Lógica e Sistemas Digitais - 1

Análise e Síntese de Circuitos Combinatórios

ISEL
Departamento de Engenharia de Electrónica
e Telecomunicações e de Computadores
Lisboa

Mário Araújo

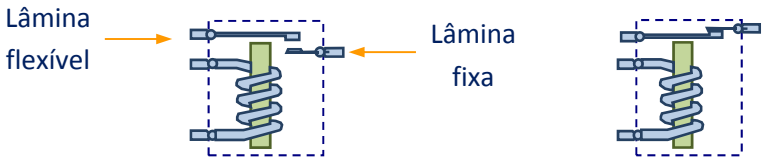
2016-1

ÁLGEBRA DE COMUTAÇÃO

Claude Shannon aplicou em 1938 os estudos de George Boole a circuitos constituídos por dispositivos binários que eram na altura interruptores e comutadores mecânicos e relés electromecânicos. Criou com isso a **Álgebra de Comutação** (**Switching Algebra**).

Um relé é um elemento de comutação constituído por um electroíman, uma armadura e um conjunto de interruptores de duas lâminas, sendo uma fixa e outra flexível.

A bobina do electroíman, ao ser percorrida por corrente eléctrica, atrai a lâmina flexível fechando ou abrindo os contactos associados consoante a configuração do relé, que pode ser do tipo **NO** (NORMALLY OPEN) ou **NC** (NORMALLY CLOSED).



Relé com contacto normalmente aberto (dito NO, de NORMALLY OPEN).

Relé com contacto normalmente fechado (dito NC, de NORMALLY CLOSED).

Isto tem a ver com a configuração dos contactos na **posição de repouso**: podem estar fechados recebendo a designação de NC, ou abertos, recebendo a designação de NO.

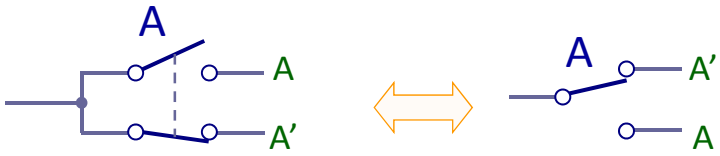
Tal como os relés, também os Interruptores podem ser do tipo NO e NC.

A representação dos contactos é sempre feita na **posição de repouso** (interruptor, comutador ou relé **desactivado**).

A cada contacto é atribuída uma designação, letra ou nome que pode ter um de dois valores possíveis, 0 ou 1.

A, **B** … ou **a, b…** são designações habituais de variáveis em qualquer álgebra.

Interruptores **SPST** (SINGLE POLE, SINGLE THROW)



Interruptores (comutadores) **SPDT** (SINGLE POLE, DOUBLE THROW).

A ligação a traço interrompido em cima significa que ambos os contactos pertencem ao mesmo comutador e são actuados simultaneamente.

Uma MALHA DE COMUTAÇÃO é obtida pela interligação dos contactos de vários actuadores mecânicos ou electromecânicos : interruptores, comutadores, ou relés.

A, **B** … **L** são designações de variáveis que utilizaremos na malha simples deste slide.

A e **B** são variáveis binárias ou **booleanas INDEPENDENTES** que representam os contactos associados a dois interruptores a que se atribui o valor lógico:

- **1** quando os interruptores se encontram **actuados**, e
- **0** quando não estão actuados.

L é uma variável binária **DEPENDENTE** que representa uma lâmpada a que se atribui o valor lógico:

- **1** quando a lâmpada se encontra acesa, e
- **0** quando a lâmpada se encontra apagada.

A, **B** e **L** apenas podem tomar valores no conjunto {0, 1}:

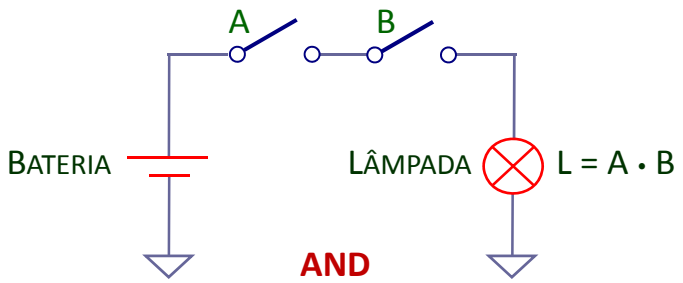
$$A = \{0, 1\}$$
$$B = \{0, 1\}$$
$$L = \{0, 1\}$$

A lâmpada **L** acende quando ambos os interruptores **A** e **B** estiverem actuados. L é s uma função booleana simples de duas variáveis booleanas gerais.

O valor de L diz-se então resultado da operação **AND** entre as variáveis A e B, pois que só toma o valor lógico 1 quando ambas as variáveis tiverem o valor 1.

L = 1 só quando **A** e (AND) **B** tiverem ambos o valor 1:

$L = A \cdot B$



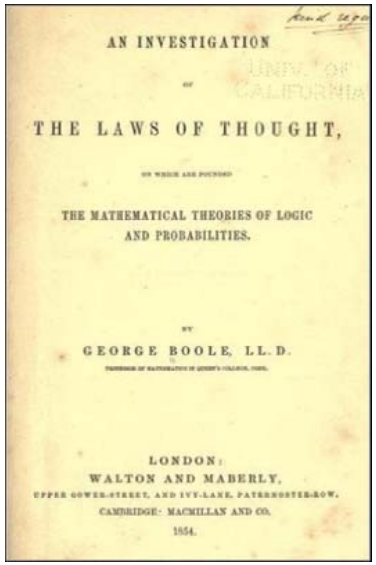
B	A	L = A • B
0	0	0
0	1	0
1	0	0
1	1	1

Representação da operação lógica AND em malha de comutação (em cima) e em tabela de verdade (ao lado).

GEORGE BOOLE E CLAUDE SHANNON

George Boole

1815-1864
(UK)



O matemático inglês George Boole é considerado o pai da lógica simbólica, e um génio esquecido. Sem preparação universitária foi um autodidata que ascendeu à cátedra de Matemática da Universidade de Cork na Irlanda. Criou um sistema algébrico de 2 valores, hoje em dia designado por Álgebra de Boole, na tentativa de dar expressão às leis fundamentais do pensamento na linguagem simbólica do Cálculo.

O seu primeiro trabalho publicado em 1847 designou-se **Mathematical Analysis of Logic** mas a sua obra prima foi um tratado de 1854 intitulado **An Investigation of the Laws of Thought (1854), on Which are Founded the Mathematical Theories of Logic and Probabilities.**

A lógica simbólica ficou esquecida por muitos anos depois de sua morte precoce devida a uma pneumonia. Foi o trabalho de Whitehead e Russel em **Principia Mathematica** (1910-1913) que o ressuscitou.

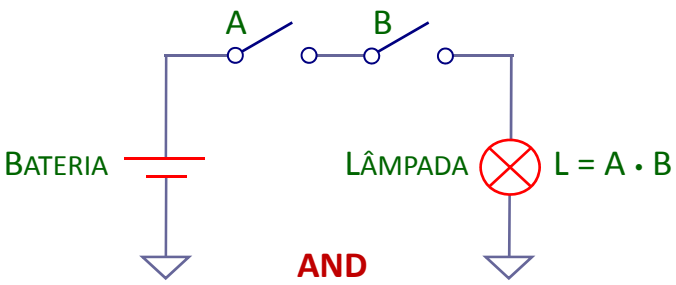
Claude Shannon

1916-2001
(USA)

Claude Shannon foi considerado o pai da Teoria da Informação era dotado de uma criatividade ímpar. Estudou inicialmente Matemática e Engenharia Electrotécnica na Universidade de Michigan. Em 1940 apresentou no MIT a sua tese de doutoramento em Matemática – intitulada **An Algebra for Theoretical Genetics** – e simultaneamente a sua tese de Mestrado em Engenharia Electrotécnica, intitulada **A Symbolic Analysis of Relay and Switching Circuits**. Esta foi considerada uma das mais influentes obras de todos os tempos no estabelecimento da modelação matemática e da fundamentação teórica dos sistemas digitais.

Fez investigações importantes no desenvolvimento dos sistemas comunicacionais em que evidenciou o uso das potencialidades do código binário. Deixou outros trabalhos seminais entre os quais se destacam **The Mathematical Theory of Communication**, com W. Weaver (1948). Trabalhou na ATT Bell Labs de 1941 a 1972 e foi Professor Emérito do MIT desde 1978 até à data da sua morte.

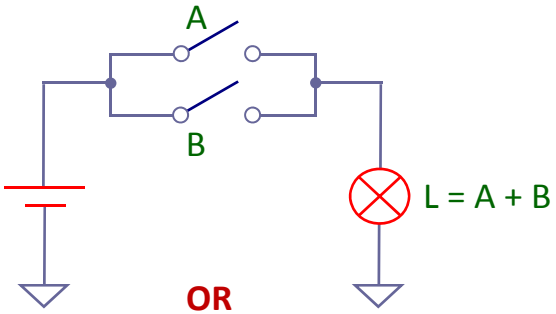
Representação em Malhas de Comutação e Tabelas de Verdade das operações lógicas AND, OR e NOT



B	A	L = A • B
0	0	0
0	1	0
1	0	0
1	1	1

L = 1 quando **A e** (AND) **B** tiverem o valor 1 (quando ambos os interruptores forem actuados):

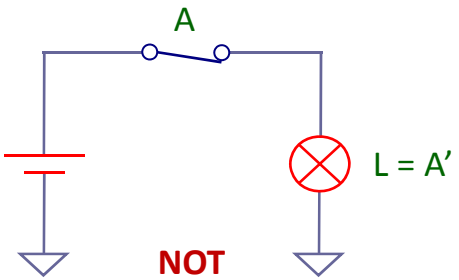
$L = A \cdot B$



B	A	L = A + B
0	0	0
0	1	1
1	0	1
1	1	1

L = 1 quando **A ou** (OR) **B** tiverem o valor 1 (quando um dos interruptores ou ambos forem actuados):

$L = A + B$



A	L = A'
0	1
1	0

L = 1 quando **A** tiver o valor 0 (NOT) (quando não for actuado o interruptor A):

$L = A'$

Representação em malha de comutação e tabela de verdade das operações lógica AND, OR e NOT.

	I	T	P	C	M
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

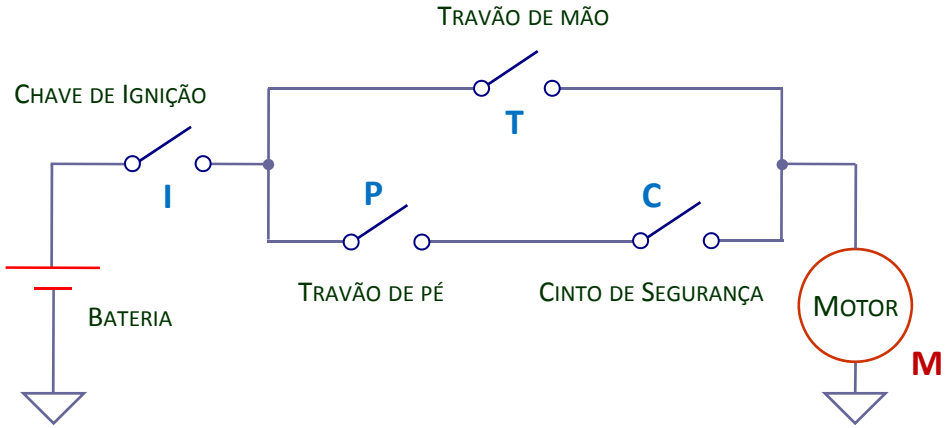
Tabela de verdade representativa das condições de funcionamento de um motor M.

O comportamento de cada malha pode ser representado sob a forma de uma tabela que lista exaustivamente as combinações das variáveis de entrada e o valor da saída para cada uma delas.

Pretende-se que o motor de arranque (M) de um automóvel com caixa automática só entre em funcionamento se:

- estiver ligada a chave de ignição (I) **e**
- estiver actuado o travão de mão (T) , **ou** em alternativa, estiver actuado o travão de pé (P) **e** apertado o cinto de segurança (C).

Sendo as variáveis independentes I, T, P e C, o comportamento pode ser representado pela tabela de verdade e malha desenhadas.

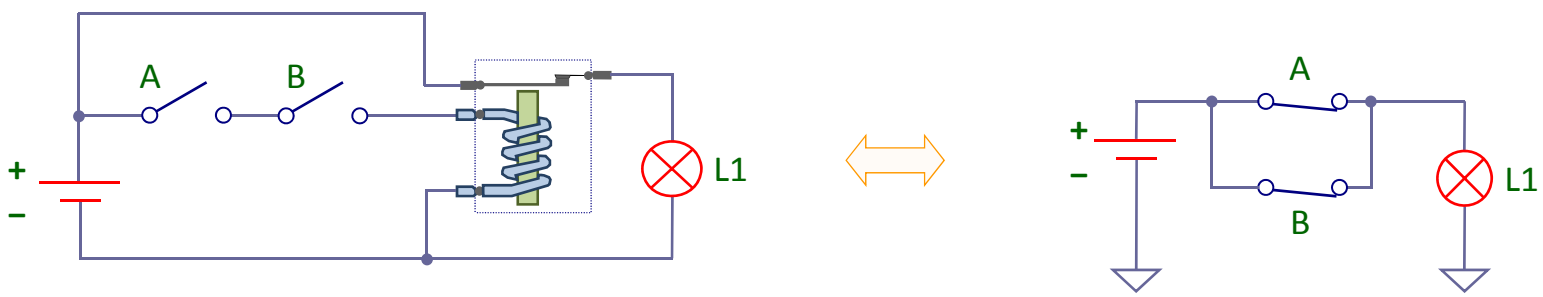


Representação em malha de comutação simbólica das condições de funcionamento do motor M.

$$M = I\,(T + P \cdot C)$$

Equação algébrica representativa das condições de funcionamento do motor M.

Representação de Funções em Malhas de Comutação

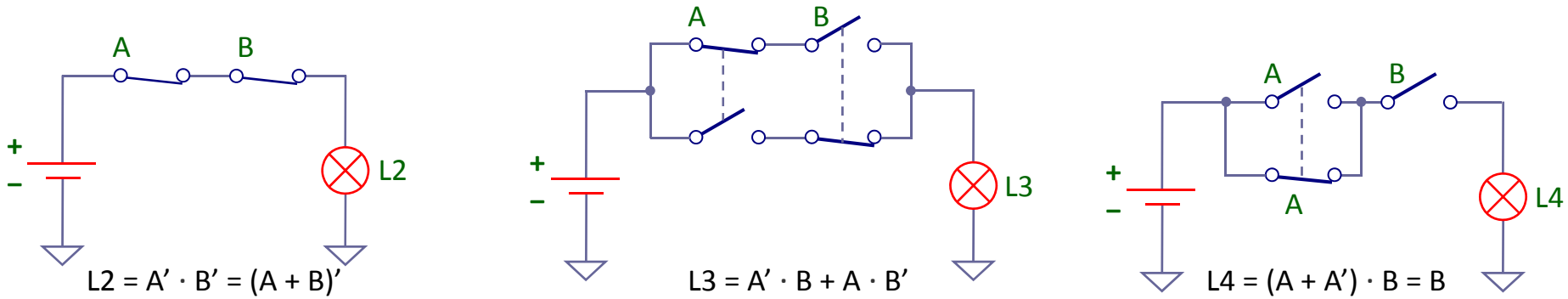


Representação em 2 malhas de comutação equivalentes da mesma função L1.

Os circuitos acima são equivalentes. A Lâmpada L1 apaga quando os interruptores A e B forem simultaneamente actuados. Isto significa o mesmo que negar a função AB.

$$L1 = A' + B' = (A \cdot B)'$$

Equação algébrica representativa das condições acendimento da lâmpada L1.



Representação em malhas de comutação e equações algébricas das funções L2, L3 e L4.

A Lâmpada L2 apaga quando um dos interruptores, ou ambos, estiverem actuados, que é o mesmo que dizer que acende quando nenhum interruptor estiver actuado.

A Lâmpada L3 acende quando um e um só dos interruptores, mas não ambos, estiver actuado.

A Lâmpada L4 acende quando interruptor B, estiver actuado, sendo independente do interruptor A.

SIMPLIFICAÇÃO DE MALHAS DE COMUTAÇÃO

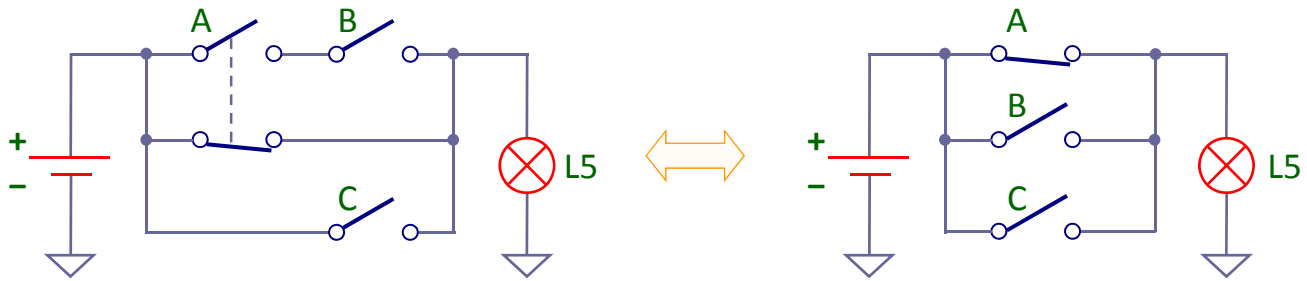
Os circuitos das lâmpadas L5 e L6 representam malhas de comutação constituídas por 3 interruptores A, B e C, que controlam o acender e apagar das referidas lâmpadas.

Determinar o comportamento do circuito é determinar as combinações de A, B e C que estabelecem os caminhos que levam a corrente eléctrica a percorrer as Lâmpadas e a acendê-las.

Os circuitos à direita estão simplificados. A simplificação é intuitiva, mas pode ser feita por métodos algébricos desenvolvidos adiante.

	C	B	A	L5
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Tabela funcional.

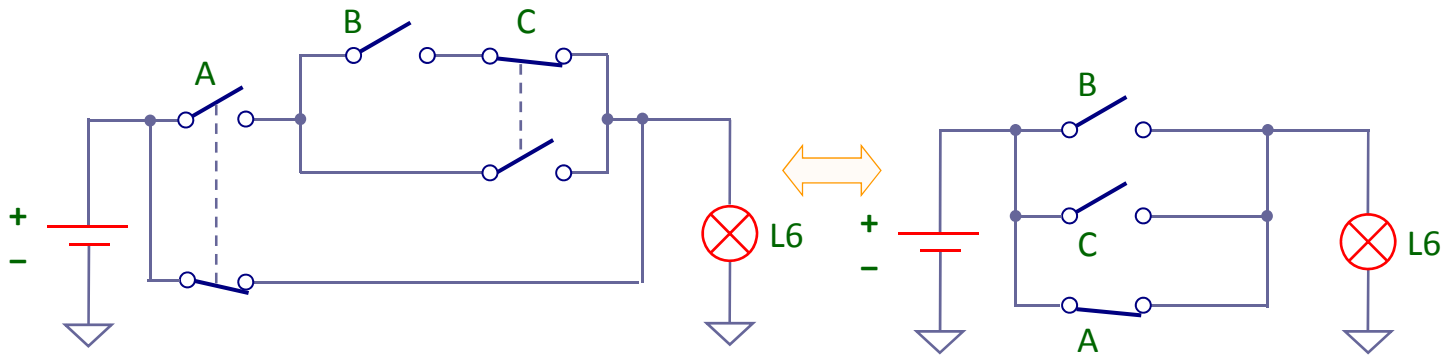


$L5 = A \cdot B + A' + C = A' + B + C$

Equação algébrica representativa das condições acendimento da lâmpada.

	C	B	A	L6
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Tabela funcional.

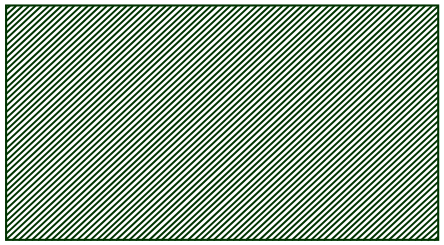


$L6 = A\,(\,BC' + C\,) + A' = A\,(\,B + C\,) + A' = B + C + A'$

Equação algébrica representativa das condições acendimento da lâmpada.

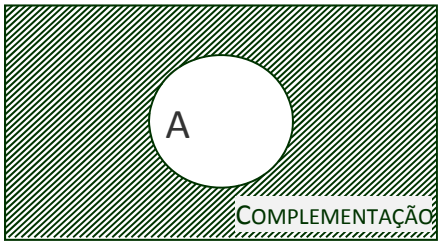
Diagramas de Venn: Visualização de Operações de Complementação, Intersecção, União e Exclusão Mútua

Conjunto vazio 0

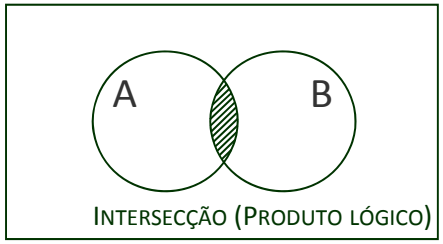


Conjunto universal 1

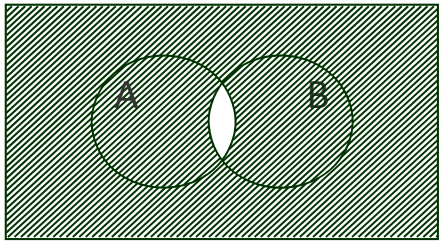
A



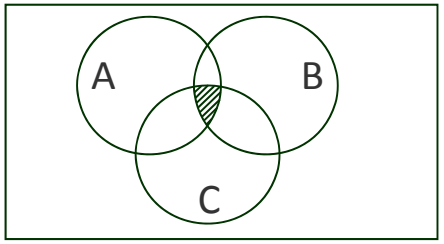
A' (Complementar de A)



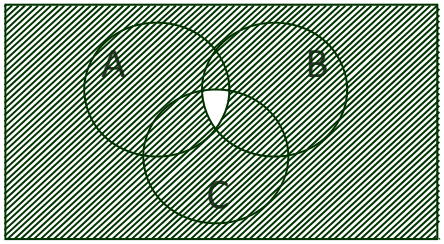
A · B



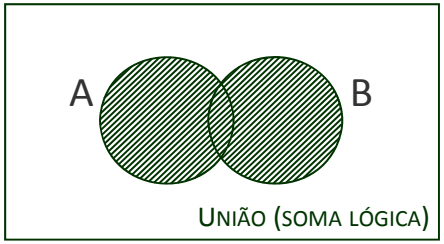
(A · B)' = A' + B'



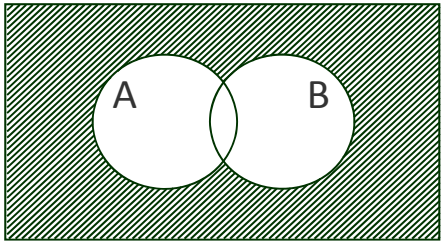
A · B · C



(A · B · C)' = A' + B' + C'

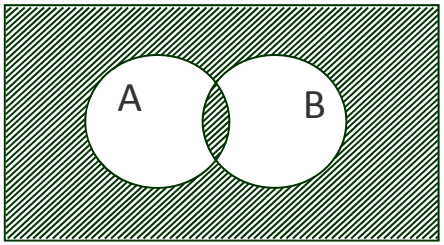


A + B

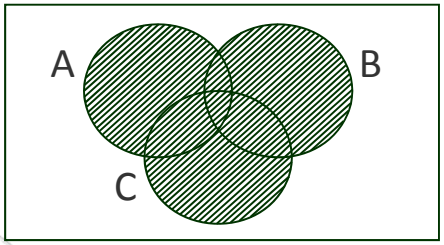


(A + B)' = A' · B'

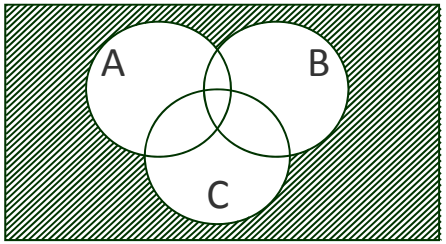
A ⊕ B



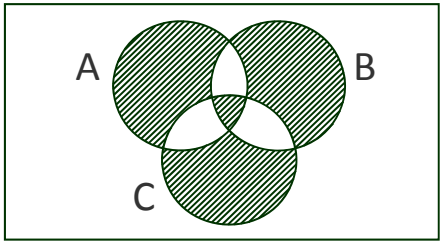
(A ⊕ B)'



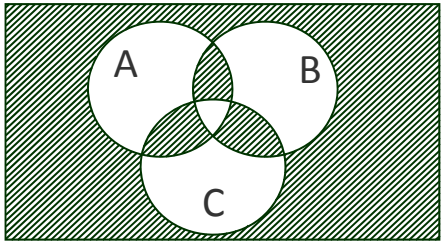
A + B + C



(A + B + C)'= A' · B' · C'

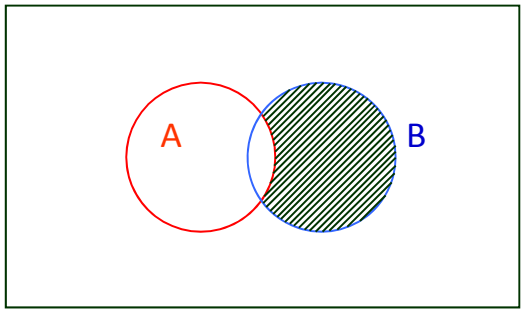


A ⊕ B ⊕ C



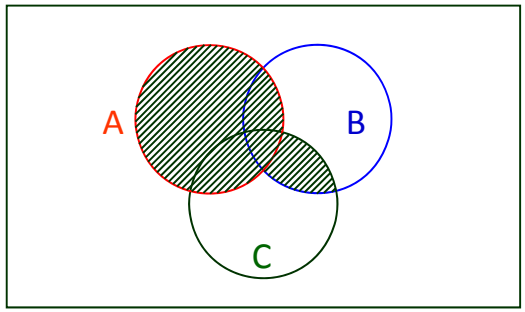
(A ⊕ B ⊕ C)'

Diagramas de Venn: Visualização de Operações de Complementação, Intersecção e União

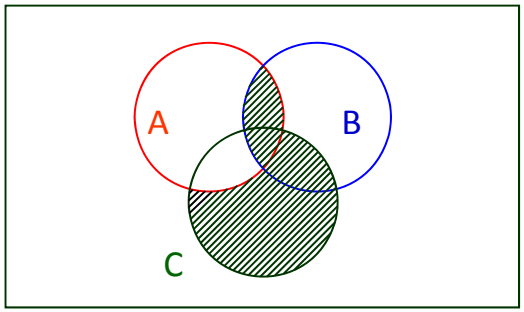


Diferença entre B e A ou complementar de A em B.

$A' \cdot B$



$A + B \cdot C$



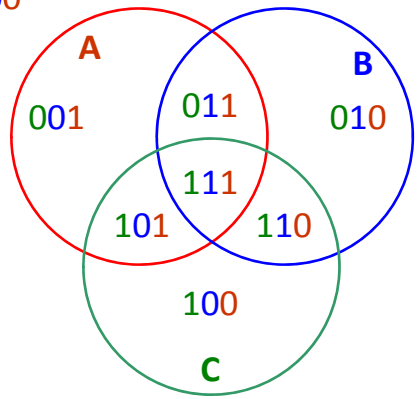
$A \cdot B + A' \cdot C$

Cada operação é indicada pelo escurecimento das áreas correspondentes às combinações de variáveis para as quais a operação é 1.

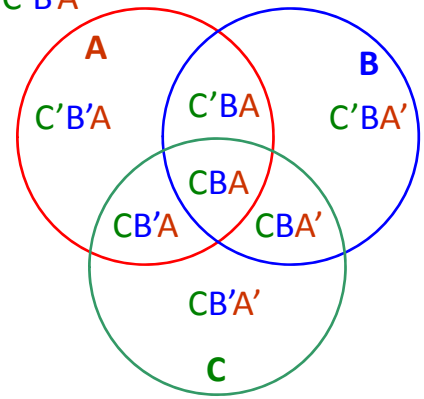
John Venn

1834-1923
(UK)

CBA=000

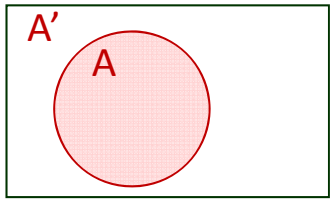


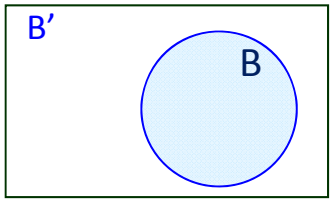
C'B'A'

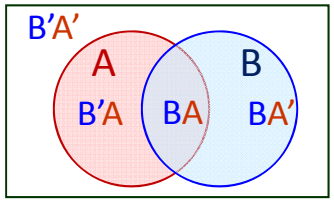


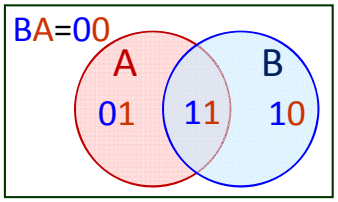
Atribuição de área a cada uma das 8 combinações de variáveis (C, B, A) para as quais a operação é 1.

Diagrama de Venn e Mapa de Karnaugh

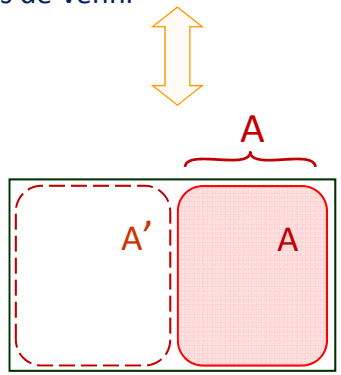


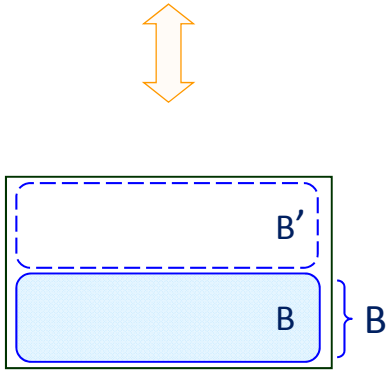


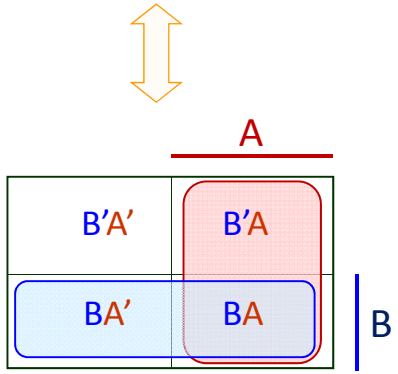


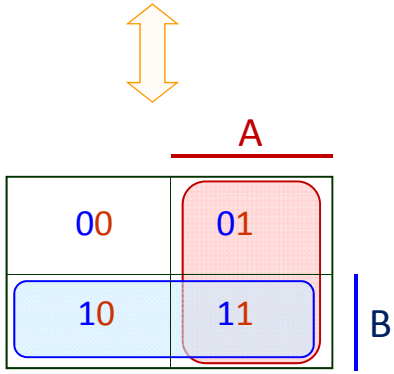


Diagramas de Venn.









Mapas de Karnaugh.

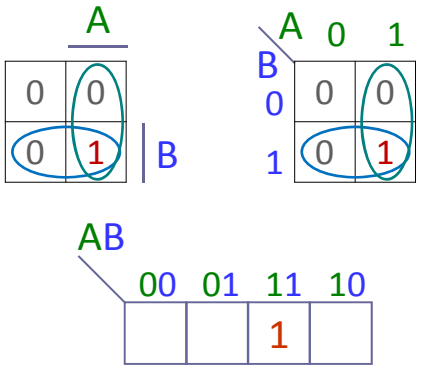
- Um **Mapa de Karnaugh** (ou Quadro de Karnaugh) pode ser visto como uma representação gráfica de uma tabela de verdade, ou como uma extensão de um diagrama de Venn.
- Cada rectângulo em cima representa o conjunto universal, que é neste caso o conjunto de todas as combinações de valores de 2 variáveis A e B.
- O rectângulo é dividido em 2^2 áreas iguais. Nesta notação, o rectângulo que representa o **AND** entre A e B, e que contém a inscrição 11 nas figuras do lado direito, é interpretado como a **intersecção de áreas**.
- O rectângulo que representa o **OR** entre A e B é interpretado como uma **união de áreas** – correspondente aos 3 rectângulos que não contêm a inscrição 00 na mesma figura.

Operações AND (Intersecção – Produto Lógico) e OR (União – Soma Lógica): Definição e Representações

Definição do **AND**: Operação sobre **n variáveis** que só toma o valor 1 quando todas as variáveis tiverem o valor 1.

Entradas		Saída
B	A	AB
0	0	0
0	1	0
1	0	0
1	1	1

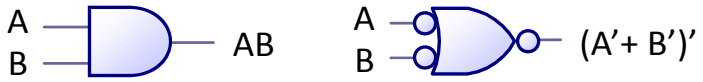
Tabela de Verdade do AND a 2 variáveis.



Mapas de Karnaugh do AND a 2 variáveis (grafismos equivalentes).

A **AND** B = A·B = AB = A ∧ B = A ∩ B

Expressões algébricas equivalentes (o símbolo '• ' do produto lógico pode ser omitido).



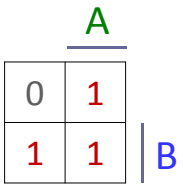
Símbolos lógicos equivalentes para uma porta AND de 2 entradas (o símbolo tradicional está à esquerda e o alternativo à direita).

Símbolo rectangular IEEE/IEC/ANSI.

Definição do **OR**: Operação sobre **n variáveis** que só toma o valor 0 quando todas as variáveis tiverem o valor 0.

Entradas		Saída
B	A	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Tabela de Verdade do OR a 2 variáveis.



Mapa de Karnaugh do OR a 2 variáveis.

A **OR** B = A + B = A ∨ B = A ∪ B

Expressões algébricas equivalentes.

Símbolos lógicos equivalentes para uma porta OR de 2 entradas (o símbolo tradicional está à esquerda e o alternativo à direita).

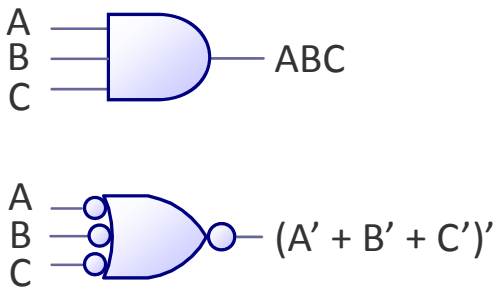
Símbolo rectangular IEEE/IEC/ANSI.

IEEE Institute of Electrical and Electronic Engineers.
IEC International Electrotechnical Comission.
ANSI American National Standards Institute.

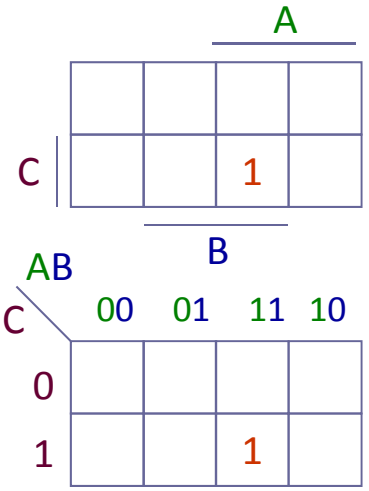
OPERAÇÕES AND E OR A 3 VARIÁVEIS

C	B	A	A B
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tabela de Verdade do AND a 3 variáveis.



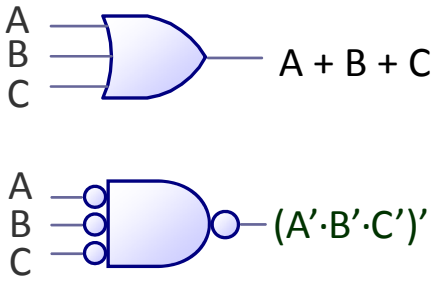
Símbolos lógicos equivalentes para uma porta AND de 3 entradas.



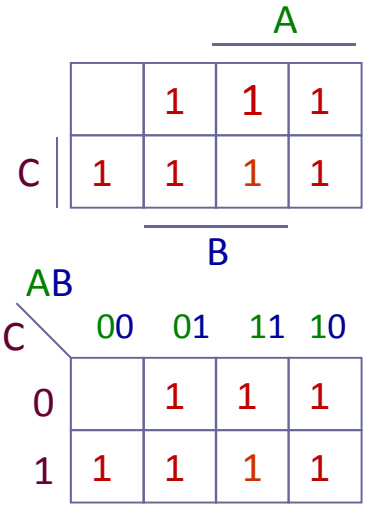
Mapa de Karnaugh do AND a 3 variáveis (dois grafismos equivalentes).

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

Tabela de Verdade do OR a 3 variáveis.



Símbolos lógicos equivalentes para uma porta OR de 3 entradas.



Mapa de Karnaugh do OR a 3 variáveis (dois grafismos equivalentes).

Operações NOT (Complementação ou Inversão Lógica), NAND e NOR: Definição e Representações

Definição do **NOT**: Operação sobre **1 variável** de que resulta a inversão do seu valor lógico.

Entrada	Saída
A	A'
0	1
1	0

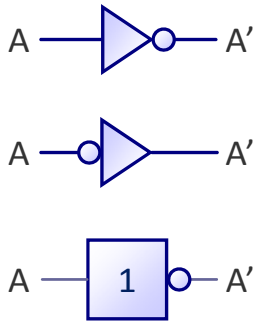
Tabela de Verdade do NOT.

A	
1	0

Mapa de Karnaugh do NOT.

NOT A = ¬ A = A' = $\overline{A}$

Expressões algébricas equivalentes (a plica é a notação usada nestes slides).



NOT: Símbolos lógicos equivalentes.

Definição do **NAND (NOT AND)**: Operação sobre **n variáveis** que só toma o valor 0 quando todas as variáveis tiverem o valor 1.

Entradas		Saída
B	A	(AB)'
0	0	1
0	1	1
1	0	1
1	1	0

Tabela de Verdade do NAND a 2 variáveis.

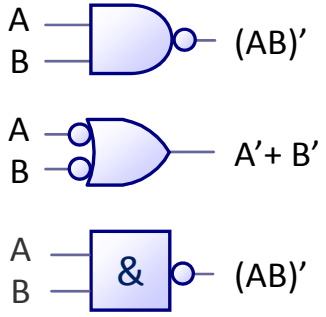
	A	
	1	1
B	1	0

Mapa de Karnaugh do NAND a 2 variáveis.

A **NAND** B = A↑ B =

= (A · B)' = A'+ B'

Expressões algébricas equivalentes.



NAND: Símbolos lógicos equivalentes (duas entradas).

Definição do **NOR (NOT OR)**: operação sobre **n variáveis** que só toma o valor 1 quando todas as variáveis tiverem o valor 0.

Entradas		Saída
B	A	(A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

Tabela de Verdade do NOR a 2 variáveis.

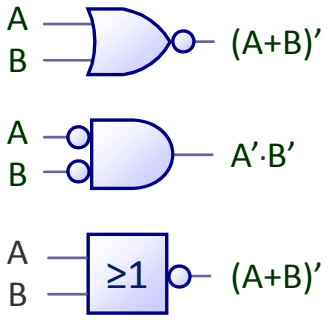
	A	
	1	0
B	0	0

Mapa de Karnaugh do NOR a 2 variáveis.

A **NOR** B = A ↓ B =

= (A + B)' = A' · B'

Expressões algébricas equivalentes.



NOR: Símbolos lógicos equivalentes (duas entradas).

CONTROLO DO FLUXO DE SINAL ATRAVÉS DE UMA PORTA AND

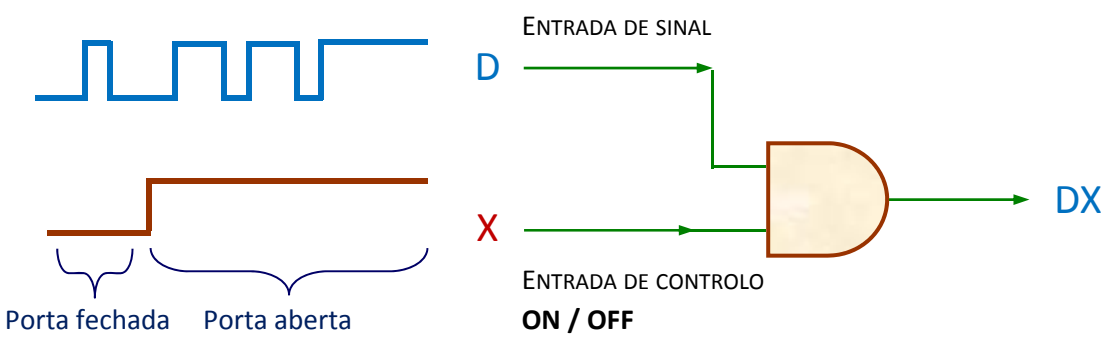


Diagrama lógico revelando a entrada de sinal, a entrada de controlo e a saída do circuito.

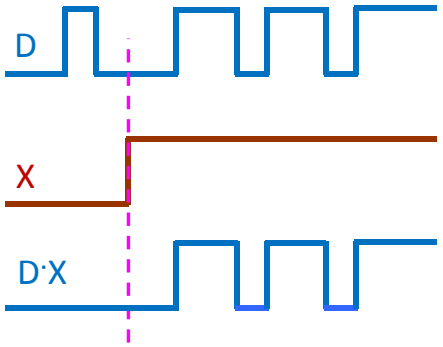


Diagrama temporal ilustrativo do funcionamento do circuito.

A designação de **porta** AND é apropriada no seguinte contexto: uma das entradas (**D**) é designada como **entrada de sinal**, a outra (**X**) como **entrada de controlo**.

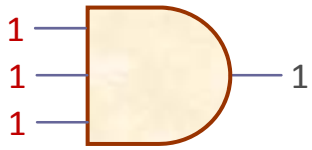
A porta AND está **aberta** para a entrada de sinal – transfere para a saída o sinal **D** presente nessa entrada – quando a entrada de controlo **X** estiver ao nível 1.

A porta AND está **fechada** para a entrada de sinal – bloqueia a passagem do sinal D para a saída – quando a entrada de controlo X estiver ao nível 0.

ENTRADAS		SAÍDA
D	X	D · X
d	0	0
d	1	d

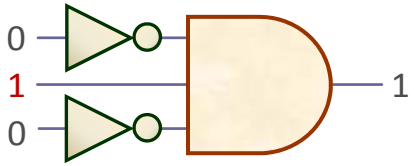
Tabela funcional descritiva do comportamento da porta AND.

Detecção de Padrões de Bits com uma Porta AND



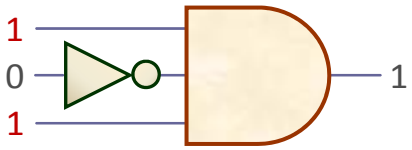
Detecção de: 111

Pode generalizar-se a operação de uma porta AND para n≥2 variáveis: nesta porta de três entradas só quando todas as variáveis estiverem a 1 a saída será 1 – em todos os outros casos a saída será 0.



Detecção de: 010

Porta AND com duas das três entradas negadas: só quando nas variáveis estiver presente a combinação 010 a saída será 1 – em todos os outros casos a saída será 0.



Detecção de: 101

Porta AND com uma das três entradas negadas: só quando nas variáveis estiver presente a combinação 101 a saída será 1 – em todos os outros casos a saída será 0.

AXIOMAS E TEOREMAS DA ÁLGEBRA DE BOOLE BINÁRIA

Uma **ÁLGEBRA DE BOOLE** BINÁRIA é um sistema algébrico $A_2 = (\bullet , +)$ formado por:

- um conjunto B = {0,1}
- a operações binária • (produto lógico AND)
- a operações binária **+** (soma lógica OR)
- a operação complemento (negação lógica NOT), tal que:

(I) $\forall\ X, Y \in B\ (X \bullet Y \in B) \wedge (X + Y \in B)$ (PROPRIEDADE DE FECHO)

(II) $\forall\ X, Y \in B$ verificam-se os axiomas seguintes:

o Propriedade Comutativa:	$X + Y = Y + X$	$X \cdot Y = Y \cdot X$
o Propriedade Associativa:	$(X + Y) + Z = X + (Y + Z)$	$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
o Propriedade Distributiva:	$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$	$X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$
o Elemento neutro:	$X + 0 = X$	$X \cdot 1 = X$
o Complemento:	$X + X' = 1$	$X \cdot X' = 0$

(a pelica denota a operação complemento).

Os primeiros axiomas são:

$X = 0$ se $X \neq 1$	$X = 1$ se $X \neq 0$	se $X = 0$, $X' = 1$
$0 . 0 = 0$	$1 + 1 = 1$	
$1 . 1 = 1$	$0 + 0 = 0$	
$0 . 1 = 1 . 0 = 0$	$1 + 0 = 0 + 1 = 1$	

Estes axiomas são apresentados em 'par', o que é típico de todos os outros axiomas desta Álgebra, e constitui a base do **PRINCIPIO DE DUALIDADE**.

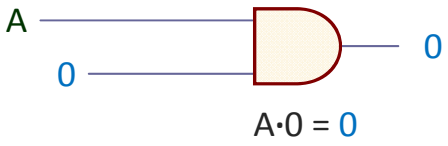
Os axiomas ou postulados de um sistema matemático são um conjunto mínimo de definições básicas que se assumem serem verdadeiras, e das quais todas as outras informações sobre o sistema podem ser obtidas.

Axiomas e Teoremas da Álgebra de Boole Binária

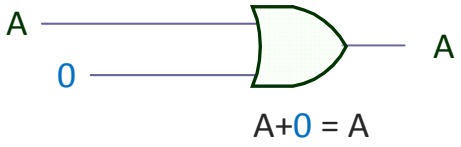
	Teorema	Forma Básica	Forma Dual	
1	Identidade (Elemento Neutro)	A + 0 = A	A · 1 = A	1 Variável
2	Elementos Absorventes	A + 1 = 1	A · 0 = 0	
3	Idempotência	A + A = A	A · A = A	
4	Involução	(A')' = A		
5	Complementaridade	A + A' = 1	A · A' = 0	
6	Comutatividade	A + B = B + A	A · B = B · A	2 ou mais Variáveis
7	Associatividade	(A + B) + C = A + (B + C)	(A·B) · C = A · (B·C)	
8	Distributividade	A·B + A·C = A (B + C)	(A + B)·(A + C) = A + (B·C)	
9	Absorção	A + A·B = A	A (A + B) = A	
10	Absorção 2 (Redundância)	A + A'·B = A + B	A (A'+ B) = AB	
11	Adjacência	A·B + A·B' = A	(A + B)(A + B') = A	
12	Leis de De Morgan	(A + B)' = A' · B'	(A · B)' = A' + B'	

Teoremas da Álgebra de Boole a 1 Variável

Elemento Absorvente



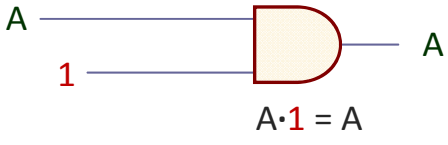
A·0 = 0



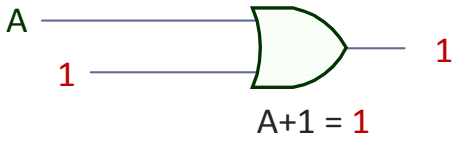
A+0 = A

Identidade

Identidade



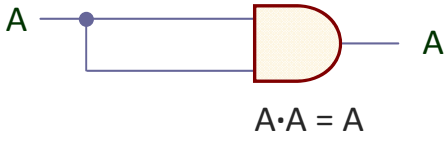
A·1 = A



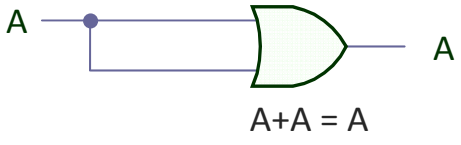
A+1 = 1

Elemento Absorvente

Idempotência



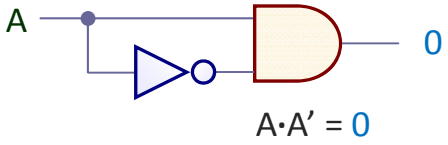
A·A = A



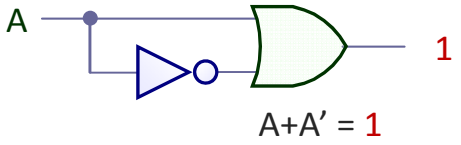
A+A = A

Idempotência

Complementaridade



A·A' = 0

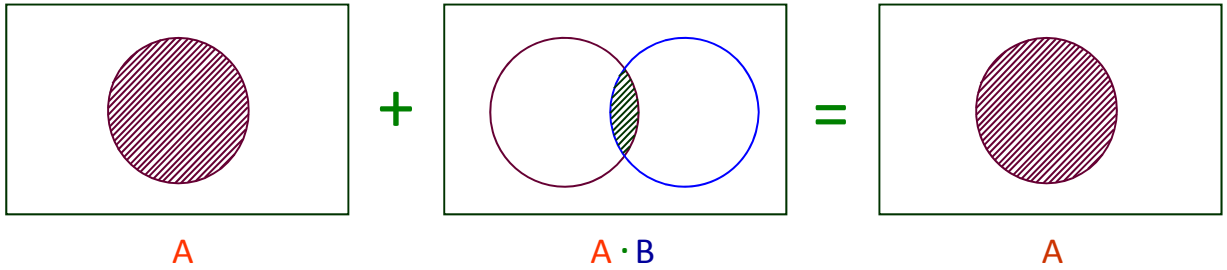


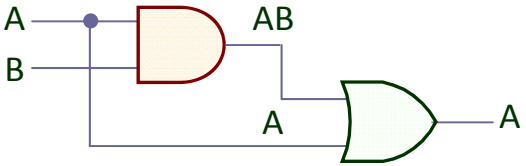
A+A' = 1

Complementaridade

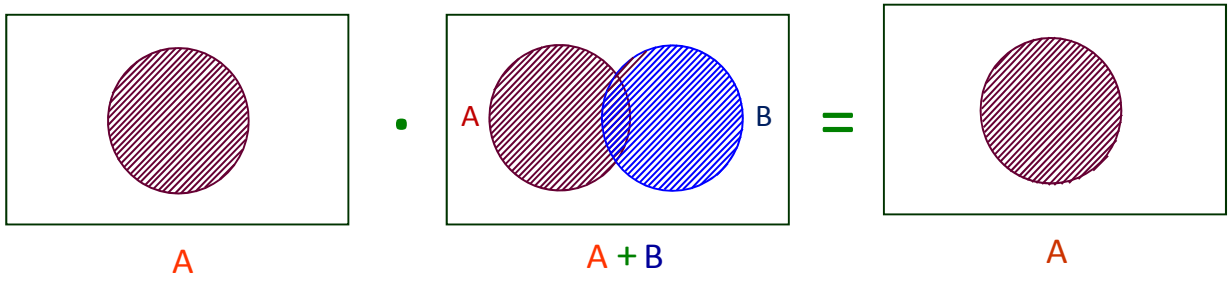
Visualização em diagrama lógico de alguns teoremas da Álgebra de Boole a uma variável.

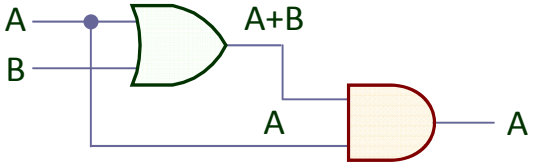
Teoremas da Absorção e Redundância: Visualização em Diagrama de Venn



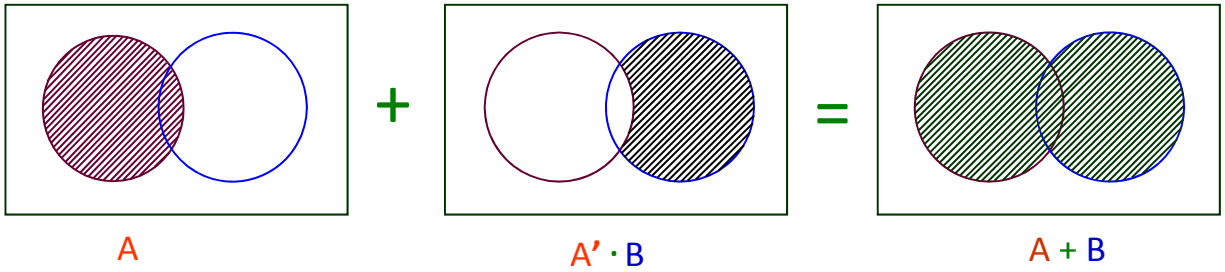


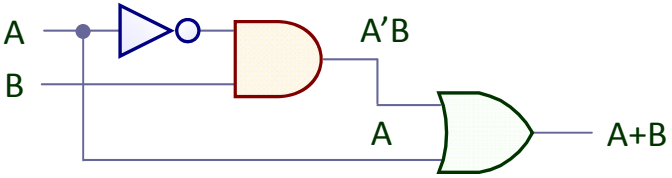
$A + AB = A\,(1 + B) = A \cdot 1 = A$





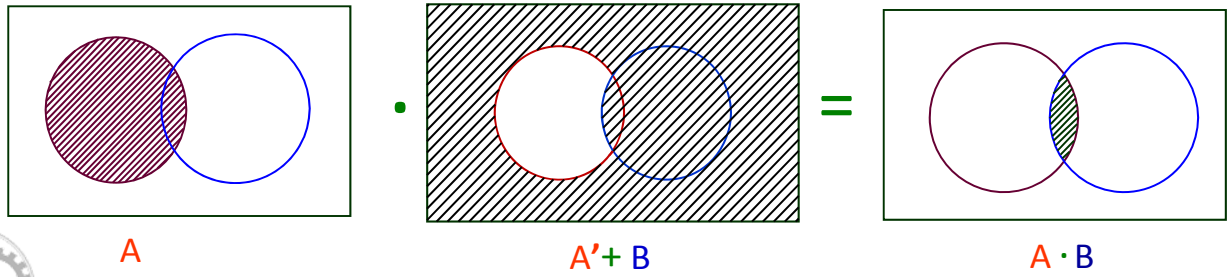
$A\,(A+B) = AA + AB = A$

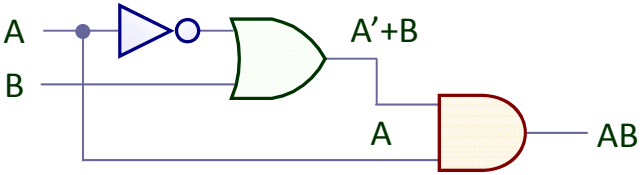




$A + A'B = (A + AB) + A'B = A + AB + A'B =$

$= A + B\,(A + A') = A + B$





$(A' + B)\,A = AB$

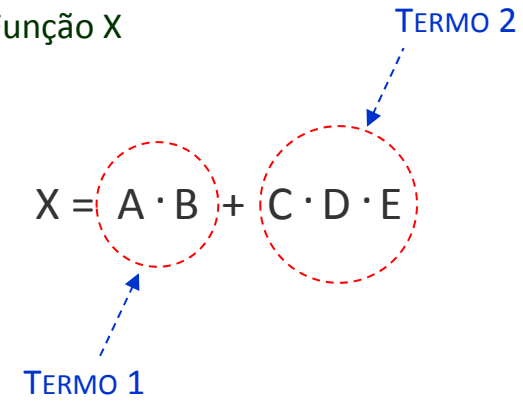
Simplificação Algébrica de Funções

Exemplos

1. $F = (A' + B' + C')' = A \cdot B \cdot C$
2. $F = (A' \cdot B' \cdot C')' = A + B + C$
3. $F = B' + A \cdot B \cdot C' = B' + A \cdot C'$
4. $F = (A \cdot B)' + A \cdot B \cdot C' = (A \cdot B)' + C' = A' + B' + C'$
5. $F = (A + A \cdot B) (C + C' \cdot D) + A' \cdot D = A (C + D) + A' D = A \cdot C + A \cdot D + A' \cdot D = A C + D$
6. $F = (((A \cdot B)' + C)' D')' = ((A \cdot B)' + C) + D = A' + B' + C + D$
7. $F = (A + A' \cdot B) (A + A \cdot B) (B + B') (C' + B \cdot C') = (A + B) \cdot A \cdot 1 \cdot C' = A C'$
8. $F = (((A \cdot B)' + C)' (A \cdot C')' + A' \cdot B)' = ((A \cdot B \cdot C') (A' + C) + A' \cdot B)' = (A' \cdot B)' = A + B'$
9. $F = (A + B + C) (A + B' + C) = A + C$
10. $F = A B + A' C + B C = A B + A' C + B C (A + A') = A B (1 + C) + A' C (1 + B) = A B + A' C$

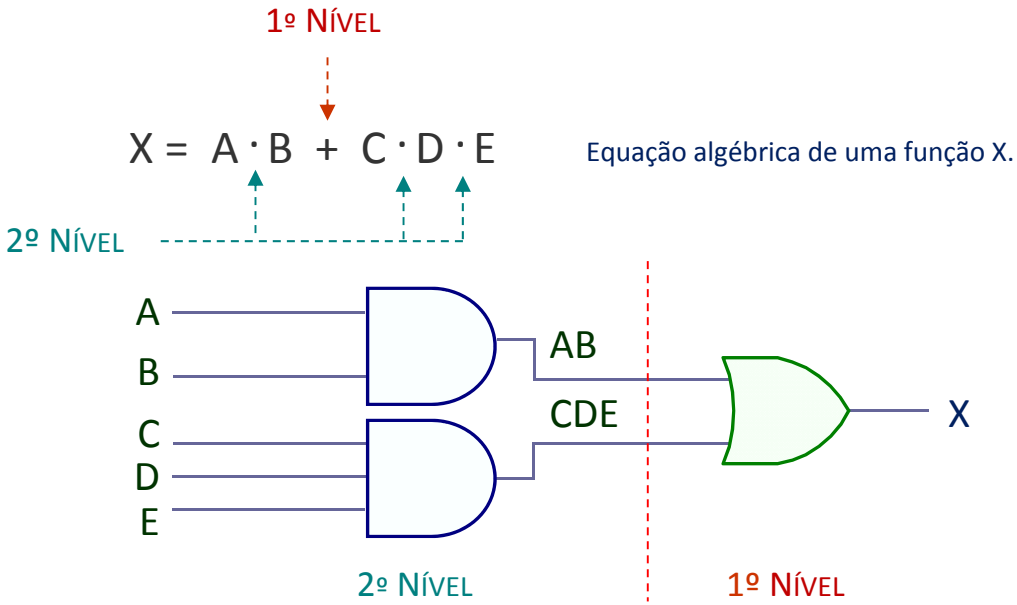
Implementação do Circuito a partir da Expressão Booleana

Exemplo de uma Função X



AB e CDE são os **Termos** da função.

A, B, C, D e E são as **Variáveis** (ou os **Literais**).



Equação algébrica de uma função X.

Diagrama lógico do circuito correspondente assinalando-se os níveis.

Em expressões que envolvam vários produtos e somas lógicas o **Produto** tem **precedência** sobre a **Soma**:

AB + CDE deve ser entendido como a realização primeiro dos produto lógicos de A por B e de C por D por E , e só em segundo lugar a soma lógica desses produtos.

De um modo geral, nas expressões lógicas envolvendo as operações de AND, OR, NOT e XOR (a operação XOR com o símbolo ⊕ é apresentada adiante), estabelece-se por convenção a seguinte hierarquia de operações: NOT, AND, XOR e OR. Esta ordem de precedência indica que o NOT é a primeira operação a ser executada, e a seguir, as operações de AND e XOR e OR na ordem em quem estiverem dispostas como se indica ao lado. O uso de parênteses altera a ordem normal dos operandos.

$X + Y\,Z = X + (Y\,Z)$

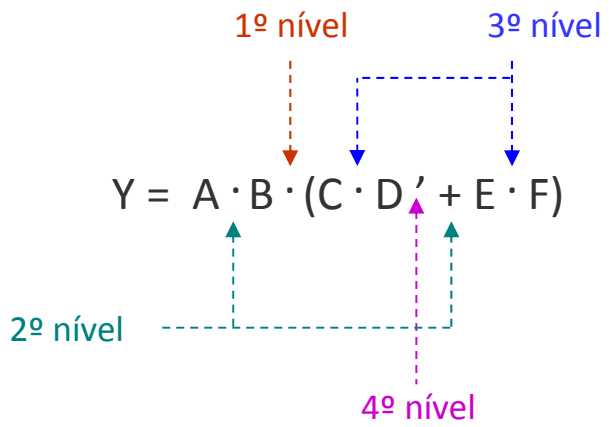
$X \oplus Y\,Z = X \oplus (Y\,Z)$

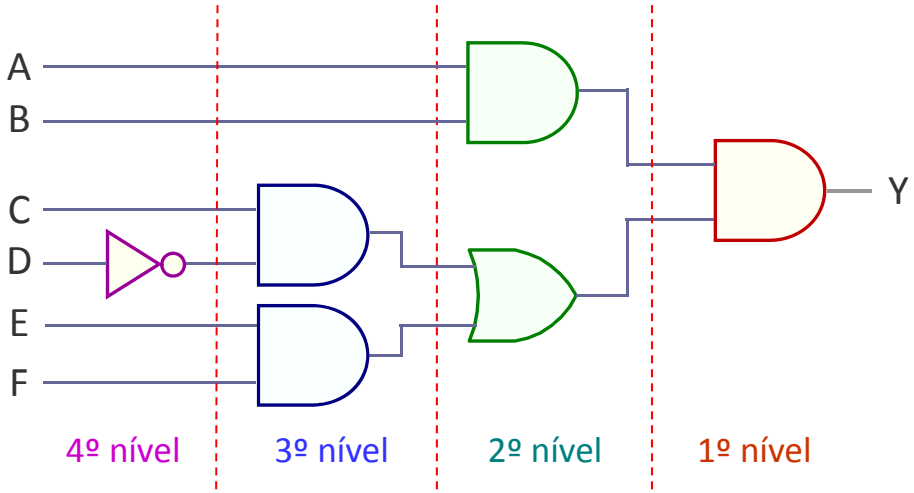
$X + Y \oplus Z = X + (Y \oplus Z)$

Exemplos de expressões lógicas mostrando a hierarquia de operações.

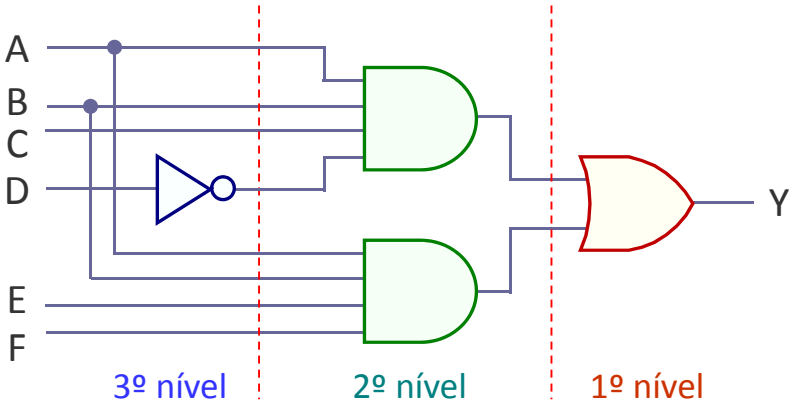
Implementação do Circuito a partir da Expressão Booleana

Exemplo de
uma Função Y





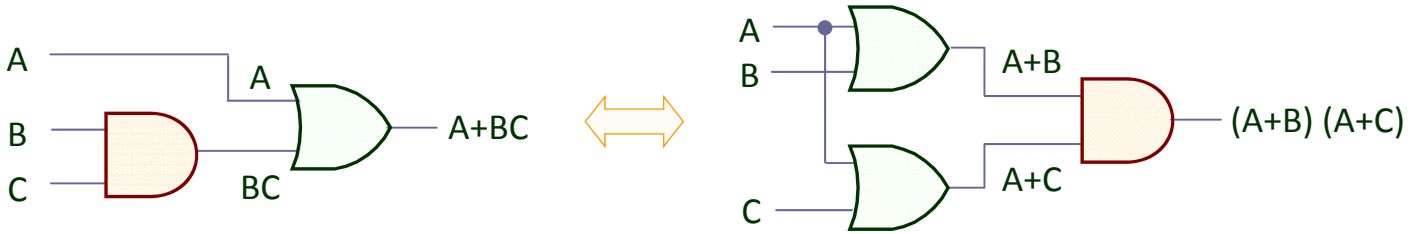
Implementação **multinível** de uma função Y.

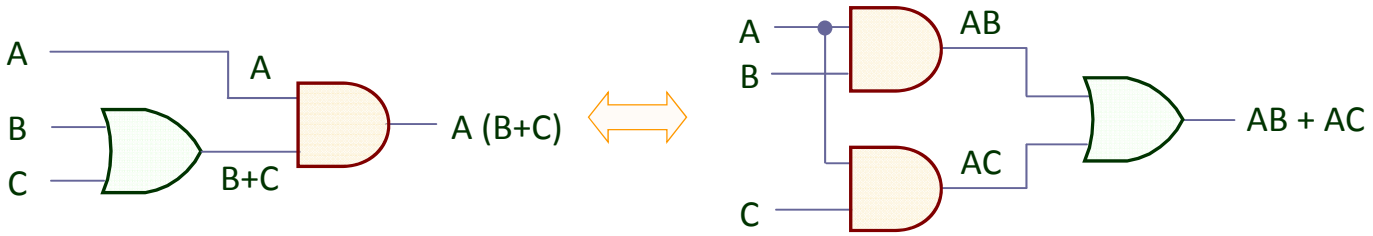


Implementação equivalente a **3 níveis**
da função Y na forma de soma de produtos.

PRINCÍPIO DAS DUALIDADE

A expressão dual da expressão 1·X é a expressão 0+X.





A equação em cima A + (BC) = (A+B)(A+C) tem uma dual em baixo que é A (B+C) = AB + AC.

A + BC = (A+B) (A+C) ⟺ A (B + C) = AB + AC

DUAL

PRINCÍPIO DA DUALIDADE

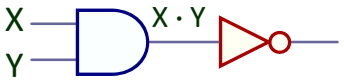
Qualquer expressão válida numa Álgebra de Boole possui uma **expressão dual**, também válida nessa álgebra, que se obtém:

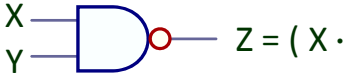
- trocando cada símbolo operatório **+** pelo símbolo operatório • e vice-versa,
- e ainda os 0s por 1s e os 1s por 0s.

Antes de se determinar a dual de uma expressão é importante manter o uso de parêntesis que assegurem as regras de precedência dos operadores.

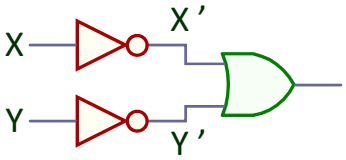
LEIS DE MORGAN

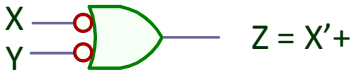
$(X \cdot Y)' = X' + Y'$





X	Y	X · Y	(X ·Y)'
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0





X	Y	X'	Y'	X' + Y'
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

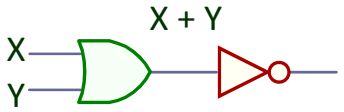
Verificação por Tabelas de Verdade.

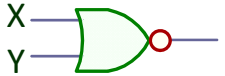
AUGUSTUS DE MORGAN

1806-1871
(India – UK)

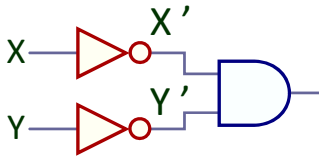
As LEIS DE MORGEN permitem transformar uma soma de produtos num produto de somas e vice-versa.

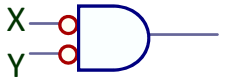
$(X + Y)' = X' \cdot Y'$





X	Y	X + Y	(X+Y)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0





X	Y	X'	Y'	X' · Y'
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

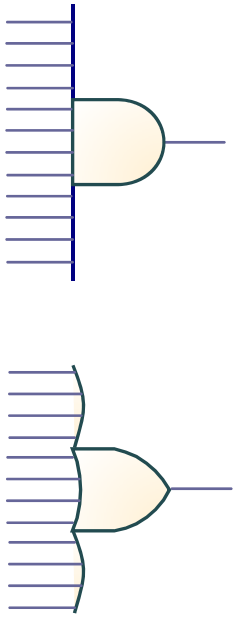
Verificação por Tabelas de Verdade.

$$(X_1 \cdot X_2 \cdots X_n)' = X_1' + X_2' + \cdots + X_n'$$

$$(X_1 + X_2 + \cdots + X_n)' = X_1' \cdot X_2' \cdots X_n'$$

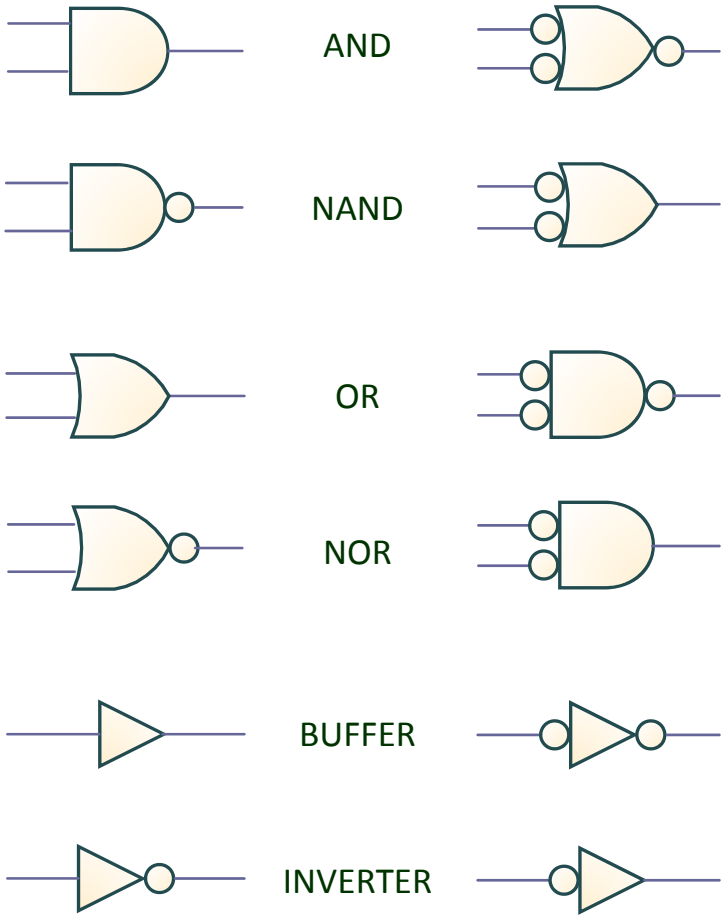
Generalização das Leis de Morgan para n variáveis.

Símbolos de Portas Lógicas



Expansão das entradas em portas AND e OR.

Os símbolos, tais como as tabelas de verdade para AND, OR, NAND e NOR, podem estender-se a portas com qualquer número de entradas.



Portas lógicas básicas e variantes fazendo uso de símbolos inversores mostrando-se a equivalência de símbolos entre as colunas da esquerda e da direita.

Usando as Leis de Morgan, podem manipular-se as expressões lógicas de portas, o que permite usar para uma porta **dois símbolos diferentes**, e ambos correctos (como apresentado em slides anteriores).

Embora os dois símbolos para cada uma das portas ao lado representem a mesma função lógica, a selecção de um ou de outro símbolo não deve ser arbitrária, mas sim obedecer aos padrões da boa documentação que torna os diagramas lógicos mais fáceis de utilizar e de entender.

UNIVERSALIDADE DA PORTA LÓGICA NAND

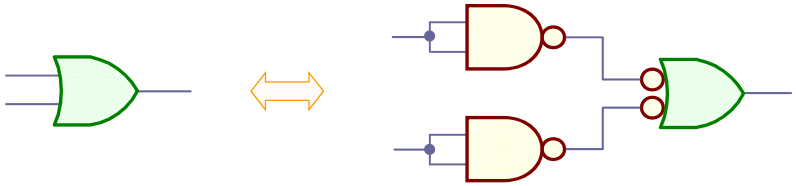
NOT

Realização de um NOT através de um NAND com as entradas em curto-circuito.

AND

Realização de um AND através de dois NAND em série.

OR



Realização de um OR através de três NAND.

A operação NOT é normalmente considerada, em sentido lato, como uma NAND de 1 entrada.

Nalgumas tecnologias (p. ex. TTL) as portas NAND são as portas mais simples (portanto menos onerosas em termos de custos de produção), pelo que é vantajosa a realização de circuitos só com NAND.

Qualquer função lógica pode ser representada por uma expressão utilizando apenas funções do tipo AND, OR e NOT.

Se for possível representar qualquer destas 3 funções AND, OR e NOT apenas por NAND, fica provado que qualquer função se pode representar usando apenas funções NAND.

A porta NAND é então considerada uma porta UNIVERSAL (ou **funcionalmente completa**) porque qualquer circuito digital pode ser realizado apenas com portas NAND por substituição directa das operações NOT, AND e OR.

VERIFICAÇÃO DA 'FUNCIONALIDADE COMPLETA' DE UM BLOCO FUNCIONAL

OBJECTIVO

Para a função de minoria F_m definida pela tabela abaixo, verificar que é funcionalmente completa.

ORDENAÇÃO	C	B	A	F_m
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Tabela de verdade da função Fm minoria a 3 variáveis.

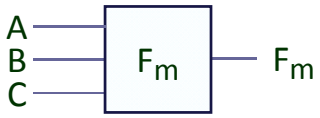
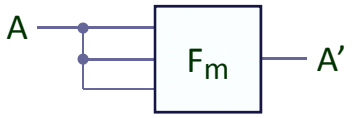


Diagrama de blocos da função Fm minoria a 3 variáveis.

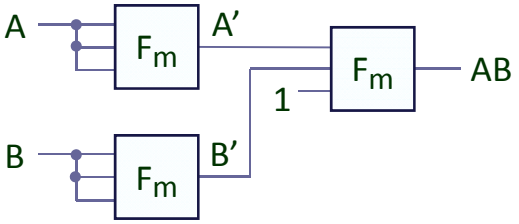
A função F_m minoria de 3 variáveis revela-se, tal como a porta NAND, universal, porque qualquer circuito digital pode ser realizado apenas na base de circuitos F_m como se verifica.

NOT

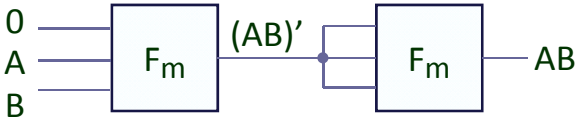


Realização de um NOT com um bloco Fm (as três entradas só podem ser simultaneamente 0 ou 1).

AND

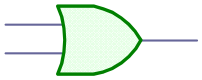


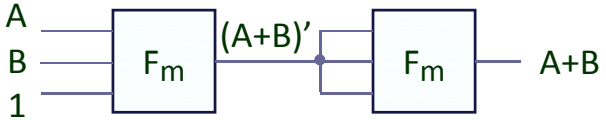
Realização de um AND com três blocos Fm.



Realização de um AND com dois blocos Fm.

OR

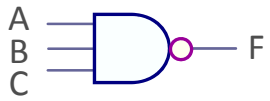


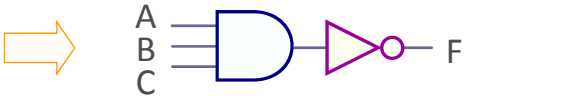


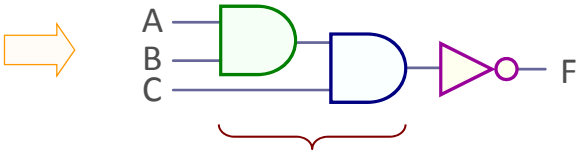
Realização de um OR com dois blocos Fm.

Síntese de portas NAND e NOR de 3-entradas a partir de portas de 2-entradas

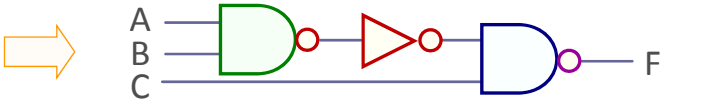
NAND

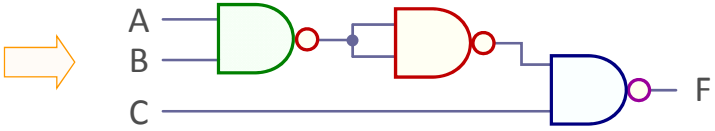






A operação **AND** é Associativa

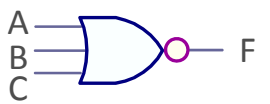


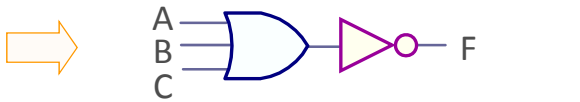


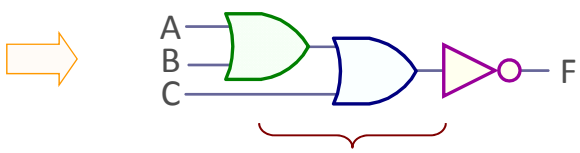
$$(A\ B\ C)' = (\ (\ (A\ B)'\ )'\cdot C)'$$

A operação **NAND <u>não é</u> Associativa** – mas por inversões parciais é sempre possível converter o NAND de n variáveis em NANDs de 2 variáveis.

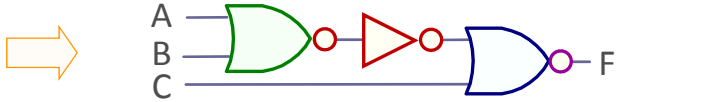
NOR

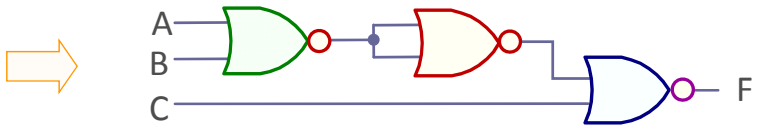






A operação **OR** é Associativa



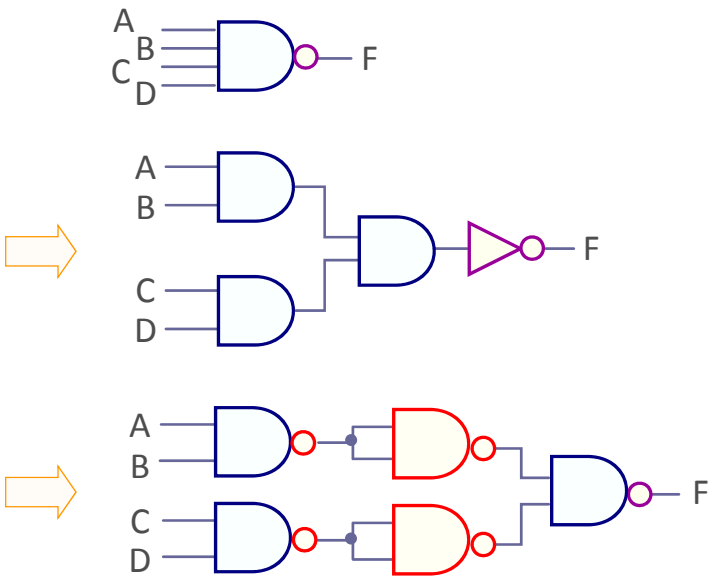


$$F = (A + B + C)' = (\ (\ (A + B)'\ )' + C)'$$

A operação **NOR <u>não é</u> Associativa** – mas por inversões parciais é sempre possível converter o NOR de n variáveis em NORs de 2 variáveis.

Síntese de portas NAND de 4-entradas a partir de portas NAND de 2-entradas

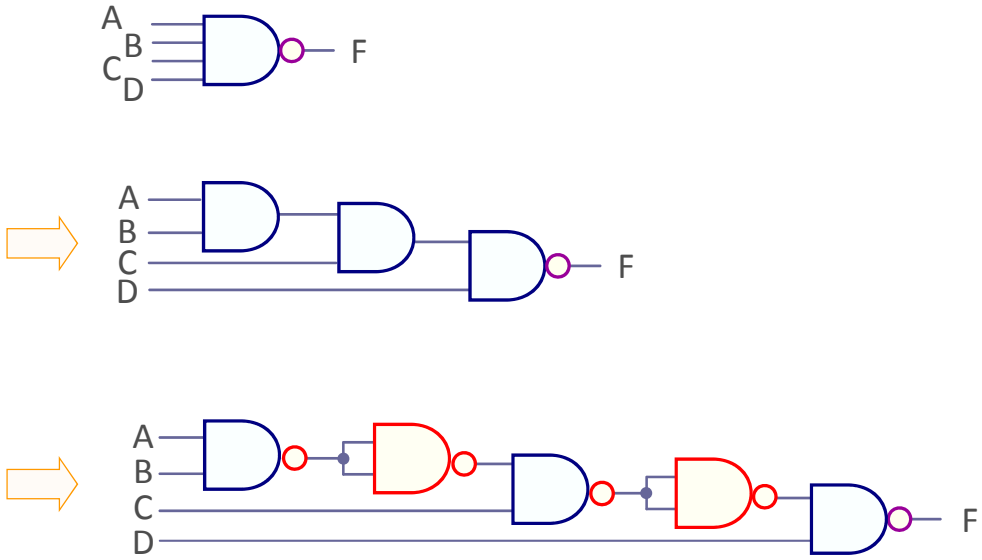
Em Paralelo



$$(A\ B\ C\ D)' = ((AB)\ (CD))' =$$

$$= (((AB)')'\ ((CD)')')'$$

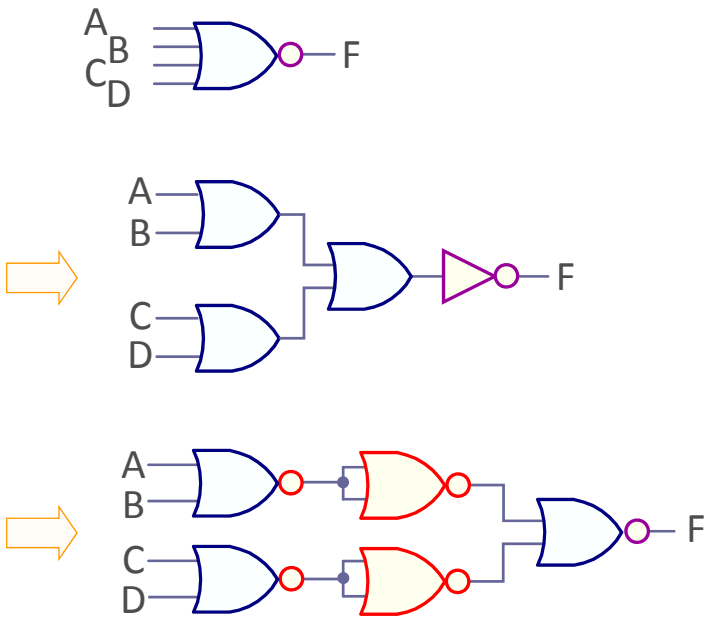
Em Série



$$(A\ B\ C\ D)' = (\ ((\ ((AB)')'\ C)')'\ D)'$$

Síntese de portas NOR de 4-entradas a partir de portas NAND de 2-entradas

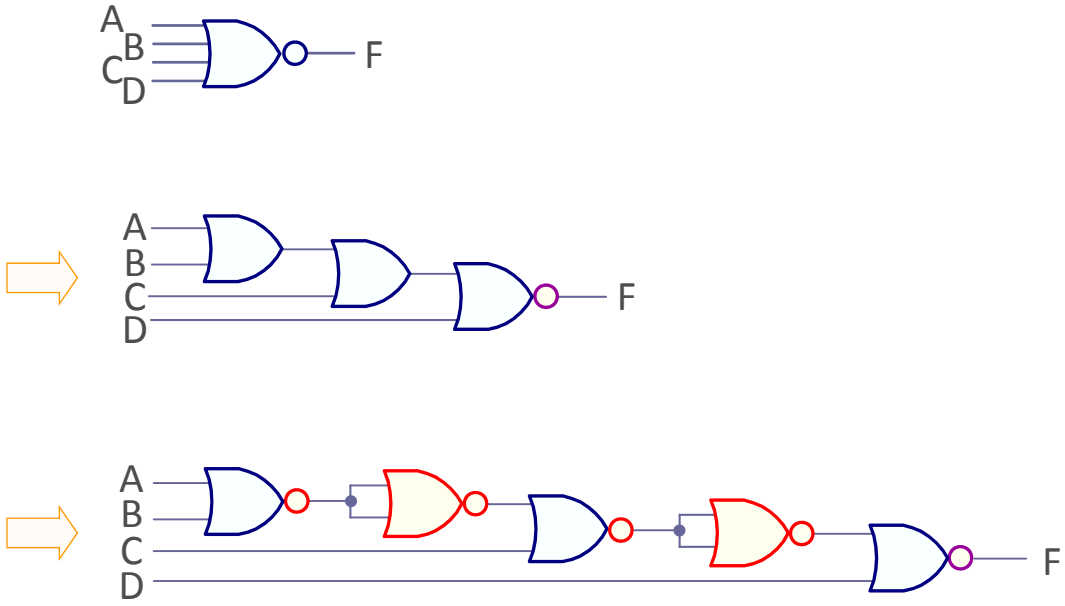
Em Paralelo



(A+B+C+D)' = ((A+B) + (C+D))' =

= (((A+B)')'+ ((C+D)')')'

Em Série

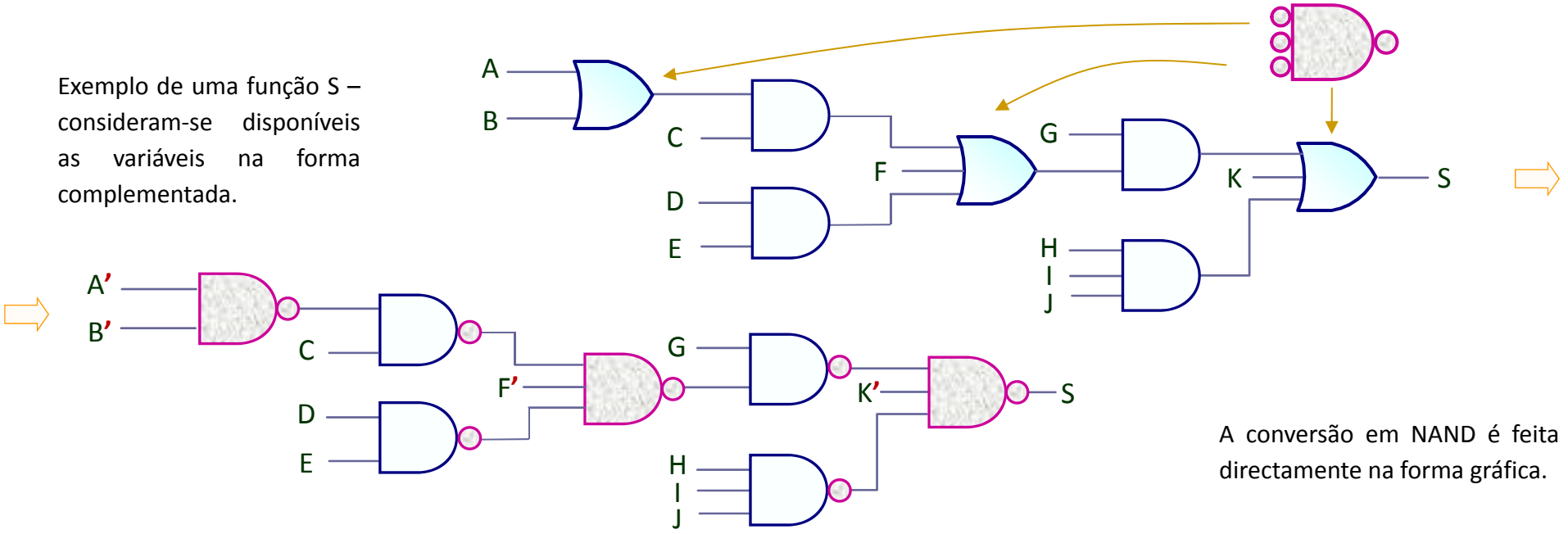


(A+B+C+D)' = (((A+B) + C) + D)' =

= (((((A+B)')'+ C)')'+ D)'

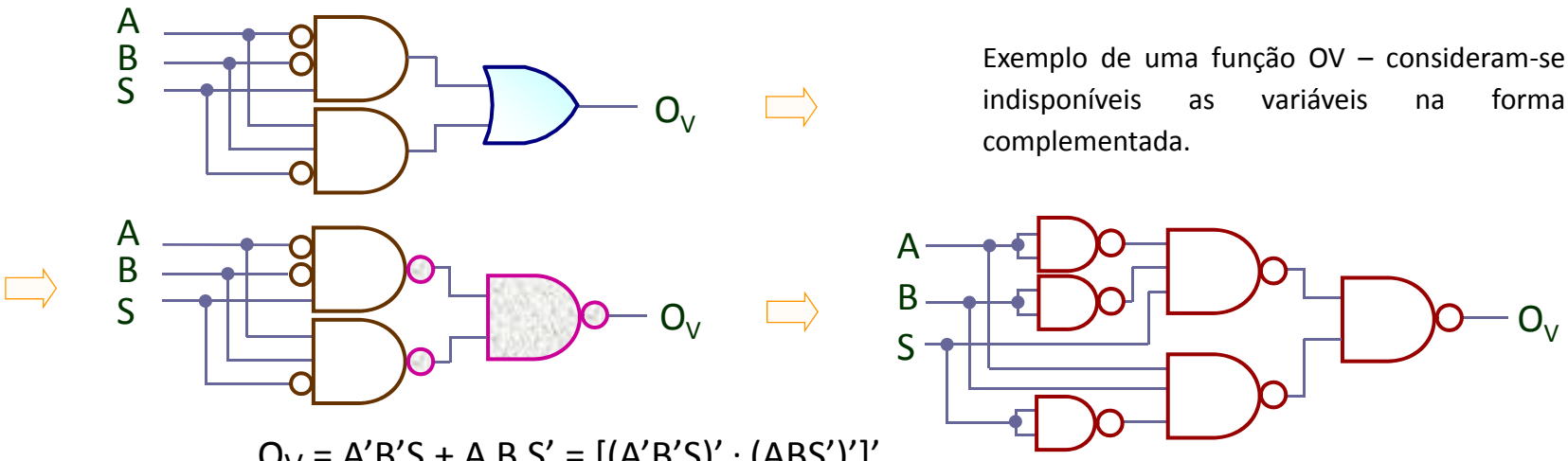
IMPLEMENTAÇÃO DE FUNÇÕES SÓ COM PORTAS NAND

Exemplo de uma função S – consideram-se disponíveis as variáveis na forma complementada.



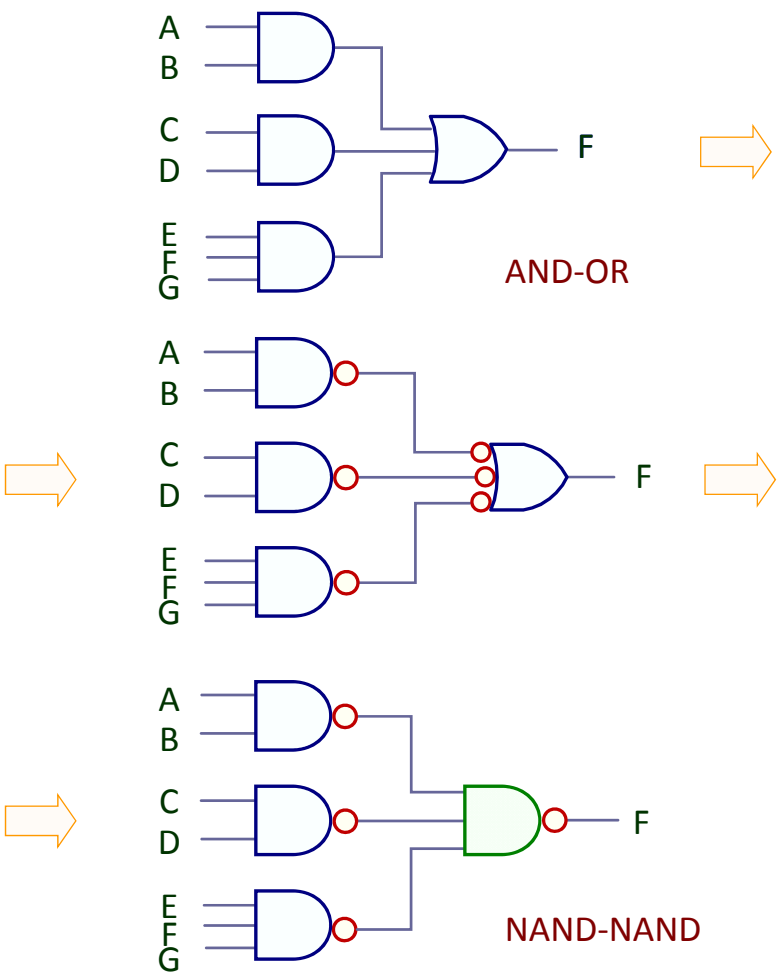
A conversão em NAND é feita directamente na forma gráfica.

Exemplo de uma função OV – consideram-se indisponíveis as variáveis na forma complementada.

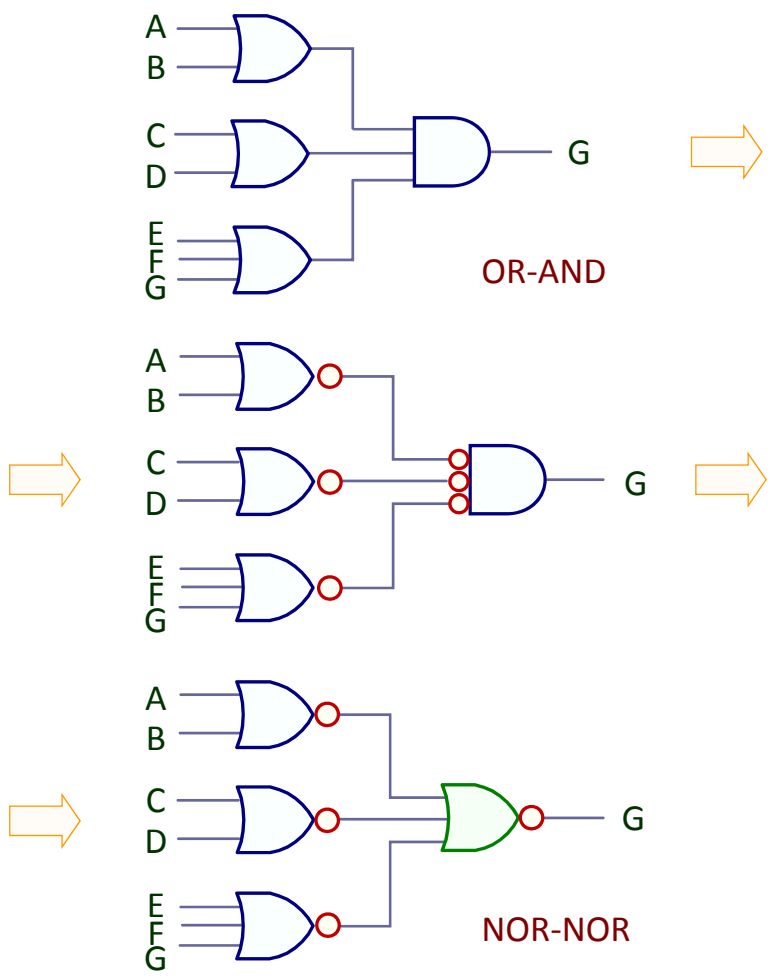


$O_V = A'B'S + A\,B\,S' = [(A'B'S)' \cdot (ABS')']'$

REDUÇÃO DE AND-OR A NAND E OR-AND A NOR



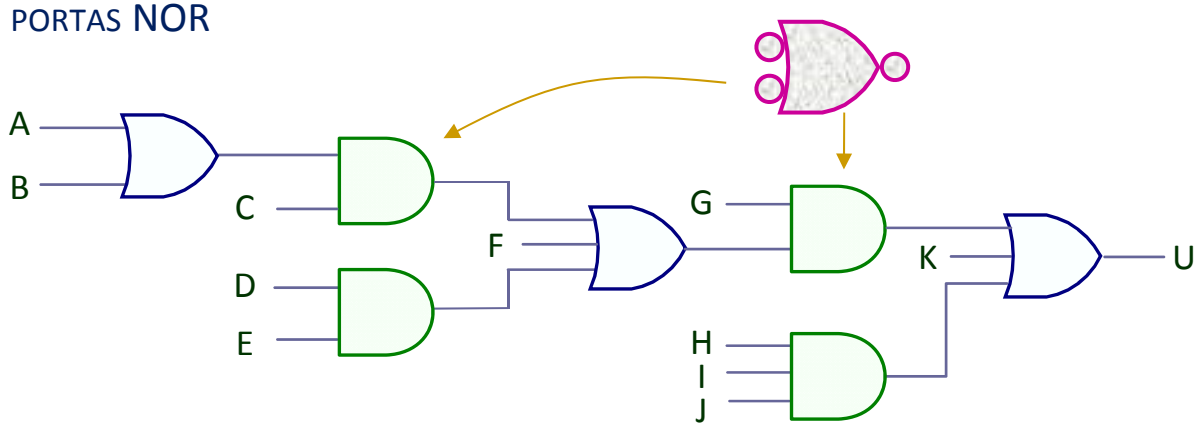
Uma função representada na forma AND-OR (soma de produtos), pode ser transformada numa forma directamente realizável apenas com portas NAND por simples aplicação da lei de De Morgan.

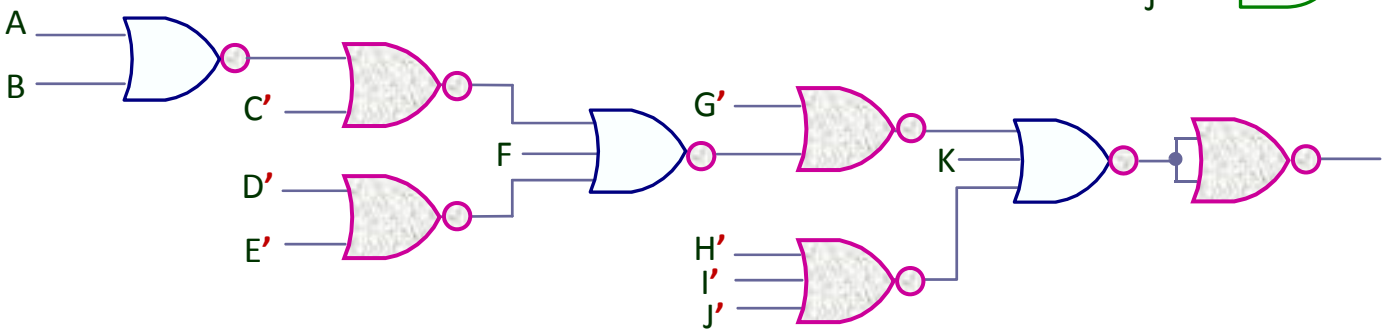


A implementação em NOR é imediata a partir de expressões na forma OR-AND.

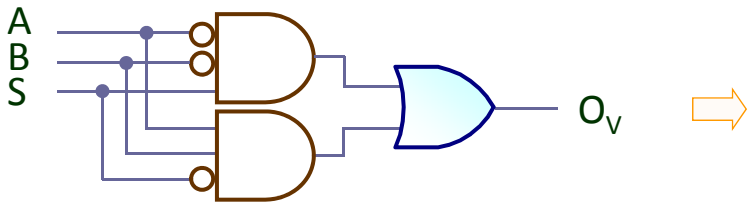
IMPLEMENTAÇÃO DE FUNÇÕES SÓ COM PORTAS NOR

Exemplo de uma função U – consideram-se disponíveis as variáveis na forma complementada.

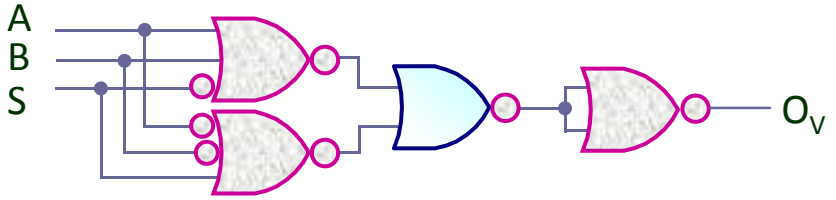


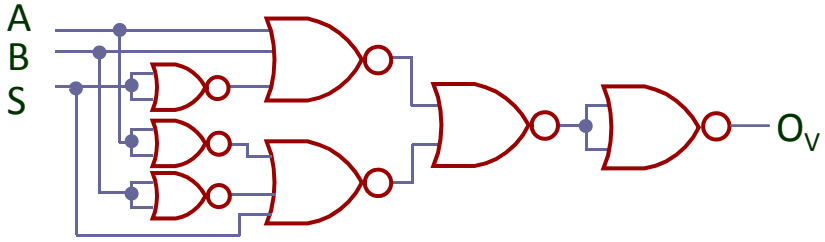


A conversão em NOR é feita directamente na forma gráfica.

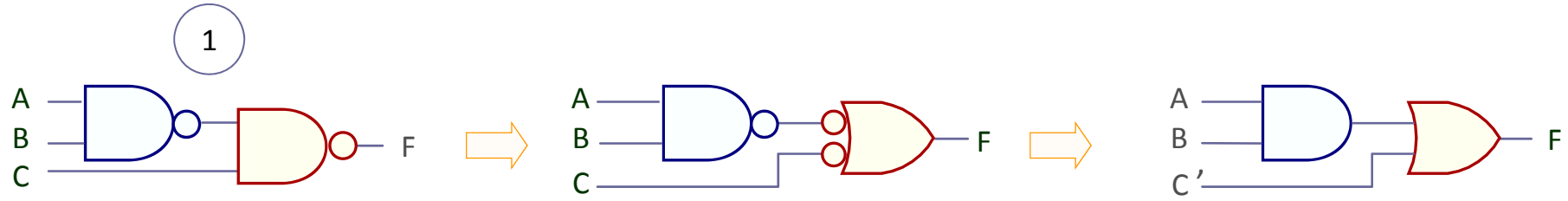


Exemplo de uma função OV – consideram-se indisponíveis as variáveis na forma complementada.

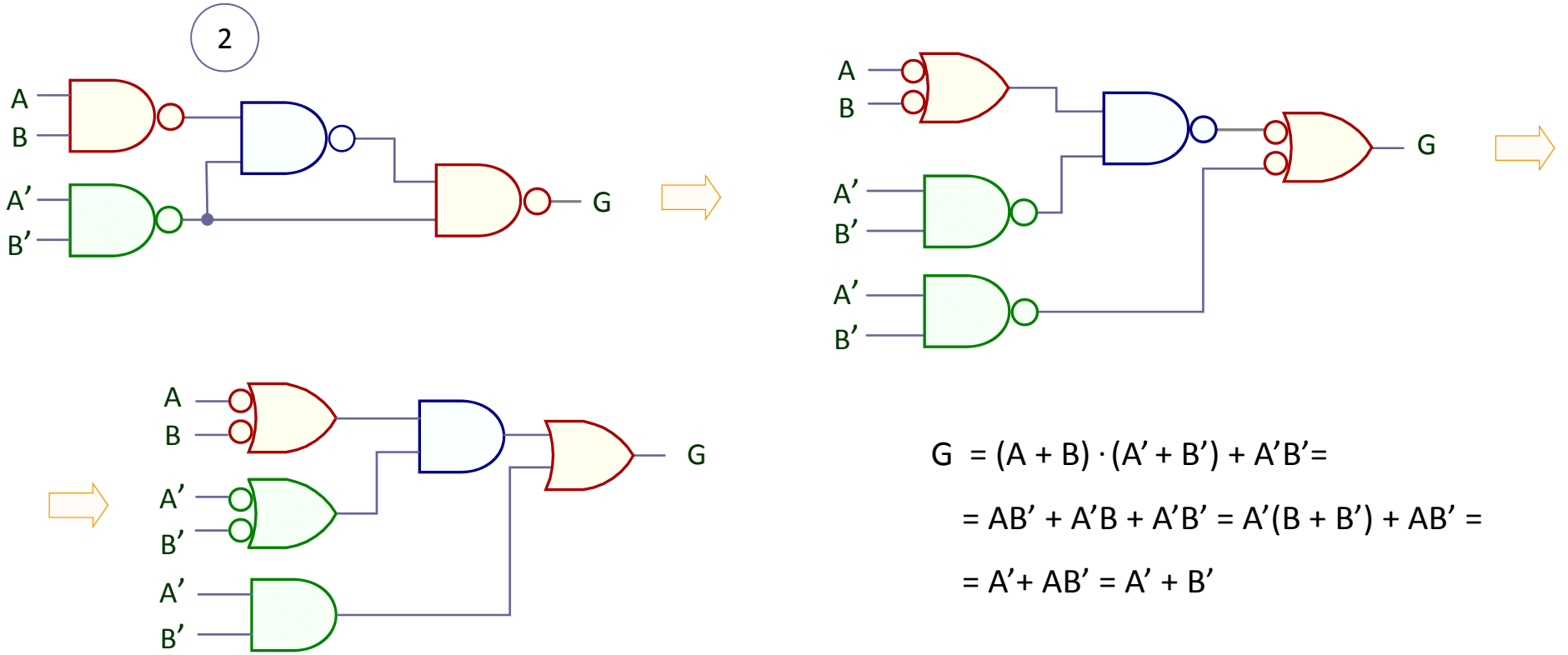




ANÁLISE DE CIRCUITOS COMBINATÓRIOS: CONVERSÃO GRÁFICA A AND, OR E NOT



$$F = AB + C'$$



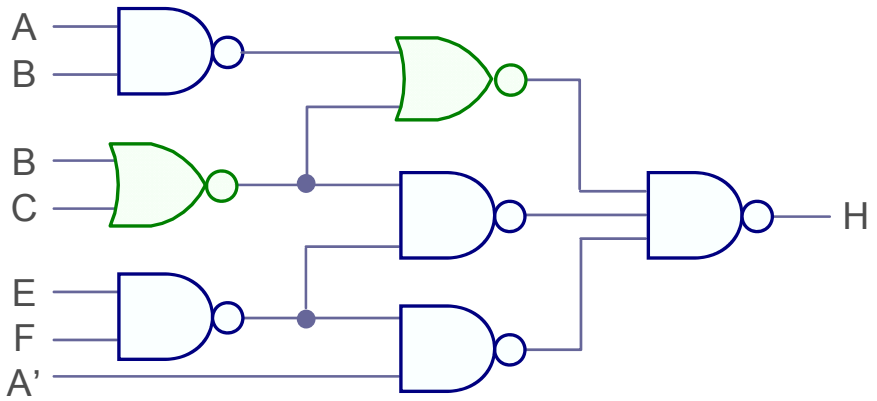
$$G = (A + B) \cdot (A' + B') + A'B' =$$

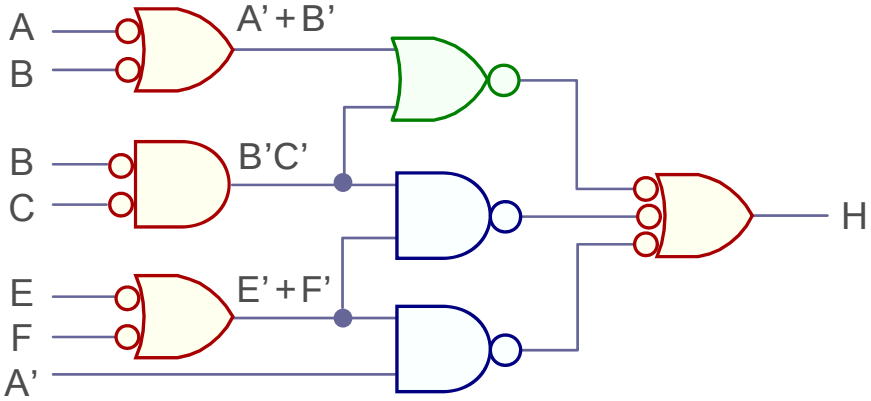
$$= AB' + A'B + A'B' = A'(B + B') + AB' =$$

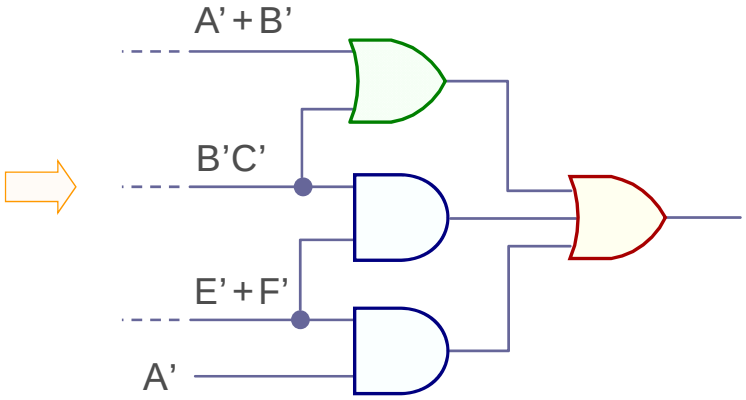
$$= A' + AB' = A' + B'$$

ANÁLISE DE CIRCUITOS COMBINATÓRIOS: CONVERSÃO GRÁFICA A AND, OR E NOT

3







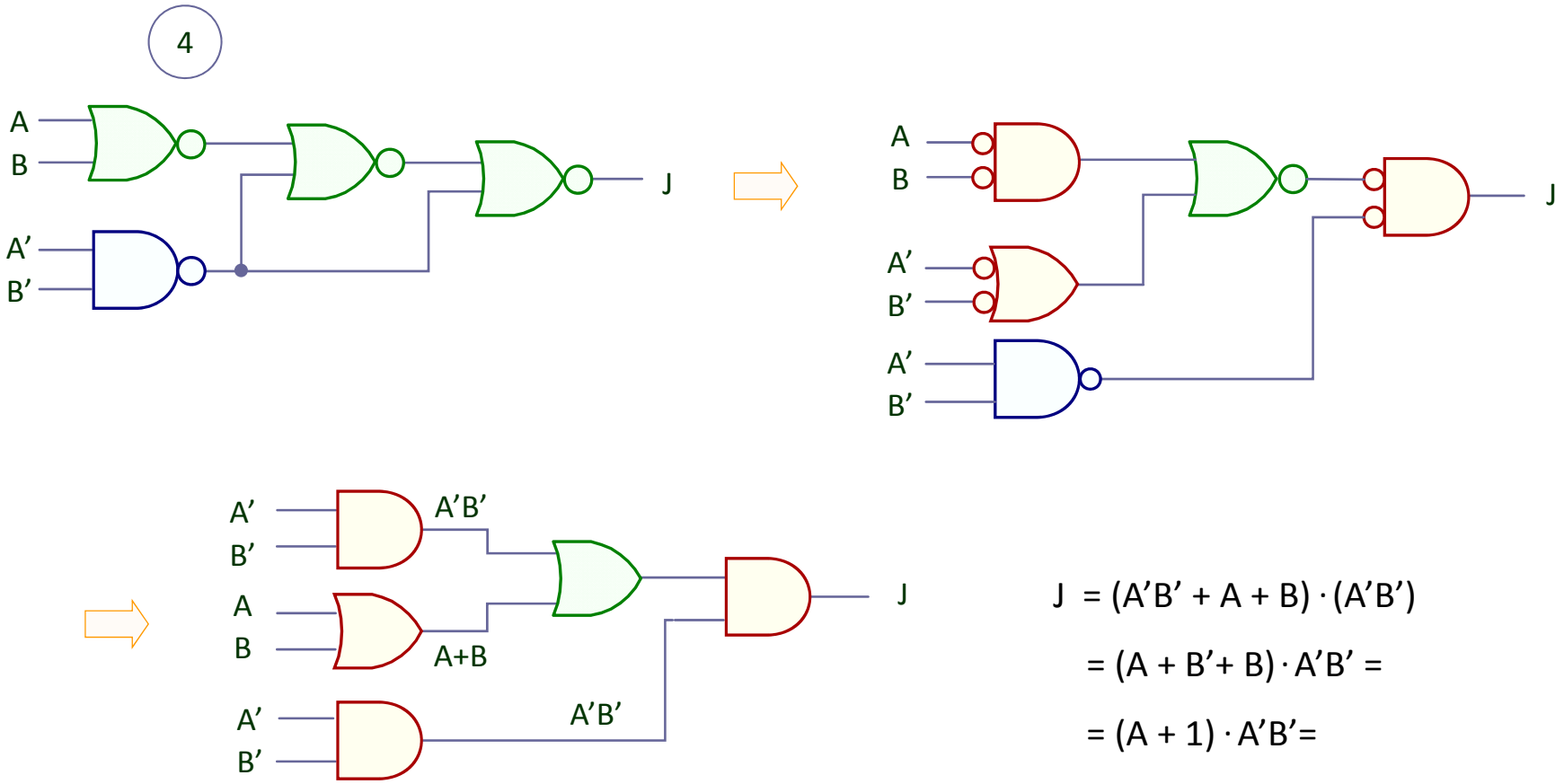
$$H = (A' + B' + B'C') + (E' + F') \cdot (B'C') + (E' + F') \cdot A' =$$

$$= A' (1 + (E' + F')) + B' + B'C' (1 + (E' + F')) =$$

$$= A' + B' + B'C' =$$

$$= A' + B'$$

ANÁLISE DE CIRCUITOS COMBINATÓRIOS: CONVERSÃO GRÁFICA A AND, OR E NOT



$$J = (A'B' + A + B) \cdot (A'B')$$

$$= (A + B' + B) \cdot A'B' =$$

$$= (A + 1) \cdot A'B' =$$

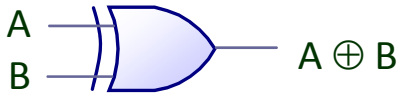
$$= A'B'$$

Operação XOR – União Exclusiva – Soma Aritmética Módulo-2: Definição e Representações

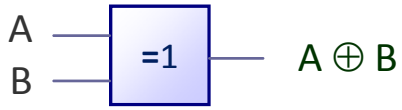
Definição do **XOR**: Operação sobre **n variáveis** que só toma o valor 1 quando um número ímpar dessas variáveis tomar o valor 1.

Entradas		Saída
B	A	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

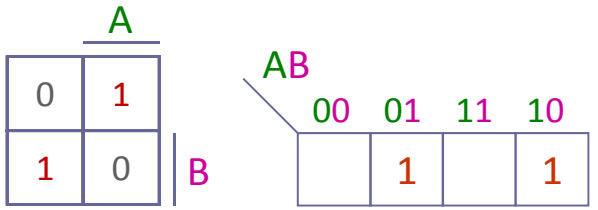
Tabela de Verdade do XOR a 2 variáveis.



Símbolo para uma porta XOR de 2 entradas.



Símbolo rectangular IEEE/IEC/ANSI do XOR .

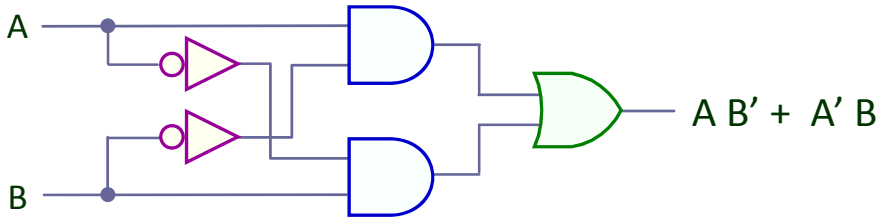


Mapa de Karnaugh do XOR a 2 variáveis (grafismos equivalentes).

$$A \oplus B = A\,B' + A'\,B$$

Equações lógicas do XOR (é utilizado o símbolo ⊕).

O XOR não é uma das funções básicas da álgebra de comutação. A nível da implementação tecnológica a sua construção é normalmente conseguida a partir do circuito:



Circuito de um XOR de 2 entradas sintetizado a partir das portas básicas AND, OR e NOT.

TEOREMAS ENVOLVENDO O XOR

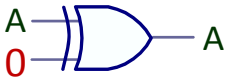
Existem vários teoremas que envolvem a função XOR de duas ou mais variáveis booleanas simples:

$A \oplus B = B \oplus A$ — **COMUTATIVIDADE** da função XOR

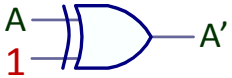
$A \oplus (B \oplus C) = (A \oplus B) \oplus C$ — **ASSOCIATIVIDADE** da função XOR

$A\,(B \oplus C) = (A\,B) \oplus (A\,C)$ — **DISTRIBUTIVIDADE** da função XOR

$A \oplus 0 = A$

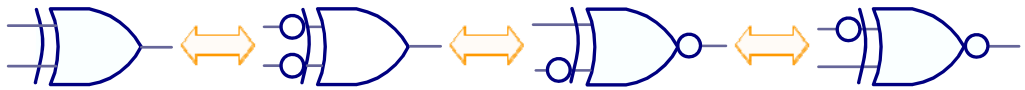


$A \oplus 1 = A'$

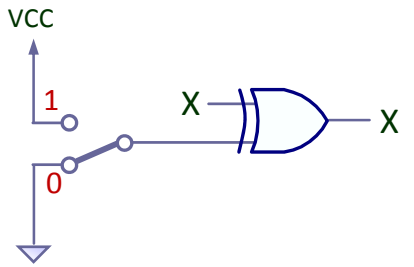


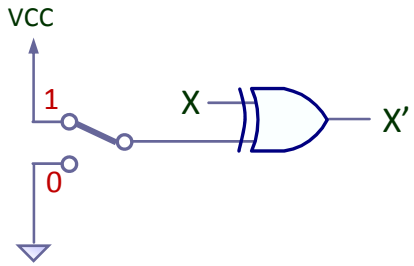
$(A \oplus B)' = A' \oplus B = A \oplus B'$

A função Equivalência ou XNOR $(A \oplus B)'$ é igual ao complemento da função XOR.



Quatro símbolos equivalentes para o XOR.





Inversor Controlado baseado num XOR de 2 entradas.

Destas propriedades decorre que há quatro símbolos equivalentes para o XOR (tal como para o XNOR adiante).

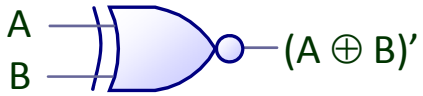
A equivalência é resultado de uma regra simples: quaisquer **dois** sinais de entrada ou saída de um XOR podem ser complementados sem que isto acarrete alteração da função lógica resultante.

Operação XNOR – Circuito de Equivalência: Definição e Representações

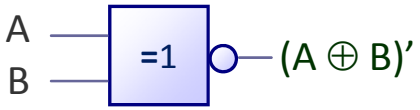
Definição do **XNOR**: Operação sobre **n variáveis** que só toma o valor 1 quando um número par dessas variáveis tomar o valor 1.

Entradas		Saída
B	A	A ⊕ B
0	0	1
0	1	0
1	0	0
1	1	1

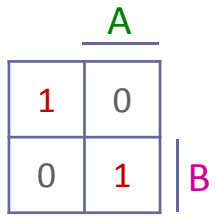
Tabela de Verdade do XNOR a 2 variáveis.



Símbolo para uma porta XNOR de 2 entradas (função de equivalência ou igualdade).



Símbolo rectangular IEEE/IEC/ANSI do XNOR .



Mapa de Karnaugh do XNOR a 2 variáveis.

$$(A \oplus B)' = A'\,B' + A\,B$$

Equação lógica do XNOR .

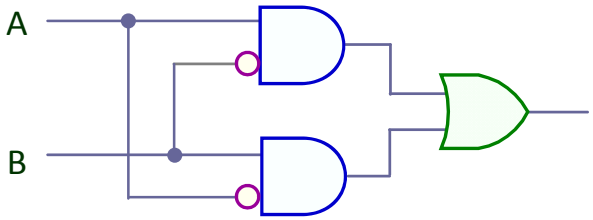
Tanto no XOR como no XNOR, podem complementar-se 2 sinais sem alteração da função lógica:

Quatro símbolos equivalentes para o XNOR.

Y	X	X XNOR Y	X '	X ' XOR Y
0	0	1	1	1
0	1	0	0	0
1	0	0	1	0
1	1	1	0	1

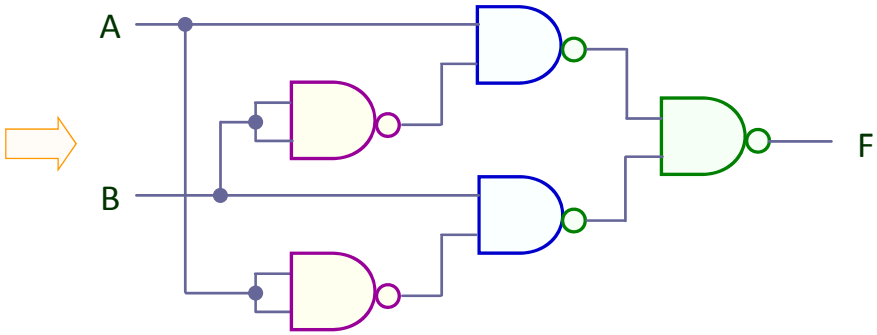
Verificação por Tabelas de Verdade da equivalência dos dois primeiros símbolos da série à esquerda.

Realização de uma Porta XOR a partir de Portas NAND

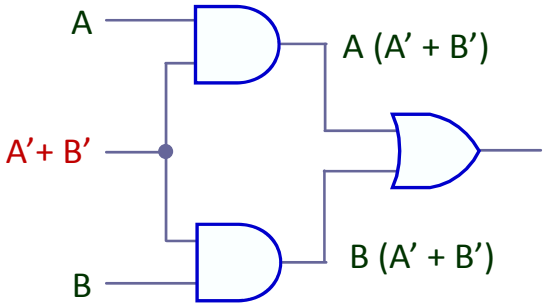


F = A B' + A' B

= A ⊕ B



Implementação de um XOR de 2 entradas com **5 portas** NAND de 2 entradas.

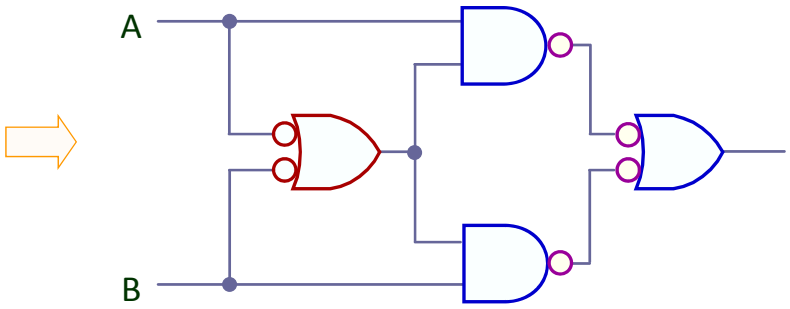


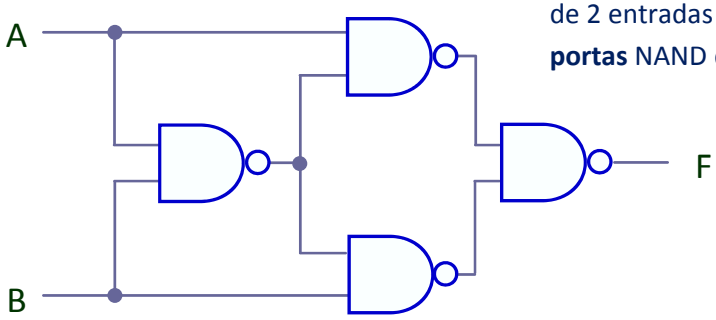
F = A (A´+ B') + B (A´+ B´) = A B' + A' B

= A ⊕ B

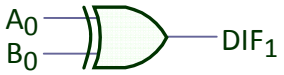
Teorema da Redundância

Implementação de um XOR de 2 entradas com apenas **4 portas** NAND de 2 entradas.





CIRCUITOS COMPARADOR E DE DETECÇÃO DE PARIDADE UTILIZANDO PORTAS XOR



Porta XOR de 2 entradas interpretada como um comparador de 1 bit.

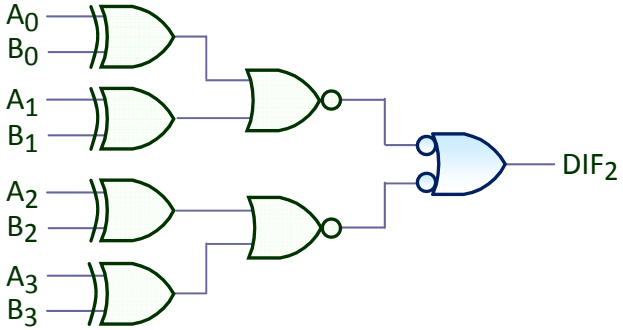


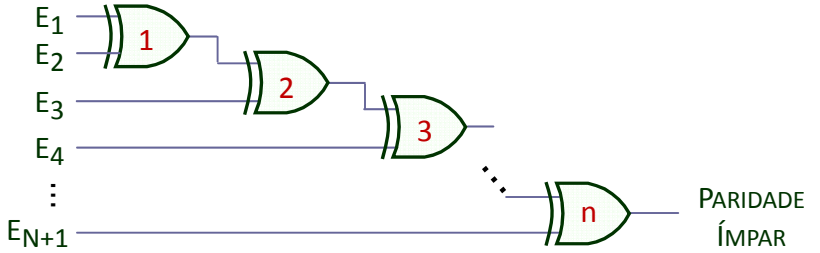
Diagrama lógico de um comparador 4 bits.

Um circuito que compara duas palavras binárias e indica se são ou não iguais é designado COMPARADOR. As portas XOR e XNOR podem ser consideradas comparador de 1 bit.

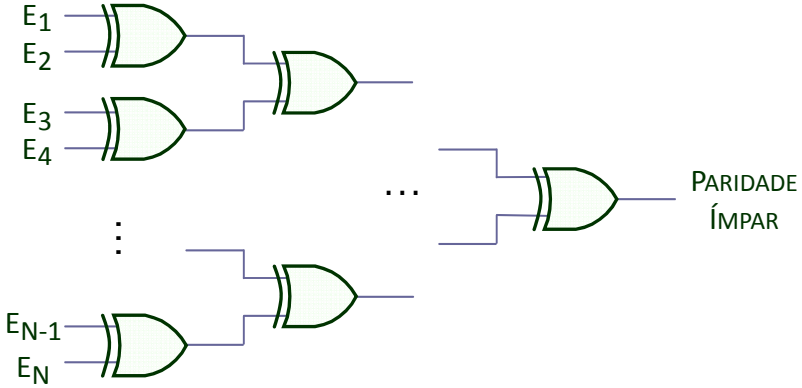
A saída DIF_1 é activada quando as entradas A_0 e B_0 são diferentes.

As saídas das quatro portas XOR comparadoras dos bits A_0 e B_0 a A_3 e B_3, são ligadas em OR para criação de um comparador de 4 bits.

A saída DIF_2 ficará activa quando quaisquer dos pares de bits de entrada assinalar uma desigualdade entre eles. Este circuito pode facilmente ser adaptado a um qualquer número de bits por palavra. Nos capítulos seguintes há mais considerações sobre COMPARADORES .



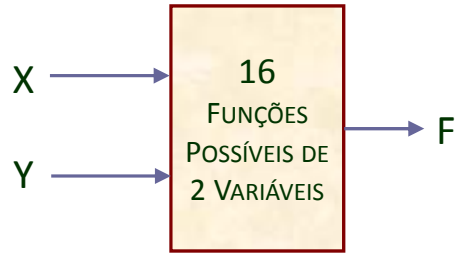
Encadeamento série de **n** portas XOR para formação de um circuito com n+1 entradas e uma saída indicativa de paridade impar.

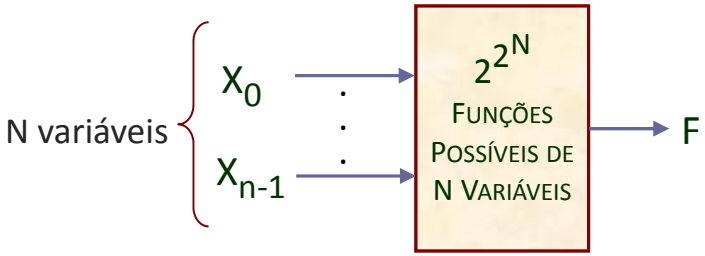


Encadeamento paralelo e série de **n** portas XOR para formação de um circuito com n+1 entradas e uma saída indicativa de paridade ímpar.

A saída dos circuitos é 1 se houver um número ímpar de entradas a 1.

Funções Lógicas Possíveis de N Variáveis



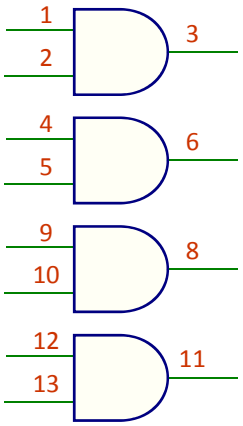


Entradas		F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
Y	X	0	AND	X<Y	Y	X>Y	X	XOR	OR	NOR	XNOR	X'	X<=Y	Y'	X>=Y	NAND	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

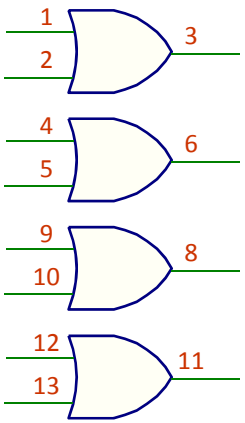
- Há **16** funções possíveis de 2 variáveis (F_0 a F_{15}), ou seja, dezasseis tabelas de verdade que podem ser construídas para duas variáveis. Algumas funções são familiares como o AND, OR, NOT, NAND, NOR, XOR, XNOR.
- Todas as funções podem ser descritas a partir de portas **AND**, **OR** e **NOT.**
- Generalizando, com N variáveis booleanas simples podem ser construídas 2^{2^N} funções booleanas simples, número que cresce exponencialmente com N.

PORTAS LÓGICAS SSI: PINOUT NA FORMA DIP

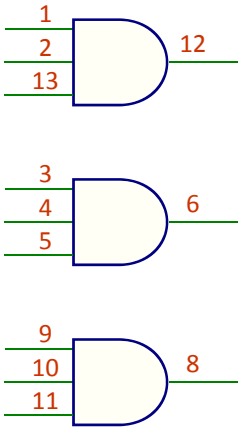
74x 08



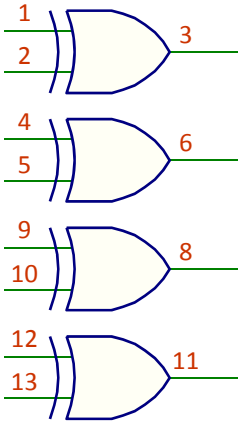
74x 32



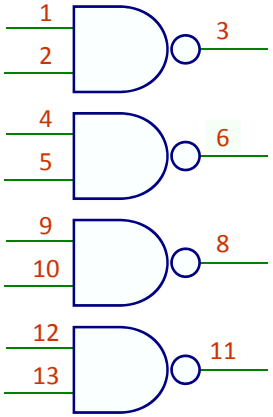
74x 11



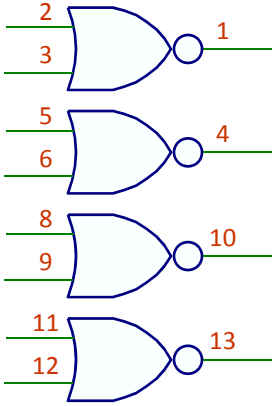
74x 86



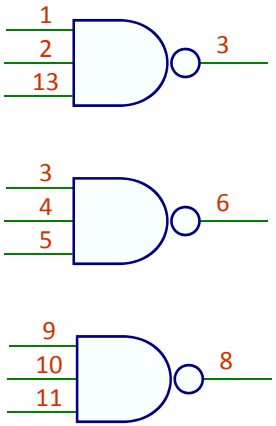
74x 00



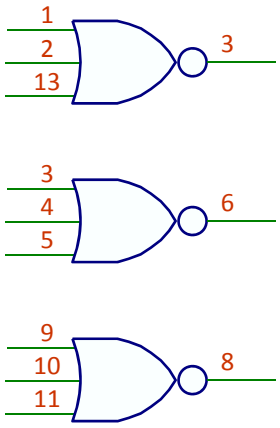
74x 02



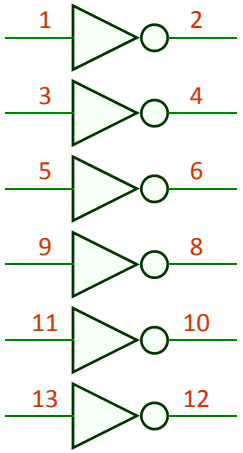
74x 10



74x 27



74x 04



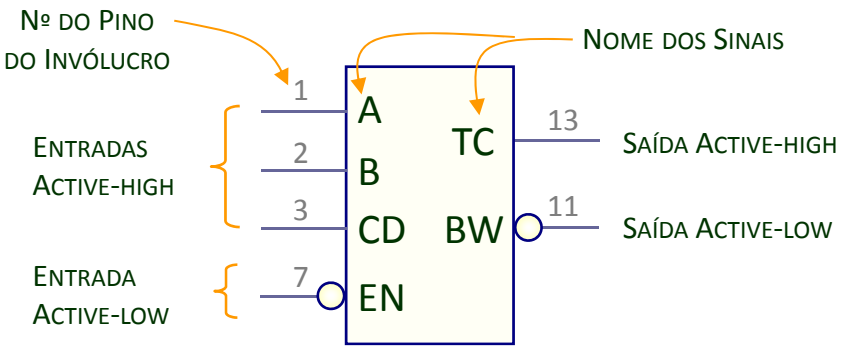
SSI – SMALL SCALE INTEGRATION

DIP – DUAL IN-LINE PACKAGE

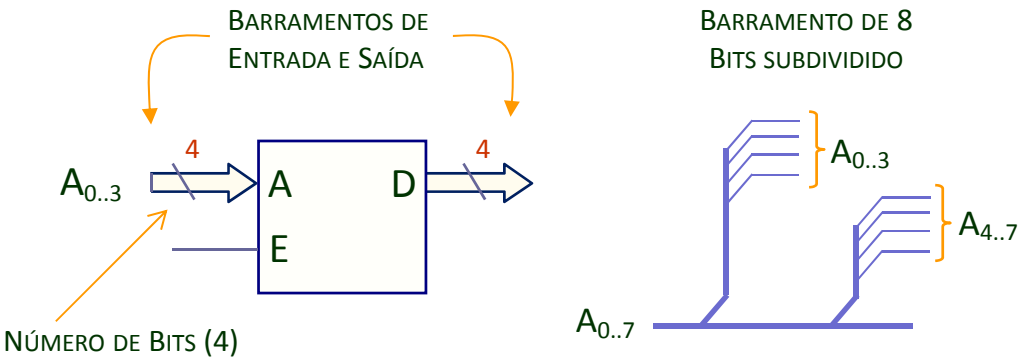
A vermelho estão indicados os pinos.

Simbologia de Circuitos e Nomenclatura de Sinais Lógicos

Nas portas lógicas elementares (AND, OR, NOT e outras) a função é especificada pelo seu símbolo.
Nas funções complexas, quando não existem símbolos padrão para cada função, usa-se uma caixa (normalmente um rectângulo vertical) com as entradas à esquerda e as saídas à direita como a indicada:



Caixa com entradas e saídas representando um circuito complexo.



Formas de representação de um barramento (bus).

No interior da caixa estão representados os nomes dos sinais.

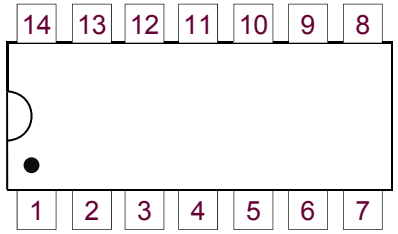
Um círculo inversor (**inversion bubble**) numa entrada ou numa saída denota um pino Active-low (L) e a sua ausência um pino Active-high (H).

Uma saída Active-low significa que, quando a saída da porta ou do circuito ficar activa, a linha de saída está a L, ou seja, ao valor lógico 0 em lógica positiva.

Uma saída Active-high significa que, quando a saída da porta ou do circuito ficar activa, a linha de saída está a H, ou seja, ao valor lógico 1 em lógica positiva.

Barramento (**bus**) é um grupo de n sinais lógicos relacionados entre si (n bits).

Usam-se nomes do tipo $A_{0..3}$ para referência de um barramento de 4 bits.



Numeração de pinos em embalagens DIP (vista de cima).

Os circuitos integrados utilizados nos trabalhos estão encapsulados num tipo de embalagem designado **DIP – Dual In-line Package** – por terem duas séries de terminais (pinos) que estão organizados em duas filas de lados opostos do encapsulamento.

O número de terminais dos circuitos integrados DIP varia entre 8 e 64 terminais. Ao lado representa-se um circuito integrado de 14 terminais. O terminal 1 e o 14 estão perto de uma pequena reentrância no encapsulamento (que pode assumir várias formas). A numeração é feita no sentido contrário ao dos ponteiros do relógio.

- Um sinal lógico (entrada ou saída de um circuito) do tipo **ACTIVE-LOW** tem no nome o sufixo **_L**.
- Um sinal **ACTIVE-HIGH** por simplicidade, não tem sufixo, por exemplo:

ACTIVE-LOW:	**READY_L**	**OPEN_L**
ACTIVE-HIGH:	**READY**	**OPEN**

- **NOME** de sinais – apenas um nome alfanumérico.
- **EXPRESSÃO LÓGICA** – combina nomes de sinais através dos operadores lógicos AND, OR, NOT.
- **EQUAÇÃO LÓGICA** – atribuição de uma expressão lógica a um nome de sinal.

READY' não é nome de sinal (' é o operador NOT).

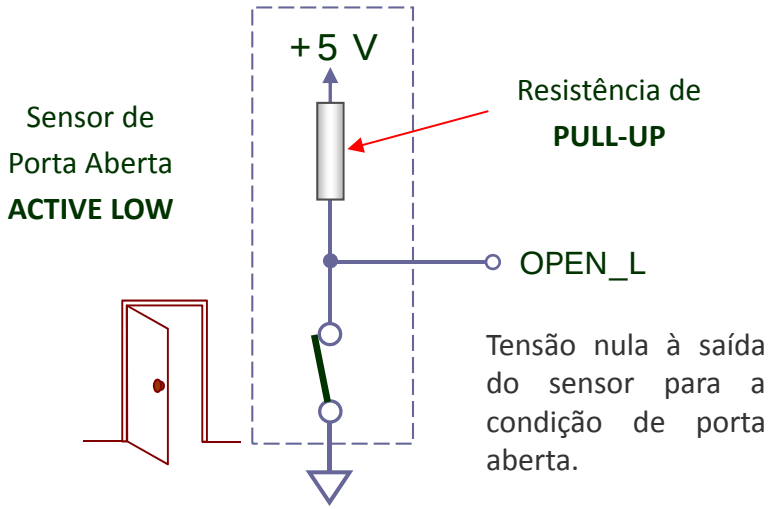
READY_L nome de sinal **ACTIVE-LOW**.

READY nome de sinal **ACTIVE-HIGH**.

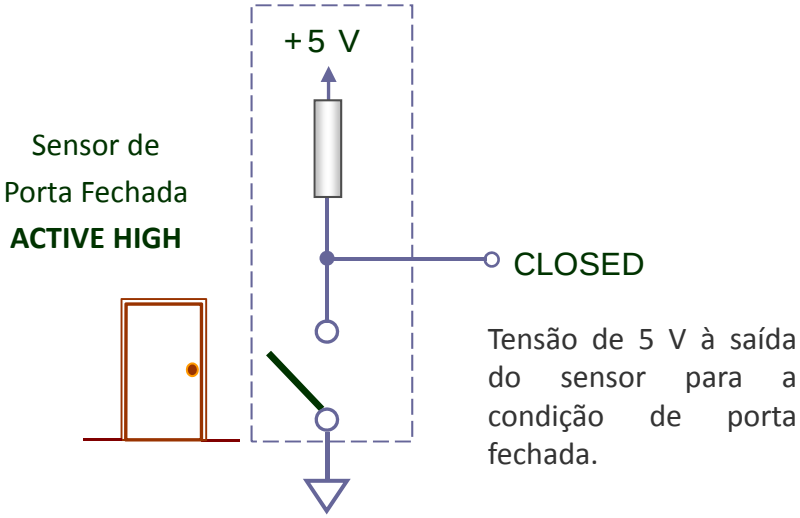
READY_L = **READY'**

O lado esquerdo da equação contêm o nome do sinal, o lado direito contêm uma expressão cujo valor será atribuído à variável: o sinal READY_L será pois obtido do sinal READY pelo operador NOT.

Nomenclatura de Sinais Lógicos



A configuração do sensor identifica uma situação em que a porta está aberta: aparece um nível 0 na linha.



A configuração do sensor identifica uma situação em que aporta está fechada: aparece um nível 1 na linha.

O sensor da condição da porta - ABERTA ou FECHADA - é constituído por um interruptor **SPST** (SINGLE POLE SINGLE THROW) e por uma resistência de PULL-UP ligada à tensão de alimentação de 5 V.

A linha que serve de suporte à variável que indica a condição da porta tem pelo menos duas designações alfanuméricas equivalentes, uma do tipo **ACTIVE HIGH**, outra do tipo **ACTIVE LOW**:

OPEN_L e CLOSED.

É opcional a escolha entre as duas.

De modo semelhante seria possível configurar o circuito de modo que o nível de tensão HIGH (1) indicasse porta ABERTA e que o nível LOW (0) indicasse porta FECHADA. Nesse caso ter-se-ia:

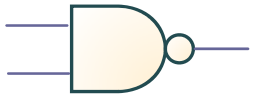
OPEN e CLOSED_L.

Na representação de diagramas lógicos e na implementação dos circuitos deve sempre conferir-se conteúdo semântico às variáveis e às funções booleanas. Cada linha deve identificar não só a função exercida como ainda explicitar o nível H ou L em que essa função se realiza.

NÍVEL DE ACTIVIDADE DAS ENTRADAS E SAÍDAS

Podem gerar-se 8 símbolos de portas AND com 2 entradas consoante a natureza das entradas e das saídas. O mesmo é válido para a porta OR. Em baixo representam-se 4 formas de obtenção de uma porta AND, e mais abaixo ainda 4 formas de obtenção de uma porta OR.

AND de duas entradas ACTIVE-HIGH que produz uma saída ACTIVE-HIGH.



AND de duas entradas ACTIVE-HIGH que produz uma saída ACTIVE-LOW (na realidade o circuito é um NAND).

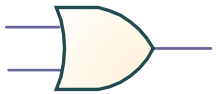
AND de duas entradas ACTIVE-LOW que produz uma saída ACTIVE-HIGH (na realidade o circuito é um NOR).

AND de duas entradas ACTIVE-LOW que produz uma saída ACTIVE-LOW (na realidade o circuito é um OR).

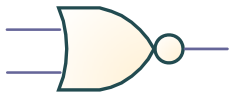
Na Fig. em cima todas as portas realizam genericamente uma operação AND com 2 entradas. A saída de cada uma é activada se ambas as entradas forem activadas. Todas elas possuem no entanto diferentes tabelas de verdade (são circuitos diferentes).

De igual modo, na Fig. em baixo todas as portas realizam uma operação OR. A saída de cada uma é activada se uma das entradas for activada.

Em ambos os casos a diferença de leitura reside no nível de actividade ACTIVE-HIGH ou ACTIVE-LOW das entradas e saídas.

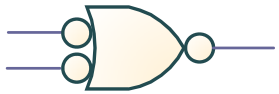


OR de duas entradas ACTIVE-HIGH que produz uma saída ACTIVE-HIGH.



OR de duas entradas ACTIVE-HIGH que produz uma saída ACTIVE-LOW (na realidade o circuito é um NOR).

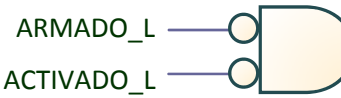
OR de duas entradas ACTIVE-LOW que produz uma saída ACTIVE-HIGH (na realidade o circuito é um NAND).



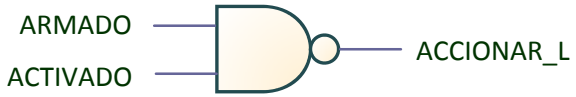
OR de duas entradas ACTIVE-LOW que produz uma saída ACTIVE-LOW (na realidade o circuito é um AND).

Documentação Correcta de um Circuito

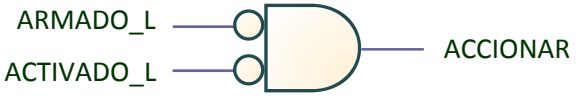
Com entradas Active-high e saída Active-high.



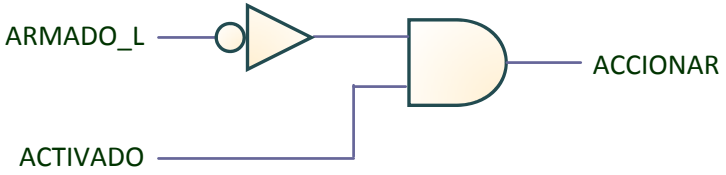
Com entradas Active-low e saída Active-low.



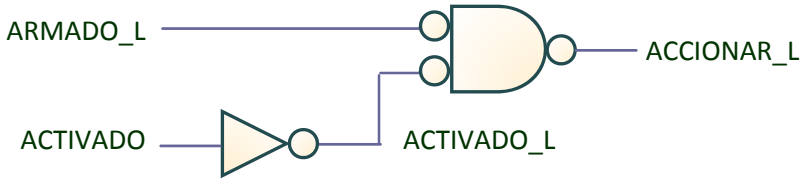
Com entradas Active-high e saída Active-low.



Com entradas Active-low e saída Active-high.



Com entradas Active-low e Active-High e saída Active-high.



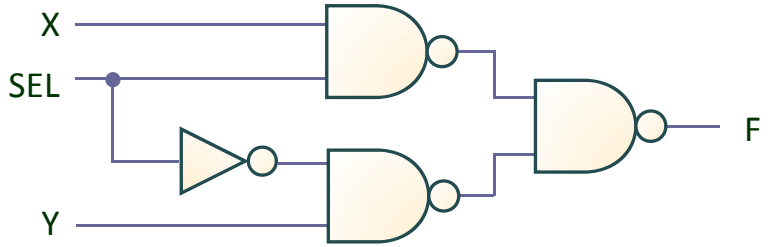
Com entradas Active-low e Active-High e saída Active-low.

Os circuitos ao lado representam 6 formas alternativas de implementação de uma mesma função Accionar, tomando em conta que os sinais de entrada e de saída podem ter natureza Active-high ou Active-low.

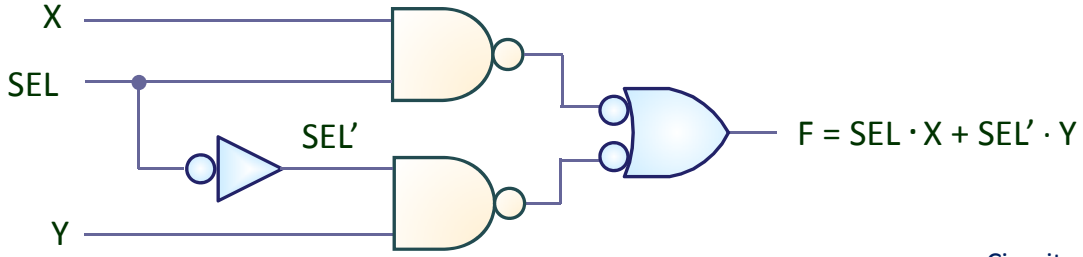
Há que utilizar o símbolo mais expressivo da função e adaptar o diagrama lógico obtido aos níveis de actividade das entradas e saídas das portas em cada caso.

A designação de cada sinal deve fazer coincidir a natureza Active-high ou Active-low da variável ou função suportada com o tipo de entrada ou saída da porta a que está ligado, de acordo com as regras do Bubble-to-bubble design.

DOCUMENTAÇÃO CORRECTA DE UM CIRCUITO



Circuito pouco inteligível, em que as designações dos sinais não revelam as suas funcionalidades.



Circuito em que a função de saída F pode ser lida directamente do diagrama.

Apesar dos 5 círculos inversores, o circuito em baixo é fácil de compreender:

- a saída F é igual a X se a entrada SEL estiver activa em 1, e
- igual a Y em caso contrário.

Utilizou-se o símbolo alternativo da porta NAND de saída adequado ao nível de actividade das saídas das portas anteriores. Genericamente devem modificar-se as variáveis e os respectivos níveis de actividade para ficarem iguais aos níveis de actividade das entradas e saídas das portas sempre que possível.

Diagrama Temporal e Tempo de Atraso

Um diagrama temporal ilustra o comportamento lógico de sinais num circuito integrado (CI) digital em função do tempo.

A Fig. ilustra o Tempo de Propagação ou Tempo de Atraso – Delay – de 2 saídas **X** e **Y** relativamente a uma entrada A que as influencia (supõe-se que as outras entradas se mantêm constantes).

Esse tempo é definido como o atraso entre a mudança de nível na entrada e a correspondente mudança de nível na saída e assinala-se entre o **ponto médio** das transições de sinal (as linhas inclinadas assinalam que não ocorrem em tempo 0). As setas assinalam relações de causalidade entre transições.

Diferentes caminhos lógicos dentro do circuito correspondem a diferentes atrasos. O atraso $\mathbf{t_X}$ de **A** para **X** é menor que o atraso $\mathbf{t_Y}$ de **A** para **Y**.

O atraso $\mathbf{t_{LH}}$ no flanco ascendente de qualquer dos dois sinais de saída (mudança de 0→1, **Low-to-High**) pode ser diferente do atraso $\mathbf{t_{HL}}$ que ocorre no flanco descendente (mudança de 1→0, **High-to-Low**).

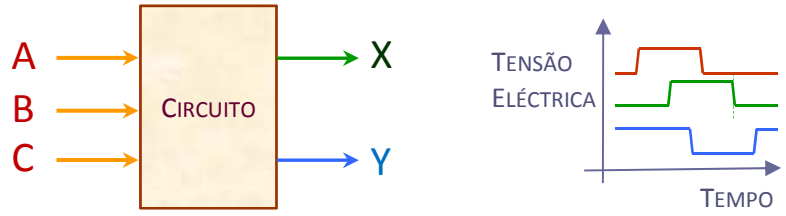
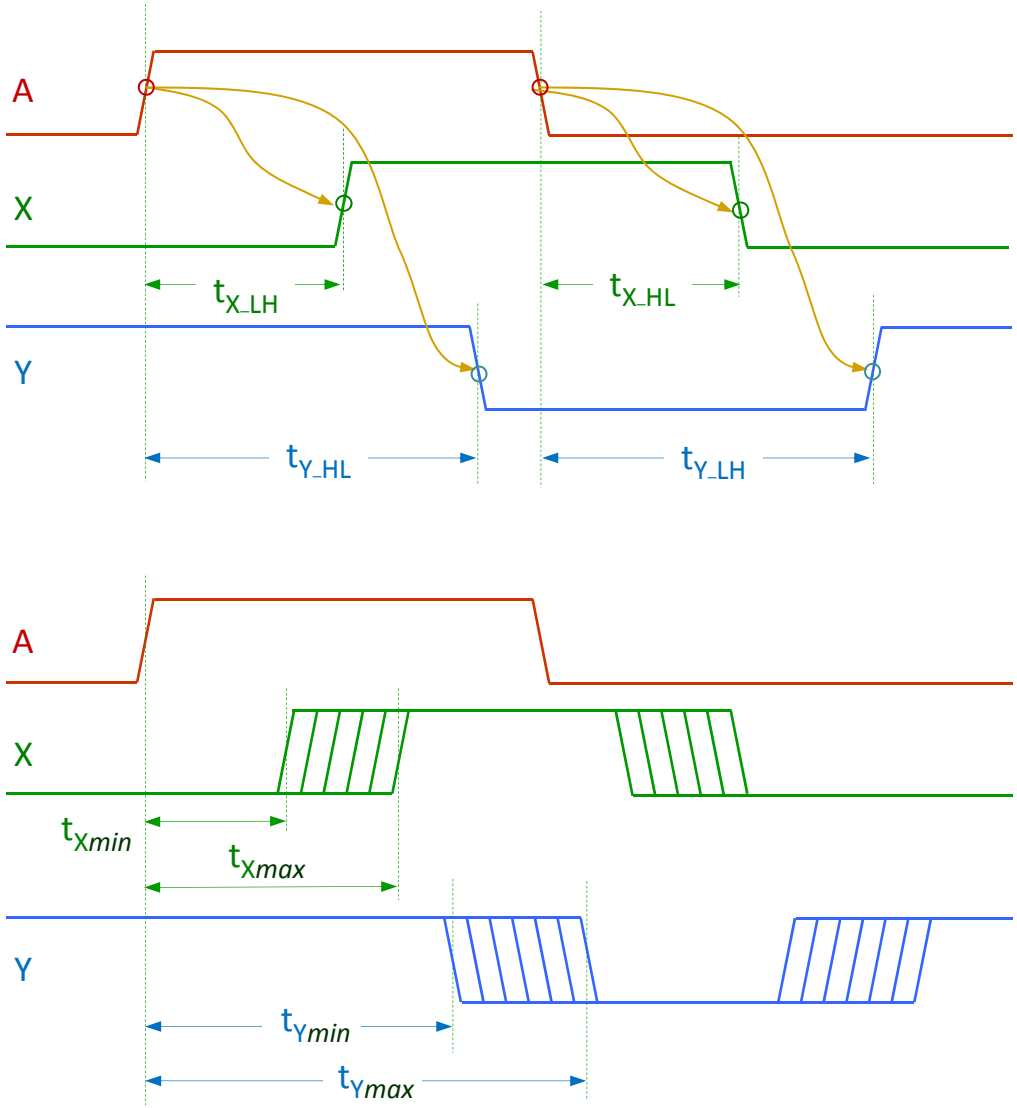
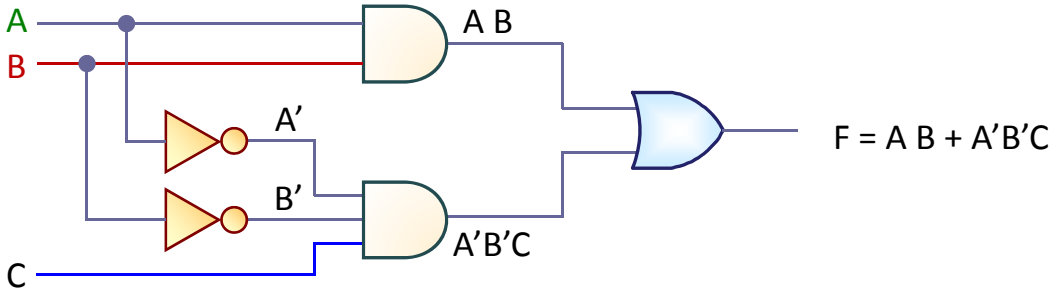


Diagrama de blocos do circuito e gráfico de tensões no tempo.

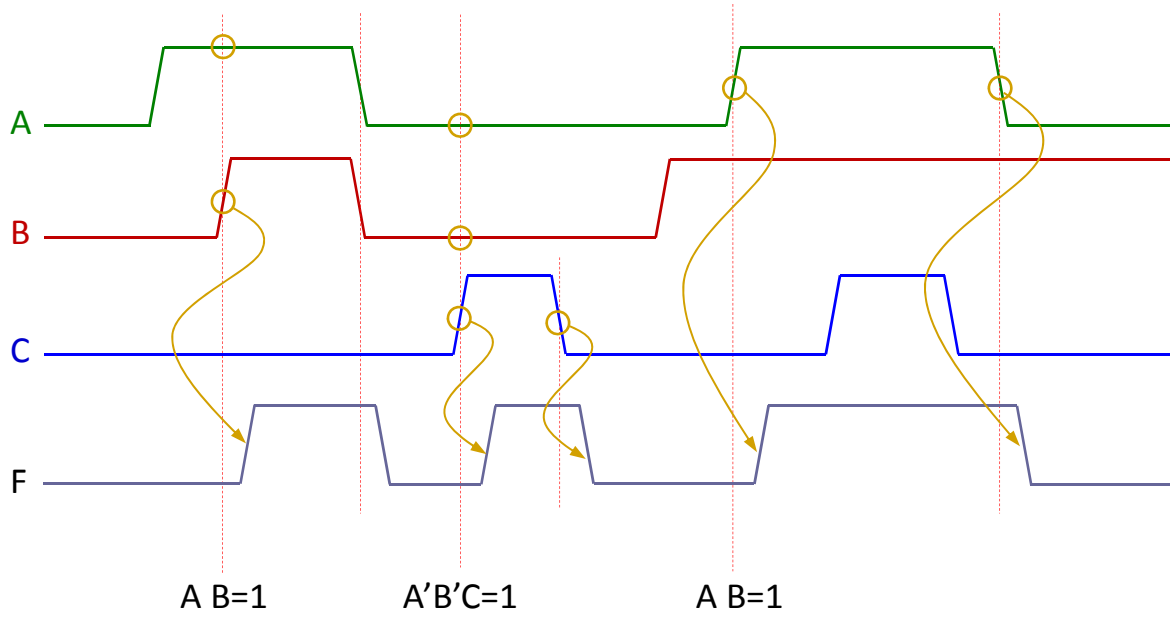


Diagramas temporais assinalando os tempos relativos (em cima) e os máximos e mínimos (em baixo).

Diagrama Temporal e Tempo de Atraso



Circuito lógico do tipo AND-OR com uma evolução temporal de sinais mostrada em baixo.

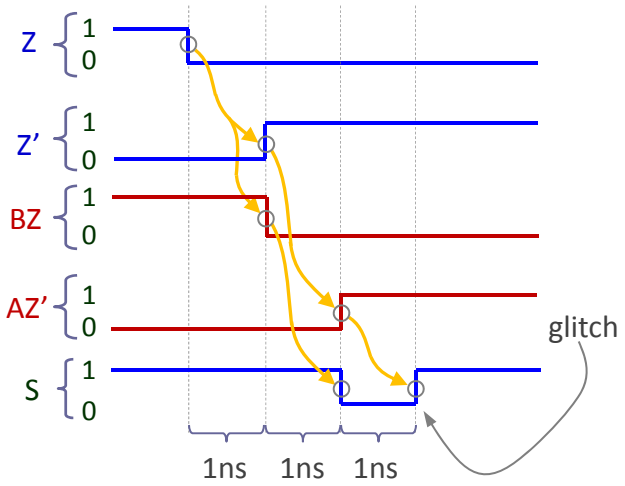


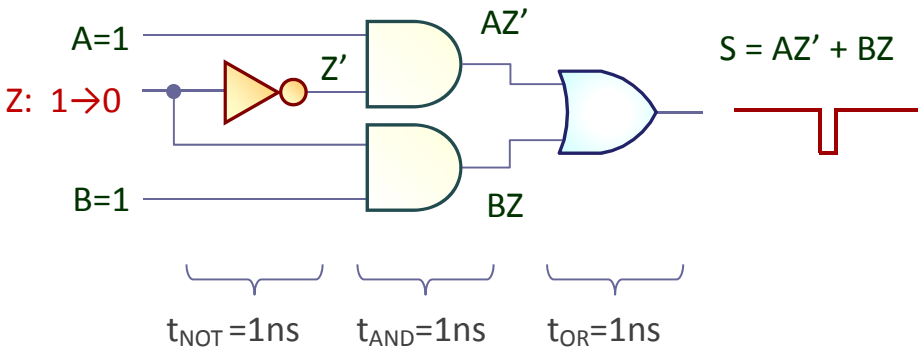
O tempo de atraso depende de valores de tensão, temperatura e parâmetros de fabrico, pelo que é usual especificar valores mínimos (t_{min}), típicos e máximos (t_{max}) como na Fig. anterior.

O tempo de propagação total de um circuito digital é a soma dos tempos de atraso das portas em série.

Diagrama temporal do circuito acima denotando os atrasos inerentes à geração da saída F face à evolução temporal das entradas A, B e C representadas.

TEMPO DE ATRASO E GERAÇÃO DE GLITCHES





Circuito típico de **HAZARD** estático e diagrama temporal evidenciando um **GLITCH** a 0 de duração igual a uma unidade do tempo de atraso.

A Fig. mostra um circuito clássico que apresenta dois pares de entradas diferindo apenas numa variável Z. A saída S apresenta trivialmente um 1-estático quando A=B=1, independentemente do valor de Z.

Porém, quando Z transita de 1 para 0, observar-se-á na saída S um impulso transitório 0 de curta duração designado **GLITCH**.

Na Fig. considerou-se um atraso hipotético de 1 nanosegundo por porta para facilidade da ilustração.

A análise do comportamento estático da saída (steady-state output) considera as entradas A, B e Z do circuito fixas no tempo e a saída S estabilizada após expiração de todos os tempos de atraso devidos à propagação do sinal Z através dos portas lógicas,

Este tipo de análise (estática) prediz que a saída S se mantém inalterada, ignorando o comportamento transitório ou dinâmico e a aparição do **GLITCH**. A análise dinâmica com a abordagem da dimensão temporal de um circuito revela-se assim de primordial importância.

GLITCHES

O termo **GLITCH** significa literalmente um **PICO DE CURTA DURAÇÃO**. É sempre provocado por uma diferença de tempos de propagação em caminhos paralelos de um sinal, que origina um variação de curta duração numa saída quando se esperava que se mantivesse inalterada.

A designação de **HAZARD** ('perigosidade') refere-se à existência da possibilidade de o circuito gerar um **GLITCH**. A existência ou não do **GLITCH** depende das características eléctricas e lógicas do circuito.

Ocorre um **HAZARD ESTÁTICO** quando existe a possibilidade de uma saída sofrer uma transição momentânea em condições em que se esperava que ela se mantivesse inalterada.

HAZARD ESTÁTICO-1

O sinal de saída está sempre a 1, o **GLITCH** ocorre a 0.

HAZARD ESTÁTICO-0

O sinal de saída está sempre a 0, o **GLITCH** ocorre a 1.

GLITCH

Um Glitch é um impulso de curta duração normalmente indesejável.

Ocorre um **HAZARD DINÂMICO** quando for possível a uma saída mudar mais do que uma vez, em condições em que se esperava que ela tivesse uma única transição (de 0 --->1 ou de 1--->0).

HAZARD DINÂMICO

O sinal de saída está a mudar para 1.

HAZARD DINÂMICO

O sinal de saída está a mudar para 0.

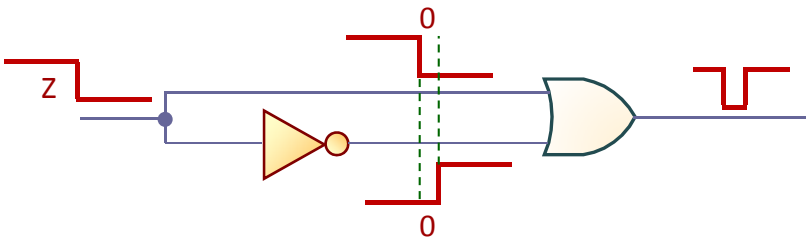
Os **HAZARDS** estáticos podem ser eliminados.

A detecção de **HAZARDS** e a eliminação de **GLITCHES** faz-se através de métodos que estão fora do alcance deste capítulo.

HAZARD ESTÁTICO-1 (STATIC-1 HAZARD)

A saída está a 1, mas vai momentaneamente a 0 (Glitch a 0), como resultado de uma mudança na entrada (verifica-se nos circuitos AND-OR).

Qualquer circuito com um **HAZARD ESTÁTICO-1** (ou **HAZARD** num **1-ESTÁTICO**) pode reduzir-se ao circuito básico equivalente da Fig. em baixo, considerando que as outras variáveis existentes assumem um valor constante.



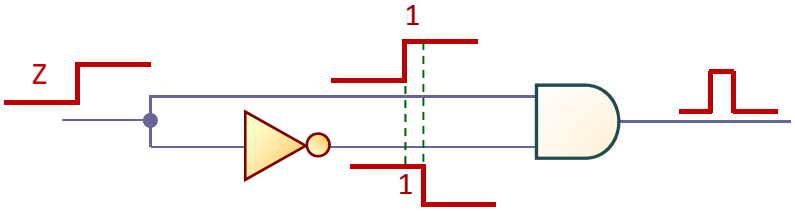
Circuito típico de **HAZARD ESTÁTICO-1** gerador de um **GLITCH** a 0 de duração igual ao tempo de atraso de uma porta.

Características: dois caminhos paralelos para uma variável Z em mudança, um deles invertido, convergindo ambos no final numa porta OR.

HAZARD ESTÁTICO-0 (STATIC-0 HAZARD)

A saída está a 0, mas vai momentaneamente a 1 (Glitch a 1), como resultado de uma mudança na entrada (verifica-se nos circuitos OR-AND).

Qualquer circuito com um **HAZARD ESTÁTICO-0** (ou **HAZARD** num **0-ESTÁTICO**) pode reduzir-se ao circuito básico equivalente da Fig. em baixo, considerando que as outras variáveis existentes assumem um valor constante.



Circuito típico de **HAZARD ESTÁTICO-0** gerador de um **GLITCH** a 1 de duração igual ao tempo de atraso de uma porta.

Características: dois caminhos paralelos para uma variável Z em mudança, um deles invertido, convergindo ambos no final numa porta AND.

Simplificação Algébrica de uma Função para Obtenção da Forma AND-OR

Exercício 1-1

$$F = ABC + \underbrace{(A' + D' + \overbrace{(B' \oplus BC)' \cdot (A' + B')}^{Y})'}_{X} + \underbrace{A \cdot (B + C')' D}_{Z}$$

Objectivo

Obter a forma AND-OR da função F definida pela equação ao lado simplificando algebricamente.

1. $X = (A' + D' + Y)' = A \cdot D \cdot Y'$ agora torna-se necessário determinar Y'

2. $Y = (B' \oplus BC)' \cdot (A' + B') \rightarrow Y' = \underbrace{(B' \oplus BC)}_{\underbrace{B' \cdot (BC)'}_{B'(B' + C')} + \underbrace{B \cdot BC}_{BC}} + AB$

$$B'(B' + C') = B'B' + B'C' = B' + B'C' = B'(1 + C') = B'$$

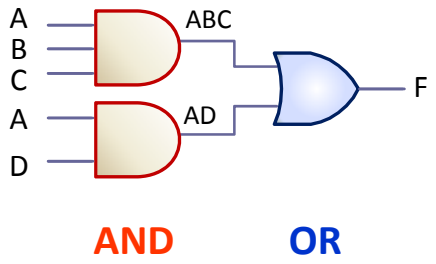
$$BC$$

$$Y' = \underbrace{B' + BC}_{B' + C} + AB = B' + C + AB = \underbrace{B' + AB}_{B' + A} + C = B' + A + C$$

3. $X = A \cdot D \cdot Y' = AD\,(A + B' + C) = AD + AB'D + ACD = AD\,(1 + B' + C) = AD$

4. $z = A \cdot (B + C')' \cdot D = A \cdot (B'\,C) \cdot D = A\,B'\,C\,D$

5. $F = ABC + X + Z = ABC + AD + AB'CD = ABC + AD\,(1 + B'C) = ABC + AD$



Simplificação Algébrica de uma Função para Obtenção da Forma AND-OR

Exercício 1-2

Objectivo

Obter a forma AND-OR da função F definida pela equação ao lado simplificando algebricamente.

$$F = A\,(\,\underbrace{(\,B' + D'\,)' + B\,C'D}_{X}\,) + [\,\underbrace{(\,C' + D\,)\cdot(\,B + (\,C \oplus D\,)'\,)}_{Y}\,]'$$

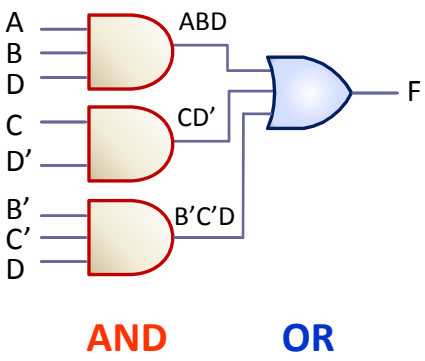
1. $X = (\,B' + D'\,)' + B\,C'\,D = B\,D + B\,C'\,D = BD\,(\,\underbrace{1 + C'}_{1}\,) = BD$

2. $Y = \underbrace{(C' + D)'}_{C\,D'} + \underbrace{(B + (C \oplus D)')'}_{B'\,(C \oplus D)}$

$$B'\,(C\,D' + C'\,D) = B'\,C\,D' + B'\,C'\,D$$

$$Y = C\,D' + B'\,C\,D' + B'\,C'\,D = C\,D' + \underbrace{(1 + B')}_{1} + B'\,C'\,D = C\,D' + B'\,C'\,D$$

3. $F = A\,X + Y = A\,B\,D + C\,D' + B'\,C'\,D$



LSD – 1 ÍNDICE-1

LSD – 1 ÍNDICE-2

LSD – 1 ÍNDICE-3