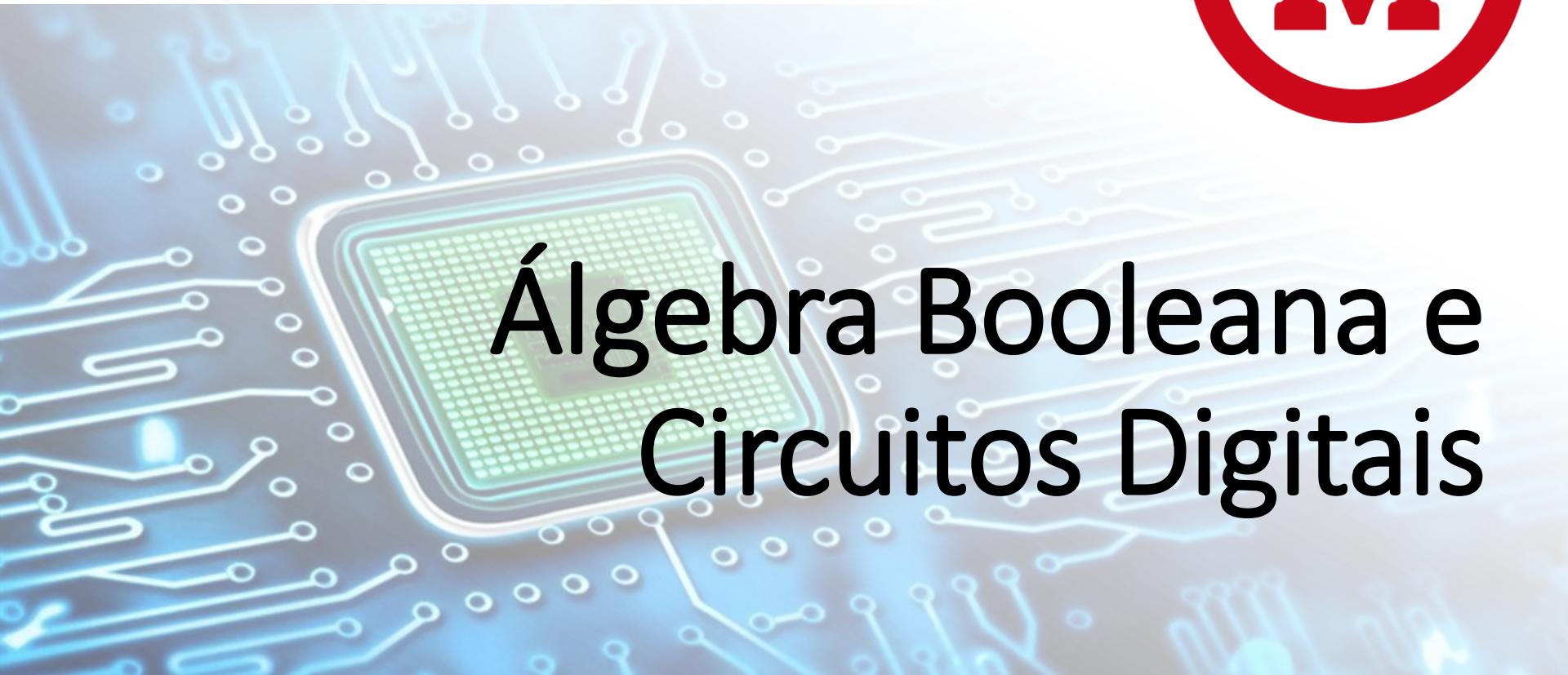




Álgebra Booleana e Circuitos Digitais



Prof. Me. Wallace Rodrigues de Santana



wallace.santana@mackenzie.br



Atribuição-NãoComercial-Compartilhalgual 3.0 Brasil (CC BY-NC-SA 3.0)

Você tem a liberdade de:



Ficando claro que:

Compartilhar — copiar, distribuir e transmitir a obra.

Remixar — criar obras derivadas.

Renúncia — Qualquer das condições acima pode ser **renunciada** se você obtiver permissão do titular dos direitos autorais.

Domínio Público — Onde a obra ou qualquer de seus elementos estiver em **domínio público** sob o direito aplicável, esta condição não é, de maneira alguma, afetada pela licença.

Outros Direitos — Os seguintes direitos não são, de maneira alguma, afetados pela licença:

- Limitações e exceções aos direitos autorais ou quaisquer **usos livres** aplicáveis;
- Os **direitos morais** do autor;
- Direitos que outras pessoas podem ter sobre a obra ou sobre a utilização da obra, tais como **direitos de imagem** ou privacidade.

Aviso — Para qualquer reutilização ou distribuição, você deve deixar claro a terceiros os termos da licença a que se encontra submetida esta obra. A melhor maneira de fazer isso é com um link para esta página.



Atribuição — Você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra).



Uso não comercial — Você não pode usar esta obra para fins comerciais.



Compartilhamento pela mesma licença — Se você alterar, transformar ou criar em cima desta obra, você poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente.



Diversidade e inclusão

Este material foi escrito visando promover a diversidade e a inclusão, respeitando e valorizando as relações humanas de modo a propiciar uma cultura mais inclusiva e que impacte positivamente a sociedade.

Ao adotar uma linguagem inclusiva, este material busca repensar os termos usados na literatura técnica para reduzir as barreiras à equidade e promover o respeito, buscando estar livre de linguagem ofensiva ou sugestiva.

A indústria de Tecnologia da Informação tem trabalhado arduamente para mudar estes termos para alternativas mais apropriadas, mas sistemas legados ainda poderão contê-los.

Quem é Wallace Santana?



Formação Acadêmica

- Tecnólogo em Mecânica de Precisão pela FATEC-SP [2001]
- Tecnólogo em Informática pela FATEC Mauá [2005]
- Mestre em Engenharia da Informação pela UFABC [2010]
- Especialista em Gestão Pública pela UNIFESP [2019]
- Engenheiro de Computação pela UNIVESP [2022]



Experiência Profissional

- Analista de Sistemas na CEAGESP [2005-2008]
- Analista de Tecnologia da Informação e Comunicação na PRODAM [2008-2010]
- Consultor Técnico Legislativo na Câmara Municipal de São Paulo [desde 2010]



Docência

- Faculdade de Mauá [2011-2015]
- FATEC São Caetano do Sul [2016-2017] [2021-2022]
- Centro Universitário Drummond [2018-2022]
- Universidade Presbiteriana Mackenzie [desde 2022]

Unidade 1

Apresentação

Objetivos

Fatos e conceitos:

- Entender as operações básicas da Álgebra Booleana;
- Conhecer expressões duais e complementares para simplificação de expressões;
- Validar teoremas da Álgebra de Boole;
- Entender como circuitos digitais são construídos através do uso da Álgebra de Boole. Como podem ser analisados e projetados a partir de expressões booleanas ou tabelas.

Objetivos

Procedimentos e habilidades:

- Aplicar a metodologia de desenvolvimento de projetos na implementação de circuitos digitais;
- Resolver problemas através de raciocínio lógico;
- Executar trabalhos em equipe.
- Definir blocos lógicos e aritméticos digitais básicos que constituem diferentes organizações de computadores.

Objetivos

Atitudes, normas e valores:

- Iniciativa, independência e responsabilidade no aprendizado;
- Capacidade de realizar trabalhos em grupo e individualmente em aulas práticas seguindo prazos determinados;
- Conscientização e realização de estudo contínuo e sistemático da disciplina durante a realização das aulas para o aproveitamento do mesmo com auxílio dos livros indicados na bibliografia;
- Respeitar a produção intelectual de terceiros, sejam colegas, professores ou autores de textos disponibilizados através de algum meio de pesquisa;
- Respeitar os princípios éticos na tomada de decisões tecnológicas que influenciam diretamente na vida de terceiros.

Conteúdo programático

PARTE I

1. Apresentação
2. Sistemas de numeração
3. Portas lógicas e Álgebra Booleana
4. Teoremas
5. Formas canônicas
6. Mapas de Veitch-Karnaugh

Conteúdo programático

PARTE II

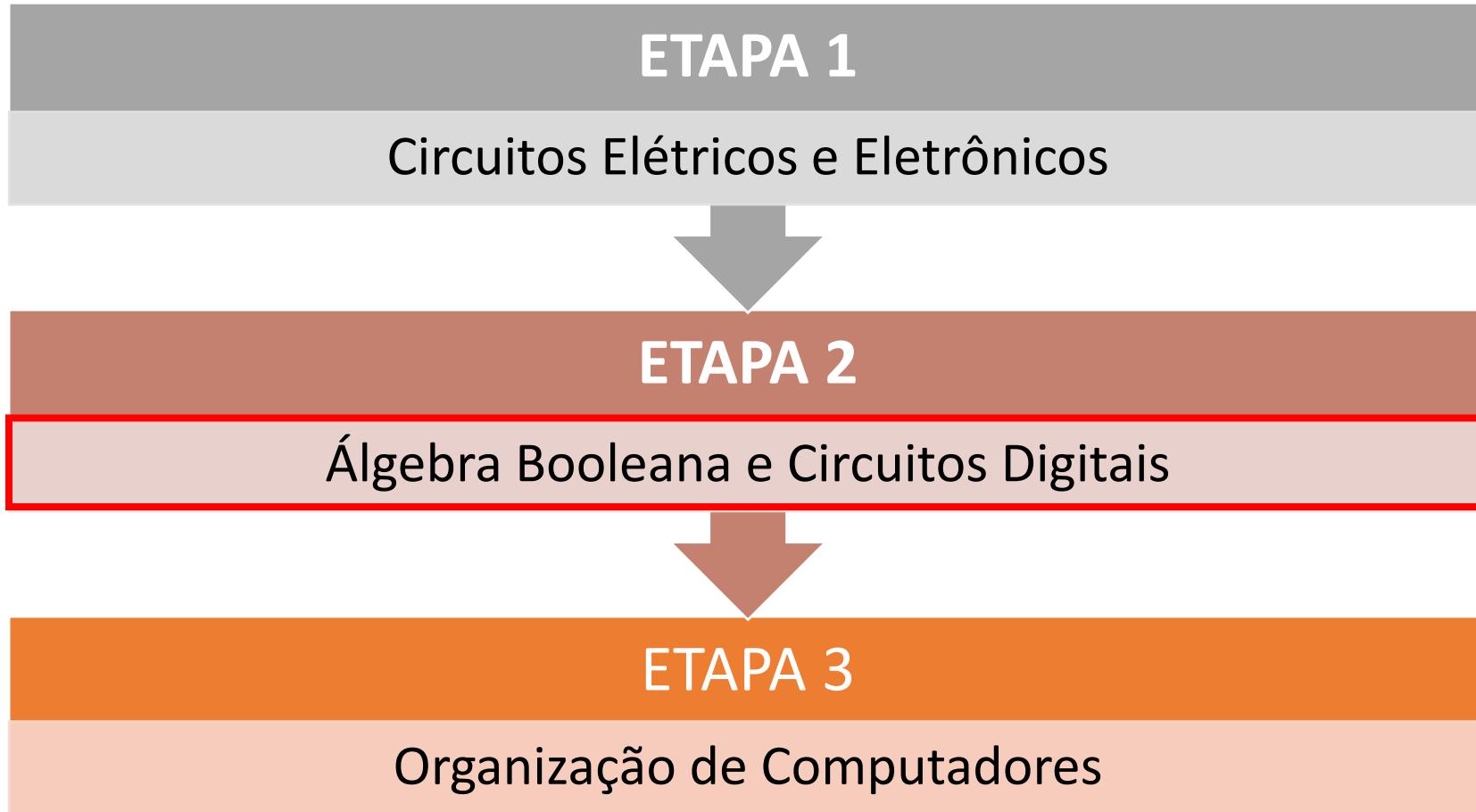
7. Somadores, subtratores e comparadores
8. Multiplexadores e demultiplexadores
9. Latches e flip-flops
10. Registradores e contadores
11. Máquinas de estado
12. Memórias

Ementa

Estudo da Álgebra Booleana, com ênfase em operações e funções binárias presentes em sistemas computacionais.

Estudo do mapeamento de operações e funções binárias em circuitos digitais como portas lógicas, circuitos sequenciais, somadores, multiplexadores, demultiplexadores, deslocadores, registradores e memórias.

Contexto da disciplina no curso



Referências

Bibliografia Básica:

TOCCI, R.J., WIDMER, N.S., MOSS, G.L. **Sistemas Digitais: Princípios e Aplicações**. 12^a ed. São Paulo: Pearson, 2018.

Bibliografia Complementar:

PIMENTA, T.C. **Circuitos Digitais**. São Paulo: Elsevier, 2017.

BIGNELL, J.W., DONOVAN, R. **Eletrônica Digital**. São Paulo: CENGAGE Learning. 2009.

FLOYD, T. **Digital Fundamentals**. New York: Pearson, 2014.

TOOLEY, M. **Electronic Circuits: Fundamentals and Applications**. 4.ed. New York: Routledge, 2015.

HUGHES, J.M. **Practical Electronics: Components and Techniques**. New York: O'Reilly Media, 2015.

SCHERZ, P., MONK, S. **Practical Electronics for Inventors**. New York: McGraw Hill, 2016.

KUMAR, A.A. **Fundamentals of Digital Circuits**. New York: Prentice Hall, 2014.

Agradecimentos

Este material é baseado nas notas de aula do **Prof. Dr. Jamil Kalil Naufal Júnior**.

Unidade 2

Sistemas de numeração

Objetivos

- Revisar o conceito de sistema de numeração posicional;
- Revisar os sistemas de numeração decimal, binário, hexadecimal e octal;
- Conhecer a representação interna e conversões típicas em sistemas computacionais;
- Praticar com representação de dados numéricos nos diversos sistemas de numeração.

Sistemas de numeração posicionais

Para processamento dos dados pelas camadas internas do Modelo de Von Neumann, eles precisam ter uma representação para armazenamento na memória do computador.

A representação convencional de dados na memória faz uso de sistemas de numeração (geralmente, o sistema binário).

Em um sistema de numeração posicional, numa determinada base numérica b , um dado numérico n representado por:

$$\pm(S_{k-1} \dots S_2 S_1 S_0, S_{-1} S_{-2} \dots S_{-l})_b$$

Terá a seguinte representação:

$$n = \pm S_{k-1} \times b^{k-1} + \dots + S_2 \times b^2 + S_1 \times b^1 + S_0 \times b^0 + S_{-1} \times b^{-1} + S_{-2} \times b^{-2} + \dots + S_{-l} \times b^{-l}$$

Sistemas de numeração posicionais

Existem quatro sistemas de numeração posicionais importantes em computação:

Sistema	Base	Símbolos	Exemplos
Binário	2	0, 1	$(1001,11)_2$
Octal	8	0, 1, 2, 3, 4, 5, 6, 7	$(156,23)_8$
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	$(2345,56)_{10}$
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	$(A2C,A1)_{16}$

Para uma determinada base b , o maior dado numérico com k dígitos que pode ser representado nesta base é dado por:

$$N_{max} = b^k - 1$$

Sistemas de numeração posicionais - exemplo

Para representar uma quantidade de **dois mil, trezentos e sessenta e quatro** elementos, usa-se a seguinte notação em cada um dos sistemas numéricos disponíveis:

Sistema binário: **100100111100₂** (12 dígitos)

Sistema octal: **4474₈** (4 dígitos)

Sistema decimal: **2364₁₀** (4 dígitos)

Sistema hexadecimal: **93C₁₆** (3 dígitos)



A vantagem de se utilizar um sistema de numeração com mais símbolos é que o número fica com menos dígitos, ou seja, mais compacto para se escrever.

Conversão de qualquer base para decimal

Para realizar a conversão de qualquer base para a base decimal, multiplica-se cada dígito com seu valor posicional no sistema original e soma-se os resultados parciais para obter o número no sistema decimal.

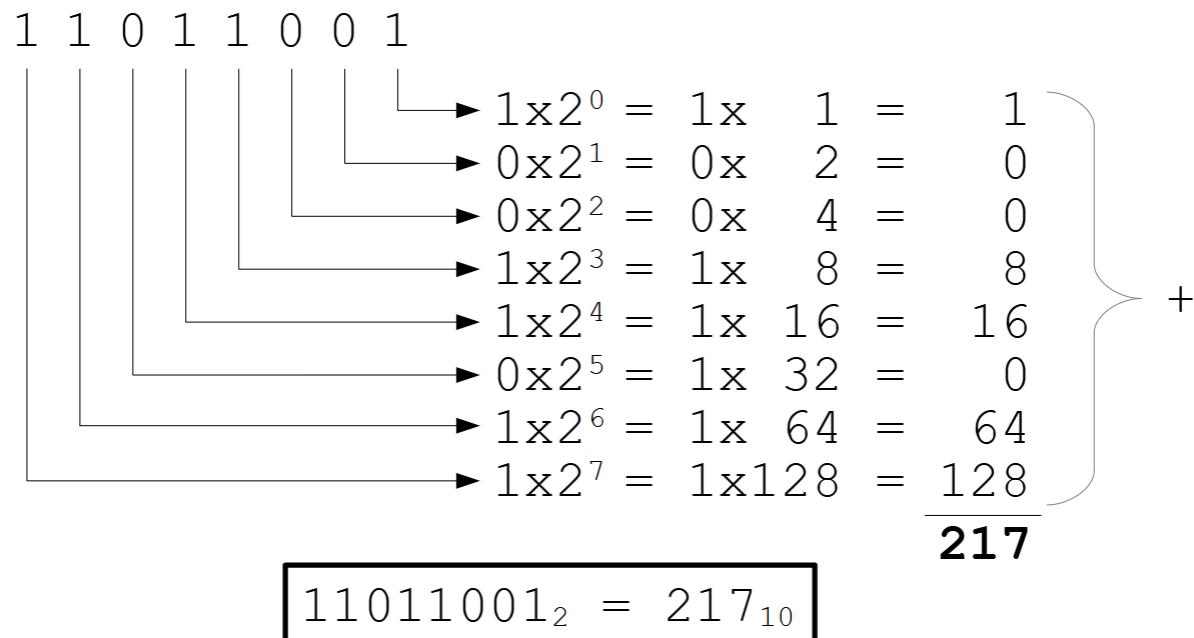
$$\begin{aligned}953,78_{10} &= 9 \times 10^2 + 5 \times 10^1 + 3 \times 10^0 + 7 \times 10^{-1} + 8 \times 10^{-2} \\&= 900 + 50 + 3 + 0,7 + 0,08 = 953,78_{10}\end{aligned}$$

$$\begin{aligned}1011,11_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \\&= 8 + 0 + 2 + 1 + 0,5 + 0,25 = 11,75_{10}\end{aligned}$$

$$\begin{aligned}A2F_{16} &= 10 \times 16^2 + 2 \times 16^1 + 15 \times 16^0 \\&= 2560 + 32 + 15 = 2607_{10}\end{aligned}$$

Conversão de binário para decimal

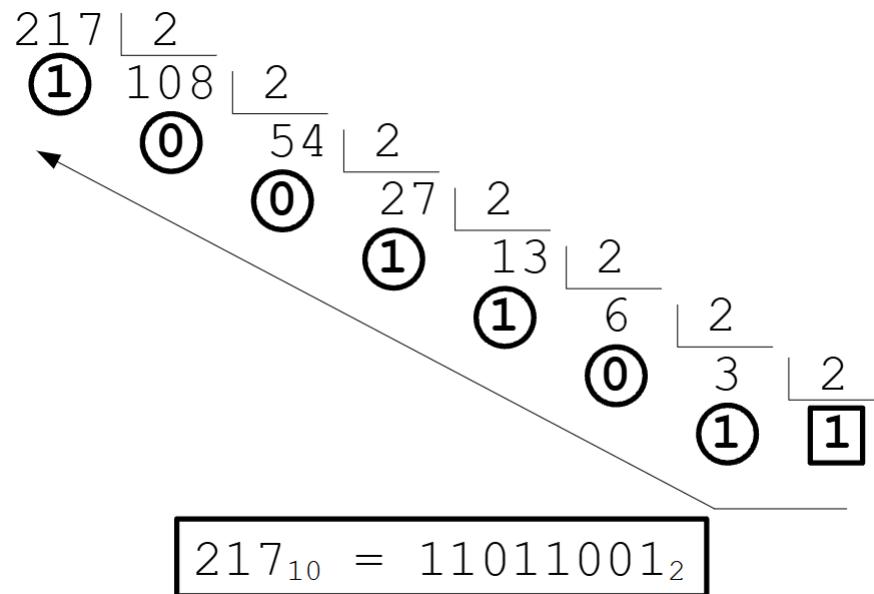
Para converter um número de binário para decimal, basta seguir o esquema abaixo:



Conversão de decimal para binário

Parte integral (ou inteira):

Para converter um número de decimal para binário, basta fazer divisões sucessivas por 2 e ao final copiar o último quociente e o demais restos de divisão na ordem inversa (de baixo para cima):



Conversão de decimal para binário

Parte fracionária:

Para converter uma fração decimal para binário, basta fazer multiplicações sucessivas por 2 até que não haja mais dígitos suficientes na base ou quando a precisão desejada já for alcançada, e ao final copiar o valor inteiro na sua ordem direta (de cima para baixo):

$$\begin{aligned}0,0625 \times 2 &= 0,1250 \\0,1250 \times 2 &= 0,2500 \\0,2500 \times 2 &= 0,5000 \\0,5000 \times 2 &= 1,0000\end{aligned}$$

$$\begin{aligned}0,375 \times 2 &= 0,750 \\0,750 \times 2 &= 1,500 \\0,500 \times 2 &= 1,000\end{aligned}$$

$$0,0625_{10} = 0,0001_2$$

$$0,375_{10} = 0,011_2$$

Conversão de decimal para binário

Parte fracionária com dízima periódica:

$$\begin{array}{rcl} 0,8 \times 2 = & | & 1,6 \\ 0,6 \times 2 = & | & 1,2 \\ 0,2 \times 2 = & | & 0,4 \\ 0,4 \times 2 = & | & 0,8 \\ 0,8 \times 2 = & | & 1,6 \\ 0,6 \times 2 = & | & 1,2 \\ & \dots & \end{array}$$

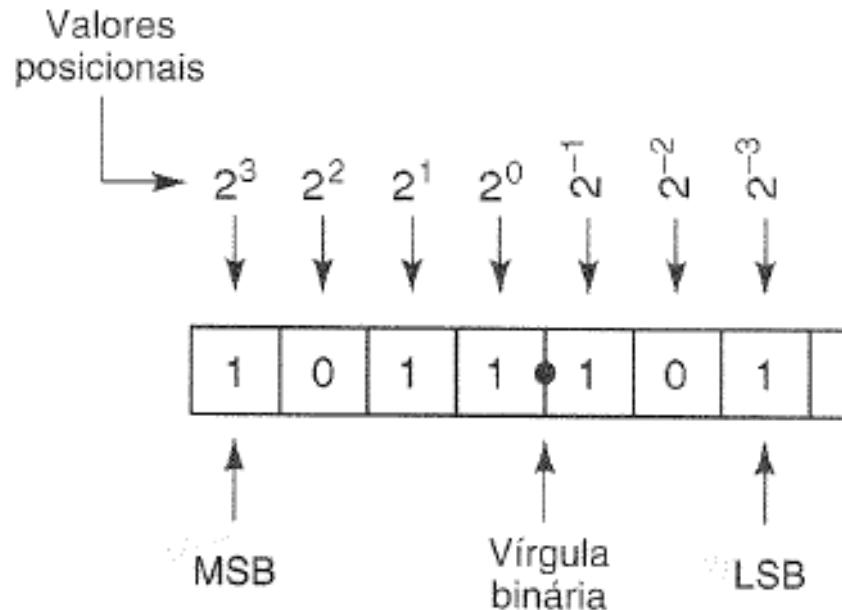
$$0,8_{10} = 0,11001100\ldots_2$$

Sistema binário

Sistema binário possui apenas dois símbolos ou valores para dígitos: 0 e 1.

Mais fácil projetar equipamentos eletrônicos precisos somente com dois níveis de tensão.

A menor unidade no sistema binário é denominado de bit (binary digit).



MSB: Most Significant Bit

LSB: Least Significant Bit

Sequência de contagem binária

A tabela a seguir apresenta a montagem e conversão entre números binários e seus valores equivalentes em decimal.

A maioria dos microcomputadores manipulam e armazenam dados binários e informações em grupos de 8 bits.

Este grupo de 8 bits torna-se uma unidade básica em computação denominado de byte.

Uma unidade inferior também é definida como **nibble** ou **nyble** como um conjunto de 4 bits.

Pesos →	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	Decimal equivalente
	0	0	0	0	0
	0	0	0	1	1
	0	0	1	0	2
	0	0	1	1	3
	0	1	0	0	4
	0	1	0	1	5
	0	1	1	0	6
	0	1	1	1	7
	1	0	0	0	8
	1	0	0	1	9
	1	0	1	0	10
	1	0	1	1	11
	1	1	0	0	12
	1	1	0	1	13
	1	1	1	0	14
	1	1	1	1	15

↑ MSB ↑ LSB

Conversão octal-binário

A vantagem do sistema octal é a sua facilidade de conversão entre binários e octais.

Dígito Octal	0	1	2	3	4	5	6	7
Equivalente Binário	000	001	010	011	100	101	110	111



Cada dígito octal equivale a três dígitos binários.

Conversão hexadecimal-binário

Assim como o octal o sistema hexadecimal é um método compacto para a representação de números binários.

HEXADECIMAL	DECIMAL	BINÁRIO
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
<i>continua...</i>		

HEXADECIMAL	DECIMAL	BINÁRIO
<i>... continuação</i>		
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111



Cada dígito hexadecimal equivale a quatro dígitos binários.

Unidades de medida

Diferente dos seres humanos, que usam um sistema numérico baseado em dez símbolos (sistema decimal) para representar quantidades, o computador usa um sistema numérico baseado em dois símbolos, 0 e 1, que recebe o nome de sistema binário.

Um conjunto de 8 dígitos binários, ou bits, formam 1 byte, que é usado pelo computador para representar 1 caracter, que pode ser uma letra, um número, um símbolo gráfico, etc.

$$\text{8 bits} = \text{1 byte} = \text{1 caracter}$$

Assim, se quisermos armazenar a palavra **DIGITAL** na memória de um computador, usaremos 7 bytes, pois esta palavra possui sete caracteres.

Unidades de medida - múltiplos

Sendo então o byte a unidade de medida principal para armazenamento de dados em um computador, tem-se os seguintes múltiplos de acordo com o Sistema Internacional de Unidades:

1.000 bytes = 1 kilobyte ou 1 kB

1.000 kilobytes = 1 megabyte ou 1 MB

1.000 megabytes = 1 gigabyte ou 1 GB

1.000 gigabytes = 1 terabyte ou 1 TB

1.000 terabytes = 1 petabyte ou 1 PB

1.000 petabytes = 1 exabyte ou 1 EB

1.000 exabytes = 1 zettabyte ou 1 ZB

1.000 zettabytes = 1 yottabyte ou 1 YB

Unidades de medida - exemplos

Capacidade de armazenamento de algumas mídias tradicionais:

Disquete 3 ½" = 1,44 MB ou 0,00144 GB



CD = 700 MB ou 0,7 GB

DVD = de 4.700 a 8.500 MB, ou de 4,7 a 8,5 GB

Blu-ray = de 25.000 a 50.000 MB, ou de 25 GB a 50 GB

LTO-9 = 18.000.000 MB, ou 18.000 GB ou 18 TB



Disquetes, Compact Disc (CD) e Digital Versatile Disc (DVD) são consideradas mídias obsoletas. LTO (Linear Tape-Open) é utilizada para backup de dados.

Unidades de medida - SI vs IEC

Os prefixos kilo (k), mega (M), giga (G) etc, são usados pelo Sistema Internacional de Unidades e são baseados no sistema decimal. Como o computador usa o sistema binário, estes prefixos não indicam a capacidade correta de armazenamento. Assim, a Comissão Eletrotécnica Internacional (IEC) estabeleceu os seguintes múltiplos:

1.024 bytes = 1 kibibyte ou 1 KiB

1.024 kibibytes = 1 mebibyte ou 1 MiB

1.024 mebibytes = 1 gibibyte ou 1 GiB

1.024 gibibytes = 1 tebibyte ou 1 TiB

1.024 tebibytes = 1 pebibyte ou 1 PiB

1.024 pebibytes = 1 exbibyte ou 1 EiB

1.024 exbibytes = 1 zebibyte ou 1 ZiB

1.024 zebibytes = 1 yobibyte ou 1 YiB

Unidades de medida - SI

Unidades de medida de acordo com o Sistema Internacional de Unidades:

Prefixo decimal - SI				
Nome	Símbolo	Múltiplo	Quantidade de bytes	
byte	B	10^0		1
kilobyte	kB	10^3		1.000
megabyte	MB	10^6		1.000.000
gigabyte	GB	10^9		1.000.000.000
terabyte	TB	10^{12}		1.000.000.000.000
petabyte	PB	10^{15}		1.000.000.000.000.000
exabyte	EB	10^{18}		1.000.000.000.000.000.000
zettabyte	ZB	10^{21}		1.000.000.000.000.000.000.000
yottabyte	YB	10^{24}		1.000.000.000.000.000.000.000.000

Unidades de medida - IEC

Unidades de medida de acordo com a Comissão Eletrotécnica Internacional:

Prefixo binário - IEC			
Nome	Símbolo	Múltiplo	Quantidade de bytes
byte	B	2^0	1
kibibyte	KiB	2^{10}	1.024
mebibyte	MiB	2^{20}	1.048.576
gibibyte	GiB	2^{30}	1.073.741.824
tebibyte	TiB	2^{40}	1.099.511.627.776
pebibyte	PiB	2^{50}	1.125.899.906.842.620
exbibyte	EiB	2^{60}	1.152.921.504.606.850.000
zebibyte	ZiB	2^{70}	1.180.591.620.717.410.000.000
yobibyte	YiB	2^{80}	1.208.925.819.614.630.000.000.000

Endianness

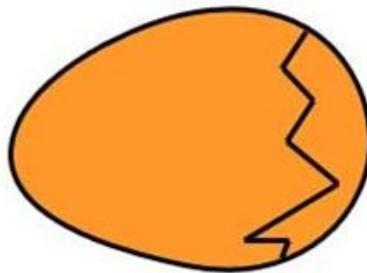
Quando se utilizam múltiplos bytes para interpretar uma dada informação (**endianness**), os diferentes fabricantes de CPUs podem adotar duas formas básicas de interpretação de dados:

- **Big-endian**, quando o byte mais significante (Most Significant Byte ou MSB) está alocado no primeiro byte de memória; ou
- **Little-endian**, quando o byte menos significante (Less Significant Byte ou LSB) está alocado no primeiro byte de memória.

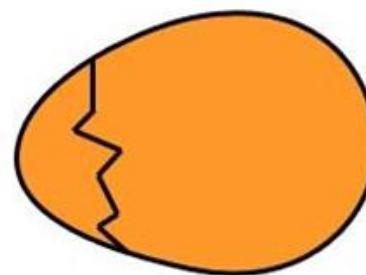
Por exemplo, o fabricante Intel adota little-endian em seus processadores, enquanto o fabricante Motorola adota big-endian.

Endianness

Os termos big-endian e little-endian foram introduzidos por Danny Cohen em 1980 na Internet Engineering Note 137, um memorando intitulado “On Holy Wars and a Plea for Peace”, posteriormente publicado em formato impresso no IEEE Computer 14(10).48 -57 (1981).



BIG ENDIAN



LITTLE ENDIAN

Ele os emprestou de Jonathan Swift, que em Gulliver's Travels (1726) os usou para descrever as posições opostas de duas facções na nação de Líliput. Os Big-Endians, que quebravam seus ovos cozidos na ponta grande, se rebelaram contra o rei, que exigia que seus súditos quebrassem os ovos na ponta menor.

Fonte: ling.upenn.edu

Big-Endian vs Little-Endian - exemplo 1

No exemplo a seguir, uma variável inteira i com valor decimal de **450**, utilizando 32 bits de tamanho, pode ser armazenada de duas formas:

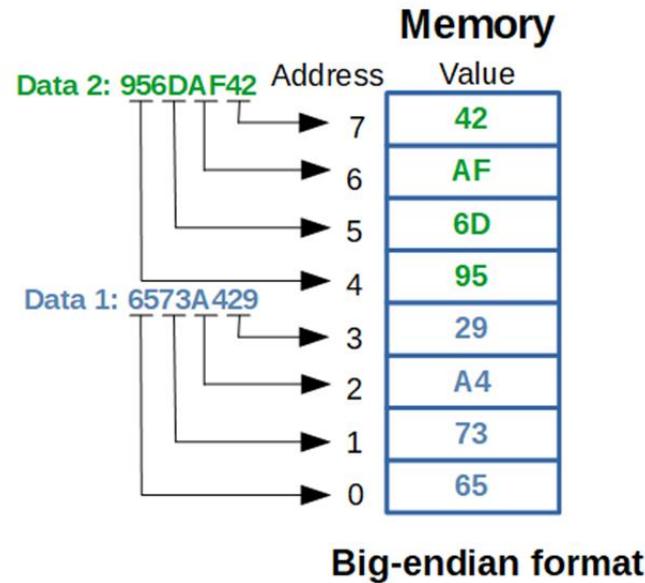
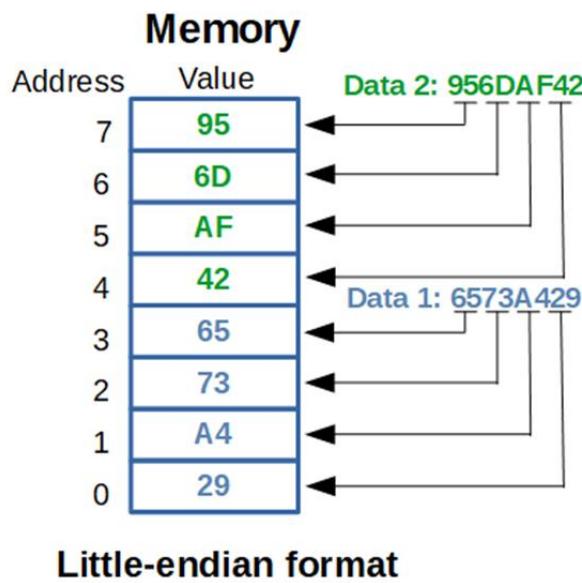
int i = 450 = 0x000001C2

LSB	Little endian		
	11000010	00000001	00000000 00000000
C2	01	00	00
lower	address → higher		
MSB	Big endian		
	00000000 00000000	00000001	11000010
00	00	01	C2

! Quando for o caso, apenas a ordem dos bytes é invertida, e não a dos bits.

Big-Endian vs Little-Endian - exemplo 2

No exemplo a seguir, dois inteiros com valores **0x6573A429** e **0x956DAF42** utilizando 32 bits de tamanho, podem ser armazenados de duas formas:



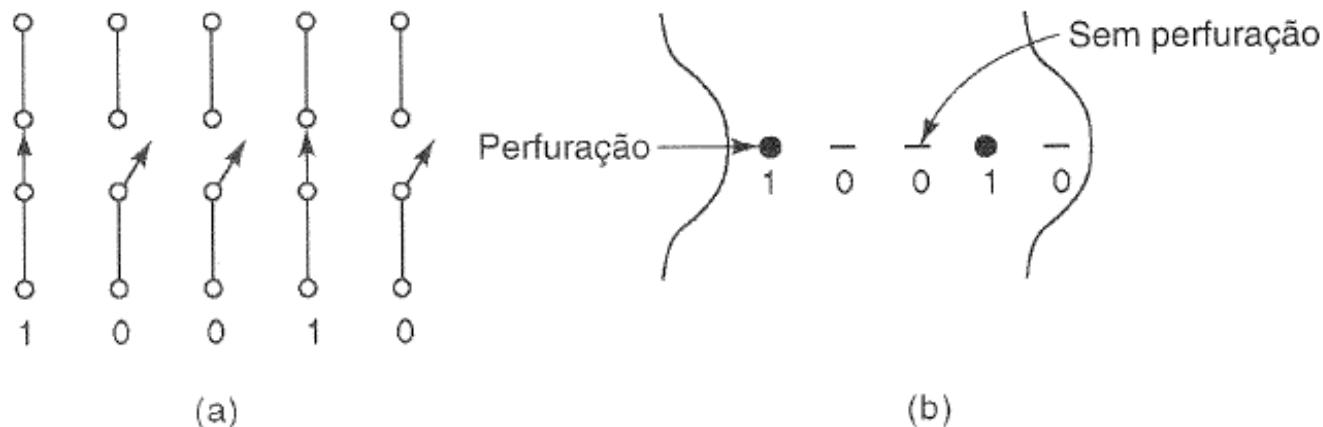
A inversão dos bytes ocorre apenas dentro do agrupamento de bits.

Fonte: open4tech.com

Representação de quantidades binárias em aplicações

Em sistemas digitais a informação sendo processada é representada de forma binária por um dispositivo eletrônico que representa apenas dois estados de operação possíveis.

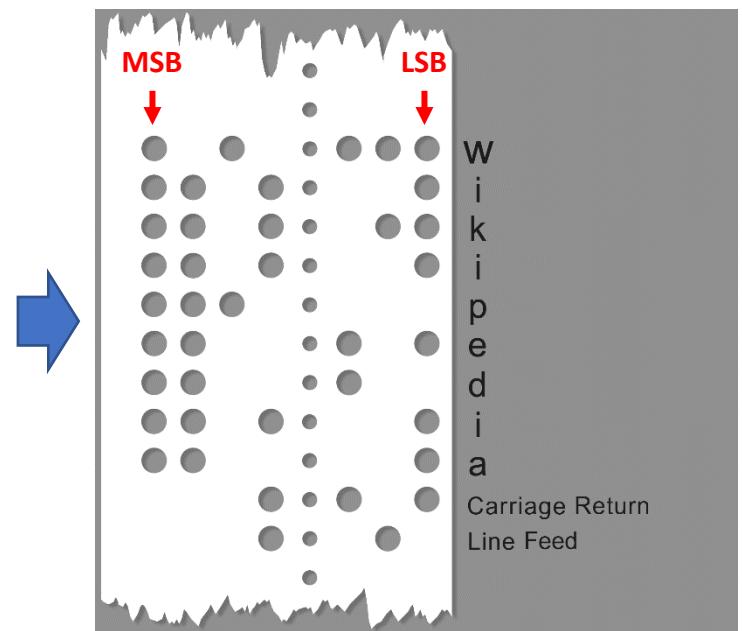
Exemplos: cartão não perfurado (nível binário 0) e perfurado (nível binário 1). Válvula aberta (nível binário zero) e fechado (nível binário 1).



Representação de números binários utilizando
(a) chaves e (b) perfurações em uma fita de papel.

Representação de quantidades binárias em aplicações

Exemplo – fita perfurada:



A codificação da fita perfurada ao lado pode ser verificada utilizando-se a tabela ASCII.

Fonte: wikipedia.org

Representação de quantidades binárias em aplicações

Exemplo – cartão perfurado:

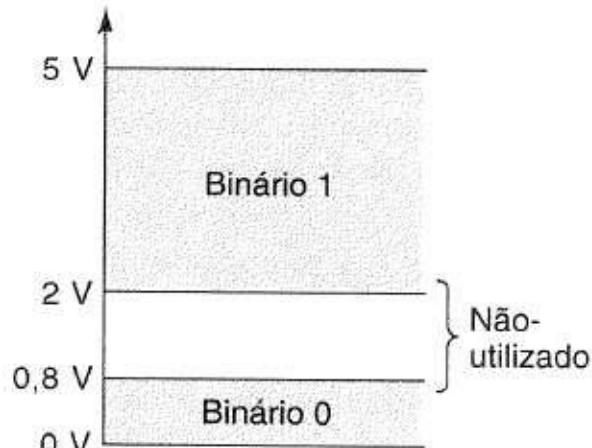


Você pode criar um cartão personalizado em <https://www.masswerk.at/keypunch/>

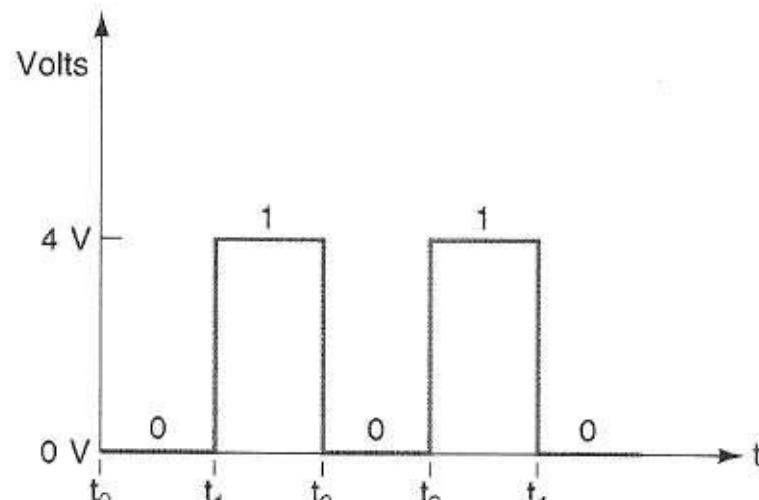
Fonte: www.masswerk.at/keypunch/

Representação de quantidades binárias em circuitos digitais eletrônicos

A representação de binária em circuitos digitais (CIs – Circuitos Integrados) é associada aos níveis de tensão em Volts típicas a seguir.



(a)

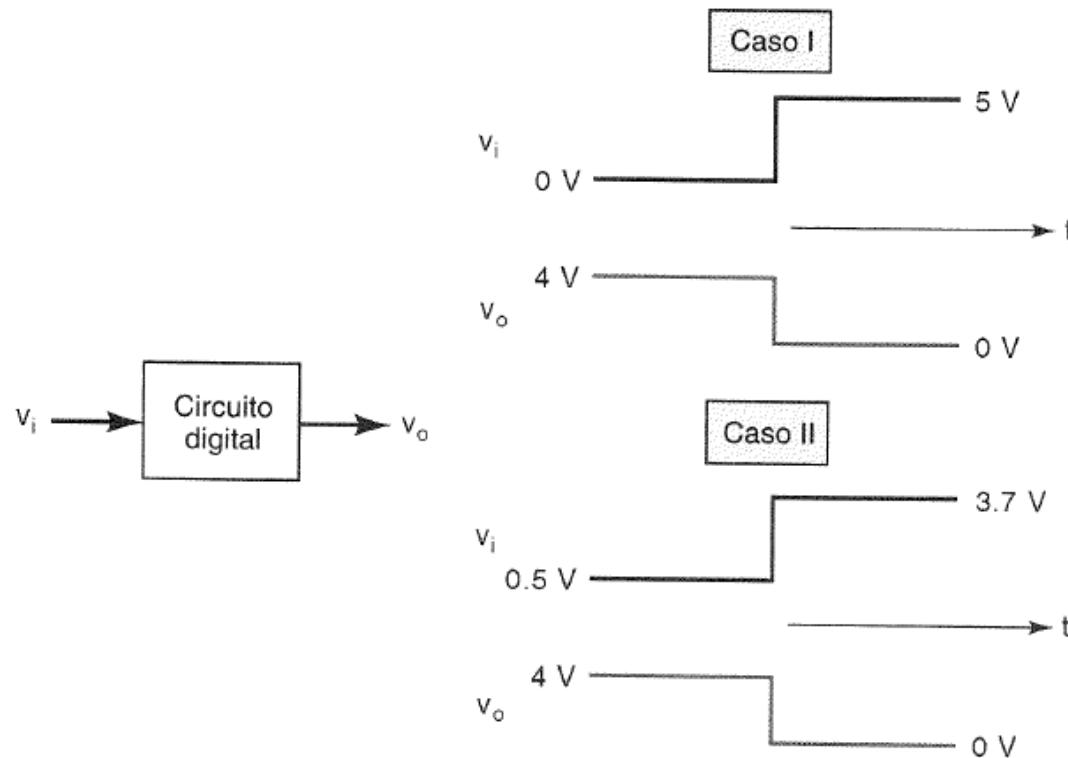


(b)

- (a) Indicação de intervalos de tensão típicos para binários 0 e 1
(b) típico diagrama de tempo de um sinal digital.

Representação de quantidades binárias em circuitos digitais eletrônicos

Um circuito digital respondendo a um nível digital exato (CASO I) e não exato (CASO II).



Codificação

Quando números, letras ou palavras são representados por um grupo especial de símbolos, diz-se que estão codificados e o grupo de símbolos é chamado de código.

Exemplo: código Morse no qual um conjunto de traços e pontos representam letras do alfabeto.

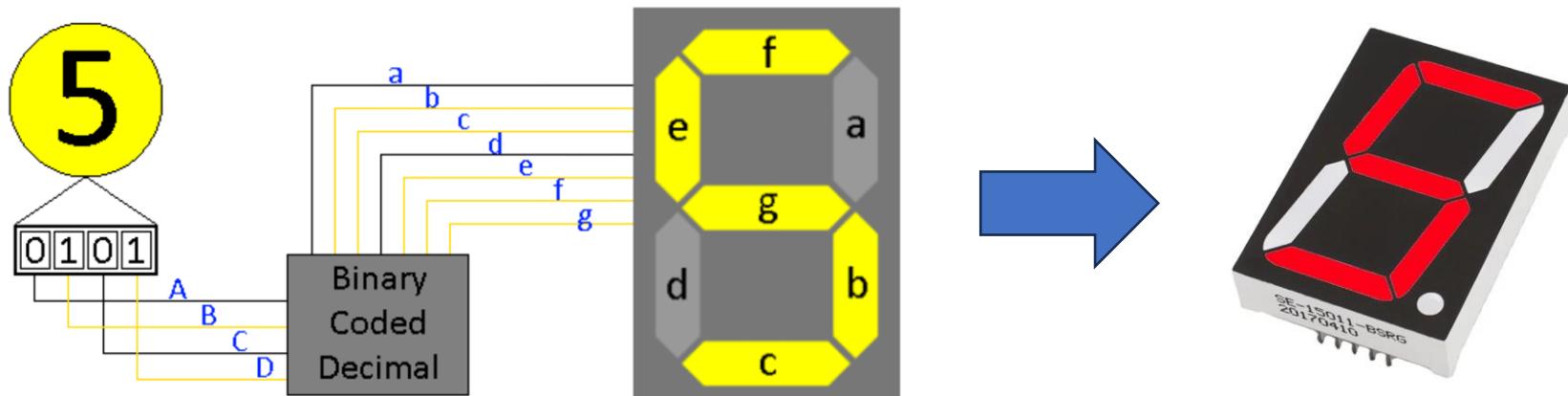
Quando um número decimal é representado por um número binário equivalente, denomina-se codificação binária pura.

Exemplo: 10 decimal em 1010 binário

Código decimal codificado em binário (BCD)

Em muitas aplicações em que a saída é um display digital (contadores de frequência, voltímetros digitais, calculadoras etc) são usados códigos de representação decimal. Um dos códigos mais conhecidos que realiza esta conversão de binário para decimal é através do código BCD.

Cada dígito de um número decimal é representado por seu equivalente em binário abreviado denominado de BCD (Binary-Coded-Decimal). Nele são necessários 4 bits para representar os símbolos decimais.



Fonte: ilustração de Tracy Gilmore, disponível em dev.to

Relação entre representações

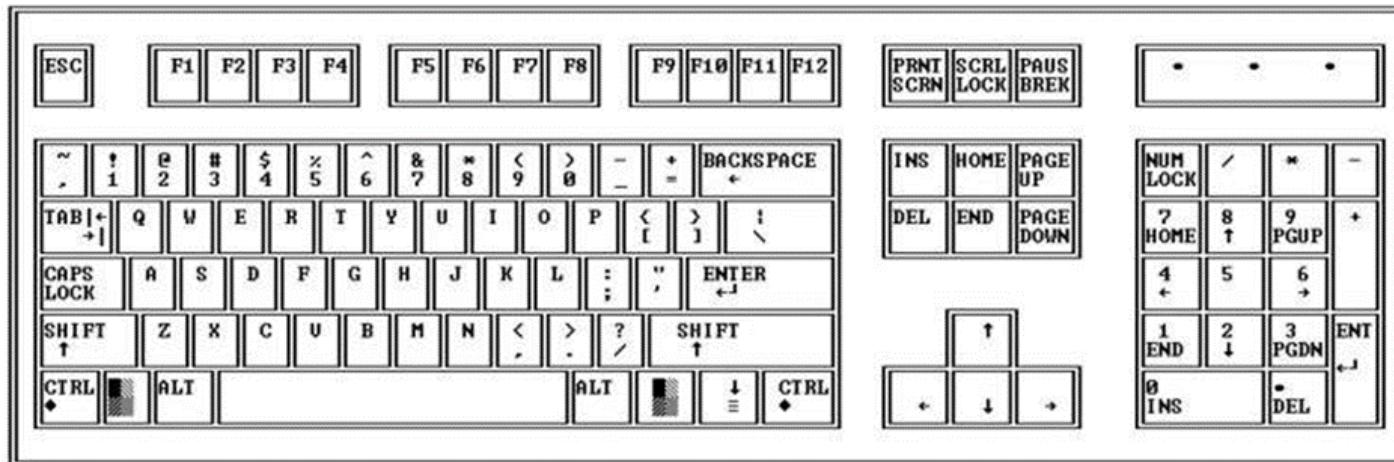
Decimal	Binário	Octal	Hexadecimal	BCD
0	0	0	0	0000
1	1	1	1	0001
2	10	2	2	0010
3	11	3	3	0011
4	100	4	4	0100
5	101	5	5	0101
6	110	6	6	0110
7	111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	0001 0000
11	1011	13	B	0001 0001
12	1100	14	C	0001 0010
13	1101	15	D	0001 0011
14	1110	16	E	0001 0100
15	1111	17	F	0001 0101

Código ASCII

O American Standard Code for Information Interchange (ASCII) é um código alfanumérico amplamente utilizado que usa 7 bits, permitindo assim 128 codificações possíveis.

Esta quantidade de codificações é suficiente para representar todos os caracteres de um teclado padrão e funções de controle.

Codifica 95 sinais gráficos (letras do alfabeto, sinais de pontuação e sinais matemáticos) e 33 sinais de controle, tais como <RETURN> e <LINEFEED>.



Fonte: deviantart.com

Código ASCII

Como a unidade básica de armazenamento de dados dos computadores modernos é o byte (8 bits), então a codificação ASCII define que cada símbolo é armazenado em um byte, sendo que o primeiro bit dos 8 disponíveis é ignorado (sempre em zero).

Este primeiro bit pode ser usado para checar, por exemplo, a integridade de uma sequência de caracteres transmitidos. Caso o bit não seja zero, então algo provavelmente chegou errado.



Posteriormente, este bit adicional permitiu a extensão do código ASCII de 128 para 256 codificações possíveis.

Tabela ASCII

ASCII control characters			ASCII printable characters				Extended ASCII characters					
00	NULL	(Null character)	32	space	64	@	96	'	128	ç	160	á
01	SOH	(Start of Header)	33	!	65	A	97	a	129	ü	161	í
02	STX	(Start of Text)	34	"	66	B	98	b	130	é	162	ó
03	ETX	(End of Text)	35	#	67	C	99	c	131	â	163	ú
04	EOT	(End of Trans.)	36	\$	68	D	100	d	132	à	164	ñ
05	ENQ	(Enquiry)	37	%	69	E	101	e	133	à	165	Ñ
06	ACK	(Acknowledgement)	38	&	70	F	102	f	134	à	166	ª
07	BEL	(Bell)	39	.	71	G	103	g	135	ç	167	º
08	BS	(Backspace)	40	(72	H	104	h	136	ê	168	¿
09	HT	(Horizontal Tab)	41)	73	I	105	i	137	è	169	®
10	LF	(Line feed)	42	*	74	J	106	j	138	è	170	¬
11	VT	(Vertical Tab)	43	+	75	K	107	k	139	í	171	½
12	FF	(Form feed)	44	,	76	L	108	l	140	í	172	¼
13	CR	(Carriage return)	45	-	77	M	109	m	141	ì	173	¡
14	SO	(Shift Out)	46	.	78	N	110	n	142	Ä	174	«
15	SI	(Shift In)	47	/	79	O	111	o	143	À	175	»
16	DLE	(Data link escape)	48	0	80	P	112	p	144	É	176	¤
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177	¤
18	DC2	(Device control 2)	50	2	82	R	114	r	146	Æ	178	¤
19	DC3	(Device control 3)	51	3	83	S	115	s	147	ô	179	¤
20	DC4	(Device control 4)	52	4	84	T	116	t	148	ö	180	¤
21	NAK	(Negative acknowl.)	53	5	85	U	117	u	149	ò	181	À
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	û	182	Â
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	ù	183	À
24	CAN	(Cancel)	56	8	88	X	120	x	152	ÿ	184	©
25	EM	(End of medium)	57	9	89	Y	121	y	153	Ö	185	†
26	SUB	(Substitute)	58	:	90	Z	122	z	154	Ü	186	
27	ESC	(Escape)	59	;	91	[123	{	155	ø	187	¶
28	FS	(File separator)	60	<	92	\	124		156	£	188	¤
29	GS	(Group separator)	61	=	93]	125	}	157	Ø	189	¢
30	RS	(Record separator)	62	>	94	^	126	~	158	×	190	¥
31	US	(Unit separator)	63	?	95	-			159	f	191	„
127	DEL	(Delete)			96	-			192	„	223	„
					97	-			224	Ó	255	nbsp

Fonte: theasciicode.com.ar

Armazenamento binário em um computador

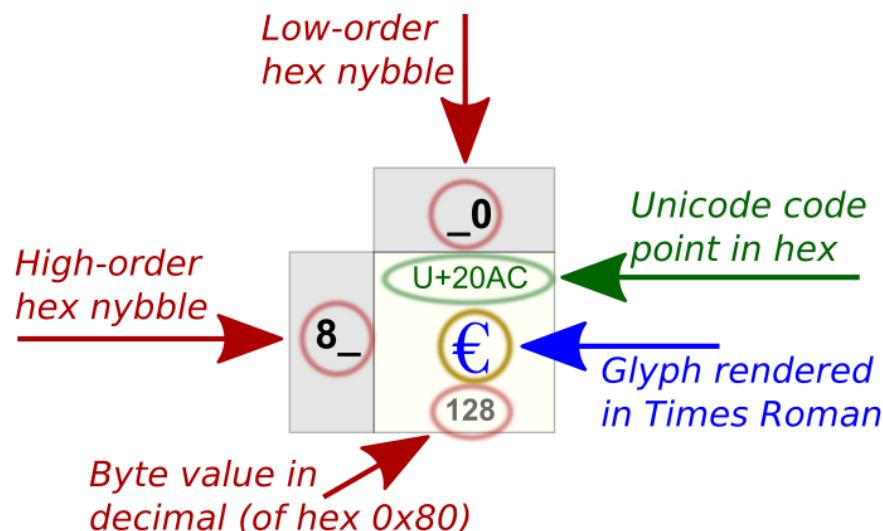
O código ASCII pode ser usado para a transferência de informações entre computador e dispositivos de E/S (terminais de vídeo e impressoras).

O computador utiliza também o código ASCII internamente para armazenar informação digitada pelo teclado.

Código ISO-8859-1

A codificação ASCII é muito útil para a língua inglesa que não acentua palavras. Para a realidade brasileira, ASCII não é muito amigável. A codificação Latin-1 (ISO-8859-1) é uma codificação parecida com a ASCII, mas define apenas caracteres imprimíveis, logo, é uma codificação com foco na grafia das pessoas e não no armazenamento em computadores.

A codificação Latin-1 (ISO-8859-1) possui 191 símbolos e pode ser representada em um byte.



Código ASCII

ASCII

	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>
0_	U+0000 NUL 0	U+0001 SOH 1	U+0002 STX 2	U+0003 ETX 3	U+0004 EOT 4	U+0005 ENQ 5	U+0006 ACK 6	U+0007 BEL 7	U+0008 BS 8	U+0009 HT 9	U+000A LF 10	U+000B VT 11	U+000C FF 12	U+000D CR 13	U+000E SO 14	U+000F SI 15
1_	U+0010 DLE 16	U+0011 DC1 17	U+0012 DC2 18	U+0013 DC3 19	U+0014 DC4 20	U+0015 NAK 21	U+0016 SYN 22	U+0017 ETB 23	U+0018 CAN 24	U+0019 EM 25	U+001A SUB 26	U+001B ESC 27	U+001C FS 28	U+001D GS 29	U+001E RS 30	U+001F US 31
2_	U+0020 SP 32	U+0021 ! 33	U+0022 " 34	U+0023 # 35	U+0024 \$ 36	U+0025 % 37	U+0026 & 38	U+0027 ' 39	U+0028 (40	U+0029) 41	U+002A * 42	U+002B + 43	U+002C ,44	U+002D - 45	U+002E .br/>46	U+002F / 47
3_	U+0030 0 48	U+0031 1 49	U+0032 2 50	U+0033 3 51	U+0034 4 52	U+0035 5 53	U+0036 6 54	U+0037 7 55	U+0038 8 56	U+0039 9 57	U+003A : 58	U+003B ; 59	U+003C < 60	U+003D = 61	U+003E > 62	U+003F ? 63
4_	U+0040 @ 64	U+0041 A 65	U+0042 B 66	U+0043 C 67	U+0044 D 68	U+0045 E 69	U+0046 F 70	U+0047 G 71	U+0048 H 72	U+0049 I 73	U+004A J 74	U+004B K 75	U+004C L 76	U+004D M 77	U+004E N 78	U+004F O 79
5_	U+0050 P 80	U+0051 Q 81	U+0052 R 82	U+0053 S 83	U+0054 T 84	U+0055 U 85	U+0056 V 86	U+0057 W 87	U+0058 X 88	U+0059 Y 89	U+005A Z 90	U+005B [91	U+005C] 92	U+005D ^ 93	U+005E — 94	U+005F _ 95
6_	U+0060 ` 96	U+0061 a 97	U+0062 b 98	U+0063 c 99	U+0064 d 100	U+0065 e 101	U+0066 f 102	U+0067 g 103	U+0068 h 104	U+0069 i 105	U+006A j 106	U+006B k 107	U+006C l 108	U+006D m 109	U+006E n 110	U+006F o 111
7_	U+0070 p 112	U+0071 q 113	U+0072 r 114	U+0073 s 115	U+0074 t 116	U+0075 u 117	U+0076 v 118	U+0077 w 119	U+0078 x 120	U+0079 y 121	U+007A z 122	U+007B { 123	U+007C } 124	U+007D ~ 125	U+007E DEL 126	U+007F 127

Fonte: gammon.com.au

Código ISO-8859-1 (Latin 1)

ISO-8859-1

	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>
8_	U+0080 PAD 128	U+0081 HOP 129	U+0082 BPH 130	U+0083 NBH 131	U+0084 IND 132	U+0085 NEL 133	U+0086 SSA 134	U+0087 ESA 135	U+0088 HTS 136	U+0089 HTJ 137	U+008A LTS 138	U+008B PLD 139	U+008C PLU 140	U+008D RI 141	U+008E SS2 142	U+008F SS3 143
9_	U+0090 DCS 144	U+0091 PU1 145	U+0092 PU2 146	U+0093 STS 147	U+0094 CCH 148	U+0095 MW 149	U+0096 SPA 150	U+0097 EPA 151	U+0098 SOS 152	U+0099 SGCI 153	U+009A SCI 154	U+009B CSI 155	U+009C ST 156	U+009D OSC 157	U+009E PM 158	U+009F APC 159
A_	U+00A0 NBSP 160	U+00A1 i 161	U+00A2 ¢ 162	U+00A3 £ 163	U+00A4 ¤ 164	U+00A5 ¥ 165	U+00A6 ₩ 166	U+00A7 ₪ 167	U+00A8 „ 168	U+00A9 „ 169	U+00AA „ 170	U+00AB „ 171	U+00AC „ 172	U+00AD „ 173	U+00AE ® 174	U+00AF – 175
B_	U+00B0 o 176	U+00B1 ± 177	U+00B2 2 178	U+00B3 3 179	U+00B4 ‘ 180	U+00B5 µ 181	U+00B6 ¶ 182	U+00B7 .br 183	U+00B8 „ 184	U+00B9 1 185	U+00BA º 186	U+00BB » 187	U+00BC ¼ 188	U+00BD ½ 189	U+00BE ¾ 190	U+00BF đ 191
C_	U+00C0 À 192	U+00C1 Á 193	U+00C2 Â 194	U+00C3 Ã 195	U+00C4 Ä 196	U+00C5 Å 197	U+00C6 Æ 198	U+00C7 Ç 199	U+00C8 È 200	U+00C9 É 201	U+00CA Ê 202	U+00CB Ë 203	U+00CC Ì 204	U+00CD Í 205	U+00CE Î 206	U+00CF Ï 207
D_	U+00D0 Đ 208	U+00D1 Ñ 209	U+00D2 Ò 210	U+00D3 Ó 211	U+00D4 Ô 212	U+00D5 Õ 213	U+00D6 Ö 214	U+00D7 × 215	U+00D8 Ø 216	U+00D9 Ù 217	U+00DA Ú 218	U+00DB Û 219	U+00DC Ü 220	U+00DD Ý 221	U+00DE Þ 222	U+00DF ß 223
E_	U+00E0 à 224	U+00E1 á 225	U+00E2 â 226	U+00E3 ã 227	U+00E4 ä 228	U+00E5 å 229	U+00E6 æ 230	U+00E7 ç 231	U+00E8 è 232	U+00E9 é 233	U+00EA ê 234	U+00EB ë 235	U+00EC ì 236	U+00ED í 237	U+00EE î 238	U+00EF ï 239
F_	U+00F0 đ 240	U+00F1 ñ 241	U+00F2 ò 242	U+00F3 ó 243	U+00F4 ô 244	U+00F5 õ 245	U+00F6 ö 246	U+00F7 ÷ 247	U+00F8 ø 248	U+00F9 ù 249	U+00FA ú 250	U+00FB û 251	U+00FC ü 252	U+00FD ý 253	U+00FE þ 254	U+00FF ÿ 255

Fonte: gammon.com.au

Código ISO-8859-7 (Latino/Grego)

iso-8859-7

	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>
8_	U+0080 PAD 128	U+0081 HOP 129	U+0082 BPH 130	U+0083 NBH 131	U+0084 IND 132	U+0085 NEL 133	U+0086 SSA 134	U+0087 ESA 135	U+0088 HTS 136	U+0089 HTJ 137	U+008A LTS 138	U+008B PLD 139	U+008C PLU 140	U+008D RI 141	U+008E SS2 142	U+008F SS3 143
9_	U+0090 DCS 144	U+0091 PU1 145	U+0092 PU2 146	U+0093 STS 147	U+0094 CCH 148	U+0095 MW 149	U+0096 SPA 150	U+0097 EPA 151	U+0098 SOS 152	U+0099 SGCI 153	U+009A SCI 154	U+009B CSI 155	U+009C ST 156	U+009D OSC 157	U+009E PM 158	U+009F APC 159
A_	U+00A0 NBSP 160	U+2018 ‘ 161	U+2019 ’ 162	U+00A3 £ 163	U+20AC € 164	U+20AF đ 165	U+00A6 — 166	U+00A7 § 167	U+00A8 .. 168	U+00A9 © 169	U+00A8 .. 170	U+037A „ 171	U+00AB „ 172	U+00AD — 173		U+2015 — 174
B_	U+00B0 ° 176	U+00B1 ± 177	U+00B2 ² 178	U+00B3 ³ 179	U+0384 ‘ 180	U+0385 ’ 181	U+0386 ‘ 182	U+00B7 · 183	U+0388 · 184	U+0389 · 185	U+038A · 186	U+00BB » 187	U+038C · 188	U+00BD ½ 189	U+038E · 190	U+038F Ω 191
C_	U+0390 č 192	U+0391 á 193	U+0392 à 194	U+0393 ã 195	U+0394 Δ 196	U+0395 é 197	U+0396 ž 198	U+0397 h 199	U+0398 θ 200	U+0399 í 201	U+039A í 202	U+039B k 203	U+039C λ 204	U+039D m 205	U+039E ñ 206	U+039F ó 207
D_	U+03A0 Π 208	U+03A1 P 209		U+03A3 Σ 210	U+03A4 Τ 211	U+03A5 Υ 212	U+03A6 Φ 213	U+03A7 Χ 214	U+03A8 Ψ 215	U+03A9 Ω 216	U+03AA Ϊ 217	U+03AB Ϋ 218	U+03AC ά 219	U+03AD έ 220	U+03AE ή 221	U+03AF ί 223
E_	U+03B0 ő 224	U+03B1 α 225	U+03B2 β 226	U+03B3 γ 227	U+03B4 δ 228	U+03B5 ε 229	U+03B6 ζ 230	U+03B7 η 231	U+03B8 θ 232	U+03B9 ι 233	U+03BA υ 234	U+03BB κ 235	U+03BC λ 236	U+03BD ν 237	U+03BE ξ 238	U+03BF ο 239
F_	U+03C0 π 240	U+03C1 ρ 241	U+03C2 ς 242	U+03C3 σ 243	U+03C4 τ 244	U+03C5 υ 245	U+03C6 φ 246	U+03C7 χ 247	U+03C8 ψ 248	U+03C9 ω 249	U+03CA ϊ 250	U+03CB ö 251	U+03CC ó 252	U+03CD ú 253	U+03CE ó 254	255

Fonte: gammon.com.au

Alfabetos ISO 8859

ISO 8859-1 - línguas da Europa Ocidental (Latin-1)

ISO 8859-2 - línguas da Europa Oriental (Latin-2)

ISO 8859-3 - línguas do sudeste da Europa e miscelâneas (Latin-3)

ISO 8859-4 - línguas escandinavas/bálticas (Latin-4)

ISO 8859-5 - latino/cirílico

ISO 8859-6 - latino/árabe

ISO 8859-7 - latino/grego

ISO 8859-8 - latino/hebraico

ISO 8859-9 - modificação do Latin-1 para turco (Latin-5)

ISO 8859-10 - línguas lapônicas/nórdicas/esquimós (Latin-6)

ISO 8859-11 - thai

ISO 8859-13 - línguas da Rim báltica (Latin-7)

ISO 8859-14 - celta (Latin-8)

ISO 8859-15 - línguas da Europa Ocidental (Latin-9)

Codificação Unicode

Unicode é padrão com propósito universal que objetiva mapear todos os tipos de símbolos, especialmente aqueles usados nos sistemas de escrita (idiomas) em todo o mundo. Além de símbolos de escrita, ele também mapeia símbolos matemáticos, formas geométricas, etc.

Unicode possui um pouco mais de 107 mil símbolos mapeados e são gerenciados pelo Unicode Consortium (organização sem fins lucrativos que coordena o desenvolvimento e a promoção do Unicode).



Codificação Unicode

Além da tabela de símbolos é previsto um conjunto de diagramas de códigos para referência visual, metodologia para codificação, enumeração de propriedades de caracteres, tais como, caixa alta e caixa baixa, regras como, por exemplo, collation, que especifica como dois símbolos devem ser comparados (para ordenação alfabética, por exemplo) entre outros.

face-smiling									
Nº	Code	Browser	Appl	Goog	FB	Wind	Twtr	Joy	Sams
1	U+1F600								
2	U+1F603								
3	U+1F604								



Uma lista completa está disponível em: <http://unicode.org/emoji/charts/full-emoji-list.html#face-smiling>

Codificação Unicode - exemplo

Crie uma arquivo com extensão HTML com o seguinte conteúdo e abra em um navegador:

```
<!DOCTYPE html>
<html>
<head>
<title>Emoji SMILE a partir do Código UNICODE U+1F600</title>
</head>
<body>
<h3>Para imprimir um simbolo UNICODE, iniciar com &# e finalizar com ;</h3>
<h3>O código hexadecimal 1F600 deve ser convertido para decimal</h3>
<p>&#128512;</p>
</body>
</html>
```

Codificação Unicode - exemplo

Emoji SMILE a partir do Código

← → C File | C:/Users/Dart/Desktop/index.html

Para imprimir um simbolo UNICODE, iniciar com &# e finalizar com ;
O codigo hexadecimal 1F600 deve ser convertido para decimal

😊

Codificação Unicode

As codificações anteriores utilizam um byte para armazenar os diferentes símbolos, no caso do Unicode para permitir um maior número de caracteres mapeados adotam-se esquemas padronizados de transformação denominado de UTF (Unicode Transformation Format).

Desta forma, existem algoritmos para a codificação e decodificação, tais como, UTF-8(UCS), UTF-16 (UCS-2), UTF-32 (UCS-4), etc.

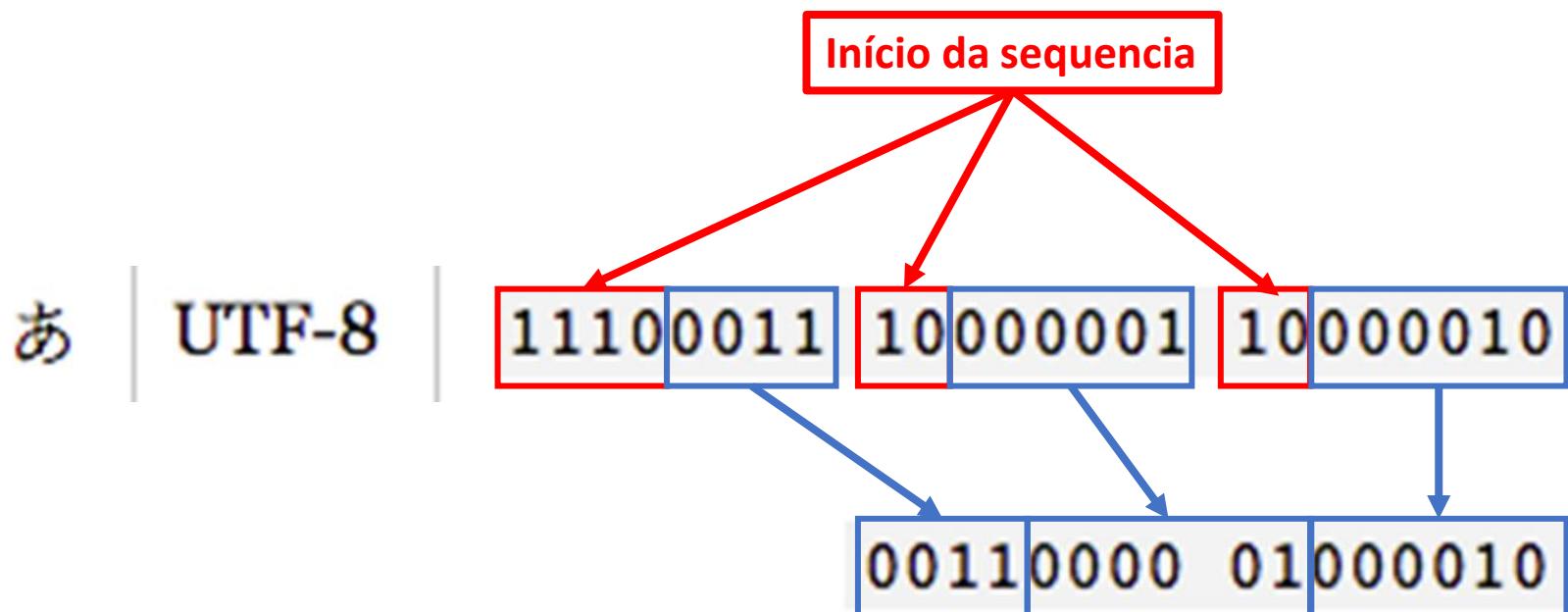
character	encoding	bits
A	UTF-8	01000001
A	UTF-16	00000000 01000001
A	UTF-32	00000000 00000000 00000000 01000001
あ	UTF-8	11100011 10000001 10000010
あ	UTF-16	00110000 01000010
あ	UTF-32	00000000 00000000 00110000 01000010



Letra “A” no silabário Hiragana

Codificação Unicode

De acordo com as regras do padrão UTF-8, caracteres entre U+0800 e U+FFFF devem ser codificados com 3 bytes, seguindo o seguinte padrão:



Codificação Unicode

Em suma, o padrão UTF-8 pode utilizar 8, 16, 24 ou 32 bits para representar um caractere.

O padrão UTF-16 pode utilizar 16 ou 32 bits para representar um caractere.

O padrão UTF-32 sempre irá utilizar 32 bits para representar um caractere.

A 00000041	Ω 000003A9	語 00008A9E	III 00010384	UTF-32
A 0041	Ω 03A9	語 8A9E	III D800 DF84	UTF-16
A 41	Ω CE A9	語 E8 AA 9E	III F0 90 8E 84	UTF-8

Fonte: jolicode.com

Agradecimentos

Este material é baseado nas notas de aula do **Prof. Dr. Jamil Kalil Naufal Júnior**.

Para saber mais...

... leia o Capítulo 1 (Conceitos Introdutórios), seção 1.4 (Representações Numéricas) e o Capítulo 2 (Sistemas de Numeração e Códigos) do livro-texto: TOCCI, R.J., WIDMER, N.S., MOSS, G.L. **Sistemas Digitais: Princípios e Aplicações.** 12^a ed. São Paulo: Pearson, 2018.

Unidade 3

Portas lógicas e Álgebra Booleana

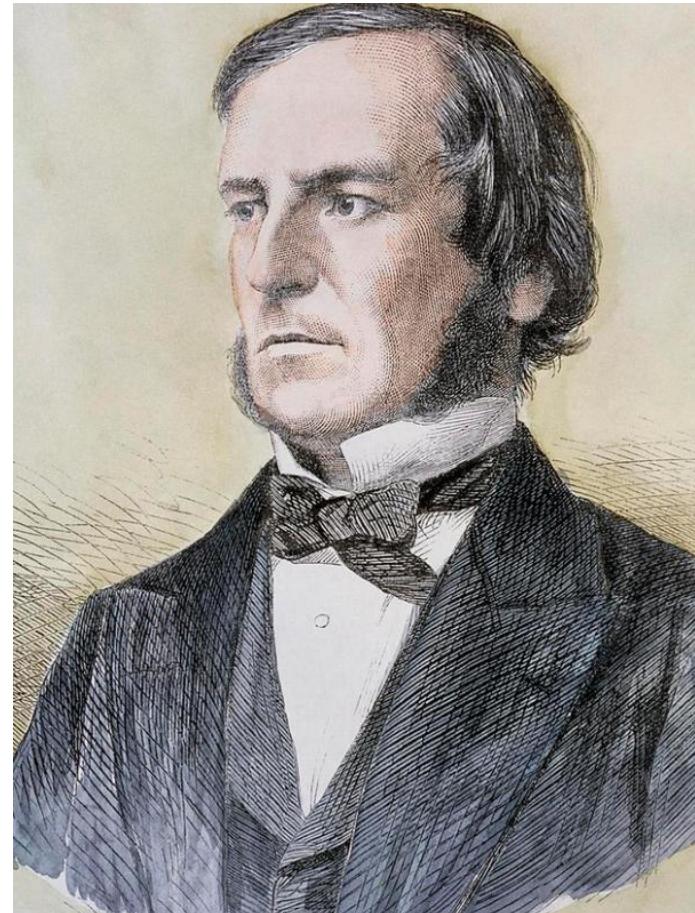
Objetivos

- Conhecer as operações lógicas AND, OR, NOT, NAND, NOR, XOR e XNOR;
- Conhecer expressões booleanas e circuitos lógicos;
- Praticar a construção de expressões e de circuitos lógicos.

Álgebra Booleana

George Boole (1815-1864) foi um matemático e filósofo britânico que criou, em 1848, um sistema matemático de análise lógica denominado álgebra de Boole ou álgebra booleana.

Esse sistema permitiu elaborar expressões lógicas que possibilitaram o desenvolvimento da eletrônica digital e, consequentemente, dos computadores.



Fonte: wikipedia.org

Portas lógicas

As portas lógicas são componentes básicos de decisão utilizados para criar circuitos digitais e circuitos integrados simples e complexos.

O funcionamento das portas lógicas consiste em verificar, durante a operação, os dados presentes em suas entradas para se determinar o valor de saída.

Existem três tipos básicos de portas:

- E (AND);
- OU (OR); e
- NÃO (NOT).

E existem quatro tipos de portas derivadas:

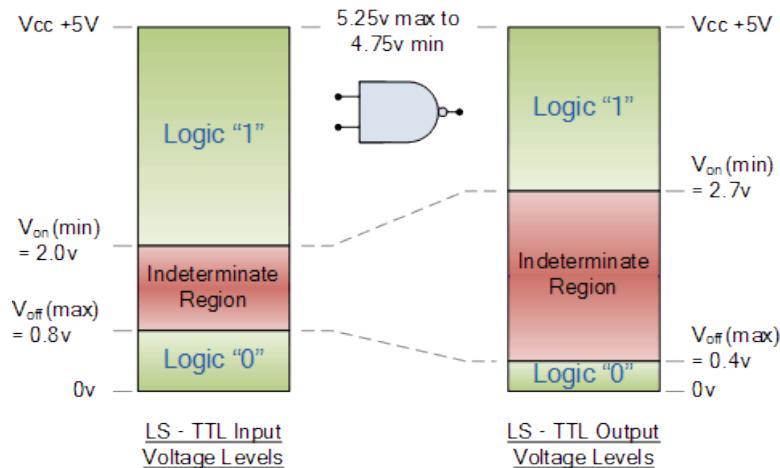
- NÃO E (NAND);
- NÃO OU (NOR);
- OU EXCLUSIVO (XOR); e
- NÃO OU EXCLUSIVO (XNOR).

Portas lógicas

As portas lógicas operam com níveis lógicos que representam o estado da entrada ou saída de um circuito.

O nível lógico é uma representação que uma variável booleana poderá assumir, como por exemplo 0 ou 1.

- Nível lógico zero (0): pode representar falso, desligado, baixo, não, aberto, entre outros;
- Nível lógico um (1): pode representar verdadeiro, ligado, alto, sim, fechado, entre outros.



Fonte: vlsiencyclopedia.com

Tabela verdade

A tabela verdade é uma maneira de descrever como a saída de um circuito lógico depende dos níveis lógicos presentes na entrada do circuito.

A seguir são mostradas tabelas verdade para duas (a), três (b) e quatro (c) variáveis de entrada e uma de saída:

Entradas			Saída
A	B		x
0	0		1
0	1		0
1	0		1
1	1		0

(a)

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

(b)

A	B	C	D		x
0	0	0	0		0
0	0	0	1		0
0	0	1	0		0
0	0	1	1		1
0	1	0	0		1
0	1	0	1		0
0	1	1	0		0
0	1	1	1		1
1	0	0	0		0
1	0	0	1		0
1	0	1	0		0
1	0	1	1		1
1	1	0	0		0
1	1	0	1		1
1	1	1	0		0
1	1	1	1		1

(c)

Fonte: TOCCI, R. J. et al. Sistemas digitais: princípios e aplicações. 12. ed. São Paulo, SP: Pearson, 2018

Tabela verdade

Observe que a tabela verdade descreve todas as combinações possíveis de entradas do sistema e indica a saída para cada caso.

O circuito lógico representa um **círcuito lógico combinacional** representado por portas lógicas individuais ou combinações das mesmas.

Circuitos combinacionais são aqueles em que o sinal de saída depende única e exclusivamente das combinações dos sinais de entrada. Os circuitos deste tipo não possuem nenhum tipo de memória, ou seja, as saídas não dependem de nenhum estado anterior do circuito.



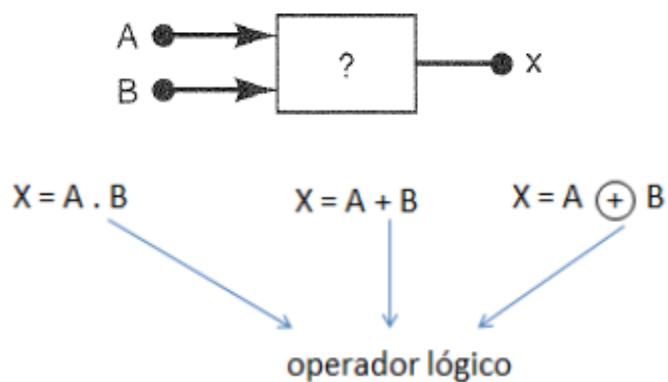
Combinacional
A saída depende apenas das entradas

Figura: notas de aula do Prof. Onur Mutlu

Expressão booleana

Uma equação ou expressão booleana é a expressão lógica que representa como as variáveis de entrada de um circuito estão associadas a saída correspondente. Tem como objetivo descrever a combinação de portas lógicas contidas dentro de um sistema.

Para um circuito lógico composto de duas entradas e uma saída, como a seguir, poderíamos ter, por exemplo a seguintes expressões lógicas:



O tipo de operador lógico define a operação que deve ser realizada entre as entradas para obter-se a saída correspondente.



Operação E (AND)

A porta lógica AND é constituída por duas entradas (A, B) e uma saída (X).

Símbolo Lógico (porta lógica E)



Circuito elétrico equivalente

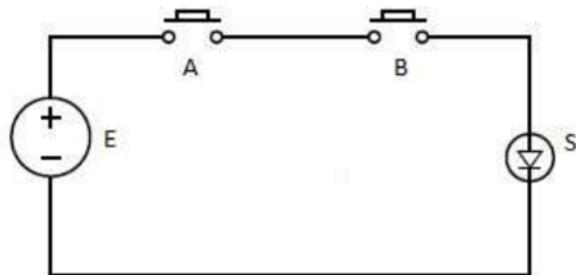
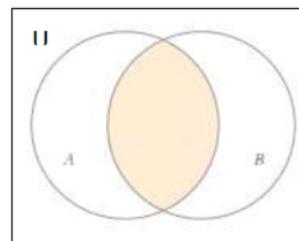


Diagrama de Venn



Expressão booleana

$$X = A \cdot B$$

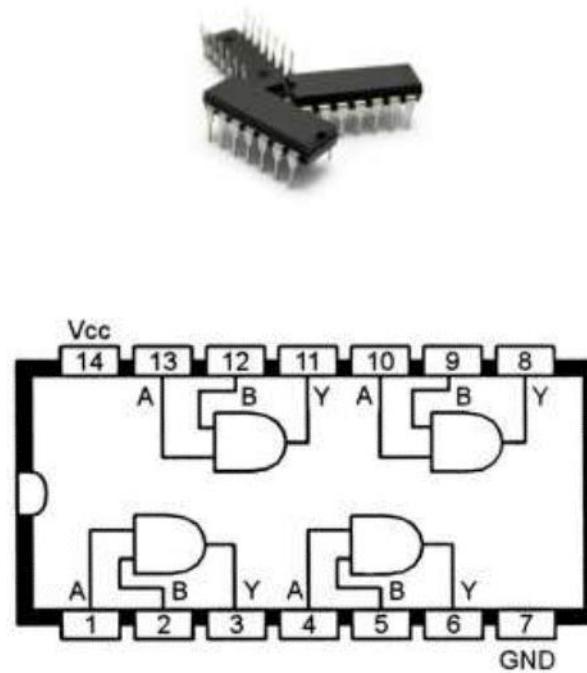
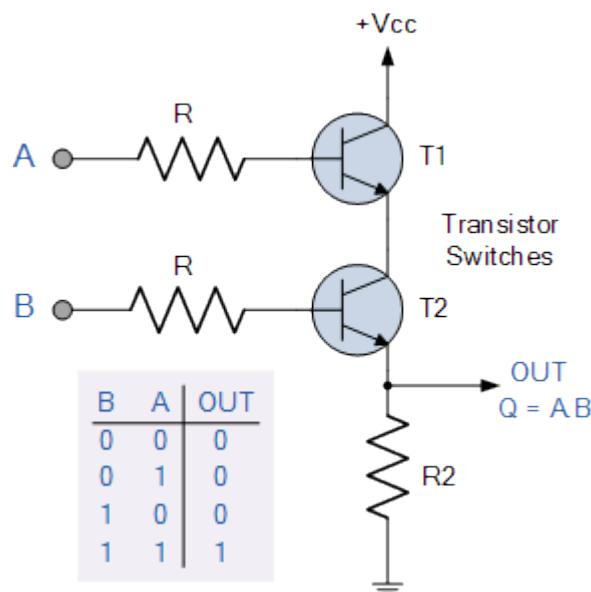
operador lógico AND



A saída irá para o nível lógico 1 se as duas entradas (uma E a outra) receberem o valor lógico 1.

Operação E (AND)

Comercialmente as portas são agregadas em circuitos integrados (CI). Exemplo: **SN7408** de tecnologia TTL (Transistor-Transistor Logic) fabricado pela Texas Instruments (SN).



Fonte: electronics-tutorials.ws

Operação OU (OR)

A porta lógica OR é constituída por duas entradas (A, B) e uma saída (X).

Símbolo Lógico (Porta Lógica OU)

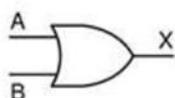
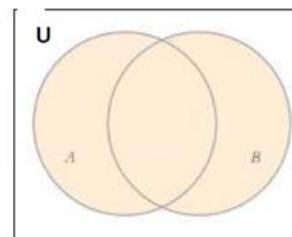
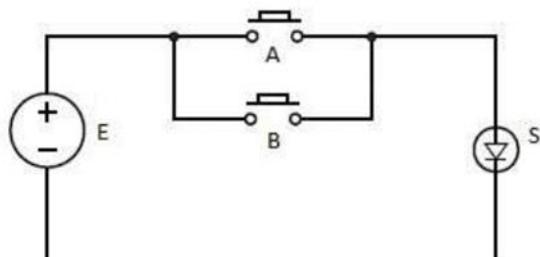


Diagrama de Venn



Círcuito elétrico equivalente



Expressão booleana

$$X = A + B$$

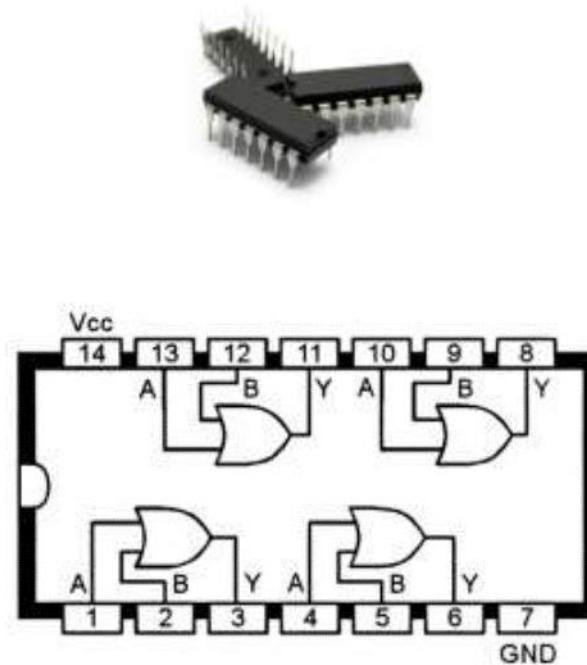
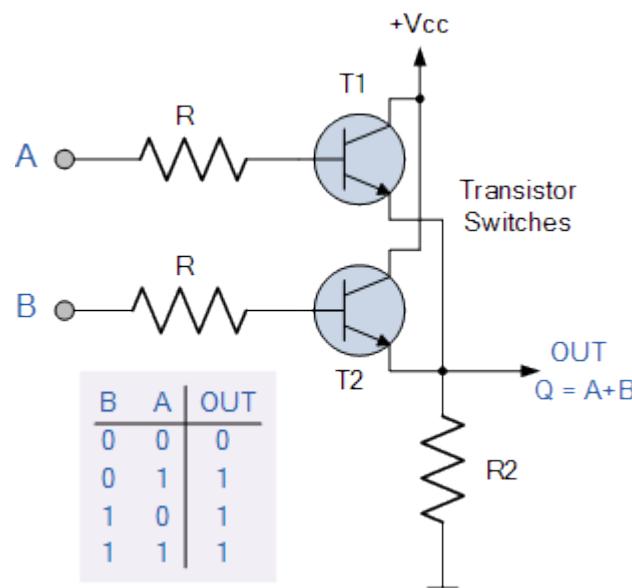
operador lógico OR



A saída irá para o nível lógico 1 se ao menos uma de suas entradas (uma OU a outra) receber um valor lógico 1.

Operação OU (OR)

Comercialmente as portas são agregadas em circuitos integrados (CI). Exemplo: **SN7432** de tecnologia TTL (Transistor-Transistor Logic) fabricado pela Texas Instruments (SN).



Fonte: electronics-tutorials.ws

Operação NÃO (NOT)

A porta lógica NOT é constituída de uma entrada (A) e uma saída (X).

Símbolo Lógico (Porta Lógica INVERSOR)

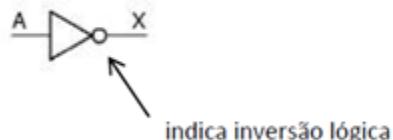
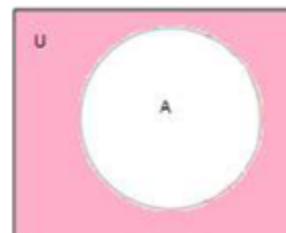
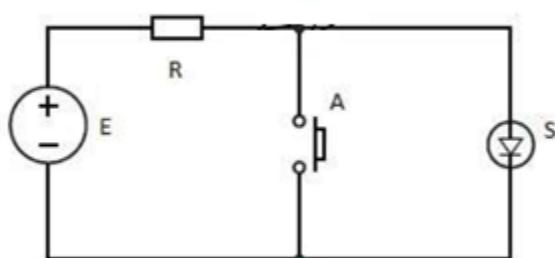


Diagrama de Venn



Círcuito elétrico equivalente



Expressão booleana

$$X = \overline{A} = A'$$

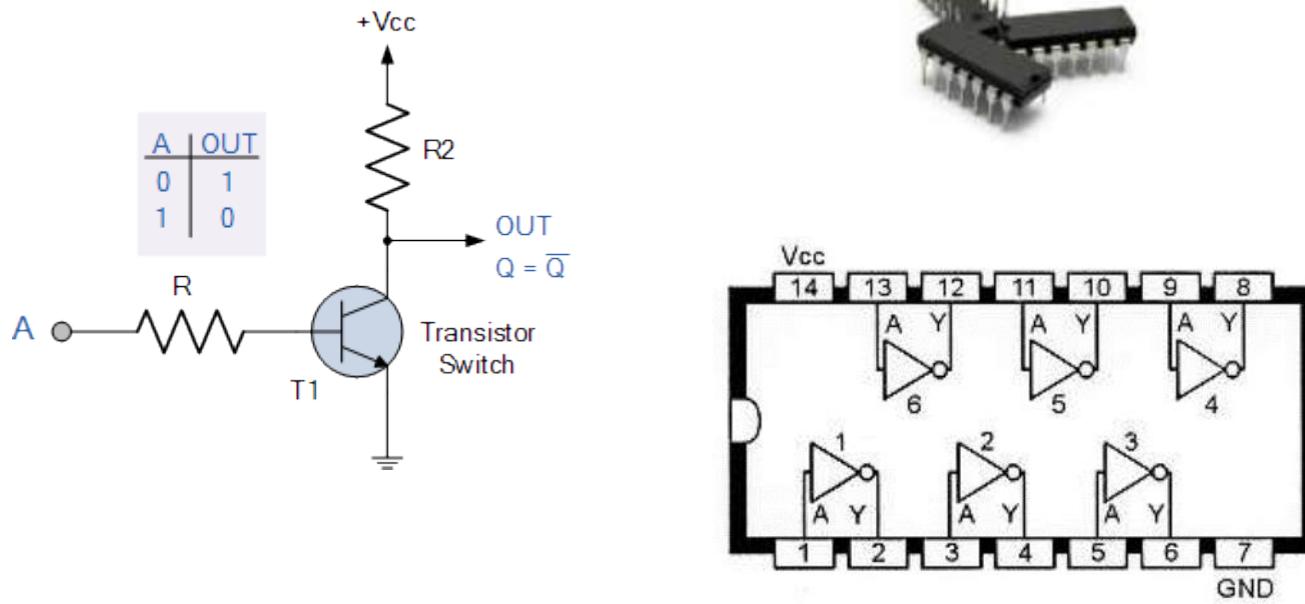
Operador lógico negação

A expressão lógica deve ser lida como “X é igual a NOT A”. Observe que a operação NOT realiza uma inversão lógica da entrada A. Se A = 0, então NOT 0 é 1; se A=1, então NOT 1 é 0.



Operação NÃO (NOT)

Comercialmente as portas são agregadas em circuitos integrados (CI). Exemplo: **SN7404** de tecnologia TTL (Transistor-Transistor Logic) fabricado pela Texas Instruments (TI).

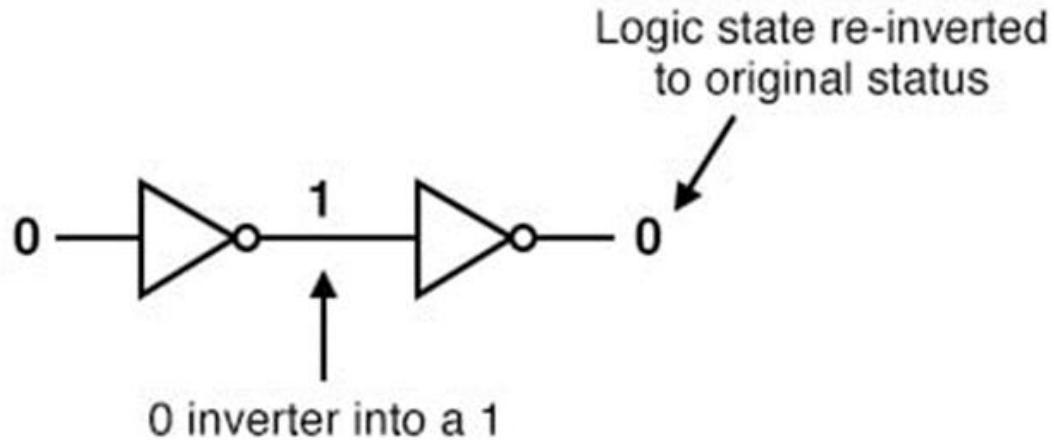


Fonte: electronics-tutorials.ws

Buffer

Ao conectar duas portas inversoras juntas para que a saída de uma alimente a entrada da outra, as duas funções de inversão se cancelam de forma que não haverá inversão da entrada para a saída final.

Embora isso possa parecer sem sentido, ela tem uma importante aplicação prática. Como os circuitos de porta são amplificadores de sinal, independentemente da função lógica que possam desempenhar, este circuito acaba atuando como um repetidor de sinal.



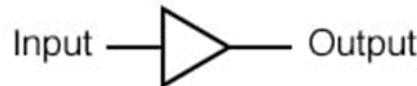
Fonte: allaboutcircuits.com

Buffer

Para desempenhar o papel de amplificador de sinal, uma porta lógica especial chamada **buffer** foi desenvolvida para desempenhar a mesma função que dois inversores.

Seu símbolo é simplesmente um triângulo, sem o sinal inversor no terminal de saída.

"Buffer" gate



Input	Output
0	0
1	1

⚠ Para saber mais sobre a aplicação do **buffer**, pesquisar as características de **fan-in** e **fan-out** dos circuitos integrados.

Fonte: allaboutcircuits.com

Operação NÃO E (NAND)

A porta lógica NAND é constituída de duas entradas (A, B) e uma saída (X).

Símbolo Lógico (Porta Lógica NAND)

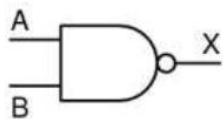
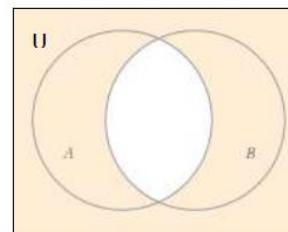
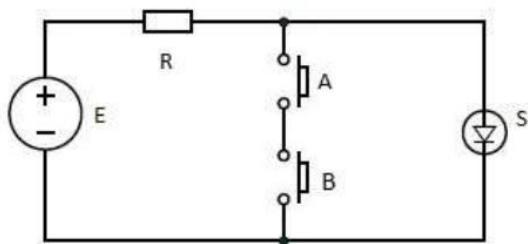


Diagrama de Venn



Circuito elétrico equivalente



Expressão booleana

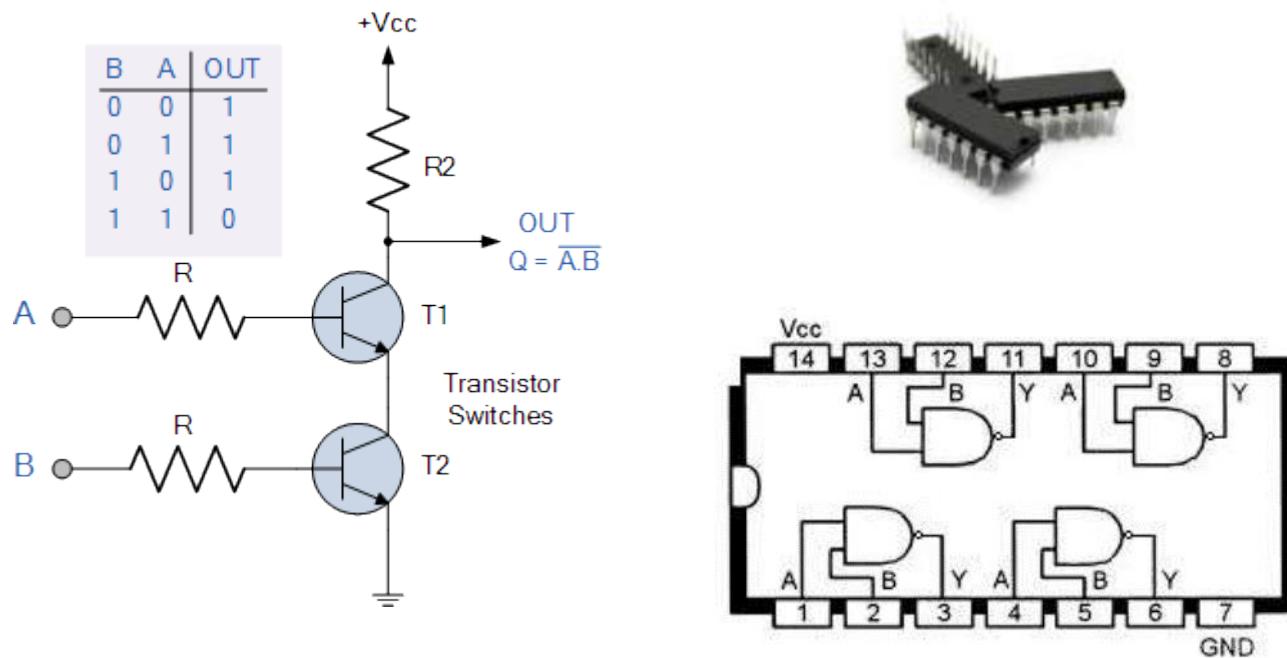
$$X = \overline{A \cdot B}$$



A porta lógica NAND corresponde a uma porta E com a saída invertida. Ela é a junção das portas lógicas AND e NOT acopladas em uma única operação.

Operação NÃO E (NAND)

Comercialmente as portas são agregadas em circuitos integrados (CI). Exemplo: **SN7400** de tecnologia TTL (Transistor-Transistor Logic) fabricado pela Texas Instruments (TI).



Fonte: electronics-tutorials.ws

Operação NÃO OU (NOR)

A porta lógica NOR é constituída de duas entradas (A, B) e uma saída (X).

Símbolo Lógico (Porta Lógica OU)

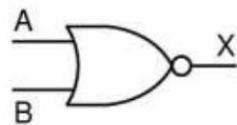
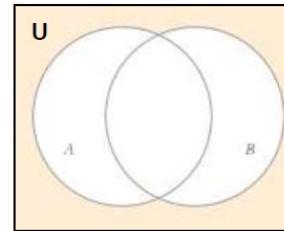
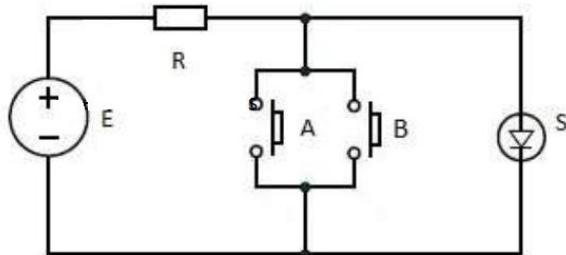


Diagrama de Venn



Circuito elétrico equivalente



Expressão booleana

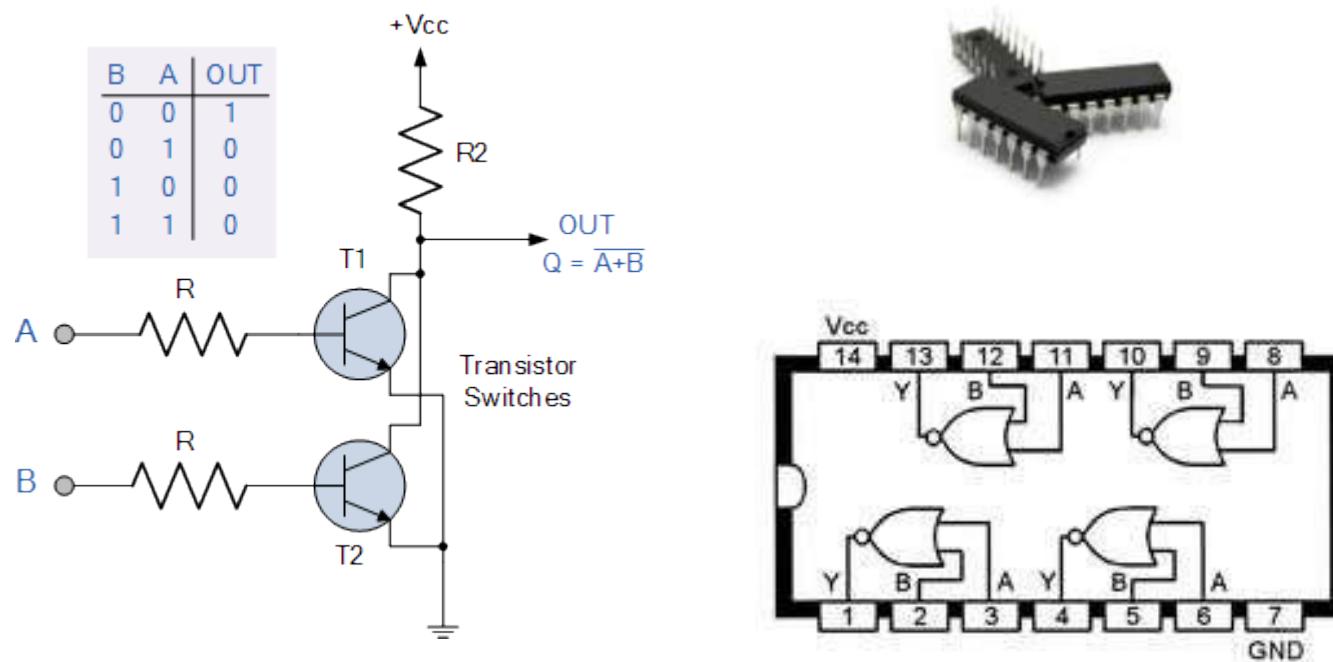
$$X = \overline{A+B}$$



A porta lógica NOR corresponde a uma porta OU com a saída invertida. Ela é a junção das portas lógicas OR e NOT acopladas em uma única porta.

Operação NÃO OU (NOR)

Comercialmente as portas são agregadas em circuitos integrados (CI). Exemplo: **SN7402** de tecnologia TTL (Transistor-Transistor Logic) fabricado pela Texas Instruments (TI).

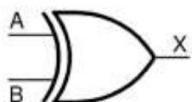


Fonte: electronics-tutorials.ws

Operação OU EXCLUSIVO (XOR)

A porta lógica XOR é constituída de duas entradas (A, B) e uma saída (X).

Símbolo Lógico (Porta Lógica OU)



Circuito elétrico equivalente

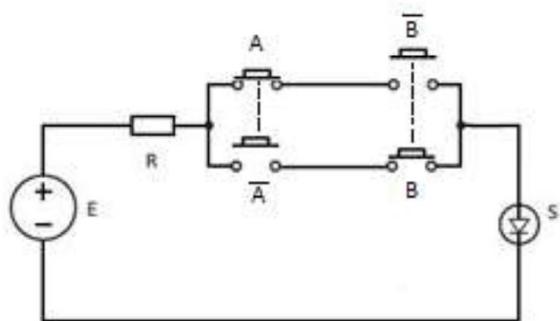
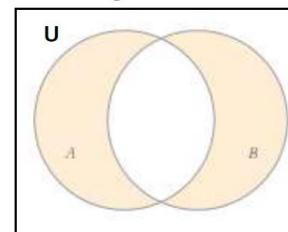


Diagrama de Venn



Expressão booleana

$$X = \overline{A}B + A\overline{B}$$

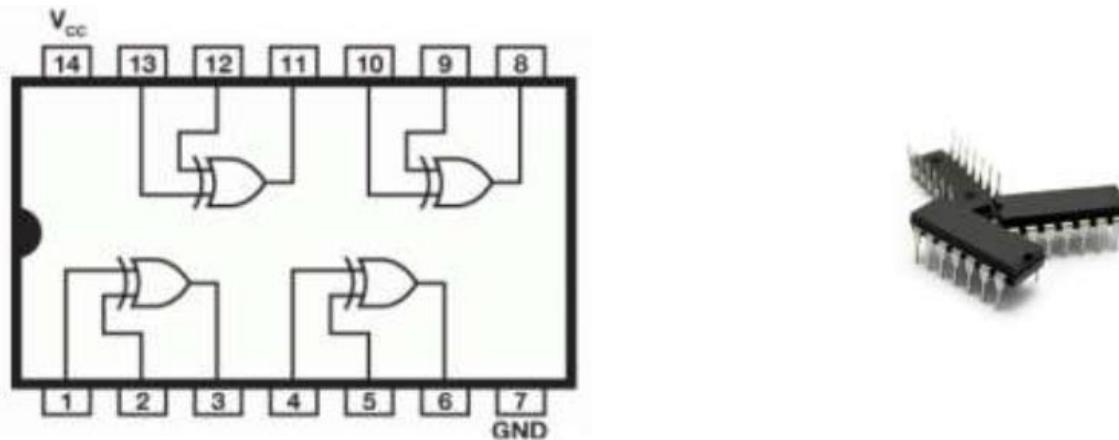
$$X = A \oplus B$$



A saída irá para o nível lógico 1 se as duas entradas receberem valores distintos.

Operação OU EXCLUSIVO (XOR)

Comercialmente as portas são agregadas em circuitos integrados (CI). Exemplo: **SN7486** de tecnologia TTL (Transistor-Transistor Logic) fabricado pela Texas Instruments (TI).



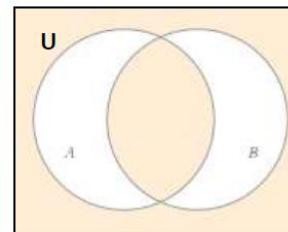
Operação NÃO OU EXCLUSIVO (XNOR)

A porta lógica XNOR é constituída de duas entradas (A, B) e uma saída (X).

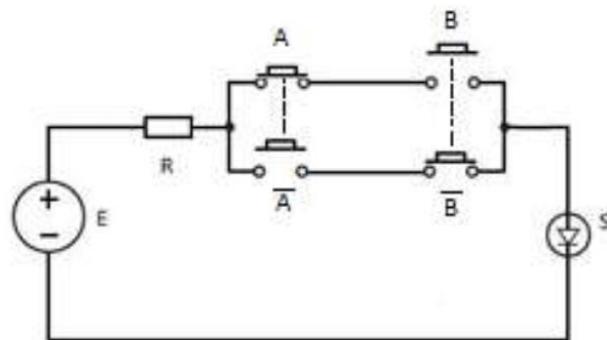
Símbolo Lógico (Porta Lógica OU)



Diagrama de Venn



Circuito elétrico equivalente



Expressão booleana

$$X = \overline{A} \cdot \overline{B} + A \cdot \overline{B}$$

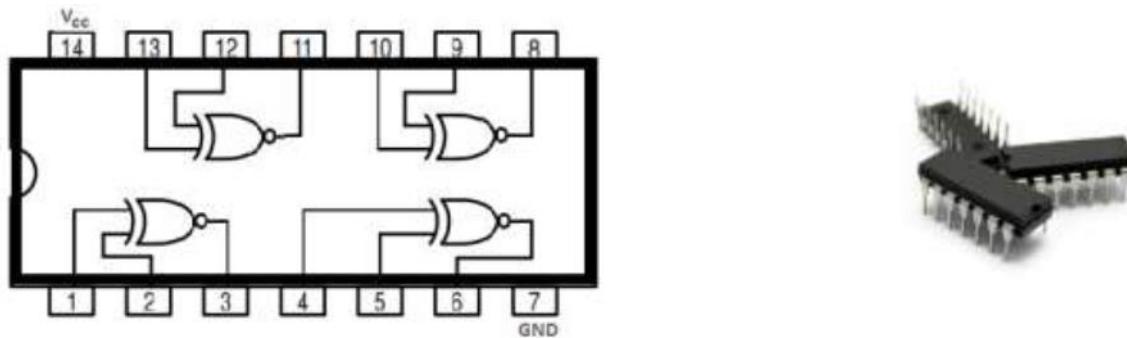
$$X = A \odot B$$



A saída irá para o nível lógico 1 se as duas entradas receberem valores iguais.

Operação NÃO OU EXCLUSIVO (XNOR)

Comercialmente as portas são agregadas em circuitos integrados (CI). Exemplo: **SN74266** de tecnologia TTL (Transistor-Transistor Logic) fabricado pela Texas Instruments (TI).



Portas lógicas - resumo

PORTA	Símbolo Usual	Tabela da Verdade	Função Lógica	Expressão															
E AND		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	S	0	0	0	0	1	0	1	0	0	1	1	1	Função E: Assume 1 quando todas as variáveis forem 1 e 0 nos outros casos.	$S = A \cdot B$
A	B	S																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OU OR		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	1	Função E: Assume 0 quando todas as variáveis forem 0 e 1 nos outros casos.	$S = A + B$
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
NÃO NOT		<table border="1"> <thead> <tr> <th>A</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </tbody> </table>	A	S	0	1	1	0	Função NÃO: Inverte a variável aplicada à sua entrada.	$S = \overline{A}$									
A	S																		
0	1																		
1	0																		

Portas lógicas - resumo

PORTA	Símbolo Usual	Tabela da Verdade	Função Lógica	Expressão															
NE NAND		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	S	0	0	1	0	1	1	1	0	1	1	1	0	Função NE: Inverso da função E.	$S = (\overline{A} \cdot \overline{B})$
A	B	S																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOU NOR		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	0	Função NOU: Inverso da função OU.	$S = (\overline{A} + \overline{B})$
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	

Portas lógicas - resumo

PORTA	Símbolo Usual	Tabela da Verdade	Função Lógica	Expressão															
OU Exclusivo XOR		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	0	Função OU Exclusivo: Assume 1 quando as variáveis assumirem valores diferentes entre si.	$S = A \oplus B$ $S = \bar{A} \cdot B + A \cdot \bar{B}$
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
Coincidência XNOR		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	1	Função Coincidência: Assume 1 quando houver coincidência entre os valores das variáveis.	$S = A \odot B$ $S = \bar{A} \cdot \bar{B} + A \cdot B$
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

Circuitos integrados - resumo

Circuitos integrados TTL (Transistor-Transistor-Logic) da série 7400:

- 7408: Quatro portas **AND** de duas entradas;
- 7432: Quatro portas **OR** de duas entradas;
- 7404: Seis portas **NOT** (inversor);
- 7400: Quatro portas **NAND** de duas entradas;
- 7402: Quatro portas **NOR** de duas entradas;
- 7486: Quatro portas **XOR** de duas entradas;
- 74266: Quatro portas **XNOR** de duas entradas (com coletor aberto).



Uma lista completa está disponível em https://en.wikipedia.org/wiki/List_of_7400-series_integrated_circuits.

Portas lógicas - XOR vs XNOR

Uma outra forma de entender o comportamento da porta **XOR** é verificar a sua tabela verdade para três entradas.

Neste caso, a saída S será igual a “1” sempre que houver um número **ímpar** de entradas “1”.

ENTRADAS			SAÍDA
A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Fonte: electrical4u.com

Portas lógicas - XOR vs XNOR

Uma outra forma de entender o comportamento da porta **XNOR** é verificar a sua tabela verdade para três entradas.

Neste caso, a saída S será igual a “1” sempre que houver um número **par** de entradas “1”.

ENTRADAS			SAÍDA
A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Fonte: electrical4u.com

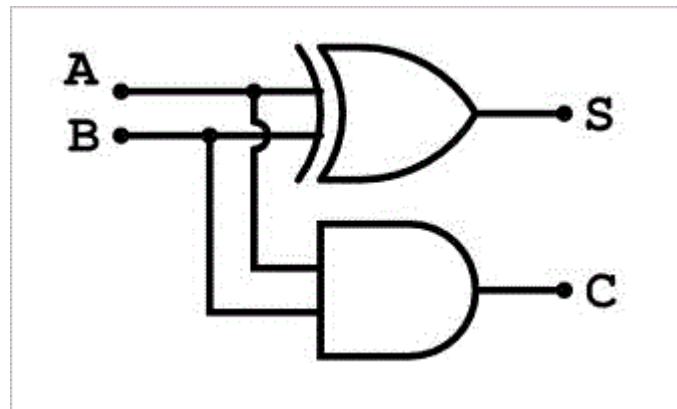
Porta lógica XOR - aplicações

Uma das aplicações da porta XOR é a sua utilização nos circuitos de meio somador e somador completo.

A partir da tabela verdade, é possível notar que os três primeiros resultados satisfazem totalmente o processo de adição binária.

A porta AND é adicionada para satisfazer a regra da adição binária, ou seja, quando ambas as entradas são 1, o resultado deve ser 0 com um *carry* 1.

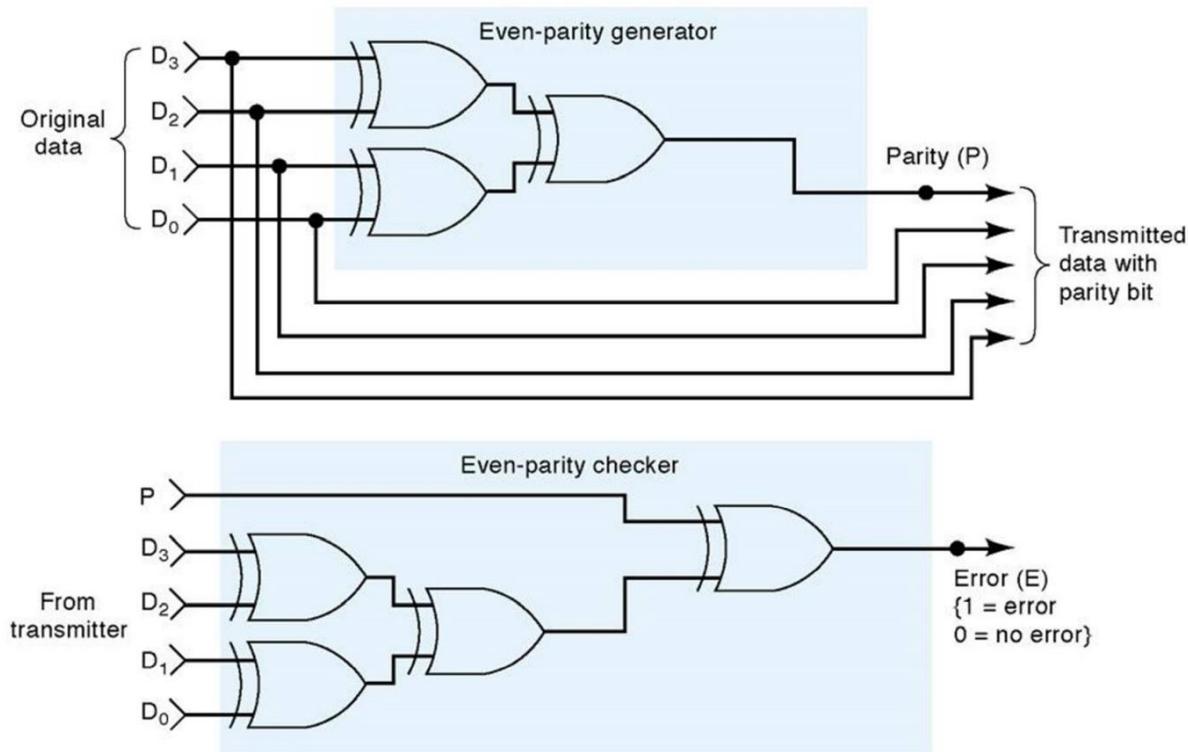
$A + B$	SUM	CARRY
$0 + 0$	0	0
$0 + 1$	1	0
$1 + 0$	1	0
$1 + 1$	0	1



Fonte: electrical4u.com

Porta lógica XOR - aplicações

Uma outra aplicação da porta XOR é a sua utilização nos circuitos de geração e checagem de paridade, usada para detectar erros de transmissão de dados.

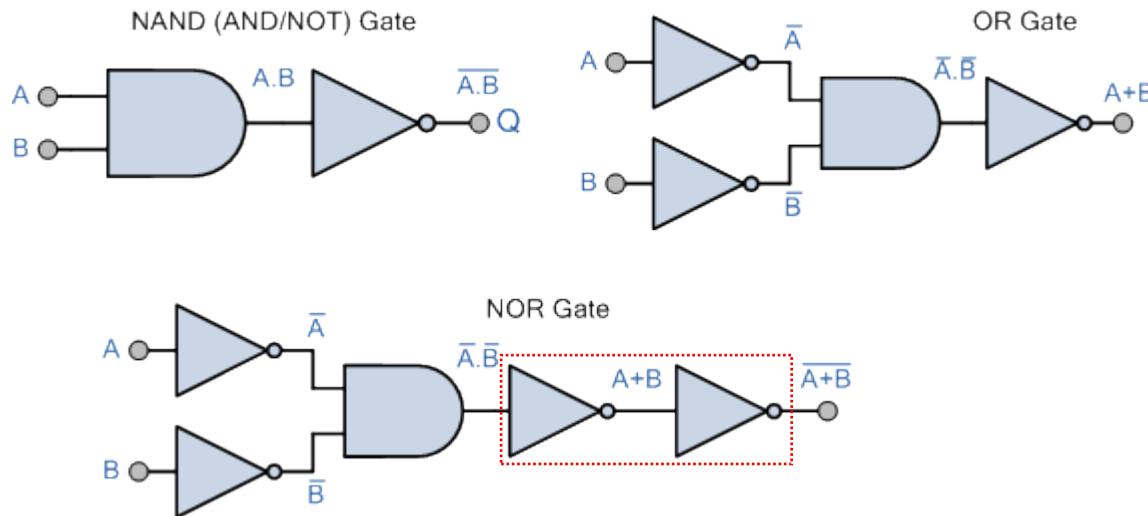


Fonte: Notas de aula do Professor Parag Parandkar

Equivalência de portas lógicas

Uma determinada função booleana pode ser obtida a partir da combinação de diferentes portas lógicas, e um circuito é equivalente quando se utilizam duas ou mais portas lógicas para implementar uma determinada função booleana.

Funções lógicas equivalentes a partir de portas AND e NOT:

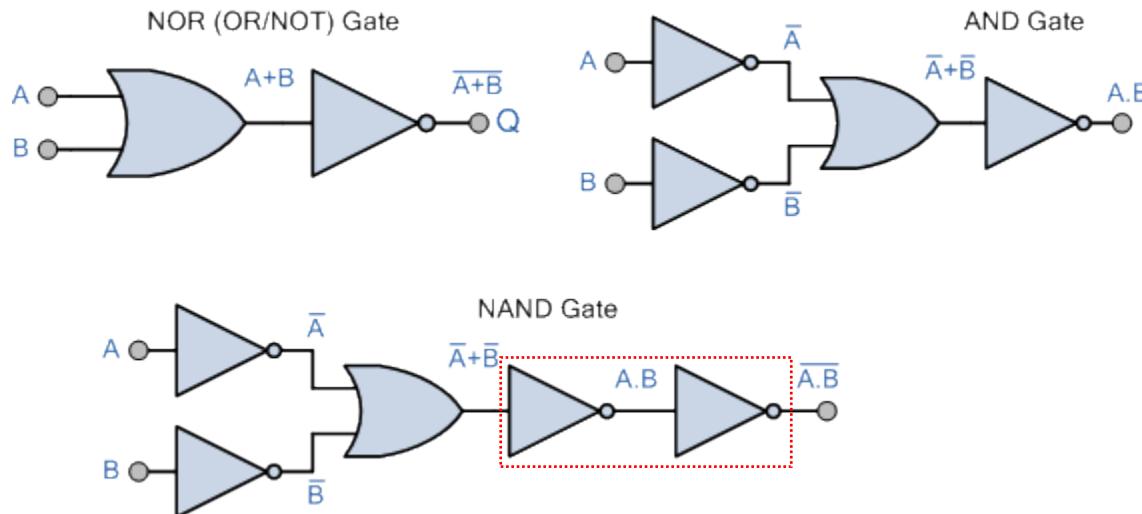


! Na função equivalente **NOR**, os dois inversores servem apenas para demonstrar o **segundo teorema de De Morgan**. Neste caso eles podem ser suprimidos.

Fonte: electronics-tutorials.ws

Equivalência de portas lógicas

Funções lógicas equivalentes a partir de portas OR e NOT:



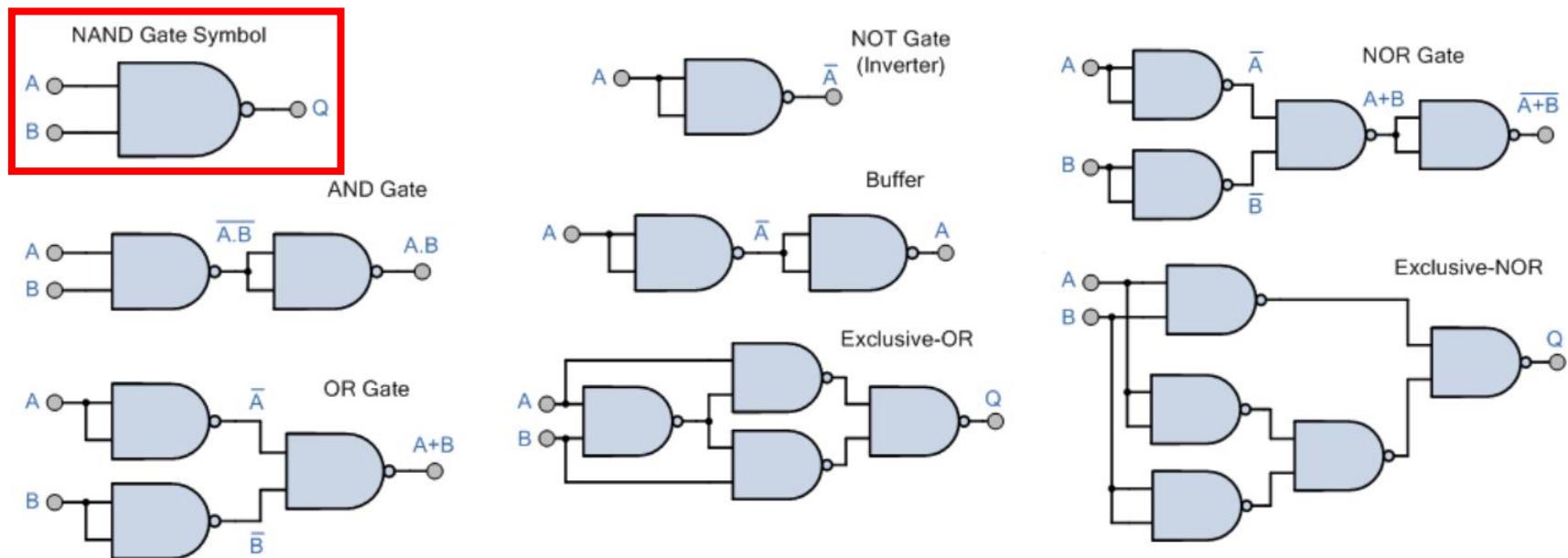
Na função equivalente **NAND**, os dois inversores servem apenas para demonstrar o **primeiro teorema de De Morgan**. Neste caso eles podem ser suprimidos.

Fonte: electronics-tutorials.ws

Portas lógicas universais NAND e NOR

As portas lógicas **NAND** e **NOR** são consideradas universais porque é possível implementar qualquer função lógica usando-se apenas combinações de uma delas.

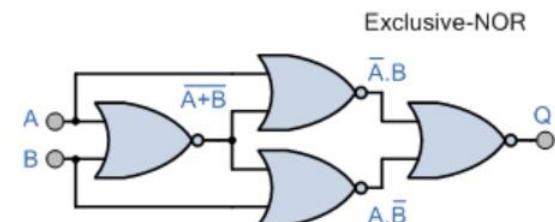
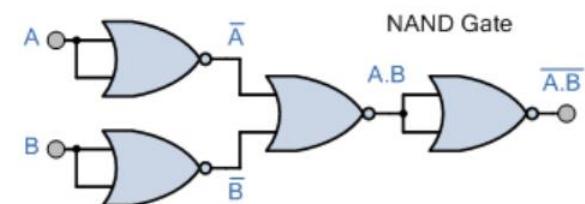
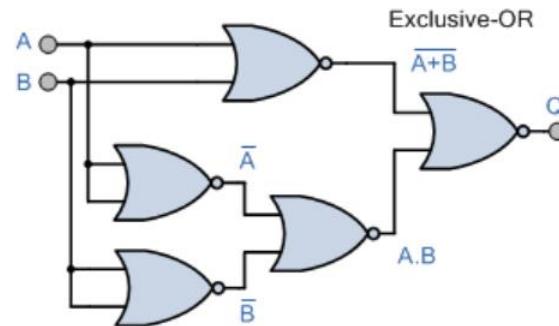
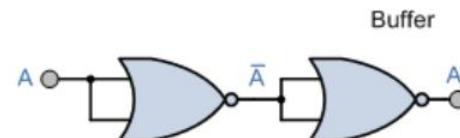
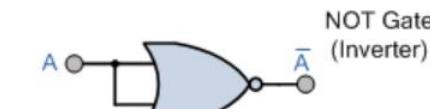
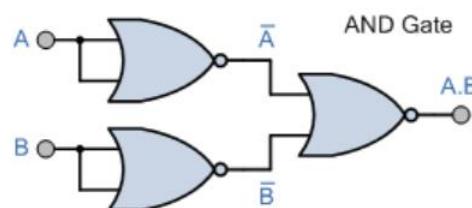
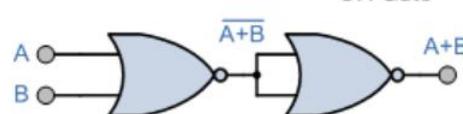
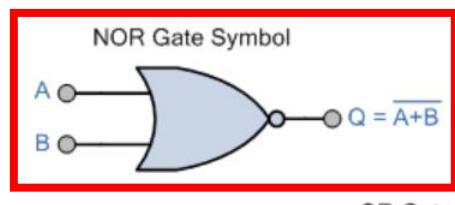
Funções lógicas universais usando-se apenas portas NAND:



Fonte: electronics-tutorials.ws

Portas lógicas universais NAND e NOR

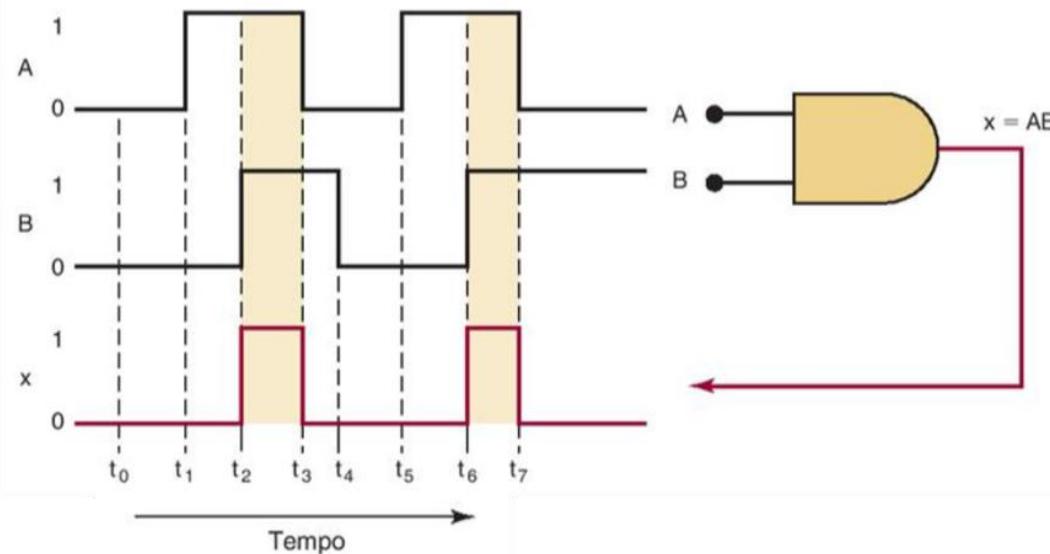
Funções lógicas universais usando-se apenas portas NOR:



Fonte: electronics-tutorials.ws

Variação temporal em circuitos combinacionais

Os valores das entradas dos circuitos lógicos podem ser modificados por diferentes motivos ao longo do tempo, e estas alterações devem ser refletidas integralmente na saída de acordo com as operações lógicas envolvidas.

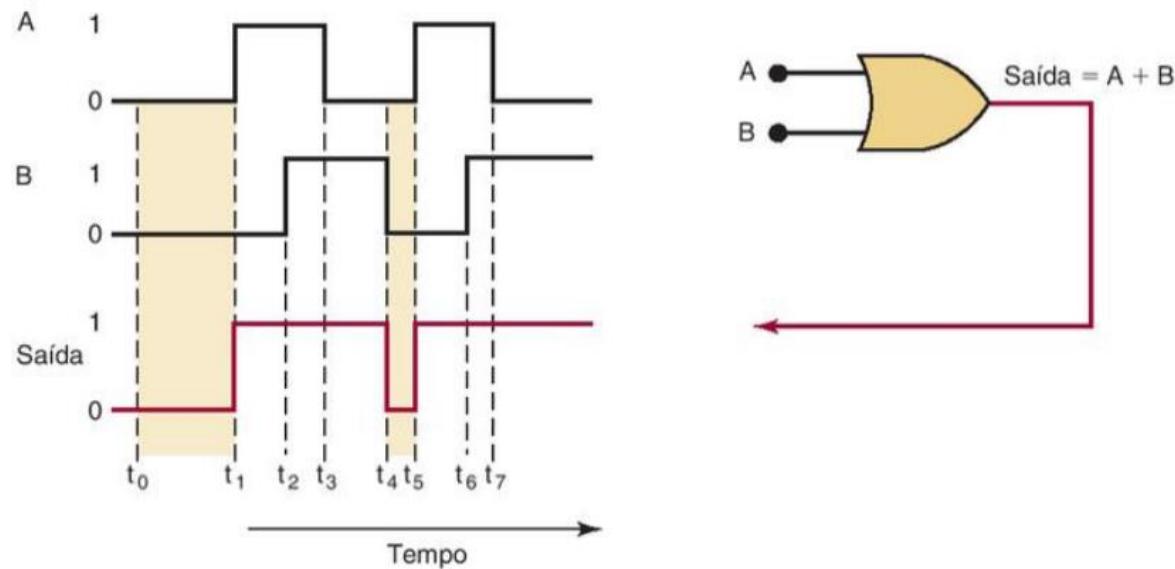


⚠️ Como um circuito combinacional não possui memória, então toda alteração em suas entradas deve ser refletida em sua saída.

Fonte: TOCCI, R. J. et al. Sistemas digitais: princípios e aplicações. 12. ed. São Paulo, SP: Pearson, 2018

Variação temporal em circuitos combinacionais

O diagrama a seguir mostra um exemplo de variação temporal da porta OR:

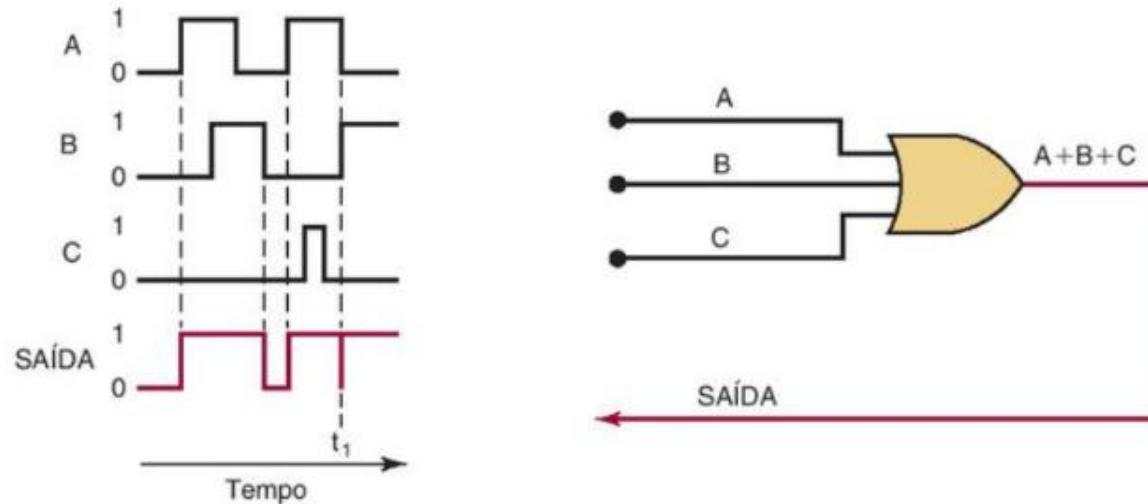


Fonte: TOCCI, R. J. et al. Sistemas digitais: princípios e aplicações. 12. ed. São Paulo, SP: Pearson, 2018

Variação temporal em circuitos combinacionais

No diagrama abaixo, no instante t_1 , em que A muda de “1” para “0” e B muda de “0” para “1”, as entradas fazem suas transições quase de maneira simultânea.

Neste intervalo, as duas entradas estão na faixa indefinida entre 0 e 1, cuja evidência aparece na saída na forma de **glitch** ou **spike**.

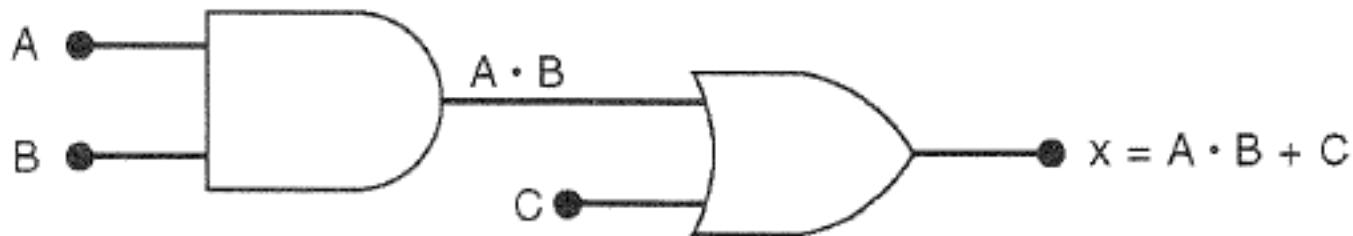


! A ocorrência de *glitch*, sua amplitude e largura dependem da velocidade em que ocorrem as transições nas entradas.

Fonte: TOCCI, R. J. et al. Sistemas digitais: princípios e aplicações. 12. ed. São Paulo, SP: Pearson, 2018

Descrição algébrica de circuitos lógicos

Circuitos lógicos utilizando duas ou mais portas podem ser descritos por meio de expressões booleanas utilizando os operadores AND, OR e NOT.



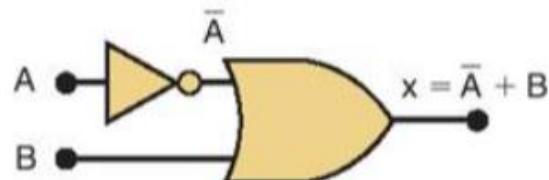
A expressão poderia ser interpretada de duas formas: (1) realiza-se primeiro a operação $A \cdot B$ **OR** C ; (2) ou a operação A **AND** $B+C$. Como a operação AND possui maior precedência sobre a operação OR, ela sempre será realizada primeiro.

A ordem de precedência pode ser alterada utilizando-se parênteses.

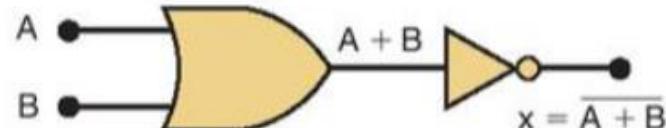
! Na expressão **$A.(B+C)$** , a operação OR é realizada primeiro devido aos parênteses.

Descrição algébrica de circuitos lógicos com inversores

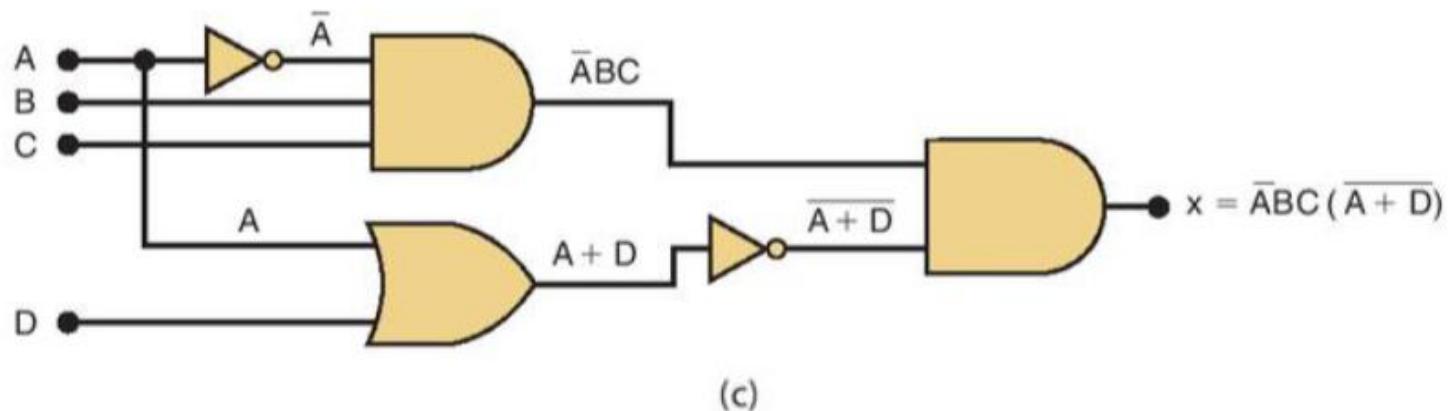
Sempre que for utilizado um **INVERSOR**, a expressão lógica de sua saída é igual à expressão de entrada com uma barra sobre ela indicando a sua inversão.



(a)



(b)



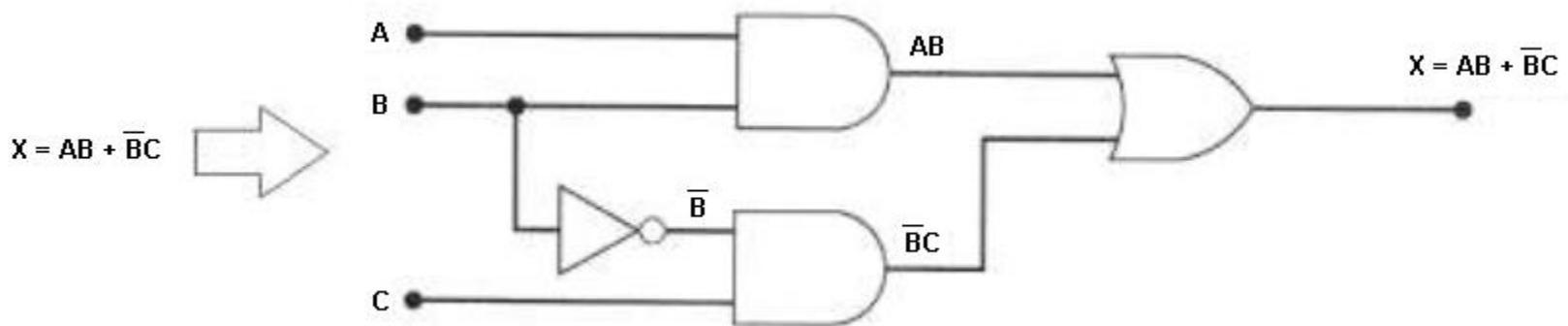
(c)

Fonte: TOCCI, R. J. et al. Sistemas digitais: princípios e aplicações. 12. ed. São Paulo, SP: Pearson, 2018

Descrevendo circuitos lógicos a partir de expressões lógicas

Um circuito lógico pode ser obtido a partir de sua correspondente expressão lógica.

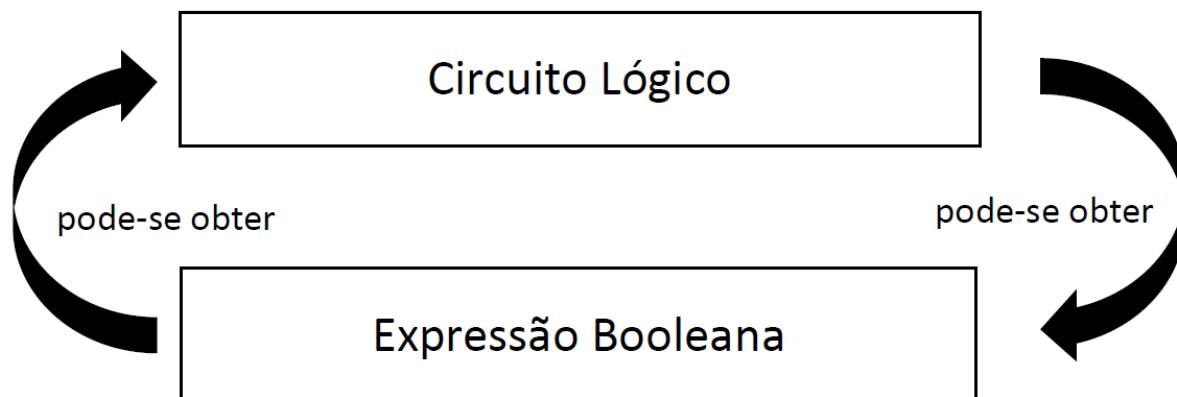
Exemplo: dada a expressão lógica $X = AB + B'C$, os termos AB e $B'C$ são entradas de uma porta OR e cada um destes pode ser gerado a partir de uma porta AND.



! Uma forma alternativa de representar um porta invertida, além do uso da barra, é utilizando o sinal de apóstrofo (').

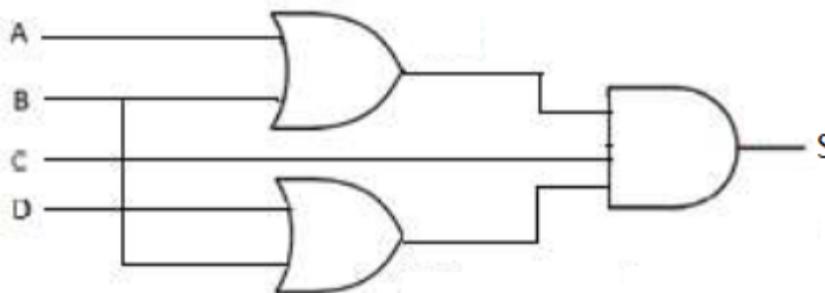
Descrevendo expressões lógicas a partir de circuitos lógicos

Igualmente uma expressão lógica pode ser obtida a partir de seu correspondente circuito lógico. Na verdade existe uma dualidade, pode-se obter a expressão lógica a partir de um circuito lógico, ou então, obter o circuito lógico a partir da expressão lógica.

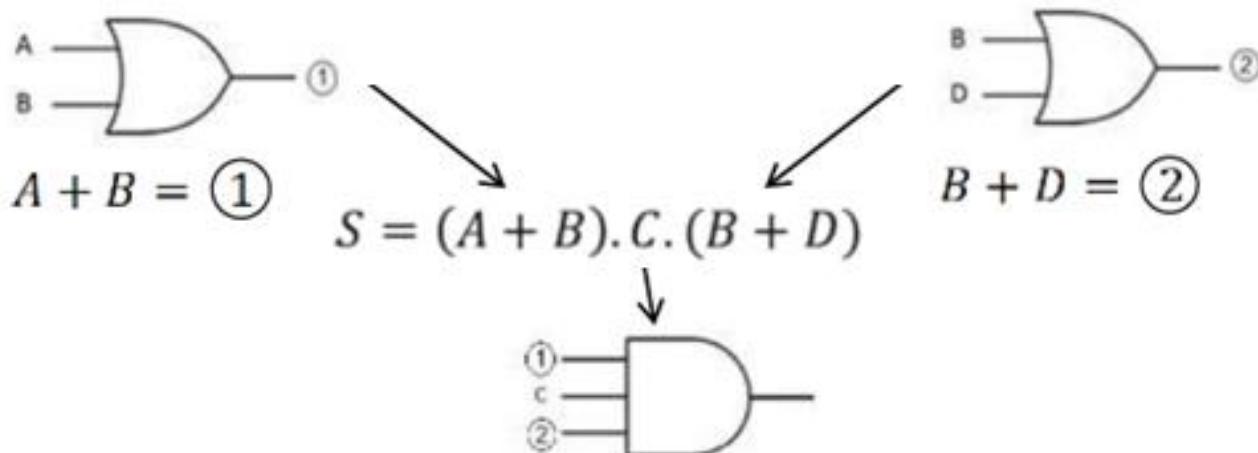


Descrevendo expressões lógicas a partir de circuitos lógicos

Exemplo. Do circuito lógico:

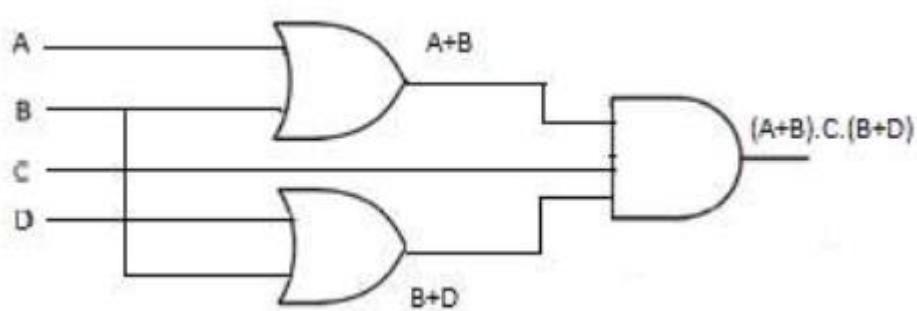


Obtém-se a expressão lógica:



Descrevendo expressões lógicas a partir de circuitos lógicos

Tabela verdade:

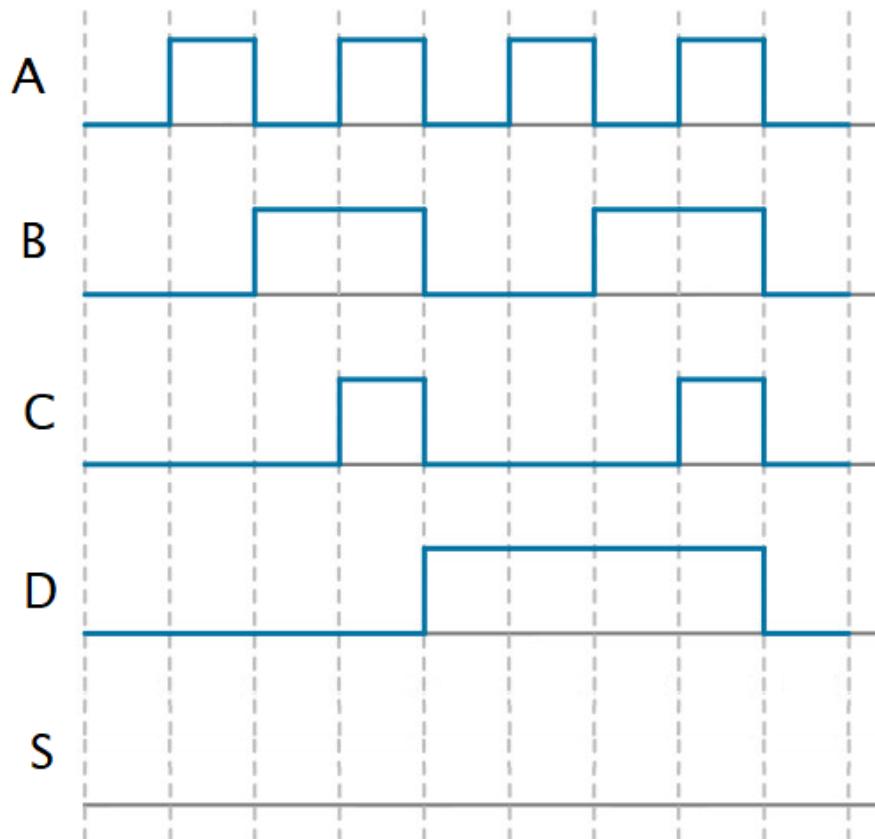


$$S = (A + B).C.(B + D)$$

A	B	C	D	A+B	B+D	(A+B).C.(B+D)
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	1
0	1	0	0	1	0	0
0	1	0	1	1	1	1
0	1	1	0	1	0	0
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	1	1
1	0	1	0	1	0	0
1	0	1	1	1	1	1
1	1	0	0	1	0	0
1	1	0	1	1	1	1
1	1	1	0	1	0	0
1	1	1	1	1	1	1

Descrevendo expressões lógicas a partir de circuitos lógicos

Diagrama de tempo:



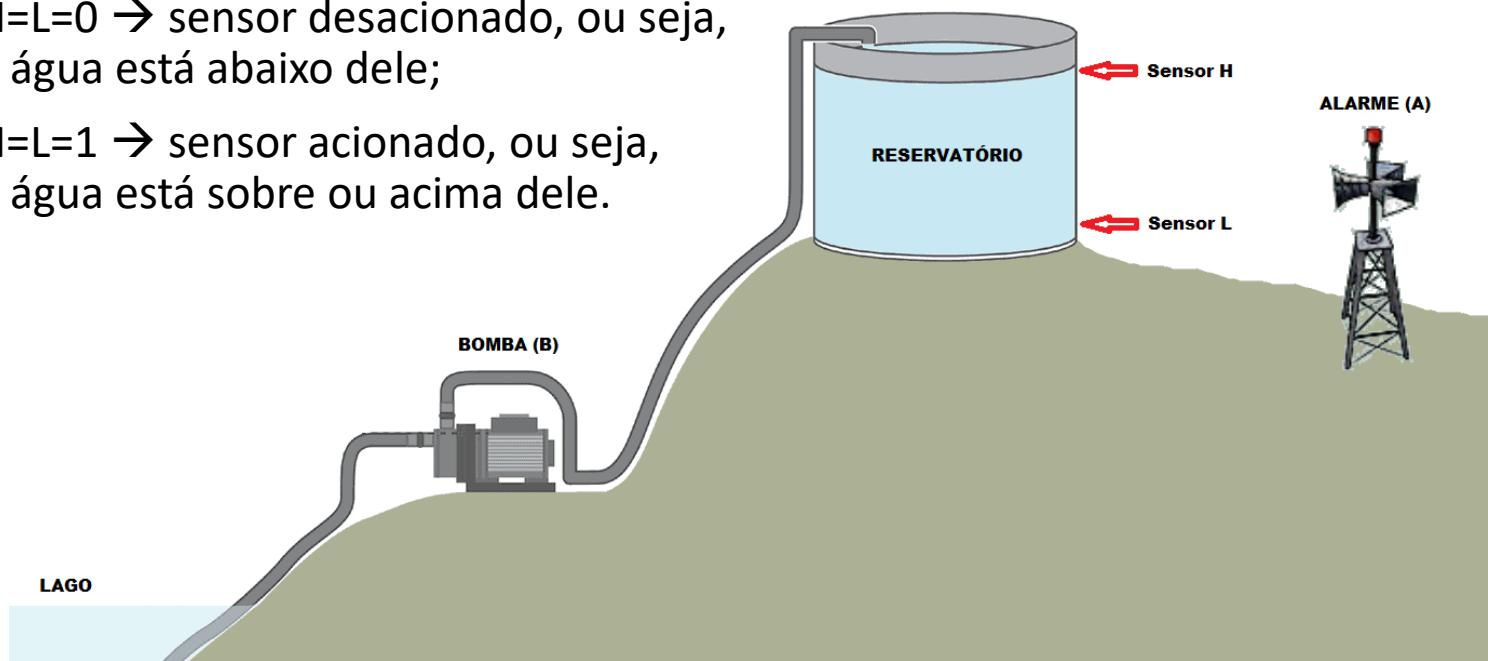
A	B	C	D	A+B	B+D	(A+B).C.(B+D)
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	0	1	1	1	0
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	0	0
1	0	1	1	1	1	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Exemplo - controle de bombeamento de água

Os sensores de nível alto (H) e de nível baixo (L) são utilizados para determinar o acionamento da bomba (B) e do alarme (A).

Os sensores funcionam da seguinte forma:

- $H=L=0 \rightarrow$ sensor desacionado, ou seja, a água está abaixo dele;
- $H=L=1 \rightarrow$ sensor acionado, ou seja, a água está sobre ou acima dele.



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Exemplo - controle de bombeamento de água

A bomba (B) deve ser acionada sempre que o nível de água do reservatório estiver abaixo do sensor H.

Se o nível do reservatório ficar abaixo do nível do sensor L, o alarme (A) deve ser acionado até que o nível do reservatório suba acima de L.

Resolução:

- Identificar as variáveis;
- Determinar quais são as variáveis de entrada;
- Determinar quais são as variáveis de saída;
- Montar a tabela verdade;
- Extrair as expressões lógicas;
- Simplificar as expressões;
- Desenhar o circuito lógico;
- Desenhar (e/ou construir) o circuito eletrônico.

Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Exemplo - controle de bombeamento de água

Tabela verdade:

H	L	B	A
0	0	1	1
0	1	1	0
1	0	X	X
1	1	0	0

Expressões:

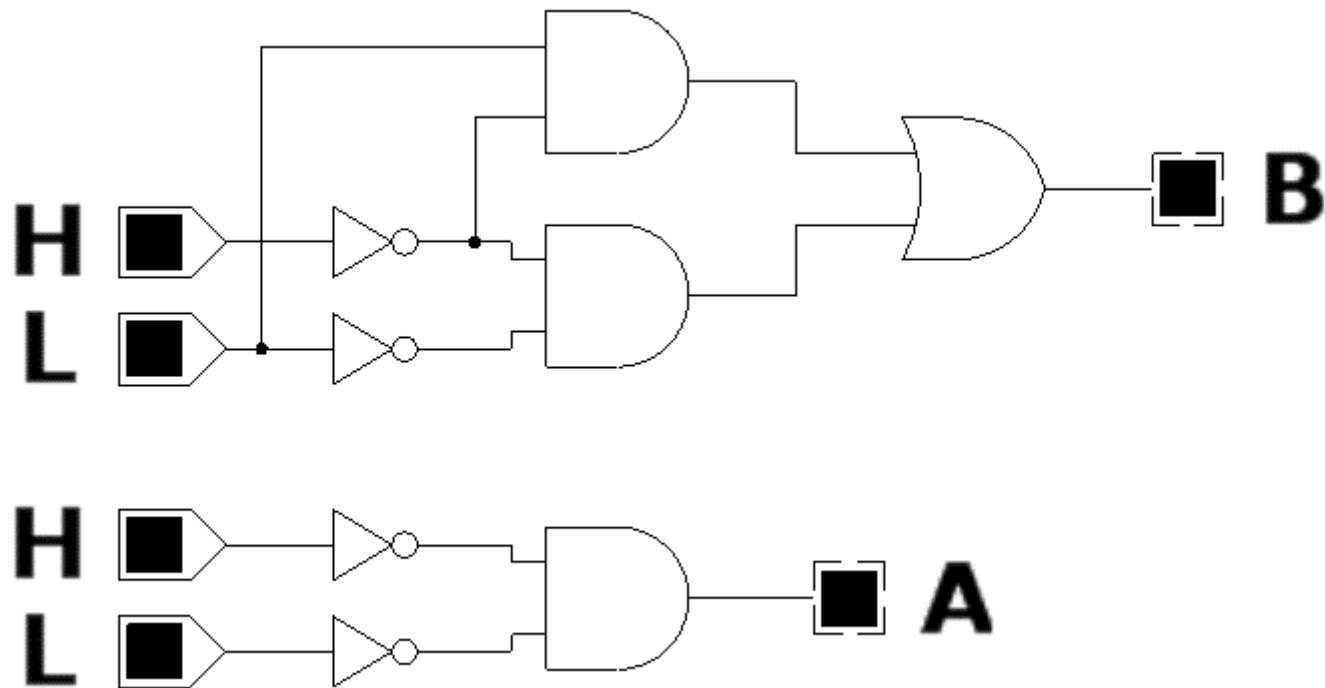
$$B = \bar{H} \cdot \bar{L} + \bar{H} \cdot L$$

$$A = \bar{H} \cdot \bar{L}$$

Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Exemplo - controle de bombeamento de água

Círcuito lógico:



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

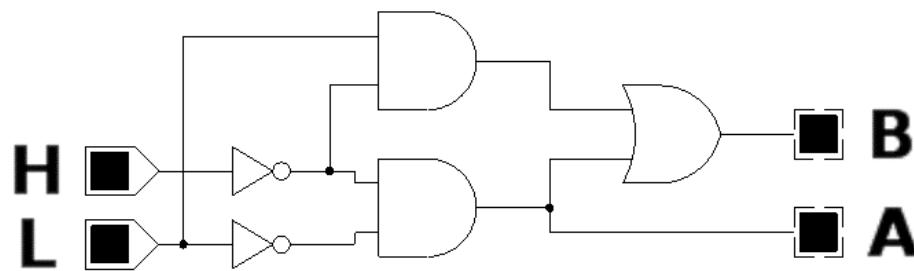
Exemplo - controle de bombeamento de água

Resumo:

H	L	B	A
0	0	1	1
0	1	1	0
1	0	X	X
1	1	0	0

$$B = \bar{H} \cdot \bar{L} + \bar{H} \cdot L$$

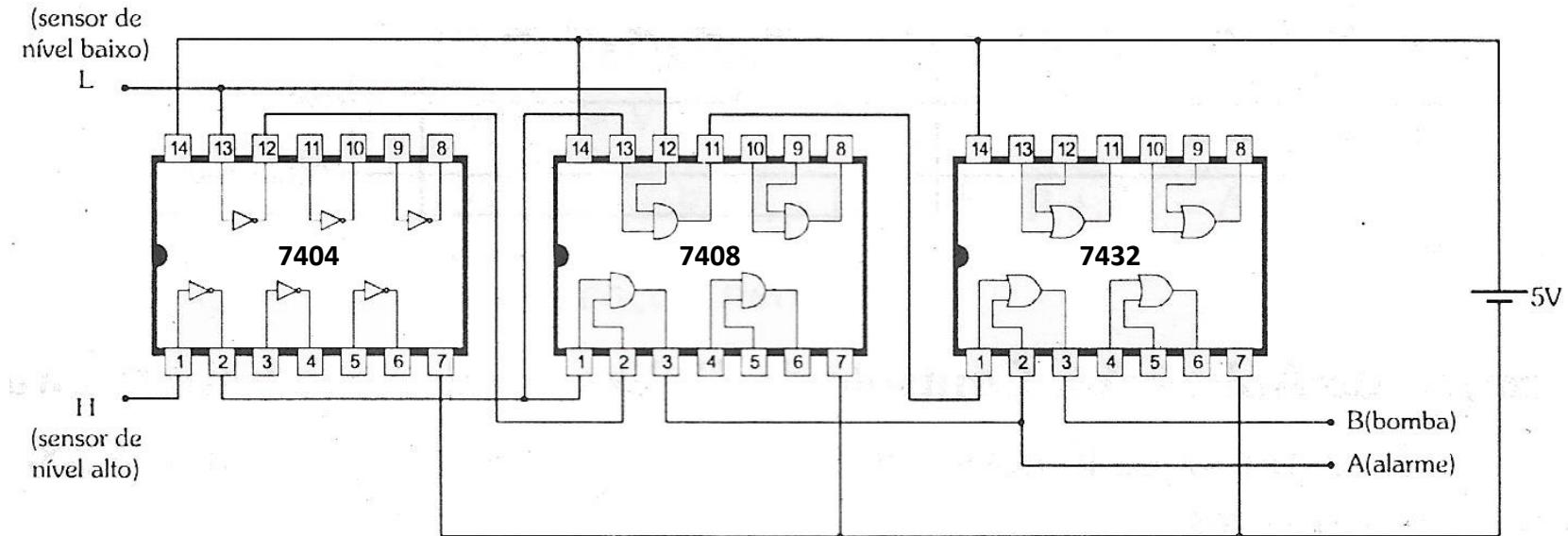
$$A = \bar{H} \cdot \bar{L}$$



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Exemplo - controle de bombeamento de água

Círcuito eletrônico:



Ainda é possível simplificar o circuito utilizando funções equivalentes.

Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Agradecimentos

Este material é baseado nas notas de aula do **Prof. Dr. Jamil Kalil Naufal Júnior**.

Para saber mais...

... leia o Capítulo 3 (Descrevendo Circuitos Lógicos), seção 3.1 (Constantes e variáveis booleanas) até 3.9 (Portas NOR e portas NAND) do livro-texto: TOCCI, R.J., WIDMER, N.S., MOSS, G.L. **Sistemas Digitais: Princípios e Aplicações**. 12^a ed. São Paulo: Pearson, 2018.

Unidade 4

Teoremas

Objetivos

- Conhecer os teoremas booleanos;
- Conhecer os teoremas de De Morgan;
- Praticar a simplificação de expressões booleanas.

Teoremas

Álgebra booleana é uma ferramenta matemática utilizada para análise e projeto de circuitos digitais.

A expressão booleana resultante de uma tabela verdade não se apresenta de imediato em um formato simplificado.

As regras e teoremas booleanos podem ser usados para simplificar a expressão de um circuito lógico e obter uma implementação otimizada.

No domínio da matemática, um teorema é uma proposição que pode ser demonstrada por meio de um processo lógico.

Teorema da complementação

O teorema booleano da complementação estabelece que \bar{A} é complemento de A . Pela complementação, pode-se estabelecer a identidade $\bar{\bar{A}} = A$.

Se $A = 1$, temos: $\bar{A} = 0$ e se $\bar{A} = 0 \rightarrow \bar{\bar{A}} = 1$

Se $A = 0$, temos: $\bar{A} = 1$ e se $\bar{A} = 1 \rightarrow \bar{\bar{A}} = 0$

Teorema da adição (OR)

- **Teorema 1:** adicionar uma variável a um nível lógico ZERO terá como resposta o valor da própria variável (elemento nulo ou identidade).

$$A + 0 = A$$

- **Teorema 2:** adicionar uma variável a um nível lógico UM terá como resposta o valor lógico UM (elemento unitário ou aniquilador).

$$A + 1 = 1$$

Teorema da adição (OR)

- **Teorema 3:** adicionar uma variável a si mesma terá como resposta a própria variável (idempotência).

$$A + A = A$$

- **Teorema 4:** adicionar uma variável ao seu complemento terá como resposta o valor lógico UM (complementariedade).

$$A + \bar{A} = 1$$

Teorema da multiplicação (AND)

- **Teorema 5:** multiplicar uma variável por um nível lógico ZERO terá como resposta o nível lógico ZERO (elemento unitário ou aniquilador).

$$A \cdot 0 = 0$$

- **Teorema 6:** multiplicar uma variável por um nível lógico UM terá como resposta a própria variável (elemento nulo ou identidade).

$$A \cdot 1 = A$$

Teorema da multiplicação (AND)

- **Teorema 7:** multiplicar uma variável por si mesma terá como resposta a própria variável (idempotência).

$$A \cdot A = A$$

- **Teorema 8:** multiplicar uma variável por seu complemento terá como resposta o valor lógico ZERO (complementariedade).

$$A \cdot \bar{A} = 0$$

Propriedades matemáticas

- **Teorema 9:** Comutativa

$$\begin{aligned} A + B &= B + A \\ A \cdot B &= B \cdot A \end{aligned}$$

- **Teorema 10:** Associativa

$$\begin{aligned} A + (B + C) &= (A + B) + C = A + B + C \\ A \cdot (B \cdot C) &= (A \cdot B) \cdot C = A \cdot B \cdot C \end{aligned}$$

- **Teorema 11:** Distributiva

$$\begin{aligned} A(B + C) &= AB + AC \\ (A + B) \cdot (C + D) &= AC + AD + BC + BD \\ A + BC &= (A + B) \cdot (A + C) \end{aligned}$$

Propriedades matemáticas

- **Teorema 12:** Absorção ou Cobertura

Pode ser obtida por meio
do Princípio da Dualidade

$$\begin{aligned} A + AB &= A \\ A(A + B) &= A \end{aligned}$$

A	B	$A+AB$
0	0	0
0	1	0
1	0	1
1	1	1

resultados
equivalentes



- **Teorema 13:** Eliminação

Pode ser obtida por meio
do Princípio da Dualidade

$$\begin{aligned} A + \bar{A}B &= A + B \\ A(\bar{A} + B) &= A \cdot B \end{aligned}$$

A	B	$A+A'B$	$A+B$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

resultados
equivalentes



Propriedades matemáticas

- **Teorema 14:** Consenso ou Fantasma

Pode ser obtida por meio
do Princípio da Dualidade

$$\overline{AB} + \overline{AC} + BC = AB + \overline{AC}$$
$$(A + B) \cdot (\overline{A} + C) \cdot (B + C) = (A + B) \cdot (\overline{A} + C)$$

- **Teorema 15:** Conversão

Pode ser obtida por meio
do Princípio da Dualidade

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

Teoremas de De Morgan

Augustus De Morgan (1806-1871) foi um matemático e lógico britânico.

Formulou as leis de De Morgan e foi o primeiro a introduzir o termo e tornar rigorosa a ideia da indução matemática.

Uma das realizações mais importantes de De Morgan foram o lançamento das fundações de relações e a preparação do caminho para o nascimento da lógica simbólica (ou matemática).



Fonte: wikipedia.org

Teoremas de De Morgan

- **1º Teorema de De Morgan:** o complemento do produto (**NAND**) é igual à soma dos complementos.

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

↑
Porta NAND

A	B	$(A \cdot B)'$	$A' + B'$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

↑
resultados
equivalentes

- **2º Teorema de De Morgan:** o complemento da soma (**NOR**) é igual ao produto dos complementos.

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

↑
Porta NOR

A	B	$(A+B)'$	$A' \cdot B'$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

↑
resultados
equivalentes

XOR e XNOR

- **XOR:** a saída é verdadeira quando as entradas são **diferentes**.

$$A \oplus B = \underbrace{\bar{A} \cdot B + A \cdot \bar{B}}_{\text{FDN}} = \underbrace{(A + B) \cdot (\bar{A} + \bar{B})}_{\text{FCN}}$$

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

- **XNOR:** a saída é verdadeira quando as entradas são **iguais**.

$$A \odot B = \underbrace{\bar{A} \cdot \bar{B} + A \cdot B}_{\text{FDN}} = \underbrace{(A + \bar{B}) \cdot (\bar{A} + B)}_{\text{FCN}}$$

A	B	S
0	0	1
0	1	0
1	0	0
1	1	1



A XNOR também pode ser representada como $\overline{A \oplus B} = A \odot B = \bar{A} \cdot \bar{B} + A \cdot B$



As formas canônicas FDN (Forma Disjuntiva Normal) e FCN (Forma Conjuntiva Normal) serão abordadas na próxima unidade.

Expressão dual

Propriedade importante da Álgebra de Boole, a expressão se mantém válida pela troca dos operadores AND e OR.

A expressão dual pode ser obtida seguindo o procedimento:

- Trocam-se (\cdot) por $(+)$ e $(+)$ por (\cdot) ;
- Trocam-se (0) por (1) e (1) por (0) .

Mantêm-se as prioridades da expressão original pela adição ou remoção de parêntesis.



O **Princípio da Dualidade** afirma que qualquer **igualdade algébrica** derivada desses axiomas sempre será válida.

Expressão dual - exemplos

$$F = A \cdot B + C \cdot D + B \cdot 1$$

$$F_{\text{dual}} = (A+B) \cdot (C+D) \cdot (B+0)$$

$$X = KLM + NT(R+S)$$

$$X_{\text{dual}} = (K+L+M) \cdot (N+T+RS)$$



De acordo com o **Princípio da Dualidade**, as expressões acima permanecem válidas, mas não necessariamente F é igual a F_{dual} e nem X é igual a X_{dual} .

Expressão dual - aplicação

O **Princípio da Dualidade** pode ser melhor entendido nas **igualdades algébricas**, onde é usado para que se possa deduzir novas expressões. Este princípio deve sempre ser aplicado nos dois lados igualdade:

- **Teorema 14:** Consenso ou Fantasma

$$\begin{aligned} AB + \bar{A}C + BC &= AB + \bar{A}C \\ \text{Expressão Dual} \rightarrow (A + B) \cdot (\bar{A} + C) \cdot (B + C) &= (A + B) \cdot (\bar{A} + C) \end{aligned}$$

- **Teorema 15:** Conversão

$$\begin{aligned} AB + \bar{A}C &= (A + C) \cdot (\bar{A} + B) \\ \text{Expressão Dual} \rightarrow (A + B) \cdot (\bar{A} + C) &= AC + \bar{A}B \end{aligned}$$

Expressão complementar

Servem para obter o valor complementar ou negado da expressão original.

A expressão complementar pode ser obtida seguindo o procedimento:

- Trocam-se (.) por (+) e (+) por (.);
- Trocam-se (0) por (1) e (1) por (0);
- Complementam-se todas as variáveis.

Mantêm-se as prioridades da expressão original pela adição ou remoção de parêntesis.



A expressão complementar se vale do **Princípio da Dualidade** com a adição da complementação de todas as variáveis.

Expressão complementar - exemplos

$$F = A \cdot B + C \cdot D + B \cdot 1$$

$$F' = (A' + B') \cdot (C' + D') \cdot (B' + 0)$$

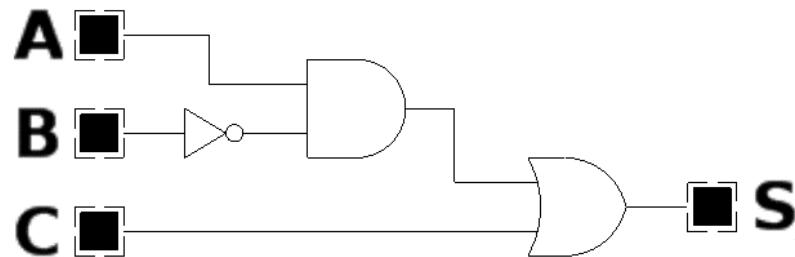
$$S = AB'C + C(D' + E)$$

$$S' = (A' + B + C') \cdot (C' + DE')$$

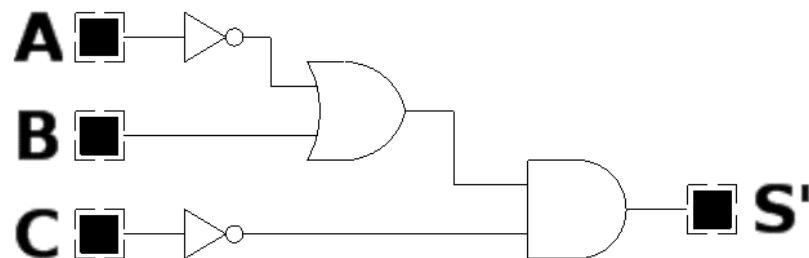


A expressão complementar sempre será o inverso da expressão original, ou seja, nos exemplos acima o complementar de F será F' e o complementar de S será S'.

Expressão complementar - aplicação



$$S = A \cdot \bar{B} + C$$



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

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

Simplificação de circuitos

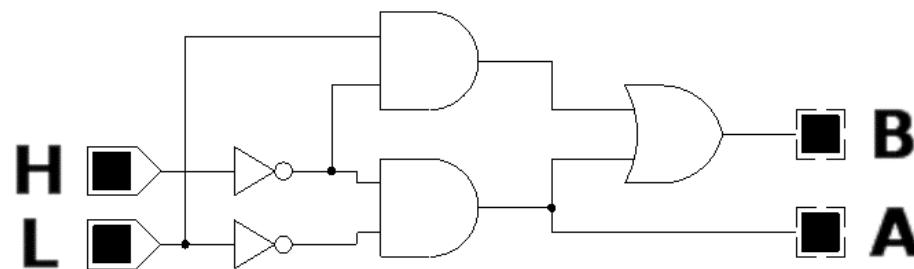
Exemplo - controle de bombeamento de água

Resumo:

H	L	B	A
0	0	1	1
0	1	1	0
1	0	X	X
1	1	0	0

$$B = \bar{H} \cdot \bar{L} + \bar{H} \cdot L$$

$$A = \bar{H} \cdot \bar{L}$$



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Simplificação de circuitos

Exemplo - controle de bombeamento de água

Simplificando expressões:

- Aplicando-se os teoremas na expressão B, tem-se:

$$\begin{aligned} B &= \bar{H} \cdot \bar{L} + \bar{H} \cdot L && \xleftarrow{\text{Distributiva}} \\ B &= \bar{H} \cdot (\bar{L} + L) && \xleftarrow{\text{Teorema da adição}} \\ B &= \bar{H} \cdot 1 && \xleftarrow{\text{Teorema da multiplicação}} \\ B &= \bar{H} \end{aligned}$$

- Aplicando-se o teorema de De Morgan na expressão A, tem-se:

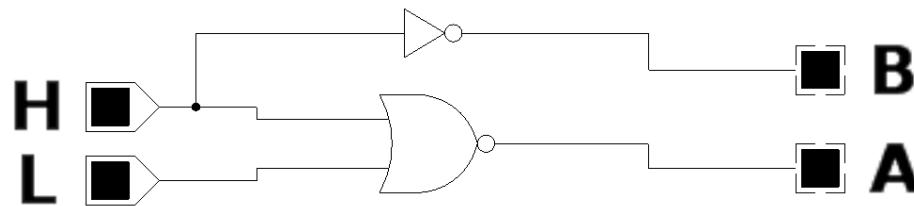
$$\begin{aligned} A &= \bar{H} \cdot \bar{L} && \xleftarrow{\text{2º Teorema de De Morgan}} \\ A &= \overline{H + L} \end{aligned}$$

Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

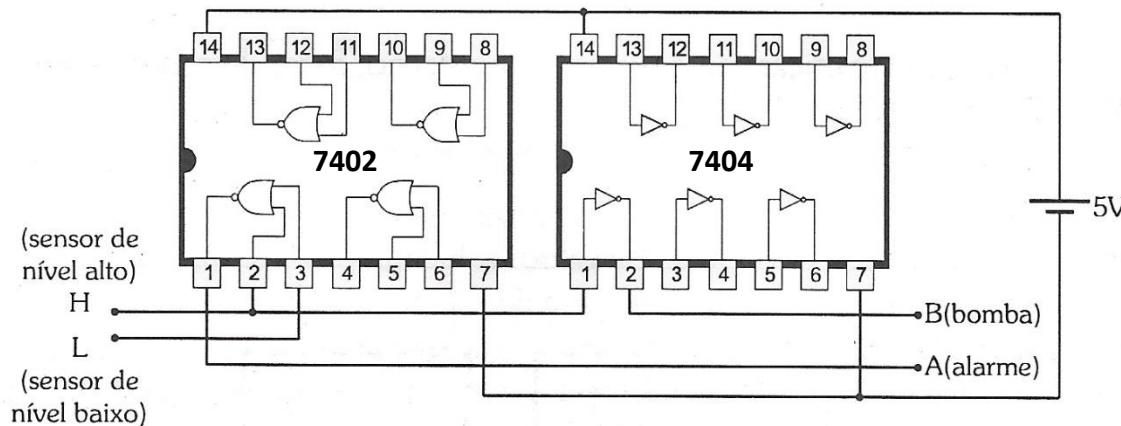
Simplificação de circuitos

Exemplo - controle de bombeamento de água

Círcuito lógico:



Círcuito eletrônico:



Ainda é possível simplificar o circuito utilizando portas universais.

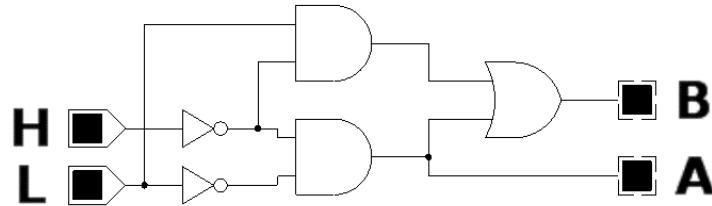
Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Simplificação de circuitos

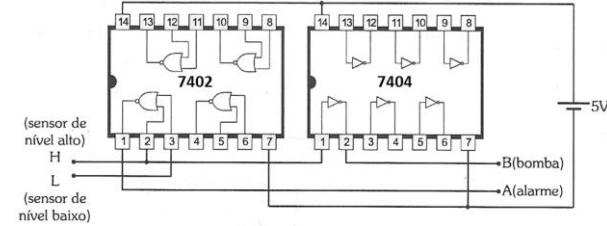
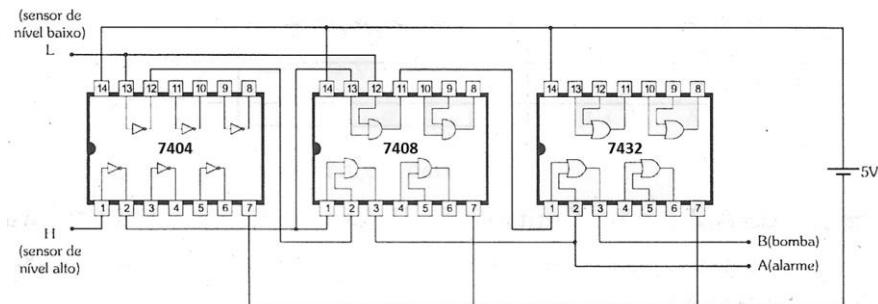
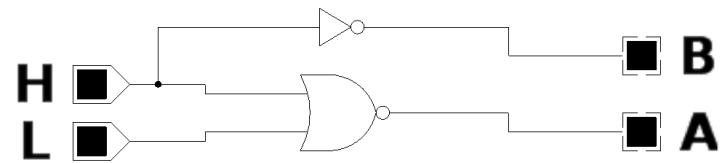
Exemplo - controle de bombeamento de água

Resumo:

Sem simplificação



Com simplificação



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Agradecimentos

Este material é baseado nas notas de aula do **Prof. Dr. Jamil Kalil Naufal Júnior**.

Para saber mais...

... leia o Capítulo 3 (Descrevendo Circuitos Lógicos), seção 3.10 (Teoremas Booleanos) até 3.11 (Teoremas de De Morgan do livro-texto: TOCCI, R.J., WIDMER, N.S., MOSS, G.L. **Sistemas Digitais: Princípios e Aplicações**. 12^a ed. São Paulo: Pearson, 2018.

Unidade 5

Formas canônicas

Objetivos

- Conhecer a extração de expressões lógicas a partir de tabelas;
- Conhecer mintermos e maxtermos;
- Praticar a obtenção de expressões por mintermos e maxtermos.

Expressões lógicas por tabelas

Uma expressão lógica pode ser obtida a partir de uma tabela verdade. As duas formas básicas para expressões lógicas são:

- Forma disjuntiva normal **FDN (soma de produtos)**, ou **SdP**; e
- Forma conjuntiva normal **FCN (produto de somas)**, ou **PdS**.

Estas são formas duais de representação e de projetos em circuitos digitais.



Métodos de simplificação de projetos utilizam usualmente a forma disjuntiva normal **FDN (soma de produtos)**.

Forma disjuntiva normal (soma de produtos)

Na forma disjuntiva normal FDN, a expressão booleana leva em consideração apenas as linhas da tabela verdade onde a saída é igual ao valor lógico um.

Cada expressão consiste em dois ou mais termos AND (produtos) conectados por uma operação OR (soma).

Cada termo AND consiste em uma ou mais variáveis que aparecem individualmente na tabela verdade.

Quando a variável de entrada possui o valor lógico zero, a sua leitura é feita como uma variável negada.

Quando a variável de entrada possui o valor lógico um, a sua leitura é feita como uma variável não negada.

A	B	S
0	0	0
0	1	1
1	0	0
1	1	1

$$S_{FDN} = (\bar{A} \cdot B) + (A \cdot B)$$

Se houver um “X” (condição irrelevante) na saída, ele é considerado como valor lógico zero.

Forma disjuntiva normal (soma de produtos)

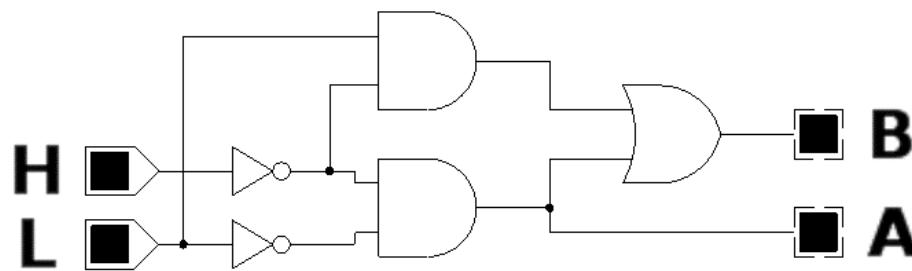
Exemplo - controle de bombeamento de água

Soma de produtos:

H	L	B	A
0	0	1	1
0	1	1	0
1	0	X	X
1	1	0	0

$$B = \bar{H} \cdot \bar{L} + \bar{H} \cdot L$$

$$A = \bar{H} \cdot \bar{L}$$



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Forma conjuntiva normal (produto de somas)

Na forma conjuntiva normal FCN, a expressão booleana leva em consideração apenas as linhas da tabela verdade onde a saída é igual ao valor lógico zero.

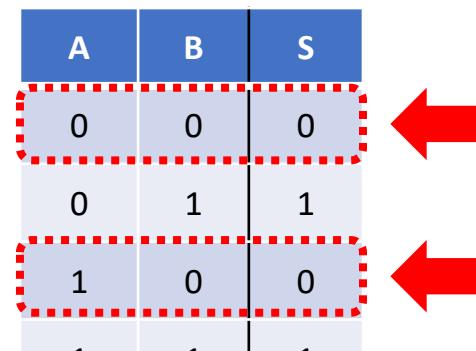
Cada expressão consiste em dois ou mais termos OR (somas) conectados por uma operação AND (produto).

Cada termo OR consiste em uma ou mais variáveis que aparecem individualmente na tabela verdade.

Quando a variável de entrada possui o valor lógico zero, a sua leitura é feita como uma variável não negada.

Quando a variável de entrada possui o valor lógico um, a sua leitura é feita como uma variável negada.

A	B	S
0	0	0
0	1	1
1	0	0
1	1	1



$$S_{FCN} = (A + B) \cdot (\bar{A} + B)$$

Se houver um "X" (condição irrelevante) na saída, ele é considerado como valor lógico zero.

Forma conjuntiva normal (produto de somas)

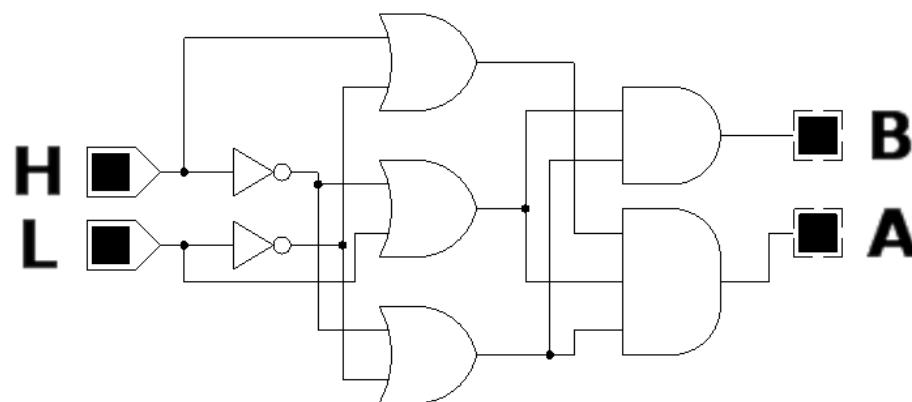
Exemplo - controle de bombeamento de água

Produto de somas:

H	L	B	A
0	0	1	1
0	1	1	0
1	0	X	X
1	1	0	0

$$B = (\bar{H} + L) \cdot (\bar{H} + \bar{L})$$

$$A = (H + \bar{L}) \cdot (\bar{H} + L) \cdot (\bar{H} + \bar{L})$$



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Forma disjuntiva vs forma conjuntiva

- Forma disjuntiva normal:

A	B	S
0	0	0
0	1	1
1	0	0
1	1	1

- Forma conjuntiva normal:

A	B	S
0	0	0
0	1	1
1	0	0
1	1	1

$$S_{FDN} = (\bar{A} \cdot B) + (A \cdot \bar{B})$$

$$S_{FCN} = (A + B) \cdot (\bar{A} + \bar{B})$$

Apesar da aparência, não se tratam de expressões que seguem a aplicação do princípio da dualidade. Note as expressões da próxima página e compare.

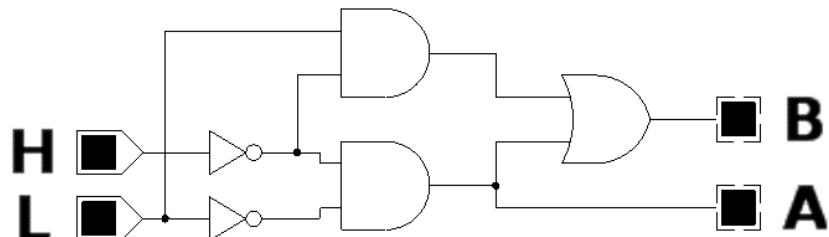
Controle de bombeamento de água - resumo

Resumo:

Soma de produtos

$$B = \bar{H} \cdot \bar{L} + \bar{H} \cdot L$$

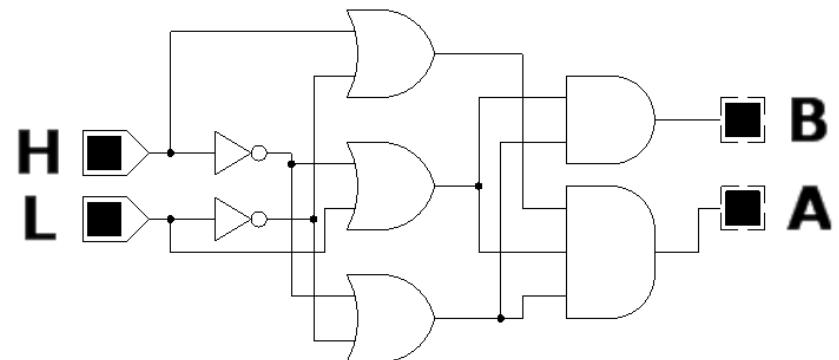
$$A = \bar{H} \cdot \bar{L}$$



Produto de somas

$$B = (\bar{H} + L) \cdot (\bar{H} + \bar{L})$$

$$A = (H + \bar{L}) \cdot (\bar{H} + L) \cdot (\bar{H} + \bar{L})$$



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Mintermos e maxtermos

A tabela verdade pode ser lida nas formas disjuntiva (FDN) e conjuntiva (FCN) normais.

As leituras podem sofrer alguma simplificação utilizando teoremas para a criação de circuitos digitais otimizados.

Quando não há simplificação, a expressão pode ser expressa utilizando as variáveis de entrada ou podem ser expressas pela posição de cada leitura na tabela verdade.

Mintermos

Mintermos é equivalente a leitura por forma disjuntiva normal **FDN (soma de produtos)**:

$$F = \sum_{i=0}^{2^n - 1} a_i \cdot m_i$$

A forma mais usual e conveniente de representar a função por mintermos:

$$F = \sum m(\text{indices})$$

Maxtermos

Maxtermos é equivalente a leitura por forma conjuntiva normal **FCN (produto de somas)**:

$$F = \prod_{i=0}^{2^n-1} (a_i + m_i)$$

A forma mais usual e conveniente de representar a função por maxtermos:

$$F = \prod M(\text{indices})$$

Mintermos e maxtermos

índice ↴

i	A	B	C	Mintermo	Maxtermo	F
0	0	0	0	$\bar{A} \cdot \bar{B} \cdot \bar{C} = m_0$	$(A + B + C) = M_0$	a_0
1	0	0	1	$\bar{A} \cdot \bar{B} \cdot C = m_1$	$(A + B + \bar{C}) = M_1$	a_1
2	0	1	0	$\bar{A} \cdot B \cdot \bar{C} = m_2$	$(A + \bar{B} + C) = M_2$	a_2
3	0	1	1	$\bar{A} \cdot B \cdot C = m_3$	$(A + \bar{B} + \bar{C}) = M_3$	a_3
4	1	0	0	$A \cdot \bar{B} \cdot \bar{C} = m_4$	$(\bar{A} + B + C) = M_4$	a_4
5	1	0	1	$A \cdot \bar{B} \cdot C = m_5$	$(\bar{A} + B + \bar{C}) = M_5$	a_5
6	1	1	0	$A \cdot B \cdot \bar{C} = m_6$	$(\bar{A} + \bar{B} + C) = M_6$	a_6
7	1	1	1	$A \cdot B \cdot C = m_7$	$(\bar{A} + \bar{B} + \bar{C}) = M_7$	a_7

 A informação do número de variáveis envolvidas pode não ser explícita, e deve-se considerar o menor número possível.

Lembrar que para n variáveis há 2^n combinações possíveis.

Mintermos - exemplo

índice

	W	X	Y	Z	F_{wxyz}
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

Forma disjuntiva normal **FDN (soma de produtos)**

$$F = \sum_{i=0}^{2^{n-1}} a_i \cdot m_i$$

$$F(W, X, Y, Z) = \sum m(1, 2, 4, 5, 7, 10, 12, 14, 15)$$

Maxtermos - exemplo

índice

	W	X	Y	Z	F_{wxyz}
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

Forma conjuntiva normal **FCN (produto de somas)**

$$F = \prod_{i=0}^{2^{n-1}} (a_i + m_i)$$

$$F(W, X, Y, Z) = \prod M(0, 3, 6, 8, 9, 11, 13)$$

Mintermos e maxtermos - resumo

índice

	W	X	Y	Z	F_{wxyz}
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

Forma disjuntiva normal **FDN (soma de produtos)**

$$F(W, X, Y, Z) = \sum m(1, 2, 4, 5, 7, 10, 12, 14, 15)$$

Forma conjuntiva normal **FCN (produto de somas)**

$$F(W, X, Y, Z) = \prod M(0, 3, 6, 8, 9, 11, 13)$$

Agradecimentos

Este material é baseado nas notas de aula do **Prof. Dr. Jamil Kalil Naufal Júnior**.

Para saber mais...

... leia o Capítulo 4 (Circuitos Lógicos Combinacionais), seções 4.1 (Forma de Soma-de-Produtos), 4.2 (Simplificação de Circuitos Lógicos), 4.2 (Simplificação Algébrica) e 4.3 (Projetando Circuitos Combinacionais) do livro-texto: TOCCI, R.J., WIDMER, N.S., MOSS, G.L. **Sistemas Digitais: Princípios e Aplicações**. 12^a ed. São Paulo: Pearson, 2018.

Unidade 6

Mapas de Veitch-Karnaugh

Objetivos

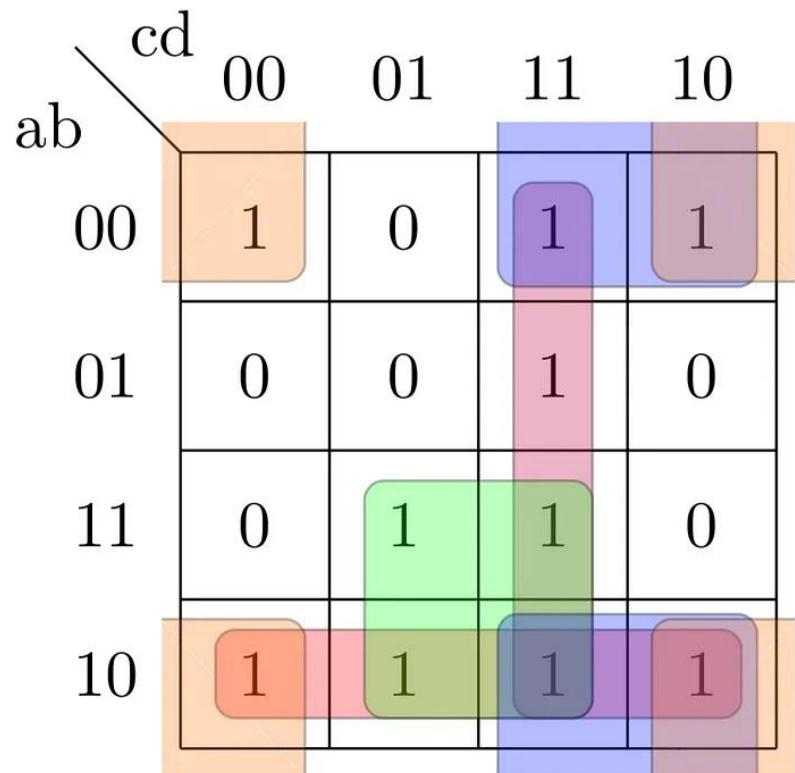
- Conhecer a extração de expressões lógicas a partir de tabelas utilizando o método de Veitch-Karnaugh;
- Praticar a minimização mapas de Veitch-Karnaugh.

Introdução

Mapa Veitch-Karnaugh é um método de minimização criado por Edward Westbrook Veitch em 1952 e aperfeiçoado por Maurice Karnaugh em 1953.

Trata-se de um método gráfico para simplificação de expressões booleanas, onde os valores da tabela verdade são dispostos em um formato matricial e um mapeamento biunívoco é aplicado para determinar a sua função.

A leitura e análise do mapa é realizada com base nas proximidades dos dados e nas adjacências formadas.



Minimização por agrupamento

O método consiste no seguinte:

- Localiza-se todos os “1” do mapa;
- Agrupa-se o maior número possível de células adjacentes (horizontal ou vertical) contendo os 1s num único agrupamento. Inicia-se sempre do maior número em grupos de 2^n : 8, 4, 2 e 1;
- Cada agrupamento fornece um produto de variáveis de entrada, retirando as variáveis que mudam de valor na adjacência;
- O número de enlaces termina quando todos os “1” forem considerados.

É permitido utilizar “1” que já foram considerados em outros agrupamentos a fim de criar novos grupos com um número maior de “1”.



Não é permitido realizar agrupamentos na diagonal e nem com quantidade de “1” diferente de 2^n .

Mapa de 2 variáveis

Formato para a criação das tabelas para **duas** variáveis:

	B	\bar{B}
\bar{A}		
A		

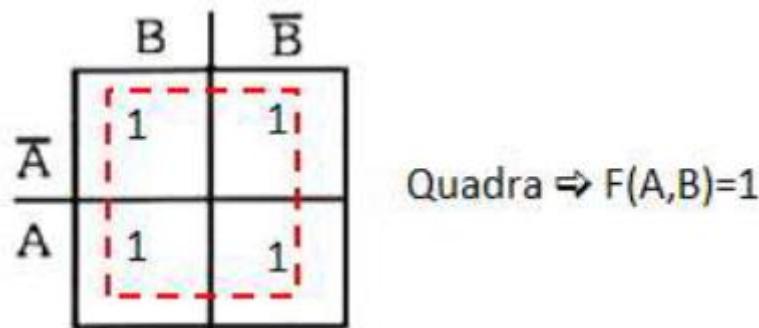
ou

A	B	0	1
0			
1			

Mapa de 2 variáveis

Tipos de agrupamento:

- **Quadras:** agrupamento máximo, onde todas as localidades valem 1.



Mapa de 2 variáveis

Tipos de agrupamento:

- **Pares:** agrupamento de pares com 2 variáveis adjacentes que valem 1.

	B	\bar{B}
\bar{A}	1	0
A	1	0

$$\text{Par} \Rightarrow F(A,B) = B$$

	B	\bar{B}
\bar{A}	1	1
A	0	0

$$\text{Par} \Rightarrow F(A,B) = A'$$

	B	\bar{B}
\bar{A}	0	1
A	0	1

$$\text{Par} \Rightarrow F(A,B) = B'$$

	B	\bar{B}
\bar{A}	0	0
A	1	1

$$\text{Par} \Rightarrow F(A,B) = A$$

Mapa de 2 variáveis

Tipos de agrupamento:

- **Termos isolados:** casos sem simplificação.

	B	\bar{B}
\bar{A}	1	0
A	0	0

$$F(A,B) = A'B$$

	B	\bar{B}
\bar{A}	1	0
A	0	1

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

	B	\bar{B}
\bar{A}	0	1
A	0	0

$$F(A,B) = A'B'$$

	B	\bar{B}
\bar{A}	0	0
A	1	0

$$F(A,B) = AB$$

Exemplo - controle de bombeamento de água

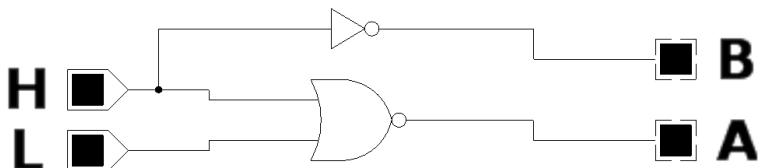
H	L	B	A
0	0	1	1
0	1	1	0
1	0	X	X
1	1	0	0

Saída B

L	L'
H'	1 1
H	0 X

Saída A

L	L'
H'	0 1
H	0 X



$$B = \bar{H}$$

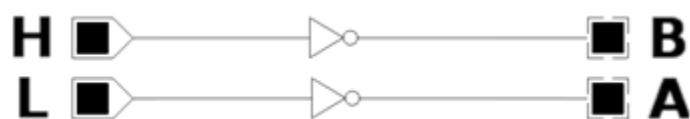
Exemplo - controle de bombeamento de água

Análise do circuito para a saída “A”

H	L	B	A
0	0	1	1
0	1	1	0
1	0	X	X
1	1	0	0

Saída A

L	L'	
H'	0	1
H	0	X



$$A = \bar{H} \cdot \bar{L}$$

<- ou ->

Saída A

L	L'	
H'	0	1
H	0	X

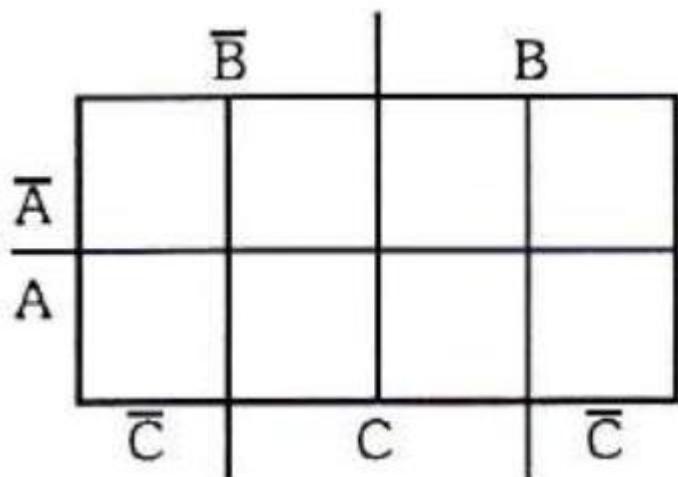
$$A = \bar{L}$$



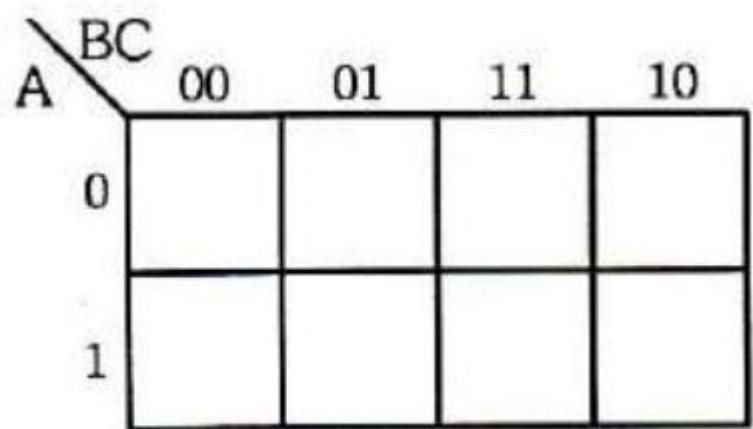
Para a saída onde o valor é igual a “X”, a decisão de incluí-lo ou não no agrupamento deve se dar de acordo com o funcionamento desejado no circuito.

Mapa de 3 variáveis

Formato para a criação das tabelas para **três** variáveis:



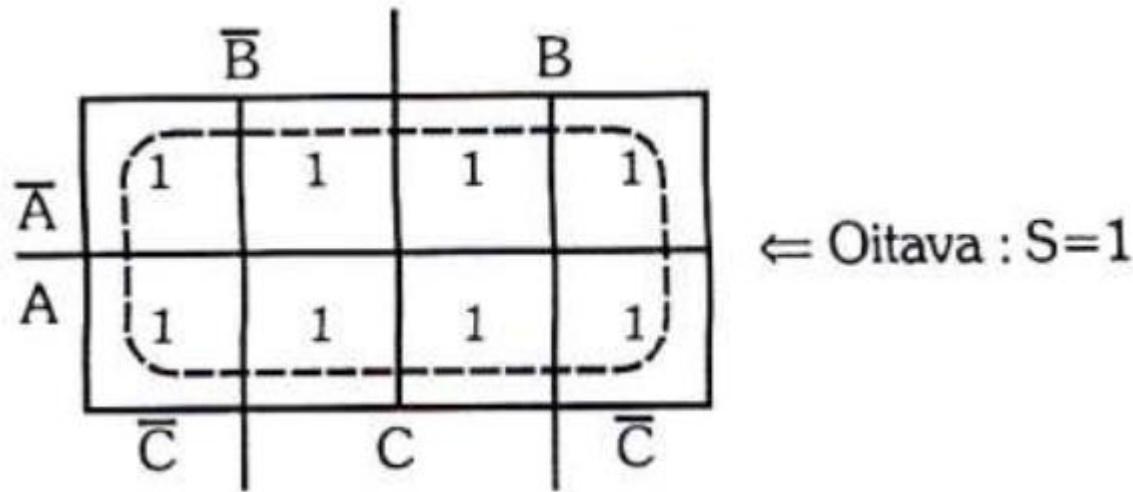
ou



Mapa de 3 variáveis

Tipos de agrupamento:

- **Oitava:** agrupamento máximo, onde todas as localidades valem 1.



Mapa de 3 variáveis

Tipos de agrupamento:

- **Quadras:** agrupamentos de quatro regiões onde S é igual a 1, adjacentes ou em sequência.

	\bar{B}		B	
\bar{A}	1	1	1	1
A	0	0	0	0
	\bar{C}	C	C	\bar{C}

Quadra \bar{A}

	\bar{B}		B	
\bar{A}	1	1	0	0
A	1	1	0	0
	\bar{C}	C	C	\bar{C}

Quadra \bar{B}

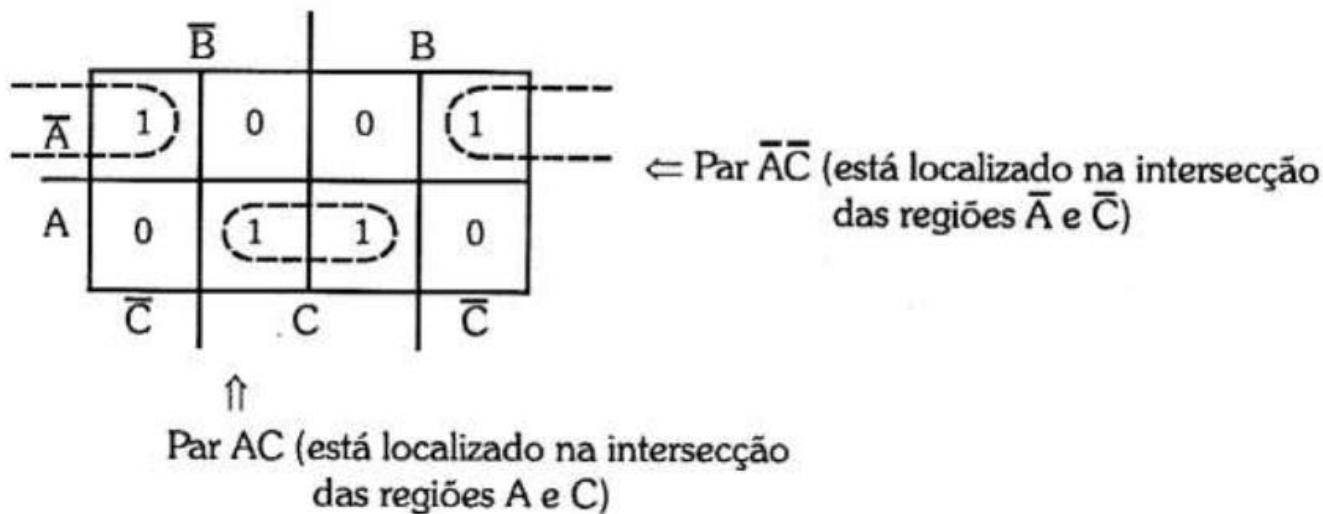
	\bar{B}		B	
\bar{A}	1	0	0	1
A	1	0	0	1
	\bar{C}	C	C	\bar{C}

Quadra \bar{C}

Mapa de 3 variáveis

Tipos de agrupamento:

- **Pares:** agrupamentos de duas regiões onde S é igual a 1, adjacentes ou em sequência.



A figura mostra dois exemplos de pares entre os doze possíveis em um diagrama de três variáveis.

Mapa de 3 variáveis

Tipos de agrupamento:

- **Termos isolados:** casos sem simplificação.

	\bar{B}		B
\bar{A}	0	(1)	0
A	0	0	(1)
\bar{C}	C	C	\bar{C}

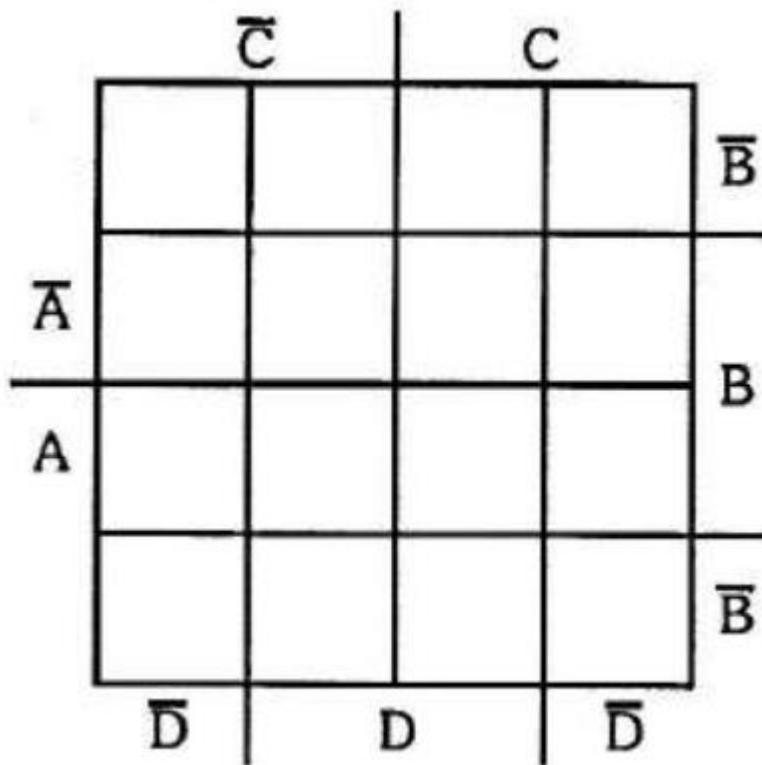
↑
Termo $\bar{A} \bar{B} C$

\Leftarrow Termo $\bar{A} B \bar{C}$

\Leftarrow Termo $A B C$

Mapa de 4 variáveis

Formato para a criação das tabelas para **quatro** variáveis:



ou

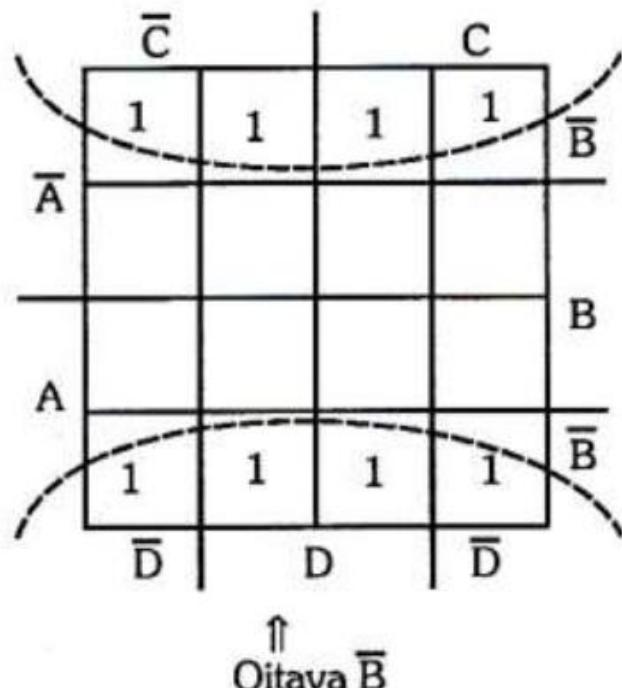
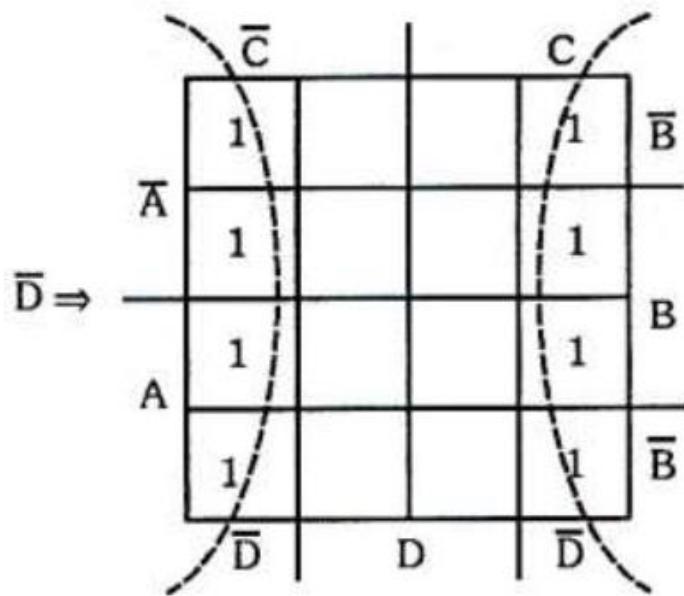
An equivalent representation of the 4-variable Karnaugh map as a truth table. The columns represent variable pairs: AB (leftmost), CD (top), and the bottom two columns (00, 01, 11, 10).

AB	CD	00	01	11	10
00	00				
01	01				
11	11				
10	10				

Mapa de 4 variáveis

Tipos de agrupamento:

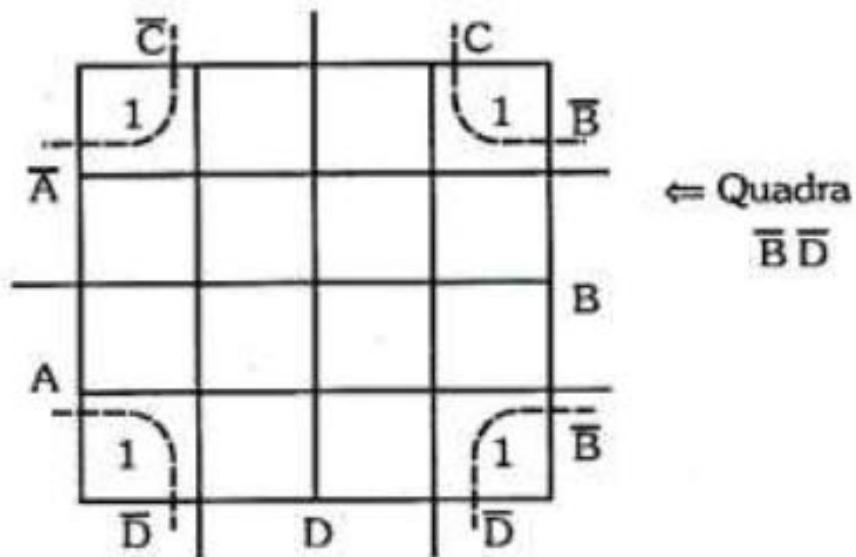
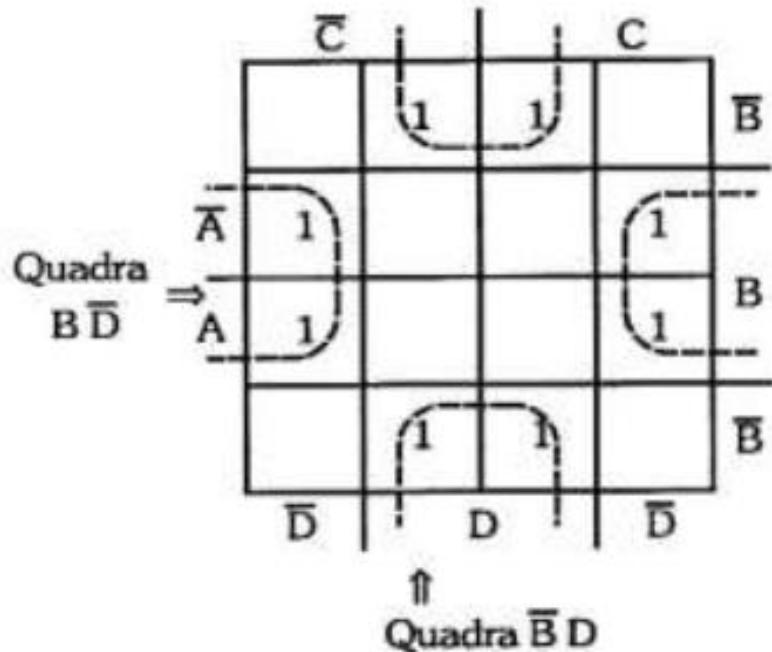
- Oitavas: exemplos



Mapa de 4 variáveis

Tipos de agrupamento:

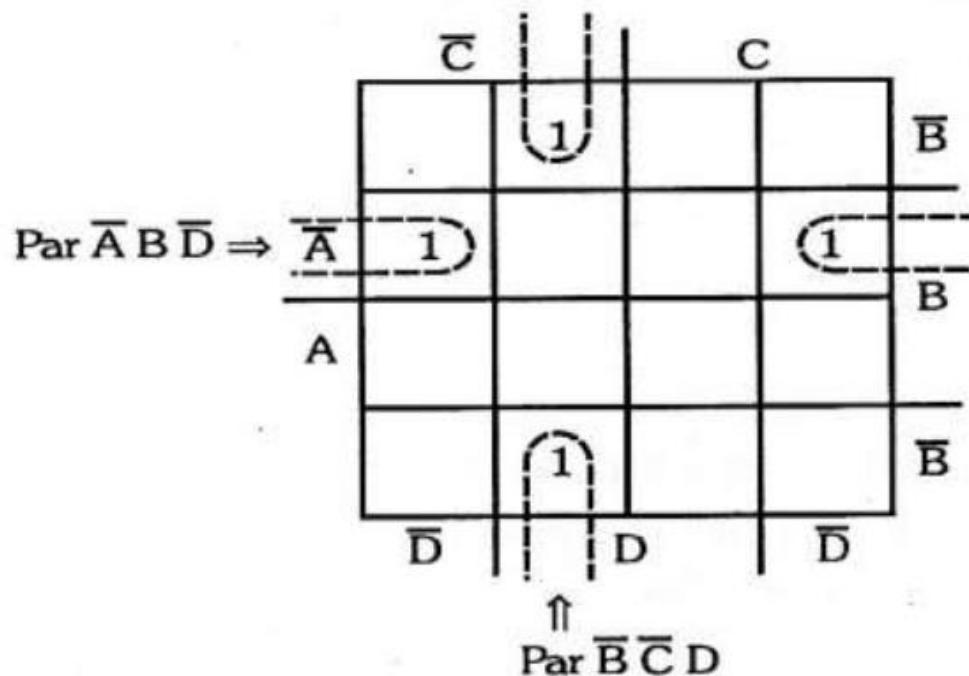
- Quadras: exemplos



Mapa de 4 variáveis

Tipos de agrupamento:

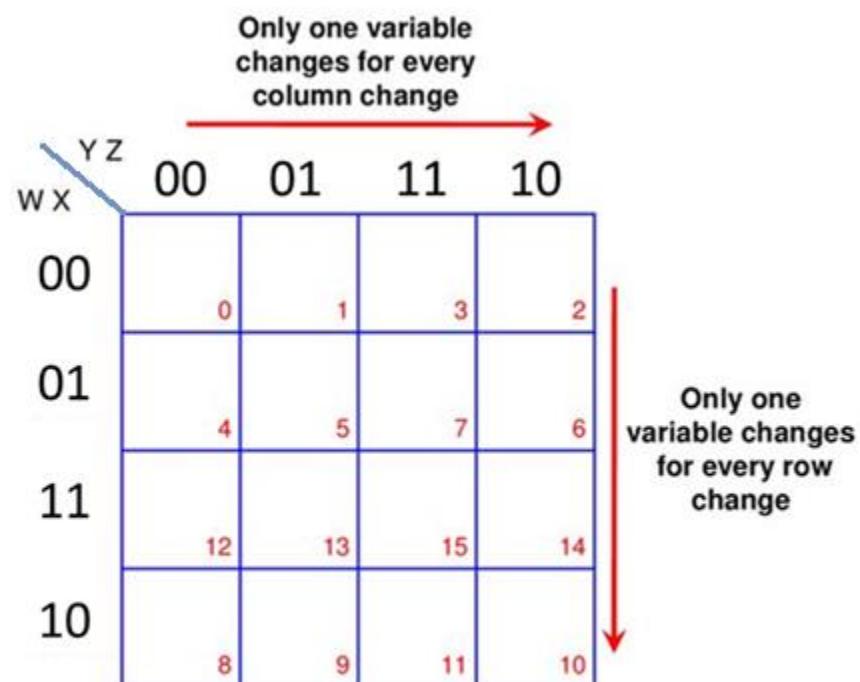
- **Pares:** exemplos



Mapa de 4 variáveis - exemplo

índice →

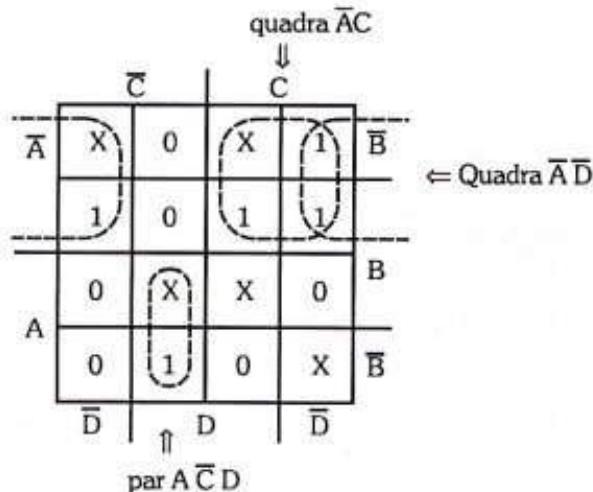
	W	X	Y	Z	F_{wxyz}
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1



Condições irrelevantes

Uma tabela verdade pode ter várias condições irrelevantes que devem ser consideradas de forma independente, de acordo com o agrupamento onde se encontram.

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

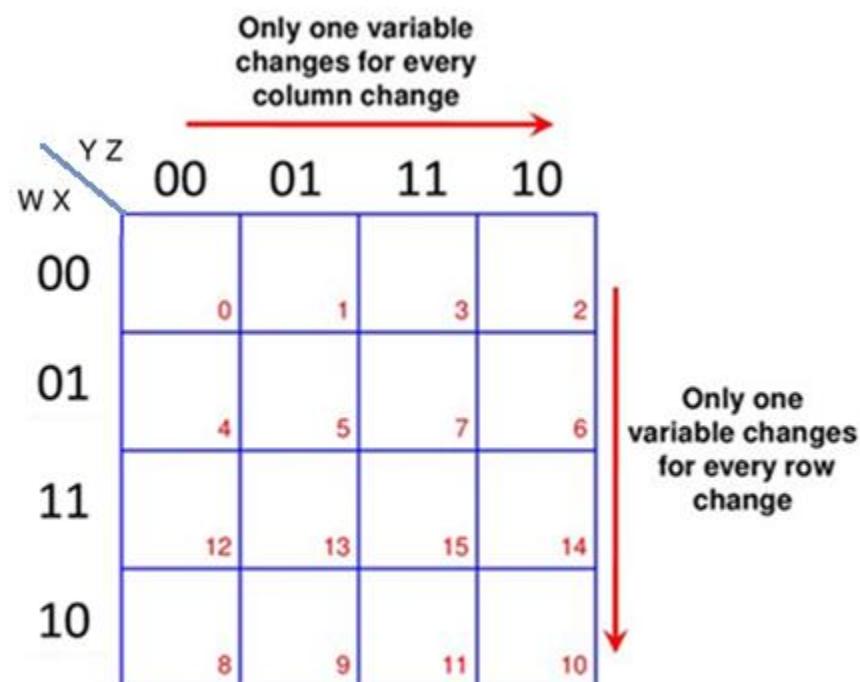


A inclusão ou não da condição irrelevante se dá de acordo com a conveniência.

Condições irrelevantes - exemplo

índice →

	W	X	Y	Z	F_{wxxyz}
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	-
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	-
14	1	1	1	0	1
15	1	1	1	1	0



Mapa de Veitch-Karnaugh - XOR e XNOR

- **XOR:** a saída é verdadeira quando as entradas são **diferentes**.

$$A \oplus B = \bar{A} \cdot B + A \cdot \bar{B} = (A + B) \cdot (\bar{A} + \bar{B})$$

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

- **XNOR:** a saída é verdadeira quando as entradas são **iguais**.

$$A \odot B = \bar{A} \cdot \bar{B} + A \cdot B = (A + \bar{B}) \cdot (\bar{A} + B)$$

A	B	S
0	0	1
0	1	0
1	0	0
1	1	1



A XNOR também pode ser representada como $\overline{A \oplus B} = A \odot B = \bar{A} \cdot \bar{B} + A \cdot B$



As formas canônicas FDN (Forma Disjuntiva Normal) e FCN (Forma Conjuntiva Normal) serão abordadas na próxima unidade.

Mapa de Veitch-Karnaugh - XOR e XNOR

		B	
		0	1
A		0	0
		1	1

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

		B	
		0	1
A		0	1
		1	0

$$\bar{A}\bar{B} + AB = A \odot B = \overline{A \oplus B}$$

Mapa de Veitch-Karnaugh - XOR e XNOR

		B C			
		00	01	11	10
A	0	0	1	0	1
	1	1	0	1	0

$$\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC = A \oplus B \oplus C$$

		B C			
		00	01	11	10
A	0	1	0	1	0
	1	0	1	0	1

$$\bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C + ABC = A \odot B \odot C$$

Mapa de Veitch-Karnaugh - XOR e XNOR

		CD			
		00	01	11	10
AB	00	0	1	0	1
	01	1	0	1	0
	11	0	1	0	1
	10	1	0	1	0

		CD			
		00	01	11	10
AB	00	1	0	1	0
	01	0	1	0	1
	11	1	0	1	0
	10	0	1	0	1

$$\begin{aligned}
 & \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \\
 & \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + \\
 & AB\bar{C}D + ABC\bar{D} + \\
 & A\bar{B}\bar{C}\bar{D} + A\bar{B}CD = A \oplus B \oplus C \oplus D
 \end{aligned}$$

$$\begin{aligned}
 & \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}CD + \\
 & \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \\
 & AB\bar{C}\bar{D} + ABCD + \\
 & A\bar{B}\bar{C}D + A\bar{B}C\bar{D} = A \odot B \odot C \odot D
 \end{aligned}$$

Agradecimentos

Este material é baseado nas notas de aula do **Prof. Dr. Jamil Kalil Naufal Júnior**.

Para saber mais...

... leia o Capítulo 4 (Circuitos Lógicos Combinacionais), seção 4.5 (Método do Mapa de Karnaugh) do livro-texto: TOCCI, R.J., WIDMER, N.S., MOSS, G.L. **Sistemas Digitais: Princípios e Aplicações**. 12^a ed. São Paulo: Pearson, 2018.

... leia o artigo VEITCH, Edward Westbrook (1952-05-03) [1952-05-02]. “A Chart Method for Simplifying Truth Functions”. **Transactions of the 1952 ACM Annual Meeting**. ACM Annual Conference/Annual Meeting: Proceedings of the 1952 ACM Annual Meeting (Pittsburgh, Pennsylvania, USA). New York, USA: Association for Computing Machinery (ACM): 127–133, disponível em:
<https://dl.acm.org/doi/pdf/10.1145/609784.609801>

... leia o artigo KARNAUGH, Maurice (November 1953) [1953-04-23, 1953-03-17]. “The Map Method for Synthesis of Combinational Logic Circuits”. **Transactions of the American Institute of Electrical Engineers**, Part I: Communication and Electronics. 72 (5): 593–599, disponível em:
<https://web.archive.org/web/20170416232229/http://philectrosophy.com/documents/The%20Map%20Method%20For%20Synthesis%20of.pdf>

Continua na Parte II