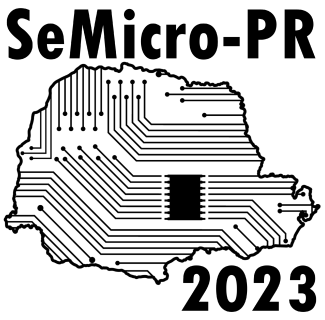

Descrição em VHDL de um Modelo Matemático para Circuito Linear Sem Memória

Vitória Mariana da Rocha Gasino, Sibilla Batista da Luz França, Eduardo Gonçalves de Lima
Grupo de Concepção de Circuitos e Sistemas Integrados
Universidade Federal do Paraná, Curitiba, Brasil
vitoriagasino@ufpr.br

Resumo- A equação de uma reta é um exemplo fundamental de modelagem matemática, que se concentra na exploração, extração e manipulação em tempo real de um modelo linear sem memória. O objetivo deste estudo é aprimorar a compreensão desse modelo linear por meio de uma implementação em hardware, utilizando FPGA. Neste trabalho, a modelagem matemática é empregada para reproduzir o comportamento de um circuito linear sem memória. O projeto é composto por dois blocos principais, um somador e um multiplicador, sendo essas operações ativadas pelo sinal de clock em nível alto. Sendo assim, o circuito final opera com dois ciclos de clock de latência, resultando em um design eficiente e rápido. Simulações tendo como alvo a FPGA Spartan 3E 500, mostram que o design utiliza 28 flip-flop e 13 LUTs (Look up Tables), indicando que o design proposto é simples e eficiente, podendo ser processado de forma ágil pela FPGA.

I. INTRODUÇÃO

A simulação de circuitos desempenha um papel fundamental no projeto e na análise de circuitos eletrônicos, permitindo antever o comportamento e o desempenho desses circuitos antes mesmo de sua implementação física. Através da utilização de modelos matemáticos, os simuladores têm a capacidade de representar com alta precisão o comportamento dos diversos componentes, possibilitando a detecção antecipada de potenciais problemas e limitações no projeto [1-2].

Os circuitos eletrônicos podem ser classificados de acordo com as suas propriedades de linearidade e memória. A linearidade de um circuito é estabelecida pela maneira como a corrente se relaciona com a tensão. Um exemplo de elemento linear é o resistor, que obedece à lei de Ohm. Essa lei estabelece que a tensão através do resistor é igual ao produto da corrente que o atravessa pelo seu valor de resistência, o que evidencia uma relação linear entre corrente e tensão [1]. Por outro lado, a memória de um sistema refere-se à dependência das saídas atuais em relação às entradas anteriores. Circuitos que possuem elementos capazes de armazenar energia, como capacitores, demonstram esse comportamento.

Para a análise de circuitos elétricos, a lei de Kirchhoff das correntes (conhecida como LKC ou lei dos nós) e a lei de Kirchhoff das tensões [2] (LKT ou lei das malhas) desempenham um papel fundamental. Essas leis são essenciais para a condução dessa análise, onde a partir da conservação de carga, estabelecem que a soma algébrica das cargas em um sistema permanece constante. A LKC, sendo a primeira dessas leis, declara que a soma das correntes que entram em um ponto de conexão é igual à soma das correntes que saem desse ponto. Por sua vez, a LKT, a segunda lei, estabelece que a soma algébrica de todas as tensões ao longo de um circuito fechado é zero [3].

Para compreender e prever o comportamento de circuitos, além das leis de Kirchhoff, é também possível modelar sistemas elétricos por meio da modelagem matemática. A modelagem matemática consiste em representar fenômenos, sistemas ou processos do mundo real utilizando equações, fórmulas e relações matemáticas. Para a análise e representação de amplificadores de potência, por exemplo, a utilização de equações, algoritmos e relações matemáticas [4] se torna extremamente vantajosa. Isso otimiza o tempo de simulação, pois permite uma análise mais rápida e eficiente dos parâmetros de desempenho do amplificador.

Esse trabalho apresenta uma implementação em linguagem VHDL de uma modelagem matemática de um circuito linear sem memória, usando a equação de uma

reta identificada a partir do método dos mínimos quadrados.

Na segunda seção, é apresentada a abordagem matemática utilizada neste trabalho. Na terceira seção, descreve-se em detalhes cada etapa do processo de modelagem, que se divide entre a aquisição de dados no simulador e a elaboração do código em Python. A quarta seção trata do circuito desenvolvido e seus resultados e análises. Na quinta seção, são expostas as considerações finais do estudo, bem como as direções futuras para a continuação do trabalho.

II. MODELAGEM MATEMÁTICA DE SISTEMAS LINEARES SEM MEMÓRIA

A criação e a formulação de representações matemáticas de sistemas, fenômenos ou situações e processos do real, constitui uma importante ferramenta para a simplificação e estudo destes processos. Conhecida como modelagem matemática, com essa abordagem, é possível traduzir as características essenciais de um sistema complexo em equações matemáticas ou modelos que possam ser analisados, simulados e compreendidos de maneira mais acessível. A modelagem matemática tem uma ampla gama de aplicações em diversas áreas, como engenharia, física, biologia, economia e muitas outras. Ela permite a previsão de comportamentos, a otimização de processos, a exploração de cenários hipotéticos e a compreensão mais profunda de sistemas reais por meio de abstrações matemáticas.

Uma das maneiras mais simples de compreender um processo linear é modelá-lo por meio da relação entre uma variável independente e outra dependente. Essa equação característica, representada como

$$y = ax + b \qquad (1)$$

é um dos componentes fundamentais da análise e descrição de sistemas lineares. Ela descreve como a variável dependente y varia em resposta a mudanças na variável independente x, sendo a o coeficiente angular e b o termo constante que determinam o comportamento da relação linear.

Uma forma poderosa de analisar equações lineares é o método dos mínimos quadrados. Esse método, a partir de um conjunto de amostras de entrada e saída, procura encontrar a linha reta que melhor se ajusta aos dados, minimizando a soma dos quadrados das diferenças entre os valores reais e os valores previstos pela equação linear. O método dos mínimos quadrados é amplamente utilizado em estatísticas e análise de dados para estimar parâmetros desconhecidos em modelos lineares, reduzindo o impacto de erros ou ruídos nos dados e obtendo uma representação mais precisa da relação subjacente entre as variáveis.

A implementação deste tipo de operação em FPGAs é interessante devido a sua capacidade de processamento em paralelo. Esse atributo permite que FPGAs executem cálculos complexos de forma eficiente, tirando proveito de múltiplos recursos de hardware simultaneamente, o que resulta em melhorias significativas no desempenho e na eficiência energética das aplicações.

III. DESENVOLVIMENTO DO MODELO

Utilizando o software *Advanced Design System* (ADS), foram coletados 50 pontos de medidas dos valores de tensão de entrada(indicada na figura pela variável $V1$, à direita) e de saída (indicada na figura pela diferença entre as variáveis V_saida1 e V_saida2 , à esquerda) , do circuito mostrado na Figura 1.

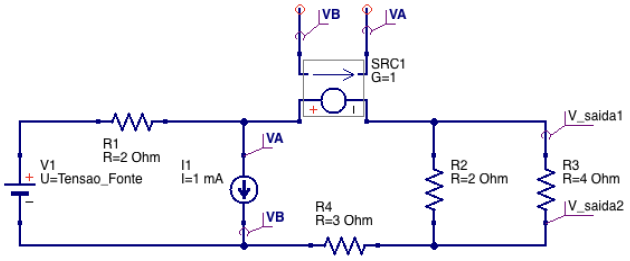

Figura 1- *Circuito de simulação para aquisição de dados*

A implementação teve início no Python, onde os coeficientes angular e linear da reta foram calculados usando aritmética de vírgula flutuante a partir de dados obtidos na simulação. O código utilizou o método dos mínimos quadrados para calcular os coeficientes da reta desejada e avaliar a discrepância entre as respostas projetadas e os resultados reais. Os valores obtidos foram normalizados e posteriormente convertidos para vírgula fixa. Em seguida, na linguagem de descrição de hardware, esses valores inteiros foram utilizados como entrada do circuito, resultando em uma redução do consumo lógico dentro da FPGA e viabilizando a validação dos resultados obtidos em VHDL com os do Python.

No processo de conversão de vírgula flutuante para vírgula fixa, testaram-se valores inteiros, incluindo 4, 8 e 16 como potências de 2. O objetivo era determinar qual deles seria mais apropriado em termos do número de bits necessários para alocar cada variável na FPGA, enquanto mantinha o erro quadrático médio entre os valores obtidos e esperados dentro de limites aceitáveis. Para isso, multiplicaram-se os valores de x (tensão de entrada) e os valores calculados par a e b por 2^nem cada caso. Posteriormente, os valores foram convertidos de volta para o formato de vírgula flutuante a fim de serem comparados com os resultados obtidos para cada valor de n.

A Figura 2 apresenta os gráficos gerados, que relacionam a resposta esperada com a resposta obtida para diferentes valores usados na conversão para vírgula fixa.

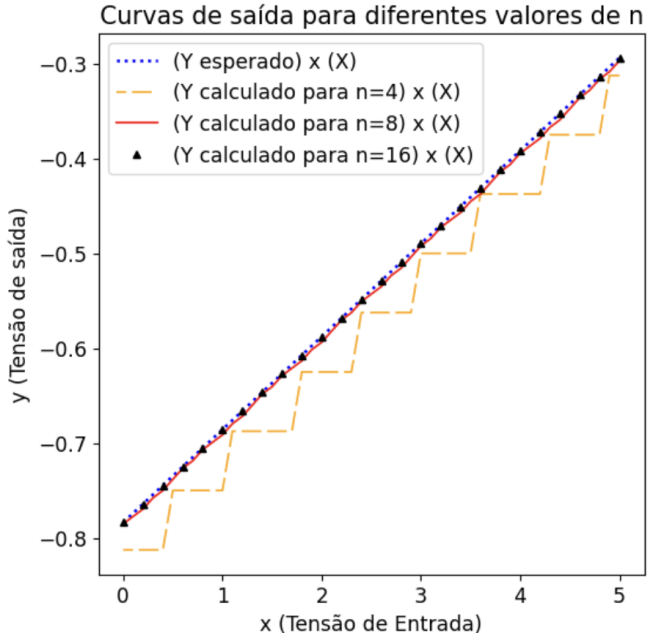

Figura 2- *Gráfico dos resultados para diferentes valores de n*

Para a multiplicação dos valores realizada com $n = 4$, foi encontrado um número mínimo de bits igual a 9 e um erro quadrático médio (MSE) de aproximadamente $6x10^{-3}$. Para $n = 8$, foram encontrados valores de 14 bits e MSE inferior a $2x10^{-5}$. Em questão de erro, a melhor resposta foi com $n = 16$, representado pela curva em verde, com MSE quase nulo (próximo a $1x10^{-10}$), porém o número de bits necessários foi de 22. Embora o valor de precisão utilizando o valor de $n = 8$ seja um valor muito alto se comparado ao erro utilizando $n = 16$, a redução na quantidade de bits de 14 para 22 resultou em uma economia significativa de recursos lógicos no hardware. Especificamente, a implementação com $n = 8$ exigiu apenas 13 LUTs e 28 flip-flops, enquanto $n = 16$ exigiria 32 LUTs e 44 flip-flops. Além disso, devido à maior quantidade de bits, o circuito com 22 bits exigiu um tempo maior para registro das operações, de 4,001 ns contra 2,472 ns com o circuito realizado com 14 bits. Como a frequência de operação desejada era de 10 ns por ciclo de clock, o tempo de registro de quase metade do valor do período(5 ns) tornaria a implementação com 22 bits praticamente inviável, já que a resposta correta ficaria disponível quase meio ciclo depois do esperado. Portanto, a decisão de trabalhar com a normalização com $n = 8$ foi baseada na otimização da relação entre precisão e economia de recursos lógicos no hardware, tornando-a uma escolha mais eficiente para esse contexto.

A equação da reta, com coeficientes linear e angular calculados em vírgula fixa, é representada por a=125 (coeficiente linear) e b=-201 (coeficiente angular), confirme:

$$y = 125x - 201 \qquad \textbf{(2)}$$

IV. DESCRIÇÃO EM HARDWARE

Para a elaboração do hardware, era necessário descrever os dois componentes que desempenhavam os principais papéis no circuito: o multiplicador(ax) e o somador($[ax] + b$). O esquemático do circuito final ficou conforme Figura 3.

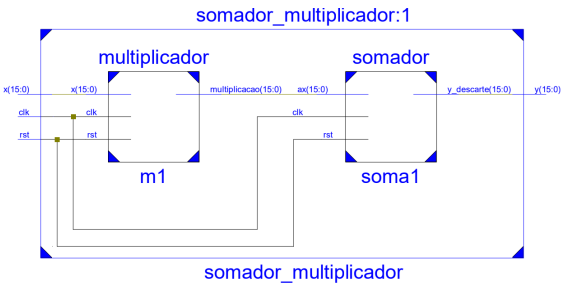

Figura 3- *Circuito gerado em VHDL*

Para validar o hardware implementado, um testbench foi elaborado para simular as saídas do circuito. Foi realizada uma simulação post-route, utilizando o software da Xilinx ISE 14.7, que leva em conta o comportamento real do circuito, considerando as características específicas da implementação física, tais como atrasos de propagação, capacitâncias parasitas, resistências e outros fatores que podem influenciar o desempenho do circuito. A FPGA utilizada para o projeto foi uma Spartan 3E 500. Foi configurado um período de clock de 10 ns, onde foram passados a cada ciclo de clock os valores de entrada, utilizando os números em vírgula fixa das tensões de entrada. Uma série de testes foi conduzida, abrangendo todas as 50 entradas geradas em vírgula fixa, pelo código em Python. Os resultados podem ser observados na Figura 4.

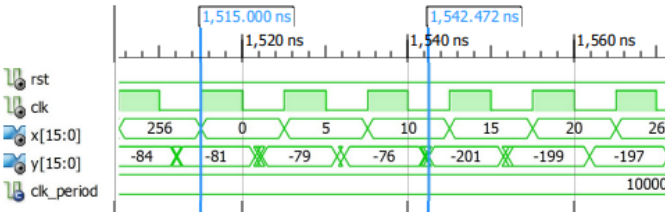

Figura 4- *Simulação considerando os atrasos da FPGA.*

Os resultados começam a se apresentar corretamente a partir de aproximadamente 100 ns, que é o tempo de inicialização do circuito. A partir deste intervalo, as respostas são constantemente obtidas a cada dois ciclos de clock após a entrada correspondente. Na Figura 4, observa-se na primeira linha o sinal de reset('rst'), na segunda o sinal de clock('clk'), na terceira as entradas do circuito (x) e na última linha as saídas do

sistema (y). O primeiro marcador à esquerda, em 1515,00 ns, indica o momento em que o valor de entrada $x = 0$ é fornecido ao circuito. Nesse cenário, a resposta esperada é $y = -201$. O segundo marcador, em 1542,472 ns, assinala o ponto em que a resposta para $x = 0$ se torna disponível. Podemos observar que a resposta y fica pronta com aproximadamente 2 ciclos após a entrada x ficar disponível, com um pequeno atraso(de 2,472 ns) devido ao tempo de registro no flip-flop. É possível notar, também na Figura 4, três valores anteriores ao resultado para $x = 0$, sendo eles -81, -79 e -76. Esses resultados são referentes às entradas $x = 122$, $x = 125$ e $x = 126$, respectivamente.

Assim, conforme previsto, o circuito apresenta uma latência de 2 ciclos de clock entre a entrada e a saída, devido ao registro de cada operação de multiplicação e de soma. O sinal de clock atua como um gatilho para o flip-flop, determinando o momento preciso em que o resultado da operação é capturado e armazenado no elemento de memória, permitindo a sincronização adequada do processamento digital [5].

O número total de LUTs utilizadas foi de 13, representando menos de 1% do total de look-up tables disponível (≅ 0,13%). Já o número de flip-flops utilizados foi de 28.

V. CONCLUSÃO

Essa pesquisa apresentou um modelo para estimar as saídas de circuitos lineares sem memória utilizando linguagem de descrição de hardware. Foi observado que é possível reproduzir o comportamento de um circuito linear por meio da regressão linear e implementar esse cálculo em um hardware lógico combinacional. O circuito descrito demonstra uma economia de recursos na FPGA, com apenas 13 LUTs utilizadas, uma fração mínima em relação à capacidade total de LUTs da FPGA, o que indica abundante capacidade de lógica disponível para incorporar outras funções ou módulos no projeto. Além disso, o uso de apenas 28 flip-flops indica um design relativamente simples, simplificando o entendimento e a depuração do projeto, resultado de uma otimização do código que garantiu que as operações desejadas fossem realizadas com o mínimo de recursos. Isso é essencial em FPGAs, onde a otimização afeta diretamente a eficiência, o desempenho e o consumo de energia. A disponibilidade de recursos não utilizados na FPGA oferece flexibilidade para futuras expansões ou modificações do projeto, sem a necessidade de uma FPGA maior ou mais cara. Outro destaque está na velocidade do sistema, que opera a uma frequência de 100 MHz, importante para aplicações como esta e na reprodução do comportamento de amplificadores de potência.

Os trabalhos futuros concentram-se em testar a efetividade do hardware na prática. Além disso, com os resultados bem-sucedidos na descrição de circuitos somadores e multiplicadores, o próximo passo consiste em realizar a modelagem de sistemas mais complexos que requerem um maior número de equações, como é o caso de amplificadores de potência. É importante notar que mesmo reproduzindo o comportamento de circuitos mais complexos, em VHDL ainda faremos uso de blocos somadores e multiplicadores.

AGRADECIMENTOS

Este trabalho foi desenvolvido no âmbito do programa PIBIC-Voluntária UFPR 2022.

REFERÊNCIAS

[1] CORDEIRO, Leticia, e Eduardo G. Lima. Análise de equilíbrio harmônico com não linearidades polinomiais descritas no domínio da frequência, 2021

[2] ALEXANDER , Charles K., e Matthew NO Sadiku. Fundamentos de circuitos elétricos. AMGH Editora, 2013.

[3] BOYLESTAD, Robert L. Introdução à análise de circuitos. 12ª ed. São Paulo. Pearson Prentice Hall, 2012.

[4] MARCONDES , Bruna Temporal . Aplicação de Pré-Distorsores em Rede de Transmissores Sem Fio Passíveis de Imprecisões no Modulador em Quadratura, 2019

[5] PEDRONI , Volnei A. Circuit design and simulation with VHDL. MIT press, 2010.