



Elementos de Sistema - Aula 7 - Handout - Unidade Lógica Aritmética

Rafael Corsi - rafael.corsi@insper.edu.br

Março - 2018

- **ULA**
 - Testando
 - Complicando
- **Adders**
 - Aquecendo
 - Half Adder
 - * planejamento
 - * FPGA
 - EXTRA - Full Adder
 - * planejamento
 - * FPGA

1 ULA

O projeto Aulas/07-Unidade-Logica-Aritmetica/Quartus-ULA/ula.sof fornece um binário da ula que vocês irão fazer no projeto D - ULA, o objetivo dessa etapa é trabalharmos com o controle dos sinais da ULA para validarmos e entendermos as operações da unidade de processamento do nosso computador.

Nessa etapa vocês terão **somente que programar a FPGA com esse binário**, para isso abram o Quartus :

1. Tools -> Programmer
2. Verifique se no Hardware Setup está : ...
3. De um clique no *File* e escolha o arquivo Z01-ULA.sof localizado na pasta :
 - Aulas/07-Unidade-Logica-Aritmetica/Quartus-ULA/ula.sof
4. Clique em Start



Usuários Windows : verifiquem a instalação do driver.

Agora que a FPGA está programada vamos entender o que esse projeto faz, os controles da ULA foram mapeados para as chaves da FPGA e suas saídas para

os LEDs, como mostrado as figuras a seguir :

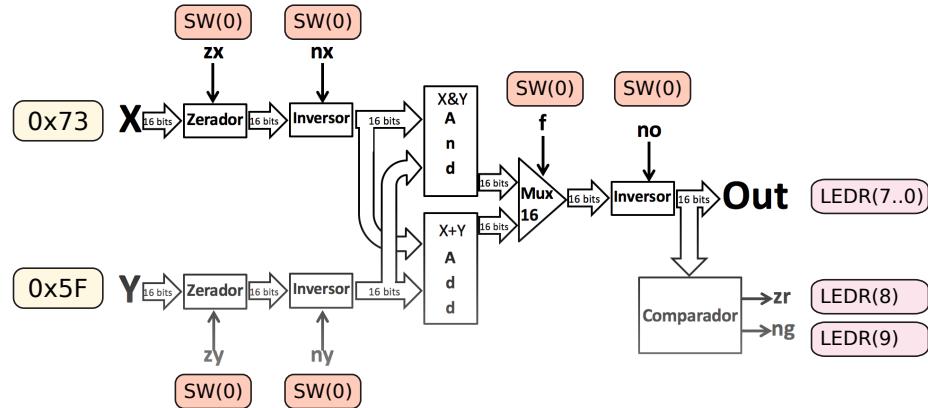


Figure 1: ULA Z01

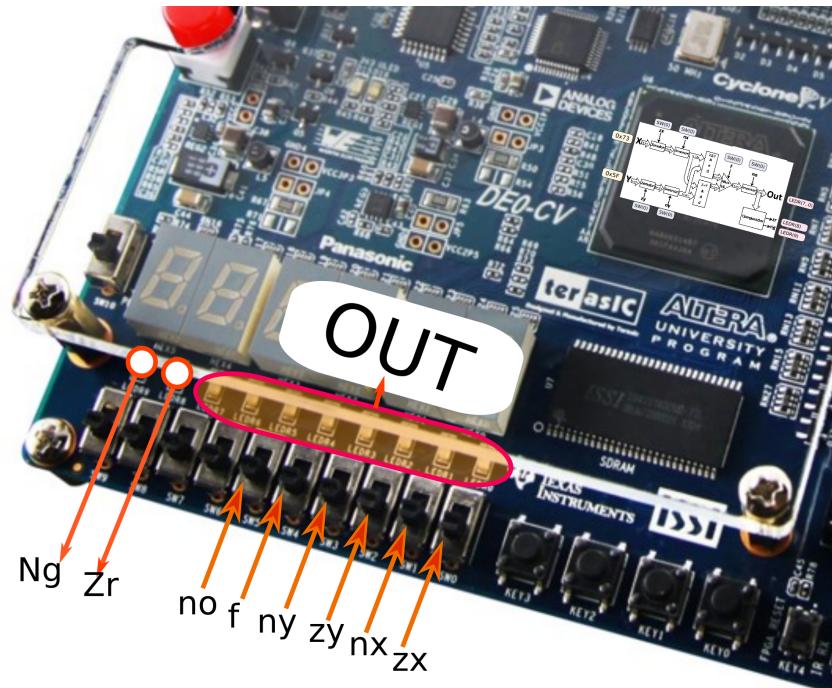


Figure 2: ULA FPGA map

Como fica isso no VHDL ?

```
ula0: ALU port map (
    x              => x,
    Y              => y,
    Zx             => SW(0),
    Nx             => SW(1),
    Zy             => SW(2),
    Ny             => SW(3),
    F              => SW(4),
    No             => SW(5),
    Zr             => LEDR(8),
    Ng             => LEDR(9),
    Saida(7 downto 0) => LEDR(7 downto 0),
    Saida(15 downto 8) => Open
);
```

Note que ULA que iremos desenvolver no projeto D terá 16 bits de largura, porém como temos um limite de LEDs na FPGA, estamos ignorando os valores mais significativos, sem afetar o projeto.

1.1 Testando

Com a FPGA programada podemos testar a ULA modificando seus sinais de controle. A seguir uma proposta de operações lógicas devem ser realizadas na FPGA, seus sinais de controle e resultado devem ser anotados nas tabelas.

O projeto possui previamente definido as entradas da ULA, sendo :

- X = 0x73
- Y = 0x5F

1.1.1 Out = X

- Configure os controles da ULA para fazer com que a saída da ULA seja a entrada X

X	0111 0011
Y	0101 1111
out	
zr	
ng	

zx	nx	zy	ny	f	no

1.1.2 Out = Y

- Configure os controles da ULA para fazer com que a saída da ULA seja a entrada Y

				X	0111 0011
				Y	0101 1111
				out	
				zr	
				ng	

						X	0111 0011
						Y	0101 1111
						out	
						zr	
						ng	

1.1.3 Out = !Y

- Configure os controles da ULA para fazer com que a saída da ULA seja a entrada a entrada Y negada

				X	0111 0011
				Y	0101 1111
				out	
				zr	
				ng	

1.1.4 Out = 0

- Faça com que a saída da ULA seja 0

				X	0111 0011
				Y	0101 1111
				out	
				zr	
				ng	

1.1.5 Out = 1

- Faça com que a saída da ULA seja 1

				X	0111 0011
				Y	0101 1111
				out	
				zr	
				ng	

1.1.6 Out = X + Y

- Faça com que a saída da ULA seja a entrada X + a entrada Y

X	0111 0011
Y	0101 1111
out	
zr	
ng	

zx	nx	zy	ny	f	no

1.2 Complicando

Algumas operações não triviais :

1.2.1 Out = X ou Y

- Faça com que a saída da ULA seja X ou Y

X	0111 0011
Y	0101 1111
out	
zr	
ng	

zx	nx	zy	ny	f	no

1.2.2 Out = X - Y

- Faça com que a saída da ULA seja a entrada X menos a entrada Y

X	0111 0011
Y	0101 1111
out	
zr	
ng	

zx	nx	zy	ny	f	no

2 Adders

Iremos agora começar o desenvolvimento da ULA recém utilizada, para isso vamos implementar alguns módulos entre eles dois somadores (HALF-ADDER e FULL-ADDER).

2.1 Aquecendo

Abra o projeto no Quartus localizado na pasta : *Aulas/07-Unidade-Logica-Aritmetica/Quartus-Add/*, compile e grave na FPGA. Esse projeto inicial faz com que o LEDR(0) seja a chave SW(0) negada. Essa lógica está inserida no TopLevel.vhd onde fazemos o mapeamento final dos pinos do projeto.

```
-- Entrada e saídas da FPGA
-----
entity TopLevel is
    port(
        SW      : in  std_logic_vector(9 downto 0);
        LEDR    : out std_logic_vector(9 downto 0)
    );
end entity;
```

SW e LEDR são os pinos da FPGA que estão mapeados respectivamente nas chaves e nos LEDs. Note que o tamanho do vetor referente as chaves a aos leds possui tamanho 10 (9 downto 0), ou seja, temos 10 chaves e 10 leds mapeados na FPGA.

Na arquitetura desse entidade temos inicialmente a seguinte implementação :

```
begin
    LEDR(0) <= NOT SW(0);

end rtl;
```

1. Compile o projeto (*Processing -> Start Compilation* ou *Ctrl+L*)
2. Analise o RTL gerado (*Tools -> NetList Viewers -> RTL Viewer*)
3. Programe a FPGA (*Tools -> Programmer*)
4. Teste mexendo na chave SW(0)

Note que no arquivo TopLevel.vhd (a entidade de maior nível) possui a declaração de dois componentes :

```
component HalfAdder is
    port(
```

```

        a,b:      in STD_LOGIC;  -- entradas
        soma,vaium: out STD_LOGIC  -- sum e carry
    );
end component;

component FullAdder is
port(
    a,b,c:      in STD_LOGIC;  -- entradas
    soma,vaium: out STD_LOGIC  -- sum e carry
);
end component;

```

Componentes em VHDL possuem conceito similar a componentes físicos (de hardware), onde podemos interligar blocos para montarmos um lógica mais complexas. Muito parecido com o que foi feito no projeto B onde usamos “componentes” para implementarmos uma lógica mais complexa.

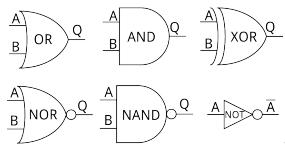
Esses componentes estão vazios, será necessário fazermos a implementação deles.

2.2 Half Adder

2.2.1 planejamento

Implemente um half-adder com as portas lógicas sugeridas a seguir:

a	b	soma	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



2.2.2 FPGA

Agora que já sabemos como é feita a implementação de um half-adder iremos implementar na FPGA esse esquema.

Para isso devemos primeiramente fazer algo que chamamos de port map, que é utilizarmos o componente e mapearmos suas entradas e/ou saídas.

Modifique o código TopLevel para :

```
-----
-- implementacao
-----
begin
    u1 : HalfAdder port map (
        a      => SW(0),
        b      => SW(1),
        soma   => LEDR(0),
        vaium  => LEDR(1)
    );
end rtl;
```

1. Compile o projeto
2. Analise o RTL gerado

2.2.2.1 Arquitetura HalfAdder

O componente HalfAdder ainda não foi implementando, abra o arquivo : Quartus-Adder/rtl/HalfAdder.vhd, o mesmo deve ter o seguinte conteúdo :

```
-- Elementos de Sistemas
-- by Luciano Soares
-- HalfAdder.vhd

-- Implementa Half Adder

Library ieee;
use ieee.std_logic_1164.all;

entity HalfAdder is
    port(
        a,b:      in STD_LOGIC;    -- entradas
        soma,vaium: out STD_LOGIC  -- sum e carry
    );
end entity;

architecture arch of HalfAdder is

begin

end architecture;
```

Note a sua arquitetura está vazia, transcreva o que foi desenvolvido com portas lógicas para o VHDL.

2.2.2.2 Testando

Para testar basta compilarmos e gravarmos o projeto :

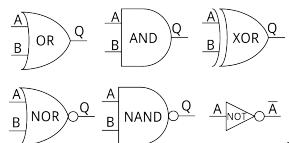
1. Compile o projeto
2. Analise o RTL gerado
3. Programe a FPGA
4. Teste mexendo nas chaves :
 - SW(0) : entrada A
 - SW(1) : entrada B
 - LEDR(0) : saída Soma
 - LEDR(1) : saída vatum (carry)

2.3 EXTRA - Full Adder

2.3.1 planejamento

Implemente um half-adder com as portas lógicas sugeridas a seguir:

a	b	c	soma	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



2.3.2 FPGA

Agora que já sabemos como é feita a implementação de um full-adder iremos implementar na FPGA esse esquema.

Para isso devemos primeiramente fazer algo que chamamos de port map, que é utilizarmos o componente e mapearmos suas entradas e/ou saídas.

Modifique o código TopLevel para :

```
-----
-- implementacao
-----
begin
    u1 : FullAdder port map (
        a      => SW(0),
        b      => SW(1),
        c      => SW(2),
        soma   => LEDR(0),
        vaium  => LEDR(1)
    );
end rtl;
```

2.3.2.1 Arquitetura FullAdder

Note a sua arquitetura está vazia, transcreva o que foi desenvolvido com portas lógicas para o VHDL.

2.3.2.2 Testando

Para testar basta compilarmos e gravarmos o projeto :

1. Compile o projeto
2. Analise o RTL gerado
3. Programe a FPGA
4. Teste mexendo nas chaves :
 - SW(0) : entrada A
 - SW(1) : entrada B
 - SW(2) : entrada C
 - LEDR(0) : saída Soma
 - LEDR(1) : saída vaium (carry)