Curso de Graduação em Engenharia Eletrônica - Faculdade Gama - Universidade de Brasília

Prática de Eletrônica Digital 1. Código: FGA0071.

Professor: Henrique Marra Taira Menegaz e-mail: henriquemenegaz@unb.br

---

EXPERIMENTO DE VHDL 3

1 REGRAS DE APRESENTAÇÃO

Os grupos deverão apresentar o experimento de forma presencial, **na sala de aul**a, **durante o horário de aula**, **até a aula seguinte** à designada a este experimento. A apresentação consiste em mostrar ao professor **o projeto implementado na Basys 3 e explicar os códigos** escritos.

2 NOTA

O experimento receberá nota entre 0 e 10 pontos.

3 PROJETO

Projete uma ULA programável de acordo com o esquema da Figura 1 e a Tabela 1. As variáveis **A** e **B** são entradas de dados de 4 bits, $\mathbf{S} = \mathbf{S_1 S_0}$ é a entrada de seleção, $\mathbf{F} = \mathbf{F_3 F_2 F_1 F_0}$ é a saída de dados, **over** é uma saída de 1 bit que indica a ocorrência de um *overflow*, e **c_out** é uma saída de *carry out*. As variáveis **A**, **B** e **F** devem ser consideradas como representadas em complemento de 2. Utilize chaves (*switches*) da Basys 3 para serem os bits de entrada e LEDs para os bits de saída (não há necessidade de usar os displays de 7 segmentos).

A operação realizada pela ULA será escolhida por **S** de acordo com a Tabela 1**.** Modos aritméticos serão realizados para $\mathbf{S} = 00$ e $\mathbf{S} = 01$, e modos lógicos para $\mathbf{S} = 10$ e $\mathbf{S} = 11$.

De fato, caso $\mathbf{S} = 00$, a ULA deverá efetuar a soma (em complemento de 2) de **A** com **B** e, caso $\mathbf{S} = 01$, a subtração (em complemento de 2) de **A** por **B**. Caso a operação de soma ou de subtração resulte em *overflow*, então **over** deverá ter o valor 1 e, em caso contrário, o valor 0. De modo semelhante, caso alguma dessas operações resulte em *carry out*, então **c_out** deverá ter o valor 1 e, em caso contrário, o valor 0.

Por outro lado, caso $\mathbf{S} = 10$, a ULA deverá efetuar a operação lógica **A** *and* **B** bit-a-bit e, caso $\mathbf{S} = 11$, a operação lógica **A** *or* **B** bit-a-bit. Nesses dois casos, **over** e **c_out** não fornecem informação alguma, logo poderão ter qualquer valor (*don't care*).

Utilize o seguinte:

- como entradas, as chaves (*switches*);
- como saídas, LEDs.

Para implementar esse projeto, a única biblioteca que poderá ser importada é a *ieee.std_logic_1164*; caso o grupo utilize outra biblioteca, como as consideradas aritméticas (por exemplo, *numeric_std*, *std_logic_arith*, *std_logic_unsigned* e *std_logic_signed*) o experimento receberá nota 0. Também não poderá ser usado o comando PROCESS. O código deve ser todo concorrencial. Basta seguir a aula disponibilizada pelo professor.

Critérios de pontuação. Este projeto será pontuado de acordo com os seguintes critérios:

- Código correto da soma ($\mathbf{S} = 00$): 1,5 ponto.
- Código correto da subtração ($\mathbf{S} = 01$): 1,5 ponto.
- Código correto da operação lógica *and* ($\mathbf{S} = 10$): 1,5 ponto.
- Código correto da operação lógica *or* ($\mathbf{S} = 11$): 1,5 ponto.
- Funcionamento na Basys 3: 4 pontos.

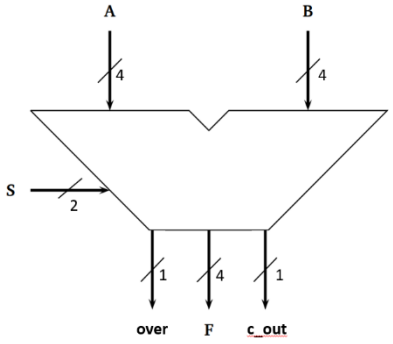

Figura 1. Esquemático da ULA a ser implementada no projeto.

Tabela 1. Tabela Verdade a ser implementada no projeto.

Operação	S_1S_0	$F_3F_2F_1F_0$	over	c_out
SOMA(**A**,**B**)	00	**A**+**B**	*overflow*	*carry out*
SUBT(**A**,**B**)	01	**A**-**B**	*overflow*	*carry out*
AND(**A**,**B**)	10	**A** *and* **B**	X	X
OR(**A**,**B**)	11	**A** *or* **B**	X	X

Tabelas de Verdades: há mais de uma possível tabela de verdade, dependendo do modo como o somador foi feito. Seguem, abaixo, algumas posições dessas tabelas, para rápido conferência.

Tabelas de Verdade 1

A+B

A	B	R	over	c_out
0000	0000	0000	0	0
0001	0010	0011	0	0
0001	0111	1000	1	0
0001	1111	0000	0	1
1000	1010	0010	1	1

A-B

A	B	R	over	c_out
0000	0000	0000	0	0
0001	0001	0000	0	1
0111	1111	1000	1	0
1000	0001	0111	1	1

Tabelas de Verdade 2

A+B

A	B	R	over	c_out
0000	0000	0000	0	0
0001	0010	0011	0	0
0001	0111	1000	1	0
0001	1111	0000	0	1
1000	1010	0010	1	1

A-B

A	B	R	over	c_out
0000	0000	0000	0	1
0001	0001	0000	0	1
0111	1111	1000	1	0
1000	0001	0111	1	1