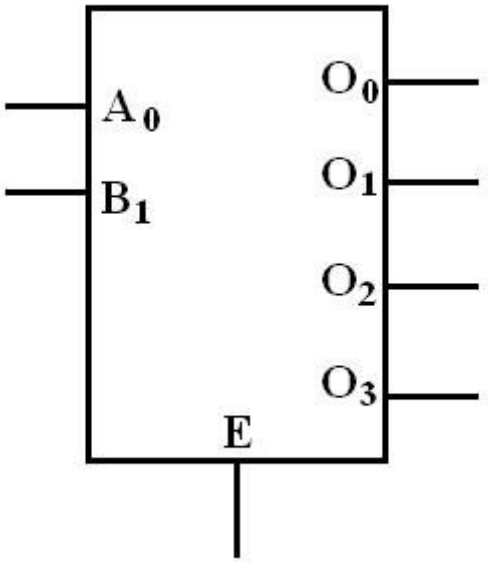
Decodificadores/Demultiplexores

Grupo 9
Javier de Gregorio Menezo
Laro de la Fuente Lastra
Raúl Fernández Díaz

Decodificadores

- Un **decodificador** (DEC) es un circuito combinacional que convierte un código binario de entrada A de N bits, en M líneas de salida Oi.
- M es el número de combinaciones del código de entrada. $M=2^N$

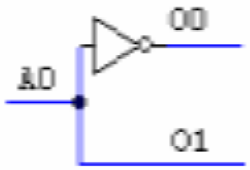
- Para cada dato binario de entrada Ai se fija una <u>única salida Oi a 1</u>, cuyo índice "i" corresponde al valor binario del dato de entrada.



Decodificadores

- Un decodificador 1 a 2 puede realizarse únicamente con un sólo inversor:

A0	O0	O1
0	1	0
1	0	1

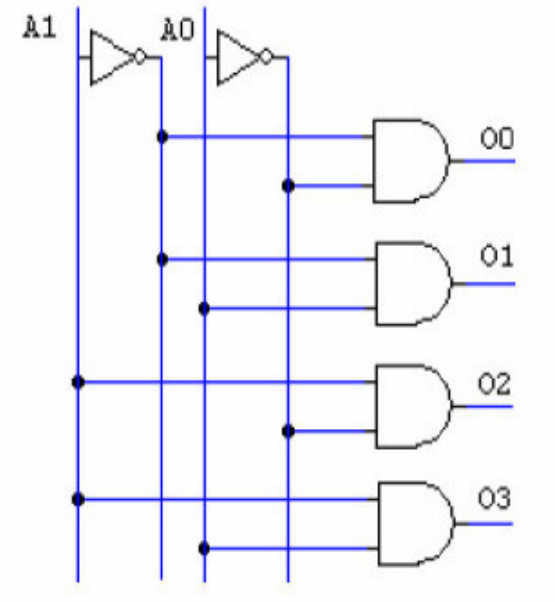


- Con una única entrada podemos obtener 2 salidas colocando un inversor en una de ellas.

Decodificadores

- Tabla de verdad:

A_1	A_0	O_0	O_1	O_2	O_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



- Nomenclatura:

  Ejemplo: **DEC 2 a 4** → (Decodificador de 2 entradas y 4 salidas)

Decodificadores

- Ejemplo de circuito comercial de la familia 74:
  74138 (Decodificador 3 a 8)

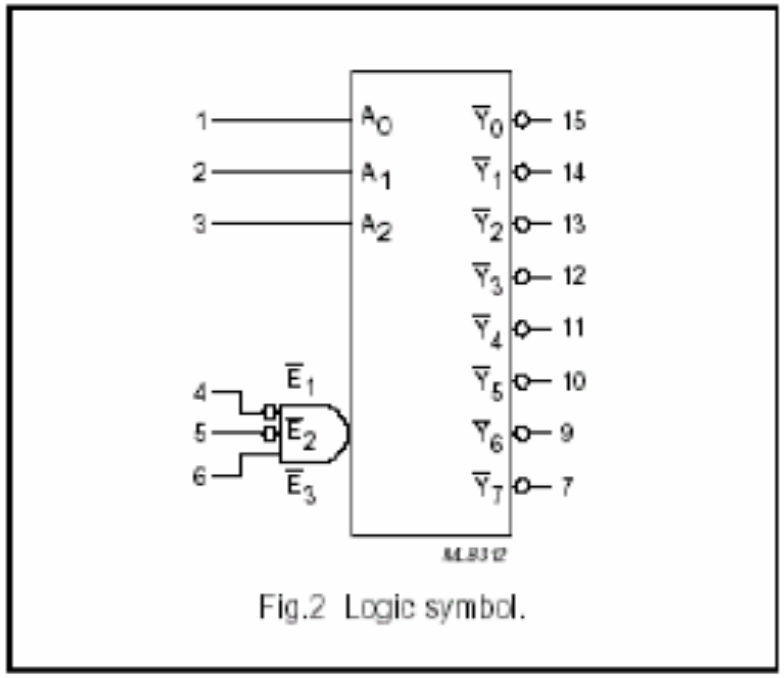


Fig.2 Logic symbol.

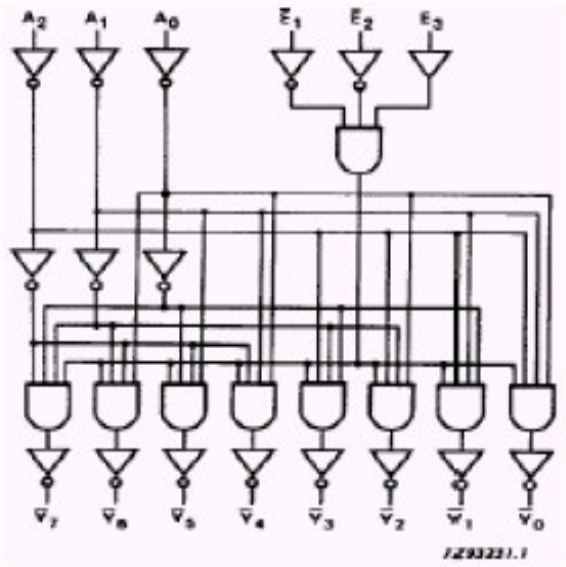


Fig.5 Logic diagram.

Decodificadores

- Tabla de verdad:

FUNCTION TABLE

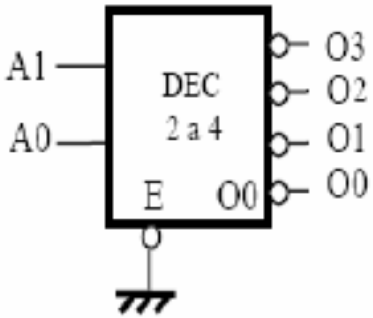
INPUTS						OUTPUTS							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

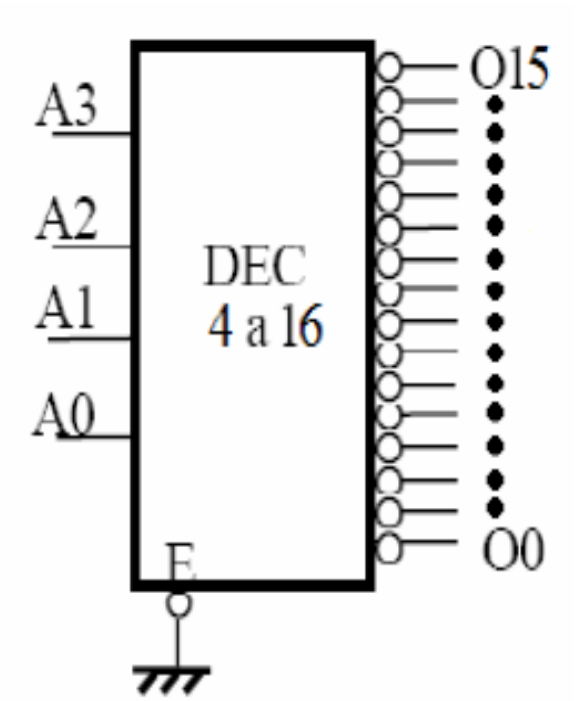
Notes

1. H = HIGH voltage level
   L = LOW voltage level
   X = don't care

Decodificadores

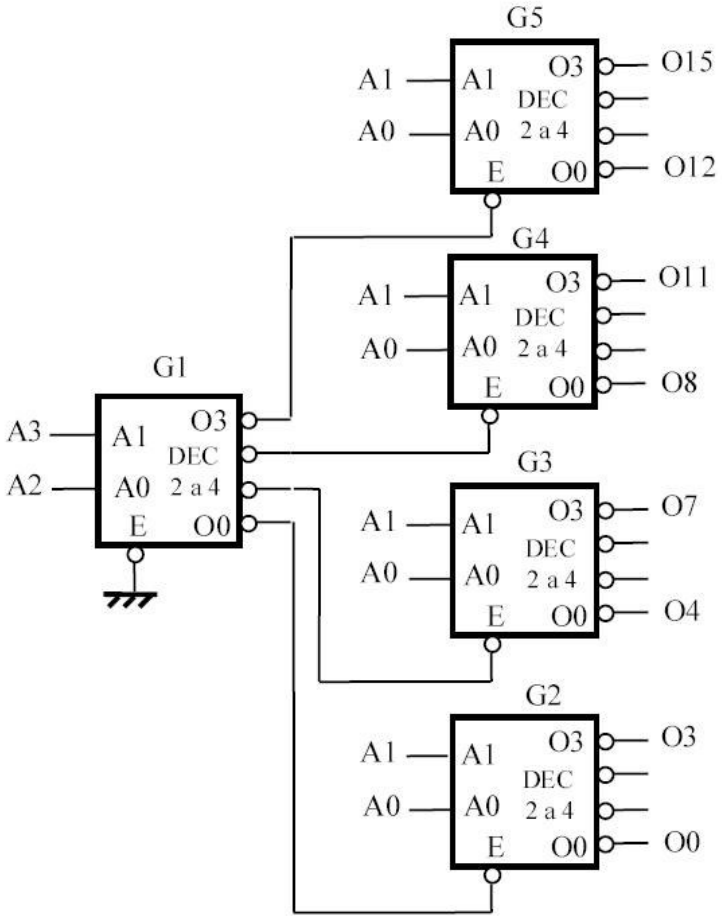
- Desarrollo de un decodificador 4 a 16 en base a un decodificador 2 a 4





Desarrollo de 4 a 16 DEC en base a 2 a 4 DEC:

- Si (A_3A_2) = 00, O_0 de G1 es 1 (L) y G2 está habilitado y en él, en función de (A_1A_0) se obtienen las salidas O_3-O_0. G3, G4, G5 están deshabilitados: sus salidas son 0.
- Si (A_3A_2) = 01, O_1 de G1 es 1 (L) y G3 está habilitado y en él, en función de (A_1A_0) se obtienen las salidas O_7- O_4. G1, G4, G5 están deshabilitados: sus salidas son 0.
- Si (A_3A_2) = 10, O_2 de G1 es 1 (L) y G4 está habilitado y en él en función de (A_1A_0) se obtienen las salidas O_{11}-O_8. G1, G3, G5 están deshabilitados: sus salidas son 0.
- Si (A_3A_2) = 11, O_3 de G1 es 1 (L) y G5 está habilitado y en él en función de (A_1A_0) se obtienen las salidas O_{15}-O_{12}. G2, G3, G4 están deshabilitados: sus salidas son 0.



Decodificadores

- Modelo **VHDL** de un decodificador 2 a 4:

```
library ieee;
use.ieee.std_logic_1164.all;

entity dec2to4 is
port (A: in std_logic_vector(1 downto 0);        ---  Entradas de dirección
        E: in std_logic;                          ---  Entrada de habilitación
        O: out std_logic_vector(3 downto 0));  ---  Salidas
end dec2to4;

architecture DEC of dec2to4 is
begin
process (A, E)
      begin
      if E = '0' then
        O <= "0000";
      else
          case A is
                when "00" => O <= "0001";
                when "01" => O <= "0010";
                when "10" => O <= "0100";
                when "11" => O <= "1000";
                when others => O <= "0000";
          end case;
      end if;
end process;
end DEC;
```

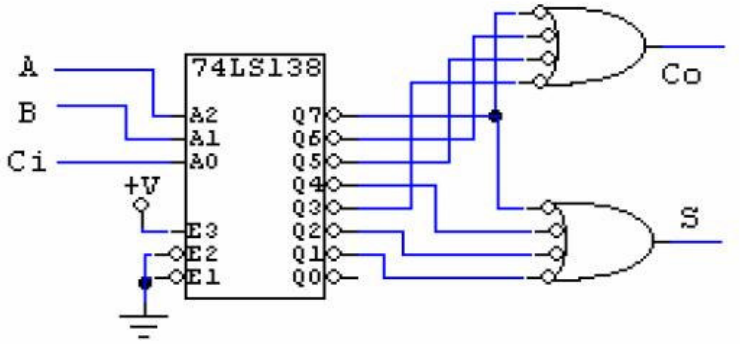
Decodificadores

- Los decodificadores permiten **implementar funciones lógicas** desde sus formas canónicas. Aplicando las entradas a las entradas de dirección del decodificador, cada una de las salidas del decodificador corresponde a cada uno de los minterms de la función lógica: O_0 es el minterm 0 (m0), O_1 el minterm 1 (m1), etc. Se puede hacer una forma canónica SOP mediante la OR de los minterms o 1s de la función lógica, es decir mediante el OR de las salidas O_i correspondientes a los minterms de la función.
- En principio hay que usar un decodificador de N a 2^N, siendo N el número de entradas de la función lógica.

Sumador completo

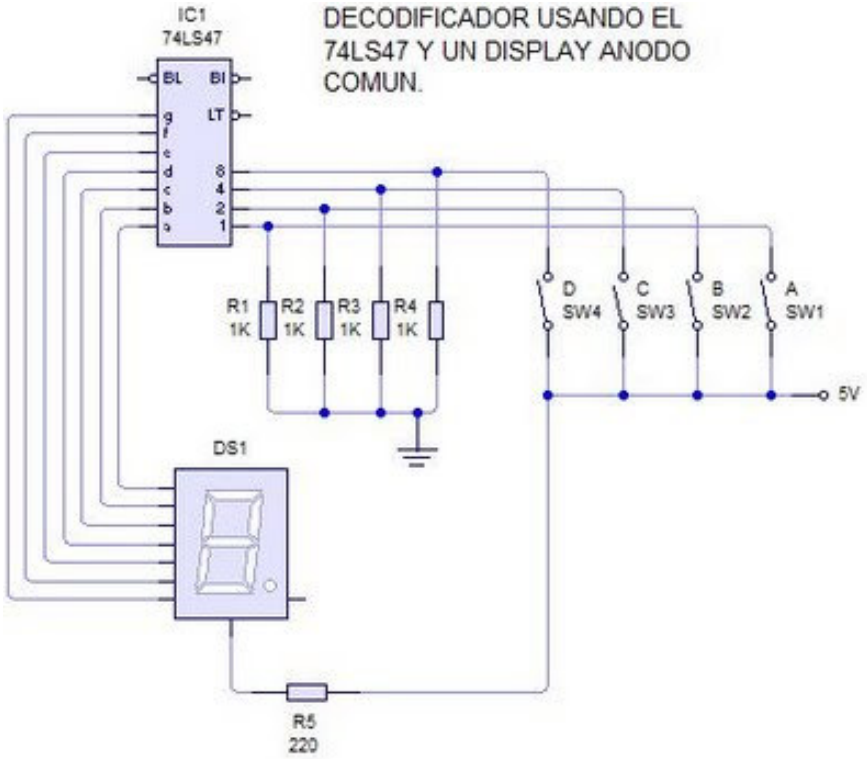
A	B	Ci	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

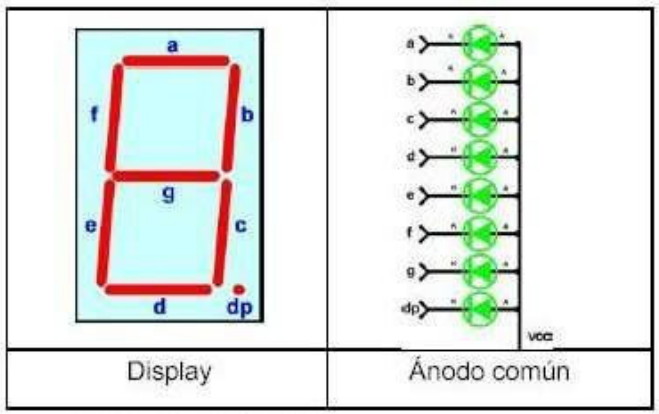
$S(A, B, Ci) = \sum(1, 2, 4, 7)$
$Co(A, B, Ci) = \sum(3, 5, 6, 7)$



Decodificadores

- Un ejemplo de aplicación es el decodificador BCD a 7 segmentos. Este tipo de decodificador acepta código BCD en sus entradas y proporciona salidas capaces de excitar un display de 7 segmentos para indicar un dígito decimal.





Decodificadores

- Decodificador BCD a 7 segmentos.
- Tabla de verdad.

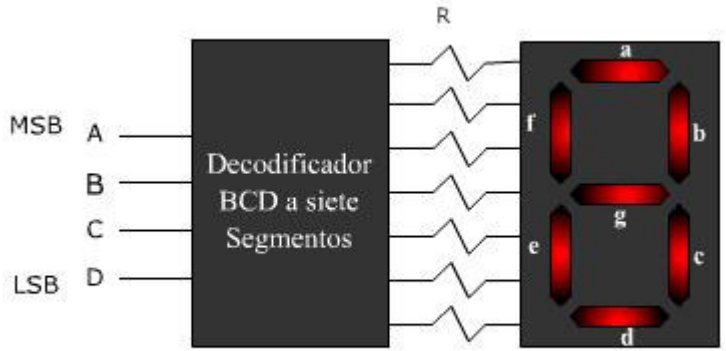


Figura 3.3.2. Diagrama de bloques de un decodificador BCD a siete segmentos

Valor decimal	Entradas				Salidas						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
...	..	..	..	..	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Tabla 3.3.1. Tabla de verdad del decodificador BCD a siete segmentos.

Demultiplexores

- Un **demultiplexor** (DEMUX) es un circuito combinacional que tiene una entrada de información de datos I y N entradas de control que sirven para seleccionar una de las M salidas, por la que ha de salir el dato que presente en la entrada.

  Siendo $M = 2^N$
- Esto se consigue aplicando a las entradas de control la combinación binaria correspondiente a la salida que se desea seleccionar.

- Los demultiplexores y los decodificadores son similares y en la práctica se usa un mismo circuito para realizar las dos funciones: decodificación de N a M y demultiplexado 1 de M.

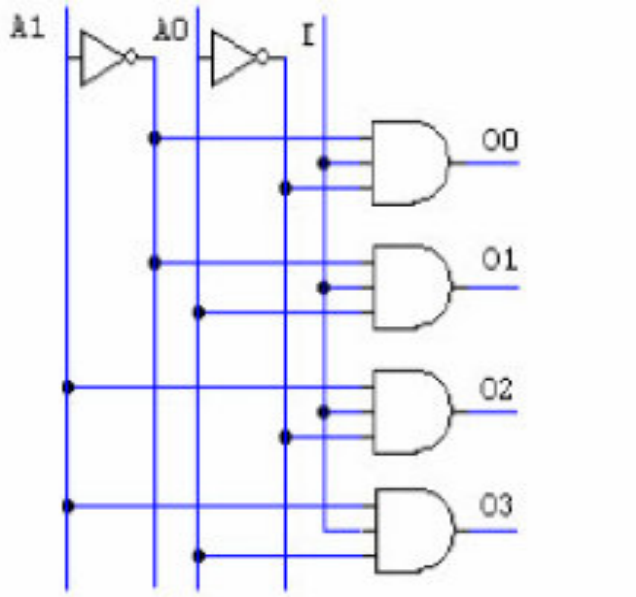
Demultiplexores

- Demultiplexor 1 a 4:

Tabla de verdad:

A_1	A_0	O_0	O_1	O_2	O_3
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

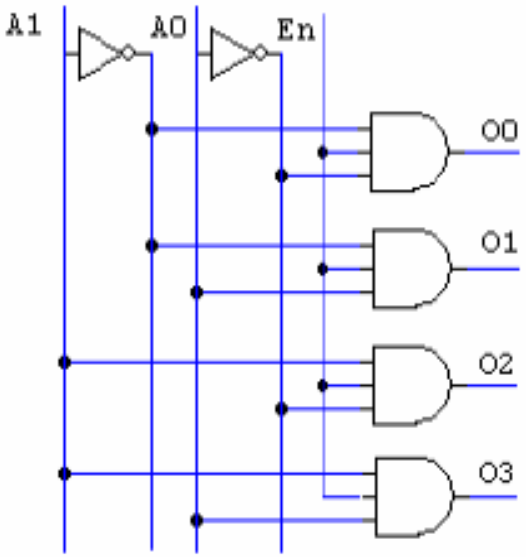
Esquema lógico:



Demultiplexores

- El **demultiplexor**, es un circuito combinacional que, aunque la función básica es la que hemos explicado, puede utilizarse en muchos casos como decodificador y adopta cualquiera de las funciones que un decodificador realiza.
- Como vemos, el demultiplexor y el decodificador con Enable se realizan con el mismo circuito.

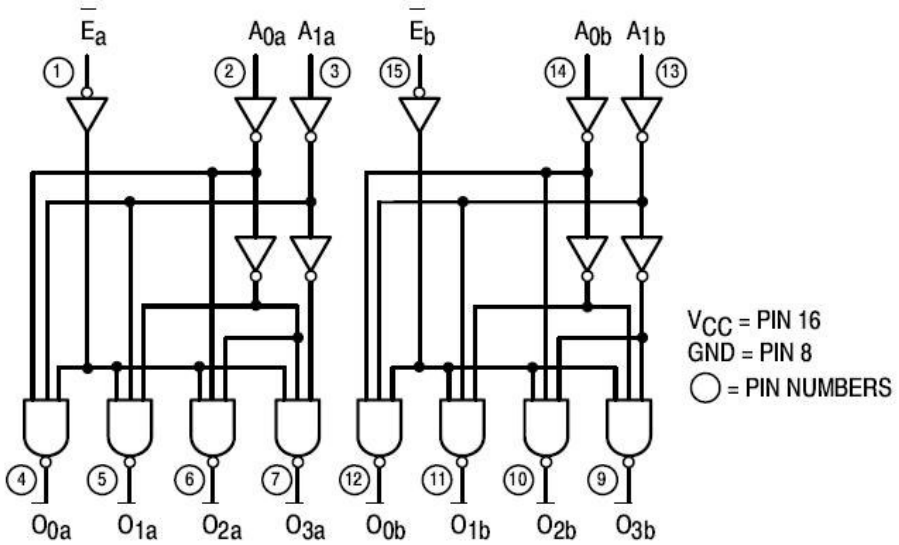
E	A_1	A_0	O_0	O_1	O_2	O_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



Demultiplexores

- Ejemplo de circuito comercial de la familia 74:
  74139 (Demultiplexor 1 a 4)

LOGIC DIAGRAM



TRUTH TABLE

INPUTS			OUTPUTS			
$\overline{E}$	A_0	A_1	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

Demultiplexores

- Modelo **VHDL** de un demultiplexor 1 a 4:

```
library ieee;
use.ieee.std_logic_1164.all;

entity dec2to4 is
port (A: in std_logic_vector(1 downto 0);        ---  Entradas de dirección
        E: in std_logic;                          ---  Entrada de habilitación
        O: out std_logic_vector(3 downto 0));  ---  Salidas
end dec2to4;

architecture DEMUX of dec2to4 is
begin
process (A, E)
      begin
      case A is
            when “00” => O(0) <= E; O(1) <= ‘0’;
                         O(2) <= ‘0’; O(3) <= ‘0’;
            when “01” => O(0) <= ‘0’; O(1) <= E;
                         O(2) <= ‘0’; O(3) <= ‘0’;
            when “10” => O(0) <= ‘0’; O(1) <= ‘0’;
                         O(2) <= E; O(3) <= ‘0’;
            when “11” => O(0) <= ‘0’; O(1) <= ‘0’;
                         O(2) <= ‘0’; O(3) <= E;
            when others => O(0) <= ‘0’; O(1) <= ‘0’;
                           O(2) <= ‘0’; O(3) <= ‘0’;
      end case;
end process;
end DEMUX;
```

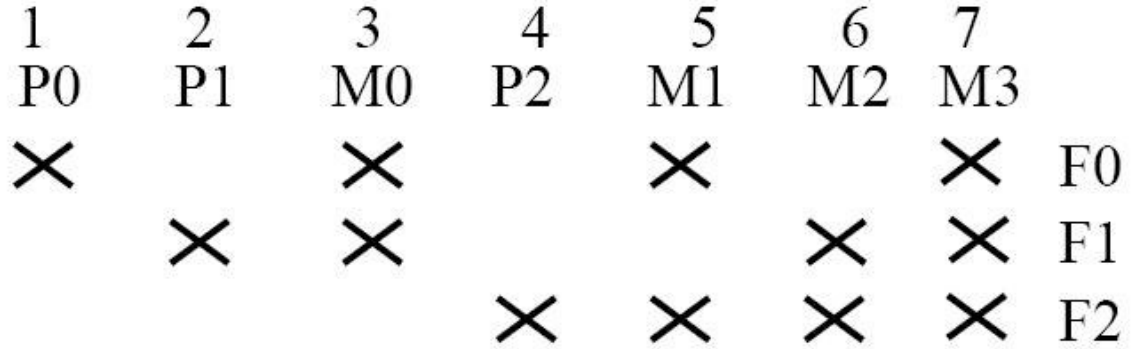
Problemas propuestos

- Diseñar un multiplexor de 16 entradas utilizando 4 multiplexores triestado de 4 entradas con habilitador (el circuito deshabilitado queda en alta impedancia) y un decodificador 2 a 4. Indicar como debería diseñarse el circuito con dos chips 74’153, un chip 74’139 y una puerta lógica.

- Se quiere diseñar un decodificador de 40 direcciones de 0 a 39 utilizando decodificadores binarios (2 a 4, 3 a 8, 4 a 16, etc). Indicar cuál es el número mínimo de decodificadores binarios que hay que utilizar y realizar el diseño del decodificador utilizando los decodificadores binarios y las puertas lógicas que sean necesarias (un inversor).

Problemas propuestos

- Diseñar un circuito decodificador del código de Hamming capaz de recuperar un error simple en un código (M0M1M2M3) con bits de paridad (P0P1P2) par utilizando puertas EXOR (para determinar F2F1F0 la dirección del bit erróneo, y para complementar dicho bit) y un 3 a 8 DEC (para indicar el bit erróneo en función de F2F1F0)



Bibliografía

- Apuntes de la asignatura (Electrónica Digital I)

- Datasheetcatalog.com
  http://www.datasheetcatalog.org/datasheet/motorola/SN54LS139J.pdf

- Wikipedia:
  - http://es.wikipedia.org/wiki/Decodificador
  - http://es.wikipedia.org/wiki/Demultiplexor

- http://www.cps.unizar.es/~fbeltran/sist_comb.pdf

Trabajo realizado por cada miembro del grupo:

- Javier → 40%
- Raúl → 30%
- Laro → 30%