ELECTRÓNICA DIGITAL

EJEMPLOS Y EJERCICIOS

Segunda parte

OSCAR IGNACIO BOTERO HENAO

RED educativa digital descartes

iCartesiLibri

Electrónica Digital

Ejemplos y Ejercicios

Segunda parte

Oscar Ignacio Botero Henao

Fondo Editorial RED Descartes

Córdoba (España)
2025

Título de la obra:
Electrónica Digital
Ejemplos y Ejercicios
Segunda parte

Autor:
Oscar Ignacio Botero Henao

Código JavaScript para el libro: Joel Espinosa Longi, IMATE, UNAM.
Recursos interactivos: DescartesJS, WebSim
Fuentes: Lato y UbuntuMono
Figura de portada: ilustración generada por Nightcafe AI

Red Educativa Digital Descartes
Córdoba (España)
descartes@proyectodescartes.org
https://proyectodescartes.org

Proyecto iCartesiLibri
https://proyectodescartes.org/iCartesiLibri/index.htm

ISBN: 978-84-10368-23-1

Esta obra está bajo una licencia Creative Commons 4.0 internacional: Reconocimiento-No Comercial-Compartir Igual.

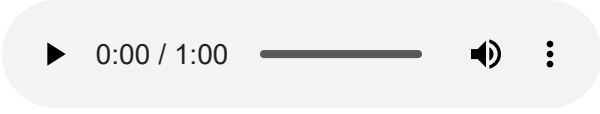
Tabla de contenido

Prefacio **7**
1. Conversión de Código **11**
1.1 Codificar (Encoder) 14
1.2 Decodificar (Decoder) 20
1.3 Ejercicios propuestos con respuestas 24
Ayudas y Complementos **33**
1.4 Videos 35
1.5 Objetos Interactivos 38
1.6 Simulaciones 39
2. Multiplexor y Demultiplexor **41**
2.1 Referencias Comerciales 44
2.2 Multiplexor (MUX) 44
2.3 Demultiplexor (DEMUX) 66
2.4 Ejercicios propuestos con respuestas 74
Ayudas y Complementos **85**
2.5 Videos 87
2.6 Objetos Interactivos 89
2.7 Simulaciones 91
3. Otros Dispositivos **93**
3.1 Comparador de magnitudes 96
3.2 Generador de Paridad y Detector de Error 112
3.3 Latch (Biestables) 123
3.4 Tri-State 126
3.5 Colector Abierto (Open Collector) 127

3.6 Schmitt Trigger 131
3.7 Ejercicios propuestos con respuestas 135
Ayudas y Complementos 141
3.8 Videos 143
3.9 Objetos Interactivos 145
3.10 Simulaciones 145
Ejemplos Anexos 147
EA1-Secuencia Up/Down 149
EA2-Tanque de Leche 152
EA3-Banda Transportadora 157
EA4-Sistema con varios Bloques 162
EA5-Multiplexar 2 BCD en 1 display 170
EA6-Multiplexar 2 BCD en 2 display 174
EA7-Comparador y Multiplexor 178
EA8-Multiplexar 2 condiciones 181
EA9-ALU y DEMUX 185
EA10- Generador de Paridad y Verificación 191
EA11- Comparador Jugadores 195
EA12- Marquesina 201
EA13- Sistema de Tuberías 211
EA14- Balanzas 215
EA15- Tanques comunicados 221

Repaso 231
Actividades de Repaso 233
Anexos 237
Anexo A Hojas de Datos 239
Anexo B Subfamilias de los CI TTL 243
Anexo C Subfamilias de los CI CMOS 244
Anexo D Fabricantes de Circuitos Integrados 245
Anexo E Principales Encapsulados CI 251
Anexo F Símbolos estándar para CI 252
Bibliografía 255

Prefacio



En el vertiginoso mundo de la Electrónica Digital, la clave del dominio radica no solo en el entendimiento teórico, sino en la aplicación práctica de los conceptos. Esta **segunda parte** del libro ***"ELECTRÓNICA DIGITAL, Ejemplos y Ejercicios"*** es la continuación del estudio de esta área de la Electrónica, enfocado exclusivamente en el aprendizaje práctico mediante ejemplos resueltos y ejercicios propuestos con sus respectivas respuestas. Conscientes de la importancia de la aplicación directa de toda la conceptualización teórica, este libro se enfoca en el fortalecimiento y la comprensión a través de problemas que reflejan escenarios reales en el diseño y análisis de circuitos digitales, donde el lector podrá desarrollar habilidades esenciales en el diseño y análisis de circuitos digitales.

Cada capítulo aborda temas fundamentales de la Electrónica Digital, incorporando **ejemplos resueltos, ejercicios con respuestas, recursos complementarios como videos explicativos, objetos interactivos, archivos de simulación y hojas de datos** de los circuitos integrados utilizados, con el fin de ofrecer una experiencia de aprendizaje dinámica e integral.

Los temas abordados que incluye este libro son:

- **Capítulo 1: Conversión de Códigos:**
  - En este primer capítulo se exploran las distintas formas de representar la información en sistemas digitales y cómo realizar la conversión de códigos (Codificar y Decodificar) utilizando como base el sistema numérico binario. A través

de ejemplos prácticos, los lectores aprenderán a implementar y analizar circuitos que transforman información de un formato a otro, lo que es esencial en la transmisión y procesamiento de información digital.

Capítulo 2: Multiplexores y Demultiplexores:

- Aquí se exploran estos dispositivos que permiten el manejo eficiente de múltiples señales a través de un único canal y viceversa que son esenciales en la optimización de recursos y en la reducción de costos en sistemas electrónicos. Se presentan diversas aplicaciones prácticas y su implementación por medio de ejercicios que ilustran cómo estos dispositivos permiten la selección y distribución de datos optimizando la gestión de los recursos en sistemas digitales.

Capítulo 3: Otros dispositivos:

- En este capítulo se abordan diversos componentes fundamentales en la Electrónica Digital y claves en la toma de decisiones y en el control de flujo de datos. Dispositivos como: Comparadores de Magnitudes, utilizados en la evaluación de señales lógicas; Generadores de Paridad, empleados en la transmisión y detección de errores para aseguran la integridad de los datos; Latches (biestables), para el almacenamiento de datos; dispositivos de Tres Estados (tri-state), que permiten la interconexión eficiente de circuitos; dispositivos con Colectores Abiertos, usados en diversas configuraciones de interfaz, y Schmitt Trigger, esencial para el acondicionamiento de señales. Cada tema se presenta con ejemplos que demuestran su aplicación práctica.

Al final del libro, tendrás **15 ejemplos resueltos** que integran varios temas de la Electrónica Digital abordados en los dos volúmenes del libro. Estos ejemplos están diseñados con el fin de consolidar tu conocimiento, demostrando cómo los diferentes elementos interactúan y se complementan en situaciones reales. Además, se incluyen Tablas de **Anexos** con información clave para el diseño como las hojas de datos con las especificaciones técnicas de varios dispositivos, símbolos estándar, subfamilias, sufijos y los principales tipos de encapsulados.

Esta segunda parte es una herramienta didáctica práctica para cualquier estudiante o profesional que desee profundizar y fortalecer su comprensión y habilidades en Electrónica Digital mediante la práctica activa y la resolución de problemas.

Deseo que los dos volúmenes que comprenden este libro, también despierten su curiosidad y pasión por este **Mundo de la Electrónica Digital.**

¡Adelante, y que disfrutes de la lectura, los ejemplos y los retos de los ejercicios propuestos; ¡practica!

Oscar Ignacio Botero Henao
Autor

Capítulo I

Conversión de Código

Codificación y Decodificación es el proceso de asignar a cada entrada una combinación única de bits.[4][5]

Codificar (Encoder): 2^n entradas y n salidas. Las filas de las entradas van a tener un solo dato que cambia y en la salida aparece un código asignado a esas entradas.

$$\boldsymbol{Entradas > Salidas}$$

Decodificar (Decoder): n salidas y 2^n salidas. Cada salida se activa como respuesta a un código de entrada único.

$$\boldsymbol{Entradas < Salidas}$$

1.1 Codificar (Encoder)

★ Caso 1:

> ***Cuando en cada fila de las entradas hay un dato que cambia de valor o estado, este se puede solucionar por Minterm (Σ) o por Maxterm (∏)***

Se tiene un teclado donde solo se utilizan las teclas del 0 al 3 y se requiere visualizar en las salidas el "CÓDIGO BINARIO" equivalente al número decimal de cada combinación de entrada; implemente el circuito combinacional con un software de simulación electrónica y verifique que cumpla su correcto funcionamiento.

$2^n = 2^2 = 4$ entradas máximo, las 4 teclas
n = 2 salidas

Se diseña la tabla de verdad:

No. Dec	Entradas T0	T1	T2	T3	Salidas SB	SA	No. Bin
0_d	1	0	0	0	0	0	0_b
1_d	0	1	0	0	0	1	1_b
2_d	0	0	1	0	1	0	2_b
3_d	0	0	0	1	1	1	3_b

Funciones lógicas para cada salida:

- Minterm (Σ):

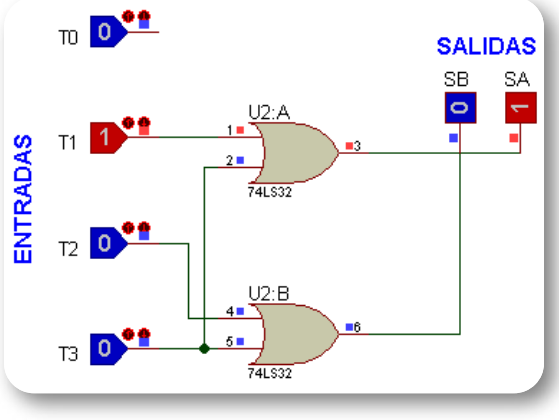
$$SA(\Sigma) = (T1) + (T3)$$
$$SB(\Sigma) = (T2) + (T3)$$

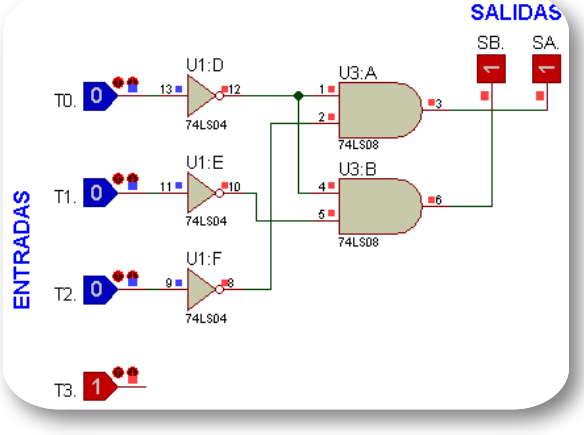
- Maxterm (∏):

$$SA(\Pi) = (\overline{T0}) \times (\overline{T2})$$
$$SB(\Pi) = (\overline{T0}) \times (\overline{T1})$$

Implementación de los circuitos combinacionales:

- Minterm (Σ):





- Maxterm (∏):

Repositorio en Github (Ejemplo_1-1-1.dsn) con el archivo de la simulación con Proteus v7.8 y en Github (Ejemplo_1-1-1.txt) el archivo de simulación con el "Constructor Virtual sobre Protoboards y Simulador de Circuitos Digitales"

★ Caso 2:

Cuando hay una fila de las entradas que ningún dato cambia de valor o estado, este se puede solucionar por Minterm (Σ) o por Maxterm (∏)

Se requiere codificar unos datos de entrada de forma que generen en sus salidas el código que se observa en la siguiente tabla, halle las funciones e implemente el circuito combinacional en un software de simulación electrónica y verifique que cumpla con todas las especificaciones de la tabla.

La tabla de verdad:

	Entradas			Salidas	
	D0	D1	D2	SB	SA
	0	1	1	0	0
	1	0	1	0	1
	1	1	0	1	0
No cambian los datos ➡	1	1	1	1	1

Funciones lógicas para cada salida:

- Minterm (Σ):

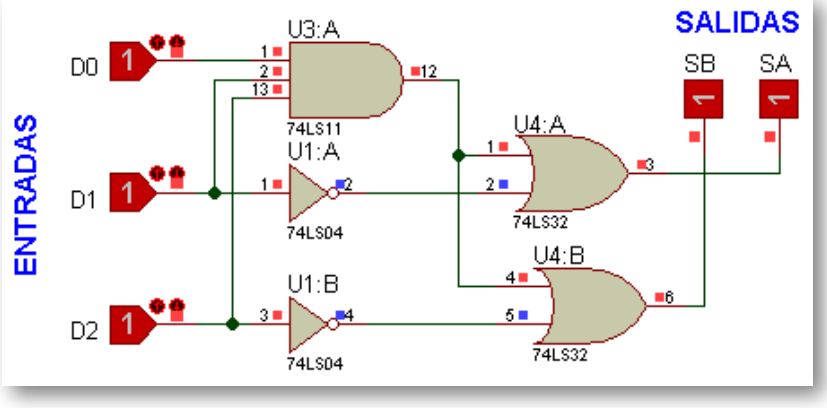
$$SA(\Sigma) = (\overline{D1}) + (D0 \times D1 \times D2)$$
$$SB(\Sigma) = (\overline{D2}) + (D0 \times D1 \times D2)$$

- Maxterm (∏):

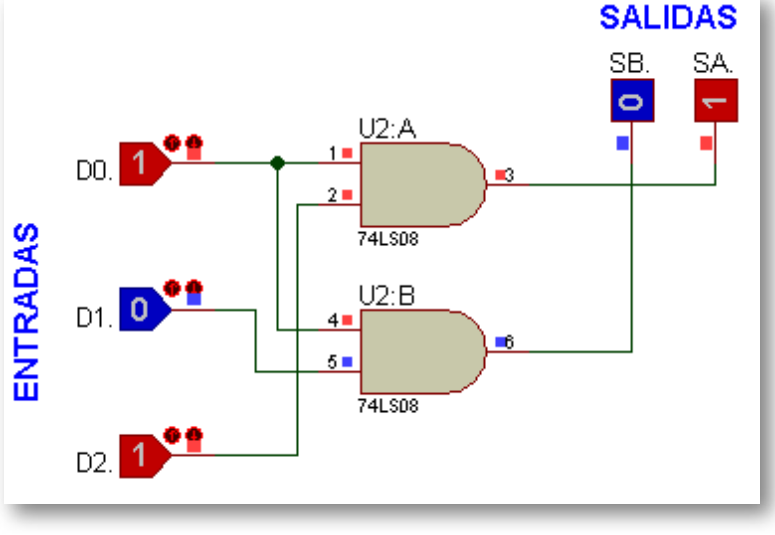
$$SA(\Pi) = (D0) \times (D2)$$
$$SB(\Pi) = (D0) \times (D1)$$

Implementación de los circuitos combinacionales:

- Minterm (Σ):



- Maxterm (∏):



Repositorio en Github (Ejemplo_1-1-2.dsn) con el archivo de la simulación con Proteus v7.8

★ Caso 3:

Cuando hay una fila de las entradas que "TODOS" los datos cambian de valor o estado, se soluciona de acuerdo al dato que tenga en la salida esa fila de datos, si tiene un '1' por Minterm (Σ) y si tiene un '0' por Maxterm (∏)

Se requiere codificar unos datos de entrada de forma que generen en sus salidas el código que se observa en la siguiente tabla, halle las funciones e implemente el circuito combinacional en un software de simulación electrónica y verifique que cumpla con todas las especificaciones de la tabla.

La tabla de verdad:

Entradas			Salidas	
D0	D1	D2	SB	SA
0	1	1	0	0
1	0	1	1	1
1	1	0	1	0
0	0	0	0	1

Todos los datos cambian

Funciones lógicas para cada salida:

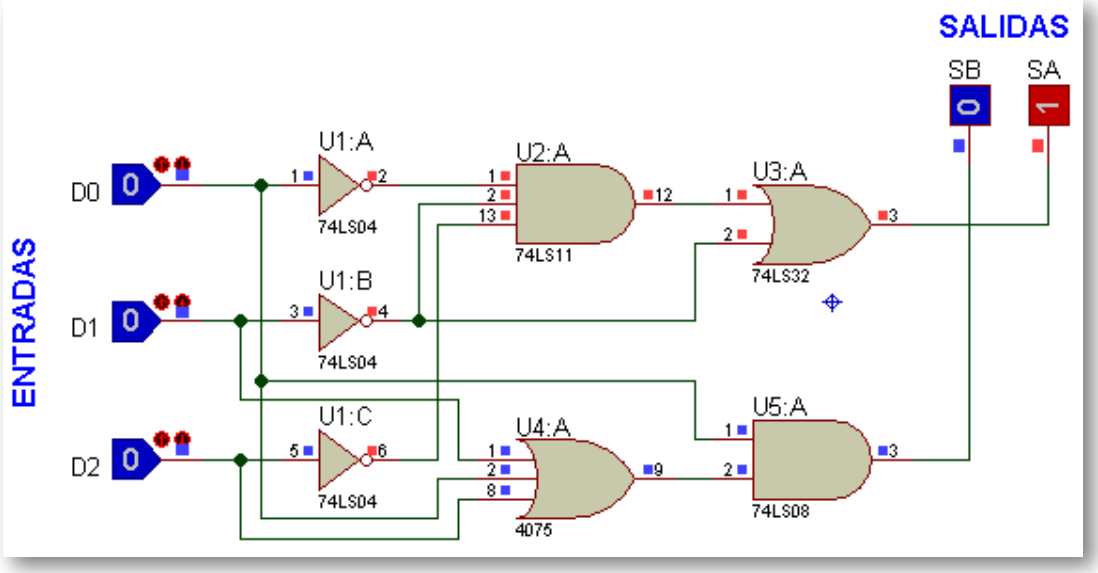
- Minterm (Σ):

$$SA(\Sigma) = (\overline{D1}) + (\overline{D0} \times \overline{D1} \times \overline{D2})$$

- Maxterm (Σ):

$$SB(\Pi) = (D0) \times (D0 + D1 + D2)$$

Implementación del circuito combinacional:



Repositorio en [Github (Ejemplo_1-1-3.dsn)] con el archivo de la simulación con Proteus v7.8

1.2 Decodificar (Decoder)

★ Ejemplo 1:

Se soluciona de acuerdo al dato que cambia en el código de salida, si es '0' por maxterm (∏) y si es '1' por minterm (Σ) y se toma todo el código de entrada como un solo término canónico.

Se tiene una tabla con una decodificación y sus respectivos códigos de salida (2 entradas y 3 salidas). Hallar las funciones lógicas de salida e implemente el circuito combinacional con un software de simulación electrónica para verificar que cumpla con todas las especificaciones de la tabla.

n = 2 entradas
$2^n = 2^2$ = 4 salidas máximo

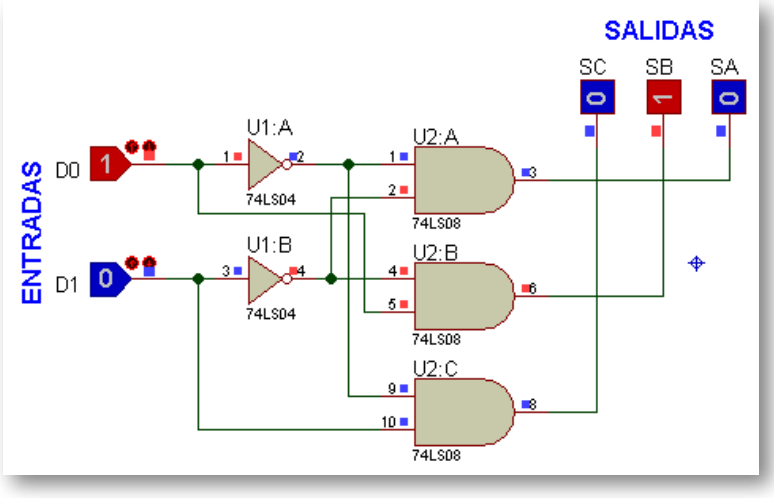
Se diseña la tabla de verdad:

Entradas		Salidas		
D0	D1	SC	SB	SA
0	0	0	0	*1*
1	0	0	*1*	0
0	1	*1*	0	0
1	1	0	0	0

Las funciones lógicas de cada salida son:

$$SA(\Sigma) = (\overline{D0} \times \overline{D1})$$
$$SB(\Sigma) = (D0 \times \overline{D1})$$
$$SB(\Sigma) = (\overline{D0} \times D1)$$

La implementación del circuito combinacional:



Repositorio en Github (Ejemplo_1-2-1.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

Se tiene una tabla con una decodificación similar a la anterior y sus respectivos códigos de salida (2 entradas y 3 salidas). Hallar las funciones lógicas de salida e implemente el circuito combinacional con un software de simulación electrónica para verificar que cumpla con todas las especificaciones de la tabla.

n = 2 entradas
$2^n = 2^2 = 4$ salidas máximo

Diseño de la tabla de verdad:

Entradas		Salidas		
D0	D1	SC	SB	SA
0	0	0	0	1
1	0	0	1	0
0	1	1	0	0
1	1	1	1	1

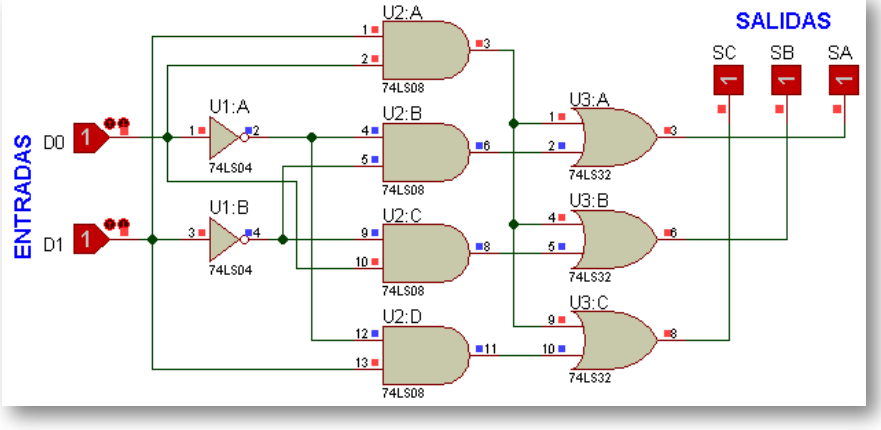
- Funciones lógicas para cada salida:
  - Minterm (Σ):

$$SA(\Sigma) = (\overline{D0} \times \overline{D1}) + (D0 \times D1)$$
$$SB(\Sigma) = (D0 \times \overline{D1}) + (D0 \times D1)$$
$$SC(\Sigma) = (\overline{D0} \times D1) + (D0 \times D1)$$

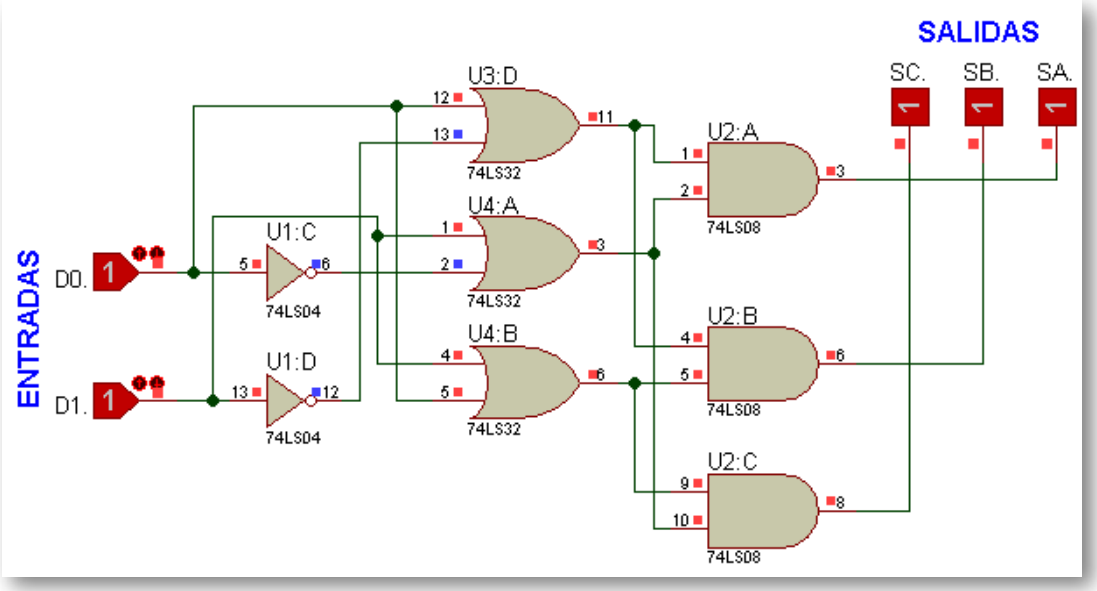
  - Maxterm (∏):

$$SA(\Pi) = (\overline{D0} + D1) \times (D0 + \overline{D1})$$
$$SB(\Pi) = (D0 + D1) \times (D0 + \overline{D1})$$
$$SC(\Pi) = (D0 + D1) \times (\overline{D0} + D1)$$

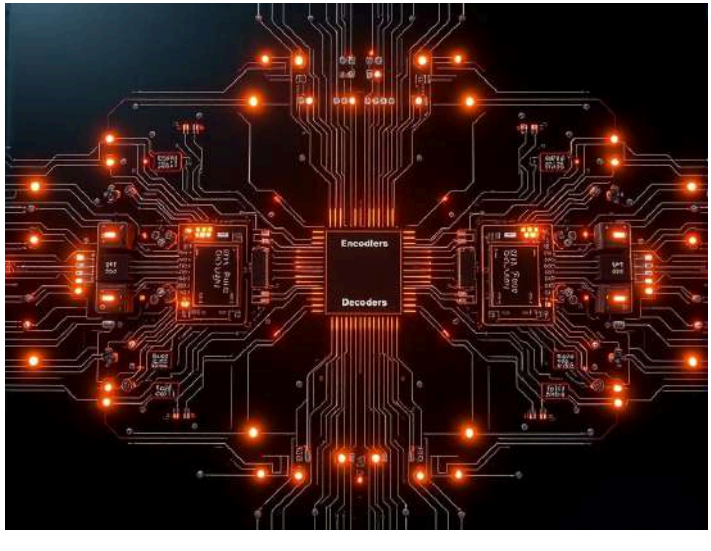
- Implementación de los circuitos combinacionales:
  - Minterm (Σ):



- Maxterm (∏):



Repositorio en [Github (Ejemplo_1-2-2.dsn)] con el archivo de la simulación con Proteus v7.8



1.3 Ejercicios propuestos con respuestas

★ Ejercicio 1:

Diseñar el circuito combinacional para el Codificador de 8 entradas Decimal a Binario de la tabla, obtenga las funciones lógicas y verifique su funcionamiento por medio de un software de simulación electrónica.

Tabla de verdad:

Entradas								Salidas		
D0	D1	D2	D3	D4	D5	D6	D7	B2	B1	B0
0	1	1	1	1	1	1	1	0	0	0
1	0	1	1	1	1	1	1	0	0	1
1	1	0	1	1	1	1	1	0	1	0
1	1	1	0	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	0	1	1	1	0	1
1	1	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	1	1	1

Funciones lógicas para cada salida:

- Minterm (Σ):

$$B0(\Sigma) = \overline{D1} + \overline{D3} + \overline{D5} + \overline{D7}$$
$$B1(\Sigma) = \overline{D2} + \overline{D3} + \overline{D6} + \overline{D7}$$
$$B2(\Sigma) = \overline{D4} + \overline{D5} + \overline{D6} + \overline{D7}$$

★ Ejercicio 2:

Diseñar el circuito combinacional para el Decodificador de 3 entradas binarias y salidas decimales de la tabla, obtenga las funciones lógicas y verifique su funcionamiento por medio de un software de simulación electrónica.

Tabla de verdad del Decodificador:

Entradas			Salidas							
B2	B1	B0	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Funciones lógicas para el circuito Decodificador:

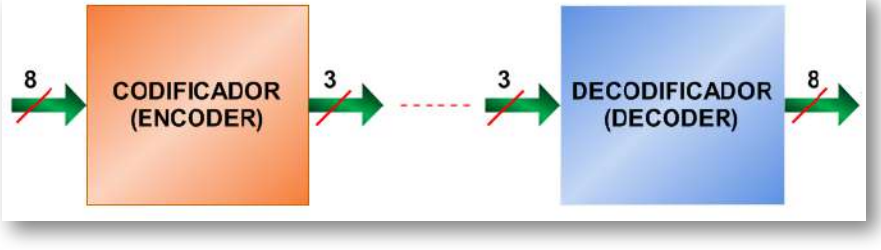
$$D0(\Pi) = (\overline{B0} + B1 + B2)$$
$$D1(\Pi) = (\overline{B0} + B1 + B2)$$
$$D2(\Pi) = (B0 + \overline{B1} + B2)$$
$$D3(\Pi) = (\overline{B0} + \overline{B1} + B2)$$
$$D4(\Pi) = (B0 + B1 + \overline{B2})$$
$$D5(\Pi) = (\overline{B0} + B1 + \overline{B2})$$
$$D6(\Pi) = (B0 + \overline{B1} + \overline{B2})$$
$$D7(\Pi) = (\overline{B0} + \overline{B1} + \overline{B2})$$

★ Ejercicio 3:

Con base en la siguiente tabla de verdad para un circuito Codificador (Encoder), diseñe el circuito combinacional y luego invierta la tabla (las 8 entradas pasan a ser las salidas y viceversa) y diseñe el Decodificador (Decoder); finalmente, conecte ambos circuitos y mediante un software de simulación electrónica verifique su funcionamiento.

Entradas						Salidas		
D0	D1	D2	D3	D4	D5	C	B	A
1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1
0	0	1	1	0	0	0	1	0
0	0	0	1	0	0	0	1	1
0	0	0	0	1	0	1	0	0
0	0	0	0	0	1	1	0	1

Diagrama general del sistema:



Funciones lógicas para el Codificador:

$$A(\Sigma) = (D1) + (D3) + (D5)$$
$$B(\Sigma) = (D2) + (D3)$$
$$C(\Sigma) = (D4) + (D5)$$

Tabla de verdad del Decodificador:

Entradas			Salidas					
C	B	A	D0	D1	D2	D3	D4	D5
0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	0
0	1	1	0	1	0	1	0	0
1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	1

Funciones lógicas para el Decodificador:

$$D0(\Sigma) = (\bar{A} \times \bar{B} \times \bar{C})$$
$$D1(\Sigma) = (A \times \bar{B} \times \bar{C})$$
$$D2(\Sigma) = (\bar{A} \times B \times \bar{C})$$
$$D3(\Sigma) = (A \times B \times \bar{C})$$
$$D4(\Sigma) = (\bar{A} \times \bar{B} \times C)$$
$$D5(\Sigma) = (A \times \bar{B} \times C)$$

★ Ejercicio 4:

Diseñe los circuitos convertidores de código que al activarse cada uno de los cuatro (4) pulsadores, se visualice en la salida en código binario cada digito perteneciente a las unidades y a las decenas (ver el diagrama de bloques)

- *Pulsador 1 = 14*
- *Pulsador 2 = 50*
- *Pulsador 3 = 23*
- *Pulsador 4 = 72*

Verifique su funcionamiento mediante un software de simulación electrónica.

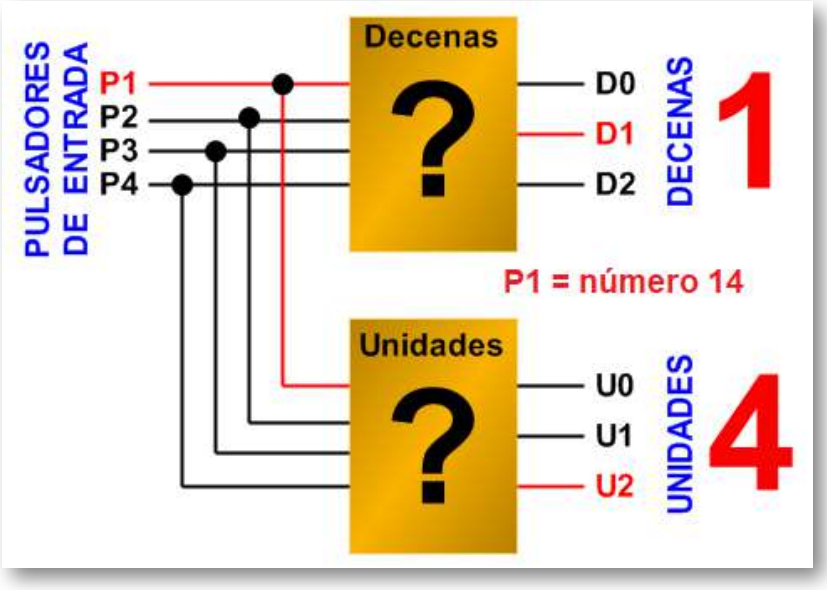


Tabla de verdad del Codificador de las Unidades:

Entradas				Salidas			
P0	P1	P2	P3	U2	U1	U0	No.
1	0	0	0	1	0	0	4
0	1	0	0	0	0	0	0
0	0	1	1	0	1	1	3
0	0	0	1	0	1	0	2

Tabla de verdad del Codificador de las Decenas:

Entradas				Salidas			
P0	P1	P2	P3	D2	D1	D0	No.
1	0	0	0	0	0	1	1
0	1	0	0	1	0	1	5
0	0	1	1	0	1	0	2
0	0	0	1	1	1	1	7

Funciones lógicas del Codificador de las Unidades:

$$\textit{Unidades}$$
$$U0(\Sigma) = (P3)$$
$$U1(\Sigma) = (P3) + (P4)$$
$$U2(\Sigma) = (P1)$$

Funciones lógicas del Codificador de las Decenas:

$$\textit{Decenas}$$
$$D0(\Pi) = (\overline{P3})$$
$$D1(\Pi) = (\overline{P1}) \times (\overline{P2})$$
$$D2(\Pi) = (\overline{P1}) \times (\overline{P3})$$

★ Ejercicio 5:

Consulte el código Gray y diseñe el circuito combinacional para el codificador de decimal (0 a 7) a código Gray; obtenga las funciones lógicas y verifique su funcionamiento por medio de un software de simulación electrónica

Tabla de verdad para el Codificador de Decimal a Gray:

Entradas								Salidas		
D0	D1	D2	D3	D4	D5	D6	D7	G2	G1	G0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	1	1	0
0	0	0	0	0	1	0	0	1	1	1
0	0	0	0	0	0	1	0	1	0	1
0	0	0	0	0	0	0	1	1	0	0

Funciones lógicas para el Codificador:

$$G0(\Sigma) = D1 + D2 + D5 + D6$$
$$G1(\Sigma) = D2 + D3 + D4 + D5$$
$$G2(\Sigma) = D4 + D5 + D6 + D7$$

★ Ejercicio 6:

Consulte el código Gray y diseñe el circuito combinacional para el decodificador de código Gray a decimal (0 a 7); obtenga las funciones lógicas y verifique su funcionamiento por medio de un software de simulación electrónica

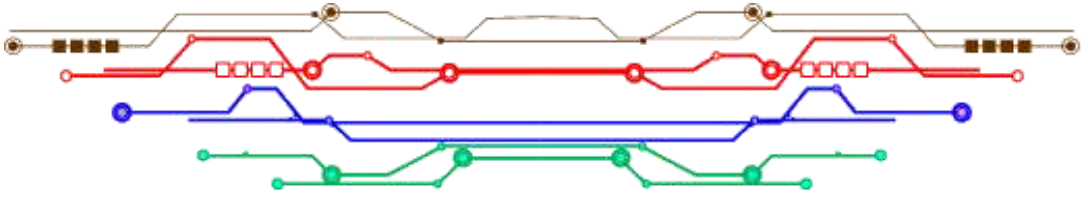
Tabla de verdad para el Decodificador de Gray a Decimal:

Entradas			Salidas							
G2	G1	G0	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1

Funciones lógicas para el Decodificador:

$$D0(\Sigma) = (\overline{G0} \times \overline{G1} \times \overline{G2})$$
$$D1(\Sigma) = (G0 \times \overline{G1} \times \overline{G2})$$
$$D2(\Sigma) = (G0 \times G1 \times \overline{G2})$$
$$D3(\Sigma) = (\overline{G0} \times G1 \times \overline{G2})$$
$$D4(\Sigma) = (\overline{G0} \times G1 \times G2)$$
$$D5(\Sigma) = (G0 \times G1 \times G2)$$
$$D6(\Sigma) = (G0 \times \overline{G1} \times G2)$$
$$D7(\Sigma) = (\overline{G0} \times \overline{G1} \times G2)$$

Repositorio en Github (Ejercicio 1.3.1-Codificador, Ejercicio 1.3.2-Decodificador, Ejercicio 1.3.3-Encoder-Decoder, Ejercicio_1-3-4-Codificador con pulsadores, Ejercicio 1.3.5-Codificador Gray, Ejercicio 1.3.6-Decodificador Gray) con los archivos de las simulaciones con Proteus v7.8. y Github (Ejercicio 1-3-4 Codificador con Pulsadores.txt) con el archivo de la simulación con el "Constructor Virtual sobre Protoboards y Simulador de Circuitos Digitales"



AYUDAS

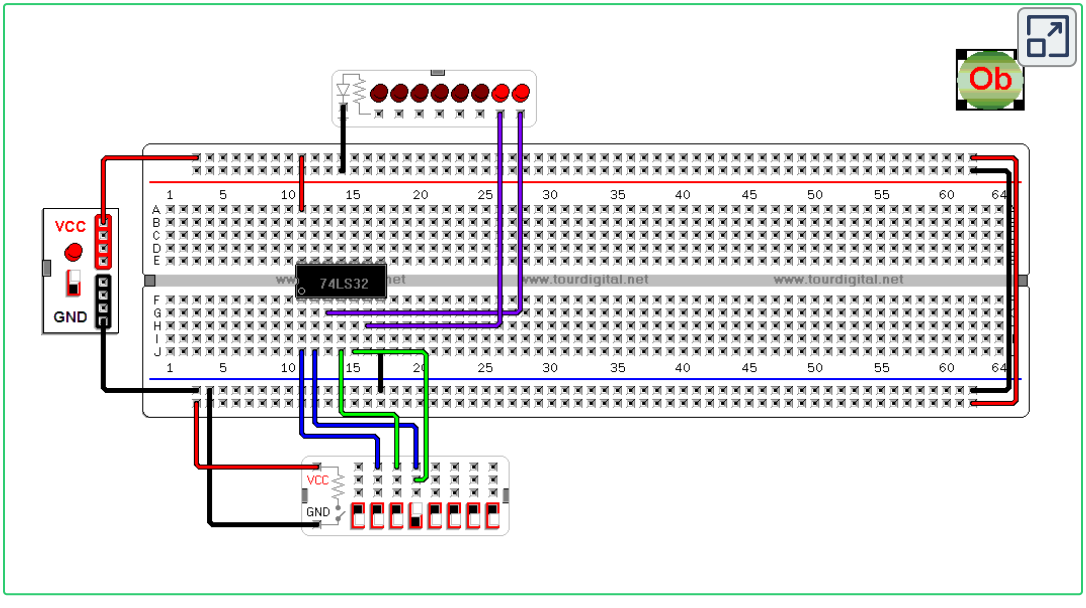
COMPLEMENTOS

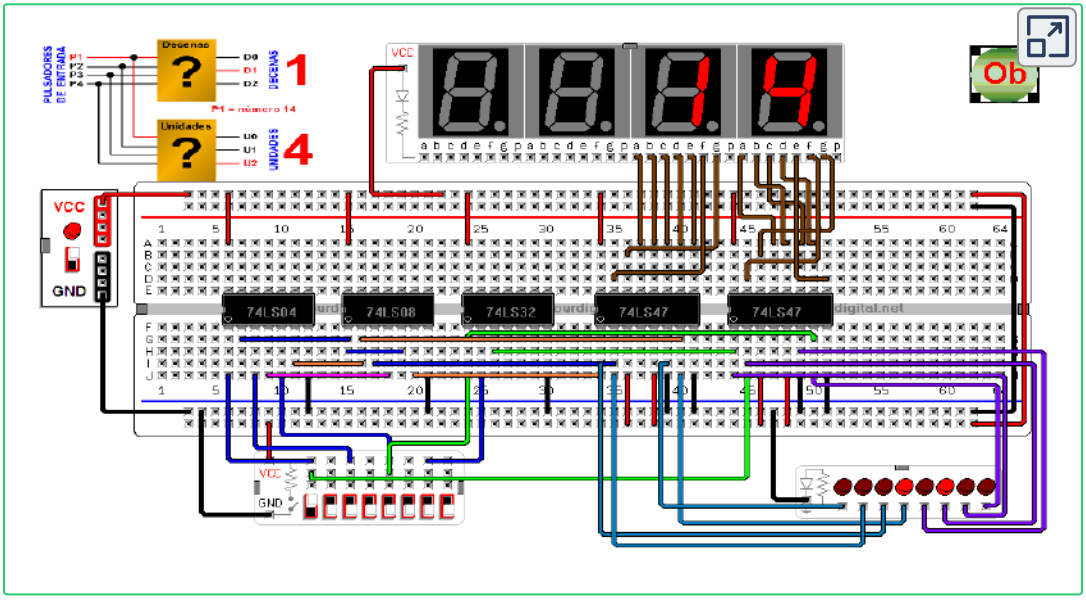
1.4 Videos

Video explicativo realizado por Oscar Botero sobre ***"Codificadores y Decodificadores"***.

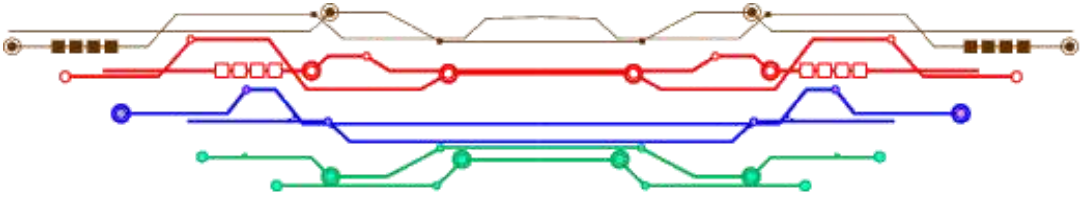
Video demostrativo creado por Oscar Ignacio Botero Henao utilizando la aplicación "LogicsSandbox", simulando el ejemplo del caso 1 de Codificadores visto en la sección 1 de este capítulo.

Videos explicativos realizados por Oscar Botero sobre ***"Codificadores y Decodificadores"***, utilizando el programa "Constructor Virtual sobre Protoboards y Simulador de Circuitos Digitales". Caso 1 del ejemplo de codificadores de la sección 1 y el ejercicio propuesto 4 de la sección 3 de este capítulo.



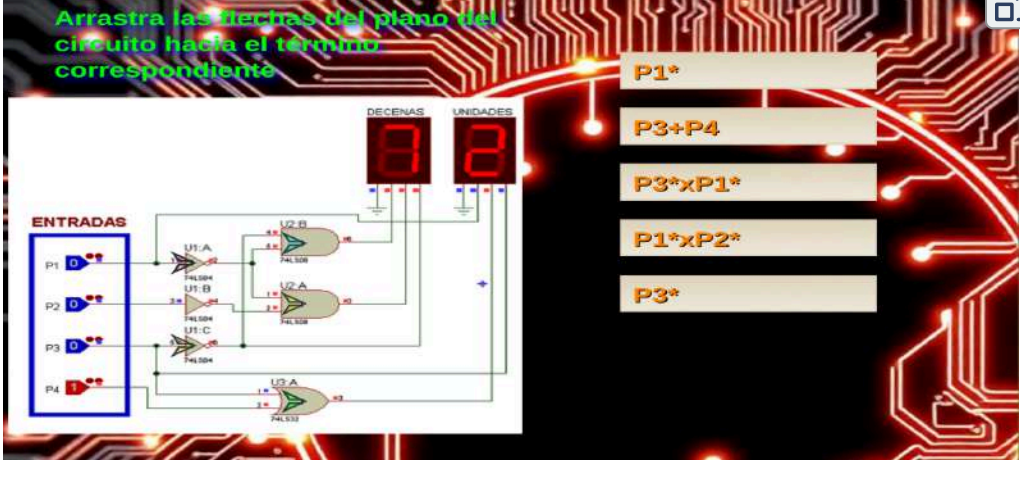


Video explicativo: [Conversores de códigos Gray y Binario natural, Circuito Combinacional - Solución y Simulación](), en el canal de YouTube de Oscar Botero [@oscarboterohenao]()



1.5 Objetos Interactivos

Objetos Interactivos con planos esquemáticos de circuitos electrónicos de Codificadores y Decodificadores utilizados en este capítulo. [1]



[1] Plantillas de los Objetos Interactivos obtenidas del sitio web del Proyecto Descartes, intervenidas por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

Haga clic sobre la imagen para abrir el objeto interactivo:

Este objeto interactivo fue generado con la asistencia de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/

1.6 Simulaciones

Repositorio en Github con los archivos de las simulaciones con Proteus v7.8 y del Constructor Virtual sobre Protoboards y Simulador de Circuitos Digitales

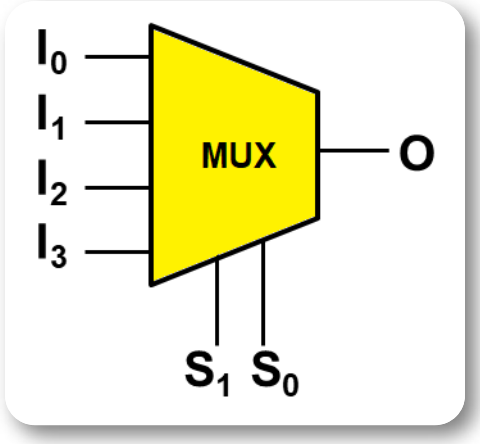
➤ Para ejecutar la aplicación "LogicsSandbox" dar clic en el siguiente ícono:

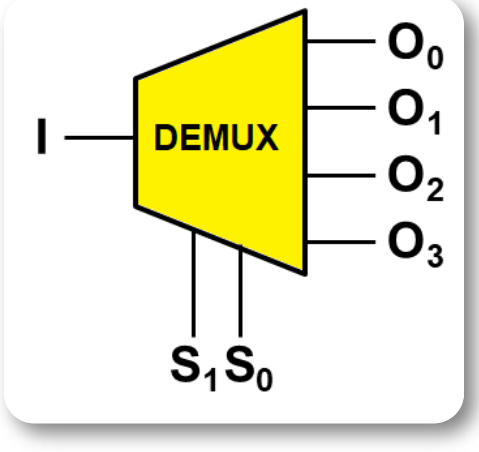
➤ Aplicación para simular circuitos digitales, "Constructor Virtual sobre Protoboards y Simulador de Circuitos Digitales" del Ing. Arturo Javier Miguel del Prieto Paz; haz clic en el ícono para descargarlo:

Capítulo II

Multiplexor y Demultiplexor

Un Multiplexor (MUX) es un circuito combinacional al que entran varios canales de datos, y sólo salen los datos del canal que se haya 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.[1][2][3][4][10]





Un Demultiplexor (DEMUX) es un circuito combinacional que permite tomar el dato en la entrada y de las líneas de control de selección, **SELECCIONAR** por cual salida entregarlo. Los Demux funcionan de forma contraria a los Mux.

Estos dispositivos son utilizados en sistemas para la transmisión de datos, distribuyendo y separando las señales de manera más eficiente.[1][2][3][4][10]

- Los Multiplexores se clasifican en: 2:1 (1 línea selectora), 4:1 (2 líneas selectoras), 8:1 (3 líneas selectoras) y 16:1 (4 líneas selectoras).
- Los Demultiplexores se clasifican en: 1:2 (1 línea selectora), 1:4 (2 líneas selectoras), 1:8 (3 líneas selectoras) y 1:16 (4 líneas selectoras).

2.1 Referencias Comerciales

Multiplexores (MUX)

- 74150 = 1 MUX de 16 a 1, salida activa en BAJO
- 74151 = 1 MUX de 8 a 1, con 2 salidas: activas en BAJO y en ALTO
- 74153 = 2 MUX de 4 a 1, salida activa en ALTO
- 74157 = 4 MUX de 2 a 1, salida activa en ALTO

Demultiplexores (DEMUX)

- 74138 = 1 DEMUX de 1 a 8, salidas activas en BAJO
- 74238 = 1 DEMUX de 1 a 8, salida activa en ALTO
- 74139 = 2 DEMUX de 1 a 4, salidas activas en BAJO
- 4555/4556 = 2 DEMUX de 1 a 4, el 4555 tiene las salidas activas en ALTO y el 4556 activas en bajo
- 74HC154 = 1 DEMUX de 1 a 16, salidas activas en BAJO
- 4067 = 1 DEMUX de 1 a 16, salidas activas en ALTO

MUX / DEMUX

- 4051 = 1 MUX-DEMUX de 8 canales, salida activa en ALTO. Como DEMUX las salidas tienen alta Z, utilizar resistencia de pulldown
- 4053 = 3 MUX-DEMUX de 2 canales, salida activa en ALTO. Como DEMUX las salidas tienen alta Z, utilizar resistencia de pulldown

2.2 Multiplexor (MUX)

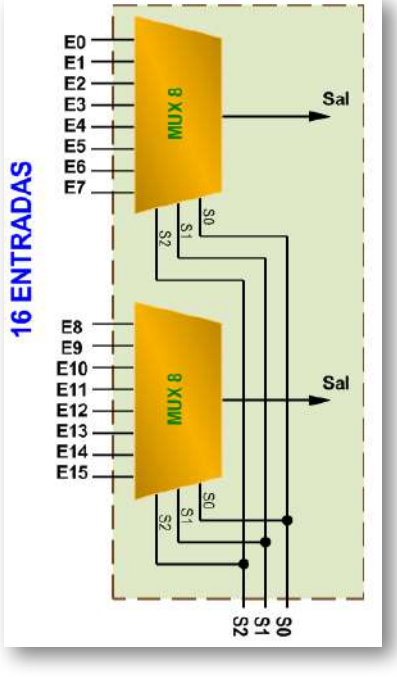
★ **Ejemplo 1:**

Diseñar un Multiplexor de 16 canales de entrada, utilizando MUX de 8 canales de entrada; luego implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

Se requieren inicialmente 2 MUX de 8 canales de entrada cada uno conformando el primer nivel (la misma columna) y de esta forma completar las 16 entradas que solicita el enunciado. Para estos 2 MUX se unen las 2 líneas selectoras **S0**, las 2 líneas **S1** y las 2 líneas **S2**, ya que pertenecen al mismo nivel (la misma columna). Donde **S0** es el **LSB (Bit Menos Significativo)** y **S2** el **MSB (Bit Más Significativo).**

2^n = Número de entradas --> 8 entradas c/u
n = líneas de control --> 3 líneas c/u (líneas selectoras)

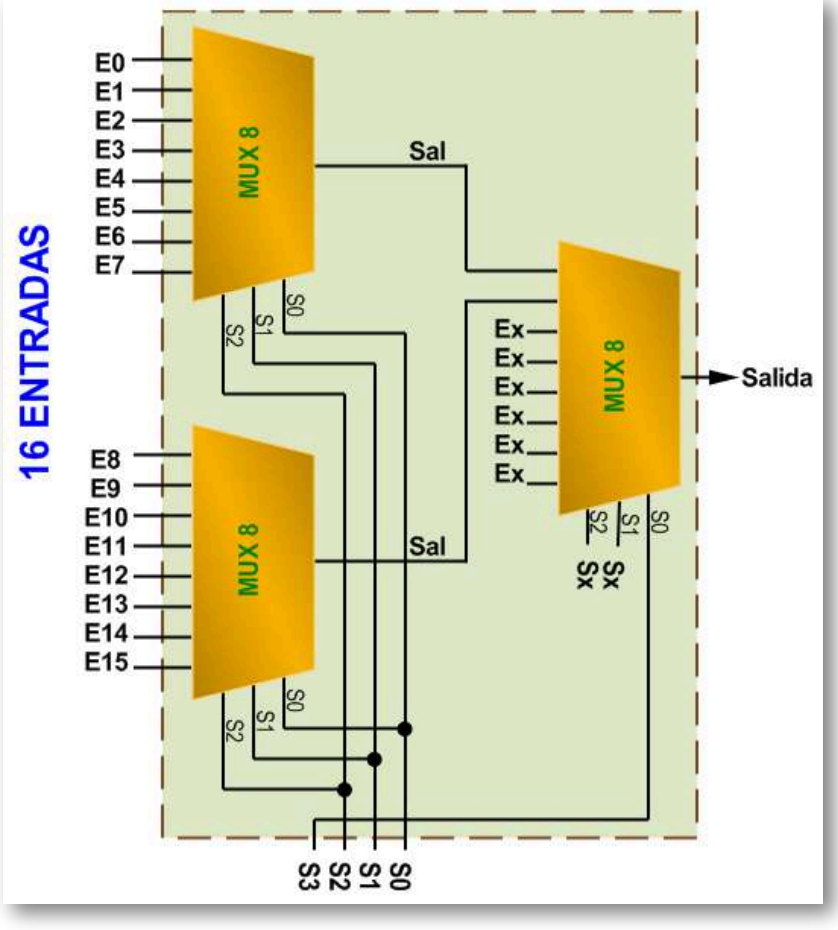
Diagrama circuital del primer nivel de los MUX:



Las 2 salidas de los MUX anteriores se conectan un tercer MUX para obtener una sola salida. De este MUX de 8 entradas solo se requieren utilizar 2 entradas, de las 3 líneas selectoras se requiere solo la línea S0 (LSB) y la terminal de salida, que equivale a la salida general del circuito.

2^n = Número de entradas --> 2 entradas
n = líneas de control --> 1 línea (línea selectora)

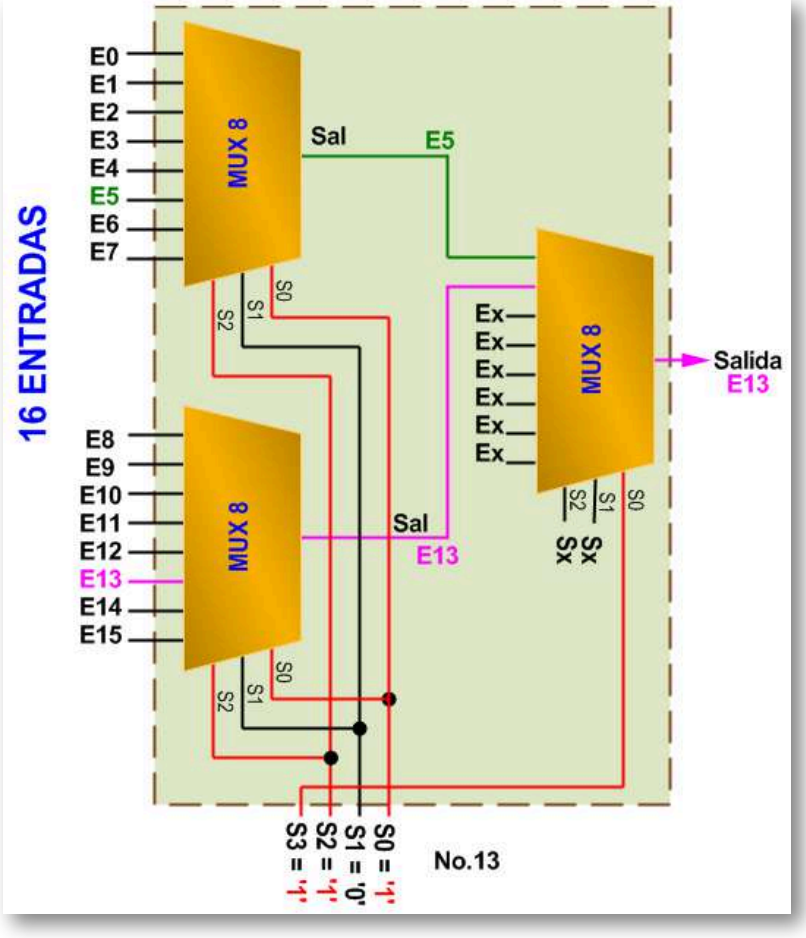
Diagrama circuital completo (Ex= entradas no asignadas, Sx= selector no asignado):



Para verificar el circuito completo, imaginemos que se requiere seleccionar el dato de la entrada 13 (E13) y transferirlo a la salida del circuito:

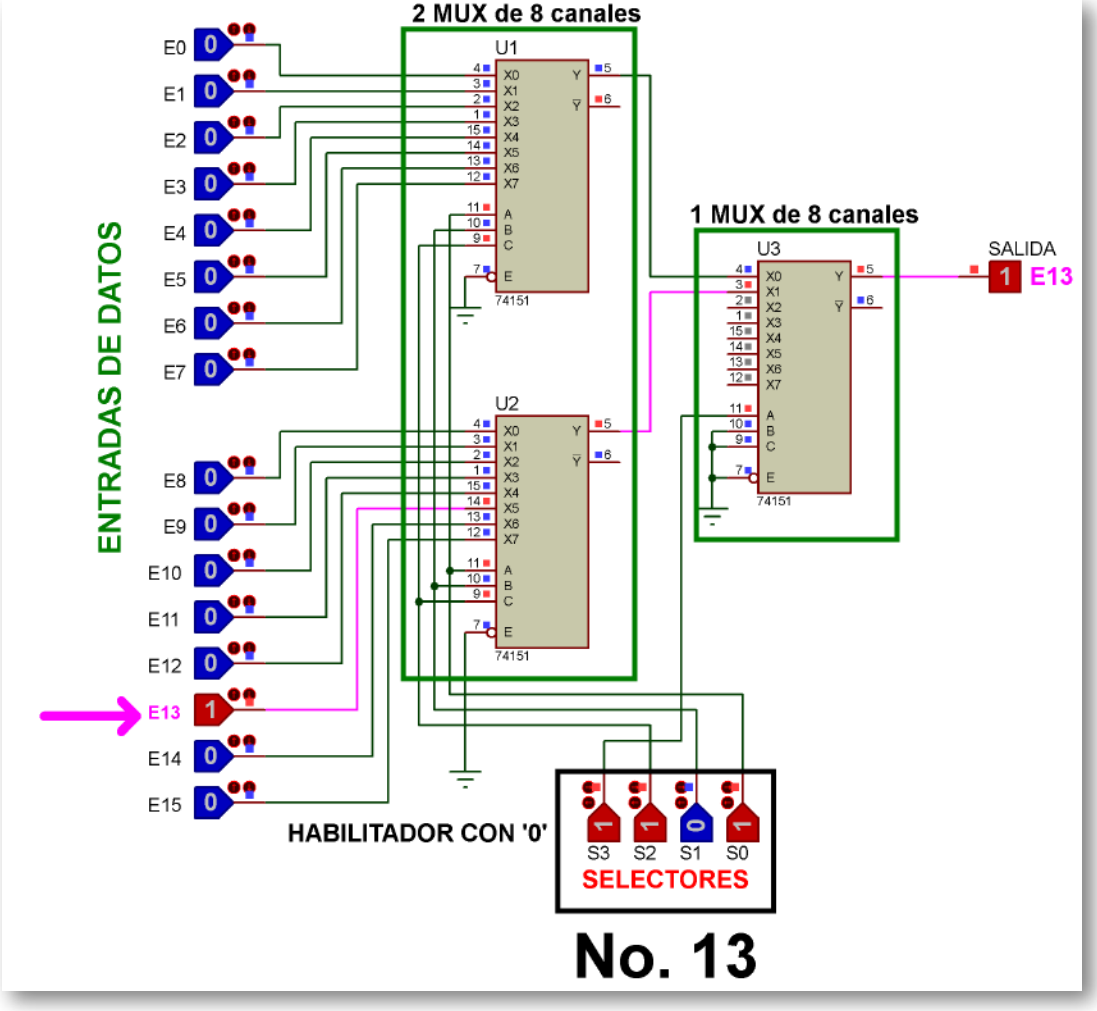
Paso 1: el número 13_d equivale a $\mathbf{1101_b}$. Para seleccionar la entrada 13, se les asigna a las líneas de selección el código binario: S3 ='1', S2 ='1', S1 ='0' y S0 ='1'.

Paso 2: como S0 ='1', S1 ='0' y S2 ='1' que equivale al número 5_d, selecciona la entrada 5 de cada MUX (E5 y E13) y los datos son transferidos a sus respectivas salidas.



Paso 3: las dos salidas se conectan a las 2 entradas del tercer MUX y por medio de la línea selectora S0 que equivale a la S3 del bloque general y que se encuentra en '1' lógico, selecciona la entrada 1 (E1), siendo esta la que proviene de la entrada E13 y es transferida a la salida.

El circuito integrado utilizado para la simulación es el [74151] y el plano esquemático es:



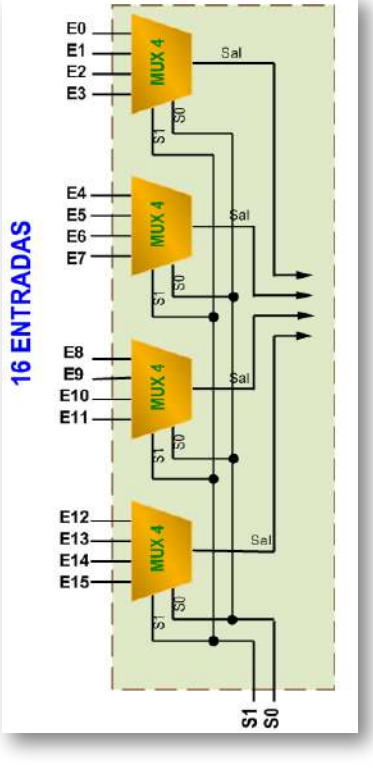
Repositorio en Github (Ejemplo_2-1-1.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

Diseñar un Multiplexor de 16 canales de entrada, utilizando MUX de 4 canales de entrada; luego implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

Se requieren inicialmente 4 MUX de 4 canales de entrada cada uno conformando el primer nivel (la misma columna) y de esta forma completar las 16 entradas que solicita el enunciado.

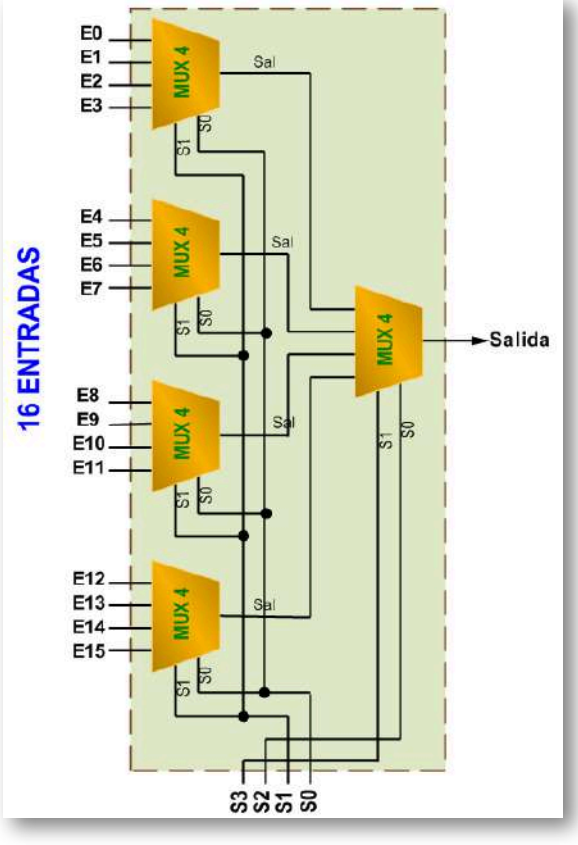
Diagrama circuital del primer nivel:



En estos 4 MUX se unen las 4 líneas selectoras **S0** y las 4 líneas **S1**, ya que pertenecen al mismo nivel (la misma columna). Donde **S0** es el **LSB (Bit Menos Significativo)** y **S1** el **MSB (Bit Más Significativo).**

2^n = Número de entradas --> 4 entradas c/u
n = líneas de control --> 2 líneas c/u (líneas selectoras)

Diagrama completo del circuito:

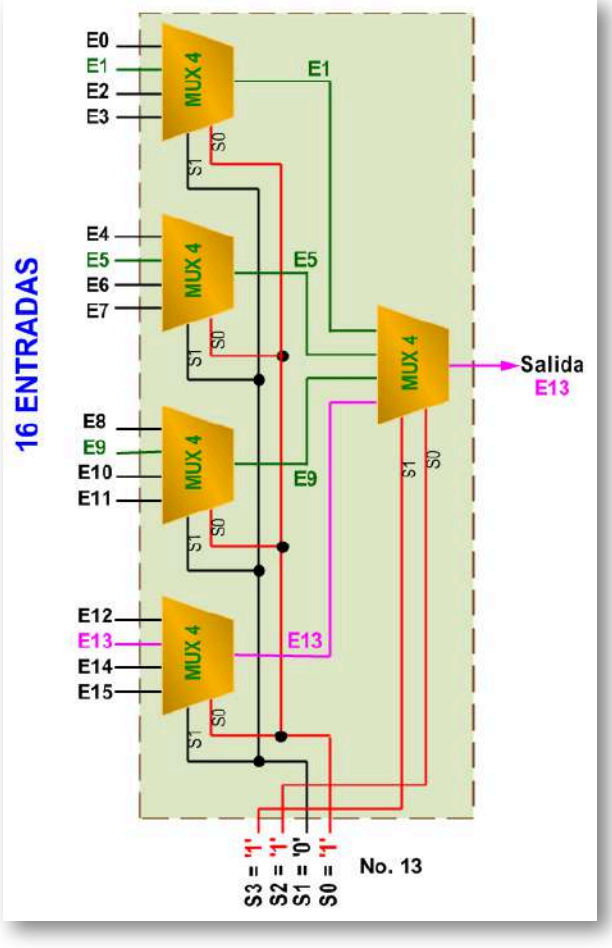


Como se tienen 4 salidas, se requiere un quinto MUX de 4 entradas para obtener una sola salida. De este MUX se utilizan las 4 entradas, las 2 líneas selectoras: la línea S0 (LSB) y la línea S1 (MSB) y la terminal de salida, que equivale a la salida general del circuito.

2^n = Número de entradas --> 4 entradas
n = líneas de control --> 1 línea (línea selectora)

Para verificar el circuito completo, imaginemos que se requiere seleccionar el dato de la entrada 13 (E13) y transferirlo a la salida del circuito:

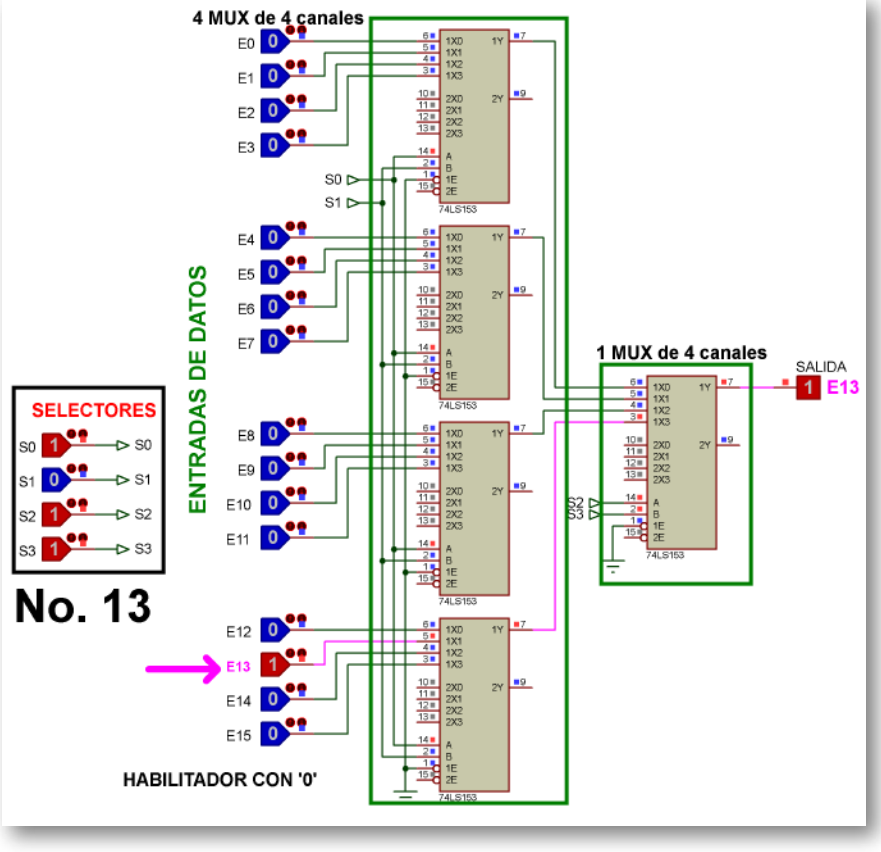
Paso 1: el número 13_d equivale a **1101_b**. Para seleccionar la entrada 13, se les asigna a las líneas de selección el código binario: S3 ='1', S2 ='1', S1 ='0' y S0='1'.



Paso 2: como S0 ='1' y S1 ='0' que equivale al número 1_d, selecciona la entrada 1 de cada MUX (E1, E5, E9 y E13) y los datos son transferidos a sus respectivas salidas.

Paso 3: las 4 salidas se conectan a las 4 entradas del quinto MUX y por medio de las líneas selectoras S0 que equivale a la S2 y S1 que equivale a S3 del bloque general y que se encuentran ambas en '1' lógicos, selecciona la entrada 3 (E3), siendo esta la que proviene de la entrada E13 y es transferida a la salida como se observa en la figura.

El circuito integrado utilizado para la simulación es el 74153 y el plano esquemático es:



Repositorio en Github (Ejemplo_2-2-2.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

Diseñar un circuito combinacional con un Multiplexor que cumpla la función lógica: $F = \overline{B} \times C + A \times B \times \overline{C}$, donde la variable 'A' es la de menor peso (LSB); luego implemente el circuito con un software de simulación electrónica utilizando el circuito integrado MUX 74153 y verifique su funcionamiento.

Se observa que la función es una expresión dada por Minterm (SOP) a la cual se le hallan las salidas que cumplan cada uno de los términos y se diseña la tabla de verdad completa:

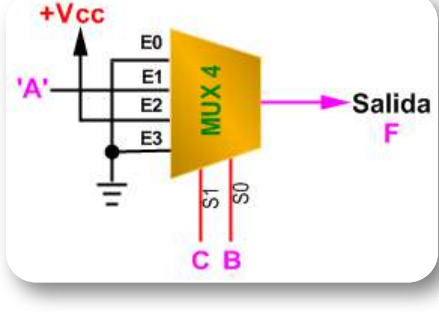
Variables Función			Salida	Término
C	B	A	F	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$A \times B \times \overline{C}$
1	0	0	1	$\overline{B} \times C$
1	0	1	1	
1	1	0	0	
1	1	1	0	

Las variables B y C son las que controlan las dos líneas de Selección del MUX de 4 entradas y por ello se analiza en la tabla de verdad cada par de combinaciones en que estas variables son iguales:

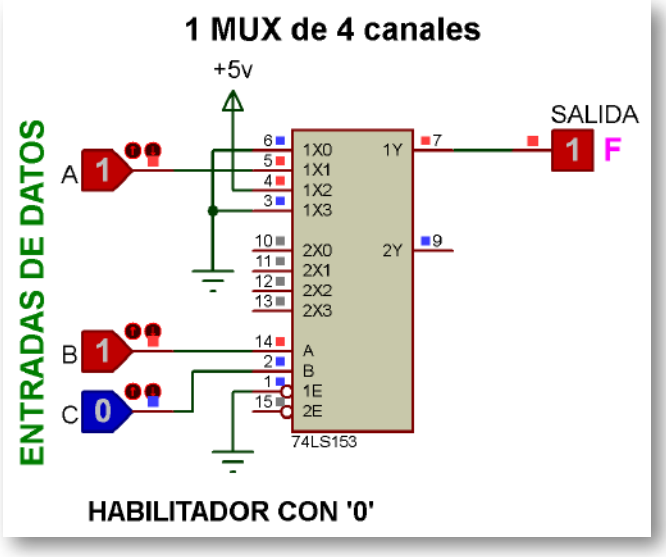
Variables Función			Salida	Entradas
C	B	A	F	MUX
0	0	0	0	E0 = '0'
0	0	1	0	
0	1	0	0	E1 = 'A'
0	1	1	1	
1	0	0	1	E2 = '1'
1	0	1	1	
1	1	0	0	E3 = '0'
1	1	1	0	

- ***E0 = '0':*** para la combinación lógica donde B y C = '0' (fondo amarillo), la Salida siempre toma el valor de '0' independiente del valor que tenga la variable A
- ***E1 =*** A***:*** para la combinación lógica donde B = '1' y C = '0' (fondo azul), la Salida toma el valor que tiene la variable A
- ***E2 = '1':*** para la combinación lógica donde B = '0' y C = '1' (fondo rojo), la Salida siempre toma el valor de '1' independiente del valor que tenga la variable A
- ***E3 = '0':*** para la combinación lógica donde B y C = '1' (fondo verde), la Salida siempre toma el valor de '0' independiente del valor que tenga la variable A

Diagrama en bloques del circuito:



Circuito simulado con el software Proteus:



Repositorio en Github (Ejemplo_2-2-3.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 4:

Diseñar un circuito combinacional con un Multiplexor de 8 entradas que cumpla la función lógica, donde la variable 'D' es la de mayor peso (MSB); luego implemente el circuito con un software de simulación electrónica utilizando el circuito integrado MUX 74151 y verifique su funcionamiento.

$$F = (\overline{A} \times \overline{B} \times \overline{C} \times D) + (\overline{A} \times \overline{B} \times C \times D) +$$

$$(\overline{A} \times B \times C \times \overline{D}) + (\overline{A} \times B \times \overline{C} \times D) + (\overline{A} \times B \times \overline{C} \times \overline{D}) +$$

$$(A \times B \times C \times D) + (A \times \overline{B} \times \overline{C} \times D) + (A \times B \times C \times \overline{D})$$

La función es una expresión dada por Minterm (SOP) a la cual se le hallan las salidas que cumplan cada uno de los términos y se diseña la tabla de verdad completa:

Variables Función				Salida	Término
D	C	B	A	F	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	$\bar{A} \times B \times \bar{C} \times \bar{D}$
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	0	
0	1	1	0	1	$\bar{A} \times B \times C \times \bar{D}$
0	1	1	1	1	$A \times B \times C \times \bar{D}$
1	0	0	0	1	$\bar{A} \times \bar{B} \times \bar{C} \times D$
1	0	0	1	1	$A \times \bar{B} \times \bar{C} \times D$
1	0	1	0	1	$\bar{A} \times B \times \bar{C} \times D$
1	0	1	1	0	
1	1	0	0	1	$\bar{A} \times \bar{B} \times C \times D$
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	$A \times B \times C \times D$

Las variables D, B y C son las que controlan las tres líneas de Selección del MUX de 8 entradas y por ello se analiza en la tabla de verdad cada par de combinaciones en que estas variables son iguales

- ***E0 = '0':*** para la combinación lógica donde B, C y D = '0' (fondo amarillo), la Salida siempre toma el valor de '0' independiente del valor que tenga la variable A.

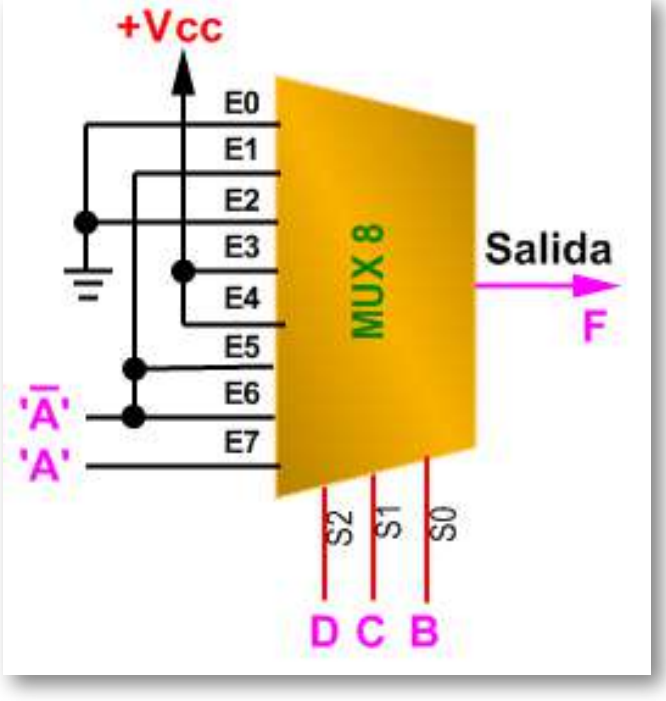
- ***E1 =*** $\overline{A}$***:*** para la combinación lógica donde B = '1', C = '0' y D = '0' (fondo azul), la Salida toma el valor que tiene la variable A pero negada

Variables Función				Salida	Término
D	C	B	A	F	
0	0	0	0	0	E0 = '0'
0	0	0	1	0	
0	0	1	0	1	E1 = $\overline{A}$
0	0	1	1	0	
0	1	0	0	0	E2 = '0'
0	1	0	1	0	
0	1	1	0	1	E3 = '1'
0	1	1	1	1	
1	0	0	0	1	E4 = '1'
1	0	0	1	1	
1	0	1	0	1	E5 = $\overline{A}$
1	0	1	1	0	
1	1	0	0	1	E6 = $\overline{A}$
1	1	0	1	0	
1	1	1	0	0	E7 = A
1	1	1	1	1	

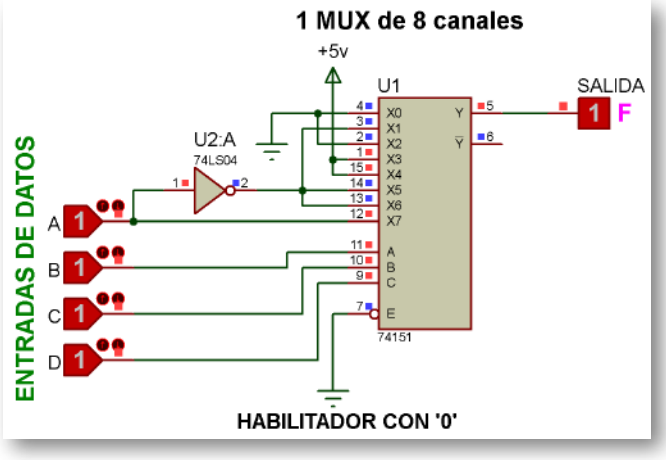
- ***E2 = '0':*** para la combinación lógica donde B = '0', C = '1' y D = '0' (fondo rojo), la Salida siempre toma el valor de '0' independiente del valor que tenga la variable A
- ***E3 = '1':*** para la combinación lógica donde B y C = '1' y D = '0' (fondo verde), la Salida siempre toma el valor de '1' independiente del valor que tenga la variable A

- ***E4 = '1':*** para la combinación lógica donde B y C = '0' y D = '1' (fondo blanco), la Salida siempre toma el valor de '1' independiente del valor que tenga la variable A
- ***E5 =*** $\overline{A}$***:*** para la combinación lógica donde B y D = '1' y C = '0' (fondo lila), la Salida toma el valor que tiene la variable A pero negada
- ***E6 =*** $\overline{A}$***:*** para la combinación lógica donde B = '0', C y D = '1' (fondo café), la Salida toma el valor que tiene la variable A pero negada
- ***E7 =*** A***:*** para la combinación lógica donde B, C y D = '1' (fondo fucsia), la Salida toma el valor que tiene la variable A

Diagrama en bloques del circuito:



Circuito simulado con el software Proteus:



Repositorio en [Github (Ejemplo_2-2-4.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 5:

Diseñar un circuito Multiplexor cuya entrada sea el número del mes del año en código binario y la salida será '1' para los meses que tienen 31 días y '0' para los meses que tiene 30 días o menos; luego implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

Inicialmente, se diseña una tabla de verdad con la información entregada en el enunciado. Como no existen los meses: 0, 13, 14 y 15 se les asigna en la salida una 'X' que representa un estado de indiferencia.

Las variables 'D', 'B' y 'C' son las que controlan las tres líneas de Selección del MUX de 8 entradas y por ello se analizan en la tabla de verdad cada par de combinaciones en que estas variables son iguales.

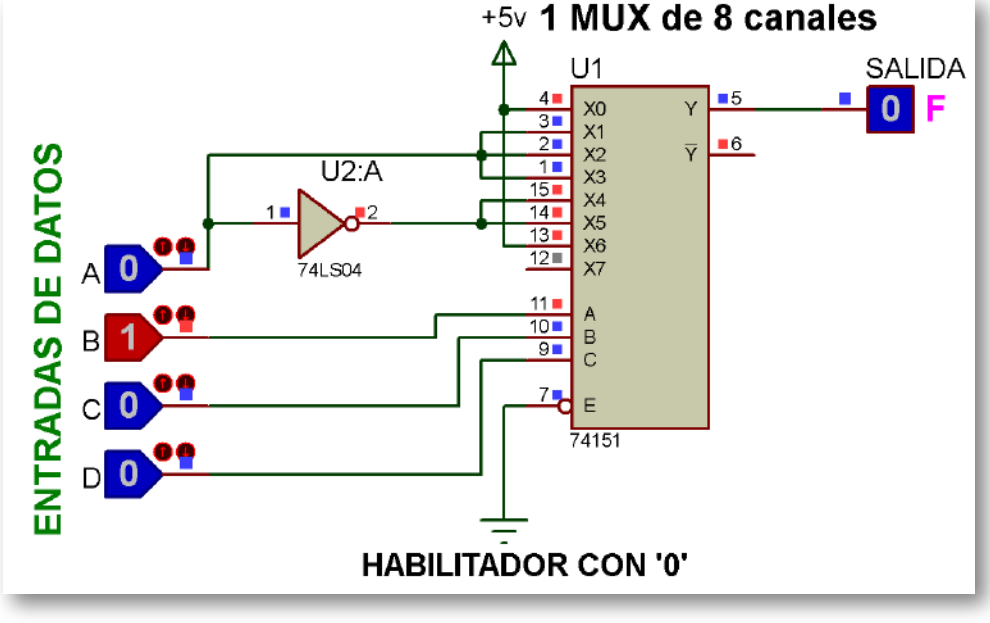
Tabla de verdad completa:

Nombre	Entradas				Salida	
MES	D	C	B	A	S	
-	0	0	0	0	X	*E0 = '1'*
Enero	0	0	0	1	1	
Febrero	0	0	1	0	0	*E1 = A*
Marzo	0	0	1	1	1	
Abril	0	1	0	0	0	*E2 = A*
Mayo	0	1	0	1	1	
Junio	0	1	1	0	0	*E3 = A*
Julio	0	1	1	1	1	
Agosto	1	0	0	0	1	*E4 = $\overline{A}$*
Septiembre	1	0	0	1	0	
Octubre	1	0	1	0	1	*E5 = $\overline{A}$*
Noviembre	1	0	1	1	0	
Diciembre	1	1	0	0	1	*E6 = '1'*
-	1	1	0	1	X	
-	1	1	1	0	X	*E7 = 'X'*
-	1	1	1	1	X	

- ***E0 = '1':*** para la combinación lógica donde B, C y D = '0' (fondo amarillo), la Salida toma el valor de '1' ya que la otra combinación es indiferente (X)
- ***E1 = A:*** para la combinación lógica donde B = '1', C y D = '0' (fondo azul), la Salida toma el valor que tiene la variable A
- ***E2 = A:*** para la combinación lógica donde B y D = '0', C = '1' (fondo rojo), la Salida toma el valor que tiene la variable A
- ***E3 = A:*** para la combinación lógica donde B y C = '1', D = '0' (fondo verde), la Salida toma el valor que tiene la variable A
- ***E4 = $\overline{A}$:*** para la combinación lógica donde B y C = '0' y D = '1' (fondo blanco), la Salida toma el valor que tiene la variable A pero negada

- ***E5 =*** $\overline{A}$***:*** para la combinación lógica donde B y D = '1' y C = '0' (fondo lila), la Salida toma el valor que tiene la variable A pero negada
- ***E6 = '1':*** para la combinación lógica donde B = '0', C y D = '1' (fondo café), la Salida toma el valor de '1' ya que la otra combinación es indiferente (X)
- ***E7= 'X':*** para la combinación lógica donde B, C y D = '1' (fondo fucsia), la Salida es indiferente del valor que tome

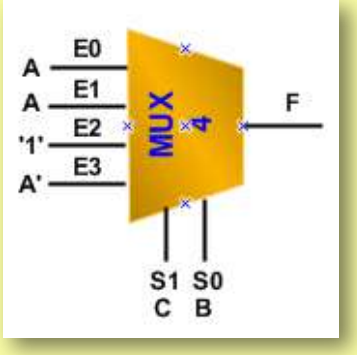
Circuito simulado con el software Proteus:



Repositorio en [Github (Ejemplo_2-2-5.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 6:

Con base en el siguiente esquema del circuito Multiplexor de 4 canales, donde la variable 'A' es el LSB y la variable 'C' es el MSB. Hallar la tabla de verdad completa, su función lógica y verificar la solución con un software de simulación electrónica.



Inicialmente se diseña la tabla con las combinaciones de las variables de entrada teniendo en cuenta que el orden es: A= LSB y C= MSB. Del esquema se toman los datos de cada entrada al MUX y de las líneas de control y estos se ubican en las salidas correspondientes.

Por ejemplo: para la primera combinación de la tabla, las líneas de control están en S0='0' y S1='0', la variable A está en '0' y luego en '1' y como E0 vale A en las salidas se colocan los mismos valores de la variable A.

Entradas			Salida	E
C	B	A	F	Mux
0	0	0	0	E0
0	0	1	1	
S1	S0	A		

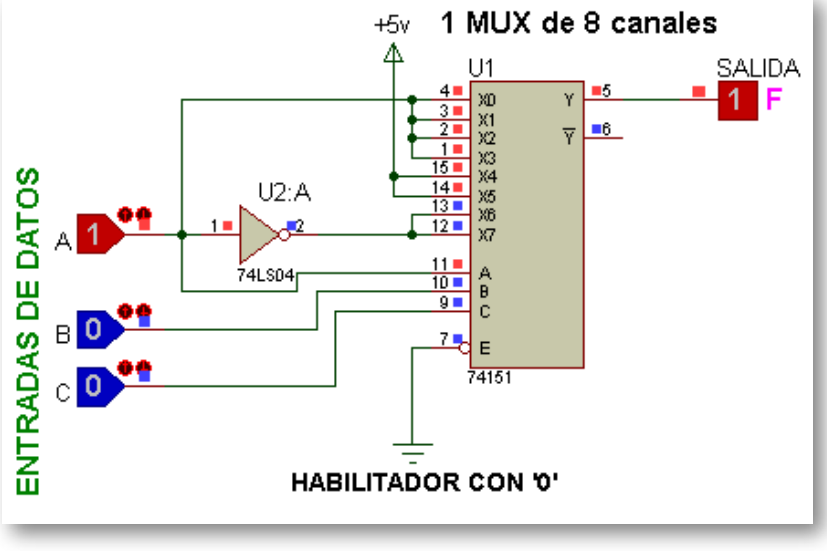
La tabla de verdad completa es:

Entradas			Salida	E
C	B	A	F	Mux
0	0	0	0	E0
0	0	1	1	
0	1	0	0	E1
0	1	1	1	
1	0	0	1	E2
1	0	1	1	
1	1	0	1	E3
1	1	1	0	

La función lógica solucionada por Maxterm (∏) es:

$$F(\Pi) = (A + B + C) \times (A + \overline{B} + C) \times (\overline{A} + \overline{B} + \overline{C})$$

Imagen de la simulación:

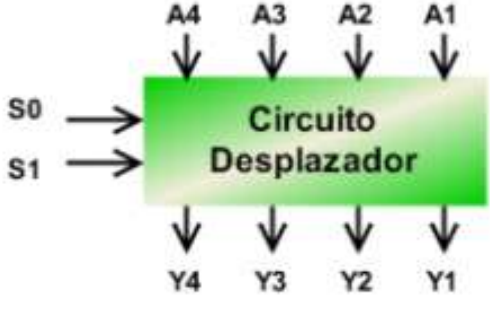


Repositorio en Github (Ejemplo_2-2-6.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 7:

Diseñar un desplazador de 4 bits a la derecha utilizando Multiplexores de 4 canales. Suponga que los MSB se llenan con '0'. Verificar la solución con un software de simulación electrónica.

El diagrama en bloque consta de 4 entradas, 4 salidas y sus 2 líneas de selección:

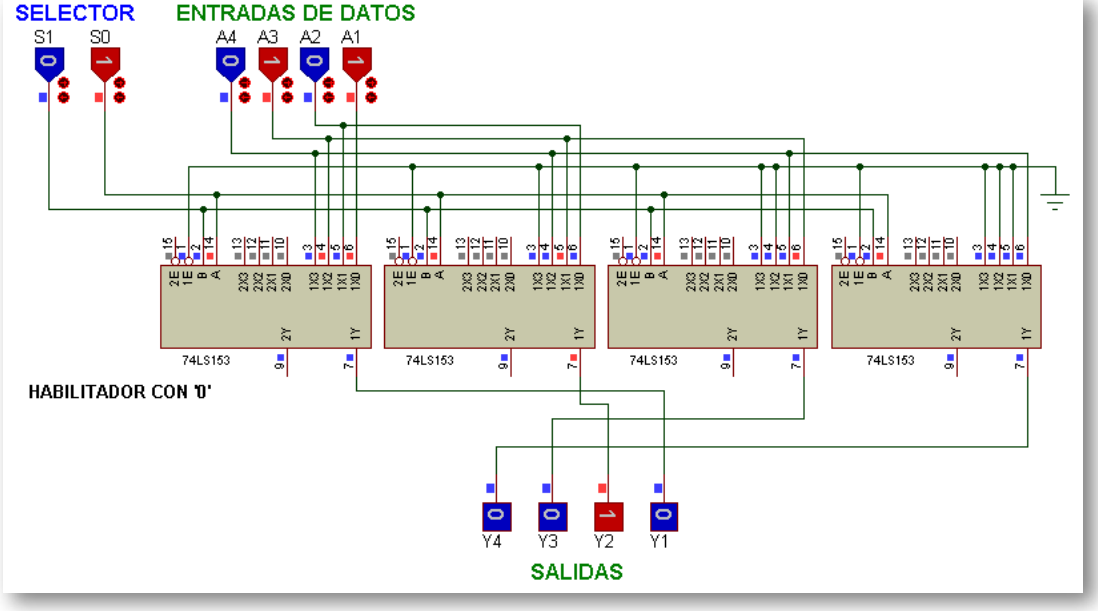


Se diseña la tabla de verdad:

Selector		Salidas			
S1	S0	Y4	Y3	Y2	Y1
0	0	A4	A3	A2	A1
0	1	0	A4	A3	A2
1	0	0	0	A4	A3
1	1	0	0	0	A4

Cada salida **Y** se obtiene de cada Multiplexor de 4 canales, donde se unen las líneas de selección **S0** y **S1**, el diagrama sería el siguiente:

Imagen de la simulación en Proteus:



Repositorio en Github (Ejemplo_2-2-7.dsn) con el archivo de la simulación con Proteus v7.8

2.3 Demultiplexor (DEMUX)

Los circuitos internos de un Demultiplexor (DEMUX) y un Decodificador (DECODER) son iguales, solo difieren en los nombres de algunas terminales, en el DEMUX: líneas de selección y entrada, en el DECODER: líneas de dirección y habilitador (Enable=E).

★ Ejemplo 1:

Diseñar un Demultiplexor de 16 canales de salida, utilizando DEMUX de 8 canales de salida; luego implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

n = líneas de control --> 1 línea (línea selectora)
2^n = Número de salidas --> 2 salidas

De las 8 salidas del primer DEMUX solo se utilizarán 2 y de las 3 líneas selectoras se empleará sola la Sel0 (LSB) que equivale a la línea selectora de mayor peso (MSB) en las "LÍNEAS DE CONTROL GENERAL" como se observa en la siguiente figura.

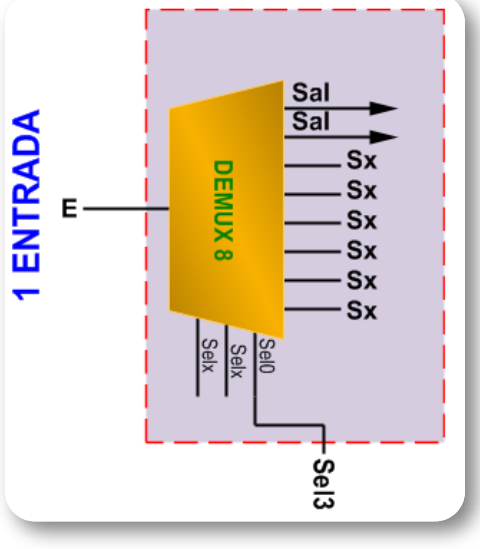
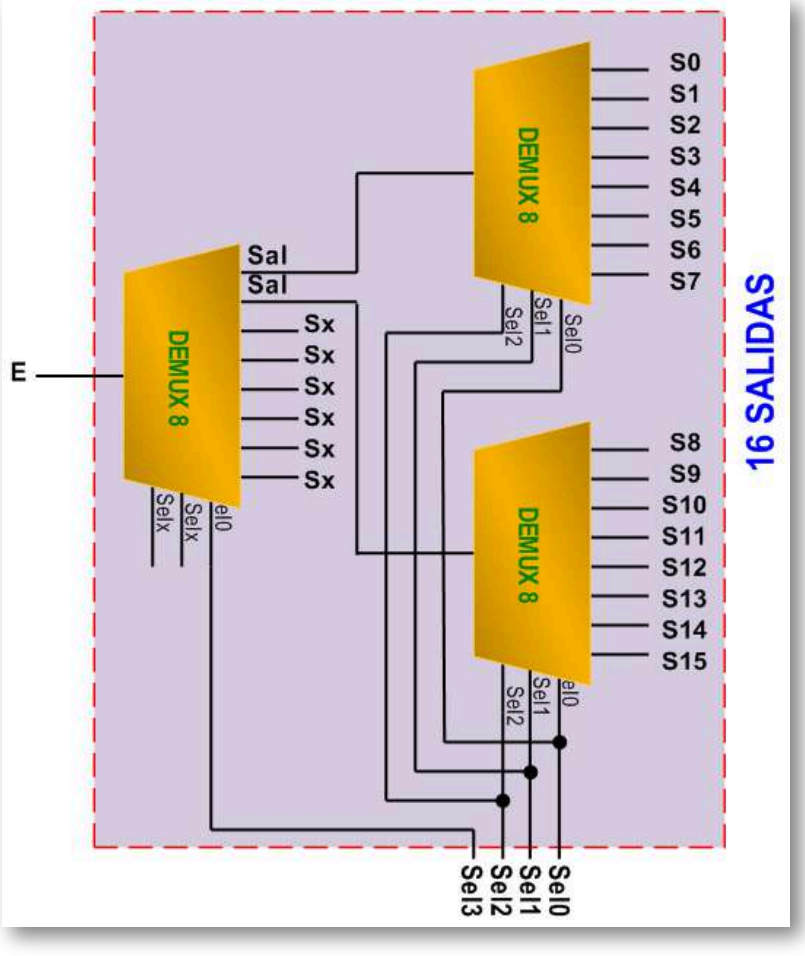


Diagrama circuital del primer nivel:

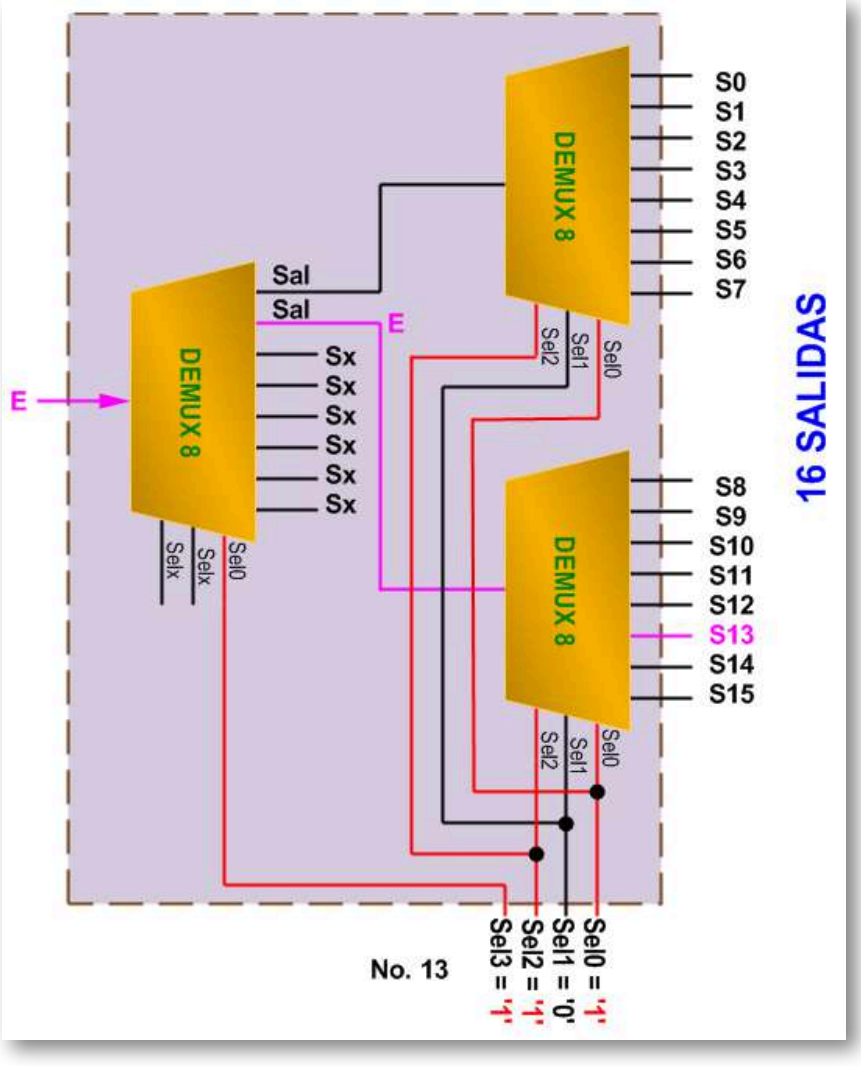
Las 2 salidas son conectadas a las entradas de 2 DEMUX de 8 canales de salida cada uno, que conforman el segundo nivel (la misma columna), completando las 16 salidas que solicita el enunciado. Para estos 2 DEMUX se unen las 2 líneas selectoras **Sel0**, las 2 líneas **Sel1** y las 2 líneas **Sel2**, que son equivalentes respectivamente a las líneas selectoras Sel0, Sel1 y Sel2 de las "LÍNEAS DE CONTROL GENERAL" como se observa en la siguiente figura.

n = líneas de control --> 3 líneas c/u (líneas selectoras)
2^n = Número de salidas --> 8 salidas c/u



Para verificar el circuito completo, imaginemos que se requiere que el dato de entrada sea transferido a la salida 13 (S13):

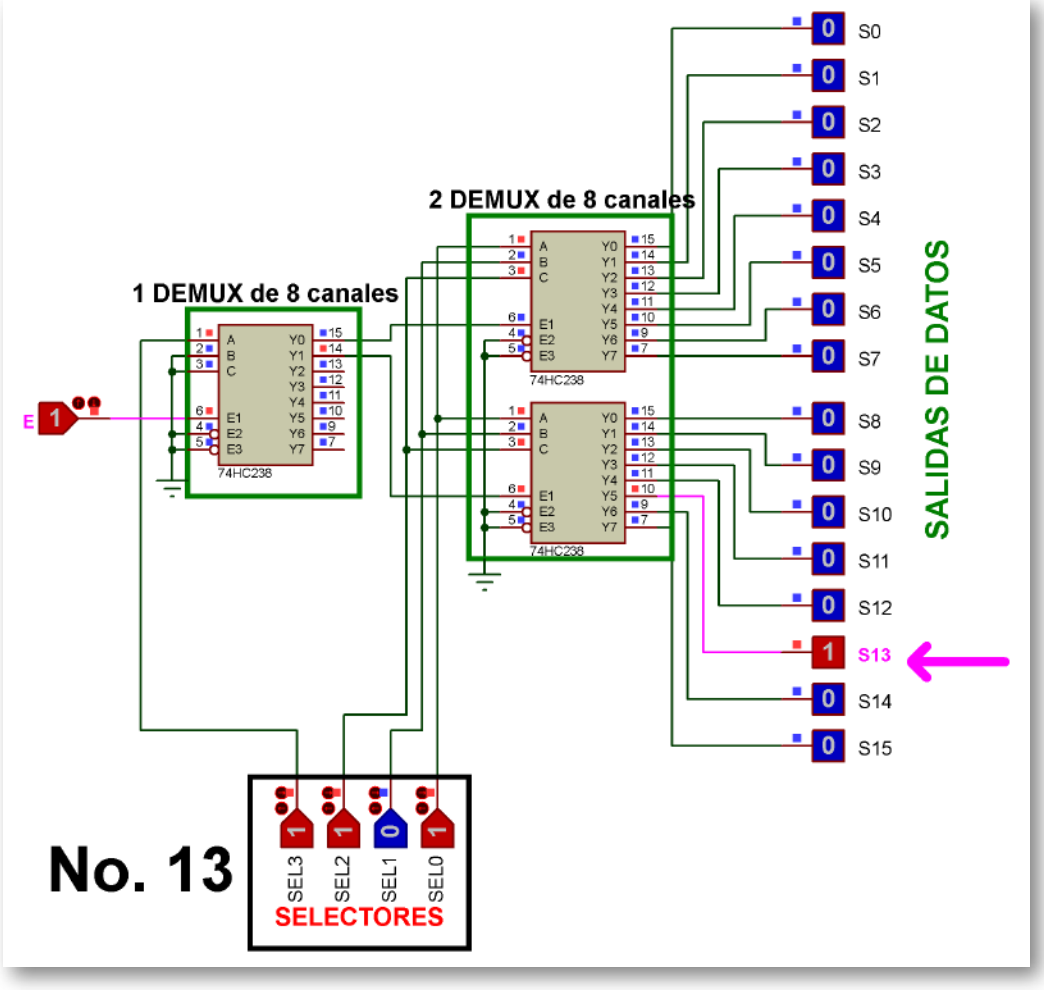
Paso 1: el número 13_d equivale al código $\mathbf{1101_b}$; por ello, los niveles lógicos de las líneas selectoras generales son: Sel3 ='1', Sel2 ='1', Sel1 ='0' y Sel0='1'; como Sel3 tiene un '1' lógico transfiere el dato de entrada a la salida S1 del primer DEMUX, como se indica en la figura.



Paso 2: las otras líneas de selección Sel2 ='1', Sel1 ='0' y Sel0='1', se seleccionan la salida 5 de cada DEMUX del segundo nivel (S5 y S13), pero es el DEMUX inferior el que tiene el dato de la entrada, el cual es transferido a la salida S13 del circuito.

El circuito integrado utilizado para la simulación es el 74HC238 y el plano esquemático es:

La implementación también se puede realizar con el circuito integrado 74138, pero su salida es activa en Bajo.



Repositorio en Github (Ejemplo_2-3-1.dsn) con el archivo de la simulación con Proteus v7.8

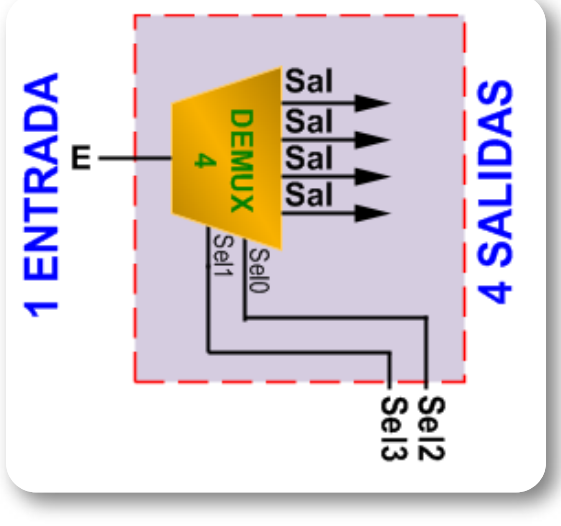
★ Ejemplo 2:

Diseñar un Demultiplexor de 16 canales de salida, utilizando DEMUX de 4 canales de salida; luego implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

El primer nivel está conformado por un DEMUX de 4 canales de salida y 2 líneas selectoras de control, siendo la línea Sel0 la del LSB y la línea Sel1 la del MSB que a su vez son equivalentes respectivamente a las líneas selectoras Sel2 y Sel3 de las "LÍNEAS DE CONTROL GENERAL" como se observa en la siguiente figura.

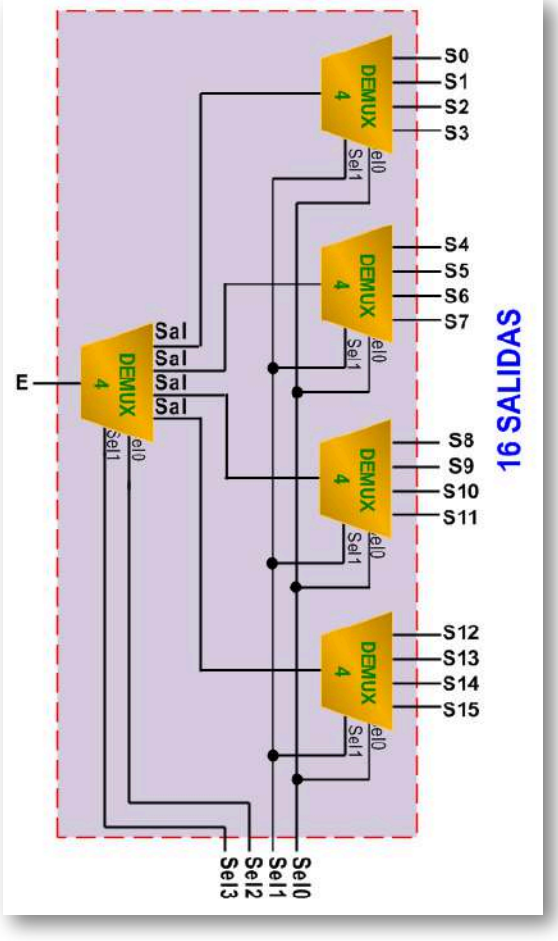
n = líneas de control --> 2 líneas (líneas selectoras)
2^n = Número de salidas --> 4 salidas

Diagrama circuital del primer nivel:



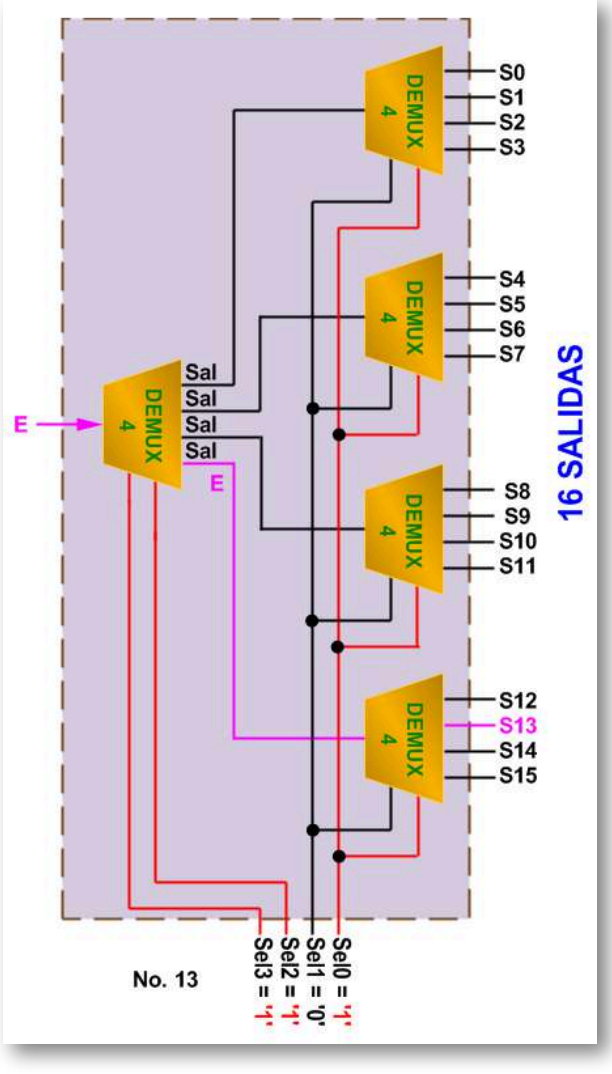
Las 4 salidas son conectadas a las entradas de 4 DEMUX de 4 canales de salida cada uno que conforman el segundo nivel (la misma columna), completando así las 16 salidas que solicita el enunciado. Para estos 4 DEMUX se unen las 4 líneas selectoras **Sel0** y las 4 líneas **Sel1**, que son equivalentes respectivamente a las líneas selectoras Sel0 y Sel1 de las "LÍNEAS DE CONTROL GENERAL" como se observa en la siguiente figura.

n = líneas de control --> 2 líneas (líneas selectoras c/u)
2^n = Número de salidas --> 4 salidas c/u



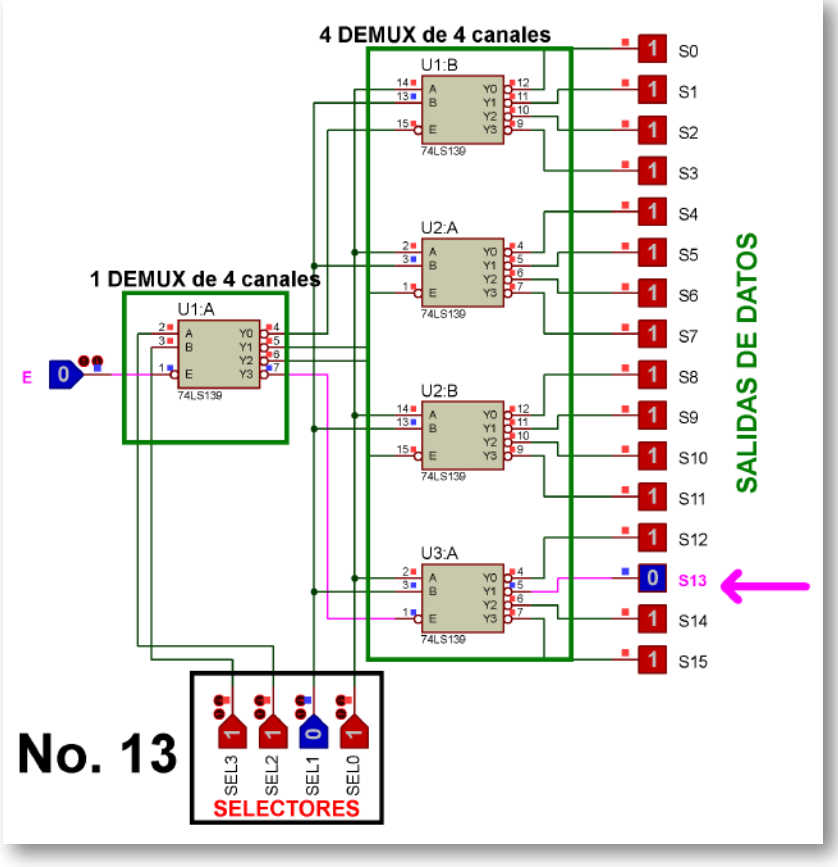
Para verificar el circuito completo, imaginemos que se requiere que el dato de entrada sea transferido a la salida 13 (S13):

Paso 1: el número 13_d equivale al código $\mathbf{1101_b}$; que corresponden a los niveles lógicos de las líneas selectoras generales: Sel3 ='1', Sel2 ='1', Sel1 ='0' y Sel0='1'; como Sel2 y Sel3 tienen un '1' lógico se transfiere el dato de la entrada a la salida S3 del primer DEMUX, como se indica en la figura.



- ***Paso 2:*** las otras líneas de selección Sel0 ='1' y Sel2 ='0', se seleccionan las salidas S1 de cada DEMUX del segundo nivel que equivalen a S1, S5, S9 y S13, pero es el DEMUX inferior el que tiene el dato de la entrada general, el cual es transferido a la salida S13 del circuito.

- El circuito integrado utilizado para la simulación es el 74139, teniendo presente que la salida es en nivel Bajo. Plano esquemático:
- La implementación también se puede realizar con el circuito integrado 74HC238, utilizando solo 4 salidas de las 8 que tiene y de las 3 líneas de control usar solo 2.



Repositorio en Github (Ejemplo_2-3-2.dsn) con el archivo de la simulación con Proteus v7.8

2.4 Ejercicios propuestos con respuestas

★ Ejercicio 1:

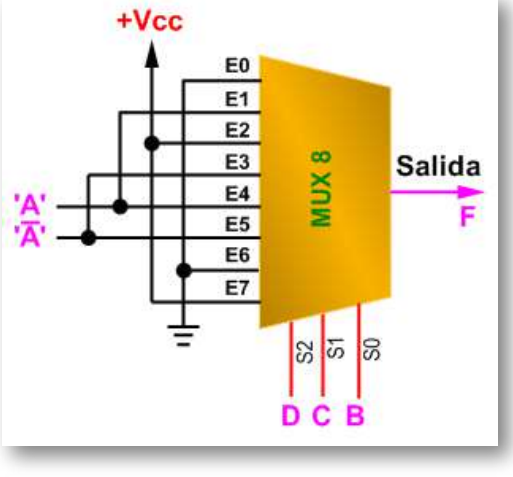
Diseñar un circuito combinacional con un Multiplexor de 8 entradas que cumpla la función lógica, donde la variable 'D' es la de mayor peso (MSB); luego implemente el circuito con un software de simulación electrónica utilizando el circuito integrado MUX 74151 y verifique su funcionamiento.

$$F = (A \times B \times \overline{C} \times \overline{D}) + (\overline{A} \times \overline{B} \times C \times \overline{D}) +$$

$$(A \times \overline{B} \times C \times \overline{D}) + (\overline{A} \times B \times C \times \overline{D}) + (A \times \overline{B} \times \overline{C} \times D) +$$

$$(\overline{A} \times B \times \overline{C} \times D) + (\overline{A} \times B \times C \times D) + (A \times B \times C \times D)$$

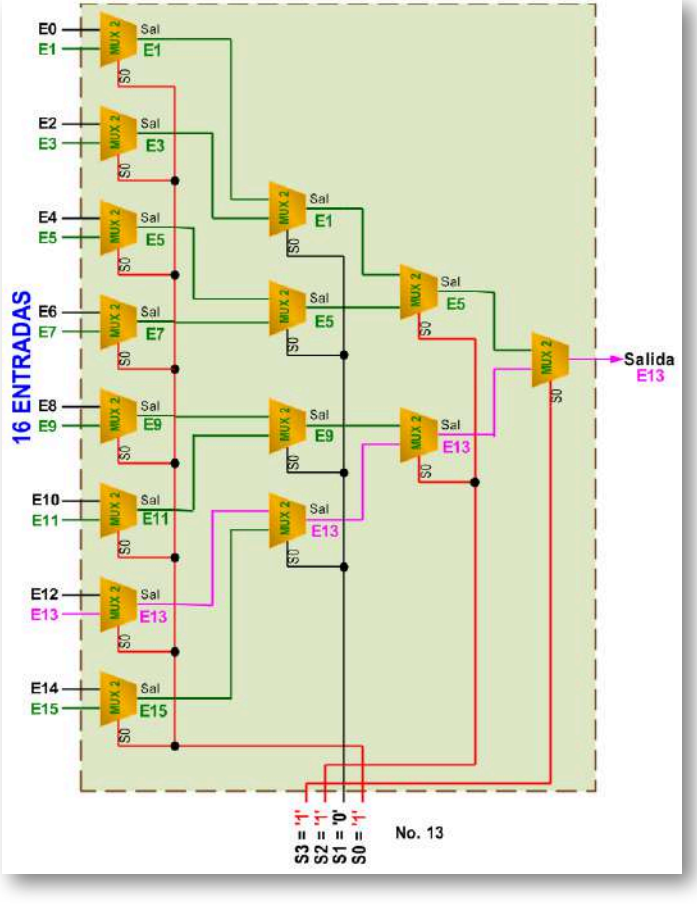
Diagrama en bloques de la solución:



★ Ejercicio 2:

Diseñar un Multiplexor de 16 canales de entrada, utilizando MUX de 2 canales de entrada; luego implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

Diagrama en bloques de la solución:

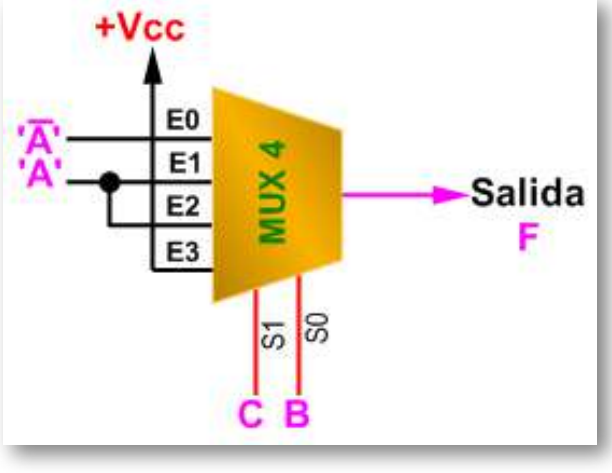


La simulación se puede realizar con el circuito integrado 74153 que contiene 2 MUX de 4 canales activos en nivel alto, pero utilizando solo 2 canales y 1 línea de selección.

★ Ejercicio 3:

Diseñar el control para una prensa que tiene 3 conmutadores, de modo que el proceso industrial se detenga por razones de seguridad exclusivamente cuando se pulsen simultáneamente dos de ellos y cuando no sean activados, en cualquier otra circunstancia el proceso continuará se funcionamiento normal. Implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

Diagrama en bloques de la solución:



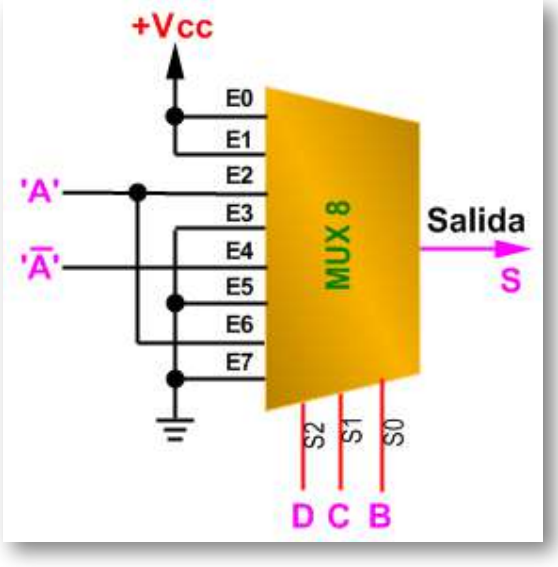
La simulación se puede realizar con el circuito integrado 74153 que contiene 2 MUX de 4 canales activos en nivel alto.

★ Ejercicio 4:

Diseñar un circuito combinacional utilizando MUX de 8 canales para un sistema que devuelva una salida en '1' si el número ingresado pertenece a la secuencia FIBONACCI y que devuelva un '0' si no lo es. Implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

Diagrama en bloques de la solución:

Secuencia Fibonacci = 0,1,1,2,3,5,8,13...



La simulación se puede realizar con el circuito integrado 74151

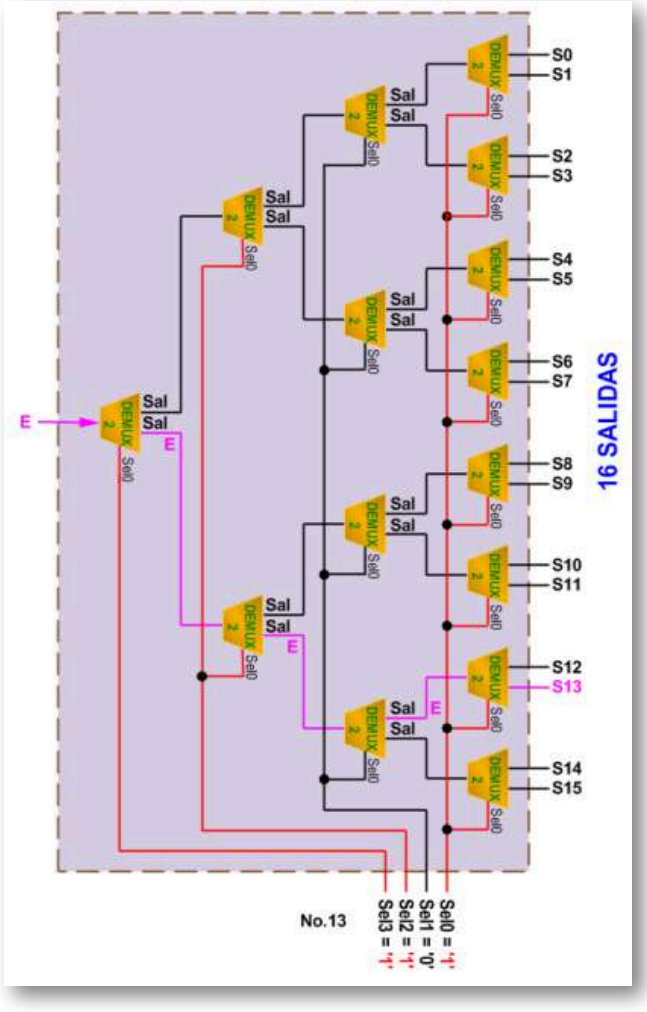
Repositorio en Github (Ejercicio_2-4-4.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejercicio 5:

Diseñar un circuito combinacional DEMUX de 16 salidas, basado en DEMUX de 2 salidas. Implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

La simulación se puede realizar con el circuito integrado 4053

Diagrama en bloques de la solución:



★ Ejercicio 6:

Con base en el siguiente esquema del circuito Multiplexor de 4 canales, donde la variable 'A' es el LSB y la variable 'C' es el MSB. Hallar la tabla de verdad completa, su función Lógica y finalmente, verificar la solución con un software de simulación electrónica.

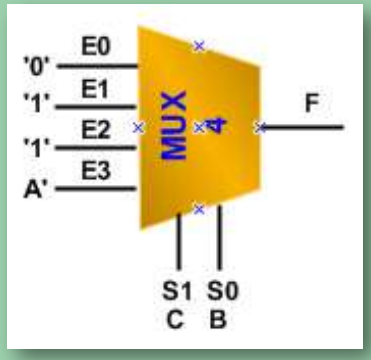


Tabla de verdad completa:

Entradas			Salida	E
C	B	A	F	Mux
0	0	0	0	E0
0	0	1	0	
0	1	0	1	E1
0	1	1	1	
1	0	0	1	E2
1	0	1	1	
1	1	0	1	E3
1	1	1	0	

Función lógica:

$$F(\Pi) = (A + B + C) \times (\overline{A} + B + C) \times (\overline{A} + \overline{B} + \overline{C})$$

Repositorio en Github (Ejercicio_2-4-6.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejercicio 7:

Con base en el siguiente esquema del circuito que consta de un Multiplexor de 4 canales, donde la variable 'A' es el LSB y la variable 'C' es el MSB. Hallar la tabla de verdad completa, su función Lógica y finalmente, verificar la solución con un software de simulación electrónica

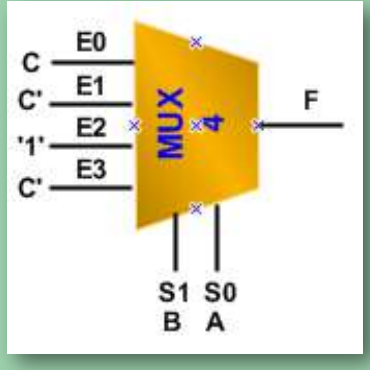


Tabla de verdad completa:

Entradas			Salida	E
C	B	A	F	Mux
0	0	0	0	E0
0	0	1	1	E1
0	1	0	1	E2
0	1	1	1	E3
1	0	0	1	E0
1	0	1	0	E1
1	1	0	1	E2
1	1	1	0	E3

Función lógica:

$$F(\Pi) = (A + B + C) \times (\overline{A} + B + \overline{C}) \times (\overline{A} + \overline{B} + \overline{C})$$

Repositorio en Github (Ejercicio_2-4-7.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejercicio 8:

Con base en el siguiente mapa de Karnaugh, donde la variable 'A' es el LSB y la variable 'C' es el MSB. Dibujar el diagrama esquemático con MUX de 4 canales y de 8 canales y hallar la tabla de verdad completa y la función lógica; finalmente, verificar la solución con un software de simulación electrónica.

A \ BC	00	01	11	10
0	0	0	1	1
1	1	1	0	1

Diagrama esquemático con MUX de 4 canales:

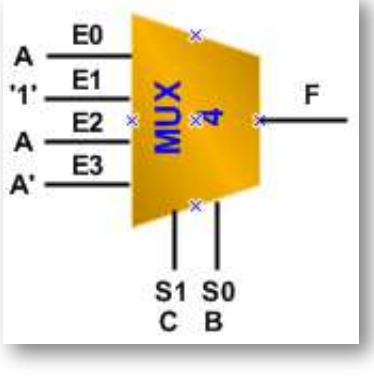


Diagrama esquemático con MUX de 8 canales:

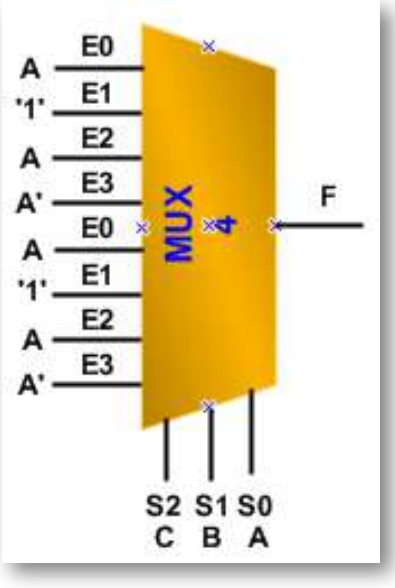


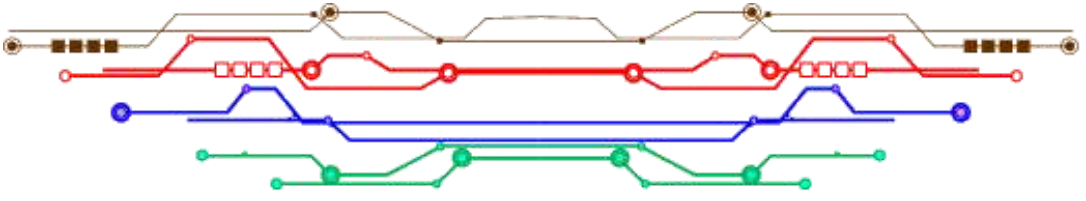
Tabla de verdad completa:

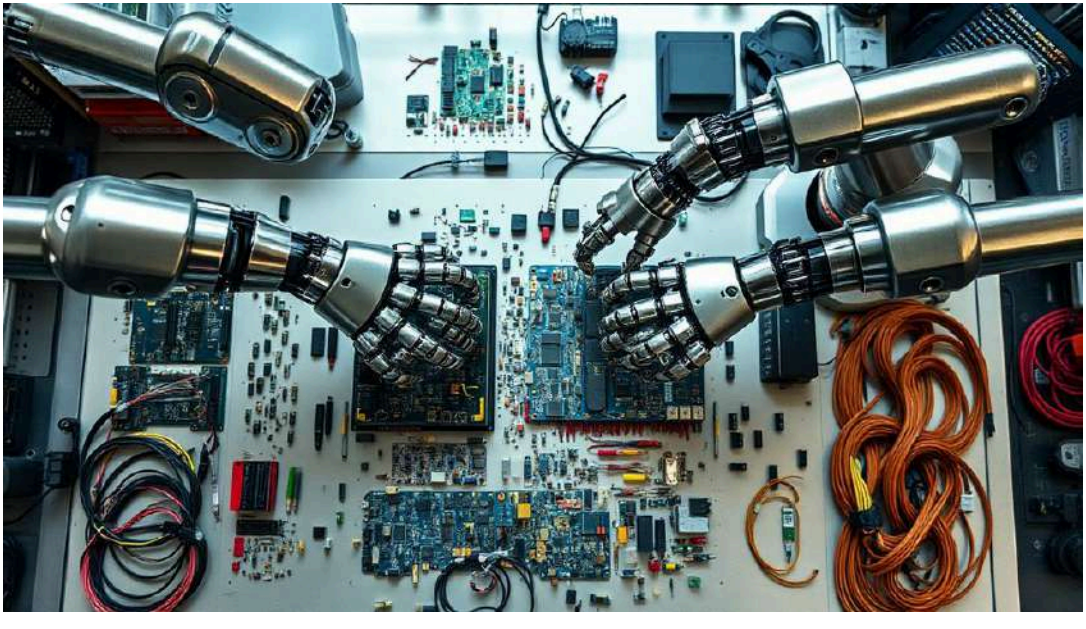
Entradas			Salida	E
C	B	A	F	Mux
0	0	0	0	E0
0	0	1	1	
0	1	0	1	E1
0	1	1	1	
1	0	0	0	E2
1	0	1	1	
1	1	0	1	E3
1	1	1	0	

Función lógica:

$$F(\Pi) = (A + B + C) \times (A + B + \overline{C}) \times (\overline{A} + \overline{B} + \overline{C})$$

Repositorio en Github (Ejercicio_2-4-8.dsn) con el archivo de la simulación con Proteus v7.8





AYUDAS

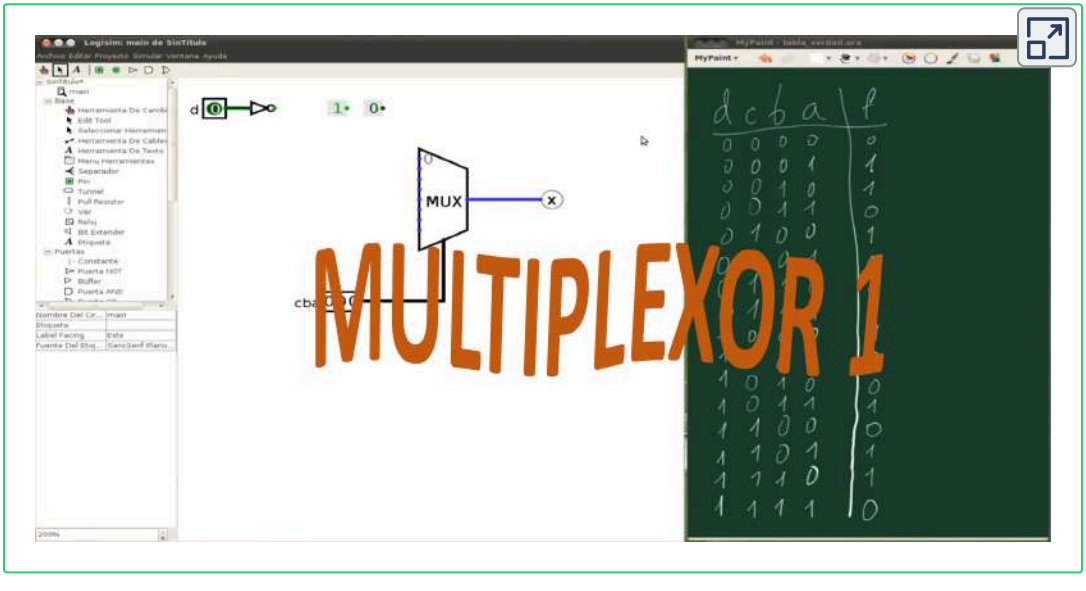
COMPLEMENTOS

2.5 Videos

Video explicativo: Multiplexores y Demultiplexores, Circuitos Combinacionales - Soluciones, en el canal de YouTube de Oscar Botero @oscarboterohenao

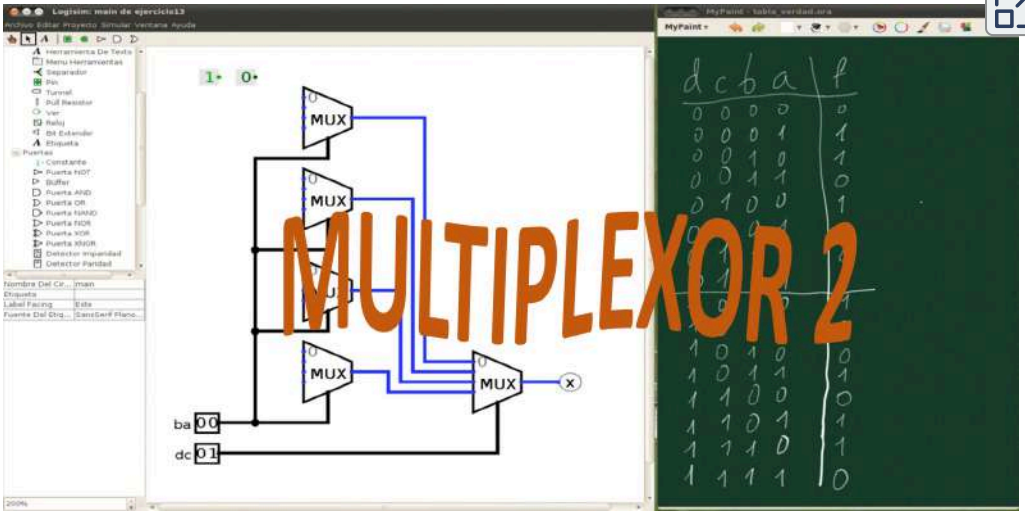
Video explicativo: Multiplexores y Demultiplexores, Circuitos Combinacionales - Simulaciones, en el canal de YouTube de Oscar Botero @oscarboterohenao

Video realizado por Agustín Borrego sobre ***"Multiplexores"***, con licencia de atribución de Creative Commons.[2]



[2] Video Youtube: multiplexores func1

Videos realizados por Agustín Borrego sobre ***"Multiplexores y Demultiplexores"***, con licencia de atribución de Creative Commons.[34]



[3] Video YouTube: multiplexores func2

[4] Video YouTube: demultiplexores

2.6 Objetos Interactivos

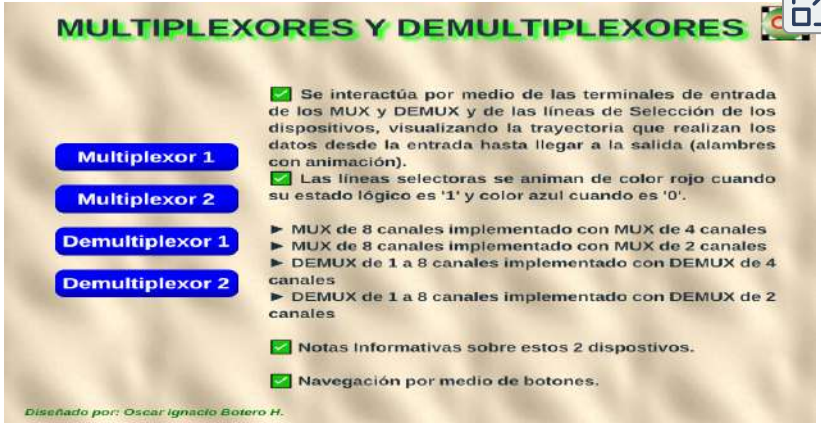
Haga clic sobre la imagen para abrir el objeto interactivo:

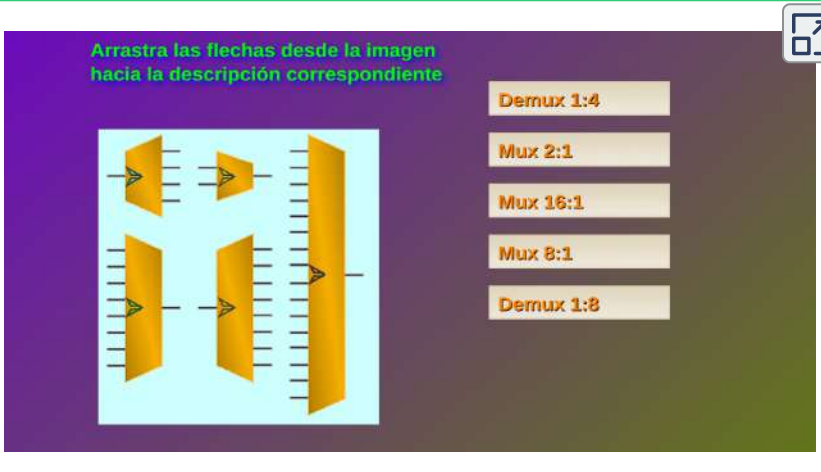
Este objeto interactivo fue generado con la asistencia de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/

Cuestionario:

Haga clic sobre en la esquina superior del objeto interactivo:

El primer Objeto Interactivo fue diseñado por Oscar Ignacio Botero donde puede interactuar por medio de los campos de entrada y las líneas de selección para observar el comportamiento de los Multiplexores y Demultiplexores. El segundo Objeto Interactivo es un emparejamiento entre los símbolos y tipos de dispositivos.[5]

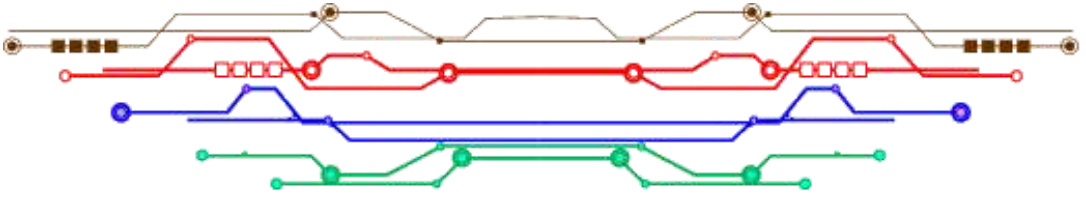


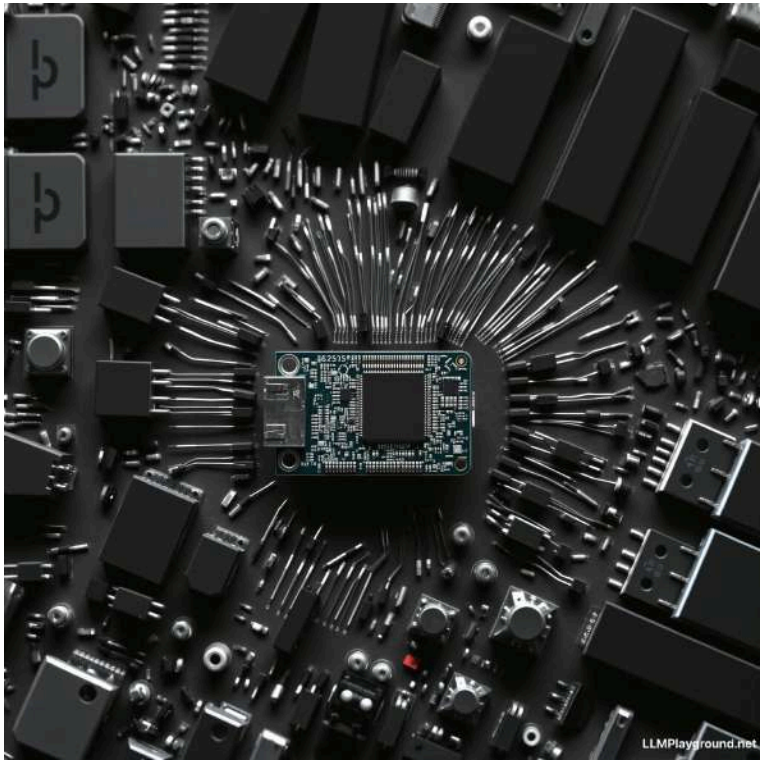


[5] La plantilla del segundo Objeto Interactivo fue obtenida del sitio web del Proyecto Descartes, intervenida por Oscar Ignacio Botero Henao. Plantilla con DescartesJS

2.7 Simulaciones

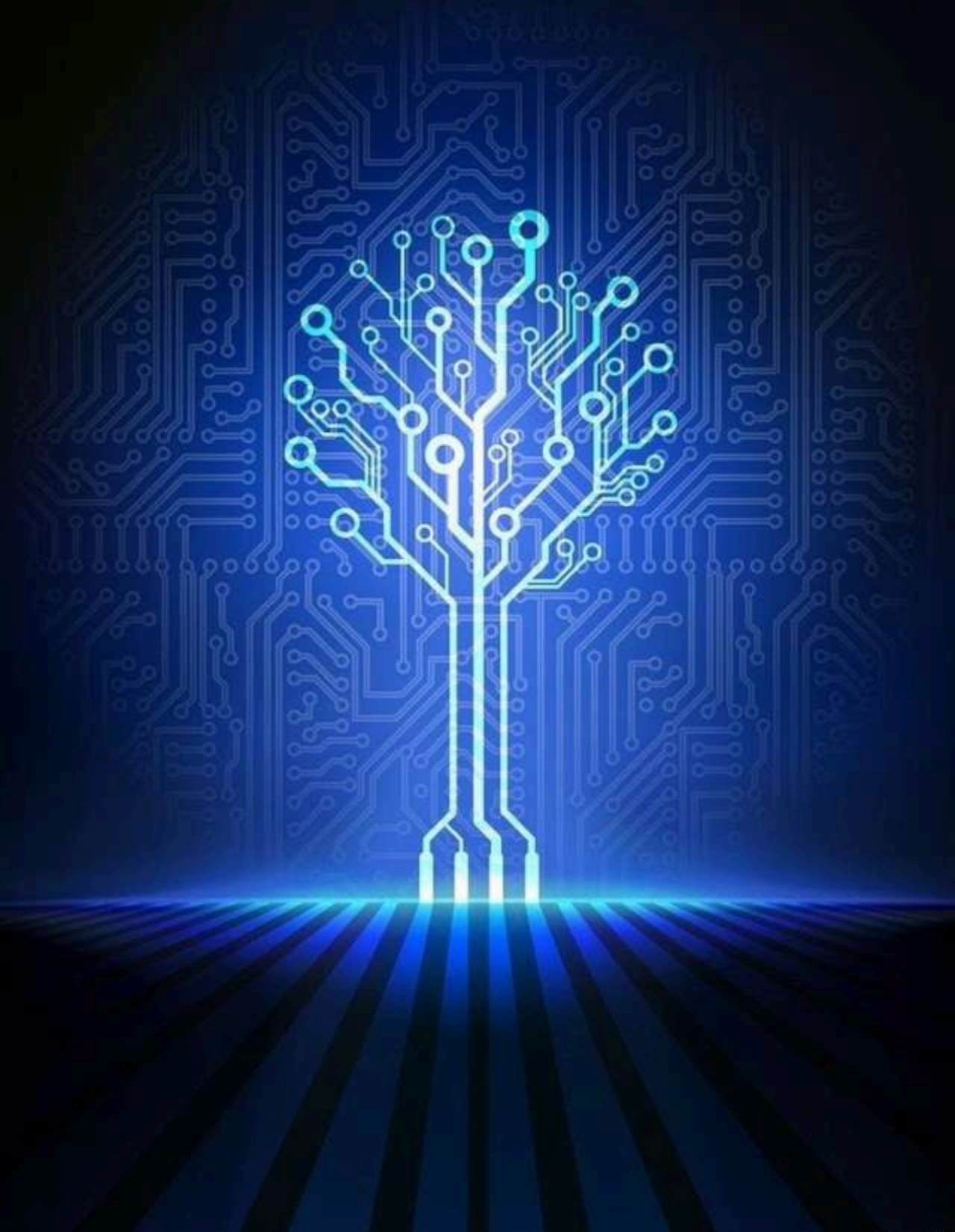
Repositorio en Github con los archivos de las simulaciones de este capítulo con Proteus v7.8



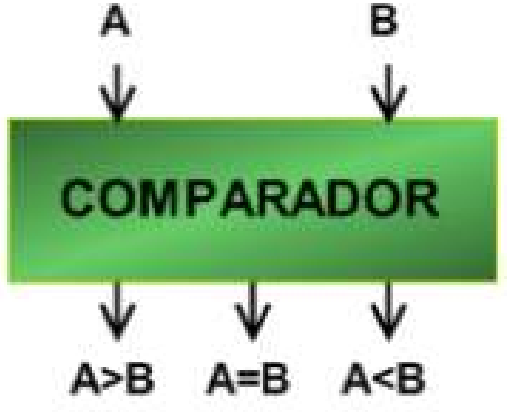


Capítulo III

Otros Dispositivos



El comparador digital es un circuito combinacional que consiste en comparar las magnitudes de dos cantidades binarias de *n* bits cada uno *(A y B)* y genera tres resultados en su salida: $A > B$, $A = B$ o $A < B$.



El comparador más básico es la compuerta lógica **XOR:**

- Cuando las entradas son iguales la salida es '0'

- Cuando las entradas son diferentes la salida es '1'

3.1 Comparador de magnitudes

[2][3][4][5]

★ Ejemplo 1:

Diseñar un circuito Comparador binario de 2 bits utilizando compuertas lógicas ***XOR*** *referencia comercial 7486 que active la salida cuando los bits* $A = B$ *y luego implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.*

Elaboración de la tabla de verdad:

ENTRADAS				SALIDA
B2	B1	A2	A1	A=B
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Se bajan los datos de la tabla de verdad al mapa de Karnaugh de 4 variables (A1, A2, B1 y B2):

B1B2 / A1A2	00	01	11	10
00	1			
01		1		
11			1	
10				1

A=B

- Las funciones Booleanas obtenidas del mapa de Karnaugh son:

$$\boldsymbol{Sal} = (\overline{\boldsymbol{A1}} \times \overline{\boldsymbol{A2}} \times \overline{\boldsymbol{B1}} \times \overline{\boldsymbol{B2}}) + (\overline{\boldsymbol{A1}} \times \boldsymbol{A2} \times \overline{\boldsymbol{B1}} \times \boldsymbol{B2})$$

$$+(\boldsymbol{A1} \times \boldsymbol{A2} \times \boldsymbol{B1} \times \boldsymbol{B2}) + (\boldsymbol{A1} \times \overline{\boldsymbol{A2}} \times \boldsymbol{B1} \times \overline{\boldsymbol{B2}})$$

- Aplicando Álgebra Booleana: se sacan términos comunes de los 2 primeros minterms:

$$(\overline{\boldsymbol{A1}}\,\overline{\boldsymbol{B1}}) \times [\overline{\boldsymbol{A2}}\,\overline{\boldsymbol{B2}} + \boldsymbol{A2}\,\boldsymbol{B2}]$$

- Aplicando Álgebra Booleana: se sacan términos comunes de los 2 últimos minterms:

$$(\boldsymbol{A1}\,\boldsymbol{B1}) \times [\boldsymbol{A2}\,\boldsymbol{B2} + \overline{\boldsymbol{A2}}\,\overline{\boldsymbol{B2}}]$$

- La función completa:

$$(\overline{\boldsymbol{A1}}\,\overline{\boldsymbol{B1}}) \times [\overline{\boldsymbol{A2}}\,\overline{\boldsymbol{B2}} + \boldsymbol{A2}\,\boldsymbol{B2}] + (\boldsymbol{A1}\,\boldsymbol{B1}) \times [\boldsymbol{A2}\,\boldsymbol{B2} + \overline{\boldsymbol{A2}}\,\overline{\boldsymbol{B2}}]$$

- Los paréntesis segundo y cuarto corresponden a la función lógica de la compuerta lógica **XNOR:**

$$\boldsymbol{Sal} = (\overline{\boldsymbol{A1}}\,\overline{\boldsymbol{B1}}) \times [\overline{\boldsymbol{A2} \oplus \boldsymbol{B2}}] + (\boldsymbol{A1}\,\boldsymbol{B1}) \times [\overline{\boldsymbol{A2} \oplus \boldsymbol{B2}}]$$

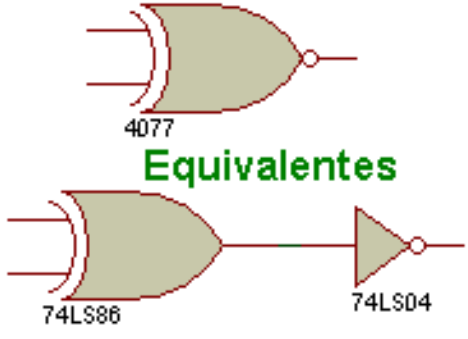
- De la función se saca de factor común el termino $[\overline{A2 \oplus B2}]$:

$$\boldsymbol{Sal} = [\overline{\boldsymbol{A2} \oplus \boldsymbol{B2}}]\{(\overline{\boldsymbol{A1}}\,\overline{\boldsymbol{B1}}) + (\boldsymbol{A1}\,\boldsymbol{B1})\}$$

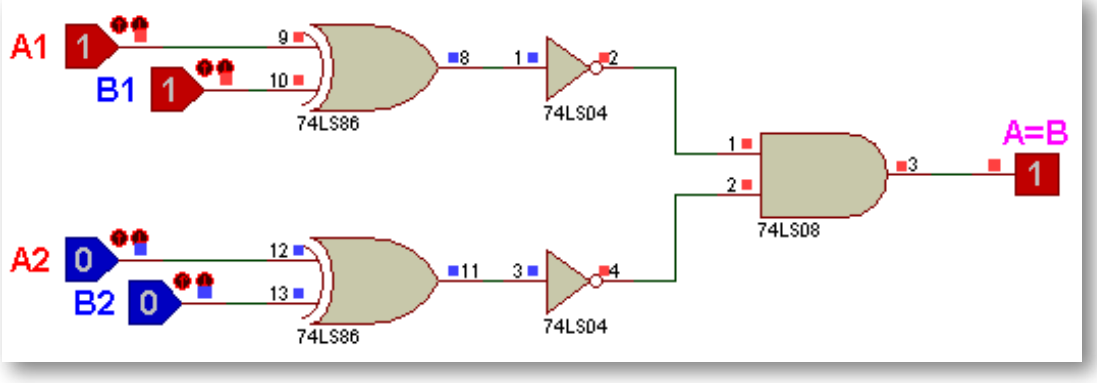
El término incluido en las llaves { } corresponde a la función lógica de una compuerta **XNOR**:

$$Sal = [\overline{A2 \oplus B2}] \times (\overline{A1 \oplus B1})$$

La función resultante contiene dos compuertas **XNOR** y el enunciado del ejemplo dice que se debe diseñar con compuertas **XOR**, para ello se implementa la equivalencia por medio de una compuerta **XOR** y en su salida se conecta una compuerta **NOT**, tal como se observa en la siguiente figura:



En la imagen del circuito simulado con Proteus se observa que el dato en la Entrada A es: $01_b = 1_d$ y en la Entrada B es: $01_b = 1_d$, activando la salida $A = B$:

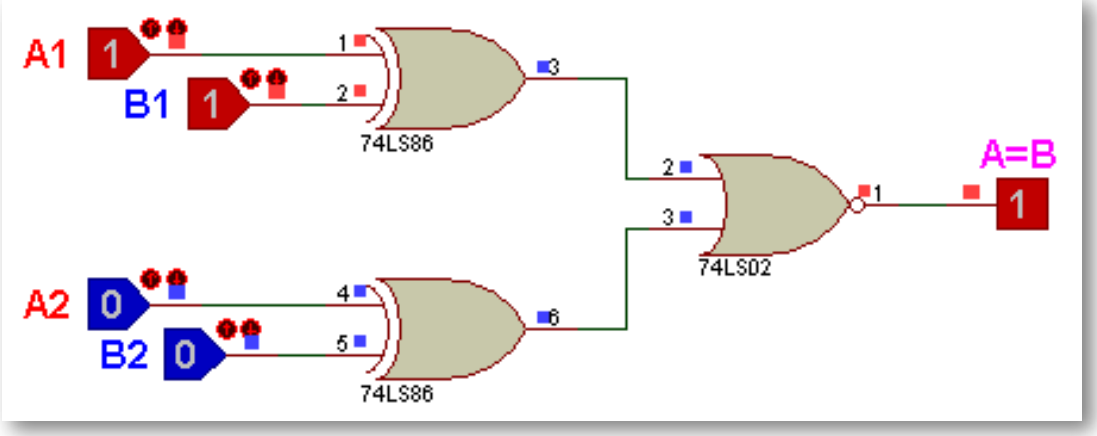


Otra forma de solucionar es aplicando la **ley de Morgan** a la función resultante:

$$\boldsymbol{Sal = (\overline{A2 \oplus B2}) \times (\overline{A1 \oplus B1})}$$

$$Sal = \overline{(A2 \oplus B2) + (A1 \oplus B1)}$$

En la imagen del circuito simulado con Proteus se observa que el dato en la Entrada A es: $01_b = 1_d$ y en la Entrada B es: $01_b = 1_d$, activando la salida $A = B$:



Repositorio en Github (Ejemplo_3-1-1.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

Diseñar un circuito Comparador binario de 2 bits utilizando compuertas lógicas que active tres salidas: $A < B$, $A = B$ y $A > B$. Implemente el circuito con un software de simulación electrónica y verifique su funcionamiento.

La tabla de verdad completa:

ENTRADAS				SALIDAS		
B2	B1	A2	A1	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
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	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Se descarga la información a los mapas de Karnaugh respectivos y se simplifican obteniendo las siguientes funciones Booleanas:

B1B2 / A1A2	00	01	11	10
00		1	1	1
01			1	
11				
10		1	1	

A<B

B1B2 / A1A2	00	01	11	10
00	1			
01		1		
11			1	
10				1

A=B

B1B2 / A1A2	00	01	11	10
00				
01	1			1
11	1	1		1
10	1			

A>B

Las funciones lógicas respectivas son:

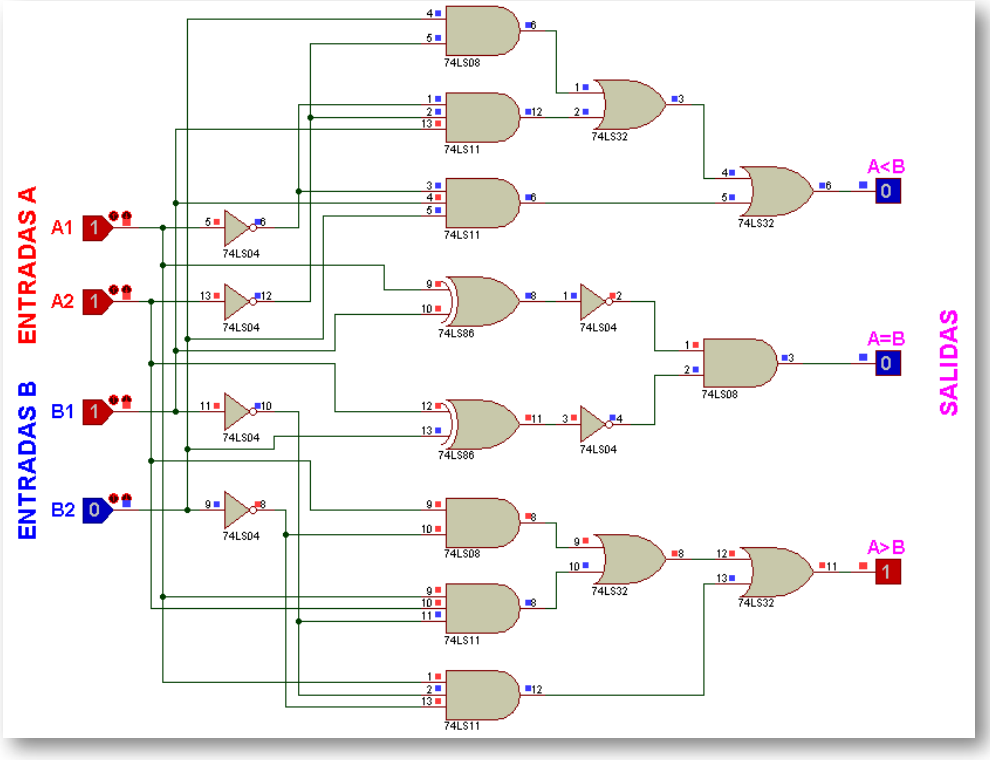
$$A < B = (\overline{A2} \times B2) + (\overline{A1} \times \overline{A2} \times B1) + (\overline{A1} \times B1 \times B2)$$

$$A > B = (A2 \times \overline{B2}) + (A1 \times A2 \times \overline{B1}) + (A1 \times \overline{B1} \times \overline{B2})$$

La función de $A = B$ se solucionó y se simplificó con álgebra Booleana en el ejemplo 1 de este capítulo y es:

$$(A = B) = [\overline{A2 \oplus B2}] \times (\overline{A1 \oplus B1})$$

En la imagen del circuito simulado con Proteus se observa que el dato en la Entrada A es: $11_b = 3_d$ y en la Entrada B es: $01_b = 1_d$, activando la salida $A > B$:



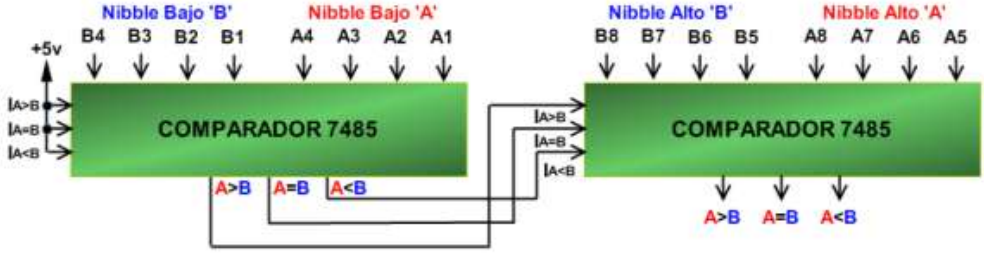
Repositorio en [Github (Ejemplo_3-1-2.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

*El circuito integrado **74LS85** es un comparador de magnitudes de cuatro bits cada uno y genera a la salida $A < B$, $A = B$ y $A > B$; además, posee tres líneas de entrada $I_{A<B}$, $I_{A=B}$ e $I_{A>B}$ que sirven para realizar expansiones utilizando dos o más circuitos integrados.*

*Implementar un comparador de 2 números binarios de 8 bits cada uno con circuitos integrados **74LS85**. Simule y verifique su funcionamiento*

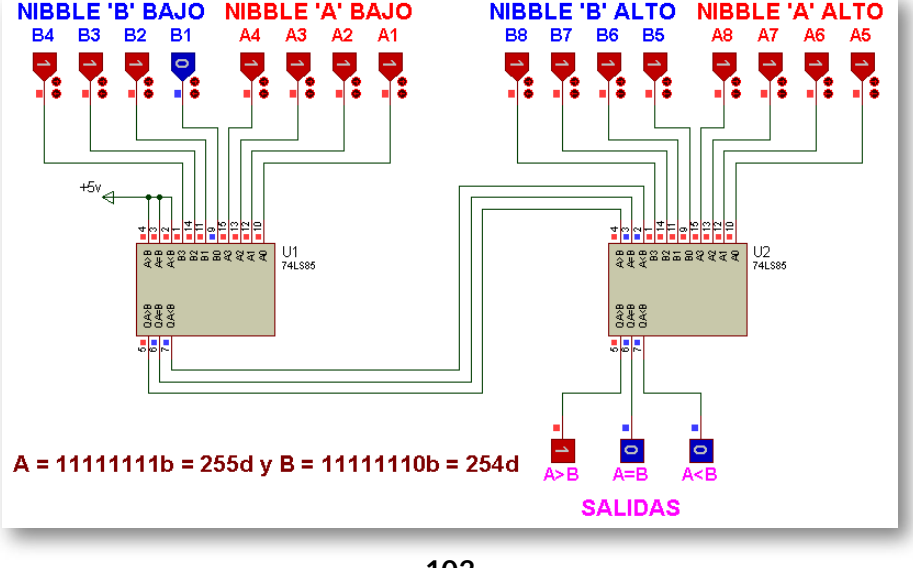
La implementación se realiza conectando 2 circuitos integrados referencia **74LS85** en cascada, cuyo diagrama en bloques es:

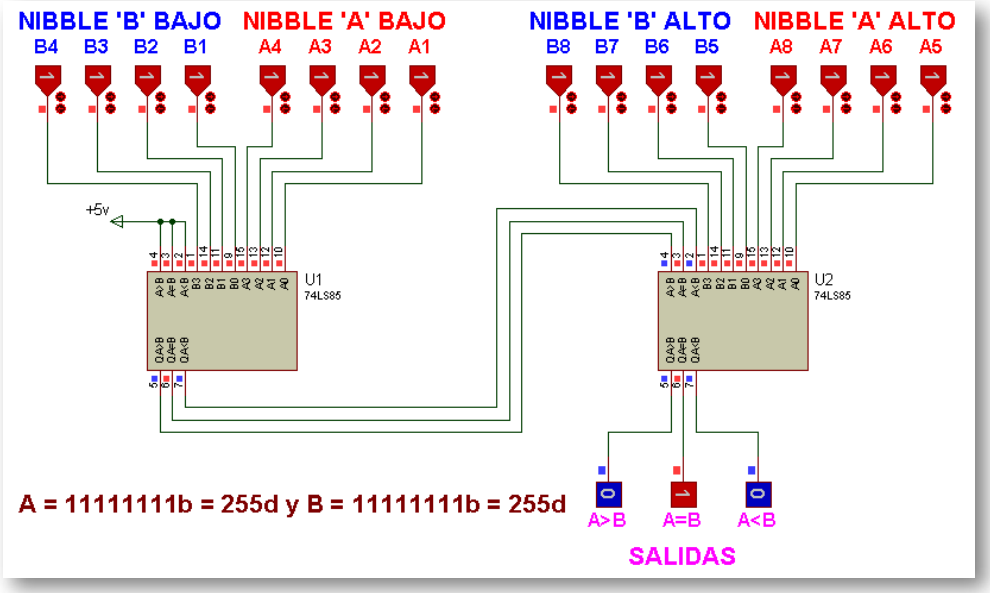


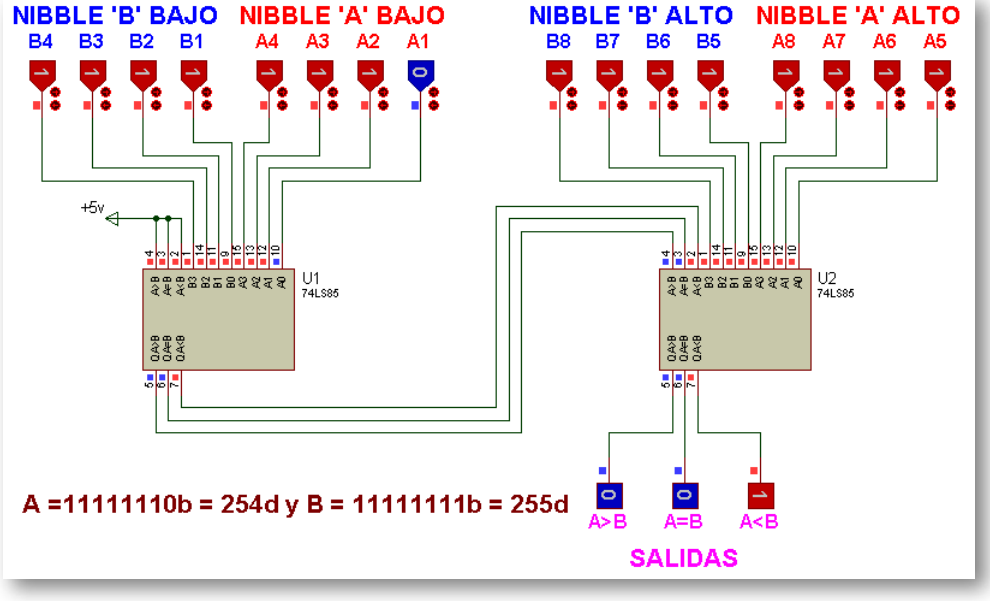
Caso 1 $(A > B)$**:** en la primera imagen se observa que el dato en la Entrada A es: $1111\ 1111_b = 255_d$ y en la Entrada B es: $1111\ 1110_b = 254_d$, activando la salida $A > B$:

Caso 2 $(A = B)$**:** en la segunda imagen se observa que el dato en la Entrada A es: $1111\ 1111_b = 255_d$ y en la Entrada B es: $1111\ 1111_b = 255_d$, activando la salida $A = B$:

Caso 3 $(A < B)$**:** en la tercera imagen se observa que el dato en la Entrada A es: $1111\ 1110_b = 254_d$ y en la Entrada B es: $1111\ 1111_b = 255_d$, activando la salida $A < B$:





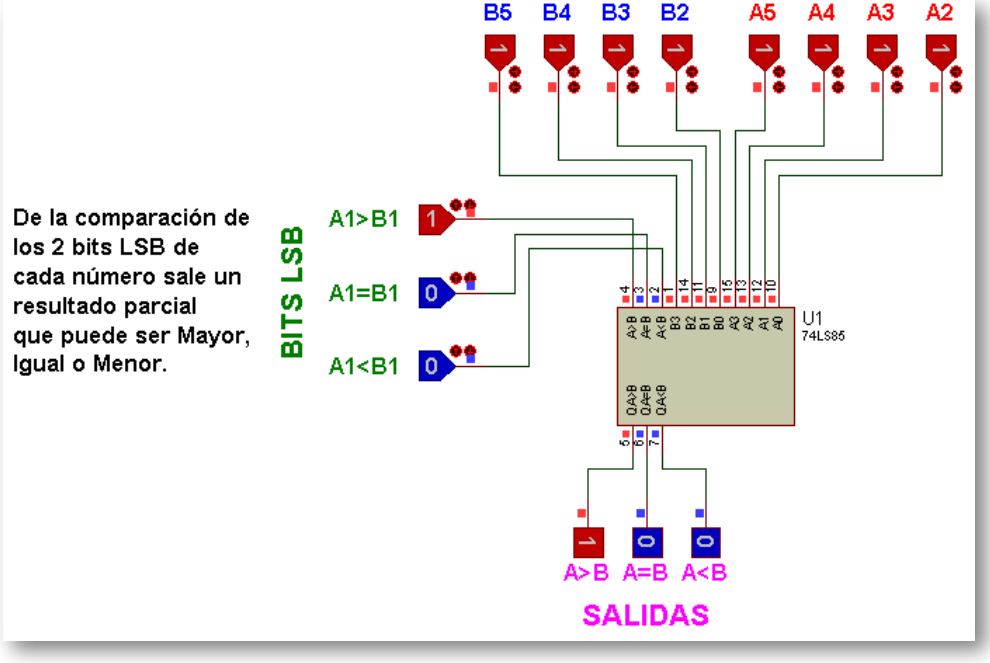


Repositorio en [Github (Ejemplo_3-1-3.dsn)] con el archivo de la simulación con Proteus v7.8

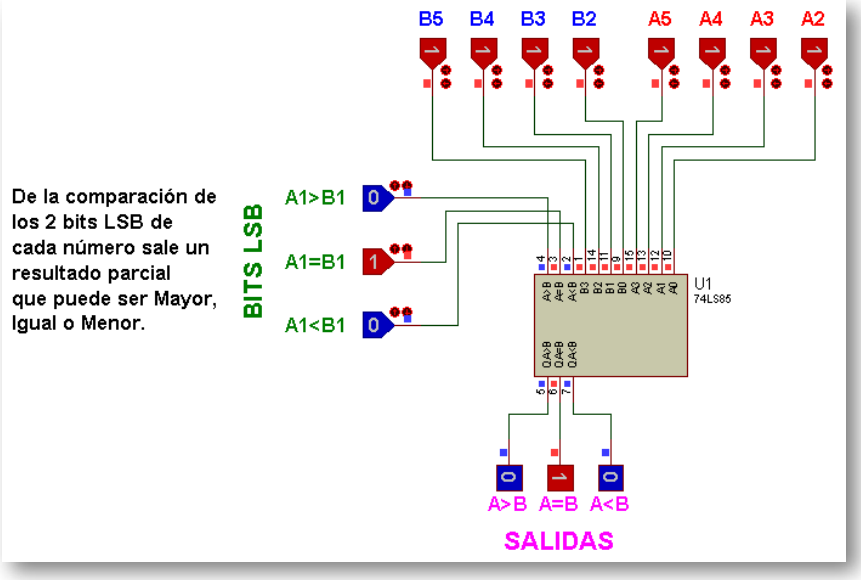
★ Ejemplo 4:

Diseñar un comparador de 5 bits utilizando un solo circuito integrado 74LS85, que muestre en sus salidas si $A < B$, $A = B$ y $A > B$. Implemente en un software de simulación y verifique su funcionamiento.

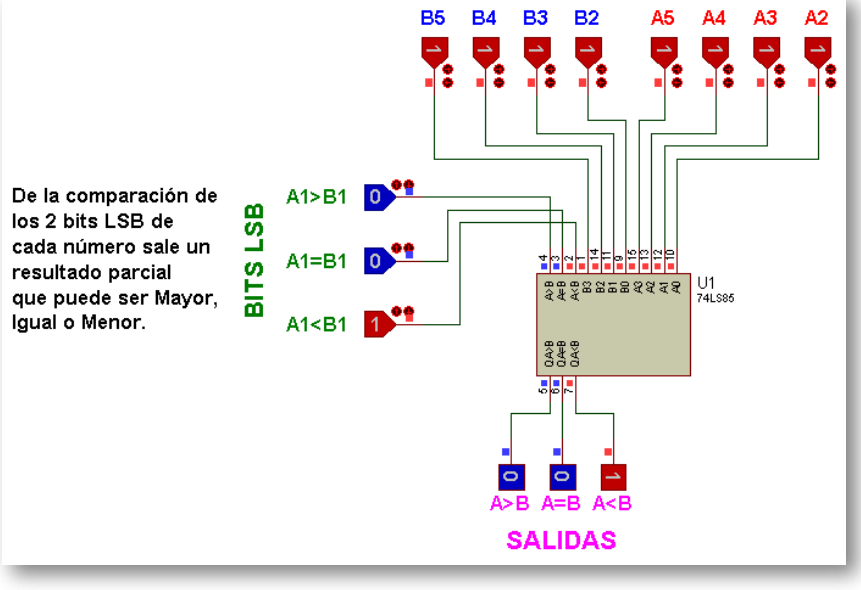
- Utilizando las entradas de expansión del circuito integrado, **se comparan los 2 Bits Menos Significativos (LSB) de cada número**, donde el resultado tendrá que ser Mayor, Igual o Menor.
- En caso de **ser diferentes los 2 bits** la salida nunca podrá ser la de $A = B$, podrá ser la de $A < B$ o la de $A > B$. Ver tabla de verdad del circuito integrado.
- **Caso 1** $(A1 > B1)$: en la imagen se observan los datos: $A = 1111\ 1$ y $B = 1111\ 0$, donde se activa la respuesta $A > B$:



Caso 2 $(A1 = B1)$: en la imagen se observan los datos: $A = 1111\ 1$ y $B = 1111\ 1$, donde se activa la respuesta $A = B$:



Caso 3 $(A1 < B1)$: en la imagen se observan los datos: $A = 1111\ 0$ y $B = 1111\ 1$, donde se activa la respuesta $A < B$:



Repositorio en Github (Ejemplo_3-1-4.dsn) con el archivo de la simulación con Proteus v7.8

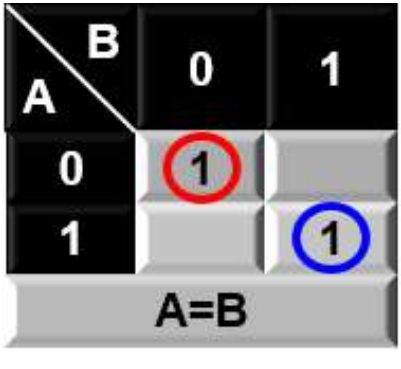
★ Ejemplo 5:

*Tomando como base el **ejemplo 4**, Diseñar los circuitos combinacionales a base de compuertas lógicas para comparar los 2 bits LSB de cada número. Implemente en un software de simulación y verifique su funcionamiento.*

Diseñar la tabla de verdad para el comparador de los 2 bits LSB:

BITS LSB		SALIDAS		
B1	A1	A1<B1	A1=B1	A1>B1
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

El mapa de Karnaugh para la salida $A = B$ es:



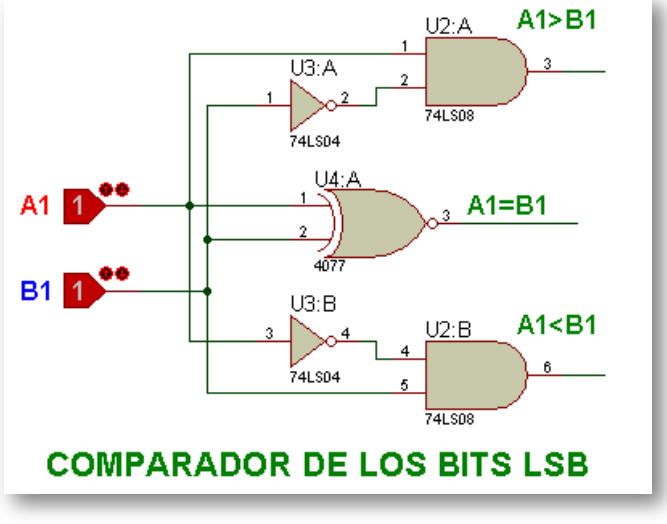
- Los mapas para las otras 2 salidas contienen 1 ISLA, lo que es equivalente al leer el Minterm en cada salida.
- Las funciones lógicas obtenidas son:

$$(A < B) = \overline{A1} \times B1$$

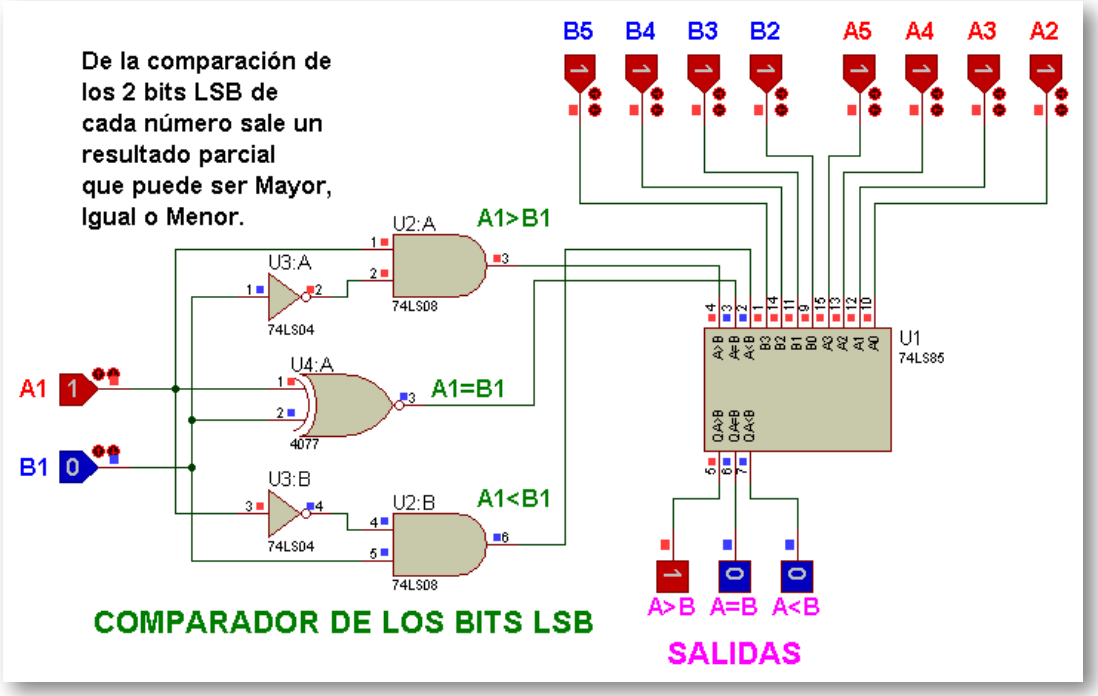
$$(A = B) = (\overline{A1} \times \overline{B1}) + (A1 \times B1) = \overline{(A1 \oplus B1)}$$

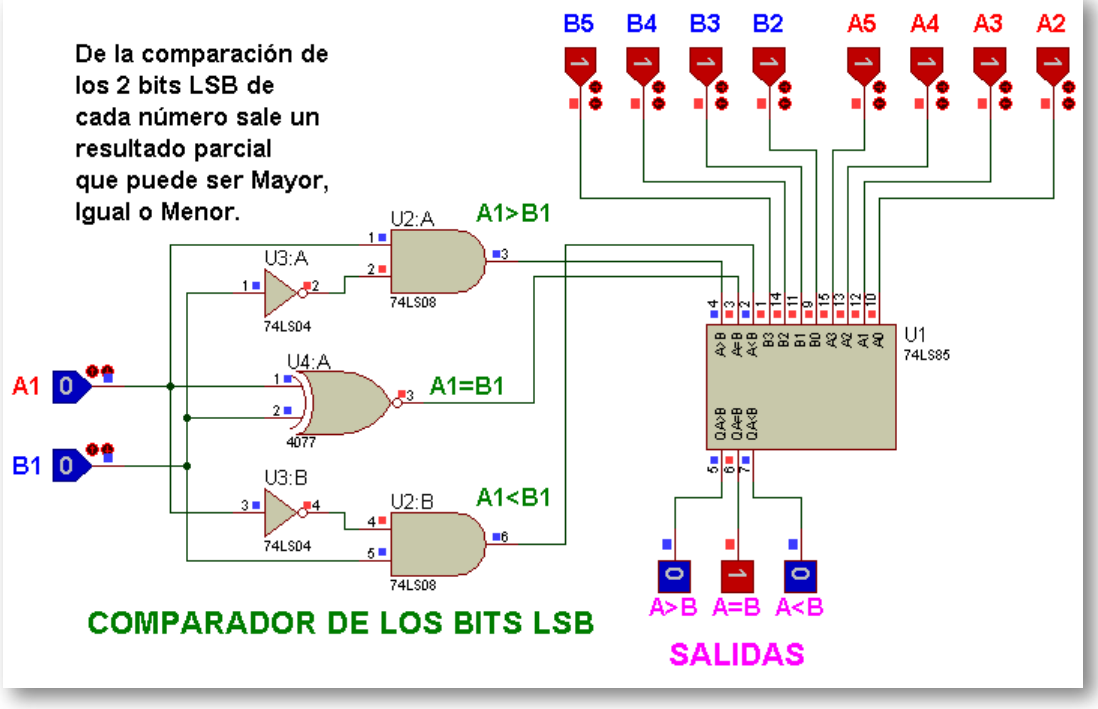
$$(A > B) = A1 \times \overline{B1}$$

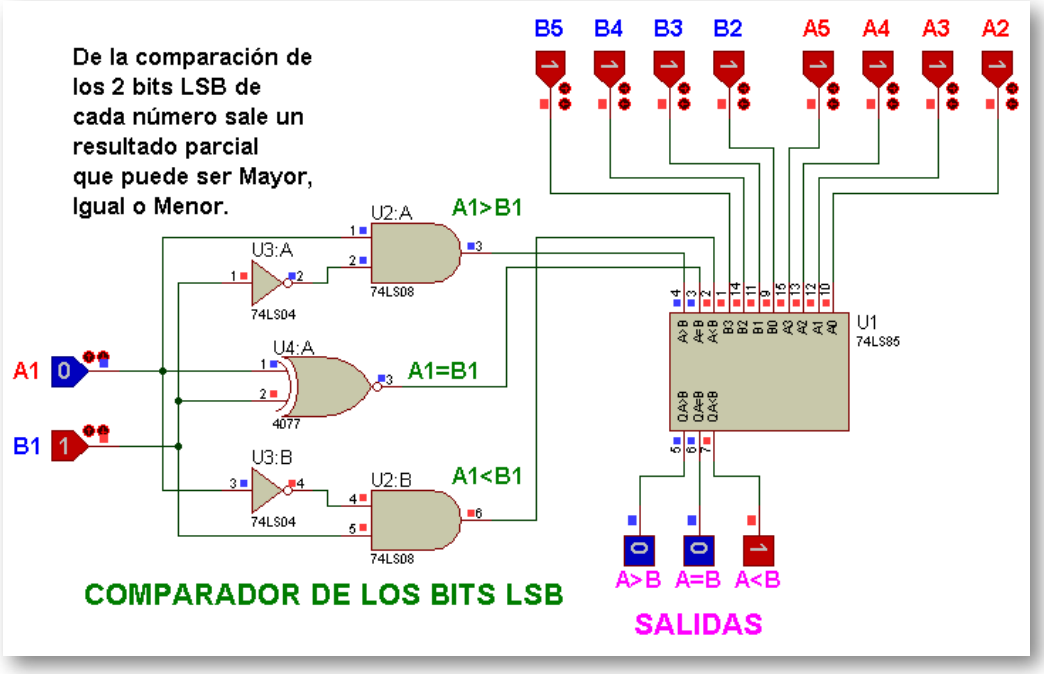
- Los circuitos del comparador de los 2 bits LSB son:



- **Caso 1** $(A1 > B1)$: en la imagen se observan los datos: $A = 1111\ 1$ y $B = 1111\ 0$, donde se activa la respuesta $A > B$.
- **Caso 2** $(A1 = B1)$: en la imagen se observan los datos: $A = 1111\ 1$ y $B = 1111\ 1$, donde se activa la respuesta $A = B$.
- **Caso 3** $(A1 < B1)$: en la imagen se observan los datos: $A = 1111\ 0$ y $B = 1111\ 1$, donde se activa la respuesta $A < B$.


De la comparación de los 2 bits LSB de cada número sale un resultado parcial que puede ser Mayor, Igual o Menor.
B5 B4 B3 B2 A5 A4 A3 A2
U2:A
A1>B1
U3:A
74LS04
74LS08
U4:A
A1=B1
4077
A1
B1
U3:B
U2:B
A1<B1
U1
74LS85
A>B A=B A<B
SALIDAS
COMPARADOR DE LOS BITS LSB


De la comparación de los 2 bits LSB de cada número sale un resultado parcial que puede ser Mayor, Igual o Menor.
B5 B4 B3 B2 A5 A4 A3 A2
U2:A
A1>B1
U3:A
74LS04
74LS08
U4:A
A1=B1
4077
A1
B1
U3:B
U2:B
A1<B1
U1
74LS85
A>B A=B A<B
SALIDAS
COMPARADOR DE LOS BITS LSB



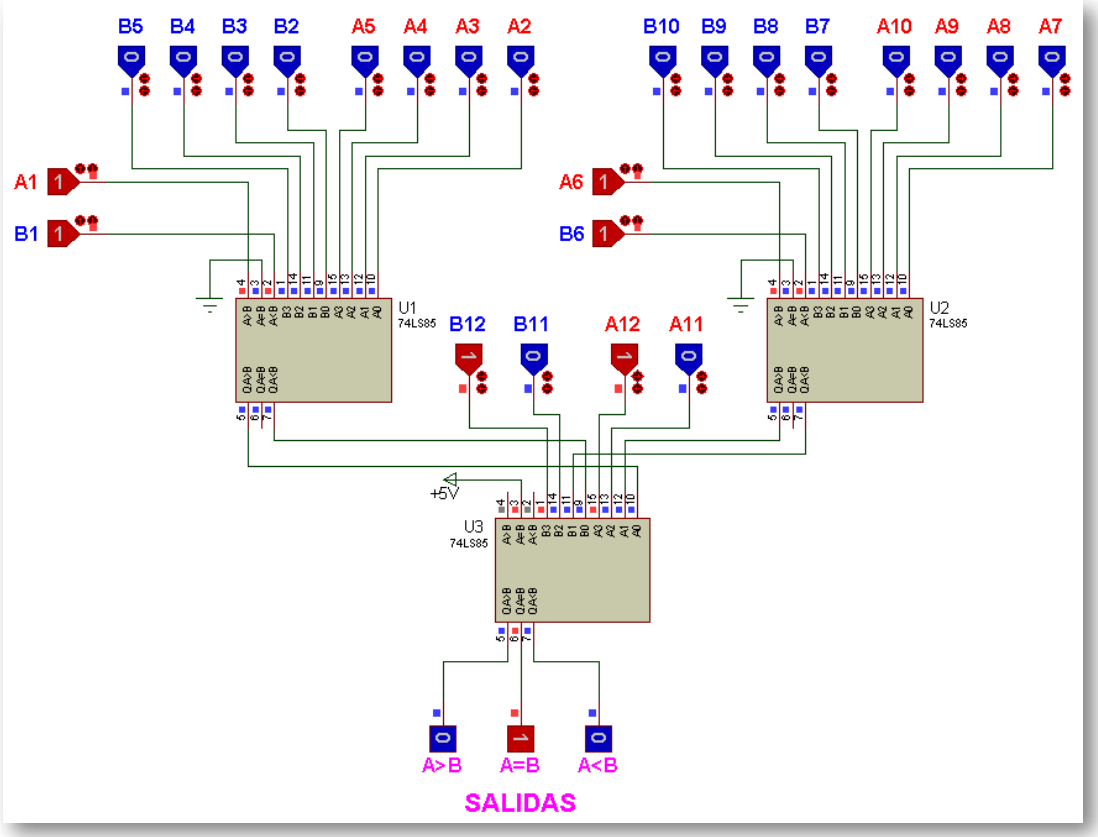
Repositorio en [Github (Ejemplo_3-1-5.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 6:

*Diseñar un comparador en **PARALELO** de 2 números binarios de 12 bits cada uno, con circuitos integrados [74LS85], el sistema contiene tres salidas que indican si el número es $A < B$, $A = B$ y $A > B$. Simular y verificar su funcionamiento.*

Para los 2 primeros circuitos integrados se utilizan las terminales de expansión $\boldsymbol{I_{A>B}}$ para $\boldsymbol{A1, A6}$; $\boldsymbol{I_{A<B}}$ para $\boldsymbol{B1, B6}$ y la entrada $\boldsymbol{I_{A=B}}$ se conecta a tierra (ver la tabla de verdad del integrado). Las salidas $\boldsymbol{A = B}$ de cada integrado

no se utilizan, esto quiere decir que el valor de los 10 primeros bits SOLO puede ser MAYOR o MENOR.



Para el tercer y último circuito integrado:

- Por la entrada del LSB de $\boldsymbol{A}$ (pin 10) se ingresa el dato de la salida $(\boldsymbol{A} > \boldsymbol{B})$ del primer circuito integrado y por la siguiente entrada de $\boldsymbol{A}$ (pin 12) se ingresa el dato de la salida $(\boldsymbol{A} > \boldsymbol{B})$ del segundo circuito integrado.
- Por la entrada del LSB de $\boldsymbol{B}$ (pin 9) se ingresa el dato de la salida $(\boldsymbol{A} < \boldsymbol{B})$ del primer circuito integrado y por la siguiente entrada de $\boldsymbol{B}$ (pin 11) se ingresa el dato de la salida $(\boldsymbol{A} < \boldsymbol{B})$ del segundo circuito integrado.

- Las demás entradas de A y B se utilizan como los bits de datos 11 y 12 respectivamente.
- En las entradas de expansión, hay que garantizar un nivel alto en la entrada $I_{A=B}$ según la tabla de verdad.

Repositorio en Github (Ejemplo_3-1-6.dsn) con el archivo de la simulación con Proteus v7.8

3.2 Generador de Paridad y Detector de Error

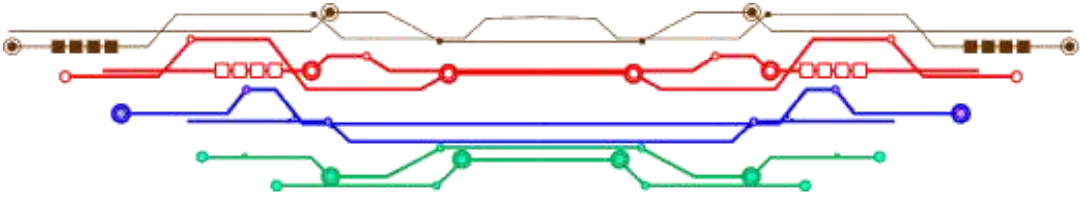
[3][6][7][8]

Al transmitir datos digitales de un lugar a otro, se pueden producir errores como el cambio de estado de uno o varios bits en la información, debido al ruido eléctrico o problemas de funcionamiento de los componentes del sistema; aunque la probabilidad es muy pequeña.

En un código de datos, un bit de paridad indica si el número de **'1s'** es PAR o IMPAR y de esta forma detectar si se presentan errores.

Descartando los Acarreos:

- La suma de un número **PAR** de **UNOS** siempre es **'0'.**
- La suma de un número **IMPAR** de **UNOS** siempre es **'1'.**



★ Ejemplo 1:

*Diseñar el circuito **Generador de Paridad** y el **Detector de Errores** para la siguiente tabla de verdad. Simular y verificar el funcionamiento:*

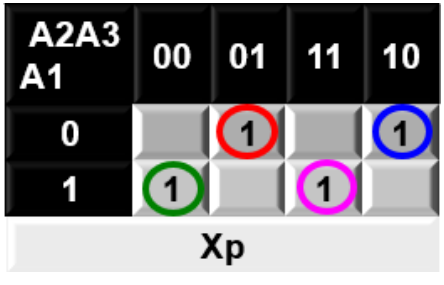
ENTRADAS		
A3	A2	A1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

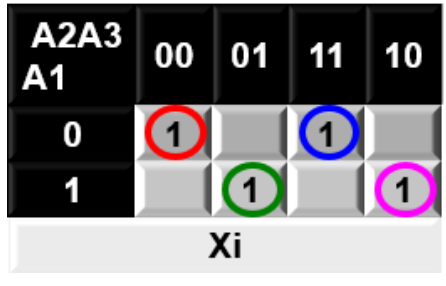
Se completa la tabla de verdad, generando en las salidas correspondientes (Xp, Xi) el valor del bit de PARIDAD:

- Para un sistema de **Paridad PAR** si la suma de '1s' da PAR, **NO** requiere agregar un '1' para completar la paridad PAR (ajuste). Si la suma de '1s' da IMPAR, requiere agregar un '1' para completar la paridad PAR (ajuste).
- Para un sistema de **Paridad IMPAR** si la suma de '1s' da IMPAR, **NO** requiere agregar un '1' para completar la paridad IMPAR (ajuste). Si la suma de '1s' da PAR, requiere agregar un '1' para completar la paridad IMPAR (ajuste).

ENTRADAS			PARIDAD	
A3	A2	A1	X_p	X_i
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

Mapas de Karnaugh:





Función Booleanas de la Paridad Par (Xp):

$$X_P = (\overline{A1} \times \overline{A2} \times A3) + (\overline{A1} \times A2 \times \overline{A3}) +$$

$$(A1 \times \overline{A2} \times \overline{A3}) + (A1 \times A2 \times A3)$$

Función Booleanas de la Paridad Impar (Xi):

$$X_I = (\overline{A1} \times \overline{A2} \times \overline{A3}) + (\overline{A1} \times A2 \times A3) +$$

$$(A1 \times \overline{A2} \times A3) + (A1 \times A2 \times \overline{A3})$$

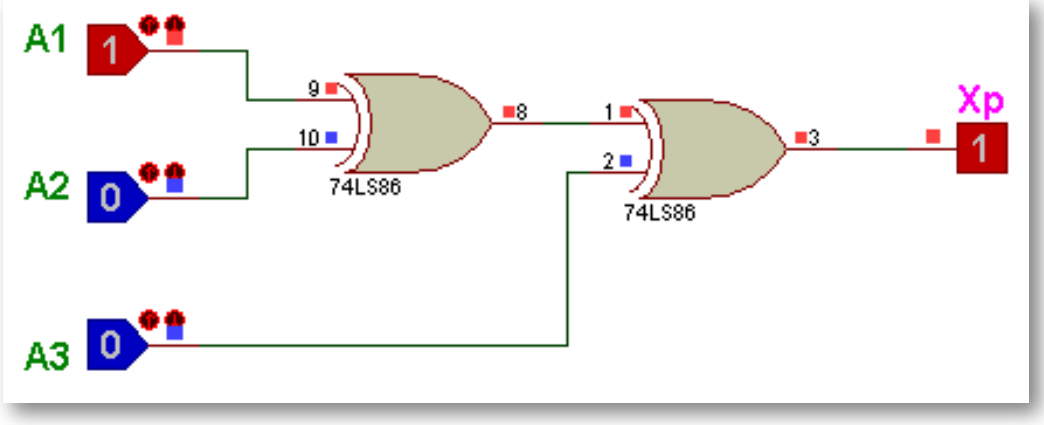
Para simplificar **Xp,** sacar factor común entre el primer y segundo término y entre el tercer y cuarto término:

$$X_P = \overline{A1}\left[(\overline{A2} \cdot A3) + (A2 \cdot \overline{A3})\right] + A1\left[(\overline{A2} \cdot \overline{A3}) + (A2 \cdot A3)\right]$$

$$X_P = \overline{A1} \cdot (A2 \oplus A3) + A1 \cdot \overline{(A2 \oplus A3)}$$

$$X_P = A1 \oplus A2 \oplus A3$$

Imagen del circuito **Xp**:



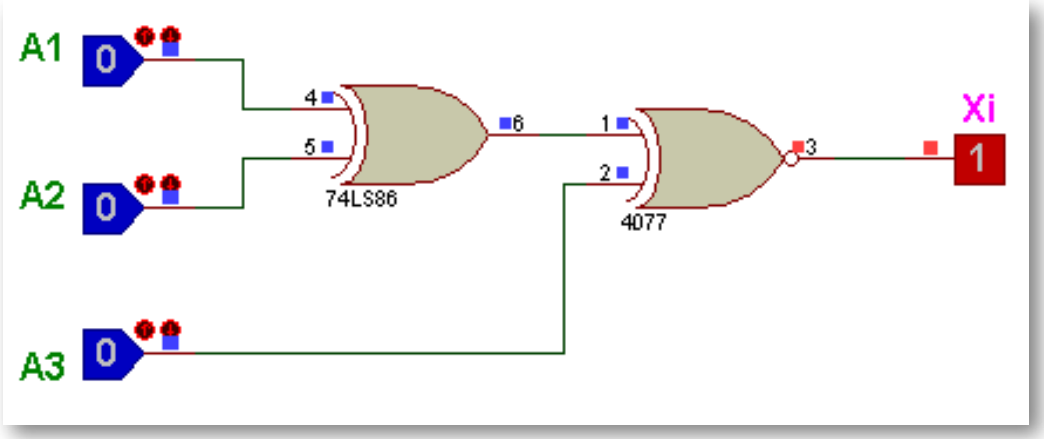
Para simplificar **Xi,** sacar factor común entre el primer y segundo término y entre el tercer y cuarto término:

$$X_I = \overline{A1}\left[(\overline{A2} \cdot \overline{A3}) + (A2 \cdot A3)\right] + A1\left[(\overline{A2} \cdot A3) + (A2 \cdot \overline{A3})\right]$$

$$X_I = \overline{A1} \cdot \overline{(A2 \oplus A3)} + A1 \cdot (A2 \oplus A3)$$

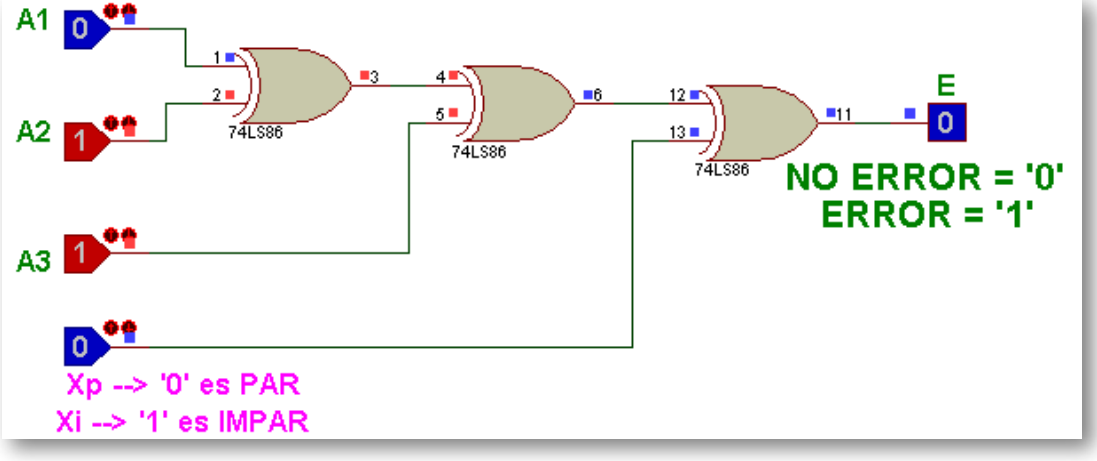
$$X_I = \overline{A1 \oplus A2 \oplus A3}$$

Imagen del circuito **Xi**:



Detector de Errores:

- Si la salida de error es = **'0'** es porque **NO HAY ERROR**
- Si la salida de error es = **'1'** es porque **HAY ERROR**



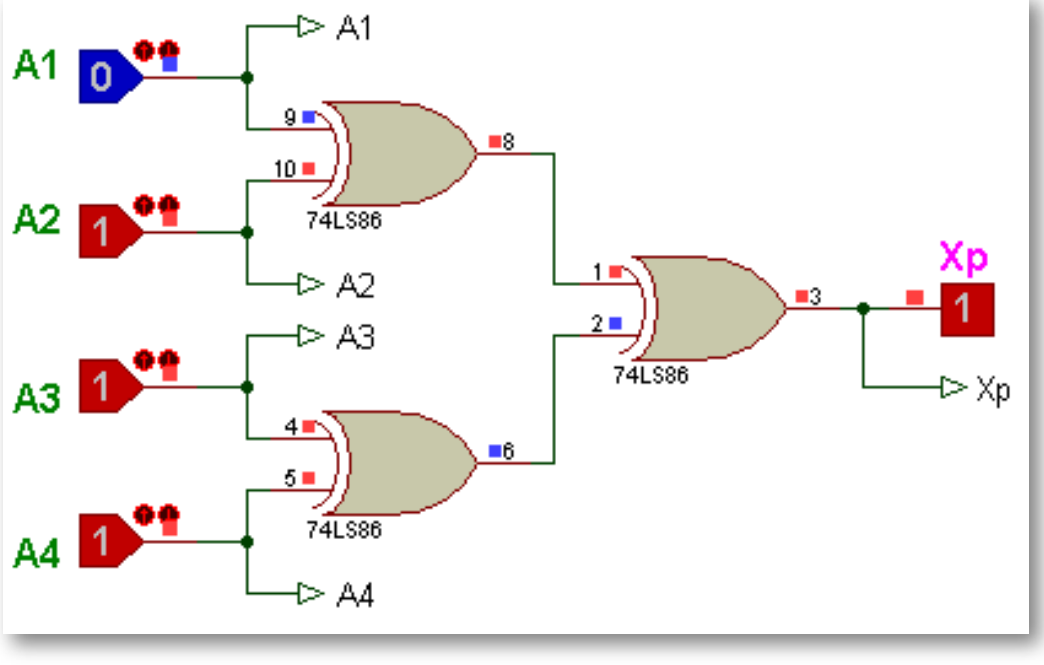
Repositorio en [Github (Ejemplo_3-2-1.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

*Diseñar un Generador de Paridad de 4 bits y hallar el **bit de paridad (X)** para los siguientes datos y verificar la solución en un simulador de circuitos electrónicos.*

No. Ej	A4	A3	A2	A1
1.	0	1	1	1
2.	1	0	0	1
3.	0	0	0	0
4.	0	1	0	0

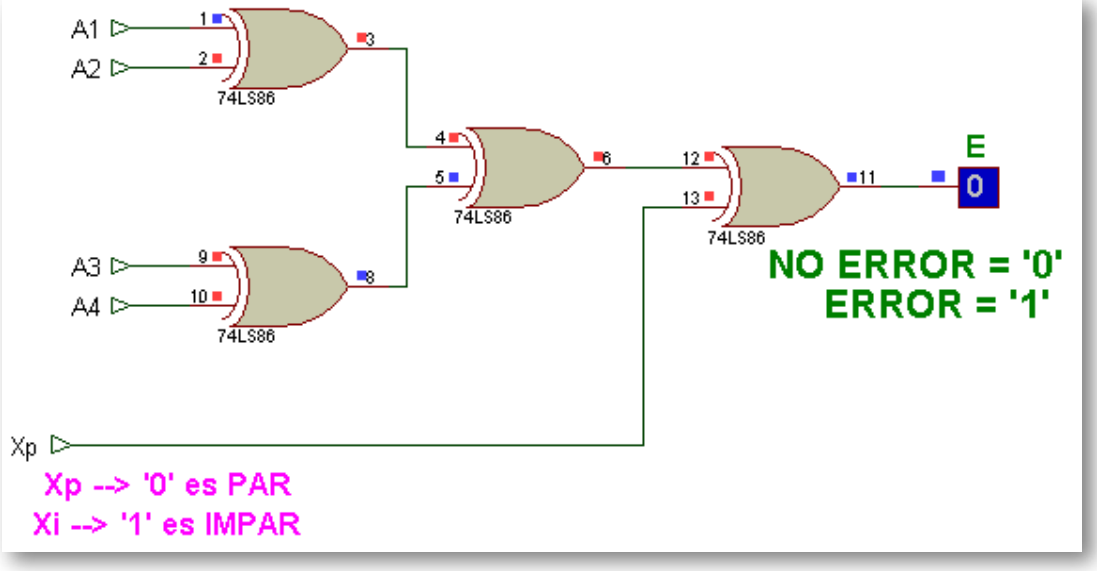
Circuito del Generador de Paridad de 4 bits:



Se aplican los niveles lógicos de los datos del enunciado a las entradas del generador de paridad (A1, A2, A3 y A4) y se halla el bit de paridad de la salida del circuito. Las respuestas para cada ejercicio son:

1. X = '1'
2. X = '0'
3. X = '0'
4. X = '1'

Circuito Detector de Error de 4 bits está conectado al circuito del Generador de Paridad para verificar las soluciones a los ejercicios:

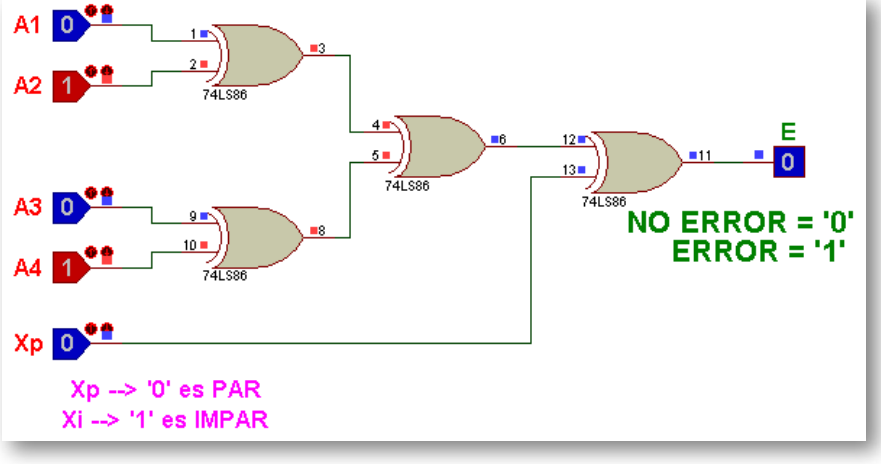


Repositorio en Github (Ejemplo_3-2-2.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

*Hallar la salida de un **Detector de Error** cuando se reciben los siguientes datos y verificar la solución con un programa de simulación de circuitos digitales electrónicos:*

No. Ej	X	A4	A3	A2	A1
1.	0	1	0	1	0
2.	1	1	1	1	0
3.	1	1	1	1	1
4.	1	0	0	0	0



Resultados:

1. E = '0', No hay Error
2. E = '0', No hay Error
3. E = '1', Error
4. E = '1', Error

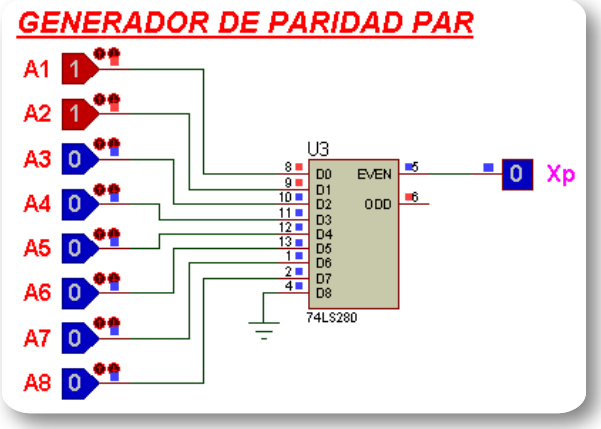
Repositorio en Github (Ejemplo_3-2-3.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 4:

*El circuito integrado 74LS280 sirve como **Generador** o como **Detector de Paridad**, consta de 9 entradas desde A hasta I y 2 salidas: ΣPar y ΣImpar.*

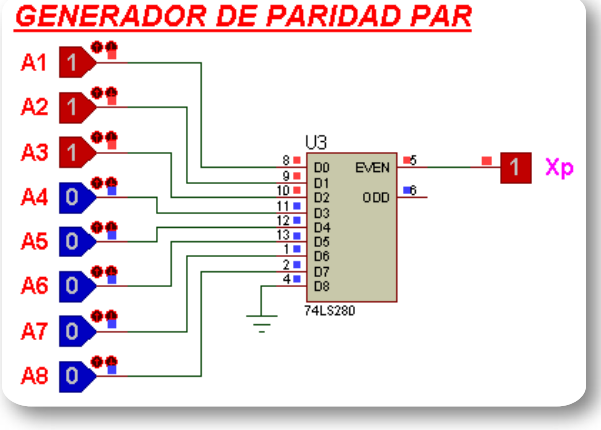
Generador de Paridad PAR:

- Si la suma de '1s' en sus entradas es **PAR** genera en su salida **ΣPar='0'** ya que **NO** requiere agregar un '1' para ajustar la PARIDAD PAR.



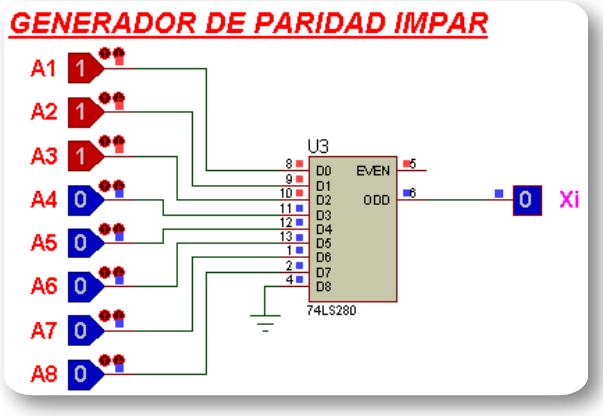
Generador de Paridad PAR:

- Si la suma de '1s' en sus entradas es **IMPAR** genera en su salida **ΣPar='1'** ya que **SI** requiere agregar un '1' para ajustar la PARIDAD PAR.



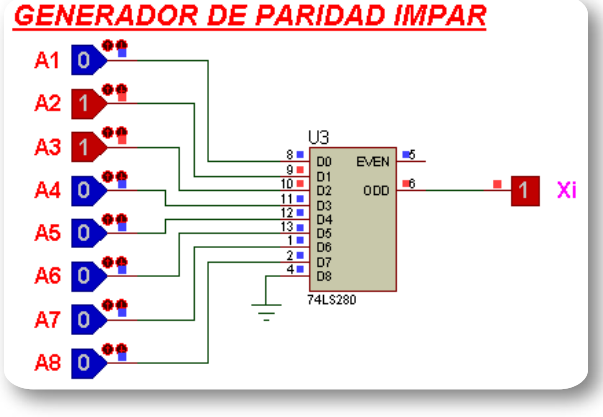
Generador de Paridad IMPAR:

- Si la suma de '1s' en sus entradas es **IMPAR** genera en la salida **ΣImpar='0'** ya que **NO** requiere agregar un '1' para ajustar la PARIDAD IMPAR.



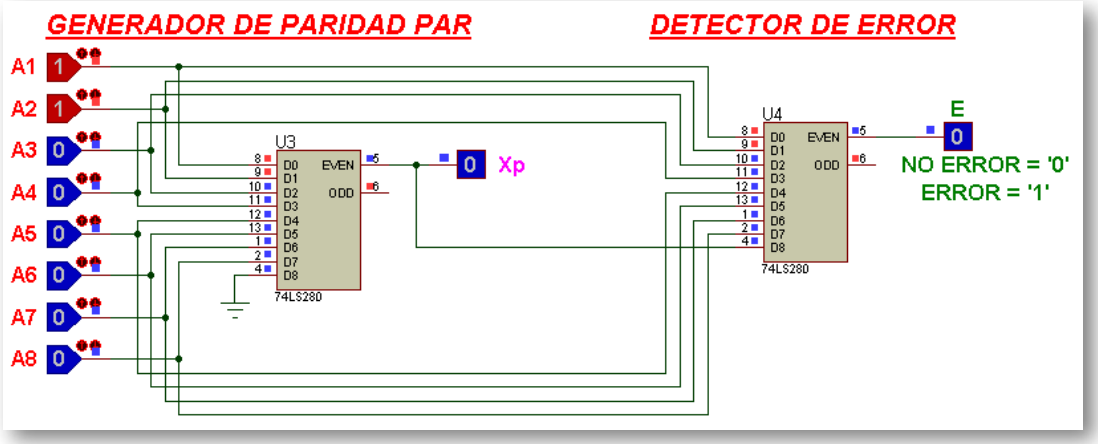
Generador de Paridad IMPAR:

- Si la suma de '1s' en sus entradas es **PAR** genera en la salida **ΣImpar='1'** ya que **SI** requiere agregar un '1' para ajustar la PARIDAD IMPAR.



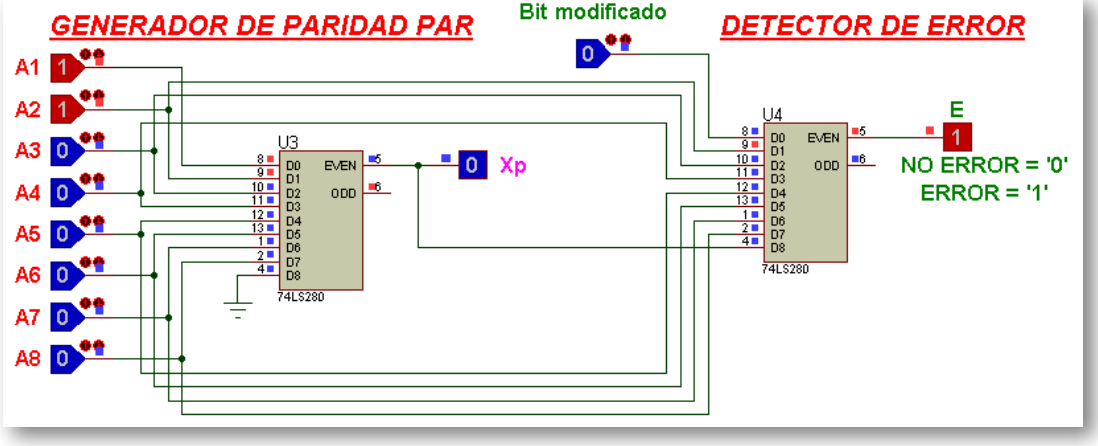
Detector de Paridad PAR:

- En la siguiente imagen se observa que la suma de '1s' en sus entradas es **PAR** y el Detector muestra en su salida **ΣPar='0'** ya que **NO** presenta **ERROR.**



Detector de Paridad PAR con ERROR:

- En la siguiente imagen se observa que la suma de '1s' en sus entradas es **PAR** pero se simula el cambio del nivel del bit **A1='0'** el Detector muestra en su salida **ΣPar='1'** ya que **SI** presenta **ERROR.**



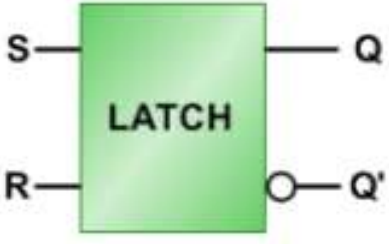
La desventaja de este método radica en que solo detecta si se presenta un número impar de errores en la información, si hubiese 2 errores simultáneamente, el Receptor **NO DETECTARÁ EL ERROR**.

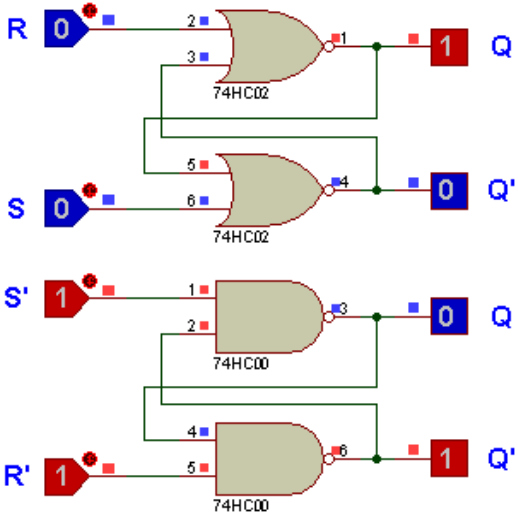
Repositorio en Github (Ejemplo_3-2-4.dsn) con el archivo de la simulación con Proteus v7.8

3.3 Latch (Biestables)

[3][4][10]

- Los **Latch** son circuitos combinacionales biestables asíncronos (sin reloj), cuya salida cambia en función del estado anterior y el estado actual de sus entradas
- Estos circuitos almacenan 1 bit de datos.





En el estado de **NO CAMBIA** conserva el estado anterior **(MEMORIZA)** y el estado **PROHIBIDO** no se utiliza ya que ambas salidas están en el mismo estado lógico.

RS compuertas NOR

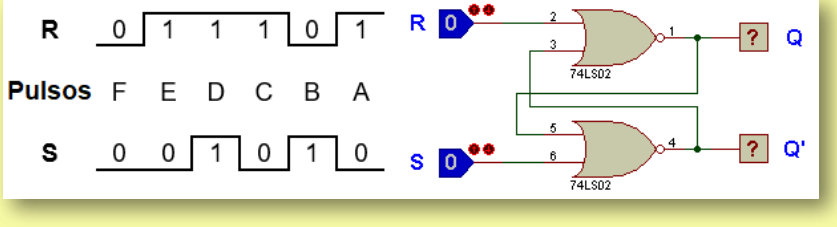
S	R	Q	$\overline{Q}$	ESTADO
0	0	Qo	$\overline{Qo}$	NO cambia, Memoriza
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	0	Prohibido

RS compuertas NAND

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	ESTADO
0	0	1	1	Prohibido
0	1	1	0	Set
1	0	0	1	Reset
1	1	Qo	$\overline{Qo}$	NO cambia, Memoriza

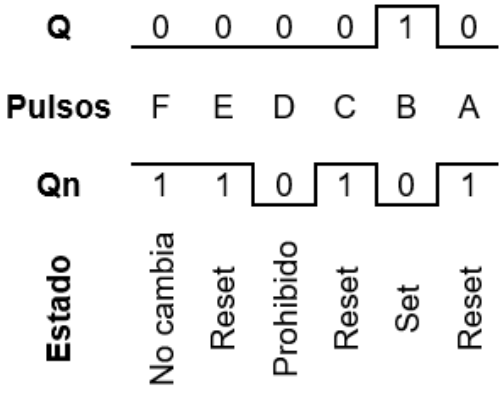
★ Ejemplo 1:

*Hallar las señales de salida y sus respectivos estados para cada pulso de entrada para el siguiente **Latch**:*



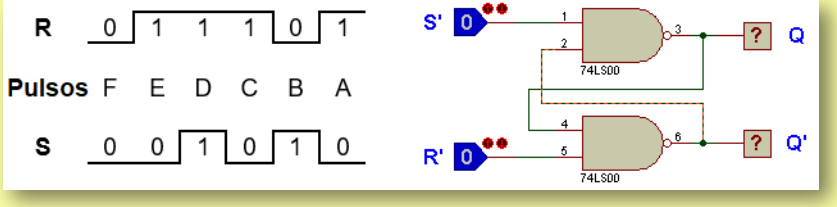
Verificar los resultados con un simulador electrónico.

Pulsos de salidas y sus respectivos estados para cada uno de los pulsos de entrada:



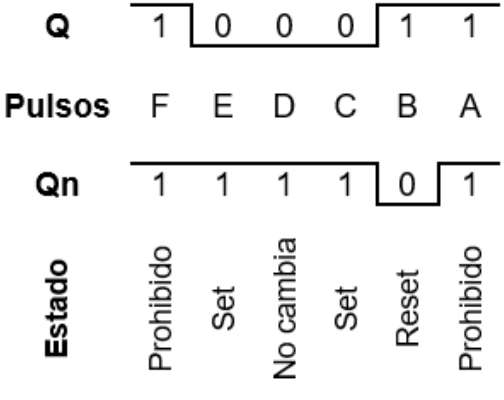
★ Ejemplo 2:

Hallar las señales de salida y sus respectivos estados para cada pulso de entrada para el siguiente ***Latch****:*



Verificar los resultados con un simulador electrónico.

Pulsos de salidas y sus respectivos estados para cada uno de los pulsos de entrada:



Repositorio en Github (Ejemplo_3-3.dsn) con el archivo de la simulación con Proteus v7.8

3.4 Tri-State

[2][4][9]

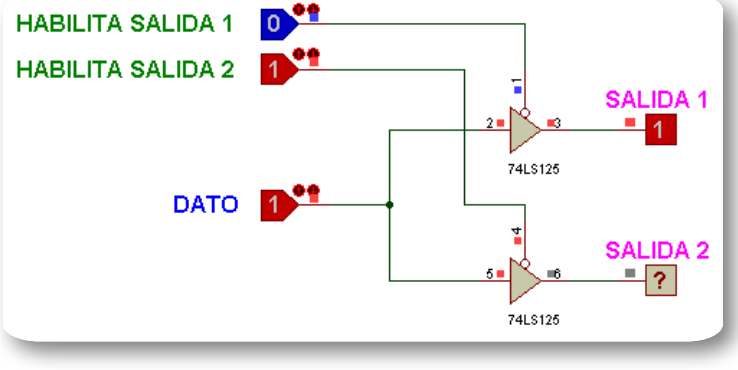
Un Buffer de Tres Estados (Tri-State) es un dispositivo que tiene tres estados de salida: bajo, alto y abierto o de alta impedancia (High-Z). Este estado desconecta la salida, para evitar conflictos con otros dispositivos que están compartiendo la línea de datos.

Comercialmente existen dispositivos activos con nivel BAJO, referencia comercial 74LS125 o activos con nivel ALTO referencia comercial 74LS126.

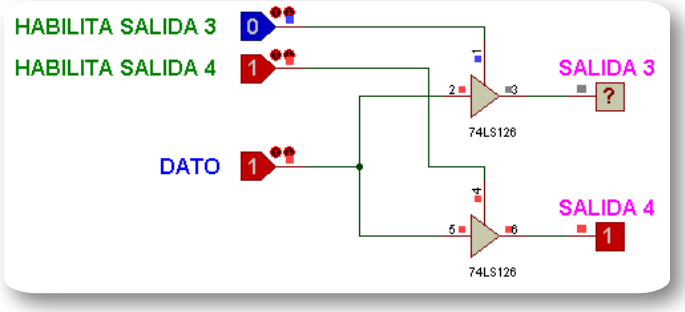
★ **Ejemplo 1:**

Demostración por medio de un programa de simulación de dispositivos electrónicos digitales el funcionamiento de los buffer tres estados referencias comerciales 74LS125 y el 74LS126.

Buffer activos con nivel **BAJO**, activa la Salida 1 y la Salida 2 queda abierta o alta impedancia (High-Z).



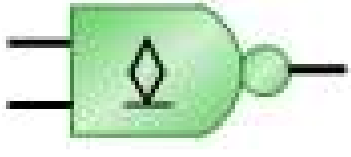
Buffer activos con nivel **ALTO,** la Salida 3 queda abierta o alta impedancia (High-Z) y se activa la Salida 4.



Repositorio en Github (Ejemplo_3-4.dsn) con el archivo de la simulación con Proteus v7.8

3.5 Colector Abierto (Open Collector)

[4][10]



Las compuertas lógicas de **Colector Abierto** funcionan igual a las compuertas tradicionales, la diferencia se presenta en que la salida está conectada a un transistor tipo **NPN** donde la terminal del **Colector** se deja abierto para manejo del usuario.

Para habilitar el uso de la compuerta se requiere cerrar el circuito, y esto se hace conectando una **Resistencia** entre el colector y el positivo de la fuente de alimentación; a esta se le llama **Resistencia de Pull-Up**.

Se puede utilizar con un voltaje diferente al de la compuerta lógica y por ende manejar más corriente y más voltaje para activar diferentes dispositivos como motores, relevos o diferentes tipos de controles. También se puede utilizar para la implementación de buses de datos.

Referencias comerciales de algunas compuertas lógicas con salida de **Colector Abierto** son:

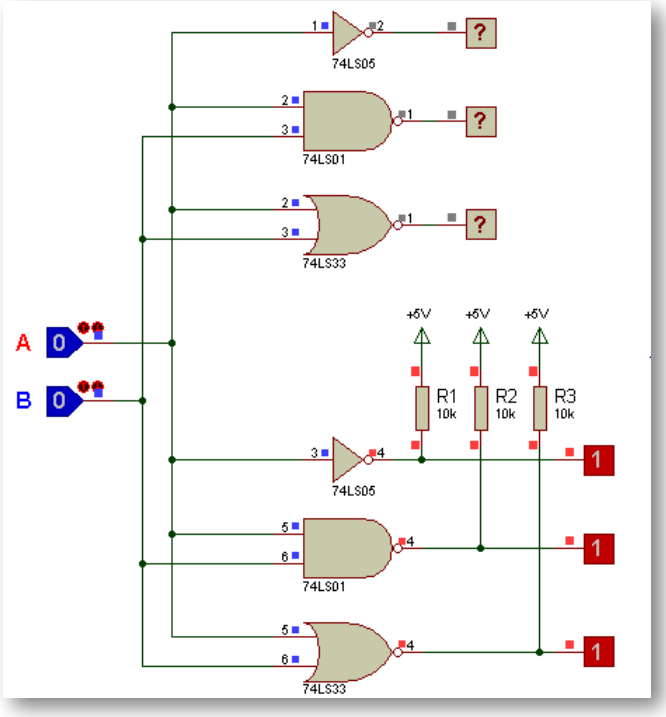
Referencias Comerciales de compuertas con Colector Abierto

TIPO	REFERENCIA	DESCRIPCIÓN
NAND	74LS01, 74LS03	2 entradas
	74LS12	3 entradas
	74LS22	4 entradas
AND	74LS09	2 entradas
	74LS15	3 entradas
NOT	74LS05, 74LS06	1 entrada
NOR	74LS33	2 entradas
XOR	74LS136	2 entradas
XNOR	74LS266	2 entradas

★ Ejemplo 1:

Demostración por medio de un programa de simulación de dispositivos electrónicos digitales el funcionamiento de algunas compuertas de ***Colector Abierto:***

En la parte superior de la imagen se observa que cuando las compuertas lógicas no tienen la **Resistencia Pull-Up** tienen sus respectivas salidas en **Alta Impedancia** y **NO** funciona correctamente. En la parte inferior de la imagen ya tiene conectadas las resistencias y las compuertas cumplen con las funciones lógicas respectivas:



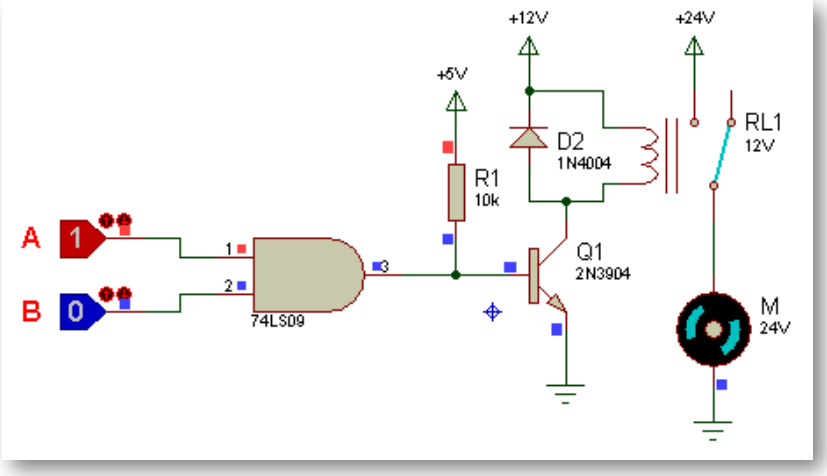
Repositorio en Github (Ejemplo_3-5-1.dsn) con el archivo de la simulación con Proteus v7.8

★ Ejemplo 2:

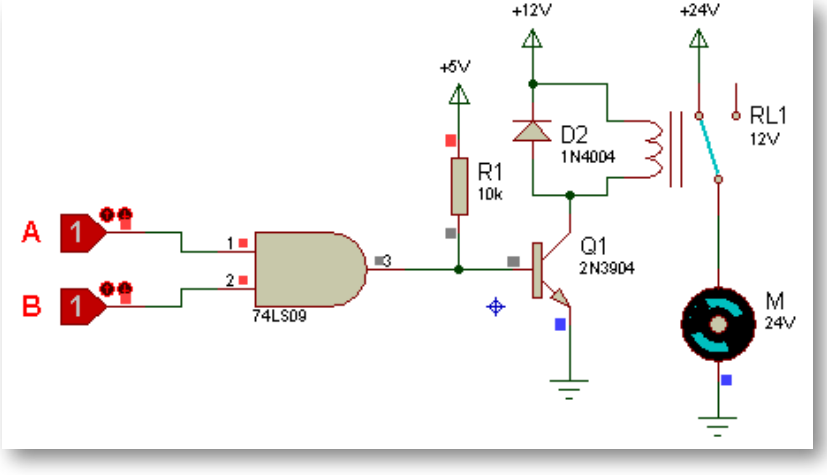
*Se requiere encender un motor de 24v utilizando una compuerta lógica **AND de 2 entradas** y un relevo de 12v. Verificar el funcionamiento con un software de simulación electrónica:*

- Se utiliza una compuerta AND con **Colector Abierto** que funciona con una alimentación de 5v
- Se coloca una **Resistencia de Pul-Up** de la alimentación de 5v a la salida de la compuerta lógica y a su vez a la base de un transistor NPN referencia 2N3904.

- El colector del transistor controla el relevo de 12v y su contacto NA alimenta el motor de 24v.
- Estados de las entradas de la compuerta A='1' y B='0' y la salida está en nivel **Bajo** y el motor está **desactivado**:



- Estados de las entradas de la compuerta A='1' y B='1' y la salida está en nivel **Alto** y el motor está **activado**:

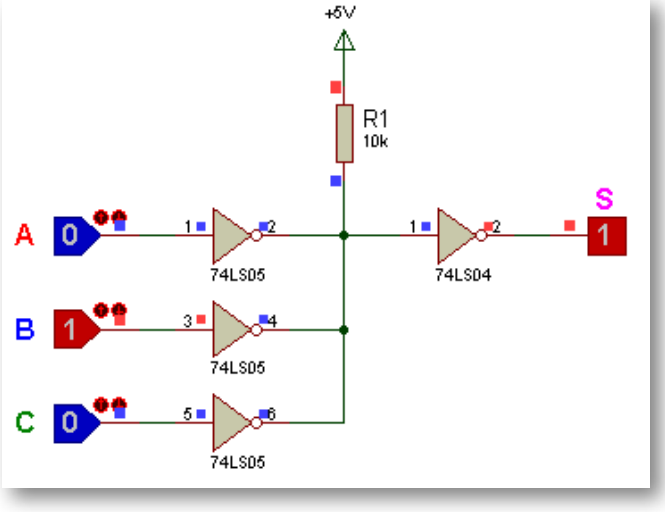


Repositorio en [Github (Ejemplo_3-5-2.dsn)] con el archivo de la simulación con Proteus v7.8

★ Ejemplo 3:

*Implementar un bus de datos de 3 líneas utilizando compuertas **NOT de Colector Abierto**. Verificar el funcionamiento con un software de simulación electrónica:*

Solo una compuerta lógica **NOT de Colector Abierto** se activa a la vez, en la imagen se observa que es la **compuerta de la línea B**.



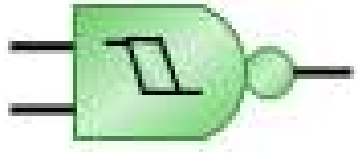
Repositorio en Github (Ejemplo_3-5-3.dsn) con el archivo de la simulación con Proteus v7.8

3.6 Schmitt Trigger

[2]

Este tipo de compuertas lógicas se utilizan para convertir señales que tienen ruido o están distorsionadas, en señales digitales bien estructuradas.

- Las compuertas Schmitt-trigger tienen las mismas funciones de las compuertas tradicionales, pero tienen una propiedad especial llamada **Histéresis** que las inmuniza al ruido entregando a la salida una **Señal Cuadrada** bien definida.



- La **Histéresis** es una brecha diferencial entre los cambios de niveles bajo a alto y de alto a bajo, superior a las compuertas tradicionales, las compuertas Schmitt Trigger solo responden cuando los voltajes aplicados superan los valores de umbral.
- Algunas referencias comerciales:

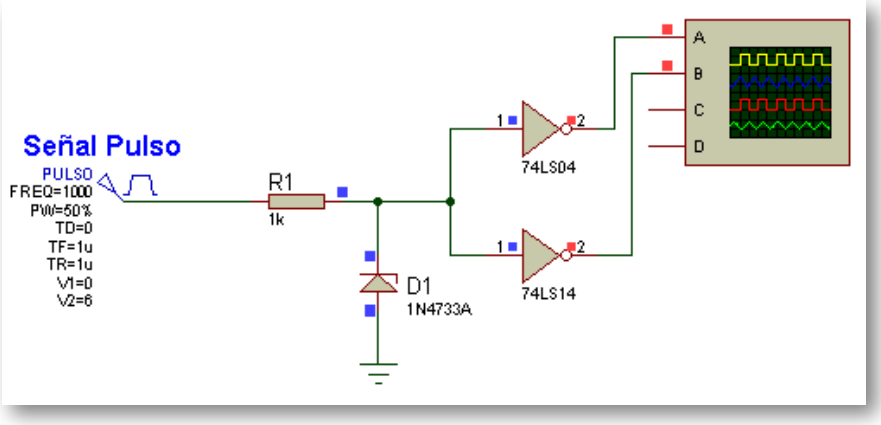
Referencias Comerciales de compuertas Schmitt-Trigger

TIPO	REFERENCIA	DESCRIPCIÓN
NAND	74132, 4093	2 entradas
NOT	74C14	1 entrada
YES (Buffer)	4010	1 entrada
AND	74HCS08	2 entradas
	74HCS21	4 entradas
NOR	74HC7002	2 entradas

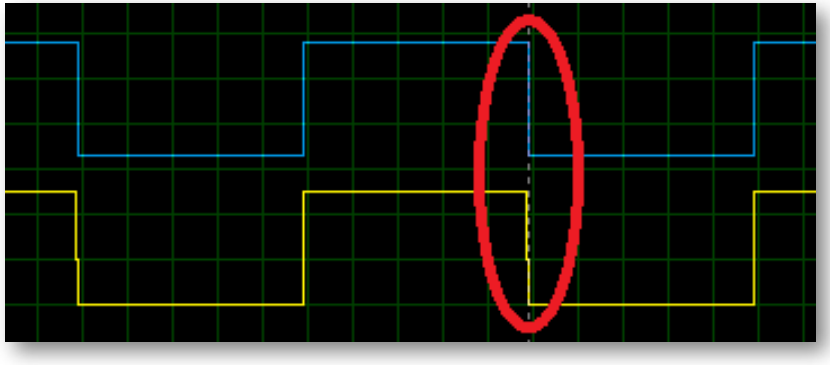
★ Ejemplo 1:

*Un Generador de Señales suministra simultáneamente una señal **PULSO** con una F=1KHz y 6v a 2 compuertas lógicas **NOT**. La tradicional referencia 74HC04 y la Schmitt Trigger 74C14. Utilizando un Osciloscopio comparar le comportamiento de las señales de salida. Utilizar un software de simulación electrónica.*

- Se coloca un Diodo Zener de 5.1v referencia 1N4733A para limitar el voltaje de entrada para las compuertas lógicas.
- Imagen de la simulación:



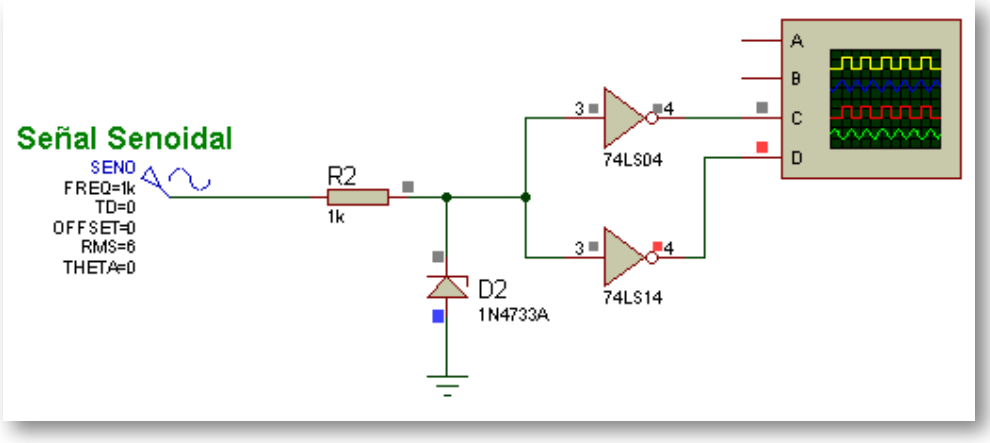
- Imagen del Osciloscopio con las respuestas de las señales de salida (señal azul del Schmitt Trigger bien definida):



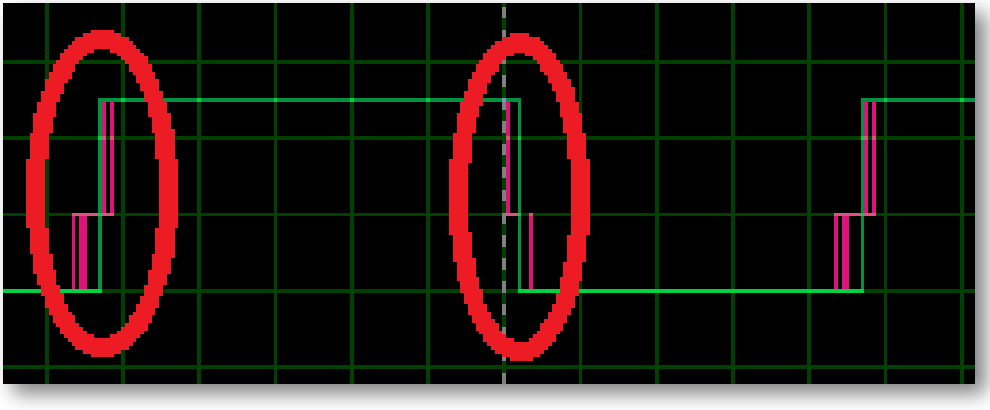
★ Ejemplo 2:

*Un Generador de Señales suministra simultáneamente una señal **SENO** con una F=1KHz y 6v a 2 compuertas lógicas **NOT**. La tradicional referencia 74HC04 y la Schmitt Trigger 74C14. Utilizando un Osciloscopio comparar el comportamiento de las señales de salida. Utilizar un software de simulación electrónica.*

- Se coloca un Diodo Zener de 5.1v referencia [1N4733A] para limitar el voltaje de entrada para las compuertas lógicas.
- Imagen de la simulación:



- Imagen del Osciloscopio con las respuestas de las señales de salida (señal verde del Schmitt Trigger bien definida):



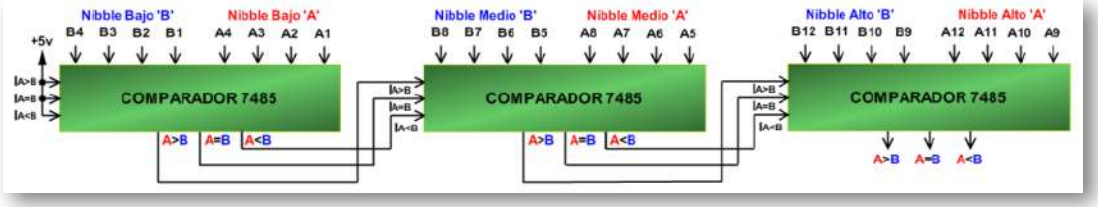
Repositorio en [Github (Ejemplo_3-6-1.dsn),(Ejemplo_3-6-2.dsn)] con los archivos de las simulaciones con Proteus v7.8

3.7 Ejercicios propuestos con respuestas

★ Ejercicio 1:

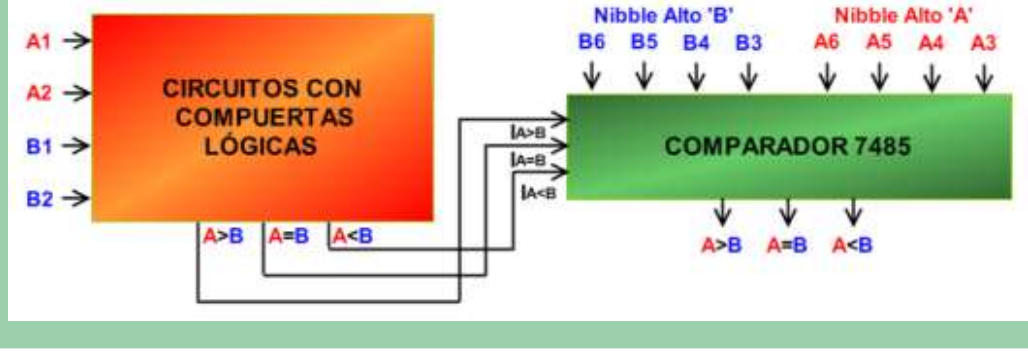
Implementar un comparador de 2 números binarios de 12 bits cada uno con los circuitos integrados 74LS85, el sistema contiene tres salidas que indican si el número es $A < B$, $A = B$ y $A > B$. Simular y verificar su funcionamiento.

Diagrama en bloques de la solución:



★ Ejercicio 2:

Implementar un comparador de 2 números binarios de 6 bits cada uno, utilizando circuitos combinacionales con compuertas lógicas y con un solo circuito integrado 74LS85 como se observa en el diagrama de la imagen, el sistema contiene tres salidas que indican si el número es $A < B$, $A = B$ y $A > B$. Simular y verificar su funcionamiento.

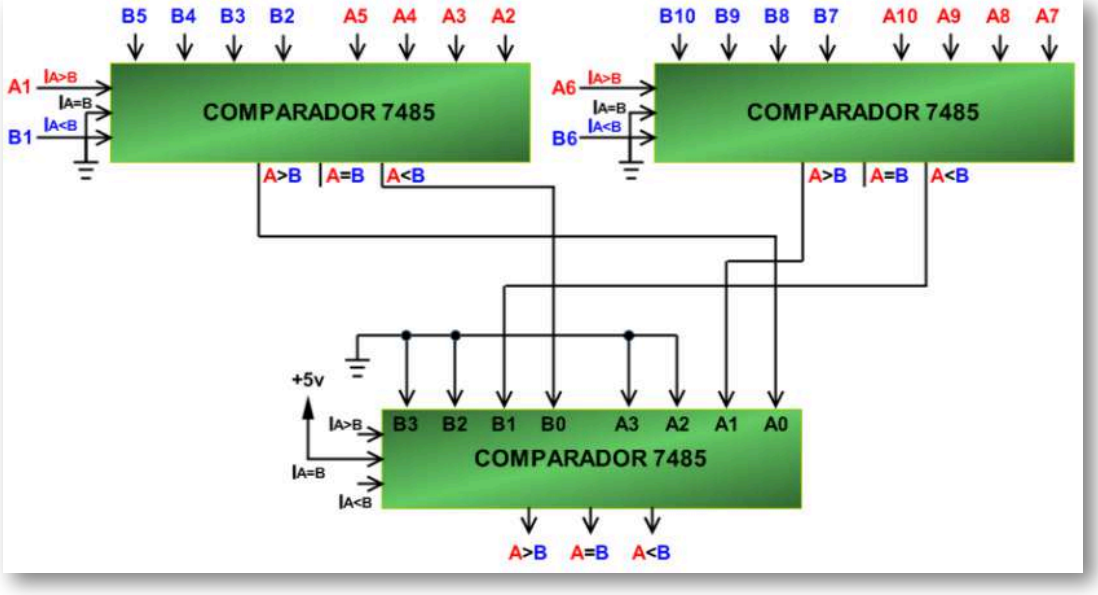


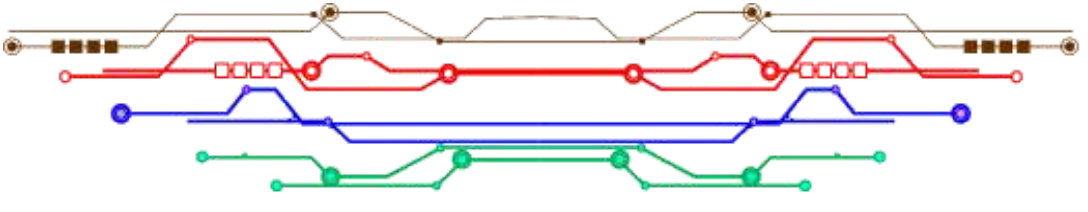
Los ejemplos 2 y 5 de este capítulo sirven como referencia para la solución del ejercicio.

★ Ejercicio 3:

*Diseñar un comparador en **PARALELO** de 2 números binarios de 10 bits cada uno, con circuitos integrados 74LS85, el sistema contiene tres salidas que indican si el número es $A < B$, $A = B$ y $A > B$. Simular y verificar su funcionamiento.*

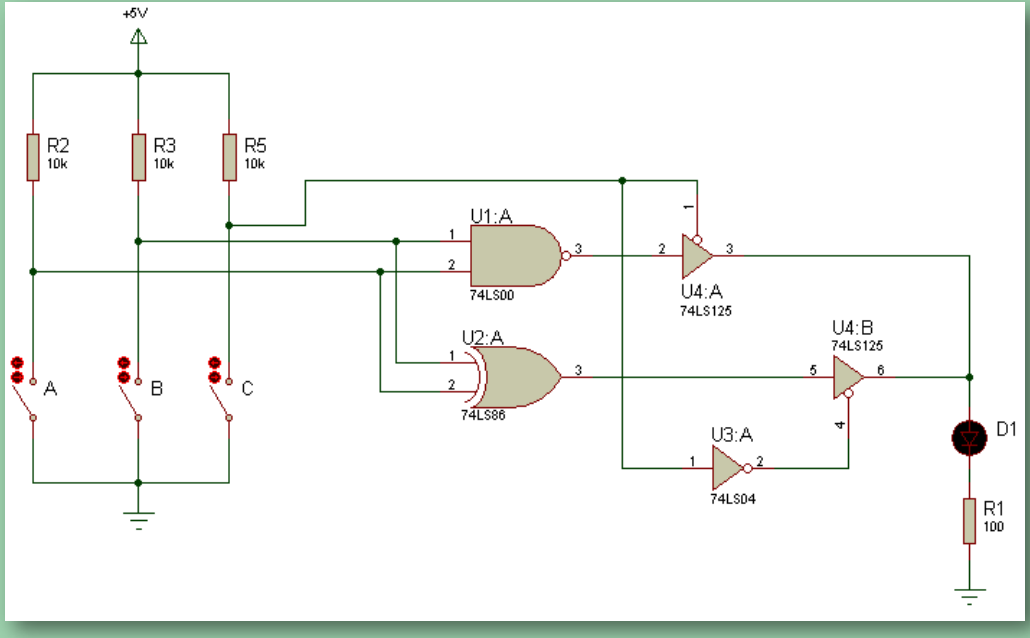
El ejemplo 6 sirve como referente para la solución.
El diagrama en bloques de la solución:





★ Ejercicio 4:

Analizar el funcionamiento del siguiente circuito digital, que consta de 3 interruptores para las señales de entrada, un conjunto de compuertas lógicas de tipo normal y Tri-State y un LED para visualizar el estado de la salida del circuito. Obtenga una tabla de verdad y su función lógica que describa el comportamiento del circuito. Simular y verificar su análisis?

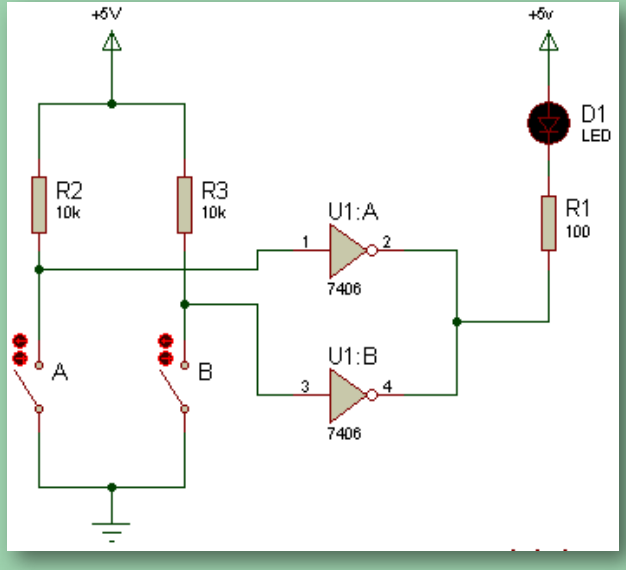


Al analizar el circuito digital se observa que la forma de conexión de los interruptores trabajan con la **"Lógica NEGATIVA"**; se obtiene la tabla de verdad y luego la función lógica:

$$Salida = (A \oplus B) + (\overline{B} \times \overline{C})$$

★ Ejercicio 5:

*Analizar el funcionamiento del siguiente circuito digital, que consta de 2 interruptores para las señales de entrada, un par de compuertas lógicas **NOT de Colector Abierto** y un LED para visualizar el estado de la salida del circuito. Obtenga una tabla de verdad y su función lógica que describa el comportamiento del circuito?. Simular y verificar su análisis.*



Al analizar el circuito digital se observa que la conexión serie entre el LED y R1 cumplen la función de la resistencia **Pull-Up** para el circuito, la forma de conexión de los interruptores es para trabajar con **"Lógica NEGATIVA"**. Obtener la tabla de verdad y luego la función lógica:

$$Led = (A + B)$$

AYUDAS

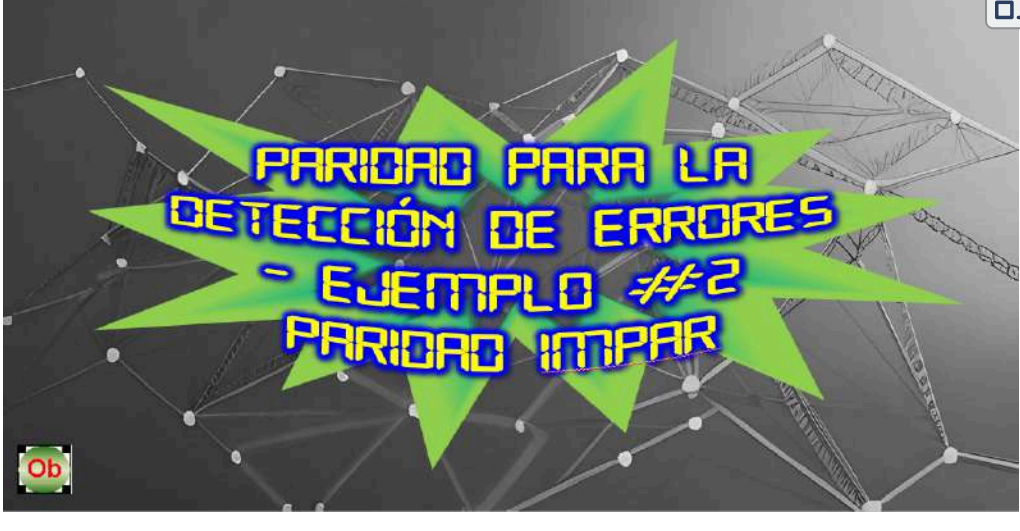
COMPLEMENTOS

3.8 Videos

Videos demostrativos sobre Comparador de Magnitudes y sobre Paridad Par e Impar [6]

[6] Videos realizados por "D&R Tutoriales" y por Marisol Maldonado Olmos en "Pasos por Ingeniería", Comparador Digital 2 bits | Compuertas lógicas y Displays | Explicación y Prueba del circuito y PARIDAD PARA LA DETECCIÓN DE ERRORES (Explicación completa) - Paridad PAR y Paridad IMPAR

Videos demostrativos sobre Paridad para la Detección de Errores ejercicio 1 - Paridad Par y ejercicio 2 - Paridad Impar [7]



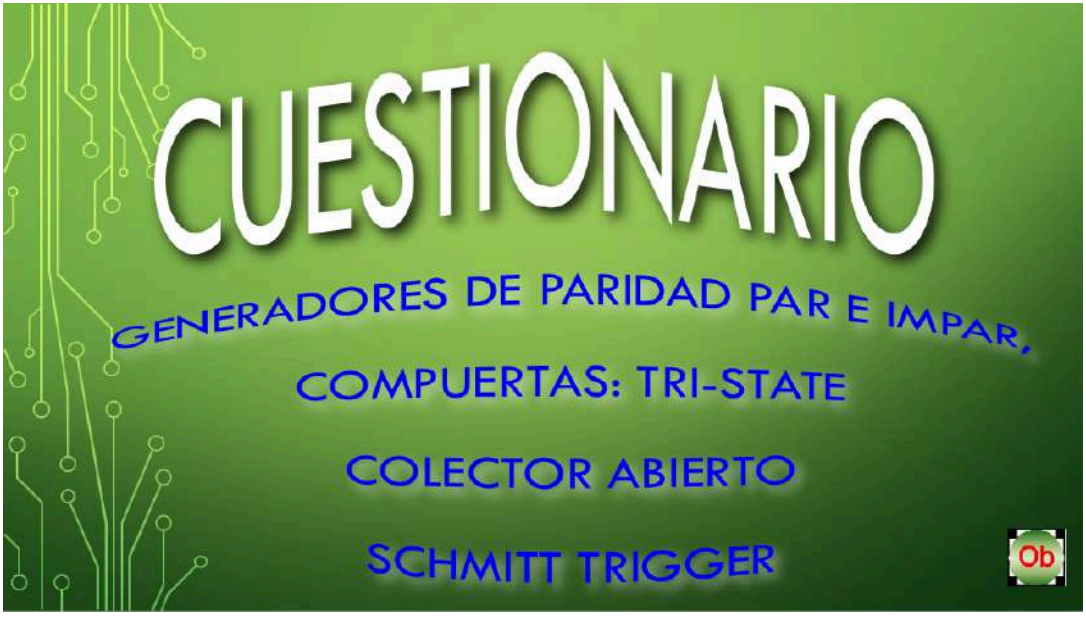
[7] Videos realizados por "Pasos por la Ingeniería", PARIDAD PARA LA DETECCIÓN DE ERRORES - Ejemplo #1 Paridad Par y PARIDAD PARA LA DETECCIÓN DE ERRORES - Ejemplo #2 Paridad Impar

3.9 Objetos Interactivos

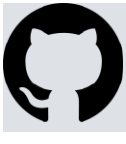
Haga clic sobre la imagen para abrir el objeto interactivo:

Este objeto interactivo fue generado con la asistencia de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/

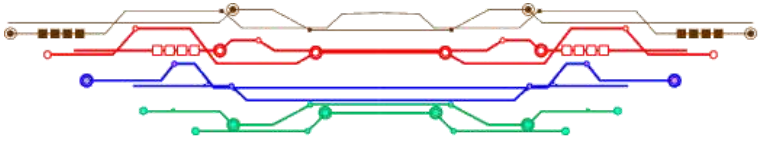
Cuestionario:



3.10 Simulaciones



Repositorio en Github con los archivos de las simulaciones de este capítulo con Proteus v7.8



EJEMPLOS
ANEXOS
Ob

Lista de Ejemplos Anexos

EA1-Secuencia Up/Down 149

EA2-Tanque de Leche 152

EA3-Banda Transportadora 157

EA4-Sistema con varios Bloques 162

EA5-Multiplexar 2 BCD en 1 display 170

EA6-Multiplexar 2 BCD en 2 display 174

EA7-Comparador y Multiplexor 178

EA8-Multiplexar 2 condiciones 181

EA9-ALU y DEMUX 185

EA10- Generador de Paridad y Verificación 191

EA11- Comparador Jugadores 195

EA12- Marquesina 201

EA13- Sistema de Tuberías 211

EA14- Balanzas 215

EA15- Tanques comunicados 221

EA1-Secuencia Up/Down

Diseñar un Sistema Combinacional que realice 2 secuencias "Manuales". Una entrada "K" controla la dirección de la secuencia, cuando está en '0' realiza la secuencia ascendente y cuando está en '1' realiza la secuencia descendente. Las secuencias son:

- *Con K='0' la secuencia es 0-1-2-3 y se recicla (repite)*
- *Con K='1' la secuencia es 3-2-1-0 y se recicla (repite)*

En caso que el sistema esté realizando una secuencia en una dirección y se cambie, debe continuar con el número correspondiente de la nueva secuencia.

Visualizar la Entrada (estado actual) y el estado siguiente en código BCD. Simule con un software las soluciones para verificar las secuencias "manuales".

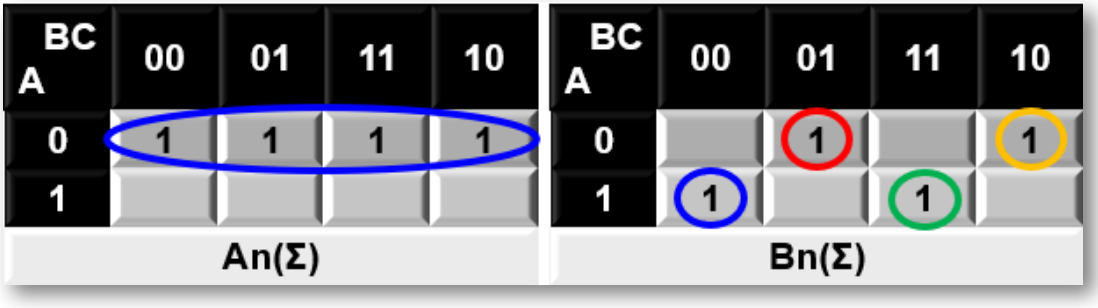
NOTA: **Este ejercicio se puede solucionar con circuitos "Secuenciales" de forma que su funcionamiento sea automático, usando Flip-Flop y un Multivibrador en configuración Astable (análisis de Máquinas de Estado Finito - MEF).**

Diagrama en bloques:

Se comienza por hallar la tabla de verdad:

Entradas			Dir	Salidas	
K	B	A	K	Bn	An
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

Se bajan los datos de la tabla a los mapas de Karnaugh para su simplificación, se realiza un mapa para cada salida binaria:



Simplificación y obtención de las funciones lógicas de cada salida binaria:

$$An(\Sigma) = \overline{A}$$

$$Bn(\Sigma) = (A \times \overline{B} \times \overline{K}) + (A \times B \times K) \times (\overline{A} + \overline{B} + K) + (\overline{A} \times B \times \overline{K})$$

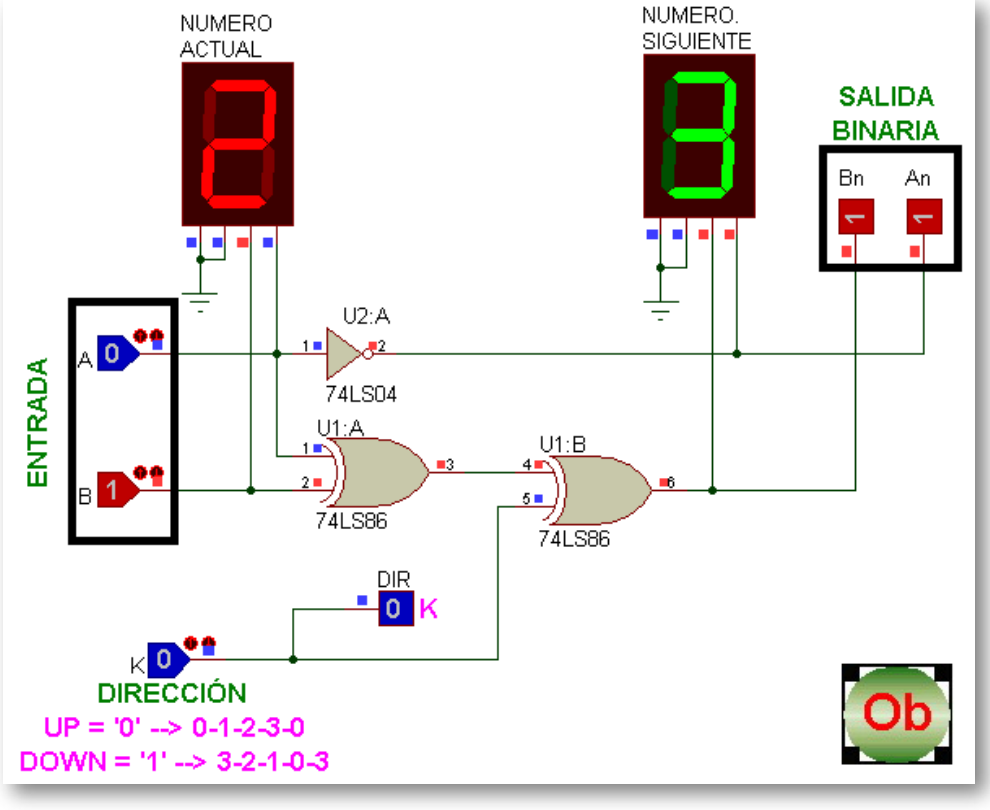
$$Bn(\Sigma) = \overline{B}\{(A \times \overline{K}) + (\overline{A} + K)\} + B\{(A \times K) + (\overline{A} \times \overline{K})\}$$

$$Bn(\Sigma) = \overline{B}\{A \oplus K\} + B\overline{\{A \oplus K\}}$$

$$\text{Si } X = A \oplus K \rightarrow Bn(\Sigma) = \overline{B}X + B\overline{X}$$

$$Bn(\Sigma) = B \oplus X \therefore Bn(\Sigma) = B \oplus A \oplus K$$

Imagen de la simulación:



Repositorio en [Github ("EA1-Secuencia UD.dsn")] con el archivo de la simulación con Proteus v7.8

EA2-Tanque de Leche

Diseñar un Sistema Combinacional teniendo en cuenta los estados de los sensores de nivel, de forma que controle la Bomba de suministro de leche (B), la Electroválvula (E) de evacuación del líquido y la Alarma (AL) por condiciones prohibidas o inconsistentes. [8]

El suministro del lácteo se realiza por medio de la bomba que utiliza dos velocidades de trabajo o activación (alta y baja). Tres sensores (S1, S2 y S3) que detectan el nivel del líquido dentro del tanque; S1 se encuentra en la parte inferior, S2 en la parte media y S3 en la parte superior. El volumen del tanque se dividide en 4 zonas denominadas: vacío, medio vacío, medio lleno y lleno. La electroválvula mediante actuadores puede regular el grado de apertura en tres grados (máxima 360°, media 180° y mínima 30°).

Cuando el nivel del lácteo se encuentra en la zona denominada "lleno", la bomba se desactiva y la electroválvula se activa a su máximo grado de apertura evacuando el producto del depósito. En la zona entre los sensores 2 y 3, la electroválvula se encuentra en el grado de apertura media y la bomba trabajando a velocidad baja. Cuando el nivel del líquido se encuentra en la zona de "medio vacío" la bomba trabaja a alta velocidad y la electroválvula tiene la mínima apertura para la evacuación. En la zona de "vacío" del tanque, la bomba trabajará a máxima velocidad de llenado y la electroválvula se desactivará.

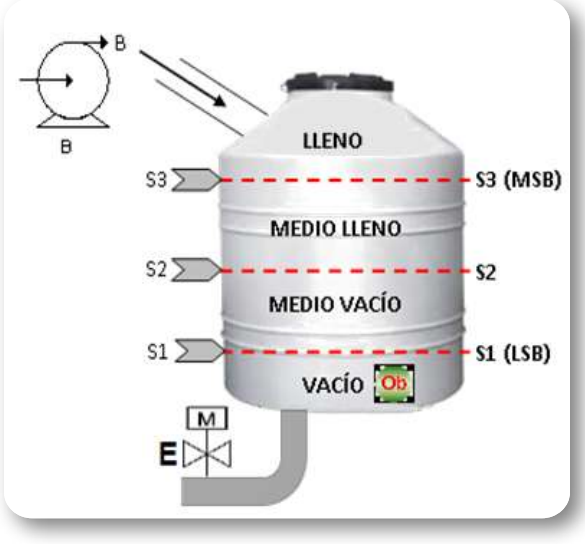
Cuando se presente un estado prohibido o de inconsistencia en el sistema, se activará una alarma auditiva y de forma automática se deben desactivar la bomba y la electroválvula por seguridad del sistema y los operarios.

[8] Condición prohibida o inconsistente: son aquellas que no se deben presentar en el sistema, son errores.

Incluir en el diseño, la visualización en código BCD las Entradas (estados actuales), los siguientes y los dispositivos.

NOTA: **Este ejercicio se puede solucionar con circuitos "Secuenciales" de forma que su funcionamiento sea automático, usando Flip-Flop y un Multivibrador en configuración Astable (análisis de Máquinas de Estado Finito - MEF).**

Diagrama del sistema:



Convenciones para el sistema:

- **BVa** = Bomba Velocidad Alta
- **BVb** = Bomba Velocidad Baja
- **E360** = Electroválvula apertura máxima
- **E180** = Electroválvula apertura media
- **E30** = Electroválvula apertura mínima
- **Al** = Alarma auditiva

Tabla de verdad completa:

Estado Act.				Estado Sgte.			Dispositivos de Salida					
Est	S3	S2	S1	S3*	S2*	S1*	Bomba		Electroválvula			
	C	B	A	C	Bn	An	BVa	BVb	E360	E180	E30	AL
E0	0	0	0	0	0	1	1	0	0	0	0	0
E1	0	0	1	0	1	0	1	0	0	0	1	0
E2	0	1	0	0	1	1	0	0	0	0	0	1
E3	0	1	1	1	0	0	0	1	0	1	0	0
E4	1	0	0	1	0	1	0	0	0	0	0	1
E5	1	0	1	1	1	0	0	0	0	0	0	1
E6	1	1	0	1	1	1	0	0	0	0	0	1
E7	1	1	1	0	0	0	0	0	1	0	0	0

Mapas de Karnaugh para las variables de Entrada (Sensores):

A \ BC	00	01	11	10
0	1	1	1	1
1				

An(Σ)

A \ BC	00	01	11	10
0			1	1
1	1	1		

Bn(Σ)

A \ BC	00	01	11	10
0		1	1	
1		1		1

Cn(Σ)

Funciones lógicas simplificadas de los Sensores:

$$An(\Sigma) = A$$
$$Bn(\Sigma) = (A \times \overline{B}) + (\overline{A} \times B)$$
$$Bn(\Sigma) = (A \oplus B)$$
$$Cn(\Sigma) = (\overline{B} \times C) + (\overline{A} \times C) + (A \times B \times \overline{C})$$

Mapas de Karnaugh para la Bomba:

A \ BC	00	01	11	10
0	1			
1	1			

BVa(Σ)

A \ BC	00	01	11	10
0				
1				1

BVb(Σ)

Funciones lógicas simplificadas de la Bomba:

$$BVa(\Sigma) = (\overline{B} \times \overline{C})$$
$$BVb(\Sigma) = (A \times B \times \overline{C})$$

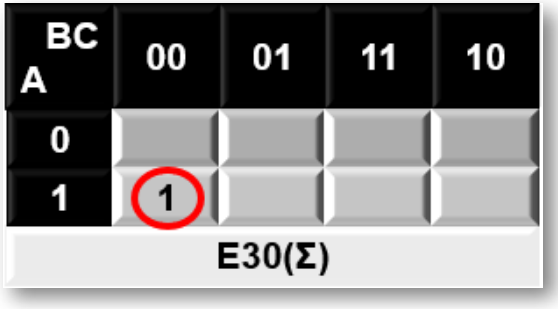
Mapas de Karnaugh para la apertura de la Electroválvula:

A \ BC	00	01	11	10
0				
1			1	

E360(Σ)

A \ BC	00	01	11	10
0				
1				1

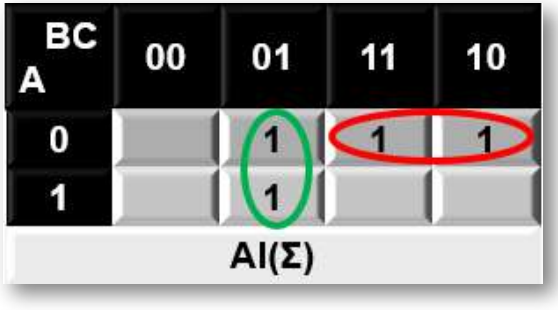
E180(Σ)



Funciones lógicas simplificadas de la Electroválvula:

$$E360(\Sigma) = (A \times B \times C)$$
$$E180(\Sigma) = (A \times B \times \overline{C})$$
$$E30(\Sigma) = (A \times \overline{B} \times \overline{C})$$

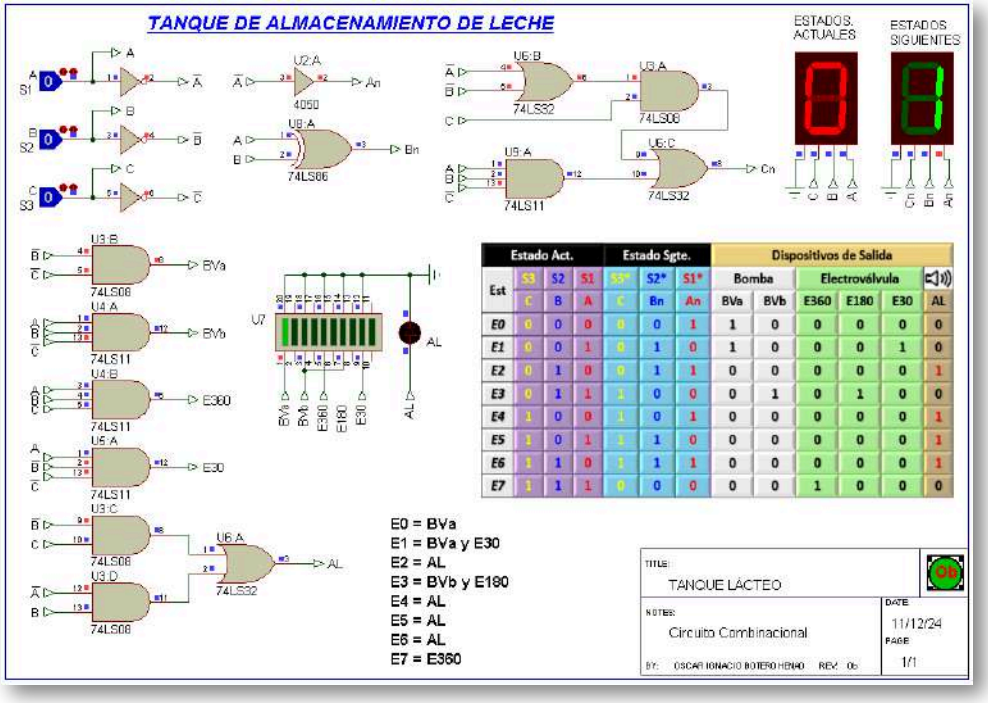
Mapas de Karnaugh para la Alarma auditiva:



Función lógica simplificada de la Alarma auditiva:

$$AL(\Sigma) = (\overline{B} \times C) + (\overline{A} \times B)$$

Imagen de la simulación:



Estado Act.				Estado Sgte.			Dispositivos de Salida					
Est	S3	S2	S1	S3*	S2*	S1*	Bomba		Electroválvula			
	C	B	A	C	Bn	An	BVa	BVb	E360	E180	E30	AL
E0	0	0	0	0	0	1	1	0	0	0	0	0
E1	0	0	1	0	1	0	1	0	0	0	1	0
E2	0	1	0	0	1	1	0	0	0	0	0	1
E3	0	1	1	1	0	0	0	1	0	1	0	0
E4	1	0	0	1	0	1	0	0	0	0	0	1
E5	1	0	1	1	1	0	0	0	0	0	0	1
E6	1	1	0	1	1	1	0	0	0	0	0	1
E7	1	1	1	0	0	0	0	0	1	0	0	0

Repositorio en Github ("EA2-Tanque de Leche.dsn") con el archivo de la simulación con Proteus v7.8

EA3-Banda Transportadora

Un sensor de proximidad detecta una pieza y envía un pulso negativo al control de secuencia con el que se inicia el proceso. Después de transcurrido el primer segundo la Banda Transportadora 1 se detiene por 2 segundos, luego de 1 segundo de estar detenida se activa por 1 segundo el Émbolo Retráctil que envía la pieza hacia la segunda Banda Transportadora; cuando la Banda Transportadora 2 recibe la pieza se activa y en simultánea la Banda Transportadora 1, hasta que el sensor de proximidad detecte otra pieza y comience de nuevo todo el proceso.

Dispositivos:

- *1 sensor de proximidad (Sensor)*
- *1 motor para la banda transportadora 1 (BT1)*
- *1 motor para la banda transportadora 2 (BT2)*
- *1 émbolo retráctil (E) para empujar las piezas*

Diseñar el circuito Combinacional, donde se visualicen en código BCD los estados actuales, los siguientes y los dispositivos del sistema.

NOTA: **Este ejercicio se puede solucionar con circuitos "Secuenciales" de forma que su funcionamiento sea automático, usando Flip-Flop y un Multivibrador en configuración Astable (análisis de Máquinas de Estado Finito - MEF).**

Imagen del sistema:

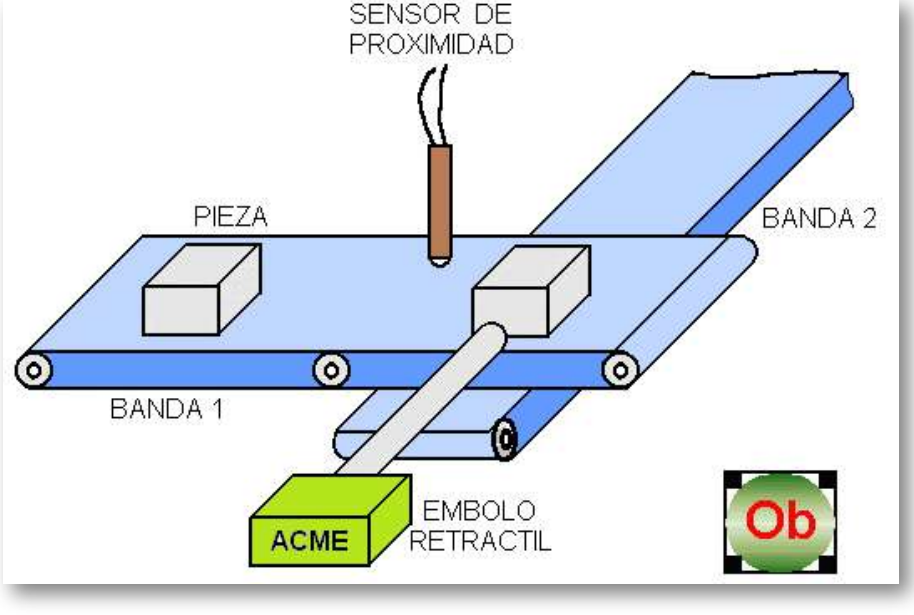


Diagrama de tiempos:

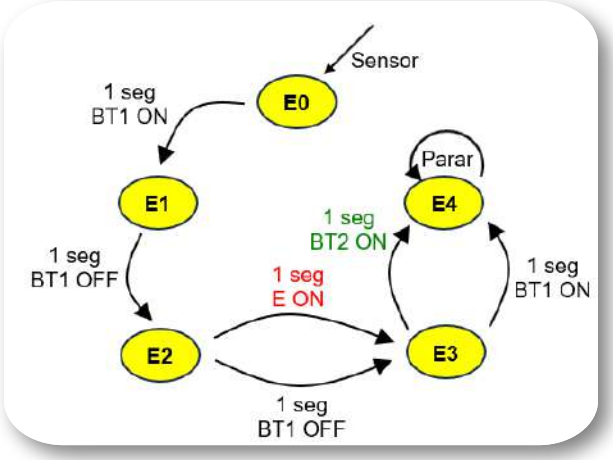
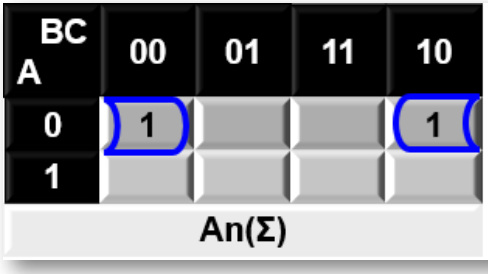


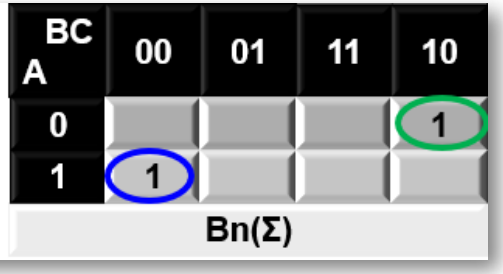
Tabla de verdad del sistema:

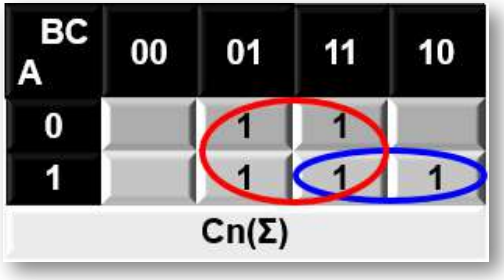
Estado Act.				Estado Sgte.			Dispositivos		
Estados							Bandas		Émbolo
	C	B	A	Cn	Bn	An	BT1	BT2	E
E0	0	0	0	0	0	1	*1*	*0*	*0*
E1	0	0	1	0	1	0	*1*	*0*	*0*
E2	0	1	0	0	1	1	*0*	*0*	*0*
E3	0	1	1	1	0	0	*0*	*0*	*1*
E4	1	0	0	1	0	0	*1*	*1*	*0*
-	1	0	1	1	0	0	-	-	-
-	1	1	0	1	0	0	-	-	-
-	1	1	1	1	0	0	-	-	-

Las combinaciones de los números binarios del 5, 6 y 7 no se utilizan en la secuencia de los estados y por seguridad las bandas transportadoras se apagan.

Mapas de Karnaugh para la secuencia de los estados:



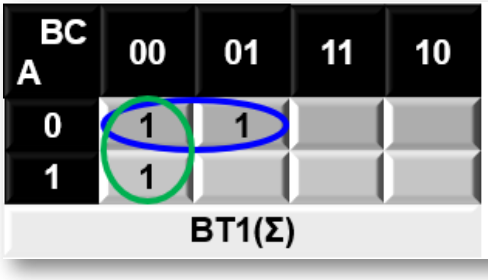


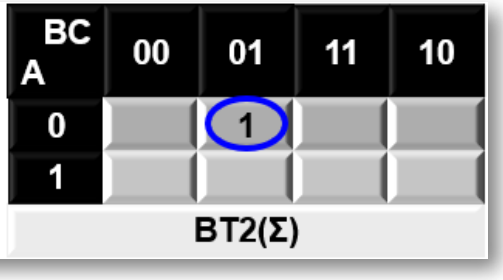


Funciones lógicas simplificadas de los estados:

$$An(\Sigma) = (\overline{A} \times \overline{C})$$
$$Bn(\Sigma) = (A \times \overline{B} \times \overline{C}) + (\overline{A} \times B \times \overline{C})$$
$$Bn(\Sigma) = \overline{C}(A \oplus B)$$
$$Cn(\Sigma) = (A \times B) + (C)$$

Mapas de Karnaugh para las Bandas Transportadoras:





Funciones lógicas simplificadas de las Bandas Transportadoras:

$$BT1(\Sigma) = (\overline{A} \times \overline{B}) + (\overline{B} \times \overline{C})$$
$$BT2(\Sigma) = (\overline{A} \times \overline{B} \times C)$$

- Mapa de Karnaugh para el Émbolo:

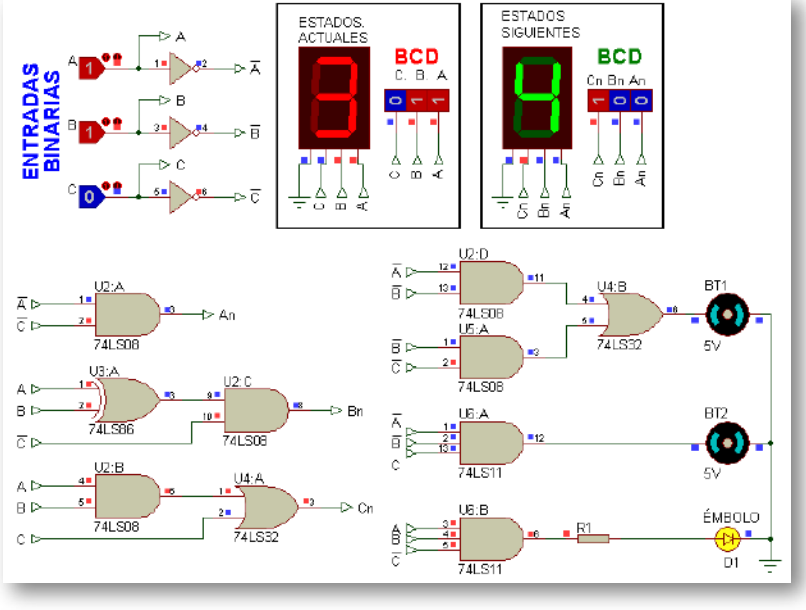
A \ BC	00	01	11	10
0				
1				1

E(Σ)

- Función lógica simplificada del Émbolo:

$$E(\Sigma) = (A \times B \times \overline{C})$$

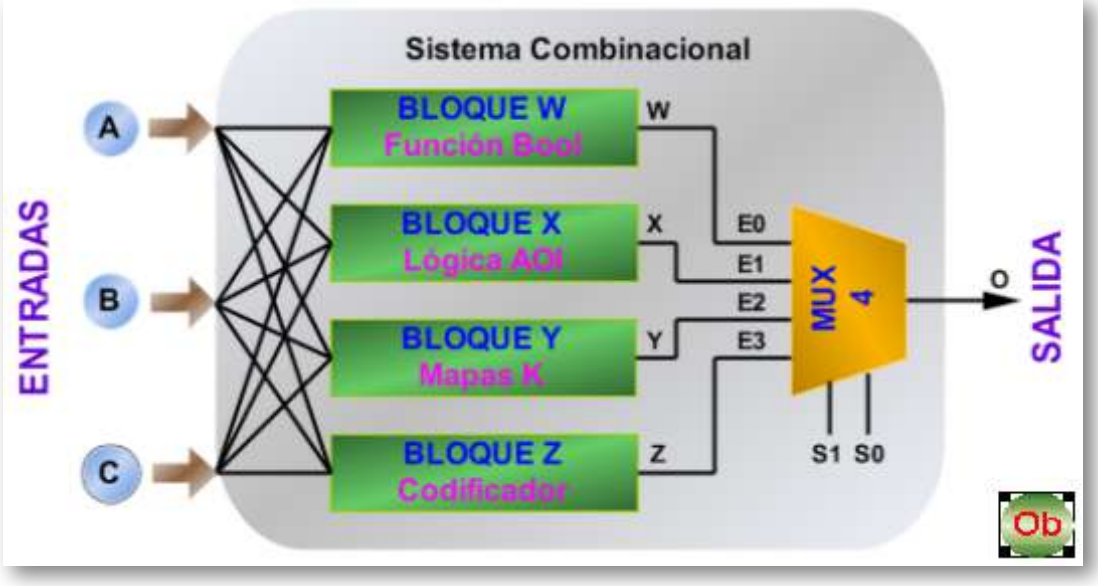
- Imagen de la simulación:



Repositorio en Github ("EA3-Bandas Transportadoras.dsn") con el archivo de la simulación con Proteus v7.8

EA4-Sistema con varios Bloques

Se tiene un Sistema Combinacional conformado por 4 bloques ***(X, Y, Z, W)****, con 3 entradas comunes a cada uno y una salida que se conecta a un Multiplexor 4:1, con el que se seleccionarán los datos del bloque que serán transferidos a la salida. Simule con un software las soluciones para verificar los resultados parciales y la salida (O).*



➤ *BLOQUE W*

Bloque W: *Halle la tabla de verdad e implemente el circuito combinacional para la función lógica:* $W = \{(\overline{A} \times B \times \overline{C}) + \overline{B}\} \oplus \overline{C}$*, simule con un software la solución y verifique el resultado.*

$$W = \underbrace{\{\overbrace{(\underbrace{\overline{A}}_{1} \times B \times \underbrace{\overline{C}}_{3})}^{4} + \underbrace{\overline{B}}_{2}\}}_{5} \oplus \underbrace{\overline{C}}_{3}$$

- Se inicia con el diseño de la tabla, colocando las variables de Entrada $''A\,, B, \, C''$ de la función, con todas las combinaciones de valores binarios posibles.

- En la función se observa que las tres variables de entrada están negadas, por ello, primero se solucionan las negaciones: "A", "B" y "C" que son las columnas etiquetadas como "1", "2" y "3"

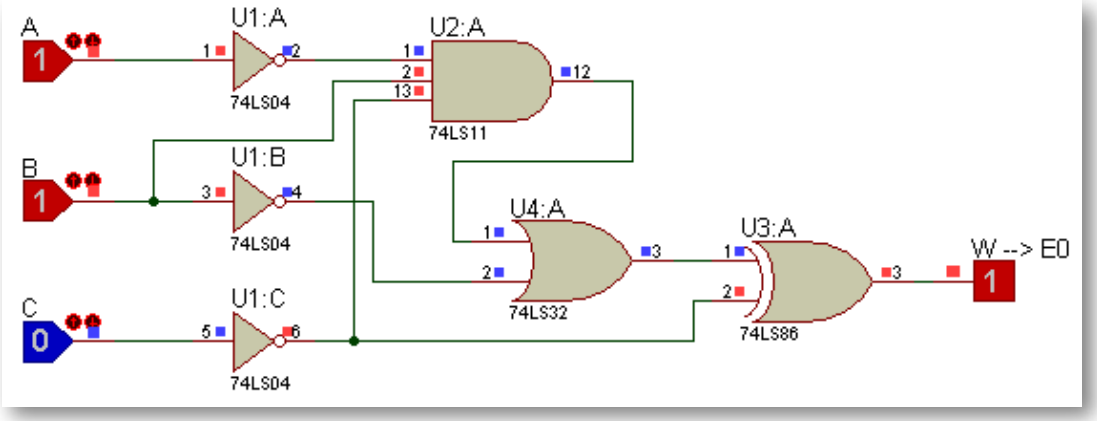
- Una compuerta **AND** de 3 entradas: 4 = $(\overline{A} \times B \times \overline{C})$

- Una compuerta **OR** de 2 entradas entre los términos etiquetados como: 5 = 4 + 2, solucionando todo el término incluido dentro de las llaves.

- Una compuerta **XOR** de 2 entradas entre los términos etiquetados como: ***W*** = 5 $\oplus$ 3, obteniendo la respuesta final.

Entradas			1	2	3	4	5	W
C	B	A	$\overline{A}$	$\overline{B}$	$\overline{C}$	1 x B x 3	4 + 2	5 $\oplus$ 3
0	0	0	1	1	1	0	1	*0*
0	0	1	0	1	1	0	1	*0*
0	1	0	1	0	1	1	1	*0*
0	1	1	0	0	1	0	0	*1*
1	0	0	1	1	0	0	1	*1*
1	0	1	0	1	0	0	1	*1*
1	1	0	1	0	0	0	0	*0*
1	1	1	0	0	0	0	0	*0*
			NOT	NOT	NOT	AND	OR	XOR

Ejecutar la simulación y validar la solución obtenida en la tabla



Repositorio en Github ("EA4-Bloque_W_E0.dsn") con el archivo de la simulación con Proteus v7.8

➤ *BLOQUE X*

Bloque X: *en un sistema de seguridad se desactiva la alarma cuando suceden las siguientes condiciones:*

- *La variable 'A' está en el estado lógico '0' y la variable 'B' no lo está.*
- *Las tres variables no se encuentren negadas.*

Simule con un software la solución y verifique el resultado

La salida se desactiva con las siguientes combinaciones de entrada:

C	B	A	X	Con	POS
0	1	0	0	a)	$(A+\overline{B}+C)$
1	1	0	0	a)	$(A+\overline{B}+\overline{C})$
1	1	1	0	b)	$(\overline{A}+\overline{B}+\overline{C})$

Se genera la tabla de verdad completa:

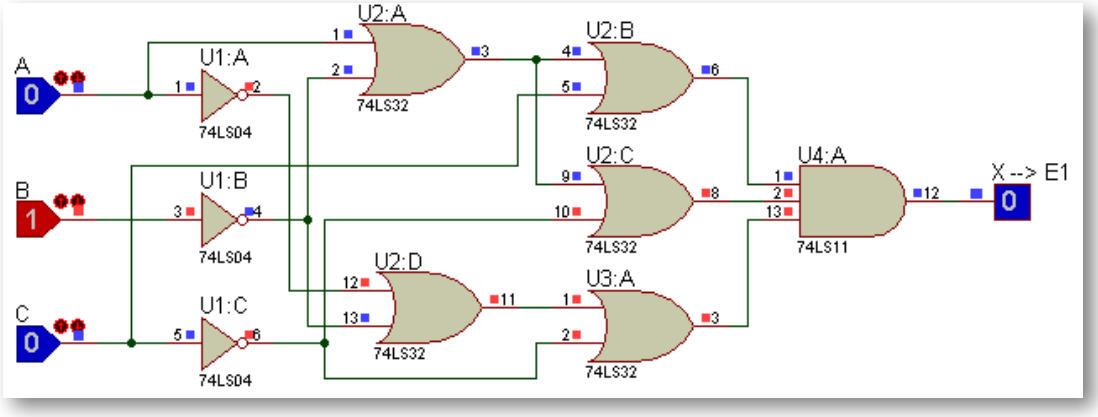
C	B	A	X	Con	POS
0	0	0	1		
0	0	1	1		
0	1	0	0	a)	$(A+\overline{B}+C)$
0	1	1	1		
1	0	0	1		
1	0	1	1		
1	1	0	0	a)	$(A+\overline{B}+\overline{C})$
1	1	1	0	b)	$(\overline{A}+\overline{B}+\overline{C})$

La función lógica es:

$$\mathbf{X} = (\mathbf{A}+\overline{\boldsymbol{B}}+\boldsymbol{C}) \times (\mathbf{A}+\overline{\boldsymbol{B}}+\overline{\boldsymbol{C}}) \times (\overline{\boldsymbol{A}}+\overline{\boldsymbol{B}}+\overline{\boldsymbol{C}})$$

Repositorio en Github ("EA4-Bloque_X_E1.dsn") con el archivo de la simulación con Proteus v7.8

El plano esquemático del circuito simulado:

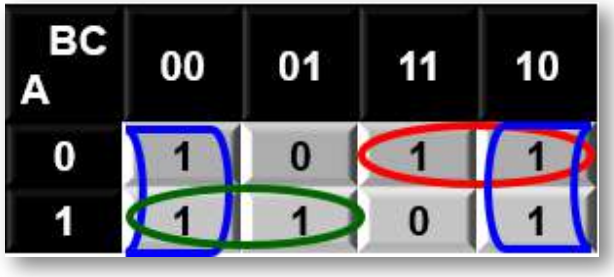


➤ *BLOQUE Y*

Bloque Y: *se tiene una tabla que se requiere simplificar al máximo por medio de mapas de Karnaugh. Simule con un software la solución y verifique el resultado.*

Entradas			Salida
C	B	A	Y = E2
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Mapa de Karnaugh y agrupaciones:



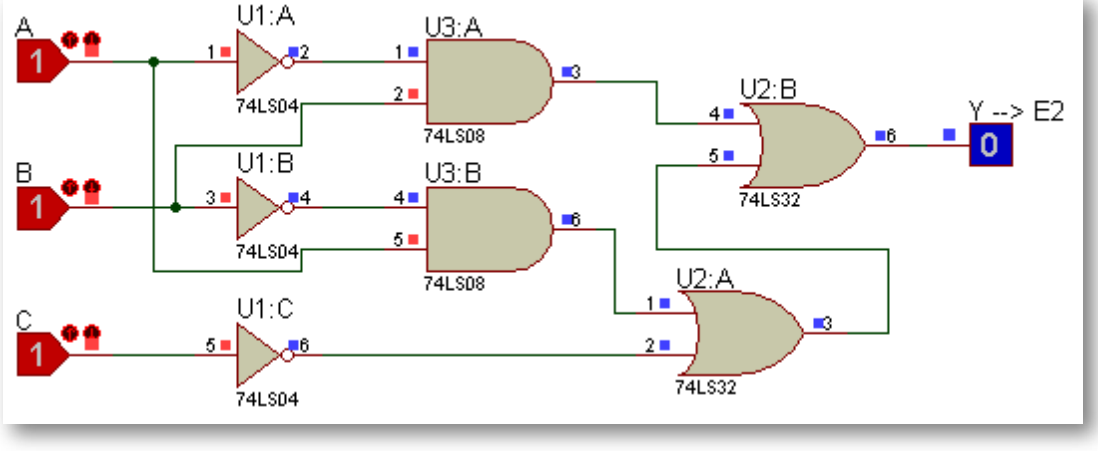
A \ BC	00	01	11	10
0	1	0	1	1
1	1	1	0	1

Función simplificada:

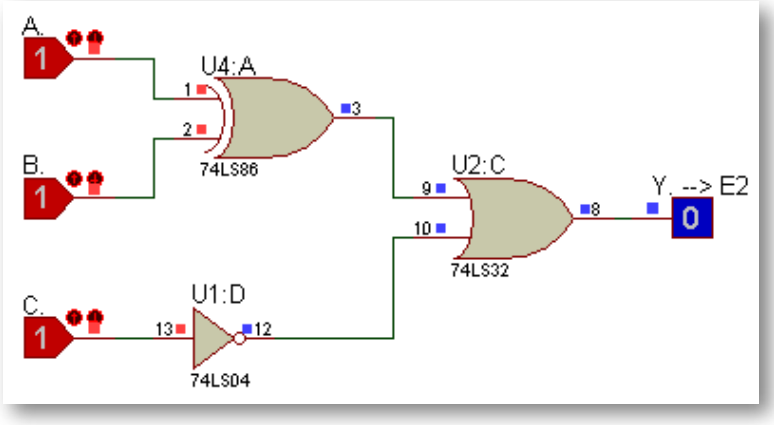
$$Y = (\overline{C}) + (\overline{A} \times B) + (A + \overline{B})$$

$$Y = (\overline{C}) + (A \oplus B)$$

Plano esquemático del circuito simulado:



Simplificando los términos: $Y = (\overline{A} \times B) + (A \times \overline{B})$ equivale a una compuerta **XOR**: $Y = (A \oplus B)$



Repositorio en [Github ("EA4-Bloque_Y_E2.dsn")] con el archivo de la simulación con Proteus v7.8

➤ *BLOQUE Z*

Bloque Z: *se tiene un sistema digital con 3 pulsadores y se requiere que cuando NO se presione algún pulsador o solo el pulsador 'A', la salida se active, de lo contrario no se activa la salida. Diseñar el circuito Codificador que cumpla con estas condiciones, simule con un software la solución y verifique el resultado obtenido.*

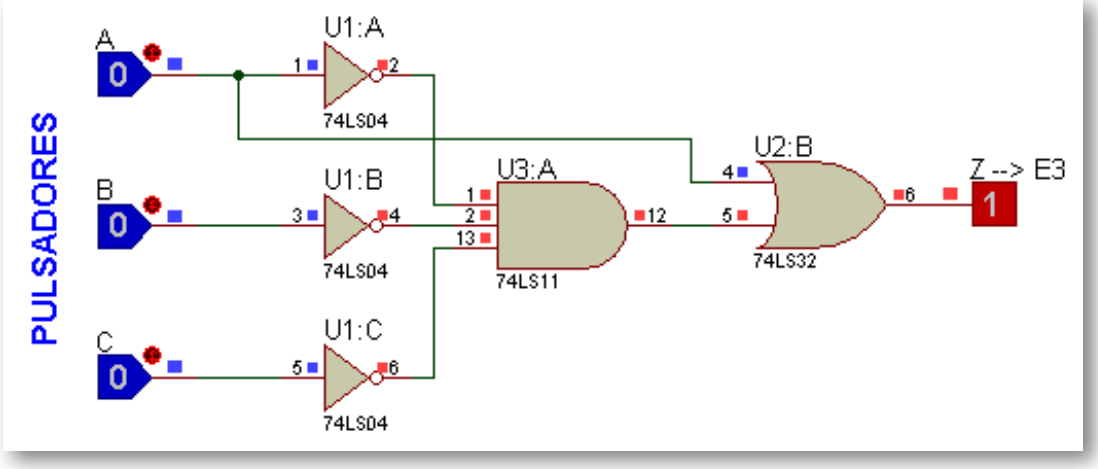
Tabla de verdad generada con las condiciones del enunciado:

Entradas			Sal
C	B	A	Z=E3
0	0	0	1
0	0	1	1
0	1	0	0
1	0	0	0

El codificador cumple con el 'Caso 2'(tratado en la sección 5.1), donde hay una fila que no presenta cambios en las variables de entrada y su función lógica es:

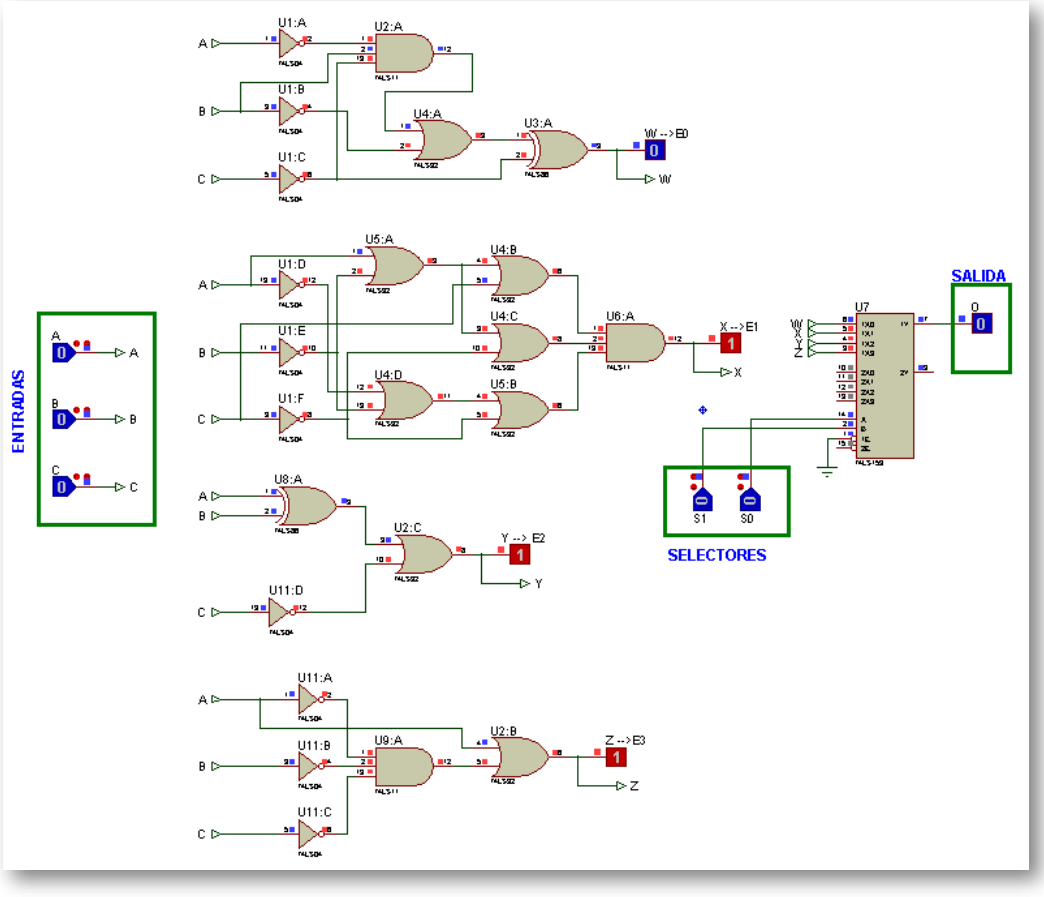
$$Z(\Sigma) = (\overline{A} \times \overline{B} \times \overline{C}) + (A)$$

Plano esquemático del circuito simulado:



Repositorio en Github ("EA4-Bloque_Z_E3.dsn") con el archivo de la simulación con Proteus v7.8

Luego de verificado cada bloque de forma individual, se procede a unificarlos en un solo archivo, de forma que se pueda ejecutar la simulación y verificación completa.

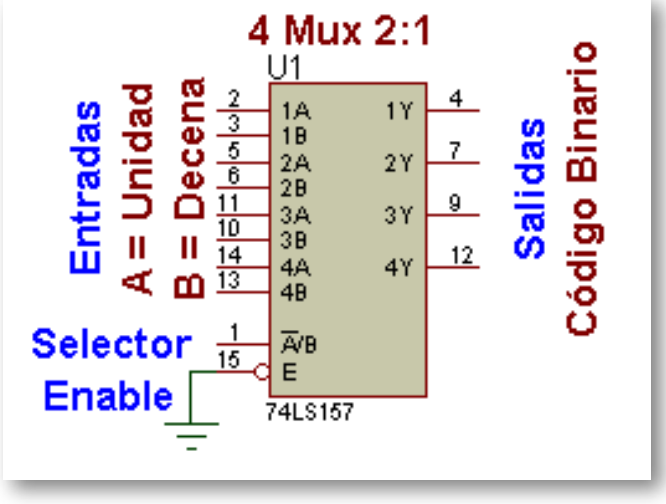


Repositorio en Github ("EA4-Bloque O.dsn") con el archivo de la simulación con Proteus v7.8

EA5-Multiplexar 2 BCD en 1 display

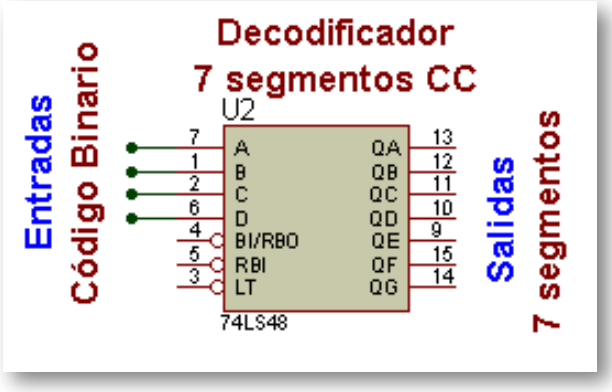
Diseñar un circuito que Multiplexe 2 números BCD teniendo un solo circuito integrado decodificador de 7 segmentos y 1 solo display de 7 segmentos. Simularlo y verificar su funcionamiento.

El cuádruple Multiplexor de 2:1 referencia 74LS157 recibe las 8 entradas pertenecientes a los 2 números BCD, 4 entradas para el número de la "Unidad" (pines 2,5,11,14) y 4 para el número de la "Decena" (pines 3,6,10,13); y entrega en sus 4 salidas (pines 4,7,9,12) el número en código binario, de acuerdo al estado de la línea de control (pin 1), con '0' lógico entrega el número del bloque "A" perteneciente a la "Unidad" y con '1' el del bloque "B" asignado a la "Decena". La entrada Enable (E, pin 15) se activa en nivel bajo y es el encargado de habilitar el circuito integrado.

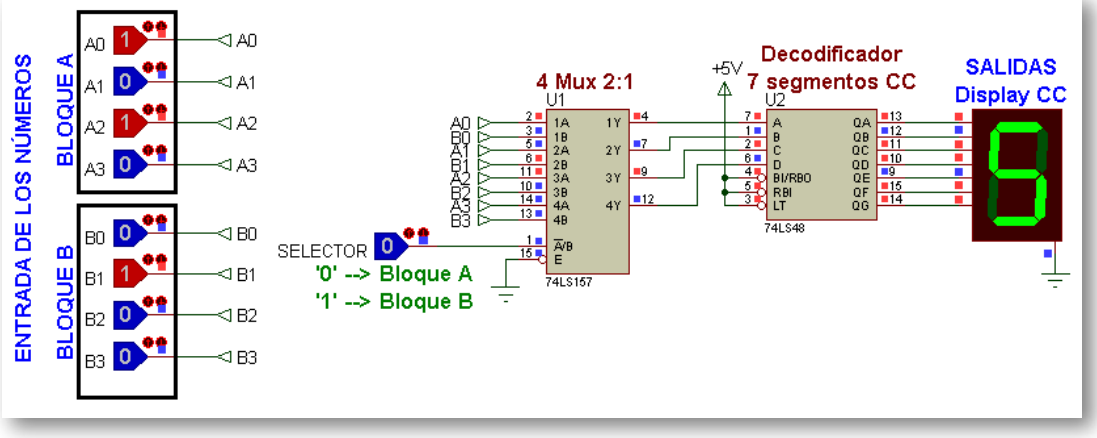


Para los display de ánodo común se utiliza el Decodificador con referencia 7447 y para los de cátodo común el Decodificador 7448.

Utilizando el Decodificador de 7 segmentos de cátodo común 7448, que recibe el número en código binario (pines 1,2,6,7) y lo entrega por las 7 salidas (pines 9,10,11,12,13,14,15) que controlan los 7 segmentos del display.

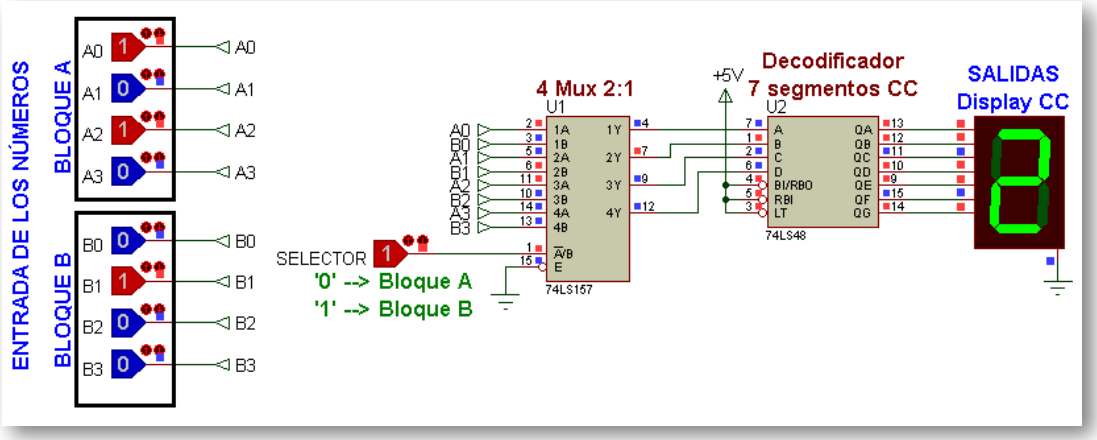


- Como ejemplo, se asigna el número "25" en los interruptores de las entradas BCD (Bloque A= 0101 y Bloque B=0010).
- Colocando en '0' el pin 1 del MUX se seleccionan los datos del bloque A y se visualiza el número '5':

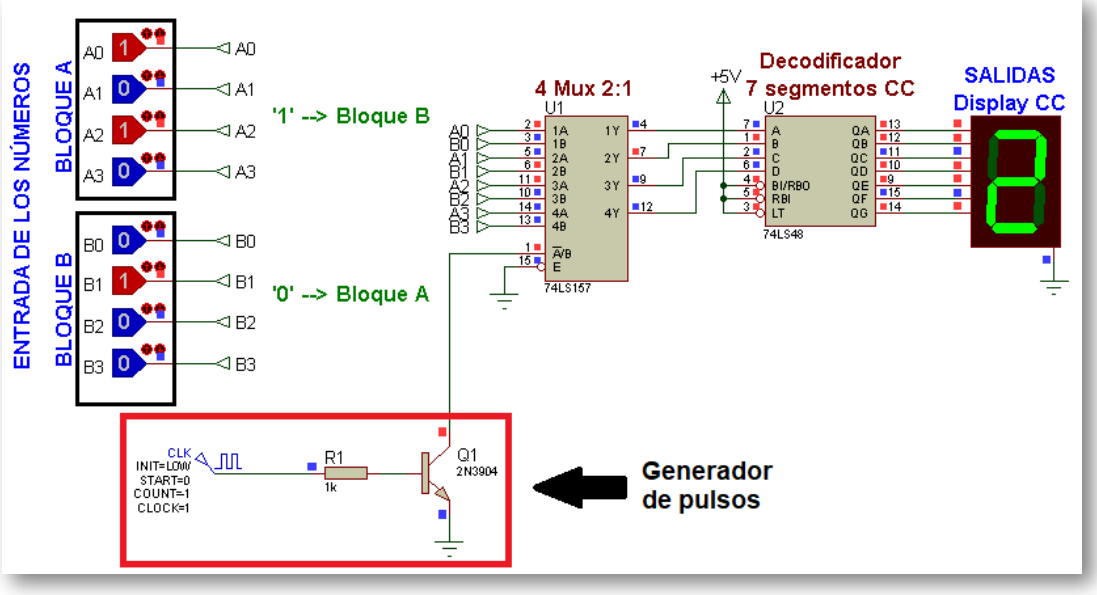


- Colocando en '1' el pin 1 del MUX se seleccionan los datos del bloque B y se visualiza el número '2':

Repositorio en [Github ("EA5-Mux_2BCD_1Display.dsn")] con el archivo de la simulación con Proteus v7.8



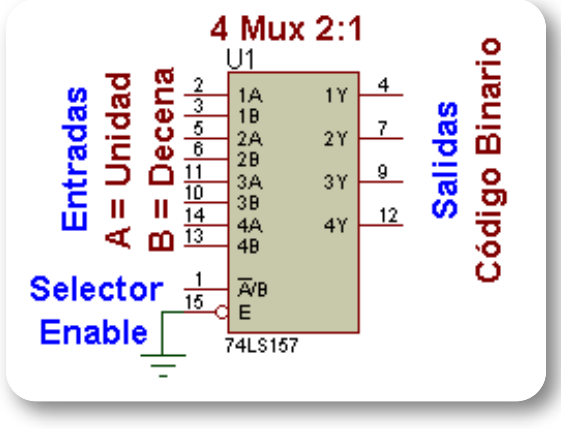
El **Selector** se puede reemplazar por un circuito que genere pulsos como se observa en la siguiente figura:



Repositorio en [Github ("Mux_2BCD_1Display_clk.dsn")] con el archivo de la simulación con Proteus v7.8

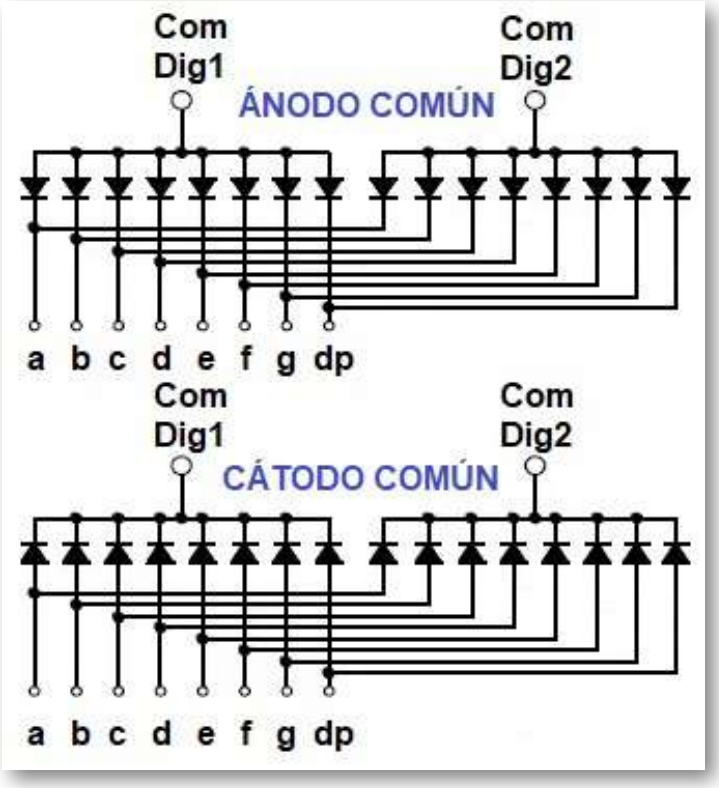
EA6-Multiplexar 2 BCD en 2 display

Diseñar un circuito Multiplexor de 2 números BCD teniendo un solo circuito integrado decodificador de 7 segmentos y un display Multidígito Multiplexado de 2 dígitos de 7 segmentos. Simular y verificar su funcionamiento.



El cuádruple Multiplexor de 2:1 referencia 74LS157 recibe las 8 entradas pertenecientes a los 2 números BCD, 4 entradas para el número de la "Unidad" (pines 2,5,11,14) y 4 para el número de la "Decena" (pines 3,6,10,13); y entrega en sus 4 salidas (pines 4,7,9,12) el número en código binario, de acuerdo al estado de la línea de control (pin 1), con '0' lógico entrega el número del bloque "A" perteneciente a la "Unidad" y con '1' el del bloque "B" asignado a la "Decena". La entrada Enable (E, pin 15) es activa en nivel bajo y se encarga de habilitar el uso del circuito integrado.

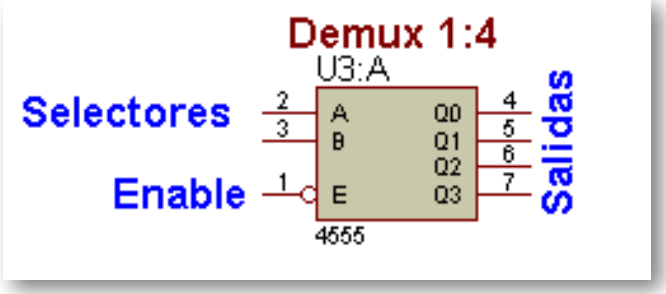
El display **Multidígito Multiplexado** que se va a utilizar consta de dos dígitos y tiene unidos los **7 segmentos equivalentes** incluyendo el punto decimal (DP). Se comercializan de 2, 3 y 4 dígitos de ánodo común y de cátodo común como se observa en la siguiente figura:



Para los display de ánodo común se utiliza el Decodificador con referencia 7447 y para los de cátodo común el Decodificador 7448. La distribución de las terminales en ambos es la misma.

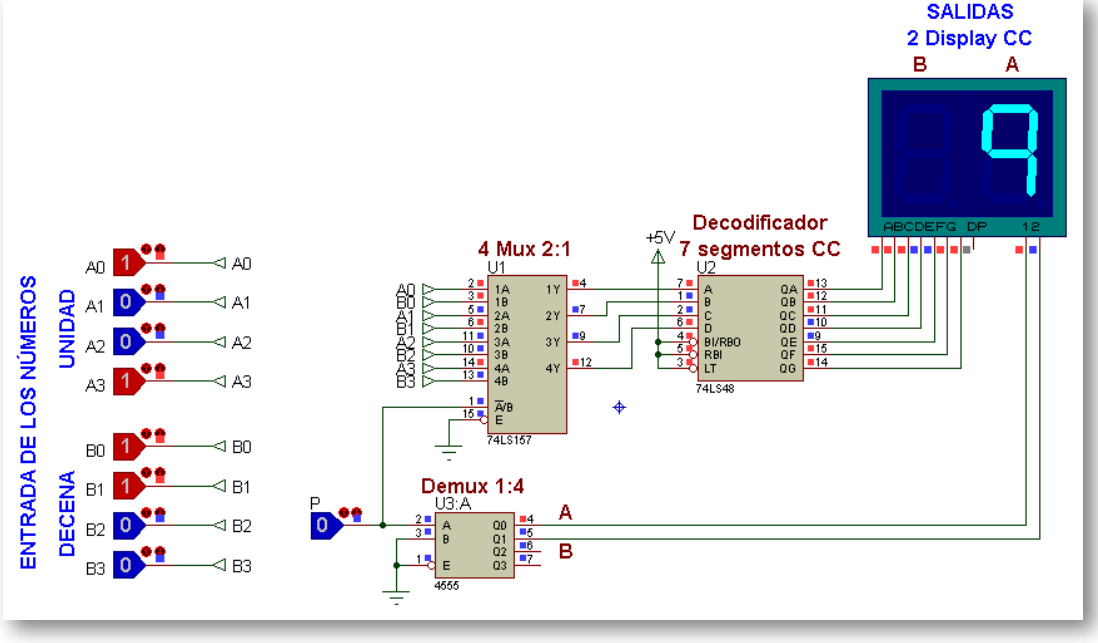
Utilizando el Decodificador de 7 segmentos de cátodo común 7448, que recibe el número en código binario (pines 1,2,6,7) y lo entrega por las 7 salidas (pines 9,10,11,12,13,14,15) que controlan los 7 segmentos del display.

Con el Demultiplexor de 1:4 referencia 4555 se activan los 2 cátodos comunes de los display, utilizando solo 2 salidas (pines 4 y 5) y una sola línea de selección (pin 2); con '0' se activa el cátodo común del display de la "Unidad" y con '1' se activa el del display de la "Decena".

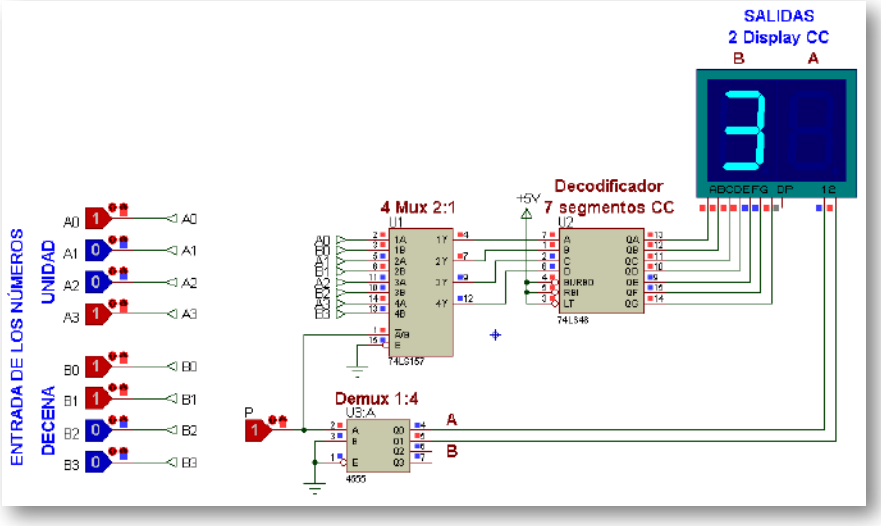


Como ejemplo, se asigna el número "39" en los interruptores de las entradas BCD (Bloque A= 1001_b y Bloque B=0011_b).

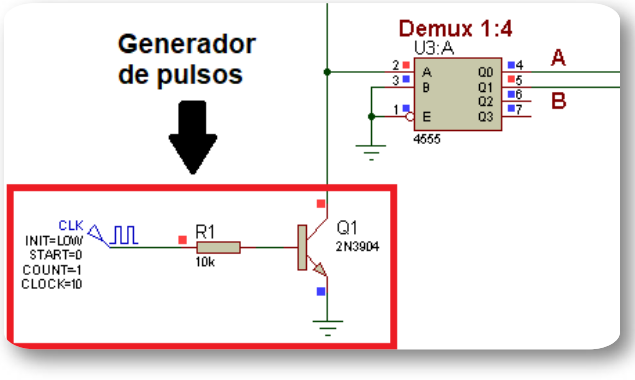
Colocando en '0' el pin 2 del Demux se activa la salida Q0 que a su vez activa el cátodo perteneciente al display de la **"Unidad"**:



Colocando en '1' el pin 2 del Demux se activa la salida Q1 que a su vez activa el cátodo perteneciente al display de la **"Decena"**:



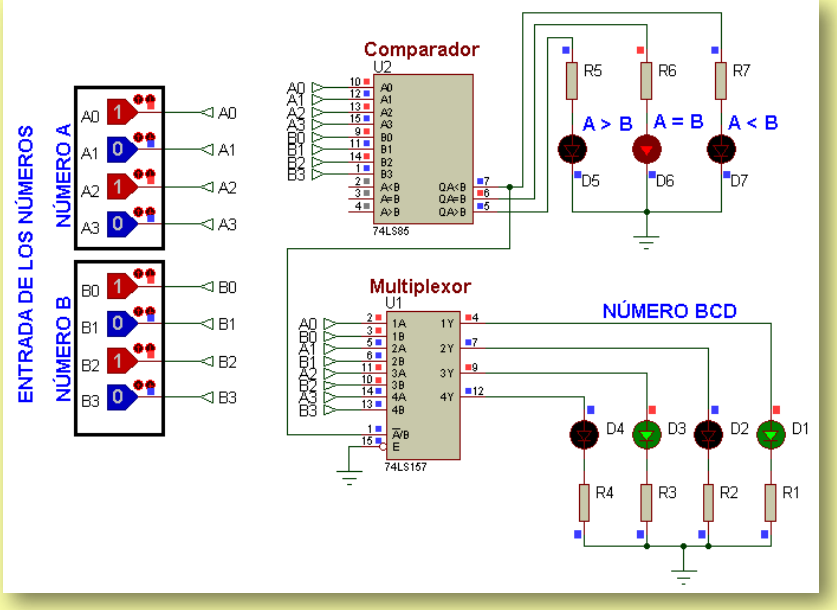
El **Selector** se puede reemplazar por un circuito que genere pulsos como se observa en la figura, tener presente que la simulación no responde correctamente al aumentar la frecuencia, pero en un circuito implementado de forma física se observarían correctamente todos los datos de forma simultánea.



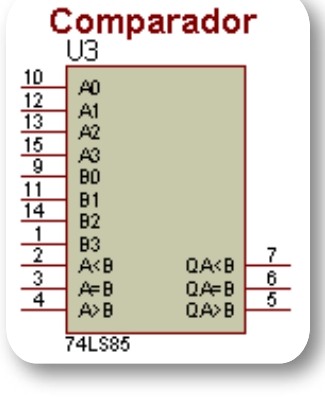
Repositorio en Github ("EA6-Mux_2BCD_2Display.dsn") y Github ("EA6-Mux_2BCD_2Display_clk.dsn") con los archivos de la simulación con Proteus v7.8

EA7-Comparador y Multiplexor

Analizar el circuito combinacional utilizando el circuito integrado comparador de magnitudes con referencia 74LS85 que permite comparar 2 números BCD y activar una de las salidas, dependiendo si el número $A > B, A = B, A < B$. y un circuito integrado Multiplexor referencia 74LS157.

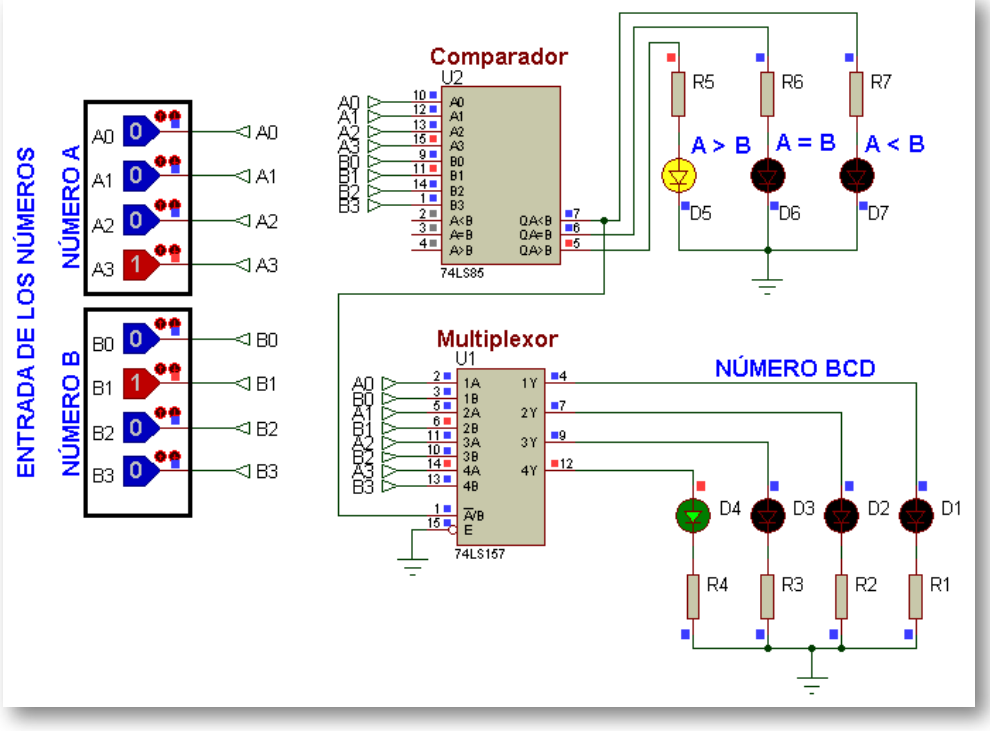


El circuito integrado 7485 compara dos números en BCD (**D**ecimal **C**odificado en **B**inario) conformado por 4 bits para el "número A" y 4 bits para el "número B" y se activa la salida que corresponde si $A > B, A = B$ o $A < B$.

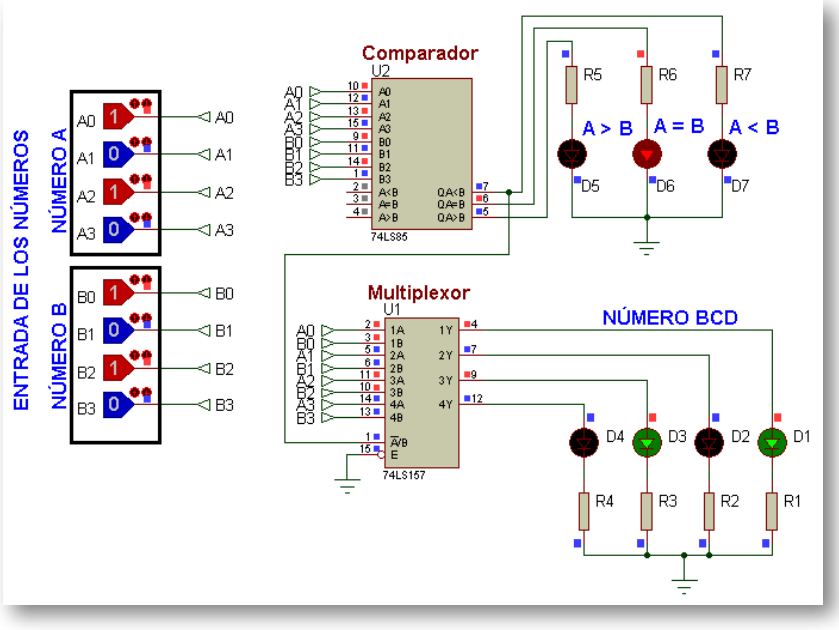


- En las 4 salidas del Multiplexor (pines 4,7,9,12) se conectan 4 led que muestran el número **MAYOR** en BCD.
- Si $A > B$, la salida pin 7 del Comparador envía un '0' a la entrada del selector del Multiplexor (pin 1) y muestra en sus salidas el "número A" que corresponde al número mayor.
- Si $A = B$, la salida pin 7 del Comparador envía un '0' a la entrada del selector del Multiplexor (pin 1) y muestra en sus salidas el "número A" que es igual al "número B".
- Si $A < B$, la salida pin 7 del Comparador envía un '1' a la entrada del selector del Multiplexor (pin 1) y muestra en sus salidas el "número B" que corresponde al mayor.

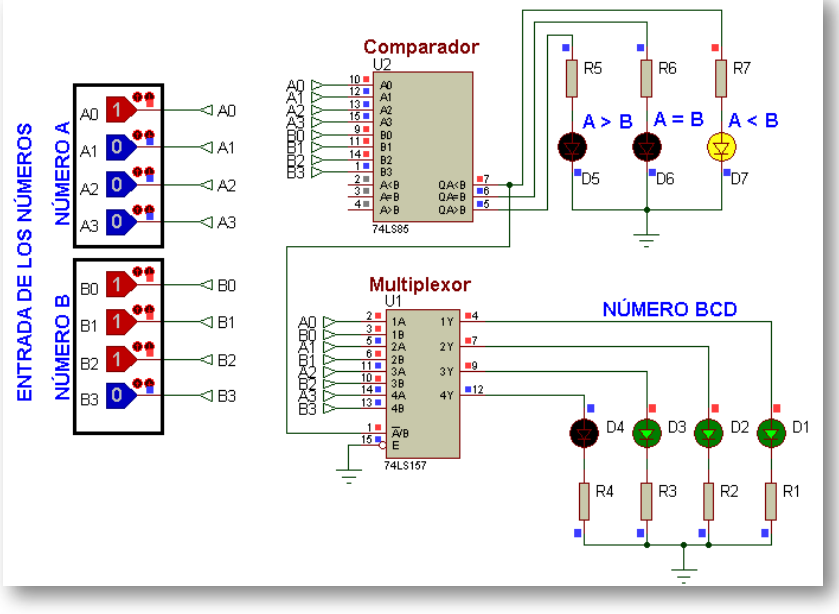
- ***Caso 1:*** si los números son el 8 y el 2 respectivamente ($A > B$):



Caso 2: si ambos números son el 5 ($A = B$):



Caso 3: si los números son el 1 y el 7 respectivamente ($A < B$):



Repositorio en Github ("EA7-Comparador_Mux.dsn") con el archivo de la simulación con Proteus v7.8

EA8-Multiplexar 2 condiciones

*Se requiere diseñar un circuito combinacional con 4 entradas que genere una salida Z que se active cuando cumpla una de las dos condiciones, pero **NO** las dos:*

1. *Las entradas A y B están activadas*
2. *Las entradas C o D o ambas estén activadas*

*Utilizar Multiplexores de 4 canales, teniendo presente que las entradas **A** y **B** son activas en **alto**. Las entradas **C** y **D** son activas en **bajo** y la salida **Z** es activa en **alto**.*

Con un software de simulación electrónica verificar el funcionamiento.

Como el circuito debe cumplir una de las 2 condiciones y no las 2, la función general **Z** cumple con la función Booleana de una compuerta lógica **XOR** de 2 entradas ya que la salida es activa en **alto**; si fuera activa en bajo sería la función de una compuerta lógica XNOR.

$$Z = C1 \oplus C2$$

$$C1 = Condición\ 1\ y\ C2 = Condición\ 2$$

Condición 1: las entradas A y B estén activadas con nivel alto y la función es:

$$C1 = A \times B$$

Condición 2: las entradas C y D o ambas estén activadas y se activan con nivel bajo y la función es:

$$C2 = \overline{C} + \overline{D}$$

Función General **Z**:

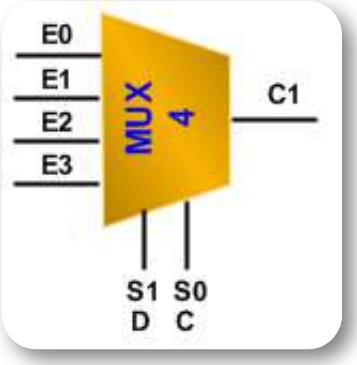
$$Z = (A \times B) \oplus (\overline{C} + \overline{D})$$

Generar la tabla de verdad: con 4 entradas A, B, C y D; y las salidas C1, C2 y Z:

ENTRADAS				SALIDAS		
D	C	B	A	C1	C2	Z
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	1	0	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	1	1	0

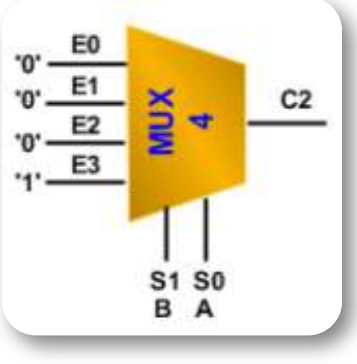
Circuito para la Condición 1:

$$C1 = A \times B$$



Circuito para la Condición 2:

$$C2 = \overline{C} + \overline{D}$$



Se debe agregar un Multiplexor para el manejo de las 2 condiciones de forma simultánea y cumplir así con la función **XOR** de la salida **Z**, debido a que la combinación A='1', B='1', C='1' y D='1' activa la salida 'Z' y se observa en la tabla que se debe **desactivar ya que cumple las 2 condiciones**.

El Multiplexor utilizado para la simulación es el 74LS153 de 4 canales y la salida activa en alto.

Repositorio en Github ("EA8-Mux_2condiciones.dsn") con el archivo de la simulación con Proteus v7.8

El diagrama en bloques del circuito completo es:

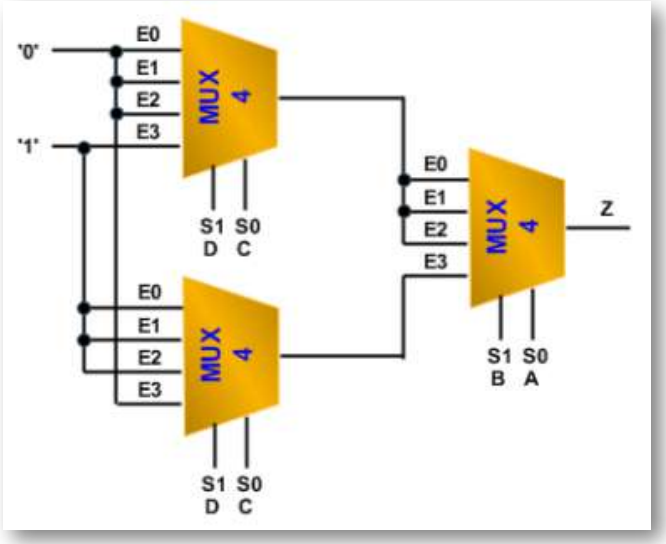
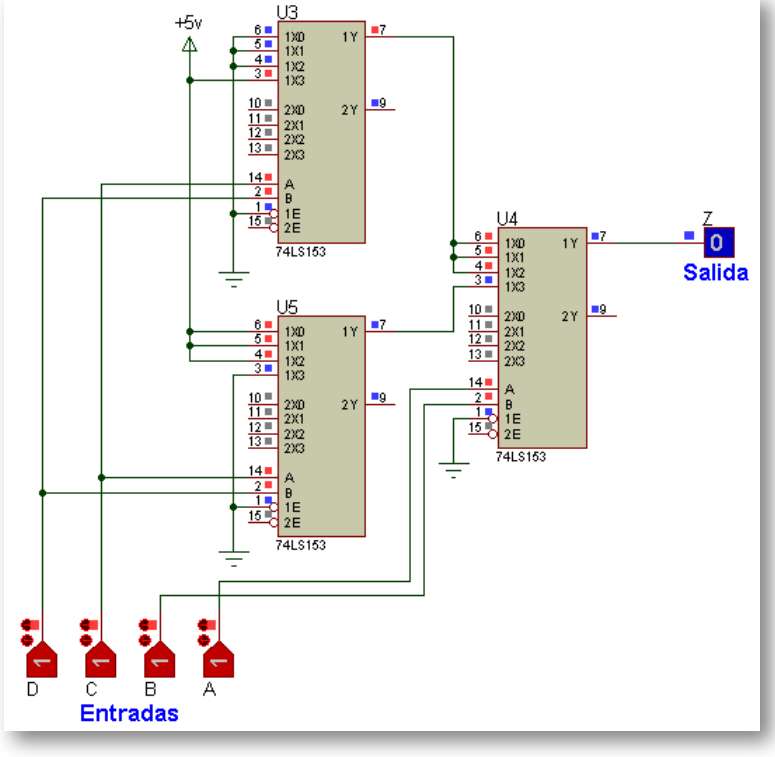


Imagen del circuito simulado en Proteus:



EA9-ALU y DEMUX

*Diseñar un circuito combinacional que partiendo de un Demultiplexor de 4 salidas se controlen las operaciones básicas de Suma, Resta, Multiplicación y División de 2x2bits. Cuando se seleccione una de las 4 operaciones se mostrará en un display de 7 segmentos de cátodo común la letra **A** para la Adición o Suma, la **S** para la Sustracción o Resta, la **P** para el Producto o Multiplicación y la **d** para la División. Las respuestas de las operaciones se visualizarán en LEDs.*

Sel B	Sel A	Operación	Display
0	0	Adición o Suma	A
0	1	Sustracción o Resta	S
1	0	Producto o Multiplicación	P
1	1	División	d

Códigos y caracteres para las operaciones

Diagrama en bloques:

- Bloque de **Funciones Lógicas** para el manejo del display de 7 segmentos de cátodo común.
- **Visualización** del tipo de operación en el **Display de Cátodo Común.**
- **Demultiplexor** de 4 canales activo en alto referencia 4555 (el 4556 es activo en bajo) que controlará cual operación se va a utilizar.
- Bloque **Habilitador** del bloque de circuitos lógicos.
- 4 Bloques con los **Circuitos Lógicos** de cada operación.
- **Visualización de las respuestas** de cada operación lógica binaria.

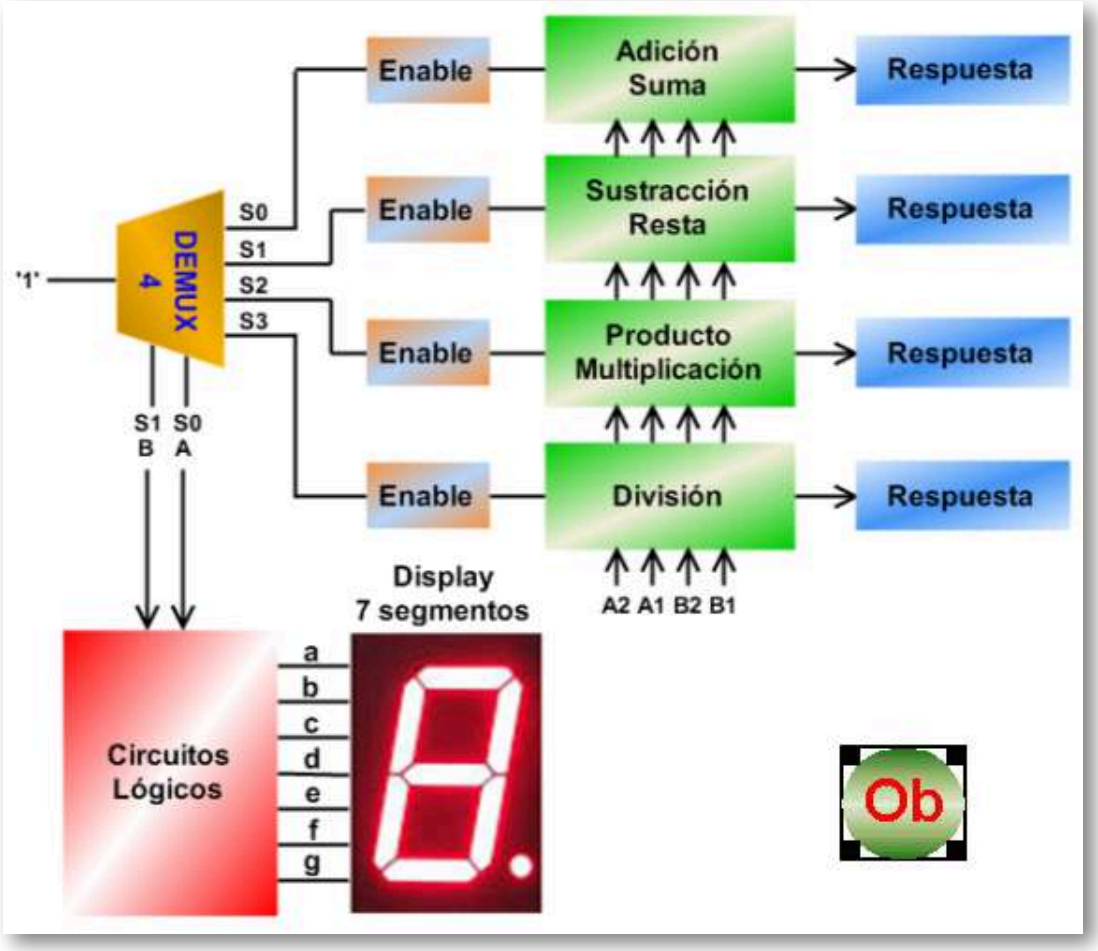
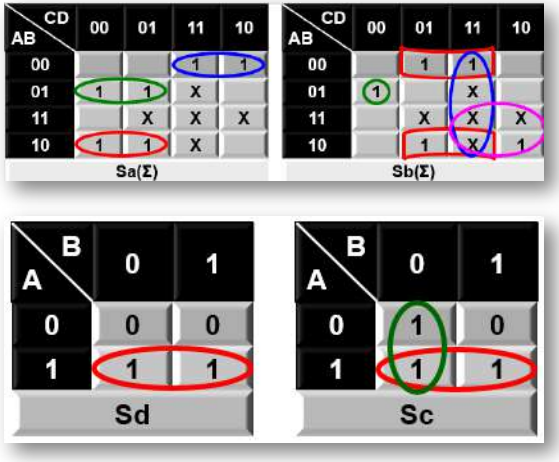


Tabla de verdad para el manejo del Display de 7 segmentos:

ENTRADAS			SALIDAS							
#	B	A	Dis	Sg	Sf	Se	Sd	Sc	Sb	Sa
0	0	0	A	1	1	1	0	1	1	1
1	0	1	S	1	1	0	1	1	0	1
2	1	0	P	1	1	1	0	0	1	1
3	1	1	d	1	0	1	1	1	1	0

Mapas de Karnaugh para la simplificación de las funciones lógicas para el control del Display:



Las funciones lógicas simplificadas por mapas de Karnaugh para encender cada segmento del Display:

$$Sa = Sf = \overline{A} + \overline{B}$$

$$Sb = Se = \overline{A} + B$$

$$Sc = A + \overline{B}$$

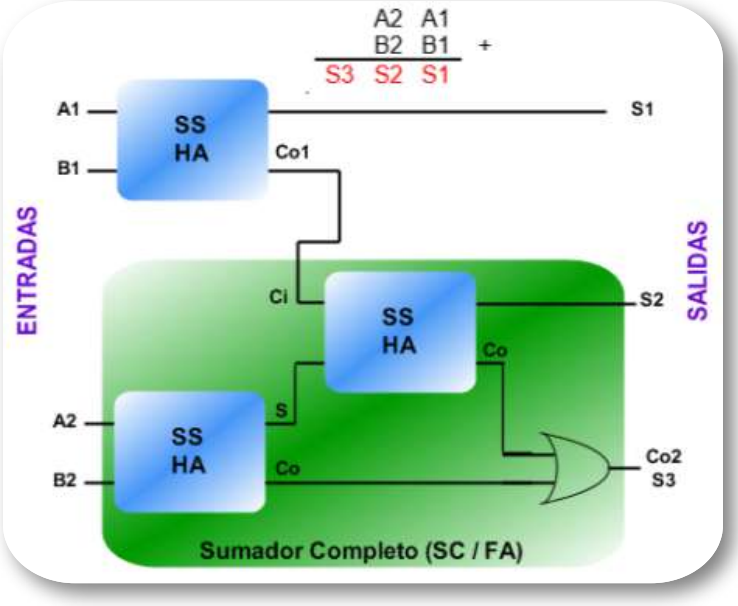
$$Sd = A$$

$$Sg = +V_{cc} = +5v$$

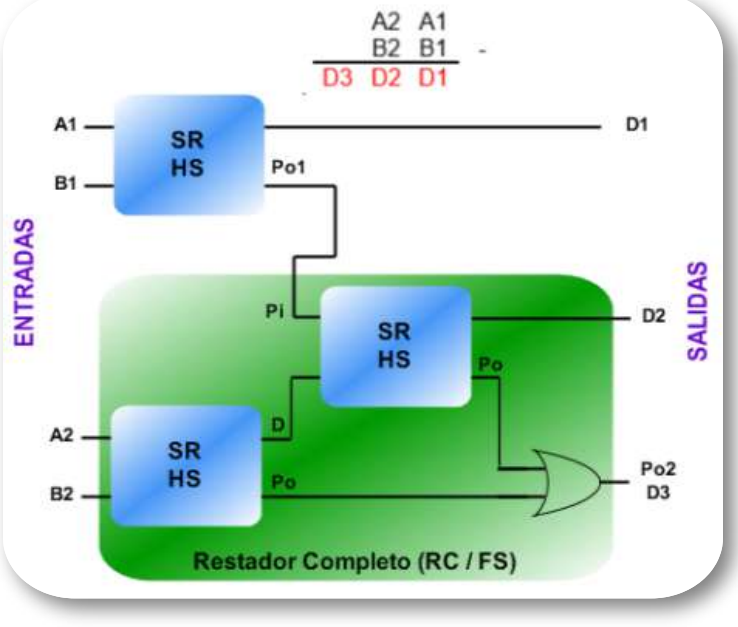
Para habilitar la entrada de los bit de datos **A** y **B** a los circuitos correspondientes para cada operación lógica binaria se utiliza una compuerta **YES o Buffer** con tecnología **Tri-State (tres estados)**, la referencia comercial es 74LS126 que tiene alta impedancia de salida (Hi-Z) y se activa con un nivel alto (el 74125 se activa con un nivel bajo).

- Las soluciones para las operaciones lógicas binarias de este ejercicio se trataron en la **Unidad 3.**

- Diagrama en bloques del Sumador 2x2:



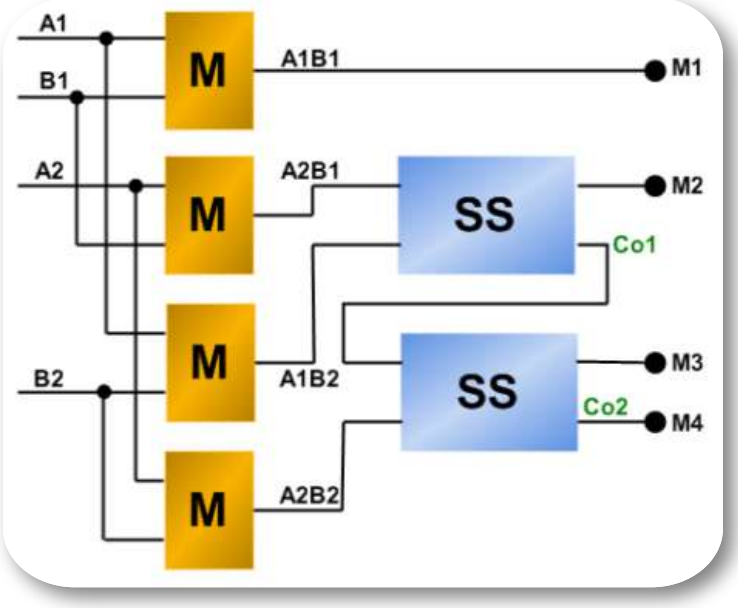
- Diagrama en bloques del Restador 2x2:



Estructura de la multiplicación 2x2:

		A2	A1	
		B2	B1	x
Co2	*Co1*	A2B1	A1B1	
	A2B2	A1B2		+
M4	M3	M2	M1	

Diagrama en bloques del Multiplicador 2x2:



Estructura de la división 2x2:

A2	A1	B2	B1
R2	R1	C2	C1

Diagrama en bloques del Divisor 2x2:

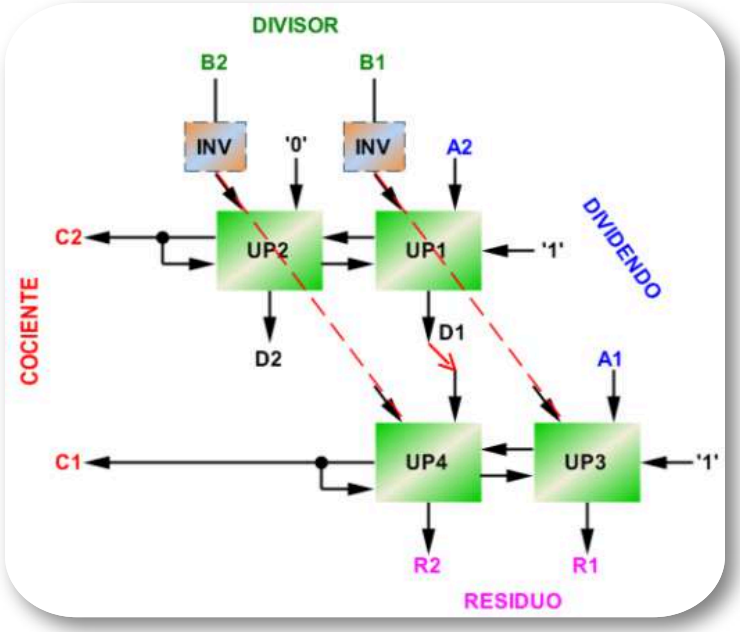
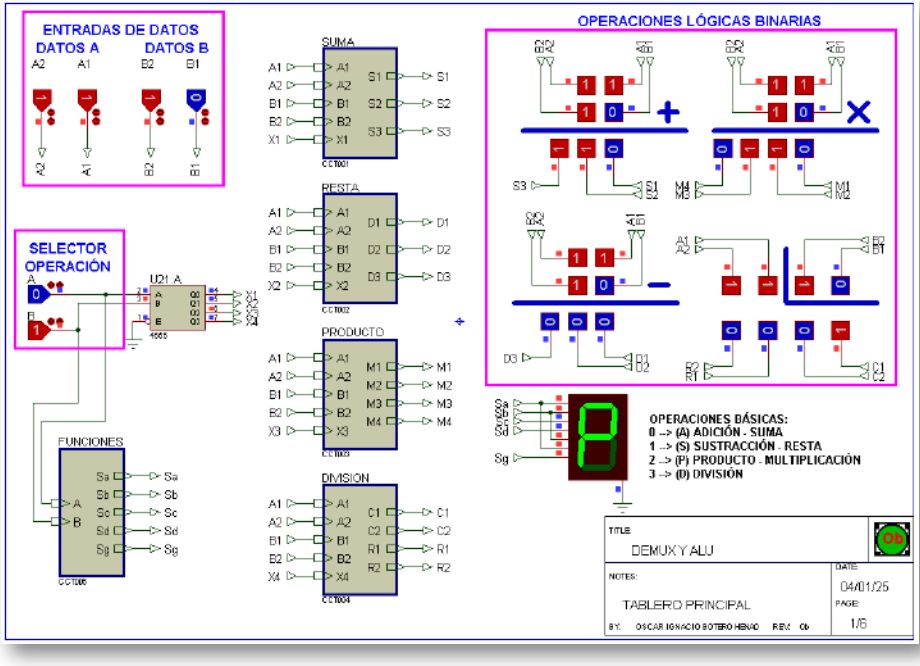


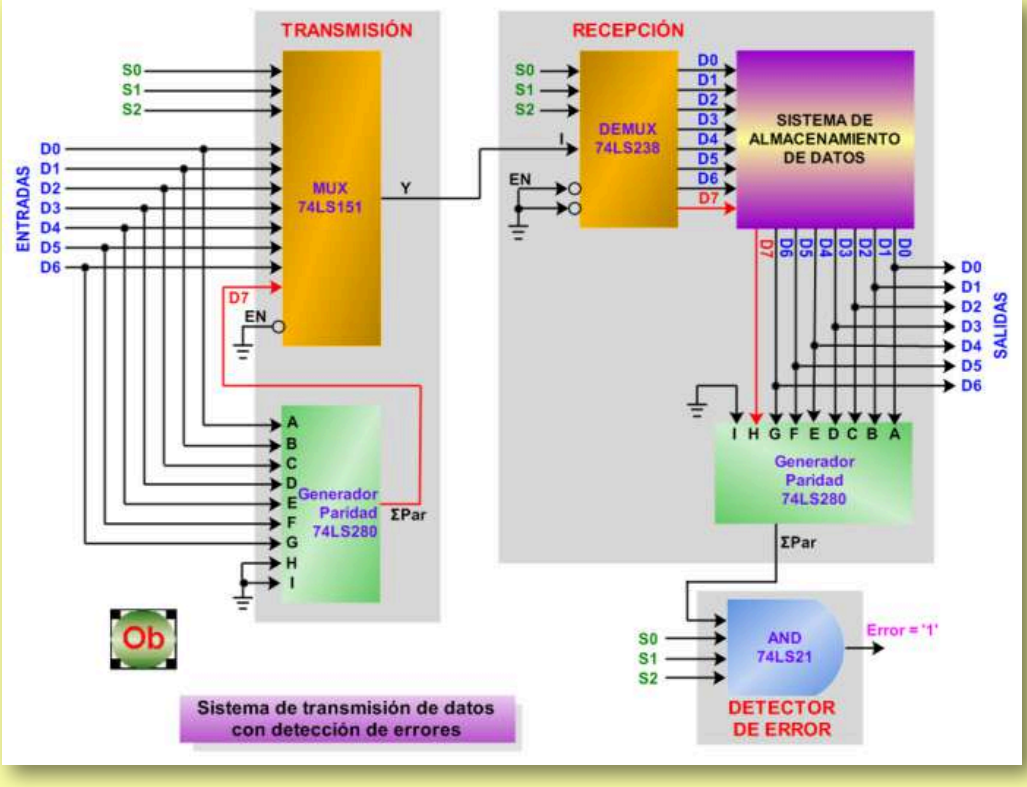
Imagen del panel principal de la simulación (5 páginas):



Repositorio en Github ("EA9-Alu_Demux.dsn") con el archivo de la simulación con Proteus v7.8 con 6 páginas

EA10- Generador de Paridad y Verificación

Sistema de Transmisión de Datos con Detección de Errores. Simular el funcionamiento con un software apropiado modificando el circuito para poder forzar errores en la recepción de un bit del paquete de datos y verificar la respuesta del sistema:



NOTA: **El sistema debe incluir un bloque para el Almacenamiento con el fin de guardar los datos hasta completar los 8 bits que constituyen el paquete de información. Dicho bloque se implementa con circuitos Digitales Secuenciales. Este tema NO está contemplado en este libro.**

- El circuito se compone de 4 bloques:
  - Transmisión de los datos
  - Recepción de los datos
  - Sistema para el almacenamiento del paquete de datos
  - Detector de Error

- Las **3 líneas de Selección (S0, S1 y S2)** se unifican para las entradas a los circuitos integrados: **MUX**, **DEMUX** y a la compuerta lógica **AND** de 4 entradas.

El bloque de ***Transmisión de Datos*** consta de:

- Un **MULTIPLEXOR** 74LS151 de 8 canales que recibe los 7 bits de datos (D0 a D6), más el **OCTAVO BIT** (D7) que proviene del **GENERADOR DE PARIDAD** 74LS280 quien tiene como función generar el **bit de paridad PAR** que entrega por la salida **ΣPar.**
- Los datos suministrados por la salida del MUX se hacen de forma SERIAL con cada valor en las líneas del Selector.
- La salida **ΣPar** genera un valor de **'0'** si la cantidad de '1s' en el paquete de datos es **PAR** y un valor de **'1'** si la cantidad de '1s' en el paquete de datos es **IMPAR**.
- De las 2 salidas que tiene el MUX, se utiliza la NO INVERTIDA (Y).

El bloque de ***Recepción de Datos*** consta de:

- Un **DEMULTIPLEXOR** 74LS238 de 1:8 canales que recibe del **MUX** cada dato del paquete transmitido por (Y); las respectivas salidas del **DEMUX** entregan la información al bloque de **Almacenamiento** que guarda los datos hasta tener disponibles los 8 bits del paquete completo.
- Los datos completos del paquete se llevan a otro circuito integrado **GENERADOR DE PARIDAD** que cumple la función de Comprobar el valor del **bit de Paridad (D7)** que llegó y enviarlo al bloque **Detector de Error**.

El bloque ***Detector de Error*** consta de:

- Una compuerta lógica **AND** 7421 de 4 entradas que recibe los datos de las 3 líneas de Selección más el bit de **Comprobación de Paridad.**
- Cuando las líneas de Selección están en **S0='1', S1='1' y S2='1'** es porque ya se ha **transmitido y recibido todo el paquete de datos** y si el bit de Comprobación de Paridad recibido es **'1'** la salida de la compuerta lógica se activa informando que **HAY ERROR DE PARIDAD PAR** en el paquete de datos **RECIBIDOS.**

El sistema solo detecta si hay un número impar de errores en los datos del paquete recibido, es decir, si en la transmisión hay 2 errores, el receptor no lo detectaría.

- Se puede implementar el sistema con un **DEMULTIPLEXOR** 74LS138, para ello se usa la salida $\overline{Y}$ del MUX ya que el DEMUX invierte sus respectivas salidas (postulado del álgebra Booleana de la doble negación $\overline{\overline{A}} = A$).

Imagen de la simulación con el paquete **CORRECTO**:

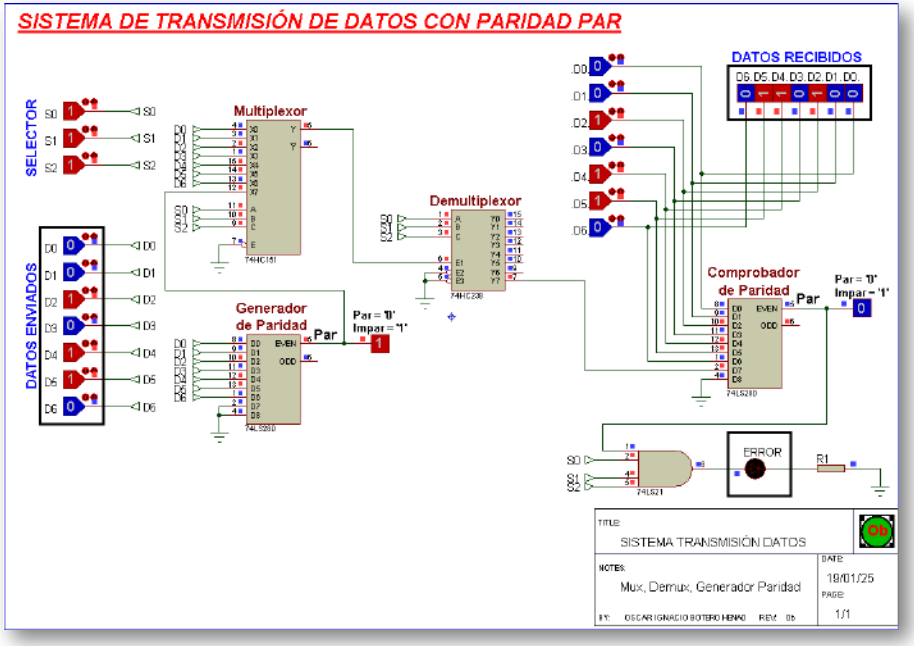
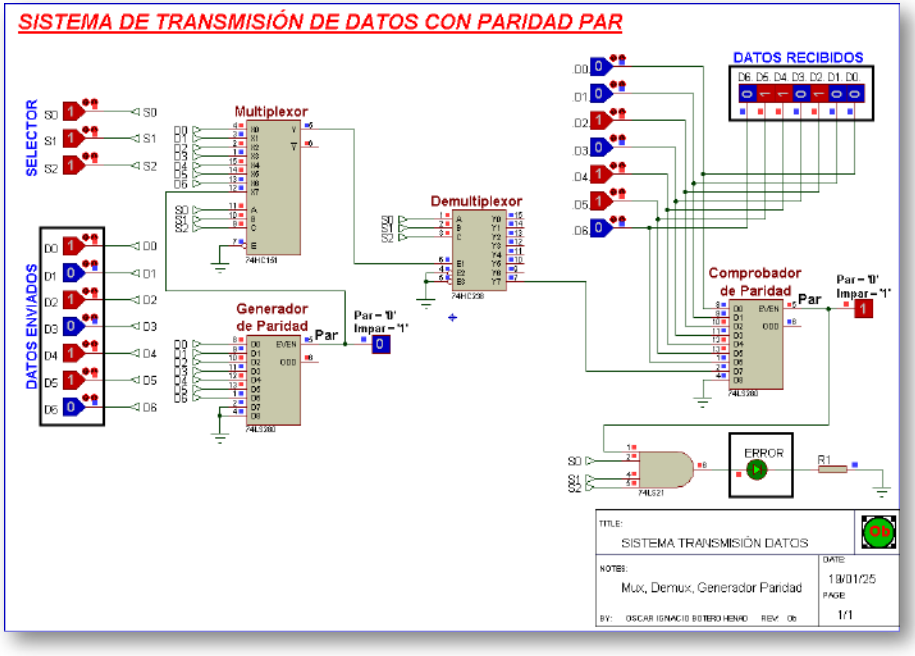


Imagen de la simulación con el paquete **INCORRECTO**:

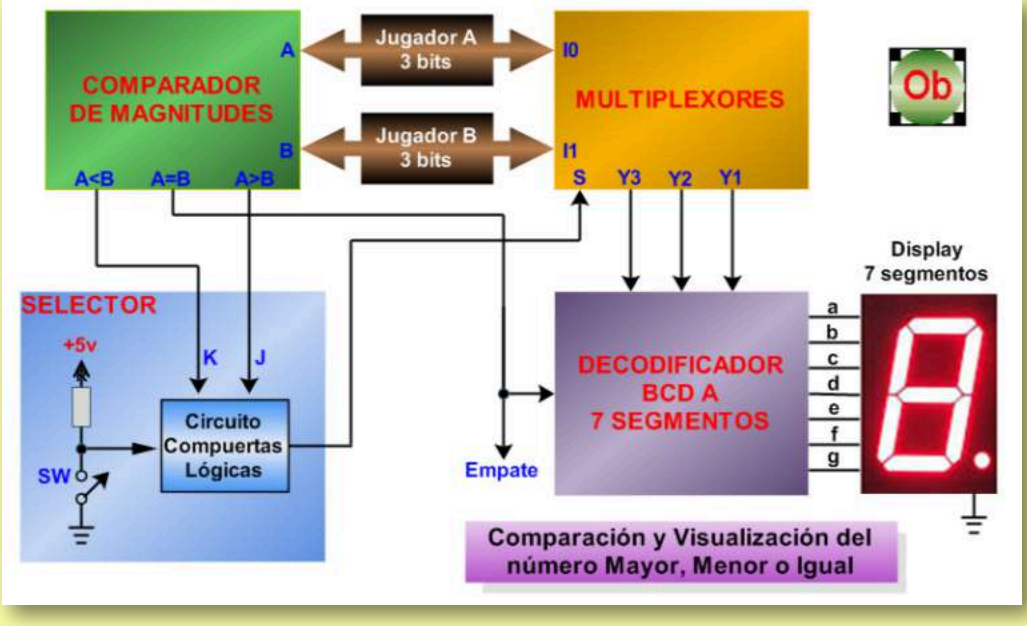


Repositorio en [Github ("EA10-Paridad y Verificación 238.dsn")] con el archivo de la simulación con Proteus v7.8

EA11- Comparador Jugadores

*Sistema Combinacional, para visualizar en un display de 7 segmentos el número **Mayor** o el número **Menor** entre 2 números ingresados por los usuarios en código Binario entre el 0_d y el 7_d, por medio de un interruptor se elige el número a visualizar en un display de 7 segmentos, en caso que se presente **Empate**, el display se apaga y se enciende un LED.*

Diagrama en bloques



Simular con un software electrónico y verificar su funcionamiento.

- Cada jugador tiene 3 bits para ingresar en código binario el número entre el 0_d y el 7_d.

- El sistema se compone de 4 bloques:
  - Comparador de magnitudes
  - Selector del número a visualizar
  - Multiplexores
  - Decodificador BCD-7 segmentos, Display de 7 segmentos y Led

- Los datos ingresados por los jugadores se llevan simultáneamente a los bloques del **Comparador** y de los **Multiplexores**.

El bloque ***Comparador de Magnitudes*** consta de:

- Un circuito integrado **COMPARADOR DE MAGNITUD** 74LS85 de 4 bits que recibe los números de los 2 jugadores denominados A y B y activa una de las 3 salidas: $A < B, A = B, A > B$.

El bloque ***Selector*** consta de:

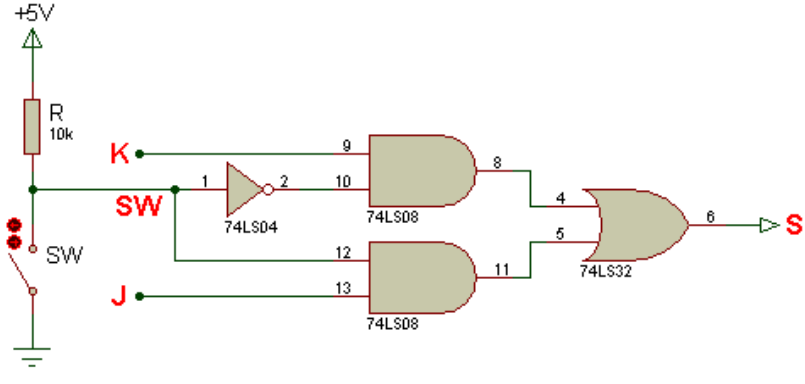
- Un **INTERRUPTOR** que sirve para elegir el número a visualizar en el display entre el **MAYOR** y el **MENOR** que provienen del bloque **Comparador**, la señal de salida de este bloque sirve como la entrada de **Selector (S)** para el bloque de los **Multiplexores**. Con el interruptor **abierto ('1')** elige la salida del número **Menor** y con el interruptor **cerrado ('0')** elige la salida del número **Mayor**.

- Para el **Circuito de las Compuertas Lógicas**:
  - Se llama ***J*** a la salida del comparador $A > B$ y ***K*** a la salida del comparador $A < B$
  - Se requiere activar la salida cuando:
    - $J = 1$ y el $SW = 1$ se implementa con una compuerta **AND**

      $$J \times SW$$

    - $K = 1$ y el $SW = 0$ se implementa con una compuerta **AND**

      $$K \times \overline{SW}$$



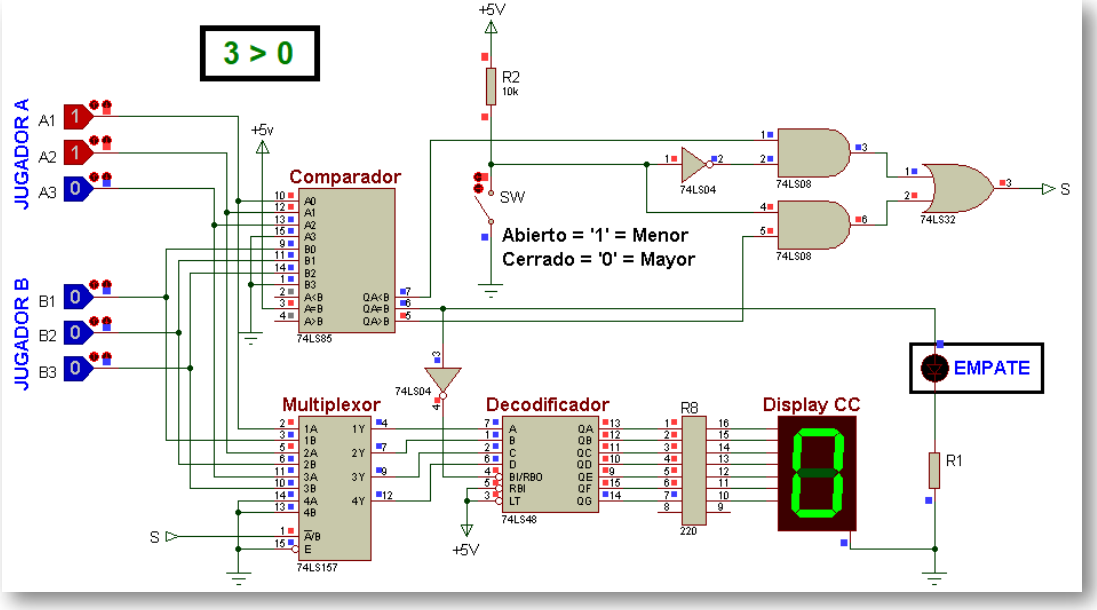
El bloque de los ***Multiplexores*** consta de:

- Un circuito integrado de **4 MULTIPLEXORES** de 2 canales cada uno, referencia 74LS157 de 4 bits que recibe los datos ingresados por los 2 jugadores y cuando el **Selector** $\boldsymbol{S = 0}$ se transfiere a la salida correspondiente el número del **Jugador A** y con $\boldsymbol{S = 1}$ se transfiere el número del **Jugador B**.

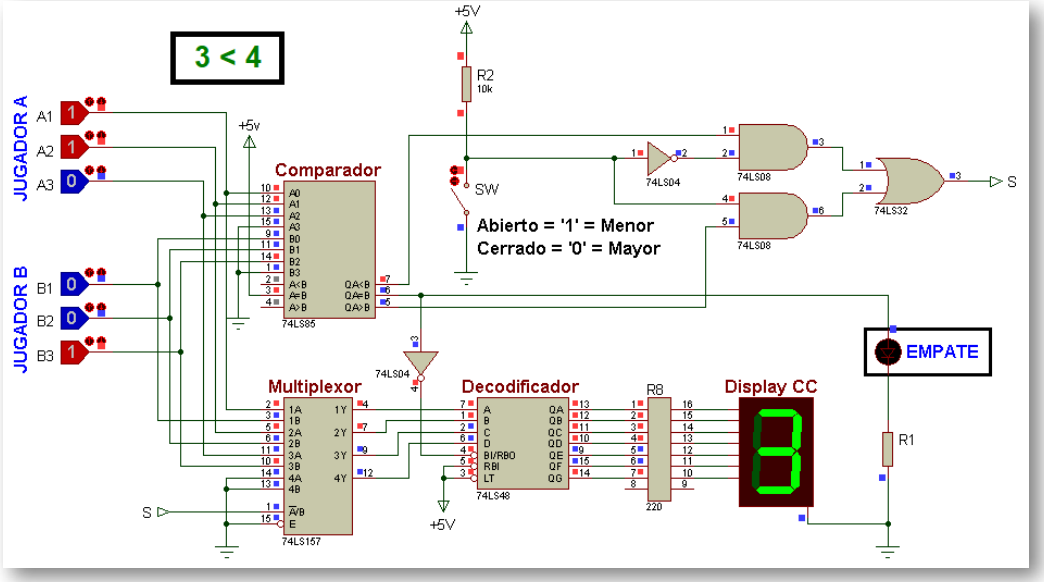
El bloque del ***DECODIFICADOR BCD A 7 SEGMENTOS*** consta de:

- Un circuito integrado 7448 que decodifica las señales provenientes del bloque de los **Multiplexores** y los convierte en 7 señales que se conectan al **Display de Cátodo Común** para la visualización del número correspondiente.
- En la tabla de verdad del circuito integrado, se observa que si la terminal BI/RBO se coloca en **Bajo**, todas las señales de las salidas se ponen en '0' y el Display se **Apaga**. Esta condición se utiliza para el caso en que los 2 jugadores ingresan el mismo número (Empate).

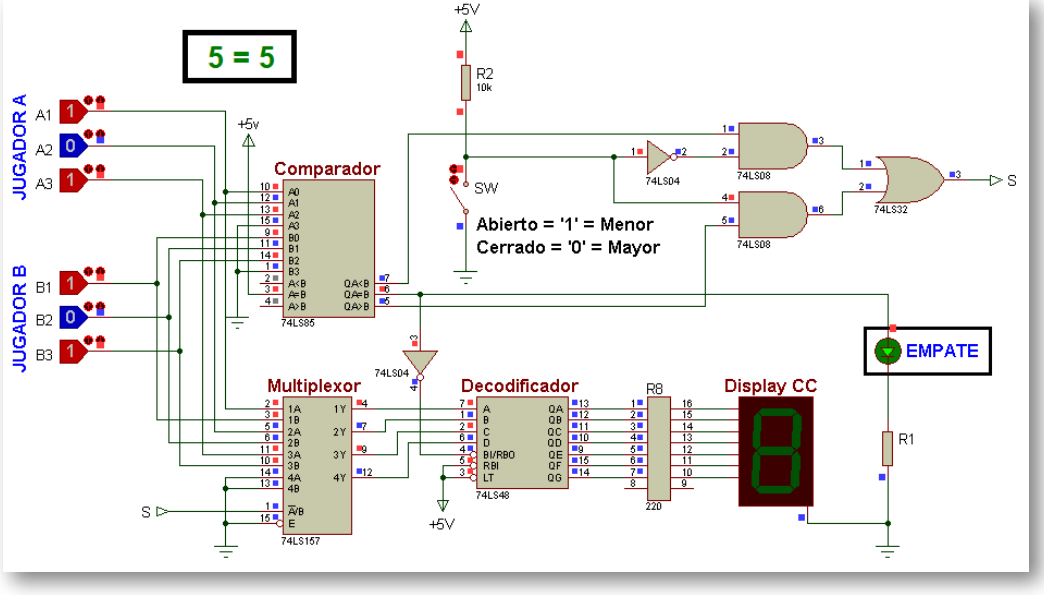
Caso 1: imagen de la simulación cuando $\boldsymbol{A > B}$ y $\boldsymbol{SW = 1}$, se visualiza el número **Menor**:



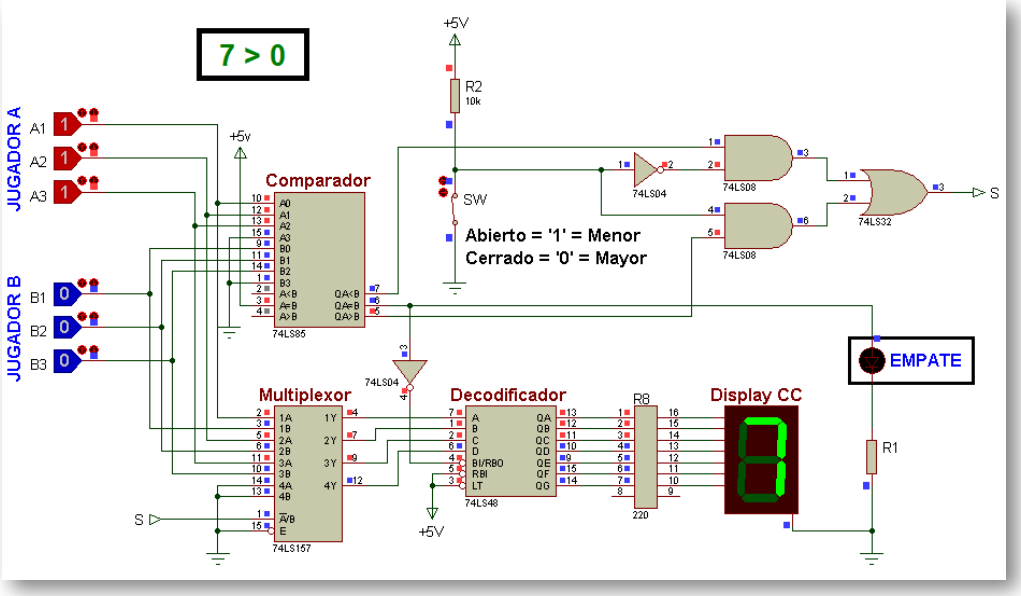
Caso 2: imagen de la simulación cuando $\boldsymbol{A < B}$ y $\boldsymbol{SW = 1}$, se visualiza el número **Menor**:



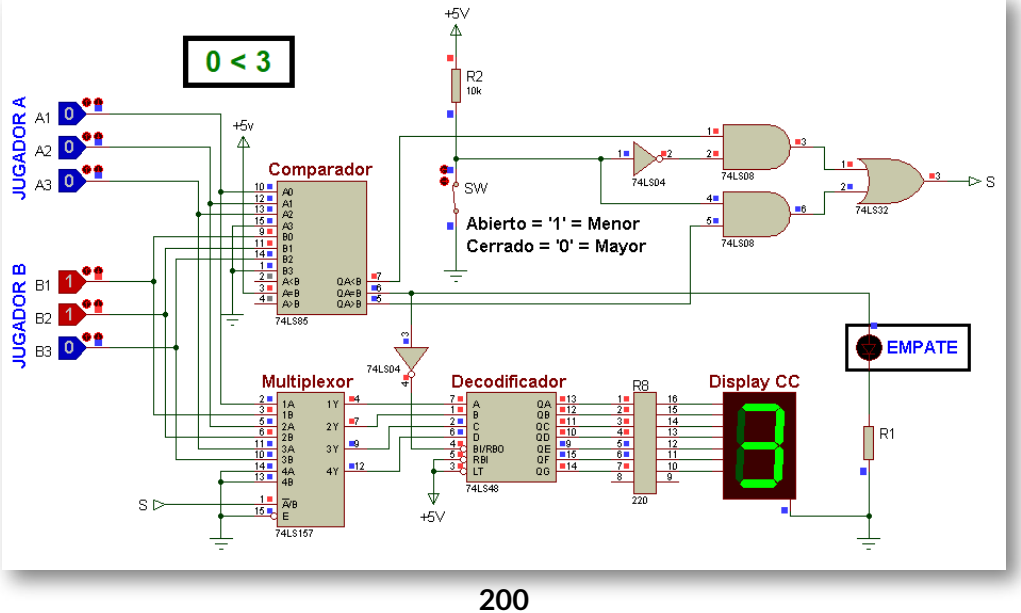
Caso 3: imagen de la simulación cuando se presenta **Empate,** $\boldsymbol{A = B}$, el **display se apaga** y el **LED** se enciende:



Caso 4: imagen de la simulación cuando $A > B$ y $SW = 0$, se visualiza el número **Mayor**:



Caso 5: imagen de la simulación cuando $A < B$ y $SW = 0$, se visualiza el número **Mayor**:



Repositorio en Github ("EA11-Comparador Jugadores.dsn") con el archivo de la simulación con Proteus v7.8

EA12- Marquesina

Diseñar un sistema combinacional que controle una MARQUESINA para una terraza, cuya función es brindar sombra, proteger de la lluvia y el viento y crear condiciones óptimas para el riego de un cultivo.

*Nomenclatura para el control de la **Marquesina**:*

- *S: Sol, si hace sol S='1'*
- *L: Lluvia, si llueve L='1'*
- *V: Viento, si hay viento V='1'*
- *F: Frío, si hace frío F='1'*
- *M: Marquesina, extendida M='1', contraída M='0'*

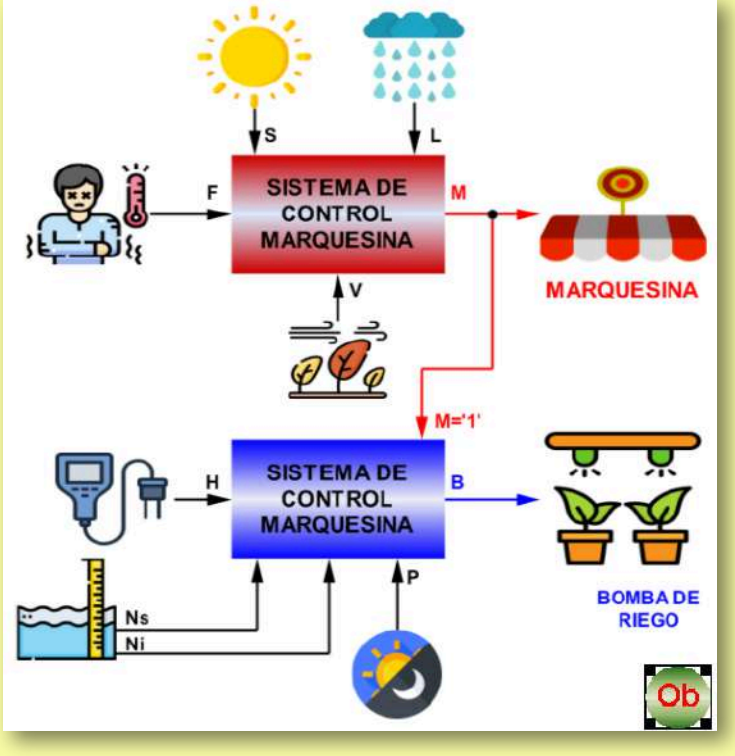
*Condiciones para el funcionamiento de la **Marquesina**:*

1. *Independiente del estado de los demás sensores, siempre que llueva se debe de extender, para evitar que se inunde la terraza. No se considerará posible que simultáneamente llueva y haga sol (se asigna '0' a la salida).*
2. *Si hace mucho viento se debe extender para evitar daños en el cultivo; pero hay una excepción: aun cuando haya viento, si el día está soleado y hace frío en la terraza, se recogerá la marquesina para climatizar el espacio.*
3. *Si no llueve ni hace viento, sólo se extenderá la marquesina en los días de sol y cuando haga calor en la terraza, para evitar el recalentamiento del cultivo y la evaporación rápida del agua suministrada por el riego.*

El sistema se complementa con el diseño de un circuito combinacional que controle el riego de un cultivo en la terraza.

Nomenclatura para el control de la Bomba de Riego:

- *Ni: sensor del Nivel Inferior del depósito de agua, se activa con '1'*
- *Ns: sensor del Nivel Superior del depósito de agua, se activa con '1'*
- *H: sensor de Humedad, tierra húmeda H='0', tierra seca H='1'*
- *P: Fotocelda, de día P='1', de noche P='0'*
- *B: Bomba de Riego, se activa con '1'*



La Bomba de riego se activa bajo las siguientes condiciones:

*Se activa solamente cuando la **Marquesina** está extendida y cuando la tierra esté seca, pero antes debe comprobar que:*

1. *No se activa la Bomba si el dispensador de agua está vacío o por debajo del nivel del Sensor Inferior (Ni='0'), con el fin de evitar que se dañe por funcionar en vacío.*
2. *No se puede presentar que esté activo el sensor **Ns** y desactivado el sensor **Ni**, para estas opciones se asigna la salida como **B='0'**.*
3. *La **Fotocelda (P)** detecta si está de día ('1') o de noche ('0'). La bomba se activa si está de día y el sensor Ni está activado.*
4. *Se activa la bomba si está de noche y están activados los 2 sensores de nivel.*

*Realizar las tablas de verdad para el control de la **Marquesina (M)** y de la **Bomba de Riego (B)** y hallar sus respectivas funciones lógicas simplificadas. Implementar los circuitos en un simulador electrónico y verificar su correcto funcionamiento.*

*En un display de 7 segmentos de cátodo común se debe visualizar el estado de la Marquesina, la letra **E** para **Extendida** y la letra **C** para **Contraída**.*

*Igualmente, implementar los controles de la **Marquesina** y de la **Bomba de Riego** para el cultivo con circuitos integrados **DECODIFICADORES** de 4 a 16.*

Diseñar una variante del circuito de la Marquesina, que mediante un interruptor se seleccione que funcione de forma automática o manual. De forma manual funciona con 2 pulsadores, uno para Expandir y el otro para Contraer la Marquesina, el circuito deberá retener la última orden dada.

Se comienza solucionando el circuito de control para la **MARQUESINA** de la terraza:

- Se elabora la tabla de verdad del sistema de control de la **Marquesina (M)** teniendo presente las condiciones en que se debe Extender (M='1') o Contraer (M='0') para proteger el cultivo.

ENTRADAS				SALIDA	
L	V	S	F	M	Cond
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	❸
0	0	1	1	0	
0	1	0	0	1	❷
0	1	0	1	1	❷
0	1	1	0	1	❷
0	1	1	1	0	
1	0	0	0	1	
1	0	0	1	1	
1	0	1	0	0	❶
1	0	1	1	0	❶
1	1	0	0	1	
1	1	0	1	1	
1	1	1	0	0	❶
1	1	1	1	0	❶y❷

- De la tabla se extraen las salidas donde M='1' y se llevan a un mapa de Karnaugh de 4 variables, este mapa se desarrolla por el método de Minterm (Σ).

FS \ VL	00	01	11	10
00		1	1	1
01	1			1
11				
10		1	1	1

M

Se simplifica el mapa de Karnaugh, obteniendo la función lógica simplificada:

$$\boldsymbol{M(\Sigma) = (\overline{S} \times L) + (\overline{S} \times V) + (\overline{F} \times S \times \overline{L})}$$

$$M(\Sigma) = \overline{S}(L + V) + (\overline{F} \times S \times \overline{L})$$

Se diseña la tabla de verdad para los segmentos del **DISPLAY** con los 2 caracteres $E = Extendida$ y $C = Contraída$:

MARQUESINA		SEGMENTOS						
M	Display	a	b	c	d	e	f	g
0	C	1	0	0	1	1	1	0
1	E	1	0	0	1	1	1	1

Se obtienen las funciones lógicas correspondientes:

$$\boldsymbol{Segmentos: a = d = e = f = +Vcc = +5v}$$

$$\boldsymbol{Segmentos: b = c = GND}$$

$$\boldsymbol{Segmento: g = M}$$

Imagen de la simulación con la **Marquesina Extendida**:

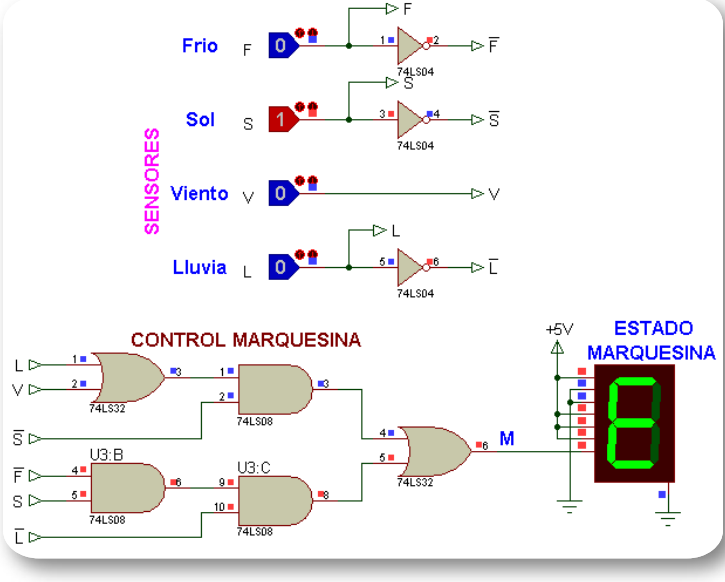
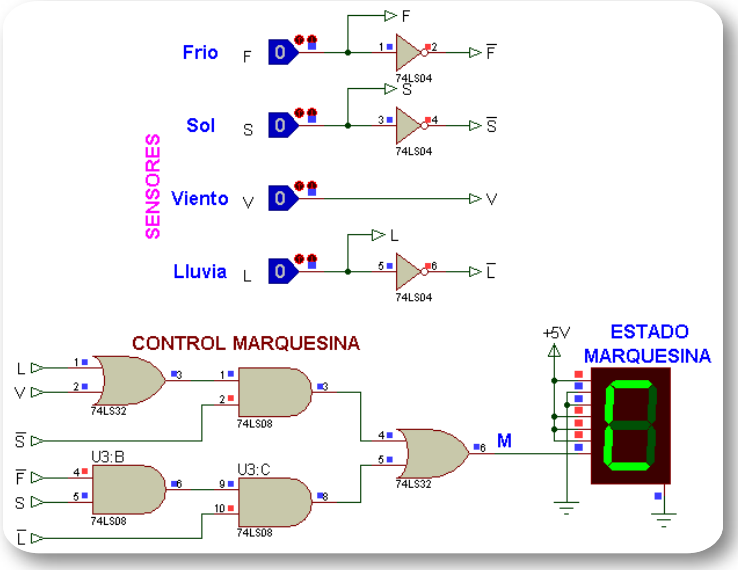


Imagen de la simulación con la **Marquesina Contraída**:



Ahora, se soluciona el circuito de control para la **BOMBA DE RIEGO** del agua para el cultivo:

ENTRADAS				SALIDA	
H	Ns	Ni	P	B	Cond
0	0	0	0	0	❶
0	0	0	1	0	❶
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	❶
0	1	0	1	0	❶
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	❸
1	1	0	0	0	❷
1	1	0	1	0	❷
1	1	1	0	1	❹
1	1	1	1	1	❸

- Se elabora la tabla de verdad del sistema de control de la **BOMBA DE RIEGO (B)** teniendo presente las condiciones del enunciado.
- De la tabla se extraen las salidas donde $B = 1$ y se llevan a un mapa de Karnaugh de 4 variables, este mapa se desarrolla por el método de Minterm (Σ).
- Se simplifica el mapa de Karnaugh, obteniendo la función lógica simplificada,
- TENER PRESENTE que **solo se activa el Riego cuando la Marquesina esté Extendida**.

PNi \ NsH	00	01	11	10
00				
01			1	
11		1	1	
10				

B

$$B(\Sigma) = M[(P \times Ni \times H) + (Ni \times Ns \times H)]$$

$$B(\Sigma) = M[Ni \cdot H(P + Ns)]$$

Imagen del sistema cumpliendo una condición de los sensores de **Riego** y **LA MARQUESINA CONTRAÍDA:**

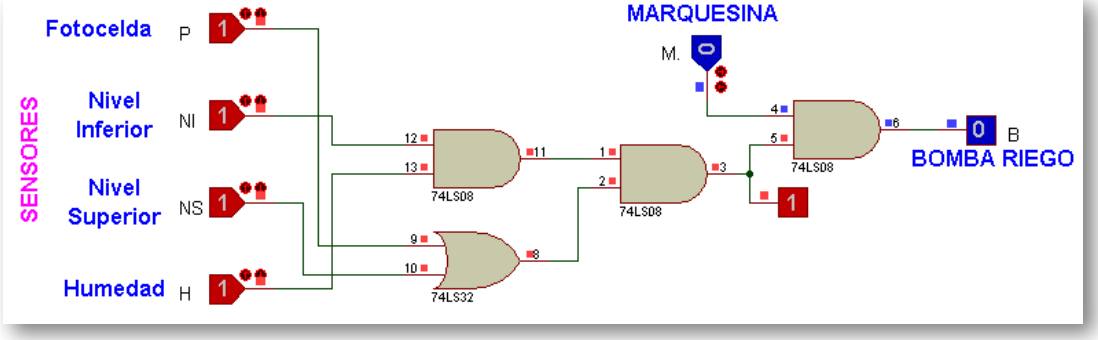
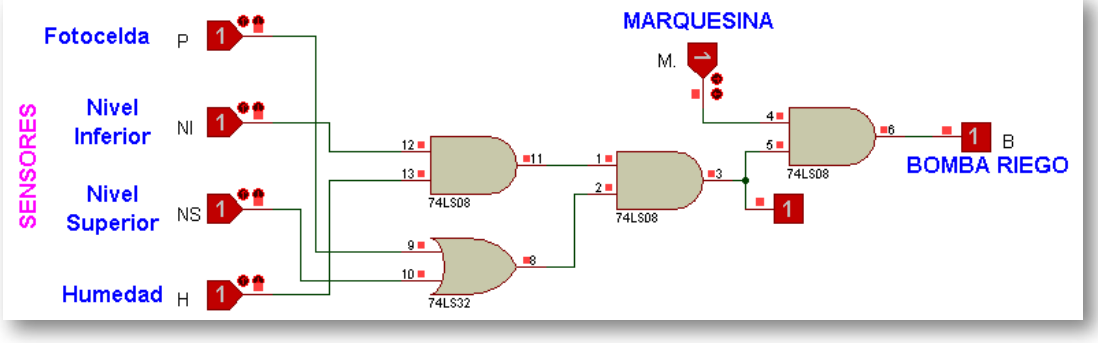
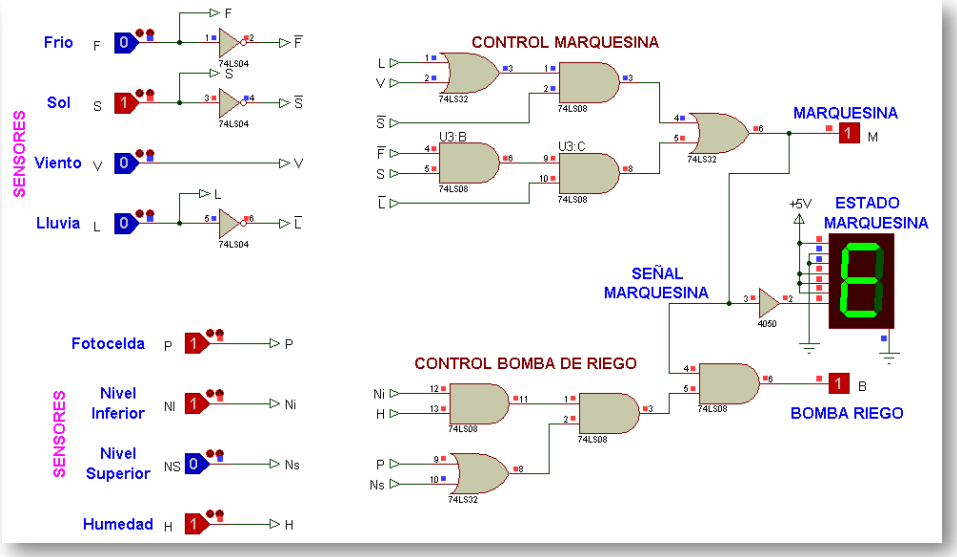
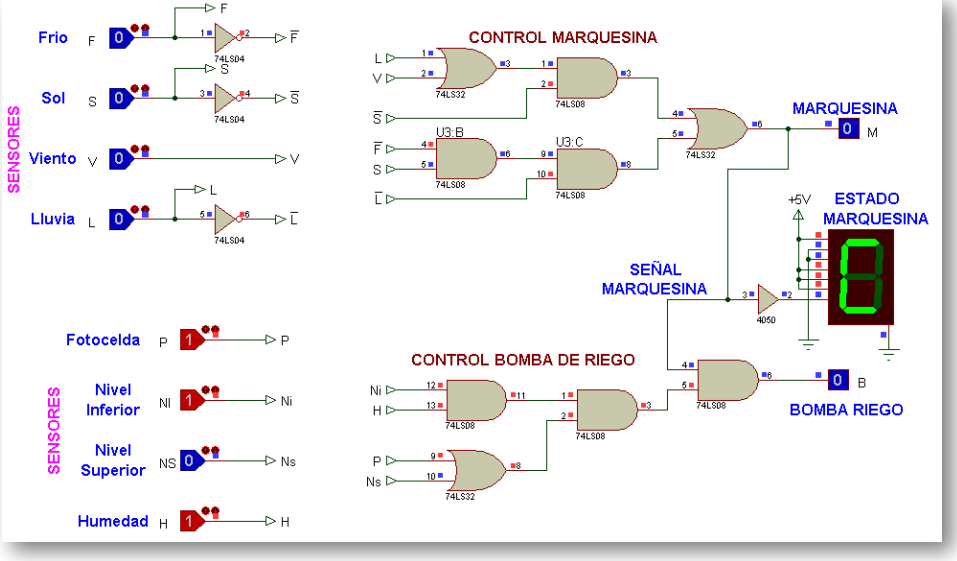


Imagen del sistema cumpliendo una condición de los sensores de **Riego** y **LA MARQUESINA EXTENDIDA:**



Imágenes del sistema completo, **Marquesina** y **Bomba de Riego**:



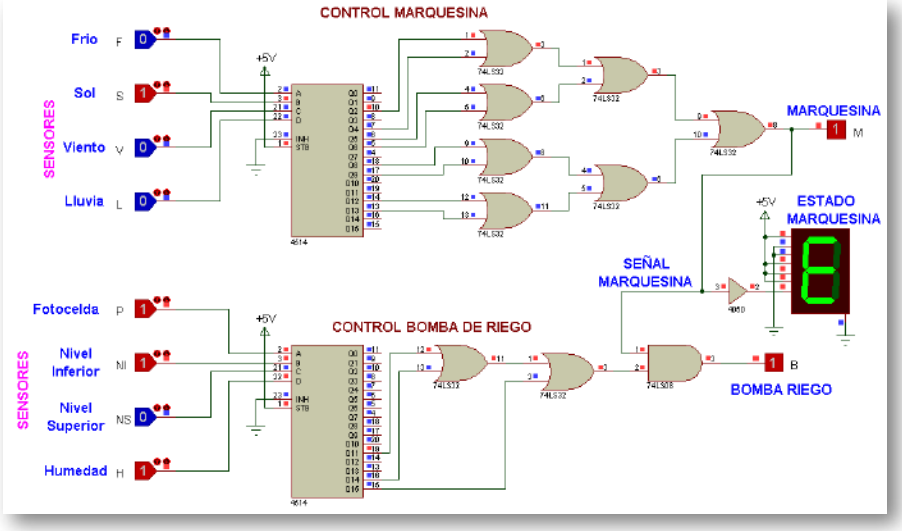


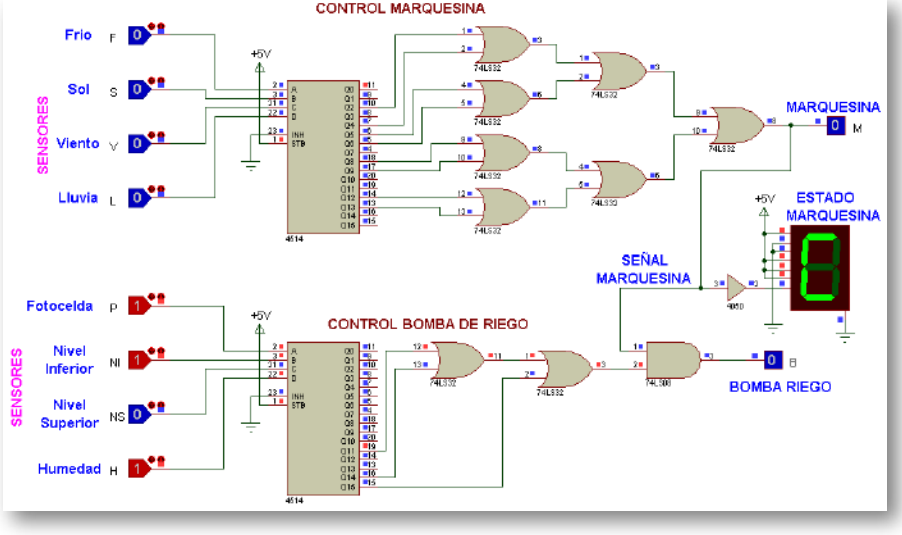
Repositorio en [Github ("EA12-Marquesina.dsn", "EA12-Marquesina-Decodificador.dsn" y "EA12-Marquesina-Deco-variante.dsn")] con el archivo de la simulación con Proteus v7.8

Para la implementación de los sistemas de control con los **DECODIFICADORES** de 4 a 16 se utilizan los circuitos integrados CMOS referencia 4514 donde las salidas son activas en ALTO y se aplica a los datos consignados en las tablas de verdad:

$$M(\Sigma) = (m2, m4, m5, m6, m8, m9, m12, m13)$$

$$B(\Sigma) = (m11, m14, m15)$$





La variante del circuito (Automático / Manual), se implementa con un **LATCH** y un **MUX**:

- Por medio de 2 pulsadores el **LATCH Extiende** o **Contrae** de forma **MANUAL** la Marquesina.
- El **Multiplexor** 74157 se encarga de controlar si trabaja de forma **AUTOMÁTICA** o **MANUAL** la Marquesina. Con el Interruptor **AUTO/MANUAL** en '1' se trabaja Automáticamente y en '0' Manualmente.

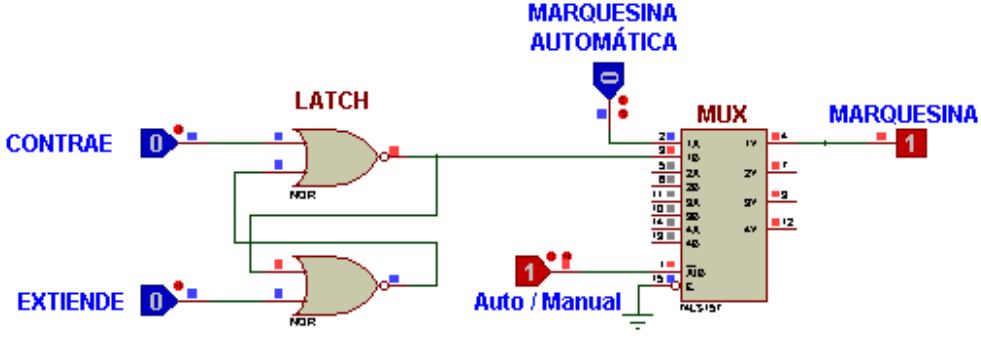
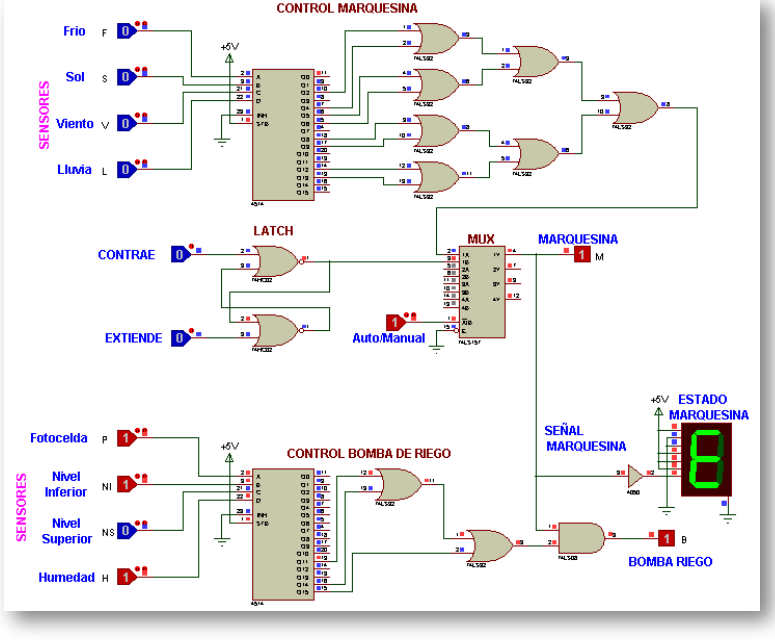


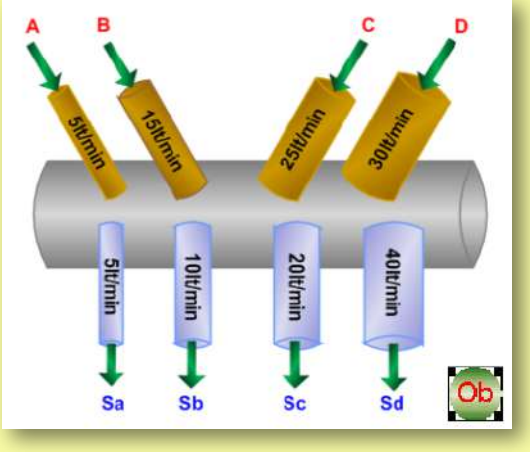
Imagen del Sistema Completo, incluída la variante:



EA13- Sistema de Tuberías

Se tiene un conjunto de tuberías, 4 de entrada y 4 de salidas. Todas las tuberías manejarán 2 estados: ***Abierta = '1'*** *y* ***Cerrada = '0'.***

- *Las tuberías de entrada poseen 4 sensores que indican por cuál tubería está circulando agua. Los caudales de las tuberías de entrada son:*
  - *La A = 5 litros/minuto*
  - *La B = 15 litros/minuto*
  - *La C = 25 litros/minuto*
  - *La D = 30 litros/minuto*



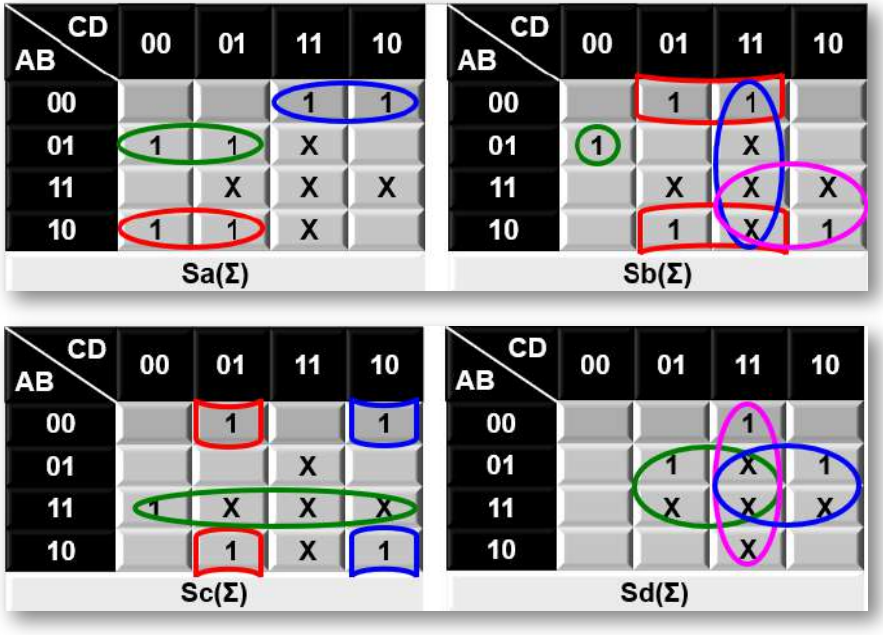
- *Las tuberías de salida son Sa, Sb, Sc y Sd y sus capacidades respectivas son:*
  - *La Sa = 5 litros/minuto*
  - *La Sb = 10 litros/minuto*
  - *La Sc = 20 litros/minuto*
  - *La Sd = 40 litros/minuto*

- *Teniendo en cuenta que sólo puede circular agua en dos tuberías de entrada de forma simultánea y activar las válvulas de las tuberías de salida necesarias para que salga tanto caudal de agua como el que entra.*
- *Para las combinaciones de las tuberías de entrada que **NO se permiten** se tomarán las salidas como estados de **Indiferencia = X** (don't care)*
- *Diseñar el sistema combinacional para las 4 válvulas de salida, halle la tabla de verdad e implemente el circuito en un software de simulación y verifique el funcionamiento.*
- *Implementar el circuito de control de las válvulas de las tuberías empleando decodificadores de 3 a 8.*

Se diseña la tabla de verdad con los sensores de las tuberías de entrada y salidas y sus respectivos caudales totales:

ENTRADAS					SALIDAS				
D	C	B	A	Tot	Sd	Sc	Sb	Sa	Tot
30	25	15	5	Ent	40	20	10	5	Sal
0	0	0	0	0	0	0	0	0	0
0	0	0	1	5	0	0	0	1	5
0	0	1	0	15	0	0	1	1	15
0	0	1	1	20	0	1	0	0	20
0	1	0	0	25	0	1	0	1	25
0	1	0	1	30	0	1	1	0	30
0	1	1	0	40	1	0	0	0	40
0	1	1	1	-	X	X	X	X	-
1	0	0	0	30	0	1	1	0	30
1	0	0	1	35	0	1	1	1	35
1	0	1	0	45	1	0	0	1	45
1	0	1	1	-	X	X	X	X	-
1	1	0	0	55	1	0	1	1	55
1	1	0	1	-	X	X	X	X	-
1	1	1	0	-	X	X	X	X	-
1	1	1	1	-	X	X	X	X	-

Se bajan los datos a cada mapa de Karnaugh:



Las funciones lógicas simplificadas de cada mapa de Karnaugh son:

$$Sa(\Sigma) = (\overline{A} \times \overline{B} \times C) + (\overline{A} \times B \times \overline{C}) + (A \times \overline{B} \times \overline{C})$$

$$Sa(\Sigma) = (\overline{A} \times \overline{B} \times C) + \overline{C} \cdot (A \oplus B)$$

$$Sb(\Sigma) = (\overline{B} \times D) + (C \times D) + (A \times C) + (\overline{A} \times B \times \overline{C} \times \overline{D})$$

$$Sb(\Sigma) = (\overline{B} \times D) + C \cdot (A + D) + (\overline{A} \times B \times \overline{C} \times \overline{D})$$

$$Sc(\Sigma) = (\overline{B} \times \overline{C} \times D) + (\overline{B} \times C \times \overline{D}) + (A \times B)$$

$$Sc(\Sigma) = \overline{B} \cdot (C \oplus D) + (A \times B)$$

$$Sd(\Sigma) = (B \times D) + (B \times C) + (C \times D)$$

$$Sd(\Sigma) = B \cdot (C + D) + (C \times D)$$

Imagen de la simulación del sistema de Tuberías:

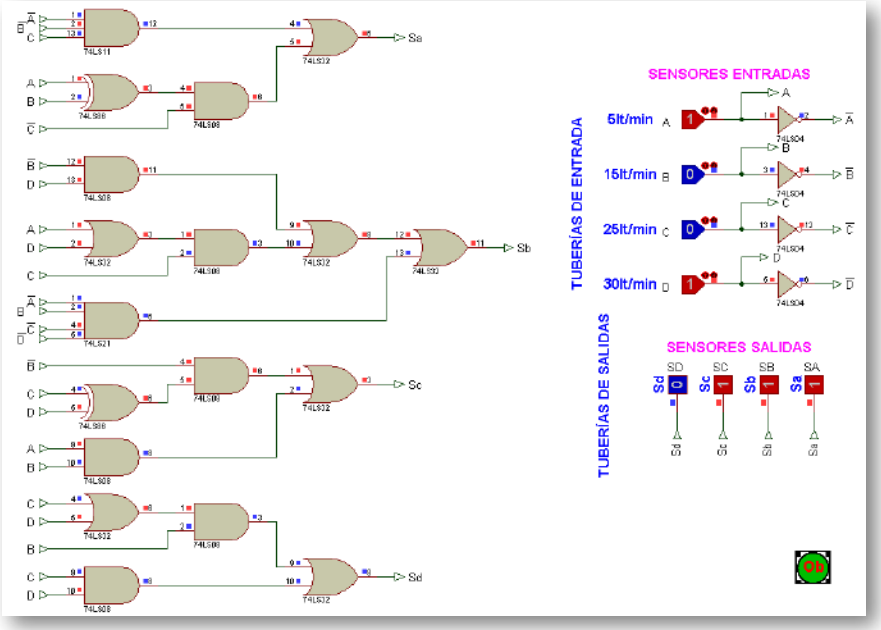
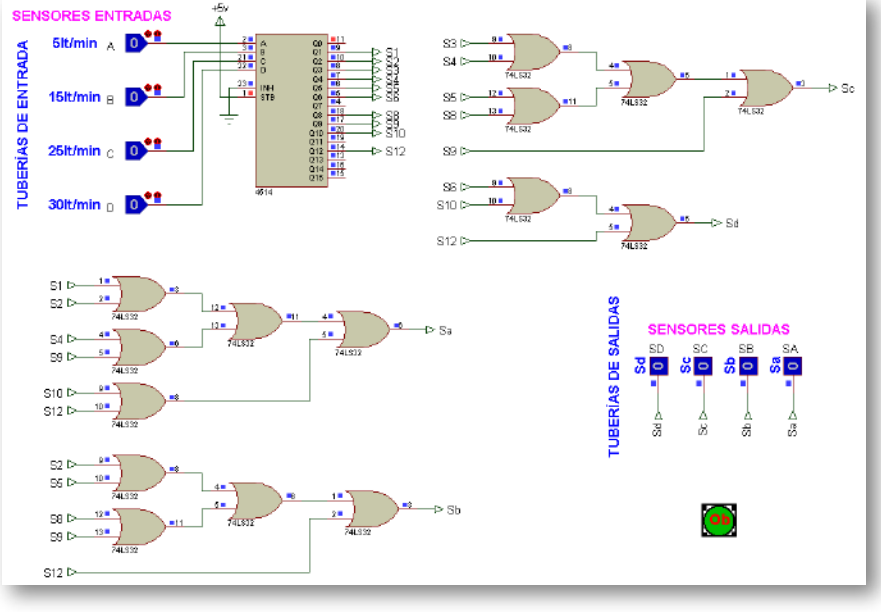


Imagen de la simulación implementada con un **DECODIFICADOR**:



Repositorio en Github ("EA13-Sistema de Tuberías.dsn" y "EA13-Sistema Tuberías y Deco.dsn") con el archivo de la simulación con Proteus v7.8

EA14- Balanzas

Una balanza tiene dos sensores "idénticos" para pesar la misma pieza, aunque pueda diferir muy poco. El valor de cada medición consta de 2 bits y se utilizan 3 displays de 7 segmentos de cátodo común, uno para visualizar el valor de la diferencia entre los 2 sensores, otro para visualizar el valor de la menor medida y el restante para visualizar el valor de la mayor medida.

Diseñar mediante bloques combinacionales estándares (multiplexores, decodificadores, sumadores, comparadores…) el circuito combinacional e impleméntelo en un software de simulación electrónica y verifique su funcionamiento.

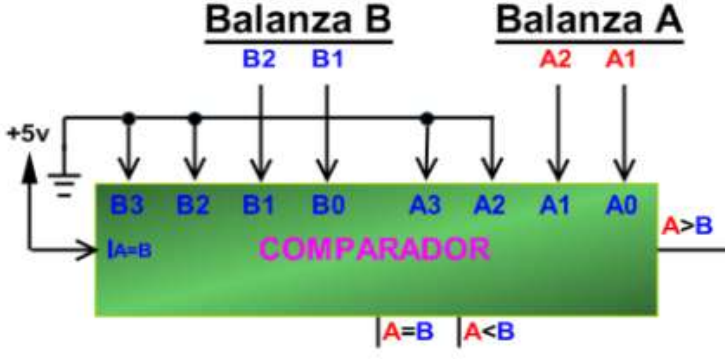
Los bloques combinacionales que se usarán son:

- Comparador de Magnitudes (IC 7485)
- Multiplexores de 2 canales (IC 74157)
- Sumador completo de 4 bits (IC 7483)
- Decodificadores BCD a 7 segmentos (IC 7448)
- Display de 7 segmentos de cátodo común

Bloque del **Comparador de Magnitudes:**

- El circuito integrado 7485 es un comparador de 2 números de 4 bits, de los cuales se usarán 2 bits del **bloque A** y 2 bits del **bloque B**, las demás entradas que no se requieren se conectan a GND.

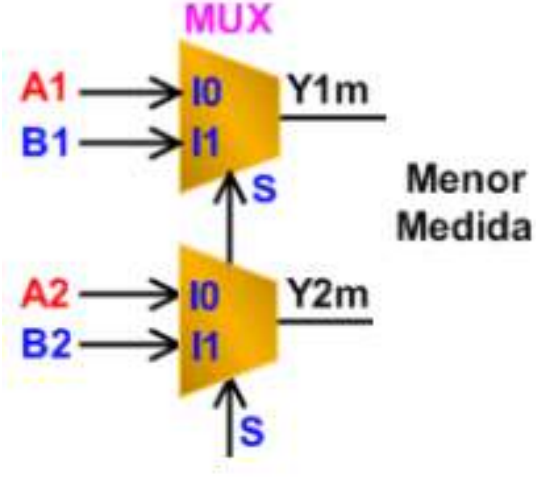
- La entrada auxiliar $I_{A=B}$ se conecta a +Vcc o +5v.
- De las 3 salidas solo se utilizará la de $A > B$, que se activa cuando el peso medido por la **Balanza A** es mayor que el peso medido por la **Balanza B.**



- La señal de la salida se usa como la **línea selectora (S)** para los Multiplexores.

Bloque de 2 **Multiplexores:**

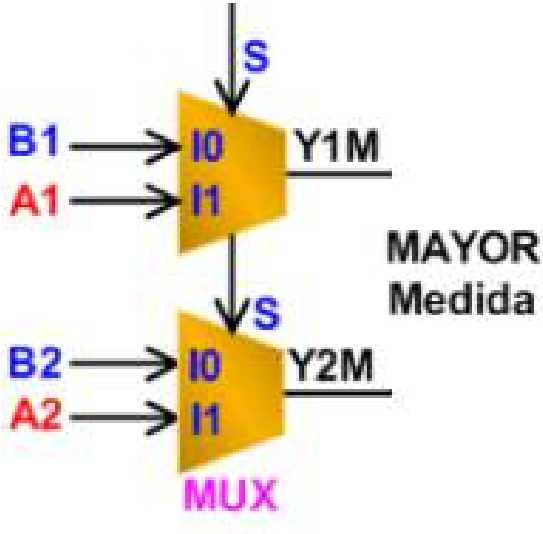
- Se requieren 2 **MUX** de 2 canales cada uno para seleccionar el valor **MENOR** entre las 2 medidas dadas por las balanzas.



- La línea de **Selección (S)** proviene de la salida $A > B$ del Comparador de Magnitudes.
- Si la terminal $A > B = 0$ o sea **S=0** se transfiere a la salida de cada MUX el bit de la **balanza A.**
- Si la terminal $A > B = 1$ o sea **S=1** se transfiere a la salida de cada MUX el bit de la **balanza B.**

Bloque de 2 **Multiplexores:**

- Se requieren 2 **MUX** de 2 canales cada uno para seleccionar el valor **MAYOR** entre las 2 medidas dadas por las balanzas.



- La línea de **Selección (S)** proviene de la salida $A > B$ del Comparador de Magnitudes.
- Si la terminal $A > B = 0$ o sea **S=0** se transfiere a la salida de cada MUX el bit de la **balanza B.**
- Si la terminal $A > B = 1$ o sea **S=1** se transfiere a la salida de cada MUX el bit de la **balanza A.**

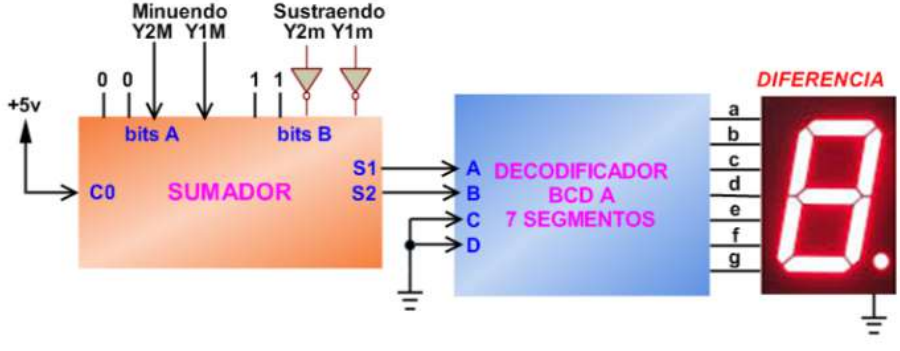
- Bloque de los **Decodificadores de BCD a 7 segmentos:**
  - Se conecta un **Decodificador** a las 2 salidas de los MUX que suministran el valor de la medición **MENOR:**

  - Se conecta un **Decodificador** a las 2 salidas de los MUX que suministran el valor de la medición **MAYOR:**

  - Las salidas de cada **Decodificador** se conectan a los segmentos respectivos de cada Display, visualizando de forma simultánea los valores de la medidas **MENOR** y **MAYOR** obtenidas en las 2 balanzas.
  - Las entradas **BI/RBO, RBI y LT** son activas en BAJO, por consiguiente, se conectan a +5v (ver la tabla de verdad en la hoja de datos).

Bloque del **Sumador completo:**

- Se utiliza el Sumador Completo para realizar la **RESTA o DIFERENCIA** aplicando el **Complemento 2** al **SUSTRAENDO** y se realiza obteniendo el **Complemento 1 y adicionando '1'**.
- Para obtener el **Complemento 1:** se toman los 2 bits que provienen de las salidas de los **MUX del MENOR (Y1m y Y2m)** y se invierten por medio de compuertas **NOT** y se conectan a las entradas de los **bits B** del circuito integrado que pertenecen al **SUSTRAENDO**.
- Para **Adicionar '1'** se conecta la entrada **C0 a +5v que equivale a '1'**.
- Las 2 entradas de los **bits B** más significativos que no se utilizan se conectan a +5v, porque es un número negativo.



- Para el **MINUENDO** se toman los 2 bits que provienen de las salidas de los **MUX del MAYOR (Y1M y Y2M)** y se conectan a las entradas de los **bits A** del circuito integrado
- Las 2 entradas de los **bits A** más significativos que no se utilizan se conectan a GND, porque es un número positivo.
- Las salidas se conectan al **Decodificador** y de allí el **Display** cómo se observa en la imagen anterior.

Diagrama completo del sistema:

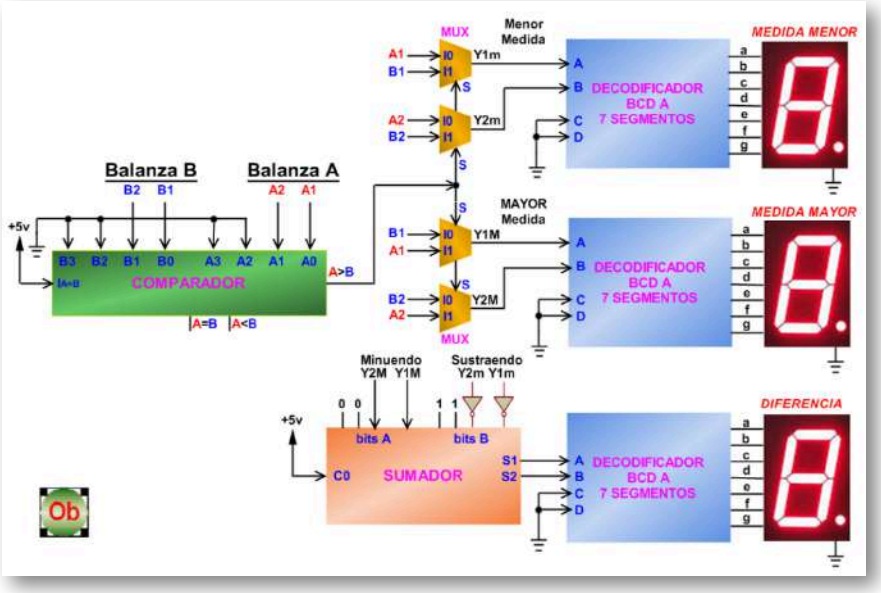
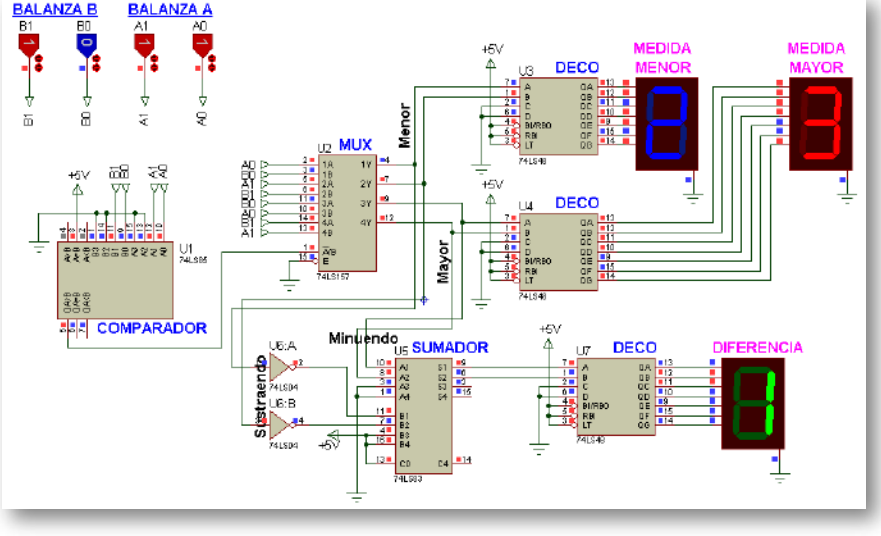


Imagen de la simulación:



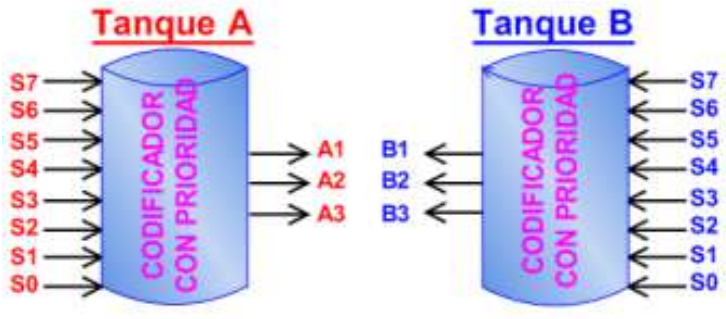
Repositorio en Github ("EA14-Balanzas.dsn") con el archivo de la simulación con Proteus v7.8

EA15- Tanques comunicados

- *Se tienen dos tanques de líquidos que se comunican por una tubería que tiene una **ELECTROVÁLVULA**. Cada tanque tiene 8 sensores distribuidos uniformemente a lo alto del tanque.*
- *Cuando el sensor se cubre de líquido entrega un '1' y cuando NO un '0', si un sensor genera un '1' lógico, todos los sensores que están por debajo de él también lo harán ya que están cubiertos de líquido (se supone que ningún sensor va a fallar).*
- *Se requiere realizar un circuito que active la **ELECTROVÁLVULA (EV='1')** cuando uno de los tanques contenga el doble o más de líquido que el otro tanque. Al activarse el tanque de mayor cantidad de líquido se vaciará al de menor cantidad de líquido (sistema de vasos comunicantes).*
- *Para la solución del circuito se pueden utilizar bloques combinacionales como: decodificadores, codificadores, comparadores, multiplexores, demultiplexores y sumadores y compuertas lógicas.*

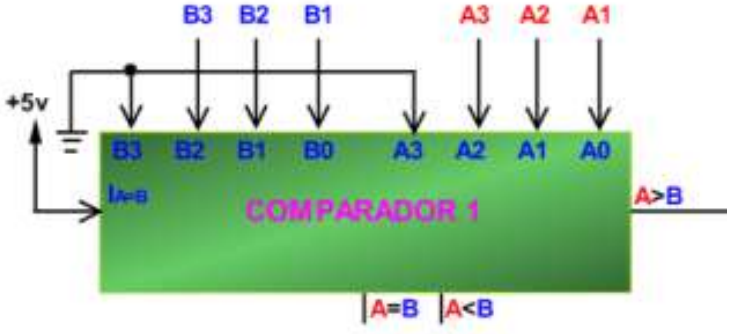
- Los bloques combinacionales que se usarán son:
  - Codificador de prioridad de 8:3 (IC 74148)
  - Comparador de Magnitudes (IC 7485)
  - Multiplexores de 2 canales (IC 74157)
  - Multiplicador x2
  - Control de la Electroválvula
  - Decodificadores BCD a 7 segmentos (IC 7448)
  - Display de 7 segmentos de cátodo común

- Bloque del **Codificador de prioridad de 8:3**:
  - El Codificador tiene 8 entradas para los 8 sensores que tiene cada tanque de líquido, las entradas y salidas son **Negadas**, lo que requiere colocarle a cada una compuerta **NOT** para trabajar con lógica positiva, ya que cuando el líquido cubre el sensor se activa ('1').



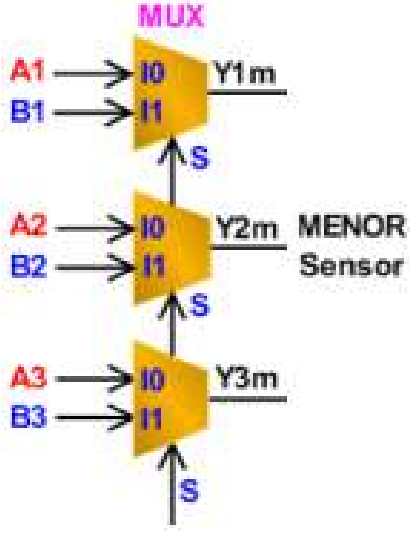
  - Las salidas están **Codificadas** entre el 0_b y el 7_b.
  - Se utilizan 2 circuitos integrados **Codificadores** uno para cada tanque de líquidos.

- Bloque del **Comparador de Magnitudes 1:**
  - El circuito integrado 7485 es un comparador de 2 números de 4 bits c/u, de los cuales se usarán 3 bits del **bloque A** y 3 bits del **bloque B**, provenientes de las salidas de los Codificadores de cada Tanque, las demás entradas que no se requieren se conectan a GND.
  - La entrada auxiliar $\boldsymbol{I_{A=B}}$ se conecta a +Vcc o +5v.
  - De las 3 salidas solo se utilizará la de $\boldsymbol{A > B}$, que se pone en ALTO cuando se han activado más sensores del **Tanque A** que del **Tanque B**.
  - La señal de la salida se usa como la **línea selectora (S)** para los Multiplexores.



Bloque de 3 **Multiplexores:**

- Se requieren 3 **MUX** de 2 canales cada uno para seleccionar el **MENOR** valor de los dos que provienen de los Codificadores de los sensores.

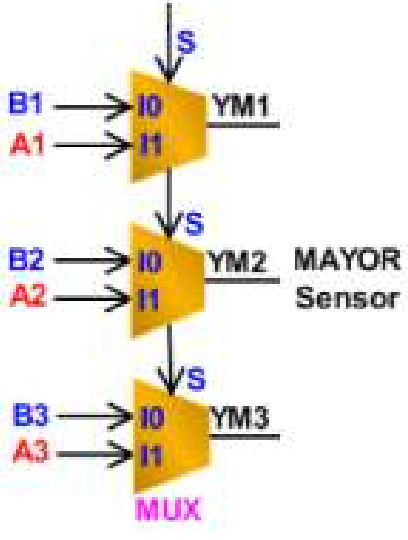


- La línea de **Selección (S)** proviene de la salida $\boldsymbol{A > B}$ del Comparador de Magnitudes.
- Si la salida $\boldsymbol{A > B = 0}$ o sea **S=0** se transfiere a la salida de cada uno de los 3 MUX los bits del **Tanque A.**

- Si la terminal $A > B = 1$ o sea **S=1** se transfiere a la salida de cada uno de los 3 MUX los bits del **Tanque B.**

Bloque de 3 **Multiplexores:**

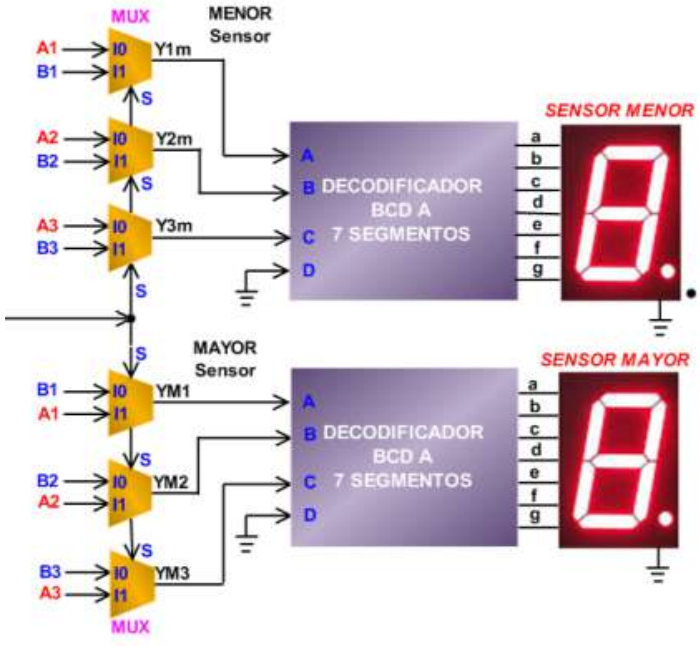
- Se requieren 3 **MUX** de 2 canales cada uno para seleccionar el valor **MAYOR** valor de los dos que provienen de los Codificadores de los sensores.



- La línea de **Selección (S)** proviene de la salida $A > B$ del Comparador de Magnitudes.
- Si la terminal $A > B = 0$ o sea **S=0** se transfiere a la salida de cada uno de los 3 MUX los bits del **Tanque B.**
- Si la terminal $A > B = 1$ o sea **S=1** se transfiere a la salida de cada uno de los 3 MUX los bits del **Tanque A.**

Bloque de los **Decodificadores de BCD a 7 segmentos:**

- Se conecta un **Decodificador** a las 3 salidas de los MUX que suministran el valor **MENOR** de los sensores:



- Se conecta un **Decodificador** a las 3 salidas de los MUX que suministran el valor **MAYOR** de los sensores y otro al valor del **MENOR**.
- Las salidas de cada **Decodificador** se conectan a los segmentos respectivos de cada Display, visualizando de forma simultánea los valores **MENOR** y **MAYOR** obtenidas de los 2 tanques.
- Las entradas **BI/RBO, RBI y LT** son activas en BAJO, por consiguiente, se conectan a +5v (ver la tabla de verdad en la hoja de datos).

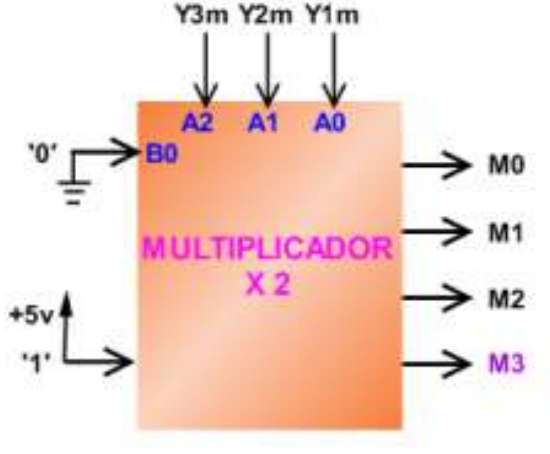
Bloque **Multiplicador:**

- Para saber cuándo un tanque tiene el doble del otro y poder activar la Electroválvula para el intercambio del líquido, se requiere comparar el valor Mínimo Multiplicado por 2 con el valor Mayor. Entonces hay que hallar el valor de la Multiplicación.

- Para ello se implementa un circuito combinacional que multiplique 2 números binarios: los 3 bits del valor **Mínimo (Y1m, Y2m, Y3m)** por el número constante 2_d que en código binario es $\mathbf{10_b}$.
- La estructura para la operación es:

	Y3m	Y2m	Y1m	
		1	0	x
0	0	0	0	
'1'•Y3m	'1'•Y2m	'1'•Y1m		+
'1'•Y3m	**'1'•Y2m**	**'1'•Y1m**	**0**	
M3	**M2**	**M1**	**M0**	

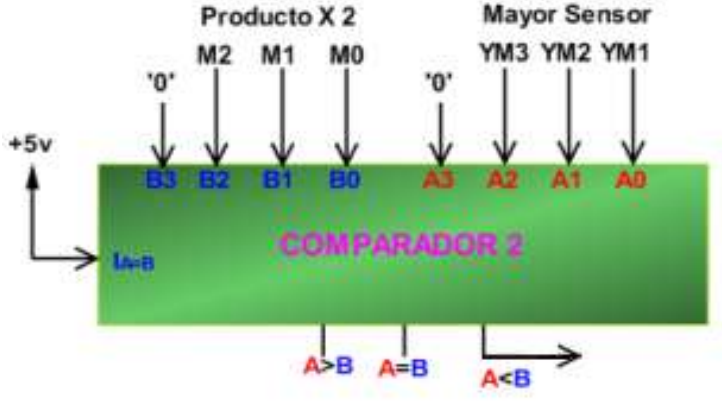
- Se implementa la operación con 3 compuertas **AND** de 2 entradas cada una y sus respectivas salidas etiquetadas como **M1, M2 y M3**, la salida **M0** se conecta a GND como se observa en la estructura de la operación binaria. Nótese en la estructura de la SUMA (color azul) no se presentan **Acarreos**.



- La salida **M3** indica que se presenta desbordamiento del líquido en el tanque, porque tiene un nivel de 8 o superior y los sensores son 8 (del 0 al 7). Si esto sucede significa que el tanque con mayor cantidad de líquido no podrá tener más del doble del otro (porque son 8 sensores).

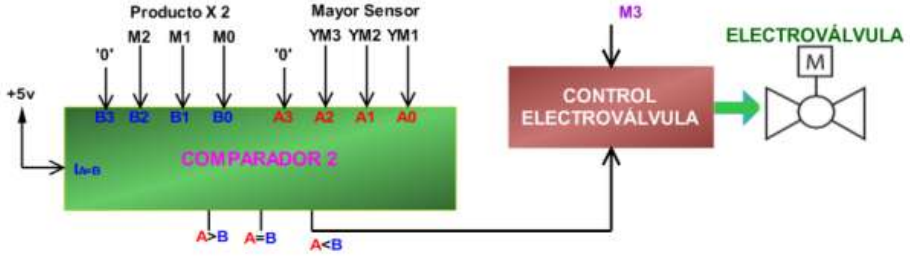
Bloque del **Comparador de Magnitudes 2:**

- El circuito integrado 7485 es un comparador de 2 números de 4 bits, de los cuales se usarán 3 bits del **bloque A** y 3 bits del **bloque B**.
- Las entradas del **bloque A** provienen del **MAYOR** valor de los sensores **(YM1, YM2, YM3)**, la tercera entrada que no se requiere se conecta a GND.
- Las entradas del **bloque B** provienen del bloque **Multiplicador (M1, M2, M3)**, la tercera entrada de este bloque se conecta a GND.
- La entrada auxiliar $\boldsymbol{I_{A=B}}$ se conecta a +Vcc o +5v.



- De las 3 salidas solo se utilizará la de $\boldsymbol{A < B}$, que se activa en ALTO cuando el **Producto** es el doble o más del tanque con el **MAYOR** valor $(Ymínimo\, x2 \geq Ymáximo \quad o \quad M \geq YM)$.
- La señal de la salida de este Comparador se utilizará como una de las entradas al bloque de **Control de la Electroválvula**.

Bloque de **Control de la Electroválvula:**



- Este bloque controla la apertura y cierre de la **Electroválvula**, que solo se **ACTIVA** cuando el contenido de líquido de un tanque sea el doble o más del otro y **NO** se activa si se presenta **Desborde (M3)** o la salida $(A < B)$ del **Comparador 2** está en ALTO.
- Bajo estas condiciones se diseña una tabla de verdad con las entradas etiquetadas como $M3$ y $(A < B)$. Se observa que esta tabla equivale a la tabla de la compuerta **NOR** de 2 entradas.

A<B	M3	EV
0	0	1
0	1	0
1	0	0
1	1	0

- La función lógica es:

$$EV = \overline{M3 + (A < B)}$$

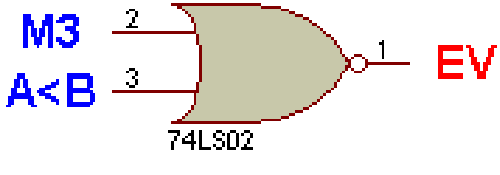


Diagrama de bloques completo:

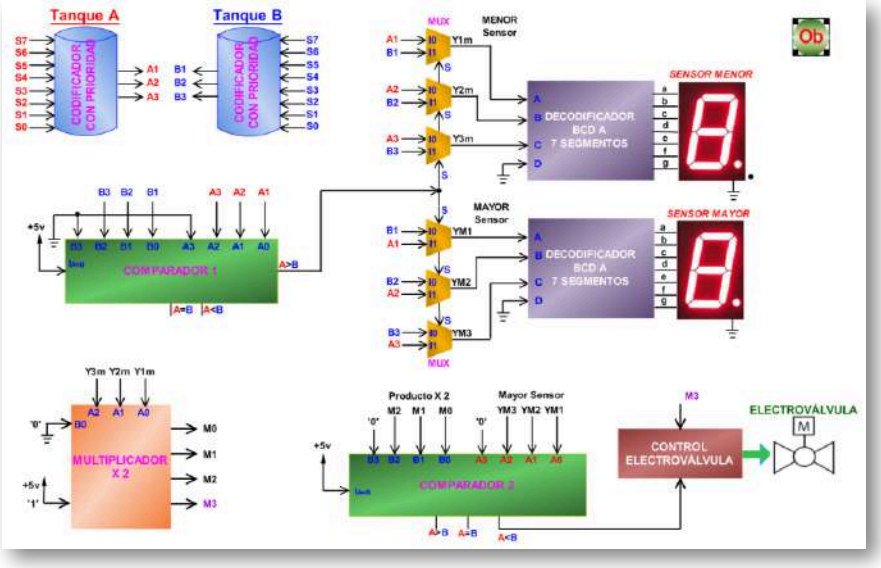
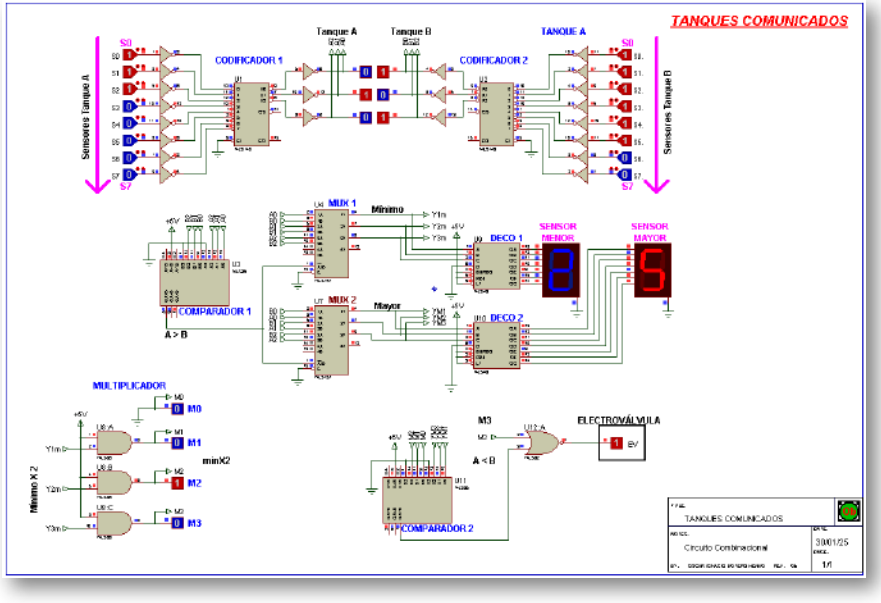
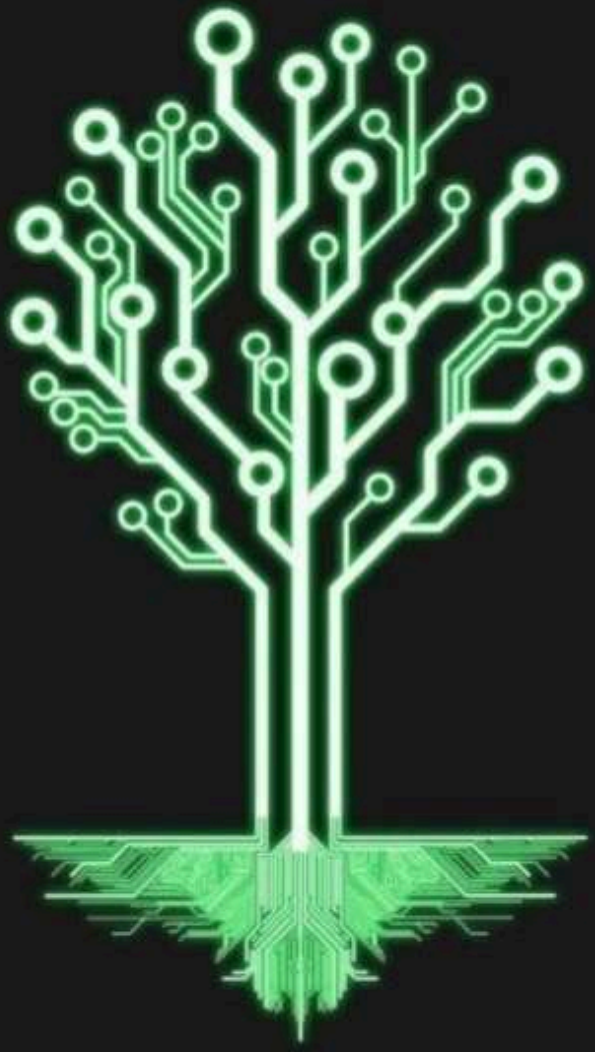


Imagen de la simulación del circuito completo:



Repositorio en Github ("EA15-Tanques comunicados.dsn") con el archivo de la simulación con Proteus v7.8

Ob
REPASO
REPASO



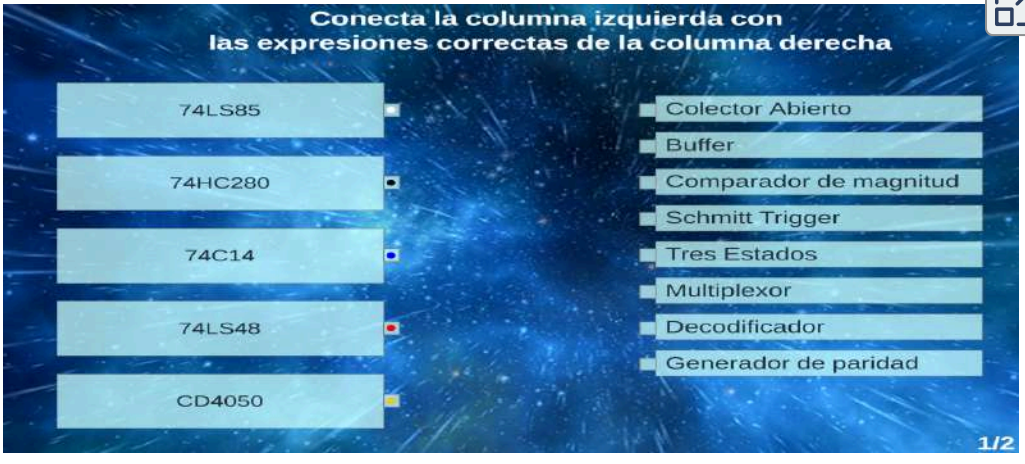
Actividades de Repaso

Sopa de letras con 10 términos pertenecientes al área de la Electrónica Digital y que fueron usados en este libro ***(parte 2).***

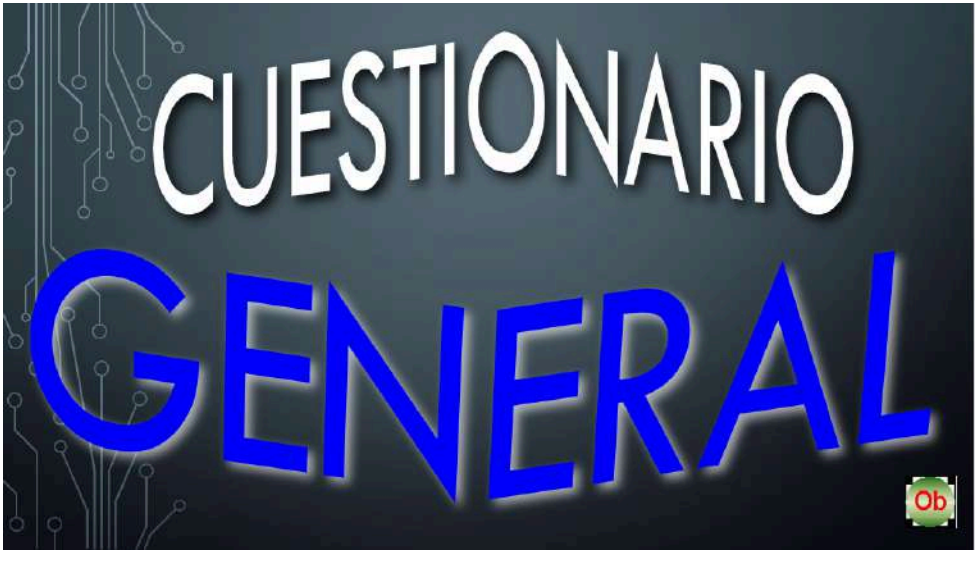
Preguntas de selección múltiple con única respuesta. [9]

[9] Las plantillas para los Objetos Interactivos fueron obtenidas del sitio web del Proyecto Descartes, intervenidas por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

Emparejamiento con términos utilizados en este libro en el *parte 2.*[10]

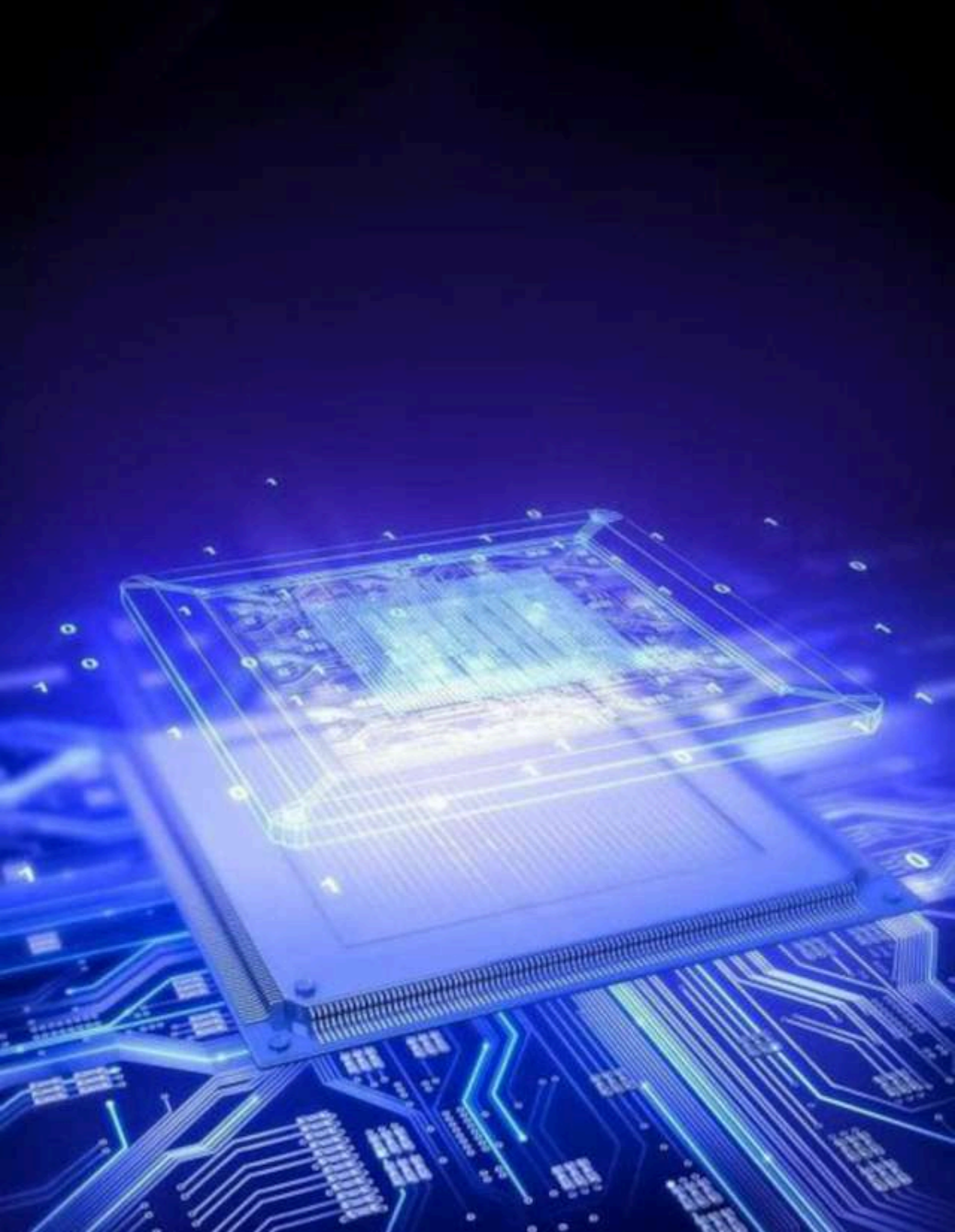


Este objeto interactivo fue generado con la *asistencia* de la Inteligencia Artificial (IA) en el sitio: https://websim.ai/



[10] Las plantillas para los Objetos Interactivos fueron obtenidas del sitio web del Proyecto Descartes, intervenidas por Oscar Ignacio Botero Henao. Plantillas con DescartesJS

Ob
ANEXOS



Anexo A
Hojas de Datos

Algunas Hojas de Datos [15]

Tipo	Familia Lógica TTL	Familia Lógica CMOS	Descripción
AND	7408	4081	4 compuertas de 2 entradas
	7411	4073	3 compuertas de 3 entradas
	7421	4082	2 compuertas de 4 entradas
NAND	7400	4011	4 compuertas de 2 entradas
	7410	4023	3 compuertas de 3 entradas
	7420	4012	2 compuertas de 4 entradas
	7430	4068	1 compuerta de 8 entradas
	74133	----	1 compuerta de 13 entradas
OR	7432	4071	4 compuertas de 2 entradas
	----	4075	3 compuertas de 3 entradas
	----	4072	2 compuertas de 4 entradas
NOR	7402	4001	4 compuertas de 2 entradas
	7427	4025	3 compuertas de 3 entradas
	7425	4002	2 compuertas de 4 entradas
	----	4078	1 compuerta de 8 entradas
NOT	7404	4069	6 compuertas de 1 entrada
YES	7407	4050	6 compuertas de 1 entrada
XOR	7486	4030	4 compuertas de 2 entradas
XNOR	----	4077	4 compuertas de 2 entradas

Familia Lógica TTL	Familia Lógica CMOS	Descripción Sumador / Comparador de magnitudes
7483	4008	Sumador completo de 4 bits
7485	----	Comparador de Magnitudes de 4 bits
Familia Lógica TTL	**Familia Lógica CMOS**	**Descripción Codificadores / Decodificadores**
74147	74HC147	Codificador de prioridad de 10 líneas a 4 líneas
----	4532	Codificador de prioridad de 8 líneas a 3 líneas
74138 74139	----	Decodificador de 3 a 8 líneas de 2 bits
----	74HC238	Decodificador de 3 a 8 líneas de 3 bits
----	4514 4515	Decodificadores de 4 a 16 líneas de 4 bits
7447 7448	----	Decodificador BCD a 7 segmentos
----	4028	Decodificador BCD a decimal
Familia Lógica TTL	**Familia Lógica CMOS**	**Descripción Multiplexores / Demultiplexores**
74150	----	1 Multiplexor de 16 a 1, salida activa en bajo
74151	----	1 Multiplexor de 8 a 1, con 2 salidas: activas en bajo y en alto
74153	----	2 Multiplexores de 4 a 1, salida activa en alto
74157	----	4 Multiplexores de 2 a 1, salida activa en alto
74138	74HC238	1 Demultiplexor de 1 a 8, salidas activas en bajo
74139	----	2 Demultiplexores de 1 a 4, salidas activas en bajo
----	4555 4556	2 Demultiplexores de 1 a 4, 4555 las salidas activas en alto y 4556 activas en bajo
----	74HC154	1 Demultiplexor de 1 a 16, salidas activas en bajo

Familia Lógica TTL	Familia Lógica CMOS	Descripción Sumador
----	4067	1 Demultiplexor de 1 a 16, salidas activas en alto
----	4051	1 Multiplexor-Demultiplexor de 8 canales, salida activa en alto. Como Demux las salidas tienen alta Z
----	4053	3 Multiplexor-Demultiplexor de 2 canales, salida activa en alto. Como Demux las salidas tienen alta Z
Familia Lógica TTL	**Familia Lógica CMOS**	**Descripción Generador de Paridad**
74280	74HC280	Generador de Paridad y Verificador
Familia Lógica TTL	**Familia Lógica CMOS**	**Descripción Tres Estados (Tri-State)**
74125 74126	----	4 Buffer tres estados

Tipo	Referencia	Descripción Colector Abierto
NAND	74LS01, 74LS03	4 compuertas de 2 entradas
	74LS12	3 compuertas de 3 entradas
	74LS22	2 compuertas de 4 entradas
AND	74LS09	4 compuertas de 2 entradas
	74LS15	3 compuertas de 3 entradas
NOT	74LS05, 74LS06	6 compuertas de 1 entrada
NOR	74LS33	4 compuertas de 2 entradas
XOR	74LS136	4 compuertas de 2 entradas
XNOR	74LS266	4 compuertas de 2 entradas

Tipo	Referencia	Descripción Schmitt Trigger
NAND	74132, 4093	4 compuertas de 2 entradas
NOT	74C14	6 compuertas de 1 entrada
YES (Buffer)	4010	6 compuertas de 1 entrada
AND	74HCS08	4 compuertas de 2 entradas
	74HCS21	2 compuertas de 4 entradas
NOR	74HC7002	4 compuertas de 2 entradas

Display de Segmentos (LED)
Display DUAL de 7 segmentos
Display DUAL de 14 segmentos
Display SENCILLO de 16 segmentos

Anexo B
Subfamilias de los CI TTL

Subfamilias TTL [11]

Subfamilia	Prefijo	Capacidad de salida FanOut	Disipación de potencia (mW)	Retardo de propagación (ns)
Estándar	74	10	10	9
Baja potencia	74L	20	1	33
Alta velocidad	74H	10	22	6
Schottky	74S	25	20	3
Schottky de baja potencia	74LS	10	2	9,5
Schottky avanzado	74AS	50	8	1,7
Schottky avanzado de baja potencia	74ALS	10	1,2	4
Alta velocidad	74F	20	4	3

Anexo C
Subfamilias de los CI CMOS

Subfamilias CMOS [12]

Subfamilia	Prefijo
Estándar	CD, HEF, HC
Bajo voltaje	LV
CMOS bajo voltaje	LVC
CMOS Avanzado	AC
CMOS Avanzado compatible con TTL	ACT

Material de la Compuerta CMOS [3]

Material	Capacidad de salida FanOut	Disipación de potencia (µW)	Retardo de propagación (ns)
Compuerta de Silicio serie 74HC	>100	2,5	8
Compuerta Metálica serie 4000	>50	1	50

Anexo D
Fabricantes de Circuitos Integrados

Los LOGOS de las empresas se utilizan con fines exclusivamente "Informativos" para contextualizar. Al final de este Anexo se encuentran las URL de cada imagen [URL logos].

Texas Instruments (EEUU) [16]

En los años 70 y 80, **Texas Instruments** fue líder en la producción de microprocesadores y semiconductores analógicos, actualmente en CI para electrónica de consumo e industria.	**Texas Instruments**

Renesas Technology (Japón) [16]

Hitachi	Mitsubishi Electric	NEC Electronics	**Renesas Technology**
Renesas Technology en el 2003 fue creada por Hitachi y Mitsubishi Electric y en el 2010 se fusionó con NEC Electronics.			

NXP Semiconductors (Países Bajos) [16]

Philips Semiconductors	Freescale Semiconductor	Motorola	**NXP Semiconductors**
En 2006, Philips Semiconductors se independizó y pasó a llamarse **NXP Semiconductors**. En 2015, compró a Freescale Semiconductor (ex-Motorola).			

GlobalFoundries (EEUU) [16]

AMD	IBM	**GlobalFoundries**
En 2009, AMD separó su negocio de fabricación de circuitos integrados, creando **GlobalFoundries**, luego adquirió la división de manufactura de la empresa IBM en 2015.		

STMicroelectronics (Europa) [16]

Thomson Semiconducteurs	SGS Microelettronic	**STMicroelectronics**
En 1987 se fusionaron la empresa francesa Thomson Semiconducteurs y la italiana SGS Microelettronica dando origen a **STMicroelectronics**.		

Broadcom Inc (EEUU) [16]

Avago Technologies	HP/Agilent	**Broadcom Inc**
De la fusión en 2016 entre Avago Technologies y Broadcom Corporation surgió **Broadcom Inc**. La empresa Avago provenía de la antigua división de semiconductores de HP/Agilent.		

Qualcomm (EEUU) [16]

NUVIA	**Qualcomm**
En 2021, **Qualcomm** compró NUVIA, una startup fundada por exingenieros de Apple para desarrollar procesadores de alto rendimiento.	

ON Semiconductor (EEUU) [16]

Motorola	Fairchild Semiconductor	**ON Semiconductor**
En 1999, nació **ON Semiconductor** como un rompimiento de la división de chips de potencia y señal mixta de Motorola Semiconductor. En 2016, adquirió la empresa Fairchild Semiconductor.		

Samsung (Corea del Sur) [16]

En 1974, **Samsung** compró la fábrica Korea Semiconductor, dando inicio al negocio de los circuitos integrados, en los 80 comenzó a fabricar memorias DRAM y luego se convirtió en el mayor productor de memorias del mundo.	**Samsung**

Infineon Technologies (Alemania) [16]

Siemens Semiconductor	Cypress Semiconductor	**Infineon Technologies**
SIEMENS	CYPRESS EMBEDDED IN TOMORROW	infineon
En 1999, Siemens Semiconductor se separó y se convirtió en **Infineon Technologies**. Luego, en 2020, adquirió Cypress Semiconductor.		

Lista de las URL de los LOGOS de las empresas: [volver]

1. Texas Instruments:
   https://companieslogo.com/img/orig/TXN-e197f953.png?t=1720244494&download=true
2. Hitachi:
   https://icon-icons.com/es/icono/hitachi-logo/168125
3. Mitsubishi Electric:
   https://e7.pngegg.com/pngimages/25/808/png-clipart-mitsubishi-motors-car-mitsubishi-outlander-mitsubishi-eclipse-mitsubishi-car-logo-brand-angle-flag-thumbnail.png
4. NEC Electronics:
   https://es.m.wikipedia.org/wiki/Archivo:NEC_logo.svg
5. Renesas Technology:
   https://companieslogo.com/img/orig/6723.T-77040c0f.png?t=1720244490
6. Philips Semiconductors:
   https://1000marcas.net/wp-content/uploads/2020/01/Phillips-Log%D0%BE-500x281.png
7. Freescale Semiconductor:
   https://upload.wikimedia.org/wikipedia/commons/thumb/b/b9/Freescale.svg/449px-Freescale.svg.png
8. Motorola:
   https://1000marcas.net/wp-content/uploads/2020/01/Motorola-Logo-historia-600x473.jpg

9. NXP Semiconductors:
https://upload.wikimedia.org/wikipedia/commons/thumb/1/13/NXP_Semiconductors_Logo.svg/150px-NXP_Semiconductors_Logo.svg.png
10. AMD:
https://upload.wikimedia.org/wikipedia/commons/thumb/7/7c/AMD_Logo.svg/800px-AMD_Logo.svg.png
11. IBM:
https://upload.wikimedia.org/wikipedia/commons/thumb/5/51/IBM_logo.svg/800px-IBM_logo.svg.png
12. GlobalFoundries:
https://isesglobal.com/wp-content/uploads/2023/12/Global-Foundries.webp
13. Thomson Semiconducteurs:
https://1000marcas.net/wp-content/uploads/2021/05/Thomson-logo-500x300.png
14. SGS Microelettronic:
https://upload.wikimedia.org/wikipedia/commons/thumb/e/e9/SGS_Logo.svg/744px-SGS_Logo.svg.png
15. STMicroelectronics:
https://icon-icons.com/icons2/3915/PNG/512/stmicroelectronics_logo_icon_249501.png
16. Avago Technologies:
https://commons.wikimedia.org/wiki/File:Avago_Technologies_logo.svg
17. HP/Agilent:
https://acsanalytical.org/wp-content/uploads/2019/04/agilent-technologies-logo.png
18. Broadcom Inc:
https://1000marcas.net/wp-content/uploads/2021/10/Broadcom-Logo-500x281.png
19. NUVIA:
https://ml.globenewswire.com/Resource/Download/f89879a8-f4dc-4f75-ae9d-1240135298f4?size=3
20. Qualcomm:
https://assets.stickpng.com/images/62bb1fb6b6173d51dfc474b2.png

21. Fairchild Semiconductor:
https://upload.wikimedia.org/wikipedia/commons/thumb/8/8f/Fairchild_Semiconductor_Logo.svg/744px-Fairchild_Semiconductor_Logo.svg.png

22. ON Semiconductor:
https://employeenetwork.com/wp-content/uploads/2017/06/ON_Semiconductor_Logo-01.png

23. Samsung:
https://1000marcas.net/wp-content/uploads/2019/12/logo-Samsung-500x333.png

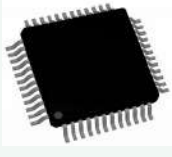
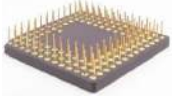
24. Siemens Semiconductor:
https://tryengineeringinstitute.ieee.org/wp-content/uploads/2024/05/Siemens-logo-300x169.png

25. Cypress Semiconductor:
https://www.logo.wine/a/logo/Cypress_Semiconductor/Cypress_Semiconductor-Logo.wine.svg

26. Infineon Technologies:
https://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/Infineon-Logo.svg/800px-Infineon-Logo.svg.png

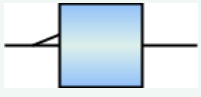
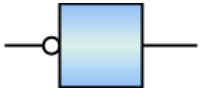
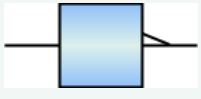
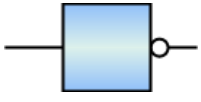
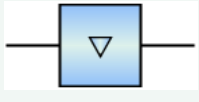
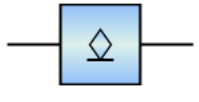
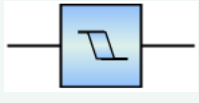
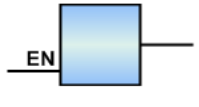
Anexo E
Principales Encapsulados CI

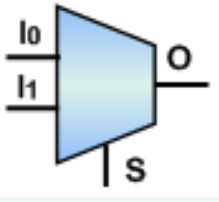
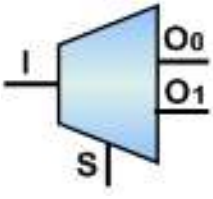
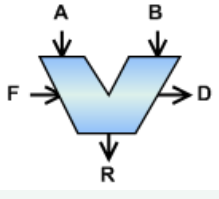
Principales Encapsulados [13][14]

Figura	Descripción
	Dual InLine Packaging (DIP): tiene dos filas de pines paralelos a lo largo del cuerpo del encapsulado
	Small Outline Package (SOP): son más pequeños y compactos; con terminales en "ala de gaviota o en J" que se ensambla sobre el PCB
	**Quad Flat Package (QFP)**: encapsulado cuadrado o rectangular con muchos pines; buena disipación de calor y alta escala de integración
	**Pin Grid Array (PGA)**: posee una gran cantidad de terminales situados en la parte inferior del encapsulado y se ensambla en zócalo
	Ball Grid Array (BGA): en vez de pines utilizan unas bolas pequeñas de soldadura en la superficie inferior que proporciona un mejor rendimiento eléctrico y disipación térmica
	Land Grid Array (LGA): los contactos se acoplan a las almohadillas correspondientes en la PCB

Anexo F
Símbolos estándar para CI

Símbolos estándar

Símbolo	Descripción
	Entrada activa en bajo, norma Comisión Internacional de Electromecánica (IEC)
	Entrada activa en bajo, norma Americana (ANSI)
	Salida activa en bajo, norma Comisión Internacional de Electromecánica (IEC)
	Salida activa en bajo, norma Americana (ANSI)
	Salida de Tres Estados (Tri-State)
	Salida de Colector Abierto (Open Collector)
	Salida Schmitt Trigger
	Entrada de Habilitación (EN = E)

Símbolo	Descripción
	Multiplexor (MUX)
	Demultiplexor (DEMUX)
	Unidad Lógica Aritmética (ALU)

Bibliografía

[1] O. I. Botero, «Electrónica Digital,» 2022. [En línea]. Disponible: https://proyectodescartes.org/iCartesiLibri/materiales_didacticos/Elec [Último acceso: 10 2024].

[2] R. L. Tokheim, Principios Digitales, España: Mc Graw Hill, 1995.

[3] T. L. Floyd, Fundamentos de Electrónica Digital, México: Limusa, 2003.

[4] R. J. Tocci, N. S. Widmer y G. L. Moss, Sistemas Digitales, Principios y Aplicaciones, México: Pearson, 2007.

[5] M. Mano, Diseño Digital, México: Pearson, 2003.

[6] J. P. Uyemura, Diseño de Sistemas Digitales - Un enfoque integrado, México: Thomson, 2000.

[7] J. F. Wakerly, Diseño Digital, Principios y Prácticas, México: Prentice Hall, 2006.

[8] E. Mandado, Sistemas Electrónicos Digitales, España: Marcombo, 2015.

[9] H. Taub y D. Schilling, Electrónica Digital Integrada, España: Marcombo, 1984.

[10] J. W. Bignell y R. L. Donovan, Electrónica Digital, México: CECSA, 1997.

[11] L. García, «Facultad de Ingeniería Eléctrica - Laboratorio de Electrónica Digital I,» [En línea]. Disponible: https://www.fie.umich.mx/lab-electronica/wp-content/uploads/sites/7/2021/10/Practica6-LEDI.pdf. [Último acceso: 12 2024].

[12] M. Ciencia, «Familias lógicas de circuitos integrados:historia,códigos BIC,RTL,DTL,ECL,TTL,MOS,CMOS,HCMOS,BiCMOS.,» 11 07 2024. [En línea]. Disponible: https://youtu.be/XCa7zquVzP8?si=kPjpU2bwwk6C59Bu. [Último acceso: 12 2024].

[13] Cadence Design Systems, Inc. , «IC Packaging Types,» 2025. [En línea]. Disponible: https://resources.pcb.cadence.com/blog/2023-ic-packaging-types. [Último acceso: 01 2025].

[14] Microchipotle, «Tipos de encapsulados en componentes electrónicos,» 2025. [En línea]. Disponible: https://microchipotle.com/tipos-de-encapsulados-en-componentes-electronicos/. [Último acceso: 01 2025].

[15] Alldatasheet, «Electronic Components Datasheet Search,» [En línea]. Available: https://www.alldatasheet.com/.

[16] ChatGPT, "Consulta sobre fabricantes de circuitos integrados", ChatGPT OpenAI, Feb. 2, 2025. [Online]. Disponible: https://chat.openai.com

DESCARTES
matemáticas interactivas