> INSTITUTO NACIONAL DE TELECOMUNICAÇÕES, JULHO / 2025<

Sintetizador DDS/NCO

Ana Carolina Vivian dos Reis
Daniel Quiteque
Felippe Silvério de Castro
Felipe Ribeiro Lobato
Gabriel Caynã Diniz Laraia
Gabriela Fidélis Jorge
Letícia da Silva Souza

Resumo -

Este trabalho apresenta o desenvolvimento de um gerador digital de formas de onda baseado na técnica de Síntese Direta Digital (DDS), com controle dinâmico de frequência por meio da interface I2C configurada no modo escravo. O sistema implementa um Oscilador Controlado Numericamente (NCO) capaz de gerar sinais senoidais e quadrados utilizando expressões matemáticas para a construção das formas de onda. A comunicação I2C permite a configuração remota dos parâmetros de geração, sendo comandada por um módulo mestre desenvolvido em Verilog (testbench). A forma senoidal é gerada a partir de funções trigonométricas aplicadas ao acumulador de fase, enquanto a forma quadrada é derivada da mesma fase com lógica simples. O projeto foi validado por meio de testes práticos e análise funcional do código-fonte, comprovando sua viabilidade para aplicações embarcadas que demandam geração precisa e programável de sinais periódicos.

***Palavras Chaves*—Oscilador Controlado Numericamente (NCO), Síntese Direta Digital (DDS), I2C Slave, Geração de Sinais.**

I. INTRODUÇÃO

O desenvolvimento de sistemas geradores de formas de onda digitais é fundamental para aplicações em instrumentação, telecomunicações e sistemas embarcados. Em particular, técnicas de síntese direta digital (DDS, ou Direct Digital Synthesis) permitem a geração precisa e programável de sinais periódicos, como ondas senoidais e quadradas, por meio de recursos digitais como tabelas de consulta (LUTs) e acumuladores de fase.

Neste trabalho, propõe-se o projeto e a implementação de um sintetizador DDS/NCO (Oscilador Controlado Numericamente) capaz de gerar sinais senoidais e quadrados, com frequência ajustável dinamicamente por meio de uma interface I2C configurada no modo escravo. A comunicação I2C possibilita o ajuste remoto e flexível dos parâmetros de geração de sinal, tornando o sistema adequado para integração em plataformas embarcadas mais amplas.

O objetivo central deste projeto é demonstrar a viabilidade de um gerador digital com saída serial, com controle de frequência por comandos via I2C, utilizando um divisor de clock como base de tempo. A geração da forma de onda senoidal será baseada em cálculos matemáticos para garantir precisão e eficiência computacional, enquanto a forma de onda quadrada será gerada por lógica simples a partir do sinal de fase acumulada.

Ao longo deste artigo, serão apresentados o projeto conceitual do sistema, a descrição da arquitetura e das interfaces utilizadas, os métodos de implementação, além de resultados experimentais e análise de desempenho. O sistema será validado por meio de testes práticos com demonstração funcional e análise do código-fonte, atendendo aos requisitos de geração de sinais com controle dinâmico via I2C.

II. NCO

Um Oscilador Controlado Numericamente (NCO) é um sistema digital usado para gerar sinais periódicos, geralmente senoidais ou quadrados, com uma frequência controlável via entrada digital. Ele é amplamente utilizado em aplicações de modulação/demodulação digital, sistemas de comunicação, sintetizadores de frequência, GPS, radares e software-defined radio (SDR).

Ao contrário dos osciladores analógicos, que usam componentes passivos e ativos (como capacitores, resistores e transistores), um NCO é implementado inteiramente por meio de operações digitais, o que garante maior estabilidade, precisão e facilidade de controle.

A implementação típica de um NCO é baseada em três componentes principais:

- **Acumulador de Fase (Phase Accumulator):** é um contador que acumula um valor de fase a cada ciclo de clock. Ele é incrementado por uma palavra de controle de frequência chamada FCW (Frequency Control Word).
- **Conversão de Fase para amplitude:** Cálculo para extração das amostras de acordo com a fase atual do sinal.
- **Conversor Digital-Analógico (DAC):** converte o valor digital obtido da NCO em uma forma de onda analógica.

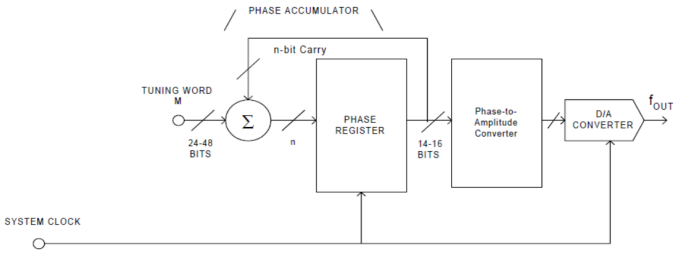

Fig. 1. Diagrama de blocos de um sintetizador digital (ANALOG DEVICES, 1999)

Cálculo da Fase

O incremento da fase do NCO é determinado pela seguinte equação:

Incremento da fase = (Freq de Saída / (Taxa de amostragem / (2 ^ 32)

A cada amostra gerada, a fase é incrementada com o valor calculado na equação do incremento. Os parâmetros de frequência de saída (variável) e taxa de amostragem (constante) são previamente determinados. A frequência de amostragem ideal envolve uma análise cuidadosa da frequência máxima do sinal que se deseja amostrar e a aplicação do

teorema de Nyquist-Shannon.
Esse controle direto e digital da frequência permite gerar sinais com resoluções extremamente finas, especialmente quando o tamanho do acumulador de fase é uma potência de 2 e maior ou igual a 32 bits.

Módulo NCO

Foi construido um módulo NCO que implementa um Oscilador Controlado Numericamente (NCO) na linguagem Verilog, capaz de gerar formas de onda digitais com frequência ajustável, para uso em um sistema de síntese direta digital (DDS). Ele possui como entradas o valor da frequência desejada em formato ponto fixo Q32.32, o tipo de forma de onda a ser gerada (senoidal ou quadrada), o ciclo ativo (duty cycle) para a forma quadrada, além do clock do sistema. A saída é um sinal digital com profundidade configurável por parâmetro.

O funcionamento do módulo pode ser dividido em três principais estágios:

1) **Cálculo do Incremento de Fase:** A frequência de saída solicitada é convertida em um valor de incremento de fase para um acumulador, utilizando uma resolução de ponto fixo. O incremento é calculado como a multiplicação da frequência de entrada por uma constante de resolução (baseada na taxa de amostragem), permitindo ajustes finos e alta precisão de frequência.

2) **Geração do Clock de Amostragem:** Um divisor de clock opcional gera um sinal de amostragem compatível com a taxa de amostragem especificada. O sistema garante que o incremento de fase ocorra apenas nos ciclos corretos do clock de amostragem, sincronizando a geração da forma de onda com a taxa de amostragem desejada.

3) **Estágio de Saída (Síntese da Forma de Onda):**
A fase acumulada é extraída dos bits mais significativos do acumulador para definir a amostra atual da forma de onda. Para a onda senoidal, o código usa uma aproximação parabólica segmentada (baseada em duas parábolas) que evita o uso de LUTs explícitas, gerando a curva senoidal com bom custo computacional. Para a forma de onda quadrada, a saída é gerada comparando a fase com o valor de duty cycle definido, criando um sinal PWM com largura de pulso configurável. O acumulador de fase é atualizado a cada pulso do clock de amostragem, garantindo continuidade e correta evolução temporal da fase.

O design usa aritmética de ponto fixo para representar tanto o incremento de fase quanto os valores de saída, permitindo alta precisão com uso eficiente de recursos lógicos. Além disso, o módulo é parametrizável em termos de profundidade de bits (BIT_DEPTH), frequência de clock (CLK_FREQ) e taxa de amostragem (SAMPLE_RATE), tornando-o facilmente ajustável para diferentes aplicações.

Abaixo uma explicação do módulo que implementa o **Numerically Controlled Oscillator (NCO)** parametrizável, capaz de gerar formas de onda (como senoidal aproximada por parábolas, onda quadrada, etc.) com frequência configurável em tempo de execução.

As diretivas *define* no início do código são constantes nomeadas usadas para facilitar a seleção do tipo de onda: Elas permitem ao usuário selecionar entre formas de onda pré-definidas com valores inteiros. No código mostrado, apenas SINE e SQUARE estão implementadas no bloco principal de saída.

Definição de alguns parâmetros:

CLK_FREQ: frequência do clock do sistema.
BIT_DEPTH: resolução em bits do sinal de saída.
SAMPLE_RATE: taxa de amostragem do gerador de formas de onda.
Esses parâmetros tornam o módulo flexível e reutilizável em diferentes contextos. As entradas e saídas são:

frequency: frequência desejada do sinal gerado, em ponto fixo Q32.32 (64 bits, com 32 bits fracionários).
wave: seleciona a forma de onda (usa os valores definidos com define).
duty_cycle: largura do pulso para onda quadrada.
clk: clock principal do sistema.
out: saída do sinal gerado em resolução BIT_DEPTH.
Constantes locais:
SAMPLE_MAX: valor máximo representável com BIT_DEPTH bits (tudo em 1).
SAMPLE_HALF: metade do valor máximo, usado como nível médio (offset DC).

Acumulador de fase

Esta é a parte central do DDS. O DDS usa um **acumulador de fase** para gerar a forma de onda. A cada ciclo de amostragem, o acumulador soma um incremento proporcional à frequência desejada. A parte mais significativa do acumulador determina a fase instantânea do sinal gerado.

Resolução do incremento

Define a resolução do incremento de fase. O valor representa 2^32 / SAMPLE_RATE, para permitir que a frequência de saída seja calculada com precisão usando aritmética de ponto fixo.

phase_acc: acumulador de fase, 32 bits.
phase_inc: incremento de fase por amostra.
phase_inc_mul_reg: resultado temporário de 128 bits para a multiplicação entre frequência (64 bits) e resolução.
phase: valor de fase usado na saída, extraído dos bits mais significativos do acumulador.

Cálculo do incremento de fase

Toda vez que *frequency* muda, calcula o incremento de fase. Multiplica *frequency* pela resolução, obtendo um valor de 128 bits. Seleciona-se os bits [63:32] como o incremento efetivo para manter a precisão do ponto fixo Q32.32.

Geração do clock de amostragem

Para garantir amostragem no SAMPLE_RATE desejado é verificado se o clock do sistema já está na frequência de amostragem.
Se estiver, usa diretamente clk. Caso contrário, usa um módulo divisor de clock (ClockDividerFF) para gerar sample_clk_cd, com a frequência de amostragem correta.

Estágio de saída

A forma de onda é gerada neste always sensível ao posedge *sample_clk*. Extrai *phase* dos bits mais significativos do acumulador.Escolhe a forma de onda com base em *wave*.

Forma de onda SENO (SINE)

```
`SINE: begin
  p_dir = phase[BIT_DEPTH - 1];
  phase = p_dir ? phase - SAMPLE_HALF : phase;
  out_fp = ((phase * phase) >> BIT_DEPTH) << 3;
  if (p_dir)
    out_fp = out_fp - (phase << 2) + SAMPLE_HALF;
  else
    out_fp = (phase << 2) - out_fp + SAMPLE_HALF;
  if (out_fp[FP_WIDTH:BIT_DEPTH] > 2) out_fp = 0;
  else if (out_fp[FP_WIDTH:BIT_DEPTH] >= 1) out_fp =
SAMPLE_MAX;
  out = out_fp[BIT_DEPTH - 1:0];
end
```

Usa aproximação por duas parábolas para gerar o seno:

- *p_dir*: seleciona se usamos a metade superior ou inferior com base no MSB da fase.
- Ajusta phase conforme *p_dir*.
- Calcula $8p^2$, aplica correções -4p+0,5 ou 4p-$8p^2$+0,5.
- Clipa o valor para faixa [0, SAMPLE_MAX].
- Extrai a porção fracional como saída.

Limita o resultado para evitar overflow/underflow.

```
out = out_fp[BIT_DEPTH - 1:0];
```

Forma de onda QUADRADA (SQUARE):

```
`SQUARE: out = phase <= duty_cycle ? SAMPLE_MAX : 0;
```

Saída é SAMPLE_MAX se a fase está abaixo de duty_cycle, caso contrário, zero. Permite controle de largura de pulso com duty_cycle.

Incremento do acumulador

```
phase_acc = phase_acc + phase_inc;
```

No final de cada ciclo de amostragem, o acumulador é incrementado. Controla o avanço da fase para a geração contínua da forma de onda.O código poderá ser acessado no link: (link github)

Módulo Divisor de Clock

Se usarmos o clock padrão do sistema, os módulos implementados não funcionarão adequadamente. Por isso, foi necessário a criação do módulo divisor de clock, que definimos como ClockDividerFF.

Este módulo faz a divisão do clock de acordo com a fórmula DIVISOR = ((1.0 / CLK_OUT_FREQ) / (1.0 / CLK_IN_FREQ)) / 2.0 utilizando um contador. Após o valor atribuído ao divisor for estabelecido, o sinal de clock tem seu estado invertido. O código poderá ser acessado no link: (link github)

III. I2C

O barramento **I2C** (Inter-Integrated Circuit) é um protocolo de comunicação serial síncrona amplamente utilizado para a conexão de circuitos integrados em sistemas embarcados. Desenvolvido pela Philips na década de 1980, o I2C é conhecido por sua simplicidade e baixo custo de implementação, permitindo a comunicação entre múltiplos dispositivos com apenas duas linhas: SDA (Serial Data) e SCL (Serial Clock).

Por meio de um esquema mestre–escravo, o mestre controla o clock e inicia as transmissões. No nosso projeto a a Testbench fará o papel de mestre enviando os comandos.

No contexto deste trabalho em Verilog, a técnica I2C será empregada para implementar um módulo de comunicação digital, possibilitando o envio e recebimento de dados entre o projeto em FPGA. A especificação do protocolo I2C será traduzida para lógica digital sintetizável.

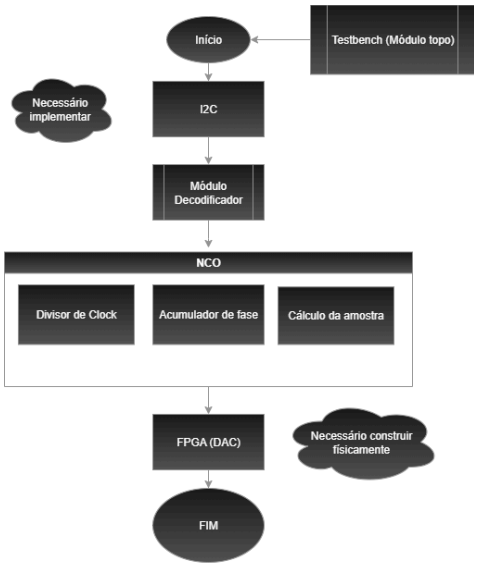

Figura 2: Diagrama de Blocos do Projeto

Star/Stop

A comunicação no barramento é delimitada por condições de início (Start) e término (Stop) da transmissão, ambas geradas exclusivamente pelo dispositivo mestre. A condição de *Start* ocorre quando a linha SDA realiza uma transição do nível lógico alto para o nível baixo enquanto a linha SCL permanece em nível alto. Por outro lado, a condição de *Stop* é caracterizada pela transição da linha SDA de nível baixo para nível alto, também com a SCL em nível alto. Em todas as demais situações, a linha SDA só deve mudar de estado quando a linha SCL estiver em nível baixo, assegurando assim a integridade dos dados.

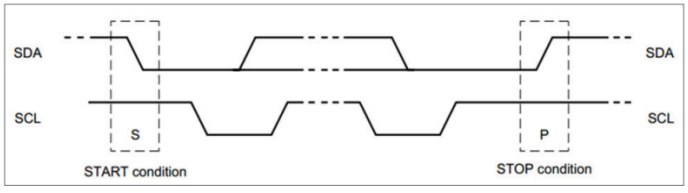

Figura 1: condições Start / Stop

Durante a transmissão, o endereço do dispositivo escravo é enviado no formato padrão de 7 bits (podendo também ser expandido para 10 bits em algumas implementações), com o bit mais significativo (MSB) transmitido primeiro. Em seguida, é enviado o bit de leitura/escrita, que indica a direção da operação: leitura quando em nível alto, ou escrita quando em nível baixo.

Quanto à transmissão dos dados, a linha SDA deve permanecer estável durante os períodos em que o clock SCL está em nível alto. Transições de estado na linha de dados só são permitidas quando o sinal de clock estiver em nível baixo. Dessa forma, o receptor lê cada bit de dados presente na linha SDA enquanto o sinal SCL está em nível alto, e o transmissor coloca cada novo bit na linha quando o sinal SCL está em nível baixo.

Máquina de Estados Finitos

A Máquina de Estados Finitos é usada para lidar com os diferentes estados no protocolo I2C. O módulo i2c_slave opera com a seguinte FSM, representada por cinco estados principais:

Estado	Descrição
IDLE	Aguarda a condição de START para iniciar a comunicação.
ADDR	Recebe o endereço do escravo (7 bits + R/W).
ADDR_ACK	Envia ACK caso o endereço corresponda ao escravo.
READ	Recebe dados do mestre (controle, frequência, duty cycle).
READ_ACK	Envia ACK após receber o dado e atualiza saídas.

O fluxo geral inicia em IDLE, detecta START, passa para ADDR para receber o endereço, confirma o endereço com ACK (ADDR_ACK) e segue para leitura dos dados (READ). Finaliza a transmissão com ACK (READ_ACK) e retorna a IDLE após STOP.
Obs: A detecção das bordas do clock scl e as transições de SDA são sincronizadas ao clock do sistema para garantir robustez.

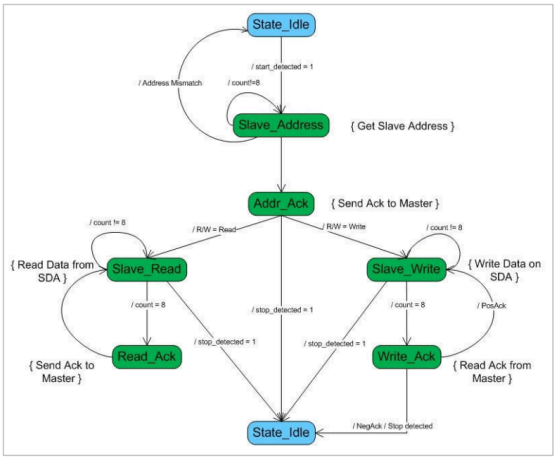

Figura 3: Máquina de Estados padrão do Protocolo

Composição do Sinal

| S | A6 | A5 | A4 | A3 | A2 | A1 | A0 | R/W | ACK | ...

<- Endereço do Escravo -> ↑

Leitura/Escrita (R/W)

S: Bit de início (START)
A6–A0: Endereço do dispositivo escravo
R/W: 0 = Escrita | 1 = Leitura
ACK: Bit de reconhecimento (Acknowledge)

| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | ACK |

↑ ↑

Atualização | Habilita/Desabilita

Duty Cycle | o NCO

- **B0**:
  0: Habilita o NCO
  1: Desabilita o NCO

- **B1/B2** — Seleção da forma de onda:
  00: Senoidal
  01: Triangular
  10: Dente de serra
  11: Quadrada

- **B3:**
  1: Atualização da frequência
  Dados enviados em seguida: 64 bits

- **B4:**
  1: Atualização do duty cycle
  Dados enviados em seguida: 16 bits

Limitações do Protocolo
O I2C possui algumas limitações:
• Baixa taxa de transferência comparada a outros protocolos (como SPI).
• Restrição no comprimento do barramento devido à capacitância.
• Sujeito a interferência e ruídos em ambientes ruidosos.
• É half-duplex onde somente o mestre ou o escravo envia dados para o barramento por vez.
• Apesar disso, é ideal para aplicações de baixa velocidade e custo reduzido.

IV. Módulo I2C Slave

Durante a implementação do módulo I2C, foram enfrentadas dificuldades iniciais relacionadas ao funcionamento da comunicação. A princípio, o sistema não respondia adequadamente aos comandos enviados, impedindo a correta configuração dos parâmetros de frequência e duty cycle. Após análise do comportamento do módulo em simulação, foram identificados três principais pontos de falha: a leitura incorreta do endereço do escravo, falhas no sincronismo do clock na testbench, e a validação incompleta das condições de recebimento dos dados. Esses problemas foram corrigidos com ajustes específicos na lógica de leitura do endereço, no alinhamento temporal entre os sinais de clock e dados, e na verificação das condições de controle na FSM receptora.

Após as correções, o módulo passou a reconhecer corretamente os comandos enviados via barramento I2C, permitindo a configuração remota dos parâmetros de geração do sinal. No entanto, ainda permaneceu pendente o recebimento completo do dado de saída, o que está relacionado a uma falha residual na lógica de leitura final. Devido a limitações de tempo, essa etapa não pôde ser finalizada até o momento, mas sua resolução está em andamento.

Tabela de Controle (ctrl_reg)

O byte de controle recebido possui campos específicos para configurar o NCO:

Bits	Função	Descrição
[0]	nco_enable	Habilita (1) ou desabilita (0) o NCO.
[2:1]	nco_wave	Seleciona o tipo de onda gerada pelo NCO:
		00: Senoidal, 01: Quadrada, 10: Triangular, 11: Dente de serra.

[4:3]	Tipo de dado a seguir	01: Frequência (64 bits), 10: Duty cycle (16 bits), outros: nenhum dado extra.
Outros bits	Reservados	Devem ser zero ou ignorados.

Após receber o byte de controle, o módulo espera pelos dados correspondentes ao tipo indicado para configurar a frequência ou o duty cycle. O código poderá ser acessado no link: (link github)

IV. TESTBENCH

A elaboração da testbench apresentou alguns desafios durante o desenvolvimento. Inicialmente, observou-se que a condição de início (sinal start) não ativava corretamente os estágios subsequentes da máquina de estados, impedindo o avanço da simulação. Após uma análise detalhada do código, identificou-se que a própria condição de start não estava sendo corretamente gerada dentro da testbench, como evidenciado na Figura 5. Para solucionar o problema, foi necessário inserir um trecho adicional de código responsável por ativar o sinal de start de forma adequada (trecho ilustrado na imagem). Com essa modificação, o sistema passou a inicializar corretamente os estados e prosseguir com a execução das demais etapas da simulação.

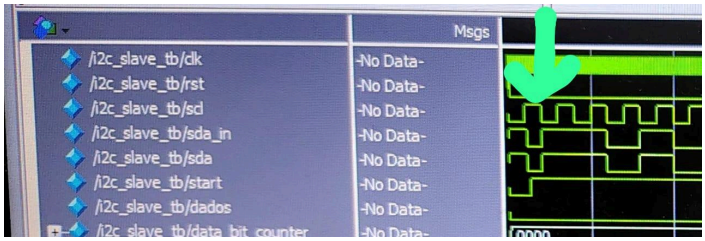

Figura 4: Erro encontrado na Testbench

Após a execução dos demais comandos, foram necessárias outras adaptações (citar adaptações) no código para que ele ficasse 100% funcional. Abaixo podemos ver as formas de onda extraidas da simulação no Software Modalsim:

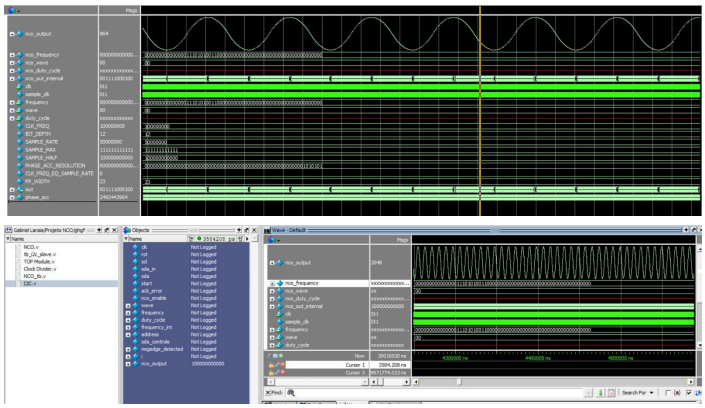

Figura 5: Imagens da Simulação da Testbench no ModalSim

Código

A testbench i2c_slave_tb tem como objetivo simular a comunicação entre um mestre I2C e um módulo escravo chamado i2c_slave, responsável por controlar um oscilador controlado numericamente (NCO). A simulação contempla a transmissão de diferentes tipos de comandos e dados para o escravo, verificando seu comportamento frente à recepção de parâmetros como forma de onda, frequência e duty cycle.

Inicialização

A simulação começa com a definição dos sinais utilizados:

clk: clock principal do sistema (100 MHz).
scl: clock da comunicação I2C (5 MHz).
sda: linha bidirecional de dados da interface I2C.
sda_in e sda_controle: controlam o valor da linha SDA durante a simulação.

frequency, duty_cycle e wave: saídas do escravo utilizadas para configurar o NCO. Também é instanciado o módulo a ser testado (uut), que representa o escravo I2C.

Controle da Linha SDA

A linha SDA é representada por um sinal do tipo wire, simulando o comportamento real de um barramento I2C com resistência de pull-up. O controle da linha é feito pela combinação dos sinais sda_in (valor lógico) e sda_controle (habilitação da escrita).

Clocks

Dois processos geradores de clock foram definidos:

- Um clock de 100 MHz para simular o sistema digital.
- Um clock de 5 MHz para representar o sinal scl da interface I2C.

Tasks de Comunicação

Para facilitar a simulação das operações I2C, foram definidas diversas tasks:

start_transmission(): simula a condição de START, seguida do envio do endereço do escravo.
stop_transmission(): simula a condição de STOP no protocolo.
send_ctrl(): envia um byte de controle com instruções ao escravo.
send_freq(): envia 64 bits representando a frequência desejada para o NCO.
send_duty(): envia 16 bits representando o duty cycle.

Essas tasks encapsulam o comportamento típico de um mestre I2C enviando dados ao escravo.

Sequência de Testes

A testbench realiza três sequências principais de comunicação:

1. Primeira transmissão:
   Envia apenas um byte de controle (0x05).
   Essa instrução pode, por exemplo, ativar o NCO ou selecionar uma forma de onda.
2. Segunda transmissão:
   Envia um novo byte de controle (0x15).
   Em seguida, transmite 64 bits representando a frequência desejada do NCO.
3. Terceira transmissão:
   Envia o byte de controle (0x0D).
   Transmite um valor de 16 bits correspondente ao duty cycle.

Ao final das transmissões, a simulação é interrompida com o comando $stop.

Sincronização com SCL

A testbench monitora as bordas de descida do sinal SCL para garantir a sincronização com o protocolo I2C, utilizando um sinal auxiliar (negedge_detected) para coordenar o envio de cada bit. O código poderá ser acessado no link: (link github)

V. DAC

Um DAC (Digital to Analog Converter) é um componente eletrônico que converte sinais digitais (valores binários) em sinais analógicos contínuos. Em sistemas de geração de sinais, como no caso de um NCO (Oscilador Controlado Numericamente), o DAC é essencial para transformar os valores numéricos da forma de onda gerada digitalmente em uma saída analógica que pode ser visualizada em um osciloscópio ou utilizada em aplicações reais.

No contexto deste projeto, o DAC recebe os valores gerados pelo NCO — que podem representar formas de onda como senoidal, quadrada, triangular ou dente de serra — e os converte em uma tensão proporcional ao valor digital. Assim, é possível observar a forma de onda gerada em tempo real através de um osciloscópio.

A interface I²C pode ser utilizada para configurar o NCO que irá transmitir os dados de saída para o DAC.

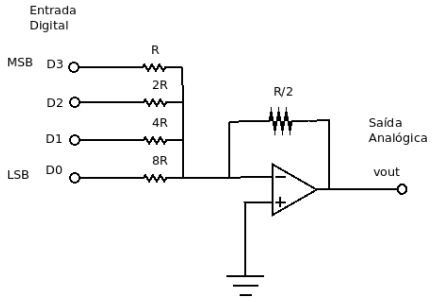

Figura 6: Esquema elétrico DAC

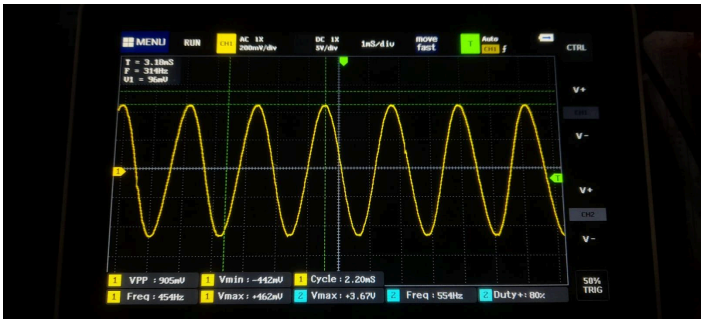

Figura 7: Tela do Osciloscópio mostrando as formas de onda

VI. FPGA

Após o desenvolvimento e simulação dos blocos funcionais do circuito NCO,, a última etapa do projeto consiste em sua integração e implementação física em um FPGA (Field-Programmable Gate Array). Essa fase é fundamental para validar o comportamento real do sistema, verificar a geração das formas de onda em tempo real e garantir a comunicação correta com outros módulos externos, como interfaces de controle via I2C.

A integração com o FPGA permite que o circuito sintetizado seja testado em ambiente de hardware, utilizando entradas reais (como switches ou comunicação serial) e observando as saídas em osciloscópios ou LEDs, por exemplo. Também é nessa etapa que se verifica o desempenho do NCO em condições práticas, analisando o clock, a estabilidade da frequência gerada e o tempo de resposta aos comandos enviados.

Realizamos a simulação utilizando uma onda Senoidal e Triangular para validar o funcionamento do circuito NCO, sendo executado com sucesso.

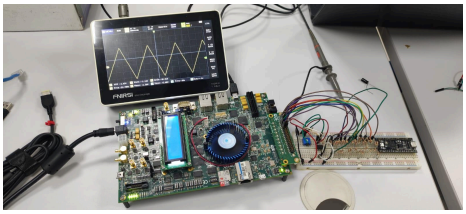

Figura 9: Imagem da onda triangular no osciloscópio

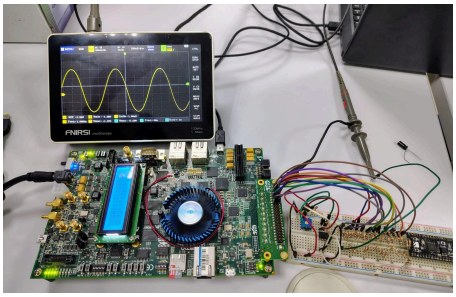

Figura 10: Imagem da onda senoidal no osciloscópio

VII. CONCLUSÃO

O desenvolvimento de um Oscilador Controlado Numericamente (NCO) com controle externo via protocolo I2C demonstrou a viabilidade de se projetar sistemas digitais configuráveis e modulares utilizando Verilog e FPGA. A construção do NCO permitiu a geração de diferentes formas de onda (senoidal, triangular, dente de serra e quadrada), com parâmetros como frequência e duty cycle ajustáveis dinamicamente.

A implementação da comunicação I2C proporcionou uma interface eficiente e de baixo custo para o controle do NCO, simulando um cenário real onde um microcontrolador mestre envia comandos para um escravo programável. A testbench desenvolvida validou o comportamento do sistema em ambiente de simulação, e a etapa final de integração no FPGA confirmou seu funcionamento em tempo real.

Por fim, o projeto evidencia a importância da combinação entre técnicas de modelagem digital, protocolos de comunicação e verificação em hardware, reforçando a aplicabilidade dos conceitos estudados em soluções práticas para eletrônica embarcada, instrumentação e geração de sinais digitais.

Referências

1. NEVES, Felipe. *DDS – "Direct digital synthesis", geração de funções arbitrárias por software*. Embarcados, 6 maio 2014. Disponível em: https://embarcados.com.br/dds-sintese-digital-direta/. Acesso em: 10 jul. 2025.
2. Gabriel A. F. Souza, Gustavo D. Colletta, Leonardo B. Zoccal, Odilon O. Dutra. *SD142 - Circuitos Digitais III. Aulas 56-57: Protocolo de Comunicação - I2C*. [Material de aula]. Programa CI Digital, Ano 2025.
3. WANTRONICS. *Comunicação I2C*. 1 jan. 2024. Disponível em: https://wantronics.com.br/2024/01/01/comunicacao-i2c/ . Acesso em: 12 jul. 2025.
4. TIRFIL. VhdI2CSlave. Disponível em: https://github.com/tirfil/VhdI2CSlave. Acesso em: 12 jul. 2025.
5. Sony, "CXD2099AR Digital Signal Processor for CD Player," *Datasheet*, [Online]. Available: https://exemplo.com/CXD2099AR.pdf. Acesso em: 12 jul. 2025.
6. Texas Instruments, "Understanding the I2C Bus," *Application Report*, [Online]. Available: https://www.ti.com/lit/an/slva704/slva704.pdf. Acesso em: 12
7. jul. 2025.