Unidade Lógica e Aritmética de 8 Bits

Thiago Cordeiro de Melo[1]

[1]Escola Superior de Tecnologia - Universidade do Estado do Amazonas (UEA)
Amazonas – AM – Brazil

tcdm.eng23@uea.edu.br

Abstract. *This paper presents the development of an 8-bit Arithmetic Logic Unit (ALU) implemented in Verilog. The ALU accepts two 8-bit operands and performs one of four selectable operations: addition, AND, OR, and NOT. It also provides status flags indicating result-specific conditions: carry (C), overflow (OV), zero (Z), and negative (N). A comprehensive testbench was developed to validate the functionality of all operations and input combinations. The implementation aims to demonstrate fundamental processor architecture concepts and serves as a foundation for more advanced digital systems design*

Resumo. *Este artigo apresenta o desenvolvimento de uma Unidade Lógica e Aritmética (ULA) de 8 bits implementada em Verilog. A ULA recebe dois operandos de 8 bits e realiza uma das quatro operações selecionáveis: soma, AND, OR e NOT. A unidade também fornece sinais de status (flags) que indicam condições específicas do resultado: carry (C), overflow (OV), zero (Z) e negativo (N). O projeto foi testado com um banco de testes abrangente que valida todas as operações e combinações de entrada. A implementação visa demonstrar conceitos fundamentais de arquitetura de processadores e serve como base para projetos mais complexos em sistemas digitais.*

1. Introdução

As Unidades Lógicas e Aritméticas (ULAs) representam componentes essenciais na arquitetura de processadores, sendo responsáveis por realizar operações matemáticas e lógicas fundamentais. A construção de uma ULA constitui uma etapa importante na formação acadêmica de estudantes das áreas de Engenharia da Computação e Ciências da Computação, por permitir o entendimento prático do funcionamento interno de processadores.

Neste trabalho, é apresentada a implementação de uma ULA de 8 bits utilizando a linguagem de descrição de hardware Verilog. A ULA foi projetada para realizar quatro operações básicas: soma, operação lógica AND, operação lógica OR e negação (NOT) de um dos operandos. Além do resultado da operação, a unidade também fornece quatro flags de estado: Carry (C), Overflow (OV), Zero (Z) e Negativo (N), que indicam condições específicas do resultado da operação aritmética ou lógica.

O objetivo principal deste artigo é demonstrar o funcionamento da ULA proposta, evidenciar a importância das flags na interpretação dos resultados e validar seu comportamento por meio de um testbench abrangente. A proposta é de fácil compreensão e pode ser utilizada como base para a construção de unidades mais complexas, como uma ULA com suporte a subtração, operações shift ou comparação.

2. Funcionamento da ULA

A Unidade Lógica e Aritmética (ULA) desenvolvida neste trabalho possui uma largura de 8 bits e opera sobre dois operandos de entrada, denominados a e b. A operação executada é determinada por duas entradas de controle, x e y, que definem qual das quatro funções será aplicada aos operandos. A Tabela 1 resume o comportamento da ULA com base na combinação das entradas de controle.

Tabela 1. Operações da ULA 8 bits

x	y	Operação
0	0	$a + b$ (soma aritmética)
0	1	$a \,\&\, b$ (AND bit a bit)
1	0	$a \mid b$ (OR bit a bit)
1	1	$\sim a$ (NOT de a)

A saída principal da ULA é o vetor s, contendo o resultado da operação selecionada. Além disso, a unidade gera quatro sinais auxiliares conhecidos como flags, utilizados para sinalizar condições especiais do resultado:

Carry (c): indica se houve um transbordamento no bit mais significativo durante uma operação de soma (bit 8 do somador estendido).

Overflow (ov): indica se houve estouro de sinal na operação de soma com operandos em complemento de dois, ou seja, quando o resultado ultrapassa o intervalo representável com 8 bits assinados.

Zero (z): indica que o resultado da operação foi igual a zero.

Negativo (n): reflete o valor do bit mais significativo da saída (s[7]), utilizado em operações com números em complemento de dois para indicar se o resultado é negativo.

Os sinais "c" e "ov" são atualizados apenas durante a operação de soma, enquanto os sinais "z" e "n" são avaliados após qualquer operação. A utilização desses flags é fundamental em arquiteturas baseadas em instruções condicionais, pois permite que o processador tome decisões com base nos resultados das operações aritméticas e lógicas.

A implementação também faz uso de um somador estendido de 9 bits internamente para a detecção precisa do sinal de Carry, além de uma expressão lógica para a detecção do Overflow, que considera os sinais dos operandos e do resultado.

3. Implementação em Verilog

O módulo `ula_8b` implementa uma Unidade Lógica Aritmética de 8 bits.

Listing 1. Módulo ULA 8 Bits

```
module ula_8b (
    input [7:0] a, b,
    input x, y,
    output reg [7:0] s,
    output reg c, ov, z, n
);

    wire [8:0] soma_ext;      // Para detectar carry
    assign soma_ext = a + b;

    always @(a, b, x, y) begin
        // Operação principal
        case ({x, y})
            2'b00: s = a + b;
            2'b01: s = a & b;
            2'b10: s = a | b;
            2'b11: s = ~a;
        endcase

        // Flags
        if ({x, y} == 2'b00) begin // Caso {soma}
            c  = soma_ext[8];                          // Carry
               out do bit 8
            ov = (~a[7] & ~b[7] & s[7]) | (a[7] & b[7] & ~s[7]);
                // Overflow
        end else begin
            c = 0;
            ov = 0;
        end

        z = (s == 8'b0);         // Zero
        n = s[7];                // Negativo (bit mais
           significativo)
    end

endmodule
```

- **Entradas:**
  - `a [7:0]`: Primeiro operando de 8 bits
  - `b [7:0]`: Segundo operando de 8 bits
  - `x`: Sinal de controle (bit mais significativo da seleção de operação)
  - `y`: Sinal de controle (bit menos significativo da seleção de operação)
- **Saídas:**
  - `s [7:0]`: Resultado da operação (8 bits)
  - `c`: Flag de carry out
  - `ov`: Flag de overflow
  - `z`: Flag de zero

  - `n`: Flag de negativo (bit de sinal)
- **Flags de Status:**
  - **Carry (`c`):**
    - Ativo apenas para operação de soma (`00`)
    - Detectado usando um bit extra: `soma_ext[8]`
  - **Overflow (`ov`):**
    - Calculado apenas para operação de soma
    - Detecta quando o resultado excede a capacidade de representação em complemento de 2
    - Fórmula: `(ã[7] & b̃[7] & s[7]) | (a[7] & b[7] & s̃[7])`
  - **Zero (`z`):**
    - Ativo quando o resultado é zero (todos os bits em 0)
    - Calculado para todas as operações
  - **Negativo (`n`):**
    - Indica se o bit mais significativo do resultado está ativo
    - Útil para operações aritméticas em representação de complemento de 2

4. Testes da ULA

Listing 2. Módulo para o test bench da ULA de 8 Bits

```
module tb_ula_8b;

    // Entradas
    reg [7:0] a, b;
    reg x, y;

    // Saídas
    wire [7:0] s;
    wire c, ov, z, n;

    // Instancia a ULA
    ula_8b dut (
        .a(a), .b(b), .x(x), .y(y),
        .s(s), .c(c), .ov(ov), .z(z), .n(n)
    );

    // Procedimento de teste
    initial begin
        $display("x y |     a     b   |    s    | c ov z n");
        $display("----+-------------+---------+----------");

        // Teste de soma sem overflow
        x = 0; y = 0; a = 8'd10; b = 8'd20; #10;
        $display("%b %b | %h %h | %h | %b  %b  %b %b", x, y, a,
           b, s, c, ov, z, n);

        // Teste de soma com carry e overflow (127 + 1)
```

```
27          x = 0; y = 0; a = 8'd127; b = 8'd1; #10;
28          $display("%b %b | %h %h | %h | %b  %b  %b %b", x, y, a,
               b, s, c, ov, z, n);
29
30          // Teste de soma negativa com overflow (-128 + -128)
31          x = 0; y = 0; a = 8'b10000000; b = 8'b10000000; #10;
32          $display("%b %b | %b %b | %b | %b  %b  %b %b", x, y, a,
               b, s, c, ov, z, n);
33
34          // Teste de soma com resultado zero (127 + -127)
35          x = 0; y = 0; a = 8'd127; b = -8'd127; #10;
36          $display("%b %b | %h %h | %h | %b  %b  %b %b", x, y, a,
               b, s, c, ov, z, n);
37
38          // Teste de AND
39          x = 0; y = 1; a = 8'b11001100; b = 8'b10101010; #10;
40          $display("%b %b | %b %b | %b | %b  %b  %b %b", x, y, a,
               b, s, c, ov, z, n);
41
42          // Teste de OR
43          x = 1; y = 0; a = 8'b11001100; b = 8'b10101010; #10;
44          $display("%b %b | %b %b | %b | %b  %b  %b %b", x, y, a,
               b, s, c, ov, z, n);
45
46          // Teste de NOT
47          x = 1; y = 1; a = 8'b11110000; b = 8'b00000000; #10; //
               b não importa no NOT
48          $display("%b %b | %b %b | %b | %b  %b  %b %b", x, y, a,
               b, s, c, ov, z, n);
49
50          $finish;
51      end
52
53  endmodule
```

O testbench `tb_ula_8b` foi desenvolvido para verificar todas as funcionalidades da Unidade Lógica Aritmética, incluindo operações aritméticas e lógicas, além das flags de status. Os testes foram projetados para cobrir casos normais e casos extremos.

4.1. Casos de Teste Implementados

A Tabela 2 resume os testes realizados para verificação da ULA:

Tabela 2. Descrição dos testes realizados

Teste	Operação	Objetivo
1	Soma simples	Verificar operação básica (10 + 20 = 30) e flags (nenhum erro esperado)
2	Limite positivo	Testar overflow com 127 + 1 (deve ativar flag ov e n)
3	Limite negativo	Verificar (-128) + (-128) (deve ativar c e ov)
4	Resultado zero	Confirmar 127 + (-127) = 0 (deve ativar z)
5	AND lógico	Testar 11001100 & 10101010 = 10001000 e flags
6	OR lógico	Verificar 11001100 — 10101010 = 11101110
7	NOT lógico	Confirmar inversão ∼11110000 = 00001111

4.2. Detalhamento dos Testes

4.2.1. Teste 1: Soma básica

```
x = 0; y = 0; a = 8'd10; b = 8'd20;
```

Verificação inicial da operação aritmética com valores positivos pequenos. Resultado esperado:

- `s = 30 (0x1E)`
- Todas flags desativadas (`c, ov, z, n = 0`)

4.2.2. Teste 2: Overflow positivo

```
x = 0; y = 0; a = 8'd127; b = 8'd1;
```

Testa o limite superior da representação em complemento de 2:

- Resultado matemático: 128
- Resultado esperado: `-128 (overflow)`
- Flags ativas: `ov=1, n=1`

4.2.3. Teste 3: Overflow negativo

```
a = 8'b10000000; b = 8'b10000000; // -128 + -128
```

Verifica o limite inferior da representação:

- Resultado matemático: -256
- Resultado esperado: `0 (com carry e overflow)`
- Flags ativas: `c=1, ov=1`

4.2.4. Teste 4: Resultado zero

```
a = 8'd127; b = -8'd127;
```

Importante para verificar a flag `z`:

- Resultado esperado: `s = 0`
- Flags ativas: `z=1`

4.2.5. Testes 5-7: Operações lógicas

- **AND**: Verifica máscara de bits (`11001100 & 10101010 = 10001000`)
- **OR**: Testa combinação de bits (`11001100 | 10101010 = 11101110`)
- **NOT**: Confirma inversão bit-a-bit (∼`11110000 = 00001111`)

Para todos os testes lógicos, espera-se:

- `c=0, ov=0`
- `z` e `n` conforme o resultado

4.3. Verifiação no ModelSim

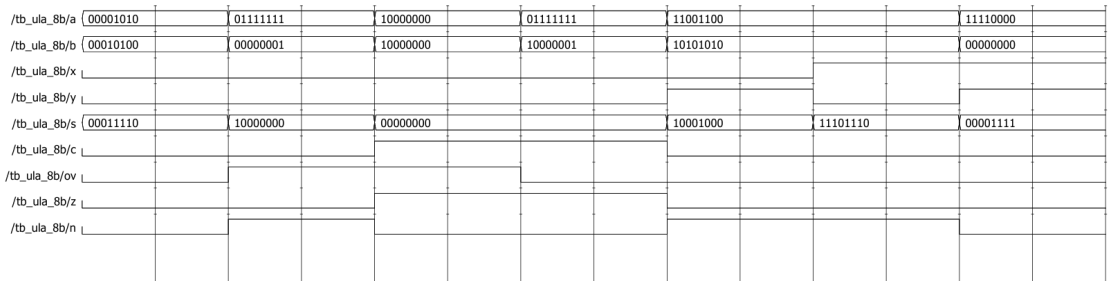

Figura 1. Verificação de cada um dos testes no ModelSim

5. Conclusão

O desenvolvimento e a implementação da ULA (Unidade Lógica Aritmética) de 8 bits em Verilog demonstraram um funcionamento robusto e eficiente, cumprindo com êxito os requisitos básicos de uma unidade de processamento. A ULA foi capaz de executar as quatro operações fundamentais programadas —soma aritmética, operações lógicas AND e OR, e inversão (NOT)— com precisão e confiabilidade, conforme evidenciado pelos testes realizados.

Um dos aspectos mais relevantes foi a correta implementação do sistema de flags, que garantiu a detecção de condições importantes durante as operações, como resultados negativos (flag `n`), resultados zero (flag `z`), overflow aritmético (flag `ov`) e carry em operações de soma (flag `c`). Essas flags são essenciais para a integração da ULA em sistemas mais complexos, como CPUs, onde decisões condicionais dependem desses indicadores.

Os testes realizados abrangeram desde operações simples, como a soma de dois números pequenos (10 + 20), até casos, como o overflow positivo (127 + 1) e negativo ($-128 + -128$), bem como a verificação de resultados zero (127 + (-127)). Em todos os cenários, a ULA respondeu conforme o esperado, validando a lógica de controle e a estrutura do projeto.

Referências

[Souza] Souza, P. Unidade lógico-aritmética usando a abordagem comportamental. https://www.youtube.com/watch?v=Ynymty6-5dM.