Laboratórios

Circuitos Digitais - I Unidades

Gabriel Nogueira da Silva Dantas - 20200149499

Jonas Peixoto da silva - 20170116140

Matheus Felipe Souto de Alcântara - 2016019731

Yuri da Silva Furtado - 20180030175

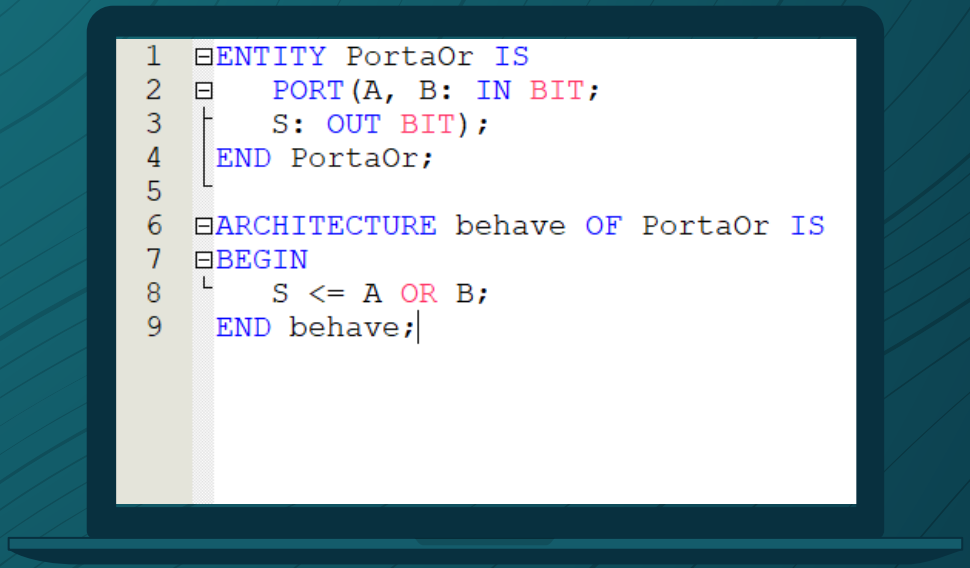
Laboratório 01

CIRCUITOS LÓGICOS

Objetivos:

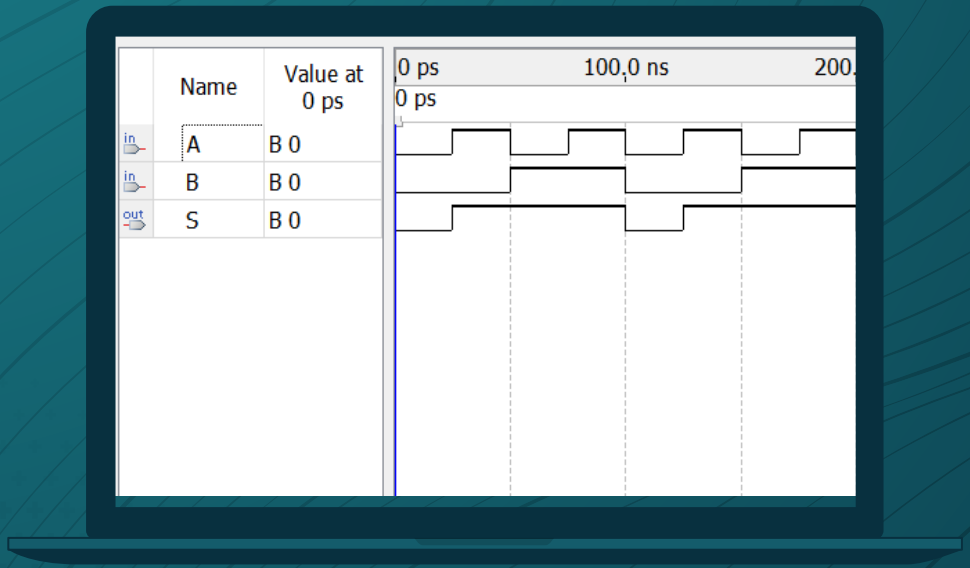
- Treinar os conceitos de portas lógicas, circuitos lógicos, funções booleanas e tabelas da verdade;
- Provar algumas das propriedades da álgebra booleana;
- Treinar a programação em VHDL de entidades básicas.

Desenvolvimento da Porta 'OR'



```
1  ENTITY PortaOr IS
2     PORT(A, B: IN BIT;
3     S: OUT BIT);
4  END PortaOr;
5
6  ARCHITECTURE behave OF PortaOr IS
7  BEGIN
8     S <= A OR B;
9  END behave;
```

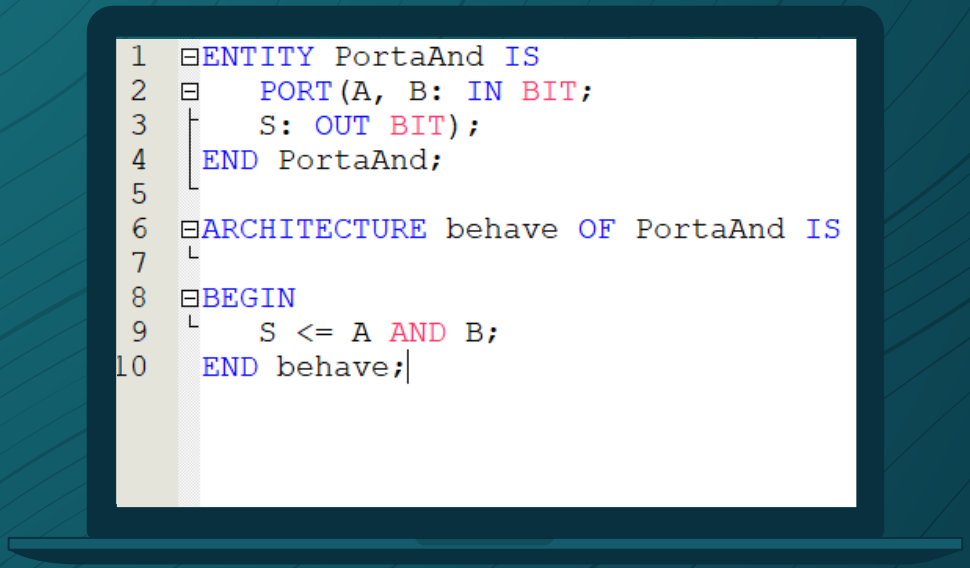
Simulação da Porta 'OR'



A	B	S = A + B
0	0	0
0	1	1
1	0	1
1	1	1

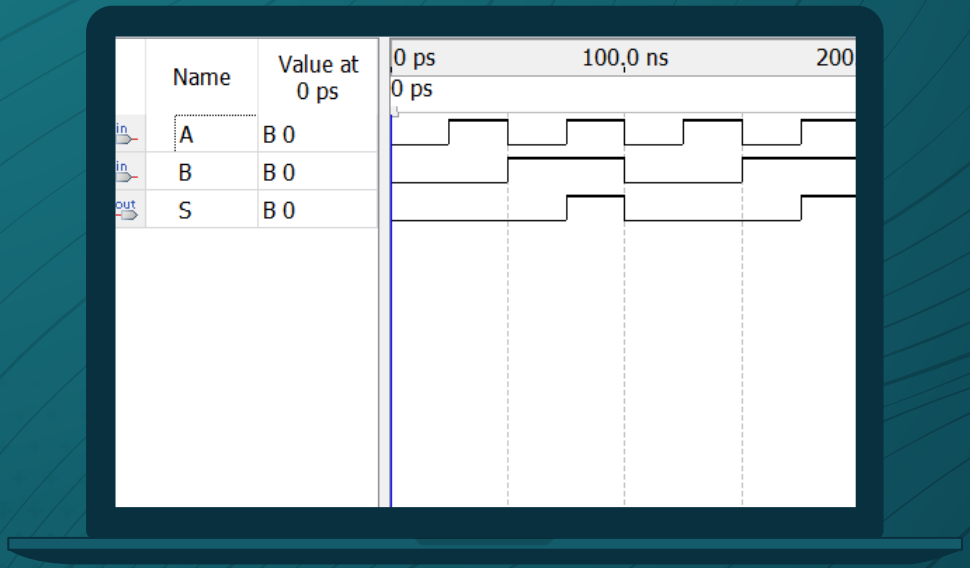
Tabela 1: Tabela Verdade da Porta OR

Desenvolvimento da Porta 'AND'



```
1  ENTITY PortaAnd IS
2      PORT(A, B: IN BIT;
3      S: OUT BIT);
4  END PortaAnd;
5
6  ARCHITECTURE behave OF PortaAnd IS
7
8  BEGIN
9      S <= A AND B;
10 END behave;
```

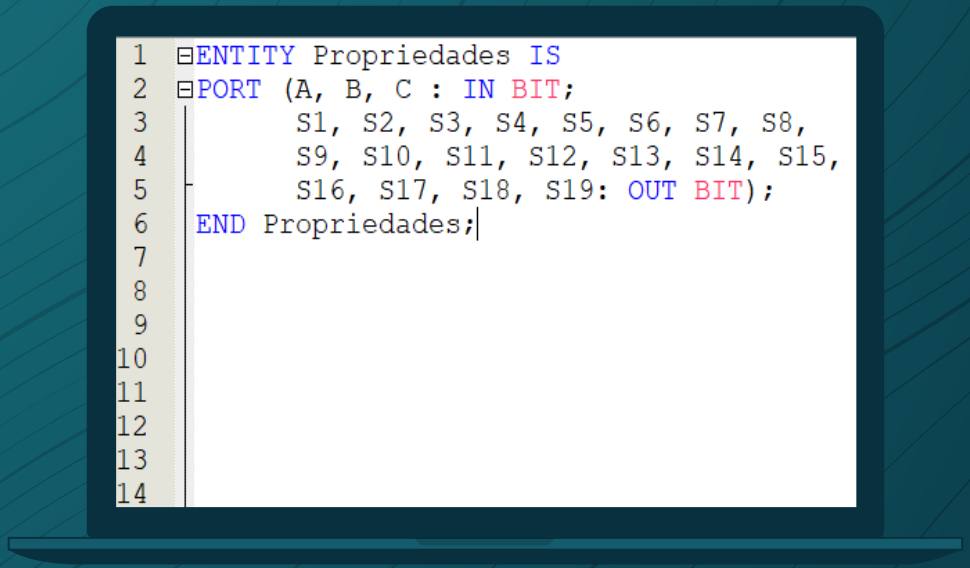
Simulação da Porta 'AND'



A	B	S = A · B
0	0	0
0	1	0
1	0	0
1	1	1

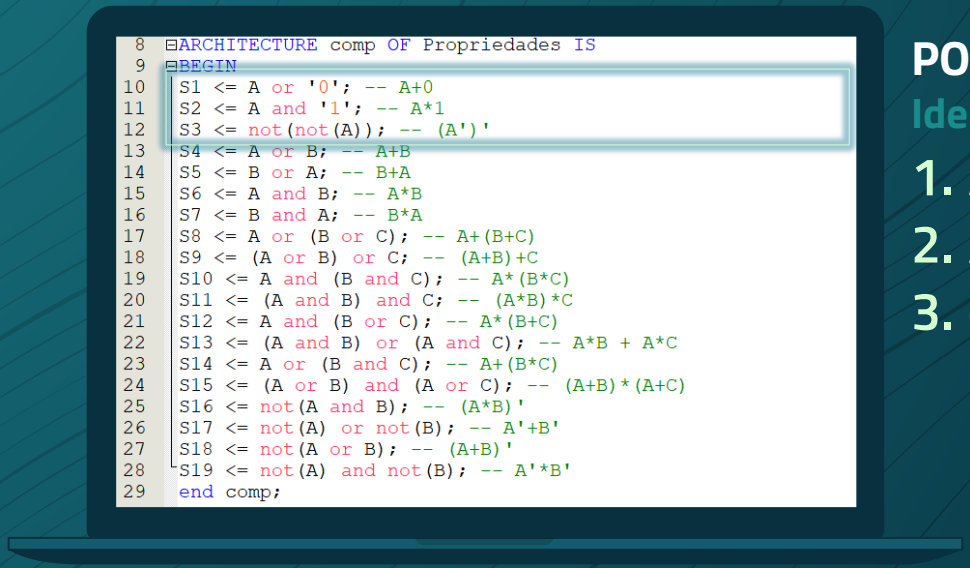
Tabela 1: Tabela Verdade da Porta OR

Desenvolvimento dos testes **Entity**



```
1  ENTITY Propriedades IS
2  PORT (A, B, C : IN BIT;
3        S1, S2, S3, S4, S5, S6, S7, S8,
4        S9, S10, S11, S12, S13, S14, S15,
5        S16, S17, S18, S19: OUT BIT);
6  END Propriedades;
7
8
9
10
11
12
13
14
```

Desenvolvimento dos testes
Architecture



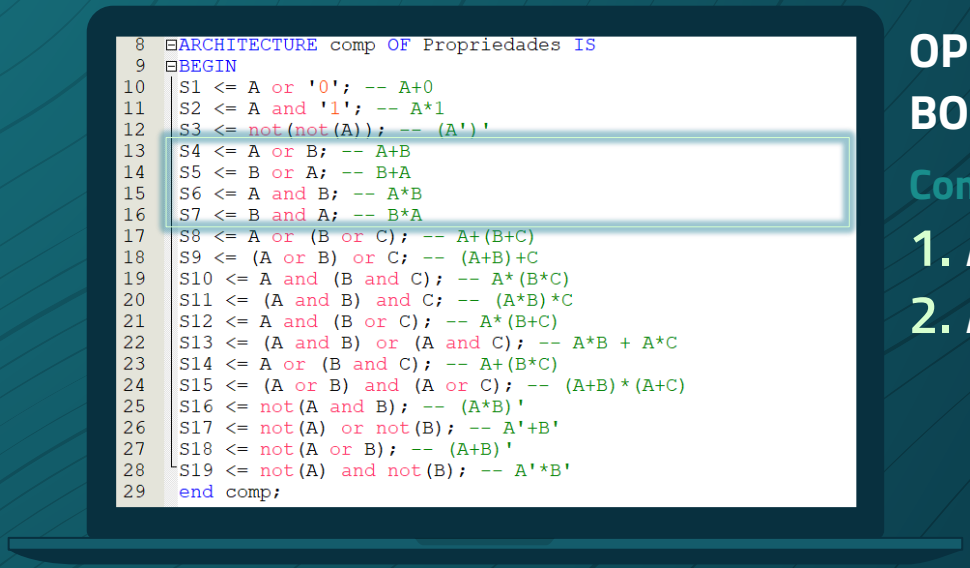
```
8  ARCHITECTURE comp OF Propriedades IS
9  BEGIN
10   S1 <= A or '0'; -- A+0
11   S2 <= A and '1'; -- A*1
12   S3 <= not(not(A)); -- (A')'
13   S4 <= A or B; -- A+B
14   S5 <= B or A; -- B+A
15   S6 <= A and B; -- A*B
16   S7 <= B and A; -- B*A
17   S8 <= A or (B or C); -- A+(B+C)
18   S9 <= (A or B) or C; -- (A+B)+C
19   S10 <= A and (B and C); -- A*(B*C)
20   S11 <= (A and B) and C; -- (A*B)*C
21   S12 <= A and (B or C); -- A*(B+C)
22   S13 <= (A and B) or (A and C); -- A*B + A*C
23   S14 <= A or (B and C); -- A+(B*C)
24   S15 <= (A or B) and (A or C); -- (A+B)*(A+C)
25   S16 <= not(A and B); -- (A*B)'
26   S17 <= not(A) or not(B); -- A'+B'
27   S18 <= not(A or B); -- (A+B)'
28   S19 <= not(A) and not(B); -- A'*B'
29 end comp;
```

POSTULADO

Identidade

1. A + 0 = A
2. A · 1 = A
3. (A')' = A

Desenvolvimento dos testes
Architecture



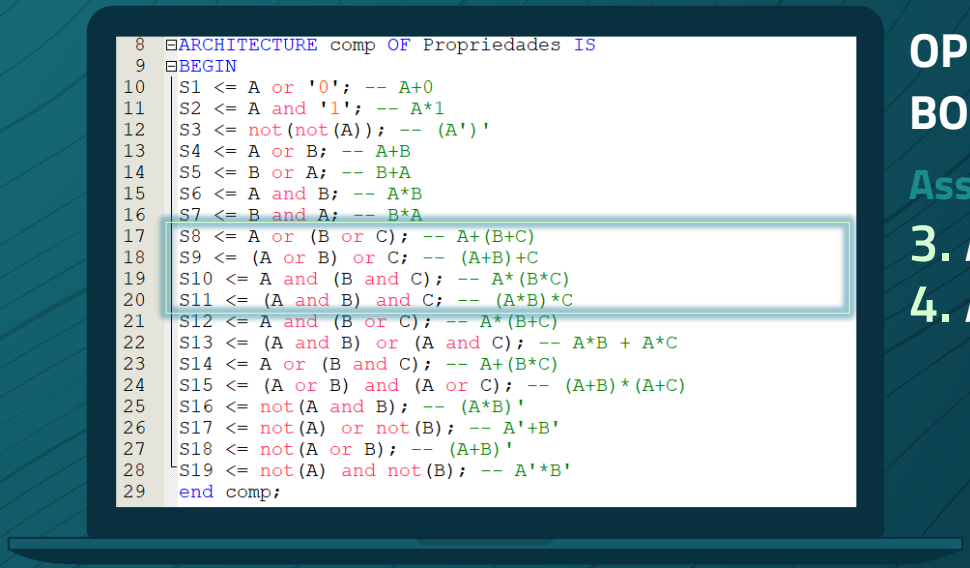
```
8  ARCHITECTURE comp OF Propriedades IS
9  BEGIN
10   S1 <= A or '0'; -- A+0
11   S2 <= A and '1'; -- A*1
12   S3 <= not(not(A)); -- (A')'
13   S4 <= A or B; -- A+B
14   S5 <= B or A; -- B+A
15   S6 <= A and B; -- A*B
16   S7 <= B and A; -- B*A
17   S8 <= A or (B or C); -- A+(B+C)
18   S9 <= (A or B) or C; -- (A+B)+C
19   S10 <= A and (B and C); -- A*(B*C)
20   S11 <= (A and B) and C; -- (A*B)*C
21   S12 <= A and (B or C); -- A*(B+C)
22   S13 <= (A and B) or (A and C); -- A*B + A*C
23   S14 <= A or (B and C); -- A+(B*C)
24   S15 <= (A or B) and (A or C); -- (A+B)*(A+C)
25   S16 <= not(A and B); -- (A*B)'
26   S17 <= not(A) or not(B); -- A'+B'
27   S18 <= not(A or B); -- (A+B)'
28   S19 <= not(A) and not(B); -- A'*B'
29 end comp;
```

OPERAÇÕES BOOLEANAS

Comutatividade

1. A + B = B + A;
2. A · B = B · A

Desenvolvimento dos testes
Architecture



```
8  ARCHITECTURE comp OF Propriedades IS
9  BEGIN
10   S1 <= A or '0'; -- A+0
11   S2 <= A and '1'; -- A*1
12   S3 <= not(not(A)); -- (A')'
13   S4 <= A or B; -- A+B
14   S5 <= B or A; -- B+A
15   S6 <= A and B; -- A*B
16   S7 <= B and A; -- B*A
17   S8 <= A or (B or C); -- A+(B+C)
18   S9 <= (A or B) or C; -- (A+B)+C
19   S10 <= A and (B and C); -- A*(B*C)
20   S11 <= (A and B) and C; -- (A*B)*C
21   S12 <= A and (B or C); -- A*(B+C)
22   S13 <= (A and B) or (A and C); -- A*B + A*C
23   S14 <= A or (B and C); -- A+(B*C)
24   S15 <= (A or B) and (A or C); -- (A+B)*(A+C)
25   S16 <= not(A and B); -- (A*B)'
26   S17 <= not(A) or not(B); -- A'+B'
27   S18 <= not(A or B); -- (A+B)'
28   S19 <= not(A) and not(B); -- A'*B'
29 end comp;
```

OPERAÇÕES BOOLEANAS

Associatividade

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

Desenvolvimento dos testes
Architecture

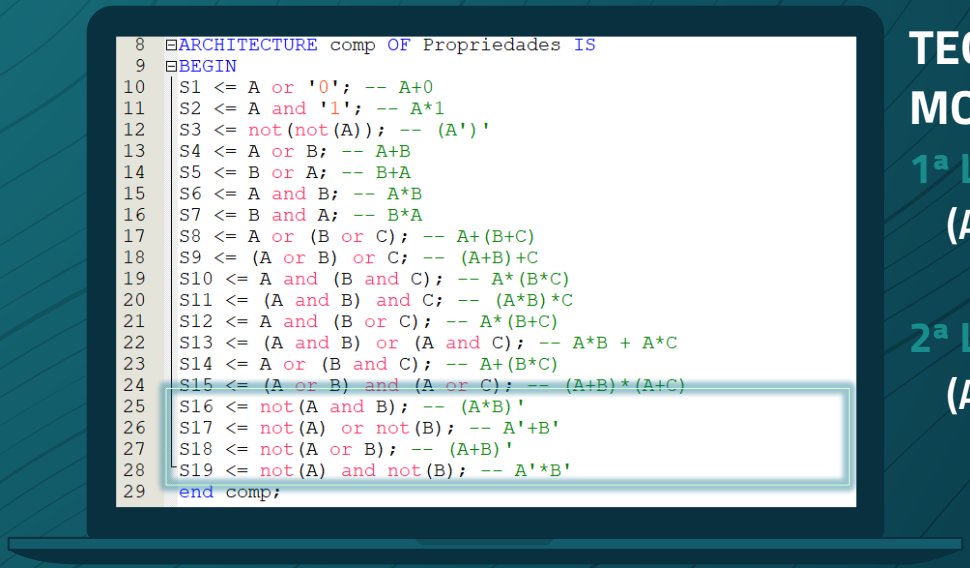
```
8  ARCHITECTURE comp OF Propriedades IS
9  BEGIN
10   S1 <= A or '0'; -- A+0
11   S2 <= A and '1'; -- A*1
12   S3 <= not(not(A)); -- (A')'
13   S4 <= A or B; -- A+B
14   S5 <= B or A; -- B+A
15   S6 <= A and B; -- A*B
16   S7 <= B and A; -- B*A
17   S8 <= A or (B or C); -- A+(B+C)
18   S9 <= (A or B) or C; -- (A+B)+C
19   S10 <= A and (B and C); -- A*(B*C)
20   S11 <= (A and B) and C; -- (A*B)*C
21   S12 <= A and (B or C); -- A*(B+C)
22   S13 <= (A and B) or (A and C); -- A*B + A*C
23   S14 <= A or (B and C); -- A+(B*C)
24   S15 <= (A or B) and (A or C); -- (A+B)*(A+C)
25   S16 <= not(A and B); -- (A*B)'
26   S17 <= not(A) or not(B); -- A'+B'
27   S18 <= not(A or B); -- (A+B)'
28   S19 <= not(A) and not(B); -- A'*B'
29 end comp;
```

OPERAÇÕES BOOLEANAS

Distributiva

5. $A \cdot (B + C) = A \cdot B + A \cdot C$
6. $A + (B \cdot C) = (A + B) \cdot (A + C)$

Desenvolvimento dos testes
Architecture



```
8  ARCHITECTURE comp OF Propriedades IS
9  BEGIN
10   S1 <= A or '0'; -- A+0
11   S2 <= A and '1'; -- A*1
12   S3 <= not(not(A)); -- (A')'
13   S4 <= A or B; -- A+B
14   S5 <= B or A; -- B+A
15   S6 <= A and B; -- A*B
16   S7 <= B and A; -- B*A
17   S8 <= A or (B or C); -- A+(B+C)
18   S9 <= (A or B) or C; -- (A+B)+C
19   S10 <= A and (B and C); -- A*(B*C)
20   S11 <= (A and B) and C; -- (A*B)*C
21   S12 <= A and (B or C); -- A*(B+C)
22   S13 <= (A and B) or (A and C); -- A*B + A*C
23   S14 <= A or (B and C); -- A+(B*C)
24   S15 <= (A or B) and (A or C); -- (A+B)*(A+C)
25   S16 <= not(A and B); -- (A*B)'
26   S17 <= not(A) or not(B); -- A'+B'
27   S18 <= not(A or B); -- (A+B)'
28   S19 <= not(A) and not(B); -- A'*B'
29 end comp;
```

TEOREMAS DE MORGAN

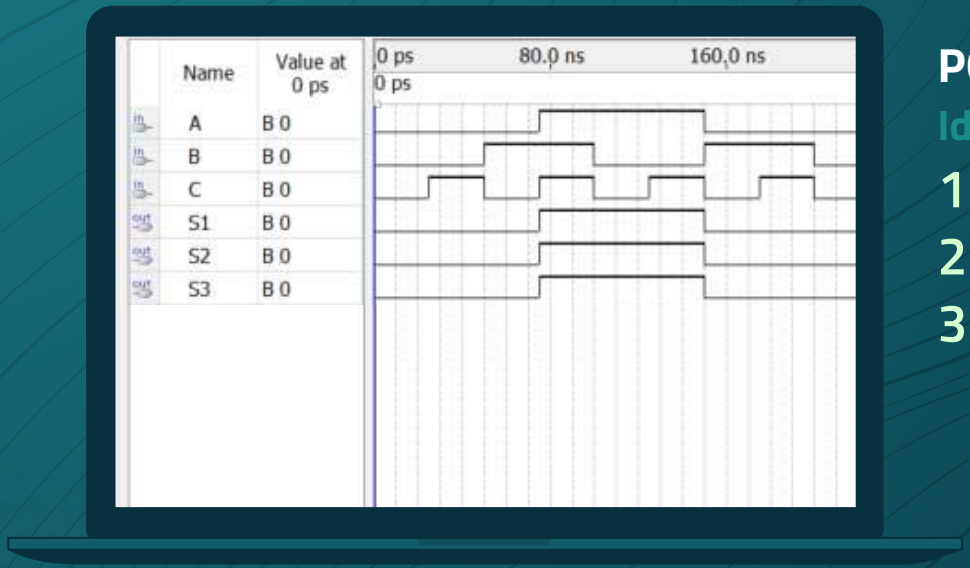
1ª Lei de Morgan

$(A \cdot B)' = A' + B'$

2ª Lei de Morgan

$(A + B)' = A' \cdot B'$

Simulação dos testes



POSTULADO

Identidade

1. A + 0 = A
2. A · 1 = A
3. (A')' = A

```
S1 <= A or '0';
S2 <= A and '1
S3 <= not[not[A]];
```

Simulação dos testes

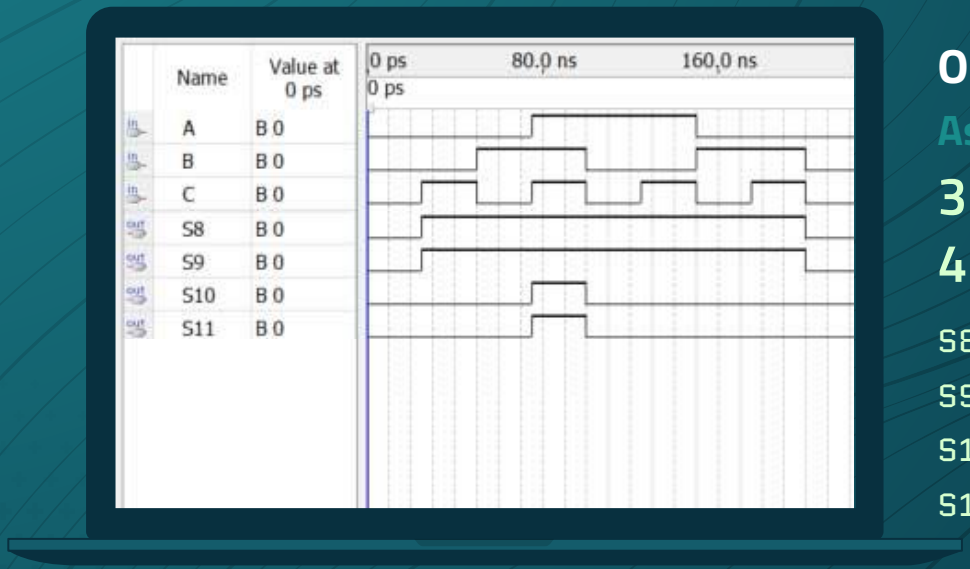
OPERAÇÕES BOOLEANAS

Comutatividade

1. A + B = B + A;
2. A · B = B · A

```
S4 <= A or B; -- A+B
S5 <= B or A; -- B+A
S6 <= A and B; -- A*B
S7 <= B and A; -- B*A
```

Simulação dos testes



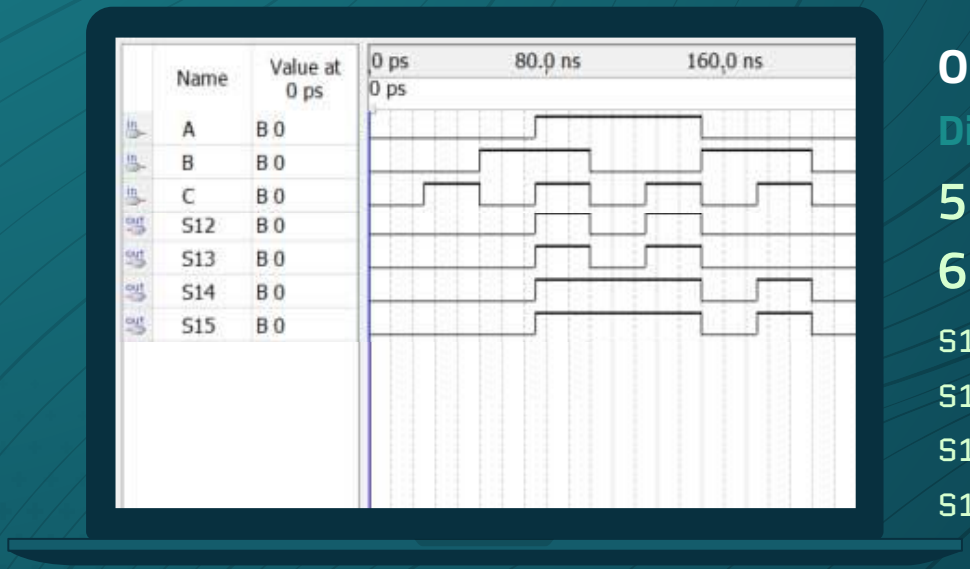
OPERAÇÕES BOOLEANAS

Associatividade

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

```
S8 <= A or (B or C); -- A+(B+C)
S9 <= (A or B) or C; -- (A+B)+C
S10 <= A and (B and C); -- A*(B*C)
S11 <= (A and B) and C; -- (A*B)*C
```

Simulação dos testes



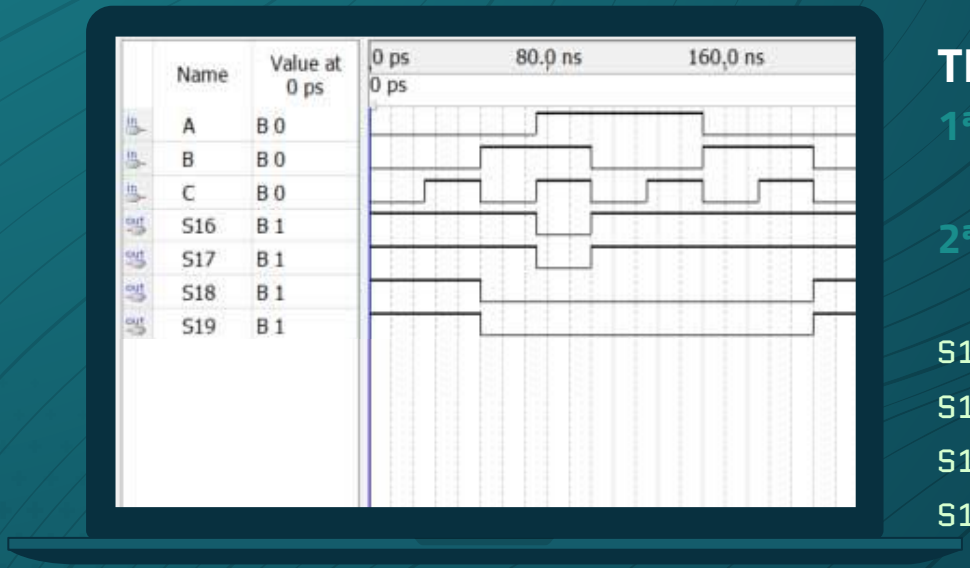
OPERAÇÕES BOOLEANAS

Distributiva

5. $A \cdot (B + C) = A \cdot B + A \cdot C$
6. $A + (B \cdot C) = (A + B) \cdot (A + C)$

```
S12 <= A and (B or C); -- A*(B+C)
S13 <= (A and B) or (A and C); -- A*B + A*C
S14 <= A or (B and C); -- A+(B*C)
S15 <= (A or B) and (A or C); -- (A+B)*(A+C)
```

Simulação dos testes



TEOREMAS DE MORGAN

1ª Lei de Morgan

$(A \cdot B)' = A' + B'$

2ª Lei de Morgan

$(A + B)' = A' \cdot B'$

```
S16 <= not(A and B); -- (A*B)'
S17 <= not(A) or not(B); -- A'+B'
S18 <= not(A or B); -- (A+B)'
S19 <= not(A) and not(B); -- A'*B'
```

Laboratório 02

CIRCUITOS LÓGICOS

Objetivos:

- Construir circuitos complexos a partir de portas lógicas simples;
- Testar a utilização do VHDL como ferramenta para criar problemas complexos a partir de subprojetos;
- Por em prática conceitos aprendidos na disciplina Circuitos Digitais - Teoria.

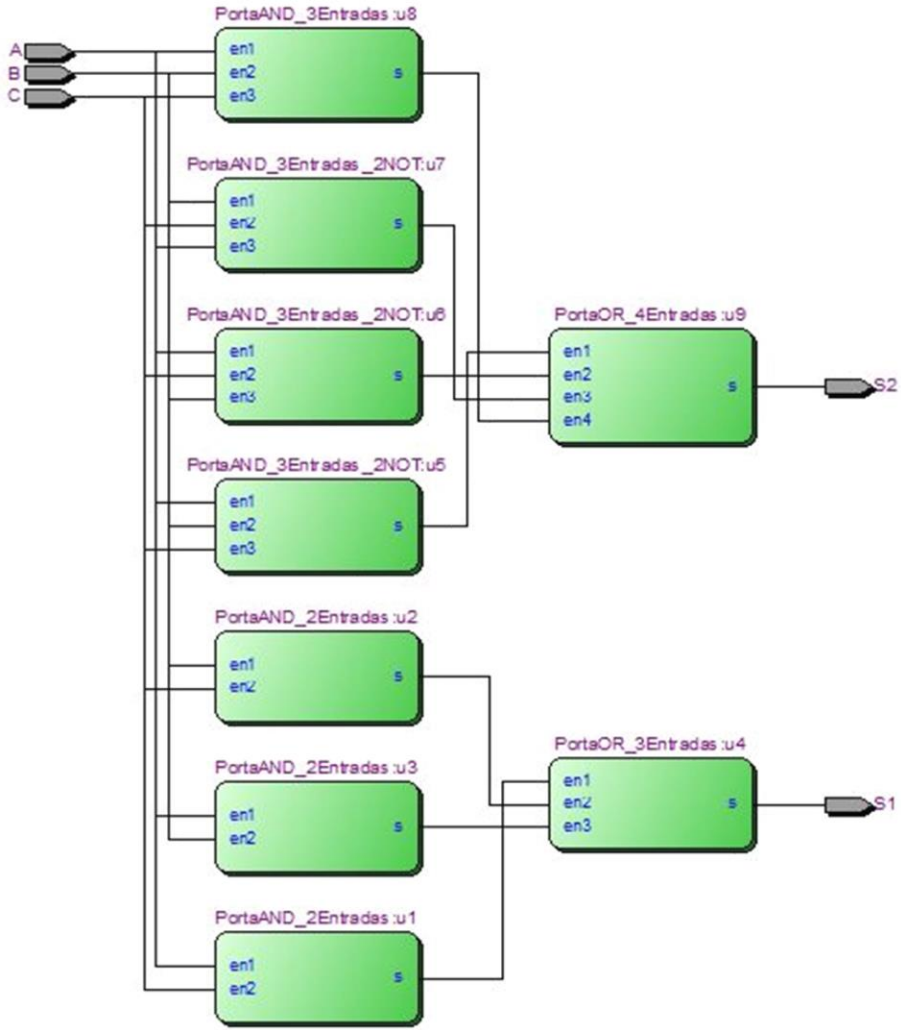
Tabela Verdade

A	B	C	S1	S2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Equação do circuito

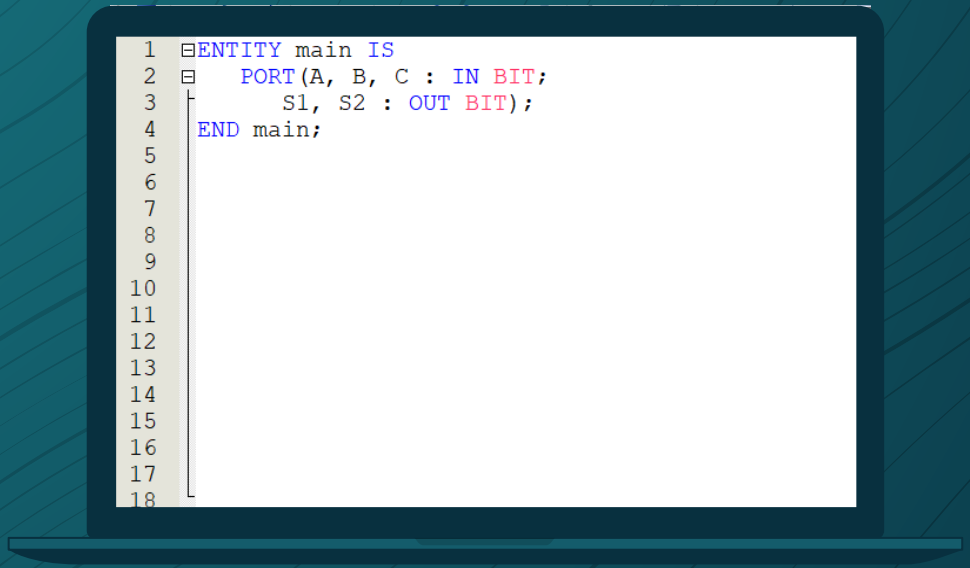
$S1 = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$
$S1 = \bar{A}BC + A(\bar{B}C + B\bar{C} + BC)$
$S1 = \bar{A}BC + A(\bar{B}C + B(\bar{C} + C))$
$S1 = \bar{A}BC + A(\bar{B}C + B \cdot 1)$
$S1 = \bar{A}BC + A(\bar{B}C + B)$
$S1 = \bar{A}BC + A((\bar{B} + B)(C + B))$
$S1 = \bar{A}BC + A(1 \cdot (C + B))$
$S1 = \bar{A}BC + A(C + B)$

$S1 = \bar{A}BC + AC + AB$
$S1 = C(\bar{A}B + A) + AB$
$S1 = C((\bar{A} + A)(B + A)) + AB$
$S1 = C(1 \cdot (B + A)) + AB$
$S1 = C(B + A) + AB$
$S1 = AC + CB + AB$
$S2 = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$



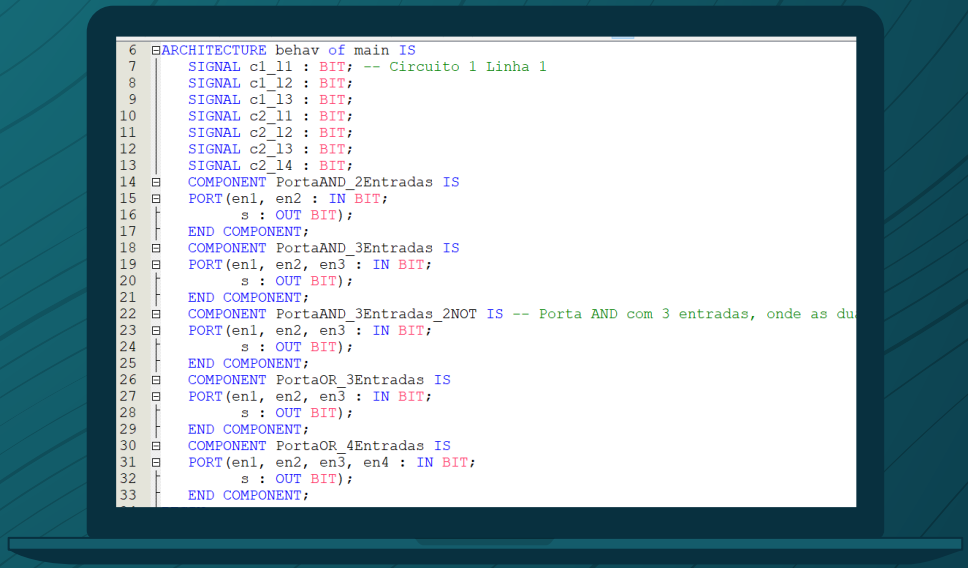
Representação dos circuitos na forma de portas lógicas

Desenvolvimento
ENTITY



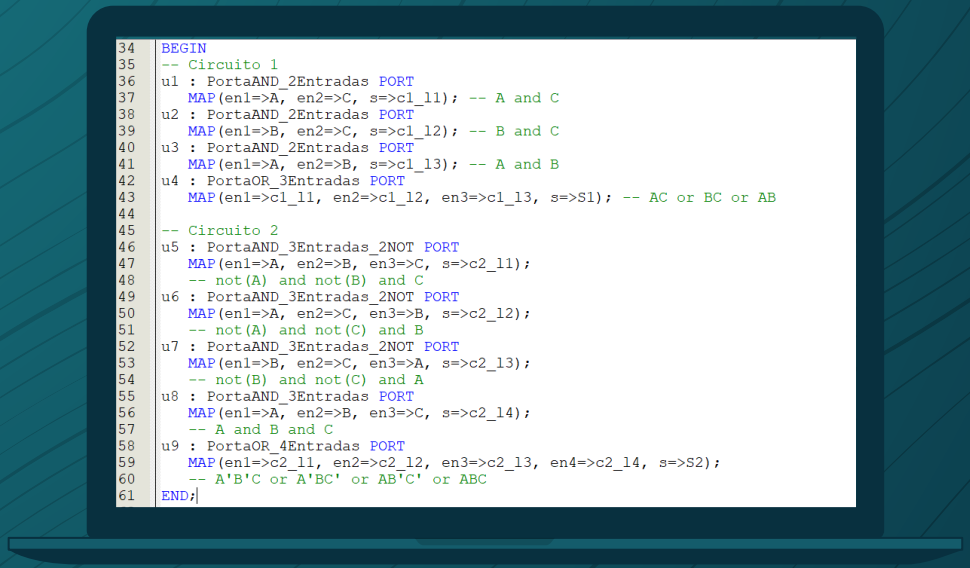
```
1  ENTITY main IS
2     PORT(A, B, C : IN BIT;
3          S1, S2 : OUT BIT);
4  END main;
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

Desenvolvimento
ARCHITECTURE



```
ARCHITECTURE behav of main IS
    SIGNAL c1_l1 : BIT; -- Circuito 1 Linha 1
    SIGNAL c1_l2 : BIT;
    SIGNAL c1_l3 : BIT;
    SIGNAL c2_l1 : BIT;
    SIGNAL c2_l2 : BIT;
    SIGNAL c2_l3 : BIT;
    SIGNAL c2_l4 : BIT;
    COMPONENT PortaAND_2Entradas IS
    PORT(en1, en2 : IN BIT;
         s : OUT BIT);
    END COMPONENT;
    COMPONENT PortaAND_3Entradas IS
    PORT(en1, en2, en3 : IN BIT;
         s : OUT BIT);
    END COMPONENT;
    COMPONENT PortaAND_3Entradas_2NOT IS -- Porta AND com 3 entradas, onde as du
    PORT(en1, en2, en3 : IN BIT;
         s : OUT BIT);
    END COMPONENT;
    COMPONENT PortaOR_3Entradas IS
    PORT(en1, en2, en3 : IN BIT;
         s : OUT BIT);
    END COMPONENT;
    COMPONENT PortaOR_4Entradas IS
    PORT(en1, en2, en3, en4 : IN BIT;
         s : OUT BIT);
    END COMPONENT;
```

Desenvolvimento
ARCHITECTURE



```
BEGIN
-- Circuito 1
u1 : PortaAND_2Entradas PORT
   MAP(en1=>A, en2=>C, s=>c1_l1); -- A and C
u2 : PortaAND_2Entradas PORT
   MAP(en1=>B, en2=>C, s=>c1_l2); -- B and C
u3 : PortaAND_2Entradas PORT
   MAP(en1=>A, en2=>B, s=>c1_l3); -- A and B
u4 : PortaOR_3Entradas PORT
   MAP(en1=>c1_l1, en2=>c1_l2, en3=>c1_l3, s=>S1); -- AC or BC or AB

-- Circuito 2
u5 : PortaAND_3Entradas_2NOT PORT
   MAP(en1=>A, en2=>B, en3=>C, s=>c2_l1);
   -- not(A) and not(B) and C
u6 : PortaAND_3Entradas_2NOT PORT
   MAP(en1=>A, en2=>C, en3=>B, s=>c2_l2);
   -- not(A) and not(C) and B
u7 : PortaAND_3Entradas_2NOT PORT
   MAP(en1=>B, en2=>C, en3=>A, s=>c2_l3);
   -- not(B) and not(C) and A
u8 : PortaAND_3Entradas PORT
   MAP(en1=>A, en2=>B, en3=>C, s=>c2_l4);
   -- A and B and C
u9 : PortaOR_4Entradas PORT
   MAP(en1=>c2_l1, en2=>c2_l2, en3=>c2_l3, en4=>c2_l4, s=>S2);
   -- A'B'C or A'BC' or AB'C' or ABC
END;
```

Desenvolvimento
COMPONENT

```
1  ENTITY PortaAND_2Entradas IS
2  PORT(en1, en2 : IN BIT;
3       s : OUT BIT);
4  END PortaAND_2Entradas;
5
6  ARCHITECTURE behav of portaAND_2Entradas IS
7  BEGIN
8  s <= en1 and en2;
9  END;
10
```

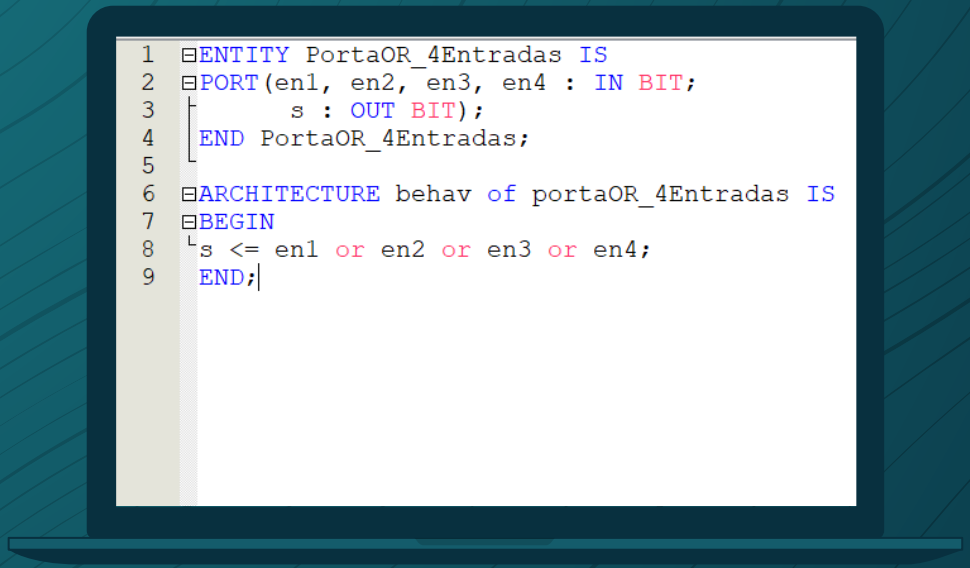
```
1  ENTITY PortaAND_3Entradas IS
2  PORT(en1, en2, en3 : IN BIT;
3       s : OUT BIT);
4  END PortaAND_3Entradas;
5
6  ARCHITECTURE behav of portaAND_3Entradas IS
7  BEGIN
8  s <= en1 and en2 and en3;
9  END;
10
```

Desenvolvimento
COMPONENT

```vhdl
ENTITY PortaAND_3Entradas_2NOT IS
PORT(en1, en2, en3 : IN BIT;
      s : OUT BIT);
END PortaAND_3Entradas_2NOT;

ARCHITECTURE behav of portaAND_3Entradas_2NOT
BEGIN
s <= not(en1) and not(en2) and en3;
END;
```

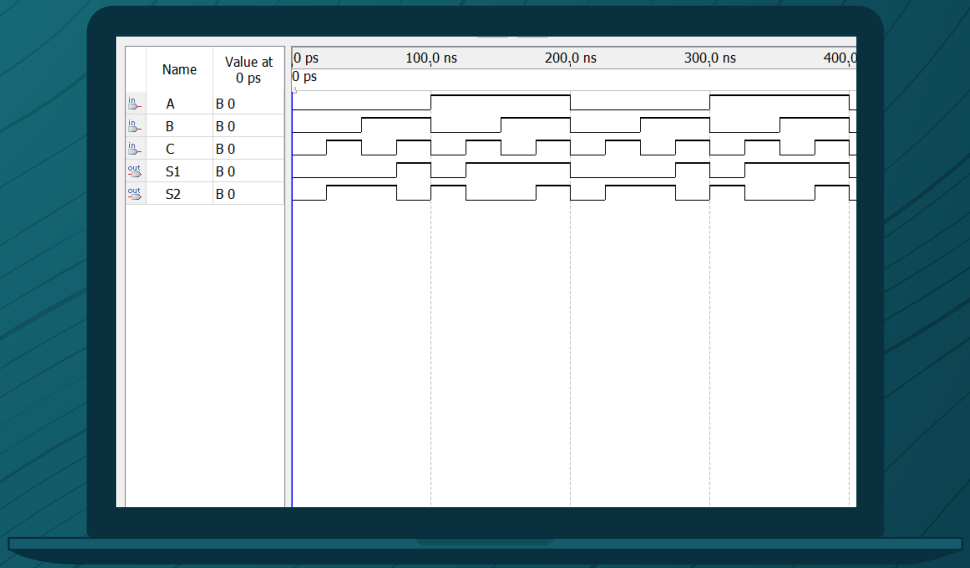
```vhdl
ENTITY PortaOR_3Entradas IS
PORT(en1, en2, en3 : IN BIT;
      s : OUT BIT);
END PortaOR_3Entradas;

ARCHITECTURE behav of portaOR_3Entradas IS
BEGIN
s <= en1 or en2 or en3;
END;
```

Desenvolvimento
COMPONENT



```
1  ENTITY PortaOR_4Entradas IS
2  PORT(en1, en2, en3, en4 : IN BIT;
3       s : OUT BIT);
4  END PortaOR_4Entradas;
5
6  ARCHITECTURE behav of portaOR_4Entradas IS
7  BEGIN
8  s <= en1 or en2 or en3 or en4;
9  END;
```

Simulação


Name
Value at 0 ps
A
B 0
B
B 0
C
B 0
S1
B 0
S2
B 0
0 ps
100,0 ns
200,0 ns
300,0 ns
400,0
0 ps

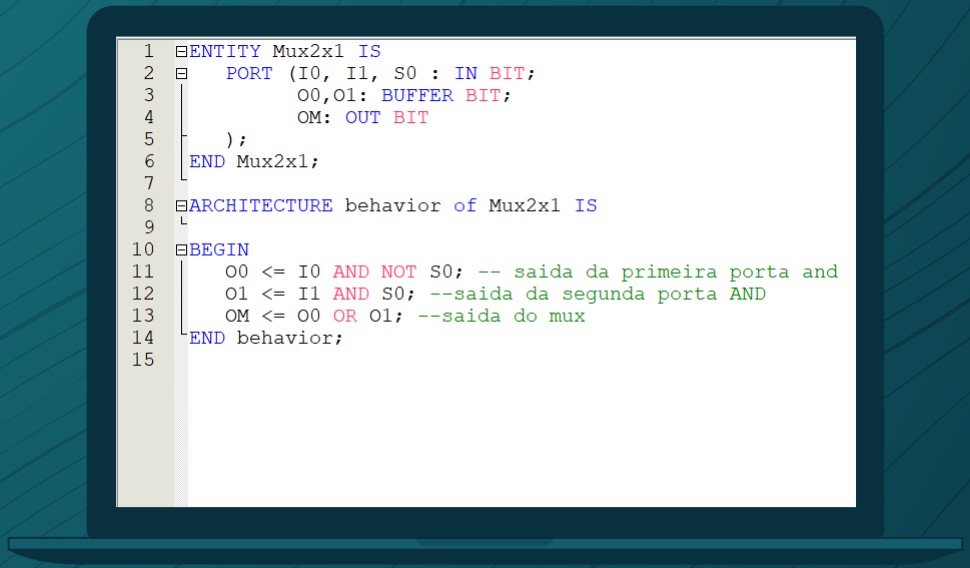
Laboratório 03

CIRCUITOS LÓGICOS

Objetivos:

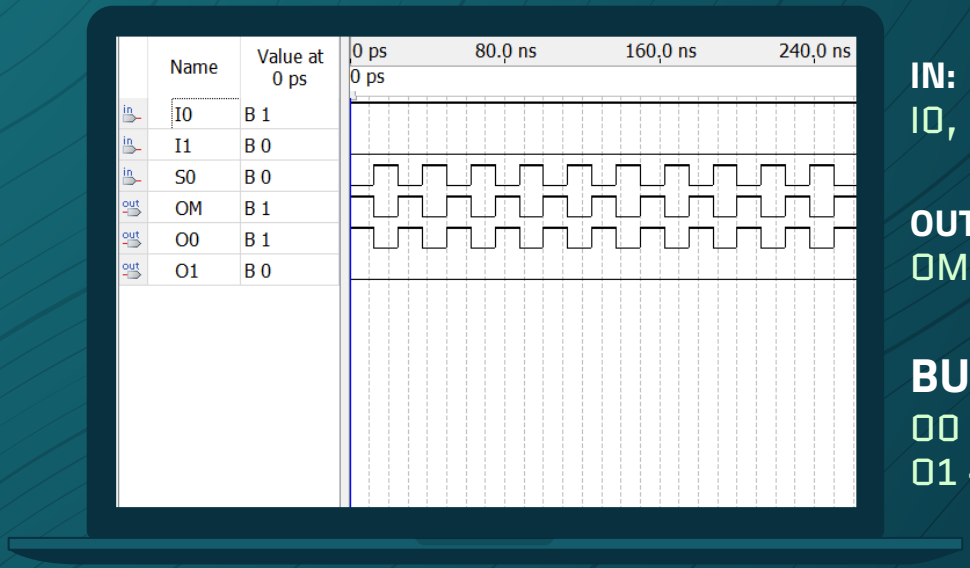
- Experimentar a descrição em VHDL de circuitos na forma comportamental;
- Reforçar os conceitos de **Multiplexadores e Demultiplexadores**;
- Por em prática conceitos aprendidos na disciplina Circuitos Digitais - Teoria.

Circuitos lógicos
Mux 2x1



```
ENTITY Mux2x1 IS
    PORT (I0, I1, S0 : IN BIT;
          O0,O1: BUFFER BIT;
          OM: OUT BIT
    );
END Mux2x1;

ARCHITECTURE behavior of Mux2x1 IS

BEGIN
    O0 <= I0 AND NOT S0; -- saida da primeira porta and
    O1 <= I1 AND S0; --saida da segunda porta AND
    OM <= O0 OR O1; --saida do mux
END behavior;
```

Simulação
Mux 2x1

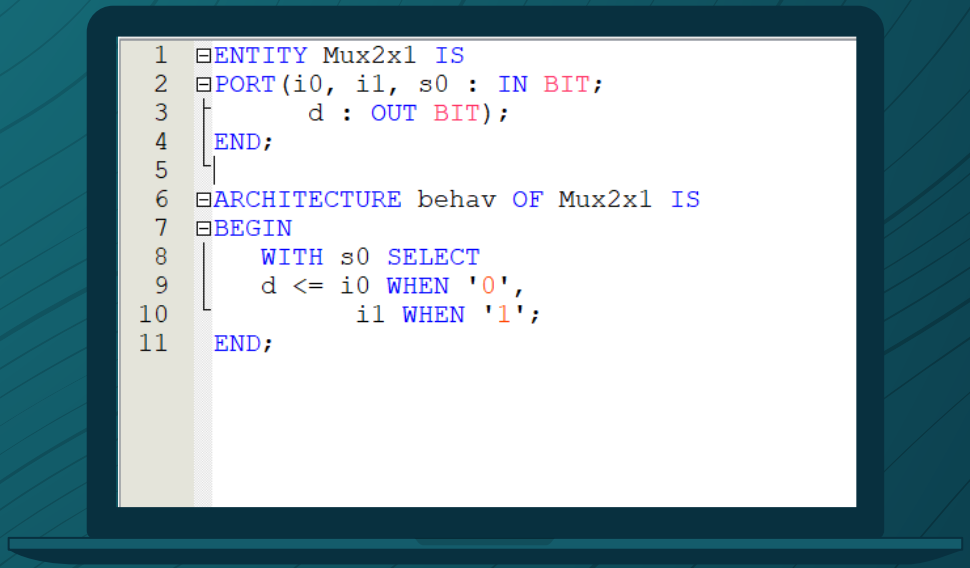


IN:
I0, I1, S0.

OUT:
OM <= O0 OR O1;

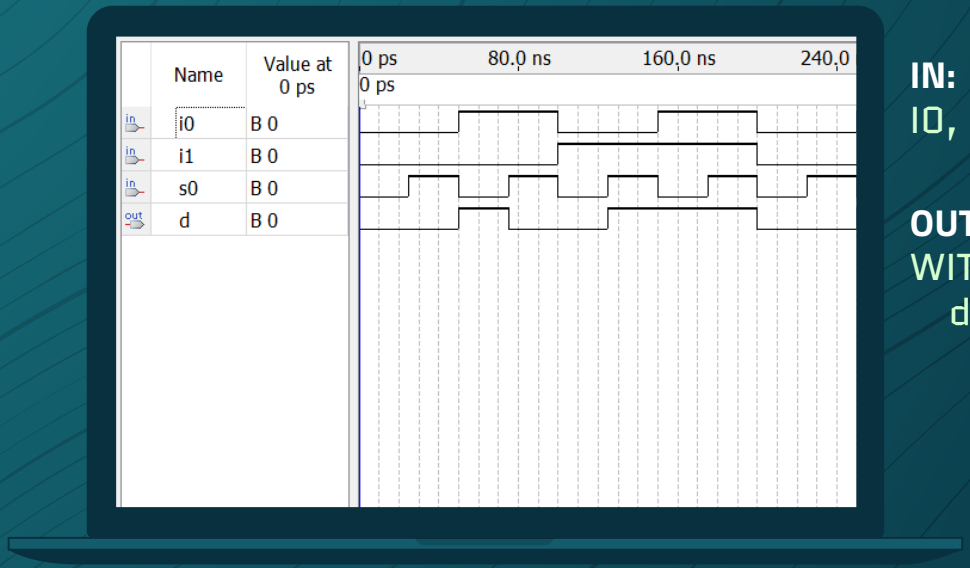
BUFFER:
O0 <= I0 AND NOT S0;
O1 <= I1 AND S0;

Descrição comportamental
Mux 2x1



```
1  ENTITY Mux2x1 IS
2  PORT(i0, i1, s0 : IN BIT;
3       d : OUT BIT);
4  END;
5
6  ARCHITECTURE behav OF Mux2x1 IS
7  BEGIN
8     WITH s0 SELECT
9     d <= i0 WHEN '0',
10          i1 WHEN '1';
11 END;
```

Simulação
Mux 2x1

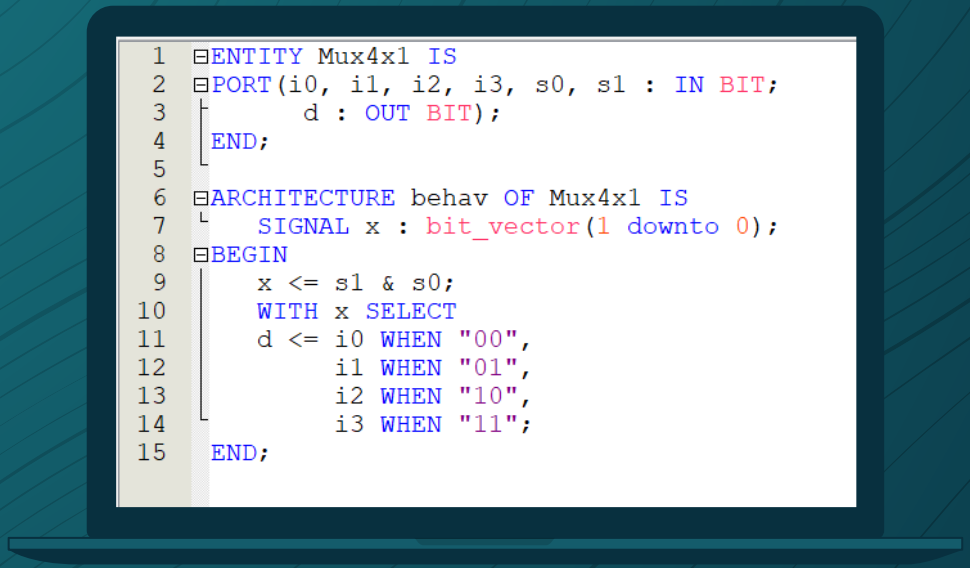


IN:
I0, I1, S0.

OUT:

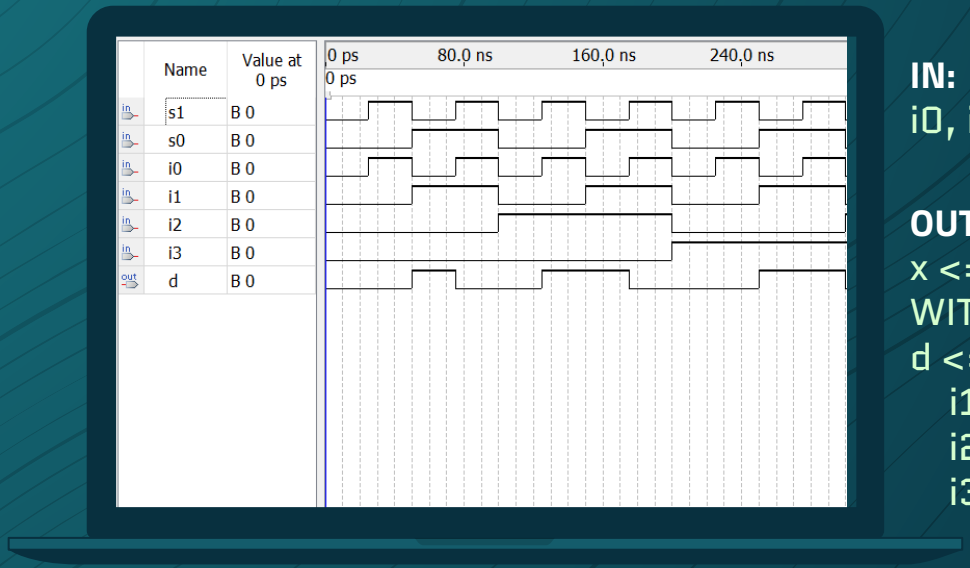
```
WITH s0 SELECT
   d <= i0 WHEN '0',
        i1 WHEN '1';
```

Descrição comportamental
Mux 4x1



```
1  ENTITY Mux4x1 IS
2  PORT(i0, i1, i2, i3, s0, s1 : IN BIT;
3       d : OUT BIT);
4  END;
5
6  ARCHITECTURE behav OF Mux4x1 IS
7     SIGNAL x : bit_vector(1 downto 0);
8  BEGIN
9     x <= s1 & s0;
10    WITH x SELECT
11    d <= i0 WHEN "00",
12         i1 WHEN "01",
13         i2 WHEN "10",
14         i3 WHEN "11";
15 END;
```

Simulação
Mux 4x1



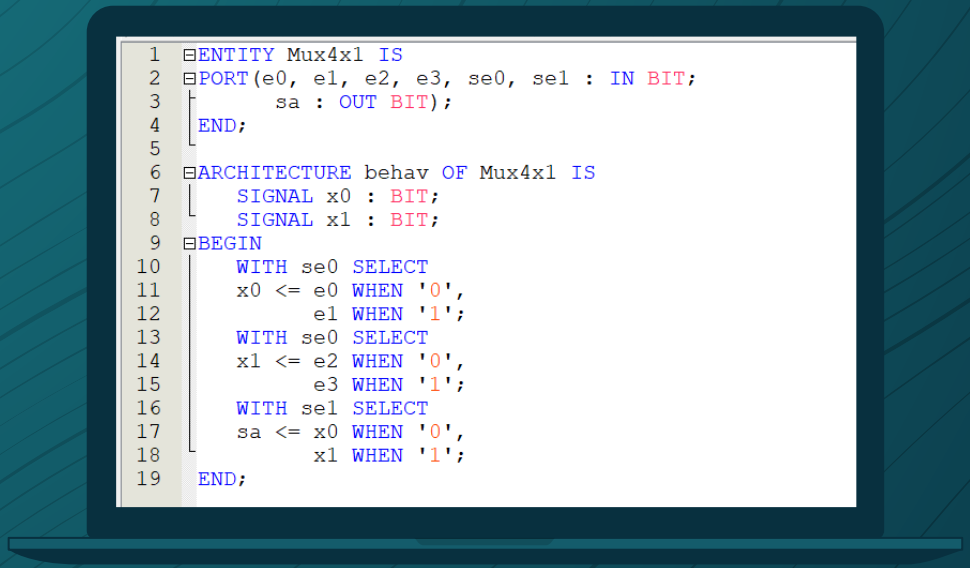
IN:

i0, i1, i2, i3, s0, s1

OUT:

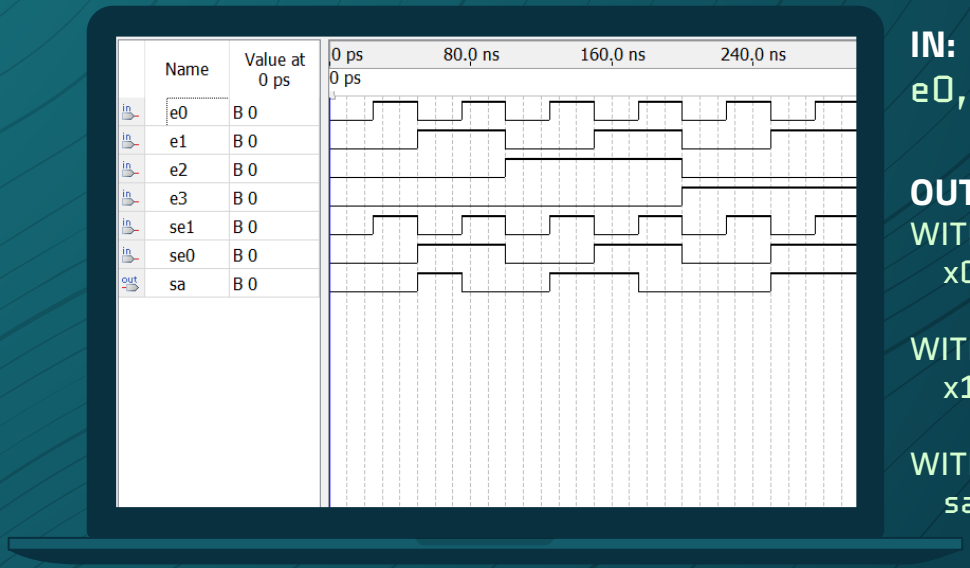
```
x <= s1 & s0;
WITH x SELECT
d <= i0 WHEN "00",
   i1 WHEN "01",
   i2 WHEN "10",
   i3 WHEN "11";
```

Descrição comportamental + Componentes
Mux 4x1 utilizando Mux 2x1.



```
1  ENTITY Mux4x1 IS
2  PORT(e0, e1, e2, e3, se0, se1 : IN BIT;
3       sa : OUT BIT);
4  END;
5
6  ARCHITECTURE behav OF Mux4x1 IS
7     SIGNAL x0 : BIT;
8     SIGNAL x1 : BIT;
9  BEGIN
10    WITH se0 SELECT
11    x0 <= e0 WHEN '0',
12          e1 WHEN '1';
13    WITH se0 SELECT
14    x1 <= e2 WHEN '0',
15          e3 WHEN '1';
16    WITH se1 SELECT
17    sa <= x0 WHEN '0',
18          x1 WHEN '1';
19 END;
```

Simulação
Mux 4x1



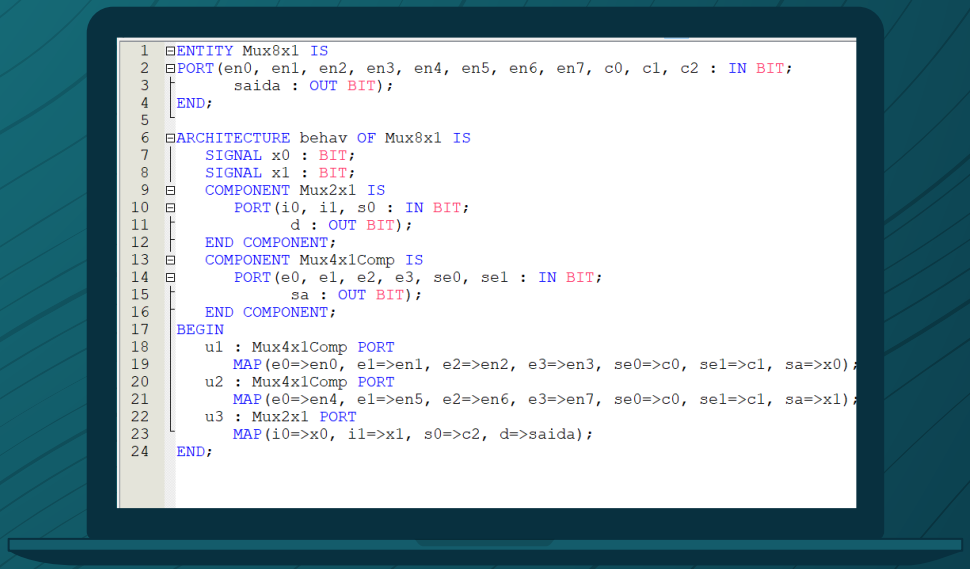
IN:

e0, e1, e2, e3, se0, se1

OUT:

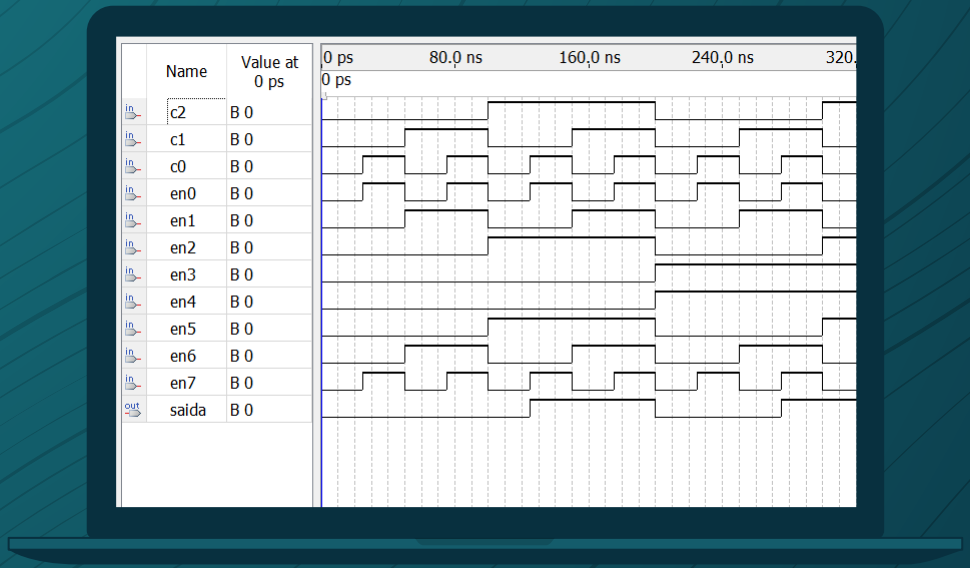
```
WITH se0 SELECT
   x0 <= e0 WHEN '0',
         e1 WHEN '1';
WITH se0 SELECT
   x1 <= e2 WHEN '0',
         e3 WHEN '1';
WITH se1 SELECT
   sa <= x0 WHEN '0',
         x1 WHEN '1';
```

Componentes
Mux 8x1 utilizando Mux 4x1

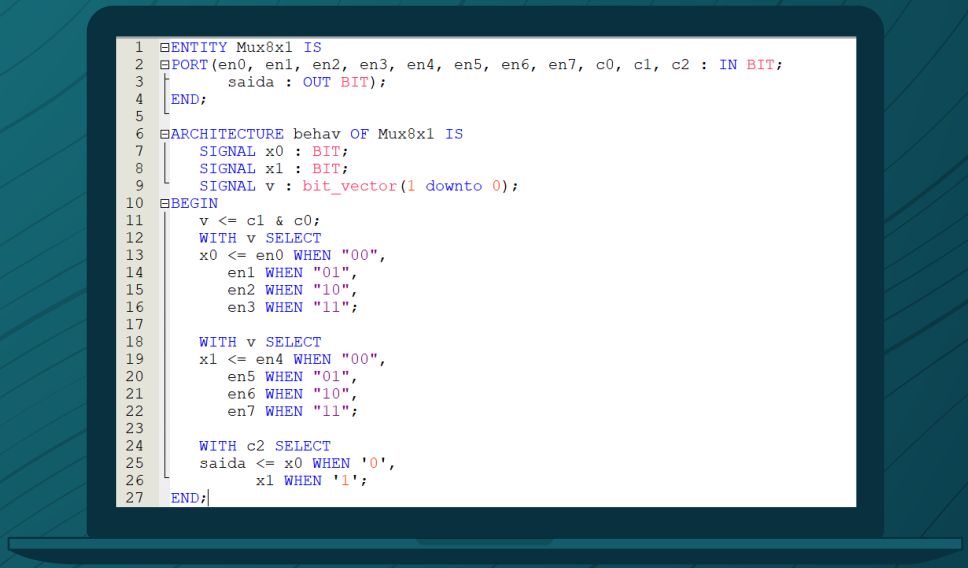


```
ENTITY Mux8x1 IS
PORT(en0, en1, en2, en3, en4, en5, en6, en7, c0, c1, c2 : IN BIT;
     saida : OUT BIT);
END;

ARCHITECTURE behav OF Mux8x1 IS
   SIGNAL x0 : BIT;
   SIGNAL x1 : BIT;
   COMPONENT Mux2x1 IS
      PORT(i0, i1, s0 : IN BIT;
           d : OUT BIT);
   END COMPONENT;
   COMPONENT Mux4x1Comp IS
      PORT(e0, e1, e2, e3, se0, se1 : IN BIT;
           sa : OUT BIT);
   END COMPONENT;
BEGIN
   u1 : Mux4x1Comp PORT
      MAP(e0=>en0, e1=>en1, e2=>en2, e3=>en3, se0=>c0, se1=>c1, sa=>x0);
   u2 : Mux4x1Comp PORT
      MAP(e0=>en4, e1=>en5, e2=>en6, e3=>en7, se0=>c0, se1=>c1, sa=>x1);
   u3 : Mux2x1 PORT
      MAP(i0=>x0, i1=>x1, s0=>c2, d=>saida);
END;
```

Simulação
Mux 8x1 utilizando Mux 4x1

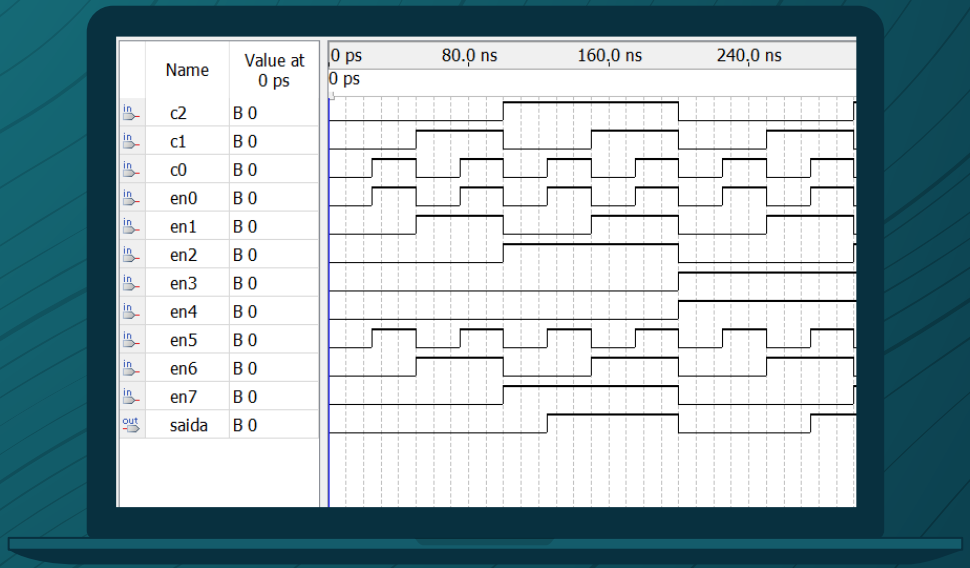


Descrição comportamental
Mux 8x1 utilizando Mux 4x1



```
ENTITY Mux8x1 IS
PORT(en0, en1, en2, en3, en4, en5, en6, en7, c0, c1, c2 : IN BIT;
     saida : OUT BIT);
END;

ARCHITECTURE behav OF Mux8x1 IS
   SIGNAL x0 : BIT;
   SIGNAL x1 : BIT;
   SIGNAL v : bit_vector(1 downto 0);
BEGIN
   v <= c1 & c0;
   WITH v SELECT
   x0 <= en0 WHEN "00",
      en1 WHEN "01",
      en2 WHEN "10",
      en3 WHEN "11";

   WITH v SELECT
   x1 <= en4 WHEN "00",
      en5 WHEN "01",
      en6 WHEN "10",
      en7 WHEN "11";

   WITH c2 SELECT
   saida <= x0 WHEN '0',
         x1 WHEN '1';
END;
```

Simulação
Mux 8x1 utilizando Mux 4x1



Obrigado!

Alguma Dúvida?