CIRCUITOS COMBINATÓRIOS TÍPICOS

CIRCUITOS COMBINATÓRIOS TÍPICOS - 2

- **SUMÁRIO**:
  - CODIFICADORES
  - DESCODIFICADORES
  - MULTIPLEXERS
  - DEMULTIPLEXERS
  - SOMADORES / SUBTRACTORES
  - COMPARADORES

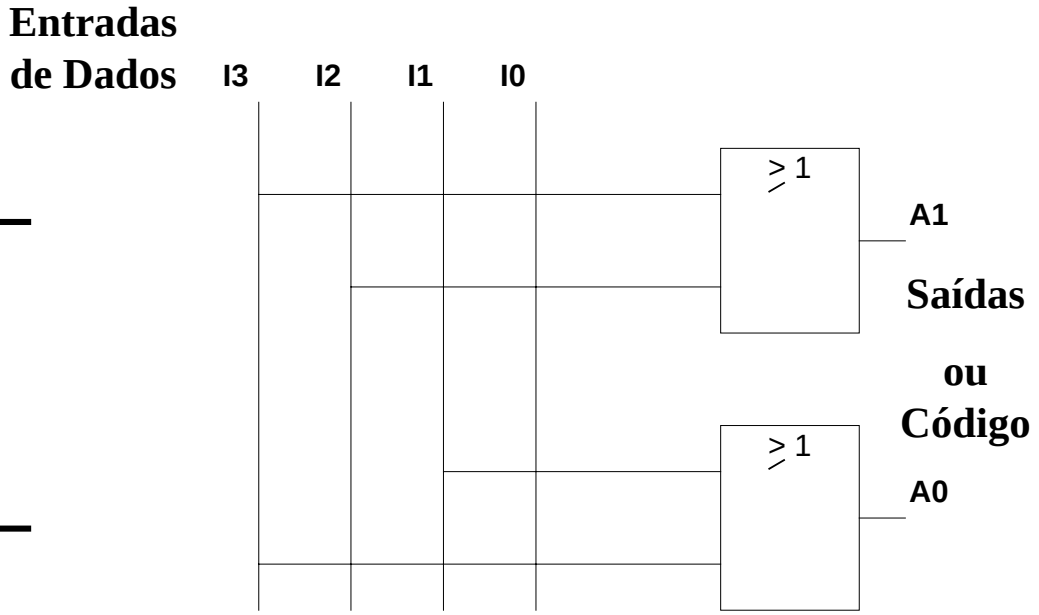
CODIFICADORES:

O CODIFICADOR é um circuito combinatório que permite, perante a activação de uma entrada, gerar uma codificação ou combinação de sinais de saída correspondente a essa entrada.

Exemplo:

I0	I1	I2	I3	A1	A0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	1	1	1	1

CODIFICADOR 4:2

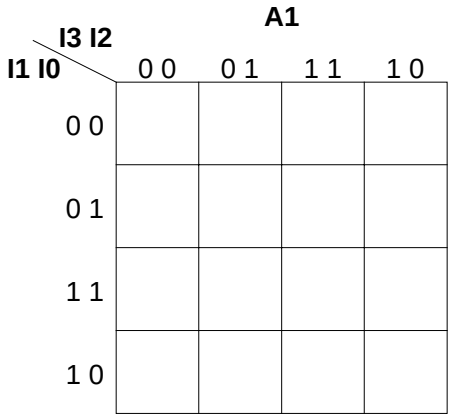


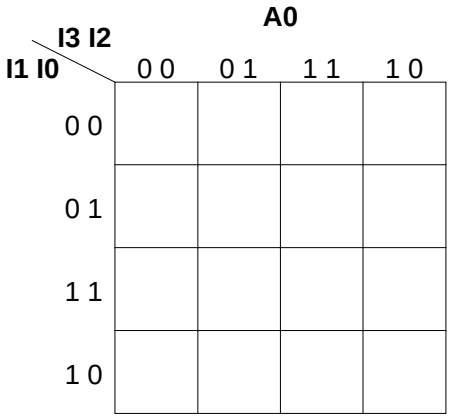
CODIFICADORES DE PRIORIDADE:

Codificadores de Prioridade: Permite a activação de mais de uma entrada simultaneamente gerando a codificação correspondente à entrada mais prioritária activa.

Exercício: Determine uma implementação possível para um conversor de prioridade 4:2, que permita a activação de entradas em simultâneo gerando sempre a codificação correspondente à entrada de maior ordem.

(Sugestão: Comece por determinar os mapas de Karnaugh para cada saída)





DESCODIFICADORES: ARQUITECTURA

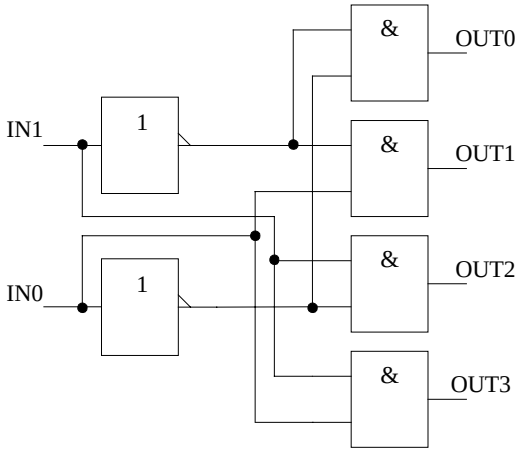
O DESCODIFICADOR é um circuito combinatório que permite, perante uma combinação de entradas, activar uma e só uma saída.

DESCODIFICADOR 2:4

IN1	IN0	OUT0	OUT1	OUT2	OUT3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Entrada de Dados ou Código

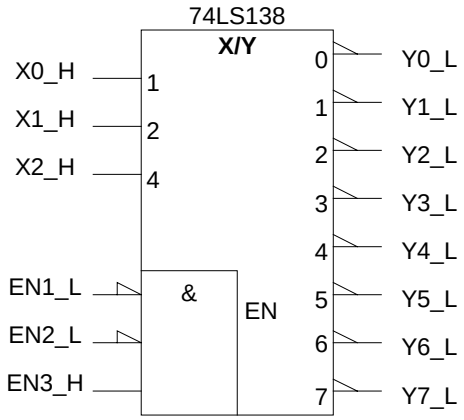
Saídas



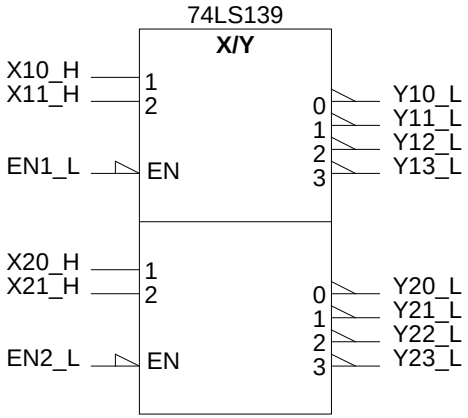
DESCODIFICADORES: SIMBOLOGIA

Exemplos:

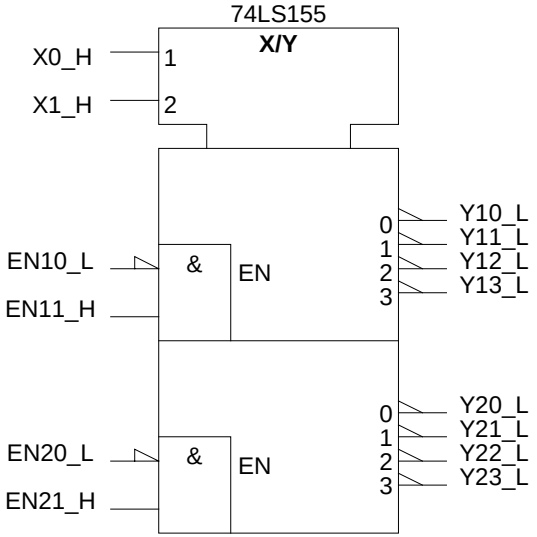
DESCODIFICADOR 3:8



DUAL DECODER 2:4



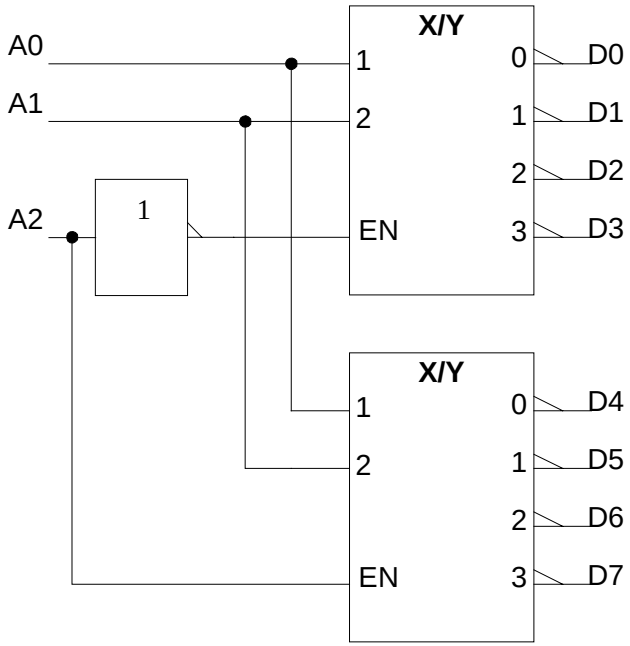
DUAL DECODER 2:4



DESCODIFICADORES: APLICAÇÕES (DESCODIFICAÇÃO)

Implementação de um DECODER 3:8 tendo por base 2 DECODERs 2:4

Exemplo:



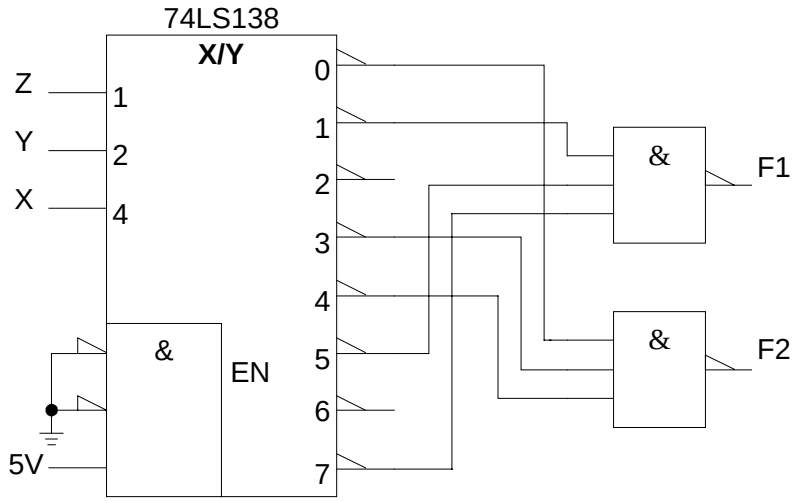
DESCODIFICADORES: APLICAÇÕES (F. COMBINATÓRIAS)

Implementação de F. Combinatórias de 3 variáveis com DECODERs 3:8

•Exemplo:

$f1(X,Y,Z) = \Sigma\, m(1,5,7)$

$f2(X,Y,Z) = \Sigma\, m(0,3,4)$



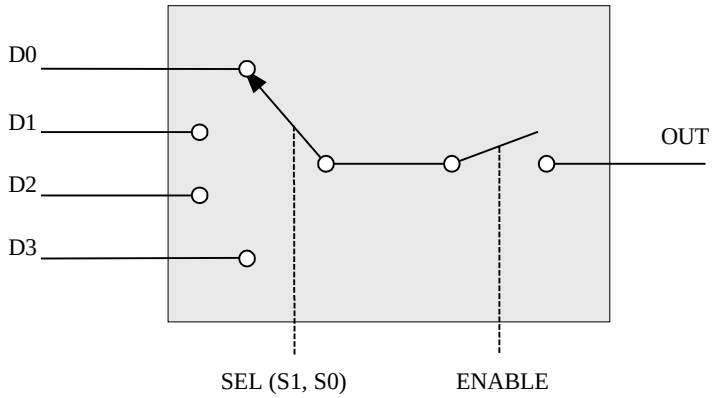
MULTIPLEXERS: ARQUITECTURA

O MULTIPLEXER é um circuito combinatório que permite, através da especificação dos sinais de selecção, encaminhar uma das n entradas de dados para a saída.

S1	S0	OUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3

MULTIPLEXER 4:1

Entradas de Dados



Saída

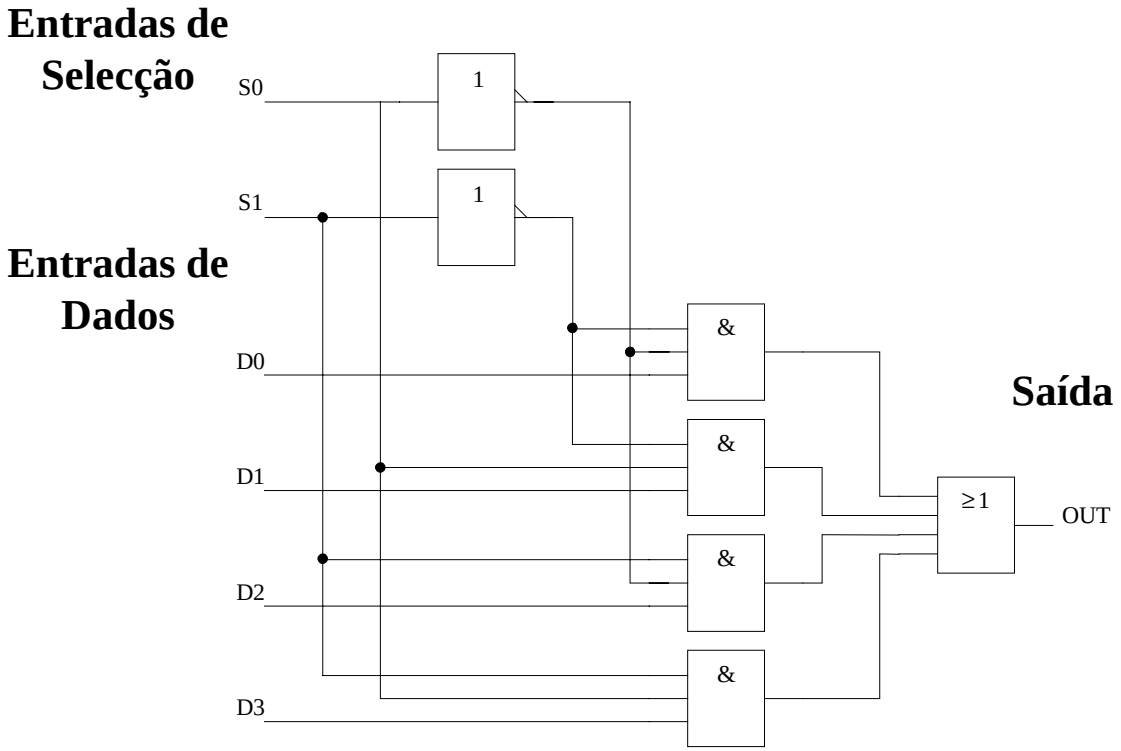
Entradas de Controlo

(Selecção e Habilitação)

MULTIPLEXERS: ARQUITECTURA

Exemplo: MULTIPLEXER 4:1

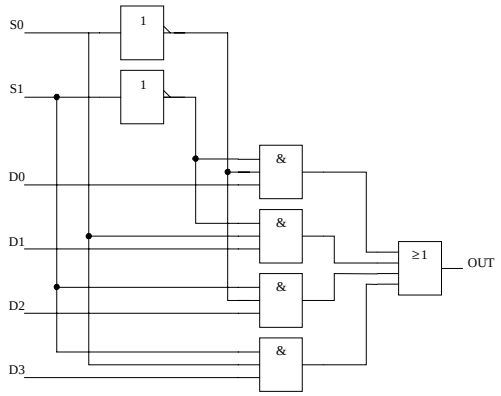
S1	S0	OUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3



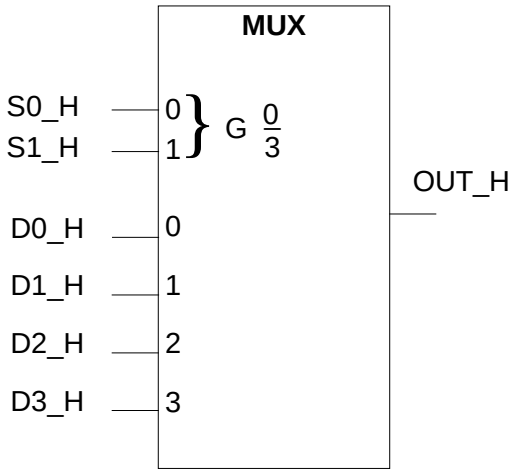
MULTIPLEXERS: SIMBOLOGIA

Exemplos:

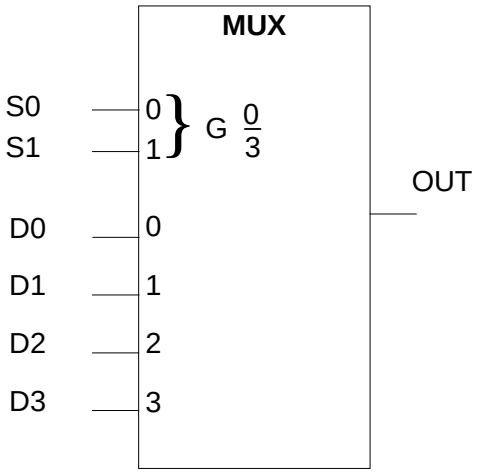
MULTIPLEXER 4:1



A designação das variáveis de entrada/saída deve indicar o nível de actividade H ou L.



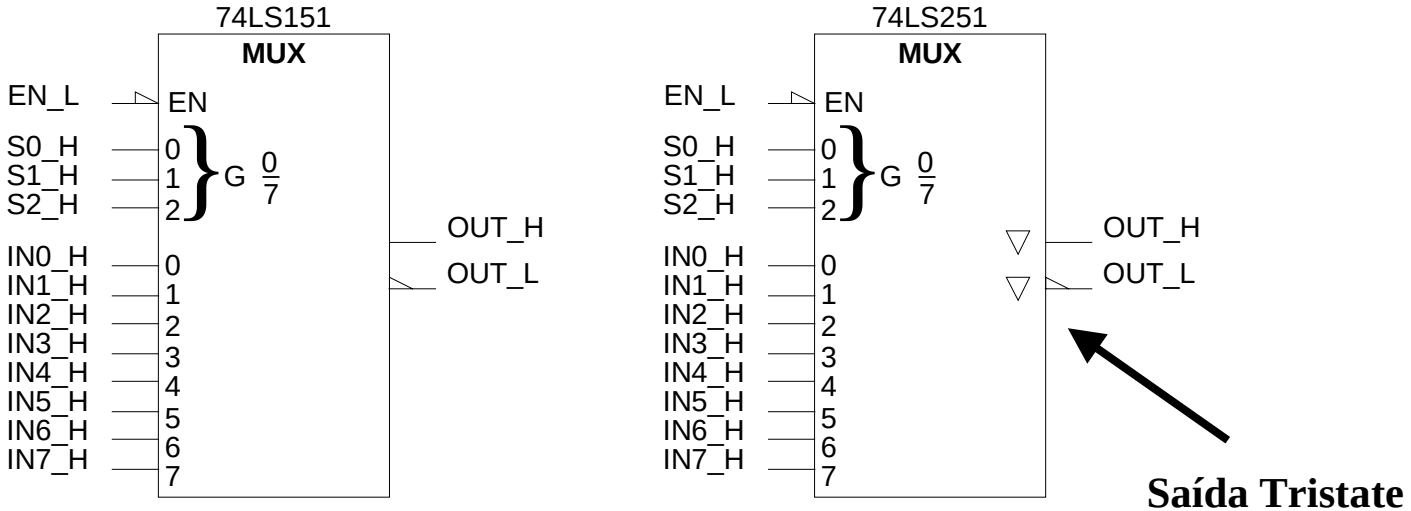
Por simplicidade sempre que não exista uma funcionalidade clara associada a uma entrada ou saída será omitido o nível de actividade. Por omissão será considerado o nível de actividade H.



MULTIPLEXERS: SIMBOLOGIA

Exemplos:

MULTIPLEXER 8:1 c/ ENABLE (74LS151), TRISTATE (74LS251)

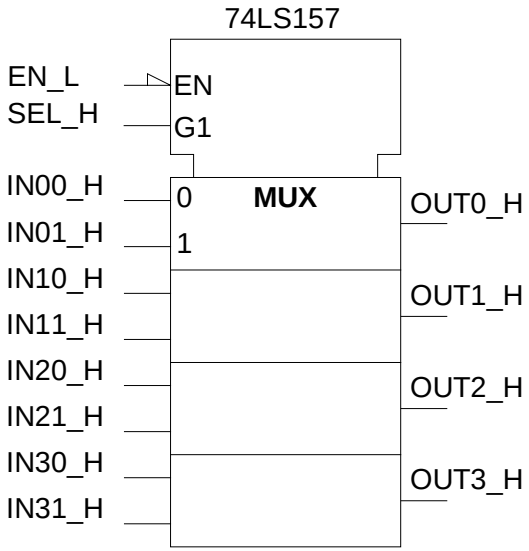


S2	S1	S0	EN_L	OUT_H	OUT_H
-	-	-	1	0	HI-Z
0	0	0	0	IN0_H	IN0_H
0	0	1	0	IN1_H	IN1_H

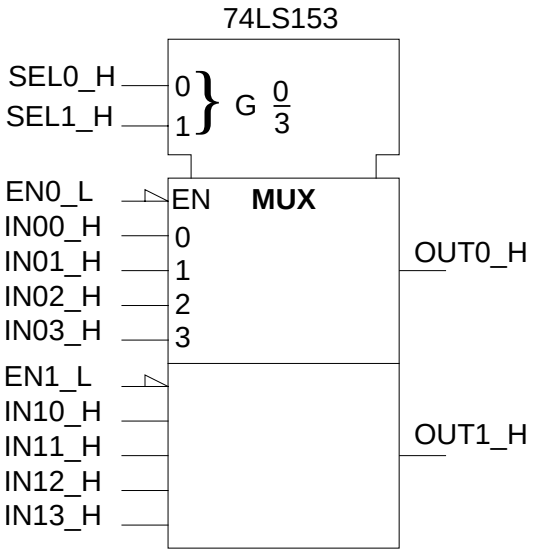
MULTIPLEXERS: SIMBOLOGIA

Exemplos:

QUAD MUX 2:1



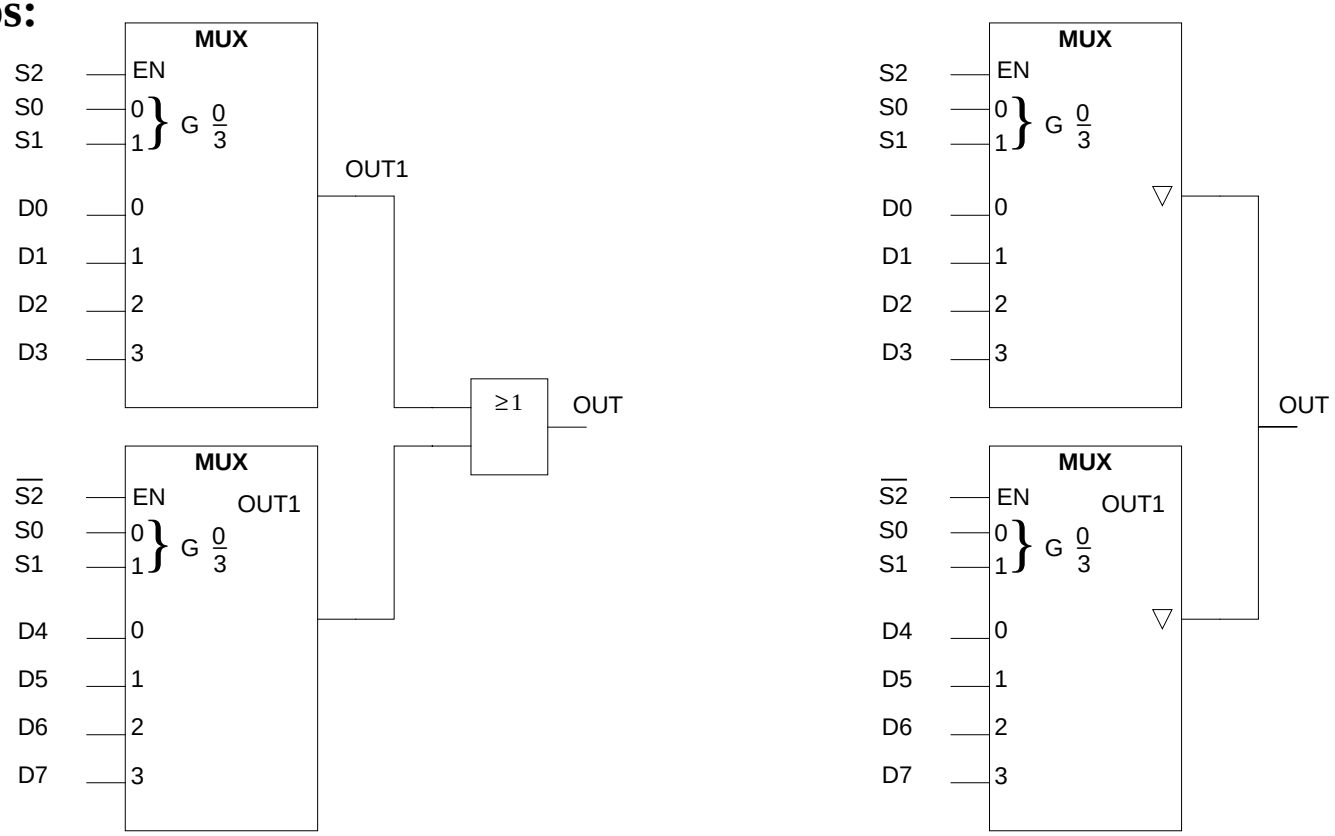
DUAL MUX 4:1



MULTIPLEXERS: APLICAÇÕES (MULTIPLEXAGEM)

Implementação de um MUX 8:1 tendo por base 2 MUX 4:1, com e sem saída tristate

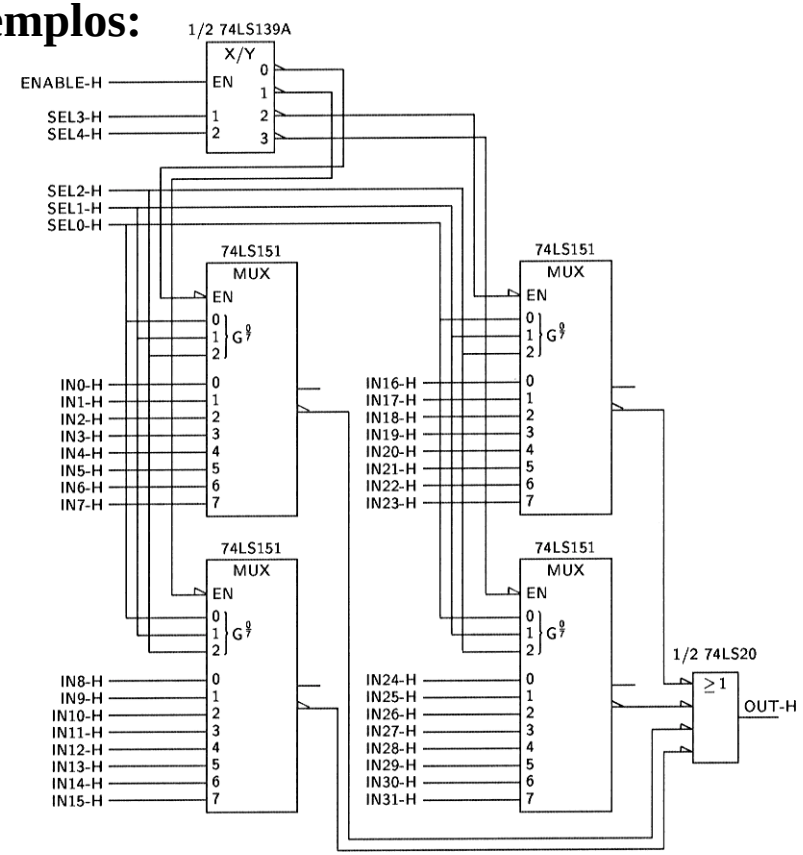
Exemplos:

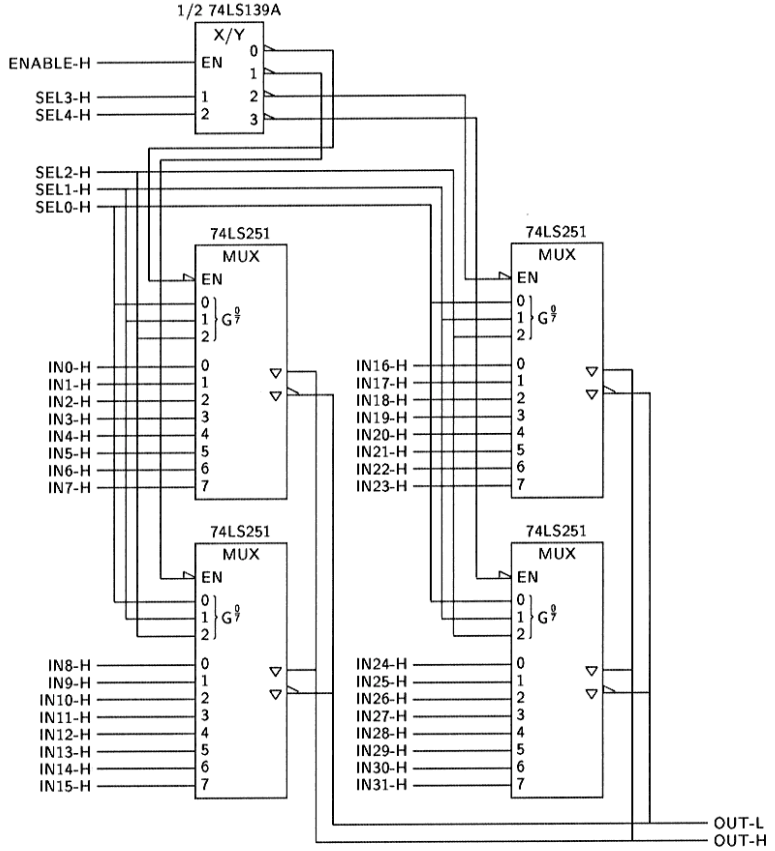


MULTIPLEXERS: APLICAÇÕES (MULTIPLEXAGEM)

Implementação de um MUX 32:1 tendo por base 4 MUX 8:1

•**Exemplos:**





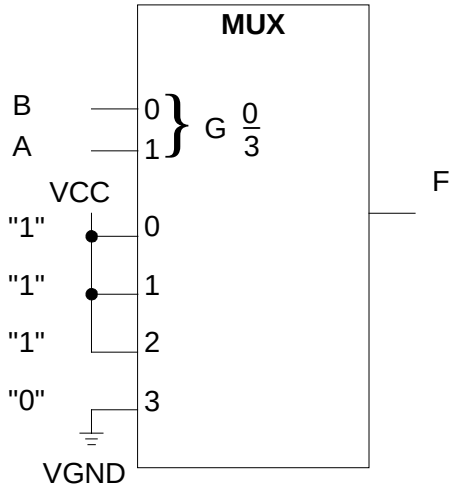
MULTIPLEXERS: APLICAÇÕES (FUNÇÕES COMBINATÓRIAS)

Implementação de F. Combinatórias de 2 variáveis com MUX 4:1

•**Exemplos:**

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

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



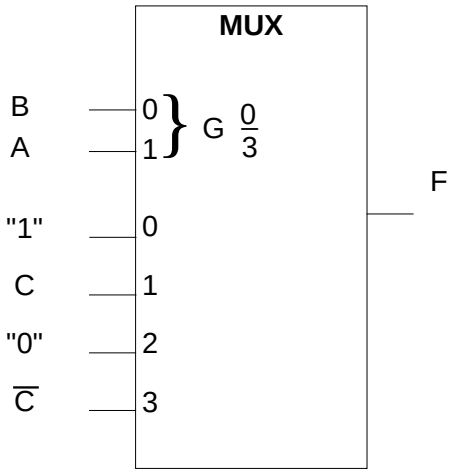
MULTIPLEXERS: APLICAÇÕES (FUNÇÕES COMBINATÓRIAS)

Implementação de F. Combinatórias de 3 variáveis com MUX 4:1

•**Exemplos:**

$$F = \overline{A}\,\overline{B} + \overline{A}\,C + AB\overline{C}$$

A	B	C	F	
0	0	0	1	$F = 1$
0	0	1	1	
0	1	0	0	$F = C$
0	1	1	1	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = \overline{C}$
1	1	1	0	

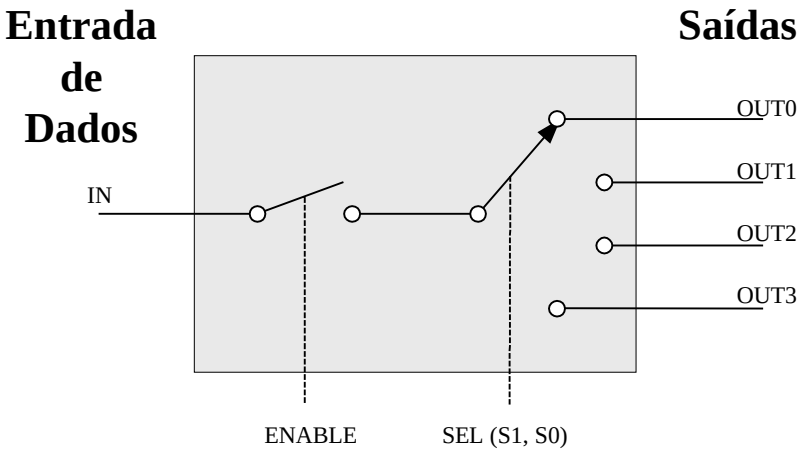


DEMULTIPLEXERS: ARQUITECTURA

O DEMULTIPLEXER é um circuito combinatório que permite, através da especificação dos sinais de selecção, encaminhar a entrada para uma das n saídas.

DEMULTIPLEXER 1:4

S1	S0	OUT0	OUT1	OUT2	OUT3
0	0	IN	0	0	0
0	1	0	IN	0	0
1	0	0	0	IN	0
1	1	0	0	0	IN



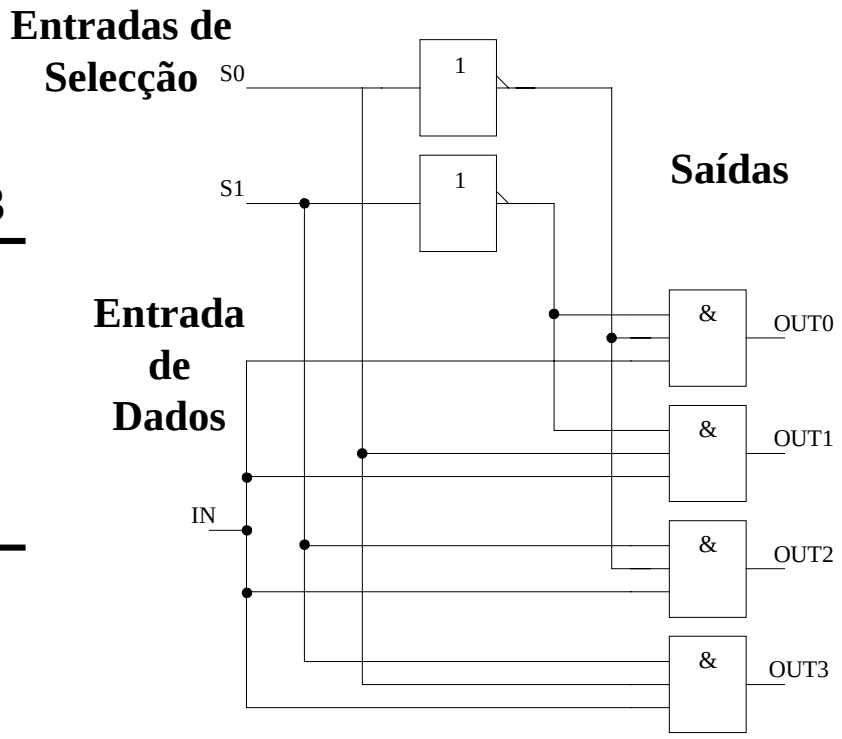
Entradas de Controlo

(Selecção e Habilitação)

DEMULTIPLEXERS: ARQUITECTURA

Exemplo: DEMULTIPLEXER 1:4

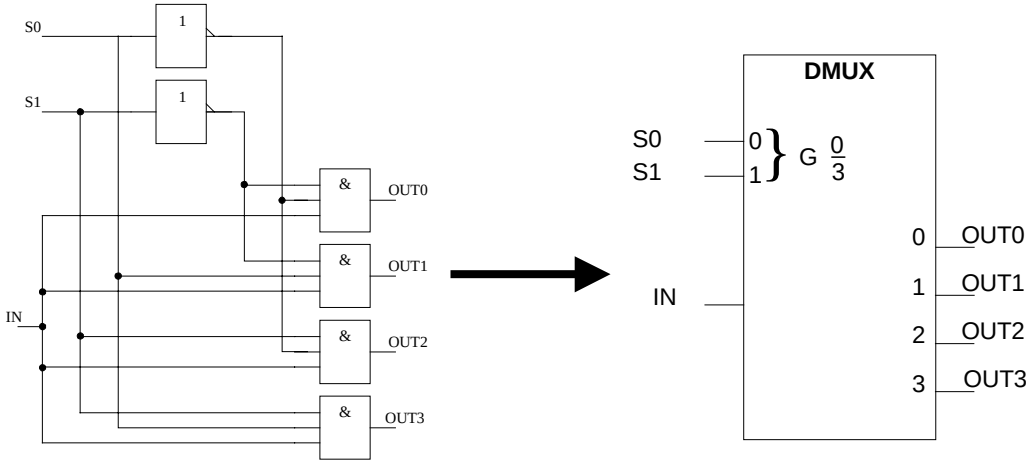
S1	S0	OUT0	OUT1	OUT2	OUT3
0	0	IN	0	0	0
0	1	0	IN	0	0
1	0	0	0	IN	0
1	1	0	0	0	IN



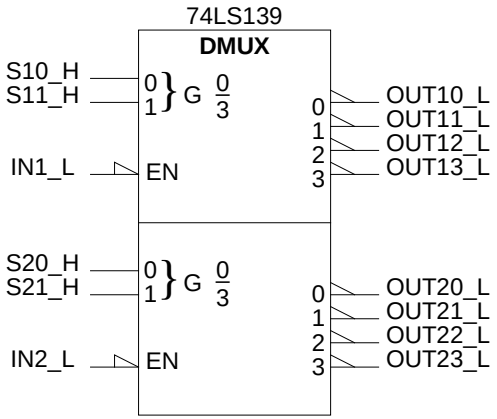
DEMULTIPLEXERS: SIMBOLOGIA

Exemplos:

DEMULTIPLEXER 1:4



DUAL DMUX 1:4



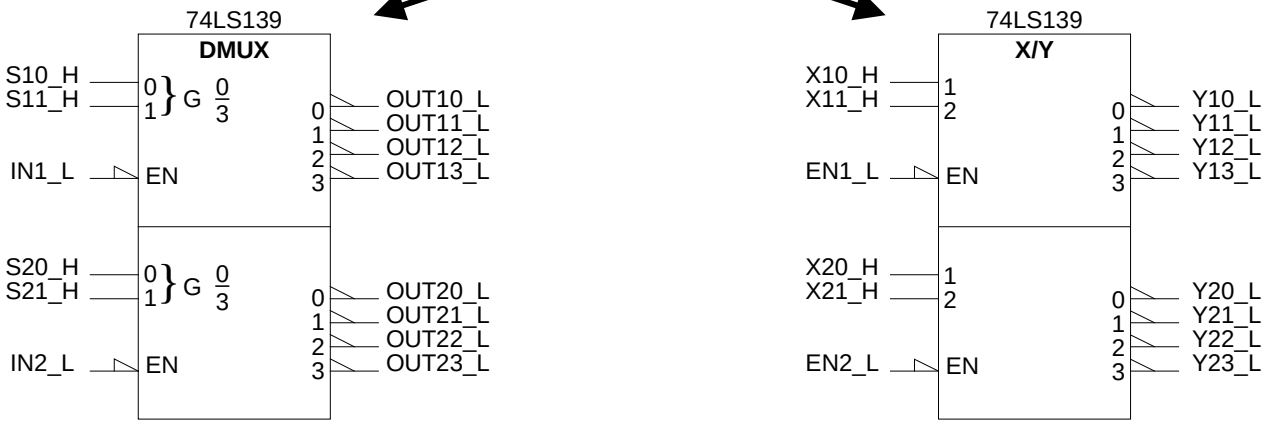
DEMULTIPLEXER E DESCODIFICADORES

Um DESCODIFICADOR com ENABLE é equivalente a um DEMULTIPLEXER, sendo as entradas de dados do primeiro as entradas de selecção do segundo e a entrada de ENABLE do primeiro a entrada de dados do segundo.

Nota: a simbologia altera-se de acordo com a funcionalidade do circuito.

DUAL DMUX 1:4 **DECODER 2:4**

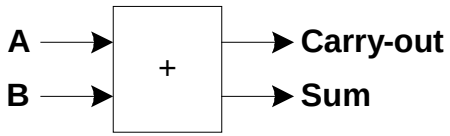
74LS139



CIRCUITO PARA SOMA ARITMÉTICA

A estrutura mais simples resolve 1 bit de cada vez.

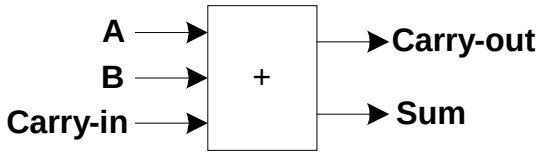
Passo 1: soma os 2 bits de menor peso e obtém 1 bit do resultado e o "carry" para o passo seguinte.



SEMI-SOMADOR

HALF-ADDER

Passo N≠1: soma os 2 bits dos operandos, mais o "carry" do passo anterior, e obtém 1 bit de resultado e um "carry-out".



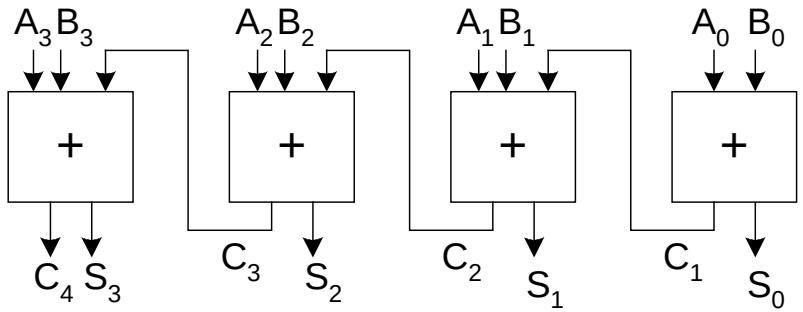
SOMADOR COMPLETO

FULL-ADDER

Exemplo

Somador de 2 números de 4 bits cada.

	1 1 0	← Carry
7	0 1 1 1	
+ 2	+ 0 0 1 0	
9	1 0 0 1	

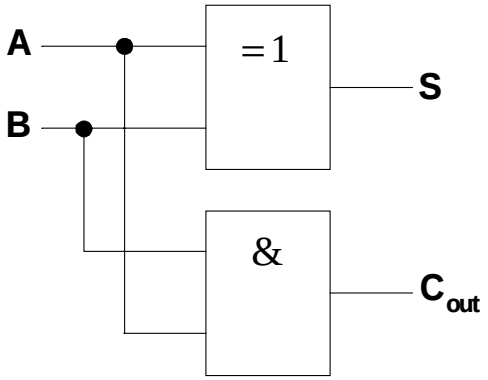


SEMI-SOMADOR

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

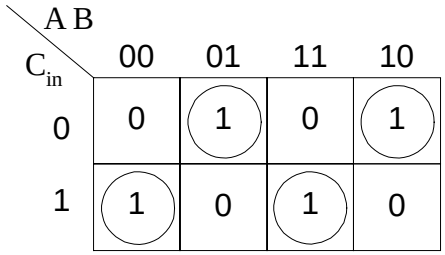
$$C_{out} = A \cdot B$$

$$S = A \oplus B$$

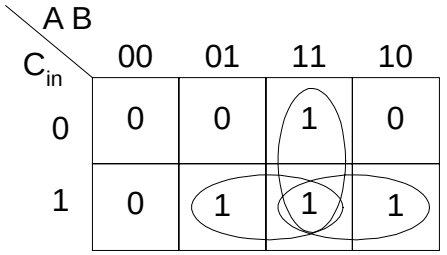


SOMADOR COMPLETO

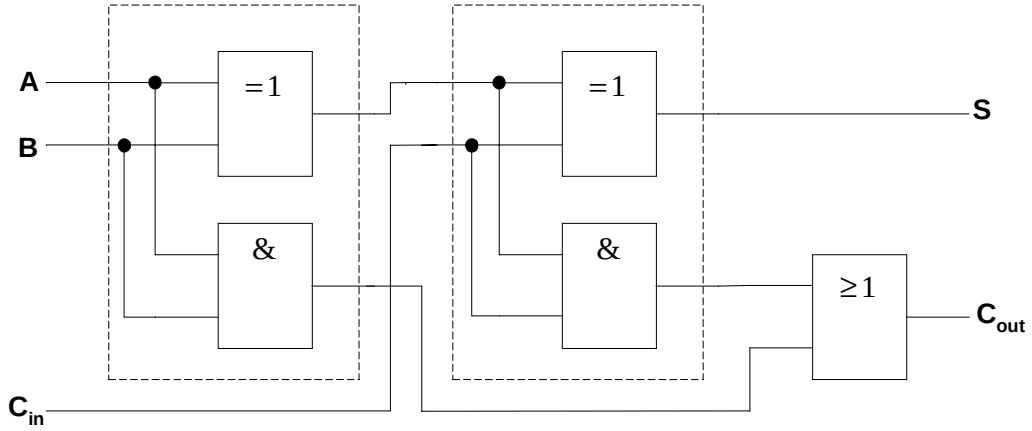
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



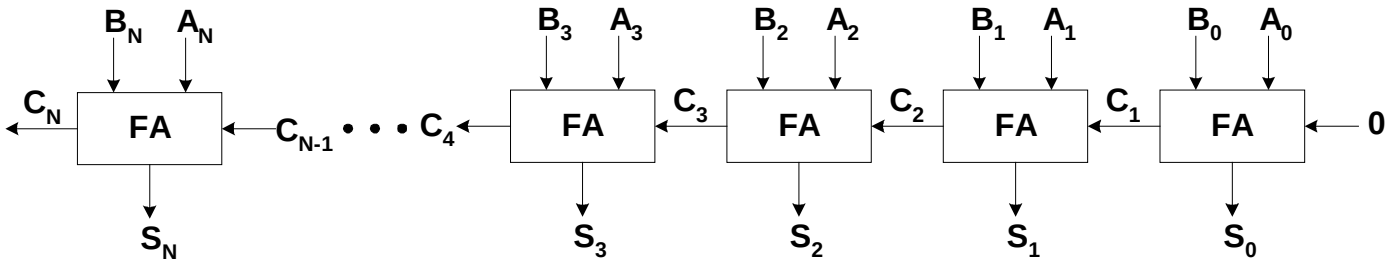
$$S = \overline{C}_{in} \cdot \overline{A} \cdot B + \overline{C}_{in} \cdot A \cdot \overline{B} + C_{in} \cdot \overline{A} \cdot \overline{B} + C_{in} \cdot A \cdot B = A \oplus B \oplus C_{in}$$



$$C_{out} = A \cdot B + C_{in} \cdot A + C_{in} \cdot B = A \cdot B + C_{in} \cdot (A \oplus B)$$



RIPPLE CARRY ADDER



A **velocidade máxima de execução é limitada** pela necessidade de propagar o "Carry" desde a soma do primeiro bit até à soma do bit mais significativo.

No pior caso, o tempo de propagação do "Carry" será $N \times t_{PFA}$.

Exemplo

A_0 comuta de 0 para 1.

$A_i = 0, \forall_{i \neq 0}$

$B_i = 1, \forall_i$

0	0	0	0				1	1	1	1	
	0	0	0	0	⇒			0	0	0	1
+	1	1	1	1			+	1	1	1	1
	1	1	1	1				0	0	0	0

O "Ripple Carry Adder" é o somador mais simples possível (que requer menos portas lógicas).

Existem inúmeros circuitos alternativos para diversos compromissos velocidade/área.

REPRESENTAÇÃO DE NÚMEROS NEGATIVOS

Sinal e Valor

O bit mais significativo representa o sinal, e os restantes bits representam o seu valor absoluto.

Ex.: -9 = 10001001

Complemento para 1

O complemento para 1 de N, em n bits, é definido como $(2^n - 1) - N$.
$2^n - 1$ é um número constituído por n 1's.
Subtrair de 1 equivale a inverter o bit: 1 – 0 = 1 e 1 – 1 = 0.
Portanto, complementar para 1 corresponde a inverter todos os bits ($0 \rightarrow 1$ e $1 \rightarrow 0$).

Ex.: -9 = 11110110 (= 11111111 – 00001001 = $255_{10} - 9_{10}$).

Complemento para 2

O complemento para 2 de N, em n bits, é definido como $2^n - N$ para $N \neq 0$, e 0 para N = 0.
Portanto, complementar para 2 corresponde a complementar para 1 e somar 1.

Ex.: -9 = 11110111 (= 100000000 – 00001001 = $256_{10} - 9_{10}$).

Na prática, o complemento para 2 pode ser formado do seguinte modo: mantém-se todos os 0's menos significativos e o primeiro 1, e invertem-se todos os outros bits mais significativos.

NÚMEROS BINÁRIOS COM SINAL

As operações usando o sistema de **sinal e valor** são mais complicadas, devido à necssidade de gerir separadamente o sinal e o valor.

Por isso, são normalmente utilizadas representações em complemento. A representação em complemento para 2 é habitualmente preferida em sistemas digitais por ter uma única representação para o 0.

Decimal	Complemento para 2	Complemento para 1	Sinal e Valor
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	-	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	-	-

SOMA ARITMÉTICA DE NÚMEROS COM SINAL USANDO COMPLEMENTO PARA 2

A soma aritmética de 2 números binários com sinal, representados em complemento para 2, é obtida pela simples adição dos 2 números **incluindo os bits de sinal**. O último "*carry out*" não é considerado.

Exemplos

				0	0	0	0					1	1	0	0
4		0	1	0	0			–4		1	1	0	0		
+ 3	+	0	0	1	1			+ (– 3)	+	1	1	0	1		
7		0	1	1	1			– 7		1	0	0	1		

				1	1	0	0					0	0	0	0
4		0	1	0	0			–4		1	1	0	0		
+ (– 3)	+	1	1	0	1			+ 3	+	0	0	1	1		
1		0	0	0	1			– 1		1	1	1	1		

SUBTRACÇÃO DE NÚMEROS COM SINAL USANDO COMPLEMENTO PARA 2

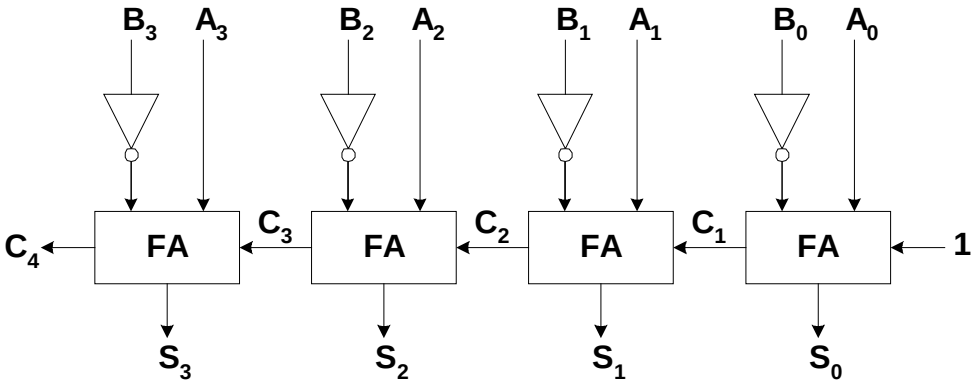
A subtracção de 2 números binários com sinal, representados em complemento para 2, é obtida do seguinte modo: forma-se o complemento para 2 do subtractor e soma-se ao subtraendo.

Exemplo

$$\begin{array}{rr} 4 & 0\ 1\ 0\ 0 \\ -\ 3 & -\ 0\ 0\ 1\ 1 \\ \hline 1 & \end{array} \quad \Rightarrow \quad \begin{array}{rr} 4 & 0\ 1\ 0\ 0 \\ +(-3) & +\ 1\ 1\ 0\ 1 \\ \hline 1 & 0\ 0\ 0\ 1 \end{array}$$

Complemento para 2 =
(Complemento para 1) + 1

A complementação para 1 é realizada invertendo todos os bits do subtractor.
A adição de 1 é efectuada pondo o *Carry* inicial a 1.

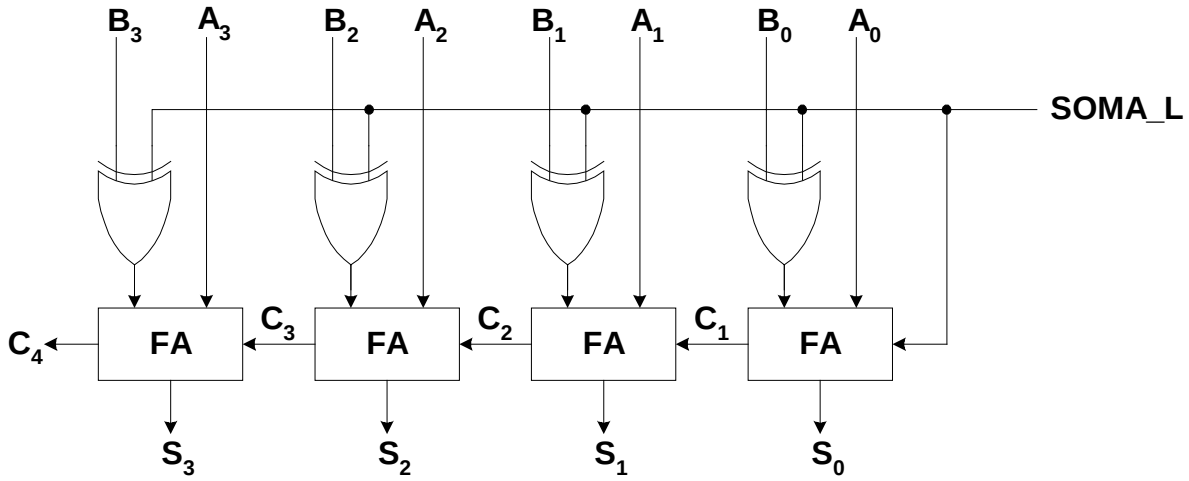


CIRCUITO SOMADOR/SUBTRACTOR

As operações de adição e subtracção são habitualmente combinadas num único somador genérico, através da inclusão de 1 porta ou-exclusivo em cada *Full-Adder*.

Quando o sinal de controlo SOMA_L = 0, é realizada a adição A + B
(os operandos B_i não são invertidos e $C_0 = 0$).

Quando o sinal de controlo SOMA_L = 1, é realizada a subtracção A – B
(os operandos B_i são invertidos e $C_0 = 1$).



OVERFLOW

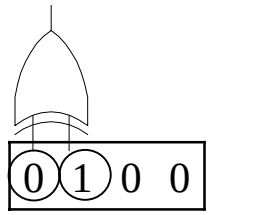
Para se obter um resultado correcto, na adição e na subtracção, é necessário assegurar que o resultado tem um número de bits suficiente. Se somarmos 2 números de N bits e o resultado ocupar N+1 bits diz-se que ocorreu um ***overflow***.

As unidades aritméticas digitais usam um número fixo de bits para armazenar os operandos e os resultados, sendo necessário detectar e sinalizar a ocorrência de um *overflow*.

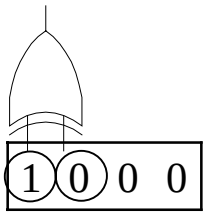
Um overflow pode ocorrer na adição, se os 2 operandos são ambos positivos ou se são ambos negativos.

$$Overflow = Carry_N \oplus Carry_{N-1}$$

Exemplos



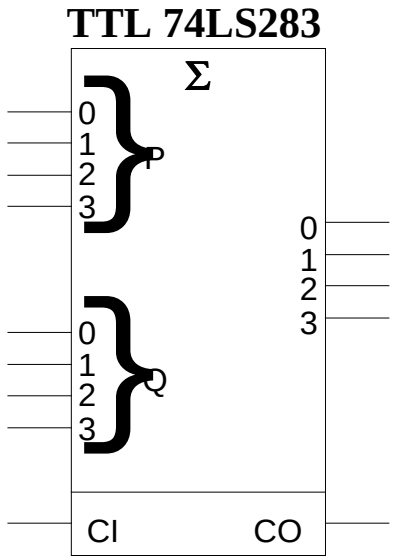
4	0 1 0 0
+ 5	+ 0 1 0 1
ovfl.	1 0 0 1



– 4	1 1 0 0
+ (– 5)	+ 1 0 1 1
ovfl.	0 1 1 1

SOMADOR DE 4 BITS

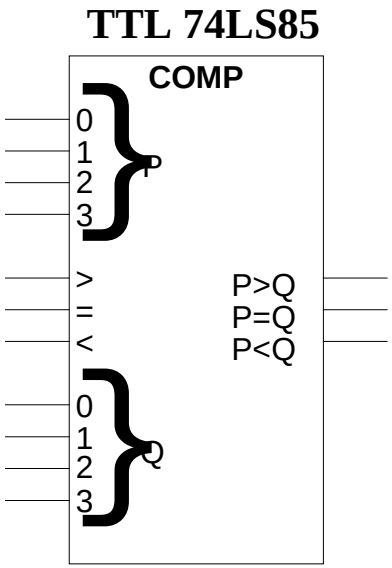
Somador de 4 bits rápido: usa internamente um "carry look-ahead" – trata-se de um somador de maior rapidez, conseguida à custa de um maior número de portas lógicas.



COMPARADOR DE NÚMEROS DE 4 BITS

Este circuito faz a comparação de 2 números binários de 4 bits.

O circuito pode ser ligado em cascata, para realizar comparações entre números de N > 4 bits.



BIBLIOGRAFIA

[1] M. Morris Mano, Charles R. Kime, “Logic and Computer Design Fundamentals”, Prentice-Hall International, Inc. (Capítulo 3, Secções 3.5-3.10)