UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

DEPARTAMENTO DE TECNOLOGIA

TEC498 PROJETO DE CIRCUITOS DIGITAIS

PROBLEMA 1: ULA DE 4 BITS

Davi Queiroz e Helena Filemon

Resumo: *Este trabalho apresenta o desenvolvimento de uma Unidade Lógica e Aritmética (ULA) de 4 bits implementada em Verilog e sintetizada na placa DE10-Lite. A solução contempla operações aritméticas (soma, subtração, multiplicação e divisão) e lógicas (AND, OR e XOR), integradas por meio de um multiplexador 8×1. Foram incorporadas flags de estado (Zero, Carry Out, Overflow e Erro) para monitoramento de condições especiais, garantindo maior robustez ao sistema. Os resultados são exibidos em LEDs e displays de sete segmentos, permitindo a validação em tempo real da arquitetura projetada. A ULA desenvolvida demonstrou funcionamento eficiente e modular, sendo facilmente escalável para aplicações digitais mais complexas.*

Palavras-chave: *ULA. Verilog. FPGA. DE10-Lite. circuitos combinacionais.*

1. INTRODUÇÃO

As Unidades Lógicas e Aritméticas (ULAs) constituem o núcleo fundamental do processamento digital em arquiteturas de computadores, microcontroladores e sistemas embarcados. Elas são responsáveis pela execução das operações aritméticas e lógicas elementares, como soma, subtração, multiplicação, divisão, operações booleanas (AND, OR, XOR), bem como pelo gerenciamento de sinais auxiliares, conhecidos como *flags*, que indicam condições especiais de operação, tais como *overflow*, *carry out*, *zero* e *erro*. Dessa forma, o estudo e a implementação de uma ULA representam um exercício essencial para a compreensão prática dos conceitos de álgebra booleana, circuitos combinacionais e arquitetura digital.

No contexto acadêmico, a construção de uma ULA utilizando linguagens de descrição de hardware, como Verilog HDL, possibilita ao estudante a transição entre teoria e prática, promovendo uma melhor assimilação dos fundamentos de sistemas digitais. Além disso, a utilização de dispositivos de hardware reconfigurável, como o FPGA presente na placa DE10-Lite, permite a prototipagem e a validação em tempo real do projeto, garantindo não apenas a simulação, mas também a observação direta do comportamento do circuito em nível físico.

O presente projeto tem como objetivo o desenvolvimento de uma ULA aritmética e lógica de 4 bits, capaz de operar sobre dois operandos de entrada, considerando ainda um sinal de *carry-in*. A solução proposta contempla a implementação de operações aritméticas (soma, subtração, multiplicação e divisão) e lógicas (AND, OR e XOR), além do monitoramento de condições especiais por meio de *flags*. Os resultados serão apresentados de

forma visual através de LEDs e de dois displays de sete segmentos da placa DE10-Lite, proporcionando clareza na interpretação dos valores obtidos.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. CIRCUITOS COMBINACIONAIS

Os circuitos combinacionais constituem a base da lógica digital e são compostos pela interconexão de portas lógicas. O comportamento desses circuitos é definido exclusivamente pela combinação atual de sinais de entrada, sem a dependência de estados passados ou elementos de memória. Isso significa que, para qualquer configuração de entradas, existe uma saída correspondente bem definida, determinada por funções booleanas.

Os circuitos combinacionais são formados pela interligação de portas lógicas. Eles recebem essa classificação porque, a cada instante, o valor lógico da saída é determinado unicamente pela combinação dos sinais aplicados nas entradas. Diferente de outros tipos de circuitos, os combinacionais não possuem memória, ou seja, sua saída é influenciada apenas pelos valores atuais de entrada, sem considerar estados anteriores. (TOCCI et al., 2011).

O projeto de uma ULA de 4 bits proposto neste problema é um exemplo de aplicação de circuitos combinacionais, pois seu funcionamento envolve a implementação de expressões booleanas que definem as operações aritméticas e lógicas, além do controle de sinais auxiliares como as flags.

2.2. PLATAFORMA DE DESENVOLVIMENTO

O FPGA (Field-Programmable Gate Array, em português, Arranjo de porta programável em campo) é um dispositivo lógico programável constituído por uma matriz de portas que podem ser configuradas pelo usuário após a fabricação. Seu elemento lógico básico é a LUT (Look-Up Table), responsável por implementar funções booleanas, e sua tecnologia de programação geralmente se baseia em antifusíveis ou em memória SRAM (FLOYD, 2007).

Para este projeto, a plataforma usada é a DE10-Lite, um kit de desenvolvimento da Intel que incorpora um FPGA e disponibiliza recursos de hardware reconfigurável. Essa placa é amplamente utilizada em ambientes acadêmicos e de pesquisa, pois possibilita a prototipagem rápida de circuitos digitais e a experimentação prática de arquiteturas reconfiguráveis.

No contexto do trabalho, a DE10-Lite será utilizada para receber dois operandos de 4 bits através das chaves, selecionar a operação aritmética ou lógica por meio dos sinais de controle, processar os dados na ULA projetada e apresentar os resultados tanto nos LEDs quanto nos displays de sete segmentos, garantindo a interpretação visual dos valores calculados.

2.3. UNIDADE LÓGICA ARITMÉTICA (ULA)

A ULA constitui um dos blocos fundamentais em processadores e microcontroladores, sendo responsável pela execução das operações matemáticas e lógicas essenciais de maneira rápida e eficiente. No âmbito dos circuitos digitais, a ULA recebe dois

operandos de entrada e, a partir de sinais de controle, realiza a operação selecionada, fornecendo o resultado correspondente na saída.

Em microprocessadores, a ULA é projetada para realizar operações como soma, subtração, multiplicação e divisão, além de funções lógicas aplicadas a dados digitais, obedecendo a um conjunto de instruções definido pela arquitetura do sistema. Uma AULA típica pode ser implementada por meio de centenas de portas lógicas interconectadas, compondo um circuito combinacional capaz de processar dados de forma confiável (FLOYD, 2007).

2.4. MULTIPLEXADOR (MUX)

Um multiplexador digital, também conhecido como seletor de dados, é um circuito lógico capaz de receber múltiplos sinais de entrada e direcionar apenas um deles para a saída em um dado instante. A escolha da entrada a ser transmitida é definida pelas linhas de seleção (ou de endereço), que determinam qual dos sinais disponíveis será comutado para a saída, conforme mostra a Figura 1.

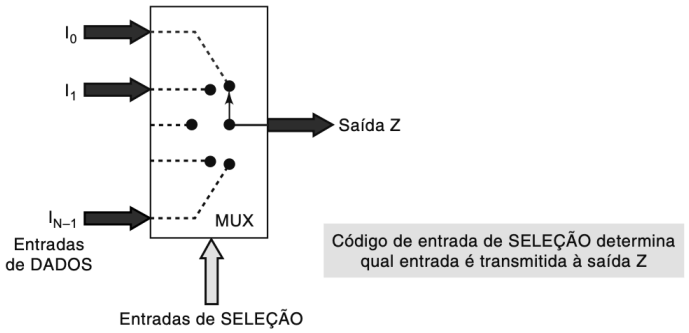


Figura 1 - Diagrama funcional de um MUX digital.
Fonte: TOCCI (2011)

Na prática, o multiplexador funciona como uma chave eletrônica de múltiplas posições controladas digitalmente. O código binário aplicado às entradas de seleção define qual entrada será conectada à saída. Por exemplo, para um determinado valor de seleção, a saída Z pode corresponder à entrada I0; para outro valor, corresponderá a I1, e assim sucessivamente. Dessa forma, o dispositivo realiza o processo de multiplexação, isto é, a transmissão de um entre vários sinais de entrada por meio de um único canal de saída. (TOCCI et al., 2011).

2.5. VERILOG ESTRUTURAL

O Verilog é uma Linguagem de Descrição de Hardware (HDL – Hardware Description Language), cuja finalidade principal é permitir a modelagem, documentação, simulação e síntese automática de circuitos digitais. O Verilog descreve diretamente o comportamento e a estrutura do hardware, permitindo que os projetos sejam implementados em dispositivos como ASICs (Application-Specific Integrated Circuits) e FPGAs.

No Verilog Estrutural, o projetista não apenas define o que o circuito deve fazer, mas detalha como ele é construído. Cada porta lógica, módulo ou componente é instanciado individualmente, e as conexões entre eles são especificadas explicitamente. Assim, o Verilog

estrutural pode ser comparado a um esquema textual que substitui o diagrama de blocos tradicional, permitindo maior escalabilidade e reutilização de módulos.

Por exemplo, ao projetar uma ULA de 4 bits, pode-se estruturar o código instanciando quatro somadores completos (full adders) e conectando-os em cascata para formar um somador de 4 bits. Da mesma forma, multiplexadores, portas lógicas e outros blocos funcionais podem ser instanciados e conectados para compor a arquitetura da ULA.

Em resumo, o Verilog estrutural representa uma forma detalhada e modular de desenvolver circuitos digitais, permitindo ao projetista especificar com precisão a interligação entre componentes básicos e garantindo a fidelidade entre a descrição textual e o hardware implementado.

2.6. QUARTUS II

O Quartus II é um Software de Automação de Projeto Eletrônico (EDA) essencial para desenvolver hardware em FPGA. Sua função é gerenciar todo o fluxo de projeto, convertendo o código de descrição de hardware (como Verilog) em uma configuração física para o chip. Isso envolve a síntese do código em lógica, o mapeamento tecnológico para os blocos internos da FPGA e o roteamento das conexões. Em resumo, ele transforma a descrição lógica de alto nível da ULA em um arquivo de programação que configura o dispositivo real.

2.7. DISPLAY DE SETE SEGMENTOS

O display de sete segmentos é um dispositivo eletrônico amplamente utilizado para a representação visual de números decimais em sistemas digitais. Ele é composto por sete segmentos de LEDs ou cristais líquidos, organizados em formato de barra (Figura 2), que podem ser acionados de maneira combinada para formar os dígitos de 0 a 9. De acordo com Tocci (2011), esse tipo de display apresenta a vantagem de simplificar a interface homem-máquina em circuitos digitais, permitindo a exibição direta de resultados numéricos a partir de sinais binários processados pelo sistema lógico. Dessa forma, o display de sete segmentos constitui um recurso fundamental em projetos didáticos e práticos de eletrônica digital.

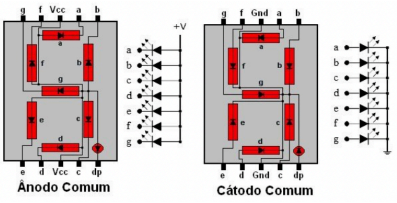


Figura 2 - Display de sete segmentos com informação decimal.
Fonte: LEDS. (2025)

3. CARACTERIZAÇÃO DA SOLUÇÃO

A solução proposta consiste no desenvolvimento de uma Unidade Lógica e Aritmética (ULA) de 4 bits implementada em hardware reconfigurável, atendendo às diretrizes estabelecidas no enunciado do projeto. O circuito foi estruturado de forma modular,

contemplando blocos aritméticos (somador, subtrator, multiplicador e divisor) e blocos lógicos (operações AND, OR e XOR), conforme representado no diagrama da Figura 3.

A arquitetura da ULA recebe como entradas dois operandos de 4 bits, representados como Entrada A e Entrada B, além de um sinal de Carry In proveniente de operações anteriores. Os resultados das operações são conduzidos para um conjunto de multiplexadores de 8×1, controlados por um seletor de 3 bits. Esse seletor é responsável por determinar qual operação será executada, roteando o resultado correto para a saída.

A saída do processamento é disponibilizada em dois formatos complementares: (i) em forma binária, diretamente nos LEDs da placa, permitindo a visualização detalhada dos bits resultantes, e (ii) em formato decimal, por meio de um decodificador para displays de sete segmentos, o que facilita a interpretação do valor obtido pelo usuário. Dessa forma, garante-se tanto a análise técnica dos resultados quanto a usabilidade no contexto educacional.

Adicionalmente, o projeto contempla a implementação de flags de estado, fundamentais para a correta interpretação das operações aritméticas e lógicas. Entre elas, destacam-se: a flag Overflow (OV), que indica estouro de capacidade de representação; a flag Zero (Z), acionada quando o resultado da operação é igual a zero; a flag Carry Out (COUT), que sinaliza transporte ou empréstimo em operações aritméticas; e a flag Erro (ERR), utilizada para sinalizar condições inválidas, como divisões por zero.

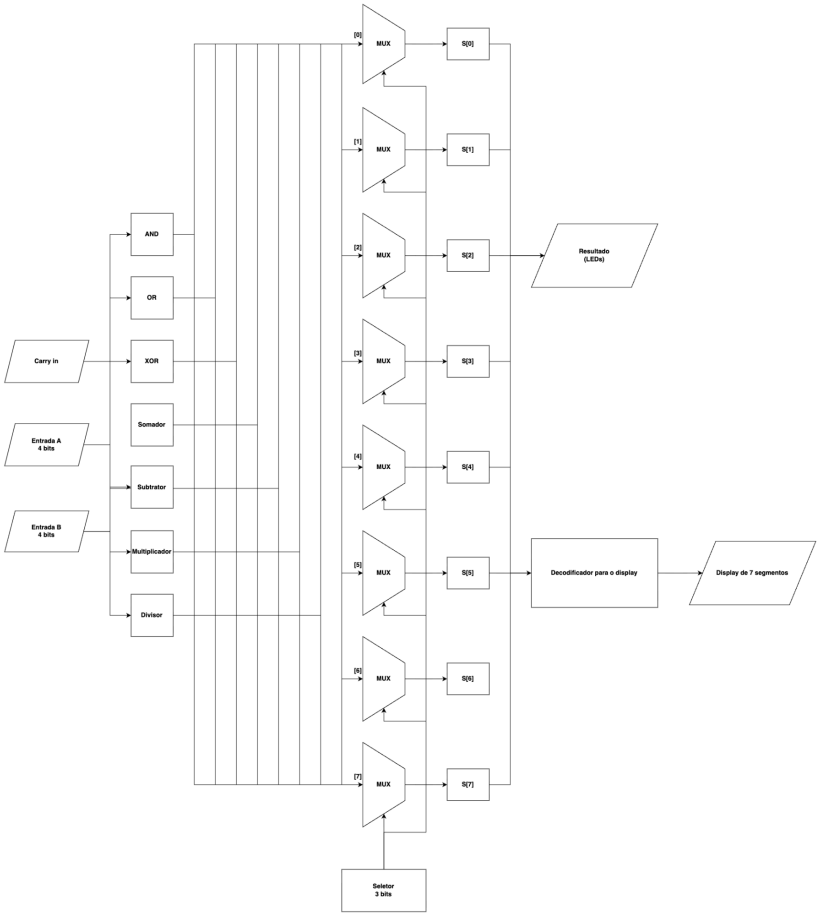


Figura 3 - Diagrama da Solução.
Fonte: autores.

3.1. INTERFACES DE ENTRADA E SAÍDA

No que se refere às interfaces de entrada e saída do circuito, foram empregadas chaves digitais para a inserção de dados e LEDs, em conjunto com dois displays de sete segmentos, para a visualização dos resultados, conforme ilustrado na Figura 4. O sistema recebe como entrada duas palavras binárias de quatro bits, fornecidas pelas chaves, além do código de operação que determina a operação aritmética ou lógica a ser executada (adição, subtração, multiplicação, divisão, AND, OR ou XOR). A saída do processamento é disponibilizada em dois formatos: nos LEDs, que exibem o resultado em representação binária, e nos displays de sete segmentos, que apresentam o valor em formato decimal, possibilitando uma interpretação mais clara e imediata por parte do usuário. Adicionalmente, o circuito implementa flags de estado, que sinalizam condições relevantes do processamento: a flag Overflow (OV) indica quando o resultado excede a capacidade de representação dos bits disponíveis; a flag Zero (Z) é ativada quando o resultado da operação é igual a zero; a flag Carry Out (COUT) informa a ocorrência de transporte ou empréstimo em operações aritméticas; e a flag Erro (ERR) é utilizada para indicar condições inválidas, como tentativas de divisão por zero.

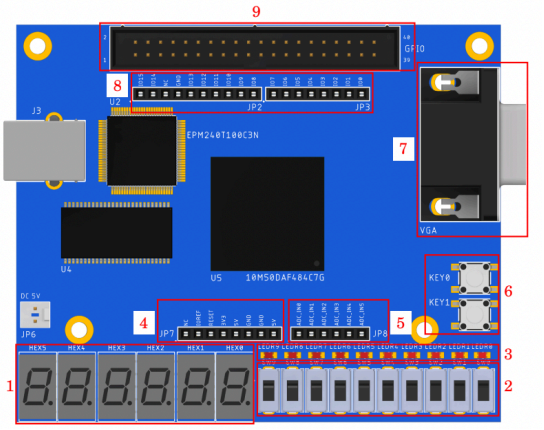


Figura 4 – Placa DE10-Lite.
Fonte: LEDS (2025).

3.2. OPERAÇÕES ARITMÉTICAS

As operações aritméticas foram implementadas em módulos específicos para soma, subtração, multiplicação e divisão. A soma utiliza um somador de 4 bits capaz de detectar transporte (Carry Out), enquanto a subtração conta com um subtrator que sinaliza resultados negativos. A multiplicação gera uma saída de até 8 bits, e a divisão inclui um sistema de detecção de erros para casos de divisão por zero. Todos os resultados dessas operações são encaminhados ao barramento final por meio de um multiplexador 8×1, controlado pelo vetor de seleção *select*.

3.2.1. Soma

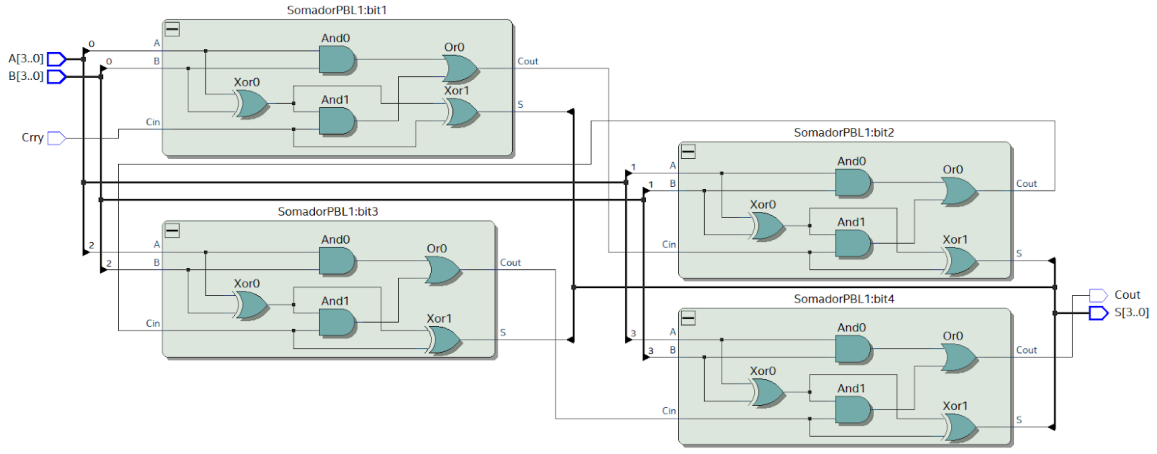


Figura 5 - RTL View do somador.
Fonte: autores.

O circuito do somador de 4 bits funciona conectando quatro módulos de somador completo de 1 bit (Full Adder) em cascata, conforme a Figura 5. Cada módulo de 1 bit soma dois bits de entrada (A e B) e um carry-in (Cin), gerando um bit de soma (S) e um carry-out (Cout). O Cout de um módulo é ligado ao Cin do módulo seguinte, propagando o "vai-um" da direita para a esquerda, até que o último módulo gere o resultado final de 4 bits (S[3:0]) e o carry-out global (Cout).

A	B	Cin	Cout	S (binário)	S (decimal)
0000	0000	0	0	0000	0
0000	0000	1	0	0001	1
…					
0001	0001	0	0	0010	2
0001	0001	1	0	0011	3
…					
1111	1111	0	1	1110	30
1111	1111	1	1	1111	31

Tabela 1 - Tabela-Verdade do Somador.
Fonte: autores.

3.2.2. Subtração

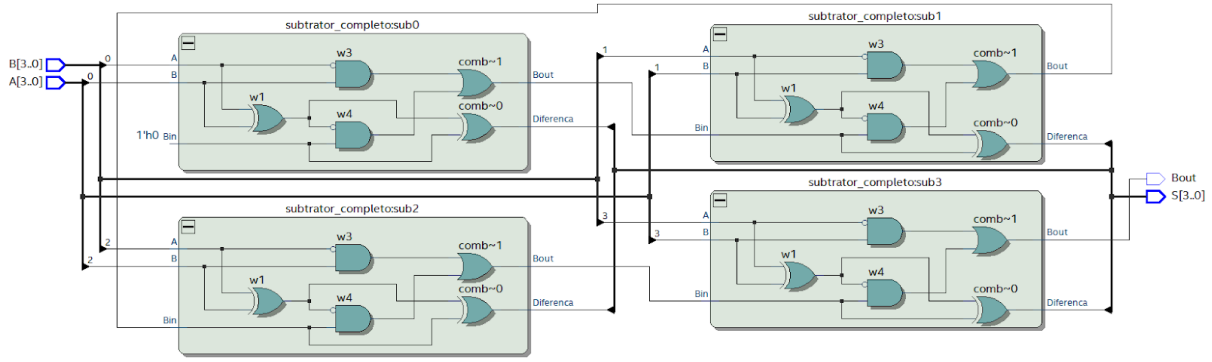


Figura 6 - RTL View do somador.
Fonte: autores.

O circuito implementa um subtrator de 4 bits, utiliza quatro módulos de subtrator completo de 1 bit em cascata, conforme a Figura 6. Cada subtrator de 1 bit realiza a operação de subtração em um par de bits, considerando o empréstimo (Bin) do estágio anterior, e gerando um bit de Diferença e um novo empréstimo de saída (Bout). A conexão em série garante que o empréstimo (Bout) de cada estágio seja propagado como o empréstimo de entrada (Bin) para o próximo bit mais significativo. A entrada inicial de empréstimo é fixada em '0', e o último Bout representa o empréstimo final da operação de 4 bits, indicando se o resultado é negativo.

A	B	Bout	S (binário)	S (decimal)
0000	0000	0	0000	0
…				
0001	0001	0	0000	0
…				
0010	0001	0	0001	1
…				
1111	1111	1	0000	0

Tabela 2 - Tabela-Verdade do Subtrator.
Fonte: autores.

3.2.3. Multiplicação

Foi implementado um multiplicador de 4 bits × 4 bits, capaz de gerar resultados de até 8 bits. O funcionamento baseia-se no método clássico de multiplicação binária, em que cada bit do operando A é combinado, por meio de portas lógicas AND, com os bits do operando B. Esses produtos parciais formam um arranjo semelhante ao da multiplicação manual em base decimal, porém adaptado ao sistema binário.

O processo utiliza o algoritmo de deslocamento e soma para combinar os produtos parciais, conforme a Figura 7. A implementação exige 16 portas AND, 4 meios somadores e 8 somadores completos, totalizando 12 somadores destinados à etapa de adição. (RAGINI et al., 2016).

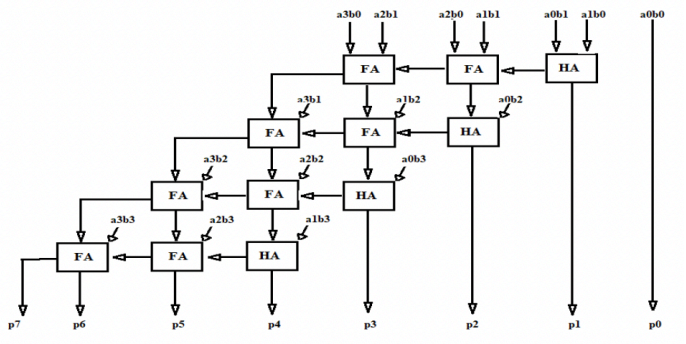


Figura 7 - Diagrama do multiplicador.
Fonte: RAGINI (2016).

Após a geração dos produtos parciais, eles são alinhados em colunas e somados por meios somadores e somadores completos, garantindo o correto posicionamento dos bits e o tratamento dos sinais de transporte (carry). O módulo é estruturado de forma hierárquica, utilizando instâncias dos componentes Meiosomador e SomadorPBL1.

O resultado é disponibilizado no barramento de saída S[7:0], no qual S[0] corresponde ao bit menos significativo e S[7] ao mais significativo. Essa largura é necessária, pois a multiplicação de dois operandos de 4 bits pode atingir até 225 em decimal (15 × 15), valor que exige 8 bits para sua representação binária.

A	**B**	**S (binário)**	**S (decimal)**
0000	0000	00000000	0
…			
0000	1111	00000000	0
…			
0001	0001	00000001	1
…			
1010	0101	00110010	50
…			
1111	1111	11100001	225

Tabela 3 - Tabela-Verdade do Multiplicador.
Fonte: autores.

3.2.4. Divisão

O módulo principal do divisor recebe como entradas dois operandos de 4 bits, A (dividendo) e B (divisor), e fornece uma saída de 4 bits S, correspondente ao quociente da operação, além do sinal Error, que indica uma tentativa de divisão por zero. A detecção do erro é feita verificando se todos os bits de B são iguais a zero. Para isso, são utilizados inversores em cada bit de B, cujas saídas são combinadas em uma porta AND, de modo que o sinal Error é ativado sempre que o divisor for nulo.

O cálculo do quociente é realizado de forma totalmente combinacional, por meio de quatro submódulos (divisor0, divisor1, divisor2 e divisor3), tal qual é mostrado na Figura 8. Cada um deles é responsável por determinar um bit específico do resultado, desde o menos significativo (S[0]) até o mais significativo (S[3]). Internamente, esses módulos utilizam portas NOT para gerar os complementos de cada bit de A e B, e um grande número de portas AND para representar todas as condições possíveis em que o bit correspondente do quociente deve assumir valor lógico 1. Em seguida, todas essas condições são unidas por portas OR, de modo que a saída final de cada submódulo é ativada sempre que qualquer uma das combinações válidas for satisfeita.

Para a obtenção das expressões lógicas, foi realizada a tradução direta da tabela verdade da operação de divisão para lógica combinacional. As simplificações foram feitas utilizando o método SOP (Sum of Products), com o auxílio de um resolvedor de mapas de Karnaugh disponível online (COLEMAN, 2023).

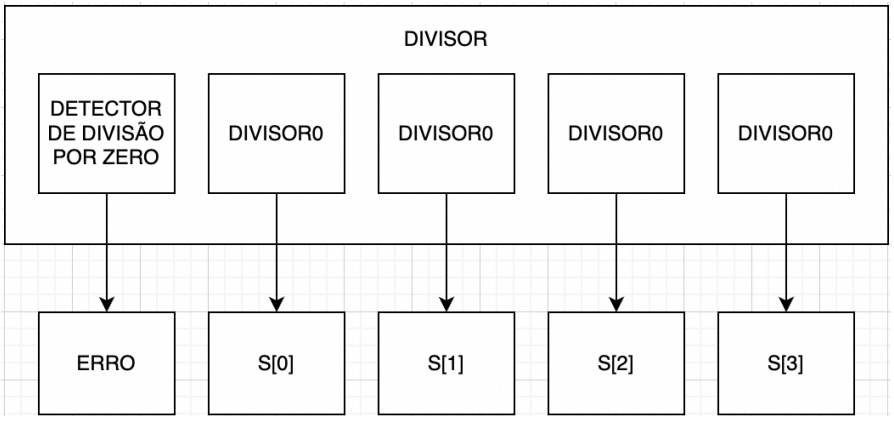


Figura 8 - Diagrama do divisor.
Fonte: autores.

Dessa forma, o divisor utiliza a tradução direta da tabela verdade da operação de divisão para lógica combinacional. Essa abordagem garante que o resultado seja produzido imediatamente em função das entradas, sem necessidade de ciclos de clock ou registros intermediários. Entre as limitações desta solução, destaca-se o fato de ele operar apenas com operandos de 4 bits, restringindo o resultado a valores de 0 a 15. Além disso, o módulo retorna apenas o quociente da divisão, sem fornecer o resto da operação.

A	B	S
0000	0000	0000
…		
1010	0000	Erro
…		
1010	0101	0010
…		
1111	0010	0111
…		
1111	1111	0001

Tabela 4 – Tabela-Verdade do divisor.
Fonte: autores.

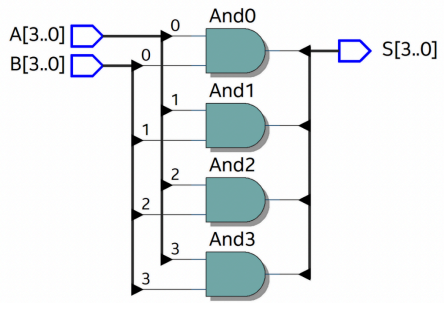
3.3. OPERAÇÕES LÓGICAS

Os módulos dos operadores lógicos básicos (AND, OR e XOR) foram aplicados de forma bit a bit entre dois vetores de 4 bits. Em todos os casos, as entradas A e B representam números binários de 4 bits, e a saída S também é um vetor de 4 bits. Cada bit da saída é obtido aplicando a operação lógica correspondente entre os bits de mesma posição das entradas. Dessa forma, o circuito realiza as operações lógicas em paralelo, garantindo que todas as posições sejam processadas simultaneamente.

No módulo opLogicoAND, a saída é obtida através da operação E lógico. Cada bit da saída S[i] só assume o valor 1 quando ambos os bits de entrada A[i] e B[i] são iguais a 1. Em qualquer outra combinação, a saída será 0. Isso torna o operador útil para situações em que se deseja identificar condições em que duas variáveis são verdadeiras ao mesmo tempo, funcionando como um filtro que mantém em 1 apenas os bits em comum entre os dois vetores, como visto na Figura 9.

A	B	S
0000	0000	0000
0011	1100	0000
0011	0011	0011
1111	1111	1111

(5)



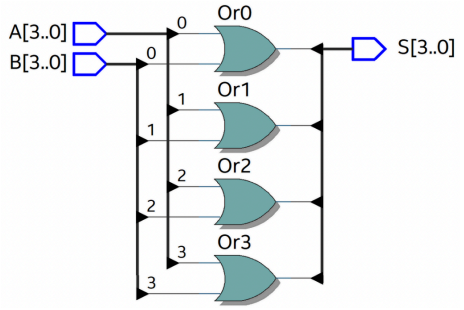
(9)

Tabela 5 - Tabela-Verdade do operador lógico AND. Figura 9: Diagrama circuito AND. Fonte: autores.

O módulo opLogicoOR utiliza a operação OU lógico. Nesse caso, cada bit da saída S[i] é igual a 1 sempre que ao menos um dos bits de entrada A[i] ou B[i] for igual a 1. Assim, o resultado reflete a união das condições representadas pelas duas entradas, como visto na Figura 10. Esse tipo de operação é muito utilizado quando é necessário verificar a ocorrência de pelo menos uma condição verdadeira, sem a necessidade de que ambas sejam satisfeitas.

A	B	S
0000	0000	0000
0011	1100	1111
0011	0011	0011
1111	1111	1111

(6)



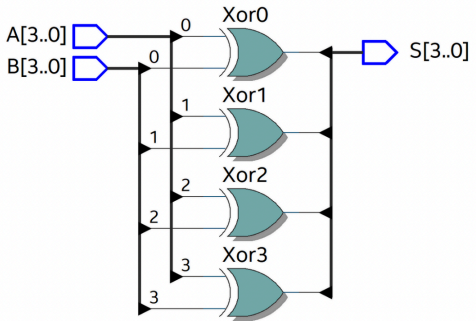
(10)

Tabela 6 - Tabela-Verdade do operador lógico OR. Figura 10: Diagrama circuito OR. Fonte: autores.

Já o módulo opLogicoXOR implementa a operação OU exclusivo. Diferentemente do OR simples, o resultado S[i] será 1 apenas quando os bits de entrada forem diferentes, isto é, quando um deles for 1 e o outro 0 (Figura 11). Caso ambos sejam iguais (00 ou 11), a saída será 0. Essa característica torna o XOR bastante útil em aplicações que envolvem comparação entre sinais, detecção de diferenças e até operações aritméticas como a soma binária.

A	B	S
0000	0000	0000
0011	1100	1111
0011	0011	0000
1111	1111	0000

(7)



(11)

Tabela 7 - Tabela-Verdade do operador lógico XOR. Figura 10: Diagrama circuito XOR. Fonte: autores.

3.4. FLAGS

3.4.1. Zero

A Flag Zero é um sinal que se ativa quando o resultado da operação da ULA é zero. No código, o módulo zero flag gera o sinal interno s_zero ao inspecionar se todos os bits da saída final (outs[7:0]) são zero. Na placa, o zero é indicado pelo LEDR9 aceso.

3.4.2. Erro

A Flag Erro é um sinal de status que alerta o sistema sobre uma condição de erro que impede o processamento normal, como uma Divisão por Zero, erro que é detectado e sinalizado internamente pelo módulo divisor DV através do sinal s_err. Na placa, o erro é indicado pelo LEDR8 aceso.

3.4.3. Carry out

A Flag Carry Out (s_add[4]), proveniente do módulo somadorde4bits SM, armazena o vai-um gerado na adição mais significativa. Seu uso é duplo: ela indica um overflow em operações com números não sinalizados (quando o resultado excede 4 bits) e é crucial para realizar operações com maior número de bits, pois serve como o carry-in para uma ULA adjacente. Na placa, o carry out é indicado pelo LEDR4 aceso.

3.4.4. Overflow

A Flag Overflow indica que o resultado de uma operação aritmética excedeu a faixa de valores representáveis para números sinalizados (representados em complemento de dois). Este status é vital para detectar resultados incorretos ao se trabalhar com valores positivos e negativos. O Overflow é logicamente calculado pelo XOR entre o Carry In e o Carry Out do bit mais significativo, sendo um sinal que o sistema deve usar para alertar sobre a impossibilidade de armazenar o resultado correto dentro da limitação de bits da arquitetura de 4 bits. Na placa, o overflow (OF) é indicado pelo display de sete segmentos.

3.5. MUX

No código do MUX, existem oito entradas (I0 a I7), três sinais de seleção (S0, S1 e S2), e uma saída única (Y). Cada combinação dos sinais de seleção corresponde a uma das entradas. Por exemplo, quando os seletores estão em 000, a saída será igual a I0; quando estão em 001, a saída será I1; e assim por diante até a combinação 111, que seleciona I7. Para implementar isso, o código gera os sinais negados dos seletores (not_S0, not_S1 e not_S2) e, em seguida, utiliza portas AND para ativar apenas a entrada correspondente à combinação desejada. Cada porta AND combina uma entrada de dados com a configuração correta dos seletores, garantindo que apenas uma delas esteja ativa em cada instante. Finalmente, todas as saídas das portas AND são reunidas por uma porta OR, que entrega na saída Y o valor da entrada selecionada, conforme mostrado na Figura 12.

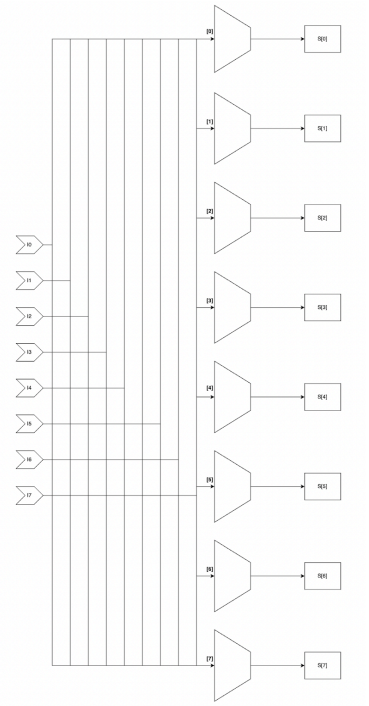


Figura 12 - Diagrama do funcionamento do multiplexador.
Fonte: autores.

Na parte final do código, o mux 8x1 é instanciado repetidas vezes para formar um barramento de 8 bits (outs[7:0]). Cada instância trata de um bit específico dos resultados de operações como divisão (s_div), operações lógicas (s_and, s_or, s_xor), soma (s_add), subtração (s_sub) e multiplicação (s_mult). Os sinais de seleção (selet[2:0]) determinam qual operação será escolhida. Por exemplo, se os seletores estiverem configurados para escolher a operação de soma, os multiplexadores encaminharão os bits do resultado da soma (s_add) para as saídas correspondentes.

Seletor	**Saída (Y)**
000	I0 (divisão)
001	I1 (AND)
010	I2 (OR)
011	I3 (XOR)
100	I4 (soma)
101	I5 (subtração)
110	I6 (multiplicação)
111	-

Tabela 8 - Tabela do MUX.
Fonte: autores.

3.6. DECODIFICADOR PARA O DISPLAY DE 7 SEGMENTOS

O Display de Sete Segmentos é um dispositivo de saída comum, utilizado para exibir informações numéricas ou alfanuméricas simples, composto por sete LEDs ou segmentos que podem ser ligados ou desligados individualmente. No contexto do projeto, este componente é a interface visual da ULA, permitindo que os resultados binários sejam apresentados em formato hexadecimal ou decimal, facilmente compreendido pelo usuário. Para que o display exiba um número corretamente, os bits de saída da ULA devem ser convertidos pelo decodificador. Os sinais la0 a lg0 e la1 a lg1 no código são as saídas do decodificador que controlam diretamente os dois displays de sete segmentos, permitindo a visualização dos 8 bits de resultado (outs) da ULA.

A saída final da ULA é um valor binário. No entanto, para exibir esse resultado de forma legível em um display de sete segmentos, é necessário um circuito de conversão. O Decodificador para Display de Sete Segmentos tem a função de traduzir o código binário em sinais lógicos que ativam os segmentos corretos (a, b, c, d, e, f, g) do display. O módulo decodificador_display do projeto realiza essa conversão, tomando os bits de outs e gerando os sinais (la0 a lg1, e la1 a lg1) necessários para controlar dois displays. Este componente é essencialmente um circuito lógico combinacional projetado para mapear cada código de entrada para o padrão de iluminação dos segmentos que representa o dígito correspondente.

Entrada Binária	**Segmentos ativos (a, b, c, d, e, f, g)**	**Dígito no Display**
0000	0000001	0
0001	1001111	1
0010	0010010	2
0011	0000110	3
0100	1001100	4
0101	0100100	5
0110	010000	6
0111	0001111	7
1000	0000000	8
1001	0000100	9

Tabela 9 - Tabela do decodificador do display 7 segmentos.
Fonte: autores.

4. CASOS DE TESTE

Nesta seção são apresentados os principais casos de teste feitos e testados pelo próprio quartus, com a utilização do Waveform, mostrando os resultados obtidos. A entrada A é de ins[0] até ins[3], a entrada B é de ins[4] até ins[7]. No waveform, os 8 bits que mostram na linha “ins” devem ser divididos em duas partes de 4 bits, para então simular as entradas. A linha “selet” é a combinação utilizada no MUX para escolher a operação que será mostrada. A linha “outs” mostra o resultado da operação feita entre as entradas A e B. A linha “zero” representa a flag zero. A linha krry é o bit de carry in na soma.

4.1. Operação de Soma

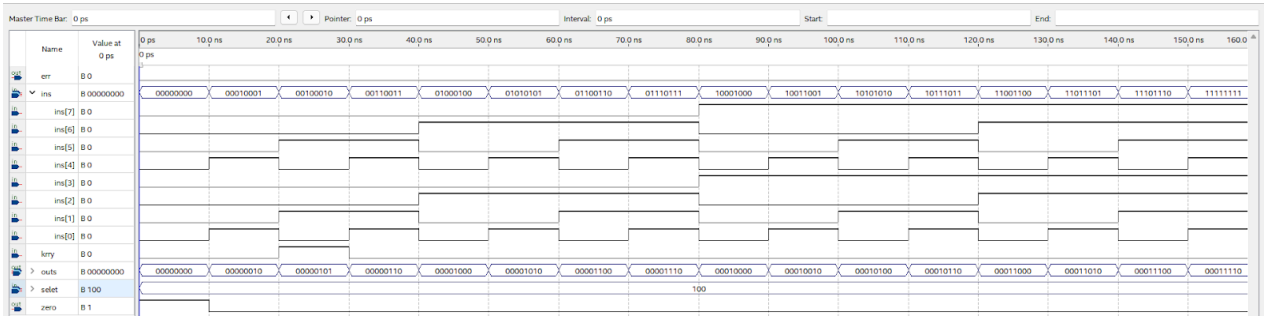


Figura 13 - Waveform da operação de soma.
Fonte: autores.

4.2. Operação de Subtração

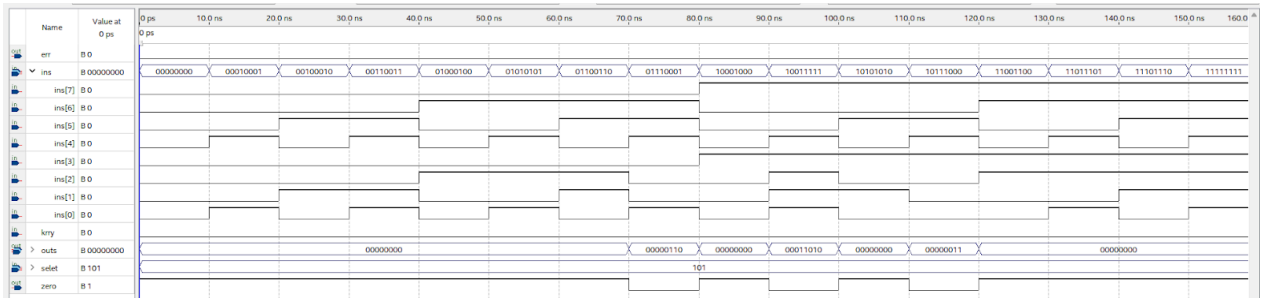


Figura 14 - Waveform da operação de subtração.
Fonte: autores.

No caso que aconteceu no período entre 90 a 100 ns, o borrow out saiu como sinal em nível lógico alto, mostrando que o resultado é negativo, e a ULA não tem capacidade de mostrar o resultado corretamente.

4.3. Operação de Multiplicação

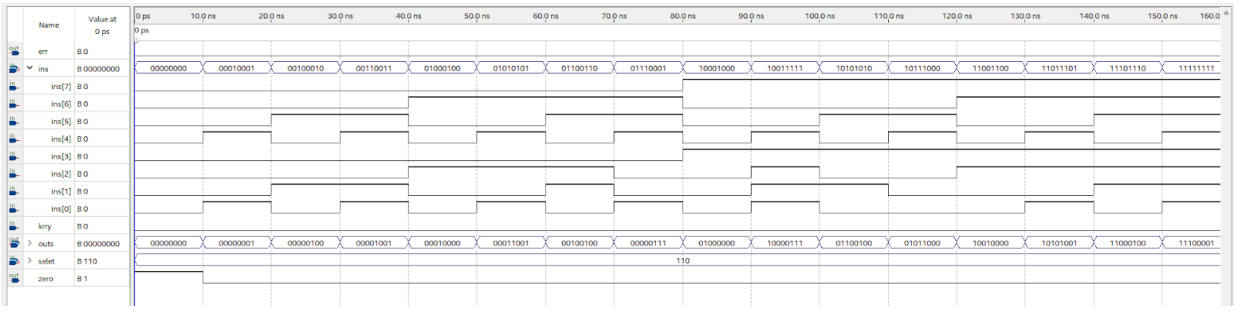


Figura 15 - Waveform da operação de multiplicação.
Fonte: autores.

4.4. Operação de Divisão

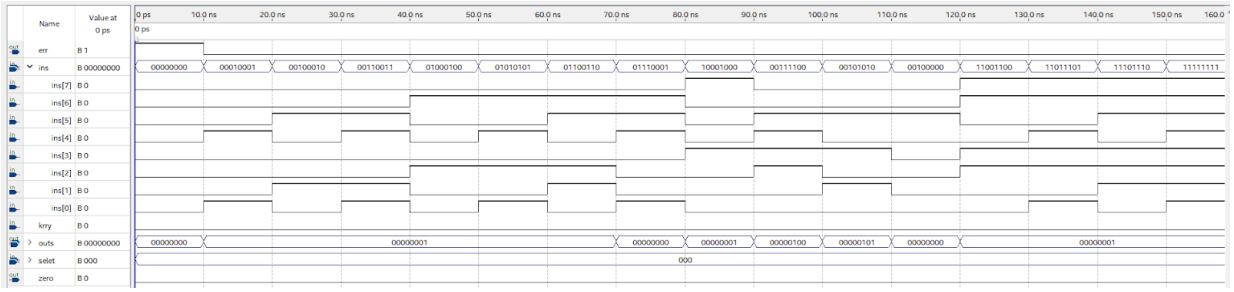


Figura 16 - Waveform da operação de divisão.
Fonte: autores.

4.5. Operação Lógica OR

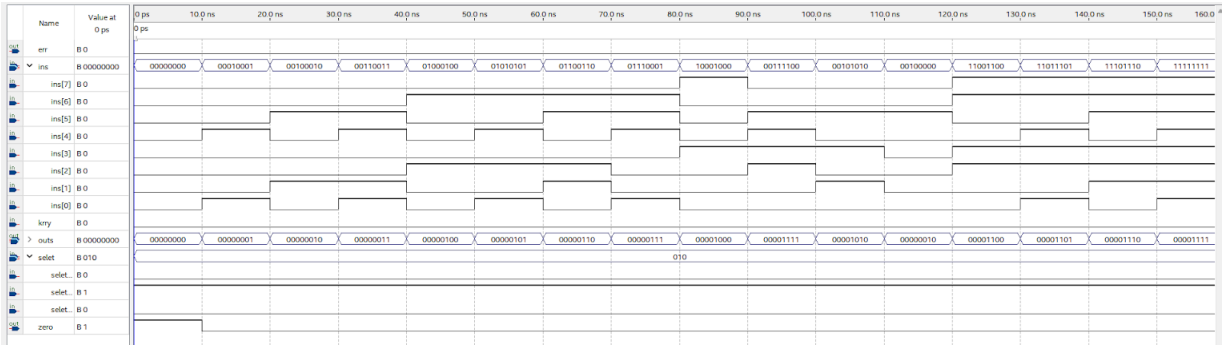


Figura 17 - Waveform da operação lógica AND.
Fonte: autores.

4.6. Operação Lógica AND

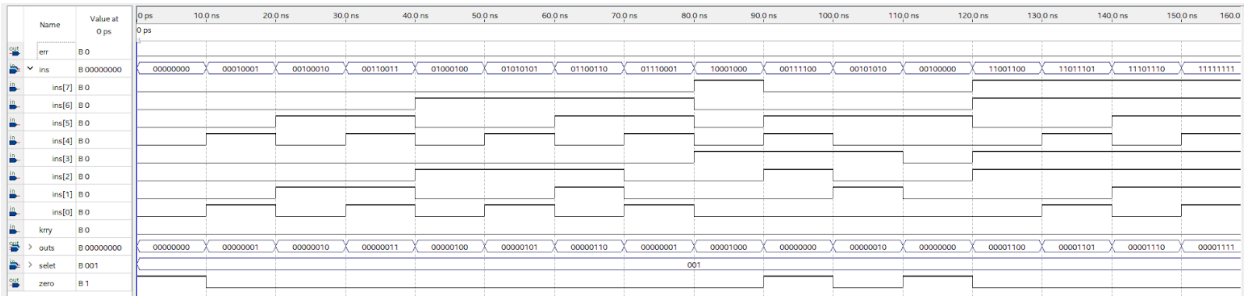


Figura 18 - Waveform da operação lógica OR.
Fonte: autores.

4.7. Operação Lógica XOR

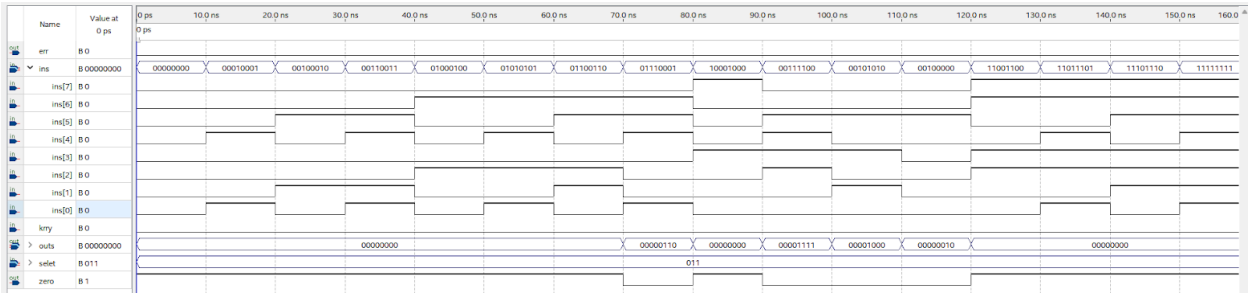


Figura 19 - Waveform da operação lógica XOR.
Fonte: autores.

5. CONCLUSÃO

A implementação da ULA de 4 bits em Verilog, sintetizada na placa DE10-Lite, resultou em uma solução funcional e eficiente para a execução de operações aritméticas (soma, subtração, multiplicação e divisão) e lógicas (AND, OR e XOR). A arquitetura proposta utilizou módulos dedicados a cada operação, integrados por meio de um multiplexador 8×1, garantindo flexibilidade na seleção da função desejada.

As condições especiais de execução foram tratadas por meio das flags de Zero, Carry Out, Overflow e Erro, que ampliam a confiabilidade do sistema ao sinalizar situações como resultados nulos, transporte em operações aritméticas, excedentes na representação em complemento de dois e divisões inválidas.

O uso do FPGA como plataforma de implementação assegurou o correto funcionamento em tempo real, permitindo validar não apenas o comportamento lógico das operações, mas também a integridade da comunicação entre entradas, processador da ULA e saídas visuais. A exibição dos resultados em LEDs e displays de sete segmentos consolidou a interpretação prática dos dados de saída, inclusive para resultados de até 8 bits oriundos da multiplicação.

Conclui-se que a solução atendeu plenamente aos requisitos do projeto, entregando uma ULA compacta, modular e confiável, capaz de executar operações fundamentais de processamento digital e facilmente expansível para arquiteturas de maior largura de dados.

REFERÊNCIAS

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. *Sistemas Digitais: Princípios e Aplicações*. 11ª ed. Pearson, 2011. ISBN 978-85-4300-694-9.

FLOYD, T. L. *Sistemas digitais: Fundamentos e Aplicações*. 9ª ed. Bookman, 2007. ISBN 978-85-7780-107-7

RAGINI, K.; NEERAJAKSHI, B. Design and Implementation of 4x4 bit Multiplier using Dadda Algorithm. *International Journal of Engineering Research & Technology (IJERT)*, v. 5, n. 12, p. 1–5, 2016.

COLEMAN, Charlie. *K-map solver*. Disponível em: https://www.charlie-coleman.com/experiments/kmap. Acesso em: 29 de setembro de 2025.