

## Circuitos Lógicos

Em meados do século XIX George Boole, um matemático inglês, desenvolveu uma teoria completamente diferente para a época. Ela era baseada em uma série de postulados e operações simples para se resolver uma infinidade de problemas.

No entanto, se bem que a Álgebra de Boole, como foi chamada, pode resolver problemas práticos de controle, fabricação de produtos, instrumentação, etc., na época não havia eletrônica e nem mesmo as máquinas eram avançadas o suficiente para utilizar seus princípios.

A Álgebra de Boole veio a se tornar importante com o advento da Eletrônica, especificamente da eletrônica digital que gerou os modernos computadores, equipamentos digitais e de telecomunicações com que estamos familiarizados.

Boole estabelece em sua teoria que só existem no universo duas condições possíveis ou estados, para a análise de qualquer situação, e estes dois estados são opostos.

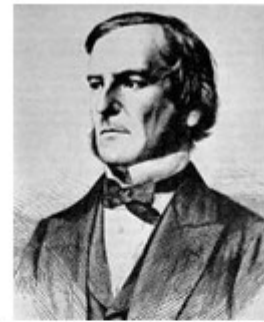
Assim, uma lâmpada só pode estar acesa ou apagada; uma torneira só pode estar aberta ou fechada; uma fonte só pode ter ou não ter tensão na sua saída; uma pergunta só pode ter como resposta verdadeiro ou falso (sim ou não).

Dizemos de uma maneira simples que, na Álgebra de Boole, as variáveis lógicas só podem adquirir dois estados:

- 0 ou 1
- Verdadeiro ou Falso
- Aberto ou Fechado
- Alto ou Baixo (HI ou LO)
- Ligado ou Desligado

Fonte: < <https://www.newtoncbraga.com.br/index.php/eletronica-digital/16291-curso-de-eletronica-digital-a-algebra-de-boole-cur5002.html> >

Em meados do século XX, o americano Claude Elwood Shannon sugeriu que a Álgebra Booleana poderia ser usada para análise e projeto de circuitos de computação. Shannon é famoso por ter fundado a teoria da informação com um artigo publicado em 1948. Mas a ele também é creditada a fundação tanto do computador digital quanto do projeto de circuito digital em 1937, quando, com 21 anos de idade e mestrando no MIT, escreveu uma tese demonstrando que uma aplicação elétrica utilizando álgebra booleana poderia resolver qualquer problema de lógica.



George Boole (1815-1864)



Claude Elwood Shannon (1916-2001)

Ao estudar os complexos circuitos ad hoc do analisador diferencial, Shannon viu que os conceitos de George Boole, inventor da álgebra booleana, poderiam ser úteis para várias coisas. Um documento elaborado a partir da sua tese de mestrado em 1937.

Fonte: < [https://pt.wikipedia.org/wiki/Claude\\_Shannon](https://pt.wikipedia.org/wiki/Claude_Shannon) >

Nos primórdios da eletrônica, todos os problemas eram solucionados por meio de sistemas analógicos. Com o avanço da tecnologia, os problemas passaram a ser solucionados pela eletrônica digital, porém na eletrônica digital, os sistemas (computadores, processadores de dados, sistemas de controle, codificadores, decodificadores, etc) empregam um pequeno grupo de circuitos lógicos básicos, que são conhecidos como portas e, ou, não e flip-flop.

Com a utilização adequadas dessas portas é possível implementar todas as expressões geradas pela álgebra de Boole. Na álgebra de Boole, há somente dois estados (valores ou símbolos) permitidos:

- Estado 0 (zero)
- Estado 1 (um)

O estado zero representa não, falso, aparelho desligado, ausência de tensão, chave elétrica desligada, etc, já o estado um representa sim, verdadeiro, aparelho ligado, presença de tensão, chave ligada, etc. Na, na álgebra booleana, se representarmos por 0 uma situação, a situação contrária é representada por 1, portanto, em qualquer bloco (porta ou função) lógico somente esses dois estados (0 ou 1) são permitidos em suas entradas e saídas.

Dito isso uma variável booleana também só assume um dos dois estados permitidos (0 ou 1)

Levando em consideração o descrito, trataremos dos seguintes blocos lógicos

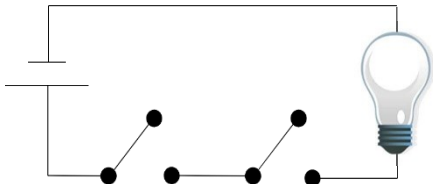
- E (AND)
- OU (OR)
- NÃO (NOT)
- NÃO E (NAND)
- NÃO OU (NOR)
- OU EXCLUSIVO (XOR)

Após, veremos a correspondência entre expressões, circuitos e tabelas verdade e por último, veremos a equivalência entre blocos lógicos.

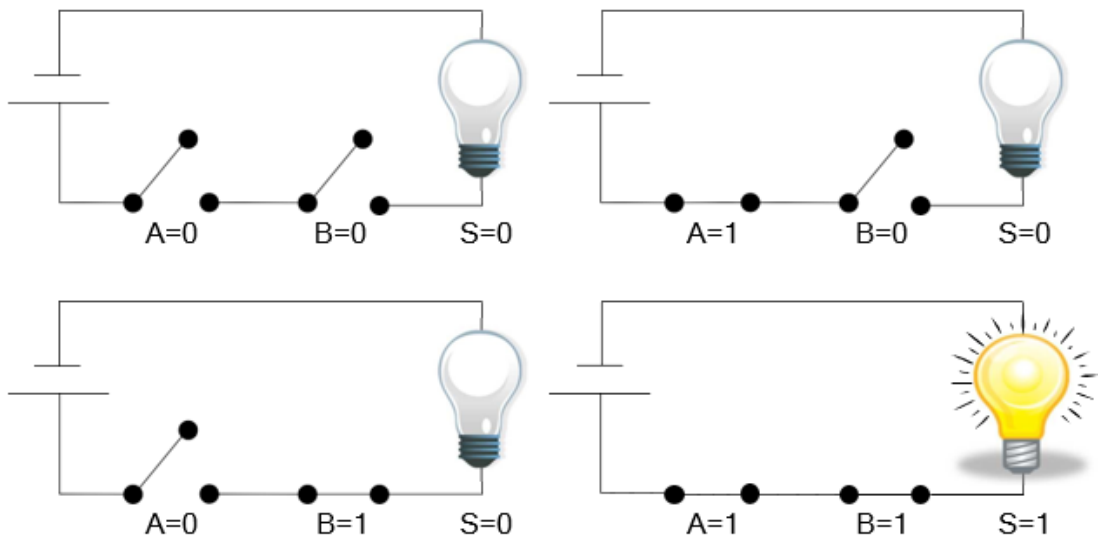
## Função E (AND)

Executa a multiplicação (conjunção) booleana de duas ou mais variáveis binárias. Por exemplo, assumamos a convenção no circuito:

- Chave aberta = 0; Chave fechada = 1
- Lâmpada apagada = 0; Lâmpada acesa = 1



Teremos as situações possíveis:



- Se a chave A está aberta ( $A=0$ ) e a chave B aberta ( $B=0$ ), não haverá circulação de energia no circuito, logo a lâmpada fica apagada ( $S=0$ )
- Se a chave A está fechada ( $A=1$ ) e a chave B aberta ( $B=0$ ), não haverá circulação de energia no circuito, logo a lâmpada fica apagada ( $S=0$ )
- Se a chave A está aberta ( $A=0$ ) e a chave B fechada ( $B=1$ ), não haverá circulação de energia no circuito, logo a lâmpada fica apagada ( $S=0$ )
- Se a chave A está fechada ( $A=1$ ) e a chave B fechada ( $B=1$ ), haverá circulação de energia no circuito e a lâmpada fica acesa ( $S=1$ )

Observando todas as quatro situações possíveis (interpretações), é possível concluir que a lâmpada fica acesa somente quando as chaves A e B estiverem simultaneamente fechadas ( $A=1$  e  $B=1$ ).

Para representar a expressão

$$S = A \text{ e } B$$

Adotaremos a representação:

$$S = A.B, \text{ onde se lê } S = A \text{ e } B$$

Porém, existem notações alternativas:

$$S = A \& B$$

$$S = A, B$$

$$S = A \wedge B$$

### **Tabela Verdade da Função E (AND)**

A tabela verdade é um mapa onde são colocadas todas as possíveis interpretações (situações), com seus respectivos resultados para uma expressão booleana qualquer. Como visto no exemplo anterior, para 2 variáveis booleanas (A e B), há 4 interpretações possíveis. Em geral, para N variáveis booleanas de entrada, há  $2^N$  interpretações possíveis.

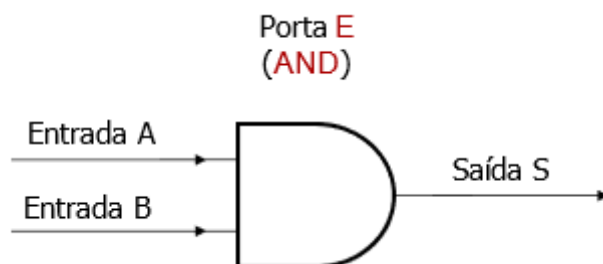
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

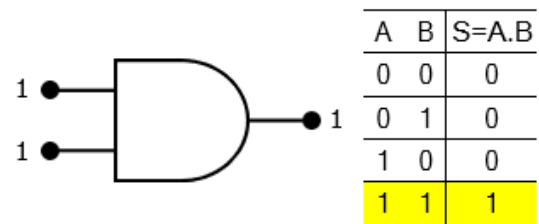
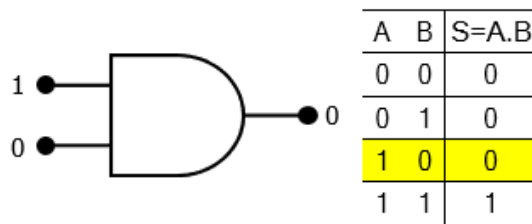
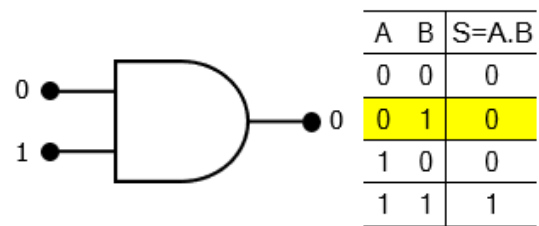
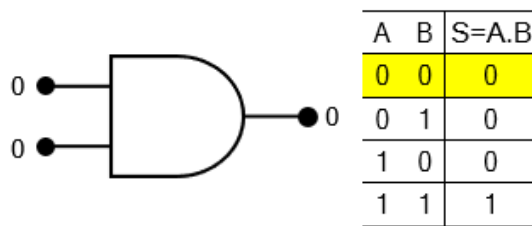
### **Porta Lógica E (AND)**

A porta E é um circuito que executa a função E

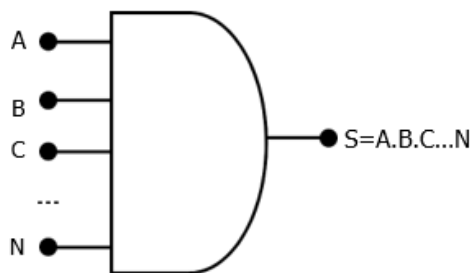
A porta E executa a tabela verdade da função E

Portanto, a saída será 1 somente se ambas as entradas forem iguais a 1; nos demais casos, a saída será 0. Seja a representação:

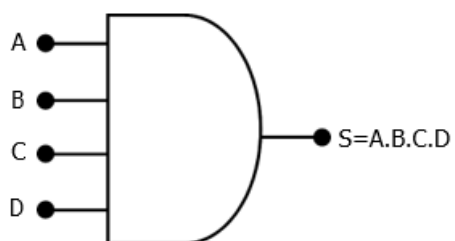




É possível estender o conceito de uma porta E para um número qualquer de variáveis de entrada A. Nesse caso, temos uma porta E com N entradas e somente uma saída. A saída será 1 se e somente se as N entradas forem iguais a 1, nos demais casos, a saída será 0.



Por exemplo,  $S=A.B.C.D$



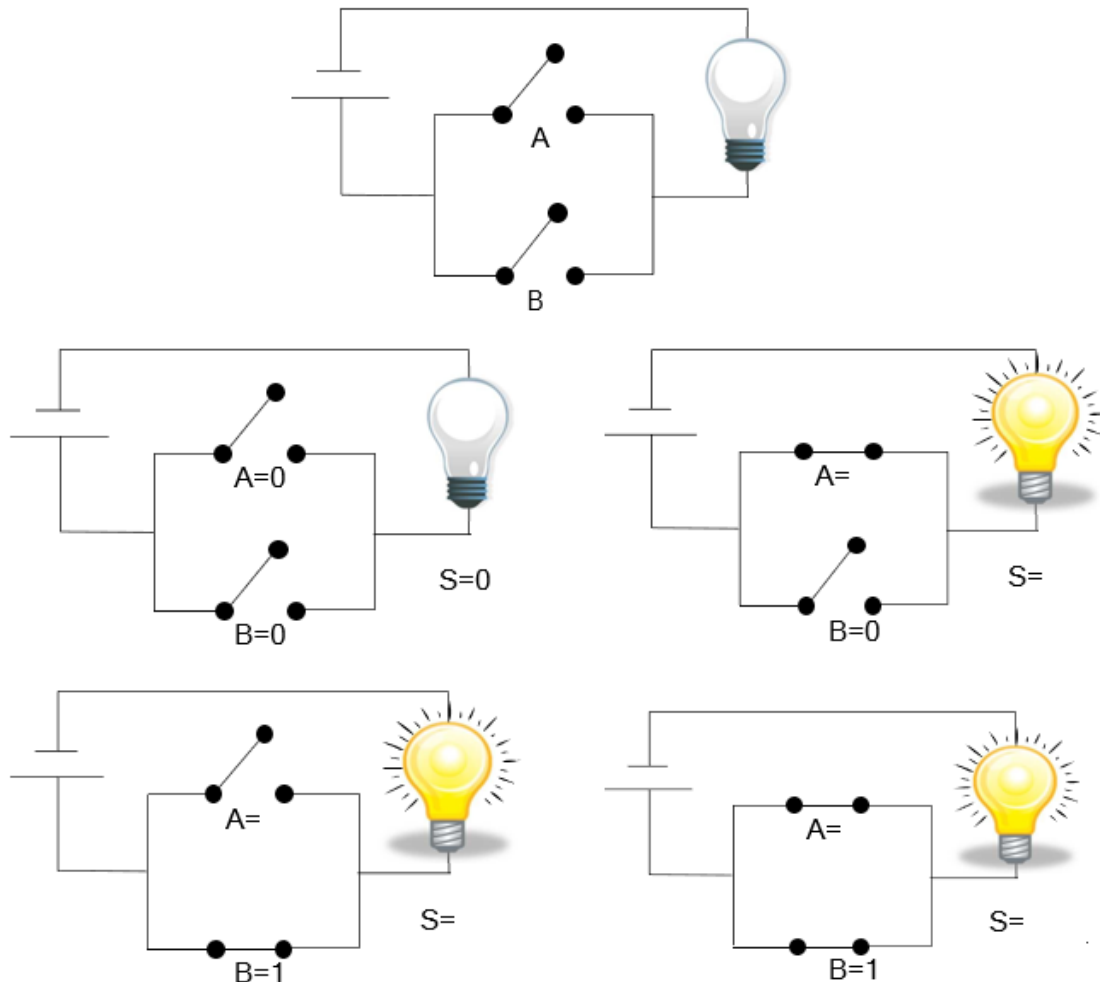
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

## Função OU (OR)

Executa a soma (disjunção) booleana de duas ou mais variáveis binárias

Por exemplo, assuma a convenção no circuito:

- Chave aberta = 0; Chave fechada = 1
- Lâmpada apagada = 0; Lâmpada acesa = 1



- Se a chave A está aberta ( $A=0$ ) e a chave B aberta ( $B=0$ ), não haverá circulação de energia no circuito, logo a lâmpada fica apagada ( $S=0$ )
- Se a chave A está fechada ( $A=1$ ) e a chave B aberta ( $B=0$ ), haverá circulação de energia no circuito e a lâmpada fica acesa ( $S=1$ )
- Se a chave A está aberta ( $A=0$ ) e a chave B fechada ( $B=1$ ), haverá circulação de energia no circuito e a lâmpada fica acesa ( $S=1$ )
- Se a chave A está fechada ( $A=1$ ) e a chave B fechada ( $B=1$ ), haverá circulação de energia no circuito e a lâmpada fica acesa ( $S=1$ )

Observando todas as quatro situações possíveis, é possível concluir que a lâmpada fica acesa somente quando a chave A ou a chave B ou ambas estiverem fechadas

Para representar a expressão

$$S = A \text{ ou } B$$

Adotaremos a representação

$$S = A+B, \text{ onde se lê } S = A \text{ ou } B$$

Porém, existem notações alternativas

$$S = A \mid B$$

$$S = A; B$$

$$S = A \vee B$$

### **Tabela Verdade da Função OU (OR)**

Observe que, no sistema de numeração binário, a soma  $1+1=10$ . Na álgebra booleana,  $1+1=1$ , já que somente dois valores são permitidos (0 e 1)

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

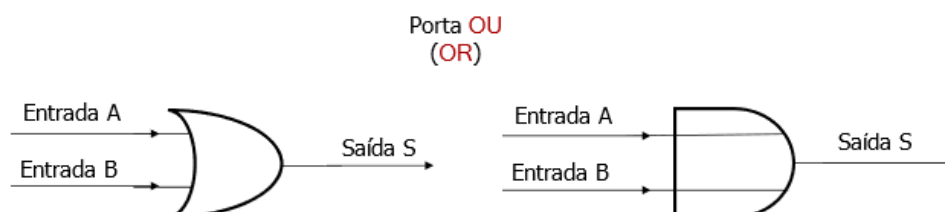
### **Porta Lógica OU (OR)**

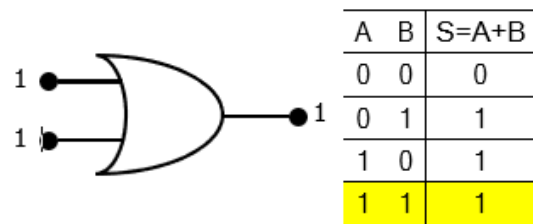
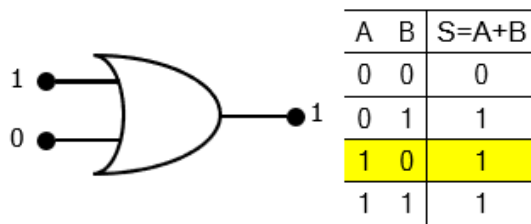
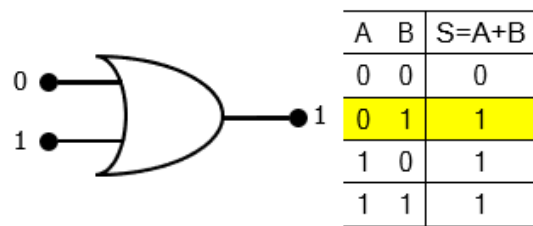
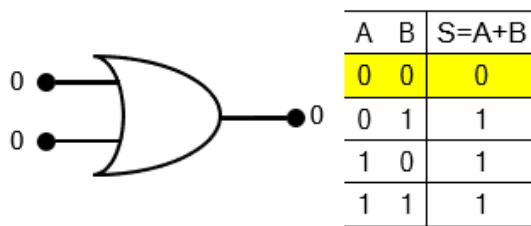
A porta OU é um circuito que executa a função OU

A porta OU executa a tabela verdade da função OU.

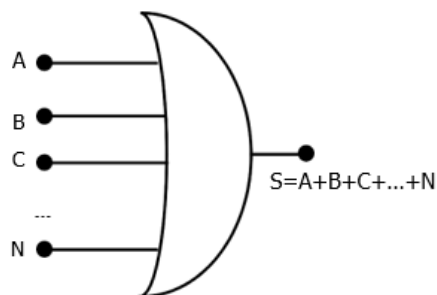
Portanto, a saída será 0 somente se ambas as entradas forem iguais a 0, nos demais casos, a saída será 1.

Representação

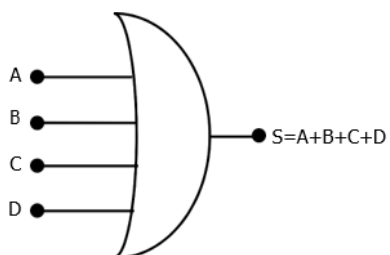




É possível estender o conceito de uma porta OU para um número qualquer de variáveis de entrada. Nesse caso, temos uma porta OU com N entradas e somente uma saída. A saída será 0 se e somente se as N entradas forem iguais a 0, nos demais casos, a saída será 1.



Por exemplo,  $S=A+B+C+D$



A	B	C	D	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



## Função NÃO (NOT)

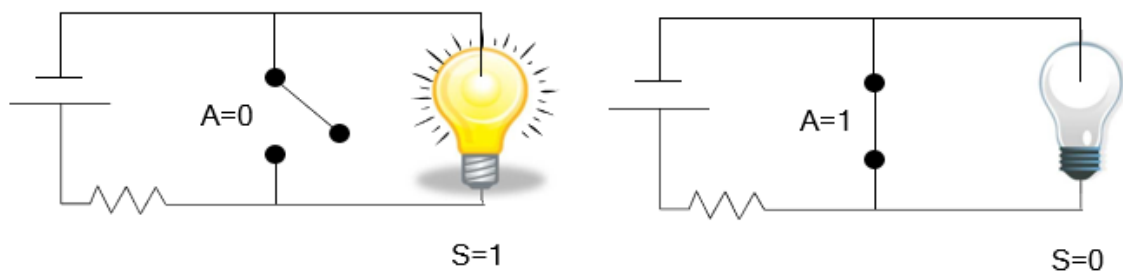
Executa o complemento (negação) de uma variável binária

- Se a variável estiver em 0, o resultado da função é 1
- Se a variável estiver em 1, o resultado da função é 0

Essa função também é chamada de inversora. Usando as mesmas convenções dos circuitos anteriores, tem-se que:

Quando a chave A está aberta ( $A=0$ ), passará corrente pela lâmpada e ela acenderá ( $S=1$ )

Quando a chave A está fechada ( $A=1$ ), a lâmpada estará em curto-circuito e não passará corrente por ela, ficando apagada ( $S=0$ )



Para representar a expressão

$$S = \text{não } A$$

Adotaremos a representação

$$S = \bar{A}, \text{ onde se lê } S = \text{não } A$$

Notações alternativas

$$S = A'$$

$$S = \neg A$$

$$S = \tilde{A} \text{ ou } \sim A$$

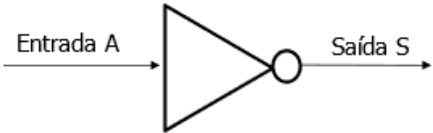
## Tabela verdade da função NÃO (NOT)

A	$\bar{A}$
0	1
1	0

A porta lógica NÃO, ou inversor, é o circuito que executa a função NÃO. O inversor executa a tabela verdade da função NÃO.

Se a entrada for 0, a saída será 1; se a entrada for 1, a saída será 0

Representação



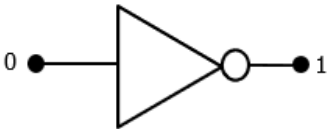
Alternativamente,



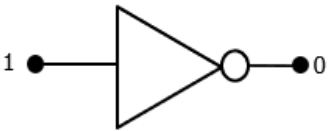
Após um  
bloco lógico



Antes de um  
bloco lógico



A	$S = \bar{A}$
0	1
1	0



A	$S = \bar{A}$
0	1
1	0

## **Função NÃO E (NAND)**

Composição da função E com a função NÃO, ou seja, a saída da função E é invertida

$$S = (\overline{A.B}) = \overline{A.B}$$

$$= (A.B)'$$

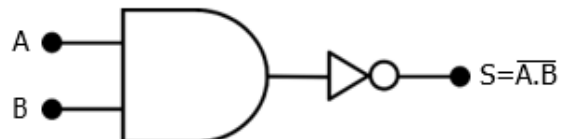
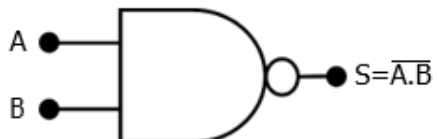
$$= \neg(A.B)$$

## **Tabela verdade**

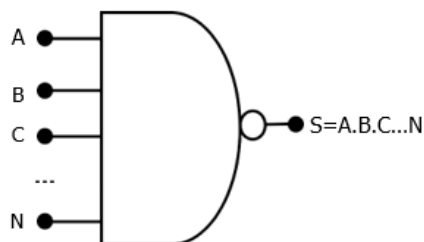
A	B	$S = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

A porta NÃO E (NE) é o bloco lógico que executa a função NÃO E, ou seja, sua tabela verdade

Representação



Como a porta E, a porta NÃO E pode ter duas ou mais entradas, nesse caso, temos uma porta NÃO E com N entradas e somente uma saída. A saída será 0 se e somente se as N entradas forem iguais a 1, nos demais casos, a saída será 1.



## Função NÃO OU (NOR)

Composição da função OU com a função NÃO, ou seja, a saída da função OU é invertida:

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

$$= (A + B)'$$

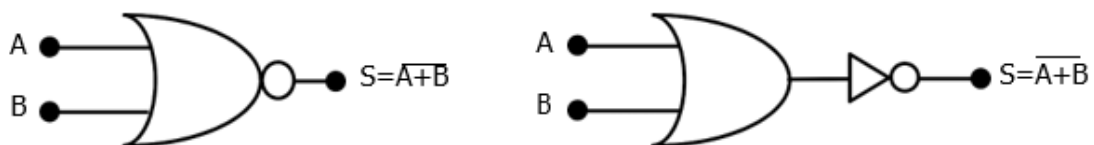
$$= \neg(A + B)$$

## Tabela verdade

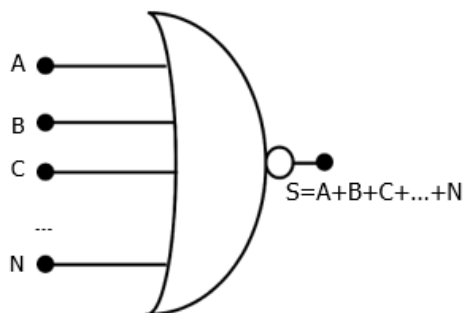
A	B	$S = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

A porta NÃO OU (NOU) é o bloco lógico que executa a função NÃO OU, ou seja, sua tabela verdade.

Representação



Como a porta OU, a porta NÃO OU pode ter duas ou mais entradas. Nesse caso, temos uma porta NÃO OU com N entradas e somente uma saída. A saída será 1 se e somente se as N entradas forem iguais a 0, nos demais casos, a saída será 0.



## **Função OU EXCLUSIVO (XOR)**

A função OU Exclusivo fornece:

- 1 na saída quando as entradas forem diferentes entre si e
- 0 caso contrário

$$S = A \oplus B$$

$$= \bar{A}.B + A.\bar{B}$$

### **Tabela verdade**

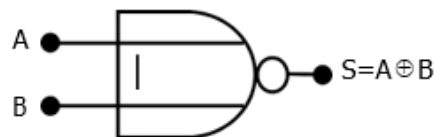
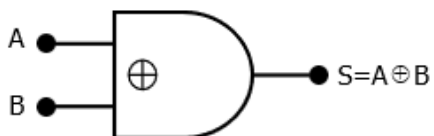
A	B	S=A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

### **Simbologia adotada**


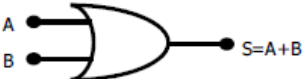
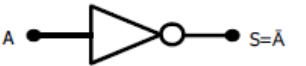
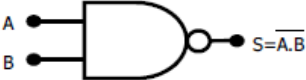

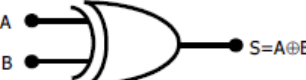
Simbologia adotada



Outros símbolos utilizados

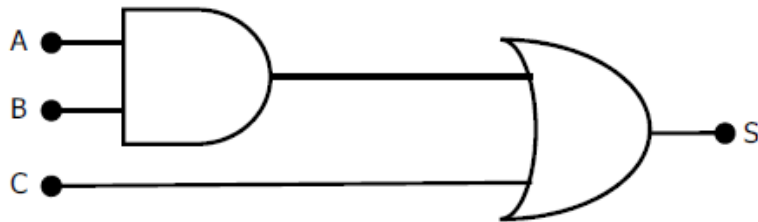


## Resumo dos Blocos Lógico Básicos

Nome	Símbolo Gráfico	Função Algébrica	Tabela Verdade															
E (AND)		$S=A \cdot B$ $S=AB$	<table><tr><th>A</th><th>B</th><th>S=A.B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S=A.B	0	0	0	0	1	0	1	0	0	1	1	1
A	B	S=A.B																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OU (OR)		$S=A+B$	<table><tr><th>A</th><th>B</th><th>S=A+B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S=A+B	0	0	0	0	1	1	1	0	1	1	1	1
A	B	S=A+B																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NÃO (NOT) Inversor		$S=\bar{A}$ $S=A'$ $S=\neg A$	<table><tr><th>A</th><th>S=<math>\bar{A}</math></th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	S= $\bar{A}$	0	1	1	0									
A	S= $\bar{A}$																	
0	1																	
1	0																	
NE (NAND)		$S=\overline{A \cdot B}$ $S=(A \cdot B)'$ $S=\neg(A \cdot B)$	<table><tr><th>A</th><th>B</th><th>S=<math>\overline{A \cdot B}</math></th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S= $\overline{A \cdot B}$	0	0	1	0	1	1	1	0	1	1	1	0
A	B	S= $\overline{A \cdot B}$																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOU (NOR)		$S=\overline{A+B}$ $S=(A+B)'$ $S=\neg(A+B)$	<table><tr><th>A</th><th>B</th><th>S=<math>\overline{A+B}</math></th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S= $\overline{A+B}$	0	0	1	0	1	0	1	0	0	1	1	0
A	B	S= $\overline{A+B}$																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR		$S=A \oplus B$	<table><tr><th>A</th><th>B</th><th>S=A⊕B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S=A⊕B	0	0	0	0	1	1	1	0	1	1	1	0
A	B	S=A⊕B																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

## Exemplos

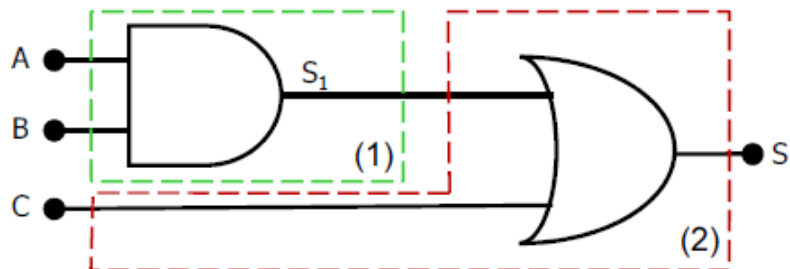
1. Seja o circuito:



Qual é a expressão lógica booleana associada a esse circuito?

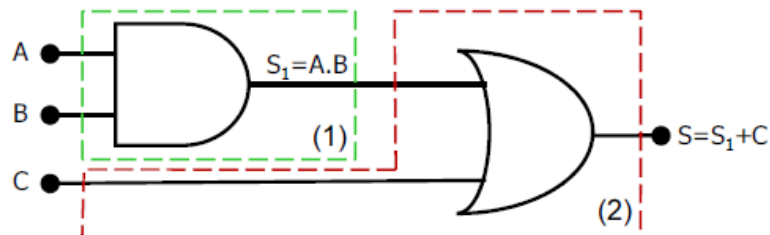
Resolução

Vamos dividi-lo em duas partes (1) e (2). No circuito (1), a saída S1 contém o produto A.B, já que o bloco é uma porta E. Portanto,  $S_1 = A.B$



No circuito (2), note que a saída S1 é utilizada como uma das entradas da porta OU. A outra entrada da porta OU corresponde à variável C, o que nos leva à:

$$S = S_1 + C$$

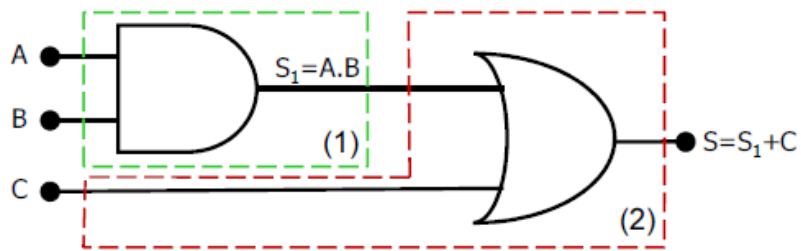


Para obter a expressão final em relação às entradas A, B e C basta substituir a expressão S1 na expressão de S, ou seja:

$$(1) S_1 = A.B$$

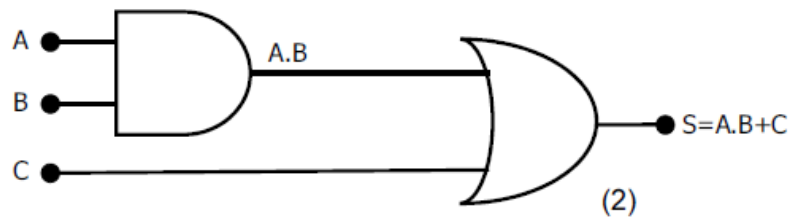
$$(2) S = S_1 + C$$

$$\text{Obtém-se } S = S_1 + C = (A.B) + C$$

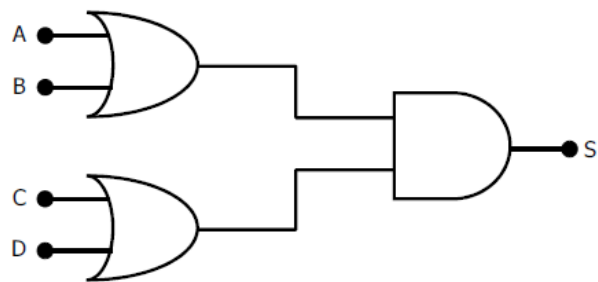


Portanto, a expressão que o circuito executa é:

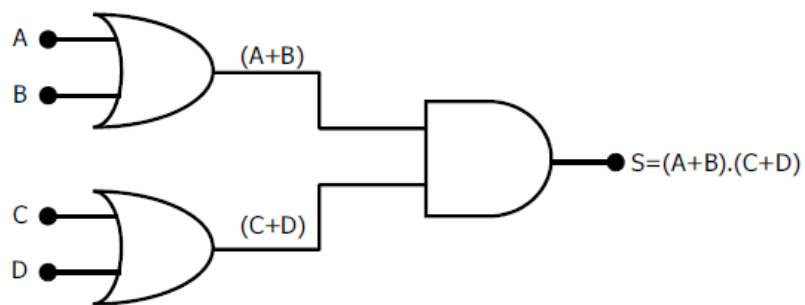
$$S = (A.B) + C = A.B + C$$



2. Escreva a expressão booleana executada pelo circuito

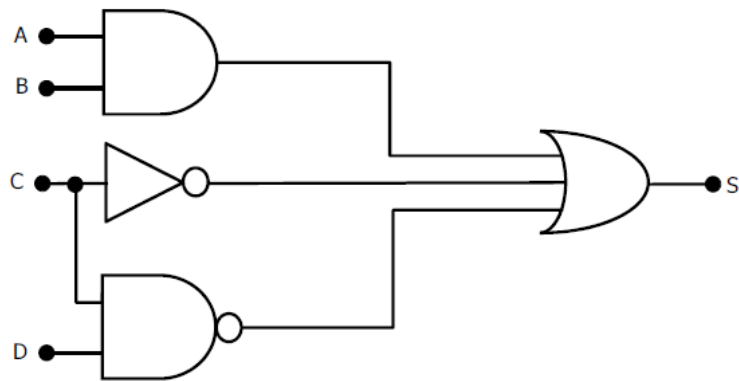


Resolução

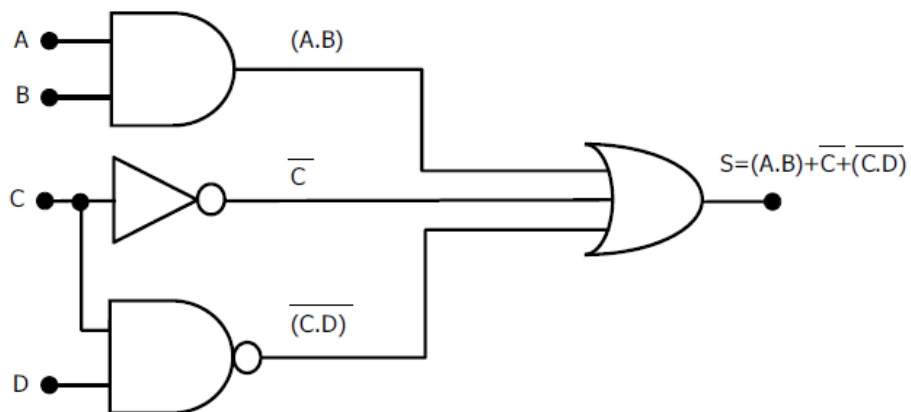




3. Determinar a expressão booleana característica do circuito



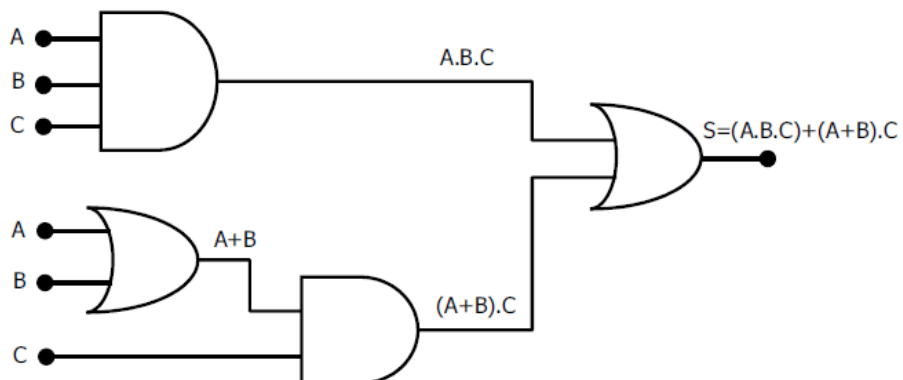
Resolução



4. Desenhe o circuito lógico que executa a seguinte expressão booleana

$$S = (A.B.C) + (A+B).C$$

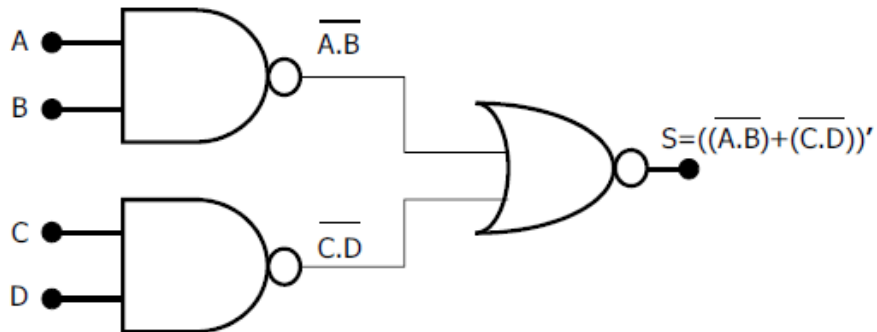
Resolução



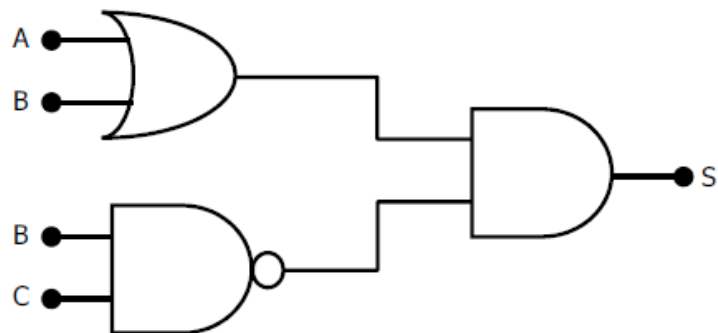
5. Desenhe o circuito lógico cuja expressão característica é

$$S = (\overline{A.B} + \overline{C.D})'$$

Resolução



6. Seja o circuito:

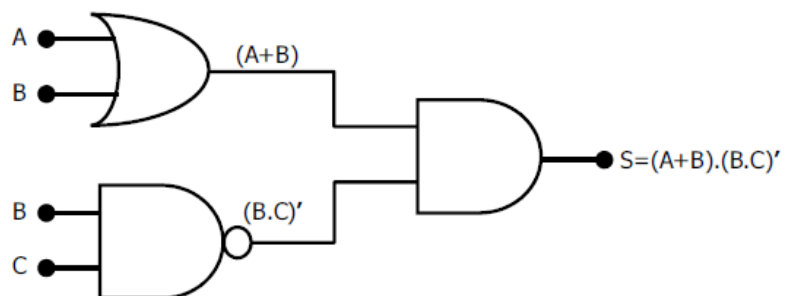


Determine:

- i) A expressão lógica do circuito
- ii) A tabela verdade da expressão

Resolução

- i) Teremos o seguinte circuito



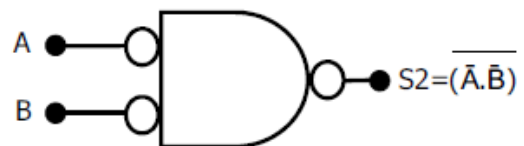
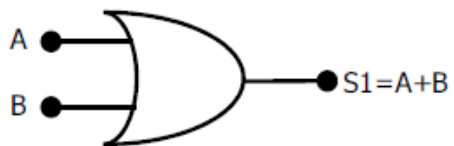
Extraímos sua expressão característica:

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

ii) A partir da expressão teremos a seguinte tabela verdade:

A	B	C	A+B	B.C	(B.C)'	S
0	0	0	0	0	1	0
0	0	1	0	0	1	0
0	1	0	1	0	1	1
0	1	1	1	1	0	0
1	0	0	1	0	1	1
1	0	1	1	0	1	1
1	1	0	1	0	1	1
1	1	1	1	1	0	0

7. Prove, usando tabela verdade, que os seguintes blocos lógicos são equivalentes:



Resolução

Teremos

A	B	$\bar{A}$	$\bar{B}$	$\bar{A} \cdot \bar{B}$	S1= A+B	S2= $\overline{A \cdot B}$
0	0	1	1	1	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	0	1	1

## **Bibliografia**

Idoeta, I.V. & Capuano, F.G.; Elementos de Eletrônica Digital, 12ª. edição, Érica, 1987.

E. Mendelson; Álgebra booleana e circuitos de chaveamento, McGraw-Hill, 1977.

Copyright© Apresentação 2012 por José Augusto Baranauskas. Universidade de São Paulo