ULA – RELATÓRIO SIMPLES

AUTOR: GUILHERME ZANINI DE AS

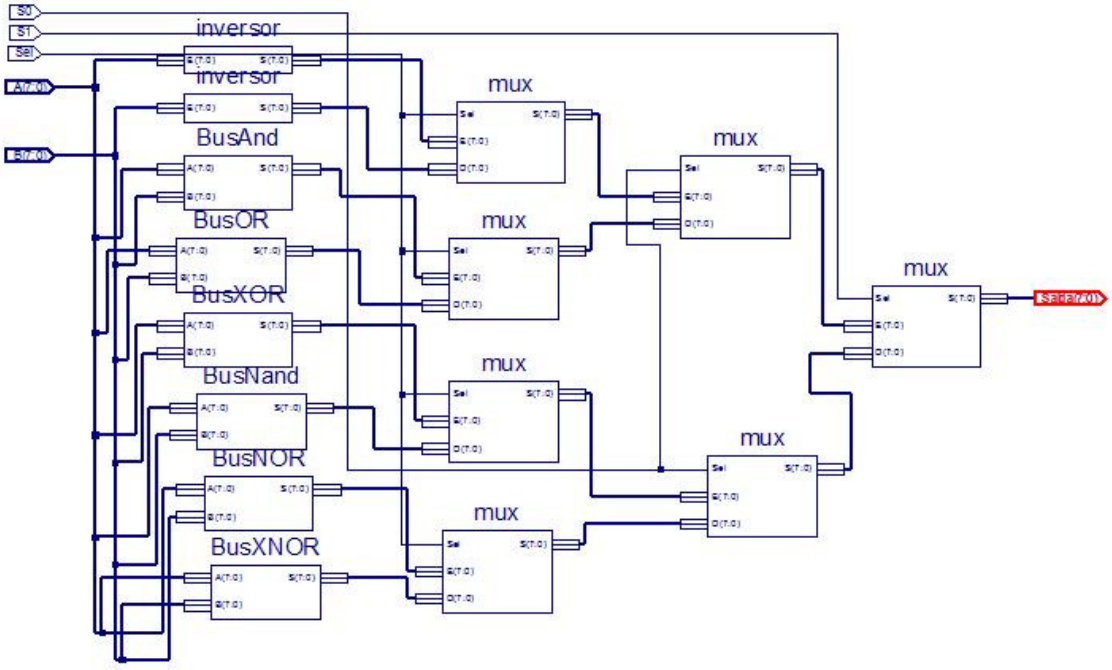
PRIMEIRAMENTE: TABELA DE OPERAÇÕES DA ULA

Tabela de Operações da ULA				
Funções Lógicas (modo=0)				
S(1)	S(0)	Cin	Função	Operação
0	0	0	A’	NOT A
0	0	1	B’	NOT B
0	1	0	AB	AND
0	1	1	A+B	OR
1	0	0	A⊕B	XOR
1	0	1	(AB)’	NAND
1	1	0	(A+B)’	NOR
1	1	1	(A⊕B)’	XNOR
Funções Aritméticas (modo=1)				
S(1)	S(0)	Cin	Função	Operação
0	0	0	A	TRANS A
0	0	1	A+1	INC A
0	1	0	A+B	A e B
0	1	1	A+B+1	INC A e B
1	0	0	A+B’	A e complemento B
1	0	1	A-B	Sub B de A
1	1	0	A’+B	B e complemento de A
1	1	1	B-A	

Após saber quais as operações possíveis na Unidade Lógica e Aritmética, começaremos a construir as partes. Das menores (através da linguagem VHDL) para as maiores (por Esquemáticos).

UNIDADE LÓGICA

A UNIDADE LÓGICA NECESSITA DOS SEGUINTES COMPONENTES:



DUAS ENTRADAS PARA SELEÇÃO, SENDO NO CASO ACIMA, **S0** E **S1**, DUAS ENTRADAS **A** E **B** COM BUS DE 8 BITS E OUTRA ENTRADA DE SELEÇÃO **SEL (DO MULTIPLEXADOR)**, COM APENAS UMA SAÍDA DE 8 BITS. COMO TODAS AS OPERAÇÕES (JÁ CITADAS NA TABELA) SÃO EXECUTADAS AO MESMO TEMPO, SURGE AÍ, A NECESSIDADE DE MULTIPLEXADORES PARA SELECIONAR O CAMINHO ATÉ A SAÍDA, POR ISSO, AS ENTRADAS DE SELEÇÃO, **S0, S1 E SEL.**

Inversor (not A e not B) em VHDL

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

  entity inversor is
      Port ( E : in  STD_LOGIC_VECTOR (7 downto 0);
             S : out  STD_LOGIC_VECTOR (7 downto 0));
  end inversor;

  architecture Behavioral of inversor is

  begin

    S <= Not E;

  end Behavioral;
```

Porta lógica AND de 8 bits em VHDL.

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

entity BusAnd is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
           B : in  STD_LOGIC_VECTOR (7 downto 0);
           S : out  STD_LOGIC_VECTOR (7 downto 0));
end BusAnd;

architecture Behavioral of BusAnd is

begin

   S<= A AND B;

end Behavioral;
```

Bus OR em VHDL

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

  entity BusOR is
      Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
             B : in  STD_LOGIC_VECTOR (7 downto 0);
             S : out  STD_LOGIC_VECTOR (7 downto 0));
  end BusOR;

  architecture Behavioral of BusOR is

  begin

     S<= A OR B;

  end Behavioral;
```

Bus XOR em VHDL

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

 entity BusXOR is
     Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
            B : in  STD_LOGIC_VECTOR (7 downto 0);
            S : out  STD_LOGIC_VECTOR (7 downto 0));
 end BusXOR;

 architecture Behavioral of BusXOR is

 begin

    S<= A XOR B;

 end Behavioral;
```

Bus NAND em VHDL

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

entity BusNand is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
           B : in  STD_LOGIC_VECTOR (7 downto 0);
           S : out  STD_LOGIC_VECTOR (7 downto 0));
end BusNand;

architecture Behavioral of BusNand is

begin

   S<= A NAND B;

end Behavioral;
```

Bus NOR em VHDL

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

entity BusNOR is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
           B : in  STD_LOGIC_VECTOR (7 downto 0);
           S : out  STD_LOGIC_VECTOR (7 downto 0));
end BusNOR;

architecture Behavioral of BusNOR is

begin

   S<= A NOR B;

end Behavioral;
```

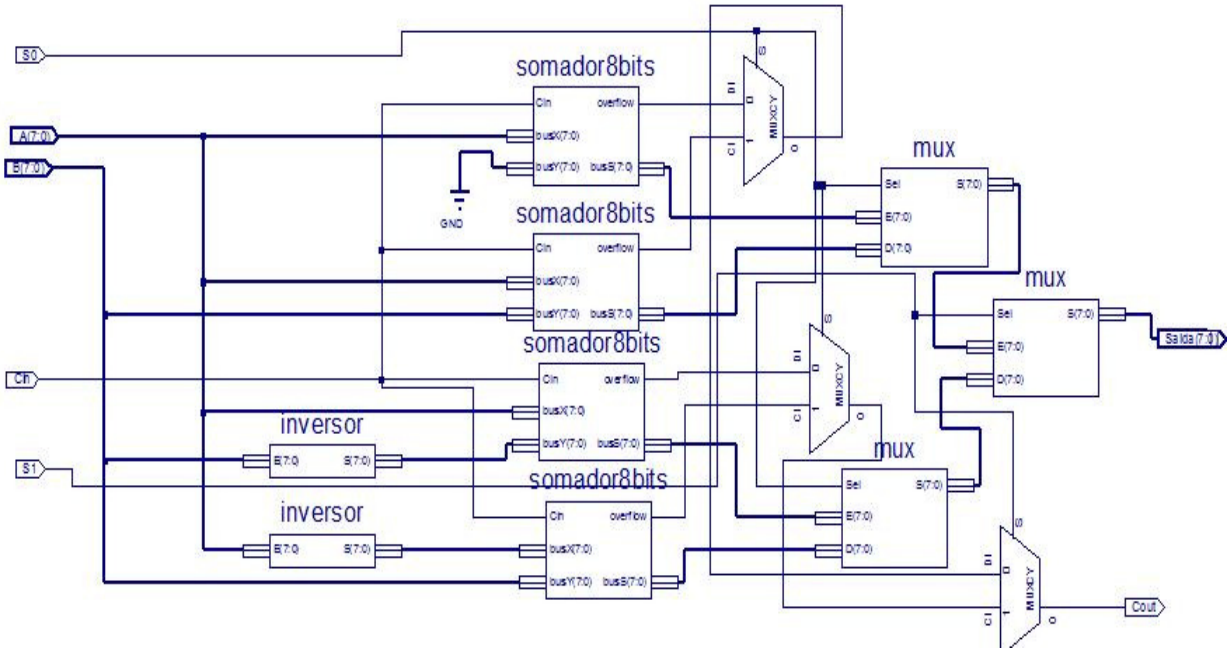
Bus XNOR em VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity BusXNOR is
     Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
            B : in  STD_LOGIC_VECTOR (7 downto 0);
            S : out  STD_LOGIC_VECTOR (7 downto 0));
 end BusXNOR;

 architecture Behavioral of BusXNOR is

 begin

  S<= A XNOR B;

 end Behavioral;
```

Pronto os componentes da Unidade faremos os multiplexadores.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity mux is
     Port ( E : in  STD_LOGIC_VECTOR (7 downto 0);
            D : in  STD_LOGIC_VECTOR (7 downto 0);
            Sel: in  STD_LOGIC;
            S : out  STD_LOGIC_VECTOR (7 downto 0));
 end mux;
'
 architecture Behavioral of mux is

 begin

'S<= E when sel='0' else D;

'end Behavioral;
```

UNIDADE ARITMÉTICA



A UNIDADE ARITMÉTICA POSSUI AS MESMAS ENTRADAS DA UNIDADE LÓGICA. A NECESSIDADE AQUI SE VALE DE 4 SOMADORES DE 8 BITS, 2 INVERSORES, 3 MUX DE 1 BIT E 3 MUX DE 8BITS E UM COMPONENTE GND (FIO TERRA) DE 8BITS.

Somador de 8 bits.

Necessita-se de somadores de 1 bit.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity somador is
    Port ( x : in  STD_LOGIC;
           y : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           cout : out  STD_LOGIC;
           s : out  STD_LOGIC);
end somador;

architecture Behavioral of somador is

begin

s <= (x xor y xor cin);
   cout <= (x and y) or (x and cin) or (y and cin);

end Behavioral;
```

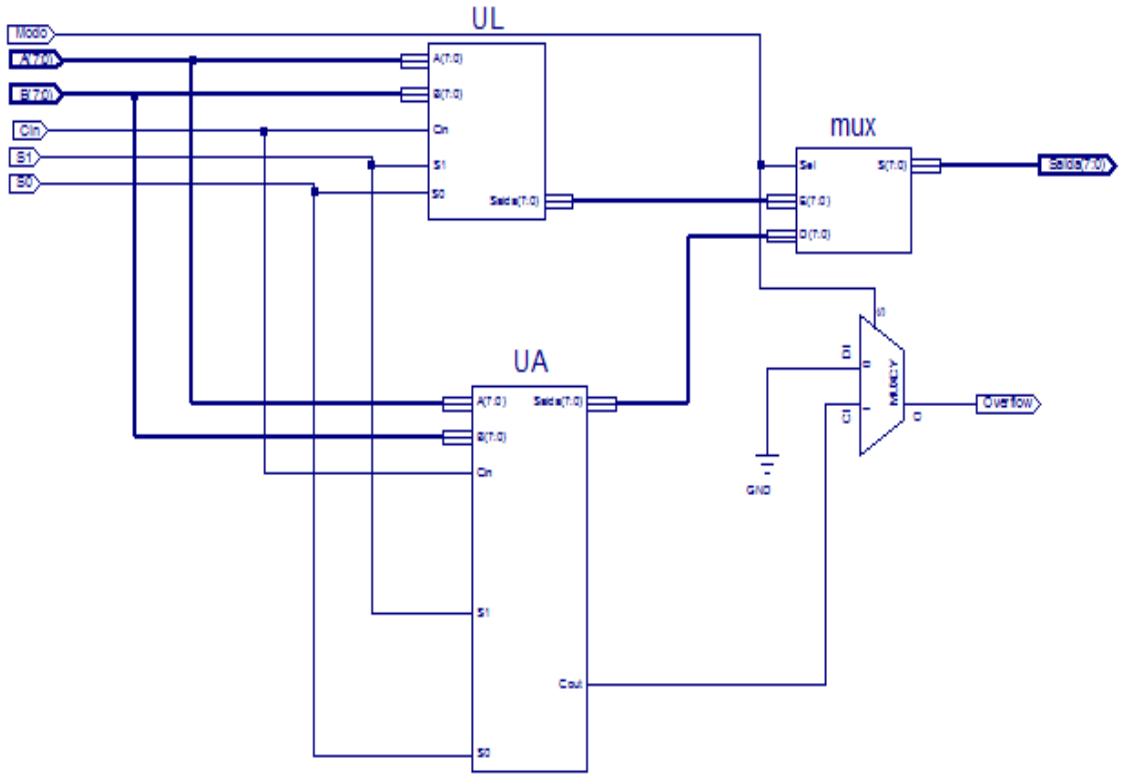
Agora o de 8 Bits

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity somador8bits is
    Port ( busX : in  STD_LOGIC_VECTOR (7 downto 0);
           busY : in  STD_LOGIC_VECTOR (7 downto 0);
           Cin : in  STD_LOGIC;
           busS : out  STD_LOGIC_VECTOR (7 downto 0);
           overflow : out  STD_LOGIC);
end somador8bits;

architecture Behavioral of somador8bits is

   COMPONENT somador
   PORT(
      X : IN std_logic;
      Y : IN std_logic;
      Cin : IN std_logic;
      S : OUT std_logic;
      Cout : OUT std_logic
      );
   END COMPONENT;
```

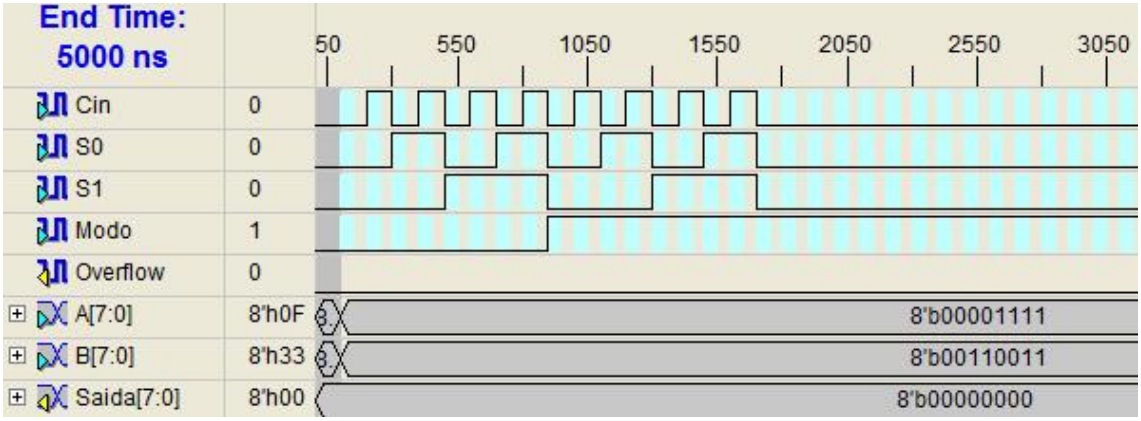
```
   SIGNAL C0 :  std_logic;
   SIGNAL C1 :  std_logic;
   SIGNAL C2 :  std_logic;
   SIGNAL C3 :  std_logic;
   SIGNAL C4 :  std_logic;
   SIGNAL C5 :  std_logic;
   SIGNAL C6 :  std_logic;
   SIGNAL C7 :  std_logic;

begin

   b0: somador PORT MAP(
      X => busX(0),
      Y => busY(0),
      Cin => Cin,
      S => busS(0),
      Cout => C0
   );
   b1: somador PORT MAP(
      X => busX(1),
      Y => busY(1),
      Cin => C0,
      S => busS(1),
      Cout => C1
   );
```

```
    b2: somador PORT MAP(
       X => busX(2),
       Y => busY(2),
       Cin => C1,
       S => busS(2),
       Cout => C2
    );
    b3: somador PORT MAP(
       X => busX(3),
       Y => busY(3),
       Cin => C2,
       S => busS(3),
       Cout => C3
    );
    b4: somador PORT MAP(
       X => busX(4),
       Y => busY(4),
       Cin => C3,
       S => busS(4),
       Cout => C4
    );
    b5: somador PORT MAP(
       X => busX(5),
       Y => busY(5),
       Cin => C4,
       S => busS(5),
       Cout => C5
    );

    b6: somador PORT MAP(
        X => busX(6),
        Y => busY(6),
        Cin => C5,
        S => busS(6),
        Cout => C6
    );
    b7: somador PORT MAP(
        X => busX(7),
        Y => busY(7),
        Cin => C6,
        S => busS(7),
        Cout => C7
    );

    overflow <= C6 XOR C7;

end Behavioral;
```

PRONTO, MONTAREMOS O ESQUEMATICO DA **ULA**, COM A **UL** E A **UA**. E LOGO APÓS FAREMOS O TESTE PARA AVERIGUAR SE ESTÁ CORRETO.



OS TESTE FORAM FEITOS EM "**TEST BENCH WAVE FORM**", E PARA MELHOR VISUALIZAÇÃO UTILIZEI NÚMEROS BINÁRIOS.



PRONTO.

SAIU O SEGUINTE RESULTADO: