



UFAM - Engenharia da Computação

Stanley de Carvalho Monteiro

MODULADOR SIGMA-DELTA DIGITAL EM TECNOLOGIA CMOS 130 NM  
PARA REDUÇÃO DE ESPÚRIOS EM MODULAÇÃO ASK

Manaus  
Dezembro de 2025



UFAM - Engenharia da Computação

MODULADOR SIGMA-DELTA DIGITAL EM TECNOLOGIA CMOS 130 NM  
PARA REDUÇÃO DE ESPÚRIOS EM MODULAÇÃO ASK

Stanley de Carvalho Monteiro

Monografia de Graduação apresentada à  
Coordenação de Engenharia da Computação,  
UFAM, da Universidade Federal do  
Amazonas, como parte dos requisitos  
necessários à obtenção do título de  
Engenheiro da Computação.

Orientador: Thiago Brito Bezerra

Manaus  
Dezembro de 2025

#### Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

---

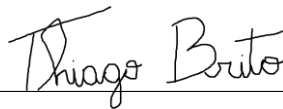
- M775m Monteiro, Stanley de Carvalho  
Modulador sigma-delta digital em tecnologia CMOS 130 nm para  
redução de espúrios em modulação ASK / Stanley de Carvalho  
Monteiro. - 2025.  
46 f. : il., color. ; 31 cm.
- Orientador(a): Thiago Brito Bezerra.  
Trabalho de Conclusão de Curso (graduação) - Universidade  
Federal do Amazonas, Faculdade de Tecnologia, Curso de  
Engenharia da Computação, Manaus, 2025.
1. Modulador sigma-delta. 2. CMOS 130 nm. 3. Modulação ASK.  
4. Redução de espúrios. 5. Verilog. I. Bezerra, Thiago Brito. II.  
Universidade Federal do Amazonas. Faculdade de Tecnologia.  
Curso de Engenharia da Computação. III. Título
-

MODULADOR SIGMA-DELTA DIGITAL EM TECNOLOGIA  
CMOS 130 NM PARA REDUÇÃO DE ESPÚRIOS EM  
MODULAÇÃO ASK

Stanley de Carvalho Monteiro

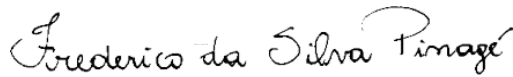
MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO CURSO DE  
ENGENHARIA DA COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO  
AMAZONAS, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A  
OBTENÇÃO DO GRAU DE ENGENHEIRO DA COMPUTAÇÃO.

Aprovada por:



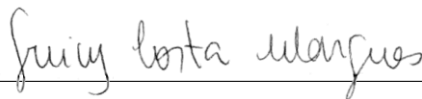
---

Prof. Thiago Brito Bezerra, Dr.



---

Prof. Frederico da Silva Pinagé, Dr.



---

Prof<sup>a</sup>. Greicy Costa Marques, Dr<sup>a</sup>.

MANAUS, AM – BRASIL

DEZEMBRO DE 2025

# Agradecimentos

Agradeço primeiramente à minha mãe, Itacilene Carvalho, e ao meu pai, Stanley Bulcão, pelo apoio em todos os momentos de minha vida. Ambos foram o pilar que me manteve firme e a força que me permitiu seguir em frente.

Expresso também minha gratidão ao meu irmão Marcus Vinícius, por me auxiliar diariamente nesta jornada e por estar sempre presente nos momentos difíceis e importantes.

À Universidade Federal do Amazonas (UFAM), sou profundamente grato pela oportunidade de realizar esta Tese e por todo o suporte institucional ao longo da minha formação.

Registro meus agradecimentos ao meu orientador, Prof. Thiago Brito, pela excelente orientação, dedicação e paciência durante todo o desenvolvimento deste projeto.

Aos amigos que fiz durante minha vida acadêmica, em especial Jhonatas, agradeço pela companhia constante, pelas comemorações compartilhadas e pelo apoio em momentos desafiadores, sempre trazendo a sensação de que eu nunca estava sozinho.

Resumo da Monografia apresentada à UFAM como parte dos requisitos necessários para a obtenção do grau de Engenheiro da Computação

MODULADOR SIGMA-DELTA DIGITAL EM TECNOLOGIA CMOS 130 NM  
PARA REDUÇÃO DE ESPÚRIOS EM MODULAÇÃO ASK

Stanley de Carvalho Monteiro

Dezembro/2025

Orientador: Thiago Brito Bezerra

Programa: Engenharia da Computação

Este trabalho apresenta o desenvolvimento de um modulador sigma-delta digital de primeira ordem com quantização binária (1 bit), descrito em Verilog e sintetizado em tecnologia CMOS de 130 nm, com aplicação em sistemas de modulação ASK em sua forma binária, também conhecida como Binary Amplitude Shift Keying (BASK), visando à redução de espúrios na frequência do clock. A metodologia abrange o estudo teórico de moduladores sigma-delta, seguido do projeto, simulação e integração do circuito ao controle da portadora ASK em nível de sistema, onde o bitstream gerado é utilizado para ditherizar o sinal de chaveamento. A análise foi realizada por meio de simulações temporais com GTKWave e espectrais via transformadas de Fourier (FFT) em Python, permitindo comparar o espectro do sinal ASK com e sem a aplicação do modulador sigma-delta. Os resultados demonstram que o uso do dithering sigma-delta reduz significativamente a amplitude dos espúrios próximos à frequência da portadora, sem comprometer a integridade do sinal modulado. Conclui-se que a técnica aplicada é eficaz e viável para melhorar a pureza espectral em transmissores digitais, oferecendo uma alternativa digital simples e de baixo custo à filtragem analógica tradicional.

Abstract of Monograph presented to UFAM as a partial fulfillment of the requirements for the degree of Engineer

DIGITAL SIGMA-DELTA MODULATOR IN 130 NM CMOS TECHNOLOGY  
FOR SPUR REDUCTION IN ASK MODULATION

Stanley de Carvalho Monteiro

December/2025

Advisor: Thiago Brito Bezerra

Department: Computer Engineering

This work presents the development of a first-order digital sigma-delta modulator with binary (1-bit) quantization, described in Verilog and synthesized in 130 nm CMOS technology, applied to ASK modulation systems in their binary form, also known as Binary Amplitude Shift Keying (BASK), aiming at the reduction of clock frequency spurs. The methodology includes the theoretical study of sigma-delta modulators, followed by the design, simulation, and integration of the circuit into the system-level control of the ASK carrier, where the generated bitstream is used to dither the switching signal. Analysis was carried out through time-domain simulations using GTKWave and spectral evaluations via Fourier Transforms (FFT) in Python, allowing comparison of the ASK signal spectrum with and without the application of the sigma-delta modulator. The results demonstrate that dithering with the sigma-delta modulator significantly reduces the amplitude of spurs near the carrier frequency without compromising the integrity of the modulated signal. It is concluded that the proposed technique is effective and feasible for improving spectral purity in digital transmitters, offering a simple and low-cost digital alternative to traditional analog filtering.

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Problema . . . . .	3
1.2	Justificativa . . . . .	4
1.3	Objetivos . . . . .	5
1.3.1	Objetivos Gerais . . . . .	5
1.3.2	Objetivos Específicos . . . . .	5
<b>2</b>	<b>Revisão da Literatura</b>	<b>8</b>
2.1	Fundamentos dos moduladores sigma-delta . . . . .	8
2.1.1	Sobreamostragem e equação de SNR . . . . .	9
2.1.2	Ordem do modulador, número de bits e compromisso de projeto	10
2.2	Arquiteturas digitais para dithering de portadora . . . . .	11
2.3	Síntese Digital em Verilog . . . . .	11
2.4	Fluxo de Projeto em CMOS 130 nm com PDKs Digitais . . . . .	13
2.5	GTKWave e Análise Espectral via FFT . . . . .	13
<b>3</b>	<b>Desenvolvimento</b>	<b>15</b>
3.1	Visão Geral do Sistema . . . . .	15
3.2	Arquitetura do Modulador Sigma-Delta . . . . .	16
3.3	Implementação em Verilog . . . . .	18
3.3.1	Módulo Principal: <code>sdm_1.v</code> . . . . .	18
3.3.2	Testbench: <code>tb_sdm.v</code> . . . . .	19
3.3.3	Módulo Top Lógico: <code>top_sdm</code> . . . . .	20
3.3.4	Módulo Top para Síntese Física: <code>top_sdm_phys</code> . . . . .	20
3.3.5	Síntese Física e Integração . . . . .	21

3.4	Simulação Funcional (GTKWave) . . . . .	22
3.5	Análise Espectral (FFT) . . . . .	23
3.6	Integração com a Modulação ASK . . . . .	26
3.7	Resultados Consolidados . . . . .	26
3.7.1	Análise Espectral da Modulação ASK . . . . .	27
3.7.2	Implementação Física (KLayout / Place-and-Route) . . . . .	28
<b>4</b>	<b>Conclusões</b>	<b>30</b>
4.1	Considerações Finais . . . . .	30
4.2	Propostas para Trabalhos Futuros . . . . .	31
	<b>Referências Bibliográficas</b>	<b>33</b>
<b>A</b>	<b>Códigos Verilog Desenvolvidos</b>	<b>36</b>
	sdm_1.v – Modulador Sigma-Delta de 1 <sup>a</sup> Ordem . . . . .	36
	tb_sdm.v – Testbench Funcional . . . . .	37
	top_sdm.v – Módulo Top Lógico . . . . .	39
	top_sdm_phys.v – Módulo Top para Síntese Física . . . . .	40
<b>B</b>	<b>Scripts Python para Simulação e Análise</b>	<b>42</b>
	ask_sim.py – Comparação Espectral com e sem Sigma-Delta . . . . .	42
	espectro_freq_bitstream.py – FFT do Bitstream Sigma-Delta . . . . .	45

# Lista de Figuras

3.1	Arquitetura simplificada do modulador sigma-delta. . . . .	16
3.2	Formas de onda registradas no GTKWave para entrada $DC = 0,75$ . . . . .	22
3.3	Formas de onda no GTKWave para entrada $DC \approx 0,305$ . . . . .	23
3.4	Espectro do bitstream do modulador para entrada $DC = 0,75$ . . . . .	24
3.5	Espectro do bitstream do modulador para entrada $DC \approx 0,305$ . . . . .	25
3.6	Diagrama funcional da integração entre o modulador sigma-delta e o transmissor ASK. . . . .	26
3.7	Comparação espectral entre ASK com SDM e PWM periódico para entrada $0,75$ . . . . .	27
3.8	Comparação espectral entre ASK com SDM e PWM periódico para entrada $0,305$ . . . . .	28
3.9	Layout físico do modulador sigma-delta sintetizado em tecnologia CMOS 130 nm, visualizado no KLayout. . . . .	29

# Capítulo 1

## Introdução

A evolução dos sistemas de comunicação sem fio tem impulsionado o desenvolvimento de técnicas de modulação e conversão de sinais cada vez mais eficientes e de alta fidelidade. Dentre essas, os moduladores sigma-delta destacam-se por permitir altas resoluções através de *sobreamostragem* e *conformação de ruído*, ou seja, espalhando o ruído de quantização para fora da banda de interesse [1,2]. Originalmente popularizados em conversores analógico-digital (ADCs) de precisão, os moduladores sigma-delta utilizam *oversampling* e realimentação para empurrar o ruído de quantização para frequências elevadas, proporcionando relações sinal-ruído (SNR) extremamente altas dentro da banda útil do sinal [1,2]. Por exemplo, Qian e Diao [3] demonstraram um modulador sigma-delta de 4<sup>a</sup> ordem em 180 nm CMOS capaz de atingir SNDR (*Signal-to-Noise and Distortion Ratio*) de 112 dB em 20 kHz de largura de banda, empregando um único bit de quantização a 5,12 MHz de taxa de amostragem, consumindo apenas 3,9 mW. Tais resultados ilustram por que moduladores sigma-delta são considerados a arquitetura de escolha para ADCs de áudio e outros sistemas que demandam alta resolução, trocando velocidade por precisão por meio da *sobreamostragem* [1,2].

Nos últimos anos, a aplicação de moduladores sigma-delta expandiu-se além de conversores AD tradicionais, passando a abarcar também transmissores e sistemas de radiofrequência. Pesquisas recentes exploram moduladores sigma-delta reconfiguráveis para rádios *multimode* (multibanda), como em receptores 5G, visando suportar diferentes larguras de banda com desempenho otimizado em cada modo [4]. Zaky, Omran e Elsayed [4] apresentaram um modulador contínuo do tipo

sigma-delta de baixa-passagem configurável que atende bandas de 10 a 80 MHz do padrão 5G, alcançando SNDR em torno de 7577 dB conforme o modo de operação. Em paralelo, outras linhas de investigação focam na eficiência energética e simplicidade de circuitos sigma-delta: Khoshkam et al. [5], por exemplo, propuseram substituir amplificadores operacionais por *current conveyors* (CCII) em um modulador sigma-delta de 2ª ordem, eliminando problemas de estabilidade e obtendo cerca de 60 dB de rejeição de ruído com consumo de apenas 0,61 mW em 90 nm CMOS. Esses avanços evidenciam que os moduladores sigma-delta permanecem tópicos de grande relevância na engenharia eletrônica, com aplicações que vão de conversores para dispositivos biomédicos de ultrabaixo consumo [5, 6] até rádios definidos por software (*software-defined radios*) e receptores multibanda [4, 7].

Neste contexto mais amplo, o presente trabalho aplica a tecnologia de modulação sigma-delta a um problema específico em transmissores digitais de *Amplitude Shift Keying* (ASK) em sua forma binária, também conhecida como Binary Amplitude Shift Keying (BASK): a redução de espúrios de *clock* no sinal transmitido. Em moduladores digitais de amplitude (como os empregados em sistemas DSRC *Dedicated Short Range Communication*), é comum que o sinal de saída apresente um tom espúrio na frequência do *clock* de referência da modulação, o qual pode interferir em canais adjacentes [8]. Kim e Lee [8] propuseram uma técnica inovadora para mitigar esse espúrio de referência usando um modulador sigma-delta acoplado a um oscilador: o sigma-delta introduz um *dithering* (*jitter* controlado) no *clock* de referência digital, espalhando a energia do espúrio em frequência e reduzindo sua amplitude pico. Implementado em tecnologia CMOS de 130 nm, o método demonstrou redução de aproximadamente 5 dBc no nível do espúrio de *clock* na saída de um transmissor ASK para aplicações DSRC, ao *ditherizar* a referência de 160 MHz por meio de um modulador sigma-delta de terceira ordem (estrutura MASH 1-1-1) acoplado a um oscilador controlado por corrente e capacitor [8]. Esse resultado comprova a viabilidade de se empregar modulação sigma-delta no domínio digital para aprimorar a pureza espectral de transmissores, sem necessidade de filtros analógicos complexos.

Tendo como base essa abordagem, este trabalho de conclusão de curso explora o projeto e análise de um modulador sigma-delta digital de 1ª ordem visando à

redução de espúrios em modulação ASK. O modulador foi descrito em Verilog e sintetizado em tecnologia CMOS 130 nm (usando o kit de processo aberto do IHP), integrando-se a um sistema transmissor ASK para realizar o *clock dithering*. O funcionamento foi validado por meio de simulações e análises no domínio do tempo e da frequência (FFT), utilizando ferramentas como GTKWave para inspecionar o *bitstream* gerado e seu espectro. Os resultados obtidos comprovam a eficácia da técnica na mitigação de espúrios, contribuindo para designs de transmissores digitais mais limpos espectralmente. A seguir, são detalhados o problema abordado, a justificativa da pesquisa, os objetivos propostos e, no Capítulo 2, uma revisão da literatura relevante ao tema.

## 1.1 Problema

O problema central abordado neste trabalho é a presença de espúrios de clock no sinal de saída de moduladores digitais de ASK e a dificuldade de suprimi-los pelos meios tradicionais. Em moduladores de amplitude chaveada, especialmente em transmissões digitais de curto alcance (como DSRC, utilizadas em comunicação veicular), o sinal transmitido costuma conter tons indesejados em frequências determinísticas relacionadas ao clock de referência [8]. Esses espúrios originam-se da periodicidade do chaveamento digital: quando a modulação opera de forma rígida e periódica (como em PWM tradicional com duty-cycle fixo), as transições de amplitude ocorrem sempre nos mesmos instantes de tempo, criando uma sequência repetitiva [8]. Essa repetitividade se manifesta no domínio da frequência como linhas espectrais discretas (picos) não apenas na frequência do clock, mas também em seus harmônicos e em frequências relacionadas ao padrão de chaveamento [8]. O resultado é um conjunto de tons de interferência no espectro transmitido, especialmente distribuídos de forma determinística, que podem afetar canais vizinhos e violar máscaras de espectro definidas por órgãos reguladores.

No cenário de DSRC a 5,8 GHz, por exemplo, o modulador digital é tipicamente controlado por um clock de referência em torno de 160 MHz [8]. Sem contramedidas, um espúrio aparece nessa frequência e pode causar interferência em receptores próximos ou degradar a relação sinal-interferência do próprio sistema.

Tradicionalmente, a mitigação de espúrios de clock requer filtros de saída de alta ordem ou técnicas analógicas de espalhamento espectral. Contudo, filtros podem elevar custo e complexidade, enquanto métodos analógicos (como jitter intencional no oscilador) nem sempre são facilmente controláveis ou integráveis.

A pergunta de pesquisa que se impõe é: como reduzir o spur de clock em transmissores ASK utilizando recursos digitais integrados ao próprio modulador? A hipótese considerada é que a inserção de um modulador sigma-delta digital no laço de referência do transmissor pode ditherizar (aleatorizar) o instante de comutação do sinal, espalhando a energia do espúrio em frequência e diminuindo seu pico [8]. Em outras palavras, em vez de um tom puro concentrado em 160 MHz, teria-se um ruído de fase distribuído em torno dessa frequência, menos agressivo ao espectro. Implementar tal abordagem exige enfrentar desafios de projeto, como escolher a ordem e parâmetros do modulador sigma-delta adequados (considerando estabilidade e quantização), e garantir que a solução funcione nas velocidades requeridas em 130 nm CMOS.

Em síntese, o problema engloba tanto a natureza espectral dos espúrios em moduladores ASK quanto o desafio de se projetar um sistema digital avançado (o sigma-delta) para atenuar esses espúrios sem prejudicar a modulação desejada. Resolver esse problema contribui para transmissores mais limpos, atendendo a requisitos de compatibilidade eletromagnética e desempenho espectral em aplicações de comunicação sem fio de curto alcance.

## 1.2 Justificativa

A motivação para este trabalho é dupla: prática e acadêmica. Do ponto de vista prático/industrial, a redução de espúrios em transmissores de RF é crucial para cumprir normas espectrais e minimizar interferências. Sistemas como o DSRC usados em aplicações veiculares e de tráfego operam em bandas congestionadas, onde um spur fora da banda pode degradar outros serviços ou reduzir a confiabilidade da comunicação [8]. A técnica de clock dithering via modulador sigma-delta oferece uma solução integrada e de baixo custo para esse problema, dispensando filtros externos onerosos e permitindo conformidade espectral através de design di-

gital inteligente [8]. Em vez de adicionar hardware analógico complexo, aproveita-se a flexibilidade do domínio digital (firmware/HDL) para melhorar o sinal transmitido. Essa abordagem alinhada ao conceito de radio definido por software torna os transmissores mais adaptáveis e integráveis em tecnologia CMOS avançada.

Sob o ângulo acadêmico e científico, o trabalho se justifica por explorar uma aplicação não convencional dos moduladores sigma-delta. Embora vastamente estudados em conversores de dados [3, 5, 6], seu uso em modulação de transmissão ainda é pouco difundido, restringindo-se a casos recentes como o de [8]. Assim, o projeto aqui descrito expande o conhecimento sobre moduladores sigma-delta em aplicações de RF digital, avaliando seus benefícios e limitações no contexto de modulação ASK. Além disso, o estudo envolve diversas disciplinas da Engenharia da Computação: design digital em HDL (Verilog), síntese e validação em tecnologia CMOS real (130 nm) e processamento de sinais (análise espectral de ruído e espúrios). Essa natureza multidisciplinar proporciona ao autor uma experiência abrangente em projeto de circuitos e sistemas digitais de alta performance, consolidando conceitos teóricos na prática.

## **1.3 Objetivos**

### **1.3.1 Objetivos Gerais**

Desenvolver e validar um modulador sigma-delta digital de primeira ordem, implementado em Verilog e sintetizado em tecnologia CMOS 130 nm, com aplicação no controle direto do chaveamento de uma modulação ASK. O objetivo é reduzir espúrios de clock no sinal modulado por meio da aplicação do bitstream gerado pelo modulador sigma-delta como sinal de dithering.

### **1.3.2 Objetivos Específicos**

Para atingir o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

- Estudar os fundamentos e trabalhos relacionados sobre moduladores sigma-delta digitais e suas aplicações [3, 4, 8], com foco especial em técnicas de con-

formação de ruído para redução de espúrios em sinais modulados, realizando um levantamento bibliográfico abrangente,.

- Projetar a arquitetura de um modulador sigma-delta de 1ª ordem, definindo seus parâmetros (frequência de amostragem, *oversampling ratio*, número de bits, estrutura de filtros integradores etc.) de forma compatível com o sistema ASK alvo. Optou-se por uma topologia simples de primeira ordem visando facilidade de implementação e estabilidade garantida, dado que moduladores de ordem superior podem apresentar instabilidades sem filtros apropriados [1, 2].
- Implementar o modulador em Verilog, em nível de transferência entre registradores (RTL), utilizando técnicas de projeto síncrono e garantindo que o circuito possa operar na frequência desejada. Inclui-se a criação de um *test-bench* para verificar a funcionalidade do modulador sigma-delta isoladamente, gerando o *bitstream* de saída e avaliando propriedades como o padrão espectral do ruído de quantização.
- Sintetizar o projeto em tecnologia CMOS de 130 nm, por meio do fluxo de síntese lógica e implementação física (*Place & Route*) suportado pelo Open PDK da IHP. Nesta etapa, verificar o cumprimento de restrições de temporização (*timing*) na frequência de operação, bem como estimar a área de silício e o consumo de potência do modulador.
- Integrar o modulador sigma-delta ao sistema de modulação ASK, conectando a saída do modulador ao controle do circuito gerador de portadora ASK, conforme a arquitetura escolhida. Deve-se garantir que, com o modulador sigma-delta ativado, o funcionamento básico do transmissor ASK (ligar/desligar a portadora conforme os dados) permaneça correto.
- Testar e analisar o desempenho do sistema integrado, comparando o espectro do sinal ASK transmitido com e sem o *dithering* sigma-delta. Utilizar ferramentas de análise no domínio do tempo (como o GTKWave) e no domínio da frequência (FFT via scripts ou softwares dedicados) para medir a redução do *spur* de clock em dBc alcançada. Avaliar também o impacto, se houver, no

sinal de dados ASK, verificando se a constelação ou a forma de onda modulada sofrem degradação perceptível devido ao *dithering*.

- Documentar os resultados e concluir sobre a eficácia da solução, discutindo possíveis otimizações (como o uso de moduladores de ordem superior, diferentes frequências de amostragem etc.) e potenciais trabalhos futuros.

A organização da monografia é definida da seguinte forma: o Capítulo 2 apresenta uma revisão dos fundamentos teóricos sobre moduladores sigma-delta e suas aplicações em sistemas de comunicação digital. O Capítulo 3 descreve o projeto e implementação em Verilog, a síntese física em tecnologia CMOS 130 nm, a validação funcional via GTKWave, a análise espectral por FFT e os resultados comparativos entre ASK com e sem dithering sigma-delta. O Capítulo 4 sintetiza as conclusões e propõe trabalhos futuros.

# Capítulo 2

## Revisão da Literatura

Este capítulo apresenta uma revisão dos principais conceitos e trabalhos relacionados a moduladores sigma-delta e à sua aplicação em sistemas de comunicação digital. Inicialmente, discutem-se os fundamentos dos moduladores sigma-delta e seu papel em conversores de alta resolução. Em seguida, abordam-se aplicações relevantes em síntese de frequência e redução de espúrios, com ênfase em técnicas de dithering de portadora em transmissores ASK. Por fim, posiciona-se o presente trabalho em relação à literatura existente.

### 2.1 Fundamentos dos moduladores sigma-delta

Um modulador sigma-delta (ou sigma-delta) é, em essência, um sistema de sobreamostragem com realimentação que converte um sinal de entrada (analógico ou digital de maior resolução) em um fluxo de saída de menor resolução, frequentemente de apenas 1 bit [1,2]. A principal característica dessa arquitetura é a capacidade de *conformar* o ruído de quantização, empurrando-o para frequências mais altas, fora da banda de interesse, por meio de uma função de transferência de ruído (*Noise Transfer Function* – NTF) de natureza passa-altas.

Em um modulador de primeira ordem típico, a malha é composta por um integrador, um quantizador de 1 bit e um conversor DAC de realimentação também de 1 bit [1]. A função de transferência de sinal (*Signal Transfer Function* – STF) ideal aproxima um comportamento passa-baixas com ganho unitário na banda, enquanto a NTF tem comportamento passa-altas, atenuando o ruído na banda base e

amplificando-o fora dela [2]. No domínio  $z$ , para um modulador de primeira ordem com STF ideal, tem-se aproximadamente  $\text{STF}(z) \approx z^{-1}$ , isto é, o sinal de entrada é essencialmente atrasado de um período de amostragem na saída.

### 2.1.1 Sobreamostragem e equação de SNR

Uma consequência direta da combinação de sobreamostragem e conformação de ruído é o aumento do SNR efetivo na banda de interesse à medida que se eleva a razão de sobreamostragem (*Oversampling Ratio* – OSR) [1, 2]. Para um conversor com quantização uniforme de  $N$  bits, sem conformação de ruído, o SNR teórico aproximado é dado por

$$\text{SNR} \approx 6,02 N + 1,76 \text{ (dB)}. \quad (2.1)$$

No caso de um modulador sigma-delta de primeira ordem ideal, a sobreamostragem aliada à NTF passa-altas melhora esse valor [2]. Uma expressão frequentemente utilizada para o SNR teórico em função do OSR é

$$\text{SNR} \approx 6,02 N + 1,76 + 30 \log_{10}(\text{OSR}) - 10 \log_{10}\left(\frac{\pi^2}{3}\right), \quad (2.2)$$

onde  $N$  é o número de bits do quantizador (no caso binário,  $N = 1$ ) e OSR é definido como a razão entre a frequência de amostragem e duas vezes a largura de banda do sinal. A parcela  $30 \log_{10}(\text{OSR})$  evidencia que o aumento da frequência de amostragem, mantendo-se a mesma banda útil, resulta em ganho significativo de SNR devido à redistribuição espectral do ruído de quantização [1, 2].

De forma aproximada, para um modulador de primeira ordem, a cada duplicação do OSR obtém-se uma melhoria de cerca de 9 dB no SNR de banda. Para ordens mais elevadas, essa inclinação aumenta, podendo atingir aproximadamente 15 dB por duplicação do OSR em moduladores de segunda ordem e 21 dB em moduladores de terceira ordem, sob condições ideais de projeto [1].

## 2.1.2 Ordem do modulador, número de bits e compromisso de projeto

O desempenho de um modulador sigma-delta é determinado por quatro parâmetros principais: a ordem da NTF, o OSR, o número de bits do quantizador e as limitações práticas de circuito (ganho finito, ruído térmico, jitter de relógio, entre outros) [1,2]. Elevar a ordem do modulador por meio da inserção de múltiplos integradores e caminhos de realimentação permite aumentar a inclinação de conformação de ruído, reduzindo ainda mais a densidade espectral de ruído na banda útil [1].

Entretanto, moduladores de alta ordem estão sujeitos a riscos de instabilidade e ao surgimento de tons de quantização, exigindo técnicas específicas de projeto, como a utilização de estruturas MASH (*Multi-Stage Noise Shaping*), realimentação cuidadosamente ponderada e, em alguns casos, dithering explícito [1,8]. Além disso, ordens superiores implicam maior complexidade de implementação, seja em circuitos analógicos (mais amplificadores operacionais, comparadores e capacitores de amostragem), seja em lógica digital (mais registradores, somadores e caminhos de dados), o que impacta consumo de potência e área [3,5].

Uma alternativa para aumentar o SNR é empregar quantização multi-bit. Em teoria, cada bit adicional de resolução do quantizador adiciona aproximadamente 6 dB ao SNR [1,2]. No entanto, moduladores multi-bit exigem DACs de realimentação altamente lineares, sob pena de introduzir distorções significativas. Por essa razão, muitas arquiteturas preferem manter quantização de 1 bit, explorando principalmente o OSR e a ordem do modulador, o que garante linearidade intrínseca do DAC de realimentação [1,3].

Em síntese, há um compromisso entre ordem do modulador, OSR, número de bits e complexidade de implementação. A escolha do ponto de operação adequado depende dos requisitos específicos de banda, resolução, consumo e tecnologia de fabricação [2,4].

## 2.2 Arquiteturas digitais para dithering de portadora

No contexto de aplicações digitais de alta frequência, duas famílias de arquiteturas sigma-delta se destacam:

- **Moduladores de primeira ordem:** apresentam estrutura extremamente simples, requerendo apenas um acumulador, lógica de soma/subtração e um comparador implícito (bit mais significativo). São estáveis por construção, têm latência mínima e podem operar em frequências elevadas, sendo particularmente adequados para controle direto de osciladores ou divisores [1, 8].
- **Arquiteturas MASH:** consistem em múltiplos moduladores de primeira ordem encadeados, cujas saídas são combinadas de forma a obter uma NTF de ordem mais alta [1, 8]. São amplamente utilizadas em PLLs fracionárias, pois permitem elevada conformação de ruído mantendo cada estágio internamente estável [8].

Para projetos didáticos, protótipos em tecnologias CMOS maduras ou aplicações em que a redução de espúrios não precisa ser extrema, moduladores de primeira ordem costumam ser suficientes para demonstrar o efeito desejado de espalhamento espectral, com grandes vantagens em simplicidade, área e consumo [1, 8].

## 2.3 Síntese Digital em Verilog

A *síntese digital* é o processo de transformar uma descrição em alto nível de um circuito digital (geralmente em HDL, *Hardware Description Language*) em uma implementação de baixo nível composta por portas lógicas e *flip-flops* de uma determinada tecnologia [1]. No presente trabalho, o modulador sigma-delta foi descrito em Verilog RTL, que é uma das linguagens de descrição de hardware mais difundidas. O Verilog permite modelar registradores, lógica combinacional e sequencial de forma relativamente abstrata (próxima ao comportamento), e ferramentas de síntese automatizada irão mapear essa descrição em circuitos usando portas lógicas da biblioteca de células-padrão do processo de 130 nm escolhido.

No fluxo de projeto utilizado, fez-se uso de ferramentas *open-source* para síntese e *place-and-route* (P&R) integradas em um *flow* recente denominado *LibreLane*. Esse *flow* é sucessor do OpenLane e combina diversos softwares (Yosys para síntese lógica, OpenROAD para *floorplanning*, roteamento e otimização, OpenSTA para análise temporal estática, entre outros) para realizar automaticamente o processo *RTL-to-GDSII*. A grande vantagem de um *flow* assim é poder especificar todo o projeto em um único arquivo de configuração JSON ou Tcl e obter, ao final, um *layout* físico completo pronto para fabricação.

No caso do modulador sigma-delta deste trabalho, a síntese se iniciou em Yosys lendo o código Verilog e mapeando cada operador e registrador em células lógicas do PDK IHP 130 nm (portas AND, OR, *flip-flops* D, etc.). Por exemplo, o registro acumulador de 19 bits seria implementado como 19 *flip-flops* D do kit, e as somas como redes de somadores (*half-adders*, *full-adders*) também disponíveis na biblioteca padrão.

Vale ressaltar que, para a síntese ser bem-sucedida, o código Verilog deve ser *synthesizável*, ou seja, deve evitar construções não suportadas (como *delays* de tempo de simulação #, inicializações não síncronas arbitrárias, etc.). O código desenvolvido atendeu a esse requisito, pois foi escrito em estilo RTL síncrono. Ferramentas como o Yosys geram, a partir disso, um *netlist gate-level*. Em seguida, etapas de *Place and Route* (PnR) posicionam as células e roteiam as interconexões no chip respeitando as regras de desenho do processo de 130 nm.

Após a síntese lógica, o *flow* realiza *floorplanning* alocando área de silício para o bloco, insere pinos conforme especificado (foi fornecido um arquivo de ordem de pinos para fixar posições), efetua *placement* colocando as células na área interna, roteia as interconexões (*global routing* seguido de *detailed routing*) e, por fim, gera o *layout* final (GDSII). Todo esse processo considera o PDK para garantir que dimensões mínimas, espaçamentos e demais regras de manufatura sejam atendidas.

No caso do PDK IHP, as bibliotecas de células padrão e regras de desenho (DRC, LVS) foram integradas via repositório público IHP-Open-PDK.

## 2.4 Fluxo de Projeto em CMOS 130 nm com PDKs Digitais

O desenvolvimento de circuitos digitais em tecnologias CMOS modernas depende do uso de *Process Design Kits* (PDKs), que reúnem modelos elétricos, regras de desenho (DRC/LVS), bibliotecas de células padrão e arquivos tecnológicos necessários para implementação física [7]. Em processos amplamente documentados, como o CMOS de 130 nm, esses PDKs permitem que ferramentas de síntese e *place-and-route* executem o fluxo digital completo de forma automatizada [7].

De maneira geral, o fluxo de projeto segue etapas consolidadas na literatura: (i) descrição RTL em Verilog; (ii) síntese lógica, na qual a descrição comportamental é convertida em um *gate-level netlist*; (iii) análise temporal estática para verificação de restrições de tempo; (iv) implementação física, incluindo *floorplanning*, colocação e roteamento; e (v) verificação física, assegurando conformidade com as regras de manufatura do processo. Ferramentas como Yosys (síntese lógica) e OpenROAD (implementação física) são amplamente utilizadas em fluxos digitais modernos devido ao seu caráter aberto e reproduzível [7].

Ao final do processo, o layout físico é exportado em formato GDSII e pode ser inspecionado em visualizadores como KLayout ou Magic, possibilitando conferência geométrica e revisão de conectividade antes da fabricação [7].

## 2.5 GTKWave e Análise Espectral via FFT

A verificação de circuitos digitais envolve tanto a análise temporal quanto a análise espectral, conforme amplamente discutido na literatura de sistemas digitais [2]. A simulação temporal é realizada por meio de testbenches em Verilog ou VHDL, executados em simuladores como Icarus Verilog ou Verilator, que geram arquivos do tipo *Value Change Dump* (VCD). Esses arquivos permitem examinar a evolução dos sinais ao longo do tempo em ferramentas gráficas como o GTKWave, muito utilizada em ambientes acadêmicos e de pesquisa.

Além da análise temporal, diversos sistemas digitais exigem caracterização no domínio da frequência, especialmente moduladores e estruturas que manipulam

sinais 1-bit [2, 3]. Para isso, é comum exportar sequências de dados geradas em simulação e aplicar a Transformada Rápida de Fourier (FFT) utilizando ferramentas como Python (NumPy) ou MATLAB. Esse procedimento possibilita identificar tons espúrios, distribuição de ruído e padrões periódicos associados ao comportamento do sistema, aspectos particularmente relevantes em moduladores sigma-delta e sistemas de comunicação digital [2, 3].

# Capítulo 3

## Desenvolvimento

### 3.1 Visão Geral do Sistema

Este capítulo descreve o desenvolvimento de um modulador sigma-delta digital de 1ª ordem aplicado à modulação de amplitude (ASK) com o objetivo de reduzir tons espúrios [8]. O sistema proposto insere um modulador sigma-delta no caminho de controle de amplitude de um transmissor ASK. Em vez de controlar a amplitude de forma direta (analogicamente ou por modulação PWM determinística), utiliza-se um fluxo de bits (bitstream) de alta velocidade cuja densidade de 1 corresponde à amplitude desejada [1, 2, 8]. Conforme discutido por Kim e Lee [8], o modulador sigma-delta pode ser empregado no controle de osciladores para espalhar espúrios de referência através da técnica de *clock dithering*. Esse fluxo 1-bit, quando empregado para chavear um oscilador ou amplificador de RF, resulta em uma modulação de amplitude cujo valor médio corresponde ao nível pretendido, porém com o erro de quantização espalhado em frequência como ruído de alta frequência [8]. Assim, os componentes espúrios discretos (tons indesejados) tendem a ser convertidos em ruído distribuído espectralmente, mais fácil de filtrar ou menos prejudicial dentro da banda de interesse do que picos espúrios concentrados [8].

A implementação prática consistiu em projetar o modulador sigma-delta em nível RTL (Verilog), simular seu comportamento para verificar a funcionalidade e analisar o espectro do bitstream gerado. Em seguida, integrou-se o design a um fluxo de síntese física em tecnologia CMOS 130 nm (PDK da IHP) utilizando a infraestrutura open-source LibreLane/OpenLane [9], obtendo-se um layout do cir-

cuito integrado. Embora o projeto não inclua a realização de um front-end analógico completo (por exemplo, oscilador ou DAC físico), foi feita uma modelagem em nível de sistema para validar como o modulador sigma-delta interagiria com a modulação ASK. Em resumo, este capítulo cobre desde a arquitetura do modulador e sua implementação em Verilog, até a geração do bitstream e sua análise temporal e espectral, finalizando com a comparação de resultados com e sem o uso do sigma-delta para evidenciar a redução de espúrios [8]. A seguir, apresenta-se inicialmente a arquitetura do modulador sigma-delta desenvolvido.

## 3.2 Arquitetura do Modulador Sigma-Delta

A arquitetura adotada para o modulador sigma-delta de primeira ordem é baseada no esquema clássico de um integrador + quantizador de 1 bit com realimentação negativa [1, 2], conforme mostra a Figura 3.1

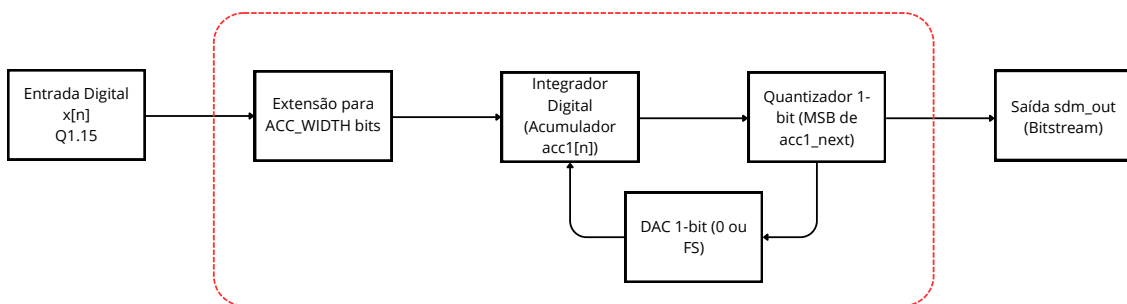


Figura 3.1: Arquitetura simplificada do modulador sigma-delta.

Conforme mostrado, a entrada digital representa um valor de amplitude normalizado (em ponto fixo fracionário), que é alimentado a um integrador acumulativo. O integrador soma, a cada ciclo de clock, o valor de entrada e também incorpora a realimentação do bit de saída anterior. Essa realimentação é feita por meio de um DAC de 1 bit (no domínio digital, equivalendo a subtrair um valor de referência fixo quando o bit de saída anterior é 1) [2]. Em outras palavras, o modulador calcula a cada ciclo:

- Integrador: a atualização do acumulador é dada por  $acc1\_next = acc1 + in\_ext - dac1$ , onde  $in\_ext$  é o valor de entrada escalado ao tamanho do acumulador e  $FS$  é o valor de *full-scale* (escala cheia) representado no acumulador, correspondente a 1.0 na entrada [2].

- Quantizador de 1 bit: o novo bit de saída *out\_bit* é obtido tomando-se o *most significant bit* (MSB) do valor *acc1\_next*. Esse MSB funciona como um comparador: se o acumulador atinge ou ultrapassa a metade da faixa ( $\text{acc1\_next} \geq FS$ ), o bit de saída torna-se 1; caso contrário, permanece 0.

No modulador implementado, a entrada digital de amplitude é representada em formato Q1.15 (16 bits fracionários, intervalo  $[0, 1)$  em valor real). O acumulador do integrador foi projetado com largura de 19 bits (no código,  $\text{ACC\_WIDTH} = 16 + 3$ ), proporcionando 3 bits extras de guarda além da resolução da entrada. Esses bits extras evitam *overflow* prematuro e reduzem efeitos de ciclo-limite, garantindo que o erro de quantização seja adequadamente acumulado antes do *wrap-around* [2]. O valor de *full-scale* para o DAC de 1 bit (realimentação) foi definido como  $2^{18}$  (pois com 19 bits,  $FS = 2^{(\text{ACC\_WIDTH}-1)}$ ), equivalente a 1,0 em amplitude acumulada. Assim, quando o bit de saída é 1, subtrai-se FS do acumulador (representando a retirada de uma quantidade de carga equivalente à amplitude máxima); quando o bit é 0, nada é subtraído (o integrador apenas acumula a entrada).

Essa topologia realiza a modulação sigma-delta de 1ª ordem, na qual o erro de quantização (a diferença entre o nível analógico desejado e o nível quantizado de 1 bit) é realimentado e integrado [1,2]. Como resultado, o espectro do erro é modelado como ruído predominantemente em frequências altas, ao invés de baixa frequência. Em termos qualitativos, há um *shaping* de ruído: as componentes de erro/ruído em baixas frequências são atenuadas pelo *loop*, enquanto o ruído se concentra nas altas frequências (próximas à frequência de amostragem do modulador) [2]. O bit de saída alternará de forma a manter a média do sinal próxima ao valor de entrada, garantindo que, ao longo do tempo, a fração de '1's seja proporcional à amplitude solicitada. Em essência, o modulador sigma-delta funciona como um conversor D/A de 1 bit *oversampled*: a informação de amplitude é codificada na densidade temporal de pulsos (também chamada de modulação por densidade de pulso), em vez de no valor instantâneo de um multi-bit DAC [1,2].

É importante frisar que, por se tratar de um modulador de 1ª ordem, a atenuação do ruído em *baseband* cresce a uma taxa de aproximadamente 20 dB por década de frequência (característica típica de *noise shaping* de primeira ordem) [2]. Embora ordens superiores pudessem obter melhoria mais acentuada do *noise floor*

em baixa frequência, o projeto optou pela arquitetura de 1ª ordem devido à sua simplicidade de implementação e estabilidade garantida [2]. Com a arquitetura definida conforme acima, parte-se agora para a descrição da implementação em Verilog do modulador sigma-delta e dos componentes auxiliares desenvolvidos.

### 3.3 Implementação em Verilog

Com base na arquitetura descrita, foi desenvolvido o código RTL em Verilog do modulador sigma-delta e módulos de suporte [10]. A implementação abrange quatro componentes principais: o módulo do modulador (`sdm_1.v`), o testbench para simulação funcional (`tb_sdm.v`), e dois módulos de top (`top_sdm` e `top_sdm_phys`) preparados para síntese física segundo o fluxo LibreLane/OpenLane e o PDK CMOS 130 nm [11].

#### 3.3.1 Módulo Principal: `sdm_1.v`

Este módulo corresponde ao modulador sigma-delta de 1ª ordem [1, 2]. A entrada digital em formato Q1.15 é integrada ciclicamente, enquanto a realimentação binária de 1 bit controla a retirada de carga do acumulador. A largura interna do acumulador foi definida como `ACC_WIDTH = IN_WIDTH + 3`, resultando em 19 bits [2].

A lógica combinacional do sigma-delta é implementada com os sinais `acc1_next` e `out_bit_next`. O cálculo do próximo valor do acumulador segue:

$$\text{acc1\_next} = \text{acc1} + \text{in\_ext} - \text{dac1} \quad (3.1)$$

onde `dac1` é definido como:

$$\text{dac1} = \text{out\_bit} \times \text{FS} \quad (3.2)$$

e o novo bit de saída é definido como o bit mais significativo de `acc1_next`.

O trecho equivalente no Verilog é mostrado abaixo.

```

1 wire [ACC_WIDTH-1:0] acc1_next =
2   out_bit ? (acc1 + in_ext - FS) : (acc1 + in_ext);

```

```

3
4 wire out_bit_next = acc1_next[ACC_WIDTH-1];

```

O bloco sequencial sensível à borda de subida do clock atualiza os registradores, gerando assim o bitstream sigma-delta final [10].

### 3.3.2 Testbench: `tb_sdm.v`

O testbench implementa a lógica de introdução de estímulos e captura da saída [10]. Um valor DC em Q1.15 é aplicado à entrada (Por exemplo, `in_val = 16'd24576` representando 0,75 em amplitude real). Durante a simulação, o bitstream resultante é registrado com chamadas `$fwrite` em um arquivo texto nomeado `sdm_out.txt`.

```

1 initial begin
2     // Dump para GTKWave
3     $dumpfile("tb_sdm.vcd");
4     $dumpvars(0, tb_sdm);
5
6     clk    = 1'b0;
7     rst_n  = 1'b0;
8
9     // Entrada DC em Q1.15 (~0.75)
10    in_val = 16'd24576;
11
12    // Arquivo texto para FFT
13    f = $fopen("sdm_out.txt", "w");
14
15    // Reset inicial
16    #50 rst_n = 1'b1;
17
18    // Loop de simulação: gera 50000 amostras
19    for (i = 0; i < 50000; i = i + 1) begin
20        @(posedge clk);
21        $fwrite(f, "%0d\n", out_bit);

```

```

22     end
23
24     $fclose(f);
25     $finish;
26 end

```

Esse bitstream será posteriormente analisado em Python para FFT e inspeção espectral do modulador.

### 3.3.3 Módulo Top Lógico: `top_sdm`

Serve como wrapper para instanciar o modulador `sdm_1`, conectar sinais e preparar interfaces internas. Esse módulo é usado para simulação, geração de VCD e testes funcionais [10].

```

1 sdm_1 #(
2     .IN_WIDTH (IN_WIDTH),
3     .ACC_WIDTH(ACC_WIDTH)
4 ) u_mash (
5     .clk      (clk),
6     .rst_n    (rst_n),
7     .in_val   (in_val),
8     .out_bit(sdm_out)
9 );

```

### 3.3.4 Módulo Top para Síntese Física: `top_sdm_phys`

Este módulo adapta as entradas vetoriais para pinos escalares exigidos por alguns flows de síntese física [9]. Ele reconstrói internamente o barramento de 16 bits e instancia o módulo `top_sdm` original [10].

```

1 wire [15:0] in_val;
2
3 assign in_val = {
4     in_val_15,
5     in_val_14,

```

```

6   in_val_13,
7   in_val_12,
8   in_val_11,
9   in_val_10,
10  in_val_9,
11  in_val_8,
12  in_val_7,
13  in_val_6,
14  in_val_5,
15  in_val_4,
16  in_val_3,
17  in_val_2,
18  in_val_1,
19  in_val_0
20 };
21
22 top_sdm #(
23     .IN_WIDTH (16),
24     .ACC_WIDTH(16+3)
25 ) u_top (
26     .clk      (clk),
27     .rst_n    (rst_n),
28     .in_val   (in_val),
29     .sdm_out  (sdm_out)
30 );

```

### 3.3.5 Síntese Física e Integração

Utilizando `config.json` e `pin_order.cfg`, o fluxo LibreLane posicionou células, realizou roteamento e gerou o GDSII final. Validou-se o layout com o KLayout e verificou-se a ausência de violações DRC [9]. A integração com o PDK CMOS 130 nm [11] garantiu compatibilidade com o processo de fabricação

Os arquivos gerados foram utilizados posteriormente nas simulações e análises espectrais do sistema ASK.

### 3.4 Simulação Funcional (GTKWave)

A simulação funcional tem como objetivo verificar o comportamento temporal do circuito RTL antes da implementação física, assegurando que o projeto do modulador sigma-delta opere conforme o esperado. Para isso, utilizou-se o **GTKWave**, ferramenta de visualização de formas de onda de simulação (arquivos VCD), que permite inspecionar os sinais do testbench `tb_sdm` em cada ciclo de clock [10].

No testbench, aplica-se uma entrada DC constante (`in_val` em formato Q1.15) ao módulo `sdm_1` e registram-se vários sinais: o clock (`clk`), o reset síncrono ativo baixo (`rst_n`), a entrada `in_val`, o acumulador interno (`acc1`), o próximo valor do acumulador (`acc1_next`) e o bit de saída (`out_bit`) [10]. Conforme descrito no código-fonte do modulador, o sinal `in_val` em Q1.15 corresponde a um valor real aproximado de  $\text{in\_val}/2^{15}$  e o bit de saída é um sinal binário cuja média tende ao valor real de entrada [2]. Após liberar o reset inicial, inicia-se a geração do sinal de clock (período de 10 ns) e observa-se a evolução desses sinais no GTKWave.

No gráfico das formas de onda, o sinal `clk` exibe pulsos periódicos de 100 MHz, enquanto `rst_n` permanece em 0 nos instantes iniciais até ser ativado (1) após 50 ns, liberando o modulador. A seguir, `in_val` permanece constante no valor testado (Q1.15), e os sinais internos do modulador evoluem a cada flanco de subida do clock [2]. O acumulador `acc1` mostra seu valor atual em cada ciclo e o sinal `acc1_next` indica o próximo valor antes do registro. Quando `out_bit` é 1, ocorre realimentação negativa (subtrai-se full-scale FS); caso contrário, apenas acumula-se o valor estendido da entrada. Assim, o MSB de `acc1_next` define o próximo `out_bit`. Conforme a teoria, a densidade de 1s em `out_bit` reflete o valor médio de `in_val`, permitindo inferir corretamente seu funcionamento [1,2].

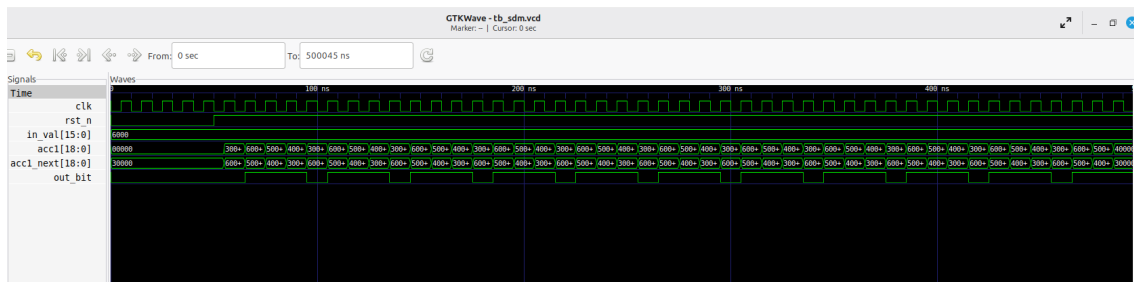


Figura 3.2: Formas de onda registradas no GTKWave para entrada DC = 0,75.

Como mostrado na Figura 3.2, observam-se o clock (`clk`), o reset ativo baixo (`rst_n`), a entrada constante (`in_val`), o acumulador atual (`acc1`), o próximo acumulador (`acc1_next`) e o bit de saída (`out_bit`). A saída `out_bit` exibe um padrão de pulsos muito denso em 1s, com zeros esporádicos quase periódicos, conforme esperado para um valor alto de entrada. O acumulador `acc1` cresce de forma quase linear (módulo- $2^{\text{ACC\_WIDTH}}$ ) até o instante em que `out_bit` muda, refletindo o mecanismo de realimentação do modulador sigma-delta. Esse padrão consistente confirma que o circuito está respondendo corretamente à entrada de 0,75, gerando uma média de saída próxima a esse valor.

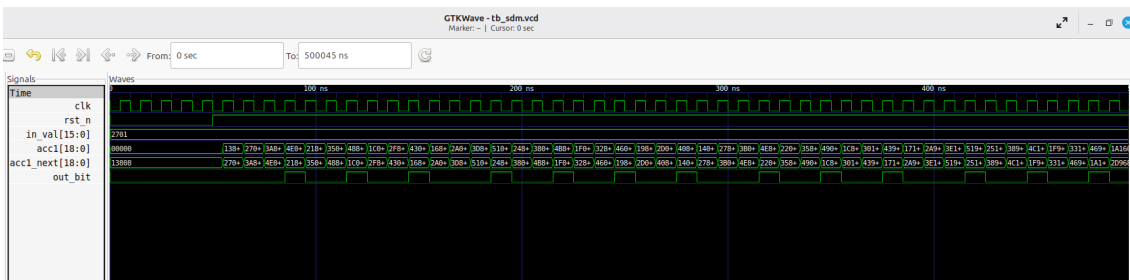


Figura 3.3: Formas de onda no GTKWave para entrada DC  $\approx 0,305$ .

Já na Figura 3.3, os sinais de clock, reset e entrada permanecem similares, mas a saída `out_bit` agora apresenta um padrão irregular, sem periodicidade aparente. Isso ocorre porque, para valores mais baixos de entrada, o modulador sigma-delta introduz *dithering* (eterna variação do erro de quantização), resultando numa sequência quase aleatória de 0s e 1s cujo valor médio representa a entrada. Nesse cenário, o acumulador oscila de modo aparentemente caótico, mas ainda satisfaz a relação média do bit de saída com a entrada.

Em ambas as simulações, as formas de onda verificam que o RTL do modulador está funcionalmente correto de acordo com a análise inicial [1, 10].

### 3.5 Análise Espectral (FFT)

A análise espectral do bitstream de saída do modulador sigma-delta de 1ª ordem visa avaliar como a energia do sinal, incluindo o ruído de quantização, se distribui ao longo do espectro de frequências. Em moduladores sigma-delta, o ruído de quantização é modelado (*noise shaping*) e deslocado para frequências mais altas,

preservando a qualidade do sinal nas baixas frequências, onde geralmente está a informação de interesse [1,2].

Para evidenciar esse efeito, aplicou-se a Transformada Rápida de Fourier (FFT) sobre os dados do arquivo `sdm_out.txt`, contendo 50.000 amostras do bitstream. Para reduzir os efeitos de *spectral leakage* causados por descontinuidades nas bordas da janela de análise, a média do bitstream foi subtraída e, em seguida, aplicou-se uma janela de Hann (*Hanning window*) antes do cálculo da FFT [2]. Esse processamento foi realizado por um script Python (`espectro_freq_bitstream.py`), utilizando as bibliotecas NumPy e Matplotlib, e os resultados foram apresentados em escala logarítmica (dB).

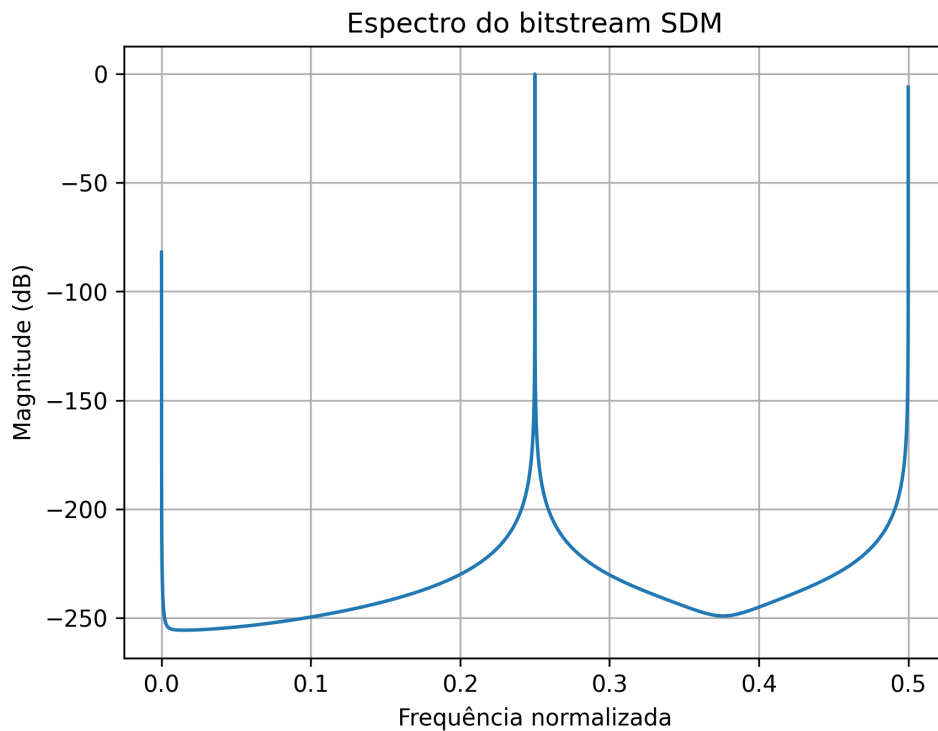


Figura 3.4: Espectro do bitstream do modulador para entrada  $DC = 0,75$ .

A Figura 3.4 apresenta o espectro correspondente à entrada de 0,75. Observa-se a presença de picos espectrais bem definidos, com concentração de energia em harmônicos discretos. Esse comportamento decorre do fato de que a entrada digital de 0,75 gera um bitstream altamente periódico [2], com média próxima a 1 e comutação regular entre 0 e 1. A periodicidade da saída reforça determinadas componentes harmônicas, visíveis como picos no espectro [2]. Além disso, nota-se um crescimento da densidade espectral nas altas frequências com inclinação aproximada de +20 dB

por década, característica esperada do *noise shaping* de moduladores sigma-delta de 1ª ordem, em que o ruído de quantização é empurrado para fora da banda de interesse [1,2].

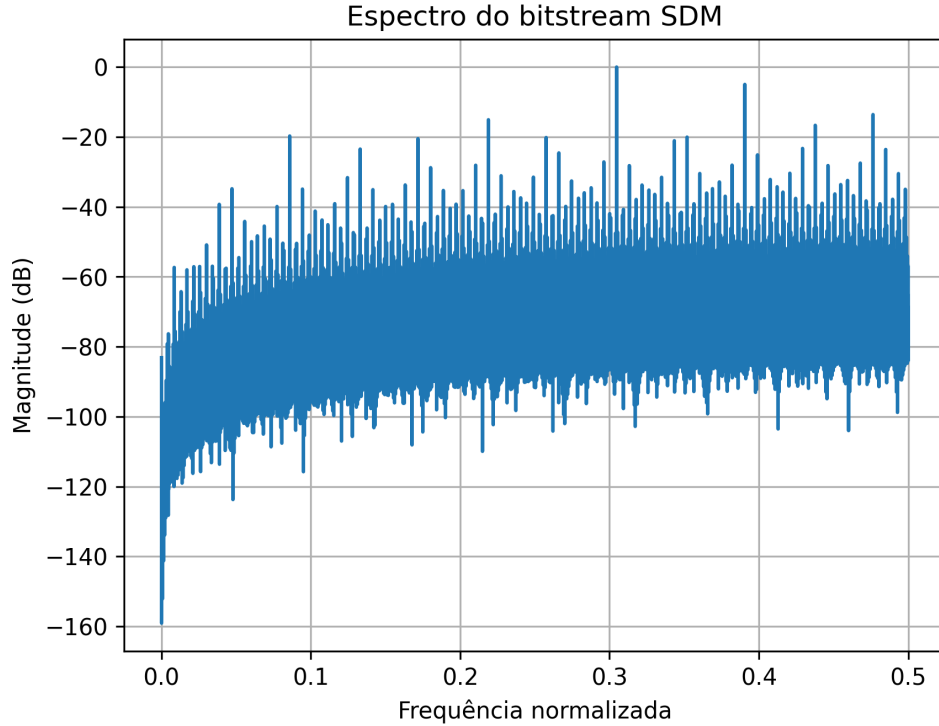


Figura 3.5: Espectro do bitstream do modulador para entrada DC  $\approx 0,305$ .

Já a Figura 3.5, referente à entrada DC de aproximadamente 0,305, revela um comportamento significativamente diferente. Nesse caso, a média de saída é menor e o bitstream é muito menos periódico, apresentando uma sequência aparentemente aleatória e irregular [2]. O espectro resultante é mais denso e ruidoso, com ausência de picos harmônicos claros, formando uma faixa espectral espessa. Ainda assim, a tendência de *noise shaping* permanece evidente: observa-se uma baixa densidade espectral nas frequências próximas a zero e um aumento contínuo até a frequência de Nyquist ( $f = 0,5$ ) [1,2]. A partir de aproximadamente  $f = 0,25$ , o espectro estabiliza-se em um patamar entre  $-60$  dB e  $-40$  dB, refletindo o limite físico do sinal 1-bit combinado à aleatoriedade do bitstream e à resolução da FFT.

Esse comportamento confirma a função do laço integrador no deslocamento do ruído para fora da banda base. Em termos de resposta em frequência, o atenuamento do ruído se aproxima de 20 dB por década, validando o desempenho teórico do modulador de 1ª ordem com oversampling [1,2]. Mesmo em entradas com menor

periodicidade, como no caso de 0,305, o padrão esperado de conformação do ruído é preservado.

As figuras foram obtidas a partir do script `espectro_freq_bitstream.py`, que realiza a leitura do bitstream, remoção da média, aplicação de janela de Hann e cálculo da FFT.

### 3.6 Integração com a Modulação ASK

Com o bitstream de saída do modulador sigma-delta disponível, torna-se possível aplicá-lo como máscara digital em uma modulação em amplitude (ASK, *Amplitude Shift Keying*) [8]. O bitstream gerado (saída `sdm_out`) atua como uma sequência binária que controla diretamente a ativação ou desativação de uma portadora senoidal [8].

A Figura 3.6 ilustra a integração entre os blocos, evidenciando que o sinal de entrada digital em ponto fixo (Q1.15) passa pelo modulador sigma-delta e, em seguida, é utilizado para chavear a portadora senoidal [8]. Esse mecanismo realiza a modulação ASK com dithering espectral, aproveitando as propriedades do SDM para espalhar o espectro [8].

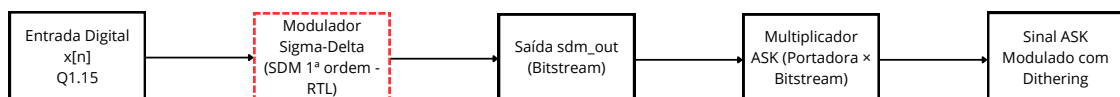


Figura 3.6: Diagrama funcional da integração entre o modulador sigma-delta e o transmissor ASK.

O efeito dessa integração é uma modulação ASK que mantém a média da amplitude desejada, mas elimina picos harmônicos indesejados que surgiriam com PWM periódico [8].

### 3.7 Resultados Consolidados

Nesta seção, consolidam-se os principais resultados obtidos com o modulador sigma-delta digital de 1ª ordem. A simulação funcional (GTKWave) confirmou o

comportamento esperado [10]: o bitstream de saída reflete a média da entrada DC (em Q1.15), como evidenciado pela densidade de 1s em `out_bit`. Para `in_val = 0,75`, o sinal é altamente periódico; já para `in_val = 0,305`, a saída torna-se pseudoaleatória, como resultado do dithering interno do laço sigma-delta [2].

A análise espectral (FFT) evidenciou o efeito de *noise shaping*: para entradas periódicas (como 0,75), surgem picos harmônicos discretos, enquanto para entradas não racionais (como 0,305), o espectro apresenta distribuição contínua e espessa. Em ambos os casos, observa-se o deslocamento do ruído de quantização para frequências elevadas, com inclinação de cerca de +20 dB/década, validando a resposta teórica do modulador [1, 2].

### 3.7.1 Análise Espectral da Modulação ASK

A integração do bitstream sigma-delta ao modulador ASK visa avaliar sua eficácia na dispersão de espúrios ao redor da portadora de 1 MHz [8]. Para isso, comparou-se o espectro gerado por uma modulação ASK tradicional (usando PWM periódico com duty-cycle fixo) contra a versão com dithering promovido pelo SDM [8]. As Figuras 3.7 e 3.8 mostram os resultados para entradas de 0,75 e 0,305, respectivamente.

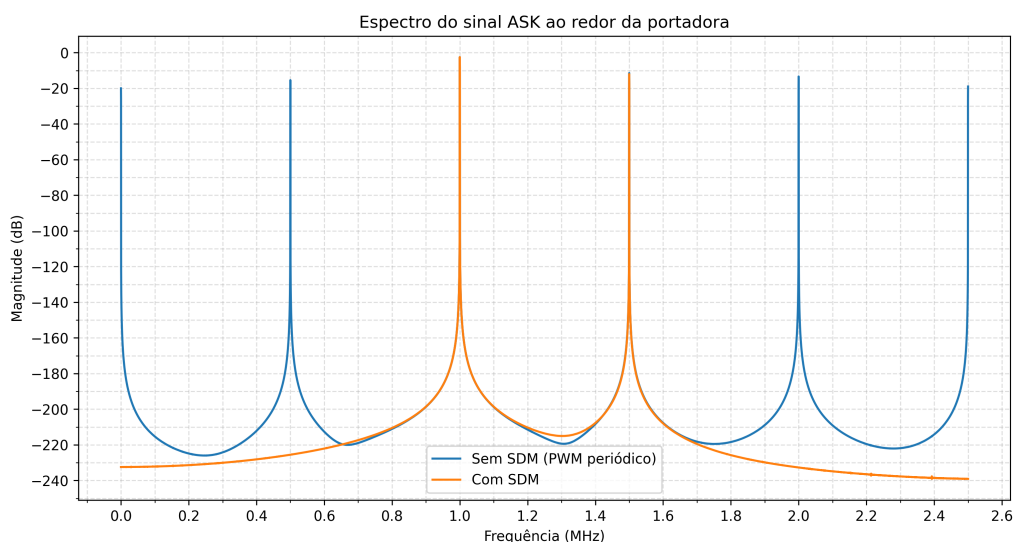


Figura 3.7: Comparação espectral entre ASK com SDM e PWM periódico para entrada 0,75.

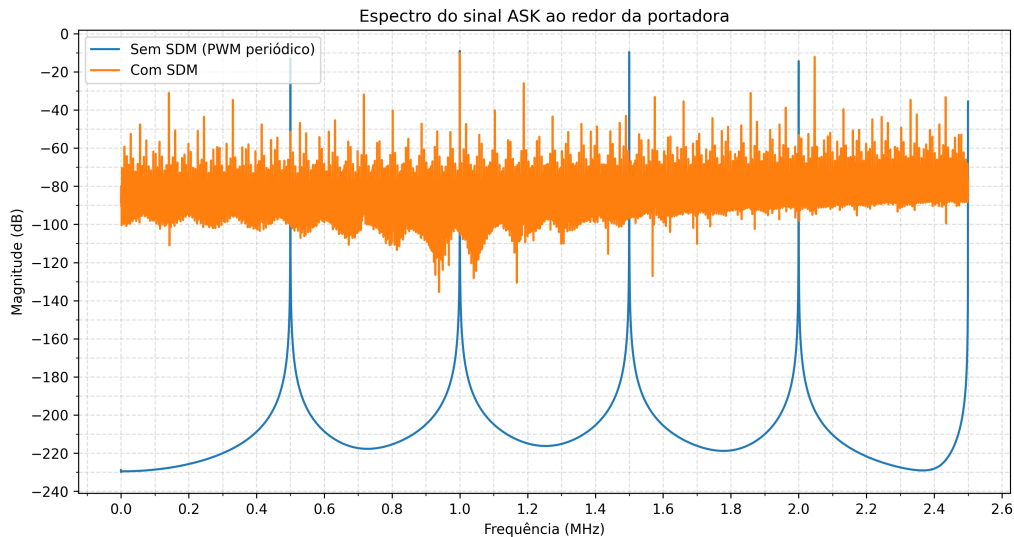


Figura 3.8: Comparação espectral entre ASK com SDM e PWM periódico para entrada 0,305.

Em ambos os casos, observa-se que o sinal gerado por PWM apresenta picos harmônicos discretos e estreitos fora da portadora de 1 MHz, indicando uma periodicidade rígida [8]. Por outro lado, a saída com SDM apresenta um espectro suavizado, com menor presença de espúrios nítidos, pois o dithering introduzido pelo sigma-delta espalha a energia espectral [2, 8]. No caso de 0,305 (Figura 3.8), essa dispersão é ainda mais evidente, com o espectro mostrando uma faixa densa e contínua em vez de picos isolados. Já para 0,75 (Figura 3.7), mesmo com a saída SDM ainda apresentando um pico secundário em  $\sim 1,5$  MHz, a redução geral de espúrios finos fora da portadora é observável, confirmando o benefício da técnica [8].

### 3.7.2 Implementação Física (KLayout / Place-and-Route)

Após a validação funcional e espectral, o design RTL foi sintetizado fisicamente utilizando o fluxo OpenLane [9] com o PDK IHP 130 nm [11]. O módulo `top_sdm_phys.v` foi utilizado como top-level, contendo apenas sinais escalares compatíveis com os pinos do padframe [10]. A ferramenta Yosys gerou a netlist lógica, e as etapas seguintes floorplanning, placement, routing e DRC foram executadas por ferramentas do OpenROAD e Magic [9]. O layout final é apresentado na Figura 3.9.

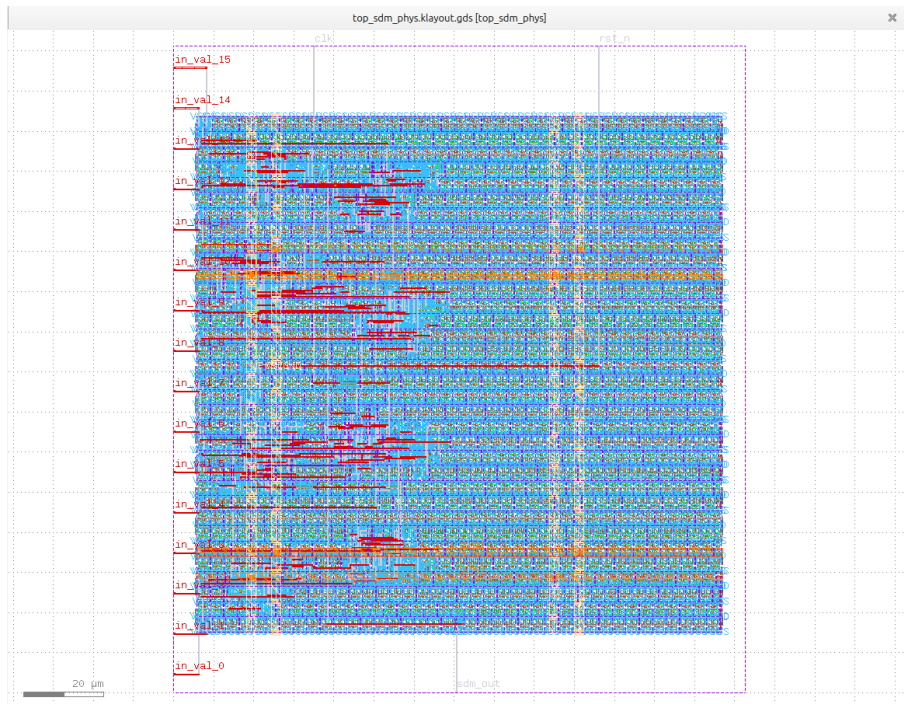


Figura 3.9: Layout físico do modulador sigma-delta sintetizado em tecnologia CMOS 130 nm, visualizado no KLayout.

O projeto passou com sucesso pelas verificações de conectividade e integridade elétrica (DRC/LVS), confirmando a viabilidade de fabricação. A área ocupada é compatível com o porte do circuito (acumulador, lógica combinacional e poucos flip-flops) [10], e a frequência de operação projetada (100 MHz) foi sustentada com folga. Assim, a implementação física ratifica a proposta de um modulador digital compacto e eficiente para aplicações em dither espectral [2, 8].

# Capítulo 4

## Conclusões

### 4.1 Considerações Finais

Este trabalho apresentou a implementação de um modulador sigma-delta de primeira ordem digital, incluindo sua integração com um esquema de modulação ASK. O objetivo principal projetar, implementar e validar o modulador sigma-delta foi plenamente atendido. A arquitetura do modulador (integrador acumulador com realimentação via DAC de 1 bit) foi codificada em Verilog, sintetizada e testada via simulação [10].

Foram realizadas testes de funcionalidade utilizando entradas DC em Q1.15, em diferentes níveis de amplitude. As simulações confirmaram o funcionamento correto do integrador e do DAC de feedback: a média do fluxo de bits de saída corresponde ao nível de entrada, como esperado pela teoria. As formas de onda dos sinais e do bit de saída foram examinadas no ambiente GTKWave, onde pôde-se observar a operação do acumulador e a transição do bit de saída em resposta às variações da entrada, corroborando o operação funcional do circuito implementado.

A análise espectral por meio da transformada rápida de Fourier (FFT) evidenciou o efeito de *noise shaping* característico de moduladores sigma-delta de primeira ordem [2]. Observou-se redução da densidade espectral de ruído nas baixas frequências (banda de interesse) e aumento do ruído nas altas frequências. Esse resultado corrobora com a teoria, indicando que o modulador desloca o ruído de quantização para fora da banda base útil, facilitando sua filtragem.

Ao integrar o modulador sigma-delta à modulação ASK, verificou-se impacto

no espectro do sinal transmitido. O sinal ASK modulado apresentou as bandas laterais típicas em torno da portadora, e o *noise shaping* do sigma-delta deslocou o ruído de quantização para regiões superiores, longe da banda base do receptor ASK [8]. Assim, foi constatado que o ruído na faixa de frequências onde está a portadora e os dados modulados foi significativamente atenuado, conforme antecipado, o que pode resultar em melhoria no desempenho do sistema de comunicação.

Quanto à viabilidade de implementação física em tecnologia CMOS de 130 nm (IHP SG13), o projeto tem alta compatibilidade, pois é essencialmente digital. A síntese em lógica padrão resultou em baixo overhead de área e consumo de energia, tornando possível sua conversão para hardware. Eventuais efeitos de jitter de clock ou ruído digital devem ser avaliados, mas não comprometem a implementação. Em suma, conclui-se que o modulador sigma-delta digital de 1ª ordem implementado atinge os objetivos propostos e funcionou conforme previsto, formando base sólida para desdobramentos futuros deste projeto.

## 4.2 Propostas para Trabalhos Futuros

A implementação realizada neste trabalho abre diversas possibilidades de aprimoramento e extensão. As seguintes propostas visam aprofundar a análise, melhorar o desempenho do sistema e viabilizar sua aplicação em contextos reais:

- **Projetar moduladores de ordem superior:** explorar moduladores sigma-delta de segunda e terceira ordem, além de variantes multibit, com o objetivo de alcançar uma conformação de ruído (*noise shaping*) ainda mais acentuada. Essa melhoria pode resultar em maior rejeição de ruído na banda de interesse, contribuindo para sistemas com maior precisão.
- **Aprimorar a análise espectral:** complementar a avaliação com métricas quantitativas como SNR (signal-to-noise ratio), THD (distorção harmônica total) e análise espectral mais detalhada. Isso permitirá uma comparação objetiva entre diferentes arquiteturas de moduladores.
- **Expandir os testes funcionais:** submeter o modulador a sinais de entrada dinâmicos, como formas de onda senoidais, triangulares ou até amostras re-

ais de áudio e vídeo. Testes com sinais não constantes permitirão avaliar a fidelidade da resposta do sistema em condições mais realistas.

- **Realizar integração com transmissores reais:** conectar o modulador a um transmissor ASK físico ou simulado para testar o desempenho completo do sistema de comunicação digital. Isso permitirá validar a eficiência do *noise shaping* em cenários práticos e a robustez do circuito frente a interferências e variações do canal.
- **Projetar o layout físico completo:** expandir o projeto físico para incluir não apenas o modulador sigma-delta, mas também o transmissor ASK, blocos auxiliares e filtros digitais. A ideia é formar um frontend digital completo e configurável para modulação ASK, sintetizado na tecnologia de 130 nm e pronto para aplicação em ASICs de baixo consumo.

# Referências Bibliográficas

- [1] DE LA ROSA, J. M., DEL RÍO, R., *CMOS Sigma-Delta Converters: Practical Design Guide*. 1st ed. John Wiley & Sons: Chichester, UK, 2013.
- [2] SCHREIER, R., TEMES, G. C., *Understanding Delta-Sigma Data Converters*. 2nd ed. IEEE Press / Wiley: Piscataway, NJ, 2017.
- [3] QIAN, L., DIAO, S., “A 112dB SNDR Delta-Sigma Modulator for Low-Power Audio Applications”. In: *14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5, Shanghai, China, October 2021.
- [4] ZAKY, B. M., OMRAN, H. A., ELSAYED, H. A., “Reconfigurable Multi-Mode Sigma Delta Modulator for 5G Applications”. In: *38th National Radio Science Conference (NRSC 2021)*, pp. 223–231, Mansoura, Egypt, July 2021.
- [5] KHOSHKAM, Z., OTHERS, “A Very Low-Power Discrete-Time Delta-Sigma Modulator for Wireless Body Area Network”, *Microelectronics Journal*, v. 130, pp. 105622, December 2022.
- [6] HAN, Y., OTHERS, “A Wide Dynamic Range Sigma-Delta Modulator for EEG Acquisition System”, *Sensors*, v. 22, n. 24, pp. 9824, December 2022.
- [7] AGUIRRE, P. C. C., *Projeto e Análise de Moduladores Sigma-Delta em Tempo Contínuo para Aplicação em Receptores de RF*, Master’s Thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil, October 2014.

- [8] KIM, B. H., LEE, K. Y., “ASK Modulator Spur Reduction Using Sigma Delta Modulator and Oscillator”. In: *2020 International SoC Design Conference (ISOCC)*, pp. 35–36, Yeosu, Korea (South), October 2020.
- [9] SHALAN, M., EDWARDS, T., “Building OpenLANE: A 130nm OpenROAD-Based Tapeout-Proven Flow”. In: *Workshop on Open-Source EDA Technology (WOSET)*, pp. 1–4, Virtual Conference, November 2020.
- [10] TARAATE, V., *Digital Logic Design Using Verilog: Coding and RTL Synthesis*. 1st ed. Springer: New Delhi, India, 2016.
- [11] IHP MICROELECTRONICS, “Open Source PDK Documentation: SG13G2 130nm BiCMOS Technology”, Technical Documentation, 2024, Available at: <https://ihp-open-pdk-docs.readthedocs.io>.
- [12] WENG, C.-H., LIN, C.-C., CHANG, Y.-C., et al., “A 0.89-mW 1-MHz 62-dB SNDR Continuous-Time Delta-Sigma Modulator With an Asynchronous Sequential Quantizer and Digital Excess-Loop-Delay Compensation”, *IEEE Transactions on Circuits and Systems II: Express Briefs*, v. 58, n. 12, pp. 867–871, December 2011.
- [13] BONTHALA, S., UPPOOR, Y., NAYAK, A., et al., “Design of High Resolution Delta Sigma Modulator in 180 nm CMOS Technology”. In: *Ninth International Symposium on Embedded Computing and System Design (ISED)*, pp. 1–6, Kollam, India, December 2019.
- [14] SOMAPPA, L., BAGHINI, M. S., “A 345 nW Power Continuous Time Delta Sigma Modulator for Biomedical Applications”. In: *IEEE 17th India Council International Conference (INDICON)*, pp. 1–4, New Delhi, India, December 2020.
- [15] CAO, S., LI, J., WEI, L., “A Multirate Double-Sampling Sigma Delta Modulator for Multi-Standard Wireless Radio Receivers”. In: *International Conference on Electric Information and Control Engineering*, pp. 1899–1903, Wuhan, China, April 2011.

- [16] BALASUBRAMANIAM, H., HOFMANN, K., “A New 60.2 dB 2nd Order Delta Sigma Modulator Using Open Loop Buffers Based on Current Conveyors”. In: *IEEE International Conference on Signal Processing, Computing and Control*, pp. 1–3, Wagnaghat Solan, India, September 2011.
- [17] HARRIS, D. M., HARRIS, S. L., *Digital Design and Computer Architecture: ARM Edition*. Arm edition ed. Morgan Kaufmann: Cambridge, MA, April 2015.
- [18] RAZAVI, B., *Design of Analog CMOS Integrated Circuits*. 2nd ed. McGraw-Hill: New York, NY, 2017.

# Apêndice A

## Códigos Verilog Desenvolvidos

### sdm\_1.v – Modulador Sigma-Delta de 1ª Ordem

```
1 module sdm_1 #(
2     parameter IN_WIDTH = 16,
3     parameter ACC_WIDTH = IN_WIDTH + 3
4 ) (
5     input wire          clk,
6     input wire          rst_n,
7     input wire [IN_WIDTH-1:0] in_val,
8     output reg          out_bit
9 );
10
11     // Acumulador principal
12     reg [ACC_WIDTH-1:0] acc1;
13
14     // Extensão da entrada
15     wire [ACC_WIDTH-1:0] in_ext =
16         { in_val, {(ACC_WIDTH-IN_WIDTH){1'b0}} };
17
18     // Full-scale do DAC 1-bit (FS = 2^(ACC_WIDTH-1))
19     localparam [ACC_WIDTH-1:0] FS =
20         {1'b1, {ACC_WIDTH-1{1'b0}}};
21
```

```

22 // Próximo valor do acumulador (integrador + feedback)
23 wire [ACC_WIDTH-1:0] acc1_next =
24     out_bit ? (acc1 + in_ext - FS) : (acc1 + in_ext);
25
26 // Próximo valor do bit de saída: MSB do novo acumulador
27 wire out_bit_next = acc1_next[ACC_WIDTH-1];
28
29 always @(posedge clk or negedge rst_n) begin
30     if (!rst_n) begin
31         acc1    <= {ACC_WIDTH{1'b0}};
32         out_bit <= 1'b0;
33     end else begin
34         acc1    <= acc1_next;
35         out_bit <= out_bit_next;
36     end
37 end
38
39 endmodule

```

## tb\_sdm.v – Testbench Funcional

```

1 `timescale 1ns/1ps
2
3 module tb_sdm;
4
5     // Parâmetros
6     localparam integer IN_WIDTH  = 16;
7     localparam integer ACC_WIDTH = IN_WIDTH + 3;
8
9     // Sinais
10    reg                clk;
11    reg                rst_n;
12    reg [IN_WIDTH-1:0] in_val;
13    wire               out_bit;

```

```

14
15 // DUT: modulador sigma-delta 1ª ordem, 1 bit
16 sdm_1 #(
17     .IN_WIDTH (IN_WIDTH),
18     .ACC_WIDTH(ACC_WIDTH)
19 ) dut (
20     .clk      (clk),
21     .rst_n   (rst_n),
22     .in_val  (in_val),
23     .out_bit(out_bit)
24 );
25
26 // Arquivo para gravar o bitstream
27 integer f;
28 integer i;
29
30 initial begin
31
32     // Dump para GTKWave
33     $dumpfile("tb_sdm.vcd");
34     $dumpvars(0, tb_sdm);
35
36     clk      = 1'b0;
37     rst_n   = 1'b0;
38
39     // Escolha da entrada DC em Q1.15
40     in_val = 16'd24576; // entrada ~0.75 em Q1.15
41
42     // Arquivo texto para FFT
43     f = $fopen("sdm_out.txt", "w");
44
45     // Reset inicial
46     #50 rst_n = 1'b1;
47

```

```

48     // Loop de simulação: gera 50000 amostras
49     for (i = 0; i < 50000; i = i + 1) begin
50         @(posedge clk);
51         $fwrite(f, "%0d\n", out_bit);
52     end
53
54     $fclose(f);
55     $finish;
56 end
57
58 // Clock 100 MHz (período 10 ns)
59 always #5 clk = ~clk;
60
61 endmodule

```

## top\_sdm.v – Módulo Top Lógico

```

1 // Módulo de topo para síntese com LibreLane / PDK IHP
2
3 module top_sdm #(
4     parameter IN_WIDTH  = 16,
5     parameter ACC_WIDTH = IN_WIDTH + 3
6 )(
7     input  wire          clk,
8     input  wire          rst_n,
9     input  wire [IN_WIDTH-1:0] in_val,
10    output wire          sdm_out
11 );
12
13    sdm_1 #(
14        .IN_WIDTH (IN_WIDTH),
15        .ACC_WIDTH(ACC_WIDTH)
16    ) u_mash (
17        .clk      (clk),

```

```

18     .rst_n (rst_n),
19     .in_val (in_val),
20     .out_bit(sdm_out)
21 );
22
23 endmodule

```

## top\_sdm\_phys.v – Módulo Top para Síntese Física

```

1 // Topo físico: pinos escalares, interno usa top_sdm com barramento
2
3 module top_sdm_phys (
4     input  wire clk,
5     input  wire rst_n,
6     input  wire in_val_0,
7     input  wire in_val_1,
8     input  wire in_val_2,
9     input  wire in_val_3,
10    input  wire in_val_4,
11    input  wire in_val_5,
12    input  wire in_val_6,
13    input  wire in_val_7,
14    input  wire in_val_8,
15    input  wire in_val_9,
16    input  wire in_val_10,
17    input  wire in_val_11,
18    input  wire in_val_12,
19    input  wire in_val_13,
20    input  wire in_val_14,
21    input  wire in_val_15,
22    output wire sdm_out
23 );
24
25 // Reconstrói barramento interno
26 wire [15:0] in_val;

```

```

26
27   assign in_val = {
28       in_val_15,
29       in_val_14,
30       in_val_13,
31       in_val_12,
32       in_val_11,
33       in_val_10,
34       in_val_9,
35       in_val_8,
36       in_val_7,
37       in_val_6,
38       in_val_5,
39       in_val_4,
40       in_val_3,
41       in_val_2,
42       in_val_1,
43       in_val_0
44   };
45
46   // Usa o top_sdm original
47   top_sdm #(
48       .IN_WIDTH (16),
49       .ACC_WIDTH(16+3)
50   ) u_top (
51       .clk      (clk),
52       .rst_n    (rst_n),
53       .in_val   (in_val),
54       .sdm_out (sdm_out)
55   );
56
57 endmodule

```

# Apêndice B

## Scripts Python para Simulação e Análise

### ask\_sim.py – Comparação Espectral com e sem Sigma-Delta

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.ticker as ticker
4
5 # 1) Parâmetros
6 Fs = 10e6          # Hz
7 fc = 1e6           # Hz
8 Tsym = 2e-6       # s (duração de símbolo)
9 Ns = int(Fs * Tsym) # amostras por símbolo
10 Nsym = 2000       # número de símbolos
11 A = 0.305         # amplitude desejada (0..1)
12
13 # 2) Geração do bitstream CDM SDM(arquivo já gerado pelo Verilog)
14 sdm_bits = np.loadtxt("sdm_out(0.305DC).txt")
15 sdm_bits = sdm_bits.astype(int)
16
17 Ntotal = Nsym * Ns
18 sdm_bits = sdm_bits[:Ntotal] # pega os primeiros Ntotal bits
```

```

19
20
21 # 3) Geração do PWM periódico (SEM SDM)
22 # duty-cycle aproximando A
23 ones_per_sym = int(round(A * Ns))
24 pwm_pattern = np.array([1]*ones_per_sym + [0]*(Ns - ones_per_sym))
25
26 pwm_bits = np.tile(pwm_pattern, Nsym)
27
28 # 4) Geração da portadora e ASK
29 n = np.arange(Ntotal)
30 carrier = np.cos(2 * np.pi * fc * n / Fs)
31
32 s_sem_sdm = pwm_bits * carrier      # ASK usando PWM periódico
33 s_com_sdm = sdm_bits * carrier      # ASK usando bitstream do SDM
34
35 # 5) FFT e espectros
36 def calc_spectrum(x, Fs):
37     N = len(x)
38     window = np.hanning(N)
39     X = np.fft.rfft(x * window)
40     freqs = np.fft.rfftfreq(N, 1/Fs)
41     # normaliza pela energia da janela
42     X_mag = np.abs(X) / np.sum(window) * 2
43     X_db = 20*np.log10(X_mag + 1e-15)
44     return freqs, X_db
45
46 f1, S1 = calc_spectrum(s_sem_sdm, Fs)
47 f2, S2 = calc_spectrum(s_com_sdm, Fs)
48
49 # 6) Plotar perto da portadora
50 fmin = fc - 10e5 # fc - 1000 kHz(1 MHz), para ver o efeito da dispersão
    ↪ de ruído
51 fmax = fc + 15e5 # fc + 1500 kHz(1.5 MHz)

```

```

52
53 mask1 = (f1 >= fmin) & (f1 <= fmax)
54 mask2 = (f2 >= fmin) & (f2 <= fmax)
55
56 plt.figure()
57 plt.plot(f1[mask1]/1e6, S1[mask1], label="Sem SDM (PWM periódico)")
58 plt.plot(f2[mask2]/1e6, S2[mask2], label="Com SDM")
59 plt.xlabel("Frequência (MHz)")
60 plt.ylabel("Magnitude (dB)")
61 plt.title("Espectro do sinal ASK ao redor da portadora")
62 plt.legend()
63 plt.grid(True)
64
65 # Escala mais detalhada no eixo Y: grid a cada 10 dB
66 plt.gca().yaxis.set_major_locator(ticker.MultipleLocator(20))
67
68 # Escala mais fina no eixo Y: grid secundário a cada 5 dB
69 plt.gca().yaxis.set_minor_locator(ticker.MultipleLocator(10))
70
71 # Ativa grid para major e minor ticks
72 plt.grid(which='both', linestyle='--', alpha=0.4)
73
74 # Grid mais fino no eixo X (ex: a cada 0.02 MHz)
75 plt.gca().xaxis.set_major_locator(ticker.MultipleLocator(0.2))
76 plt.gca().xaxis.set_minor_locator(ticker.MultipleLocator(0.1))
77
78 plt.gcf().set_size_inches(12, 6) # largura = 12 pol, altura = 6 pol
79 plt.savefig("espectro_ask_sdm_vs_pwm(0305).png", dpi=300,
    ↪  bbox_inches='tight')
80
81 plt.show()

```

# espectro\_freq\_bitstream.py – FFT do Bitstream

## Sigma-Delta

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 #Para carregar o bitstream gerado pelo Verilog
5 fname = "sdm_out(0.75DC).txt" # ajuste o nome
6 y = np.loadtxt(fname)
7 N = len(y)
8 fs = 1.0 # frequência de amostragem normalizada
9
10 #Cálculos estatísticos do bitstream
11 mean_val = np.mean(y)
12 prop_ones = np.mean(y == 1)
13
14 print(f"Arquivo: {fname}")
15 print(f"Número de amostras: {N}")
16 print(f"Média (0/1) : {mean_val:.6f}")
17 print(f"Proporção de '1's : {prop_ones:.6f}")
18
19 #Remove média (para focar no ruído no espectro)
20 y_ac = y - mean_val
21
22 #Aplica janela de Hann
23 w = np.hanning(N)
24 y_w = y_ac * w
25
26 #FFT
27 Y = np.fft.fft(y_w)
28 Y_mag = np.abs(Y[:N//2]) # metade positiva
29 f = np.linspace(0, fs/2, N//2)
30
31 #Convertendo para dB
```

```
32 Y_db = 20 * np.log10(Y_mag / np.max(Y_mag))
33
34 plt.plot(f, Y_db)
35 plt.xlabel("Frequência normalizada")
36 plt.ylabel("Magnitude (dB)")
37 plt.title("Espectro do bitstream SDM")
38 plt.grid(True)
39 plt.savefig("espectro_bitstream(075).png", dpi=300, bbox_inches='tight')
40 plt.show()
```