



Prática N<sup>o</sup>6 - SEL0606

Alunos (Bancada 5):  
Vitor Alexandre Garcia Vaz - 14611432  
Gabriel Dezejácomo Maruschi - 14571525

# Sumário

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Introdução</b>  | <b>3</b> |
| 1.1      | <i>Objetivos</i>   | 3        |
| 1.2      | <i>Unidade Lógica-Aritmética</i>                           | 3        |
| <b>2</b> | <b>Materiais e métodos</b>                                 | <b>4</b> |
| 2.1      | <i>Interface</i>   | 4        |
| 2.2      | <i>Unidade Lógica Aritmética</i>                           | 5        |
| <b>3</b> | <b>Conclusão</b>   | <b>7</b> |
| 3.1      | <i>Círculo RTL</i>   | 7        |
| 3.2      | <i>Número de células lógicas</i>                           | 8        |
| 3.3      | <i>Funcionamento círculo para operação soma</i>            | 8        |
| 3.4      | <i>Funcionamento do círculo para operação de subtração</i> | 9        |
| 3.5      | <i>Funcionamento círculo para operação AND</i>             | 10       |
| 3.6      | <i>Funcionamento círculo para operação OR</i>              | 11       |
| 3.7      | <i>Funcionamento do círculo para operação SLT</i>          | 13       |

## Lista de Imagens

|    |   |    |
|----|---|----|
| 1  | Esquemático de Unidade Lógico-Aritmética                                | 3  |
| 2  | Código da interface   | 4  |
| 3  | Kit DE10-LITE   | 5  |
| 4  | Tabela de operações da ULA  | 5  |
| 5  | Código da ULA   | 6  |
| 6  | Visualização geral do círculo formado                                   | 7  |
| 7  | Círculo interno da Unidade Lógico-Aritmética                            | 7  |
| 8  | Resumo de funcionamento   | 8  |
| 9  | Acendimento do display para soma de $A = 0011_2$ e $B = 0001_2$         | 8  |
| 10 | Acendimento dos display para soma de $A = 0110_2$ e $B = 0001_2$        | 9  |
| 11 | Acendimento do display para subtração de $A = 0001_2$ por $B = 0001_2$  | 9  |
| 12 | Acendimento dos display para subtração de $A = 0001_2$ por $B = 0101_2$ | 10 |
| 13 | Acendimento dos display para subtração de $A = 0111_2$ por $B = 0101_2$ | 10 |
| 14 | Acendimento do display para $A = 0101_2$ AND $B = 0001_2$               | 11 |
| 15 | Acendimento dos display para $A = 0101_2$ AND $B = 0001_2$              | 11 |
| 16 | Acendimento do display para $A = 0010_2$ OR $B = 0001_2$                | 12 |
| 17 | Acendimento dos display para $A = 0111_2$ OR $B = 0101_2$               | 12 |
| 18 | Tabela verdade da operação SLT para as entradas A e B                   | 13 |
| 19 | Acendimento do display para a comparação $A = 0011_2 < B = 0011_2$      | 13 |
| 20 | Acendimento do display para a comparação $A = 0011_2 < B = 0111_2$      | 14 |
| 21 | Acendimento do display para a comparação $A = 0111_2 < B = 0011_2$      | 14 |

# 1 Introdução

## 1.1 Objetivos

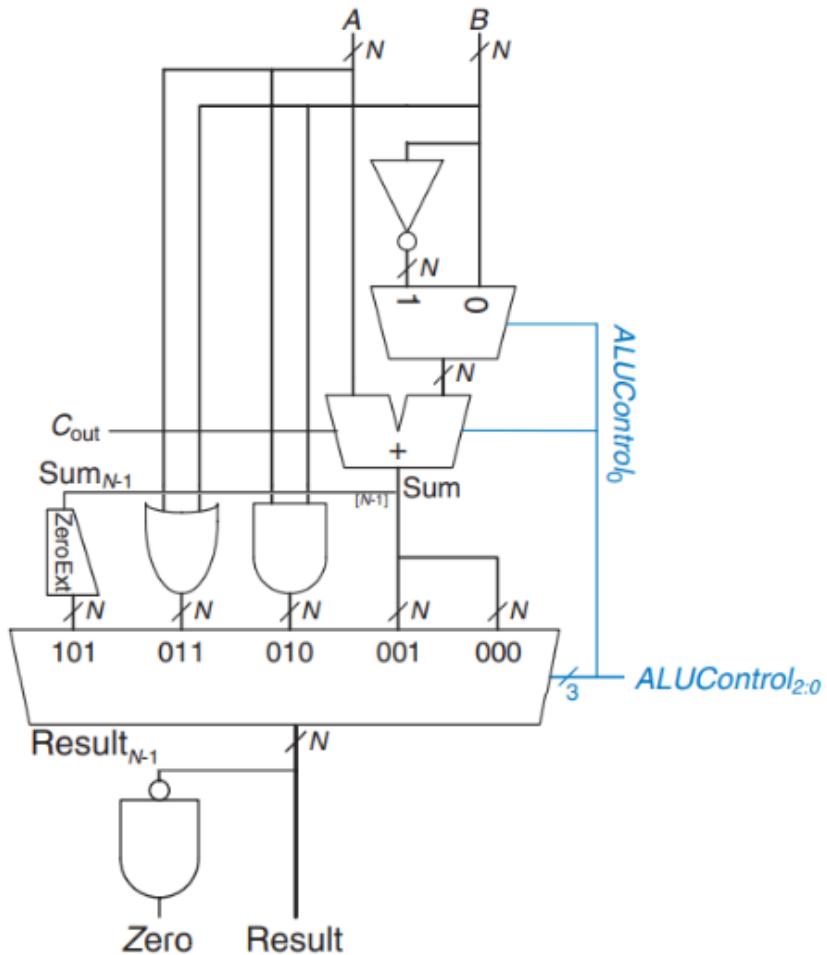
Nesta prática do Laboratório de Sistemas Digitais, tivemos como objetivo implementar uma Unidade Lógico-Aritmética (ULA) utilizando a linguagem VHDL, em conjunto com a ferramenta quartus, para entradas de 4 bits.

Além disso, aplicando o código na FPGA pertencente ao ki DE10-Lite, testamos o funcionamento do circuito visualizando a resposta das entradas na forma de acendimento dos segmentos do display de sete segmentos, de acordo com as entradas indicadas pelas chaves e pelo primeiro botão do dispositivo.

## 1.2 Unidade Lógica-Aritmética

A ULA deve possibilitar todas as operações que o computador deve realizar durante o seu funcionamento. No caso da aplicação desse experimento, abordaremos as operações de adição, subtração, AND, OR e SLT (verificação se a primeira entrada é menor que a segunda). Para isso, usaremos as estruturas lógicas necessárias (somadores, multiplexadores e decodificadores), levando em conta que a ULA é um circuito combinacional e o modelo apresentado nas intruções da prática (correspondente ao apresentado na fig.1).

Figure 1: Esquemático de Unidade Lógico-Aritmética



Fonte: [Site da internet](#)

## 2 Materiais e métodos

O código utilizado para gerar a Unidade Lógica Aritmética foi escrito em VHDL e compilado no Quartus. Os códigos dos módulos funcionais da ULA foram escritos de maneira modular e interligados por meio de uma interface. Além disso, com o uso da interface, direcionamos os sinais de entrada e saída para o DE10-LITE. A ULA é manipulada pelas entradas:

- A (4 bits),
- B (4 bits),
- ALUControl (3 bits).

A e B são os valores manipulados nas operações lógicas e aritméticas e o ALUControl controla os multiplexadores. A partir disto, são gerados:

- Result (4 bits),
- Zero (1 bit).

Sendo Result a saída da operação entre A e B e Zero uma flag que liga apenas quando o resultado é nulo em todo o barramento.

### 2.1 Interface

A interface, codificada como é mostrado abaixo, direciona as entradas e saídas para as portas e displays do kit DE10-LITE:

Figure 2: Código da interface

```
1  --Copyright:
2  --Date: 23/10/24
3  --Version: 1.0
4  --Owners: Gabriel D. Maruschi
5  --          Vitor Alexandre Garcia Vaz
6
7
8
9  entity DE10_LITE_ALU is
10  port
11  (
12      HEX0    : out bit_vector(7 downto 0);
13      SW      : in  bit_vector(9 downto 0);
14      KEY     : in  bit_vector(1 downto 0);
15      LEDR    : out bit_vector(9 downto 0)
16  );
17
18 end DE10_LITE_ALU;
19
20
21
22 architecture top of DE10_LITE_ALU is
23
24     signal ALUREsult: bit_vector(3 downto 0);
25
26 begin
27
28     -- Component Instantiation Statement
29
30     ALU0: work.alu
31     port map (
32         A => SW(3 downto 0), B => SW(7 downto 4), ALUControl => SW(9 downto 8) & KEY(0),
33         Result => ALUREsult, Zero => LEDR(0)
34     );
35
36
37     display0: work.hex27seg
38     port map (
39         hexa => ALUREsult,
40         segments => HEX0(7 downto 0)
41     );
42
43 end top;
```

Fonte: os autores

### Instância da ULA

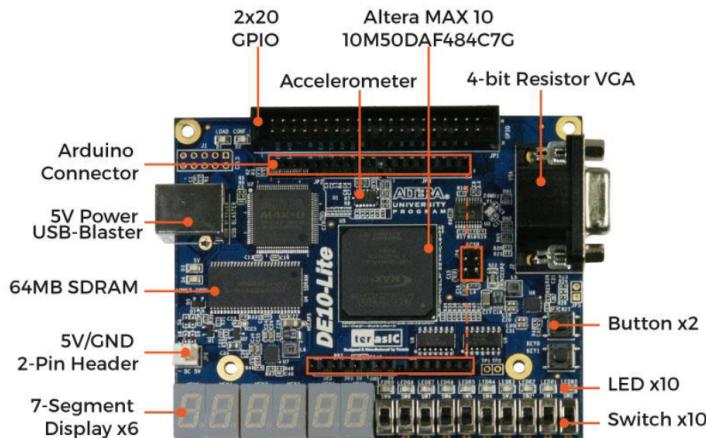
Observa-se do código do módulo da ULA, que as entradas **A** e **B** são provenientes das chaves de 0 a 3 e de 4 a 7, respectivamente, enquanto a **ALUControl** é selecionada pelas chaves 8, 9 e o botão 0, sendo este último o bit menos significativo.

Além disso, **Result** é direcionado para o sinal auxiliar **ALUResult** e a flag **Zero** para o led vermelho 0.

### Instância do display

A fim de mostrar o resultado, chamamos a instância do display de binário para 7 segmentos, para o qual direcionamos como entrada o resultado **ALUResult**, e do qual a saída foi mapeada no display hexadecimal 0 (HEX0).

Figure 3: Kit DE10-LITE



Fonte: DE10-Lite User Manual

## 2.2 Unidade Lógica Aritmética

A estrutura do código da ULA desenvolvida visa a utilização de multiplexadores para selecionar para a saída um dos resultados gerados entre A e B. É justamente **ALUControl** que faz o papel de seletor nos MUX, de acordo com a tabela:

Figure 4: Tabela de operações da ULA

| $ALUControl_{2:0}$ | Função             |
|--------------------|--------------------|
| 000                | $A + B$            |
| 001                | $A - B$            |
| 010                | $A \text{ and } B$ |
| 011                | $A \text{ or } B$  |
| 100                | -                  |
| 101                | SLT                |
| 110                | -                  |
| 111                | -                  |

Fonte: instrução da prática

Figure 5: Código da ULA

```

1  --Copyright:
2  --Date: 23/10/24
3  --Version: 1.3
4  --Owners: Gabriel D. Maruschi
5  --          Vitor Alexandre Garcia Vaz
6
7  entity alu is
8      port
9      (
10         A, B      : in bit_vector(3 downto 0); -- Signals A e B do MODO de entrada (in) e do TIPO bit_vector
11         ALUControl : in bit_vector(2 downto 0); -- Signal ALUControl do MODO de entrada (in) e do TIPO bit_vector
12
13         -- Saída(s)
14         Result    : out bit_vector(3 downto 0); -- Signal Result do MODO de saída (out) e do TIPO bit_vector de tamanho 3
15         Zero      : out bit; -- Signal Zero de MODO de saída (out) e do tipo bit
16         -- A última declaração de porta não tem ponto e vírgula (;)
17     );
18 end alu;
19
20
21 architecture estrutural of alu is -- ARQUITETURA 'estrutural' associada à ENTIDADE 'alu'
22
23     signal fb      : bit_vector(3 downto 0); -- recebe b ou b negado (dependendo do ALUControl(0) )
24
25     signal adderResult : bit_vector(3 downto 0); -- recebe resultado do adder (pode ser uma subtração ou uma soma)
26     signal andresult  : bit_vector(3 downto 0); -- recebe resultado da operação lógica and
27     signal orResult   : bit_vector(3 downto 0); -- recebe resultado da operação lógica or
28     signal sltResult  : bit_vector(3 downto 0); -- recebe resultado da comparação slt
29     signal tempResult : bit_vector(3 downto 0); -- buffer de resultado final da ULA
30
31     -- Início da declaração da ARQUITETURA
32 begin
33
34     -- Operações aritméticas
35
36     mux0: work.mux21 port map (I0 => B, I1 => (not B), S => ALUControl(0), Z => fb);
37
38     add: work.adder port map(A => A, B => fb, Cin => ALUControl(0), RESULT => adderResult);
39
40     -- Operação and
41     andresult <= A and B;
42
43     -- Operação orr
44     orResult <= A or B;
45
46     -- Operação slt
47     sltResult <= "000" & adderResult(3);
48
49     -- Mux 5 para 1 para selecionamento da operação que definirá a saída
50     with ALUControl select
51         tempResult <= adderResult when "000"; --soma
52             adderResult when "001"; --subtração
53             andresult when "010"; --and
54             orResult when "011"; --or
55             sltResult when "101"; --A < B
56             "000" when others;
57
58     -- Flag zero (igual a 1 para resultado nulo)
59     Zero <= not ((tempResult(0) or tempResult(1)) or (tempResult(2) or tempResult(3)));
60
61     Result <= tempResult; -- Resultado final
62
63 end estrutural;

```

Fonte: os autores

### Instância do MUX21

A instância **mux0** de um mux21 utiliza do bit menos significativo do **ALUControl** para selecionar entre o sinal **B** e a negação **not B**. Isto será útil na operação de soma e subtração, já que ao inverter os bits de **B** e somar 1 bit, é possível utilizar o módulo do somador para realizar também subtrações.

### Instância do somador

A instância **add** do somador utiliza como entradas o sinal de **A** e a saída do mux citado anteriormente. Além disso, utiliza o bit menos significativo do **ALUControl** como carry in (Cin). Assim, quando o código da operação da subtração é selecionado, o mux **mux0** roteia o sinal negado de **B** o carry in (valendo 1) é somado.

### AND e OR

Os sinais **andresult** e **orResult** são provenientes das operações de and e or entre os sinais **A** e **B** diretamente neste módulo do código.

### SLT

A operação SLT verifica se **A** é estritamente menor que **B**. Para isso, pega o bit mais significativo do resultado de adder. Observando a tabela, nota-se que quando a operação de SLT for selecionada, adder estará realizando uma subtração entre **A** e **B**, de maneira que, quando **A** for menor que **B**, o resultado será negativo, que, em complemento de 2, significa ter o bit mais significativo igual à 1.

### MUX 5 para 1 e flag Zero

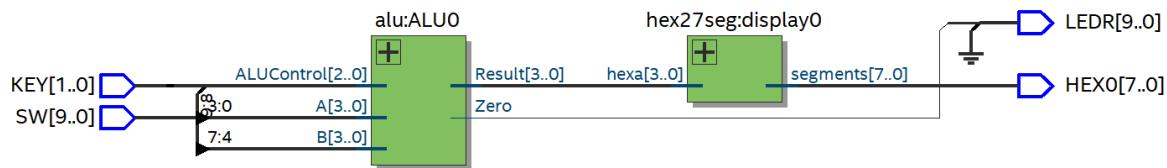
Por fim, implementamos no próprio módulo da ULA um MUX51 que roteia um dos resultados citados acima para um sinal temporário **tempResult**. O sinal seletor, como já foi citado, é **ALUControl**. Para entradas inesperadas de **ALUControl**, convencionamos a saída 0 em todos os bits do barramento. Obtido **tempResult**, verificamos se todos os bits da saída são zero a fim de determinar a flag **Zero** e direcionamos **tempResult** para o output **Result**.

### 3 Conclusão

#### 3.1 Circuito RTL

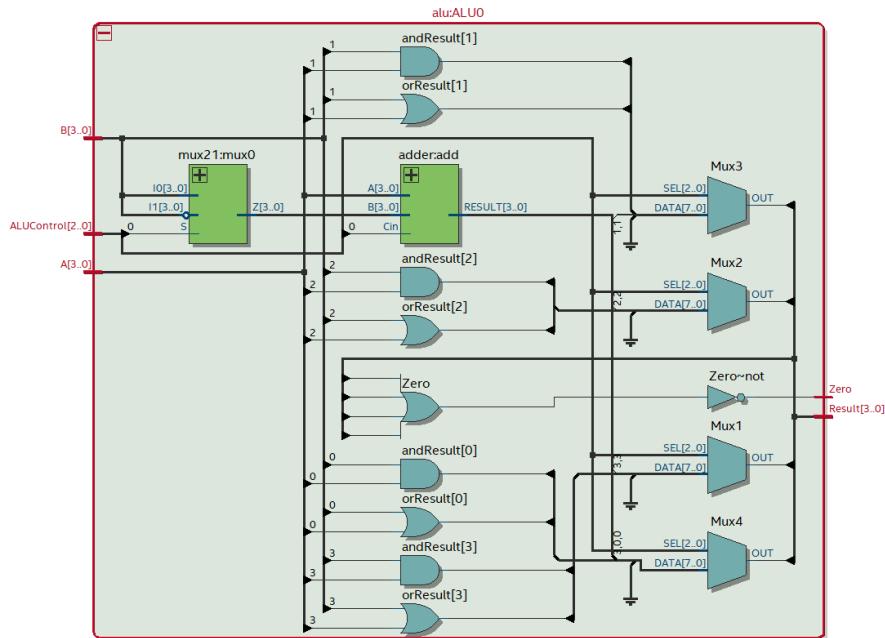
A partir do código em VHDL feito em laboratório, e usando a ferramenta de síntese disponibilizada pelo quartus, os seguintes circuitos foram obtidos:

Figure 6: Visualização geral do circuito formado



Fonte: os autores

Figure 7: Circuito interno da Unidade Lógico-Aritmética



Fonte: os autores

### 3.2 Número de células lógicas

Ademais, por meio do resumo do funcionamento e constituição do circuito descrito em VHDL no software (fig8), chegamos à conclusão de que o circuito sintetizado apresentou um uso de 36 células lógicas e 30 pinos do dispositivo reconfigurável (FPGA).

Figure 8: Resumo de funcionamento

| Table of Contents                |  | Flow Summary  |
|----------------------------------|--|---|
| Flow Summary                     |  | <input type="text"/> <<Filter>>                                   |
| Flow Settings                    |  | Flow Status Successful - Wed Oct 23 11:07:59 2024                 |
| Flow Non-Default Global Settings |  | Quartus Prime Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition |
| Flow Elapsed Time                |  | Revision Name DE10_LITE_ALU                                       |
| Flow OS Summary                  |  | Top-level Entity Name DE10_LITE_ALU                               |
| Flow Log                         |  | Family MAX 10   |
| > Analysis & Synthesis           |  | Device 10M50DAF484C7G   |
| > Fitter                         |  | Timing Models Final   |
| Flow Messages                    |  | Total logic elements 36 / 49,760 (< 1 %)                          |
| Flow Suppressed Messages         |  | Total registers 0   |
| > Assembler                      |  | Total pins 30 / 360 (8 %)   |
| > Timing Analyzer                |  | Total virtual pins 0  |
|                                  |  | Total memory bits 0 / 1,677,312 (0 %)                             |
|                                  |  | Embedded Multiplier 9-bit elements 0 / 288 (0 %)                  |
|                                  |  | Total PLLs 0 / 4 (0 %)  |
|                                  |  | UFM blocks 0 / 1 (0 %)  |
|                                  |  | ADC blocks 0 / 2 (0 %)  |

Fonte: os autores

### 3.3 Funcionamento circuito para operação soma

A respeito da operação de soma, o circuito funcionou da forma esperada, já que, ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  correspondentemente aos valores binários  $0011_2$  e  $0001_2$  visualizamos o acendimento do display correspondentemente ao valor  $4_{10}$ , ou seja,  $0100_2$  conforme à fig.9. Portanto, a saída do circuito (visualizada por meio do acendimento do display) foi igual à soma dos valores de entrada recebidos via ligamento das chaves.

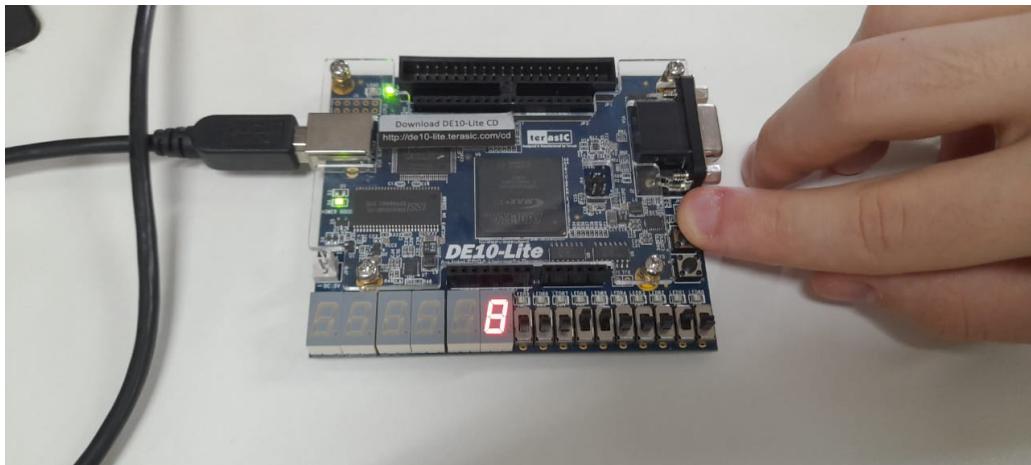
Figure 9: Acendimento do display para soma de  $A = 0011_2$  e  $B = 0001_2$



Fonte: os autores

Ademais, o mesmo comportamento foi observado ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  nas posições correspondentes aos valores binários  $0110_2$  e  $0010_2$ , respectivamente, o que resultou na visualização do valor  $1000_2$ , ou seja,  $8_{10}$  por meio do acendimento do display, conforme à fig.10.

Figure 10: Acendimento dos display para soma de  $A = 0110_2$  e  $B = 0001_2$

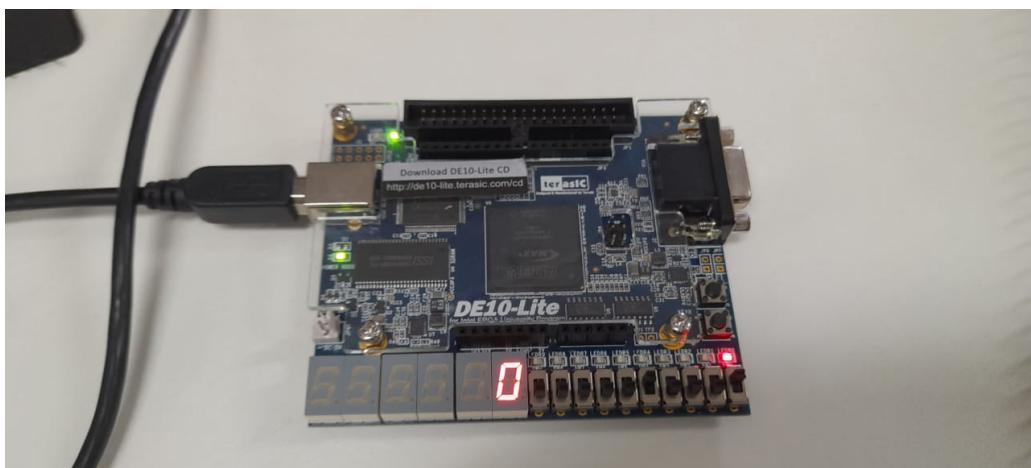


Fonte: os autores

### 3.4 Funcionamento do circuito para operação de subtração

Para operação de subtração, o circuito funcionou da forma esperada, já que, ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  ambamente correspondentes ao valor binário  $0001_2$ , visualizamos o acendimento do display correspondente ao valor  $0_{10}$ , ou seja  $0000_2$ , conforme à fig.9. Portanto a saída do circuito (visualizada por meio do acendimento do display) foi igual à subtração dos valores de entrada recebidos via ligamento das chaves, valendo ressaltar que o led LEDR(0) ascendeu para esta operação, indicando a efetividade do funcionamento da flag de resultado zero em nossa implementação.

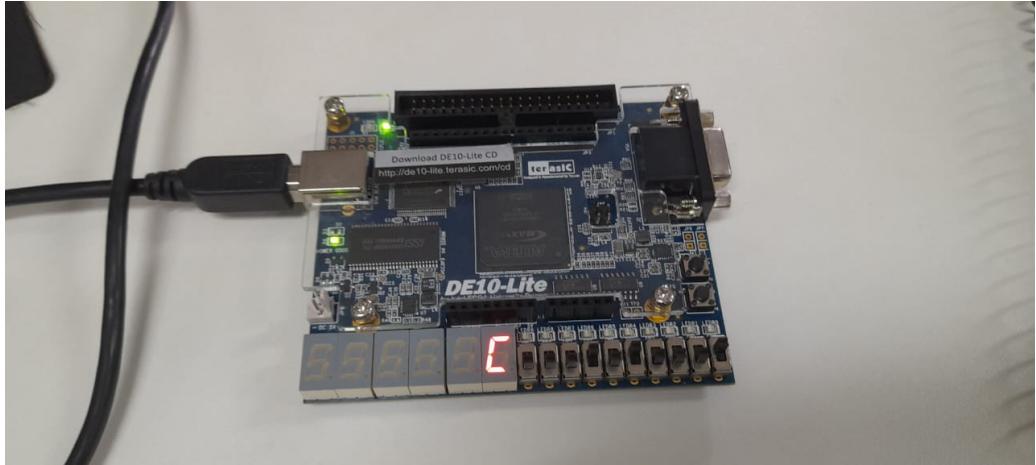
Figure 11: Acendimento do display para subtração de  $A = 0001_2$  por  $B = 0001_2$



Fonte: os autores

O mesmo comportamento foi observado ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  nas posições correspondentes aos valores binários  $0001_2$  e  $0101_2$ , respectivamente, o que resultou na visualização do valor  $1100_2$ , ou seja,  $-4_{10}$  por meio do acendimento do display, levando em conta que adotamos a representação em complemento de 2 para a subtração, já que estamos considerando o bit mais significativo do resultado como indicador de seu sinal (1 para negativo e 0 para positivo), conforme à fig.12.

Figure 12: Acendimento dos display para subtração de  $A = 0001_2$  por  $B = 0101_2$



Fonte: os autores

Por fim, um comportamento análogo foi observado ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  nas posições correspondentes aos valores binários  $0111_2$  e  $0101_2$ , respectivamente, o que resultou na visualização do valor  $0010_2$ , ou seja,  $2_{10}$  por meio do acendimento do display, resultado que está em conformidade com a representação por complemento de 2 e com a subtração  $7 - 5 = 2$ , conforme à fig.13.

Figure 13: Acendimento dos display para subtração de  $A = 0111_2$  por  $B = 0101_2$

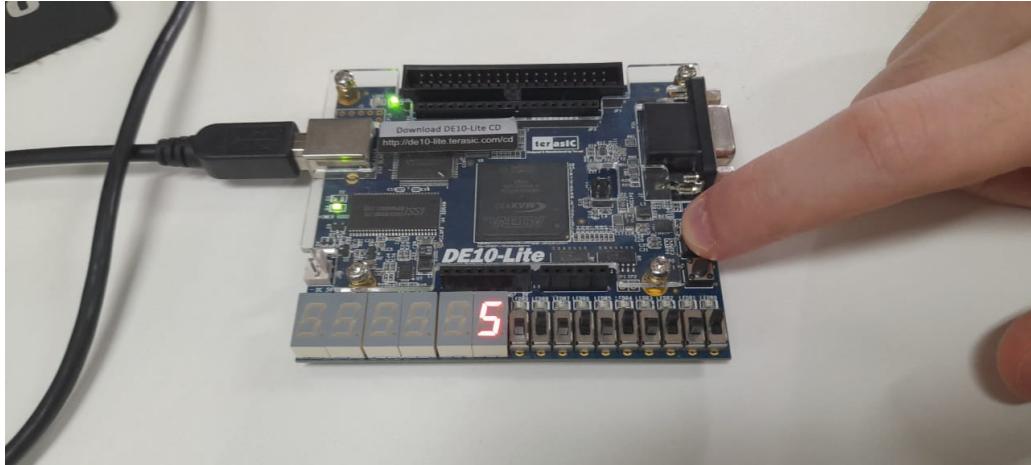


Fonte: os autores

### 3.5 Funcionamento circuito para operação AND

Para operação AND, o circuito funcionou da forma esperada, já que, ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  ambivamente correspondentes ao valor binário  $0101_2$ , visualizamos o acendimento do display correspondente ao valor  $5_{10}$ , ou seja,  $0101_2$  conforme à fig.14. Portanto, a saída do circuito (visualizada por meio do acendimento do display) foi igual a operação AND bit a bit (do menos ao mais significativo) das duas entradas.

Figure 14: Acendimento do display para  $A = 0101_2$  AND  $B = 0001_2$



Fonte: os autores

Ademais, um comportamento semelhante foi observado ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  nas posições correspondentes aos valores binários  $0101_2$  e  $0001_2$ , respectivamente, o que resultou na visualização do valor  $0001_2$ , ou seja,  $1_{10}$  por meio do acendimento do display, conforme à fig.15. Nesse sentido, tal resultado demonstrou-se correto, pois a operação AND , à cerca de cada bit, só resultou em 1 quando ambos os bits (  $A(i)$  e  $B(i)$  ) foram também iguais a 1.

Figure 15: Acendimento dos display para  $A = 0101_2$  AND  $B = 0001_2$

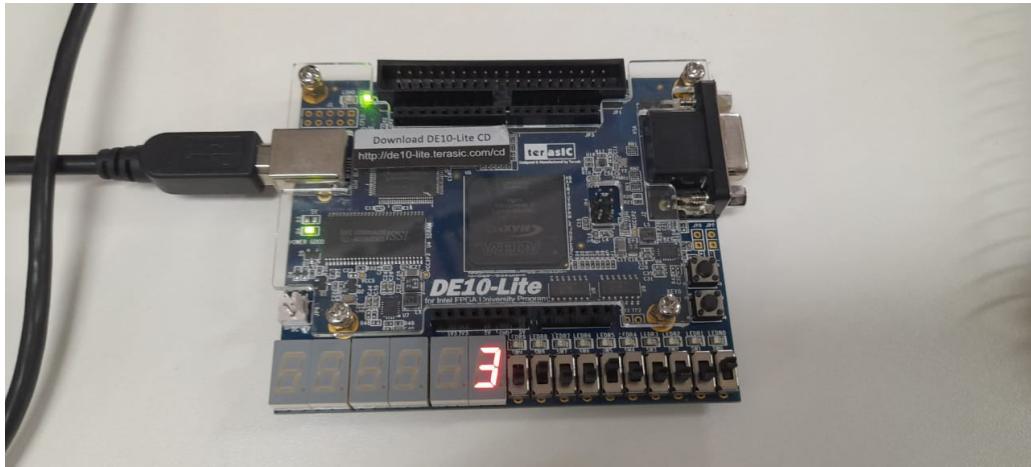


Fonte: os autores

### 3.6 Funcionamento circuito para operação OR

Já para operação OR, o circuito funcionou da forma esperada, já que, ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  foram ativadas correspondentemente aos valores binários  $0001_2$  e  $0010_2$ , respectivamente, e visualizamos o acendimento do display correspondentemente ao valor  $3_{10}$ , ou seja,  $0011_2$  conforme à fig.16. Portanto, a saída do circuito (visualizada por meio do acendimento do display) foi igual a operação OR bit a bit (do menos ao mais significativo) das duas entradas.

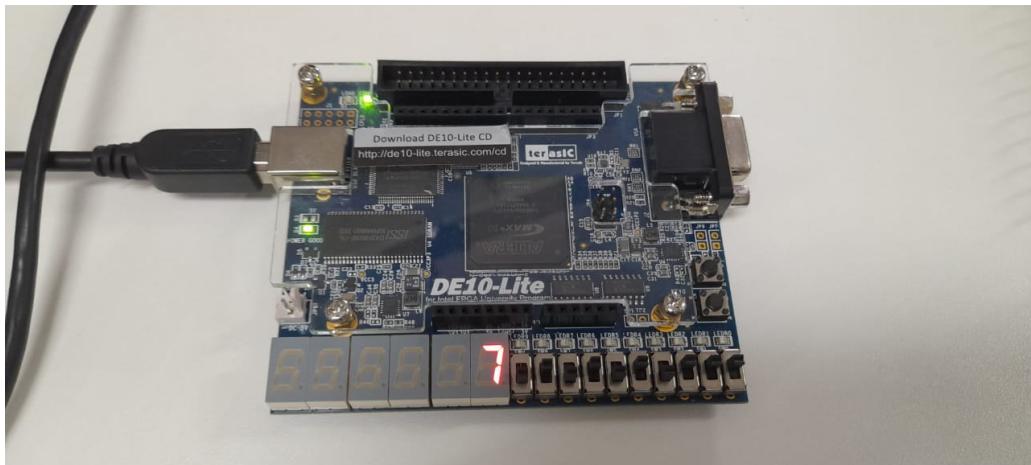
Figure 16: Acendimento do display para  $A = 0010_2$  OR  $B = 0001_2$



Fonte: os autores

Um comportamento semelhante foi observado ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  nas posições correspondentes aos valores binários  $0111_2$  e  $0101_2$ , respectivamente, o que resultou na visualização do valor  $0111_2$ , ou seja,  $7_{10}$  por meio do acendimento do display, conforme à fig.17. Nesse sentido, tal resultado demonstrou-se correto, já que a operação OR, atuante em cada bit, somente resultou em nível lógico 1 quando pelo menos um dos bits (  $A(i)$  ou  $B(i)$  ) foi equivalente ao nível lógico 1.

Figure 17: Acendimento dos display para  $A = 0111_2$  OR  $B = 0101_2$



Fonte: os autores

### 3.7 Funcionamento do circuito para operação SLT

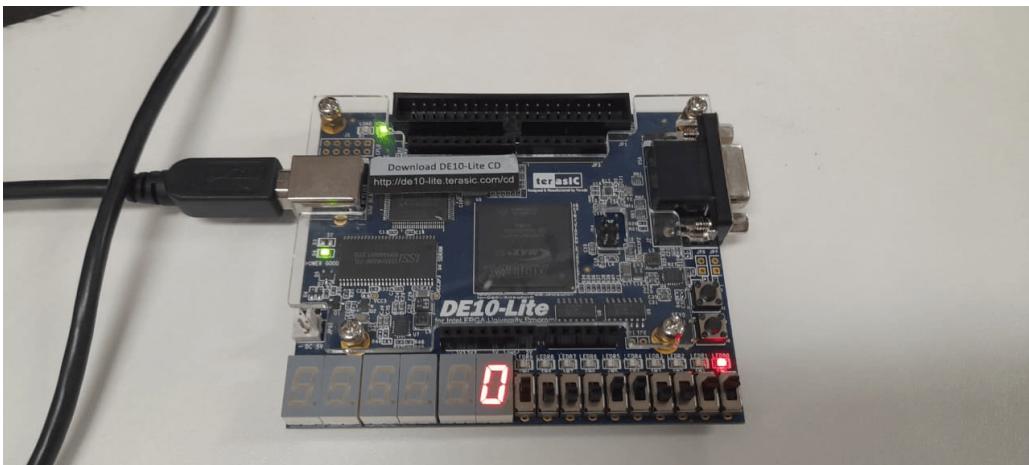
Figure 18: Tabela verdade da operação SLT para as entradas A e B

| SLT        |         |
|------------|---------|
| $A \geq B$ | $A < B$ |
| 0000       | 0001    |

Fonte: instruções da prática 6

Para operação de verificação da minoridade de uma entrada A em relação a outra B, o circuito também funcionou da forma que era prevista, já que, ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  ambas correspondentes ao valor binário  $0011_2$ , visualizamos o acendimento do display correspondente ao valor  $0_{10}$ , ou seja  $0000_2$ , conforme à fig.19. Resultado o qual era esperado, de acordo com a tabela verdade fig.18, pois as entradas foram numericamente iguais. Além disso, vale ressaltar que o led LEDR(0) ascendeu para esta operação, fato o qual mostrou a efetividade do funcionamento da flag de resultado zero em nossa implementação.

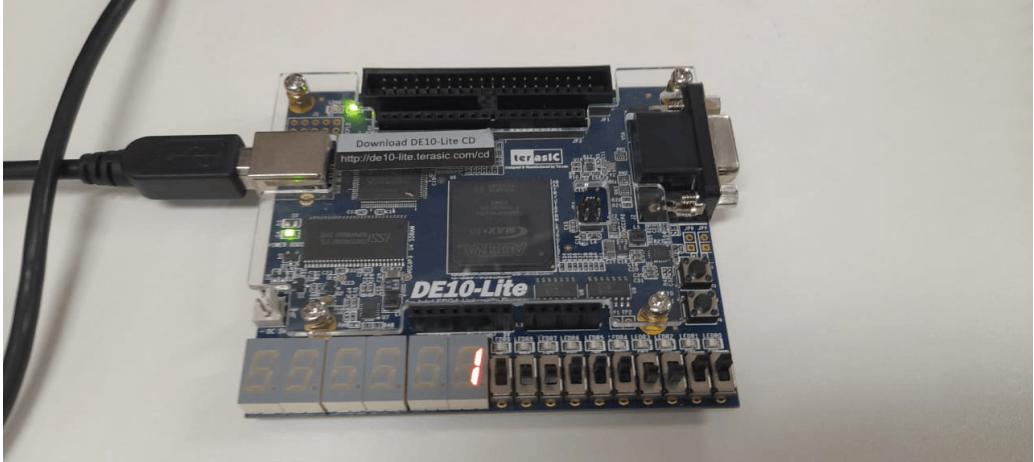
Figure 19: Acendimento do display para a comparação  $A = 0011_2 < B = 0011_2$



Fonte: os autores

O mesmo comportamento foi observado ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  nas posições correspondentes aos valores binários  $0011_2$  e  $0111_2$ , respectivamente, o que resultou na visualização do valor  $0001_2$ , ou seja,  $1_{10}$  por meio do acendimento do display, conforme à fig.20. Nesse sentido, tal resultado mostrou-se coerente ao esperado, levando em conta a tabela em fig.18, já que a primeira entrada (A) foi de fato menor do que a segunda (B).

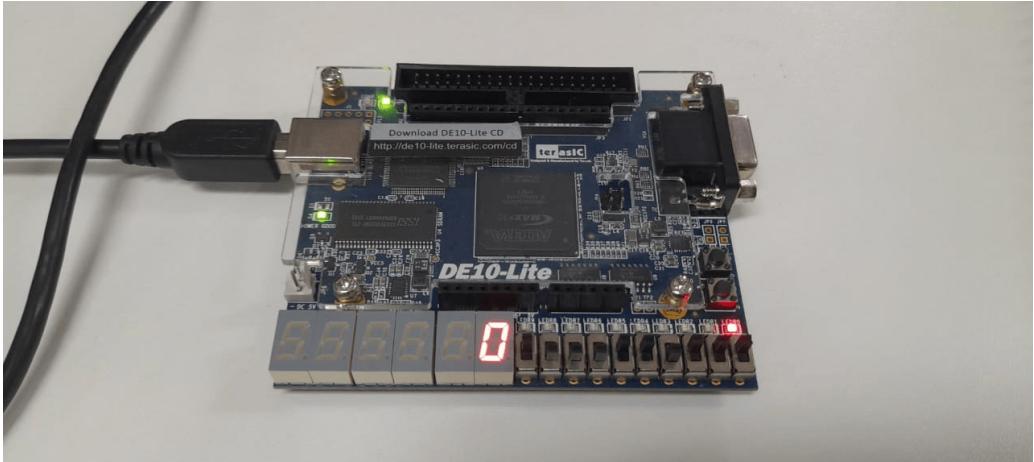
Figure 20: Acendimento do display para a comparação  $A = 0011_2 < B = 0111_2$



Fonte: os autores

Por fim, um comportamento análogo foi observado ao ligarmos as chaves  $Sw(3 : 0)$  e  $Sw(4 : 7)$  nas posições correspondentes aos valores binários  $0111_2$  e  $0011_2$ , respectivamente, o que resultou na visualização do valor  $0000_2$ , ou seja,  $0_{10}$  por meio do acendimento do display, conforme à fig.21. Resultado o qual também era esperado pela dupla, já que a primeira entrada (A) foi na verdade maior que a segunda (B), o que implica que a comparação  $A < B$  realizada através da operação SLT deveria de fato resultar no nível lógico 0, de acordo com a tabela em fig.18.

Figure 21: Acendimento do display para a comparação  $A = 0111_2 < B = 0011_2$



Fonte: os autores