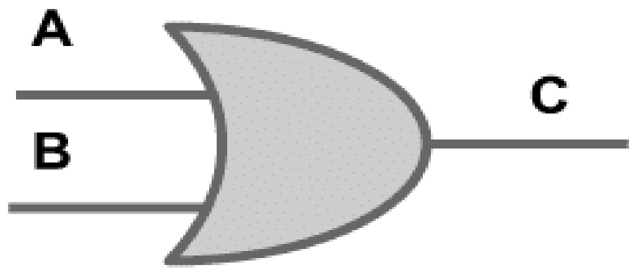
Portas Lógicas

Osmar de Oliveira Braz Junior
Márcia Cargnin Martins Giraldi



Objetivos

- Apresentar porta lógicas e sua função na composição de circuitos digitais;
- Apresentar as funções E, OU e NÂO suas representações gráficas e matemáticas;
- Mostrar a correspondência entre expressões booleanas e circuitos lógicos.

Histórico

- 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.

Introdução

- Um computador digital é uma máquina projetada para armazenar e manipular informações representadas através de **dígitos binários** (bits).
- A informação **binária** (0 ou 1) é representada em um sistema digital por quantidades físicas, sinais elétricos, os quais são gerados e mantidos internamente ou recebidos de elementos externos, em **dois níveis de intensidade**, cada um correspondendo a um valor binário.

Portas Lógicas

- Uma **porta lógica** é um circuito eletrônico, portanto uma peça de hardware, que se constitui no elemento básico e mais elementar de um sistema de computação.
- Grande parte do hardware do sistema é fabricado através da adequada combinação de **milhões desses elementos**, como a CPU, memória, interfaces de E/S, etc..

Portas Lógicas

- Há diversos tipos bem definidos de **portas lógicas**, cada uma delas sendo capaz de implementar uma operação ou função lógica específica.
- Uma **operação lógica** realizada sobre um ou mais valores lógicos produz um certo resultado (também um valor lógico), conforme a regra definida para a específica operação lógica.

Portas Lógicas

- As **portas lógicas** são circuitos criados para manipular um bit ou um conjunto de bits em sua entrada de forma a fornecer um sinal (também binário) na saída de acordo com o estado dos bits da entrada.

Portas Lógicas

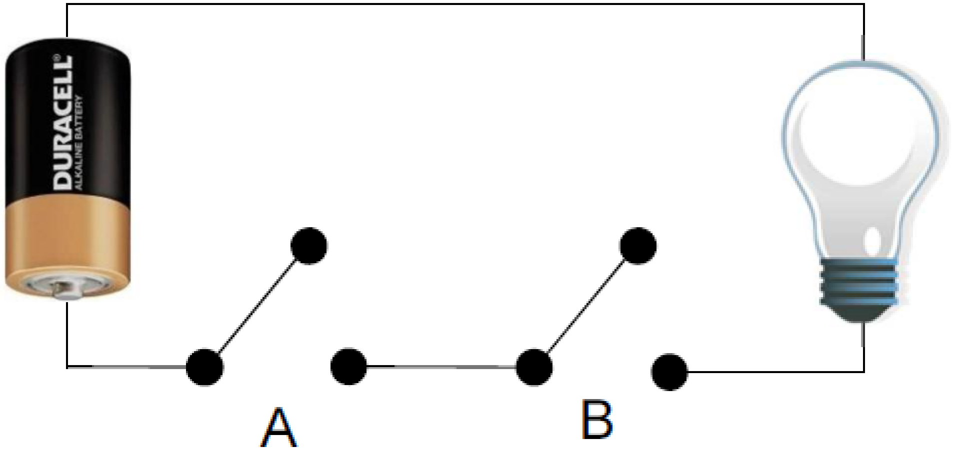
- Apesar de ser possível imaginar inúmeras portas lógicas diferentes que manipulam bits de entrada para produzir uma saída específica, existem 3 portas lógicas elementares que, quando combinadas, são responsáveis por criar praticamente qualquer lógica digital.
  - Porta NOT
  - Porta OR
  - Porta AND

Portas Lógicas

- Uma **operação lógica** produz um resultado que pode assumir somente dois valores (0 ou 1), os quais são relacionados na álgebra booleana às declarações FALSO (F=0) ou VERDADEIRO (V=1).
- Se as variáveis de entrada só podem assumir os valores F ou V e se o **resultado também**, então podemos definir previamente todos os possíveis valores de resultado de uma dada operação lógica, conforma a combinação possível de valores de entrada.

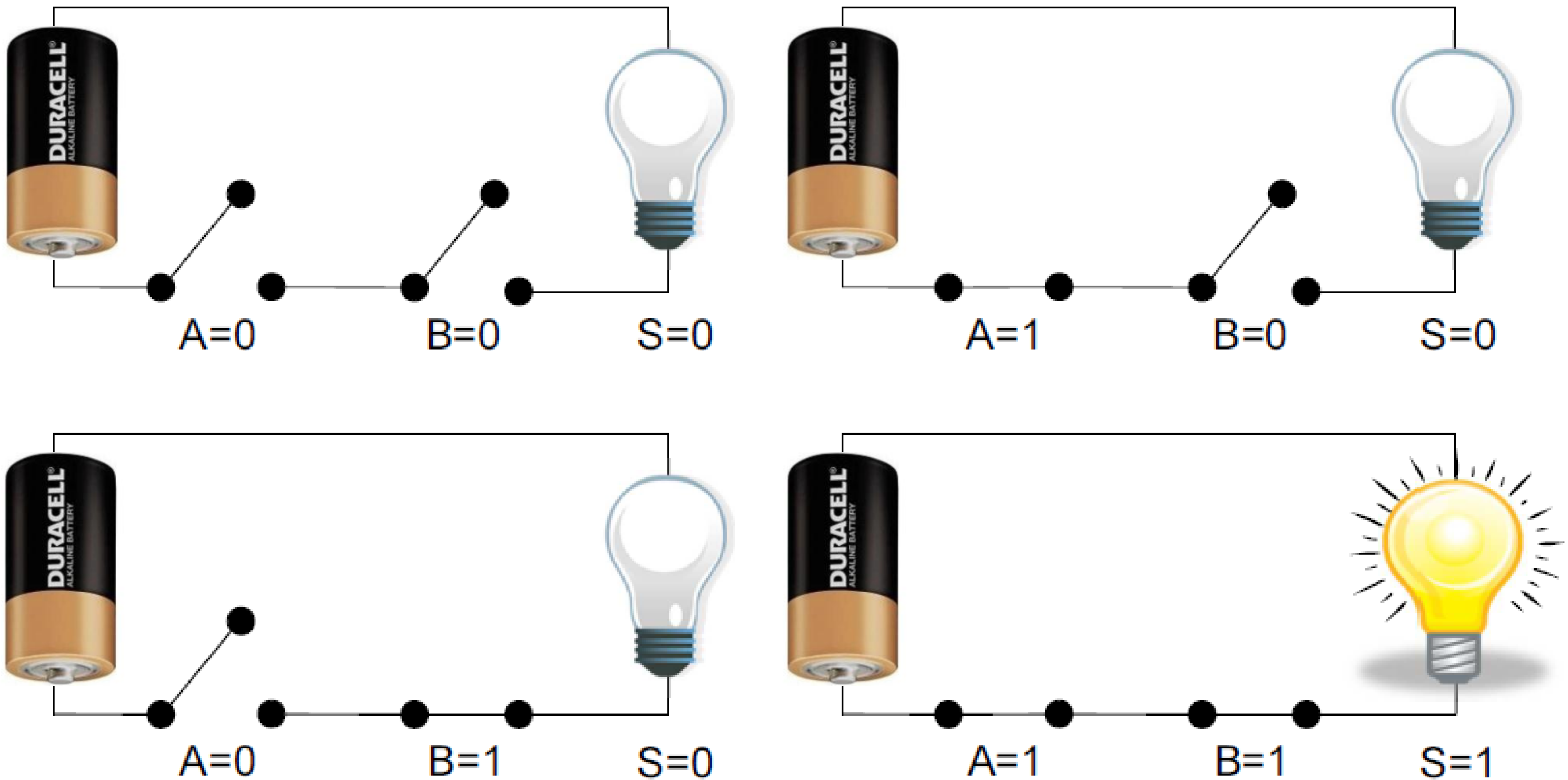
Função E (AND)

- Executa a multiplicação (conjunção, circuito em série) 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



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

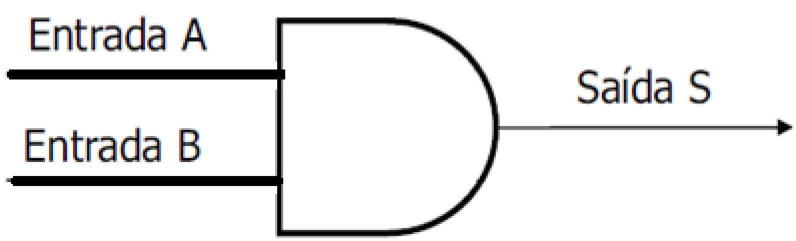
- As possibilidades das variáveis são representadas de forma tabular e chama-se o conjunto de **Tabela Verdade**.
- Cada **operação lógica** possui sua **própria** tabela verdade, estabelecida de acordo com a regra que define a respectiva operação lógica.
- De um modo geral, a **tabela verdade** de uma dada operação lógica possui **2^n linhas** ou combinações de valores de entrada, sendo **n** igual à quantidade de elementos(variáveis) de entrada.
  - Exemplo anterior, para 2 variáveis booleanas (A e B), há 4 interpretações possíveis

Tabela Verdade da Função E(AND)

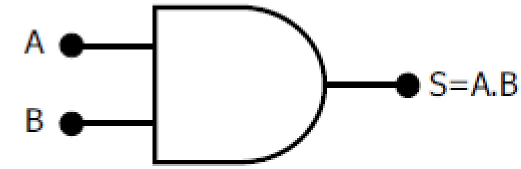
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

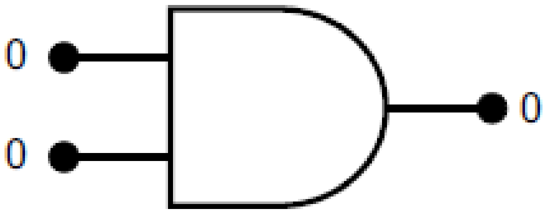
Porta Lógica E(AND)

- 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

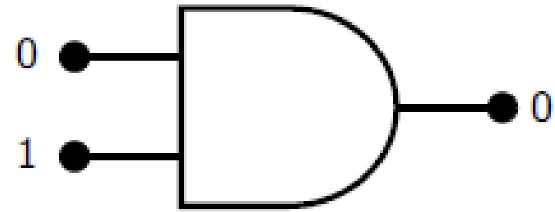


Porta Lógica E(AND)

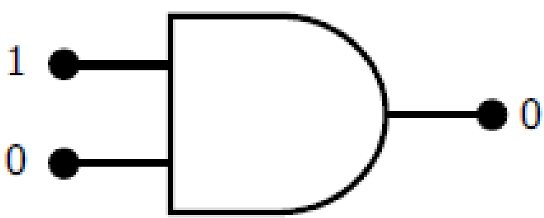




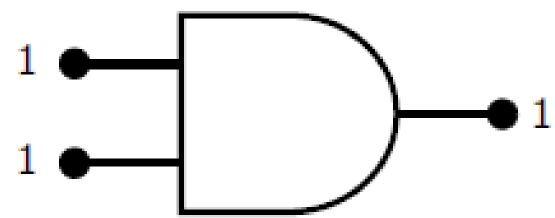
A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1



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



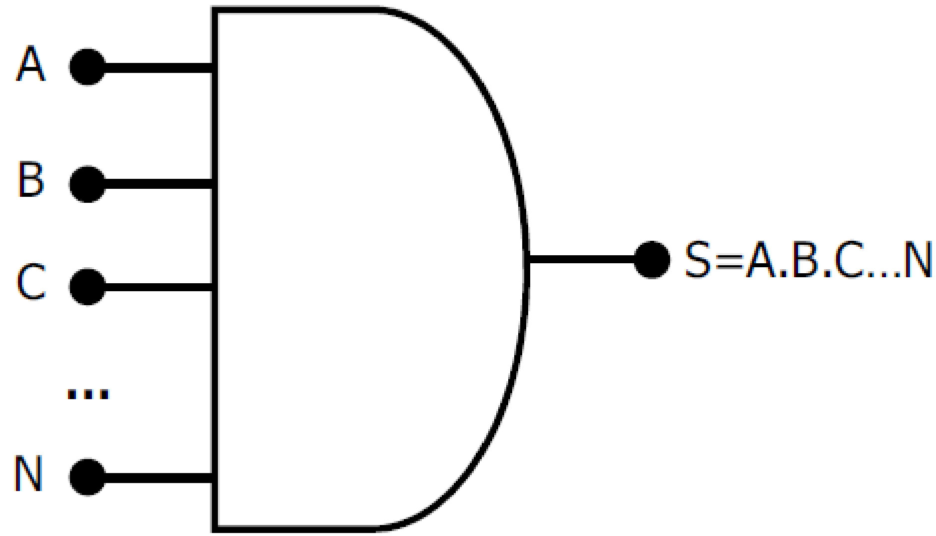
A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1



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

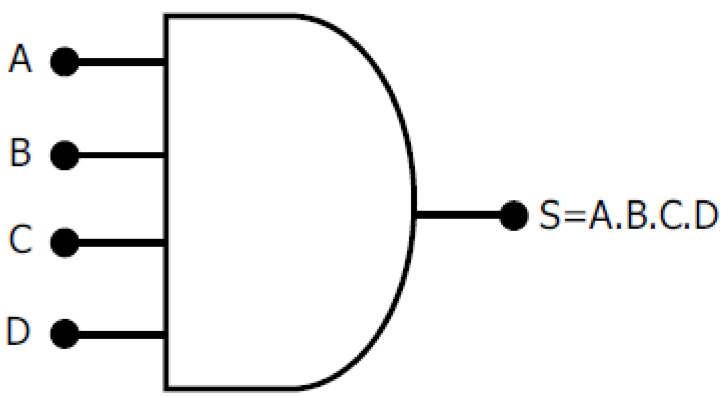
Porta Lógica E(AND)

- 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



Porta Lógica E(AND)

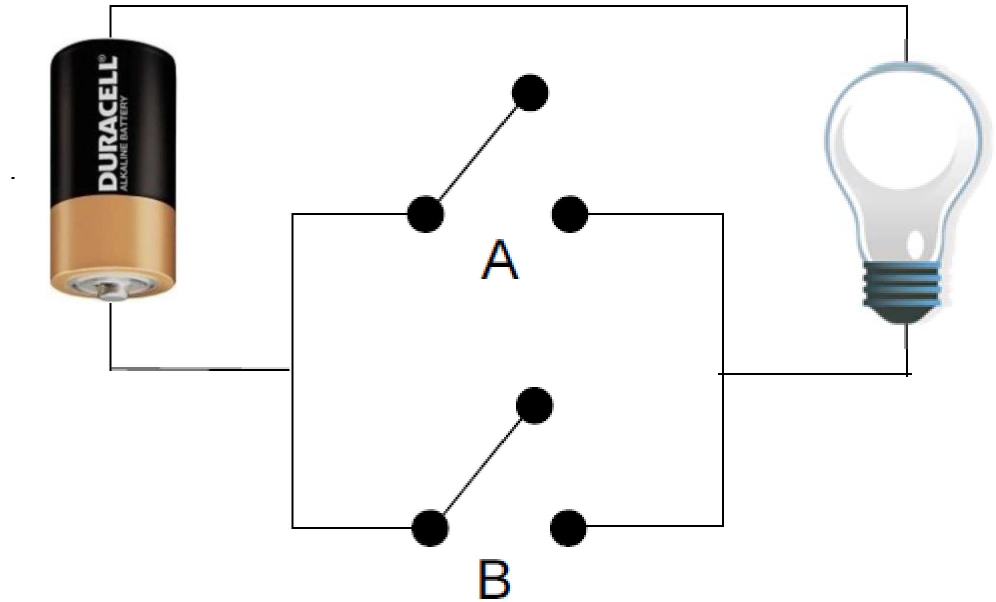
- Por exemplo,
  - S=A.B.C.D



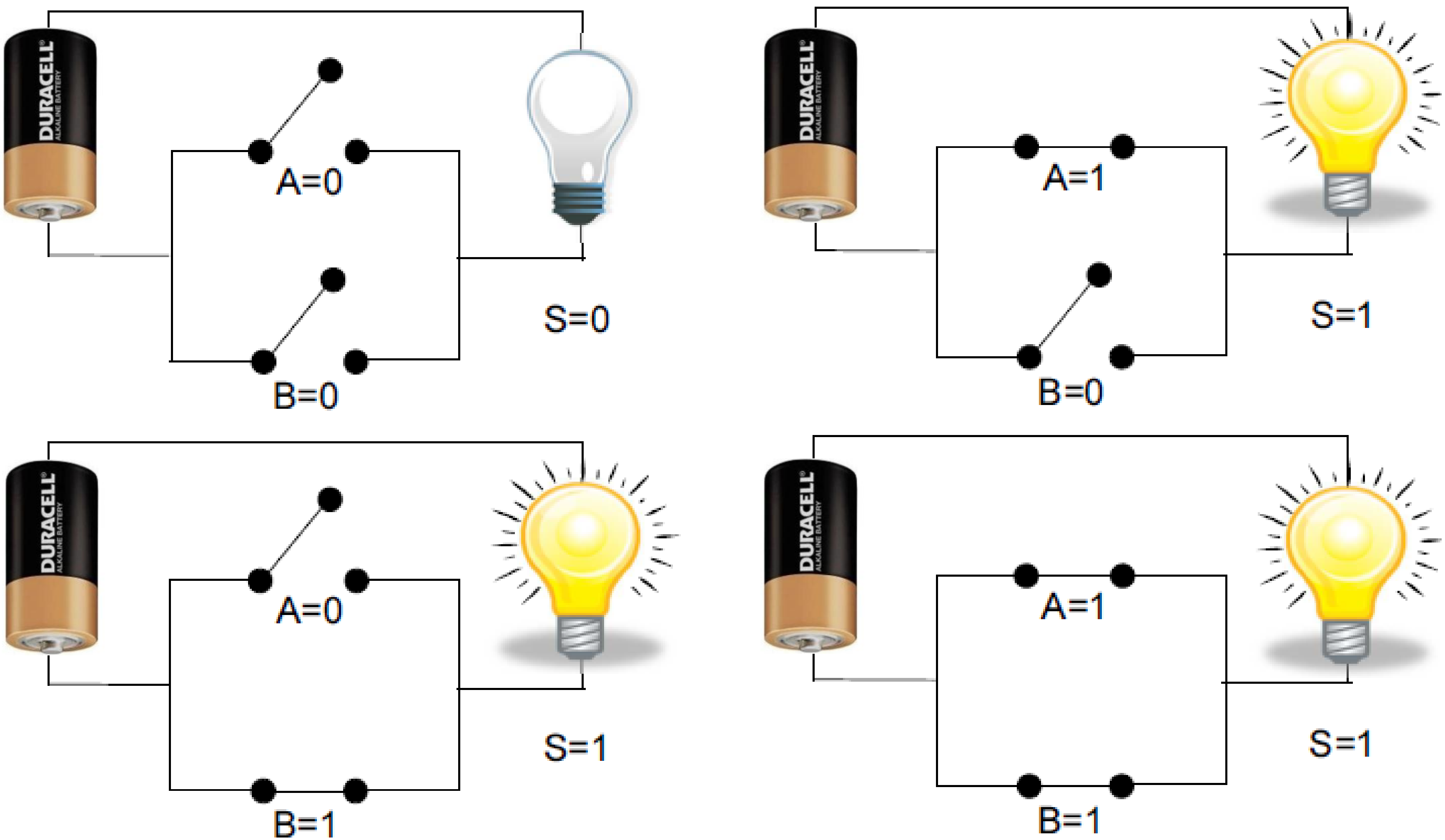
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Função OU (OR)

- Executa a soma (disjunção, circuito em paralelo) 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



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 | B
  - S = A; B
  - S = A v B

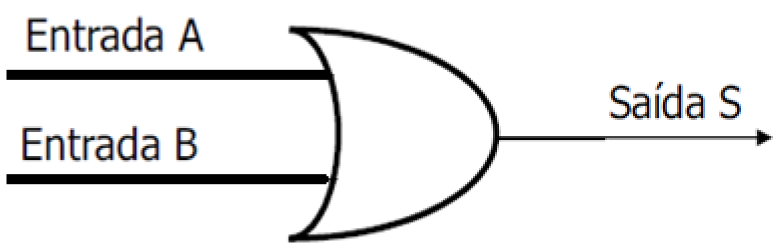
Tabela Verdade da Função OU (OR)

- 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)

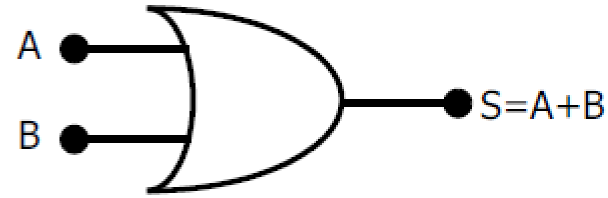
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

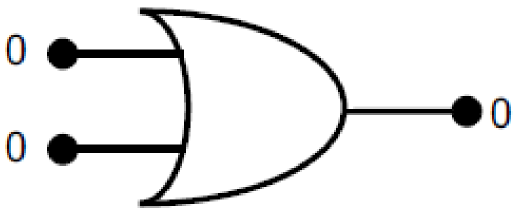
Porta Lógica OU (OR)

- 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

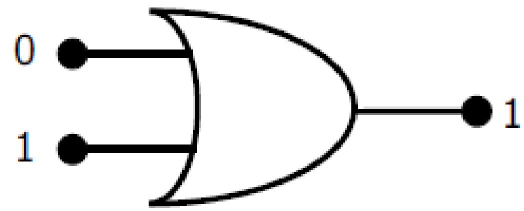


Porta Lógica OU (OR)

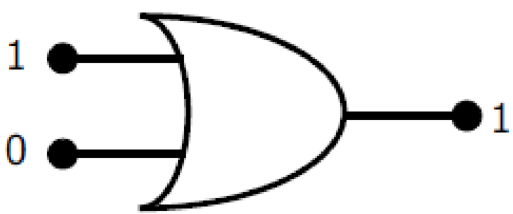




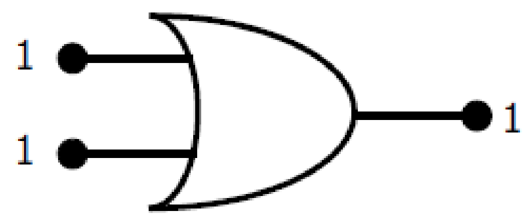
A	B	S=A+B
0	0	0
0	1	1
1	0	1
1	1	1



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



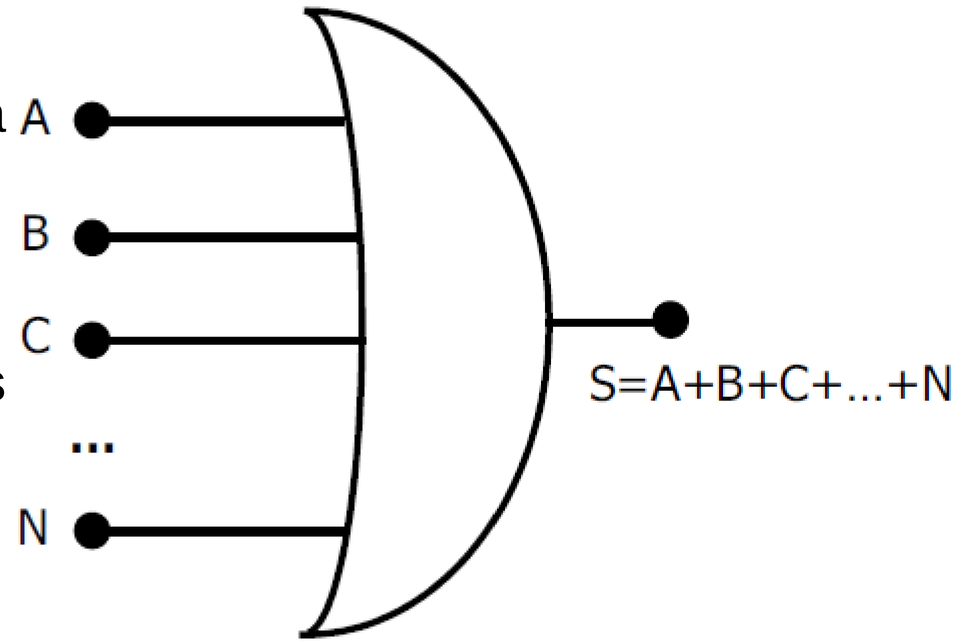
A	B	S=A+B
0	0	0
0	1	1
1	0	1
1	1	1



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

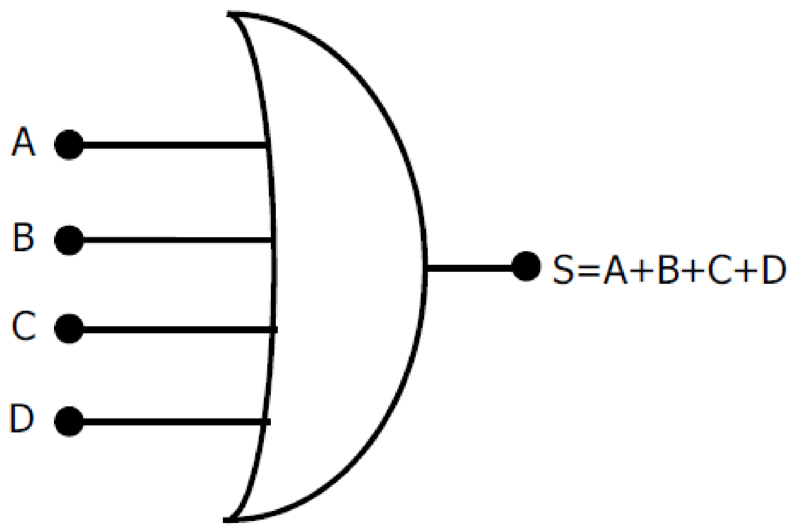
Porta Lógica OU (OR)

- 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 1



Porta Lógica OU (OR)

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



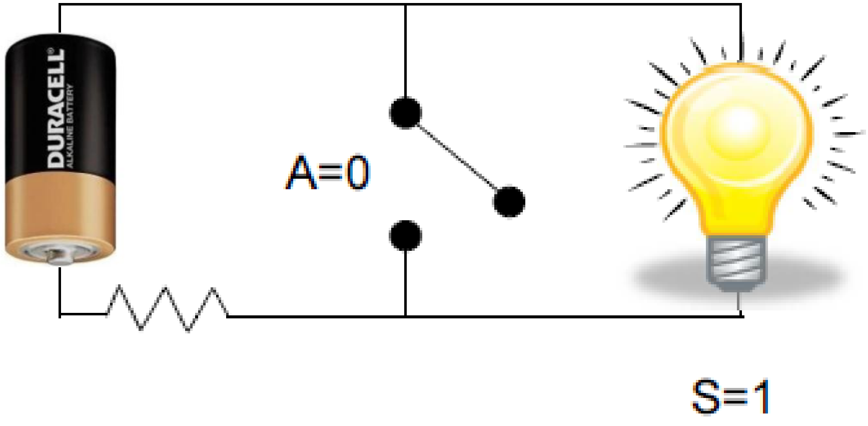
A	B	C	D	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

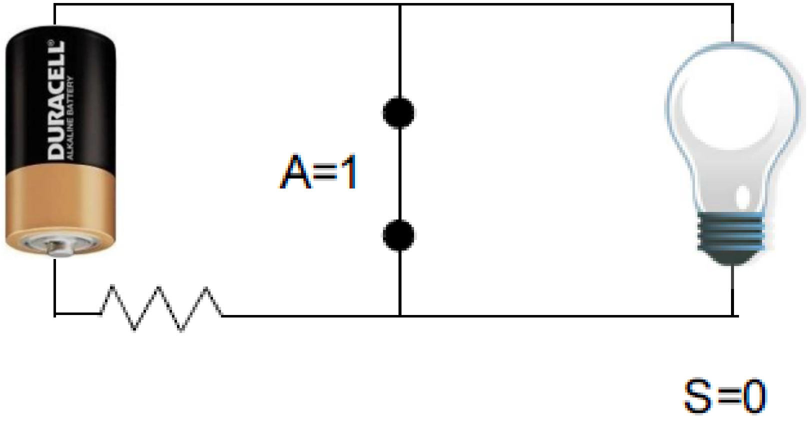
Função NÃO(NOT) – Inversor

- A função NOT é também chamada de ***inversor, ou inversora*** ou ***função complemento***.
- Ela executa a inversão do valor de um sinal binário colocado em sua entrada, produzindo na saída o sinal oposto. É um circuito lógico que requer apenas um valor de entrada.
  - Se a variável estiver em 0, o resultado da função é 1
  - Se a variável estiver em 1, o resultado da função é 0

Função NÃO(NOT) – Inversor

- 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) – Inversor

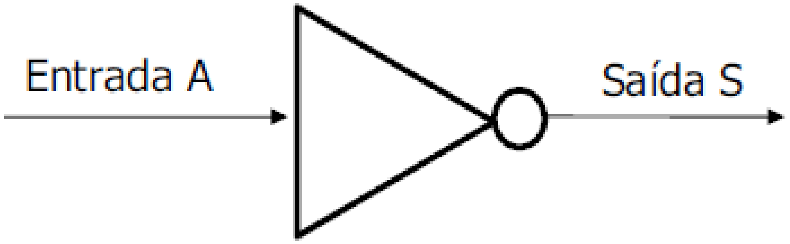
- Para representar a expressão
  - S = não A
- Adotaremos a representação $S = \bar{A}$, onde se lê S = não A
- Notações alternativas
  - S = A’
  - $S = \neg A$
  - $S = \tilde{A}$

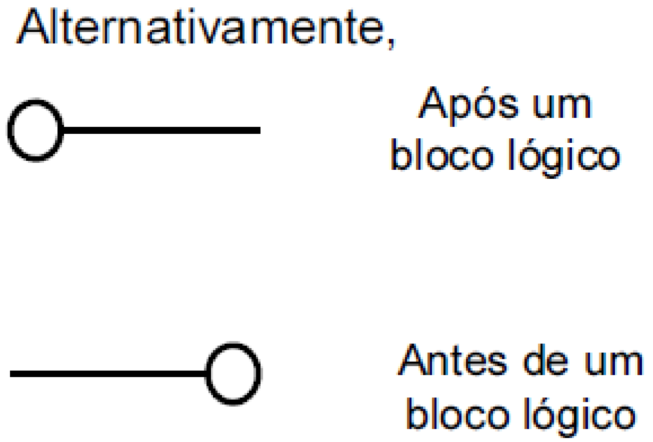
- Tabela verdade da função NÃO (NOT)

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

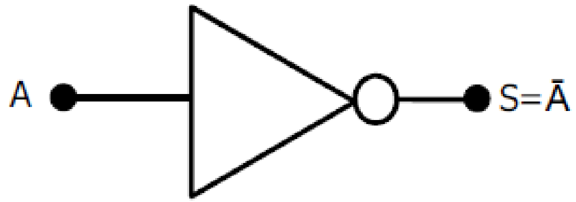
Porta Lógica NÃO(NOT)

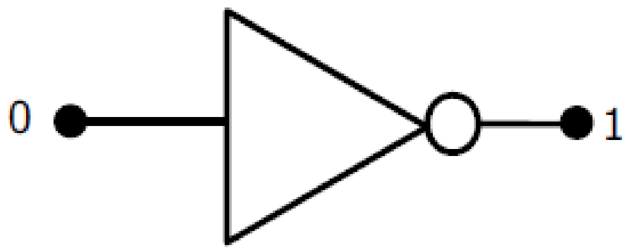
- A porta lógica NÃO, ou inversor, é o circuito que executa a função NÃO
- O inversor executa a tabela verdade da função NÃO
  - Se a entrada for 0, a saída será 1; se a entrada for 1, a saída será 0
- Representação



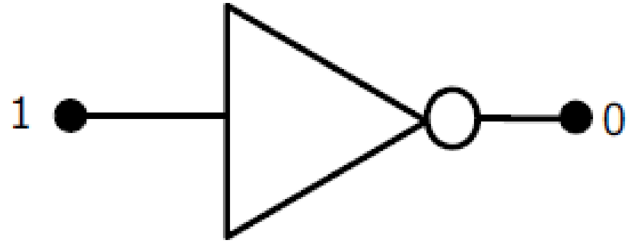


Porta Lógica NÃO(NOT)





A	S=Ā
0	1
1	0



A	S=Ā
0	1
1	0

Função NÃO E(NAND)

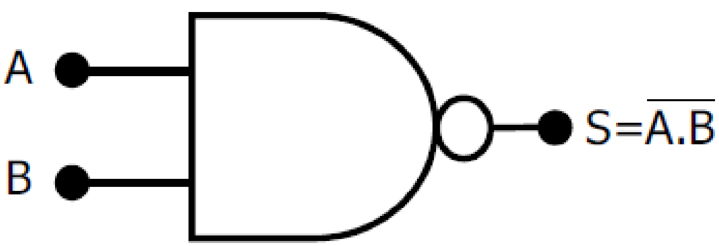
- Composição da função E com a função NÃO, ou seja, a saída da função E é invertida
  - $S = \overline{(A.B)} = \overline{A.B}$
  - $= (A.B)'$
  - $= \neg(A.B)$

- Tabela Verdade

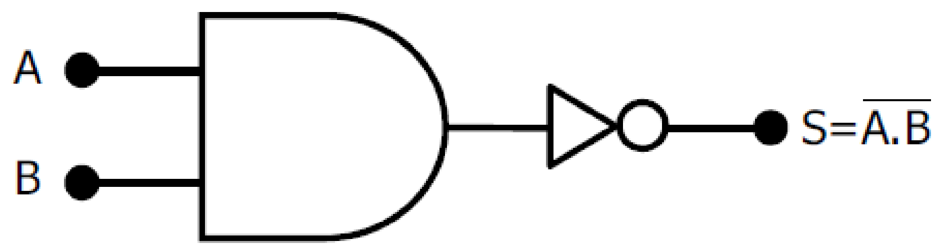
A	B	$S=\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Porta NÃO E(NAND)

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

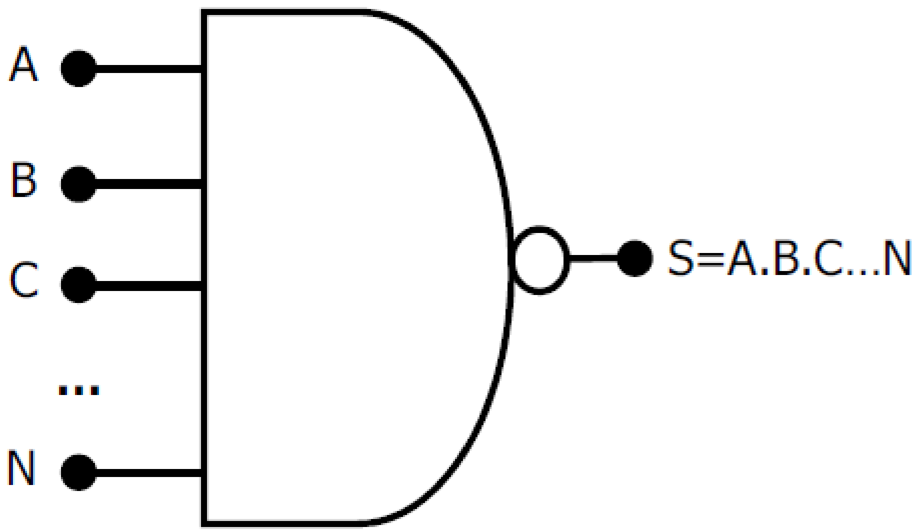


ou



Porta NÃO E(NAND)

- 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)

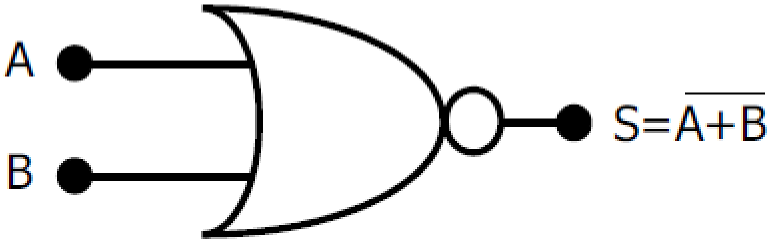
- Composição da função **OU** com a função **NÃO**, ou seja, a saída da função **OU** é **invertida**
  - $S = \overline{(A+B)} = \overline{A+B}$
  - $= (A+B)'$
  - $= \neg(A+B)$

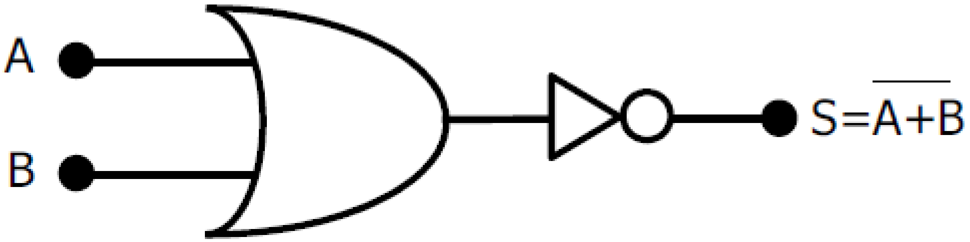
- Tabela Verdade

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

Porta NÃO OU(NOR)

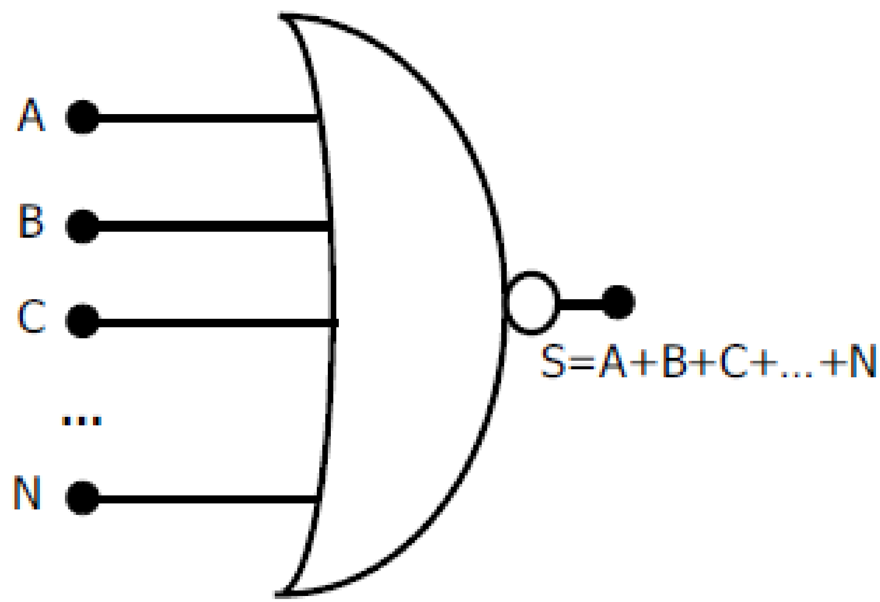
- A porta NÃO OU (NOR) é o bloco lógico que executa a função NÃO OU, ou seja, sua tabela verdade
- Representação





Porta NÃO OU(NOR)

- 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)

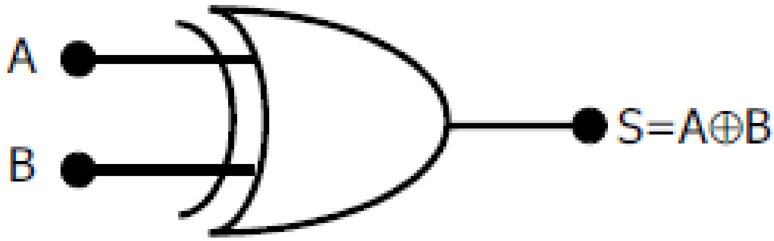
- A função OU Exclusivo
  - fornece 1 na saída quando as entradas forem diferentes entre si e 0 caso contrário
- $S = A \oplus B$
  $= \bar{A}.B + A.\bar{B}$

- Tabela Verdade

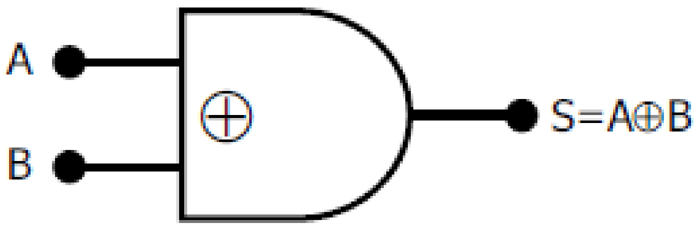
A	B	$S=A\oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

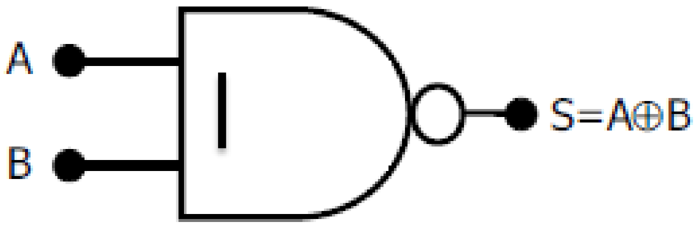
Porta OU-Exclusivo(XOR) como bloco básico

Simbologia utilizada

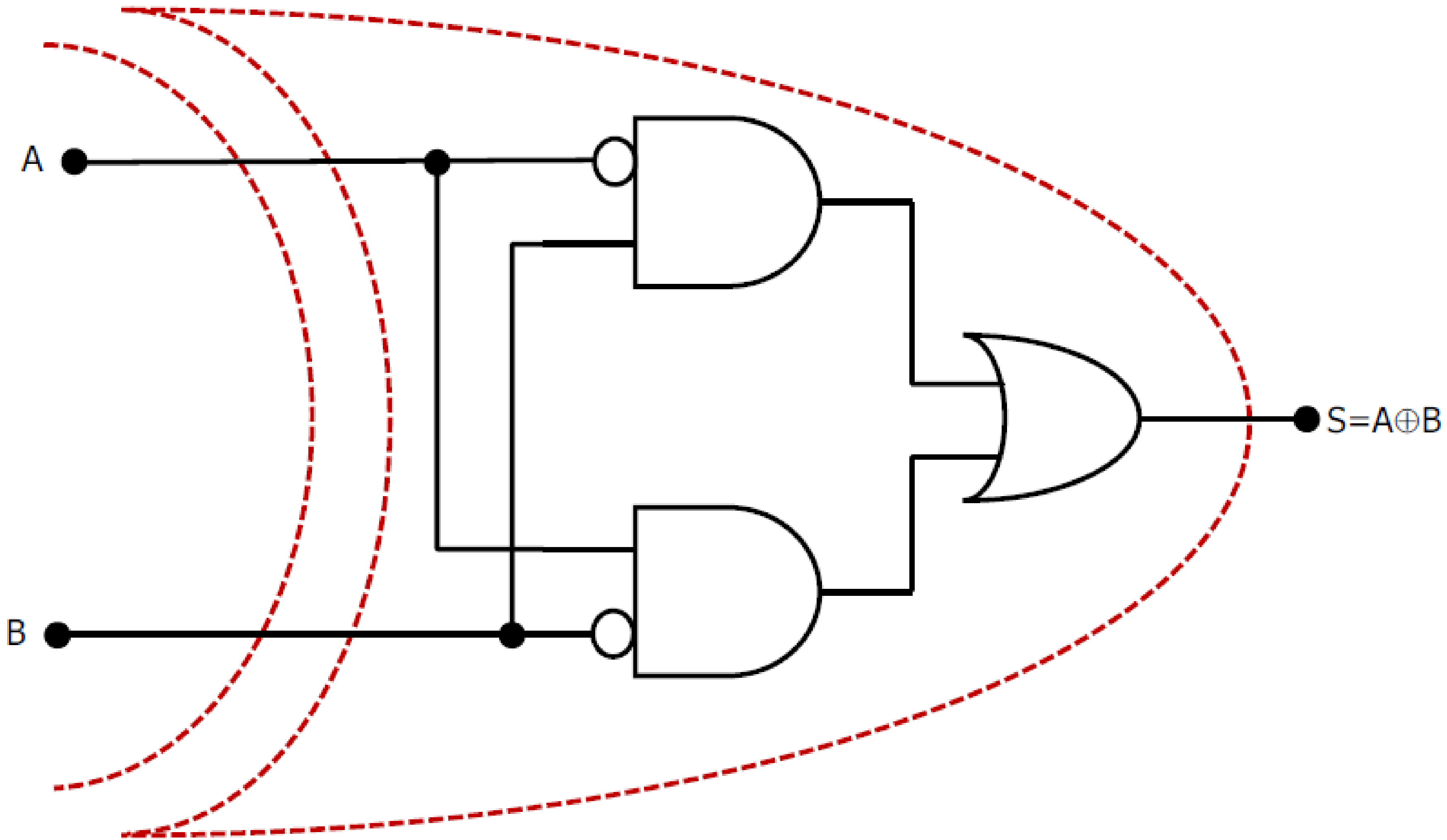


Outras simbologias

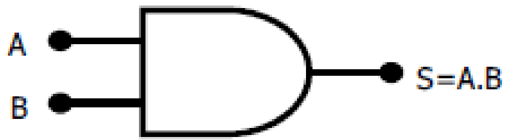
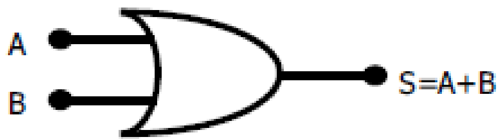
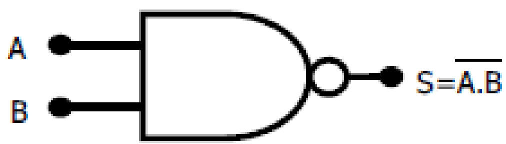
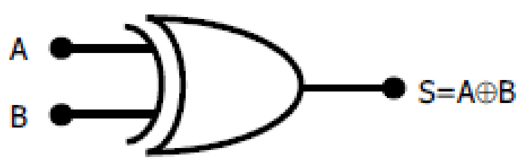




Porta OU-Exclusivo(XOR) como Circuito Combinacional



Resumo dos Blocos Lógicos Básicos

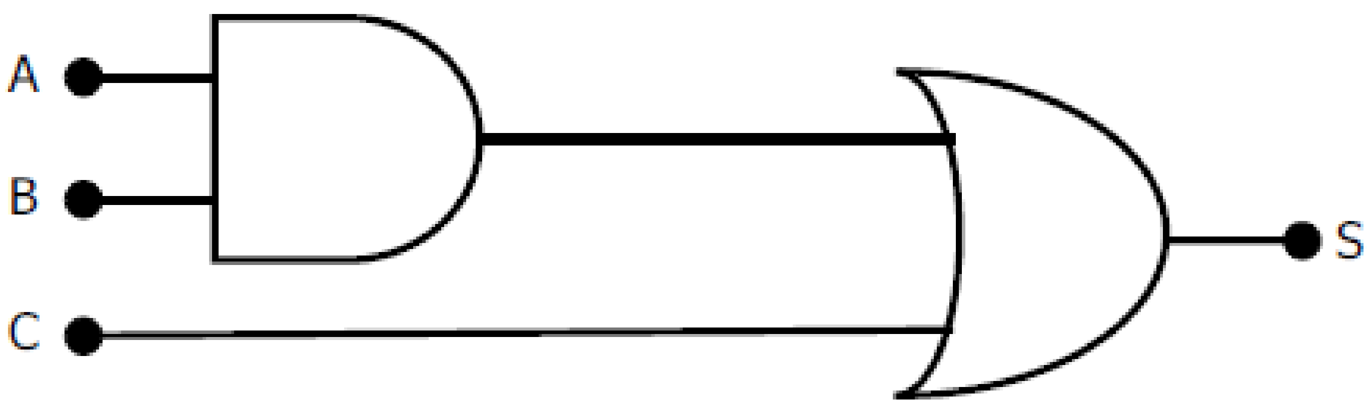
Nome	Símbolo Gráfico	Função Algébrica	Tabela Verdade
E (AND)		S=A.B S=AB	A B S=A.B 0 0 0 0 1 0 1 0 0 1 1 1
OU (OR)		S=A+B	A B S=A+B 0 0 0 0 1 1 1 0 1 1 1 1
NÃO (NOT) Inversor		S=$\bar{A}$ S=A' S= $\neg$A	A S=$\bar{A}$ 0 1 1 0
NE (NAND)		S=$\overline{A.B}$ S=(A.B)' S= $\neg$(A.B)	A B S=$\overline{A.B}$ 0 0 1 0 1 1 1 0 1 1 1 0
NOU (NOR)		S=$\overline{A+B}$ S=(A+B)' S= $\neg$(A+B)	A B S=$\overline{A+B}$ 0 0 1 0 1 0 1 0 0 1 1 0
XOR		S=A$\oplus$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

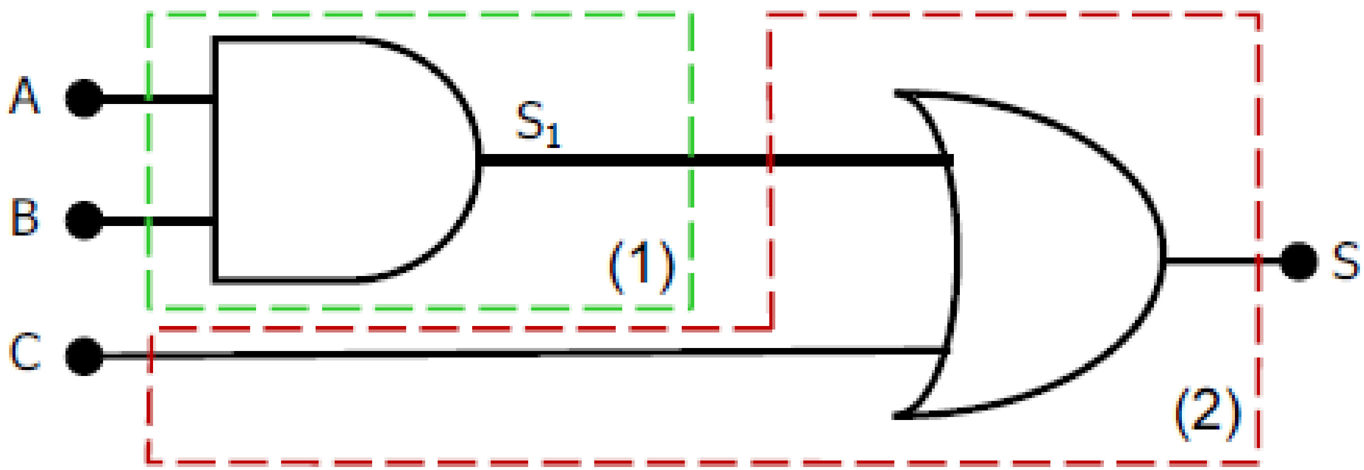
Expressões Booleanas Geradas por Circuitos Lógicos

- Seja o circuito:



Expressões Booleanas Geradas por Circuitos Lógicos

- 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, S1 = A.B

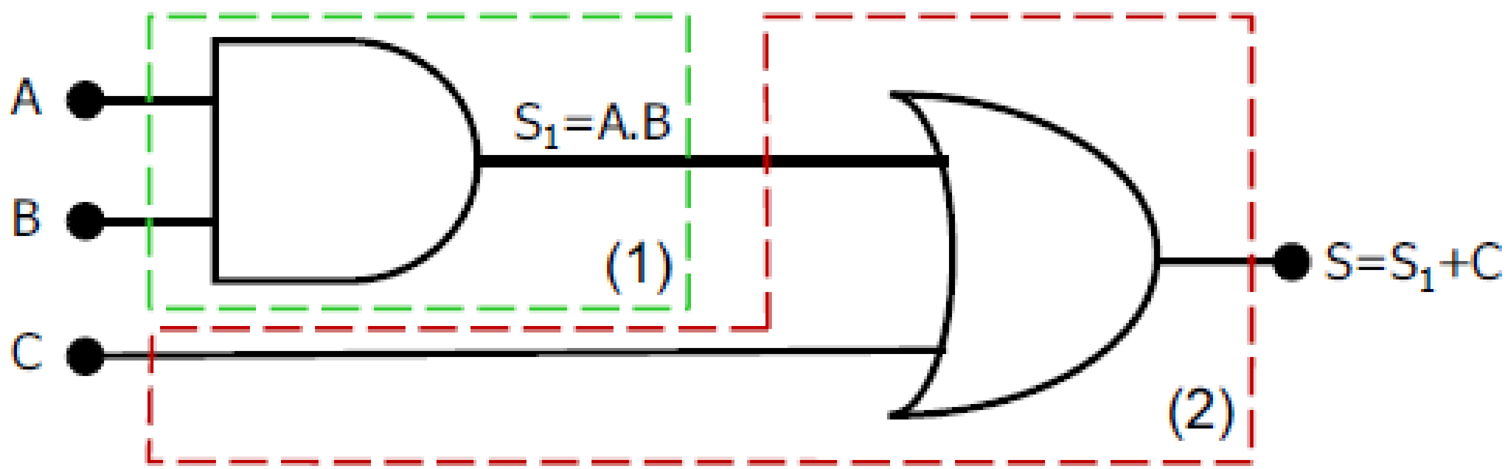


Expressões Booleanas Geradas por Circuitos Lógicos

- 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

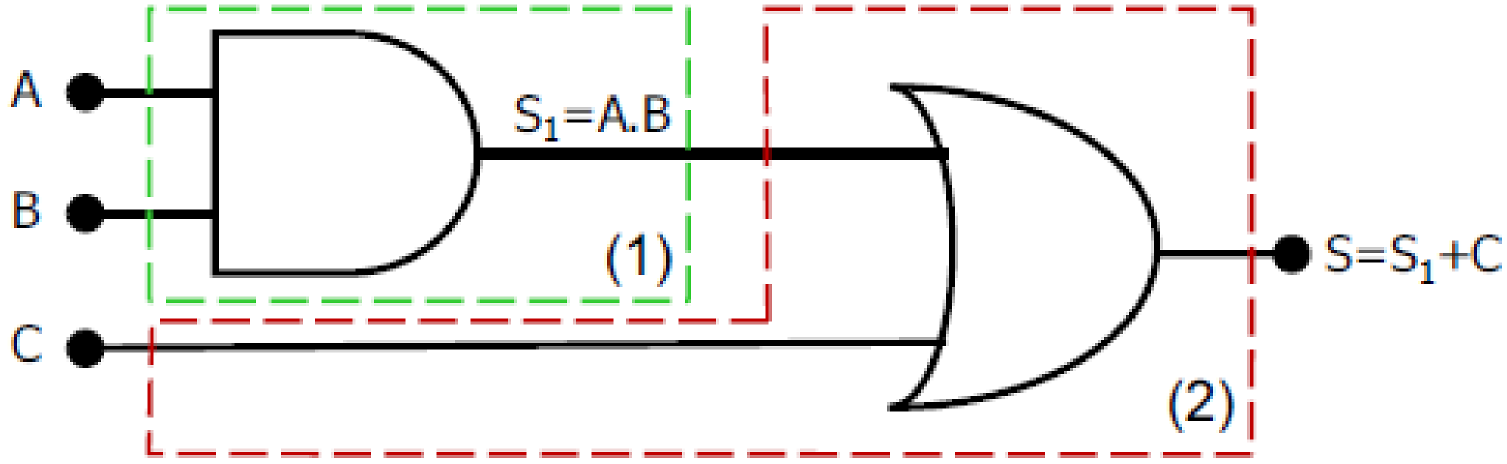
variável C, o que nos leva à:

  - S = S1 + C



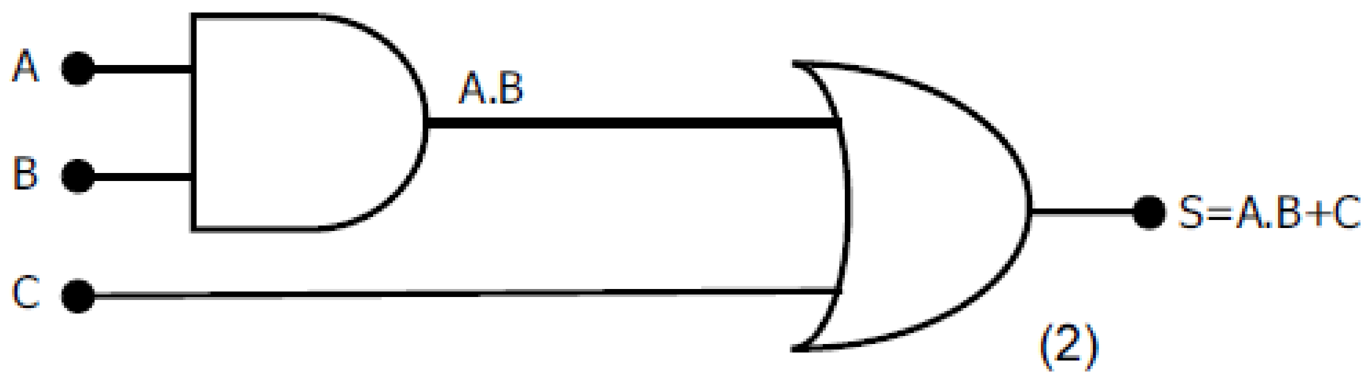
Expressões Booleanas Geradas por Circuitos Lógicos

- 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) S1 = A.B
  - (2) S = S1 + C
  - Obtém-se S = S1 + C = (A.B) + C



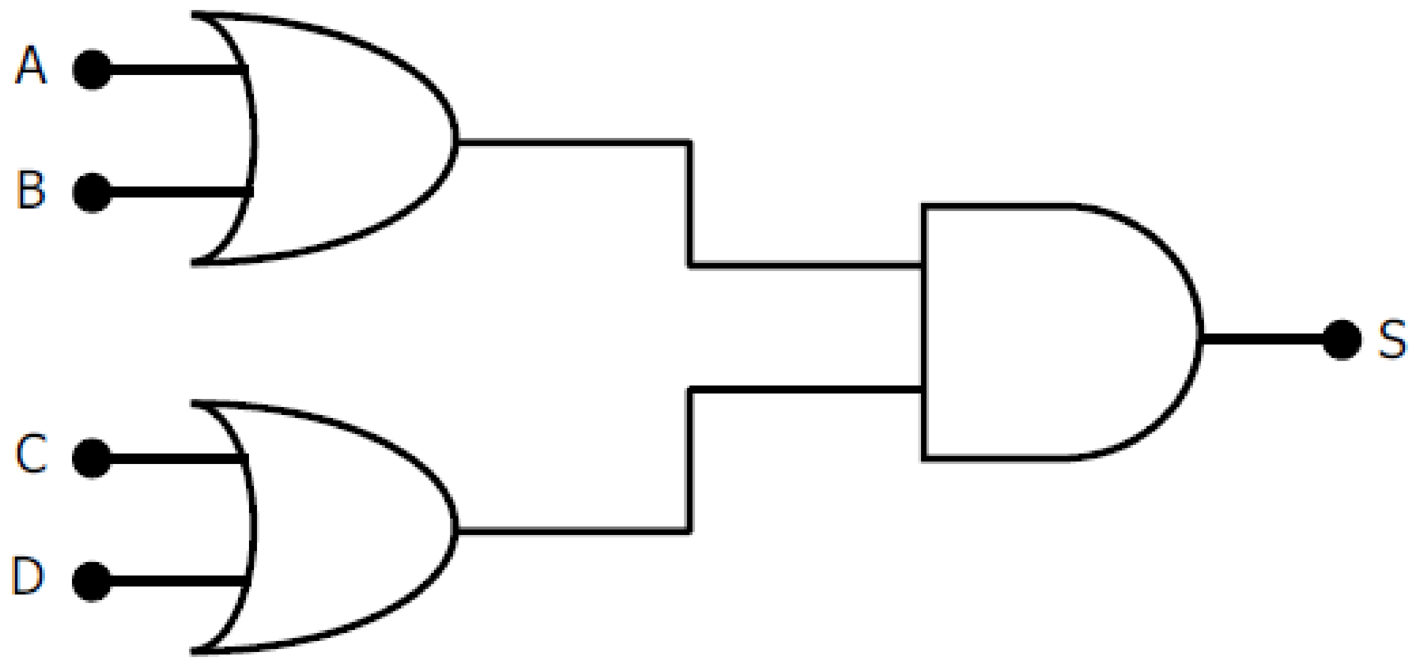
Expressões Booleanas Geradas por Circuitos Lógicos

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

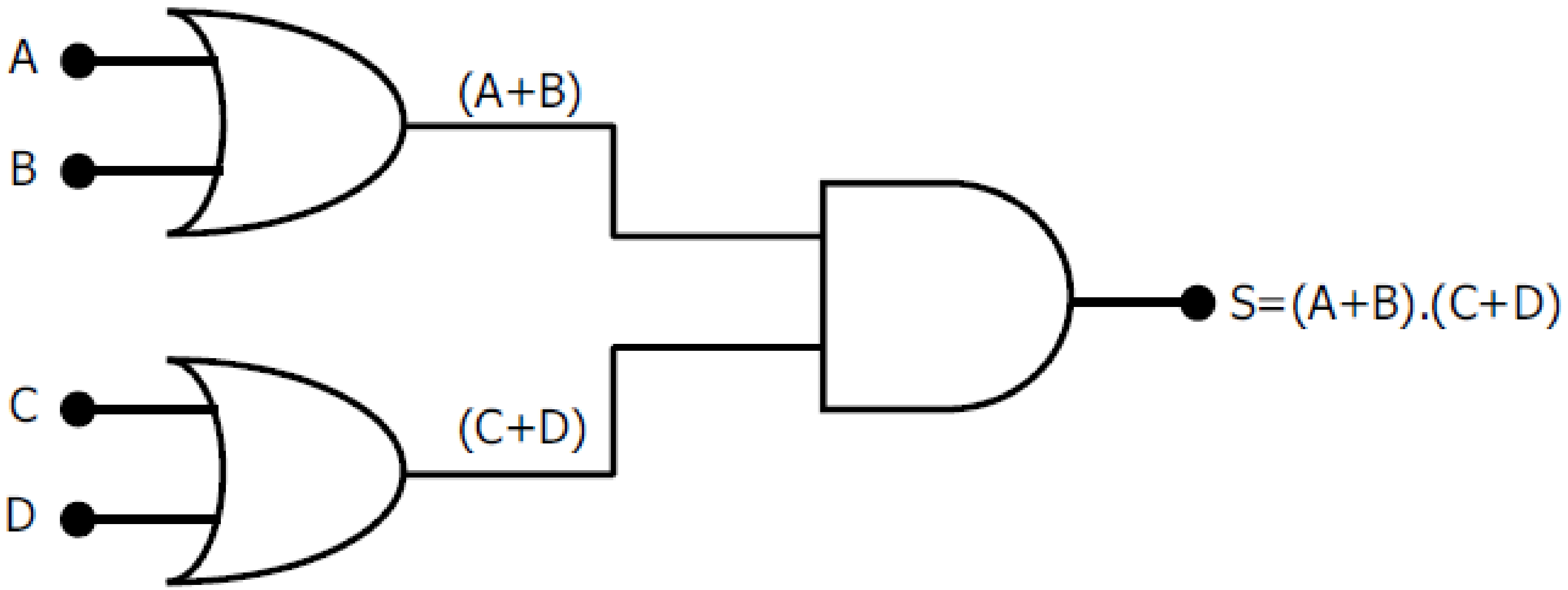


Exercício

- 1) Escreva a expressão booleana executada pelo circuito

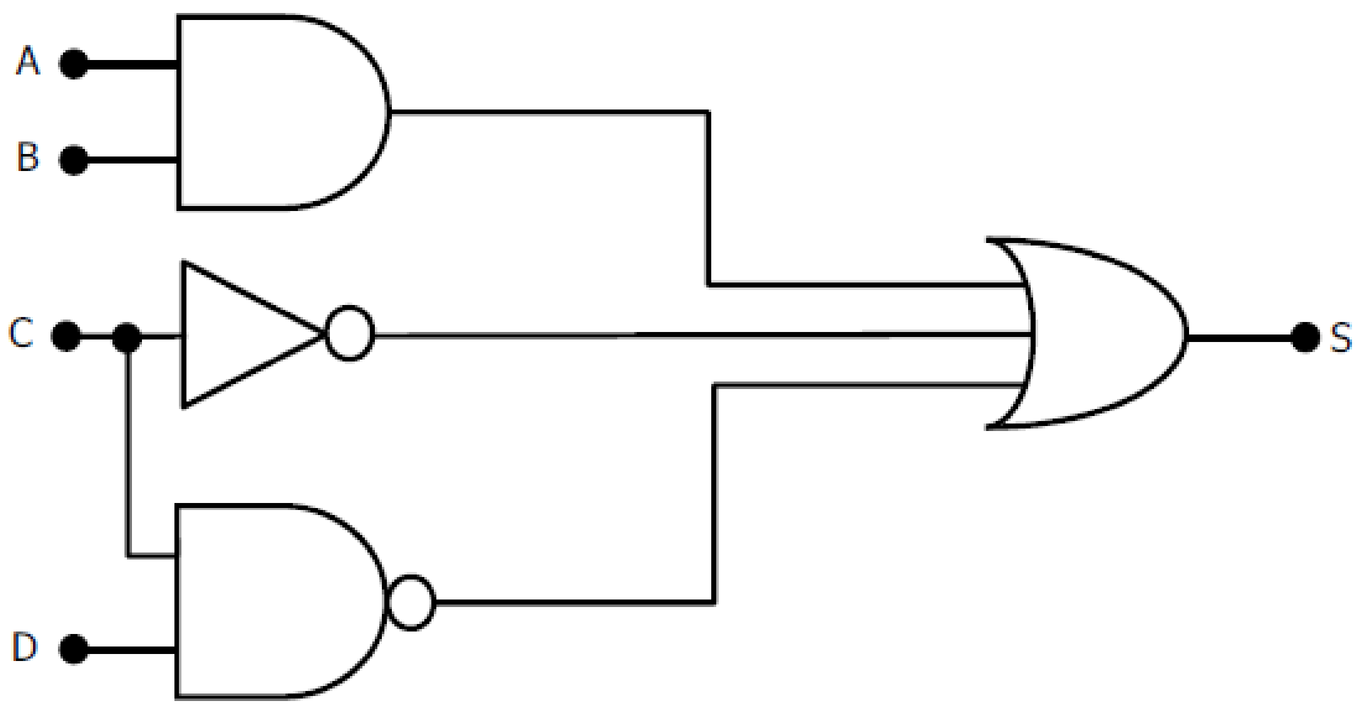


Solução Exercício 1

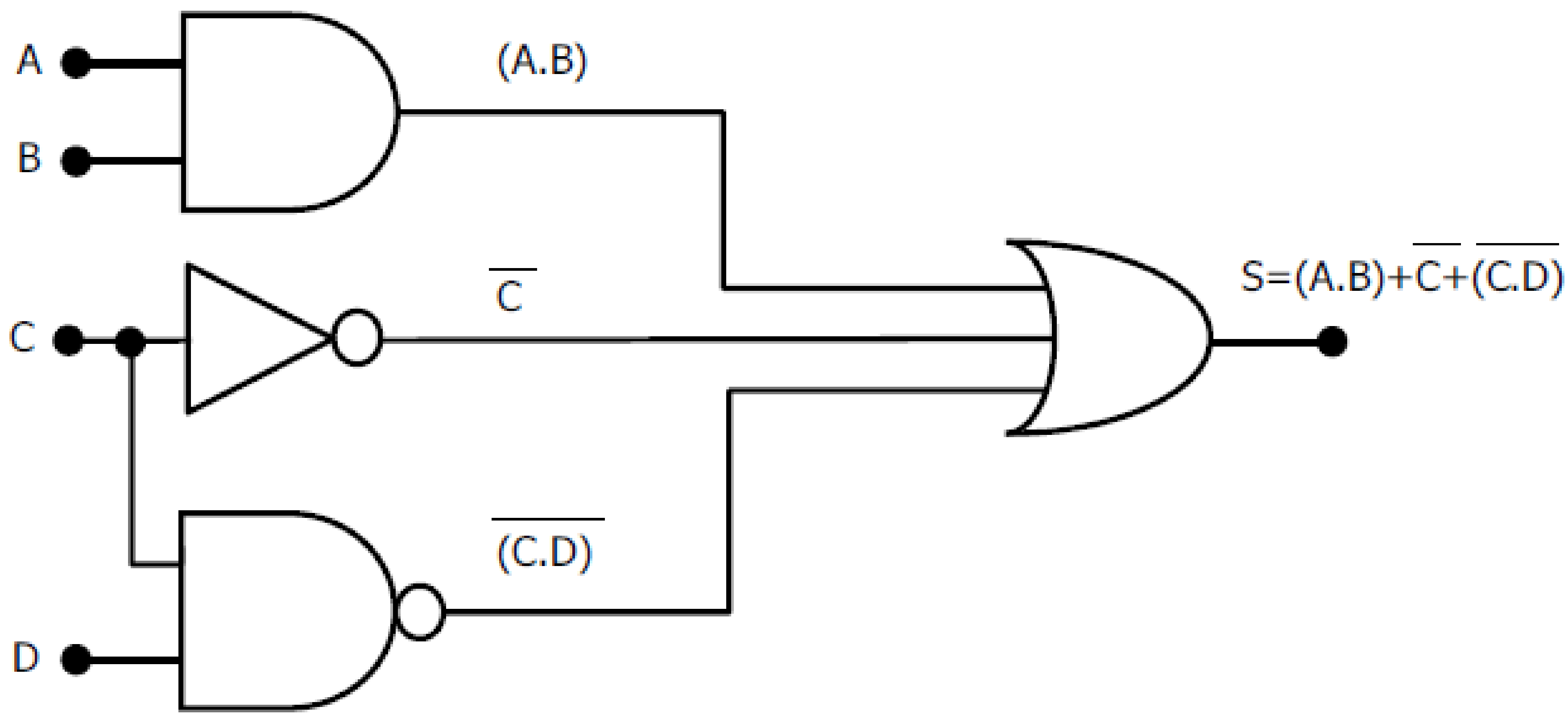


Exercício

- 2) Escreva a expressão booleana executada pelo circuito



Solução Exercício 2

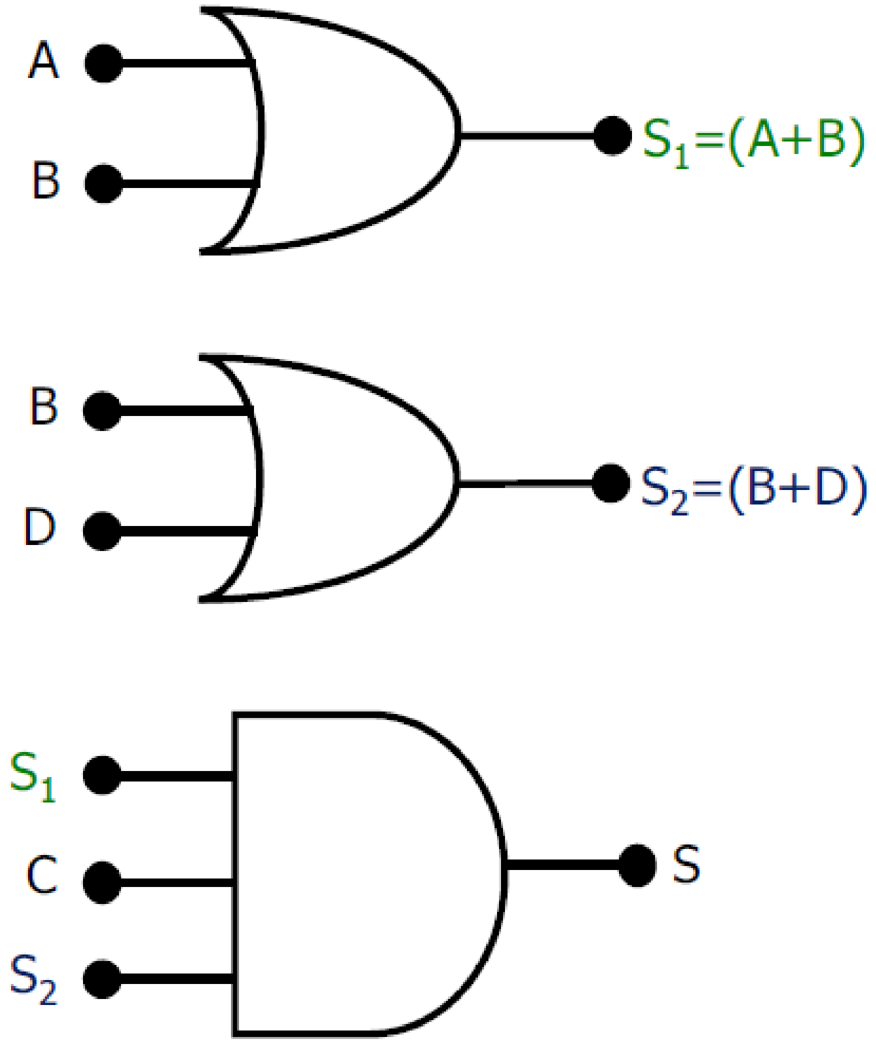


Circuitos Gerados por Expressões Booleanas

- Vimos como obter uma expressão característica a partir de um circuito
- Também é possível obter um circuito lógico, dada uma expressão booleana
- Nesse caso, como na aritmética elementar, parênteses têm maior prioridade, seguidos pela multiplicação (função E) e, por último, pela soma (função OU)

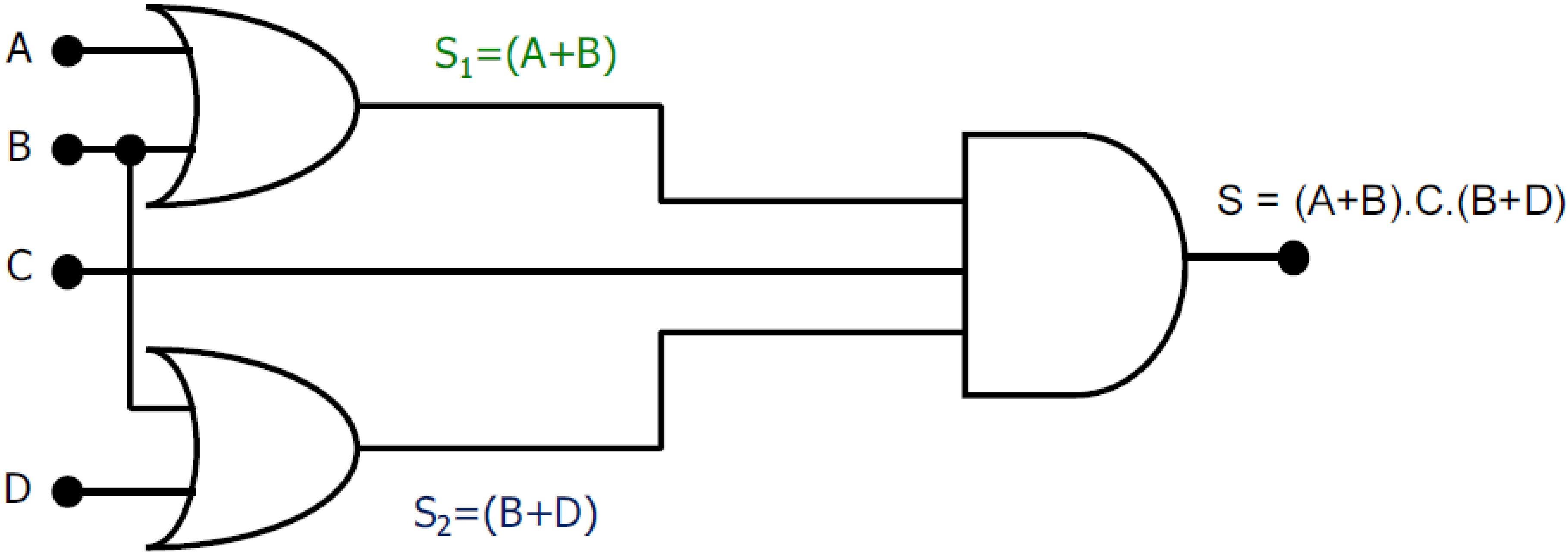
Circuitos Gerados por Expressões Booleanas

- Seja a expressão
  - S = (A+B).C.(B+D)
- Vamos separar as subfórmulas da expressão, ou seja:
  - S = (A+B) . C . (B+D)
- Dentro do primeiro parêntese temos a soma booleana S1=(A+B), portanto o circuito que executa esse parêntese será uma porta OU
- Dentro do segundo parêntese temos a soma booleana S2=(B+D). Novamente, o circuito que executa esse parêntese será uma porta OU
- Portanto, temos:
  - S = S1 . C . S2
- Agora temos uma multiplicação booleana e o circuito que a executa é uma porta E



Circuitos Gerados por Expressões Booleanas

- O circuito completo é:

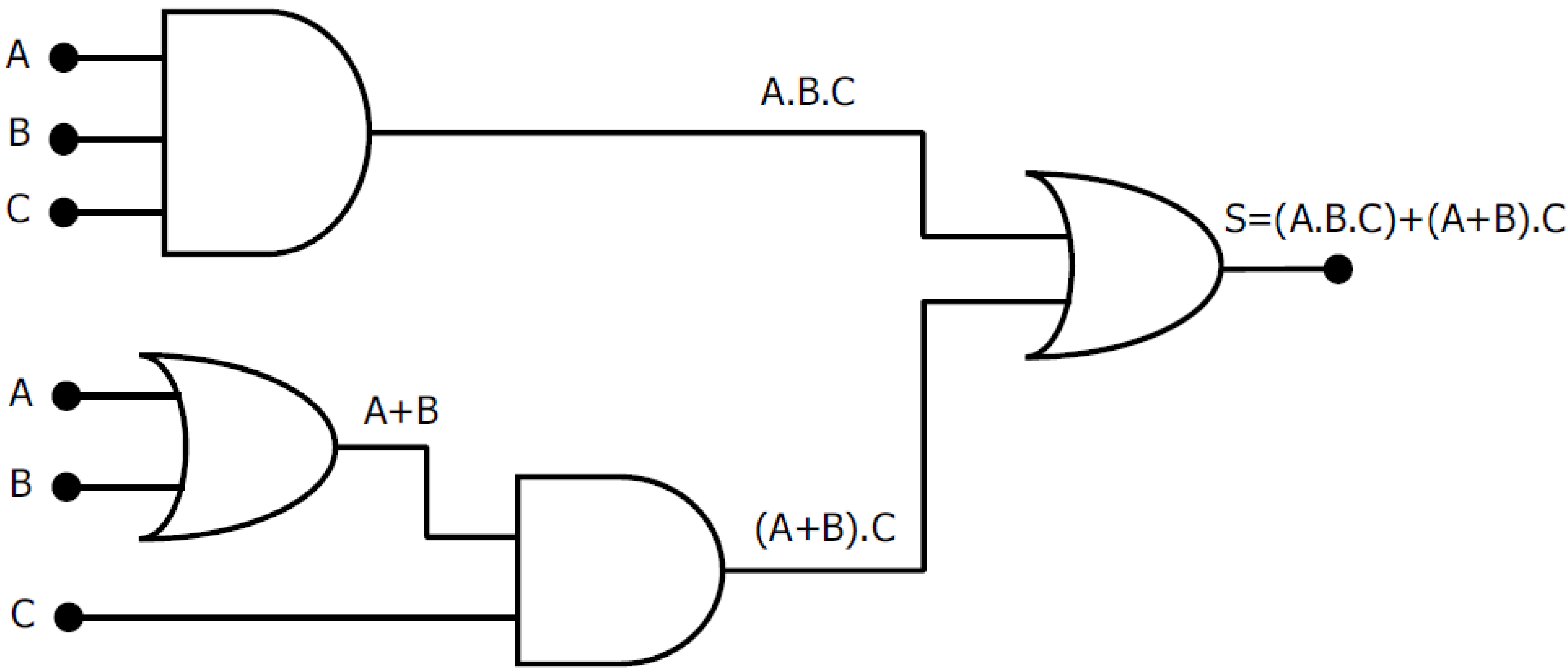


Exercício

- 3) Desenhe o circuito lógico que executa a seguinte expressão booleana
  - S = (A.B.C) + (A+B).C

Solução Exercício 3

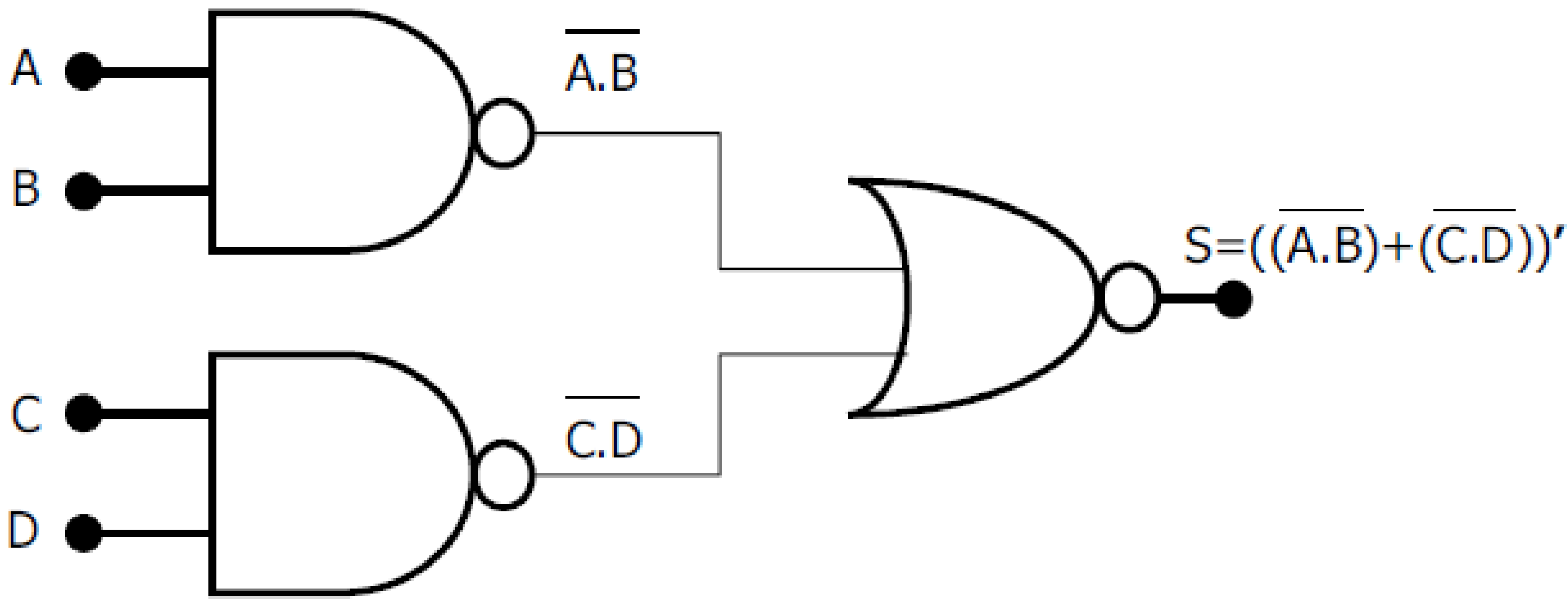
- É importante lembrar que as entradas que representam a mesma variável estão interligadas
- Contudo o desenho sem interligações facilita a interpretação do circuito



Exercício

- 4) Desenhe o circuito lógico cuja expressão característica é
  - $S = (\overline{A.B} + \overline{C.D})'$

Solução Exercício 4

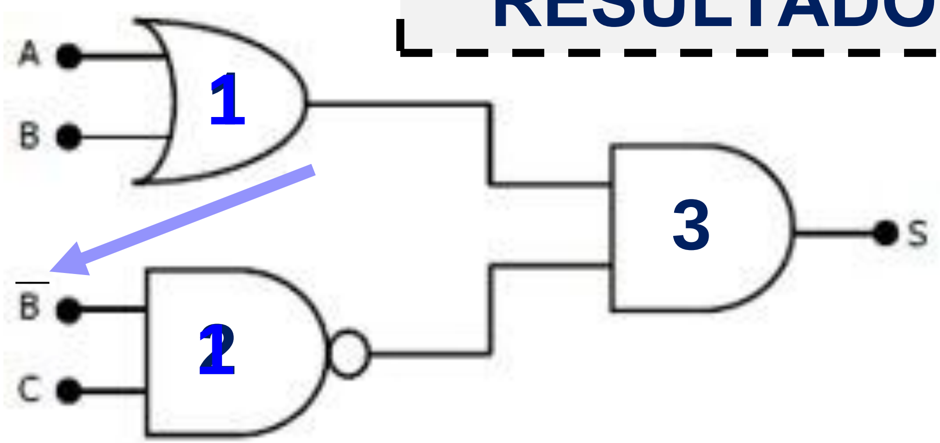


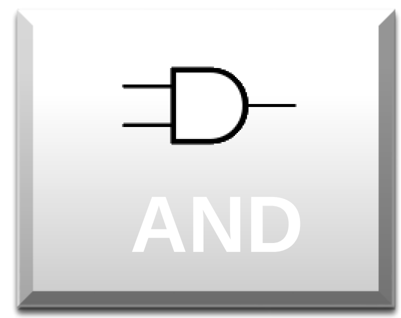
Exercício

4) Considerando os valores de A=0, B=1 e C=0.

Qual o resultado?

RESULTADO = 1



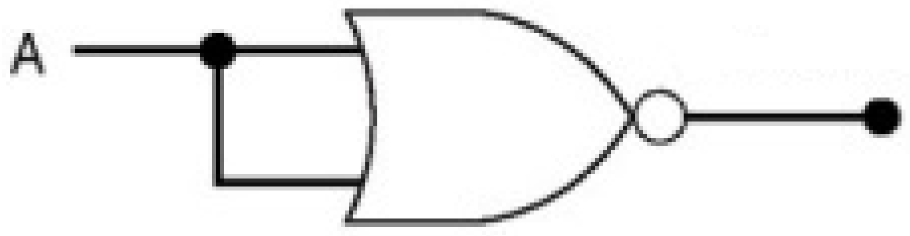


ENTRADA 1	ENTRADA 2	SAÍDA
0	0	0 / 1
1	0	1 / 0
0	1	1 / 0
1	1	1 / 0

Exercícios

5) Considerando os valores de A=0, B=1 e C=0.

Qual o resultado?



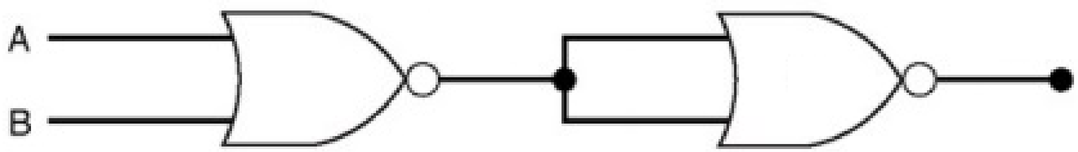
$$\overline{(A + A)}$$

$$= \overline{(0 + 0)}$$

$$= 1$$

Exercícios

6) Considerando os valores de A=0, B=1 e C=0.
Qual o resultado?



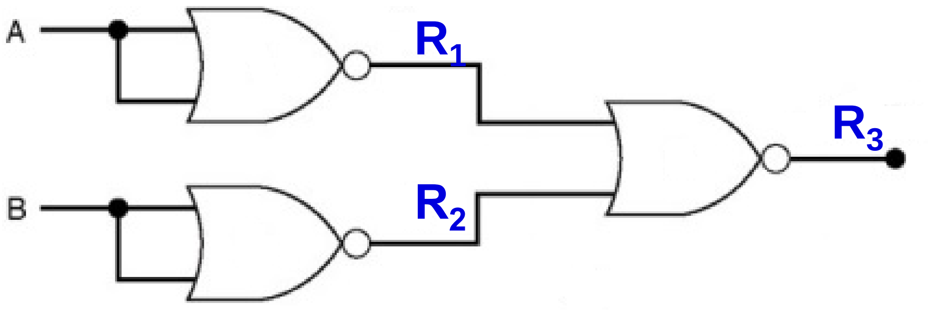
$$R = \overline{(A + B)}$$
$$= \overline{(0 + 1)}$$
$$= \overline{1}$$
$$= 0$$

$$\overline{(R + R)}$$
$$= \overline{(0 + 0)}$$
$$= \overline{0}$$
$$= 1$$

Exercícios

7) Considerando os valores de A=1 e B=1.

Qual o resultado?



$R_1 = (\overline{A + A})$
$= (\overline{1 + 1})$
$= \overline{1} = 0$

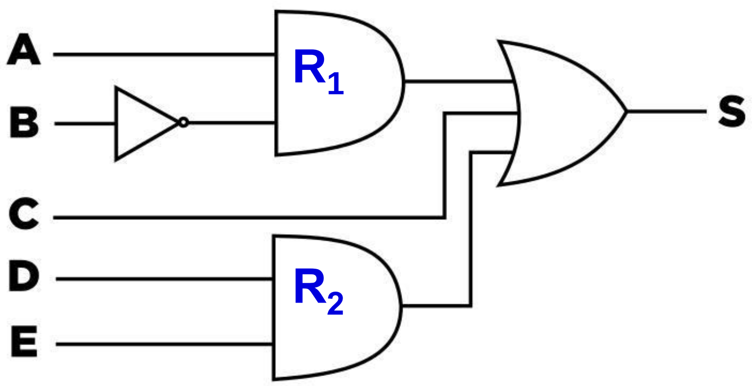
$R_2 = (\overline{B + B})$
$= (\overline{1 + 1})$
$= \overline{1} = 0$

$R_3 = (\overline{R_1 + R_2})$
$= (\overline{0 + 0})$
$= \overline{0} = 1$

Exercícios

8) Considerando os valores de A=0, B=1, C=0

D=1 e E=1. Qual o resultado?



$$R_1 = (\ A * \overline{B}\ )$$
$$= (\ 0 + \overline{1}\ )$$
$$= 0$$

$$R_2 = (\ D * E\ )$$
$$= (\ 1 * 1\ )$$
$$= 1$$

$$S = (R_1 + C + R_2)$$
$$= (\ 0 + 0 + 1\ )$$
$$= 1$$

Exercícios

9) Considerando os valores de A=0, B=1 e C=0.

Qual o resultado?

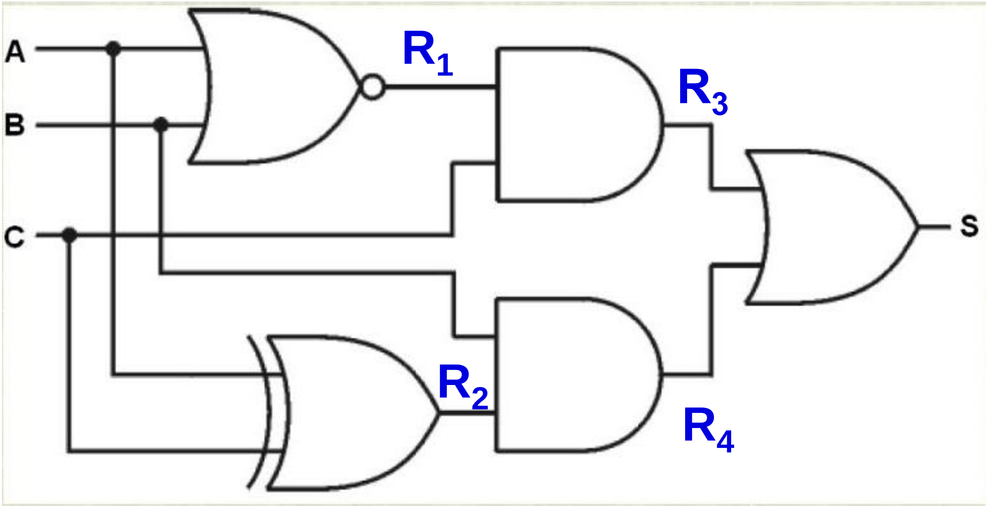
$R_1 = \overline{(A + B)} = \overline{(0 + 1)} = 0$

$R_2 = (A \oplus C) = (0 \oplus 0) = 0$

$R_3 = (R_1 * C) = (0 * 0) = 0$

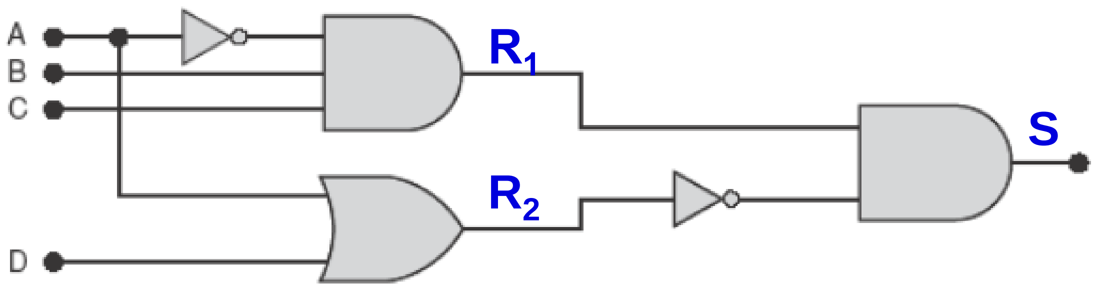
$R_4 = (B * R_2) = (1 * 0) = 0$

$S = (R_3 + R_4) = (0 + 0) = 0$



Exercícios

10) Escreva as expressões booleanas e resolva.

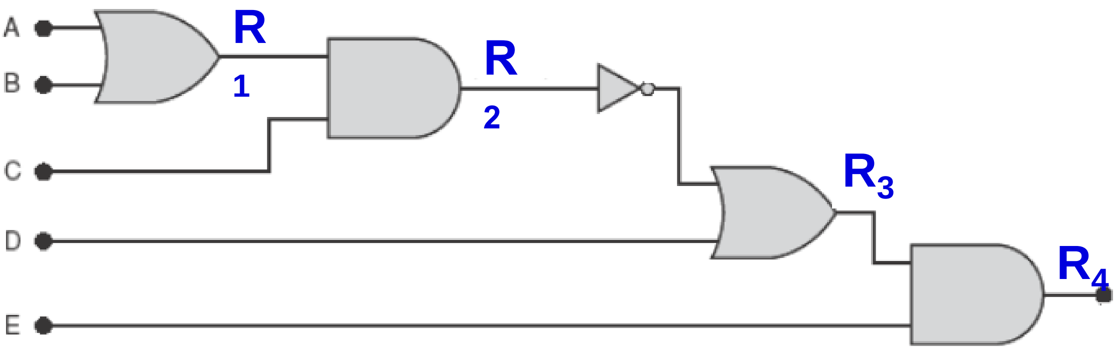


A = 0
B = 1
C = 1
D = 0

$$R_1 = (\overline{A} * B * C) = (\overline{0} * 1 * 1) = 1$$
$$R_2 = (A + D) = (0 + 0) = 0$$
$$S = (R_1 * \overline{R_2}) = (1 * \overline{0}) = 1$$

Exercícios

11) Escreva as expressões booleanas e resolva.



A = 0
B = 1
C = 1
D = 0
E = 1

$$R_1 = (A + B) = (0 + 1) = 1$$

$$R_2 = (R_1 * C) = (1 * 1) = 1$$

$$R_3 = (\overline{R_2} + D) = (\overline{1} + 0) = 0$$

$$R_4 = (R_3 * E) = (0 * 1) = 0$$

Conclusão

- Conhecemos as porta lógicas E, OU e NÃO e suas combinações em circuitos.
- Elas tem a mesma função de operadores E, OU e Não.
- Podem ser combinados em circuitos e gerar expressões booleanas.

Referências

- WEBER, Raul Fernando. Fundamentos de arquitetura de computadores. 4. ed. Porto Alegre: Bookman, 2012. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788540701434
- STALLINGS, William. Arquitetura e organização de computadores. 8.ed. São Paulo: Pearson, 2010. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/459/epub/0
- HOGLUND, Greg. Como quebrar códigos: a arte de explorar (e proteger) software. São Paulo: Pearson, 2006. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/179934/epub/0

Fim