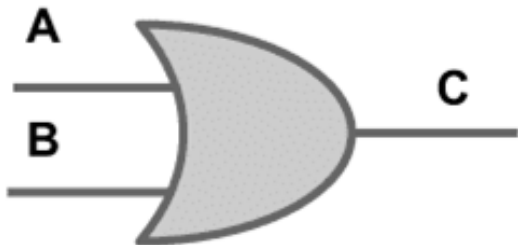



Sistemas computacionais e segurança

Portas Lógicas

Prof. Théo Augustus Luz
Prof. Richard





9ª Maratona CiberEducação Cisco Brasil

- Segurança de Endpoint

(em português) – 27 hs

- https://skillsforall.com/course/endpoint-security?courseLang=pt-BR&instance_id=68c9e5fc-9fc6-4d60-9624-2044f94e3041
- Garanta já a sua vaga nesta iniciativa do Cisco Networking Academy e do Brasil Digital e Inclusivo.
 - Os participantes que concluírem o curso “Segurança de Endpoint”, poderão concorrer a **1500 bolsas** de estudo profissionalizante gratuitas!
- Após a conclusão o aluno deve submeter o certificado de conclusão do curso, através do forms (<https://forms.gle/kDrso7yAhesM51aJA>), para **garantir as 40 hs de extensão**.
- **Obs.: O curso inicia dia 02/09 e vai até 22/09.**

Objetivos

- Apresentar porta lógicas e sua função na composição de circuitos digitais;
- Apresentar as funções **E**, **OU** e **NÃO** 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

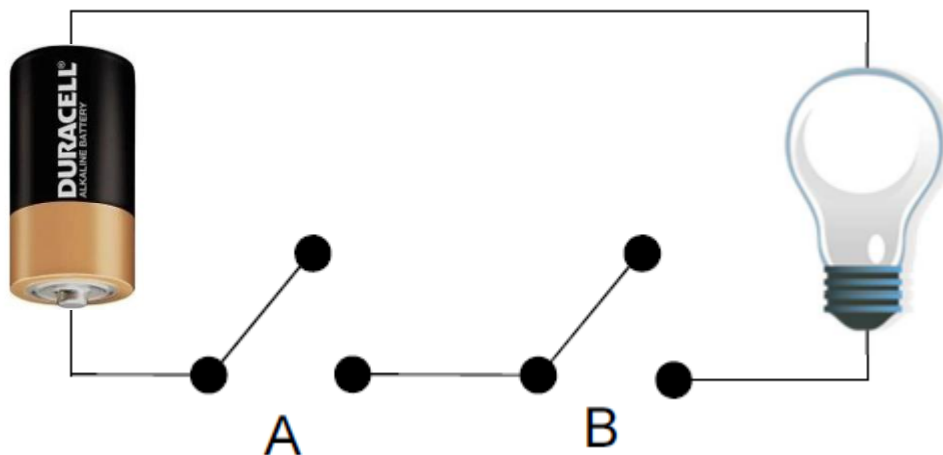
- Apesar de ser possível imaginar inúmeras portas lógicas diferentes que manipulam bits de entrada para produzir uma saída específica, existem 3 **portas lógicas** elementares que, quando combinadas, são responsáveis por criar praticamente qualquer lógica digital.
 - Porta **NOT**
 - Porta **OR**
 -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

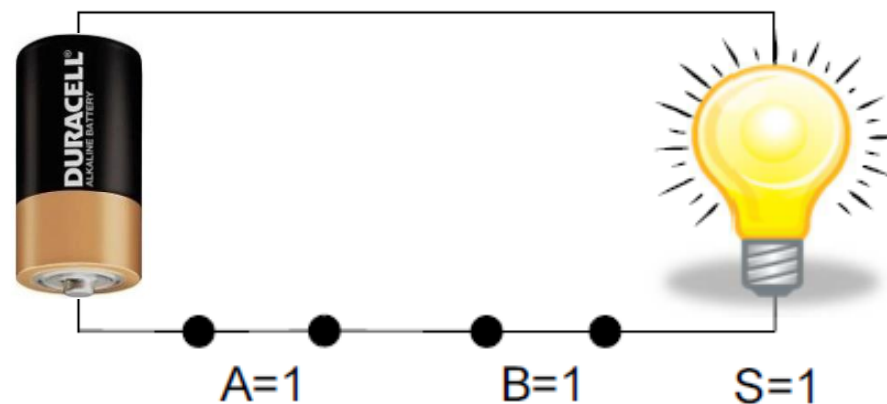
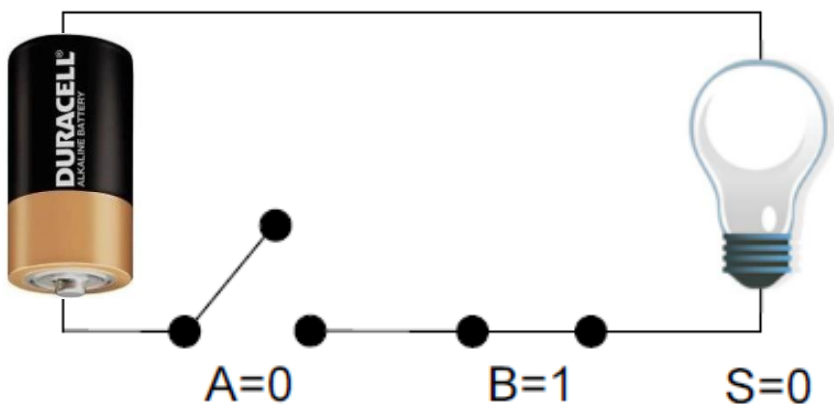
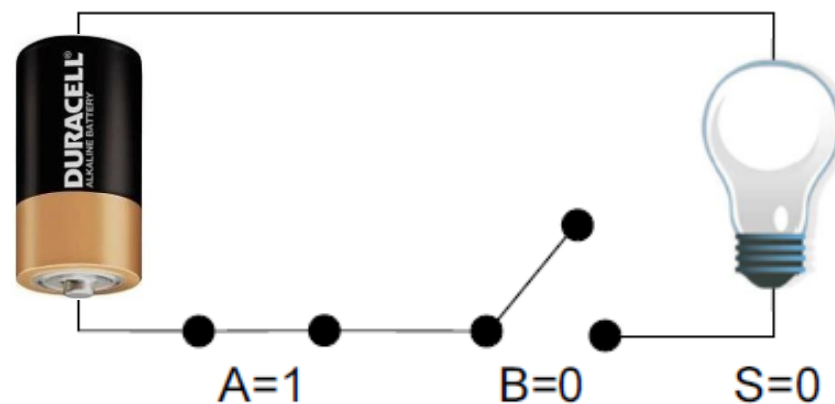
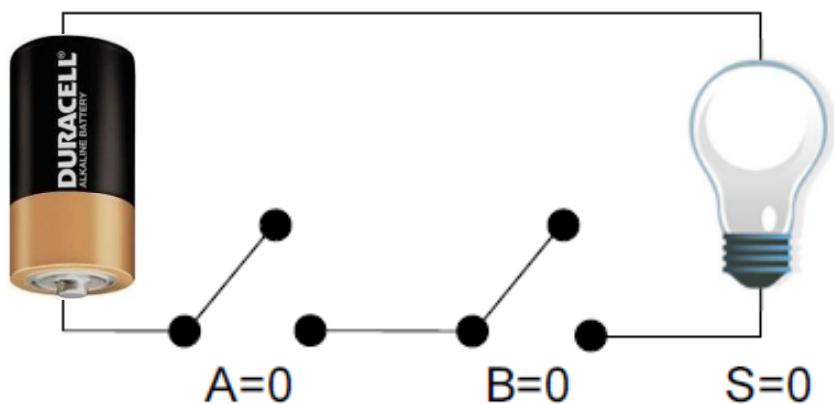
Função E (AND)

- 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



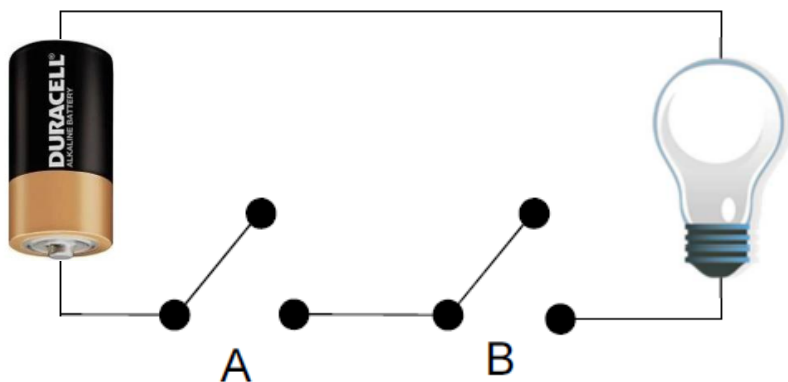
Função E (AND)

- Para representar a expressão
 - $S = A \text{ e } B$
- Adotaremos a representação
 - $S = A.B$, onde se lê $S = A \text{ e } B$
- Porém, existem **notações alternativas**
 - $S = A \& B$
 - $S = A, B$
 - $S = A \wedge B$

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

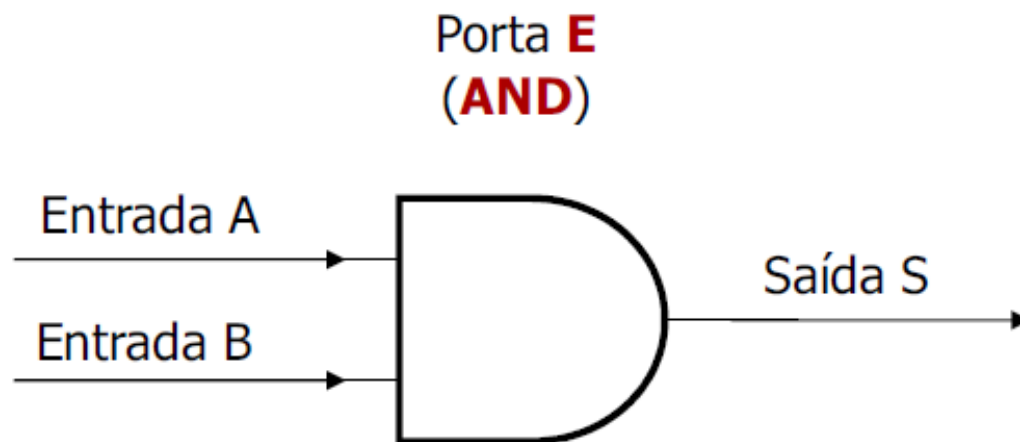
Tabela Verdade da Função E(AND)



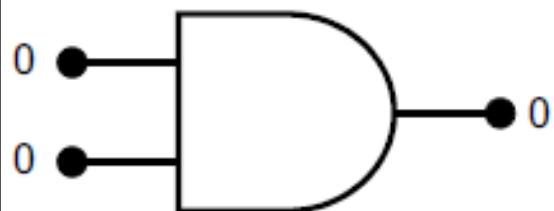
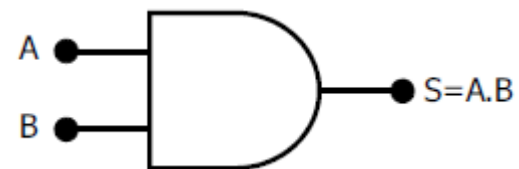
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Porta Lógica E(AND)

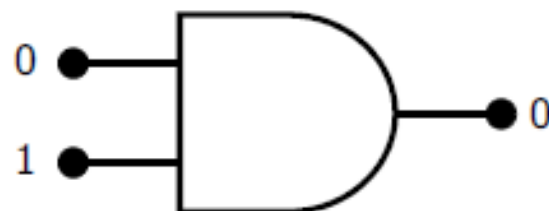
- A porta E é um circuito que executa a função E
- A porta E executa a tabela verdade da função E
 - Portanto, a saída será 1 somente se ambas as entradas forem iguais a 1; nos demais casos, a saída será 0
- Representação



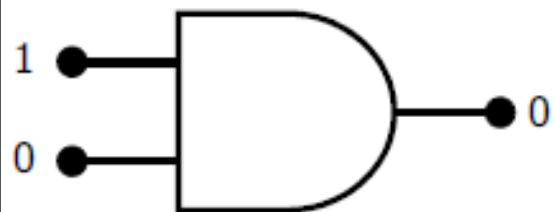
Porta Lógica E(AND)



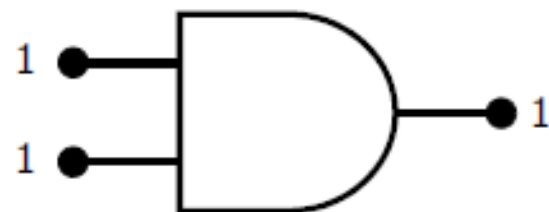
A	B	$S=A.B$
0	0	0
0	1	0
1	0	0
1	1	1



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



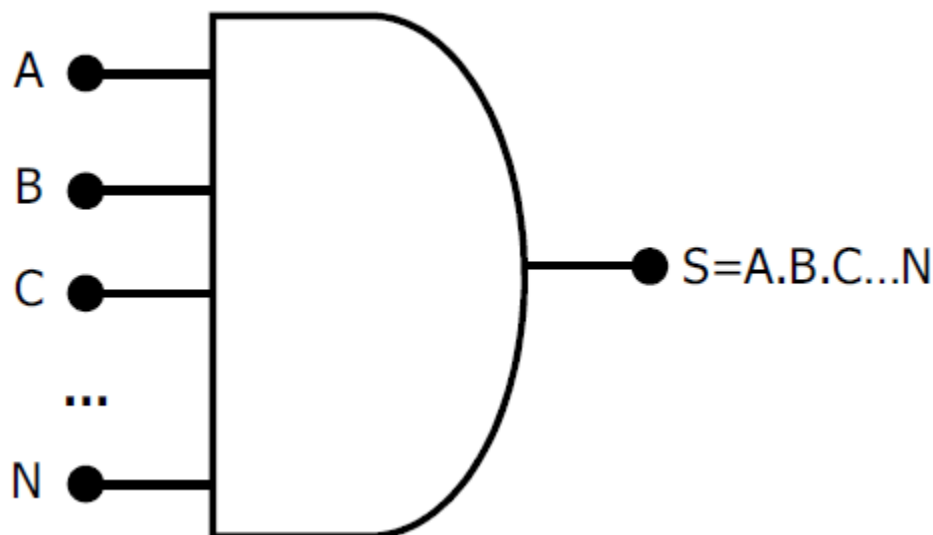
A	B	$S=A.B$
0	0	0
0	1	0
1	0	0
1	1	1



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

Porta Lógica E(AND)

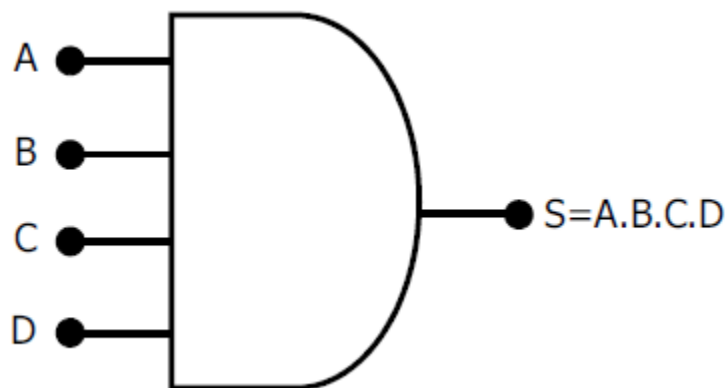
- É possível estender o conceito de uma porta E para um número qualquer de variáveis de entrada
- Nesse caso, temos uma porta **E** com N entradas e somente **uma saída**
- A saída será 1 se e somente se as N entradas forem iguais a 1; nos demais casos, a saída será 0



Porta Lógica E(AND)

■ Por exemplo,

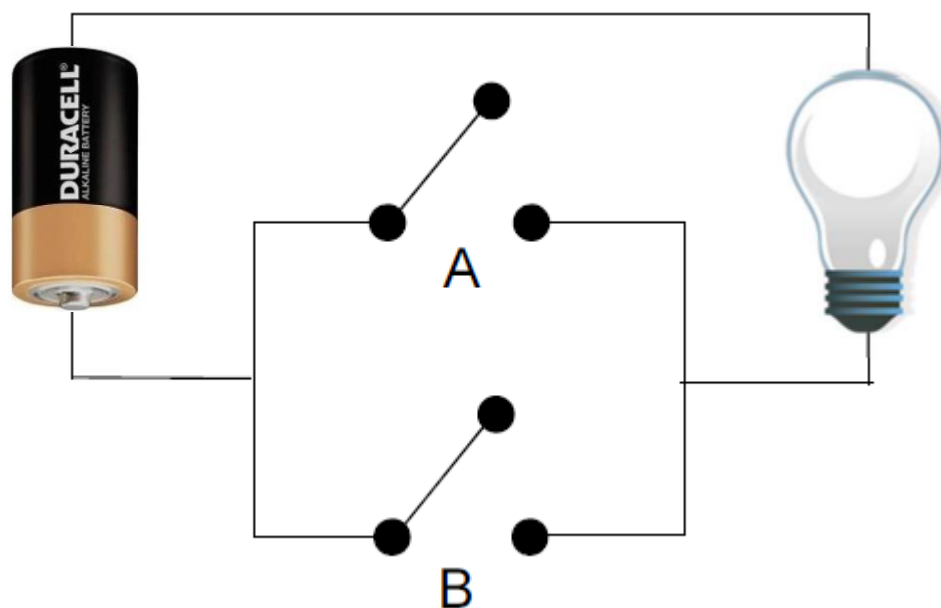
□ $S = A.B.C.D$



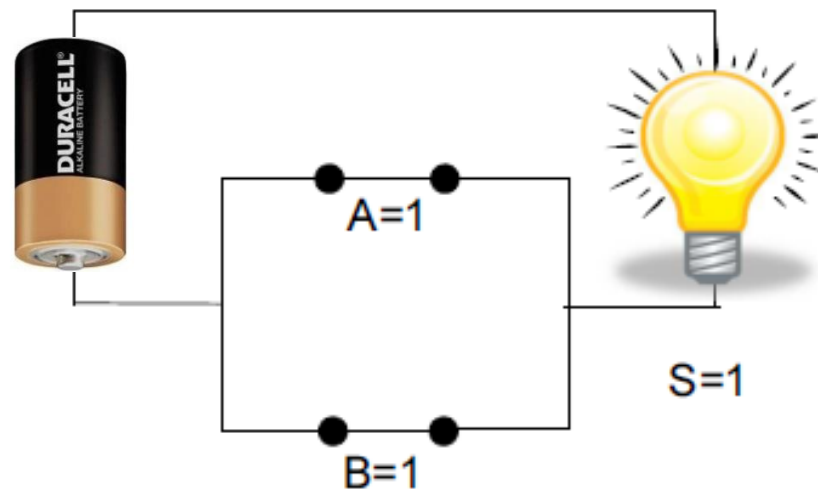
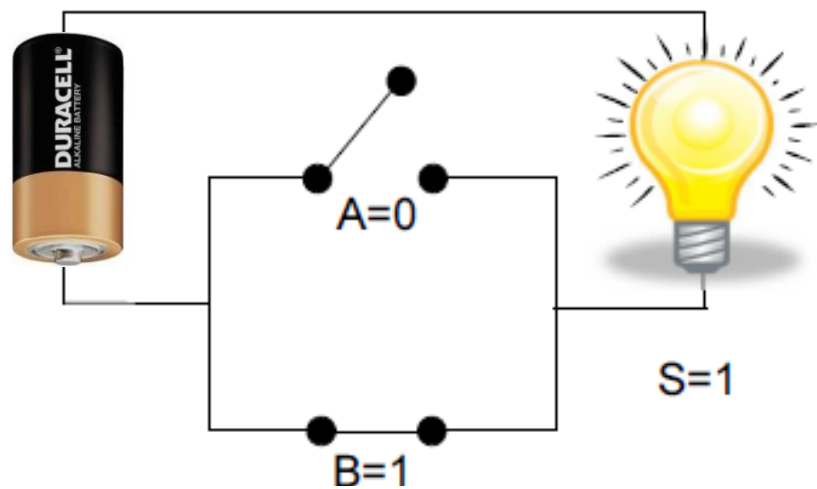
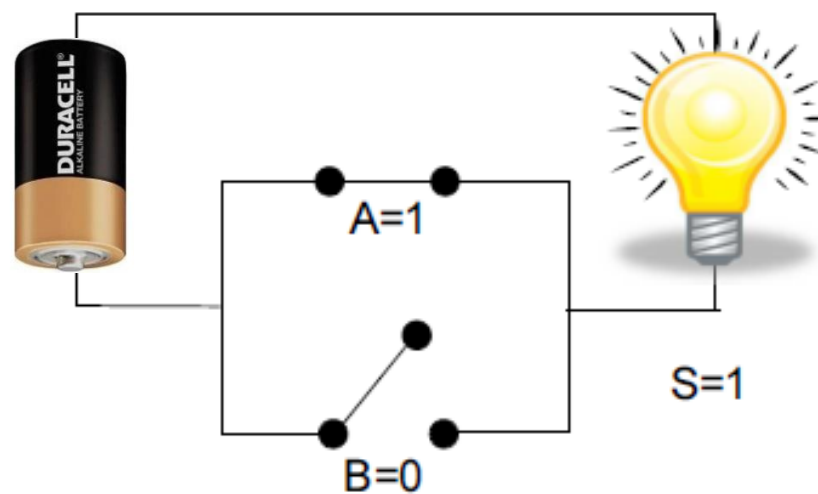
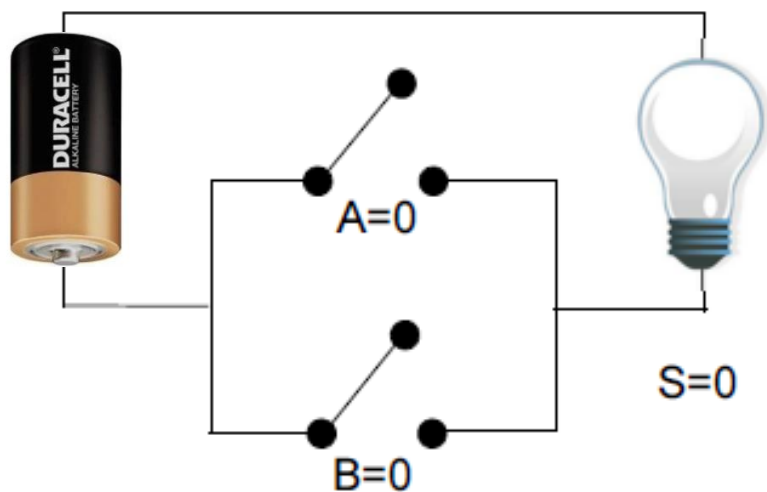
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

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)

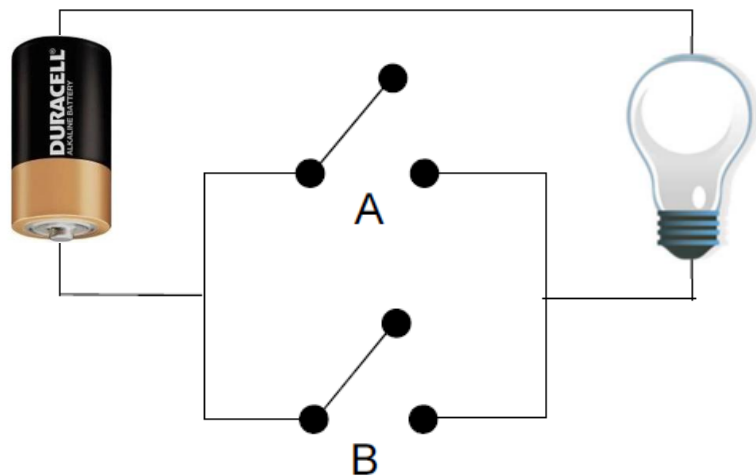


Função OU (OR)

- Para representar a expressão
 - $S = A \text{ ou } B$
- Adotaremos a representação
 - $S = A+B$, 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)

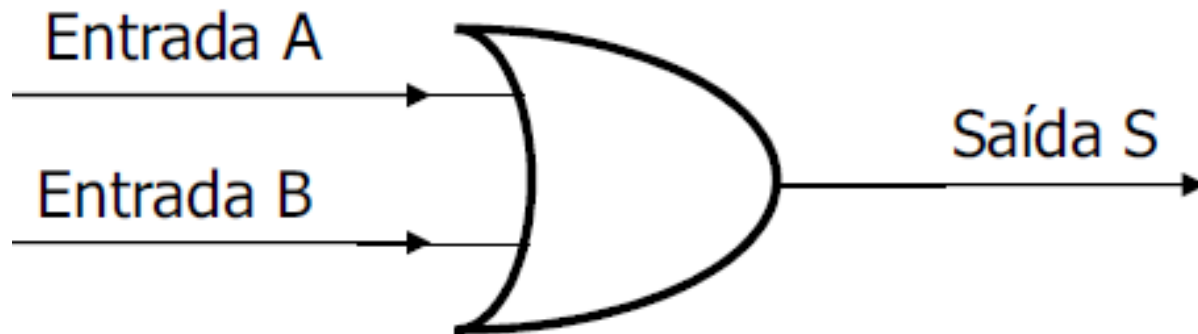
- Observe que, no sistema de numeração binário, a soma $1+1=10$
- Na álgebra booleana, $1+1=1$, já que somente dois valores são permitidos (0 e 1)



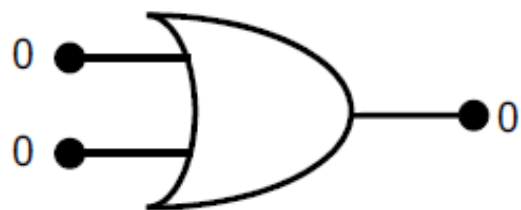
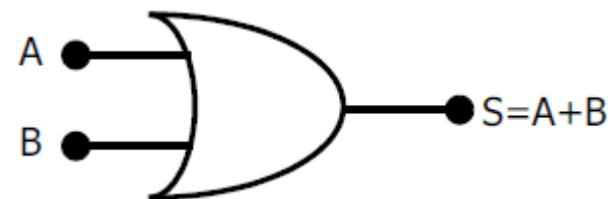
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Porta Lógica OU (OR)

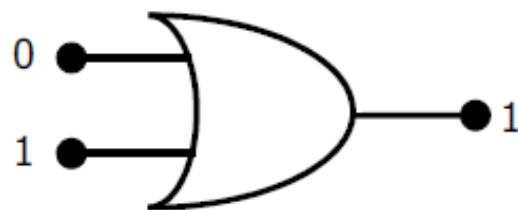
- A porta **OU** é um circuito que executa a função **OU**
- A porta **OU** executa a tabela verdade da função **OU**
 - Portanto, a saída será 0 somente se ambas as entradas forem iguais a 0; nos demais casos, a saída será 1
- Representação



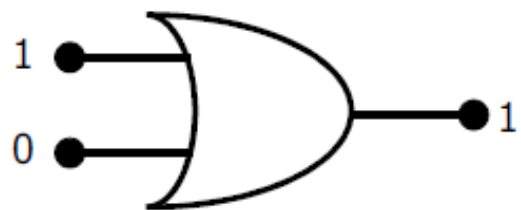
Porta Lógica OU (OR)



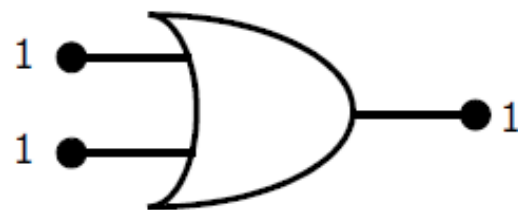
A	B	S=A+B
0	0	0
0	1	1
1	0	1
1	1	1



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



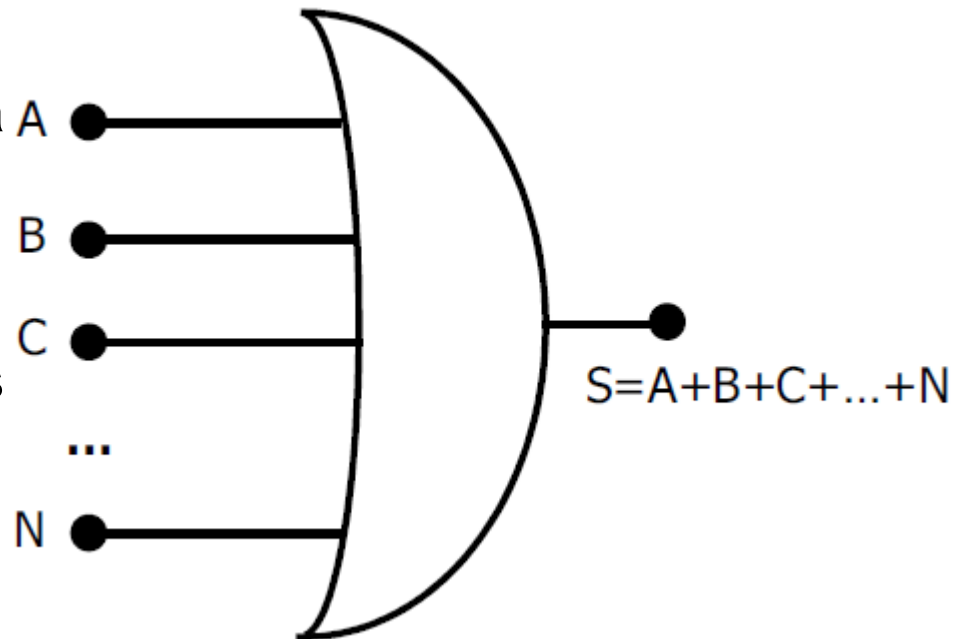
A	B	S=A+B
0	0	0
0	1	1
1	0	1
1	1	1



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

Porta Lógica OU (OR)

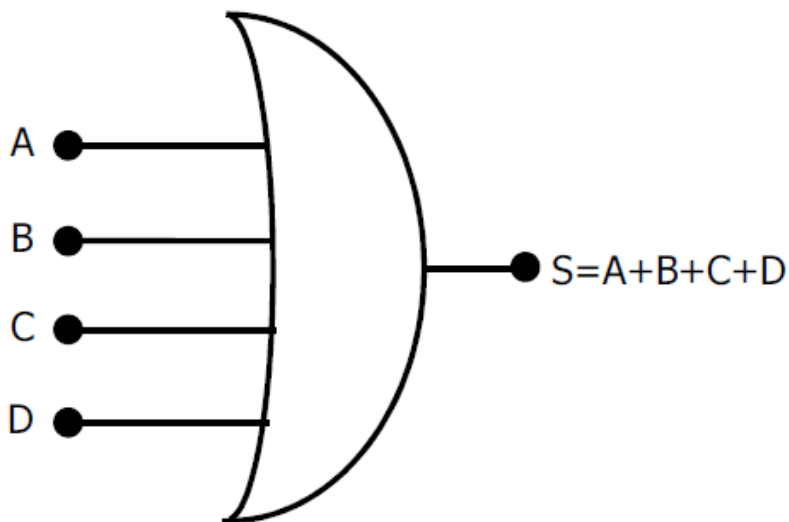
- É possível estender o conceito de uma porta **OU** para um número qualquer de variáveis de entrada
- Nesse caso, temos uma porta **A OU** com N entradas e somente **uma saída**
- A saída será 0 se e somente se as N entradas forem iguais a 0; nos demais casos, a saída será 1



Porta Lógica OU (OR)

■ Por exemplo,

□ $S=A+B+C+D$



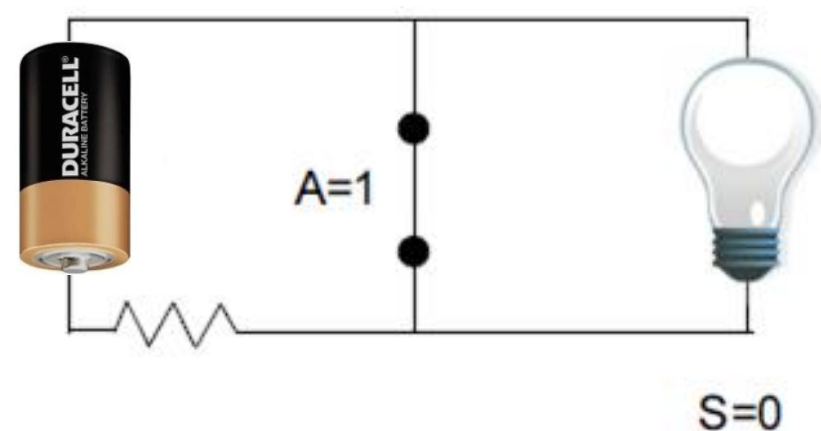
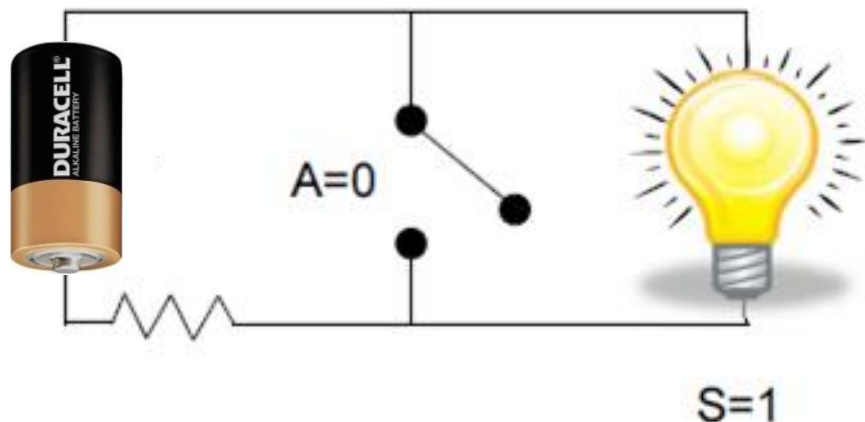
A	B	C	D	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

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

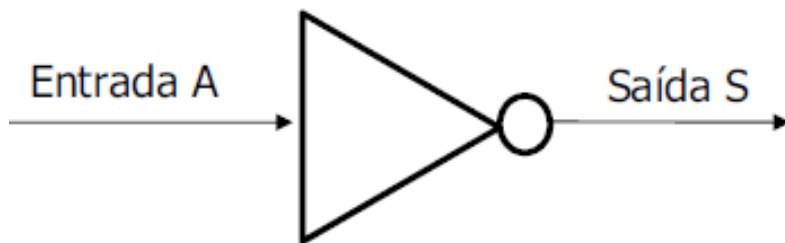
- Para representar a expressão
 - $S = \text{não } A$
- Adotaremos a representação $S = \bar{A}$, onde se lê $S = \text{não } A$
- Notações alternativas
 - $S = A'$
 - $S = \neg A$
 - $S = \tilde{A}$

- Tabela verdade da função NÃO (NOT)

A	\bar{A}
0	1
1	0

Porta Lógica NÃO(NOT)

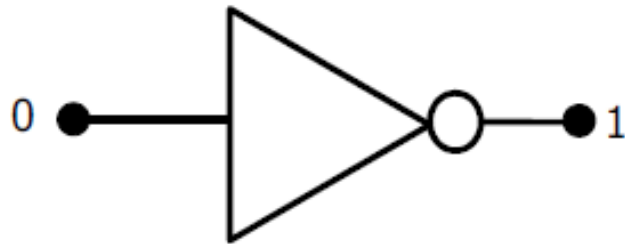
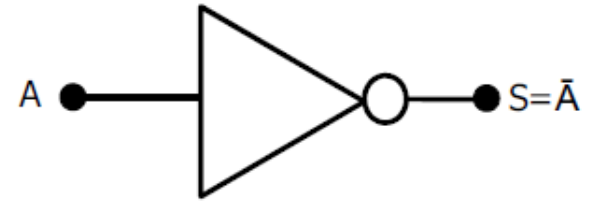
- A porta lógica NÃO, ou inversor, é o circuito que executa a função NÃO
- O inversor executa a tabela verdade da função NÃO
 - Se a entrada for 0, a saída será 1; se a entrada for 1, a saída será 0
- Representação



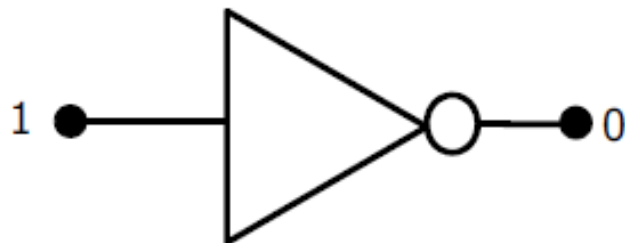
Alternativamente,



Porta Lógica NÃO(NOT)



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

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

- $= (A.B)'$

- $= \neg(A.B)$

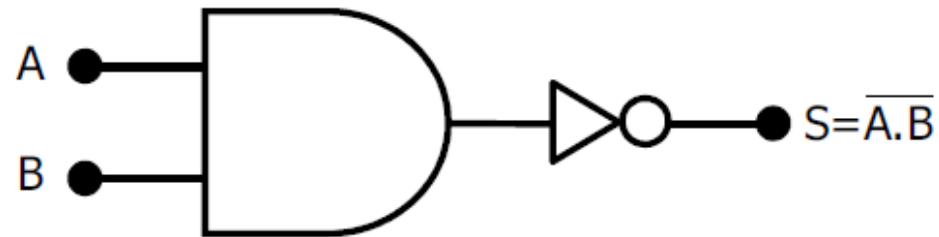
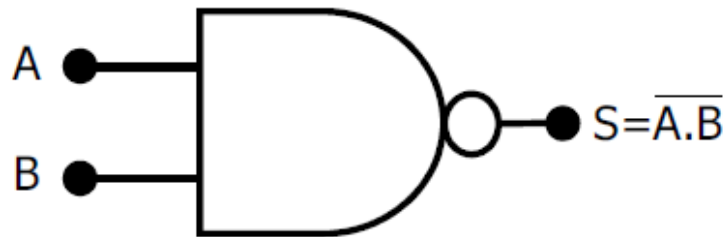
- Tabela Verdade

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

Porta NÃO E(NAND)

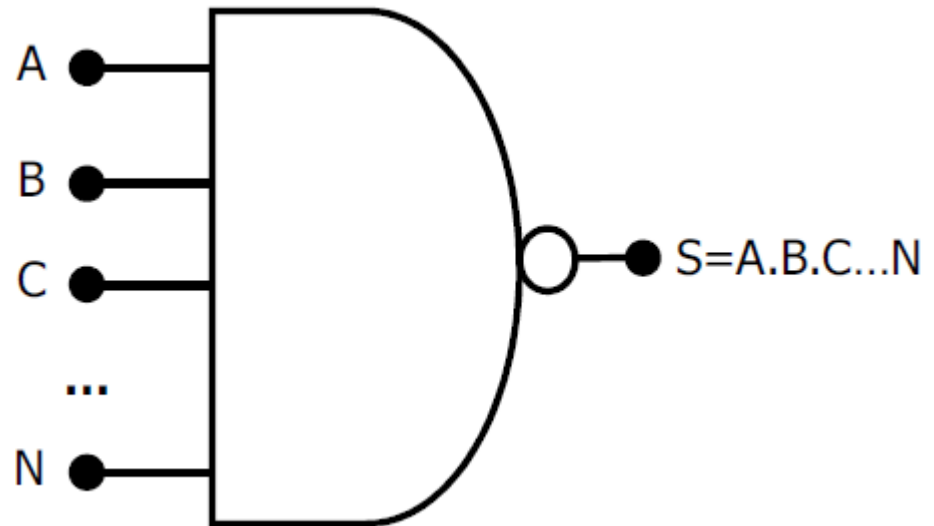
- A porta **NÃO E** (NE) é o bloco lógico que executa a função **NÃO E**, ou seja, sua tabela verdade
- Representação:

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

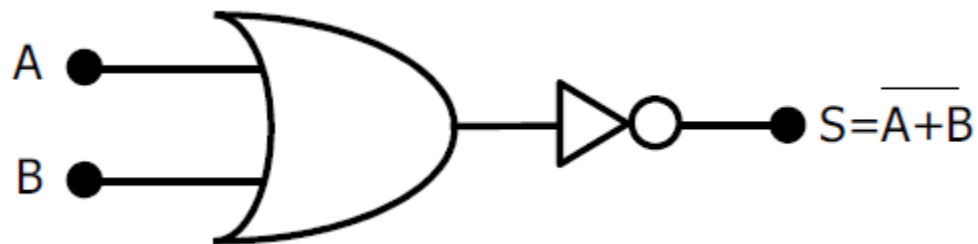
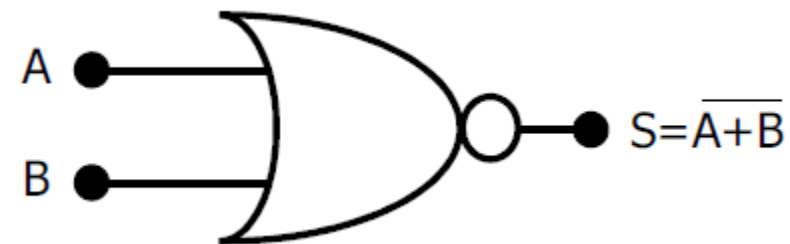
- $S = \overline{(A+B)} = \overline{A+B}$
- $= (A+B)'$
- $= \neg(A+B)$

- Tabela Verdade

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

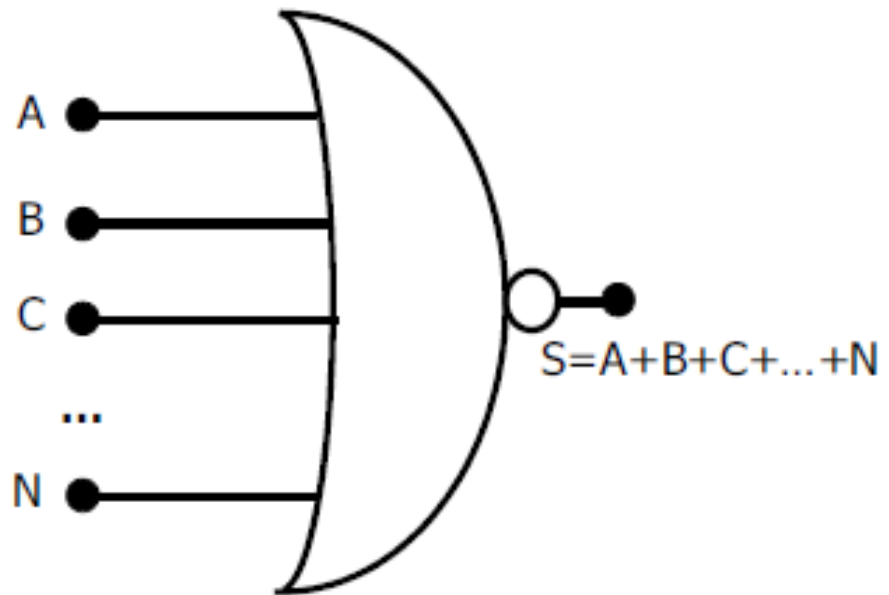
Porta NÃO OU(NOR)

- 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)

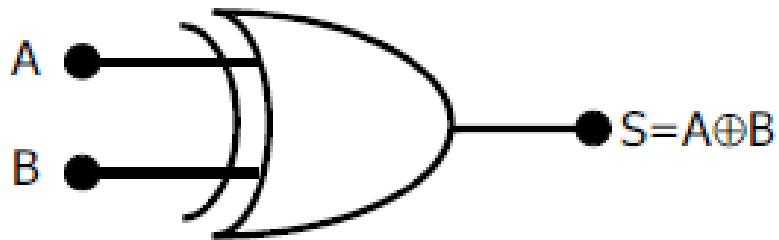
- A função OU Exclusivo
 - fornece 1 na saída quando as entradas forem diferentes entre si e 0 caso contrário
- $S = A + B$
- $= \bar{A}.B + A.\bar{B}$

■ Tabela Verdade

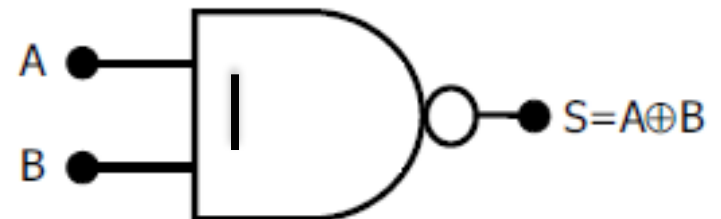
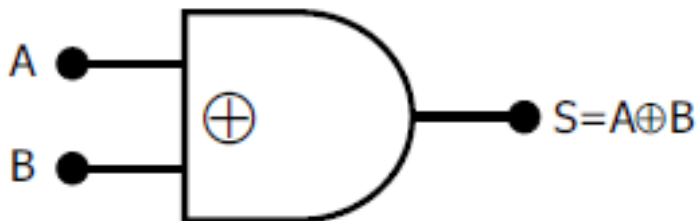
A	B	$S = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Porta OU-Exclusivo(XOR) como bloco básico

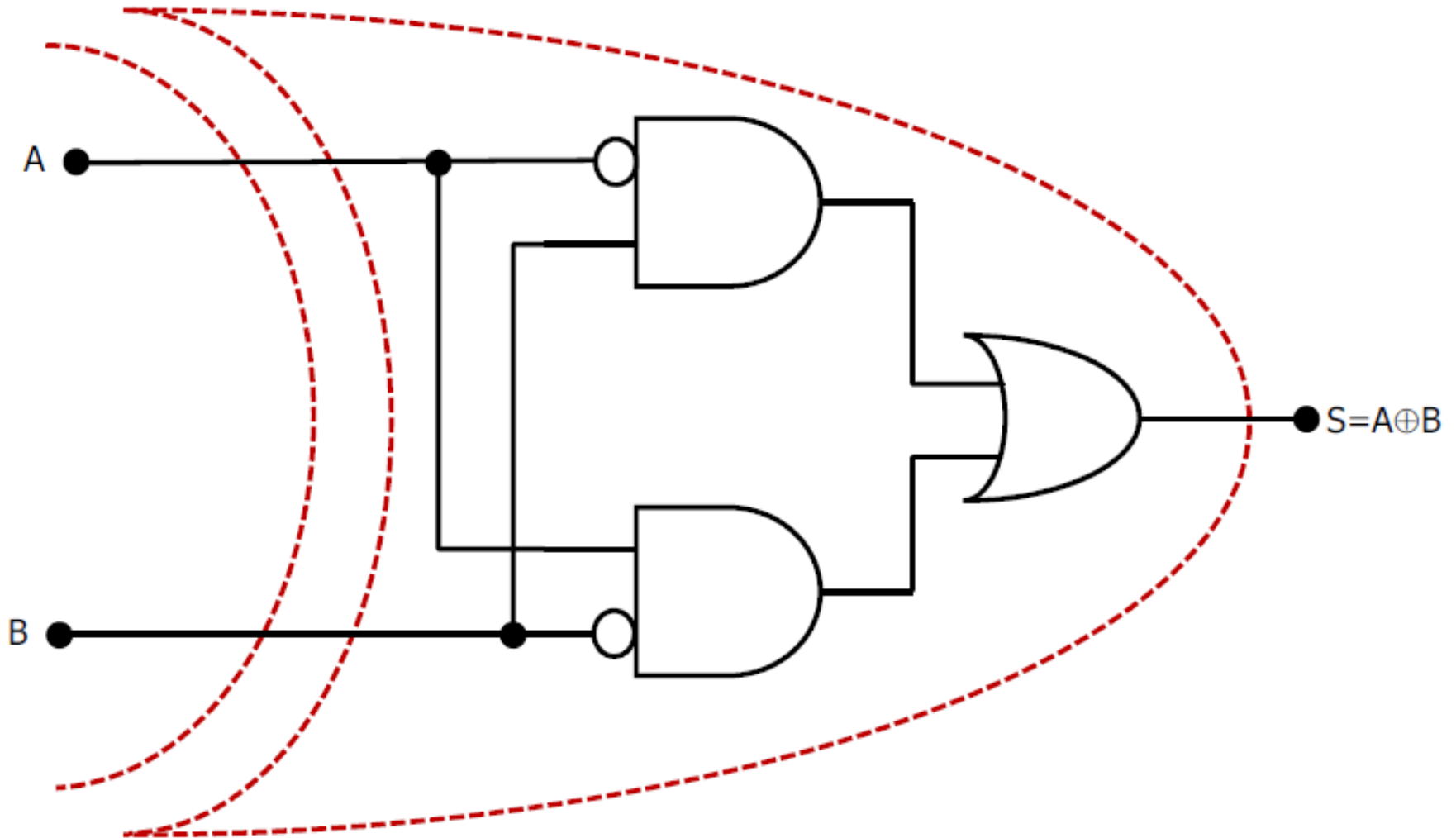
Simbologia utilizada



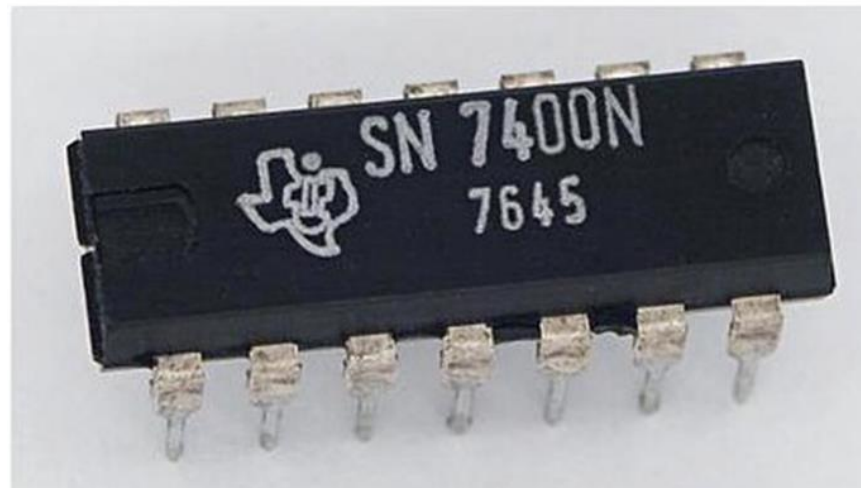
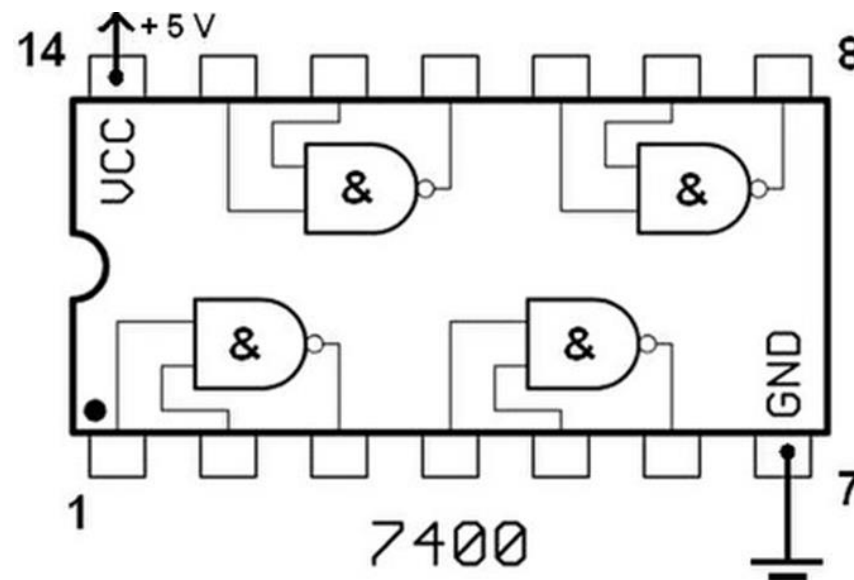
Outras simbologias



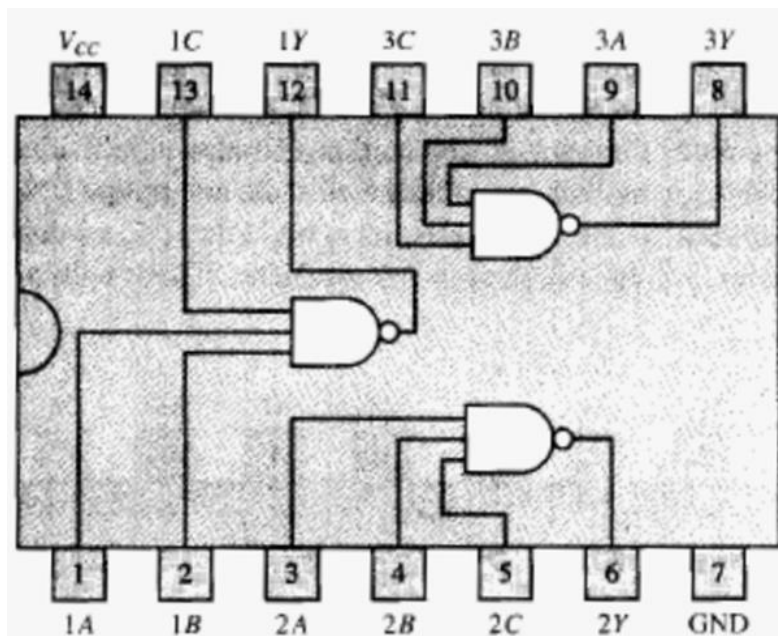
Porta OU-Exclusivo(XOR) como Circuito Combinacional



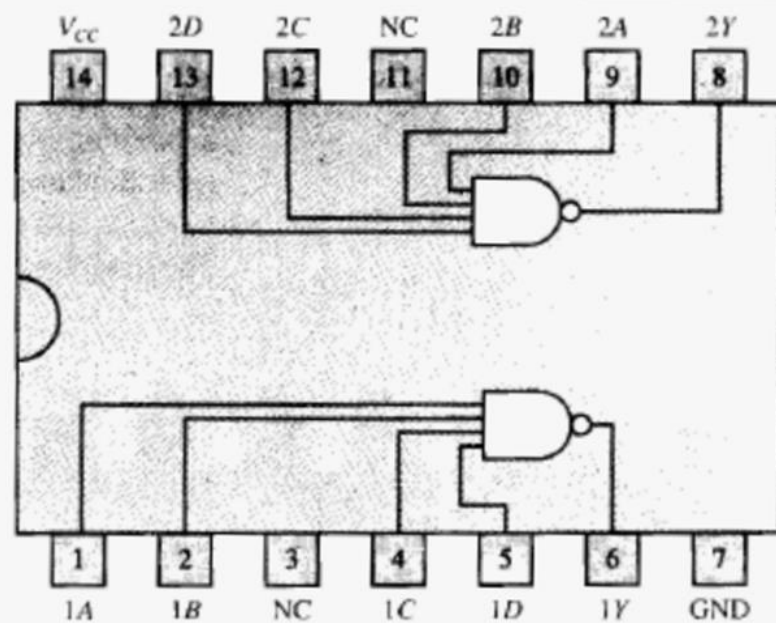
NAD - 7400 – Familia TTL



NAND – 7410 e 7420 – Familia TTL




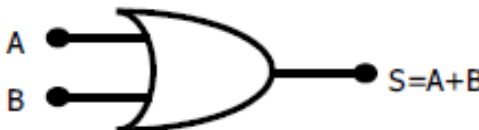
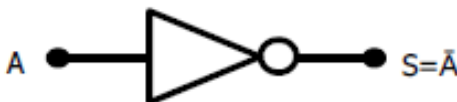



$$7410: Y = \overline{ABC}$$



$$7420: Y = \overline{ABCD}$$



Resumo dos Blocos Lógicos Básicos

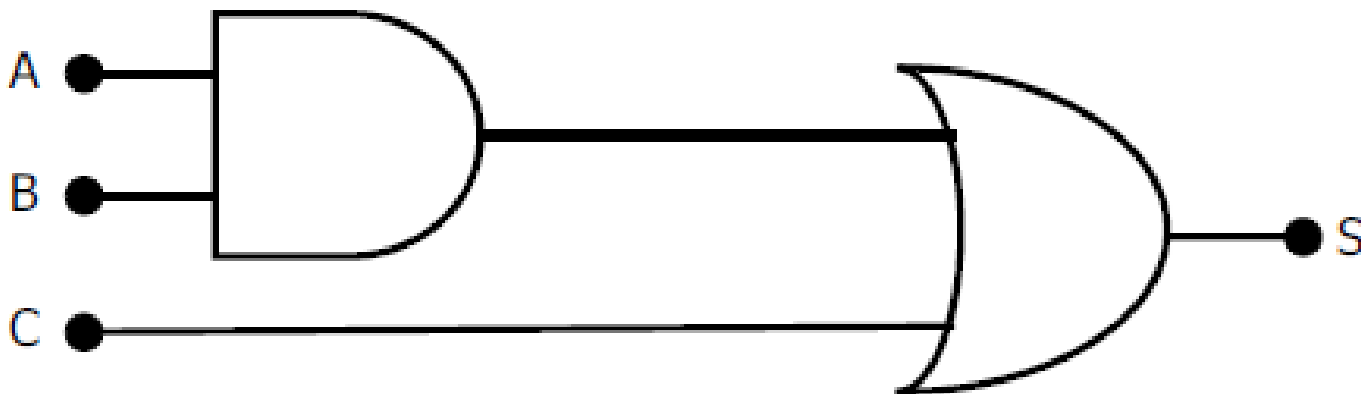
Nome	Símbolo Gráfico	Função Algébrica	Tabela Verdade															
E (AND)		$S=A.B$ $S=AB$	<table><tr><th>A</th><th>B</th><th>$S=A.B$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></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>$S=\bar{A}$</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)		$S=\overline{A.B}$ $S=(A.B)'$ $S=\neg(A.B)$	<table><tr><th>A</th><th>B</th><th>$S=\overline{A.B}$</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	$S=\overline{A.B}$	0	0	1	0	1	1	1	0	1	1	1	0
A	B	$S=\overline{A.B}$																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOU (NOR)		$S=\overline{A+B}$ $S=(A+B)'$ $S=\neg(A+B)$	<table><tr><th>A</th><th>B</th><th>$S=\overline{A+B}$</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>$S=A\oplus B$</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	$S=A\oplus B$	0	0	0	0	1	1	1	0	1	1	1	0
A	B	$S=A\oplus B$																
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

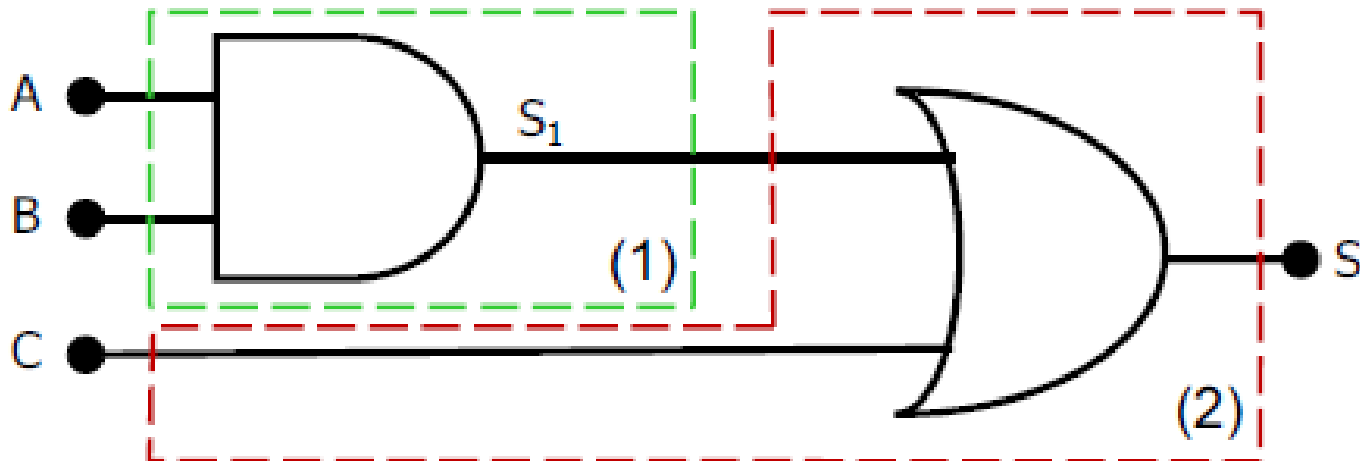
Expressões Booleanas Geradas por Circuitos Lógicos

- Seja o circuito:



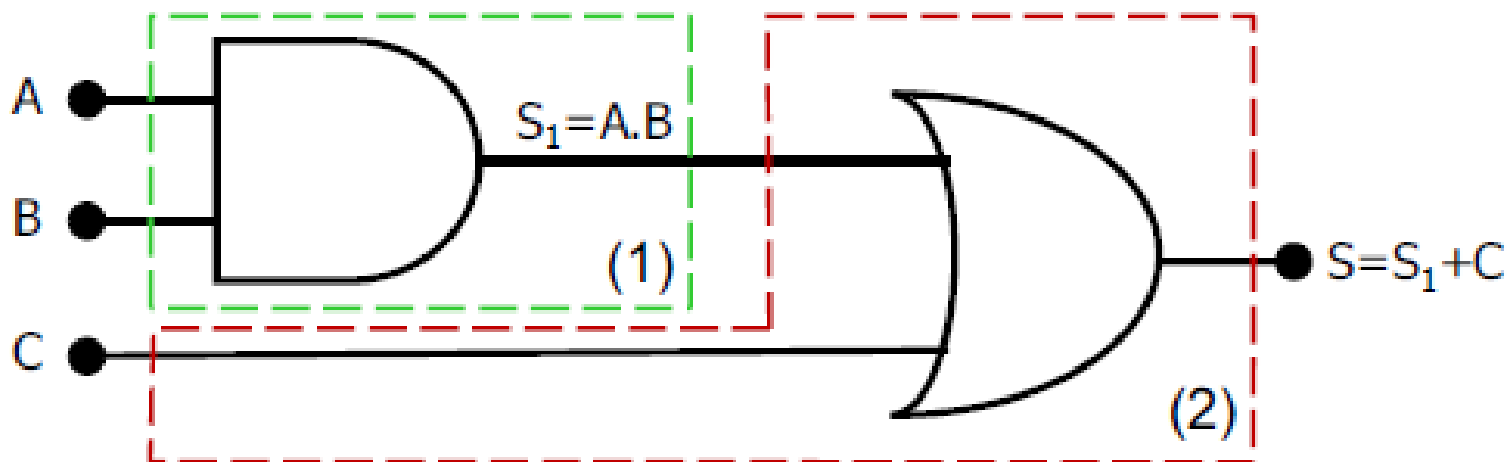
Expressões Booleanas Geradas por Circuitos Lógicos

- 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** = $A.B$



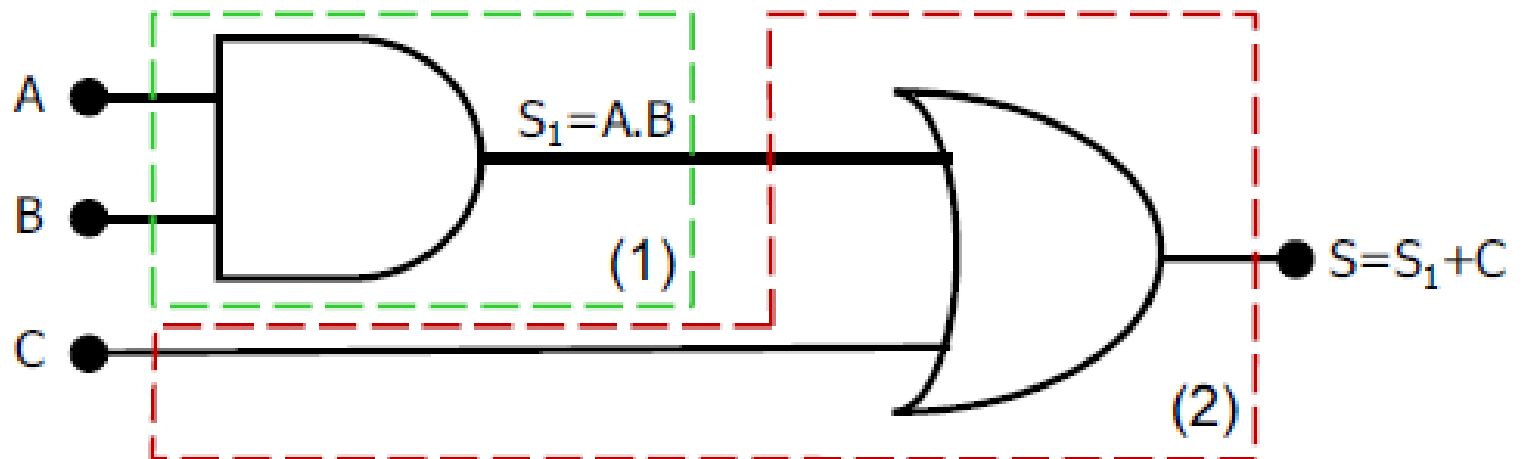
Expressões Booleanas Geradas por Circuitos Lógicos

- 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:
 - $S = S_1 + C$



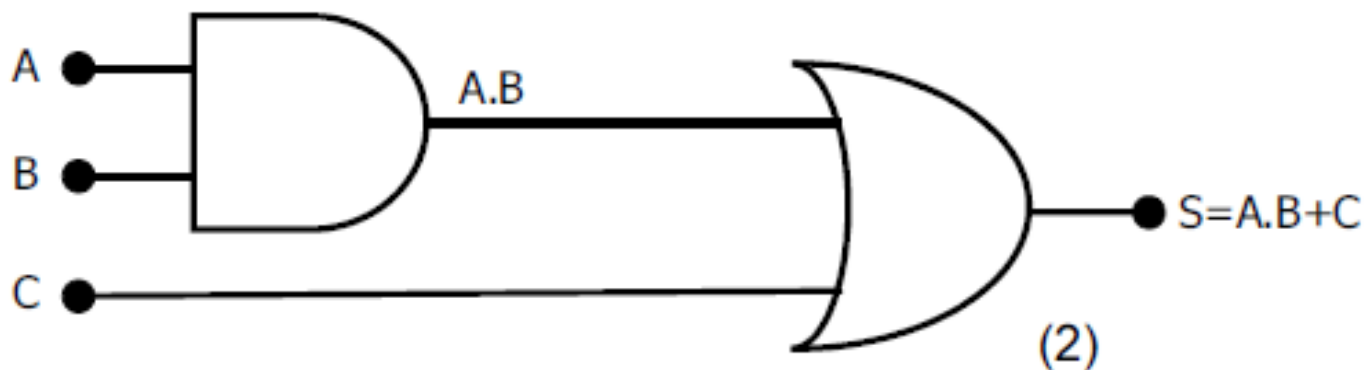
Expressões Booleanas Geradas por Circuitos Lógicos

- 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) $S_1 = A.B$
 - (2) $S = S_1 + C$
 - Obtém-se $S = S_1 + C = (A.B) + C$



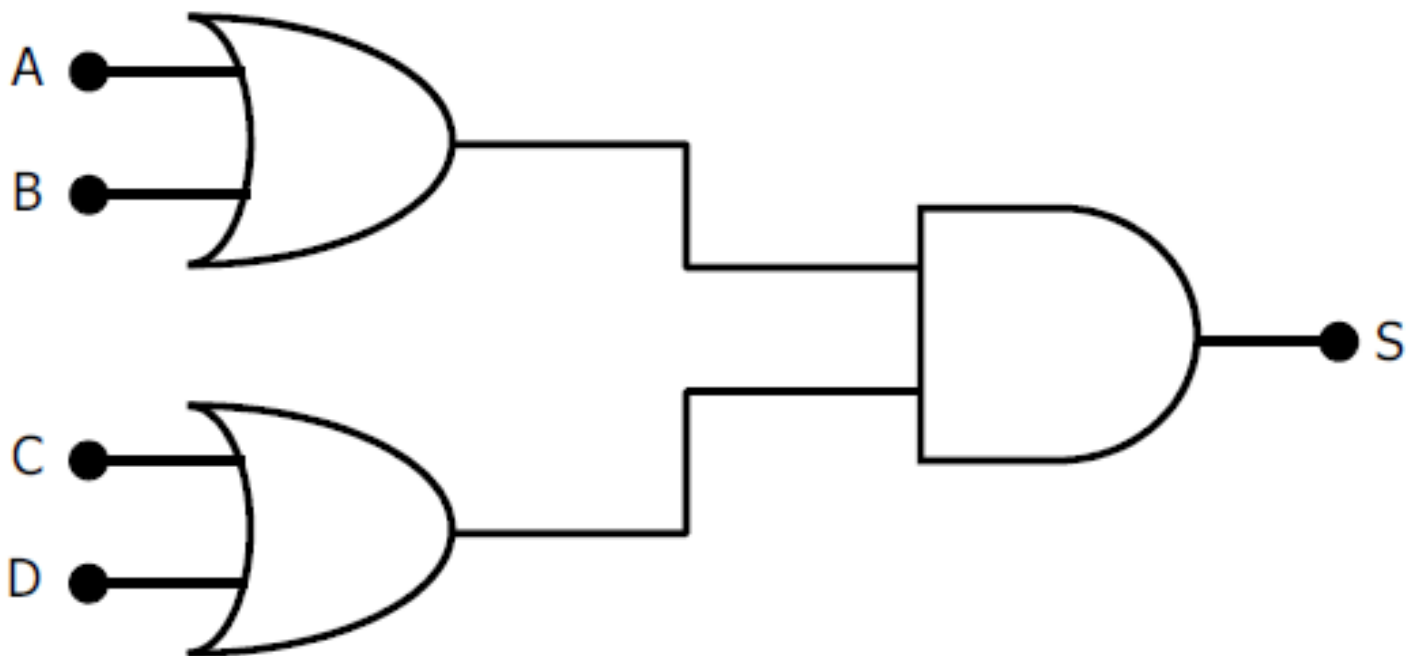
Expressões Booleanas Geradas por Circuitos Lógicos

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



Exercício

- 1) Escreva a expressão booleana executada pelo circuito

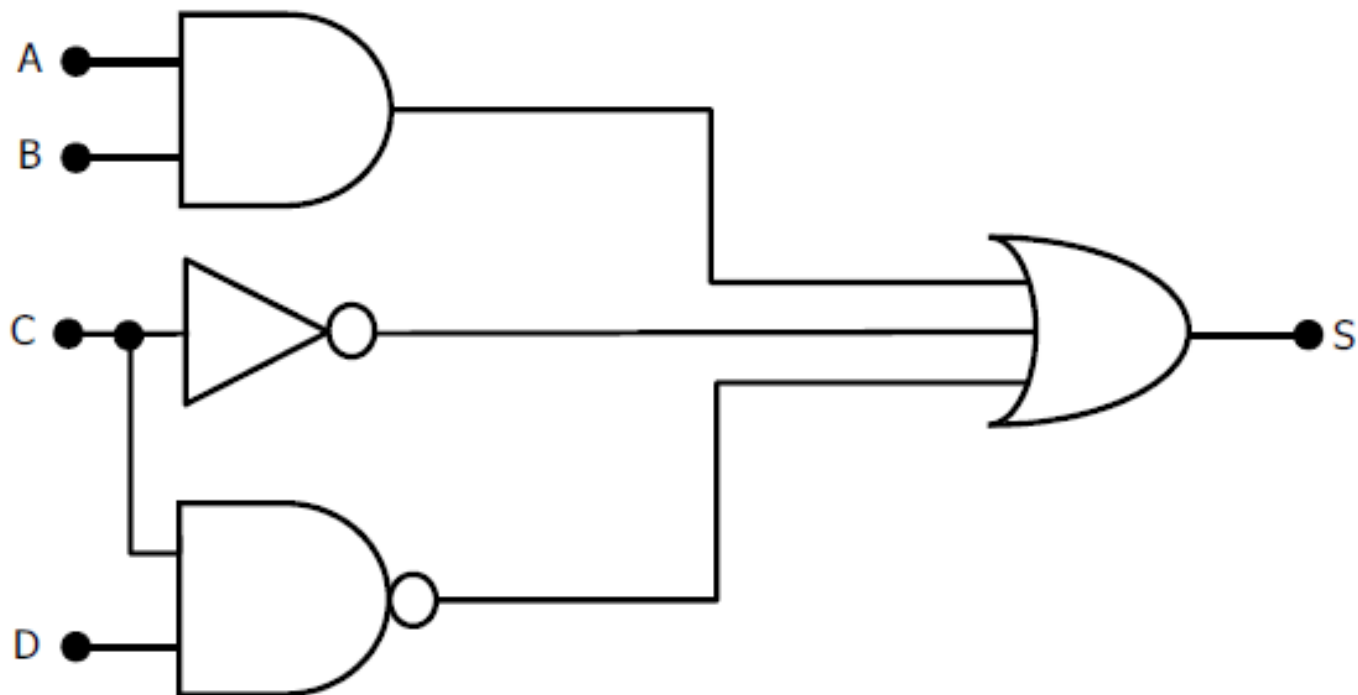




Solução Exercício 1

Exercício

- 2) Escreva a expressão booleana executada pelo circuito

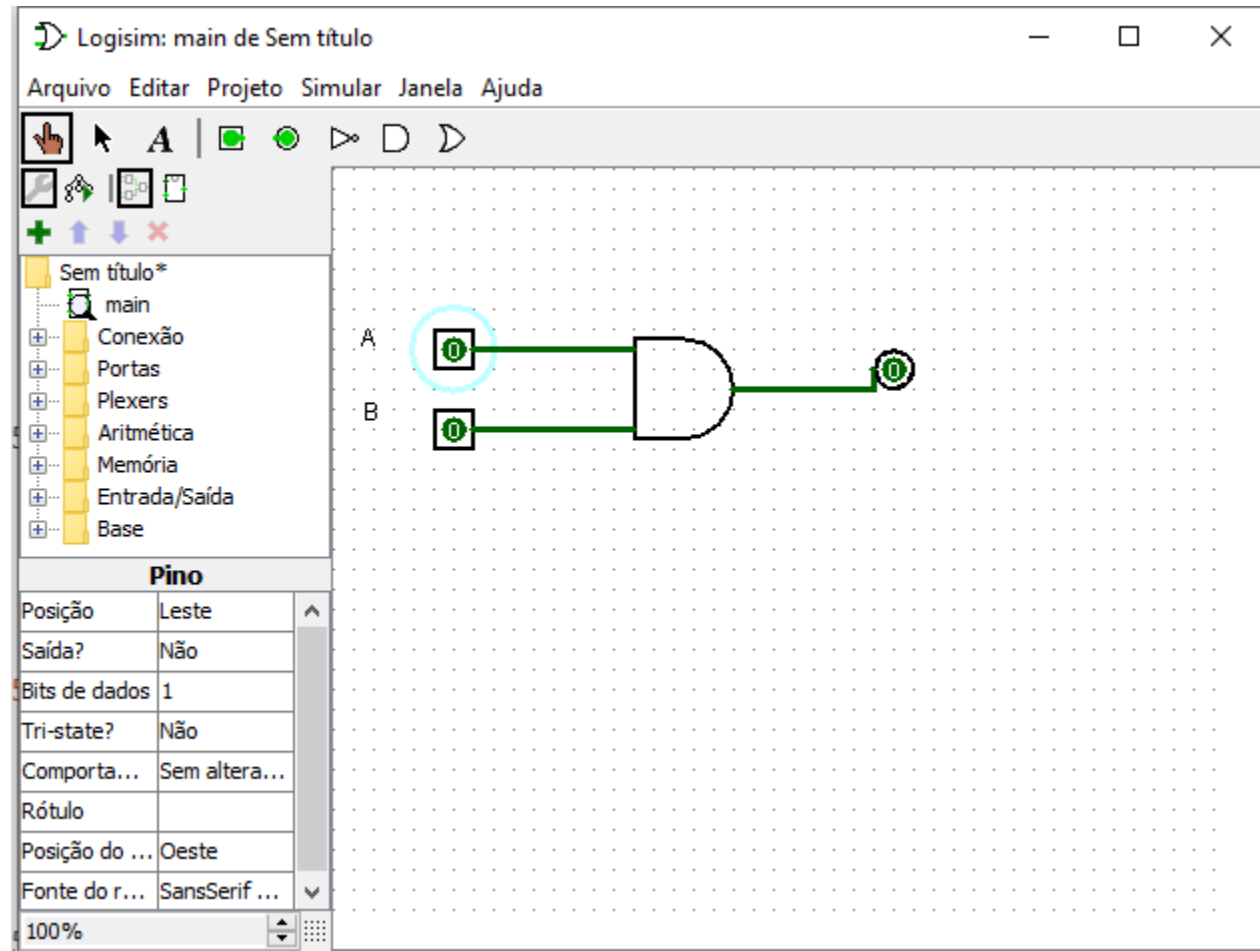




Solução Exercício 2

Logisim

- <http://www.cburch.com/logisim/pt/download.html>

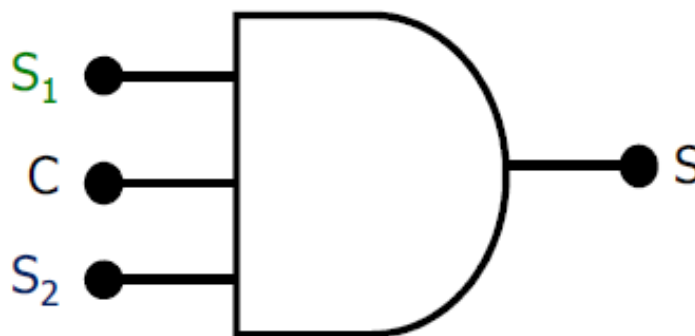
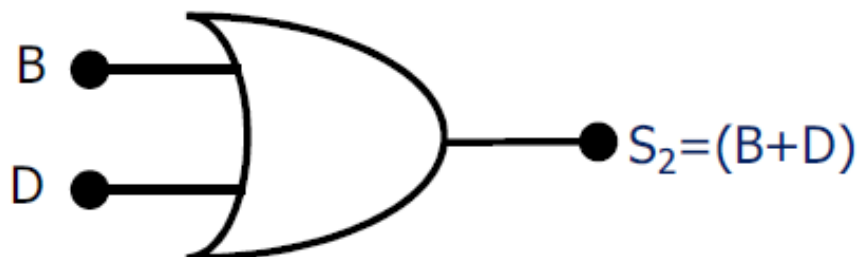
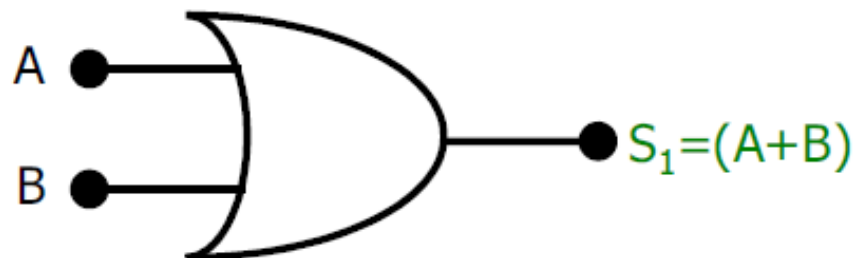


Circuitos Gerados por Expressões Booleanas

- 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**)

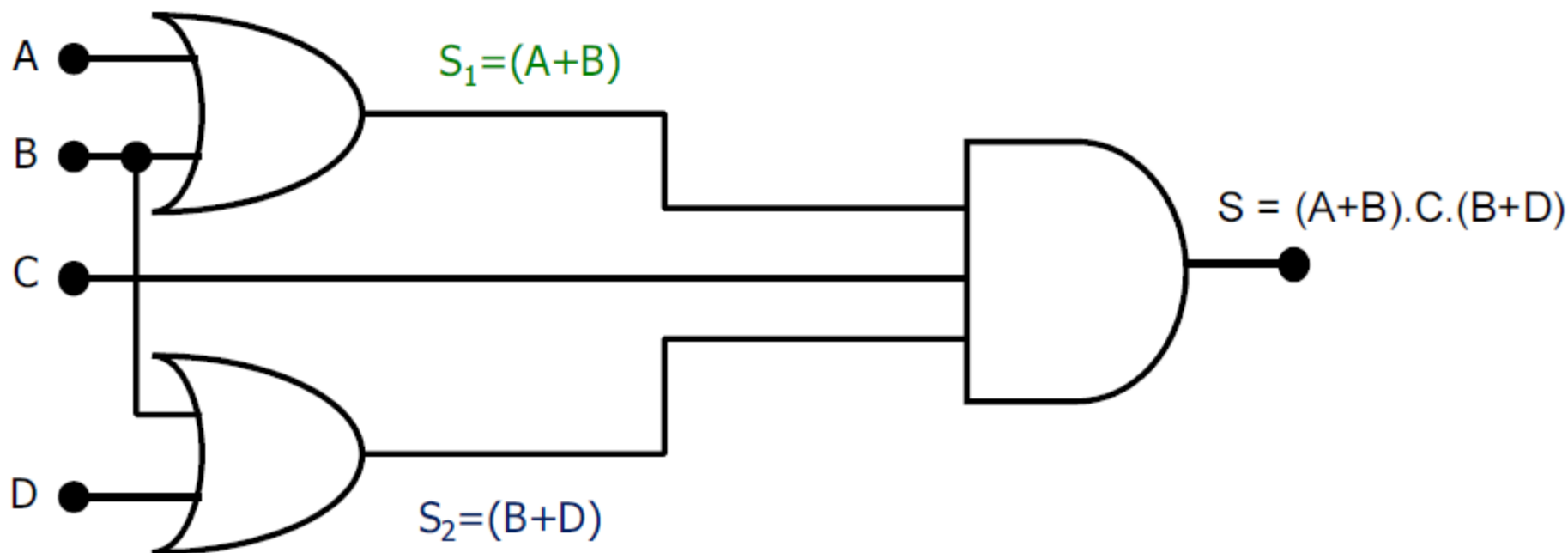
Circuitos Gerados por Expressões Booleanas

- Seja a expressão
 - $S = (A+B).C.(B+D)$
- Vamos separar as sub-fórmulas da expressão, ou seja:
 - $S = (A+B) . C . (B+D)$
- Dentro do primeiro parêntese temos a soma booleana $S_1=(A+B)$, portanto o circuito que executa esse parêntese será uma **porta OU**
- Dentro do segundo parêntese temos a soma booleana $S_2=(B+D)$. Novamente, o circuito que executa esse parêntese será uma **porta OU**
- Portanto, temos:
 - $S = S_1 . C . S_2$
- 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 – Busca Ativa

- Baixe o Logisim
 - E refaça os exercícios e os exemplos anteriores

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

$$S = \overline{\overline{A.B + C.D}} \quad \text{ou}$$

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

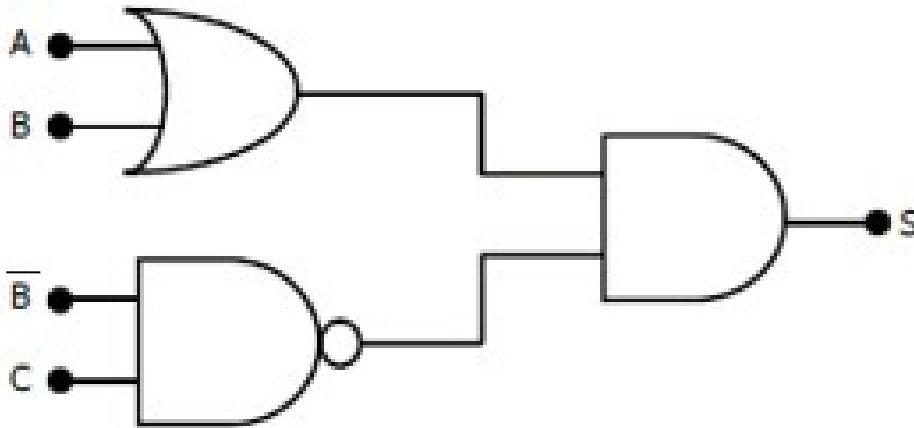


Solução Exercício 4

Exercício

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

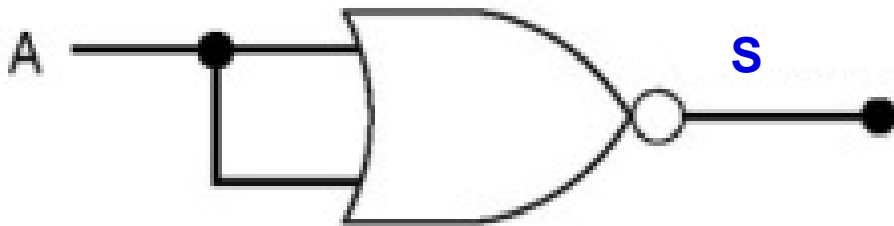
Qual o resultado?



Exercício

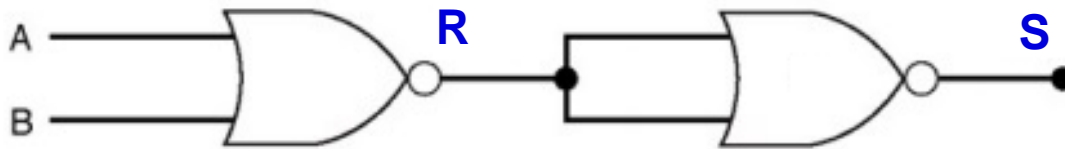
6) Considerando os valores de $A=0$.

Qual o resultado?



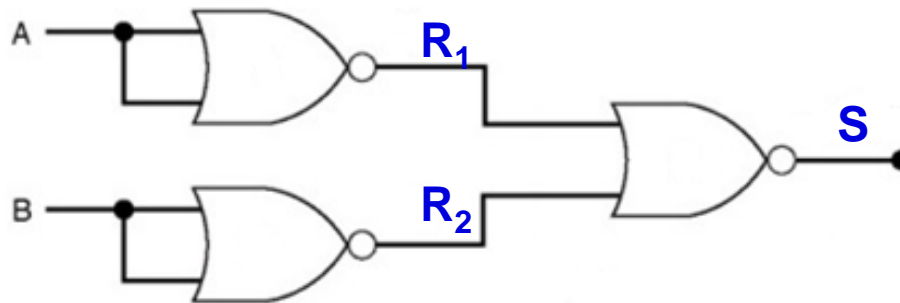
Exercício

7) Considerando os valores de $A=0$ e $B=1$.
Qual o resultado?



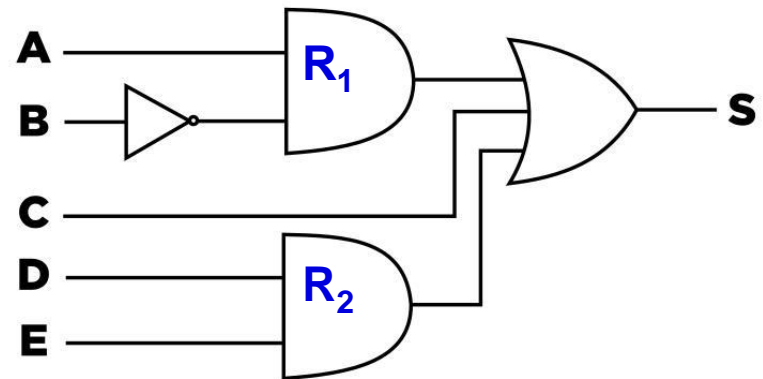
Exercício

8) Considerando os valores de $A=1$ e $B=1$.
Qual o resultado?



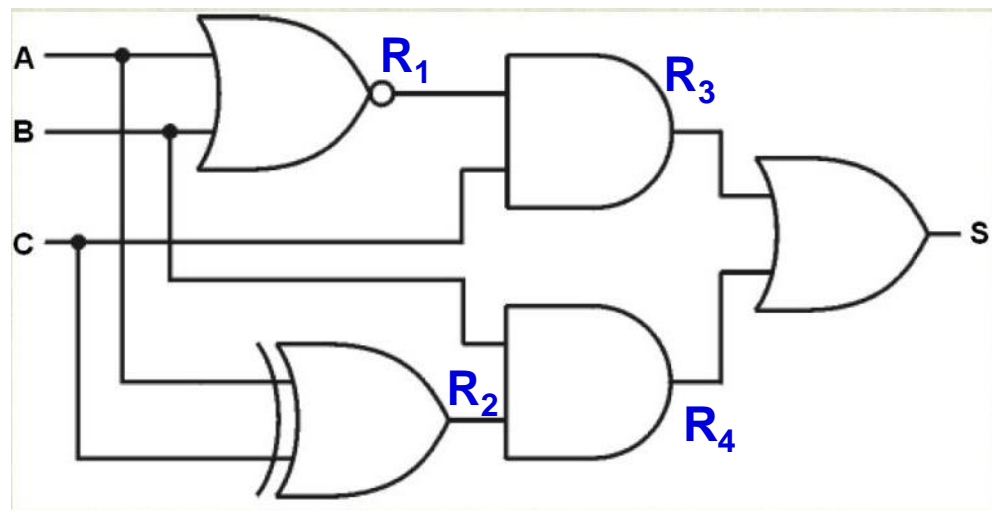
Exercício

9) Considerando os valores de $A=0$, $B=1$, $C=0$, $D=1$ e $E=1$. Qual o resultado?



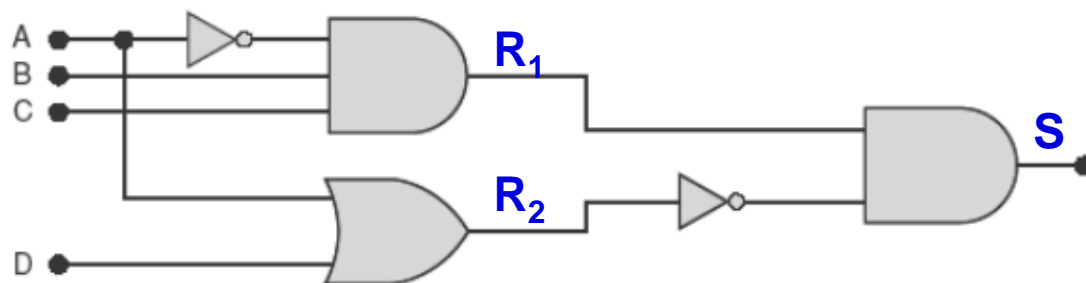
Exercício

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



Exercício

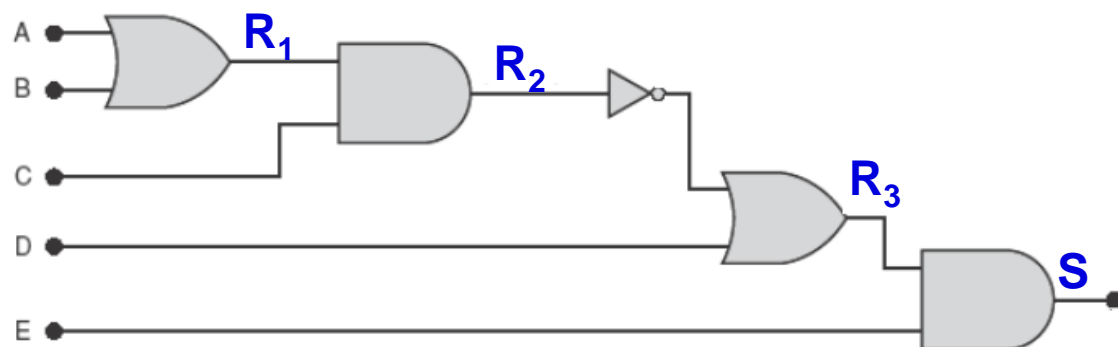
11) Escreva as expressões booleanas e resolva.



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

Exercício

12) Escreva as expressões booleanas e resolva.



$$A = 0$$

$$B = 1$$

$$C = 1$$

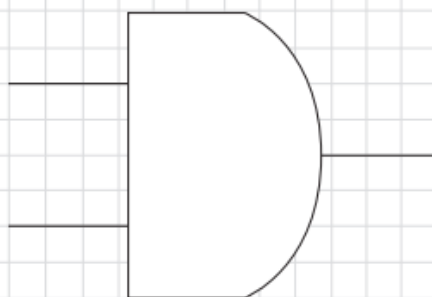
$$D = 0$$

$$E = 1$$

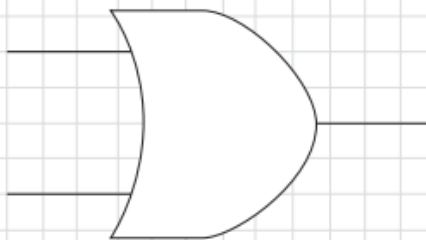
Questão ENADE



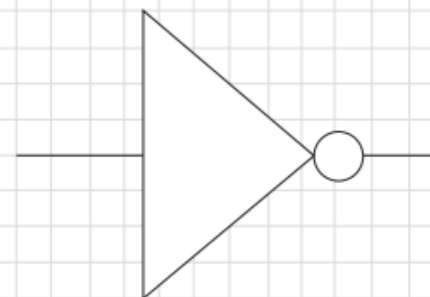
(ENADE 2021) Em 1938, o matemático americano Claude Shannon notou o paralelismo entre a lógica proposicional e a lógica dos circuitos e percebeu que a álgebra booleana teria um papel importante na sistematização deste ramo da eletrônica. Cada um dos conectivos básicos da lógica são instâncias das operações básicas da álgebra booleana (“+”, “.” e “' ”). Expressões booleanas combinando operações e variáveis podem ser usadas para representar circuitos combinacionais formados por portas lógicas. GERSTING, J. L. Mathematical Structures for Computer Science. New York: W. H. Freeman and Company, 2002.



Porta “e” (and)



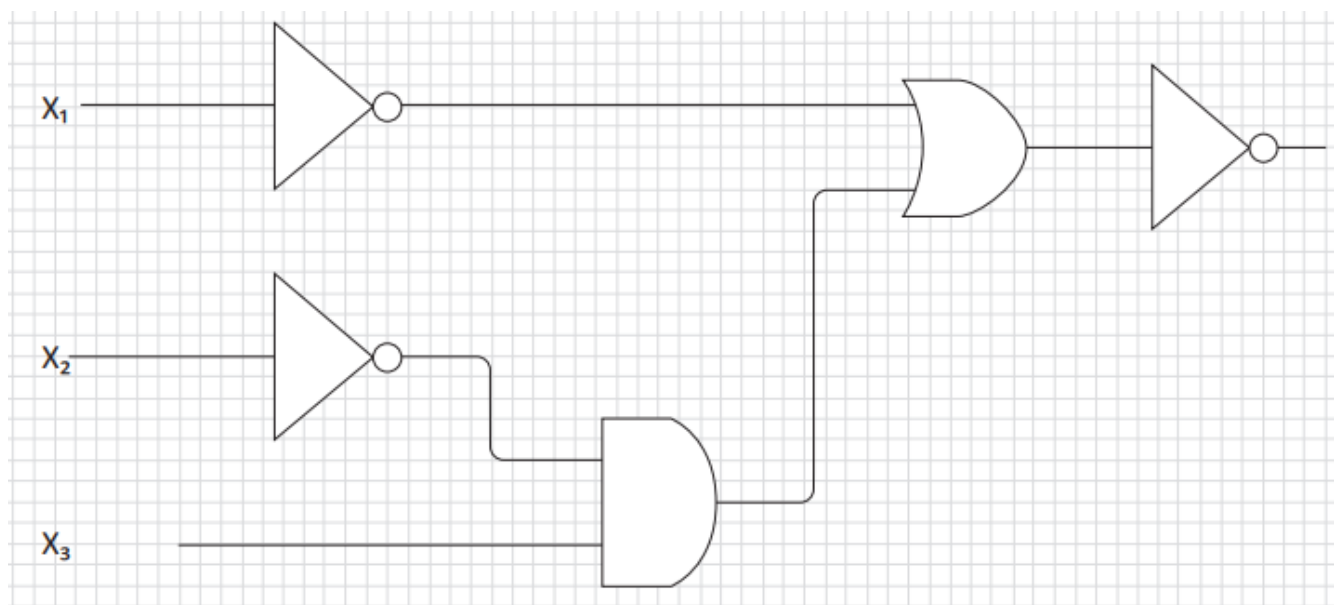
Porta “ou” (or)



Porta “não” (not)

Questão ENADE

A partir das informações apresentadas, considere o circuito combinacional da figura a seguir. A figura a seguir apresenta as portas básicas.



Qual das alternativas apresenta a expressão booleana correspondente?

- A $(X3 \cdot X2') + X1'$
- B $(X3 \cdot (X2') + (X1'))'$
- C $((X3 \cdot X2)' + X1')'$
- D $(X3 \cdot X2)' + X1'$
- E $((X3 \cdot X2')' + X1')'$

Exercício

13) Vamos construir um circuito sinalizar a disponibilidade de banheiro de aeronave;

A aeronave possui 3 lavatórios: cada lavatório tem um sensor que possui valor lógico 1 se a porta estiver fechada (A, B, C);

A lâmpada (sinal S) acende se houver pelo menos um lavatório disponível (algum sensor de porta retorna 0);

Qual o circuito e a expressão booleana para a lâmpada acender?



Exercício 13

Meio Somador



- As regras básicas para adição binária são:
 - $0 + 0 = 0$
 - $0 + 1 = 1$
 - $1 + 0 = 1$
 - $1 + 1 = 0 \Rightarrow 10$ (e “troca” para o dígito de ordem superior)
- Estas operações são realizadas por um circuito lógico denominado **Meio Somador** (*Half Adder*)

Meio Somador

- Um **Meio Somador** recebe dois bits de entrada A e B e produz dois bits de saída: o **Bit de Soma** ($\Sigma = A \oplus B$) e o **Bit de Carry** (C_{out}):

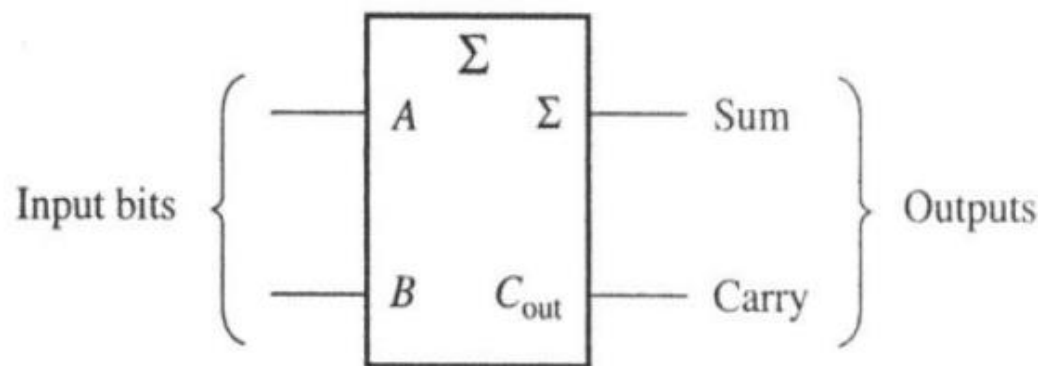
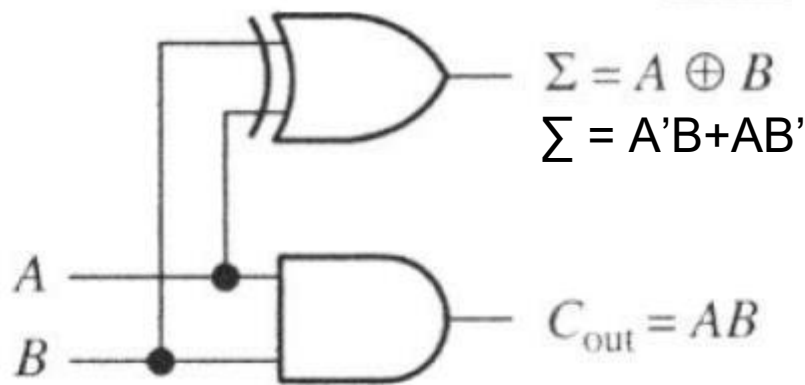
- **Somar:**

- $0110_2 = 6_{10}$

- $0100_2 = 4_{10}$

- **Resultado:**

- $1010_2 = 10_{10}$



$$\begin{array}{r}
 A \\
 + B \\
 \hline
 C_{out} \Sigma
 \end{array}$$

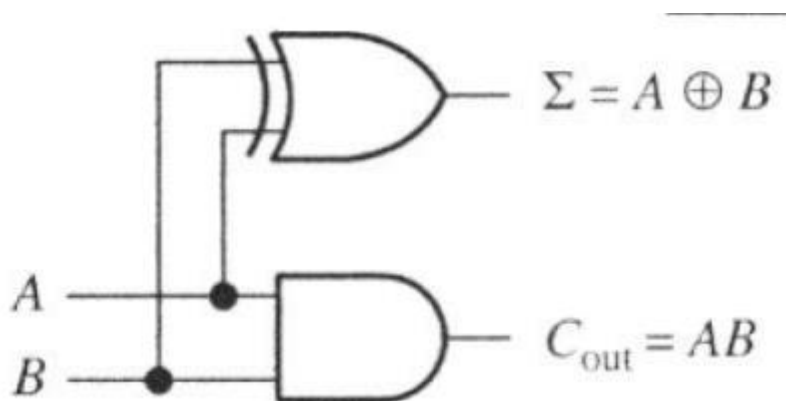
A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Meio Somador

- Considerem os dois números binários em 4 bits:

- $0110_2 = 6_{10}$

- $0100_2 = 4_{10}$

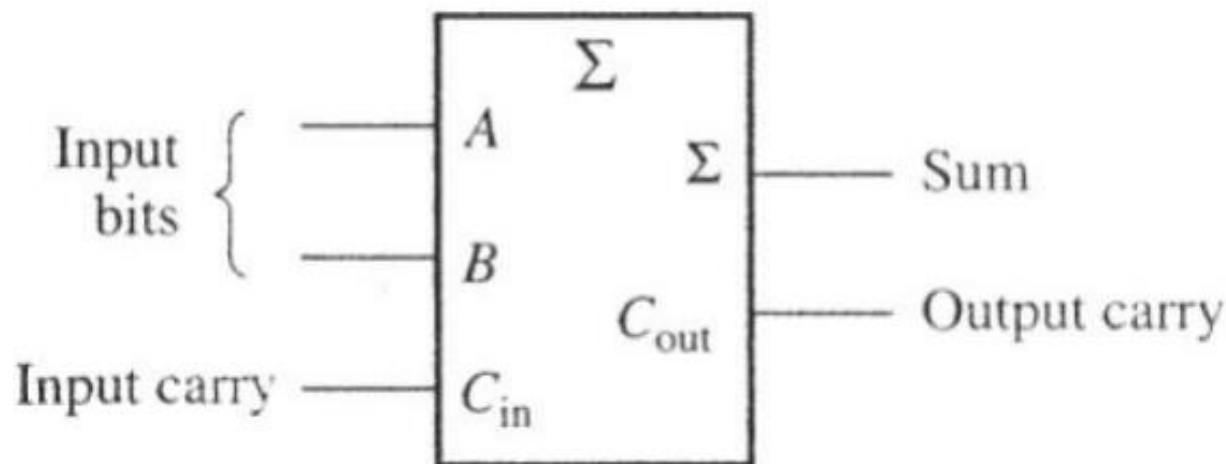


$$\begin{array}{r} 1 \text{ troca/ vai um carry} \\ \downarrow \\ (0110)_2 \\ \oplus (0100)_2 \\ \hline (1010)_2 \end{array}$$

- $6 + 4 = 10$

Somador Completo

- Um **Somador Completo** (***Full Adder***) recebe três bits de entrada A , B e C_{in} (o último correspondendo ao eventual *carry* gerado na operação com bits menos significativos) e produz dois bits de saída: o **Bit de Soma** ($\Sigma = A + B$) e o **Bit de Carry** (C_{out}).



Somador Completo - Soma

■ Tabela Verdade

Entrada			Saída	
A	B	C _{in}	C _{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S

AB\C _{in}	0	1
00	0	1
01	1	0
11	0	1
10	1	0

$$S = A \oplus B \oplus C_{in}$$

$$\Sigma = A'.B'.C_{in} + A'.B.C_{in}' + A.B'.C_{in}' + A.B.C_{in}$$

Somador Completo – Carry_{out}

■ Tabela Verdade

Entrada			Saída	
A	B	C _{in}	C _{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

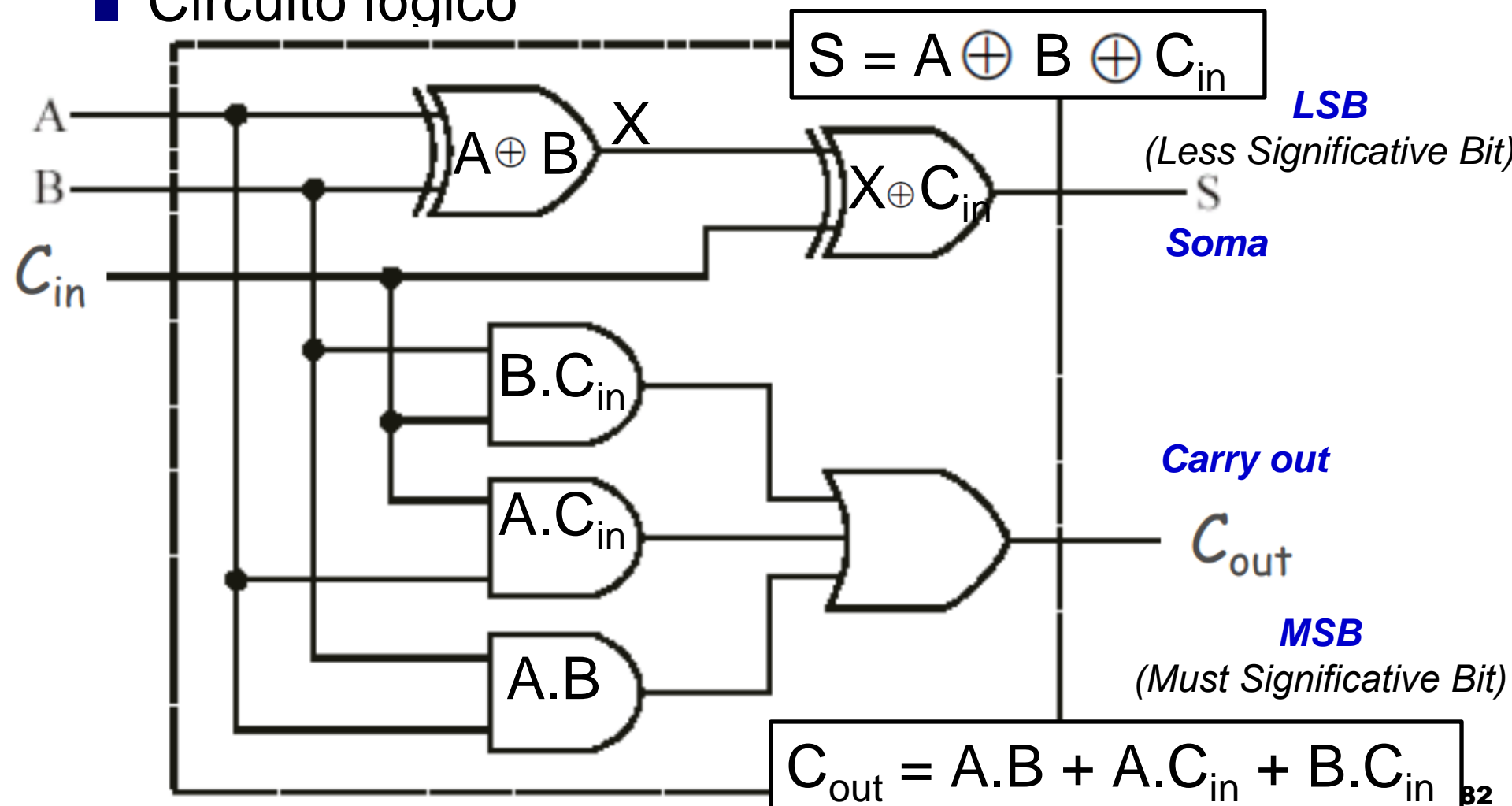
C_{out}

AB\C _{in}	0	1
00	0	1
01	1	1
11	0	1
10	0	0

$$C_{out} = A.B + A.C_{in} + B.C_{in}$$

Somador Completo

■ Circuito lógico



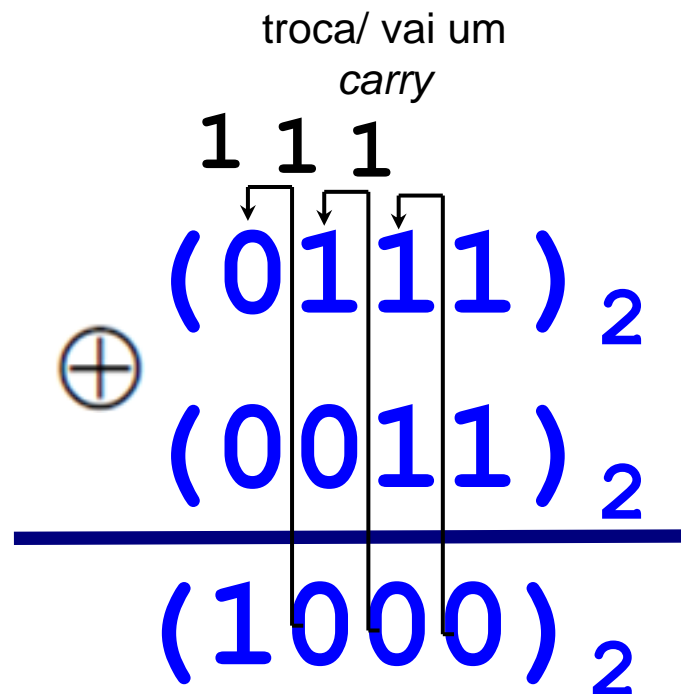
Somador Completo

- Considerem os dois números binários em 4 bits:

- $0111_2 = 7_{10}$

- $0011_2 = 3_{10}$

- Utiliza-se 4 somadores completos, um para cada bit;
- Conecta-se cada C_{out} no C_{in} do próximo bit;
- Para o LSB(Bit da extrema direita) pode ser utilizado um meio somador.



- $7 + 3 = 10$

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

BARANAUSKAS, José Augusto. Funções Lógicas e Portas Lógicas. FFCLRP-USP. 2012. Disponível em:

<http://dcm.ffclrp.usp.br/~augusto/teaching/aba/ABFuncoes-Logicas-Portas-Logicas.pdf>. Acesso em: 19 mar. 2021.

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