UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

DEPARTAMENTO DE TECNOLOGIA

CURSO DE ENGENHARIA DE COMPUTAÇÃO

HEITOR ABDALLA MASCARENHAS, SIMEONY ALVES LIMA DE ABREU

RELATÓRIO 01: CONSTRUÇÃO DE UMA CALCULADORA DIGITAL

Feira de Santana - BA

Setembro, 2025

HEITOR ABDALLA MASCARENHAS, SIMEONY ALVES LIMA DE ABREU

RELATÓRIO 01: CONSTRUÇÃO DE UMA CALCULADORA DIGITAL

Trabalho apresentado como requisito parcial na disciplina **Módulo Integrativo de Circuitos Digitais (TEC498)**, ministrada pelo professor **Anfranserai Morais Dias**, no curso de **Engenharia de Computação da Universidade Estadual de Feira de Santana (UEFS)**. Relatório referente à **primeira entrega do PBL**, descrevendo a construção de uma **calculadora digital de 4 bits** com resultados exibidos em LEDs e displays de sete segmentos, integrando teoria e prática em circuitos digitais conforme as normas da **ABNT**.

Feira de Santana - BA

Setembro, 2025

SUMÁRIO

1. INTRODUÇÃO

Segundo Tocci, Widmer e Moss (2011), a Unidade Lógica e Aritmética (ULA) é um circuito combinacional que executa operações aritméticas, como soma e subtração, e operações lógicas fundamentais em sistemas digitais, como OR, AND e XOR. Estas operações são essenciais para o processamento rápido de dados e para a execução integrada de cálculos matemáticos e decisões lógicas.

Neste contexto, o presente projeto visa desenvolver uma ULA de 4 bits capaz de realizar operações selecionáveis sobre dois operandos binários de 4 bits, controladas por chaves seletoras na placa FPGA. As saídas serão exibidas em LEDs e displays de sete segmentos, permitindo validação prática das operações implementadas. Além disso, a implementação busca consolidar o conhecimento em circuitos digitais por meio da linguagem Verilog, promovendo habilidades técnicas e documentação científica adequada.

2. FUNDAMENTAÇÃO TEÓRICA

A construção de uma calculadora digital de 4 bits requer a integração de múltiplos módulos capazes de realizar operações aritméticas e lógicas, controladas por sinais binários e implementadas em linguagem Verilog. Conforme explicam Mano e Ciletti (2013), a abordagem modular garante que cada operação seja testada e validada individualmente antes da integração no circuito principal, assegurando maior precisão e clareza no desenvolvimento.

Sobre as operações aritméticas, os somadores e subtratores de 4 bits são implementados a partir de blocos de 1 bit, garantindo a propagação correta de carry e borrow, o que naturalmente assegura precisão quando faz-se as operações. A multiplicação é realizada através de somas sucessivas de produtos parciais gerados por portas lógicas *AND*, enquanto a divisão se baseia em subtrações repetidas para determinar quociente e resto, acionando *flags* de erro em casos especiais, como divisão

por zero. As operações lógicas, *AND*, *OR* e *XOR*, por sua vez, são implementadas em módulos únicos, permitindo que a calculadora execute também funções não aritméticas.

A escolha do resultado a ser exibido é controlada por um módulo multiplexador que foi instanciado internamente oito vezes (bit a bit), que determina qual módulo de cada uma das sete operações enviará seu valor à saída final. Para tornar os resultados legíveis, usou-se um decodificador que converte os números binários em sinais capazes de acender displays de sete segmentos, apresentando valores decimais de forma direta e intuitiva. Isso ocorre através do uso das oito saídas do multiplexador, em que os quatro bits menos significativos são convertidos para um número decimal mostrado no display, enquanto os quatro mais significativos apenas indicam a presença de uma *flag Overflow*, por intermédio de uma porta *OR* que está conectada a esses bits. Por fim, as *flags* informam condições especiais do circuito, como *Zero* (quando resultado for zero), *Overflow* (quando há sobrecarga na memória e, por conseguinte, não é possível representar a saída inteiramente), *Error* (quando ocorre um evento adverso, como uma divisão por zero) e *Carry Out* (quando ocorre um carry out ou borrow out dos módulos de soma e subtração, respectivamente).

Dessa forma, demonstra-se evidentemente que a combinação de módulos bem estruturados, integração hierárquica e descrição em Verilog possibilita a implementação de uma calculadora digital funcional, capaz de realizar e exibir operações aritméticas e lógicas de forma eficiente e confiável.

3. METODOLOGIA

O desenvolvimento da calculadora seguiu uma abordagem sistemática, combinando princípios de **desenvolvimento modular**, **top-down** e **incremental**. O **desenvolvimento modular** consiste em dividir o sistema em submódulos independentes, facilitando implementação, teste e manutenção. Segundo Pressman (2014), o **desenvolvimento incremental** permite que o sistema seja construído e validado em etapas, o que assegura maior controle sobre falhas. De forma complementar, Mano e Ciletti (2013) destacam que a combinação das abordagens

top-down e **modular** simplifica a implementação, permitindo uma integração progressiva e eficiente.

Para este projeto, utilizou-se apenas **Verilog combinacional**, com portas lógicas e instâncias de módulos, sem elementos sequenciais ou memória. O sistema foi decomposto em módulos — **somadores, subtratores, multiplicadores, divisores, operações lógicas, multiplexadores e decodificadores** — que foram implementados e testados isoladamente, incluindo propagação de carry e borrow, verificação de overflow e tratamento de divisões por zero.

Nessa perspectiva, após a devida validação individual, os módulos foram integrados por instanciação ao *CircuitoMain*, que coordenou todas as operações e direcionou os resultados para o multiplexador e para os displays de sete segmentos via decodificador. Simulações finais confirmaram o funcionamento correto do sistema e dos indicadores, assegurando confiabilidade antes da implementação física.

Desse modo, a metodologia aplicada garantiu que a calculadora digital fosse desenvolvida de maneira estruturada, eficiente e didática, evidenciando a eficácia do *desenvolvimento modular*, *incremental* e *top-down* utilizando apenas **Verilog combinacional**, sem nenhum uso de memória.

4. PROJETO E IMPLEMENTAÇÃO

4.1 Lista de módulos principais

- **SomComp1bit** e **SomComp4bits**: realizam a soma de números binários de 1 e 4 bits, respectivamente.
- **SubComp1bit** e **SubComp4bits**: realizam a subtração de 1 e 4 bits.
- **Mult1bit** e **Multiplicador4bits**: o módulo de 1 bit serve como base para o multiplicador de 4 bits.
- **Divisor4bits**: calcula divisões de 4 bits, incluindo verificação de divisor zero.
- **MUX1bit** e **Multiplexador**: selecionam qual operação será exibida.
- **And4bits**, **Or4bits**, **Xor4bits**: realizam operações lógicas.

- **Decoder** e **Segmentos (A–G)**: convertem resultados binários em sinais para os displays de sete segmentos.
- **Comparador4bits**: dá suporte à divisão e aos indicadores de *flags*.
- **CircuitoMain**: integra todos os módulos em um sistema funcional único (Universidade Estadual de Feira de Santana, 2025).

4.2 Funcionamento sistemático dos módulos

Os somadores e subtratores propagam resultados parciais bit a bit, enquanto os multiplicadores utilizam somas de produtos parciais para gerar resultados de 8 bits. Conforme David Harris e Sarah Harris, em *Digital Design and Computer Architecture* (2012), os divisores realizam subtrações sucessivas e acionam indicadores em caso de divisão por zero. As operações lógicas comparam os bits correspondentes, e o multiplexador seleciona qual saída será exibida. Segundo Tocci, Widmer e Moss, em *Fundamentos de Eletrônica Digital* (2011), o decoder converte os resultados binários em formato decimal para exibição nos displays.

4.3 Hierarquia e instâncias

O projeto segue uma arquitetura hierárquica, iniciando com blocos simples, como somadores de 1 bit, e expandindo até a calculadora completa, permitindo reaproveitamento de módulos e facilidade em testes e correções. M. Morris Mano e Michael D. Ciletti, em *Circuitos Digitais e Projeto de Sistemas Digitais* (2013), destacam que a abordagem top-down garante integração eficiente e manutenção simplificada. O **CircuitoMain** centraliza todas as operações, reunindo os módulos em um único circuito funcional e bem estruturado.

5. SIMULAÇÃO E TESTES

Neste capítulo, serão apresentados os testes e simulações realizados para validar o funcionamento correto da calculadora digital de 4 bits antes da síntese em FPGA. Serão abordados **os módulos individuais instanciados no *CircuitoMain***, detalhando como cada um foi testado, quais casos de teste foram considerados, e

haverá espaço reservado para **inserção das imagens das simulações e das tabelas verdade** correspondentes. O objetivo é assegurar a confiabilidade do projeto e a precisão das operações aritméticas e lógicas. Em cada seção, **as simulações são referentes ao módulo completo instanciado no *CircuitoMain* (exceto em casos excepcionais)**, enquanto **as tabelas verdade se referem à menor instância do módulo (normalmente a de 1 bit)**, garantindo que a lógica esteja bem estruturada.

5.1 Somador

Os somadores foram testados isoladamente para verificar a propagação correta de *carry* entre os bits e a precisão das somas de 1 e 4 bits. Foram considerados casos de teste com números pequenos, iguais e máximos possíveis (1111 em binário). Abaixo é possível ver a tabela verdade (**Imagem 01**) e a simulação (**Imagem 02**) do somador:

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Expressão:	
S = A xor B xor Cin	
Cout = B(A + Cin) + A Cin	

Imagem 01 - Tabela Verdade de um Somador Completo (1 bit)

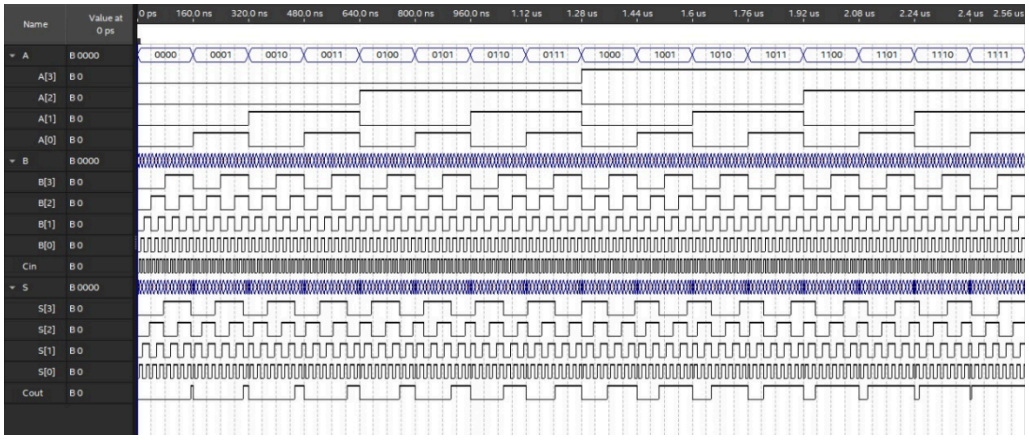

Imagem 02 - Simulação do Somador Completo de 4 bits

5.2 Subtrator

Os subtratores foram simulados para validar a subtração bit a bit, incluindo propagação de *borrow*. Casos de teste incluíram subtração de números iguais, menores e maiores, verificando a ativação correta da *flag* de resultado *zero* e a detecção de *overflow*. Segue tabela verdade (**Imagem 03)** e a simulação (**Imagem 04**) do subtrator:

A	B	Bin	S	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Expressão:
S = A xor B xor Bin
Bout = B(A' + Bin) + A'C

Imagem 03 - Tabela Verdade de um Subtrator Completo (1 bit)

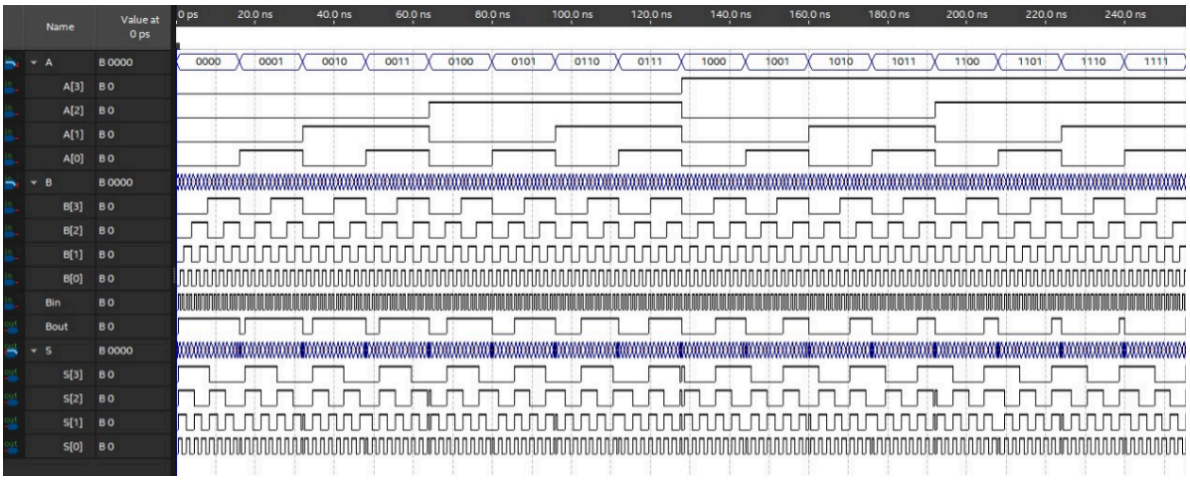

Imagem 04 - Simulação de um Subtrator Completo de 4 bits

5.3 Multiplicador

O módulo multiplicador foi testado verificando a soma correta de produtos parciais (que são instâncias de somadores) para gerar o resultado de 8 bits. Casos de teste incluem multiplicações por zero, por números iguais e combinações variadas de 4 bits. A tabela verdade (**Imagem 05**) e a simulação (**Imagem 06**) do módulo multiplicador atrelado ao *CircuitoMain* estão abaixo:

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Imagem 05 - Tabela Verdade de um Multiplicador (1 bit)

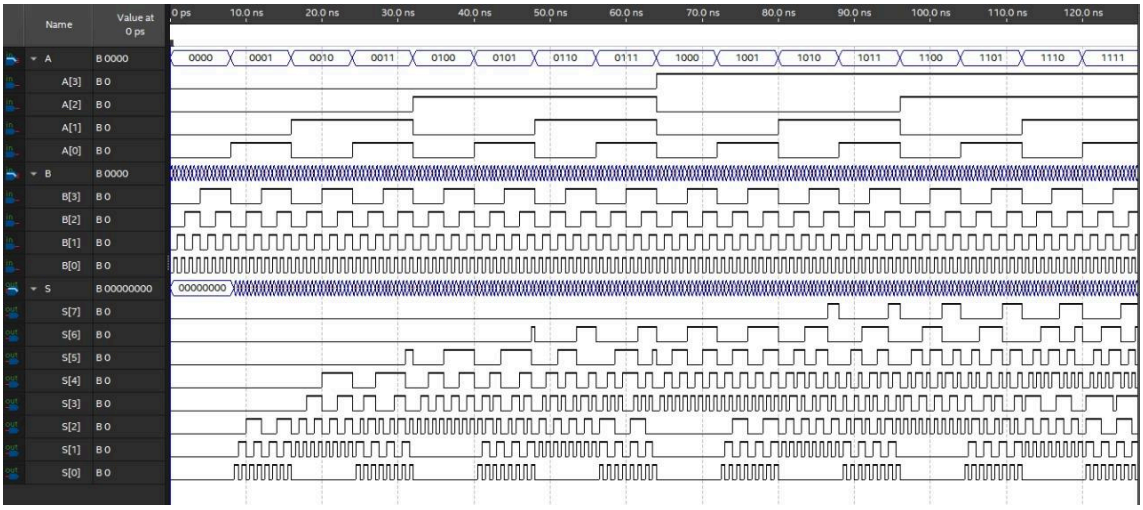

Imagem 06 - Simulação do Multiplicador (4 bits)

5.4 Divisor

O divisor foi validado realizando subtrações sucessivas para calcular quociente e resto, além do uso do multiplexador de 1 bit. Foram testadas divisões normais e divisões por zero, assegurando que a *flag Error* fosse corretamente acionada. Vale ressaltar que o divisor usa inúmeras instanciações do subtrator e um multiplexador de 1 bit para a tomada de decisões e, portanto, ele não possui uma tabela verdade própria. Abaixo está a imagem da simulação do divisor (**Imagem 07**) atrelado ao *CircuitoMain*:

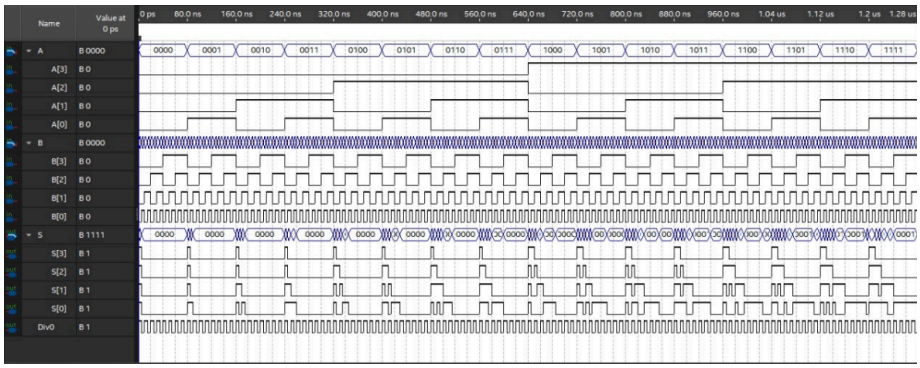

Imagem 07 - Simulação do Divisor (4 bits)

5.5 Operações lógicas

Os módulos de operações lógicas foram simulados para verificar a correspondência entre entradas e saídas, testando todas as combinações possíveis de 4 bits para *AND*, *OR* e *XOR*. Não colocou-se neste documento as tabelas verdade porque são compostas de quatro repetições da própria porta que ela opera. Contudo, está inserido as simulações AND (**Imagem 08**), OR (**Imagem 09**) e XOR (**Imagem 10**):

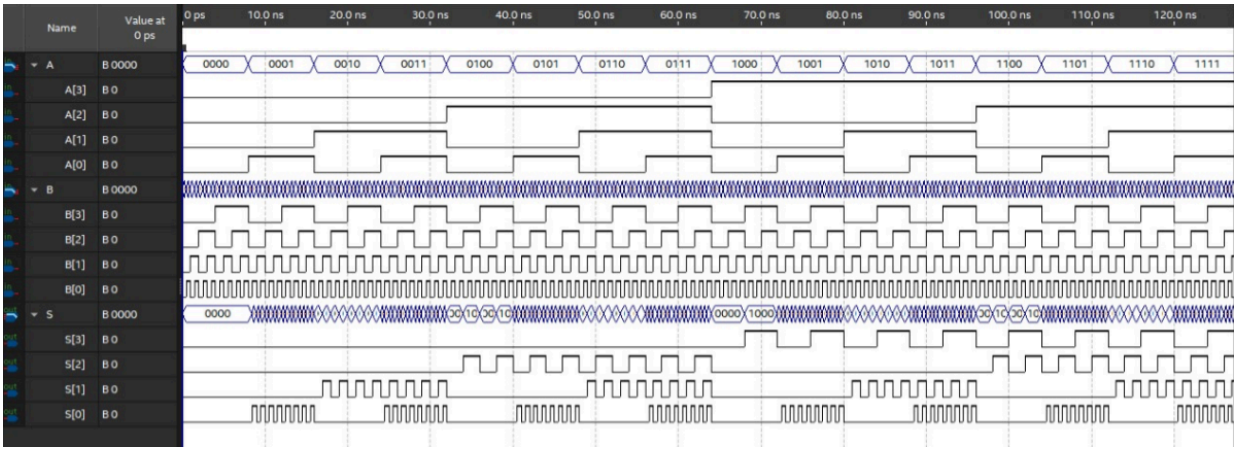

Imagem 08 - Simulação do AND (4 bits)

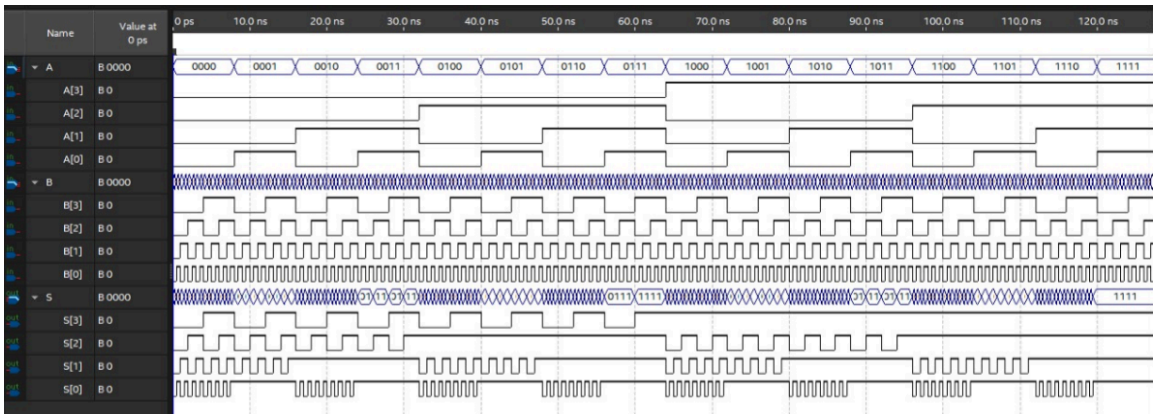

Imagem 09 - Simulação do OR (4 bits)

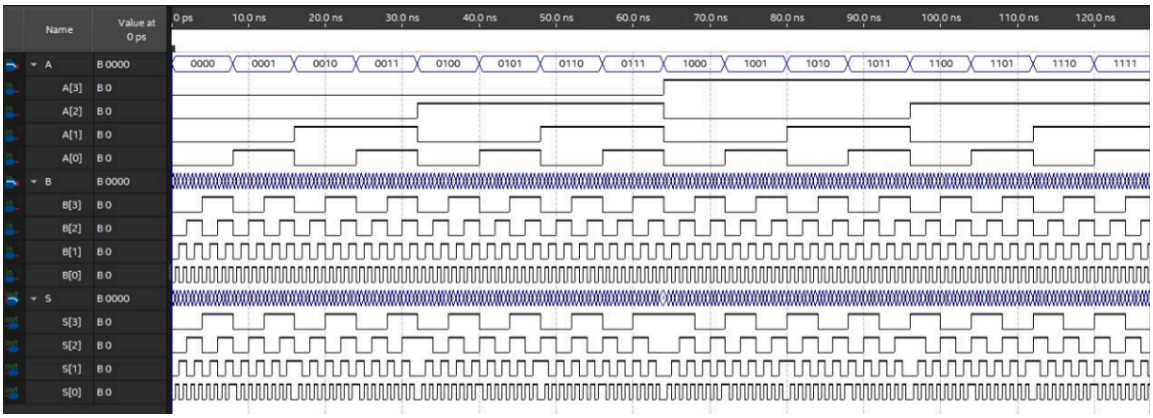

Imagem 10 - Simulação do XOR (4 bits)

5.6 Multiplexador

O multiplexador foi testado para garantir que seleciona corretamente o resultado de cada operação aritmética ou lógica a ser exibido na saída, conforme o código de seleção de 3 bits, sendo a seleção 000 considerada uma *flag Error*. Vale ressaltar que a simulação foi feita em cima do multiplexador de 1 bit, visto que o módulo multiplexador, que instancia o multiplexador de 1 bit oito vezes, possui uma simulação de tamanho que impossibilita abordar inteiramente neste documento e é possível compreender inteiramente o multiplexador apenas pela sua simulação de um único bit, dado que o módulo principal é apenas instâncias desse multiplexador, feito bit a bit. Segue abaixo a tabela verdade (**Imagem 11**) e a simulação do MUX1 (**Imagem 12**):

OP2	OP1	OP0	S
0	0	0	ERR
0	0	1	SUM
0	1	0	SUB
0	1	1	MULT
1	0	0	DIV
1	0	1	AND
1	1	0	OR
1	1	1	XOR

Imagem 10 - Tabela Verdade do Multiplexador (1 bit)

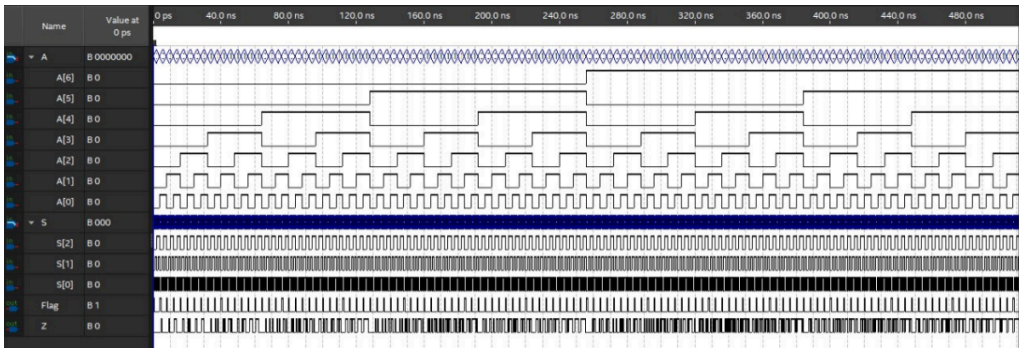

Imagem 12 - Simulação do Multiplexador (1 bit)

5.7 Conversor

O conversor, ou ainda decodificador, foi validado para transformar corretamente os resultados binários pós-multiplexador em sinais que acionam os displays de sete segmentos, garantindo a correta exibição decimal dos valores calculados. Vale relembrar que o display apenas apresenta os quatro bits menos significativos e, assim sendo, apenas representa até a palavra binária com valor decimal equivalente a 15, enquanto os mais significativos apenas acionam a *flag Overflow*, sem anular a representação decimal. Segue abaixo a tabela verdade (**Imagem 13**) e a simulação (**Imagem 14**) desse decodificador instanciado no *CircuitoMain* para melhor visualização:

PALAVRA[3]	PALAVRA[2]	PALAVRA[1]	PALAVRA[0]	DEZENA[3]	DEZENA[2]	DEZENA[1]	DEZENA[0]	UNIDADE[3]	UNIDADE[2]	UNIDADE[1]	UNIDADE[0]
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	1	0	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	0	1	0	1
0	1	1	0	0	0	0	0	0	1	1	0
0	1	1	1	0	0	0	0	0	1	1	1
1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	0	1	0	0	1
1	0	1	0	0	0	0	1	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	1
1	1	0	0	0	0	0	1	0	0	1	0
1	1	0	1	0	0	0	1	0	0	1	1
1	1	1	0	0	0	0	1	0	1	0	0
1	1	1	1	0	0	0	1	0	1	0	1

Imagem 13 - Tabela Verdade do Conversor

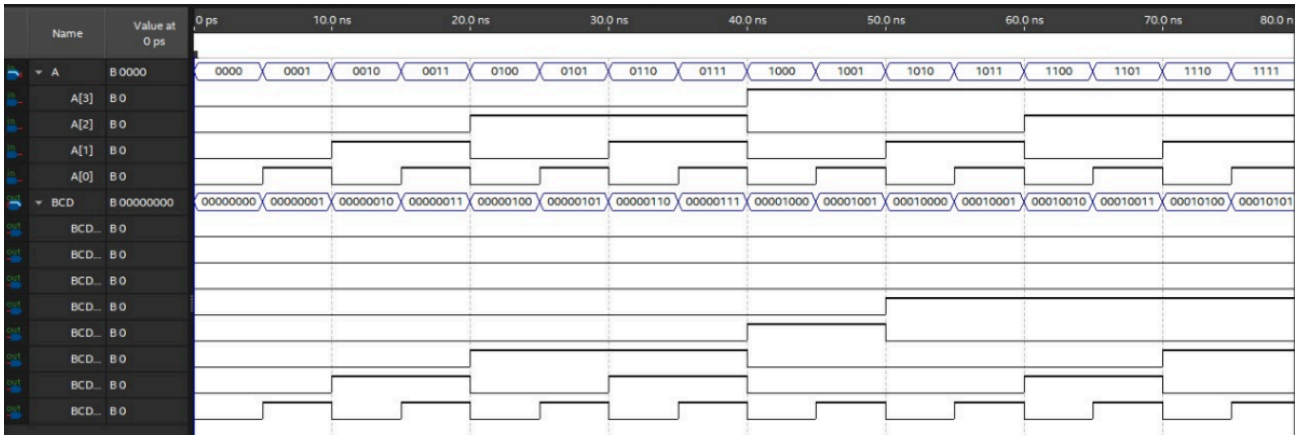

Imagem 14 - Simulação do Conversor de Binário para Decimal

6. CONCLUSÃO

O desenvolvimento deste projeto demonstrou que a aplicação prática da lógica digital permite criar sistemas funcionais de forma estruturada. A decomposição em módulos e a ***abordagem top-down*** e ***incremental*** facilitaram a implementação das operações e a compreensão da interação entre os blocos, evidenciando a importância da modularidade, sendo essenciais para o problema proposto.

Nessa ótica, ainda que a implementação tenha sido apenas em **Verilog combinacional**, com portas lógicas e instâncias de módulos, validada por simulações e tabelas verdade, foi possível assegurar o correto funcionamento do sistema, incluindo propagação de *carry* e *borrow*, detecção de *overflow* e tratamento de erros (*flag Error*).

Portanto, este projeto mostrou a relevância de uma metodologia clara e estruturada, permitindo testes isolados, integração controlada e depuração eficiente. Em sentido amplo, comprova que a combinação de teoria, prática, metodologia e validação sistemática resulta em aprendizado sólido, entrega de um sistema confiável e desenvolvimento de competências essenciais para **Engenharia de Computação**, com potencial de escalabilidade para sistemas digitais mais complexos e vindouros.

7. REFERÊNCIAS

HARRIS, David; HARRIS, Sarah. ***Digital Design and Computer Architecture***. 2. ed. Burlington: Morgan Kaufmann, 2012.

MANO, M. Morris; CILETTI, Michael D. ***Circuitos Digitais e Projeto de Sistemas Digitais***. 5. ed. São Paulo: Pearson, 2013.

PRESSMAN, Roger S. ***Engenharia de Software: uma abordagem profissional***. 7. ed. Porto Alegre: AMGH, 2014.\

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L. ***Sistemas Digitais: Princípios e Aplicações***. 9. ed. São Paulo: Pearson, 2011.