UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

CIÊNCIA DA COMPUTAÇÃO

CIRCUITOS DIGITAIS II

RELATÓRIO DA UNIDADE LÓGICA E ARITMÉTICA

Data: 29/05/2018

Professor: Paulo Roberto de Oliveira

Discentes

R.A	Nome
103277	Gabriel Arruda Andrella
103235	Caio Eduardo Kikuti Machado
103491	Luiz Fellipe Machi Pereira

Sumário

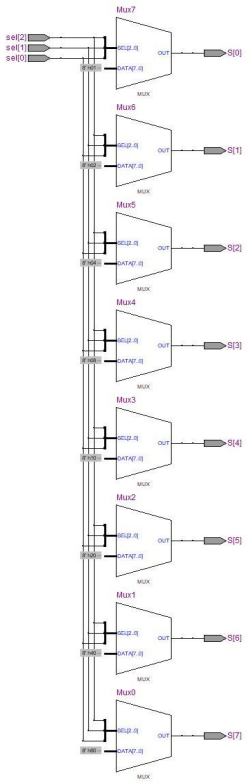
Descrição do Projeto

Montagem do Pacote do decodificador

Como a ULA de 1 bit possui 8 operações se faz necessária a montagem de um circuito decodificador para que seja possível selecionar a operação de escolha. O decodificador implementado deve conter 3 entradas de controle para poder alternar entre os 8 procedimentos.

Para a montagem do circuito foi utilizada o software *Quartus II* , através da utilização de pacotes (*packages*) e modelagem estrutural. O circuito decodificador está contido no pacote "Decodificador". Esse circuito foi implementado utilizando a função "*CASE WHEN*" da linguagem VHDL. Dessa maneira foram utilizados dois vetores, um vetor de três posições para as entradas de controle e um vetor de oito posições para ativação da saída.

O Circuito Decodificador trabalha alternando entre as oito combinações geradas pelas entradas de controle. Cada uma dessas combinações ativa uma operação correspondente na ULA quer sejam operações lógicas: AND, NAND, OR , NOR, XOR, XNOR, ou aritméticas: soma e subtração realizadas pelos circuitos somador e subtrator respectivamente.



Código do Circuito Decodificador (VHDL)

```
ENTITY decod IS
      PORT(
            sel: IN BIT_VECTOR(2 DOWNTO 0);
            S: OUT BIT_VECTOR(7 DOWNTO 0));
END decod;

ARCHITECTURE logic OF decod IS
BEGIN
      PROCESS(sel)
      BEGIN
            CASE(sel) IS
                  WHEN "000" => S <= "00000001";
                  WHEN "001" => S <= "00000010";
                  WHEN "010" => S <= "00000100";
                  WHEN "011" => S <= "00001000";
                  WHEN "100" => S <= "00010000";
                  WHEN "101" => S <= "00100000";
                  WHEN "110" => S <= "01000000";
                  WHEN "111" => S <= "10000000";
                  WHEN OTHERS => S <= "00000000";
            END CASE;
      END PROCESS;
END logic;

PACKAGE seletor IS
      COMPONENT decod IS
            PORT(
                  sel: IN BIT_VECTOR(2 DOWNTO 0);
                  S: OUT BIT_VECTOR(7 DOWNTO 0)
            );
      END COMPONENT;
END seletor;
```

Montagem do Pacote da Unidade Lógica

Portas logicas utilizadas:

Todas essas operações estão no pacote lógico e foram implementadas em VHDL utilizando modelagem comportamental através do comando *"IF THEN ELSE"*.

AND

Operação lógica que resulta em valor lógico verdadeiro se e somente se todas as entradas forem '1' e falso para os demais. Expressão booleana $A \cdot B$

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Codigo VHDL AND_1

```
ENTITY and_1 IS
      PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
END and_1;

ARCHITECTURE logic OF and_1 IS
      BEGIN
            PROCESS(E0,E1)
            BEGIN
                  IF(E0 = '1' and E1 = '1') THEN
                        S <= '1';
                  ELSE
                        S <= '0';
                  END IF;
            END PROCESS;
END logic;
```

OR

Operação lógica que resulta em valor lógico verdadeiro se alguma das entradas tiverem o valor '1' e falso para os demais. Expressão booleana $A + B$. De forma mais específica, podemos notar através do código do pacote dos componentes que existe um porta lógica nominada "*or_2*", essa porta, na ULA de 1 bit, tem a função de afunilar a saída tornando-a única.

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Codigo VHDL OR_1

```
ENTITY or_1 IS
      PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
END or_1;
ARCHITECTURE logic OF or_1 IS
      BEGIN
            PROCESS(E0,E1)
            BEGIN
                  IF(E0 = '0' and E1 = '0') THEN
                        S <= '0';
                  ELSE
                        S <= '1';
                  END IF;
            END PROCESS;
END logic;
```

Codigo VHDL OR_2

```
ENTITY or_2 IS
      PORT(
            A,B,C,D,E,F,G,H: IN BIT;
            S: OUT BIT);
END or_2;
```

```
ARCHITECTURE logic OF or_2 IS
      BEGIN
            PROCESS(A,B,C,D,E,F,G)
            BEGIN
                  IF(A = '1' OR B = '1' OR C = '1' OR D = '1' OR E = '1' OR
F = '1' OR G = '1' OR H = '1') THEN
                        S <= '1';
                  ELSE
                        S <= '0';
                  END IF;
            END PROCESS;
END logic;
```

NAND

Operação lógica que terá nível lógico falso se as entradas forem '1' e nível lógico verdadeiro para os demais.Expressão booleana $\overline{A.B}$

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Codigo VHDL NAND_1

```
ENTITY nand_1 IS
      PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
END nand_1;

ARCHITECTURE logic OF nand_1 IS
      BEGIN
            PROCESS(E0,E1)
            BEGIN
                  IF(E0 = '0' and E1 = '0') THEN
                        S <= '0';
                  ELSE
                        S <= '1';
```

```
                END IF;
            END PROCESS;
END logic;
```

NOR

Operação lógica que terá nível lógico verdadeiro se as entradas forem '0' e falso para os demais.
Expressão booleana $\overline{A+B}$

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Codigo VHDL NOR_1

```
ENTITY nor_1 IS
      PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
END nor_1;

ARCHITECTURE logic OF nor_1 IS
      BEGIN
            PROCESS(E0,E1)
            BEGIN
                  IF(E0 = '0' and E1 = '0') THEN
                        S <= '1';
                  ELSE
                        S <= '0';
                  END IF;
            END PROCESS;
END logic;
```

XOR

Operação lógica que terá nível lógico falso se as entradas forem iguais, e nível lógico verdadeiro para as demais entradas.Expressão booleana A $\oplus$ B

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Codigo VHDL XOR_1

```
ENTITY xor_1 IS
      PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
END xor_1;

ARCHITECTURE logic OF xor_1 IS
      BEGIN
            PROCESS(E0,E1)
            BEGIN
                  IF(E0 = E1) THEN
                        S <= '0';
                  ELSE
                        S <= '1';
                  END IF;
            END PROCESS;
END logic;
```

XNOR

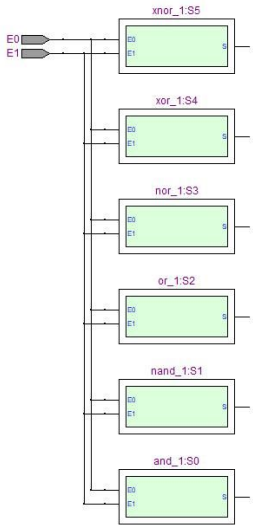
Operação Lógica que terá nível lógico verdadeiro se as entradas forem iguais, e nível lógico falso para as demais entradas. Expressão booleana $A \odot B$ ou $\overline{A \oplus B}$

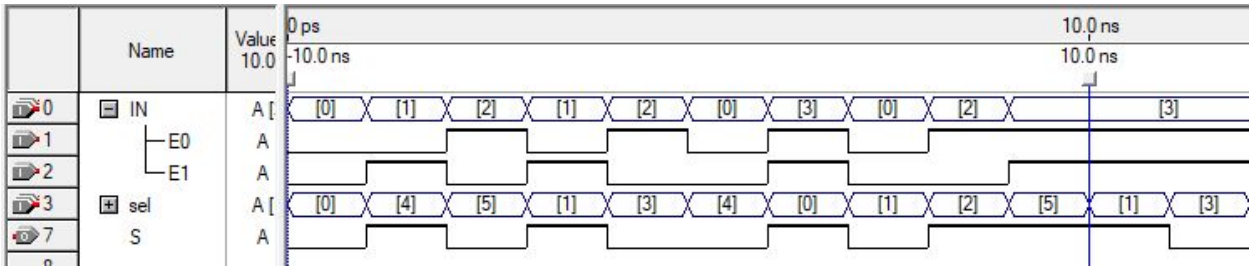
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Codigo VHDL XNOR_1

```
ENTITY xnor_1 IS
      PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
END xnor_1;
ARCHITECTURE logic OF xnor_1 IS
      BEGIN
            PROCESS(E0,E1)
            BEGIN
                  IF(E0 = E1) THEN
                        S <= '1';
                  ELSE
                        S <= '0';
                  END IF;
            END PROCESS;
END logic;
```

Representação e simulação da Unidade Lógica





Pacote da Unidade Lógica

```
PACKAGE portas IS
     COMPONENT and_1 IS
          PORT(
               E0,E1: IN BIT;
               S: OUT BIT);
     END COMPONENT;

     COMPONENT nand_1 IS
          PORT(
               E0,E1: IN BIT;
               S: OUT BIT);
     END COMPONENT;
```

```
    COMPONENT or_1 IS
        PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
    END COMPONENT;

    COMPONENT or_2 IS
        PORT(
            A,B,C,D,E,F,G,H: IN BIT;
            S: OUT BIT);
    END COMPONENT;

    COMPONENT nor_1 IS
        PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
    END COMPONENT;

    COMPONENT xor_1 IS
        PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
    END COMPONENT;

    COMPONENT xnor_1 IS
        PORT(
            E0,E1: IN BIT;
            S: OUT BIT);
    END COMPONENT;
END portas;
```

Montagem do Pacote da Unidade Aritmética

Circuito Somador

O Circuito Somador completo pode ser representado por 3 diferentes entradas que resultam em uma soma aritmética. Dessa maneira temos duas entradas E1 e E2 e uma entrada *Carry in* e temos uma saída de sinal S e um *Carry out.* O funcionamento deste circuito pode ser descrito pela expressão booleana: $E1 \oplus E2 \oplus Cin$ ou então através da tabela verdade abaixo.

E1	E2	Cin	S	Cout
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

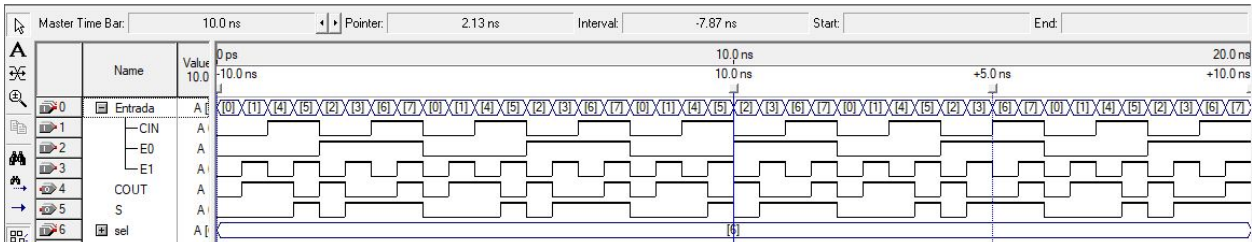
A implementação deste circuito feita através da linguagem de programação VHDL utilizando o comando "*WHEN ELSE*" . Este comando é concorrente e é utilizado para transferir o valor de uma expressão para um sinal destino satisfeita uma condição. Para tanto foram utilizados dois vetores um de três posições para guardar as entradas e um de duas posições para receber o resultado da soma aritmética.

Código VHDL

```
ENTITY somador IS
     PORT(
          E0,E1,CIN: IN BIT;
          S, COUT: OUT BIT);
END somador;

ARCHITECTURE logic OF somador IS
SIGNAL INPUTS: BIT_VECTOR(2 DOWNTO 0);
```

```
SIGNAL OUTPUTS: BIT_VECTOR(1 DOWNTO 0);
BEGIN
      INPUTS(2) <= E0;
      INPUTS(1) <= E1;
      INPUTS(0) <= CIN;

      OUTPUTS <= "00" WHEN INPUTS = "000" ELSE
            "10" WHEN INPUTS = "010" ELSE
            "10" WHEN INPUTS = "100" ELSE
            "01" WHEN INPUTS = "110" ELSE
            "10" WHEN INPUTS = "001" ELSE
            "01" WHEN INPUTS = "011" ELSE
            "01" WHEN INPUTS = "101" ELSE
            "11";
      S <= OUTPUTS(0);
      COUT <= OUTPUTS(1);
END logic;
```

Simulação



Circuito Subtrator

O circuito subtrator possui três entradas E1, E2 e o *Carry in*, com saidas S e *Carry out*. Ele é utilizado para subtrair números de pelo menos duas casas.O funcionamento deste circuito pode ser descrito pela expressão booleana: $E1 \oplus E2 \oplus Cin$ ou então através da tabela verdade abaixo.

E1	E2	Cin	S	Cout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

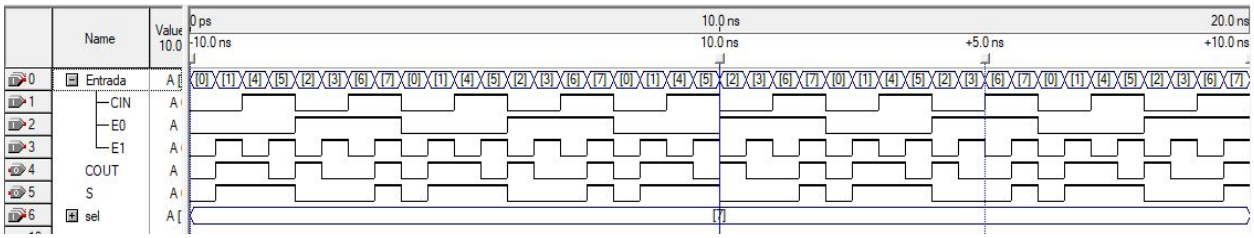
A implementação deste circuito feita através da linguagem de programação VHDL utilizando o comando *" CASE WHEN"* .Este comando é concorrente e é utilizado para transferir o valor de uma expressão para um sinal destino satisfeita uma condição. Para tanto foram utilizados dois vetores um de três posições para guardar as entradas e um de duas posições para receber o resultado da soma aritmética.

Codigo VHDL

```
ENTITY subtrator IS
      PORT(
            E0,E1,CIN: IN BIT;
            S, COUT: OUT BIT);
END subtrator;

ARCHITECTURE logic OF subtrator IS
SIGNAL INPUTS: BIT_VECTOR(2 DOWNTO 0);
SIGNAL OUTPUTS: BIT_VECTOR(1 DOWNTO 0);
BEGIN
      PROCESS(INPUTS)
```

```
      BEGIN
            INPUTS(2) <= E0;
            INPUTS(1) <= E1;
            INPUTS(0) <= CIN;

            CASE INPUTS IS
                  WHEN "000" => OUTPUTS <= "00";
                  WHEN "010" => OUTPUTS <= "11";
                  WHEN "100" => OUTPUTS <= "10";
                  WHEN "110" => OUTPUTS <= "00";
                  WHEN "001" => OUTPUTS <= "11";
                  WHEN "011" => OUTPUTS <= "01";
                  WHEN "101" => OUTPUTS <= "00";
                  WHEN "111" => OUTPUTS <= "11";
            END CASE;

            S <= OUTPUTS(0);
            COUT <= OUTPUTS(1);
      END PROCESS;
END logic;
```

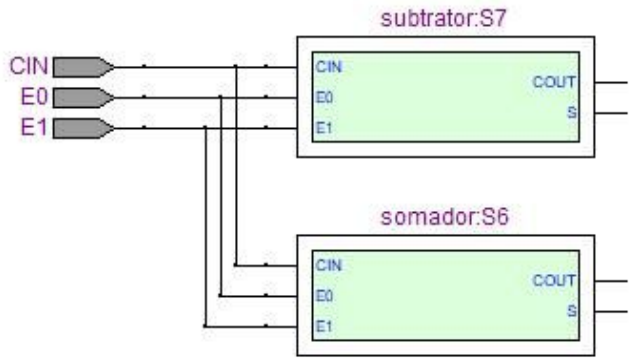
Simulação



Pacote Unidade Aritmética

```
PACKAGE UA IS
      COMPONENT somador IS
            PORT(
                  E0,E1,CIN: IN BIT;
                  S, COUT: OUT BIT);
      END COMPONENT;

      COMPONENT subtrator IS
            PORT(
                  E0,E1,CIN: IN BIT;
                  S, COUT: OUT BIT););
      END COMPONENT;
END UA;
```

Representação da Unidade Aritmética



Funcionamento Geral da ULA de 1 bit

A ULA de 1 bit realiza no total oito operações selecionadas através de um decodificador que possui 3 entradas de controle. Selecionada uma das oito combinações geradas pelo decodificador a ULA executa uma operação específica. Os procedimentos encontram-se divididos em operações lógicas e aritméticas e ocorrem conforme descrito na tabela abaixo.

ATIVAÇÃO	OPERAÇÃO	ALGEBRA DE BOOLE
000	SOMADOR	$E1 \oplus E2 \oplus Cin$
001	SUBTRATOR	$E1 \oplus E2 \oplus Cin$
010	AND	$E1 \,.\, E2$
011	OR	$E1 + E2$
100	NAND	$\overline{E1.E2}$
101	NOR	$\overline{E1 + E2}$
110	XOR	$E1 \oplus E2$
111	XNOR	$E1 \odot E2$

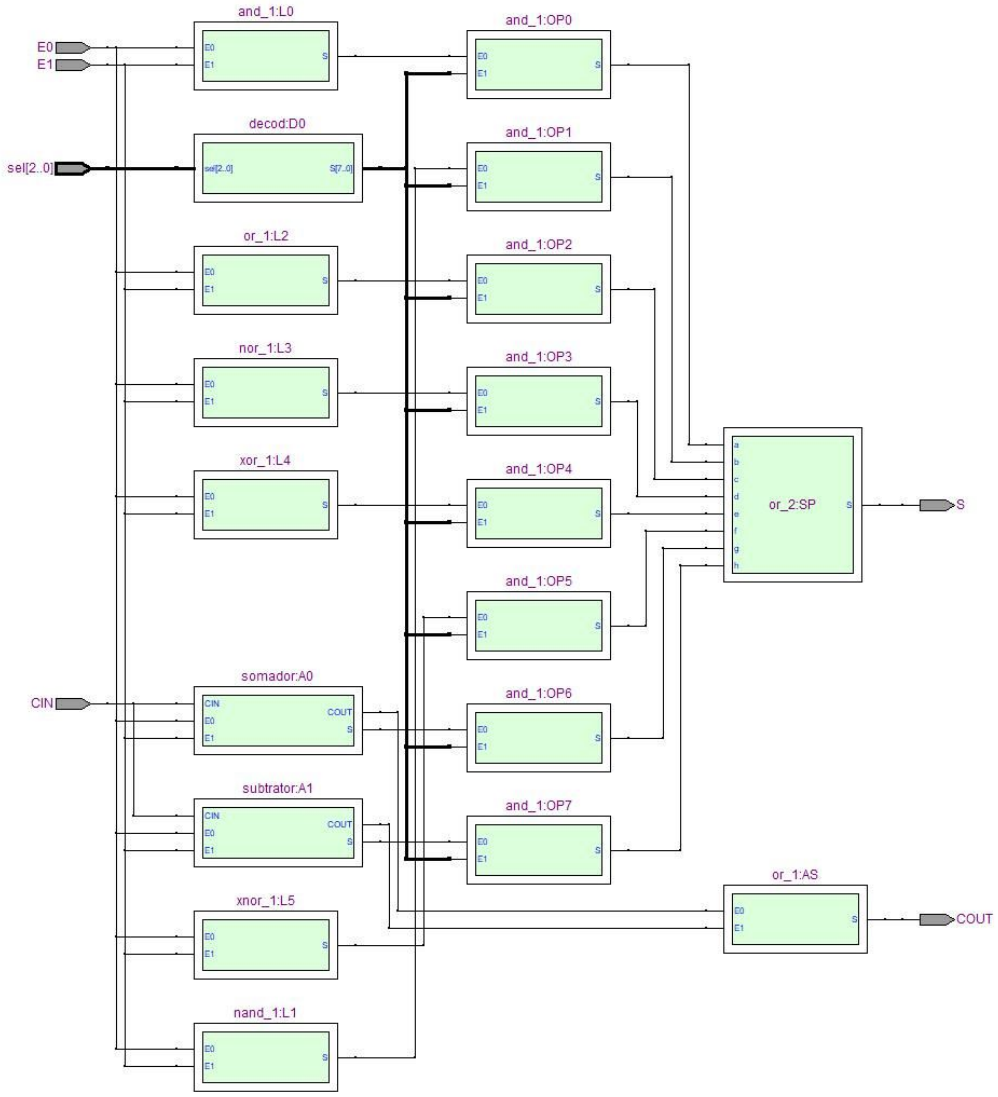
Codigo VHDL

```
LIBRARY decodi,logic,ari;
USE decodi.seletor.all, logic.portas.all, ari.UA.all;

ENTITY ULA1BIT IS
     PORT(
          sel: IN BIT_VECTOR(2 DOWNTO 0);
          E0, E1, CIN: IN BIT;
          COUT, S: OUT BIT);
END ULA1BIT;
```

```
ARCHITECTURE des OF ULA1BIT IS

SIGNAL P0, P1, P2, P3, P4, P5, RSO, RSU, OSO, OSU, R1, R2, R3, R4,
R5, R6, R7, R8: BIT;
SIGNAL PORT_DECOD : BIT_VECTOR(7 DOWNTO 0);

BEGIN
     -------------ATRIBUI A SELEÇÃO-----------------------------
     selecao: decod PORT MAP(sel, PORT_DECOD);
     -------------REALIZA AS OPERAÇÕES---------------------
     S0: and_1 PORT MAP(E0,E1,P0);
     S1: nand_1 PORT MAP(E0,E1,P1);
     S2: or_1 PORT MAP(E0,E1,P2);
     S3: nor_1 PORT MAP(E0,E1,P3);
     S4: xor_1 PORT MAP(E0,E1,P4);
     S5: xnor_1 PORT MAP(E0,E1,P5);
     S6: somador PORT MAP(E0,E1,CIN,RSO,OSO);
     S7: subtrator PORT MAP(E0,E1,CIN,RSU,OSU);
     -----JUNÇÃO DO DECODIFICADOR COM AS OPERAÇÕES----------
     oAND: and_1 PORT MAP(P0,PORT_DECOD(0),R1);
     oNAND: and_1 PORT MAP(P1,PORT_DECOD(1),R2);
     oOR: and_1 PORT MAP(P2,PORT_DECOD(2),R3);
     oNOR: and_1 PORT MAP(P3,PORT_DECOD(3),R4);
     oXOR: and_1 PORT MAP(P4,PORT_DECOD(4),R5);
     oXNOR: and_1 PORT MAP(P5,PORT_DECOD(5),R6);
     oSOM: and_1 PORT MAP(RSO,PORT_DECOD(6),R7);
     oSUB: and_1 PORT MAP(RSU,PORT_DECOD(7),R8);
     -------------SAIDA COUT-------------------------
     sCOUT: or_1 PORT MAP(OSO, OSU, COUT);
     ------------SAIDA 1 BIT-------------------------
     sULA: or_2 PORT MAP(R1, R2, R3, R4, R5, R6, R7, R8, S);
END des;
```

Resultados

Os resultados deste projeto são descritos através das telas de simulação do software Quartus II que processa a linguagem VHDL. Para os pacotes que possuem todos os componentes da utilizados na ULA de 1 Bit teremos uma tela de simulação e para as ligações dos componentes da ULA teremos outra.



Juntamente às telas de simulação o resultado do projeto pode ser descrito por meio de formas de onda, que representam o funcionamento do circuito em questão.