como C_{IN}, representadas como $C_{IN}1$ e $C_{IN}2$, conforme mostrado na Figura 12. O parâmetro *f* é a frequência do sinal de clock que controla o circuito ou a trilha em análise.

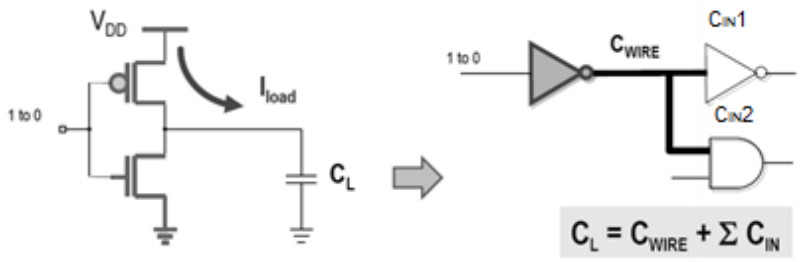

Figura 12 - Representação dos parâmetros da potência de chaveamento

A potência interna, presente como um dos componentes da potência dinâmica, é dada em função da corrente de curto circuito interna presente no momento em que os transistores PMOS e NMOS do dispositivo estão parcialmente ligados durante a transição do sinal de entrada na porta de ambos por um curto período de tempo enquanto o valor do sinal de tensão da entrada está entre a tensão de limiar do transistor NMOS (V_{Tn}) e V_{DD} - $|V_{Tp}|$, onde V_{Tp} é a tensão de limiar do transistor PMOS. Este parâmetro pode ser desconsiderado dos cálculos, se a transição dos sinais de entrada da célula não for mais lenta que a transição de saída (normalmente é considerada nestes casos como menor que 10% da corrente de carga da saída da célula) (WEST e HARRIS, 2005).

Por se tratar de uma parte significativa, podendo chegar em média a 70% do valor total de potência dissipada , segundo (WEST e HARRIS, 2005), (CHURIWALA e GARG, 2011) *e* (VAHID, 2011), este trabalho estará focado em técnicas de redução de potência dinâmica e estática, até mesmo porque, a nível de descrição de hardware, não existem otimizações possíveis para corrente de fuga, que são características do processo de fabricação empregado na confecção do circuito.

2.9.1.1 Inversão de Barramento (*Bus inversion*)

Esta técnica é muito empregada para o envio de dados entre chips diferentes e por barramentos de placas. Passou a ser utilizada internamente no chip devido à facilidade de implementação e geração de redução significativa na potência, pois reduz o número de chaveamentos dos transistores, avaliando o dado a ser transmitido. Com essa redução de chaveamentos tem-se uma redução da potência dinâmica.

Na Figura 13 é possível ver o funcionamento da técnica de inversão de dados no barramento que reduz o chaveamento dos transistores. A coluna da esquerda representa a sequência de transmissão dos bits de modo normal, onde os bits em cinza representam os bits transitados de um estado para outro (cada linha representa um estado de transmissão), a coluna da direita representa a sequência de transmissão dos bits com a técnica de inversão de barramento, onde se acrescenta um bit a mais na palavra de dados transmitida para controle do modo de representação, da mesma forma os bits em cinza representam bits transitados de um estado de transmissão para o outro.

A função do bit de controle adicionado é indicar quando o dado representado no barramento é invertido, caso o próximo dado a ser transmitido possui mais de 50 % de seus bits invertidos em relação ao dado anterior, ele é representado de modo invertido no barramento, reduzindo os chaveamentos dos transistores como pode se observar pelo total de transições em cada um dos casos (DAHAN, 2007).

Um ponto importante a ser levado em conta é a necessidade do desenvolvimento de uma lógica de controle para a inversão do barramento e a inserção de armazenamentos para realizar a comparação dos estados, o que pode aumentar a complexidade da lógica e o número de portas utilizadas (ARORA, 2012), (STAN e BURLESON, 1995).

A vantagem desta técnica, apesar do acréscimo de lógica para realizar a inversão dos dados, é que ela reduz o chaveamento dos sinais em cada trilha do barramento de forma a reduzir a dissipação de potência de chaveamento nas trilhas. Para cada trilha do barramento é associada uma capacitância. Logo, se o comprimento do barramento for muito longo, essa capacitância é maior, de forma que sempre que houver uma transição de um sinal em cada trilha, haverá uma dissipação maior de potência de chaveamento do que em trilhas com menor comprimento. Reduzindo-se o número de transições dos bits nas trilhas do barramento, diminui-se o número de bits que mudam e, portanto, reduz a dissipação de potência de chaveamento no barramento.

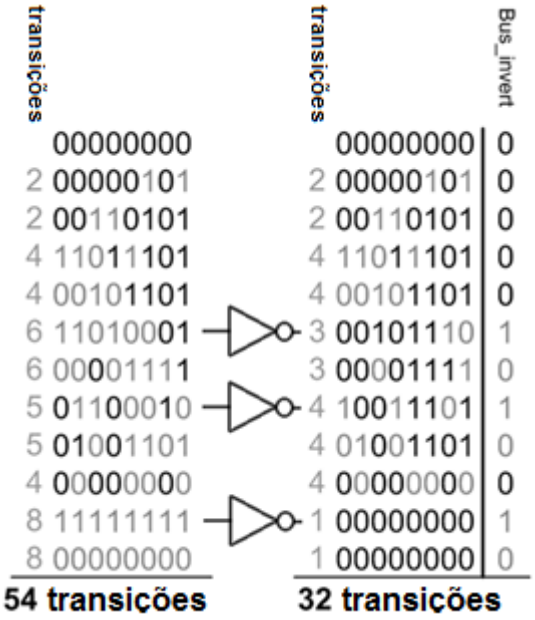

Figura 13 - Técnica de inversão de barramento - número de transições (ARORA, 2012) (STAN e BURLESON, 1995)

2.9.1.2 Representação de números binários (Binary Number Representation)

Esta técnica consiste no simples fato de analisar a melhor maneira de representar um número binário no projeto, dependendo da faixa numérica que se quer representar.

A representação em complemento de dois é a mais usada na prática. Embora haja a questão da inversão de maior número de bits em representações que variem de números negativos a positivos perto do zero maior do que em magnitude de sinal, como pode ser visto na Figura 14, um ponto positivo para a representação em complemento de dois é o da lógica aritmética para somas e subtrações ser mais simplificada do que na representação magnitude de sinal.

Complemento de 2	Magnitude de sinal
0 – 00000000	**0** – 00000000
-1 – 11111111	**-1** – 10000001

Figura 14 - Representação Numérica

Nesta situação, existe um *trade-off* que precisa ser considerado caso a caso, pesando-se transições na representação dos bits e aumento de portas lógicas, devido ao fato de se necessitar de lógicas mais complexa para realizar somas e subtrações na representação sinal magnitude (DAHAN, 2007), (ARORA, 2012).

2.9.1.3 Habilitar/desabilitar Lógica (Enabling/Disabling Logic Clouds)

Esta técnica consiste em implementar mecanismos que desabilitem partes do circuito que não estejam sendo usadas, geralmente desabilitando o clock do bloco ou lógica. Também conhecida como *clock gating*, sua implementação deve ser empregada com cuidado, pois precisa de mecanismos auxiliares que garantam o sincronismo do clock no momento da habilitação do circuito (EMNETT e BIEGEL, 2000). Normalmente, a implementação de clock gating é feita pelas ferramentas de síntese, desde que na descrição da lógica exista um sinal de habilitação comum para um conjunto de flip-flops.

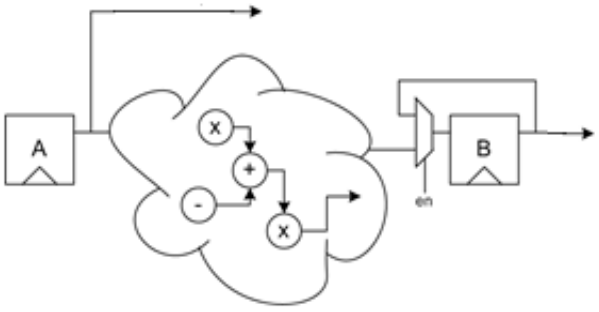

Figura 15 - Controle de dados da lógica – habilitação do caminho de dados

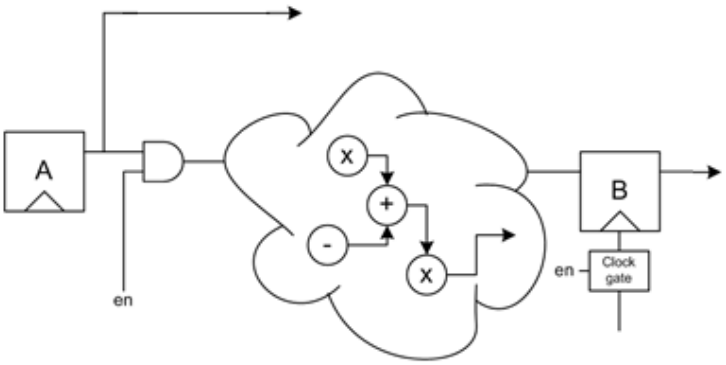

Figura 16 - Controle de habilitação de clock

A Figura 15 e a Figura 16 demonstram dois casos de aplicação de habilitação / desabilitação de lógica para reduzir chaveamentos desnecessários. Na Figura 15, é feito o controle de transição de um dado novo através de multiplexador inserido na entrada do registrador B, de forma que sempre que existir um dado novo, habilita-se o mux através do sinal *en*, permitindo que o registrador carregue o novo dado para a sua saída, caso contrário, mantém-se o dado antigo, evitando a transição dos dados desnecessários à cada ciclo de relógio. Na Figura 15, o controle da transição do dado pelo registrador B é feito através de *clock gating*, onde se insere uma porta lógica E na entrada do sinal de relógio do registrador e conecta a linha de relógio em uma das entradas da porta e a outra entrada é ligada ao sinal de controle *en*, permitindo desta forma, o controle da mudança do sinal de relógio de forma que o sinal de relógio ocorra apenas quando é habilitado o sinal *en*.

2.9.1.4 Lógica com alta taxa de comutação (High Activity Nets)

Esta técnica consiste em discriminar sinais que tenham um alto nível de transição nas expressões lógicas, e que ocasionam a cada transição o chaveamento de muitos dos demais transistores envolvidos nas expressões.

Os casos de interesse para otimização ocorrem em expressões lógicas que têm sinais atualizados esporadicamente juntamente com outros atualizados, por exemplo, a cada comutação do relógio, onde a variação de um dos dados implica em toda uma modificação dos resultados intermediários da lógica até ser propagada para o resultado final (DAHAN, 2007), (ARORA, 2012).

A Figura 17 e a Figura 18 mostram como a separação do sinal pode ser realizada. São replicadas as nuvens de lógica com o sinal que possui alta transição de estado, no caso o sinal Y, fixado para cada uma das nuvens em um valor diferente. Estas nuvens de lógica são conectadas a um multiplexador que selecionará para a saída o valor desejado conforme o sinal Y. Note que a cada transição do sinal Y apenas o chaveamento do multiplexador é necessário, e os resultados da nuvem lógica não são alterados. Com isso ocorre uma diminuição dos chaveamentos, e consequente diminuição da potência dinâmica, porém cabe salientar a existência de um aumento de portas lógicas que deve ser levado em consideração no momento de se aplicar a técnica.

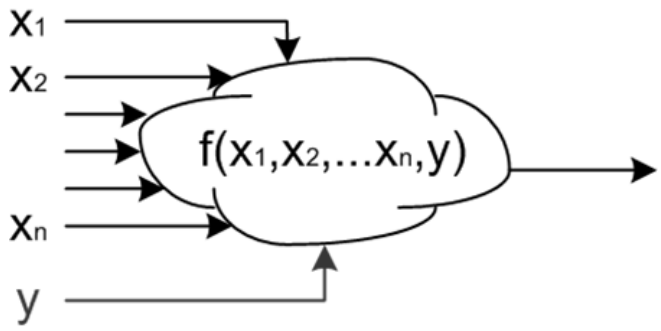

Figura 17 - Sinais com várias transições misturados a lógica normal

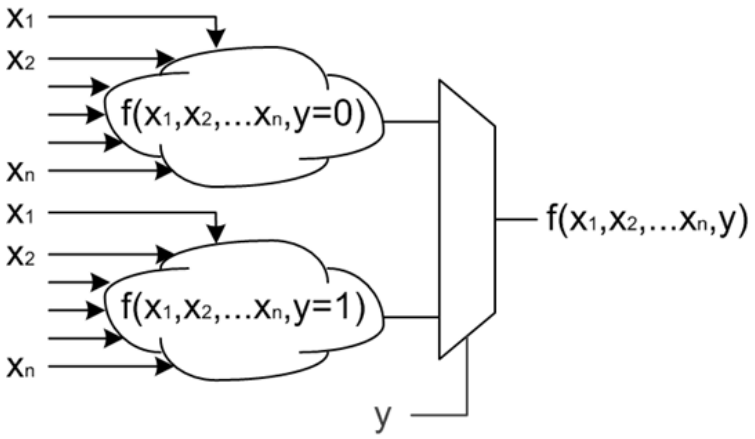

Figura 18 - Sinal com muitas transições separado do resto da lógica

2.9.1.5 Codificação de Máquina de Estado e Decomposição

Entre os diversos métodos de codificação de máquinas de estado finito (*Finite State Machine* (FSM)), a codificação em código Grey é indicada como uma das melhores estratégias de codificação para projetos de baixo consumo. A Figura 19 compara uma máquina de estado com os estados codificados em binário com outra codificada em código Grey. Nesta imagem, pode-se perceber que na codificação binária (figura à esquerda), há mais de um flip-flop que pode chavear durante a transição de estados como, por exemplo, a transição do estado D (“011”) para o estado E (“100”), consumindo maior potência do que a mesma máquina implementada em código Grey (figura à direita), no mesmo caso de transição, onde somente um flip-flop muda de estado durante a transição.

A implementação da máquina de estado com esse tipo de codificação ajuda a eliminar possíveis pulsos e riscos nas equações combinacionais que dependem do estado.

Outra proposta para implementação de máquinas de estado é a decomposição delas. A ideia básica é de decompor o gráfo de transição de estados (*State Transition Graph*) de uma máquina de estado em dois STGs que juntos, produzem o comportamento equivalente das entradas e saídas. Com esta estratégia, consegue uma redução de potência, pois somente uma das submáquinas necessita de *clock*, com exceção para transições entre as duas submáquinas.

A técnica consiste em particionar os estados, procurando por um conjunto pequeno conjunto de estados com maior probabilidade de transição entre estes estados e uma baixa probabilidade de transições de/para outros estados. Este subconjunto então irá constituir uma pequena submáquina de estado que estará ativa a maior parte do tempo, permitindo que a outra submáquina possa ser desabilitada.

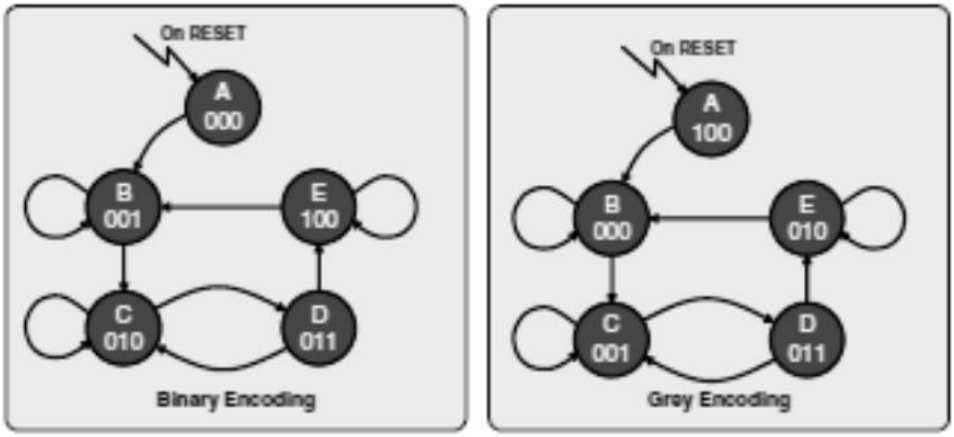

Figura 19 - Codificação binária versus codificação Grey (ARORA, 2012)

2.9.1.6 Multiplexador codificado “One-Hot”

Existem diversas formas de inferir multiplexadores em RTL, seja por declarações do tipo “case”, “if-else” ou máquinas de estado. A forma mais comum de se representar um multiplexador usando codificação binária é apresentada na Figura 20. Note que se cada entrada do MUX é composta de barramentos de vários bits, haverá uma

quantidade significativa de chaveamento e, consequentemente, alto consumo de potência.

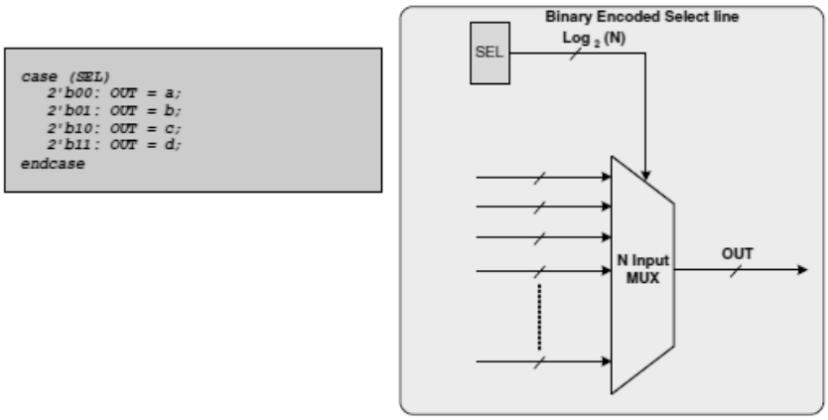

Figura 20 - Codificação binária da linha de seleção do MUX (ARORA, 2012)

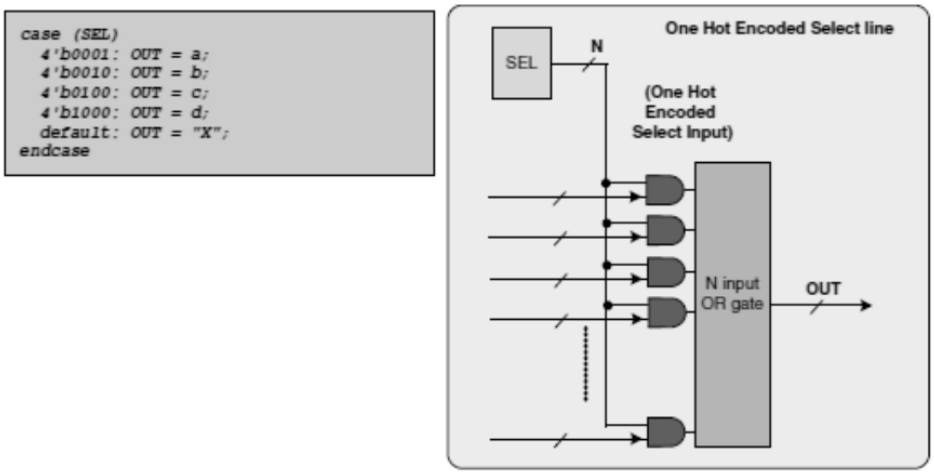

Figura 21 - Codificação "one-hot" da linha de seleção do MUX (ARORA, 2012)

Uma alternativa para esta implementação, é a codificação do sinal de seleção do MUX através de codificação do tipo “one hot”, como mostra a Figura 21. Esta implementação possui menos efeitos de estabilização do dado, uma rápida resposta de saída e as mudanças que ocorrem em barramentos não selecionados são mascaradas previamente, evitando falsas transições de dados, reduzindo significativamente o consumo de potência, conforme (ARORA, 2012).

2.9.1.7 Removendo transações redundantes

Muitas vezes num sistema, os dados presentes num barramento podem mudar de um valor para outro, caso não tenha um estado padrão definido, gerando operações redundantes e um consumo elevado de potência. Dada essa situação, é recomendado para que não haja alteração de dados no barramento enquanto estes dados não são amostrados pelo sistema.

A Figura 22 mostra um exemplo de transição redundante onde todos os operandos *a_in, b_in, c_in* e *d_in* são carregados, mas todas as saídas não são usadas. Note que o sinal *load_op* não pode ser ativado antes que o sinal *load_out* seja ativado, para economizar consumo de potência.

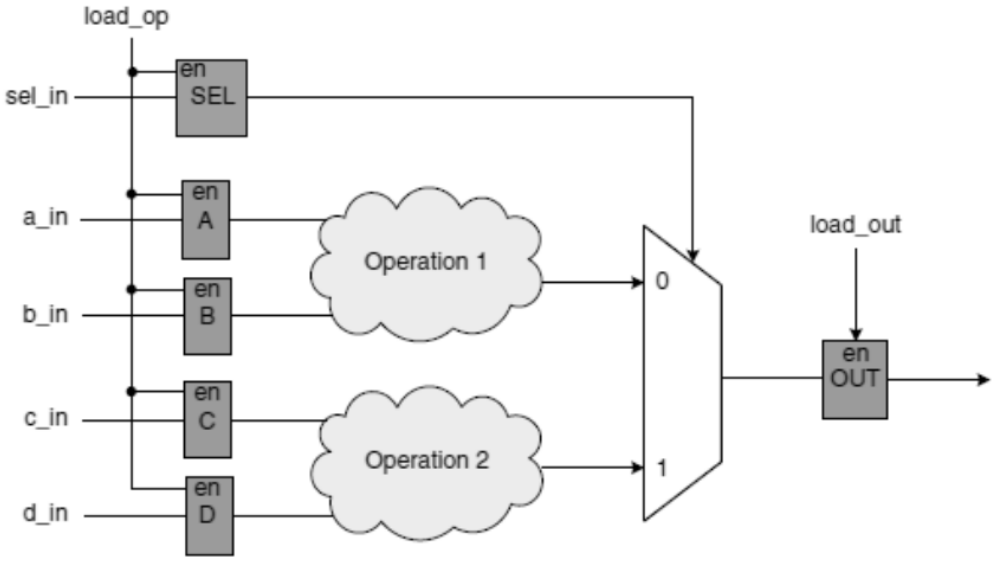

Figura 22 - Transições redundantes que consomem potência

A Figura 23 mostra a lógica modificada para conter as transições redundantes. “A” e “B” são carregados somente quando “SEL” é igual a “0”, enquanto “C” e “D” são carregados somente quando “SEL” é igual a “1”.

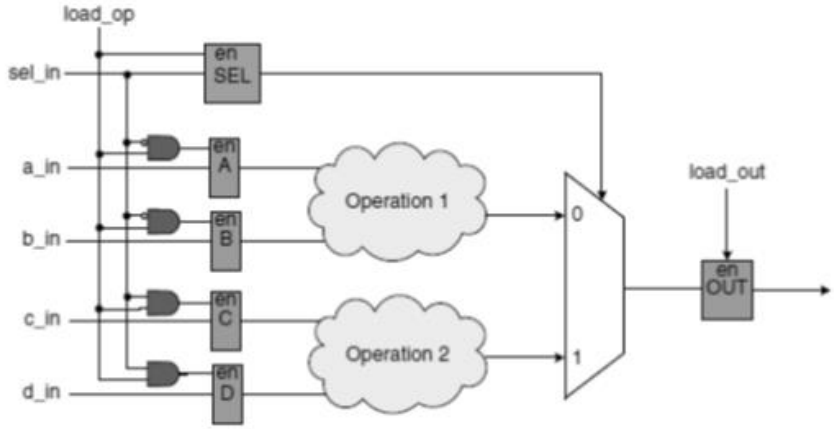

Figura 23 - Transições redundantes contidas para redução de potência

A Figura 24 mostra outro exemplo onde o mesmo “input_data” é ligado a todos os barramentos de dados de destino, onde “data_sel” indica o dado válido. Nesta implementação, somente um destino amostrará o dado de entrada, mas o barramento estará chaveando em todos os quatro ramos, consumindo energia desnecessária.

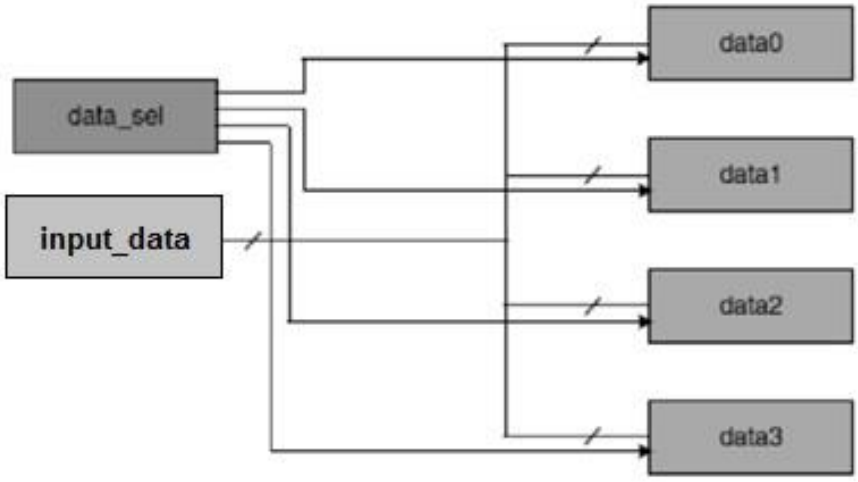

Figura 24 - Transições redundantes durante barramento ponto para multi-ponto

A Figura 25 mostra a mesma lógica modificada de forma a conter as transições redundantes. Existe um gasto de potência com a adição de portas lógicas ao negar a de-seleção do barramento, mas somente o ponto de destino que amostrará o dado eventualmente irá chavear, economizando energia.

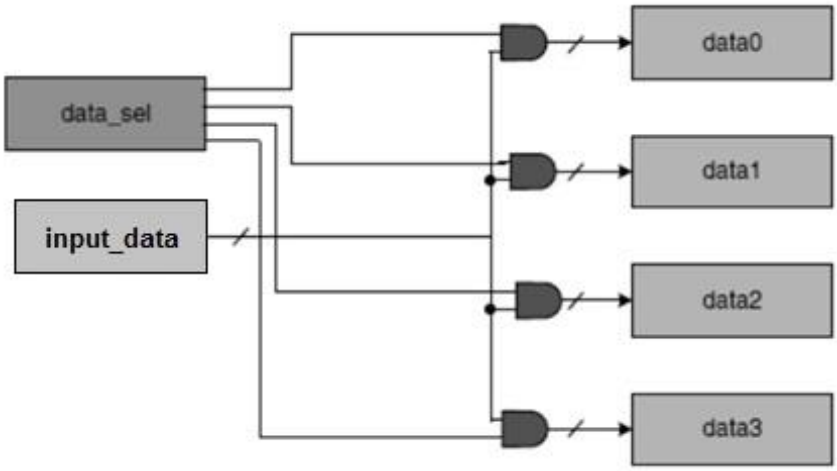

Figura 25 - Transações redundantes contidas em barramento ponto para multi-ponto

2.9.1.8 Memória Circular

Alguns circuitos digitais, como os filtros digitais, possuem a necessidade de realizar diversas operações com amostras de entrada, que devem ser armazenadas e deslocadas internamente para operações em domínio de tempo. Uma das formas mais utilizadas é através de implementação de registradores de deslocamento, cuja profundidade lógica depende da quantidade de iterações e amostras necessárias para serem processadas por operações aritméticas com coeficientes. A desvantagem do uso de registradores de deslocamento é sempre que uma amostra nova é disponibilizada na entrada, as demais amostras são deslocadas no mesmo ciclo de relógio do sistema, gerando uma grande quantidade de chaveamento de dados, aumentando o consumo de potência dinâmica. Uma proposta a este sistema para redução da dissipação de potência dinâmica, é a implementação de uma memória circular. Esta memória circular pode ser implementada através de uma matriz de registros e um circuito complementar que controla os ponteiros de leitura e escrita de uma amostra dentro dessa matriz. Com esse sistema, ao chegar uma amostra nova ao circuito, apenas o ponteiro de escrita é calculado e a escrita ocorre pontualmente numa posição dada pelo ponteiro de escrita, enquanto que no caso de uma leitura, outro ponteiro é calculado para o endereço do registro da matriz onde o dado armazenado é lido e usado para posterior cálculo.

A Figura 26 mostra um diagrama de uma memória circular em operação (CHU, 2008).

Na Figura 26(a), os ponteiros de leitura (rd ptr) e de escrita (wr ptr) começam na posição 0 da memória circular. Na figura (b), é feita a escrita de um dado na posição 0 e o ponteiro “wr ptr” é incrementado para a posição 1. A figura (c) mostra o ponteiro de escrita na posição 4 após terem sido feitas 3 escritas dentro da memória. No item (d), manteve-se o ponteiro de escrita na mesma posição e realizou-se uma leitura do dado armazenado na posição 0. Uma vez feita a leitura do conteúdo, o ponteiro de leitura “rd ptr” é deslocado para a posição 1. Em (e), a memória circular é atualizada sendo feitas 4 escritas, preenchendo a memória até a posição 7 e o ponteiro de escrita é posicionado na posição 0, onde continua vazia. Em (f), é feita a escrita na posição 0 e então, o ponteiro de escrita é deslocado para a posição 1 junto com o ponteiro de leitura. Nessa situação, a memória encontra-se cheia, não sendo permitido realizar uma nova escrita. Em (g), (h) e (i), são realizadas novas leituras, deslocando-se o ponteiro de leitura até que a memória esteja vazia e o ponteiro de leitura é posicionado em 1 junto com o ponteiro de escrita.

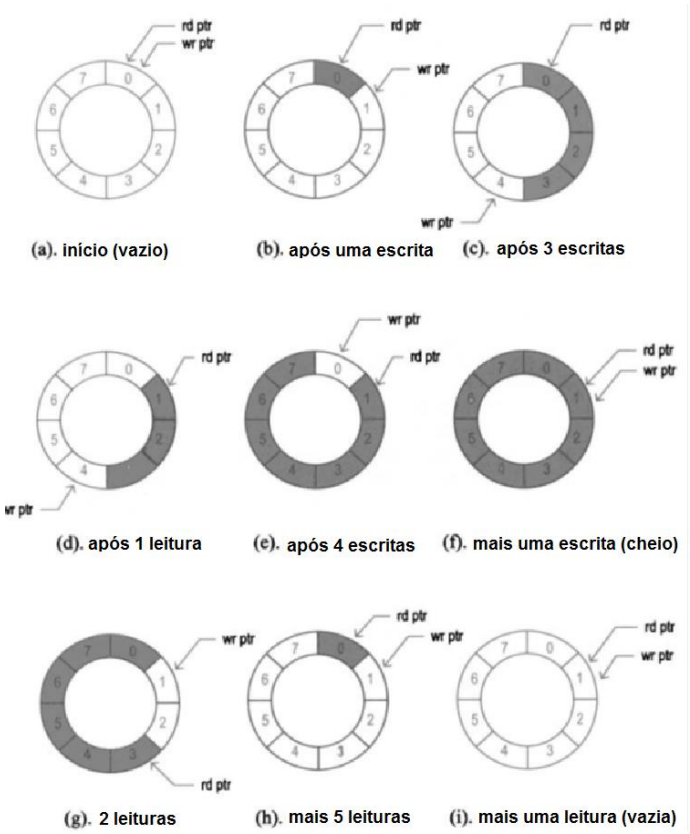

Figura 26 - Memória circular

2.9.2 Técnicas de codificação para otimização de área

O estilo de codificação na escrita de um código HDL pode afetar o desempenho do sistema a ser implementado tanto em FPGA como em ASIC. Entretanto, algumas regras e técnicas de codificação podem ser seguidas para se obter um hardware otimizado após a etapa de síntese lógica. A forma como o hardware é codificado pode influenciar nos resultados de síntese lógica, pois os algoritmos de síntese podem interpretar de formas diferentes e até interpretar de forma errada o que pretende-se implementar durante a descrição do hardware conforme estudo levantado por (SUHAILI e WATANABE, 2015).

Basicamente para redução de área de silício, é necessária a diminuição do número de portas lógicas, através de simplificações de expressões, minimização de estados em máquinas de estado finito (*Finite State Machine* (FSM)), e eliminação de lógica replicada. Relatam-se aqui duas técnicas que consistem basicamente na eliminação de portas desnecessárias para a lógica, mantendo-se o seu comportamento funcional original. Algumas das técnicas vistas para redução de dissipação de potência na seção anterior, contribuem também para redução de área.

2.9.2.1 Extração de MUX desnecessários

Na Figura 27 é possível verificar a eliminação de MUXES gerados desnecessariamente pela codificação RTL normal. Codificando-se de maneira a utilizar as primitivas lógicas é possível forçar a ferramenta de síntese lógica a substituir o multiplexador apenas por portas lógicas *and* e *or*.

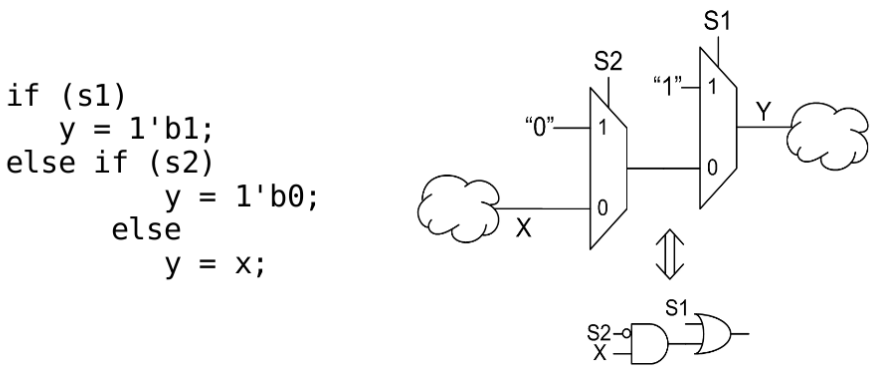

Figura 27 - Exemplo de extração de um MUX desnecessário

Esta técnica pode ser empregada em casos onde a codificação do MUX tem um dos sinais de entrada fixo. Na Figura 28 são apresentados alguns exemplos de equivalências de codificação, utilizando-se primitivas lógicas ao invés da codificação normal Verilog.

Esta técnica como se pode observar, pode produzir uma redução bastante grande de portas lógicas, ao substituir um muliplexador composto por várias portas lógicas, por apenas uma ou duas lógicas primitivas.

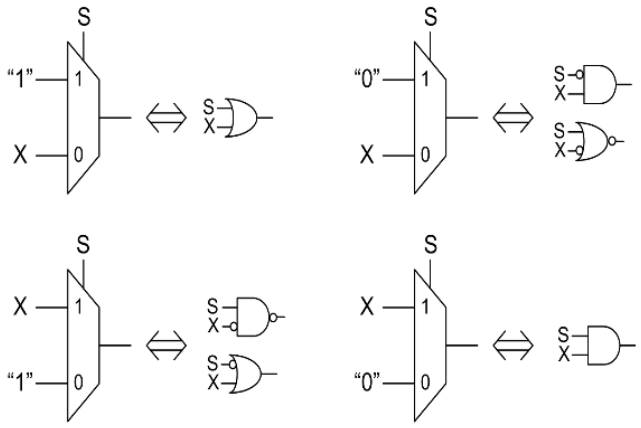

Figura 28 - Exemplos de equivalências utilizando-se primitivas lógicas para a codificação

2.9.2.2 Compartilhamento de recursos

Para projetos de circuitos que possuem muitas operações matemáticas, é necessário ter o cuidado de evitar duplicação de operações aritméticas onde o mesmo operando é utilizado em vários lugares. A Figura 29 mostra um exemplo de implementação onde não há compartilhamento de recursos, onde para cada condição do valor SEL, existe um operador diferente, tendo no total 6 operadores diferentes.

Essa lógica duplicada, além de consumir mais potência, aumenta o uso de área do circuito a ser sintetizado.

Já na Figura 30, é mostrada a mesma lógica modificada cobrindo todas as condições do “case” com apenas um comparador (“==”) e um comparador aritmético (“>”) com outro par de condições complementares.

```
always@(*)
  case (SEL)
    3'b000:  OUT = 1'b0;
    3'b001:  OUT = 1'b1;
    3'b010:  OUT = (value1 == value2);
    3'b011:  OUT = (value1 != value2);
    3'b100:  OUT = (value1 >= value2);
    3'b101:  OUT = (value1 <= value2);
    3'b110:  OUT = (value1 < value2);
    3'b111:  OUT = (value1 > value2);
  endcase
```

Figura 29 - Logica sem compartilhamento de recursos

```
assign cmp_equal = (value1 == value2);
assign cmp_greater = (value1 > value2);

always@(*)
  case (SEL)
    3'b000:  OUT = 1'b0;
    3'b001:  OUT = 1'b1;
    3'b010:  OUT = cmp_equal;                          // ==
    3'b011:  OUT = !cmp_equal;                         // !=
    3'b100:  OUT = (cmp_equal  || cmp_greater);        // >=
    3'b101:  OUT = !cmp_greater;                       // <=
    3'b110:  OUT = (!cmp_equal && !cmp_greater);       // <
    3'b111:  OUT = cmp_greater;                        // >
  endcase
```

Figura 30 - Lógica com compartilhamento de recursos

2.9.2.3 Simplificações Lógicas

A simplificação lógica de expressões na codificação RTL reduz o número de portas lógicas resultantes, o que garante uma menor área utilizada pelo projeto.

As complexidades do circuito lógico e da expressão lógica que o circuito representa estão diretamente ligadas. Embora a tabela da verdade que representa uma determinada função seja única, devido às diferentes possibilidades de simplificações

a serem utilizadas, a expressão lógica resultante pode ser escrita de diferentes formas.

A utilização da simplificação algébrica para minimização de funções lógicas não segue regras claras e sequenciais para a correta manipulação algébrica; assim, em muitas vezes, simplificações realizadas pelas ferramentas nem sempre são as mais otimizadas em determinados casos. Esta técnica é um procedimento fortemente dependente da experiência e do treinamento do projetista, que neste caso deve saber identificar e aplicar corretamente o processo de simplificação lógica no circuito.

Dentre alguns métodos de simplificação lógica, podem-se citar os postulados e teoremas como o de DeMorgan, métodos como o de Karnaugh e de Quine-McCluskey além das leis de associação e distribuição de termos e eliminação de lógica desnecessária (WEST e HARRIS, 2005), (VAHID, 2011). Para a correta aplicação destes métodos de simplificação e minimização lógica é necessário ao projetista conhecer individualmente cada um deles e saber de suas restrições de aplicação, bem como um possível aumento de atraso no circuito devido ao fato de se encadear um maior nível de lógica a expressão, mesmo diminuindo o número de portas lógicas empregadas.

2.9.3 Considerações a respeito das técnicas de codificação

As ferramentas de síntese automática utilizam otimização no seu processo de tradução, possuindo modelos subjacentes que podem ser bastante complexos. No entanto, o projetista não precisa dominar todos os conceitos associados a estes modelos para explorar os recursos da ferramenta, visto que seu papel é parametrizar a ferramenta para que esta gere descrições corretas. Ele apenas precisa dominar a influência desta parametrização no desempenho dos modelos de síntese. A aplicação destas técnicas à descrição HDL, nada mais é do que a parametrização mais rígida da ferramenta de síntese automática, visando que a mesma gere resultados mais otimizados.

Entretanto, existem algumas considerações que são importantes no desenvolvimento do hardware através da sua codificação em linguagem de hardware e que são essenciais ao se aplicar as técnicas discutidas:

- Bom estilo de codificação é importante:
    - Código fonte em HDL implica em estrutura inicial do circuito a ser desenvolvido;
    - Não conte com a ferramenta de síntese para “consertar” um projeto codificado sem cuidado;
    - A Qualidade dos Resultados (*Quality Of Results – QoR*) está relacionada diretamente à qualidade do código fonte (*Quality Of Source – QoS*);
- Pense em Hardware quando descreve um circuito utilizando HDL:
    - Nem toda estrutura HDL é sintetizável;
    - Mantenha a arquitetura do circuito a ser projetado em mente durante a descrição do projeto em HDL;
    - Circuitos que são simulados corretamente não são necessariamente sintetizados corretamente;
    - Escreva descrições funcionais dos blocos de hardware em HDL;
    - Não escrever modelos dos circuitos;
    - Descreva claramente a arquitetura do circuito;
- Separe Lógica Combinatória da Lógica Sequencial;
- Cobrir todos os possíveis estados quando se utiliza declarações de tomada de decisão (*if-else, case*);
- Menor código HDL não implica em menor área de Silício;
- Evitar o uso de laços do tipo *loop* para descrição de circuito.

3. IMPLEMENTAÇÃO DO CIRCUITO

3.1. Arquitetura implementada do NCO

O módulo NCO proposto para esta aplicação foi implementado utilizando o CORDIC como submódulo de Controle de Fase para Amplitude, utilizando apenas o modo rotacional, cobrindo o algoritmo apresentado na Seção 2.2.2 para o cálculo das funções seno e cosseno. O sistema foi implementado conforme apresentado em (CARVALHO, PALACIO e VAN NOIJE, 2016).

Para atender os requisitos do Gerador de sinais para o sistema EBE, o NCO foi implementado buscando atender as seguintes características:

- Tamanho da palavra de ângulo de entrada – 10 bits;
- Frequências dos sinais seno e cosseno – 100 Hz até 1 MHz;
- Tamanho da palavra de saída – 10 bits para cada função;
- Número de iterações: 10;
- Relógio principal do sistema: frequência de *clock* de 100 MHz;
- 5 bits de entrada para programação do módulo gerador de ângulos para determinar a frequência a ser gerada nas saídas (até 32 frequências a serem selecionadas);

O diagrama de blocos interno do módulo NCO é representado pela Figura 31.

A análise de aplicabilidade das técnicas de otimização levantadas na seção anterior foi feita durante a implementação de cada bloco do sistema, onde se adotou como metodologia de projeto o fluxo *middle-out*, onde primeiramente, implementou-se a arquitetura de topo e em seguida, foram projetadas as arquiteturas internas dos blocos. Dentre as técnicas estudadas, a única que foi possível de se implementar e que mostrou resultados significativos foi a técnica de compartilhamento de recursos em um dos módulos, que será melhor detalhada na descrição dos blocos.

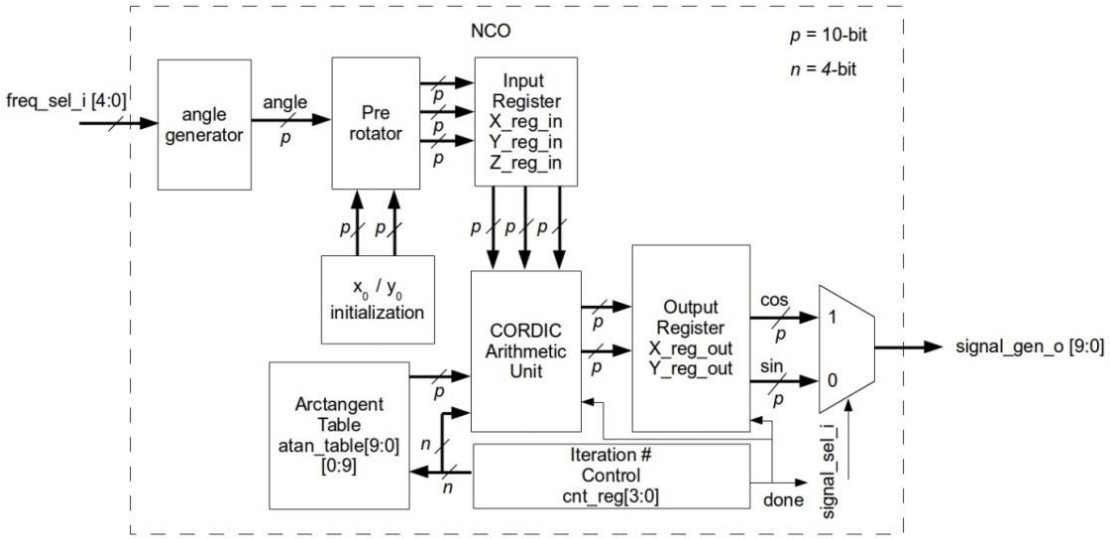

Figura 31 - Diagrama de blocos do circuito NCO

As especificações das entradas e saídas dos módulos são descritas na Tabela 1.

Tabela 1 - Interfaces do módulo CORDIC

Porta	Bits	E/S	Descrição
freq_sel_i	5 bits	Entrada	Entrada de palavra para configuração da frequência do sinal de saída de seno e coseno (seleção de 32 valores de ângulos)
clock	1	Entrada	Entrada de sinal de clock
reset_n	1	Entrada	Entrada de reset ativo nível baixo
signal	10	Saída	Saída 10 bits – seno ou cosseno
signal_sel_i	1	Entrada	Entrada de seleção do tipo de sinal a ser obtido na saída *signal* (0 – *signal* = *seno;* 1 – *signal* = *cosseno*)
done	1	Saída	Flag que indica que um dado novo está pronto nas saídas

A descrição da implementação dos blocos será abordada nas próximas subseções.

3.1.1 Gerador de ângulos - Angle Generator

O gerador de ângulos (*Angle Generator*) é um submódulo responsável por calcular os ângulos de entrada para a unidade aritmética do CORDIC, gerando ângulos de 0º a 360º em representação de 10 bits, fornecendo valores de ângulos de acordo com a palavra de programação definida pela entrada "*freq._sel*". O diagrama de blocos é representado na Figura 32.

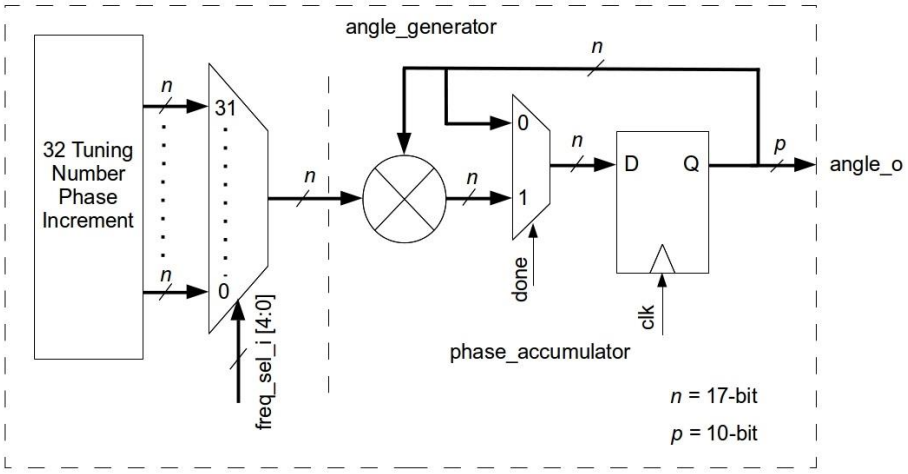

Figura 32 - Diagrama de blocos do circuito gerador de ângulos

Esse módulo é constituído de um sintonizador de fase de 10 bits, que contém 32 valores pré-definidos de incrementos de fase, calculados para fornecer o passo de incremento para o circuito seguinte que é o acumulador de fase, com o objetivo de fornecer a resolução dos valores de ângulos calculados pelo próprio circuito acumulador, gerando uma forma de onda triangular com valores definidos em módulo 2^n. Cada valor representa os valores de ângulos que vão de 0º à 360º, fornecidos a cada ciclo de relógio do sistema.

Para otimizar a dissipação de potência e o controle do acumulador de fase, um MUX foi implementado em RTL que utiliza o sinal de controle *done* fornecido pelo submódulo Controle de Iteração (*Iteration Controller*), para controlar o momento em que o acumulador de fase deve fornecer o próximo valor de ângulo para o submódulo seguinte que é o submódulo de pré-rotação (*pre-rotator*).

A eq.(10) mostra como foi determinado o valor de cada ângulo para sua respectiva representação em binário em 10 bits, onde $(Angle)_x$ é o tipo de representação binária/decimal.

$$(Angle)_2 = \frac{(Angle)_{10} * 2^{10}}{360} \qquad (10)$$

A distribuição dos valores de cada passo de incremento foi dimensionada para fornecer uma quantidade de amostras de ângulos dentro da faixa de 0º a 360º, para poder cobrir a faixa de variação da frequência de saída. Considerando que a

frequência principal do sistema foi definida em 100 MHz, o gerador de ângulos fornece um novo valor de ângulo para o submódulo *pre-rotator* a cada 10 ciclos do clock de sistema, ou seja, a uma taxa de 10 MHz, pelo fato de que cada valor de ângulo é liberado quando recebe o sinal de controle *done*. Este sinal é enviado a cada 10 ciclos do clock do sistema de acordo com o número de iterações que o CORDIC precisa para completar o cálculo dos valores de seno e cosseno para o valor de ângulo fornecido pelo submódulo *pre-rotator*.

Entretanto, para obter o menor passo de incremento do acumulador de fase (valor de incremento igual a 1), foi necessário expandir internamente o número de bits da representação do valor do passo e o barramento do acumulador de fase para 17 bits em ponto fixo, uma vez que 10 bits não seria o suficiente para representar esse valor, além do fato de evitar exceder a representação do número de bits (*overflow*). A representação binária dos valores de ângulo em 17 bits foi obtida através da eq. (11).

$$(Angle)_2 = \frac{(Angle)_{10} * 2^{17}}{360} \qquad (11)$$

O valor do ângulo 360º, por exemplo, é representado pelo valor em decimal $(131071)_{10}$. Considerando o fato de que um novo valor de ângulo é gerado a cada 10 MHz, para se obter um sinal senoidal na saída do NCO com frequência de 1 MHz, o gerador de ângulos deve fornecer 10 amostras por período para o *pre-rotator*. Nesta condição, o passo calculado é igual a $(13107)_{10}$. Utilizando a representação dos valores internamente em 17 bits, o menor passo $(1)_{10}$ fornece 131.071 amostras. Para essa amostragem, a frequência do sinal obtido na saída do NCO é de 76 Hz. A distribuição das 32 frequências dos sinais de saída do NCO foi definida calculando-se o fator de passo dividindo o máximo valor de passo por 32, obtendo-se desta forma o fator de passo igual à $(409)_{10}$.

O valor final de ângulo calculado pelo acumulador de fase com representação em 17 bits é então arredondado para 10 bits e então disponibilizado nesse formato para o *pre-rotator*.

A contagem de cada passo do sinal triangular obtido da saída do gerador de ângulos de 0º a 360º cobre uma rotação completa dentro do círculo trigonométrico, onde uma

rotação completa corresponde à um período do sinal senoidal obtido na saída do NCO.

A relação entre o incremento de fase com a saída gerada pelo NCO é representada pela Figura 33, onde A e B correspondem ao valor de ângulo do acumulador de fase, na figura à esquerda, e à direita, sua correspondência ao respectivo valor do sinal de saída do circuito na forma de amplitude.

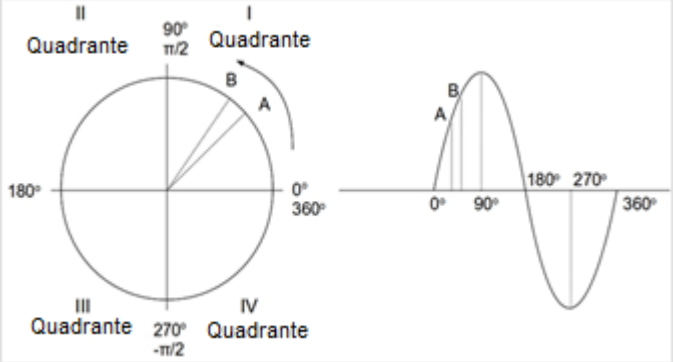

Figura 33 - Representação da fase do ângulo

3.1.2 Pré-rotação *(Pre-rotator)*

Este submódulo é responsável por verificar o valor do ângulo de entrada e definir a ordem dos valores de condição iniciais para X_0 e Y_0 e realizar a pré-rotação do ângulo de entrada que será carregado para a entrada da unidade aritmética do CORDIC (entrada Z_0). Esta operação é necessária devido à limitação do CORDIC em convergir apenas para ângulos dentro da faixa de 1º e 4º quadrantes da circunferência trigonométrica, entre -π/2 and π/2.

A determinação do quadrante é avaliada através dos dois bits mais significativos (MSB) da representação binária do valor do ângulo. Convertendo os valores de ângulo na faixa de 0º a 360º para representação binária de 10 bits, é possível identificar que os dois bits MSBs do valor do ângulo é o mesmo dentro de cada faixa de valores para seu quadrante específico, como pode ser visto na Tabela 2.

Tabela 2 - Representação binária dos quadrantes

Faixa de valores de ângulos (º)	Valor dois bits MSBs
0-89	00_XXXX_XXXX
90-179	01_XXXX_XXXX
180-269	10_XXXX_XXXX
270-359	11_XXXX_XXXX

Observando os valores da Tabela 2, quando os 2 bits MSBs são iguais à $(00)_2$ ou $(11)_2$, o valor do ângulo está no quadrante I que corresponde à faixa de valores de ângulos entre 0º e 89º ou no quadrante IV que corresponde à faixa de 270º a 359º, respectivamente. Para estas condições, o ângulo de entrada não é modificado, permanecendo o mesmo valor a ser usado para o CORDIC, sendo armazenado no registro *Z_reg_in* e transferido para a entrada Z_0, e o registrador denominado *X_reg_in* fornece o valor inicial igual a 0,60725 (constante K_n descrita na Seção 2.2.2) para a entrada X_0 do CORDIC e *Y_reg_in* fornece o valor inicial igual a 0 para a entrada Y_0 do CORDIC.

Quando os dois bits MSBs são iguais a $(01)_2$, significa que o ângulo calculado está dentro da faixa do quadrante II (90º a 179º). Nesta condição, o ângulo de entrada é pré-rotacionado para o seu valor correspondente ao primeiro quadrante e *X_reg_in* é iniciado com o valor 0 e *Y_reg_in* com a constante 0,607253.

Para os dois bits MSBs iguais a $(10)_2$, significa que o ângulo calculado está dentro da faixa do quadrante III (180º a 269º). Nesta condição, o ângulo de entrada é pré-rotacionado para o seu valor correspondente ao quarto quadrante e *X_reg_in* é iniciado com o valor 0 e *Y_reg_in* com a constante -0,607253.

O valor de constante 0,60725 é transformada para um número em representação de 10 bits utilizando a eq. (12).

$$X_{reg_{in}} = 0{,}60725 \times 2^{10} = (621{,}824)_{10} \qquad (12)$$

3.1.3 Módulo CORDIC (CORDIC Arithmetic Unit)

Este submódulo possui a estrutura que compõem os algoritmos discutidos na Seção 2.2.2, responsáveis pelo cálculo e geração dos valores de seno e cosseno, para um determinado ângulo de entrada e as condições iniciais.

A arquitetura implementada do CORDIC é a arquitetura Paralela Iterativa, onde se aplicou a técnica codificação de compartilhamento de recursos com o objetivo de ocupar a menor área e usar a menor quantidade de recursos de hardware, no qual se utiliza uma estrutura única de somadores e registradores de deslocamento e produz uma nova saída a cada N ciclos, onde N é definido pela resolução da palavra de ângulo. Como este sistema foi implementado com 10 bits, o CORDIC realizará 10 iterações, sendo que a cada 10 iterações, um valor de seno e de cosseno válidos será disponibilizado nas respectivas saídas. O diagrama de tempo é mostrado na Figura 34. Sinal RST_N mantém o circuito em estado de RESET quando está em nível lógico 0, colocando suas saídas em nível lógico 0 e o mantém inoperante, independente dos valores presentes nas entradas do circuito, até que este seja colocado e mantido em nível lógico 1. O sinal START inicia o ciclo de iteração do CORDIC, sendo gerado na forma de um pulso de pelo menos um ciclo de relógio, e carrega o dado In0 presente no barramento de entrada ANGLE e os dados de inicialização do CORDIC nas entradas x_0 e y_0 do módulo CORDIC. Normalmente, o próximo pulso do sinal START deve ser carregado após o final do ciclo do CORDIC atual, ou seja, após o pulso de sinal DONE ser detectado na saída do CORDIC. Caso o pulso de START seja acionado antes do término do ciclo de cálculo do algoritmo, uma nova conversão será realizada e irá descartar os resultados intermediários calculados neste momento.

Uma vez que o CORDIC termina o cálculo, este irá gerar um sinal de pulso DONE com duração de 1 ciclo de relógio CLK e os resultados de seno e coseno calculados serão disponibilizados nas saídas COS e SIN, representados por Out1. O próximo sinal de START é então gerado, e um novo dado de entrada In1 é carregado ao módulo CORDIC, iniciando um novo ciclo de cálculo. Cada ciclo de cálculo tem a duração de 10 ciclos de relógio, onde cada ciclo de relógio corresponde à uma iteração do algoritmo CORDIC.

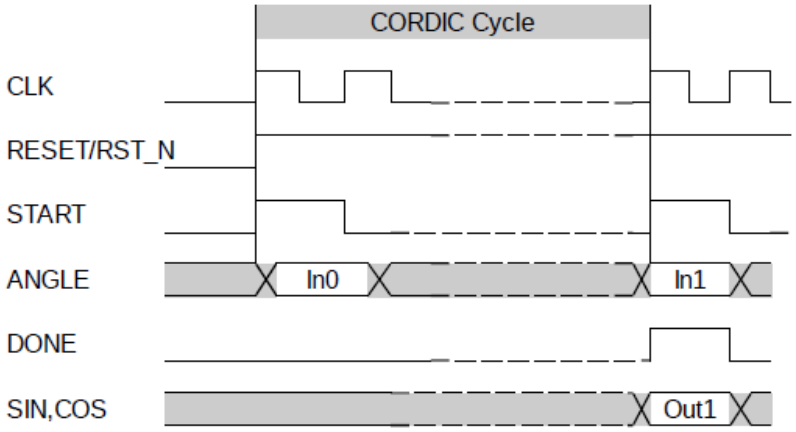

Figura 34 - Diagrama de tempo do CORDIC

A arquitetura interna do CORDIC implementado é representada pela Figura 35. Essa arquitetura corresponde às equações apresentadas na Seção 2.2.2.

O percurso do dado processado em cada estrutura de somador/subtrator até que tenha um valor calculado nas respectivas saídas de cada somador, demora um ciclo de relógio para ficar disponível. Cada ciclo corresponde a uma iteração.

Nessa estrutura apresentada, X representa os valores de cosseno, Y representa os valores de Seno e Z corresponde ao ângulo. Na Figura 35, a tabela de arco tangente mostrada fica separada do CORDIC e é discutida na próxima seção.

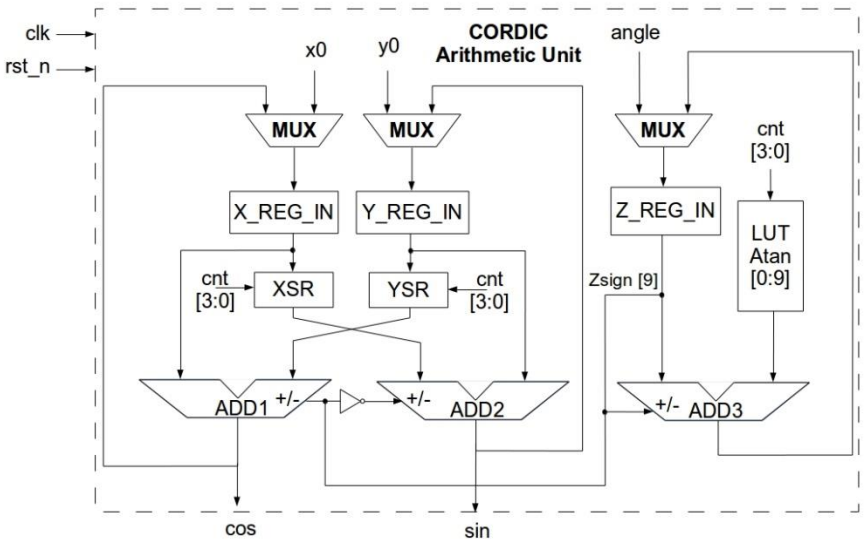

Figura 35 - Arquitetura da Unidade Aritmética do CORDIC

3.1.4 Tabela de Arco tangente - Arctangent Table

A tabela de arco tangente armazena valores pré-calculados do arco tangente de (2^{-i}) ($\tan^{-1} 2^{-i}$) para cada iteração. Os valores foram implementados numa matriz em RTL de profundidade igual a 10 e tamanho da palavra referente aos ângulos de 10 bits, correspondente ao número de iterações do sistema. Os valores implementados são mostrados na Tabela 3.

Tabela 3 - Valores de $\tan^{-1}$ calculados e convertidos para implementação da tabela de arco tangente em RTL

iteração i	2^{-i}	$\tan^{-1} 2^{-i}$ (radianos)	G = $\tan^{-1} 2^{-i}$ (graus)	N = 2^{10} * G / 360	Valor N (hexadecimal)
0	1,0000	0,7853	45,0000	128,0000	080
1	0,5000	0,4636	26,5650	75,5628	04B
2	0,2500	0,2449	14,0362	39,9253	027
3	0,1250	0,1243	7,1250	20,2667	014
4	0,0625	0,0624	3,5763	10,1726	00A
5	0,0312	0,0312	1,7899	5,0913	005
6	0,0156	0,0156	0,8951	2,5462	002
7	0,0078	0,0078	0,4476	1,2732	001
8	0,0039	0,0039	0,2238	0,6366	000
9	0,0019	0,0019	0,1119	0,3183	000

3.1.5 Iteration Controller

Submódulo responsável por calcular o ponteiro de endereço da tabela de arco tangente para leitura do valor armazenado numa determinada posição feita pela unidade aritmética para realizar o calculo do ângulo. O número de iterações a ser gerado por este módulo são 10 iterações. Após realizar a última iteração, este bloco gera um sinal de controle *"done"* que sinaliza que o ciclo de iterações terminou e que o dado de saída de seno e cosseno está disponível. Este sinal de controle é mapeado para um pino de saída para que possa ser usado como sinal de clock para o circuito analógico DAC para sincronização da entrada do DAC com a saída do NCO, amostrando o dado gerado pelo NCO no tempo correto em que a amostra do sinal de saída está pronta.

3.1.6 Multiplexador de saída

Implementou-se um multiplexador na saída de sinal do circuito NCO para ter controle de escolha do tipo de sinal a ser observado na saída *signal*. Este MUX é controlado pelo sinal de entrada *signal._sel_i*. Quando está em nível lógico '0', o sinal a ser

observado na saída é o sinal senoidal. Quando é colocado o nível lógico ‘1’, o sinal a ser observado na saída é o cosseno.

Este MUX foi implementado devido à limitação da quantidade de pinos do encapsulamento do chip fabricado (precisariam de pelo menos 20 pinos para poder disponibilizar as saídas de seno e cosseno, sendo 10 pinos para cada uma).

3.2. Implementação do circuito NCO

O circuito NCO descrito na Seção 3.1 foi implementado em RTL utilizando a linguagem Verilog. Durante a implementação dos circuitos, foi feita uma análise de quais técnicas de codificação que foram estudadas poderiam ser aplicadas de forma efetiva para melhorar o processo de redução de área. A técnica que mostrou resultados diretos foi a de compartilhamento de recursos que foi aplicada diretamente ao submódulo CORDIC, transformando-o em um sistema iterativo.

O circuito foi validado através de simulações digitais, por meio de códigos Verilog que implementam um *testbench*, contendo os componentes necessários para gerar os estímulos para as entradas do circuito durante as simulações.

O circuito foi sintetizado utilizando a metodologia de fluxo de projeto digital para ASIC, utilizando o processo 180 nm da fabricante TSMC. O motivo da escolha desse processo foi devido a submissão e aprovação desse projeto junto a outros circuitos no programa Free Mini@sic oferecido pelo IMEC que permitiu a fabricação gratuita do chip em MPW. A síntese lógica foi feita através da ferramenta de síntese da Cadence, o Encounter RTL Compiler (RC 14.2). Como resultado da síntese, obteve-se um arquivo Verilog contendo a descrição do circuito em *gate level*. Esse circuito foi verificado novamente através de simulações e os resultados foram comparados com os resultados de simulação do circuito em nível RTL.

Nessa etapa de síntese, implementou-se um NCO utilizando um circuito CORDIC de arquitetura paralela desenvolvida (*unrolled*) de 10 estágios de iteração. O circuito foi sintetizado no mesmo processo e, com os resultados obtidos da síntese, foi possível fazer as comparações de área e de dissipação de potência em relação ao circuito NCO otimizado.

O leiaute do circuito foi feito utilizando a ferramenta da Cadence, o Encounter Digital Implementation (EDI 14.2) utilizando o *design kit* do processo 180 nm da TSMC, seguindo o fluxo de projeto automatizado, de forma que o processo de posicionamento e roteamento do circuito são feitos guiados pela ferramenta de forma automática. No leiaute foram utilizados 5 camadas de metal, sendo que as camadas mais altas (4 e 5) foram utilizadas para roteamento de trilhas de alimentação e de aterramento. O leiaute é mostrado na Figura 36.

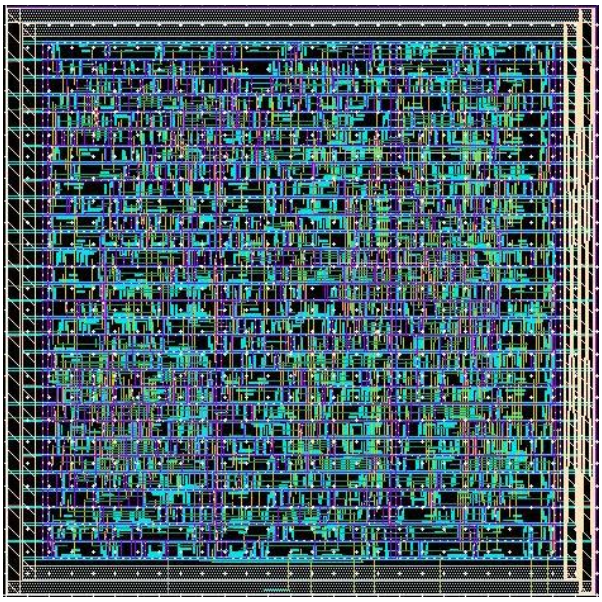

Figura 36 - Layout do NCO implementado

Uma vez tendo o leiaute do circuito NCO pronto, gerou-se o GDSII do circuito e o mesmo foi importado para ferramenta da Cadence, o IC 6.1.5 Virtuoso, onde foram feitas as simulações com um circuito analógico de um conversor DA de 10 bits implementado com componentes ideais para avaliação funcional do circuito pós-leiaute, utilizando a ferramenta ADE da Cadence, integrada com o Virtuoso.

Dentro do ambiente da ferramenta Virtuoso, foram feitas as verificações físicas de DRC e LVS para confirmar a integridade do circuito implementado.

O circuito NCO também foi sintetizado em FPGA, para avaliação funcional e verificação da quantidade de recursos utilizados. A FPGA utilizada foi a Cyclone III modelo EP3C5E144C8 do fabricante Altera, utilizando o software Quartus II versão 11.0.

Nesta mesma FPGA, implementou-se um NCO utilizando um circuito CORDIC de arquitetura paralela desenvolvida (*unrolled*) de 10 estágios de iteração. Com isso, foi possível fazer a comparação do uso de recursos de cada arquitetura e avaliar os resultados de consumo de recursos de elementos lógicos que compõem a FPGA (as chamadas *Logic Cells* (LC)).

A avaliação do sinal de saída do circuito NCO implementado em FPGA foi feita utilizando um conversor DA modelo AD5433YRU do fabricante Analog Devices e um

filtro passa-baixas. Ambos os circuitos fazem parte da placa FPGA utilizada nos testes. Para efetuar o controle da seleção de frequências de saída e do tipo de sinal de observação, as entradas de controle foram mapeadas para um conjunto de chaves montadas na placa da FPGA.

O circuito NCO foi integrado ao topo do chip e os pinos de entrada e saída foram roteados para os respectivos PADs, onde é feito a ligação entre o pino do encapsulamento com o pino do circuito. A Figura 37 mostra no lado esquerdo, a imagem do GDSII do chip com a delimitação da área aonde foi posicionado o circuito NCO, e à direita, a foto do chip que foi tirada depois de feita a fabricação. Na imagem, destaca-se a região aonde se encontra o circuito NCO.

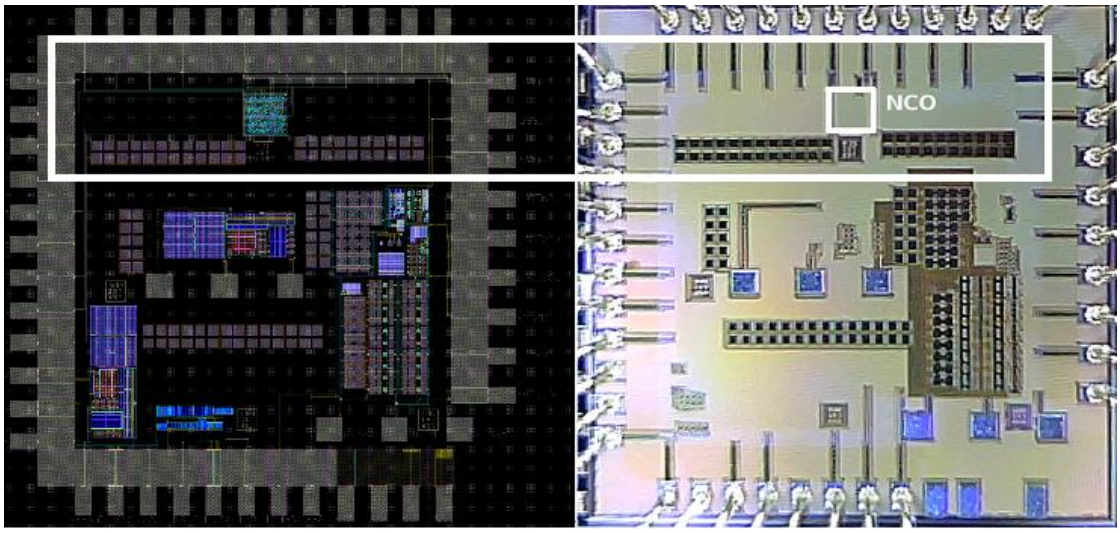

Figura 37 - Imagem do GDS (esquerda) e foto do chip fabricado (direita)

O chip contendo o circuito NCO fabricado foi encapsulado em um padrão QFN48 e montado em uma placa de testes que foi desenvolvida pelo grupo de pesquisa, para avaliação e controle dos circuitos que foram fabricados no chip. A placa de testes é mostrada na Figura 38.

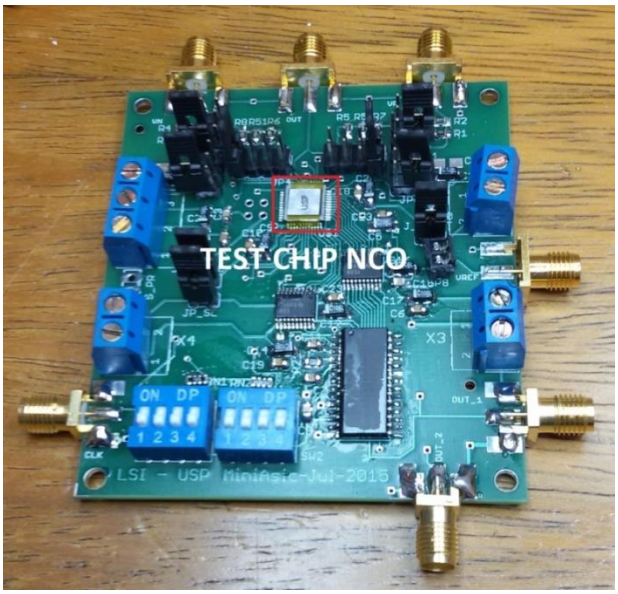

Figura 38 - Foto da placa de avaliação para testes do chip fabricado

Montou-se nesta placa de testes um conector para inserção de sinal de *clock* externo, utilizando-se um equipamento gerador de sinais para que fosse possível controlar o *clock* principal do circuito NCO. As saídas de sinal e de controle (*done*) do NCO foram ligadas às respectivas entradas de dados e de *clock* de amostragem de um CI DAC de 10 bits modelo THS5651 da Texas Instruments, que foi montado na placa de testes para realizar a conversão do sinal gerado pelo NCO.

As entradas de seleção de frequência foram ligadas à um conjunto de chaves do tipo DIP switches, utilizadas para selecionar a frequência do sinal de saída do NCO. O procedimento de testes consistiu em configurar a palavra de seleção da frequência e em seguida, media-se o sinal de saída convertido pelo CI DAC, utilizando o osciloscópio Infiniium 54852A da Agilent. O sinal de reset e a escolha do sinal de saída do NCO também eram definidos através das chaves DIP switch.

O chip é alimentado com tensão de 1.8V fornecida por uma fonte externa e as medidas de consumo foram feitas por meio de pontos de observação montados na placa de testes.

3.3. Resultados Experimentais

Nesta seção serão analisados os resultados obtidos de área, consumo e desempenho do NCO implementado. Serão feitas comparações de área e consumo

entre o NCO utilizando o CORDIC otimizado em arquitetura iterativa e o NCO utilizando CORDIC *unrolled*.

A seguir, na Tabela 4, são apresentados os resultados de área e dissipação de potência total entre as duas arquiteturas do circuito NCO, extraídos em nível de portas lógicas após a etapa de síntese física.

Tabela 4 - Resultados de área e dissipação de potência totais obtidos entre duas arquiteturas do NCO extraídos em nível de portas lógicas (pós síntese física)

Submódulo	NCO (CORDIC Iterativo)		NCO (CORDIC *unrolled*)	
	Àrea (mm^2)	Potência (mW)	Àrea (mm^2)	Potência (mW)
CORDIC	0,0040	2,62	0,0730	18,56
Angle Generator	0,0050	0,58	0,0004	0,58
Módulos de controle / Tabela ATan	0,0080	0,06	0,0046	0,12
Total	**0,0170**	**3,26**	**0,0780**	**19,26**

A potência dissipada para cada uma das versões do circuito foi extraída através da ferramenta Encounter Power System da Cadence, tendo como cenário onde os circuitos foram sintetizados utilizando os modelos de células para o melhor caso (tensão de operação igual a 1,98 V, temperatura 0ºC) e utilizando arquivos gerados pela ferramenta de simulação funcional, que fornecem informações de chaveamento dos sinais nas trilhas de interconexão de cada circuito. As informações de dissipação de potência apresentadas na Tabela 4 foram obtidas para uma simulação funcional onde ambos os circuitos geravam um sinal senoidal de 1 MHz.

Através dos resultados apresentados pela Tabela 4, a melhoria de área utilizando a técnica de codificação de compartilhamento de recursos aplicada diretamente ao submódulo CORDIC foi significativa, permitindo uma redução de área de aproximadamente 78% em relação à área ocupada pelo CORDIC *unrolled*.

A área dos módulos de controle somada à área da tabela de arco tangente foi maior na versão com o módulo iterativo devido à adição de lógica de controle para o CORDIC iterativo, uma vez que é necessário o acompanhamento de cada iteração do algoritmo. O mesmo ocorreu para o módulo *Angle Generator*.

Apesar do acréscimo de área dos módulos de controle, foi possível obter uma redução da dissipação de potência total significativa. Houve uma redução de

aproximadamente 83% de dissipação de potência total em relação a potência dissipada pelo circuito NCO que utiliza o CORDIC *unrolled*.

Observou-se nos resultados de análise de dissipação de potência que o valor de dissipação de potência total é praticamente dado pela potência dinâmica estimada. A potência estática observada é menor que 1% do valor da potência dinâmica, sendo possível desprezá-la dessa análise. Com isso, percebe-se que com a mudança da arquitetura do circuito através da codificação do sistema, é possível obter melhorias significativas na redução da dissipação de potência dinâmica do circuito.

A potência dissipada no chip fabricado foi medida durante os testes de bancada e resultado obtido foi um consumo de 2,7 mW no modo de operação em que a frequência do sinal de saída foi selecionado para 1 MHz, e para a frequência de saída de 76 Hz, foi medido um consumo de 1,6 mW.

A Tabela 5 mostra a comparação entre os valores de dissipação de potência medida no chip de testes e a dissipação de potência extraída em nível de portas lógicas, obtida pela ferramenta Encounter Power System. Ambos os valores foram obtidos para a mesmas condições de operação funcional (geração de sinal senoidal de frequência de 1 MHz) e tensão de alimentação 1,98 V.

Tabela 5 – Comparação de resultados de dissipação de potência real versus extraído em nível de portas lógicas

Potência (mW) NCO (chip de testes)	Potência (mW) NCO (extraído em nível de portas lógicas)
2,7	**3,26**

Existe uma consideração importante nesta comparação com relação a um dos parâmetros que é a temperatura de operação. A medida da dissipação de potência no chip de testes foi feita em temperatura ambiente, enquanto que a medida da dissipação de potência feita através da extração do valor feita pela ferramenta de análise de consumo foi obtida utilizando a biblioteca de células com caracterização e

informações de consumo para o cenário conhecido como *Best Case*, onde a tensão de alimentação é de 1,98 V e a temperatura é de 0ºC.

A comparação dos resultados obtidos na implementação em FPGA, em termos de números de elementos lógicos combinatórios e sequenciais são mostrados na Tabela 6.

Tabela 6 - Análise comparativa de recursos lógicos obtida na implementação em FPGA das duas arquiteturas

Tipo de Células Lógicas (*Logic Cells* (LC))	**NCO (CORDIC Iterativo)**	**NCO (CORDIC *unrolled*)**
Combinatório	164	383
Sequencial	54	256
Total LCs	**218**	**639**

Através dos resultados apresentados na Tabela 6, é possível visualizar a redução do uso de células lógicas e recursos utilizados em FPGA, confirmando a eficiência da codificação do sistema para obterem-se melhorias de área do circuito implementado.

A Tabela 7 apresenta a comparação dos resultados obtidos na implementação do NCO deste trabalho com demais trabalhos levantados nas referências bibliográficas, de acordo com estudo feito em (CARVALHO, PALACIO e VAN NOIJE, 2016).

Tabela 7 - Comparação dos resultados de área e consumo do circuito fabricado com trabalhos de referência

Projeto	*Área (mm^2)*	*Clock (MHz)*	*Potência dissipada (mW/MHz)*	*Arquitetura*	*Processo (nm)*
Este trabalho	0,017	100	0,027	CORDIC Iterative Bit-Parallel	180
(HUANG, LEE e WANG, 2014)	0,332	227	0,350	ROM-less Parabolic Polynomial Interpolation	130
(LANGLOIS e AL-KHALILI, 2003)	0,282	100	0,234	Optimized linear interpolation	350

Os sinais de saída seno e cosseno do circuito NCO testado foram medidos após a saída do CI DAC montado na placa de testes. Foram feitas 32 medidas para avaliar a qualidade e a frequência obtida dos sinais gerados pelo circuito fabricado. A menor frequência obtida foi de 76 Hz e a maior frequência obtida foi de 1 MHz.

A Tabela 8 mostra os valores teóricos de frequências obtidos com base nos cálculos dos valores dos passos gerados pelo módulo *Angle Generator*, descrito na Seção 3.1.1 e os valores medidos nos testes do circuito em bancada.

Tabela 8 – Frequência do sinal de saída do NCO

Palavra de seleção de frequência (freq_sel_i [4:0])	Frequência do sinal de saída calculado (Hz)	Frequência do sinal de saída medido (Hz)
00000	76,29	76,30
00001	31.280,52	31.284,06
00010	62.484,74	62.489,57
00011	93.688,97	93.692,67
00100	124.893,19	124.908,15
00101	156.097,41	156.115,41
00110	187.301,63	187.329,49
00111	218.505,86	218.524,91
01000	249.710,08	249.728,71
01001	280.914,31	280.952,81
01010	312.118,53	312.147,64
01011	343.322,75	343.356,54
01100	374.526,98	374.577,27
01101	405.731,20	405.793,73
01110	436.935,43	437.287,28
01111	468.139,65	468.171,12
10000	499.343,87	499.343,80
10001	530.548,10	530.650,20
10010	561.752,32	561.778,60
10011	592.956,54	592.942,75
10100	624.160,77	624.225,76
10101	655.364,99	655.443,10
10110	686.569,22	686.555,16
10111	717.773,44	717.941,66
11000	748.977,66	749.107,29
11001	780.181,89	780.220,07
11010	811.386,11	811.517,12
11011	842.590,33	842.665,87
11100	873.794,56	873.866,62
11101	904.998,78	905.102,73
11110	936.203,01	930.991,03
11111	999.984,74	1.000.093,90

A comparação entre o valor calculado e o valor medido das frequências foi levantada no gráfico da Figura 39.

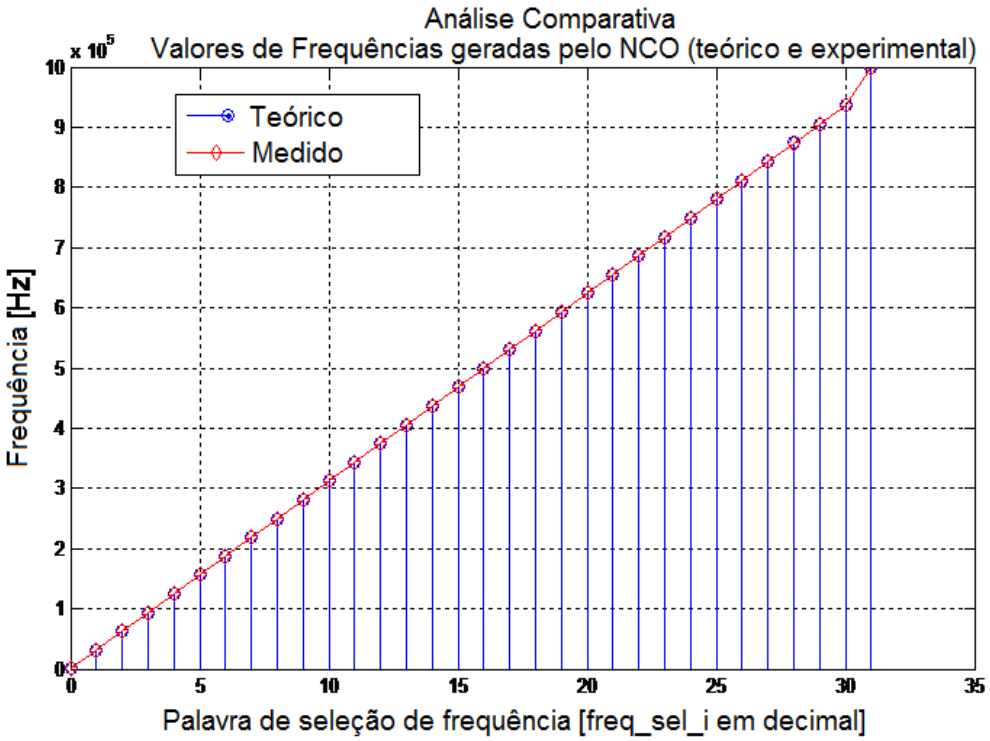

Figura 39 - Análise comparativa entre os valores de frequência calculados e medidos

Através dos valores de frequência calculados e medidos apresentados, calculou-se a variação entre ambos os valores, tendo como resultado uma variação de 0,006% entre os resultados, sendo possível observar a qualidade e a precisão do circuito implementado. Observou-se que a faixa de frequências coberta pelo circuito ficou centralizada em frequências do sinal de saída na banda de kHz, devido à distribuição do passo que foi implementada dentro da tabela no submódulo *Angle Generator*. O passo entre a primeira frequência para a palavra de seleção “00000” e a seleção seguinte “00001” ficou relativamente alto, fazendo com que a primeira frequência fosse 76 Hz e a segunda frequência fosse 31,28 kHz.

Para que o circuito possa cobrir uma maior variação de frequências, será necessário aumentar a tabela de passos do circuito *Angle Generator* e rever o passo entre os valores de incremento, de forma que a variação de frequências não fique desproporcional. A outra abordagem que deveria ser feita e que pode ser implementada em uma nova versão seria de recalcular a distribuição dos 32 valores de frequências em função do sinal de saída e não em função dos valores de incremento, como foi feito nessa abordagem. Essas alterações, no entanto não afetarão a área e a dissipação de potência do circuito implementado.

Os sinais senoidais de saída do chip testado para as frequências de 76 Hz, 156,11 kHz e 1 MHz, obtidos através de medidas feitas com o osciloscópio são apresentados nas Figuras 40, 41 e 42.

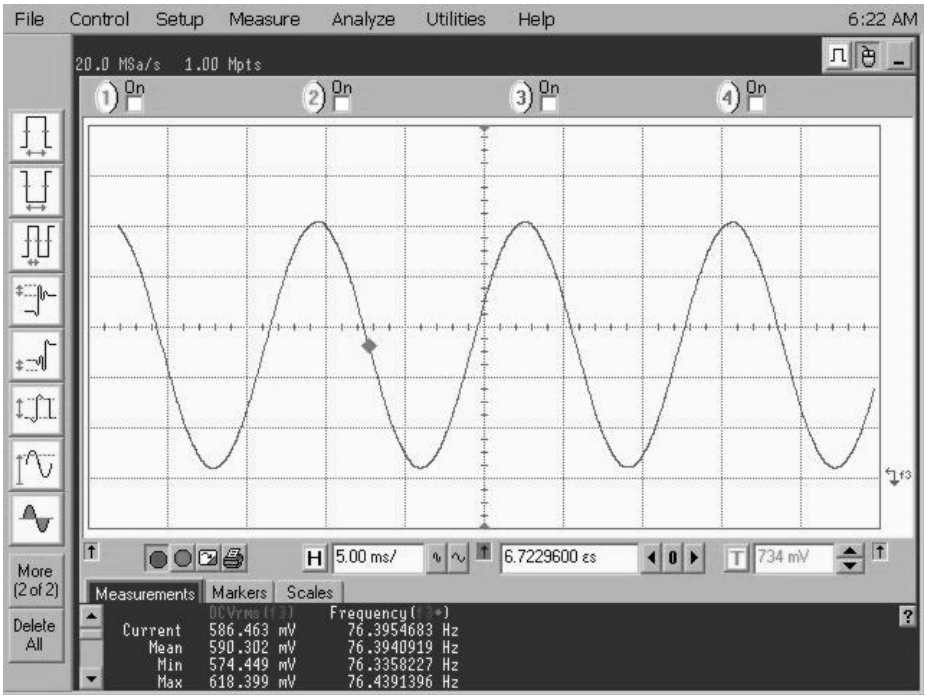

Figura 40 - Medida de saída do NCO de testes para frequência de saída 76 Hz.

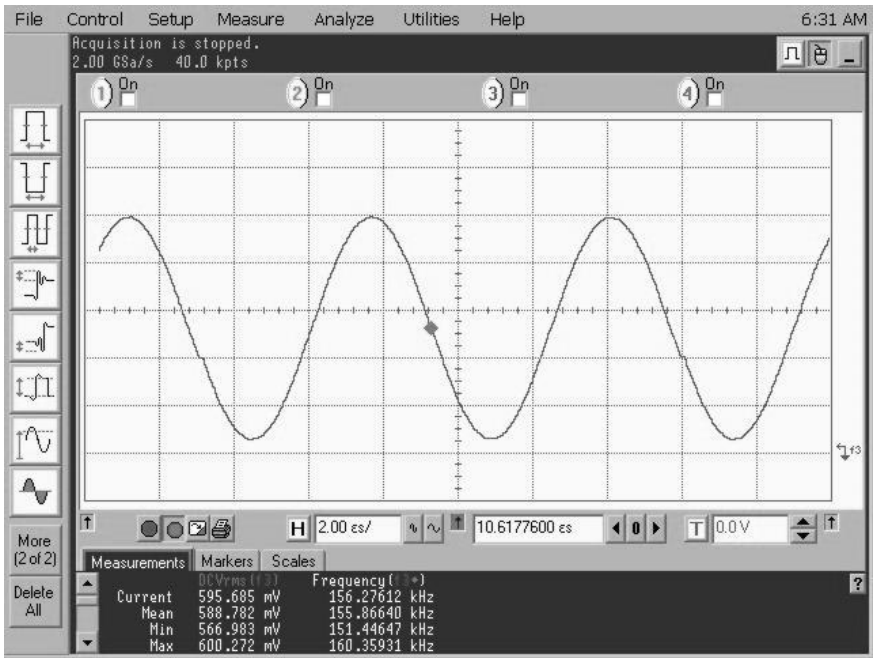

Figura 41 - Medida da saída do NCO de testes para frequência de 156,11 kHz

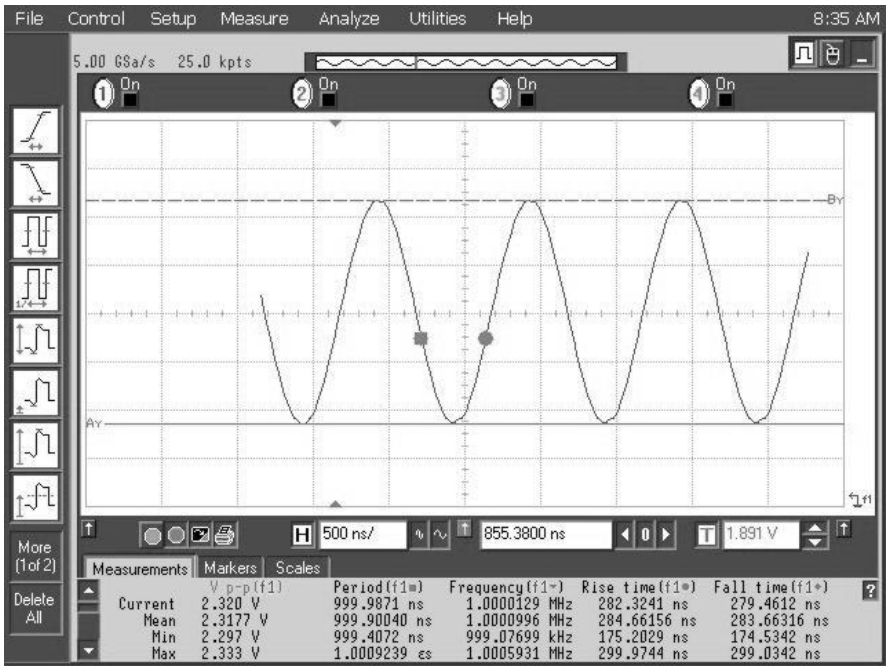

Figura 42 - Medida da saída do NCO de testes para frequência de 1 MHz

A Figura 43 mostra o gráfico obtido da medida feita para o sinal senoidal de saída de 156,11 kHz no domínio da frequência através de análise FFT do sinal. O desempenho do sinal gerado pelo circuito através do valor medido de Range Dinâmico Livre de Espúrios (*Spurious Free Dynamic Range* (SFDR)) foi medido para todas as frequências e o valor obtido foi de 52,47 dB.

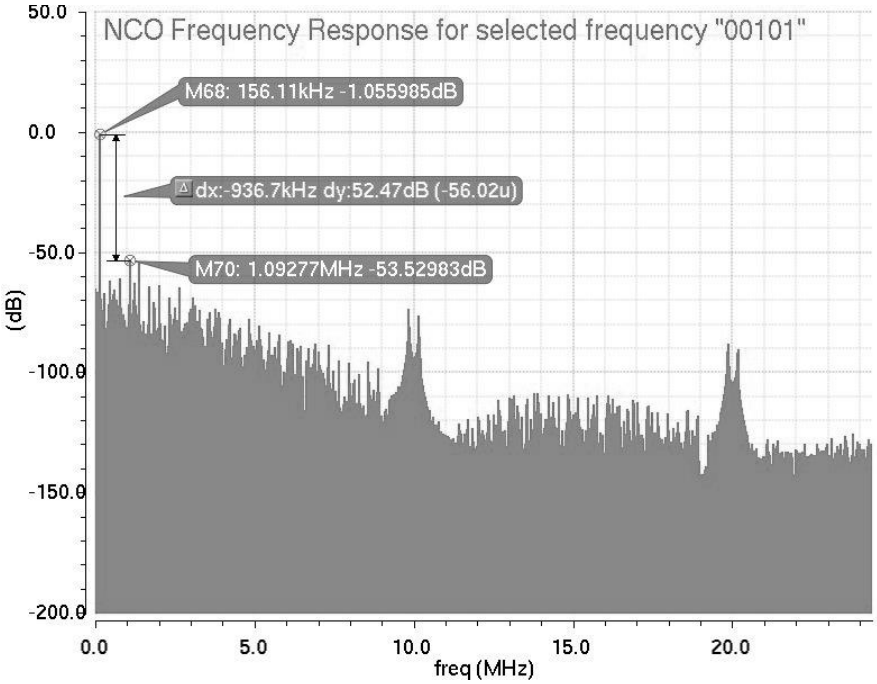

Figura 43 - Medida SFDR da saída do circuito

Observou-se que o desempenho do circuito em termos de SFDR, pode ser melhorado incluindo um filtro digital na saída do NCO, ou através do filtro analógico a ser implementado após o circuito DAC. Como o circuito trabalha com uma resolução de 10 bits, o SFDR é menor comparado com um NCO com resolução de bits maior. Entretanto aumentando o número de bits da saída do circuito NCO, implica em aumentar a resolução do DAC, aumentando a complexidade do circuito analógico e comprometendo a área do sistema EBE.

4. CONCLUSÕES

Dos fatos apresentados nas pesquisas do estudo bibliográfico e nos resultados e discussões levantados, conclui-se que:

- É possível implementar otimizações de descrição de hardware em um nível de abstração maior, durante a etapa de codificação do circuito em RTL, de forma a se obter melhorias dos resultados de síntese lógica, tanto para a redução de área de silício a ser utilizada (relacionada com o número de portas e elementos base) como para a potência dissipada, através de técnicas apresentadas na literatura e artigos que mostraram resultados positivos com o uso dessas técnicas;
- Pode-se observar que, além das técnicas de otimização, outro fator que pode afetar o resultado da síntese lógica é a qualidade do código descrito em HDL, sendo necessário o uso de boas práticas de descrição do sistema;
- Algumas técnicas de otimização de potência podem afetar a área a ser implementada, exigindo circuitos que ocupem uma área maior para compensar a redução de dissipação de potência, por necessitar de uma lógica adicional para controle, dependendo da técnica a ser aplicada;
- Através dos resultados obtidos, pode-se comprovar o funcionamento do circuito NCO atendendo as especificações do sistema EBE, e com a aplicação de uma das técnicas de codificação apresentadas (compartilhamento de recursos), foi possível obter um circuito de área otimizada, tendo uma redução de 78% de área e com menor dissipação de potência, sendo esta otimizada em 83% em relação ao circuito NCO de referência utilizando o submódulo CORDIC com arquitetura *unrolled*. Os resultados obtidos, comparados com outros trabalhos, foram positivos, demonstrando que as técnicas de codificação podem influenciar nos resultados de área e de dissipação de potência;
- A qualidade do sinal de saída foi analisada em termos da relação sinal-ruído no domínio da frequência, tendo como resultado um sinal com valor de 52,47 dB em termos de SFDR. Isso se deve ao fato da resolução do

sinal de saída ser relativamente baixo, 10 bits, tendo o erro de quantização como o principal fator que contribui para a degradação do sinal. Outro fator que contribui para o aumento de espúrios na saída é o truncamento de fase que é feito no módulo *Angle Generator*, reduzindo de 17 bits para 10 bits a representação da fase. A redução de espúrios pode ser feita filtrando o sinal gerado na saída do NCO.

4.1. Considerações Futuras

Faz-se uma sugestão para melhoria da qualidade do sinal gerado pelo NCO: a implementação de um filtro digital passa-baixa na saída do circuito, antes do DAC. Outra opção discutida em (SHRIMALI, 2007), é a implementação de um circuito chamado *dither*, que gera números randômicos, para inserir ruído branco nos bits menos significativos. Após o circuito *dither*, pode ser feito o truncamento do sinal de saída para que seja compatível com o número de bits de entrada do DAC. Com isso, a resolução do sistema NCO pode ser de 16 bits ou até 24 bits. Essa melhora terá um custo de área que precisará ser avaliado.

Para que o circuito possa cobrir uma maior variação de frequências, será necessário aumentar a tabela de passos do circuito *Angle Generator* e rever o passo entre os valores de incremento, de forma que a variação de frequências não fique desproporcional. A outra abordagem que pode ser feita e implementada em uma nova versão seria de recalcular a distribuição dos 32 valores de frequências em função do sinal de saída e não em função dos valores de incremento, como foi feito nessa abordagem.

Outra possibilidade de melhorar a varredura de frequências seria implementar uma interface serial onde seja possível programar diretamente o passo de incremento no acumulador de fase, dentro do circuito *Angle Generator*, aumentando a faixa de valores de incremento em 2^N.

Este estudo pode ser ampliado utilizando-se outras linguagens de descrição de *hardware*, como VHDL e também linguagens com maior nível de abstração, como o SystemC, por exemplo, explorando as possibilidades de otimização do circuito em uma camada comportamental acima da descrição em RTL, utilizando Modelagem

em Nível de Transações (*Transaction Level Model* (TLM)) e ferramentas de síntese de alto nível (*High Level Synthesis* (HLS)).

REFERÊNCIAS BIBLIOGRÁFICAS

ABDUL, S. et al. The use of electrical impedance spectroscopy in the detection of cervical intraepithelial neoplasia. **International Journal of Gynaecological Cancer**, [S.l.], v. 16, p. 1823-1832, 2006.

ARORA, M. **The Art of Hardware Architecture:** Design Methods and Techniques for Digital Circuits. New York: Springer, 2012.

BARROW, A. J.; WU, S. M. Impedance measurements for cervical cancer diagnosis. **Gynecologic Oncology**, v. 107, n. 1, Supplement, p. S40--S43, #oct# 2007. ISSN ISSN: 0090-8258. Disponivel em: <http://www.sciencedirect.com/science/article/pii/S009082580700501X>.

BHATNAGAR, H. **Advanced ASIC Chip Synthesis**. 2ª. ed. New York: Kluwer Academic Publishers, 2002.

BROWN, B. H. et al. Detection of cervical intraepithelial neoplasia using impedance spectroscopy: a prospective study. **An International Journal of Obstetrics and Gynaecology**, Sheffield, v. 112, p. 802-806, 2005.

CADENCE. Cadence Design Systems. **Encounter RTL Compiler**, 2015. Disponivel em: <http://www.cadence.com/eu/Pages/rtl_compiler.aspx>. Acesso em: Maio 2016.

CARVALHO, P. R. B.; PALACIO, J. A. A.; VAN NOIJE, W. A. M. Area optimized CORDIC-based Numerically Controlled Oscillator for Electrical Bioimpedance Spectroscopy. **2016 IEEE International Frequency Control Symposium Proceedings**, New Orleans, Agosto 2016. 83-88.

CHU, P. P. **FPGA Prototyping by Verilog Examples**. New Jersey: Wiley, 2008.

CHURIWALA, S.; GARG, S. **Principles of VLSI RTL Design:** A Practical Guide. New York: Springer, 2011.

CUMMINGS, C. E. Nonblocking Assignments in Verilog Synthesis, Coding Styles That Kill! **SNUG**, San Jose, 2000.

DAHAN, N. Adventures in ASIC Digital Design. **Low Power Techniques - Reducing Switching**, 2007. Disponivel em:

<https://asicdigitaldesign.wordpress.com/2007/06/15/low-power-techniques-reducing-switching/>. Acesso em: Maio 2016.

DAS, L.; DAS, S.; CHATTERJEE, J. Electrical Bioimpedance Analysis: A New Method in Cervical Cancer Screening. **Journal of Medical Engineering**, [S.l.], 2015.

DURAISAMY, K.; JAGANATHAN, K. S.; BOSE, J. C. Methods of Detecting Cervical Cancer. **Advances in Biological Research 5**, Hyderabad, 2011. 226-232.

EMNETT, F.; BIEGEL, M. Power Reduction Through RTL Clock Gating. **SNUG**, San Jose, 2000.

FERREIRA, E. C. Aula 23 - DDS. **Site do Departamento de Eletrônica e Microeletrônica**, 2016. Disponivel em: <http://www.demic.fee.unicamp.br/~elnatan/ee610/23a%20Aula.pdf>. Acesso em: Fevereiro 2016.

HUANG, J. M.; LEE, C. C.; WANG, C. C. A ROM-less Direct Digital Frequency Synthesizer Based on the 16-Segment Parabolic Polynomial Interpolation. **The Scientific World Journal**, [S.l.], v. 2014, 2014.

IARC. IARC Monographs on the evaluation of carconogenic risks to humans. **Human Papilomaviruses**, v. 90, 2007.

IARC. IARC Monographs: A review of human carcinogens - Part B: biological agents. **Lancet Oncol.**, v. 10, p. 321-322, 2009.

INCA. Instituto Nacional de Câncer. **Colo do Útero**, 2016. Disponivel em: <http://www2.inca.gov.br/wps/wcm/connect/tiposdecancer/site/home/colo_utero/definicao>. Acesso em: Maio 2016.

IN-GI, L.; WHAN-WOO, K. A Numerically Controlled Oscillator with a Fine Phase Tuner and Rounding Processor. **ETRI**, December 2004.

KADAM, S. et al. **Comparison of various numerically controlled oscillators**. Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on. [S.l.]: [s.n.]. 2002. p. III-200--III-202 vol.3.

KASSANOS, P.; TRIANTIS, F. I.; DEMOSTHENOUS, A. A CMOS Magnitude/Phase Measurement Chip for Impedance Spectroscopy. **IEEE Sensors Journal**, [S.l.], v. 13, n. 6, 2013.

KITCHENER, H. C.; CASTLE, P. E.; COX, J. T. Chapter 7: Achevements and limitations of cervical citology screening. In: _____ **Vaccine**. [S.l.]: [s.n.], v. Supplement 3, 2006. p. S63-S60.

KUMAR, N.; SAPPAL, A. S. Coordinate Rotation Digital Computer Algorithm: Design and Architectures. **International Journal of Advanced Computer Science and Applications**, [S.l.], v. 2, n. 4, p. 68-71, 2011.

LANGLOIS, J. M. P.; AL-KHALILI, D. Novel approach to the design of Direct Digital Frequency Synthesizers based on linear interpolation. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, v. 50, n. 9, p. 567-578, 2003.

MIRANDA, D. A. **Detección Precoz de Cancer de Cuello Uterino basada en Espectro de Impedancia Eléctrica**. Bucaramanga: Disertación Tesis de Maestría, 2005.

MIRANDA, D. A.; CORZO, S. P.; CORREA, C. A. G. Cervical Cancer Detection by Electrical Impedance in a Colombian setting. **Journal of Physics: Conference Series 434**, [S.l.], 2013.

MIRANDA, D.; BARRERO, J.; ECHEVERRI, J. **Estudio para la detección temprana de Cáncer de Cuello Uterino.** 1. ed. Bucaramanga: Sic - Proyecto Cultural de Sistemas y Computadores S.A., v. No. 1, 2007.

NEWS MEDICAL. News Medical Life Sciences & Medicine. **Health:** What is Spectroscopy, 2015. Disponivel em: <http://www.news-medical.net/health/What-is-Spectroscopy.aspx>. Acesso em: 2016.

PALACIO, J. A. A.; VAN NOIJE, W. A. M. High Stability Voltage Controlled Current Source For Cervical Cancer Detection using Electrical Impedance Spectroscopy. **VI IEEE Latin American Symposium On Circuits and Systems (LASCAS 2015)**, Montevideo, 2015. 205-208.

PALNITKAR, S. **Verilog HDL:** A Guide to Digital Design and Synthesis. 2ª. ed. New Jersey: Prentice Hall, 2003.

RAIRIGH, D. et al. **Sinusoid signal generator for on-chip impedance spectroscopy**. Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on. [S.l.]: [s.n.]. 2009. p. 1961-1964.

SARAVANAN, P.; RAMASANY, S. Sine/Cos generator for Direct Digital Frequency Synthesizer using Pipelined CORDIC processor. **4th International Conference on Computing, Communication and Networking Technologies (ICCCNT)**, [S.l.], p. 1-6, Julho 2013.

SEOANE, F.; BRAGOS, R.; LINDECRANTZ, K. **Current Source for Multifrequency Broadband Electrical Bioimpedance Spectroscopy Systems. A Novel Approach**. Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE. [S.l.]: [s.n.]. 2006. p. 5121-5125.

SHRIMALI, S. **Direct Digital Frequency Synthesizer**. Dissertação Tese de Mestrado. ed. Lubbock: Texas Tech University, 2007.

SILVERIO, A.; SILVERIO, A. A high output impedance current source for wideband bioimpedance spectroscopy using 0.35 um TSMC CMOS Technologgy. **International Journal of Engineering and Applied Sciences**, dez. 2012.

STAN, M. R.; BURLESON, W. P. Bus-invert coding for low power I/O. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 3, n. 1, p. 49-58, March 1995. ISSN 1063-8210.

SUHAILI, S. B.; WATANABE, T. Performance Evaluation of Verilog Coding Techniques for Optimized MD5 Algorithm. **Journal of Industrial Engineering Research**, [S.l.], 2015. 8-14.

VACHHANI, L.; SRIDHARAN, K.; MEHER, P. K. Efficient CORDIC Algorithms and Architectures for Low Area and High Throughput Implementation. **IEEE Transactions on Circuits and Systems II: Express Briefs**, [S.l.], Jan. 2009. 61-65.

VAHID, F. **Digital Design with RTL Design, VHDL and Verilog**. 2ª. ed. [S.l.]: John Wiley and Sons, 2011.

VAHID, F.; GIVARGIS, T. **Embedded System Design:** A Unified Hardware/Software Introduction. [S.l.]: John Wiley & Sons, 2002.

VOLDER, J. E. The CORDIC Trigonometric Computing Technique. **IRE Transactions on Electronic Computers**, [S.l.], Sept. 1959. 330-334.

WALKER, D. C. **Modelling the Electrical Properties of Cervical Epithelium**. University of Sheffield. Sheffield, p. 266. 2001.

WEST, N. H. E.; HARRIS, D. **CMOS VLSI Design:** A circuits and systems perspective. 3ª. ed. San Francisco, CA: Pearson, 2005.

WHO. World Health Organization. **The Wold´s Health Organization figth against Cancer:** Strategies That Prevent, Cure and Care., 2007.

ZHANG, W. VHDL Tutorial: Learn by example. **Embedded System Design:** A Unified Hardware/Software Introduction, 2001. Disponivel em: <http://esd.cs.ucr.edu/labs/tutorial/>. Acesso em: 2016.

APÊNDICE A – Código Verilog do Oscilador Controlado Numericamente e submódulos

```
//////////////////////////////////////////////
// Company: University of São Paulo - EPUSP
// Engineer: Paulo Roberto Bueno de Carvalho
// Design Name: NCO
// Module Name: nco
// Project Name: NCO
// Target Devices: ASIC, FPGA Xilinx Artix-7, Altera
// Description: Signal Generator 10-bit top level
// for Sine and Cosines generation with frequency programmable
//Dependencies:cordic_iterative.v,angle_gen.v
// Revision:1.0
//////////////////////////////////////////////
module nco(
    freq_sel_i,
    signal_sel_i,
    clk,
    rst_n,
    done_o,
    signal_gen_o
    );
//----PARAMETERS DECLARATION--------
parameter FSEL_WIDTH = 5;
parameter WORD_WIDTH = 10;
//---------INPUTS/OUTPUTS----------
    input signed [FSEL_WIDTH-1:0] freq_sel_i;
    input signal_sel_i;
    input clk;
    input rst_n;
    output       signed       [WORD_WIDTH-1:0] signal_gen_o;
    output done_o;
//-REGISTERS AND WIRES DEFINITIONS-
wire signed [WORD_WIDTH-1:0] sin_w;
wire signed [WORD_WIDTH-1:0] cos_w;
wire signed [WORD_WIDTH-1:0] angle_w;
wire done_w;
//--------INSTANTIATION MODULES--------------
cordic_iterative cordic_iterative(
    .angle(angle_w),
    .clk(clk),
    .rst_n(rst_n),
    .done (done_w),
    .cos(cos_w),
    .sin(sin_w)
                            );
angle_gen_5bit angle_gen_5bit(
    .angle_o (angle_w),
    .clk (clk),
    .rst_n (rst_n),
    .done (done_w),
    .freq_sel (freq_sel_i)
    );
//------OUTPUT ASSIGNMENTS-------------------
assign done_o = done_w;
assign signal_gen_o = signal_sel_i ? cos_w : sin_w;

endmodule
```

```
//////////////////////////////////////////////
// Company: University of São Paulo - EPUSP
// Engineer: Paulo Roberto Bueno de Carvalho
// Create Date: 07.03.2015 12:59:28
// Design Name: Angle Generator
// Module Name: angle_gen
// Project Name: NCO
// Target Devices: ASIC, FPGA Xilinx Artix-7, Altera
//Description:10-BIT      Angle      Generator programmable for CORDIC
// Revision:1.0
//////////////////////////////////////////////
module angle_gen_5bit(
    angle_o,
    clk,
    rst_n,
    done,
    freq_sel
    );
//----------PARAMETERS DECLARATION-----------
parameter Z_WIDTH = 10;
//--------------INPUTS/OUTPUTS---------------
    input clk;
    input rst_n;
    input done;
    input [4:0] freq_sel;
    output signed [Z_WIDTH-1:0] angle_o;
//-------REGISTERS AND WIRES DEFINITIONS-----
    reg signed [16:0] phase_acc_reg;
    reg signed [15:0] freq_acc_reg;
    reg [4:0] freq_sel_reg;
//----------ANGLE GENERATOR CONTROL----------
   always@(posedge clk or negedge rst_n)
   begin
    if (!rst_n)
    begin
        phase_acc_reg <= 17'd0;
        freq_sel_reg <= 5'd0;
    end
    else
    begin
       freq_sel_reg <= freq_sel;
       if (done)
       begin
         if (phase_acc_reg < 17'h1FFFF)
         begin
           phase_acc_reg  <=  phase_acc_reg  + freq_acc_reg;
         end
         else
         begin
           phase_acc_reg <= 17'd0;
         end
       end
       else
       begin
         phase_acc_reg <= phase_acc_reg;
       end
    end
   end
//-----------------------------------------
(continua)
```

```
(continuação)
//-----------------------------------------
always@(*)
begin
case (freq_sel_reg)
  5'd0 : freq_acc_reg = 16'h0001;
  5'd1 : freq_acc_reg = 16'h019A;
  5'd2 : freq_acc_reg = 16'h0333;
  5'd3 : freq_acc_reg = 16'h04CC;
  5'd4 : freq_acc_reg = 16'h0665;
  5'd5 : freq_acc_reg = 16'h07FE;
  5'd6 : freq_acc_reg = 16'h0997;
  5'd7 : freq_acc_reg = 16'h0B30;
  5'd8 : freq_acc_reg = 16'h0CC9;
  5'd9 : freq_acc_reg = 16'h0E62;
  5'd10 : freq_acc_reg = 16'h0FFB;
  5'd11 : freq_acc_reg = 16'h1194;
  5'd12 : freq_acc_reg = 16'h132D;
  5'd13 : freq_acc_reg = 16'h14C6;
  5'd14 : freq_acc_reg = 16'h165F;
  5'd15 : freq_acc_reg = 16'h17F8;
  5'd16 : freq_acc_reg = 16'h1991;
  5'd17 : freq_acc_reg = 16'h1B2A;
  5'd18 : freq_acc_reg = 16'h1CC3;
  5'd19 : freq_acc_reg = 16'h1E5C;
  5'd20 : freq_acc_reg = 16'h1FF5;
  5'd21 : freq_acc_reg = 16'h218E;
  5'd22 : freq_acc_reg = 16'h2327;
  5'd23 : freq_acc_reg = 16'h24C0;
  5'd24 : freq_acc_reg = 16'h2659;
  5'd25 : freq_acc_reg = 16'h27F2;
  5'd26 : freq_acc_reg = 16'h298B;
  5'd27 : freq_acc_reg = 16'h2B24;
  5'd28 : freq_acc_reg = 16'h2CBD;
  5'd29 : freq_acc_reg = 16'h2E56;
  5'd30 : freq_acc_reg = 16'h2FEF;
  5'd31 : freq_acc_reg = 16'h3333;
  default: freq_acc_reg = 16'h0001;
endcase
end
//-------------OUTPUT ASSIGNMENTS------------
assign angle_o = phase_acc_reg[16:7];

endmodule
```

```
//////////////////////////////////////////////
// Company: University of São Paulo - EPUSP
// Engineer: Paulo Roberto Bueno de Carvalho
// Create Date: 07.03.2015 12:59:28
// Design Name: CORDIC Iterative 10-BIT
// Module Name: cordic_iterative
// Project Name: Signal Generator
// Target Devices: ASIC, FPGA Xilinx Artix-7,
Altera
// Tool Versions:
//   Description:   10-BIT   CORDIC   Iterative
architecture in rotation mode
// for Sine and Cosines generation
// Revision:1.0
//////////////////////////////////////////////
module cordic_iterative(
    angle,
    clk,
    rst_n,
    done,
    cos,
    sin
    );
//-------PARAMETERS DECLARATION------
parameter XY_WIDTH = 10;
parameter Z_WIDTH = 10;
localparam ITERATION_NUMBER = 4;
localparam ATAN_WIDTH = Z_WIDTH;
localparam ATAN_LENGTH = 10;
//-----------INPUTS/OUTPUTS--------------
    input signed [Z_WIDTH-1:0] angle;
    input clk;
    input rst_n;
    output signed [XY_WIDTH-1:0] cos;
    output signed [XY_WIDTH-1:0] sin;
    output done;
//------REGISTERS AND WIRES DEFINITIONS------
    reg signed [XY_WIDTH-1:0] X_reg_in;
    reg signed [XY_WIDTH-1:0] X_reg;
    reg signed [XY_WIDTH-1:0] Y_reg_in;
    reg signed [XY_WIDTH-1:0] Y_reg;
    reg signed [Z_WIDTH-1:0] Z_reg_in;
    reg signed [Z_WIDTH-1:0] Z_reg_out;
    reg [ITERATION_NUMBER-1:0] cnt_reg;
    reg signed [XY_WIDTH-1:0] X_reg_out;
    reg signed [XY_WIDTH-1:0] Y_reg_out;
    wire  signed  [ATAN_WIDTH-1:0]  atan_table
[0:ATAN_LENGTH-1];
    wire signed [XY_WIDTH-1:0] Xsr;
    wire signed [XY_WIDTH-1:0] Ysr;
    wire Zsign;
    wire [1:0] quadrant;
//---------ARC TAN TABLE DEFINITION----------
assign  atan_table[0]  =  10'h080;  //angle  45
degrees
assign   atan_table[1]   =   10'h04B;   //angle
26.56505 degrees
assign   atan_table[2]   =   10'h027;   //angle
14,03624 degrees
assign   atan_table[3]   =   10'h014;   //angle
7,12501 degrees
assign   atan_table[4]   =   10'h00A;   //angle
3,57633 degrees
assign   atan_table[5]   =   10'h005;   //angle
1,789910 degrees
assign   atan_table[6]   =   10'h002;   //angle
0,895173 degrees
(continua)
```

```
(continuação)
assign   atan_table[7]   =   10'h001;   //angle
0,447614 degrees
   assign  atan_table[8]  =  10'h000;  //angle
0,2238105 degrees
   assign  atan_table[9]  =  10'h000;  //angle
0,1119056 degrees
//----SIGN  SHIFT  RIGHT  AND  SIGNAL  ANGLE
ASSIGNMENT-----------------------------------
    assign Xsr = X_reg_in >>> cnt_reg;
    assign Ysr = Y_reg_in >>> cnt_reg;
    assign Zsign = Z_reg_in[9];
    assign quadrant = angle[9:8];

//---------ITERATION POINTER CONTROL---------
   always@(posedge clk or negedge rst_n)
   begin
    if (!rst_n)
    begin
        cnt_reg <= 4'd0;
    end
    else
    begin
       if (cnt_reg<9)
              begin
              cnt_reg <= cnt_reg+1;
              end
       else
              begin
              cnt_reg <= 4'd0;
              end
    end
   end
   assign  done  =  (cnt_reg==4'd9)  ?  1'b1  :
1'b0;
//---------------------------------------------
//-----INITIALIZATION OF X AND Y / LOAD
CALCULATED VALUES
//-----PRE-ROTATE INPUT ANGLES TO STAY FROM -
PI/2 TO PI/2 RANGE-------
always@(*)
  begin
    if (cnt_reg==4'd0)
    begin
       case(quadrant)
         //QUADRANTS 0 - PI/2 AND -PI/2 - 0
         2'b00,2'b11: begin
           X_reg_in = 308; // Constant
0.60725
           Y_reg_in = 0;
           Z_reg_in = angle;
         end
         //QUADRANTS PI/2 - PI
         2'b01: begin
           X_reg_in = 0;
           Y_reg_in = 308; // Constant
0.60725
           Z_reg_in = {2'b00,angle[7:0]};
         end
         //QUADRANTS PI - 3*PI/2
         2'b10: begin
           X_reg_in = 0;
          Y_reg_in = -308; // Constant
0.60725
          Z_reg_in = {2'b11,angle[7:0]};
         end
       endcase
    end
 (continua)
```

```
(continuação)
    else
    begin
       X_reg_in = X_reg;
       Y_reg_in = Y_reg;
       Z_reg_in = Z_reg_out;
    end
  end

//--------ITERATION SIN COS CALCULATION------
   always@(posedge clk or negedge rst_n)
   begin
    if (!rst_n)
    begin
        X_reg <= 0;
        Y_reg <= 0;
        X_reg_out <= 0;
        Y_reg_out <= 0;
        Z_reg_out <= 0;
    end
    else
    begin
       X_reg  <=  Zsign  ?  X_reg_in  +  Ysr  :
X_reg_in - Ysr;
       Y_reg  <=  Zsign  ?  Y_reg_in  -  Xsr  :
Y_reg_in + Xsr;
       Z_reg_out  <=  Zsign  ?  Z_reg_in  +
atan_table[cnt_reg]     :     Z_reg_in     -
atan_table[cnt_reg] ;
       X_reg_out  <=  (done)  ?  X_reg  :
X_reg_out;
       Y_reg_out  <=  (done)  ?  Y_reg  :
Y_reg_out;
    end
   end
//-----------OUTPUT ASSIGNMENTS--------------
   assign cos = X_reg_out;
   assign sin = Y_reg_out;

endmodule
```