Circuitos Combinacionales Básicos

Electrónica Digital — 2º GIERM (Mecatrónica) *Apuntes propios · enfoque worked-example para examen*

Mapa del tema

Un **circuito combinacional** es aquel cuya salida depende **solo** del valor actual de las entradas (no tiene memoria). Lo opuesto serán los circuitos secuenciales.

En este tema estudiamos los bloques combinacionales "estándar" que el fabricante vende como pastillas (ICs) y que reutilizamos como ladrillos:

Bloque	Qué hace	Idea de una línea
Multiplexor (MUX)	Selecciona 1 de 2^k entradas	"Llave selectora de datos"
Demultiplexor (DEMUX)	Reparte 1 entrada a 1 de 2^n salidas	El MUX al revés
Decodificador (DEC)	Activa 1 de 2^n salidas según el código de entrada	"Traductor binario → 1-de-N"
Codificador (COD)	Da el código binario de la entrada activa	El DEC al revés
Buffer triestado	Conecta/desconecta una salida del bus	"Interruptor electrónico"

A lo largo del notebook dibujamos los símbolos con `schemdraw` y verificamos las tablas de verdad con `numpy`.

```
Entorno listo:
  numpy    1.26.4
  schemdraw 0.22
```

1. Multiplexores (MUX)

Definición

Un multiplexor $2^k:1$ tiene:

- 2^k **entradas de datos** $i_0, i_1, \ldots, i_{2^k-1}$
- k **entradas de selección** $S_0, S_1, \ldots, S_{k-1}$
- **1 salida** z
- a menudo una entrada de habilitación **ST (Strobe) / E (Enable)**

La salida copia la entrada de datos cuyo índice (en binario) coincide con el valor de las líneas de selección:

$$z = i_{(S_{k-1}\ldots S_1 S_0)_2}$$

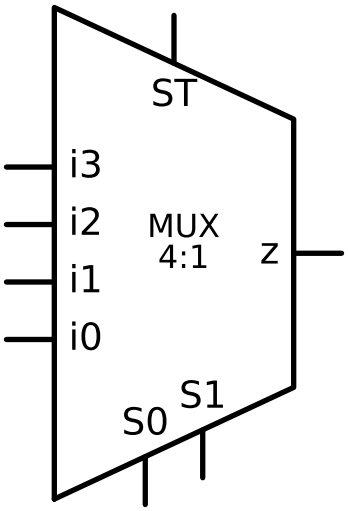
> **Regla mnemotécnica:** las líneas de selección forman un número binario; ese número es el "número de puerta" que se abre hacia la salida.

Ecuación del MUX 4:1

Para $k = 2$ (selección $S_1 S_0$), la salida en suma de productos es:

$$z = \overline{S_1}\,\overline{S_0}\,i_0 + \overline{S_1}\,S_0\,i_1 + S_1\,\overline{S_0}\,i_2 + S_1\,S_0\,i_3$$

S_1	S_0	z
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3



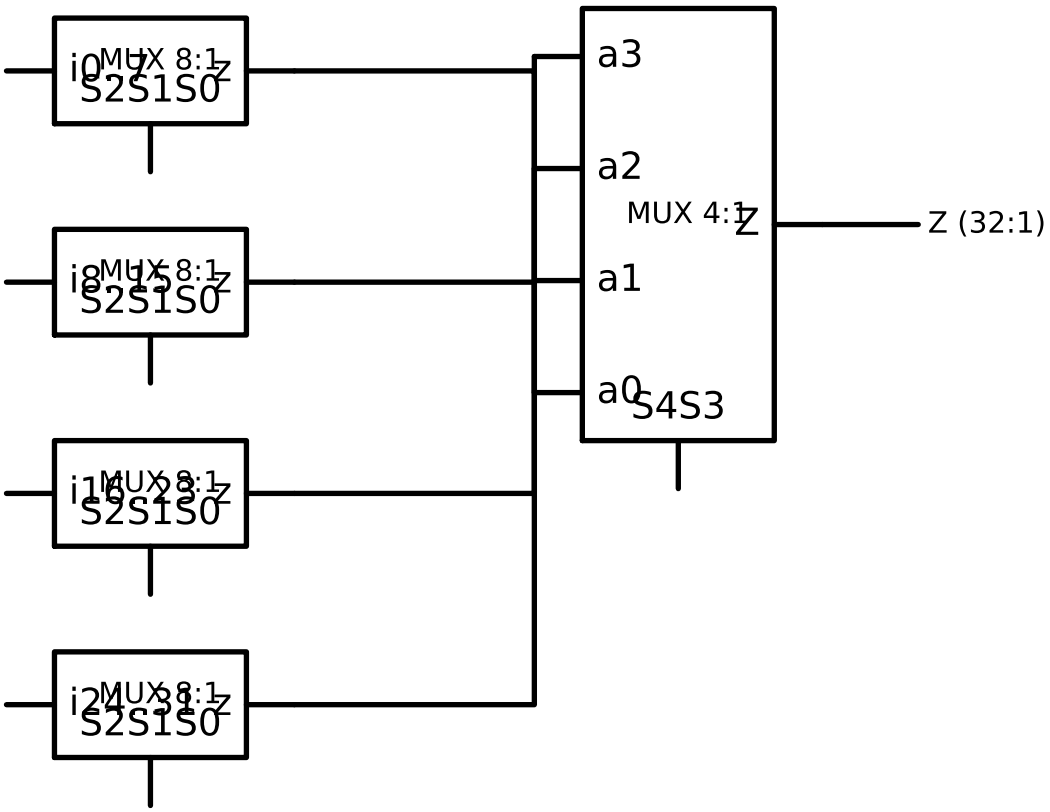

```
 S1 S0 |  z (modelo) | z (ecuacion)
-------+-------------+-------------
  0  0  |      0      |     0   OK
  0  1  |      1      |     1   OK
  1  0  |      1      |     1   OK
  1  1  |      0      |     0   OK
```

1.1 Extensión de multiplexores (MUX grande con MUXes pequeños)

Si solo tienes pastillas $8:1$ y necesitas un $32:1$, se construye en **árbol**:

- **Primer nivel:** cuatro MUX $8:1$ multiplexan los 32 datos en grupos de 8. Las 3 líneas bajas $S_2S_1S_0$ van **a los cuatro** a la vez.
- **Segundo nivel:** un MUX $4:1$ elige entre las 4 salidas, gobernado por S_4S_3.

$S_4S_3S_2S_1S_0$ → 5 bits → $2^5 = 32$ entradas. El **ST/Enable** permite además apilar pastillas.



1.2 Implementación de funciones con MUX (clave de examen)

Un MUX es un **generador universal de funciones**: conectas las variables a la selección y en cada i_m pones lo que vale la función en ese minterm.

Método directo (n variables → MUX $2^n:1$): las n variables a la selección; en cada i_m, un `0` o `1`.

Método económico (n variables → MUX $2^{n-1}:1$):

1. Usa $n-1$ variables en la selección; deja x "libre".
2. Agrupa la tabla por pares. En cada par la salida vale, según x: `0`, `1`, `x` o `x̄`, y eso se conecta a i_m.

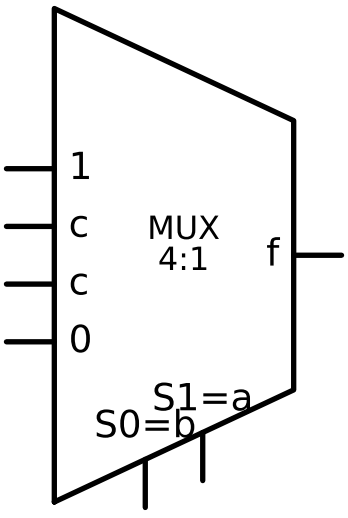
Ejemplo resuelto

Implementar $f(a,b,c) = \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c} + a\,b\,c$ con un **MUX 4:1**. Selección = a, b. Variable libre = c.

a	b	c	f	Dato	Valor
0	0	0	0	i_0	
0	0	1	0	i_0	0
0	1	0	0	i_1	
0	1	1	1	i_1	c
1	0	0	0	i_2	
1	0	1	1	i_2	c
1	1	0	1	i_3	
1	1	1	1	i_3	1

Resultado: $i_0 = 0,\ i_1 = c,\ i_2 = c,\ i_3 = 1$, con $S_1 = a,\ S_0 = b$.

```
 a b c | f_obj  f_mux
-------+-------------
 0 0 0 |   0     0
 0 0 1 |   0     0
 0 1 0 |   0     0
 0 1 1 |   1     1
 1 0 0 |   0     0
 1 0 1 |   1     1
 1 1 0 |   1     1
 1 1 1 |   1     1

Las dos columnas coinciden -> True
```



2. Demultiplexores (DEMUX)

El DEMUX es el **inverso** del MUX: **1 entrada de datos**, n líneas de selección y 2^n salidas. Encamina la entrada hacia la salida seleccionada; el resto quedan inactivas.

$$S_m = \text{dato si } (S_{n-1} \dots S_0)_2 = m, \qquad \text{resto} = 0$$

> **Dato clave:** un **decodificador con Enable** *es* un demultiplexor: el Enable hace de dato y las direcciones de selección.

```
DEMUX 1:4 con dato=1
 S1 S0 | S0 S1 S2 S3
-------+------------
  0  0  |  1  0  0  0
  0  1  |  0  1  0  0
  1  0  |  0  0  1  0
  1  1  |  0  0  0  1
```

3. Decodificadores (DEC)

Definición

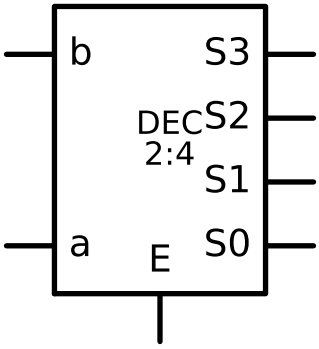
Un decodificador $n:2^n$ tiene n entradas y 2^n salidas, de las que **activa exactamente una**: la del índice igual al número de entrada. Cada salida es un **minterm**: $S_m = 1 \iff \text{entrada} = m$.

Decodificador 2:4

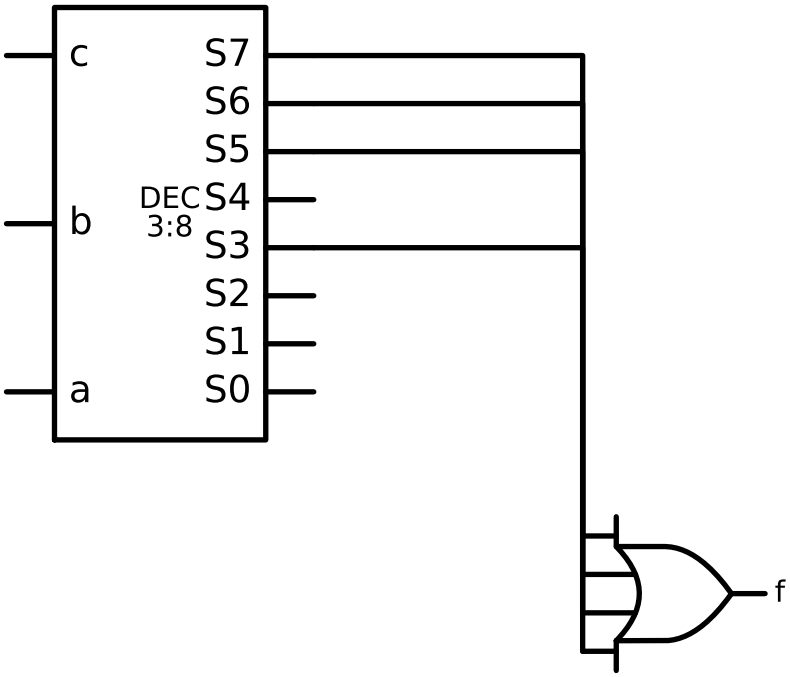
a	b	S_0	S_1	S_2	S_3	**Minterm**
0	0	1	0	0	0	$\overline{a}\,\overline{b}$
0	1	0	1	0	0	$\overline{a}\,b$
1	0	0	0	1	0	$a\,\overline{b}$
1	1	0	0	0	1	$a\,b$

Aplicaciones

1. **Selector / decodificador de direcciones** (mapas de memoria).
2. **Demultiplexor** (Enable como dato).
3. **Generación de funciones:** cada salida es un minterm → función = OR de las salidas que toca.
4. Parte de un **convertidor de código**.



a	b	S0	S1	S2	S3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1




```
Funcion con DEC+OR coincide con la suma de minterms -> True
```

4. Codificadores (COD)

Definición

Es el **inverso del decodificador**: 2^n entradas, n salidas. Suponiendo **una sola entrada activa**, da el **código binario del índice** de esa entrada. Cada bit de salida es la OR de las entradas cuyo índice lo tiene a 1:

$$S_0 = E_1 + E_3 + E_5 + E_7, \quad S_1 = E_2 + E_3 + E_6 + E_7, \quad S_2 = E_4 + E_5 + E_6 + E_7$$

> **Limitación:** si se activan dos entradas a la vez, la salida es errónea → **codificador de prioridad**.

```
 Entrada activa | S2 S1 S0
----------------+---------
       E0       |  0  0  0
       E1       |  0  0  1
       E2       |  0  1  0
       E3       |  0  1  1
       E4       |  1  0  0
       E5       |  1  0  1
       E6       |  1  1  0
       E7       |  1  1  1

Ecuaciones OR del apunte verificadas -> True
```

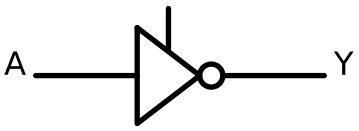
5. Buffers triestado (tri-state)

Una puerta normal solo da `0` o `1`. Un **buffer triestado** añade un tercer estado: **alta impedancia (Z)**, que *desconecta* eléctricamente la salida del nodo.

Enable E	**Entrada** A	**Salida** Y
1	0	0
1	1	1
0	X	**Z**

Para qué sirve: que varios dispositivos **compartan un bus**. En cada instante solo se habilita uno; los demás quedan en Z y no estorban. Conectar dos salidas activas al mismo cable sería un cortocircuito lógico.

> **Regla de oro del bus:** con N emisores triestado, **como mucho un Enable a 1** en cada instante.



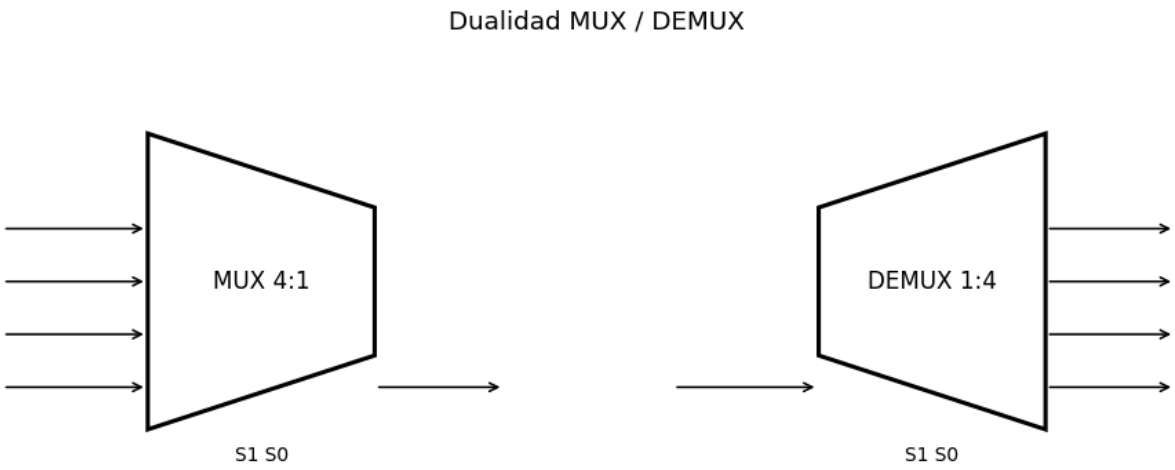

```
Tabla del buffer triestado:
  E A |  Y
  1 0 |  0
  1 1 |  1
  0 0 |  Z
  0 1 |  Z

Bus con 2 emisores [(A,E), (A,E)]:
  Emisor0 habilitado     -> 1
  Emisor1 habilitado     -> 1
  Ninguno habilitado     -> Z (bus flotante)
  Los DOS habilitados    -> CONFLICTO (cortocircuito)
```

6. Resumen y dualidades

```
MUX  <-->  DEMUX        (mismo bloque, sentido inverso del flujo)
DEC  <-->  COD          (uno expande 1->2^n minterms; el otro
comprime)
DEC con Enable  ==  DEMUX
MUX  =  generador universal de funciones (sel = variables, datos =
valores)
DEC + OR  =  generador de funciones (suma de minterms)
Triestado  =  desconexion controlada -> base de los buses compartidos
```

Checklist de examen

- ☐ MUX $2^k:1$ → k selecciones, salida $= i_{(\text{sel})_2}$.
- ☐ Función con MUX de **menos** entradas: deja una variable libre; en cada dato pon $0, 1, x, \bar{x}$.
- ☐ DEC: cada salida es un minterm → función con DEC + OR.
- ☐ DEC + Enable = DEMUX.
- ☐ Codificador: ojo a entradas múltiples → prioridad.
- ☐ Triestado: tercer estado **Z**; un solo Enable activo por bus.



Fuente: apuntes oficiales «ED4 — Circuitos Combinacionales Básicos», Electrónica Digital, Universidad de Sevilla. Notebook de estudio propio.