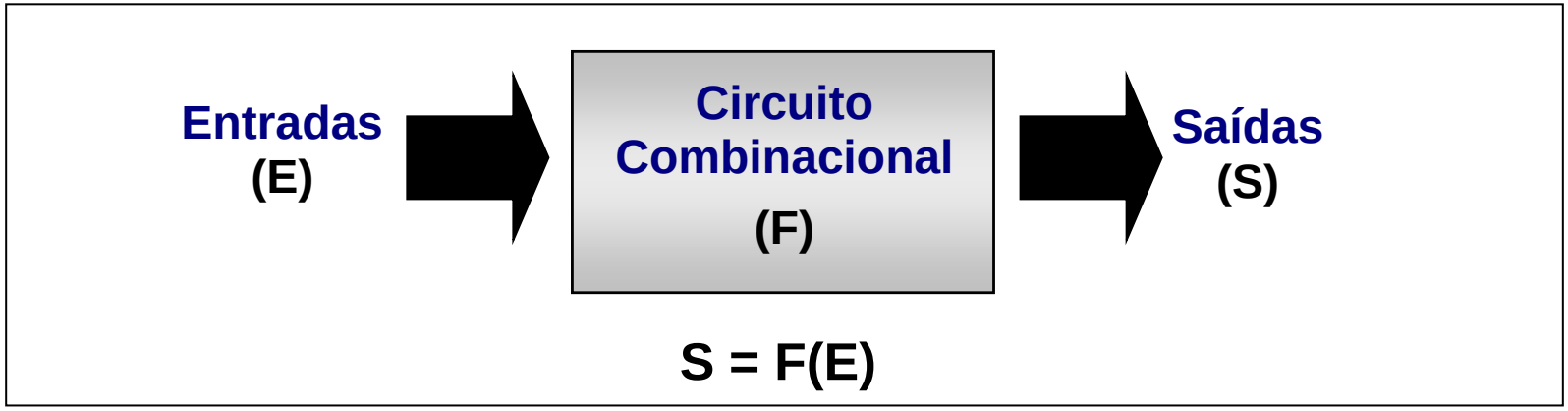
Circuitos Combinacionais

Circuitos combinacionais

Um **circuito combinacional** é um circuito em que o valor das saídas em qualquer instante de tempo, depende unicamente dos valores lógicos presentes nas entradas, nesse instante.

Um circuito deste tipo executa uma função lógica **F**, através da interligação de várias **portas lógicas**.



Modelo genérico de um circuito combinacional

Circuitos combinacionais dedicados

Existem diversos circuitos combinacionais que são frequentemente utilizados em projectos de sistemas digitais, devido às funções lógicas que implementam.

Estes circuitos encontram-se comercialmente disponíveis sob a forma de circuitos integrados e, por isso, designam-se por **circuitos combinacionais dedicados**.

Exemplos destes circuitos:

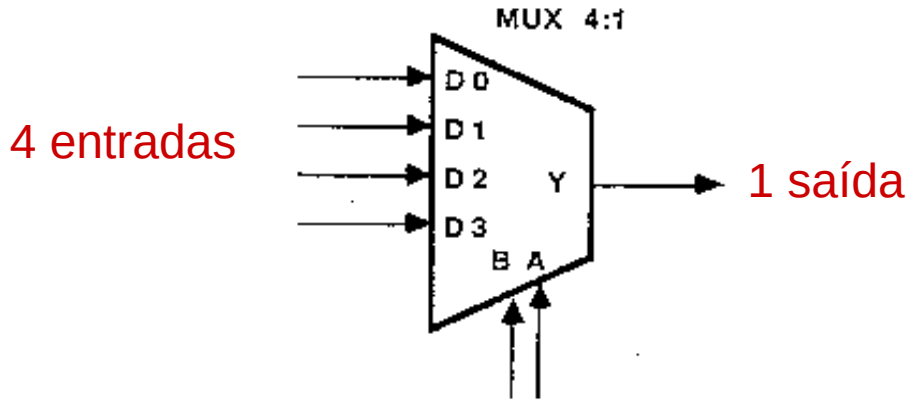
Codificadores (*Encoders*), **Descodificadores** (*Decoders*), **Multiplexadores** (*Multiplexers*), **Desmultiplexadores** (*Demultiplexers*), **Comparadores** (*Comparators*), **Somadores** (*Binary Adders*), ...

Multiplexadores (MUX)

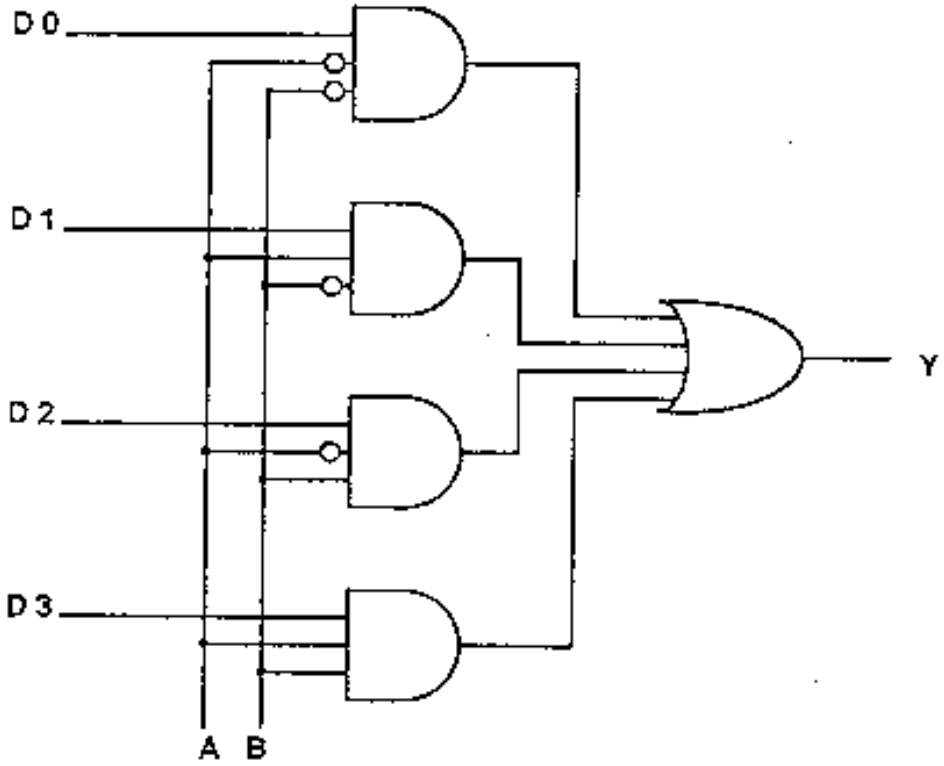
Os **Multiplexadores** ou **MUX** são circuitos que, possuindo **várias entradas** e **uma só saída**, permitem seleccionar uma dessas entradas reproduzindo-a na saída. Essa selecção é feita através de um **código binário** que é aplicado às **entradas** (**ou linhas**) **de selecção**.

Estes circuitos são utilizados para diversas finalidades, como sejam: selecção de dados, conversão paralelo-série, implementação de funções lógicas, …

Exemplo de um **MUX** com **4** entradas de dados, ou **MUX 4:1**:



Implementação de um **MUX 4:1**:



$$Y = (\bar{B} \,.\, \bar{A}).D_0 + (\bar{B} \,.\, A).D_1 + (B \,.\bar{A}).D_2 + (B \,.A).D_3$$

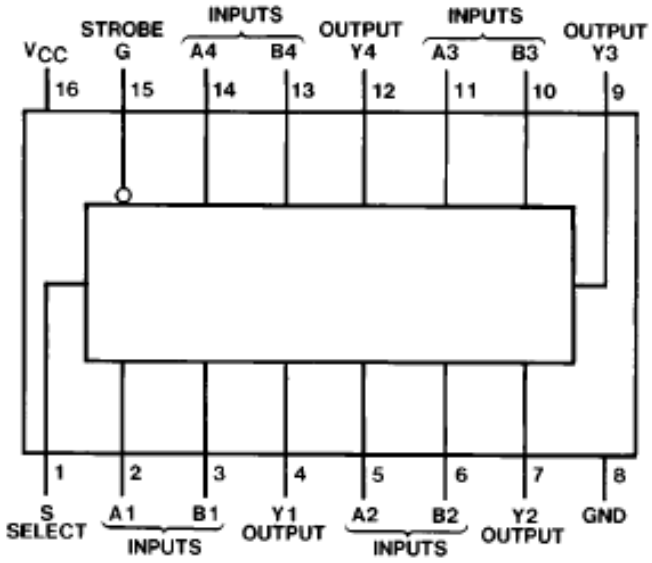
Apresentam-se em seguida, alguns exemplos de Multiplexadores disponíveis no mercado sob a forma de circuitos integrados da família TTL.

A informação relativa aos diversos circuitos integrados, foi retirada das respectivas *data sheets* (*www.datasheetcatalog.com*).

- Circuito comercial **74157** (4 MUX 2:1):

Connection Diagram

Dual-In-Line Package



Function Table

Inputs				Output Y
Strobe	Select	A	B	
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

H = High Level, L = Low Level, X = Don't Care

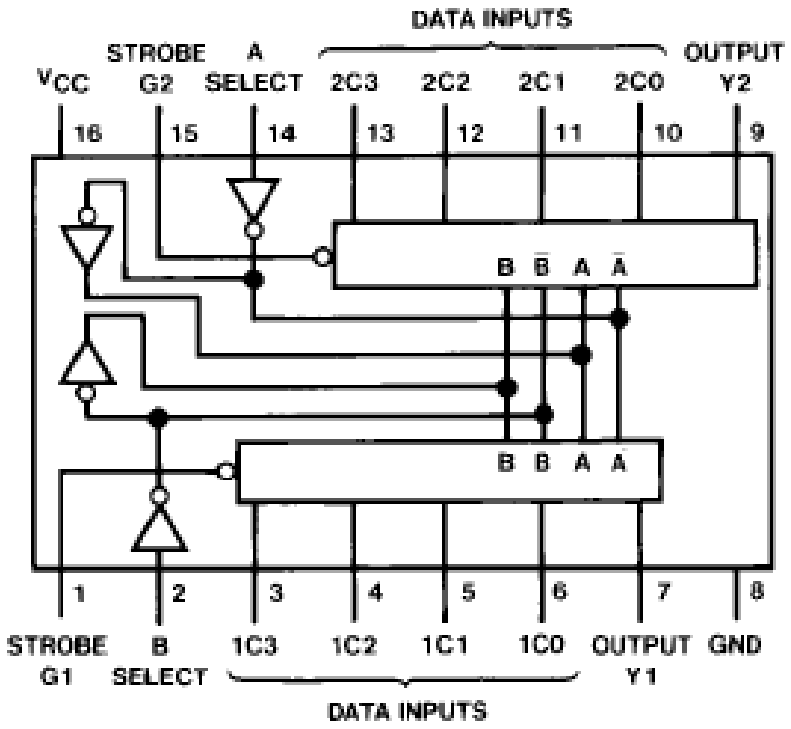
Relativamente ao circuito anterior, verifica-se o seguinte:

- **Strobe** = **H** → "desactiva" o circuito
- **Strobe** = **L** → "deixa o circuito funcionar"
- As letras **X** presentes nas entradas da tabela de verdade, significam: "*qualquer que seja o valor lógico da entrada...*"
- **Select** selecciona uma das entradas **A** ou **B** → esta selecção é comum aos 4 Multiplexadores
- A saída **Y** reproduz a entrada seleccionada

- Circuito comercial **74153** (2 MUX 4:1):

Connection Diagram

Dual-In-Line Package



Function Table

Select Inputs		Data Inputs				Strobe	Output
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

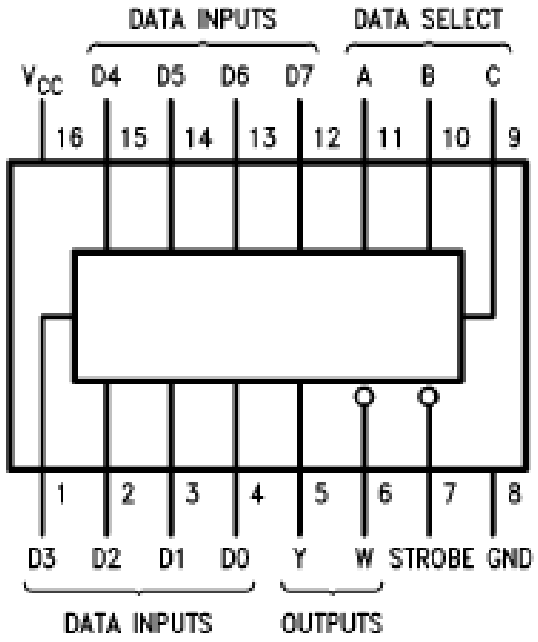
Select inputs A and B are common to both sections.

H = High Level, L = Low Level, X = Don't Care

- Circuito comercial **74151** (1 MUX 8:1):

Connection Diagram

Dual-In-Line Package



Function Table

54151A/75151A

Inputs				Outputs	
Select			Strobe S	Y	W
C	B	A			
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

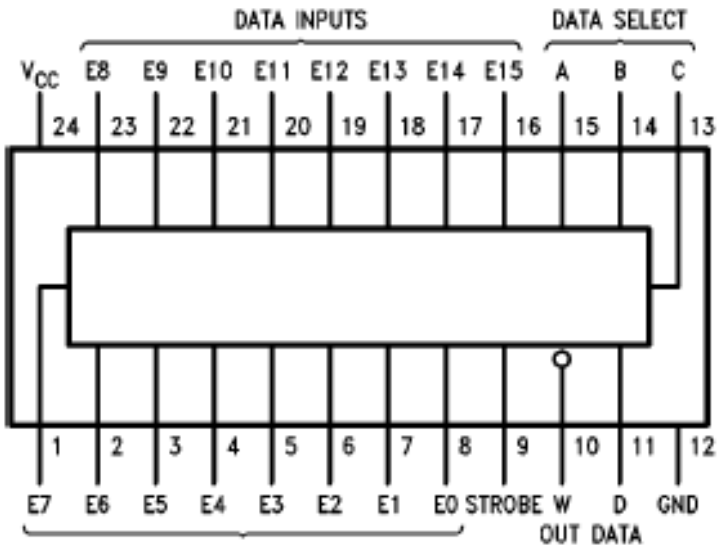
H = High Level, L = Low Level, X = Don't Care

Note-se que as saídas **Y** e **W** são complementares uma da outra.

- Circuito comercial **74150** (1 MUX 16:1):

Connection Diagram

Dual-In-Line Package



Function Table

54150/74150

Inputs					Outputs
Select				Strobe	
D	C	B	A	S	W
X	X	X	X	H	H
L	L	L	L	L	$\overline{E0}$
L	L	L	H	L	$\overline{E1}$
L	L	H	L	L	$\overline{E2}$
L	L	H	H	L	$\overline{E3}$
L	H	L	L	L	$\overline{E4}$
L	H	L	H	L	$\overline{E5}$
L	H	H	L	L	$\overline{E6}$
L	H	H	H	L	$\overline{E7}$
H	L	L	L	L	$\overline{E8}$
H	L	L	H	L	$\overline{E9}$
H	L	H	L	L	$\overline{E10}$
H	L	H	H	L	$\overline{E11}$
H	H	L	L	L	$\overline{E12}$
H	H	L	H	L	$\overline{E13}$
H	H	H	L	L	$\overline{E14}$
H	H	H	H	L	$\overline{E15}$

H = High Level, L = Low Level, X = Don't Care

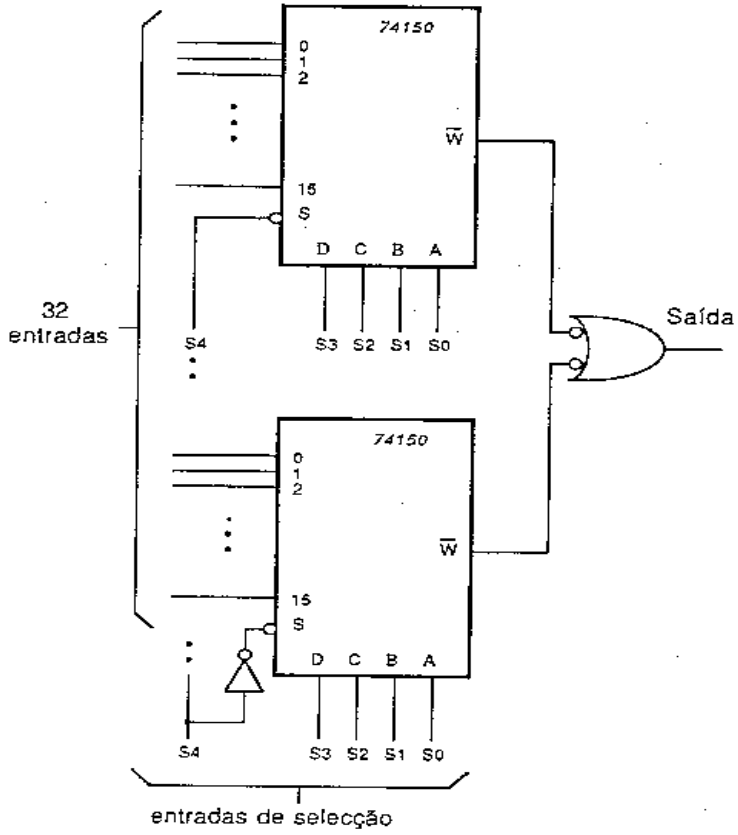
$\overline{E0}$, $\overline{E1}$ . . . $\overline{E15}$ = the complement of the level of the respective E input

Expansão de Multiplexers

Para se multiplexarem mais sinais do que aqueles que um MUX permite, **associam-se vários MUX**.

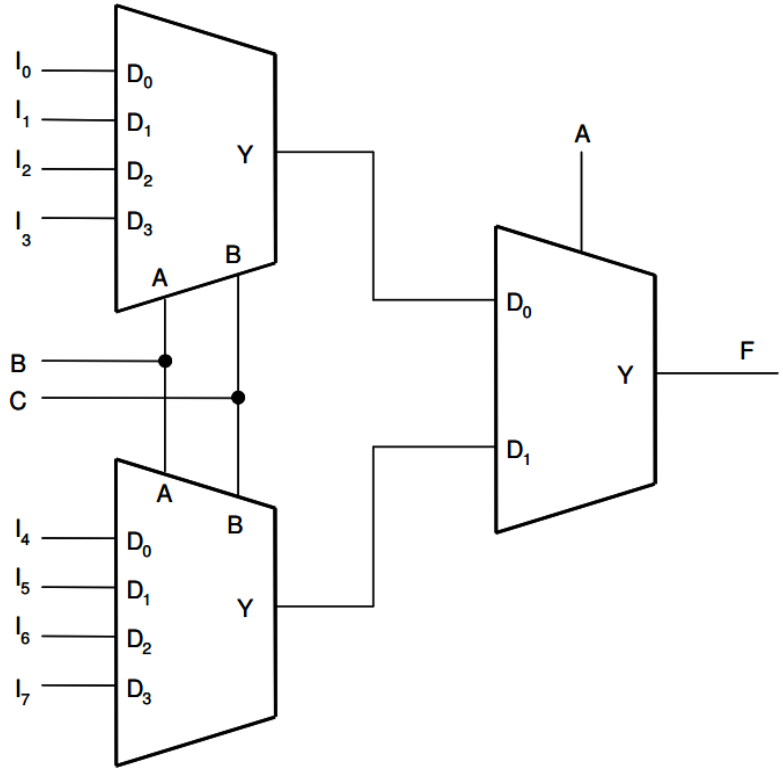
Por exemplo, para se obter um MUX de 32 entradas, podem associar-se dois MUX de 16 entradas cada.

- O sinal **Strobe** comporta-se como "mais uma linha de selecção", **S4**
- Como no **74150** a saída **W** é negada e é **H** quando circuito inactivo, há que invertê-la (⇒ **L**) e há que fazer um **OR** das saídas



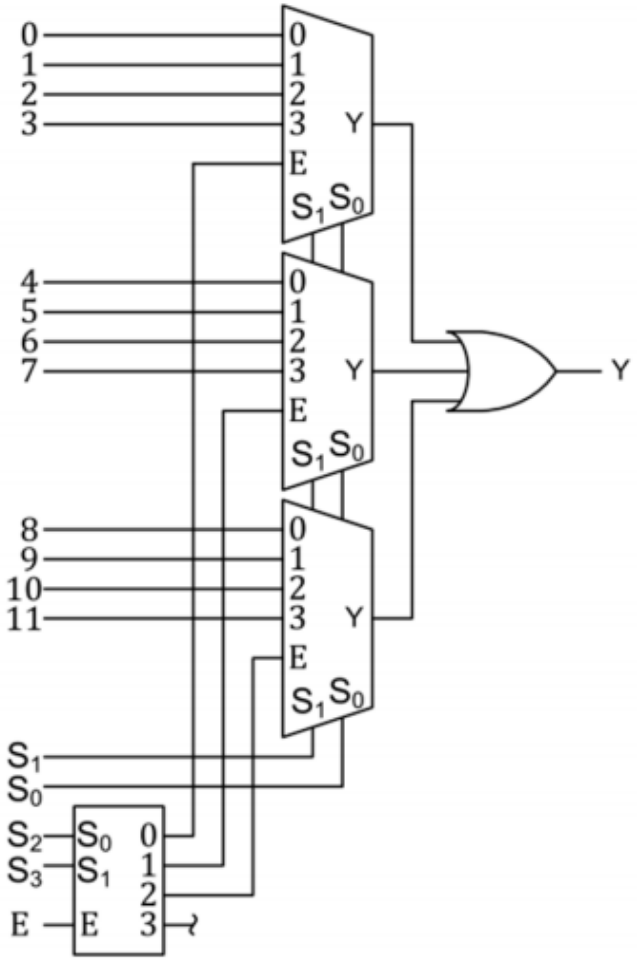
Expansão de Multiplexers

A figura apresenta um MUX de 8 entradas construído a partir de dois multiplexadores de 4 entradas e um outro multiplexador de duas entradas, considerando que a entrada A é a mais significativa.



Expansão de Multiplexers com descodificador

Outro método alternativo utiliza um descodificador para seleccionar um dos multiplexers.

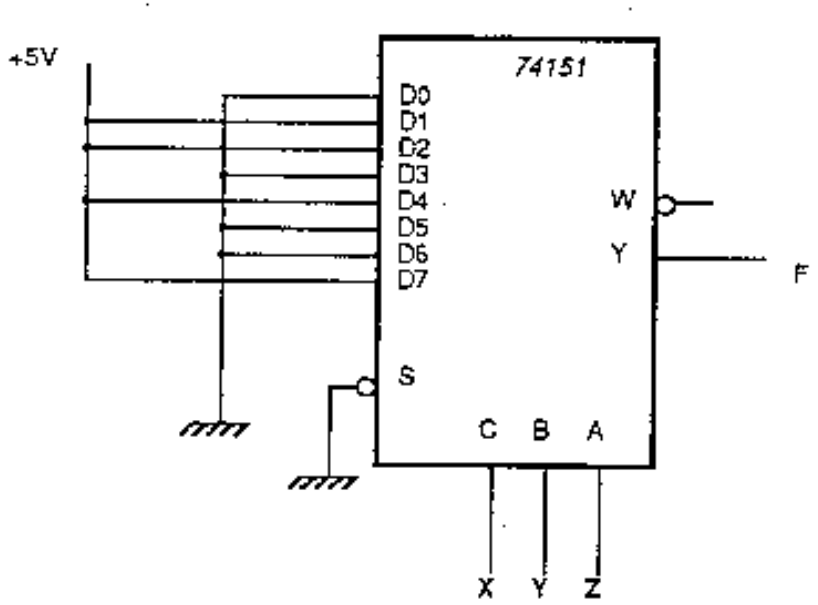


Implementação de funções lógicas com Multiplexers

Como foi inicialmente referido, os MUX podem ser usados para **implementar funções lógicas**.

Por exemplo, a seguinte **função F de 3 variáveis** pode ser implementada com, unicamente, um **MUX 8:1**:

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



Com efeito, verifica-se que qualquer função de **n** variáveis pode ser implementada usando apenas um **MUX** de $\mathbf{2^n:1}$.

Mas, para além disso, um **MUX** também permite implementar uma função lógica com redução de **uma ou mais variáveis**.

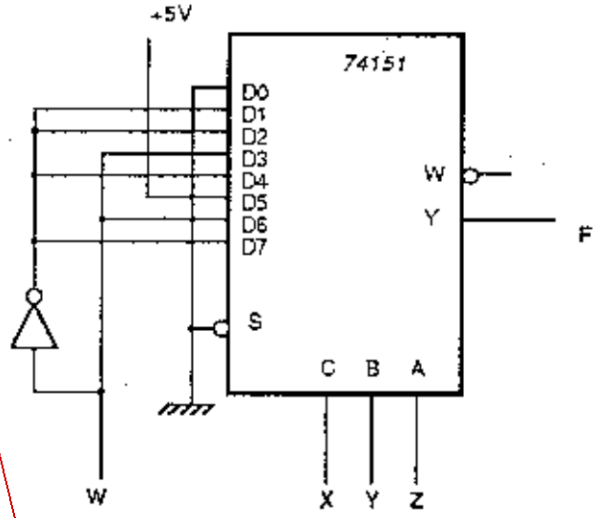
Exemplo

Imagine que se pretende implementar a **função F de 4 variáveis**, dada pela tabela ao lado, com base num **MUX 8:1**.

X	Y	Z	W	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

X	Y	Z	F
0	0	0	0
0	0	1	$\overline{W}$
0	1	0	$\overline{W}$
0	1	1	W
1	0	0	$\overline{W}$
1	0	1	1
1	1	0	W
1	1	1	$\overline{W}$

Construção da Tabela Reduzida



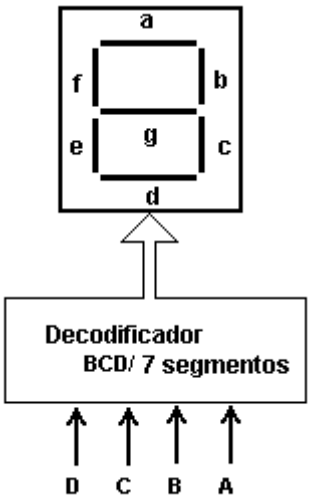
Implementação com um MUX 8:1

Descodificadores (DEC)

Os **Descodificadores**, ou **DEC**, são circuitos que recebem um conjunto de entradas que representam um dado número binário, e activam apenas a saída que corresponde a esse número (todas as outras saídas permanecem inactivas).

Se um descodificador possuir **n** entradas, há $\mathbf{2^n}$ possibilidades de combinações, ou códigos, de entrada. No entanto, existem descodificadores que não utilizam todas as $\mathbf{2^n}$ possibilidades, mas apenas algumas delas.

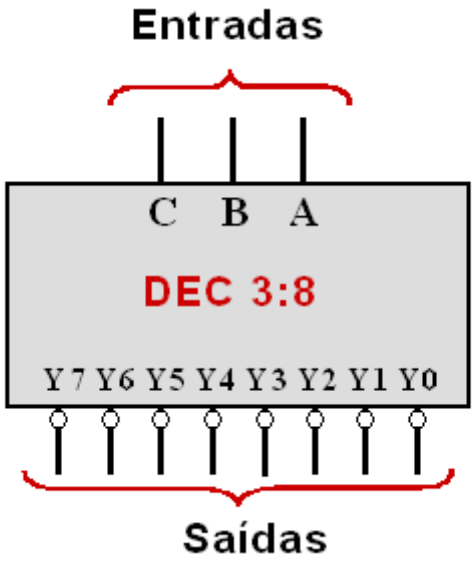
Algumas aplicações dos Descodificadores



Uma das maiores aplicações dos **Descodificadores** consiste no accionamento de *displays*: interpretando o código binário presente nas entradas (representante de letras ou números) e gerando os sinais adequados para ligar o dígito correspondente a esse código.

Por outro lado, se as entradas de um **Descodificador** forem geradas por um circuito sequencial designado por **Contador** que gera uma sequência de códigos binários (a estudar mais à frente), as suas saídas podem ser usadas como sinais de temporização ou de sequenciamento, para ligar/desligar determinados dispositivos em dados momentos.

Exemplo de um **Descodificador** ou **DEC 3:8** (3 linhas de entrada e 8 linhas de saída):



C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Note-se que as saídas são ***active-low*** (ou seja, activas ao ***nível baixo***)

Desmultiplexadores (DEMUX)

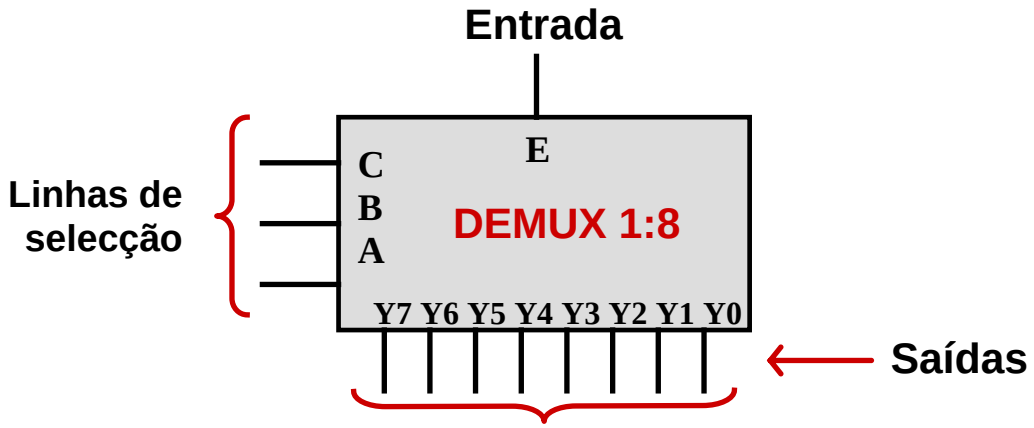
Um **Desmultiplexador**, ou **DEMUX**, é um circuito que realiza a operação inversa de um **Multiplexador**. Possuindo uma única entrada e várias saídas, transfere o valor da entrada para a saída seleccionada pelo código binário presente nas linhas de selecção.

Algumas aplicações dos Desmultiplexadores

Um **DEMUX** pode, por exemplo, ser usado como Desmultiplexador de *Clock* (relógio), direccionando este sinal para o destino determinado pelas linhas de selecção.

Pode igualmente ser usado como parte de um sistema síncrono de transmissão de dados em série, entre um emissor e um receptor remoto (existirá um **MUX** no lado emissor e um **DEMUX** no lado receptor).

Exemplo de um **Desmultiplexador** de 1 entrada e 8 saídas, ou **DEMUX 1:8**:



C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	E	1	1	1	1	1	1	1
0	0	1	1	E	1	1	1	1	1	1
0	1	0	1	1	E	1	1	1	1	1
0	1	1	1	1	1	E	1	1	1	1
1	0	0	1	1	1	1	E	1	1	1
1	0	1	1	1	1	1	1	E	1	1
1	1	0	1	1	1	1	1	1	E	1
1	1	1	1	1	1	1	1	1	1	E

Note-se que **E** é a informação presente na entrada de dados

Pela observação das tabelas de verdade relativas ao **Descodificador** e ao **Desmultiplexador**, pode concluir-se que é simples combiná-los a ambos num único dispositivo.

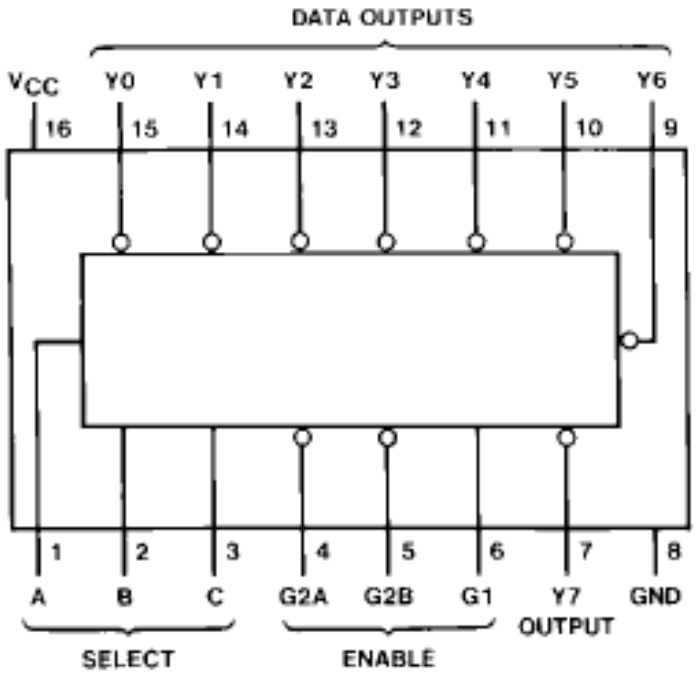
Com efeito, esta solução é adoptada pelos circuitos comercialmente disponíveis, os quais podem ser usados quer como Desmultiplexadores, quer como Descodificadores. É por este motivo que são designados por ***Descodificadores/Desmultiplexadores.***

Apresentam-se em seguida alguns exemplos destes circuitos, disponíveis no mercado sob a forma de CIs da família TTL.

- Circuito comercial **74138** (um DEC/DEMUX 3:8):

Connection Diagram

DM74LS138



Function Table

DM74LS138

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 1)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

Note 1: G2 = G2A + G2B

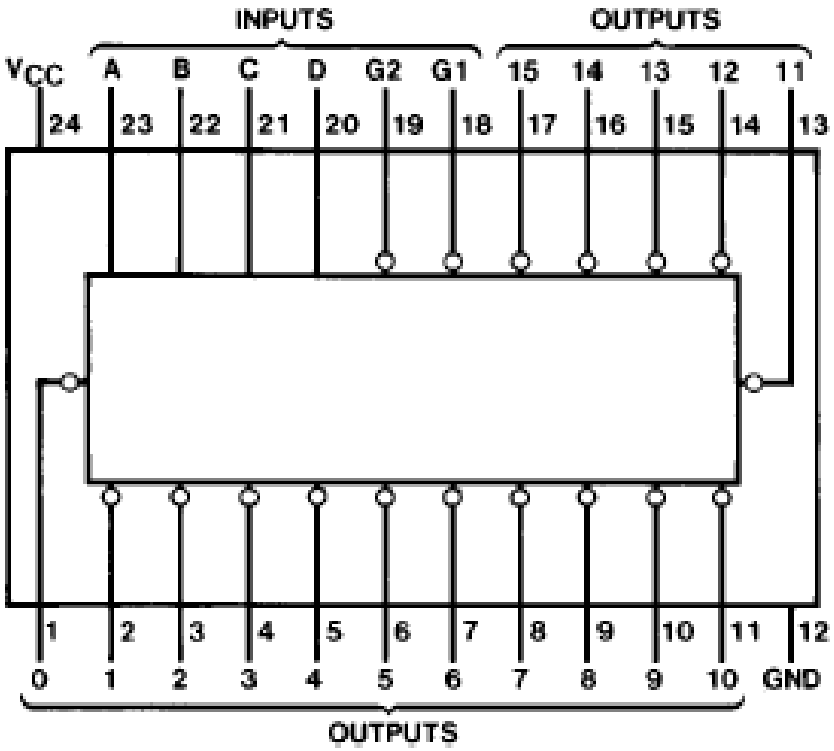
Notas:

- Este circuito tem três entradas de *Enable*: duas activas ao nível **L** e uma activa ao nível **H**.
- Estas entradas evitam a necessidade de lógica externa em caso de expansão.
- Uma destas entradas pode ser usada como entrada de dados em situações de “desmultiplexação”.

- Circuito comercial **74154** (um DEC/DEMUX 4:16):

Connection Diagram

Dual-In-Line Package



Function Table

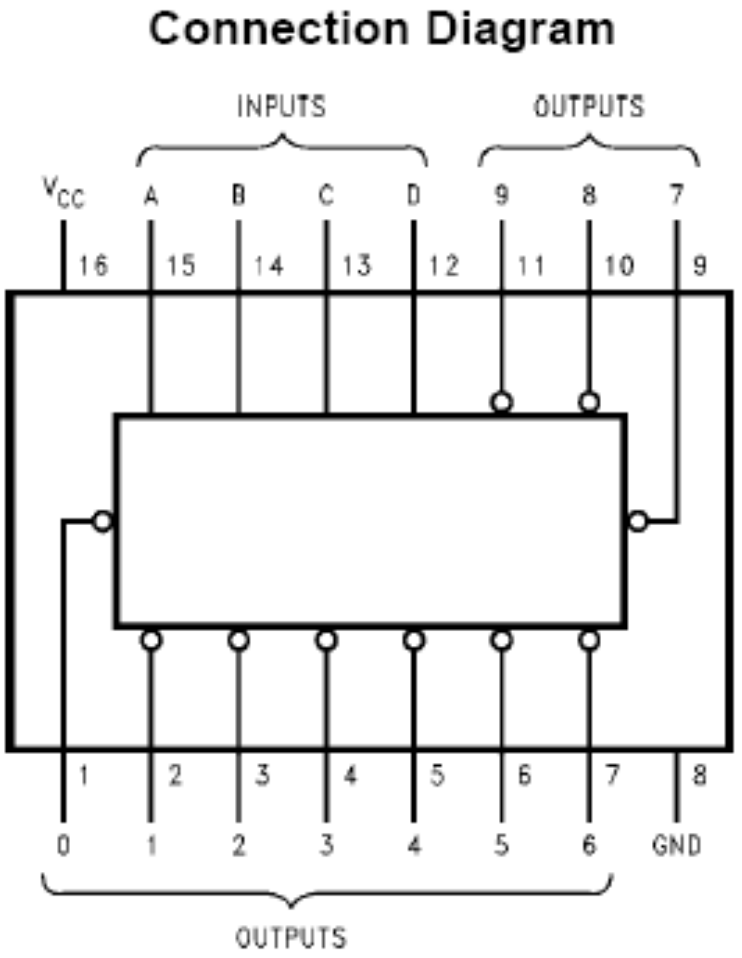
Inputs						Outputs															
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = High Level, L = Low Level, X = Don't Care

- Circuito comercial **7442** (um DEC/DEMUX 4:10):

Este é um descodificador de BCD para decimal:

- Tem apenas 10 combinações de entrada válidas (as outras 6 são inválidas)
- Utiliza-se para "fazer corresponder" aos códigos binários de 0000 a 1001, dez saídas representativas dos dez dígitos decimais 0...9



Function Table

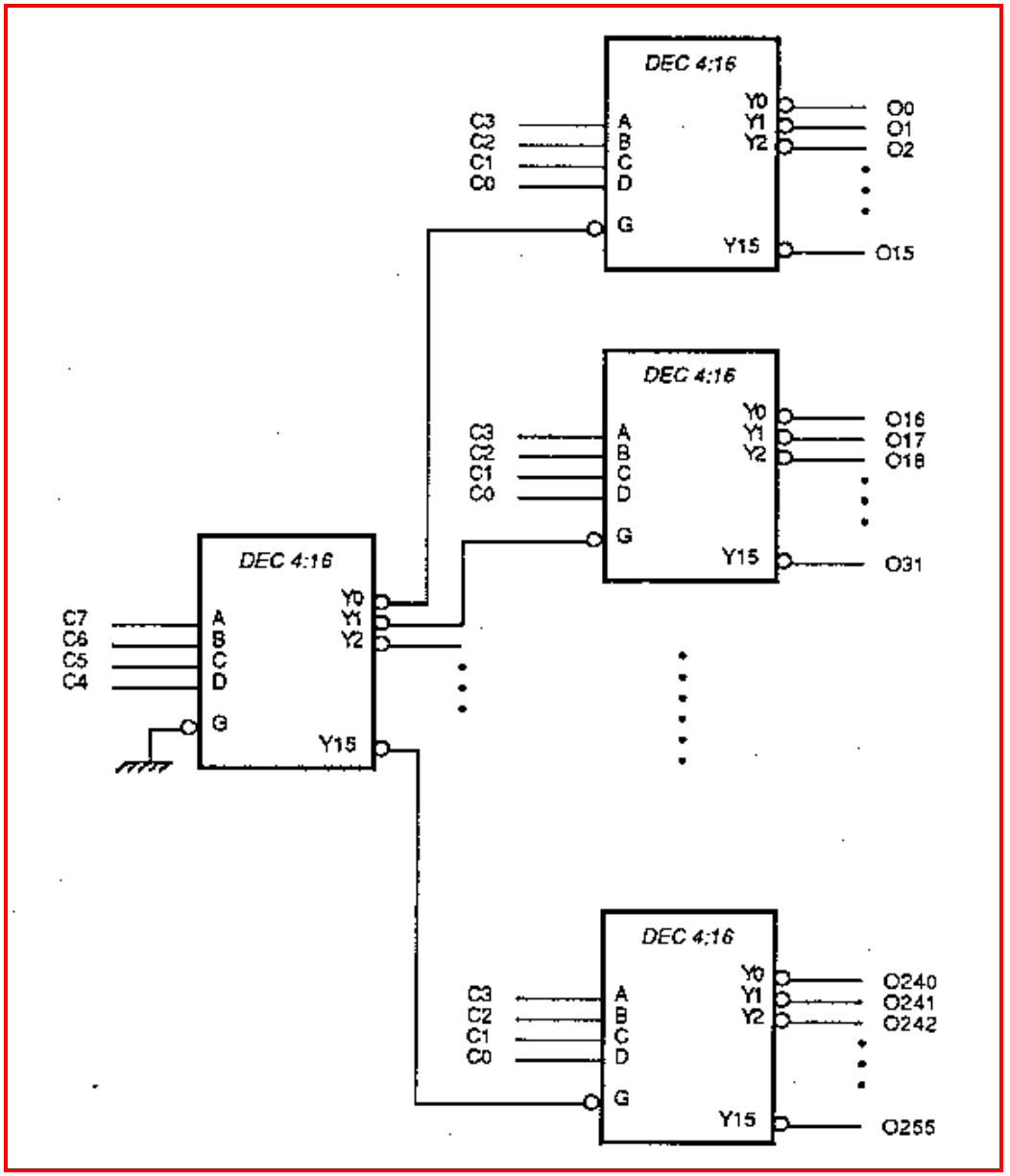
No.	BCD Input				Decimal Output									
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
I	H	L	H	L	H	H	H	H	H	H	H	H	H	H
N	H	L	H	H	H	H	H	H	H	H	H	H	H	H
V	H	H	L	L	H	H	H	H	H	H	H	H	H	H
A	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	H	H	H	L	H	H	H	H	H	H	H	H	H	H
I	H	H	H	H	H	H	H	H	H	H	H	H	H	H
D														

H = HIGH Level
L = LOW Level

Expansão de descodificadores

Para se obterem circuitos descodificadores com mais saídas do que aquelas que um só **DEC** permite, **associam-se vários DEC**.

Veja-se, por exemplo, como é possível obter um **DEC 8:256**, à custa de 1+16=**17 DEC 4:16**.



Implementação de funções lógicas com descodificadores

Tal como os **Multiplexadores**, também os **Descodificadores** podem ser usados para a implementação de Funções Lógicas.

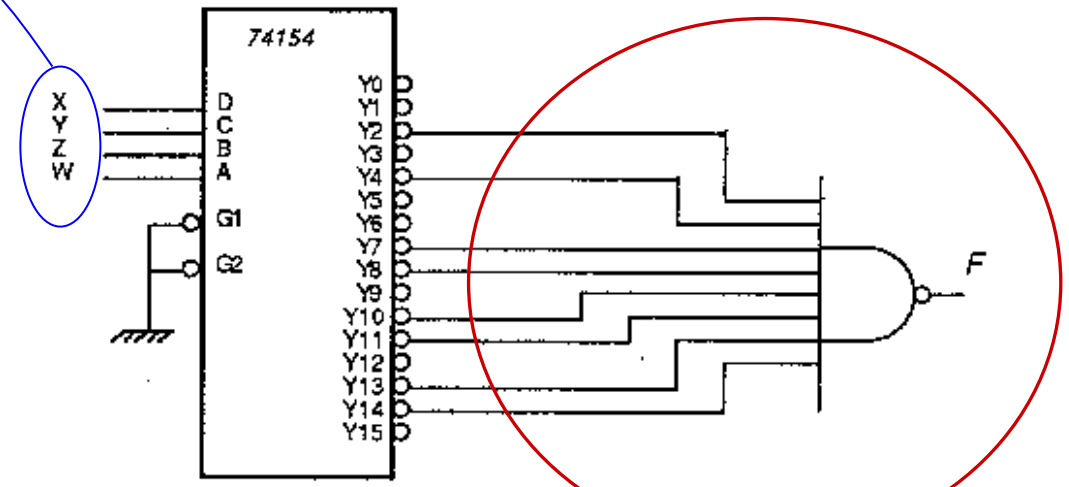
Com efeito, verifica-se que cada saída de um **Descodificador** corresponde a um termo da função de **n** variáveis que ele descodifica.

Por este motivo, para implementar uma função à custa de um Descodificador, basta fazer o **OR** dos termos que compõem a função a implementar.

Consegue-se assim implementar a função a partir da **Forma Canónica Soma de Produtos**.

Exemplo

X	Y	Z	W	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

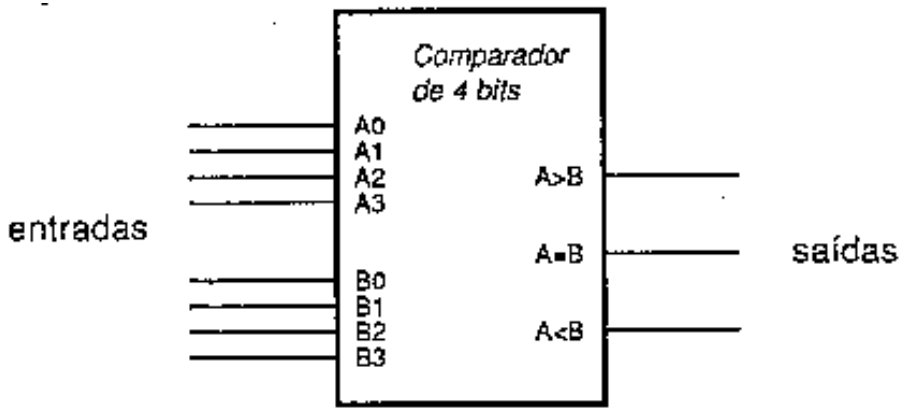


Como no 74154 as saídas são o complemento dos termos, em vez do **OR** usa-se um **NAND** (que é um OR com as entradas negadas). Ligam-se ao NAND os termos que dão valor **1** à função F.

Comparadores

Os **Comparadores** permitem determinar se dois números binários são iguais e, se não forem, indicam qual deles é maior.

- Circuito comercial **7485** (compara dois números binários de 4 *bits*)



  - A3, B3 = *Most Significant bit* (MSb)
  - A0, B0 = *Less Significant bit* (LSb)

Algoritmo utilizado para a comparação:

Se A3>B3 então A>B

Senão Se A3<B3 então A<B

Senão

Se A2>B2 então A>B

Senão Se A2<B2 então A<B

Senão

...

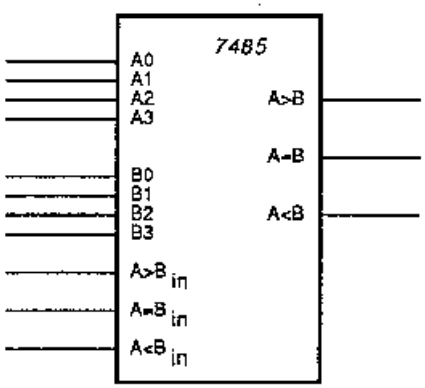
Senão A=B

Entradas de comparação				Saídas		
A3,B3	A2,B2	A1,B1	A0,B0	A > B	A < B	A = B
A3 > B3	X	X	X	1	0	0
A3 < B3	X	X	X	0	1	0
A3 = B3	A2 > B2	X	X	1	0	0
A3 = B3	A2 < B2	X	X	0	1	0
A3 = B3	A2 = B2	A1 > B1	X	1	0	0
A3 = B3	A2 = B2	A1 < B1	X	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 > B0	1	0	0
A3 = B3	A2 = B2	A1 = B1	A0 < B0	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	0	1

Como um “algoritmo” deste tipo pode ser estendido a **N** *bits*, os comparadores comerciais incluem 3 entradas que permitem ligá-los a um comparador “anterior”, em **cascata** (***cascading inputs***).

Podem assim comparar-se **palavras** de **Nx4** *bits* (**palavra** = grupo de *bits* que representa um determinado tipo de informação).

Comparação de palavras de **N**x4 *bits*:



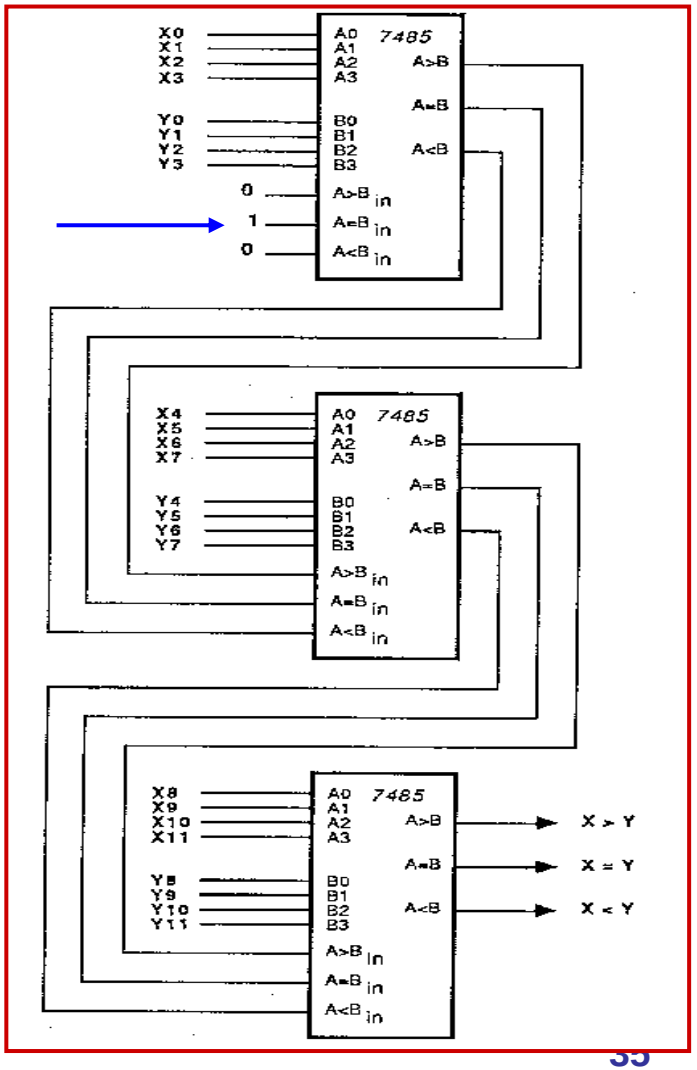
Entradas de expansão			Entradas de comparação				Saídas		
A>B	A<B	A=B	A3,B3	A2,B2	A1,B1	A0,B0	A>B	A<B	A=B
X	X	X	A3 > B3	X	X	X	1	0	0
X	X	X	A3 < B3	X	X	X	0	1	0
X	X	X	A3 = B3	A2 > B2	X	X	1	0	0
X	X	X	A3 = B3	A2 < B2	X	X	0	1	0
X	X	X	A3 = B3	A2 = B2	A1 > B1	X	1	0	0
X	X	X	A3 = B3	A2 = B2	A1 < B1	X	0	1	0
X	X	X	A3 = B3	A2 = B2	A1 = B1	A0 > B0	1	0	0
X	X	X	A3 = B3	A2 = B2	A1 = B1	A0 < B0	0	1	0
1	0	0	A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	0	0
0	1	0	A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	1	0
X	X	1	A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	0	1
1	1	0	A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	0	0
0	0	0	A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	1	0

Se do comparador anterior vem **A>B** e no actual **A=B** (**a3=b3**...**a0=b0**), então o resultado final é ainda **A>B**

Veja-se o seguinte exemplo, em que um **Comparador** de **12** *bits* é implementado à custa de três **Comparadores** de **4** *bits*:

No primeiro comparador (*bits* menos significativos) ligam-se as entradas "**>**", "**=**" e "**<**", a **0**, **1** e **0**, respectivamente, para activar a saída "**A=B**", no caso de **x3=y3**, **x2=y2**, **x1=y1** e **x0=y0**.

Entradas de expansão			Entradas de comparação				Saídas		
A>B	A<B	A=B	A3,B3	A2,B2	A1,B1	A0,B0	A>B	A<B	A=B
X	X	X	A3 > B3	X	X	X	1	0	0
X	X	X	A3 < B3	X	X	X	0	1	0
X	X	X	A3 = B3	A2 > B2	X	X	1	0	0
X	X	X	A3 = B3	A2 < B2	X	X	0	1	0
X	X	X	A3 = B3	A2 = B2	A1 > B1	X	1	0	0
X	X	X	A3 = B3	A2 = B2	A1 < B1	X	0	1	0
X	X	X	A3 = B3	A2 = B2	A1 = B1	A0 > B0	1	0	0
X	X	X	A3 = B3	A2 = B2	A1 = B1	A0 < B0	0	1	0
1	0	0	A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	0	0
0	1	0	A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	1	0
X	X	1	A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	0	1
1	1	0	A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	0	0
0	0	0	A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	1	0



Somadores

Os **Somadores** são circuitos que adicionam dois números que se encontram, habitualmente, em código binário natural.

- A saída é expressa no mesmo código
- Produzem um ***carry*** (transporte) ou ***overflow***, se o resultado não couber no número de *bits* das parcelas:

```
          1111
+         1111
-------------
Carry →  (1)1110
```

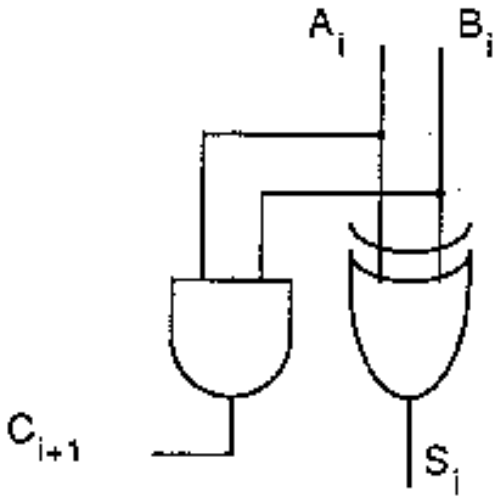
Semi-somador binário

Chama-se **Semi-somador binário**, ou ***Half-adder***, ao circuito seguinte que adiciona 2 *bits*, produzindo um *carry* de saída (ou *output carry*) $\mathbf{C_{out}}$:

A_i	B_i	S_i	C_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S_i = A_i \oplus B_i$$

$$C_{i+1} = A_i \cdot B_i$$



- Este circuito não suporta um *carry* de entrada (ou *input carry*).

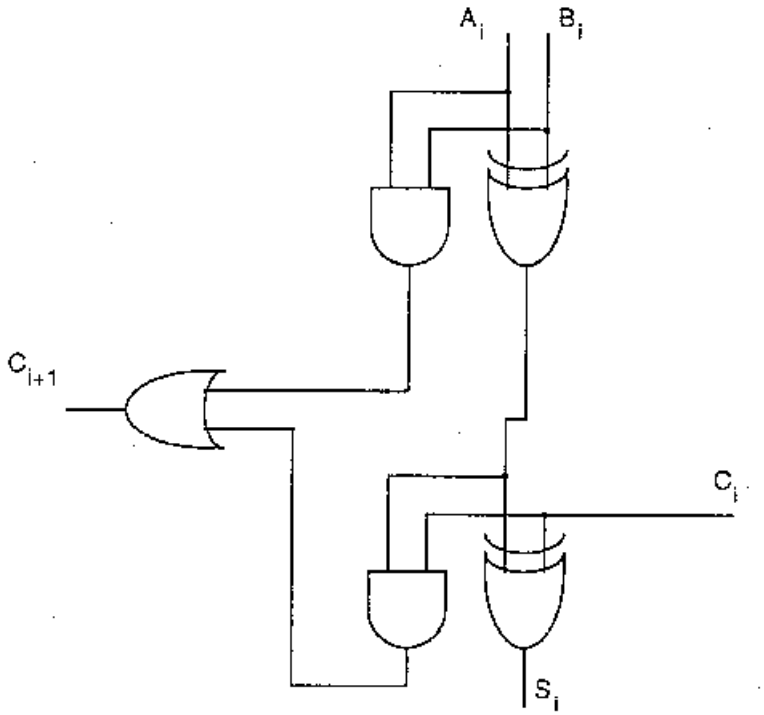
Somador binário completo

Chama-se **Somador binário completo**, ou ***Full-Adder***, ao seguinte circuito que adiciona 2 *bits*, produzindo um *carry* de saída (**C_{out}** ou **C_{i+1}**) e admitindo um *carry* de entrada (**C_{in}** ou **C_i**):

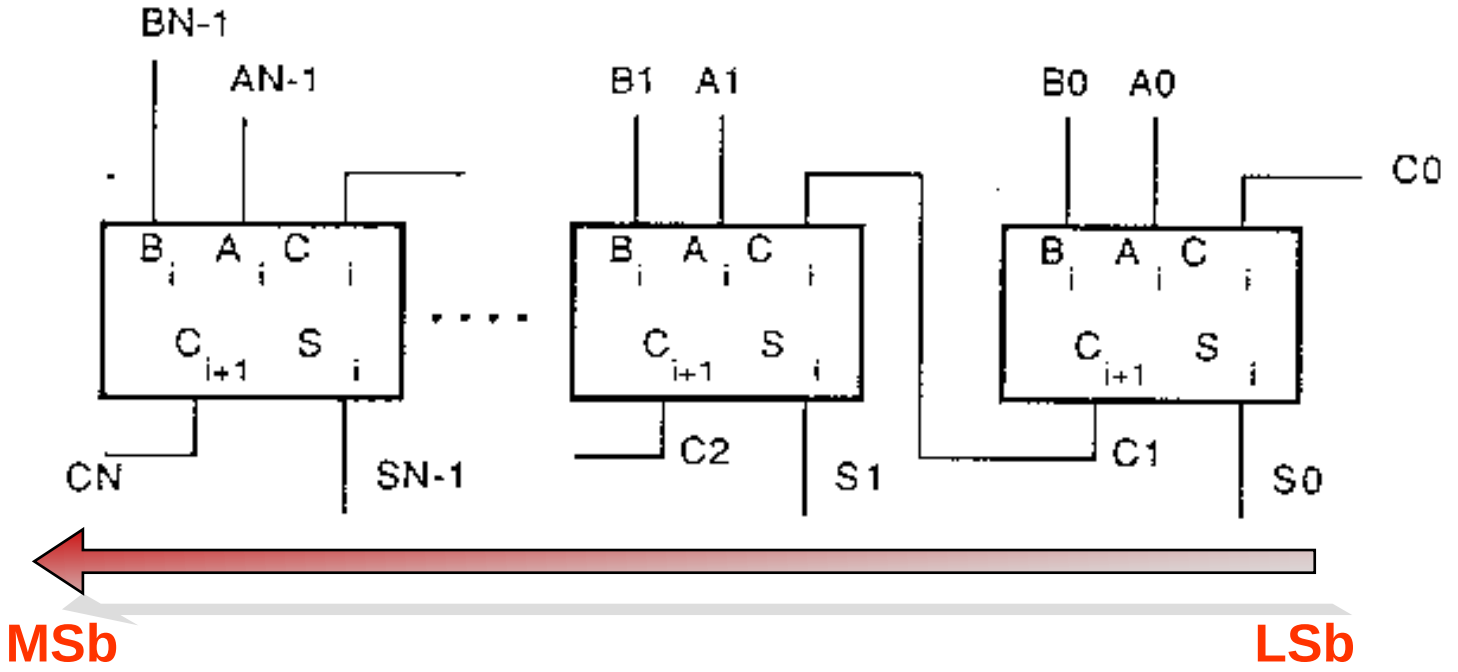
C_i	A_i	B_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = A_i \oplus B_i \oplus C_i$$

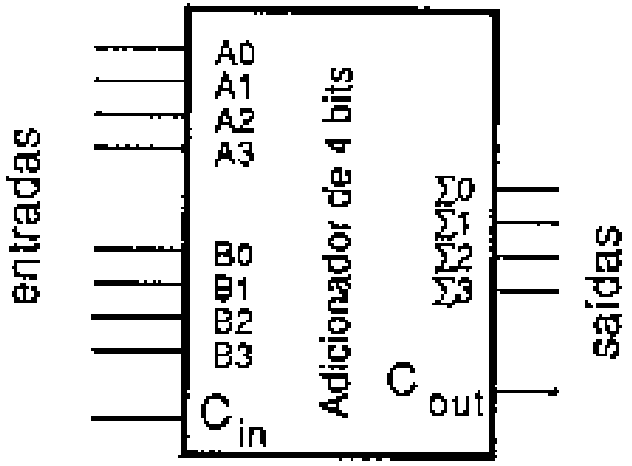
$$C_{i+1} = A_i . B_i + C_i . (A_i \oplus B_i)$$



Um **somador de N *bits*** pode então ser implementado da seguinte forma:



A família TTL disponibiliza **Somadores** de 4 *bits* sob a designação comercial de **7483**:



À semelhança dos **Comparadores**, podem ligar-se **Somadores** em cascata para realizarem somas com um maior nº de *bits*.

Para tal, o *carry* de saída (**C_{out}**) do somador anterior, liga-se ao *carry* de entrada (**C_{in}**) do somador seguinte.