SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DO SUL E SUDESTE DO PARÁ - UNIFESSPA
INSTITUTO DE GEOCIÊNCIAS E ENGENHARIAS - IGE
FACULDADE DE COMPUTAÇÃO E ENG. ELÉTRICA – FACEEL
CURSO ENGENHARIA DE COMPUTAÇÃO

Microeletrônica

T-2018

Prof. José Carlos Da Silva
jcdsilv@hotmail.com
jose-carlos.silva@unifesspa.edu.br
whatsApp: 94-981431852

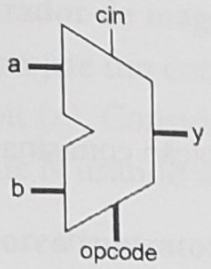
Maio/2022

ATIVIDADE 01
(ULA - 01)

21.5 ALU

ALU (arithmetic-logic unit) foi estudada na Seção 12.14. Seu símbolo é mostrado à esquerda na Figura 21.8, enquanto que as especificações para o presente exemplo são mostradas à direita (emprestadas da Figura 12.17). O projeto dessa ALU, usando VHDL, é apresentado a seguir.

ATIVIDADE 01
(ULA - 01)



Unit	Instruction	Operation	opcode
Logic	Transfer a	y = a	0000
	Complement a	y = NOT a	0001
	Transfer b	y = b	0010
	Complement b	y = NOT b	0011
	AND	y = a AND b	0100
	NAND	y = a NAND b	0101
	OR	y = a OR b	0110
	NOR	y = a NOR b	0111
Arithmetic	Increment a	y = a+1	1000
	Increment b	y = b+1	1001
	Add a and b	y = a+b	1010
	Sub b from a	y = a–b	1011
	Sub a from b	y = –a+b	1100
	Add negative	y = –a–b	1101
	Add with 1	y = a+b+1	1110
	Add with carry	y = a+b+cin	1111

FIGURA 21.8. Símbolo e especificações da ALU projetada na Seção 21.5.

Lembre-se de que ALUs são circuitos combinacionais, portanto podemos usar código concorrente ou código sequencial. O primeiro foi escolhido nesse exemplo (veja a instrução `SELECT` no código a seguir); se o último tivesse sido escolhido, então um `PROCESS` seria necessário (e `CASE` seria a instrução mais adequada).

Observe também que `STD_LOGIC_VECTOR` foi especificado como o tipo dos sinais principais (linhas 7 e 10). Contudo, o pacote original para esse tipo de dado, *std_logic_1164* (linhas 2 e 3), não define operações aritméticas. Por essa razão, o pacote *std_logic_unsigned* foi acrescentado ao código (linha 4), que, como mencionado na Seção 19.3, contém operações aritméticas para `STD_LOGIC_VECTOR`. Poderíamos considerar a utilização de `INTEGER` em vez de `STD_LOGIC_VECTOR`, porque o primeiro suporta operações aritméticas, mas então o problema passaria para as operações lógicas, que não são definidas no pacote original (*standard*) para `INTEGER`.

ATIVIDADE 01
(ULA - 01)

```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE ieee.std_logic_unsigned.all;
5  ------------------------------------------------
6  ENTITY alu IS
7     PORT (a, b: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
8           cin: IN STD_LOGIC;
9           opcode: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
10          y: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
11 END alu;
12 ------------------------------------------------
13 ARCHITECTURE alu OF alu IS
14 BEGIN
15    WITH opcode SELECT
16       --logic part:--------
17       y <= a WHEN "0000",
18       NOT a WHEN "0001",
19       b WHEN "0010",
20       NOT b WHEN "0011",
21       a AND b WHEN "0100",
22       a NAND b WHEN "0101",
23       a OR b WHEN "0110",
24       a NOR b WHEN "0111",
25       --arithmetic part:---
26       a+1 WHEN "1000",
27       b+1 WHEN "1001",
28       a+b WHEN "1010",
29       a-b WHEN "1011",
30       -a+b WHEN "1100",
31       -a-b WHEN "1101",
32       a+b+1 WHEN "1110",
33       a+b+cin WHEN OTHERS;
34 END alu;
35 ------------------------------------------------
```

ATIVIDADE 01
(ULA - 01)

Resultados de simulação são mostrados na Figura 21.9. Os valores *a*="00010001" (=17) e *b*="01010001" (=81) foram adotados para as entradas, e sete opcodes foram testados, para os quais espera-se os resultados listados a seguir (observe a ocorrência dos mesmos na Figura 21.9).

opcode = "0100" → $y = a$ AND b = "00010001" (=17)

opcode = "0110" → $y = a$ OR b = "01010001" (=81)

opcode = "0111" → $y = a$ NOR b = "10101110" (=174, ou – 82 em notação com sinal)

opcode = "1000" → $y = a + 1$ = "00010010" (=18)

opcode = "1010" → $y = a + b$ = "01100010" (=98)

opcode = "1111" → $y = a + b + cin$ = "01100011" (=99)

opcode = "1101" → $y = -a - b$ = "10011110" (= –98, ou 158 em notação sem signal)

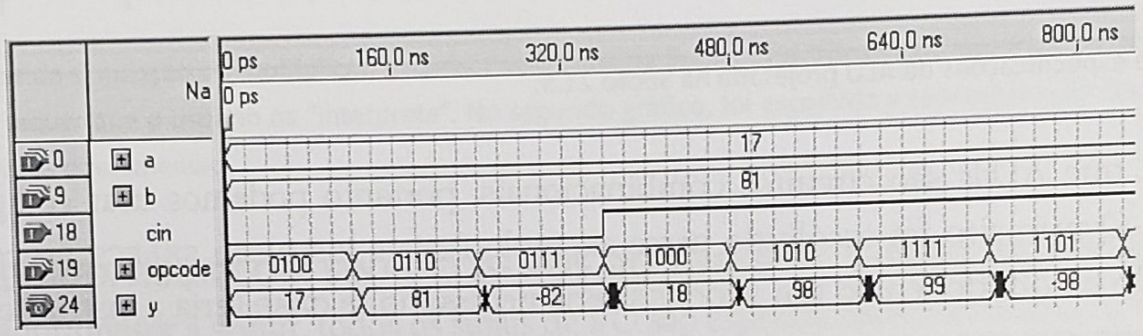


FIGURA 21.9. Resultados de simulação da ALU projetada na Seção 21.5.

ATIVIDADE 01
(ULA - 02)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ULA is
Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
B : in STD_LOGIC_VECTOR (3 downto 0);
C : in STD_LOGIC_VECTOR (1 downto 0);
F : out STD_LOGIC_VECTOR (3 downto 0));
end ULA;

architecture Behavioral of ULA is
Begin
process (A, B, C)
begin
CASE C is
WHEN "00" => F <= A+B;
WHEN "01" => F <= A-B;
WHEN "10" => F <= A xor B;
WHEN others => F <= not A;
END CASE;
end process;
end Behavioral;
```

OBRIGADO