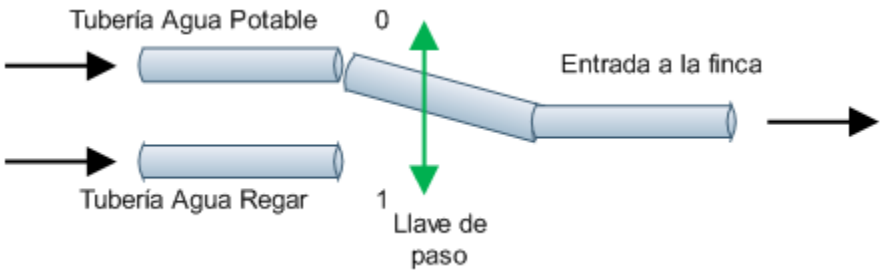
CIRCUITOS MULTIPLEXORES Y DEMULTIPLEXORES

MULTIPLEXOR (MUX)

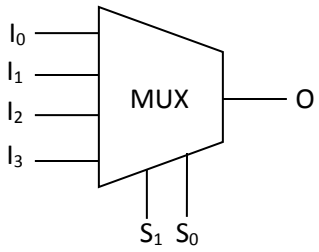
Un Multiplexor (MUX) es un circuito combinacional al que entran varios canales de datos, y sólo salen los datos del que hayamos seleccionado. Es decir, que es un circuito que nos permite SELECCIONAR que datos pasan a través de dicho componente. Es la versión Electrónica de un conmutador rotatorio o llave selectora.

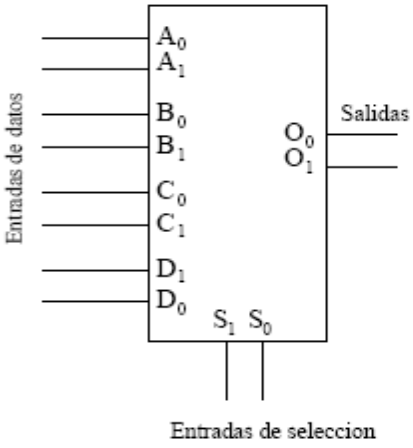
Supongamos, que hay dos tuberías (canales de datos) por el que circulan distintos fluidos (datos). Una tubería transporta agua para regar y la otra agua potable. Estas tuberías llegan a una finca, en la cual hay una sola manguera por la que va a salir el agua (la potable o la para regar), según lo que se seleccione en la llave de paso, la posición "0" es para el agua potable y "1" para regar.

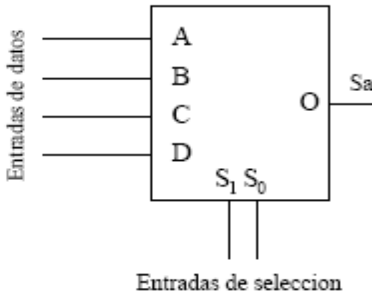


Con este ejemplo es muy fácil entender la idea de multiplexor. Es como una llave de paso, que sólo conecta uno de los canales de datos de entrada con el canal de datos de salida.

Los Mux están compuestos por Entrada de Datos (las tuberías), Selector de Datos (llave de paso) y la única Salida.





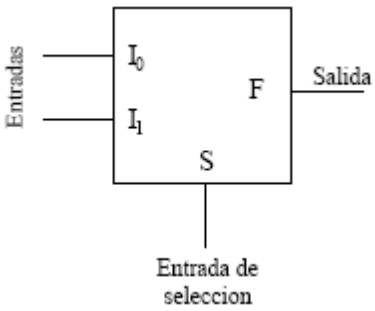


Ambos MUX tienen 4 canales de entrada de datos y para ello se necesitan 2 bit de Selector de datos ($2^2 = 4$, para poder seleccionar los 4 canales posibles). Sin

embargo, el del lado izquierdo tiene 2 bit de entrada por canal y 2 bit de salida, el de la derecha tiene 1 bit de entrada por cada canal y un bit de salida.

Mux con 1 Entrada de Selección

Permite seleccionar entre dos datos de entrada (S=0 y S=1)

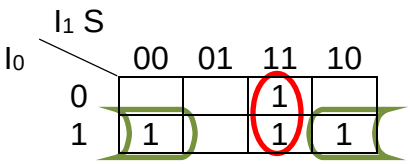


Construyamos la tabla de verdad:

- Son 3 entradas (I_0, I_1, S), $2^3 = 8$ combinaciones
- Si S = "0" entonces F = I_0
- Si S = "1" entonces F = I_1

S	I_1	I_0	F	
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	1	S = 0 → F = I_0
1	0	0	0	
1	0	1	0	
1	1	0	1	
1	1	1	1	S = 1 → F = I_1

Por mapas de Karnaugh:



$$F = (\bar{S} \times I_0) + (S \times I_1)$$

Reemplazando en la Función:

- Si $S = 0 \rightarrow F = (1 \times I_0) + (0 \times I_1) \therefore F = I_0$
- Si $S = 1 \rightarrow F = (0 \times I_0) + (1 \times I_1) \therefore F = I_1$

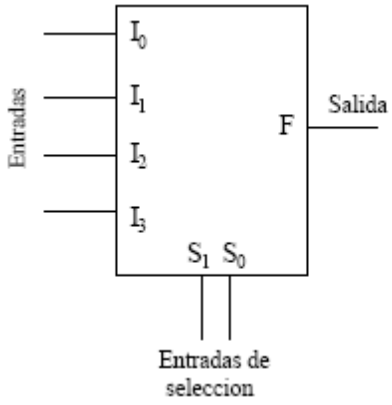
La salida toma el valor de una de las entradas, depende del valor que tome la entrada de selección.

La función F que describe el comportamiento de un multiplexor con una entrada de selección, está descrita en la siguiente tabla:

S	F
0	I_0
1	I_1

Mux con 2 Entradas de Selección

Como tiene 2 entradas de selección tiene 4 posibles entradas de datos ($2^2 = 4$). En total son 6 entradas, realizar la tabla es algo dispendioso ya que $2^6 = 64$ posibles combinaciones y el mapa de Karnaugh sería de 6 variables, una forma más simple de describir este MUX sería mediante la siguiente tabla:



S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

La salida del MUX valdrá I_0, I_1, I_2, o I_3 según el valor de las variables de entrada S_0 y S_1.

$$F = (\overline{S_1} \times \overline{S_0} \times I_0) + (\overline{S_1} \times S_0 \times I_1) + (S_1 \times \overline{S_0} \times I_2) + (S_1 \times S_0 \times I_3)$$

Verifiquemos, si $S_1 = 0$ y $S_0 = 0$:

$$F = (\overline{S_1} \times \overline{S_0} \times I_0) + (\overline{S_1} \times S_0 \times I_1) + (S_1 \times \overline{S_0} \times I_2) + (S_1 \times S_0 \times I_3)$$
$$F = (\overline{0} \times \overline{0} \times I_0) + (\overline{0} \times 0 \times I_1) + (0 \times \overline{0} \times I_2) + (0 \times 0 \times I_3)$$
$$F = (1 \times 1 \times I_0) + (1 \times 0 \times I_1) + (0 \times 1 \times I_2) + (0 \times 0 \times I_3)$$
$$F = I_0$$

Mux con 3 Entradas de Selección

Como tiene 3 entradas de selección tiene 8 posibles entradas de datos ($2^3 = 8$). En total son 11 entradas, la tabla sería $2^{11} = 2048$ combinaciones posibles; entonces la forma simple de describir este MUX sería:

$$F = (\overline{S_2} \times \overline{S_1} \times \overline{S_0} \times I_0) + (\overline{S_2} \times \overline{S_1} \times S_0 \times I_1) + (\overline{S_2} \times S_1 \times \overline{S_0} \times I_2) + (\overline{S_2} \times S_1 \times S_0 \times I_3) + (S_2 \times \overline{S_1} \times \overline{S_0} \times I_4) + (S_2 \times \overline{S_1} \times S_0 \times I_5) + (S_2 \times S_1 \times \overline{S_0} \times I_6) + (S_2 \times S_1 \times S_0 \times I_7)$$

S_2	S_1	S_0	F
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

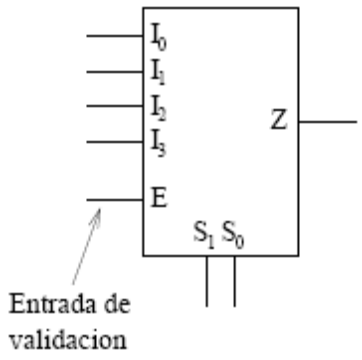
Mux con Entrada de Validación

Los Mux pueden disponer de una entrada adicional llamada de Validación (*Enable* = *E*). Esta entrada funciona como un interruptor. Si E = "1", el circuito funcionará normalmente; pero si E = "0" el circuito sacará el valor '0' por todas sus salidas, independiente de lo que llegue por sus entradas, se dice que está deshabilitado.

Las entradas de validación pueden ser de dos tipos: activas en nivel alto o activas en nivel bajo.

Entrada de validación activa en alto

Si E=1 el multiplexor funciona normalmente, si E=0 entonces su salida será '0' (estará desconectado).



E	S_1	S_0	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

Resumiendo la tabla queda:

Las "x" indica que cuando E=0, independiente de los valores que tengan las entradas S_0 y S_1 la salida siempre será "0".

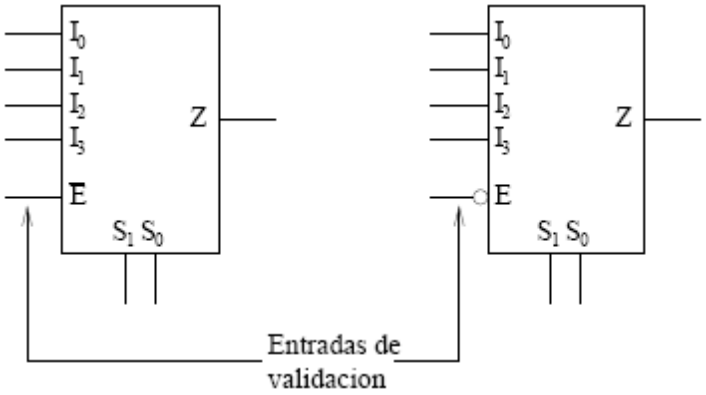
E	S_1	S_0	Z
0	**x**	**x**	**0**
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

Y la ecuación es:

$$Z = [(\overline{S_1} \times \overline{S_0} \times I_0) + (\overline{S_1} \times S_0 \times I_1) + (S_1 \times \overline{S_0} \times I_2) + (S_1 \times S_0 \times I_3)] \times E$$

Entrada de validación activa en bajo

Si E=0 el multiplexor funciona normalmente, si E=1 entonces su salida será '0' (estará desconectado).



E	S_1	S_0	Z
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	**x**	**x**	**0**

Y la ecuación es:

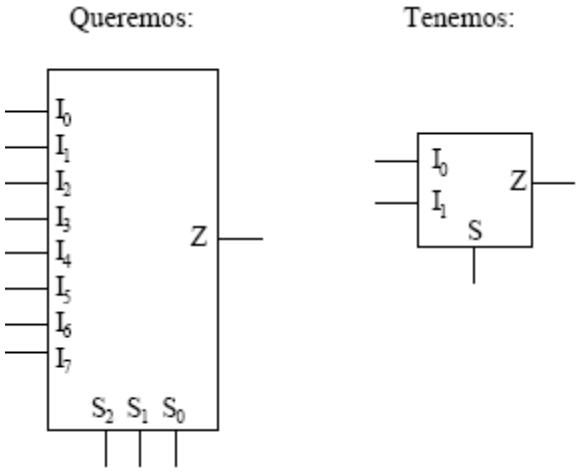
$$Z = [(\overline{S_1} \times \overline{S_0} \times I_0) + (\overline{S_1} \times S_0 \times I_1) + (S_1 \times \overline{S_0} \times I_2) + (S_1 \times S_0 \times I_3)] \times \overline{E}$$

Extensión de Multiplexores

Es poder obtener multiplexores más grandes a partir de otros más pequeños. La extensión puede ser aumentando el número de entradas o aumentando el número de bits por cada canal de datos.

Aumento del número de Entradas

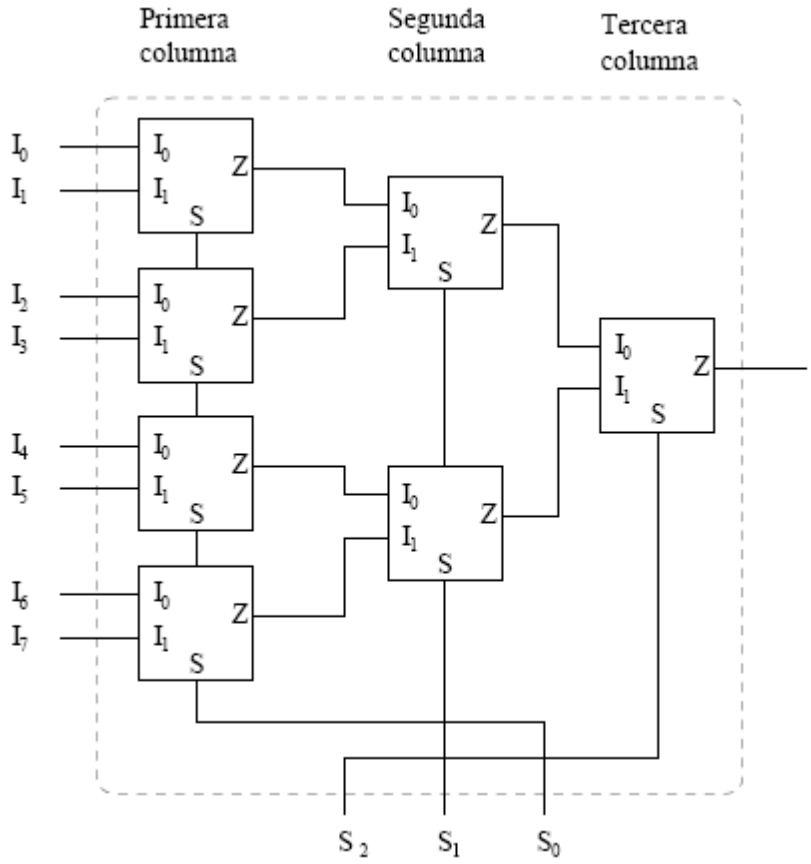
Necesitamos un multiplexor de 8 canales, pero sólo disponemos de varios Mux de 2 canales:



La solución es conectarlos en cascada. Primero colocamos una columna de 4 multiplexores de dos entradas, para tener en total 8 entradas. Todas las entradas de selección de esta primera columna se unen (la representamos mediante una línea vertical que une la salida S de un multiplexor con el de abajo).

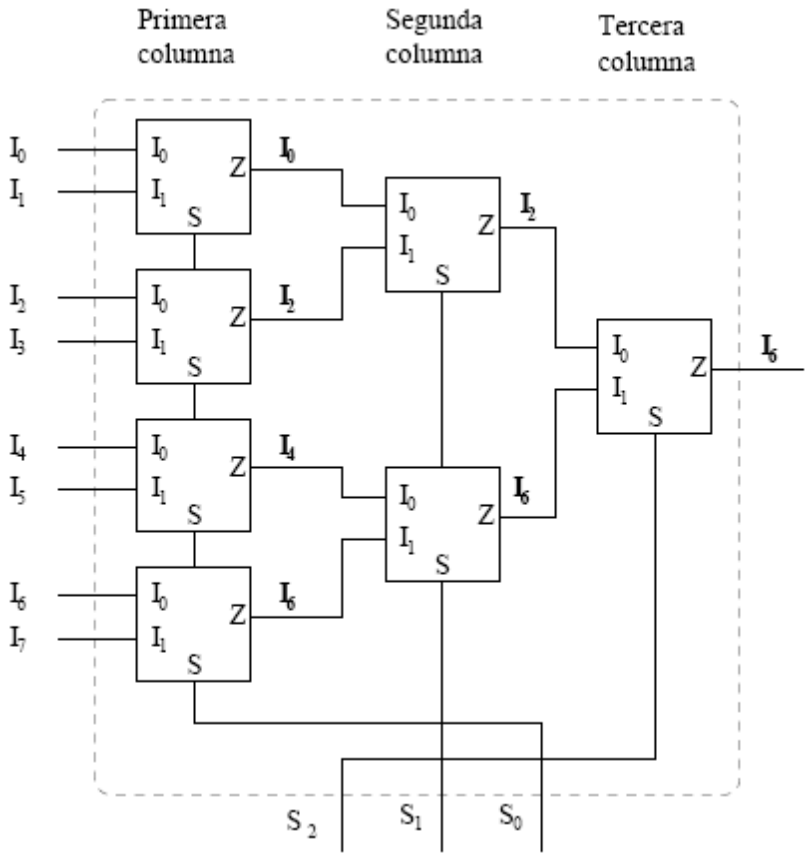
A continuación colocamos una segunda columna de 2 multiplexores de 2 entradas, también con sus entradas de selección unidas. Finalmente colocamos una última columna con un único multiplexor de 2 entradas.

La entrada de selección de los multiplexores de la primera columna tiene un peso de 0, la segunda un peso de 1 y la última un peso de 2.

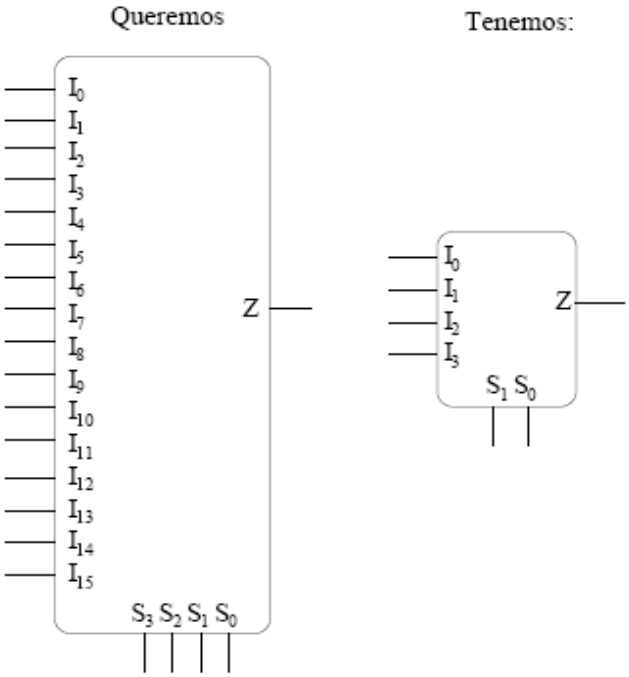


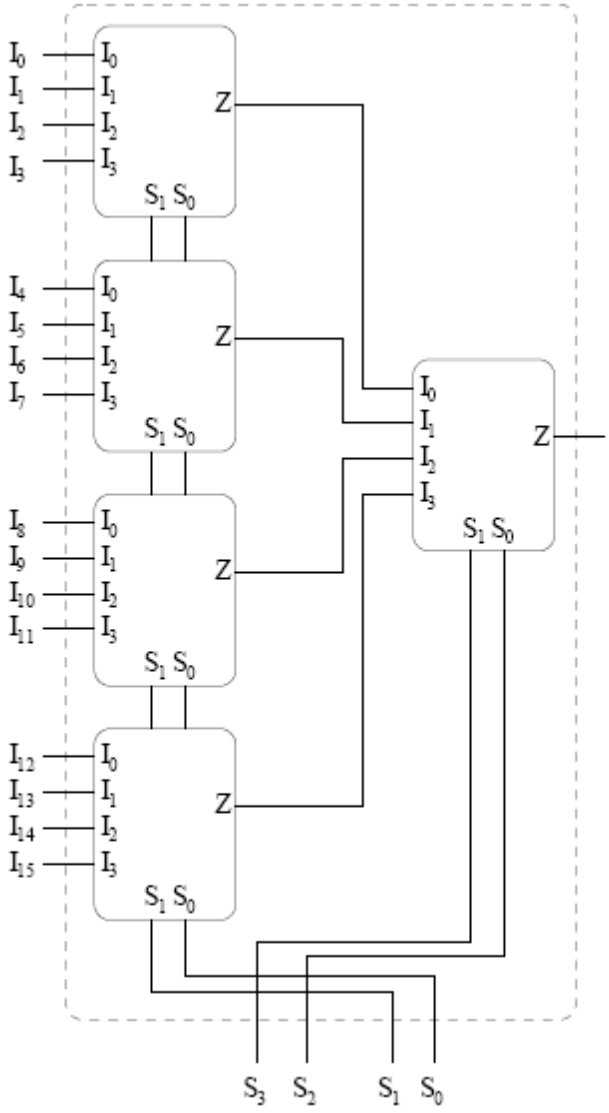
Por ejemplo: seleccionamos el canal 6, se introduce el número 6 en binario en las entradas de Selección (S_2=1, S_1=1 y S_0=0). Por los Mux de la primera columna se introduce “0” y sacan por sus salidas lo que hay en las entradas I_0, I_2, I_4, I_6.

Por la entrada de selección de los Mux de la segunda columna se introduce un “1” por lo que se están seleccionando las entradas de los canales I_1 y la salida de estos Mux serán I_2, I_6; y en la entrada del Mux de la tercera columna se ingresa un “1” lo que elige su entrada I_1 y la salida final es I_6.



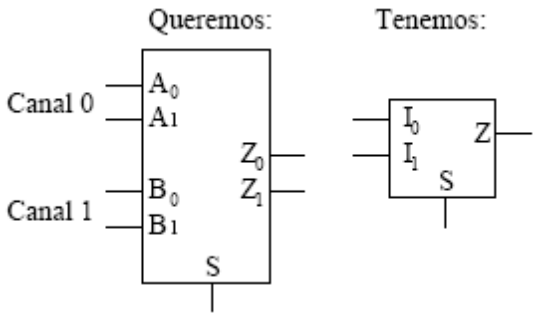
Por ejemplo: construir un multiplexor de 16 entradas usando multiplexores de 4.



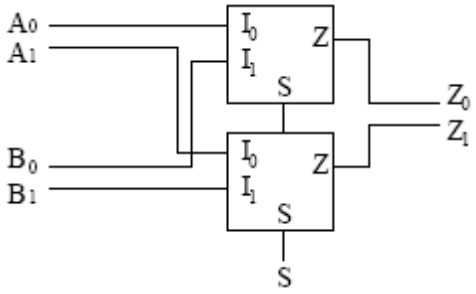


Aumento del número de bits por canal

Los Mux se conectan en paralelo. Se necesita construir un multiplexor de dos canales de entrada, cada uno de ellos de 2 bits, y para ello disponemos de multiplexores de 2 canales de un bit:

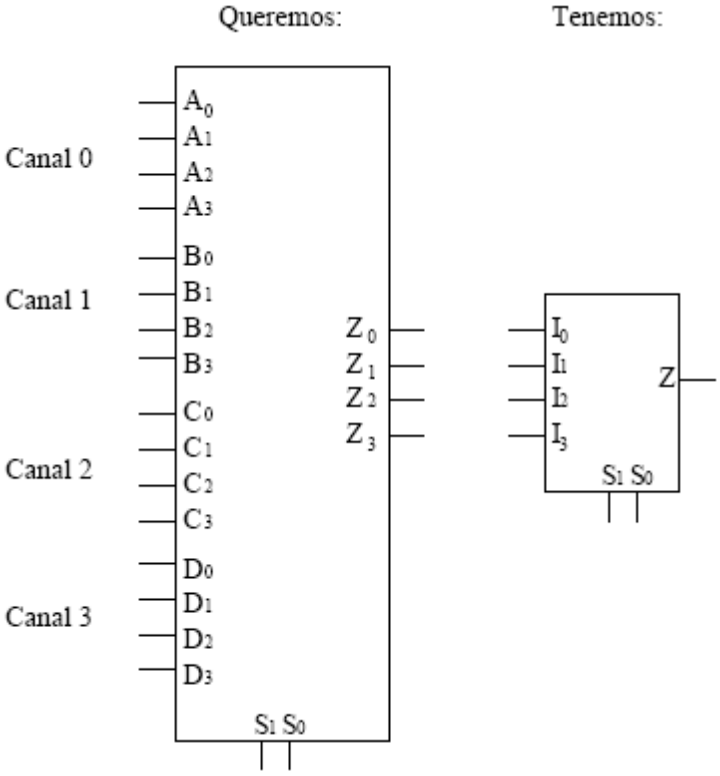


Utilizaremos dos multiplexores de los que tenemos, uno por cada bit de entrada (A_0 y B_0). Como los canales en el nuevo multiplexor son de 2 bits, necesitaremos 2 multiplexores (uno para cada bit). A un Mux van los bits de menor peso de los canales de entrada y al otro los de mayor peso. Las entradas de selección de ambos están unidas:

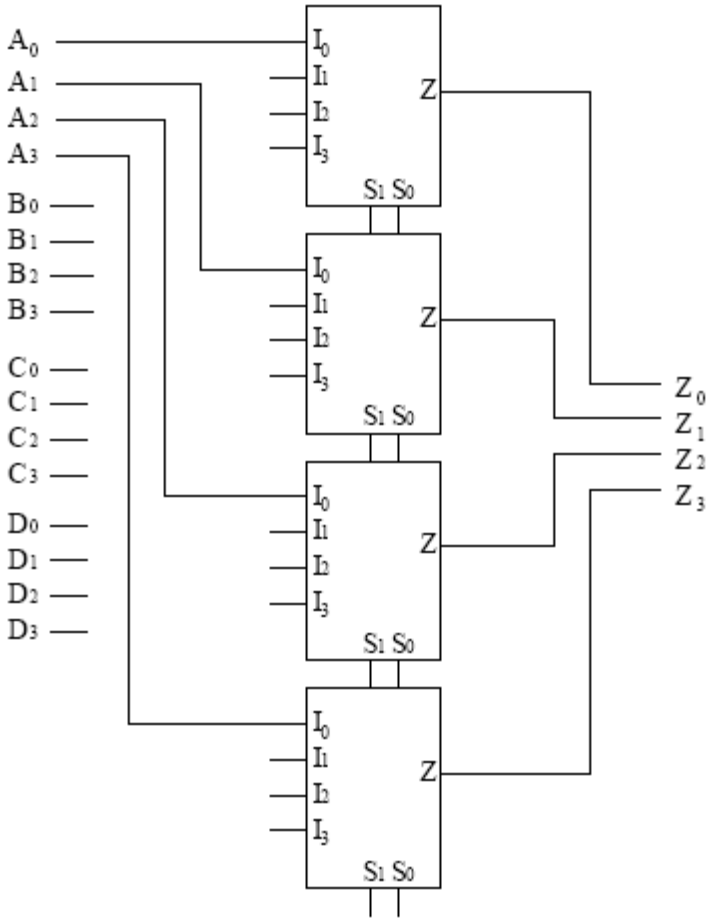


- Si tenemos S=0, las salidas son: $Z_0=A_0$ y $Z_1=A_1$
- Si tenemos S=1, las salidas son: $Z_0=B_0$ y $Z_1=B_1$

Ejemplo: construir un multiplexor de 4 canales de 4 bits, usando multiplexores de 4 entradas de 1 bit.



Necesitaremos 4 multiplexores de los que tenemos, a cada uno de los cuales les llegan los bits del mismo peso de los diferentes canales. Por el primer multiplexor entran los bits de menor peso (A_0, B_0, C_0 y D_0) y por el último los de mayor peso (A_3, B_3, C_3 y D_3). En el dibujo no se muestran todas las conexiones:

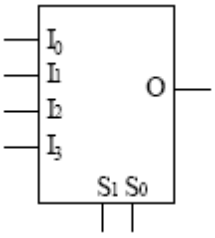


Implementación de funciones con Mux

Utilizando multiplexores es posible implementar funciones booleanas. En general, cualquier función de n variables se puede implementar utilizando un multiplexor.

$$F = (\bar{X} \times Y \times Z) + (X \times \bar{Y}) + (\bar{X} \times \bar{Y} \times \bar{Z})$$

Tiene 3 variables la función y se puede implementar utilizando un multiplexor de 2 entradas de control ($2^2 = 4$):



Utilizamos un método basado en la tabla de verdad, aunque algunas funciones se pueden implementar de manera más fácil si utilizamos la entrada de validación.

1. Se divide la tabla en tantos grupos como canales de entrada halla. En este caso hay 4 entradas, por lo que hacemos 4 grupos. Las variables de mayor peso se introducen directamente por las entradas de selección S_0 y S_1.

	Z	Y	X	F		
I_0	0	0	0	1	X = 0 → F =1 X = 1 → F =1	I_O="1"
	0	0	1	1		
I_1	0	1	0	0	X = 0 → F =0 X = 1 → F =0	I_1="0"
	0	1	1	0		
I_2	1	0	0	0	X = 0 → F =0 X = 1 → F =1	$I_2=X$
	1	0	1	1		
I_3	1	1	0	1	X = 0 → F =1 X = 1 → F =0	$I_3=\bar{X}$
	1	1	1	0		

2. Las variables Y e Z son las que se han introducido por las entradas de selección (S_0=Y y S_1=Z), o sea que se forman 4 grupos de filas, el primer grupo corresponde con la entrada I_0, el siguiente I_1, el siguiente I_2 y el último I_3.

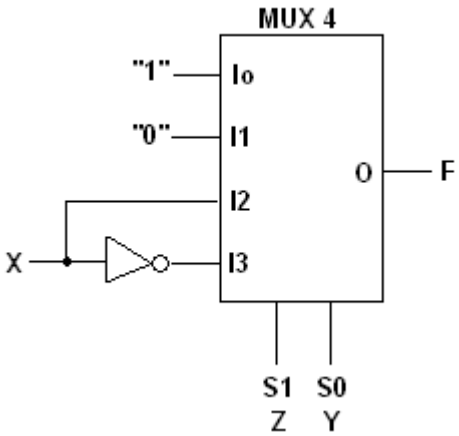
3. El valor a ingresar por las entradas I_0, I_1, I_2 e I_3 lo obtenemos mirando las columnas de la derecha (la columna X y la F).

   a. El primer grupo: cuando X=0 → F=1 y cuando X=1→ F=1 por tanto $F =$ "1". Esa será la salida cuando se seleccione el canal 0, por tanto en la entrada I_0 se ingresa un "1".

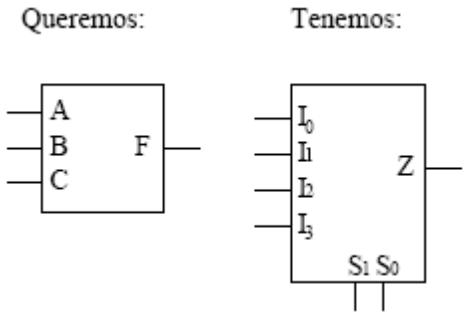
   b. El segundo grupo: cuando X=0 → F=0 y cuando X=1→ F=0 por tanto $F =$ "0". Esa será la salida cuando se seleccione el canal 1, por tanto en la entrada I_1 se ingresa un "0".

   c. El tercer grupo: cuando X=0 → F=0 y cuando X=1 → F=1 por tanto $F = X$. Esa será la salida cuando se seleccione el canal 2, por tanto en la entrada I_2 se ingresan los valores de la variable X.

   d. El cuarto grupo: cuando X=0 → F=1 y cuando X=1 → F=0 por tanto $F = \bar{X}$. Esa será la salida cuando se seleccione el canal 3, por tanto en la entrada I_3 se ingresan los valores de la variable $\bar{X}$.

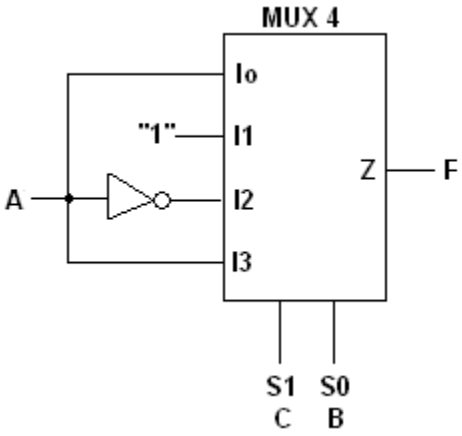


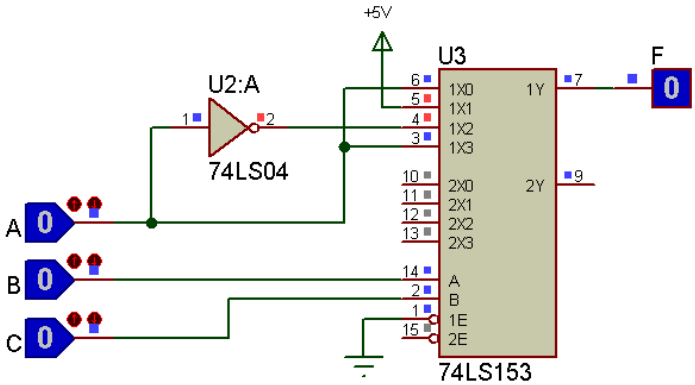
Ejemplo: implementar la función $F = (A \times B) + (\bar{A} \times B \times \bar{C}) + (A \times \bar{B} \times \bar{C}) + (\bar{A} \times \bar{B} \times C)$ utilizando un mux sin entrada de validación.



	C	B	A	F	
I_0	0	0	0	0	A = 0 → F =0
	0	0	1	1	A = 1 → F =1
I_1	0	1	0	1	A = 0 → F =1
	0	1	1	1	A = 1 → F =1
I_2	1	0	0	1	A = 0 → F =1
	1	0	1	0	A = 1 → F =0
I_3	1	1	0	0	A = 0 → F =0
	1	1	1	1	A = 1 → F =1

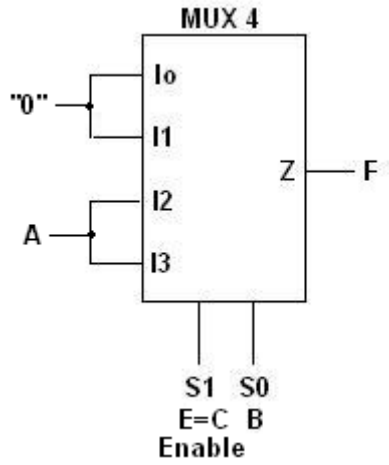
$I_0 = A$; $I_1 = "1"$; $I_2 = \bar{A}$; $I_3 = A$





Ejemplo: implementar la función $F = (A \times \bar{B} \times C) + (A \times B \times C)$ utilizando un mux con entrada de validación (E).

	C	B	A	F	
I_0	0	0	0	0	C = 0 → F =0 Enable Desactivado
	0	0	1	0	E= C = I_0 = I_1= 0
I_1	0	1	0	0	
	0	1	1	0	
I_2	1	0	0	0	A = 0 → F =0
	1	0	1	1	A = 1 → F =1 $I_2=A$
I_3	1	1	0	0	A = 0 → F =0
	1	1	1	1	A = 1 → F =1 $I_3=A$

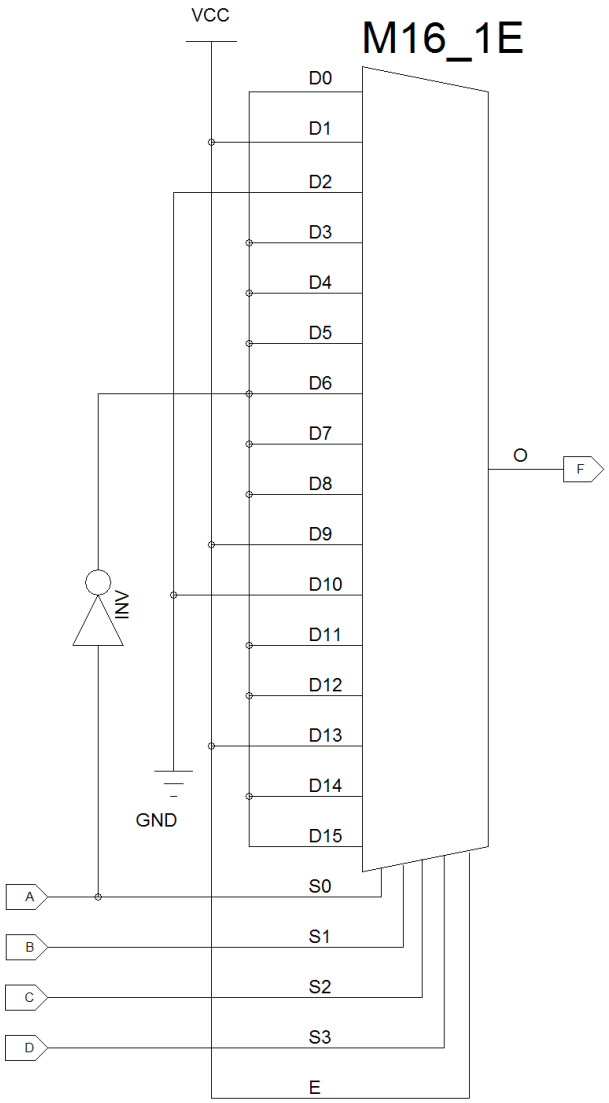


Ejemplo: implementar la función $F = \{\overline{(\bar{A} \times C)} \oplus (B + C)\} + \{\overline{(B + \bar{D})} \times D\}$ utilizando un mux de 16 entradas con entrada de validación.

D	C	B	A	F	
0	0	0	0	1	→ $I_0=\bar{A}$
0	0	0	1	1	→ I_1=1
0	0	1	0	0	→ I_2=0
0	0	1	1	0	→ $I_3=\bar{A}$
0	1	0	0	1	→ $I_4=\bar{A}$
0	1	0	1	0	→ $I_5=\bar{A}$
0	1	1	0	1	→ $I_6=\bar{A}$
0	1	1	1	0	→ $I_7=\bar{A}$
1	0	0	0	1	→ $I_8=\bar{A}$
1	0	0	1	1	→ I_9=1
1	0	1	0	0	→ I_{10}=0
1	0	1	1	0	→ $I_{11}=\bar{A}$
1	1	0	0	1	→ $I_{12}=\bar{A}$

1	1	0	1	1	$\rightarrow I_{13}=1$
1	1	1	0	1	$\rightarrow I_{14}=\bar{A}$
1	1	1	1	0	$\rightarrow I_{15}=\bar{A}$

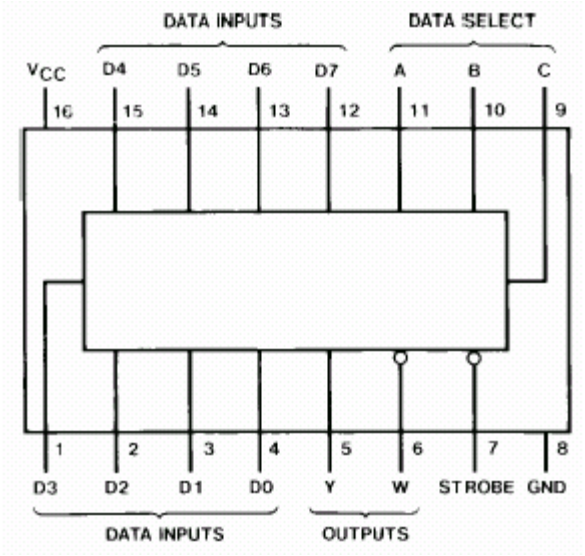
$\bar{A} = I_0, I_3, I_4, I_5, I_6, I_7, I_8, I_{11}, I_{12}, I_{14}, I_{15}$
"1" $= I_1, I_9, I_{13}$
"0" $= I_2, I_{10}$



Circuitos Integrados Multiplexores

Multiplexor de 8 entradas – 74151

Las 8 entradas son seleccionadas por el selector de datos (A – B – C, pines 9 – 10 – 11), posee una terminal que habilita el selector de datos con un nivel bajo (“0”) llamado strobe (pin 7) y dos salidas (Y) que es normal y (W) que es complementaria



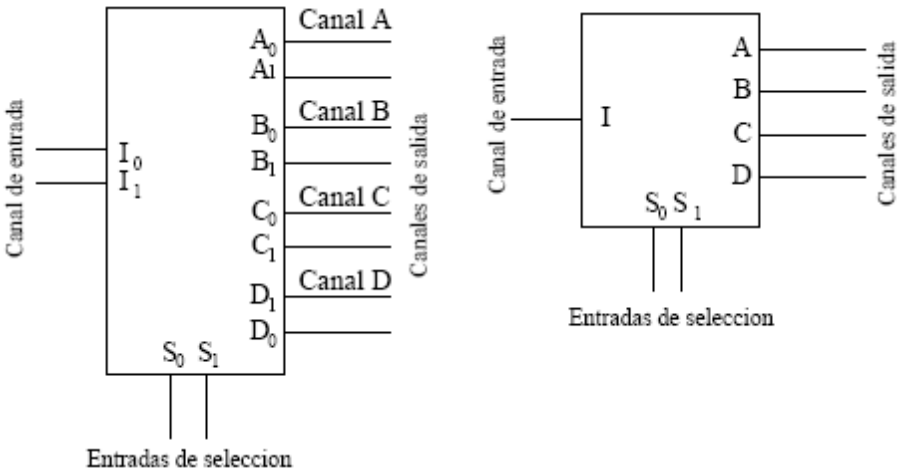
Inputs				Outputs	
Select			Strobe		
C	B	A	S	Y	W
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

DEMULTIPLEXOR (DEMUX)

En los Demultiplexores hay un único canal de entrada de datos que puede exhibirse por múltiples salidas (una a la vez).

Si usamos la analogía de la finca y las tuberías sería: supongamos, a la finca solo llega una única tubería con agua, pero al interior de la finca hay varias mangueras que se destinan a lugares diferentes por consiguiente no se pueden utilizar varias mangueras a la vez ya que están en sitios diferentes. Por medio de una llave de paso se selecciona la manguera por la que saldrá el agua.

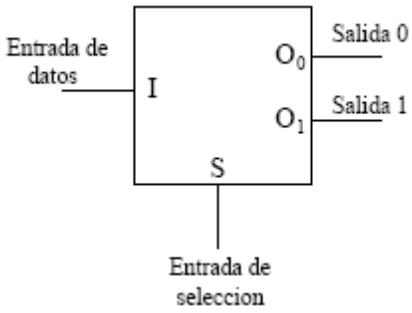
Los Demux están compuestos por la única Entrada de Datos (manguera), Selector de Datos (llave de paso) y las Salidas (múltiples mangueras).



Ambos DEMUX tienen 4 canales de salida de datos y para ello se necesitan 2 bit de Selector de datos ($2^2 = 4$, para poder seleccionar los 4 posibles canales). El del lado izquierdo tiene 2 bit de entrada como único canal y 2 bit de salida por cada canal, el de la derecha tiene 1 bit de entrada como único canal y un bit en cada uno de los 4 canales de salida.

Demux con 1 entrada de Selección

Este Demux tiene una entrada de datos y 2 salidas, según el valor de la entrada del Selector designará la salida O_0 o la O_1



Para obtener la tabla aplicamos la definición de Demultiplexor y vamos comprobando caso por caso que valores aparecen en las salidas. Por ejemplo, si S=1 e I=1, se estará seleccionando la salida O_1 y por ella saldrá el valor de I = "1". La salida O_0 no estará seleccionada y tendrá el valor 0.

S	I	O_1	O_0	
0	0	0	0	S = 0 → O_0 = I
0	1	0	1	
1	0	0	0	S = 1 → O_1 = I
1	1	1	0	

La forma canónica es:

$$O_1 = S \times I$$
$$O_0 = \bar{S} \times I$$

Reemplazando en las Funciones:

- Si se selecciona la salida 0 (S = 0):

$$O_1 = S \times I \therefore O_1 = 0 \times I \therefore O_1 = 0$$

$$O_0 = \bar{S} \times I \therefore O_0 = \bar{0} \times I \therefore O_0 = 1 \times I \ \therefore O_0 = I$$

- Si se selecciona la salida 1 (S = 1):

$$O_1 = S \times I \therefore O_1 = 1 \times I \therefore O_1 = I$$
$$O_0 = \bar{S} \times I \therefore O_0 = \bar{1} \times I \therefore O_0 = 0 \times I \ \therefore O_0 = 0$$

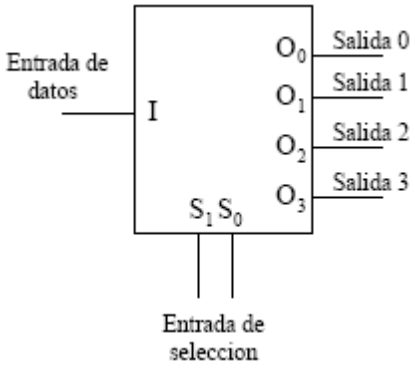
Podemos considerar que las funciones O_0 y O_1 solo dependen de la entrada de selección (S), teniendo la entrada I como parámetro y podemos simplificar su descripción así:

S	O_1	O_0
0	0	I
1	I	0

La salida O_0 vale I cuando S = 0 y la salida O_1 vale I cuando S = 1.

Demux con 2 entradas de Selección

Tiene dos entradas de selección y cuatro salidas (2^2 = 4 posibles combinaciones de salida)



La tabla de verdad “abreviada” es:

S_1	S_0	O_3	O_2	O_1	O_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

La entrada I se saca por la salida indicada en las entradas de selección. Las ecuaciones de las funciones de salida son:

$$O_0 = \bar{S_1} \times \bar{S_0} \times I$$
$$O_1 = \bar{S_1} \times S_0 \times I$$
$$O_2 = S_1 \times \bar{S_0} \times I$$
$$O_3 = S_1 \times S_0 \times I$$

Analizando la ecuación de O_0: O_0 = I sólo cuando S_1 =”0” y S_0 =”0”.

Demux con 3 entradas de Selección

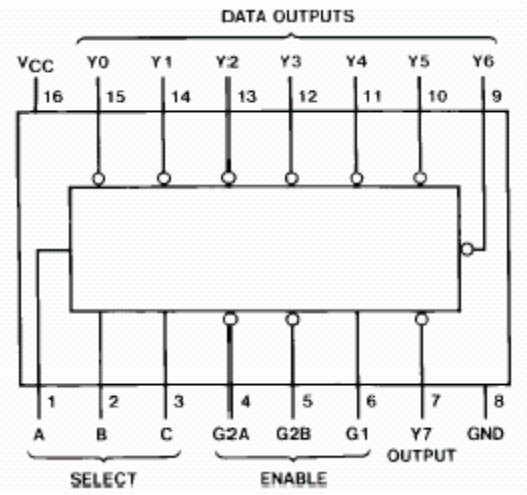
Un Demux con 3 entradas de Selección (S_2, S_1, S_0) tendrá 8 salidas, las ecuaciones para las salidas será:

$$O_5 = S_2 \times \overline{S_1} \times S_0 \times I \ldots . O_0 = \overline{S_2} \times \overline{S_1} \times \overline{S_0} \times I$$

Circuitos Integrados Demultiplexores

Demultiplexor de 8 salidas – 74138

El demultiplexor (DEMUX) selecciona una de las 8 salidas basadas en las condiciones de las 3 entradas binarias (A – B – C, pines 1 – 2 –3) y las 3 entradas de habilitación (enables, G1 – G2A – G2B, pines 4 – 5 – 6). Una sola entrada de habilitación puede utilizarse como una entrada de datos para la demultiplexación. G2 = G2A + G2B.



LS138

Inputs					Outputs							
Enable		Select										
G1	G2*	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

* G2 = G2A + G2B

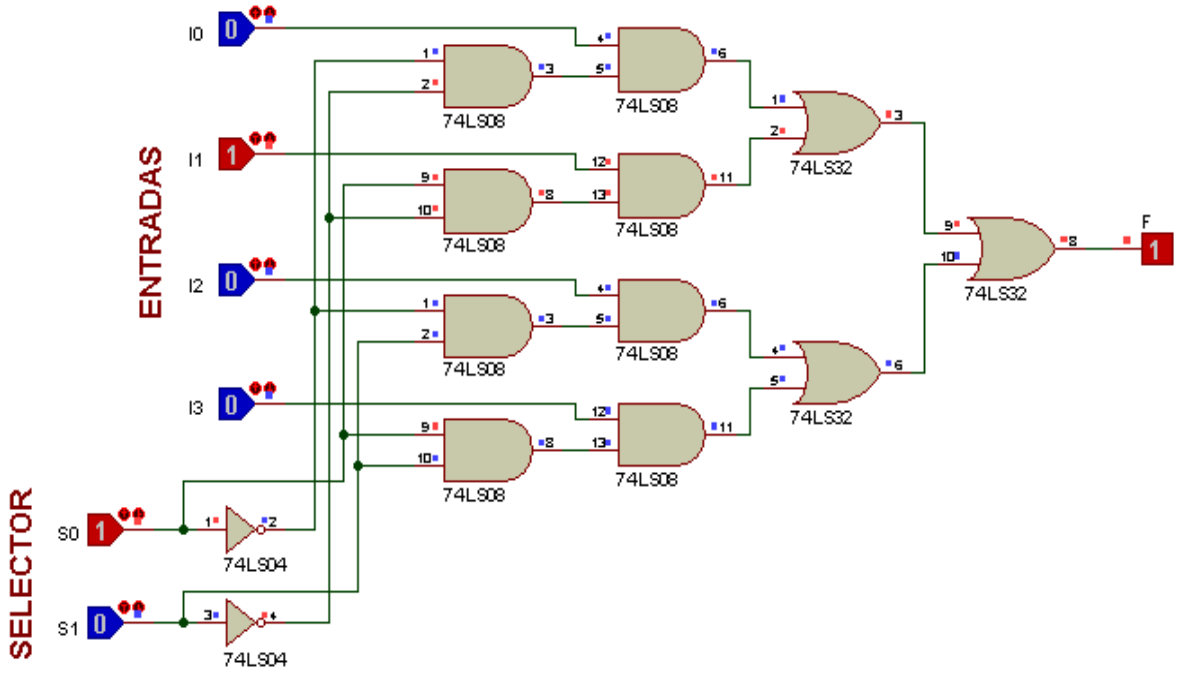
H = High Level, L = Low Level, X = Don't Care

EJEMPLOS DE SIMULACIÓN Y PROGRAMACIÓN

Multiplexor de 4 entradas sin validación.

Simulación con Proteus

$$F = (\overline{S_1} \times \overline{S_0} \times I_0) + (\overline{S_1} \times S_0 \times I_1) + (S_1 \times \overline{S_0} \times I_2) + (S_1 \times S_0 \times I_3)$$



Programación con ISE de Xilinx

Método de Flujo de Datos

```
----------------------------------------------------------------------------------
-- MULTIPLEXOR DE 4 ENTRADAS - FLUJO DE DATOS
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Mux4_flujo is
    Port ( I0 : in  STD_LOGIC;
           I1 : in  STD_LOGIC;
           I2 : in  STD_LOGIC;
           I3 : in  STD_LOGIC;
           S0 : in  STD_LOGIC;
           S1 : in  STD_LOGIC;
           F : out  STD_LOGIC);
end Mux4_flujo;

architecture Flujo of Mux4_flujo is

begin
F<=(NOT S0 AND NOT S1 AND I0) OR (S0 AND NOT S1 AND I1) OR (NOT S0
AND S1 AND I2) OR (S0 AND S1 AND I3);
```

```
end Flujo;
```

Método Estructural

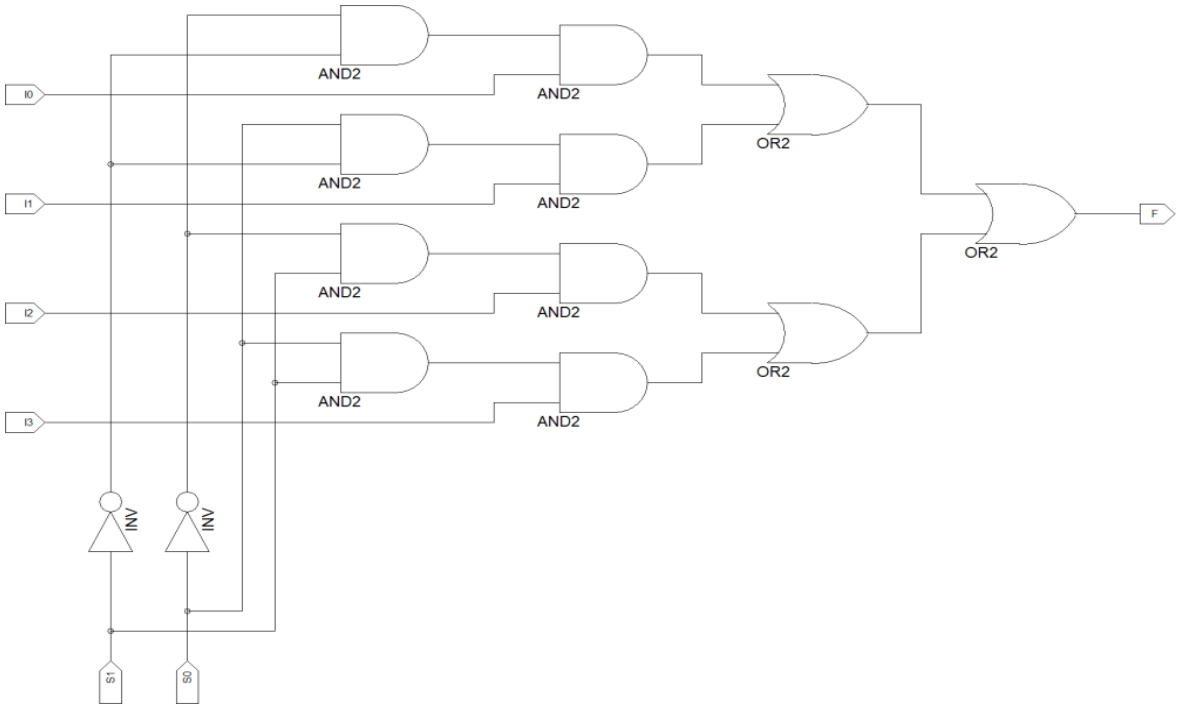
```
--------------------------------------------------------------------------------
-- MULTIPLEXOR DE 4 ENTRADAS - ESTRUCTURAL
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Mux4_estruc is
    Port ( I0 : in  STD_LOGIC;
           I1 : in  STD_LOGIC;
           I2 : in  STD_LOGIC;
           I3 : in  STD_LOGIC;
           S0 : in  STD_LOGIC;
           S1 : in  STD_LOGIC;
           F : out  STD_LOGIC);
end Mux4_estruc;
architecture Estructural of Mux4_estruc is
        SIGNAL      S0N,S1N,Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8,Y9,Y10:STD_LOGIC;      -- Señales Intermedias

        COMPONENT INV           -- Estructura de la Compuerta
                PORT (I:IN STD_LOGIC;O:OUT STD_LOGIC);
        END COMPONENT;

        COMPONENT AND2
                PORT (I0,I1:IN STD_LOGIC;O:OUT STD_LOGIC);
        END COMPONENT;

        COMPONENT OR2
                PORT (I0,I1:IN STD_LOGIC;O:OUT STD_LOGIC);
        END COMPONENT;

begin
        U1: INV PORT MAP(I=>S0,O=>S0N);    -- Conexión de las Compuertas
        U2: INV PORT MAP(I=>S1,O=>S1N);
        U3: AND2 PORT MAP(I0=>S0N,I1=>S1N,O=>Y1);
        U4: AND2 PORT MAP(I0=>S0,I1=>S1N,O=>Y2);
        U5: AND2 PORT MAP(I0=>S0N,I1=>S1,O=>Y3);
        U6: AND2 PORT MAP(I0=>S0,I1=>S1,O=>Y4);
        U7: AND2 PORT MAP(I0=>Y1,I1=>I0,O=>Y5);
        U8: AND2 PORT MAP(I0=>Y2,I1=>I1,O=>Y6);
        U9: AND2 PORT MAP(I0=>Y3,I1=>I2,O=>Y7);
```

```
        U10: AND2 PORT MAP(I0=>Y4,I1=>I3,O=>Y8);
        U11: OR2 PORT MAP(I0=>Y5,I1=>Y6,O=>Y9);
        U12: OR2 PORT MAP(I0=>Y7,I1=>Y8,O=>Y10);
        U13: OR2 PORT MAP(I0=>Y9,I1=>Y10,O=>F);
end Estructural;
```

Método Comportamental

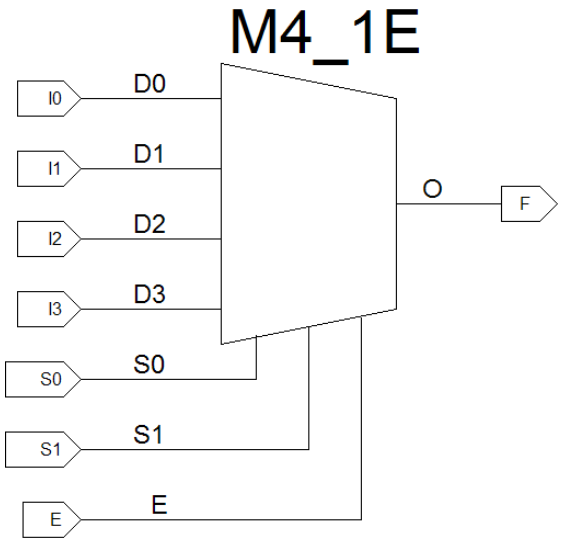
```
--------------------------------------------------------------------------------
-- MULTIPLEXOR DE 4 ENTRADAS - COMPORTAMENTAL
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Mux4_compor is
    Port ( I0 : in  STD_LOGIC;
           I1 : in  STD_LOGIC;
           I2 : in  STD_LOGIC;
           I3 : in  STD_LOGIC;
           S0 : in  STD_LOGIC;
           S1 : in  STD_LOGIC;
           F : out  STD_LOGIC);
end Mux4_compor;

architecture Comportamental of Mux4_compor is
begin
        PROCESS(S0,S1,I0,I1,I2,I3)  -- Lista de Sensibilidad
begin
        IF (NOT S0 AND NOT S1)='1' THEN -- Si...entonces...
                F<=I0;
                ELSIF (S0 AND NOT S1)='1' THEN – Si No…entonces…
                        F<=I1;
                                ELSIF (NOT S0 AND S1)='1' THEN
                                F<=I2;
                                        ELSIF (S0 AND S1)='1' THEN
                                        F<=I3;
                ELSE
                F<='0';
        END IF;
END PROCESS;

end Comportamental;
```

Método Esquemático – Compuertas Lógicas



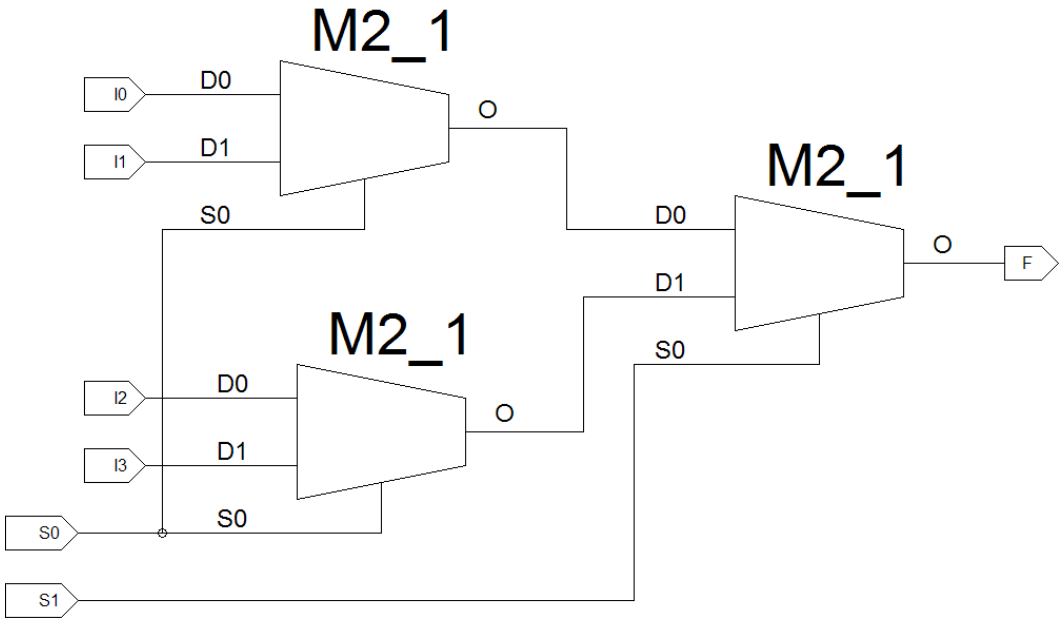
Método Esquemático – Módulo M4_1E

Se implementó con el módulo del Mux denominado M4_1E (con validación – E), debido a que no lo hay sin el *Enable*. La validación (E) se activa en alto "1".



Método Esquemático – Módulo M2_1

Implementado con Mux de 2 entradas (módulos M2_1), extensión de Multiplexores por aumento del número de entradas.



Programación con ISE de Xilinx

Demultiplexor de 8 salidas.

C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

$$O_0 = I(\bar{A} \times \bar{B} \times \bar{C})$$
$$O_1 = I(A \times \bar{B} \times \bar{C})$$
$$O_2 = I(\bar{A} \times B \times \bar{C})$$
$$O_3 = I(A \times B \times \bar{C})$$
$$O_4 = I(\bar{A} \times \bar{B} \times C)$$
$$O_5 = I(A \times \bar{B} \times C)$$
$$O_6 = I(\bar{A} \times B \times C)$$
$$O_7 = I(A \times B \times C)$$

Método de Flujo de Datos

```
--------------------------------------------------------------------------------
-- DEMULTIPLEXOR DE 8 SALIDAS - FLUJO DE DATOS
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Demux8_flujo is
    Port ( I : in  STD_LOGIC;
           A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : in  STD_LOGIC;
          O0 : out  STD_LOGIC;
           O1 : out  STD_LOGIC;
           O2 : out  STD_LOGIC;
           O3 : out  STD_LOGIC;
           O4 : out  STD_LOGIC;
           O5 : out  STD_LOGIC;
           O6 : out  STD_LOGIC;
           O7 : out  STD_LOGIC);
end Demux8_flujo;

architecture Flujo of Demux8_flujo is
          begin
          O0<=NOT A AND NOT B AND NOT C AND I;
          O1<=A AND NOT B AND NOT C AND I;
          O2<=NOT A AND B AND NOT C AND I;
          O3<=A AND B AND NOT C AND I;
          O4<=NOT A AND NOT B AND C AND I;
          O5<=A AND NOT B AND C AND I;
          O6<=NOT A AND B AND C AND I;
          O7<=A AND B AND C AND I;
end Flujo;
```

Método Estructural

```
--------------------------------------------------------------------------------
-- DEMULTIPLEXOR DE 8 SALIDAS - ESTRUCTURAL
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Demux8_estructural is
```

```
    Port ( I : in  STD_LOGIC;
           A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : in  STD_LOGIC;
           O0 : out  STD_LOGIC;
           O1 : out  STD_LOGIC;
           O2 : out  STD_LOGIC;
           O3 : out  STD_LOGIC;
           O4 : out  STD_LOGIC;
           O5 : out  STD_LOGIC;
           O6 : out  STD_LOGIC;
           O7 : out  STD_LOGIC);
end Demux8_estructural;

architecture Estructural of Demux8_estructural is
        SIGNAL An,Bn,Cn:STD_LOGIC; -- Señales intermedias
                COMPONENT INV           -- Estructura de la Compuerta
                        PORT (I:IN STD_LOGIC;O:OUT STD_LOGIC);
                END COMPONENT;

                COMPONENT AND4
                        PORT (I0,I1,I2,I3:IN STD_LOGIC;O:OUT STD_LOGIC);
                END COMPONENT;
        begin
        U1: INV PORT MAP(I=>A,O=>An);       -- Conexión de las Compuertas
        U2: INV PORT MAP(I=>B,O=>Bn);
        U3: INV PORT MAP(I=>C,O=>Cn);
        U4: AND4 PORT MAP(I0=>An,I1=>Bn,I2=>Cn,I3=>I,O=>O0);
        U5: AND4 PORT MAP(I0=>A,I1=>Bn,I2=>Cn,I3=>I,O=>O1);
        U6: AND4 PORT MAP(I0=>An,I1=>B,I2=>Cn,I3=>I,O=>O2);
        U7: AND4 PORT MAP(I0=>A,I1=>B,I2=>Cn,I3=>I,O=>O3);
        U8: AND4 PORT MAP(I0=>An,I1=>Bn,I2=>C,I3=>I,O=>O4);
        U9: AND4 PORT MAP(I0=>A,I1=>Bn,I2=>C,I3=>I,O=>O5);
        U10: AND4 PORT MAP(I0=>An,I1=>B,I2=>C,I3=>I,O=>O6);
        U11: AND4 PORT MAP(I0=>A,I1=>B,I2=>C,I3=>I,O=>O7);
end Estructural;
```

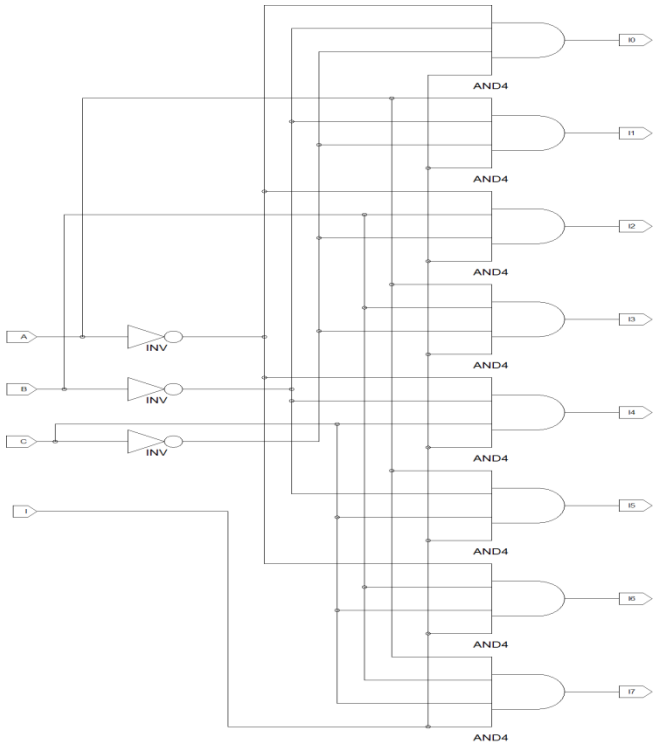
Método Comportamental

```
--------------------------------------------------------------------------------
-- DEMULTIPLEXOR DE 8 SALIDAS - COMPORTAMENTAL
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity Demux8_comporta is
   Port ( I : in  STD_LOGIC;
          A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          C : in  STD_LOGIC;
          O0 : out  STD_LOGIC;
          O1 : out  STD_LOGIC;
          O2 : out  STD_LOGIC;
          O3 : out  STD_LOGIC;
          O4 : out  STD_LOGIC;
          O5 : out  STD_LOGIC;
          O6 : out  STD_LOGIC;
          O7 : out  STD_LOGIC);
end Demux8_comporta;

architecture Comportamental of Demux8_comporta is
        begin
                Process(A,B,C,I) -- Lista de sensibilidad
        begin
        IF (NOT A AND NOT B AND NOT C)='1' THEN -- Si...entonces...
        O0<=I;
           ELSIF (A AND NOT B AND NOT C)='1' THEN -- Si no Si... entonces...
           O1<=I;
                   ELSIF (NOT A AND B AND NOT C)='1' THEN
                   O2<=I;
                        ELSIF (A AND B AND NOT C)='1' THEN
                        O3<=I;
                                ELSIF (NOT A AND NOT B AND C)='1' THEN
                                O4<=I;
                                        ELSIF (A AND NOT B AND C)='1' THEN
                                        O5<=I;
                                                ELSIF (NOT A AND B AND C)='1' THEN
                                                O6<=I;
                                                        ELSIF (A AND B AND C)='1' THEN
                                                        O7<=I;
                                                        ELSE --Si no...
                                                        O0<='0';
                                                        O1<='0';
                                                        O2<='0';
                                                        O3<='0';
                                                        O4<='0';
                                                        O5<='0';
                                                        O6<='0';
                                                        O7<='0';
        END IF;
END PROCESS;
```

end Comportamental;

Método Esquemático – Compuertas Lógicas



$$O_0 = I(\bar{A} \times \bar{B} \times \bar{C})$$
$$O_1 = I(A \times \bar{B} \times \bar{C})$$
$$O_2 = I(\bar{A} \times B \times \bar{C})$$
$$O_3 = I(A \times B \times \bar{C})$$
$$O_4 = I(\bar{A} \times \bar{B} \times C)$$
$$O_5 = I(A \times \bar{B} \times C)$$
$$O_6 = I(\bar{A} \times B \times C)$$
$$O_7 = I(A \times B \times C)$$