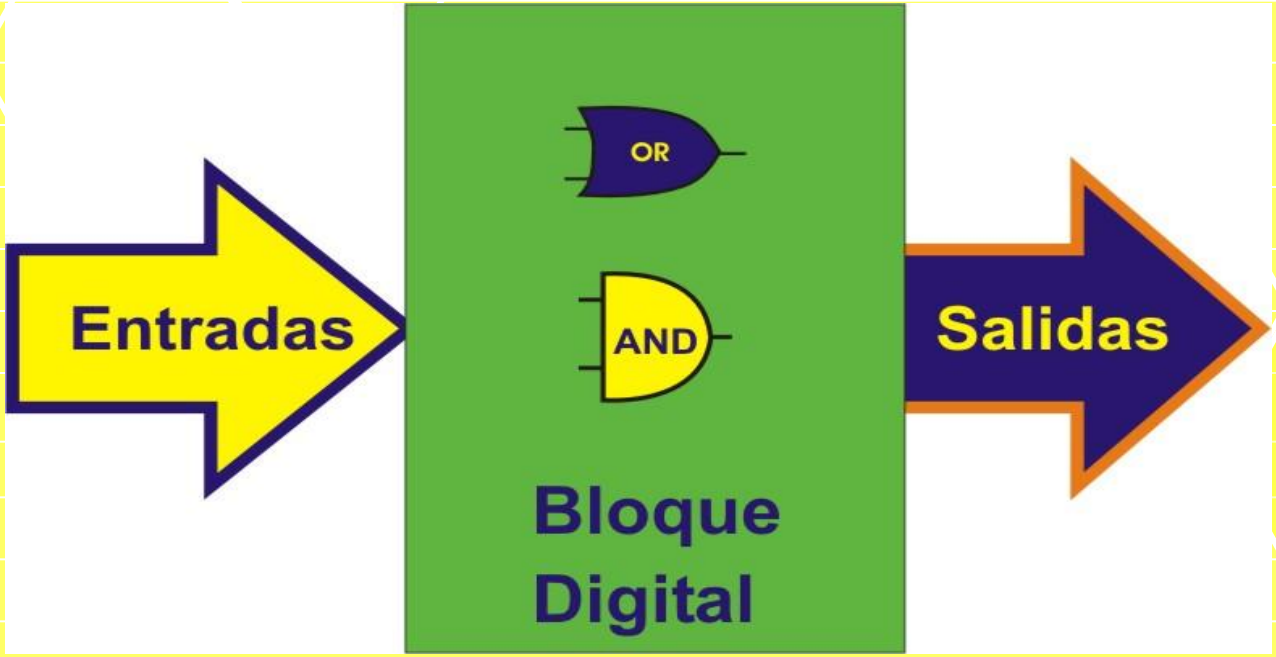
Diseño de Sistemas Combinacionales

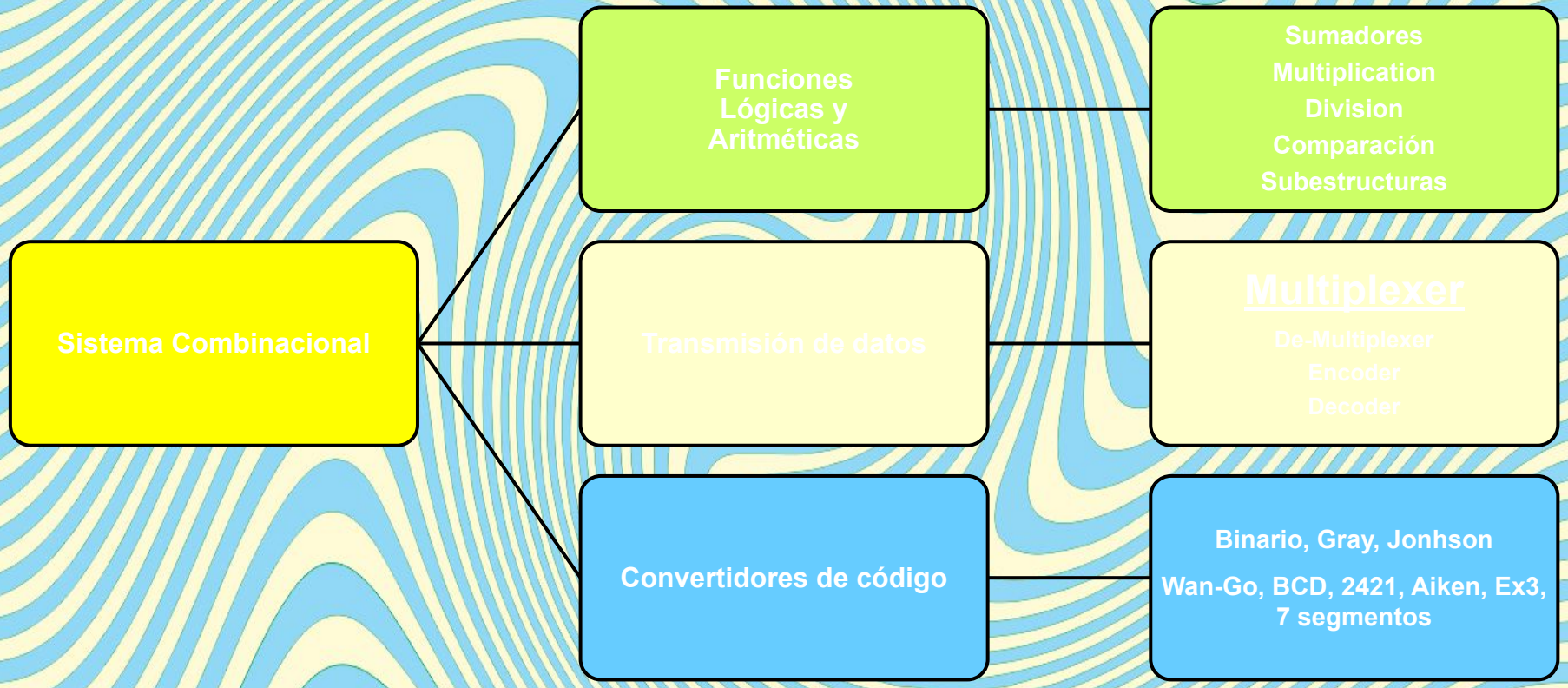
"No me digas las horas que trabajas, dime qué consigues." *Vincet castellano"*

Sistema Combinacional

Es aquel bloque digital en donde los valores de salida dependen únicamente de las combinaciones de entrada.



Aplicaciones de los Sistemas Combinacionales



Método del Diseño Combinacional

HDL equations

1.- Especificar el Sistema

2.- Determinar entradas y salidas

3.- Construir la Tabla de Verdad

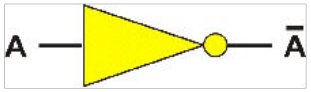
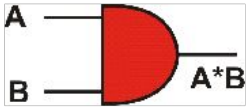
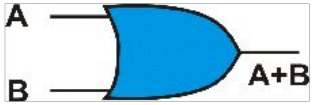
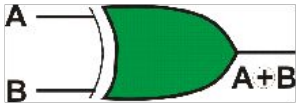
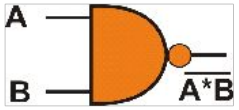
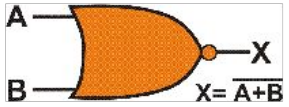
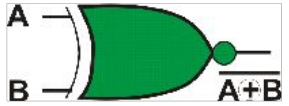
4.- Ecuaciones Mínimas

~~5.- Diagrama Esquemático~~

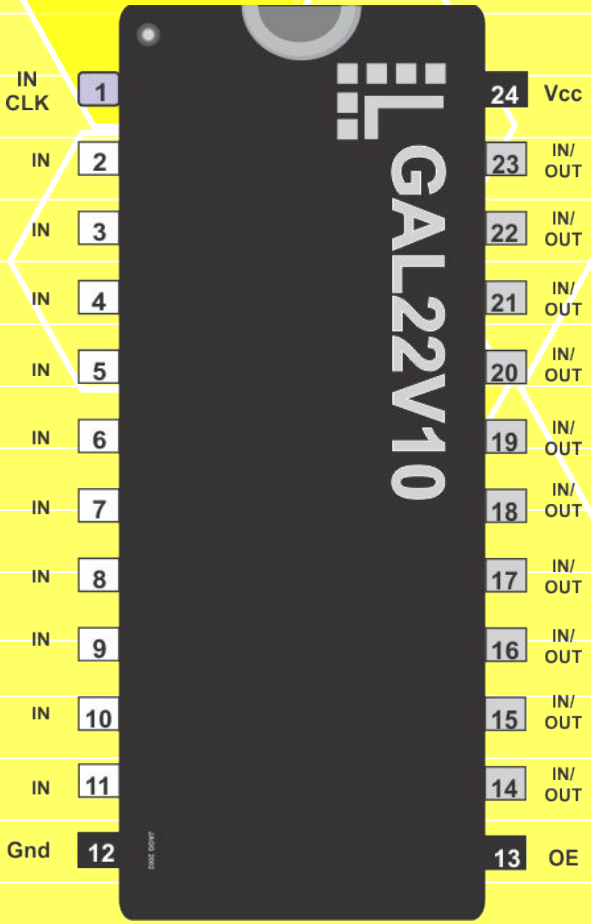
6.- Simulación

7.- Construir un Prototipo

Lenguaje de Descripción de Hardware
ABEL-HDL, equations

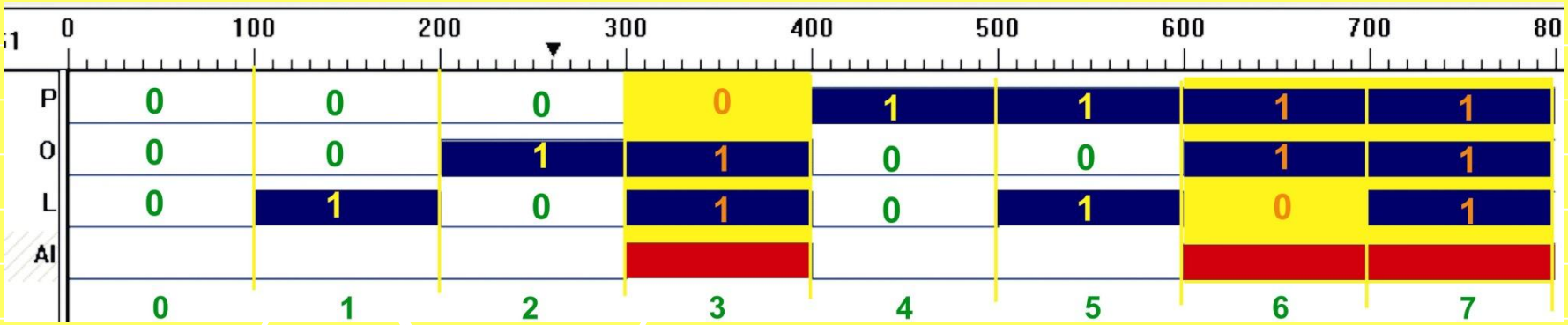
Operador	Descripción	Ecuación en ABEL-HDL	Símbolo
!	NOT	!A	
&	AND	A&B	
#	OR	A#B	
$	EXOR	A$B	
!&	NAND	!(A&B)	
!#	NOR	!(A#B)	
!$	EXNOR	!(A$B)	

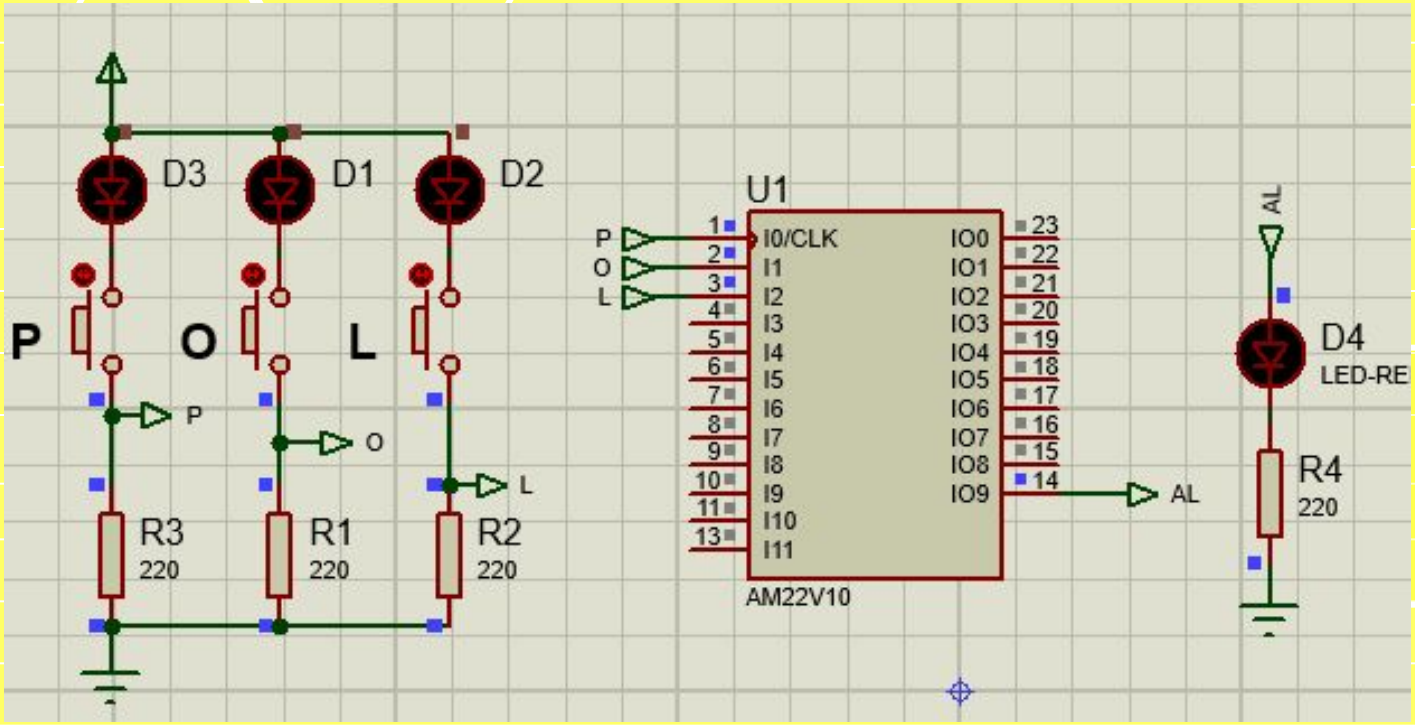
Archivo en formato ABEL-HDL incluyendo la simulación




```
MODULE ovejas
"entradas
P,O,L pin 1,2,3;
"salida
Al pin 14 istype 'com';
equations
Al=O&(P#L);
test_vectors
([P,O,L]->Al)
[0,0,0]->.x.;
[0,0,1]->.x.;
[0,1,0]->.x.;
[0,1,1]->.x.;
[1,0,0]->.x.;
[1,0,1]->.x.;
[1,1,0]->.x.;
[1,1,1]->.x.;
END
```

Simulación


0
100
200
300
400
500
600
700
P
O
L
Al
0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1
0 1 2 3 4 5 6 7


D3
D1
D2
P
O
L
R3
220
R1
220
R2
220
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
AL
D4
LED-RE
R4
220

Método del Diseño Combinacional

1.- Especificar el Sistema

2.- Determinar entradas y salidas

3.- Construir la Tabla de Verdad

~~4.- Ecuaciones Mínimas~~

~~5.- Diagrama Esquemático~~

6.- Simulación

7.- Construir un Prototipo

ABEL-HDL
Truth_TABLE

Programación en ABEL-HDL

```
MODULE cpistas
"23 marzo 2020 Diseño combinacional Aeropuerto de 4 pistas
A,B,C,D pin 1..4;
ANTONOV,AIRBUS pin 15,14 istype
' ' ' '
Truth_Table
([A,B,C,D]->[ANTONOV,AIRBUS])
[0,0,0,0]->[0,0];
[0,0,0,1]->[0,0];
[0,0,1,0]->[0,0];
[0,0,1,1]->[0,1];
[0,1,0,0]->[0,0];
[0,1,0,1]->[0,0];
[0,1,1,0]->[0,1];
[0,1,1,1]->[1,0];
[1,0,0,0]->[0,0];
[1,0,0,1]->[0,0];
[1,0,1,0]->[0,0];
[1,0,1,1]->[0,1];
[1,1,0,0]->[0,1];
[1,1,0,1]->[0,1];
[1,1,1,0]->[1,0];
[1,1,1,1]->[1,0];
```

m	A	B	C	D	ANTONOV	AIRBUS
0	0	0	0	0	0	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	0	1	1	0	1
4	0	1	0	0	0	0
5	0	1	0	1	0	0
6	0	1	1	0	0	1
7	0	1	1	1	1	0
8	1	0	0	0	0	0
9	1	0	0	1	0	0
10	1	0	1	0	0	0
11	1	0	1	1	0	1
12	1	1	0	0	0	1
13	1	1	0	1	0	1
14	1	1	1	0	1	0
15	1	1	1	1	1	0

Programación en ABEL-HDL

```
MODULE cpistas
"23 marzo 2020
"Aeropuerto de 4 pistas
A,B,C,D pin 1..4;
ANTONOV,AIRBUS pin 15,14 istype 'com';
```

```
Truth_Table
([A,B,C,D]->[ANTONOV,AIRBUS])
[0,0,1,1]->[0,1];
[0,1,1,0]->[0,1];
[0,1,1,1]->[1,0];
[1,0,1,1]->[0,1];
[1,1,0,0]->[0,1];
[1,1,0,1]->[0,1];
[1,1,1,0]->[1,0];
[1,1,1,1]->[1,0];
```

Las salidas de las combinaciones no listadas la toma como cero

m	A	B	C	D	ANTONOV	AIRBUS
0	0	0	0	0	0	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	0	1	1	0	1
4	0	1	0	0	0	0
5	0	1	0	1	0	0
6	0	1	1	0	0	1
7	0	1	1	1	1	0
8	1	0	0	0	0	0
9	1	0	0	1	0	0
10	1	0	1	0	0	0
11	1	0	1	1	0	1
12	1	1	0	0	0	1
13	1	1	0	1	0	1
14	1	1	1	0	1	0
15	1	1	1	1	1	0

En **ABEL-HDL** se pueden definir **conjuntos de valores binarios** para agrupar varias señales en una sola variable compuesta.
Por ejemplo: ***P=[A,B,C,D]; S=[AN,AI];***
Esto permite representar un grupo de bits como una variable única, similar a un valor decimal

P=[A,B,C,D];
S=[AN,AI];

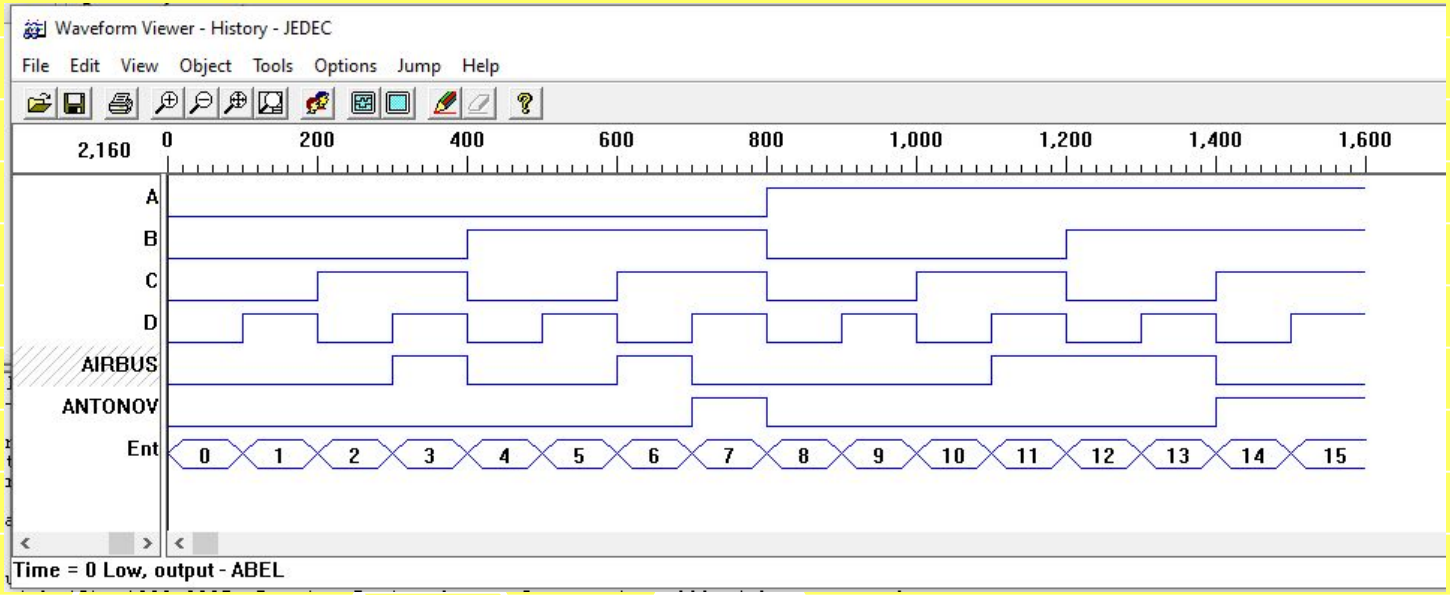
Truth_Table
(P->S)
3->1;
6->1;
7->2;
11->1;
12->1;
13->1;
14->2;
15->2;

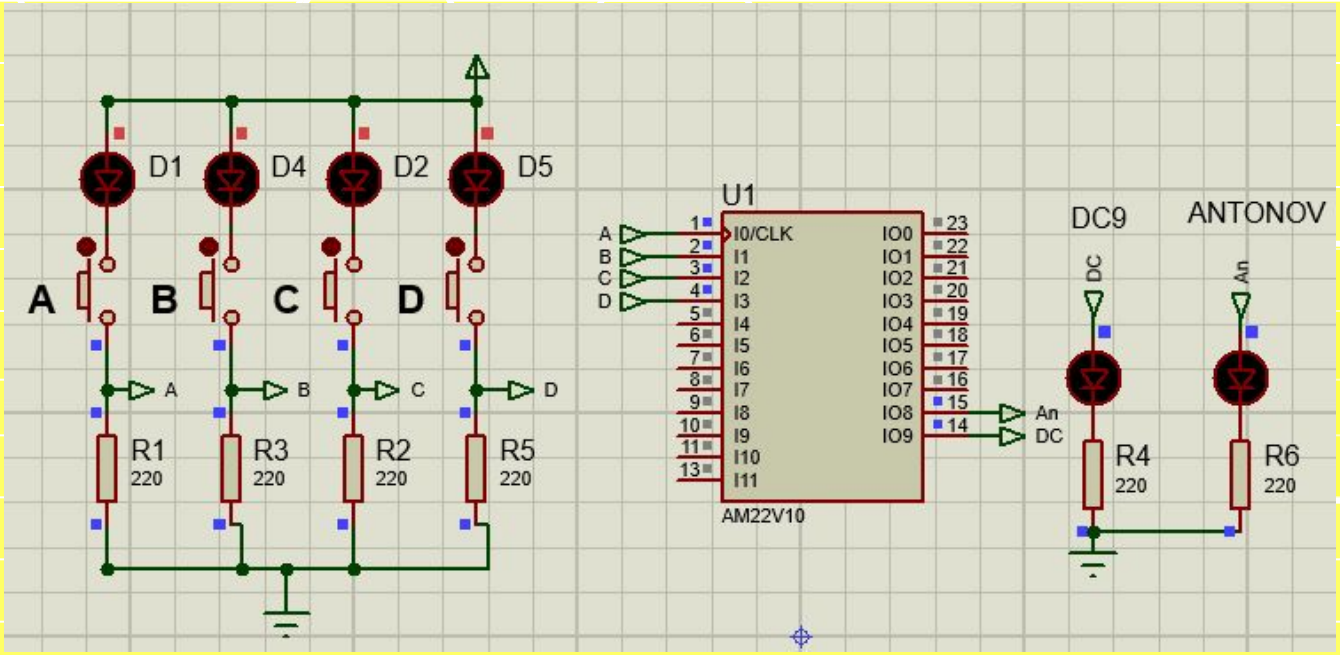
m	A	B	C	D	ANTONOV	AIRBUS	N(10)
0	0	0	0	0	0	0	
1	0	0	0	1	0	0	
2	0	0	1	0	0	0	
3	0	0	1	1	0	1	1
4	0	1	0	0	0	0	
5	0	1	0	1	0	0	
6	0	1	1	0	0	1	1
7	0	1	1	1	1	0	2
8	1	0	0	0	0	0	
9	1	0	0	1	0	0	
10	1	0	1	0	0	0	
11	1	0	1	1	0	1	1
12	1	1	0	0	0	1	1
13	1	1	0	1	0	1	1
14	1	1	1	0	1	0	2
15	1	1	1	1	1	0	2

```
MODULE cpistas
A,B,C,D pin 1..4;
ANTONOV,AIRBUS pin 15,14 istype 'com';
P=[A,B,C,D]; S=[AN,AI];
```

```
Truth_Table
(P->S)
3->1;
6->1;
7->2;
11->1;
12->1;
13->1;
14->2;
15->2;
```

```
Test_vectors
(P->S)
0->.x.;
1->.x.;
2->.x.;
3->.x.;
4->.x.;
5->.x.;
6->.x.;
7->.x.;
8->.x.;
9->.x.;
10->.x.;
11->.x.;
12->.x.;
13->.x.;
14->.x.;
15->.x.;
END
```


Waveform Viewer - History - JEDEC
File Edit View Object Tools Options Jump Help
2,160
0
200
400
600
800
1,000
1,200
1,400
1,600
A
B
C
D
AIRBUS
ANTONOV
Ent
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Time = 0 Low, output - ABEL

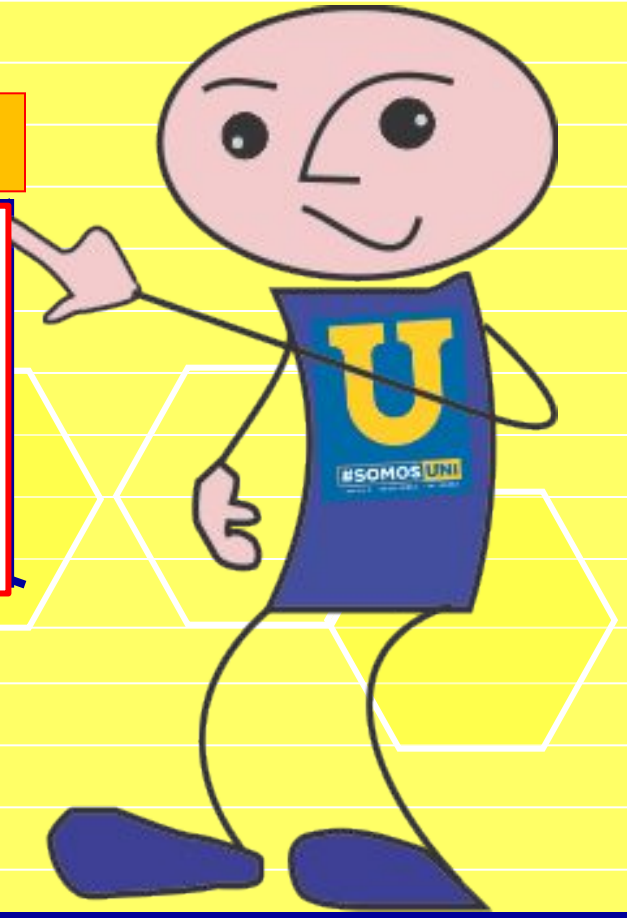

D1
D4
D2
D5
A
B
C
D
R1
220
R3
220
R2
220
R5
220
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
An
DC
DC9
ANTONOV
R4
220
R6
220

Método del Diseño Combinacional

1.- Especificar el Sistema

2.- Determinar entradas y salidas

ABEL-HDL
Equations
When Then Else

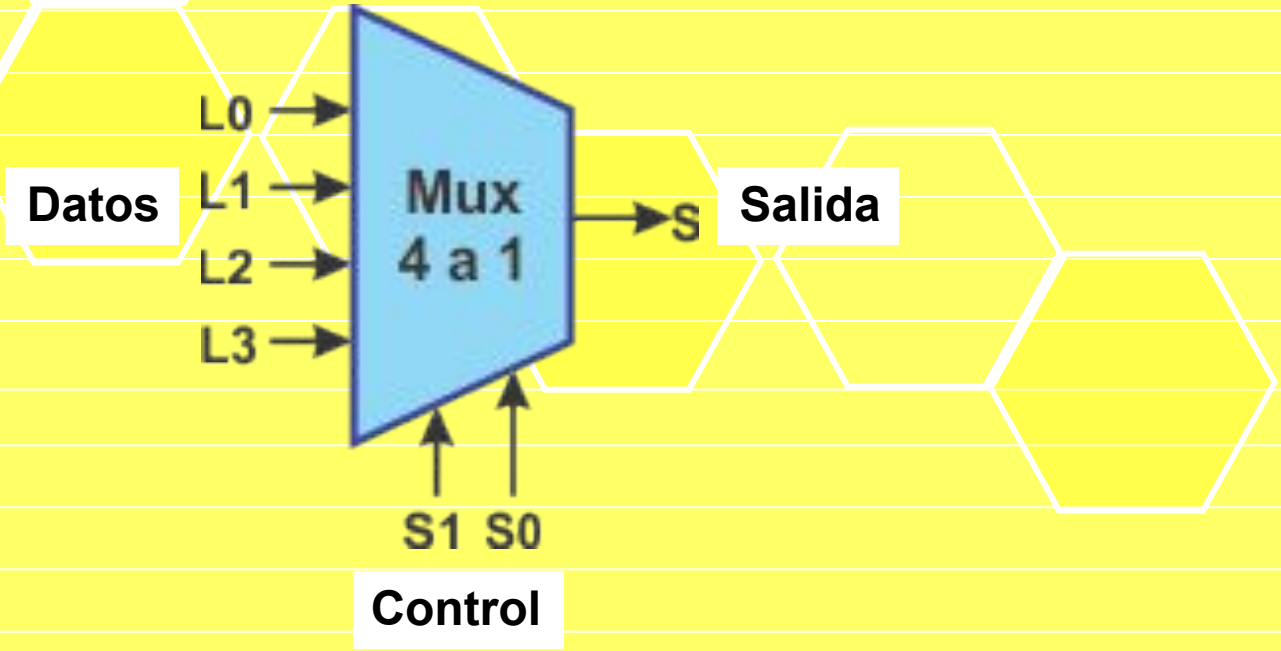


6.- Simulación

7.- Construir un Prototipo

Multiplexor (selector de datos)

Un multiplexor (MUX) es un circuito combinacional que selecciona solo una de varias entradas y la dirige hacia una única salida, según el valor de las señales de control.



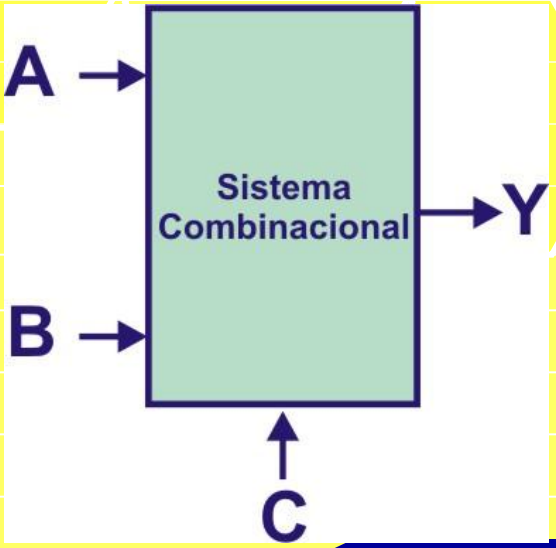
Diseñe un Multiplexor

(selector datos)

de 2 a 1 línea

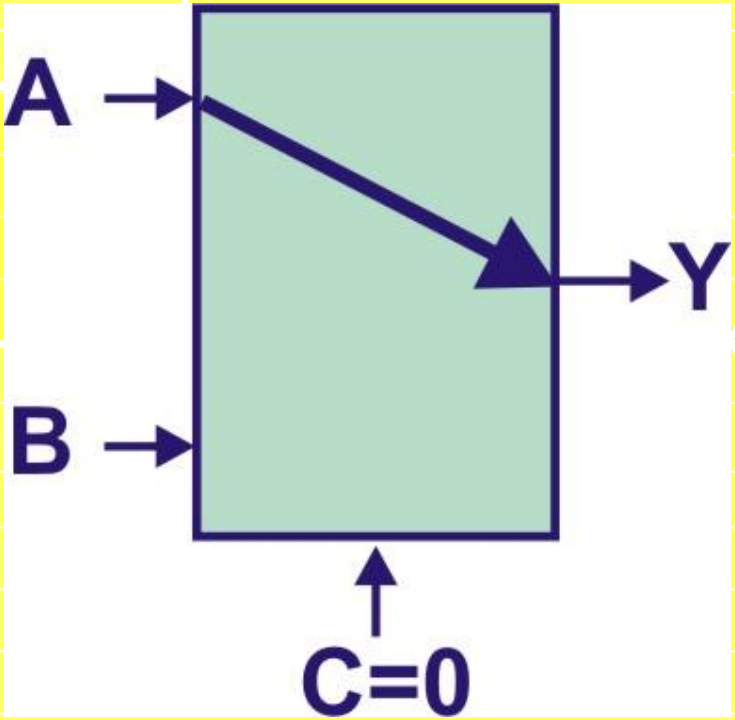
Data Selectors/Multiplexers
2-Line To 1-Line

Dos entradas de datos A y B
Una entrada de control C Una salida Y



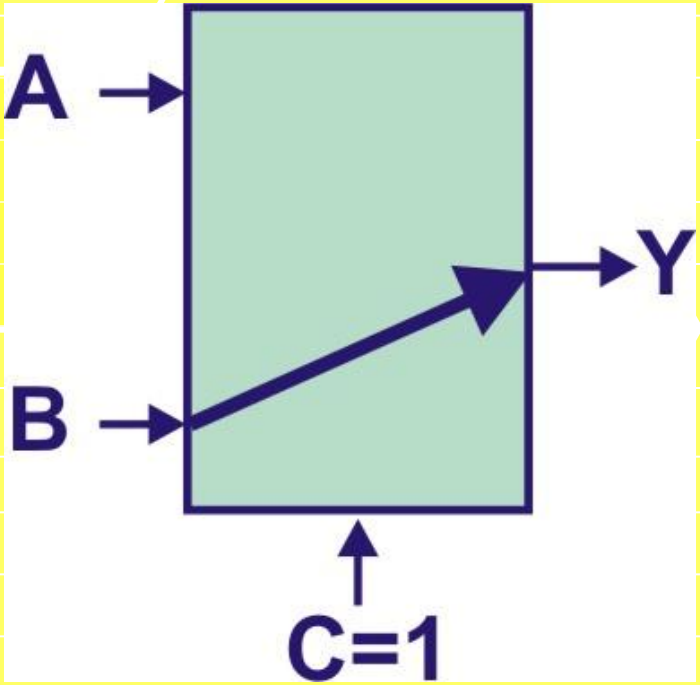
Multiplexor de 2 a 1 línea

Si C=0 entonces la salida Y=A



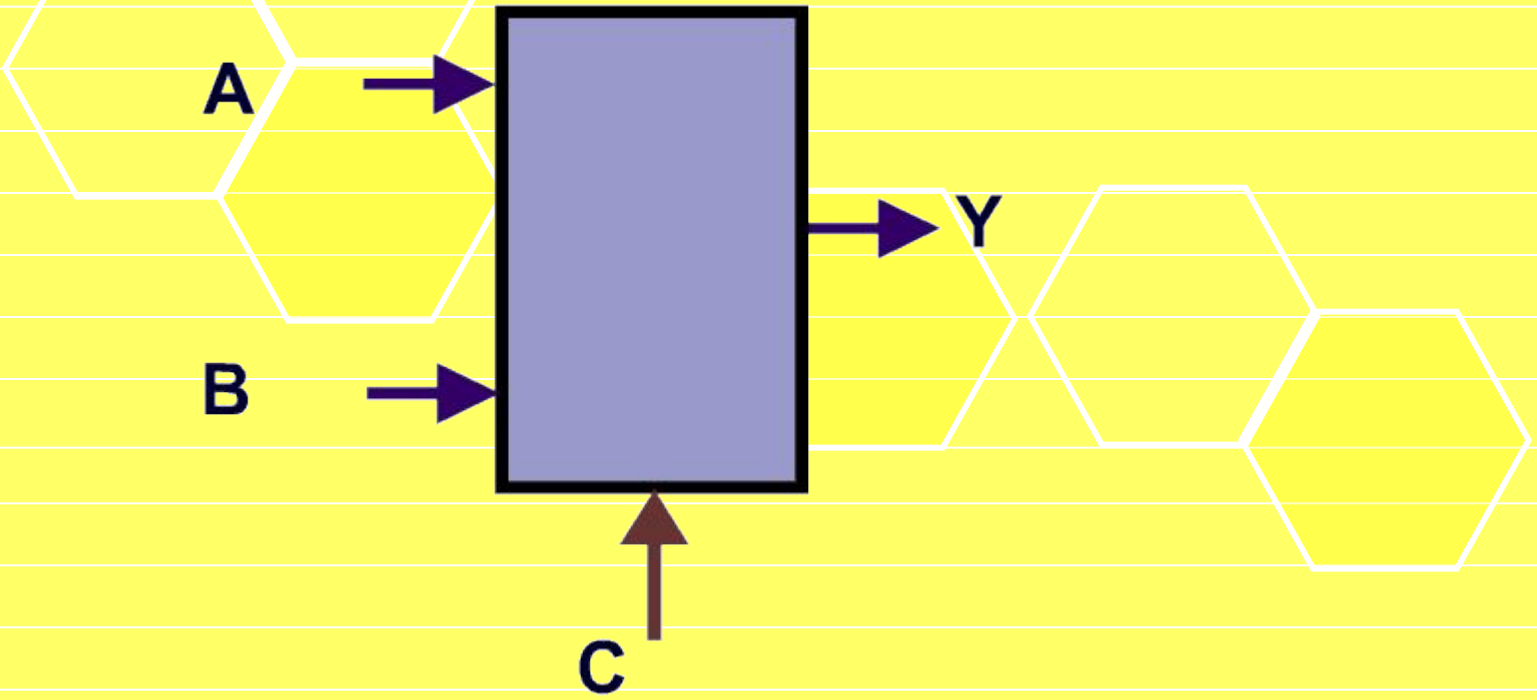
Multiplexor de 2 a 1 línea

Si C=1 entonces la salida Y=B



Multiplexor de 2 a 1 línea

Si C=0 entonces la salida Y=A
Si C=1 entonces la salida Y=B



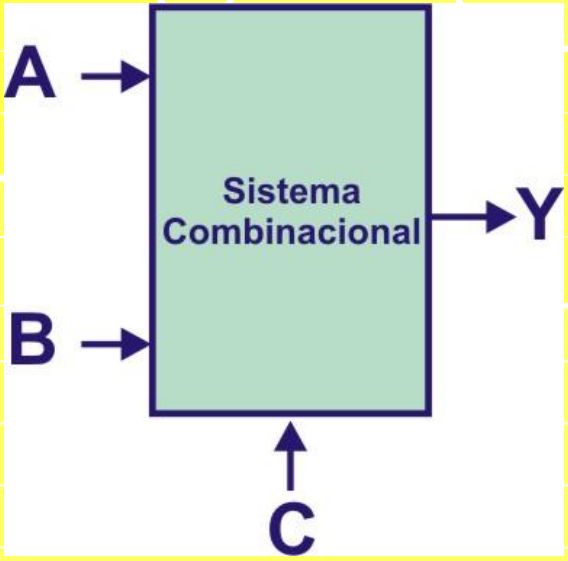
Multiplexor de 2 a 1 línea

1.-Especificar el Sistema

En la redacción del problema está aclarado el propósito y las variables que intervienen en el problema.

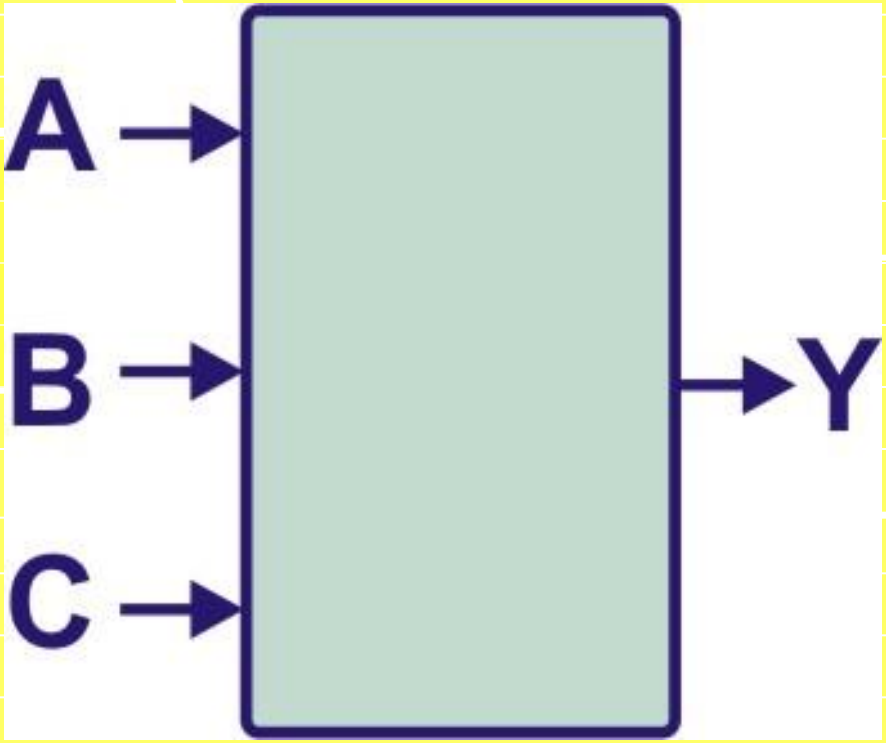
Si C=0 entonces la salida Y=A

Si C=1 entonces la salida Y=B



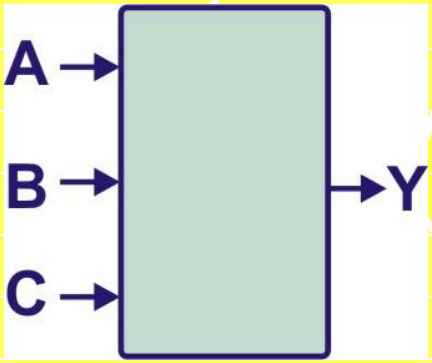
Multiplexor de 2 a 1 línea

2.- Determinar entradas y salidas



Multiplexor de 2 a 1 línea

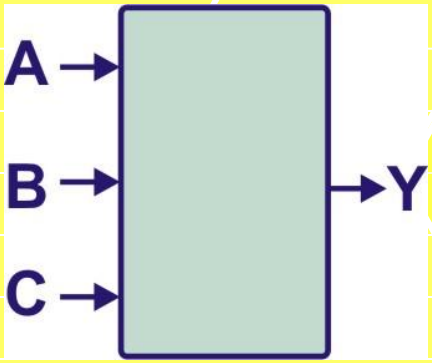
3.- tabla de verdad



m	C	A	B	Y
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

Multiplexor de 2 a 1 línea

3.- tabla de verdad

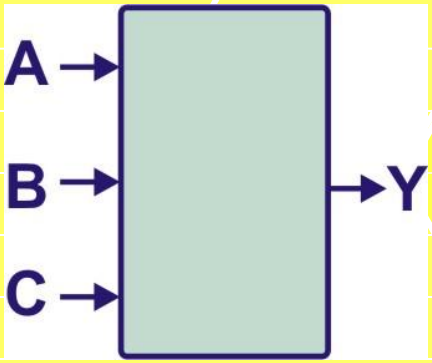


m	C	A	B	Y
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

Multiplexor de 2 a 1 línea

3.- tabla de verdad

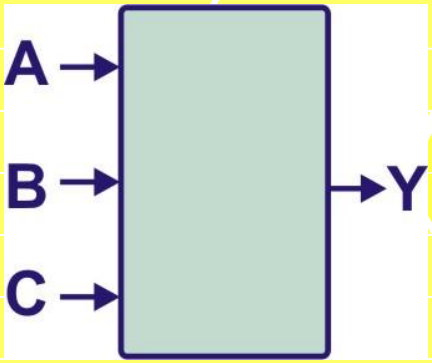
m	C	A	B	Y
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	



C=0,Y=A

Multiplexor de 2 a 1 línea

3.- tabla de verdad



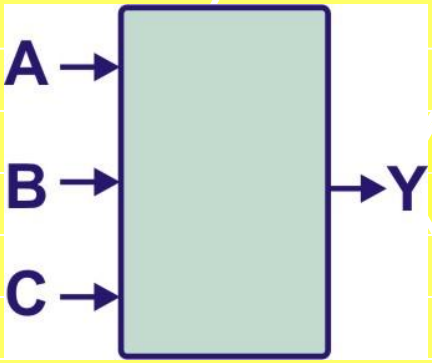
m	C	A	B	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

C=0,Y=A

C=1,Y=B

Multiplexor de 2 a 1 línea

3.- tabla de verdad

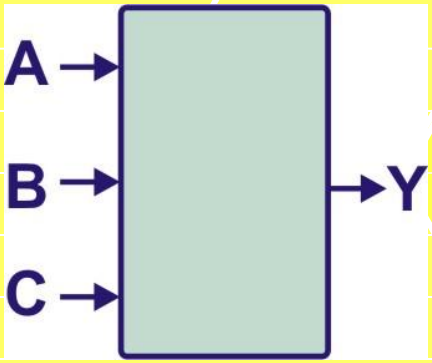


m	C	A	B	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

C=1, Y=B

Multiplexor de 2 a 1 línea

3.- tabla de verdad

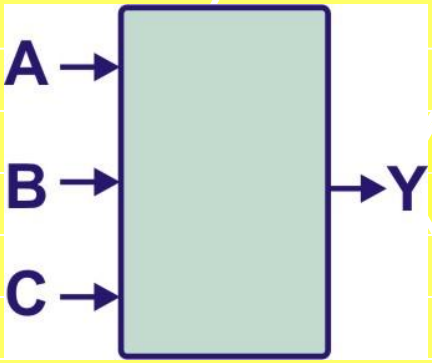


m	C	A	B	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

C=1, Y=B

Multiplexor de 2 a 1 línea

3.- tabla de verdad



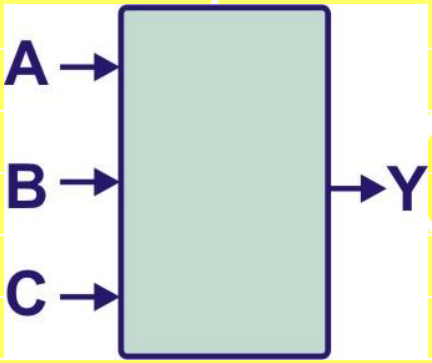
m	C	A	B	Y
0	0	0	X	0
2	0	1	X	1
4	1	X	0	0
7	1	X	1	1

C=1, Y=B

X = Don't Care No importa

3.- tabla de verdad

Multiplexor de 2 a 1 línea

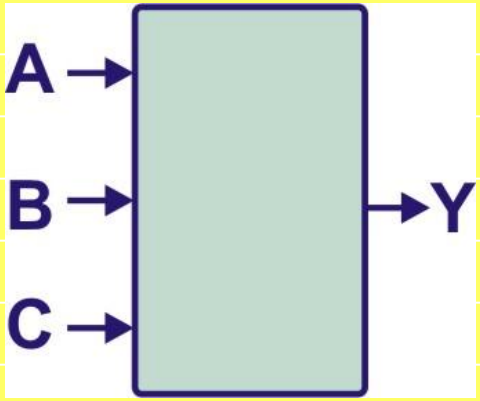


m	C	A	B	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

4.- ecuaciones mínimas

Multiplexor de 2 a 1 línea

m	C	A	B	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1



$$FY_{(C,A,B)} = C'A + CB$$

Terms1

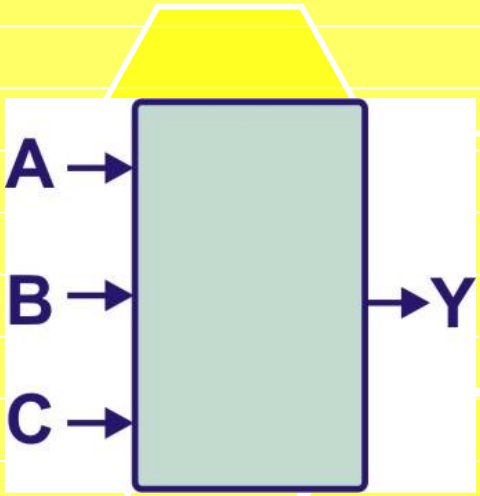
```
Input Variable Names:  C  A  B
Output Function Names:  Y

2 3 5 7.
```

Terms1_O

```
     Simplification Routine:  PI Chart

Y  =  C B + C'A

     Input Cost = 6           Gate Cost  = 3

*************************************************

     Simplification Routine:  PI Chart

Y  =  (C' + B ) (C + A )
```

6.- ABEL-HDL por ecuaciones

Multiplexor de 2 a 1 línea



$$FY_{(C,A,B)} = C'A + CB$$

```
MODULE muxeq
"entradas
A,B,C pin 1,2,3;
"Salida
Y pin 19 istype 'com';
equations
Y= !C&A#C&B;
END
```

Multiplexor de 2 a 1 línea

6.- ABEL-HDL por tabla de Verdad

```
MODULE muxtt
"entradas
A,B,C pin 1,2,3;
"Salida
Y pin 19 istype 'com';
Truth_table
([C,A,B]->Y)
[0,0,0]->0;
[0,0,1]->0;
[0,1,0]->1;
[0,1,1]->1;
[1,0,0]->0;
[1,0,1]->1;
[1,1,0]->0;
[1,1,1]->1;
END
```

m	C	A	B	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Multiplexor de 2 a 1 línea

6.- ABEL-HDL por tabla de Verdad

```
MODULE muxtt
"entradas
A,B,C pin 1,2,3;
"Salida
Y pin 19 istype 'com';
Truth_table
([C,A,B]->Y)
[0,1,0]->1;
[0,1,1]->1;
[1,0,1]->1;
[1,1,1]->1;
END
```

m	C	A	B	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Multiplexor de 2 a 1 línea

6.- Abel-HDL When, Then, Else

WHEN !C THEN Y=A ;

" Si (WHEN) C=0 (!C) entonces (THEN) la salida Y=A

WHEN C THEN Y=B;

" Si (WHEN) C=1 (C) entonces (THEN) la salida Y=B

WHEN !C THEN Y=A else Y=B;

Multiplexor de 2 a 1 línea

6.- Abel-HDL When, Then, Else

```
WHEN C==0 THEN Y=A ;
" Si (WHEN) C=0 (!C) entonces (THEN) la salida Y=A
  WHEN C==1 THEN Y=B;
" Si (WHEN) C=1 (C) entonces (THEN) la salida Y=B
```

```
WHEN C==0 THEN Y=A else Y=B;
```

Multiplexor de 2 a 1 línea

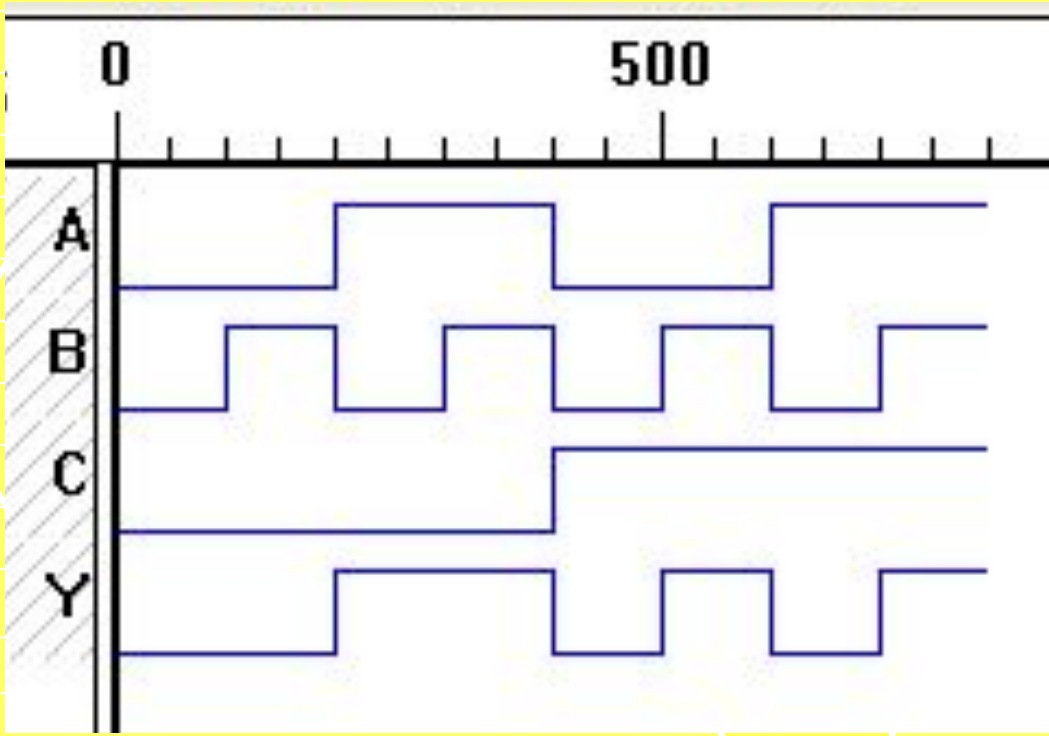
When, Then, Else

```
MODULE muxwte
"entradas
A,B,C pin 1,2,3;
"Salida
Y pin 19 istype 'com';
equations
When !C THEN Y=A else Y=B;
END
```

Multiplexor de 2 a 1 línea

```
MODULE muxwte
"entradas
A,B,C pin 1,2,3;
"Salida
Y pin 19 istype 'com';
equations
WHEN !C THEN Y=A else
Y=B;
Test_vectors
([C,A,B]->Y)
[0,0,0]->0;
[0,0,1]->0;
[0,1,0]->1;
[0,1,1]->1;
[1,0,0]->0;
[1,0,1]->1;
[1,1,0]->0;
[1,1,1]->1;
END
```

Simulación

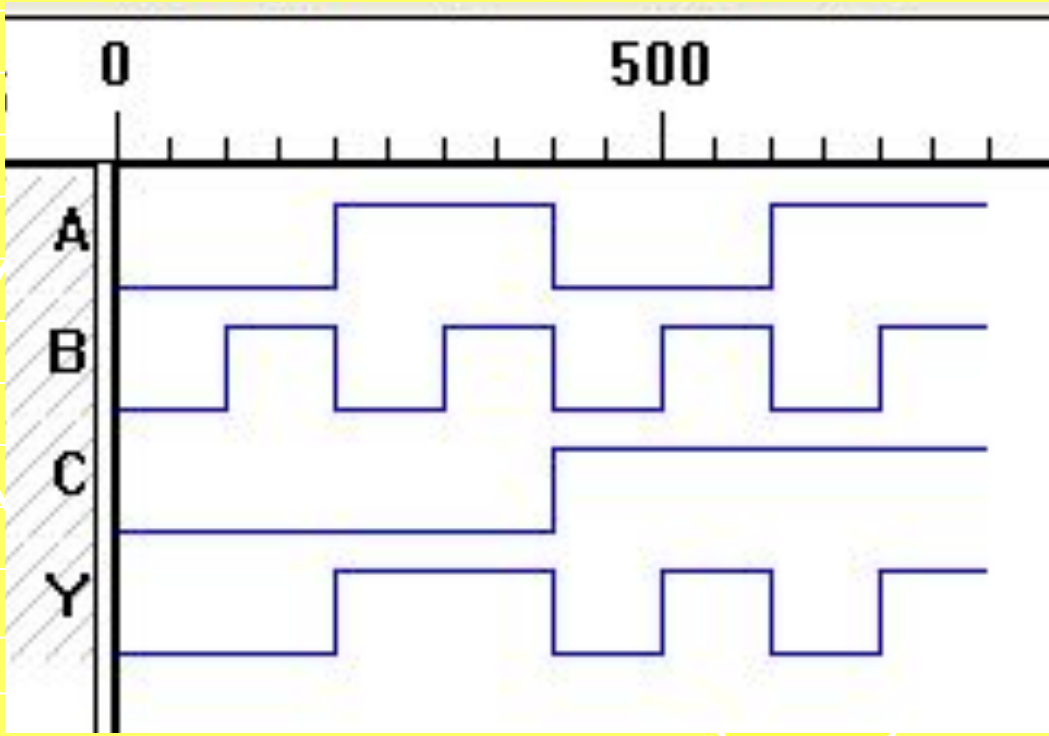


Multiplexor de 2 a 1 línea

Simulación

```
MODULE muxwte
"entradas
A,B,C pin 1,2,3;
"Salida
Y pin 19 istype 'com';
equations
WHEN !C THEN Y=A else
Y=B;
Test_vectors
([C,A,B]->Y)
[0,0,.x.]->0;
[0,1,.x.]->1;
[1,.x.,0]->0;
[1,.x.,1]->1;
END
```

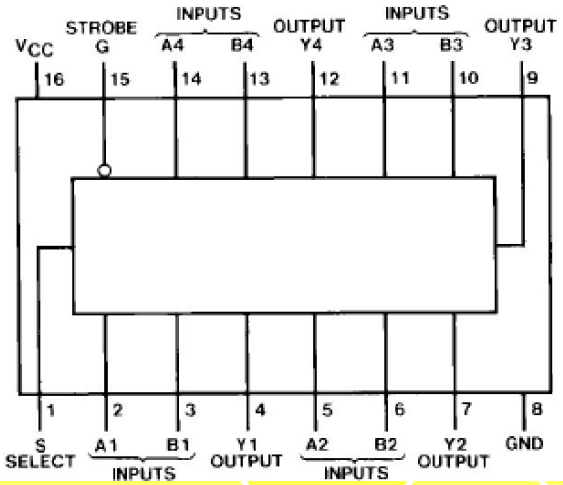
```
([C,A,B]->Y)
[0,0,0]->0;
[0,0,1]->0;
[0,1,0]->1;
[0,1,1]->1;
[1,0,0]->0;
[1,0,1]->1;
[1,1,0]->0;
[1,1,1]->1;
```



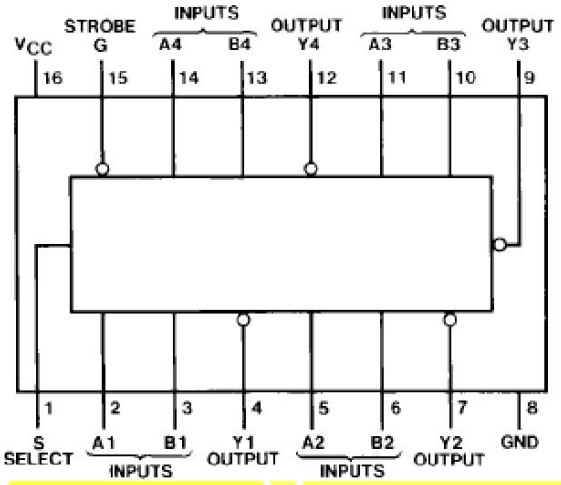
54LS157/DM54LS157/DM74LS157, 54LS158/DM54LS158/DM74LS158 Quad 2-Line to 1-Line Data Selectors/Multiplexers

Connection Diagrams

Dual-In-Line Package



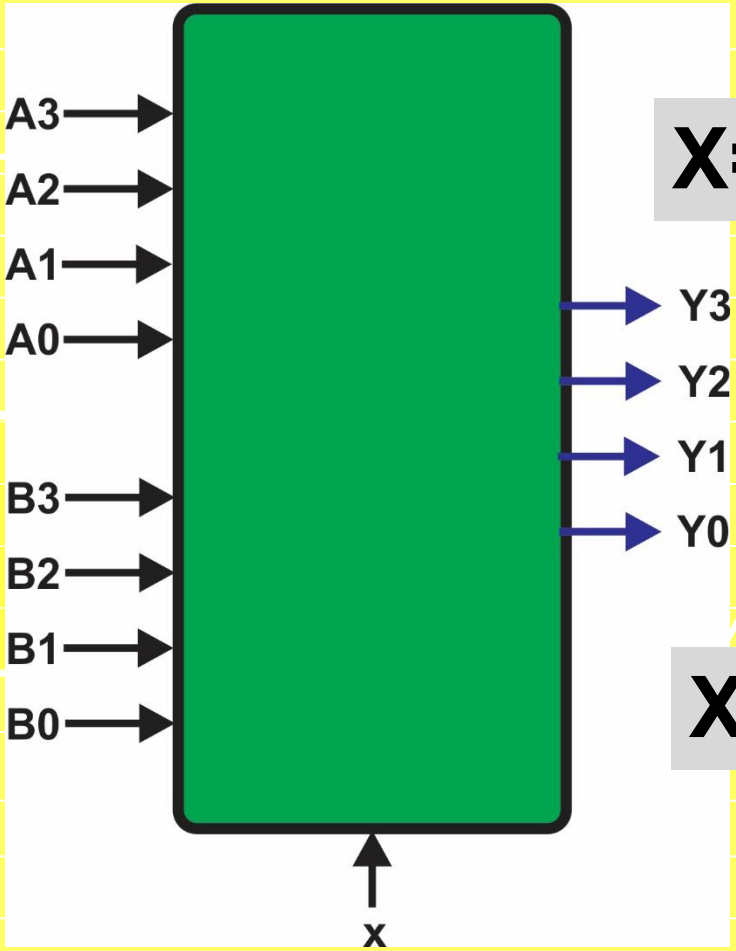
Dual-In-Line Package



Inputs				Output Y	
Strobe	Select	A	B	LS157	LS158
H	X	X	X	L	H
L	L	L	X	L	H
L	L	H	X	H	L
L	H	X	L	L	H
L	H	X	H	H	L

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

Multiplexor de 2 a 1 (4 bits)



A= [A3, A2, A1, A0]
de 0 a 15

B= [B3, B2, B1, B0]
de 0 a 15

X=1 Y=A

Y= [Y3, Y2, Y1, Y0]
de 0 a 15

X=0 Y=B

9 entradas 512 combinaciones

Multiplexor de 2 a 1 línea de 4 bits

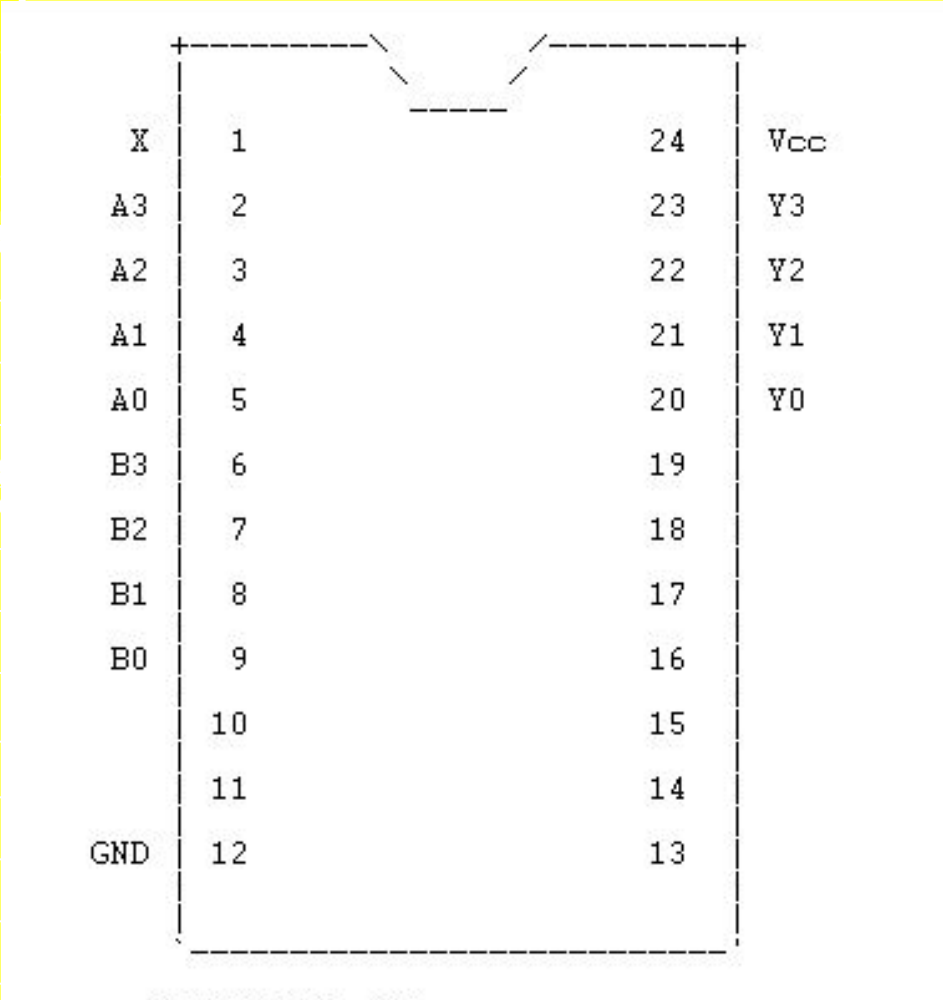
```
MODULE MUX
X,A3..A0,B3..B0 PIN 1..9;
Y3..Y0 PIN 23..20 ISTYPE 'COM';
A=[A3,A2,A1,A0];
B=[B3,B2,B1,B0];
Y=[Y3..Y0];
EQUATIONS
WHEN X THEN Y=A ELSE Y=B;
```

```
Test_vectors
[X,A,B]->Y)
[0,.x.,0]->.x.;
[0,.x.,1]->.x.;
[0,.x.,2]->.x.;
[0,.x.,3]->.x.;
[0,.x.,4]->.x.;
[0,.x.,5]->.x.;
[0,.x.,6]->.x.;
[0,.x.,7]->.x.;
[0,.x.,8]->.x.;
[0,.x.,9]->.x.;
[0,.x.,10]->.x.;
[0,.x.,11]->.x.;
[0,.x.,12]->.x.;
[0,.x.,13]->.x.;
[0,.x.,14]->.x.;
[0,.x.,15]->.x.;
[1,0,.x.]->.x.;
[1,1,.x.]->.x.;
[1,2,.x.]->.x.;
[1,3,.x.]->.x.;
[1,4,.x.]->.x.;
[1,5,.x.]->.x.;
[1,6,.x.]->.x.;
[1,7,.x.]->.x.;
[1,8,.x.]->.x.;
[1,9,.x.]->.x.;
[1,10,.x.]->.x.;
[1,11,.x.]->.x.;
[1,12,.x.]->.x.;
[1,13,.x.]->.x.;
[1,14,.x.]->.x.;
[1,15,.x.]->.x.;
END
```

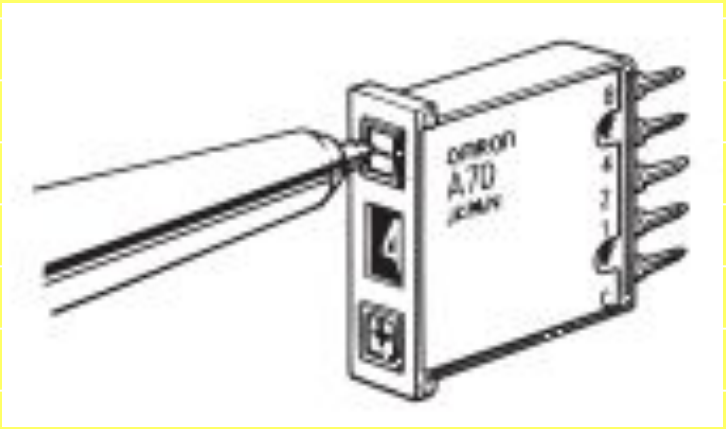
```
[X,A3,A2,A1,A0,B3,B2,B1,B0]->[Y3,Y2,Y1,Y0]
[0,.X.,.X.,.X.,.X.,0,0,0,0]->[.X.,.X.,.X.,.X.]:
```

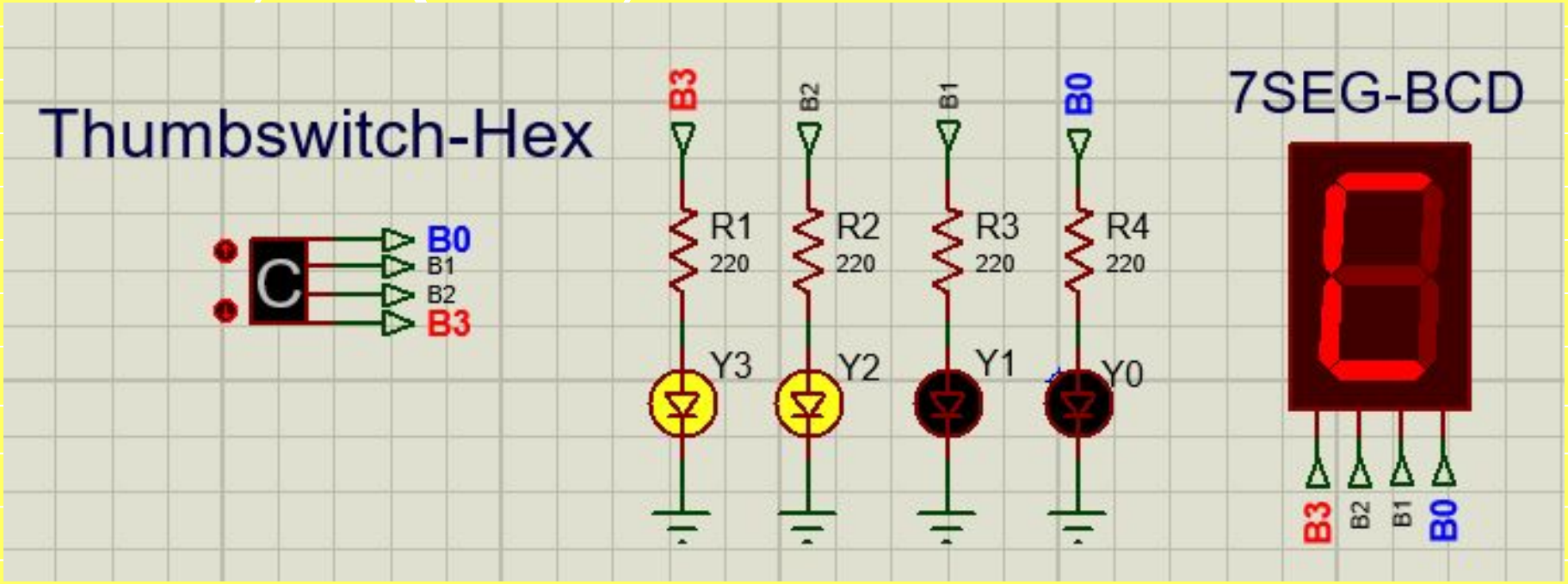
Multiplexor de 2 a 1 línea de 4 bits

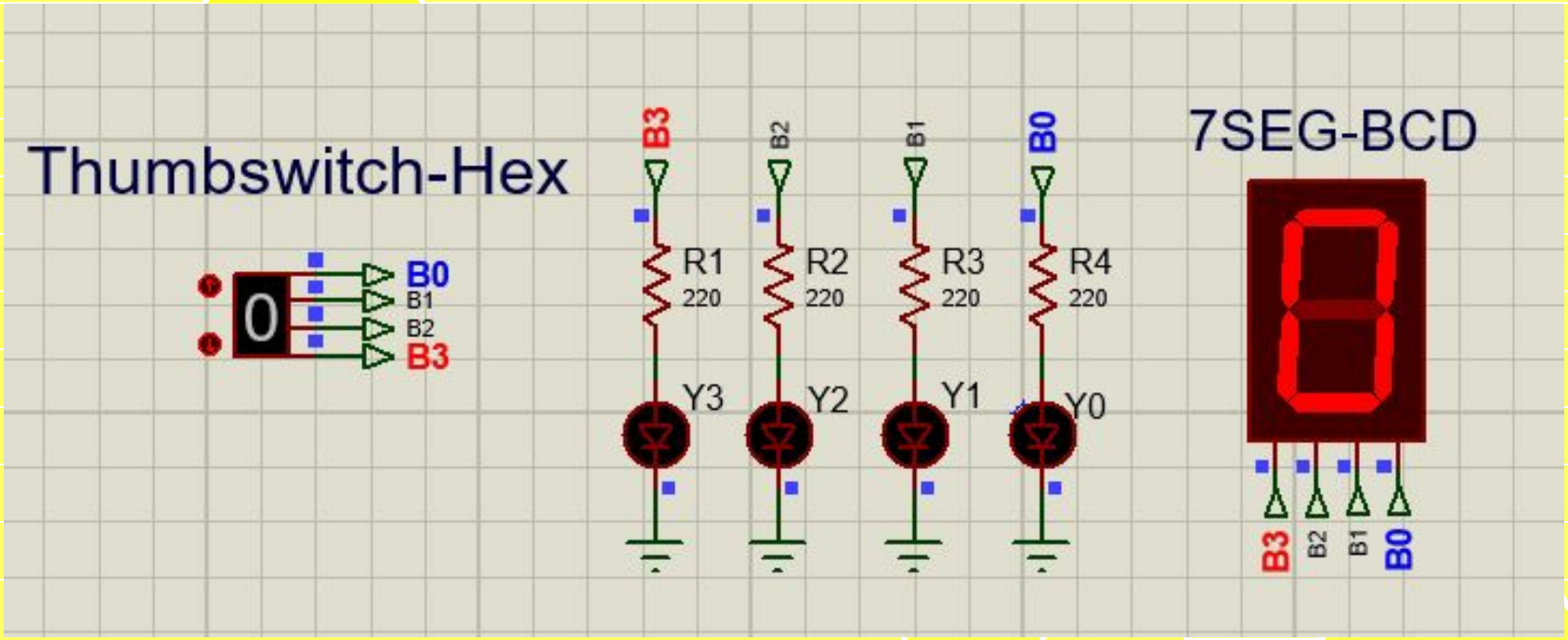
Distribución de terminales (pin Out)

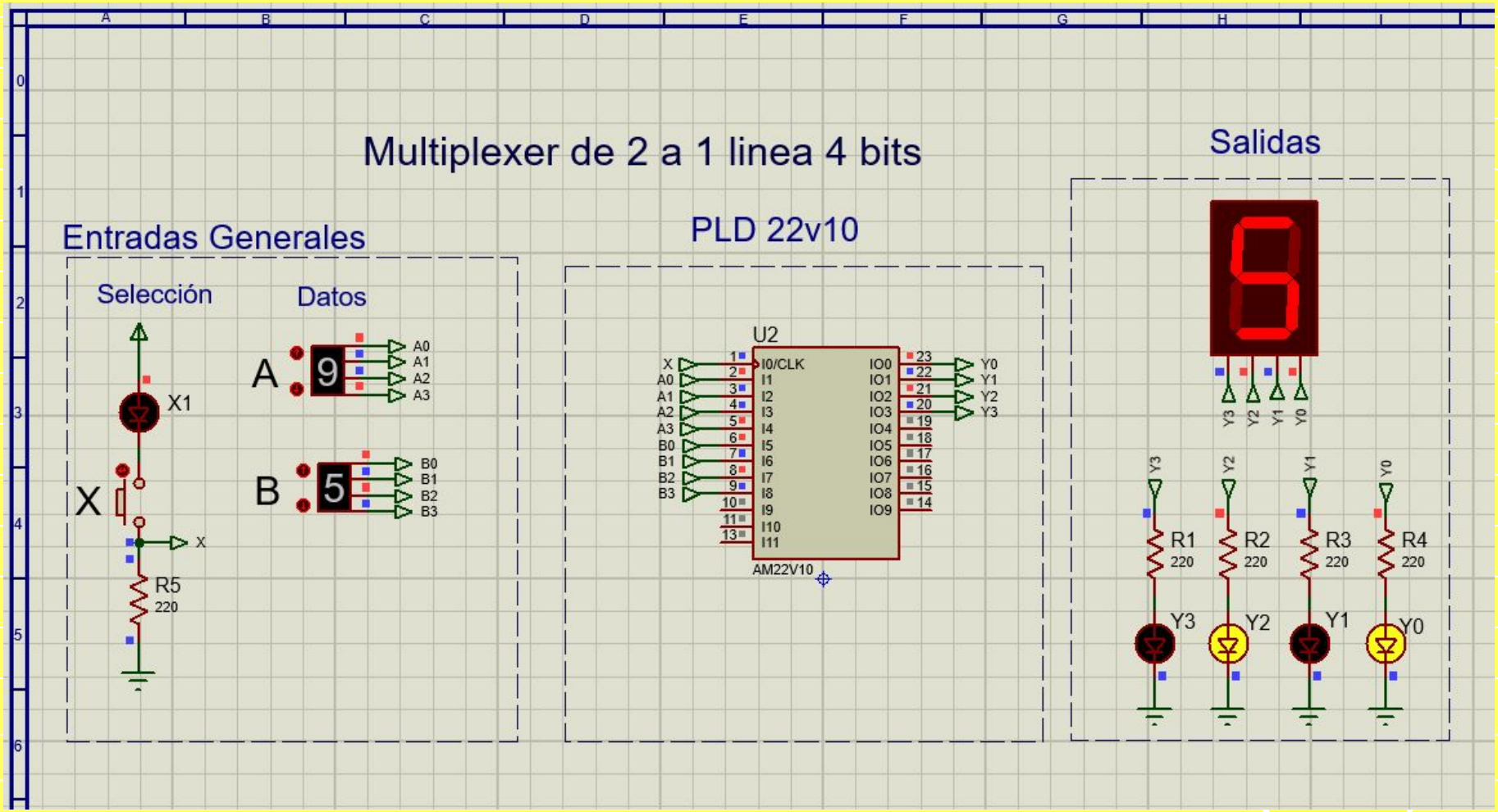


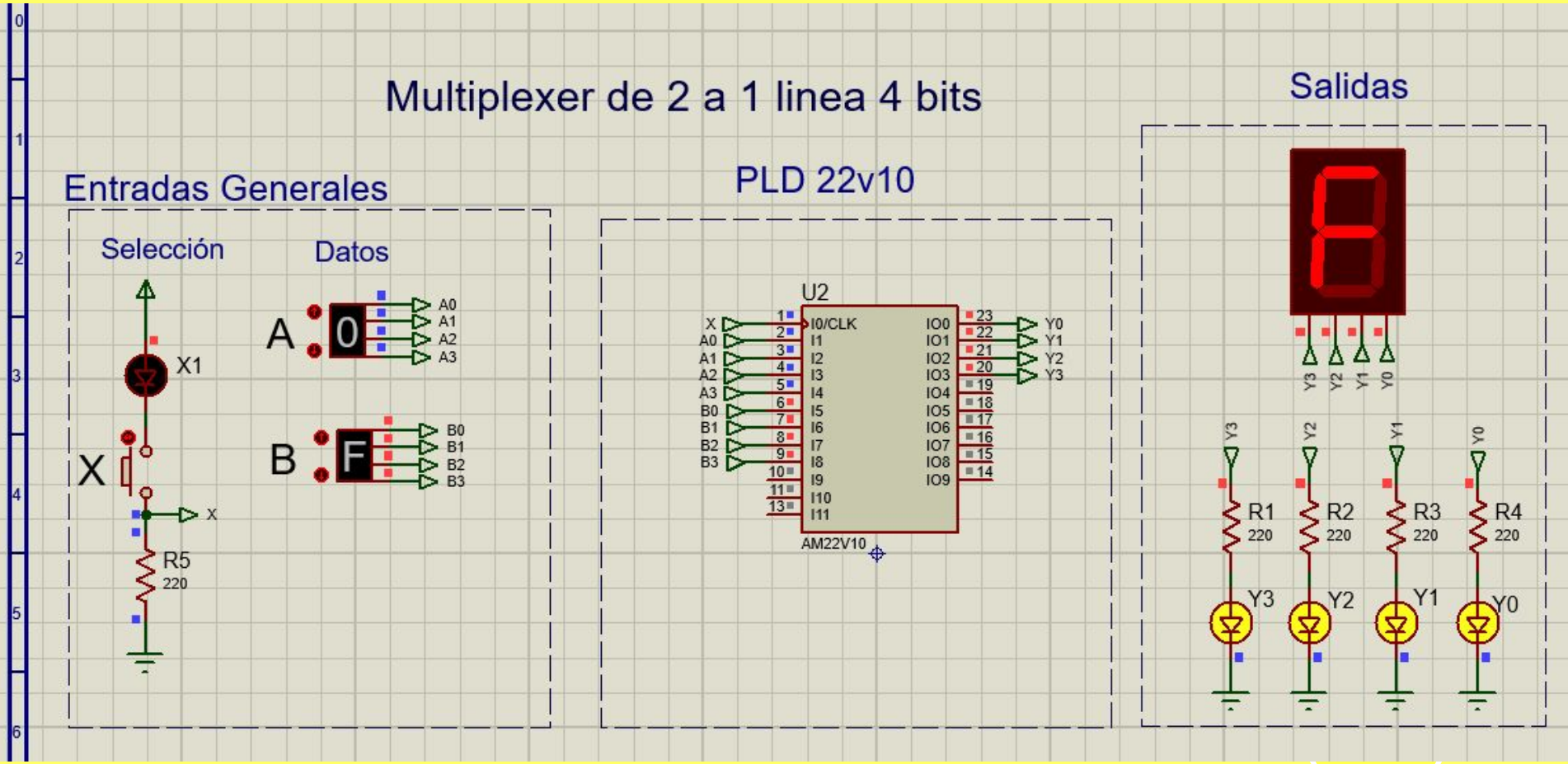
ThumbWell Switch
Interruptores con Selector de Rueda





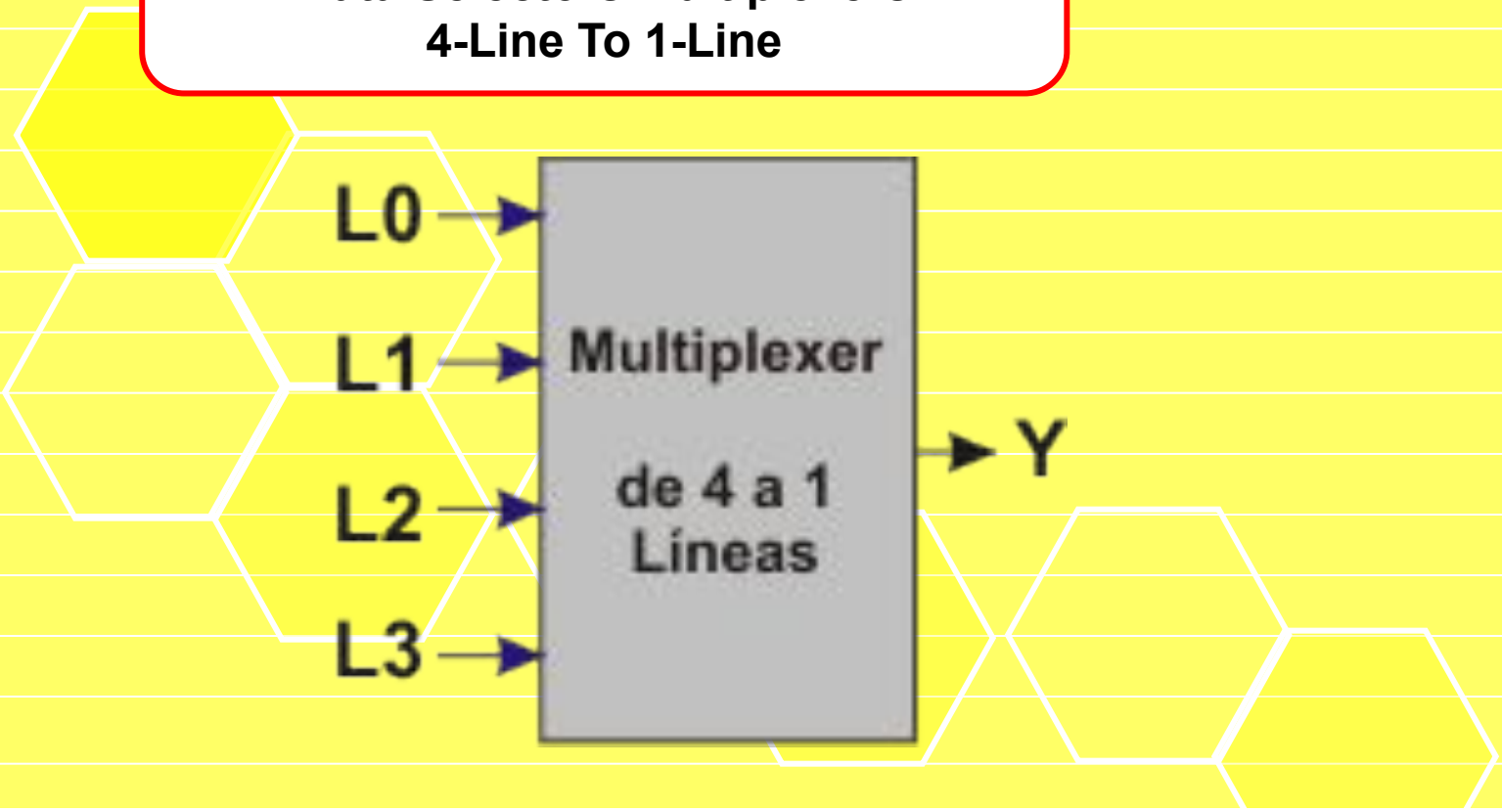

Thumbswitch-Hex
0
B0
B1
B2
B3
B3
B2
B1
B0
R1
220
R2
220
R3
220
R4
220
Y3
Y2
Y1
Y0
7SEG-BCD
B3
B2
B1
B0


Multiplexer de 2 a 1 linea 4 bits
Entradas Generales
Selección
Datos
X1
X
R5
220
A
9
A0
A1
A2
A3
B
5
B0
B1
B2
B3
PLD 22v10
U2
X
A0
A1
A2
A3
B0
B1
B2
B3
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
Y0
Y1
Y2
Y3
AM22V10
Salidas
R1
R2
R3
R4
220
Y3
Y2
Y1
Y0


Multiplexer de 2 a 1 linea 4 bits
Entradas Generales
Selección
Datos
X1
X
R5
220
A
0
B
F
A0
A1
A2
A3
B0
B1
B2
B3
PLD 22v10
U2
X
A0
A1
A2
A3
B0
B1
B2
B3
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
Y0
Y1
Y2
Y3
AM22V10
Salidas
R1
R2
R3
R4
220
Y3
Y2
Y1
Y0

Multiplexor de 4 a 1 línea

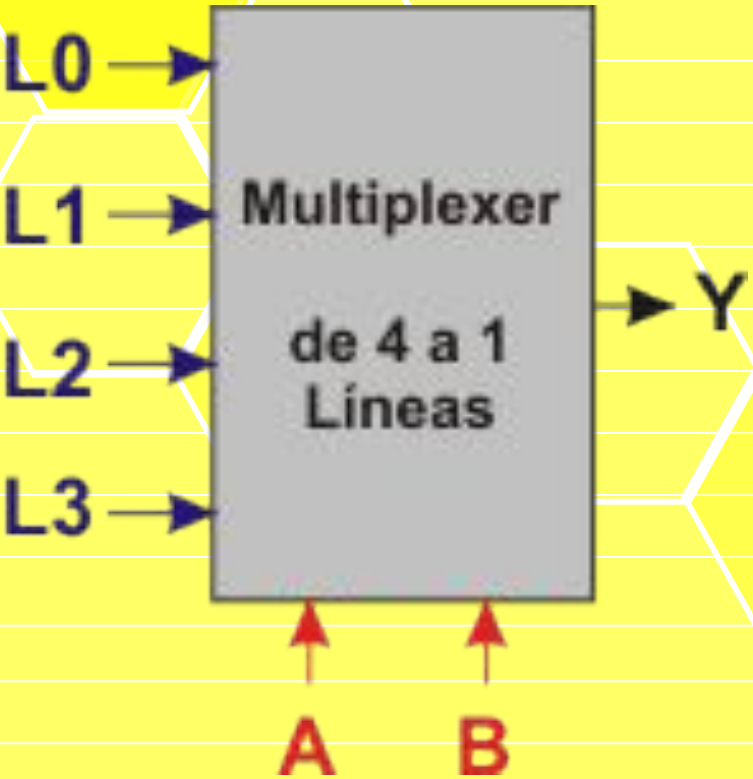
Data Selectors/Multiplexers
4-Line To 1-Line



- Cuantas entradas mínimo de control se requieren para seleccionar cada una de las líneas

Multiplexor de 4 a 1 línea

Data Selectors/Multiplexers
4-Line To 1-Line

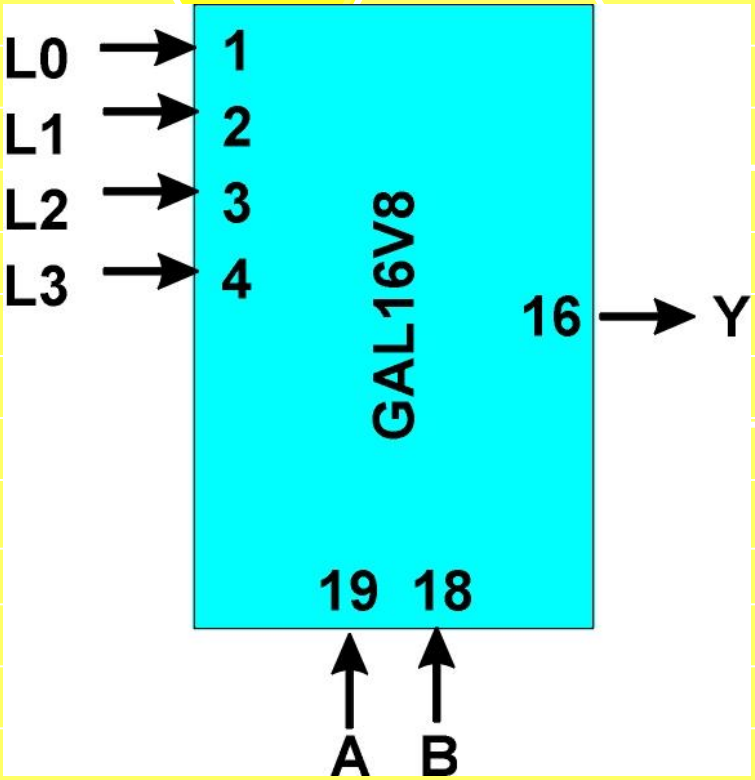


m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

When, Then, Else

Multiplexor de 4 a 1 línea

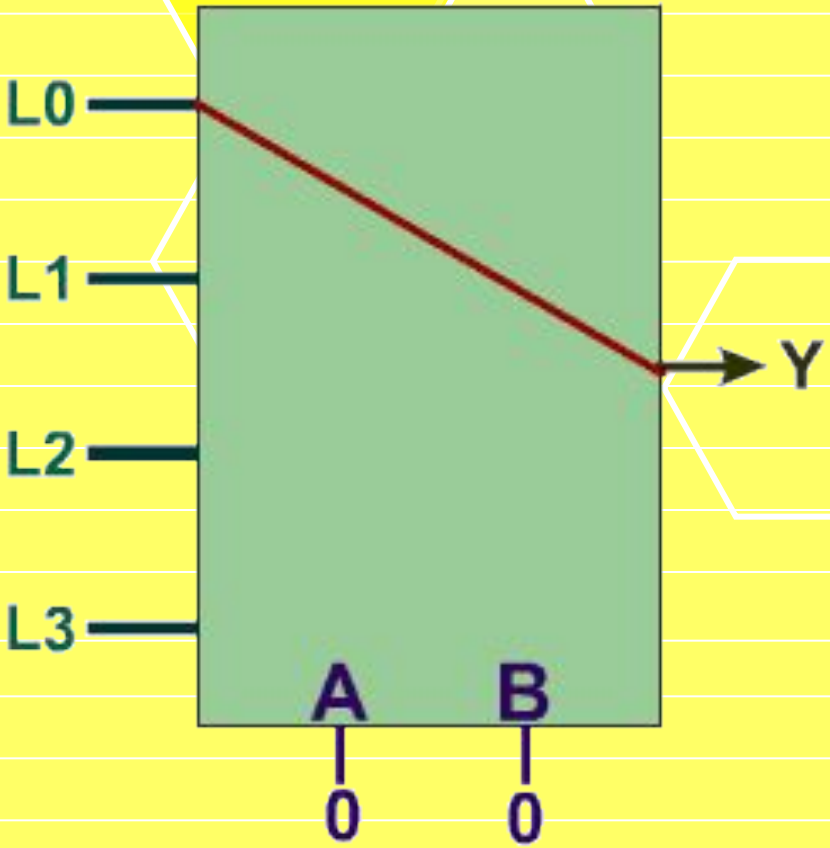
Data Selectors/Multiplexers
4-Line To 1-Line



m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

Multiplexor de 4 a 1 línea

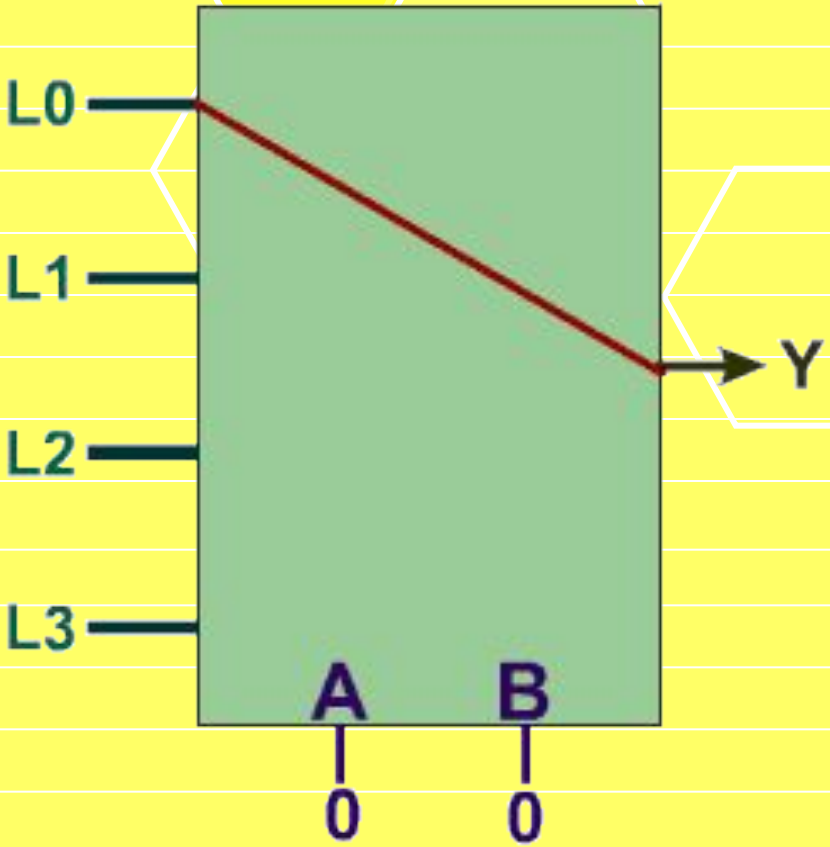
Cuantas entradas se tienen en total ?



m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

Multiplexor de 4 a 1 línea

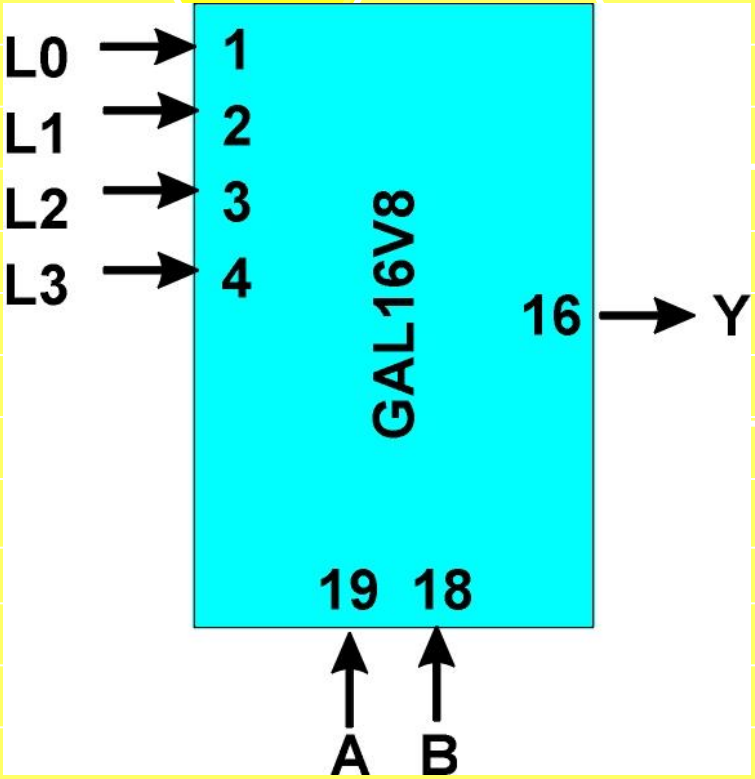
Cuantas combinaciones se pueden generar ?



m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

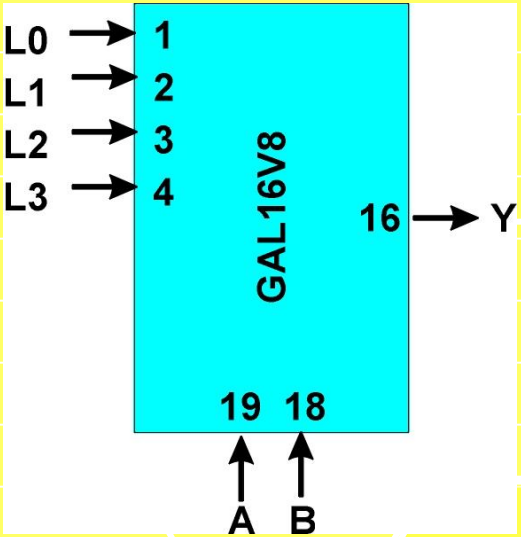
Multiplexor de 4 a 1 línea

Elabore el archivo en ABEL-HDL usando los comandos When, Then, Else



m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

Multiplexor de 4 a 1 línea

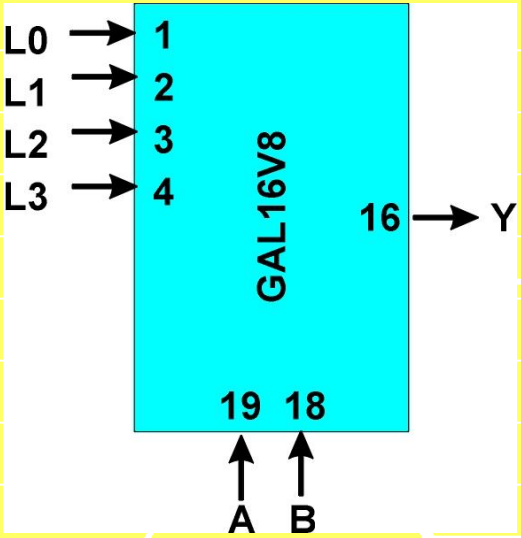


When, Then, Else

```
MODULE muxeq
"entradas
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';



END
```

m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

Multiplexor de 4 a 1 línea

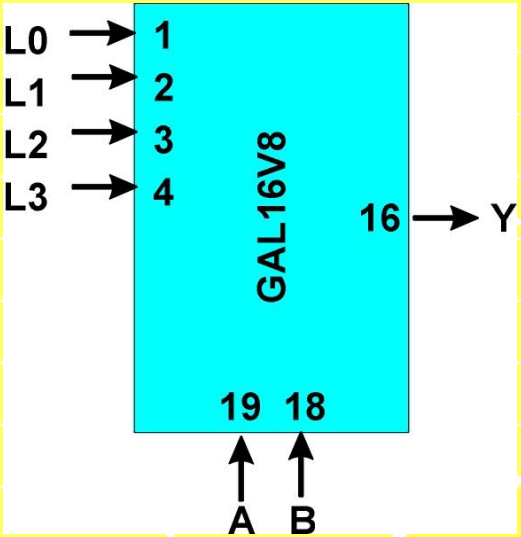


m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

When, Then, Else

```
MODULE muxeq
"entradas
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';
equations
WHEN



END
```

Multiplexor de 4 a 1 línea

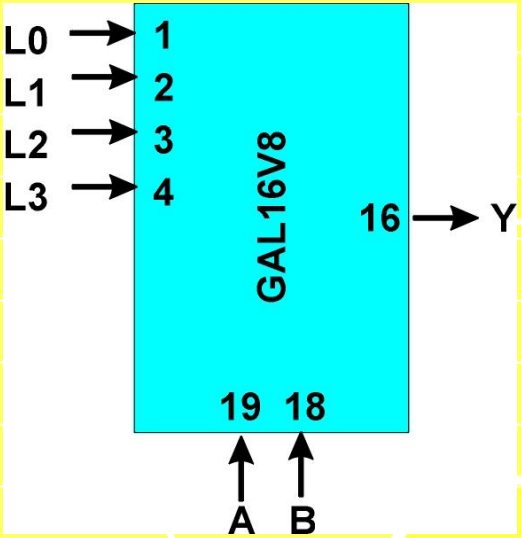


When, Then, Else

```
MODULE muxeq
"entradas
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';
equations
WHEN          THEN Y=L0;




END
```

m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

Multiplexor de 4 a 1 línea

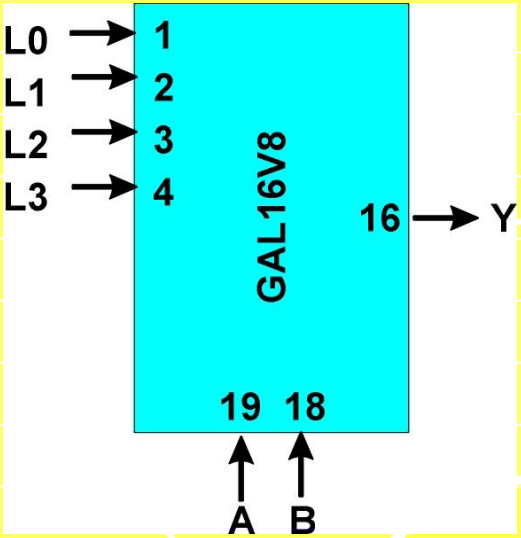


When, Then, Else

```
MODULE muxeq
"entradas
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';
equations
WHEN !A&!B THEN Y=L0;


END
```

m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

Multiplexor de 4 a 1 línea



When, Then, Else

```
MODULE muxeq
"entradas
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';
equations
WHEN !A&!B THEN Y=L0;
WHEN  !A&B THEN Y=L1;
WHEN  A&!B THEN Y=L2;
WHEN   A&B THEN Y=L3;
END
```

m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

Multiplexor de 4 a 1 línea

Para no listar las 64 combinaciones Usamos el Don´t Care .X.

```
MODULE muxeq
"entradas
X=.x.;
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';
equations
WHEN !A&!B THEN Y=L0;
WHEN  !A&B THEN Y=L1;
WHEN  A&!B THEN Y=L2;
WHEN   A&B THEN Y=L3;
END
```

```
X=.x.;
Test_vectors
([A,B,L3,L2,L1,L0]->[Y])
  [0,0, X, X, X, 0 ]-> [X];
  [0,0, X, X, X, 1 ]-> [X];
  [0,1, X, X, 0, X ]-> [X];
  [0,1, X, X, 1, X ]-> [X];
  [1,0, X, 0, X, X ]-> [X];
  [1,0, X, 1, X, X ]-> [X];
  [1,1, 0, X, X, X ]-> [X];
  [1,1, 1, X, X, X ]-> [X];
```

Multiplexor de 4 a 1 línea

Para no listar las 64 combinaciones Usamos el Don´t Care .X.

```
MODULE muxeq
"entradas
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';
equations
WHEN !A&!B THEN Y=L0;
WHEN  !A&B THEN Y=L1;
WHEN  A&!B THEN Y=L2;
WHEN   A&B THEN Y=L3;
END
```

```
X=.x.;
Test_vectors
([A,B,L3,L2,L1,L0]->[Y])
  [0,0, X, X, X, 0 ]-> [0];
  [0,0, X, X, X, 1 ]-> [1];
  [0,1, X, X, 0, X ]-> [0];
  [0,1, X, X, 1, X ]-> [1];
  [1,0, X, 0, X, X ]-> [0];
  [1,0, X, 1, X, X ]-> [1];
  [1,1, 0, X, X, X ]-> [0];
  [1,1, 1, X, X, X ]-> [1];
```

Multiplexor de 4 a 1 línea

Para no listar las 64 combinaciones Usamos el Don´t Care .X.

```
MODULE muxeq
"entradas
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';
equations
WHEN !A&!B THEN Y=L0;
WHEN  !A&B THEN Y=L1;
WHEN  A&!B THEN Y=L2;
WHEN   A&B THEN Y=L3;
END
```

```
X=.x.;
Test_vectors
([A,B,L3,L2,L1,L0]->[Y])
  [0,0, X, X, X, 0 ]-> [0];
  [0,0, X, X, X, 1 ]-> [1];
  [0,1, X, X, 0, X ]-> [0];
  [0,1, X, X, 1, X ]-> [1];
  [1,0, X, 0, X, X ]-> [0];
  [1,0, X, 1, X, X ]-> [1];
  [1,1, 0, X, X, X ]-> [0];
  [1,1, 1, X, X, X ]-> [1];
```

Multiplexor de 4 a 1 línea

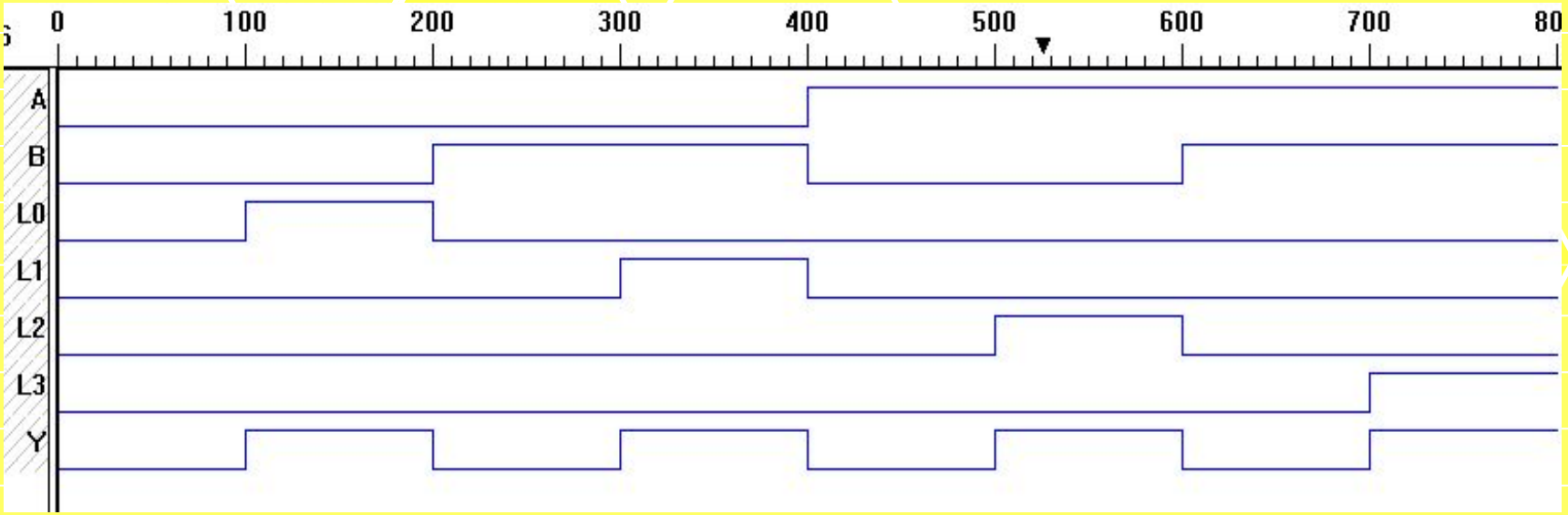
Para no listar las 64 combinaciones Usamos el Don´t Care .X.

```
MODULE muxeq
"entradas
A,B,L0..L3 pin 1,2,4..7;
"Salida
Y pin 14 istype 'com';
S=[A,B];
equations
WHEN S==0 THEN Y=L0;
WHEN S==1 THEN Y=L1;
WHEN S==2 THEN Y=L2;
WHEN S==3 THEN Y=L3;
END
```

```
X=.x.;
Test_vectors
([S,L3,L2,L1,L0]->[Y])
  [0, X, X, X, 0 ]-> [0];
  [0, X, X, X, 1 ]-> [1];
  [1, X, X, 0, X ]-> [0];
  [1, X, X, 1, X ]-> [1];
  [2, X, 0, X, X ]-> [0];
  [2, X, 1, X, X ]-> [1];
  [3, 0, X, X, X ]-> [0];
  [3, 1, X, X, X ]-> [1];
```

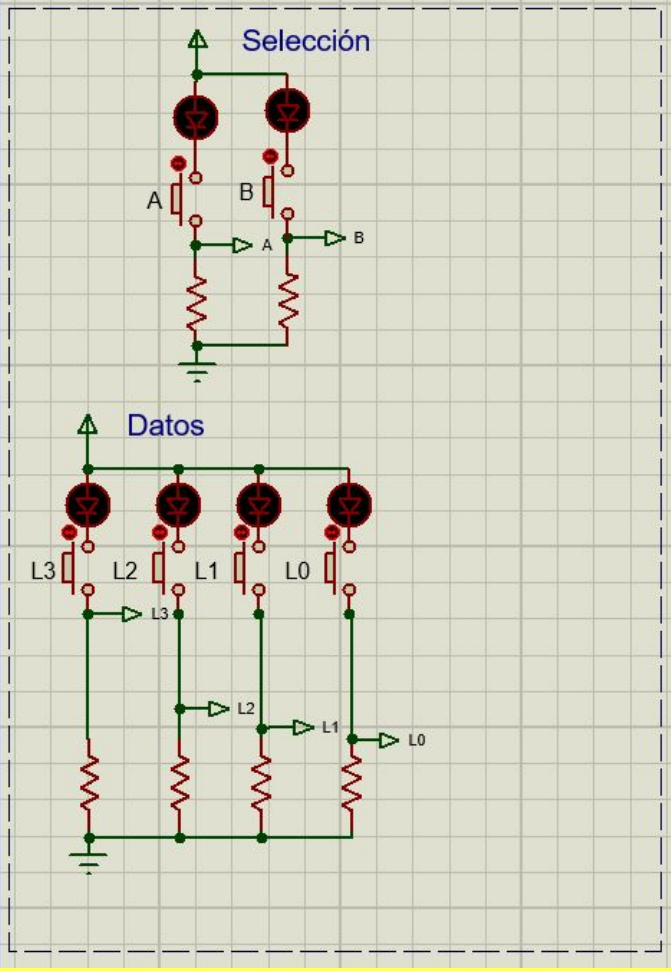
```
MODULE mux
X=.x.;
"entradas
A,B,L0..L3 pin 19,18,1..4;
"Salida
Y pin 16 istype 'com';
equations
WHEN !A&!B THEN Y=L0;
WHEN  !A&B THEN Y=L1;
WHEN  A&!B THEN Y=L2;
WHEN   A&B THEN Y=L3;
```

```
Test_vectors
([A,B,L3,L2,L1,L0]->[Y])
[0,0,X,X,X,0]->[0];
[0,0,X,X,X,1]->[1];
[0,1,X,X,0,X]->[0];
[0,1,X,X,1,X]->[1];
[1,0,X,0,X,X]->[0];
[1,0,X,1,X,X]->[1];
[1,1,0,X,X,X]->[0];
[1,1,1,X,X,X]->[1];
END
```

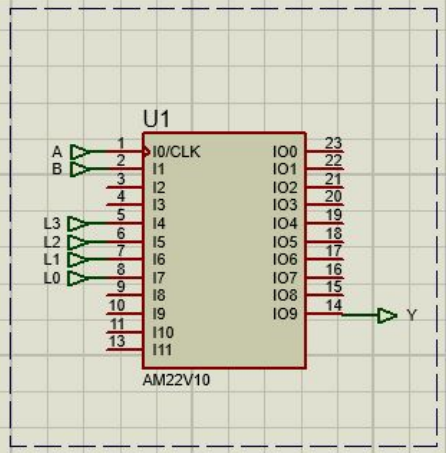


Multiplexer de 4 a 1 linea 1 bit

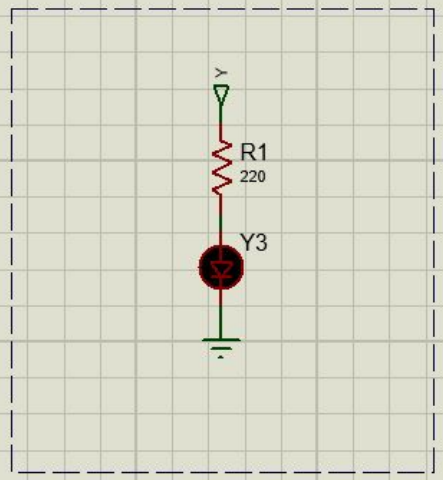
Entradas Generales



PLD 22v10



Salida

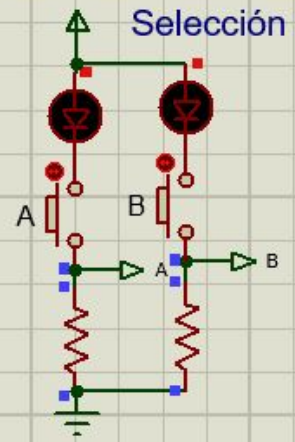


DESIGN TITLE: mux de 4 a 1 1bn.pdsprj
SHEET TITLE: MUx 4 a 1 1 bits
DATE: 06/10/2022
REV: 1.0
BY: JAGG
PAGE: 1 of 1

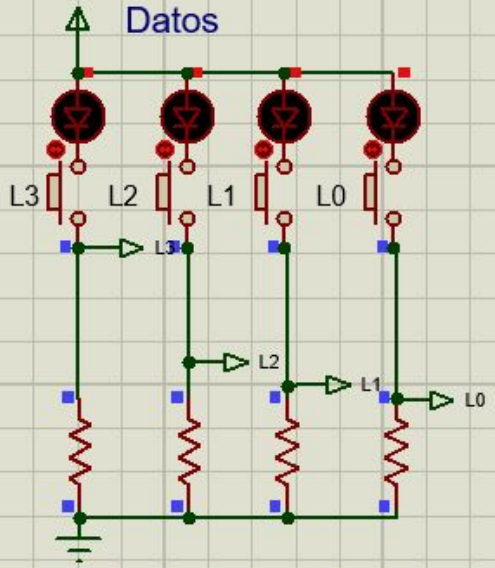
Multiplexer de 4 a 1 linea 1 bit

Entradas Generales

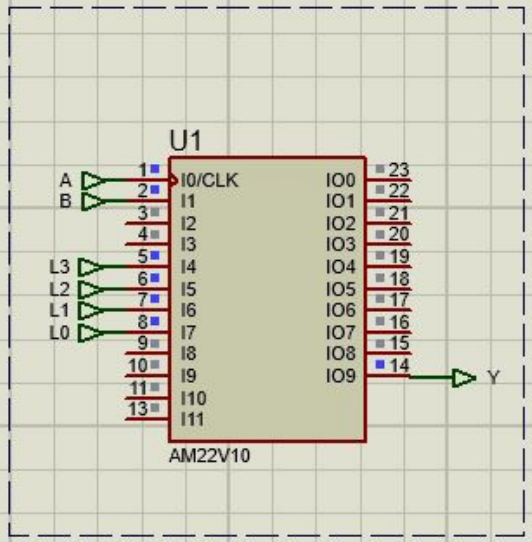
Selección



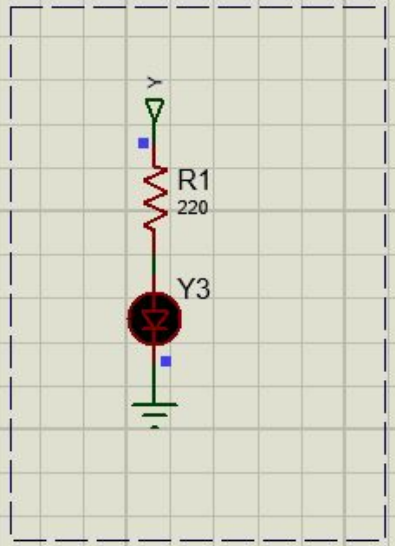
Datos



PLD 22v10



Salida



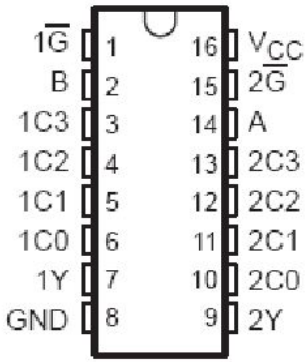
DESIGN TITLE: mux de 4 a 1 1bn.p
SHEET TITLE: MUx 4 a 1 1 bits

SN54HC153, SN74HC153
DUAL 4-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

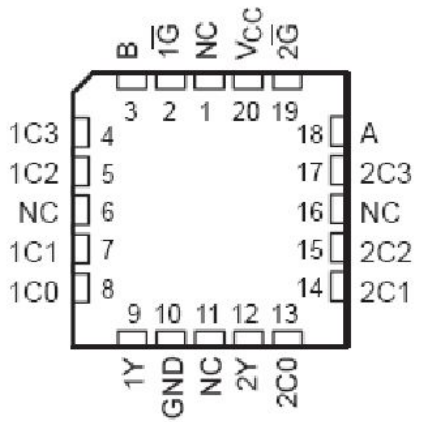
SCLS112D – DECEMBER 1982 – REVISED OCTOBER 2003

- Wide Operating Voltage Range of 2 V to 6 V
- Outputs Can Drive Up To 15 LSTTL Loads
- Low Power Consumption, 80-μA Max I_{CC}
- Typical t_{pd} = 9 ns
- ±6-mA Output Drive at 5 V
- Low Input Current of 1 μA Max
- Permit Multiplexing from n Lines to One Line
- Perform Parallel-to-Serial Conversion
- Strobe (Enable) Line Provided for Cascading (N Lines to n Lines)

SN54HC153 . . . J OR W PACKAGE
SN74HC153 . . . D, N, NS, OR PW PACKAGE
(TOP VIEW)

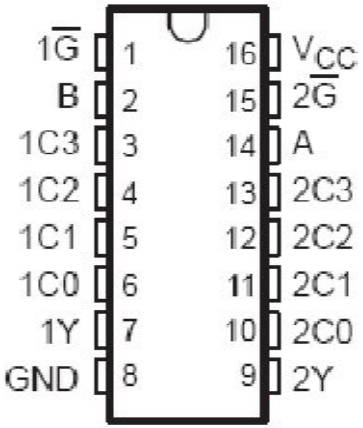


SN54HC153 . . . FK PACKAGE
(TOP VIEW)



SN54HC153, SN74HC153
DUAL 4-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

SN54HC153 . . . J OR W PACKAGE
SN74HC153 . . . D, N, NS, OR PW PACKAGE
(TOP VIEW)



FUNCTION TABLE

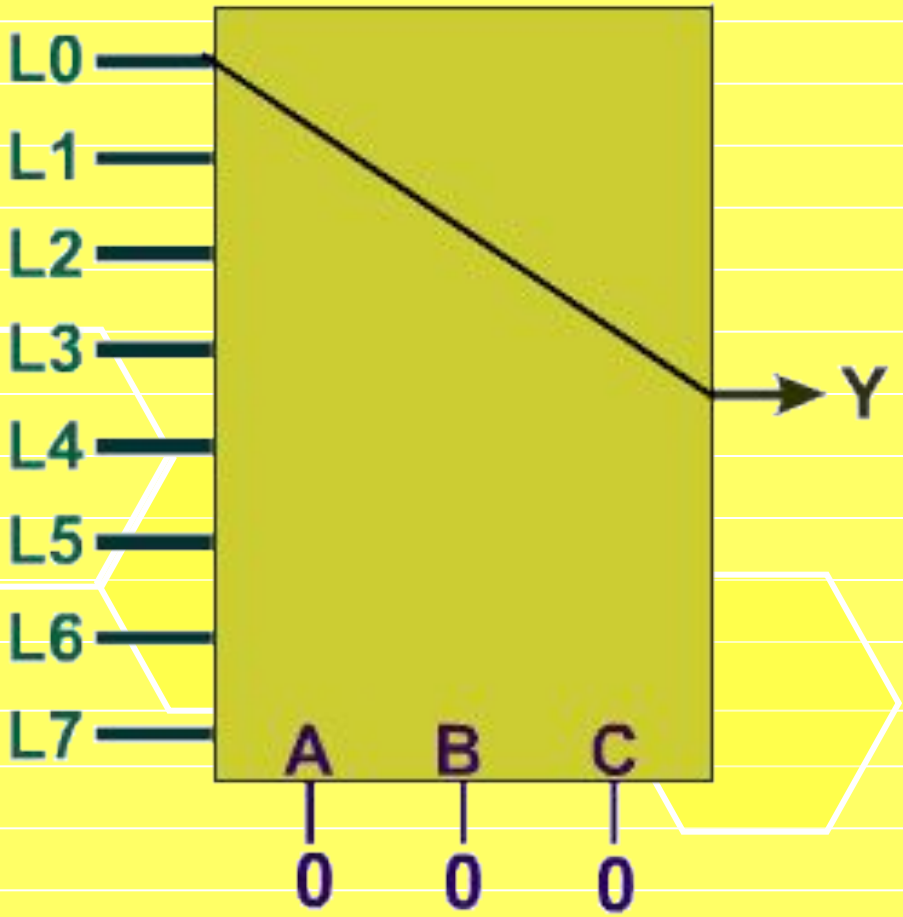
INPUTS							OUTPUT
SELECT†		DATA				$\overline{G}$	Y
B	A	C0	C1	C2	C3		
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

† Select inputs A and B are common to both sections.

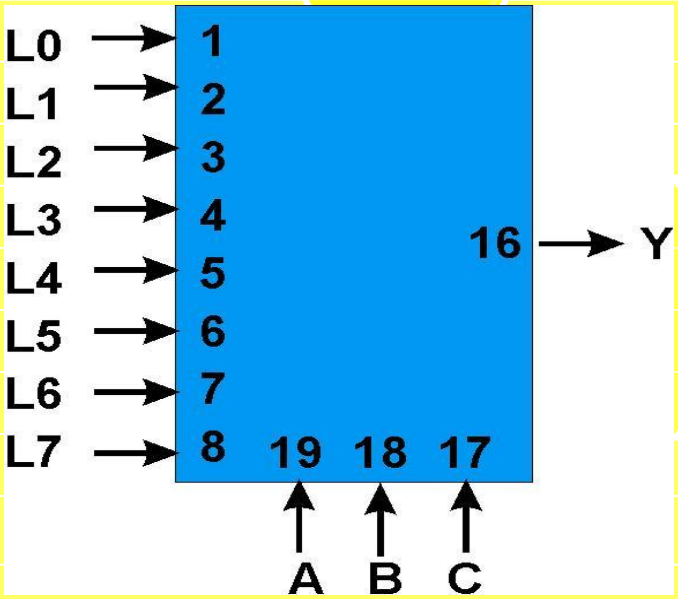
When, Then, Else

m	A B C	Y
0	0 0 0	L0
1	0 0 1	L1
2	0 1 0	L2
3	0 1 1	L3
4	1 0 0	L4
5	1 0 1	L5
6	1 1 0	L6
7	1 1 1	L7

Multiplexor de 8 a 1 línea

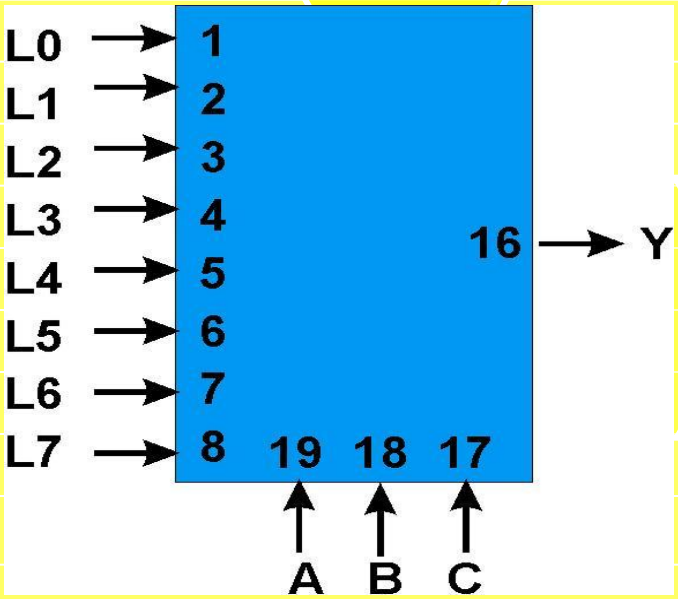


Multiplexor de 8 a 1 línea



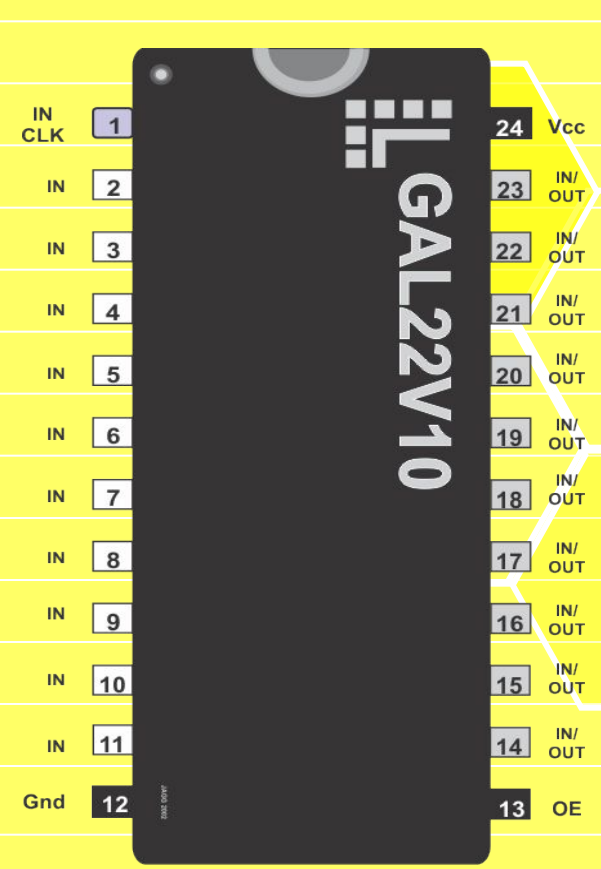

```
MODULE muxeq
" Entradas de datos
   L0..L7 pin 1..8;
"Entradas de control
   A,B,C pin 23..21;
"Salida
   Y pin 16 istype 'com';
Equations
   WHEN !A&!B&!C THEN Y=L0;
   WHEN !A&!B&C  THEN Y=L1;
   WHEN !A&B&!C  THEN Y=L2;
   WHEN !A&B&C   THEN Y=L3;
   WHEN A&!B&!C  THEN Y=L4;
   WHEN A&!B&C   THEN Y=L5;
   WHEN A&B&!C   THEN Y=L6;
   WHEN A&B&C    THEN Y=L7;
END
```

Multiplexor de 8 a 1 línea



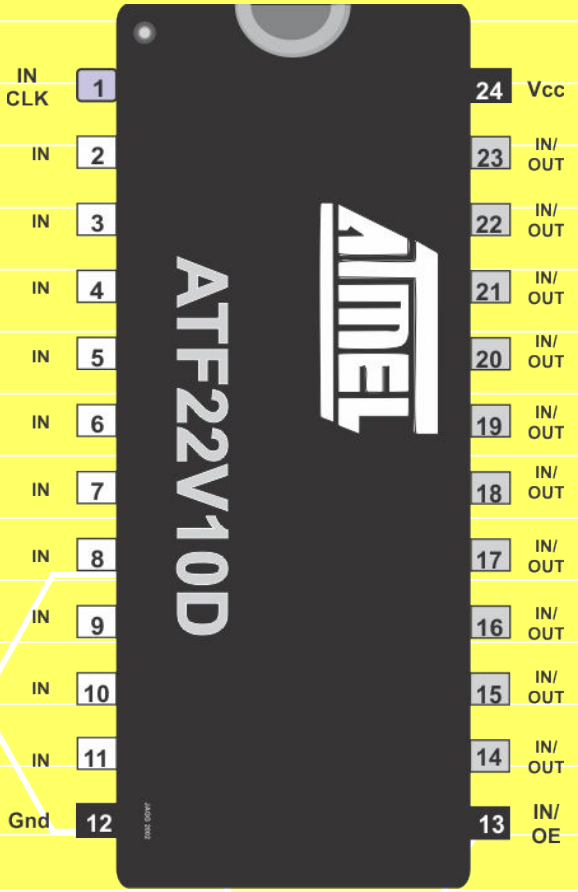

```
MODULE muxeq
" Entradas de datos
   L0..L7 pin 1..8;
"Entradas de control
   A,B,C pin 23..21;
"Salida
   Y pin 16 istype 'com';
S=[A,B,C];
Equations
   WHEN  S==0  THEN Y=L0;
   WHEN  S==1  THEN Y=L1;
   WHEN  S==2  THEN Y=L2;
   WHEN  S==3  THEN Y=L3;
   WHEN  S==4  THEN Y=L4;
   WHEN  S==5  THEN Y=L5;
   WHEN  S==6  THEN Y=L6;
   WHEN  S==7  THEN Y=L7;
END
```

GAL22V10



Entradas (22) :
1 a la 11 , 13 a la 23.

Salidas (10) :
14 a la 23.

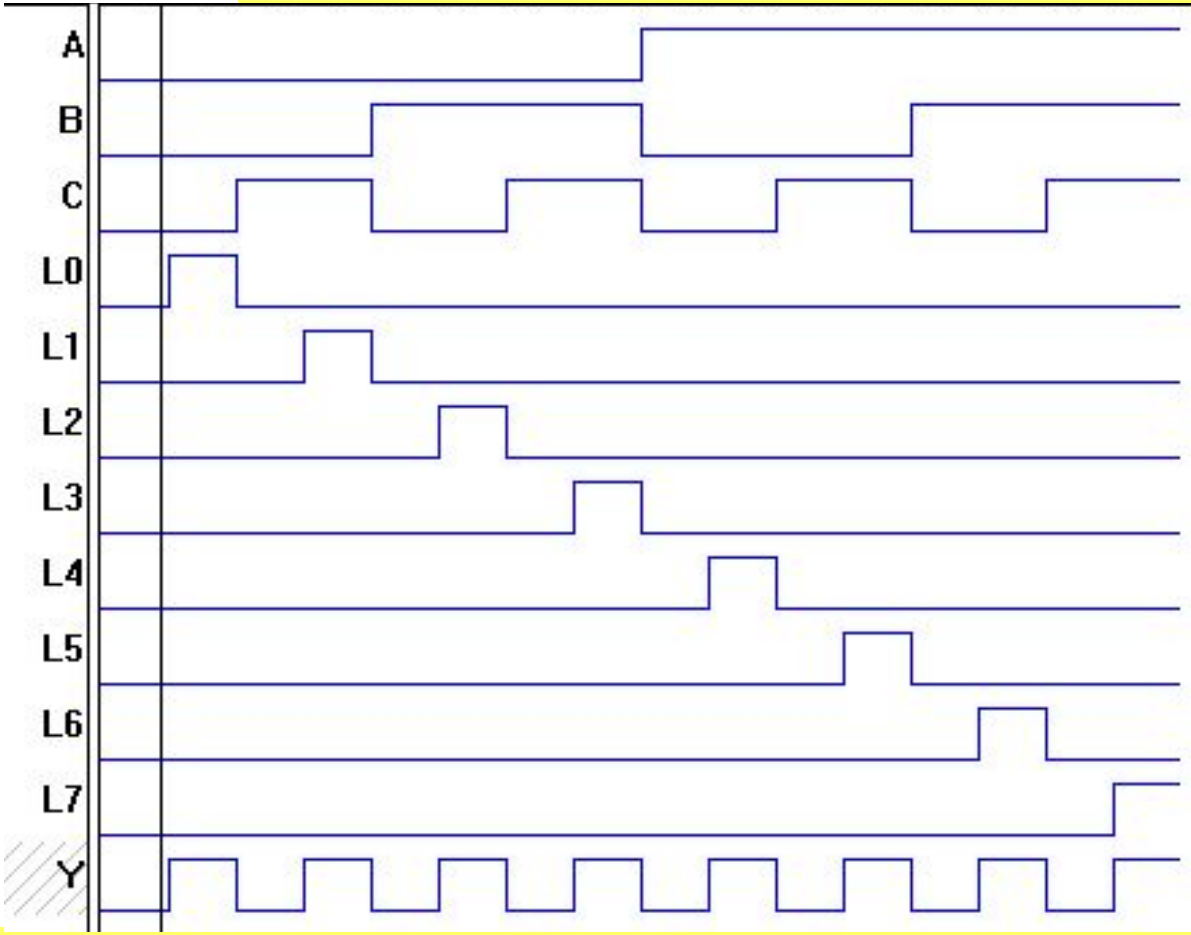


La terminal 1 puede ser usada también como señal de sincronía Clk (circuitos secuenciales).

La terminal 13 también es una entrada de control OE Output Enable.

Test_vectors del Multiplexor de 8 a 1 línea

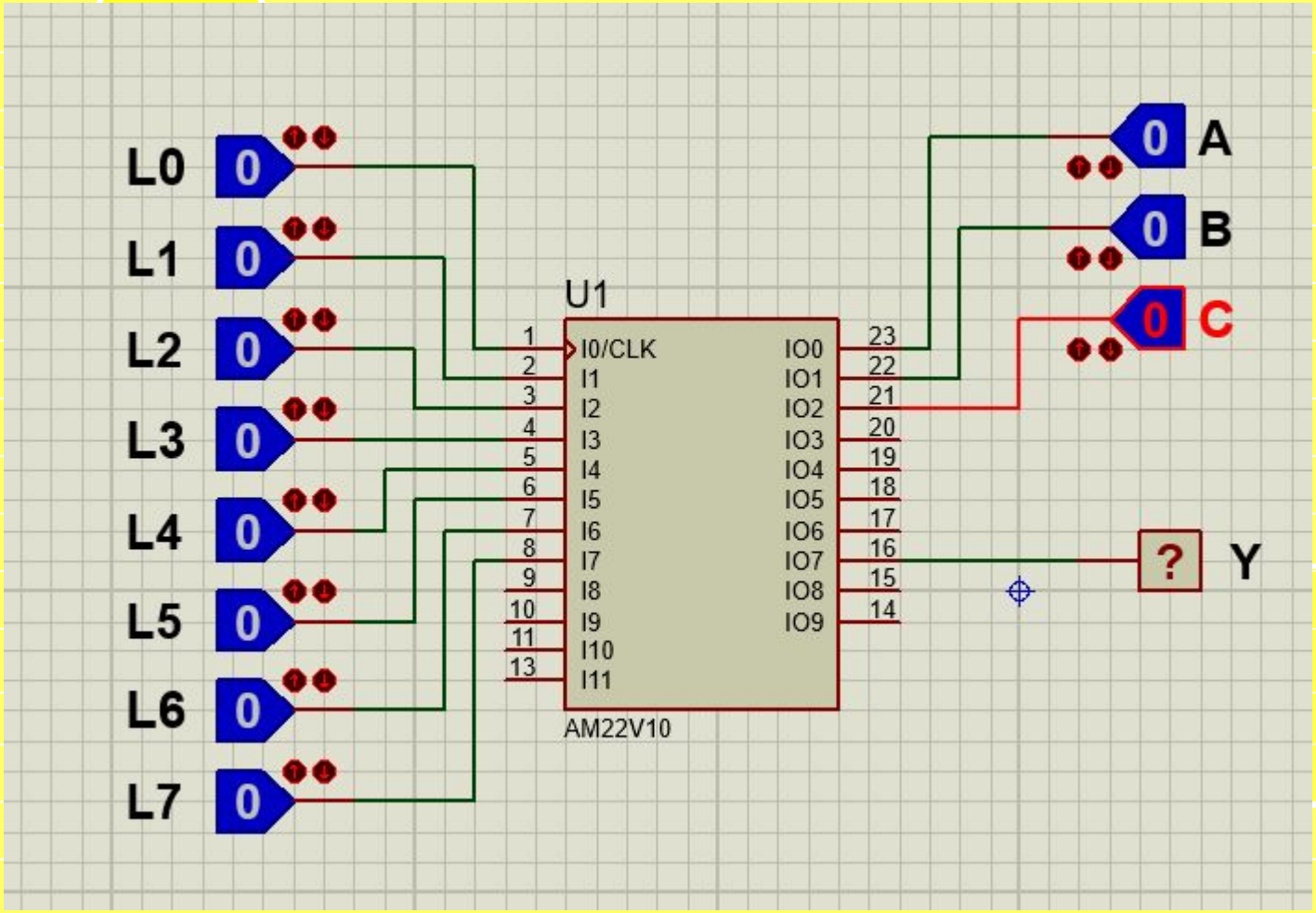
```
Test_vectors
([A,B,C,L7,L6,L5,L4,L3,L2,L1,L0]->[Y])
  [0,0,0,X,X,X,X,X,X,X,0]->[0];
  [0,0,0,X,X,X,X,X,X,X,1]->[1];
  [0,0,1,X,X,X,X,X,X,0,X]->[0];
  [0,0,1,X,X,X,X,X,X,1,X]->[1];
  [0,1,0,X,X,X,X,X,0,X,X]->[0];
  [0,1,0,X,X,X,X,X,1,X,X]->[1];
  [0,1,1,X,X,X,X,0,X,X,X]->[0];
  [0,1,1,X,X,X,X,1,X,X,X]->[1];
  [1,0,0,X,X,X,0,X,X,X,X]->[0];
  [1,0,0,X,X,X,1,X,X,X,X]->[1];
  [1,0,1,X,X,0,X,X,X,X,X]->[0];
  [1,0,1,X,X,1,X,X,X,X,X]->[1];
  [1,1,0,X,0,X,X,X,X,X,X]->[0];
  [1,1,0,X,1,X,X,X,X,X,X]->[1];
  [1,1,1,0,X,X,X,X,X,X,X]->[0];
  [1,1,1,1,X,X,X,X,X,X,X]->[1];
```



Test_vectors del Multiplexor de 8 a 1 línea

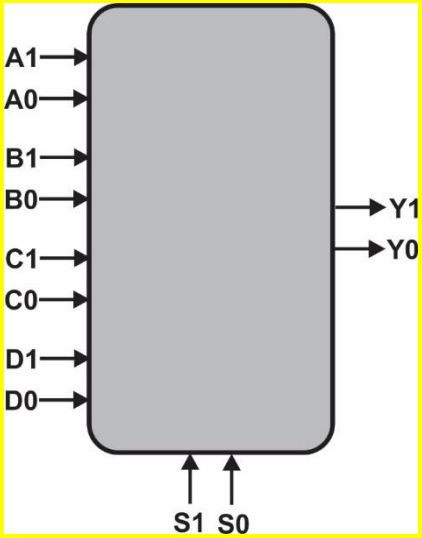
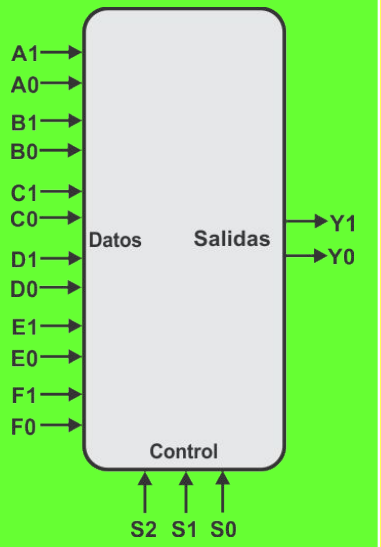
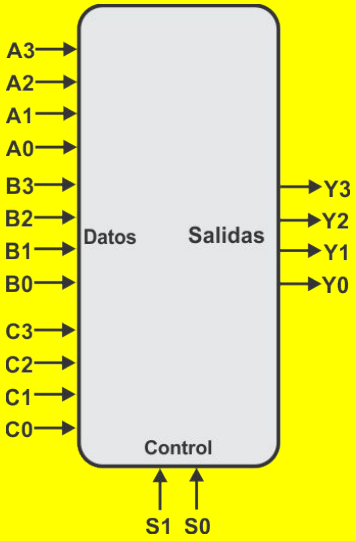
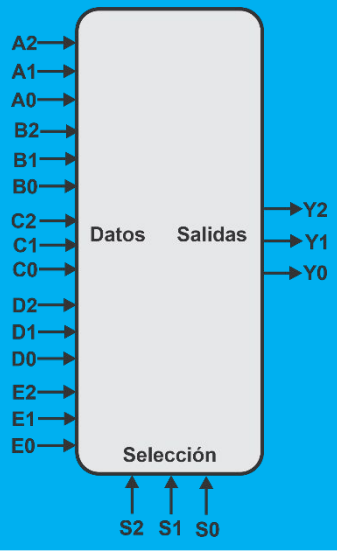
```
Test_vectors
([A,B,C,L7,L6,L5,L4,L3,L2,L1,L0]->[Y])
  [0,0,0, X , X , X, X,  X,  X,  X, 0]->[0];
  [0,0,0,X,X,X,X,X,X,X,1]->[1];
  [0,0,1,X,X,X,X,X,X,0,X]->[0];
  [0,0,1,X,X,X,X,X,X,1,X]->[1];

  [1,1,1,0,X,X,X,X,X,X,X]->[0];
  [1,1,1,1,X,X,X,X,X,X,X]->[1];
```

PROTEUS

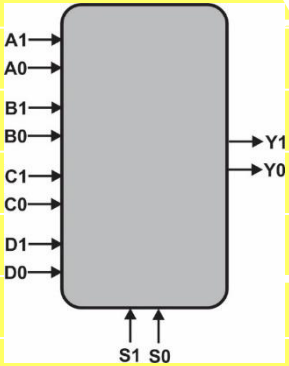

L0
L1
L2
L3
L4
L5
L6
L7
0
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
A
B
C
?
Y

Proyecto Formativo 8

Diseñe un selector de datos

M1	M2	M3	M4
			
4 a 1, 2 bits	6 a 1, 2 bits	4 a 1, 4 bits	5 a 1, 3 bits

M1 Multiplexer de4 a 1, 2 bits

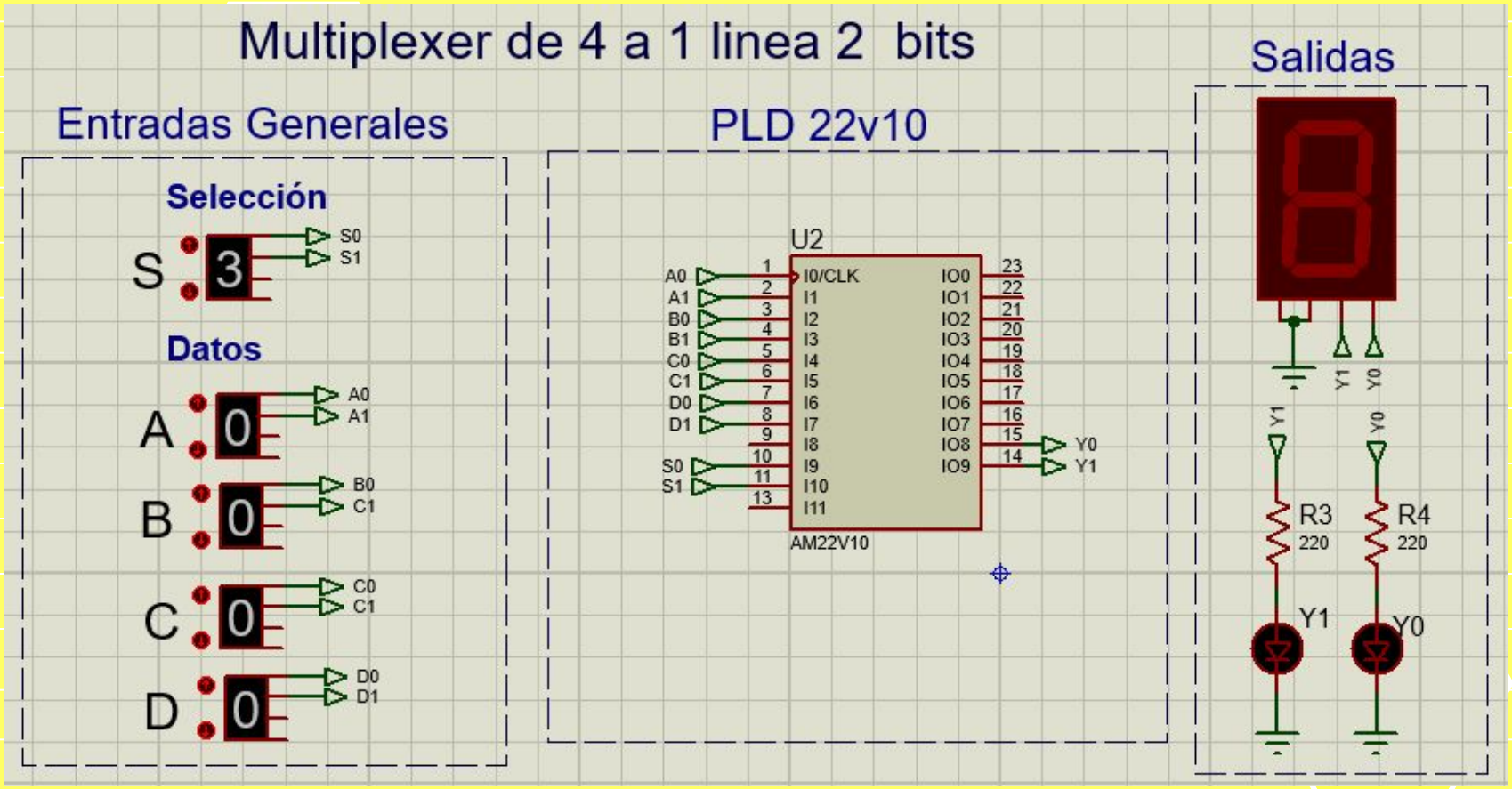



```
A0,A1,B0,B1,C0,C1,D0,D1,S0,S1 pin 1..8,10,11;
Y1,Y0 pin 14,15 istype 'com';
A= [A1,A0];
B= [B1,B0];
C= [C1,C0];
D= [D1,D0];
S= [S1,S0];
Y= [Y1,Y0];
X=.x.;
equations
when S==0 then Y=A;
when S==1 then Y=B;
when S==2 then Y=C;
when S==3 then Y=D;
```

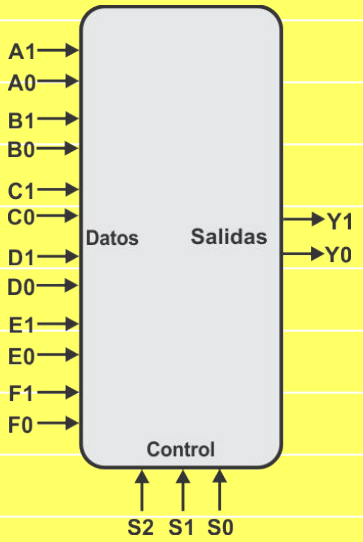
```
test_Vectors
([S,D,C,B,A]->Y)
[0,X,X,X,0]->X;
[0,X,X,X,1]->X;
[0,X,X,X,2]->X;
[0,X,X,X,3]->X;
[1,X,X,0,X]->X;
[1,X,X,1,X]->X;
[1,X,X,2,X]->X;
[1,X,X,3,X]->X;
[2,X,0,X,X]->X;
[2,X,1,X,X]->X;
[2,X,2,X,X]->X;
[2,X,3,X,X]->X;
[3,0,X,X,X]->X;
[3,1,X,X,X]->X;
[3,2,X,X,X]->X;
[3,3,X,X,X]->X;
END
```

M1

4 a 1, 2 bits


Multiplexer de 4 a 1 linea 2 bits
Entradas Generales
Selección
S
3
S0
S1
Datos
A
0
A0
A1
B
0
B0
C1
C
0
C0
C1
D
0
D0
D1
PLD 22v10
U2
A0
A1
B0
B1
C0
C1
D0
D1
S0
S1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
Y0
Y1
AM22V10
Salidas
Y1
Y0
R3
220
R4
220
Y1
Y0

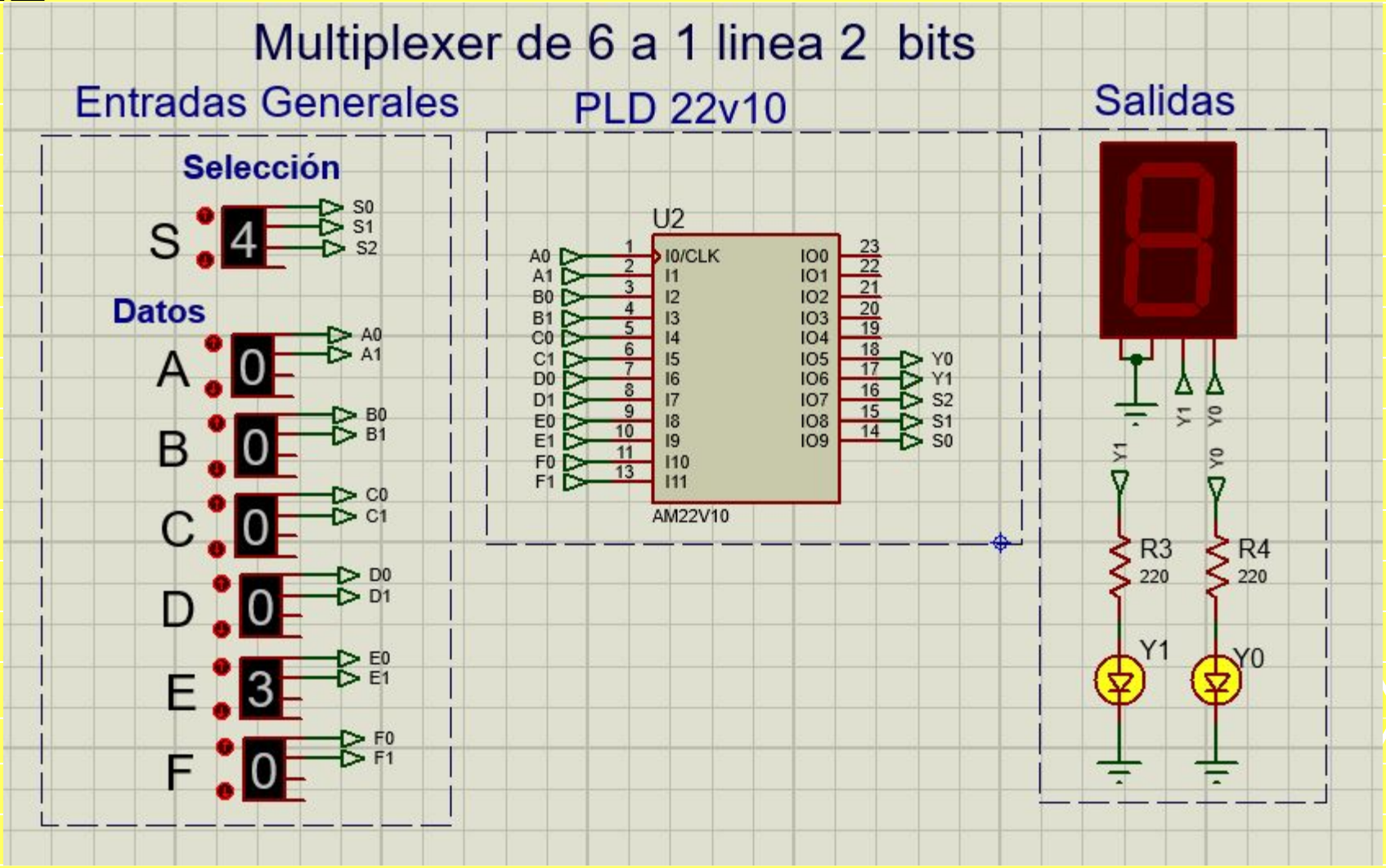
M2 Multiplexer de 6 a 1 línea 2 bits



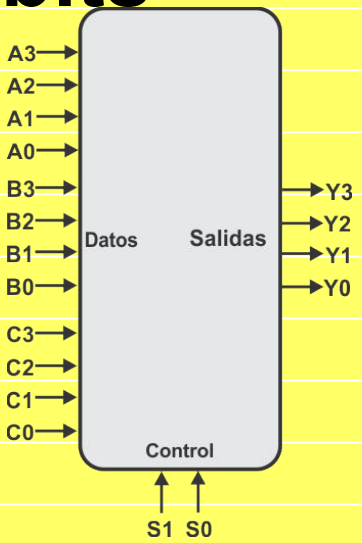

```
MODULE muxED
"Multiplexer de 6 a 1 linea de 2 bits
"4 abril 2021
"JAGG
A0,A1,B0,B1,C0,C1,D0,D1,E0,E1,F0,F1, S0,S1,S2 pin 1..11,13..16;
Y1,Y0 pin 17,18 istype 'com';
A= [A1,A0];
B= [B1,B0];
C= [C1,C0];
D= [D1,D0];
E= [E1,E0];
F= [F1,F0];
S= [S2,S1,S0];
Y= [Y1,Y0];
X=.x.;
```

```
equations
when S==0 then Y=A;
when S==1 then Y=B;
when S==2 then Y=C;
when S==3 then Y=D;
when S==4 then Y=E;
when S==5 then Y=F;
```

M2


Multiplexer de 6 a 1 linea 2 bits
Entradas Generales
PLD 22v10
Salidas
Selección
S
4
S0
S1
S2
Datos
A
0
A0
A1
B
0
B0
B1
C
0
C0
C1
D
0
D0
D1
E
3
E0
E1
F
0
F0
F1
U2
A0
A1
B0
B1
C0
C1
D0
D1
E0
E1
F0
F1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
Y0
Y1
S2
S1
S0
AM22V10
R3
220
R4
220
Y1
Y0

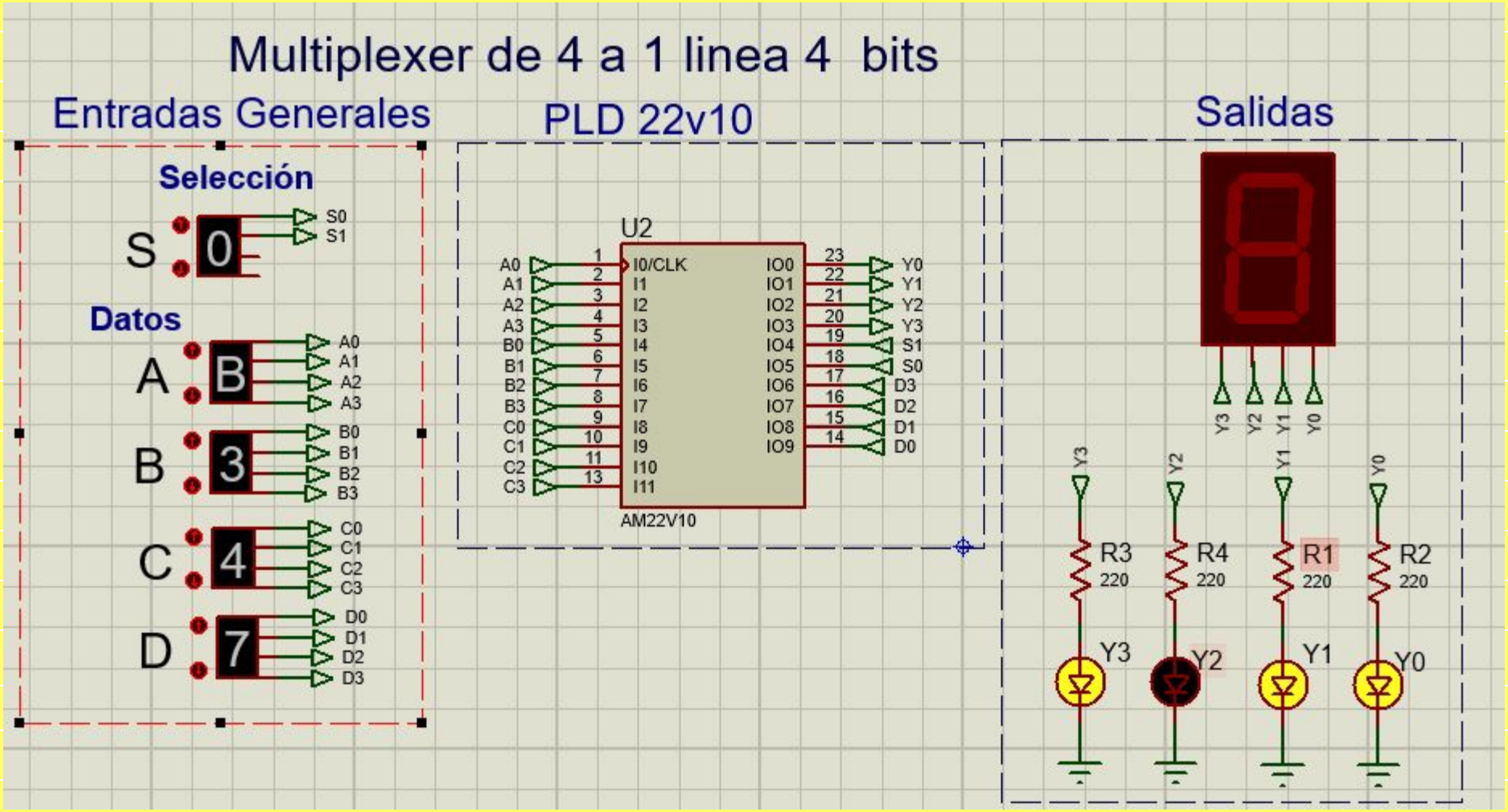
M3 Multiplexer de 4 a 1 línea 4 bits




```
MODULE muxEDx
"Multiplexex de 4 a 1 linea de 4 bits
"4 abril 2021
"JAGG
A0,A1,A2,A3, B0,B1,B2,B3, C0,C1,C2,C3, D0,D1,D2,D3 pin 1..11,13..17;
S1,S0 pin 19,18;
Y3,Y2,Y1,Y0 pin 20..23 istype 'com';
A= [A3,A2,A1,A0];
B= [B3,B2,B1,B0];
C= [C3,C2,C1,C0];
D= [D3,D2,D1,D0];
S= [S1,S0];
Y= [Y3,Y2,Y1,Y0];
X=.x.;
```

```
equations
when S==0 then Y=A;
when S==1 then Y=B;
when S==2 then Y=C;
when S==3 then Y=D;
```

M3

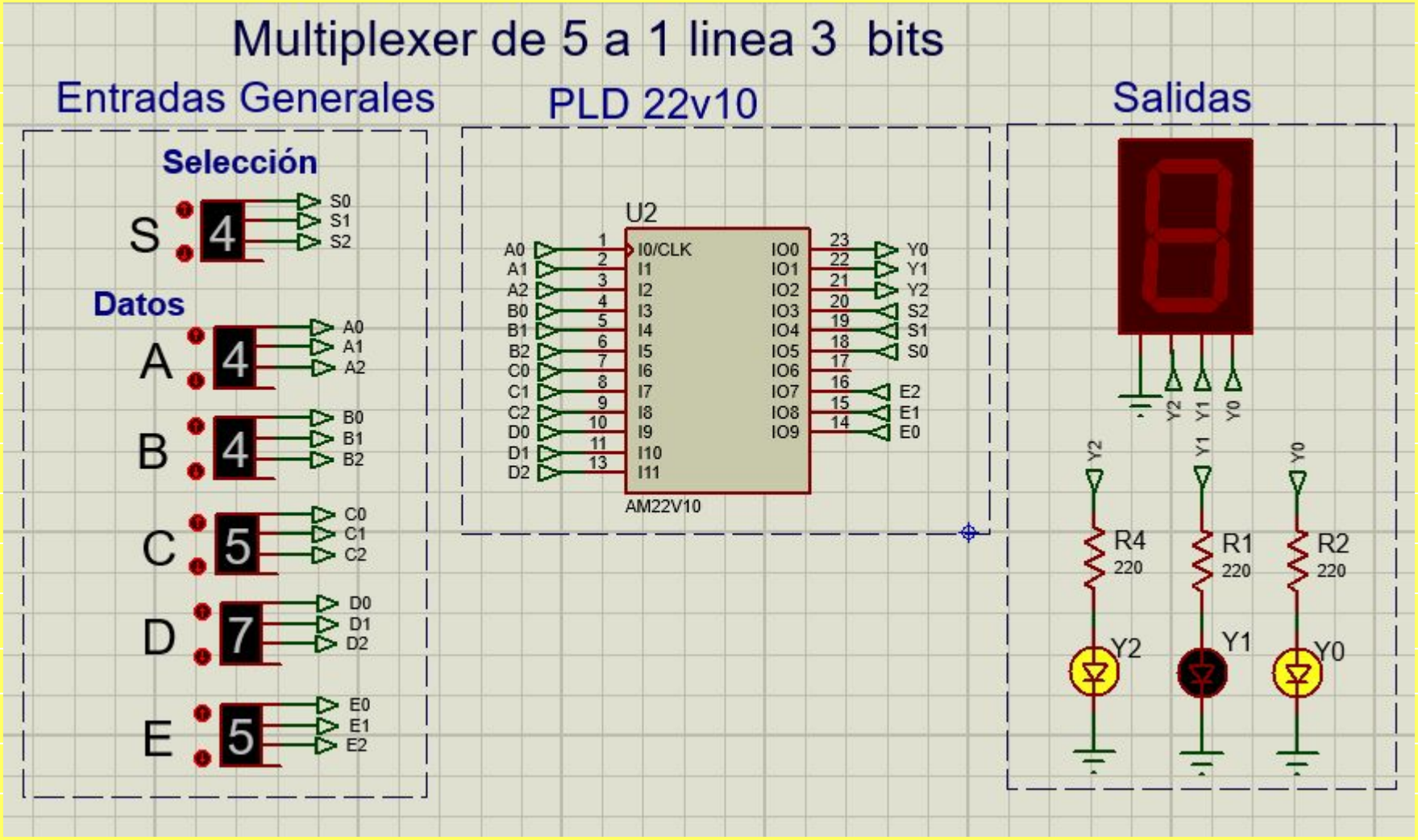

Multiplexer de 4 a 1 linea 4 bits
Entradas Generales
PLD 22v10
Salidas
Selección
S
0
S0
S1
Datos
A
B
3
C
4
D
7
A0
A1
A2
A3
B0
B1
B2
B3
C0
C1
C2
C3
D0
D1
D2
D3
U2
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
Y0
Y1
Y2
Y3
S1
S0
D3
D2
D1
D0
AM22V10
R3
R4
R1
R2
220
Y3
Y2
Y1
Y0

M4 Multiplexor de 5 a 1 líneas de 3 bits

```
MODULE muxEDY
"Multiplexer de 6 a 1 lineas de 3 bits
"5 abril 2021
"JAGG
A0,A1,A2, B0,B1,B2, C0,C1,C2, D0,D1,D2, E0,E1,E2 pin 1..11, 13..16;
S2, S1,S0 pin 20..18;
Y2,Y1,Y0 pin  21..23 istype 'com';
A= [A2,A1,A0];
B= [B2,B1,B0];
C= [C2,C1,C0];
D= [D2,D1,D0];
E= [E2,E1,E0];
S= [S2,S1,S0];
Y= [Y2,Y1,Y0];
X=.x.;
```

```
equations
when S==0 then Y=A;
when S==1 then Y=B;
when S==2 then Y=C;
when S==3 then Y=D;
when S==4 then Y=E;
```

M4


Multiplexer de 5 a 1 linea 3 bits
Entradas Generales
PLD 22v10
Salidas
Selección
S
4
S0
S1
S2
Datos
A
4
A0
A1
A2
B
4
B0
B1
B2
C
5
C0
C1
C2
D
7
D0
D1
D2
E
5
E0
E1
E2
U2
A0
A1
A2
B0
B1
B2
C0
C1
C2
D0
D1
D2
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
Y0
Y1
Y2
S2
S1
S0
E2
E1
E0
AM22V10
R4
220
R1
220
R2
220
Y2
Y1
Y0