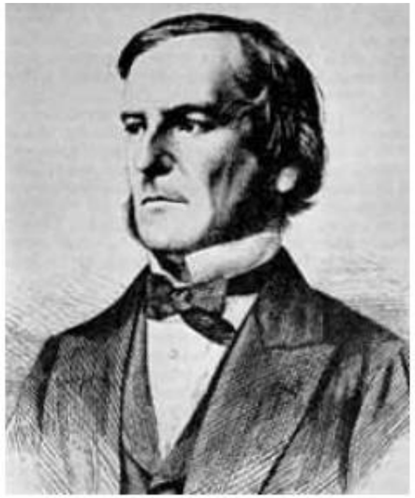
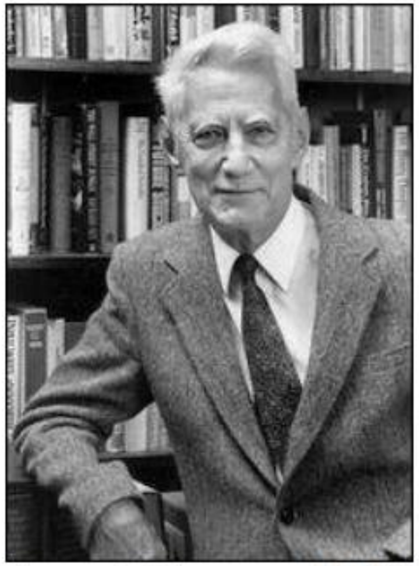
Funções Lógicas e Portas Lógicas

Prof. Demétrius de Castro

demdecastro@gmail.com

83 9 8773-0383


George Boole (1815-1864)

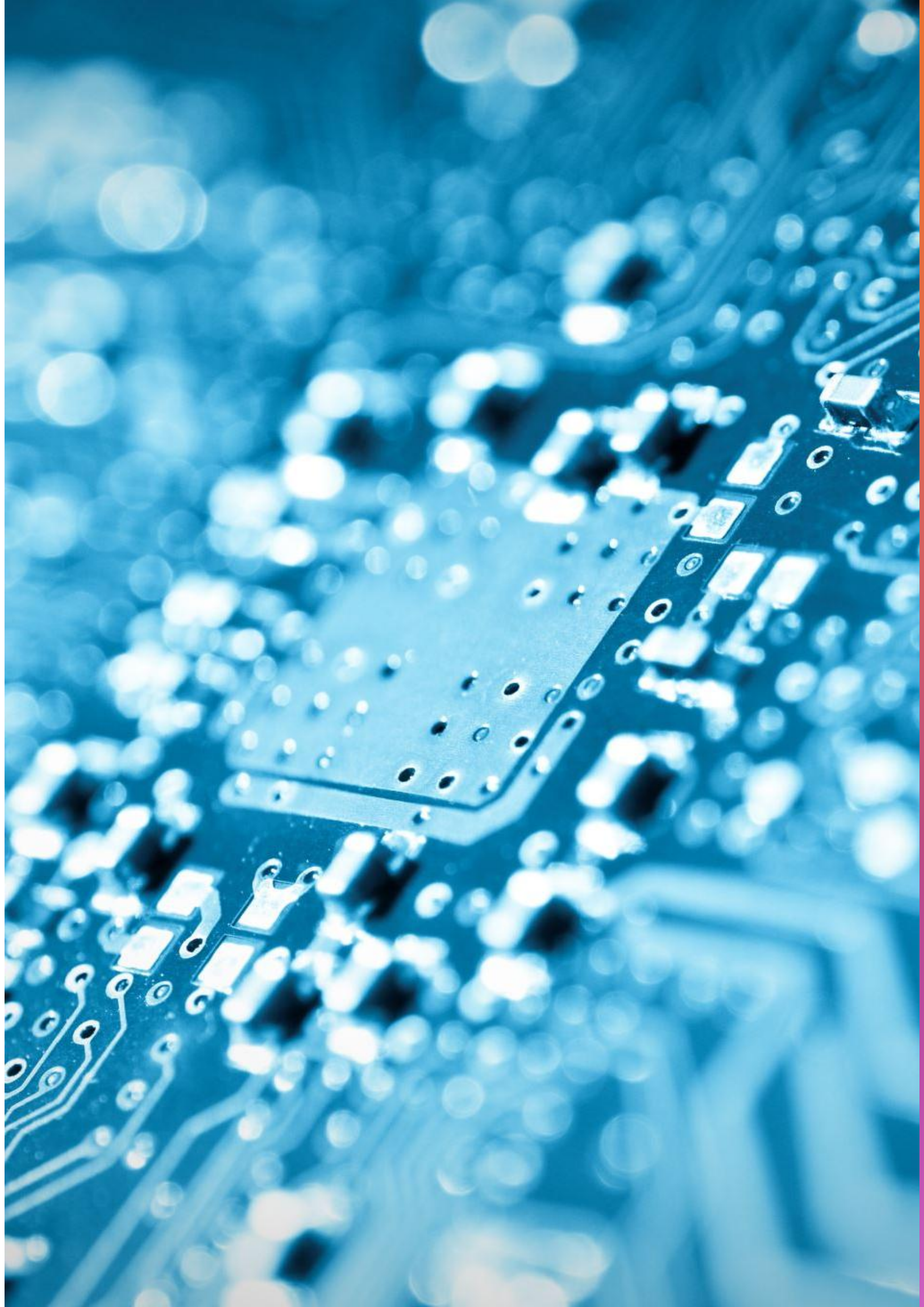

Claude Elwood Shannon (1916-2001)

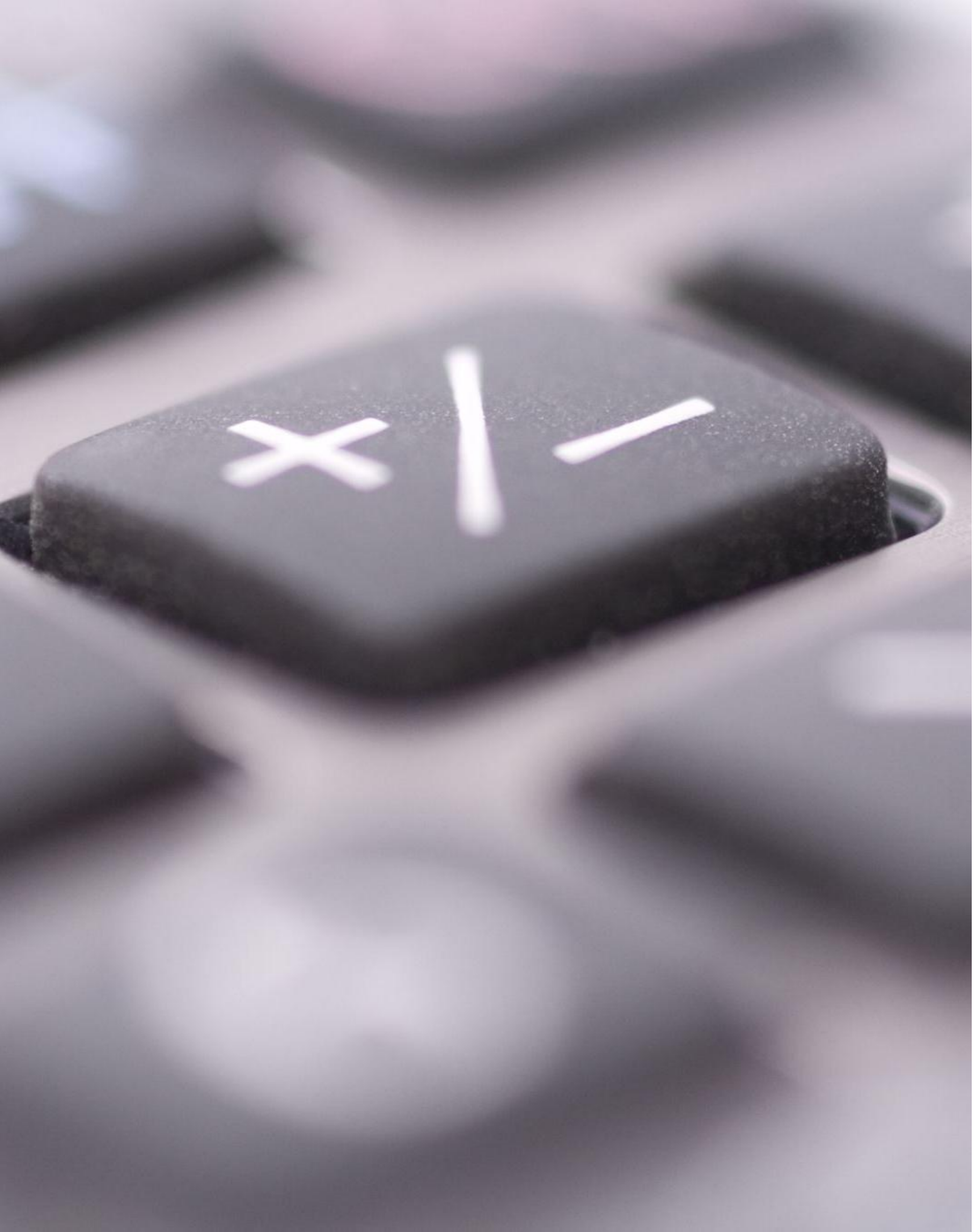
História

Em meados do século XIX o matemático inglês George Boole desenvolveu um sistema matemático de análise lógica

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 comutação

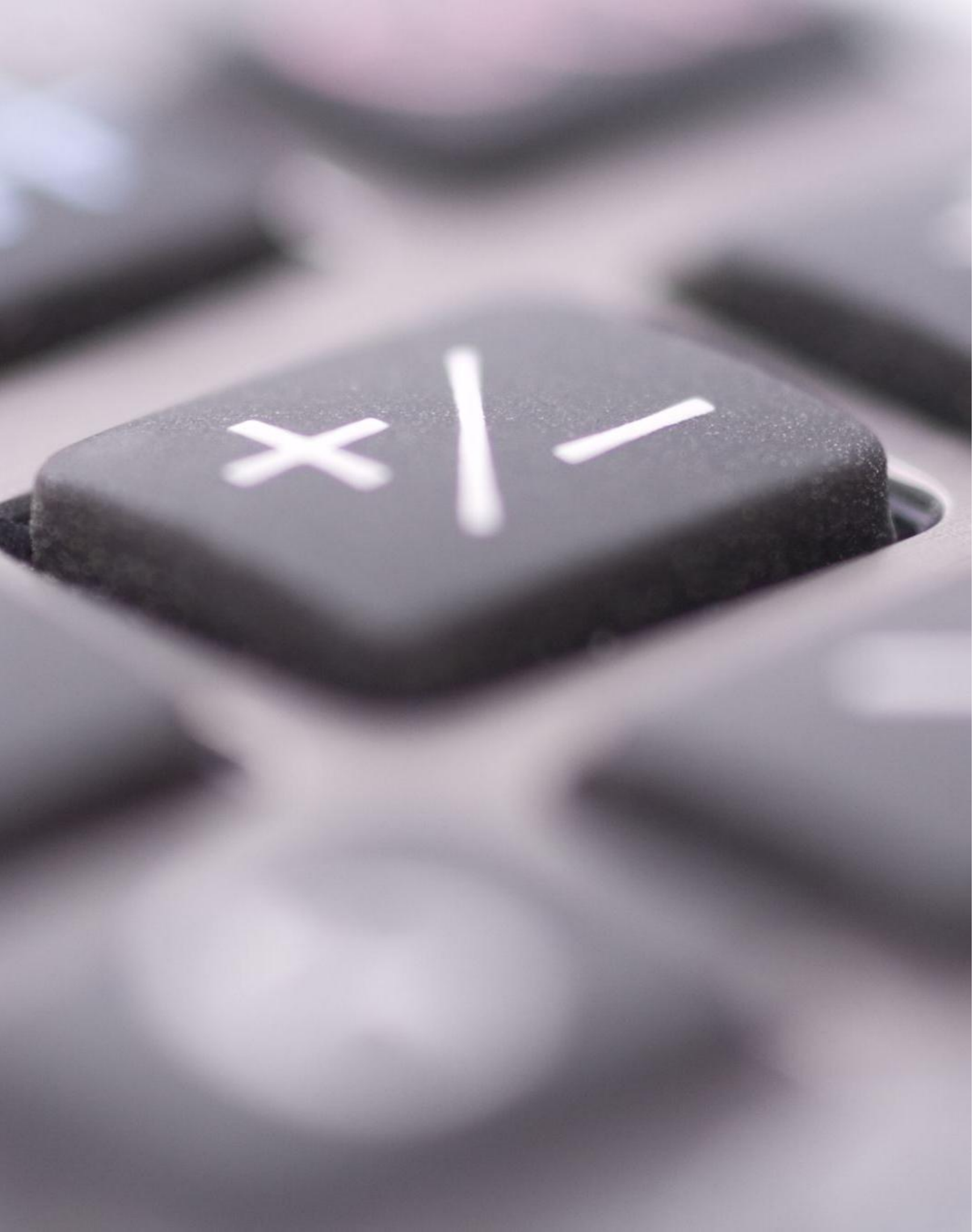
- Nos primórdios da eletrônica, todos os problemas eram solucionados por meio de sistemas analógicos
- Com o avanço da tecnologia, os problemas passaram a ser solucionados pela eletrônica digital
- Na eletrônica digital, os sistemas (computadores, processadores de dados, sistemas de controle, codificadores, 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





Álgebra de Boole

- Na álgebra de Boole, há somente dois estados (valores ou símbolos) permitidos
  - Estado 0 (zero)
  - Estado 1 (um)
- Em geral
  - O estado zero representa não, falso, aparelho desligado, ausência de tensão, chave elétrica desligada, etc
  - O estado um representa sim, verdadeiro, aparelho ligado, presença de tensão, chave ligada, etc



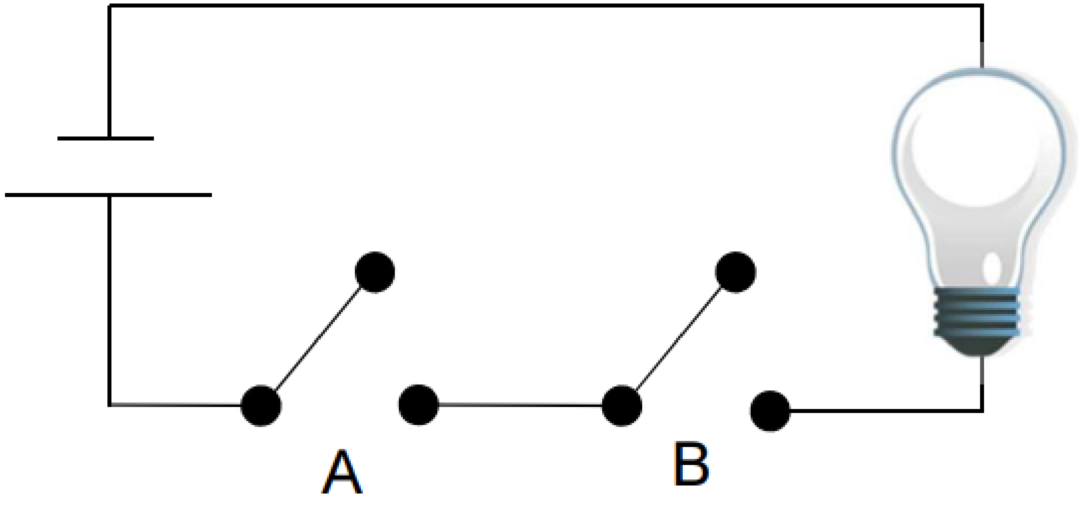
Álgebra de Boole

- Assim, 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
- Uma variável booleana também só assume um dos dois estados permitidos (0 ou 1)

Álgebra de Boole

- Nesta aula 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
- Por último, veremos a equivalência entre blocos

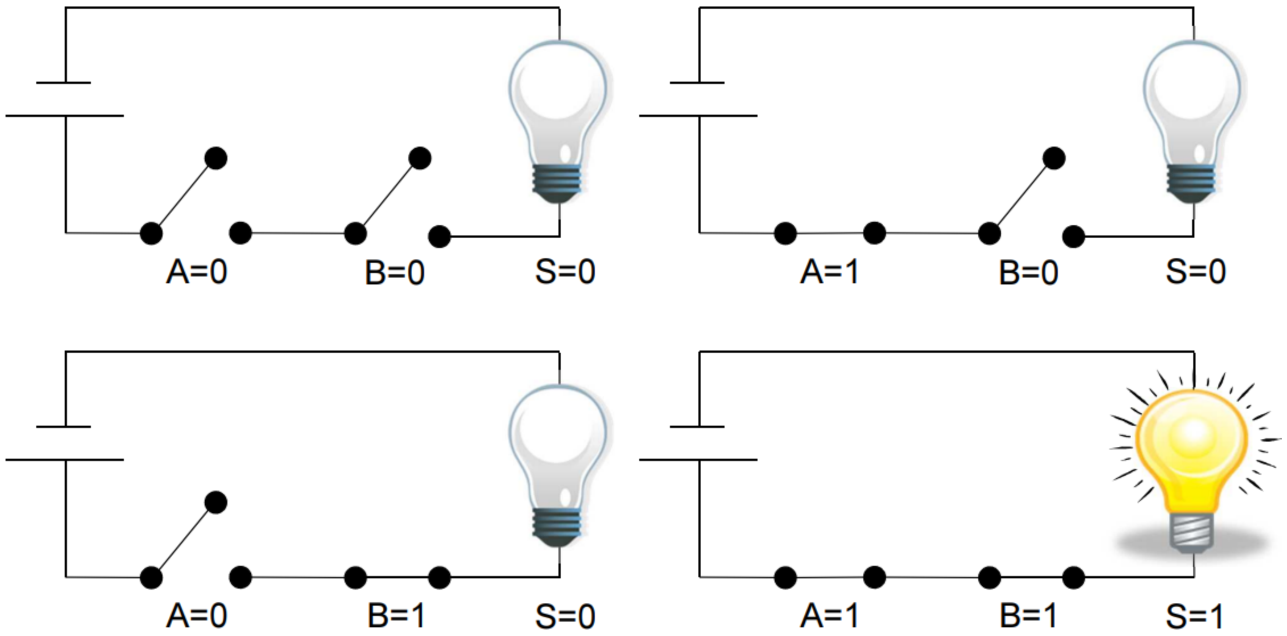
Função E (AND)



- Executa a multiplicação (conjunção) booleana de duas ou mais variáveis binárias
- Chave aberta = 0
- Chave fechada = 1

Função E (AND)

Situações Possíveis



Função E (AND)

- Para representar a expressão
  - S = A e B
- Adotaremos a representação
  - S = A.B, onde se lê S = A e B
- Porém, existem notações alternativas
  - S = A & B
  - S = A, B
  - S = A ∧ B

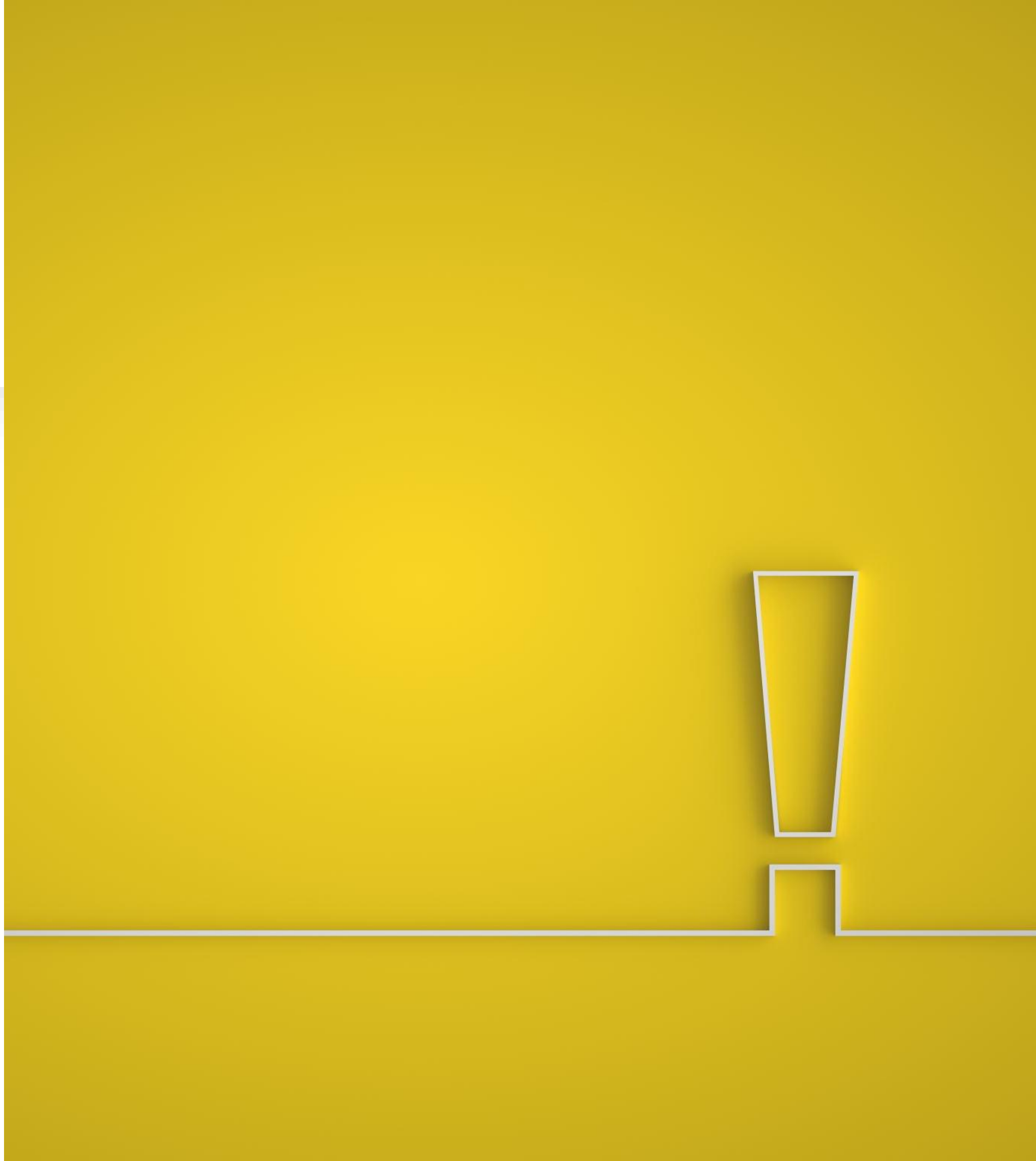


Tabela Verdade

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 2N interpretações possíveis

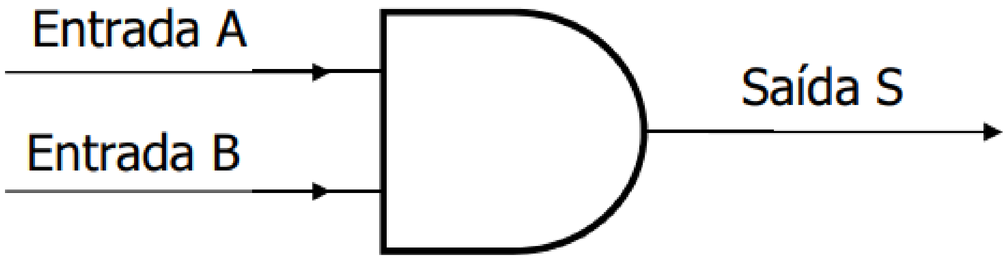
Tabela Verdade da Função E (AND)

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

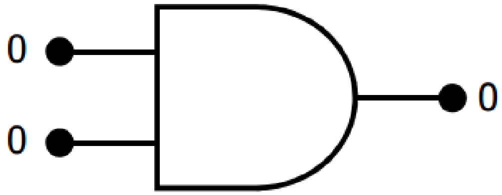
Porta Lógica E (AND)

- 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

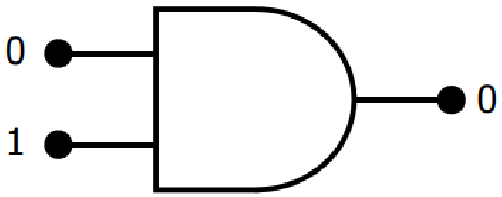
Representação



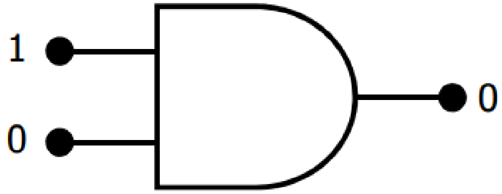
Posições



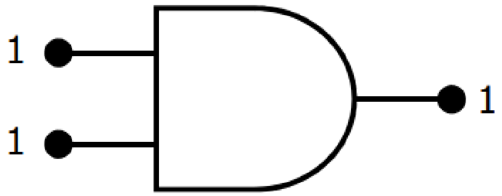
A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1



A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1



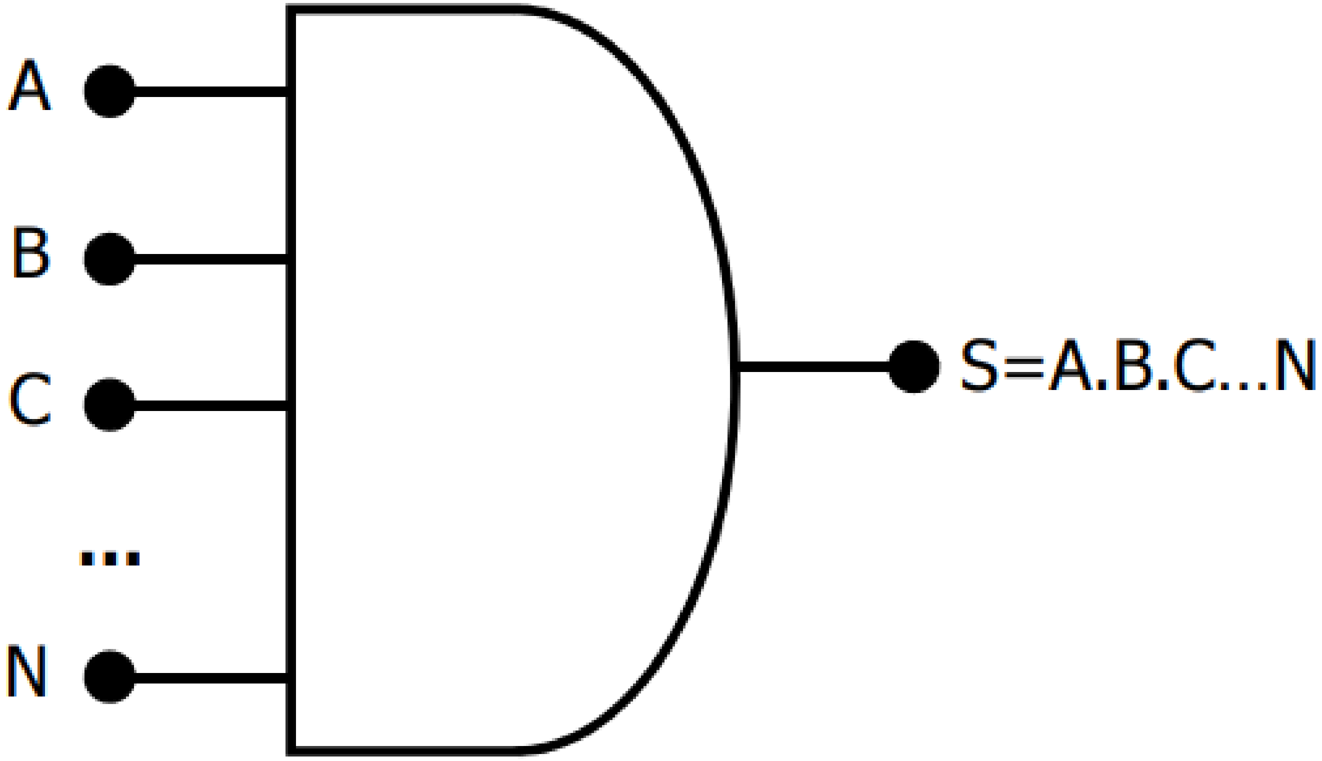
A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1



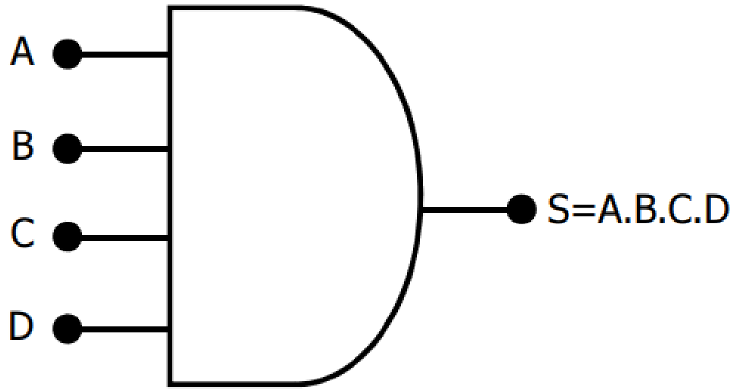
A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Porta Lógica E (AND)

- É possível estender o conceito de uma porta E para um número qualquer de variáveis de entrada
- 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



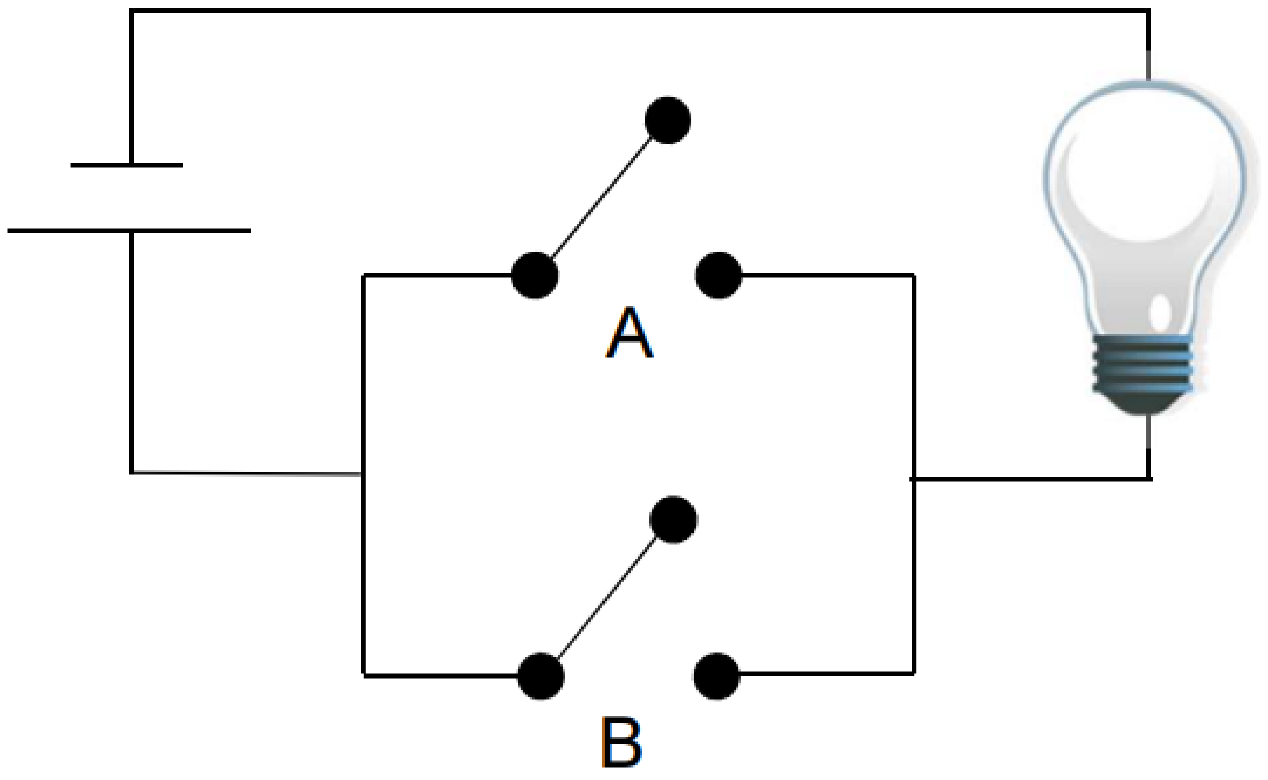
Exemplo
S = A.B.C.D



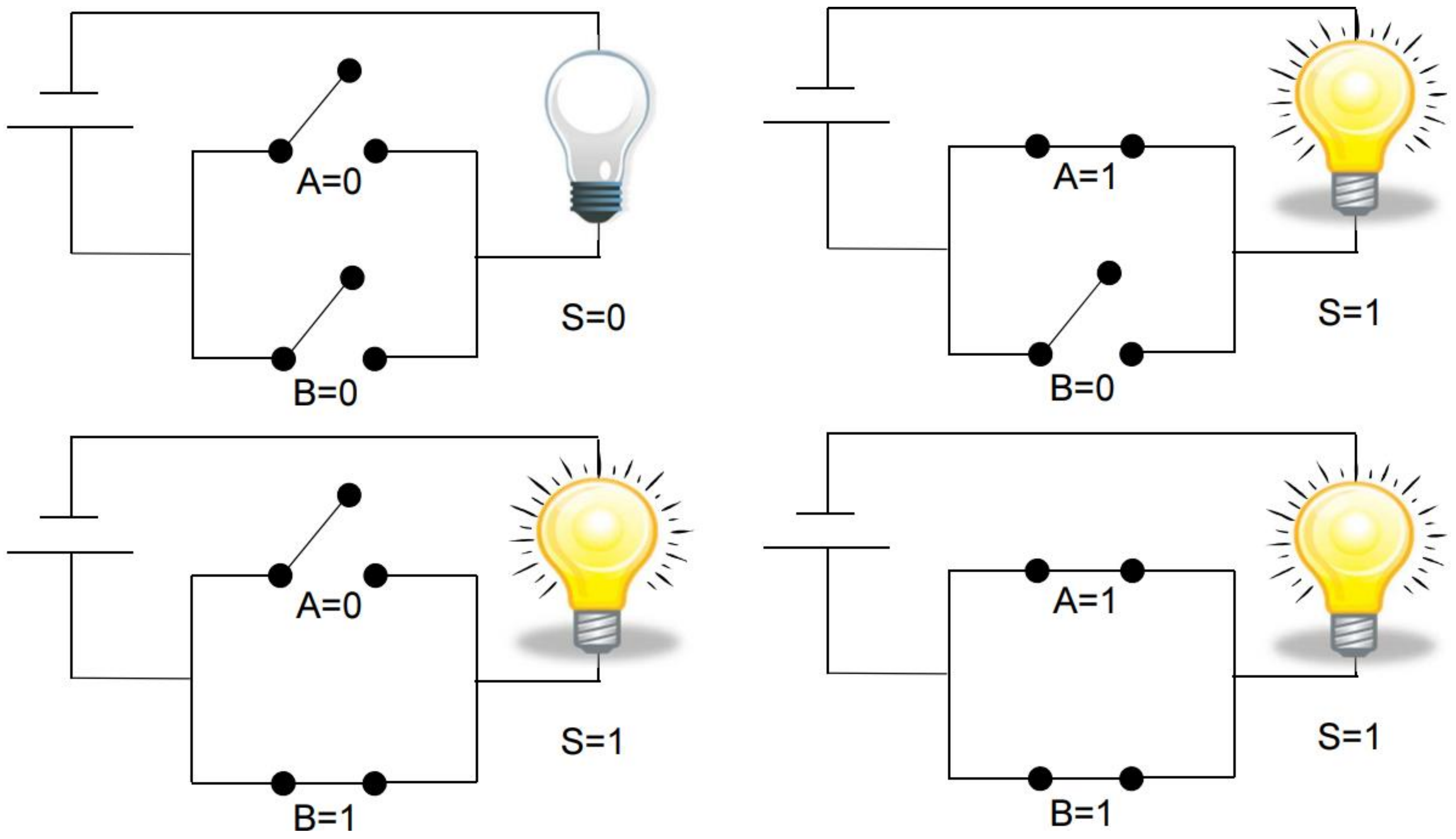
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

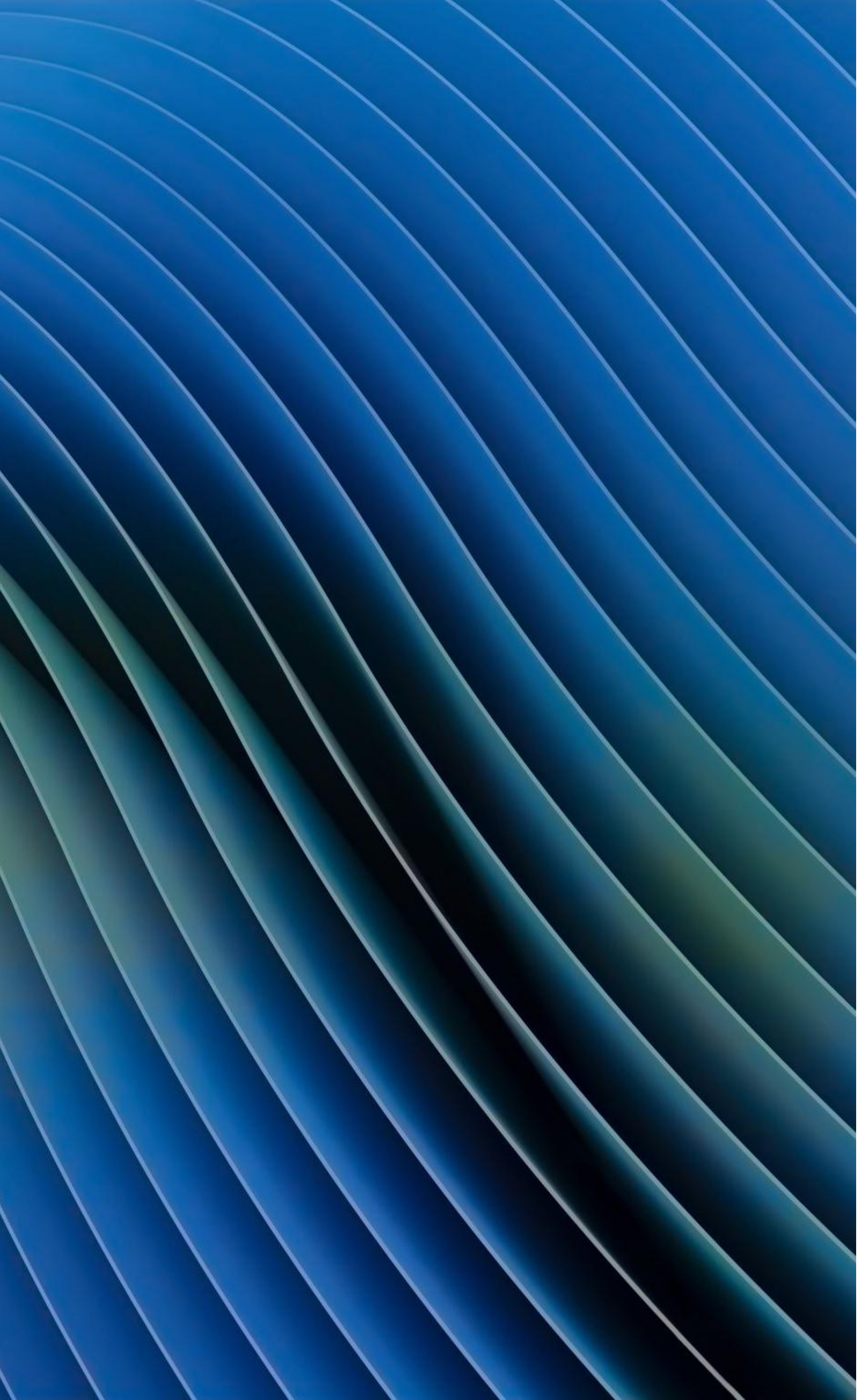
Função OU (OR)

- Executa a soma (disjunção) booleana de duas ou mais variáveis binárias
- Por exemplo, assume a convenção no circuito
  - Chave aberta = 0
  - Chave fechada = 1



Função OU (OR)





Função OU (OR)

Para representar a expressão

- $S = A$ ou B

Adotaremos a representação

- $S = A+B$, onde se lê $S = A$ ou B

Porém, existem notações alternativas

- $S = A \mid B$
- $S = A; B$
- $S = A \vee B$

Tabela Verdade da Função OU (OR)

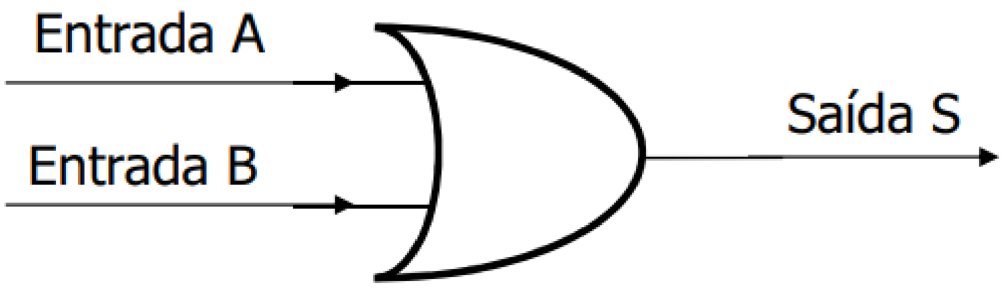
- Na álgebra booleana, 1+1=1, já que somente dois valores são permitidos (0 e 1)

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

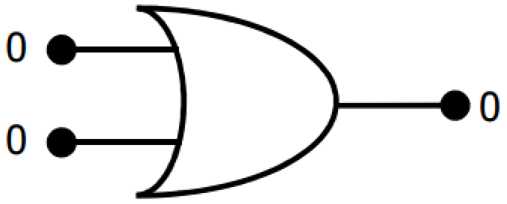
Tabela Verdade da Função OU (OR)

- 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.

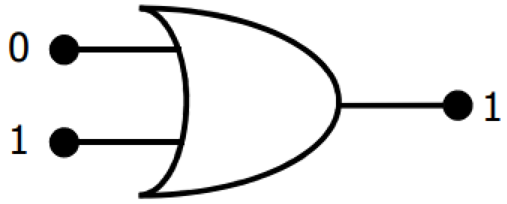
Representação



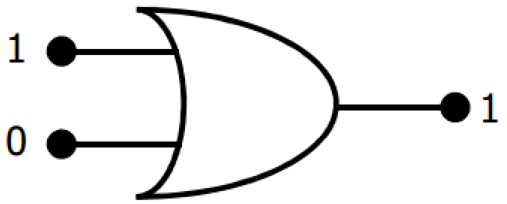
Posições



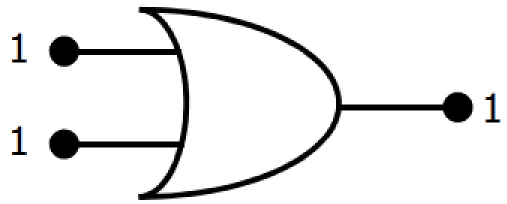
A	B	S=A+B
0	0	0
0	1	1
1	0	1
1	1	1



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



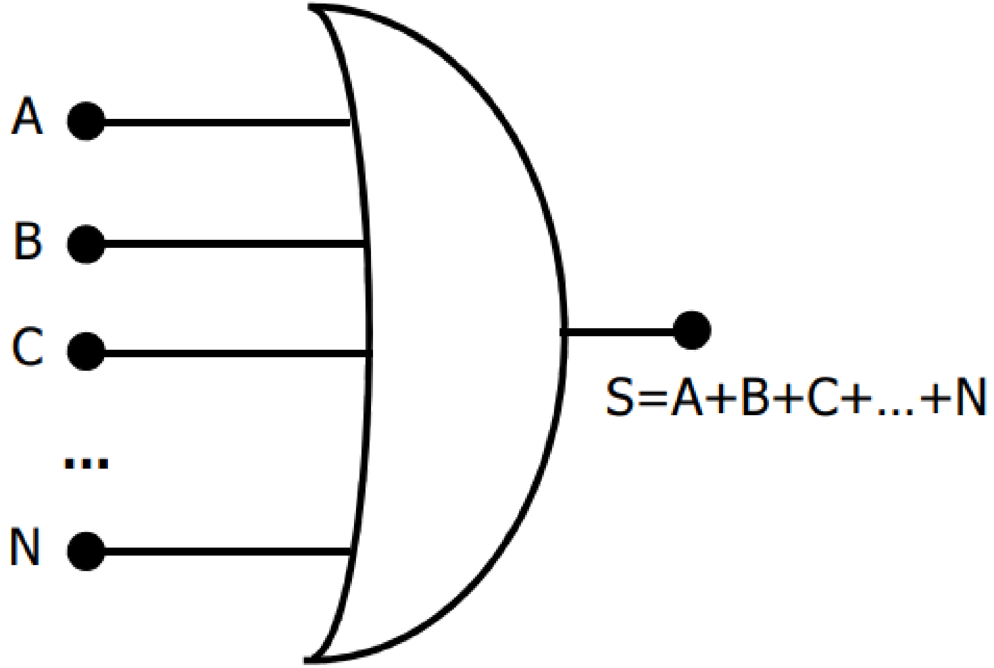
A	B	S=A+B
0	0	0
0	1	1
1	0	1
1	1	1



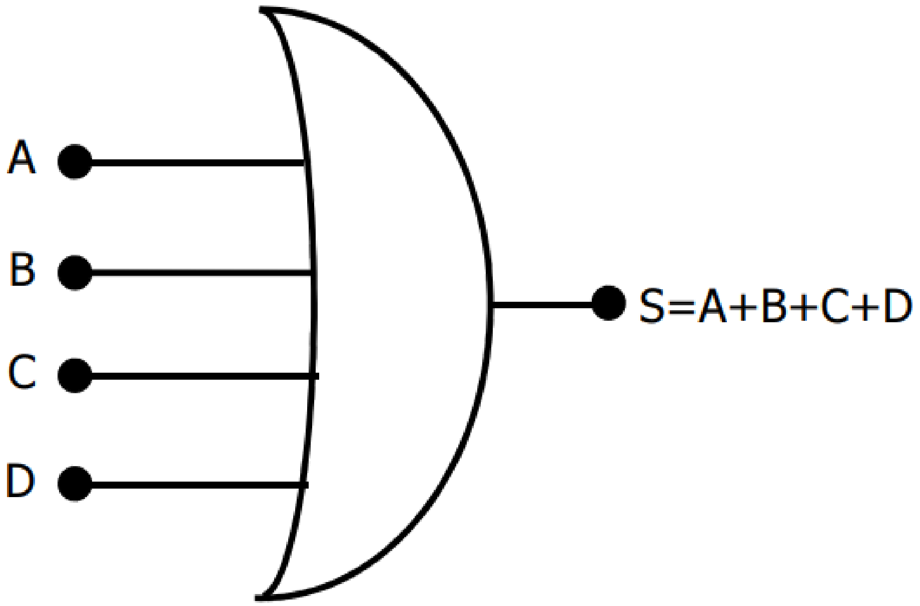
A	B	S=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Portas Lógicas
OU (OR)

- 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



Exmplo
S = A+B+C+D



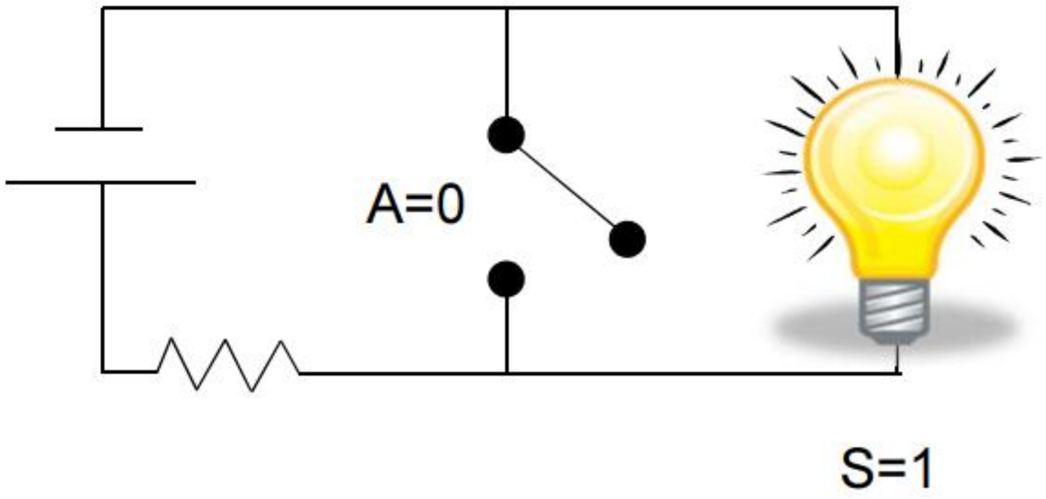
A	B	C	D	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

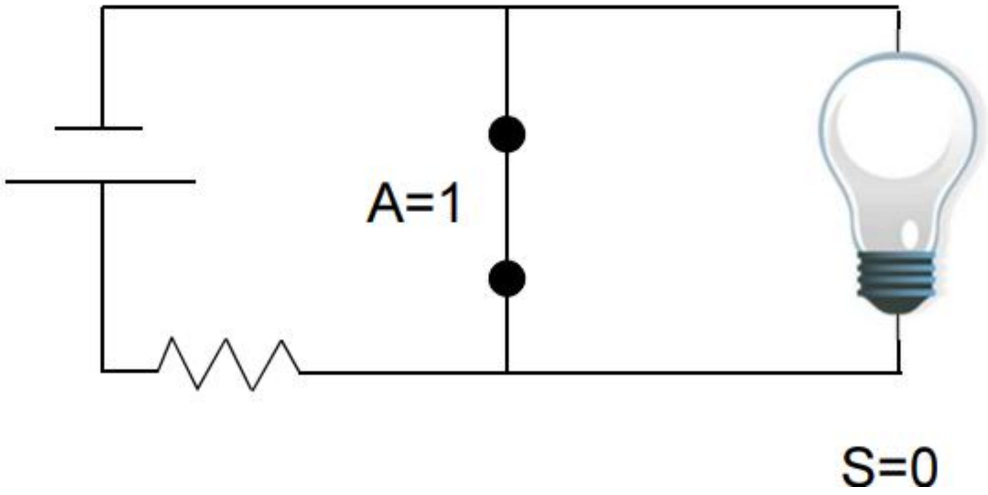
Função NÃO (NOT)

- 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 inversor

Função NÃO (NOT)

- 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)





Função NÃO (NOT)

- Para representar a expressão
- S = não A
- Adotaremos a representação
- S = A’, onde se lê S = não A
- Notações alternativas
- S = $\bar{A}$
- S = ¬ A
- S = Ã

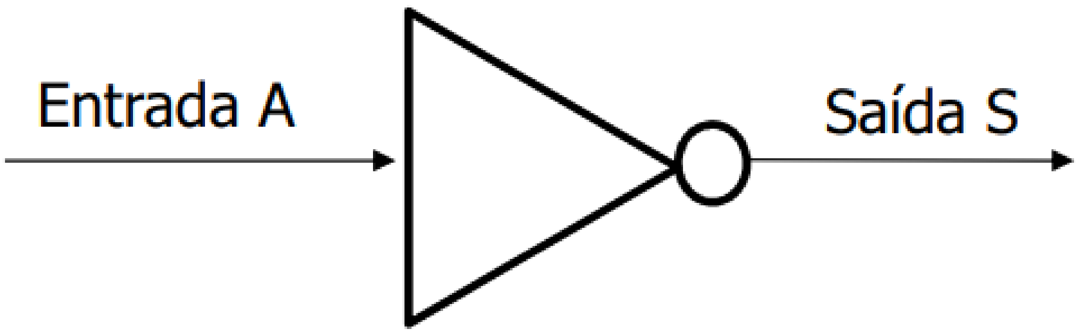
Tabela Verdade da Função NÃO (NOT)

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

Função NÃO (NOT)

- 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

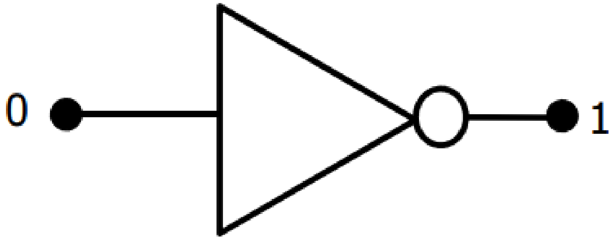


Alternativas

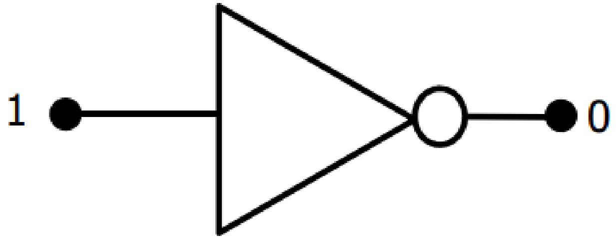
Após um bloco lógico

Antes de um bloco lógico

Função NÃO (NOT)



A	S=Ā
0	1
1	0



A	S=Ā
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 = (A.B)’

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

$= \overline{A.B}$

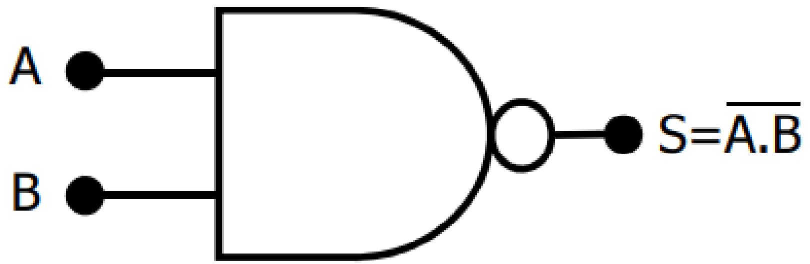
= ¬(A.B)

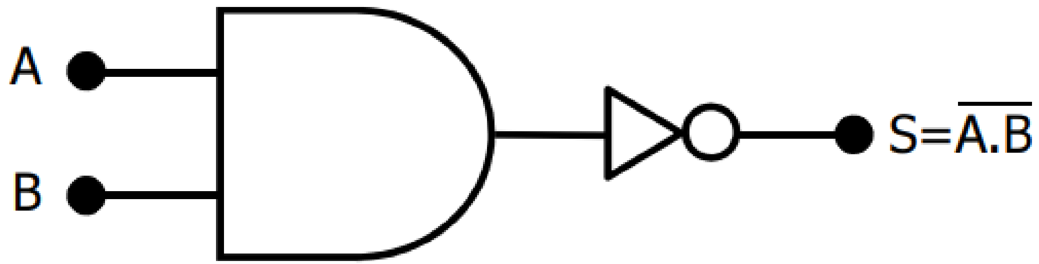
Tabela Verdade

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

Função NÃO E (NAND)

Representação





Função NÃO OU (NOR)

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

S = (A+B)’

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

= $\overline{A+B}$

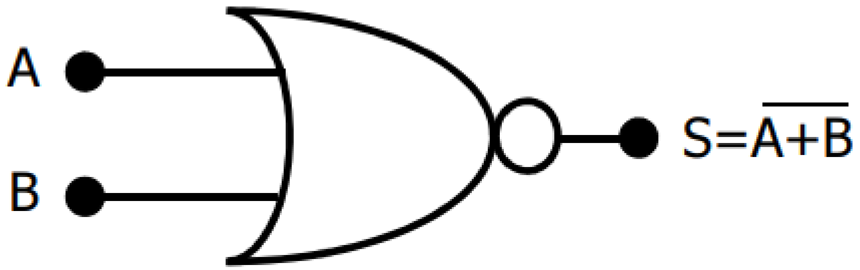
= ¬(A+B)

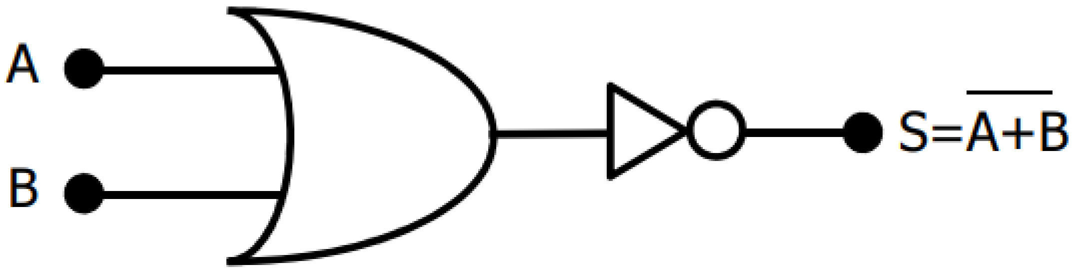
Tabela Verdade

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

Função NÃO OU (NOR)

Representação





Função OU Exclusivo (XOR)

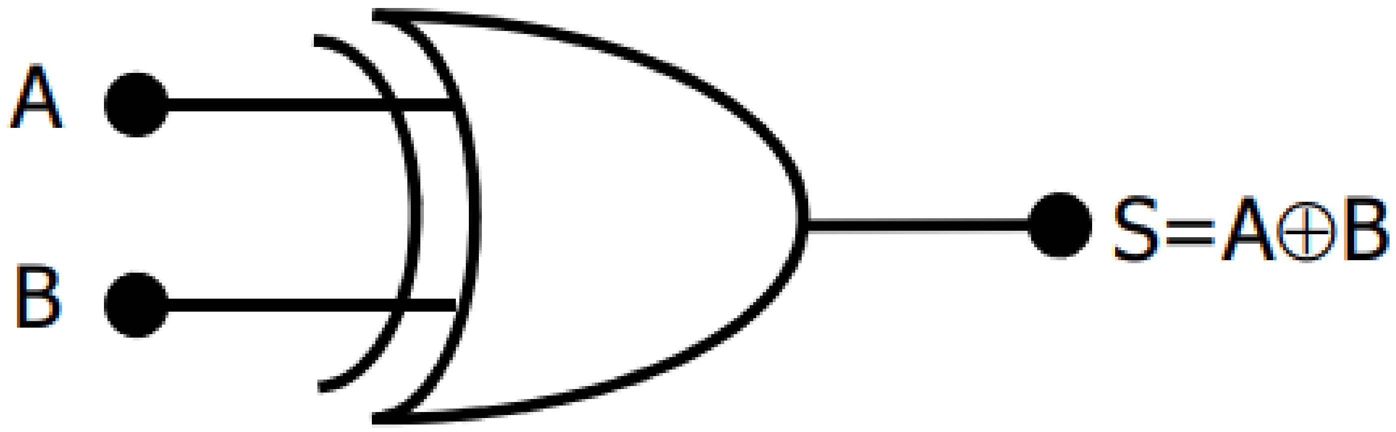
- A função OU Exclusivo fornece
  - 1 na saída quando as entradas forem diferentes entre si e
  - 0 caso contrário
- $S = A \oplus B = \bar{A}.B + A.$

Tabela Verdade

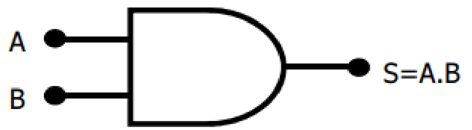
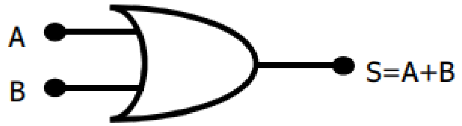
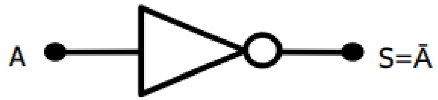
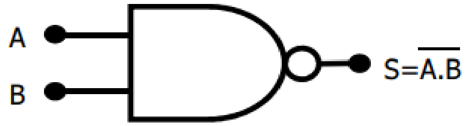
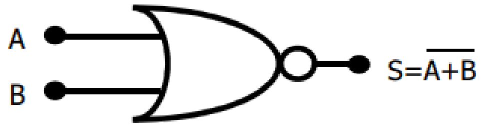
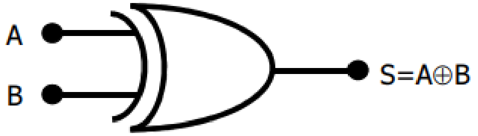
A	B	$S = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

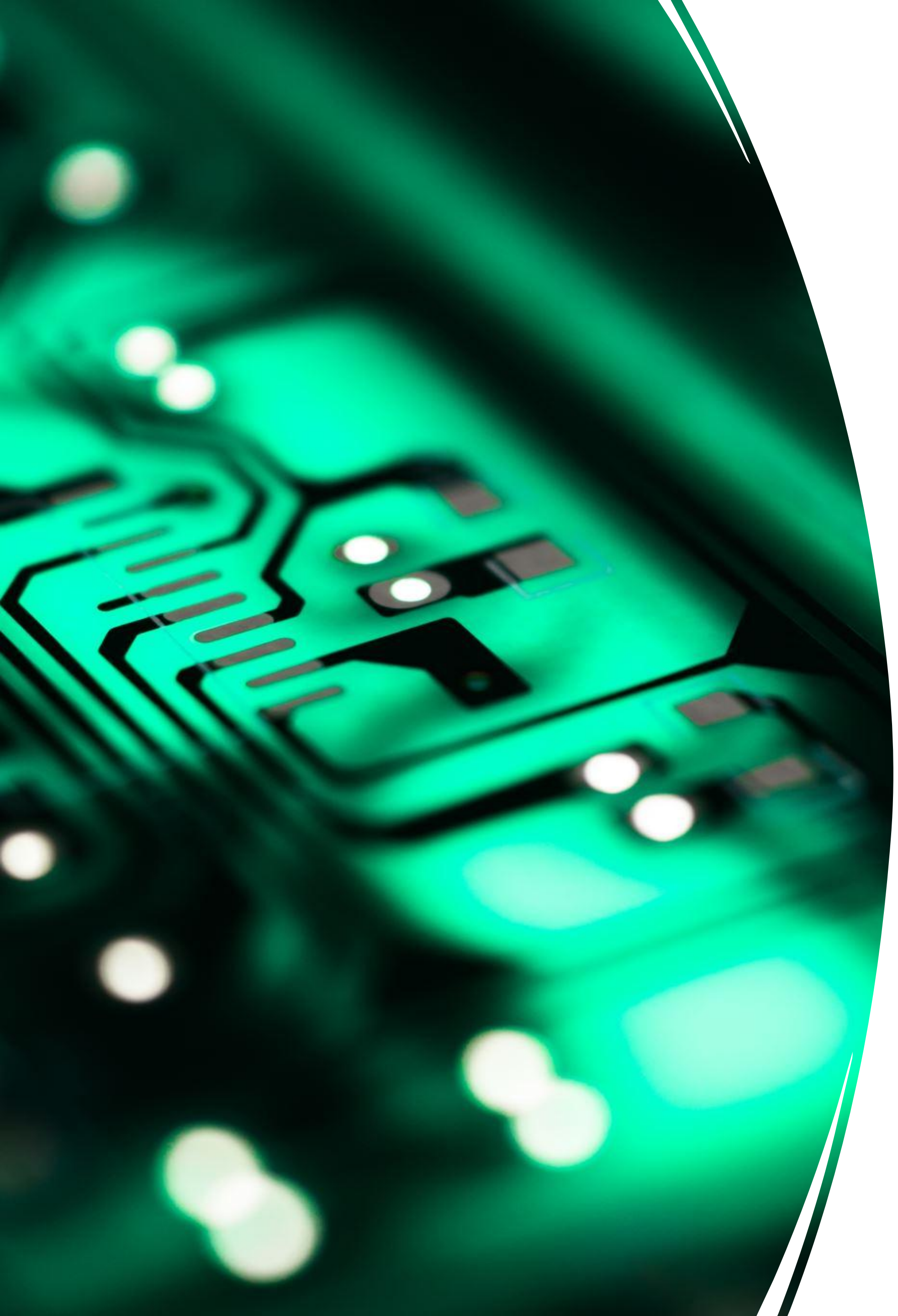
Função OU Exclusivo (XOR)

Representação



Resumo

Nome	Símbolo Gráfico	Função Algébrica	Tabela Verdade
E (AND)		S=A.B S=AB	A B | S=A.B 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OU (OR)		S=A+B	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= ¬ A	A | $S=\bar{A}$ 0 | 1 1 | 0
NE (NAND)		$S=\overline{A.B}$ S=(A.B)’ S= ¬(A.B)	A B | $S=\overline{A.B}$ 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOU (NOR)		$S=\overline{A+B}$ S=(A+B)’ S= ¬(A+B)	A B | $S=\overline{A+B}$ 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		S=A⊕B	A B | S=A⊕B 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

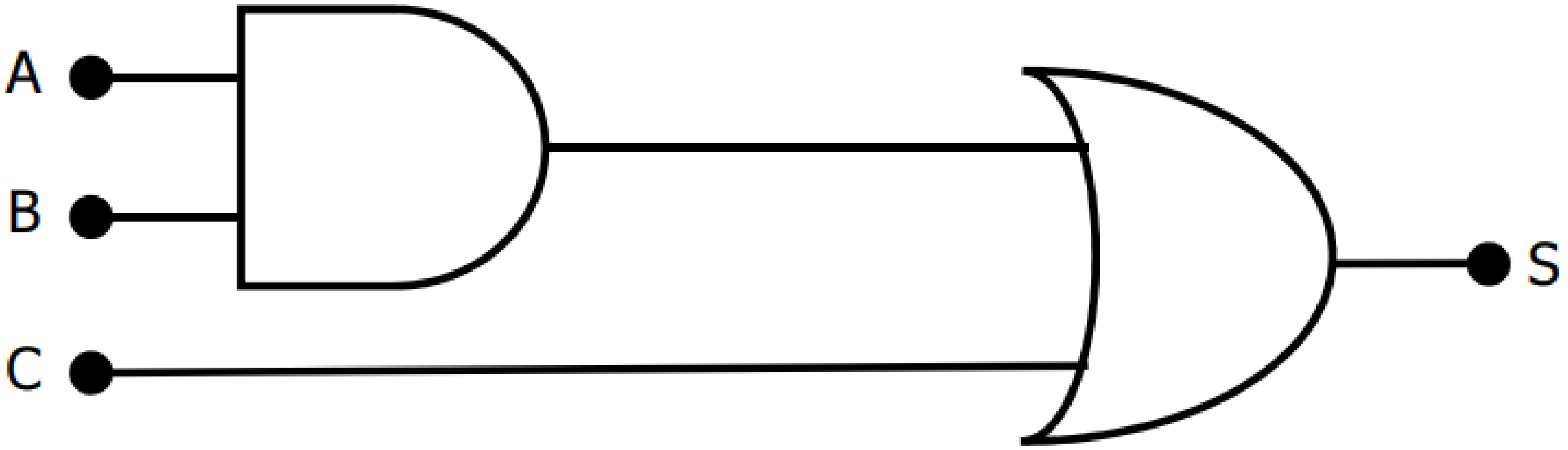


Correspondência entre expressões, circuitos e tabelas verdade

- Todo circuito lógico executa uma expressão booleana
- Um circuito, por mais complexo que seja, é composto pela interligação dos blocos lógicos básicos
- Veremos, a seguir, como obter as expressões booleanas geradas por um circuito lógico

Correspondência entre expressões, circuitos e tabelas verdade

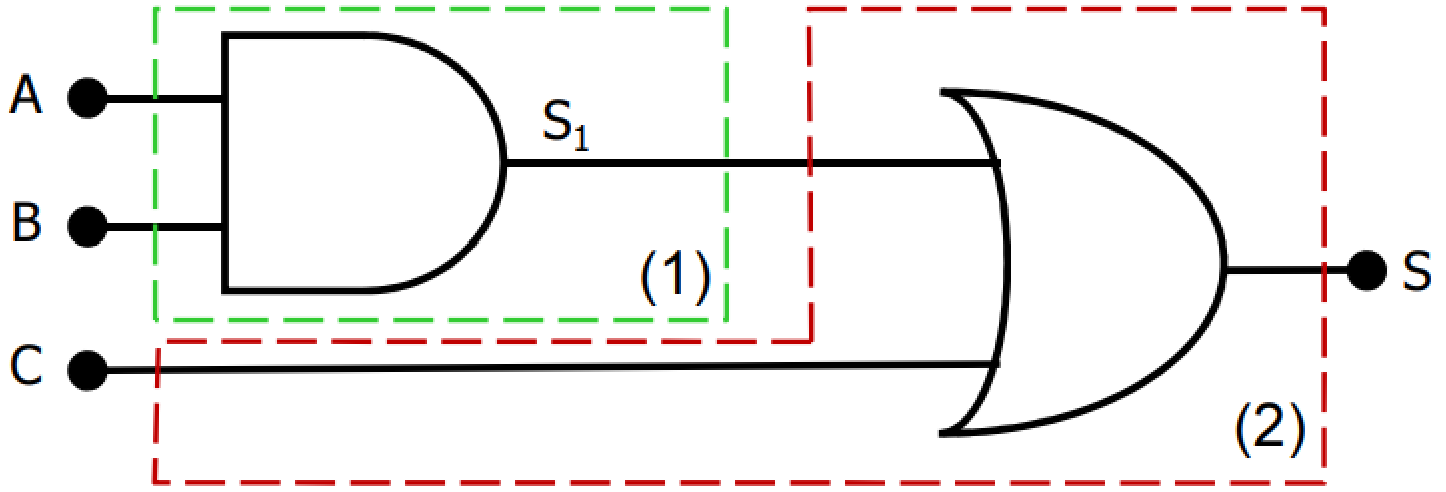
Seja o circuito



Correspondência entre expressões, circuitos e tabelas verdade

Vamos dividi-lo em duas partes (1) e (2)

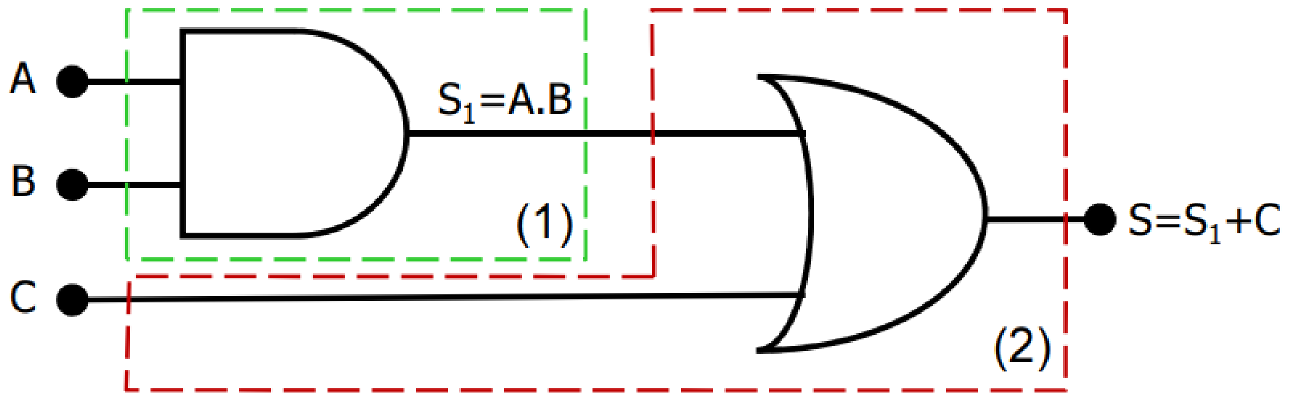
- No circuito (1), a saída S 1 contém o produto A.B, já que o bloco é uma porta E
- Portanto, S 1 = A.B



Correspondência entre expressões, circuitos e tabelas verdade

Vamos dividi-lo em duas partes (1) e (2)

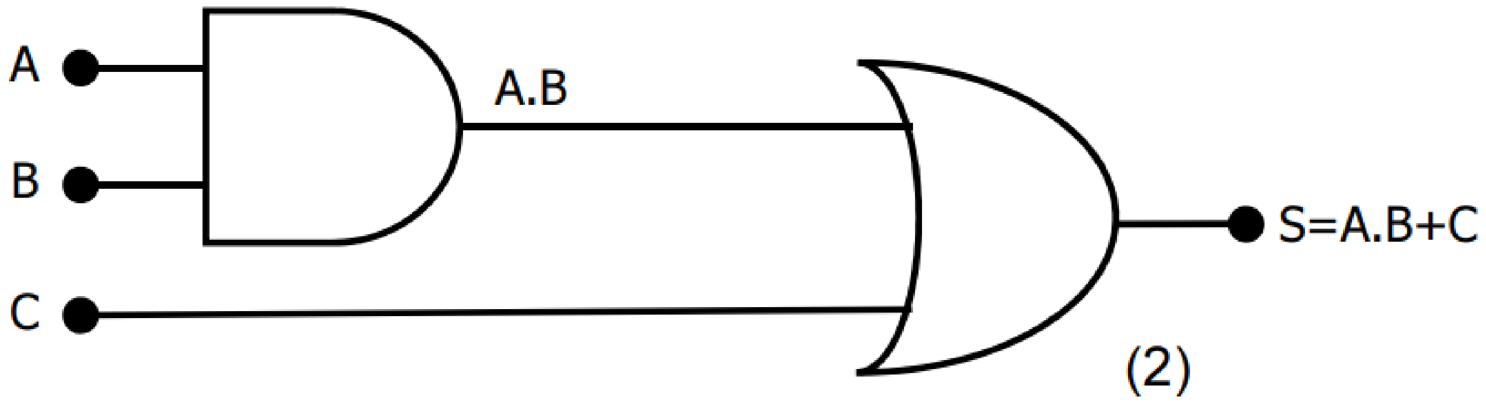
- Para obter a expressão final em relação às entradas A, B e C basta substituir a expressão S 1 na expressão de S, ou seja:
  - (1) S1 = A.B
  - (2) S = S1 + C
  - Obtém-se S = S1 + C = (A.B) + C



Correspondência entre expressões, circuitos e tabelas verdade

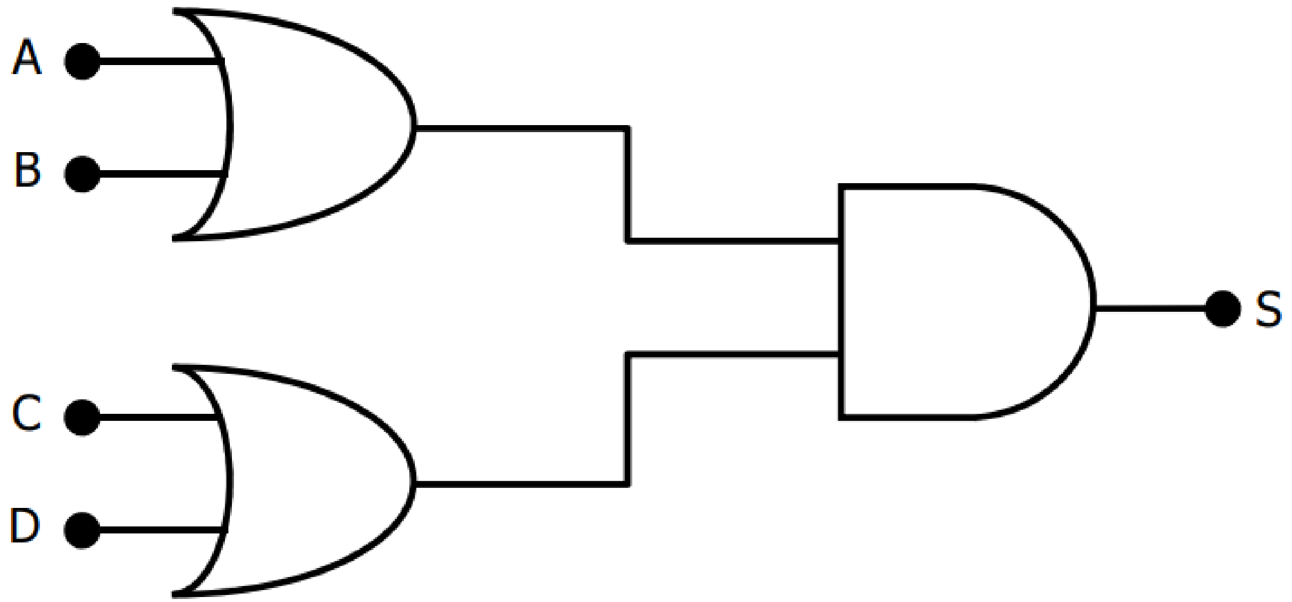
Vamos dividi-lo em duas partes (1) e (2)

- Portanto, a expressão que o circuito executa é:
  - $S = (A.B) + C = A.B + C$

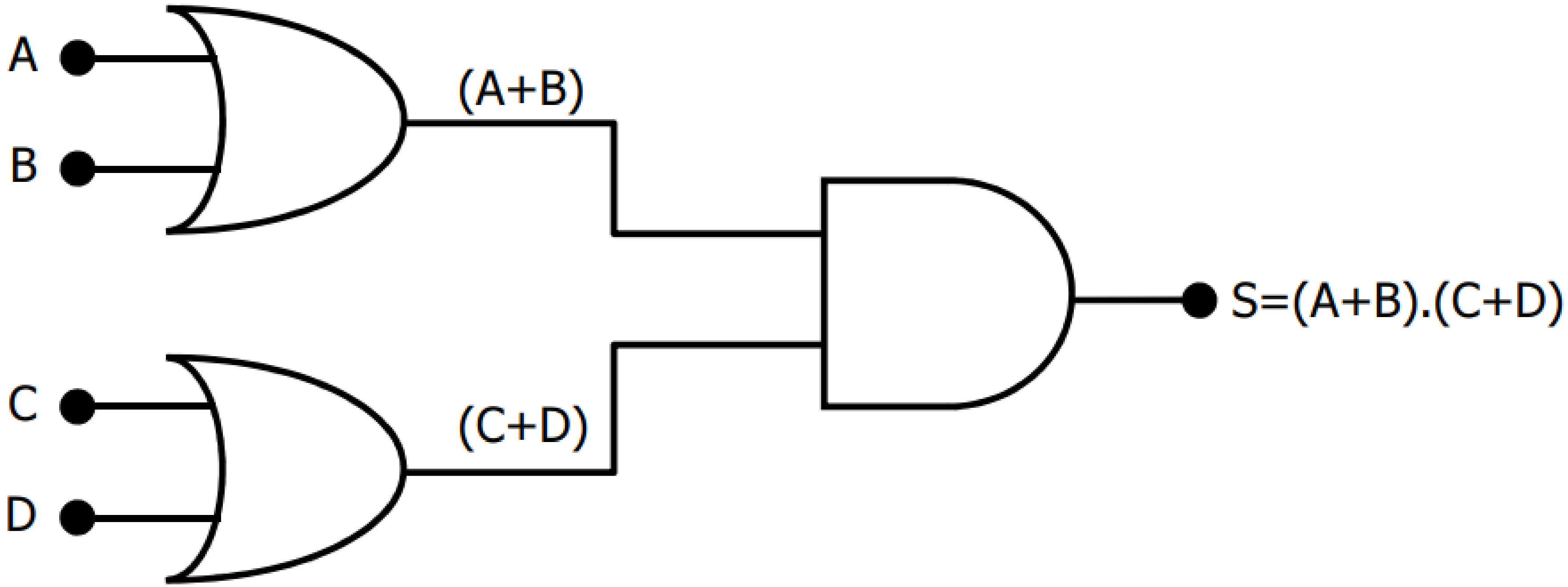


FAÇA

Escreva a expressão booleana executada pelo circuito

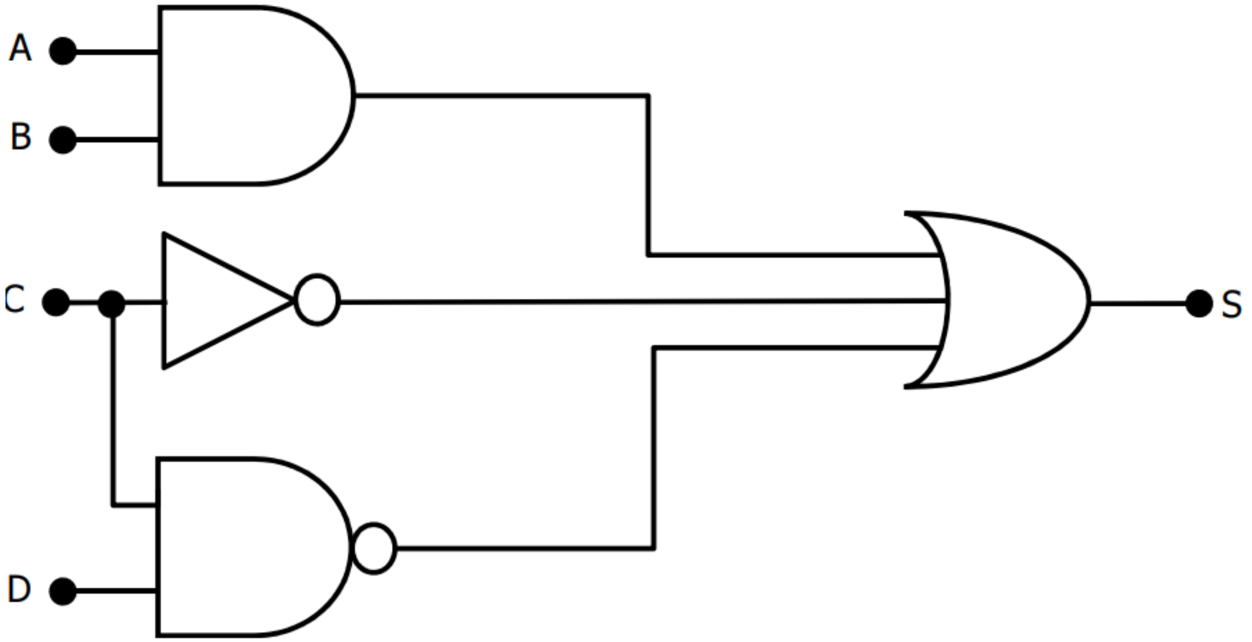


Solução

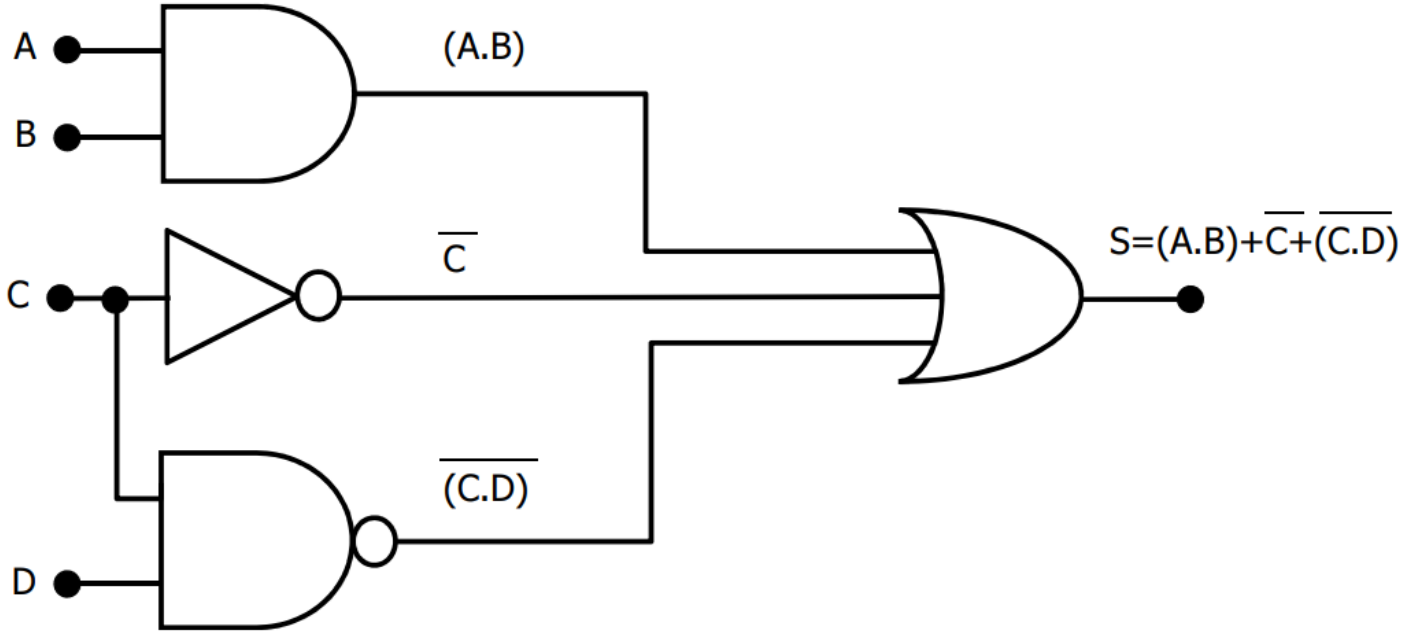


FAÇA

Escreva a expressão booleana executada pelo circuito

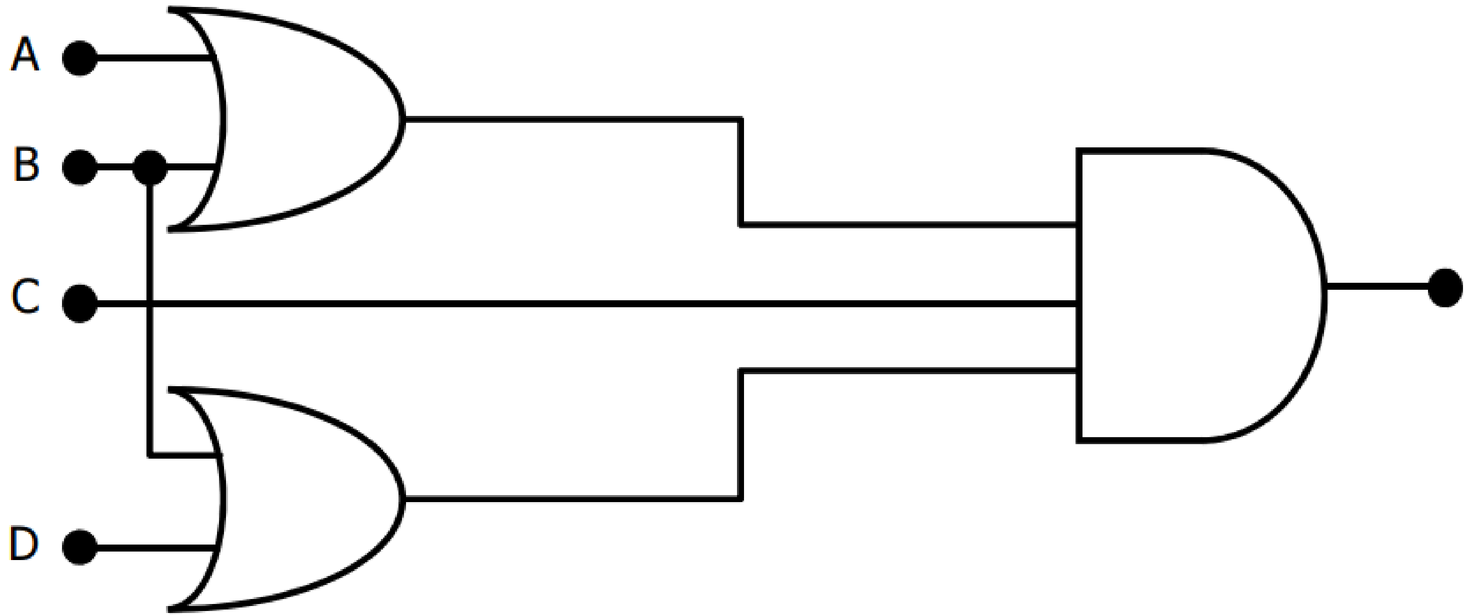


Solução

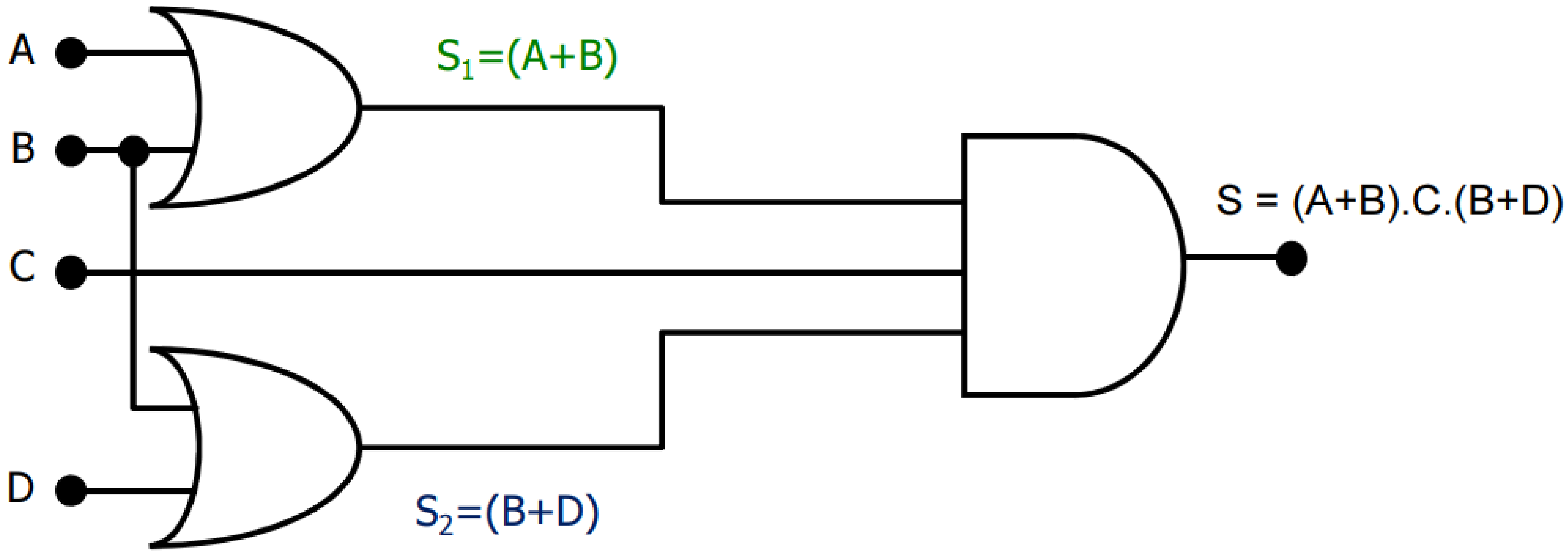


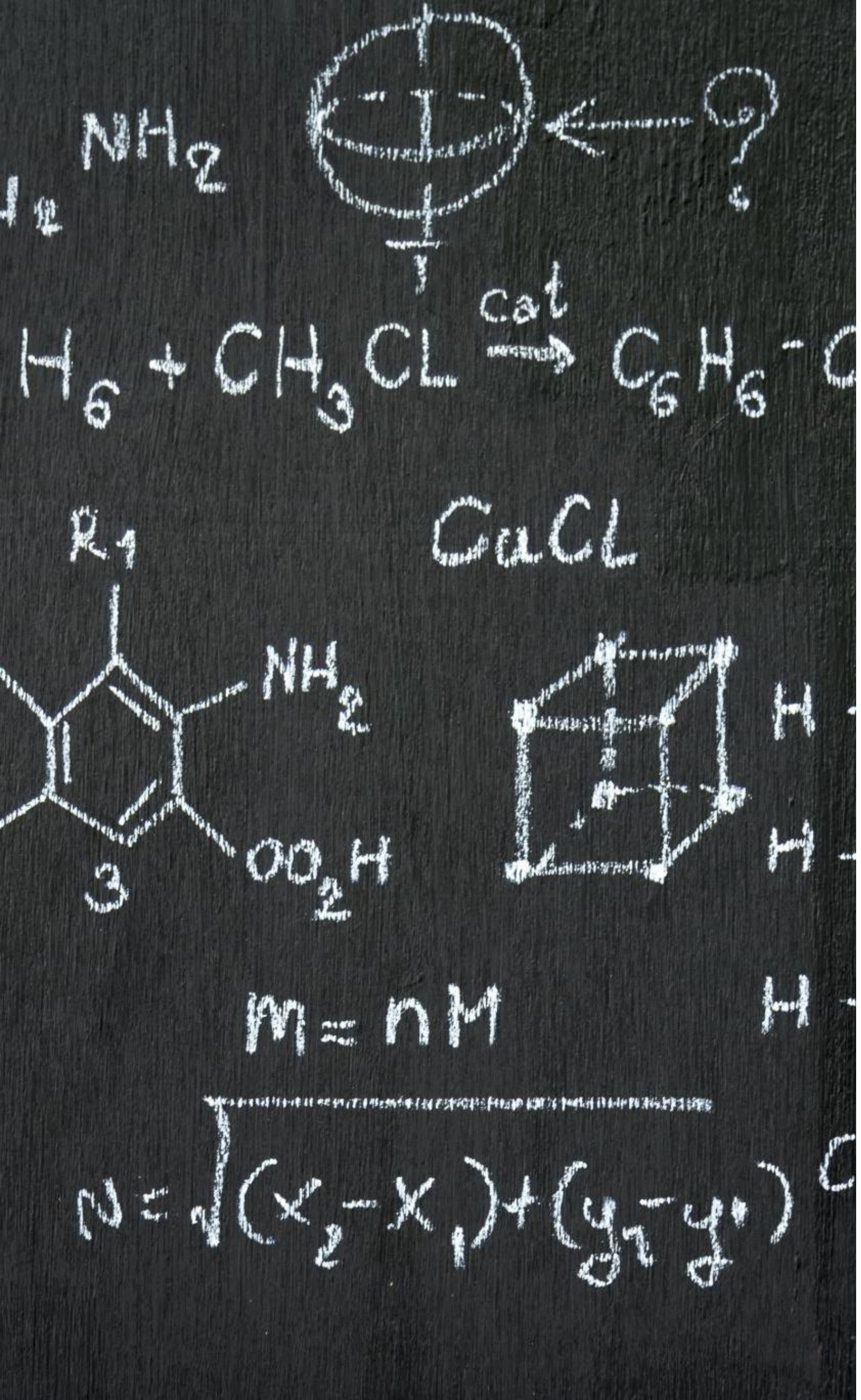
FAÇA

Escreva a expressão booleana executada pelo circuito



Solução

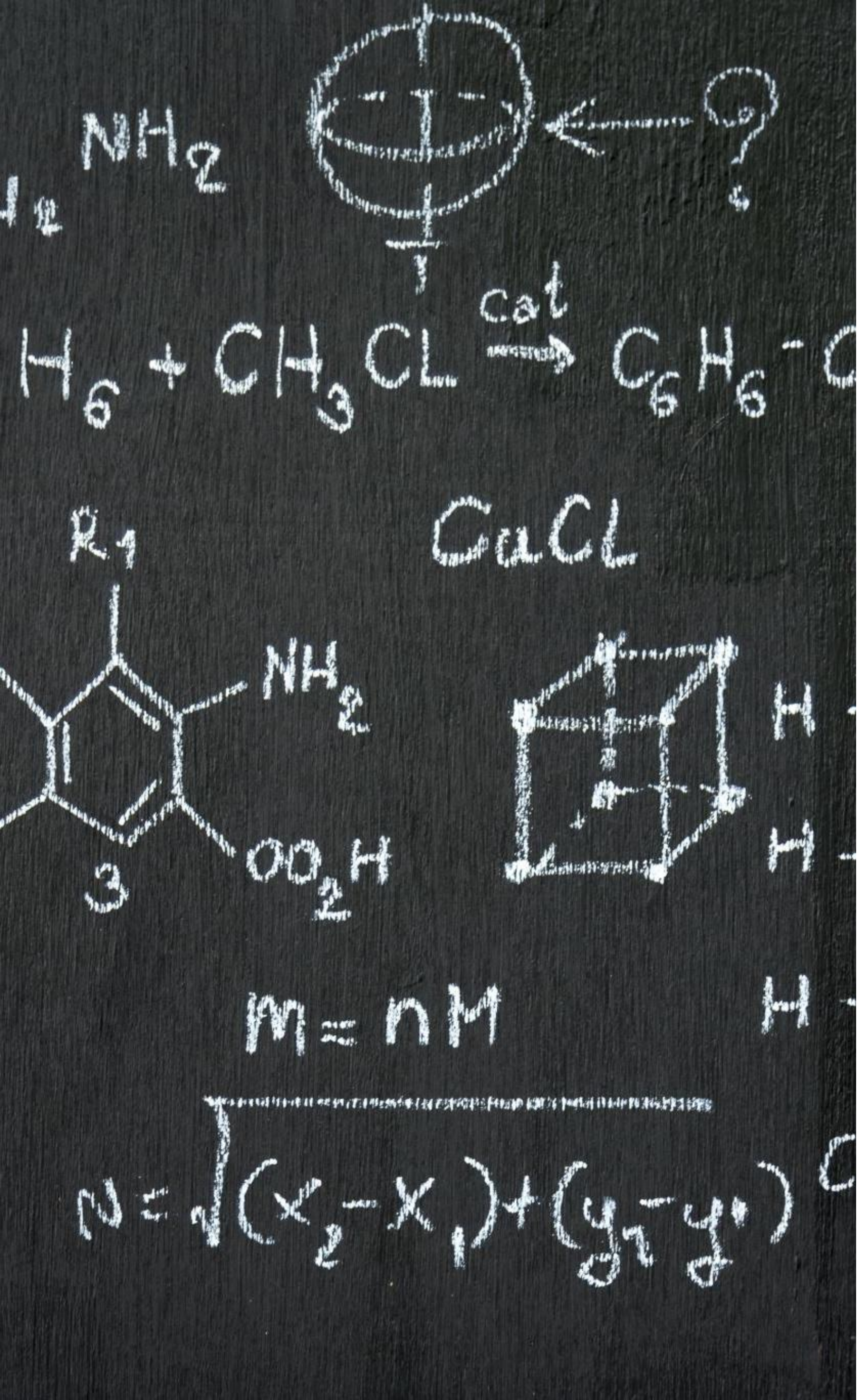




Faça

Desenhe o circuito lógico que executa a seguinte expressão booleana

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



Faça

Desenhe o circuito lógico que executa a seguinte expressão booleana

S = ((A.B)’ + (C.D)’)’

Expressões ou Circuitos representados por Tabelas Verdade

- Uma forma de estudar uma função booleana consiste em utilizar sua tabela verdade
- Como visto anteriormente, há uma equivalência entre o circuito lógico e sua expressão característica
  - Podemos obter um circuito a partir de sua expressão
  - Podemos obter expressões a partir dos circuitos
- Uma tabela verdade representa o comportamento tanto do circuito como de sua expressão característica

Como obter a Tabela Verdade a partir de uma Expressão

- Colocar todas as possibilidades (interpretações) para as variáveis de entrada
  - Lembrar que para N variáveis, há 2 N possibilidades
- Adicionar colunas para cada subfórmula da expressão
  - Preencher cada coluna com seus resultados
- Adicionar uma coluna para o resultado final
  - Preencher essa coluna com o resultado fina

Exemplo

- Considere a expressão S = A.B.C + A.D + A.B.D
- Como há 4 variáveis de entrada (A, B, C, D), há 24=16 interpretações
  - Variação 8 zeros, 8 um
  - Variação 4 zeros, 4 um
  - Variação 2 zeros, 2 um
  - Variação 1 zero, 1 um

Exemplo

- S = A.B.C + A.D + A.B.D
- A seguir, adicionar uma coluna para cada subfórmula de S, além de uma coluna para o resultado final S
  - A.B.C
  - A.D
  - A.B.D
- Preencher cada coluna com seu respectivo resultado

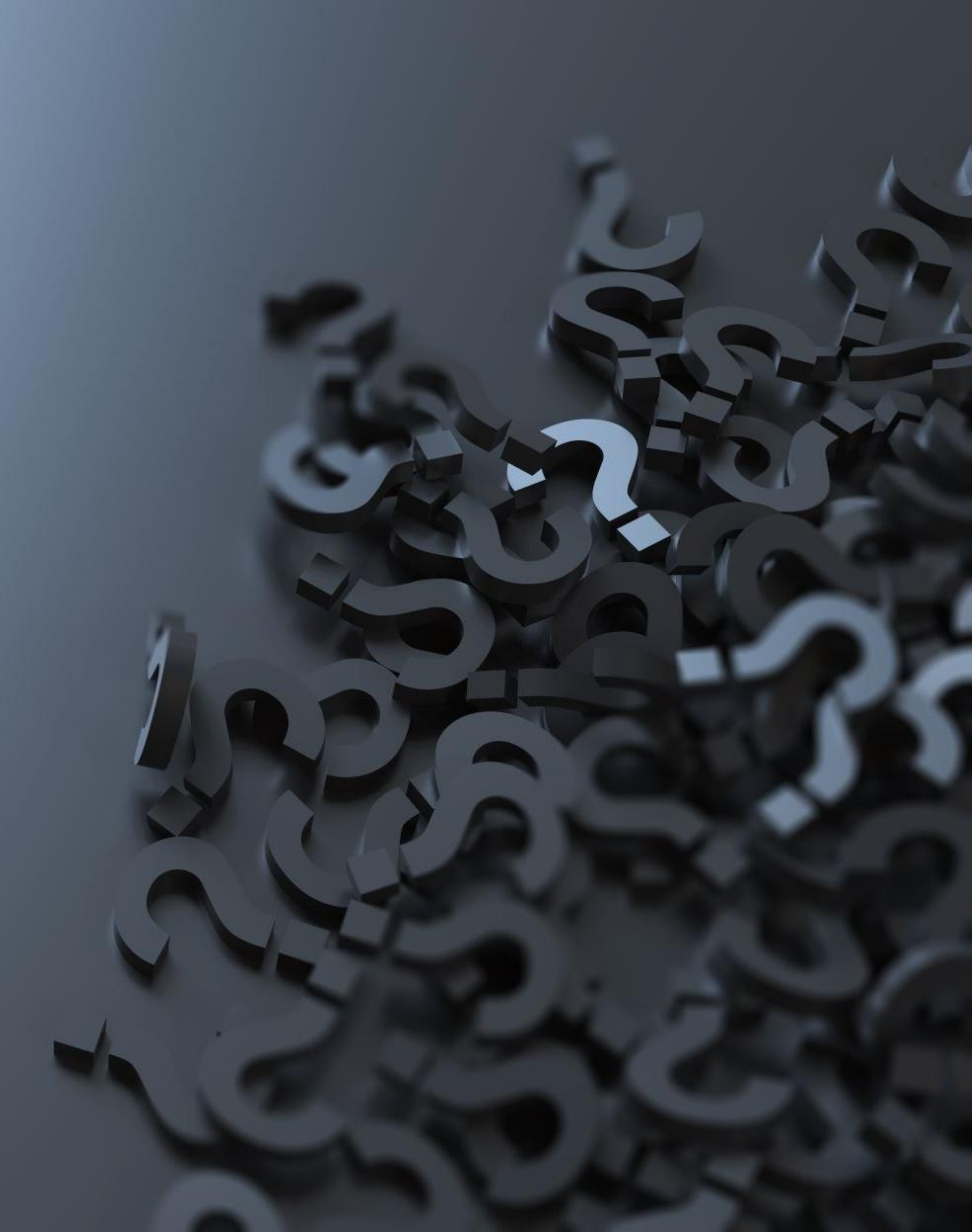
Exemplo

- S = A.B.C + A.D + A.B.D
- Por último, preencher a coluna do resultado final

Resultado

S = A.B.C + A.D + A.B.D

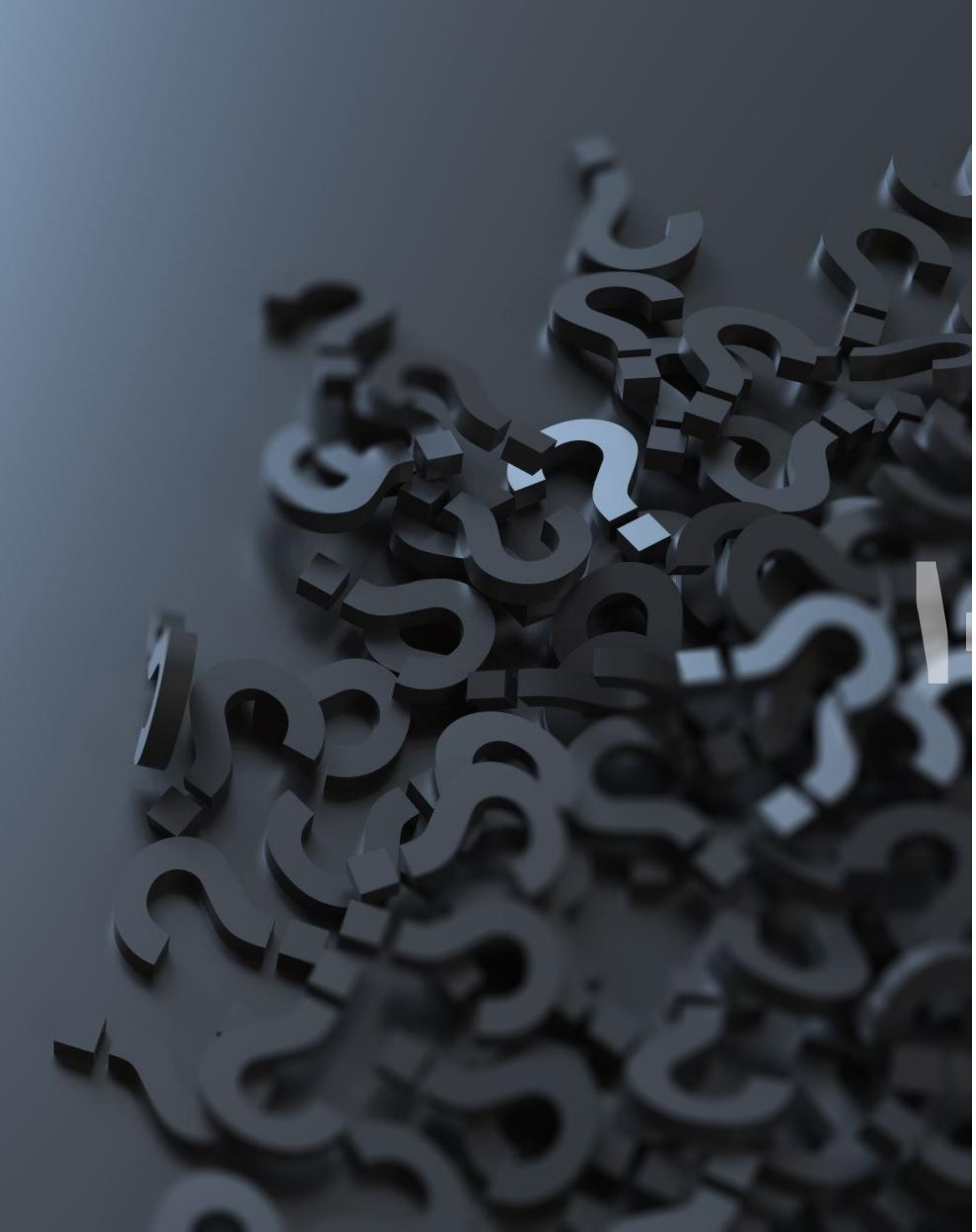
A	B	C	D	A.B.C	A.D	A.B.D	S
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	1
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	0	0
1	1	0	1	0	1	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	1	1	1



Faça

Encontre a tabela verdade da expressão

$S = A'+B+A.B.C'$

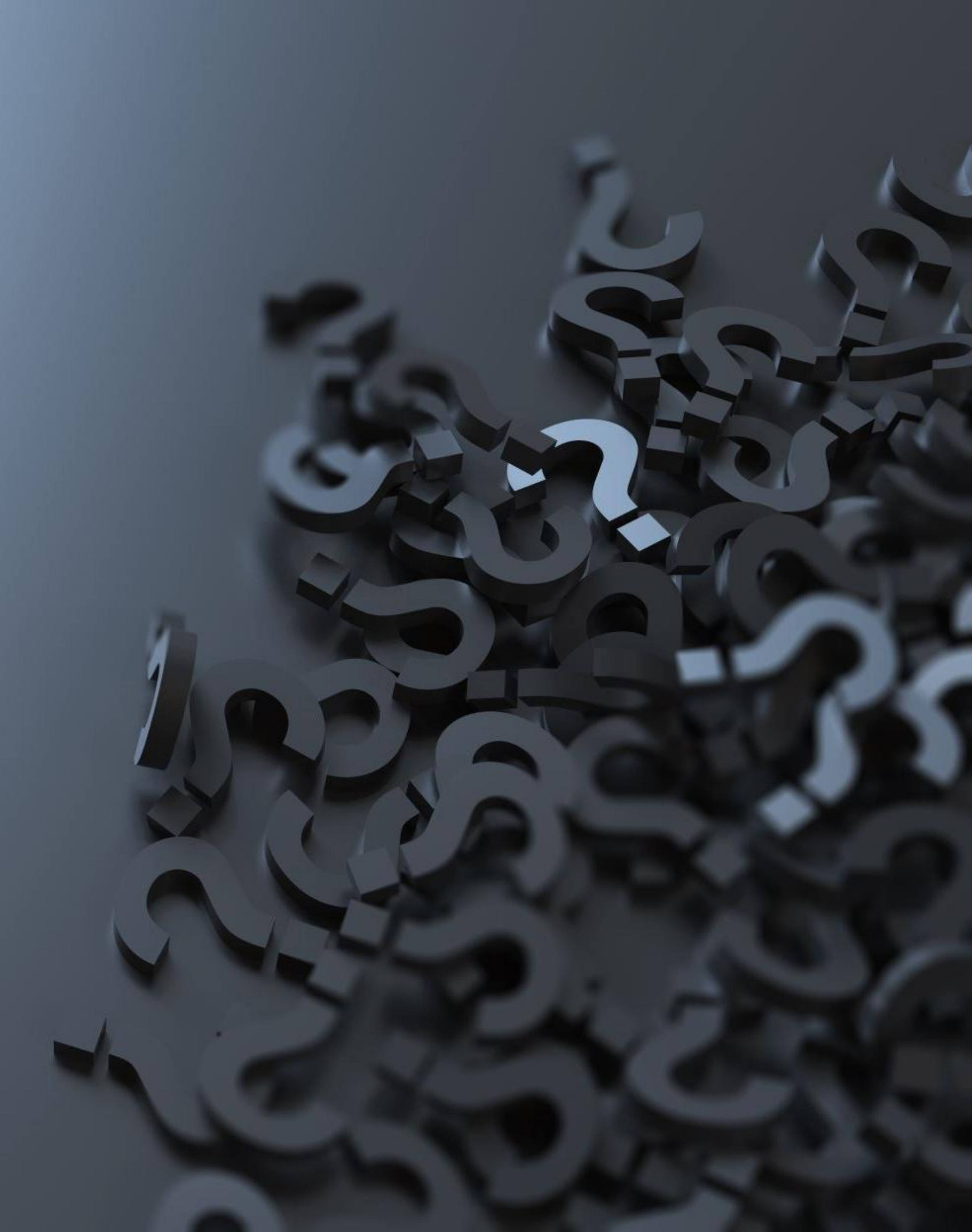


Faça

Encontre a tabela verdade da expressão

S = A’+B+A.B.C’

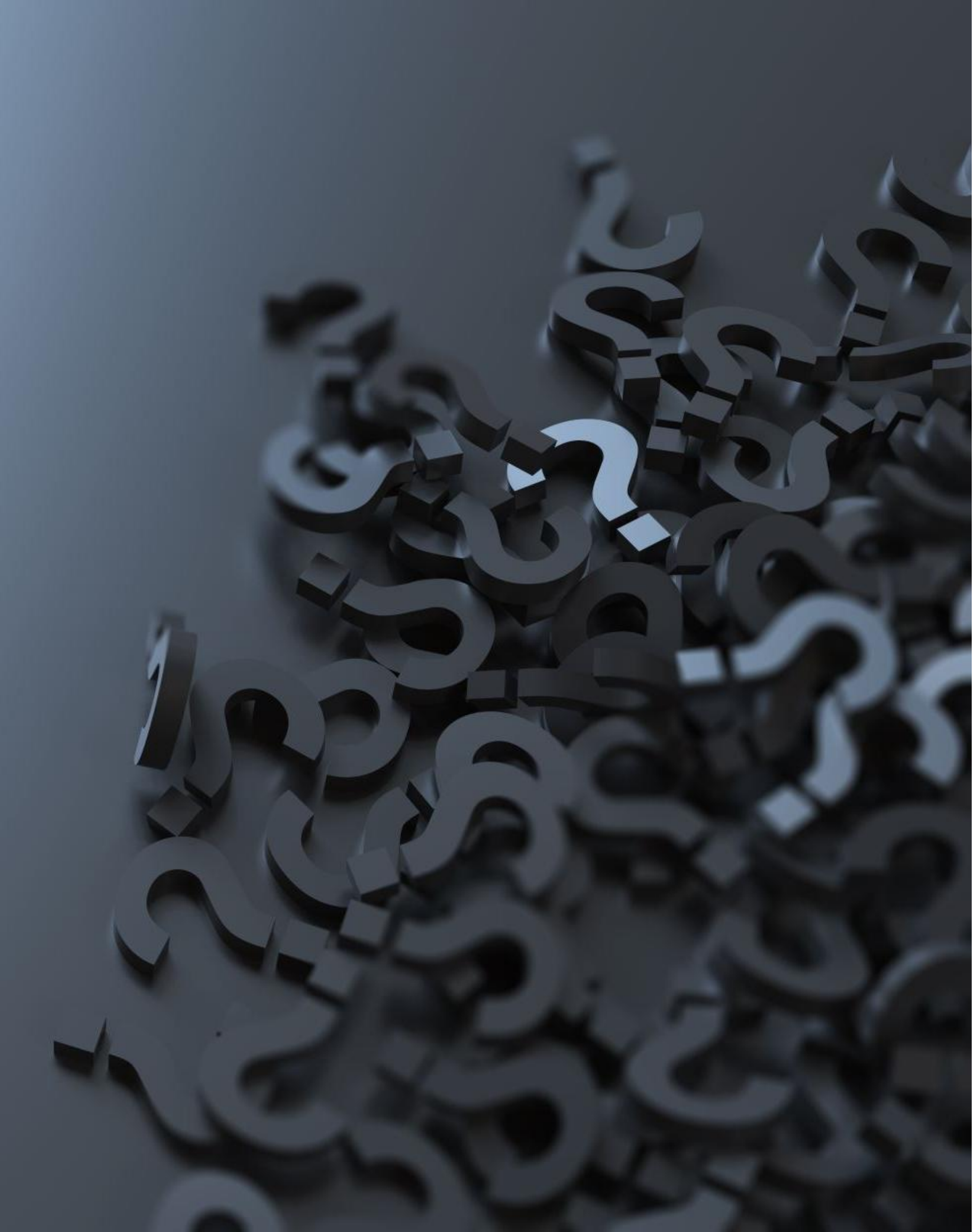
A	B	C	A’	C’	A.B.C’	S
0	0	0	1	1	0	1
0	0	1	1	0	0	1
0	1	0	1	1	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	1	0	0	0	0
1	1	0	0	1	1	1
1	1	1	0	0	0	1



Faça

Montar a tabela verdade da expressão

$S = A.B.C + A.B'.C + A'.B'.C + A'.B'.C'$



Faça

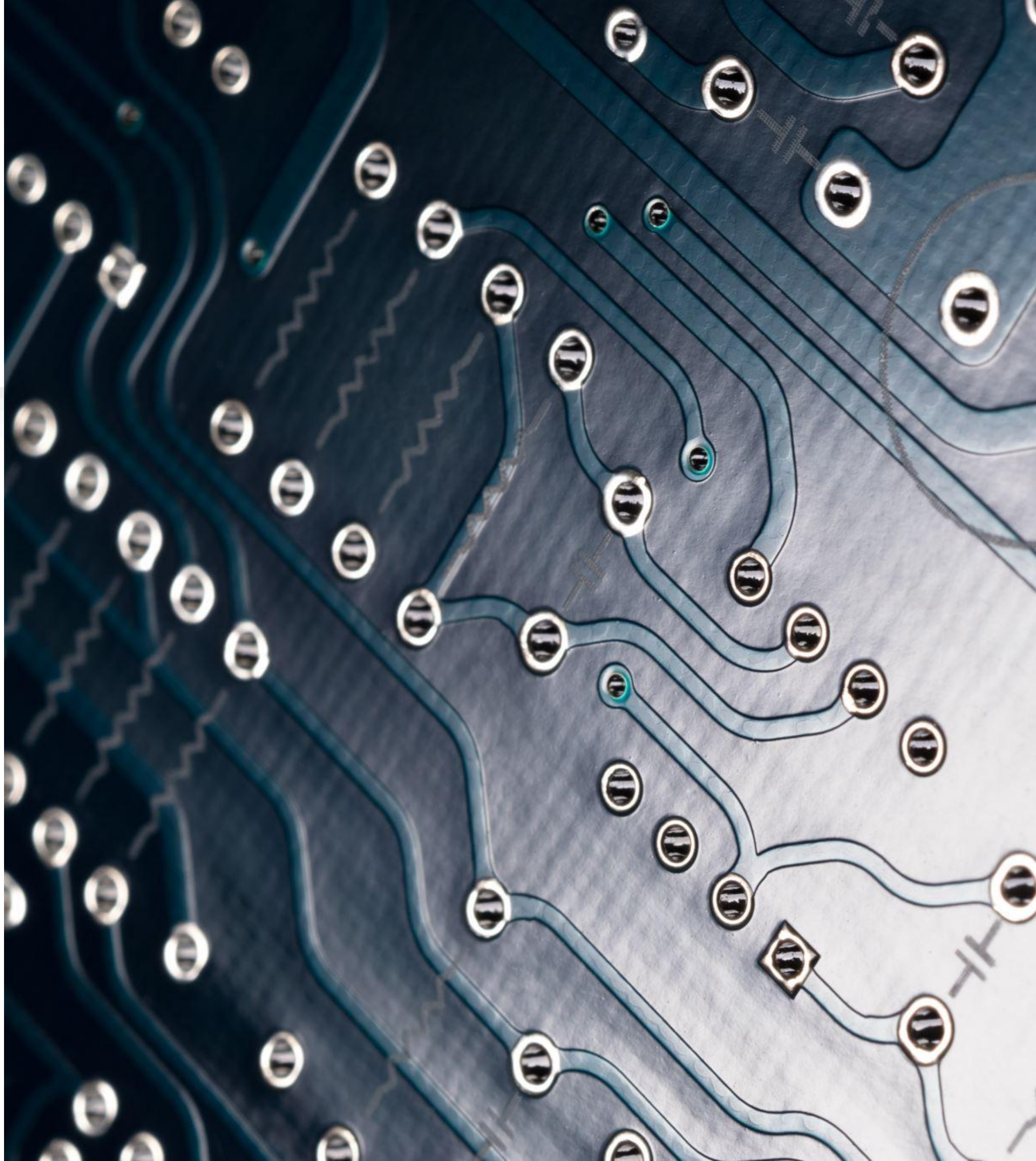
Montar a tabela verdade da expressão

S = A.B.C + A.B'.C + A'.B'.C + A'.B'.C'

A	B	C	A'	B'	C'	A.B.C	A.B'.C	A'.B'.C	A'.B'.C'	S
0	0	0	1	1	1	0	0	0	1	1
0	0	1	1	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0	0
1	0	1	0	1	0	0	1	0	0	1
1	1	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	1	0	0	0	1

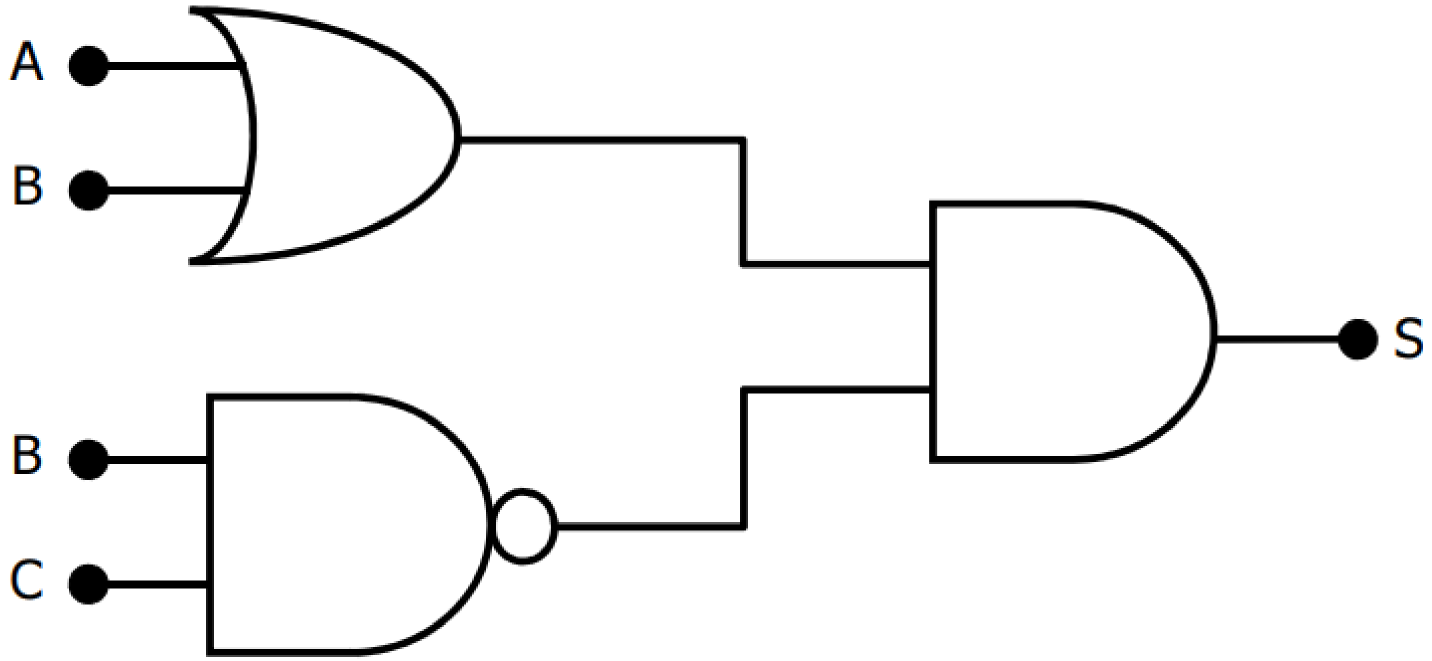
Obtendo a tabela verdade a partir de um circuito

- De forma análoga, é possível estudar o comportamento de um circuito por meio da sua tabela verdade
- Dado um circuito, é necessário extrair sua expressão característica; a partir dela é possível montar a tabela verdade correspondente



Exemplo

A partir do circuito

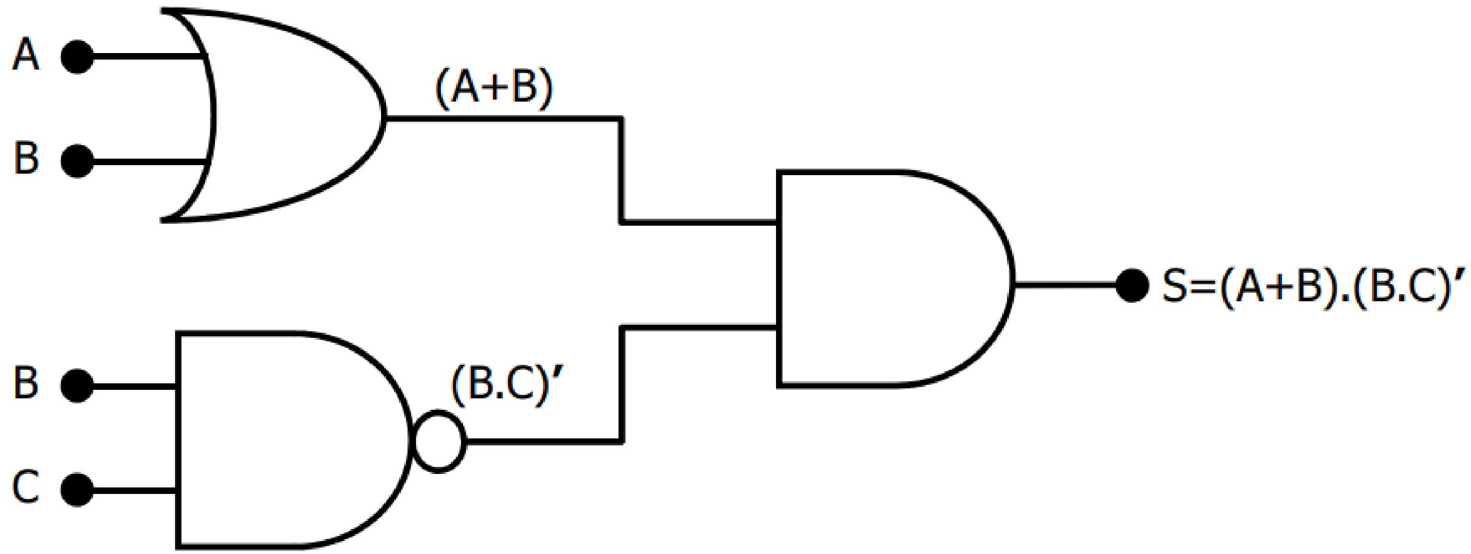


Exemplo

A partir do circuito

Extraímos sua expressão

S = (A+B) . (B.C)’



Exemplo

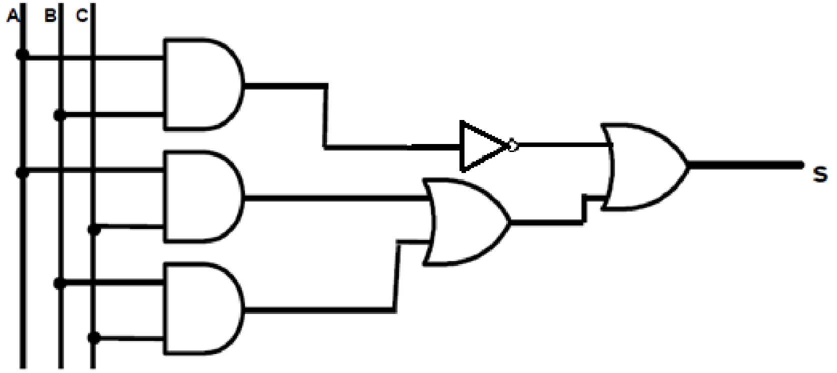
- A partir do circuito
- Extraímos sua expressão
  - S = (A+B) . (B.C)’
- A partir da expressão
- Obtém-se a tabela verdade, como anteriormente explicado

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

Exercicios

- Dada a expressão
  - S = A' + (B . C) + D
- Desenhe o circuito e monte a sua tabela verdade

- Dado o circuito



- Encontre a expressão lógica e faça a sua tabela verdade

Referências

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

Professores são convidados a utilizarem esta apresentação da maneira que lhes for conveniente, desde que esta nota de *copyright* permaneça intacta.

Slides baseados em:

- 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.