UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

NESTOR FRANÇA DE ANDRADE JÚNIOR

PROPOSTA DE OSCILADOR DIGITAL EM FPGA PARA SISTEMAS DE COMUNICAÇÕES

PAU DOS FERROS
2026

NESTOR FRANÇA DE ANDRADE JÚNIOR

PROPOSTA DE OSCILADOR DIGITAL EM FPGA PARA SISTEMAS DE COMUNICAÇÕES

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Pedro Thiago Valério de Souza, Prof. Dr.

PAU DOS FERROS

2026

A553p Andrade Júnior, Nestor França de .
Proposta de oscilador digital em FPGA para sistemas de comunicações / Nestor França de Andrade Júnior. - 2026.
44 f. : il.

Orientador: Pedro Thiago Valério de Souza.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Engenharia de Computação, 2026.

1. FPGA. 2. ASK. 3. FSK. 4. PSK. 5. Oscilador Digital. I. Souza, Pedro Thiago Valério de, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

NESTOR FRANÇA DE ANDRADE JÚNIOR

PROPOSTA DE OSCILADOR DIGITAL EM FPGA PARA SISTEMAS DE COMUNICAÇÕES

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Engenharia de Computação.

Defendida em: 03 / 07 / 2026

BANCA EXAMINADORA

Pedro Thiago Valério de Souza, Prof. Dr. (UFERSA)
Presidente

Francisco Carlos Gurgel da Silva Segundo, Prof. Dr. (UFERSA)
Membro Examinador

Kennedy Reurison Lopes, Prof. Dr. (UFERSA)
Membro Examinador

DEDICATÓRIA

Dedico este trabalho à minha família e a todos que, de alguma forma, contribuíram para a realização deste momento. (*Presentes*)

AGRADECIMENTOS

Primeiramente, agradeço a Deus, que me deu oportunidade, força e coragem para superar todos os desafios.

Agradeço também aos meus pais, Nestor e Francisca, por todo o apoio que me deram durante esse ciclo, pelo incentivo constante e por nunca medirem esforços para que eu chegasse até aqui. Agradeço também à minha irmã, Andyária, que sempre esteve ao meu lado, me apoiando e torcendo pelo meu sucesso.

Aos professores do curso de Engenharia de Computação, em particular a Pedro Thiago e Petrucio Ricardo, pelos ensinamentos e orientações que foram fundamentais para minha formação.

Aos amigos que fiz em Computação, em particular Anderson Augusto e Gustavo Mendes, que estiveram comigo desde o início do curso e me ajudaram em tudo. Agradeço também a Herbert Juan e Libhinny Karolayne, grandes amigos que conheci em C&T e que tornaram essa jornada mais fácil e memorável.

Agradeço também a todos os "membros honorários"do antigo LASBIO.

Por fim, quero agradecer ao meu professor do ensino médio, Francivaldo, por me apresentar a disciplina de robótica, que me possibilitou escolher esse curso.

EPÍGRAFE

"A mente que se abre a uma nova ideia
jamais voltará ao tamanho original."
(Albert Einstein)

RESUMO

Sistemas de comunicações digitais apresentam desafios na representação de sinais analógicos, especialmente em plataformas com recursos limitados como as *Field-Programmable Gate Array* (FPGA). Neste contexto, este trabalho propõe o projeto e a implementação de um oscilador digital em FPGA capaz de gerar sinais modulados nos esquemas *Amplitude Shift Keying* (ASK), *Frequency Shift Keying* (FSK) e *Phase Shift Keying* (PSK). A contribuição central deste trabalho consiste em uma implementação do oscilador através de uma equação de diferenças, ao invés da técnica convencional de utilização de uma tabela de consulta. O sistema foi implementado utilizando aritmética em ponto fixo e um conversor digital-analógico sigma-delta, com um circuito de filtragem e amplificação na saída para reconstrução do sinal analógico. Os resultados obtidos demostra que o oscilador gera um sinal senoidal limpos e com baixo ruído, validando experimentalmente as três modulações propostas.

Palavras-chave: FPGA; oscilador digital; ASK; FSK; PSK.

ABSTRACT

Digital communication systems present challenges in the representation of analog signals, especially on resource-constrained platforms such as FPGA. In this context, this work proposes the design and implementation of a digital oscillator on FPGA capable of generating modulated signals using ASK, FSK, and PSK schemes. The central contribution of this work consists of an implementation of the oscillator through a difference equation, rather than the conventional technique of using a lookup table. The system was implemented using fixed-point arithmetic and a sigma-delta digital-to-analog converter, with a filtering and amplification circuit at the output for reconstruction of the analog signal. The results obtained demonstrate that the oscillator generates a clean sinusoidal signal with low noise, experimentally validating the three proposed modulation schemes.

Keywords: FPGA; digital oscillator; ASK; FSK; PSK.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	*Analog-to-Digital Converter*
AM	*Amplitude Modulation*
ASIC	*Application-Specific Integrated Circuit*
ASK	*Amplitude Shift Keying*
BASK	*Binary Amplitude Shift Keying*
BER	*Bit Error Rate*
BFSK	*Binary Frequency Shift Keying*
BPSK	*Binary Phase Shift Keying*
DAC	*Digital-to-Analog Converter*
DPSK	*Differential Phase Shift Keying*
DSP	*Digital Signal Processor*
EDP	*Energy-Delay Product*
EVM	*Error Vector Magnitude*
FFT	*Fast Fourier Transform*
FM	*Frequency Modulation*
FPGA	*Field-Programmable Gate Array*
FSK	*Frequency Shift Keying*
FXU	*Fixed-point processing Unit*
GND	*Ground*
HDL	*Hardware Description Language*
ISA	*Instruction Set Architecture*
LSB	*Least Significant Bit*
LUT	*Look-Up Table*
OOK	*On-Off Keying*
OSR	*Oversampling Ratio*
PM	*Phase Modulation*
PMSM	*Permanent Magnet Synchronous Motor*
PSK	*Phase Shift Keying*
QAM	*Quadrature Amplitude Modulation*
QPSK	*Quadrature Phase Shift Keying*
SDR	*Software Defined Radio*

LISTA DE SÍMBOLOS

$h[n]$	Resposta ao impulso do sistema (sinal cossenoidal discreto)
A	Amplitude do sinal
ω_0	Frequência angular digital
ϕ	Fase inicial do sinal
$u[n]$	Degrau unitário
$x[n]$	Sinal de entrada do sistema discreto
$y[n]$	Sinal de saída do sistema discreto
$\delta[n]$	Impulso unitário discreto
z	Variável complexa da Transformada Z
$H(z)$	Função de transferência no domínio Z
$X(z)$	Sinal de entrada no domínio Z
$Y(z)$	Sinal de saída no domínio Z
$s(t)$	Sinal modulado
f_c	Frequência da portadora
f_{1c}, f_{2c}	Frequências da portadora na modulação FSK
ϕ_1, ϕ_2	Fases da portadora na modulação PSK
$m(t)$	Sinal de mensagem (modulante)
$Q_{(m,n)}$	Notação de ponto fixo, com m bits na parte inteira e n bits na parte fracionária
S	Bit de sinal na representação em ponto fixo
2^n	Fator de escala em ponto fixo
m_x, n_x	Bits da parte inteira e fracionária do operando x
m_y, n_y	Bits da parte inteira e fracionária do operando y
m_z, n_z	Bits da parte inteira e fracionária do resultado z
Σ-Δ	Conversor sigma-delta
f_s	Frequência de amostragem

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de comunicações digitais têm ganhado crescente relevância no cenário tecnológico contemporâneo, impulsionados pelos avanços nas tecnologias de circuitos integrados e pelo aumento da capacidade de processamento dos sistemas eletrônicos. Em razão das vantagens inerentes ao processamento digital de sinais — como maior imunidade a ruídos, flexibilidade, confiabilidade e possibilidade de otimização conjunta de desempenho e consumo energético (Proakis; Manolakis, 2006; Engelen; Plassche, 2013), observa-se, em diversas áreas, uma progressiva substituição das técnicas analógicas por soluções digitais.

Embora os sistemas de comunicação digitais apresentem muitas vantagens, devido ao ambiente físico que esses sistemas operam ser predominantemente analógico, o que torna necessária a realização de processos de amostragem e quantização (Oppenheim; Schafer, 2010). Assim sendo, técnicas de processamento digital são empregadas para mapear os sinais digitais em formas de onda adequadas para transmissão e recepção (Proakis; Manolakis, 2006).

Sistemas de comunicações digitais podem ser implementados em diversas plataformas, entre as quais se destacam as FPGA. As FPGAs consistem em plataformas de *hardware* reconfigurável, proporcionando elevada flexibilidade (Nito, 2017; Babu; Parthasarathy, 2021). Além disso, esses dispositivos apresentam capacidade de processamento paralelo, permitindo a implementação de sistemas digitais complexos. Contudo, a representação e a geração de sinais analógicos nesses dispositivos ainda constituem um desafio significativo (Silva *et al.*, 2020; Nito, 2017). Por exemplo, as FPGAs apresentam limitações relacionadas à área de implementação e ao consumo de energia, o que torna o uso de aritmética em ponto flutuante menos adequado em determinadas aplicações (Winandy *et al.*, 2025). Como alternativa, a aritmética em ponto fixo pode ser adotada, por proporcionar uma implementação mais simples e eficiente em *hardware* (Przybył, 2021; Födisch *et al.*, 2016).

Assim sendo, neste trabalho, propõe-se a modelagem e implementação de um sistema de modulação digital em FPGA utilizando aritmética em ponto fixo. O sistema proposto visa gerar sinais modulados utilizando os esquemas de ASK, PSK e FSK, os quais serão posteriormente convertidos para o domínio analógico por meio de um conversor sigma-delta. Uma contribuição relevante deste trabalho é a proposição de um método para a geração de sinais cossenoidais em FPGA utilizando um oscilador digital baseado em uma equação em diferenças, ao invés de uma utilização de uma tabela de consulta, diminuindo assim o requisito de memória para implementação do modulador.

1.1 Justificativa

A necessidade de representação de sinais analógicos em *hardware* reconfigurável, como FPGAs, mostra-se cada vez mais relevante no contexto atual dos sistemas de comunicações digitais e sistemas embarcados. A utilização de aritmética de ponto fixo contribui para a redução do custo computacional e para o uso mais eficiente dos recursos de *hardware*, além de aumentar a previsibilidade do sistema, tornando-o mais robusto e adequado para aplicações práticas.

Além disso, os conversores digital-analógicos baseados em arquitetura sigma-delta destacam-se pelo baixo custo de implementação e pela capacidade de produzir sinais de alta resolução, mesmo utilizando estruturas relativamente simples, o que os torna amplamente empregados em sistemas digitais modernos.

Dessa forma, o desenvolvimento e a implementação de um sistema capaz de gerar e modular sinais digitais em FPGA proporcionam não apenas um aprofundamento teórico, mas também uma aplicação prática dos conceitos estudados, consolidando conhecimentos nas áreas de processamento digital de sinais, eletrônica digital e comunicações.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver e implementar um modulador digital em FPGA, capaz de gerar sinais modulados nos esquemas ASK, PSK e FSK, realizando a conversão de dados binários em um sinal analógico por meio de um conversor digital-analógico baseado em modulação sigma-delta.

1.2.2 Objetivos Específicos

- Modelar o sinal cossenoidal no domínio discreto por meio da Transformada Z, obtendo a equação de diferenças correspondente.
- Implementar, na linguagem *Python*, a equação em diferenças de forma iterativa, utilizando aritmética de ponto fixo para determinação dos coeficientes e das condições iniciais.
- Determinar e validar os coeficientes e parâmetros necessários para a implementação em *hardware*.
- Implementar o gerador de sinal em *SystemVerilog*, utilizando os parâmetros obtidos na etapa de simulação.

- Desenvolver um módulo de reescalonamento para adequação do sinal à faixa de entrada do conversor digital-analógico.
- Implementar um conversor digital-analógico baseado em modulação sigma-delta, visando à reconstrução do sinal no domínio analógico.
- Implementar os esquemas de modulação ASK, PSK e FSK em *Python*, com o objetivo de obter os coeficientes e parâmetros específicos de cada técnica.
- Desenvolver os moduladores ASK, PSK e FSK em *SystemVerilog*.
- Validar os sinais gerados por meio de simulações e medições experimentais com o uso de osciloscópio.
- Desenvolver um módulo de controle para seleção dinâmica do tipo de modulação.

1.3 Estrutura da Monografia

O restante do trabalho está organizado da seguinte forma:

No Capítulo 1, será apresentada uma visão geral do tema, a justificativa e os objetivos do trabalho.

No Capítulo 2, será apresentada a teoria utilizada na elaboração deste trabalho.

No Capítulo 3, serão apresentados os trabalhos relacionados.

No Capítulo 4, será apresentada a metodologia utilizada na implementação do experimento.

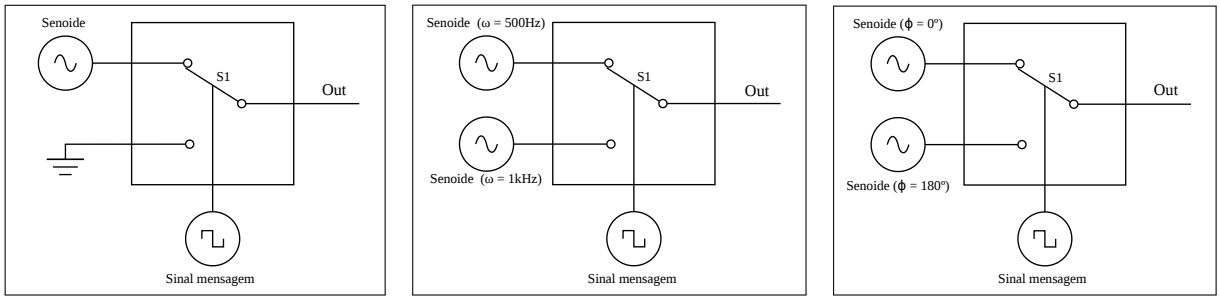
No Capítulo 5, serão apresentados os resultados obtidos na simulação e na implementação.

Por fim, no Capítulo 6, serão discutidas as considerações finais acerca do trabalho apresentado.

2 REFERENCIAL TEÓRICO

2.1 Modulações

Figura 1 – Diagrama de Bloco: (a) ASK, (b) FSK e (c) PSK.



(a) (b) (c)

Fonte: Autoria própria, (2026).

2.1.1 Modulação ASK

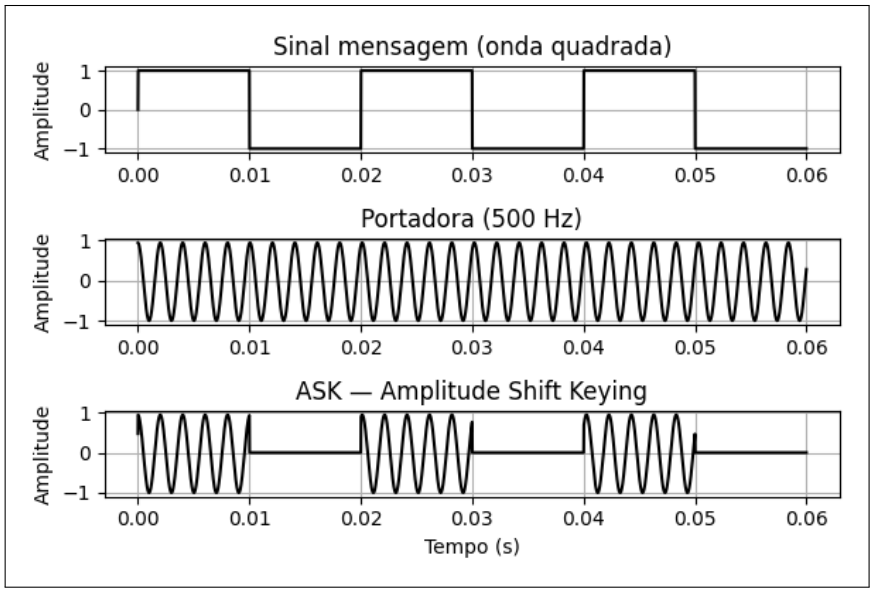
A modulação ASK consiste na variação da amplitude da portadora, mantendo sua frequência e fase constantes (Yue, 2025). Essa modulação tem como característica a simplicidade, o que facilita sua implementação e resulta em baixa complexidade de circuito. No entanto, uma desvantagem significativa está em sua sensibilidade ao ruído, pois interferências no canal afetam diretamente a amplitude do sinal, tornando a ASK menos adequada para ambientes com elevado nível de interferências eletromagnéticas. Sua representação matemática é dada por:

$$s(t) = \begin{cases} A\cos(2\pi f_c t + \phi), & \text{se } m(t) = 1 \\ 0, & \text{se } m(t) = 0 \end{cases} \tag{2.1}$$

Onde $s(t)$ representa o sinal modulado, A, f_c e ϕ representam, respectivamente, a amplitude, a frequência da portadora e a fase do sinal. O sinal de mensagem é definido como $m(t) \in \{0,1\}$. Na modulação ASK, a informação é transmitida por meio da variação da amplitude da portadora, enquanto a frequência f_c e a fase ϕ permanecem constantes. Quando a amplitude assume o valor zero, o sinal da portadora é desligado, caracterizando o caso particular conhecido como *On-Off Keying* (OOK), ou modulação liga-desliga, em que os símbolos transmitidos correspondem aos níveis lógico 0 e 1.

A Figura 1a apresenta o diagrama em blocos da implementação da modulação ASK, na qual a chave $S1$ realiza a seleção entre a portadora senoidal e o nível lógico zero (*Ground* (GND)), de acordo com o sinal de mensagem aplicado. Dessa forma, quando o sinal de mensagem possui nível lógico alto (1), a portadora é transmitida para a saída, enquanto para o nível lógico baixo (0) a saída permanece em zero.

Figura 2 – Forma de onda da modulação ASK



Fonte: Autoria própria, (2026).

A Figura 2 apresenta as formas de onda esperadas na saída do modulador ASK. Observa-se que a portadora é presente apenas nos intervalos em que o sinal de mensagem possui valor igual a 1, caracterizando a modulação por chaveamento de amplitude. Quando o sinal de mensagem assume o valor 0, a portadora é suprimida, resultando em ausência de sinal na saída.

2.1.2 Modulação FSK

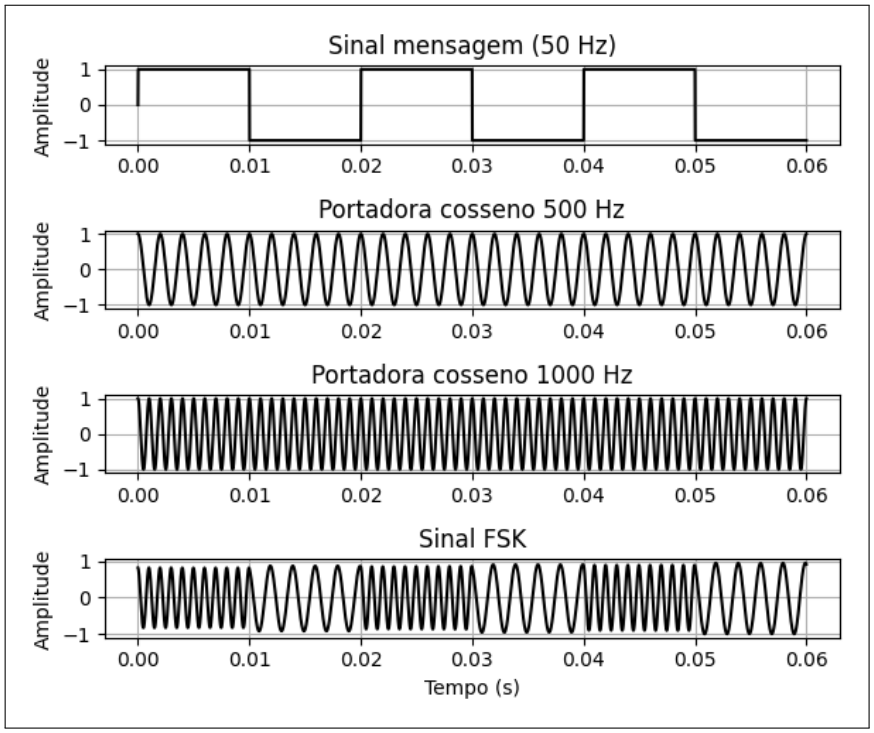
A modulação FSK consiste na variação da frequência da portadora conforme o símbolo transmitido, em que cada símbolo é representado por uma frequência distinta, enquanto a amplitude e a fase permanecem constantes (Swain, 2020). Por não depender da amplitude para transmitir a informação, essa modulação é menos suscetível ao ruído e a variações de potência no canal, tornando-a mais adequada para ambientes com elevado nível de interferências em comparação à ASK. Como desvantagem, a FSK exige uma largura de banda maior do que a ASK, pois cada símbolo ocupa uma faixa de frequência diferente, o que pode ser uma limitação em sistemas com espectro restrito. Sua representação matemática é dada por:

$$s(t) = \begin{cases} A\cos(2\pi f_{1c}t + \phi), & \text{se } m(t) = 1 \\ A\cos(2\pi f_{2c}t + \phi), & \text{se } m(t) = 0 \end{cases} \tag{2.2}$$

onde $s(t)$ representa o sinal modulado, A e ϕ representam, respectivamente, a amplitude e a fase do sinal, e f_{1c} e f_{2c} são as duas frequências da portadora. O sinal de mensagem é definido como $m(t) \in \{0, 1\}$. Na modulação FSK, a informação é transmitida por meio da variação da frequência da portadora, enquanto a amplitude A e a fase ϕ permanecem constantes.

Na Figura 1b é apresentado o diagrama de blocos da implementação FSK, no qual a chave $S1$ realiza a seleção entre a portadora senoidal de 500 Hz e a portadora de 1 kHz, de acordo com o sinal de mensagem aplicado. Neste trabalho foram utilizadas as frequências de 1 kHz e 10 kHz. Dessa forma, quando o sinal de mensagem possui nível lógico alto (1), o sinal modulado assume a portadora de maior frequência, enquanto para o nível lógico baixo (0) a portadora de menor frequência é transmitida na saída.

Figura 3 – Forma de onda da modulação FSK.



Fonte: Autoria própria, (2026).

Na Figura 3 é apresentada a forma de onda esperada na modulação FSK. Observa-se que o sinal de saída assume a portadora de maior frequência quando o sinal de mensagem é igual a 1, e a portadora de menor frequência quando o sinal de mensagem é igual a 0.

2.1.3 Modulação PSK

A modulação PSK, por sua vez, é caracterizada pela variação da fase da portadora conforme o símbolo transmitido, mantendo a frequência e a amplitude constantes (Islam *et al.*, 2009; Bomfin; Chafii; Fettweis, 2019). Por codificar a informação exclusivamente na fase, essa modulação apresenta maior imunidade ao ruído em comparação às modulações ASK e FSK, uma vez que variações de amplitude no canal não comprometem a recuperação do sinal. Como desvantagem, a PSK exige maior complexidade no receptor, pois a demodulação coerente requer um sinal de referência de fase sincronizado com o transmissor, tornando-a a mais complexa das três modulações abordadas neste trabalho. Sua representação matemática é dada por:

$$s(t) = \begin{cases} A\cos(2\pi f_c t + \phi_1), & \text{se } m(t) = 1 \\ A\cos(2\pi f_c t + \phi_2), & \text{se } m(t) = 0 \end{cases} \tag{2.3}$$

onde $s(t)$ representa o sinal modulado, A e f_c representam, respectivamente, a amplitude e a frequência da portadora, e ϕ_1 e ϕ_2 são as duas fases do sinal. O sinal de mensagem é definido como $m(t) \in \{0,1\}$. Na modulação PSK, a informação é transmitida por meio da variação da fase da portadora, enquanto a amplitude A e a frequência f_c permanecem constantes.

A Figura 1c apresenta o diagrama de blocos correspondente à modulação PSK, no qual a chave $S1$ realiza a seleção entre a portadora senoidal com fase $\phi = 0^\circ$ e a portadora com fase $\phi = 180^\circ$, de acordo com o sinal de mensagem aplicado. Assim, quando o sinal de mensagem possui nível lógico alto (1), a portadora com fase de 0° é transmitida para a saída, enquanto para o nível lógico baixo (0) a portadora com fase de 180° é transmitida.

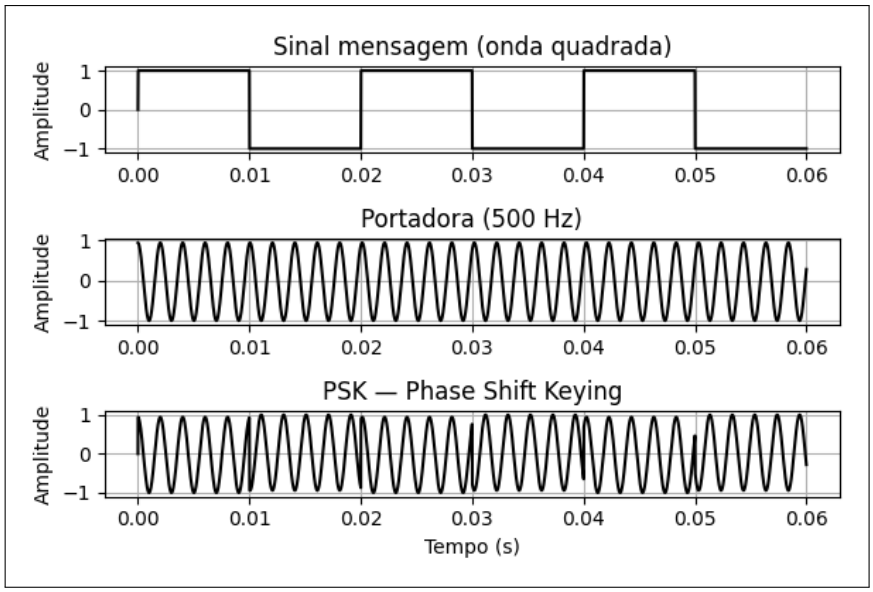
Na Figura 4 são apresentadas as formas de onda esperadas na saída do modulador PSK. Observa-se que o sinal de saída inverte sua fase sempre que o sinal de mensagem muda de nível lógico. Quando o sinal de mensagem assume o valor 1, o sinal de saída apresenta fase 0°, enquanto para o valor 0, o sinal de saída apresenta fase 180°.

2.1.4 Discretização

O oscilador é implementado como um oscilador recursivo baseado na discretização do cosseno. O sinal gerado é descrito por:

$$h[n] = A\cos\left(\omega_0 n + \phi\right) u[n] \tag{2.4}$$

Figura 4 – Forma de onda da modulação PSK.



Fonte: Autoria própria, (2026).

onde A é a amplitude, ω_0 é a frequência angular digital, ϕ é a fase inicial e $u[n]$ é o degrau unitário. Usando a identidade de Euler:

$$\cos(\omega_0 n + \phi) = \frac{1}{2}\left[e^{j(\omega_0 n+\phi)} + e^{-j(\omega_0 n+\phi)}\right] \tag{2.5}$$

A transformada Z bilateral é definida como:

$$H(z) = \sum_{n=-\infty}^{\infty} h[n] z^{-n} \tag{2.6}$$

Substituindo a Equação 2.5 na Equação 2.4 e aplicando a transformada Z unilateral, obtemos:

$$H(z) = \sum_{n=0}^{\infty} A\frac{1}{2}\left[e^{j(\omega_0 n+\phi)} + e^{-j(\omega_0 n+\phi)}\right] z^{-n} \tag{2.7}$$

Separando os termos e fatorando as fases constantes:

$$H(z) = A\frac{1}{2}\left[e^{j\phi}\sum_{n=0}^{\infty}\left(e^{j\omega_0 z^{-1}}\right)^n + e^{-j\phi}\sum_{n=0}^{\infty}\left(e^{-j\omega_0 z^{-1}}\right)^n\right] \tag{2.8}$$

Aplicando a propriedade da série geométrica $\sum_{n=0}^{\infty} a^n = \frac{1}{1-a}$, válida para $|a| < 1$:

$$H(z) = A\frac{1}{2}\left[\frac{e^{j\phi}}{1-e^{j\omega_0 z^{-1}}} + \frac{e^{-j\phi}}{1-e^{-j\omega_0 z^{-1}}}\right] \tag{2.9}$$

Multiplicando numerador e denominador de cada fração por z, combinando as frações e aplicando a identidade $e^{ja} + e^{-ja} = 2\cos(a)$, obtemos a função de transferência:

$$H(z) = \frac{Y(z)}{X(z)} = A\left[\frac{z^2\cos(\phi) - z\cos(\omega_0 - \phi)}{z^2 - 2\cos(\omega_0)z + 1}\right] \tag{2.10}$$

Os polos da Equação 2.10 são $z = e^{\pm j\omega_0}$, que possuem módulo unitário e, portanto, estão exatamente sobre o círculo unitário. Isso caracteriza um sistema marginalmente estável — um oscilador que nem cresce nem decai em amplitude (Alexander; Alexander; Alexander, 2026). Multiplicando a Equação 2.10 por z^{-2}/z^{-2} e rearranjando:

$$Y(z)\left[1 - 2\cos(\omega_0)z^{-1} + z^{-2}\right] = X(z)A\left[\cos(\phi) - \cos(\omega_0 - \phi)z^{-1}\right] \tag{2.11}$$

Aplicando a transformada Z inversa na Equação 2.11:

$$y[n] - 2\cos(\omega_0)y[n-1] + y[n-2] = A\cos(\phi)x[n] - A\cos(\omega_0 - \phi)x[n-1] \tag{2.12}$$

Isolando $y[n]$, obtemos a equação de diferenças que implementa o oscilador recursivo em tempo discreto (Poczekajło; Wirski, 2025):

$$y[n] = A\cos(\phi)x[n] - A\cos(\omega_0 - \phi)x[n-1] + 2\cos(\omega_0)y[n-1] - y[n-2] \tag{2.13}$$

2.1.5 Ponto Fixo

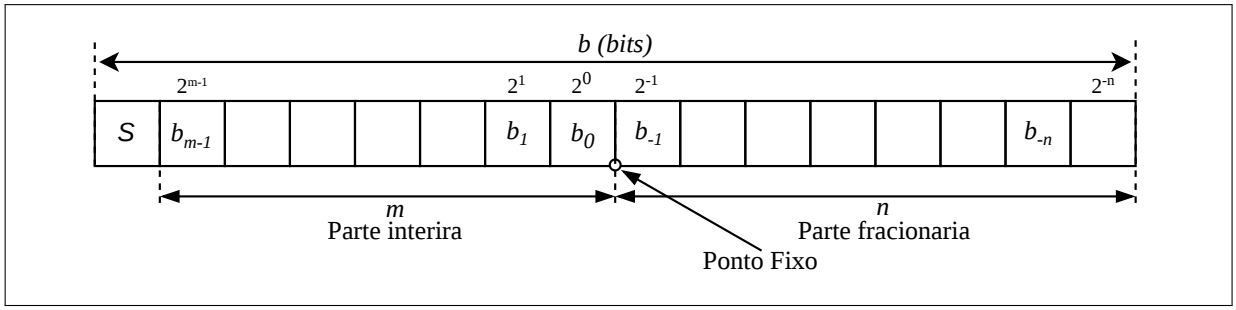
A aritmética de ponto fixo permite que sistemas realizem cálculos utilizando apenas números inteiros, diferentemente da aritmética de ponto flutuante, que representa explicitamente valores fracionários para melhorar a precisão (Menard; Rocher; Sentieys, 2008; Chovatiya; Rutsch; Ecker, 2024). A principal ideia do ponto fixo consiste em manter o ponto binário em uma posição fixa, dividindo os bits do número em duas partes: uma destinada à parte inteira e outra à parte fracionária (Menard; Rocher; Sentieys, 2008; Yates, 2009). Essa característica é justamente a origem do nome “ponto fixo”.

Na representação de ponto fixo é utilizada a notação Q, que descreve quantos bits são reservados para cada parte do número. A representação $Q_{(I,F)}$ define a quantidade de bits

utilizados, onde I corresponde ao número de bits reservados para a parte inteira e F representa a quantidade de bits destinados à parte fracionária (Zoni; Galimberti, 2022).

Um número em ponto fixo possui a estrutura apresentada na Figura 5. Nessa representação de n bits, o bit mais significativo S corresponde ao bit de sinal. Os bits compreendidos entre 2^0 e 2^{m-1} representam a parte inteira do número, enquanto os bits entre 2^{-1} e 2^{-n} correspondem à parte fracionária.

Figura 5 – Representação de um número em ponto fixo $Q_{(7,8)}$



Fonte: Autoria própria, (2026).

Cada bit na posição k possui peso 2^k. Os bits fracionários possuem expoentes negativos: o bit $F-1$ (mais significativo da fração) vale $2^{-1} = 0{,}5$, enquanto o bit menos significativo vale 2^{-n}.

2.1.5.1 Conversão

2.1.5.1.1 Real para ponto fixo

A conversão de números reais para o formato ponto fixo $Q_{(m,n)}$ é realizada multiplicando-se o número pelo fator de escala 2^n e arredondando o resultado para o inteiro mais próximo.

```
int_pf = round(real * 2**n)
```

Exemplo Se quisermos converter o número 12 para a representação $Q_{(7,8)}$, o fator de escala será:

$$2^8 = 256 \tag{2.14}$$

Logo:

$$12 \cdot 256 = 3072_{10} \equiv 0000\ 1100\ 0000\ 0000_2 \tag{2.15}$$

Números negativos utilizam complemento de dois. Assim, para representarmos o número $-3{,}6$ em ponto fixo $Q_{(7,8)}$, temos:

$$3{,}6 \cdot 256 = 921{,}6 \tag{2.16}$$

Arredondando para o inteiro mais próximo:

$$\text{round}(921{,}6) = 922_{10} \equiv 0000\ 0011\ 1001\ 1010_2 \tag{2.17}$$

Para obter o complemento de dois, invertem-se todos os bits e soma-se 1 ao resultado:

$$0000\ 0011\ 1001\ 1010_2 \rightarrow 1111\ 1100\ 0110\ 0101_2 \tag{2.18}$$

$$1111\ 1100\ 0110\ 0101_2 + 1_2 = 1111\ 1100\ 0110\ 0110_2 \tag{2.19}$$

2.1.5.1.2 Ponto fixo para real

A operação inversa consiste apenas em dividir o valor pelo fator de escala 2^n. Para números representados em complemento de dois, o bit mais significativo é o bit de sinal, de modo que o cálculo já leva em consideração o sinal automaticamente.

```
real = int_pf / 2**n
```

2.1.5.2 Operações

2.1.5.2.1 Adição e Subtração

Na adição e na subtração as operações são diretas, seguindo o mesmo princípio das demais representações numéricas: basta somar os inteiros. O único detalhe é que os operandos precisam estar no mesmo formato $Q_{(m,n)}$; caso contrário, é necessário alinhar os valores deslocando-os para a esquerda ou para a direita (*shift*) antes de operar.

A única preocupação nessas operações é o *overflow*, que ocorre quando o resultado excede o intervalo dinâmico da representação. Para evitá-lo, a variável que recebe o resultado deve ser declarada com um formato $Q_{(m_z,n_z)}$ adequado. Para calcular esse novo formato, considera-se que a parte inteira do resultado pode crescer em relação aos operandos, portanto acrescenta-se 1 bit à maior parte inteira. A parte fracionária é mantida com a maior resolução entre os dois operandos, pois um eventual estouro da parte fracionária é absorvido pela parte inteira:

$$m_z = \max(m_x, m_y) + 1 \tag{2.20}$$

$$n_z = \max(n_x, n_y) \tag{2.21}$$

Observação: Ao somar dois números no formato $Q_{(m,n)}$, o resultado correto requer o formato $Q_{(m+1,n)}$, que possui um bit adicional na parte inteira (*guard bit*) para evitar *overflow*. Caso seja necessário manter a mesma quantidade de bits do formato original, existem duas alternativas, ambas com custo: descartar o *guard bit*, retornando ao formato $Q_{(m,n)}$ porém com risco de *overflow* caso o resultado ultrapasse o intervalo representável; ou realizar um *shift right*, descartando o *Least Significant Bit* (LSB) fracionário e obtendo o formato $Q_{(m+1,n-1)}$, que possui a mesma quantidade de bits do original porém com parte inteira maior e resolução fracionária reduzida em 1 bit.

2.1.5.2.2 Multiplicação

Na multiplicação, os inteiros são simplesmente multiplicados entre si. Entretanto, o resultado exige um formato maior para acomodar o produto sem perda de informação. O novo formato é calculado da seguinte forma:

$$m_z = m_x + m_y + 1 \tag{2.22}$$

$$n_z = n_x + n_y \tag{2.23}$$

A parte fracionária do resultado é obtida somando-se as partes fracionárias dos dois operandos, pois ao multiplicar dois inteiros escalados por 2^{n_x} e 2^{n_y} o resultado fica escalado

por $2^{n_x+n_y}$. A parte inteira é calculada somando-se as partes inteiras dos dois operandos e adicionando 1 bit extra para evitar *overflow*.

Um detalhe importante é que o resultado terá um formato maior do que os operandos originais. Isso pode não ser adequado para sistemas com recursos limitados, como FPGAs. Nesses casos, é necessário deslocar o resultado *n* posições à direita (*shift right*), descartando os bits menos significativos e retornando ao formato original.

2.1.5.2.3 Divisão

Na divisão, o resultado também requer um novo formato $Q_{(m_z,n_z)}$, cujas partes são calculadas como:

$$m_z = m_x + n_y \tag{2.24}$$

$$n_z = n_x + m_y \tag{2.25}$$

Esses valores são necessários para acomodar os dois casos extremos da divisão: quando o numerador é maior que o divisor, o resultado possui parte inteira maior; quando o numerador é menor que o divisor, o resultado é estritamente fracionário. O formato calculado acima é suficiente para representar corretamente qualquer divisão entre dois operandos nesses formatos.

Assim como na multiplicação, essa operação também exige um deslocamento à direita de *n* posições para retornar ao formato original, descartando os bits menos significativos.

2.1.5.2.4 *Shift*

A operação de *shift* consiste em deslocar os bits do valor para a esquerda $(<<)$ ou para a direita $(>>)$. O deslocamento à esquerda multiplica o valor por uma potência de dois, enquanto o deslocamento à direita divide por uma potência de dois. Essa operação é muito utilizada para alterar a resolução ou a escala do sinal. A única desvantagem é a perda de precisão que ocorre ao deslocar para a direita e descartar os bits menos significativos.

2.1.6 Conversor DAC

2.1.6.1 Sigma-Delta (Σ-Δ)

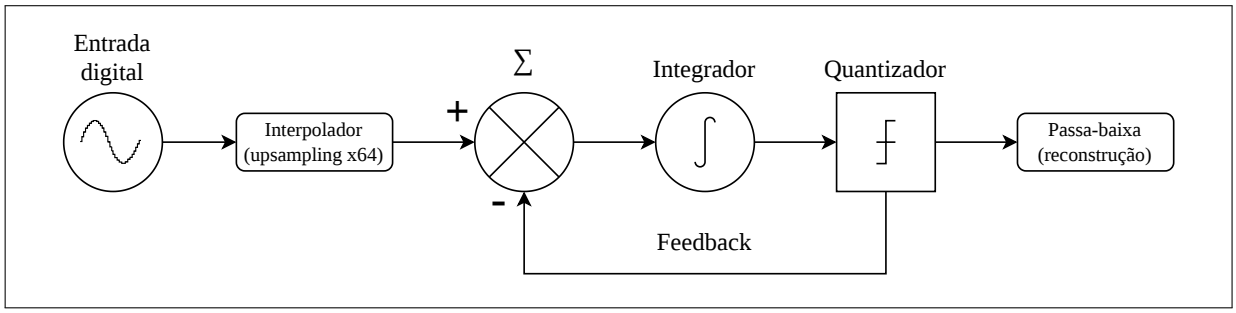
Diferentemente dos conversores convencionais, o *Digital-to-Analog Converter* (DAC) sigma-delta opera com uma frequência de amostragem significativamente superior à frequência mínima necessária para representar um sinal, definida pelo critério de Nyquist (Aboutabikh; Shokyfeh; Garib, 2024; Aziz; Sorensen; Spiegel, 1996). A frequência de Nyquist corresponde à metade da frequência de amostragem do sistema. Dessa forma, o DAC utiliza a técnica de sobreamostragem (*oversampling*), que distribui o ruído de quantização por uma faixa espectral mais ampla, reduzindo sua densidade na banda de interesse (Engelen; Plassche, 2013; Kester, 2008). Além disso, o DAC sigma-delta emprega realimentação para deslocar grande parte do ruído de quantização para frequências mais elevadas, processo conhecido como modelagem de ruído (*noise shaping*) (Kulka; Woszczek, 2008). Após essa etapa, um filtro passa-baixa pode ser utilizado para atenuar o ruído fora da banda útil do sinal(Kester, 2008; Logue, 1999).

O DAC sigma-delta é uma técnica de conversão ideal para dispositivos com recursos de *hardware* limitados como as FPGA (Logue, 1999). Eles também permitem a implementação de moduladores de ordem superiores em FPGA, sem perder a estabilidade e fidelidade do sinal modulado (Kulka; Woszczek, 2008).

2.1.6.2 Funcionamento

A Figura 6 ilustra a representação de um conversor sigma-delta. Abaixo será detalhado cada um dos passos que tem no DAC.

Figura 6 – DAC $(\Sigma - \Delta)$



Fonte: Autoria própria, (2026).

Entrada Digital

Nessa etapa, o sinal de entrada é uma palavra digital de 16 *bits*, representando a amplitude do sinal a ser convertido em cada instante de tempo.

Oversampling

Nessa etapa, o sinal passa por um interpolador que aumenta a taxa de amostragem por um fator igual ao *Oversampling Ratio* (OSR), neste caso 64. Ao operar em uma taxa muito superior à mínima necessária, o ruído de quantização — que é fixo em energia total — se redistribui por uma banda de frequência muito mais larga, reduzindo sua densidade espectral dentro da banda de interesse.

Nó somador

Nessa etapa, o sistema calcula a diferença entre o sinal de entrada e o sinal de realimentação, gerando o erro de quantização que será processado pelo integrador.

Integrador

O erro de quantização é acumulado sucessivamente pelo integrador. Essa acumulação contínua do erro é o mecanismo responsável pela moldagem espectral do ruído (*noise shaping*), que empurra o ruído de quantização para as altas frequências, fora da banda de interesse.

Quantizador

O valor acumulado pelo integrador é comparado com um limiar: caso seja positivo, a saída é 1; caso seja negativo, a saída é 0. Isso gera um fluxo binário de 1 *bit* na saída da FPGA, cuja densidade de pulsos representa a amplitude do sinal original.

Realimentação

A saída do quantizador é subtraída do sinal de entrada no nó somador. Esse laço de realimentação negativa força o modulador a corrigir continuamente os erros de quantização a cada ciclo de *clock*.

Filtro passa-baixas

O fluxo binário gerado pela FPGA é então processado por um filtro passa-baixas, que reconstrói o sinal analógico a partir da média dos bits. Por ter frequência de corte inferior às frequências para as quais o ruído foi moldado, o filtro elimina simultaneamente o ruído de quantização, entregando um sinal analógico limpo na saída.

3 TRABALHOS RELACIONADOS

Para a geração das formas de onda necessárias às modulações ASK, FSK e PSK, (Zubaidy; Qaddoori; Gadawe, 2023) propõe uma metodologia otimizada baseada em tabelas de busca (*Look-Up Table* (LUT)) com apenas 256 amostras para a geração da onda portadora senoidal. Essa abordagem minimiza o uso de memória e permite o chaveamento em tempo real entre onze tipos de modulação — incluindo ASK, FSK, PSK, *Quadrature Phase Shift Keying* (QPSK), 8PSK, 8*Quadrature Amplitude Modulation* (QAM) e 16QAM — em um único sistema implementado na FPGA Spartan 3E, com apenas 534 LUTs de 4 entradas e ocupação máxima de 6% dos slices disponíveis no dispositivo. A saída analógica é obtida por meio de um conversor DAC de 8 bits projetado especificamente para o sistema. Esse trabalho é relevante para o presente projeto pois demonstra que é possível implementar múltiplos esquemas de modulação digital em FPGA de baixo custo com consumo mínimo de recursos, o que justifica a abordagem adotada neste projeto.

Em (Logue, 1999), demonstra-se que o uso de DACs Sigma-Delta é uma solução de custo extremamente reduzido para FPGAs, exigindo *hardware* interno mínimo: um DAC de 8 bits consome apenas 5 slices, enquanto um de 10 bits utiliza apenas 6 slices, com frequência máxima de operação de até 219 MHz e 215 MHz, respectivamente, na família Virtex. O único circuito externa necessária é um filtro passa-baixa composto por um resistor e um capacitor. Diferente dos DACs convencionais de soma de corrente, que sofrem com variações térmicas à medida que o número de bits aumenta, a arquitetura Sigma-Delta utiliza técnicas digitais e retroalimentação de um bit para gerar um fluxo de pulsos cujo ciclo de trabalho médio é proporcional ao valor binário de entrada, garantindo imunidade a variações de temperatura e alta linearidade. Essa característica torna o DAC Sigma-Delta particularmente adequado para geração de formas de onda analógicas a partir de valores pré-calculados, como é o caso da saída analógica dos moduladores implementados no presente projeto.

O trabalho de (Zoni; Galimberti, 2022) apresenta uma comparação detalhada entre diferentes arquiteturas aritméticas para sistemas embarcados baseados na *Instruction Set Architecture* (ISA) RISC-V: ponto flutuante, ponto fixo utilizando a extensão padrão M e ponto fixo utilizando a extensão Zm proposta pelos autores. A motivação do estudo reside na ineficiência de plataformas de 32 bits ao executarem operações de ponto fixo em unidades inteiras convencionais, uma única multiplicação em ponto fixo pode exigir até cinco instruções (incluindo *shifts* e operações lógicas), enquanto a divisão pode demandar centenas de ciclos de clock devido à necessidade

de rotinas de *software* para divisões de 64 bits. Para mitigar esse gargalo, os autores propõem a extensão Zm, que introduz instruções dedicadas de multiplicação e divisão em ponto fixo com latências reduzidas (5 e 14 ciclos, respectivamente), valores que se aproximam do desempenho de FPUs de *hardware*. Os resultados experimentais demonstram que o sistema em ponto flutuante (binary32) utiliza 32% mais recursos de área (LUTs) do que o sistema base, enquanto a extensão Zm apresenta um *overhead* de apenas 4%. Em termos de eficiência energética, a abordagem baseada apenas em unidades inteiras (extensão M) apresentou um *Energy-Delay Product* (EDP) normalizado de 1,796, comparado a apenas 0,651 do sistema Zm, mantendo uma perda de precisão média insignificante de 0,0003% em relação ao padrão de ponto flutuante. Este estudo é altamente relevante para o presente projeto, pois valida tecnicamente a escolha da aritmética de ponto fixo em plataformas de recursos limitados, como as FPGAs. A evidência de que é possível obter alta precisão com um consumo de energia e área significativamente menores reforça a viabilidade da implementação dos moduladores ASK, FSK e PSK em *hardware* reconfigurável, especialmente quando associados a técnicas de economia de recursos como o DAC Sigma-Delta

As operações aritméticas em ponto fixo são amplamente utilizadas em circuitos digitais, porém seu uso em FPGAs ainda é pouco explorado na literatura, sendo comum que os projetistas se limitem aos operadores nativos das linguagens *Hardware Description Language* (HDL). O trabalho (Bečvář; Štukjunger, 2005) investiga essa lacuna ao comparar diferentes estruturas de unidades aritméticas de ponto fixo implementadas em FPGA, tendo como critérios de avaliação a área ocupada em slices e o atraso de propagação. Para as operações de adição, foram testadas estruturas de 16, 32 e 64 bits, sendo que o somador gerado pela ferramenta de síntese (Generated Adder) obteve os melhores resultados em todas as larguras de palavra, tanto em área quanto em atraso, devido ao uso das cadeias de carry dedicadas da FPGA Xilinx Virtex-II. Para as multiplicações, as estruturas foram divididas em dois modos: serial e paralelo. Os resultados mostram que os multiplicadores no modo serial consomem significativamente menos área do que os no modo paralelo, sendo vantajosos em aplicações que exigem uso compacto de recursos. Por outro lado, os multiplicadores paralelos apresentaram menor tempo de resposta, porém para operandos de 64 bits, o consumo de área se aproxima dos limites do dispositivo. Esse trabalho é relevante para o presente projeto pois orienta a escolha das estruturas aritméticas de ponto fixo a serem utilizadas na implementação dos moduladores em FPGA, onde o compromisso entre área e desempenho é uma restrição fundamental.

Dispositivos que empregam tecnologia de circuitos integrados de aplicação específica

(*Application-Specific Integrated Circuit* (ASIC)s) são utilizados em muitos casos para resolver problemas específicos. No entanto, uma desvantagem apresentada é a impossibilidade de modificar o algoritmo embarcado após a fabricação. Para contornar essa limitação, o trabalho (Przybył, 2021) propõe o uso de FPGAs, que se destacam por permitir o ajuste flexível de sua estrutura de *hardware* para implementar diferentes algoritmos, além de apresentarem menor consumo energético e, em alguns casos, desempenho superior ao de ASICs. O artigo apresenta uma unidade de processamento em ponto fixo superescalar (*Fixed-point processing Unit* (FXU)), dotada de um mecanismo de escalonamento integrado que determina o formato dos números em tempo de compilação, e não durante a execução, simplificando e acelerando o processamento em tempo real. Os testes foram realizados com um algoritmo de controle vetorial de motor síncrono de ímã permanente (*Permanent Magnet Synchronous Motor* (PMSM)) implementado em uma FPGA Xilinx Spartan-6. Os resultados mostraram que a FXU operou a uma frequência de até 80 MHz, consumindo apenas 14% dos LUTs disponíveis, e executou os algoritmos de controle cerca de dez vezes mais rápido do que a implementação equivalente em um processador *soft-core* convencional, reduzindo o tempo total de processamento de mais de 65 µs para menos de 20 µs. Esse trabalho é relevante para o presente projeto pois demonstra a viabilidade e a eficiência do uso de aritmética em ponto fixo em FPGAs de baixo custo, reforçando a escolha dessa abordagem para a implementação dos moduladores ASK, FSK e PSK.

Visando unificar múltiplas técnicas de modulação em *hardware* reconfigurável, (Mohammed; Abdullah, 2020) exploraram a versatilidade das FPGAs frente aos processadores *Digital Signal Processor* (DSP) convencionais ao implementar oito esquemas distintos — cinco digitais (*Binary Amplitude Shift Keying* (BASK), *Binary Frequency Shift Keying* (BFSK), *Binary Phase Shift Keying* (BPSK), *Differential Phase Shift Keying* (DPSK) e QPSK) e três analógicos (*Amplitude Modulation* (AM), *Frequency Modulation* (FM) e *Phase Modulation* (PM)) — em uma única plataforma Artix-7 (Nexys 4 DDR), utilizando o Xilinx System Generator integrado ao Vivado. Os resultados de *hardware* evidenciaram uma disparidade expressiva no consumo de recursos: enquanto as técnicas digitais BASK e QPSK ocuparam apenas 0,07% das LUTs disponíveis, as modulações analógicas demandaram área significativamente maior, com FM exigindo 6,59% e PM alcançando 12,06%, diferença atribuída à maior quantidade de multiplicadores necessários para representar estruturas analógicas em *hardware* digital. Apesar da abrangência das técnicas implementadas, o sistema limitou-se ao domínio digital, com saídas restritas a ferramentas de simulação e LEDs da placa, sem conversão analógica física dos sinais modulados. O presente

trabalho avança nesse aspecto ao incorporar um DAC sigma-delta com aritmética de ponto fixo diretamente na arquitetura da FPGA, viabilizando a geração física da portadora modulada e a validação experimental do sistema em tempo real.

Complementarmente, (Corrêa Jr *et al.*, 2018) destaca que a viabilidade de transmissões em radiofrequência em *hardware* reconfigurável advém do paralelismo inerente às FPGAs. Enquanto processadores sequenciais programados em linguagem C possuem limitações quanto à frequência e ao número de bits de saída, a arquitetura em FPGA permite a execução de múltiplas operações matemáticas simultâneas, superando os gargalos de largura de banda. No projeto proposto pelos autores, implementado na FPGA Altera DE2-115 com linguagem Verilog, o algoritmo gera oito pontos de saída simultaneamente por ciclo, permitindo atingir taxas de símbolo de até 6,25 MS/s com largura de banda básica de até 3 MHz. O sistema suporta quatro tipos de modulação — 4-QAM, 16-QAM, 8-PSK e 16-PSK — com filtro de saída do tipo cosseno levantado e seleção configurável do fator *roll-off*. Os resultados experimentais demonstraram um erro vetorial de magnitude (*Error Vector Magnitude* (EVM)) de 4,56%, superando o desempenho de 10,03% obtido pelo kit comercial Dreamcatcher ME1100 utilizado como referência. Essa capacidade de processamento paralelo aliada ao baixo custo de implementação reforça a viabilidade do uso de FPGAs para a geração de modulações digitais no presente projeto.

Os Rádios Definidos por *Software* (*Software Defined Radio* (SDR)) são definidos por (Bimbi Jr; Oliveira; Júnior, 2015) como sistemas de comunicação de rádio nos quais componentes tradicionalmente implementados em *hardware* — como filtros digitais, moduladores e demoduladores — passam a ser implementados em *software*, utilizando computadores pessoais ou sistemas embarcados. Essa abordagem é particularmente eficiente em plataformas de lógica programável, como as FPGAs, que são responsáveis pelo tratamento digital do esquema de modulação e pela interação com os conversores *Analog-to-Digital Converter* (ADC) e DAC conectados ao *front end* de radiofrequência. A flexibilidade de reconfiguração dessas plataformas permite adaptar o sistema a diferentes padrões de modulação e condições de operação, tornando o SDR uma solução versátil tanto em aplicações comerciais quanto acadêmicas. Nesse contexto, a implementação de conversores digital-analógicos integrados torna-se um desafio devido às restrições de área e consumo energético do dispositivo.

4 METODOLOGIA

4.1 Sistema proposto

Neste trabalho, o oscilador proposto consiste em criar uma sistema cuja a resposta ao impulso é o sinal senoidal desejado, ou seja:

$$h[n] = A\cos(\omega_0 n + \phi)u[n] \tag{4.1}$$

em que ω_0 é a frequência angular do sinal cossenoidal e ϕ é a fase inicial. A partir dessa definição, é possível obter uma equação em diferenças que descreve o sistema, a qual pode ser implementada em FPGA. Demonstra-se que a equação em diferenças que descreve o sistema é dada por:

$$y[n] = A\cos(\phi)x[n] - A\cos(\omega_0 - \phi)x[n-1] + 2\cos(\omega_0)y[n-1] - y[n-2] \tag{4.2}$$

Assim sendo, quando $x[n] = \delta[n]$, a saída $y[n]$ do sistema é o sinal cossenoidal desejado, ou seja, $y[n] = A\cos(\omega_0 n + \phi)u[n]$.

A implementação do oscilador digital é realizada utilizando aritmética em ponto fixo $Q_{(2,14)}$. Esse formato foi escolhido com base em testes realizados durante o desenvolvimento do projeto, tendo sido determinado como o que oferece a melhor resolução para a aplicação. Os coeficientes do sistema são calculados por meio de um *script* em *Python*, e posteriormente convertidos para o formato de ponto fixo para implementação em *SystemVerilog*. Por fim, a saída é convertida para o domínio analógico utilizando um conversor sigma-delta seguido de um filtro passa-baixas ativo, conforme apresentado na Figura 7.

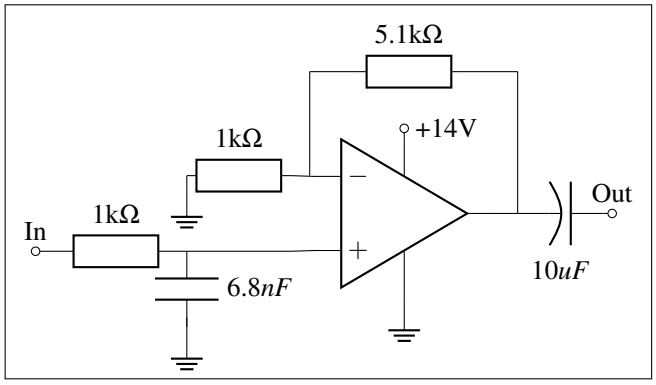
O filtro passa-baixas possui frequência de corte de aproximadamente 23,4 kHz, valor calculado a partir dos componentes $R = 1\,\text{k}\Omega$ e $C = 6{,}8\,\text{nF}$, conforme a Equação 4.3:

$$f_c = \frac{1}{2\pi RC} \tag{4.3}$$

Essa frequência de corte é mais que suficiente para a maior frequência utilizada no sistema, de 10 kHz. A amplificação do sinal foi realizada por meio de um amplificador operacional LM358 em configuração não inversora, resultando em um ganho de 6,1 vezes, conforme mostrado na Equação 4.4:

$$G = 1 + \frac{R_f}{R_1} = 1 + \frac{5{,}1\,\text{k}\Omega}{1\,\text{k}\Omega} = 6{,}1 \tag{4.4}$$

Figura 7 – Circuito analógico.



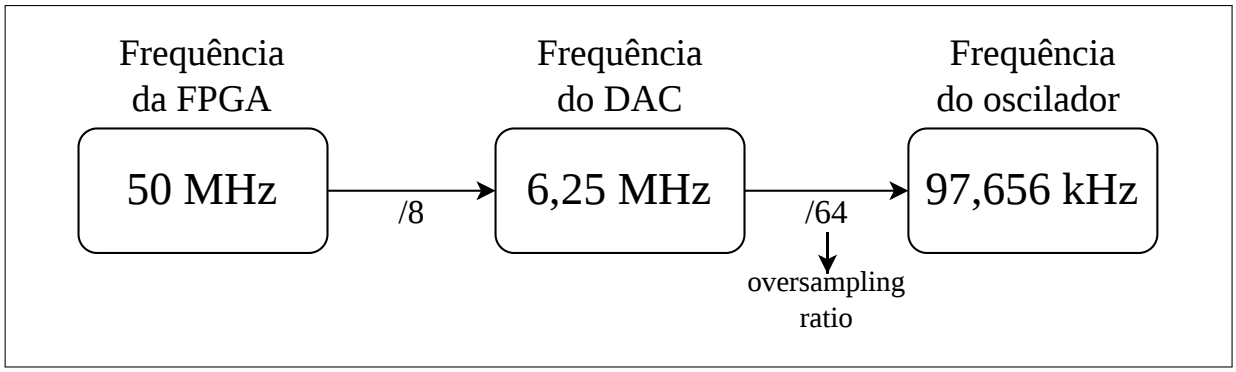
Fonte: Autoria própria, (2026).

4.2 Frequência

Para o oscilador, foi projetado inicialmente para operar a uma frequência de amostragem de 100 kHz. Como o modulador sigma-delta opera com sobreamostragem, a frequência do DAC deve ser superior à do oscilador por um fator igual ao *oversampling ratio*. Assim, multiplicando a frequência de amostragem de 100 kHz pelo fator de 64, obtém-se a frequência de operação do DAC de 6,4 MHz.

Como a FPGA, modelo EP4CE115F29C7, opera a uma frequência fixa de 50 MHz, não é possível gerar diretamente as frequências calculadas. Para contornar isso, foram implementados divisores de frequência a fim de aproximar os valores obtidos dos ideais, resultando no sistema apresentado na Figura 8.

Figura 8 – Divisores de frequência aplicados ao sistema DAC sigma-delta.



Fonte: Autoria própria, (2026).

Para obter a frequência do DAC mais próxima do valor calculado, a frequência base foi dividida por 8, resultando em 6,25 MHz. Em seguida, aplicou-se o fator de divisão correspondente ao *oversampling ratio* de 64, obtendo 97,656 kHz como frequência de operação do oscilador, valor que representa um desvio de aproximadamente 2,3% em relação ao projeto original de 100 kHz. Esse valor foi adotado como frequência de amostragem do oscilador e utilizado para o cálculo dos parâmetros do módulo implementado em SystemVerilog, conforme descrito na seção seguinte.

5 RESULTADOS

5.1 Simulação

Após a fase de projeto, foi realizada a simulação no *software* ModelSim. Para simular o clock de 50 MHz da FPGA, foi o *'timescale* de 1ns/1ps, com isso bastou converter o clock em ns para termos a mesma base.

$$CLK\_PERIOD = \frac{1}{50\,\text{MHz}} = 20\,\text{ns} \tag{5.1}$$

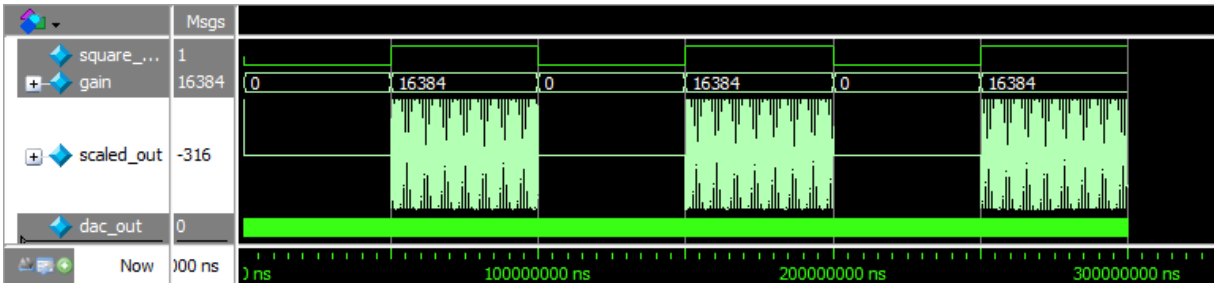
Com o valor do clock calculado pela Eq. 5.1, configuramos o *testbanch* para operar nessa frequência. Após isso foi calculado qual periodo de simulação para 3 mudanças de onda. Para isso, pegamos o *clock* de 10 Hz da onda quadrada calculamos o periodo que presisaria para representar 3 mudanças dechaveamento. Com a Equação 5.2 descobrimos que para cada ciclo completo precisariamos de 100 ms, como iriamos representar 3 ciclos, configuramos o simulador para 300 ms.

$$T = \frac{1}{10\,\text{Hz}} = 100\,\text{ms} \tag{5.2}$$

5.1.1 Simulação ASK

Na Figura 9, o sinal mostrado é exatamente o que esperávamos da modulação ASK. A modulação conseguiu recriar perfeitamente o sinal, com a característica do sinal de final ser igual a 0 quando o sinal mensagem for nulo, e quando o sinal da mensagem foi 1, o sinal de final assume a amplitude da portadora.

Figura 9 – Simulação da modulação ASK.

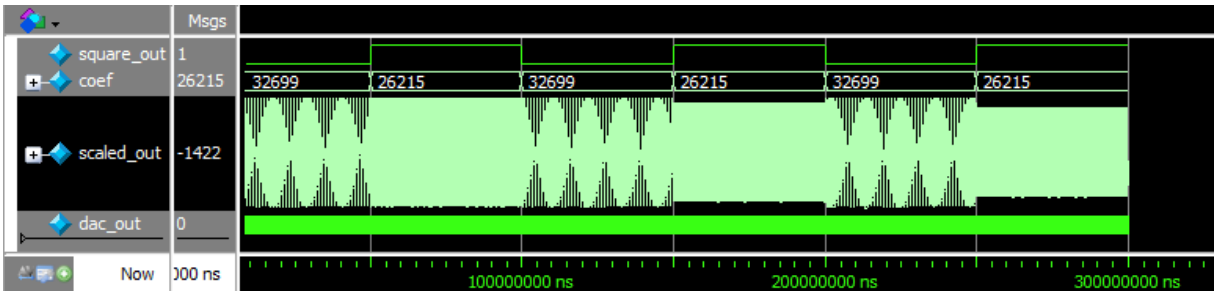


Fonte: Autoria própria, (2026)

5.1.2 Simulação FSK

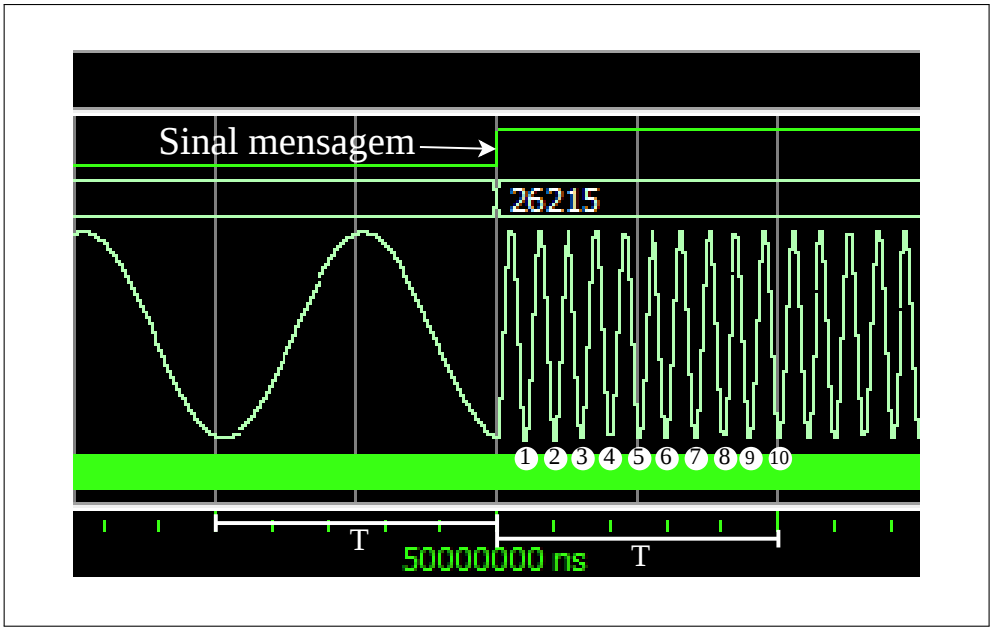
A simulação do FSK foi realizada com uma frequência de 1 kHz e 10 kHz. A Figura 10 mostra a simulação de 3 ciclos de 100 ms cada com a variação do sinal mensagem de 0 para 1. Na Figura 11 temos a ampliação das ondas do FSK, onde podemos contar os períodos das ondas quando o sinal mensagem transaciona de 0 para 1, comprovando que a segunda frequência é 10 x maior que a outra num período T. Isso comprovou que a implementação do FSK foi bem sucedida.

Figura 10 – Simulação da modulação FSK



Fonte: Autoria própria, (2026).

Figura 11 – Zoom da modulação FSK

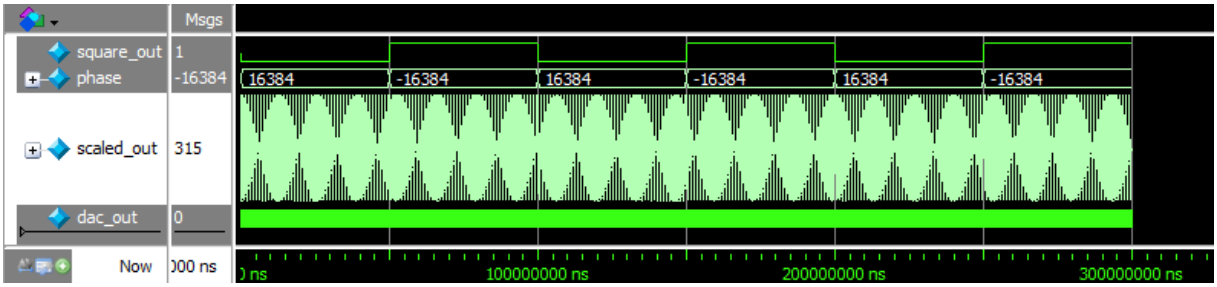


Fonte: Autoria própria, (2026).

5.1.3 Simulação PSK

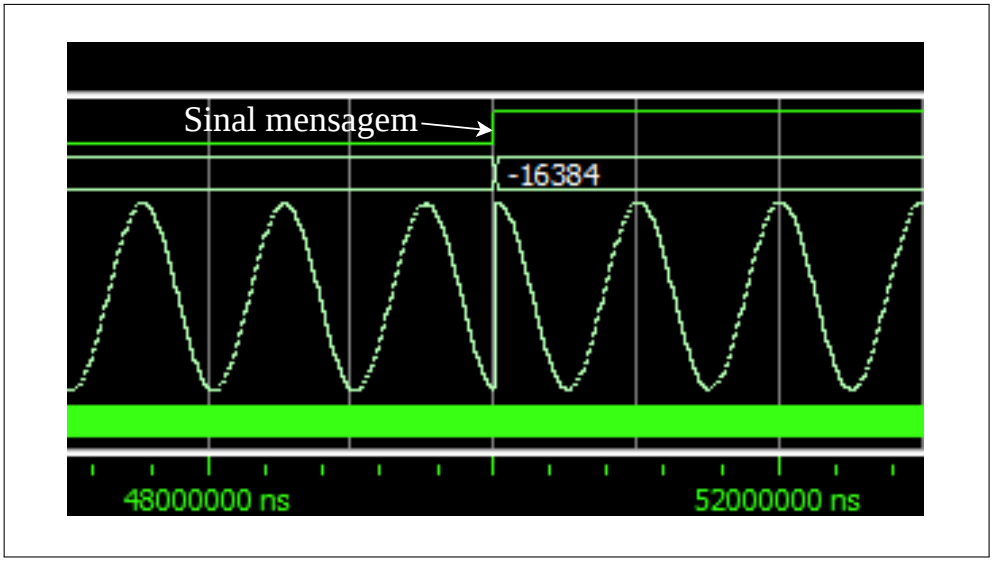
O PSK foi simulado com a configuração de 0° para quando a mensagem tiver valor 0 e 180° para quando a mensagem tiver valor 1. Na Figura 12 temos os três ciclos de onda de 100 ms. A Figura 13 evidencia a transação de phase da senoide quando o sinal mensagem troca do valor 0 para o 1.

Figura 12 – Simulação da modulação PSK.



Fonte: Autoria própria, (2026).

Figura 13 – Zoom da modulação PSK.



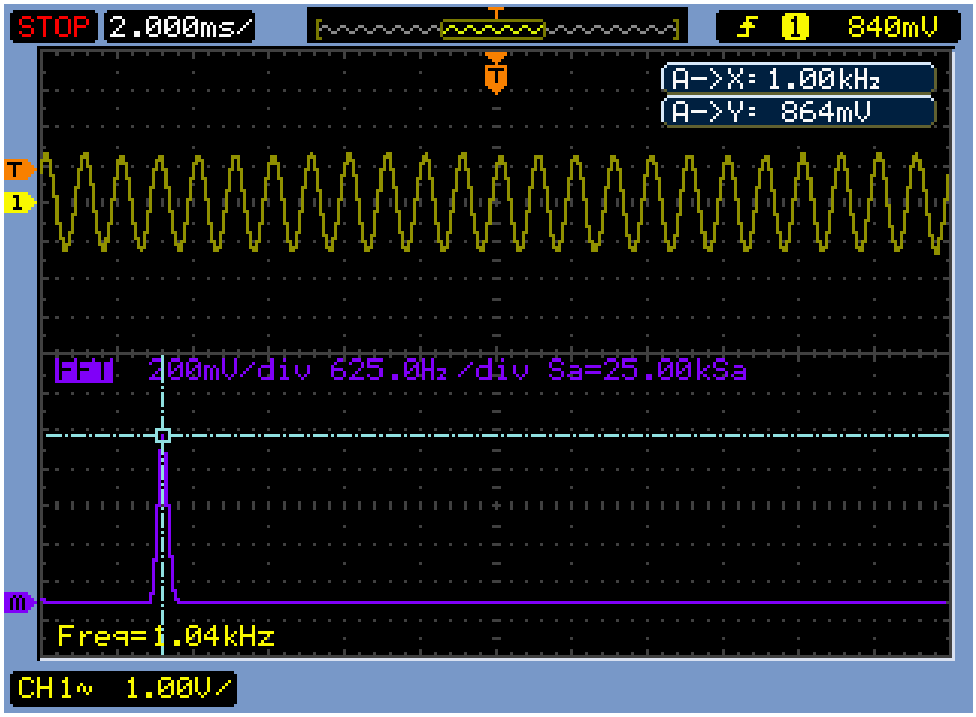
Fonte: Autoria própria, (2026).

A simulação serviu para atestar a efetividade da implementação em FPGA, evidenciando o baixo ruido e a alta precisão obtida na saída do modulador.

5.2 Implementação

Inicialmente verificou-se o desempenho do oscilador. Para isso, implementou-se na FPGA (kit DE2-115) um sistema para gerar um cosseno de frequência de 1 kHz, obtendo o resultado apresentado na Figura 14. Conforme observado, o sistema representou o sinal cossenoidal com sucesso, o que pode ser verificado pela *Fast Fourier Transform* (FFT), cujo pico ocorre exatamente em 1 kHz. A frequência medida pelo osciloscópio foi de 1,04 kHz, apresentando um desvio de 0,04 kHz em relação ao valor esperado, o que pode ser atribuído à resolução da aritmética em ponto fixo utilizada na implementação.

Figura 14 – Onda e espectro do cosseno de 1 *kHz*.

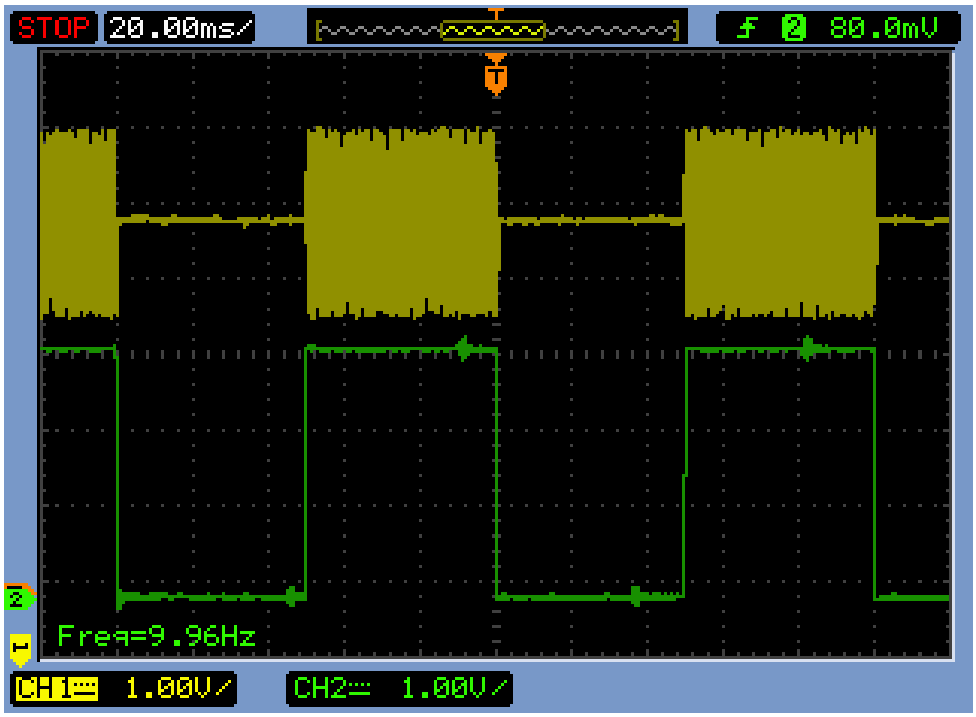


Fonte: Autoria própria, (2026).

5.2.1 Implementação ASK

Após obter sucesso na geração do cosseno, realizou-se o teste do modulador ASK, cujos resultados estão apresentados na Figura 15. O sinal modulante foi gerado de forma sintética na própria FPGA. Nota-se que a modulação foi representada com sucesso, exibindo a portadora cossenoidal nos intervalos em que o sinal modulante assume valor 1, e ausência de sinal nos intervalos em que assume valor 0.

Figura 15 – Modulação ASK.



Fonte: Autoria própria, (2026).

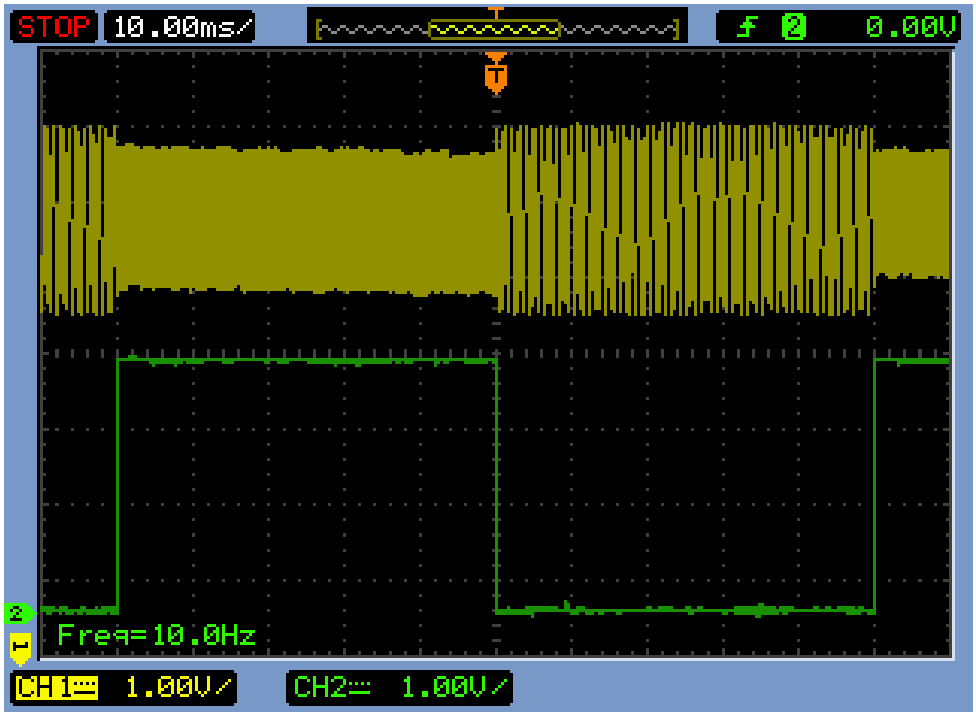
5.2.2 Implementação FSK

Por sua vez, o sinal obtido na modulação FSK é representado por duas frequências distintas, uma de 1 kHz e outra de 10 kHz, conforme apresentado na Figura 16. Observa-se que a frequência da portadora varia entre os dois valores de acordo com o nível do sinal modulante, assumindo 10 kHz quando o modulante é 1 e 1 kHz quando o modulante é 0. Nota-se que a amplitude das senoides foi diferente, fato que pode ser explicado por erros na representação numérica em ponto fixo.

5.2.3 Implementação PSK

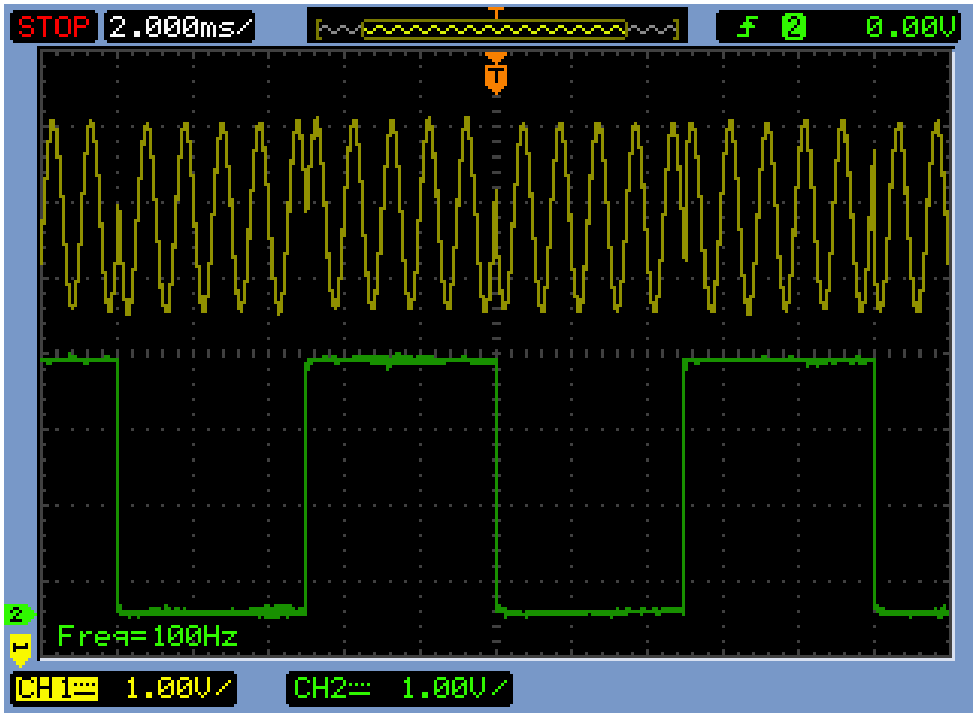
Por fim, a modulação PSK foi reconstruída alternando a fase do sinal entre 0° e 180°. A Figura 17 ilustra a saída dessa modulação, onde observa-se a portadora chaveando em resposta ao sinal modulante, com a inversão de fase ocorrendo sempre que o sinal modulante realiza uma transição entre os níveis lógicos.

Figura 16 – Modulação FSK.



Fonte: Autoria própria, (2026).

Figura 17 – Modulação PSK.



Fonte: Autoria própria, (2026).

Os resultados da implementação mostraram-se fiéis aos obtidos na simulação, apresentando apenas um ruído residual que pode ser atribuído a fatores como interferência eletromagnética, tolerâncias dos componentes do circuito ou imprecisões na montagem. Vale ressaltar que, na simulação, os valores foram coletados na saída do módulo de reescala (*scaled_out*) e não na saída do módulo sigma-delta (*dac_out*). Isso foi necessário porque a saída do modulador sigma-delta consiste em um fluxo de bits de alta frequência, cuja conversão para tensão analógica requer um filtro passa-baixas externo, não presente na simulação digital.

6 CONCLUSÕES E TRABALHOS FUTUROS

A implementação realizada neste trabalho demonstrou resultados satisfatórios para as três modulações, com sinais gerados de forma limpa e com boa precisão. A representação numérica por meio de aritmética em ponto fixo mostrou-se eficiente para este tipo de sistema, apresentando baixo erro nas operações realizadas, embora que a modulação FSK tenha apresentado um pequeno desvio em relação a amplitude das senoides geradas. Essa característica é especialmente relevante em sistemas de tempo real implementados em FPGA, nos quais a aritmética em ponto flutuante impõe custos elevados de área e consumo energético.

O conversor digital-analógico sigma-delta apresentou desempenho satisfatório na conversão do sinal digital para o domínio analógico. Contudo, verificou-se que esse tipo de conversor introduz um *offset* DC elevado na saída, que se torna mais pronunciado após a etapa de amplificação. O capacitor de desacoplamento de 10 μF utilizado não foi suficiente para eliminar completamente esse *offset*. Como solução, recomenda-se a substituição por um circuito de acoplamento CA posicionado após o amplificador operacional, o que permitiria remover o *offset* e centralizar o sinal em torno de 0 V.

Como trabalhos futuros, propõe-se a implementação de um demodulador correspondente a cada esquema de modulação, bem como a avaliação do desempenho do sistema em canais com ruído, visando caracterizar a taxa de erro de *bit* (*Bit Error Rate* (BER)) em condições práticas de transmissão.

REFERÊNCIAS

ABOUTABIKH, K.; SHOKYFEH, A.-A.; GARIB, A. Design and implementation of a digital amplitude shift keying (ask), frequency shift keying (fsk), binary phase shift keying (bpsk) and quadrature phase shift keying (qpsk) modulators using fpga. **International Multidisciplinary Research Journal Reviews (IMRJR)**, v. 1, n. 1, Setembro 2024.

ALEXANDER, R.; ALEXANDER, J.; ALEXANDER, A. **A Direct Form 2 Digital Filter Algorithm for Circle Rasterization**. 2026. TechRxiv. Disponível em: https://doi.org/10.36227/techrxiv.177154102.29020090/v1. Preprint.

AZIZ, P. M.; SORENSEN, H. V.; SPIEGEL, J. Vn der. An overview of sigma-delta converters. **IEEE signal processing magazine**, IEEE, v. 13, n. 1, p. 61–84, 1996.

BABU, P.; PARTHASARATHY, E. Reconfigurable fpga architectures: A survey and applications. **Journal of The Institution of Engineers (India): Series B**, Springer, v. 102, n. 1, p. 143–156, 2021.

BEČVÁŘ, M.; ŠTUKJUNGER, P. Fixed-point arithmetic in fpga. **Acta Polytechnica**, v. 45, n. 2, 2005.

BIMBI JR, S.; OLIVEIRA, V. C. d.; JÚNIOR, G. B. Rádios definidos por software com aplicações GNU Radio. In: **SET EXPO PROCEEDINGS**. [S.l.: s.n.], 2015. SET - Brazilian Society of Television Engineering.

BOMFIN, R.; CHAFII, M.; FETTWEIS, G. A novel modulation for iot: Psk-lora. In: IEEE. **2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)**. [S.l.], 2019. p. 1–5.

CHOVATIYA, V.; RUTSCH, G.; ECKER, W. Addressing fixed-point format issues in fpga prototyping with an open-source framework. In: **Design and Verification Conference Europe (DVCon Europe)**. Munich, Germany: [s.n.], 2024.

CORRÊA JR, L. A. *et al.* Modulador digital educacional em FPGA para uso em aulas de laboratório de telecomunicações. **Revista Principia - Divulgação Científica e Tecnológica do IFPB**, n. 39, 2018.

ENGELEN, J. V.; PLASSCHE, R. J. van de. **Bandpass sigma delta modulators: stability analysis, performance and design aspects**. [S.l.]: Springer Science & Business Media, 2013.

FÖDISCH, P. *et al.* Implementing high-order fir filters in fpgas. **arXiv preprint arXiv:1610.03360**, 2016.

ISLAM, M. *et al.* Modulation technique for software defined radio application. **Australian Journal of Basic and Applied Sciences**, INSInet Publications, v. 3, n. 3, p. 1780–1785, 2009.

KESTER, W. Adc architectures iii: Sigma-delta adc basics. **Analog Devices, MT022**, 2008.

KULKA, Z.; WOSZCZEK, P. Implementation of digital sigma-delta modulators for high-resolution audio... **Archives of Acoustics**, v. 33, n. 1, p. 93–101, 2008.

LOGUE, J. **Virtex Synthesizable Delta-Sigma DAC**. [S.l.], 1999. Application Note XAPP154, v1.1.

MENARD, D.; ROCHER, R.; SENTIEYS, O. Analytical fixed-point accuracy evaluation in linear time-invariant systems. **IEEE Transactions on Circuits and Systems I: Regular Papers**, IEEE, v. 55, n. 10, p. 3197–3208, 2008.

MOHAMMED, R. K.; ABDULLAH, H. A. Implementation of digital and analog modulation systems using fpga. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 18, n. 1, p. 485–493, 2020.

NITO, E. S. A. **Sistemas embarcados com FPGA**. 2017.

OPPENHEIM, A. V.; SCHAFER, R. W. **Discrete-Time Signal Processing**. 3. ed. [S.l.]: Prentice Hall, 2010.

POCZEKAJŁO, P.; WIRSKI, R. Error analysis of digital filters using fixed point arithmetic. **International Journal of Electronics and Telecommunication**, v. 71, n. 3, p. 1–9, 2025.

PROAKIS, J. G.; MANOLAKIS, D. G. **Digital Signal Processing: Principles, Algorithms, and Applications**. 4th. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2006. ISBN 978-0-13-187374-2.

PRZYBYŁ, A. Fixed-point arithmetic unit with a scaling mechanism for fpga-based embedded systems. **Electronics**, MDPI, v. 10, n. 10, p. 1164, 2021.

SILVA, E. *et al.* All-digital fpga-based dac with none or few external components. **arXiv preprint arXiv:2008.00572**, 2020.

SWAIN, K. Fpga implementation of ask and fsk modulator based on matlab/xilinx system generator. 2020.

WINANDY, I. *et al.* Automated fixed-point precision optimization for fpga synthesis. **IEEE Open Journal of Circuits and Systems**, IEEE, 2025.

YATES, R. Fixed-point arithmetic: An introduction. **Digital Signal Labs**, v. 81, n. 83, p. 198, 2009.

YUE, Y. Analysis of amplitude shift keying under varying carrier amplitudes and channel conditions. In: IEEE. **2025 2nd International Conference on Computer Communication, Networks and Information Science (CCNIS)**. [S.l.], 2025. p. 211–215.

ZONI, D.; GALIMBERTI, A. Cost-effective fixed-point hardware support for risc-v embedded systems. **Journal of Systems Architecture**, Elsevier, v. 126, p. 102476, 2022.

ZUBAIDY, M. A. A.; QADDOORI, S. L.; GADAWE, N. T. Efficient design and implementation of the realtime multi types digital modulations system based FPGA. **Journal of Engineering Science and Technology**, v. 18, n. 2, p. 974–989, 2023.