Circuitos digitais

André Furlan - andre.furlan@unesp.br

Universidade Estadual Paulista Júlio de Mesquita Filho

2024

unesp

Sistema binário, álgebra de Boole e lógica

unesp

Números binários

Vantagens

- Dois estados (mais difícil deteriorar)
- Discretos
- Manipulação algébrica (álgebra de Boole)

Mundo ideal: Junção do analógico com o digital

Desantagens

- Dois estados em oposição a infinitos estados
- Discretos em oposição a contínuo

unesp

Sistema numérico posicional

Dada a base com uma certa quantidade de elementos B e símbolos $d \in D = \{d_0, \ldots, d_n\}$, então um valor numérico decimal V **posicional** pode ser representado por

$$V(B) = \sum_{i=0}^{n} d_i.B^{n-i} \qquad (1)$$

Sendo $||$ o operador de cardinalidade então $n = |D| - 1$

Por exemplo, para o número **decimal** 249:

$B = 10$

$D = \{2, 4, 9\} \implies |D| = 3 \implies n = 2$

$$V(10) = \sum_{i=0}^{2} d_i.10^{2-i} =$$

$$d_0.10^{2-0} + d_1.10^{2-1} + d_2.10^{2-2} = \qquad (2)$$

$$2.10^2 + 4.10^1 + 9.10^0 =$$

$$\boxed{2.100 + 4.10 + 9.1 = 249}$$

unesp

Sistema numérico posicional

Dada a base com uma certa quantidade de elementos B e símbolos $d \in D = \{d_0, \ldots, d_n\}$, então um valor numérico decimal V **posicional** pode ser representado por

$$V(B) = \sum_{i=0}^{n} d_i . B^{n-i} \qquad (1)$$

Sendo $||$ o operador de cardinalidade então $n = |D| - 1$

Por exemplo, para o número **binário** 11:

$B = 2$

$D = \{1, 1\} \implies |D| = 2 \implies n = 1$

$$V(2) = \sum_{i=0}^{1} d_i . 2^{1-i} =$$

$$d_0 . 2^{1-0} + d_1 . 2^{1-1} = \qquad (2)$$

$$1.2^1 + 1.2^0 =$$

$$\boxed{2 + 1 = 3}$$

unesp

Circuitos lógicos e algébra Booleana

Antes de avançarmos no estudo e na representação de circuitos lógicos, vamos entender como os estados de "ligado" e "desligado" dos componentes podem ser utilizados para gerar resultados úteis. Portanto, antes de abordarmos esses tópicos, faremos uma imersão na álgebra booleana, explorando tabelas verdade e formas de composição.

unesp

Álgebra de Boole: Operadores **AND**, **OR** e **NOT**

Sendo o sistema binário posicional aplica-se a ele muitas das regras as quais podemos aplicar aos números decimais o que nos leva a Álgebra de Boole e seus operadores:

AND denotado como **.** ou $\wedge$

O operador **AND** é uma função de 2 variáveis definida como:

$$
\begin{aligned}
&AND(a,b) = a.b \\
&AND(a,b) = 1, \forall a = b = 1 \\
&AND(a,b) = 0, \forall a \neq b
\end{aligned}
\tag{3}
$$

a	b	a . b
0	0	0
0	1	0
1	0	0
1	1	1

Tabela: Tabela verdade para **AND**.

unesp

Álgebra de Boole: Operadores **AND**, **OR** e **NOT**

Sendo o sistema binário posicional aplica-se a ele muitas das regras as quais podemos aplicar aos números decimais o que nos leva a Álgebra de Boole e seus operadores:

OR denotado com $+$ ou $\vee$

O operador **OR** é uma função de 2 variáveis definida como:

$$
\begin{aligned}
&OR(a,b) = a + b \\
&OR(a,b) = 0, \forall a = b = 0 \\
&OR(a,b) = 1, \forall a \neq b, a = b = 1
\end{aligned}
\tag{3}
$$

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

Tabela: Tabela verdade para **OR**.

unesp

Álgebra de Boole: Operadores **AND**, **OR** e **NOT**

Sendo o sistema binário posicional aplica-se a ele muitas das regras as quais podemos aplicar aos números decimais o que nos leva a Álgebra de Boole e seus operadores:

NOT denotado com $\overline{a}$ ou $\neg a$

O operador **NOT** é uma função de 1 variável definida como:

$$\begin{aligned} &NOT(a) = \overline{a} \\ &NOT(a) = 0, \forall a = 1 \\ &NOT(a) = 1, \forall a = 0 \end{aligned} \tag{3}$$

a	$\overline{a}$
0	1
1	0

Tabela: Tabela verdade para **NOT**.

unesp

Álgebra de Boole: Composição de de operadores

É importante destacar que assim com na álgebra tradicional os operadores tem prioridade no momento de sua aplicação na seguinte ordem: **NOT**, **AND** e por fim **OR**. Se usarmos os sinais as regras ficam bem parecidas com a álgebra tradicional: $\overline{a}$, . e $+$.

A partir dos operadores apresentados vamos brincar um pouquinho com eles:

$$(a + b).c \qquad (4)$$

a	b	c	a+b	$a + b$. c
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Tabela: Tabela verdade para **(a + b) . c**.

unesp

Álgebra de Boole: Composição de operadores NAND e NOR

Alguns operadores comuns são compostos:

O operador **NAND** (NOT AND) é definido segundo a seguinte tabela verdade:

a	b	a . b	$\overline{a.b}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Tabela: Tabela verdade para **NAND**.

O operador **NOR** (NOT OR) é definido segundo a seguinte tabela verdade:

a	b	a + b	$\overline{a+b}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Tabela: Tabela verdade para **NOR**.

unesp

Álgebra de Boole: Expressão booleanas equivalentes

Uma expressão boleana equivalente é aquela que, dada uma mesma entrada, retorna exatamente o mesmo resultado. Expressões boleanas equivalentes podem ser ou não a versão otimizada uma da outra.

O que foi dito acima só é possível se alguns **axiomas**[1] forem definidos.
A partir desses axiomas podemos definir **teoremas**[2].

Veremos ambos no próximo slide.

[1] hipóteses básicas ou pré-supostos baseados na experiência empírica ou filosófica
[2] Teoremas são proposições demonstráveis a partir dos axiomas

unesp

Álgebra de Boole: Expressão booleanas equivalentes

Axiomas e teoremas

Axiomas:

1. $0.0 = 0$
2. $1.1 = 1$
3. $0.1 = 1.0 = 0$
4. $1 + 1 = 1$
5. $0 + 0 = 0$
6. $1 + 0 = 0 + 1 = 1$
7. $x = 0 \implies \overline{x} = 1$
8. $x = 1 \implies \overline{x} = 0$

Teoremas para AND

1. $x.0 = 0$
2. $x.1 = x$
3. $x.x = x$
4. $x.\overline{x} = 0$

Teoremas para OR

1. $x + 1 = 1$
2. $x + 0 = x$
3. $x + x = x$
4. $x + \overline{x} = 1$

Teorema para NOT

1. $\overline{(\overline{x})} = x$

unesp

Dada uma expressão lógica que expressa uma igualdade então é possível criar uma **expressão dual** na qual a igualdade continua verdadeira.

Uma expressão dual é obtida quando se troca os valores de 0 para 1, de 1 para 0 e os operadores de and para or e de or para and. Perceba que **o resultado da expressão pode mudar** porém a **igualdade é preservada**.

Voltando ao slide 10 você perceberá que os teoremas para *OR* são **dualidades de uma variável** dos teoremas de *AND* e vice-versa.

$$x.y = y.x \Leftrightarrow x + y = y + x \text{ comutacao}$$
$$x + (x.y) = x \Leftrightarrow x.(x + y) = x \text{ absorcao}$$
$$(x.y) + (x.\overline{y}) = x \Leftrightarrow (x + y).(x + \overline{y}) = x \text{ combinacao} \quad (5)$$
$$\boxed{\text{Teorema de De Morgan: } \overline{(x.y)} = \overline{x} + \overline{y}} \Leftrightarrow \overline{(x + y)} = \overline{x}.\overline{y}$$
$$x + (\overline{x}.y) = x + y \Leftrightarrow x.(\overline{x} + y) = x.y$$

$$
\begin{aligned}
x.(y.z) = (x.y).z \Leftrightarrow x + (y + z) = (x + y) + z \text{ associação} \\
x.(y + z) = (x.y) + (x.z) \Leftrightarrow x + (y.z) = (x + y).(x + z) \text{ distribuição} \\
(x.y) + (y.z) + (\overline{x}.z) = (x.y) + (\overline{x}.z) \Leftrightarrow (x + y).(y + z).(\overline{x} + z) = (x + y).(\overline{x} + z) \\
\text{consenso (sumiço :D)}
\end{aligned}
\tag{6}
$$

Em tempo: Eu sei que em alguns lugares tem parenteses sobrando, isso foi intencional pra facilitar a visualização das relações.

unesp

Teorema de D.Morgan

Tabela: Prova exaustiva do teorema de De Morgan: $\overline{(x.y)} = \overline{x} + \overline{y}$

x	y	$x.y$	$\overline{x.y}$	$\overline{x}$	$\overline{y}$	$\overline{x} + \overline{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Tabela: Prova exaustiva para o dual do teorema de De Morgan: $\overline{(x + y)} = \overline{x}.\overline{y}$

x	y	$x + y$	$\overline{x + y}$	$\overline{x}$	$\overline{y}$	$\overline{x}.\overline{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

unesp

Provas de expressões na álgebra de Boole

- indução perfeita: Devido a pequena quantidade de valores possíveis (0 ou 1) geralmente é viável provar as igualdades dessa forma. A indução perfeita (prova exaustiva) se dá ao fazer a tabela verdade da igualdade, verificando se os valores literais realmente coincidem. Quando a expressão começa a ter muitas variáveis e/ou se tornar muito grande, esse método pode se tornar inviável devido a grande quantidade de estados possíveis que o sistema pode ter.
- manipulação algébrica: Levando em consideração os axiomas e teoremas é possível manipular as igualdades de forma que a igualdade seja provada.
- diagrama de Ven: *Nos próximos capítulos*.

Provas de expressões na álgebra de Boole com diagrama de Venn

Se interpretarmos **AND** ou **.** como $\cap$ e **OR** ou **+** como $\cup$ é possível fazer a prova de uma igualdade usando o diagrama de Venn.

Para começar, uma dica interessante é representar as relações do espaço para identificar cada uma das regiões de acordo com a expressão avaliada.

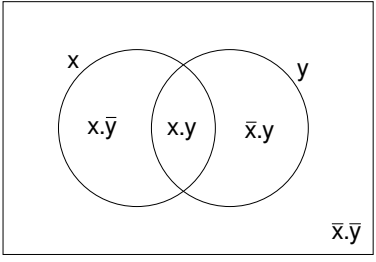


Figura: Relações de um espaço com duas variáveis

A partir disso é possível escrever outras relações:

$$\begin{aligned} (x.\overline{y}) + (x.y) &= x \\ (\overline{x}.y) + (x.y) &= y \\ (x.\overline{y}) + (x.y) + (\overline{x}.y) &= x + y \end{aligned} \tag{7}$$

unesp

Provas de expressões na álgebra de Boole com diagrama de Venn

Como exemplo provaremos o do teorema de DeMorgan: $\overline{(x.y)} = \overline{x} + \overline{y}$

Lado esquerdo da igualdade.

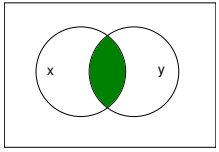


Figura: $(x.y)$

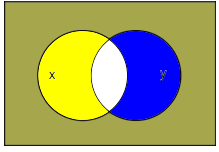
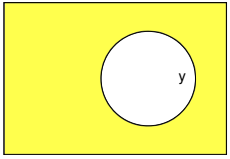


Figura: $\overline{(x.y)}$

Lado direito da igualdade.



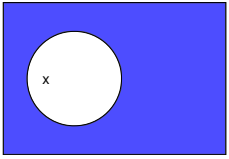


Figura: $\overline{x}$ e $\overline{y}$

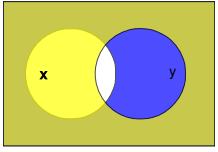


Figura: $\overline{x} + \overline{y}$

unesp

Provas de expressões na álgebra de Boole com diagrama de Venn

Como exemplo provaremos $(x.y) + (y.z) + (\overline{x}.z) = (x.y) + (\overline{x}.z) \Leftrightarrow$
$(x + y).(y + z).(\overline{x} + z) = (x + y).(\overline{x} + z)$ consenso (sumiço :D)

Lado esquerdo da igualdade.

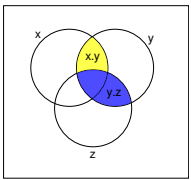


Figura: $(x.y) + (y.z)$

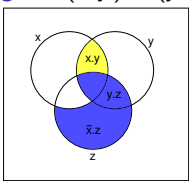


Figura: $(x.y) + (y.z) + (\overline{x}.z)$

Lado direito da igualdade.

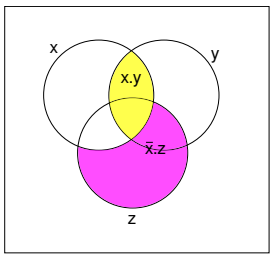


Figura: $(x.y) + (\overline{x}.z)$

unesp

Mini prova 02

A partir dos teoremas e propriedades apresentados mostre que $(x + y).(x + z) = x + y.z$. Indique quais os teoremas e/ou propriedades usados na solução.

Crie dualidades para

- 0.0
- 1.1
- $x = 0 \implies \overline{x} = 0$

unesp

Prática dirigida

Crie a tabela verdade para a porta lógica AND.
Crie a tabela verdade para a porta lógica OR.
Crie a tabela verdade para a porta lógica NOT.

unesp

Prática dirigida

Dada a tabela verdade a seguir, construa a expressão lógica correspondente:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

unesp

Prática dirigida

Dada a expressão lógica $F = A \cdot \overline{B}$, desenhe o circuito combinacional correspondente.
Dada a expressão lógica $F = A \cdot B + \overline{A} \cdot \overline{B}$, construa a tabela verdade correspondente.

unesp

Circuitos combinacionais

unesp

Introdução

Até agora, revisamos a lógica, as tabelas-verdade e a álgebra booleana. No entanto, ainda não abordamos os circuitos digitais propriamente ditos. A partir de agora, exploraremos a correlação entre as expressões lógicas da álgebra booleana e o projeto de circuitos elétricos.

unesp

A chave binária é um componente que permite ou impede a passagem da corrente elétrica dado seus estado. A partir dela criaremos circuitos mais complexos.

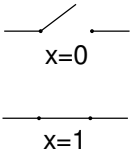


Figura: Chave binária

A chave binária pode ser representada de outra forma:

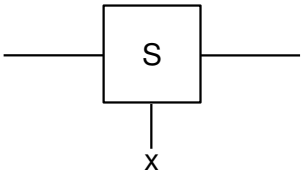


Figura: Chave binária

Agora considere uma função que depende um argumento x para definir seu estado. Tal função pode ser representada no circuito abaixo:

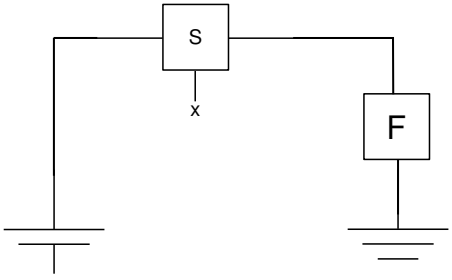


Figura: Função lógica de uma variável. No circuito F pode ser ligada a qualquer elemento de saída com um led ou outro circuito.

$F = 1 \forall x = 1$
$F = 0 \forall x = 0$
Portanto F é uma função lógica de **uma** variável tal que $F(x) = x$

unesp

A partir de agora é possível construir circuitos mais complexos como **OR** e **AND**.

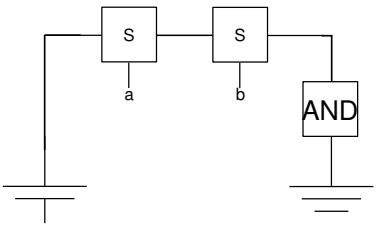


Figura: Circuito **AND**: *a* e *b* são suas entradas. Aqui se pode constatar que este circuito pode receber de 2 a *n* entradas.

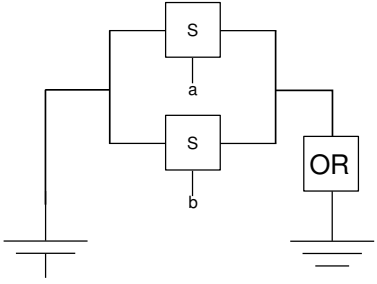


Figura: Circuito **OR**: *a* e *b* são suas entradas. Aqui se pode constatar que este circuito pode receber de 2 a *n* entradas.

unesp

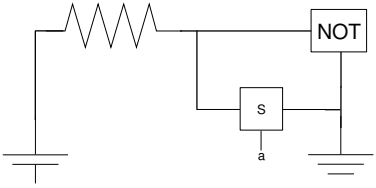


Figura: Circuito **NOT**: Inverte a entrada *a*.

Nada aqui por questões estéticas...

E assim como vimos na álgebra de Boole **OR** e **AND** podem ser combinados.

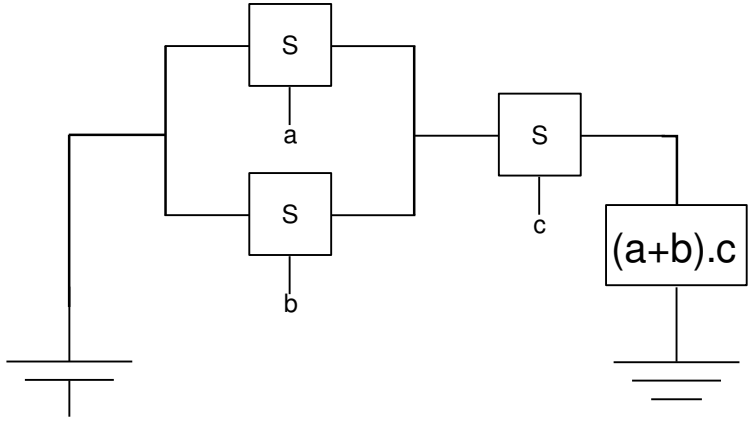


Figura: O **OR** recebe as entradas *a* e *b*, o circuito **AND** recebe *c* e a saída de **OR**

Portas lógicas

unesp

Para representar circuitos simples os símbolos já vistos são suficientes, porém, quando a complexidade aumenta pode ficar beeeeeemmmm complicado entender o que se passa. Então precisamos aumentar o nível de abstração encapsulando esse componentes.

Portas lógicas

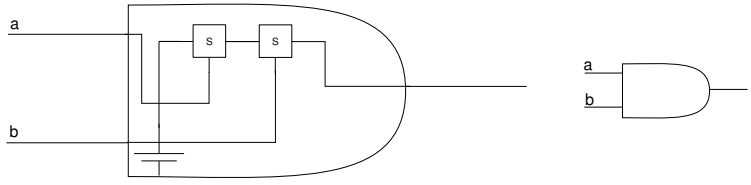


Figura: Abstração de circuito **AND** para a porta **AND**

Portas lógicas

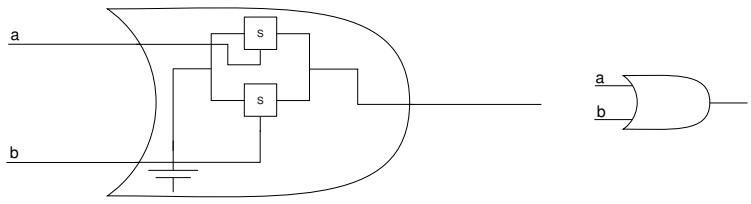


Figura: Abstração de circuito **OR** para a porta **OR**

Portas lógicas

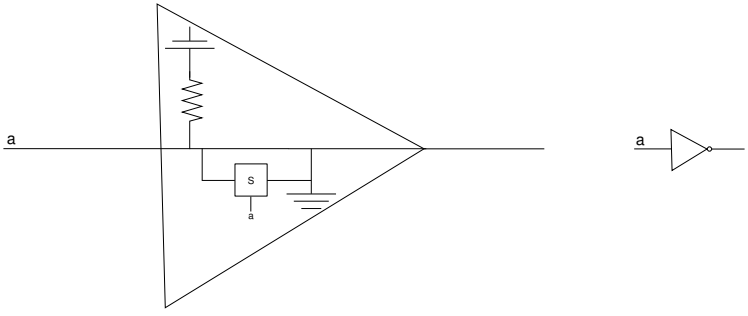


Figura: Abstração de circuito **NOT** para a porta **NOT**

Portas lógicas

Veja como fica o circuito $(a + b).c$ com as novas abstrações.

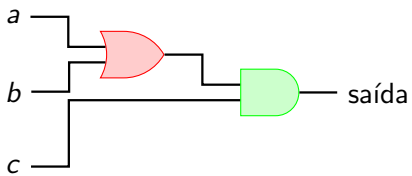


Figura: Circuito correspondente a expressão lógica (a+b).c

Portas lógicas

Agora que temos acesso a novos circuitos que foram abstraídos em portas podemos, combinando essas portas, criar outras como a seguir.

XOR: $\boxed{\overline{a}.b + a.\overline{b} = a \oplus b}$

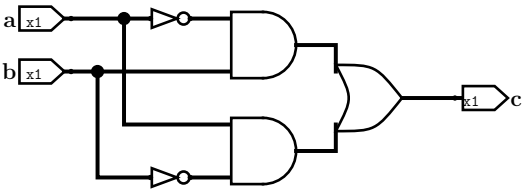


Figura: O circuito da porta XOR

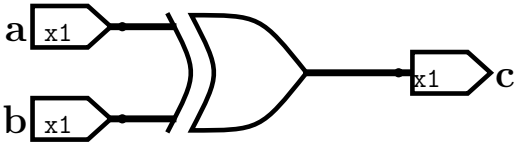


Figura: Porta XOR

unesp

Portas lógicas

NOR: $\boxed{\overline{a+b} = a \downarrow b}$

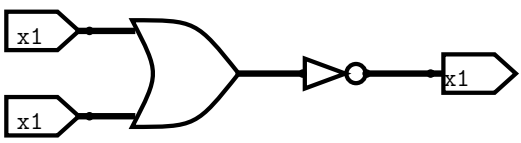


Figura: O circuito da porta NOR

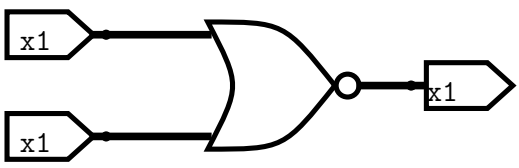


Figura: Porta NOR

Portas lógicas

NAND: $\boxed{\overline{a.b} = a \uparrow b}$

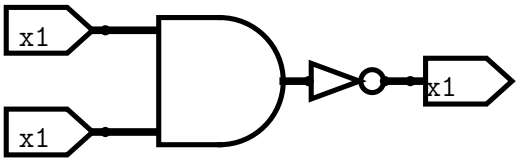


Figura: O circuito da porta NAND

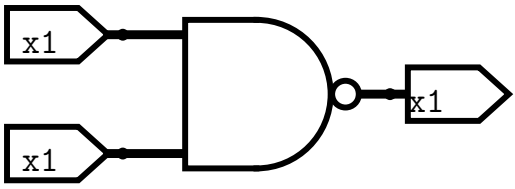
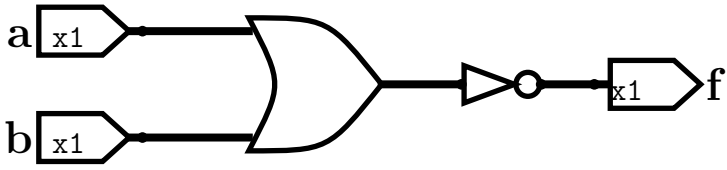


Figura: Porta NAND

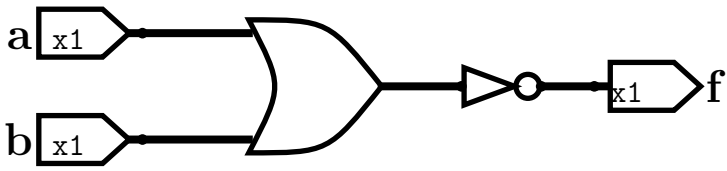
Prática dirigida 01

Dado o circuito combinacional a seguir, construa a expressão lógica e a tabela verdade:



Prática dirigida 01

Dado o circuito combinacional a seguir, construa a expressão lógica e a tabela verdade:



Expressão: $f = \overline{(a + b)}$

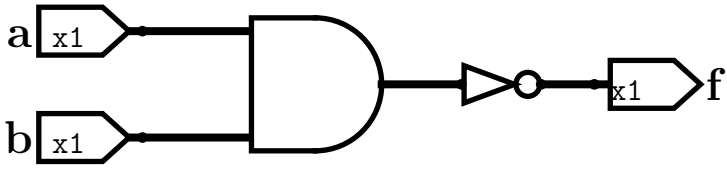
Tabela verdade:

x	y	z
0	0	1
0	1	0
1	0	1
1	1	1

unesp

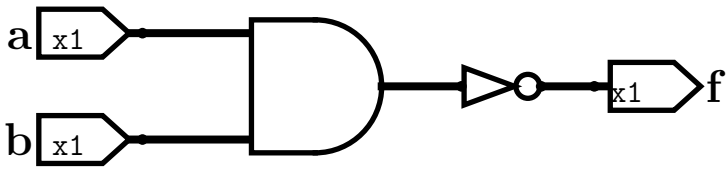
Prática dirigida 02

Dado o circuito combinacional a seguir, construa a expressão lógica e a tabela verdade:



Prática dirigida 02

Dado o circuito combinacional a seguir, construa a expressão lógica e a tabela verdade:



Expressão: $f = \overline{(a.b)}$

Tabela verdade:

a	b	x
0	0	1
0	1	1
1	0	1
1	1	0

unesp

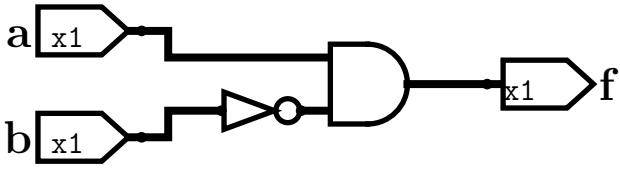
Prática dirigida 03

Dada a expressão lógica $f = a \cdot \overline{b}$, desenhe o circuito combinacional correspondente.

unesp

Prática dirigida 03

Dada a expressão lógica $f = a \cdot \overline{b}$, desenhe o circuito combinacional correspondente.



Prática dirigida 04

Dada a expressão lógica $f = a \cdot b + \overline{a} \cdot \overline{b}$, construa a tabela verdade correspondente.

unesp

Prática dirigida 04

Dada a expressão lógica $f = a \cdot b + \overline{a} \cdot \overline{b}$, construa a tabela verdade correspondente.

Tabela verdade:

a	b	f
0	0	1
0	1	0
1	0	0
1	1	1

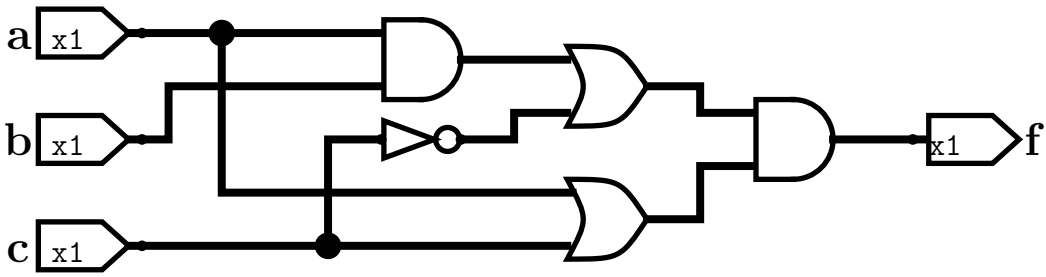
´

Prática dirigida 05

Dada a expressão lógica $f = (a \cdot b + \overline{c}) \cdot (a + c)$, desenhe o circuito combinacional correspondente.

Prática dirigida 05

Dada a expressão lógica $f = (a \cdot b + \overline{c}) \cdot (a + c)$, desenhe o circuito combinacional correspondente.



unesp

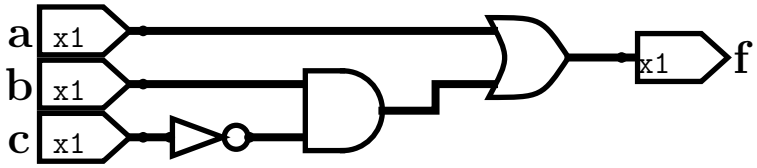
Prática dirigida 06

Dada a expressão lógica $f = a + b \cdot \overline{c}$, desenhe o circuito combinacional correspondente.

unesp

Prática dirigida 06

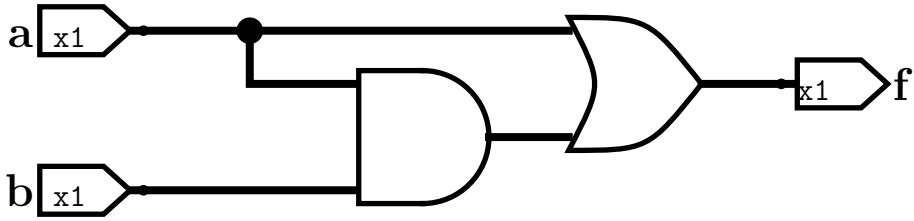
Dada a expressão lógica $f = a + b \cdot \overline{c}$, desenhe o circuito combinacional correspondente.



unesp

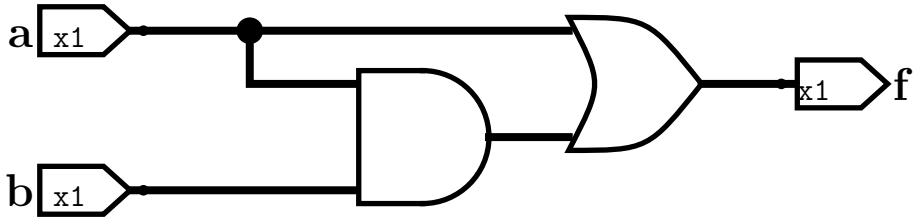
Prática dirigida 07

Dado o circuito combinacional a seguir, construa a expressão lógica e a tabela verdade correspondente:



Prática dirigida 07

Dado o circuito combinacional a seguir, construa a expressão lógica e a tabela verdade correspondente:



Expressão: $f = a.b + a$

Tabela verdade:

a	b	f
0	0	0
0	1	0
1	0	1
1	1	1

unesp

Diagrama de Tempo (*Waveform*)

unesp

Diagrama de Tempo (*Waveform*)

Quando fazemos o diagrama de tempo ou o *waveform* apresentamos ao circuito uma sequência de sinais de forma que o circuito responde a essa sequência gerando outras sequências.

- Mostra os estados do circuito ao longo do tempo.
- Permite a visualização do comportamento do circuito durante o tempo.

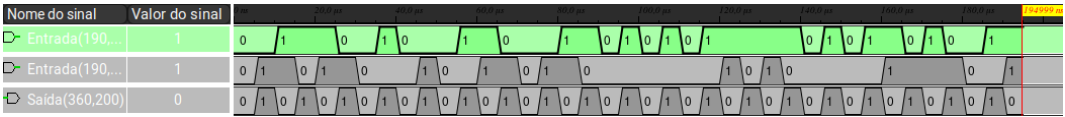


Figura: Consegue saber qual porta é esta?

unesp

Prática dirigida 01

Construção de Circuito a partir de Expressão Algébrica

- Dada a expressão algébrica $f(x, y, z) = \overline{x} \cdot y + x \cdot \overline{y} \cdot z$, construa:
  1. O circuito correspondente usando portas lógicas básicas (AND, OR, NOT).
  2. A tabela verdade que representa o comportamento da função.
  3. O diagrama de tempo (waveform) que mostra os estados do circuito para todas as combinações de x, y e z ao longo do tempo.

unesp

Prática dirigida 01

Construção de Circuito a partir de Expressão Algébrica

- Dada a expressão algébrica $f(x, y, z) = \overline{x} \cdot y + x \cdot \overline{y} \cdot z$, construa:
  1. O circuito correspondente usando portas lógicas básicas (AND, OR, NOT).
  2. A tabela verdade que representa o comportamento da função.
  3. O diagrama de tempo (waveform) que mostra os estados do circuito para todas as combinações de x, y e z ao longo do tempo.

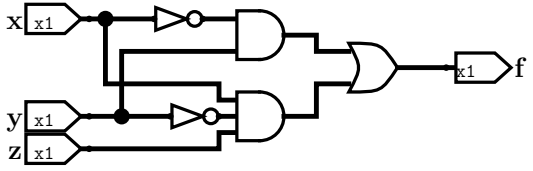
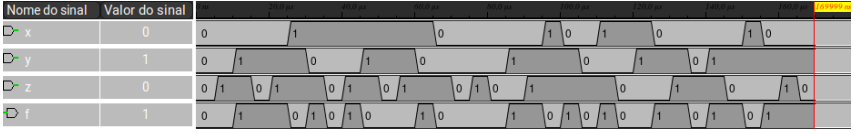


Tabela verdade:

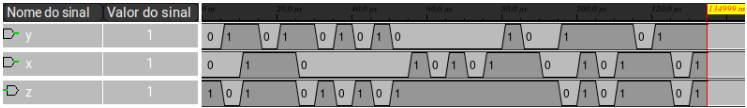
x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



unesp

Prática dirigida 02

Análise de Diagrama de Tempo

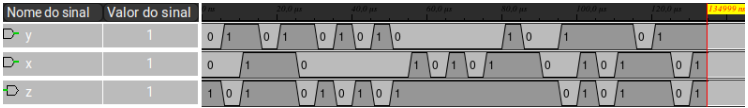


- Considerando o diagrama dado:
  1. O circuito que pode gerar o comportamento mostrado no diagrama.
  2. A tabela verdade associada ao circuito.
  3. A expressão algébrica correspondente ao circuito.

unesp

Prática dirigida 02

Análise de Diagrama de Tempo



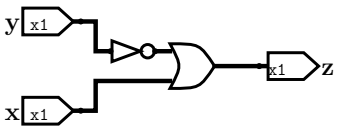
- Considerando o diagrama dado:
  1. O circuito que pode gerar o comportamento mostrado no diagrama.
  2. A tabela verdade associada ao circuito.
  3. A expressão algébrica correspondente ao circuito.

Tabela verdade:

x	y	z
0	0	1
0	1	0
1	0	1
1	1	1

Expressão:

$z = \overline{y} + x$



unesp

Mintermos e Maxtermos

Até agora, determinamos as expressões algébricas de forma intuitiva, analisando os circuitos e os resultados das tabelas verdade para, geralmente, identificar a expressão lógica correspondente ao circuito. A partir de agora, utilizaremos ferramentas que oferece um método bem definido para montar a expressão algébrica correspondente ao circuito analisado.

Tais ferramentas se chamam *Mintermos* e *Maxtermos* que, embora não produzam uma expressão otimizada, fornecem uma função inicial que pode ser manipulada e minimizada se assim o desejarmos.

- **Mintermos (Soma de Produtos Canônica (SOP)):** A função é expressa como a soma (OR) das linhas onde a função vale 1.
- **Maxtermos (Produto de Somas Canônica (POS)):** A função é expressa como o produto (AND) das linhas onde a função vale 0.

- A implementação canônica de uma função booleana que seleciona os produtos das variáveis onde o resultado é 1.
- Propriedades:
  - Unicidade: Existe uma única SOP canônica para cada função booleana.
  - Importância: Útil na análise e síntese de circuitos digitais.
  - Minimização: Pode ser simplificada com Mapas de Karnaugh ou Quine-McCluskey.

- Forma canônica que seleciona as somas das variáveis onde o resultado é 0.
- Propriedades:
  - Unicidade: Existe uma única POS canônica para cada função booleana.
  - Importância: Utilizada em design digital para implementar lógica com portas OR e AND.
  - Minimização: Pode ser simplificada como a SOP canônica.

Vamos determinar a **função lógica** a partir da tabela verdade abaixo:

a	**b**	**c**	**f(a, b, c)**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Mintermos e Maxtermos

Determinação da função lógica usando Mintermos

Mintermos: Linhas onde $f(a, b, c) = 1$.
Maxtermos Linhas onde $f(a, b, c) = 0$.

a	b	c	f(a, b, c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Passo 1: Identifique as linhas onde $f(a, b, c) = 1$.

- 02: $a = 0,\ b = 0,\ c = 1 \rightarrow$ Mintermo: $\overline{a}.\overline{b}.c$
- 04: $a = 0,\ b = 1,\ c = 1 \rightarrow$ Mintermo: $\overline{a}.bc$
- 05: $a = 1,\ b = 0,\ c = 0 \rightarrow$ Mintermo: $a\overline{b}.\overline{c}$
- 06: $a = 1,\ b = 0,\ c = 1 \rightarrow$ Mintermo: $a\overline{b}.c$

Passo 2: Escreva a função como a soma (OR) desses mintermos.

A função lógica pode ser expressa como a soma dos mintermos correspondentes:

$$f(a, b, c) = \overline{a}.\overline{b}.c + \overline{a}bc + a\overline{b}\overline{c} + ab\overline{c}$$

unesp

Mintermos e Maxtermos

Determinação da função lógica usando Maxtermos

Mintermos: Linhas onde $f(a, b, c) = 1$.
Maxtermos Linhas onde $f(a, b, c) = 0$.

a	b	c	f(a, b, c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Passo 1: Identifique as linhas onde $f(a, b, c) = 0$.

- 01: $a = 0,\ b = 0,\ c = 0 \rightarrow$ Maxtermo: $a + b + c$
- 03: $a = 0,\ b = 1,\ c = 0 \rightarrow$ Maxtermo: $a + \overline{b} + c$
- 07: $a = 1,\ b = 1,\ c = 0 \rightarrow$ Maxtermo: $\overline{a} + b + c$
- 08: $a = 1,\ b = 1,\ c = 1 \rightarrow$ Maxtermo: $\overline{a} + b + \overline{c}$

Passo 2: Escreva a função como o produto (AND) desses maxtermos.

A função lógica pode ser expressa como o produto dos maxtermos correspondentes: $f(a, b, c) =$

$$(a + b + c) \cdot (a + \overline{b} + c) \cdot (\overline{a} + b + c) \cdot (\overline{a} + b + \overline{c})$$

SOP - Prática guiada 01

- Considere a função f(a, b, c) definida pela tabela verdade:
- **tabela verdade:**

a	b	c	f(a, b, c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

- **Mintermos:**
  - $\overline{a}\overline{b}c, \overline{a}bc, a\overline{b}\overline{c}, ab\overline{c}, abc$
- **SOP Canônica:** $f(a,b,c) = \overline{a}\overline{b}c + \overline{a}bc + a\overline{b}\overline{c} + ab\overline{c} + abc$

unesp

POS - Prática guiada 02

- Considere a função f(a, b, c) definida pela tabela verdade:
- **tabela verdade:**

a	b	c	f(a, b, c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

- **Maxtermos:**
  - $(a + b + c), (a + \overline{b} + c), (\overline{a} + b + \overline{c})$
- **POS Canônica:** $f(a, b, c) = (a + b + c) \cdot (a + \overline{b} + c) \cdot (\overline{a} + b + \overline{c})$

unesp

Exercício

Criação de Circuito a partir da tabela verdade

- Dada a seguinte tabela verdade:

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

1. Construa o circuito digital que implementa essa função.
2. Derive a expressão algébrica na forma de Soma de Produtos Canônica (SOP) a partir da tabela verdade.
3. Crie o diagrama de tempo correspondente ao circuito.

unesp

Exercício - Resolução

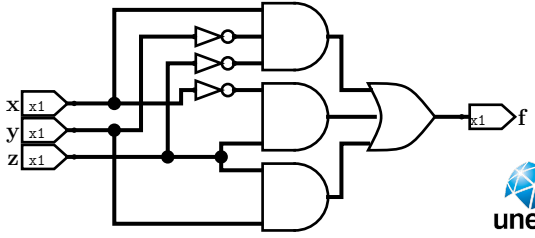
Selecionando os mintermos (Soma de Produtos Canônica)

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$f(x,y,z) = (\overline{x}y\overline{z}) + (\overline{x}yz) + (x\overline{y}\overline{z}) + (xyz)$

Agrupando
$(\overline{x}yz)+(xyz) = (yz).(\overline{x}+x) = (yz).1 = \boxed{yz}$
Agrupando yz com o restante da expressão
$\boxed{yz} + (\overline{x}y\overline{z}) + (x\overline{y}\overline{z})$.

Fatorando z: $z.(y + \overline{x}\overline{y}) + (x\overline{y}\overline{z}) =$
$z.(\overline{x} + y) + (x\overline{y}\overline{z}) = \boxed{\overline{x}z + yz + x\overline{y}\overline{z}}$



unesp

Mini prova 03

- Dado o circuito abaixo, determine:
  1. A tabela verdade correspondente ao circuito **não simplificado**.
  2. A expressão algébrica **simplificada** que representa a função lógica do circuito.
  3. O circuito simplificado.
  4. O diagrama de tempo (waveform) que reflete o comportamento do circuito.

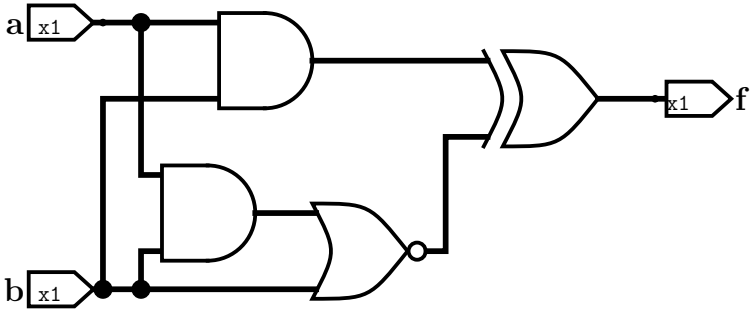


Figura: Passarinho... Resolve esse... Qual é a expressão desse?[3]

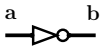
[3] *referência de velho*

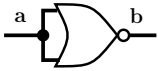
♪All you need is NAND/NOR... ♪

unesp

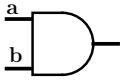
O que fazer se você não tem as portas lógicas que vc precisa

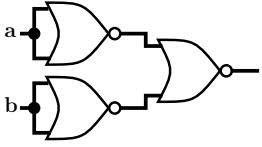
$\overline{a} = a \downarrow a = a \uparrow a$

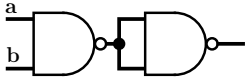




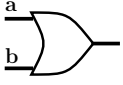
$a.b = (a \downarrow a) \downarrow (b \downarrow b) = (a \uparrow b) \uparrow (a \uparrow b)$

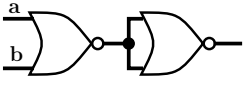


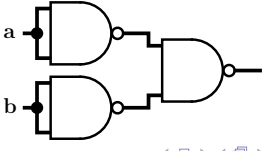




$a + b = (a \downarrow b) \downarrow (a \downarrow b) = (a \uparrow a) \uparrow (b \uparrow b)$







unesp

O que mídia não quer que você saiba sobre o XOR [4]

[4] Coringando já...

Equivalências do XOR

Até agora nos foram apresentadas as equivalências das portas AND e OR, porém, muitas vezes nos depararemos com portas XOR. Seria interessante sabermos sua equivalências também:

a	b	$a \oplus b$	$\overline{a}b + a\overline{b}$	$\overline{\overline{a}\overline{b} + ab}$	$\overline{a}$	$\overline{b}$	ab	$\overline{a}b$	$a\overline{b}$	$\overline{a}\overline{b}$
0	0	0	0	0	1	1	0	0	0	1
0	1	1	1	1	1	0	0	1	0	0
1	0	1	1	1	0	1	0	0	1	0
1	1	0	0	0	0	0	1	0	0	0

Tabela: Tabela verdade das equivalências de XOR

$$a \oplus b = \overline{a}b + a\overline{b} \; = \overline{\overline{a}\overline{b} + ab}$$

unesp

Equivalências do XOR - Exercício

Usando a tabela verdade prove que $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ e portanto que XOR é comutativo.

unesp

Revelação!

Que idade você tinha ao ficar sabendo que XOR só tem saída verdadeira quando o número de entradas verdadeiras é ímpar??

unesp

Bom... Agora vamos fazer algumas coisa legais:
Somadores e subtratores.

unesp

Como fazer a soma binária

Expressão de exemplo

Abaixo foi realizada uma operação de soma simples entre dois números binários, a e b, cujo resultado é mostrado na linha marcada com a palavra 'Sum'. 'C_{in}' e 'C_{out}' significa 'Carry in' e 'Carry out', o nosso famoso 'Vai um'. A partir deste exemplo, vamos criar uma tabela-verdade que nos permitirá formular a expressão algébrica booleana e, consequentemente, o circuito somador correspondente.

C_{out}	C_{in}			C_{in} / C_{out}
1	1			
	1	1	0	a
	1	1	1	b
1	1	0	1	Sum

Como fazer a soma binária

Tabela verdade da soma

Abaixo, representamos a tabela-verdade da soma binária. O objetivo da desta é modelar o comportamento da soma de dois bits. Como vimos na soma binária anterior, quando um bit 1 é somado a outro bit 1, o valor resultante é zero; no entanto, a próxima soma recebe um bit 1 adicional. Esse bit adicional é chamado de 'Carry in' quando é recebido, e de 'Carry out' quando é emitido. O objetivo da tabela-verdade abaixo é representar a soma e o 'Carry out' resultantes da soma de dois bits.

a	b	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabela: Tabela verdade da soma binária

unesp

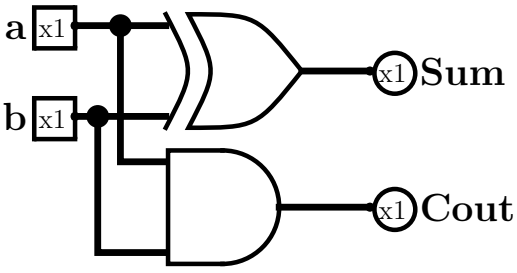
Como *Sum* e C_{out} se comportam? O que C_{out} representa em termos de uma soma computacional? Qual o circuito correspondente?

a	b	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabela: Tabela verdade da soma binária

a	b	Sum	C_{out}	$a \oplus b$	$a \cdot b$
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	1	0	1

Tabela: Tabela verdade da soma binária

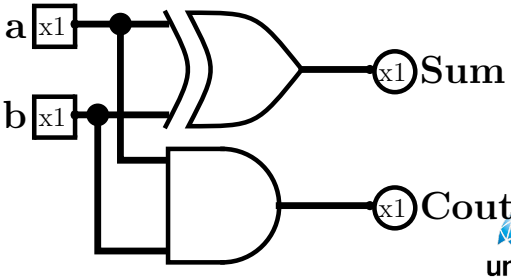


Meio somador / Somador incompleto / Half adder

A tabela-verdade e o circuito mostrados abaixo representam o que chamamos de **meio somador**, **somador incompleto** ou ainda **Half Adder**. O meio somador tem essa denominação porque, apesar de realizar a soma de dois bits, não considera um bit que possa vir para complementar sua operação. Dessa forma, o circuito é capaz de trabalhar apenas com a soma de dois bits, informando se houve um bit C_{out}. Isso impede a criação de somadores para números maiores por meio da concatenação de vários circuitos

a	b	Sum	C_{out}	$a \oplus b$	$a \cdot b$
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	1	0	1

Tabela: Tabela verdade da soma binária



unesp

Somador completo

Para determinar um somador completo, é necessário considerar não apenas as entradas a e b, mas também uma entrada C_{in}, que representa o possível bit vindo do C_{out} de outro circuito. Dessa forma, o resultado da soma é alterado, o que, por sua vez, modifica a expressão final e o circuito resultante.

a	b	C_{in}	Sum
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Tabela: Tabela verdade de um somador completo

unesp

Somador completo

Prática guiada

Considerando a tabela verdade mostrada determine a expressão algébrica boleana e o circuito correspondentes.

a	b	C_{in}	Sum
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Tabela: Tabela verdade de C_{in} para somador completo

unesp

Considerando a tabela verdade mostrada determine a expressão algébrica boleana e o circuito correspondentes.

a	b	C_{in}	Sum
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Tabela: Tabela verdade de C_{in} para somador completo

Para facilitar a notação considerando $c = C_{in}$ Temos por *mimtermos*:

$$
\begin{aligned}
&(\overline{a}\overline{b}c) + (\overline{a}b\overline{c}) + (a\overline{b}\overline{c}) + (abc) = \\
&\overline{a}.(\overline{b}c + b\overline{c}) + a.(\overline{b}\overline{c} + bc) = \\
&\overline{a}.(b \oplus c) + a.(\overline{b \oplus c}) = \\
&\boxed{a \oplus (b \oplus c)}
\end{aligned}
\tag{8}
$$

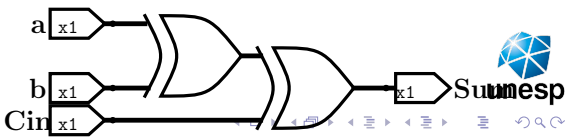
Considerando a tabela verdade mostrada determine a expressão algébrica boleana e o circuito correspondentes.

a	b	C_{in}	Sum
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Tabela: Tabela verdade de C_{in} para somador completo

Para facilitar a notação considerando $c = C_{in}$ Temos por *mimtermos*:

$$
\begin{aligned}
&(\overline{a}\overline{b}c) + (\overline{a}b\overline{c}) + (a\overline{b}\overline{c}) + (abc) = \\
&\overline{a}.(\overline{b}c + b\overline{c}) + a.(\overline{b}\overline{c} + bc) = \\
&\overline{a}.(b \oplus c) + a.(\overline{b \oplus c}) = \\
&\boxed{a \oplus (b \oplus c)}
\end{aligned}
\tag{8}
$$



Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	

Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	

Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	

Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	

Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	

unesp

Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	

unesp

Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	

unesp

Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	

unesp

Somador completo

Um somador completo deve não apenas receber um C_{in} como também gerar um C_{out}. Portanto, é necessário determinar como o C_{out} é obtido.

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tabela: Tabela verdade de C_{out} para somador completo

unesp

A partir da tabela-verdade mostrada determine a expressão algébrica booleana correspondente e o respectivo circuito de C_{out}[5].

a	b	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tabela: Tabela-verdade de C_{out} para somador completo

Para facilitar a notação considerando $c = C_{in}$ temos por *mimtermos*:

$$(\bar{a}bc) + (a\bar{b}c) + (ab\bar{c}) + (abc) = \quad (9)$$

[5]boa sorte...

Somador completo

Prática dirigida

$$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$$

Somador completo

Prática dirigida

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

Somador completo

Prática dirigida

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

pondo **a** em evidência:

$a.(\overline{b}c + b\overline{c} + bc) + (\overline{a}bc) =$

Somador completo

Prática dirigida

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

pondo **a** em evidência:

$a.(\overline{b}c + b\overline{c} + bc) + (\overline{a}bc) =$

pondo **c** em evidência

$a.(c.(\overline{b}b) + b\overline{c}) + (\overline{a}bc) =$

unesp

Somador completo

Prática dirigida

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

pondo **a** em evidência:

$a.(\overline{b}c + b\overline{c} + bc) + (\overline{a}bc) =$

pondo **c** em evidência

$a.(c.(\overline{b}b) + b\overline{c}) + (\overline{a}bc) =$

$a.(c.1 + b\overline{c}) + (\overline{a}bc) =$

unesp

Somador completo

Prática dirigida

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

pondo **a** em evidência:

$a.(\overline{b}c + b\overline{c} + bc) + (\overline{a}bc) =$

pondo **c** em evidência

$a.(c.(\overline{b}b) + b\overline{c}) + (\overline{a}bc) =$

$a.(c.1 + b\overline{c}) + (\overline{a}bc) =$

$a.(c + b\overline{c}) + (\overline{a}bc) =$

unesp

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

pondo **a** em evidência:

$a.(\overline{b}c + b\overline{c} + bc) + (\overline{a}bc) =$

pondo **c** em evidência

$a.(c.(\overline{b}b) + b\overline{c}) + (\overline{a}bc) =$

$a.(c.1 + b\overline{c}) + (\overline{a}bc) =$

$a.(c + b\overline{c}) + (\overline{a}bc) =$

(10)

usando o teorema:$x + \overline{x}y = x + y$

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

pondo **a** em evidência:

$a.(\overline{b}c + b\overline{c} + bc) + (\overline{a}bc) =$

pondo **c** em evidência

$a.(c.(\overline{b}b) + b\overline{c}) + (\overline{a}bc) =$

$a.(c.1 + b\overline{c}) + (\overline{a}bc) =$

$a.(c + b\overline{c}) + (\overline{a}bc) =$

(10)

usando o teorema:$x + \overline{x}y = x + y$

$a.(c + b) + (\overline{a}bc) =$

$ac + ab + \overline{a}bc =$

colocando **b** em evidência:

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

pondo **a** em evidência:

$a.(\overline{b}c + b\overline{c} + bc) + (\overline{a}bc) =$

pondo **c** em evidência

$a.(c.(\overline{b}b) + b\overline{c}) + (\overline{a}bc) =$

$a.(c.1 + b\overline{c}) + (\overline{a}bc) =$

$a.(c + b\overline{c}) + (\overline{a}bc) =$

(10)

usando o teorema:$x + \overline{x}y = x + y$

$a.(c + b) + (\overline{a}bc) =$

$ac + ab + \overline{a}bc =$

colocando **b** em evidência:

$ac + b.(a + \overline{a}c) =$

usando o teorema:$x + \overline{x}y = x + y$

$(\overline{a}bc) + (a\overline{b}c) + (ab\overline{c}) + (abc) =$

reorganizando:

$(a\overline{b}c) + (ab\overline{c}) + (abc) + (\overline{a}bc) =$

pondo **a** em evidência:

$a.(\overline{b}c + b\overline{c} + bc) + (\overline{a}bc) =$

pondo **c** em evidência

$a.(c.(\overline{b}b) + b\overline{c}) + (\overline{a}bc) =$

$a.(c.1 + b\overline{c}) + (\overline{a}bc) =$

$a.(c + b\overline{c}) + (\overline{a}bc) =$

(10)

usando o teorema:$x + \overline{x}y = x + y$

$a.(c + b) + (\overline{a}bc) =$

$ac + ab + \overline{a}bc =$

colocando **b** em evidência:

$ac + b.(a + \overline{a}c) =$

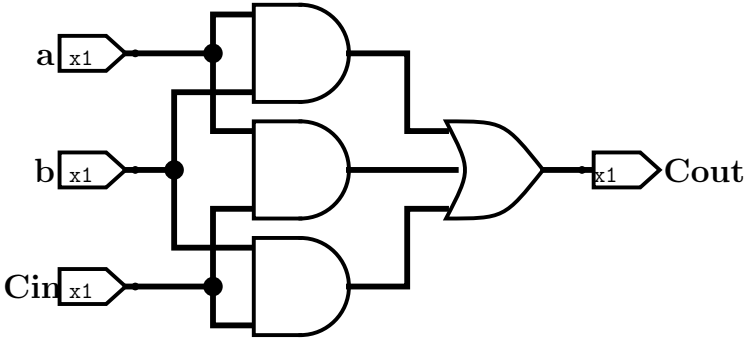
usando o teorema:$x + \overline{x}y = x + y$

$ac + b.(a + c) =$

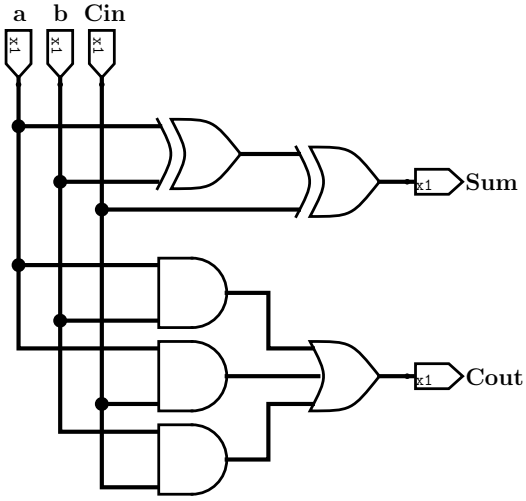
$\boxed{ac + ba + bc}$

(11)

Portanto o circuito do C_{out} fica assim:

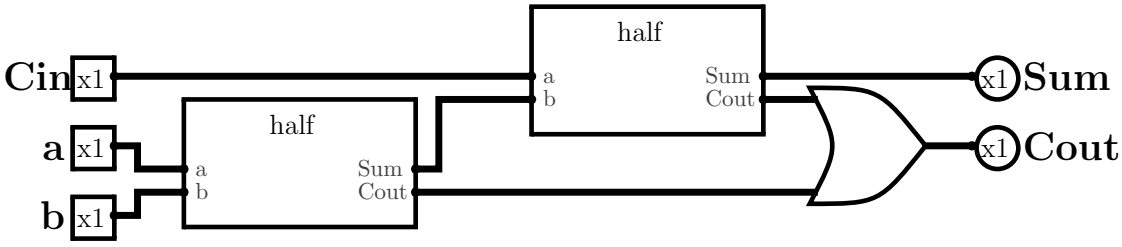


Portanto o circuito **somador completo** ou **Full adder** fica assim:



Somador completo - Gambiarra

Usando somadores incompletos podemos fazer um completo:



Subtração binária

Expressão de exemplo

Abaixo foi realizada uma operação de subtração simples entre dois números binários, a e b, cujo resultado é mostrado na linha marcada com a palavra 'Diff'. 'B_{out}' significa 'Borrow out', o nosso famoso 'emprestar'. A partir deste exemplo, vamos criar uma tabela-verdade que nos permitirá formular a expressão algébrica booleana e, consequentemente, o circuito subtrator correspondente.

	10	10	10	B_{out}
1	1	0	0	a
	1	1	1	b
	1	0	1	$Diff$

Meio subtrator / Subtrator incompleto / Half subtractor

A tabela-verdade e o circuito mostrados abaixo representam o que chamamos de **meio subtrator**, **subtrator incompleto** ou ainda **Half subtrator**. O meio subtrator tem essa denominação porque, apesar de realizar a subtração de dois bits, não considera um bit que possa ter sido emprestado por uma operação de subtração anterior. Dessa forma, o circuito é capaz de trabalhar apenas com a subtração de dois bits, informando se houve necessidade de empréstimo de um bit B_{out}. Isso impede a criação de subtratores para números maiores por meio da concatenação de vários circuitos

a	b	Diff	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tabela: Tabela verdade da diferença binária

unesp

Como *Diff* e B_{out} se comportam? Qual o circuito correspondente?

a	b	Diff	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tabela: Tabela verdade da diferença binária

Meio subtrator / Subtrator incompleto / Half subtractor

Prática dirigida - Tabela verdade da subtração

a	b	Diff	B_{out}	$a \oplus b$	$\overline{a}.b$
0	0	0	0	0	0
0	1	1	1	1	1
1	0	1	0	1	0
1	1	0	0	0	0

Tabela: Tabela verdade da diferença binária

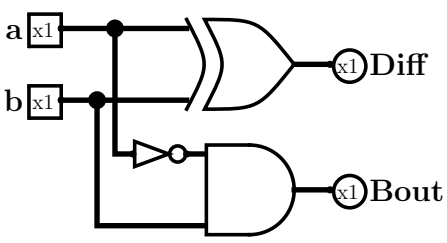
unesp

Meio subtrator / Subtrator incompleto / Half subtractor

Prática dirigida - Tabela verdade da subtração

a	b	Diff	B_{out}	$a \oplus b$	$\overline{a}.b$
0	0	0	0	0	0
0	1	1	1	1	1
1	0	1	0	1	0
1	1	0	0	0	0

Tabela: Tabela verdade da diferença binária



Subtrator completo

Para determinar um subtrator completo, é necessário considerar não apenas as entradas a e b, mas também uma entrada B_{in}, que representa o possível bit subtrator vindo do B_{out} de outro circuito. Dessa forma, o resultado da subtração é alterado, o que, por sua vez, modifica a expressão final e o circuito resultante.
Tenha em mente que $Diff = a - b - B_{in}$.

a	b	B_{in}	Diff	B_{out}
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

Tabela: Tabela verdade de um subtrator completo

unesp

Subtrator completo

Prática guiada

Considerando a tabela verdade mostrada determine as expressões algébricas boleanas e os circuitos de *Diff* e B_{out} correspondentes.

a	b	B_{in}	Diff	B_{out}
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

Tabela: Tabela verdade de B_{in} para subtrator completo

unesp

Subtrator completo

Prática guiada

Considerando a tabela verdade mostrada determine as expressões algébricas boleanas e os circuitos de *Diff* e B_{out} correspondentes.

a	b	B_{in}	Diff	B_{out}
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

Tabela: Tabela verdade de B_{in} para subtrator completo

Se você olhou bem, economizou tempo!
Pois $Diff = a \oplus b \oplus c$ $\therefore$

$$\boxed{Diff = a \oplus b \oplus B_{in}}$$

unesp

Subtrator completo

Prática guiada

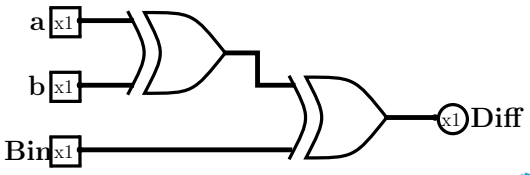
Considerando a tabela verdade mostrada determine as expressões algébricas boleanas e os circuitos de *Diff* e B_{out} correspondentes.

a	b	B_{in}	Diff	B_{out}
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

Tabela: Tabela verdade de B_{in} para subtrator completo

Se você olhou bem, economizou tempo!
Pois $Diff = a \oplus b \oplus c \therefore$

$$\boxed{Diff = a \oplus b \oplus B_{in}}$$



unesp

Subtrator completo

Prática dirigida

A partir da tabela-verdade mostrada determine a expressão algébrica booleana correspondente e o respectivo circuito de B_{out}.

a	b	B_{in}	Diff	B_{out}
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

Tabela: Tabela-verdade de B_{out} para subtrator completo

Para facilitar a notação considerando $c = B_{in}$ temos por *mimtermos*:

$$(\bar{a}\bar{b}c) + (\bar{a}b\bar{c}) + (\bar{a}bc) + (abc) \tag{12}$$

unesp

Subtrator completo

Prática dirigida

$(\overline{a}\overline{b}c) + (\overline{a}b\overline{c}) + (\overline{a}bc) + (abc) =$

pondo em evidência $\overline{a}$:

$\overline{a}.(\overline{b}c + b\overline{c} + bc) + (abc) =$

$\overline{a}.(\overline{b}c + b.(\overline{c} + c)) + (abc) =$

$\overline{a}.(\overline{b}c + b.1) + (abc) =$ (13)

$\overline{a}.(\overline{b}c + b) + (abc) =$

usando o teorema:$x + \overline{x}y = x + y$

$\overline{a}.(b + c) + (abc) =$

$\overline{a}b + \overline{a}c + abc =$

$\overline{a}b + \overline{a}c + abc =$

fatorando c :

$\overline{a}b + c.(\overline{a} + ab) =$

usando o teorema:$x + \overline{x}y = x + y$ (14)

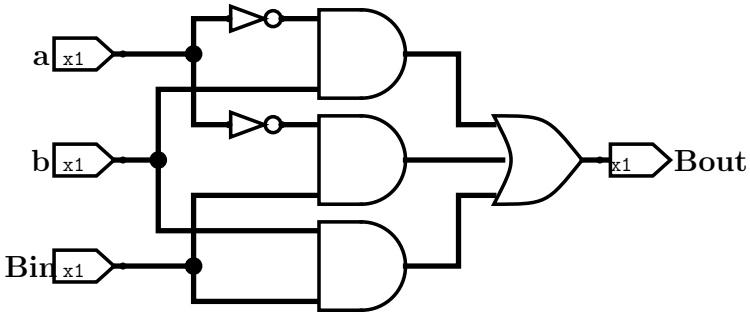
$\overline{a}b + c.(\overline{a} + b) =$

$\overline{a}b + \overline{a}c + bc \therefore$

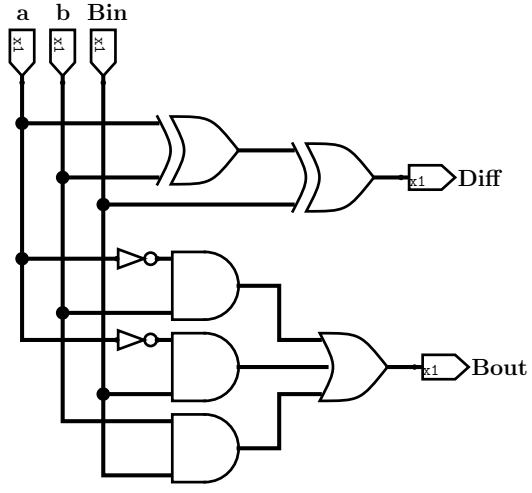
$\boxed{B_{out} = \overline{a}b + \overline{a}B_{in} + bB_{in}}$

unesp

Portanto o circuito do B_{out} fica assim:



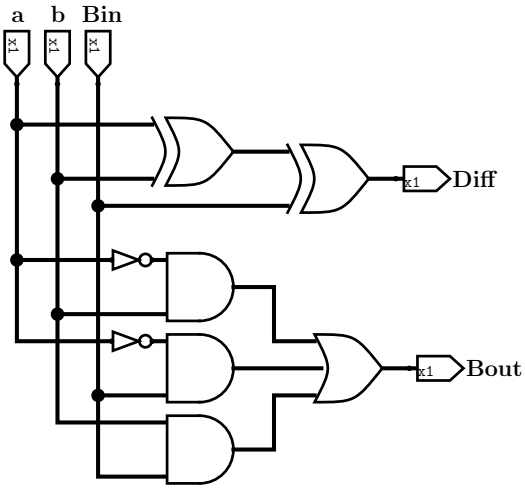
Portanto o circuito **subtrator completo** ou **Full subtractor** fica assim:

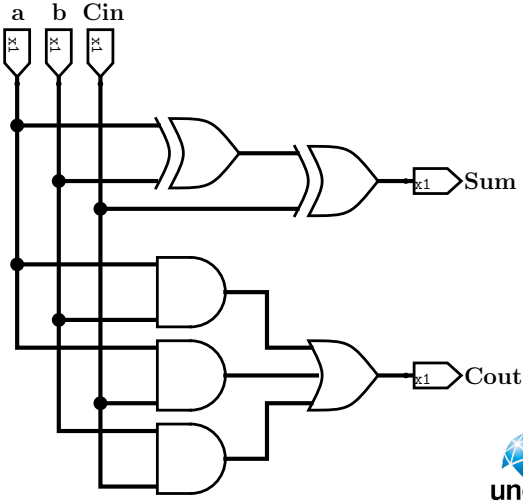


Mas... mas... *Oh... wait a fucking moment!*

unesp

KKKKKKKK





unesp

Somador completo

Demonstração

Vamos fazer um somador completo no logisim.

Mini prova 04

Altere o circuito somador completo para que o mesmo tenha um modo somador e um modo subtrator.

unesp