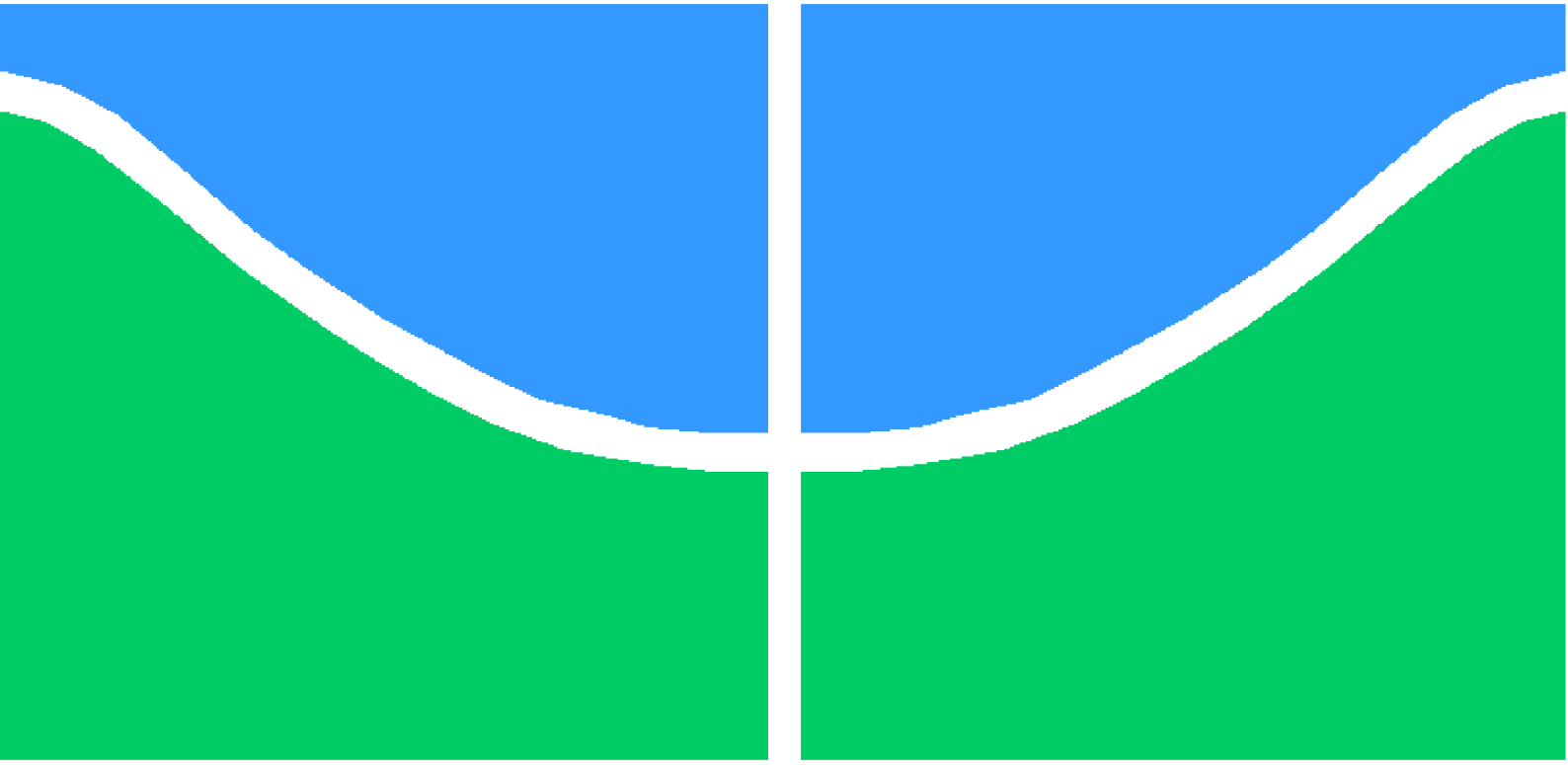


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia Eletrônica

Projeto de Quantizadores Não Lineares para Conversores Analógico-Digitais com Base na Transformada da Incerteza

Autor: Laryssa Lorrany Olinda Costa

Orientador: Prof. Dr. Sandro Augusto Pavlik Haddad

Brasília, DF

2017

Laryssa Lorrany Olinda Costa

Projeto de Quantizadores Não Lineares para Conversores Analógico-Digitais com Base na Transformada da Incerteza

Monografia submetida ao curso de graduação em Engenharia Eletrônicada Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Sandro Augusto Pavlik Haddad

Coorientador: Prof. Dr. Wellington Avelino do Amaral

Brasília, DF

2017

Laryssa Lorrany Olinda Costa

Projeto de Quantizadores Não Lineares para Conversores Analógico-Digitais com Base na Transformada da Incerteza / Laryssa Lorrany Olinda Costa. – Brasília, DF, 2017-

130 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Sandro Augusto Pavlik Haddad

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2017.

1. Quantizador. 2. (*Unscented Transform*. I. Prof. Dr. Sandro Augusto Pavlik Haddad. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Projeto de Quantizadores Não Lineares para Conversores Analógico-Digitais com Base na Transformada da Incerteza

CDU 02:141:005.6

Laryssa Lorrany Olinda Costa

Projeto de Quantizadores Não Lineares para Conversores Analógico-Digitais com Base na Transformada da Incerteza

Monografia submetida ao curso de graduação em Engenharia Eletrônicada Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, Julho de 2017:

Prof. Dr. Sandro Augusto Pavlik Haddad
Orientador

Prof. Dr. Wellington Avelino do Amaral
Convidado 1

Prof. Dr. José Edil Guimarães de Medeiros
Convidado 2

Brasília, DF
2017

Este trabalho é dedicado a mim.

Agradecimentos

Primeiramente agradeço a minha família, que sempre me apoiou e me deu a base necessária para que eu conseguisse permanecer na faculdade durante esses anos. Agradeço ao Prof. Dr. Sandro Augusto Pavilik Haddad e ao Prof. Dr. Wellington Avelino do Amaral pelo incentivo, orientação e compartilhamento de conhecimento. Agradeço também ao Prof. Dr. José Edil Guimarães de Medeiros pela oportunidade de participar do seu trabalho.

Por fim, gostaria de agradecer aos meus amigos e colegas de estágio, em especial o José Alberto, por toda experiência e conhecimento com eles adquiridos.

.

"It is the quality of one's convictions that determines success, not the number of followers."

-Remus Lupin

Resumo

No caso de sistemas de quantização lineares ou uniformes, pode haver grandes erros de quantização para sinais pequenos. Esses erros podem ser tão grandes que a relação sinal-ruído (SNR) não será suficiente para recuperar toda a informação do sinal. Como alternativa para esse problema pode-se utilizar quantizadores não lineares. O objetivo deste trabalho é a modelagem de um quantizador não linear, que utiliza uma distribuição arco-seno, para aplicação em um conversor analógico-digital com arquitetura Σ-Δ. O projeto foi desenvolvido com auxílio das ferramentas Cadence seguindo a metodologia de projetos *Top-Down*. Na modelagem foi utilizada a linguagem de descrição de *hardware* Verilog-A, que possibilita a análise comportamental e simulação mista. Este trabalho foi dividido em duas partes: modelagem dos quantizadores linear e não linear com distribuição arco-seno e modelagem do modulador Σ-Δ com aplicação dos quantizadores. Na primeira parte modelou-se todos os blocos em Verilog-A e implementou-se um quantizador linear e um quantizador não linear utilizando-se topologia do tipo $flash$. Na segunda parte modelou-se todos os blocos do modulador Σ-Δ, aplicando-se os quantizadores projetados anteriormente e fazendo uma comparação de desempenho entre o modulador com quantizador linear e o modulador com quantizador arco-seno.

Palavras-chaves: Conversor A/D. Quantizador não linear. Modulador Σ-Δ

Abstract

In case of linear or uniform quantization systems, there may be large quantization errors for small signals. These errors can be so large that the signal-to-noise ratio (SNR) will not be sufficient to recover all the signal information. Alternatively to this problem can be used non-linear quantizers. The objective of this work is the modeling of a non-linear quantizer, which uses an arcsine distribution, for application in an analog-digital converter with Σ-Δ architecture. The project was developed with the help of Cadence tools following the *Top-Down* project methodology. In the modeling, the hardware description language, $Verilog - A$, was used, which allows behavior analysis and mixed simulation. This work was divided in two parts: modeling of linear and non-linear quantizers with arcsine distribution and modeling of the modulator Σ-Δ with application of the quantizers. In the first part, all the blocks were modeled in Verilog-A, implementing a linear quantizer and a non-linear quantizer using a $flash$ topology. In the second part all the modulator blocks Σ-Δ were modeled by applying the previously designed quantizers and performing a performance comparison between the linear quantizer modulator and the arcsine modulator.

Key-words: A/D Converter. Non-linear quantizer. Σ-Δ modulator.

Sumário

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

A/D	*Analog-to-Digital*
ADC	*Analog-to-Digital Converter*
AMS	*Analog Mixed-Signal*
BER	*Bit Error Rate*
D/A	*Digital-to-Analog*
DAC	*Digital-to-Analog Converter*
DNL	*Differential Non-Linearity*
ENOB	*Effective Number of Bits*
HDL	*Hardware Description Language*
HDTV	*High Definition Television*
INL	*Integral Non-Linearity*
LSB	*Least Significant Bit*
MMC	*Monte Carlo Methods*
PAM	*Pulse Amplitude modulation*
PSRR	*Power Supply Rejection Ratio*
SINAD	*Signal*-to-*Noise* and *Distortion Ratio*
SNDR	*Signal-to-Noise Plus Distortion Ratio*
SNR	*Signal-to-Noise Ratio*
S/H	*Sample and Hold*
SFDR	*Spurious-Free Dynamic range*
THD	*Total Harmonic Distortion*
UT	*Unscented Transform*
Verilog	VERIfying LOGic
VHDL	VHSIC *Hardware Description Language*
OSR	*Oversampling Rate*

Lista de símbolos

Σ	Sigma
Δ	Delta
$avdd$	Tensão de referência positiva
$avss$	Tensão de referência negativa
$A0$	Amplitude do sinal de entrada
$A_{integrador}$	Ganho do integrador
A_{loop}	Ganho de realimentação
dB	Decibéis
clk	Clock
f	Femto
$f0$	Frequência do sinal
fs	Frequência de amostragem
fb	Largura de banda do sinal
fin	Frequência do sinal de entrada
Hn	Amplitude das n harmônicas
Hz	Hertz
k	Quilo
m	mili
n	Nano
N	Número de bits
OS	Oversampling
td	Tempo de atraso
tf	Tempo de transição

μ	Micro
Vin	Tensão de entrada
$Vout$	Tensão de saída

1 Introdução

Neste capítulo serão apresentados os objetivos gerais e específicos, motivações e justificativas e por fim a organização deste trabalho.

1.1 Objetivos

Este trabalho tem como objetivo modelar quantizadores para aplicações em sistemas de conversão de dados. Dentro do seu escopo, o objetivo geral é a modelagem de um quantizador linear e de um quantizador não linear para a aplicação em um modulador Σ-Δ *Multi-bit* de 1ª ordem afim de comparar o desempenho dos dois quantizadores.

A modelagem dos quantizadores consiste em descrever cada bloco que compõe o mesmo em linguagem de alto nível, por meio da HDL Verilog-A. Os blocos a serem descritos são: comparador, encoder, decoder e chave. Para o modulador Σ-Δ serão modelados os blocos: integrador, subtrator e amplificador. Ambas as etapas serão desenvolvidas através do fluxo de projeto das ferramentas Cadence. O Virtuoso será usado para edição dos esquemático e o Spectre será a ferramenta foco de simulação.

1.2 Justificativa

Quando um sinal analógico é digitalizado por um conversor A/D, ele passa por um processo de quantização onde os valores das amostras do sinal são aproximados para um dos 2^N valores digitais, denominados níveis de quantização. A diferença entre o valor original da amostra e o valor digitalizado é chamada de erro de quantização. Dessa forma, uma vez digitalizado, o sinal original não pode mais ser recuperado com exatidão.

No caso da quantização linear ou uniforme, haverá erros de quantização grandes para sinais de valores pequenos. Estes erros podem ser da mesma grandeza que o próprio sinal e a relação sinal-ruído ($Signal\text{-}to\text{-}Noise\ Ratio$ - SNR) não será suficientemente grande para que toda a informação do sinal seja recuperada.

Por esse motivo, pode-se utilizar intervalos de quantização de larguras diferentes, caracterizando assim a quantização não linear.

Para muitos tipos de sinais, a quantização linear não é a mais eficiente. Na comunicação pela fala, por exemplo, observa-se estatisticamente a predominância de amplitudes menores, sendo que amplitudes mais altas quase não são utilizadas. O esquema de quantização linear será então, sem sentido pois muitos níveis de quantização não serão utilizados. O exemplo pode ser observado na Figura 1.

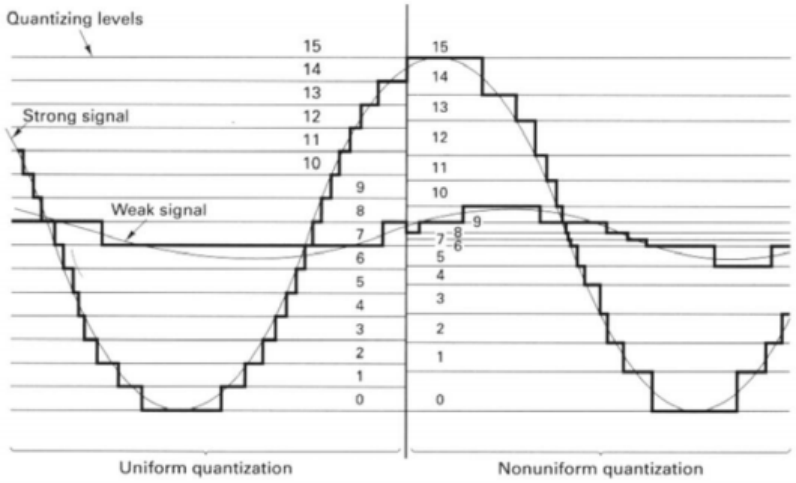


Figura 1 – Comparação entre quantização linear e não linear (LATHI, 1990).

Um método eficiente para resolver esse problema é implementar um sistema de quantização não linear, onde serão utilizados passos menores para amplitudes. Neste caso haverá pequenos intervalos de quantização para sinais de valor pequeno e intervalos maiores para sinais de maior valor, fazendo com que a relação entre o sinal e o seu erro de quantização seja aproximadamente igual para toda a faixa de amplitude do sinal de entrada (LATHI, 1990).

1.3 Organização do Trabalho

Para um melhor entendimento, este documento foi dividido em três partes: Revisão Bibliográfica, Projeto, Implementação e Resultados e Conclusão. A parte I apresenta um resumo de todos os assuntos presentes nesse documento. A parte II apresenta toda a idealização do projeto, bem como sua implementação e os resultados a obtidos. A parte III apresenta a conclusão, onde são expostas considerações sobre o projeto e possíveis trabalhos futuros.

Capítulo 1 - Introduz a problemática, bem como sua justificativa e organização do trabalho.

Capítulo 2 - Aborda os aspectos fundamentais da metodologia *top-down* que será usada no projeto.

Capítulo 3 - Introduz os conceitos básicos de funcionamento de um conversor A/D, parâmetros importantes, e as topologias do tipo de interesse para a aplicação neste trabalho.

Capítulo 4 - Introduz assuntos importantes para o entendimento sobre os processos de modelagem de quantização e proposta de implementação do circuito.

Capítulo 5 - Descreve a modelagem do quantizador linear de 4 bits com topologia $flash$.

Capítulo 6 - Descreve a modelagem quantizador arco-seno de 4 bits com topologia $flash$.

Capítulo 7 - Faz a comparação entre o quantizador linear e o quantizador arco-seno.

Capítulo 8 - Descreve a modelagem do modulador Σ-Δ *single-bit* e *multi-bit* de 1ª ordem com os dois quantizadores, linear e arco-seno, e compara o desempenho dos dois sistemas.

Capítulo 9 -Expõe as considerações finais do projeto e apresenta uma breve descrição de trabalhos futuros que possam dar continuidade ao projeto.

Apêndice A - Apresenta os códigos completos de cada bloco da modelagem em Verilog-A.

Parte I

Revisão Bibliográfica

2 Metodologias de Projeto

Metodologias de projeto são formas de iniciar, planejar e executar projetos. Basicamente são um conjunto de técnicas cuja finalidade é sintetizar as etapas de fluxo de um projeto. As metodologias facilitam a comunicação entre integrantes de equipe, asseguram o cumprimento de prazos e formalizam a execução e avanço dos projetos, tendo assim um papel fundamental no êxito dos mesmos (ZURITA, 2013).

No contexto deste trabalho, as abordagens mais interessantes de fluxo de projetos a serem abordadas são: *Botton-up* e *Top-down*. Segundo (KUNDERT; CHANG, 2005), a abordagem *bottom-up* é do tipo ascendente, vai do baixo para o mais alto nível de abstração. Nela, o projeto começa com os blocos individuais, que possuem determinadas especificações, e são combinados para a formação de subsistemas, esses subsistemas são combinados para formarem subsistemas maiores, e assim sucessivamente até chegar no sistema completo. O grande risco é que o desempenho do sistema exigido pode não ser possível com os blocos individuais projetados e conectados entre si, o que significa que um ou mais blocos terão que ser reprojetados.

No projeto *top-down* o desempenho de cada bloco individual necessário para satisfazer os requisitos de desempenho do sistema, é cuidadosamente estudado e compreendido antes dos blocos serem desenvolvidos. Isso reduz a necessidade de reprojeto dos blocos individuais, mas com o risco de que o desempenho esperado por um ou mais blocos seja irrealizável, o que exigiria a revisão do projeto do sistema (ZINKE, 2004).

2.1 Metodologia *Top-Down*

Como explicado acima, uma metodologia de projeto é considerada *top-down* se ela parte de uma descrição abstrata e faz a decomposição do problema em níveis, gerando problemas mais detalhados em cada nível, até a descrição final necessária (JOHANN, 1997). O uso dessa metodologia é útil em projetos de sistemas grandes e complexos. A premissa básica é projetar e verificar o sistema em um resumo ou nível "diagrama de blocos", antes de iniciar o projeto detalhado dos blocos individuais (ZINKE, 2004).

Um processo de projeto *top-down* procede de uma arquitetura para um sistema a nível de transistores. Cada nível é totalmente projetado antes de se projetar o próximo e cada nível é aproveitado no projeto do próximo. Com essa ideia, pode-se particionar o projeto em blocos menores e bem definidos, permitindo que mais projetistas possam trabalhar juntos e de forma produtiva. Isso tende a reduzir o tempo de projeto, além de melhorar a comunicação entre projetistas, e permitir que estes trabalhem juntos em

diferentes locais. Seguir uma metodologia de projeto *top-down* também reduz o impacto das mudanças que podem ocorrer ao longo do ciclo de projeto. Se por qualquer razão o circuito tiver que ser parcialmente reprojetado, a alteração será feita rapidamente, os modelos podem ser atualizados e o impacto sobre o resto do sistema pode ser rapidamente avaliado (ZINKE, 2004).

Segundo (JOHANN, 1997), para caracterizar o ciclo de projeto de forma simplificada, adota-se um modelo sequencial de projeto *top-down*, no qual cada etapa é completamente realizada e passa-se a etapa seguinte. O exemplo pode ser observado na Figura 2.

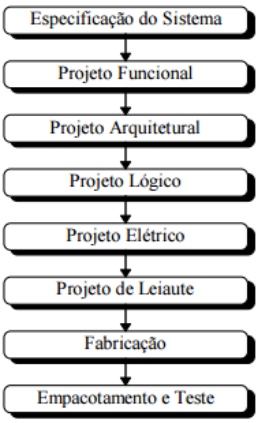


Figura 2 – Ciclo de projeto de um Circuito Integrado utilizando a metodologia *top-down* (JOHANN, 1997).

2.2 Linguagens de Descrição de *Hardware*

Linguagens de descrição de *hardware* (*HDLs*) são utilizadas no projeto de *hardware* por meio da descrição do comportamento dos mesmos. Diferente das linguagens de programação tradicionais, que descrevem algoritmos e sequências de operações para os dados da memória ou sobre periféricos, nos sistemas de hardware típicos existem muitos componentes individuais que operam simultaneamente. Para descrevê-los adequadamente é necessário descrever tanto o comportamento dos componentes individuais quanto a ligação entre eles.

HDLs têm duas aplicações principais: simulação e síntese. A síntese é o processo de projeto do *hardware*, utiliza-se a *HDL* para descreve-lo em um nível abstrato usando modelos de componentes que ainda não têm uma implementação física. A simulação é o processos onde aplica-se vários estímulos a uma modelo executável, criado na síntese, a fim de prever como ele irá reagir. Isso permite analisar a complexidade dos sistemas e diminuir tempo e custo de implementação (ZINKE, 2004).

As $HDLs$ descrevem sistemas digitais, analógicos ou mistos. Atualmente existem duas $HDLs$ disponíveis para descrever sistemas mistos, Verilog-AMS e $VHDL$-AMS, que são extensões das $HDLs$ digitais Verilog e $VHDL$ que apoiam a modelagem de sistemas analógicos e de sinal misto.

2.2.1 Verilog

Como explicado anteriormente, o Verilog-AMS é uma linguagem de modelagem para sistemas de sinais mistos, ou seja, sistemas constituídos por partes que processam sinais digitais e por partes que processam sinais analógicos. O Verilog-AMS é a incorporação e ampliação de duas línguas, Verilog-HDL e Verilog-A. Estas três linguagens atualmente compõem a família Verilog. O Verilog-HDL permite a descrição dos componentes digitais e o Verilog-A permite a descrição dos componentes analógicos. O Verilog-AMS combina estas duas línguas e adiciona capacidade de descrição de componentes de sinal misto (ZINKE, 2004).

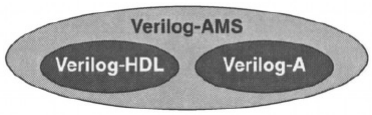


Figura 3 – Relação entre Verilog-AMS, Verilog-A e Verilog-HDL. (ZINKE, 2004)

Neste trabalho será utilizada a HDL Verilog-A. A seção a seguir é dedicada à exposição desta linguagem.

2.2.2 Verilog-A

O Verilog-A é um subconjunto do Verilog-AMS que processa sinais apenas analógicos. Ele pode ser utilizado tanto em sistemas elétricos quanto em não elétricos como mecânica, dinâmica dos fluidos e termodinâmica. Na eletrônica é muito utilizado no projeto e validação de sistemas analógicos e circuitos integrados por meio da criação de módulos que descrevem o comportamento de alto nível e a estrutura de sistemas e componentes. O comportamento de cada módulo pode ser descrito matematicamente em termos de seus terminais e parâmetros externos aplicados ao módulo. E a estrutura de cada componente pode ser descrita em termos de subcomponentes.

Neste trabalho utilizou-se o Verilog-A na modelagem de quantizadores com uma topologia $flash$ e na modelagem de moduladores Σ-Δ *single-bit* e *multi-bit*.

3 Conversores Analógico-Digitais

Conversores analógico-digitais (*Analog-to-Digital Converters-ADCs*) são dispositivos eletrônicos capazes de gerar uma representação digital a partir de uma grandeza analógica, normalmente um sinal representado por um nível de tensão ou intensidade de corrente elétrica.

Os *ADCs* são muito úteis na interface entre dispositivos digitais (microprocessadores, microcontroladores, *DSPs*) e dispositivos analógicos e são utilizados em aplicações para produtos de consumo, como câmeras, modems e televisões de alta definição (*HDTV*), bem como em sistemas especializados, tais como imagiologia médica, processamento de voz, instrumentação, controle industrial, leitura de sensores (RAZAVI, 1995).

A Figura 4 mostra o diagrama de blocos de um *ADC* genérico.

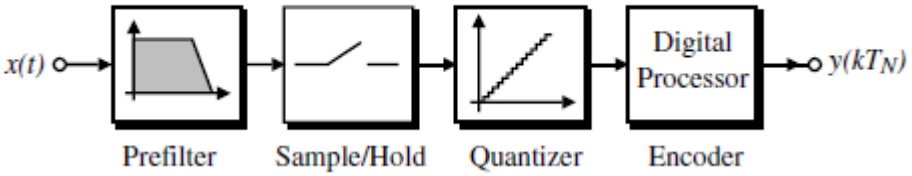


Figura 4 – Diagrama de blocos de um *ADC* (ALLEN; HOLBERG, 2002).

O primeiro bloco do *ADC* é um pré-filtro, também chamado de filtro *anti-aliasing* , tipicamente um passa-baixa ou passa-banda, necessário para atenuar as componentes de alta frequência do sinal. Uma vez que, pelo teorema de *Nyquist* , a frequência de amostragem (fs) tem que ser pelo menos duas vezes maior ou igual a componente de maior frequência do sinal, a filtragem descarta as frequências que não são essenciais para a informação contida nele.

O filtro *anti-aliasing* é seguido por um circuito *sample-and-hold* (S/H) que executa a etapa de amostragem. O S/H possui a função de aquisição do sinal (*sample*), em que a tensão é registrada, e a de retenção (*hold*), em que a tensão não é alterada por um determinado período tempo igual a $1/fs$ como mostra a Figura 5.

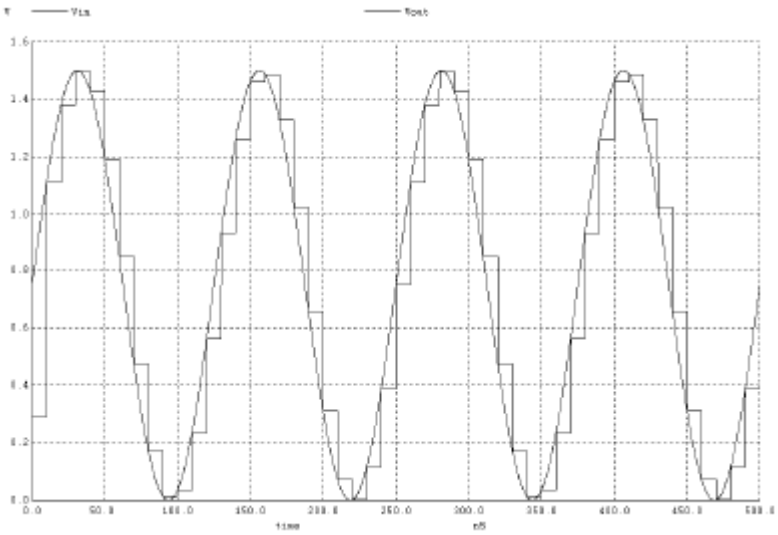


Figura 5 – S/H ideal para um sinal de entrada de 8 MHz (BAKER, 2008).

Depois de amostrado o sinal é quantizado, ou seja, é atribuído um valor de amplitude a cada sinal amostrado. Esse valor depende da resolução do ADC que define o número de níveis de quantização.

Por último, ocorre a codificação, realizada por meio de um *encoder*, que consiste na representação do sinal já amostrado e quantizado em códigos digitais, dentre os mais utilizados estão o código binário, gray e o termômetro que foi utilizado nesse trabalho.

3.1 Parâmetros Estáticos em Conversores A/D

A caracterização completa de um ADC envolve a análise de seu comportamento estático e dinâmico. Essas características originam os erros estáticos e erros dinâmicos. Os erros estáticos são erros do próprio conversor A/D e podem ser verificados por meio da análise do espaçamento entre os códigos da saída do conversor. A presença de espaçamento não ideal ou uniforme nos níveis de transição entre os códigos pode ser indicativo de erros estáticos no componente (TILDEN; LINNENBRINK; GREEN, 1999). Os erros estáticos dos $ADCs$ podem ser avaliados por meio da resposta do mesmo a uma rampa que excursione por toda a sua escala.

O erro de $Offset$, que pode ser visualizado na Figura 8, pode ser definido como o desvio da função de transferência do conversor com relação a reta de referência quando o sinal de entrada for 0 LSB ($Least\ Significant\ Bit$). Quando a transição de 0 para 1 não ocorre com uma entrada de 0.5 LSB, ocorre esse erro e sua compensação é feita subtraindo o valor do $offset$ das amostras digitais.

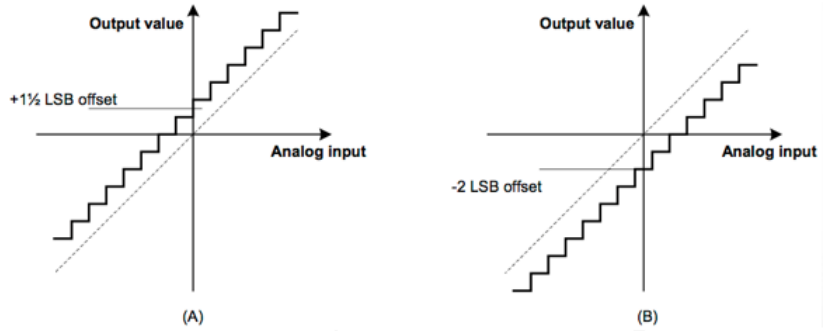


Figura 6 – Curva de um conversor A/D com erro de offset a) de +1.5 LSB; b) de -2 LSB (ALLEN; HOLBERG, 2002).

O erro de ganho acontece , visto na Figura 7, pelo desvio do último passo na saída do conversor com relação à reta de referência após compensar o erro por $offset$. Ele faz com que a inclinação da curva de transferência seja alterada e sua compensação pode ser feita escalonando os valores das amostras digitais.

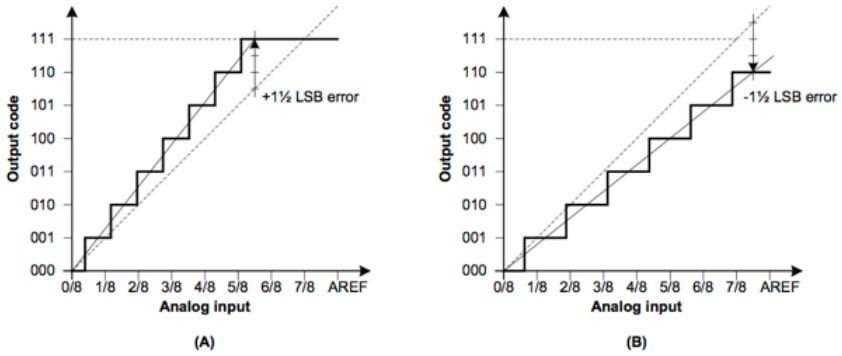


Figura 7 – Erro de ganho para um ADC : a) para +1.5 LSB, b) -1.5 LSB (ALLEN; HOLBERG, 2002).

Os erros de não-linearidade são o DNL ($Differential\ Non\text{-}Linearity$) e INL ($Integral\ Non\text{-}Linearity$). Esses erros representam uma deformação no sinal de saída do ADC conforme representado na Figura 8 (FLORES, 2003).

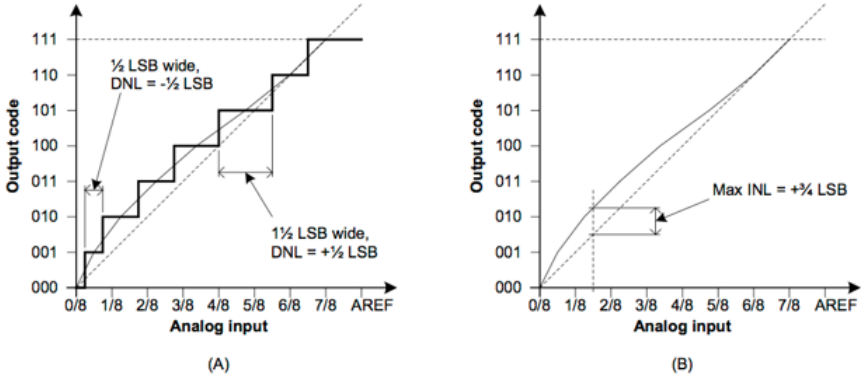


Figura 8 – Erros de linearidade para a curva de um ADC: a) DNL; b) INL (ALLEN; HOLBERG, 2002).

O erro DNL de um cógigo x corresponde ao desvio, expresso em LSB, do valor ideal e pode ser expresso pela equação 3.1 (TILDEN; LINNENBRINK; GREEN, 1999).

$$DNL = \frac{H(x) - H(x)ideal}{H(x)ideal} = \frac{H(x)}{H(x)ideal} - 1 \tag{3.1}$$

Sendo que $H(x)$ é o comprimento do código x no conversor em teste e $H(x)ideal$ é o comprimento do código x para um conversor com a mesma resolução ideal.

Os erros DNL para o mesmo código x podem ser acumulados por uma série finita de códigos, o que causará uma variação na curva ideal e formará o erro INL expresso pela equação 3.2 (TILDEN; LINNENBRINK; GREEN, 1999).

$$INL(x) = \sum_{k=1}^{x} DNL(k) \tag{3.2}$$

Outros parâmetros estáticos em $ADCs$ são a potência dissipada pelo ADC, sua impedância de entrada, as correntes de polarização de entrada, impedância de saída das referências de tensão e corrente e especificações de clock.

3.2 Parâmetros Dinâmicos em Conversores A/D

As características estáticas dos $ADCs$ são de grande importância para a determinação do comportamento desses componentes. Para casos onde o sinal de entrada não varia rapidamente, as características estatísticas do ADC são suficientes. No entanto, para aplicações que requerem a conversão de um sinal que varia rapidamente, a caracterização DC são é suficiente, sendo necessária também a caracterização AC assim como os parâmetros dinâmicos (FLORES, 2003).

A determinação dos parâmetros estáticos do ADC é baseada na análise no tempo, já a determinação dos parâmetros dinâmicos é baseada no espectro do sinal.

A SNR de um conversor A/D é uma das especificações dinâmicas mais importantes e pode ser definida como a relação entre os passos de quantização e o sinal de entrada. A SNR depende da resolução do conversor, da sua linearidade, distorção, incertezas na amostragem, ruídos eletrônicos e tempo de estabilização. Seu máximo teórico pode ser calculado pela equação 3.3.

$$SNR_{dB} = N \times 0.62 + 1.76 \tag{3.3}$$

Onde N é o número de bits do ADC.

Já para um ADC com sobreamostragem, o máximo teórico pode ser calculado pela equação 3.4.

$$SNR_{dB} = N \times 0.62 - 1.25 + 10log_{\frac{fs}{f0}} \tag{3.4}$$

Onde fs é a frequência de amostragem e $f0$ é a frequência do sinal.

A distorção total harmônica (*Total Harmonic Distortion* - THD) relaciona o somatório do valor das amplitudes das harmônicas presentes no sinal de saída do ADC com a amplitude do sinal de entrada, como mostra a equação 3.9 (TILDEN; LINNENBRINK; GREEN, 1999).

$$THD = \frac{\sqrt{H1^2 + H2^2 + H3^2 + H4^2 + H5^2}}{A0} \tag{3.5}$$

Onde Hn corresponde a amplitude das n harmônicas no sinal e $A0$ é a amplitude do sinal de entrada.

A especificação normalmente inclui a distorção harmônica (THD) no erro de quantização e passa a se chamar $SNDR$ (*Signal to Noise Distortion Ratio*) ou SINAD (*Signal to noise and Distortionratio*)

A faixa dinâmica livre de espúrios (*Spurious-Free Dynamic Range-SFDR*) é a medida da pureza espectral proporcionada pelo conversor. Ela está diretamente relacionada a quantidade de distorção dinâmica causada pelo circuito. Como pode ser observado na figura abaixo a $SFDR$ é a diferença entre a componente de sinal e a maior componente de distorção avaliados no domínio da frequência.

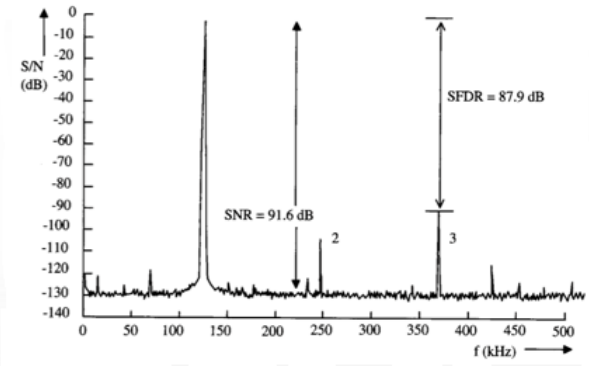


Figura 9 – SNR e $SFDR$ de um sinal aleatório (FLORES, 2003).

$$SFDR_{dB} = 20 \times log\frac{Fundamental}{HighestSpurious} \tag{3.6}$$

O número efetivo de bits ($Effective\ Number\ of\ Bits$-$ENOB$) é o parâmetro que avalia a resolução efetiva do ADC real, ou seja, é a quantidade de bits úteis que o circuito

apresenta. Esse parâmetro é muito importante pois um ADC de N bits, por exemplo, pode ter alguns bits corrompidos por ruído e distorção.

Um conversor Sigma-Delta pode, em tese, apresentar o mesmo $ENOB$ e resolução dependendo da taxa de sobreamostragem utilizada.

$$ENOB = \frac{SNDR_{measured} - 1.76}{6.02} \tag{3.7}$$

A taxa de erro de bit (*Bit Error Rate*-BER) define a quantidade de decisões erradas tomadas pelo comparador do conversor A/D. Em $ADCs$ Sigma-Delta, estes erros são diluídos na média. Quanto maior a taxa de sobreamostragem, menos relevante a BER.

Outros parâmetros dinâmicos são os *glitches*, ruído térmico, máxima taxa de amostragem, $PSRR$ (*Power Supply Rejection Ratio*), *settling time*, ganho diferencial e fase diferencial, entre outros.

3.3 Topologias de Conversores A/D

Os $ADCs$ podem ser classificados de acordo com a relação entre a largura de banda do sinal de entrada (fb) e a frequência de amostragem (fs) e também pela sua taxa de conversão. $Nyquist$ $ADCs$ possuem fb muito próximo ou igual a $0.5fs$. Já os $ADCs$ com sobreamostragem ($Oversampling$), possuem fb muito menor do que a frequência de amostragem fs (ALLEN; HOLBERG, 2002).

A Tabela 1 mostra melhor essa classificação de acordo com a topologia de $ADCs$ que serão apresentadas nas próximas seções.

Tabela 1 – Classificação dos $ADCs$ de acordo com sua arquitetura (ALLEN; HOLBERG, 2002).

Conversion Rate	**Nyquist ADCs**	**Oversampled ADCs**
Slow	Integrating (Serial)	Very high resolution <14-16 bits
Medium	Successive Approximation1-bit Pipeline Algorithmic	Moderate resolution <10-12 bits
Fast	Flash Multiple-bit Pipeline Folding andinterpolating	Low resolution <6-8 bits

3.3.1 Conversor A/D Paralelo

O ADC do tipo $flash$ ou paralelo, é umas das arquiteturas de conversores que atualmente apresenta a maior velocidade de operação. Esse tipo de conversor é recomendado

para aplicações que envolvem sinais com grande largura de banda (KESTER, 2004).

A topologia *flash* que pode ser observada na Figura 10, é baseada em comparadores de tensão de alta velocidade. Para um conversor de N bits, serão necessários $2^N - 1$ comparadores.

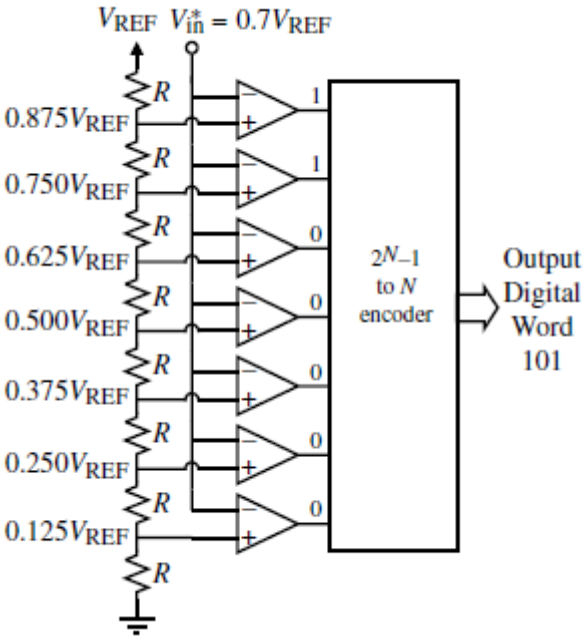


Figura 10 – Exemplo de um conversor *flash* de 3 bits (ALLEN; HOLBERG, 2002).

Pode-se observar na Figura 10 que as entradas não-inversoras dos comparadores são conectadas entre si e é nesse ponto onde é aplicado o sinal analógico a ser convertido. Já as tensões aplicadas nas entradas inversoras de cada um dos comparadores, são de um divisor resistivo, onde a tensão de referência para cada um é um bit menos significativo, maior que o do comparador abaixo. Cada comparador apresentará a saída com nível lógico alto quando a entrada analógica for maior que a sua tensão de referência (ALLEN; HOLBERG, 2002).

Na saída dos comparadores é necessário colocar um circuito de codificação que irá receber os sinais dos comparadores e codificar o sinal de saída em código binário ou GRAY. A maioria das aplicações dos *ADC flash* são no processamento de sinais de alta frequência, como sinais de vídeo, por exemplo, que necessitam de taxa de conversão da ordem de 5 a 50MHz (ALLEN; HOLBERG, 2002).

A grande desvantagem dos conversores *A/D flash* é o aumento do número de comparadores e complexidade do codificador à medida que se aumenta a resolução, isso ocasiona um enorme aumento na área de silício utilizada e consumo de potência, devido ao número elevado de componentes (ALLEN; HOLBERG, 2002).

3.3.2 Conversor A/D Sigma-Delta

Os conversores baseados na modulação sigma-delta(Σ-Δ) são conversores sobreamostrados, ou seja, operam em frequências muito superiores à frequência de *Nyquist*. Sua estrutura, indicada na Figura 11 inclui um modulador Σ-Δ, um filtro digital, responsável pela remoção do ruído localizado fora da largura de banda de interesse e um decimador, cuja finalidade é reduzir a taxa de dados de saída de volta à taxa de *Nyquist*, funcionando assim como um filtro passa-baixas.

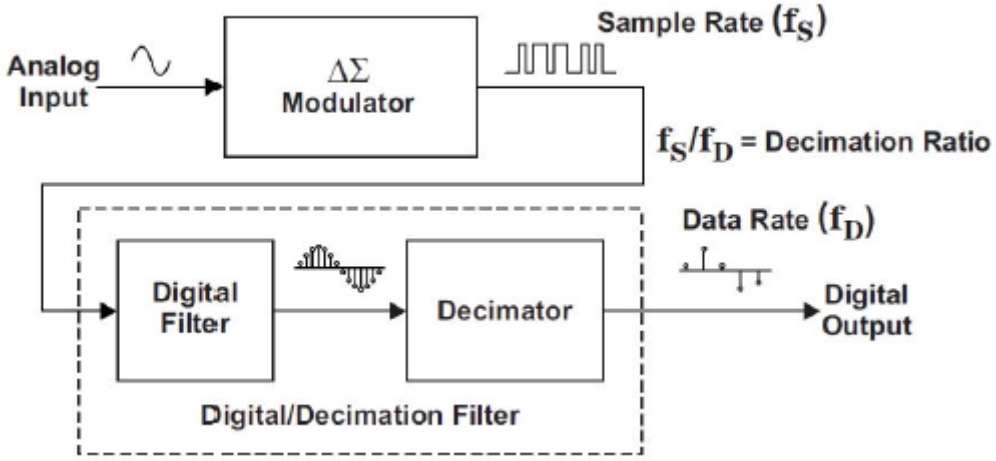


Figura 11 – Diagrama de blocos de um *ADC* Sigma-Delta (BAKER, 2011).

A modulação sigma-delta faz com que o ruído de quantização seja jogado em altas frequências no espectro e não espalhado por todo ele como ocorre em outros projetos. Esse efeito é chamado de *noise-shaping* e serve para reduzir o ruído de quantização dentro da banda do sinal.

A sobreamostragem junto com o processo de decimação faz com que esse tipo de conversor atinja uma alta resolução. O *ADC* Σ-Δ é bastante utilizado pois apresenta como vantagens a alta precisão, alta resolução além da menor sensibilidade às imperfeições dos circuitos analógicos. Por outro lado ele aplica-se apenas em situações onde não seja necessária uma alta velocidade, uma vez que o conversor é relativamente lento se comparado com outras topologias (SCHREIER; TEMES et al., 2005).

3.3.3 Fundamentos Teóricos da Modulação Sigma-Delta

Como dito anteriormente, o modulador sigma-delta é baseado em *oversampling* e implementa o conceito de *noise shaping*. A técnica de *oversampling* trata-se da amostragem do sinal analógico com frequência muito maior que a frequência de amostragem de *Nyquist* como forma de evitar o fenômeno de *aliasing*. Então a frequência de amos-

tragem (fs) deve ser uma frequência de amostragem $Nyquist$ muito maior. Isso fica claro na fórmula 3.8.

$$OSR = \frac{fs}{2fb} \tag{3.8}$$

Sendo OSR ($Oversampling\ Rate$) a taxa de $oversampling$, fs a frequência de amostragem e fb a frequência do sinal.

A sobreamostragem reduz a potência de ruído de quantização fixa, propagando-a sobre uma largura de banda muito maior do que a banda de sinal (LI; HE, 2007).

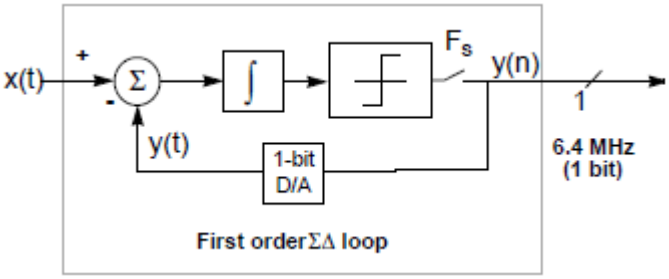
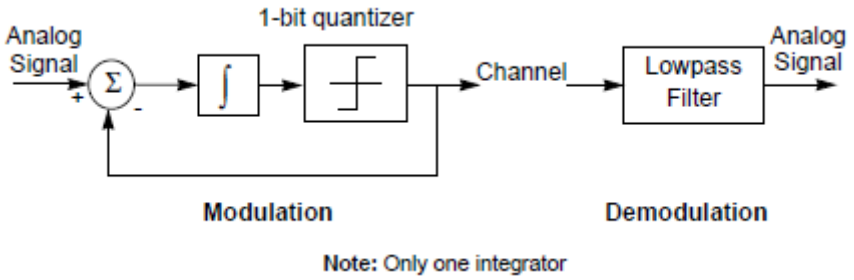


Figura 12 – Diagrama de blocos do modulador Σ-Δ de 1ª ordem (PARK, 1999).

A Figura 12 mostra o diagrama de blocos de um modulador Σ-Δ de primeira ordem. Ele é constituído por um nó de diferença analógico, um integrador, um quantizador de 1 bit (conversor A/D) e um conversor D/A de 1 bit num sistema realimentado. A saída do modulador tem apenas 1 bit de informação. A entrada para o integrador no modulador é a diferença entre o sinal de entrada $x(t)$ e o valor de saída quantizado $y(n)$ convertido de volta para o sinal analógico previsto. Idealmente, essa diferença entre o sinal de entrada $x(t)$ e o sinal de retorno $y(t)$ na entrada do integrador é igual ao erro de quantização. Este erro é somado no integrador e depois quantizado pelo conversor A/D de 1 bit (PARK, 1999).



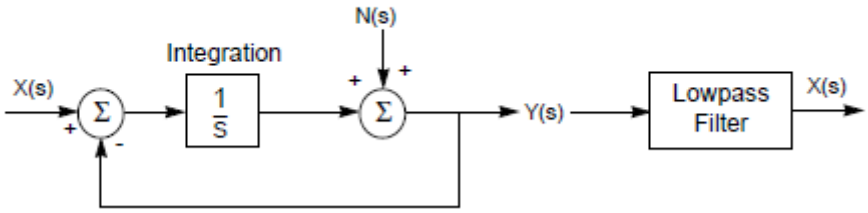


Figura 13 – Diagrama de blocos do modulador Σ-Δ (PARK, 1999).

Analisando a Figura 13, nota-se que quando o ruído gerado pelo quantizador é nulo, $N(S) = 0$, a função de transferência será:

$$\frac{Y(S)}{X(S)} = \frac{\frac{1}{S}}{1 + \frac{1}{S}} = \frac{1}{1 + S} \tag{3.9}$$

Que é na verdade um filtro passa-baixas. O sinal então permanece inalterado desde que a sua frequência não exceda a frequência de corte do filtro. De forma parecida, quando o sinal de entrada é nulo, $X(S) = 0$, a função de transferência do ruído será:

$$\frac{Y(S)}{N(S)} = \frac{1}{1 + \frac{1}{S}} = \frac{S}{1 + S} \tag{3.10}$$

Que se comporta como um filtro passa-altas, indicando que o ruído foi empurrado para uma banda de frequência mais alta. Ou seja, neste processo de formação de ruído pelo modulador Σ-Δ, ele empurra a potência do ruído de quantização da banda do sinal para frequências mais elevadas.

A Figura 14 mostra as formas de onda de entrada de saída para um modulador Σ-Δ de 1ª ordem. Quando a entrada senoidal está mais próxima de uma escala maior, a saída é positiva durante a maioria dos ciclos de *clock*. O mesmo é válido quando a senoide está mais próxima da escala máxima negativa. Quando a entrada é próxima de zero, o valor da saída do modulador varia rapidamente entre uma escala máxima e mais baixa, com média aproximadamente zero (PARK, 1999).

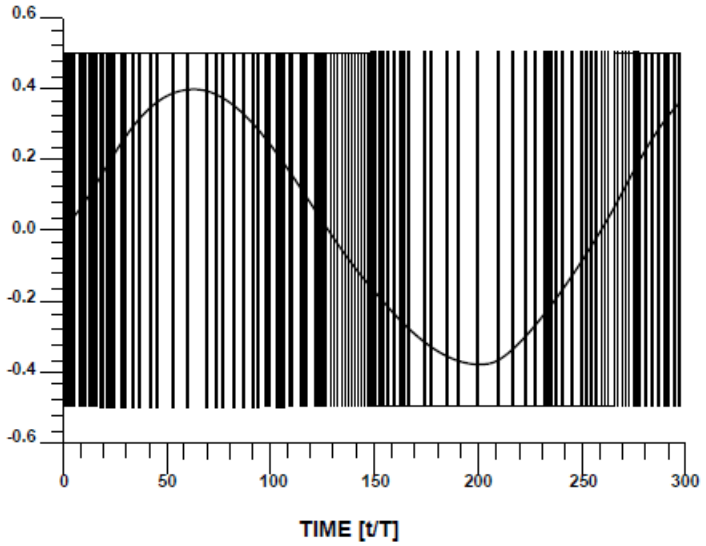


Figura 14 – Entrada e saída de um modulador Σ-Δ *single-bit* de 1ª ordem (PARK, 1999).

Os moduladores Σ-Δ podem ser classificados com *single-bit* ou *multi-bit* de acordo com o número de bits do seu quantizador. O modulador da Figura 12 trata-se de um modulador *single-bit*, já o modulador *multi-bit* será apresentado na próxima seção.

3.3.3.1 Técnica de Quantização Multi-bit

A principal vantagem dos moduladores de um bit (*single-bit*) além da simplicidade do circuito, é a linearidade do DAC, essa linearidade no $feedback$ é importante porque a saída do DAC é subtraída diretamente do sinal de entrada. Qualquer distorção, não-linearidade ou ruído na saída do DAC afetará diretamente o desempenho do modulador (BAKER, 2008). No entanto, o quantizador de um único bit possui um grande ruído de quantização e precisa de um *oversampling* maior para suprimir esse ruído (HAN et al., 2011).

O modulador *multi-bit* apresenta como vantagens uma maior precisão na conversão, a redução do ruído de quantização e uma melhor estabilidade do sistema. As desvantagens do uso de topologias *multi-bit* são o aumento da complexidade do ADC e a necessidade de um DAC preciso (BAKER, 2008). A Figura 15 mostra o diagrama de blocos de um modulador *multi-bit*.

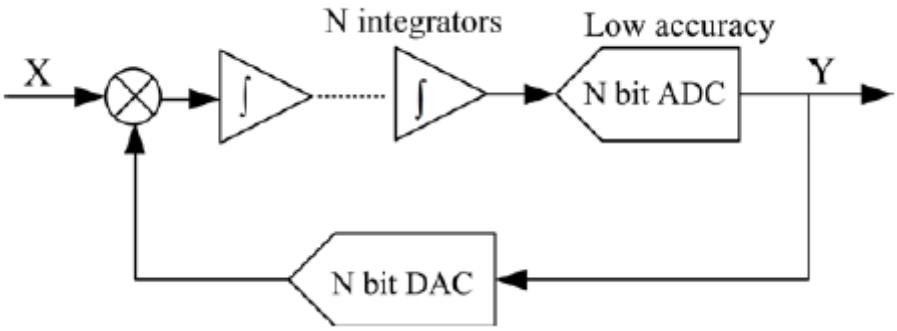


Figura 15 – Diagrama de blocos de um modulador Σ-Δ *multi-bit* (ZIQUAN et al., 2013).

A Figura 16 mostra uma implementação em nível de circuito de um modulador Σ-Δ multi-bit de 1ª ordem, usando um *ADC* e *DAC* de 4 bits. A Figura 17 mostra as saídas de simulação deste modulador nos domínios de tempo e frequência.

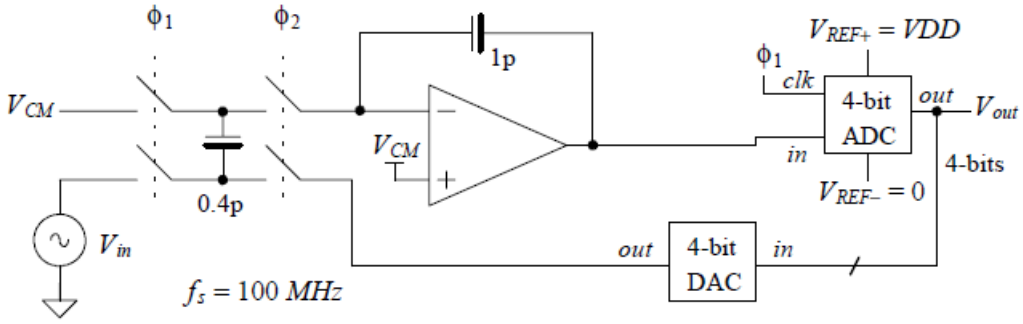


Figura 16 – Circuito de um modulador *noise shaping* multi-bit de 1ª ordem (BAKER, 2008).

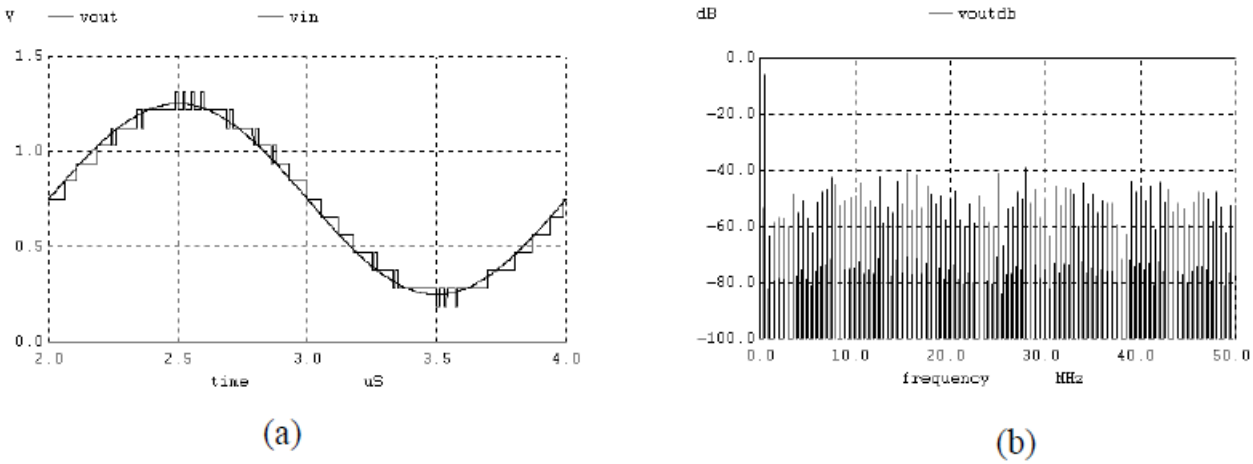


Figura 17 – Saída do modulador multi-bit no domínio do tempo a) e no domínio da frequência b) (BAKER, 2008).

4 Modelagem de Processos de Quantização

Neste capítulo será mostrado como a transformada da incerteza ($Unscented\,Transform$ - UT) pode ser entendida como uma estrutura para o projeto de quantizadores com aplicações em conversão de dados.

4.1 A Transformada da Incerteza

A UT foi desenvolvida para resolver problemas de linearização e apresentar uma forma mais direta e explícita para a informação e transformação de média e covariância (UHLMANN, 1994). Ela foi criada para realizar o cálculo das estatísticas de uma variável aleatória que sofreu uma transformação não-linear. A ideia geral da UT mostra que é mais fácil aproximar uma distribuição de probabilidade do que aproximar uma função ou transformação de uma função não linear arbitrária (MERWE; DOUCET, 2000). Essa aproximação pode ser visualizada na Figura 18.

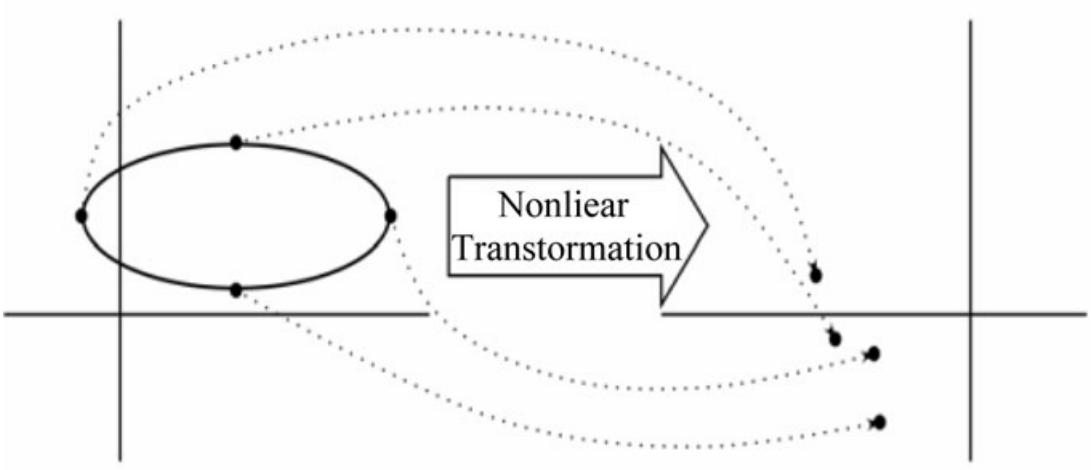


Figura 18 – Princípio da UT (ALI; ZOHDY, 2012).

Um conjunto de pontos sigma (S), deterministicamente escolhidos, é selecionado de tal forma que sua média seja $\bar{x}$ e sua covariância seja $\sum x$. A função não-linear é aplicada em cada ponto para se obter uma nuvem de pontos transformados e a estatística dos pontos transformados pode ser calculada para a formação de uma estimativa da média e covariância transformadas de forma não linear.

A UT é uma estrutura matemática que modela uma função de densidade de probabilidade contínua, $py(y)$, em uma discreta, $pyq(yq)$. (MEDEIROS; HADDAD, 2017) propõe a utilização da UT como modelo para o processo de quantização em conversores de dados. A Figura 19 ilustra essa ideia na qual projetou-se um quantizador que imita as

propriedades UT quando modela seus sinais de entrada e saída por meio de funções de probabilidade.

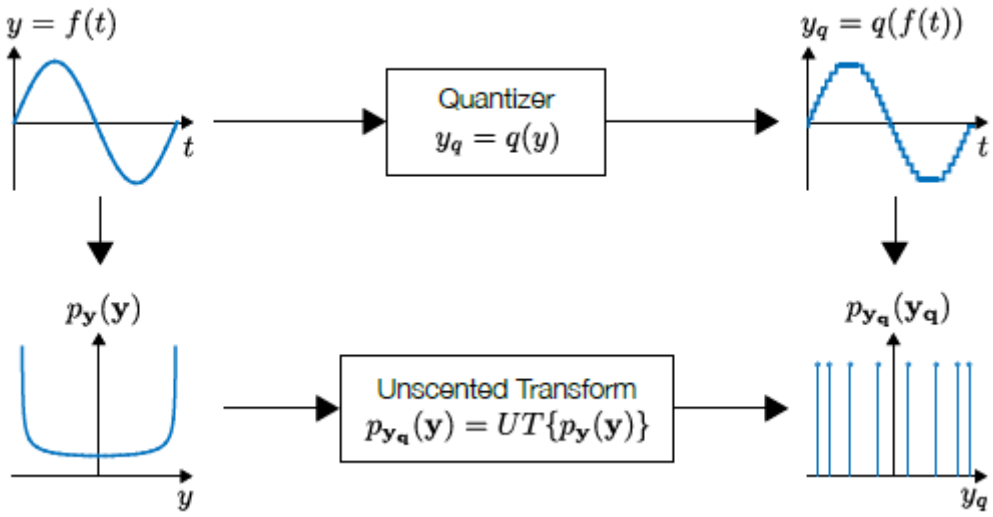


Figura 19 – Modelagem de um sinal contínuo como uma função de probabilidade. Reprodução autorizada pelo autor (MEDEIROS; HADDAD, 2017).

A Figura 19 mostra a ligação entre a UT e a quantização de preservação de momento. Um quantizador tomou um sinal de entrada $y = f(t)$ e o mapeou em um sinal quantizado $yq = q(y)$. A UT, por outro lado, assumiu uma função de probabilidade contínua $py(y)$ e a mapeou em uma função de probabilidade discreta $Pyq(yq)$.

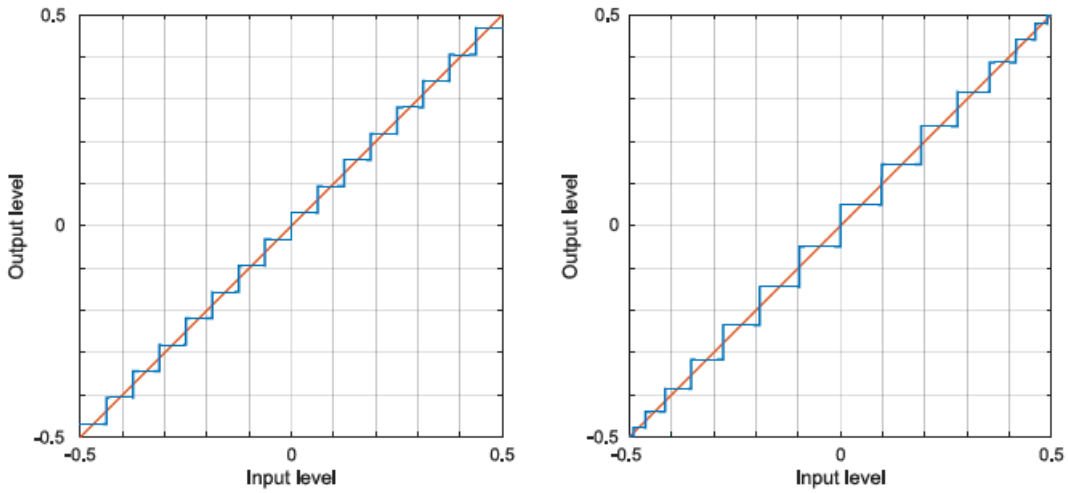
Segundo (PAPOULIS; PILLAI, 2002), um sinal senoidal pode ser modelado por meio de uma distribuição de arco-seno. É importante notar que esta análise é válida para qualquer frequência à medida que o tempo é abstraído no processo de modelagem. Os sinais laterais esquerdos mostrados na Figura 19 ilustram essa relação.

4.2 A Análise do Quantizador Arco-seno

Essa seção é baseada no trabalho de (MEDEIROS; HADDAD, 2017), que é o ponto de partida para as análises e implementações que serão feitas adiante. Nela será analisado um quantizador baseado na distribuição arco-seno a ser implementado em um sistema de conversão de dados. Para isso, considerou-se um sinal de entrada com amplitude unitária. Comparou-se o quantizador de distribuição do arco-seno com o quantizador uniforme caracterizado por pontos de transição igualmente espaçados (RAZAVI, 1995) que será chamada de quantizador linear.

Primeiro, (MEDEIROS; HADDAD, 2017) mostrou na Figura 20 as curvas características de entrada e saída para o quantizador linear e para o quantizador de distribuição arco-seno. O quantizador linear é caracterizado por passos de quantização constantes.

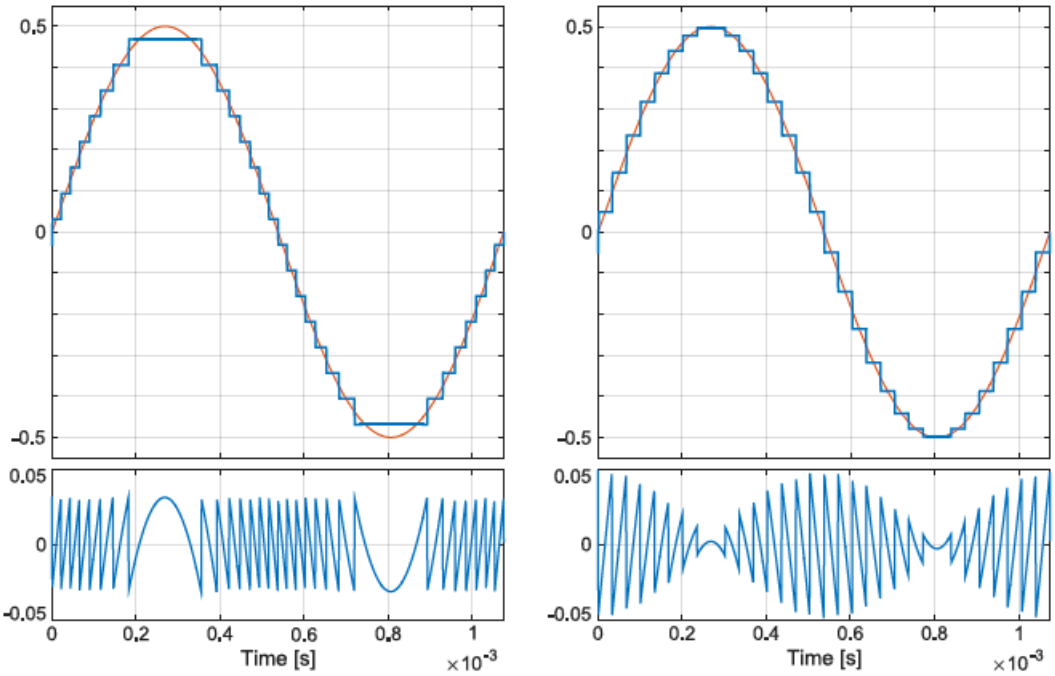
Nota-se que o quantizador arco-seno tem passos de quantização menores e, portanto, melhor resolução nas extremidades do intervalo de entrada. Por outro lado, apresenta maiores passos de quantização e, portanto, resolução menor no centro da faixa dinâmica de entrada.



(a) Linear quantizer. (b) Arcsine quantizer.

Figura 20 – Curvas características para os quantizadores: a) Linear; b) Com distribuição arco-seno. Reprodução autorizada pelo autor (MEDEIROS; HADDAD, 2017).

A Figura 21 mostra em uma análise transiente o efeito dos diferentes esquemas de quantização nos sinais senoidais e o erro de quantização para os dois casos.

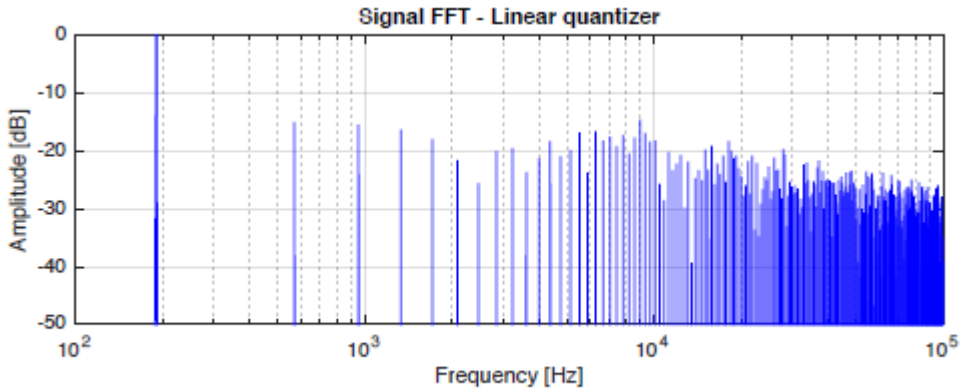


(a) Linear quantizer. (b) Arcsine quantizer.

Figura 21 – Análise transiente para um sinal de entrada senoidal processado pelos quantizadores: a) Linear; b) Com distribuição arco-seno. Reprodução autorizada pelo autor (MEDEIROS; HADDAD, 2017).

Na Figura 21-a, para o quantizador linear, obteve-se o padrão clássico de erro de quantização. O quantizador de distribuição arco-seno , no entanto, apresenta um comportamento modular onde tem-se menor erro de quantização perto dos extremos do sinal de entrada e maior erro perto do cruzamento com zero. No caso da senoide, isto significa que o quantizador está optando por introduzir menos erro onde o sinal apresenta uma variação mais lenta e assim pode-se esperar ter menos ruído de quantização para frequências mais baixas (MEDEIROS; HADDAD, 2017).

Baseado nos resultados encontrados em sua tese, (MEDEIROS; HADDAD, 2017) sugere que o espectro em frequência do sinal quantizado pela distribuição arco-seno será semelhante ao de uma modulação por amplitude de pulso (*Pulse Amplitude modulation - PAM*). No caso de um sinal de entrada senoidal, a análise do espectro conta com um impulso em $f0$, a frequência do sinal, com duplos impulsos centrados em nfs e espaçados por $2f0$ como mostrado na Figura 22 para um sinal com $f0 = 190.7Hz$ e um quantizador de 4 bits.



(a) Linear quantizer output spectrum.

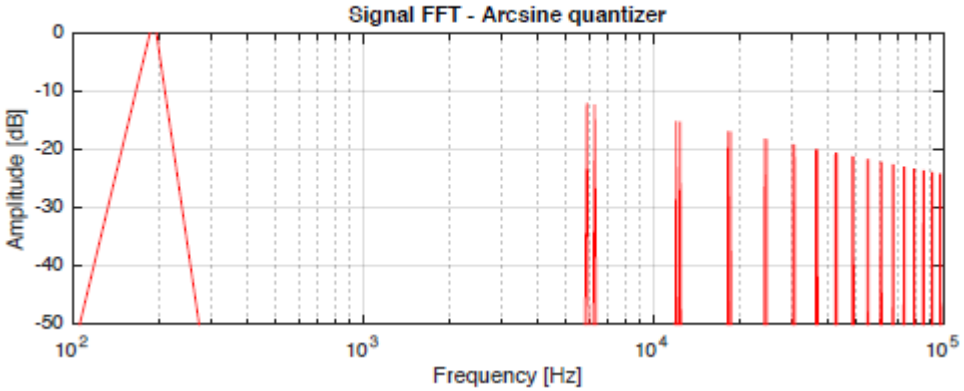


Figura 22 – Análise espectral para um sinal quantizado com $f0 = 197.7Hz$: a) Quantizador linear 4-bits; b) Quantizador arco-seno 4-bits. Reprodução autorizada pelo autor (MEDEIROS; HADDAD, 2017).

A Figura 22 apresenta o espectro para o quantizador linear e arco-seno. Observa-se que no quantizador arco-seno os primeiros harmônicos importantes aparecem em uma frequência muito maior do que no quantizador linear, sugerindo que pode-se obter um espectro de frequência mais limpo e uma melhor reconstrução do sinal após o processo de amostragem no ADC utilizando-se um filtro passa-baixa.

4.3 Proposta de Implementação de Circuito

Para melhor compreensão do quantizador arco-seno, (MEDEIROS; HADDAD, 2017) apresentou em seu trabalho uma proposta de implementação baseado em uma arquitetura de conversor A/D paralela.

A Figura 23 mostra o diagrama de blocos do sistema, incluindo o ADC e o DAC. O ADC pode ser implementado por uma rede de escala de tensão / corrente que resultará em um conjunto de $N-1$ níveis de tensão que serão comparados com o sinal de entrada. Este processo resultará em um código termômetro que pode ser codificado. O DAC pode ser implementado usando o mesmo esquema.

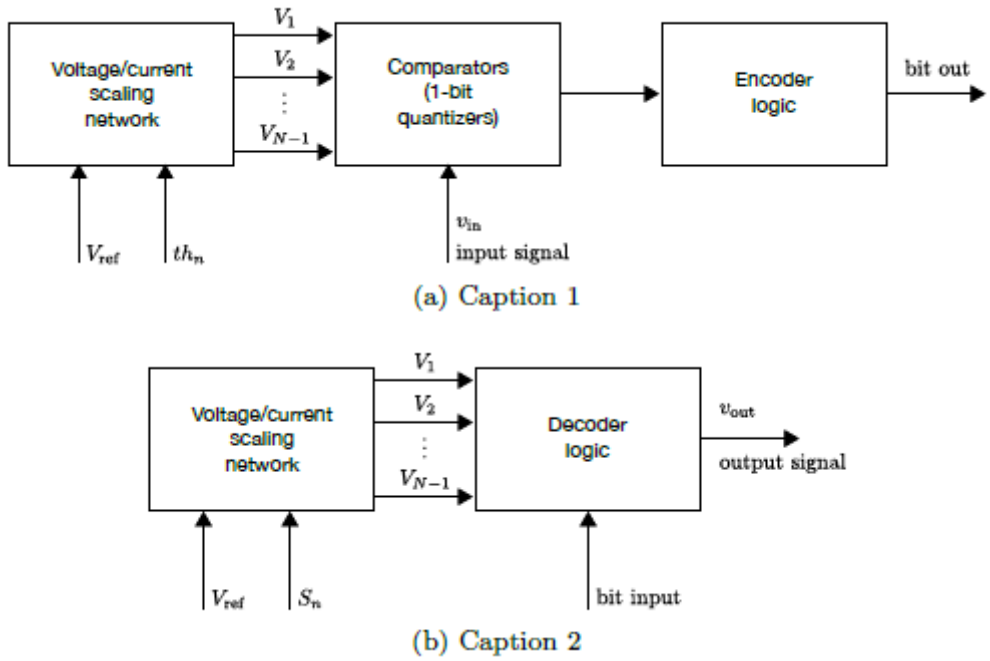


Figura 23 – Diagramas de blocos para implementação do quantizador arco-seno. Reprodução autorizada pelo autor (MEDEIROS; HADDAD, 2017).

A Figura 24-a mostra um esquema de circuito conceitual de um *ADC* paralelo. Nesse circuito, uma rede resistiva implementa a parte de dimensionamento de tensão. Para um quantizador de nível N, os valores das resistências serão dados por

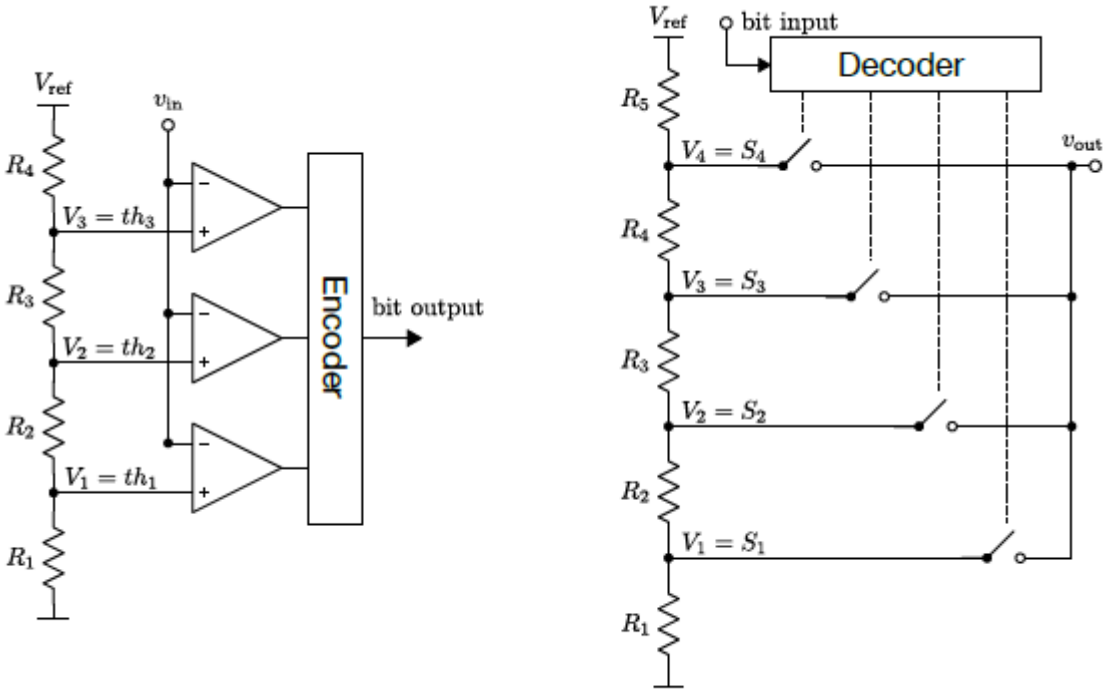
$$R_n = th_n \times R - \left(\sum_{k=0}^{n-1} R_k \right), \tag{4.1}$$

para todo n =1, 2,..., N, onde R é um resistor de valor arbitrário , $R0 = 0$ e $th_n = 1$.

A Figura 24-b mostra um esquema de circuito conceitual de um *DAC* paralelo. Neste circuito, uma rede resistiva implementa a função de escala, agora considerando os níveis de saída desejados, Si. Nesse circuito, um *DAC* com N níveis terá os valores de resistência dados por:

$$R_n = S_n \times R - \left(\sum_{k=0}^{n-1} R_k \right), \tag{4.2}$$

para todo n =1, 2,..., $N + 1$, onde R é um resistor de valor arbitrário , $R0 = 0$ e $S_{N+1} = 1$.



(a) Caption 1 (b) Caption 2

Figura 24 – Proposta de implementação utilizando topologia $flash$:a) ADC; b)DAC. Reprodução autorizada pelo autor (MEDEIROS; HADDAD, 2017).

Parte II

Projeto, Implementação e Resultados

5 Modelagem do Quantizador Linear

Como dito anteriormente, será projetado um ADC e um DAC ideais de 4 bits com topologia do tipo $flash$ para validar o funcionamento dos quantizadores linear e não linear. A proposta de implementação, sugerida por (MEDEIROS; HADDAD, 2017) será modelada, utilizando-se a HDL Verilog-A.

Para melhor entendimento do trabalho serão projetados o quantizador linear e o quantizador arco-seno e serão feitas comparações entre os resultados obtidos.

Se tratando da arquitetura $flash$, para uma resolução de N bits são necessários $2^N - 1$ comparadores e igual quantidade de níveis de referência. Para o ADC de 4 bits desse projeto, foram utilizados 15 comparadores e 15 níveis de referência. Na saída dos comparadores colocou-se um circuito de codificação que recebe os sinais dos comparadores e codifica o sinal de saída em código binário.

Para o projeto do DAC, que tem como função converter uma grandeza digital em uma grandeza analógica, utilizou-se a topologia sugerida por (MEDEIROS; HADDAD, 2017), mostrada na figura 24 que conta com dois blocos simples, uma chave e um decoder.

Na sequência, cada um dos blocos pertencentes ao sistema serão apresentados a partir de sua descrição funcional, tabela de pinos e simulação.

A Tabela 2 é composta por alguns dos parâmetros que são comuns à maioria dos blocos modelados. Nota-se que a alimentação dos blocos é simétrica de 1 V e os tempos de transição e de *delay* dos modelos são de 1 fs, a fim de estabelecer um modelo mais próximo do ideal.

Tabela 2 – Parâmetros de simulação comuns aos blocos.

Parâmetro	**Descrição**	**Valor**
avdd	Tensão de referência positiva	1.0 V
avss	Tensão de referência negativa	-1 V
tf	Tempo de transição de subida e descida do modelo	1fs
td	Tempo de atraso do modelo	1fs

5.1 Comparador

Um circuito comparador tem a função de comparar dois sinais distintos. No caso dessa aplicação, o comparador é síncrono e irá comparar o sinal de entrada Vin com uma tensão de referência $Vref$ na borda de subida do clock. Se a tensão de entrada for maior

que a tensão de referência o dispositivo ficará saturado e a saída será *avdd*, caso ocorra o inverso, devido a mesma saturação em sentido inverso, a saída será *avss*.

5.1.1 Descrição do Bloco

A descrição em Verilog-A deste bloco é bastante simples. Seu comportamento é baseado no uso da função *cross*, que detecta o cruzamento por 0 e nesse caso a borda de subida do clock. Desta forma, a saída $V(out)$ vai para nível alto quando $V(in)$ é maior ou igual a $V(ref)$ e para nível baixo caso contrário.

```
analog begin
        @(cross((V(Clk)-vth), +1)) begin
        if (V(in)>=V(ref))
                state = V(vdd);
        else
            state= V(vss);
        V(out) <+ transition(state, td, tf);
end
```

5.1.2 Descrição dos Pinos

Tabela 3 – Descrição dos pinos do comparador.

Pino	**Descrição**	**Tipo**
vin	Tensão do sinal de entrada	*input*
vref	Tensão de referência	*input*
vout	Tensão de saída	*output*
gnd	ground (0V)	*inout*
vdd	Tensão de alimentação (1V)	*inout*
clk	Clock	*input*

5.1.3 Simulação

A simulação e os parâmetros utilizados são apresentados a seguir. As ondas de saída, apresentadas na Figura 26, estão de acordo com o comportamento esperado para o circuito, validando assim a modelagem deste bloco.

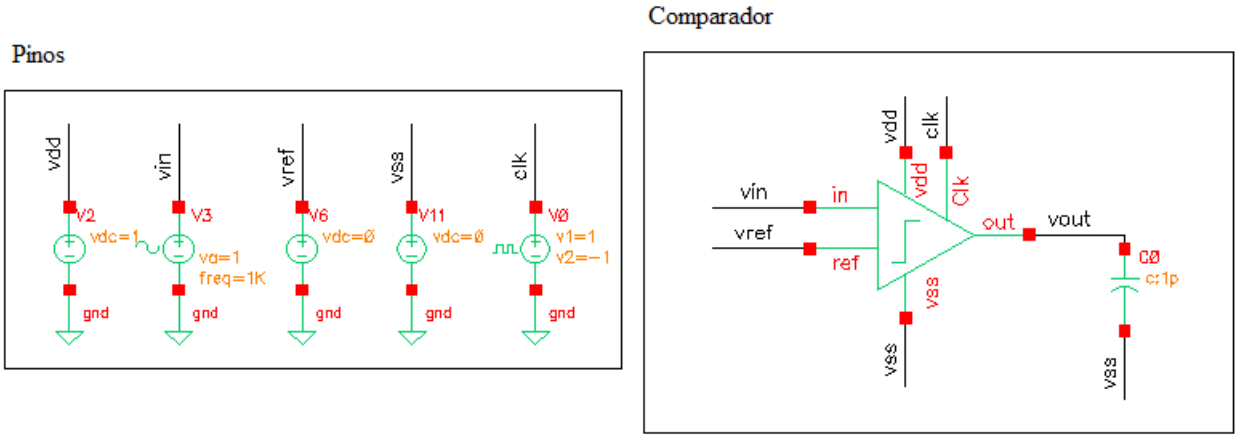


Figura 25 – Testbench do comparador.

Tabela 4 – Parâmetros de simulação do comparador.

Parâmetro	**Descrição**	**Valor**
frequência	Frequência de operação da fonte de teste	1KHz
ciclos	Quantidade de ciclos utilizados na simulação transiente	3

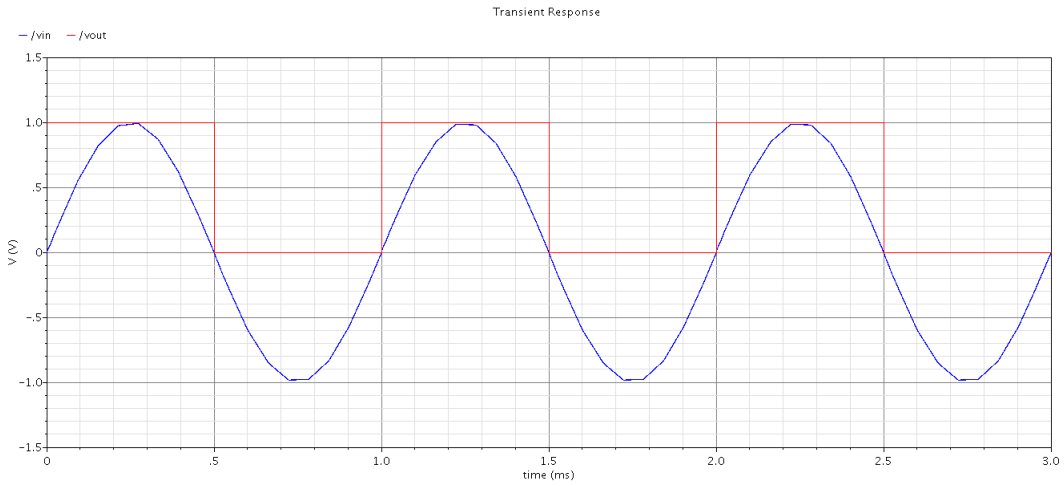


Figura 26 – Simulação do comparador.

5.2 Encoder

Um codificador ou encoder, é um circuito digital que faz a conversão de um número ou um código qualquer para um código binário. As saídas dos comparadores formam o chamado código termômetro. A Tabela 5 mostra este código para um conversor de 4 bits.

Tabela 5 – Códigos utilizados no projeto do encoder.

Nível	Código Termômetro	Código Binário
0	000000000000000	0000
1	000000000000001	0001
2	000000000000011	0010
3	000000000000111	0011
4	000000000001111	0100
5	000000000011111	0101
6	000000000111111	0110
7	000000001111111	0111
8	000000011111111	1000
9	000000111111111	1001
10	000001111111111	1010
11	000011111111111	1011
12	000111111111111	1100
13	001111111111111	1101
14	011111111111111	1110
15	111111111111111	1111

Projetou-se então um encoder de 4 bits que fizesse a conversão de código termômetro para código binário.

5.2.1 Descrição do Bloco

A descrição em Verilog-A deste bloco foi baseada em operações lógicas AND e OR, que consistem simplesmente em analisar os bits do código de entrada e gerar uma sequência de bits para a saída de acordo com a tabela 5. Abaixo encontra-se um pequeno trecho do código implementado, o código completo pode ser encontrado no apêndice.

```
1 analog begin
2 if ((V(t1)==V(agnd))&&(V(t2)==V(agnd))&&(V(t3)==V(agnd))&&(V(t4)
     ==V(agnd))&&(V(t5)==V(agnd))&&(V(t6)==V(agnd))&&(V(t7)==V(agnd)
     )&&(V(t8)==V(agnd))&&(V(t9)==V(agnd))&&(V(t10)==V(agnd))&&(V(
     t11)==V(agnd))&&(V(t12)==V(agnd))&&(V(t13)==V(agnd))&&(V(t14)==
      V(agnd))&&(V(t15)==V(agnd)))
3         begin
4                 done = V(agnd);
5                 dtwo = V(agnd);
6                 dthree = V(agnd);
7                 dfour= V(agnd);
8         end
9 end
```

5.2.2 Descrição dos Pinos

Tabela 6 – Descrição dos pinos do Encoder

Pino	Descrição	Tipo
$t1$ a $t15$	Bits de entrada que formam o código termômetro	*input*
$d1$ a $d4$	Bits de saída do código binário	*output*

Observando-se que os bits $t1$ e $d1$ são os mais significativos.

5.2.3 Simulação

A simulação e os parâmetros utilizados são apresentados a seguir. Para realizar o *testbench* do encoder, aplicou-se um código termômetro na entrada, simulado por fontes DC e observou-se o código gerado na saída. Na Figura 28 pode-se observar um dos testes feitos para o código de entrada "000000111111111"que corresponde ao nível 9 da Tabela 5.

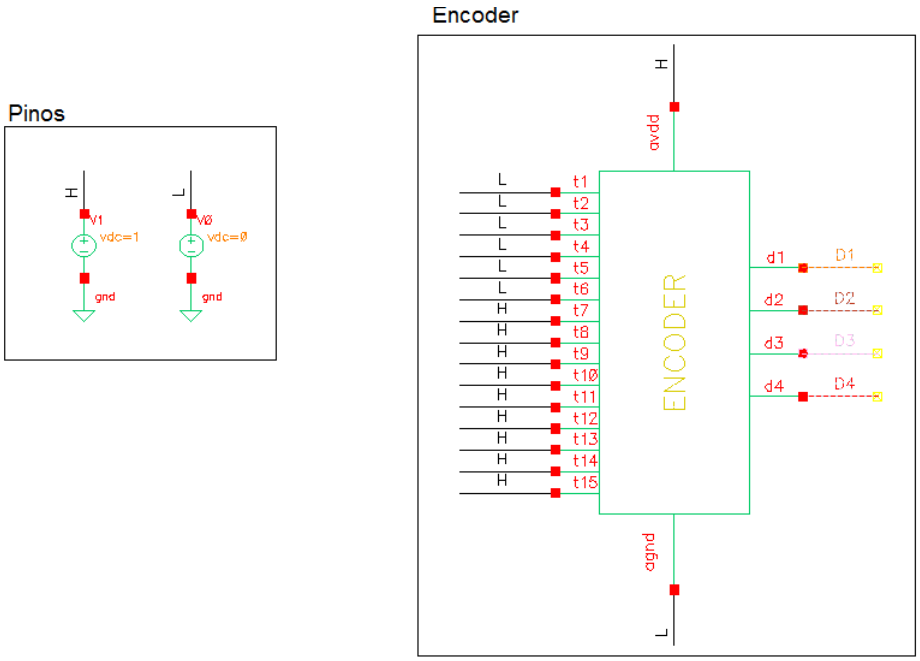


Figura 27 – *Testbench* do encoder.

Tabela 7 – Parâmetros de simulação do encoder.

Parâmetro	Descrição	Valor
t1 a t6	Tensão aplicada a entrada do encoder	0 V
t7 a t15	Tensão aplicada a entrada do encoder	1V

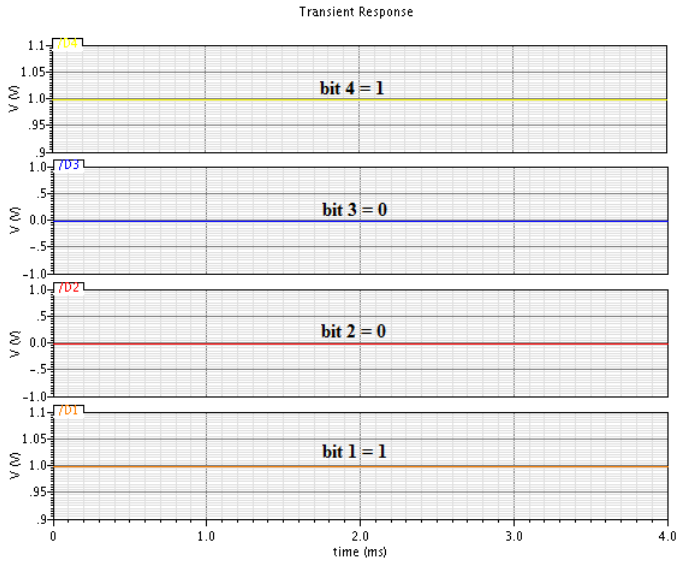


Figura 28 – Simulação do encoder para o código de entrada “000000111111111”.

5.3 *ADC Flash*

Para o projeto do *ADC*, conectou-se os comparadores modelados onde um dos pinos recebe o sinal de entrada e o outro a tensão de referência fornecida pela divisão de tensão nos resistores, sendo que valor dos resistores escolhidos para essa topologia foi de $10K$. Por se tratar de um quantizador linear, todas os resistores utilizados possuem o mesmo valor. Na saída dos comparadores colocou-se o encoder descrito na seção anterior, completando assim o esquemático do *ADC flash*.

Na Figura 29 encontra-se parte do esquemático do *ADC flash* de 4 bits projetado.

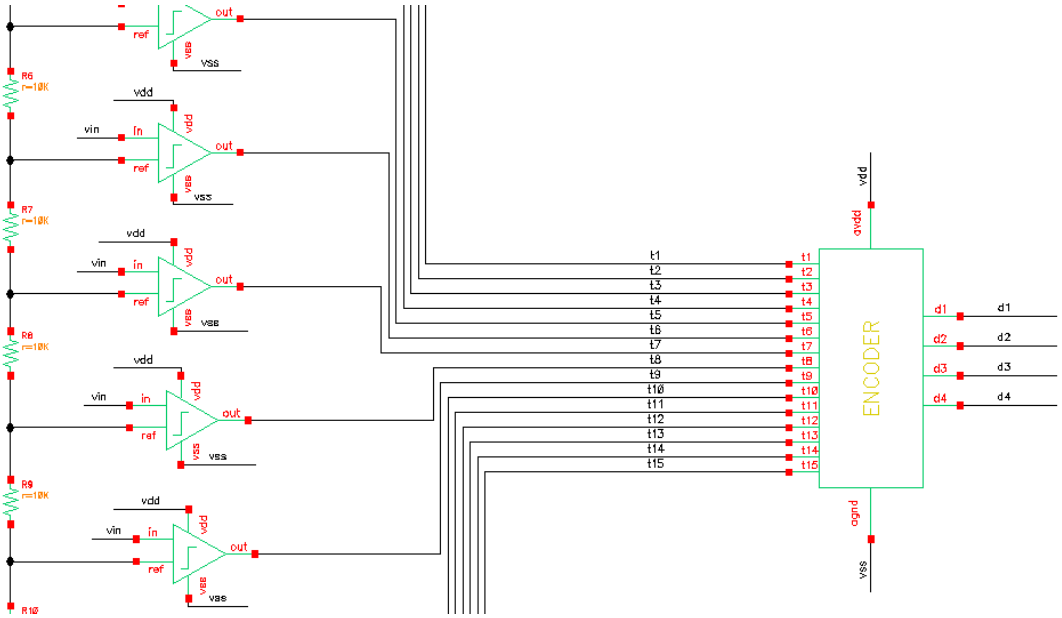


Figura 29 – Parte do esquemático do *ADC flash* de 4 bits.

5.3.1 Descrição dos Pinos

Tabela 8 – Descrição dos pinos do *ADC*

Pino	Descrição	Tipo
Vin	Sinal de entrada aplicado no conversor	*input*
d1 a d4	Bits de saída gerados pelo conversor	*output*

5.3.2 Simulação

Para a simulação do *ADC*, montou-se o esquemático de *testbench* da Figura 30 e colocou-se na entrada um sinal senoidal com $offset$ de $500mV$ e $1Vpp$ e observou-se a saída, que gera um código binário para cada nível de tensão como mostra a Figura 31.

Tabela 9 – Parâmetros de simulação do *ADC*.

Parâmetro	Descrição	Valor
frequência	Frequência de operação da fonte de teste de entrada	1KHz
ciclos	Quantidade de ciclos utilizados na simulação transiente	3

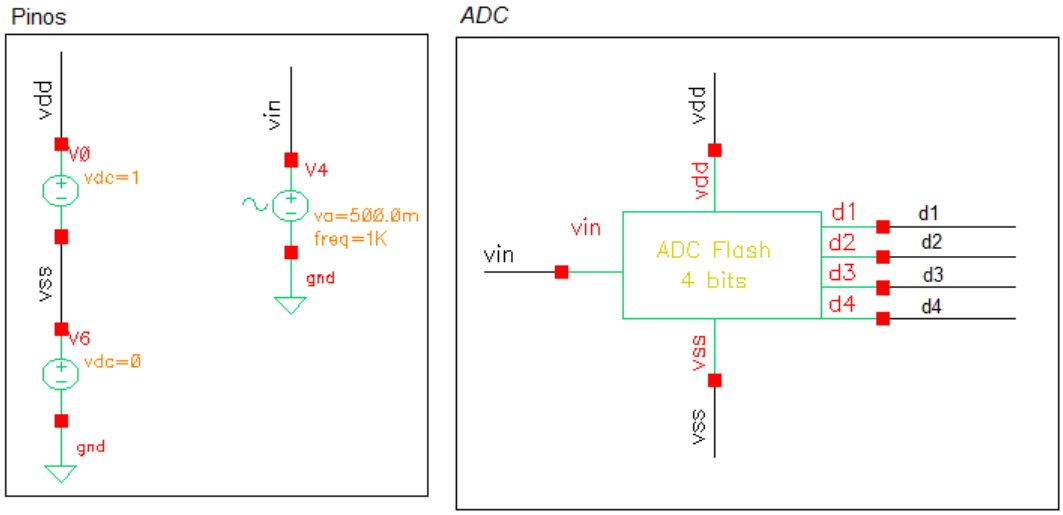


Figura 30 – *Testbench* do *ADC*.

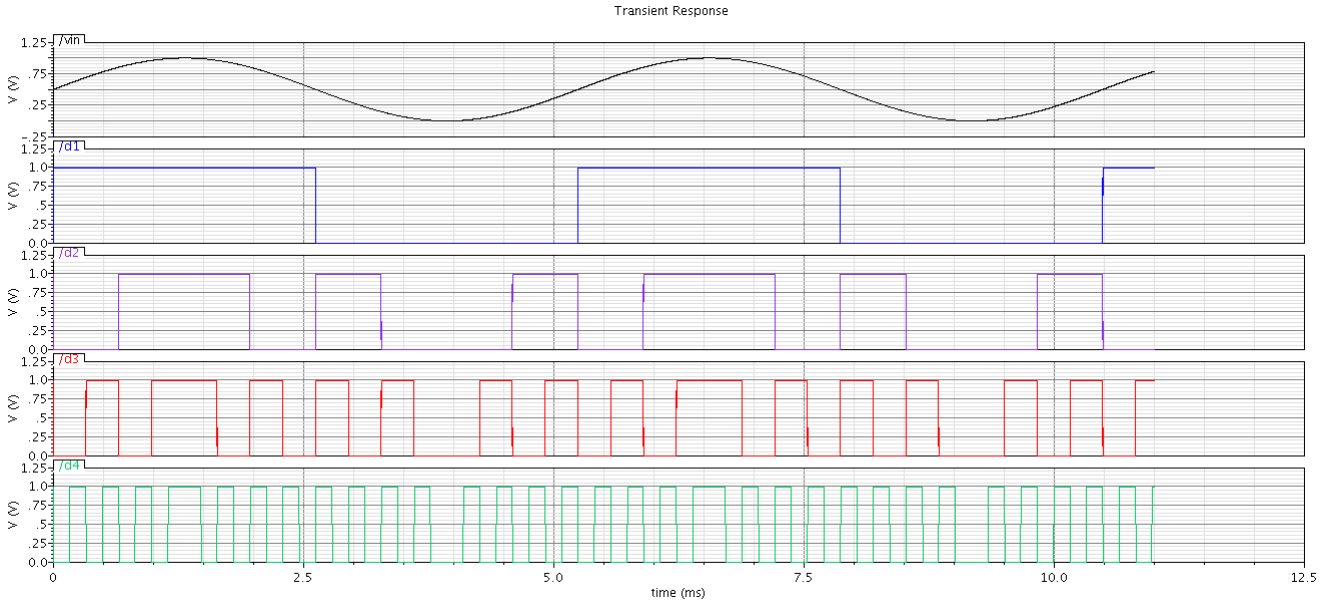


Figura 31 – Simulação do *ADC*.

Também foram feitos testes colocando-se fontes DC na entrada e observando se as saídas dos comparadores e encoder eram coerentes com a tensão de entrada.

5.4 Decoder

Um decodificador ou decoder, é um circuito digital que faz a conversão de um código binário para um código qualquer. A saída do *ADC* projetado gera um código binário. Para converter esse código novamente para um sinal analógico é necessário um decodificador que converta o código binário em um código $One - hot$, liberando assim a tensão adequada para os níveis do quantizador. Projetou-se então um decoder que atende essa necessidade. A Tabela 10 mostra esse código para o conversor de 4 bits.

Tabela 10 – Códigos utilizados no projeto do decoder.

Nível	Código Binário	Código One-hot
0	0000	0000000000000001
1	0001	0000000000000010
2	0010	0000000000000100
3	0011	0000000000001000
4	0100	0000000000010000
5	0101	0000000000100000
6	0110	0000000001000000
7	0111	0000000010000000
8	1000	0000000100000000
9	1001	0000001000000000
10	1010	0000010000000000
11	1011	0000100000000000
12	1100	0001000000000000
13	1101	0010000000000000
14	1110	0100000000000000
15	1111	1000000000000000

5.4.1 Descrição do Bloco

Assim como no encoder, a descrição em Verilog-A deste bloco foi baseada em operações lógicas AND e OR, que consistem simplesmente em analisar os bits do código de entrada e gerar uma sequência de bits para a saída de acordo com a Tabela 10. Abaixo encontra-se um pequeno trecho do código implementado, o código completo pode ser encontrado no apendice.

```
11 analog begin
12         if ((V(d1) == V(agnd))&&(V(d2) == V(agnd))&&(V(d3) == V(
               agnd))&&(V(d4) == V(agnd)))
13         begin
14         tone =          V(agnd);
15         ttwo=           V(agnd);
16         tthree=         V(agnd);
17         tfour=          V(agnd);
18         tfive=          V(agnd);
19         tsix=           V(agnd);
20         tseven=         V(agnd);
21         teight=         V(agnd);
22         tnine=          V(agnd);
23         tten=           V(agnd);
24         televen=        V(agnd);
25         ttwelve=        V(agnd);
```

```
26          tthirteen=      V(agnd);
27          tfourteen=      V(agnd);
28          tfifteen=       V(agnd);
29          end
30  end
```

5.4.2 Descrição dos Pinos

Tabela 11 – Descrição dos pinos do Decoder

Pino	Descrição	Tipo
$d1$ a $d4$	Bits de entrada que formam o código binário	*input*
$t1$ a $t15$	Bits de saída do código One-hot	*output*

5.4.3 Simulação

Para realizar o *testbench* do decoder, aplicou-se um código binário na entrada, simulado por fontes DC e observou-se o código gerado na saída. Na figura 12 pode-se observar um dos testes feitos para o código de entrada "1010"que corresponde ao nível 10 da tabela 10.

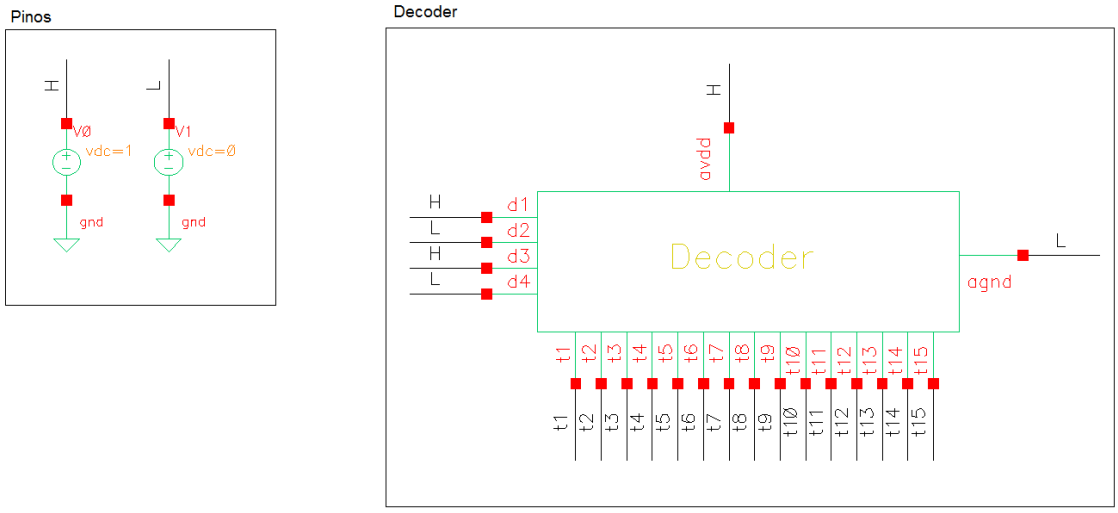


Figura 32 – *Testbench* do decoder.

Tabela 12 – Parâmetros de simulação do decoder.

Parâmetro	Descrição	Valor
d2 e d4	Tensão aplicada a entrada do decoder	0 V
d1 e d3	Tensão aplicada a entrada do decoder	1V

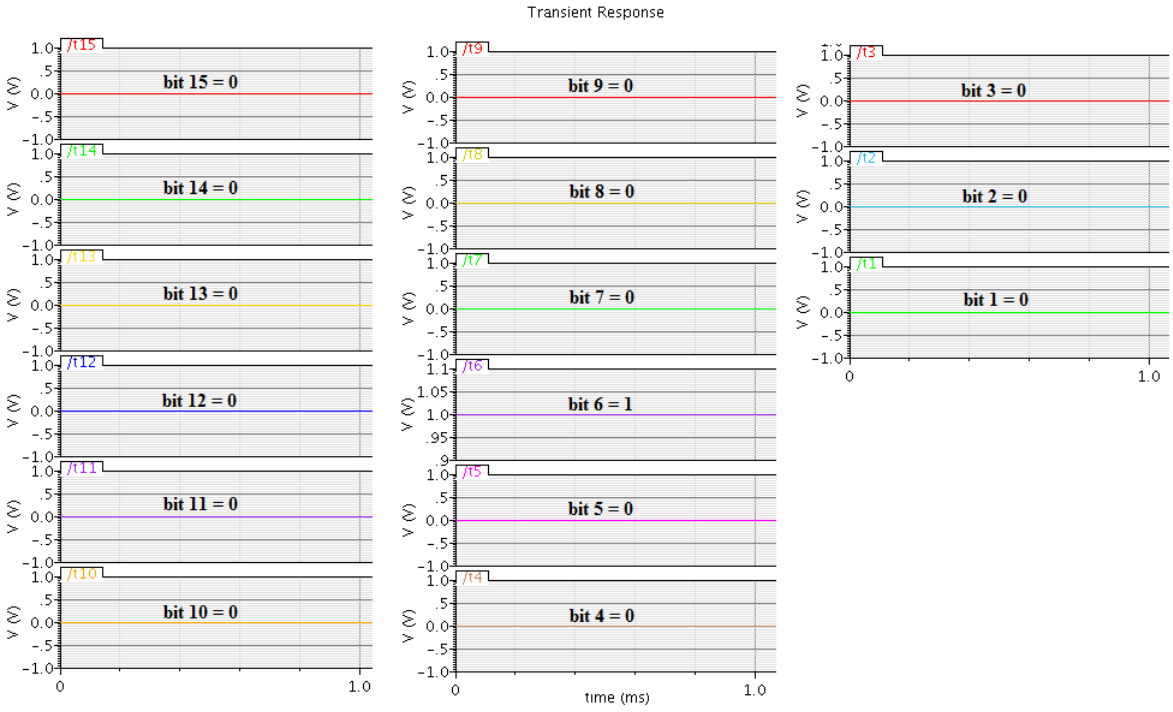


Figura 33 – Simulação do decoder para o código de entrada “1010”.

No projeto do DAC modelou-se também uma chave ideal para fazer a distribuição dos níveis de tensão vindos dos resistores. O *testbench* do DAC completo foi realizado conectando-se todos os blocos e fazendo a conversão do sinal de analógico para digital e novamente para analógico validando assim também, o processo de conversão de dados. Essa simulação será vista na próxima seção.

5.5 Quantizador Completo

Após a modelagem e simulação funcional de todos os blocos do sistema, conectou-se o $ADC\ flash$ ao DAC e esse a um capacitor de $1pF$ como carga, afim de simular o quantizador linear.

5.5.1 Simulação

Para a simulação do quantizador, aplicou-se um sinal senoidal e do tipo rampa na entrada do sistema, com os parâmetros especificados nas Tabelas 13 e 14, e observou-se a saída.

Como pode-se observar na Figura 35 e na Figura 36 em ambos os sinais os passos de quantização são constantes, o que dá a característica de linearidade ao sistema.

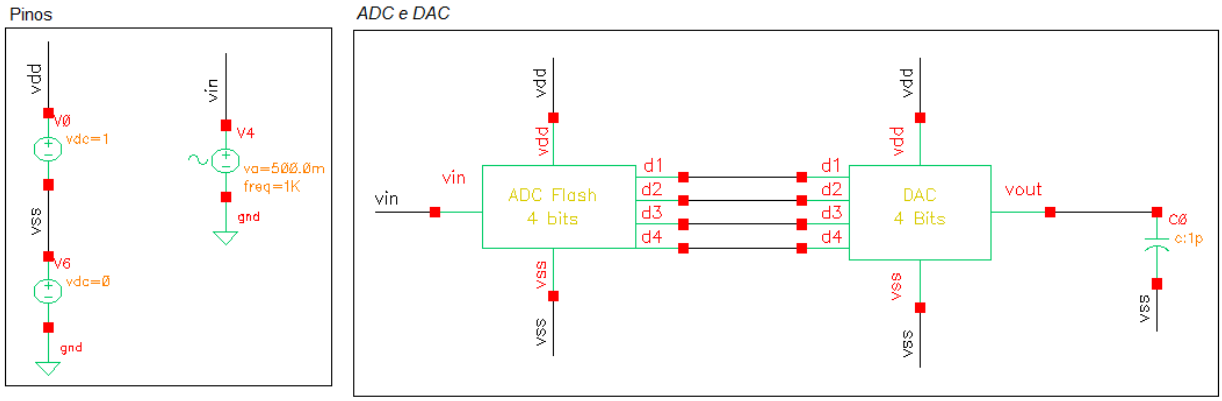


Figura 34 – *Testbench* do quantizador.

Tabela 13 – Parâmetros de simulação do conversor com sinal de entrada senoidal

Parâmetro	**Descrição**	**Valor**
frequência	Frequência de operação da fonte de teste senoidal	48.828 Hz
ciclos	Quantidade de ciclos utilizados na simulação	2

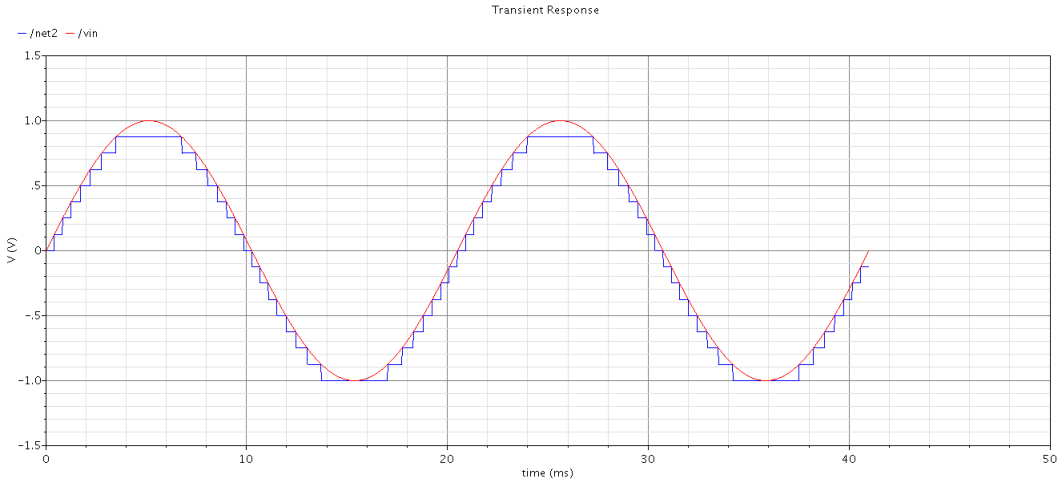


Figura 35 – Simulação do quantizador linear para um sinal senoidal.

Tabela 14 – Parâmetros de simulação do conversor com sinal de entrada rampa.

Parâmetro	**Descrição**	**Valor**
Tensão 1	Primeira tensão do sinal	0
Tensão 2	Segunda tensão do sinal	1
tr	tempo de subida do sinal	10 ms
tf	tempo de descida do sinal	1 ms
ciclos	Quantidade de ciclos utilizados na simulação	1

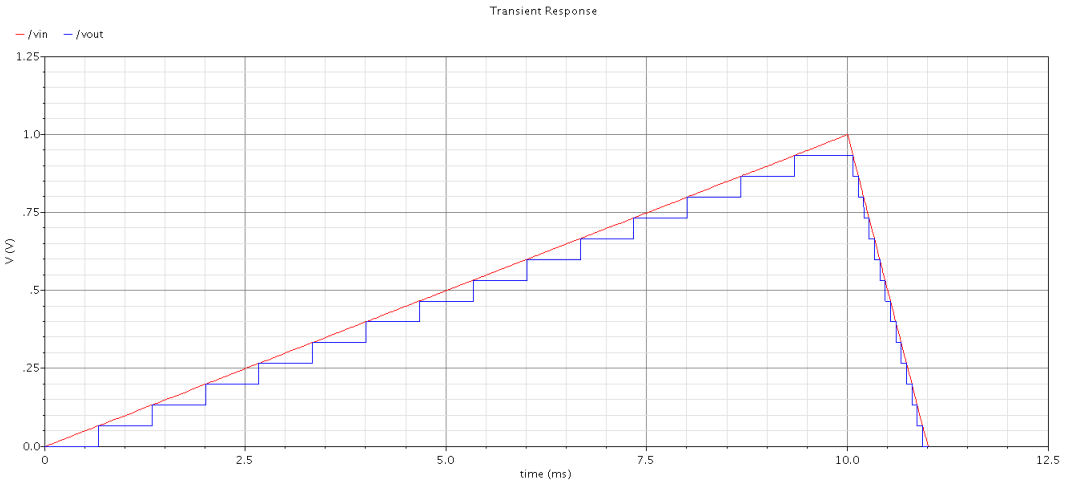


Figura 36 – Simulação do conversor $flash$ para um sinal do tipo rampa.

6 Modelagem do Quantizador Arco-Seno

O procedimento de desenvolvimento do modelo do ADC e do DAC paralelos para aplicação no quantizador com distribuição arco-seno foi exatamente o mesmo utilizado na aplicação do quantizador linear. Todos os blocos modelados serão utilizados nessa simulação. A única diferença entre os dois modelos está nos valores dos resistores utilizados. No projeto do conversor linear utilizou-se resistores iguais de $10K$ tanto para o ADC quanto para o DAC. Já para o caso do quantizador arco-seno, esses valores tiveram que ser calculados de acordo com as Equações 4.1 e 4.2 sugeridas por (MEDEIROS; HADDAD, 2017) em seu artigo. Com as equações citadas, a ajuda do software MATLAB e com o valor arbitrado de $R = 10K$ calculou-se então os valores das resistências para o ADC e para o DAC mostrados na Tabela 15.

Tabela 15 – Resistências calculadas.

RN	**RADC(Ω)**	**RDAC(Ω)**
R16	96	24
R15	285	191
R14	462	375
R13	621	545
R12	758	693
R11	865	815
R10	938	906
R9	975	961

RN	**RADC(Ω)**	**RDAC(Ω)**
R8	975	980
R7	938	961
R6	865	906
R5	758	815
R4	621	693
R3	462	545
R2	285	375
R1	96	191

Sendo R16 o resistor mais próximo da tensão de referência e R1 o resistor mais próximo ao *ground*.

6.1 Simulação

A simulação completa do quantizador arco-seno também foi feita da mesma forma que a do quantizador linear, conectando-se o ADC ao DAC, como pode ser visto na Figura 34, aplicando-se sinais de entrada e observando-se as saídas. Os parâmetros utilizados na simulação também foram os mesmos da Tabela 13 para o sinal senoidal e da Tabela 14 para o sinal do tipo rampa.

Nota-se que para ambas as simulações, o quantizador arco-seno tem passos de quantização menores, e por consequência , melhor resolução nas extremidades dos sinais.

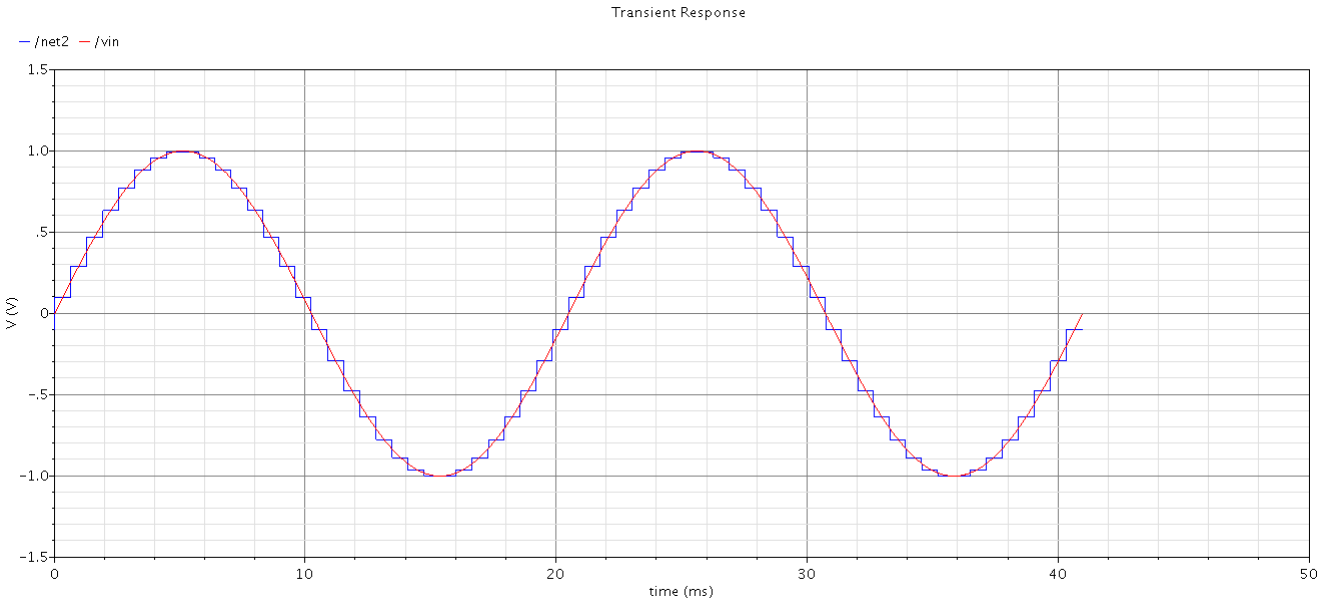


Figura 37 – Simulação do quantizador arco-seno para um sinal senoidal.

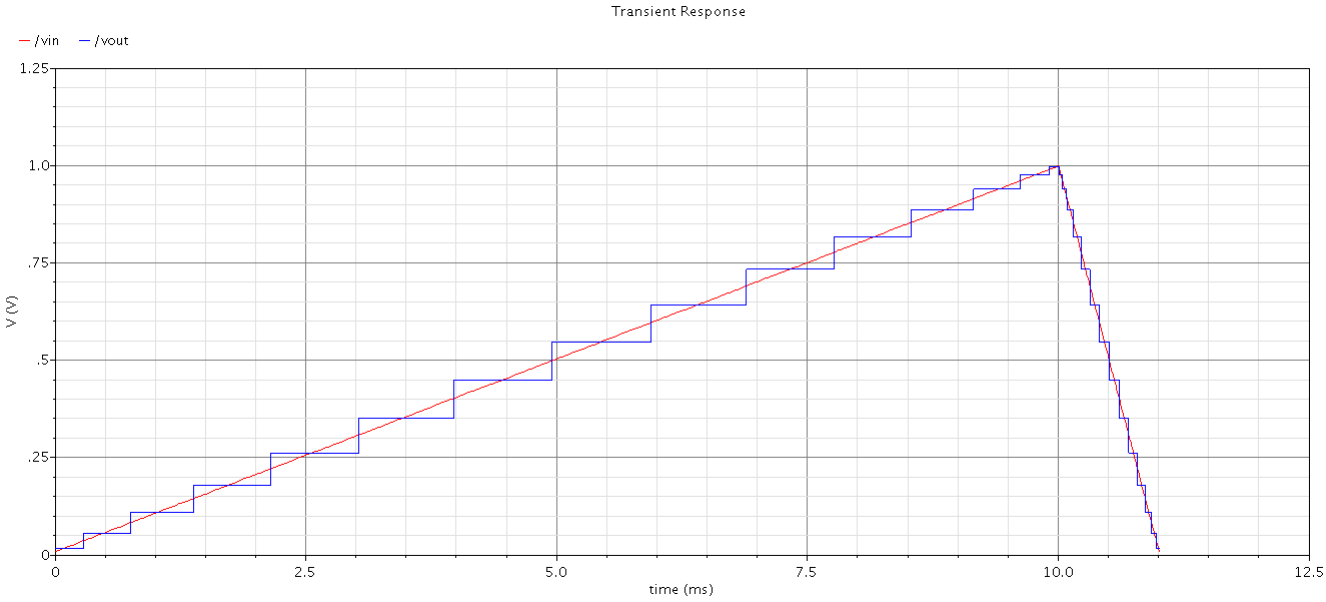


Figura 38 – Simulação do quantizador arco-seno para um sinal do tipo rampa.

7 Comparação entre Modelos

Nesse capítulo, será feita a comparação entre os resultados obtidos na modelagem do quantizador linear e do quantizador arco-seno.

A Figura 39 mostra os sinais de entrada e saída dos quantizadores projetados e seus respectivos erros de quantização. Como esperado para o quantizador linear, pode-se observar em 39-a que os passos de quantização são constantes, já para o quantizador arco-seno em 39-b os passos de quantização são menores nas extremidades do sinal de entrada, o que permite recuperar melhor a informação do sinal nesses pontos.

A diferença entre o sinal de saída e o sinal de entrada constitui o erro de quantização, que pode ser visto na mesma figura. Em 39-a obteve-se o padrão clássico de erro de quantização sino/dente de serra. Em 39-b obteve-se um erro de quantização menor nos extremos do sinal e comportamento modular.

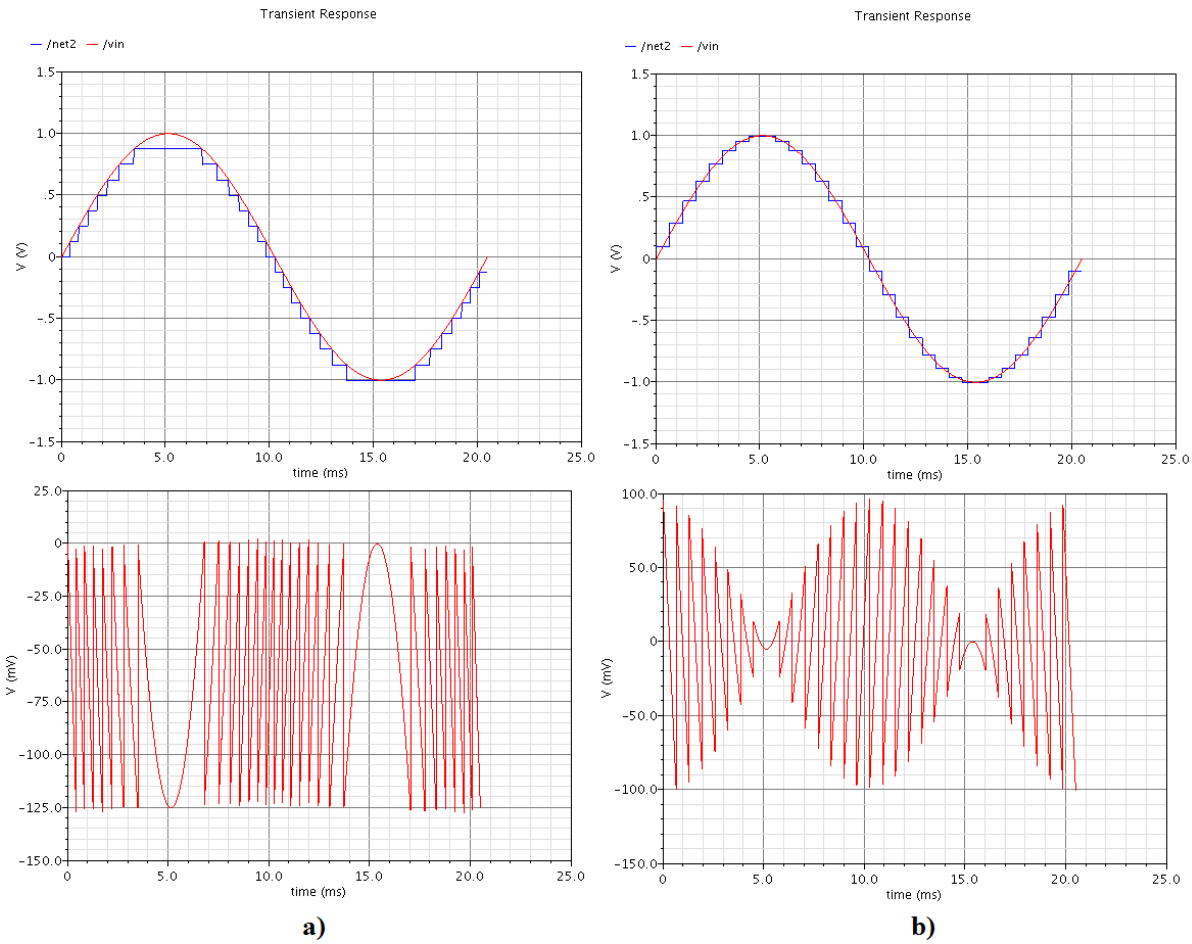


Figura 39 – Análise transiente para um sinal de entrada senoidal processado pelo quantizador linear e pelo quantizador arco-seno com seus respectivos erros de quantização.

Para melhor análise do comportamento dos quantizadores, tirou-se a DFT ($Discrete\ Fourier\ Transform$) dos sinais de saída dos mesmos. Os parâmetros utilizados na DFT dos sinais se encontram na Tabela 16.

A Figura 40 mostra o espectro em frequência da saída do quantizador linear e arco-seno, ambos com entradas senoidais de frequência $f0 = 48.828Hz$. Nas duas figuras há um impulso em $f0$, a frequência do sinal, além do ruído de quantização, harmônicos e espúrios que compõe o espectro. Nota-se na figura 40-b que no quantizador arco-seno os primeiros harmônicos importantes aparecem em uma frequência muito maior do que no quantizador linear sugerindo que pode-se obter um espectro de frequência mais limpo após o processo de amostragem no ADC. Além disso, os harmônicos importantes encontram-se centrados em $n * fs$ e espaçados por $2 * f0$ assim como nos resultados obtidos por (MEDEIROS; HADDAD, 2017) na análise do seu sistema.

Tabela 16 – Parâmetros de simulação da análise em frequência para os quantizadores linear e arco-seno.

Signal	Output
From	19.86m
To	183.70m
Number of Samples	19264
Window Type	Rectangular
Smoothing Factor	1
Coherent Gain	dB20
Coherent Gain Factor	1

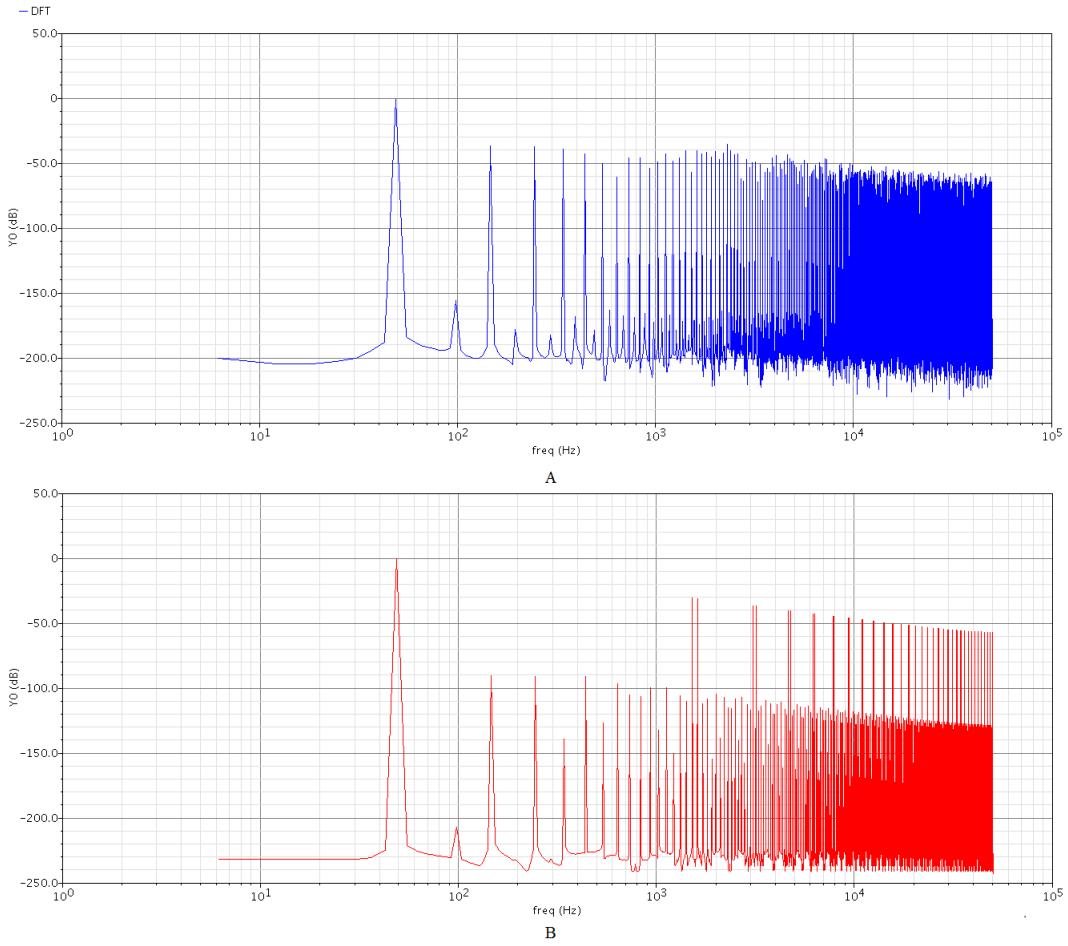


Figura 40 – Análise espectral para um sinal senoidal com frequência $f0 = 48.828Hz$ processado por um *ADC flash* de 4 bits: a) Com quantizador linear; b) Com quantizador arco-seno.

8 Modelagem do Modulador Σ-Δ

Esse capítulo apresenta a modelagem e simulações em alto nível do modulador Σ-Δ *Multi-bit* de primeira ordem. O objetivo dessas simulações é de aplicar os quantizadores projetados nos capítulos anteriores, tendo assim uma estimativa sobre o desempenho alcançável dos mesmos e também sobre os requisitos de *design* da arquitetura escolhida, antes de iniciar o projeto a nível de transistor. Realizou-se também a modelagem de um modulador Σ-Δ *single-bit* para comparar seu desempenho com relação ao *multi-bit*. Essas simulações foram realizadas com as ferramentas da Cadence Design Systems e os blocos foram modelados utilizando-se a *HDL* Verilog-A. Na sequência, serão apresentados os blocos pertencentes ao sistema a partir de sua descrição funcional, tabela de pinos e simulações.

8.1 Integrador

Um integrador é um circuito eletrônico que realiza o processo de integração em um determinado sinal de entrada. Na aplicação do modulador Σ-Δ em que consiste esse trabalho, o integrador integra o erro entre o sinal amostrado e sinal de entrada por meio da malha de realimentação como mostra a Figura 13.

8.1.1 Descrição do Bloco

A descrição em Verilog-A desse bloco teve como base a função *idt* que retorna a integral no tempo da expressão entre parênteses. Forneceu-se uma condição inicial que define o valor da função para análises DC e utilizou-se o parâmetro *scale* para acrescentar um ganho ao integrador, dispensando assim um novo bloco para essa função.

```
32 analog begin
33         @(cross((V(Clk)-vth), +1)) begin
34                 state = V(in);
35         V(out) <+ idt(scale*(state-vref), 0.5);
36 end
```

8.1.2 Descrição dos Pinos

Tabela 17 – Descrição dos pinos do integrador.

Pino	Descrição	Tipo
Clk	Clock	*input*
in	Tensão do sinal de entrada	*input*
out	Tensão do sinal de saída	*output*

8.1.3 Simulação

Para a simulação do integrador aplicou-se um sinal quadrado na entrada,com os parâmetros especificados na Tabela 18. O sinal de saída observado na Figura 42 é um sinal triangular como era esperado, validando assim o funcionamento do bloco.

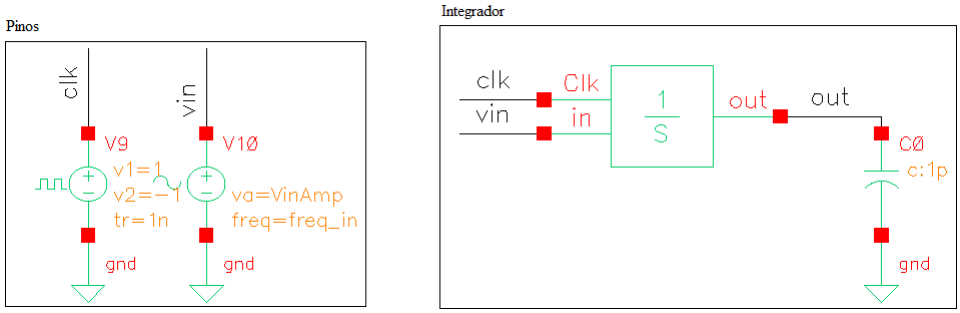


Figura 41 – *Testbench* do integrador.

Tabela 18 – Parâmetros de simulação do integrador.

Parâmetro	Descrição	Valor
Frequência	frequência da fonte de teste	1K
Amplitude	Amplitude do sinal de teste	1,-1
ciclos	Quantidade de ciclos na análise transiente	*3*

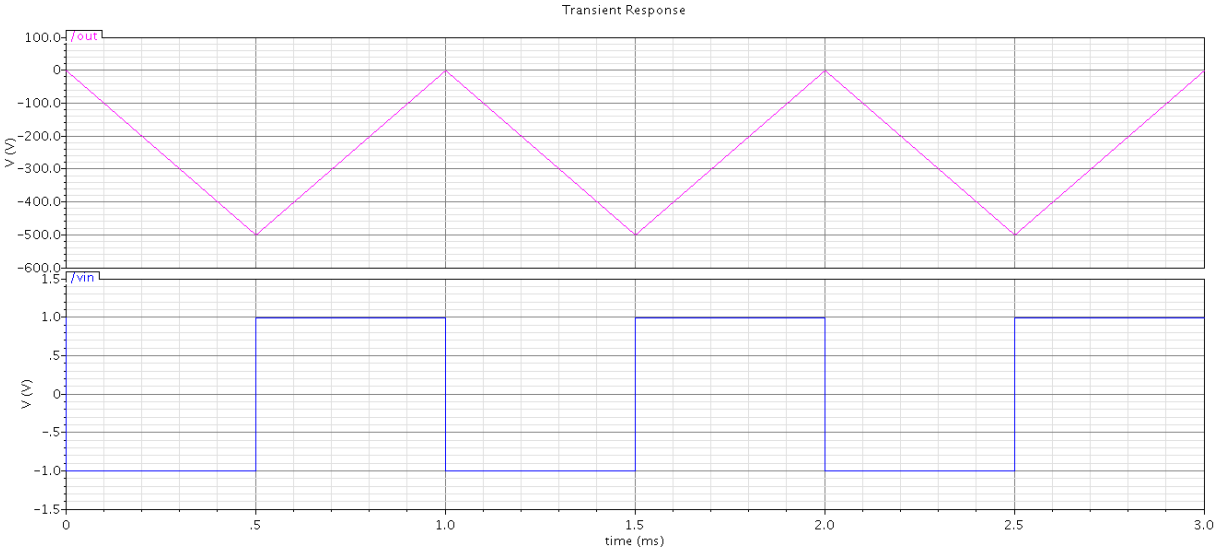


Figura 42 – Simulação do integrador.

8.2 Subtrator

O bloco subtrator, também chamado de ponto de soma em sistemas de controle, simplesmente subtrai dois sinais, nesse caso o sinal de entrada e a saída do modulador Σ-Δ.

8.2.1 Descrição do Bloco

Sua descrição em Verilog-A atribui ao pino de saída a tensão resultante da diferença entre dois sinais, $in1$ e $in2$.

```
1 analog begin
2         V(out) <+ (V(in1)-V(in2));
3 end
```

8.2.2 Descrição dos Pinos

Tabela 19 – Descrição dos pinos do subtrator.

Pino	Descrição	Tipo
in1	Tensão do primeiro sinal	*input*
in2	Tensão do segundo sinal	*input*
out	Tensão do sinal de saída	*output*

8.2.3 Simulação

Para a simulação do subtrator, montou-se o *testbench* da Figura 43 onde aplicou-se dois sinais as entradas do bloco, o primeiro deles um sinal senoidal ($in1$) e o segundo um sinal quadrado ($in2$). Os parâmetros de simulações encontram-se na Tabela 20. O resultado da simulação, mostrado na Figura 44 apresenta as formas de onda dos dois sinais de entrada e a subtração dos mesmos, validando assim o funcionamento do bloco.

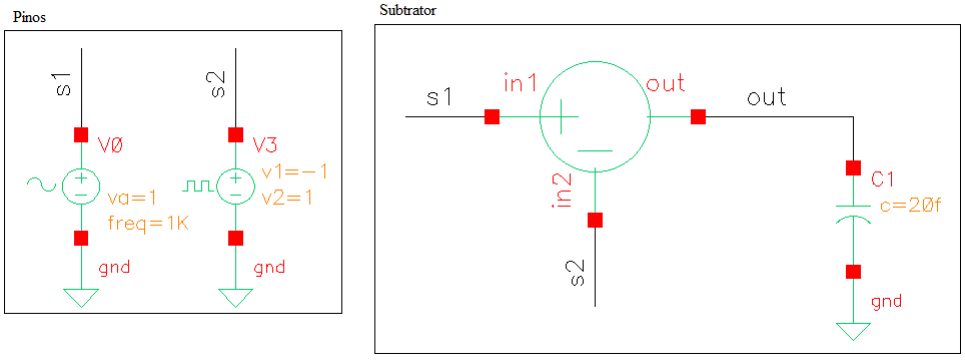


Figura 43 – *Testbench* do subtrator.

Tabela 20 – Parâmetros de simulação para o subtrator.

Parâmetro	Descrição	Valor
Frequência	Frequências das fontes de teste	$in1 = 1\text{K}$, $in2 = 5\text{K}$
Amplitude	Amplitudes das fontes de teste	$in1 = 1$, $in2 = 1$
Ciclos	Quantidade de ciclos na análise transiente	5

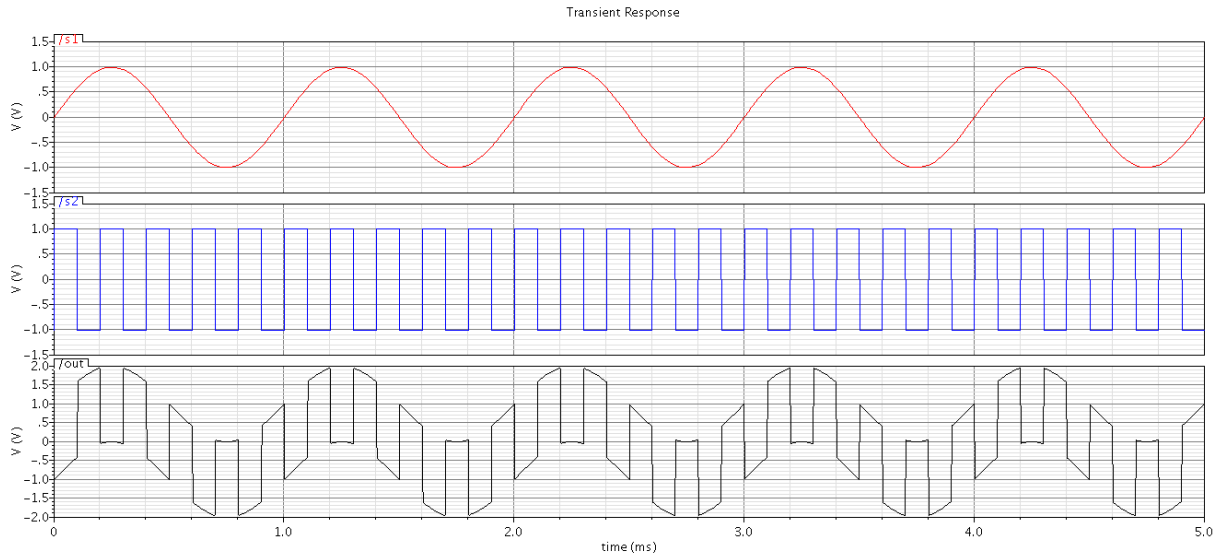


Figura 44 – Simulação do subtrator.

8.3 Amplificador

O amplificador modelado multiplica a amplitude do sinal de entrada por uma constante escolhida. Esse bloco foi projetado para equilibrar o ganho na malha.

8.3.1 Descrição do Bloco

A descrição em Verilog-A do bloco atribui ao pino de saída (out) a tensão resultante do sinal de entrada (in) multiplicado por uma constante de ganho ($gain$).

```
37 analog begin
38         V(out) <+ gain*V(in);
39 end
```

8.3.2 Descrição dos Pinos

Tabela 21 – Descrição dos pinos do amplificador.

Pino	Descrição	Tipo
in	Tensão do sinal de entrada	*input*
out	Tensão do sinal de saída	*output*

8.3.3 Simulação

Para a simulação do amplificador, montou-se o *testbench* da Figura 45 onde aplicou-se um sinal senoidal de entrada , cujos parâmetros encontram-se na Tabela 22, e escolheu-se um valor para o coeficiente de ganho. Realizou-se então uma análise transiente. O resultado da simulação, mostrado na Figura 46 apresenta as formas de onda de entrada e saída do amplificador, mostrando o sinal amplificado pelo fator *gain*.

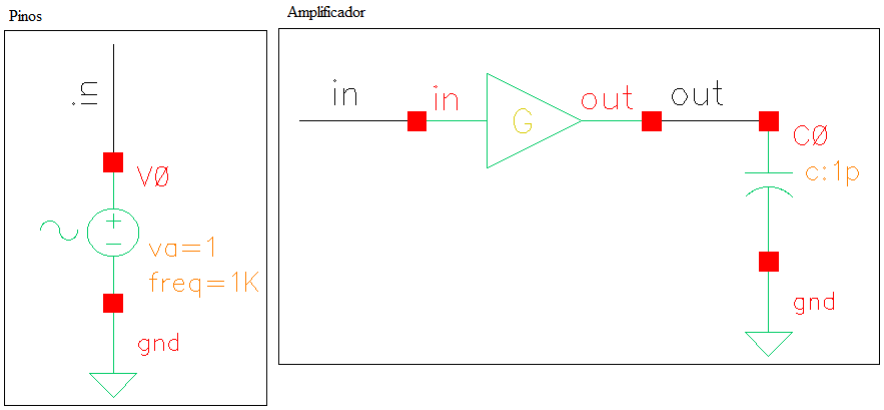


Figura 45 – *Testbench* do amplificador.

Tabela 22 – Parâmetros de simulação para o amplificador.

Parâmetro	Descrição	Valor
Frequência	Frequência da fonte de teste	1 K
Amplitude	Amplitude da fonte de teste	1
gain	Coeficiente de ganho do amplificador	5

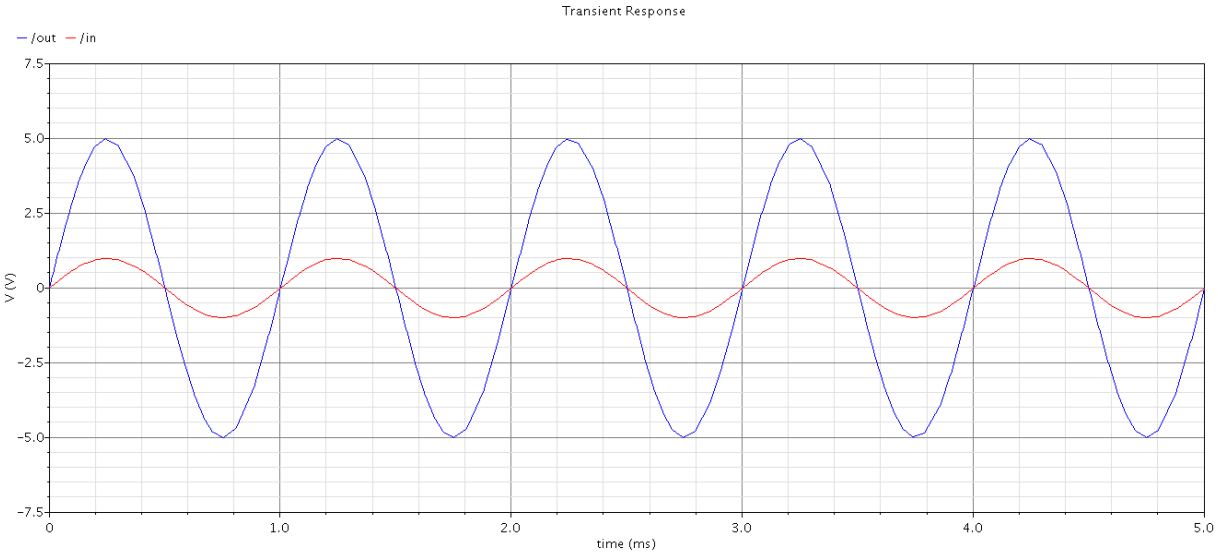


Figura 46 – Simulação do amplificador.

8.4 Quantizador

O quantizador utilizado trata-se do quantizador de 4 bits projetado por meio de uma topologia *flash* que foi modelado no Capítulo 5. Sua arquitetura, já vista anteriormente, conta com um *ADC* de 4 bits seguido seguido por um *DAC* de 4 bits como pode ser visto na Figura 47.

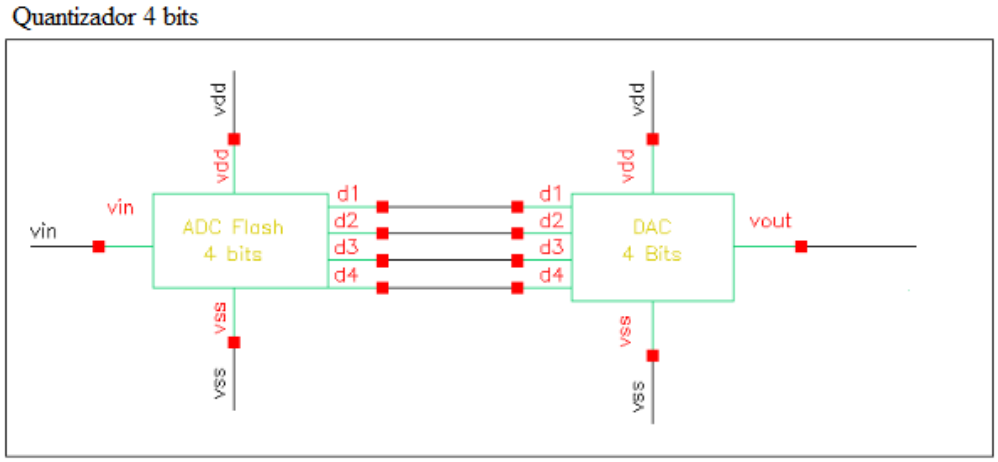


Figura 47 – Quantizador de 4 bits.

Afim de obter um esquemático mais compacto , criou-se um símbolo para o quantizador que pode ser visto na Figura 48. Esse quantizador já foi simulado e validado no Capítulo 5, onde pode ser encontrado seu *testbench*, tabela de pinos e simulações.

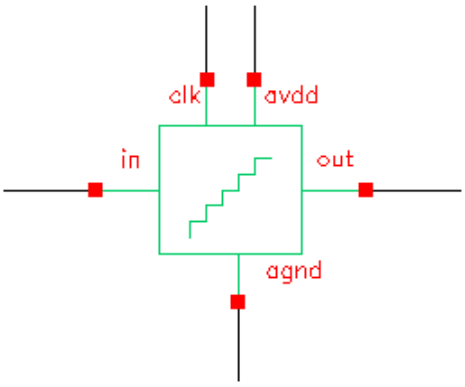


Figura 48 – Símbolo do quantizador de 4 bits.

8.5 Modulador Σ-Δ *Single-bit*

Como dito anteriormente, os moduladores Σ-Δ podem ser *single-bit* ou *multi-bit* dependendo da quantidade de bits do seu quantizador. Nessa seção realizou-se a modelagem e caracterização de um modulador *single-bit* de 1ª ordem projetado para as especificações da Tabela 23. A Figura 49 mostra o *testbench* utilizado para o modulador *single-bit*. O código usado na modelagem encontra-se no Apendice A.4.

Tabela 23 – Especificações do modulador Σ-Δ *single-bit* de 1ª ordem.

Especificações do Modulador Σ-Δ *single-bit*	
Frequência de amostragem: fs	2.048 MHz
Tensão de referência positiva ($avdd$)	1 V
Tensão de referência negativa ($avss$)	-1 V
Frequência do sinal de entrada (fin)	1 KHz
Oversampling (OS)	1024

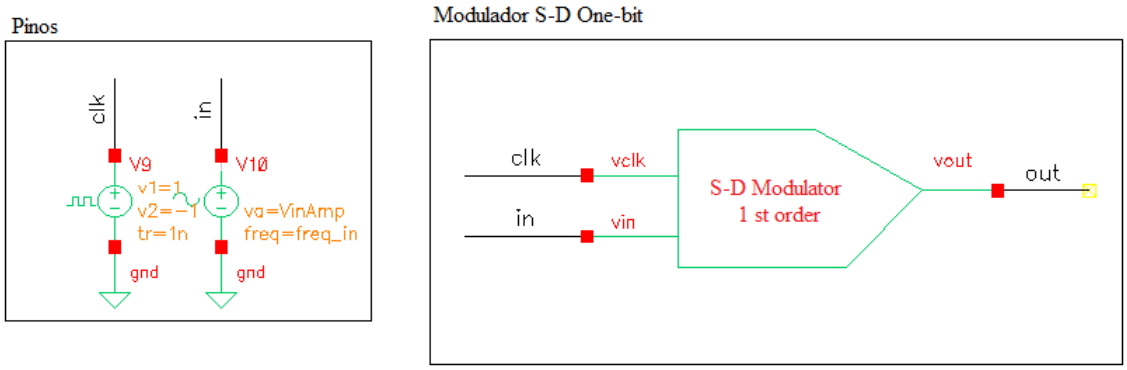


Figura 49 – *Testbench* do modulador Σ-Δ *single-bit*.

8.5.1 Descrição dos Pinos

Tabela 24 – Descrição dos pinos do modulador Σ-Δ *single-bit* de 1ª ordem.

Pino	Descrição	Tipo
clk	Clock	*input*
in	Tensão do sinal de entrada	*input*
out	Tensão do sinal de saída	*output*

8.5.2 Simulação

Para a simulação do modulador *single-bit* aplicou-se um sinal senoidal na entrada com os parâmetros especificados na Tabela 25 e realizou-se uma análise transiente.

A Figura 50 mostra o resultado da simulação para três nós destacados. O primeiro deles, *in*, representa o sinal de entrada, o segundo , *out*, o sinal de saída, e o terceiro, erro, representa o erro de quantização do modulador. Como pode ser observado na figura , o erro de quantização é grande uma vez que a saída do modulador possui apenas dois bits.

Tabela 25 – Parâmetros de simulação para o modulador Σ-Δ *single-bit*.

Parâmetro	Descrição	Valor
Frequência	Frequência da fonte de teste	1 KHz
Amplitude	Amplitude da fonte de teste	850 m
Ciclos	Quantidade de ciclos da simulação	4

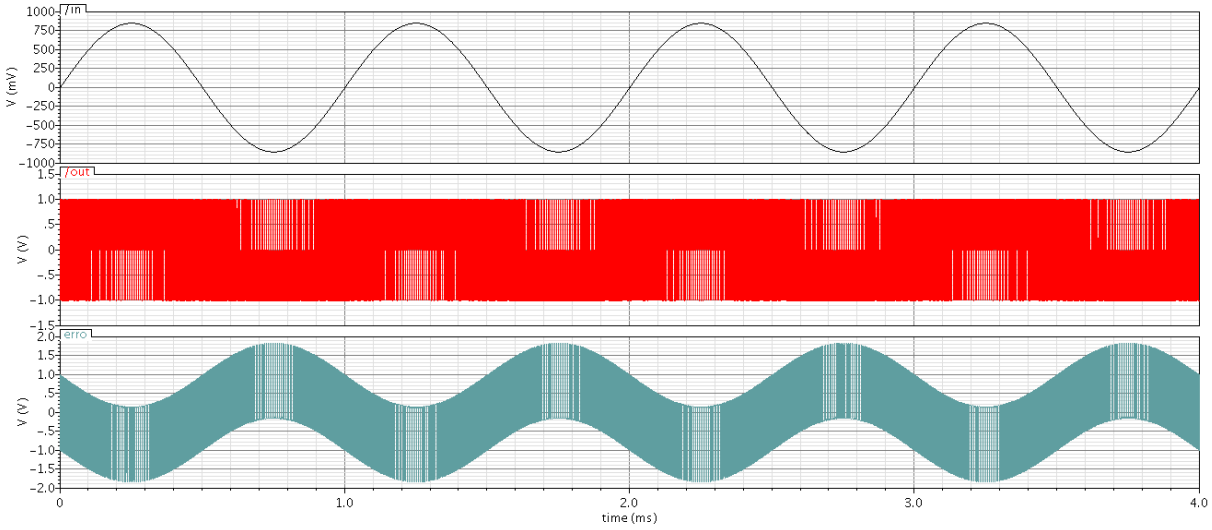


Figura 50 – Simulação do modulador Σ-Δ *single-bit* de 1ª ordem.

A Figura 51 mostra os sinais de entrada e saída do modulador *single-bit* sobrepostos com apenas um ciclo de simulação.

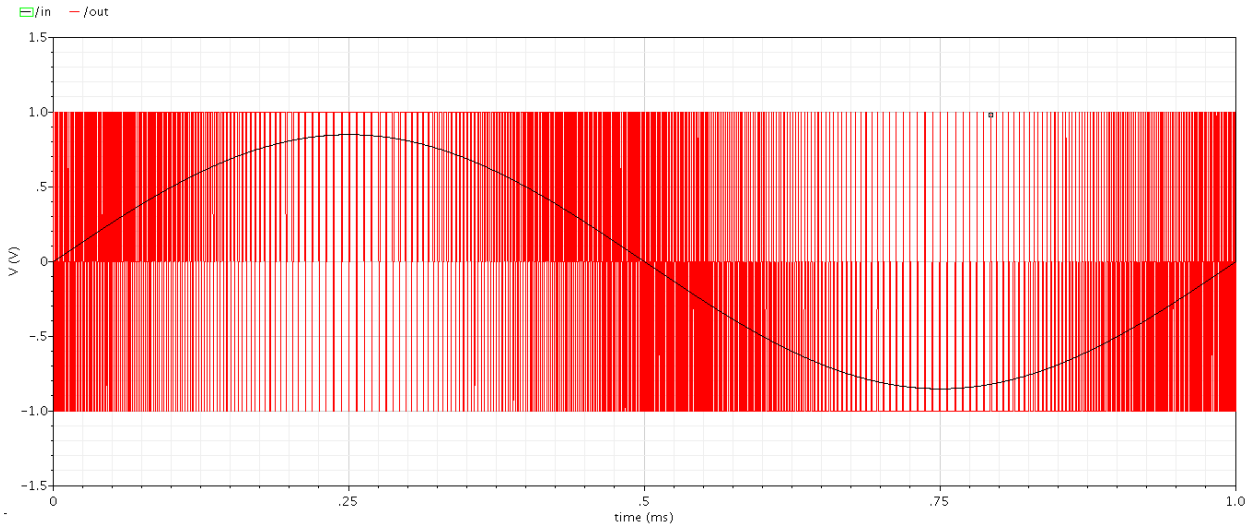


Figura 51 – Simulação do modulador Σ-Δ *single-bit* de 1ª ordem.

Para fazer uma análise em frequência do modulador, calculou-se a *DFT* do sinal de saída, como mostra a Figura 52. No espectro observa-se a frequência do sinal e o *noise shaping* que joga o ruído para altas frequências. Extraiu-se por fim alguns parâmetros dinâmicos do modulador que podem ser visualizados na Tabela 26

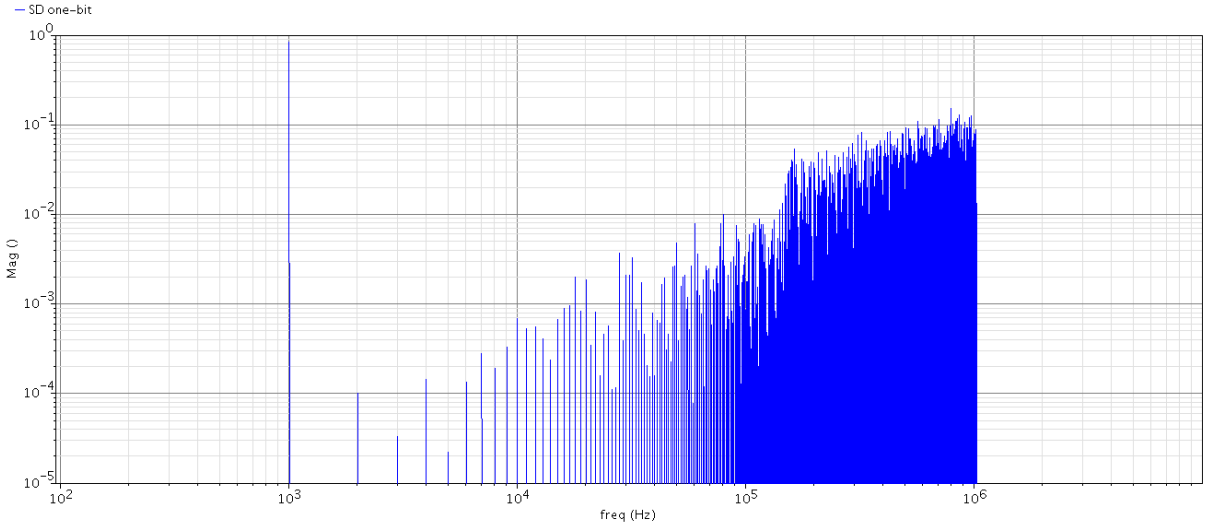


Figura 52 – DFT do modulador Σ-Δ *single-bit* de 1ª ordem.

Tabela 26 – Parâmetros dinâmicos do modulador Σ-Δ *single-bit*

Parâmetros Dinâmicos	**Valor**
SNHR	198.6 dB
SINAD	59.72 dB
SFDR	61.72 dB
ENOB	9.628 bits
THD	133.0 %

8.6 Modulador Σ-Δ *Multi-bit* com Quantizador Linear

Após a modelagem dos blocos, projetou-se então o modulador Σ-Δ *Multi-bit* de 1ª Ordem com o quantizador linear de 4 bits modelado anteriormente. O esquemático utilizado para simulação pode ser visto na Figura 53. A Tabela 27 apresenta as especificações do modulador projetado.

Tabela 27 – Especificações do modulador Σ-Δ de 1ª ordem.

Especificações do Modulador Σ-Δ de 1ª Ordem	
Frequência de amostragem: fs	2.048 MHz
Tensão de referência positiva ($avdd$)	1 V
Tensão de referência negativa ($avss$)	-1 V
Coeficientes de malha aberta	$A_{integrador} = 1000$
Coeficientes de realimentação	$A_{loop} = 1$
Frequência do sinal de entrada (fin)	1 KHz
Oversampling (OS)	1024

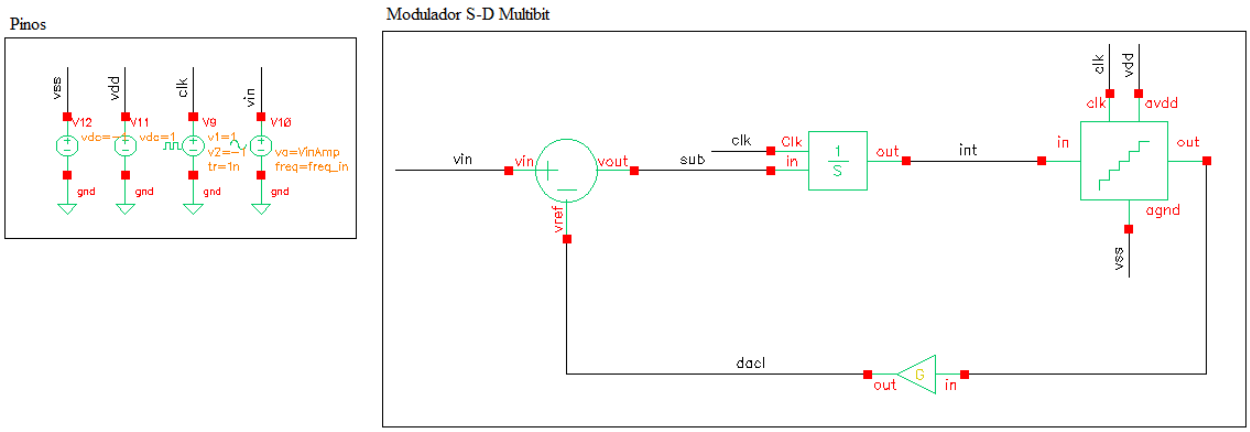


Figura 53 – *Testbench* do modulador Σ-Δ *multi-bit* de 1ª ordem.

8.6.1 Descrição dos Pinos

Tabela 28 – Descrição dos pinos do modulador Σ-Δ *multi-bit* de 1ª ordem.

Pino	**Descrição**	**Tipo**
vdd	Tensão de referência positiva (1V)	*inout*
vss	Tensão de referência negativa (-1V)	*inout*
clk	Clock	*input*
vin	Tensão do sinal de entrada	*input*

8.6.2 Simulação

Para a simulação do modulador aplicou-se um sinal senoidal na entrada do sistema, com parâmetros especificados na Tabela 29, e observou-se como os diferentes nós se comportam.

A Figura 54 mostra o resultado da simulação para os quatro nós destacados no esquemático da Figura 53. O primeiro deles (vin), representado pelo sinal em vermelho, mostra o sinal de entrada. O segundo (dacl), representado pelo sinal em azul, mostra a saída do modulador. O terceiro nó (sub), representado pelo sinal em lilás, mostra a saída do subtrator que é o erro, ou seja, a diferença entre o sinal de entrada e a saída do modulador. Por fim o quarto nó (int), representado em verde, mostra a integral do erro.

Tabela 29 – Parâmetros de simulação para o modulador Σ-Δ linear.

Parâmetro	**Descrição**	**Valor**
Frequência	Frequência da fonte de teste	1 KHz
Amplitude	Amplitude da fonte de teste	850 m
Ciclos	Quantidade de ciclos da simulação	4

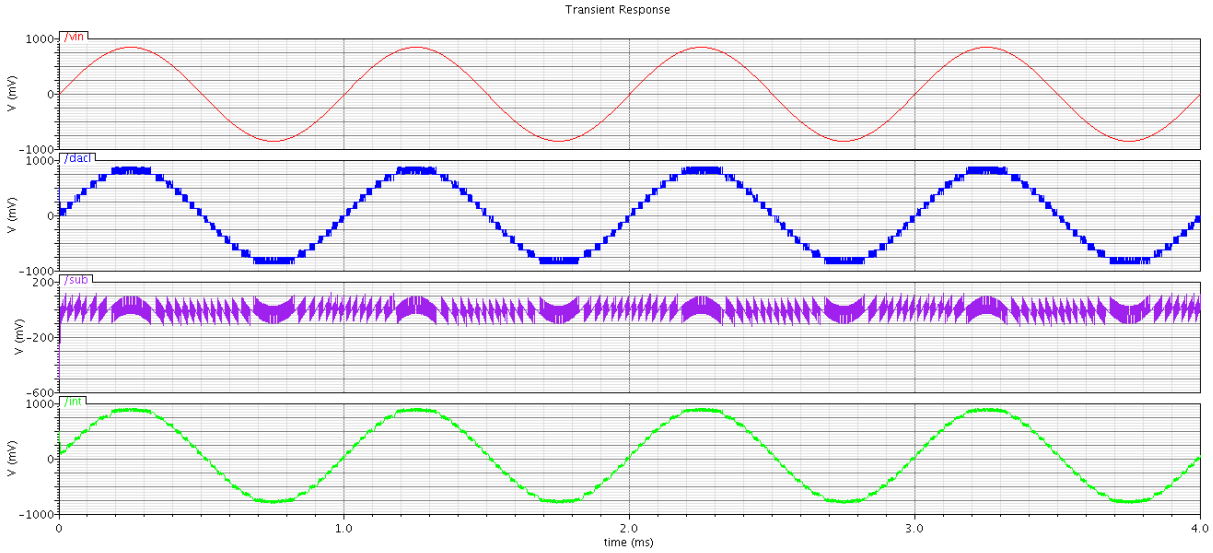


Figura 54 – Simulação do modulador Σ-Δ *multi-bit* de 1ª ordem com quantizador linear.

A Figura 55 mostra agora apenas os sinais de entrada e saída do modulador sobrepostos com 1 ciclo de simulação. Observa-se que a saída apresenta o comportamento esperado de um modulador Σ-Δ *multi-bit* descrito na seção 3.3.3.1 e que os passos de quantização são constantes devido ao uso do quantizador linear de 4 bits.

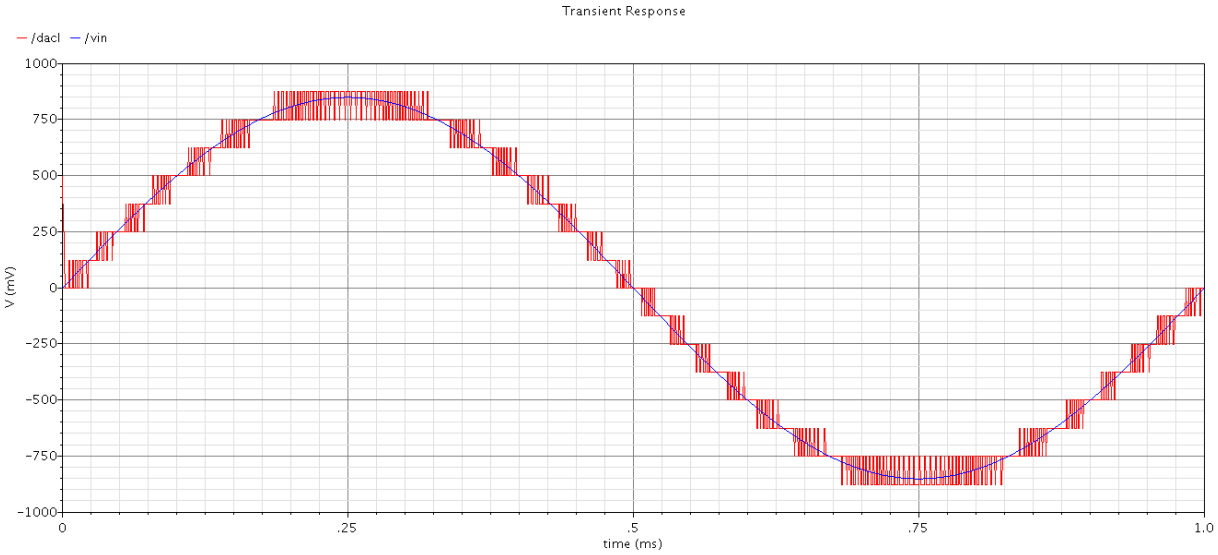


Figura 55 – Simulação do modulador Σ-Δ *multi-bit* de 1ª ordem com quantizador linear.

Realizou-se também uma análise em frequência do modulador afim de observar melhor seu comportamento. A Figura 57 mostra a DFT do sinal de saída do modulador. No espectro pode-se observar o impulso em $f0$ que é a frequência do sinal de entrada e o *noise shaping* característico do ADC Σ-Δ. Os parâmetros utilizados na DFT para realizar uma amostragem coerente encontram-se na Figura 56.

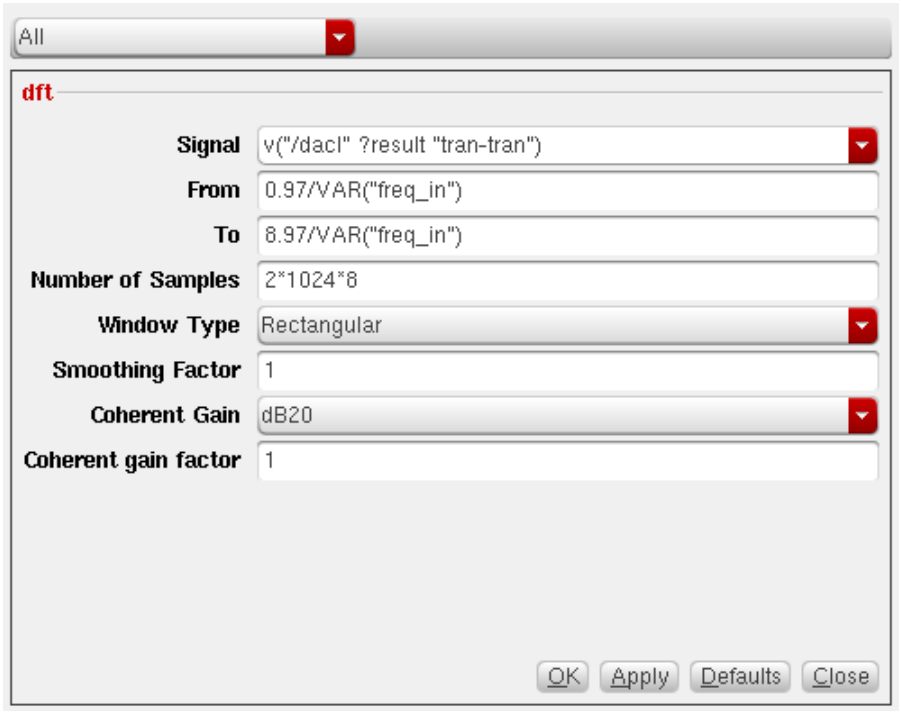


Figura 56 – Parâmetros utilizados na DFT do modulador.

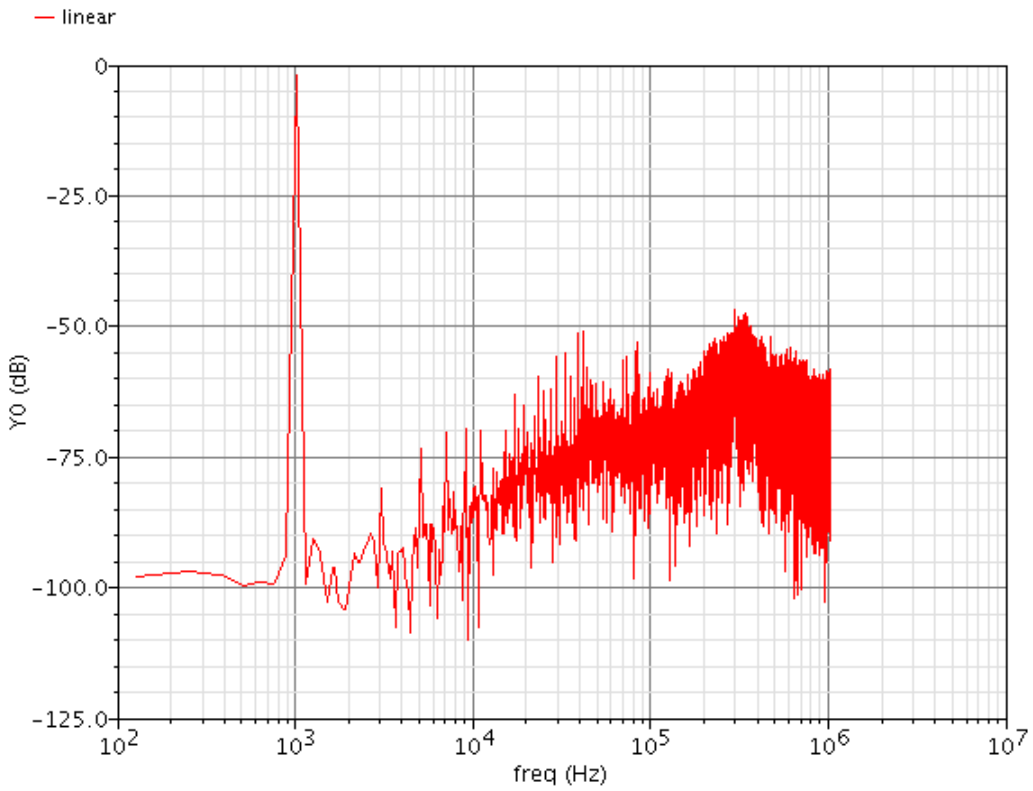


Figura 57 – DFT do modulador Σ-Δ *multi-bit* de 1ª ordem com quantizador linear.

A Tabela 30 mostra os parâmetros dinâmicos do modulador *multi-bit* e *single-bit* extraídos do sinais de saída dos moduladores com o auxílio da calculadora do Cadence para uma frequência de corte de 5K.

Observa-se que o desempenho geral do modulador *multi-bit* é melhor do que o do outro. Como dito anteriormente, o quantizador *single-bit* apresenta um grande erro de quantização pois sua saída é composta de apenas dois bits, isso fica evidente ao analisar sua distorção harmônica muito alta e a $SFDR$ menor.

Tabela 30 – Parâmetros dinâmicos do modulador Σ-Δ *single-bit* e *multi-bit* de 1ª ordem com quantizador linear.

Parâmetros Dinâmicos	Σ-Δ *multi-bit*	Σ-Δ *single-bit*
SNHR	77.33 dB	198.6 dB
SINAD	70.09 dB	59.72 dB
SFDR	71.74 dB	61.72 dB
ENOB	11.35 bits	9.628 bits
THD	8.351 %	133.0 %

8.7 Modulador Σ-Δ *Multi-bit* com Quantizador Arco-seno

Na implementação do modulador Σ-Δ *multi-bit* de 1ª ordem com quantizador arco-seno utilizou-se os mesmos blocos usados no modulador linear, com exceção do quantizador. O esquemático utilizado no *testbench* também foi o da Figura 53 mas dessa vez o quantizador utilizado foi o quantizador arco-seno de 4 bits com topologia $flash$ modelado do Capítulo 6. As especificações do modulador projetado também são as mesmas mostradas na tabela 27.

8.7.1 Simulação

Assim como no modulador linear, a simulação do modulador arco-seno foi feita aplicando-se um sinal senoidal na entrada do sistema, com parâmetros especificados na Tabela 31, e observando-se como os diferentes nós se comportam.

A Figura 58 mostra o resultado da simulação para quatro nós destacados. O sinal (vin), em vermelho, mostra o sinal de entrada. O sinal (dac), em azul, mostra a saída do modulador. O sinal (sub), em lilás, mostra o erro, que agora apresenta um comportamento modular, como esperado do quantizador arco-seno e o sinal (int), em verde, mostra a integral do erro.

Tabela 31 – Parâmetros de simulação para o modulador Σ-Δ arco-seno.

Parâmetro	**Descrição**	**Valor**
Frequência	Frequência da fonte de teste	1 KHz
Amplitude	Amplitude da fonte de teste	980 m
Ciclos	Quantidade de ciclos da simulação	4

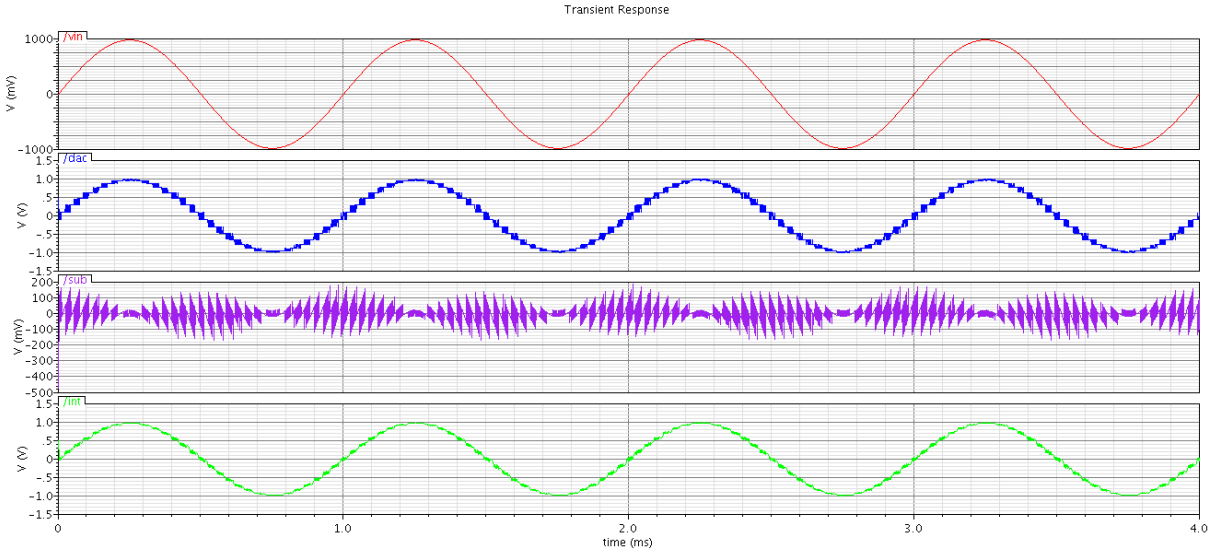


Figura 58 – Simulação do modulador Σ-Δ *multi-bit* de 1ª ordem com quantizador arco-seno.

A Figura 59 mostra agora os sinais de entrada (azul) e saída (vermelho) do modulador sobrepostos com 1 ciclo de simulação. Observa-se novamente o comportamento típico de um modulador Σ-Δ *multi-bit* mas agora com passos de quantização variáveis, que são menores nas extremidades do sinal.

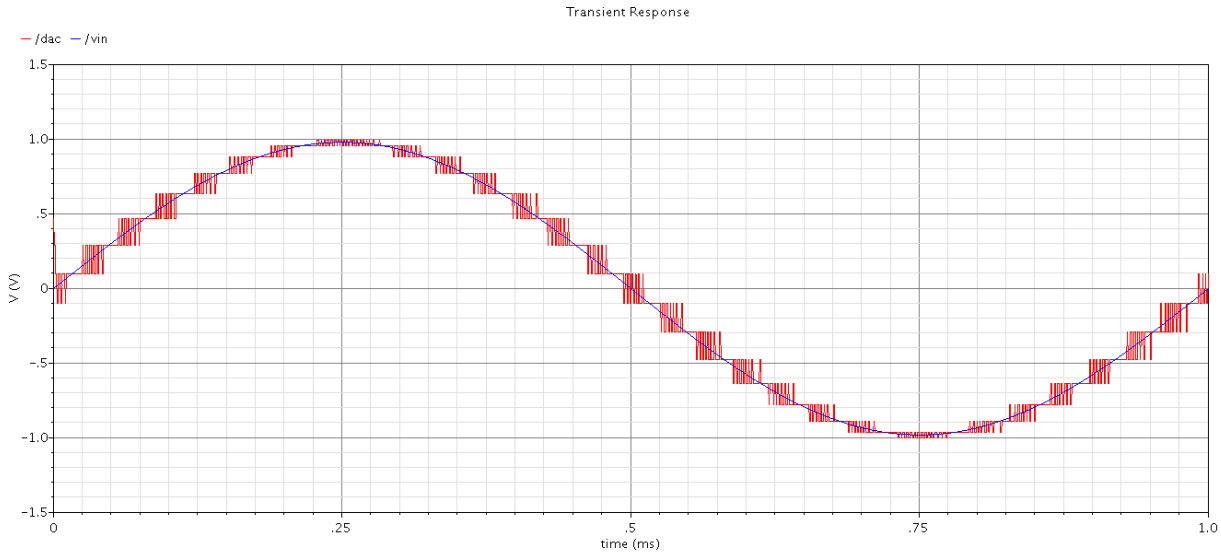


Figura 59 – Simulação do modulador Σ-Δ *multi-bit* de 1ª ordem com quantizador arco-seno.

Realizou-se a análise em frequência do sinal de saída do modulador através de sua DFT, que teve os parâmetros setados iguais aos do modulador linear afim de manter a amostragem coerente, e obteve-se então o espectro da Figura 60. No espectro pode-se observar a frequência fundamental e *noise shaping* característico e a partir dele extraiu-se alguns parâmetros dinâmicos mostrados na Tabela 32 para uma frequência de corte de 5 KHz.

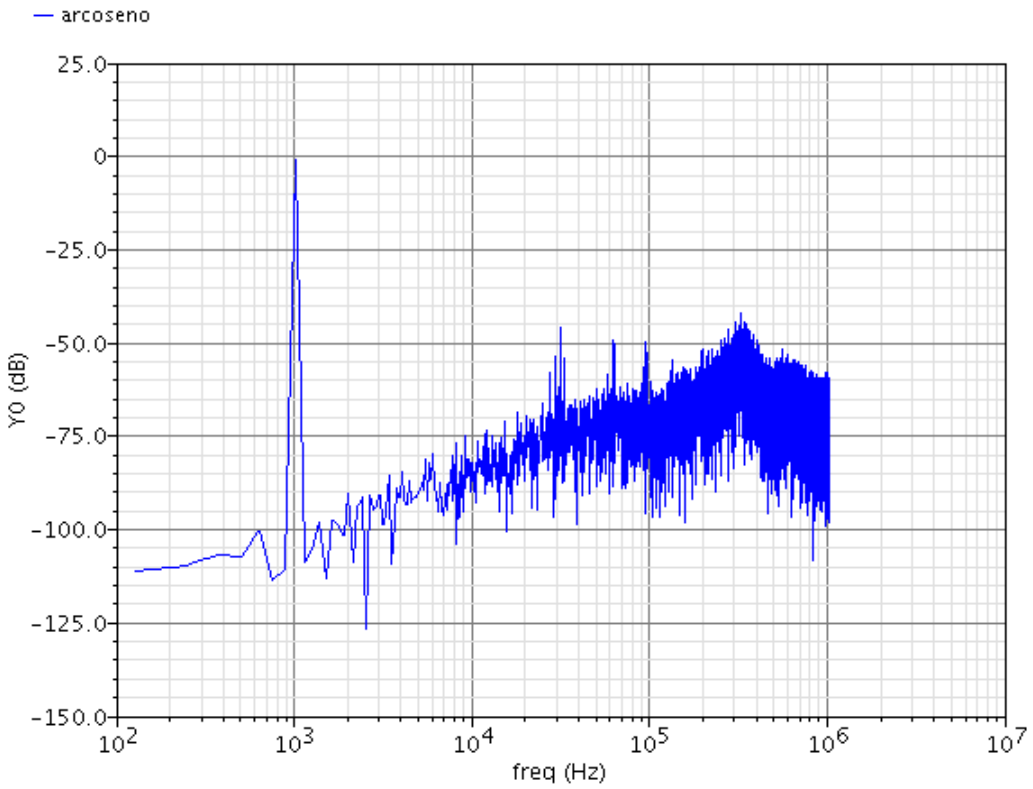


Figura 60 – DFT do modulador Σ-Δ *multi-bit* de 1ª ordem com quantizador arco-seno.

Tabela 32 – Parâmetros dinâmicos do modulador Σ-Δ multi-bit de 1ª ordem com quantizador arco-seno.

Parâmetros Dinâmicos	**Valor**
SNHR	77.82 dB
SINAD	76.18 dB
SFDR	83.63 dB
ENOB	12.36 bits
THD	7.947 %

8.8 Comparação entre Moduladores *Multi-bit*

Nessa seção será feita a comparação entre o desempenho obtido pelos dois moduladores Σ-Δ *multi-bit* de 1ª ordem que foram modelados, o modulador com quantizador linear e o modulador com quantizador arco-seno. A comparação foi feita extraindo-se as respostas no domínio do tempo e da frequência para os dois moduladores e analisando-se alguns parâmetros dinâmicos dos mesmos. Os parâmetros utilizados nas simulações foram os mesmos descritos nas seções anteriores.

A Figura 61 mostra os sinais de entrada (preto) e saída (vermelho) dos moduladores projetados e seus respectivos erros (azul). Em 61-a além dos passos de quantização

constantes e do erro com formato serra/sino nota-se que o modulador com quantizador linear permite um sinal de entrada com amplitude menor do que o modulador com quantizador arco-seno. Já em 61-b, na resposta do modulador arco-seno, observa-se os passos de quantização diferentes, o erro com comportamento modular e a excursão maior do sinal de entrada que chega quase a VDD sem saturar, uma vez que o erro é menor nas extremidades do sinal.

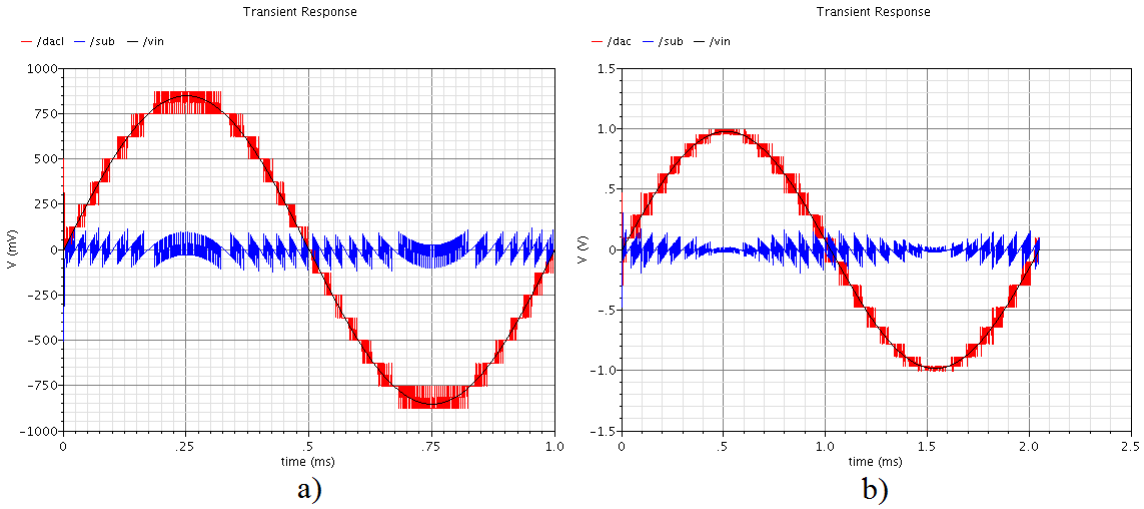


Figura 61 – Análise transiente do modulador linear a) e arco-seno b).

A Figura 62 mostra a DFT dos dois moduladores sobrepostas. Nota-se que o o *noise floor* do modulador arco-seno é menor que o do modulador linear , principalmente perto da banda do sinal. Observa-se também que os harmônicos ímpares apresentam uma potência maior no modulador linear do que no modulador arco-seno, isso pode ser melhor visualizado na Figura 63 que é um *zoom* da imagem dos espectros na primeira década.

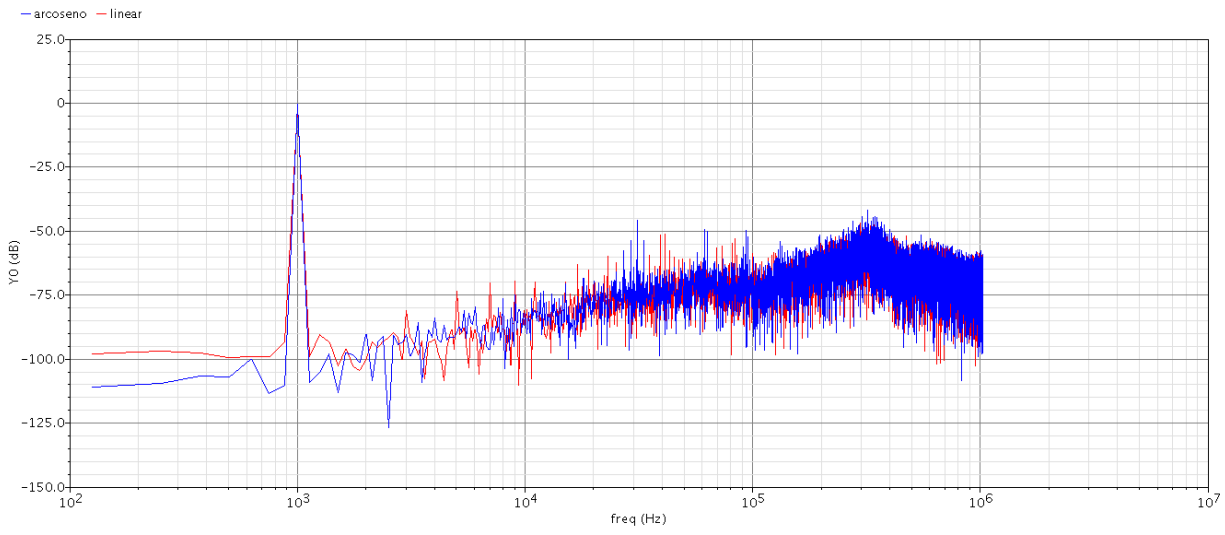


Figura 62 – DFT do modulador linear (vermelho) e arco-seno (azul).

Essa maior potência dos harmônicos ímpares tem um grande impacto na distorção harmônica total do sinal, o que fica evidente nas próximas análises.

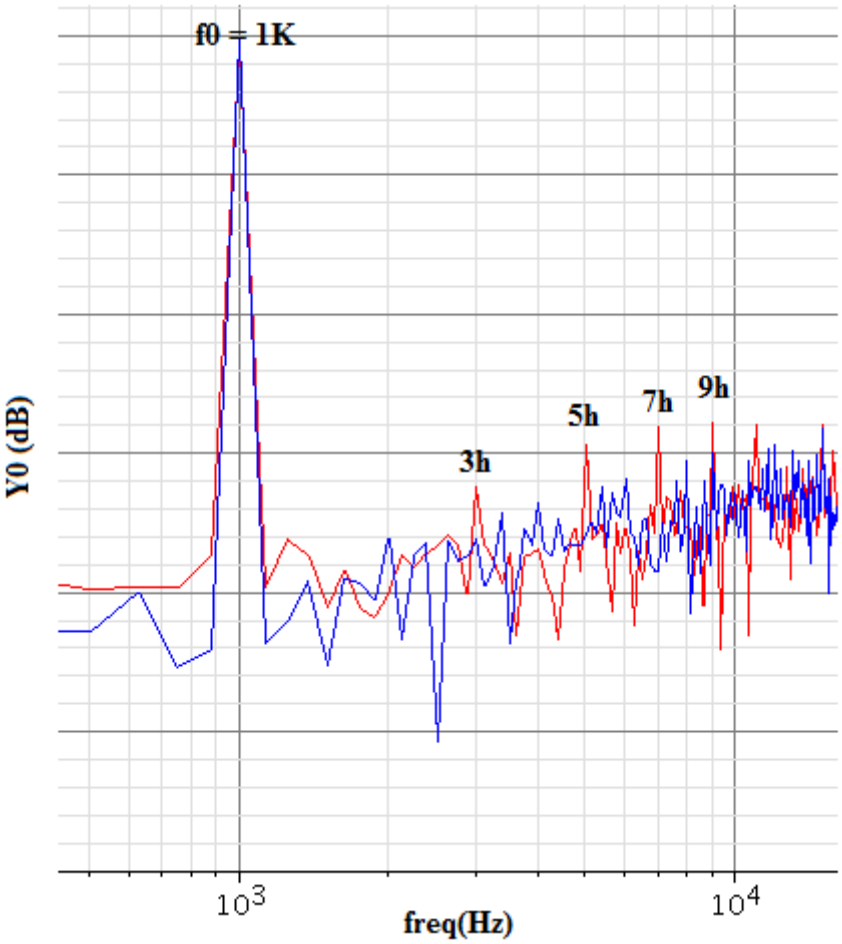


Figura 63 – DFT do modulador linear (vermelho) e arco-seno(azul).

A partir dos resultados obtidos por meio das simulações e com o auxílio da calculadora do Cadence, extraiu-se alguns parâmetros dinâmicos dos sinais de saída dos moduladores. Esses parâmetros são THD, $SNHR$, $SINAD$, $ENOB$ e $SFDR$, cujas características foram apresentadas na seção 3.2. Os parâmetros foram ordenados na Tabela 33 como forma de comparar o desempenho dos dois moduladores e para uma análise mais precisa, variou-se o *oversampling* de 256 a 4096.

Tabela 33 – Parâmetros dinâmicos dos moduladores Σ-Δ linear e arco-seno.

Oversampling : 256

Parâmetros	**Modulador Σ-Δ linear**	**Modulador Σ-Δ arco-seno**
SNHR	54.64 dB	61.61 dB
SINAD	49.49 dB	49.9 dB
SFDR	54.16 dB	52.77 dB
ENOB	7.929 bits	7.997 bits
THD	70.78 %	62.0 %

Oversampling : 512

Parâmetros	**Modulador Σ-Δ linear**	**Modulador Σ-Δ arco-seno**
SNHR	73.75 dB	78.54 dB
SINAD	73.44 dB	75.01 dB
SFDR	82.84 dB	78.82 dB
ENOB	11.91 bits	12.17 bits
THD	9.063 %	8.586 %

Oversampling : 1024

Parâmetros	**Modulador Σ-Δ linear**	**Modulador Σ-Δ arco-seno**
SNHR	77.33 dB	77.82 dB
SINAD	70.09 dB	76.18 dB
SFDR	71.74 dB	83.63 dB
ENOB	11.35 bits	12.36 bits
THD	8.351 %	7.947 %

Oversampling : 2048

Parâmetros	**Modulador Σ-Δ linear**	**Modulador Σ-Δ arco-seno**
SNHR	83.64 dB	83.07 dB
SINAD	67.58 dB	76.36 dB
SFDR	68.75 dB	78.36 dB
ENOB	10.93 bits	12.39 bits
THD	8.193 %	7.743 %

Oversampling : 4096

Parâmetros	**Modulador Σ-Δ linear**	**Modulador Σ-Δ arco-seno**
SNHR	85.89 dB	87.31 dB
SINAD	66.47 dB	81.03 dB
SFDR	67.9 dB	83.72 dB
ENOB	10.75 bits	13.17 bits
THD	8.113 %	7.658 %

Para melhor visualizar os resultados da Tabela 33, plotou-se os pontos em forma de gráficos para cada parâmetro analisado. A análise para o *oversampling* de 256 não foi levada em consideração por apresentar um desempenho ruim nos dois moduladores, a distorção harmônica para esse caso atingiu valores superiores a 70%, sugerindo que o sistema não funciona para baixos *oversamplings*.

A Figura 64 mostra a distorção harmônica total medida para os dois moduladores. Pode-se notar que a THD é menor no quantizador arco-seno e que com o aumento do

oversampling essa distorção diminui mais ainda. Esse resultado concorda com o resultado obtido na análise das $DFTs$ (Figura 63) onde o quantizador linear apresentou harmônicos ímpares com potências maiores que o arco-seno, já então sugerindo um aumento nessa distorção.

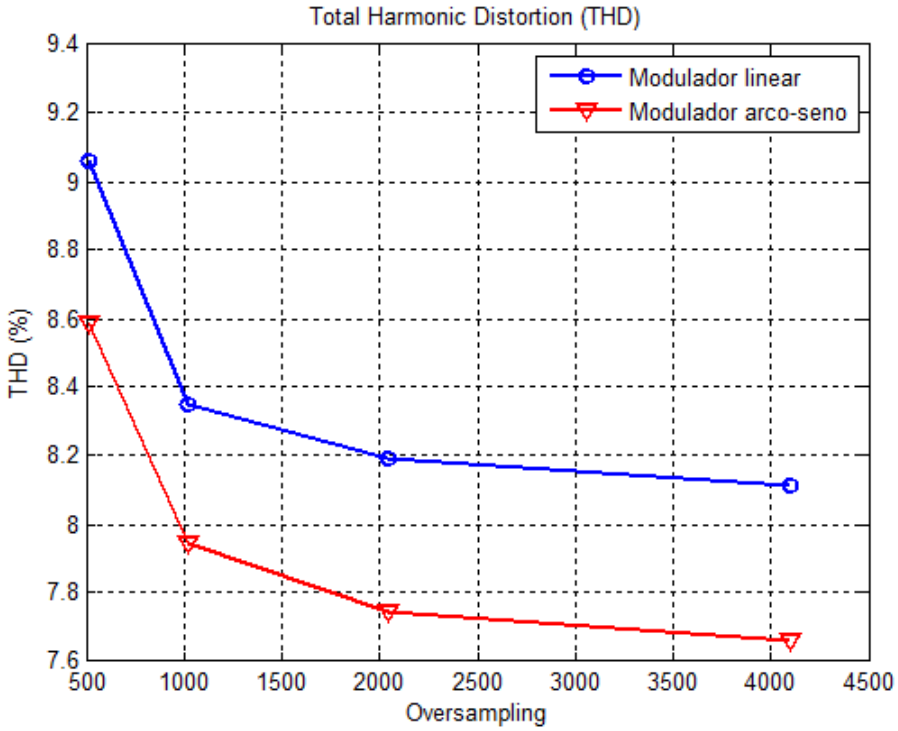


Figura 64 – Gráficos de THD para os moduladores linear e arco-seno.

A Figura 65 mostra a $SNHR$ dos moduladores. Como mencionado anteriormente, a SNR ou $SNHR$ aumenta com o aumento do *oversampling*, o que concorda com o gráfico obtido.

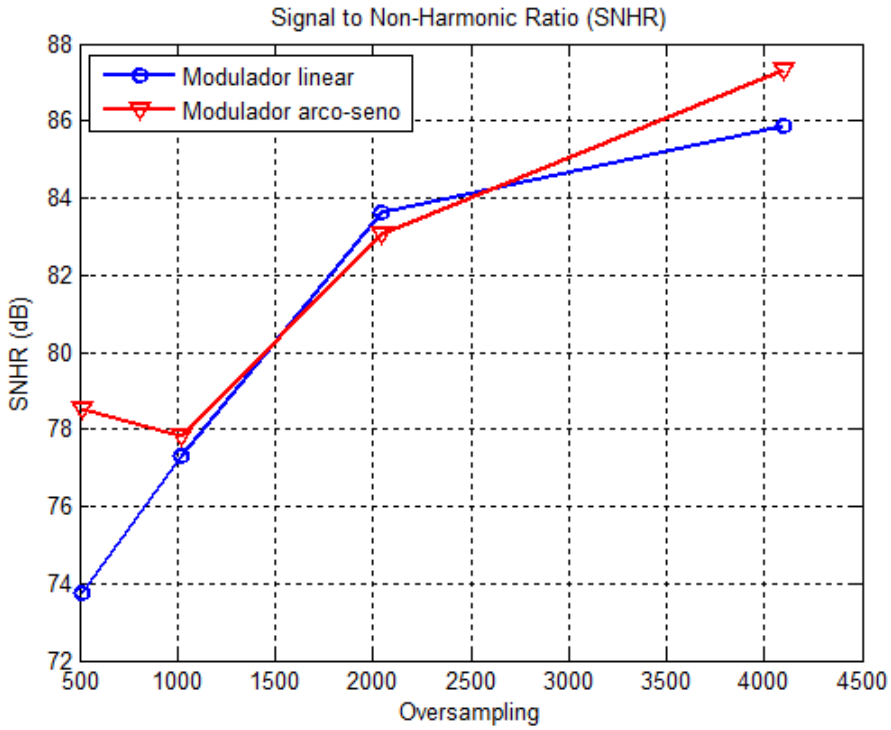


Figura 65 – Gráficos de $SNHR$ para os moduladores linear e arco-seno.

A Figura 66 mostra a variação do $ENOB$ dos moduladores com o aumento do *oversampling*. Observa-se que para o quantizador arco-seno o $ENOB$ aumenta e para o linear ele diminui.

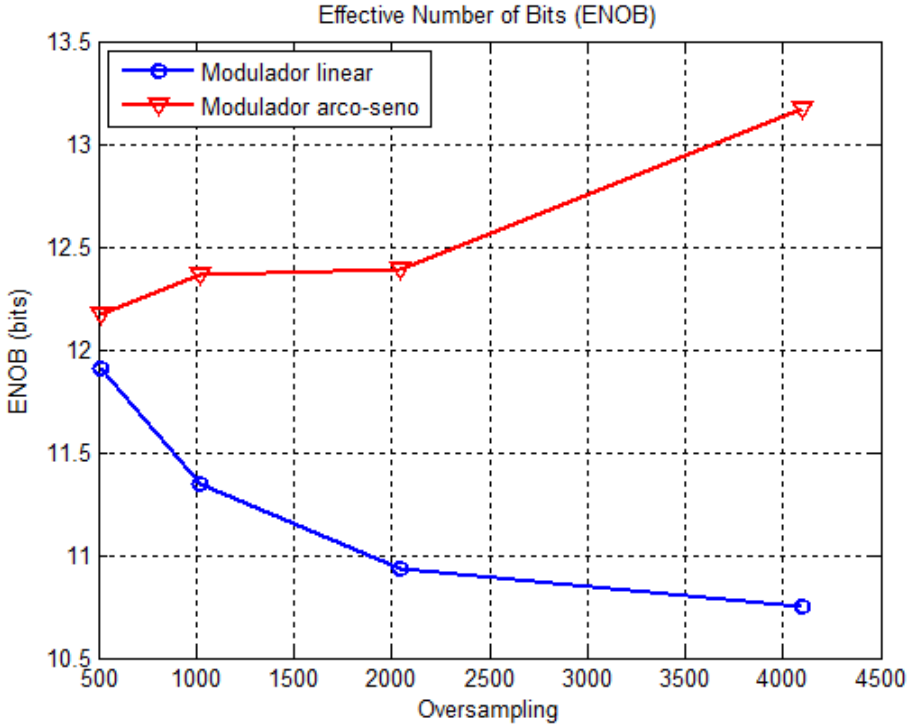


Figura 66 – Gráficos de $ENOB$ para os moduladores linear e arco-seno.

Na Figura 67 pode-se observar a variação da $SFDR$. Para o quantizador linear

esse valor decresce com o aumento do *oversampling*, já para o quantizador arco-seno o valor oscila dependendo do *oversampling* que esta sendo utilizado.

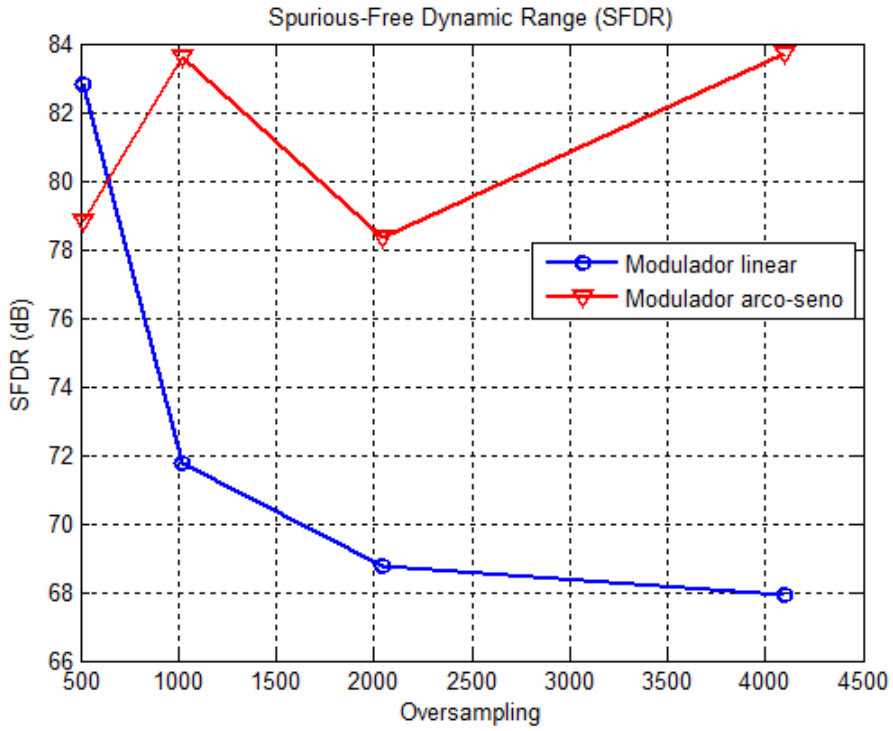


Figura 67 – Gráficos de $SFDR$ para os moduladores linear e arco-seno.

Por fim a Figura 68 mostra a variação do $SINAD$ para os moduladores. Observa-se que assim como no caso do $ENOB$, a $SINAD$ apresenta comportamento divergente entre os dois moduladores. No modulador com quantizador arco-seno ela aumenta com o *oversampling* e no outro ela diminui.

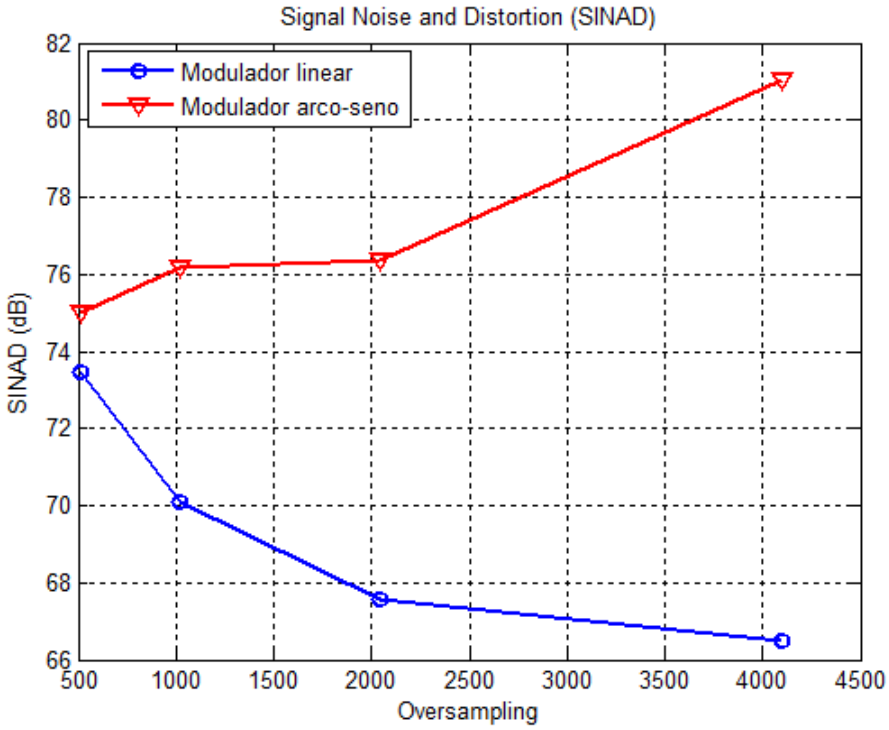


Figura 68 – Gráficos de $SINAD$ para os moduladores linear e arco-seno.

Parte III

Conclusão

9 Conclusão

Para muitos tipos de sinais, a quantização linear não é a mais eficiente podendo gerar grandes erros de quantização e consequentemente a perda de informação do sinal. Um método eficiente para resolver esse problema é implementar um sistema de quantização não linear, onde o tamanho dos passos de quantização não é constante.

Ao longo da primeira parte deste trabalho, apresentou-se conceitos fundamentais para a elaboração do mesmo. Abordou-se informações sobre conversores A/D, seu funcionamento, características, topologias, parâmetros estáticos e dinâmicos; a teoria sobre modelagem de processos de quantização, transformada da incerteza, análise do quantizador arco-seno e proposta de implementação do circuito; os fundamentos da metodologia de projeto *Top-Down* e os conceitos básicos da linguagem Verilog-A. A partir da fundamentação teórica, modelou-se um quantizador linear e um quantizador com distribuição arco-seno utilizando-se topologia $flash$ de 4 bits e a HDL Verilog-A. Em seguida foram feitas as simulações dos dois sistemas para validação dos resultados e comparação entre os mesmos.

Com os resultados obtidos na modelagem dos dois quantizadores, pode-se observar que o quantizador linear apresentou passos de quantização constantes e consequentemente seu erro de quantização se apresentou constante exceto nas extremidades do sinal. Já o quantizador não linear apresentou um erro de quantização menor nas extremidades do sinal e comportamento modular. Na análise em frequência dos quantizadores notou-se que os primeiros harmônicos importantes apareceram em frequências muito maiores que a frequência do sinal no quantizador arco-seno, sugerindo assim que é mais fácil filtrar esses harmônicos e conseguir um espectro mais limpo com o quantizador arco-seno do que com o quantizador linear.

Na segunda parte do trabalho realizou-se a modelagem e validação de todos os blocos necessários para a implementação de um modulador Σ-Δ *multi-bit* de 1ª ordem e aplicou-se os dois quantizadores modelados anteriormente afim de comparar o desempenho dos mesmos. Por fim analisou-se as repostas dos moduladores no tempo e na frequência e levantou-se os parâmetros dinâmicos: THD, $ENOB$, $SNHR$, $SINAD$ e $SFDR$, comparando os resultados obtidos.

Os dois quantizadores aplicados no modulador Σ-Δ funcionaram bem no sistema, na análise feita no tempo observou-se que o erro de quantização para o modulador linear apresenta um formato de serra/sino e o do modulador arco-seno um comportamento modular, uma vez que os passos de quantização são menores nas extremidades do sinal, isso gera um erro de quantização menor nessas extremidades e permite um sinal com excur-

são de entrada maior. Na análise em frequência observou-se que as harmônicas ímpares apresentam uma potência maior no modulador linear fazendo com que ele tenha uma distorção harmônica maior do que no modulador o arco-seno.

9.1 Trabalhos Futuros

Para trabalhos futuros é importante realizar a modelagem dos quantizadores linear e arco-seno com um numero maior de bits, para que possa ser feita uma análise de desempenho dos moduladores com relação ao número de bits dos quantizadores. Deve ser feito também a extração dos parâmetros estáticos do conversor para uma melhor caracterização do sistema.

É importante também implementar outras topologias de conversores A/D e aplicar nelas o quantizador arco-seno para validar seu funcionamento em várias arquiteturas.

Por fim, deverá ser feito o projeto a nível de transistores do quantizador arco-seno, seguindo todas as etapas de projeto, simulação e otimização dos blocos a nível de circuito, validação de corners e Monte Carlo, geração do layout e a prototipação do quantizador para fabricação.

Referências

ALI, M. N.; ZOHDY, M. A. Interactive kalman filtering for differential and gaussian frequency shift keying modulation with application in bluetooth. *Journal of Signal and Information Processing*, Scientific Research Publishing, v. 3, n. 01, p. 63, 2012. Citado 2 vezes nas páginas 19 e 51.

ALLEN, P. E.; HOLBERG, D. R. *CMOS analog circuit design*. [S.l.]: Oxford Univ. Press, 2002. Citado 6 vezes nas páginas 19, 23, 39, 41, 44 e 45.

BAKER, B. How delta-sigma adcs work, part 1. *Analog Applications*, 2011. Citado 2 vezes nas páginas 19 e 46.

BAKER, R. J. *CMOS: mixed-signal circuit design*. [S.l.]: john Wiley & sons, 2008. Citado 4 vezes nas páginas 19, 40, 49 e 50.

FLORES, M. d. G. C. C. Teste embarcado de conversores analógico-digitais. 2003. Citado 4 vezes nas páginas 19, 41, 42 e 43.

HAN, Y. et al. Design of analog circuits in multi-bit quantized audio dac [j]. *Journal of Zhejiang University (Engineering Science*, p. 1571–1575, 2011. Citado na página 49.

JOHANN, M. Estrutura de roteamento em circuitos vlsi. *Porto Alegre: CPGCC da UFRGS*, p. 16, 1997. Citado 3 vezes nas páginas 19, 35 e 36.

KESTER, W. Analog digital conversion. usa: Analog devices. *Inc*, v. 5, 2004. Citado na página 45.

KUNDERT, K.; CHANG, H. Top-down design and verification of mixed-signal circuits. *www. designers-guide. com*, 2005. Citado na página 35.

LATHI, B. P. *Modern digital and analog communication systems*. [S.l.]: Oxford University Press, Inc., 1990. Citado 2 vezes nas páginas 19 e 30.

LI, Y.; HE, L. First-order continuous-time sigma-delta modulator. In: IEEE. *Quality Electronic Design, 2007. ISQED'07. 8th International Symposium on*. [S.l.], 2007. p. 229–232. Citado na página 47.

MEDEIROS, J. E. G.; HADDAD, S. A. P. Nonlinear quantizer design in data conversion systems using the unscented transform. In: *ISCAS*. [S.l.]: IEEE, 2017. Citado 12 vezes nas páginas 19, 20, 51, 52, 53, 54, 55, 56, 57, 61, 75 e 78.

MERWE, R. van der; DOUCET, A. N de freitas n and e wan, the unscented particle filter. *Adv. Neural Inform. Process. Syst*, 2000. Citado na página 51.

PAPOULIS, A.; PILLAI, S. U. *Probability, random variables, and stochastic processes*. [S.l.]: Tata McGraw-Hill Education, 2002. Citado na página 52.

PARK, S. Principles of sigma-delta modulation for analog-to-digital converters. Motorola, 1999. Citado 4 vezes nas páginas 19, 47, 48 e 49.

RAZAVI, B. *Principles of Data Conversion System Design*. [S.l.]: IEEE Press, 1995. ISBN 0780310934,9780780310933. Citado 2 vezes nas páginas 39 e 52.

SCHREIER, R.; TEMES, G. C. et al. *Understanding delta-sigma data converters*. [S.l.]: IEEE press Piscataway, NJ, 2005. v. 74. Citado na página 46.

TILDEN, S. J.; LINNENBRINK, T. E.; GREEN, P. J. Overview of ieee-std-1241 "standard for terminology and test methods for analog-to-digital converters". In: IEEE. *Instrumentation and Measurement Technology Conference, 1999. IMTC/99. Proceedings of the 16th IEEE*. [S.l.], 1999. v. 3, p. 1498–1503. Citado 3 vezes nas páginas 40, 42 e 43.

UHLMANN, J. K. *Simultaneous map building and localization for real time applications*. [S.l.], 1994. Citado na página 51.

ZINKE, K. S. K. *The designer's guide to Verilog-AMS*. 1st edition. ed. [S.l.]: Kluwer Academic Publishers, 2004. (Designer's guide book series). Citado 4 vezes nas páginas 19, 35, 36 e 37.

ZIQUAN, T. et al. The design of a multi-bit quantization sigma-delta modulator. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, v. 6, n. 5, p. 265–274, 2013. Citado 2 vezes nas páginas 19 e 50.

ZURITA, M. *Metodologia e Fluxo de Projeto de Sistemas VLSI Digitais*. Tese (Doutorado) — Universidade Federal do Piauí, 2013. Citado na página 35.

Apêndices

APÊNDICE A – Códigos da Modelagem em Verilog-A.

A.1 Decoder

```
41 // VerilogA for ADC_flash, decoder_16bits
42 // Autor : Laryssa Lorrany Olinda Costa
43 // Matr cula : 12/0060973
44
45 `include "constants.vams"
46 `include "disciplines.vams"
47
48 module decoder_16bits(t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,
       t14,t15,d1,d2,d3,d4, avdd, agnd);
49 inout t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,d1,d2,d3
       ,d4,avdd,agnd;
50 electrical t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,d1,
       d2,d3,d4,avdd,agnd;
51 real tone, ttwo, tthree, tfour, tfive, tsix, tseven, teight,
       tnine, tten, televen, ttwelve, tthirteen, tfourteen, tfifteen;
52
53 analog begin
54         if ((V(d1) == V(agnd))&&(V(d2) == V(agnd))&&(V(d3) == V(
              agnd))&&(V(d4) == V(agnd)))
55         begin
56         tone =          V(agnd);
57         ttwo=           V(agnd);
58         tthree=         V(agnd);
59         tfour=          V(agnd);
60         tfive=          V(agnd);
61         tsix=           V(agnd);
62         tseven=         V(agnd);
63         teight=         V(agnd);
64         tnine=          V(agnd);
65         tten=           V(agnd);
66         televen=        V(agnd);
67         ttwelve=        V(agnd);
68         tthirteen=      V(agnd);
```

```
69          tfourteen=      V(agnd);
70          tfifteen=       V(agnd);
71
72          end
73          if ((V(d1) == V(agnd))&&(V(d2) == V(agnd))&&(V(d3) == V(
               agnd))&&(V(d4) == V(avdd)))
74          begin
75          tone =          V(agnd);
76          ttwo=           V(agnd);
77          tthree=         V(agnd);
78          tfour=          V(agnd);
79          tfive=          V(agnd);
80          tsix=           V(agnd);
81          tseven=         V(agnd);
82          teight=         V(agnd);
83          tnine=          V(agnd);
84          tten=           V(agnd);
85          televen=        V(agnd);
86          ttwelve=        V(agnd);
87          tthirteen=      V(agnd);
88          tfourteen=      V(agnd);
89          tfifteen=       V(avdd);
90
91          end
92          if ((V(d1) == V(agnd))&&(V(d2) == V(agnd))&&(V(d3) == V(
               avdd))&&(V(d4) == V(agnd)))
93          begin
94          tone =          V(agnd);
95          ttwo=           V(agnd);
96          tthree=         V(agnd);
97          tfour=          V(agnd);
98          tfive=          V(agnd);
99          tsix=           V(agnd);
100         tseven=         V(agnd);
101         teight=         V(agnd);
102         tnine=          V(agnd);
103         tten=           V(agnd);
104         televen=        V(agnd);
105         ttwelve=        V(agnd);
106         tthirteen=      V(agnd);
107         tfourteen=      V(avdd);
108         tfifteen=       V(agnd);
```

```
109
110         end
111         if ((V(d1) == V(agnd))&&(V(d2) == V(agnd))&&(V(d3) == V(
               avdd))&&(V(d4) == V(avdd)))
112         begin
113         tone =          V(agnd);
114         ttwo=           V(agnd);
115         tthree=         V(agnd);
116         tfour=          V(agnd);
117         tfive=          V(agnd);
118         tsix=           V(agnd);
119         tseven=         V(agnd);
120         teight=         V(agnd);
121         tnine=          V(agnd);
122         tten=           V(agnd);
123         televen=        V(agnd);
124         ttwelve=        V(agnd);
125         tthirteen=      V(avdd);
126         tfourteen=      V(agnd);
127         tfifteen=       V(agnd);
128
129         end
130         if ((V(d1) == V(agnd))&&(V(d2) == V(avdd))&&(V(d3) == V(
               agnd))&&(V(d4) == V(agnd)))
131         begin
132         tone =          V(agnd);
133         ttwo=           V(agnd);
134         tthree=         V(agnd);
135         tfour=          V(agnd);
136         tfive=          V(agnd);
137         tsix=           V(agnd);
138         tseven=         V(agnd);
139         teight=         V(agnd);
140         tnine=          V(agnd);
141         tten=           V(agnd);
142         televen=        V(agnd);
143         ttwelve=        V(avdd);
144         tthirteen=      V(agnd);
145         tfourteen=      V(agnd);
146         tfifteen=       V(agnd);
147
148         end
```

```
149         if ((V(d1) == V(agnd))&&(V(d2) == V(avdd))&&(V(d3) == V(
                agnd))&&(V(d4) == V(avdd)))
150         begin
151         tone =          V(agnd);
152         ttwo=           V(agnd);
153         tthree=         V(agnd);
154         tfour=          V(agnd);
155         tfive=          V(agnd);
156         tsix=           V(agnd);
157         tseven=         V(agnd);
158         teight=         V(agnd);
159         tnine=          V(agnd);
160         tten=           V(agnd);
161         televen=        V(avdd);
162         ttwelve=        V(agnd);
163         tthirteen=      V(agnd);
164         tfourteen=      V(agnd);
165         tfifteen=       V(agnd);
166
167         end
168         if ((V(d1) == V(agnd))&&(V(d2) == V(avdd))&&(V(d3) == V(
                avdd))&&(V(d4) == V(agnd)))
169         begin
170         tone =          V(agnd);
171         ttwo=           V(agnd);
172         tthree=         V(agnd);
173         tfour=          V(agnd);
174         tfive=          V(agnd);
175         tsix=           V(agnd);
176         tseven=         V(agnd);
177         teight=         V(agnd);
178         tnine=          V(agnd);
179         tten=           V(avdd);
180         televen=        V(agnd);
181         ttwelve=        V(agnd);
182         tthirteen=      V(agnd);
183         tfourteen=      V(agnd);
184         tfifteen=       V(agnd);
185
186         end
187         if ((V(d1) == V(agnd))&&(V(d2) == V(avdd))&&(V(d3) == V(
                avdd))&&(V(d4) == V(avdd)))
```

```
188         begin
189         tone =          V(agnd);
190         ttwo=           V(agnd);
191         tthree=         V(agnd);
192         tfour=          V(agnd);
193         tfive=          V(agnd);
194         tsix=           V(agnd);
195         tseven=         V(agnd);
196         teight=         V(agnd);
197         tnine=          V(avdd);
198         tten=           V(agnd);
199         televen=        V(agnd);
200         ttwelve=        V(agnd);
201         tthirteen=      V(agnd);
202         tfourteen=      V(agnd);
203         tfifteen=       V(agnd);
204
205         end
206         if ((V(d1) == V(avdd))&&(V(d2) == V(agnd))&&(V(d3) == V(
               agnd))&&(V(d4) == V(agnd)))
207         begin
208         tone =          V(agnd);
209         ttwo=           V(agnd);
210         tthree=         V(agnd);
211         tfour=          V(agnd);
212         tfive=          V(agnd);
213         tsix=           V(agnd);
214         tseven=         V(agnd);
215         teight=         V(avdd);
216         tnine=          V(agnd);
217         tten=           V(agnd);
218         televen=        V(agnd);
219         ttwelve=        V(agnd);
220         tthirteen=      V(agnd);
221         tfourteen=      V(agnd);
222         tfifteen=       V(agnd);
223
224         end
225         if ((V(d1) == V(avdd))&&(V(d2) == V(agnd))&&(V(d3) == V(
               agnd))&&(V(d4) == V(avdd)))
226         begin
227         tone =          V(agnd);
```

```
228         ttwo=           V(agnd);
229         tthree=         V(agnd);
230         tfour=          V(agnd);
231         tfive=          V(agnd);
232         tsix=           V(agnd);
233         tseven=         V(avdd);
234         teight=         V(agnd);
235         tnine=          V(agnd);
236         tten=           V(agnd);
237         televen=        V(agnd);
238         ttwelve=        V(agnd);
239         tthirteen=      V(agnd);
240         tfourteen=      V(agnd);
241         tfifteen=       V(agnd);
242
243         end
244         if ((V(d1) == V(avdd))&&(V(d2) == V(agnd))&&(V(d3) == V(
               avdd))&&(V(d4) == V(agnd)))
245         begin
246         tone =          V(agnd);
247         ttwo=           V(agnd);
248         tthree=         V(agnd);
249         tfour=          V(agnd);
250         tfive=          V(agnd);
251         tsix=           V(avdd);
252         tseven=         V(agnd);
253         teight=         V(agnd);
254         tnine=          V(agnd);
255         tten=           V(agnd);
256         televen=        V(agnd);
257         ttwelve=        V(agnd);
258         tthirteen=      V(agnd);
259         tfourteen=      V(agnd);
260         tfifteen=       V(agnd);
261
262         end
263         if ((V(d1) == V(avdd))&&(V(d2) == V(agnd))&&(V(d3) == V(
               avdd))&&(V(d4) == V(avdd)))
264         begin
265         tone =          V(agnd);
266         ttwo=           V(agnd);
267         tthree=         V(agnd);
```

```
268         tfour=          V(agnd);
269         tfive=          V(avdd);
270         tsix=           V(agnd);
271         tseven=         V(agnd);
272         teight=         V(agnd);
273         tnine=          V(agnd);
274         tten=           V(agnd);
275         televen=        V(agnd);
276         ttwelve=        V(agnd);
277         tthirteen=      V(agnd);
278         tfourteen=      V(agnd);
279         tfifteen=       V(agnd);
280
281         end
282         if ((V(d1) == V(avdd))&&(V(d2) == V(avdd))&&(V(d3) == V(
               agnd))&&(V(d4) == V(agnd)))
283         begin
284         tone =          V(agnd);
285         ttwo=           V(agnd);
286         tthree=         V(agnd);
287         tfour=          V(avdd);
288         tfive=          V(agnd);
289         tsix=           V(agnd);
290         tseven=         V(agnd);
291         teight=         V(agnd);
292         tnine=          V(agnd);
293         tten=           V(agnd);
294         televen=        V(agnd);
295         ttwelve=        V(agnd);
296         tthirteen=      V(agnd);
297         tfourteen=      V(agnd);
298         tfifteen=       V(agnd);
299
300         end
301         if ((V(d1) == V(avdd))&&(V(d2) == V(avdd))&&(V(d3) == V(
               agnd))&&(V(d4) ==V(avdd)))
302         begin
303         tone =          V(agnd);
304         ttwo=           V(agnd);
305         tthree=         V(avdd);
306         tfour=          V(agnd);
307         tfive=          V(agnd);
```

```
308         tsix=           V(agnd);
309         tseven=         V(agnd);
310         teight=         V(agnd);
311         tnine=          V(agnd);
312         tten=           V(agnd);
313         televen=        V(agnd);
314         ttwelve=        V(agnd);
315         tthirteen=      V(agnd);
316         tfourteen=      V(agnd);
317         tfifteen=       V(agnd);
318
319         end
320         if ((V(d1) == V(avdd))&&(V(d2) == V(avdd))&&(V(d3) == V(
               avdd))&&(V(d4) == V(agnd)))
321         begin
322         tone =          V(agnd);
323         ttwo=           V(avdd);
324         tthree=         V(agnd);
325         tfour=          V(agnd);
326         tfive=          V(agnd);
327         tsix=           V(agnd);
328         tseven=         V(agnd);
329         teight=         V(agnd);
330         tnine=          V(agnd);
331         tten=           V(agnd);
332         televen=        V(agnd);
333         ttwelve=        V(agnd);
334         tthirteen=      V(agnd);
335         tfourteen=      V(agnd);
336         tfifteen=       V(agnd);
337
338         end
339         if ((V(d1) == V(avdd))&&(V(d2) == V(avdd))&&(V(d3) == V(
               avdd))&&(V(d4) == V(avdd)))
340         begin
341         tone =          V(avdd);
342         ttwo=           V(agnd);
343         tthree=         V(agnd);
344         tfour=          V(agnd);
345         tfive=          V(agnd);
346         tsix=           V(agnd);
347         tseven=         V(agnd);
```

```
348         teight=         V(agnd);
349         tnine=          V(agnd);
350         tten=           V(agnd);
351         televen=        V(agnd);
352         ttwelve=        V(agnd);
353         tthirteen=      V(agnd);
354         tfourteen=      V(agnd);
355         tfifteen=       V(agnd);
356
357         end
358         V(t1)<+ transition(tone,1f,1f);
359         V(t2)<+ transition(ttwo,1f,1f);
360         V(t3)<+ transition(tthree,1f,1f);
361         V(t4)<+ transition(tfour,1f,1f);
362         V(t5)<+ transition(tfive,1f,1f);
363         V(t6)<+ transition(tsix,1f,1f);
364         V(t7)<+ transition(tseven,1f,1f);
365         V(t8)<+ transition(teight,1f,1f);
366         V(t9)<+ transition(tnine,1f,1f);
367         V(t10)<+ transition(tten,1f,1f);
368         V(t11)<+ transition(televen,1f,1f);
369         V(t12)<+ transition(ttwelve,1f,1f);
370         V(t13)<+ transition(tthirteen,1f,1f);
371         V(t14)<+ transition(tfourteen,1f,1f);
372         V(t15)<+ transition(tfifteen,1f,1f);
373
374         end
375 endmodule
```

A.2 Encoder

```
376 // VerilogA for ADC_flash, therm-to-binary-4bits
377 // Autor: Laryssa Lorrany Olinda Costa
378 // Matr cula: 12/0060973
379
380 `include "constants.vams"
381 `include "disciplines.vams"
382
383 module therm4bits(t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,
    t15,d1,d2,d3,d4,avdd,agnd);
```

```
384 inout t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,d1,d2,d3
    ,d4,avdd,agnd;
385 parameter td =1f, tf =1f;
386 electrical t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,d1,
    d2,d3,d4,avdd,agnd;
387 real done, dtwo, dthree, dfour;
388
389 analog begin
390
391
392     if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(agnd)
       )&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6) == V(agnd
       ))&&(V(t7) == V(agnd))&&(V(t8) == V(agnd))&&(V(t9) == V(
       agnd))&&(V(t10) == V(agnd))&&(V(t11) == V(agnd))&&(V(t12)
       == V(agnd))&&(V(t13) == V(agnd))&&(V(t14) == V(agnd))&&(V(
       t15) == V(agnd)))
393             begin
394                     done = V(agnd);
395                     dtwo = V(agnd);
396                     dthree = V(agnd);
397                     dfour= V(agnd);
398         end
399
400
401         if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
           agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
           == V(agnd))&&(V(t7) == V(agnd))&&(V(t8) == V(agnd))&&(V
           (t9) == V(agnd))&&(V(t10) == V(agnd))&&(V(t11) == V(
           agnd))&&(V(t12) == V(agnd))&&(V(t13) == V(agnd))&&(V(
           t14) == V(agnd))&&(V(t15) == V(avdd)))
402             begin
403                     done = V(agnd);
404                     dtwo = V(agnd);
405                     dthree = V(agnd);
406                     dfour= V(avdd);
407         end
408
409          if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
           agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
            == V(agnd))&&(V(t7) == V(agnd))&&(V(t8) == V(agnd))
           &&(V(t9) == V(agnd))&&(V(t10) == V(agnd))&&(V(t11) ==
           V(agnd))&&(V(t12) == V(agnd))&&(V(t13) == V(agnd))&&(V
```

```
            (t14) == V(avdd))&&(V(t15) == V(avdd)))
410         begin
411                 done = V(agnd);
412                 dtwo = V(agnd);
413                 dthree = V(avdd);
414                 dfour= V(agnd);
415         end
416
417     if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
            agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
            == V(agnd))&&(V(t7) == V(agnd))&&(V(t8) == V(agnd))&&(V
            (t9) == V(agnd))&&(V(t10) == V(agnd))&&(V(t11) == V(
            agnd))&&(V(t12) == V(agnd))&&(V(t13) == V(avdd))&&(V(
            t14) == V(avdd))&&(V(t15) == V(avdd)))
418     begin
419                 done = V(agnd);
420                 dtwo = V(agnd);
421                 dthree = V(avdd);
422                 dfour= V(avdd);
423     end
424     if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
            agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
            == V(agnd))&&(V(t7) == V(agnd))&&(V(t8) == V(agnd))&&(V
            (t9) == V(agnd))&&(V(t10) == V(agnd))&&(V(t11) == V(
            agnd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
            t14) == V(avdd))&&(V(t15) == V(avdd)))
425     begin
426                 done = V(agnd);
427             dtwo = V(avdd);
428                 dthree = V(agnd);
429                 dfour= V(agnd);
430     end
431     if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
            agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
            == V(agnd))&&(V(t7) == V(agnd))&&(V(t8) == V(agnd))&&(V
            (t9) == V(agnd))&&(V(t10) == V(agnd))&&(V(t11) == V(
            avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
            t14) == V(avdd))&&(V(t15) == V(avdd)))
432     begin
433                 done = V(agnd);
434                 dtwo = V(avdd);
435                 dthree = V(agnd);
```

```
436                 dfour= V(avdd);
437         end
438          if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
                agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
                 == V(agnd))&&(V(t7) == V(agnd))&&(V(t8) == V(agnd))
                &&(V(t9) == V(agnd))&&(V(t10) == V(avdd))&&(V(t11) ==
                V(avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V
                (t14) == V(avdd))&&(V(t15) == V(avdd)))
439         begin
440                 done = V(agnd);
441                 dtwo = V(avdd);
442                 dthree = V(avdd);
443                 dfour= V(agnd);
444         end
445
446         if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
               agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
               == V(agnd))&&(V(t7) == V(agnd))&&(V(t8) == V(agnd))&&(V
               (t9) == V(avdd))&&(V(t10) == V(avdd))&&(V(t11) == V(
               avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
               t14) == V(avdd))&&(V(t15) == V(avdd)))
447          begin
448                 done = V(agnd);
449                 dtwo = V(avdd);
450                 dthree = V(avdd);
451                 dfour= V(avdd);
452          end
453         if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
               agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
               == V(agnd))&&(V(t7) == V(agnd))&&(V(t8) == V(avdd))&&(V
               (t9) == V(avdd))&&(V(t10) == V(avdd))&&(V(t11) == V(
               avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
               t14) == V(avdd))&&(V(t15) == V(avdd)))
454         begin
455                 done = V(avdd);
456                 dtwo = V(agnd);
457                 dthree = V(agnd);
458                 dfour= V(agnd);
459         end
460         if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
               agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
               == V(agnd))&&(V(t7) == V(avdd))&&(V(t8) == V(avdd))&&(V
```

```
            (t9) == V(avdd))&&(V(t10) == V(avdd))&&(V(t11) == V(
            avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
            t14) == V(avdd))&&(V(t15) == V(avdd)))
461         begin
462                     done = V(avdd);
463                     dtwo = V(agnd);
464                     dthree = V(agnd);
465                     dfour= V(avdd);
466         end
467         if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
            agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(agnd))&&(V(t6)
            == V(avdd))&&(V(t7) == V(avdd))&&(V(t8) == V(avdd))&&(V
            (t9) == V(avdd))&&(V(t10) == V(avdd))&&(V(t11) == V(
            avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
            t14) == V(avdd))&&(V(t15) == V(avdd)))
468         begin
469                     done = V(avdd);
470                     dtwo = V(agnd);
471                     dthree = V(avdd);
472                     dfour= V(agnd);
473         end
474
475         if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
            agnd))&&(V(t4) == V(agnd))&&(V(t5) == V(avdd))&&(V(t6)
            == V(avdd))&&(V(t7) == V(avdd))&&(V(t8) == V(avdd))&&(V
            (t9) == V(avdd))&&(V(t10) == V(avdd))&&(V(t11) == V(
            avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
            t14) == V(avdd))&&(V(t15) == V(avdd)))
476         begin
477                     done = V(avdd);
478                     dtwo = V(agnd);
479                     dthree = V(avdd);
480                     dfour= V(avdd);
481         end
482
483         if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
            agnd))&&(V(t4) == V(avdd))&&(V(t5) == V(avdd))&&(V(t6)
            == V(avdd))&&(V(t7) == V(avdd))&&(V(t8) == V(avdd))&&(V
            (t9) == V(avdd))&&(V(t10) == V(avdd))&&(V(t11) == V(
            avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
            t14) == V(avdd))&&(V(t15) == V(avdd)))
484         begin
```

```
485                 done = V(avdd);
486                     dtwo = V(avdd);
487                     dthree = V(agnd);
488                     dfour= V(agnd);
489             end
490             if ((V(t1) == V(agnd))&&(V(t2) == V(agnd))&&(V(t3) == V(
                    avdd))&&(V(t4) == V(avdd))&&(V(t5) == V(avdd))&&(V(t6)
                    == V(avdd))&&(V(t7) == V(avdd))&&(V(t8) ==V(avdd))&&(V(
                    t9) == V(avdd))&&(V(t10) ==V(avdd))&&(V(t11) == V(avdd)
                    )&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(t14) ==
                     V(avdd))&&(V(t15) == V(avdd)))
491             begin
492                     done = V(avdd);
493                     dtwo = V(avdd);
494                     dthree = V(agnd);
495                     dfour= V(avdd);
496             end
497             if ((V(t1) == V(agnd))&&(V(t2) == V(avdd))&&(V(t3) == V(
                    avdd))&&(V(t4) == V(avdd))&&(V(t5) == V(avdd))&&(V(t6)
                    == V(avdd))&&(V(t7) == V(avdd))&&(V(t8) == V(avdd))&&(V
                    (t9) == V(avdd))&&(V(t10) == V(avdd))&&(V(t11) == V(
                    avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
                    t14) == V(avdd))&&(V(t15) == V(avdd)))
498             begin
499                     done = V(avdd);
500                     dtwo = V(avdd);
501                     dthree = V(avdd);
502                     dfour= V(agnd);
503             end
504             if ((V(t1) == V(avdd))&&(V(t2) == V(avdd))&&(V(t3) == V(
                    avdd))&&(V(t4) == V(avdd))&&(V(t5) == V(avdd))&&(V(t6)
                    == V(avdd))&&(V(t7) == V(avdd))&&(V(t8) == V(avdd))&&(V
                    (t9) == V(avdd))&&(V(t10) == V(avdd))&&(V(t11) == V(
                    avdd))&&(V(t12) == V(avdd))&&(V(t13) == V(avdd))&&(V(
                    t14) == V(avdd))&&(V(t15) == V(avdd)))
505             begin
506                     done = V(avdd);
507                     dtwo = V(avdd);
508                     dthree = V(avdd);
509                     dfour= V(avdd);
510             end
511
```

```
512         V(d1)<+ transition(done,td,tf);
513         V(d2)<+ transition(dtwo,td,tf);
514         V(d3)<+ transition(dthree,td,tf);
515         V(d4)<+ transition(dfour,td,tf);
516 end
517
518 endmodule
```

A.3 Chave

```
520 // VerilogA for Relay
521 // Autor : Laryssa Lorrany Olinda Costa
522 // Matr cula : 12/0060973
523
524   include    " constants . vams "
525 `include "disciplines.vams"
526
527 module relay (p, n, ps, ns);
528 parameter real thresh=0; // threshold
529 output p, n;
530 input ps, ns; // contacts
531 electrical p, n, ps, ns;
532
533 analog begin
534
535         if (V(ps,ns) > thresh)
536                 V(p,n) <+ 0;
537         else
538                 I(p,n) <+ 0;
539         end
540
541 endmodule
```

A.4 Modulador Σ-Δ *Single-bit*

```
542 / VerilogA for Modulador-SD one-bit
543 // Matr cula : 12/0060973
544
545 `include "discipline.h"
546 `include "constants.h"
```

```
547
548 // Based on the OVI Verilog-A Language Reference Manual, version
      1.0 1996
549
550 module sigmadelta_1storder(vin, vclk, vout);
551 input vin, vclk;
552 output vout;
553 electrical vin, vout, vclk ;
554 parameter real vth=0.0 ;
555 parameter real vout_high=5.0 ;
556 parameter real vtrans_clk=2.5 ;
557 parameter real trise=20n from (0:inf);
558 parameter real tfall=20n from (0:inf);
559 parameter real tdel=0.0 from [0:inf);
560
561    real vsum;
562    real vd;
563    real vint ;
564    real vout_val;
565
566    real hi, lo;
567
568    analog begin
569
570       @ ( initial_step ) begin
571          vout_val = 1.0;
572          hi=1.0;
573          lo=-1.0;
574
575       end
576
577       @ (cross(V(vclk)-vtrans_clk, 1.0))begin
578
579          // summing junction
580          vsum = V(vin) - vd ;
581
582          // integrator
583          vint = vsum + vint;
584
585          // quantizer
586          if (vint > vth)
587             vout_val = hi ;
```

```
588             else
589                 vout_val = lo ;
590
591             // D2A
592             vd = vout_high * vout_val ;
593
594         end
595         V(vout) <+ transition(vout_val, tdel, trise, tfall);
596     end
597 endmodule
598
599
600
601
602
603
604
605 \section{Integrador}
606
607 \begin{lstlisting}[frame=single][!h]
608 // VerilogA for Modulador-SD, integrator
609 // Autor : Laryssa Lorrany Olinda Costa
610 // Matr cula : 12/0060973
611
612 `include "constants.vams"
613 `include "disciplines.vams"
614
615 module integrador(in, out, Clk);
616             inout in, out, Clk;
617             electrical in, out, Clk;
618             parameter real scale = 1000000;
619             parameter real vref = 0.5;
620             parameter real vth = 0;
621             real state;
622         analog begin
623                 @(cross((V(Clk)-vth), +1)) begin
624                         state = V(in);
625                 end
626                         V(out) <+ idt(scale*(state-vref), 0.5);
627             end
628
629 endmodule
```

A.5 Subtrator

```
631
632 // VerilogA for Modulador-SD, subtrator
633 // Autor : Laryssa Lorrany Olinda Costa
634 // Matr cula : 12/0060973
635
636 `include "constants.vams"
637 `include "disciplines.vams"
638
639 module subtrator(in1,in2,out);
640
641         inout in1,in2,out;
642         electrical in1,in2,out;
643         analog begin
644                 V(out) <+ (V(in1)-V(in2));
645         end
646 endmodule
```

A.6 Amplificador

```
648
649 // VerilogA for Modulador-SD, amplifier
650 // Autor : Laryssa Lorrany Olinda Costa
651 // Matr cula : 12/0060973
652
653 `include "constants.vams"
654 `include "disciplines.vams"
655
656 module amplificador(in, out);
657         parameter real gain = 5;
658         inout in, out;
659         electrical in, out;
660
661         analog
662                 V(out) <+ gain*V(in);
663 endmodule
```