

IMPLEMENTAÇÕES

Unidade I

DCA0212.1 - Circuitos Digitais

Componentes:

- IGOR SERGIO DE FRANCA CORREIA
- NEUMAN FABRICIO DE OLIVEIRA FERNANDES
- SARAH CAVALCANTE BRAZIL SILVA
- THIAGO THEIRY DE OLIVEIRA

LAB 01

PORTAS LÓGICAS

- 1.** Desenvolva a porta OR em VHDL e simule utilizando o Quartus;
- 2.** Faça o teste da porta AND em VHDL e simule utilizando o Quartus;
- 3.** Desenvolva um código para que seja possível testar TODAS as propriedades, postulados e lei de De Morgan mostradas na primeira seção deste roteiro.

I. Porta OR

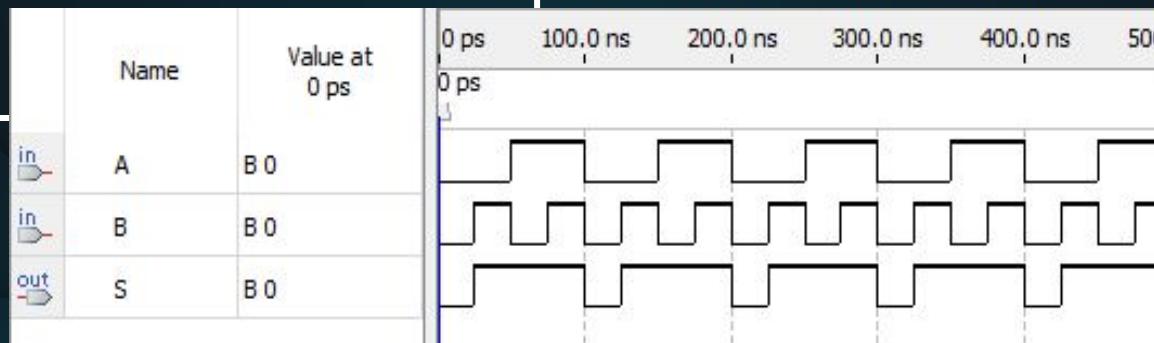
```
ENTITY PortaOr IS
    PORT(A, B: IN BIT;
        S : OUT BIT);
END PortaOr;

ARCHITECTURE behav of PortaOr IS
BEGIN
    S <= A OR B;
END behav;
```

Código VHDL

A	B	$(A \vee B)$
F	F	F
F	T	T
T	F	T
T	T	T

Tabela verdade



Simulação

2. Porta AND

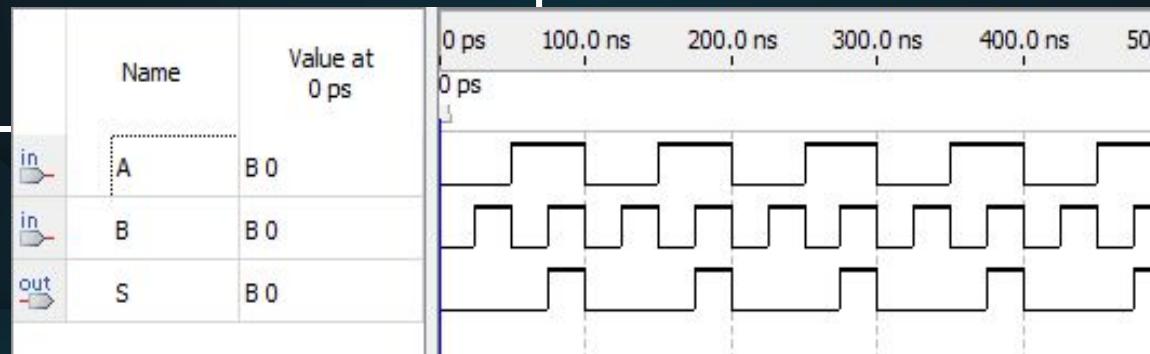
```
ENTITY PortaAnd IS
    PORT(A, B: IN BIT;
        S : OUT BIT);
END PortaAnd;

ARCHITECTURE behav of PortaAnd IS
BEGIN
    S <= A AND B;
END behav;
```

Código VHDL

A	B	$(A \wedge B)$
F	F	F
F	T	F
T	F	F
T	T	T

Tabela verdade



Simulação

3. Propriedades, Postulados e Leis de De Morgan

```
ENTITY TesteDePropriedades IS
  PORT(A, B, C: IN BIT;
        Si1, Si2, Si3, ← Saídas dos postulados de identidade
        Sc1, Sc2, Sc3, Sc4, ← Saídas das propriedades comutativas
        Sa1, Sa2, Sa3, Sa4, ← Saídas das propriedades associativas
        Sd1, Sd2, Sd3, Sd4, ← Saídas das propriedades distributivas
        Sm1, Sm2, Sm3, Sm4: OUT BIT); ← Saídas das Leis de De Morgan
END TesteDePropriedades;
```

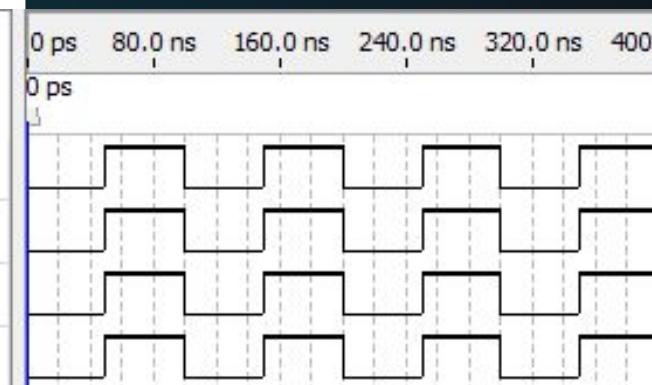
```
-- Identidade
ARCHITECTURE identidade OF TesteDePropriedades IS
BEGIN
  Si1 <= A OR '0';
  Si2 <= A AND '1';
  Si3 <= NOT (NOT A);
END identidade;
```

Código VHDL

	Name	Value at 0 ps
in	A	B 0
out	Si1	B 0
out	Si2	B 0
out	Si3	B 0

1. $A + 0 = A$
2. $A \cdot 1 = A$
3. $\bar{\bar{A}} = A$

Postulados de identidade



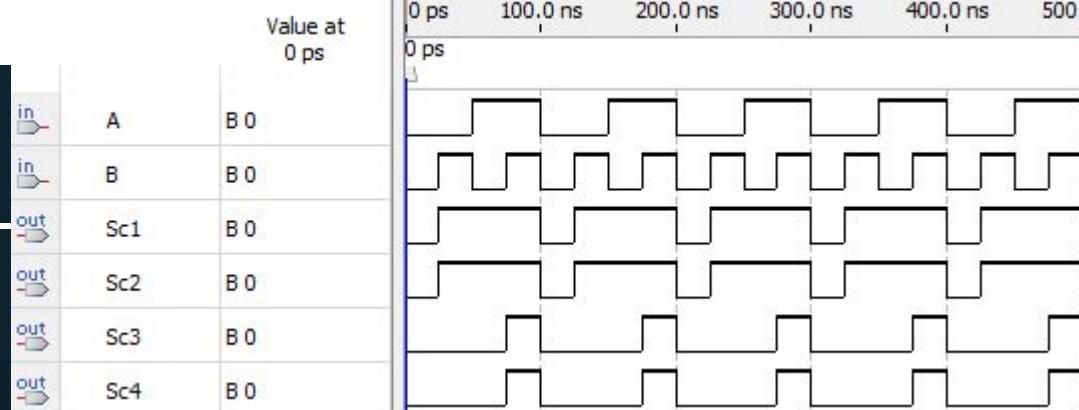
Simulação - Postulados de identidade

3. Propriedades, Postulados e Leis de De Morgan

```
-- Comutatividade
ARCHITECTURE comutatividade OF TesteDePropriedades IS
BEGIN
    -- Adição
    Sc1 <= A OR B;
    Sc2 <= B OR A;

    -- Multiplicação
    Sc3 <= A AND B;
    Sc4 <= B AND A;
END comutatividade;
```

Código VHDL



1. Adição: $A + B = B + A$;
2. Multiplicação: $A \cdot B = B \cdot A$

Propriedades comutativas

Simulação - Propriedades comutativas

3. Propriedades, Postulados e Leis de De Morgan

```
-- Associatividade
ARCHITECTURE associatividade OF TesteDePropriedades IS
BEGIN
    -- Adição
    Sa1 <= A OR (B OR C);
    Sa2 <= (A OR B) OR C;

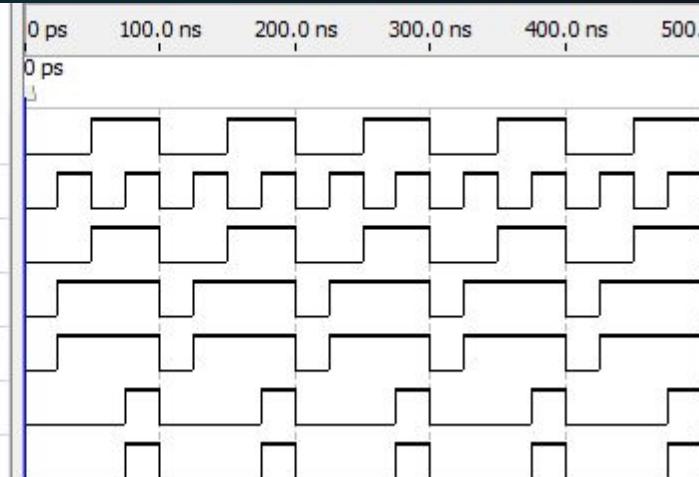
    -- Multiplicação
    Sa3 <= A AND (B AND C);
    Sa4 <= (A AND B) AND C;
END associatividade;
```

Código VHDL

	Value at 0 ps	
in	A	B 0
in	B	B 0
in	C	B 0
out	Sa1	B 0
out	Sa2	B 0
out	Sa3	B 0
out	Sa4	B 0

3. Adição: $A + (B + C) = (A + B) + C$;
4. Multiplicação: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

Propriedades associativas



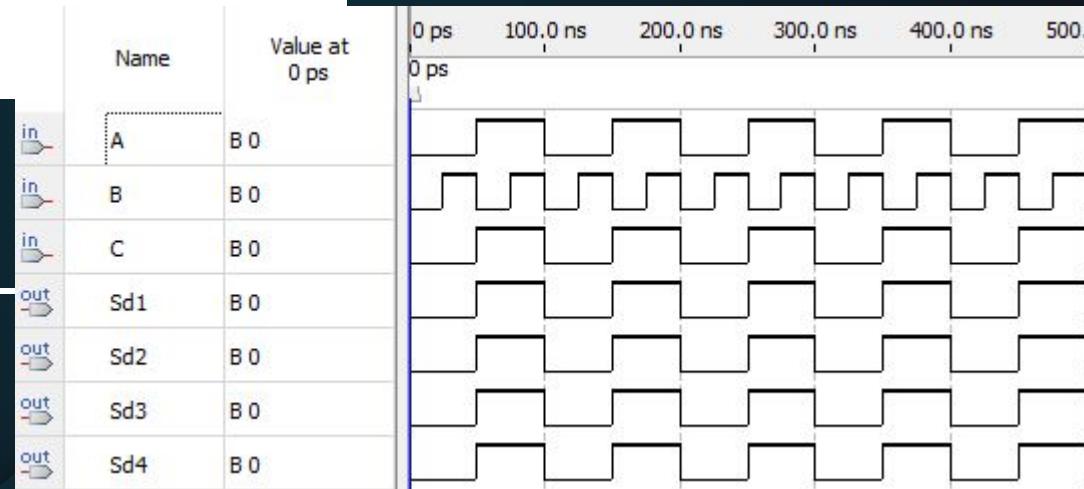
Simulação - Propriedades associativas

3. Propriedades, Postulados e Leis de De Morgan

```
-- Distributividade
ARCHITECTURE distributividade OF TesteDePropriedades IS
BEGIN
    Sd1 <= A AND (B OR C);
    Sd2 <= (A AND B) OR (A AND C);

    Sd3 <= A OR (B AND C);
    Sd4 <= (A OR B) AND (A OR C);
END distributividade;
```

Código VHDL



$$A \cdot (B + C) = A \cdot B + A \cdot C;$$
$$A + (B \cdot C) = (A + B) \cdot (A + C);$$

Propriedades distributivas

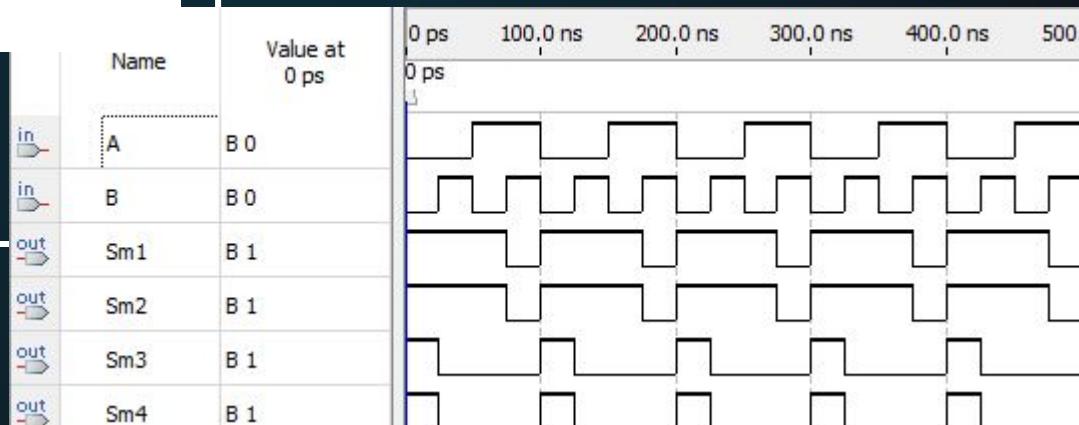
Simulação - Propriedades distributivas

3. Propriedades, Postulados e Leis de De Morgan

```
-- Leis de De Morgan
ARCHITECTURE DeMorgan of TesteDePropriedades IS
BEGIN
    Sm1 <= NOT (A AND B);
    Sm2 <= NOT (A) OR NOT (B);

    Sm3 <= NOT (A OR B);
    Sm4 <= NOT (A) AND NOT (B);
END DeMorgan;
```

Código VHDL



Simulação - Leis de De Morgan

- 1º Lei de De Morgan:

$$(\overline{A \cdot B}) = \overline{A} + \overline{B}$$

- 2º Lei de De Morgan

$$(\overline{A + B}) = \overline{A} \cdot \overline{B}$$

LAB 02

COMPONENTS

Um circuito que conta o número de 1s presente em três entradas a,b,c e, como saída, fornece esse número em binário, por meio de duas saídas S1 e S2.

1. Monte a tabela verdade do circuito, explicitando quais são as entradas e quais são as saídas e todas as possibilidades que o circuito lógico pode valer.
2. A equação do circuito é

$$S1 = A' \cdot B \cdot C + A \cdot B' \cdot C + A \cdot B \cdot C' + A \cdot B \cdot C$$

$$S2 = A' \cdot B' \cdot C + A' \cdot B \cdot C' + A \cdot B' \cdot C' + A \cdot B \cdot C$$

mostre qual a menor equação do circuito que pode ser obtido.

3. Represente os circuitos na forma de portas lógicas (o menor circuito possível).
4. Implemente o circuito utilizando o Quartus e VHDL. Para utilizar as portas lógicas, crie um projeto separado para cada porta e utilize o comando "COMPONENT" e "PORT MAP".

I. Tabela Verdade

Entradas			Saídas		Conversão das saídas	
A	B	C	S1	S2	S(S1S2)	S(Decimal)
0	0	0	0	0	00	0
0	0	1	0	1	01	1
0	1	0	0	1	01	1
1	0	0	0	1	01	1
0	1	1	1	0	10	2
1	1	0	1	0	10	2
1	0	1	1	0	10	2
1	1	1	1	1	11	3

2. Equações

É possível simplificar a equação S1 do circuito utilizando princípios da lógica Booleana da seguinte forma:

$$S1 = A'BC + AB'C + ABC' + ABC ;$$

$S1 = A(BC + B'C + BC')$ + $A'BC$ (Pela primeira propriedade distributiva);

$S1 = A(BC + BC' + B'C)$ + $A'BC$ (Pela propriedade comutativa);

$S1 = A(B(C + C')) + B'C$ + $A'BC$ (Pela comutatividade e distributiva);

$S1 = A(B(1) + B'C)$ + $A'BC$ (Pela propriedade do complemento);

$S1 = A(B + B'C)$ + $A'BC$ (Pela propriedade da identidade);

$S1 = A((B+B')(B+C)) + A'BC$ (Pela segunda propriedade distributiva);

$S1 = A((1)(B+C)) + A'BC$ (Pela propriedade do complemento);

$S1 = A(B+C)$ + $A'BC$ (Pela propriedade da identidade);

$S1 = AB + AC + A'BC$ (Pela distribuição);

$S1 = AC + B(A + A'C)$ (Pela primeira propriedade distributiva);

$S1 = AC + B(A + C)$ (Pela lei da absorção);

$S1 = AC + BA + BC$ (Pela distribuição);

Já a equação S2 já está na forma mais simplificada possível:

$$S2 = A'B'C + A'BC' + AB'C' + ABC ;$$

2- and 3-variable Boolean Algebra Theorems

Commutative Laws	(1a) (1b)	$x+y=y+x$ $x \cdot y = y \cdot x$
Associative Laws	(2a) (2b)	$x + (y+z) = (x+y) + z$ $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Distributive Laws	(3a) (3b)	$x + (y \cdot z) = (x+y) \cdot (x+z)$ $x \cdot (y+z) = (x \cdot y) + (x \cdot z)$
Unity	(4a) (4b)	$(x \cdot y) + (x' \cdot y) = y$ $(x+y) \cdot (x'+y) = y$
Absorption	(5a) (5b)	$x + (x \cdot y) = x$ $x \cdot (x+y) = x$
	(6a) (6b)	$x + (x' \cdot y) = x+y$ $x \cdot (x'+y) = x \cdot y$
DeMorgan's		$(x+y)' = x' \cdot y'$ $(xy)' = x' + y'$

1) Associatividade das operações OU e E.

$$(X + Y) + Z = X + (Y + Z)$$

$$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$$

2) Comutatividade das operações OU e E.

$$X + Y = Y + X$$

$$X \cdot Y = Y \cdot X$$

3) Elemento unitário para a operação OU (dígito 0).

$$0 + X = X$$

4) Elemento unitário para a operação E (dígito 1).

$$1 \cdot X = X$$

5) Distributividade de E sobre a operação OU.

$$X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$$

$$(W + X) \cdot (Y + Z) = W \cdot Y + X \cdot Y + W \cdot Z + X \cdot Z$$

6) Distributividade de OU sobre a operação E.

$$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$$

7) Existência de um elemento complemento.

$$X \cdot \bar{X} = 0$$

$$X + \bar{X} = 1$$

8) Lei da Absorção.

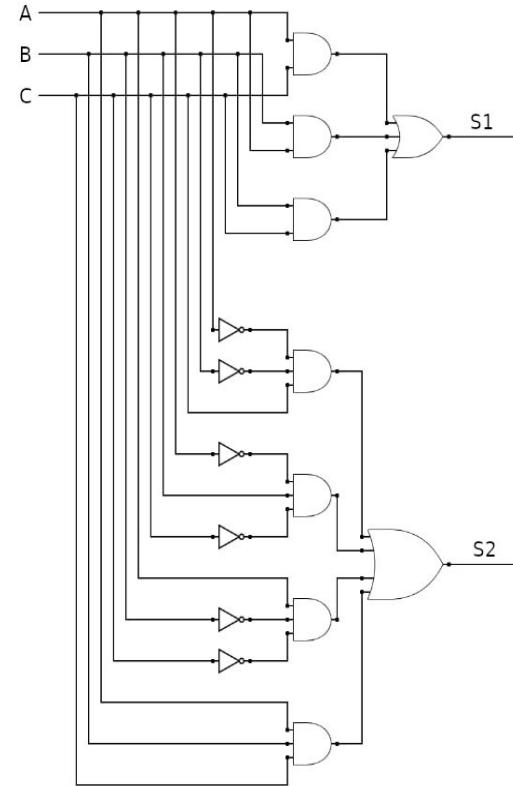
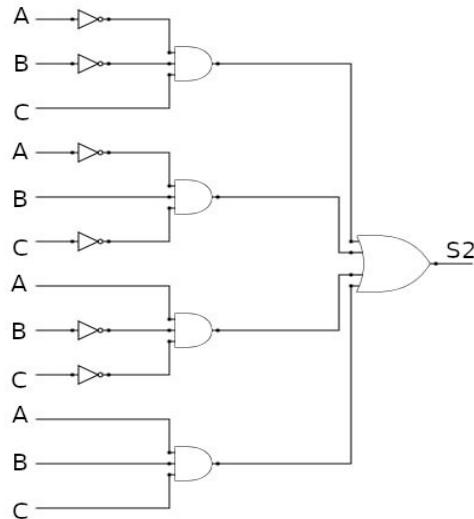
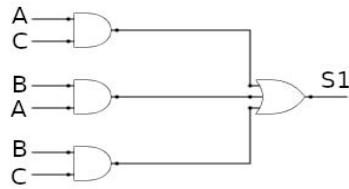
$$X + X \cdot Y = X$$

$$X + \bar{X} \cdot Y = X + Y$$

9) Adição lógica.

$$X + 1 = 1$$

3.Circuitos - Portas lógicas



4. QUARTUS + VHDL

Projetos - placas lógicas

```
1  ENTITY PortaAnd IS
2  PORT(entrada_and1,entrada_and2: IN BIT;
3      saída_and: OUT BIT);
4
5  END PortaAnd;
6
7
8  ARCHITECTURE behav OF PortaAnd IS
9  BEGIN
10    saída_and <= entrada_and1 AND entrada_and2;
11  END;
```

Porta AND - 2 inputs

```
1  ENTITY PortaAnd2 IS
2  PORT(entrada2_and1,entrada2_and2,entrada2_and3: IN BIT;
3      saída_and2: OUT BIT);
4
5  END PortaAnd2;
6
7
8  ARCHITECTURE behav OF PortaAnd2 IS
9  BEGIN
10    saída_and2 <= entrada2_and1 AND entrada2_and2 AND entrada2_and3;
11  END;
```

Porta AND - 3 inputs

```
1  ENTITY PortaOr IS
2  PORT(input1,input2,input3: IN BIT;
3      saída_or: OUT BIT);
4
5  END PortaOr;
6
7
8  ARCHITECTURE behav OF PortaOr IS
9  BEGIN
10    saída_or <= input1 or input2 or input3;
11  END;
```

Porta OR - 3 inputs

```
1  ENTITY PortaOr2 IS
2  PORT(cin1,cin2,cin3,cin4: IN BIT;
3      saída_or2: OUT BIT);
4
5  END PortaOr2;
6
7
8  ARCHITECTURE behav OF PortaOr2 IS
9  BEGIN
10    saída_or2 <= cin1 or cin2 or cin3 or cin4;
11  END;
```

Porta OR - 4 inputs

```
1  ENTITY PortaNot IS
2  PORT(entrada_not: IN BIT;
3      saída_not: OUT BIT);
4
5  END PortaNot;
6
7
8  ARCHITECTURE behav OF PortaNot IS
9  BEGIN
10    saída_not <= NOT(entrada_not);
11  END;
```

Porta NOT

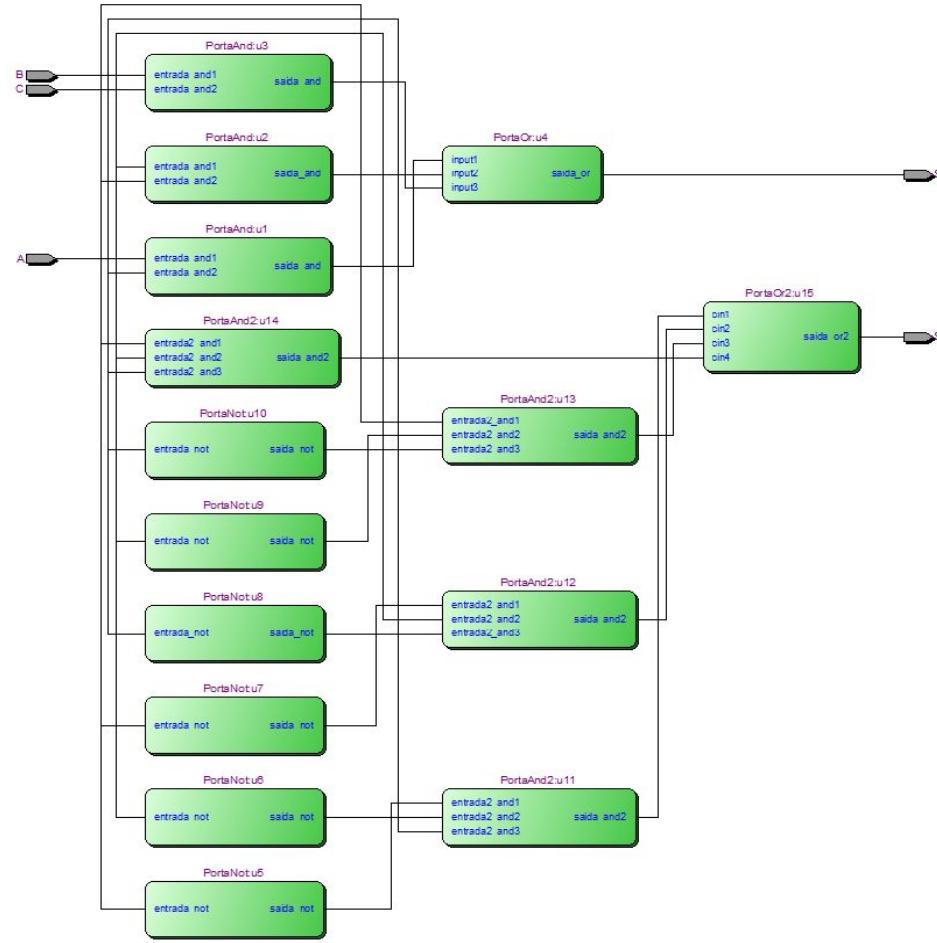
Circuito - SIGNAL & COMPONENT

```
1  ENTITY circuito IS
2  PORT (A,B,C:IN BIT;
3        S1,S2: OUT BIT);
4
5  END circuito;
6
7  ARCHITECTURE behav OF circuito IS
8
9    SIGNAL AND1:BIT;
10   SIGNAL AND2:BIT;
11   SIGNAL AND3:BIT;
12   SIGNAL AND4:BIT;
13   SIGNAL AND5:BIT;
14   SIGNAL AND6:BIT;
15   SIGNAL AND7:BIT;
16   SIGNAL NOT1:BIT;
17   SIGNAL NOT2:BIT;
18   SIGNAL NOT3:BIT;
19   SIGNAL NOT4:BIT;
20   SIGNAL NOT5:BIT;
21   SIGNAL NOT6:BIT;
22
23  COMPONENT PortaOr IS
24    PORT(input1,input2,input3: IN BIT;
25        saida_or: OUT BIT);
26  END COMPONENT;
27
28  COMPONENT PortaOr2 IS
29    PORT(cin1,cin2,cin3,cin4: IN BIT;
30        saida_or2: OUT BIT);
31  END COMPONENT;
32
33  COMPONENT PortaNot IS
34    PORT(entrada_not: IN BIT;
35        saida_not: OUT BIT);
36  END COMPONENT;
37
38  COMPONENT PortaAnd IS
39    PORT(entrada_and1,entrada_and2: IN BIT;
40        saida_and1: OUT BIT);
41  END COMPONENT;
42
43  COMPONENT PortaAnd2 IS
44    PORT(entrada2_and1,entrada2_and2,entrada2_and3: IN BIT;
45        saida_and2: OUT BIT);
46  END COMPONENT;
47
```

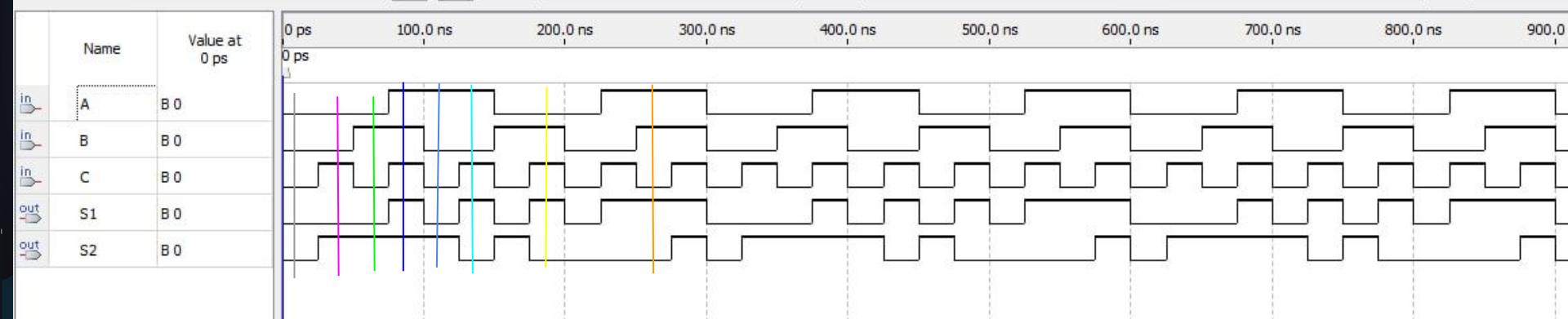
Circuito - PORT MAP

```
47
48 BEGIN
49 u1: PortaAnd PORT MAP(entrada_and1 => A, entrada_and2 => C, saida_and => AND1);
50 u2: PortaAnd PORT MAP(entrada_and1 => B, entrada_and2 => A, saida_and => AND2);
51 u3: PortaAnd PORT MAP(entrada_and1 => B, entrada_and2 => C, saida_and => AND3);
52 u4: PortaOr PORT MAP (input1 => AND1, input2 => AND2, input3 => AND3, saida_or => S1);
53 u5: PortaNot PORT MAP(entrada_not => A, saida_not => NOT1);
54 u6: PortaNot PORT MAP(entrada_not => B, saida_not => NOT2);
55 u7: PortaNot PORT MAP(entrada_not => A, saida_not => NOT3);
56 u8: PortaNot PORT MAP(entrada_not => C, saida_not => NOT4);
57 u9: PortaNot PORT MAP(entrada_not => B, saida_not => NOT5);
58 u10: PortaNot PORT MAP(entrada_not => C, saida_not => NOT6);
59 u11: PortaAnd2 PORT MAP(entrada2_and1 => NOT1, entrada2_and2 => NOT2, entrada2_and3 => C, saida_and2 =>AND4);
60 u12: PortaAnd2 PORT MAP(entrada2_and1 => NOT3, entrada2_and2 => B, entrada2_and3 => NOT4, saida_and2 =>AND5);
61 u13: PortaAnd2 PORT MAP(entrada2_and1 => A, entrada2_and2 => NOT5, entrada2_and3 => NOT6, saida_and2 =>AND6);
62 u14: PortaAnd2 PORT MAP(entrada2_and1 => A, entrada2_and2 => B, entrada2_and3 => C, saida_and2 =>AND7);
63 u15: PortaOr2 PORT MAP(cin1 => AND4, cin2 => AND5, cin3 => AND6, cin4 => AND7, saida_or2 => S2);
64 END;
```

RTL VIEWER



Simulação



Entradas			Saídas		Conversão das saídas	
A	B	C	S1	S2	S(S1S2)	S(Decimal)
0	0	0	0	0	00	0
0	0	1	0	1	01	1
0	1	0	0	1	01	1
1	0	0	0	1	01	1
0	1	1	1	0	10	2
1	1	0	1	0	10	2
1	0	1	1	0	10	2
1	1	1	1	1	11	3

LAB 03

MULTIPLEXADORES E DEMULTIPLEXADORES

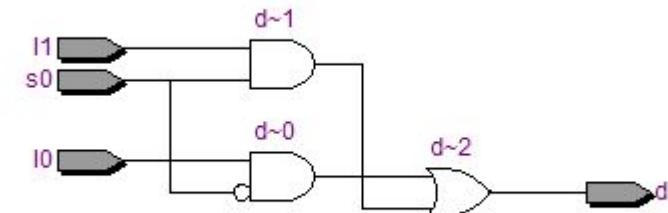
03

- 1.** Implemente em VHDL, utilizando circuitos lógicos, um Mux 2x1;
- 2.** Implemente em VHDL, utilizando a descrição comportamental, um Mux 2x1;
- 3.** Implemente em VHDL, utilizando a descrição comportamental, um Mux 4x1;
- 4.** Implemente em VHDL, utilizando a descrição comportamental + componentes, um mux 4x1 utilizando mux 2x1.
- 5.** Implementem em VHDL, utilizando a descrição comportamental + componentes, um Mux 8x1 utilizando os mux 4x1 implementados na questão anterior (juntamente com um mux 2x1).

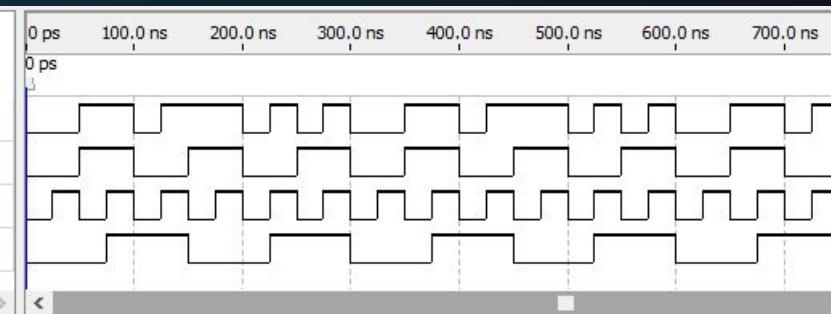
I. Mux 2x1 utilizando circuitos lógicos

```
ENTITY mux2x1 IS
PORT(I0,I1,s0 : IN BIT;
      d : OUT BIT);
END mux2x1;
ARCHITECTURE behav OF mux2x1 IS
BEGIN
  d <= (I0 and (not s0)) or (I1 and s0) ;
end behav;
```

Código VHDL



	Name	Value at 0 ps
	d	B 0
	I0	B 0
	I1	B 0
	s0	B 0

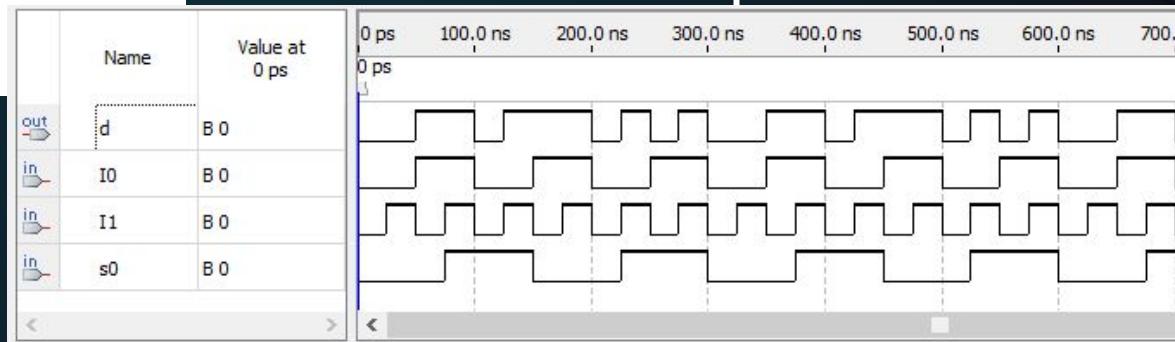


Simulação

2. Mux 2x1

```
ENTITY Mux2x1 IS
PORT(I0,I1,s0 : IN BIT;
      d : OUT BIT);
END Mux2x1;
ARCHITECTURE behav OF Mux2x1 IS
BEGIN
  WITH s0 SELECT
    d <= I0 WHEN '0',
    I1 WHEN '1';
END behav;
```

Código VHDL



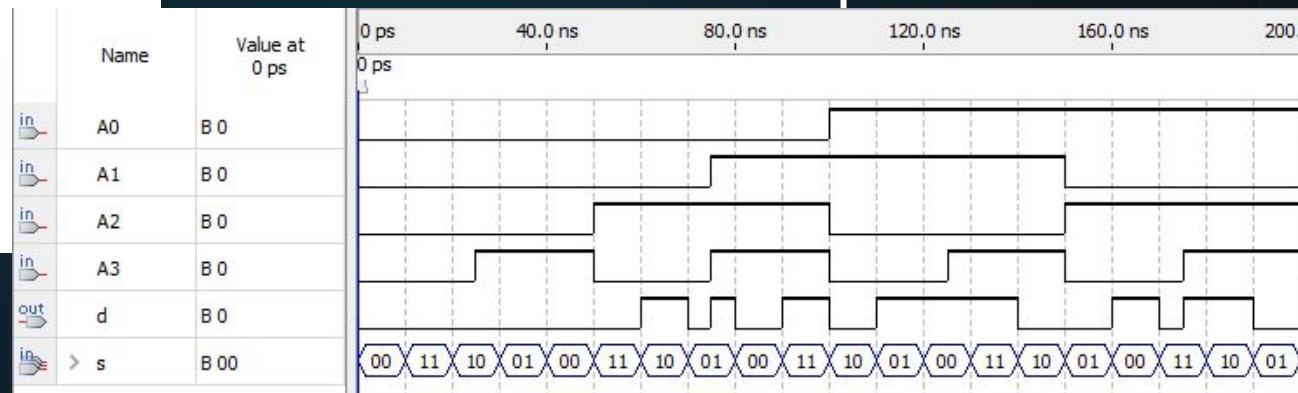
Simulação

3. Mux 4x1

```
ENTITY Mux4x1 IS
PORT(A0,A1,A2,A3 : IN BIT;
s : IN BIT_VECTOR(1 DOWNTO 0);
d : OUT BIT);
END;

ARCHITECTURE behav OF Mux4x1 IS
BEGIN
WITH s SELECT
d <= A0 WHEN '0' & '0',
A1 WHEN '0' & '1',
A2 WHEN '1' & '0',
A3 WHEN '1' & '1';
END;
```

Código VHDL



Simulação

4. Mux 4x1 utilizando Mux 2x1

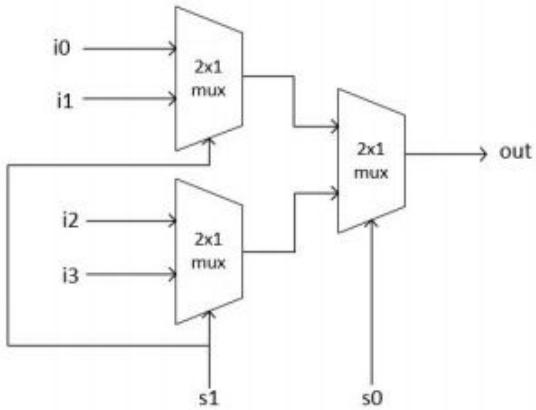


Figura 4: Multiplexador 4x1 utilizando multiplexadores 2x1

```
ENTITY Mux4x1 IS
PORT(I0,I1,I2,I3,s0,s1 : IN BIT;
      d : OUT BIT);
END;

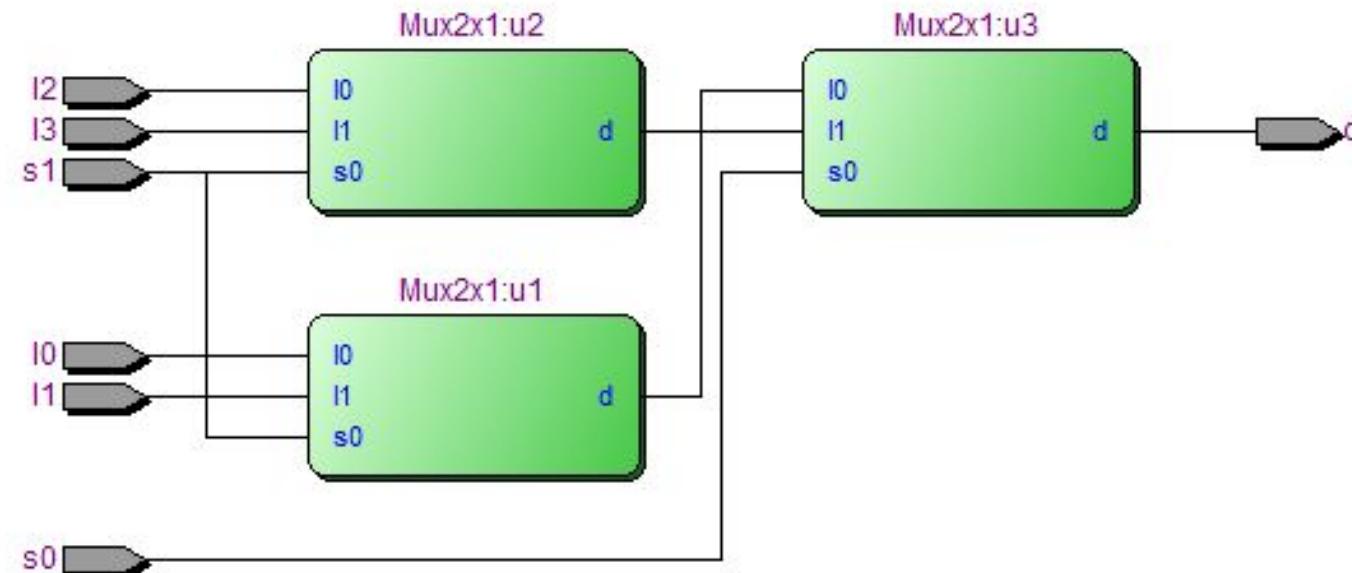
ARCHITECTURE behav OF Mux4x1 IS
SIGNAL M1: BIT;
SIGNAL M2: BIT;

COMPONENT Mux2x1 IS
PORT(I0,I1,s0 : IN BIT;
      d : OUT BIT);
END COMPONENT;

BEGIN
u1: Mux2x1 PORT MAP(I0 => I0, I1 => I1, s0 => s1, d => M1);
u2: Mux2x1 PORT MAP(I0 => I2, I1 => I3, s0 => s1, d => M2);
u3: Mux2x1 PORT MAP(I0 => M1, I1 => M2, s0 => s0, d => d);
END;
```

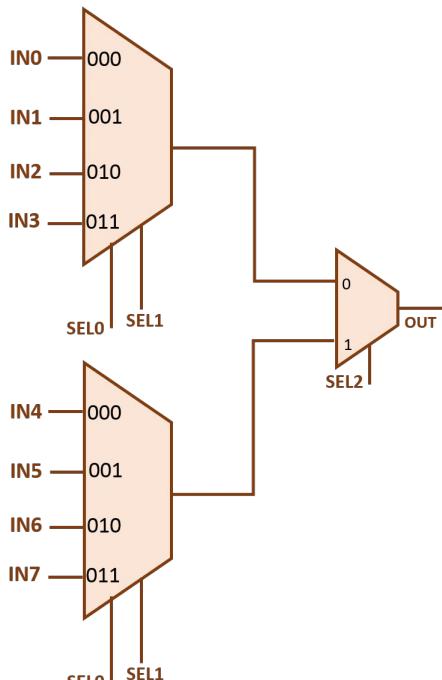
Código VHDL

4. Mux 4x1 utilizando Mux 2x1



RTL Viewer

5. Mux 8x1 utilizando Mux 4x1 e Mux 2x1



```
ENTITY Mux8x1 IS
PORT(I0,I1,I2,I3,I4,I5,I6,I7,s0,s1,s2 : IN BIT;
      d : OUT BIT);
END;

ARCHITECTURE behav OF Mux8x1 IS
  SIGNAL M1: BIT;
  SIGNAL M2: BIT;

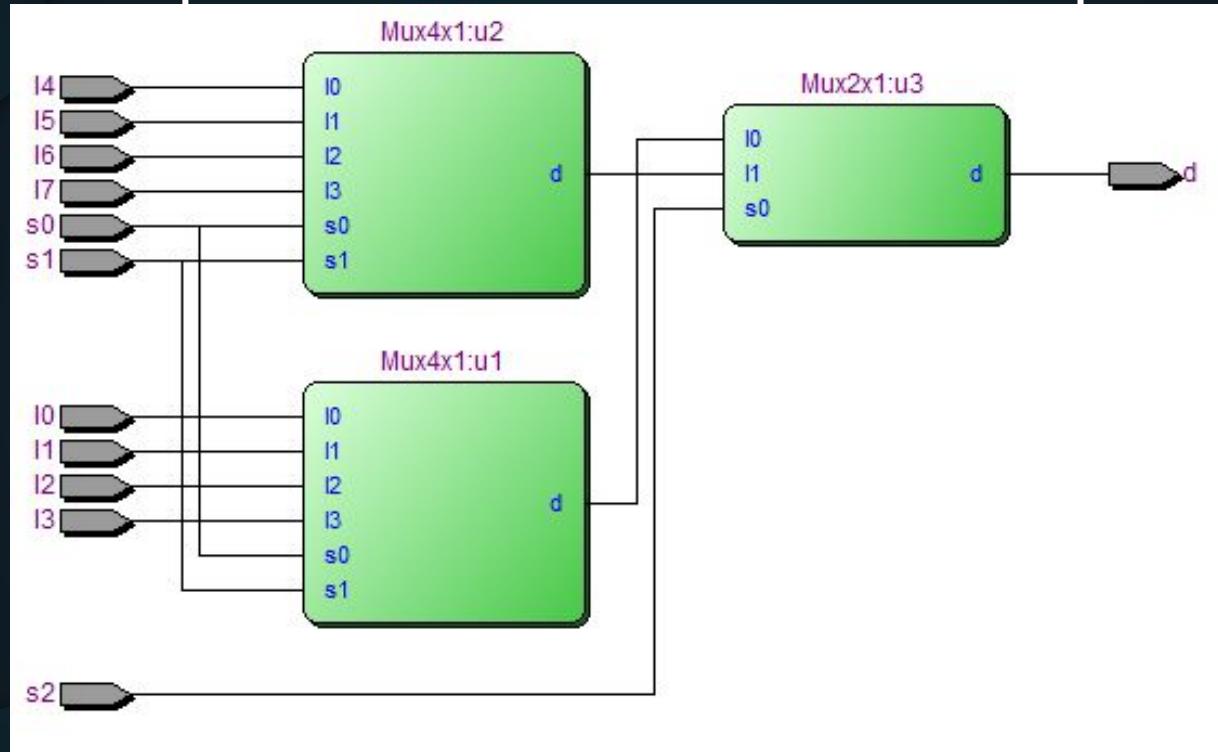
COMPONENT Mux2x1 IS
PORT(I0,I1,s0 : IN BIT;
      d : OUT BIT);
END COMPONENT;

COMPONENT Mux4x1 IS
PORT(I0,I1,I2,I3,s0,s1 : IN BIT;
      d : OUT BIT);
END COMPONENT;

BEGIN
  u1: Mux4x1 PORT MAP(I0 => I0, I1 => I1, I2 => I2, I3 => I3, s0 => s0, s1 => s1, d => M1);
  u2: Mux4x1 PORT MAP(I0 => I4, I1 => I5, I2 => I6, I3 => I7, s0 => s0, s1 => s1, d => M2);
  u3: Mux2x1 PORT MAP(I0 => M1, I1 => M2, s0 => s2, d => d);
END;
```

Código VHDL

5. Mux 8x1 utilizando Mux 4x1 e Mux 2x1



RTL Viewer

Componentes:

- IGOR SERGIO DE FRANCA CORREIA
- NEUMAN FABRICIO DE OLIVEIRA FERNANDES
- SARAH CAVALCANTE BRAZIL SILVA
- THIAGO THEIRY DE OLIVEIRA

Obrigado pela atenção!