TRABALHO DE GRADUAÇÃO

MODELAGEM EM ALTO NÍVEL DA SEÇÃO DE RECEPÇÃO DE UM TRANSCEPTOR RF

Breno Nascimento

Brasília, agosto de 2010

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

MODELAGEM EM ALTO NÍVEL DA SEÇÃO DE RECEPÇÃO DE UM TRANSCEPTOR RF

Breno Nascimento

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof. José Camargo da Costa, ENE/UnB _____
Orientador

Prof. Carlos Humberto Llanos Quintero, ENM/UnB _____
Examinador externo

Mestre Gilmar Silva Beserra, ENE/UnB _____
Examinador interno

Agradecimentos

Gostaria de agradecer primeiramente minha família que sempre me apoiou nessa minha difícil jornada, e em especial à minha mãe que sempre esteve ao meu lado. Agradeço também meu falecido pai que sempre manterá sua imagem e presença de espírito junto a mim dando forças aos desafios que se passaram e que irão surgir.
Sou grato pelo meu orientador José Camargo da Costa por apoio e dedicação durante o trabalho, assim como a equipe do LDCI que me auxiliaram no desenvolvimento do projeto e estavam sempre dispostos a ajudar.
Agradeço também a todos meus amigos nos momentos de descontração e companhia nas noites de estudo.

Breno Nascimento

RESUMO

O projeto descrito neste documento apresenta a proposta de uma descrição em alto nível na linguagem Verilog-AMS da seção de recepção de um transceptor RF de um sistema integrado CMOS desenvolvido no Laboratório de Dispositivos e Circuitos Integrados (LDCI) do departamento de Engenharia Elétrica da Universidade de Brasília - UnB. A modelagem do sistema completo possibilita a co-simulação entre os diversos componentes, permitindo a análise do comportamento do sistema e identificar as características e funcionalidades do mesmo. Com isso é possível otimizar a arquitetura em projetos futuros.

ABSTRACT

The project described in this document presents the proposal of a high level description in Verilog-AMS language of a RF transceiver reception section of a CMOS integrated system developed at the Integrated Circuit and Devices Laboratory (LDCI - Laboratório de Dispositivos e Circuitos Integrados) of the Electrical Engineering Department at University of Brasília - UnB. The complete system model enables the co-simulation among the various components, allowing the analysis of the behavior of the system and identify its features and functionality. With this approach, it is possible to optmize the system architecture in future projects.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

K	Constante de Boltzmann	[J/K]
T	Temperatura Absoluta da Resistência	[K]

Siglas

ABNT	Associação Brasileira de Normas Técnicas
LDCI	Laboratório de Dispositivos e Circuitos Integrados
UnB	Universidade de Brasília
RF	Rádio Frequência
ANATEL	Agência Nacional de Telecomunicações
AM	$AmplitudeModulation$
FM	$FrequencyModulation$
FSK	$Frequency - ShiftKeying$
PA	$PowerAmplifier$
PLL	$Phase - LockedLoop$
NRZ	$Non - Return - to - Zero$
LSB	$LowSignificantBit$
HSB	$HighSignificantBit$
NF	$Noise - Figure$
SNR	$Signal - NoiseRatio$
RMS	$RootMeanSquare$
LNA	$LowNoiseAmplifier$
VHDL	$VHSICHardwareDescriptionLanguage$

Capítulo 1

Introdução

Esta seção tem por objetivo apresentar o contexto em que foi concebida a proposta de se implementar o modelo do transceptor RF (receptor), definir o problema, assim como os objetivos deste trabalho.

1.1 CONTEXTUALIZAÇÃO

1.1.1 Sistema de Monitoramento de Cargas

Aplicações ambientais têm oferecido um amplo leque para o desenvolvimento de novos sistemas. A atividade de exportação crescente de alimentos em escala global apresenta vários desafios para os produtores. No caso de bens perecíveis, como frutas frescas, a manutenção de condições adequadas no ambiente é um objetivo crucial. Monitorar uma carga de frutas ao longo de sua cadeia de exportação inteiro é um requisito essencial para garantir a qualidade do produto para o seu consumidor final. Através do monitoramento de produtos alimentares em todas as fases da produção, como armazenamento, processamento e distribuição, é possível evitar disseminação de praga, ajudando os consumidores, agricultores e outros [17].

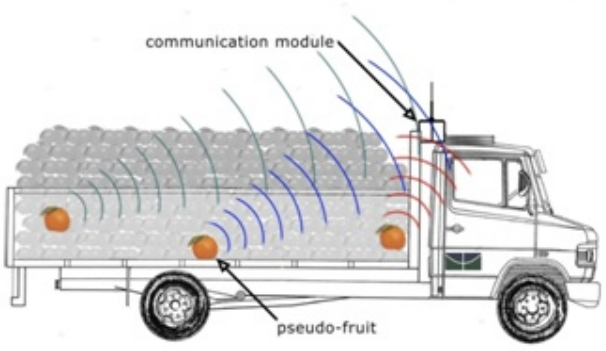

Figura 1.1: Sistema de monitoramento de frutas proposto [17].

Monitoramento de frutas em redes de distribuição comercial pode avaliar a condição do produto durante procedimentos de manipulação após a colheita. O transporte dos frutos pode causar sérios danos, como a compressão mecânica e aumento da temperatura, aumentando as perdas. Um RSoC

pode ser usado em conjunto com diferentes sensores para montar uma "fruta artificial"(pseudo-fruto), que seria enviada juntamente com a carga em containers utilizados para entregar esses produtos a consumidores finais e intermediários. Esse dispositivo autônomo irá recolher dados ambientais, bem como as respostas espectrais IR a partir de frutos, a fim de avaliar as condições de transporte e do estado da fruta durante o transporte.

O sistema é composto por módulos de controle local (pseudo-frutos); um conjunto de sensores ambientais (temperatura, umidade e impacto); comunicação e módulo de supervisão local, de dados e controle de troca de informações com links terrestres: receptor e transmissor.

Na Fig. 1.1 pode ser visto que o sistema consiste em uma rede de pseudo-frutos inseridos no interior do recipiente de carga, substituindo os frutos reais, juntamente com o restante da carga. Um APS (Active Pixel Sensor) que opera na faixa de matriz IR irá realizar o monitoramento dos frutos e adquirir as respostas espectrais. O microprocessador RSoC enviará dados recolhidos através do transceptor de RF operando em 920 MHz para o módulo de comunicação. Os dados coletados pelo módulo de comunicação podem ser analisados em centros de supervisão externa, permitindo ações preventivas e sanitárias durante o deslocamento da carga.

Como o objetivo principal do pseudo-fruto é coletar dados do ambiente, alguns outros sensores discretos podem ser incluídos posteriormente para medir outros parâmetros importantes. Os valores adquiridos pelos sensores devem ser analisados e comparados com os valores ótimos, por exemplo, o espectro esperado, a temperatura e a umidade, a fim de avaliar se os frutos serão em condições adequadas de consumo ou não. O objetivo é monitorar as condições de carga do ambiente e avaliar os danos mecânicos ao produto, umidade e efeitos da temperatura.

Ao analisar esses dados, as ações preventivas podem ser tomadas para minimizar as perdas potenciais. Eventualmente, um atuador pode ser adicionado (por exemplo, um alarme para alertar o motorista sobre condições críticas de frutas ou de bateria fraca no módulo de monitoramento) como auxílio e alerta na tomada de atitudes preventivas.

1.1.2 Modelagem

Incorporar a modelagem de um sistema como parte de um projeto possui a vantagem de extrair informações pertinentes e necessárias a partir de dados de simulações e ao mesmo tempo estimar parâmetros de interesse, caracterizando o sistema de forma mais completa. Isso significa que o processo pelo qual o modelo é desenvolvido e os componentes resultantes permitem especular sobre como o sistema funciona de forma mais pormenorizada, simulando-os, poupando tempo e dinheiro.

Os modelos oferecem uma visão abstrata do projeto em um dado momento, o que representa certos aspectos da realidade escondendo-se outros que não são relevantes ou ainda não conhecidos. Abstração é uma técnica poderosa para a concepção e implementação de sistemas complexos. Ela permite abordar a complexidade de forma que se interesse para a sua análise, omitindo aspectos desnecessários em certos momentos, para depois trabalhá-los de maneira adequada e em conjunto com o bloco por completo. Diferentes quantidades de dados correspondem a diferentes níveis de abstração. Assim, modelos de projeto em cada nível de abstração fornecem a base para aplicação

de análise, síntese e verificação técnicas [4, 5].

1.2 DEFINIÇÃO DO PROBLEMA

Tendo em vista um sistema complexo, como o sistema de monitoramento de carga exposto anteriormente, se torna necessário ter em mãos uma ferramenta capaz de analisar características e a funcionalidade do sistema antes mesmo de realizar o projeto elétrico, para escolhas de parâmetros que garantam através de co-simulações entre os diversos blocos as especificações desejadas.

1.3 OBJETIVOS E METAS

O objetivo aqui almejado é propor o modelo em alto nível na linguagem Verilog-AMS da seção de recepção do transceptor RF, sendo este componente integrante do sistema de monitoramento de carga descrito anteriormente, desenvolvido no Laboratório de Dispositivos e Circuitos Integrados (LDCI) do departamento de Engenharia Elétrica da Universidade de Brasília - UnB. Com isso em mãos, será possível realizar a co-simulação entre os diversos componentes do chip, analisando seu comportamento por completo, assim como suas características e funcionalidades.

É importante notar que toda vez que for citado transceptor RF nesse documento como referência do trabalho desenvolvido, entende-se como a seção de recepção e não este como um todo (recepção e transmissão).

1.4 APRESENTAÇÃO DO MANUSCRITO

Este relatório está organizado conforme é explicado a seguir. No capítulo 2 é apresentada uma revisão bibliográfica de forma sucinta, abordando os temas estudados durante o trabalho. Posteriormente, no capítulo 3, é apresentada a metodologia seguida para o desenvolvimento do projeto, tendo logo após o capítulo 4, onde de fato se descreve o projeto. Por fim, ao final do documento, no capítulo 5, os resultados são expostos assim como a discussão dos mesmos.

Por fim a conclusão (capítulo 6), contém a síntese de tudo o que foi apresentado, assim como propostas de melhorias a serem realizadas e diretrizes para a continuidade do trabalho.

Ao final nos Anexos é apresentado o tutorial da síntese lógica e física do CADENCE referente à algumas etapas adotadas durante o desenvolvimento do projeto. Apresenta-se ainda nessa seção os códigos desenvolvidos, assim como os passos (script) para simulação.

Capítulo 2

Revisão Bibliográfica

Nesta seção serão abordados alguns temas referentes aos elementos necessários para o entendimento básico teórico do assunto proposto, facilitando as relações entre o problema e o conhecimento existente do mesmo.

2.1 LINGUAGEM DE PROGRAMAÇÃO

A linguagem de programação é um instrumento que consiste de um método padronizado para expressar instruções tendo como objetivo possibilitar modelar o sistema que se deseja, e/ou projetar os próprios sistemas, facilitando expressar e desenvolver projetos desejáveis comparado com a linguagem que um computador entende nativamente, no código de máquina. Assim, linguagens de programação são projetadas para adotar uma sintaxe de nível mais alto, que pode ser mais facilmente entendida por programadores humanos, tornando-se então em uma ferramenta que tem como vantagem a possibilidade de compilação e simulação, fazendo com que possa ser realizada a análise dos dados simulados dando uma prévia do sistema em funcionamento, possibilitando a correção de erros ou apenas corroborando a idéia principal, validando-o.

Uma alternativa à entrada esquemática de um circuito digital em um sistema de projeto auxiliado por computador é utilizar a técnica de projeto de dispositivos lógicos programáveis (PLDs) com uma ferramenta de projeto baseado em texto ou linguagem de descrição de hardware (HDL). Exemplos de HDLs são o AHDL (Altera Hardware Description Language) e os padrões VHDL e Verilog [19].

O projetista cria um arquivo de texto, seguindo certo conjunto de regras, conhecido como sintaxe da linguagem, e usa um compilador para criar dados de programação do dispositivo lógico programável (ou PLD). Esta descrição de hardware pode ser usada para gerar projetos hierárquicos, ou seja, um componente definido em uma descrição pode ser usado para gerar um hardware específico ou ser usado como parte de outro projeto.

As HDLs têm uma grande semelhança às linguagens de programação, mas são especificamente orientadas à descrição da estruturas e do comportamento do hardware. Uma grande vantagem das HDLs em relação à entrada esquemática é que elas podem representar diretamente equações booleanas, tabelas verdade e operações complexas como operações aritméticas.

Uma descrição estrutural descreve a interconexão entre os componentes que fazem parte do circuito. Esta descrição é usada como entrada para uma simulação lógica da mesma forma que uma entrada esquemática. Uma descrição comportamental descreve o funcionamento de cada um dos componentes do circuito.

Uma HDL pode ser usada na descrição em vários níveis do circuito em desenvolvimento. Partindo de uma descrição de alto nível, pode ser usada para refinar e particionar esta descrição em outras de nível mais baixo durante o processo de desenvolvimento. A descrição final deve conter componentes primitivos e blocos funcionais.

Uma grande razão para o uso de HDLs é a síntese lógica. Uma descrição em HDL em conjunto com uma biblioteca de componentes é usada por uma ferramenta de síntese para a geração automática de um circuito digital. Além disto, estas ferramentas incluem uma etapa de otimização da lógica interna do circuito gerado, antes da geração das estruturas internas de armazenamento, da lógica combinatória e da estrutura de conexão dos componentes (*netlist*). A Fig. 2.1 mostra o das etapas principais de síntese lógica.

Figura 2.1: Fluxo das etapas de alto nível da Síntese Lógica.

Atualmente, as HDLs mais utilizadas são o VHDL e o Verilog, porém essas linguagens permitem a modelagem apenas de circuitos estritamente digitais. Para análises que envolvam sistemas com sinais analógicos há expansões dessas linguagens, tais como VHDL-AMS e Verilog-AMS que permitem a co-simulação entres os diversos tipos de sinais (analógicos e digitais).

2.1.1 Verilog-AMS

Verilog-AMS é uma linguagem de programação derivada da linguagem de descrição de hardware Verilog. Nessa linguagem são incluídos os sinais analógicos e as extensões digitais (AMS - Analog and Mixed-Signals) que fazem a descrição dos sistemas que se utilizam tanto de sinais analógicos quanto digitais.

Verilog-AMS é a fusão e ampliação de duas linguagens: Verilog HDL e Verilog-A. Essas três línguas atualmente compõem a família de linguagem Verilog. Verilog HDL permite a descrição dos componentes digitais e Verilog-A permite a descrição do analógico. Verilog-AMS combina estas duas linguagens e adiciona funcionalidade adicional para permitir a descrição dos componentes de sinal misto [14].

O padrão Verilog-AMS foi criado com a intenção de habilitar os desenvolvedores de sistemas analógicos e digitais e circuitos integrados de criar e utilizar módulos descritos em alto nível, possibilitando a simulação do funcionamento de ambos os sistemas - analógicos e digitais - e verificando sua funcionalidade, aumentando a confiabilidade do sistema.

Verilog-AMS é esperado para ter um grande impacto sobre a concepção de sistemas de sinal

misto porque ela fornece uma única linguagem e um simulador único que é compartilhado entre designers analógicos e digitais, e entre designers de blocos e designers de sistema. Ele será muito mais fácil para fornecer um fluxo de projeto único, que suporta naturalmente blocos analógicos, digitais e de sinais mistos, tornando mais simples para estes designers trabalharem juntos [14].

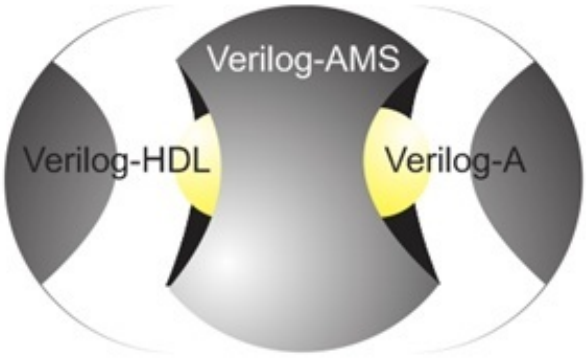

Figura 2.2: Universo linguagem Verilog-AMS.

A estrutura dos códigos descritos em linguagem Verilog-AMS segue um padrão, onde tem-se no corpo principal a definição dos parâmetros de escala de tempo, instância de bibliotecas e a descrição do sistema onde tem-se a sessão digital separada da analógica.

```
Timescale settings;
Invoking libraries;
Module;
    Description of the inputs and outputs;
    Parameters settings;
    Comportamental description of the module;
    Analog description;
endmodule
```

Figura 2.3: Padrão linguagem Verilog-AMS.

Para simulação e validação do sistema, a estrutura do código do testbench (código de teste para validação do sistema) segue o mesmo padrão, como mostra a Fig. 2.4.

```
Timescale settings;
Invoking libraries;
Module testbench;
    Description of the inputs and outputs;
    Instancing the module under test;
    Setting parameters of the signals for simulation;
endmodule
```

Figura 2.4: Padrão linguagem Verilog-AMS testbench.

2.2 ARQUITETURAS DE TRANSCEPTORES RF

Um transceptor é um dispositivo que combina um transmissor e um receptor utilizando componentes de circuito comuns para ambas funções num só aparelho. Se tiver a capacidade de apenas transmitir ou receber, não simultaneamente, é chamado de *half-duplex*, caso contrário é um *full-dlupex*. Se esses componentes não forem comuns, esse aparelho designa-se transmissor-receptor.

Em geral, é o receptor que determina a performance geral de um sistema de rádio. As principais considerações que devem ser observadas para a escolha de uma topologia para o receptor envolvem simplicidade custo, tamanho e alcance. O receptor super-heteródino é o mais utilizado em aplicações de curta distância, mas o conhecimento de outras topologias permite ao projetista selecionar a mais adequada para a sua aplicação [7].

Uma maneira verificar o trade-off em um projeto RF, tendo uma análise mais crítica dos itens citados acima, está expresso no hexágono apresentado na Fig. 2.5, onde há o detrimento de algumas características em função de outras tendo em vista a prioridade de certas especificações.

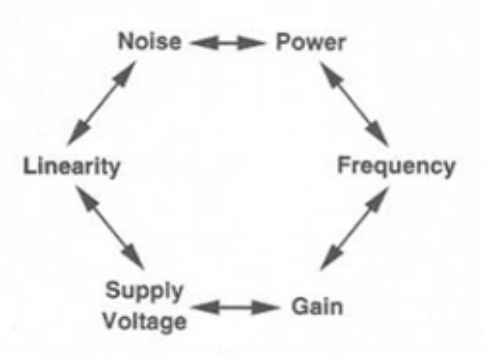

Figura 2.5: Hexágono de um projeto de RF [1].

2.2.1 LNA

Sendo o LNA o primeiro o primeiro bloco e o primeiro estágio de ganho no caminho de recepção do sinal, há de ser ter um certo cuidado em sua especificação já que ele irá inserir diretamente a figura de ruído no restante do sistema segundo a fórmula de Friis, apresentada a seguir.

$$NF_{TOTAL} = 1 + (NF_1 - 1) + \frac{NF_2 - 1}{G_{A1}} + \frac{NF_3 - 1}{G_{A1}G_{A2}} + ... + \frac{NF_m - 1}{G_{A1}G_{A2}...G_{Am-1}} \tag{2.1}$$

onde NF é a figura de ruído.

Assim sendo, para um alto ganho e uma baixa figura de ruído, implicará em um ruído transmitido ao resto do sistema baixo e não influenciará de forma significativa no sinal recebido, não comprometendo o sistema.

Na Fig. 2.6 é mostrada a arquitetura geral de um LNA. Os dois estágios, inicial e final, são a respeito de casamento de impedância, e o estágio do meio é o amplificador propriamente dito.

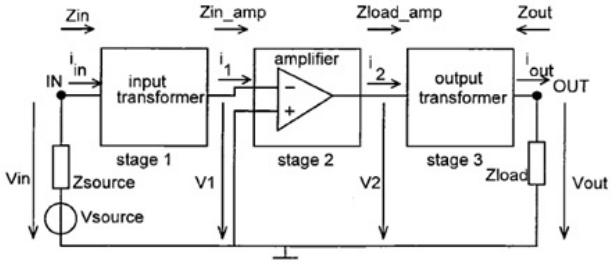

Figura 2.6: Arquitetura de um LNA [1].

2.2.2 Mixer

O mixer é um bloco que comporta duas entradas com duas frequências distintas podendo realizar operações com essas freqüências, sendo assim usado para fazer a translação da frequência do sinal de entrada para uma frequência menor (segundo sinal de entrada).

O sinal amplificado proveniente do LNA é aplicado ao mixer e a segunda entrada aplicada vem do oscilador como mostra a Fig. 2.7.

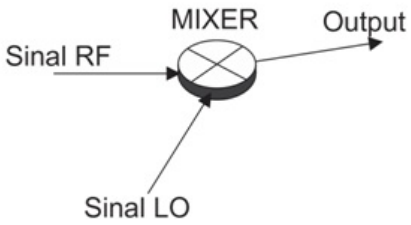

Figura 2.7: Diagrama Mixer.

Da figura define-se:

$$v_{RF} = A_{RF} sen(w_{RF} t) \tag{2.2}$$

$$v_{LO} = A_{LO} sen(w_{LO} t) \tag{2.3}$$

Na saída do mixer tem-se:

$$v_{out} = v_{RF} \cdot v_{LO} \tag{2.4}$$

Da propriedade de multiplicação de senóides, resulta em:

$$senA \cdot senB = \frac{1}{2}(cos(A - B) - cos(A + B)) \tag{2.5}$$

Logo, substituindo 2.2 e 2.3 em 2.5:

$$v_{out} = v_{RF} \cdot v_{LO} = \frac{A_{RF}A_{LO}}{2}(cos(w_{RF} - w_{LO})t - cos(w_{RF} + w_{LO})t) \tag{2.6}$$

Aplicando a transformada de Fourier para um sinal V_{RF} genérico:

$$V_{out}(w) = \frac{1}{2}(V_{RF}(w_{RF} - w_{LO}) + V_{RF}(w_{RF} + w_{LO})) \tag{2.7}$$

tal que,

$$v_{RF}(t) \leftrightarrow V_{RF}(w) \tag{2.8}$$

Nota-se claramente a translação da freqüência do sinal RF de entrada em relação a freqüência do sinal LO, conforme Fig. 2.8

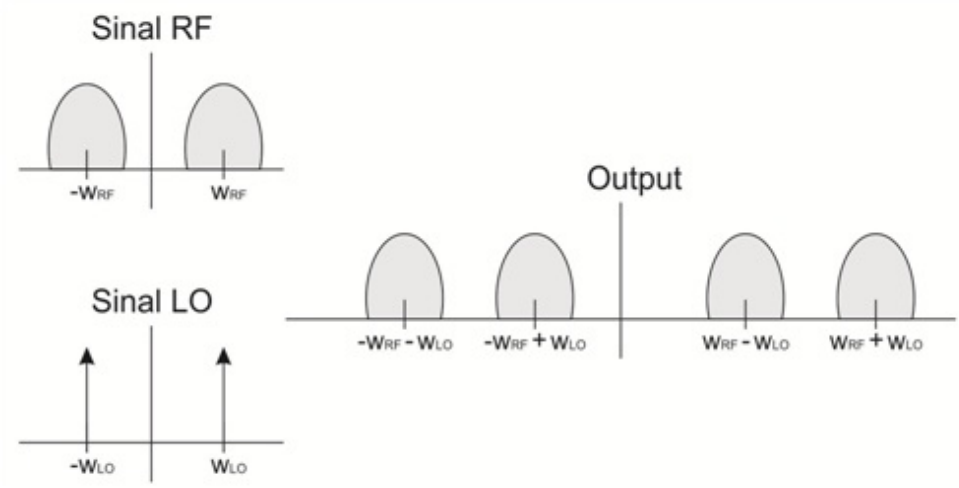

Figura 2.8: Análise na frequência do mixer.

2.2.3 Conversor A/D

O Conversor A/D é um estágio que basicamente quantitatiza um sinal analógico e o transforma em digital. Ele se torna importante já que há blocos no sistema de RF em que o processamento de sinais é realizado com sinais digitais.

Na conversão de um sinal é necessário que esse processamento seja acompanhado de um passo de referência, já que será justamente discretizado o sinal analógico de entrada, sendo necessário portanto um clock. O sinal digital de saída dependerá da resolução do sistema, ou seja, possíveis valores a serem associados ao valor da tensão de entrada. Caso tenha-se N bits disponíveis, então ter-se-á 2^N valores que definirão a resolução do conversor.

Para um sinal de entrada de 0 a V_{REF} volts, e uma resolução em que $N = 3$, podemos observar na Fig. 2.9 que o erro associado à quantização é dado em LSB, onde LSB é a menor diferença entre os valores de saída.

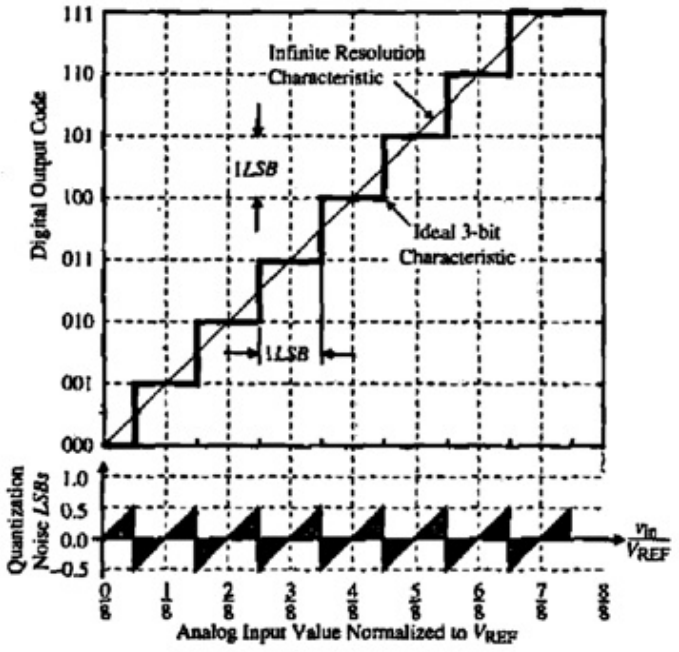

Figura 2.9: Erro de quantização [11].

Em relação a um sistema RF em que os dados estão em modulação FSK, a informação está de fato na freqüência, não sendo significativamente interessante então analisar os erros de quantização.

O transceptor RF elaborado e discutido nesse documento contém apenas o Conversor A/D, que está posicionado após o Mixer. É bem verdade que o bloco que gera sinais para excitar e validar o sistema é um Conversor D/A, mas ressalta-se que este não faz parte do transceptor, sendo apenas um bloco de teste. Assim sendo, o conversor D/A não é discutido nesta seção, mas os fundamentos teóricos são análogos aos apresentados aqui a respeito do conversor A/D [11].

2.2.4 Demodulador

O demodulador é o último estágio no receptor do transceptor RF. Sua finalidade é recuperar os dados recebidos, estes codificados em NRZ, Manchester, ou qualquer outro protocolo de codificação realizando o processo inverso, decodificando-os, retirando a informação para que seja disponibilizada diretamente em registradores específicos para esse fim na memória.

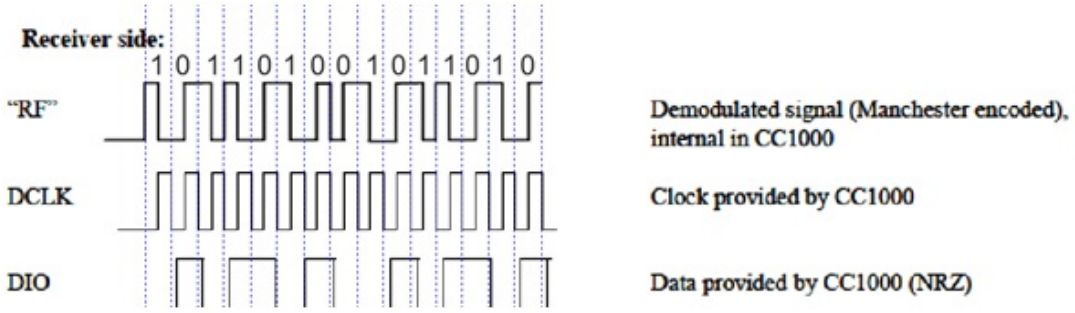

Figura 2.10: Demodulação de um sinal Manchester [12].

Na Fig. 2.10 nota-se que a entrada do decodificador é o sinal RF, codificado em Manchester, e referenciado por um sinal de clock. Assim é retido o dado de interesse puro, de acordo com o protocolo NRZ.

2.2.5 Comparador

O comparador tem por finalidade realizar a comparação entre dois sinais analógicos e apresenta, na saída, um sinal binário baseado no resultado da comparação. Funcionalmente, o comparador opera apresentando um sinal positivo na sua saída quando for positiva a diferença entre suas entradas não inversora e inversora, e negativo quando negativo for o resultado de tal subtração.

No entanto, o comparador a que iremos nos referir tem o objetivo de realizar a comparação entre o valor do sinal de entrada em referência ao *ground* e definir na saída valores com maior magnitude para diferenças positivas e menor magnitude para diferenças negativas. No caso de 2 bits, atribui-se 1 e 0, respectivamente, ao contexto.

Nota-se, portanto, que o comparador identifica-se com o Conversor A/D, transformando o sinal de entrada analógico em um sinal digital na saída, tendo resolução $N = 2$.

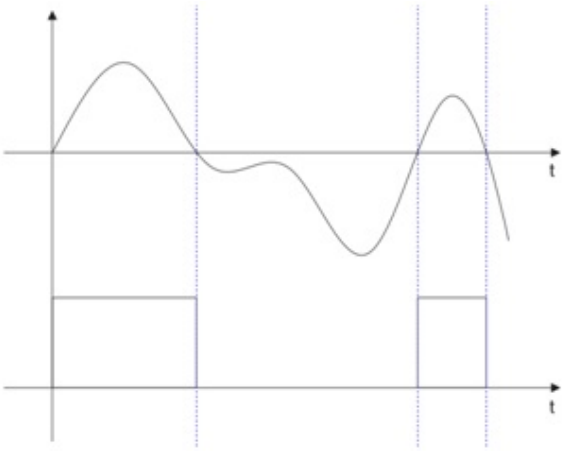

Figura 2.11: Comparador.

2.3 CONCEITOS

2.3.1 Linearidade

Um sistema linear excitado com entradas $x(t)_i$, tendo como saída $y(t)_i$ associadas à entrada, é definido linear se a superposição se tornar válida para as entradas e saídas deste mesmo sistema. Considere portanto as i entradas citadas acima, tais como suas saídas. Logo, têm de ser verdade as seguintes propriedades:

$$x(t)_0 + x(t)_1 + \cdots + x(t)_i \rightarrow y(t)_0 + y(t)_1 + \cdots + y(t)_i \tag{2.9}$$

$$a_i x(t)_i \rightarrow a_i y(t)_i \tag{2.10}$$

Assim, ao analisar as equações 2.9 e 2.10, verificamos a propriedade da superposição:

$$a_0 x(t)_0 + a_1 x(t)_1 + \cdots + a_i x(t)_i \rightarrow a_0 y(t)_0 + a_1 y(t)_1 + \cdots + a_i y(t)_i \tag{2.11}$$

Enquanto muitos circuitos analógicos RF podem ser aproximados por um modelo linear para obter suas respostas para pequenos sinais, não-linearidades geralmente levam a fenômenos interessantes e importantes. Para simplificar, vamos limitar a análise para um sistema sem memória, não variante no tempo e assumir [1]:

$$y(t) \rightarrow \alpha_1 x(t) + \alpha_2 x^2(t) + \alpha_3 x^3(t) \tag{2.12}$$

2.3.2 Ponto de interceptação de 3^a ordem

Se em um sistema não linear é aplicado um sinal senoidal na entrada, tal como $Acoswt$ são gerados harmônicos, estes múltiplos inteiros da freqüência fundamental:

$$y(t) = \alpha_1 Acoswt + \alpha_2 A^2 cos^2 wt + \alpha_3 A^3 cos^3 wt \tag{2.13}$$

$$y(t) = \alpha_1 Acoswt + \frac{\alpha_2 A^2}{2}(1 + cos2wt) + \frac{\alpha_3 A^3}{4}(3coswt + cos3wt) \tag{2.14}$$

$$y(t) = \frac{\alpha_2 A^2}{2} + \left(\alpha_1 A + \frac{3\alpha_3 A^3}{4}\right) coswt + \frac{\alpha_2 A^2}{2} cos2wt + \frac{\alpha_3 A^3}{4} cos3wt \tag{2.15}$$

Quando dois sinais com freqüências diferentes são aplicadas a um sistema não-linear, a saída exibe em geral algumas componentes que não são harmônicos das frequências da entrada. Chamado de intermodulação, esse fenômeno surge a partir da multiplicação de dois sinais quando sua soma é elevada para uma potência maior que a unidade [1].

Assumindo um sinal de entrada $x(t) = A_1 cosw_1 t + A_2 cosw_2 t$, então

$$y(t) = \alpha_1(A_1 cosw_1 t + A_2 cosw_2 t) + \alpha_2(A_1 cosw_1 t + A_2 cosw_2 t)^2 + \alpha_3(A_1 cosw_1 t + A_2 cosw_2 t)^3 \tag{2.16}$$

Expandindo a equação acima, obtem-se os produtos de intermodulação:

$$w = w_1 \pm w_2 : \alpha_2 A_1 A_2 cos(w_1 + w_2)t + \alpha_2 A_1 A_2 cos(w_1 - w_2)t \tag{2.17}$$

$$w = 2w_1 \pm w_2 : \frac{3\alpha_3 A_1^2 A_2}{4} cos(2w_1 + w_2)t + \frac{3\alpha_3 A_1^2 A_2}{4} cos(2w_1 - w_2)t \tag{2.18}$$

$$w = 2w_2 \pm w_1 : \frac{3\alpha_3 A_2^2 A_1}{4} cos(2w_2 + w_1)t + \frac{3\alpha_3 A_2^2 A_1}{4} cos(2w_2 - w_1)t \tag{2.19}$$

Então as componentes fundamentais são expostas a seguir:

$$w = w_1, w_2 : \left(\alpha_1 A_1 + \frac{3\alpha_3 A_1^3}{4}\frac{3\alpha_3 A_1 A_2^2}{2}\right) cosw_1 t + \left(\alpha_1 A_2 + \frac{3\alpha_3 A_2^3}{4}\frac{3\alpha_3 A_1^2 A_2}{2}\right) cosw_2 t \tag{2.20}$$

Pode-se observar que o produto de intermodulação $2w_1 - w_2$ e $2w_2 - w_1$ contêm valores significantes interferindo na não linearidade e distorção do sinal de sinal de entrada. Essa ilustração pode ser visualizada na Fig. 2.12.

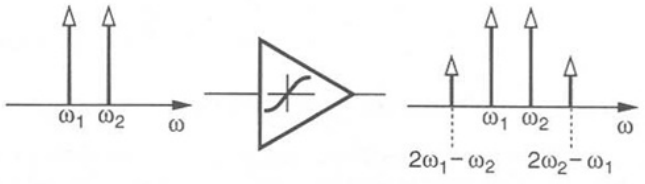

Figura 2.12: Intermodulação [1].

A distorção causada pela intermodulação de 3ł ordem é chamada de ponto de interceptação de 3^a ordem, IP_3 (third intercept point). Na figura a seguir pode-se verificar que quando a amplitude A aumenta as fundamentais crescem linearmente enquanto a intermodulação de 3^a ordem crescem em um fator de A_3. O ponto de interceptação de terceira ordem é definido como a interseção das duas linhas.

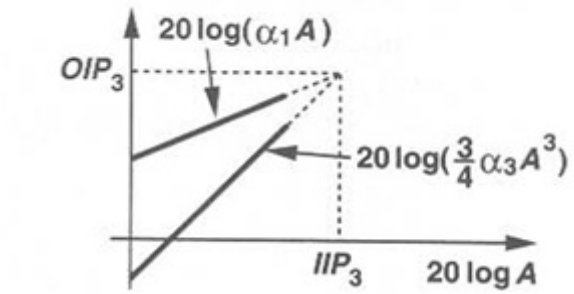

Figura 2.13: Crescimento das componentes de saída em um teste de intermodulação [1].

Do gráfico podemos deduzir o cálculo de IP_3:

$$y(t) = \left(\alpha_1 + \frac{9\alpha_3 A^2}{4}\right) Acosw_1 t + \left(\alpha_1 + \frac{9\alpha_3 A^2}{4}\right) Acosw_2 t + \frac{3\alpha_3 A^3}{4} cos \mid 2w_1 - w_2 \mid t + \frac{3\alpha_3 A^3}{4} cos \mid 2w_2 - w_1 \mid t + \cdots \tag{2.21}$$

Considerando $\alpha_1 \gg \frac{3\alpha_3 A^3}{4}$, então

$$\mid \alpha_1 \mid A_{IP3} = \frac{3 \mid \alpha_3 \mid A_{IP3}^3}{4} \tag{2.22}$$

$$A_{IP3} = \sqrt{\frac{4}{3} \mid \frac{\mid \alpha_1 \mid}{\mid \alpha_3 \mid} \mid} \tag{2.23}$$

Logo,

$$IP_3 = \alpha_1 A_{IP3} \tag{2.24}$$

2.3.3 Sensibilidade

A sensibilidade em um receptor RF é definida como a magnitude do sinal que o sistema consegue detectar com um sinal ruído aceitável [1].

O cálculo da sensibilidade é dado por

$$P_{in,min|dBm} = P_{RS|\frac{dBm}{Hz}+NF|_{dB}+SNR_{min|dB}+10logB} \tag{2.25}$$

onde

$$NF = \frac{SNR_{in}}{SNRout} \tag{2.26}$$

$$NF = \frac{P_{sig}/P_{RS}}{SNR_{out}} \tag{2.27}$$

e

$$P_{RS} = \frac{4kTR_S}{4}\frac{1}{R_{in}} \tag{2.28}$$

$$P_{RS} = -174\frac{dBm}{Hz} \tag{2.29}$$

Logo,

$$P_{in,min} = -174\frac{dBm}{Hz} + NF + SNR_{min} + 10logB \tag{2.30}$$

2.3.4 Ruído

Há vários tipos de ruídos: ruído branco, ruído vermelho, ruído rosa, ruído térmico, etc. Independentemente disso, o ruído gerado em um amplificador é quantificado em uma série de maneiras. O fator de ruído (NF - noise-factor) indica a degradação razão sinal-ruído (SNR) causada pelos componentes em um sistema de sinal RF. Pode-se definir ruído como qualquer sinal indesejado introduzido ao sistema.

A razão sinal-ruído, ou apenas SNR (signal-to-noise) quantifica o quanto o sinal de entrada é prejudicado pelo ruído em termos absolutos, sendo definida pela relação entre a potência do sinal de entrada e a potência do sinal de ruído:

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{2.31}$$

O fator de ruído é definido como:

$$NF = \frac{P_{no}}{G_A P_{ni}} \tag{2.32}$$

Onde tem-se que o G_A é o ganho de potência e P_ni é o ruído inserido na fonte. Para a inserção de um ruído branco, pode-se gerar valores aleatórios a partir de uma distribuição normal sendo o desvio padrão adotado como o valor RMS referente à raiz quadrada do desvio quadrático médio de um sinal de uma linha de base determinada, ou seja

$$\sigma = V_{RMS} \tag{2.33}$$

Em relação ao ruído térmico, imagine o sistema onde há uma fonte seguida de uma impedância de entrada praticamente nula e de um resistor convencional. Assim, a potência de ruído disponível a partir de uma resistência a uma temperatura T é kTB, onde k é a constante de Boltzmann, T é a temperatura e B é a largura de banda.

Tomando-se que não há carga no sistema e que as duas extremidades do sistema estejam interligadas, tem-se que

$$e_n^2 = 4kTBR \tag{2.34}$$

Dessa forma pode-se calcular a potência do ruído na saída do sistema a partir do fator de ruído (Equação 2.32):

$$NF = \frac{V_{out}^2}{V_{in}^2 G_A} \tag{2.35}$$

$$NF = \frac{V_{out}^2}{\sqrt{4kTBR^2} G_A} \tag{2.36}$$

$$V_{out}^2 = NF \cdot V_{innonlinear}^2 \tag{2.37}$$

$$V_{out} = \sqrt{NF \cdot V_{innonlinear}^2} \tag{2.38}$$

$$V_{out} = \sqrt{NF \cdot e_{n-nonlinear}^2} \tag{2.39}$$

Onde os dados apresentados são expostos na figura abaixo:

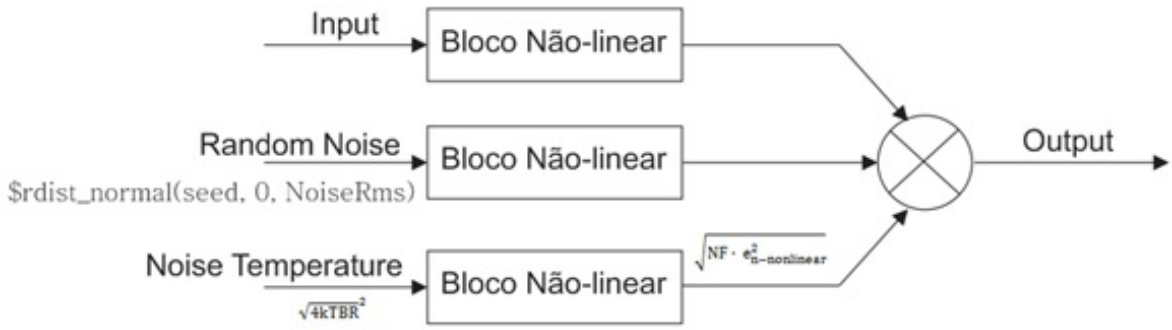

Figura 2.14: Diagrama insersão ruído.

2.3.5 Ponto de compressão

Tomando-se nota novamente da equação 2.15 e ressaltando a expansão dos harmônicos, tem-se a amplitude resultante no primeiro harmônico para um sinal $Acoswt$ tal que

$$y(t) = \cdots + \left(\alpha_1 A + \frac{3\alpha_3 A^3}{4}\right) coswt + \cdots \tag{2.40}$$

Assim pode-se verificar que se a amplitude do sinal aumentar o ganho dos harmônicos também começa a variar, e é evidente que a amplitude do 3^o harmônico com magnitude $\frac{3\alpha_3 A^2}{4}$ a partir de um ponto começa a afetar significativamente o sinal de interesse.

O ponto de compressão de 1 dB pode ser verificado na Fig. 2.15 onde é mostrado em escala logarítmica a função entre a amplitude do sinal de entrada e o sinal de saída. Esse ponto é visto quando o valor da amplitude do sinal de saída diferencia em 1 dB do sinal ideal.

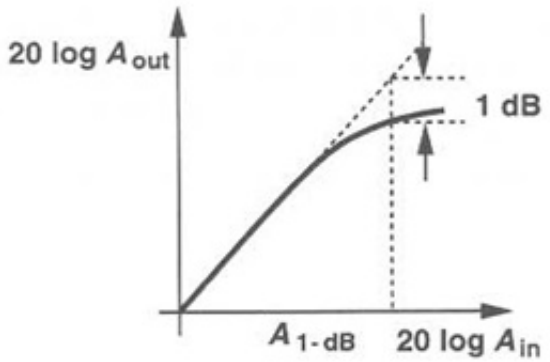

Figura 2.15: Ponto de compressão de 1 dB [1].

O cálculo do ponto de compressão de 1 dB pode ser escrito a partir de

$$20log\left(\alpha_1 + \frac{3}{4}\alpha_3 A_{1-dB}^2\right) = 20log(\alpha_1) - 1dB \tag{2.41}$$

$$A_{1-dB} = \sqrt{0,145 \mid \frac{\alpha_1}{\alpha_3} \mid} \tag{2.42}$$

2.4 MODULAÇÃO

Modulação e detecção analógica e digital são funções essenciais em sistemas de comunicação. Enquanto novos métodos de realizar estas funções continuam a surgir, determinados regimes têm amadurecido ao longo dos anos e são utilizados em muitas aplicações RF [1].

Os sinais transmitidos/recebidos em comunicações de sistemas RF são em alta freqüência devido a algumas vantagens como na transmissão de dados compensando as não idealidades do canal. A modulação adiciona portanto uma portadora com a freqüência de interesse para a transmissão do sinal fazendo com que no espectro do sinal que está na banda base seja adicionado a freqüência w_c da portadora, como é verificado na Fig. 2.16.

Figura 2.16: (a) Sinal em banda base. (b) Sinal modulado na frequência da portadora.

A demodulação segue o mesmo princípio da modulação, sendo porém no sentido inverso. A partir de algumas técnicas, retira-se o sinal filtrando-se na banda passante de interesse, estando esta agora em banda base. Técnicas de interesse serão apresentadas *a posteriori*.

Assim, na Fig. 2.17 é apresentado o sistema de comunicação de forma macro, onde tem-se o modulante que adiociona uma portadora ao sinal a ser transmitido por motivos de melhor desempenho; especificações técnicas; conveniência do tamanho da antena; regulações da ANATEL; e/ou formas de evitar as não idealidades do canal. Logo após a transmissão do sinal através do canal tem-se o demodulador, que realiza o processo inverso para recepção dos dados transmitidos.

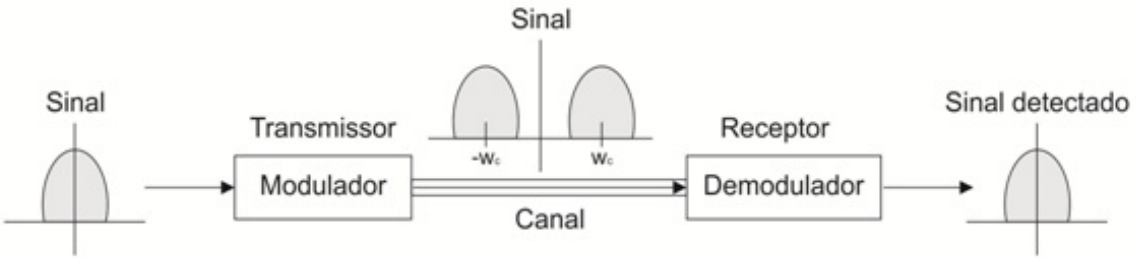

Figura 2.17: Visão macro de um sistema de comunicação.

2.4.1 Modulação em Amplitude

Modulação em amplitude é uma técnica de modulação em que a amplitude de um sinal com a freqüência de interesse, este chamado de portadora, varia de acordo com o sinal modulante, ou seja, do sinal a ser transmitido.

Na Fig. 2.18 pode-se ver que a amplitude do sinal até o tempo $t = 0$ (nota-se que o tempo é designado para análises qualitativas, desprezando a incoerência do tempo negativo) é puramente a portadora, sendo uma senóide de amplitude V_C fixa e freqüência w_c. A partir de $t = 0$, o sinal mantém a freqüência, mas varia de acordo com o sinal modulante, percebendo-se ser a própria envoltória do sinal resultante.

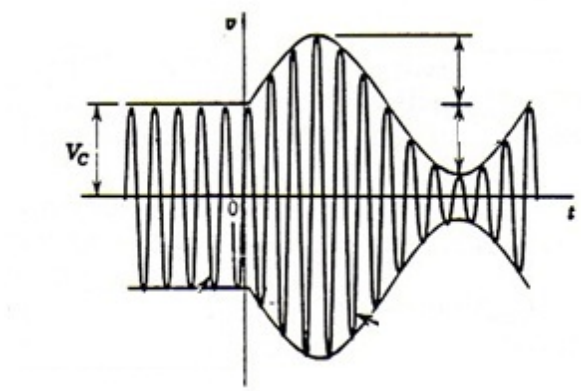

Figura 2.18: Modulação em amplitude.

Tendo-se o sinal $m(t)$ e um sinal modulante $Acos(w_c t + \theta_c)$, conforme o sistema apresentado abaixo, e tomando-se θ_c nulo sem prejudicar a análise geral. Então, se

$$m(t) \leftrightarrow M(w) \tag{2.43}$$

teremos

$$m(t)cosw_c t \leftrightarrow \frac{1}{2}\left[M(w + w_c) + M(w - w_c)\right] \tag{2.44}$$

A representação dessa tranalação em frequência, pode ser vista na Fig. 2.19, onde o sinal de entrada $m(t)$ é multiplicado pela portadora $Acos(w_c t)$.

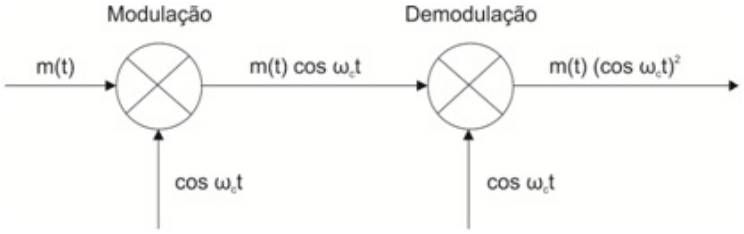

Figura 2.19: Modulação/Demodulação em amplitude.

Sendo assim, podemos observar o espectro do sinal modulado no domínio da freqüência, conforme a Fig. 2.20.

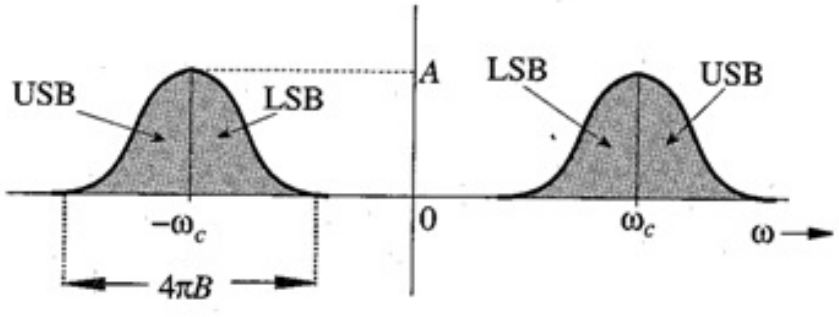

Figura 2.20: Espectro do sinal modulado[8].

Matematicamente, é uma aplicação direta da propriedade de deslocamentos em frequências da transformada de Fourier, assim como da propriedade da convolução.

Para realizar a demodulação em modulação AM, basta multiplicar novamente pela portadora, ocorrendo novamente as propriedades apresentadas anteriormente, e após filtrar a banda de interesse em banda base.

$$e(t) = m(t)(cosw_c t)^2 \tag{2.45}$$

$$e(t) = \frac{1}{2}[m(t) + m(t)cosw_c t] \tag{2.46}$$

Aplicando a transformada de Fourier,

$$E(w) \leftrightarrow \frac{1}{2}M(w) + \frac{1}{4}[M(w + 2w_c) + M(w - 2w_c)] \tag{2.47}$$

Outra forma bem simples de se retirar a informação do sinal modulado é aplicar um sistema que detecte apenas a envoltória do sinal, já que é esta que contém os dados transmitidos (notar apenas que o offset seja maior que a amplitude para não ocasionar a detecção de um sinal *falso* - entende-se por $falso$ o sinal não coerente aos dados de entrada).

2.4.2 Modulação em Frequência

Na modulação em freqüência, ou apenas modulação FM (Frequency Modulation), ao contrário da modulação AM que contém a informação na amplitude, e como o próprio nome já indica, os dados estão contidos na frequência. Da mesma forma que exposto anteriormente, há uma portadora na qual sua freqüência irá variar de acordo com o sinal modulante. Pode-se verificar de forma mais nítida na Fig. 2.21.

Nota-se que a onda modulada tem a freqüência instantânea linearmente proporcional ao valor instantâneo do sinal modulante. Portanto, uma onda FM sofre diretamente desvios de freqüência e fase. A sua fase sofre uma variação proporcional à integral do sinal modulante.

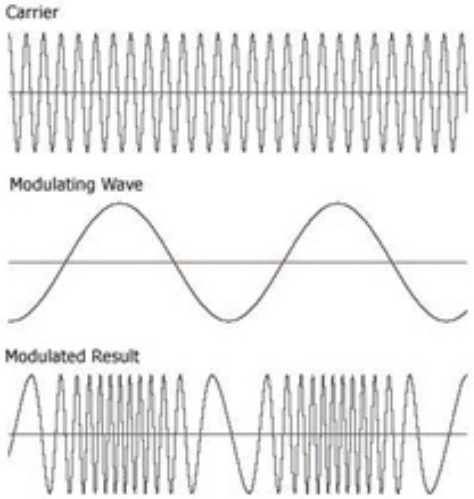

Figura 2.21: Modulação FM.

Podemos verificar de forma geral a situação apresentada acima, considerando-se um sinal modulante $Acos\theta(t)$, onde $\theta(t)$ é um ângulo geral. Assim, para um sinal $m(t)$ e sabendo-se que

$$\theta(t) = w_c t + \theta_0 \tag{2.48}$$

então pode-se verificar que o ângulo $\theta(t)$ varia linearmente com o sinal $m(t)$,

$$\theta(t) = w_c t + \theta_0 + k_p m(t) \tag{2.49}$$

onde k_p é uma constante. Logo, tomando-se $\theta_0 = 0$,

$$\varphi_{PM}(t) = A\ cos[w_c t + k_p m(t)] \tag{2.50}$$

Tem-se finalmente:

$$\Phi_{FM}(t) = A\ cos\left[w_c t + k_f \int_{-\infty}^{t} m(\alpha) d\alpha\right] \tag{2.51}$$

tal que k_f é uma constante.

Um das desvantagens dos receptores FM é de apresentarem uma característica conhecida como efeito de captura, ou seja, se existirem dois ou mais sinais de FM emitidos na mesma frequência, o receptor de FM irá responder ao sinal de maior potência e ignorar os menores (os restantes), ou apenas ocorrerá a interferência de ambos, forçando que haja então banda reservada para transmissão desses sinais.

2.4.3 Modulação FSK

A modulação FSK é uma forma de modulação em freqüência, onde atribui-se freqüências diferentes para a portadora em função do bit que é transmitido, ou seja, o sinal modulante é digital. Portanto, quando um bit 0 é transmitido, a portadora assume a freqüência correspondente a um bit 0 durante o período de duração de um bit, e em contrapartida quando um bit 1 é transmitido, a freqüência da portadora é modificada para um valor correspondente a um bit 1 e analogamente permanece nesta freqüência durante o período de duração de 1 bit, como mostrado na Fig. 2.22.

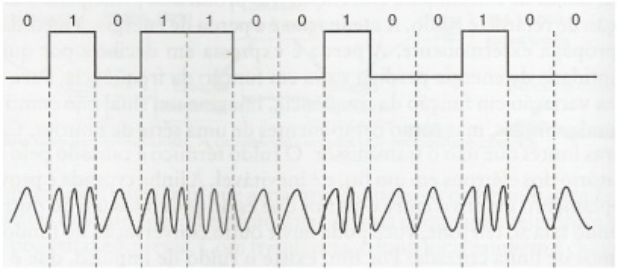

Figura 2.22: Modulação FSK.

A modulação FSK apresenta o inconveniente de ocupar uma banda de freqüência bastante alta, devido a estas variações bruscas de freqüência em função da transição de bits, além de possibilitar taxas de transmissão relativamente baixas.

2.5 RECEPTOR SUPER-HETERÓDINO

A configuração super-heteródina é a mais utilizada para comunicação via rádio. Sua operação consiste em levar os sinais recebidos para uma freqüência intermediária onde seja mais fácil amplificar, filtrar e detectar o sinal recebido. A alta performance desta topologia é possível pelo fato de que os estágios de filtragem e amplificação são realizados numa freqüência que não muda conforme se altera o canal de recepção. Além disso, numa freqüência menor que a utilizada para a transmissão é possível realizar amplificadores de maior ganho sem o risco de instabilidade [7].

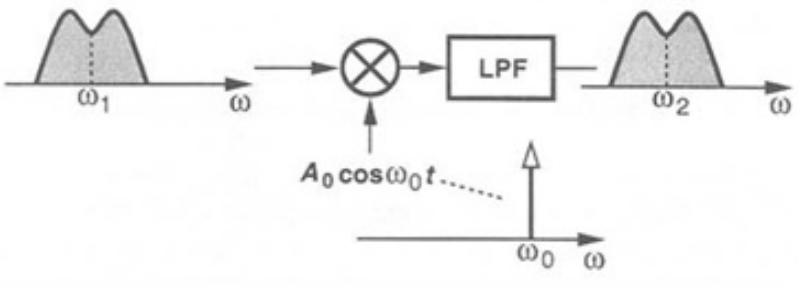

Figura 2.23: Receptor super-heteródino [1].

Pode-se ver pela figura acima que o sinal em uma freqüência w_1 é transladado a uma freqüência w_2 por um sinal modulante $A_0\ cosw_0t$. Renomeando a freqüência w_0 para w_{LO}:

$$w_2 = w_1 - w_{LO} \tag{2.52}$$

Assim, o sinal é recuperado tendo em vista que a frequência de interesse é justamente a w_2.

2.6 HALF-DUPLEX

O transceptor RF pode ser definido como *half-duplex* ou *full-duplex*. Essas definições dependem da propriedade do transceptor. Tomemos como exemplo o esquemático do chip CC1000, onde é um *single-chip UHF transceptor* desenvolvido para aplicações wireless de baixas potências e de baixas voltagens.

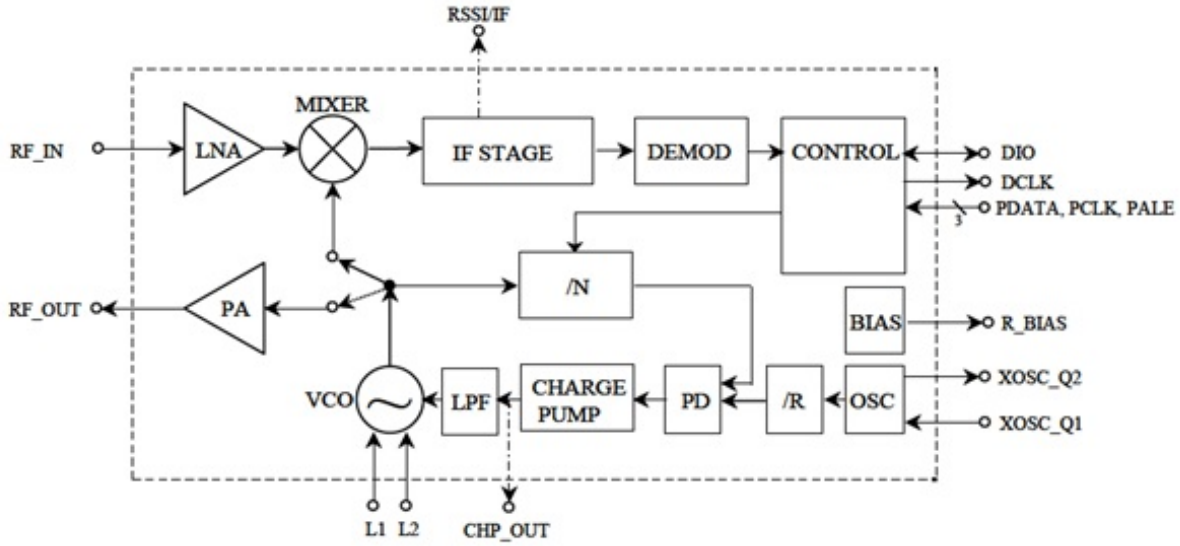

Figura 2.24: Diagrama Transceptor Half-Duplex [12].

Note que no esquemático ocorre um chaveamento para os blocos PA (Power amplifier) e o mixer. Quando chaveado para o PA, a potência do sinal codificado é amplificada e o sinal é transmitido, e quando é chaveado para o mixer, este é alimentado com o oscilador fornecendo uma freqüência para transladar o sinal RF recebido. Dessa forma fica fácil de verificar que *half-duplex* significa que o transceptor apenas transmite ou recebe dados, não faz os dois ao mesmo tempo (*full-duplex*). Sendo assim, tem uma chave seletora para indicar qual tarefa está fazendo, comando que é dado a partir do controlador (CONTROL nesse caso).

2.7 FILTROS

A rede de um filtro geralmente é projetada para permitir a passagem de um sinal com uma faixa específica de freqüência e rejeitar ou atenuar sinais cujo espectro de freqüência esteja fora dessa faixa. Os filtros mais comuns são os filtros passa-baixa, que permitem a passagem das baixas freqüências e rejeitam as altas freqüências; os filtros passa-alta, que permitem a passagem das altas freqüências e bloqueiam as baixas freqüências; os filtros passa-banda, que permitem a passagem

de uma banda específica de freqüências e rejeitam todas as freqüências fora dessa faixa; e os filtros rejeita-banda, que são projetados com a finalidade específica de rejeitar uma banda particular de freqüências e permitem a passagem de todas as demais freqüências [2].

Para análise de um filtro passa-baixa, veremos uma configuração básica construída com um resistor e um capacitor, de acordo com a Fig. 2.25.

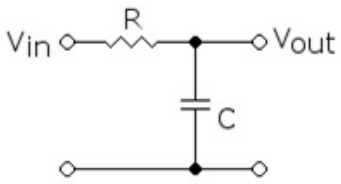

Figura 2.25: Filtro RC passa-baixa.

Dele pode-se verificar que o ganho é dado por:

$$G_v(jw) = \frac{1}{1 + jwRC} \tag{2.53}$$

Assim sendo, a magnitude do sinal é dado por:

$$M(w) = \frac{1}{\sqrt{1 + (wRC)^2}} \tag{2.54}$$

A figura abaixo ilustra a ponderação no domínio da freqüência.

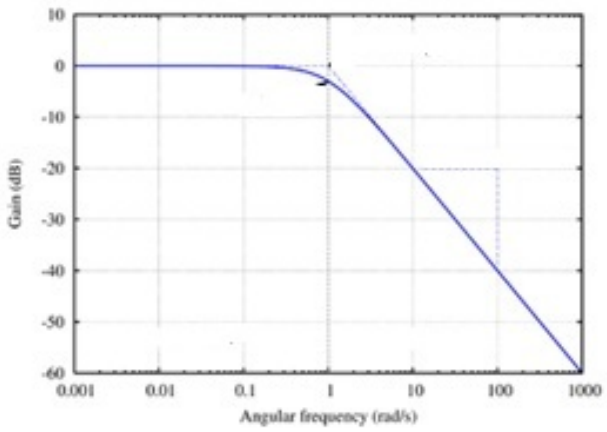

Figura 2.26: Resposta no domínio da frequência de um filtro passa-baixa.

De maneira análoga, pode-se verificar o comportamento de um filtro passa-banda com uma configuração básica mostrada na Fig. 2.27 juntamente com sua resposta no domínio da freqüência.

A função de transferência para a tensão é

$$G_v(jw) = \frac{R}{R + j(wL - \frac{1}{wC})} \tag{2.55}$$

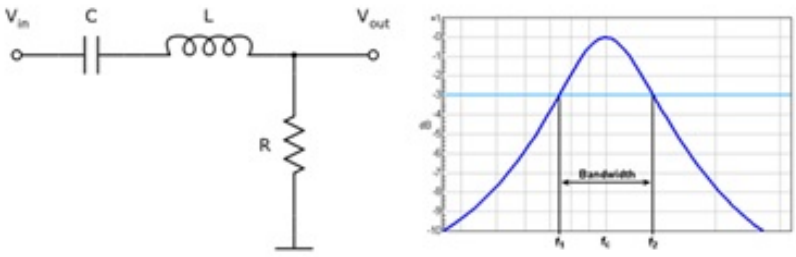

Figura 2.27: Filtro passa-banda.

Logo,

$$M(jw) = \frac{RCw}{\sqrt{(RCw)^2 + (w^2LC - 1)^2}} \tag{2.56}$$

Isso implica

$$w_{BAIXA} = \frac{-\left(\frac{R}{L}\right) + \sqrt{\left(\frac{R}{L}\right)^2 + 4w_0^2}}{2} \tag{2.57}$$

$$w_{ALTA} = \frac{+\left(\frac{R}{L}\right) + \sqrt{\left(\frac{R}{L}\right)^2 + 4w_0^2}}{2} \tag{2.58}$$

Então, a largura de banda é:

$$LB = w_{ALTA} - w_{BAIXA} = \frac{R}{L} \tag{2.59}$$

2.7.1 Butterworth

Um filtro ideal não deve apenas rejeitar as freqüências não desejadas, mas também deve ter um ganho uniforme para as freqüências desejáveis. Os filtros de Butterworth são especificados de modo a terem uma função de transferência com o mínimo de oscilações tanto na banda passante como na banda de corte.

No caso de um filtro passa-baixa, se f_0 é a freqüência de corte e $f(xf_0)$ é qualquer outra freqüência, então a razão da saída pela entrada é na forma:

$$F = (I + x^m)^{-1} \tag{2.60}$$

onde m aumenta de acordo com o número de elementos empregados. A topologia básica pode ser visualizada na Fig. 2.28.

Para o caso de dois elementos, conforme apresentado, o fator do filtro é

$$F = (I + x^8)^{-1} \tag{2.61}$$

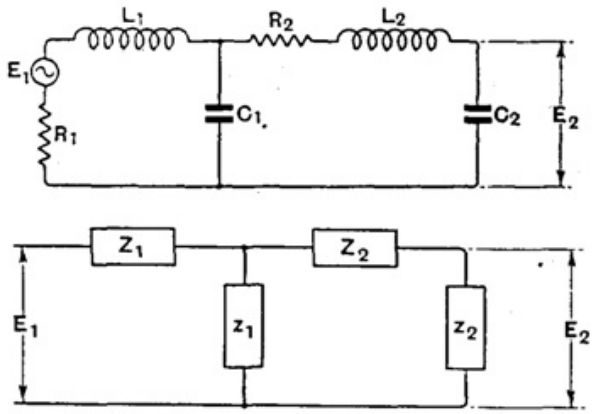

Figura 2.28: Topologia filtro passa-baixa Butterworth [3].

que pode ser obtido satisfazendo as equações envolvendo f_0, R_1, L_1, C_1, R_2, L_2, C_2.

Generalizando o circuito, tem-se:

$$L_1 C_1 = \frac{1}{w_i^2} \tag{2.62}$$

$$\frac{R_1}{w_1 L_1} = P_1 \tag{2.63}$$

Logo,

$$Z_i = R_i + jwL_i \tag{2.64}$$

$$z_i = \frac{I}{jwC_i} \tag{2.65}$$

Então

$$I + \frac{Z_i}{z_i} = I - \frac{w^2}{w_i^2} + \frac{jP_i w}{w_i} \tag{2.66}$$

Considerando:

$$\frac{w}{w_1} = \frac{x}{a} \tag{2.67}$$

$$\frac{w}{w_2} = ax \tag{2.68}$$

$$\frac{C_2}{C_1} = \beta \tag{2.69}$$

Tem-se

$$x^2 = \frac{w^2}{w_1 w_2} \tag{2.70}$$

Da Fig. 2.28 tem-se

$$\frac{E_1}{E_2} = \left(I + \frac{Z_1}{z_1}\right)\left(I + \frac{Z_2}{z_2}\right) + \frac{Z_1}{z_2} \tag{2.71}$$

Substituindo-se os resultados encontrados:

$$\frac{E_1}{E_2} = \left(I - \frac{x^2}{\alpha^2} + \frac{jP_1 x}{\alpha}\right)\left(I - \alpha^2 x^2 + jP_1 x\right) + \beta\left(\frac{-x^2}{\alpha^2} + \frac{jP_1 x}{\alpha}\right) \tag{2.72}$$

Separando partes reais e imaginárias e simplificando, resultam as seguintes equações:

$$A = \frac{I + \beta}{\alpha^2} + \alpha^2 + P_1 P_2 \tag{2.73}$$

$$B = \frac{P_1(I + \beta)}{\alpha} + P_2 \alpha \tag{2.74}$$

$$C = P_1 \alpha + \frac{P_2}{\alpha} \tag{2.75}$$

Assim, calcula-se P_1, P_2, α e β para escolher as freqüências desejadas.

De maneira análoga calcula-se os parâmetros para o filtro Butterworth passa-banda.

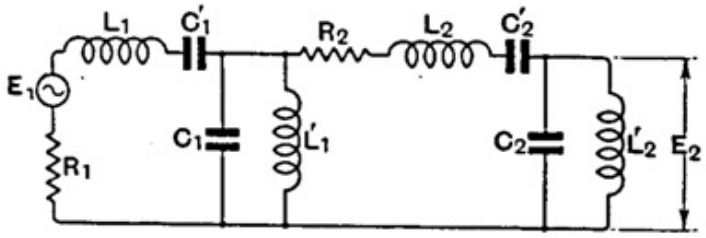

Figura 2.29: Filtro Butterworth passa-banda [3].

A partir desses cálculos, tabelas com valores desejados são elaboradas para construção dos filtros, permitindo calcular numericamente valores dos componentes que compõem o filtro de forma iterativa.

2.8 CODIFICAÇÃO

Para que dados sejam transmitidos é necessário que haja a codificação dos mesmos, visando a otimização na transmissão, já que se forem puramente enviados o risco de se perderem, seja por não-linearidade, ruído, interferência, não idealidade do canal, entre outros, é alto. Tendo isso em mente, modificam-se as características de um sinal para torná-lo mais apropriado para uma

aplicação específica, como transmissão ou armazenamento de dados, facilitando a identificação na recepção.

2.8.1 NRZ

O código NRZ (não retorno a zero) é uma forma de codificação de dados binários onde os 1's são representados por uma condição significativa, um alto valor, como 3,3 V, e os 0's por valores pequenos, tais como 0 V, sem qualquer interrupção (valor neutro ou condição de repouso) no envio dos dados, mesmo consecutivos.

O problema que pode ocorrer é de o receptor perder a sincronização ao codificar uma ligação síncrona caso tenha sequências consecutivas com o mesmo valor, já que não há nenhuma informação de que o bit tenha sido finalizado de enviar. Outro problema que ocorre com NRZ inclui sequências de dados contendo o mesmo número de 1's e 0's produzindo um nível DC, exigindo uma grande largura de banda.

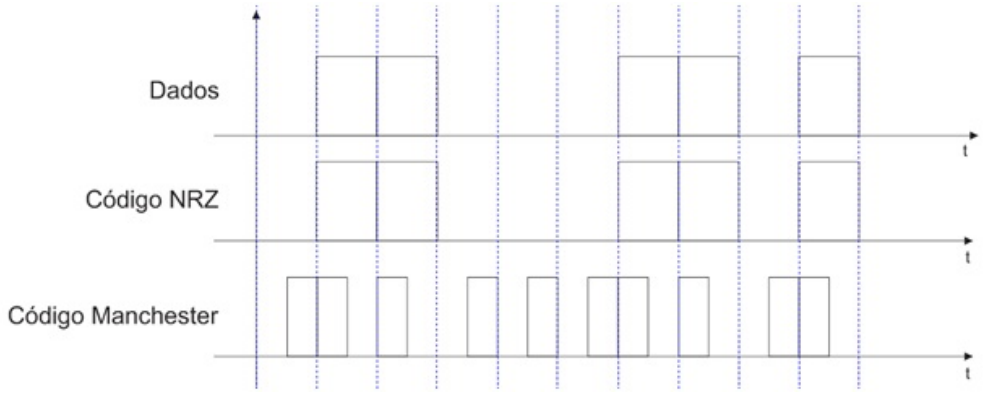

Figura 2.30: Codificação NRZ e Manchester.

2.8.2 Manchester

No código Manchester, a codificação de dados de cada bit tem pelo menos uma transição e ocupa o mesmo tempo. Não tem, portanto, nenhum componente DC (casos em que 1's e 0's são representados pelo mesmo valor, mas com magnitude diferente, tais como 3,3 V e -3,3V), e é auto-sincronizado, o que significa que pode ser induzido ou capacitivamente acoplados, e que um sinal de clock pode ser recuperado a partir dos dados codificados.

Cada bit é transmitido em tempo fixo, onde os 0's são expressos por uma baixa para alta transição e os 1's são representados por uma alta para baixa transição, segundo as especificações do protocolo. As transições que significam 0 ou 1 ocorrem no ponto médio do período, sendo assim as transições nos limites período não carregam informação, são gerais e não significam dados, sendo que existem apenas para colocar o sinal no estado correto para permitir a transição da meia-bit. A existência de transições permite que o sinal possa ser auto-sincronizado, e também permite que o receptor sincronize os dados recebidos. A codificação Manchester pode ser vista na Fig. 2.30.

Capítulo 3

Metodologia

Este capítulo tem por finalidade expor técnicas empregadas no desenvolvimento da proposta, apresentando o fluxo adotado para a modelagem do transceptor RF.

3.1 FLUXO DE MODELAGEM

Em uma ocasião em que se queira elaborar um programa, modelagem de um sistema, ou qualquer outro projeto em geral há a necessidade de se guiar por uma seqüência lógica, esta apresentada na parte esquerda da Fig. 3.2 que indica através de blocos as etapas a serem seguidas. Independentemente do projeto, algumas etapas se mantêm fixas e são aplicadas da mesma maneira, mas sendo passíveis de alteração, pulando etapas ou adicionando outras.

Para o projeto em questão, o fluxo de projeto adotado está apresentado na lateral direita da Fig. 3.2.

3.2 METODOLOGIA DA MODELAGEM DO RECEPTOR RF

A modelagem do transceptor RF envolveu a consulta e apoio de toda a equipe do Laboratório de Dispositivos e Circuitos Integrados (LDCI) da UnB que desenvolveu o circuito integrado, tanto dos que trabalharam com blocos analógicos quanto na seção digital. Isso foi necessário já que o proposto era exatamente modelar o sistema desenvolvido de forma a possibilitar a validação com o Chip em mãos e não se ter desvio de atenção à essência do trabalho, a modelagem, evitando assim se preocupar com definições de parâmetros em cada bloco.

Ferramentas CADENCE foram utilizadas para descrição na linguagem desejada em nível de programação, compilação, simulação e validação de todo o processo. Foram utilizados os softwares NC-Verilog Simulator, NCLaunch e SimVision.Em resumo, toda a atividade foi desenvolvida no LDCI no ambiente CADENCE, além do uso de editores de texto básicos. Na Fig. 3.1 é apresentado o sistema a ser desenvolvido. Pode-se ver os blocos que constituem o sistema e que devem ser modelados.

Faz-se necessário apresentar o sistema desde já para mostrar a metodologia adotada, tornando-a

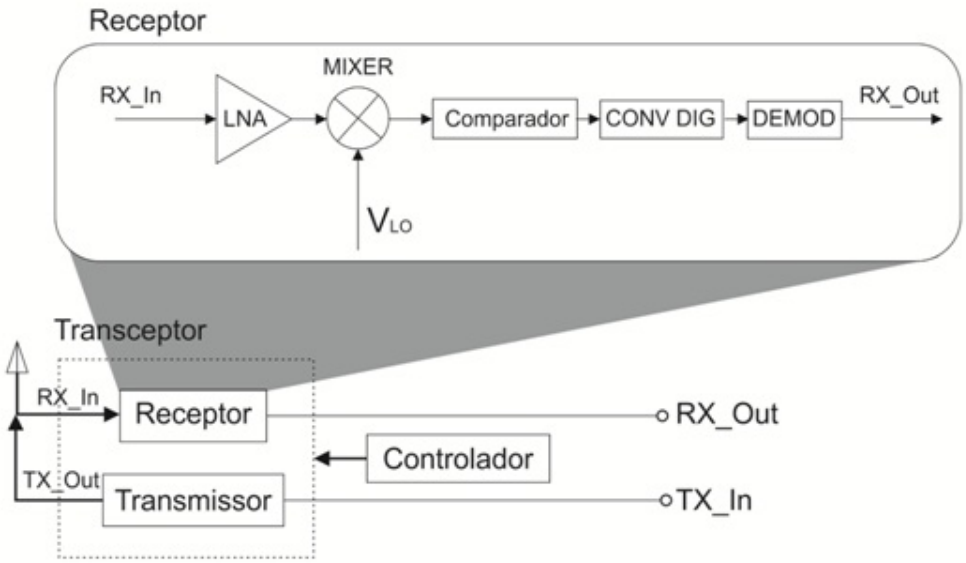

Figura 3.1: Diagrama de blocos Transceptor RF.

mais clara e fácil de ser compreendida. Atacar o problema como um todo (top-down) não seria ideal, já que cada bloco foi desenvolvido separadamente e mantém suas características individuais. Além disso, simular o bloco como um todo desde o princípio ocasionaria muitas modificações no código a cada detalhamento dos blocos. Embora essa verificação seja necessária, já que posteriormente será ela quem dará a validação do sistema completo. Desenvolver blocos separadamente é mais vantajoso, possibilitando validações desde o princípio e ajustes com blocos acoplados, que são muito menos complexos e tornam mais fácil a verificação de problemas que possam ocorrer.

Assim sendo, deu-se início ao desenvolvimento do projeto da maneira bottom-up. A cada bloco acrescentado, a validação do conjunto como um todo se fazia necessária. Portanto, realmente foi um fluxo crescente, partindo de apenas um bloco, para dois, três, ..., e finalmente o sistema completo. Nota-se portanto que essa abordagem é bottom-up e se aplica para o sistema como um todo, já que a forma de desenvolvimento de cada bloco do sistema tomou abordagens diferentes, tendo em vista os diferentes níveis de abstração, como por exemplo a forma de se desenvolver o bloco do demodulador, que já é um bloco digital e desenvolvido em VHDL, é diferente de um bloco que se parte de especificações e projeto elétrico como o LNA.

De qualquer maneira, a 3.2 representa a metodologia seguida em qualquer nível de abstração de forma genérica a direcionar a abordagem ao problema, sendo exposto separadamente na seção de projeto detalhamento metodolóico de topologias funcionais dos blocos individuais. Assim sendo, observa-se na Fig. 3.2 a metodologia adotada para desenvolvimento do projeto proposto. É possível observar no fluxograma basicamente dois fluxos, um na esquerda e outro na direita.

Vê-se que no fluxo apresentado na esquerda a metodologia básica para projeto de blocos analógicos de circuito integrado. Eventualmente algumas etapas podem ser ignoradas, refletindo a pouca relevância para o projeto em questão.

Já no esquemático da direita é apresentado o fluxo da modelagem, em específico desse projeto em que foi elaborado e adotado. Dele, tem-se as seguintes etapas:

• **Dados de entrada:** Os dados de entrada contemplam os parâmetros e as bibliotecas. As bibliotecas inseridas contemplam tanto blocos comportamentais como elementos analógicos em baixo nível. Já os parâmetros são obtidos a partir das especificações do projeto inicial.

• **Construção de um modelo do sistema:** Essa etapa consiste em elaborar a topologia funcional de como será implementada a modelagem, definindo aspectos gerais de como será a descrição de conceitos importantes levados em consideração. Na topologia é definida a funcionalidade do subsistema em análise inserindo aspectos importantes e relevantes definidos a serem implementados no modelo, organizados em blocos e interagindo entre si através de fluxos, de forma que ao final se tenha o conceito geral comportamental. Note que aqui, como dito anteriomente, a abordagem não segue de forma bottom-up, e sim top-down, de maneira que se parte da funcionalidade geral do subsistema e futuramente se desenvolve os elementos primários.

• **Construção de um modelo de teste:** Aqui é onde é realizada a descrição em código de fato, tomando como base a topologia funcional elaborada na seção anterior. Não confunda esse modelo como um modelo *para* teste, ou seja, uma testbench. O modelo aqui referenciado é um modelo preliminar, sendo uma primeira versão implementada tal que passará por testes para validação da topologia da seção anterior.

• **Simulação:** A simulação é de fato a etapa que exibirá resultados *concretos* (visuais) que irão permitir confrontar os resultados obtidos a partir do projeto desenvolvido com as especificações iniciais, indicando portando a necessidade de modificações e/ou otimização. Aqui é realizada a elaboração de um testbench que irá excitar o sistema a partir de dados de interesse e na saída serão observadas figuras de méritos de interesse.

• **Verificação de funcionalidade do modelo:** A partir dos resultados das simulações na etapa anterior se valida primeiramente a funcionalidade do sistema. O interesse aqui é verificar apenas as operações comportamentais do sistema para validação da topologia. Já na etapa seguinte, o confronto com as especificações definidas inicialmente, indicará não erros de topologia propriamente dito, mas erro na descrição dos elementos agregados à topologia funcional.

• **Confronto com especificações:** Tendo a validação da funcionalidade do sistema, nessa etapa se realiza o confronto com o sistema em maiores detalhes, verificando especificações definidas para o sistema. Caso não seja atingido, há a necessidade de otimização do sistema em um nível menor de abstração, sendo necessária a modificação na maneira de descrição do código.

• **Otimização:** Aqui deve ser realizado otimizações no modelo, para se atingir um nível maior de abstração levando em consideração aspectos cada vez mais próximos à realidade, como por exemplo, implementação de filtros que não sejam ideais, inserção de ruídos, e outros. Como dito, há a possibilidade de se otimizar a funcionalidade do sistema, por meio de uma nova proposta de topologia funcional, quanto da própria forma de descrição dos elementos que são contemplados na topologia.

• **Validação:** Na validação é onde se valida o sistema como um todo, simulando o sistema e observando tanto funcionalidade quanto especificações gerais do sistema por completo.

• **Confronto de dados da modelagem e testes do chip fabricado:** Aqui ocorrerá o confronto dos resultados obtidos a partir dos resultados obtidos do sistema já fabricado com os

obtidos a partir do modelo desenvolvido.

Note que os fluxogramas se correlacionam na etapa de simulação do esquemático da esquerda com o da direita. Entenda que o fluxograma da direita não representa a simulação do projeto elétrico, e sim mostra uma comparação dos resultados da simulação com os obtidos na modelagem, sendo mais uma forma de validação do projeto.

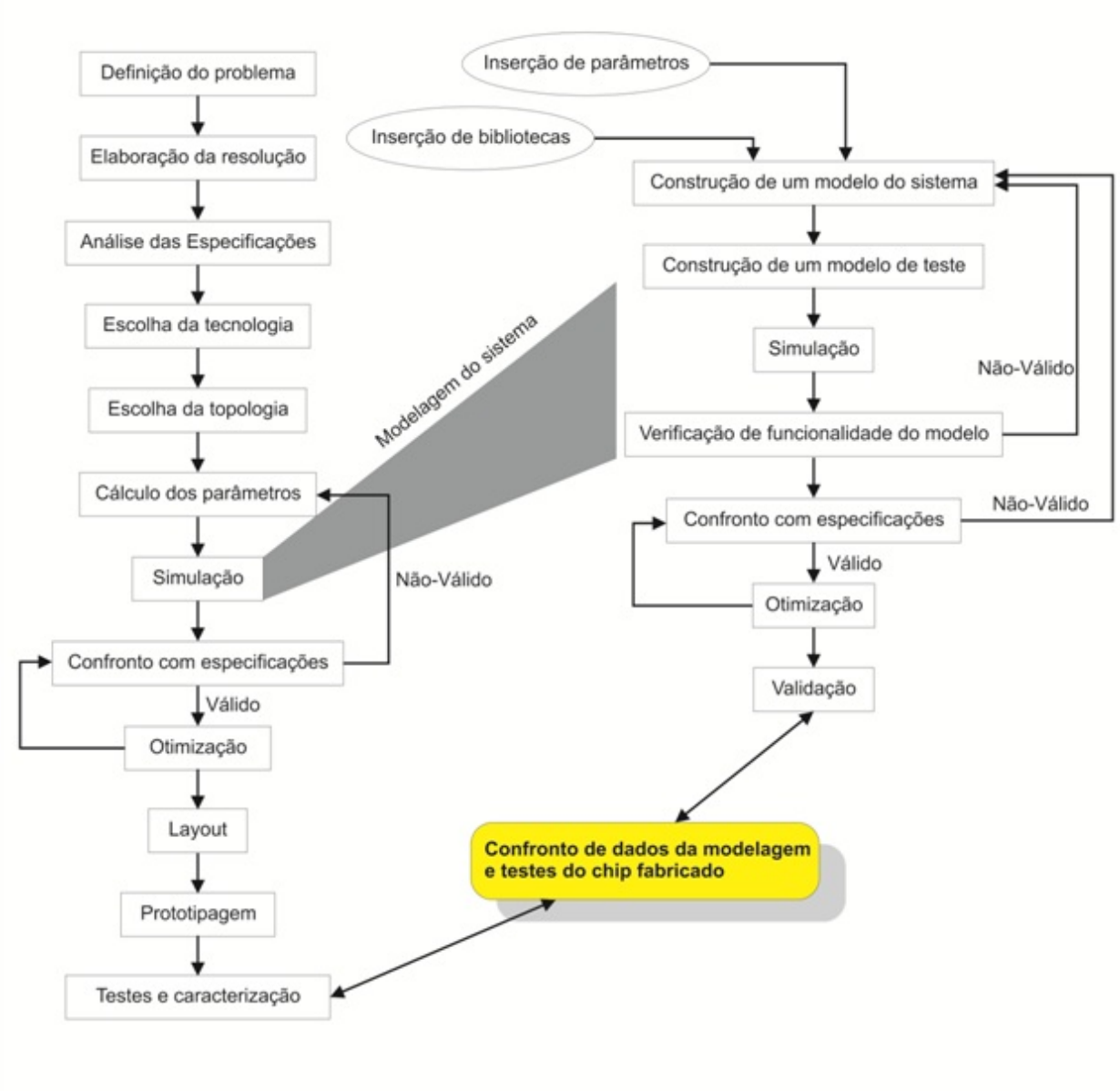

Figura 3.2: Fluxo básico de projeto.

Capítulo 4

Projeto

Este capitulo tem como objetivo apresentar a modelagem da seção de recepção do transceptor RF, assim como as especificações de cada bloco.

4.1 TRANSCEPTOR RF - RECEPTOR

A partir da Fig. 3.1 para que possamos discutir de maneira mais detalhada e crítica o sistema desenvolvido e porque tal topologia do esquemático apresentado.

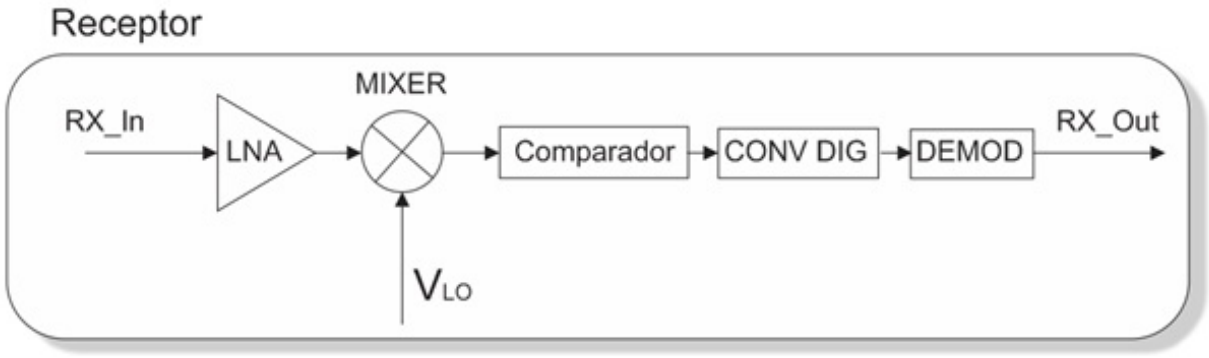

Figura 4.1: Esquemático do Receptor.

O diagrama de blocos apresentado na Fig. 4.1 foi de fato a estrutura resultante ao final do projeto e não condiz precisamente à realidade do transceptor desenvolvido no laboratório, ou seja, sua topologia original, como mostra na Fig. 4.2.

Seguindo o proposto no capítulo 3, acompanharemos a evolução da modelagem do projeto, assim como alterações e/ou omissões necessárias em certos pontos, tanto por conveniência, quanto por necessidade ou por primeira aproximação não voltando às otimizações conforme apresentado na Fig. 3.2.

Apresenta-se então inicialmente as especificações globais do transceptor RF na Tabela 4.1.

Algumas informações mais específicas não foram levadas em consideração, estas não prejudicando e nem se fazendo necessárias na modelagem em alto nível.

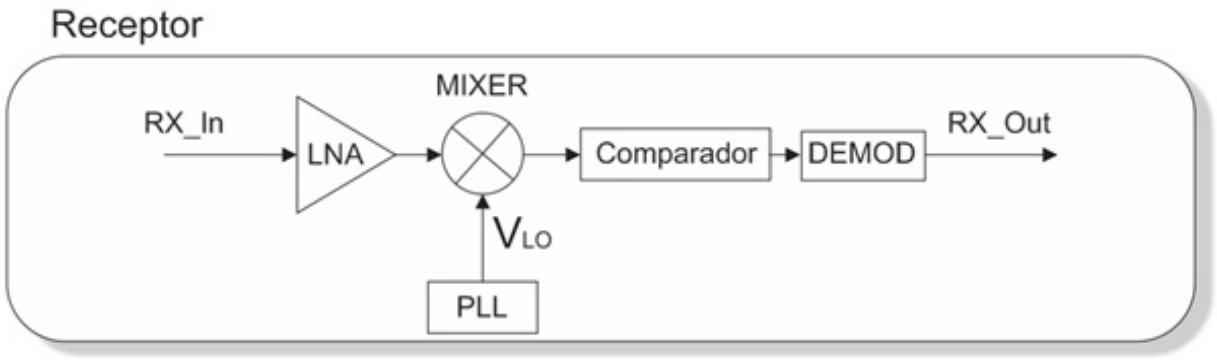

Figura 4.2: Esquemático do projeto original.

Tabela 4.1: Especificação Receptor RF

Descrição	**Especificação**
Frequência de operação	915 a 927,5 MHz
Número de canais	1
Modulação	FSK binário
Sensibilidade	-90 dBm para BER de 10^{-3}
Comunicação Half-duplex	
Os sinais devem ser preferencialmente single-ended	
Tensão de alimentação	3,3 V
Tecnologia	C35B4C3 da AMS

4.2 LNA

O primeiro passo é desenvolver o LNA, sendo este o primeiro estágio do receptor. Os parâmetros do LNA são definidos na Tabela 4.2.

A topologia adotada para a modelagem do LNA foi de acordo com o diagrama apresentado na Fig. 4.3.

Nota-se que antes do sinal de RF ($Input\_LNA$) ser processado de fato pelo LNA, ele passa por um filtro passa faixa, que vai eliminar frequências indesejáveis e que irá também facilitar o tratamento do sinal ao eliminar qualquer offset, já que a componente DC será ignorada.

Tomando-se então como 10 MHz a largura de banda do LNA, e por se desejar um filtro suave com a função de transferência com o mínimo de oscilações tanto na banda passante como na banda de corte, foi desenvolvido o filtro Butterworth.

A partir de um software [6] que realiza de maneira iterativa o projeto do filtro desejado segundo o procedimento de projeto indicado por Jon B. Hagen [13], poupando cálculos extensos, foi implementado um filtro de 5^a ordem para o LNA, tendo-se assim uma precisão satisfatória, tal como vê-se na Fig. 4.4.

As informações setadas para que fosse obtido o filtro foram:

Tabela 4.2: Parâmetros do LNA

Parâmetro	**Especificação**
Frequência de operação	921/922 MHz
Ganho de tensão	25,3 dB
Figura de ruído	1,85 dB
Largura de banda	10 MHz
IIP_3	-6 dBm
Corrente de polarização	1,5 mA
Potência consumida	4,95 mW
Impedância de entrada	50Ω

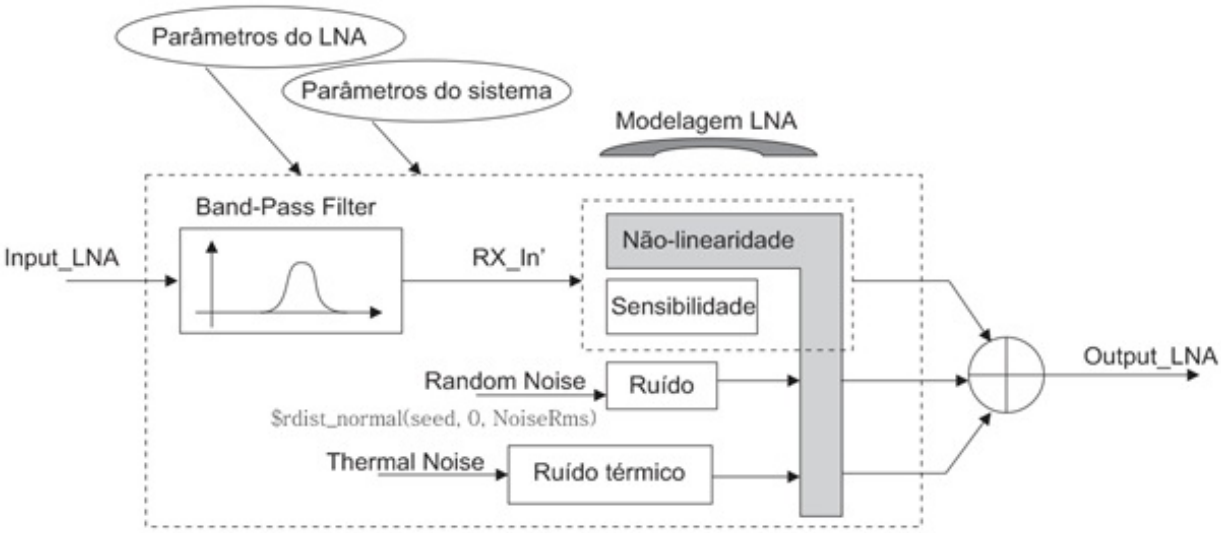

Figura 4.3: Modelagem LNA.

- Filtro Butterworth passa-faixa.
- 5^a ordem.
- Frequência baixa de corte: 915 MHz.
- Frequência alta de corte: 925 MHz.
- Característica da impedância de entrada: 50Ω
- Configuração do filtro: série

O esquemático elétrico do filtro é apresentado na Fig. 4.5.

Os valores calculados dos componentes são:

$R1 : 50\Omega$

$L1 : 0,378299 \cdot 10^{-07} H$

$C1 : 7.88523 \cdot 10^{-14} F$

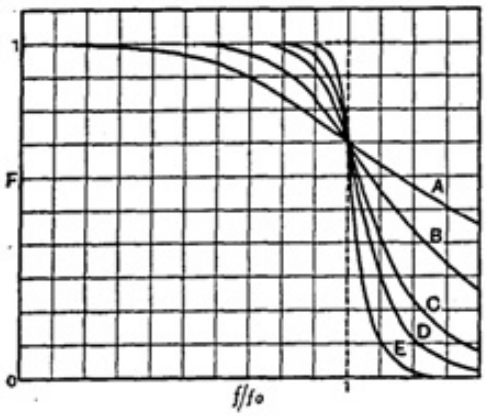

Figura 4.4: Ordens filtro Butterworth.

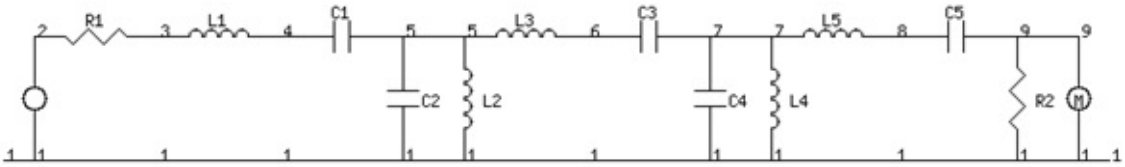

Figura 4.5: Esquemático elétrico do filtro LNA.

$C2 : 3.96173 \cdot 10^{-10} F$

$L2 : 7.52947 \cdot 10^{-11} H$

$L3 : 1.22427 \cdot 10^{-06} H$

$C3 : 2.43654 \cdot 10^{-14} F$

$C4 : 3.96173 \cdot 10^{-10} F$

$L4 : 7.52947 \cdot 10^{-11} H$

$L5 : 3.78299 \cdot 10^{-07} H$

$C5 : 7.88523 \cdot 10^{-14} F$

$R2 : 50\Omega$

Assim, a resposta em freqüência do filtro é apresentada na Fig. 4.6.

Após o sinal RF passar pelo filtro passa-banda, é inserido nele a não-linearidade e os ruídos, é observado também a sensibilidade do sistema em relação ao sinal. A não linearidade é introduzida então da seguinte maneira:

onde os parâmetros de ganho, R e IIP_3 são definidos no início do código.

Repare que a não linearidade é introduzida calculando-se a partir da distorção causada pela intermodulação de 3^a ordem de acordo com a fórmula 2.23. Calcula-se o ganho do sinal de terceira ordem:

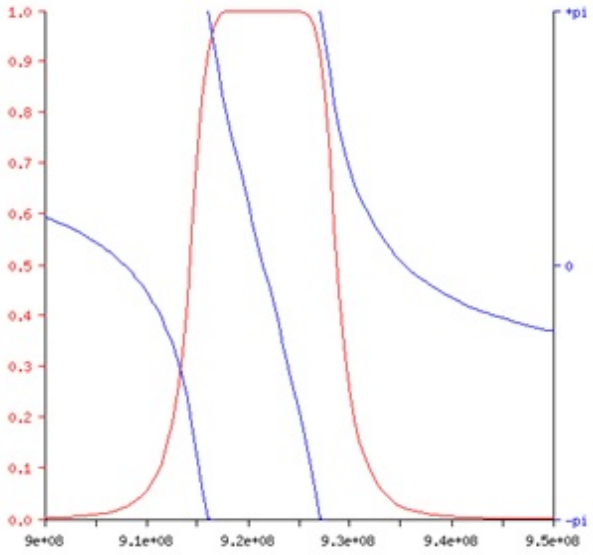

Figura 4.6: Resposta em frequência do filtro do LNA.

```
//NONLINEARITY
        Gain = 10**(Gaindb/10);
        IIP3 = (10**((IIP3dbm - 30)/10));                    //IIP3 absoluto
        IIP3v = (IIP3*R)**0.5;                               //voltage
        nonlin3 = 4*Gain/(3*IIP3v*IIP3v);                    //A3 = 4*Gain/(3*IIP3v*IIP3v);
        nonlin2 = 0;                                         //A2 = 0;
        nonlin1 = Gain;                                      //A1 = Gain;
        nonlin0 = 0;                                         //nonlin = [A3 A2 A1 0];
```

Figura 4.7: Código de inserção da não-linearidade.

$$\alpha_3 = \frac{4}{3}\frac{\alpha_1}{A_{IP3}^2} \tag{4.1}$$

Tal que $\alpha_1 = Gain$ e $A_{IP3} = IIP_{3v}$, sendo

$$IIP_{3v} = \sqrt{IIP_3 R} \tag{4.2}$$

Logo,

$$\alpha_3 = nonlinear3 = \frac{4}{3}\frac{|\,Gain\,|}{IIP_{3v}^2} \tag{4.3}$$

Já o ruído é inserido de duas maneiras: ruído térmico e ruído randômico. Vamos analisar em primeira instância o ruído térmico.

O cuidado a se tomar aqui é perceber a sutileza de que o ruído antes de ser introduzido ao sistema é submetido a uma não-linearidade, como mostrado na Fig. 4.8. Vamos analisar essa situação, retomando a partir da equação 2.35:

```
//NOISE
        NoiseFactor = 10**(NoiseFigure/10);             //noise ratio in numbers
        InputNoisems = 4*kBoltzmann*T*BW*R;             //impedance is 50 Ohms
        InputNoiserms = InputNoisems**0.5;              //rms noise voltage at input

        OutputNoiseDueToInputrms = nonlin3*InputNoiserms*InputNoiserms*InputNoiserms + nonlin1*InputNoiserms;

        OutputNoisems = NoiseFactor*((OutputNoiseDueToInputrms)**2);
        OutputNoiserms = OutputNoisems**0.5;            //Total output noise root mean square voltage
```

Figura 4.8: Código de inserção do ruído térmico.

$$NF = \frac{V_{out}^2}{\left(\sqrt{4kTBR}\right)^2 \cdot G_A} \tag{4.4}$$

Aqui pode-se verificar explicitamente a introdução da não-linearidade ao ruído, onde vamos chamar esse ruído *distorcido* de $V_{innonlinear}$ (OutputNoisems). Então,

$$V_{out}^2 = NF \cdot V_{innonlinear}^2 \tag{4.5}$$

O código de inserção do ruído randômico é apresentado na Fig. 4.9.

```
//Gerador de ruído
//              $rdist_normal ( seed , mean , standard_deviation ) ;
                @ (initial_step)
                begin
                        randseed = 1 ;
                        t1 = 0;
                        t2 = 0;
                end

                V(NoiseVoltage) <+ $rdist_normal(randseed, 0, OutputNoiserms);
```

Figura 4.9: Código de inserção do ruído randômico.

Através do código apresentado, confere-se que o ruído randômico é inserido através da função de distribuição normal, onde o desvio padrão é exatamente o valor da amplitude do sinal de ruído térmico calculado anteriormente, resultando portanto no sinal do ruído inserido ao LNA.

Ao final foram calculados os três sinais principais, e esses são somados resultando no sinal de saída do LNA, seguindo para o Mixer. Antes de analisarmos o Mixer, é apresentado a seguir, o esquemático do bloco LNA desenvolvido, na Fig. 4.10.

Verifica-se que o esquemático apresentado é realmente apenas um bloco, o próprio LNA, devido a ser um esquemático em alto nível. Ao lado esquerdo é possível visualizar os parâmetros do sistema setados no início do código, e ao lado direito as variáveis envolvidas na modelagem do LNA.

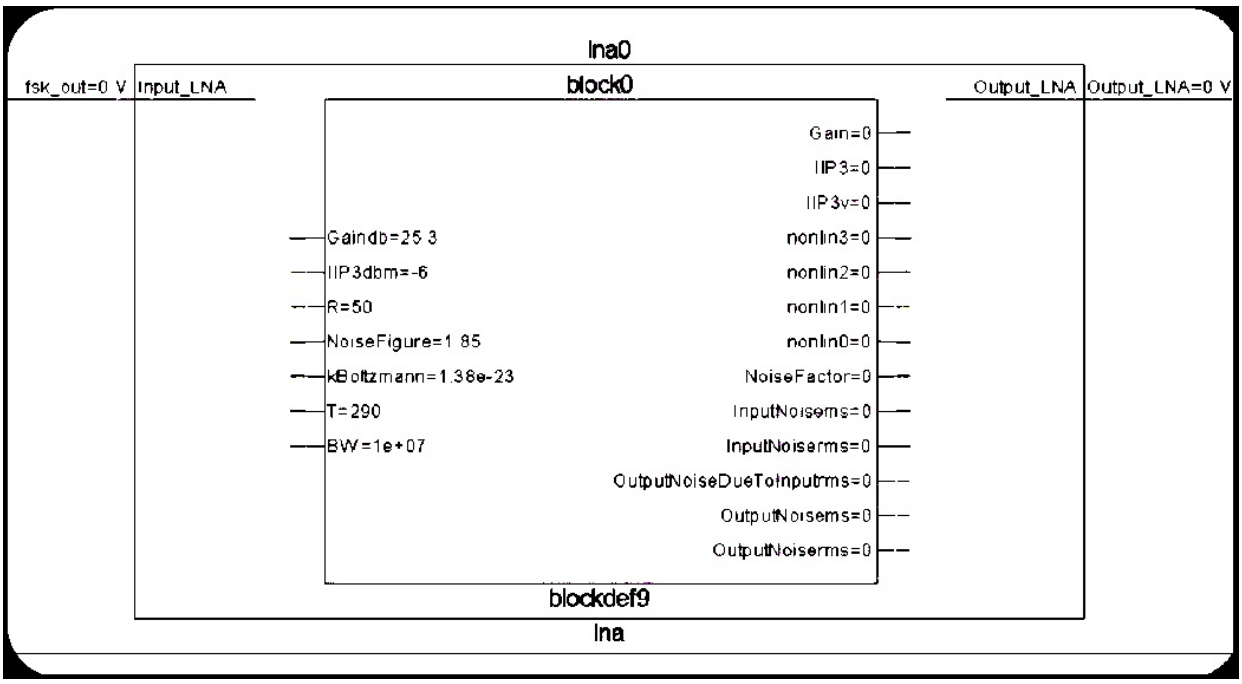

Figura 4.10: Diagrama LNA.

4.3 MIXER

O Mixer foi modelado de forma análoga ao LNA. Os parâmetros do MIXER são definidos abaixo:

Tabela 4.3: Parâmetros do Mixer

Parâmetro	**Especificação**
Frequência de operação	921/922 MHz
Ganho de tensão	15 dB
Figura de ruído	23 dB
Largura de banda	1/2 MHz
Ponto de Compressão (CP1) Cai 1 dB	-15.2 dBmV
Ponto de Compressão (CP2) Cai 70 dBm	45 dBmV
IIP_3	-6,3 dBm
Potência consumida	1,1 mW

O esquemático seguido ao se realizar a modelagem do Mixer é apresentado na Fig. 4.11

Comparando-se com o esquemático de modelagem do LNA o que difere, além dos sinais, é o filtro que em vez de estar no começo, está ao final do processo. Isso se deve ao fato de que o sinal ao sair do LNA já está na frequência adequada, adotando que os sinais de distorção não alterem significativamente a frequência do sinal.

Assim, o filtro passa-baixa inserido ao final da cadeia, tem por finalidade extrair apenas o sinal transladado, já que como foi visto na equação 2.6, tem-se:

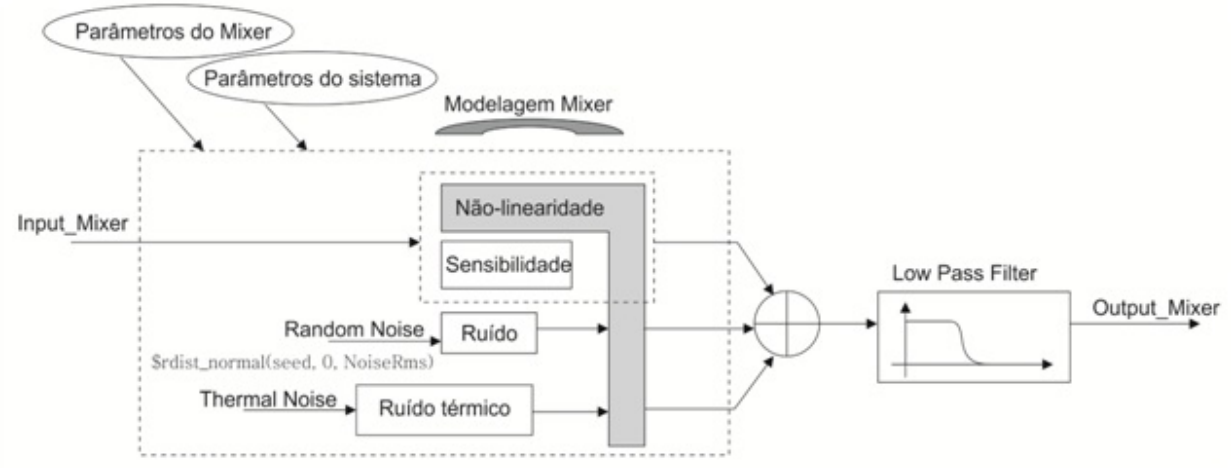

Figura 4.11: Esquemático modelo Mixer.

$$Output\_Mixer(w) \leftrightarrow \frac{1}{2}(V_{REF}(921/922MHz + 920MHz) + V_{REF}(921/922MHz - 920MHz)) \tag{4.6}$$

$$Output\_Mixer(w) \leftrightarrow \frac{1}{2}(V_{REF}(1,841/1,842GHz) + V_{REF}(1/2MHz)) \tag{4.7}$$

Na Fig. 4.2 pode-se ver que o V_{LO} é inserido pelo PLL (uma onda quadrada), mas no nosso caso por motivo de simplificação, foi adotado um oscilador, inserindo assim uma senóide.

Então, aplicando um filtro passa-baixa, retira-se o sinal em freqüência baixa, e recupera-se os dados de interesse.

O filtro por sua vez, foi calculado da mesma forma que o do LNA, inserindo-se os seguintes parâmetros:

- Filtro Butterworth passa-faixa.
- 5^a ordem.
- Frequência de corte: 5 MHz.
- Característica da impedância de entrada: 50Ω
- Configuração do filtro: série

O esquemático elétrico do filtro é apresentado abaixo:

Os valores dos componentes calculados são:

$R1 : 50\Omega$

$L1 : 6.14736 \cdot 10^{-07} H$

$C1 : 6.43782 \cdot 10^{-10} F$

$L2 : 1.98944 \cdot 10^{-06} H$

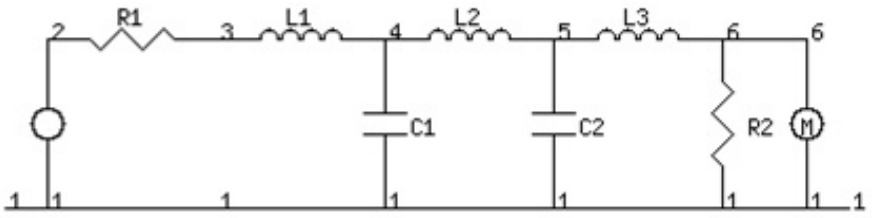

Figura 4.12: Esquemático elétrico do filtro do Mixer.

$C2 : 6.43782 \cdot 10^{-10} F$

$L3 : 6.14736 \cdot 10^{-07} H$

$R2 : 50\Omega$

A resposta em freqüência do filtro calculado é apresentada na Fig. 4.13.

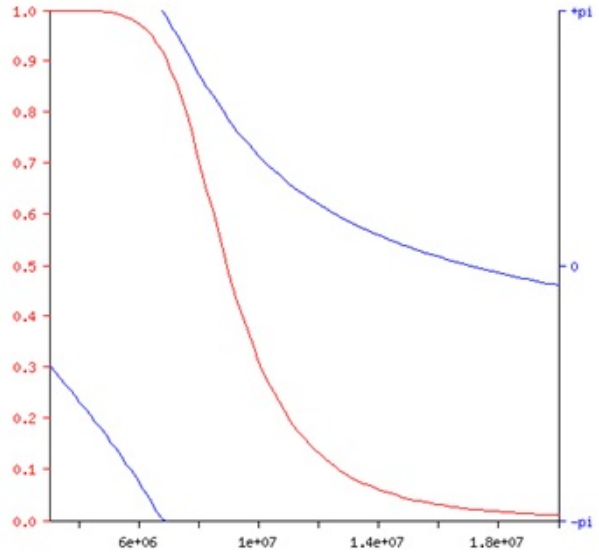

Figura 4.13: Resposta em frequência do filtro do Mixer.

Quanto à forma de inserção dos sinais (sinal de entrada e sinais de ruído) do Mixer segue a mesma metodologia apresentada ao LNA. O esquemático do Mixer desenvolvido é apresentado na Fig. 4.14.

Novamente vê-se ao lado esquerdo os parâmetros do sistema setados no início do código, e ao lado direito as variáveis envolvidas na modelagem do Mixer.

4.4 COMPARADOR

O comparador tem como finalidade ser um estágio de comparação, conforme o nome já diz, servindo portanto de um conversor A/D, já que irá digitalizar o sinal de entrada, uma senóide, em um trem de bits. Esse conversor A/D de 1 bit implementado é suficiente, visto que o sinal foi modulado em FSK binário, transportando portanto os dados na frequências, desprezando a

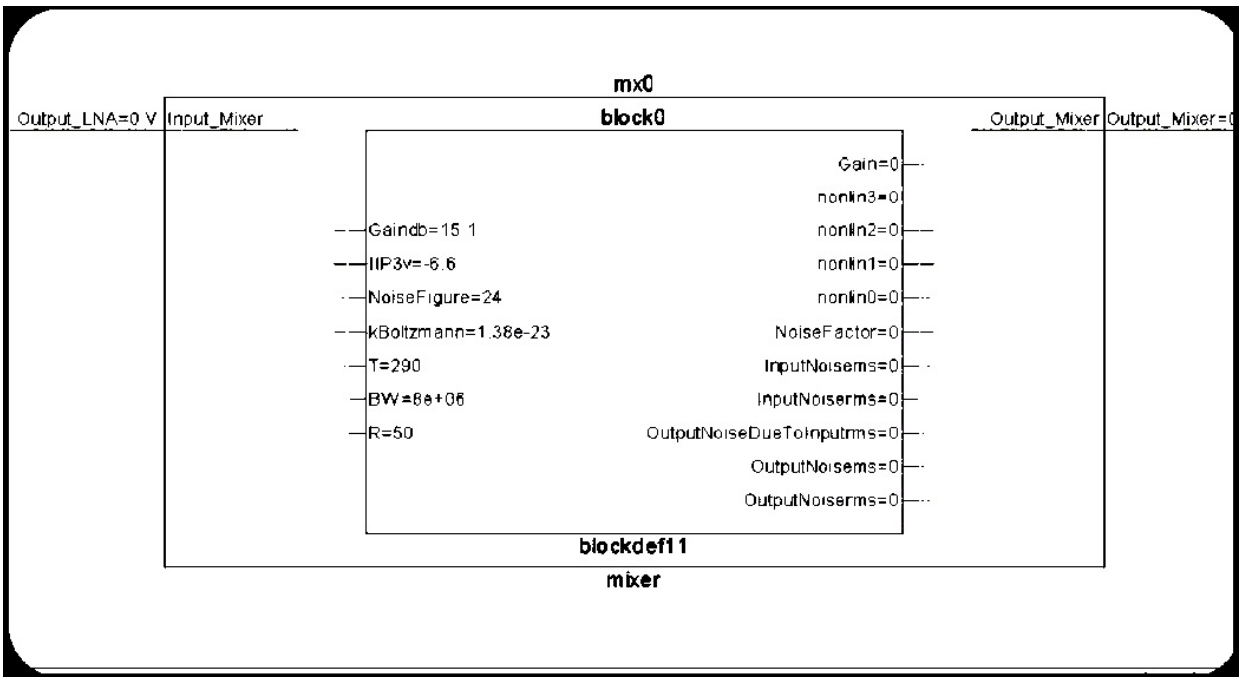

Figura 4.14: Esquemático Mixer.

amplitude. Assim o comparador clipa a amplitude em 3,3 V para diferenças positivas e 0 V para diferenças negativas, nesse caso a comparação é referenciada ao ground.

A implementação dessa lógica é bem simples, sendo através de sentenças condicionais:

```
if ((V(Input_Comparador) > -35e-6) && (V(Input_Comparador) < 35e-6)) begin
        V(Sens_Comparador) <+ vth;
end

else if ((V(Input_Comparador) < -35e-6) || (V(Input_Comparador) > 35e-6)) begin
        V(Sens_Comparador) <+ V(Input_Comparador);
end
```

Figura 4.15: Implementação código comparador.

Note que no código sinais entre $-35\ \mu V$ e $35\ \mu V$ não são notados pelo comparador. Isso se deve ao fato de que em simulação DC do comparador real, constatou-se um ganho de 10^5, ou seja,

$$Ganho = 10^5 = \frac{3,3V}{min} \tag{4.8}$$

$$min = 30\ \mu V \tag{4.9}$$

Logo, essa sensibilidade é setada na própria condicional do código.

O comparador descrito finaliza por aqui. De fato não há um detalhamento mais crítico, sendo que não foi otimizado. Para funcionamento do sistema como um todo isso basta, mas implicações

de um comparador simplificado e quase ideal serão discutidas mais adiante.

4.5 CONVERSOR DIGITAL

Nessa etapa é onde ocorre uma das maiores dificuldades do projeto. A solução é simples, mas a idéia envolvida não é tão trivial. O problema surge quando se chega na etapa de se integrar os blocos funcionando com o demodulador. Se posicione da seguinte forma:

• 1.A seção $LNA \rightarrow MIXER \rightarrow Comparador$ está pronta, testada e validada.

• 2.O demodulador foi inteiramente desenvolvido pela equipe digital e processa somente dados digitais. Bloco também simulado e validado.

• 3.A etapa seguinte é acoplar o sinal de saída do comparador ao demodulador.

Aparentemente à primeira vista não há com o que se preocupar, mas veja que apesar de o comparador digitalizar a entrada, ele não digitaliza o sinal de fato. O que ocorre é que ele apenas clipa o sinal de entrada em 3,3 V e 0 V, sendo assim um sinal elétrico, por mais que signifique 1's e 0's. Agora analisando do lado do demodulador, este somente irá aceitar sinais digitais, ou seja, significativos 1's e 0's, declarados como *std_logic*. E ainda mais, o demodulador é descrito em linguagem VDHL, diferenciando-se da família do Verilog, complicando ainda mais a integração entre ambos. Uma solução para o problema é realizar através das ferramentas CADENCE um envelope, onde este irá criar uma espécie de conversor para que estes blocos possam se comunicar e aí sim realizar a co-simulação.

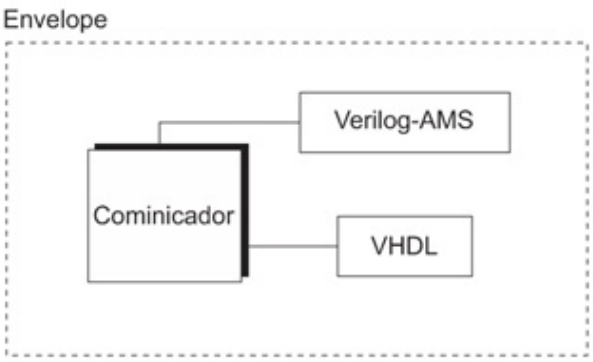

Figura 4.16: Envelope para co-simulação.

A questão é que realizar essa solução não é nada trivial, envolvendo noções complexas da ferramenta CADENCE. A segunda solução ao problema, e sendo uma forma mais extensa e até mesmo mais completa, é apresentada a seguir.

Na figura acima pode-se ver como realizar o comunicador de maneira manual a mais completa. Do código VHDL realiza-se a síntese lógica, também através da ferramenta CADENCE, ou seja, transforma-se a descrição em nível RTL em uma netlist de portas lógicas através da otimização, que gera um hardware genérico, e do mapeamento tecnológico, no qual o hardware é mapeado para células disponíveis na biblioteca. Os objetivos são otimizar a área, o atraso e o consumo utilizando constraints, inserindo parâmetros de atraso, capacitância, indutância, ou seja, informações sobre

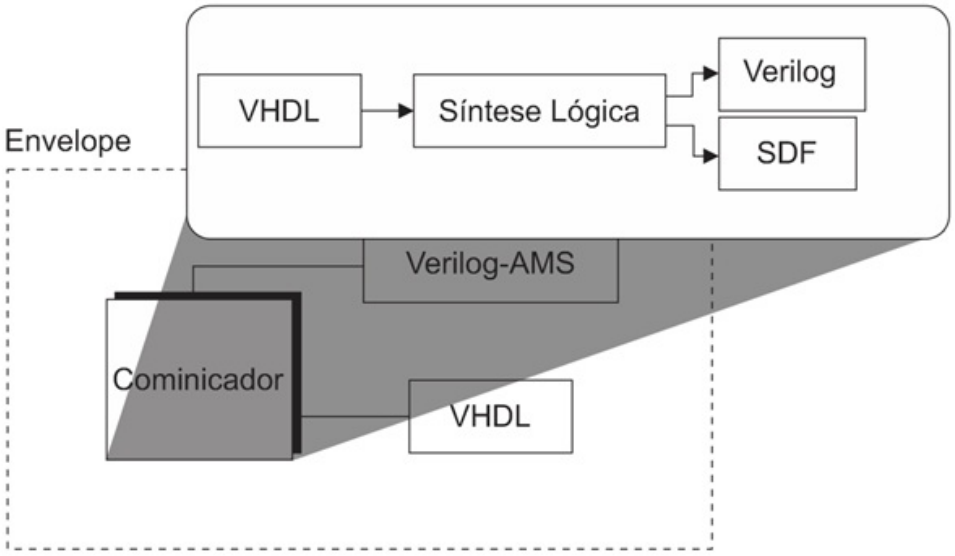

Figura 4.17: Integração Verilog-AMS/VHDL.

parasitários e imperfeições introduzidas na realidade ao sistema ao ser construído.

A partir da síntese lógica é gerado um arquivo em linguagem Verilog e outro arquivo contendo as informações de não idealidades, este chamado de SDF. Assim, tendo-se arquivos em Verilog-AMS e Verilog, já é passível de comunicação, tendo em vista que pertencem à mesma família de linguagem. Porém, não esqueça que além do problema de comunicação de linguagem, há o problema de que a saída do comparador é um sinal elétrico e o demodulador processa apenas sinais puramente digitais. Assim sendo, se faz necessária a implementação de um bloco que realize a conversão de analógico para digital.

Mesmo tendo em vista que a linguagem Verilog-AMS trabalha com os dois sinais (analógico e digital), há a real necessidade de se implementar um estágio extra já que apesar do Verilog-AMS permitir trabalhar com sinais mistos, ela é geralmente restrita a não permitir ações de sinais digitais na seção analógica e vice-versa, ou seja, há uma certa forma de separação no tratamento dos sinais. Em análise mais crítica vê-se que isso não é totalmente verdade, já que operações mistas podem ocorrer, mas para sinais declarados como entrada e saída, isso é de fato restrito.

Enfim, é necessária a criação de um bloco para realizar a conversão digital, assim como um clock para que haja essa digitalização. De forma a esse estágio ser apenas um contorno a um limitação da linguagem, ele foi construído como ideal, inclusive tendo um clock absurdamente grande e não plausível.

O código que descreve esse bloco é apresentado na Fig. 4.18.

onde o Sample é a amostragem (*clock*), e a saída $Output\_ConvDig$ é do tipo registrador, que amostra e digitaliza do sinal de entrada $Input\_ConvDig$, sendo na verdade o ($Output\_Comparador$).

Como dito, para uma escala de tempo de 1 ps, o *Sample* gerado é de:

$$Sample = \frac{1}{10^{-12}} \tag{4.10}$$

```
module convdig(Sample, Input_ConvDig, Output_ConvDig);
        reg Output_ConvDig;
        electrical Input_ConvDig;                          //Sinal de entrada
        always @(Sample) begin

                a_smp = V(Input_ConvDig);

                if (a_smp > 0.5) begin
                        Output_ConvDig = 1;
                end
                else begin
                        if (a_smp < 0.5) begin
                                Output_ConvDig = 0;
                        end
                end


        end
```

Figura 4.18: Código de inserção do Conversor Digital.

$$Sample \simeq 1\ THZ \tag{4.11}$$

Ressalta-se portanto que esse tempo de processamento não tem significado nenhum explícito, já que esse bloco foi uma necessidade de se implementar para contornar um problema de co-simulação entre os blocos analógicos e digitais. Em suma, na realidade não seria utilizado, e nem se faz necessário, portanto ao inserir esse bloco, deve-se ter o menor erro possível para não interferir no resultado do sistema, sendo assim se escolhendo a maior amostra possível na escala de tempo determinada.

4.6 DEMODULADOR

O demodulador por sua vez foi desenvolvido inteiramente por um integrante da equipe digital do LDCI. O interessante e necessário saber do demodulador é que ele opera da seguinte maneira:

Sinais para o Demodulador:

Desligar:

Enable: "0000"

Funcionamento:

- **operação como demodulador:**
  - Enable = "0101";
- **modo de operação 8-bits:**
  - Mode = "00";

- **Bloco conectado:**
  - Isolate = "0";
- **Frequência de Clock:**
  - 20 MHz, na entrada Clk;
- **Reset inicial:**
  - reset = '0', depois reset = '1';
- **Entrada que se conecta ao comparador:** RxIN
- **Saída digital de dados:** RxOUT

Foi testado as duas formas de operação, com codificação NRZ e Manchester. Algumas especificações também são importantes:

- Baudrate: 1,22 - 50 kbps
- Faixa FSK:
  - '0': 600 - 1200 kHz
  - '1': 1500 - 2100 kHz
- Clock 20 MHz
- Funções
  - Controle
  - Demodulação
  - Decodificação Manchester
  - Decodificação NRZ

Capítulo 5

Análise do Receptor

Este capítulo tem por objetivo apresentar e discutir resultados obtidos a partir de simulações realizadas da modelagem dos blocos de acordo com o exposto na seção anterior.

5.1 INTRODUÇÃO

A fim de realizar a validação do sistema desenvolvido no capítulo 4, foi necessário descrever um bloco capaz de excitar o sistema com dados de maneira que estes dados pudessem ser controlados e variados de acordo com a necessidade de cada simulação.

O bloco de teste proposto é apresentado abaixo na Fig. 5.1.

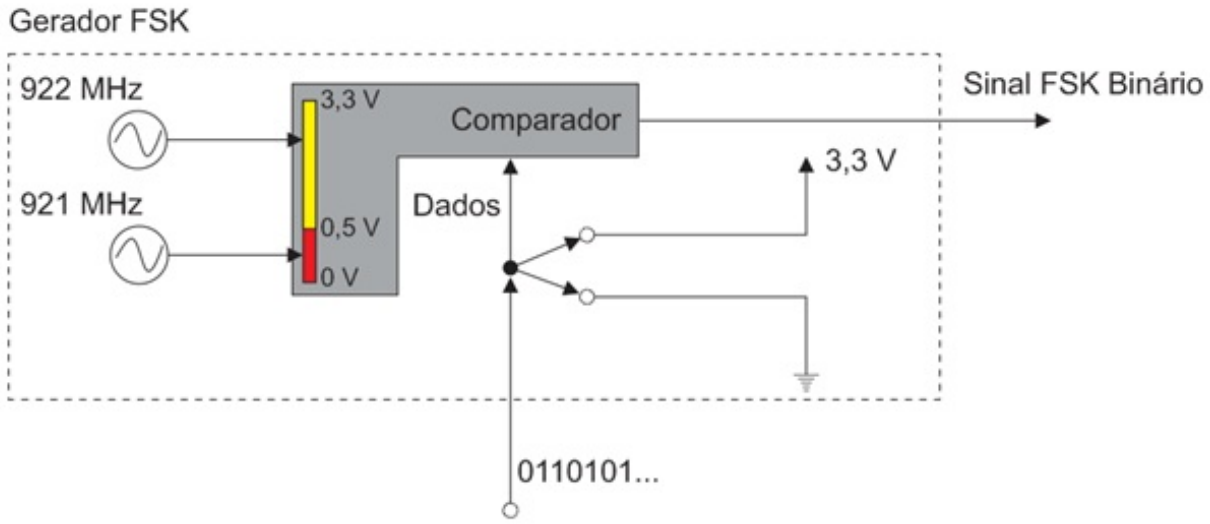

Figura 5.1: Gerador FSK.

Os dados inseridos são apenas bits 0's e 1's. Vamos chamar o bloco de Gerador FSK já que irá modular o dado a ser transmitido em FSK binário. Assim é usado apenas um comparador internamente e dois osciladores nas freqüências que o sinal será enviado. Note também que a princípio não há codificação, sendo o sinal codificado então em NRZ. O código do gerador é mostrado na Fig. 5.2.

```
if (V(fsk_in) >= 0.5) begin
        V(fsk_out) <+ ampl * sin(2 * 3.14 * freq2 * $abstime) + offset;
        $bound_step(0.05/freq1);
end

else if (V(fsk_in) < 0.5) begin
        V(fsk_out) <+ ampl * sin(2 * 3.14 * freq1 * $abstime) + offset;
        $bound_step(0.05/freq2);
end
```

Figura 5.2: Código gerador FSK.

A partir desse ponto, temos então uma ferramenta que nos possibilita excitar o sistema e verificar através de simulações o seu funcionamento.

É importante notar desde já que as simulações executadas seguem a metodologia apresentada no capítulo 3 e conforme é indicado na Fig. 5.3. Nela pode-se observar que o único bloco a ser analisado de fato individualmente é o gerador FSK. Quanto ao resto dos blocos, estes são verificados de tal forma que o sinal de entrada de um estágio é o sinal de saída do anterior, e conseqüentemente o sinal de saída desse mesmo estágio é o sinal de entrada do próximo.

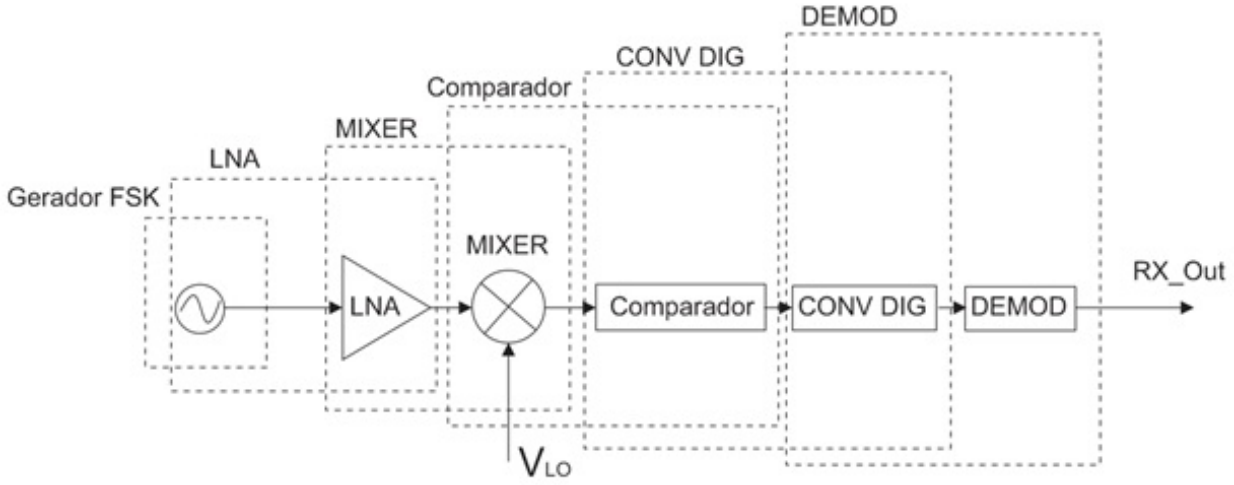

Figura 5.3: Fluxo de simulação.

5.2 ANÁLISE DO GERADOR FSK

Para teste, aplica-se à entrada do sistema os seguintes dados: 10011010101, mas não somente isso. Define-se também o tempo de transição de cada bit (vide Fig. 5.4).

É verdade que os bits não têm duração padrão, o que não interessa no momento já que a codificação é NRZ, interessando portanto apenas os dados 0's e 1's sem se preocupar com o período de bits para decodificação.

Esses tempos serão necessários e interessantes para verificação e comparação na saída do re-

```
// Sinal de entrada (1001101)
                #2000000 signal = 1;    // Em t=2us
                #10000000 signal = 0;   // Em t=12us
                #10000000 signal = 0;
                #10000000 signal = 1;   // Em t=32us
                #10000000 signal = 1;
                #10000000 signal = 0;   // Em t=52us
                #10000000 signal = 1;   // Em t=62us
                #5000000 signal = 0;    // Em t=67us
                #5000000 signal = 1;    // Em t=72us
                #5000000 signal = 0;    // Em t=77us
                #5000000 signal = 1;    // Em t=82us
```

Figura 5.4: Inserção de bits na entrada.

ceptor, facilitando a verificação da saída e quantizando atrasos.

5.3 ANÁLISE DO LNA

Tendo em mente os parâmetros do LNA, definidos na Tab. 4.2, vamos calcular os resultados esperados para um sinal de entrada -40 dBm.

$$V_{amp} = \sqrt{R \cdot \frac{10^{P_{in}/10}}{1000}} \tag{5.1}$$

onde $R = 50\ \Omega$ e P_{in} são os próprios -40 dBm. Logo,

$$V_{amp} = 2,23607mV \tag{5.2}$$

Então, esse deve ser o sinal a entrar no LNA. Já o sinal de intermodulação de 3^a ordem, é calculado abaixo:

$$Sinal_{nonlinear} = \alpha_3 \cdot Input_{LNA}^3 + \alpha_1 \cdot Input_{LNA} \tag{5.3}$$

$$Sinal_{nonlinear} = \frac{4}{3}\frac{\mid Gain \mid}{(IIP_{3v})^2} \cdot Input_{LNA}^3 + Gain \cdot Input_{LNA} \tag{5.4}$$

$$Sinal_{nonlinear} = \frac{4}{3}\frac{\mid Gain \mid}{IIP_3 \cdot R} \cdot Input_{LNA}^3 + Gain \cdot Input_{LNA} \tag{5.5}$$

$$Sinal_{nonlinear} = \frac{4}{3}\frac{\mid 10^{2,53} \mid}{10^{(-6-30)/10} \cdot 50} \cdot Input_{LNA}^3 + 10^{2,53} \cdot Input_{LNA} \tag{5.6}$$

$$Sinal_{nonlinear} = 35972,3 \cdot Input_{LNA}^3 + 338,844 \cdot Input_{LNA} \tag{5.7}$$

Ou seja, para uma amplitude de 2,23607 mV, tem-se então as seguintes magnitudes dos sinais:

$$Sinal_{nonlinear} = 4,02 \cdot 10^{-4}[Input_{LNA}^3] + 0,76[Input_{LNA}] \tag{5.8}$$

Já para o ruído, tem-se

$$NF = \frac{V_{out}^2}{\left(\sqrt{4kTBR}\right)^2 \cdot G_A} \tag{5.9}$$

$$NF = \frac{V_{out}^2}{\alpha_3(4 \cdot 1,38 \cdot 10^{-23} \cdot 290 \cdot 10^6)^3 + \alpha_1(4 \cdot 1,38 \cdot 10^{-23} \cdot 290 \cdot 10^6)} \tag{5.10}$$

onde,

$$NF = \frac{10^{NoiseFigure}}{10} = \frac{10^{1,85}}{10} = 1,53109 \tag{5.11}$$

Logo,

$$V_{out} = \sqrt{1,53109 \cdot [\alpha_3(4 \cdot 1,38 \cdot 10^{-23} \cdot 290 \cdot 10^6)^3 + \alpha_1(4 \cdot 1,38 \cdot 10^{-23} \cdot 290 \cdot 10^6)]} \tag{5.12}$$

Tal que $\alpha_1 = 338,844$ e $\alpha_3 = 35972,3$.

Então, a amplitude do ruído térmico será:

$$V_{out} = 0,00118619 \tag{5.13}$$

Calculados os valores de interesse, vamos analisar a simulação do LNA.

Na simulação apresentada, podemos conferir o sinal de entrada ($Input\_LNA$), o sinal de intermodulação de 3^a ordem ($GainIP3$), o ruído randômico ($NoiseVoltage$), o sinal de entrada já com as características não lineares (NonLinearLNA), o ruído térmico ($OutputNoiserms$) e finalmente a saída do LNA ($Output\_LNA$).

Do gráfico, nota-se que o sinal de intermodulação de 3^a ordem apresenta sua característica de uma senóide de terceira ordem, como esperado. Já o sinal não linear aparenta ser uma senóide pelo fato de a amplitude do sinal de 3^a ordem ser bem inferior ao de 1^a ordem, mas é na verdade a soma dos dois sinais, como indicado anteriormente. Já o ruído randômico tem desvio padrão do valor do ruído térmico.

Conferindo os valores esperados do LNA pelos valores observados na simulação vê-se o seguinte:

- $Input\_LNA$: Amplitude de 0,00223585 V

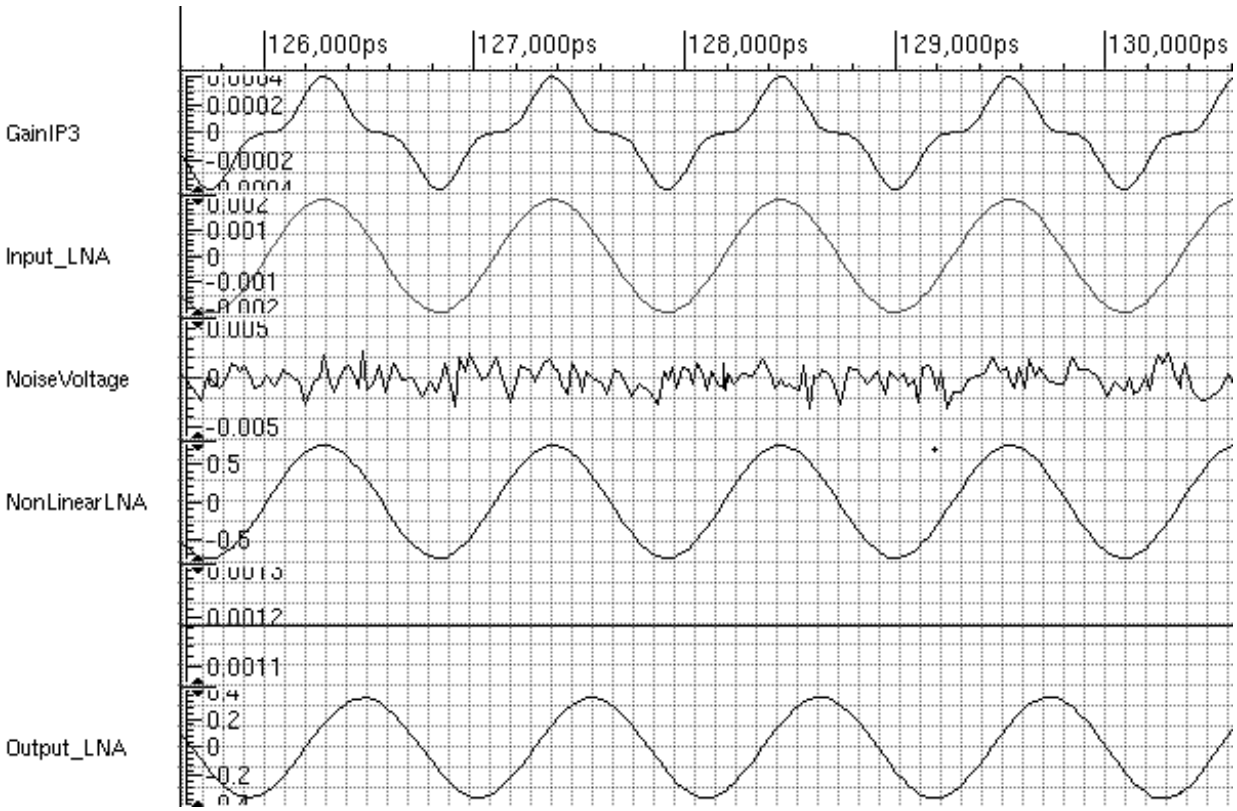

Figura 5.5: Simulação LNA.

- $Output\_LNA$: 0,366292 V
- GainIP3: 0,000402065 V
- NonLinearLNA: 0,758007 V
- OutputNoiserms: $1,40704 \cdot 10^{-6}$ V

Tabela 5.1: Resultados Obtidos - LNA.

Sinal	**Valor Teórico**	**Valor Obtido**
Ganho	338,844	163,8267
$Input\_LNA$	2,23607 mV	0,00223585 V
GainIP3	$4,02 \cdot 10^{-4}$	0,000402065 V
NonLinear LNA	$4,02 \cdot 10^{-4}[Input_{LNA}^3] + 0,76[Input_{LNA}]$	0,758007 V
OutputNoiserms	0,00118619 V	0,00118619 V

Vê-se que os resultados obtidos foram bastante coerentes com os calculados na teoria, salvo o ganho. O que acontece é que no ganho foi introduzido não linearidade com o fator de IP_3 e ruído. De fato, o ruído não afeta tanto a diminuição do ganho por ter uma magnitude pequena. Mas ao retornarmos a Eq. 5.5, vemos que o IP_3 afeta diretamente na magnitude do sinal de 3^a ordem. Com a diferença de fase em relação ao sinal original, isso acaba degradando o sinal original, implicando na diminuição do ganho.

5.4 ANÁLISE DO MIXER

Retomando os parâmetros do Mixer definidos na Tab. 4.3, vamos calcular de forma análoga ao LNA os resultados esperados para um sinal de entrada proveniente do LNA.

Tem-se que para um sinal de entrada proveniente do LNA de 0,366292 V, então o sinal de intermodulação de 3^a ordem, é calculado abaixo:

$$Sinal_{nonlinear} = \frac{4}{3}\frac{\mid Gain \mid}{IIP_3 \cdot R} \cdot Input_{LNA}^3 + Gain \cdot Input_{LNA} \tag{5.14}$$

$$Sinal_{nonlinear} = \frac{4}{3}\frac{\mid 10^{1,51} \mid}{10^{(-6,3-30)/10} \cdot 50} \cdot Input_{Mixer}^3 + 10^{1,51} \cdot Input_{Mixer} \tag{5.15}$$

$$Sinal_{nonlinear} = 0,990492 \cdot Input_{Mixer}^3 + 32,3594 \cdot Input_{Mixer} \tag{5.16}$$

Já para o ruído, tem-se

$$NF = \frac{V_{out}^2}{\alpha_3(4 \cdot 1,38 \cdot 10^{-23} \cdot 290 \cdot 10^6)^3 + \alpha_1(4 \cdot 1,38 \cdot 10^{-23} \cdot 290 \cdot 10^6)} \tag{5.17}$$

onde,

$$NF = \frac{10^{NoiseFigure}}{10} = \frac{10^{23}}{10} = 251,189 \tag{5.18}$$

Logo,

$$V_{out} = \sqrt{251,189 \cdot [\alpha_3(4 \cdot 1,38 \cdot 10^{-23} \cdot 290 \cdot 10^6)^3 + \alpha_1(4 \cdot 1,38 \cdot 10^{-23} \cdot 290 \cdot 10^6)]} \tag{5.19}$$

Tal que $\alpha_1 = 32,3594 e \alpha_3 = 0,990492$.

Então, a amplitude do ruído térmico será:

$$V_{out} = 0,00129777 \tag{5.20}$$

Aqui é mais difícil calcular os valores esperados de cada sinal. No LNA como era apenas um sinal, pudemos realizar análises gerais através de aproximações, como para o valor esperado de não-linearidade. Mas no Mixer, note que agora estamos tratando com dois sinais distintos, já que o sinal de entrada de 921/922 MHz foi transladado tanto para um lado quanto para outro em 900 MHz. Assim fica uma forma bem complicada de admitir qual seria o sinal esperado na não linearidade, elevando o sinal à terceira ordem, não sendo apenas elevando a amplitude ao cubo e multiplicando a α_3.

Sabendo as operações que devem ocorrer no Mixer, vamos analisar a simulação.

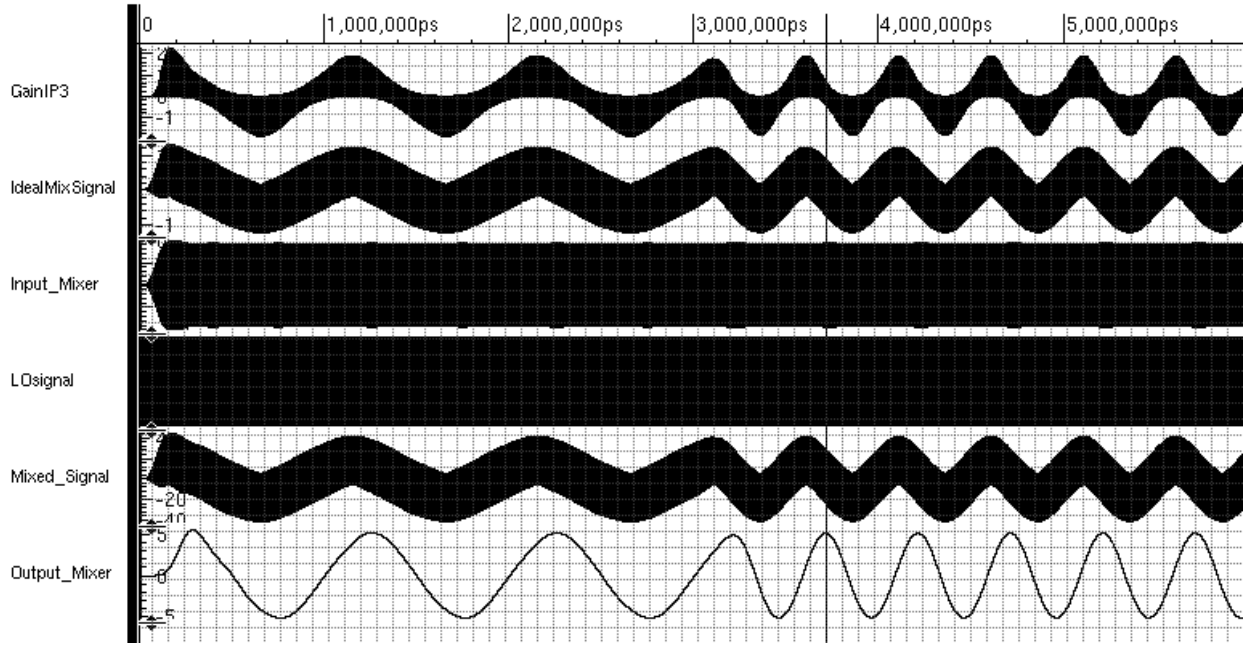

Figura 5.6: Simulação Mixer.

Na simulação apresentada acima pode-se ver o sinal de entrada do Mixer ($Input\_Mixer$), o sinal LO ($LOsignal$), o sinal de intermodulação de 3^a ordem ($GainIP3$), mostrando apenas o termo de 3^a ordem, o ruído randômico ($NoiseVoltage$), o sinal de entrada já com as características não lineares ($NonLinearMixer$), o ruído térmico ($OutputNoiserms$) e finalmente a saída do Mixer ($Output\_Mixer$).

A escala de visualização na Fig. 5.6 é grande justamente para permitir a análise dos sinais em frequência menor.

• 1. O $Input\_Mixer$ está em freqüência de 921/922 MHz.

• 2. O $LOsignal$ está em frequência de 920 MHz.

• 3. O $GainIP3$ é calculado a partir da terceira ordem do $IdealMixSignal$, sinal este que é a multiplicação do sinal de entrada pelo $LOsignal$. Note que no $IdelMixSignal$ aparece as duas frequências à que a entrada foi transladada, a de 1/2 MHz, visível no gráfico, e de 1,841/1,842 MHz, também implícita na imagem.

• 4. O sinal de ruído, $NoiseVoltage$ tem desvio padrão do valor do ruído térmico calculado.

• 5. O $NonLinearMixer$ é o sinal de intermodulação de 3^a ordem completo, ou seja, com os termos de 1^a e 3^a ordem somados.

• 6. O $Mixed\_Signal$ apresenta o sinal do Mixer somado da não linearidade e do ruído.

• 7. A saída do Mixer, $Output\_Mixer$ é o sinal em baixa frequência após ter passado pelo filtro passa-baixa.

Nessa segunda simulação pode-se ver os sinais em uma escala maior. Aqui notamos o comportamento do termo de 3^a ordem do sinal de intermodulação, assim como a distorção causada do sinal do Mixer. Vemos também que o filtro suaviza essas distorções ao capturar apenas a frequência

Figura 5.7: Simulação Mixer em escala ampliada.

mais baixa.

Das Figs. 5.6 e 5.7 pode-se retirar o ganho do Mixer:

$$Ganho = \frac{5,31}{0,366292} = 11,21dB \tag{5.21}$$

O ganho especificado foi de $15,1dB$, e o ganho calculado foi de $11,21dB$. Pode-se notar que ocorreu uma queda no ganho.

5.5 ANÁLISE DO COMPARADOR

Recebendo o sinal do Mixer, já em frequência baixa, o comparador clipa o sinal em 3,3 V e 0 V para valores positivos e negativos respectivamente conforme pode ser visualizado na Fig. 5.8.

Na figura nota-se exatamente esse processamento do comparador. A frequência do sinal entregue ao próximo estágio é de $999,133kHz$ para bits 0 e $1,9975MHz$ para bits 1. Note que é bastante próximo às frequências baixas de operação: $1MHz$ e $2MHz$.

Tirando o fato de que foi implementada sensibilidade do comparador em relação apenas a sinais de pouca amplitude, pode-ser verificar que o esse estágio, por ser praticamente ideal, faz com que qualquer perturbação no sinal proveniente do Mixer ao redor do zero, faz com que o comparado

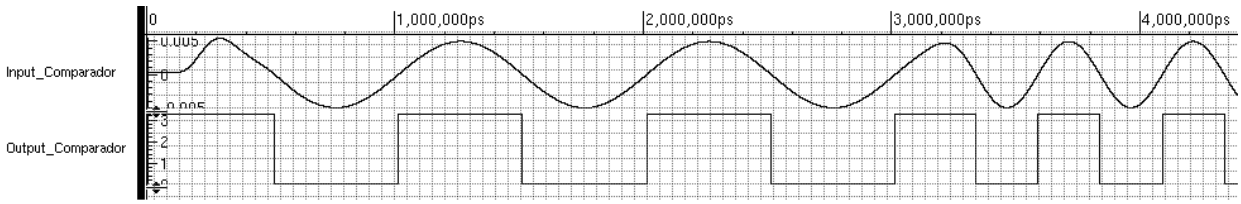

Figura 5.8: Simulação Comparador.

comece a clipar o sinal onde na verdade não deveria ser positivo ou negativo, e sim é apenas o ruído distorcendo os dados do sistema.

Vejamos um exemplo do mesmo trem de bits agora com potência de entrada de -80 dBm - Fig. 5.9.

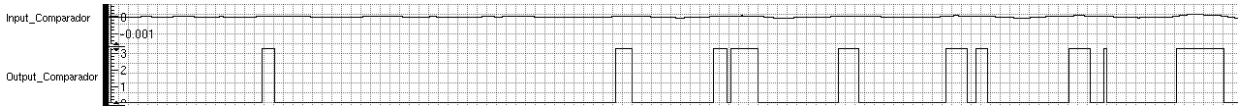

Figura 5.9: Simulação Comparador com potência de entrada -80 dBm.

Através da simulação pode-se verificar que a comparação realmente não exprime dados confiáveis, tampouco coerentes. É verdade que o sinal de entrada do comparador também é bastante degradado e distorcido, induzindo o comparador ao erro.

5.6 ANÁLISE DO CONVERSOR DIGITAL

Após a discussão sobre como realizar a integração entre o comparador e o demodulador (capítulo **??**), foi visto que era necessária a implementação de um bloco que realizasse a conversão do sinal analógico do comparador para um sinal digital.

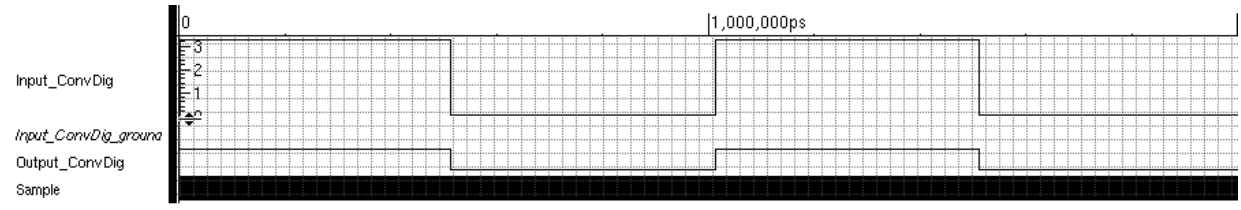

Figura 5.10: Simulação conversor digital.

Na simulação do conversor digital vê-se exatamente a conversão referenciado por uma amostragem, *Sample*. O *Sample* como verificado tem frequência de 1 THz para que a conversão não perdesse nenhuma propriedade e nem introduzisse distorção digital, como atrasos, no sinal necessário. O *Sample* é apresentado na figura para mostrar a grandeza da frequência do sinal de amostragem em relação aos outros de entrada e saída. Assim pode-se visualizar o sinal de entrada analógico *Input_ConvDig* proveniente do comparador variando de 0 a 3,3 V e o sinal *Output_ConDig* a saída do conversor digital sendo idêntico ao sinal de entrada, porém sendo um sinal binário, 0 ou 1.

Após a conversão o sinal está pronto para prosseguir e ser demodulado, retirando os dados recebidos, pelo demodulador.

5.7 ANÁLISE DO DEMODULADOR

Não entraremos no mérito de funcionamento ou qualquer análise mais crítica do demodulador já que este foi inteiramente desenvolvido e testado pela equipe digital do LDCI, como já dito, já tendo sido simulado e validado, além de ter sido esse bloco próprio que foi inserido no chip e enviado para fabricação. Assim o escopo na modelagem aqui prevê apenas a integração do sistema, se limitando a conhecimentos básicos de como excitar o bloco para que funcione na especificação dos protocolos desejados.

Na simulação a seguir já se pode ver o sinal convertido para digital e se pode distinguir a frequência mais alta da mais baixa, sendo fácil de verificar qual dado deve ser o sinal de saída.

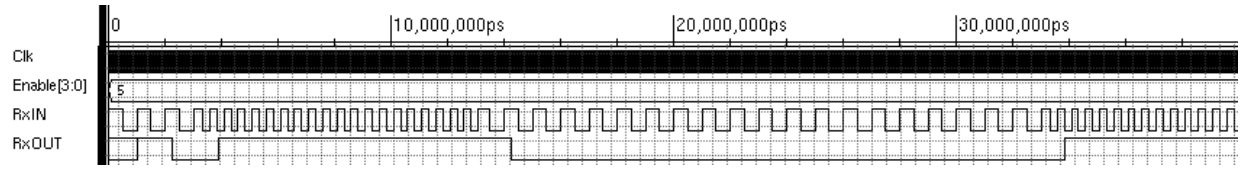

Figura 5.11: Simulação Demodulador.

5.8 ANÁLISE DO RECEPTOR COMPLETO

Tendo em vista a funcionalidade dos blocos individualmente, será apresentado então a análise e simulação do receptor completo, de forma a validar as especificações definidas inicialmente. Na Fig. 5.13 é apresentado o esquemático do sistema completo.

Abaixo é apresentada o resultado da simulação para o sinal da Fig. 5.5 com -40dBm de potência, para os primeiros 60 μs tempo suficiente para analisar o comportamento do sistema.

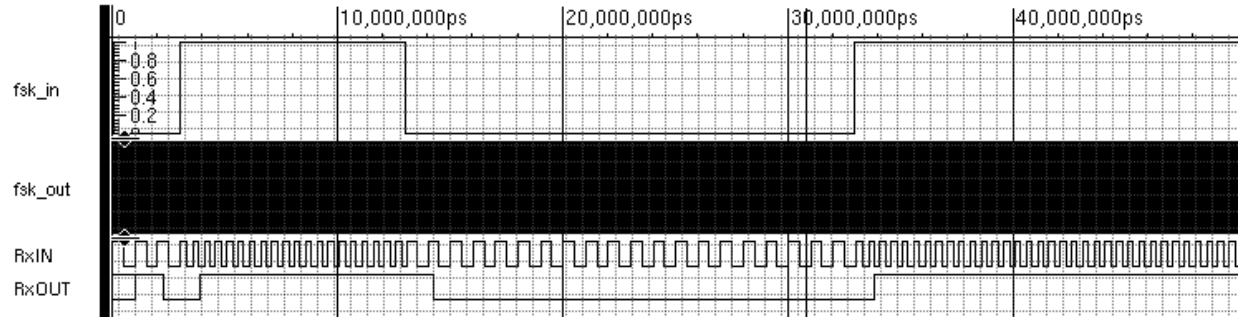

Figura 5.12: Simulação receptor.

Nota-se na figura o sinal de ao qual estamos excitando os receptor, fsk_in, este corresponde à entrada do gerador FSK na Fig. 5.1. Já o sinal fsk_out é a saída do gerador com os bits de entrada já modulados em FSK binário. Não é visualizável nem traz nenhuma informação importante, já

que com uma freqüência de 921/922 MHz não é possível verificar o sinal modulado, contudo o sinal foi introduzido na simulação para lembrar de que o dado real que transcorre pelo sistema é o fsk_out.

Em contrapartida, podemos verificar na entrada do demodulador o sinal $RxIN$, este sendo o sinal já discretizado e já passado pelo Mixer, estando portanto a uma freqüência de 1/2 MHz, sendo possível então distinguir e verificar os bits 1 e 0, de frequência maior e menor respectivamente.

Já na saída do receptor, pode-se constatar o dado sendo demodulado e capturado, validando portanto o sistema, recebendo os bits de entrada, com um tempo de atraso médio de $1,28\ \mu s$ em bordas de descida e $0,88\ \mu s$ em bordas de subida para dois pontos.

$$t_{atraso,desc} = 14,276 - 13\ [\mu s] \tag{5.22}$$

$$t_{atraso,desc} = 1,276\ \mu s \tag{5.23}$$

e

$$t_{atraso,sub} = 33,876 - 33\ [\mu s] \tag{5.24}$$

$$t_{atraso,sub} = 0,876\ \mu s \tag{5.25}$$

Isso se deve ao fato das propriedades dos protocolos de demodulação do demodulador.

Para encontrarmos a sensibilidade do sistema, foi variada a potência de entrada de maneira a analisar o sinal de saída. A princípio o sistema funcionaria a partir do momento que captasse sinais no comparador, já recebendo dados, mas conforme se vê na simulação da Fig. 5.14, o sistema de potências menores a $-80dBm$ praticamente não são captadas pelo sistema, sendo capturados apenas alguns pulsos, estes sem significado nenhum em termos de dados, sendo apenas flutuações geradas por ruídos.

A partir de $-80dBm$ começa a se verificar atividade no comparador, mas em um olhar mais crítico, podemos conferir que esses dados não correspondem a sinais recebidos de forma correta, já que não segue a freqüência do receptor, tampouco o período de duração dos bits. Dados concretos começam a ser verificados a partir de $-73dBm$, sendo assim de fato capturados os bits recebidos no receptor.

Essa sensibilidade de fato não corresponde aos $-90dBm$ do parâmetro do receptor definido originalmente, isso se deve ao fato de o ganho nos estágios de LNA e Mixer serem prejudicados e amortecidos pelas perturbações, tais como ruído e intermodulação de 3^a ordem. É interessante notar que ao se aumentar apenas o ganho sem se alterar os outros parâmetros, o sistema prontamente já começa a responder em potências de sinais de entrada cada vez menores, o que é bem intuitivo e lógico, assim como diminuir IP_3 dos blocos.

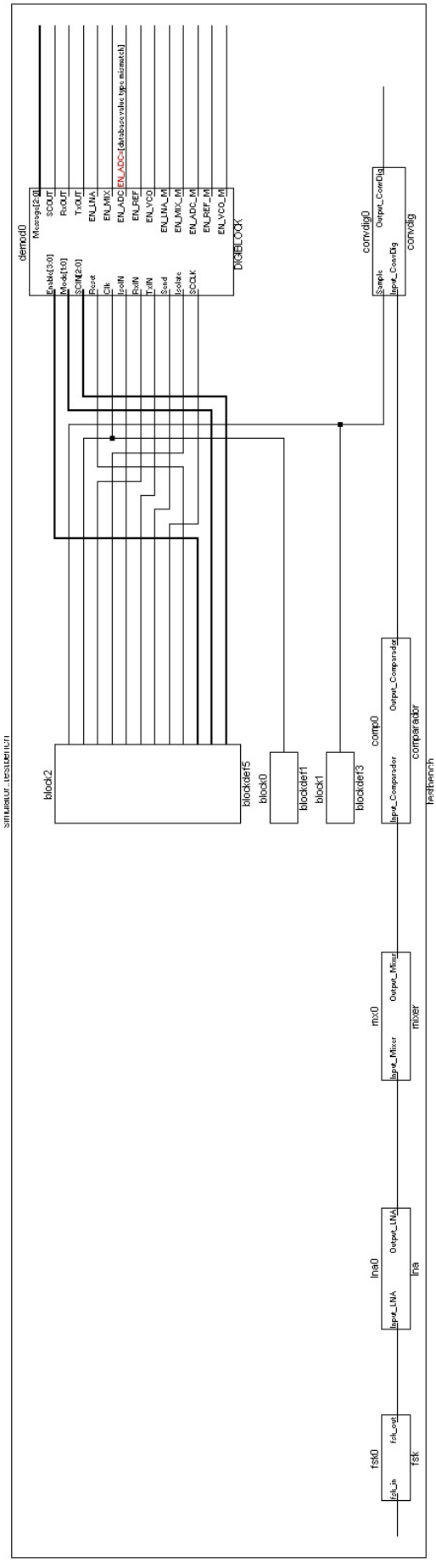

Figura 5.13: Esquemático receptor.

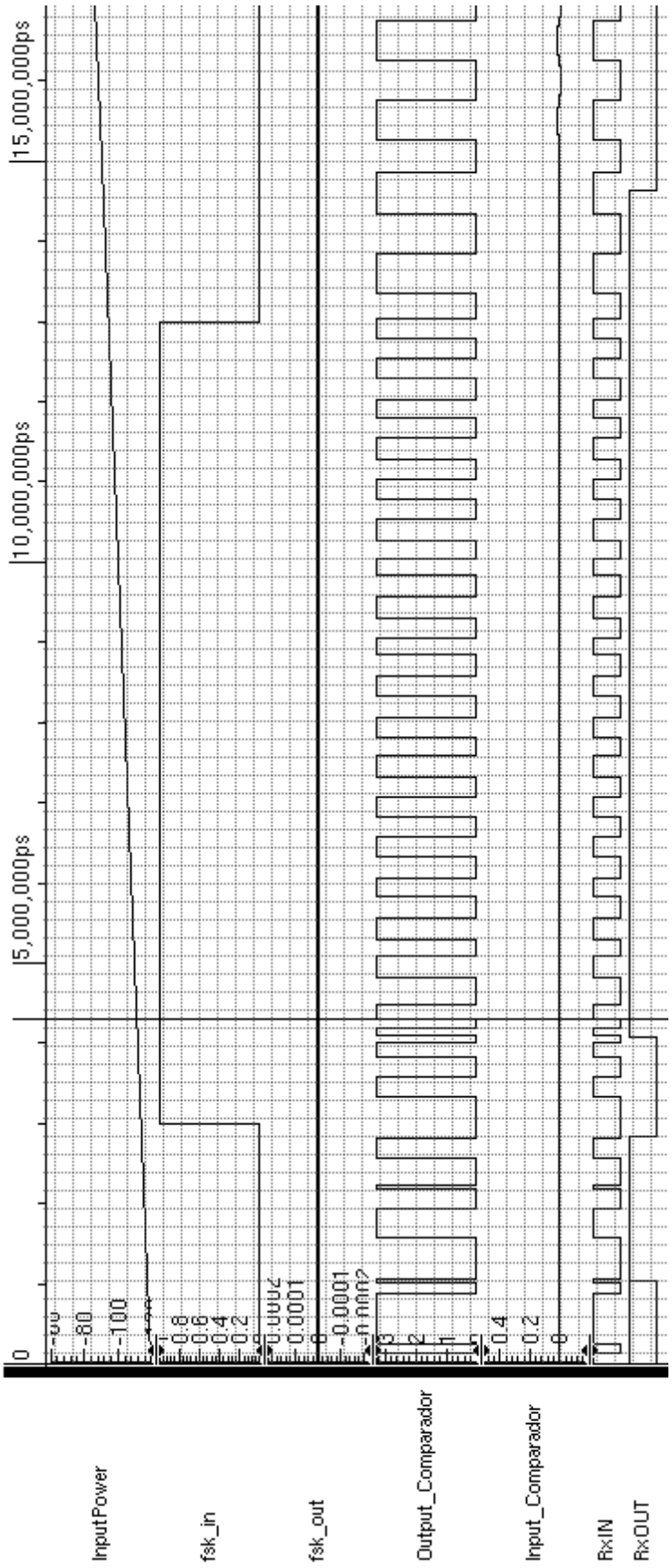

Figura 5.14: Simulação variando potência de entrada.

Capítulo 6

Conclusões

Foi apresentada nesse documento a proposta da descrição em alto nível de um modelo da seção de recepção do transceptor RF desenvolvido no laboratório de projetos de circuitos integrados do departamento de engenharia elétrica da UnB com a finalidade de ter em mãos um instrumento capaz de realizar simulações, fazendo com que possa ser feita a análise crítica dos dados simulados dando uma prévia do sistema em funcionamento, possibilitando a correção de erros e ajuste de parâmetros adequados.

6.1 DESCRIÇÃO DA MODELAGEM REALIZADA

A modelagem do transceptor RF envolveu a consulta e apoio a toda equipe do LDCI que participou do design do chip assim como utilização das ferramentas CADENCE para desenvolvimento dos códigos, compilação e simulação.

Na revisão bibliográfica foram abordados conceitos teóricos pertinentes de forma sucinta tendo em vista a extensão do assunto evitando portanto tornar o documento muito extenso.

A modelagem do receptor, considerando esse como uma junção de vários blocos, adotou-se o método de abordagem do problema bottom-up, mas ao final foi validado o sistema como um todo.

Verificou-se nas simulações que o sistema opera de forma adequada nas frequências de operações desejadas, tanto para o sinal recebido com codificação NRZ quanto Manchester. Porém em análise aos blocos foi possível constatar que o ganho foi prejudicado em função da não-linearidade e ruídos introduzidos no sistema. Portanto o LNA tinha uma expectativa de ganho de $25,3dB$ e teve somente $22,54dB$. Já o Mixer passou de $15,1dB$ para $11,21dB$.

Em relação à análise do sistema completo pode-se constatar que para um sinal de entrada com potência menor que $-120dBm$ não consegue ser detectado pelo sistema, sendo assim para potências maiores que essa necessárias para que o sistema comece de fato a receber dados. Mas mesmo para sinais com potência maior que $-117dBm$ verifica-se que não há coerência nos dados recebidos devido à estar em uma sensibilidade ainda pequena fazendo com que o ruído e a não linearidade interfiram na confiabilidade dos dados. Nota-se então que somente a partir de $-117dBm$ que os dados recebidos são coerentes.

A partir disso, pode-se analisar que para os parâmetros obtidos para os blocos, o sistema vai responder a uma sensibilidade desejada de $-90dBm$. Por isso nota-se a importância de se ter uma ferramenta dessa pronta antes de se projetar o chip ou qualquer que seja o sistema para que problemas como esse possam ser constatados e contornados definindo-se outros parâmetros.

6.2 PROPOSTAS DE TRABALHOS FUTUROS

É verdade que a modelagem aqui proposta não cobriu de forma perfeita o sistema definido, como descrito durante o desenvolvimento do projeto. Há blocos que necessitam de revisões e aperfeiçoamentos, como o comparador que é quase ideal e o PLL que insere uma senóide ao MIXER e não uma onda quadrada.

O chip desenvolvido foi enviado para prototipagem em julho de 2010, com previsão de chegada no final de 2010. Tendo isso em vista a falta de detalhamento de alguns blocos, como citado acima, segue abaixo sugestões de trabalhos futuros:

- Atualizar parâmetros dos blocos, tendo em vista que últimas alterações não foram modificadas;
- Teste com o chip;
  - Teste de cada bloco;
  - Teste do receptor por completo;
  - Confronto com os resultados obtidos;
- Aperfeiçoar o Comparador;
- Aperfeiçoar o LNA e Mixer para verificação de outros parâmetros mais específicos;
- Descrição do PLL;
- Inserção de diferentes tipos de ruídos nos blocos analógicos;
- Implementação do Gerador FSK em codificação Manchester;
  - Teste do receptor funcionando em codificação Manchester.

Outra proposta futura é simular a seção $LNA \rightarrow MIXER \rightarrow Comparador$ com o demodulador a partir do software Ultrasim tendo como base a metodologia comentada na Fig. 4.16. Isso é importante justamente pelo fato de que ao se modelar o sistema em alto nível se tem a vantagem de simular todo o projeto de forma mais rápida.

Através da solução proposta, o problema foi contornado, porém através da síntese lógica são inseridas características da realidade fazendo com que essas informações tornem a simulação bastante lenta. Para um sistema mais complexo isso pode vir a prejudicar a análise do sistema, tirando assim a vantagem do modelo em alto nível.

6.2.1 Proposta de um Gerador FSK/Codificador Manchester

Anteriormente foi apresentado o esquema do gerador em modulação FSK binário para teste do receptor. Aqui é exposta uma sugestão de descrição de um gerador que seja também um codificador Manchester, para que possa ser testado o sistema com essa característica.

Retomando então a Fig. 5.1, tem-se que inserir um bloco interno que faça a conversão do sinal em NRZ para sinal Manchester, tal como apresentado na Fig. 6.1.

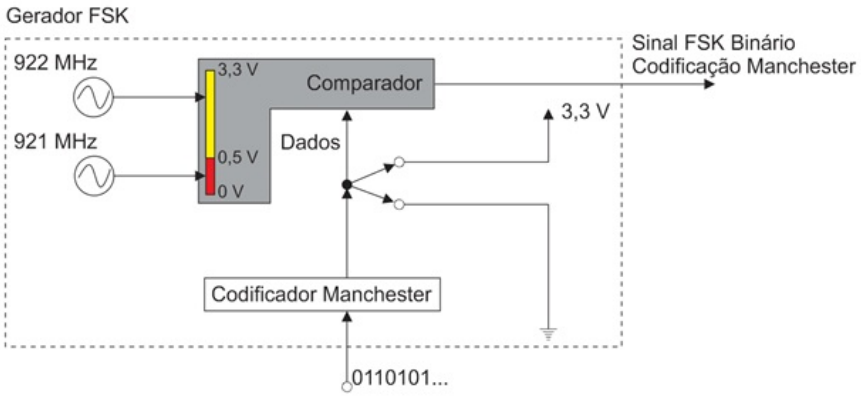

Figura 6.1: Gerador FSK com codificação Manchester.

O estágio codificador Manchester na verdade não é complicado, pelo contrário, sendo bem simples. O cuidado deve ser tomado na inserção dos bits levando em consideração que o clock do demodulador é de 20 MHz.Assim cada bit deve respeitar o período de 50 ps. Então o codificador apenas analisa o bit de entrada durante cada período e insere a informação correspondente em Manchester, ou seja, transições de 0 para 1 para bits 0 e 1 para 0 para bits 1, lembrando que as bordas não transmitem informação, sendo os dados contidos na transição no meio do período de duração do bit.

É importante notar também que segundo o protocolo Manchester do demodulador ele funciona tendo que setar os sinais de entrada da seguinte maneira:

- Enable: 0100
- Mode
  - 00: Para transmitir uma palavra de 8 bits;
  - 01: Para transmitir uma palavra de 40 bits;
  - 10 ou 11: P ara transmitir uma palavra de 128 bits.

Note que antes de enviar a palavra é necessário enviar a sequencia de bits 0101 e ao terminar tem que enviar a sequencia de bits 1010. Essa adição de 8 bits é necessária para que o demodulador saiba quando iniciar a demodulação e quando termina a palavra enviada, ficando então da seguinte maneira: 0101(Dados)1010.

```
always @(posedge Clk) begin
        if (V(fsk_in) >=0.5) begin
                vout = 0;
                #XXXXXXX;       // Ajustar ao período da frequência de amostragem.
                vout = 1;
                #XXXXXXX;       // Ajustar ao período da frequência de amostragem.
        end
        else if (V(fsk_in) < 0.5) begin
                vout = 1;
                #XXXXXXX;       // Ajustar ao período da frequência de amostragem.
                vout = 0;
                #XXXXXXX;       // Ajustar ao período da frequência de amostragem.
        end
end
```

Figura 6.2: Gerador FSK com codificação Manchester.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] RAZAVI, B. *RF Microelectronics.*, 1. ed. [S.1.]: Prentice-Hall, 1998.

[2] RAZAVI, B. *AnÃąlise BÃąsica de Circuitos Para Engenharia.*, 7. ed. [S.1.]: LTC Editora, 2003.

[3] BUTTERWORTH, S. *Theory of filter amplifiers.*

[4] GAJSKI, D.; ABDI, S.; GERSTLAUER, A.; SCHIRNER, G. *Embedded System Design: Modeling, Synthesis and Verification.*, 1. ed. [S.I.]: Springer, 2009.

[5] GRÃŰTKER, T.; LIAO, S.; MARTIN, G.; SWAN, S. *System Design with SystemC.*, 1. ed. [S.I.]: Springer, 2002.

[6] Site EDUCYPEDIA: *http://www.educypedia.be/electronics.*

[7] BENSKY, A. *Short-range Wireless Communication: Fundamentals of RF System Design and Application.*, 11. ed. [S.I.]: Newnes, 2004.

[8] LATHI, B. *Mordern Digital and Analog Communication Systems.*, 3. ed. [S.l.]: Oxford University Press, 1998.

[9] ROGERS, J.; PLETT, C. *Radio Frequency Integrated Circuit Design.*, [S.l.]: Artech House, 2003.

[10] RAZAVI, B. *Design of Analog CMOS Integrated Circuits.*, 1. ed. [S.l.]: McGraw Hill, 2001.

[11] ALLEN, P. E.; HOLDBERG, D. R. *CMOS Analog Circuit Design.*, 2. ed. [S.l.]: Oxford University Press, 2002.

[12] TEXAS INSTRUMENTS *CC1000 Single Chip Very Low Power RF Transceiver datasheet.*

[13] HAGEN, J. *Radio-Frequency Electronics: Circuits and Applications.*, [S.l.]: Cambridge University Press, 1996.

[14] KUNDERT, K.; ZINKE, O. *The DesignerâĂŹs Guide to Verilog-AMS.*, 1. ed. [S.l]: kluwer Academic Publishers, 2004.

[15] PARK, J. L. S. *The Level Modeling for Hardware Architecture Exploration with IEEE 802.11n Receiver Example.*

[16] CALDARI, M.; CONTI, M.; COPPOLA, M.; CURUBA, S.; PIERALISI, L.; TURCHETTI, C. *Transaction-Level Models for AMBA Bus Architecture Using SystemC 2.0.*

[17] BESERRA, G. S.; MEDEIROS, J. E. G.; MADUREIRA, H. M. G.; CARNEIRO, J. L. C.; EUSSE, J. F.; JACOBI, R. P.; COSTA, J. C. *System-level Modeling of a Reconfigurable System on Chip for Wireless Sensor Networks Applications*

[18] VASILEVSKI, M.; PECHEUX, F.; ABOUSHADY, H.; LAMANE, L. *Modeling Heterogeneous Systems Using SystemC-AMS Case Study: A Wireless Sensor Network Node.*

[19] MIDORIKAWA, E. T. *Uma IntroduÃğÃčo Ãăs Linguagens de DescriÃğÃčo de Hardware.*, 2001

[20] NASCIMENTO, B. *GeraÃğÃčo de Estruturas e Vetores de Teste para Blocos Codificados*, 2001

ANEXOS

I. TUTORIAL CADENCE

Esta seção tem por objetivo apresentar o tutorial de utilização da ferramenta Cadence para realizar a síntese lógica e síntese física de um bloco digital. Note que os passos aqui apresentados são diretos, sendo mostrado o passo a passo. Observe também que o bloco utilizado é um conversor A/D após ser adicionada estruturas de teste pelo Encounter Test, esta parte não é relevante para o exposto aqui, portanto desconsidere essas considerações quando houver.

Para verificar o documento completo, vide referência [20].

I.1 Síntese Lógica

A SÍNTESE LÓGICA transforma uma descrição em nível RTL em uma netlist de portas lógicas através da otimização, que gera um hardware genérico, e do mapeamento tecnológico, no qual o hardware é mapeado para células disponíveis na biblioteca. Os objetivos são otimizar a área, o atraso e o consumo utilizando constraints.

Primeiramente criar a pasta onde será feita a SÍNTESE LÓGICA. Nesse caso, a pasta criada é:

/home/breno/Desktop/PROJETOS/dg_isr/uart_sintese_logica/uartdetsl_05022009 - uart(unidade sintetisada)det(depois do encounter test)sl(SÍNTESE LÓGICA)

Dentro dessa pasta criar outras três pastas:

deliverables - pasta vazia **reports** - pasta vazia **scripts** - deixar o arquivo 'reports.tcl' e o arquivo 'script.tcl'

Alterar o arquivo 'script.tcl' como o nome do arquivo, nome do clock, etc. Habilitar também os FS (flip-flops scan), já que o ENCOUNTER TEST implementa esses tipos de flip-flops.

Iniciar no diretório 'synth' o programa PKS com o comando:

pks &

Abrir o arquivo 'script.tcl' e seguir no terminal de comando do programa PKS.

Após isso, será feita a simulação pós SÍNTESE LÓGICA. Para isso, criar uma subpasta de onde se estava sendo trabalhada, nesse caso:

/home/breno/Desktop/PROJETOS/dg_isr/uart_sintese_logica/uartdetsl _05022009/sim_p

Nela colocar:

Arquivo VERILOG gerado do PKS que se encontra na pasta 'deliverables'

Arquivo SDF gerado do PKS que se encontra na pasta 'deliverables'

Bibliotecas da tecnologica (nesse caso c35_CORELIB.v e c35_UDP.v)

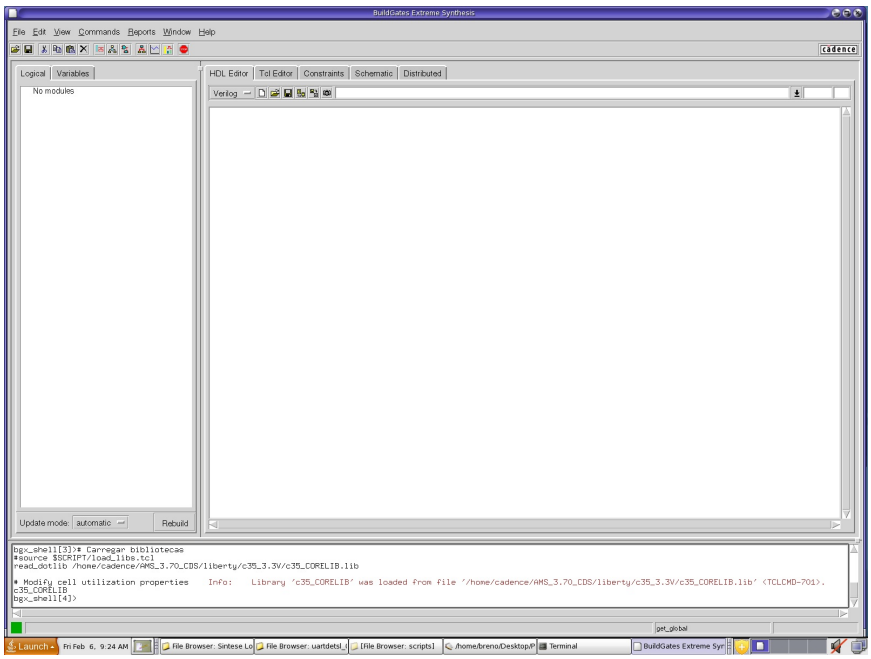

Figura I.1: Interface gráfica do programa PKS.

Testbench (nesse caso em VHDL)

Arquivo sdf_parameters.cmd

No arquivo **sdf_parameters.cmd** alterar os valores de **SCOPE** e **COMPILED_SDF_FILE**. SCOPE se refere ao nome da entidade instanciada na testbench. COMPILED_SDF_FILE é o caminho para o arquivo SDF compilado.

Após isso, abrir o programa NCLAUNCH com o comando:

nclaunch -new &

Clicar em **'Multiple Step'** –> **'Create cds lib'** –> **'Save'** –> **'Ok'**. No campo **'Filters'** adicionar *.sdf e dar Enter. Selecionar todas as entidades e clicar em **'Tools'** –> **'VHDL Compiler'** e habilitar as opções:

Enable VHDL 93 features Enable order independent compilation

Clicar em **'Ok'**. Depois disso compilar as entidades em ordem de hierarquia:

c35_CORELIB.v

c35_UDP.v

Arquivo SDF

Arquivo VERILOG

Testbench (se houver entidades que a testbench precisa instanciar, entao compilar essas entidades antes)

Na parte direita, abrir **'Worklib'** e selecionar a testbench, nesse caso e **'tb_serialinterface'**. Clicar em **'Tools'** –> **'Elaborator'**. No campo **'Other Options'** colocar o seguinte comando

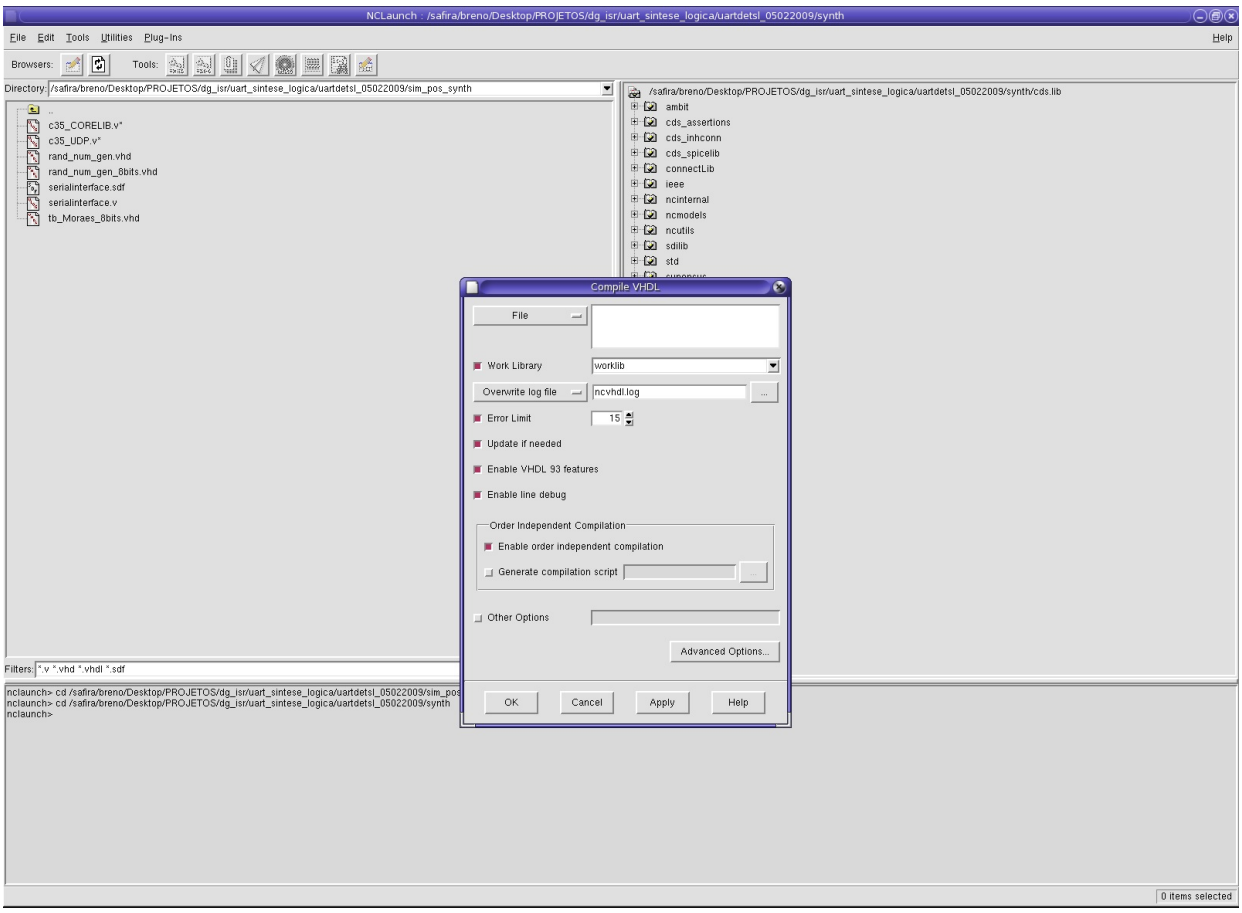

Figura I.2: Interface gráfica do programa NCLAUNCH.

(vide Fig. I.3):

-sdf_simtime -SDF_CMD_FILE sdf_parameters.cmd

Clicar em **'Advanced Options'**. Em **'General'** habilitar **'Enable VHDL 93 features'** e em 'Default timescale' colocar **'1 ns / 1 ps'**.

Clicar em 'Ok' e 'Ok' novamente.

No lado direto clicar em abrir **'Snapshots'** e selecionar a entidade ali existente. Clicar no atalho para chamar o simulador.

Selecionar na parte esquerda a entidade a ser simulada. Clicar na interface gráfica em para ver o esquemático. Depois, clicar em para abrir o programa de simulação.

No terminal de comando do programa, digitar **run** e o tempo suficiente para analisar a estrutura. Nesse caso:

run 3 ms

Ver e conferir a simulação.

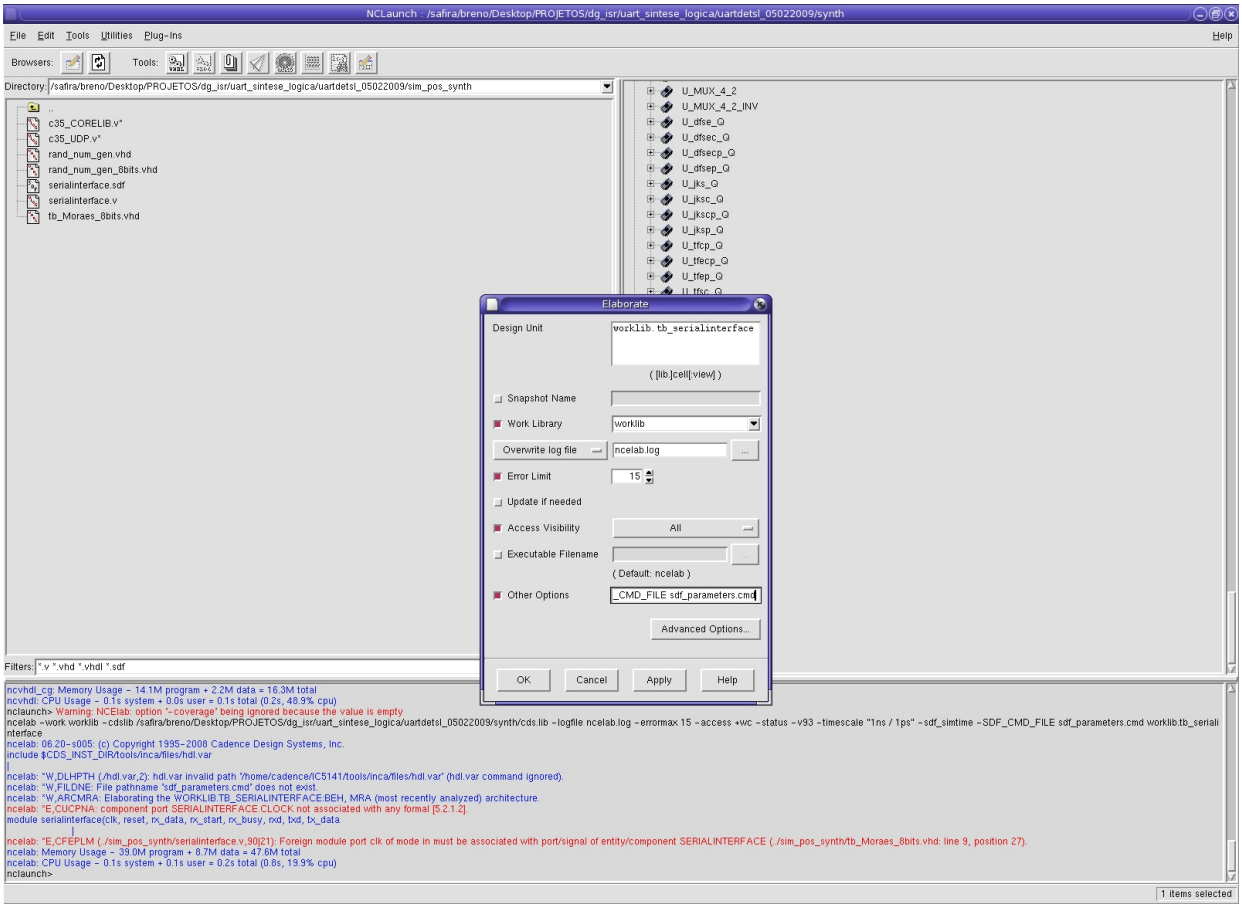

Figura I.3: Inserindo configurações no NCLAUNCH.

I.2 Síntese Física

A SÍNTESE FÍSICA é a seqüência de etapas que visa transformar uma netlist de portas lógicas em layout, que consiste de uma representação de circuitos para a fabricação das mascaras que serão usadas pela foundry.

Primeiramente criar uma pasta onde será feita a SÍNTESE FÍSICA. Nesse caso, a pasta criada e:

/home/breno/Desktop/PROJETOS/dg_isr/uart_sintese_fisica/uartdetsf_05022009 - uart(unidade sintetisada)det(depois do encounter test)sf(SÍNTESE FÍSICA)

Dentro dessa pasta criar a pasta layout. E dentro dessa pasta, criar as seguintes subpastas:

floorplan - pasta vazia

cts - pasta vazia

place - pasta vazia

route - pasta vazia

deliverables - pasta vazia

reports - pasta vazia

scripts - deixar os arquivos 'script.tcl', variaveis.tcl, **load.tcl**, powerplan.tcl, placement.tcl, clocktree.tcl, postCTS.tcl, nanoroute.tcl, metalfill.tcl, verify.tcl, signoff.tcl e generate.tcl

data - deixar os arquivos 'enc.pref.tcl', (nome da unidade).CONF, (nome da unidade).ctstch e os arquivos gerados da SÍNTESE LÓGICA: (nome da unidade).SDC e (nome da unidade).v

Alterar os seguintes arquivos: Na pasta **scripts**:

variaveis.tcl constraints.tcl generate.tcl

Na pasta data:

(nome da unidade).conf

(nome da unidade).ctstch - alterar o clock

Iniciar no diretório **'layout'** o programa SOC ENCOUNTER com o comando:

encounter

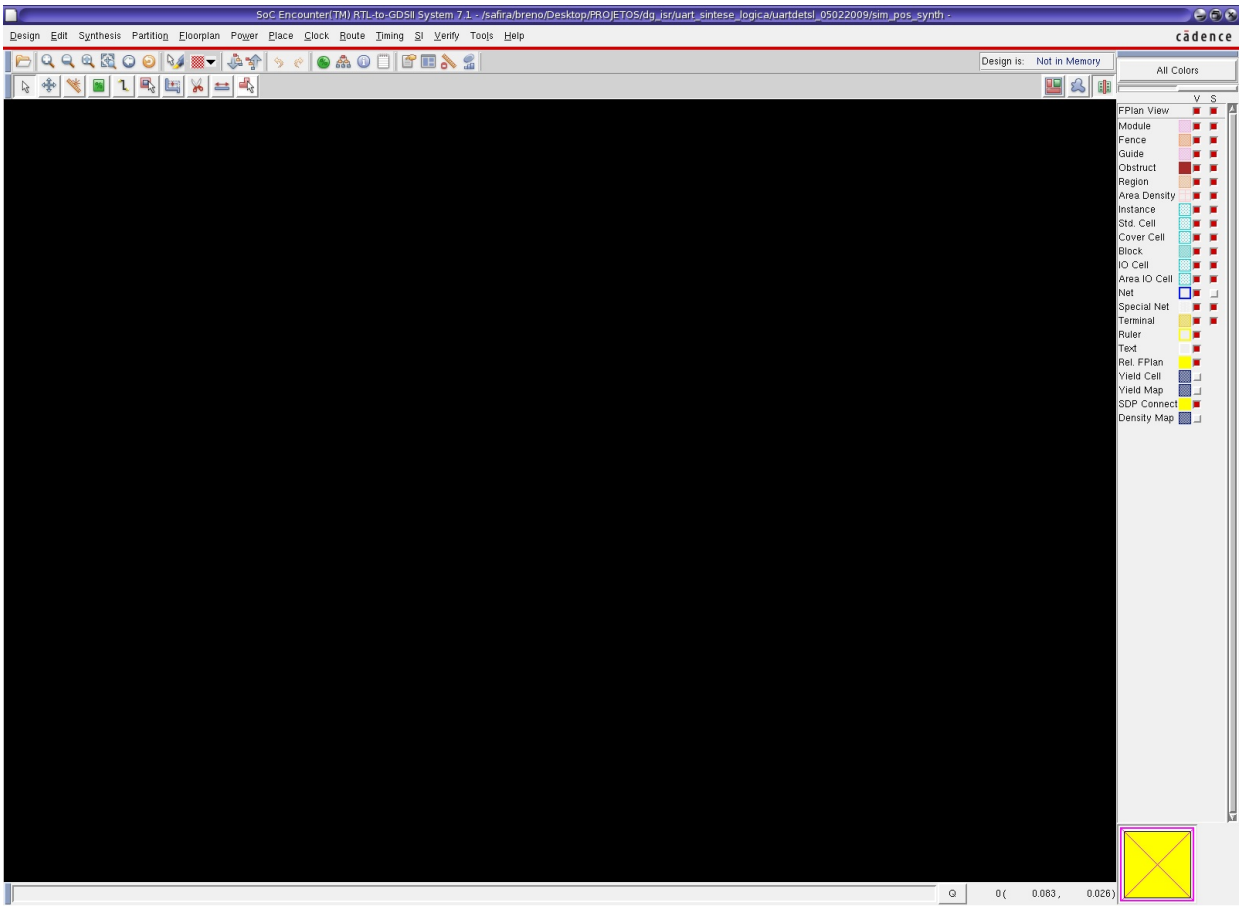

Figura I.4: Interface gráfica SOC ENCOUNTER.

Abrir o arquivo **'script.tcl'** que se encontra na pasta **'scripts'** e seguir os comandos no próprio terminal (vide Fig. I.5).

O resultado após a execução do script é mostrado na Fig. I.6.

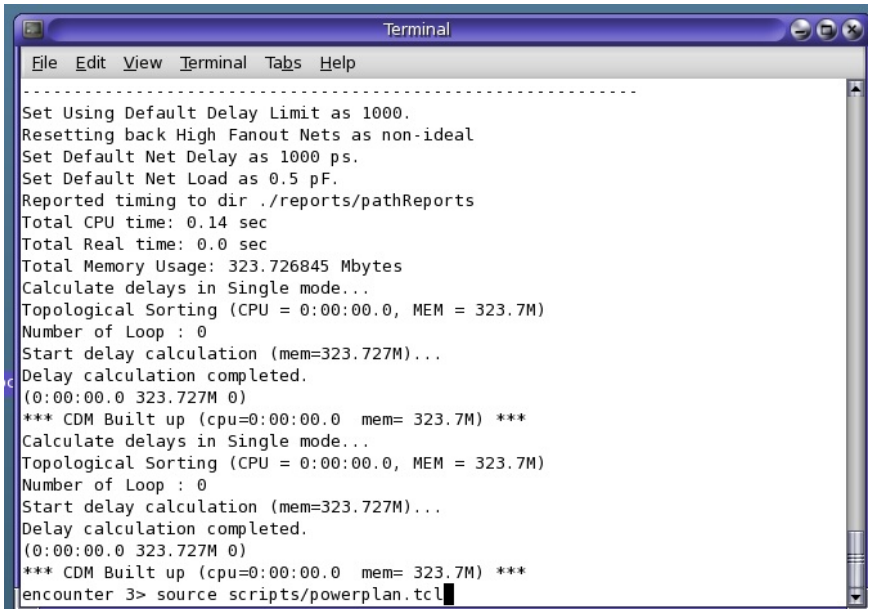

Figura I.5: Executando script no terminal.

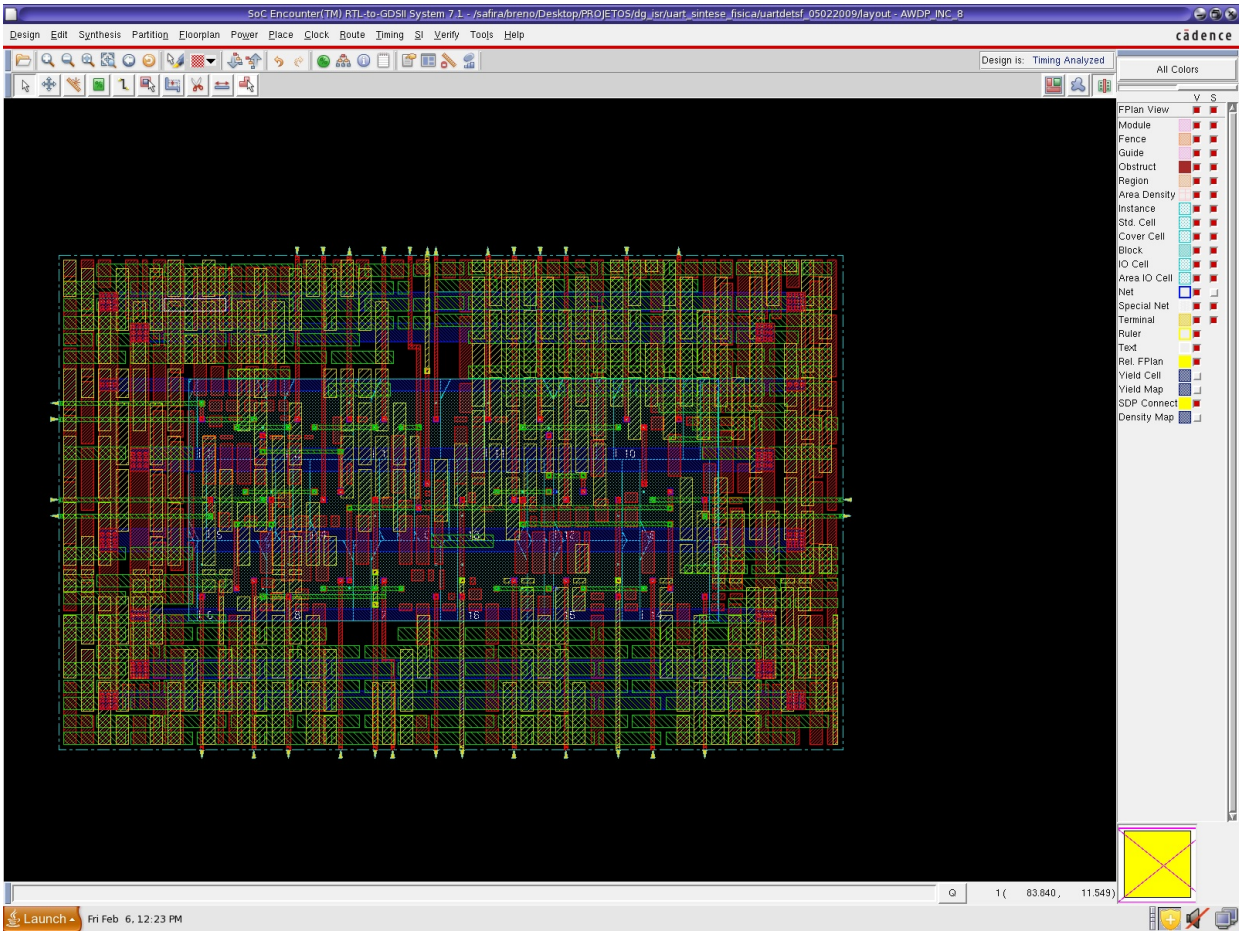

Figura I.6: Executando script no terminal.

II. CÓDIGOS DESCRITOS

II.1 SCRIPT PRINCIPAL

rm -rf worklib

mkdir worklib

ncvlog -ams fsk.vams -messages

ncvlog -ams lna.vams -messages

ncvlog -ams mixer.vams -messages

ncvlog -ams comparador.vams -messages

ncvlog -ams convdig.vams -messages

#Compila demodulador

ncvlog -work worklib -cdslib cds.lib -logfile ncvlog.log -errormax 15 -update -linedebug -status c35_ CORELIB.v

ncvlog -work worklib -cdslib cds.lib -logfile ncvlog.log -errormax 15 -update -linedebug -status c35_ UDP.v

ncsdfc -logfile ncsdfc.log -compile -cdslib cds.lib -status digiblock_ test.sdf

ncvlog -work worklib -cdslib cds.lib -logfile ncvlog.log -errormax 15 -update -linedebug -status digiblock_ test.v

ncvlog -ams testbench.vams -messages

ncelab testbench -timescale 1ps/1ps -messages

ncsim testbench -messages -analogcontrol run.scs -gui &

*rm *.log*

II.2 GERADOR FSK - CODIFICAÇÃO NRZ

‘include "disciplines.vams"

‘timescale 10ps / 1ps

module fsk (fsk_ in, fsk_ out);

input fsk_ in;

output fsk_ out;

electrical fsk_ in;

electrical fsk_ out;

parameter real freq1 = 921M from (0:inf);

parameter real freq2 = 922M from (0:inf);

parameter real offset = 0;

parameter real InputPower = -40; //Potência do sinal de entrada em dBm

parameter real R = 50; //Matching resistence

real ampl;

analog begin

@ (initial_ step)

begin

ampl = (R (10**(InputPower/10)) /1000)**(0.5); //Cálculo da amplitude de entrada a partir da potência*

end

if (V(fsk_ in) >= 0.5) begin

*V(fsk_ out) <+ ampl * sin(2 * 3.14 * freq2 * $abstime) + offset;*

$bound_ step(0.05/freq1);

end

else if (V(fsk_ in) < 0.5) begin

*V(fsk_ out) <+ ampl * sin(2 * 3.14 * freq1 * $abstime) + offset;*

$bound_ step(0.05/freq2);

end

else V(fsk_ out) <+ 0;

end endmodule

II.3 LNA

‘include "disciplines.vams"

‘timescale 1ps / 1ps

(cds_ ams_ schematic *)*

module lna(Input_ LNA, Output_ LNA);

input Input_ LNA;

output Output_ LNA;

electrical Input_ LNA; //Sinal de entrada

electrical Output_ LNA; //Sinal de saída

electrical NonLinearLNA;

electrical GainIP3;

electrical Noise;

electrical NoiseVoltage;

electrical LNA_ NonLinear_ Noise;

electrical gnd;

//Parametros de ENTRADA

parameter real Gaindb = 25.3; //Ganho de voltagem (dB)

parameter real NoiseFigure = 1.85; //Medida de degradação do SNR da saída comparado a entrada (dB)

parameter real IIP3dbm = -6; //Output IP3 - Fator de linearidade (dBm)

parameter real BW = 10e6; //Largura de banda

//Parametros do sistema

parameter real kBoltzmann = 1.38e-23;

parameter real T = 290; //Temperature in Kelvin

parameter real R = 50; //Matching resistence

parameter real vth=0; //Parametro para detectar frequencia

real Gain, IIP3, IIP3v, NoiseFactor, InputNoisems, InputNoiserms, OutputNoiseDueToInputrms, OutputNoisems, OutputNoiserms;

real nonlin3, nonlin2, nonlin1, nonlin0;

real f, t1, t2;

integer randseed;

initial begin

//NONLINEARITY

*Gain = 10**(Gaindb/10);*

*IIP3 = (10**((IIP3dbm - 30)/10)); //IIP3 absoluto*

*IIP3v = (IIP3*R)**0.5; //voltage*

*nonlin3 = 4*Gain/(3*IIP3v*IIP3v); //A3 = 4*Gain/(3*IIP3v*IIP3v);*

nonlin2 = 0; //A2 = 0;

nonlin1 = Gain; //A1 = Gain;

nonlin0 = 0; //nonlin = [A3 A2 A1 0];

//NOISE

*NoiseFactor = 10**(NoiseFigure/10); //noise ratio in numbers*

*InputNoisems = 4*kBoltzmann*T*BW*R; //impedance is 50 Ohms*

*InputNoiserms = InputNoisems**0.5; //rms noise voltage at input*

*OutputNoiseDueToInputrms = nonlin3*InputNoiserms*InputNoiserms*InputNoiserms + nonlin1*InputNoiserms;*

OutputNoisems = NoiseFactor((OutputNoiseDueToInputrms)**2);*

*OutputNoiserms = OutputNoisems**0.5; //Total output noise root mean square voltage, the same as standard_deviation*

end

analog begin

//Gerador de ruído

// $rdist_normal (seed , mean , standard_deviation) ;

@ (initial_step)

begin

randseed = 1 ;

t1 = 0;

t2 = 0;

end

V(NoiseVoltage) <+ $rdist_normal(randseed, 0, OutputNoiserms);

//Detector de frequencia

@(cross(V(Input_LNA)-vth,+1))

begin

t1=t2;

t2=$abstime;

f = 1/(t2-t1);

end

//Não linearidade de 3ł ordem

*V(GainIP3) <+ nonlin3*V(Input_LNA)*V(Input_LNA)*V(Input_LNA);*

*V(NonLinearLNA) <+ V(GainIP3) + Gain*V(Input_LNA);*

//Saída do LNA aplicada ao filtro de largura de banda

```
V(LNA_NonLinear_Noise) <+ V(NonLinearLNA) + V(NoiseVoltage);
V(gnd) <+ 0; //Instanciando Ground
end
//Filtro passa-faixa Butterworth 5a ordem 915MHz - 928MHz
resistor #(.r(50)) (*integer library_binding = "analogLib";*)
R1 (LNA_NonLinear_Noise, net3);
resistor #(.r(50)) (*integer library_binding = "analogLib";*)
R2 (Output_LNA, gnd);
capacitor #(.c(7.88523e-14)) (*integer library_binding = "analogLib";*)
C1 (net4, net5);
capacitor #(.c(3.96173e-10)) (*integer library_binding = "analogLib";*)
C2 (net5, gnd);
capacitor #(.c(2.43654e-14)) (*integer library_binding = "analogLib";*)
C3 (net6, net7);
capacitor #(.c(3.96173e-10)) (*integer library_binding = "analogLib";*)
C4 (net7, gnd);
capacitor #(.c(7.88523e-14)) (*integer library_binding = "analogLib";*)
C5 (net8, Output_LNA);
inductor #(.l(3.78299e-07)) (*integer library_binding = "analogLib";*)
L1 (net3, net4);
inductor #(.l(7.52947e-11)) (*integer library_binding = "analogLib";*)
L2 (net5, gnd);
inductor #(.l(1.22427e-06)) (*integer library_binding = "analogLib";*)
L3 (net5, net6);
inductor #(.l(7.52947e-11)) (*integer library_binding = "analogLib";*)
L4 (net7, gnd);
inductor #(.l(3.78299e-07)) (*integer library_binding = "analogLib";*)
L5 (net7, net8);
endmodule
```

II.4 MIXER

```
‘include "disciplines.vams"
‘timescale 1ns / 1ps
(* cds_ ams_ schematic *)
module mixer(Input_ Mixer, Output_ Mixer);
input Input_ Mixer;
output Output_ Mixer;
electrical Input_ Mixer; //Sinal de entrada
electrical Output_ Mixer; //Sinal de saída
electrical Mixed_ Signal;
electrical LOsignal; //Sinal LO - transfere para frequencia intermediaria
electrical IdealMixSignal;
electrical NonLinearMixer;
electrical GainIP3;
electrical Noise;
electrical NoiseVoltage;
electrical gnd;
//Parametros de ENTRADA
parameter real Gaindb = 15.1; //Ganho de voltagem (dB)
parameter real NoiseFigure = 24; //Medida de degradação do SNR da saída comparado a entrada (dB)
parameter real IIP3v = -6.6; //Output IP3 - Fator de linearidade (dBm)
parameter real BW = 8e6; //Largura de banda
parameter real OF = 921e6; //Frequencia de operaçao
parameter real lo_ amp = 1.65; //Amplitude do sinal LO (analogico)
parameter real lo_ offset = 1.65; //Amplitude do sinal LO (analogico)
parameter real freq_ lo = 920e6; //Frequencia do sinal LO (analogico)
//Parametros do sistema
parameter real kBoltzmann = 1.38e-23;
parameter real T = 290; //Temperature in Kelvin
parameter real R = 50; //Matching resistence
```

parameter real vth = 0; //Parametro para detectar frequencia

real lo_phase; //Fase do sinal LO

real Gain, IIP3, NoiseFactor, InputNoisems, InputNoiserms, OutputNoiseDueToInputrms, OutputNoisems, OutputNoiserms;

real nonlin3, nonlin2, nonlin1, nonlin0;

real f, t1, t2, filter;

real f2, t3, t4, filter2;

real f3, t5, t6;

real vmax, vmin;

integer randseed;

initial begin

//NAO-LINEARIDADE

*Gain = 10**(Gaindb/10);*

*nonlin3 = 4*Gain/(3*IIP3v*IIP3v); //A3 = 4*Gain/(3*IIP3v*IIP3v);*

nonlin2 = 0; //A2 = 0;

nonlin1 = Gain; //A1 = Gain;

nonlin0 = 0; //nonlin = [A3 A2 A1 0];

//NOISE

*NoiseFactor = 10**(NoiseFigure/10); //noise ratio in numbers*

*InputNoisems = 4*kBoltzmann*T*BW*R; //impedance is 50 Ohms*

*InputNoiserms = InputNoisems**0.5; //rms noise voltage at input*

*OutputNoiseDueToInputrms = nonlin3*InputNoiserms*InputNoiserms*InputNoiserms + nonlin1*InputNoiserms;*

OutputNoisems = NoiseFactor((OutputNoiseDueToInputrms)**2);*

*OutputNoiserms = OutputNoisems**0.5; //Total output noise root mean square voltage*

end

analog begin

//Gerador de ruído

@ (initial_step)

begin

randseed = 1 ;

t1 = 0;

```
t2 = 0;
vmax = 0;
vmin = 0;
end
V(NoiseVoltage) <+ $rdist_normal(randseed, 0, OutputNoiserms);
//Detector de frequencia
@(cross(V(Input_Mixer)-vth,+1))
begin
t1=t2;
t2=$abstime;
f = 1/(t2-t1);
end
lo_phase = 2*3.14*freq_lo*$abstime;
V(LOsignal) <+ lo_amp*sin(lo_phase) + lo_offset; // Creates a sinusoidal voltage source. Signal coming from PLL.
V(IdealMixSignal) <+ V(LOsignal)*V(Input_Mixer);
//V(IdealMixSignal) <+ ((V(LOsignal) - lo_offset)/1000)*V(Input_Mixer);
V(GainIP3) <+ nonlin3*V(IdealMixSignal)*V(IdealMixSignal)*V(IdealMixSignal);
V(NonLinearMixer) <+ V(GainIP3) + Gain*V(IdealMixSignal);
//Saída do Mixer aplicada ao filtro de largura de banda
V(Mixed_Signal) <+ V(NonLinearMixer) + V(NoiseVoltage);
V(gnd) <+ 0; //Instanciando Ground
end
//Filtro passa-baixa
resistor #(.r(50)) (*integer library_binding = "analogLib";*)
R1 (Mixed_Signal, net3);
resistor #(.r(50)) (*integer library_binding = "analogLib";*)
R2 (Output_Mixer, gnd);
capacitor #(.c(1.03005e-9)) (*integer library_binding = "analogLib";*)
C1 (net4, gnd);
capacitor #(.c(1.03005e-9)) (*integer library_binding = "analogLib";*)
C2 (net5, gnd);
```

```
inductor #(.l(9.83578e-7)) (*integer library_ binding = "analogLib";*)
L1 (net3, net4);
inductor #(.l(3.1831e-6)) (*integer library_ binding = "analogLib";*)
L2 (net4, net5);
inductor #(.l(9.83578e-7)) (*integer library_ binding = "analogLib";*)
L3 (net5, Output_ Mixer);
endmodule
```

II.5 COMPARADOR

```
‘include "disciplines.vams"
‘timescale 1ps / 1ps
module comparador(Input_ Comparador, Output_ Comparador);
input Input_ Comparador;
output Output_ Comparador;
electrical Input_ Comparador; //Sinal de entrada
electrical Output_ Comparador; //Sinal de saída
electrical Sens_ Comparador; //Sinal de saída discretizado
parameter real vth = 0; //Parametro para detectar frequencia
parameter real vth1 = 0.5; //Parametro para detectar frequencia
parameter real td = 1p; // time delay (s)
parameter real tt = 1p; // output transition time (s)
real f, t1, t2, Vout;
analog begin
if ((V(Input_ Comparador) > -35e-6) && (V(Input_ Comparador) < 35e-6)) begin
V(Sens_ Comparador) <+ vth;
end
else if ((V(Input_ Comparador) < -35e-6) || (V(Input_ Comparador) > 35e-6)) begin
V(Sens_ Comparador) <+ V(Input_ Comparador);
end
@(cross(V(Output_ Comparador) - vth, 0));
Vout = ((V(Sens_ Comparador) > vth) ? 3.3 : 0);
```

```
V(Output_Comparador) <+ transition(Vout, td, tt);
@(cross(V(Output_Comparador)-vth1,+1))
begin
t1=t2;
t2=$abstime;
f = 1/(t2-t1);
end
end
endmodule
```

II.6 CONVERSOR DIGITAL

```
‘include "disciplines.vams"
‘timescale 1ps / 1ps
module convdig(Sample, Input_ConvDig, Output_ConvDig);
input Sample;
input Input_ConvDig;
output Output_ConvDig;
reg Output_ConvDig;
electrical Input_ConvDig; //Sinal de entrada
reg a_smp;
always @(Sample) begin
a_smp = V(Input_ConvDig);
if (a_smp > 0.5) begin
Output_ConvDig = 1;
end
else begin
if (a_smp < 0.5) begin
Output_ConvDig = 0;
end
end
end
```

endmodule

II.7 TESTBENCH

‘timescale 1ps / 1ps

‘include "disciplines.vams"

module testbench ();

electrical gnd;

ground gnd;

electrical fsk_in;

reg signal;

reg Clk;

reg [0:3] Enable; // [0 1 0 1]

reg [0:1] Mode; // [0 0]

reg Isolate;

reg Reset;

reg IsoIN;

reg TxIN;

reg Send;

reg [2:0] SCIN;

reg SCCLK;

reg Sample;

parameter real td = 1p; // time delay (s)

parameter real tt = 1p; // output transition time (s)

// Instanciando o DUT

fsk fsk0 (fsk_in, fsk_out);

lna lna0 (fsk_out, Output_LNA);

mixer mx0 (Output_LNA, Output_Mixer);

comparador comp0 (Output_Mixer, Output_Comparador);

convdig convdig0 (Sample, Output_Comparador, Output_ConvDig);

DIGIBLOCK demod0 (Enable, Reset, Clk, IsoIN, Mode, Output_ConvDig, TxIN, Send, Isolate, SCIN, SCCLK, SCOUT, Message, RxOUT, TxOUT, EN_LNA, EN_MIX, EN_ADC, EN_REF, EN_VCO, EN_LNA_M, EN_MIX_M, EN_ADC_M, EN_REF_M, EN_VCO_M);

// Criando um clock

always #25000 Clk= Clk; //Clock de 20MHz (25000 passos de 1ps, equivalendo a uma troca de 40MHz e a um clock de 20MHz)

// Taxa de amostragem do Conversor Digital

always #1 Sample= Sample;

// Iniciando simulacao digital

initial begin

// Valores iniciais

Sample = 0; // Em t=0

Clk = 0; // Em t=0

signal = 0;

Enable = 0101;

Mode = "00";

Isolate = 0;

IsoIN = 0;

TxIN = 0;

Send = 0;

SCIN = "00";

SCCLK = 0;

Reset = 0;

#1000000 Reset = 1;

// Sinal de entrada (1001101)

#2000000 signal = 1; // Em t=2us

#10000000 signal = 0; // Em t=12us

#10000000 signal = 0;

#10000000 signal = 1; // Em t=32us

#10000000 signal = 1;

#10000000 signal = 0; // Em t=52us

#10000000 signal = 1; // Em t=62us

#5000000 signal = 0; // Em t=67us

#5000000 signal = 1; // Em t=72us

#5000000 signal = 0; // Em t=77us

#5000000 signal = 1; // Em t=82us

end

// Excitando o sistema com senoide de duas frequencias

analog begin

V(fsk_in) <+ transition(signal, td, tt);

end

endmodule