S. R.

MINISTÉRIO DA DEFESA NACIONAL
FORÇA AÉREA PORTUGUESA
CENTRO DE FORMAÇÃO MILITAR E TÉCNICA

Curso de Formação de Praças - RC

COMPÊNDIO

ELECTRÓNICA DIGITAL 1

EPR: SAJ João Marques

CCF 335-16

Julho 2008

S.  R.

MINISTÉRIO DA DEFESA NACIONAL
FORÇA AÉREA PORTUGUESA
CENTRO DE FORMAÇÃO MILITAR E TÉCNICA

CARTA DE PROMULGAÇÃO

JULHO 2008

1. O Compêndio de "Electrónica Digital 1" é uma Publicação "NÃO CLASSIFICADA".

2. Esta publicação entra em vigor logo que recebida.

3. É permitido copiar ou fazer extractos desta publicação sem autorização da entidade promulgadora.

O COMANDANTE

Vítor Manuel Alves Francisco

COR/PILAV

REGISTO DE ALTERAÇÕES

IDENTIFICAÇÃO DA ALTERAÇÃO, Nº DE REGISTO, DATA	DATA DE INTRODUÇÃO	DATA DE ENTRADA EM VIGOR	ASSINATURA, POSTO E UNIDADE DE QUEM INTRODUZIU A ALTERAÇÃO

Cursos:	Curso de Formação de Praças - RC
Nome do Compêndio:	Electrónica Digital 1
Disciplina:	Electrónica Digital e Técnicas Digitais
Data de elaboração:	Abril 2008
Elaborado Por:	SAJ/Meleca João Marques
Verificado Por:	Gabinete da Qualidade da Formação
Comando G. Formação:	TCOR / ENGAER José Saúde
Director de Área:	MAJ / TMMEL Abílio Carmo
Director de Curso:	TEN / TMMEL António Graveto
Formador:	SAJ / Meleca João Marques

ATENÇÃO:

Esta publicação destina-se a apoiar os formandos a frequentarem o Curso de Formação de Praças das especialidades MELECA, MELECT, MELIAV e OPINF na disciplina de Electrónica Digital e Técnicas Digitais.

Não pretendendo ser uma publicação exaustiva do curso em questão, apresenta-se como uma ferramenta de consulta quer durante a duração do curso, quer após a sua conclusão.

ÍNDICE

SISTEMAS DE NUMERAÇÃO **5**

DEFINIÇÕES 5
SISTEMA DECIMAL 5
SISTEMA BINÁRIO 6
SISTEMA OCTAL 7
SISTEMA HEXADECIMAL 8
BINARY CODED DECIMAL BCD 9
CONVERSÃO BINÁRIO DECIMAL 10
CONVERSÃO OCTAL DECIMAL 10
CONVERSÃO DECIMAL BINÁRIO 10
CONVERSÃO DECIMAL OCTAL 12
CONVERSÃO DECIMAL HEXADECIMAL 13
CONVERSÃO BINÁRIO OCTAL 13
CONVERSÃO BINÁRIO HEXADECIMAL 14
ARITMÉTICA BINÁRIA (SOMA) 15
ARITMÉTICA BINÁRIA (SUBTRACÇÃO) 16
ARITMÉTICA BINÁRIA (MULTIPLICAÇÃO) 16
ARITMÉTICA BINÁRIA (DIVISÃO) 17
COMPLEMENTO A 2 18
SUBTRACÇÃO COM O MÉTODO COMPLEMENTO A 2 21
OPERAÇÕES LÓGICAS ELEMENTARES 23

ÁLGEBRA DE BOOLE **25**

EXISTÊNCIA DO ELEMENTO BINÁRIO 25
PORTAS LÓGICAS ELEMENTARES 26
POSTULADOS 30
PROPRIEDADES 31
TEOREMAS 33
FORMAS CANÓNICAS DE REPRESENTAÇÃO 34
REGRAS DA ÁLGEBRA DE BOOLE 36
MAPAS DE KARNAUGH 38

ESCALAS DE INTEGRAÇÃO **43**

FAMÍLIAS LÓGICAS **45**

TTL (TRANSISTOR – TRANSISTOR LOGIC) 45
NMOS (N-TYPE METAL – OXIDE – SEMICONDUCTOR) 47

pMOS (p-type Metal – Oxide – Semiconductor) 48
CMOS (Complementary Metal – Oxide – Semiconductor) 48
ECL (Emitter – Coupled Logic) 49

CIRCUITOS COMBINATÓRIOS 51

Semi – Somadores e Somadores 51
Semi – Subtractores e Subtractores 54
Descodificadores 58
Codificadores 62
Multiplexers 65
De-multiplexers 66
Comparadores 67

BIBLIOGRAFIA 71

LISTA DE PÁGINAS EM VIGOR LPV-1

SISTEMAS DE NUMERAÇÃO

DEFINIÇÕES

Base de um sistema de numeração: Representa o número de símbolos distintos usados para representar qualquer quantidade, dentro desse sistema.

Número: É uma abstracção matemática que é utilizada para fins de quantificação

Valor intrínseco: Valor propriamente dito do próprio dígito ou algarismo por si só.

Valor posicional. É o valor a que cada dígito está associado e que depende da posição que ele ocupa dentro do número.

SISTEMA DECIMAL

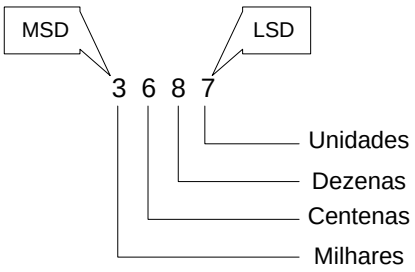
É o sistema que utilizamos no nosso dia a dia; foi introduzido na Europa pelos árabes e baseia-se no facto do homem possuir 10 dedos nas mãos.

Utiliza 10 dígitos; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Após o número 9 voltamos ao 0, assim 9 + 1 = 10, ou seja, é 0 e vai 1

A base do sistema é 10

Exemplo:



MSD – Most Significant Digit

(Digito Mais Significativo)

LSD – Least Significant Digit

(Digito Menos Significativo)

Decomposição do número em potências da sua base:

$3687_{(10)} = 3 \times 10^3 + 6 \times 10^2 + 8 \times 10^1 + 7 \times 10^0 = 3000 + 600 + 80 + 7$

Exemplo de um número fraccionário:

$423{,}45_{(10)} = 4 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$

SISTEMA BINÁRIO

Neste sistema de numeração utilizam-se somente dois símbolos: 0 e 1

Após o número 1 voltamos ao 0, assim 1 + 1 = 10, ou seja, é 0 e vai 1

Normalmente designa-se por sistema de numeração de base 2 ou binário natural.

Cada dígito denomina-se Bit (**Bi**nary Digi**t**)

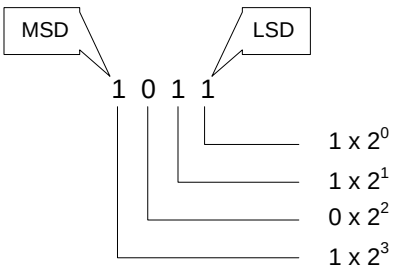
Aqui os pesos são 1, 2, 4, 16, 32, … que correspondem às potências 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, …

Os pesos fraccionários são $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, etc, que correspondem a 2^{-1}, 2^{-2}, 2^{-3}

Exemplos:

$1011_{(2)} = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$101{,}101_{(2)} = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$



SISTEMA OCTAL

É um sistema que utiliza 8 dígitos (Octal): 0, 1, 2, 3, 4, 5, 6, 7

Após o número 7 voltamos ao 0, assim 7 + 1 = 10, ou seja, é 0 e vai 1

Este sistema tornou-se muito prático no tratamento de informação digital, a qual é costume utilizar números de oito elementos binários.

Facilita a representação de números binários com muitos bits.

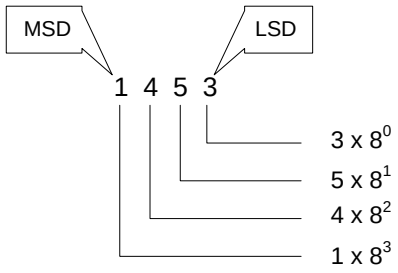
Cada dígito octal equivale a um número binário com 3 dígitos:

Exemplo: $111_{(2)} = 7_{(8)}$

$101_{(2)} = 5_{(8)}$

$100_{(2)} = 4_{(8)}$

Também neste sistema, cada dígito tem um valor numérico e um valor posicional:



Representação de um número octal através do desenvolvimento de potências de base octal:

Exemplo: $123_{(8)} = 1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0$

Os expoentes de um número fraccionário (octal) através das posições dos dígitos.

Representação de um número fraccionário (octal) através do desenvolvimento de potências:

Exemplo: $756{,}205_{(8)} = 7 \times 8^2 + 5 \times 8^1 + 6 \times 8^0 + 2 \times 8^{-1} + 0 \times 8^{-2} + 5 \times 8^{-3}$

Os expoentes à direita da vírgula são negativos e os seus valores indicam quantas casas estão desviados.

SISTEMA HEXADECIMAL

É um sistema que utiliza 16 dígito, ou seja, algarismos de 0 a 9 e letras de A a F:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Após o número F voltamos ao 0, assim F + 1 = 10, ou seja, é 0 e vai 1

Comparando com o sistema octal, este sistema tem mais vantagens, pois torna-se mais fácil a representação de números binários com muitos bits.

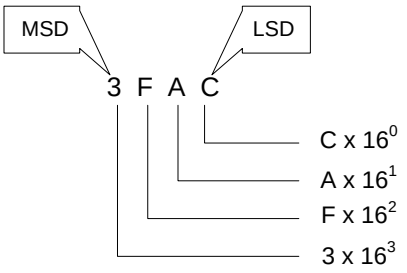
Cada dígito hexadecimal equivale a um número binário com 4 dígitos.

Exemplo: $1111_{(2)} = F_{(16)}$

$1010_{(2)} = A_{(16)}$

$1001_{(2)} = 9_{(16)}$

Também aqui, cada dígito tem um valor numérico e um valor posicional:



Representação de um número através do desenvolvimento de potências de base 16 (Hexadecimal):

Exemplo: $AFA2_{(16)} = A \times 16^3 + F \times 16^2 + A \times 16^1 + 2 \times 16^0$

NOTA: Mais uma vez se verifica que os valores das potências referem-se aos valores das posições dos dígitos:

Os pesos dos dígitos do número $AFA2_{(16)}$, por ordem crescente, são: 16^3, 16^2, 16^1, 16^0

Número fraccionário:

Exemplo: $F16,FAC_{(16)} = F \times 16^2 + 1 \times 16^1 + 6 \times 16^0 + F \times 16^{-1} + A \times 16^{-2} + C \times 16^{-3}$

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

BINARY CODED DECIMAL BCD

No sistema BCD (Decimal codificado em Binário), os dígitos são agrupados em nibbles de quatro bits, cada nibble representando um dígito decimal.

O código BCD tem a desvantagem de usar somente dez das dezasseis combinações de quatro bits.

A representação binária dos números decimais de 10 a 15 é excluída.

O sistema BCD é usado normalmente em contadores de frequência, voltímetros digitais, e calculadoras.

Decimal	Binário
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Exemplo: 5 2 9 Decimal

0101 0010 1001 BCD

CONVERSÃO BINÁRIO DECIMAL

Recorrendo ao método do desenvolvimento em potências de 2, obteremos para o seguinte exemplo:

$11011,101_{(2)} = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

$= 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 + 1 \times 0,5 + 0 \times 0,25 + 1 \times 0,125$

$= 16 + 8 + 2 + 1 + 0,5 + 0,125$

$= 27,625_{(10)}$

NOTA: No caso do sistema binário, o desenvolvimento em potências da base é equivalente à soma dos pesos.

CONVERSÃO OCTAL DECIMAL

Aplica-se também o método do polinómio resultante do desenvolvimento das potências da base:

Exemplo: $734,45_{(8)} = 7 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 + 4 \times 8^{-1} + 5 \times 8^{-2}$

$= 448 + 24 + 4 + 0,5 + 0,078125$

$= 476,578125_{(10)}$

Conversão Hexadecimal Decimal

Utiliza-se o método anterior, mudando apenas a base.

Exemplo: $2AF3,5_{(16)} = 2 \times 16^3 + A \times 16^2 + F \times 16^1 + 3 \times 16^0 + 5 \times 16^{-1}$

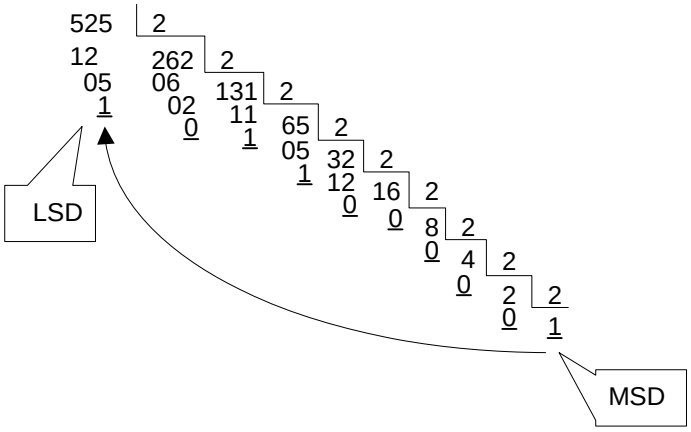
$= 2 \times 4096 + 10 \times 256 + 15 \times 16 + 3 \times 1 + 5 \times 0,0625$

$= 10995,3125_{(10)}$

CONVERSÃO DECIMAL BINÁRIO

Para passar um número inteiro da base decimal para binário, divide-se o número inteiro por 2; o quociente torna a dividir-se por 2, e assim sucessivamente; os restos obtidos e o último quociente constituem o número no sistema binário.

Exemplo: converter o número $525_{(10)}$ para binário.



$525_{(10)} = 1000001101_{(2)}$

MSD – É o bit mais significativo, o quociente da última divisão.

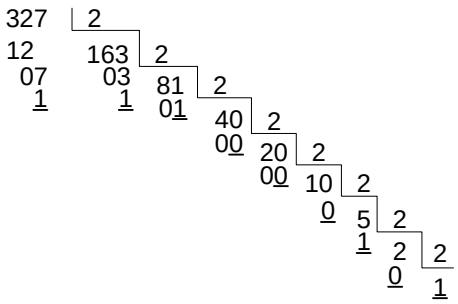
LSD – É o bit menos significativo, o resto da primeira divisão.

Se o número decimal tiver parte fraccionária, a parte inteira converte-se do mesmo modo que vimos anteriormente e a parte fraccionária multiplica-se por 2; a parte inteira deste produto é o algarismo mais significativo da parte fraccionária do número.

Se a parte fraccionária restante for de novo multiplicada por 2, a nova parte inteira será o algarismo mais significativo e assim sucessivamente.

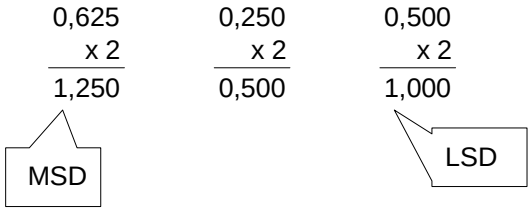
Exemplo: Converter o número $327{,}625_{(10)}$ em binário.

A parte inteira é $327_{(10)}$, que se converte assim:



$327_{(10)} = 101000111_{(2)}$

Para obter a parte fraccionária faremos:



$0,625_{(10)} = 0,101_{(2)}$

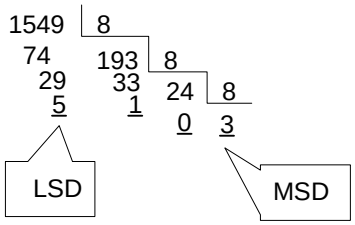
Ou seja: $327,625_{(10)} = 101000111,101_{(2)}$

CONVERSÃO DECIMAL OCTAL

Neste caso, procede-se da mesma forma que para o saco da conversão decimal binário, ressalvando-se o facto de estarmos a trabalhar agora com um sistema de base 8.

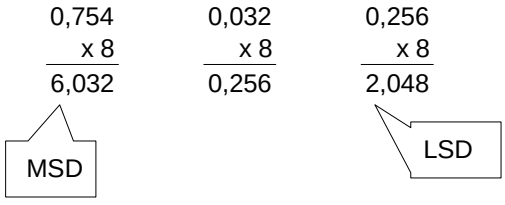
Exemplo: Converter $1549,754_{(10)}$ em octal.

1º Vamos converter a parte inteira:



$1549_{(10)} = 3015_{(8)}$

2º Vamos converter agora a parte fraccionária:



$0,754_{(10)} = 0,602_{(8)}$

Ou seja:

$1549,754_{(10)} = 3015,602_{(8)}$

CONVERSÃO DECIMAL HEXADECIMAL

A parte fraccionária, quando a houver, é obtida através do método das multiplicações sucessivas, à semelhança do que se faz para as conversões de decimal para binário ou octal.

Exemplo: Converter $1549{,}754_{(10)}$ em hexadecimal.

Parte inteira:

```
1549 | 16
 109   96 | 16
  13   00 |  6
  ↙
  D
```

Parte fraccionária:

```
0,754      0,064      0,024
 x 16       x 16       x 16
 4524        384        144
+754       + 64       + 24
12,064     1,024      0,384
↙
C
```

Ou seja:

$1549{,}754_{(10)} = 60D{,}C10_{(16)}$

CONVERSÃO BINÁRIO OCTAL

Esta conversão baseia-se no princípio de que escrever cada dígito octal, são necessários três (3) dígitos binários (bits).

Tabela de correspondências	
Binário	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Adicionando zeros à esquerda da parte inteira e à direita da parte fraccionária (não se alterando portanto o valor do número), transforma-se o nº de bits do número binário num múltiplo de 3.

Depois, agrupando-os 3 a 3 a partir do ponto binário e em ambos os sentidos, utiliza-se a tabela de

correspondências, para obter o equivalente octal.

Exemplos:

a) $1101,11_{(2)}$ = 001 101, 110 $_{(2)}$ = $15,6_{(10)}$

1 5, 6

b) $25,7_{(8)}$ = 010 101, $111_{(2)}$

2 5, 7

CONVERSÃO BINÁRIO HEXADECIMAL

À semelhança da conversão binário/octal, esta conversão também se baseia no princípio de que, para escrever cada dígito hexadecimal, são necessários 4 dígitos binários (bits).

Tabela de correspondências	
Binário	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Adicionando zeros à esquerda da parte inteira e à direita da parte fraccionária, transforma-se o nº de bits do número binário num múltiplo de 4.

Depois, agrupando-os 4 a 4, a partir do ponto binário e em ambos os sentidos, utiliza-se a tabela de

correspondências, para obter o equivalente hexadecimal.

Exemplos:

a) $11101,11_{(2)}$ = 0001 1101, $1100_{(2)} = 1D,C_{(16)}$

1 D, C

b) $78,E_{(16)}$ = 0111 1000, $1110_{(2)}$

7 8, E

ARITMÉTICA BINÁRIA (SOMA)

A aritmética binária é fundamental em todos os computadores digitais e em muitos outros sistemas digitais.

As operações em binário efectuam-se de modo semelhante aos decimais, utilizando-se neste caso as tabuadas em binário.

Tabuada da adição	
X + Y	Soma
0 + 0	0 e vai 0
0 + 1	1 e vai 0
1 + 0	1 e vai 0
1 + 1	0 e vai 1

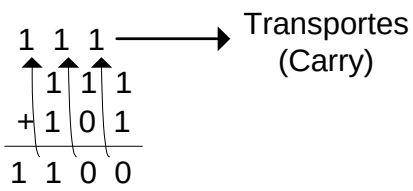
Exemplo:

$7_{(10)} + 5_{(10)} = 12_{(10)}$

$7_{(10)} = 111_{(2)}$

$5_{(10)} = 101_{(2)}$

então:



Confirmando:

$$1100_{(2)} = 1 \times 2^3 + 1 \times 2^2 + 0 + 0 = 8 + 4 = 12_{(10)}$$

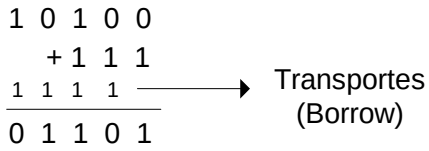
ARITMÉTICA BINÁRIA (SUBTRACÇÃO)

A operação subtracção em binário baseia-se também na tabuada que para ela foi criada:

Tabuada da subtracção	
X – Y	Subtracção
0 - 0	0 e vai 0
0 - 1	1 e vai 1
1 - 0	1 e vai 0
1 - 1	0 e vai 0

Exemplo:

$10100_{(2)} - 111_{(2)} = 01101_{(2)}$



Confirmação:

$10100_{(2)} = 16 + 4 = 20_{(10)}$

$111_{(2)} = 4 + 2 + 1 = 7_{(10)}$

$20_{(10)} - 7_{(10)} = 13_{(10)}$

$1101_{(2)} = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13_{(10)}$

ARITMÉTICA BINÁRIA (MULTIPLICAÇÃO)

Tabuada da multiplicação	
X . Y	Multiplicação
0 x 0	0
0 x 1	0
1 x 0	0
1 x 1	1

Exemplo:

$110_{(2)} \times 101_{(2)} = 11110_{(2)}$

```
    1 1 0
  x 1 0 1
  -------
    1 1 0
  0 0 0
+ 1 1 0
  -------
  1 1 1 1 0
```

Confirmação:

$110_{(2)} = 6_{(10)}$

$101_{(2)} = 5_{(10)}$

$$6_{(10)} \times 5_{(10)} = 30_{(10)}$$

$11110_{(2)} = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 16 + 8 + 4 + 2 = 30_{(10)}$

ARITMÉTICA BINÁRIA (DIVISÃO)

Para a divisão não existe tabuada pois na sua resolução utilizam-se apenas operações de multiplicação e subtracção.

Exemplo:

$11000_{(2)} \div 111_{(2)}$

```
1 1 0 0 0 | 1 1 1
- 1 1 1   | -----
-------   | 1 1
0 1 0 1 0
  - 1 1 1
---------
0 0 0 1 1
```

$Q = 11_{(2)}$

$R = 11_{(2)}$

Confirmação:

$11000_{(2)} = 1 \times 2^4 + 1 \times 2^3 + 0 + 0 + 0 = 16 + 8 = 24_{(10)}$

$111_{(2)} = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 4 + 2 + 1 = 7_{(10)}$

```
24 | 7
3  | 3
```

$Q = 3_{(10)}$

$R = 3_{(10)}$

COMPLEMENTO A 2

A necessidade de representar, em binário, números relativos (+ ou -) e o benefício que seria para alguns dispositivos digitais se conseguíssemos transformar as subtracções em somas, levou à criação da representação em complemento a 2.

O que é o complemento a 2 de um número binário?

O complemento a 2 é uma forma de representar números binários negativos.

Quando escrevemos uma quantidade em algarismos, podemos simplesmente colocar atrás do valor o sinal + ou - ,para indicar respectivamente se o número é positivo ou negativo.

No entanto, num circuito digital, apenas existem 0's e 1's, por isso, tem que haver um modo de representar o sinal.

Estabeleceu-se então, que o dígito de maior peso (MSD) indica o sinal de um número.

Se esse bit for "1", então o número é negativo, se for "0", o número será positivo.

Quando usamos bit de sinal, a capacidade de representação é diferente de quando não usamos bit de sinal.

Exemplo:

Capacidade de representação com 3 bits			
Sem bit de sinal		Com bit de sinal	
Binário	Decimal	Binário	Decimal
000	0	011	+3
001	1	010	+2
010	2	001	+1
011	3	000	0
100	4	111	-1
101	5	110	-2
110	6	101	-3
111	7	100	-4

A capacidade de representação com sinal para n bits determina-se através do seguinte intervalo:

$$\left[-\left(2^{n-1}\right); 2^{n-1}-1\right]$$

Exemplo:

Se o número de bits que estamos a usar é 3, como poderá ser efectuada uma soma sem exceder a capacidade de representação?

$$\left[-\left(2^{3-1}\right); 2^{3-1}-1\right] = [-4;3]$$

Então com 3 bits podemos representar números desde -4 até +3.

A capacidade de representação sem bit de sinal determina-se do seguinte modo:

$$\text{Número máximo} = 2^{n}-1$$

Exemplo:

Se o número de bits que estamos a usar é 3, qual o valor máximo em decimal que podemos representar sem exceder a capacidade de representação?

$$\text{Número máximo} = 2^{3}-1 = 7$$

O que acontece quando excedemos a capacidade de representação?

Exemplos com 4 bits:

a) $+5_{(10)} + 6_{(10)} = 11_{(10)}$ (Com bit sinal)

$+5_{(10)} = 0101_{(2)}$

$+6_{(10)} = 0110_{(2)}$

$$\begin{array}{r} 0101 \\ +\ 0110 \\ \hline 1011_{(2)} = -5_{(10)} \end{array}$$

Como se pode verificar, o resultado do exercício está errado porque com 4 bits a soma não pode dar um valor que esteja fora do intervalo:

$$\left[-\left(2^{4-1}\right); 2^{4-1}-1\right] = [-8;+7]$$

b) $8_{(10)} + 8_{(10)} = 16_{(10)}$ (Sem bit sinal)

$$\begin{array}{r} 1000 \\ +\ 1000 \\ \hline 1\,0000_{(2)} = 0 \end{array}$$

↑ Overflow

Neste caso o resultado está incorrecto porque o resultado ultrapassou a capacidade de representação com 4 bits: $2^{4}-1 = 15$

c) $+7_{(10)} + 7_{(10)} = 14_{(10)}$ (Com bit sinal)

$+7_{(10)} = 0111_{(2)}$

$$\begin{array}{r} 0111 \\ +\ 0111 \\ \hline 1110_{(2)} = -2_{(10)} \end{array}$$

Como se pode verificar, o MSD é 1, logo a resultado é negativo e portanto a soma está errada.

Como se forma o complemento a 2 de um número?

1º - Invertem-se todos os bits do número.

2º - Soam-se "1" ao resultado da inversão.

Exemplo:

a) $+2_{(10)} = 0010_{(2)}$

Para representar o número $-2_{(10)}$, começamos por representar a quantidade indicada como um número positivo:

$$0010_{(2)}$$

Invertemos todos os bits, ficando:

$$1101_{(2)}$$

Somamos 1 ao resultado da inversão:

$$\begin{array}{r} 1101 \\ +1 \\ \hline 1110_{(2)} = -2_{(10)} \end{array}$$

Nota: o MSB indica que o número é negativo.

b) $+7_{(10)} = 0111_{(2)}$

Para representar o número $-7_{(10)}$, começamos por representar a quantidade indicada como um número positivo:

$$0111_{(2)}$$

Invertemos todos os bits, ficando:

$$1000_{(2)}$$

Somamos 1 ao resultado da inversão:

$$\begin{array}{r} 1000 \\ +1 \\ \hline 1001_{(2)} \end{array} = -7_{(10)}$$

Nota: o MSB indica que o número é negativo.

c) $+17_{(10)} = 010001_{(2)}$

Para representar o número $-17_{(10)}$, começamos por representar a quantidade indicada como um número positivo:

$$010001_{(2)}$$

Invertemos todos os bits, ficando:

$$101110_{(2)}$$

Somamos 1 ao resultado da inversão:

$$\begin{array}{r} 101110 \\ +1 \\ \hline 101111_{(2)} \end{array} = -17_{(10)}$$

Nota: o MSB indica que o número é negativo.

SUBTRACÇÃO COM O MÉTODO COMPLEMENTO A 2

$$X - (+Y) = X + (-Y)$$

Método: Calcula-se o complemento a 2 do subtractivo e transforma-se a subtracção numa soma.

Exemplos:

a) $13_{(10)} - 9_{(10)} = 4_{(10)} \rightarrow 13_{(10)} + (-9)_{(10)} = 4_{(10)}$

1º passo: começamos por calcular o complemento a 2 de $+9_{(10)}$

$+9_{(10)}$ = $01001_{(2)}$

Complemento a 1: 10110

Soma-se 1 : + 1

Resultado: $10111_{(2)} = -9_{(10)}$

2º passo: Somam-se as duas parcelas, desprezando os transportes para além do MSD:

Nota: Despreza-se o "1" do transporte porque estamos a subtrair números com 5 dígitos, o resultado nunca poderá aparecer na forma de 6 bits.

O MSD passou a ser o "0", querendo isto dizer que se trata de um número positivo.

$+9_{(10)}$ = $01101_{(2)}$

01101

+10111

$1\ 00100_{(2)} = +4_{(10)}$

MSD

Este bit é desprezado

b) $9_{(10)} - 13_{(10)} = 4_{(10)} \rightarrow 9_{(10)} + (-13)_{(10)} = -4_{(10)}$

1º passo: começamos por calcular o complemento a 2 de $+13_{(10)}$

$+13_{(10)}$ = $01101_{(2)}$

Complemento a 1: 10010

Soma-se 1: + 1

Resultado: $10011_{(2)} = -13_{(10)}$

2º passo: Somam-se as duas parcelas:

01001 ⟶ $9_{(10)}$

+10011

$11100_{(2)} = -4_{(10)}$

Bit sinal

Nota: O bit de sinal é "1", logo o resultado é um número negativo e está representado em complemento a 2.

Como fazer para saber qual o valor desse número?

Calculamos o seu complemento a 2:

$-4_{(10)}$ = $11100_{(2)}$

Complemento a 1: 00011

Soma-se 1 : + 1

Resultado: $00100_{(2)} = +4_{(10)}$

OPERAÇÕES LÓGICAS ELEMENTARES

As operações lógicas elementares têm semelhança com operações aritméticas comuns, inclusive alguns símbolos são idênticos, mas não são necessariamente coincidentes:

1) Operação OR

É similar à adição comum, mas a correspondência não é plena. Símbolo usual é o mesmo da adição.

Exemplo: X = A + B (lê-se X igual a A **ou** B). Um outro símbolo, comum em linguagem de programação, é a barra vertical (X = A | B).

2) Operação AND

É similar à multiplicação comum e há correspondência, como poderá ser visto adiante. Símbolo usual é o mesmo da multiplicação.

Exemplo: X = A . B (lê-se X igual a A **e** B). Muitas vezes, também de forma semelhante à álgebra comum, o sinal de ponto é suprimido: X = AB. O *e comercial* (&) é um símbolo usado em algumas linguagens (X = A & B).

3) Operação NOT

Também denominada negação ou complemento, pode ser considerada similar ao negativo da álgebra comum. Entretanto, não há correspondência plena porque a álgebra de Boole não usa sinal negativo. Símbolo usual é uma barra acima (ou antes) da variável.

Exemplo: $X=\overline{A}$ (lê-se X igual a **não** A). Alguns outros símbolos são o sinal de exclamação (X = !A) e o apóstrofo (X = A').

ÁLGEBRA DE BOOLE

EXISTÊNCIA DO ELEMENTO BINÁRIO

George Boole desenvolveu nas primeiras décadas do século XIX uma álgebra para investigar as leis fundamentais das operações da mente humana ligadas ao raciocínio.

Naquela época não se podia imaginar até que ponto este sistema matemático influenciaria de maneira tão profunda o projecto de circuitos electrónicos e, em consequência, o desenvolvimento de toda a indústria.

A álgebra de Boole, como a álgebra tradicional, tem, em princípio, como objectivo, definir uma série de símbolos para representar objectos ou fenómenos que, encadeados convenientemente, dão lugar a expressões matemáticas mais complexas, denominadas funções. Posteriormente, devem ser definidas as leis que governam tais funções, assim como as relações entre elas, mediante um conjunto de enunciados, postulados, teoremas, etc.

No entanto, existem diferenças entre ambos os sistemas. Enquanto a álgebra tradicional ópera com relações quantitativas, a álgebra de Boole fá-lo com relações lógicas.

No primeiro caso os sinais (+) e (x) representam algoritmos da soma e produto, respectivamente, enquanto na álgebra de Boole representam relações lógicas. Por outro lado, na álgebra convencional utilizam-se símbolos tais como x, y, z, etc., denominados variáveis, para representar quantidades numéricas. Estas variáveis podem tomar muitos valores e, relacionadas através dos algoritmos próprios deste sistema, dão lugar às funções, das quais importa saber a amplitude de certas variáveis quando varia o valor das outras, das quais dependem.

Na álgebra de Boole as variáveis, denominadas binárias, podem tomar somente dois valores distintos: verdadeiro ou falso. Estes dois valores representam-se simbolicamente por 1 e 0, respectivamente. Os símbolos 1 e 0 não exprimem quantidades mas estados das variáveis.

Os componentes electrónicos mais elementares (díodos, transístores, etc.) e, portanto, todos os circuitos digitais, seja qual for a escala de integração, assim como outros elementos eléctricos, tais como motores, lâmpadas, etc., apresentam dois estados estáveis de funcionamento: a lâmpada pode estar acesa ou apagada; o motor, girando ou não.

Estas circunstâncias fazem da álgebra de Boole ou álgebra lógica, assim como do sistema de numeração binário, o suporte matemático ideal para o projecto e análise de circuitos digitais, chamados assim

precisamente por os seus sinais eléctricos de entrada ou saída se ajustarem a este sistema, que utiliza os dígitos 1 e 0 como sinais de representação.

Tudo isto torna possível concretizar um problema numa ou mais expressões que poderão ser manipuladas e simplificadas convenientemente para depois darem origem a um circuito constituído pelo menor número de elementos possível.

O armazenamento de dados numa memória, por exemplo, realiza-se mediante zeros e uns, que fisicamente se traduzem, em relação aos elementos básicos que constituem essa memória, num dos dois estados possíveis de funcionamento. O mesmo acontece com os microprocessadores e outros dispositivos da mesma complexidade. A conversão a valores de significado real realiza-se convenientemente com dispositivos electrónicos, alguns dos quais teremos oportunidade de estudar em capítulos posteriores.

PORTAS LÓGICAS ELEMENTARES

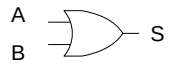
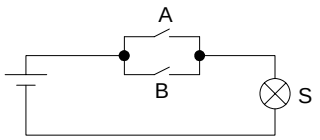
Portas lógicas são dispositivos práticos que executam funções booleanas básicas, isto é, as operações fundamentais OR, AND, NOT e algumas delas derivadas. Na actualidade, a sua implementação é quase sempre em circuitos electrónicos integrados, mas podem ser componentes discretos, circuitos eléctricos com relés, dispositivos ópticos, circuitos hidráulicos ou mesmo mecanismos.

Considerando circuitos eléctricos ou electrónicos, deve-se notar que os valores lógicos 0 e 1 são representados por tensões ou correntes, normalmente em determinadas faixas. Entretanto, na análise lógica, esse dado não é levado em conta e os valores de entradas e saídas são sempre referidos a 0 ou a 1.

Porta OR

Nesta porta, a saída S é igual à operação booleana OR entre os valores das entradas. Na figura abaixo, está representado o símbolo usual e, a tabela de verdade da função.

A função booleana (ou lógica) é S = A + B.

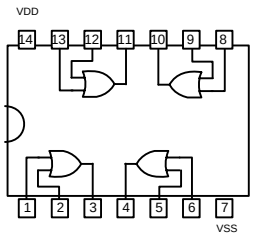
Símbolo	Tabela de Verdade			Circuito equivalente
	A	B	S	
	0	0	0	
	0	1	1	
	1	0	1	
	1	1	1	

A figura mostra um circuito eléctrico simples com dois interruptores e uma lâmpada. Neste caso, o

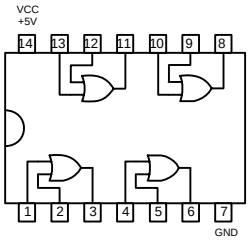
interruptor desligado é nível lógico 0 e interruptor ligado é o nível lógico 1. A lâmpada liga quando pelo menos um dos interruptores está ligado. Portanto, o circuito opera conforme a tabela de verdade ao lado.

As figuras ao lado dão a identificação dos pinos do circuito integrado 4071 (CMOS) e 7432 (TTL). Cada circuito integrado é constituído por 4 portas OR. O pino 7 é ligado à massa (Ground) e o pino 14 é ligado a 5 Volts.

4071

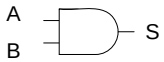
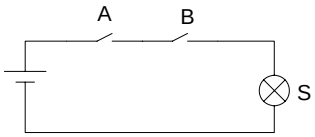


7432



Porta AND

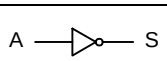
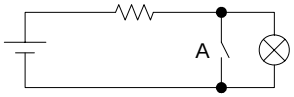
A saída S é igual à operação booleana E entre os valores das entradas. Símbolo usual conforme a figura e a tabela de verdade em baixo. A figura mostra um circuito simples com interruptores. Agora, os contactos estão em série e a saída só terá nível 1 quando todas as entradas forem também 1. A função S = A . B

Símbolo	Tabela de Verdade			Circuito equivalente
	A	B	S	
	0	0	0	
	0	1	0	
	1	0	0	
	1	1	1	

As figuras ao lado dão a identificação dos pinos do circuito integrado 4081 (CMOS) e 7408 (TTL). Cada circuito integrado é constituído por 4 portas AND. O pino 7 é ligado à massa (Ground) e o pino 14 é ligado a 5 Volts.

Porta NOT

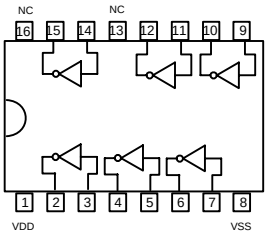
Na porta NOT, a saída S está invertida relativamente à entrada A. Na Figura abaixo podemos ver o símbolo usual, a tabela de verdade e circuito eléctrico simples para a função.

Símbolo	Tabela de verdade		Circuito equivalente
	A	S	
	0	1	
	1	0	

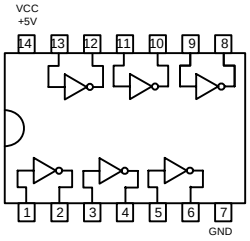
A função lógica é S = $\overline{A}$. A porta NOT é também denominada inversor. Para simplificar os diagramas, o símbolo é apenas um pequeno círculo se estiver junto de uma entrada ou saída de outra porta lógica.

As figuras ao lado dão a identificação dos pinos do circuito integrado 4049 (CMOS) e 7404 (TTL). Cada circuito integrado é constituído por 6 portas NOT.

4049



7404

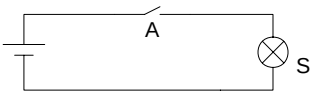


Porta Não inversora

Na porta Não inversora, a saída S é igual à entrada A. Na Figura abaixo podemos ver o símbolo usual, a tabela de verdade e circuito eléctrico simples para a função.

Símbolo	Tabela de verdade	
	A	S
	0	0
	1	1

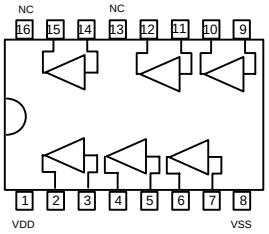
Circuito equivalente



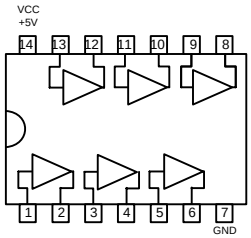
A função lógica é S =A. Esta porta é também denominada como buffer não inversor.

As figuras ao lado dão a identificação dos pinos do circuito integrado 4010 (CMOS) e 7407 (TTL). Cada circuito integrado é constituído por 6 portas Não inversoras.

4010

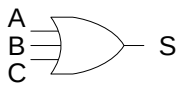
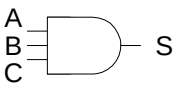


7407



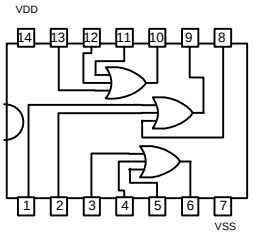
Portas com mais de duas entradas

Em razão da operação que executa, a porta NÃO admite apenas uma entrada. As portas OR e AND (e outras delas derivadas) podem ter qualquer número n ≥ 2 de entradas.

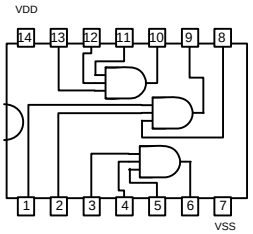
OR de 3 entradas					AND de 3 entradas				
Símbolo	A	B	C	S	Símbolo	A	B	C	S
	0	0	0	0		0	0	0	0
	0	0	1	1		0	0	1	0
	0	1	0	1		0	1	0	0
	0	1	1	1		0	1	1	0
	1	0	0	1		1	0	0	0
	1	0	1	1		1	0	1	0
	1	1	0	1		1	1	0	0
	1	1	1	1		1	1	1	1

Na página anterior, pode-se ver o símbolo e a tabela de verdade para a porta OR de 3 entradas, S = A + B + C, e ao lado a porta AND de três entradas, S = A . B . C.

4075



4073

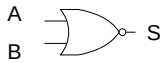
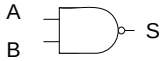


Porta NOR

É uma porta OR com um inversor (NOT) na saída, que, nos diagramas, pode ser representado por um pequeno círculo conforme já comentado.

Expressão lógica segundo álgebra de Boole: S = $\overline{A+B}$

Devido à acção do inversor, os resultados são complementares aos da porta OR.

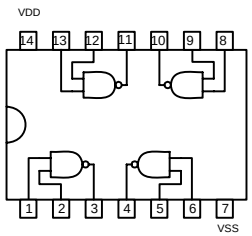
NOR				NAND			
Símbolo	A	B	S	Símbolo	A	B	S
	0	0	1		0	0	1
	0	1	0		0	1	1
	1	0	0		1	0	1
	1	1	0		1	1	0

Porta NAND

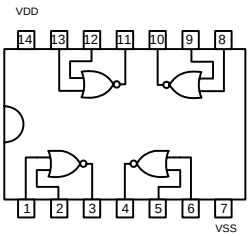
De forma similar à anterior, apresenta resultados complementares aos da porta E devido ao inversor na saída. Símbolo usual e tabela de verdade para duas entradas nas Figuras 01-c e 01-d deste tópico.

Função lógica: S = $\overline{A.B}$

4011



4001

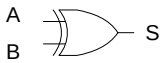
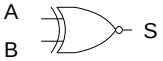


Porta XOR e XNOR

Conforme visto em, a operação booleana OR não oferece plena equivalência com a soma aritmética comum. A função XOR (OR Exclusivo) permite essa correspondência. A porta XOR é a função diferença, a saída é 1 sempre que A≠B.

Expressão booleana: S = A⊕ B.

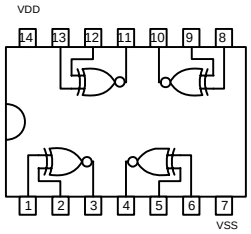
Nota: na realidade, a porta XOR é implementada como uma combinação de portas básicas do tópico anterior. Entretanto, devido à sua importância, ela tem o símbolo próprio aqui exibido e um operador especial para a expressão lógica (⊕).

XOR				XNOR			
Símbolo	A	B	S	Símbolo	A	B	S
	0	0	0		0	0	1
	0	1	1		0	1	0
	1	0	1		1	0	0
	1	1	0		1	1	1

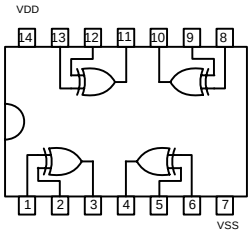
A porta XNOR é a porta XOR com um inversor na saída, resultando em valores complementares aos da anterior. A porta XNOR é a função igualdade, a saída é 1 sempre que A=B.

Expressão lógica: S = $\overline{A \oplus B}$ ou S = A⊙ B

4077



4070



POSTULADOS

Os postulados da álgebra de Boole definem os resultados das operações básicas informadas no tópico anterior.

Postulados da operação OR

X + 0 = X

X + 1 = 1

X + X = X

X + $\overline{X}$ = 1

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

Postulados da operação AND

$X \cdot 0 = 0$

$X \cdot 1 = X$

$X \cdot X = X$

$X \cdot \overline{X} = 0$

A	B	S	Postulados	
0	0	0	$A \cdot 0 = 0$	$A \cdot A = A$
0	1	0	$A \cdot 1 = A$	$A \cdot \overline{A} = 0$
1	0	0	$A \cdot 0 = 0$	$A \cdot \overline{A} = 0$
1	1	1	$A \cdot 1 = A$	$A \cdot A = A$

Postulados da operação XOR

$X \oplus 0 = X$

$X \oplus 1 = \overline{X}$

$X \oplus X = 0$

$X \oplus \overline{X} = 1$

A	B	S	Postulados	
0	0	0	$A \oplus 0 = A$	$A \oplus A = 0$
0	1	1	$A \oplus 1 = \overline{A}$	$A \oplus \overline{A} = 1$
1	0	1	$A \oplus 0 = A$	$A \oplus \overline{A} = 1$
1	1	0	$A \oplus 1 = \overline{A}$	$A \oplus A = 0$

Postulados da operação NOT

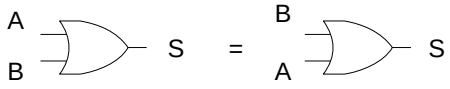
$X = \overline{X}$

$X = \overline{\overline{X}}$

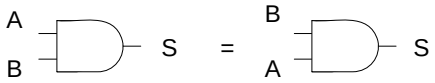
PROPRIEDADES

Propriedade comutativa

$A + B = B + A$

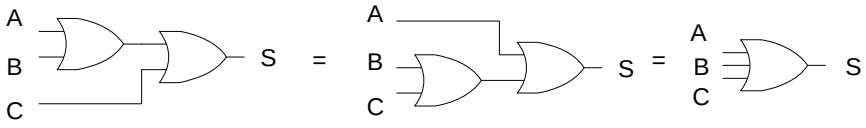


$A \cdot B = B \cdot A$

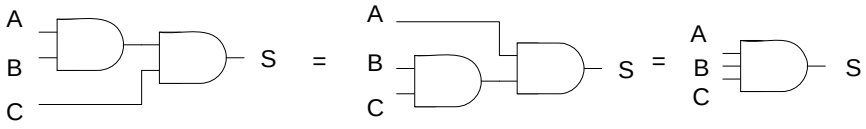


Propriedade associativa

$(A + B) + C = A + (B + C) = A + B + C$

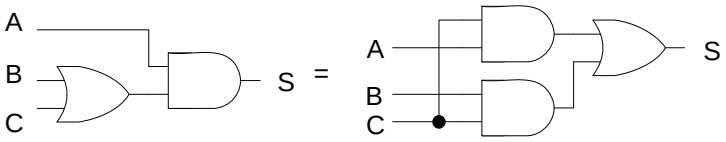


$(A \cdot B) \cdot C = A \cdot (B \cdot C) = A \cdot B \cdot C$

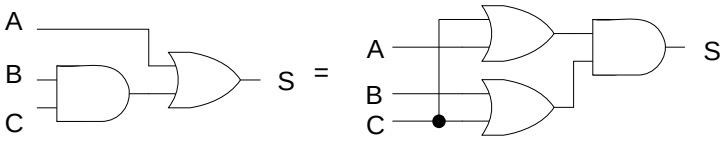


Propriedade distributiva

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



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



TEOREMAS

Teorema 1

a) $A + A \cdot B = A$

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

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

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

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

Teorema 2

a) $A + \overline{A} \cdot B = A + B$

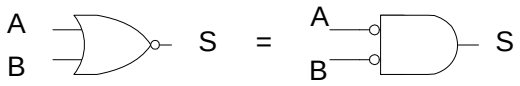
$A + \overline{A} \cdot B = (A + \overline{A}) \cdot (A + B) = 1 \cdot (A + B) = A + B$

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

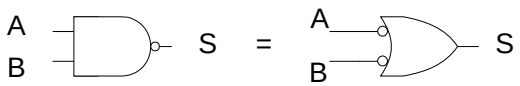
$A \cdot (\overline{A} + B) = A \cdot \overline{A} + A \cdot B = 0 + A \cdot B = A \cdot B$

Teoremas de De Morgan

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



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



FORMAS CANÓNICAS DE REPRESENTAÇÃO

Chama-se forma canónica de uma função booleana a toda a expressão lógica na qual a aparecem todas as variáveis em cada um dos termos que constituem a expressão.

A primeira forma canónica chama-se mintermos e obtém-se a partir de todos as expressões que dão à função o valor 1.

A segunda forma canónica chama-se maxtermos e obtém-se a partir de todas as expressões que dão à função o valor 0.

OR					AND				
A	B	S	Mintermos	Maxtermos	A	B	S	Mintermos	Maxtermos
0	0	0	-	$A + B$	0	0	0	-	$A + B$
0	1	1	$\overline{A} \cdot B$	-	0	1	0	-	$A + \overline{B}$
1	0	1	$A \cdot \overline{B}$	-	1	0	0	-	$\overline{A} + B$
1	1	1	$A \cdot B$	-	1	1	1	$A \cdot B$	-
$S = \overline{A} \cdot B + A \cdot \overline{B} + A \cdot B$ $S = \overline{A} \cdot B + A \cdot (\overline{B} + B)$ $S = \overline{A} \cdot B + A \cdot 1$ $S = \overline{A} \cdot B + A$ (Teorema 2) $S = A + B$					$S = (A + B) \cdot (A + \overline{B}) \cdot (\overline{A} + B)$ $S = A \cdot (B + \overline{B}) \cdot (\overline{A} + B)$ $S = A \cdot 1 \cdot (\overline{A} + B)$ $S = A \cdot (\overline{A} + B)$ (Teorema 2) $S = A \cdot B$				

NOR					NAND				
A	B	S	Mintermos	Maxtermos	A	B	S	Mintermos	Maxtermos
0	0	1	$\overline{A} \cdot \overline{B}$		0	0	1	$\overline{A} \cdot \overline{B}$	
0	1	0		$A + \overline{B}$	0	1	1	$\overline{A} \cdot B$	
1	0	0		$\overline{A} + B$	1	0	1	$A \cdot \overline{B}$	
1	1	0		$\overline{A} + \overline{B}$	1	1	0		$\overline{A} + \overline{B}$
$S = (A + \overline{B}) \cdot (\overline{A} + B) \cdot (\overline{A} + \overline{B})$ $S = (A + \overline{B}) \cdot \overline{A} \cdot (B + \overline{B})$ $S = (A + \overline{B}) \cdot \overline{A} \cdot 1$ $S = (A + \overline{B}) \cdot \overline{A}$ (Teorema 2) $S = \overline{A} \cdot \overline{B} = \overline{A + B}$					$S = \overline{A} \cdot B + A \cdot \overline{B} + \overline{A} \cdot \overline{B}$ $S = \overline{A} \cdot B + \overline{B} \cdot (\overline{A} + A)$ $S = \overline{A} \cdot B + \overline{B} \cdot 1$ $S = \overline{A} \cdot B + \overline{B}$ (Teorema 2) $S = \overline{A} + \overline{B} = \overline{A \cdot B}$				

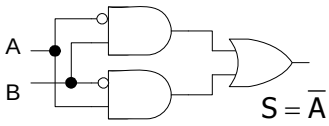
XOR					XNOR				
A	B	S	Mintermos	Maxtermos	A	B	S	Mintermos	Maxtermos
0	0	0		$A + B$	0	0	1	$\overline{A} \cdot \overline{B}$	
0	1	1	$\overline{A} \cdot B$		0	1	0		$A + \overline{B}$
1	0	1	$A \cdot \overline{B}$		1	0	0		$\overline{A} + B$
1	1	0		$\overline{A} + \overline{B}$	1	1	1	$A \cdot B$	

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

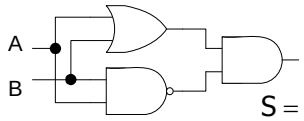
$S = A \cdot \overline{A} + A \cdot \overline{B} + \overline{A} \cdot B + B \cdot \overline{B}$

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

$S = A \cdot \overline{B} + \overline{A} \cdot B = A \oplus B$



$S = \overline{A} . B + A . \overline{B}$



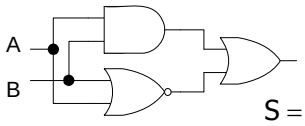
$S = (A + B) . \overline{A . B}$

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

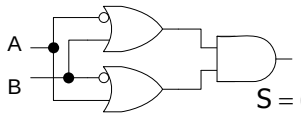
$S = A \cdot \overline{A} + A \cdot B + \overline{A} \cdot \overline{B} + B \cdot \overline{B}$

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

$S = A \cdot B + \overline{A} \cdot \overline{B} = \overline{A \oplus B} = A \odot B$



$S = A . B + \overline{A + B}$



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

Vamos verificar que negando a primeira forma canónica do XOR, vamos obter a segunda forma canónica do XNOR.

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

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

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

Vamos verificar que negando a primeira forma canónica do XNOR, vamos obter a segunda forma canónica do XOR.

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

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

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

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

Para as portas lógica de 3 ou mais entradas é utilizado o mesmo processo.

OR					NAND				
A	B	C	S	Expressão lógica	A	B	C	S	Expressão lógica
0	0	0	0	$S = \overline{A}.\overline{B}.C+\overline{A}.B.\overline{C}+\overline{A}.B.C+A.\overline{B}.\overline{C}+A.\overline{B}.C+A.B.\overline{C}+A.B.C$	0	0	0	0	S = A.B.C
0	0	1	1		0	0	1	0	
0	1	0	1		0	1	0	0	
0	1	1	1		0	1	1	0	
1	0	0	1		1	0	0	0	
1	0	1	1		1	0	1	0	
1	1	0	1		1	1	0	0	
1	1	1	1		1	1	1	1	

REGRAS DA ÁLGEBRA DE BOOLE

Exemplos de simplificação e desenvolvimento de expressões lógicas.

Exemplo 1:

$S = \overline{A}.\overline{B}.C+\overline{A}.B.\overline{C}+\overline{A}.B.C+A.\overline{B}.\overline{C}+A.\overline{B}.C+A.B.\overline{C}+A.B.C$

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

$S = \overline{A}.(B+C)+A.1$

$S = \overline{A}.(B+C)+A$ (Teorema 2)

$S = A+(B+C)$

$S = A+B+C$

Exemplo 2:

$\overline{A} \oplus B = A \odot B$

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

Exemplo 3:

A	B	C	S	Expressão
0	0	0	0	$S = A.\overline{B}.C+A.B.\overline{C}+A.B.C$
0	0	1	0	$S = A.(\overline{B}.C+B.\overline{C}+B.C)$
0	1	0	0	$S = A.(B+C)$
0	1	1	0	
1	0	0	0	
1	0	1	1	
1	1	0	1	
1	1	1	1	

Exemplo 4:

A	B	C	S	Expressão
0	0	0	0	$S = \overline{A}.\overline{B}.C+\overline{A}.B.\overline{C}+A.\overline{B}.\overline{C}+A.B.C$
0	0	1	1	$S = \overline{A}.(\overline{B}.C+B.\overline{C})+A.(\overline{B}.\overline{C}+B.C)$
0	1	0	1	$S = \overline{A}.(B\oplus C)+A.(\overline{B\oplus C})$
0	1	1	0	$Z= B\oplus C$
1	0	0	1	$S = \overline{A}.Z+A.\overline{Z}$
1	0	1	0	$S = A\oplus Z$
1	1	0	0	$S = A\oplus (B\oplus C) = A\oplus B\oplus C$
1	1	1	1	

Exemplo 5:

$XY+YZ+\overline{Z}Y = XY+Y.(Z+\overline{Z})$

$= XY+Y.1$

$= XY+Y$ (Teorema 1)

$= Y$

Exemplo 6:

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

$= 0+A.B+\overline{A}.C+B.C$

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

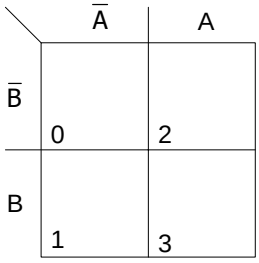
MAPAS DE KARNAUGH

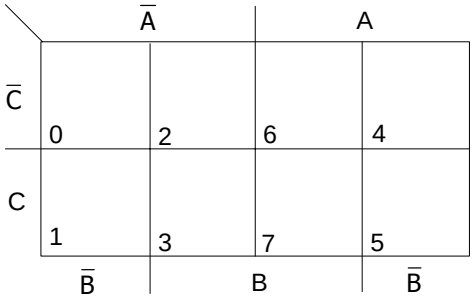
O método gráfico de Karnaugh é um método eficaz e rápido para simplificar funções de até quatro variáveis.

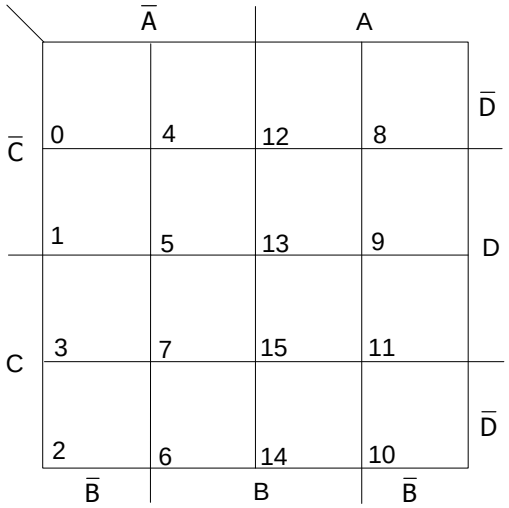
Embora seja válido para cinco ou seis variáveis, é desaconselhável utiliza-lo nestes casos porque apresenta mais dificuldades que vantagens.

Para aplicação deste método é necessário construir um quadrilátero que se divide em 2^n quadrados elementares (células), em que n representa o número de variáveis da função.

Os gráficos adequados para a redução de funções de duas, três e quatro variáveis são:







Cada rectângulo incluído nos mapas pode representar tanto um mintermo como um maxtermo.

Quando tivermos uma função representada na forma de mintermos, vamos pôr "1" nos rectângulos correspondentes aos termos que constituem a expressão da função.

Pelo contrário, se a expressão estiver representada na forma de maxtermos, vamos pôr "0" nos rectângulos correspondentes.

As regras a respeitar são as seguintes:

a) Coloca-se "1" em cada casa de acordo com a função obtida da tabela de verdade.

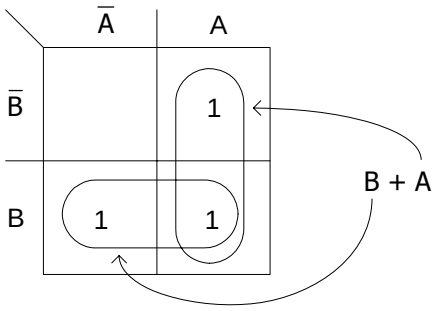
b) A seguir agrupam-se os "1's" em conjuntos de 2, 4, 8 ou 16 segundo os eixos das coordenadas, mas nunca segundo eixos diagonais.

c) Podem se formar grupos de células de lados coincidentes nos bordos do mapa opostos entre si.

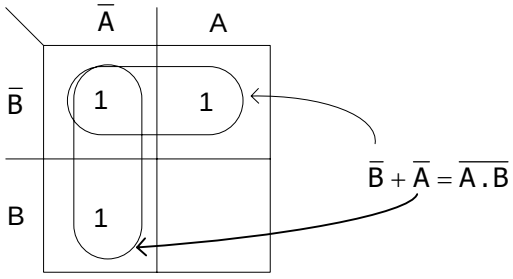
d) Devem-se formar grupos com o maior número de "1's" possíveis.

Exemplos:

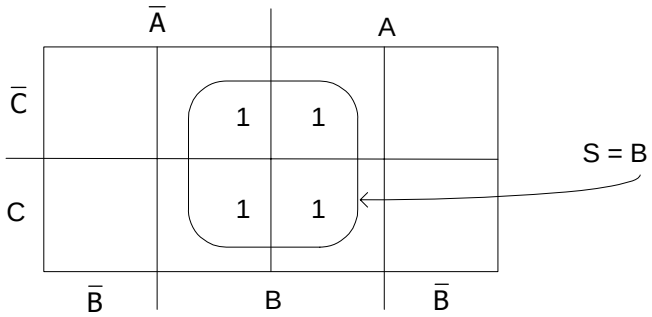
OR			
A	B	C	Mintermos
0	0	0	
0	1	1	$\overline{A}.B$
1	0	1	$A.\overline{B}$
1	1	1	A.B



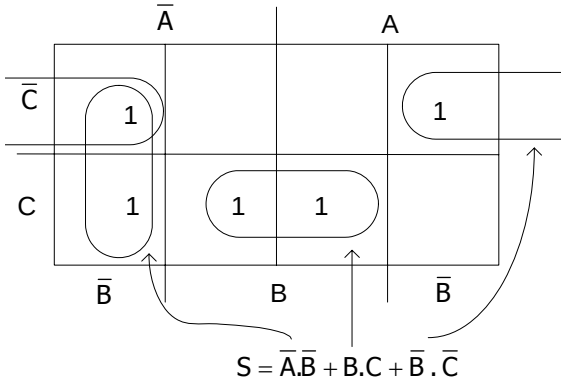
NAND			
A	B	C	Mintermos
0	0	1	$\overline{A}.\overline{B}$
0	1	1	$\overline{A}.B$
1	0	1	$A.\overline{B}$
1	1	0	

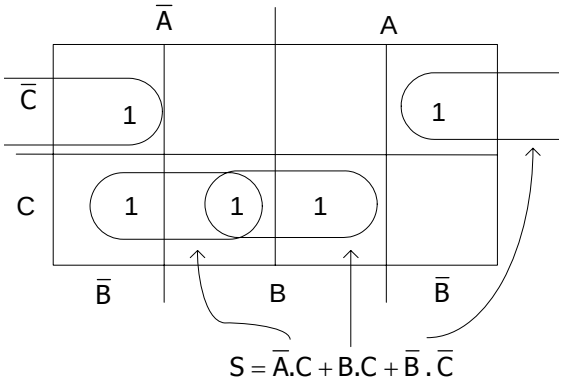


$S = \overline{A}.B.\overline{C} + \overline{A}.B.C + A.B.\overline{C} + A.B.C$



$S = \overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C + \overline{A}.B.C + A.\overline{B}.\overline{C} + A.B.C$

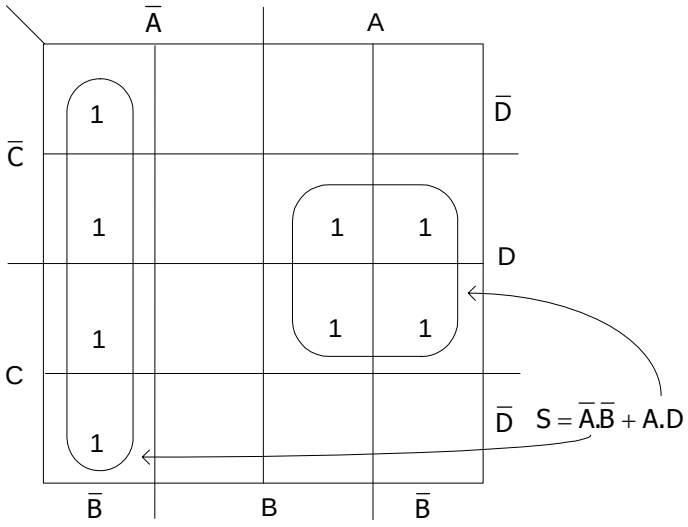


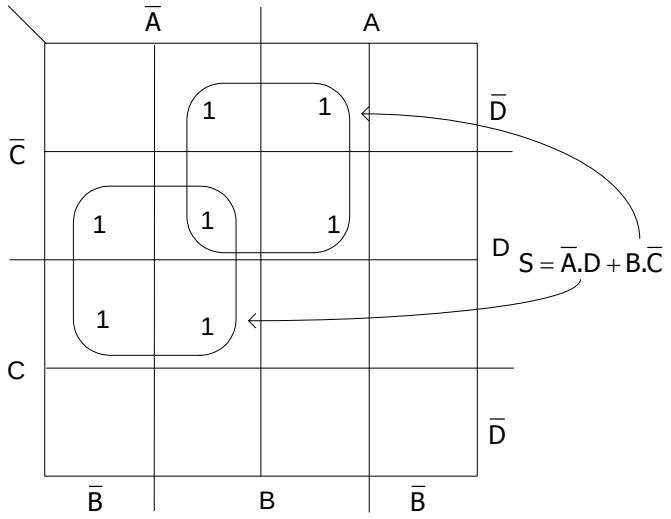


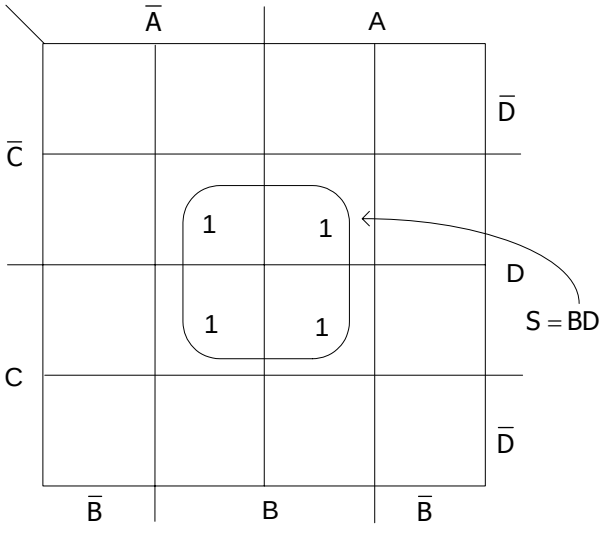
Além deste método de associação poderíamos ainda fazer:

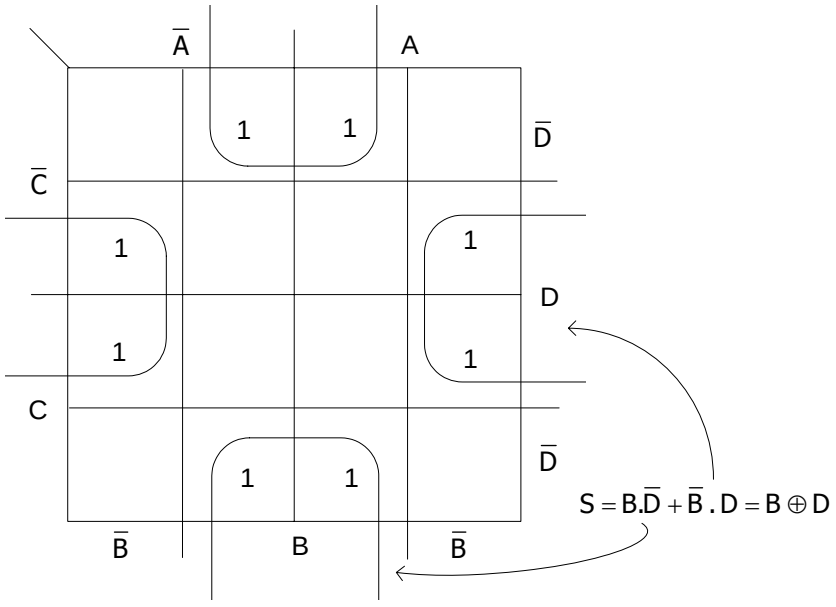
$\overline{A}.C+B.C+\overline{B}.\overline{C} = \overline{A}.\overline{B}+B.C+\overline{B}.\overline{C}$

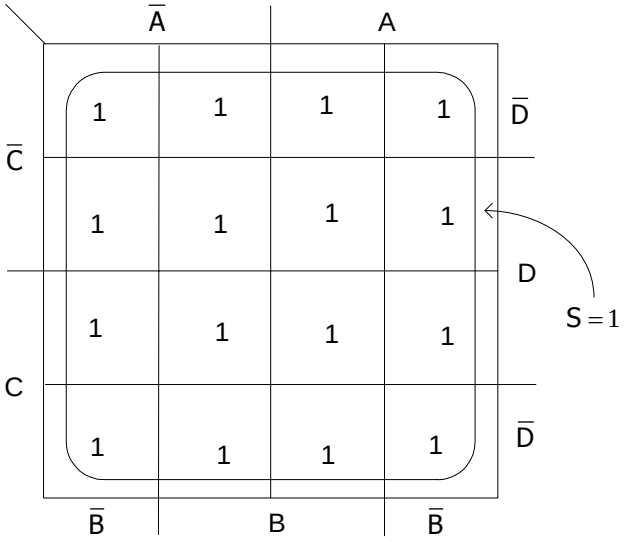
$(\overline{A}+B).C+\overline{B}.\overline{C} = (\overline{A}+\overline{C}).\overline{B}+B.C$




$\overline{A}$
A
$\overline{D}$
$\overline{C}$
D
C
$\overline{D}$
$\overline{B}$
B
$\overline{B}$
1
$S = \overline{A}.D + B.\overline{C}$


$\overline{A}$
A
$\overline{D}$
$\overline{C}$
D
C
$\overline{D}$
$\overline{B}$
B
$\overline{B}$
1
$S = BD$


$\overline{A}$
A
$\overline{D}$
$\overline{C}$
D
C
$\overline{D}$
$\overline{B}$
B
$\overline{B}$
1
$S = B.\overline{D} + \overline{B}.D = B \oplus D$


$\overline{A}$
A
$\overline{C}$
C
$\overline{D}$
D
D
$\overline{D}$
$\overline{B}$
B
$\overline{B}$
1
S = 1

ESCALAS DE INTEGRAÇÃO

A evolução natural da indústria electrónica tem caminhado no sentido da integração, numa só pastilha (chip), de um maior número de componente.

Podemos definir como vantagens da integração:

- Redução do tamanho dos componentes
- Redução do volume e do peso dos equipamentos
- Redução da potência dissipada
- Redução do custo
- Maior fiabilidade

Do ponto de vista da densidade de integração (componentes por mm^2), a classificação dos circuitos integrados disponíveis no mercado é a seguinte:

SSI (*Small Scale Integration*) – Estão neste grupo os CI's de funções lógicas elementares. Podem ter até 100 componentes e realizam cerca de 10 portas lógicas.

MSI (*Medium Scale Integration*) – Podem incluir-se neste grupo os codificadores, multiplexers, contadores, etc. Em cada chip poderão existir entre 100 e 1000 componentes, sendo o número máximo de portas lógicas de 100.

LSI (*Large Scale Integration*) – Incluem-se neste lote os dispositivos próprios da lógica programável: memórias, microprocessadores, calculadoras, etc. Cada circuito terá entre 1000 e 10000 componentes sendo o número máximo de portas lógicas 1000.

VLSI (*Very Large Scale Integration*) – É a tecnologia dos anos 80/90 com mais de 10000 componentes em cada chip.

SLSI (*Super-Large Scale Integration*) - é uma tecnologia de construção que permite ter 50000 ou 100000 componentes num circuito integrado.

FAMÍLIAS LÓGICAS

Existem várias famílias lógicas utilizadas no fabrico dos circuitos integrados entre elas podemos destacar as seguintes: TTL, PMOS/NMOS, CMOS e ECL.

Os principais parâmetros indicados nos catálogos para todas as famílias são:

- Tensão de alimentação e sua tolerância
- Temperatura máxima de funcionamento
- Fan-out – É um número inteiro que nos dá a quantidade de entradas de portas lógicas da mesma família, que se podem ligar à saída de uma porta.
- Níveis lógicos de funcionamento – São as faixas de valores de tensão que o fabricante do circuito integrado garante para cada um dos estados lógico à entrada e à saída.

  Nível 1 = Nível H (alto)

  Nível 0 = Nível L (baixo)

- Imunidade ao ruído – Define-se como a margem de ruído que a porta é capaz de suportar sem que se produzam alterações no seu funcionamento.
- Tempo de propagação – É o tempo que decorre entre o momento em que se introduz a informação na entrada da porta lógica e o instante em que aparece a resposta na saída desta. Esta característica é muito importante e o inverso deste valor define a frequência máxima de funcionamento da porta.
- Dissipação de potência – Habitualmente, indica-se a dissipação por cada porta lógica ou função.

Cada uma das famílias lógicas tem as suas vantagens e os seus inconvenientes em relação às outras, e por isso, se escolherá a mais adequada ao projecto a desenvolver.

No entanto podemos definir como características ideais de uma família lógica as seguintes:

- ✓ Grande densidade de integração.
- ✓ Alta velocidade de comutação.
- ✓ Mínimo consumo.
- ✓ Máxima imunidade ao ruído e às variações de temperatura.
- ✓ Compatibilidade com outras famílias lógicas.
- ✓ Baixo custo

TTL (TRANSISTOR – TRANSISTOR LOGIC)

A lógica TTL foi inventada em 1961 por James L. Buie.

O primeiro circuito integrado comercial foi fabricado em Sylvania em 1963, chamado "Sylvania Universal High-Level Logic family (SUHL)". Os componentes fabricados em Sylvania eram usados no controlo dos mísseis Phoenix. A tecnologia TTL tornou-se popular com os sistemas que a Texas Instruments introduziu através da série 5400 para aplicações militares em 1964, e mais tarde a série 7400, em 1966.

O termo TTL é aplicado a sucessivas gerações da lógica bipolar, com melhoramentos graduais na velocidade e consumo durante cerca de duas décadas.

As suas principais características são:

Tensão de alimentação compreendida entre 4,5V e 5,5V.

- Temperatura de funcionamento entre 0ºC e 70ºC
- Fan-out igual a 10
- Zona de transição entre 1Volt e 1,5Volts
- Tempo de propagação médio de 10ns.
- Dissipação de potência, de 10mW por porta.

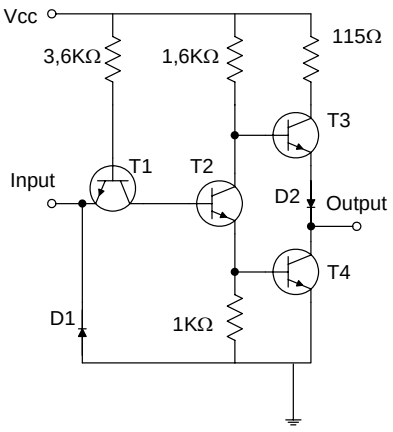
Tomando como referência a série standard e com o objectivo de melhorar, principalmente, os tempos de propagação e a dissipação de potência, desenvolveram-se as seguintes séries:

a) Série 54/74L (Low power), obtendo-se menor consumo (1mW por porta), mas sacrificando o tempo de propagação que é agora de 33ns.

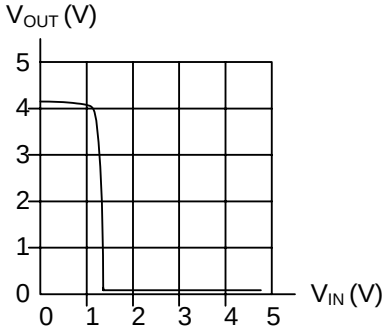
b) Série 54/74S que aparece devido à incorporação de um componente chamado "díodo de Schottky" e que melhora o tempo de propagação (3ns) com uma dissipação de potência de 19mW.

c) Série 54/74LS (Low power, Schottky) com tempos de propagação de 5ns e potência por porta de 2mW.

O esquema do inversor TTL está representado na figura ao lado:



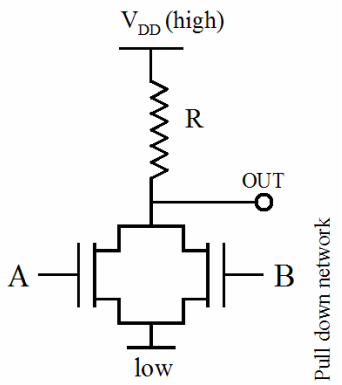
A zona de transição da família TTL é muito próxima de 1,2 Volts, isso deve-se ao facto de que são necessários 0,6V(VBE de T4) + 0,6V(VBE de T2) + 0,6V(VBC de T1) – 0,6(VBE de T1) para os transístores entrarem na zona de condução.



NMOS (N-TYPE METAL – OXIDE – SEMICONDUCTOR)

Esta família lógica utiliza transístores de efeito de campo (MOSFET) para implementar circuitos lógicos. A tecnologia nMOS tem três modos de funcionamento; cut-off; tríodo e saturação.

A tecnologia nMOS é também chamada "pull-down" entre a saída e o negativo da fonte alimentação de acordo com o seguinte circuito:



Este é um exemplo de uma NOR construída com a tecnologia nMOS. Se uma das entradas é 1, o respectivo transístor está à saturação, tornando a saída igual a 0.

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

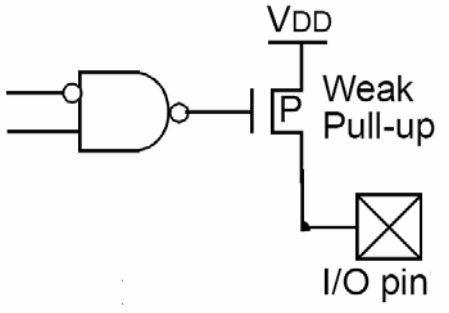
Esta tecnologia é muito fácil de fabricar, mas o problema é que existe corrente quando um dos transístores está a conduzir e isto provoca dissipação de energia.

A tecnologia nMOS também é muita lenta a comutar de 0 para 1. Quando a saída comuta de 1 para 0, o transístor fornece uma resistência baixa que permite uma rápida mudança de nível, mas quando muda de 0 para 1, a resistência R tem um valor muito mais elevado o que torna a transição mais lenta.

pMOS (p-type Metal – Oxide – Semiconductor)

Esta família lógica utiliza transístores de efeito de campo (MOSFET) para implementar circuitos lógicos. A tecnologia pMOS tem três modos de funcionamento; cut-off; tríodo e saturação.

A tecnologia pMOS é também chamada "pull-up" entre a saída e o positivo da fonte alimentação.



As suas características são semelhantes à tecnologia nMOS.

CMOS (Complementary Metal – Oxide – Semiconductor)

Esta tecnologia existe na maior parte dos circuitos integrados. A tecnologia CMOS aparece como série 4000 e é usada em microprocessadores, micro controladores e SRAM. A tecnologia CMOS também é usada numa grande variedade de circuitos analógicos tal como sensores de imagem, conversores de dados e equipamentos de comunicações. Frank Wanlass registou a patente CMOS em 1967 (US Patent 3,356,858).

A tecnologia CMOS é também chamada (complementary - symmetry metal – oxide – semiconductor). As palavras "complementary - symmetry" referem-se ao facto que o circuito é constituído por um par de transístores (MOSFETs) complementares.

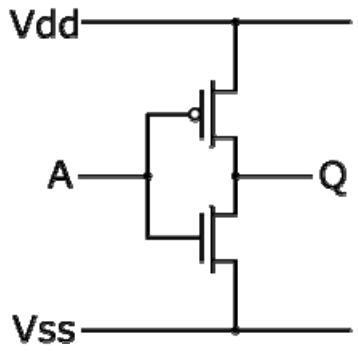
Duas características importantes dos dispositivos CMOS são a maior imunidade ao ruído e o seu baixo consumo quando em repouso. Consequentemente os dispositivos CMOS não produzem tanto calor como as outras famílias lógicas. CMOS também permite uma maior densidade de funções lógicas no mesmo chip.

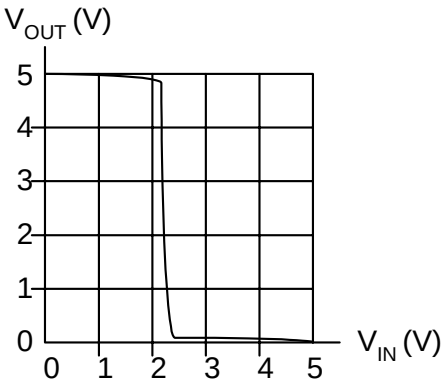
A frase "metal – oxide – semiconductor" é uma referência à estrutura física dos transístores de efeito de campo, têm um eléctrodo metálico (gate) em cima de um óxido isolador, que por sua vez está em cima de um material semicondutor.

As características mais significativas são:

- Tensão de alimentação variável ente e 3 e 15V.
- Gama de temperaturas de funcionamento até aos 85ºC
- Fan-out superior a 50
- Zona de transição perto de 2Volts (para VDD=5V)
- Maior imunidade ao ruído
- Os tempos de propagação variam inversamente com a tensão de alimentação, sendo 125ns para 5V e de 45ns para 15V.
- A potência dissipada por porta é 10nW.

O esquema de um inversor CMOS está representado na figura seguinte:



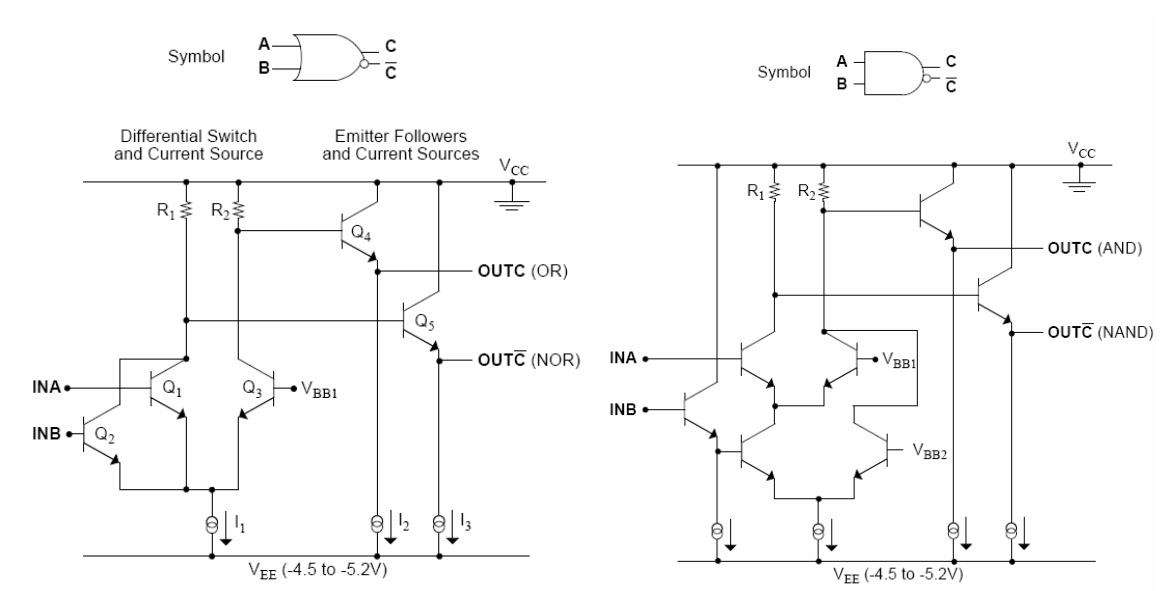


A zona de transição na tecnologia CMOS está muito próximo do meio, bastante distante da zona de ruído, ao contrário da tecnologia TTL.

ECL (EMITTER – COUPLED LOGIC)

ECL foi inventado em Agosto de 1956 na IBM por Hannon S. Yourke, originalmente chamada "*current steering logic*" e foi usada nos computadores IBM7090 e IBM7094.

Nos meados de 1960 até 1990, a lógica ECL consistia numa entrada com amplificador diferencial, seguida de um seguidor de emissor para alimentar as saídas. O interruptor de corrente de Yourke, também conhecido por ECL consistia apenas em amplificadores diferenciais.



CIRCUITOS COMBINATÓRIOS

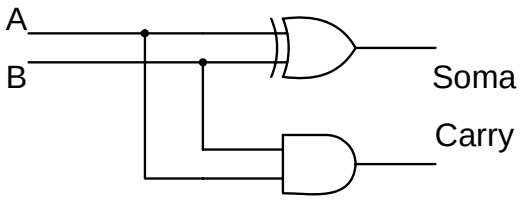
SEMI – SOMADORES E SOMADORES

O semi - somador é um circuito lógico que realiza uma soma entre dois números de um bit. O somador completo soma o carry mais dois números de um bit.

Vamos agora ver como construir um circuito semi - somador a partir da respectiva tabela de verdade.

A	B	Soma	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabuada da soma



$Soma = A\overline{B} + B\overline{A} = A \oplus B$

$Carry = A \cdot B$

O somador completo é constituído por dois semi - somadores. Tem três entradas e duas saídas. Uma das entradas é o carry proveniente da soma anterior. Na figura seguinte apresenta-se a tabela de verdade.

A soma pode ser efectuada de duas formas; (A + B) + Cin ou A + (B + Cin).

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = \overline{A} . \overline{B} . C + \overline{A} . B . \overline{C} + A . \overline{B} . \overline{C} + A . B . C$

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

$S = \overline{A} . (B \oplus C) + A . (\overline{B \oplus C})$

Supondo que $Z = B \oplus C$, então:

$S = \overline{A} . Z + A . \overline{Z} = A \oplus Z$

$S = A \oplus B \oplus C$

Nota: O Cin está a ser representado apenas com a letra C.

O Cout tem duas soluções, uma delas corresponde à soma (A + B) + Cin, a outra corresponde à soma A + (B + Cin).

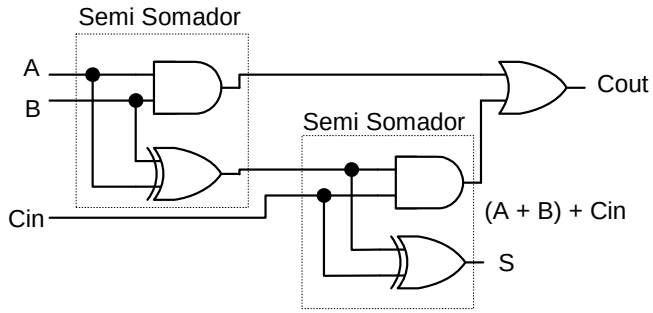
1ª solução, vamos colocar A .B em evidencia:

$Cout = \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$

$Cout = A.B.(\overline{C} + C) + C(\overline{A}.B + A.\overline{B})$

$Cout = A.B.1 + C.(A \oplus B)$

$Cout = A.B + C.(A \oplus B)$



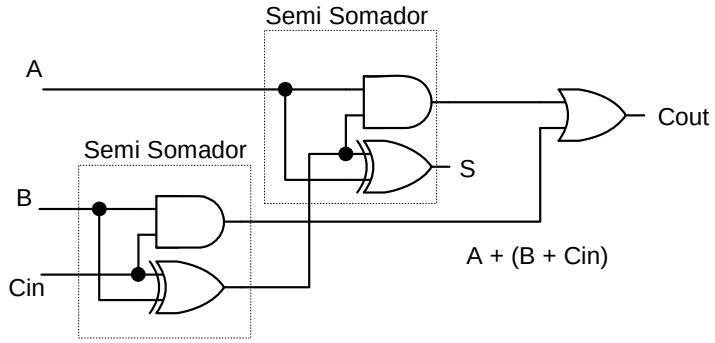
2ª solução, vamos colocar B .C em evidencia:

$Cout = \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$

$Cout = B.C.(\overline{A} + A) + A(\overline{B}.C + B.\overline{C})$

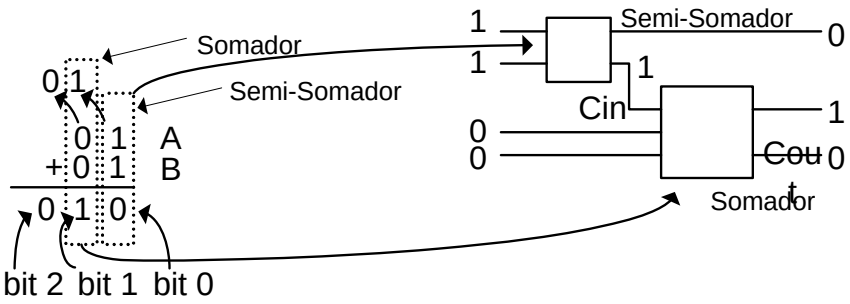
$Cout = B.C.1 + A.(B \oplus C)$

$Cout = B.C + A.(B \oplus C)$

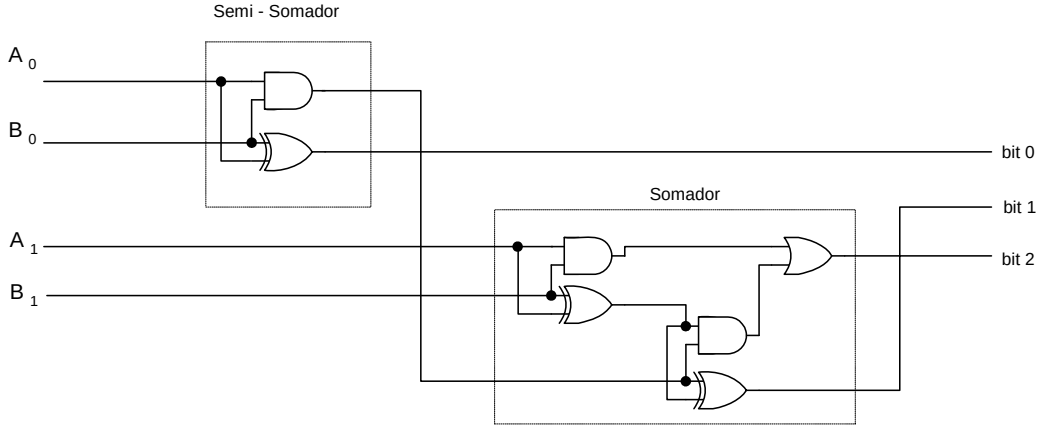


Vamos agora ver o funcionamento do circuito somador de dois números de dois bits.

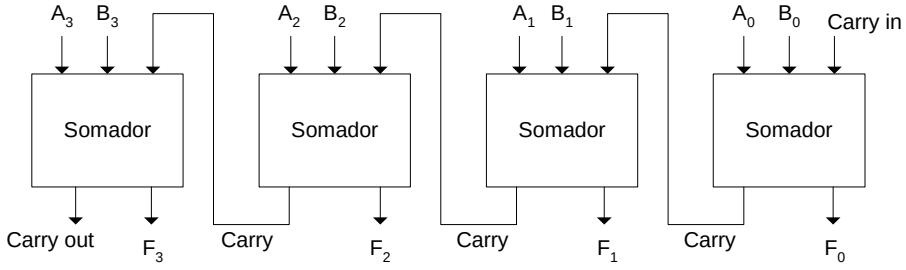
Nesta soma temos um semi - somador e dois somadores. À direita podemos ver como ligar um semi - somador a um somador.



Este circuito pode ser implementado com portas lógicas. O circuito do semi – somador e do somador estão nos seus devidos lugares.



Desta forma, podemos ligar vários somadores, de acordo com o número de bits que queremos somar, conforme a figura seguinte:



Neste circuito, constituído por quatro somadores, podemos somar o número A de 4 bits com o número B de 4 bits mais o Carry in. O resultado aparece nas saídas F_0 a F_3, e também na saída Carry out.

Exemplo com Carry in = 0, A = 1011, e B = 1000

$$\begin{array}{r} 1011 \\ +\ 1000 \\ \hline 10011 \end{array}$$

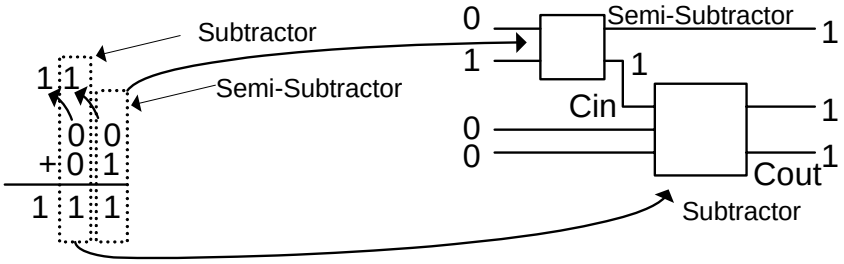
Exemplo com Carry in = 1, A = 1011, e B = 1000

```
    11
   1011
   1000
+     1
  10100
```

SEMI – SUBTRACTORES E SUBTRACTORES

O semi - subtractor é um circuito lógico que realiza uma subtracção entre dois números de um bit. O subtractor completo subtrai dois números de um bit cada menos o borrow (Cin).

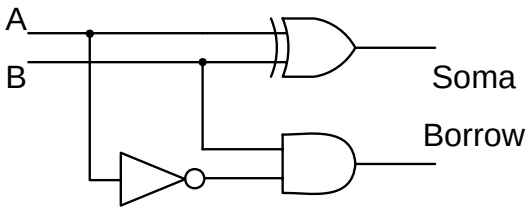
Nesta subtracção temos um semi - subtractor e dois subtractores. À direita podemos ver como ligar um semi - subtractor a um subtractor.



Vamos agora ver como construir um circuito semi - subtractor a partir da respectiva tabela de verdade.

A	B	Soma	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tabuada da subtracção



Soma = $A\overline{B} + B\overline{A} = A \oplus B$

Borrow = $\overline{A}$. B

O subtractor completo é constituído por dois semi - subtractores. Tem três entradas e duas saídas. Uma das entradas é o Cin proveniente da subtracção anterior. Na figura seguinte apresenta-se a tabela de verdade.

A subtracção pode ser efectuada de duas formas; (A - B) + Cin ou A - (B + Cin).

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$S = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.B.C$

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

$S = \overline{A}.(B \oplus C) + A.(\overline{B \oplus C})$

Supondo que $Z = B \oplus C$, então:

$S = \overline{A}.Z + A.\overline{Z} = A \oplus Z$

$S = A \oplus B \oplus C$

Nota: O Cin está a ser representado apenas com a letra C.

Pode verificar que a saída S é igual à da soma.

O Cout tem duas soluções, uma delas corresponde à soma (A - B) - Cin, a outra corresponde à soma A - (B + Cin).

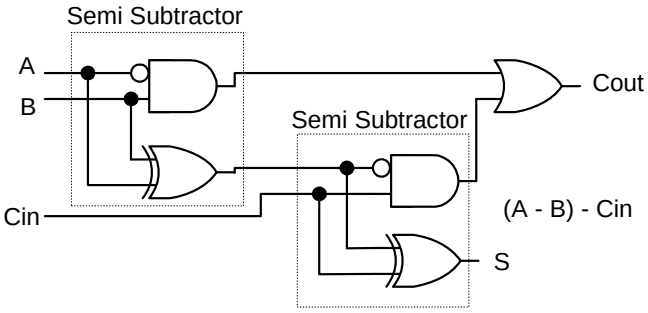
1ª solução, vamos colocar $\overline{A}.B$ em evidencia:

$Cout = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.B.C$

$Cout = \overline{A}.B.(\overline{C} + C) + C(\overline{A}.\overline{B} + A.B)$

$Cout = \overline{A}.B.1 + C.(\overline{A \oplus B})$

$Cout = \overline{A}.B + C.(\overline{A \oplus B})$



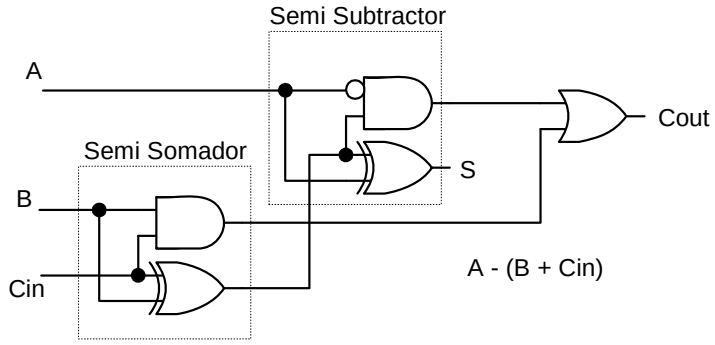
2ª solução, vamos colocar B .C em evidencia:

$Cout = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.B.C$

$Cout = B.C.(\overline{A} + A) + \overline{A}(\overline{B}.C + B.\overline{C})$

$Cout = B.C.1 + \overline{A}.(B \oplus C)$

$Cout = B.C + \overline{A}.(B \oplus C)$



O semi – (somador / subtractor) é constituído de um modo semelhante ao semi – somador e ao semi – subtractor de acordo com a seguinte tabela:

	S	A	B	F	Cout
S	0	0	0	0	0
O	0	0	1	1	0
M	0	1	0	1	0
A	0	1	1	0	1
S	1	0	0	0	0
U	1	0	1	1	1
B	1	1	0	1	0
T	1	1	1	0	0

Quando S = 0, o circuito efectua uma soma, quando S = 1, o circuito efectua uma subtracção.

$F = \overline{S} . \overline{A} . B + \overline{S} . A . \overline{B} + S . \overline{A} . B + S . A . \overline{B}$

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

$F = \overline{S} . (A \oplus B) + S . (A \oplus B)$

$F = (A \oplus B) . (\overline{S} + S)$

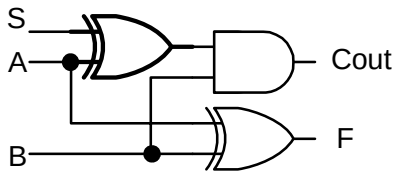
$F = (A \oplus B) . 1$

$F = A \oplus B$

$Cout = \overline{S} . A . B + S . \overline{A} . B$

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

$Cout = B . (A \oplus S)$



A diferença entre a soma e a subtracção é a inversão do A.

Na soma, Cout = A . B

Na subtracção, Cout = $\overline{A} . B$

Por isso este circuito utiliza um XOR para realizar essa função:

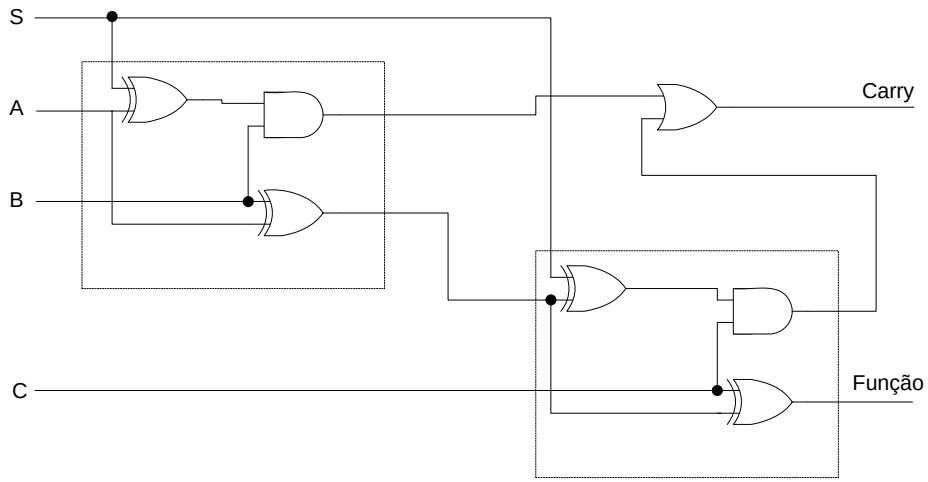
Quando S = 0, temos $A \oplus 0 = A$

Quando S = 1, temos $A \oplus 1 = \overline{A}$

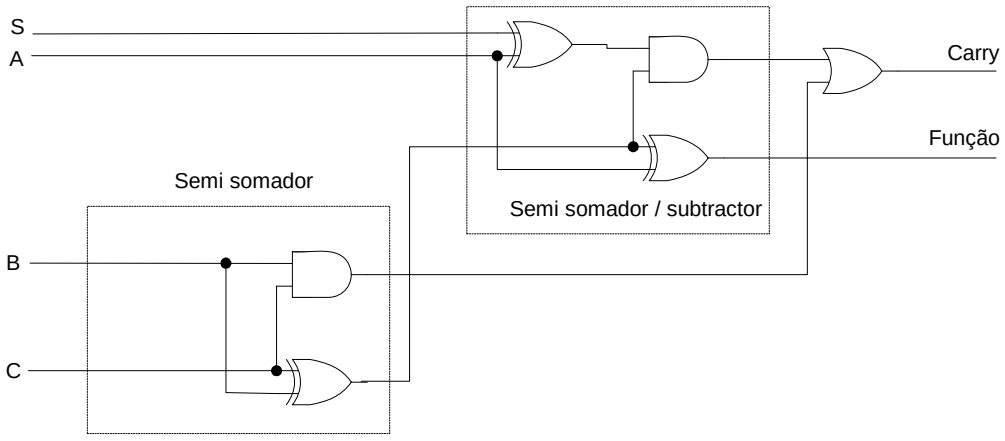
Verifique que este XOR foi colocado no lugar do inversor do semi – subtractor, desta forma podemos utilizar o circuito subtractor e transformar em somador / subtractor.

Podemos utilizar o circuito anterior para construir o somador/subtractor, de acordo com os circuitos seguintes:

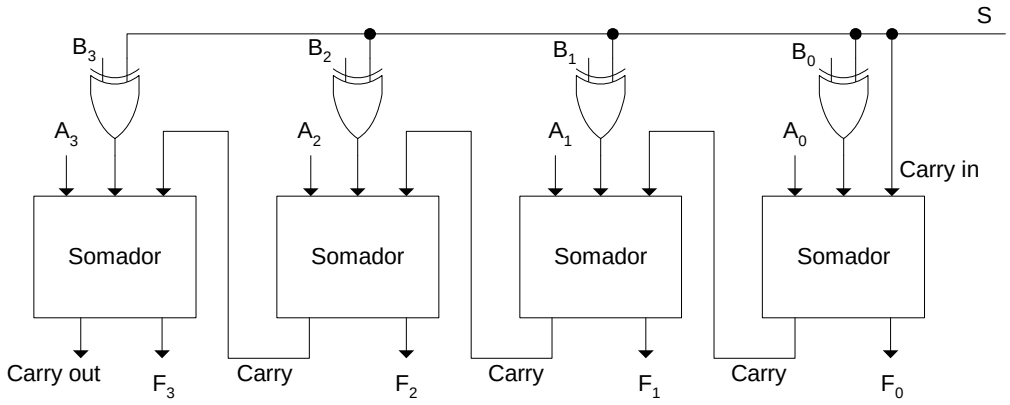
Este circuito pode realizar a operação soma, (A + B) + C quando S = 0 e a operação subtracção, (A – B) – C quando S = 1



Este circuito pode realizar a operação soma, A + (B + C) quando S = 0 e a operação subtracção, A – (B + C) quando S = 1



A subtracção pode também utilizar o método de complemento a 2, que consiste em efectuar uma soma com o complemento a 2 do segundo numero. Para esse efeito inverte-se o numero B com a função XOR e soma-se 1 através da entrada Carry.



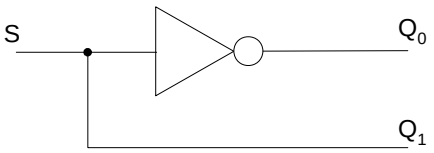
DESCODIFICADORES

Um descodificador selecciona uma das saídas dependendo da combinação binária na entrada. O número de saídas é igual a 2^n, sendo n o número de bits.

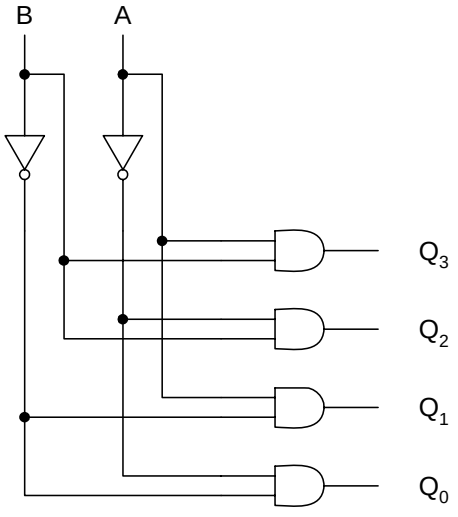
Vamos analisar o descodificador mais simples que tem apenas um bit de selecção e é constituído apenas por um inversor.

O circuito que corresponde a esta tabela é o seguinte:

Descodificador com 1 bit de selecção		
S	Q_0	Q_1
0	1	0
1	0	1



Descodificador com 2 bits de selecção					
Entradas		Saídas			
S_1 B	S_0 A	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



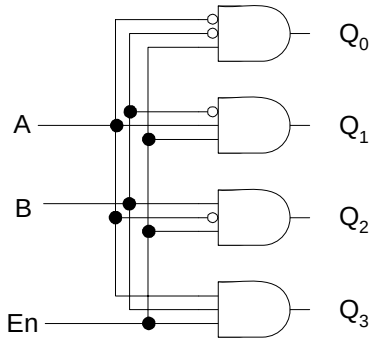
$Q_0 = \overline{A} . \overline{B}$

$Q_1 = A . \overline{B}$

$Q_2 = \overline{A} . B$

$Q_3 = A . B$

Descodificador com enable e 2 bits de selecção						
Entradas			Saídas			
En	B	A	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



$Q_0 = \overline{A} . \overline{B} . En$

$Q_1 = A . \overline{B} . En$

$Q_2 = \overline{A} . B . En$

$Q_3 = A . B . En$

Descodificador 74139

O 74139 é um circuito integrado TTL de 16 pinos com dois descodificadores independentes.

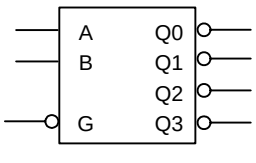
74139						
Entradas			Saídas			
G	B	A	Q_0	Q_1	Q_2	Q_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

$Q_0 = A + B + G$

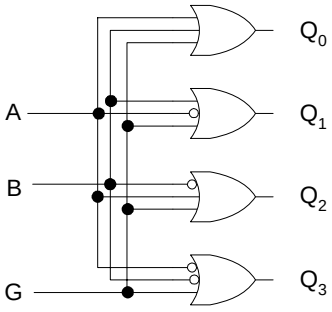
$Q_1 = \overline{A} + B + G$

$Q_2 = A + \overline{B} + G$

$Q_3 = \overline{A} + \overline{B} + G$



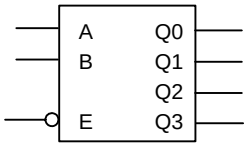
74139



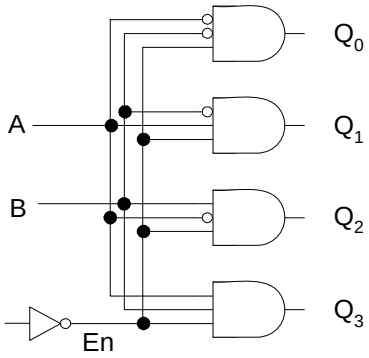
Descodificador 4555

O 4555 é um circuito integrado CMOS de 16 pinos com dois descodificadores independentes.

4555						
Entradas			Saídas			
En	B	A	Q_0	Q_1	Q_2	Q_3
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0



4555



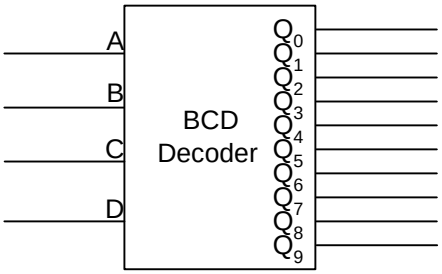
$Q_0 = \overline{A} . \overline{B} . \overline{En}$

$Q_1 = A . \overline{B} . \overline{En}$

$Q_2 = \overline{A} . B . \overline{En}$

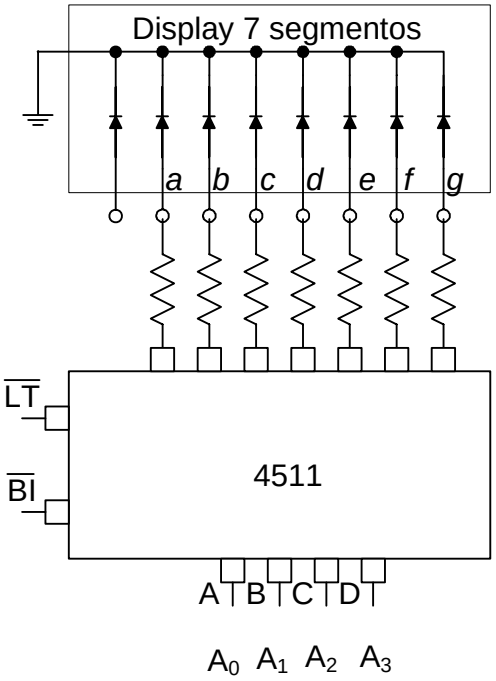
$Q_3 = A . B . \overline{En}$

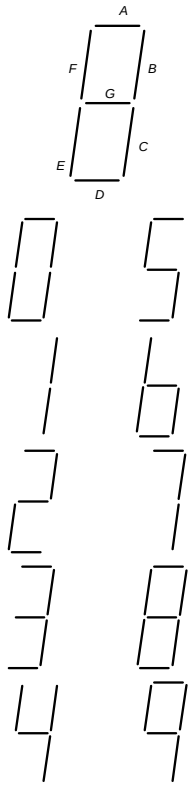
Descodificador BCD



Descodificador BCD													
Entradas				Saídas									
D	C	B	A	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Descodificador BCD – 7SEG





O descodificador 4511 é utilizado com um display de 7 segmentos em cátodo comum. As saídas são activadas a nível lógico 1.

Entradas BCD				**Saídas LED's**						
D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

CODIFICADORES

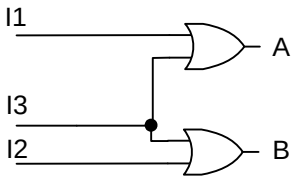
Um codificador é um circuito combinatório formado por 2^n entradas e n saídas cuja função é tal que, quando uma só entrada é activada, as saídas representam o número de ordem da entrada activada.

O codificador sem prioridade é utilizado quando apenas uma das entradas é activada, as saídas assumem o valor que corresponde ao número de ordem dessa entrada

Codificador sem prioridade					
Entradas				Saídas	
I_0	I_1	I_2	I_3	B	A
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

$A = I_1 + I_3$

$B = I_2 + I_3$



A entrada I_0 não é ligada, pois não vai ser necessária para activar nenhuma saída.

O codificador com prioridade é utilizado quando há a possibilidade de haver mais do que uma entrada activada simultaneamente, a saída assume o valor que corresponde à entrada activada de maior valor decimal.

Codificador com prioridade						
Entradas				Saídas		Prioridade
I_3	I_2	I_1	I_0	B	A	
0	0	0	0	0	0	I_0
0	0	0	1	0	0	I_0
0	0	1	0	0	1	I_1
0	0	1	1	0	1	I_1
0	1	0	0	1	0	I_2
0	1	0	1	1	0	I_2
0	1	1	0	1	0	I_2
0	1	1	1	1	0	I_2
1	0	0	0	1	1	I_3
1	0	0	1	1	1	I_3
1	0	1	0	1	1	I_3
1	0	1	1	1	1	I_3
1	1	0	0	1	1	I_3
1	1	0	1	1	1	I_3
1	1	1	0	1	1	I_3
1	1	1	1	1	1	I_3

$B = (I_3 + I_2 + I_1 + I_0) \,.\, (I_3 + I_2 + I_1 + \overline{I_0}) \,.\, (I_3 + I_2 + \overline{I_1} + I_0) \,.\, (I_3 + I_2 + \overline{I_1} + \overline{I_0})$

$B = (I_3 + I_2) + (I_1 + I_0) \,.\, (I_1 + \overline{I_0}) \,.\, (\overline{I_1} + I_0) \,.\, (\overline{I_1} + \overline{I_0})$

$B = (I_3 + I_2) + I_1 \,.\, \overline{I_1}$

$B = I_3 + I_2$

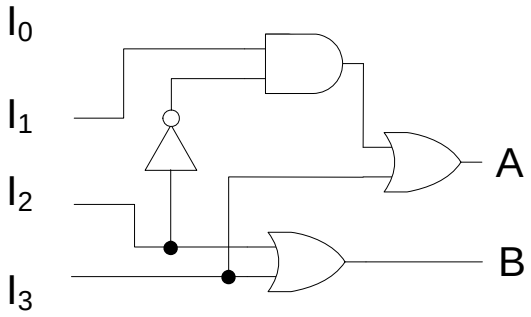
$A = (I_3 + I_2 + I_1 + I_0) \cdot (I_3 + I_2 + I_1 + \overline{I_0}) \cdot (I_3 + \overline{I_2} + I_1 + I_0) \cdot (I_3 + \overline{I_2} + I_1 + \overline{I_0}) \cdot (I_3 + \overline{I_2} + \overline{I_1} + I_0) \cdot (I_3 + \overline{I_2} + \overline{I_1} + \overline{I_0})$

$A = (I_3 + I_2 + I_1) \cdot (I_3 + \overline{I_2} + I_1) \cdot (I_3 + \overline{I_2} + \overline{I_1})$

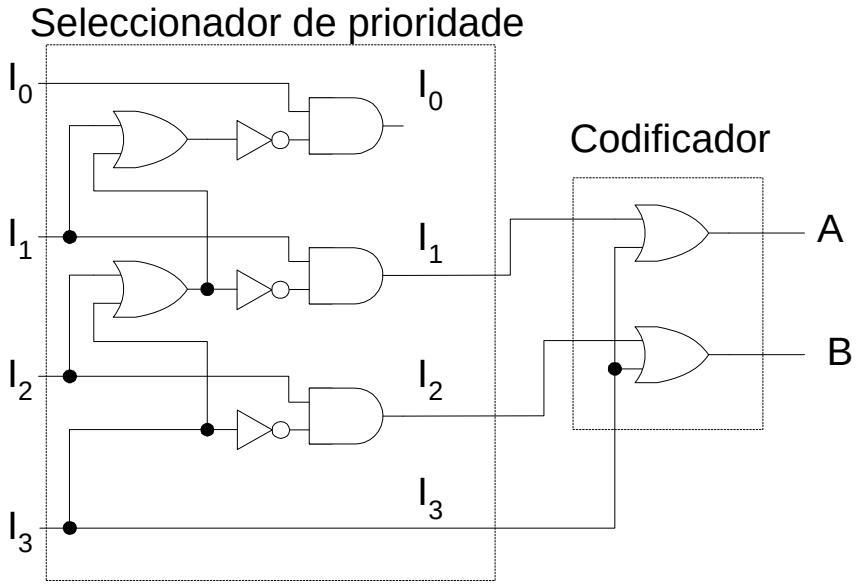
$A = (I_3 + I_1) \cdot (I_3 + \overline{I_2} + \overline{I_1})$

$A = I_3 + I_1 \cdot (\overline{I_2} + \overline{I_1})$

$A = I_3 + I_1 \cdot \overline{I_2}$



Podemos também atribuir uma ordem de prioridade a cada uma das entradas e depois ligar essa prioridade a um codificador sem prioridade.



MULTIPLEXERS

A função do multiplexer consiste em transmitir por uma saída, a informação presente numa das várias entradas. Este circuito é constituído por N linha de entrada, uma saída e n linhas de selecção. A relação entre as entradas de dados e de selecção é: $N = 2^n$

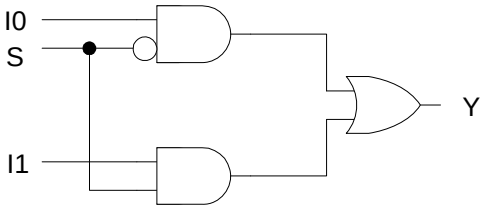
O multiplexer mais simples tem uma entrada de selecção e duas entradas de dados.

Multiplexer				
Selecção	Dados		Saída	
S	I1	I0	Y	
0	0	0	0	I0
0	0	1	1	I0
0	1	0	0	I0
0	1	1	1	I0
1	0	0	0	I1
1	0	1	0	I1
1	1	0	1	I1
1	1	1	1	I1

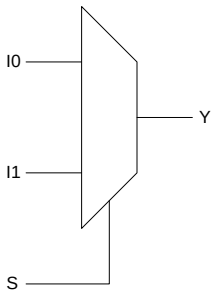
$Y = \overline{S} . \overline{I1} . I0 + \overline{S} . I1 . I0 + S . I1 . \overline{I0} + S . I1 . I0$

$Y = \overline{S} . I0 . (\overline{I1} + I1) + S . I1 . (\overline{I0} + I0)$

$Y = \overline{S} . I0 + S . I1$



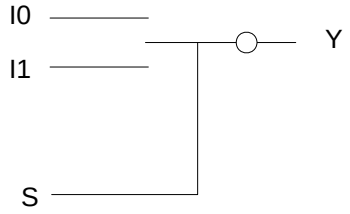
O símbolo deste multiplexer é:



Quando S = 0, a entrada I0 está ligada à saída Y

Quando S = 1, a entrada I1 está ligada à saída Y

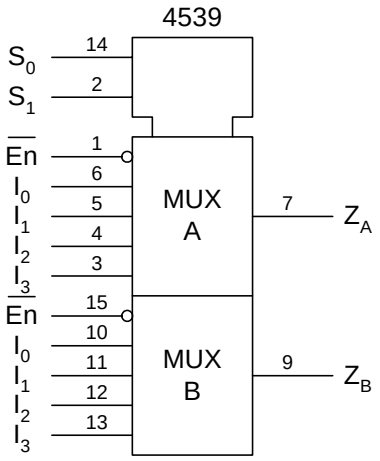
Este circuito é semelhante a um comutador com duas entradas e uma saída, sendo o S o botão de selecção.



Multiplexer de 4 entradas

Selecção		Entradas				Saída
S1	S0	I0	I1	I2	I3	Z
0	0	y	x	x	x	y
0	1	x	y	x	x	y
1	0	x	x	y	x	y
1	1	x	x	x	y	y

4539



O símbolo para o 4539 é mostrado na figura. Há dois multiplexers em cada circuito integrado, são designados por A e B. O multiplexer A tem a sua entrada de enable, $\overline{En}$, e 4 linhas de entrada, I_0, I_1, I_2 e I_3. A saída do multiplexer A é Z_a Um conjunto similar de pinos está disponível no multiplexer B. As linhas de selecção são comuns a ambos os multiplexers.

Sempre que a entrada $\overline{En}$ é 1 o multiplexer é desactivado, e a saída Z permanece a zero. Sempre que $\overline{En}$ é 0, o multiplexer funciona normalmente, e a informação de cada uma das entradas é canalizada para a saída consoante a selecção em S_0 e S_1.

Selecção		$\overline{En}$	Entradas				Saída
S1	S0		I0	I1	I2	I3	Z
0	0	0	y	x	x	x	y
0	0	0	x	y	x	x	y
0	1	0	x	x	y	x	y
0	1	0	x	x	x	y	y
1	0	1	y	x	x	x	0
1	0	1	x	y	x	x	0
1	1	1	x	x	y	x	0
1	1	1	x	x	x	y	0

DE-MULTIPLEXERS

Os de-multiplexers são circuitos com uma só entrada, N saídas e n entradas de selecção, sendo $N = 2^n$. Os dados de entrada são transmitidos à saída seleccionada pelas linhas de selecção.

Selecção		Dados	Saídas			
S1	S0	I	Y0	Y1	Y2	Y3
0	0	**0**	**0**	0	0	0
0	0	**1**	**1**	0	0	0
0	1	**0**	0	**0**	0	0
0	1	**1**	0	**1**	0	0
1	0	**0**	0	0	**0**	0
1	0	**1**	0	0	**1**	0
1	1	**0**	0	0	0	**0**
1	1	**1**	0	0	0	**1**

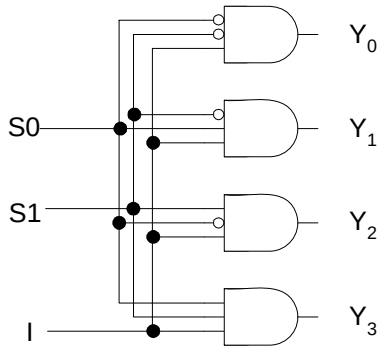
Tecnicamente o de-multiplexer é um descodificador com enable, o mesmo circuito pode ter duas aplicações diferentes.

$Q_0 = \overline{S1} . \overline{S0} . I$

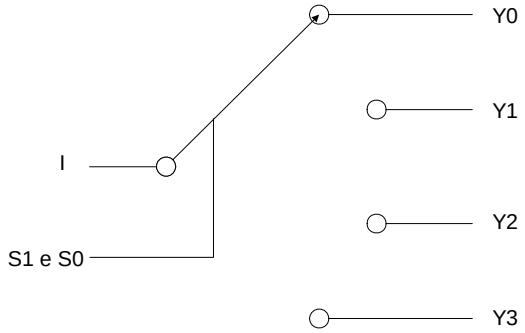
$Q_1 = \overline{S1} . S0 . I$

$Q_2 = S1 . \overline{S0} . I$

$Q_3 = S1 . S0 . I$



Este circuito é semelhante a um comutador com 4 saídas, sendo a selecção feita pelos 2 bits S1 e S0.

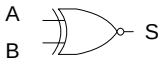


COMPARADORES

Os comparadores são circuitos combinatórios que, ao colocarmos nas suas entradas duas palavras de *n* bits, detectam se são, ou não iguais, e neste caso qual das entradas é maior ou menor.

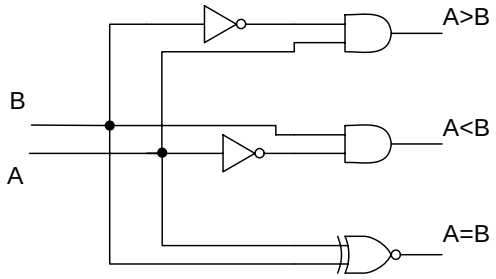
A porta XNOR é uma célula elementar comparadora.

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



O método de projectar um comparador é o mesmo utilizado nos casos anteriores. Na figura seguinte mostra-se a tabela de verdade de um comparador de duas palavras de um bit cada. O processo seguido para o seu projecto é válido para outros comparadores com um maior número de bits por palavra.

A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0



$A>B \rightarrow A.\overline{B}$

$A=B \rightarrow \overline{A \oplus B}$

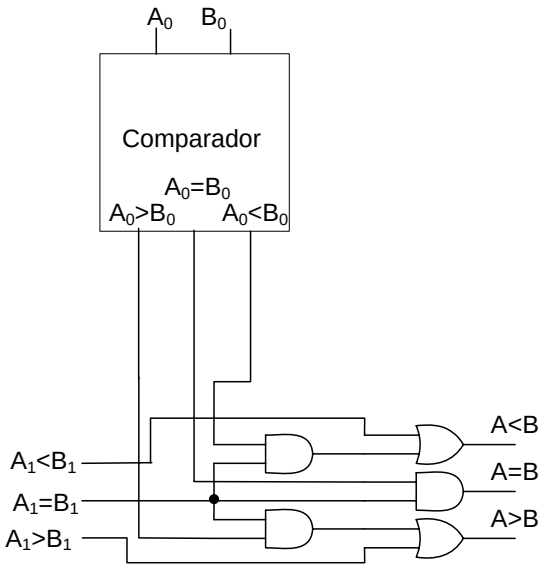
$A<B \rightarrow \overline{A}.B$

É possível ligar dois ou mais comparadores em simultâneo de modo que seja possível comparar números de dois ou mais bits.

Considerando o Nº 01 para "A" e o Nº 10 para "B", podemos dizer imediatamente que B>A. Repare que o bit menos significativo de B é inferior ao bit menos significativo de A, isso significa que o bit mais significativo impõe a condição de ser superior ou inferior. No caso de os bits mais significativos serem iguais, então os bits menos significativos indicarão condição de igualdade.

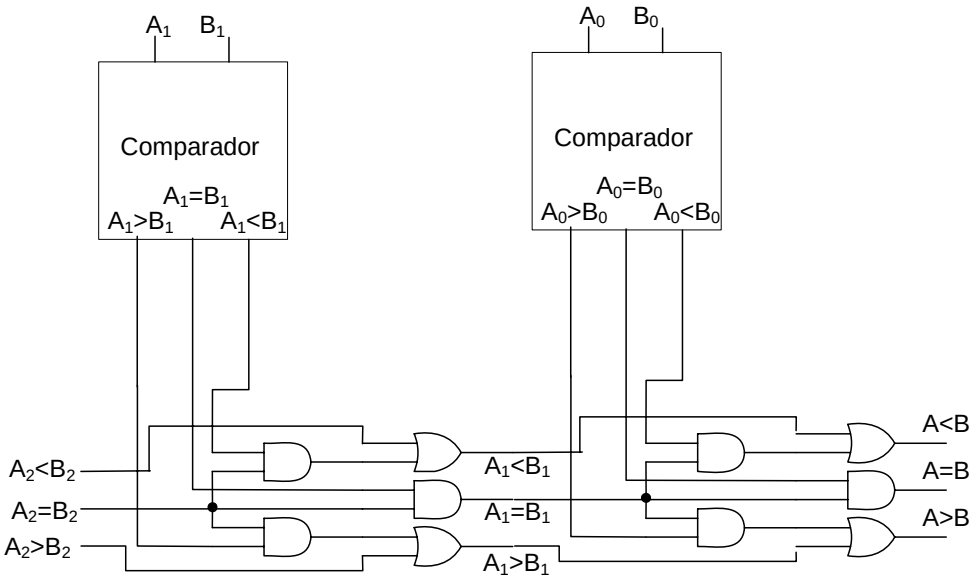
Faça uma análise da tabela de verdade e repare nas condições de igualdade e diferença de cada um dos bits.

B1	B0	A1	A0	A B	A0 B0	A1 B1
0	0	0	0	A=B	A0=B0	A1=B1
0	0	0	1	A>B	A0>B0	A1=B1
0	0	1	0	A>B	A0=B0	A1>B1
0	0	1	1	A>B	A0>B0	A1>B1
0	1	0	0	A<B	A0<B0	A1=B1
0	1	0	1	A=B	A0=B0	A1=B1
0	1	1	0	A>B	A0<B0	A1>B1
0	1	1	1	A>B	A0=B0	A1>B1
1	0	0	0	A<B	A0=B0	A1<B1
1	0	0	1	A<B	A0>B0	A1<B1
1	0	1	0	A=B	A0=B0	A1=B1
1	0	1	1	A>B	A0>B0	A1=B1
1	1	0	0	A<B	A0<B0	A1<B1
1	1	0	1	A<B	A0=B0	A1<B1
1	1	1	0	A<B	A0<B0	A1=B1
1	1	1	1	A=B	A0=B0	A1=B1



Saídas	Condição
A<B	$A_1<B_1$ ou $A_1=B_1$ e $A_0<B_0$
A=B	$A_1=B_1$ e $A_0=B_0$
A>B	$A_1>B_1$ ou $A_1=B_1$ e $A_0>B_0$

A partir deste circuito é possível ligar quantos comparadores forem necessários de acordo com a figura seguinte:



BIBLIOGRAFIA

Padilla, António J. G .-Sistemas Digitais.

Hall, Douglas V. Hall-Microprocessors and Interfacing Programming and Hardware

Matemática II

Apontamentos Pessoais

LISTA DE PÁGINAS EM VIGOR

PÁGINAS	EM VIGOR
CAPA (Verso em branco)	ORIGINAL
CARTA DE PROMULGAÇÃO (Verso em branco)	ORIGINAL
REGISTO DE ALTERAÇÕES (Verso em branco)	ORIGINAL
1 (Verso em branco)	ORIGINAL
3 a 22	ORIGINAL
23 (Verso em branco)	ORIGINAL
25 a 42	ORIGINAL
43 (Verso em branco)	ORIGINAL
45 a 48	ORIGINAL
49 (Verso em branco)	ORIGINAL
51 a 68	ORIGINAL
69 (Verso em branco)	ORIGINAL
71 (Verso em branco)	ORIGINAL
LPV-1 (Verso em branco)	ORIGINAL