Sistemas Digitales

Preparado por

Carlos Novillo M.

Feliz el hombre que halla la sabiduría,
y que obtiene inteligencia;
porque valen más que la plata,
y produce más beneficios que el oro.
La sabiduría vale más que las piedras preciosas;
¡Todas las cosas que puedas desear, no se pueden comparar a ellas!
Proverbios 3.13-15

PROGRAMA DE ESTUDIO DE SISTEMAS DIGITALES

OBJETIVOS DE LA ASIGNATURA

Capacitar al estudiante para que diseñe circuitos combinacionales y secuenciales de pequeña y mediana complejidad, con circuitos integrados comerciales y con la ayuda de técnicas computacionales.

SÍNTESIS DEL PROGRAMA

Sistemas de numeración y aritmética binaria. Álgebra de Boole. Diseño de circuitos combinacionales. Simplificación de funciones. Redes de salidas múltiples [dispositivos lógicos MSI]: sumador binario, codificadores, decodificadores, multiplexor, demultiplexor, comparador de magnitud, generador/chequeador de paridad. Multivibrador biestable: RS, D, JK y T. Retenedor de datos [Latch] y registros. Contadores/Divisores de frecuencia binarios. Registro de desplazamiento: Conversión S/P y P/S. Análisis y diseño de circuitos secuenciales sincrónicos. Memorias de semiconductor: ROM y RAM. Diseño combinacional y secuencial utilizando memorias ROM.

PROGRAMA DETALLADO

1. ***ÁLGEBRA DE BOOLE***

OBJETIVO.- Al terminar este capítulo el estudiante será capaz de reconocer los diferentes sistemas de numeración relacionados con los dispositivos digitales: Compuertas lógicas, memorias, microprocesadores y microcomputadores. Realizar operaciones aritméticas con el sistema de numeración binario. Conocer los códigos binarios alfanuméricos. Utilizar los postulados, teoremas y conectivos del álgebra de Boole para representar y simplificar las funciones lógicas que se utilizarán en el diseño digital.

1.1 Sistemas analógicos y digitales
1.2 Sistemas de numeración
1.2.1 Aritmética binaria
1.2.2 Complemento restringido [complemento a 1]
1.2.3 Complemento verdadero [complemento a 2]
1.2.4 Otros códigos binarios: BCD, EXC-3, Gray, etc.
1.2.5 Códigos alfanuméricos: EBCDIC y ASCII
1.3 Proposiciones y conectivos binarios
1.3.1 Conectivo AND
1.3.2 Conectivo OR
1.3.3 Operador NOT
1.3.4 Compuerta NAND
1.3.5 Compuerta NOR
1.4 Postulados y teoremas del Álgebra de Boole
1.5 Universalidad de las compuertas NAND y NOR
1.6 Simplificación de funciones utilizando Álgebra de Boole
1.7 Formas estándar de las funciones Booleanas
1.8 Representación y simplificación de funciones

Booleana usando el mapa de Karnaugh
1.8.1 Funciones incompletamente especificadas

HABILIDADES DESARROLLADAS:
- Diferenciar entre fenómenos físicos analógicos y digitales.
- Conocer diferentes tipos de numeración, especialmente el binario, decimal y el hexadecimal.
- Utilizar los postulados y teoremas del álgebra de Boole para simplificar las funciones booleanas.
- Representar las funciones booleanas en sus formas canónicas [normalizadas] y simplificadas.

2. ***DISPOSITIVOS LÓGICOS MSI***

OBJETIVO.- Al terminar este capítulo el estudiante será capaz de construir circuitos combinacionales optimizados a partir de diseños que utilicen circuitos integrados de baja y mediada escala de integración (SSI y MSI).

2.1 Dispositivos Lógicos MSI [Redes de salida múltiple]
2.1.1 Definiciones
2.1.2 Decodificadores de BCD-a-7 segmentos
2.1.3 Sumador aritmético binario
2.1.4 Codificadores y decodificadores
2.1.5 Multiplexores y demultiplexores
2.1.6 Comparadores de magnitud
2.1.7 Generador/Chequeador de paridad
2.2 Diseño usando circuitos MSI

HABILIDADES DESARROLLADAS
- Simplificar funciones booleanas mediante el uso del mapa-K.
- Diseñar circuitos combinacionales de mediana escala de integración.
- Utilizar la tecnología de CIs MSI para implementar circuitos combinacionales de mayor complejidad.

3. ***MULTIVIBRADORES BIESTABLES***

OBJETIVO.- Al terminar este capítulo el estudiante será capaz de relacionar los diferentes multivibradores biestables como las células básicas para el diseño de circuitos binarios secuenciales.

3.1 Dispositivos Multivibradores.
3.1.1 Biestables RS asincrónico y sincrónico
3.1.2 Biestable tipo D
3.1.3 Biestable RS, JK, D y T Maestro-Esclavo [Master-Slave]
3.1.4 Entradas asincrónicas: Preset y Clear
3.1.5 Biestable Disparado por transición [Edge-Triggered]
3.2 Aplicaciones de Flip-Flops
3.2.1 Contadores/divisores de frecuencia asincrónicos
3.2.2 Contadores Ripple-Clock

HABILIDADES DESARROLLADAS
- Analizar el funcionamiento de los diferentes tipos de multivibradores biestables.
- Ilustrar la conversión entre los diferentes tipos de

biestables.

4. ***ANÁLISIS Y DISEÑO SECUENCIAL SINCRÓNICO***

OBJETIVO.- Al terminar este capítulo el estudiante será capaz de construir circuitos digitales secuenciales a partir de diseños que utilicen circuitos integrados de mediana complejidad.

4.1 Análisis y diseño de circuitos secuenciales sincrónicos
4.1.1 Análisis de circuitos secuenciales
4.2 Diseño de circuitos secuenciales
4.3.1 Contadores sincrónicos
4.3.2 Contadores Up/Down
4.3.3 Contadores programables
4.4 Registros de desplazamiento
4.4.1 Conversión Serie-Paralelo y Paralelo-Serie
4.4.2 Contadores de anillo y Johnson
4.5 Detectores de secuencia

HABILIDADES DESARROLLADAS

- Diseñar circuitos secuenciales asincrónicos.
- Diseñar circuitos secuenciales sincrónicos.
- Diseñar contadores binarios sincrónicos programables.
- Diseñar contadores binarios sincrónicos con CIs MSI.

5. ***MEMORIAS***

OBJETIVO.- Al terminar esta unidad el estudiante será capaz de identificar los diferente tipos de memorias

y su arquitéctura para utilizarlas con otros circuitos digitales. Reconocer los diagramas de tiempo en los diferentes tipos de memorias. Modificar el formato de las memorias. Realizar diseños de circuitos combinacionales y secuenciales utilizando memorias ROM.

5.1 Conexión memoria-microprocesador
5.1.1 Terminología usada
5.2 Clasificación de las memorias: ROM, PROM, EPROM, EEPROM, RAM estáticas y dinámicas
5.3 Memorias solo para lectura [ROM]
5.3.1 Memoria ROM como encoder
5.3.2 Memoria PROM
5.3.3 Memorias EPROM, EEPROM y Flash
5.3.4 Temporización de la EPROM
5.4 Memoria de lectura/escritura [RAM]
5.4.1 Arquitectura de la RAM
5.4.2 Temporización de la RAM
5.5 Arreglos de memorias
5.6 Diseño de circuitos digitales utilizando memorias ROM
5.6.1 Diseño combinacional
5.6.2 Diseño secuencial

HABILIDADES DESARROLLADAS

- Relacionar los diferentes tipos de memorias con un microprocesador y con el microcomputador.
- Conocer las diferencias y semejanzas con otros tipos de memorias.
- Conocer la arquitectura [partes constitutivas] y la

temporización [formas de onda] de una ROM.
- Modificar el formato de las memorias RAM y ROM, para aumentar la capacidad de almacenamiento de información.
- Utilizar memorias para el diseño de circuitos combinacionales y secuenciales.

ANEXOS

1- Método tabular Quine-McCluskey
2- Otras funciones booleanas
3- Dispositivos Lógicos Programables [PLDs]
4- Multivibradores [Temporizadores]
5- Punta de prueba digital
6- Resumen de Circuitos Integrados
7- Diagrama de un reloj digital
8- Matriz de 8x8 LEDs

BIBLIOGRAFÍA: [Autor. Título. Editorial. Ciudad año]

➤ ***Libros de texto***:

- Ronald J. Tocci/Neal S. Widmer. Sistemas Digitales, principios y aplicaciones, [Octava Edición]. Prentice Hall Hispanoamericana. México 2003.
- M. Morris Mano. Diseño Digital. Prentice Hall Hipanoamericana. México 1987.

➤ ***Libros recomendados para consulta***:

- F. Hill y G. Peterson. Switching Theory and Logical Design. John Wiley & Sons. New York 1981.
- John F. Wakerly. Diseño Digital, principios y prácticas. Prentice Hall Hipanoamericana. México 2001.

- M. Morris Mano. Arquitectura de Computadoras. Prentice Hall Hipanoamericana. México 1993.
- Texas Instruments. Diseño con Circuitos Integrados TTL. McGraw-Hill 1975.
- Manuales de los fabricantes de CIs TTL: Texas Instruments, National Semiconductors, Motorola, ECG, NTE, Intel, Optoelectrónica, etc.
- Revistas técnicas y cualquier otro tipo de material relacionado con esta asignatura.
- Sitios de Internet.

Sistemas Digitales

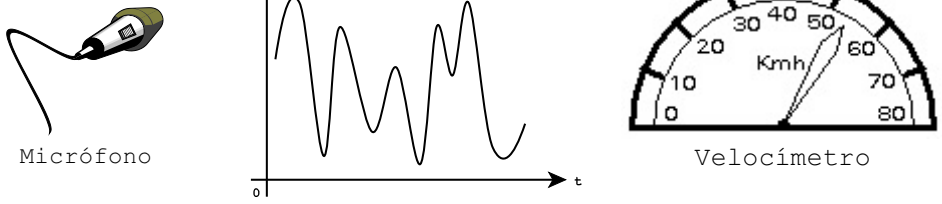
CAPÍTULO 1

Lógica.- Disciplina filosófica cuyo objeto es el estudio de la estructura, fundamento y usos de las expresiones del conocimiento humano. Disposición natural para raciocinar con acierto.

INTRODUCCIÓN

SISTEMAS ANALÓGICOS Y SISTEMAS DIGITALES

Representación Analógica.- Cantidad que se representa por medio de otra que es proporcional a la primera. La deflexión de la aguja de un velocímetro es proporcional a la velocidad de desplazamiento del móvil. La posición angular de la aguja representa el valor de la velocidad y sigue cualquier cambio que ocurra cuando el móvil acelera o frena.



Micrófono

Señal Analógica

Velocímetro

FIGURA 1.1

Característica de las cantidades analógicas.- Pueden variar gradualmente sobre un intervalo continuo de valores.

Representación Digital.- No se utilizan valores proporcionales sino símbolos denominados dígitos.

Señal Digital

Reloj Digital

FIGURA 1.2

Por ejemplo, en un reloj digital el tiempo se mide en horas, minutos y segundos. El tiempo varía continuamente, pero la lectura digital no lo hace de la misma manera, sino que muestra el tiempo cada segundo. Una señal digital tiene un número finito de valores discretos [fig.1.2], a diferencia de una señal analógica que puede tener un número infinito de valores en un rango finito de tiempo [fig. 1.1].

Sin embargo, para fines prácticos, una señal digital se limita a solamente dos niveles: alto o bajo, como se indica en la fig. 1.3, en la que puede verse que el "nivel bajo" corresponde a un rango de valores que va desde 0V hasta 0,8V para voltajes de entrada y desde 0V hasta 0,4V para voltajes de salida.

El "nivel alto" corresponde a un rango de voltajes que va desde 2V hasta 5V para la entrada y desde 2,4V hasta 5V para la salida. Estos valores de voltaje para entrada y salida, que proporcionan los fabricantes, corresponden a la tecnología de circuitos integrados conocida como TTL [Lógica de Transistor con Transistor] que se utilizará en las Prácticas.

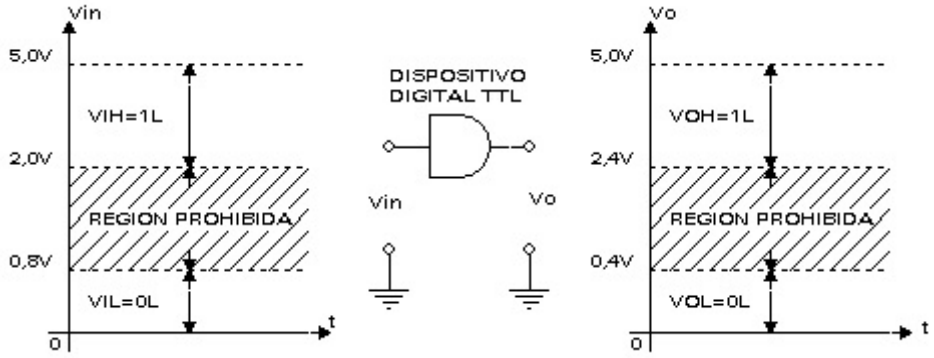


FIGURA 1.3

V_{IH}	VOLTAJE DE ENTRADA ALTO	2V - 5V
V_{IL}	VOLTAJE DE ENTRADA BAJO	0V - 0,8V
V_{OH}	VOLTAJE DE SALIDA ALTO	2,4V - 5V
V_{OL}	VOLTAJE DE SALIDA BAJO	0V - 0,4V
I_{IH}	CORRIENTE DE ENTRADA ALTO	20µA - 50 µA
I_{IL}	CORRIENTE DE ENTRADA BAJO	-1,6mA
I_{OH}	CORRIENTE DE SALIDA ALTO	-400µA
I_{OL}	CORRIENTE DE SALIDA BAJO	16mA

Los valores que se indican en la tabla anterior corresponden a la tecnología TTL estándar y varían

de acuerdo con las sub-tecnologías de fabricación.

Característica de las cantidades digitales. – Varían en etapas discretas.

ANALÓGICO: Variación Continua
DIGITAL: Variación Discreta

La lectura de fenómenos físicos **analógicos se presta a interpretaciones.**

La lectura **digital no presenta ambigüedades.**

Sistema Analógico. – Dispositivo que maneja información física representada en forma analógica. Las cantidades varían en un intervalo continuo de valores.

SISTEMA DIGITAL. – Maneja información discreta, puede ser electrónico, mecánico, magnético o neumático.

Ventajas de las Técnicas Digitales

- MAYOR FACILIDAD PARA DISEÑAR CON CIs
- MAYOR FLEXIBILIDAD PARA IMPLEMENTAR LOS DISEÑOS
- FACILIDAD PARA ALMACENAR INFORMACIÓN
- MAYOR EXACTITUD Y PRECISIÓN
- PROGRAMACIÓN DE LA OPERACIÓN
- MAYOR INMUNIDAD AL RUIDO
- MAYOR GRADO DE INTEGRACIÓN

LIMITACIÓN DE LAS TÉCNICAS DIGITALES

EL MUNDO REAL ES FUNDAMENTALMENTE ANALÓGICO

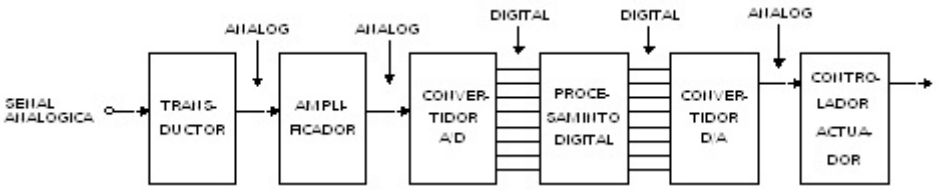


FIGURA 1.4

Aplicaciones de los Circuitos Digitales

- COMPUTADORAS, CALCULADORAS
- MEDICIÓN DEL TIEMPO: RELOJES Y CRONÓMETROS
- TELEFONÍA DIGITAL
- RADIO Y TELEVISIÓN DIGITAL [ALTA FIDELIDAD]
- GRABACIÓN DE AUDIO Y VIDEO
- FOTOGRAFÍA MODERNA Y PROCESAMIENTO DIGITAL DE IMÁGENES
- EQUIPO MÉDICO
- MEDICINA COMPUTARIZADA A DISTANCIA
- ÁREA INDUSTRIAL
- EXPLOTACIÓN PETROLERA
- SIMULACIÓN
- GENERADORES DE SEÑAL
- CONTROL ELECTRÓNICO EN AUTOMÓVILES
- CONTROL INTELIGENTE DE TRÁFICO
- EQUIPO DE MEDICIÓN: OSCILOSCOPIOS, ANALIZADORES Y MULTÍMETROS DIGITALES
- ELECTRODOMÉSTICOS: LAVADORAS, HORNOS DE MICROONDAS, ETC.
- VIDEO JUEGOS

SISTEMAS DE NUMERACIÓN

Sistema de Numeración.- Se define como un conjunto de cifras y siglas reunidas según algunas leyes matemáticas para representar valores numéricos. Por ejemplo, al número 352.91 se lo puede representar de la siguiente forma.

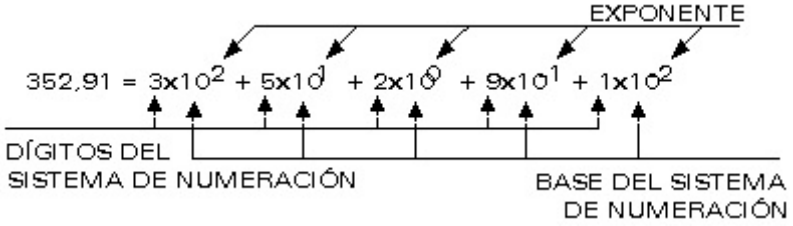
MSD ┐ ┌ **LSD**
352.91 = 300 + 50 + 2 + 0.9 + 0.01,
└ **PUNTO DECIMAL**

[MSD = Most Significant Digit ➔ Dígito más significante]
[LSD = Least Significant Digit ➔ Dígito menos significante]

Otra forma de escribir el número 352.91 es

352,91 = 3x100 + 5x10 + 2x1 + 9x0,1 + 1x0,01,

o también,



Del ejemplo se deduce que un sistema de numeración está caracterizado por los parámetros: Base, Dígitos y Ponderación.

1. **La Base del Sistema de Numeración: B,** puede ser cualquier entero positivo diferente de 0 y 1. Entonces B puede tomar los valores 2, 3, 4, 5, 6, ..., etc.

BASE	SISTEMA DE NUMERACIÓN	
2	BINARIO	BIN
8	OCTAL	OCT
10	DECIMAL	DEC
16	HEXADECIMAL	HEX

2. **Los Dígitos del Sistema de Numeración,** son los símbolos que usan los sistemas de numeración para representar cantidades o valores numéricos. Un sistema de numeración de base B tiene B dígitos [símbolos o guarismos] diferentes, estos son: 0, 1, 2, ..., etc., hasta [B - 1]. En consecuencia, los sistemas de numeración antes indicados usan los siguientes símbolos o dígitos.

BASE	SISTEMA DE NUMERACIÓN	DÍGITOS DEL SISTEMA DE NUMERACIÓN
2	BINARIO	0 y 1
8	OCTAL	0, 1, 2, 3, 4, 5, 6 y 7
10	DECIMAL	0, 1, 2, 3, 4, 5, 6, 7, 8 y 9
16	HEXADECIMAL	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F

Con la ayuda de estos símbolos como dígitos, se puede expresar cualquier cantidad.

3. **Ponderación,** la base elevada a un exponente se denomina ponderación o peso. Un valor numérico puede expresarse como un sumatorio de productos entre los dígitos del sistema y una serie ordenada de ponderaciones, correspondientes a las potencias positivas o negativas de la base como se indica a continuación.

$$N_B = \alpha_m B^m + \alpha_{m-1} B^{m-1} + \ldots + \alpha_0 B^0 + \alpha_{-1} B^{-1} + \ldots + \alpha_{-p+1} B^{-p+1} + \alpha_{-p} B^{-p}$$

Este es un ***Sistema de Numeración Posicional*** en el que la ponderación del dígito depende de su posición dentro del número. De manera que, el dígito de la derecha tiene la menor ponderación [menos significante] y el de la izquierda, la mayor ponderación [más significante].

Desarrollo Polinomial.- A un número cualquiera N_B se lo puede expresar de la siguiente manera.

$$N_B = \underbrace{\alpha_m B^m + \alpha_{m-1} B^{m-1} + \ldots + \alpha_0 B^0}_{\text{Parte Entera}} + \underbrace{\alpha_{-1} B^{-1} + \ldots + \alpha_{-p} B^{-p}}_{\text{Parte Fraccionaria}}$$

Que en forma simplificada puede escribirse así

$$N_B = \sum_{i=-p}^{m} \alpha_i B^i$$

donde

B =	Base del sistema de numeración correspondiente.
α =	Cualquiera de los dígitos del sistema de numeración.
i =	Lugar que ocupa el dígito en la serie ordenada que representa una cantidad o un valor numérico.
m + 1 =	Número de dígitos correspondiente a las potencias positivas (**parte entera**).
p =	Número de dígitos correspondiente a las potencias negativas (**parte fraccionaria**).

Los dígitos correspondientes a las potencias positivas y los correspondientes a las potencias negativas están separados por una coma o un punto, dividiendo así en dos partes a los dígitos representativos.

Los dígitos a la izquierda del punto corresponden a la parte entera [ponderaciones ≥ 1].
Los dígitos a la derecha del punto corresponden a la parte fraccionaria [ponderaciones < 1].

Entonces, el número, en la base de numeración B, quedaría como:

$$N_B = \alpha_m\alpha_{m-1}\ldots\alpha_0,\alpha_{-1}\alpha_{-2}\ldots\alpha_{-p+1}\alpha_{-p}$$
Parte Entera, Parte Fraccionaria

Conversión de la Base Decimal a una Base Cualquiera

B.- El procedimiento para convertir un número decimal $[X_{10}]$ a su equivalente en base B $[X_B]$, consiste en dividir el número en dos partes: entera $[E_{10}]$ y fraccionaria $[F_{10}]$.

$$\Rightarrow X_{10} = E_{10}\langle x\rangle, F_{10}\langle x\rangle$$

Donde

1. $E_{10}\langle X\rangle$ es la parte entera de X_{10}, tal que

$$E_{10}\langle x\rangle = \alpha_m B^m + \alpha_{m-1}B^{m-1} + \ldots + \alpha_1 B + \alpha_0$$

2. $F_{10}\langle X\rangle$ es la parte fraccionaria de X_{10}, tal que

$$F_{10}\langle x\rangle = \alpha_{-1}B^{-1} + \alpha_{-2}B^{-2} + \ldots + \alpha_{-p+1}B^{-p+1} + \alpha_{-p}B^{-p}$$

Para determinar los coeficientes α, que vendrían a ser los dígitos en el nuevo sistema de numeración,

se procede en dos partes.

1° **Parte entera $E_{10}<X>$**

$$E_{10}\langle x\rangle = \alpha_m B^m + \alpha_{m-1}B^{m-1} + \ldots + \alpha_1 B + \alpha_0$$

Si a este polinomio se lo divide por B, se tiene

$$\frac{E\langle x\rangle}{B} = Q_1 = \alpha_m B^{m-1} + \alpha_{m-1}B^{m-2} + \ldots + \alpha_2 B + \alpha_1$$

$$\text{Residuo} = R_0 = \alpha_0$$

$$\frac{Q_1}{B} = Q_2 = \alpha_m B^{m-2} + \alpha_{m-1}B^{m-3} + \ldots + \alpha_3 B + \alpha_2$$

$$\text{Residuo} = R_1 = \alpha_1$$

$$\frac{Q_2}{B} = Q_3 = \alpha_m B^{m-3} + \alpha_{m-1}B^{m-4} + \ldots + \alpha_4 B + \alpha_3$$

$$\text{Residuo} = R_2 = \alpha_2$$

-
-
-

$$\frac{Q_{m-2}}{B} = Q_{m-1} = \alpha_m B + \alpha_{m-1}$$

$$\text{Residuo} = R_{m-2} = \alpha_{m-2}$$

$$\frac{Q_{m-1}}{B} = Q_m = \alpha_m$$

$$\text{Residuo} = R_{m-1} = \alpha_{m-1}$$

$$\frac{Q_m}{B} = 0$$

$$\text{Re siduo} = R_m = \alpha_m$$

El número en base B quedaría como sigue

$$\boxed{\begin{array}{l} R_m R_{m-1} R_{m-2} \ldots R_2 R_1 R_0 \\ \alpha_m \alpha_{m-1} \alpha_{m-2} \ldots \alpha_2 \alpha_1 \alpha_0 \end{array}}$$

Donde α_m, ..., α_0, representan los dígitos de la parte entera en el nuevo sistema de numeración.

2° **Parte fraccionaria $F_{10}<X>$**

$$F_{10}\langle x\rangle = \alpha_{-1}B^{-1} + \alpha_{-2}B^{-2} + \ldots + \alpha_{-p+1}B^{-p+1} + \alpha_{-p+1}B^{+p}$$

Si a este polinomio se multiplica por B, se tiene:

$$F\langle x\rangle \times B = \frac{\alpha_{-1}}{E_1} + \frac{\alpha_{-2}B^{-1} + \alpha_{-3}B^{-2} + \ldots + \alpha_{-p+1}B^{-p+2} + \alpha_{-p}B^{-p+1}}{F_1}$$

$$F_1 \times B = \frac{\alpha_{-2}}{E_2} + \frac{\alpha_{-3}B^{-1} + \alpha_{-4}B^{-2} + \ldots + \alpha_{-p+1}B^{-p+3} + \alpha_{-p}B^{-p+2}}{F_2}$$

$$F_2 \times B = \frac{\alpha_{-3}}{E_3} + \frac{\alpha_{-4}B^{-1} + \alpha_{-5}B^{-2} + \ldots + \alpha_{-p+1}B^{-p+4} + \alpha_{-p}B^{-p+3}}{F_3}$$

Donde E_1, E_2, E_3, etc. representan las partes enteras de los resultados y F_1, F_2, F_3, etc., las nuevas partes fraccionarias. Este proceso continúa hasta que $F_p = 0$, siempre que esto sea posible o hasta obtener un error $\leq \varepsilon$. Donde ε ***es el máximo error permisible***. La parte fraccionaria del número quedaría de la siguiente manera.

E_1	E_2	E_3	...	E_{p-1}	E_p
α_{-1}	α_{-2}	α_{-3}	...	α_{-p+1}	α_{-p}

En este caso, α_{-1}, ..., α_{-p}, representan los dígitos de la parte fraccionaria en el nuevo sistema de numeración.

Ejemplo.- Transformar el número 5142.36_{10} a base: hexadecimal, octal y binaria.

1°. Parte entera [hexadecimal].

$\frac{5142}{16} = Q_1 = 321$ Residuo $R_0 = 6 = \alpha_0$ [LSD]

$\frac{321}{16} = Q_2 = 20$ Residuo $R_1 = 1 = \alpha_1$

$\frac{20}{16} = Q_3 = 1$ Residuo $R_2 = 4 = \alpha_2$

$\frac{1}{16} = Q_4 = 0$ Residuo $R_3 = 1 = \alpha_3$ [MSD]

$\Rightarrow E_{10}\langle X\rangle = 1416_{16}$

1ro. Parte entera [octal].

$\frac{5142}{8} = Q_1 = 642$ Residuo $R_0 = 6 = \alpha_0$ [LSD]

$\frac{642}{8} = Q_2 = 80$ Residuo $R_1 = 2 = \alpha_1$

$\frac{80}{8} = Q_3 = 10$ Residuo $R_2 = 0 = \alpha_2$

$\frac{10}{8} = Q_4 = 1$ Residuo $R_3 = 2 = \alpha_3$

$\frac{1}{8} = Q_5 = 0$ Residuo $R_4 = 1 = \alpha_4$ [MSD]

$\Rightarrow E_{10}\langle X\rangle = 12026_8$

1°. Parte entera [binario].

5142	0	α_0	[LSB]
2571	1	α_1	
1285	1	α_2	
642	0	α_3	
321	1	α_4	
160	0	α_5	
80	0	α_6	
40	0	α_7	
20	0	α_8	
10	0	α_9	
5	1	α_{10}	
2	0	α_{11}	
1	1	α_{12}	[MSB]
0			
Cociente	Resid.	Dígito	

$$\Rightarrow E_{10}\langle X\rangle = 1\ 0100\ 0001\ 0110_{16}$$

2°. Parte fraccionaria [hexadecimal]

$0{,}36 \times 16 = 5.76 \quad E_1 = 5 = \alpha_{-1} \quad F_1 = 0{,}76 \quad \varepsilon_1 = 0{,}76 \times 16^{-1}$

$0{,}76 \times 16 = 12.16 \quad E_2 = 12 = \alpha_{-2} \quad F_2 = 0{,}16 \quad \varepsilon_2 = 0{,}16 \times 16^{-2}$

$0{,}16 \times 16 = 2.56 \quad E_3 = 2 = \alpha_{-3} \quad F_3 = 0{,}56 \quad \varepsilon_3 = 0{,}56 \times 16^{-3}$

$$\Rightarrow F_{10}\langle X\rangle = 0.5C2_{16}$$

2°. Parte fraccionaria [octal]

$0.36 \times 8 = 2.88 \quad E_1 = 2 = \alpha_{-1} \quad F_1 = 0.88 \quad \varepsilon_1 = 0.88 \times 8^{-1}$

$0.88 \times 8 = 7.04 \quad E_2 = 7 = \alpha_{-2} \quad F_2 = 0.04 \quad \varepsilon_2 = 0.04 \times 8^{-2}$

$0.04 \times 8 = 0.32 \quad E_3 = 0 = \alpha_{-3} \quad F_3 = 0.32 \quad \varepsilon_3 = 0.32 \times 8^{-3}$

$0.32 \times 8 = 2.56 \quad E_4 = 2 = \alpha_{-4} \quad F_4 = 0.56 \quad \varepsilon_4 = 0.56 \times 8^{-4}$

$$\Rightarrow F_{10}\langle X\rangle = 0.2702_8$$

2°. Parte fraccionaria [binario]

$0{,}36 \times 2 = 0{,}72 \quad E_1 = 0 = \alpha_{-1} \quad F_1 = 0{,}72 \quad \varepsilon_1 = 0{,}72 \times 2^{-1}$

$0{,}72 \times 2 = 1{,}44 \quad E_2 = 1 = \alpha_{-2} \quad F_2 = 0{,}44 \quad \varepsilon_2 = 0{,}44 \times 2^{-2}$

$0{,}44 \times 2 = 0{,}88 \quad E_3 = 0 = \alpha_{-3} \quad F_3 = 0{,}88 \quad \varepsilon_3 = 0{,}88 \times 2^{-3}$

$0{,}88 \times 2 = 1{,}76 \quad E_4 = 1 = \alpha_{-4} \quad F_4 = 0{,}76 \quad \varepsilon_4 = 0{,}76 \times 2^{-4}$

$0{,}76 \times 2 = 1{,}52 \quad E_5 = 1 = \alpha_{-5} \quad F_5 = 0{,}52 \quad \varepsilon_5 = 0{,}52 \times 2^{-5}$

$0{,}52 \times 2 = 1{,}04 \quad E_6 = 1 = \alpha_{-6} \quad F_6 = 0{,}04 \quad \varepsilon_6 = 0{,}04 \times 2^{-6}$

$0{,}04 \times 2 = 0{,}08 \quad E_7 = 0 = \alpha_{-7} \quad F_7 = 0{,}08 \quad \varepsilon_7 = 0{,}08 \times 2^{-7}$

$0{,}08 \times 2 = 0{,}16 \quad E_8 = 0 = \alpha_{-8} \quad F_8 = 0{,}16 \quad \varepsilon_8 = 0{,}16 \times 2^{-8}$

$0{,}16 \times 2 = 0{,}32 \quad E_9 = 0 = \alpha_{-9} \quad F_9 = 0{,}32 \quad \varepsilon_9 = 0{,}32 \times 2^{-9}$

$0{,}32 \times 2 = 0{,}64 \quad E_{10} = 0 = \alpha_{-10} \quad F_{10} = 0{,}64 \quad \varepsilon_{10} = 0{,}64 \times 2^{-10}$

$0{,}64 \times 2 = 1{,}28 \quad E_{11} = 1 = \alpha_{-11} \quad F_{11} = 0{,}28 \quad \varepsilon_{11} = 0{,}28 \times 2^{-11}$

$0{,}28 \times 2 = 0{,}56 \quad E_{12} = 0 = \alpha_{-12} \quad F_{12} = 0{,}56 \quad \varepsilon_{12} = 0{,}56 \times 2^{-12}$

$$\Rightarrow F_{10}\langle X\rangle = 0{,}010111000010_2$$

De manera que

$$5142{,}36_{10} \equiv 1\ 416{,}5C2_{16}$$

$5142,36_{10}$, ≡ 12 026,217 27_8

$5142,36_{10}$, ≡ 1 0100 0001 0110,0101 1100 0010_2

El error es $\varepsilon = 0,56 \times 8^{-4} = 0,56 \times 16^{-3} = 0,56 \times 2^{-12} = 136,72 \times 10^{-6}$.

Conversión desde Cualquier Base B a Decimal.- Para convertir un número expresado en base B a decimal, se usa directamente la ecuación del desarrollo polinomial.

Ejemplo 1.- Convertir el número $EC9,0B5_H$ a su equivalente decimal (N_{10}).

$N_{10} \equiv E \times 16^2 + C \times 16 + 9 + 0 \times 16^{-1} + B \times 16^{-2} + 5 \times 16^{-3}$

$N_{10} = 14 \times 246 + 12 \times 16 + 9 + 0 + 11 \times 0,00390625 +$
$+ 13 \times 0,000244141$

$N_{10} = 3584 + 192 + 9 + 0,04296875 + 0,0012207031$

$$\boxed{N_{10} = 3785,0441894531}$$

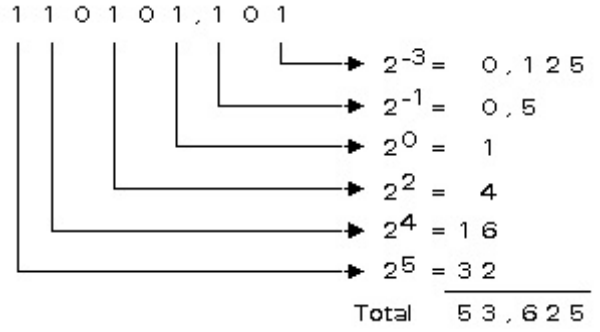
Ejemplo 2.- Convertir el número 11 $0101,101_2$ a su equivalente en base decimal (N_{10})

$N_{10} = 1 \times 2^5 + 1 \times 2^4 + 0 + 1 \times 2^2 + 0 + 1 + 1 \times 2^{-1} + 0 + 1 \times 2^{-3}$

$N_{10} = 32 + 16 + 4 + 1 + 0,5 + 0,125$

$$\boxed{N_{10} = 53,625}$$

Otra forma, sería sumando las ponderaciones de los 1s que aparecen en el número binario, como se indica a continuación.



La siguiente tabla muestra algunas potencias de 2 útil para facilitar la conversión de binario natural a decimal o viceversa.

n	2^n	2^{-n}
0	1	1
1	2	0,5
2	4	0,25
3	8	0,125
4	16	0,0625
5	32	0,03125
6	64	0,015625
7	128	0,0078125
8	256	0,00390625
9	512	0,00195313
10	1024	0,0009766

Dirección IP [IP Address].- Una dirección IP [Internet Protocol] es única y sirve para direccionar a un computador específico conectado a Internet o a una red local. La dirección tiene el formato a.b.c.d donde a, b, c y d son números entre 0 y 255 inclusive y se pueden expresar en decimal o en binario, están sujetos a una serie de reglas y convenciones. Todas las comunicaciones entre los computadores que se encuentran conectados a Internet se basan en direcciones IP.

Ejemplo.- La dirección IP: 192.137.205.10, expresada en decimal, representarla en binario.

Entonces: $192_{10} = 1100\ 0000_2$
$137_{10} = 1000\ 1001_2$
$205_{10} = 1100\ 1101_2$
$10_{10} = 0000\ 1010_2$

Por tanto, la dirección IP correspondiente es 11000000.10001001.11001101.00001010 en binario.

Una tabla que resulta útil para trabajar con direcciones IP, se indica a continuación.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	DEC
1	0	0	0	0	0	0	0	128
1	1	0	0	0	0	0	0	192
1	1	1	0	0	0	0	0	224

1	1	1	1	0	0	0	0	240
1	1	1	1	1	0	0	0	248
1	1	1	1	1	1	0	0	252
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	2
0	0	0	0	0	1	0	0	4
0	0	0	0	1	0	0	0	8
0	0	0	1	0	0	0	0	16
0	0	1	0	0	0	0	0	32
0	1	0	0	0	0	0	0	64
1	0	0	0	0	0	0	0	128

Dirección MAC.- Es una dirección única que se adjudica a toda estación final [computador conectado a Internet] dentro de la infraestructura (entre ellos se encuentran los adaptadores de LAN en la placa base, a puertos de conmutadores y puertos de enrutadores o routers). También se la conoce como dirección física o Ethernet de un host.

Aritmética Binaria.- Todas las operaciones aritméticas conocidas en el sistema de numeración decimal, pueden también realizarse en cualquier otro sistema de numeración, para ello se aplican las mismas reglas de la aritmética común. Aquí se estudiaran las cuatro operaciones básicas: suma, resta, multiplicación y división, aplicadas al sistema de numeración binario.

Suma Binaria

TABLA DE LA SUMA
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Ejemplo.- Dados los valores binarios de A y B obtener S = A + B.

Donde

A = 101 1001,1110

B = 100 0111,0011

	1	0	0	0	0	1	1	1	1	1	0	←	Carry [Exceso]	
		1	0	1	1	0	0	1	,	1	1	1	0	[= $89{,}875_{10}$]
+		1	0	0	0	0	1	1	,	0	0	1	1	[= $67{,}1875_{10}$]
	1	0	0	1	1	1	0	1	,	0	0	0	1	[= $157{,}0625_{10}$]

Entonces,

S = $1001\ 1101.0001_2$ [≡ 157.0625_{10}]

Resta Binaria

TABLA DE LA RESTA
0 - 0 = 0
1 - 0 = 1
1 - 1 = 0

Ejemplo.- Dados los valores binarios de A y B obtener R = A - B.

Donde:

A = 110 1101,1001

B = 101 1110,0101

	0	0	0	0	1	0	0	1	0	0	←	Borrow [Pedir prestado]	
	1	1	0	1	1	0	1	,	1	0	0	1	[= $109{,}5625_{10}$]
-	1	0	0	0	0	1	1	,	0	1	0	1	- [= $67{,}3125_{10}$]
	0	1	0	1	0	1	0	,	0	1	0	0	[= $42{,}2500_{10}$]

Entonces

R = $10\ 1010{,}01_2$ [$42{,}25_{10}$]

Multiplicación Binaria

TABLA DE LA MULTIPLICACIÓN
0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Ejemplo.- Dados los valores binarios de A y B obtener P = A x B.

Donde:

A = 1101,101 [$13{,}625_{10}$]

B = 1010,011 [$10{,}375_{10}$]

				1	1	0	1	,	1	0	1		[= $13{,}625_{10}$]	
x				1	0	1	0	,	0	1	1		[= $10{,}375_{10}$]	
				1	1	0	1		1	0	1			
			1	1	0	1	1		0	1				
		1	1	0	1	1	0	1	0		0			
	1	1	0	1	1	0	1	0						
1	0	0	0	1	1	0	1	,	0	1	0	1	1	1	[= $141{,}359375_{10}$]

Entonces

$$P = 1000\ 1101,0101\ 11_2 \quad [\equiv 141,359372_{10}]$$

División Binaria

TABLA DE LA DIVISIÓN
0 ÷ 1 = 0
1 ÷ 1 = 1

Ejemplo.- Dados los valores binarios de A y B obtener Q = A ÷ B y el Residuo.

$$A = 110\ 0101,101 \quad [101,625_{10}]$$
$$B = \quad 1101,01 \quad [13,25_{10}]$$

```
                ) ) ) ) )
1 1 0 0 1 0 1 1 0 , 1          110101
  1 1 0 1 0 1                  111,101
  ---------
  1 1 0 0 0 0 1
    1 1 0 1 0 1
    ---------
    1 0 1 1 0 0 0
      1 1 0 1 0 1
      ---------
      1 0 0 0 1 1   1
        1 1 0 1 0   1
        -----------
          1 0 0 1   0 0 0
            1 1 0   1 0 1
            -------------
              1 0   0 1 1
```

Entonces

$$Q = 111,101_2 \quad [\equiv 7,625_{10}]$$
$$RESIDUO = 0,10011_2 \quad [\equiv 0,59375_{10}]$$

Finalmente, conviene indicar que ***cualquier operación matemática, simple o compleja, puede resolverse en forma de sumas.***

Representación de Números Bipolares Utilizando Complementos.- La representación por medio de complementos sirve para trabajar con números positivos y negativos, es decir con cantidades bipolares. Para indicar el signo se emplea un dígito adicional. En el caso del sistema de numeración binaria, que es el que se utiliza en las computadoras, generalmente el 0 indica el signo positivo y el 1 el signo negativo [convenio que se usará]. El dígito para el signo ocupa la posición "más significante".

Para trabajar con complementos es necesario establecer un determinado número de dígitos, tanto para la parte entera como para la fraccionaria y, como se mencionó, un bit[1] más para el signo.

La representación de cantidades por medio de complementos facilita la realización de las operaciones aritméticas básicas, puesto que se usan los mismos circuitos sumadores binarios; esto se debe al hecho de que a la resta se la puede implementar

1.- El término bit significa dígito binario, del inglés binary digit.

mediante una suma entre el minuendo y el complemento del substraendo.

En las computadoras, los dispositivos digitales [sumadores, comparadores, registros, etc.] trabajan con un determinado número bien definido de dígitos binarios, por tanto conviene acostumbrarse a representar las cantidades binarias con un mismo número de cifras. Por ejemplo, para un microprocesador [CPU] de 8-bits, el 0_{10} se escribe como

$$0_{10} \equiv 0000\ 0000_2,$$

el 1_{10}, como

$$1_{10} \equiv 0000\ 0001_2,$$

el 127_{10}, como

$$127_{10} \equiv 0111\ 1111_2.$$

En los sistemas de numeración existen dos tipos de complemento, que se los utiliza con mucha frecuencia: ***Complemento Restringido*** y ***Complemento Verdadero***.

Complemento Restringido (a B-1).- El Complemento Restringido[2] de un número se encuentra indicando

2.- En el sistema binario, al Complemento Restringido [a B-1] se lo denomina Complemento a 1 y en decimal, Complemento a 9.

primero, con cuantos dígitos efectivos se va a trabajar, luego se agrega el dígito del signo. ***Generalmente se usa el 0 para indicar una cantidad positiva y (B - 1) para indicar una cantidad negativa.*** Una vez realizado este proceso, al número así obtenido se lo resta de [B - 1]s, tantos como dígitos tenga la nueva representación del número. En el caso del sistema de numeración binario [base 2], se resta de 1s [2 - 1]s, como puede verse a continuación.

Ejemplo: Encontrar el complemento restringido [complemento a 1] del número 11 1001.0110 1_2. Considere que se va a trabajar con 11 dígitos para la parte entera 8 dígitos para la parte fraccionaria y el dígito adicional para el signo.

Signo
↕
⇨ + 11 1001,0110 1 = **0**000 0011 1001,0110 1000
___ valor numérico ___

Observe que la parte entera del número original solamente tiene 6-bits, por lo que es necesario completar con 5 ceros a la izquierda; de la misma manera, la parte fraccionaria se completa con los 0s necesarios hacia la derecha, a esto hay que agregar el bit del signo, que es el 0 que está al extremo izquierdo. Por claridad se han realizado agrupaciones de 4-bits.

A continuación, se procede a restar el número así obtenido de un valor formado por tantos 1s como bits tenga el nuevo número.

```
   Signo
   ↕
    | 1 | 1 1 1  1 1 1 1  1 1 1 1 , 1 1 1 1  1 1 1 1
  - | 0 | 0 0 0  0 0 1 1  1 0 0 1 , 0 1 1 0  1 0 0 0
  ------------------------------------------------------
    | 1 | 1 1 1  1 1 0 0  0 1 1 0 , 1 0 0 1  0 1 1 1
      ____Dígitos correspondientes al valor numérico_____
```

En este caso, el bit del extremo izquierdo de la respuesta, indica que el resultado es un número con signo negativo, es decir

$$\overset{\mathbf{S}}{+\ 11\ 1001{,}0110\ 1} = \mathbf{0}000\ 0011\ 1001{,}0110\ 1000 \quad [\equiv +57{,}40625_{10}]$$

$$\overset{\mathbf{S}}{-\ 11\ 1001{,}0110\ 1} = \mathbf{1}111\ 1100\ 0110{,}1001\ 0111 \quad [\equiv -57{,}40625_{10}]$$

es la representación de los números positivo y negativo en complemento a 1 respectivamente.

Una forma fácil [algoritmo] para obtener el complemento a 1 de un número binario es: primero completar el número de bits requerido, y luego cambiar los 0s por 1s y los 1s por 0s. Por ejemplo, para obtener el complemento a 1 de: A = 1010 1101,1001, con el número de bits del ejemplo anterior, se tendrá

$$\overset{\mathbf{S}}{+\ A} = \mathbf{0}000\ 1010\ 1101{,}1001\ 0000,$$

entonces su complemento a 1 será

$$\overset{\mathbf{S}}{-\ A} = \mathbf{1}111\ 0101\ 0010{,}0110\ 1111.$$

Una aplicación práctica de la representación de cantidades usando complementos, es en operaciones de sustracción puesto que se la puede convertir a suma, si previamente se obtiene el complemento del sustraendo. Para realizar la operación

$$\mathbf{R = A - B,}$$

se obtiene el complemento de B, que se representará como $\mathbf{B^*}$, entonces

$$\mathbf{R = A - B = A + (-B) = A + B^*}$$

puesto que $\mathbf{B^* = -B}$, representa el complemento de B.

Sustracción con Complemento a 1.- Los siguientes ejemplos ilustrarán la metodología que se debe seguir cuando se trabaja con complemento a 1.

Ejemplo 1.- Mediante el uso del complemento a 1, realice la operación A - B, con los siguientes datos.

A = 111 0110,101

B = 100 1100,10

111 0110,101 - 100 1100,10

$[118{,}625 - 76{,}5]_{10}$.

Como se indicó, es necesario que el minuendo y el substraendo tengan el mismo número de dígitos, tanto para la parte entera como para la fraccionaria y que se añada un bit para el signo. En este ejemplo se utilizarán 11-bits para la parte entera, ocho para la fraccionaria y 1 para el signo, de manera que las cantidades originales tendrían la siguiente representación.

S

+ A = 111 0110,101 = **0**000 0111 0110,1010 0000

+ B = 100 1100,10 = **0**000 0100 1100,1000 0000

ahora, se debe sacar el complemento a 1 del substraendo, como se indicó anteriormente.

S

B* = - B = **1**111 1011 0011,0111 1111

Luego se procede a realizar la suma entre el minuendo y el complemento a 1 del substraendo.

Signo

↕

	0	0 0 0 0 1 1 1 1 0 1 1 0 , 1 0 1 0 0 0 0 0
-	**1**	1 1 1 1 0 1 1 1 0 0 1 1 , 0 1 1 1 1 1 1 1 1
Exceso → 1	**0**	0 0 0 0 0 0 1 0 1 0 1 0 , 0 0 0 1 1 1 1 1

[Carry]

Dígitos correspondientes al valor numérico

42,121093751

Se observa la generación de un exceso [carry], también se ve que la respuesta no es exacta. Para generar la respuesta correcta, es necesario sumar el exceso, que se formó, al bit menos significante del resultado previo, como se indica a continuación.

S

	0	0 0 0 0 0 0 1 0 1 0 1 0 , 0 0 0 1 1 1 1 1
†		1
	0	0 0 0 0 0 0 1 0 1 0 1 0 , 0 0 1 0 0 0 0 0

$[42{,}125_{10}]$

Este nuevo valor sí corresponde al resultado exacto de la resta pedida. Este procedimiento, de sumar el exceso al dígito menos significante, debe seguirse cada vez que se genere un carry al realizar operaciones con complemento restringido.

Al analizar la respuesta de este ejemplo, se ve que el bit del signo es 0, lo que implica un valor positivo como era de esperarse al restar un número menor de uno mayor.

En el siguiente ejemplo, se estudia el caso de restar una cantidad mayor de otra menor.

Ejemplo 2.- Realice la siguiente operación:

1010 1101,0011 - 1 1101 0001,101,

$[173{,}1875 - 465{,}625]_{10}$

utilice complementos a 1, 11-bits para la parte entera, 8-bits para la parte fraccionaria y el bit del signo.

```
                                S
                                ↕
+A = +   1010 1101,0011 = 0000 1010 1101,0011 0000
+B = + 1 1101 0001,101  = 0001 1101 0001,1010 0000

B* = - 1 1101 0001,101  = 1110 0010 1110,0101 1111
```

Como siempre, la respuesta se obtiene sumando el minuendo con el complemento a 1 del substraendo.

	S																			
	0	0	0	0	1	0	1	0	1	1	0	1	,	0	0	1	1	0	0	0	0
†	**1**	1	1	0	0	0	1	0	1	1	1	0	,	0	1	0	1	1	1	1	1
	1	1	1	0	1	1	0	1	1	0	1	1	,	1	0	0	0	0	1	1	

Como puede verse, no se ha generado un carry. El bit del signo es 1, lo que implica un resultado negativo. Cuando se resta un número mayor de uno menor, usando complementos, se genera un carry = 0. Como el resultado es negativo, para obtener la magnitud de la respuesta, es necesario sacar nuevamente el complemento de la respuesta, entonces

```
S
1110 1101 1011,1000 1111 ≡ -001 0010 0100,0111 0000
```


El equivalente decimal de la respuesta sería

$$-292,43751$$

En este caso, ya no es necesario hacer el ajuste para obtener la respuesta exacta.

Complemento Verdadero (a B).- Para obtener el Complemento Verdadero[3] de un número se procede de un modo similar que para obtener el complemento restringido. Es decir, se trabaja con un número definido de dígitos para la parte entera y para la fraccionaria a más del dígito del signo que sigue siendo [B - 1] y que se escribe en el extremo izquierdo del número [dígito más significante]. En el caso de Complemento Verdadero, la resta se realiza de un 1 seguido de tantos 0s como dígitos tenga el nuevo número; el 1 se lo escribe antes de la columna del signo.

Ejemplo.- Obtener el complemento verdadero [complemento a 2] del número binario $A = 11\ 0101.01_2$. Trabaje con 7-bits para la parte entera, 4-bits para la parte fraccionaria y 1-bit para el signo.

3.- En el sistema binario, al Complemento Verdadero [a B] se lo denomina Complemento a 2 y en decimal, Complemento a 10.

S

⇨ +A = + 11 0101,01 = **0**011 0101,0100

Observe que la parte entera del número original solamente tiene 6-bits, por lo que es necesario completar con 0s a la izquierda; de la misma manera, la parte fraccionaria se completa con los 0s necesarios hacia la derecha, a esto hay que agregar el bit del signo, que es el 0 que está al extremo izquierdo. Después se procede a restar el número así obtenido de un valor formado por tantos 0s como bits tenga el nuevo número a los que se agrega un 1 al extremo izquierdo, como se muestra en seguida.

	S	
1	**0**	0 0 0 0 0 0 0 0 , 0 0 0 0
-	**0**	0 1 1 0 1 0 1 , 0 1 0 0
	1	1 0 0 1 0 1 0 , 1 1 0 0
		_____ valor numérico _____

En este caso el bit del extremo izquierdo, indica que el resultado es un número con signo negativo, es decir

S

+ 11 0101,01 = **0**011 0101,0100 [= + $53{,}25_{10}$]
- 11 0101,01 = **1**100 1010,1100 [= - $53{,}25_{10}$]

es la representación de los números positivo y

negativo en complemento a 2 respectivamente.

Otra forma de conseguir el complemento a 2 de un número binario es obtener, en primer lugar, el complemento a 1 del número y luego sumar 1 al bit menos significante [al bit del extremo derecho]. También puede sacarse el complemento a 2 de un número binario, empezando por el extremo derecho [menos significante]: se copian todos los 0s hasta encontrar el primer 1 que también se lo copia, a partir de ese punto todos los demás dígitos se complementan uno a uno [es decir, se cambian los 0s por 1s y los 1s por 0s].

Aritmética con Complemento Verdadero.- Al igual que en el caso del complemento a 1, el complemento a 2 puede emplearse para convertir una operación de sustracción en una de suma, si previamente se obtiene el complemento a 2 del substraendo. De manera que, para realizar la operación

$$\mathbf{R = A - B,}$$

se obtiene el complemento a 2 de B, que también se representará como B^*, entonces

$$\mathbf{A - B = R = A + B^*}$$

puesto que $\mathbf{B^* = -B}$, representa el complemento a 2 de B.

Ejemplo 1.- Realice la operación A - B, usando complemento a 2. Emplee 10-bits para la parte entera, 4-bits para la parte fraccionaria y uno para el signo. Los valores de A y B se indican en el ejemplo.

A = 1110 0001,1011 [= $225,6875_{10}$]
B = 1101 0000,1101 [= $208,8125_{10}$]

⇛ + A = **0**00 1110 0001,1011 [= + $225,6875_{10}$]
+ B = **0**00 1101 0000,1101 [= + $208,8125_{10}$]

⇛ B^* = **1**11 0010 1111,0011 [= - $208,8125_{10}$]

La sustracción, usando complemento a 2 se realiza sumando el minuendo con el complemento a 2 del substraendo, como se observa a continuación.

	S															
A =	**0**	0	0	1	1	1	0	0	0	0	1	,	1	0	1	1
+ B* =	**1**	1	1	0	0	1	0	1	1	1	1	,	0	0	1	1
Exceso → 1	**0**	0	0	0	0	0	1	0	0	0	0	,	1	1	1	0
se deshecha																

Respuesta = $16,875_{10}$

En el caso del trabajar con complemento verdadero, el dígito del carry se deshecha. Esto simplifica el proceso aritmético. Debido a esto, el complemento

a 2 es el más utilizado en las computadoras digitales.

En el ejemplo anterior, se ve que el bit del signo es 0, lo que implica un resultado positivo. Si se tuviera un resultado negativo [bit del signo igual a 1], habría que obtener el complemento a 2 del resultado para conocer su magnitud, como se estudia en el siguiente ejemplo.

Ejemplo 2.- Realice la operación A - B, usando complemento a 2. Emplee 11-bits para la parte entera, 4-bits para la parte fraccionaria y uno para el signo. Los valores de A y B se indican a continuación.

A = 110 1001,0011 [= $105,1875_{10}$]
B = 1011 0110,1001 [= $182,5625_{10}$]
⇛ + A = **0**000 0110 1001,0011 [= + $105,1875_{10}$]
+ B = **0**000 1011 0110.1001 [= + $182,5625_{10}$]

⇛ B^* = **1**100 0100 1001,0111 [= - $182,5625_{10}$]

La sustracción, usando complemento a 2 se realiza sumando el minuendo con el complemento a 2 del substraendo, como se indica a continuación.

	S															
A =	**0**	0	0	0	1	1	0	1	0	0	1	,	0	0	1	1
+ B* =	**1**	1	1	1	0	1	0	0	1	0	0	1	,	0	1	1	1
	1	1	1	1	1	0	1	1	0	0	1	0	,	1	0	1	0

Puesto que el dígito del signo es 1, la respuesta es negativa, si se quiere obtener la respuesta en decimal con signo, es necesario sacar el complemento a 2 de la respuesta binaria, como se ve en seguida.

$R = -000\ 0100\ 1101,0110_2$, o lo que es lo mismo

Respuesta = $-77,375_{10}$

Ejemplo 3.- Con los siguientes datos binarios realice la operación aritmética indicada. Todo el proceso debe realizarlo en complemento a 2, únicamente el resultado final convertirlo a decimal.

Datos:

A = 1 1 1 0 1 1 0 1 , 1 0 0 1

B = 1 1 0 0 1 1 0 1 1 0 1 , 1 0 1

C = 1 1 0 1 1 1 0 1 1 0 , 1 1 1

D = 1 1 0 1 0 1 0 0 1 1 , 1 0 0 1

Operación aritmética

$R_1 = (D - A)$ $R_2 = (C - B)$ $R = R_1 - R_2$

Se utilizarán 13-bits para la parte entera, 5-bits para la parte fraccionaria y 1-bit para el signo. Entonces

	S																
+ A =	0	0	0	0	0	0	1	1	1	0	1	1	0	1	,	1	0	0	1	0
+ B =	0	0	0	1	1	0	0	1	1	0	1	1	0	1	,	1	0	1	0	0
+ C =	0	0	0	0	1	1	0	1	1	1	0	1	1	0	,	1	1	1	0	0
+ D =	0	0	0	0	1	1	0	1	0	1	0	0	1	1	,	1	0	0	1	0

	S																
- A =	1	1	1	1	1	1	0	0	0	1	0	0	1	0	,	0	1	1	1	0
- B =	1	1	1	0	0	1	1	0	0	1	0	0	1	0	,	0	1	1	0	0

Entonces

	S																
+ D =	0	0	0	0	1	1	0	1	0	1	0	0	1	1	,	1	0	0	1	0
- A =	1	1	1	1	1	1	0	1	1	0	0	1	1	0	,	0	1	1	1	0
R_1 =	1	0	0	0	0	1	0	0	1	0	0	1	1	0	,	0	0	0	0	0

	S																
+ C =	0	0	0	0	1	1	0	1	1	1	0	1	1	0	,	1	1	1	0	0
- B =	1	1	1	0	0	1	1	0	0	1	0	0	1	0	,	0	1	1	0	0
R_2 =	0	1	1	1	1	0	1	0	0	0	0	1	0	0	1	,	0	1	0	0	0

puesto que este resultado parcial [R_2] es negativo para realizar la operación correctamente hay que volver a obtener el complemento a 2 de este valor. Es decir

S

$R_2 = 11\ 1101\ 0000\ 1001,0100\ 0$

por tanto,

R2 = 00 0010 1111 0110,1100 0

finalmente

	S																	
$+ R_1 =$	0	0	0	0	1	0	0	1	1	0	0	1	1	0	,	0	0	0	0	0
$+ R_2 =$	0	0	0	0	1	0	1	1	1	1	0	1	1	0	,	1	1	0	0	0
$R =$ 1	0	0	0	1	0	1	0	1	0	1	1	1	0	0	,	1	1	0	0	0

La respuesta binaria es

$$\text{Respuesta} = +000\ 0101\ 0101\ 1100{,}1100\ 0_2$$

y en decimal

$$R_{10} = 2^{10} + 2^8 + 2^6 + 2^4 + 2^3 + 2^2 + 2^{-1} + 2^{-2}$$

$$R_{10} = 1024 + 256 + 64 + 16 + 8 + 4 + 0{,}5 + 0{,}25 = +1372{,}75$$

$$\text{Respuesta} = +1372{,}75_{10}$$

Códigos de Numeración Binaria.- La representación de cantidades por medio de algún arreglo de dígitos se denomina ***número, código*** o ***"palabra"***. En el sistema de numeración binaria existen varias formas de

codificar o de representar cantidades. A continuación se muestran las más comunes.

Código Binario Natural.- En este código, los bits a la izquierda del punto se denominan enteros y los de la derecha fraccionarios. Las ponderaciones son positivas y ascendentes hacia la izquierda a partir del punto y negativas y descendentes hacia la derecha del punto. La siguiente tabla muestra los números enteros de 4-bits [binario] con sus equivalentes en: octal, decimal, hexadecimal, BCD, EXC-a-3 y GRAY observe que en BCD existen 6-códigos binarios que no se utilizan.

Otros Códigos Binarios.- El Binario Natural es el código más comúnmente usado; sin embargo, existen otras formas de codificar la información, dependiendo del procesamiento que se le dará a la misma.

BIN	OCT	DEC	HEX	BCD	EXC-3	GRAY
0 0 0 0	0	0	0	0 0 0 0	0 0 1 1	0 0 0 0
0 0 0 1	1	1	1	0 0 0 1	0 1 0 0	0 0 0 1
0 0 1 0	2	2	2	0 0 1 0	0 1 0 1	0 0 1 1
0 0 1 1	3	3	3	0 0 1 1	0 1 1 0	0 0 1 0
0 1 0 0	4	4	4	0 1 0 0	0 1 1 1	0 1 1 0
0 1 0 1	5	5	5	0 1 0 1	1 0 0 0	0 1 1 1
0 1 1 0	6	6	6	0 1 1 0	1 0 0 1	0 1 0 1
0 1 1 1	7	7	7	0 1 1 1	1 0 1 0	0 1 0 0
1 0 0 0	10	8	8	1 0 0 0	1 0 1 1	1 1 0 0

1 0 0 1	11	9	9	1 0 0 1	1 1 0 0	1 1 0 1
1 0 1 0	12	10	A	1 0 0 0 0	1 0 0 0 0 1 1	1 1 1 1
1 0 1 1	13	11	B	1 0 0 0 1	1 0 0 0 1 0 0	1 1 1 0
1 1 0 0	14	12	C	1 0 0 1 0	1 0 0 0 1 0 1	1 0 1 0
1 1 0 1	15	12	D	1 0 0 1 1	1 0 0 0 1 1 0	1 0 1 1
1 1 1 0	16	13	E	1 0 1 0 0	1 0 0 0 1 1 1	1 0 0 1
1 1 1 1	17	15	F	1 0 1 0 1	1 0 0 1 0 0 0	1 0 0 0

Decimal Codificado en Binario (BCD).- En el código BCD [**B**inary **C**oded **D**ecimal = Decimal Codificado en Binario], cada dígito decimal está representado por un grupo de 4-bits, a esta agrupación se la denomina ***"quad"***. Cada quad tiene 4-bits [con ponderaciones: 8, 4, 2 y 1] con 10 valores permisibles de 0 a 9. En la codificación BCD, los quads con valores superiores a 9 [1010, 1011, 1100, 1101, 1110, 1111] no están permitidos, por tanto, nunca se usan en BCD. De modo que para representar el número 12_{10} en BCD sería 1 0010_{BCD}. Al código BCD se lo utiliza principalmente en diferentes tipos de medidores de panel, por ejemplo en voltímetros digitales.

Código Exceso de 3.- Puede decirse que el ***código exceso de 3*** es una modificación del código BCD, puesto que el primero se forma añadiendo 3 al código BCD. Eventualmente se lo utiliza en lugar del BCD debido a que posee ventajas en algunas operaciones aritméticas. La tabla anterior muestra el código exceso de 3 y su equivalente BCD.

Código de Gray [Reflejado].- Es un código binario en el que la posición del bit no tiene significación numérica [ponderación]; sin embargo, cada código de Gray corresponde a un mismo número decimal. Fácilmente se lo puede transformar a su equivalente binario. En la tabla anterior se presentan los códigos de Gray y binario natural para los números del 0 hasta el 15. Después se hace una comparación entre los dos códigos para determinar las relaciones que permitan convertir el uno en el otro y viceversa.

Como puede verse en esta tabla, en el código de Gray, cuando el valor de un número cambia, la transición de un código al siguiente implica el cambio de un solo dígito a la vez.

Por observación de la tabla, puede decirse que la ***conversión del código de Gray al código binario*** se realiza de la siguiente manera: El bit correspondiente al extremo izquierdo ["MSB"] es el mismo tanto en el código de Gray como en el binario; al continuar hacia la derecha, si el siguiente bit de Gray es "1", entonces el próximo bit binario es el complemento del anterior bit binario. Pero si el siguiente bit de Gray en "0", entonces el próximo bit binario es la copia del bit binario anterior.

Ejemplo: 1010 [Gray] ⇛ 1100 [binario]

$$1110\ 0110\ 0011_{CG} \Rightarrow 1011\ 1011\ 1101_{CB}$$

De igual manera, la ***conversión de código binario a código de Gray*** puede deducirse a partir de la tabla anterior. El MSB binario es el mismo "MSB" de Gray; continuando la lectura hacia la derecha, cada cambio en el código binario produce un "1" y cada no cambio produce un "0" en el código de Gray.

Ejemplo: $1011_{CB} \Rightarrow 1110_{CG}$

$1110\ 0101\ 1000_{CB} \Rightarrow 1001\ 0111\ 0100_{CG}$

El código de Gray es útil en aquellas aplicaciones en las que pueden presentarse códigos intermedios falsos, que podrían ocurrir en otros códigos.

Códigos Bipolares.- Existe una gran variedad de códigos binarios, entre otros: Signo-Magnitud, Complemento a 1, Complemento a 2, Binario Desplazado [Offset], Todo Complementado, etc. Estos códigos sirven para representar cantidades tanto positivas como negativas [para lo cual un dígito representa el signo y los otros la "***magnitud***" del número]. Los códigos bipolares más comunes [para 4-bits incluido el signo] se indican en la siguiente tabla.

VALOR DECIMAL	SIGNO MAGNITUD	BINARIO OFFSET	COMPLEMENTO a-1	COMPLEMENTO a-2
7	0 1 1 1	1 1 1 1	0 1 1 1	0 1 1 1
6	0 1 1 0	1 1 1 0	0 1 1 0	0 1 1 0
5	0 1 0 1	1 1 0 1	0 1 0 1	0 1 0 1
4	0 1 0 0	1 1 0 0	0 1 0 0	0 1 0 0
3	0 0 1 1	1 0 1 1	0 0 1 1	0 0 1 1
2	0 0 1 0	1 0 1 0	0 0 1 0	0 0 1 0
1	0 0 0 1	1 0 0 1	0 0 0 1	0 0 0 1
0	0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 0
0	1 0 0 0	1 0 0 0	1 1 1 1	0 0 0 0
-1	1 0 0 1	0 1 1 1	1 1 1 0	1 1 1 1
-2	1 0 1 0	0 1 1 0	1 1 0 1	1 1 1 0
-3	1 0 1 1	0 1 0 1	1 1 0 0	1 1 0 1
-4	1 1 0 0	0 1 0 0	1 0 1 1	1 1 0 0
-5	1 1 0 1	0 0 1 1	1 0 1 0	1 0 1 1
-6	1 1 1 0	0 0 1 0	1 0 0 1	1 0 1 0
-7	1 1 1 1	0 0 0 1	1 0 0 0	1 0 0 1
-8		0 0 0 0		1 0 0 0

Los códigos Signo-Magnitud y Binario Offset conceptualmente son simples, pero representan dificultades al querer implementarlos en software. Mucho más fácil es implementar los códigos Complemento a-1 y Complemento a-2, que son los más usados en las computadoras. El código signo-magnitud y el complemento a 1 tienen dos códigos binarios para representar el valor decimal 0, lo que constituye un problema.

Códigos Alfanuméricos.- Son códigos que sirven para

representar caracteres tanto numéricos como alfabéticos, en los que también se incluyen los códigos correspondientes a los signos de puntuación, de control y otros: ¡, ¿, #, $, %, /, &, *, (,), _, -, +, <, >, etc. Uno de ellos es el código **EBCDIC** [**E**xtended **B**inary-**C**oded **D**ecimal **I**nterchange **C**ode]. Es un código que usa 8 dígitos binarios para representar un carácter simple, dando un máximo posible de 256 caracteres. Es utilizado como un sistema de código en muchos computadores. El código EBCDIC es simplemente el código BCD extendido a 8-bits.

Asignación de Códigos EBCDIC

Parte I

HEX	BITS				MSD→	0	1	2	3	4	5	6	7
					b7	0	0	0	0	0	0	0	0
					b6	0	0	0	0	1	1	1	1
					b5	0	0	1	1	0	0	1	1
LSD↓	b3	b2	b1	b4 / b0		0	1	0	1	0	1	0	1
0	0	0	0	0		NUL	DLE	DS		SP	&	-	
1	0	0	0	1		SOH	DC1	SOS					
2	0	0	1	0		STX	DC2	FS	SYN				
3	0	0	1	1		ETX	DC3						
4	0	1	0	0		PF	RES	BYP	PN				
5	0	1	0	1		HT	NL	LF	RS				
6	0	1	1	0		LC	BS	EOB ETB	UC				
7	0	1	1	1		DEL	IL	PRE ESC	EOT				
8	1	0	0	0			CAN						
9	1	0	0	1		RLF	EM						\
A	1	0	1	0		SMM	CC	SM		¢	!	|	:
B	1	0	1	1		VT				.	$	'	#
C	1	1	0	0		FF	IFS		DC4	<	*	%	@
D	1	1	0	1		CR	IGS	ENQ	NAK	(	)	_	´
E	1	1	1	0		SO	IRS	ACK		+	;	>	=
F	1	1	1	1		SI	IUS	BEL	SUB	|	¬	?	"

Caracteres de Comando

NUL	Null	PF	Punch Off
SOH	Start of Heading	HT	Horizontal Tab
STX	Start of Text	LC	Lower Case
ETX	End of Text	DEL	Delete

RLF	Reverse Line Feed
SMM	Start of Manual Message
VT	Vertical Tabulation
FF	Form Feed
CR	Carriage Return
SO	Shift Out
SI	Shift In
DLE	Data Link Escape
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
RES	Restore
NL	New Line
BS	Backscape
IL	Idle
CAN	Cancel
EM	End of Medium
CC	Cursor Control
IFS	Interchange File Separator
IGS	Interchange Group Separator
IRS	Interchange Record Separator
IUS	Interchange Unit Separator
DS	Digit Select
SOS	Start of Significance
FS	Field Separator
BYP	Bypass
LF	Line Feed
EOB/ETB	End of Block/End of Transmission Block
PRE/ESC	Prefix/Escape
SM	Set Mode
ENQ	Enquiry
ACK	Ackmowledge
BEL	Bell
SYN	Synchronous Idle
PN	Pench On
RS	Reader Stop
UC	Upper Case
EOT	End of Transmission
DC4	Device Control 4
NAK	Negative Acknowledge
SUB	Substitute
SP	Space

Asignación de Códigos EBCDIC

Parte II

HEX				MSD→	8	9	A	B	C	D	E	F
				b7	1	1	1	1	1	1	1	1
	BITS			b6	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1
LSD↓	b3	b2	b1	b0 \ b4	0	1	0	1	0	1	0	1
0	0	0	0	0					{	}	\	0
1	0	0	0	1	a	j	~		A	J		1
2	0	0	1	0	b	k	s		B	K	S	2
3	0	0	1	1	c	l	t		C	L	T	3
4	0	1	0	0	d	m	u		D	M	U	4
5	0	1	0	1	e	n	v		E	N	V	5
6	0	1	1	0	f	o	w		F	O	W	6
7	0	1	1	1	g	p	x		G	P	X	7
8	1	0	0	0	h	q	y		H	Q	Y	8
9	1	0	0	1	i	r	z		I	R	Z	9
A	1	0	1	0								
B	1	0	1	1								
C	1	1	0	0								
D	1	1	0	1								
E	1	1	1	0								
F	1	1	1	1								

bits

7654 3210

Ej. Código de la letra N = 1101 0101 = D5H

Ejemplo.- Encuentre el código EBCDIC [HEX] del siguiente texto: Politécnica Nacional.

P	o	l	i	t	e	c	n	i	c	a	
D7	96	93	89	A3	85	83	95	89	83	81	40

N	a	c	i	o	n	a	l	.
D5	81	83	89	96	95	81	93	4B

Otro código alfanumérico de 7-bits, muy utilizado por la mayoría de fabricantes de computadoras, es el **ASCII** [**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange], cuya tabla se muestra a continuación.

Asignación de Códigos ASCII

HEX				MSD →	0	1	2	3	4	5	6	7
				b7	0	0	0	0	0	0	0	0
	BITS			b6	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1
LSD↓	b3	b2	b1	b4 / b0	0	1	0	1	0	1	0	1
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
1	0	0	0	1	SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	STX	DC2	"	2	B	R	b	r
3	0	0	1	1	ETX	DC3	#	3	C	S	c	s
4	0	1	0	0	EOT	DC4	$	4	D	T	d	t
5	0	1	0	1	ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0	ACK	SYN	&	6	F	V	f	v
7	0	1	1	1	BEL	ETB	'	7	G	W	g	w
8	1	0	0	0	BS	CAN	(	8	H	X	h	x
9	1	0	0	1	HT	EM	)	9	I	Y	i	y
A	1	0	1	0	LF	SUB	*	:	J	Z	j	z
B	1	0	1	1	VT	ESC	+	;	K	[	k	{
C	1	1	0	0	FF	FS	,	<	L	\	l	|
D	1	1	0	1	CR	GS	-	=	M	]	m	}
E	1	1	1	0	SO	RS	.	>	N	^	n	~
F	1	1	1	1	SI	US	/	?	O	_	o	DEL

Caracteres de Comando

NUL	Null, or all zeros
SOH	Start of Heading
STX	Start of Text
ETX	End of Text
EOT	End of Transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell (audible or attention signal)
BS	Backspace
HT	Horizontal Tabulation (punched

	card skip)
LF	Line Feed
VT	Vertical Tabulation
FF	Form Feed
CR	Carriage Return
SO	Shift Out
SI	Shift In
DLE	Data Link Escape
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4 (stop)
NAK	Negative Acknowledge
SYN	Synchronic Idle
ETB	End of Transmission Block
CAN	Cancel
EM	End of Medium
SUB	Substitute
ESC	Escape
FS	File Separator
GS	Group Separator
RS	Record Separator
US	United Separator
DEL	Delete
SP	Space

Ejemplo.- Encuentre el código ASCII [HEX] del siguiente texto: Politécnica Nacional.

P	o	l	i	t	e	c	n	i	c	a	
50	6F	6C	69	74	65	63	6	69	63	61	20

N	a	c	i	o	n	a	l	.
4	61	63	69	6F	6	61	6C	2

D:\~\SD_Cpas\SD-Cap1Col.wpd

Revisión: Septiembre - 2008

Álgebra de Boole

El Álgebra de Boole utiliza variables que tienen solo dos valores posibles, esto lo sintetizó Shannon usando ideas que inicialmente las expresó el matemático inglés: George Boole[1]. A diferencia de las variables del álgebra común [que pueden tomar un número infinito de valores en un rango determinado], una variable booleana, por ejemplo A, puede tomar solamente 2 valores, que generalmente se los relaciona con **VERDADERO** y **FALSO**. Sin embargo, se les puede asignar otros valores, tal como: caliente/frío, macho/hembra, alto/bajo, etc. Para representar los 2 posibles valores de las variable booleanas se utilizan los símbolos 0 y 1. Generalmente **A = 1** significa que **A** es *VERDADERO* en un sentido booleano, mientras que **A = 0** indica que **A** es *FALSO*. Entonces una variable booleana puede estar relacionada a algún ítem de información, por ejemplo, A = 1, significa que un interruptor asociado con A está abierto y A = 0 significa que el mismo interruptor está cerrado. Otra variable, B, puede relacionarse a la temperatura de una habitación, siendo VERDADERA cuando la temperatura exceda los 21°C y FALSA en otro caso o viceversa.

1.- George Boole, matemático inglés del siglo XIX, inventó el álgebra binaria o lógica que lleva su nombre: "Álgebra booleana".

Las variables booleanas no toman valores cuantitativos, pero pueden usarse para representar información cuantitativa. Por ejemplo, se pueden usar 4-variables booleanas para representar un número binario de 4-dígitos. Cada variable puede estar relacionada a uno de los coeficientes del número binario, indicando que el coeficiente tiene un valor de 1 cuando la variable es VERDADERA y un valor 0 cuando es FALSA [o el inverso de esto]. De esta manera las 16 posibles combinaciones pueden estar relacionadas a las cantidades $0\text{-}15_{10}$, que puede tomar el número binario. Conociendo los valores VERDADERO/FALSO de cada una de las variables, posibilitará el cálculo de la cantidad que ella representa. Para trabajar con variables booleanas, se utilizan operadores similares a los del álgebra común. A estos operadores booleanos comúnmente se los conoce como ***conectivos lógicos***.

Proposiciones y Conectivos Lógicos

Proposición	Planteamiento de un teorema o de un problema que se debe demostrar o resolver.
Premisa	Supuesto material, no necesariamente válido lógicamente, a partir del que se infiere una conclusión.
Conectivo	Son los operadores [o compuertas] del álgebra de Boole, similares a los del álgebra común, y representan a los circuitos digitales más fundamentales. En este capítulo se describe su operación mediante el uso del álgebra de Boole. Se estudia cómo pueden combinarse entre sí varias compuertas para implementar circuitos lógicos más complejos.

Variable Booleana	Las variables booleanas sólo pueden tomar dos valores lógicos: "0" o "1". En un circuito lógico, una variable booleana puede representar ausencia o presencia de voltaje. En una proposición lógica, la variable booleana puede ser falsa o verdadera. En general sólo tienen dos opciones posibles.

A continuación se dan algunos ejemplos de variables booleanas.

0_L	1_L
FALSO	VERDADERO
BAJO	ALTO
ABIERTO	CERRADO
ARRIBA	ABAJO
APAGADO	ENCENDIDO
FRÍO	CALIENTE
NOCHE	DÍA
DESCONECTADO	CONECTADO
SIN VOLTAJE	CON VOLTAJE
NEGATIVO	POSITIVO
NO	SI

Conectivo AND [Conjunción "Y"].- Sirve para unir dos o más proposiciones que pueden ser verdaderas o falsas, por ejemplo, sea la proposición compuesta:

Y = Somos estudiantes de la EPN y asistimos a la clase de Sistemas Digitales

Para analizar cuándo la proposición Y es verdadera o cuándo es falsa, se la divide en dos subpro-

posiciones: A y B.

A = Somos estudiantes de la EPN
B = Asistimos a la clase de Sistemas Digitales

Con la ayuda de la siguiente tabla, se puede determinar cuando la proposición Y es verdadera o falsa.

B	A	Y
FALSO	FALSO	FALSO
FALSO	VERDADERO	FALSO
VERDADERO	FALSO	FALSO
VERDADERO	VERDADERO	VERDADERO

B	A	Y
F	F	F
F	V	F
V	F	F
V	V	V

TABLA DE VERDAD DEL CONECTIVO AND DE 2-ENTRADAS

En la tabla anterior, si no somos estudiantes de la EPN, entonces la proposición A es FALSA y si no asistimos a la clase de Sistemas Digitales, la proposición B también es FALSA, por tanto Y es FALSA. De igual manera, si somos estudiantes de la EPN, A es VERDADERA, si no asistimos a la clase de Sistemas Digitales, B es FALSA, entonces Y es FALSA. Si no somos estudiantes de la EPN, A es FALSA, si asistimos a la clase de Sistemas Digitales, B en VERDADERA, pero Y sigue siendo FALSA. Finalmente si somos estudiantes de la EPN, A es VERDADERA; si asistimos a la clase de Sistemas Digitales, B es VERDADERA, por tanto Y es VERDADERA.

El conectivo AND implica que una proposición es

VERDADERA cuando todas las subproposiciones que la conforman son VERDADERAS.

La conjunción es la proposición de que A y B son ciertos. A los conectivos lógicos se los puede implementar físicamente de diferentes maneras, entre otras con interruptores y recibe el nombre de **compuerta AND**, cuyo circuito se muestra en la fig. 1.5 (a) y los símbolos lógicos utilizados en las representaciones esquemáticas en la fig. 1.5 (b). La fig. 1.5 c) muestra el símbolo IEEE del CI-7400, junto con la tabla de verdad. Si a una respuesta FALSA se le asigna el valor lógico 0 [0_L] y a una respuesta VERDADERA se le asigna el valor lógico 1 [1_L], la tabla anterior puede escribirse como se muestra en la siguiente tabla, que es la forma más común de presentar una tabla de verdad. Cuando se hace así, esta tabla puede relacionarse con un producto lógico [no producto aritmético] y la proposición Y puede expresarse así

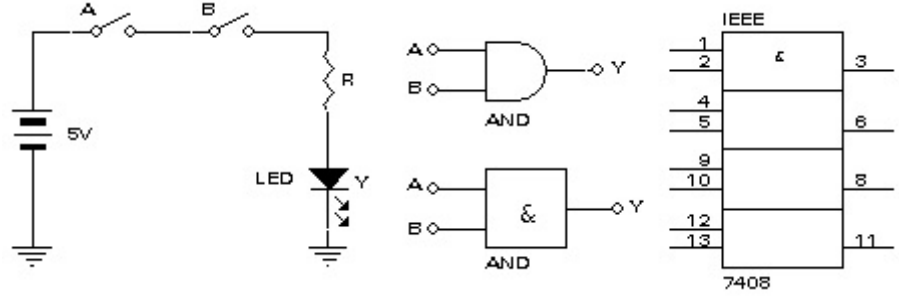


FIGURA 1.5 a) b) c)

B	A	Y
0_L	0_L	0_L
0_L	1_L	0_L
1_L	0_L	0_L
1_L	1_L	1_L

TABLA DE VERDAD DEL CONECTIVO AND PARA 2-ENTRADAS

$$Y = A.B = AB = A \wedge B = A \cap B$$

En el circuito de la fig. 1.5 (a), un interruptor abierto significa 0_L y un interruptor cerrado, 1_L, un LED apagado = 0_L y un LED encendido = 1_L.

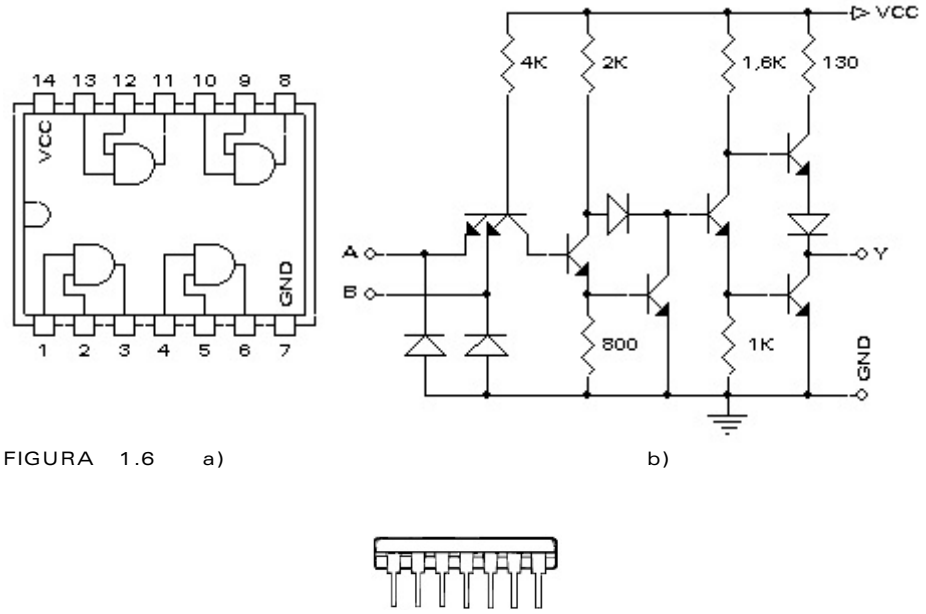


FIGURA 1.6 a) b)

La fig. 1.6 a) muestra la distribución de pines del CI-7408 que tiene 4 compuertas AND de 2-entradas.

La fig. 1.6 b) muestra la circuitería de una compuerta AND con tecnología TTL, con salida Totem-Pole. Los diodos de las entradas sirven para proteger a la compuerta de voltajes negativos y reciben el nombre inglés de diodos "clamp".

Conectivo OR [Disyunción "O"].- Sirve para separar dos o más proposiciones que pueden ser VERDADERAS o FALSAS. Sea la proposición compuesta:

Y = Jaime, sabe jugar ¿fútbol o básquet?

Para saber cuándo la proposición Y es VERDADERA o cuándo es FALSA, se la divide en dos subproposiciones: A y B.

A = Jaime sabe jugar fútbol
B = Jaime sabe jugar básquet

La siguiente tabla permite analizar en qué condiciones la proposición Y es verdadera o falsa.

B	A	Y
FALSO	FALSO	FALSO
FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	VERDADERO
VERDADERO	VERDADERO	VERDADERO

B	A	Y
F	F	F
F	V	V
V	F	V
V	V	V

TABLA DE VERDAD DEL CONECTIVO OR DE 2-ENTRADAS

En la tabla anterior, si Jaime no sabe jugar fútbol, entonces la proposición A es FALSA y si no sabe jugar básquet, la proposición B también es FALSA, por tanto Y es FALSA. Si Jaime sabe jugar fútbol, A

es VERDADERA, pero no sabe jugar básquet, B es FALSA, entonces Y es VERDADERA. Si Jaime no sabe jugar fútbol, A es FALSA, pero si sabe jugar básquet, B en VERDADERA, entonces Y es VERDADERA. Finalmente si Jaime sabe jugar fútbol, A es VERDADERA, y sabe jugar básquet, B es VERDADERA, por tanto Y es VERDADERA. Si a una respuesta FALSA se le asigna el valor lógico 0 [0_L] y a una respuesta VERDADERA se le asigna el valor lógico 1 [1_L], la tabla anterior puede escribirse como se muestra en la siguiente tabla. Cuando se hace así, esta tabla puede relacionarse con una suma lógica [no suma aritmética] y la proposición Y puede expresarse así

B	A	Y
0_L	0_L	0_L
0_L	1_L	1_L
1_L	0_L	1_L
1_L	1_L	1_L

TABLA DE VERDAD DEL CONECTIVO OR PARA 2-ENTRADAS.

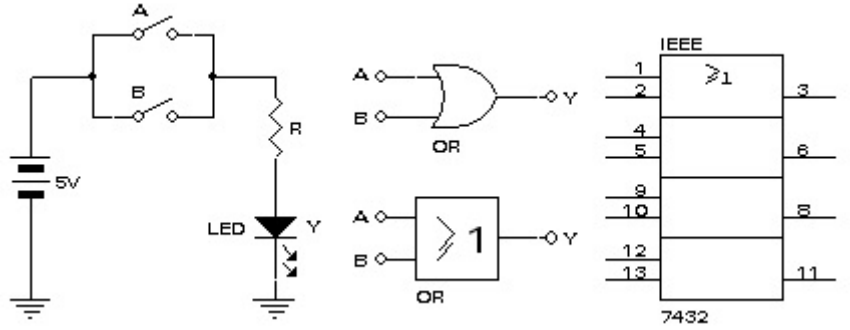


FIGURA 1.7 a) b) c)

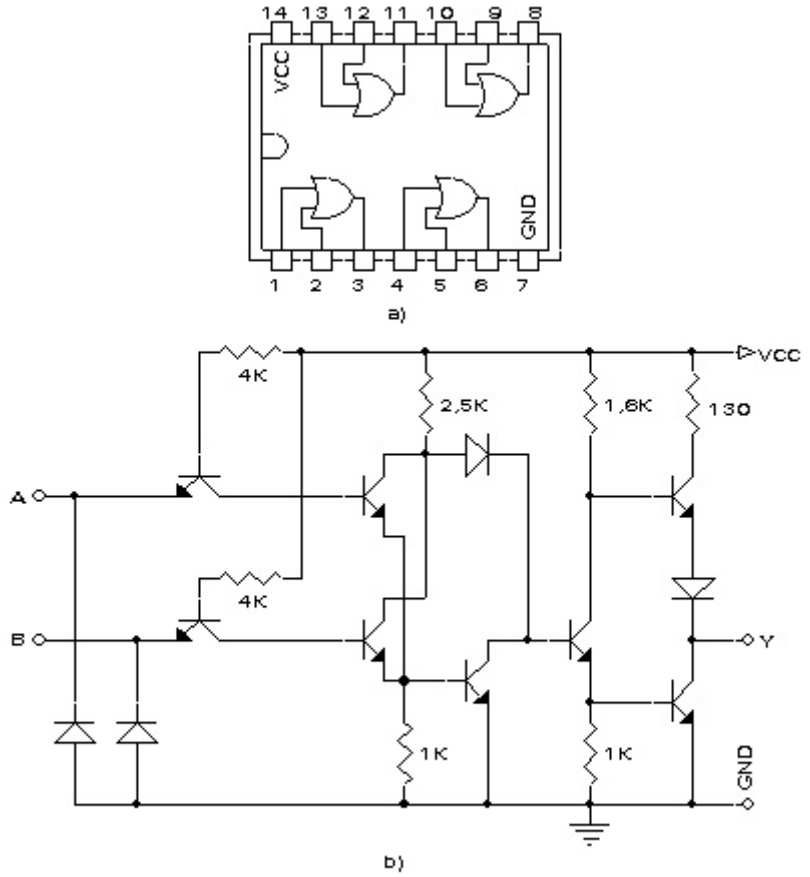


FIGURA 1.8 COMPUERTA OR TTL DE 2-ENTRADAS SALIDA TOTEM-POLE

La fig. 1.8 a) muestra la distribución de pines del CI-7432 que tiene 4 compuertas OR de 2-entradas. La fig. 1.8 b) muestra la circuitería de una compuerta OR con tecnología TTL con salida Totem-Pole.

Operador NOT [Inverter o Inversor].- Se lo define para un solo argumento; el operador NOT invierte el valor

lógico del argumento de entrada; también se lo conoce como **Inversor** o **Complemento.**

A	Y
F	V
V	F

A	Y
O_L	1_L
1_L	O_L

TABLA DE VERDAD DEL OPERADOR NOT

La función lógica del inversor se la representa mediante la siguiente ecuación booleana.

$$Y = \overline{A} = A' = A^*$$

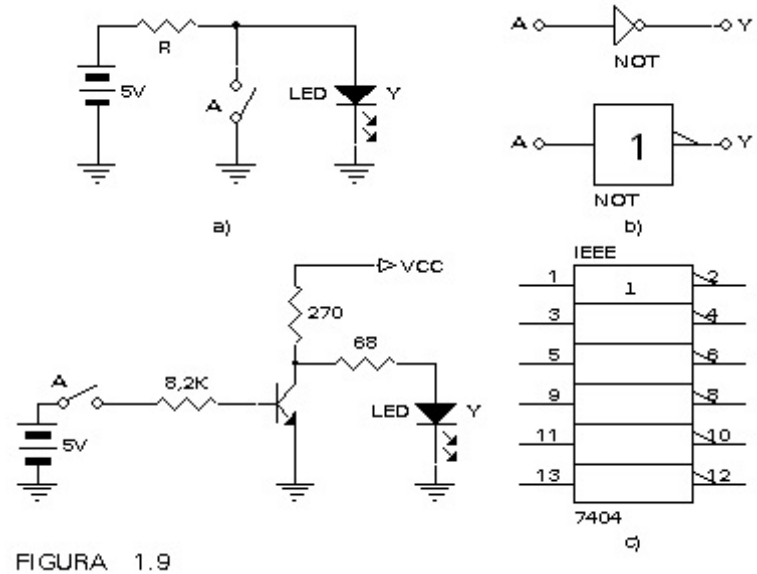
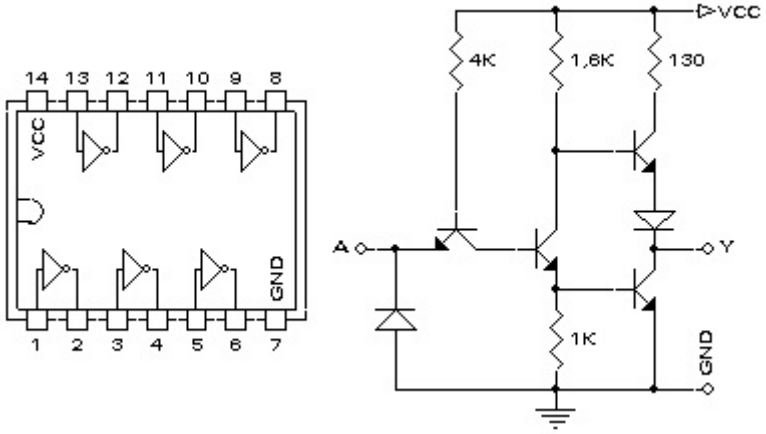


FIGURA 1.9

El circuito del inversor con interruptor y con transistor se muestra en la fig. 1.9 a); los símbolos

lógicos en la fig. 1.9 b) y el símbolo IEEE en la fig. 1.9 c). La fig. 1.10 a) muestra la distribución de pines del CI-7404 que tiene 6 compuertas NOT. La fig. 1.10 b) muestra la circuitería de una compuerta NOT con tecnología TTL con salida Totem-Pole.



a) b)

FIGURA 1.10 COMPUERTA NOT TTL [TOTEM-POLE]

Compuerta NAND [Conectivo NAND].- Es un dispositivo compuesto por un conectivo NOT conectado a la salida de un compuerta AND, como se muestra en la fig. 1.11 a); las figs. 1.11 b) y c) corresponden a los símbolos lógicos.

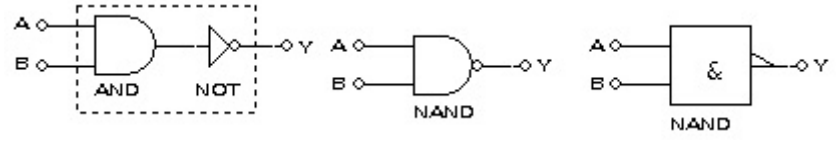


FIGURA 1.11 a) b) c)

$$Y = \overline{AB} = \overline{A \times B} = \overline{A \wedge B} = \overline{A \cap B} = A \uparrow B$$

B	A	Y
0_L	0_L	1_L
0_L	1_L	1_L
1_L	0_L	1_L
1_L	1_L	0_L

TABLA DE VERDAD DEL CONECTIVO NAND PARA 2-ENTRADAS

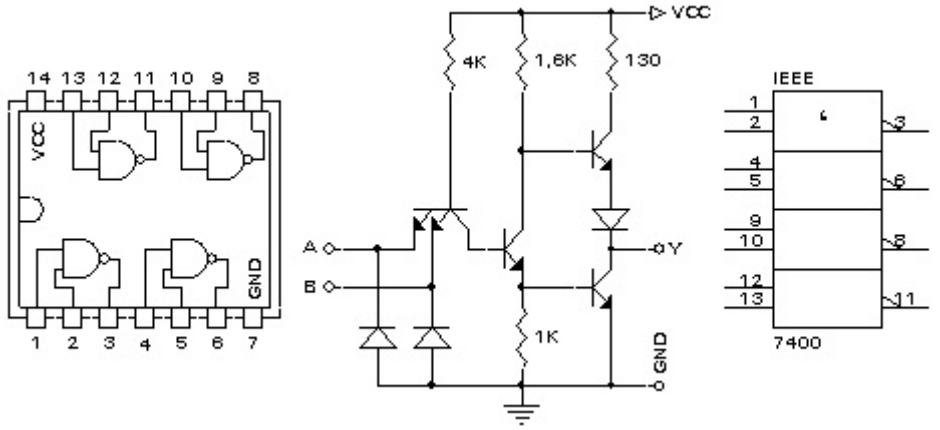


FIGURA A.12 a) b) c)

La fig. 1.12 a) muestra la distribución de pines del CI-7400 que tiene 4 compuertas NAND de 2-entradas. La fig. 1.12 b) muestra la circuitería de una compuerta NAND con tecnología TTL con salida Totem-Pole. Se observa que la estructura circuital es idéntica al de la compuerta NOT, la única diferencia es que el transistor de entrada tiene un solo emisor en la compuerta NOT y varios emisores en las compuertas NAND [en este caso dos].

La fig. 1.13 a) muestra la distribución de pines del CI-7401 que tiene 4 compuertas NAND de 2-entradas.

La fig. 1.13 b) muestra la circuitería de una compuerta NAND de tecnología TTL con salida en Colector Abierto [O. C. = Open Collector].

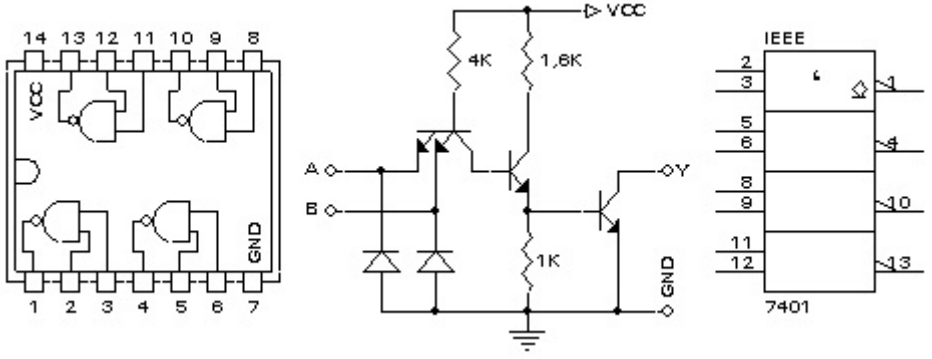


FIGURA 1.13 4-COMPUERTA NAND DE 2-ENTRADAS SALIDA COLECTOR ABIERTO

La fig.1.13 c) muestra el símbolo lógico IEEE del CI-7401, observe el rombo subrayado a la salida de la compuerta, que indica que se trata de salidas en colector abierto.

Compuerta NOR [Conectivo NOR].- Se obtiene conectando una compuerta NOT a la salida de una compuerta OR, como se indica en la fig. 1.14 a); las figs. 1.14 b) y c) muestran los símbolos lógicos de la compuerta NOR, la fig. 1.14 d) corresponde al símbolo IEEE.

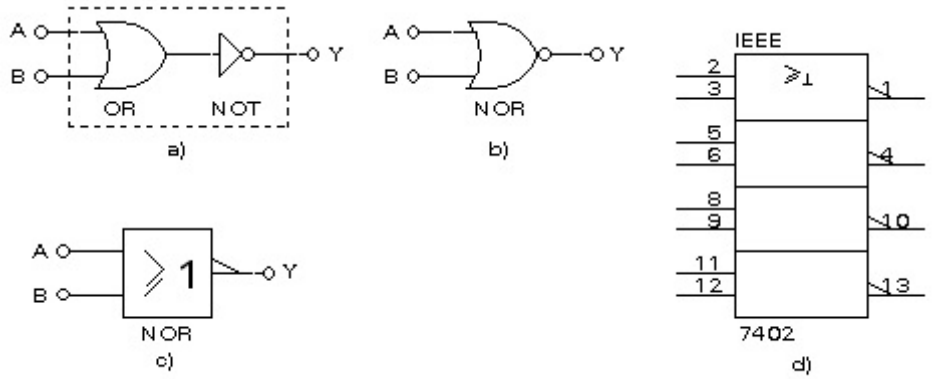


FIGURA 1.14

$$Y = \overline{A + B} = \overline{A \vee B} = \overline{A \cup B} = A \downarrow B$$

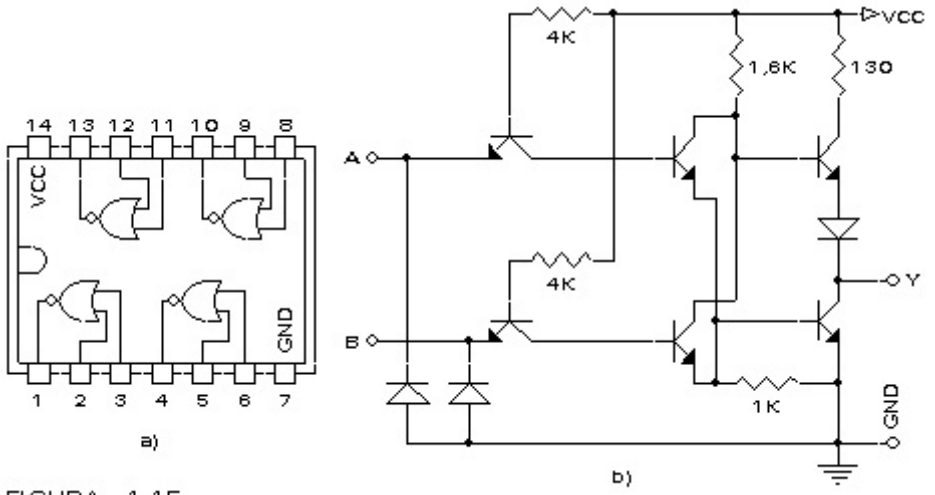


FIGURA 1.15

La fig. 1.15 a) muestra la distribución de pines del CI-7402 que tiene 4 compuertas NOR de 2-entradas. La fig. 1.15 b) muestra la circuitería de una compuerta NOR con tecnología TTL con salida Totem-

Pole.

Conjuntos Universales o Completos.- El conjunto de compuertas AND-OR-NOT [A-O-N] constituye un conjunto universal o funcionalmente completo, porque usando exclusivamente estas 3-compuertas se puede implementar cualquier circuito lógico, desde el más simple hasta el más complejo. Por ejemplo, el computador digital más grande está constituido por millones de compuertas A-O-N combinadas de alguna manera.

Como un ejemplo de ello se va a implementar la función **OR-Exclusiva** [XOR] usando compuertas A-O-N. Un ejemplo de proposición XOR sería: "En este momento, Jaime se encuentra jugando fútbol o está esquiando", Es obvio que Jaime no puede realizar los dos deportes al mismo tiempo. La siguiente tabla de verdad muestra la definición de la función XOR.

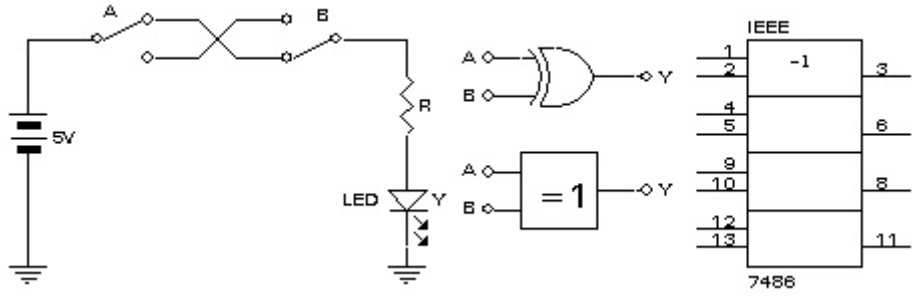


FIGURA 1.16 a) b) c)

El circuito de la compuerta XOR requiere interruptores de doble posición, y se muestra en la

fig. 1.16 a). La fig. 1.16 b) corresponde a los símbolos de la compuerta XOR y la fig. 1.16 c) corresponde al símbolo IEEE.

B	A	Y
0_L	0_L	0_L
0_L	1_L	1_L
1_L	0_L	1_L
1_L	1_L	0_L

$$Y = A\bar{B} + \bar{A}B = A \oplus B$$

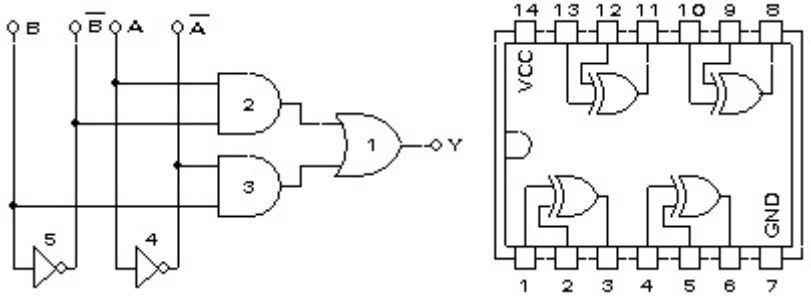
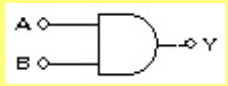
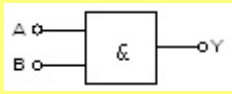
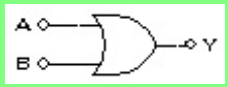
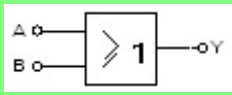
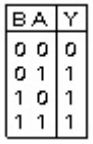
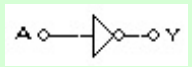
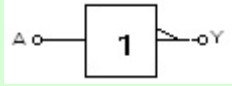
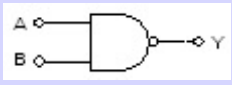
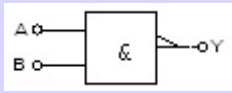
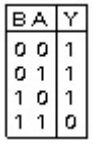
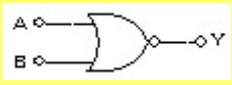
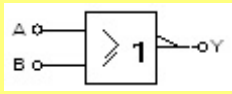
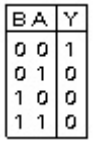
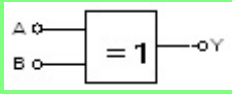
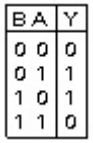


FIGURA 1.17 a) b)

La fig. 1.17 a) muestra la implementación de la compuerta XOR utilizando el conjunto de compuertas A-O-N, mientras que la fig. 1.17 b) muestra la distribución de pines del CI-7486/386 que corresponde a 4 compuertas XOR; los inversores sirven para generar $\bar{A}$ y $\bar{B}$; la compuerta 2 genera el término $A\bar{B}$; la compuerta 3 genera el término $\bar{A}B$, finalmente la compuerta 1 genera la función $A\bar{B} + \bar{A}B = A \oplus B$, que es la función XOR.

Resumen de Compuertas Lógicas Básicas

CI Y FUNCIÓN	SÍMBOLO-1 [TRADICIONAL]	SÍMBOLO-2 [IEEE - ANSI]	TABLA DE VERDAD
AND 7408 $Y = A \times B$			
OR 7432 $Y = A + B$			
NOT 7404 $Y = \overline{A}$			
NAND 7400 $Y = \overline{A \times B}$			
NOR 7402 $Y = \overline{A + B}$			
XOR 7486 $Y = A \oplus B$			

Postulados y Teoremas del Álgebra de Boole.- En el álgebra de Boole existen varios ***postulados, identidades y teoremas*** básicos.

Postulado.- Principio cuya admisión es necesaria para establecer una demostración. ***Verdad evidente que no necesita demostrarse.***

Identidad.- Igualdad cuyos dos miembros son idénticos.

Teorema.- Enunciado de una proposición o de una propiedad que se demuestra por un razonamiento lógico a partir de hechos dados o de hipótesis, incluídos en este enunciado. ***Proposición científica que se puede demostrar.***

Postulados [de Huntington]

0	x	0	=	0
0	x	1	=	0
1	x	0	=	0
1	x	1	=	1
		$\bar{1}$	=	0

PRODUCTO LÓGICO

1	+	0	=	1
1	+	0	=	1
0	+	1	=	1
0	+	1	=	0
		$\bar{0}$	=	1

SUMA LÓGICA

← Complemento

Principio de Dualidad.- Si se observa los postulados y las relaciones algebraicas anteriores, se ve que hay dos formas para cada uno de ellos. Esto parece implicar que debería comprobarse ambas relaciones.

Sin embargo, el **principio de dualidad**[2] simplifica el esfuerzo. Este principio establece que cada teorema tiene un dual que se puede obtener:

a) INTERCAMBIANDO LOS OPERADORES AND Y OR DE LAS EXPRESIONES.
b) INTERCAMBIANDO LOS ELEMENTOS 0 Y 1 DE LAS EXPRESIONES.
c) LA FORMA DE LAS VARIABLES [SI LAS HUBIERA] NO CAMBIA.

$$0 \;.\; 1 = 0 \qquad\qquad a \;.\; 1 = a$$
$$\updownarrow \;\updownarrow\; \updownarrow \quad \updownarrow \qquad\qquad \updownarrow \;\updownarrow\; \updownarrow \quad \updownarrow$$
$$1 \;+\; 0 = 1 \qquad\qquad a \;+\; 0 = a$$

En el caso de que existan variables, estas permanecen sin cambios.

ADVERTENCIA.- Si $G<X_1, \ldots, X_n>$ es el dual de la función $F<X_1, \ldots, X_n>$ esto no implica que las dos expresiones sean iguales. La verdad de esta advertencia se verifica fácilmente examinando las funciones en los ejemplos dados arriba.

Este principio permite demostrar dos teoremas con el esfuerzo de una sola prueba. Si se puede probar, con una serie de pasos lógicos, que un teorema dado es verdadero, entonces, inmediatamente se sabe que el dual del teorema original también es verdadero, puesto que el dual de los pasos lógicos que prueban el teorema original, prueban el teorema dual.

2.- Taylor L. Booth.- Digital Network and Computer Systems.- Wiley International Edition.- 1978.

Proposiciones Elementales.- Las proposiciones básicas o elementales del álgebra de Boole se establecen a partir de las tablas de verdad de los conectivos AND y OR, como se indica en la siguiente tabla.

$a \cdot a = a$	$a + a = a$	← Idempotencia [Tautología]
$a \cdot \bar{a} = 0$	$a + \bar{a} = 1$	← Complementos
$a \cdot 1 = a$	$a + 0 = a$	← Identidades
$a \cdot 0 = 0$	$a + 1 = 1$	← Elementos nulos
$\bar{\bar{a}} = a$		← Involución

Leyes Fundamentales

Ley ***CONMUTATIVA***

$$X \cdot Y = Y \cdot X \qquad\qquad X + Y = Y + X$$

Ley ***ASOCIATIVA***

$$(X \cdot Y)Z = X(Y \cdot Z) \qquad (X + Y) + Z = X + (Y + Z)$$

Ley ***DISTRIBUTIVA***

$$X \cdot (Y + Z) = X \cdot Y + X \cdot Z \qquad X + Y.Z = (X + Y)(X + Z)$$

Teoremas

Teorema de ***ABSORCIÓN (COBERTURA)***

$$X + X \cdot Y = X \qquad X(X + Y) = X$$

Teorema de ***REDUNDANCIA***

$$X + \bar{X} \cdot Y = X + Y \qquad X(\bar{X} + Y) = X \cdot Y$$

Demostración Tabular

Y	X	$\bar{X}$	$\bar{X}Y$	$X + \bar{X}Y$	$X + Y$
0	0	1	0	0	0
0	1	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

La tabla anterior es una forma válida de realizar la demostración de una igualdad [identidad] booleana. Recibe el nombre de **"demostración por inducción completa"**, porque se analizan todas las posibles combinaciones de las variables de entrada. En este caso se observa que las dos columnas de la derecha son iguales, lo que implica que los dos lados de la identidad booleana son iguales.

Teorema de ***CONSENSO***

$$X \cdot Y + Y \cdot Z + \bar{X} \cdot Z = X \cdot Y + \bar{X} \cdot Z$$

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

Teorema de ***COMBINACIÓN***

$$X \cdot Y + X \cdot \bar{Y} = X \qquad (X + Y)(X + \bar{Y}) = X$$

Teorema de ***DeMORGAN***

$$\overline{X \cdot Y} = \bar{X} + \bar{Y} \qquad \overline{X + Y} = \bar{X} \cdot \bar{Y}$$

Teorema de ***Expansión de SHANNON***

$$F<X_1, \ldots, X_n> = X_1F<1, \ldots, X_n> + \bar{X}_1F<0, \ldots, X_n>$$

$$F<X_1, \ldots, X_n> = [X_1 + F<0, \ldots, X_n>][\bar{X}_1 + F<1, \ldots, X_n>]$$

Ejemplo.- Aplicación del teorema de expansión de Shannon. Expandir la función simplificada: F<A,B,C> = $AB + \bar{A}C$. En primera instancia se expandirá la variable B que falta en el segundo término y después se completará la variable C que falta en el primer término.

$$AB + \bar{A}C = B(A.1 + \bar{A}C) + \bar{B}(A.0 + \bar{A}C)$$

$$= AB + \bar{A}BC + \bar{A}\bar{B}C$$
$$= C(AB + \bar{A}B.1 + \bar{A}\bar{B}.1) + \bar{C}(AB + +\bar{A}B.0 + \bar{A}\bar{B}.0)$$
$$= ABC + \bar{A}BC + \bar{A}\bar{B}C + AB\bar{C}$$

Simplificación de Funciones Booleanas Utilizando los Teoremas del Álgebra de Boole.- La ecuación booleana de una función lógica se la puede obtener de su tabla de verdad; en general será posible simplificar esa ecuación para obtener la función más simple posible, la función booleana simplificada es la que se implementará con las compuertas lógicas. La importancia de la simplificación se debe a que al reducir el número de compuertas se disminuye el número de conexiones, el tamaño físico del circuito, la potencia disipada por el mismo, el costo total e, inclusive, el número de errores que pueden introducirse cuando se implementa el circuito. **El circuito que se implementará es el que tenga el menor número de compuertas y el menor número de conexiones.** Una forma de simplificar una ecuación booleana es mediante el uso de los postulados y teoremas del álgebra de Boole que se acaba de estudiar. Esto se ilustra con los siguientes ejemplos.

Ejemplo 1.- Utilizando compuertas A-O-N, implementar la siguiente función booleana. Después simplificar la función, implementarla con compuertas A-O-N. Comparar los dos circuitos.

$$F = A + \bar{A} \cdot B$$

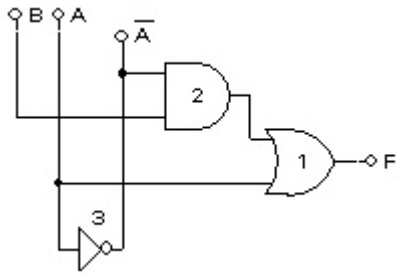


FIGURA 1.18 CIRCUITO NO SIMPLIFICADO

FIGURA 1.19 CIRCUITO SIMPLIFICADO

$F = A \cdot 1 + \bar{A}B$ IDENTIDAD

$F = A(B + \bar{B}) + \bar{A}B$ COMPLEMENTOS

$F = AB + A\bar{B} + \bar{A}B$ DISTRIBUTIVA

$F = AB + AB + A\bar{B} + \bar{A}B$ IDEMPOTENCIA

$F = A(B + \bar{B}) + B(A + \bar{A})$ DISTRIBUTIVA

$F = A \cdot 1 + B \cdot 1$ COMPLEMENTOS

$F = A + B$ IDENTIDAD

El circuito no simplificado, correspondiente a la ecuación original se muestra en la fig. 1.18 y la función simplificada se indica en el circuito de la fig. 1.19; se observa que este último es mucho más sencillo que el circuito sin simplificar. De modo que el circuito de la fig. 1.19 es el que debe utilizarse en la práctica.

Ejemplo2.- Utilizando compuertas A-O-N, implementar la siguiente función booleana. Después simplificar la función e implementarla con compuertas A-O-N. Comparar los dos circuitos.

$$F<X,Y,Z> = \bar{X}\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + X\bar{Y}\bar{Z} + X\bar{Y}Z + XY\bar{Z}$$

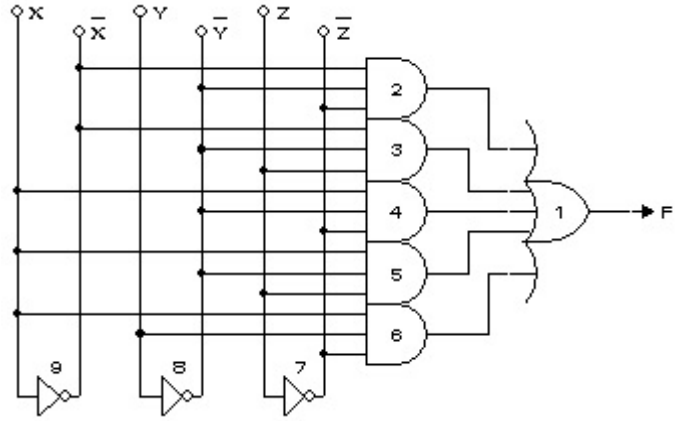


FIGURA 1.20 CIRCUITO NO SIMPLIFICADO

La función simplificada es $F = \bar{Y} + X\bar{Z}$.

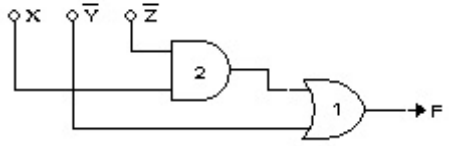


FIGURA 1.21 CIRCUITO SIMPLIFICADO

El circuito no simplificado, correspondiente a la ecuación original se muestra en la fig. 1.20 y la función simplificada se indica en la fig. 1.21. Se observa que el segundo circuito es mucho más sencillo que el circuito sin simplificar, ambos cumplen la misma función, sin embargo, el ingeniero necesariamente debe optar por el segundo [más simplificado].

Problemas.- Simplificar las siguientes funciones booleanas.

$$F<D,C,B,A> = \bar{D}\bar{C}BA + D\bar{B}A + DB\bar{A} + \bar{D}CB + D\bar{B}\bar{A} + \bar{D}\bar{C}B\bar{A}$$

$$G<C,B,A> = CBA + \bar{C}BA + C\bar{B}A + \bar{C}\bar{B}A$$

$$H<Z,Y,X> = ZX + Z\bar{Y} + \bar{Y}$$

$$F_1 = CBA + C\bar{B}A + \bar{C}BA$$

$$F_2 = CBA + CB\bar{A} + \bar{C}BA + \bar{C}B\bar{A}$$

Demostrar que

$$D \oplus C + C = \bar{D}\bar{C}$$

$$(b + a)(c + a)(d + a) = a + dcb$$

Ejemplo.- Determinar la ecuación booleana del circuito de la fig. 1.22.

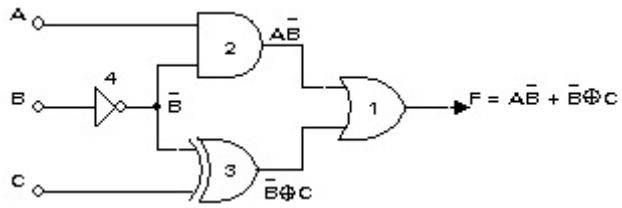


FIGURA 1.22

En el circuito de la fig 1.22, el inversor [compuerta 4] genera $\bar{B}$; la compuerta AND [2], genera el término $A\bar{B}$; la compuerta XOR [3], el término $\bar{B} \oplus C$; finalmente, la compuerta OR [1], genera la función: $\boxed{F = A\bar{B} + \bar{B} \oplus C}$, que es la respuesta.

Universalidad de las Compuertas NAND y NOR.- De la misma manera en que las compuertas A-O-N, constituyen un conjunto completo, la compuerta NAND, por si sola, constituye un conjunto completo, es decir utilizando exclusivamente compuertas NAND se puede implementar cualquier red lógica, por compleja que sea. Lo mismo podemos decir de la compuerta NOR.

Ejemplo.- Utilizando solamente compuertas NAND, implementar la compuerta XOR, cuya función está definida como

$$Y = A\bar{B} + \bar{A}B$$

$$Y = \overline{\overline{A\bar{B} + \bar{A}B}} \qquad \text{[Involución]}$$

$$Y = \overline{\overline{A\bar{B}}}\,.\,\overline{\overline{\bar{A}B}} \qquad \text{[DeMorgan]}$$

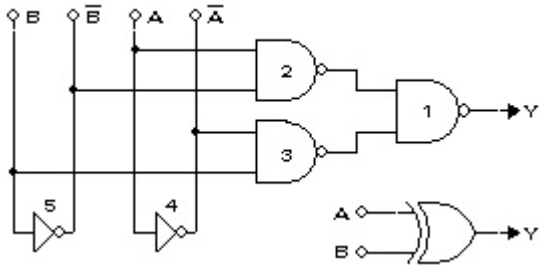


FIGURA 1.23

La salida del circuito de la fig. 1.23 es $\boxed{Y = A\oplus B}$. En la fig. 1.23, un circuito XOR con compuertas NAND, se ve que el número de conectivos que se ha requerido para implementar la compuerta XOR, utilizando compuertas NAND, es el mismo que el que se usó con compuertas A-O-N.

El circuito de la fig. 1.24, con solo 4-compuertas NAND de 2-entradas, también corresponde a una compuerta XOR, es decir, $\boxed{Y = A\oplus B}$.

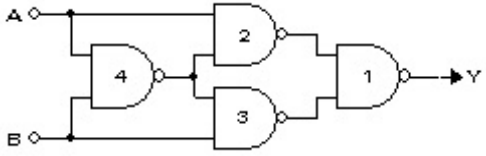


FIGURA 1.24

Ejemplo.- Utilizando solamente compuertas NOR, implementar la compuerta XOR, cuya función está definida como

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

$$F = \overline{\overline{(\overline{A} + \overline{B})}\ (A + B)} \qquad \text{[Involución]}$$

$$F = \overline{\overline{(\overline{A} + \overline{B})} + \overline{(A + B)}} \qquad \text{[DeMorgan]}$$

En la fig. 1.25 se ve que el número de compuertas NOR que se requieren para implementar la compuerta XOR, es el mismo que el que se usó con compuertas A-O-N o con compuertas NAND [fig. 1.23].

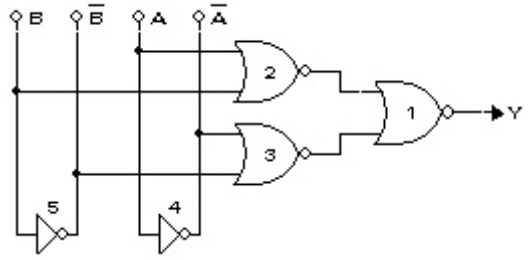


FIGURA 1.25

Ejemplo.- Simplificar la función que se indica a continuación, implementar la función simplificada con compuertas A-O-N y con compuertas NAND.

$$\underset{1}{ABCD} + \underset{2}{AB\overline{C}D} + \underset{3}{A\overline{B}CD} + \underset{4}{\overline{A}BCD} + \underset{5}{A\overline{B}\overline{C}D} + \underset{6}{A\overline{B}\overline{C}\overline{D}}$$

Si se agrupan los términos 1 y 3 se elimina la

variable B, y que daría el término ACD, al agrupar los términos 2 y 5, también se elimina la variable B, el término que queda es $A\overline{C}D$, al agrupar los términos 1 y 4, se elimina la variable A, el término que queda es BCD, así mismo, al agrupar los términos 5 y 6, se elimina la variable D y el término que queda es $A\overline{B}\overline{C}$. Por tanto la función en una primera simplificación quedaría como

$$F = ACD + A\overline{C}D + BCD + A\overline{B}\overline{C}$$

En la ecuación anterior pueden agruparse los término 1 y 2, porque solo cambia la variable C, que es la que se eliminará. Finamente la función simplificada quedaría como

$$F = AD + BCD + A\overline{B}\overline{C}$$

Que puede implementarse con compuertas A-O-N. Para hacerlo con compuertas NAND, puede utilizarse la misma metodología que se utilizó para implementar la función XOR con compuertas NAND y NOR, que fueron Involución y el teorema de DeMorgan. De modo que la función booleana para la implementación con compuertas NAND queda como

$$F = \overline{\overline{AD} \cdot \overline{BCD} \cdot \overline{A\overline{B}\overline{C}}}$$

La fig. 1.26 , muestra el circuito implementado

con A-O-N y con NAND. Las compuertas 5 y 6 [NOT], también pueden implementarse con NAND.

Ejemplo.- Simplificar la función que se indica a continuación, implementar la función simplificada con compuertas A-O-N y con compuertas NOR.

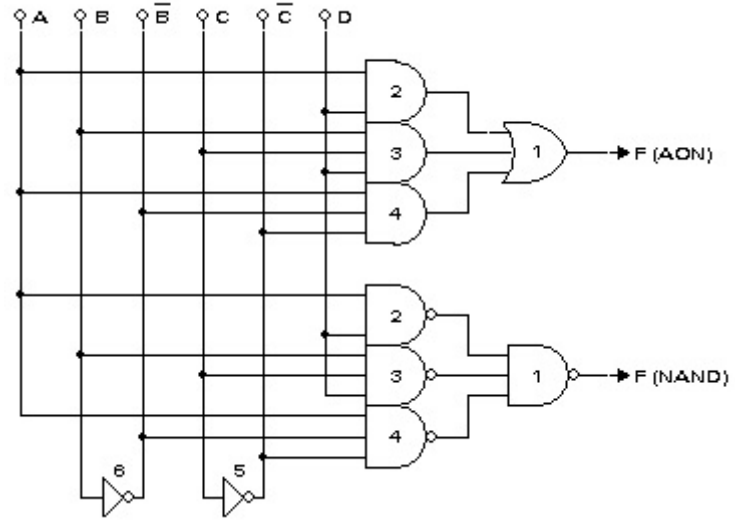


FIGURA 1.26

$$F = \underset{1}{(\overline{X} + \overline{Y} + Z)}\underset{2}{(\overline{X} + Y + \overline{Z})}\underset{3}{(X + \overline{Y} + Z)}\underset{4}{(X + Y + Z)}\underset{5}{(X + \overline{Y} + \overline{Z})}$$

Si se agrupan los términos 1 y 3 se elimina la variable X, lo que genera el término $(\overline{Y} + Z)$, al agrupar los términos 3 y 4, se elimina la variable y, el término que queda es [X + Z], y al agrupar los términos 3 y 5, se elimina la variable Z, el término

que queda es $[X + \overline{Y}]$, puede observarse que el término 2 no puede agruparse con ninguno y por tanto no se puede simplificar. La función simplificada quedaría como

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

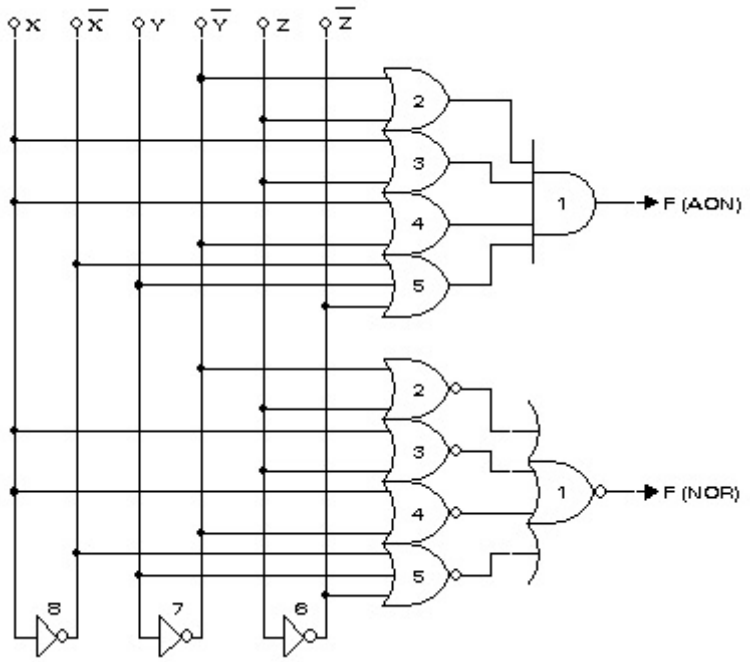


FIGURA 1.27

Que puede implementarse con compuertas A-O-N. Para hacerlo con compuertas NOR, puede utilizarse la misma metodología que se utilizó para implementar la función XOR con compuertas NAND y NOR, que fueron Involución y el teorema de DeMorgan. De modo que la función

booleana para la implementación con compuertas NOR quedaría como

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

La fig. 1.27 muestra el circuito implementado con A-O-N y con NOR. Las compuertas 6, 7 y 8 [NOT], también pueden implementarse con NOR.

Representación de las variables booleanas . – Para representar una variable booleana [en el Laboratorio], por ejemplo la variable A, se puede utilizar un interruptor y una resistencia y un voltaje de $5V_{DC}$.

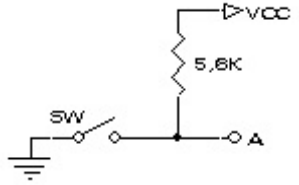


FIGURA 1.28

La fig. 1.28, muestra el circuito, de manera que cuando el interruptor esté abierto la variable A toma el valor 1_L y cuando esté cerrado la variable A toma el valor 0_L.

Cuando se tiene un grupo de variables booleanas, se puede usar el circuito que se muestra en la fig. 1.29 En este caso se utiliza un DIP-Switch de 8 interruptores, con lo que pueden tener hasta 8 posibles variables [A, B, C, D, E, F, G y H].

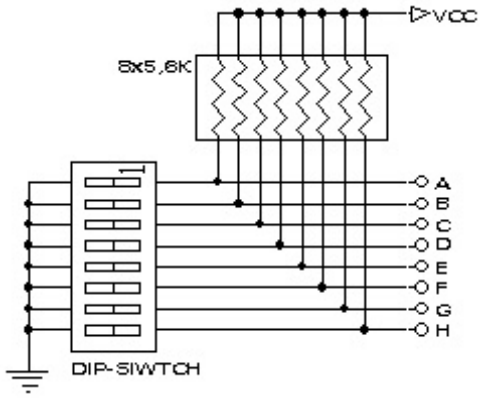


FIGURA 1.29

Para poder observar el valor que toma una variable de salida, por ejemplo la variable Y, se puede utilizar el circuito de la fig. 1.30, que usa un LED y un transistor NPN, que funciona como amplificador Emisor-Común que trabaja en corte y saturación.

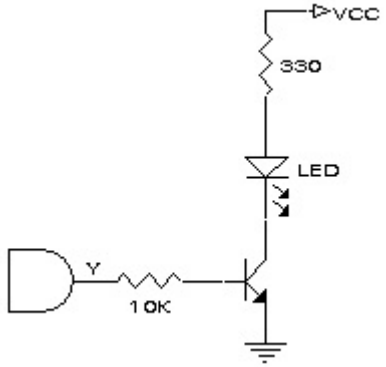


FIGURA 1.30

Cuando la señal Y [salida de una compuerta AND, por ejemplo] toma el valor 0_L el transistor está en

corte y el LED no se enciende, cuando la señal Y = 1_L, el transistor se satura aproximadamente a 10mA [V_{CC} = 5V] y el LED se enciende.

Formas Estándar de las Funciones Booleanas.- Se ha visto que es posible describir una función booleana mediante una tabla de verdad que muestra los valores de la función para todas las posibles combinaciones de 0s y 1s de sus argumentos o variables de entrada. De la misma manera, se ha visto que otra forma de presentar el comportamiento de una función es mediante una ecuación booleana. En esta sección se estudiará cómo obtener una ecuación booleana que está descrita por una tabla de verdad.

Representación de una Función Booleana Utilizando los 1s de la Tabla de Verdad [Minterms].- Para esto se utilizará el siguiente ejemplo: Diseñar un circuito lógico que tiene de 3-variables de entrada [C, B y A] y una variable de salida [Y], de tal manera que cuando en las entradas haya un número impar de 1s, la salida [Y] tome el valor 1_L, en cualquier otro caso la salida debe ser 0_L. Este circuito recibe el nombre de **detector/generador de paridad.**

Solución.- La siguiente tabla de verdad muestra el comportamiento del circuito lógico pedido. Para resolver este problema se han utilizado 4-variables auxiliares: Y_1, Y_2, Y_3 y Y_4, una por cada 1_L que tiene

la variable de salida Y. Cada variable auxiliar genera un producto lógico de las variables de entrada [por ejemplo, Y_4 = CBA], además tiene un mínimo de 1s y un máximo de 0s. Por esta razón, a los términos generados por cada una de las variables auxiliares [1s, en la tabla de verdad], se lo denomina ***término mínimo*** (***minterm***).

C	B	A	Y	Y_4	Y_3	Y_2	Y_1	minterms
0	0	0	0	0	0	0	0	
0	0	1	1	0	0	0	1	$\bar{C}\bar{B}A = m_1$
0	1	0	1	0	0	1	0	$\bar{C}B\bar{A} = m_2$
0	1	1	0	0	0	0	0	
1	0	0	1	0	1	0	0	$C\bar{B}\bar{A} = m_4$
1	0	1	0	0	0	0	0	
1	1	0	0	0	0	0	0	
1	1	1	1	1	0	0	0	$CBA = m_7$

$$Y_1 = \bar{C}\bar{B}A = m_1 \qquad Y_2 = \bar{C}B\bar{A} = m_2$$
$$Y_3 = C\bar{B}\bar{A} = m_4 \qquad Y_4 = CBA = m_7$$

Puede observarse que en cada uno de los términos generados, están presentes las 3-variables de entrada, en su forma normal o en su forma complementada. Ahora bien, la variable de salida Y, corresponde a la suma lógica de las 4-variables auxiliares, es decir

$$Y<C,B,A> = Y_1 + Y_2 + Y_3 + Y_4$$
$$Y<C,B,A> = \bar{C}\bar{B}A + \bar{C}B\bar{A} + C\bar{B}\bar{A} + CBA$$

A este tipo de ecuación booleana, en la que en cada término están presentes todas las variables de entrada, en su forma normal o en su forma complementada, se la denomina ***forma estándar*** o ***forma canónica***. En este caso

FORMA CANÓNICA DISYUNTIVA
SUMA DE TÉRMINOS MÍNIMOS [MINTERMS]
SUMA EXPANDIDA DE PRODUCTOS
DESCOMPOSICIÓN EN MINTERMS

A los minterms, se los representa con una m [minúscula] y un subíndice que corresponde al equivalente decimal del número binario del que proviene; por ejemplo, $m111_2 \equiv m7_{10}$. De modo que, en el ejemplo anterior, la correspondiente ecuación también se expresa de las siguientes maneras

$$Y<C, B, A> = m_1 + m_2 + m_4 + m_7$$

$$Y<C, B, A> = \sum m(1,2,4,7)$$

En general, una función de N-variables de entrada puede tener hasta 2^N minterms. Para el caso de 3-variables de entrada, los correspondientes minterms serían: m_0, m_1, m_2, m_3, m_4, m_5, m_6 y m_7. Cada minterm se genera de la siguiente manera: si la variable de entrada tiene el valor 0_L, la variable aparece complementada; si la variable de entrada tiene el valor 1_L la variable aparece en su forma normal [sin

complemento].

En la mayoría de ocasiones se puede simplificar una función canónica booleana. En el ejemplo propuesto, es posible hacer esto, en cuyo caso la ecuación simplificada es la que se indica a continuación.

$$Y = A \oplus B \oplus C$$

El circuito lógico se indica en la fig. 1.31.

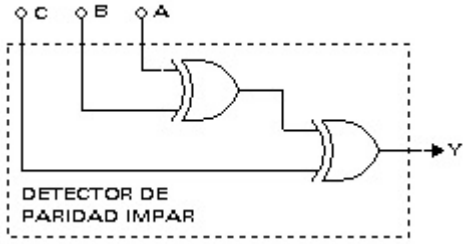


FIGURA 1.31

En algunos casos es posible generalizar el diseño de un circuito lógico. De la ecuación anterior se ve que para implementar un detector/generador de paridad impar de mayor número de variables de entrada puede generalizarse. Por ejemplo para 4-variables de entrada [D, C, B, A], la función de salida será

$$Y = A \oplus B \oplus C \oplus D$$

que requiere 3 compuertas XOR como se muestra en la

fig. 1.32.

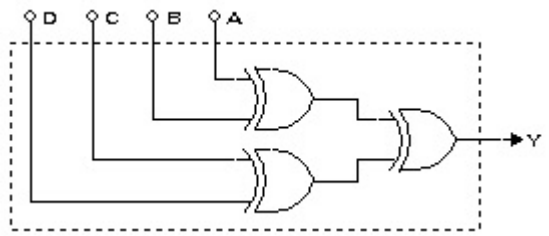


FIGURA 1.32

Representación de una Función Booleana Utilizando los 0s de la Tabla de Verdad [Maxterms].- La función booleana de un circuito lógico puede escribirse utilizando los 0s de la tabla, en vez de los 1s como se hizo anteriormente. En este caso, en lugar de tener sumas de productos se tienen productos de sumas y cada 0 genera un factor en la ecuación correspondiente.

Ejemplo.- Diseñar un circuito digital que dispone de 3-entradas [C, B y A] y una salida [Y]. La salida debe ser 1 cuando en las entradas haya un número impar de 1s [**detector/chequeador de paridad**].

C	B	A	Y	Maxterms
0	0	0	0	$M_0 = C + B + A$
0	0	1	1	
0	1	0	1	
0	1	1	0	$M_3 = C + \overline{B} + \overline{A}$
1	0	0	1	

1	0	1	0	$M_5 = \overline{C} + B + \overline{A}$
1	1	0	0	$M_6 = \overline{C} + \overline{B} + A$
1	1	1	1	

A los términos generados por cada uno de los 0s de la tabla de verdad, se los denomina ***término máximo*** (***maxterm***). Puede observarse que en cada uno de los términos generados, están presentes las 3-variables de entrada, en su forma normal [cuando la variable correspondiente vale 0_L] o en su forma complementada [cuando la variable correspondiente vale 1_L]. De manera que la ecuación completa utilizando los 0s de la tabla de verdad quedaría como se muestra en la siguiente ecuación.

$$F\langle C, B, A\rangle = (C + B + A)(C + \overline{B} + \overline{A})(\overline{C} + B + \overline{A})(\overline{C} + \overline{B} + A)$$

Esta ecuación booleana, también es una ***forma estándar*** o ***forma canónica***. En este caso

FORMA CANÓNICA CONJUNTIVA
PRODUCTO DE TÉRMINOS MÁXIMOS [MAXTERMS]
PRODUCTO EXPANDIDO DE SUMAS
DESCOMPOSICIÓN EN MAXTERMS

En general, una función de N-variables de entrada puede tener hasta 2^N maxterms. Para el caso de 3-variables de entrada, los correspondientes maxterms serían: M_0, M_1, M_2, M_3, M_4, M_5, M_6 Y M_7. Cada maxterm

se genera de la siguiente manera: si la variable de entrada tiene el valor 0_L, la variable aparece en su forma normal [sin complemento]; si la variable de entrada tiene el valor 1_L la variable aparece complementada. A los maxterms, se los representa con una M [mayúscula] y un subíndice que corresponde al equivalente decimal del número binario del que proviene. También puede expresarse de las siguientes maneras.

$$Y<C,B,A> = M_0.M_3.M_5.M_6$$
$$Y<C,B,A> = \prod M(0,3,5,6)$$

Ejemplo.- Diseñar un circuito lógico que convierta un número expresado en código binario natural de 3-bits $[B_2B_1B_0]$ en el código de Gray correspondiente $[G_2G_1G_0]$. Obtener las ecuaciones de las variables de salida en las formas canónicas conjuntiva y disyuntiva.

BIN			GRAY		
B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

La tabla de función para este convertidor de código se muestra a continuación. La forma canónica disyuntiva de las ecuaciones de las variables de salida se muestran a continuación.

$$\begin{cases} G_2<B_2,B_1,B_0> = B_2\bar{B}_1\bar{B}_0 + B_2\bar{B}_1B_0 + B_2B_1\bar{B}_0 + B_2B_1B_0 \\ G_2<B_2,B_1,B_0> = \bar{B}_2B_1\bar{B}_0 + \bar{B}_2B_1B_0 + B_2\bar{B}_1\bar{B}_0 + B_2\bar{B}_1B_0 \\ G_2<B_2,B_1,B_0> = \bar{B}_2\bar{B}_1B_0 + \bar{B}_2B_1\bar{B}_0 + B_2\bar{B}_1B_0 + B_2B_1\bar{B}_0 \end{cases}$$

que puede escribirse como

$$\begin{cases} G_2<B_2,B_1,B_0> = m_4 + m_5 + m_6 + m_7 \\ G_1<B_2,B_1,B_0> = m_2 + m_3 + m_4 + m_5 \\ G_0<B_2,B_1,B_0> = m_1 + m_2 + m_5 + m_6 \end{cases}$$

o también

$$\begin{cases} G_2<B_2,B_1,B_0> = \sum (4,5,6,7) \\ G_1<B_2,B_1,B_0> = \sum (2,3,4,5) \\ G_0<B_2,B_1,B_0> = \sum (1,2,5,6) \end{cases}$$

cualquiera de estas opciones, representa una forma canónica disyuntiva.

La forma canónica conjuntiva de las ecuaciones de las variables de salida son

$$\begin{cases} G_2<B_2,B_1,B_0> = (B_2 + B_1 + B_0)(B_2 + B_1 + \bar{B}_0)(B_2 + \bar{B}_1 + B_0)(B_2 + \bar{B}_1 + \bar{B}_0) \\ G_1<B_2,B_1,B_0> = (B_2 + B_1 + B_0)(B_2 + B_1 + \bar{B}_0)(\bar{B}_2 + \bar{B}_1 + B_0)(\bar{B}_2 + \bar{B}_1 + \bar{B}_0) \\ G_0<B_2,B_1,B_0> = (B_2 + B_1 + B_0)(B_2 + \bar{B}_1 + \bar{B}_0)(\bar{B}_2 + B_1 + B_0)(\bar{B}_2 + \bar{B}_1 + \bar{B}_0) \end{cases}$$

o también

$$\begin{cases} G_2<B_2,B_1,B_0> = M_0 M_1 M_2 M_3 \\ G_1<B_2,B_1,B_0> = M_0 M_1 M_6 M_7 \\ G_0<B_2,B_1,B_0> = M_0 M_3 M_4 M_7 \end{cases}$$

o, lo que es lo mismo

$$\begin{cases} G_2<B_2, B_1, B_0> = \prod M(0, 1, 2, 3) \\ G_1<B_2, B_1, B_0> = \prod M(0, 1, 6, 7) \\ G_0<B_2, B_1, B_0> = \prod M(0, 3, 4, 7) \end{cases}$$

todas de estas opciones, representan una forma canónica conjuntiva.

$$\begin{cases} G_2 = B_2 \\ G_1 = B_2 \oplus B_1 \\ G_0 = B_1 \oplus B_0 \end{cases}$$

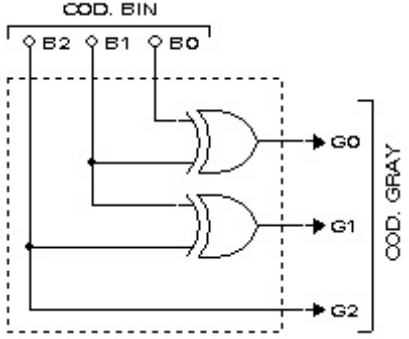


FIGURA 1.33 CODIFICADOR DE BINARIO NATURAL A CÓDIGO DE GRAY DE 3-BITS

Para simplificar estas funciones pueden utilizarse las leyes y teoremas del álgebra de Boole. En cuyo caso las funciones simplificadas, serían El circuito con compuertas XOR, se muestra en la fig. 1.33.

En es caso, también es posible la generalización del diseño. Por ejemplo para implementar un circuito lógico que convierta un número en código binario natural de 4-bits $[B_{3B}B_2B_1B_0]$ en el código de Gray correspondiente $[G_3G_2G_1G_0]$. Observando el grupo de ecuaciones que se obtuvieron antes, se deduce que

$$\begin{cases} G_3 = B_3 \\ G_2 = B_3 \oplus B_2 \\ G_1 = B_2 \oplus B_1 \\ G_0 = B_1 \oplus B_0 \end{cases}$$

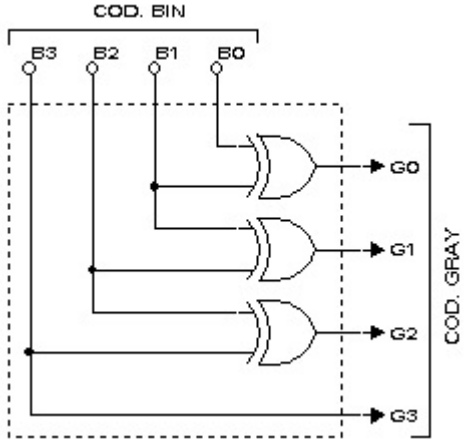


FIGURA 1.34 CODIFICADOR DE BIN - A -GRAY DE 4-BITS

el circuito se muestra en la fig. 1.34.

Sugerencia.- Diseñar una red lógica combinacional que convierta un número en código de Gray de 3-bits $[G_2G_1G_0]$ en el correspondiente código binario natural

[$B_2B_1B_0$]. Obtener las ecuaciones de las variables de salida en las formas canónicas conjuntiva y disyuntiva. Implementar el circuito el menor número de compuertas XOR. Ver si es factible generalizar para mayor número de entradas e implementarlas con compuertas XOR.

Representación y Simplificación de Funciones Booleanas Usando el Mapa de Karnaugh o de Veitch .- El mapa-K es un método gráfico para visualizar de mejor manera la tabla de verdad [tabla de función] de una función booleana. Está diseñada de tal manera que facilita la agrupación de los términos mínimos [o maxterms] que puedan combinarse para obtener la función más simplificada. También se puede decir que el mapa-K es un conjunto universal dividido en tantas partes [subconjuntos] como el máximo número de minterms [o maxterms] pueda tener la función booleana. Entonces, para una función de n-variables de entrada, el mapa-K tendrá 2^n subconjuntos, donde cada subconjunto representa un minterm [o maxterm].

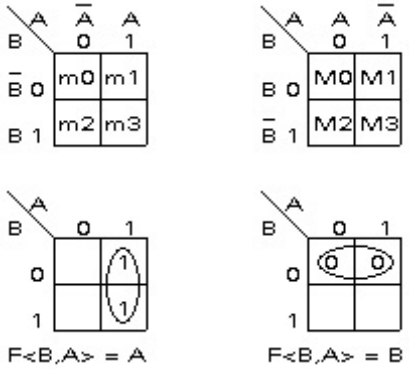
El método de simplificación mediante el mapa-K, utiliza los siguientes teoremas básicos del álgebra de Boole.

$$XY + X\overline{Y} = X \qquad \text{y} \qquad (X + Y)(X + \overline{Y}) = X$$

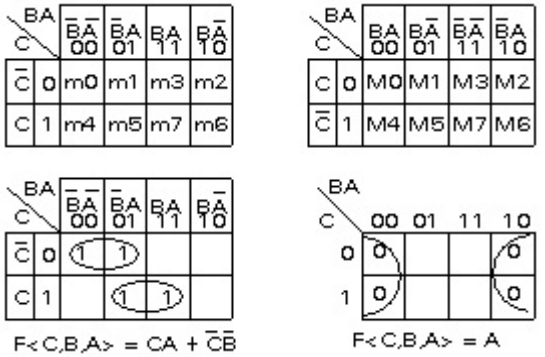
Cuando se elabora el mapa-K, debe tenerse en cuenta

esto para facilitar la agrupación de los minterms [o maxterms], de manera que genere la función más simplificada posible. Para cada variable de salida se debe elaborar un mapa-K. A continuación se presentan algunos ejemplos de cómo construir el mapa-K, para diferente número de variables de entrada.

Ejemplos con 2-variables

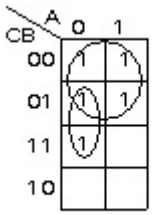


Ejemplos con 3-Variables

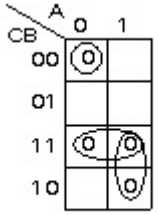


CB \ A	$\bar{A}$ 0	A 1
$\bar{C}\bar{B}$ 00	m0	m1
$\bar{C}B$ 01	m2	m3
CB 11	m6	m7
$C\bar{B}$ 10	m4	m5

CB \ A	A 0	$\bar{A}$ 1
CB 00	M0	M1
$C\bar{B}$ 01	M2	M3
$\bar{C}\bar{B}$ 11	M6	M7
$\bar{C}B$ 10	M4	M5



$F1 = B\bar{A} + \bar{C}$

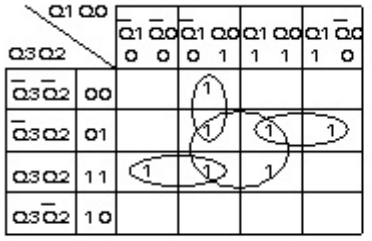


$F2 = (C + B + A)(\bar{C} + \bar{B})(\bar{C} + \bar{A})$

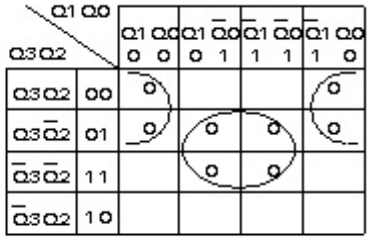
Ejemplos con 4-Variables

Q3Q2 \ Q1Q0	$\bar{Q}1\bar{Q}0$ 0 0	$\bar{Q}1Q0$ 0 1	Q1Q0 1 1	$Q1\bar{Q}0$ 1 0
$\bar{Q}3\bar{Q}2$ 00	m0	m1	m3	m2
$\bar{Q}3Q2$ 01	m4	m5	m7	m6
Q3Q2 11	m12	m13	m15	m14
$Q3\bar{Q}2$ 10	m8	m9	m11	m10

Q3Q2 \ Q1Q0	$\bar{Q}1\bar{Q}0$ 0 0	$\bar{Q}1Q0$ 0 1	Q1Q0 1 1	$Q1\bar{Q}0$ 1 0
$\bar{Q}3\bar{Q}2$ 00	M0	M1	M3	M2
$\bar{Q}3Q2$ 01	M4	M5	M7	M6
Q3Q2 11	M12	M13	M15	M14
$Q3\bar{Q}2$ 10	M8	M9	M11	M10



$F = \bar{Q}3\bar{Q}1Q0 + \bar{Q}3Q2Q1 + Q3Q2\bar{Q}1 + Q2Q0$



$G = (Q3 + Q0)(\bar{Q}2 + \bar{Q}0)$

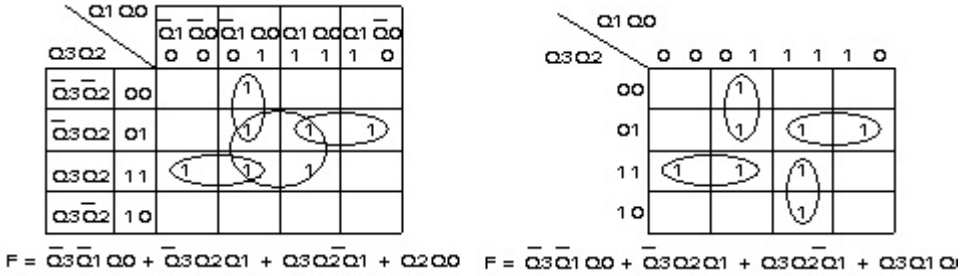
Reglas para la Agrupación de Términos Mínimos

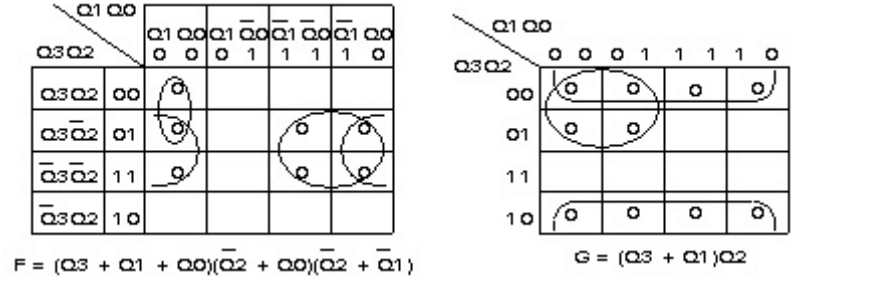
1 Las agrupaciones solo pueden ser de 2^n cuadros adyacentes, donde n es cualquier entero positivo inclusive cero.
2 Cada cuadro que contenga un 1_L, debe tomarse en cuenta, por lo menos una vez.
3 Cualquier combinación deberá ser tan grande como sea posible. Así, un cuadro con un 1_L, no será tomado en cuenta separadamente si se lo puede considerar como parte de 2-cuadros adyacentes; un grupo de 2-cuadros adyacentes, no será considerado separadamente, si se lo puede combinar en una agrupación de 4-cuadros adyacentes; etc.
4 A todos los 1s se los debe tomar en cuenta en un mínimo número de grupos de cuadros adyacentes.

Algoritmo para Encontrar la Función Mínima

1 Identifique con un círculo todos los cuadros que tengan un 1_L y que no puedan combinarse con ningún otro cuadro.
2 Identifique todos los cuadros que tengan un 1_L que puedan agruparse sólo con otro cuadro. Use este par para formar grupos de dos cuadros.
3 Identifique todos los cuadros que puedan combinarse en grupos de cuatro de una sola manera, siempre que todos los cuadros no hayan sido cubiertos por las agrupaciones del paso 2. Use estos cuadros para formar grupos de 4-cuadros. Deje los cuadros que puedan combinarse en más de una forma hasta más tarde.
4 Repita el proceso de combinación para grupos de 8-cuadros, siempre y cuando todos los cuadros del grupo no hayan sido tomados en cuenta antes [en otras agrupaciones].

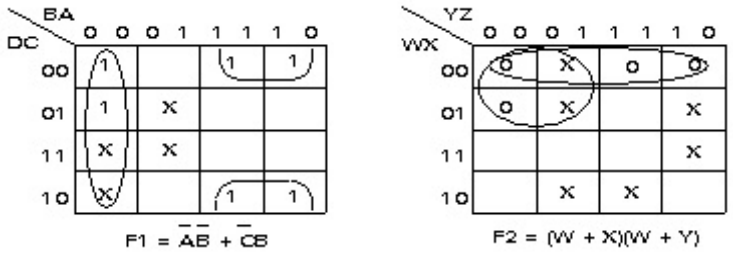
5 Luego investigue los cuadros a los que no se les haya asignado un grupo. Arbitrariamente forme los grupos más grandes que se puedan formar y que cubran la mayoría de los cuadros no cubiertos. Añada los suficientes términos hasta que todos los cuadros estén cubiertos.





Funciones Incompletamente Especificadas.- Cuando se diseña un circuito lógico, hay ocasiones en las cuales, ciertas condiciones de las entradas no producirán ningún efecto en las salidas, en este caso, "***no importa***" si la salida es un 0 o un 1. En otros casos, puede darse que ciertas condiciones de las entradas

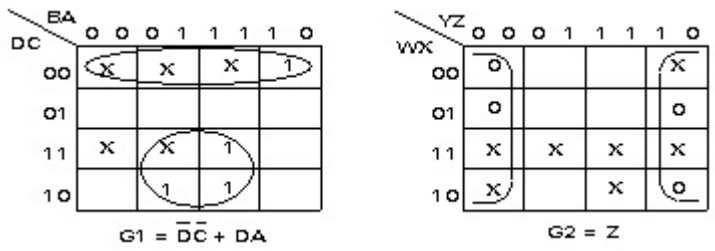
nunca ocurran, [esto puede deberse a algunas restricciones de las entradas].



Cuando se presenta este tipo de situación, se dice que esa salida no está especificada. Esto se indica anotando una "X" o con una "d" del inglés "**don't care**" [como valor funcional, en vez de 0 o 1] en la tabla de verdad o en el mapa-K.

En los ejemplos anteriores, se puede ver que el diseñador de circuitos puede hacer que una condición "no importa" sea 0 o 1 para producir la función lógica más simplificada [de acuerdo a lo que más convenga en una función dada]. De la misma manera, se ve que no es necesario utilizar todas las condiciones "no importa", sino exclusivamente aquellas que contribuyan a la máxima simplificación de la función.

Otros ejemplos



Diseño de Circuitos Combinacionales.- Los circuitos combinacionales, como se indica en la fig. 1.35, pueden tener algunas variables de entrada [desde X_0 hasta X_n] y algunas variables de salida {desde Y_0 hasta Y_m}. Cada una de las salidas depende exclusivamente del valor actual de las variables de entrada.

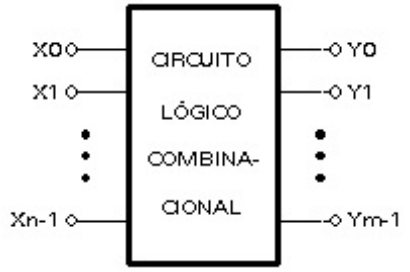


FIGURA 1.35 CIRCUITO LÓGICO COMBINACIONAL

El proceso de diseño empieza por el planteamiento del problema, en forma clara y bien definida. Con esa información se elabora una tabla de función, en la que se indicará con 1_L cuando se cumple la condición del planteamiento y con 0_L cuando no cumple

[la asignación inversa también es válida]. Una vez obtenida la tabla, el siguiente paso consiste en simplificar la función, utilizando cualquiera de los métodos de simplificación que el usuario conozca, lo importante es obtener el circuito más simplificado, que utilice el menor número de compuertas y de CIs. Finalmente se implementa el circuito simplificado. Los dispositivos lógicos para el diseño combinacional son las compuertas [conectivos] que se han estudiado [A-O-N, NAND o NOR].

Ejemplo.- Diseñar un circuito lógico que acepte un dígito decimal codificado en binario [BCD] y cuya salida esté expresada en Exceso-3 [fig. 1.36].

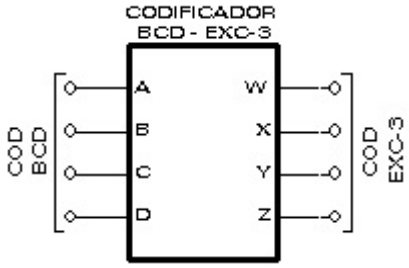


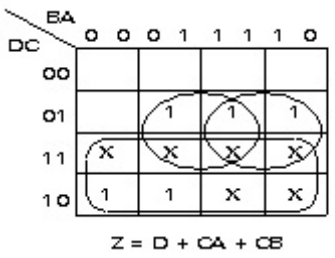
FIGURA 1.36 CONVERTIDOR DE BCD-A-EXCESO DE 3

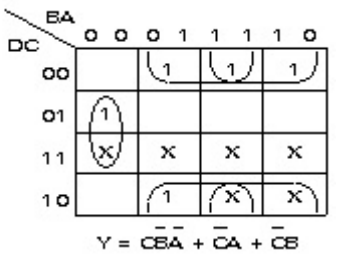
En la tabla de función se ve que, a pesar de que el código BCD solamente acepta las diez primeras combinaciones de 0s y 1s, se muestran todas las 16-combinaciones posibles de las variables de entrada, las condiciones de entrada que no corresponden al

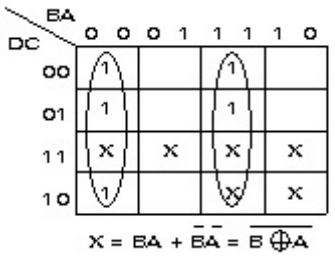
código BCD, se expresan como condiciones "no importa" en la salida correspondiente. Se procede de esta manera para obtener las ecuaciones más simplificadas, también se observa que en las variables de salida se ha usado condiciones "no importa" en las combinaciones no válidas del código BCD.

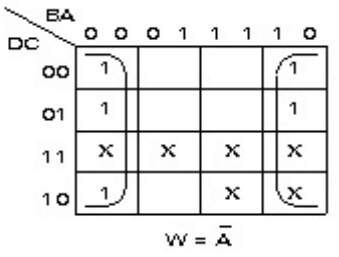
	BCD				EXC-3			
	D	C	B	A	Z	Y	X	W
	0	0	0	0	0	0	1	1
	0	0	0	1	0	1	0	0
	0	0	1	0	0	1	0	1
	0	0	1	1	0	1	1	0
	0	1	0	0	0	1	1	1
	0	1	0	1	1	0	0	0
	0	1	1	0	1	0	0	1
	0	1	1	1	1	0	1	0
	1	0	0	0	1	0	1	1
	1	0	0	1	1	1	0	0
NO EXIS-TEN EN BCD	1	0	1	0	X	X	X	X
	1	0	1	1	X	X	X	X
	1	1	0	0	X	X	X	X
	1	1	0	1	X	X	X	X
	1	1	1	0	X	X	X	X
	1	1	1	1	X	X	X	X

Los siguientes mapas-K permiten obtener las ecuaciones booleanas simplificadas del circuito que se está diseñando.









El circuito combinacional que genera el código Exceso-3, a partir de código BCD, se muestra en la fig. 1.37.

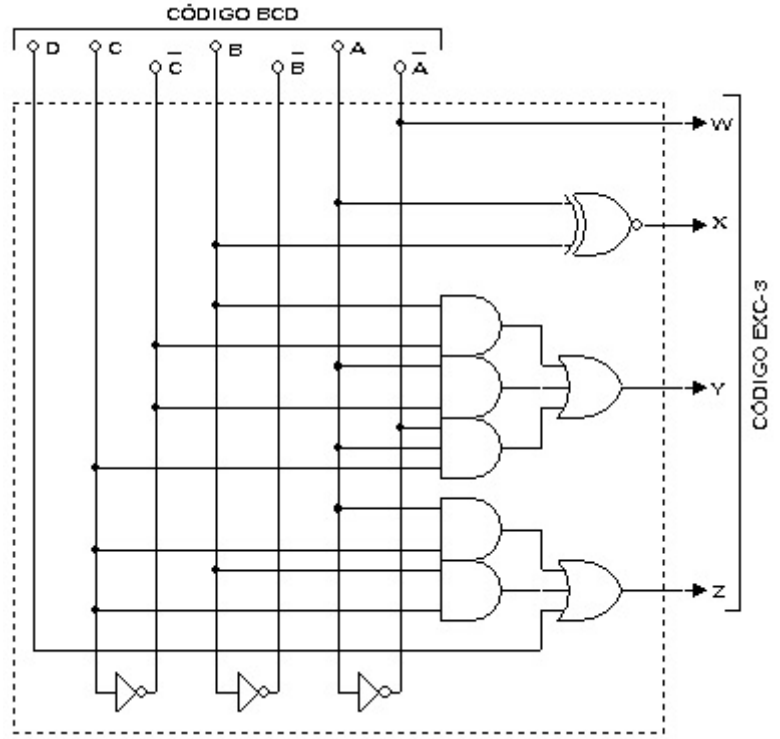


FIGURA 1.37 CONVERTIDOR DE BCD - A - EXC-3

Problemas propuestos.-

1) Demostrar la universalidad de las compuertas NOR.

2) Utilizando exclusivamente los teoremas del Álgebra de Boole, simplificar las siguientes funciones lógicas. Implementar las funciones originales y las simplificadas con compuertas A-O-N.

$$F = DCBA + DCB\overline{A} + DC\overline{B}A + DC\overline{B}\,\overline{A}$$

$$G = (D+B+C+A)\,(D+\overline{C}+B+A)\,(D+C+B+\overline{A})\,(D+C+\overline{B}+\overline{A})$$

2) Simplificar las siguientes funciones lógicas. Implementar las funciones originales y las simplificadas con compuertas A-O-N y con compuertas NAND.

$$F_1 < D,C,B,A > = \sum m(1,3,4,8,9,11,14) + \sum d(6,7,12)$$

$$F_2 = \sum m(0,1,3,9,11,12,15,18,24,25) + \sum d(2,13,14,29,30,31)$$

C:\~\DIGITALES\SD-Cp1BC.wpd

Revisión: Septiembre - 2008

Dispositivos Lógicos MSI

(Dispositivos Lógicos con Salidas Múltiples)

Hasta aquí se han estudiado los conectivos lógicos [compuertas] básicos y se los ha utilizado para implementar circuitos digitales combinacionales simples. Los circuitos que se estudiarán en este capítulo se los clasifica como ***circuitos lógicos combinacionales*** porque, en cualquier instante, el nivel lógico de la salida depende de la combinación de los niveles lógicos presentes en las entradas. Un circuito combinacional no tiene características de memoria, es decir, su salida sólo depende del valor actual de sus entradas.

Mediante el uso de compuertas básicas se pueden implementar circuitos combinacionales más complejos que realizan funciones prácticas como suma aritmética binaria, comparación de la magnitud de dos operandos, etc. Estas y otras funciones se encuentran disponibles en el mercado en forma de CIs y reciben el nombre genérico de dispositivos lógicos MSI. Con la aparición de este tipo de integración, el método tradicional de diseño digital, queda limitado. El diseño con CIs MSI se basa en el conocimiento de las funciones disponibles en el mercado y la forma de utilizarlas eficazmente.

DEFINICIONES

Circuitos SSI [Small Scale Integration = Integración en Pequeña Escala].- Este grupo incluye las compuertas y los flip-flops elementales. Para el diseño con estos elementos hay que tomar en cuenta el número de compuertas y flip-flops que se utilizarán en un circuito digital dado. Esta tecnología hace énfasis en la necesidad de simplificar o minimizar el número total de compuertas y FFs requeridos. Generalmente cada circuito integrado [CI], contiene desde una hasta alrededor de 12 compuertas o una circuitería de igual complejidad.

Dispositivos MSI [Medium Scale Integration = Integración en Mediana Escala].- Es un concepto utilizado para definir un subsistema o un sistema funcional completo, implementado en un mismo microcircuito [CI]. Se considera que este tipo de integración contiene de 12 a 100 compuertas o el equivalente a una circuitería de igual complejidad. Comprende, entre otros: decodificadores, multiplexers, contadores, comparadores de magnitud, sumadores, registros de desplazamiento, etc. Generalmente el diseño debe ajustarse a los circuitos MSI existentes.

Dispositivos LSI [Large Scale Integration = Integración

en Gran Escala].- Este es un concepto utilizado para definir un subsistema o un sistema funcional completo más grande, fabricado en un mismo microcircuito. Se considera que contiene desde 100 hasta 10000 compuertas o circuitos de similar complejidad. Comprende las memorias $R/\overline{W}$ -RAM, ROM y PLAs. En este caso hay que tomar en consideración el número de bytes de memoria y el número de elementos de soporte. Este método de diseño reemplaza toda una circuitería lógica por elementos de memoria.

Dispositivos VLSI [Very Large Scale Integration = Integración en muy Alta Escala].- Son circuitos lógicos muy complejos con un equivalente de 10000 a 100000 compuertas básicas. Esta tecnología comprende los microprocesadores y los micro-controladores. Para diseñar con circuitos VLSI hay que tomar en cuenta el número de instrucciones y el número de circuitos de soporte. Se podría decir que es un dispositivo "inteligente", controlado por un programa almacenado en una memoria.

Dispositivos ULSI [Ultra Large Scale Integration = Integración en Altísima Escala].- Microprocesadores y microcontroladores de mayor capacidad y complejidad. Tienen un equivalente de más de 100000 compuertas.

Dispositivos GSI [Giga Scale Integration].- Micro-

controladores de muy alta capacidad que trean incluidos: memoria de programa, memoria de datos y puertos de entrada/salida. Tienen un equivalente de más de 1000 000 de compuertas.

Dispositivos Combinacionales MSI.- Dentro de los dispositivos MSI comerciales, en este capítulo se estudiarán, de entre los más importantes, los siguientes.

- ➢ Decodificador de BCD-a-7 segmentos
- ➢ Sumador Aritmético Binario
- ➢ Multiplexer digital
- ➢ Demultiplexer/Decoder
- ➢ Codificadores de prioridad
- ➢ Comparadores de magnitud
- ➢ Chequeador/Generador de paridad

Decodificadores de BCD-a-7 Segmentos.- Una gran parte del equipo digital cuenta con algún medio para presentar información de manera que el operador o el usuario puedan entenderla fácilmente.

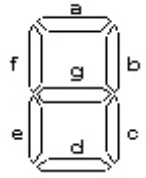


FIGURA 2.1
Arreglo de 7 segmentos

Esta información por lo general es de carácter numérico [aunque puede ser alfanumérica]. Una forma sencilla de mostrar información numérica utiliza un arreglo de 7-segmentos [a, ..., g], como el que se muestra en la fig. 2.1; con este arreglo se pueden formar los dígitos del 0 al 9; para esto, normalmente, se emplea un LED para cada segmento. Para formar los respectivos dígitos, algunos LEDs se encienden, mientras que otros permanecen apagados. Para comandar el encendido y apagado de los LEDs, se emplean decodificadores. Un circuito lógico muy utilizado para comandar un display de 7 segmentos y presentar valores decimales es el decodificador de BCD-a-7 segmentos. Existen dos tipos de arreglo, por tanto, hay dos tipos de decodificadores de BCD-a-7 segmentos: 1) los ánodos de los 7 LEDs que forman el arreglo están conectados en un mismo punto [display de 7 segmentos de ánodo común] y 2) los cátodos de los 7 LEDs que forman el arreglo están conectados en un mismo punto [display de 7 segmentos de cátodo común]. Cada uno de estos arreglos requiere un decodificador especial.

Decodificador de BCD-a-7 Segmentos de Ánodo Común (CI-7446/47).- Este tipo de decodificador [fig. 2.2] sirve para comandar un display de 7-segmentos de ánodo común, [fig. 2.3]. Todos los ánodos están conectados a V_{CC} [+5V], mientras que los cátodos son independientes y representan cada uno de los

segmentos del display. Estos cátodos deben ser comandados por un decodificador de 7-segmentos, que enviará la información necesaria para que se enciendan los segmentos adecuados para la formación de cada dígito decimal, como se muestra en la tabla de función del CI-7446/47. Para evitar que el LED se destruya, se requiere limitar la corriente que circula por él, para ello, generalmente se utiliza una resistencia de 330Ω aproximadamente para cada segmento [7 resistencias en total].

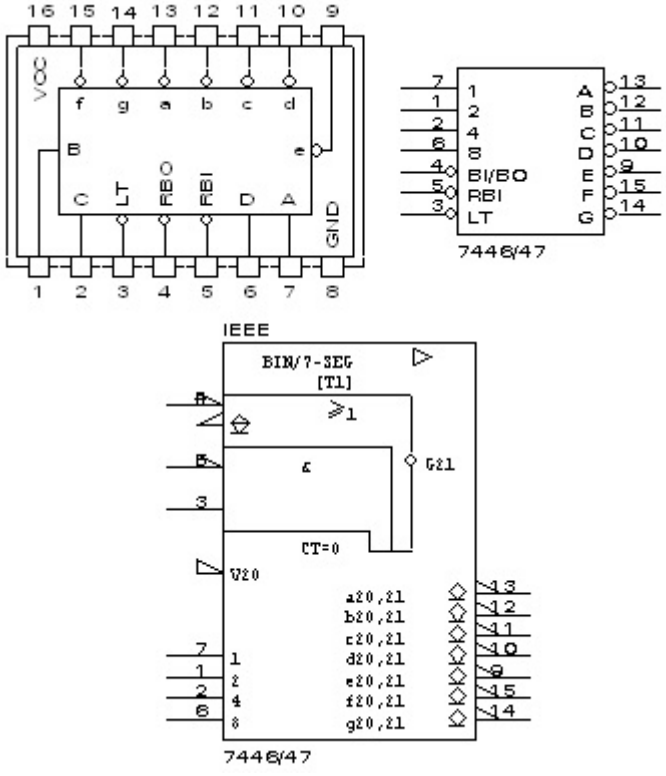


FIGURA 2.2 CI-7446/47 DECODIFICADOR DE BCD A 7-SEGMENTOS DE ÁNODO COMÚN

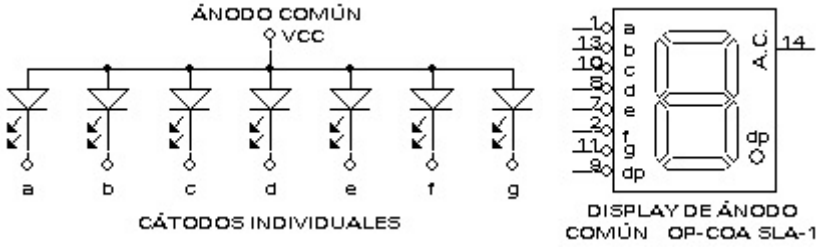


FIGURA 2.3 ARREGLO DE 7 LEDs CUYOS ÁNODOS ESTÁN CONECTADOS A UN PUNTO COMÚN

DECIMAL O FUNCIÓN	ENTRADAS						BI/ BO	SALIDAS							NOTA
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
O	1	1	0	0	0	0	1	O	O	O	O	O	O	F	
1	1	X	0	0	0	1	1	F	O	O	F	F	F	F	
2	1	X	0	0	1	0	1	O	O	F	O	O	F	O	
3	1	X	0	0	1	1	1	O	O	O	O	F	F	O	
4	1	X	0	1	0	0	1	F	O	O	F	F	O	O	
5	1	X	0	1	0	1	1	O	F	O	O	F	O	O	
6	1	X	0	1	1	0	1	F	O	O	O	O	O	O	
7	1	X	0	1	1	1	1	O	F	O	F	F	F	F	
8	1	X	1	0	0	0	1	O	O	O	O	O	O	O	1
9	1	X	1	0	0	1	1	O	O	O	F	F	O	O	
1O	1	X	1	0	1	0	1	F	F	F	O	O	F	O	
11	1	X	1	0	1	1	1	F	F	O	O	F	F	O	
12	1	X	1	1	0	O	1	F	O	F	F	F	O	O	
13	1	X	1	1	0	1	1	O	F	F	O	F	O	O	
14	1	X	1	1	1	O	1	F	F	F	O	O	O	O	
15	1	X	1	1	1	1	1	F	F	F	F	F	F	F	
BI	X	X	X	X	X	X	0	F	F	F	F	F	F	F	2
RBI	1	0	0	0	0	0	0	F	F	F	F	F	F	F	3
LT	0	X	X	X	X	X	1	O	O	O	O	O	O	O	4

TABLA DE FUNCIÓN DEL CI-7446/47

Nota 1. La entrada BI (Blanking-Input) puede estar abierta o mantenida en un nivel lógico alto, cuando en la salida se necesitan las funciones 0-15. RBI

(Ripple-Blanking-Input), debe estar abierta o alta si no se desea blanquear un cero decimal.

Nota 2. Cuando se aplica un nivel lógico bajo directamente a la entrada Blanking-Input (BI), todas las salidas de los segmentos se apagan, sin importar el nivel de cualquiera de las entradas.

Nota 3. Cuando la entrada Ripple-Blanking Input (RBI) y las entradas D, C, B y A están en un nivel bajo con la entrada prueba de lámparas (Lamp Test, LT) en nivel alto, todas las salidas de los segmentos pasan a apagado (Off) y la salida Ripple-Blanking Output (RBO) pasa a un nivel bajo (condición de respuesta).

Nota 4. Cuando la salida Blanking Input/Ripple Blanking Output (BI/RBO) está abierta o se mantiene en un nivel alto y se aplica un nivel bajo a la entrada Lamp-Test, todas las salidas de los segmentos se encienden (On).

En la tabla anterior [del CI-7447], O = On, F = oFf. La fig. 2.4 muestra los resultados que se obtienen en el display para las diferentes combinaciones binarias de las entradas: D = 8, C = 4, B = 2 y A = 1, de acuerdo con la tabla del CI-7447.

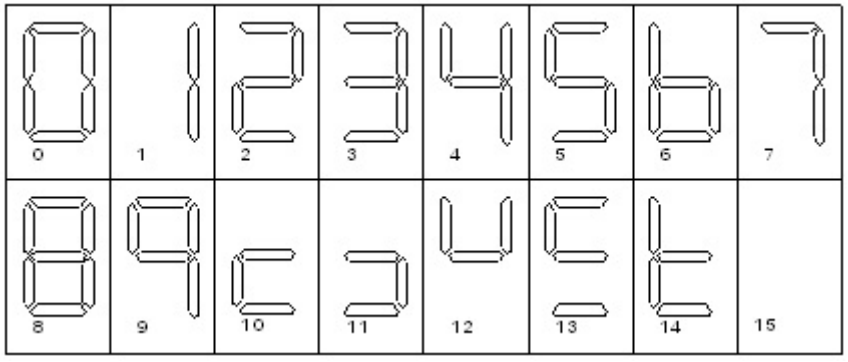


FIGURA 2.4 RESULTADO DE LO QUE SE VE EN EL DISPLAY CUANDO SE INGRESA UN CÓDIGO BINARIO DE 4-Bits

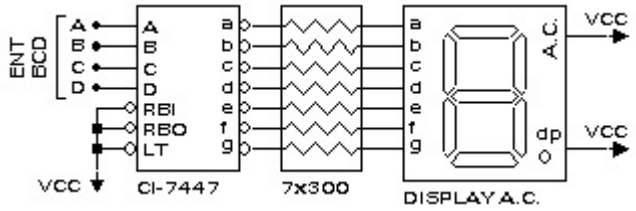


FIGURA 2.5 FORMA DE CONECTAR UN DECODIFICADOR DE BCD-A-7 SEGMENTOS DE ÁNODO COMÚN

La fig. 2.5 muestra la conexión del decodificador CI-7447 con un display de 7 segmentos de ánodo-común [las resistencias son para limitar la corriente que circula por los LEDs]. Se observa que cuando no se utilizan las entradas LT, RBI y RBO, se las conecta a VCC.

Decodificador de BCD-A-7 Segmentos de Cátodo Común (CI-7448).- Este decodificador [fig. 2.6] sirve para comandar un display de 7-segmentos de cátodo común, como se muestra en la fig. 2.7. Todos los cátodos están conectados a tierra [GND], mientras que los ánodos son independientes y representan cada uno de los segmentos del display. Al igual que antes, también es necesario limitar la corriente que circula por el LED que se enciende, para ello se utiliza una resistencia de 330Ω aproximadamente para cada segmento.

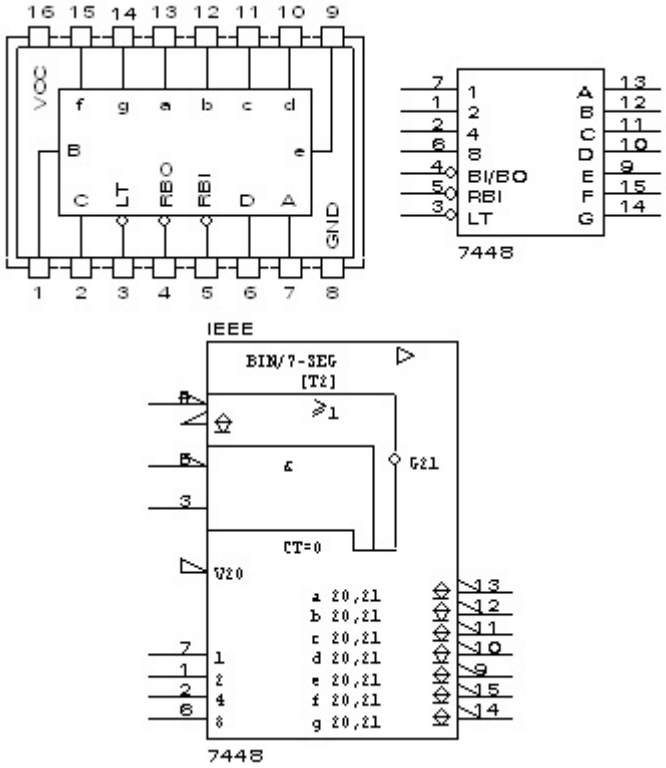


FIGURA 2.6 CI-7448 DECODIFICADOR DE BCD A 7-SEGMENTOS DE CÁTODO COMÚN

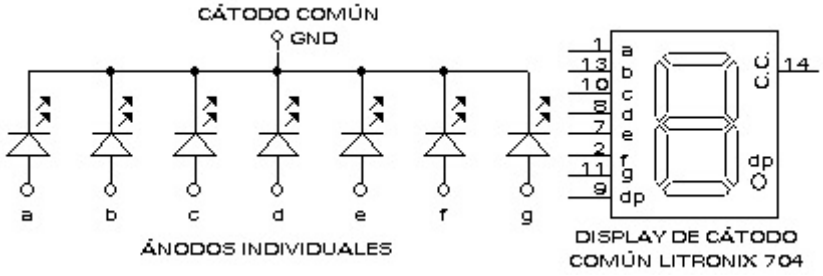


FIGURA 2.7 ARREGLO DE 7 LEDs CUYOS CÁTODOS ESTÁN CONECTADOS A UN PUNTO COMÚN

La fig. 2.8 muestra la forma de conectar el decodificador CI-7448 con un display de 7 segmentos de cátodo-común. También aquí se observa que cuando no se utilizan las entradas LT, RBI y RBO, se las conecta a VCC.

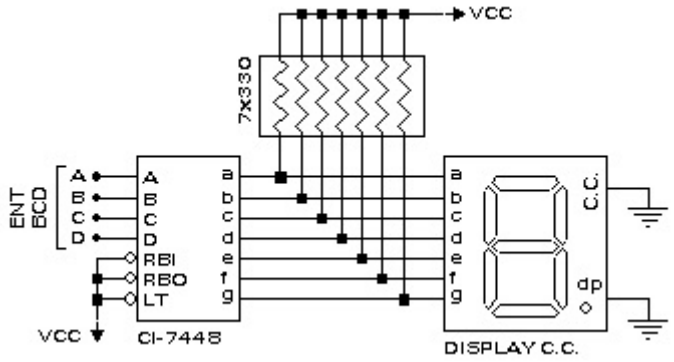


FIGURA 2.8 CONEXIONES DE UN DECODIFICADOR BCD-A-7 SEGMENTOS DE CÁTODO COMÚN

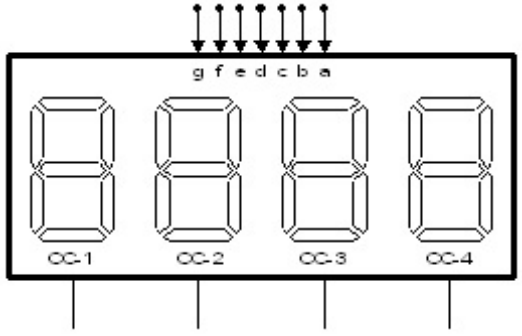


FIGURA 2.9 ARREGLO DE 4 Displays

Displays Multiplexados.- En muchas ocasiones es necesario comandar varios displays pero con un sólo

decodificador de BCD-a-7 segmentos; en esos casos se utilizan los arreglos de displays en forma multiplexada. Un ejemplo se muestra en la fig. 2.9, en la que se han ensamblado 4-displays de 7-segmentos de cátodo común. Aquí, los segmentos ***a*** de cada display están conectados en forma común, lo que mismo ocurre con los segmentos ***b***, ***c***, etc.

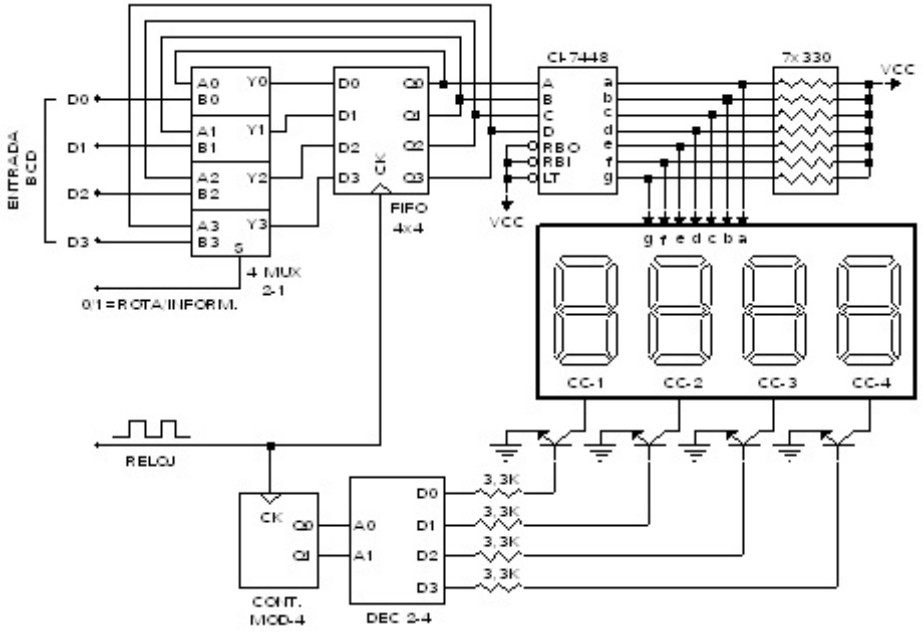


FIGURA 2.10 CONEXIONES DE UN ARREGLO DE DISPLAYS DE CÁTODO COMÚN

En este tipo de display, sólo debe encenderse un dígito cada vez, mientras que los otros están apagados o desactivados, para comandar esta situación, se utilizan transistores NPN, conectados

en la forma que se muestra en la fig. 2.10. El contador módulo-4, conjuntamente con el decodificador de 2-a-4, sirve para asegurar que sólo se active uno de los displays, mientras los otros permanezcan apagados. El arreglo de 4-MUX de 2-a-1 sirve para permitir el ingreso de nueva información BCD cuando la entrada de selección está en 1L o para mostrar la información previamente almacenada en la memoria FIFO [First In - First Out], cuando la entrada de selección es 0L. La información para cada display sale de la memoria FIFO y debe ingresar por las entradas [D, C, B, A] del decodificador, la memoria debe estar sincronizada con el contador comandado por el reloj [oscilador] de barrido libre.

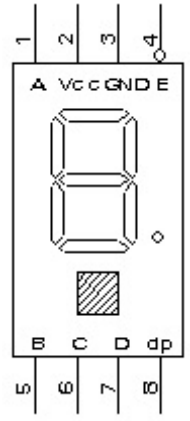


FIGURA 2.11

Displays con Decodificador Incluido.- La fig. 2.11 muestra un display de 7-segmentos que incluye, en el mismo

chip, el decodificador y las resistencias limitadoras de corriente. Este es un CI muy práctico porque ocupa poco espacio y disminuye las conexiones externas, la desventaja es que resulta costoso. En el mercado existen algunos tipos de tales dispositivos.

Til311 Display Hexadecimal con Decodificador.- El display y la lógica MSI-TTL se encuentran en el mismo CI. Contiene un retenedor de 4-bits, un decodificador, y un arreglo de 4x7 LEDs para los caracteres y 2-LEDs más para punto decimal [dp] comandados externamente [fig. 2.12]. Este decodificador acepta un número binario de 4-bits y muestra la información en hexadecimal.

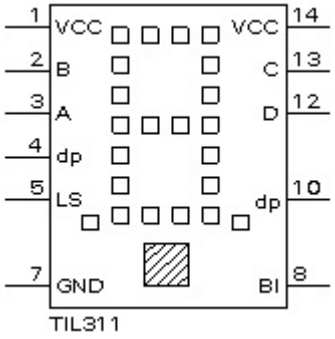


FIGURA 2.12 TIL311

Descripción de los pines	
Pin 1	Fuente de polarización para los LEDs [V_{cc}]
Pin 2	Retenedor para la entrada del dato B [Latch Strobe]
Pin 3	Retenedor para la entrada del dato A
Pin 4	LED para el punto decimal [izquierda]

Pin 5	Retenedor para habilitar la entrada
Pin 6	Omitido [No se utiliza]
Pin 7	Tierra común [GND]
Pin 8	Entrada para blanqueo [Blanking Input]
Pin 9	Omitido [No se utiliza]
Pin 10	LED para el punto decimal [derecha]
Pin 11	Omitido [No se utiliza]
Pin 12	Retenedor para la entrada del dato D
Pin 13	Retenedor para la entrada del dato C
Pin 14	Fuente de polarización para la lógica MSI-TTL [V_{CC}]

FUNCIÓN	PIN N°	DESCRIPCIÓN
Retenedor para habilitar la entrada [LS = Lactch Strobe Input]	5	Cuando es 0L, el dato en los retenedores de datos sigue las variaciones de los datos de entrada. Cuando es 1L, el dato de los retenedores no cambia. Si se blanquea el display y entonces se lo restablece, mientras la entrada de habilitación está en 1L, se vuelve a presentar el carácter anterior.
Entrada para blanqueo [BI = Blanking Input]	8	Cuando es 1L, el display se blanquea sin importar el nivel de las otras entradas. Cuando es 0L, se muestra un carácter que está determinado por el dato de los retenedores. La entrada de blanqueo puede utilizarse para modular la intensidad.

Retenedores para las entradas de datos [D, C, B, A]	12, 13, 2, 3	Los datos de estas entradas ingresan a los retenedores cuando la entrada de habilitación está en 0L. La ponderación de estas entradas es: D = 8; C = 4; B = 2; A = 1
LEDs para el punto decimal	4, 10	Estos LEDs no están conectados a la lógica del chip. Si se utiliza un punto decimal, se debe conectar una resistencia externa u otro mecanismo de limitación de corriente en serie con el pin.
Fuente de polarización para los LEDs	1	Esta conexión permite usar una fuente DC regulada separada para polarizar los LEDs, o se puede conectar externamente a V_{CC}.
Fuente de polarización para la lógica	14	Conexión a V_{CC} para la lógica del chip.
Tierra común [GND]	7	Este es el terminal negativo para toda la lógica interna y para los LEDs de los caracteres, excepto para los puntos decimales.

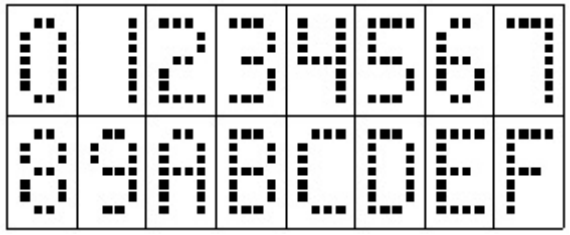


FIGURA 2.13 RESULTADOS EN EL TIL311, DISPLAY HEXADECIMAL

Los resultados que se observan en el display TIL311 para los valores de los datos binarios en

los retenedores de entrada se muestran en la fig. 2.13.

Dispositivos de Cristal Líquido (Liquid Crystal Device LCD) .- Los displays del tipo LCD [fig. 2.14] requieren de una señal alterna de bajo voltaje [3 - 15 V_{RMS}] y de baja frecuencia [25 - 60 Hz], consumen muy poca energía.

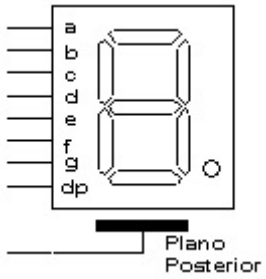


FIGURA 2.14 DISPLAY LCD

El voltaje ac que se necesita para encender un ***segmento***, se aplica entre el segmento y el ***plano posterior***, que es común a todos los segmentos. El segmento y el plano posterior forman un capacitor que requiere muy poca corriente si la frecuencia ac se mantiene baja, no inferior a 25Hz, porque esto produciría un parpadeo visible. Puesto que necesitan mucha menos energía que los LEDs, los LCDs se utilizan ampliamente en dispositivos que trabajan con baterías. Los LCDs no emiten luz sólo la "reflejan", por eso requieren una fuente de luz

externa.

Manejo de un LCD.- Un segmento LCD se "activa" cuando se le aplica un voltaje ac entre el segmento y el plano posterior y se "apaga" cuando no hay voltaje entre esos terminales. En vez de generar una señal ac para un segmento, es práctica común producir el voltaje requerido aplicando ondas cuadradas desfasadas entre el segmento y el plano posterior, como se muestra en la fig. 2.15.

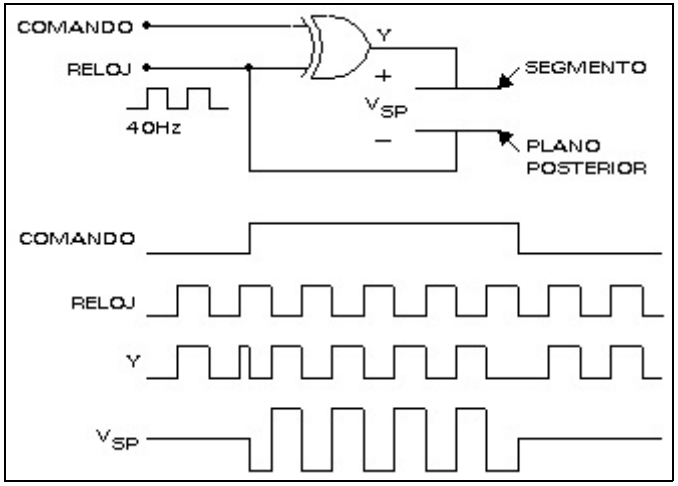


FIGURA 2.15 FORMA DE COMANDAR UN SEGMENTO LCD

Una onda cuadrada de 40Hz se aplica al plano posterior y también a la entrada de una compuerta XOR. La otra entrada a la compuerta XOR sirve para comandar el "encendido" y "apagado" del segmento. Cuando la entrada de comando está en un nivel bajo

(0_L), la salida de la XOR será exactamente la misma que la onda de entrada de 40Hz; la diferencia de potencial entre el segmento y el plano posterior (Vsp) es cero, entonces el segmento se "apaga". Cuando la entrada de comando está en un nivel alto (1_L), la salida de la compuerta XOR es el complemento de la onda de entrada de 40Hz, entonces la diferencia de potencial entre el segmento y el plano posterior varía entre +5V y -5V a una frecuencia de 40Hz, esta señal permite el "activado" del segmento.

La fig. 2.16 muestra un display LCD comandado por un decodificador de BCD-a-7 segmentos tipo MOSFET.

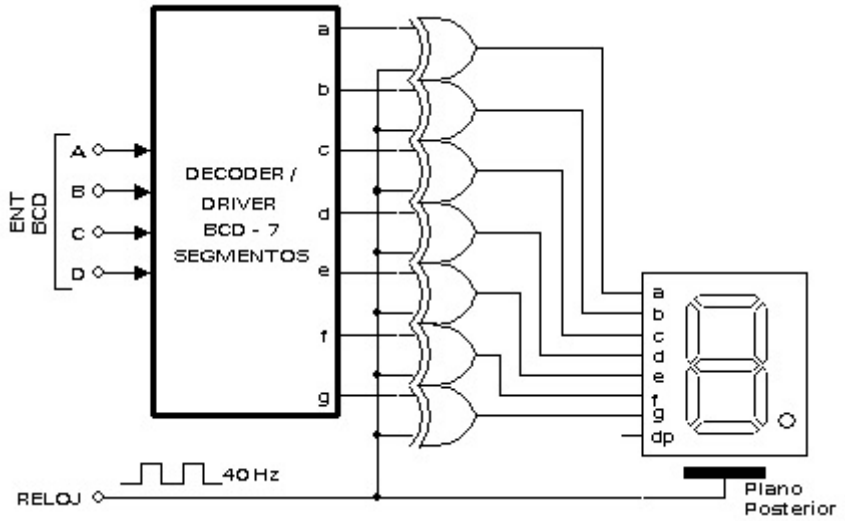


FIGURA 2.16 CIRCUITO PARA COMANDAR UN DISPLAY DE CRISTAL LÍQUIDO

Los cristales líquidos son conjuntos de moléculas orgánicas transparentes y alargadas que tienden a formar redes regulares, pero que se deforman fácilmente. Además, poseen otras características eléctricas y ópticas, las propiedades de la luz que atraviesa una capa de cristal líquido dependen de la orientación de la molécula. La característica eléctrica proporciona el medio para hacer girar las moléculas mediante la aplicación de un campo eléctrico.

Dispositivos de Cristal Líquido.- El término "cristal líquido" presenta una ambigüedad en su nombre que concuerda plenamente con su cualidad de ser una sustancia que exhibe al mismo tiempo características de líquido y de sólido. Esto se debe a que las moléculas de este tipo de sustancia pueden desplazarse unas respecto de otras con mucha facilidad [como en un líquido], pero aun así, tienden a conservar una orientación común, de manera análoga a lo que pasa en un sólido cuando forma estructuras cristalinas. Tienen la facultad de presentar características electro-ópticas, que fueron descubiertas por el año 1970, estas propiedades que presentan algunos líquidos forman cristales que polarizan la luz cuando se los somete a la acción de un campo eléctrico. El ámbito de aplicaciones es muy amplio: relojes digitales, calculadoras, instrumentos de medida, monitores de computadores, etc.

Displays de Plasma.- [Plasma Displays Panel PDP] de manera simple, una celda de plasma ac consiste de dos placas de vidrio separadas por un espacio lleno de gas. En la superficie interior de las placas hay dos conjuntos, horizontal y vertical, de electrodos transparentes cubiertos con una delgada capa aislante.

Esto crea un display monocromático nítido que tiende a emitir un brillo desde el naranja hasta el azul una vez que se excita al gas encerrado; el color depende de la mezcla de gas. Para añadir color, en cada celda se pone algún compuesto de fósforo que emite el color deseado cuando incide luz ultravioleta. Los compuestos de fósforo están separados por pequeños surcos de modo que cuando se aplica voltaje a una capa de fósforo, las otras capas no se excitan a menos que así se desee. Las TV de pantalla plana crean imágenes utilizando una matriz plana y fija de forma cuadrada o rectangular de pixeles [Picture Elements]. Las imágenes que se forman de esta manera, son muy nítidas porque la información del color y la luz de la pantalla se controla digitalmente a nivel de pixel [un punto de la pantalla]. Cada sub-pixel individual, se enciende o se apaga con su propio electrodo. Controlando cuidadosamente el voltaje aplicado, la intensidad de cada sub-pixel puede tener un rango de 256 tonos. Combinando los 3 sub-pixeles, se genera una paleta de 16,7 millones de colores; [256 tonos de rojo x 256 tonos de verde x 256 tonos de azul]. Este increible nivel de precisión, junto con una pantalla totalmente plana, permite que la TV tenga una imagen geométrica perfecta y enfocada de arriba hacia abajo y de esquina a esquina. Otra ventaja de este tipo de pantallas es que no se ven afectadas por campos eléctricos.

Sumador Aritmético Binario.- Una de las operaciones básicas que realiza un computador es la suma aritmética, en base a la cual se pueden realizar operaciones matemáticas más complejas como multiplicación, división, exponenciación, integración, diferenciación, etc., de ahí que resulte importante conocer cómo funciona y cómo diseñar este dispositivo. Para ello se realizará un ejemplo de una suma binaria de dos números de 4-bits cada uno, como se indica a continuación.

	1	**1**	**0**	**1**	←	**C**arry [Exceso]
	0	1	0	1	←	A = A_3 A_2 A_1 A_0
	1	1	0	1	←	B = B_3 B_2 B_1 B_0
Exceso final →	①	0	0	1	0	← S = C_0 S_3 S_2 S_1 S_0

En la operación aritmética del ejemplo anterior, se puede observar que lo primero que se suma son los bits menos significantes [B_0, A_0] lo que genera una suma parcial, que es lo que se escribe [S_0] y un exceso parcial [carry] que se lleva a la siguiente columna. Para la suma de los siguientes bits, se suman los bits de A_i y B_i correspondientes más lo que se trae de la columna anterior C_i. El exceso final es parte de la respuesta.

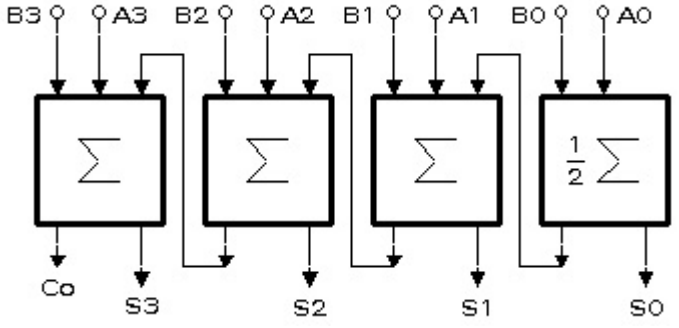


FIGURA 2.17 SUMADOR BINARIO DE 4-Bits

Esta operación puede visualizarse de mejor manera con el diagrama de bloques de la fig. 2.17, como puede verse, el primer bloque del sumador tiene dos entradas de datos: A_0 y B_0 y dos salidas: S_0 y un

exceso parcial C_o que ingresa al siguiente bloque, por eso recibe el nombre de ***medio sumador*** [Half-Adder]; los otros bloques todos ellos tienen tres entradas de datos: A_i, B_i [bits de A y B] y C_i [exceso parcial de la columna anterior] y generan dos salidas: S_i y C_o, cada uno de ellos recibe el nombre de ***sumador completo*** [Full-Adder].

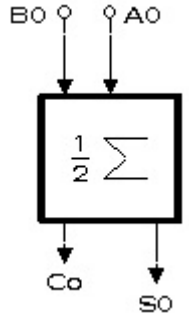
En primer lugar se realizará el ***diseño del medio sumador*** [fig. 2.18], para esto se utilizará la tabla de función que se indica a continuación.

B_o	A_o	S	C_o
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

De aquí se deduce que

$$S_0 = A_0 \oplus B_0$$

$$C_0 = A_0.B_0$$



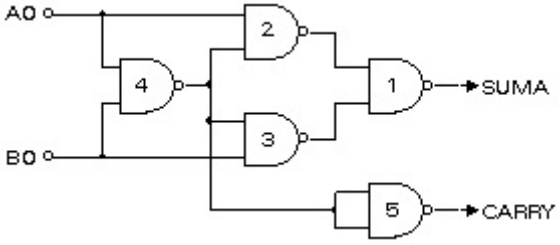
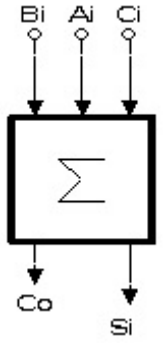


FIGURA 2.19 SUMADOR BINARIO CON COMPUERTAS NAND

La fig. 2.19 muestra el circuito lógico del medio sumador con compuertas NAND.

De igual manera, ***el sumador completo***, cuyo

diagrama de bloques se muestra en la fig. 2.20, se diseña en base a la tabla de función correspondiente.



C_i	B_i	A_i	S_i	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1		1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

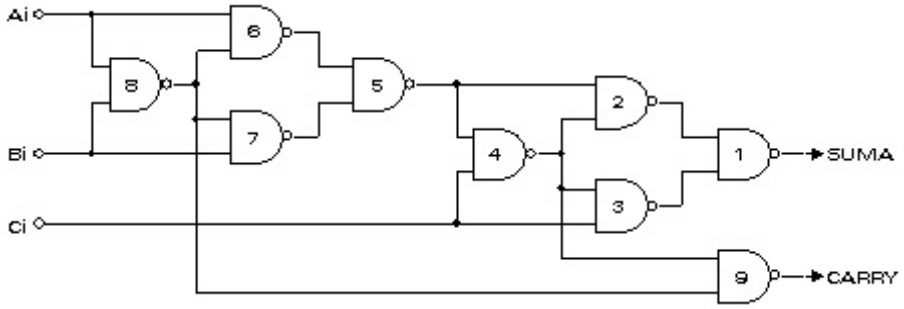


FIGURA 2.21 SUMADOR COMPLETO CON COMPUERTAS NAND

De la tabla se obtienen las ecuaciones booleanas para las funciones de salida.

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_0 = \overline{\overline{A_iB_i}.\overline{(A_i \oplus B_i)C_i}}$$

El circuito lógico de un sumador completo, utilizando compuertas NAND, se muestra en el diagrama de la fig. 2.21.

CI-7483 y CI-74283.- Con los sumadores medio y completo se puede implementar un sumador de cualquier número de bits. En el mercado existen sumadores aritméticos binarios para números de 4-bits, el CI-7483, cuya distribución de pines se muestra en la fig. 2.22 [el CI-7483 (superior) y el CI-74283 (inferior) es el mismo, pero con una distribución de pines deferente].

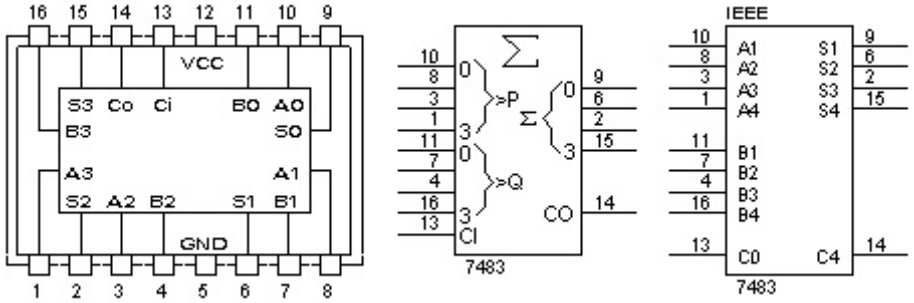


FIGURA 2.22 SUMADORES ARITMÉTICOS BINARIOS DE 4-Bits

En realidad, este CI dispone de 4-sumadores completos, lo que facilita la construcción de sumadores binarios de mayor número de bits. Estos sumadores completos realizan la suma de 2-números de 4-bits. Las salidas de suma [S], se proporcionan para cada bit y el exceso [carry] resultante [C_o]

se obtiene del cuarto bit. Los sumadores se diseñan de manera que los niveles lógicos de las entradas y las salidas, incluso el carry, están en su forma verdadera. Diseñados para media y alta velocidad, los circuitos utilizan lógica TTL [Lógica Transistor-Transistor] de alta velocidad y alto fan-out, pero son compatibles con las familias DTL. La siguiente es la tabla de verdad del sumador de 4-bits.

ENTRADAS				SALIDAS					
				CUANDO CO = 0			CUANDO CO = 1		
				CUANDO C2 = 0			CUANDO C2 = 1		
A1 / A3	B1 / B3	A2 / A4	B2 / B4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4
0	0	0	0	0	0	0	1	0	0
1	1	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	1	0
1	1	0	0	0	1	0	1	1	0
0	0	1	0	0	1	0	1	1	0
1	1	1	0	1	1	0	0	0	1
0	0	1	0	1	1	0	0	0	1
1	1	1	0	0	0	1	1	0	1
0	0	0	1	0	1	0	1	1	0
1	1	0	1	1	1	0	0	0	1
0	0	0	1	1	1	0	0	0	1
1	1	0	1	0	0	1	1	0	1
0	0	1	1	0	0	1	1	0	1
1	1	1	1	1	0	1	0	1	1
0	0	1	1	1	0	1	0	1	1
1	1	1	1	0	1	1	1	1	1

La característica **full look ahead** [exceso anticipado (adelantado)] del CI-7483 en los 4-bits para generar el carry final, típicamente es de 10ns. Estos circuitos se fabrican con inversión-simple, circuitos de conexión darlington con carry serial de alta velocidad dentro de cada bit.

Aplicaciones del Sumador en CI.- Como aplicaciones prácticas del CI-7483 se estudiarán los siguientes casos: a) Sumador binario de 8-bits; b) Sumador-Restador de 4- y 8-bits [incluido el signo] y c) Sumador BCD de 1- y 2-dígitos BCD.

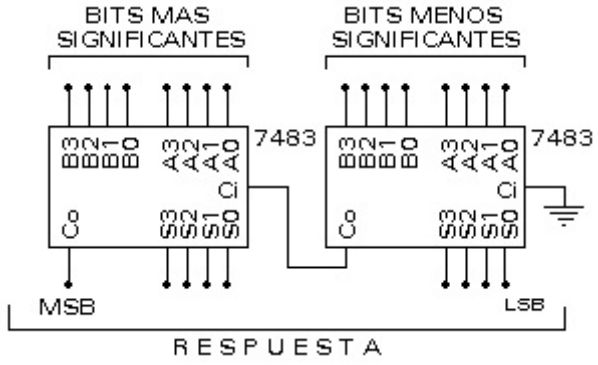


FIGURA 2.23 SUMADOR BINARIO DE 8-Bits

Sumador Binario de 8-bits.- Para implementar este circuito digital se requieren dos CI-7483, conectados de la forma que se indica en la fig. 2.23. De esta manera se puede expandir el número de bits del sumador, el problema que se acumula el retardo de

tiempo para generar el carry final.

Sumador - Restador.- Mediante el uso del CI-7483, se puede implementar un circuito que, en base a una señal externa, pueda sumar cuando la señal externa sea 0_L o restar cuando la señal externa valga 1_L. La fig. 2.24 muestra una forma de hacerlo [complemento a 2].

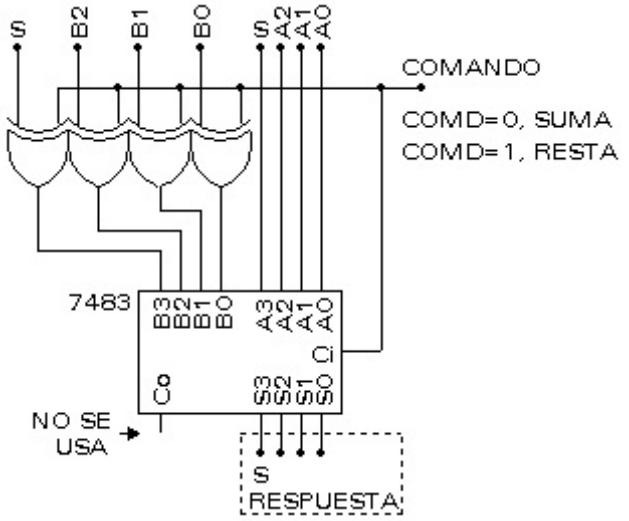


FIGURA 2.24 SUMADOR-RESTADOR DE NÚMERO DE 4-Bits [INCLUIDO EL SIGNO] EN COMPLEMENTO a 2

En este caso la magnitud de los números está definida por los 3-bits menos significantes, el bit "más significante", en realidad, representa el signo [S] de A, B y de la respuesta respectivamente. Las compuertas XOR, permiten dejar que el valor de B

pase sin complemento cuando la entrada de comando sea 0_L [suma] y que se complemente el valor de B cuando la entrada de comando sea 1_L [resta]. En el circuito de la fig.2.24, la respuesta está limitada a ±7 y está expresada en complemento a-2.

La fig. 2.25 es un Sumador-Restador para números de 8-bits, la letra "S" representa el signo de los números A, B y de la respuesta que se limita a ±127.

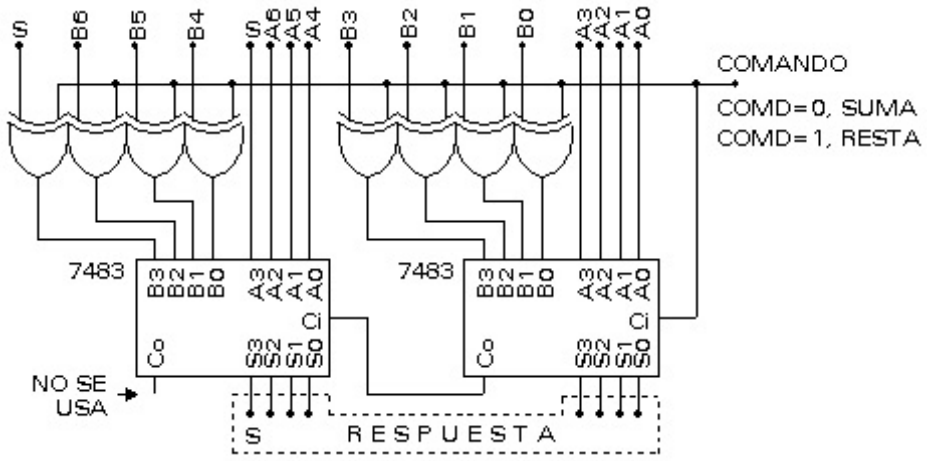
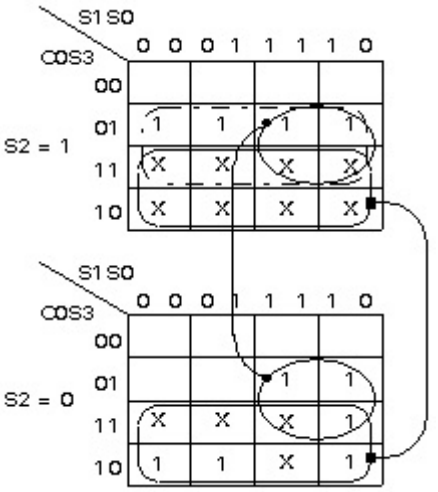


FIGURA 2.25 SUMADOR-RESTADOR DE NÚMEROS DE 8Bits [INCLUIDO EL SIGNO] EN COMPLEMENTO a 2

Sumador BCD.- Muchas veces se deben realizar operaciones aritméticas con datos expresados en BCD y se requiere un resultado también BCD. Sin embargo, el sumador solo produce resultados en binario natural y no en BCD como se necesita. Para obtener el valor BCD correcto se requiere hacer un ajuste. El algoritmo consiste en sumar el valor 0 [0000_2] cuando

el resultado de la suma de A y B sea menor que 10 y sumar 6 [0110_2] cuando el resultado es mayor que 9, este procedimiento recibe el nombre de **Ajuste Decimal** [Decimal Adjust DA].

C_o	S_3	S_2	S_1	S_o	D. A.
O	O	O	O	O	O
O	O	O	O	1	O
O	O	O	1	O	O
O	O	O	1	1	O
O	O	1	O	O	O
O	O	1	O	1	O
O	O	1	1	O	O
O	O	1	1	1	O
O	1	O	O	O	O
O	1	O	O	1	O
O	1	O	1	O	1
O	1	O	1	1	1
O	1	1	O	O	1
O	1	1	O	1	1
O	1	1	1	O	1
O	1	1	1	1	1
1	O	O	O	O	1
1	O	O	O	1	1
1	O	O	1	O	1



La tabla anterior muestra todos los posibles resultados que se obtendrían al sumar dos números de 1-dígito BCD cada uno. Mediante el uso del Mapa-K se deduce la ecuación booleana de la función para el DA. Después de realizar las agrupaciones adecuadas, el ajuste decimal está dado por la

siguiente ecuación.

$DA = S_3S_2 + S_3S_1 + C_o$, o también

$DA = S_3(S_2 + S_1) + C_o$

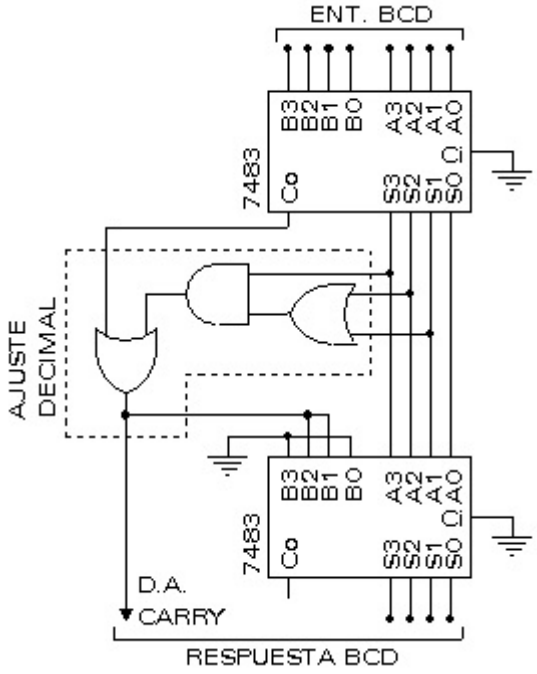


FIGURA 2.26 SUMADOR DE 2 NÚMEROS DE UN DÍGITO BCD

El circuito de la fig. 2.26 muestra un sumador de 1-dígito BCD, con Ajuste Decimal [Carry]. Para valores que requieren un mayor número de dígitos BCD, para cada dígito BCD se utiliza la misma circuitería del sumador de 1 dígito BCD, indicada

en la fig. 2.26, las veces que sea necesario. La salida D.A. [carry] es parte de la respuesta, no así la salida C_o del CI-7483 que realiza el ajuste decimal.

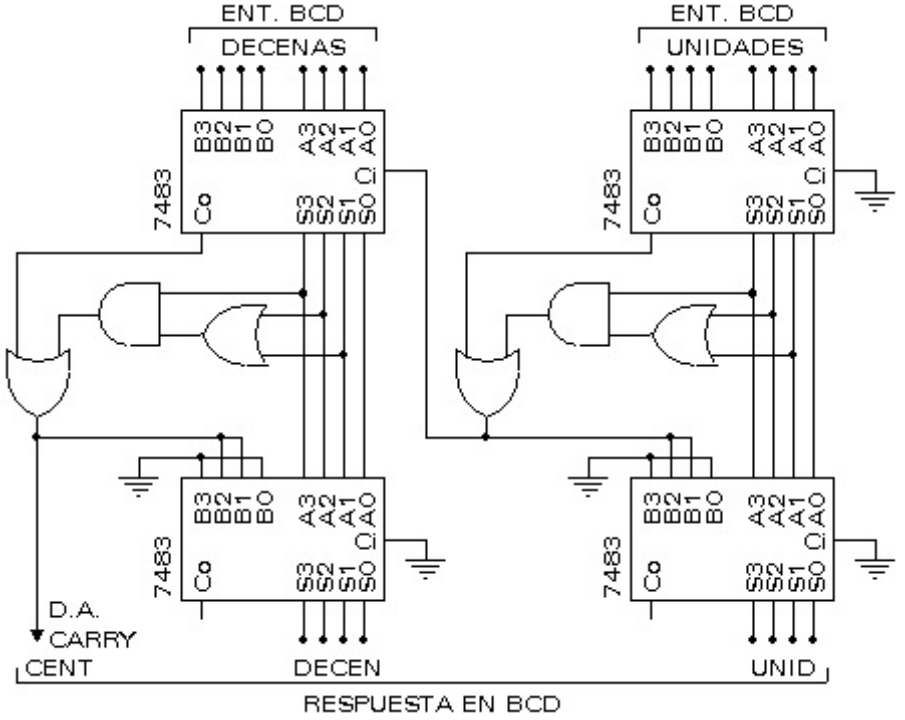


FIGURA 2.27 SUMA DE 2 NÚMEROS DE 2-DÍGITOS BCD

La fig. 2.27 muestra un sumador que acepta operandos de 2-dígitos BCD cada uno, y genera una respuesta también expresada en BCD. Este procedimiento puede expandirse para obtener sumadores BCD de mayor número de dígitos BCD.

Unidad Aritmética y Lógica [Arithmetic Logic Unit ALU].- Es un dispositivo que realiza operaciones aritméticas [suma, resta] y varias operaciones lógicas [AND, OR, etc.] con dos operandos [registros]: Reg-A y Reg-B de x-bits [donde x = 4, 8, 16, etc.] cada uno, para eso requiere de algunas entradas de selección [S] de la operación que se realizará. El resultado de la operación normalmente se almacena en el Registro A [Acumulador]. La ALU es una parte muy importante dentro de la CPU. La fig. 2.28 muestra el diagrama de bloques de una ALU típica de 8-bits.

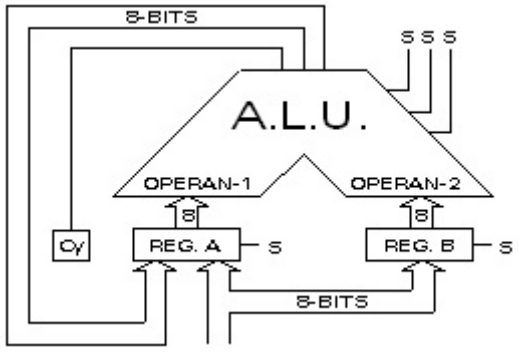


FIGURA 2.28 A. L. U.

CI-74LS181 [ALU/Generador de Función].- Las ALU MSI tienen operandos de 4-bits y de tres a cinco entradas de selección de operaciones, permitiendo la realización de hasta 32 funciones diferentes.

La fig. 2.29 muestra la distribución de pines de la ALU 74LS181 de 4-bits. El tipo de operación que realiza el CI-74181 se selecciona mediante la

entrada M y las entradas de selección [S_3, S_2, S_1, S_0] como se indica en la siguiente tabla. Obsérvese que los identificadores A, B y F en la tabla se refieren a palabras de 4-bits [A_3, A_2, A_1, A_0], [B_3, B_2, B_1, B_0] y [F_3, F_2, F_1, F_0] y los símbolos . y + se refieren a las operaciones AND y OR lógicas.

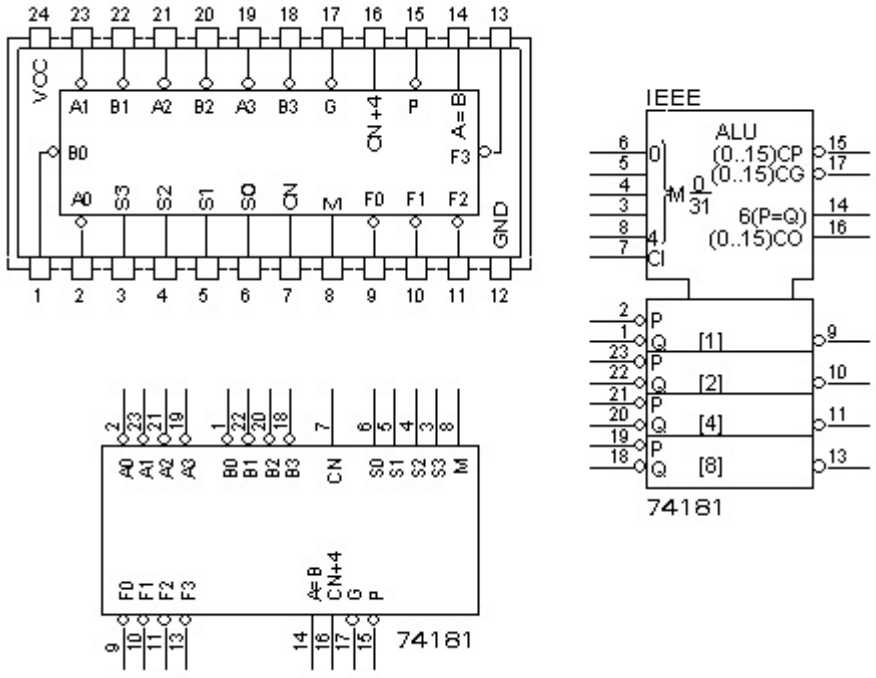


FIGURA 2.29 CI-74181 ALU/GENERADOR DE FUNCIONES

La entrada M del '181 selecciona entre las operaciones aritméticas y lógicas. Cuando M = 1, se seleccionan las operaciones lógicas y cada salida Fi es función sólo de las correspondientes entradas de datos, Ai y Bi [bit-a-bit]. No se propagan los

excesos [carry] entre las etapas y se ignora la entrada CN [carry in]. Las entradas [S_3, S_2, S_1, S_0] seleccionan una operación lógica particular; puede seleccionarse cualquiera de las 16 operaciones combinacionales lógicas diferentes sobre dos variables.

Tabla de Funciones realizadas por la ALU 74LS181 de 4-bits.

Entradas				Función	
S3	S2	S1	S0	M = O [aritmética]	M = 1 [lógica]
0	0	0	0	F = A menos 1 más Ci	F = A'
0	0	0	1	F = A.B menos 1 más Ci	F = A' + B'
0	0	1	0	F = A.B' menos 1 más Ci	F = A' + B
0	0	1	1	F = 1111 más Ci	F = 1111
0	1	0	0	F = A más (A + B') más Ci	F = A' . B'
0	1	0	1	F = A.B más (A + B') más Ci	F = B'
0	1	1	0	F = A menos B menos 1 más Ci	F = (A ⊕ B)'
0	1	1	1	F = A + B' más Ci	F = A + B'
1	0	0	0	F = A más (A + B) más Ci	F = A' . B
1	0	0	1	F = A más B más Ci	F = A ⊕ B
1	0	1	0	F = AB más (A + B) más Ci	F = B
1	0	1	1	F = A + B más Ci	F = A + B
1	1	0	0	F = A más A más Ci	F = 0000
1	1	0	1	F = A.B más A más Ci	F = A . B'
1	1	1	0	F = A.B' más A más Ci	F = A . B
1	1	1	1	F = A más Ci	F = A

Cuando M = O, se seleccionan las operaciones aritméticas, se propagan los excesos entre las etapas y se usa el Ci como una entrada de exceso a la etapa más significante. Para operaciones de más de 4-bits,

pueden conectarse varias ALU en cascada, con el exceso de salida [Co] de cada ALU conectado al exceso de entrada [Ci] de la etapa siguiente más significante. Las mismas señales de selección de función [M, S_3, S_2, S_1, S_0] se aplican a todos los '181 en la cascada.

Para realizar una suma en complemento a dos, se usa [S_3,S_2,S_1,S_0] para seleccionar la operación "A más B más Ci". La entrada Ci de la ALU menos significante está normalmente en 0 durante las operaciones de suma. Para realizar una resta en complemento a dos, se usa S_3 y S_0 para seleccionar la operación A menos B menos 1 más Ci. En este caso la entrada Ci de la ALU menos significante está en 1, ya que Ci actúa como el complemento del carry durante la resta.

El '181 proporciona otras operaciones aritméticas, como "A menos 1 más Ci", que son útiles en algunas aplicaciones [por ejemplo, decrementar en 1]. También proporciona muchas operaciones aritméticas raras, como "A.B' más (A + B') más Ci", que casi nunca se usan en la práctica.

Obsérvese que las entradas del operando [A_3,A_2,A_1,A_0] y [B_3,B_2,B_1,B_0] y las salidas de la función [F_3,F_2,F_1,F_0] del '181 son activas bajas. El '181 también puede usarse con las entradas de los operandos y las salidas de la función en activa alta. En este caso, debe construirse una versión diferente de la tabla de la función. Cuando M = 1,

las operaciones lógicas aún se realizan, pero para una combinación de entrada dada en $[S_3,S_2,S_1,S_0]$, la función obtenida es precisamente la dual de la listada en la tabla anterior. Cuando M = 0, se realizan las operaciones aritméticas, pero la tabla de función es de nuevo diferente. Refiérase a la hoja de datos del '181 para más detalles.

CI-74LS381 y CI-74LS382 [ALU/Generadores de Función].- Otra ALU MSI, el CI-74LS381 [y el CI-74LS382] que se muestra en la fig. 2.30, codifica sus entradas de selección más compactamente y proporciona sólo ocho diferentes funciones útiles, como se detalla en la tabla siguiente.

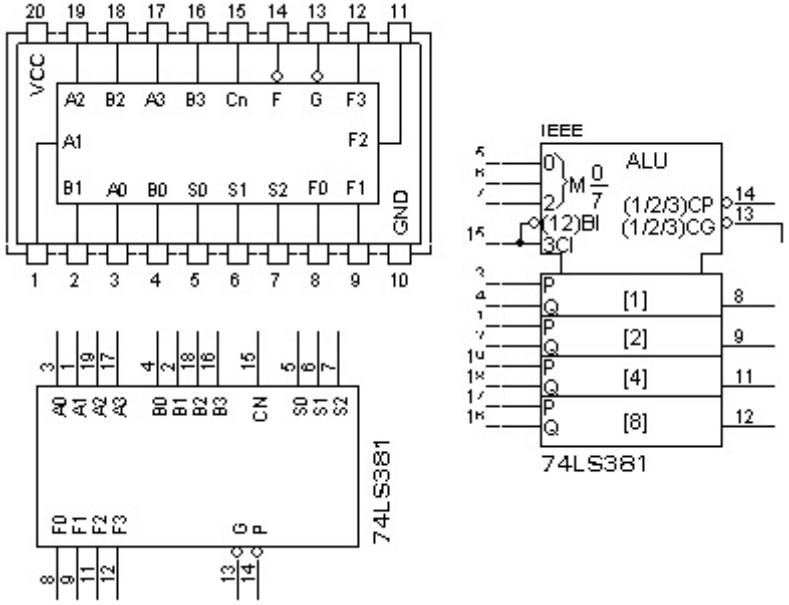


FIGURA 2.30 CI-74LS381 ALU/GENERADOR DE FUNCIONES

La única diferencia entre el '381 y el '382 es que uno proporciona salidas de exceso anticipado en grupo mientras que el otro proporciona salidas de exceso y desborde propagado.

Entradas			Función
S2	S1	S0	
0	0	0	F = 0000
0	0	1	F = B menos A menos 1 más Ci
0	1	0	F = A menos B menos 1 más Ci
0	1	1	F = A más B más Ci
1	0	0	F = A⊕B
1	0	1	F = A + B
1	1	0	F = A.B
1	1	1	F = 1111

Multiplexor o Selector de Datos.- Es un dispositivo que acepta varias entradas de datos, pero solo deja pasar una de ellas a la salida. Cuando se ha seleccionado una señal, las otras no tienen efecto sobre la salida. Los multiplexers pueden ser analógicos o digitales. El MUX-analógico [fig. 2.31], generalmente es mecánico y la selección de los canales se la hace manualmente [también puede ser electromecánico].

En el MUX-digital, la selección de los canales se la hace mediante entradas de comando digital, a veces conocidas como entradas de dirección [fig. 2.32].

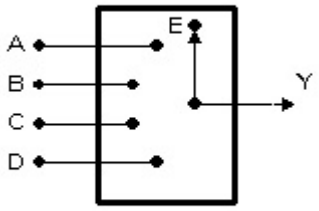


FIGURA 2.31

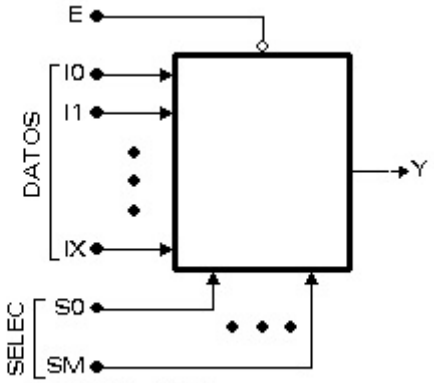


FIGURA 2.32

A continuación se muestra un ejemplo de un MUX-digital de 2-entradas y 1-salida (MUX 2-1). La fig. 2.33 muestra el diagrama de bloques del MUX 2-1, que se va a diseñar. Para esto se incluye la tabla de función requerida.

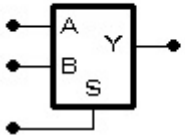


FIGURA 2.33
MUX 2-1 SIN
ENABLE

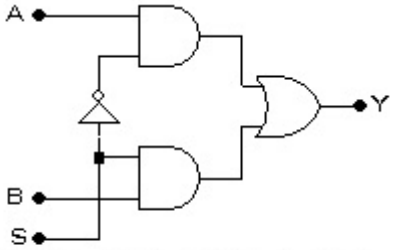


FIGURA 2.34 MUX 2-1 CON
COMPUERTAS A - O - N SIN
ENABLE

S	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0

S	Y
0	A

0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

1	B

La ecuación para Y, en cualquiera de las dos formas de presentar el comportamiento del MUX, es

$$Y = \bar{S}A + SB$$

En la tabla se puede observar que cuando $S = 0_L$, $Y = A$ y cuando $S = 1_L$, entonces $Y = B$. La fig. 2.34 muestra la implementación del MUX 2-1 utilizando compuertas A-O-N.

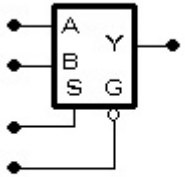


FIGURA 2.35
MUX 2-1 CON
ENABLE

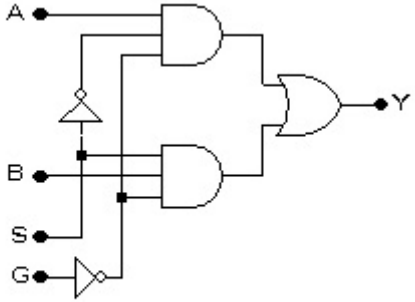


FIGURA 2.36 MUX 2-1 CON
ENABLE

En muchas ocasiones es necesario incluir una entrada para habilitación del circuito integrado [CI], conocida como Chip-Enable [CE = Habilitación

del Circuito]. Existen dos formas de hacer esto, una de ellas utilizando compuertas AND de 3-entradas, en las que la tercera entrada sirve para habilitar o deshabilitar al multiplexer [figs. 2.35 y 2.36].

La entrada enable puede resultar útil para comandar la función del MUX. En este caso, se ha incluido una entrada adicional a las compuertas AND. En otras ocasiones puede ser más conveniente usar una compuerta de salida [OR] de 3-estados, con lo que se consigue que cuando el MUX esté deshabilitado del sistema, prácticamente está desconectado del mismo. En el mercado existen MUX de 2-1, 4-1, 8-1, etc. que tienen entradas de selección, la salida y una entrada de habilitación. Entre otros: el CI-74157, que tiene 4-MUX de 2-1, con entrada de selección común para los 4-MUX. El CI-74151, tiene un MUX de 8-1, con 3-entradas de selección.

CI-74151.- Data Selector/Multiplexer 8-1. Contiene en el chip toda la decodificación binaria para seleccionar la fuente de datos deseada. El CI-74151 selecciona 1-de-8 fuentes de datos [fig. 2.37]. Tiene una entrada strobe [habilitación] que debe estar en un nivel lógico bajo para habilitar a este dispositivo. Un nivel alto en la entrada strobe hace que la salida Y esté en un nivel alto y la salida $\overline{Y}$ [o W cuando se la use] a un nivel bajo. A continuación se muestra la tabla de función del CI-74151 [MUX de 8-a-1].

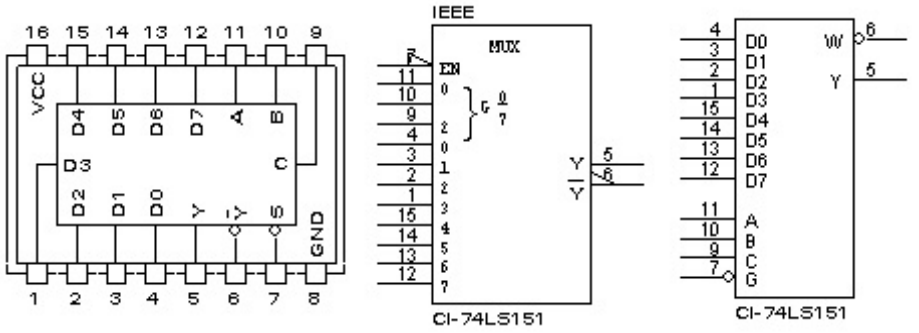


FIGURA 2.37 CI-74LS151 DATA SELECTOR/MULTIPLEXER 8-a-1

ENTRADAS				SALIDAS	
SELECCIÓN			STROBE	Y	$W = \overline{Y}$
C	B	A	S		
X	X	X	1	0	1
0	0	0	0	D0	$\overline{D0}$
0	0	1	0	D1	$\overline{D1}$
0	1	0	0	D2	$\overline{D2}$
0	1	1	0	D3	$\overline{D3}$
1	0	0	0	D4	$\overline{D4}$
1	0	1	0	D5	$\overline{D5}$
1	1	0	0	D6	$\overline{D6}$
1	1	1	0	D7	$\overline{D7}$

CI-74157.- Cuatro selectores de datos/multiplexers de 2-líneas-a-1-línea comandados por una entrada de selección [S] común. Estos selectores de datos/multiplexers tienen en el integrado inversores y drivers para proporcionar la selección completa de los datos a las cuatro compuertas de salida.

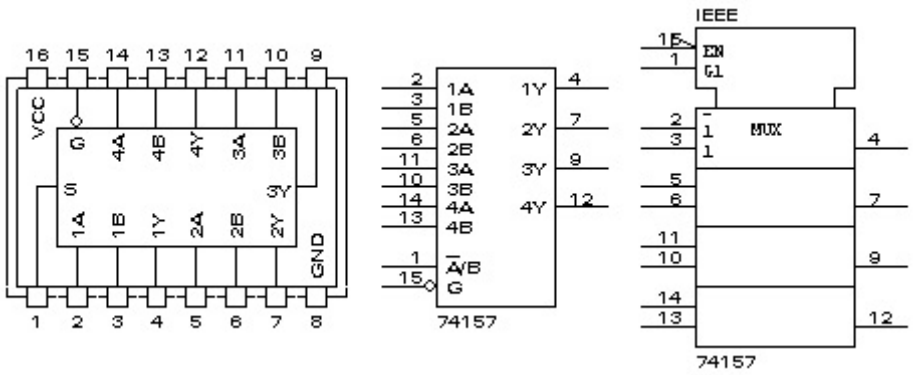


FIGURA 2.38 CI-74LS157 CUATRO DATA SELECTOR/MULTIPLEXER 2-a-1

Dispone de una entrada [strobe] para habilitación del chip, cuando G = 0 se habilita el chip y cuando G = 1 se deshabilita y sus salidas permanecen en nivel bajo [0_L]. Una palabra de 4-bits se selecciona de entre dos fuentes [A y B] y se la enruta a las 4-salidas [Y]. El CI-74157 [fig. 2.38] presenta los datos reales, mientras que el CI-74158 presenta los datos con inversión para minimizar los tiempos de propagación. A continuación se muestra la tabla de función proporcionada por el fabricante para los CI-74157 y 74158 respectivamente.

ENTRADAS				SALIDA Y	
STROBE	SELECT	A	B	74157	74158
1	X	X	X	0	1
0	0	0	X	0	1
0	0	1	X	1	0
0	1	X	0	0	1
0	1	X	1	1	0

Aplicaciones de los Multiplexers.- En los últimos tiempos se ha incrementado el uso de los multiplexers en el diseño de dispositivos digitales. En este caso se estudian dos aplicaciones: a) Incremento del número de entradas; b) Implementación de una función booleana combinacional.

Ejemplo 1).- Se dispone de MUX 2-1, implementar un MUX 8-1. La solución se muestra en la fig. 2.39.

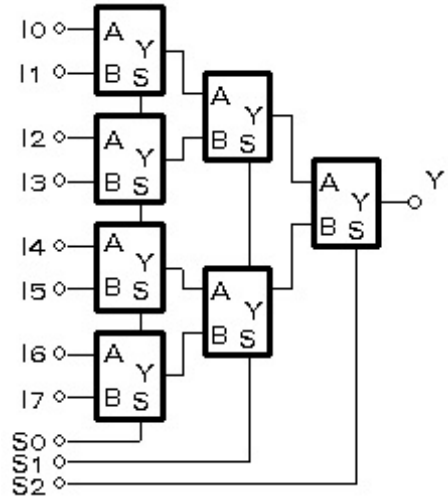


FIGURA 2.39

Ejemplo 2).- Implementar la siguiente función booleana.

$$F<D, C, B, A> = \sum (0,2,5,6,8,9,12,14,15)$$

Para solucionar este tipo de problemas, primero

conviene escribir la tabla d verdad. Luego se determina el tipo de MUX que se utilizará, eso depende del número de variables de entrada que tenga la función que se va a implementar. Si la función tiene n-variables de entrada, entonces se requiere de un MUX de [n-1] entradas de selección, la cuarta variable de entrada de la función se la utiliza para conectarla, de manera apropiada, en las entradas de datos del MUX [esta variable puede ser cualquiera de las n-variables, pero es recomendable utilizar la más significante, que es lo que se hará en este caso].

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

En este caso la función tiene 4-variables de entrada, por tanto el MUX debe tener 3-entradas de selección ([S_2, S_1, S_0] para las variables menos significantes de la función), por tanto será un MUX de 8-a-1. Para saber qué se conectará en cada una de las 8-entradas del MUX se utiliza una tabla que tiene 2-filas, en ella se muestran las entradas del MUX [I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7] y los valores de la función F<D,C,B,A> como se indica a continuación.

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7		
1	0	1	0	0	1	1		$\bar{D}$	[D = 0]
1	1	0	0	1	0	1	1	D	[D =1]
1	D	$\bar{D}$	0	D	$\bar{D}$	1	D		

La fila superior corresponde a la variable complementada [$\bar{D}$] porque en las 8 primeras combinaciones de 0s y 1s, la variable D vale 0, mientras que la fila inferior corresponde a D porque las 8 combinaciones finales D vale 1. Cuando en una columna hay dos-1s debajo se pone 1, como en las columnas I_0 e I_6; cuando hay dos-0s debajo se pone 0, como en la columna I_3; cuando en el casillero superior hay un 0 y en el inferior hay un 1, debajo se pone la variable más significante D [columnas I_1 e I_4]; cuando en el casillero superior hay un 1 y en el inferior hay un 0, debajo se pone la variable más significante complementada [$\bar{D}$], columnas I_2 e I_5. El circuito resultante se muestra en la fig. 2.40.

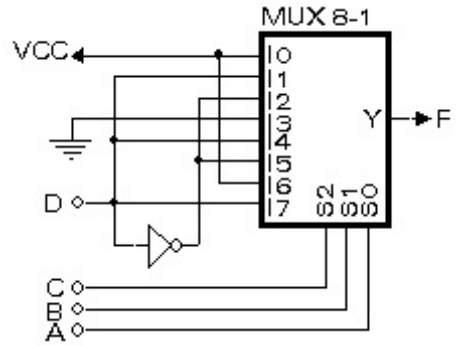


FIGURA 2.40

Ejemplo 3).- Utilizando un MUX apropiado implementar la siguiente función booleana.

$$F<E, D, C, B, A> = \sum m(1,3,4,7,9,10,16,19,22,24,25,26,29,30,31)$$

Se requiere de un MUX 16-1, en las entradas de datos I se ubicará la variable de entrada E, en su forma normal o en su forma complementada de la manera que se deduce de la siguiente tabla.

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

El criterio para la conexión de las entradas al MUX es el mismo que el ejemplo anterior. El circuito resultante se muestra en la fig. 2.41.

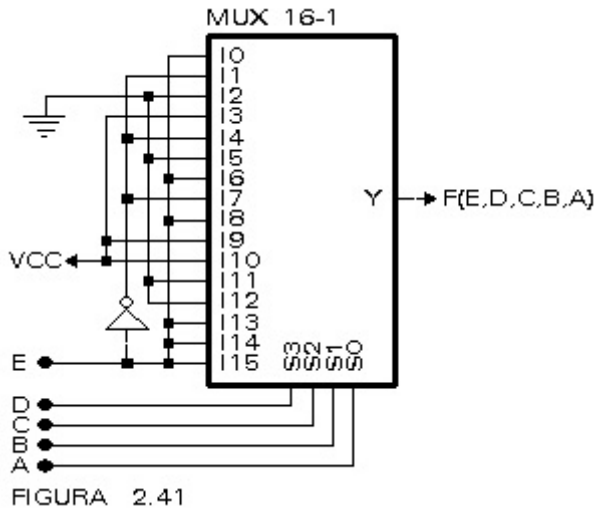


FIGURA 2.41

Demultiplexer/Decoder [Distribuidor de Datos].- El demultiplexer realiza la función inversa al multiplexer, es decir, tiene una señal de entrada de datos I, y varias salidas hacia una de las cuales se enviará la información de la entrada [en base a un código de selección], por tanto, a este circuito también se lo conoce como **"enrutador".**

Ejemplo.- Diseñar un DEMUX de 1-a-4 [una línea de entrada: I, y 4 líneas de salida: Y_0, Y_1, Y_2 y Y_3].

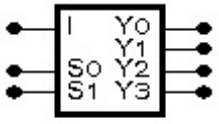


FIGURA 2.42

Para llevar a cabo este proceso se requieren 2

líneas adicionales de selección [S_1 y S_0], fig. 2.42.

A continuación se muestra la tabla de función del DEMUX 1-a-4, juntamente con las ecuaciones booleanas para las funciones de salida. La tabla adjunta es una forma resumida de la primera, pero en ella está toda la información requerida para el diseño, por tanto, de ella se pueden deducir las ecuaciones de las variables de salida.

I	S0	S1	Y0	Y1	Y2	Y3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

S1	S0	Y0	Y1	Y2	Y3
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

$$Y_0 = \bar{S}_1\bar{S}_0 I$$
$$Y_1 = \bar{S}_1 S_0 I$$
$$Y_2 = S_1\bar{S}_0 I$$
$$Y_3 = S_1 S_0 I$$

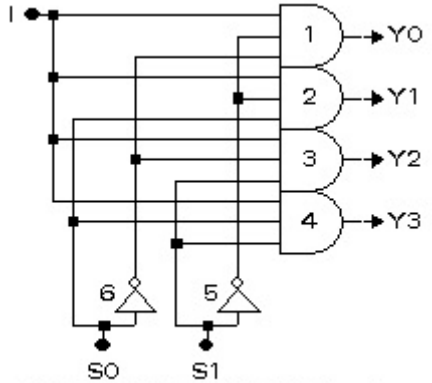


FIGURA 2.43 DEMUX 1-a-4

El circuito lógico combinacional, con compuertas A-O-N, se muestra en la fig. 2.43.

Decodificador [Decoder].- Diseñar un decoder de 2-a-4 [dos líneas de entrada: I_1, I_0, y 4 líneas de salida: Y_0, Y_1, Y_2 y Y_3]. En este ejemplo se utilizará una línea adicional de habilitación [G], fig. 2.44.

También se muestra la tabla de función del decoder 2-a-4, juntamente con las ecuaciones booleanas para las funciones de salida y las ecuaciones de las variables de salida.

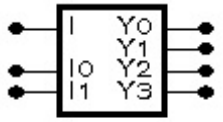


FIGURA 2.44

G	I1	I0	Y0	Y1	Y2	Y3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$$Y_0 = \bar{I}_1\bar{I}_0 G$$
$$Y_1 = \bar{I}_1 I_0 G$$
$$Y_2 = I_1\bar{I}_0 G$$
$$Y_3 = I_1 I_0 G$$

El circuito lógico combinacional, con compuertas A-O-N, se muestra en la fig. 2.45.

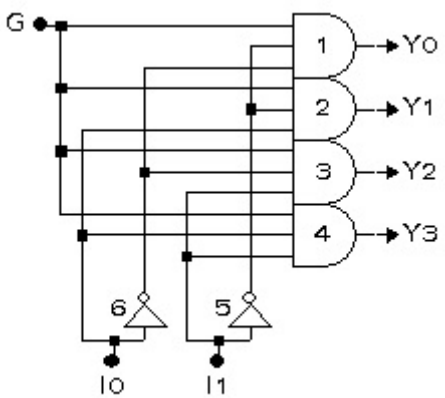


FIGURA 2.45 DECODER 2-a-4

Al comparar el diagrama de bloques y el circuito A-O-N del DEMUX con los del decodificador que se acaban de diseñar, se deduce que estructuralmente son iguales, por eso se los conoce como Decoder/DEMUX, debido a que cumplen exactamente la misma función. En los manuales de los fabricantes constituyen un mismo circuito, solamente que se debe tener cuidado de cómo utilizar las entradas.

CI-74138.- Decoder/Demultiplexer 3-8. Diseñado para aplicaciones de alta velocidad como: decodificadores de dirección de memorias o en aplicaciones que requieren enrutamiento de datos. El CI-74138 decodifica 1 de 8 líneas dependiendo de las condiciones de 3 entradas de selección [C, B, A] y de 3 entradas de habilitación [G_1, G_{2A}, G_{2B}], 2 de ellas con nivel activo bajo [$G_2 = G_{2A}G_{2B}$ en la tabla de función del CI-74138] y una con nivel activo alto

[G_1], esto reduce la necesidad de compuertas o inversores externos cuando se desea expandir. La fig. 2.46 muestra la distribución de pines y el símbolo esquemático del Demux/Decoder 3-8. La tabla de función se muestra a continuación.

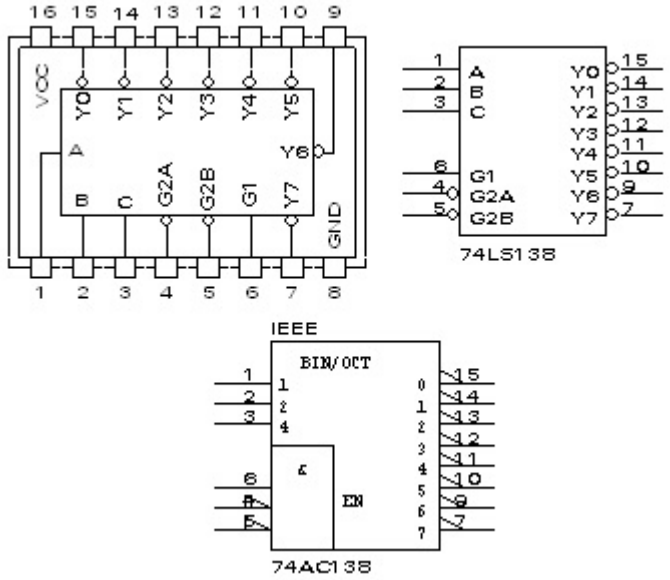


FIGURA 2.46 CI-74LS138 DECODER/DEMULTIPLEXER 3-8

ENTRADAS					SALIDAS							
HABILI-TACIÓN		SELEC-CIÓN										
G1	G2	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
X	1	X	X	X	1	1	1	1	1	1	1	1
0	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1

1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

CI-74139.- Decoder/Demultiplexer 2-4 [fig. 2.44]. Dispone de 2 decodificadores individuales de 2-lineas a 4-líneas, en un solo paquete. La entrada de habilitación de nivel activo bajo se la puede utilizar como una línea de datos en aplicaciones de multiplexado. La fig. 2.47 muestra el símbolo del Demux/Decoder 2-4 adjunto a la tabla de función que da el fabricante.

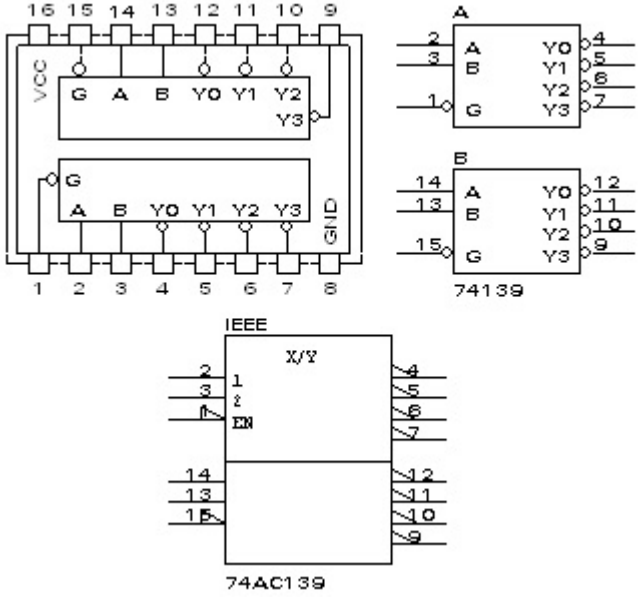


FIGURA 2.47 CI-74LS139 2 DECODER/DEMUX 2-4

ENTRADAS			SALIDAS			
HABILITACIÓN	SELECCIÓN					
G	B	A	Y_0	Y_1	Y_2	Y_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Aplicaciones de los Decodificadores.- Puesto que los decodificadores generan todos los minterms, por ejemplo Y_0 genera el minterm m_0, etc. [fig. 2.48], se los puede utilizar para generar funciones booleanas expresadas en la forma canónica disyuntiva.

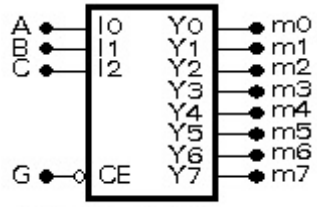


FIGURA 2.48

El número de variables de la función booleana indica el tipo de decodificador que se requiere; por ejemplo, para implementar una función de 3-variables de entrada, se requiere de un decodificador de 3-8, como en el siguiente ejemplo.

$$F<C, B, A> = \sum (1,2,4,6)$$

entonces el decodificador de 3-a-8, es decir 3-líneas de entrada y 8-líneas de salida, cada salida genera un término mínimo. Para esta función se requieren las salidas $Y_1 = m_1$; $Y_2 = m_2$; $Y_4 = m_4$ y $Y_6 = m_6$. de modo que

$$F<C,B,A> = Y_1 + Y_2 + Y_4 + Y_6$$

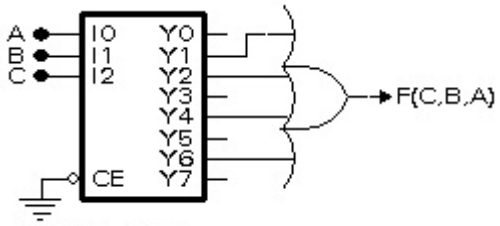


FIGURA 2.49

La fig. 2.49 muestra la implementación de la función booleana pedida, para esto se utilizó un decodificador de 3-a-8.

Ejemplo.- Utilizar un decodificador del número de entradas adecuado, para implementarun circuit o que genere las siguientes funciones lógicas.

$$F_1 <D, C, B, A> = \sum m(0,3,6,7,9,12,15)$$
$$F_2 <D, C, B, A> = \prod M(0,2,4,5,7,8,9,14)$$

El circuito de la fig. 2.50 muestra el generador de funciones booleanas pedido, se utiliza un decodificador de 4-8 con salidas de nivel activo alto, una compuerta OR para la función F1 y otra

para F2. Se ha incluido la tabla de verdad para facilitar la implementación.

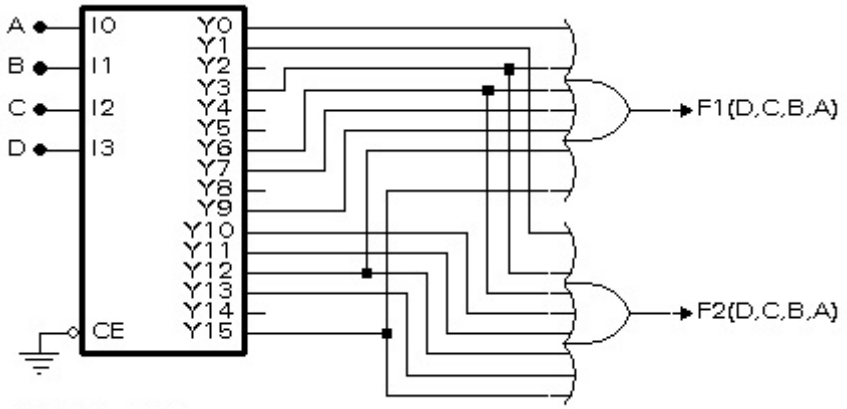


FIGURA 2.50

ENTRADAS				SALIDAS	
D	C	B	A	F1	F2
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	1	0	0	0
1	1	1	1	1	1

Decodificador [Decoder].- Es un circuito combinacional que convierte información binaria de n-bits de entrada a un máximo de 2^n líneas de salida única.

Codificador [Encoder].- Es una función digital que produce una operación inversa a la de un decodificador. Un codificador tiene 2^n líneas de entrada y n líneas de salida.

Codificador de Prioridad.- El problema de los codificadores estudiados, es que se debe activar [habilitar] una sola entrada a la vez, porque de lo contrario, en las salidas , se producirán errores. Para evitar este problema sa han diseñado los codificadores con prioridad, en los cuales, cuando se habilita más de una entrada, en la salida se obtiene el código de la entrada de mayor valor [prioridad]. Los CI-74147 y 74148 son codificadores con prioridad BCD y octal respectivamente.

CI-74147.- Es un codificador de prioridad de 10-líneas decimales a 4-líneas BCD [fig. 2.51]. Sus aplicaciones son: codificadores de teclado y selección de rango.

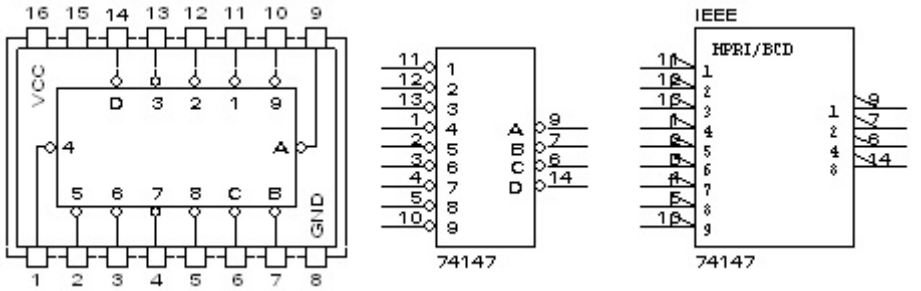


FIGURA 2.51 CI-74147 CODIFICADOR CON PRIORIDAD DE 10-LÍNEAS DECIMALES A 4-LÍNEAS BCD

A continuación se indica la tabla de función dada por el fabricante. Se observa que las entradas se habilitan con nivel activo bajo [0] y que las salidas generan el complemento del valor BCD correspondiente.

ENTRADAS									SALIDAS			
1	2	3	4	5	6	7	8	9	D	C	B	A
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

CI-74148.- Es un codificador de prioridad de 8-líneas de datos a 3-líneas binarias u octal [fig. 2.52]. Para conexiones en cascada se han proporcionado las entradas de habilitación EI y de salida EO para la

expansión octal sin necesidad de circuitería externa. Aplicaciones: codificadores de N-bits, convertidores y generadores de código. A continuación se muestra la tabla de función, en la que "X" significa "no importa" o "irrelevante" y GS es la bandera de prioridad.

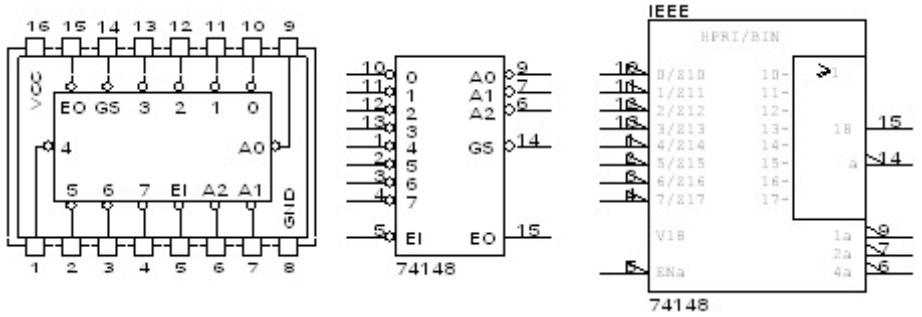


FIGURA 2.52 CI-74148 CODIFICADOR CON PRIORIDAD DE 8-LÍNEAS DE DATOS A 3-LÍNEAS BINARIAS

ENTRADAS									SALIDAS				
EI	O	1	2	3	4	5	6	7	A_2	A_1	A_2	GS	EO
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

CI-74180 [Chequeador/Generador de Paridad].- Es un chequeador/generador de paridad par/impar de 9-bits

[8-bits de datos más 1-bit de paridad] [fig. 2.53]. Las salidas odd/even [impar/par] y las entradas de control facilitan la operación en cualquier aplicación par o impar.

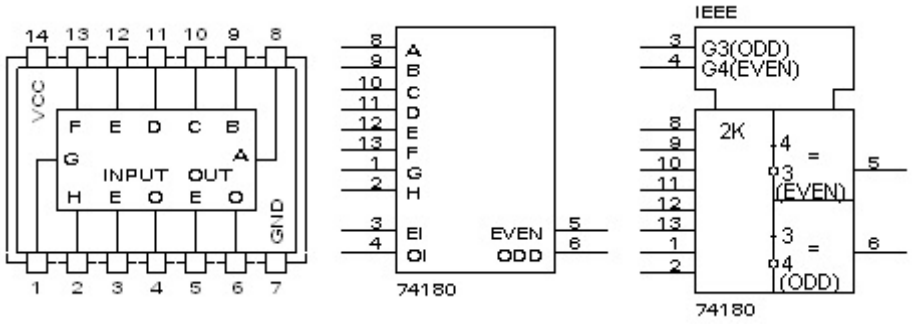


FIGURA 2.53 CI-74180 GENERADOR/CHEQUEADOR DE PARIDAD/IMPARIDAD DE 9-Bits

ENTRADAS			SALIDAS	
Σ de 1s en A hasta H	PAR	IMPAR	Σ PAR	Σ IMPAR
PAR	1	0	1	0
IMPAR	1	0	0	1
PAR	0	1	0	1
IMPAR	0	1	1	0
X	1	1	0	0
X	0	0	1	1

Dependiendo de si está generando o chequeando paridad o imparidad, las entradas par o impar pueden usarse como la entrada de paridad o el noveno bit. La capacidad de la longitud de una palabra puede expandirse fácilmente conectándolos en cascada. [Este tipo de red combinacional se estudió en el capítulo

1]. En la fig. 2.53, E = Even = Par; O = Odd = Impar; en la tabla, EI = Even Input, OI = Odd Input.

Comparador de Magnitud.- En muchas aplicaciones de sistemas digitales, es necesario saber la relación que existe entre 2 cantidades y en base a ello tomar una decisión, en el caso de los lenguajes de programación se hace mediante la proposición IF, o alguna instrucción similar. La fig. 2.54 muestra el diagrama de flujo de la instrucción IF A > B THEN. Si no se cumple la condición, el programa salta a otra parte del programa; si se cumple continúa con la siguiente instrucción.

Otro ejemplo se da en los microcontroladores de la serie 8051/52, con la instrucción *CJNE A,#datos,rel*, que *Compara-Salta-si-No-es-Igual* (*A ≠ datos*) una cantidad de pasos igual a *rel*. Ver fig. 2.55.

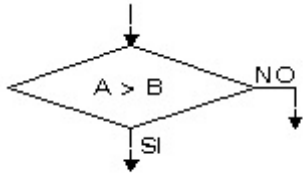


FIGURA 2.54

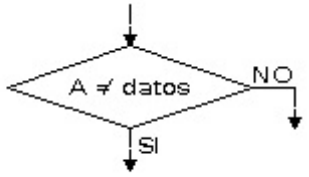


FIGURA 2.55

La comparación de la magnitud de dos cantidades puede realizarse con circuitos lógicos que permitan comparar dos números binarios (A y B) y cuya salida indique cuando A > B, A = B o A < B. Como ejemplo

se hace el diseño de un comparador de magnitud para números de 2-bits cada uno [A = A_1A_0 y B = B_1B_0]. Las salidas [A > B; A = B; A < B] toman el valor 1_L cuando se cumple la condición respectiva y 0_L en cualquier otro caso.

En la siguiente tabla de verdad se muestra la información de este comparador de magnitud. Abajo se presentan las ecuaciones booleanas en su forma canónica disyuntiva.

ENTRADAS				SALIDAS		
B1	B2	A1	A0	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1		0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

$$\begin{cases} A > B = \sum (1,2,3,6,7,11) \\ A = B = \sum (0,5,10,15) \\ A < B = \sum (4,8,9,12,13,14) \end{cases}$$

CI-7485 Comparador de Magnitud.- [Para números de 4-bits] Este comparador de magnitud es un circuito lógico combinacional que permite comparar la magnitud de dos cantidades binarias y genera tres salidas: una para indicar si A es mayor que B [A > B], otra para indicar si A es igual a B [A = B] y una tercera para indicar si A es menor que B [A < B].

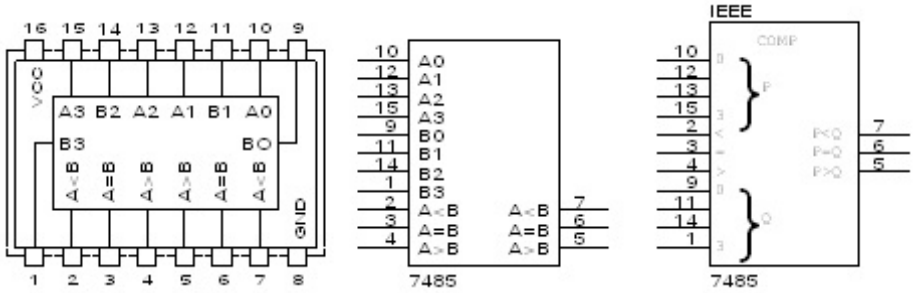


FIGURA 2.56 CI-7485 COMPARADOR DE MAGNITUD DE 4-Bits

La fig. 2.56 muestra la distribución de pines del CI-7485, a continuación se muestra la tabla de función proporcionada por el fabricante.

ENTRADAS DE COMPARACIÓN				ENTRADAS EN CASCADA			SALIDAS		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A = B	A < B	A > B	A = B	A < B
A3 > B3	X	X	X	X	X	X	1	O	O
A2 < B2	X	X	X	X	X	X	O	O	1
A3 = B3	A2 > B2	X	X	X	X	X	1	O	O
A3 = B3	A2 < B2	X	X	X	X	X	O	O	1
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	1	O	O
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	O	O	1
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	1	O	O
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	O	O	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	O	O	1	O	O
A3 = B3	A2 = B2	A1 = B1	A0 = B0	O	1	O	O	1	O
A3 = B3	A2 = B2	A1 = B1	A0 = B0	O	O	1	O	O	1

'85, 'LS85, 'S85

A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	1	O	O	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	1	O	O	O	O
A3 = B3	A2 = B2	A1 = B1	A0 = B0	O	O	O	1	1	O

'L85

A3 = B3	A2 = B2	A1 = B1	A0 = B0	O	1	1	O	1	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	O	1	1	O	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	1	1	1	1	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	1	O	1	1	O
A3 = B3	A2 = B2	A1 = B1	A0 = B0	O	O	O	O	O	O

Diseño con Circuitos MSI. Muchas aplicaciones requieren circuitos lógicos MSI, en ellos puede verse que el diseño tradicional en base a tablas de verdad y métodos de simplificación ya no puede aplicarse. Lo importante, en este caso, es conocer las diferentes funciones lógicas y los tipos de CI disponibles en el mercado y utilizarlos con ingenio

y creatividad. Esto se podrá observar en los siguientes casos que se presentan para que el alumno los estudie a profundidad y saque sus propias conclusiones.

Ejemplo 1.- En base al CI-74139 [decodificador 2-a-4-líneas] construir un decodificador 4-a-16-líneas.

Una posible solución se muestra en la fig. 2.57.

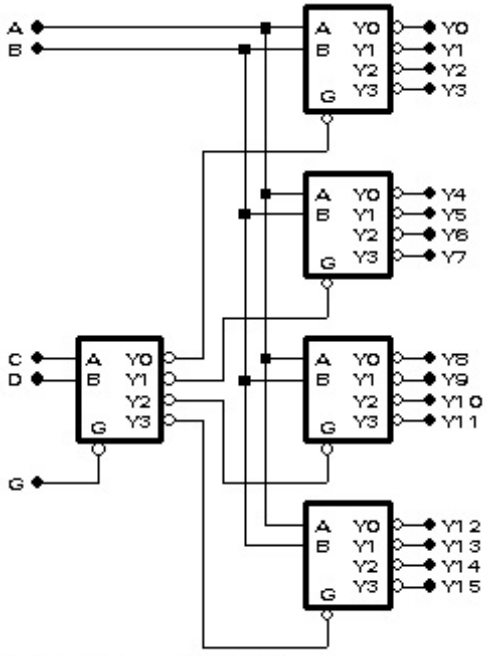


FIGURA 2.57 EJEMPLO 1

Ejemplo 2.- Utilizar un MUX y un DEMUX para hacer un control remoto que detecte la apertura de 8-puertas de una vivienda. Mientras las puertas estén

cerradas los LEDs estarán apagados; cuando se abra una de las puertas, el LED correspondiente a esa puerta debe encenderse.

La solución se muestra en la fig. 2.58. El contador sirve para monitorizar en forma continua la posible apertura de una o más puertas.

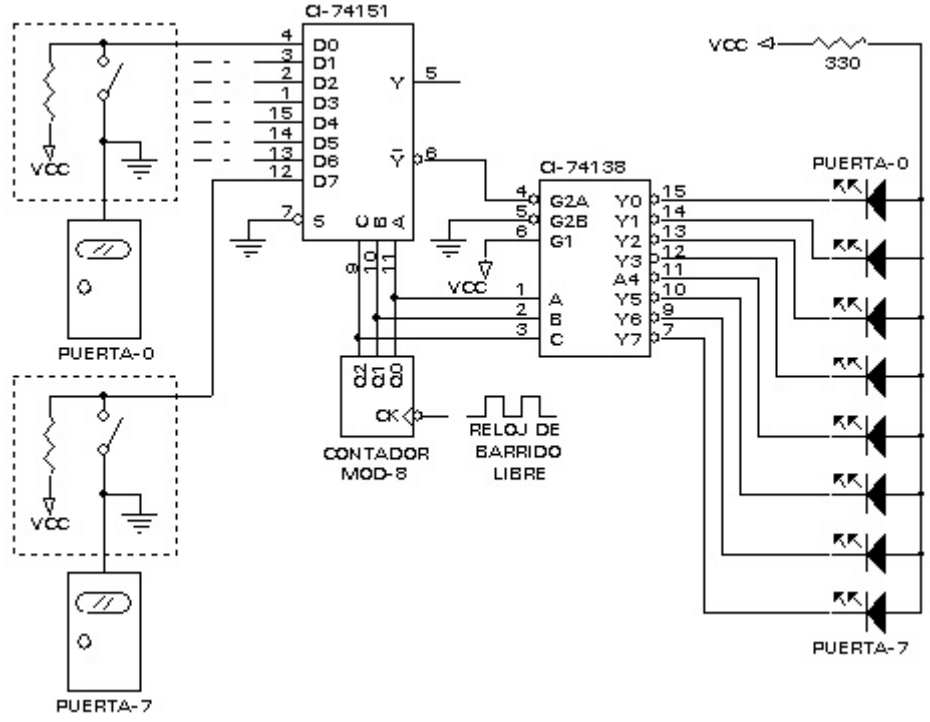


FIGURA 2.58 EJEMPLO 2

Ejemplo 3: Incremento del número de bits del un comparador de magnitud.- La fig. 2.59 muestra una forma de obtener un comparador de magnitud para números de 8-bits mediante el uso del CI-7485. Para

mayor número de bits, el fabricante da, en el manual, otra alternativa para optimizar el tiempo de respuesta.

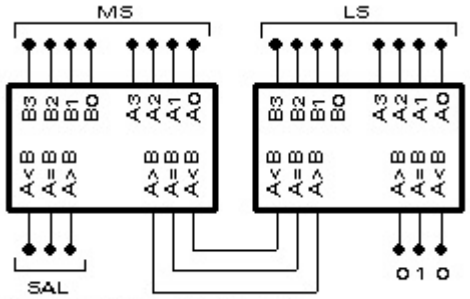


FIGURA 2.59 EJEMPLO 3

Ejemplo 4.- El circuito de la fig. 2.60 acepta en sus entradas dos números de un dígito-BCD cada uno $[A_3A_2A_1A_0]$ y $[B_3B_2B_1B_0]$. En el display de cátodo común se muestra el mayor de ellos. Analizar el comportamiento del circuito.

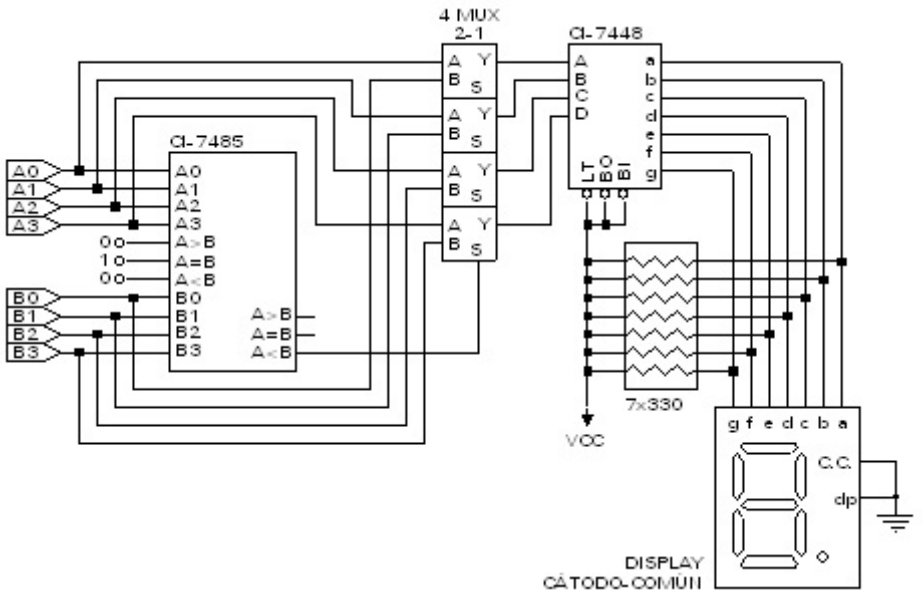


FIGURA 2.60 EJEMPLO 4

Ejemplo 5.- Mediante el uso de CIs MSI, diseñar un circuito lógico que disponga de 2-entradas [A y B] cada una de las cuales recibe un número de 1-dígito codificado en BCD y una entrada de selección [S], de manera que cuando S = 0 en la salida [Y] de 4-bits aparezca el menor entre A y B y que cuando S = 1, la salida muestre el número mayor entre A y B; en caso de que A = B, la salida se debe blanquear. El resultado debe aparecer en un display de 7-segmentos de ánodo común.

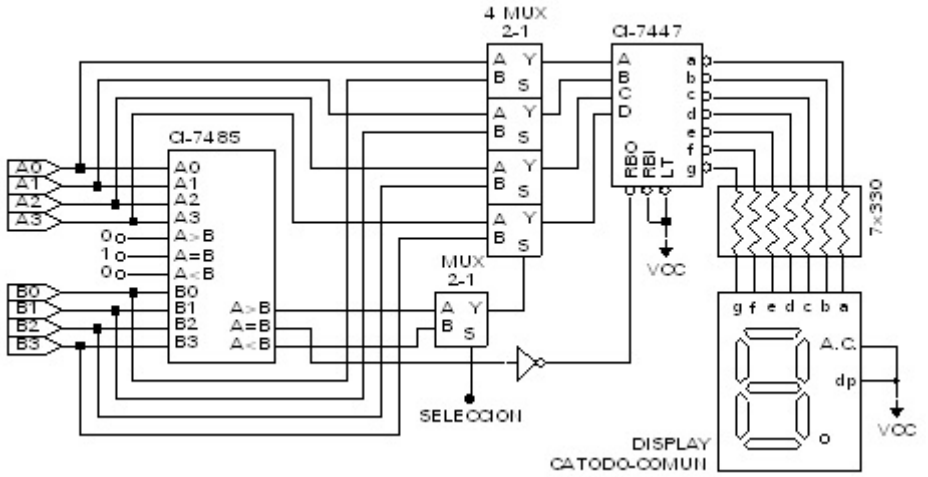


FIGURA 2.61 EJEMPLO 5

Ejemplo 6.- Diseñar un circuito lógico que disponga de 4-entradas [A, B, C y D] cada una de las cuales recibe un número de 1-dígito codificado en BCD y una salida [Y] de 4-bits. En la salida debe aparecer el número de mayor magnitud de los 4 de las entradas. El resultado debe aparecer en un display TIL 311.

La fig. 2.62 muestra el circuito pedido, a la izquierda se muestran dos comparadores de magnitud para comparar entre A y B, el primero y entre C y D en segundo, a continuación se usan multiplexers para escoger entre los números señalados por los comparadores; el siguiente comparador y el multiplexer de la derecha determinan cual es el número mayor de entre los 4 de las entradas.

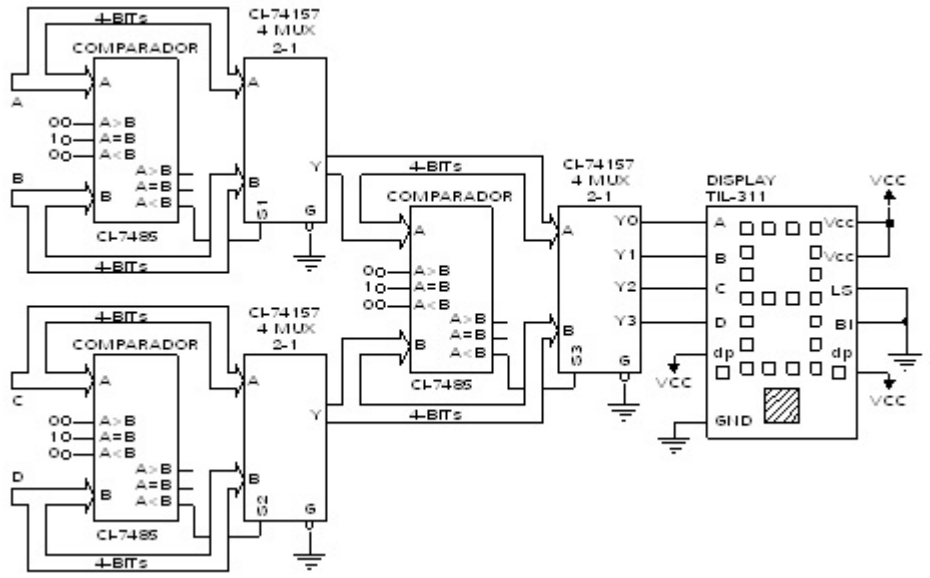


FIGURA 2.62 EJEMPLO 6

Ejemplo 7.- Diseñar un circuito lógico que disponga de 2-entradas [A y B] cada una de las cuales recibe un número de 1-dígito codificado en BCD y una entrada de selección [S], de manera que cuando S = 0 en la salida [Y] de 4-bits aparezca el menor entre A y B y que cuando S = 1, la salida muestre la suma entre A y B [utilice un LED para mostrar las decenas (Carry)]. El resultado debe aparecer en un display TIL311.

S	FUNCIÓN
0	SALE EL MENOR
1	SALE LA SUMA

La solución se muestra en la fig. 2.63.

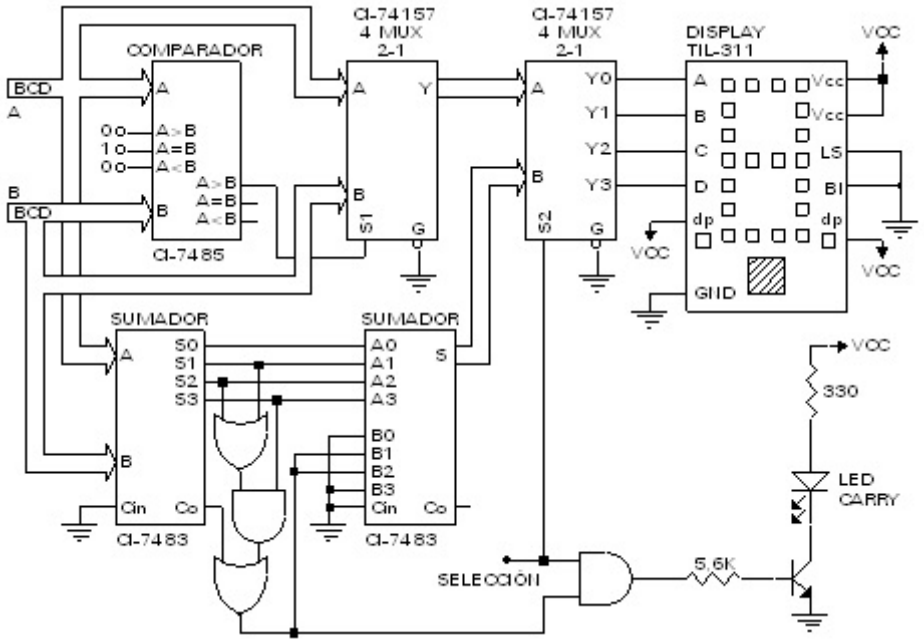


FIGURA 2.63 EJEMPLO 6

Ejemplo 8.- Utilizar dispositivos MSI, y las compuertas adicionales que requiera, para diseñar un circuito combinacional cuyas entradas son dos números binarios de 4-bits cada uno A [$A_3A_2A_1A_0$] y B [$B_3B_2B_1B_0$] y dos líneas de selección de función [S_1 y S_0] que cumpla con la tabla que se indica a continuación. La salida consistirá de un display hexadecimal con decodificador incluido [TIL 311], un LED rojo que se encenderá solamente cuando haya un exceso [carry] en la opción de suma y un LED verde que deberá encenderse en caso de que A = B.

S1	S0	FUNCIÓN
0	0	En el display sale el menor entre A y B
0	1	En el display sale el mayor entre A y B
1	0	En el display sale la suma [A + B] y en un LED rojo el carry
1	1	En el display sale A, solo si A > B, en otro caso sale 0

La solución se muestra en la fig. 2.64.

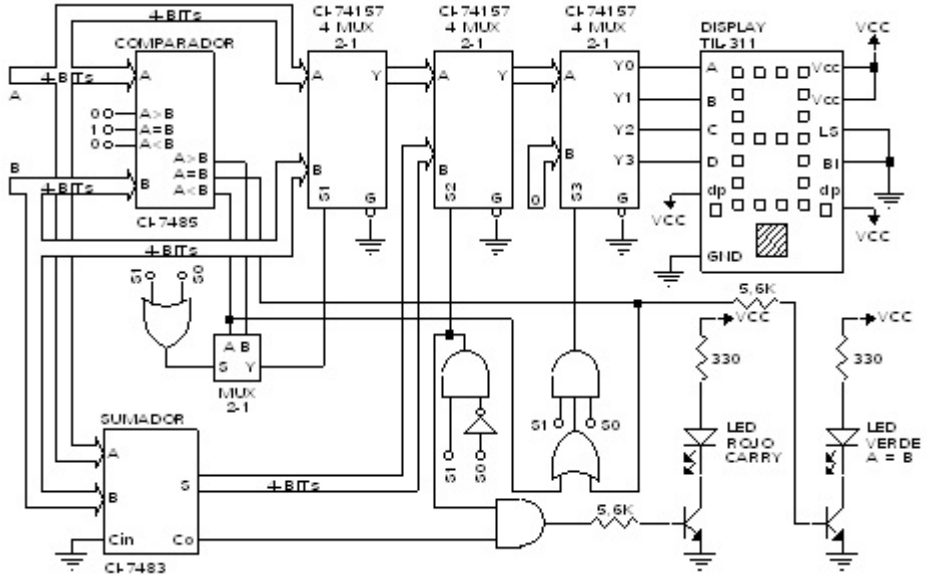


FIGURA 2.64 EJEMPLO 8

Sugerencia.- Analice las soluciones propuestas para cada uno de los ejemplos y comp8uebe la función que ellos cumplen. Modifique el circuito de la fig. 2.63 para que cuando [$S_1 = S_0 = 1$] en el display salga B, siempre que B < A y en cualquier otro caso salga 0, todas las demás opciones siguen igual.

Ejercicios Propuestos

1. Diseñar un circuito digital detector de paridad impar de 4-entradas, utilice un multiplexer del número de entradas adecuado. Implementar el mismo circuito pero con un decoder del número de entradas adecuado.

2. Utilice el CI-7483 para implementar un sumador BCD para números de 3-dígitos BCD cada uno. El resultado debe presentarlo en displays de ánodo común.

3. Diseñar un decodificador de BCD-a-7 segmentos para un display de cátodo común. Implemente el circuito utilizando compuertas NAND.

4. Muestre los bloques principales que, según su criterio, debe tener una calculadora que realiza las cuatro operaciones básicas.

5. Utilice un mismo decodificador del número de entradas adecuado para implementar las siguientes funciones booleanas.

$$G<D, C, B, A> = \sum m(1,2,3,5,6,8,10,11,13,15)$$

$$H<D, C, B, A> = \prod M(0,1,3,5,6,9,11,13,15)$$

6. Mediante el uso de circuitos integrados MSI, diseñar e implementar un circuito lógico que disponga de 2-entradas [A y B] cada una de las cuales recibe un número de 1-dígito codificado en BCD, de manera que en la salida [Y] de 4-bits aparezca el mayor entre A y B, en cualquier otro caso debe salir cero. El resultado debe mostrarse en un display de 7-segmentos de ánodo común.

7. En un manual de CI-TTL, consultar los convertidores de BCD a BIN. Implementar un circuito lógico que realice la conversión de 6-bits BCD a su equivalente BIN.

8. Repita el problema anterior, pero para la conversión de BIN a BCD.

9. Implementar un circuito lógico que realice la conversión de 6-bits BIN a su equivalente BCD.

10. Implementar un circuito lógico que realice la conversión de 8-bits BIN a su equivalente BCD.

11. Utilizar dispositivos MSI, y las compuertas adicionales que requiera, para diseñar un circuito combinacional cuyas entradas son dos números expresados en binario natural [BIN] de 4-bits cada uno A [$A_3A_2A_1A_0$] y B [$B_3B_2B_1B_0$] y dos

líneas de selección de función [S_1 y S_0] que cumpla con la tabla que se indica a continuación. La salida consistirá de un display hexadecimal con decodificador incluido [TIL 311], un LED rojo que se encenderá solamente cuando haya un exceso [carry] en la opción de suma aritmética y un LED verde que deberá encenderse en caso de que A = B.

S1	S0	FUNCIÓN
0	0	En el display sale la suma aritmética de A y B
0	1	En el display sale el mayor entre A y B
1	0	En el display sale la función XOR entre [A + B]
1	1	En el display sale A, solo si A > B, en otro caso sale 0

NOTA: No deje nada indicado, muestre todo el proceso de diseño para cada uno de los problemas, muestre el circuito completo e indique todas las conexiones.

D:\DIGITALES\SD-Cap2.wpd

Revisión: Septiembre - 2008

Multivibradores Biestables

Existen 3-tipos de multivibradores.

MONOESTABLE.-	Un estado estable y uno semi-estable
BIESTABLE.-	Dos estados estables
AESTABLE.-	Ningún estado estable [oscilador]

Los circuitos estudiados hasta ahora son dispositivos lógicos combinacionales, cuyos niveles de salida, en cualquier instante dependen de los niveles presentes en las entradas en ese momento. Es decir, cualquier cambio que ocurra en las entradas tiene un efecto inmediato en las salidas [si se desprecia el retardo natural de los elementos físicos]. Cualquier condición anterior en los niveles de las entradas no tiene efecto en las salidas. Los circuitos combinacionales no tienen realimentación y no disponen de elementos para almacenar información [memoria].

Circuito Combinacional.- En cualquier momento dado, el valor actual de las salidas está determinado exclusivamente por el valor actual de las entradas. En otras palabras, las variables de salida del sistema no dependen del tiempo. Se sobre entiende que los valores de todas las variables son esos en algún

instante único. La fig. 3.1 es un ejemplo de un dispositivo combinacional, en este caso para activar la clave, no importa el orden en que se pongan los números de dicha clave, lo único que interesa es el valor correcto.

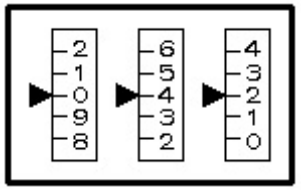


FIGURA 3.1 DISPOSITIVO COMBINACIONAL

Circuito Secuencial.- El valor actual de las salidas depende no sólo del valor actual de las entradas, sino también de la historia del sistema. Es decir, las variables de salida dependen del tiempo.

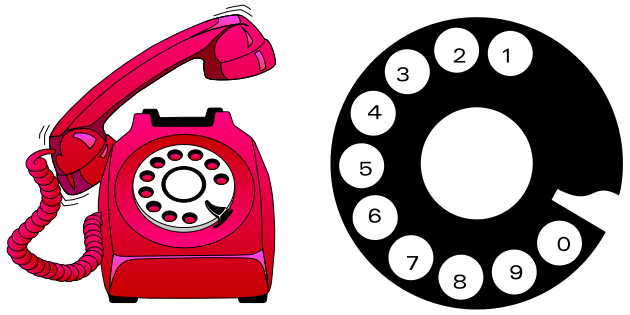


FIGURA 3.2

Un circuito secuencial tiene realimentación y

elementos de memoria para almacenar la información. La fig. 3.2 muestra un ejemplo de dispositivo secuencial, para que la comunicación con otro teléfono se inicie, es necesario que se marque el número correcto y en la secuencia correcta.

En la práctica, la mayoría de los sistemas digitales está constituido por circuitos combinacionales y multivibradores biestables como puede verse en la fig. 3.3. En un sistema digital general hay una realimentación en el sistema, por tanto, las salidas dependen tanto del valor actual de las entradas como de los valores anteriormente almacenados en el circuito de la memoria. El elemento de memoria más importante es el multivibrador biestable, también conocido como flip-flop.

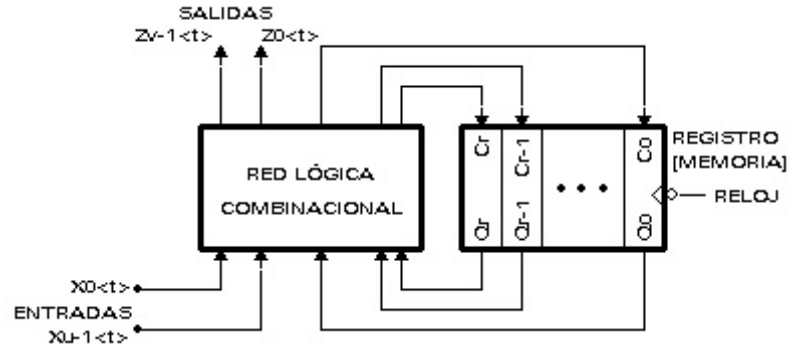


FIGURA 3.3 CIRCUITO SECUENCIAL SINCRÓNICO

Multivibrador Básico con Compuertas NAND

Flip-Flop S-R Asincrónico.- En la fig. 3.4 se muestra el

circuito de un flip-flop [FF] básico implementado con compuertas NAND [también pueden utilizarse compuertas NOR].

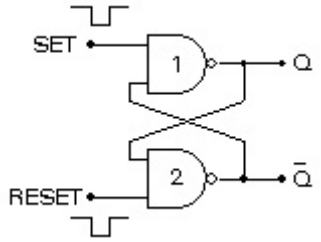
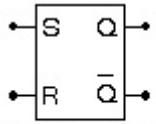


FIGURA 3.4

El FF básico está constituido por dos compuertas NAND con dos entradas de datos: Set y Reset [**S**, **R**] y dos salidas: Q y $\bar{Q}$ que realimentan, en forma cruzada, a las entradas de las compuertas. La principal característica de los FFs es que sus salidas sean complementarias. En los circuitos secuenciales el tiempo es un factor importante, en los FFs hay que diferenciar claramente el valor que tiene la salida antes de que se establezcan las nuevas condiciones en las entradas y que, en este caso, se llamará Q<t> [Qn o Q0, etc.] y el valor que tomará la salida después de que establezcan las nuevas condiciones de las entradas y que se denomina Q<t+1> [Qn+1, Q1, etc.].

Debido a la realimentación, Q<t> se constituye en una entrada al circuito del FF. Por tanto, la tabla de verdad [tabla de función] para este FF, es la que se indica a continuación. La fig.3.5 muestra el símbolo

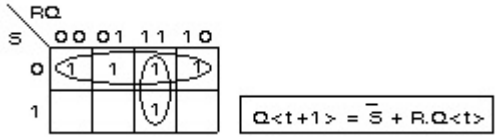
o esquemático del FF - SR asincrónico.



Q<t> = Qn
Q<t+1> = Qn+1
FIGURA 3.5

ENTRADAS			SALIDAS	
S	R	Q<t>	Q<t+1>	$\bar{Q}$<t+1>
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

De la tabla de función del FF S-R asincrónico puede obtenerse la ecuación de Q<t+1>, en función de S, R y Q<t>.



En la tabla se observa que cuando S = 0 y R = 1, Q<t+1> = 1, sin importar el valor de Q<t>; así mismo, cuando S = 1 y R = 0, Q<t+1> = 0, sin importar el valor de Q<t>, de modo que estas dos condiciones de las entradas permiten el ingreso del dato que se quiere almacenar. También, en la tabla se observa que cuando S = 1 y R = 1, Q<t+1> = Q<t>, condición que permite mantener almacenado el último dato que ingresó al FF.

Sin embargo, cuando S = 0 y R = 0, las dos salidas tienen en mismo valor: 1_L, y no son complementarias, razón por la cual esta opción no está permitida, es decir es una "**condición prohibida**", que debe evitarse para que este FF funcione adecuadamente.

ENTRADAS		SALIDAS		
S	R	Q<t+1>	$\bar{Q}$<t+1>	
0	0	1	1	CONDICIÓN PROHIBIDA
0	1	1	0	INGRESO DE DATOS
1	0	0	1	
1	1	Q<t>	$\bar{Q}$<t>	ALMACENAJE DE DATOS

La información contenida en la tabla original se da en forma condensada en la tabla anterior. Algunos autores usan la nomenclatura Qn y Qn+1 en vez de Q<t> y Q<t+1> respectivamente.

La fig. 3.6 muestra cómo sería la respuesta del flip-flop S-R asincrónico para diferentes valores de las entradas S-R.

Al final se puede observar que cuando las entradas S y R son igual a 0 simultáneamente, las dos salidas se ponen en 1, lo cual no corresponde con que sean complementarias; cuando las dos entradas regresan a 1, en las salidas no se sabe cuál será su estado final, por esta razón, la condición S = R = 0, no está permitida, debido a que en la salida se genera una inestabilidad.

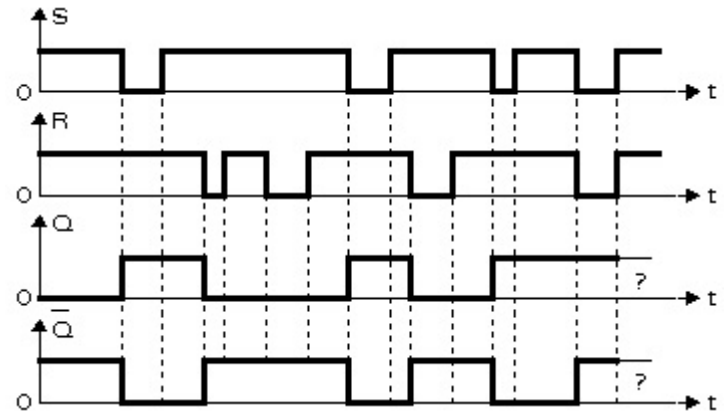


FIGURA 3.6

Eliminación de los Rebotes de los Contactos [Debouncing].- En muchas aplicaciones se utilizan interruptores como el que se muestra en la fig. 3.7 para generar señales digitales.

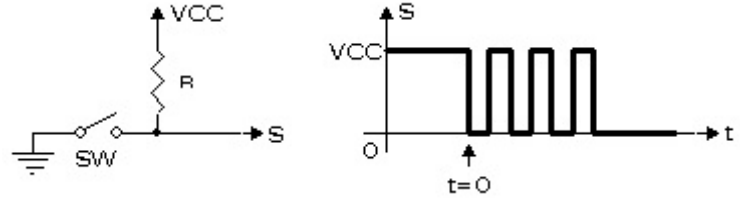


FIGURA 3.7

Debido al coeficiente de elasticidad que tienen los materiales, el interruptor no hace un contacto sólido y definitivo cuando se cierra sino que rebota varias veces [durante algunos milisegundos] hasta que queda en reposo conectando la S a tierra. Estos rebotes

pueden causar molestias en dispositivos electrónicos de alta velocidad, puesto que producen ruido y por esta razón hay que eliminarlos. La fig. 3.8 muestra un flip-flop S-R básico utilizado para eliminar los rebotes. En la fig. 3.9 se muestran las formas de onda en el interruptor y en las salidas del FF-SR.

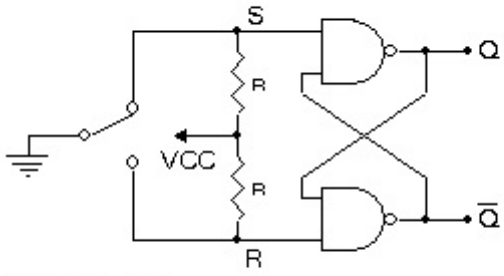


FIGURA 3.8

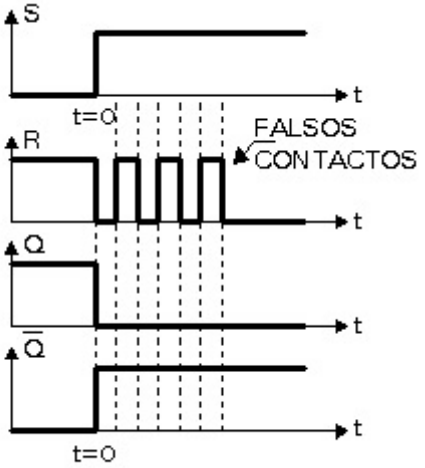


FIGURA 3.9

CI-74279.- En el mercado existe el CI-74279 que tiene 4 flip-flops SR como se indica en la fig. 3.10 y la tabla de función se indica a continuación.

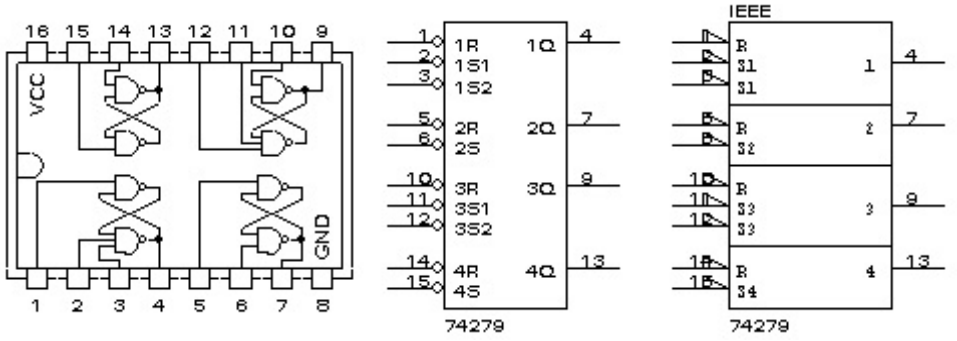


FIGURA 3.10 a) b) c)

ENTRADAS		SALIDA
$\overline{S}$ †	$\overline{R}$	Q
1	1	Q_o
0	1	1
1	0	0
0	0	1*

*Este nivel de salida es seudo estable; esto es, puede no persistir cuando las entradas regresen a su nivel inactivo [1_L], también se conoce como condición prohibida.

†Para los FFs S-R con doble entrada $\overline{S}$, 1 significa que ambas entradas deben ser 1 y 0 significa que una o ambas entradas son 0.

Q_O = nivel de Q antes de que se establezcan las condiciones de entrada indicadas = Q<t>.

FF S-R Sincronizado por Reloj. - En el FF S-R asincrónico, puede darse el caso de que señales indeseables ingresen por las entradas S-R y generen a la salida información no válida.

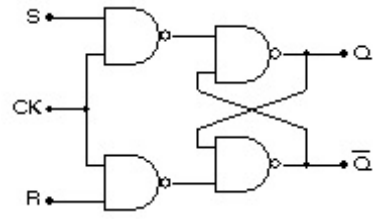


FIGURA 3.11

Para evitar que se almacene información no deseada, se pueden agregar 2-compuertas en las entradas del FF S-R asincrónico, como se muestra en la fig. 3.11. La función de estas compuertas es controlar el instante en que las señales S-R estén habilitadas para producir algún efecto en las salidas Q y $\overline{Q}$.

CK	S	R	Q<t>	Q<t+1>	$\overline{Q}$<t + 1>	
0	X	X	X	Q<t>	$\overline{Q}$<t>	ALMACENA DATO
1	0	0	0	0	1	ALMACENA DATO
1	0	0	1	1	0	
1	0	1	0	0	1	INGRESA DATO
1	0	1	1	0	1	
1	1	0	0	1	0	
1	1	0	1	1	0	
1	1	1	0	1	1	COND. PROHIB.
1	1	1	1	1	1	

En este caso, el ingreso de los datos está sincronizado con una señal de reloj, de modo que cuando la entrada de reloj es 0_L, los datos de las entradas S y R no tienen ninguna influencia en la salida que mantiene su estado anterior [Q<t>] sin importar el valor de S o R o de la salida anterior; la tabla anterior muestra el comportamiento del FF S-R sincrónico. La fig. 3.12 corresponde al símbolo del FF-SR sincrónico; también se incluye la tabla de función simplificada de este FF.

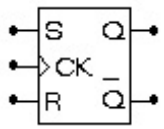


FIGURA 3.12

CK	S	R	Q<t+1>	$\bar{Q}$<t+1>
0	X	X	Q<t>	$\bar{Q}$<t>
1	0	0	Q<t>	$\bar{Q}$<t>
1	0	1	0	1
1	1	0	1	0
1	1	1	CONDICIÓN PROHIBIDA	

Mientras la señal de reloj sea 0_L, las salidas del FF mantienen la información anteriormente almacenada, sin importar los cambios que puedan ocurrir en S y R. Para almacenar un dato, es necesario que la señal de reloj esté en 1_L, en cuyo caso, con S = 0 y R = 1 la salida Q<t> se pone en 0_L; con S = 1 y R = 0, Q<t> = 1_L; con S = 0 y R = 0, la salida mantiene la información anterior [Q<t + 1> = Q<t>]; con S = 1 y R = 1, se presenta la condición prohibida o inestable.

Se dice que este FF tiene entradas sincronizadas, porque el intervalo de muestreo puede ser "temporizado" para que coincida con la aparición de la información deseada en las entradas S y R.

Las señales de entrada Set y Reset no necesitan ser pulsos o cambios momentáneos de nivel. En este caso es la señal de reloj la que cumple ese trabajo. Esta es una característica muy importante cuando las entradas S y R provienen de otro circuito lógico. Como en el caso del FF-SR asincrónico, se puede obtener la ecuación de Q<t>.

$$Q<t> = S + \bar{R}.Q<t>$$

FF Tipo "D" o Retenedor de Datos [Data Latch].- Una manera de evitar la condición prohibida o inestable es con el FF tipo "D" [Data Latch o retenedor de datos] que se muestra en la fig. 3.13, la función del inversor [compuerta 5] es hacer que las entradas S y R, siempre sean el complemento la una de la otra, de esa forma nunca se tendrá la condición prohibida [S = 1 y R = 1]. Este circuito tiene una sola entrada de datos y una señal de reloj. De esta manera se puede ingresar un 0_L o 1_L como se indica en la siguiente tabla de verdad.

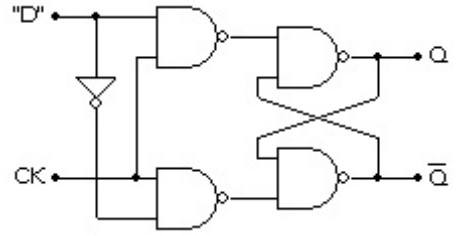


FIGURA 3.13

D	CK	Q<t>
0	1	0
1	1	1
X	0	Q<t>

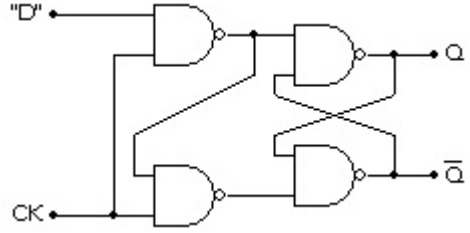
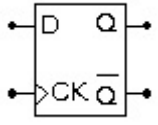


FIGURA N 3.14 a)



b)

La fig. 3.14 (a) muestra una modificación del FF

tipo-D y la fig. 3.14 (b) corresponde al símbolo esquemático. Al retenedor de datos se lo usa con frecuencia para almacenar información proveniente de contadores y computadores hasta que se realice una lectura.

FF S-R Master-Slave [Maestro-Esclavo].- Está construido con 2-FFs S-R sincrónicos conectados en cascada: uno para mantener el estado de la salida [Slave] y otro para guardar la información de las entradas [Master], presente al comienzo del pulso de reloj para posteriormente transferirla a la salida del FF. Como se muestra en la fig. 3.15.

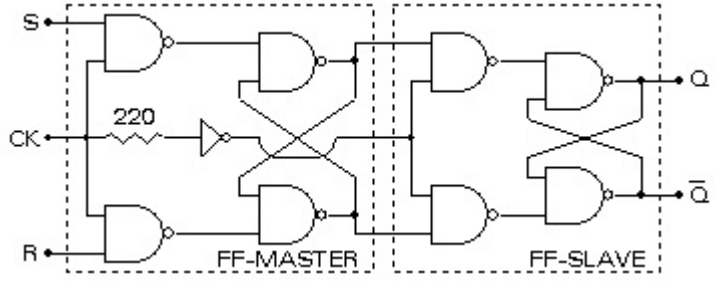


FIGURA 3.15 FLIP-FLOP RS MAESTRO-ESCLAVO

El FF S-R master-slave cumple la misma tabla de verdad del FF S-R sincrónico. La operación del FF S-R/M-S a través de un pulso completo de reloj [fig. 3.16] se describe a continuación.

FIGURA 3.16

1) LAS COMPUERTAS 3 Y 4 SE CIERRAN AISLANDO EL SLAVE DEL MASTER.
2) LAS COMPUERTAS 7 Y 8 SE ABREN HABILITANDO LAS ENTRADAS S Y R DEL MASTER.
3) LAS COMPUERTAS 7 Y 8 SE CIERRAN DESHABILITANDO LAS ENTRADAS DEL MASTER.
4) LAS COMPUERTAS 3 Y 4 SE ABREN CONECTANDO EL MASTER AL SLAVE.

Descripción del Funcionamiento del FF S-R Master-Slave a través de un Pulso de Reloj.- La entrada de reloj normalmente está en 0_L lo que mantiene en 1_L las salida de las compuertas 7 y 8, esto evita que los cambios en las entradas S y R tengan algún efecto en el circuito. Con un 1_L en cada entrada, el flip-flop formado por las compuertas 5 y 6 puede estar en cualquier estado. El FF slave será reconocido como el circuito de memoria sincrónico al que el FF master le proporciona las señales de entrada con la señal de reloj invertida. Cuando la entrada de reloj es 0_L, la salida de la compuerta 9 es 1_L de manera que las compuertas 3 y 4 están abiertas. De donde el FF de las compuertas 1 y 2 estará en el mismo estado del FF de las compuertas 5 y 6 del FF-master.

El nivel 0_L de la señal de reloj está conectado, a través de una resistencia de 220Ω, a la entrada de la compuerta 9. Esto hace que la compuerta 9 esté un poco más próxima al estado 1_L que las compuertas 7

y 8 [fig. 3.17]: cuando la fuente V es 0V, la base del transistor está polarizada directamente e IB ≈ 1mA, esta corriente circula por el emisor del transistor, en cuyo caso, el voltaje en el punto X es ≈ 0,2V más alta que el valor de CK. Cuando se aplica el pulso de reloj, ocurre una secuencia de 4 pasos.

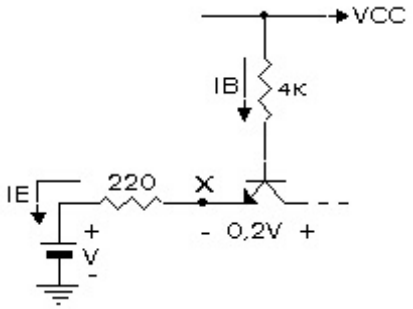


FIGURA 3.17 CIRCUITERÍA DE UNA ENTRADA TTL

Primero, cuando el reloj se hace positivo, debido a la resistencia de 220Ω antes mencionada, la compuerta 9 alcanza el estado 1_L antes que las compuertas 7 y 8. Un 1_L a la entrada de la compuerta 9 produce un 0_L en su salida que cierra las compuertas 3 y 4 y aísla el FF slave del master. Este aislamiento ocurre antes de que pueda darse algún cambio en el estado del master. Así el estado del FF master se almacena en las salidas Q y $\bar{Q}$ del slave.

Segundo, las compuertas 7 y 8 se abren con la señal de reloj y la información de las entradas S y R determina el estado del FF master de acuerdo con la

tabla de verdad del FF sincrónico.

Tercero, el pulso de reloj empieza a descender cerrando las compuertas 7 y 8 que aíslan el FF master de las entradas S y R.

Cuarto, la compuerta 9 recibe un 0_L en su entrada, lo que permite abrir las compuertas 3 y 4. En este momento las salidas del FF master se transfieren al FF slave y aparecen en los terminales del circuito.

CK	S	R	Q<t+1>	$\bar{Q}$<t+1>
0	X	X	Q<t>	$\bar{Q}$<t>
⊓	0	0	Q<t>	$\bar{Q}$<t>
⊓	0	1	0	1
⊓	1	0	1	0
⊓	1	1	CONDICIÓN PROHIBIDA	

De esta manera, los cambios de la salida no ocurren hasta que haya terminado el pulso de reloj. Por esta razón, los efectos de los cambios de la salida no pueden aparecer en los terminales de entrada durante el pulso de reloj; es decir, durante el pulso de reloj los datos de los terminales S y R deben permanecer estables. La tabla de verdad anterior muestra el comportamiento del FF S-R master-slave.

FF J-K Master-Slave.- Este tipo de FF ofrece una gran versatilidad, evita problemas de temporización, condición prohibida y formas de onda independientes

del acoplamiento DC. La fig. 3.18 muestra el circuito del FF - JK maestro-esclavo implementado con compuertas NAND y la fig. 3.19 corresponde al símbolo esquemático del FF-JK.

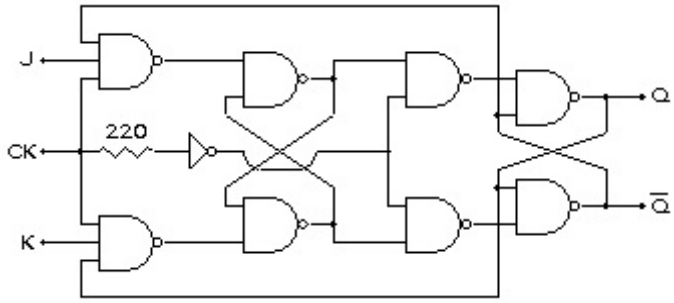


FIGURA 3.18 FF - JK MAESTRO-ESCLAVO

CK	J	K	Q<t>	Q<t+1>	$\bar{Q}$<t+1>
0	X	X	X	Q<t>	$\bar{Q}$<t>
⊓	0	0	0	0	1
⊓	0	0	1	1	0
⊓	0	1	0	0	1
⊓	0	1	1	0	1
⊓	1	0	0	1	0
⊓	1	0	1	1	0
⊓	1	1	0	1	0
⊓	1	1	1	0	1

En esta tabla se observa que este tipo de FF no tiene condiciones prohibidas, cuando J = K = 1, la salida próxima es el complemento del estado anterior, por tanto este es el FF que se utilizará para aplicaciones

de diseño junto con el FF tipo D. La ecuación de este FF se deduce de la tabla de función para CK = 1 y se indica a continuación.

$$Q<t+1> = J\bar{Q}<t> + \bar{K}Q<t>$$

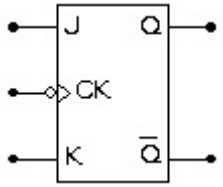


FIGURA 3.19

CK	J	K	Q<t+1>	$\bar{Q}$<t+1>
0	X	X	Q<t>	$\bar{Q}$<t>
⊓	0	0	Q<t>	$\bar{Q}$<t>
⊓	0	1	0	1
⊓	1	0	1	0
⊓	1	1	$\bar{Q}$<t>	Q<t>

El FF J-K master-slave, es el mismo que el S-R Master-Slave excepto que las salidas están conectadas en forma cruzada a las entradas para obtener una operación de complemento [Toggle], cuando J = 1_L y K = 1_L. La tabla de verdad anterior muestra la función del FF J-K M-S.

Tablas de excitación de los FFs JK y tipo D.- En ocasiones conviene saber lo que se debe poner en las entradas de los flip flops, por ejemplo del JK para obtener un valor de Q<t+1> deseado. Para esto se elabora las tablas de excitación de dichos flip flops, en este caso de los flip flops JK y D. En las tablas se observa que para mantener el valor 0L en la salida del flip flop JK [Q<t> = 0 y Q<t+1> = 0], hay que pone 0L en J y no importa "X" en K; mientras que en el flip flop

tipo D en la entrada hay que poner 0. De esa manera se continúa el análisis.

Q<t>	Q<t+1>	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Q<t>	Q<t+1>	D
0	0	0
0	1	1
1	0	0
1	1	1

TABLAS DE EXCITACIÓN DE LOS FLIP-FLOPS J-K Y TIPO D

Flip-Flop tipo "T".- El FF-T es una versión de una sola entrada del FF-JK. El FF-T se obtiene cortocircuitando las entradas J y K. La denominación T proviene de la propiedad del FF para "conmutar" [del inglés Toggle = complementar], es decir, que cambia de estado con cada pulso de reloj. La fig. 3.20 muestra el símbolo lógico y las formas de onda del FF-T disparado por transición negativa. Puede verse que la onda de salida Q del FF-T tiene una frecuencia que es la mitad de la onda del reloj cuando la entrada T es alta [1_L].

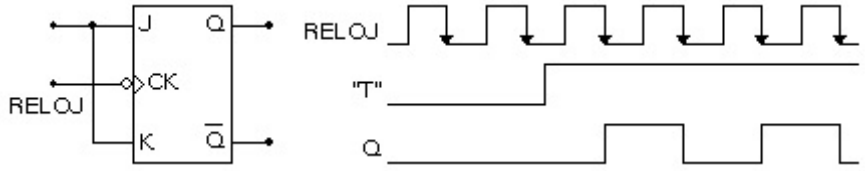


FIGURA 3.20 a) b)

La tabla de verdad, se muestra en seguida. De ella se deducen las ecuaciones para las entradas a los FFs J-K y D respectivamente, para que funcionen como FFs

tipo "T".

CK	T	Q<t>	Q<t+1>	J	K	D
⊓	0	0	0	0	X	0
⊓	0	1	1	X	0	1
⊓	1	0	1	1	X	1
⊓	1	1	0	X	1	0

$J = T$ $K = T$ Para el Flip-Flop JK

$D = T \oplus Q<t>$ Para el Flip-Flop

El FF-T se lo obtiene de un FF-JK manteniendo las entradas J y K en 1_L.

La fig. 3.21 muestra cómo puede obtenerse un FF-T a partir de un FF-D. Los FFs-T se utilizan a menudo en contadores asincrónicos y divisores de frecuencia.

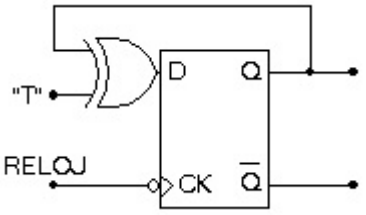


FIGURA 3.21

Entradas Asincrónicas.- Las entradas de los FFs que hasta aquí se han estudiado, están sincronizadas con la señal de reloj, lo que significa que las salidas sólo cambiarán cuando se den las condiciones necesarias en las entradas S-R, D, J-K o T y la señal de reloj.

Hay ocasiones en las que conviene poder cambiar los datos del FF de manera independiente del reloj, para eso se han incluido las llamadas entradas asincrónicas, son 2 y reciben el nombre de ***Preset*** la una y ***Clear*** la otra. Generalmente el nivel activo de estas entradas asincrónicas es bajo [0_L]. De manera que cuando la entrada Preset = 0_L, entonces, la salida Q = 1, [$\bar{Q} = 0$], sin importar el valor de las otras entradas incluida la entrada de reloj; y cuando la entrada Clear = 0_L, entonces, la salida Q = 0, [$\bar{Q} = 1$], sin importar el valor de las otras entradas incluida la entrada de reloj. No está permitido que las dos entradas asincrónicas [CL y PR] tengan el nivel activo [0_L] al mismo tiempo, porque esto produce una condición prohibida o inestable.

Flip-Flops Disparados por Transición.- Una mejora en el sincronismo de los FFs que disminuye los problemas de tiempo, son los FFs disparados por transición [flanco]. Se puede usar la transición positiva [↑] de la señal de reloj o la transición negativa [↓] del mismo. Las ventajas de no tener la entrada activa durante el pulso del reloj en el diseño de un sistema, son: velocidad de operación y control del ancho del pulso. Sin embargo, generalmente son más complejos.

Flip-Flop tipo "D" Disparado por Transición Positiva.- La fig. 3.22 nuestra el circuito y la tabla de verdad. La fig. 3.23 es el símbolo del FF tipo-D disparado por

transición positiva. La entrada de datos y su transferencia ocurren con la transición positiva [↑] de la señal de reloj.

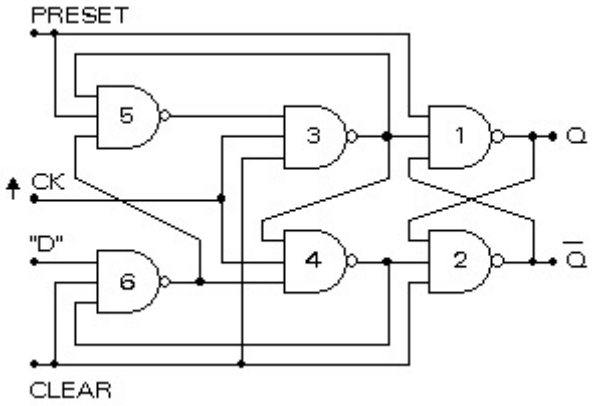


FIGURA 3.22 CI-7474 FF TIPO "D", DISPARADO POR TRANSICIÓN

ENTRADAS				SALIDAS	
PR	CL	CK	D	Q<t+1>	$\bar{Q}$<t+1>
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1*	1*
1	1	↑	0	0	1
1	1	↑	1	1	0
1	1	0	X	Q<t>	$\bar{Q}$<t>

*Condición prohibida [inestable].

Las compuertas 1 y 2 conforman el circuito biestable con las funciones Preset y Clear. La salida de las compuertas 3 y 4 determinan el estado de la salida del FF. Las compuertas 5 y 6 determinan qué salida

[de las compuertas 3 y 4, pero no ambas] será 0_L, en respuesta a la señal de disparo aplicada en la entrada del reloj.

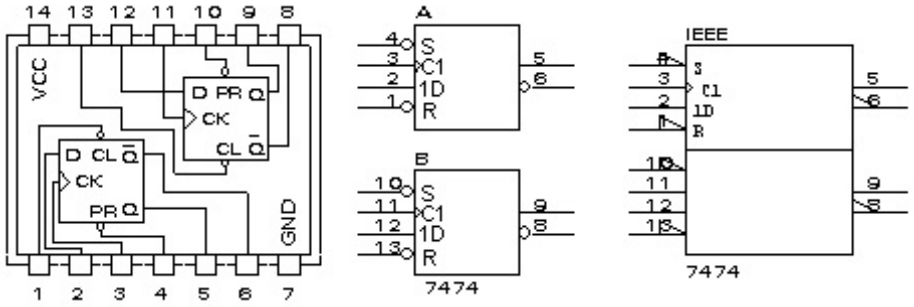


FIGURA 3.23 CI-7474 2-FFs TIPO "D"

Ecuaciones de Salida de los FFs.- El comportamiento de un biestable o FF puede describirse mediante una ecuación característica que especifica el estado siguiente en función de sus entradas y estado actual. Las ecuaciones características de los FFs se presentan en la siguiente tabla.

Tipo de Flip-Flop	Ecuación
S-R Sincrónico	$Q<t+1> = S + \bar{R}Q<t>$
S-R Master/Slave	$Q<t+1> = S + \bar{R}Q<t>$
D	$Q<t+1> = D$
D disparado por transición	$Q<t+1> = D$
J-K Master/Slave	$Q<t+1> = J\bar{Q}<t> + \bar{K}Q<t>$
J-K disparado por transición	$Q<t+1> = J\bar{Q}<t> + \bar{K}Q<t>$
T	$Q<t+1> = \bar{Q}<t>$

Por convención, Q<t> significa el valor que tiene el FF antes del pulso de reloj y Q<t+1> el valor que adquiere la salida del FF después del pulso o de la transición del reloj.

A continuación se estudian algunos FFs J-K disponibles en el mercado.

74LS76.- La versión 7476 comparte la misma distribución de pines y tiene 2-FF-JK-MS, mientras que el 74LS76 tiene 2-FF-JK disparados por transición negativa.

La fig. 3.24 (a) corresponde a la distribución de pines del CI-7476, la fig. 3.24 (b), muestra como están distribuidos los dos FFs J-K, dentro del integrado; la fig. 3.24 c) corresponde al símbolo IEEE.

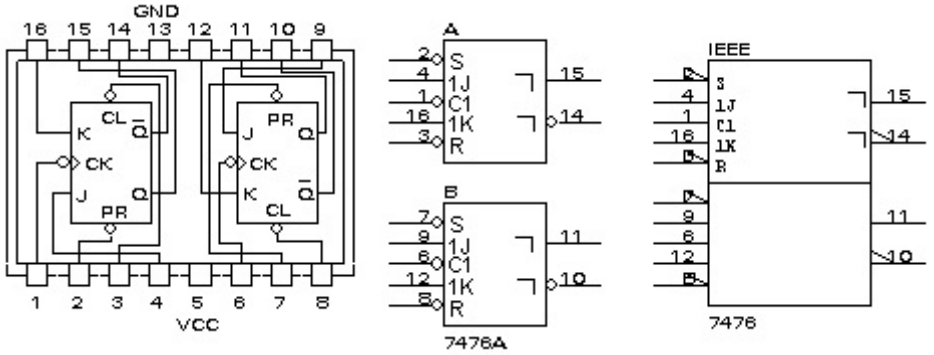


FIGURA 3.24 a) b) c)

CI-7476						
ENTRADAS					SALIDAS	
PR	CLR	CK	J	K	Q	$\bar{Q}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1

74LS76						
ENTRADAS					SALIDAS	
PR	CLR	CK	J	K	Q	$\bar{Q}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1

0	0	X	X	X	1*	1*
1	1	⊓	0	0	Q_0	$\overline{Q}_0$
1	1	⊓	0	1	0	1
1	1	⊓	1	0	1	0
1	1	⊓	1	1	$\overline{Q}_0$	Q_0

0	0	X	X	X	1*	1*
1	1	↓	0	0	Q_0	$\overline{Q}_0$
1	1	↓	0	1	0	1
1	1	↓	1	0	1	0
1	1	↓	1	1	TOGGLE	
1	1	1	X	X	Q_0	$\overline{Q}_0$

* Esta condición es inestable; esto es, no permanecerá cuando las entradas Preset y el Clear regresen a su nivel inactivo [1L].

74LS107.- La versión 74107 comparte la misma distribución de pines y tiene 2-FF-JK-MS, mientras que el 74LS107 tiene 2-FF-JK disparados por transición negativa. La fig. 3.25 a) corresponde a la distribución de pines del CI-74107 y la fig. 3.25 b), muestra como están distribuidos los dos FFs J-K, dentro del integrado [IEEE], la fig. 25 c), muestra los FFs de manera tradicional. Observe que solo se dispone de la entrada asincrónica Clear.

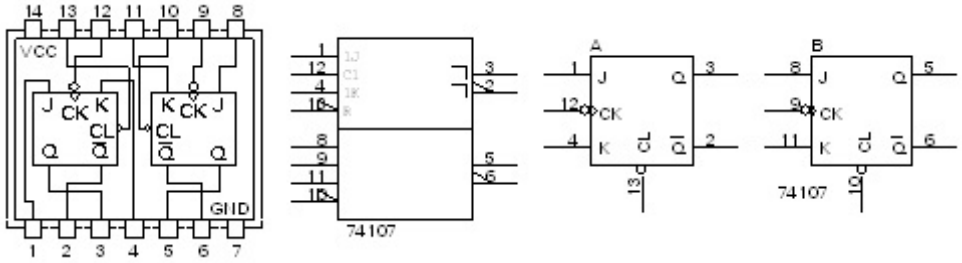


FIGURA 3.25 a) b) c)

CI-747107					
ENTRADAS				SALIDAS	
CL	CK	J	K	Q	$\overline{Q}$
0	X	X	X	0	1
1	⊓	0	0	Q_0	$\overline{Q}_0$
1	⊓	0	1	0	1
1	⊓	1	0	1	0
1	⊓	1	1	TOGGLE	

CI-74LS107					
ENTRADAS				SALIDAS	
CL	CK	J	K	Q	$\overline{Q}$
0	X	X	X	0	1
1	↓	0	0	Q_0	$\overline{Q}_0$
1	↓	0	1	0	1
1	↓	1	0	1	0
1	↓	1	1	TOGGLE	
1	1	X	X	Q_0	$\overline{Q}_0$

74LS112.- El 74LS112 tiene 2-FF-JK disparados por transición negativa. La fig. 3.26 a) corresponde a la distribución de pines del CI-74LS112 y la fig. 3.26 b), muestra como están distribuidos los dos FFs J-K, dentro del integrado [IEEE]. La tabla de función se muestra a continuación.

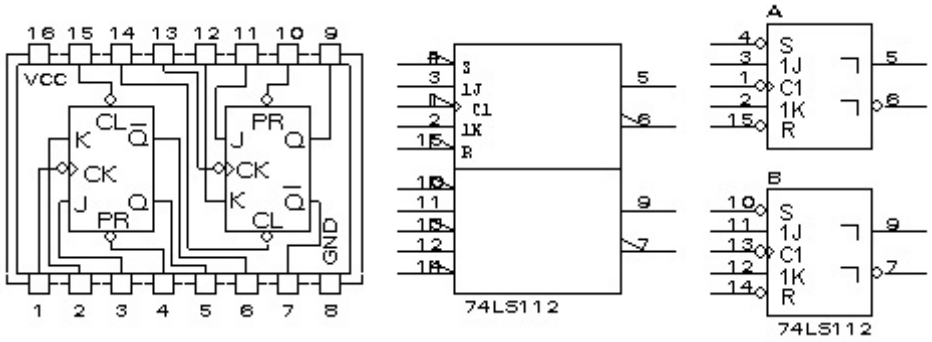


FIGURA 3.26 a) b) c)

ENTRADAS					SALIDAS	
PR	CL	CK	J	K	Q	$\overline{Q}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1*	1*
1	1	↓	0	0	Q_0	$\overline{Q}_0$
1	1	↓	0	1	0	1
1	1	↓	1	0	1	0
1	1	↓	1	1	TOGGLE	
1	1	1	X	X	$\overline{Q}_0$	Q_0

* Esta condición es inestable; esto es, no permanecerá cuando las entradas preset y el clear regresen a su nivel inactivo [1L].

Para los FFs 7474, 7476, 74107 y 74112, Q_0 = Q<t> y Q = Q<t+1>.

Aplicaciones del FF - JK.- Debido a que el FF-JK no tiene condiciones prohibidas, es el que se encuentra disponible en el mercado y se presentan de dos tipos: Master-Slave y disparados por transición. Existe un número ilimitado de aplicaciones con FFs, algunas de ellas se estudian a continuación.

Contadores/Divisores de Frecuencia.- En muchas ocasiones es necesario contar eventos que se producen en la naturaleza, o controlar la secuencia en las que se realizan. En otras situaciones será necesario medir la frecuencia con la que ocurren los eventos. En estos casos es necesario disponer de un contador. Conviene indicar que los contadores constituyen una clase de registros [agrupación de FFs], que son muy utilizados en sistemas digitales. Los contadores se dividen en dos grupos: Asincrónicos y Sincrónicos.

Contadores Asincrónicos [MOD-2^n].- Son aquellos en los que cada FF dispone de una señal de reloj diferente. Se los construye conectando FFs J-K en cascada, la señal del reloj principal se la conecta a la entrada CK del FF menos significante; la salida Q de este FF será la entrada de reloj del siguiente FF y así sucesivamente; las entradas J y K de todos los FFs que conforman el contador asincrónico se conectan a 1_L. Para implementar el contador asincrónico MOD-2^n, se requieren n-FFs, donde MOD = módulo y significa el número de valores diferentes que se generan a la salida del contador.

La fig. 3.27 muestra un contador binario asincrónico básico módulo-8 [MOD-8]. En ella se presentan las formas de onda para la señal de reloj y para las salidas de los FFs. Si a cada salida se le asigna una ponderación, fácilmente puede deducirse que corresponde a una cuenta ascendente desde 0 hasta 7 inclusive.

De modo que este circuito es un contador MOD-8 porque tiene 8 estados [valores] diferentes. En la fig. 3.27, la entrada asincrónica *clear* [CL] se conecta a una señal de borrado constituida por una resistencia conectada a V_{CC} y un interruptor pulsante conectado a tierra [GND]. El borrado sirve para iniciar con cero [0] el contador. A veces, la entrada de borrado, recibe el nombre de Master-Reset [M. R.]. Cada vez que se pulsa el interruptor se genera un cero en la entrada clear de cada FF con lo que la salida Q de todos los FFs se pone en cero.

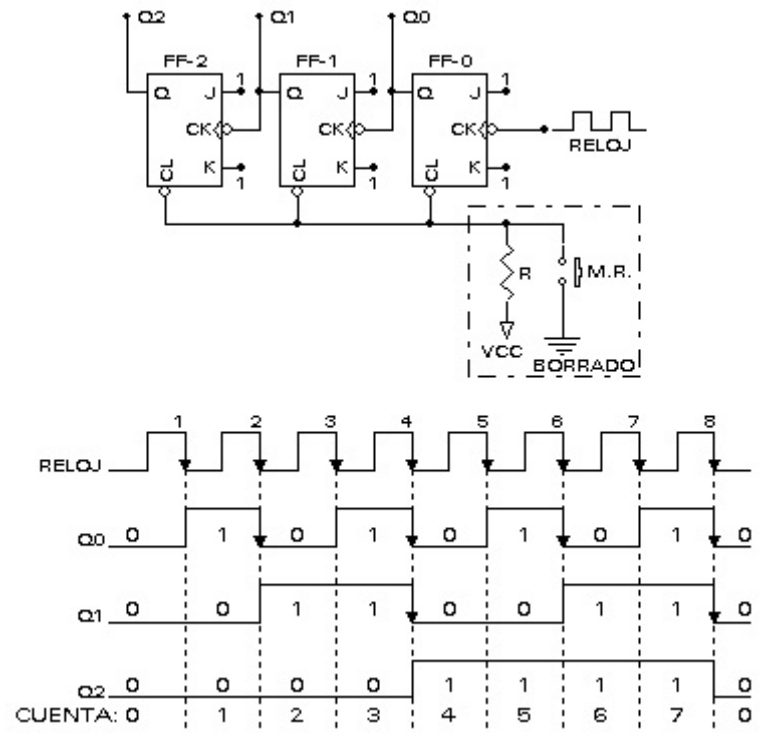


FIGURA 3.27

Si se hace un análisis más detenido de estas formas de onda, se puede ver que si la señal de reloj es una onda de período T [frecuencia f], el período de Q_0<t> corresponden a 2T, lo que representa una frecuencia que sería la mitad de la frecuencia del reloj original [f/2]. Así mismo, Q_1<t> tiene un período de 4T, con respecto al período del reloj, es decir, una frecuencia que es la cuarta parte de la frecuencia del reloj. De igual manera, el período de Q_2<t> es 8T y su frecuencia es $f_{(reloj)}/8$. De aquí se concluye que un contador también es un divisor de frecuencia. Cada FF divide la frecuencia de entrada por 2. Entonces, añadiendo más FF se pueden tener divisores de frecuencia para 2, 4, 8, etc. hasta 2^n, donde n representa el número de FFs utilizados. Se puede ver que las entradas J y K de todos los FFs están a 1_L [funcionan como FFs tipo "T"]. Se debe indicar que este tipo de conexión [asincrónica] hace que el retardo de los FFs se acumule, por ejemplo, a la salida del primer FFs, se tiene un retardo Δt, a la salida del segundo $2\Delta t$, etc. de modo que solo sirven para contadores de baja frecuencia.

Contador Asincrónico MOD-M [diferente de 2^n].- En ocasiones se necesitan contadores de módulo diferente a 2^n, por ejemplo MOD-10, etc., en ese caso, se implementa primero un contador MOD 2^n y luego se lo modifica mediante el uso de la entrada asincrónica

Clear [borrado].

Ejemplo.- Implementar un contador asincrónico MOD-10. Primero se implementa un contador MOD-16 y luego se obtiene el equivalente binario del valor del módulo deseado, en este caso $10_{10} \equiv 1010_2$.

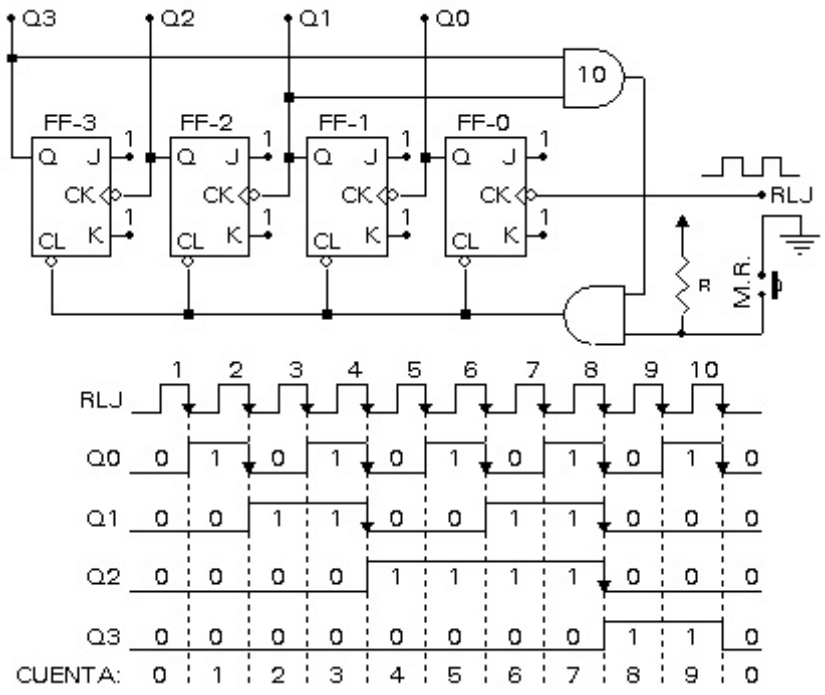


FIGURA 3.28 CONTADOR BINARIO ASINCRÓNICO MÓDULO 10

Si las salidas del contador se etiquetan como $Q_3Q_2Q_1Q_0$, entonces se utilizarán las salidas que, en el equivalente binario, generan 1s [$Q_3 = 1$ y $Q_1 = 1$, en este caso] para conectarlas a las entradas de una

compuerta NAND cuya salida, a su vez, se conectará a la entrada CL de todos los FFs, a través de una compuerta AND, para incluir el borrado manual.

El circuito completo se muestra en la fig. 3.28, en la que se ha incluido un borrado manual y las formas de onda del reloj y de las salidas del contador MOD-10, empezando en 0.

Los contadores asincrónicos son fáciles de implementar y útiles para aplicaciones de baja frecuencia. Debido a que al modificar el módulo se presentan pulsos espurios de corta duración [del orden de los ns], no son recomendables para aplicaciones de alta frecuencia, además debido a la acumulación de los retardos de tiempo de cada flip-flop.

Contadores Ripple-Clock.- El problema de los pulsos espurios, en parte, se soluciona mediante el uso de los contadores Ripple-Clock [R. C.] que son otro tipo de contadores asincrónicos. La fig. 3.29 muestra una forma en la que se puede diseñar este tipo de contadores. Para estudio se ha seleccionado un contador R. C. MOD-11, diseñado con FFs-JK.

Al realizar el diseño de este tipo de contadores es conveniente dibujar las formas de onda del reloj, de la salida Q de cada uno de los FFs y los valores que deberá ponerse en las entradas J y K de dichos FFs, como se indica en la fig. 3.29.

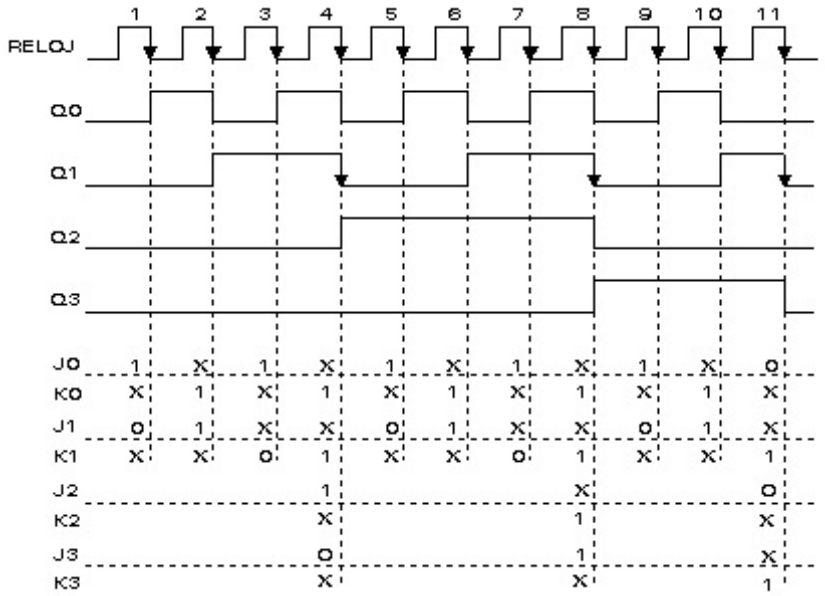


FIGURA 3.29 FORMAS DE ONDA DEL CONTADOR MÓDULO 11

Una de las primeras cosas que hay que hacer en el diseño de este tipo de contadores, es determinar cuál será la señal de reloj que comandará individualmente a cada FF, esto se evidencia en el ejemplo. Para generar las formas de onda del FF-0, siempre se utilizan las transiciones [negativas o positivas, dependiendo del tipo de FF, en este caso negativas] de la señal del reloj principal. De las formas de onda de los FFs utilizados, se determinan los valores que deben ponerse en las entradas de los FFs y de ellas se obtienen las ecuaciones para las entradas J_0 y K_0, J_1 y K_1, J_2 y K_2, J_3 y K_3, respectivamente como se indica a continuación.

Las ecuaciones para el FF-0 son

$$\left.\begin{array}{l} J_0 = \bar{Q}_1 + \bar{Q}3 = \overline{Q_1 Q_3} \\ K_0 = 1 \end{array}\right\} \quad CK_0 = \text{Reloj principal}$$

Para generar las formas de onda del FF-1, se analiza si se puede utilizar, como señal de reloj, la salida del FF-0, para esto es necesario que por cada cambio de nivel de Q_1, haya una transición negativa correspondiente en la salida Q_0. Si esto no se da, se repite el análisis con la señal anterior, en esta ocasión con el reloj principal. Para este ejemplo, Q_0 no cumple con lo requerido como señal de reloj para el FF-1, por tanto se utilizará el reloj principal. De las formas de onda de Q_1, y de los valores de J_1 y K_1, se deducen las siguientes ecuaciones.

$$\left.\begin{array}{l} J_1 = Q_0 \\ K_1 = Q_0 + Q_3 \end{array}\right\} \quad CK_1 = \text{Reloj principal}$$

Para determinar la señal de reloj de los restantes FFs, se realiza un análisis similar: se empieza con la señal Q del FF inmediato-anterior y se observa si dispone de una transición negativa para cada cambio de nivel de la señal de salida del FF que se está analizando, si cumple este requisito, se utiliza esta señal como reloj, sino se procede a analizar con la salida anterior hasta encontrar la que cumpla la condición. Para el FF-2, se observa que Q_1 tiene una

transición negativa para cada cambio de nivel de la salida Q_2, por tanto Q_1 será la señal de reloj para el FF-2; de los valores obtenidos para J_2 y K_2, respectivamente se deducen las ecuaciones para este FF.

$$\left.\begin{matrix} J_2 = \bar{Q}_3 \\ K_2 = 1 \end{matrix}\right\} \quad CK_2 = Q_1$$

Puesto que Q_2 no dispone de una transición negativa para cada cambio de nivel de la salida del FF-3, se procede a realizar el análisis con la salida Q_1, la cual cumple con el requisito, por lo que Q_1 será utilizada como reloj del FF-3, cuyas ecuaciones de salida se muestran a continuación.

$$\left.\begin{matrix} J_3 = Q_2 \\ K_3 = 1 \end{matrix}\right\} \quad CK_3 = Q_1$$

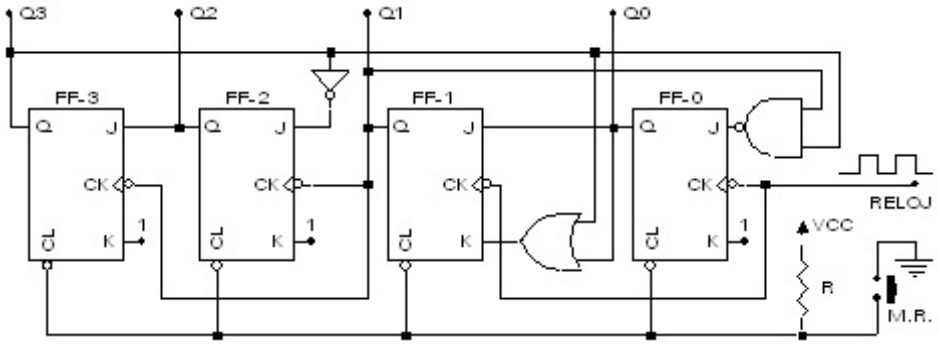


FIGURA 3.30 CONTADOR RIPPLE CLOCK MÓDULO 11

El contador MOD-11, ripple-clock, con FFs-JK se muestra en la fig. 3.30.

Contadores Ripple-Clock en CI.- Los fabricantes de CIs han diseñado algunos contadores del tipo Ripple-Clock.

MOD-10 [Década]: 7490, 74176, 74196, 74290, 74390
MOD-12 [Divisor por 12]: 7492
MOD-16 [Binarios de 4-bits]: 7493, 74177, 74293, 74393

La fig. 3.31 muestra el diagrama de bloques de estos 3 CIs; se puede observar que disponen de dos contadores internos que se pueden usar de forma independiente con sus propias entradas de reloj: A y B; también se los puede conectar en cascada.

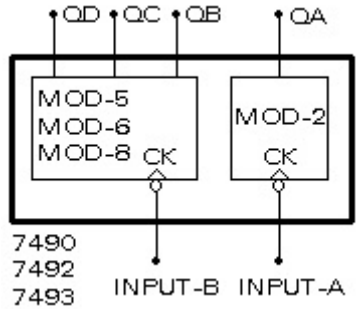


FIGURA 3.31

En esta sección se estudiarán algunos de los más utilizados en aplicaciones prácticas de baja frecuencia, como es el caso del CI-7490, CI7492 y CI-7493.

CI-7490.- Junto con el CI-7492 y el CI-7493, que son contadores monolíticos implementados con 4-FFs M-S y compuertas adicionales para proporcionar un contador-divisor por 2 y un contador binario de 3-FFs para el que la longitud del ciclo de cuenta es división por 5 ['90], por 6 ['92] y por 8 ['93]. La fig. 3.32 muestra la distribución de pines del CI-7490.

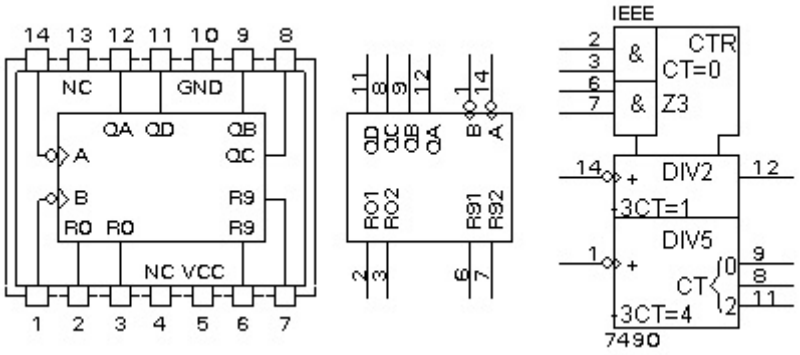


FIGURA 3.32 CI-7490

Todos estos contadores tienen dos entradas [a través de una compuerta NAND] para inicialización con 0 [borrar]. El CI-7490 también dispone de otras dos entradas similares para establecer una salida igual a 9 para utilizarla en aplicaciones de complemento a 9 en aplicaciones BCD.

Para utilizar la máxima longitud de cuenta de estos contadores [década, división por 12, o binario de 4-bits], la salida Q_A se conecta a la entrada de reloj B. Los pulsos de entrada se aplican a la entrada de reloj A y las salidas se muestran en las respectivas

tablas de función proporcionadas por los fabricantes. Una cuenta con división por 10 son salida simétrica se puede obtener con el CI-7490 conectando la salida Q_D a la entrada A, mientras que los pulsos de entrada se aplican a la entrada B que proporciona una onda cuadrada con división por 10 en la salida Q_A.

Secuencia de cuenta BCD del CI-7490 [ver nota A]				
Cuenta	Salidas			
	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Secuencia bi-quinaria [5-2]) del CI-7490 [ver nota B]				
Cuenta	Salidas			
	Q_A	Q_D	Q_C	Q_B
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

ENTRADAS RESET				SALIDAS			
R0(1)	R0(2)	R9(1)	R9(2)	Q_D	Q_C	Q_B	Q_A
1	1	0	X	0	0	0	0
1	1	X	0	0	0	0	0
X	X	1	1	1	0	0	1
X	0	X	0	CUENTA			
0	X	0	X	CUENTA			
0	X	X	0	CUENTA			
X	0	0	X	CUENTA			

Notas: A Para cuenta BCD, la salida Q_A se conecta a la entrada B.

B Para cuenta bi-quinaria, la salida Q_D se conecta a la entrada A.

Aplicaciones del CI-7490.- Son muchas las aplicaciones que se le puede dar a este contador década.

Ejemplo.- Diseñar un contador MOD = 100 y un contador MOD = 85. En primer lugar se diseña el contador MOD-100_{10}. El circuito resultante para implementar el contador MOD = 100_{10} se muestra en la fig. 3.33.

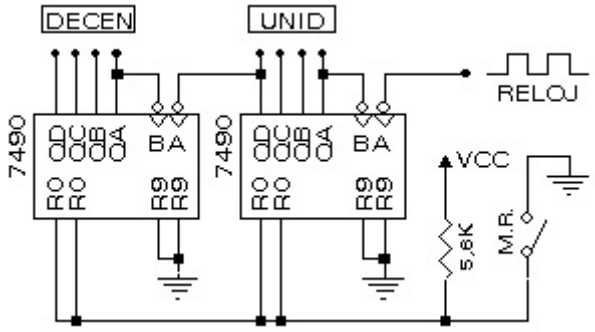


FIGURA 3.33 CONTADOR R. C. MÓDULO 100

El diseño del contador MOD-85_{10}, se obtiene modificando el contador MOD = 100, para lo cual se escribe el equivalente BCD del valor del módulo, en este caso 85_{10} = 1000 0101_{BCD}, y se lo detecta a través de una compuerta AND, para eso se realimentan las salidas Q que generan los 1s del número BCD obtenido, en este caso: Q_D de las decenas, Q_C y Q_A de las unidades, [para señalar esta detección se ha puesto el número 85 dentro de la compuerta AND]. La compuerta OR, junto con la

resistencia y el interruptor, sirven para incluir un borrado manual que puede necesitarse en cualquier momento para inicializar con 0 el contador [fig. 3.34].

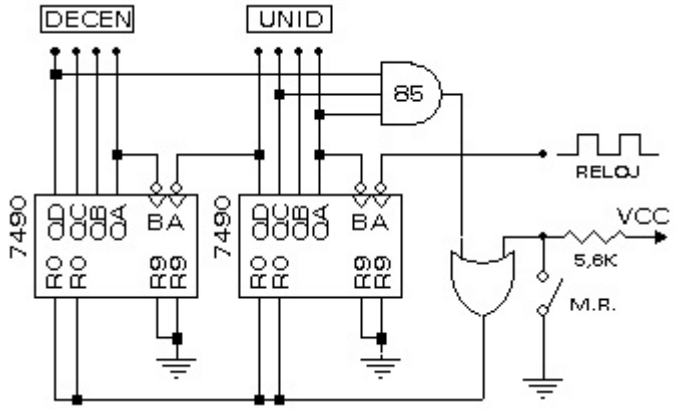


FIGURA 3.24 CONTADOR RIPPLE CLOCK MÓDULO 85

Ejemplo.- Diseñar un contador MOD = 1000 y un contador MOD = 742. El contador módulo 1000, se muestra en la fig. 3.35.

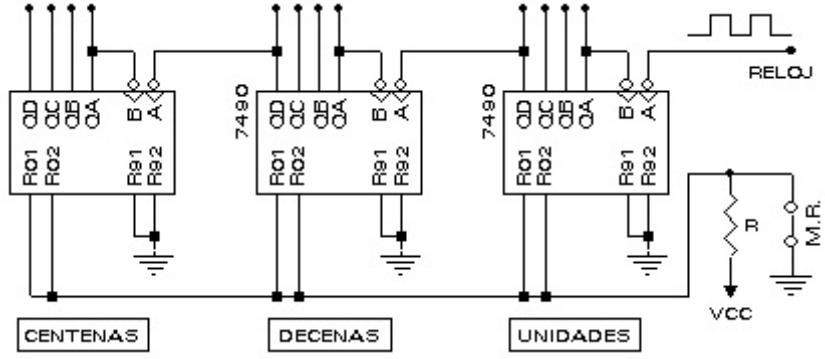


FIGURA 3.35 CONTADOR RIPPLE CLOCK MÓDULO 1000

En este caso se requieren 3 CI-7490. La circuitería adicional sirve para incluir un borrado manual o Master-Reset.

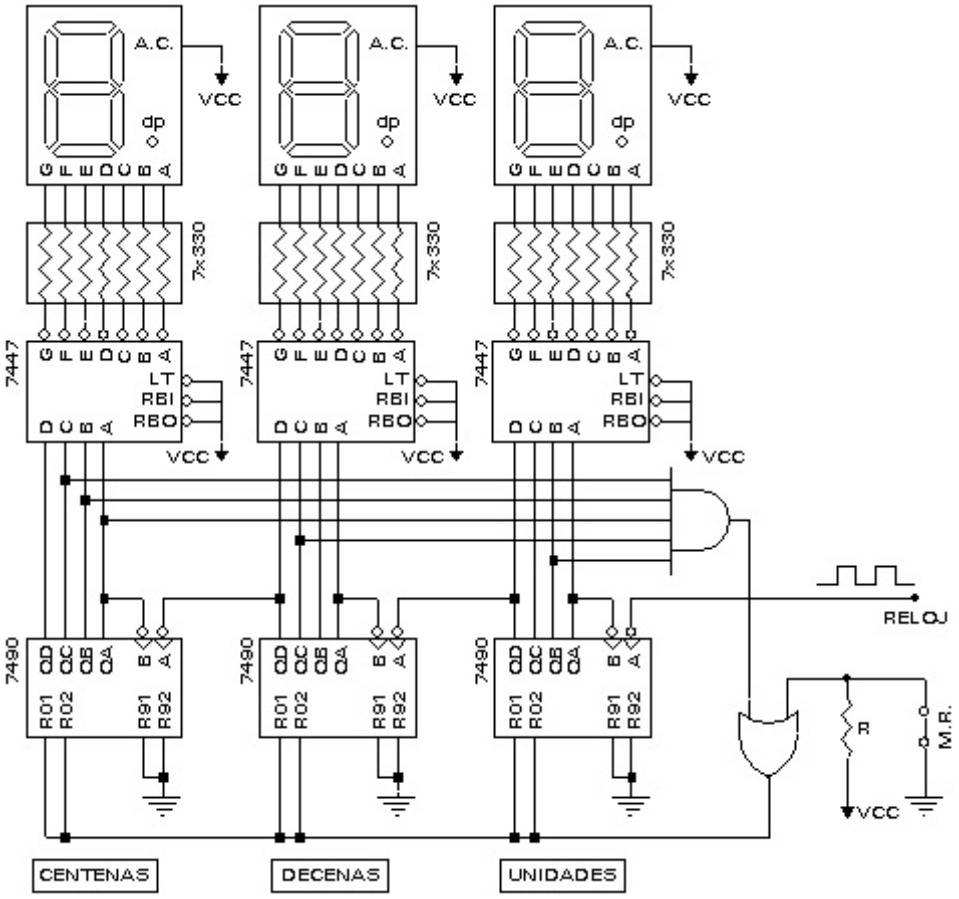


FIGURA 3.36 CONTADOR RIPPLE CLOCK MÓDULO 742

El contador MOD 742, se muestra en la fig. 3.36. Se requieren 3 contadores década y una compuerta AND para detectar el valor del módulo del contador [742_{10} = 0111

$0100\ 0010_{BCD}$], realimentando los 1s del valor BCD, como se hizo antes. La compuerta OR sirve para incluir un borrado manual, cuando el interruptor está cerrado, el contador cuenta normalmente y cuando está abierto, el contador, se borra.

Otros contadores ripple-clock son el CI-7492 [MOD = 12] y el CI-7493 [MOD = 16]. La distribución de pines de los estos integrados se muestra en la fig. 3.37 a) y 3.37 b) respectivamente. Las tablas de función se encuentran en los manuales de los fabricantes.

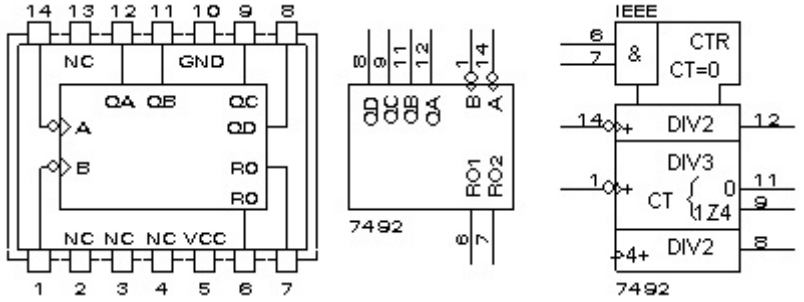


FIGURA 3.37 a) CI-7492

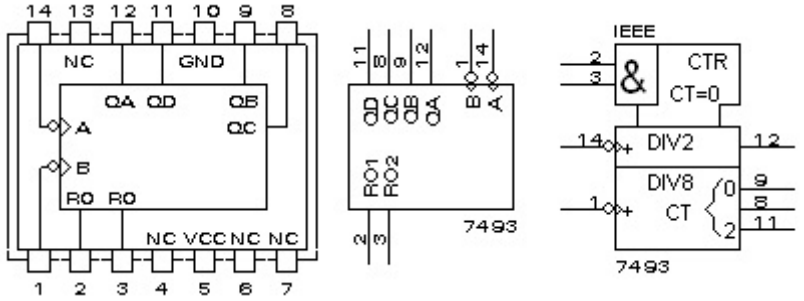


FIGURA 3.37 b) CI-7493

El circuito de la fig. 3.38 muestra el diagrama de bloques de un reloj digital de 24 horas. Se implementan dos contadores MOD-60, uno para los segundos y otro para los minutos, el contador MOD-24 es para las horas. Además se utilizan los decodificadores de BCD-a-7-SEG, las resistencias y finalmente los displays.

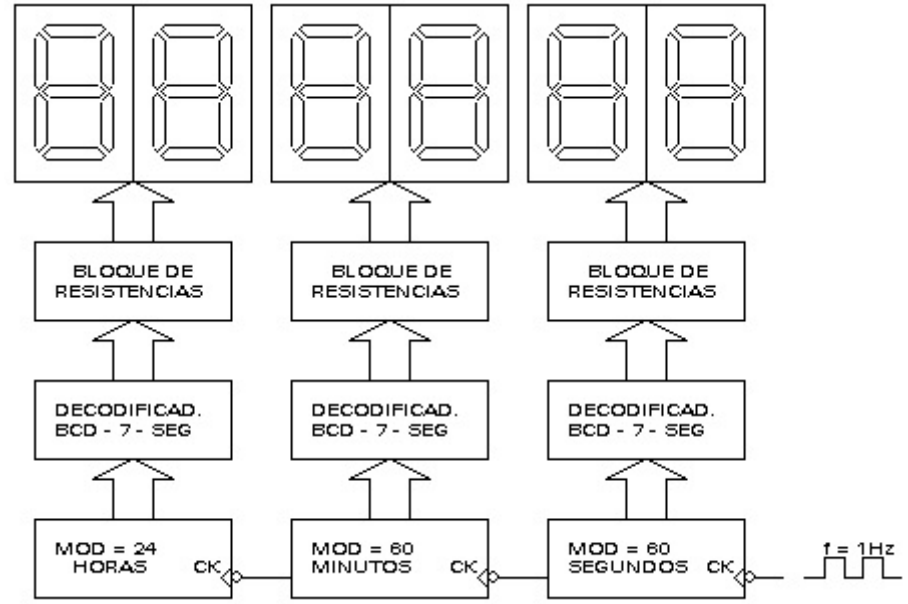


FIGURA 3.38 RELOJ DIGITAL DE 24 HORAS

El circuito de la fig. 3.39 es un contador módulo 60 implementado con un CI-7490 y un CI-7492, que podría utilizarse como contador de segundos o de minutos para el reloj digital. Se ha incluido la salida en displays de ánodo común y el sistema de igualación para minutos.

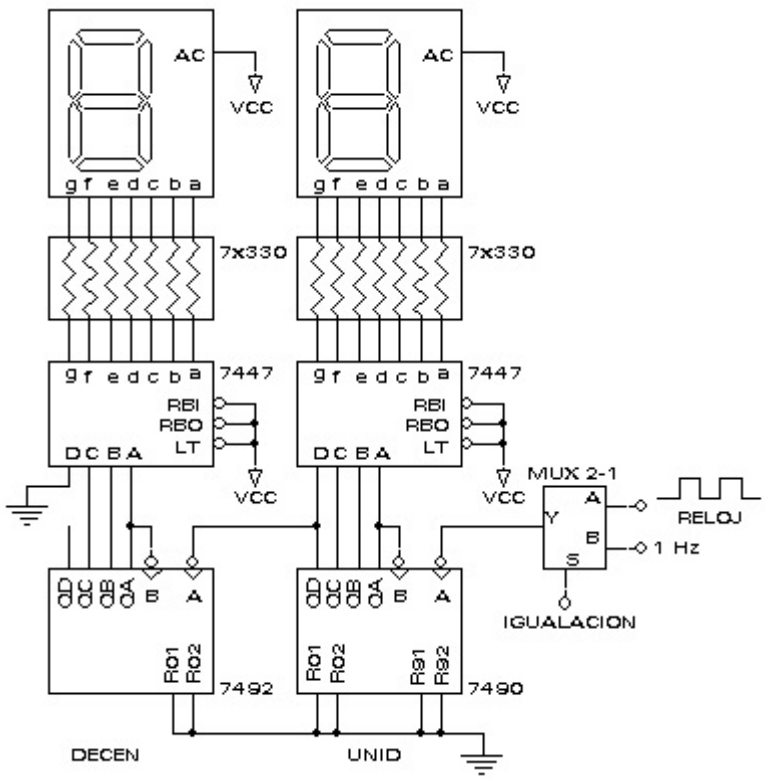


FIGURA 3.39 CONTADOR MÓDULO 60

Junio - 2008

Introducción al Análisis y Diseño de Circuitos Secuenciales Sincrónicos

Las redes secuenciales sincrónicas son circuitos digitales constituidos de una parte de lógica combinacional y de dispositivos para almacenamiento de información [FFs o memoria].

Estas redes pueden recibir señales de entrada y generar señales de salida que son funciones de las entradas actuales y del comportamiento anterior de la red. Pueden tomar una variedad de formas y generalmente se las clasifica en términos de su modo de operación y de la función que realizan.

En una red secuencial sincrónica el contenido de la información básica de los dispositivos de almacenamiento sólo puede cambiar durante la ocurrencia de un pulso de reloj. Entre pulsos de reloj, las operaciones lógicas se realizan con las señales de entrada y la información almacenada, pero no hay cambio en la información contenida en los dispositivos de almacenamiento de información.

Análisis de una Red Secuencial Sincrónica. – Cualquier red secuencial sincrónica puede representarse en la forma general que se muestra en la fig. 4.1.

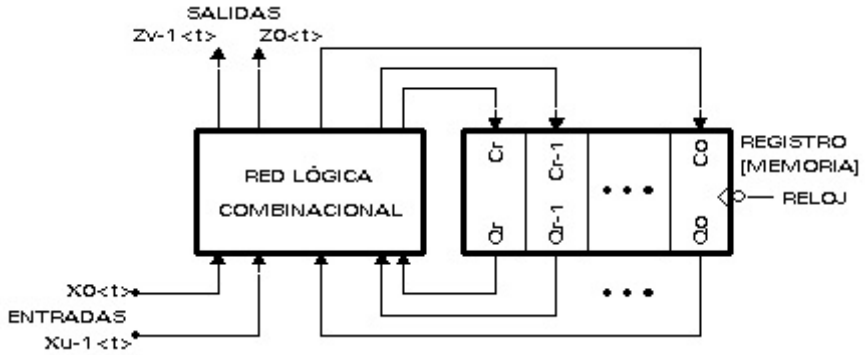


FIGURA 4.1 MODELO DE RED SECUENCIAL SINCRÓNICA

El registro está conformado con cualquier tipo de flip-flop y actúa como dispositivo interno para almacenamiento de información o memoria, que guarda la información de los eventos importantes de las entradas pasadas que influenciarán el comportamiento futuro de la red. La red lógica combinacional cumple dos funciones importantes: primero, en función de las señales de entrada $X_1<t>$, ..., $X_u<t>$ y de las salidas actuales de los flip-flops $Q_1<t>$, ..., $Q_r<t>$, genera las señales de comando necesarias para modificar el contenido del registro cuando se presenta el siguiente pulso de reloj. Segundo: la red lógica combinacional forma las salidas de la red, en función de las mismas variables[1].

Puesto que es una red sincrónica, el valor de todas las variables debe permanecer constante en el momento en que aparece el pulso de reloj. Después del pulso

1 "Digital Networks and Computer Systems" de Taylor L. Booth.

de reloj, el contenido del registro cambia a un nuevo valor que depende de la entrada de comando C_i al registro, en el momento en que ocurre el pulso de reloj. Igualmente las entradas pueden cambiar entre pulsos de reloj. Estos cambios hacen que las salidas de la red lógica combinacional cambien. El siguiente pulso de reloj puede ocurrir en cualquier momento después de que las salidas hayan alcanzado un estado estacionario.

Variables de Estado y Estados.- De aquí se ve que el estado actual de cada flip-flop, en el momento en que ocurre el pulso de reloj, influye en la salida actual de la red secuencial sincrónica y en el estado próximo de los flip-flops. Debido a esto, a las variables $Q_{r-1}, \ldots, Q_0$ se las conoce como ***variables de estado*** de la red secuencial sincrónica y a los distintos valores que toman las salidas Q de los flip-flops, tomadas en conjunto $[Q_{r-1}, \ldots, Q_0]$, constituye el ***estado*** de la red.

Un registro con 4 flip-flops, tendrá 4-variables de estado: Q_3, Q_2, Q_1 y Q_0; y 16-estados diferentes, desde [0,0,0,0] hasta [1,1,1,1]. Los distintos estados de la red corresponden a los diferentes ítemes de información que puede "***recordar***" la red.

El ***estado actual*** de una red secuencial sincrónica corresponde al estado [contenido] del registro un instante antes de que aparezca el pulso de reloj. Igualmente, se dice que las entradas $[X_{u-1}, \ldots, X_0]$,

las salidas $[Z_{v-1}, \ldots, Z_0]$ y las señales de control $[C_{r-1}, \ldots, C_0]$ al momento en que ocurre el pulso de reloj representan las señales de entrada actual, salida actual y control actual, respectivamente.

Cuando ocurre el pulso de reloj, el contenido del registro permanece constante hasta que se completa el pulso de reloj. En ese momento, el contenido del registro cambia a un nuevo valor. El nuevo valor se denomina ***estado próximo*** de la red.

Resumen

Variable de Estado	Cada una de las salidas de los FFs, Q_1<t>, ..., Q_n<t>, tomada separadamente es una variable de estado.
Estado	Cada una de las posibles combinaciones de las variables de estado tomadas en conjunto [Q_{n-1}<t>, ..., Q_0<t>] constituye el estado de la red.
Estado Actual	[O anterior] Es valor que tiene el registro antes del pulso de reloj.
Estado Próximo	Es el valor que tomará el registro después del pulso del reloj.

Puesto que se trata de una red sincrónica, los valores de las variables deben permanecer constantes durante el tiempo que dura el pulso [o la transición] del reloj.

Análisis de Máquinas de Estado Sincronizadas por Reloj.- "***Máquina de estado***" es un nombre genérico dado a estos circuitos secuenciales; por "reloj" se refiere al

hecho de que sus elementos de almacenamiento [flip-flops] emplean una entrada de reloj; y "sincronizados", debido a que todos los flip-flops utilizan la misma señal de reloj. La máquina de estado cambia de estado sólo cuando ocurre una transición o un "pulso" de disparo en la señal de reloj.

Estructura de la Máquina de Estado. - La fig. 4.2 muestra la estructura general de la máquina de estado denominada ***máquina de Mealy***. La memoria de estado es un conjunto de *n* flip-flops que almacena el estado presente de la máquina que tiene 2^n estados diferentes. Todos los flip-flops están conectados a una señal de reloj común que hace que los flip-flops cambien de estado con cada pulso de reloj. Lo que constituye una pulso depende del tipo de flip-flop (disparado por transición [↑ o ↓] o por pulso [M-S]).

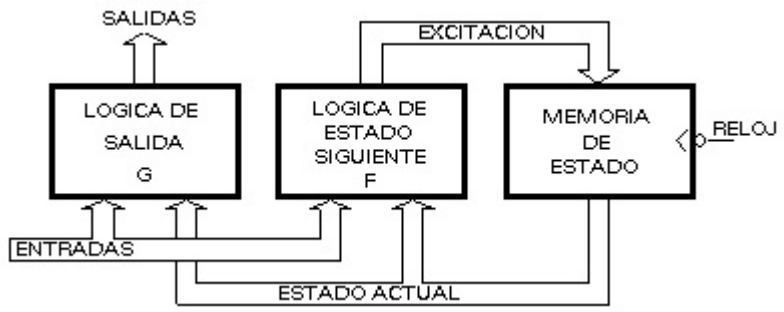


FIGURA 4.2 MÁQUINA DE MEALY

El estado siguiente de una máquina de estado, está determinado por la lógica de estado-siguiente F, como

una función del estado actual y de las entradas actuales. La salida G determina la salida del circuito en función del estado y de las entradas actuales. Tanto F como G son estrictamente circuitos lógicos combinacionales. Se puede escribir

Estado siguiente = F<Estado actual, entrada actual>
Salida actual = G<Estado actual, entrada actual>

Al circuito secuencial cuyas salidas dependen tanto del estado como de la entrada, como se estableció antes, se lo conoce como máquina de Mealy.

En algunos circuitos secuenciales, la salida sólo depende del estado actual.

Salida actual = G<Estado actual>

A tal circuito se lo conoce como ***máquina de Moore***, su estructura general se muestra en la fig. 4.3. La única diferencia entre los dos modelos de máquina de estados radica en cómo se generan las salidas.

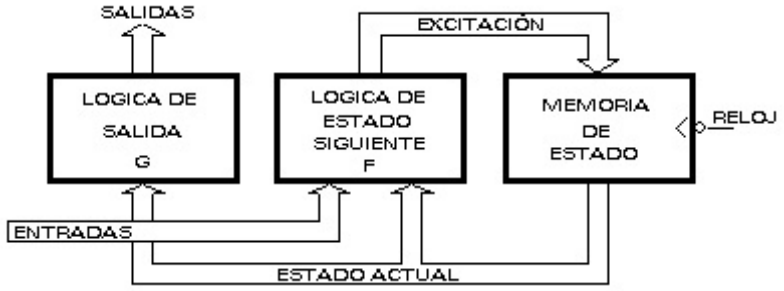


FIGURA 4.3 MÁQUINA DE MOORE

En la práctica, la mayoría de las máquinas de estado puede clasificarse como máquinas de Mealy, debido a tienen una o más salidas tipo Mealy que depende de la entrada y de su estado. Sin embargo, muchas de estas mismas máquinas tienen una o más salidas el tipo Moore, que dependen sólo del estado.

En la actualidad, la mayoría de las máquinas de estado se diseña con dispositivos lógicos programables con flip-flops D disparados con transición positiva. Sin embargo, se puede usar cualquier otro tipo de flip-flop, con transición positiva o negativa.

En el siguiente análisis es de extremada importancia tener en mente las relaciones concernientes al tiempo entre entrada-actual, salida-actual y estado-próximo. Para ilustrar estas relaciones, en el siguiente ejemplo se analiza una red secuencial simple.

Ejemplo 1.- Analizar la red lógica que se muestra en la fig. 4.4.

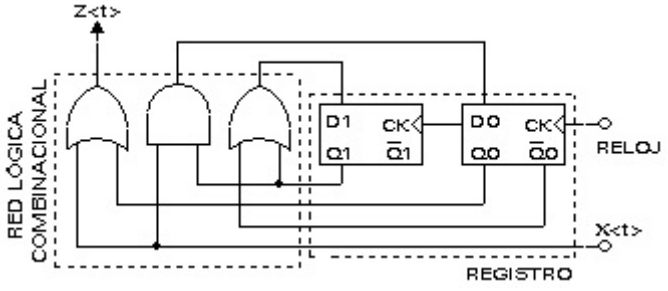


FIGURA 4.4

Se asumen los siguientes datos:

Entrada actual $X<t=0> = 1$

Estado actual $[Q_1<0>, Q_0<0>] = [0,0]$

De la red combinacional se tiene:

Salida actual $Z<t> = X<t> + Q_0<t>$

Control actual $D_0<t> = Q_1<t> \cdot X<t>$

$$D_0<t> = Q_1<t>\, Q_0<t>$$

De estas expresiones se ve que, para t = 0:

Entrada actual $X<0> = 1$

Estado actual $[Q_1<0>, Q_0<0>] = [0,0]$

Salida actual $Z<0> = 1$

Control actual $D_0<0> = 0$

$D_1<0> = 1$

Del comportamiento del FF tipo-D, se obtiene:

Estado próximo $[Q_1<1>, Q_0<1>] = [1,0]$

Ejemplo2.- Realizar el análisis completo del circuito secuencial sincrónico de la fig. 4.5.

De la red combinacional se obtienen las siguientes ecuaciones.

$$Z_{<t>} = X_{<t>}\, Q_{1<t>}\, \bar{Q}_{0<t>}$$

$$J_{1<t>} = \bar{X}_{<t>} + \bar{Q}_{1<t>}\, Q_{0<t>} \qquad K_{1<t>} = \bar{X}_{<t>}$$

$J_{0<t>} = X_{<t>}$ $K_{0<t>} = Q_{0<t>}$

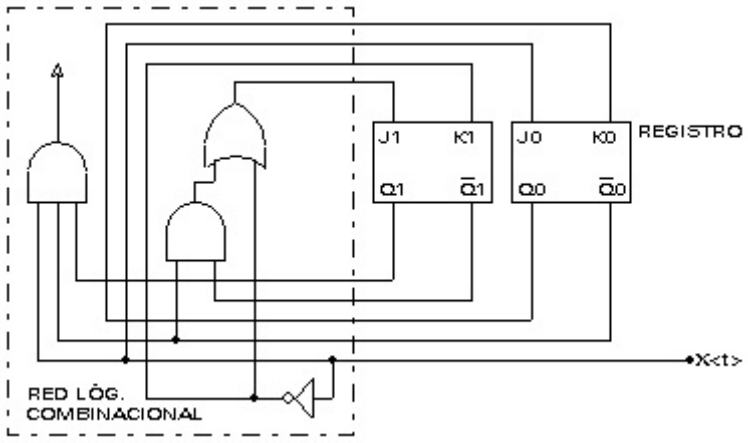


FIGURA 4.5

Estas ecuaciones booleanas pueden evaluarse para obtener la información que se muestra en la siguiente tabla, conocida como Tabla del comportamiento de la Red Secuencial Sincrónica.

ENT. ACT.	ESTADO ACTUAL		COMANDO ACTUAL				ESTADO PRÓXIMO		SAL. ACT.
X<t>	Q_1<t>	Q_0<t>	J_1<t>	K_1<t>	J_0<t>	K_0<t>	Q_1<t+1>	Q_0<t+1>	Z<t>
0	0	0	1	1	0	0	1	0	0
0	0	1	1	1	0	1	1	0	0
0	1	0	1	1	0	0	0	0	0
0	1	1	1	1	0	1	0	0	0
1	0	0	0	0	1	0	0	1	0
1	0	1	1	0	1	1	1	0	0
1	1	0	0	0	1	0	1	1	1
1	1	1	0	0	1	1	1	0	0

TABLA DEL COMPORTAMIENTO DE LA RED SECUENCIAL SINCRÓNICA

Aunque la información de la tabla describe el comportamiento de la red, a esta información generalmente se la presenta en una forma diferente, como una "***tabla de transición***".

Tabla de Transición y Diagrama de Estados.- El estado-próximo y la salida-actual pueden describirse y representarse como una representación tabular denominada ***tabla de transición*** o en forma gráfica como un ***diagrama de transición de estados***; ambas se estudian a continuación.

La representación como tabla de transición de una red secuencial, muestra las propiedades del estado-próximo y de la salida-actual en forma tabular. Las columnas de la tabla corresponden a las posibles señales de entrada y las filas corresponden a los posibles estados de la red. La entrada que se encuentra en la intersección de la fila-***k*** y de la columna-***j*** corresponde a

Estado-próximo / Salida-actual

Por ejemplo, la información presentada en la tabla anterior concerniente al estado-próximo y salida-actual de la red en estudio, puede representarse mediante la tabla de transición de estados indicada a continuación.

ESTADO ACTUAL $[Q_1,Q_o]$ \ ENTRADA ACTUAL X<t>	0	1
[0,0]	[1,0]/0	[0,1]/0
[0, 1]	[1,0]/0	[1,0]/0
[1, 0]	[0,0]/0	[1,1]/1
[1, 1]	[0,0]/0	[1,0]/0
	EST. PRX	SAL. ACT

TABLA DE TRANSICIÓN DE ESTADOS

Cuando se estudian redes secuenciales a menudo es más conveniente indicar el estado en forma simbólica. Por ejemplo, podemos denotar los distintos estados del ejemplo 2 como: A, B, C y D respectivamente. En general, la representación simbólica conviene más cuando se debe trabajar con redes que tienen un gran número de variables de estado. Esta representación también es útil para el diseño de circuitos secuenciales sincrónicos.

Asignación de Estados.- En el diseño de circuitos secuenciales sincrónicos, un aspecto muy importante es la asignación de estados. Es quizá la parte más difícil del diseño por cuanto hay que determinar el circuito más simplificado posible. A continuación, sólo para propósitos de información, se muestra el número de posibles asignaciones que pueden darse a un circuito secuencial sincrónico [T. L. Booth].

$$\frac{(2^r - 1)!}{(2^r - m)!\,r!} = \text{Asignaciones posibles}$$

donde: r = Número de variables de estado
m = Número de estados utilizados

En este ejemplo $r = 3$ y $m = 4$, por tanto se tienen 3 posibles asignaciones diferentes, de las que se toma la que se muestra en la siguiente tabla de asignación.

ESTADO $[Q_1,Q_o]$	ASIGNACIÓN
[0,0]	A
[0,1]	B
[1,0]	C
[1, 1]	D

Si en vez de utilizar valores lógicos para los estados que puede tomar la red, se hace la asignación de nombres indicada en la tabla anterior, se tiene la siguiente ***tabla de estados***.

ESTADO \ ENTRADA ACTUAL	0	1
A	C/0	B/0
B	C/0	C/0
C	A/0	D/1
D	A/0	C/0
	EST. PRX.	SAL. ACT.

TABLA DE ESTADOS

El problema con las tablas de transición y de estados es que a menudo es difícil visualizar el comportamiento de la red bajo condiciones de entrada diferentes. De ahí que convenga transformar esta información en un diagrama de estados. El **diagrama de estados proporciona una representación gráfica de la operación de la red secuencial.** Cada diagrama consiste de un conjunto de vértices etiquetados con el correspondiente estado de la red. Para cada par ordenado de estados [no necesariamente distintos], E_i y E_j, una línea conecta los vértices E_i a E_j; sí y sólo sí existe un valor, a_k, en las señales de entrada tal que

$$E_j = F_y<a_k, E_i>$$

Si una línea directa conecta E_i a E_j cuando la entrada es a_k, entonces a la línea se la etiqueta con

$$a_k/F_Z<a_k, E_i> <=> \text{Ent. actual/Sal. Actual}$$

Así los vértices del diagrama de estados corresponden al estado-actual de la red; la etiqueta indica la entrada-actual y la salida-actual. La cabeza de la flecha en cada línea indica el estado-próximo de la red, como se indica en la fig. 4.6.

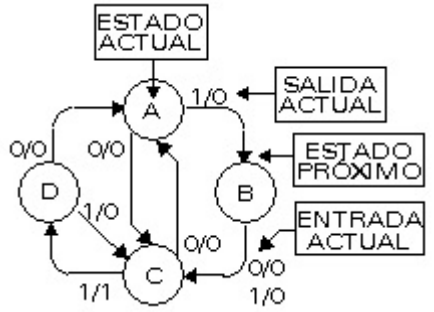


FIGURA 4.6 DIAGRAMA DE ESTADOS

Entrada, Salida y Secuencia de Estados. - Cuando se trabaja con una red secuencial que forma parte de un sistema complejo, generalmente lo que interesa es el comportamiento externo de la red. En particular, si se aplica una secuencia en la entrada.

X<O>, X<1>, X<2>, ..., X<k>

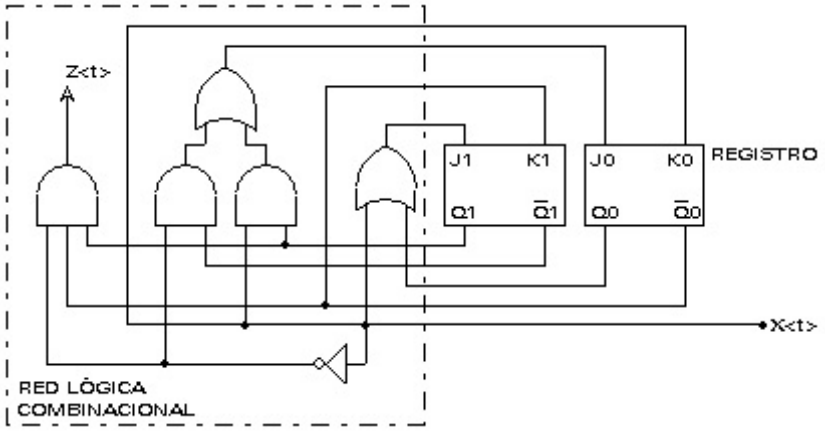
Se desearía conocer cuál será la secuencia de salida actual resultante.
Z<O>, Z<1>, Z<2>, ..., Z<k>

La respuesta a esta pregunta no es única puesto que depende del estado inicial de la red, al tiempo t = 0. Por ejemplo, considere la red secuencial en estudio y cuyo diagrama de estados se muestra en la fig. 4.6. Asuma que a la red se le aplica la secuencia de entrada presentada en la tabla que se indica a continuación; en ella se muestran los posibles valores

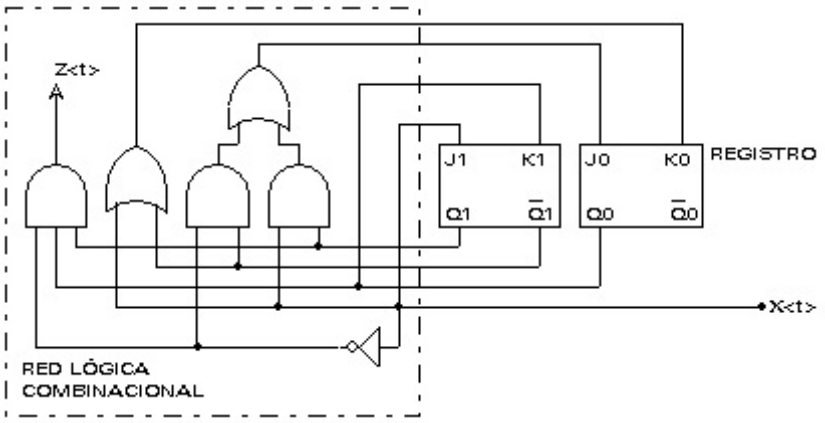
de X<t>, la consecuente salida actual y el estado próximo de la red.

t [pulsos]	0	1	2	3	4	5	6	7	8	9	10	11
X<t>	0	1	1	0	0	1	0	1	0	1	0	
Z<t>	0	1	0	0	0	1	0	0	0	1	0	
ESTADO	A	C	D	C	A	C	D	A	B	C	D	A

Si el estado inicial no fuera A, se obtendría una secuencia de estados diferente. De este análisis se ve que el problema de examinar el comportamiento de una red secuencial dada, puede manejarse de manera directa. Aunque las técnicas analíticas desarrolladas son de importancia en sistemas, esta discusión también ha servido para otro propósito importante: proporciona material necesario para resolver el problema de diseño de redes secuenciales sincrónicas para realizar un trabajo específico.



Problemas relacionados.-



Diseño de Redes Secuenciales Sincrónicas.- El problema de analizar el comportamiento de una red secuencial sincrónica se realiza fácilmente, siempre que se tenga un diagrama circuital de la red. El problema inverso, el de diseñar una red secuencial para que realice la operación de procesamiento de información, es considerablemente más difícil de resolver. En esta situación, se asume que se dispone de una descripción de la operación que debe realizar la red y se pregunta por el desarrollo de una red secuencial que realice estas operaciones.

Por ejemplo, se necesita diseñar una red secuencial que calcule el vuelto correcto que, una máquina de ventas que opera con monedas, debe entregar al cliente. En este caso, la entrada consiste de una secuencia de monedas depositadas en la máquina y la salida es un comando que retorna el vuelto específico que espera el cliente. Las redes secuenciales se usan

también como unidades de comando en varios sistemas digitales en los que se controla la secuencia en la que el sistema realizará una tarea de procesamiento de información.

El diseño de una red secuencial para ejecutar una operación dada es un arte y una ciencia. En varias etapas del proceso, el diseñador debe usar procedimientos heurísticos [forma de buscar una solución a un problema mediante métodos no rigurosos [por tanteo], reglas empíricas, etc.] y a la experiencia para tomar decisiones acerca de la mejor manera de proceder, mientras que en otras etapas se pueden emplear procedimientos algorítmicos directos para llevar a cabo los pasos de diseño asociados con esas etapas. Cada problema de diseño puede dividirse en las siguientes etapas.

Etapa 1.- Descripción de la operación deseada de la red .- Se debe preparar un conjunto completo de especificaciones que describan la operación de la red. Todas las entradas y salidas deben estar identificadas y la relación entre las cantidades debe definirse de manera consistente.

Etapa 2.- Determinación del diagrama de estados .- Usando las especificaciones establecidas en la etapa 1, se debe definir un diagrama de estados para la red. Debe chequearse el diagrama de estados para asegurarse que satisfaga todos los requisitos del problema.

Etapa 3.- Determinación de la tabla de estados .- El diagrama

de estados de la etapa 2 se lo transforma en una tabla de transición de estados.

Etapa 4.- Minimización de la tabla de estados . - En el proceso de desarrollo de un diagrama de estados, para satisfacer las necesidades del problema dado, puede introducirse un gran número de estados innecesarios. Puesto que el número de elementos de almacenaje de información en un circuito aumenta cuando aumenta el número de estados, es deseable eliminar de la tabla los estados redundantes.

Etapa 5.- Asignación de estados .- La información contenida en la tabla de estados debe codificarse en forma binaria. Este no es un proceso único y la codificación usada puede influir considerablemente en la complejidad del circuito resultante. El objetivo de esta etapa es transformar la tabla de estados en una tabla de transición.

Etapa 6.- Realización de la red . - Una vez obtenida la tabla de transiciones y tomada una decisión respecto de los FFs que se usarán, se pueden obtener las expresiones lógicas [ecuaciones booleanas] que relacionan el estado actual, la salida y las señales de comando.

En este proceso de diseño, sólo los pasos 3 y 5 pueden ejecutarse de manera completamente algorítmica. El problema de asignación de estados de la etapa 4 podría, en teoría, realizarse de manera algorítmica simplemente tratando todas las posibles asignaciones de estados y luego seleccionando la mejor de acuerdo

con algún criterio. Desafortunadamente el número de posibles asignaciones de estados es tan grande que esto es una aproximación irreal. Para ayudar a la solución de este problema se han desarrollado técnicas analíticas heurísticas y avanzadas.

Excepto en situaciones muy simples, las dos primeras etapas del proceso de diseño no pueden manejarse de manera completamente algorítmica. Conforme el diseñador gana experiencia aprende un conjunto de procedimientos heurísticos. Afortunadamente el proceso inicial de aprendizaje necesario para desarrollar un conjunto útil de heurísticas puede obtenerse resolviendo 3 o 4 problemas típicos. Para comprender mejor estos conceptos se realizarán algunos ejemplos, empezando con los contadores sincrónicos y luego se harán un diseños más genéricos.

Contadores Sincrónicos.- Todos los FFs que conforman este tipo de contador están conectados a una señal de reloj única [común a todos los FFs].

Ejemplo.- Diseñar un contador sincrónico módulo 10.

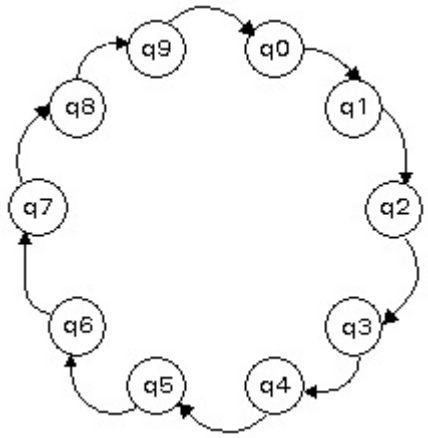


FIGURA 4.7

El diseño se empieza con el diagrama de estados que se muestra en la fig. 4.7.

ESTADO ACTUAL	ESTADO PRÓXIMO
q0	q1
q1	q2
q2	q3
q3	q4
q4	q5
q5	q6
q6	q7
q7	q8
q8	q9
q9	q0

ESTADO	ASIGNACIÓN			
	Q3	Q2	Q1	Q0
q0	0	0	0	0
q1	0	0	0	1
q2	0	0	1	0
q3	0	0	1	1
q4	0	1	0	0
q5	0	1	0	1
q6	0	1	1	0
q7	0	1	1	1
q8	1	0	0	0
q9	1	0	0	1

Luego se obtiene la tabla de estados, a continuación se realiza la asignación de estados, posteriormente

la tabla de transición de estados y finalmente se implementa la tabla del comportamiento del contador pedido, en la que se indica la cuenta actual y la cuenta próxima.

ESTADO ACTUAL				ESTADO PRÓXIMO			
Q3<t>	Q2<t>	Q1<t>	Q0<t>	Q3<t+1>	Q2<t+1>	Q1<t+1>	Q0<t+1>
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0

La tabla del comportamiento de la red secuencial sincrónica se muestra a continuación. Los valores de las variables de comando-actual [J y K] de cada FF, se obtienen en base al estado-actual, el estado próximo y la tabla de excitación del FF; los estados próximos que no se utilizan se los reemplaza por condiciones "no importa", por ejemplo los estados desde el 10 [1010] hasta el 15 [1111], no son necesarios en el contador MOD = 10, por tanto en las columnas de estado próximo se las llena con condiciones "no importa".

CNT.	Estado Actual <t>				Estado Próximo <t+1>				Comando Actual <t>							
	Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
1	0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
2	0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
3	0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
4	0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
5	0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
6	0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
7	0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
8	1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
9	1	0	0	1	0	0	0	0	X	1	0	X	0	X	X	1
10	1	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X

Mediante el uso de mapas-K, se obtienen las siguientes funciones booleanas simplificadas para las variables de comando de cada FF-JK.

$$J_3 = Q_2 Q_1 Q_0 \qquad K_3 = Q_0$$

$$J_2 = Q_1 Q_0 \qquad K_2 = Q_1 Q_0$$

$$J_1 = \overline{Q}_3 Q_0 \qquad K_1 = Q_0$$

$$J_0 = 1 \qquad K_0 = 1$$

El circuito del contador sincrónico MOD-10, con

FFs J-K, y sus formas de onda se muestra en la fig. 4.8. Las salidas de los flip flops [$Q_3Q_2Q_1Q_0$] corresponden con las salidas del contador.

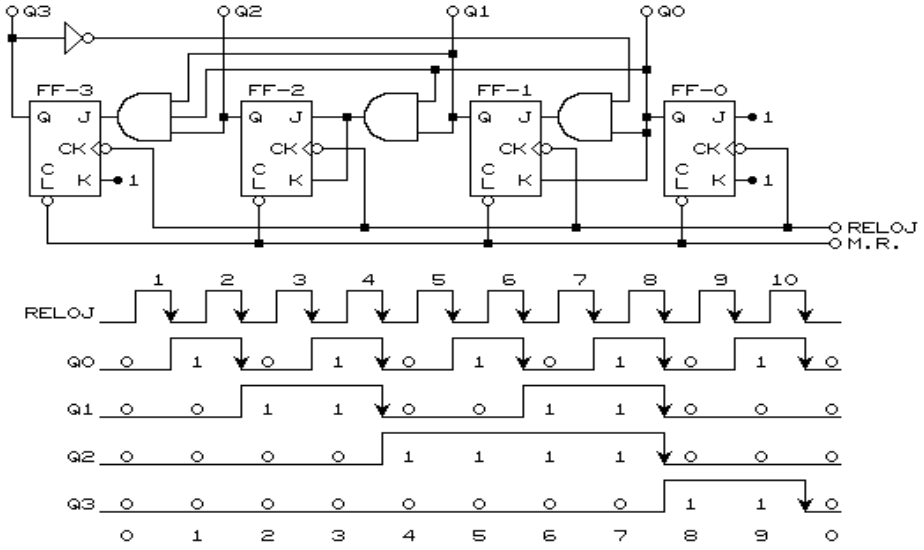


FIGURA 4.8 CONTADOR SINCRÓNICO MÓDULO 10

Contador Sincrónico Up-Down.- Son dispositivos que permiten realizar la cuenta en forma ascendente o descendente mediante una señal de comando $\overline{U}/D$ externa [que por facilidad se llamará X<t>]. Normalmente se acepta que con $\overline{U}/D$ = 0, el contador cuente ascendentemente y con $\overline{U}/D$ = 1, el contador cuente descendentemente.

Ejemplo.- Diseñar un contador sincrónico $\overline{U}/D$ módulo 7.

Lo primero que se obtiene es un diagrama de estados

[fig. 4.9] que facilite visualizar en qué condición cuanta ascendentemente y cuando cuenta descendentemente, también servirá para obtener la tabla de estados para luego obtener una tabla de función que cumpla este propósito.

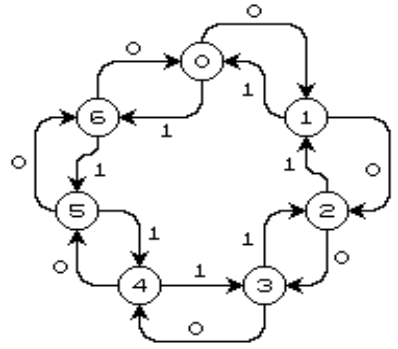


FIGURA 4.9

Cuenta Actual \ X<t>	0	1
0	1	6
1	2	0
2	3	1
3	4	2
4	5	3
5	6	4
6	0	5

Cuenta DEC	Cuenta BIN		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
	Q_2	Q_1	Q_0

Esta tabla tiene una entrada de comando de cuenta X<t> [$\overline{U}/D$], y el estado actual, constituido por las

salidas de cada uno de los FFs que conforman el contador al tiempo <t>, en este caso se requieren 3-FFs tipo JK. Así mismo, debe incluir el estado próximo, constituido por las tres salidas de los FFs al tiempo <t+1>.

De esta información, pueden obtenerse las ecuaciones de comando para los FFs, las que se muestran a continuación.

Ent. Act.	Estado Actual <t>			Estado Próximo <t+1>			Comando Actual <t>					
X<t>	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	X	0	X	1	X
0	0	0	1	0	1	0	0	X	1	X	X	1
0	0	1	0	0	1	1	0	X	X	0	1	X
0	0	1	1	1	0	0	1	X	X	1	X	1
0	1	0	0	1	0	1	X	0	0	X	1	X
0	1	0	1	1	1	0	X	0	1	X	X	1
0	1	1	0	0	0	0	X	1	X	1	0	X
0	1	1	1	X	X	X	X	X	X	X	X	X
1	0	0	0	1	1	0	1	X	1	X	0	X
1	0	0	1	0	0	0	0	X	0	X	X	1
1	0	1	0	0	0	1	0	X	X	1	1	X
1	0	1	1	0	1	0	0	X	X	0	X	1
1	1	0	0	0	1	1	X	1	1	X	1	X
1	1	0	1	1	0	0	X	0	0	X	X	1
1	1	1	0	1	0	1	X	0	X	1	1	X
1	1	1	1	X	X	X	X	X	X	X	X	X

$$J_2 = Q_0Q_1\bar{X} + \bar{Q}_0\bar{Q}_1X \qquad J_1 = Q_0 \oplus X$$
$$K_2 = \bar{Q}_0\bar{Q}_1X + Q_1\bar{X} \qquad K_1 = Q_0 \oplus X + Q_2$$

$$J_0 = \bar{Q}_2\bar{X} + Q_2\bar{Q}_1 + Q_1X$$
$$K_0 = 1$$

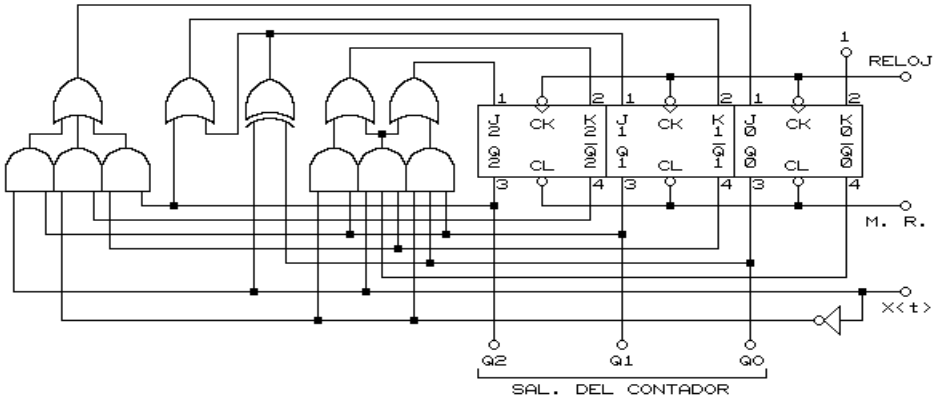


FIGURA 4.10 CONTADOR SINCRÓNICO U/D MÓDULO 7

El circuito correspondiente se muestra en la fig. 4.10, en ella se ha incluido una entrada M. R. conectada al Clear de los FFs, para borrar el contador [inicializar con ceros].

Ejemplo.- Diseñar un contador $\bar{U}/D$ sincrónico módulo 10.

El diagrama de estados se indica en la fig. 4.11. El resto del proceso es igual al caso anterior.

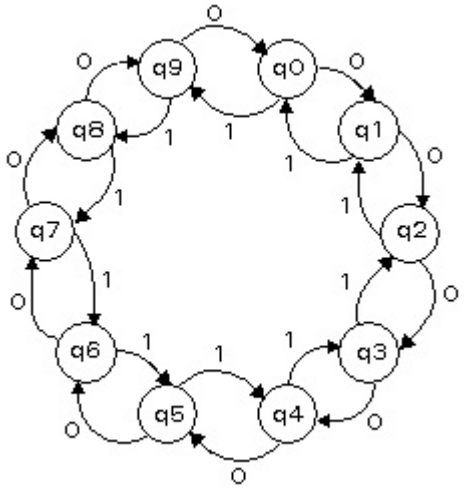


FIGURA 4.11

Contadores Sincrónicos Programables.- En muchas ocasiones es necesario iniciar la cuenta desde un valor conocido diferente de 0, para esto se implementan los contadores programables que disponen de entradas adicionales que permiten cargar [Load = LD] un valor en el contador desde el que continuará la cuenta con cada pulso de reloj. Para esto es necesario que los FFs, que forman el contador, dispongan de las **entradas asincrónicas clear** y **preset.** La fig. 4.11 muestra las compuertas NAND y las conexiones que se deben realizar para obtener un contador con entradas programables [en paralelo] y la entrada Load, que se activa con nivel bajo, es decir, cuando la entrada LD = 0, la salida Q correspondiente toma el valor de su entrada de datos, y cuando LD = 1, el contador cuenta normalmente con

cada pulso de reloj.

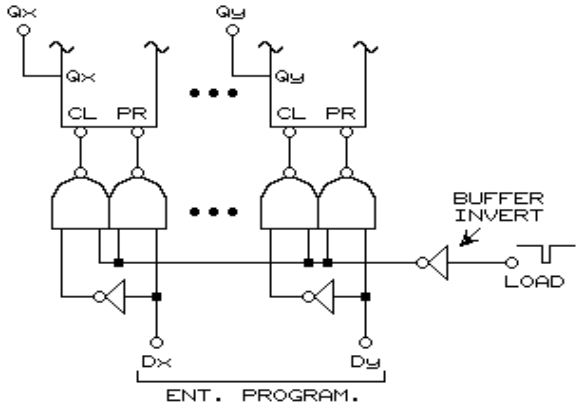


FIGURA 4.11

En el mercado existen varios contadores $\overline{U}/D$ sincrónicos programables. Entre ellos se encuentra la serie de los CI-74190; '191; '192 y '193, la distribución de pines de estos integrados se muestra en la fig. 4.12.

CI-74190 ['191] [fig. 4.12] es un contador sincrónico reversible [$\overline{U}/D$] y programable que tiene una complejidad equivalente a 58 compuertas. El '190 es un contador BCD y el '191 es un contador binario de 4-bits. Los 4-FFs que conforman estos contadores están conectados a una señal de reloj, de modo que las salidas cambian simultáneamente con la señal de reloj. Este modo de operación elimina los picos espurios que se generan en las salidas de los contadores asincrónicos tipo ripple-clock. La fig. 4.13 muestra

la temporización del CI-74190 indica las formas de onda del CI-74190, que proporciona el fabricante para que el usuario pueda realizar sus diseños.

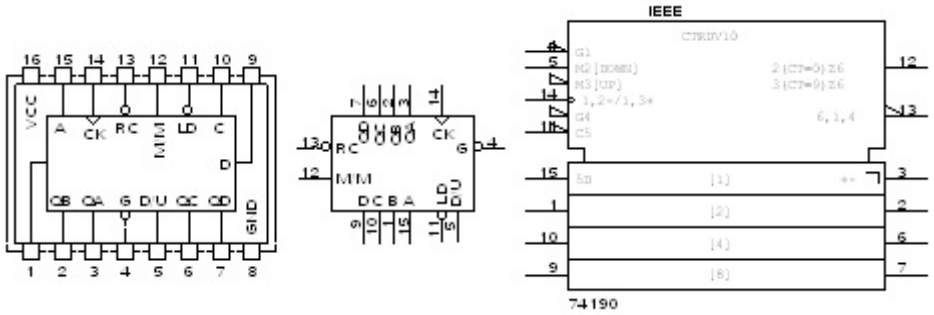


FIGURA 4.12 CI-74190/192 CONTADOR U/D SINCRÓNICO PROGRAMABLE

Estos contadores son totalmente programables; esto es, poniendo el valor deseado en las entradas de datos programables [D, C, B, A] y un valor bajo [0_L] en la entrada Load [carga], se puede inicializar el contador con cualquier valor deseado. Las salidas tomarán el valor de las entradas independientemente del nivel de la entrada de reloj. Esta característica permite que los contadores puedan usarse como divisores MOD-N simplemente modificando la longitud de la cuenta con las entradas programables.

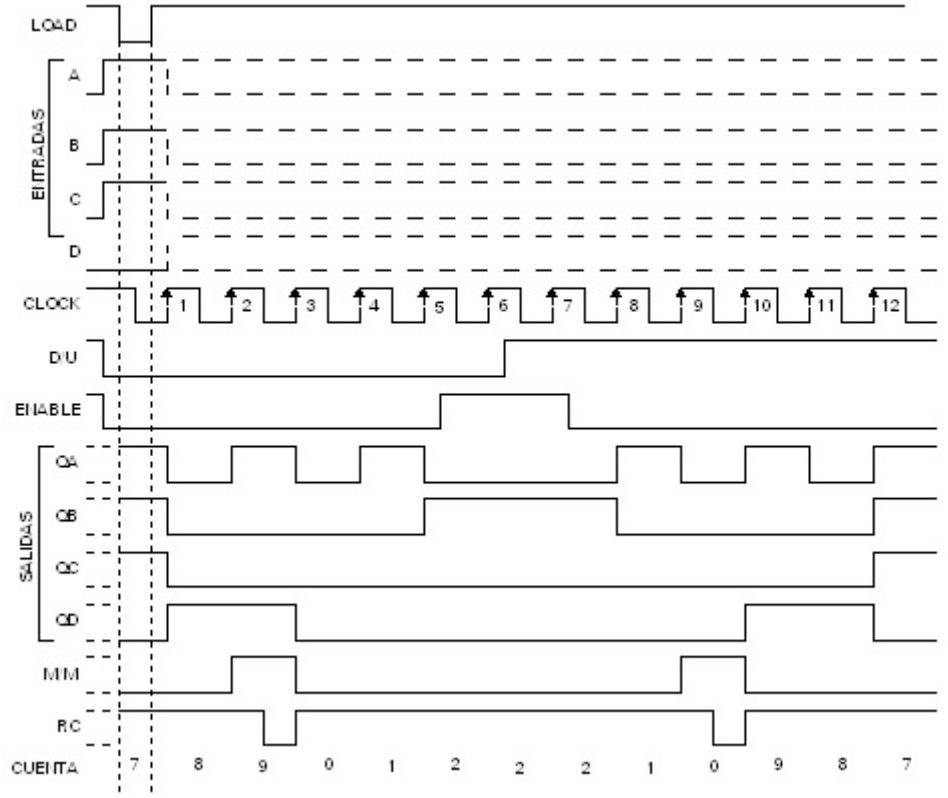


FIGURA 4.13 TEMPORIZACIÓN DEL CI-74190

Las entradas de reloj, $\overline{U}/D$ y carga [Load], disponen de un buffer de entrada para disminuir la carga a las señales de entrada respectivas.

Para poder conectar en cascada, se tienen dos salidas: ripple-clock [R.C.] y máximo/mínimo [M/M]. Esta última produce un nivel-alto con una duración aproximadamente igual a un ciclo completo de reloj cuando el contador sobrepasa la cuenta ascendente o descendente. La salida ripple-clock produce un pulso

de nivel bajo, igual en ancho al nivel-bajo de la entrada de reloj, cuando el contador sobrepasa la cuenta ascendente o descendente.

Los contadores fácilmente puede conectarse en cascada uniendo la salida ripple-clock con la entrada de habilitación [G] del siguiente contador, en este caso es necesario que la señal de reloj se conecte en forma común a cada uno de los CIs [funcionamiento sincrónico], o a la entrada de reloj si las entradas de habilitación se conectan en paralelo. La salida max/min se puede usar para realizar operaciones de alta velocidad look-ahead [adelanto].

La fig. 4.14 muestra un contador $\overline{U}/D$, sincrónico, programable MOD-100, para lo que se utilizan dos CI-74190.

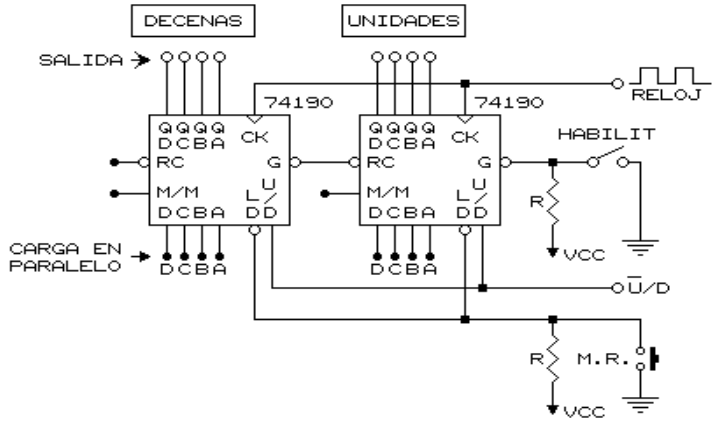


FIGURA 4.14 CONTADOR U/D MÓDULO 100

La fig. 4.14 muestra un contador $\overline{U}/D$, sincrónico,

programable MOD-85_{10}, lo que se ha hecho es modificar el contador MOD-100_{10} de la sección anterior. En primera instancia, cuando el contador cuenta ascendentemente, se debe generar un pulso de nivel-bajo cuando la cuenta llega al valor 85_{10} [1000 0101_{BCD}] para la entrada de carga y en las entradas de datos poner el número 0000 0000_{BCD}.

De igual manera, cuando el contador cuenta descendentemente se debe detectar el número 99_{10} [1001 1001_{BCD}] para generar un pulso de nivel-bajo para la entrada load, en esa situación, en las entradas de datos debe estar presenta el número 84_{10} [1000 0100_{BCD}], por tanto, A, B y D del contador de unidades y las entradas A, B y C del contador de decenas se conectan a 0, mientras que las restantes se conectan a la línea U/D que es 0, cuando cuenta hacia arriba y 1 cuando cuenta hacia abajo. La fig. 4.15 muestra el circuito completo y las conexiones que deben realizarse para que cumpla la función deseada. En la entrada de habilitación [G] del contador de unidades se ha colocado una circuitería constituida por una resistencia y un interruptor que sirve para habilitar la cuenta cuando el interruptor está cerrado y para que inhabilite la cuenta cuando el interruptor está abierto.

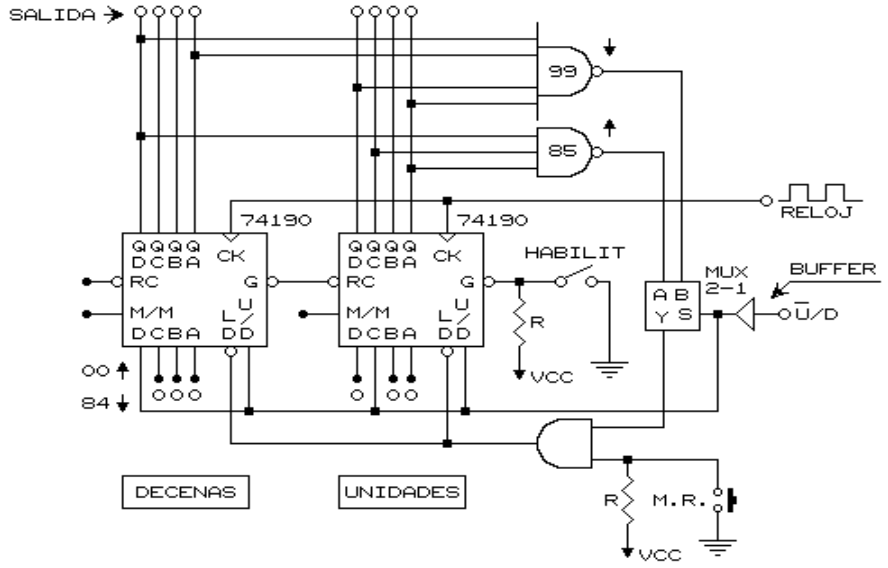


FIGURA 4.15 CONTADOR U/D MÓDULO 85

Otro ejemplo, un contador módulo 1000, se muestra en la fig. 4.16. En este caso se requieren 3 contadores 74190.

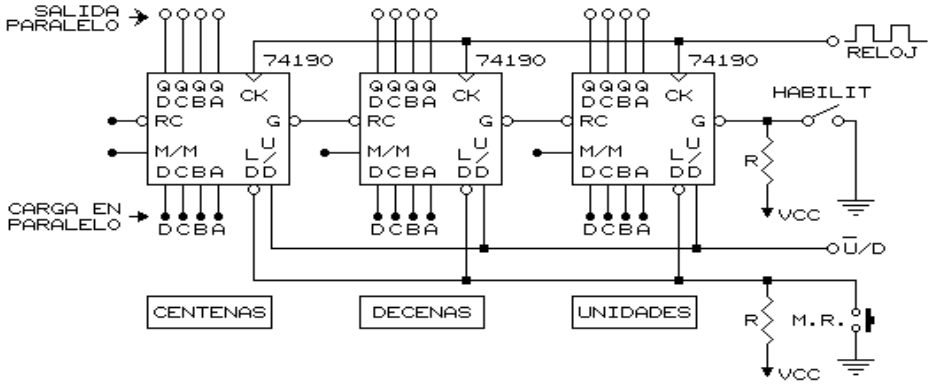


FIGURA 4.16 CONTADOR U/D SINCRÓNICO MOD-1000

La entrada de habilitación del contador de unidades está comandada por un circuito que permite poner 0 cuando se quiere habilitar la cuenta o 1 cuando se la quiere inhabilitar. Así mismo, las entradas LD están comandadas por un interruptor pulsante para reiniciar el contador con el valor de las entradas en paralelo [d, c, b, a] de cada CI.

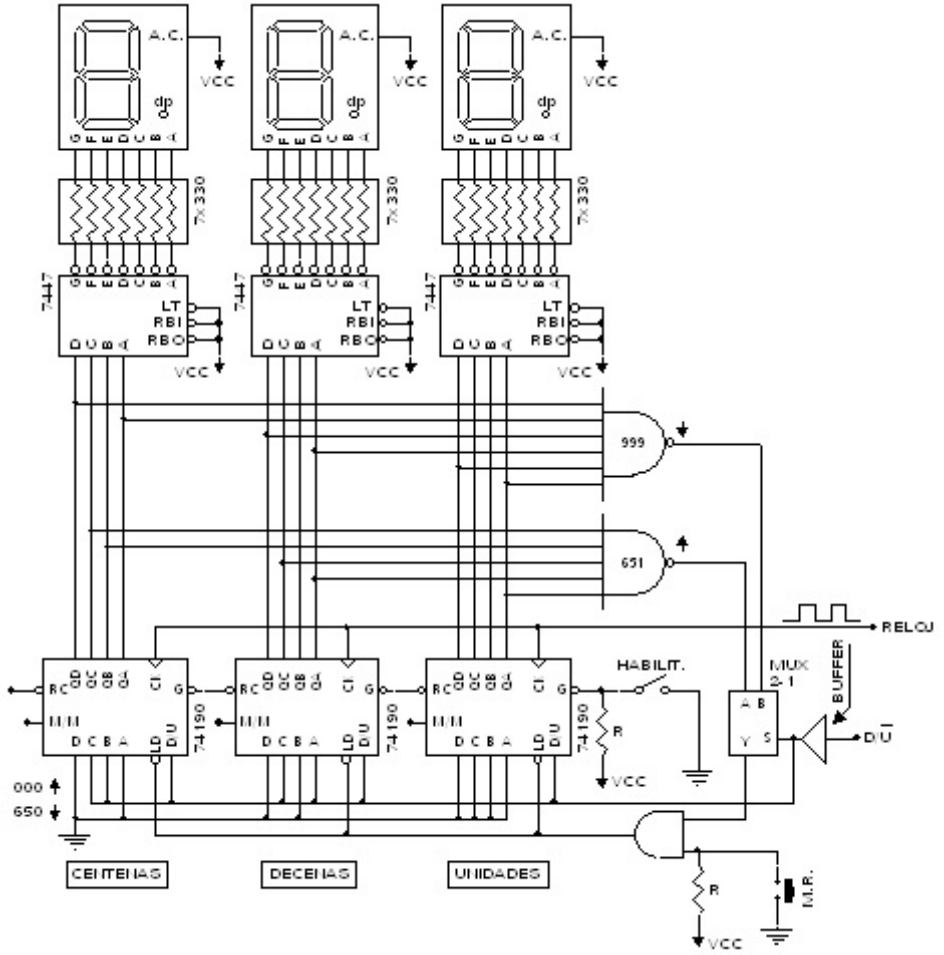


FIGURA 4.17 CONTADOR U/D SINCRÓNICO MÓDULO 651

Una modificación del contador $\overline{U}/D$ MOD-1000 se muestra en la fig. 4.17 [contador módulo 651]. Las compuertas NAND sirven para detectar el valor del módulo del contador. La compuerta AND se la utiliza para incluir un borrado manual o Master-Reset.

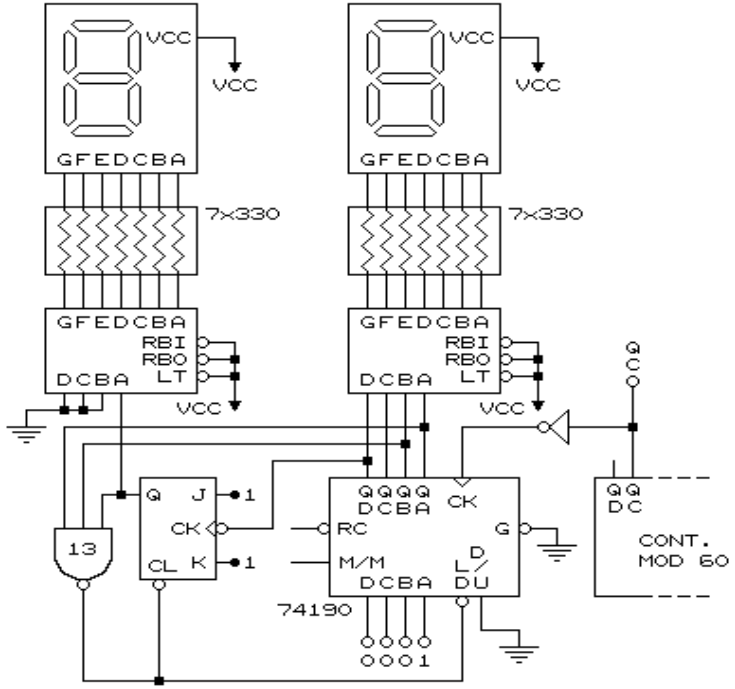


FIGURA 4.18 CONTADOR MOD = 12

El circuito de la fig. 4.18 muestra un contador MOD = 12, en el que la cuenta va desde 1 hasta 12 inclusive y que serviría para el contador de horas del reloj digital que se diseñó anteriormente. El CI-74190 cuenta las unidades de horas mientras que el FF-JK se utiliza para contar las decenas de horas.

La compuerta NAND sirve para detectar el valor 13, y generar un 0, para borrar el FF y para inicializar al CI-74190 con 0001 y volver a empezar la cuenta desde 1 y no desde 0. El inversor sirve para sincronizar el disparo del CI-74190 que lo hace con transiciones positivas.

CI-74192/193.- [Fig. 4.19]. Estos CIs son contadores $\overline{U}/D$ sincrónicos programables. El CI-74192 es contador década, mientras que el CI-74193 es hexadecimal. La diferencia con los CIs 74190/191 es que estos tienen una entrada $\overline{U}/D$ común, mientras que los CI-74192/193 tiene una entrada U y una entrada D, independientes. Cuando los pulsos ingresan por la entrada U, entonces la entrada D debe permanecer en nivel alto y viceversa. Las formas de onda para los CIs 74192/193 está disponible en los manuales TTL.

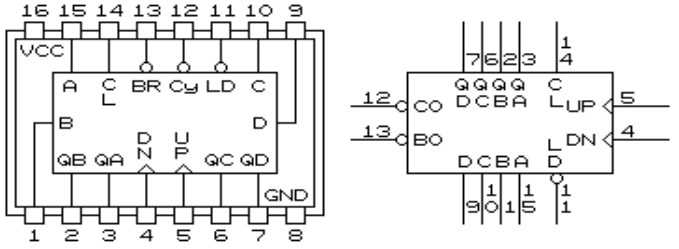


FIGURA 4.19 CI-74192/193

Contadores Sincrónicos 74160, '161, '162 y '163.- Estos contadores sincrónicos, programables, tienen la característica de carry-adelantado [**Carry look-ahead**]

interno para aplicaciones de diseño de cantadores de alta velocidad. Los CI-74160 y 74162 son contadores década y los CI-74161 y 74163 [fig. 4.20] son contadores binarios de 4-bits. En la operación sincrónica, los 4 flip-flops están conectados a la misma señal de reloj, de modo que las salidas cambian simultáneamente cuando las entradas P y T habilitan la cuenta. Este modo de operación elimina los picos en las salidas de conteo asociados normalmente con los contadores asincrónicos [ripple-clock].

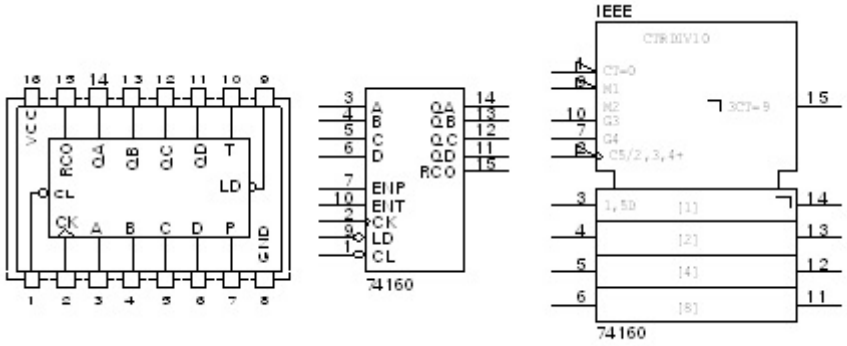


FIGURA 4.20 CI-74160/'161/'162/'163

Esta serie 74160 de contadores se dispara con las transiciones positivas de la señal de reloj. La fig. 4.20 muestra la distribución de pines del la serie de CI74160/161/162/163.

Estos contadores son totalmente programables; es decir, las salidas pueden inicializarse con cualquier valor. Las entradas de datos son sincrónicas, así

que un nivel bajo en la entrada de carga [Load], deshabilita al contador y hace que las salidas coincidan con los datos de entrada después de la siguiente transición positiva del pulso de reloj, sin importar los niveles de las entradas de habilitación.

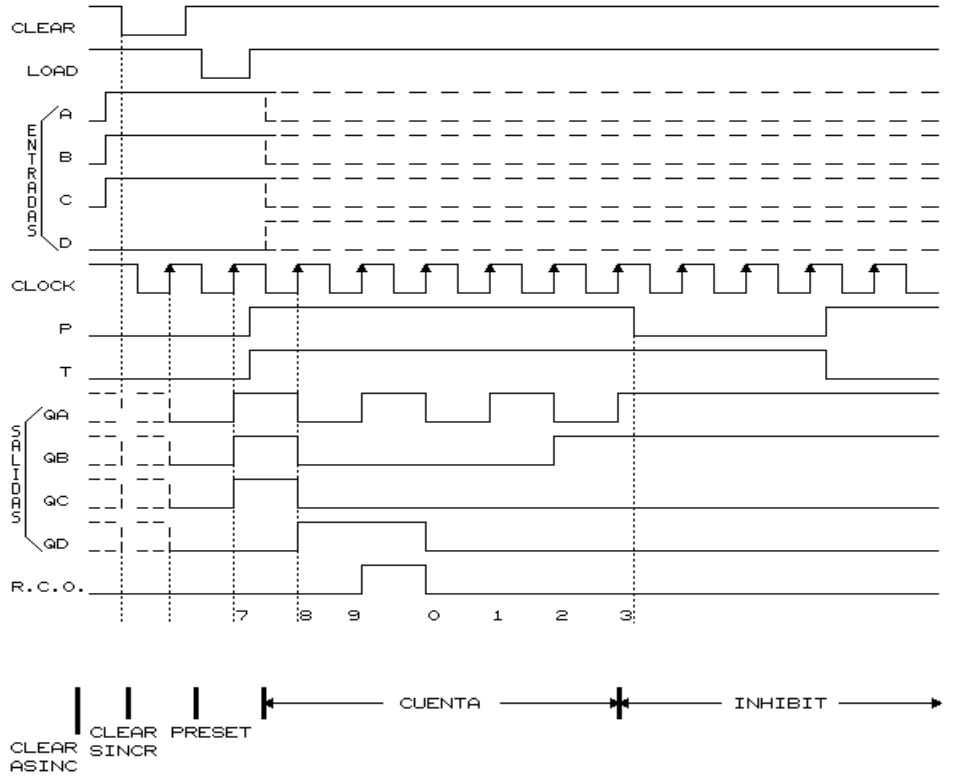


FIGURA 4.21TEMPORIZACIÓN DE LOS CI-74160/162 - CONTADORES DÉCADA SINCRÓNICOS PROGRAMABLES. [Clear Asincrónico/Sincrónico y Load Sincrónico].

En los contadores 74160 hasta 74163, se debe evitar una transición positiva en la señal de reloj si las

entradas de habilitación están en nivel alto en o antes de la transición. La función clear para los '160 y '161 es asincrónica y un nivel bajo en esta entrada pone un cero en la salida de los 4 flip-flops, sin importar el nivel de la señal de reloj o de las entradas de habilitación.

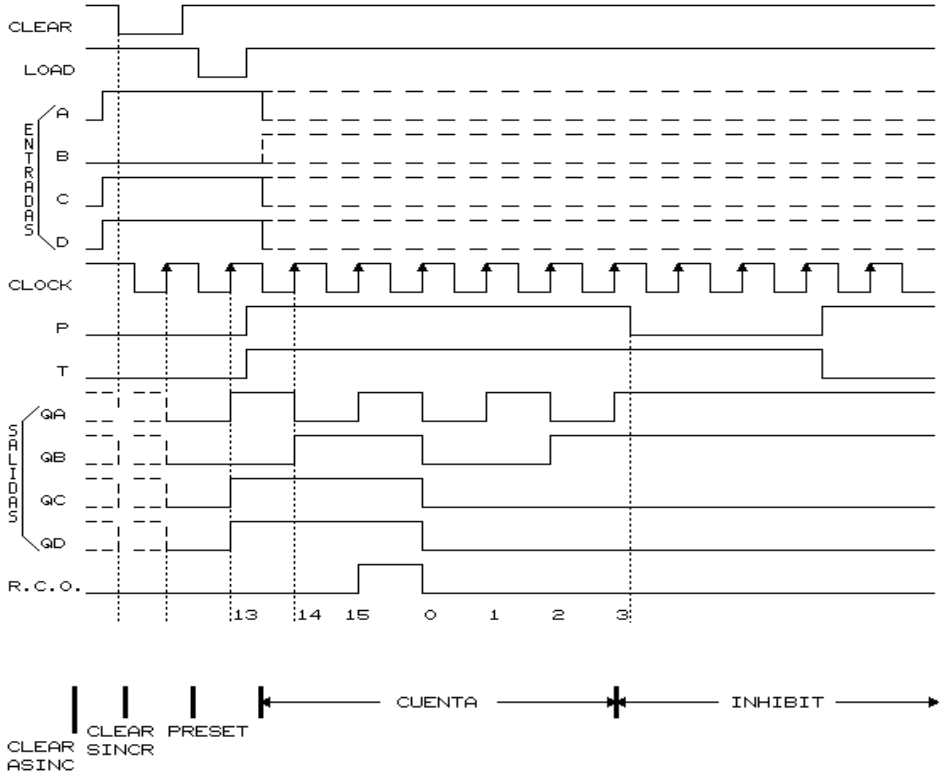


FIGURA 4.22 TEMPORIZACIÓN DE LOS CI-74161/163 - CONTADORES BINARIOS SINCRÓNICOS PROGRAMABLES. [Clear Asincrónico/Sincrónico y Load Sincrónico].

La función clear para los '162 y '163 es sincrónica y un nivel bajo en esta entrada pone un cero en la

salida de los 4 flip-flops después del siguiente pulso de reloj, sin importar el nivel de las entradas de habilitación. Este borrado sincrónico permite que se pueda modificar la longitud de la cuenta. La fig. 4.21 muestra la temporización de los CI-74160/162 - contadores década sincrónicos programables, [Clear Asincrónico/Sincrónico y Load Sincrónico]. La fig 4.22 muestra la temporización de los CI-74161/163 - contadores binarios sincrónicos programables. [Clear Asincrónico/Sincrónico y Load Sincrónico].

Registros de Desplazamiento [Shift-Register].- Los Registros de Desplazamiento [R. D.] son circuitos secuenciales sincrónicos en los que los FFs se conectan de tal manera que cuando se aplica una transición activa a la entrada del reloj [común a todos ellos], la información presente en la entrada I<t> ingresa al primer FF, [FF-0], la información previa de éste pasa al segundo [FF-1], la del segundo al tercero [FF-2], y así sucesivamente. En otras palabras, la información que ingresa por la entrada I<t> se desplaza bi-a-bit [serialmente] hacia el R. D. y la información almacenada en cada biestable se desplaza hacia el siguiente FF con cada transición activa del reloj.

Existen R. D. que desplazan la información de derecha-a-izquierda, como los que se muestran en las figs. 4.23 y 4.24, así mismo se pueden implementar R. D. con desplazamiento de izquierda-a-derecha o

inclusive bidireccionales, que mediante una señal de dirección podrán desplazar la información en un sentido o en otro. Los R. D. tienen muchísimas aplicaciones prácticas en relación a los computadores y en otros sistemas digitales. De ahí la importancia de su estudio. En el mercado existen algunos tipos de R. D. en CI.

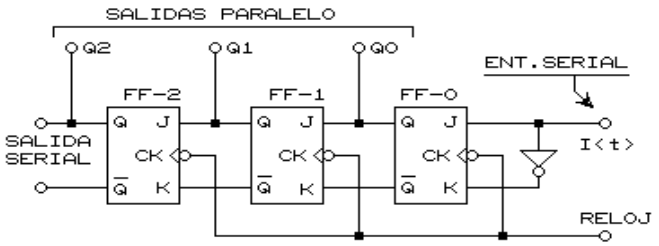


FIGURA 4.23 REGISTRO DE DESPLAZAMIENTO

Los R. D. pueden implementarse conectado en cascada FFs tipo KJ [como el circuito de la fig. 4.33] o con FFs tipo D. La fig. 4.24 muestra un registro de desplazamiento de 3-bits [3-FFs] implementado don FFs tipo-D, y las formas de onda de la señal de reloj, de la entrada serial de datos y de las salidas de cada uno de los FFs. Puede observarse que las formas de onda en las salidas de los FFs, son básicamente iguales a la señal de entrada I<t>, solamente desplazadas en el tiempo, un período de reloj por cada FF. De manera que el R. D. puede usarse como un dispositivo para generar retardos de tiempo.

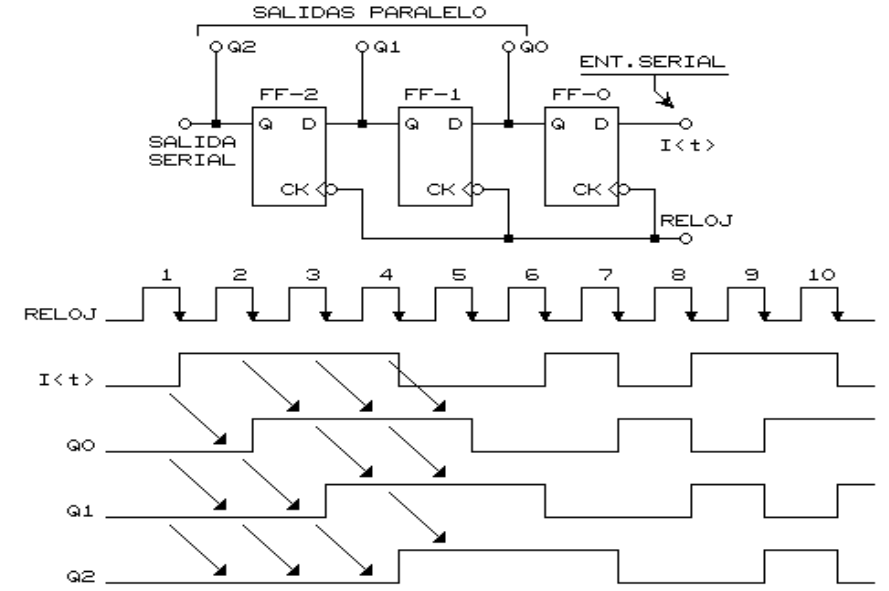


FIGURA 4.24

Registro de Desplazamiento Bidireccional.- La fig. 4.25 muestra un R. D. que, en base a una señal de control de dirección [DIR], puede desplazar la información de derecha-a-izquierda [cuando DIR = 0] o de izquierda-a-derecha [cuando DIR = 1], a este tipo de R. D. se lo define como bidireccional.

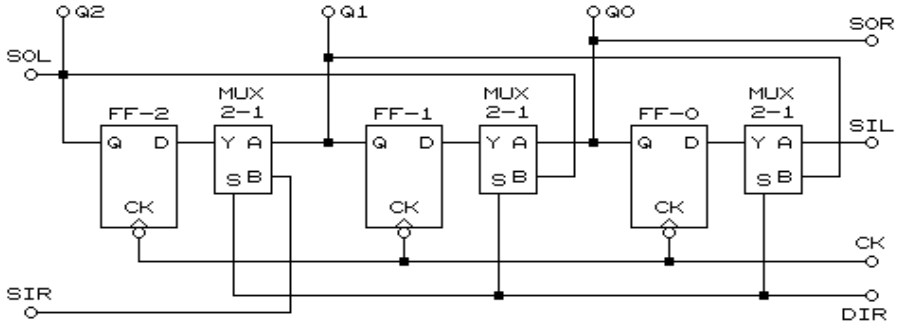


FIGURA 4.25 R. D. BIDIRECCIONAL

En la fig. 4.25: SOR = Serial-Output-Right; SOL = Serial-Output-Left; SIR = Serial-Input-Right; SIL = Serial-Input-Left.

Registro de Desplazamiento con Carga de Datos en Paralelo.- El circuito de la fig. 4.26 permite ingresar datos en forma paralela, [similar al caso de los contadores programables]. Se lo conoce con el nombre de convertidor paralelo-serie porque los datos que ingresan en forma paralela, se los puede obtener en la salida serial [SO]. Se puede observar que también admiten el ingreso de datos desde la entrada serial [Serial Input = SI].

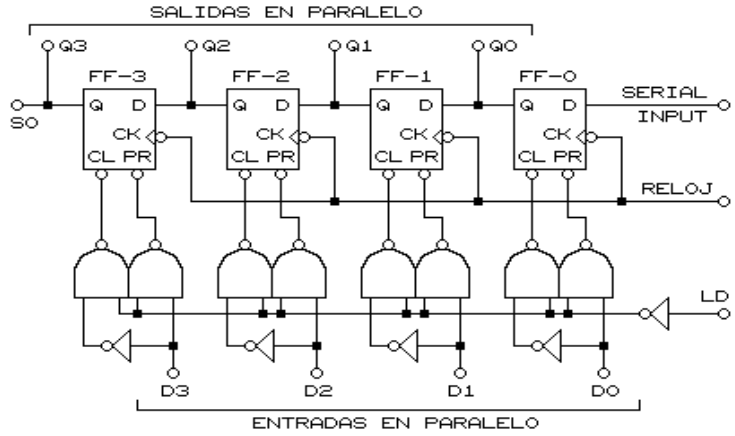


FIGURA 4.26 R. D. CON CARGA PARALELA

CI-74195.- Es un R. D. de acceso en paralelo de 4-bits. El desplazamiento es de QA hacia QD. La entrada de datos en serie es a través de J y $\overline{K}$.

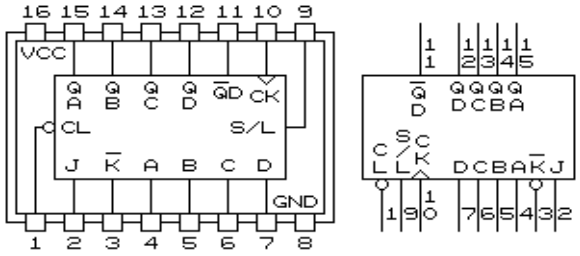


FIGURA 4.27 CI-74195

La carga en paralelo se realiza poniendo los 4-bits de datos y colocando la entrada de carga/desplazamiento [S/L] en 0. La fig. 4.27 muestra la distribución de pines de este CI. A continuación se presenta la tabla de función del CI-74195.

ENTRADAS									SALIDAS				
			Serial		Paralelo								
CL	Shift/Load	CK	J	$\overline{K}$	D	C	B	A	$\overline{Q}_D$	QD	QC	QB	QA
0	X	X	X	X	X	X	X	X	1	0	0	0	0
1	0	↑	X	X	d	c	b	a	$\overline{d}$	d	c	b	a
1	1	0	X	X	X	X	X	X	$\overline{Q}_{Dn}$	Q_{D0}	Q_{C0}	Q_{B0}	Q_{A0}
1	1	↑	0	1	X	X	X	X	$\overline{Q}_{Cn}$	Q_{Cn}	Q_{Bn}	Q_{A0}	Q_{A0}
1	1	↑	0	0	X	X	X	X	$\overline{Q}_{Cn}$	Q_{Cn}	Q_{Bn}	Q_{An}	0
1	1	↑	1	1	X	X	X	X	$\overline{Q}_{Cn}$	Q_{Cn}	Q_{Bn}	Q_{An}	1
1	1	↑	1	0	X	X	X	X	$\overline{Q}_{Cn}$	Q_{Cn}	Q_{Bn}	Q_{An}	$\overline{Q}_{An}$

Registro de Desplazamiento Universal.- El circuito de la fig. 3.60 es un R. D. universal, porque tiene todas las características: Entrada serial, salida serial,

entrada en paralelo, salida en paralelo de datos y además es bidireccional. Existen algunos CIs de este tipo en el comercio.

CI-74194.- [Fig. 4.28]. Este R. D. de 4-bits, está diseñado para incorporar virtualmente todas las características que puede necesitar un diseñador de sistemas.

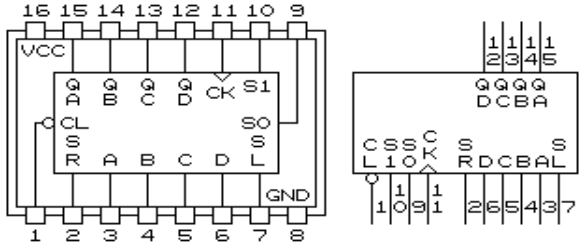


FIGURA 4.28 CI-74194

ENTRADAS										SALIDAS				
	Modo			Serial		Paralelo								
CL	S1	S0	CK	L	R	D	C	B	A	QD	QC	QB	QA	
0	X	X	X	X	X	X	X	X	X	0	0	0	0	Borra
1	X	X	0	X	X	X	X	X	X	QD_o	QC_o	QB_o	QA_o	Mantiene los datos
1	1	1	↑	X	X	d	c	b	a	d	c	b	a	Carga en paralelo
1	0	1	↑	X	1	X	X	X	X	QC_n	QB_n	QA_n	1	Desplaza a la izquierda
1	0	1	↑	X	0	X	X	X	X	QC_n	QB_n	QA_n	0	
1	1	0	↑	1	X	X	X	X	X	1	QD_n	QC_n	QB_n	Desplaza a la derecha

1	1	0	↑	0	X	X	X	X	X	0	QD_n	QC_n	QB_n	
1	0	0	X	X	X	X	X	X	X	X	QD_o	QC_o	QB_o	QA_o	Mantiene los datos

X = No importa [irrelevante]

↑ = Transición de bajo a alto [Transición Positiva]

d, c, b, a = nivel de la entrada en estado estacionario en las entradas D, C, B, A, respectivamente

QD_o, QC_o, QB_o, QA_o = El nivel de QD, QC, QB, o QA, respectivamente, antes de que se estabilicen las condiciones indicadas de estado estacionario de las entradas

QDn, QCn, QBn, QAn = El nivel de QD, QC, QB, o QA, respectivamente, antes de la más reciente transición positiva [↑] del reloj.

Tiene cuatro modos de operación distintos.

- Carga paralela
- Desplazamiento a la derecha [en la dirección desde QA hacia QD].
- Desplazamiento a la izquierda [en la dirección desde QD hacia QA].
- Reloj deshabilitado [no hace nada]

A continuación se presenta la tabla de función del CI-74194.

Convertidores con Registros de Desplazamiento.- A los R. D. se los puede usar como convertidores. Existen cuatro tipos de convertidores: Entrada Serial-Salida Serial [SI-SO]; Entrada Serial-Salida Paralelo [SI-PO]; Entrada Paralela-Salida Serial [PI-SO] y Entrada Paralela-Salida Paralela [PI-PO].

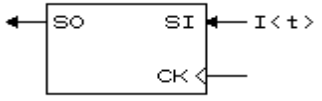
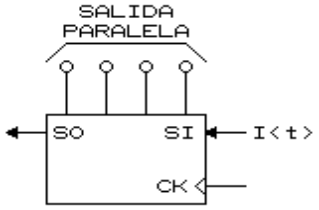


FIGURA 4.29

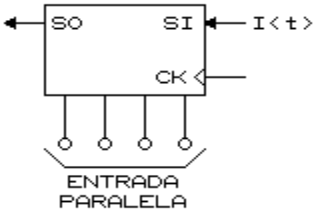
Entrada Serial-Salida Serial [Serial Input - Serial Ouput =

SI-SO]



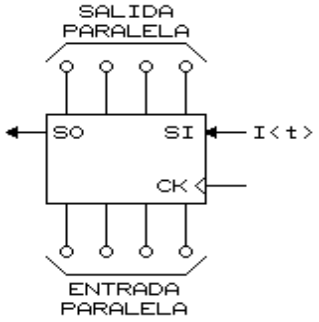
Entrada Serial-Salida Paralela [Serial Input - Parallel Output = SI-PO]

FIGURA 4.30



Entrada Paralela-Salida Serial [Parallel Input - Serial Output = PI-SO]

FIGURA 4.31



Entrada Paralela-Salida Paralela [Parallel Input - Parallel Output = PI-PO]

FIGURA 4.32

Registro de Desplazamiento Circular [Ring Counter].- En los registros de desplazamiento estudiados no se puede guardar la información porque la del último FF se pierde. Si se conecta esta salida a la entrada I<t>,

entonces la información almacenada en el R. D. circulará con cada pulso de reloj y la información ya no se pierde.

Para poder ingresar nueva información en este tipo de R.D., se debe incluir un MUX 2-1 en la entrada I<t>, como se muestra en la fig. 4.33. Donde DES = desplaza [S = 0] y N. I. ingresa nueva información [S = 1].

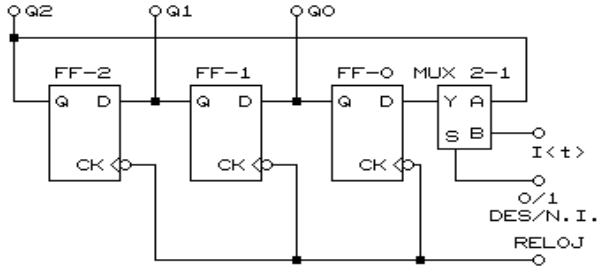


FIGURA 4.33

Contador Johnson.- La fig. 4.34 muestra un cantador Johnson. Los contadores Johnson [también conocidos como de anillo-torcido o Möebius] difieren de los contadores de anillo en que la realimentación se la realiza con $\overline{Q}$ de la última etapa. El resultado es un contador con 2N estados [donde N es el número de bits [FFS] del registro de desplazamiento].

Si el registro empieza en 000, los siguientes estados que se presentan en este tipo de registro son los que se muestran en la tabla adjunta al gráfico.

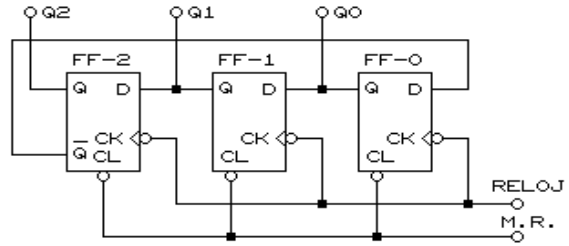


FIGURA 4.34 CONTADOR JOHNSON

Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0
0	0	0

La fig. 4.35 muestra el diagrama de bloques, muy simplificado, de una C. P. U. [Central Processing Unit = Unidad Central de Procesamiento], se muestran solamente los registros más importantes, la complejidad interna de la C. P. U. depende del circuito real.

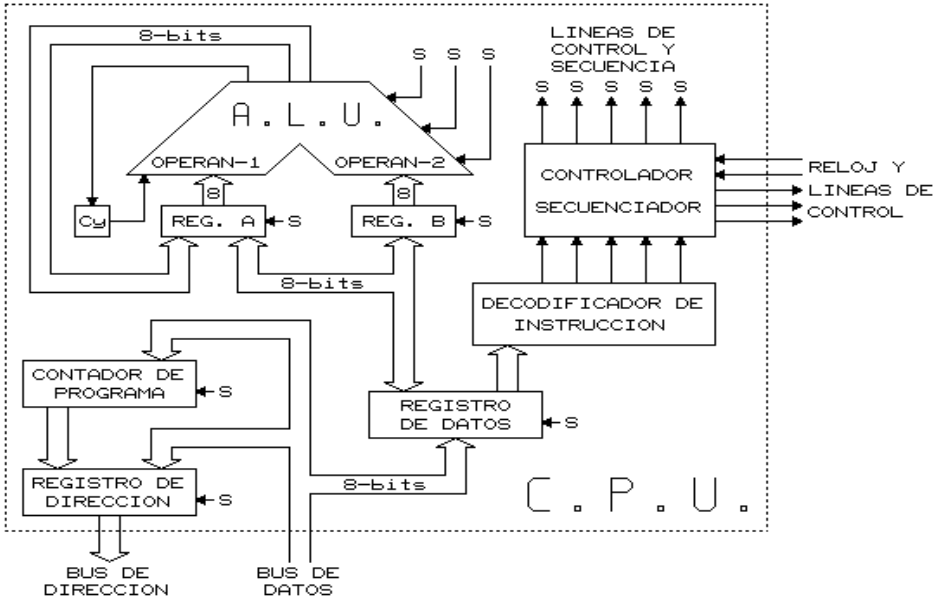


FIGURA 4.35 C. P. U. BÁSICO

Se la ha incluido aquí para mostrar el uso de varios de los dispositivos digitales que se han estudiado hasta este momento. Entre otros puede observarse un Registro de Datos que está constituido por un grupo de FFs [retenedores] que guardan información temporal que puede ser el código de una instrucción o un dato que irá al Registro A o al Registro B. La A. L. U. [Unidad Aritmética y Lógica], que realiza operaciones aritméticas y lógicas entre dos operandos [Registro-A y Registro-B] de 8-bits cada uno [en este ejemplo], también existe un Registro adicional de 1-bit denominado Carry [Cy] para almacenar el exceso que puede generarse en una operación aritmética. El Decodificador de Instrucciones, que como su nombre indica, su función es decodificar el valor binario de sus entradas y, entonces, decidir que tipo de operación deberá realizar la C. P. U. esta información pasa a un circuito Controlador-Secuenciador que decide la secuencia en la que se ejecutará la operación decodificada, qué circuitos se activarán y en qué momento. También dispone de un Contador de Programa [Program Counter] que tiene la característica de ser programable y su función es la de permitir que las instrucciones se ejecuten una a continuación de otra, sin embargo, permite cargar un nuevo valor en el momento que se necesite realizar un salto a otra localidad de la memoria. Finalmente, el Registro de Dirección sirve para indicar la dirección de memoria

desde donde se sacará la siguiente instrucción o un dato.

Ejemplo 3.- Utilice flip-flops tipo-JK para diseñar un circuito secuencial sincrónico que permita detectar la secuencia de bits que se muestra en la siguiente tabla. Una vez terminada la detección, la salida Z<t> debe tomar el valor 1 y el circuito debe regresar al estado inicial para empezar una nueva detección; en cualquier otro caso, Z<t> debe ser 0. Incluir una entrada de inicialización manual o M.R.

t	0	1	2	3	4	5
X<t>	0	1	1	0	0	1

Siempre se empieza con un estado inicial que indica que todavía no ha llegado el primer dato válido de la secuencia pedida [o que se ha llegado a este estado mediante la entrada M-R]. A este estado se lo llama A, fig. 4.35, [en vez de A se puede usar cualquier otro nombre adecuado, por ejemplo q_0, etc.].

FIGURA 4.35

A partir del estado inicial, se analizan todos los posibles valores que pueden ingresar a través de las variables de entrada, en este caso X<t>, que

solo puede tomar los valores 0 o 1. En primer lugar se asume que llega el valor 0, que corresponde al primer valor de la secuencia pedida, entonces se crea un nuevo estado [B], fig. 4.36. [Cada vez que llegue un dato correcto de la secuencia se crea un nuevo estado].

Para pasar del estado A al estado B es necesario que X = 0.

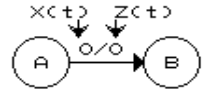


FIGURA 4.36

Si el circuito está en el estado A y llega el valor 1, que no corresponde al primer dato de la secuencia, entonces el circuito todavía se mantiene en el estado inicial [fig. 4.37], hasta que llegue un dato válido. Con esto se ha terminado de analizar todos los posibles valores de la variable de entrada, desde el estado inicial.

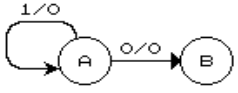


FIGURA 4.37

Ahora se hace el mismo análisis pero desde el estado B. El estado B recuerda que ha llegado el primer dato válido de la secuencia pedida, en este caso 0. Si, estando en B, el próximo valor de X es

0, entonces el próximo estado será B mismo [fig. 4.38], recuerde que el estado B significa que ha llegado el primer valor de la secuencia 0, en este ejemplo.

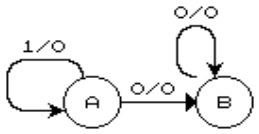


FIGURA 4.38

Si el estado es B y el próximo valor de X es 1, que corresponde al siguiente dato de la secuencia, se crea el estado C [fig. 4.39] que indica que ha llegado el segundo dato consecutivo de la secuencia deseada. Con esto se termina el análisis desde el estado B.

Para llegar al estado C, es necesario que, a través de X hayan llegado los valores 0 y 1, en forma consecutiva y en ese orden.

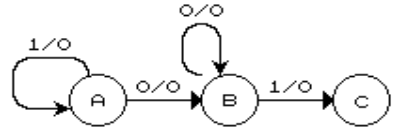


FIGURA 4.39

Si el estado actual es C [que quiere decir que ha llegado 01] y el valor actual de X es 0, la secuencia se rompe, pero se puede ver que el último

valor de X es 0 y a su vez este dato corresponde al estado B, entonces el circuito regresa al estado B [fig. 4.40]. Para saber a qué estado se debe regresar, conviene realizar el siguiente análisis; se comparan los últimos valores que han llegado a través de X con un número igual de los primeros bits de la secuencia deseada, por ejemplo, estando en C, si llega 0, entonces se tiene 010 y se compara con 011, se ve que no son iguales; luego se toman los dos últimos valores llegados: 10 y se compara con 01 [de la secuencia], tampoco son iguales; ahora se compara con el último valor ingresado: 0 y se lo compara con 0 [de la secuencia], que sí corresponden, pero 0 implica el estado B, por tanto el circuito debe regresar al estado B, como se dijo antes.

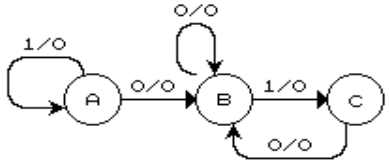


FIGURA 4.40

Pero si estando en C, el próximo valor de X es 1, que corresponde al tercer dato consecutivo de la secuencia, entonces se crea el estado D [fig. 4.41]. El estado D significa que ha llegado el tercer dato consecutivo de la secuencia [011]. Con esto se ha terminado el análisis desde el estado C.

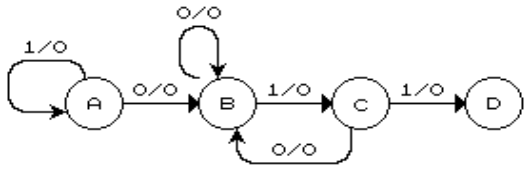


FIGURA 4.41

Si el estado actual es D [que significa que ha llegado 011] y el próximo valor de X es 0, puesto que corresponde al cuarto dato consecutivo de la secuencia, se crea el estado E [fig. 4.42]. El estado E significa que ha llegado el cuarto dato consecutivo de la secuencia [0110, en este ejemplo].

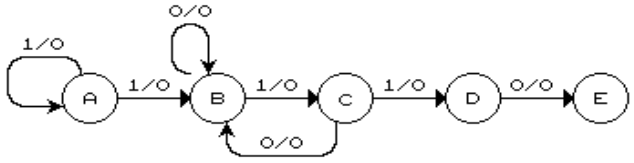


FIGURA 4.42

Pero si el estado es D y el próximo valor de X es 1, que no corresponde al siguiente bit de la secuencia, se analizan los valores que han llegado antes para ver si se puede usar alguno de los estados que se han creado hasta aquí. Es decir, comparamos los 4 últimos bits llegados [0111] y se los compara con los 4 primeros bits de la secuencia [0110], se ve que no corresponden, entonces se utilizan los 3 últimos bits que llegaron [111] se los compara con los 3 primeros bis de la secuencia [011], no

corresponden, luego se utilizan los 2 últimos de X [11] y se los compara con los 2 primeros bits de la secuencia [01], tampoco corresponden, finalmente se compara el último bit que ha llegado en X [1] con el primer bit de la secuencia [0] y no corresponde. En este caso se dice que la secuencia se rompe completamente y por tanto el circuito debe regresar al estado inicial A, para empezar una nueva detección de la secuencia [fig. 4.43]. Con esto se termina el análisis desde el estado D.

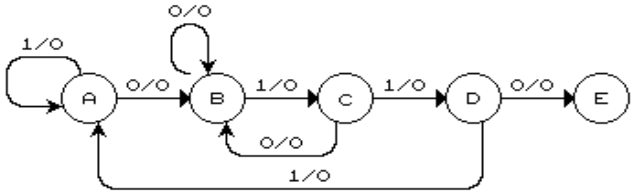


FIGURA 4.43

Estando en E [que significa que han llegado los bits 0110, en forma consecutiva], si el nuevo valor de X es 0, que pertenece al siguiente dato de la secuencia, se crea el estado F [fig. 4.44].

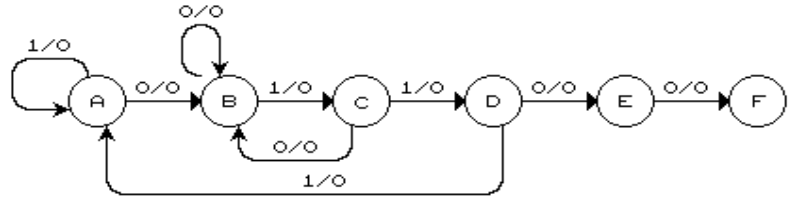


FIGURA 4.44

El estado F, significa que ha llegado el quinto bit consecutivo de la secuencia, es decir, 01100. Pero si estando en E, el próximo valor de X es 1, que no corresponde al siguiente dato consecutivo de la secuencia, se analizan los valores que han llegado antes para ver si se puede usar alguno de los estados que se han creado con anterioridad. Es decir, se compara [01101 llegados en X] con [01100 de la secuencia], como no corresponden, ahora se analiza [1101 de X] con [0110 de la secuencia], tampoco corresponden, entonces se comparan los tres últimos bits llegados [101] con los 3 primeros bits de la secuencia [011], que no son iguales, después se comparan [01 de X] con [01 de la secuencia], se ve que son iguales y corresponde al estado C [que significa que ha llegado 01]. En este caso se ve que la secuencia se rompe parcialmente y por tanto el circuito debe regresar al estado C, porque pueden usarse los valores 01 que fueron los dos últimos que ingresaron a través de la variable de entrada X [fig. 4.45]. Con esto se termina el análisis desde el estado E.

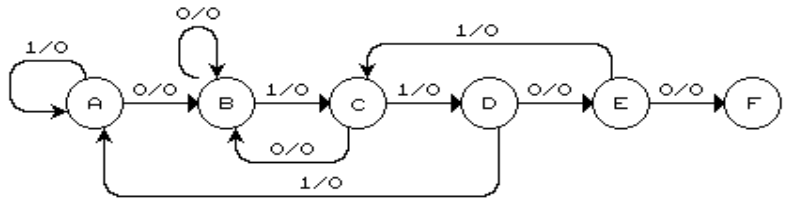


FIGURA 4.45

Estando en F, si el nuevo valor de X es 0, que no corresponde al siguiente dato consecutivo de la secuencia. En este caso, haciendo el mismo tipo de comparaciones que se hicieron antes, se ve que la secuencia se rompe parcialmente y por tanto el circuito debe regresar al estado B, porque puede usarse el valor 0 que fue el último que ingresó a través de la variable de entrada X [fig. 4.46]. Recuerde que B significa que solo ha llegado el primer bit válido de la secuencia.

Hasta aquí todas las salidas actuales valen 0, porque no se ha completado la secuencia pedida.

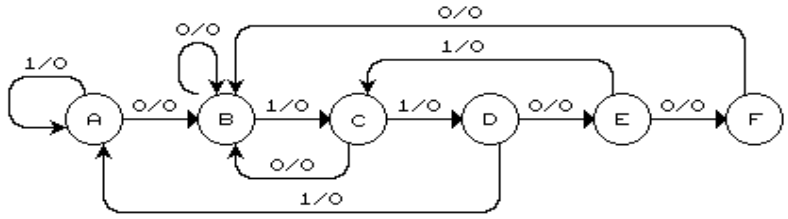


FIGURA 4.46

Si el estado actual es F [que significa que la llegado 01100, en forma consecutiva] y el nuevo valor de X es 1, que corresponde al último dato válido de la secuencia, y como el problema pide que una vez que se ha terminado de detectar la secuencia, la salida Z<t> [salida actual] tome el valor 1 y, además, se regrese al estado inicial A [fig. 4.47]. Con esto se termina el análisis desde el estado F y al no

haberse creado nuevos estados, el análisis también termina.

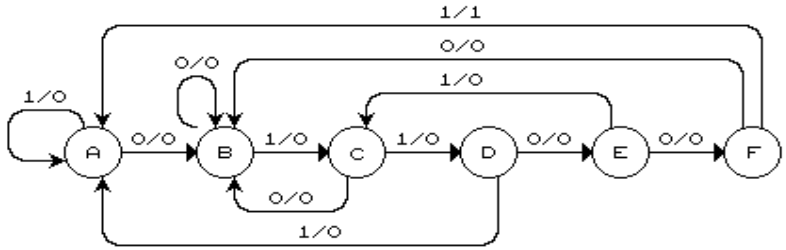


FIGURA 4.47

La fig. 4.48, muestra el diagrama de estados completo, en él se ha incluido la entrada ***Master-Reset*** (***M.R.***) que permite reinicializar el circuito en cualquier momento, en la mayoría de los casos puede considerársela igual a la ***entrada de borrado***.

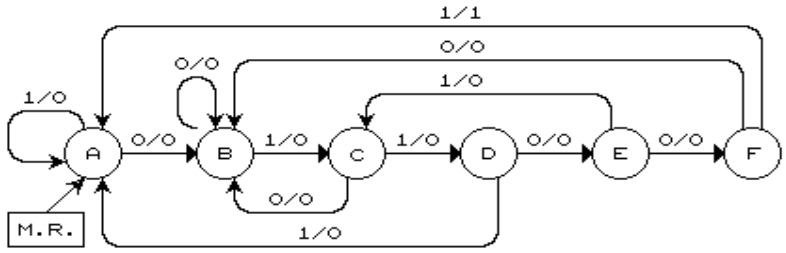


FIGURA 4.48 DIAGRAMA DE ESTADOS COMPLETO

De este diagrama se obtiene la tabla de estados que se muestra a continuación. Inmediatamente se hace la asignación de estados, en este caso se ha escogido la opción de estados continuos del 0 al 5, esto se

lo hace por facilidad, aunque de ninguna manera garantice que sea el circuito lógico más simple. En base a la asignación, se obtiene la tabla de transición de estados.

ESTADO ACTUAL \ ENTRADA ACTUAL	0	1
A	B/0	A/0
B	B/0	C/0
C	B/0	D/0
D	E/0	A/0
E	F/0	C/0
F	B/0	A/1

ESTADO	ASIGNACIÓN Q2	Q1	Q0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0
F	1	0	1

ESTADO ACTUAL \ ENTRADA ACTUAL	0	1
[000]	[001]/0	[000]/0
[001]	[001]/0	[010]/0
[010]	[001]/0	[011]/0
[011]	[100]/0	[000]/0
[100]	[101]/0	[010]/0
[101]	[001]/0	[000]/1

La asignación de estados que se ha utilizado para este ejemplo, es la más obvia, aunque no garantiza que se obtenga el circuito lógico más simplificado. De la tabla de transiciones se obtiene la siguiente tabla que muestra el comportamiento de la red que

se está diseñando, en ella se han incluido las columnas para las señales de comando actual.

ENT ACT	EST. ACTUAL <t>			EST. PRÓX. <t + 1>			SAL ACT	COMANDO ACTUAL <t>					
X<t>	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Z<t>	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	X	0
0	0	1	0	0	0	1	0	0	X	X	1	1	X
0	0	1	1	1	0	0	0	1	X	X	1	X	1
0	1	0	0	1	0	1	0	X	0	0	X	1	X
0	1	0	1	0	0	1	0	X	1	0	X	X	0
0	1	1	0	X	X	X	X	X	X	X	X	X	X
0	1	1	1	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	X	0	X	0	X
1	0	0	1	0	1	0	0	0	X	1	X	X	1
1	0	1	0	0	1	1	0	0	X	X	0	1	X
1	0	1	1	0	0	0	0	0	X	X	1	X	1
1	1	0	0	0	1	0	0	X	1	1	X	0	X
1	1	0	1	0	0	0	1	X	1	0	X	X	1
1	1	1	0	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X

Mediante cualquiera de los métodos de simplificación, se obtienen las ecuaciones de salida y de comando actuales.

$$Z_{<t>} = X_{<t>}Q_{2<t>}Q_{0<t>}$$

$$J_{2<t>} = \overline{X}_{<t>}Q_{1<t>}Q_{0<t>} \qquad K_{2<t>} = X_{<t>} + Q_{0<t>}$$

$$J_{1<t>} = X_{<t>}(Q_{2<t>} \oplus Q_{0<t>}) \qquad K_{1<t>} = \overline{X}_{<t>} + Q_{0<t>}$$

$$J_{0<t>} = \overline{X}_{<t>} + Q_{1<t>} \qquad K_{0<t>} = X_{<t>} + Q_{1<t>}$$

La implementación del circuito secuencial sincrónico que se muestra en la fig. 4.49.

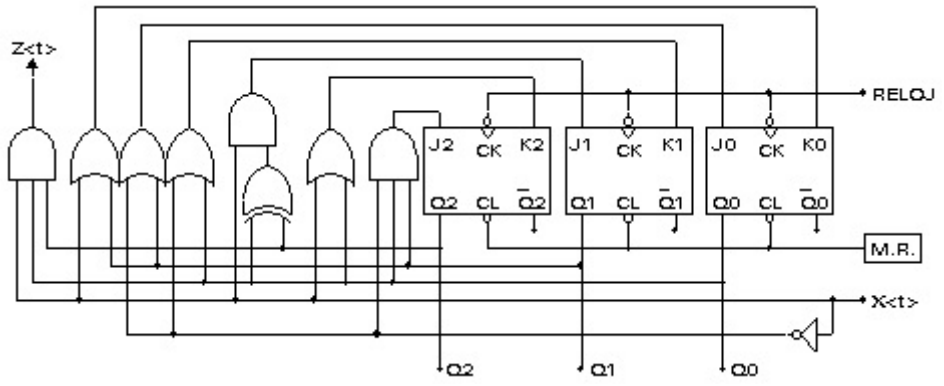


FIGURA 4.49

Una variación del diseño anterior sería si se desea que, una vez detectada la secuencia, la salida Z<t> tome el valor 1 pero con el siguiente pulso de reloj y que además el circuito esté listo para detectar todas las secuencias válidas que lleguen a través de X<t>.

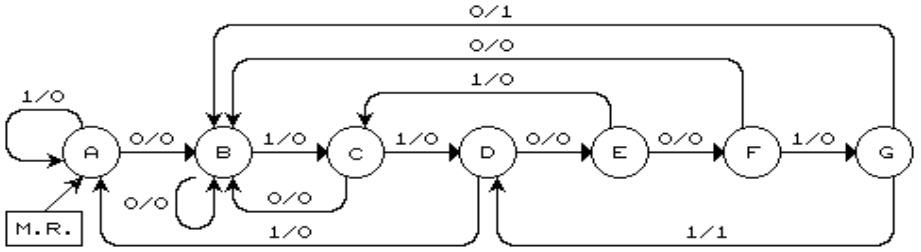


FIGURA 4.50

En esa situación el diagrama de estados tendrá una alteración después del estado F. Esto se muestra en la fig. 4.50.

La fig. 4.50 muestra los cambios que se obtienen según el nuevo planteamiento del problema y siguiendo el mismo procedimiento anterior se obtiene el diagrama de estados en el que se ha incluido una entrada para la inicialización manual [M.R.]. Del diagrama de estados se obtiene la tabla de estados que se muestra a continuación.

ESTADO ACTUAL \ ENTRADA ACTUAL	0	1
A	B/0	A/0
B	B/0	C/0
C	B/0	D/0
D	E/0	A/0
E	F/0	C/0
F	B/0	G/0
G	B/1	D/1

ESTADO	ASIGNACIÓN Q2	Q1	Q0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

ESTADO ACTUAL \ ENTRADA ACTUAL	0	1
[000]	[001]/0	[000]/0
[001]	[001]/0	[010]/0
[010]	[001]/0	[011]/0
[011]	[100]/0	[000]/0
[100]	[101]/0	[010]/0

F	1	0	1
G	1	1	0

[101]	[001]/0	[110]/0
[110]	[001]/1	[011]/1

De la tabla de transición de estados se obtiene la tabla que muestra el comportamiento de la red que se está diseñando, en ella se han incluido las columnas para las señales de comando actual.

ENT ACT	EST. ACTUAL <t>			EST. PRÓX. <t + 1>			SAL ACT	COMANDO ACTUAL <t>					
X<t>	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Z<t>	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	X	0
0	0	1	0	0	0	1	0	0	X	X	1	1	X
0	0	1	1	1	0	0	0	1	X	X	1	X	1
0	1	0	0	1	0	1	0	X	0	0	X	1	X
0	1	0	1	0	0	1	0	X	1	0	X	X	0
0	1	1	0	0	0	1	1	X	1	X	1	1	X
0	1	1	1	X	X	X	X	X	X	X	X	X	X
1	0	0	0	0	0	0	0	0	X	0	X	0	X
1	0	0	1	0	1	0	0	0	X	1	X	X	1
1	0	1	0	0	1	1	0	0	X	X	0	1	X
1	0	1	1	0	0	0	0	0	X	X	1	X	1
1	1	0	0	0	1	0	0	X	1	1	X	0	X
1	1	0	1	1	1	0	0	X	0	1	X	X	1
1	1	1	0	0	1	1	1	X	1	X	0	1	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X

Mediante cualquiera de los métodos de simplificación, se obtienen las ecuaciones de salida y de comando actuales.

$$Z_{<t>} = Q_{2<t>}Q_{1<t>}$$

$$J_{2<t>} = \bar{X}_{<t>}Q_{1<t>}Q_{0<t>}$$
$$K_{2<t>} = X_{<t>} \oplus Q_{0<t>} + Q_{1<t>}$$

$$J_{1<t>} = X_{<t>}Q_{2<t>} + X_{<t>}Q_{0<t>} \qquad K_{1<t>} = \bar{X}_{<t>} + Q_{0<t>}$$

$$J_{0<t>} = \bar{X}_{<t>} + Q_{1<t>} \qquad K_{0<t>} = X_{<t>} + Q_{1<t>}$$

El circuito correspondiente se muestra en la fig. 4.51. En él pueden verse algunos cambios, como era de esperar. Se ha agregado las salidas [Q_2, Q_1 y Q_0] de los FFs para poder observar los estados de la red secuencial.

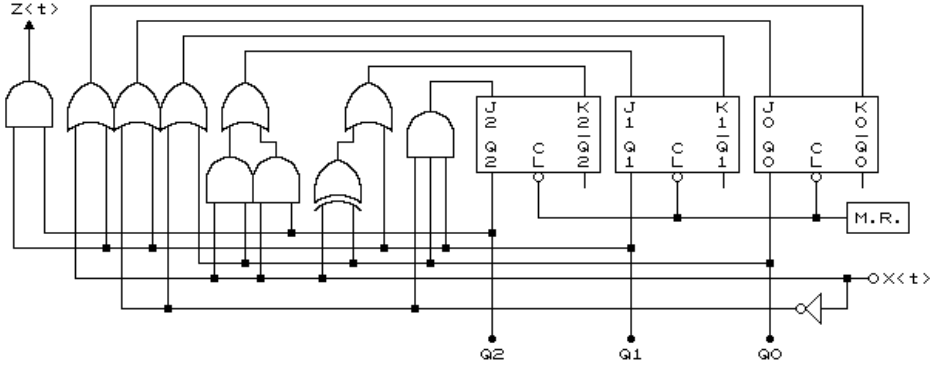


FIGURA 4.51

Ejemplo 4.- Utilice flip-flops tipo-JK para diseñar un circuito secuencial sincrónico que permita detectar la siguiente secuencia de bits. Una vez detectada

la secuencia, el circuito debe regresar al estado inicial para empezar una nueva detección.

t	0	1	2	3	4	5
X<t>	1	0	1	1	1	0

Estado inicial, no ha llegado ningún dato de la secuencia.

FIGURA 4.52

Cero no es el primer dato de la secuencia, entonces el circuito permanece en el estado q0 hasta que llegue el primer dato de la secuencia pedida, fig 4.53.

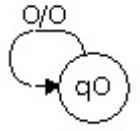


FIGURA 4.53

Llega el primer válido bit de la secuencia, por tanto se crea un nuevo estado, [q1], fig. 4.54.

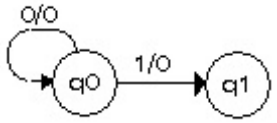


FIGURA 4.54

Llega el segundo dato consecutivo de la secuencia, se crea un nuevo estado [q2] fig. 4.55.

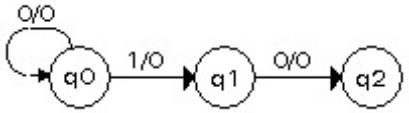


FIGURA 4.55

Solo se puede utilizar el último dato que ha llegado y que corresponde al estado q1, fig. 4.56.

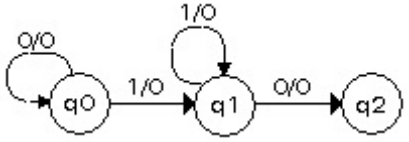


FIGURA 4.56

Si estando en el estado q2, llega un cero, se rompe la secuencia y el circuito debe regresar al estado inicial [q0], para empezar de nuevo la detección de la secuencia, fig. 4.57.

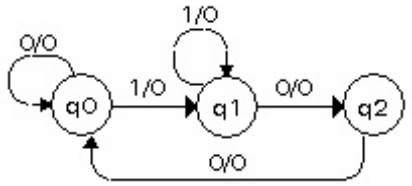


FIGURA 4.57

Cuando llega el tercer valor de la secuencia, se crea un nuevo estado [q3] que recuerda que ha llegado el tercer dato consecutivo de la secuencia deseada [en este caso, 101], fig. 4.58.

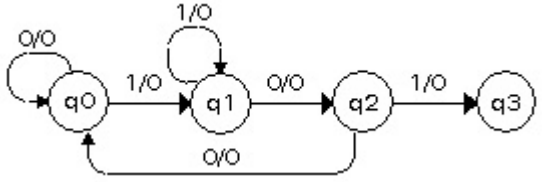


FIGURA 4.58

Estando en q3 llega un 0, entonces se compara [1010] de X con [1011] de la secuencia, no corresponden. Luego se compara [010] de X con [101] de la secuencia, tampoco corresponden. Entonces se comparan los dos últimos bits llegados a través de X [10] con los dos primeros bits de la secuencia pedida [10], si son iguales y corresponden al estado q2, por tanto el circuito regresa al estado q2 [fig. 4.59].

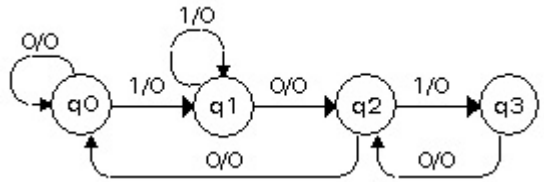


FIGURA 4.59

Llega el cuarto valor consecutivo de la secuencia, fig. 4.60.

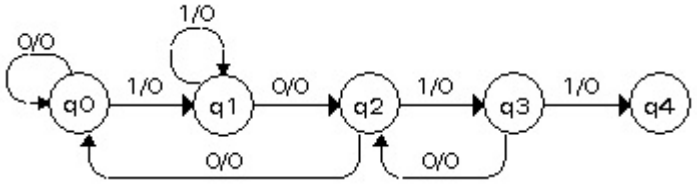


FIGURA 4.60

Estando en q4 [1011] llega un 0, entonces se compara [10110 de X] con [10111 de la secuencia], y se ve que no son iguales, de manera que ahora se comparan los 4 últimos bits que llegaron a través de X [0110] con los 4 primeros bits de la secuencia [1011], tampoco son iguales; se sigue la comparación y ahora se lo hace con los 3 últimos bits de X [110] y los 3 primeros de la secuencia [101] y no corresponden, entonces se comparan los 2 últimos bits llegados en X [10] con 2 primeros bits de la secuencia [10], son iguales y corresponden al estado q2 [q2 significa que han llegado 2 valores consecutivos de la secuencia], por tanto, el circuito debe regresar al estado indicado, fig. 4.61.

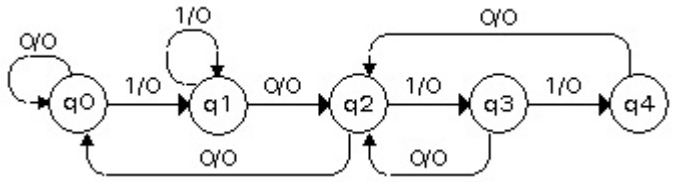


FIGURA 4.61

Llega el quinto valor consecutivo de la secuencia, fig. 4.62.

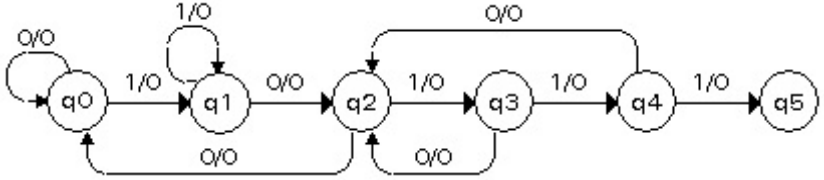


FIGURA 4.62

Llega el sexto valor consecutivo que completa la secuencia pedida, la salida actual Z<t> toma el valor 1 y el circuito regresa al estado inicial para empezar una nueva detección, fig. 4.63.

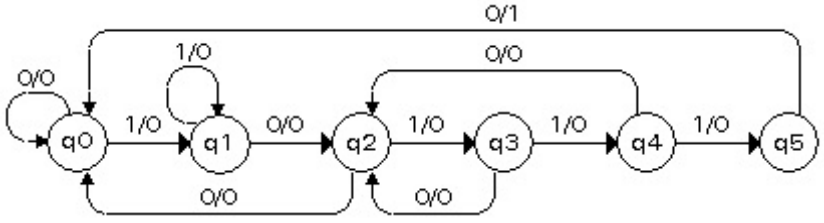


FIGURA 4.63

Si estando en q5 y llega un 1, no se completa la secuencia pero el circuito debe regresar al estado

q1, que indica que ha llegado el primer valor de la secuencia, fig. 4.64.

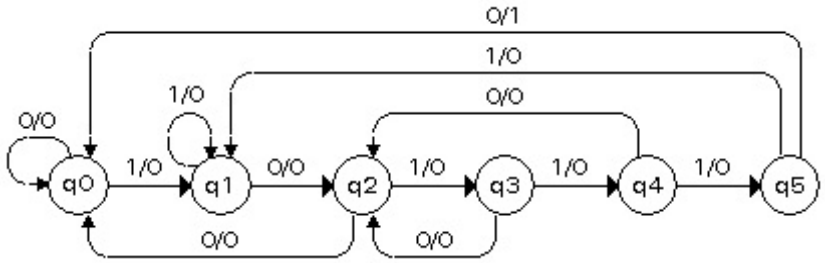


FIGURA 4.64

En la fig. 4.65 se muestra el diagrama de estados completo en el que se ha incluido la entrada M.R. para reiniciar el circuito cuando sea necesario, se adjunta la tabla de estados.

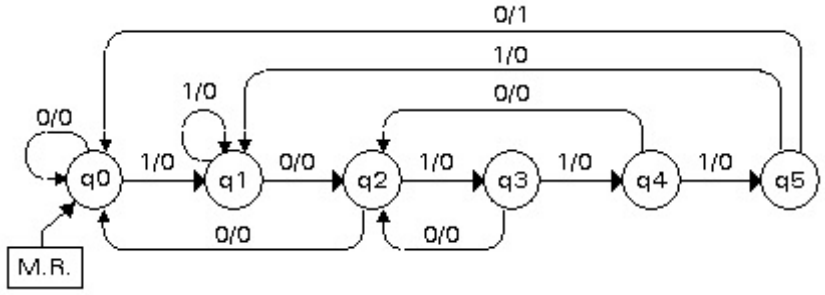


FIGURA 4.65

A continuación y siguiendo los pasos indicados anteriormente se obtiene la tabla de estados, que se muestra a continuación.

ESTADO \ ENTRADA ACTUAL	0	1
q0	q0/0	q1/0
q1	q2/0	q1/0
q2	q0/0	q3/0
q3	q2/0	q4/0
q4	q2/0	q5/0
q5	q0/1	q0/0

A continuación sería la asignación de estados, la tabla de transición de estados, finalmente la tabla del comportamiento de la red secuencial que se está diseñando, estas tablas mo se muestran en este ejemplo. Mediante cualquiera de los métodos de simplificación, se obtienen las ecuaciones simplificadas de comando y de salida actuales para implementar la red lógica secuencial sincrónica que se presenta en la fig. 4.66.

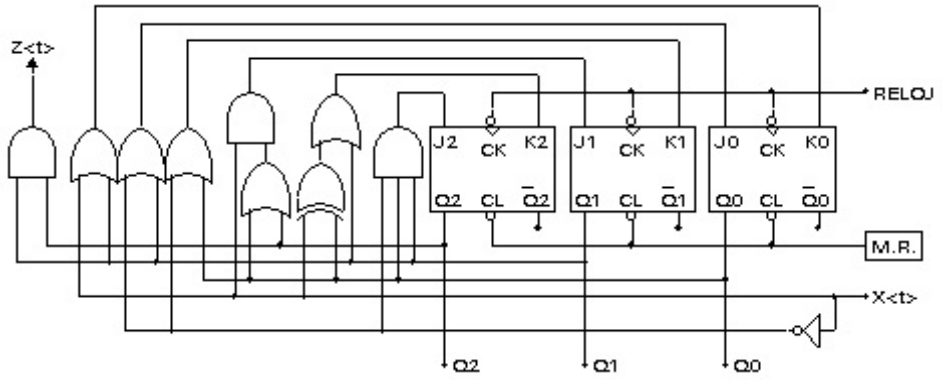


FIGURA 4.66

Otra vez, puede presentarse una variación al diseño anterior que sería si se desea que, una vez detectada la secuencia, la salida Z<t> tome el valor 1 pero con el siguiente pulso de reloj y que además el circuito esté listo para detectar todas secuencias que lleguen a través de X<t>. El diagrama de estados resultante se muestra en la fig. 4.67.

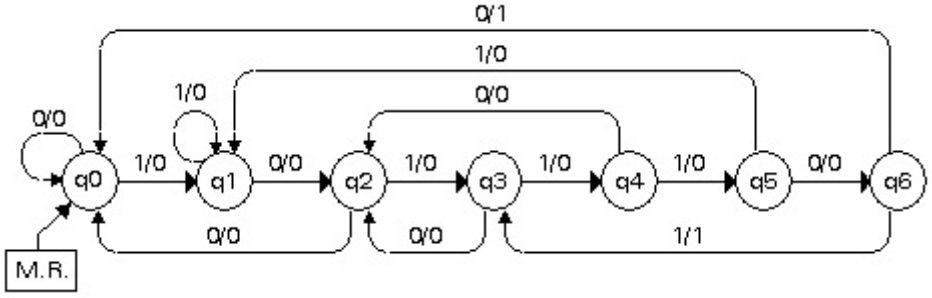


FIGURA 4.67

El resto del diseño sigue los mismos pasos que se han indicado en ejemplos anteriores.

Ejemplo 5.- Utilice flip-flops tipo-JK para diseñar un circuito secuencial sincrónico que permita detectar la siguiente secuencia de bits. Una vez detectada la secuencia, el circuito debe regresar al estado inicial para empezar una nueva detección.

t	0	1	2	3	4	5
X<t>	0	0	1	1	0	1

Estado inicial, fig. 4.68.

FIGURA 4.68

Primer valor de la secuencia, fig. 4.69.

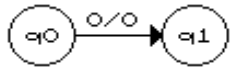


FIGURA 4.69

Si en el estado inicial llega un 1, que no corresponde al primer bit de la secuencia, el circuito permanece en q0, fig. 4.70. Se completa el análisis desde q0.

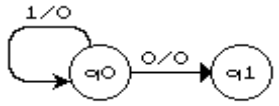


FIGURA 4.70

Ahora se analiza desde q1, si el siguiente bit es 0, que corresponde al segundo bit de la secuencia pedida, se crea el tercer estado [q2 que equivale a 00], fig. 4.71.

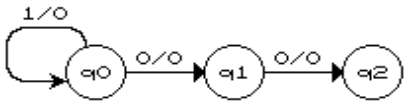


FIGURA 4.72

Si estando en q1 llega un 1, se rompe la secuencia que se había empezado a detectar y la red regresa al estado inicial q0, fig. 4.73. Se termina el análisis desde q1.

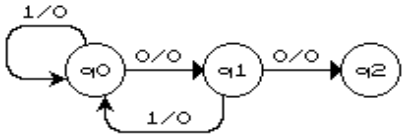


FIGURA 4.73

Esta vez se analiza desde el estado q2 que significa que ha llegado [00]. Si el siguiente bit es 0, se comparan

```
 X<t>      Secuencia
[000] con [001], no son iguales, entonces
 [00] con [00],  son iguales, por tanto q2,
```

entonces el circuito permanece en q2, fig. 4.74.

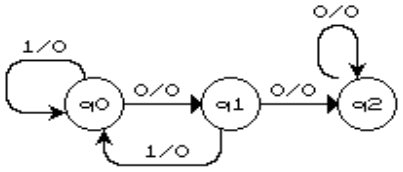


FIGURA 4.74

Si el estado actual es q2 [00] y llega un 1,

corresponde al tercer bit consecutivo de la secuencia, y se crea el estado q3 que recuerda que ha llegado el tercer bit de la secuencia, fig. 4.75. Se termina el análisis desde q2.

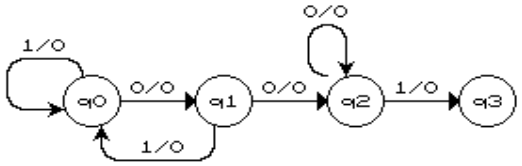


FIGURA 4.75

Se inicia el análisis desde el estado q3. Si llega un 0, entonces se realizan las siguientes comparaciones.

```
 X<t>      Secuencia
[0010] con [0011], no son iguales
 [001] con [001],  no corresponden
  [01] con [01],   no corresponden
   [0] con [0],    igual al estado q1.
```

Por tanto, la red regresa a q1, fig. 4.76.

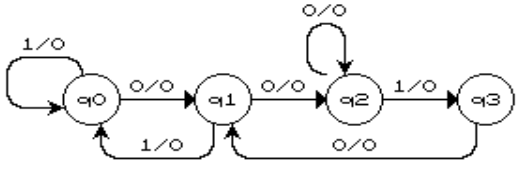


FIGURA 4.76

Desde q3 [001] si llega un 1, es decir hasta este momento han llegado los valores 0011 en forma

consecutiva, que corresponde al cuarto bit de la secuencia pedida, por tanto se crea el estado q4 que recuerda que ha llegado el cuarto bit consecutivo de la secuencia, fig. 4.77. Aquí termina el análisis desde el estado q3.

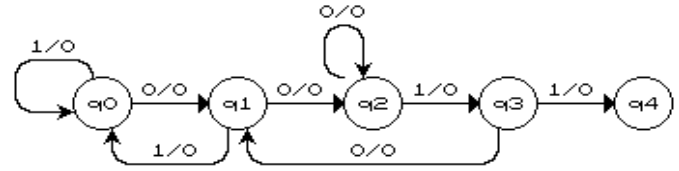


FIGURA 4.77

Ahora si el estado es q4 [0011] y si llega un 0 que corresponde al quinto bit consecutivo de la secuencia, entonces se crea el estado q5 que indica que ha llegado un bit más de la secuencia, fig. 4.78.

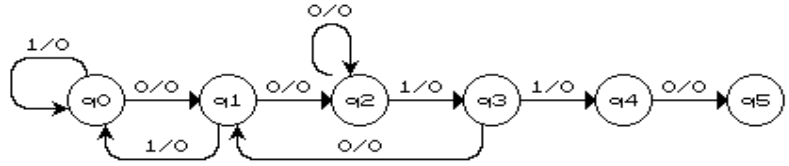


FIGURA 4.78

Si el estado actual es q4, y si llega un 1, entonces se realizan las siguientes comparaciones.

```
  X<t>       Secuencia
[00111] con [00110], no son iguales
 [0111] con [0011],  no corresponden
```



```
  [111] con [001],   no corresponden
   [11] con [00],    no son iguales
    [1] con [0],     no son iguales
```

Por tanto, la secuencia se rompe totalmente y la red regresa al estado inicial q0, fig. 4.79.

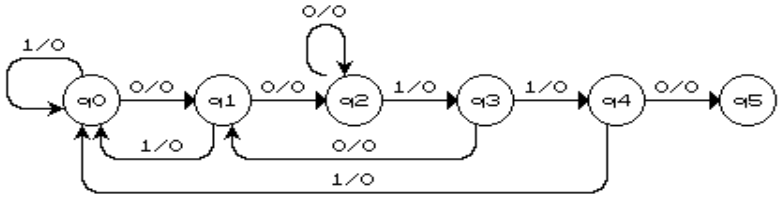


FIGURA 4.79

Ahora, si el estado actual es q5, y si llega un 0, entonces se realizan las siguientes comparaciones.

```
 X<t>         Secuencia
[001100] con [001101], no son iguales
 [01100] con [00110],  no corresponden
  [1100] con [0011],   no corresponden
   [100] con [001],    no son iguales
    [00] con [00],     son iguales y corresponde al
```

estado q2.

Entonces, el circuito regresa al estado q2, fig. 4.80.

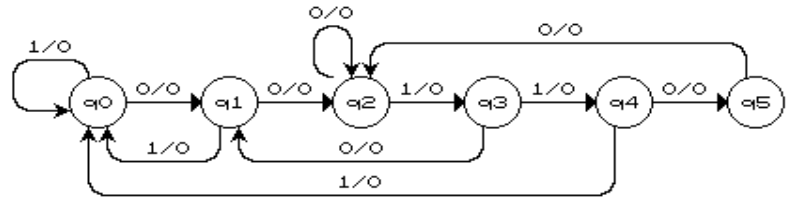


FIGURA 4.80

Si el estado actual es q5 [00110] y si llega un 1 que corresponde al último bit consecutivo que completa la secuencia, entonces se regresa al estado inicial q0 y la salida toma el valor 1, fig. 4.81.

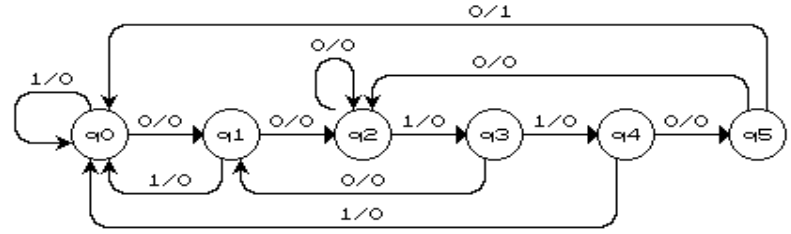


FIGURA 4.81

En la fig. 4.82 se muestra el diagrama de estados completo en el que se ha incluido la entrada M.R. y se adjunta la tabla de estados. Siguiendo los pasos estudiados anteriormente [tabla de estados, asignación de estados, tabla de excitación y tabla del comportamiento de la red secuencial] se obtienen las ecuaciones simplificadas de comando y de salida actuales para implementar la red lógica secuencial sincrónica que se presenta en la fig. 4.83.

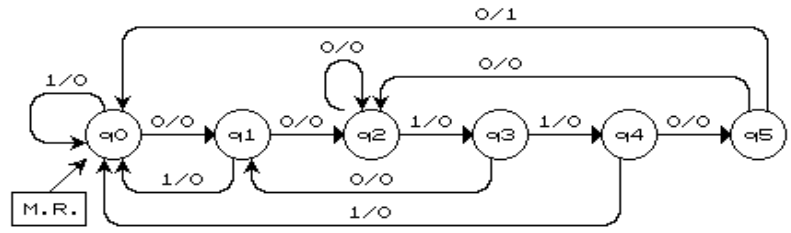


FIGURA 4.82

ESTADO \ ENTRADA ACTUAL	0	1
q0	q1/0	q0/0
q1	q2/0	q0/0
q2	q2/0	q3/0
q3	q1/0	q4/0
q4	q5/0	q0/0
q5	q2/0	q0/1

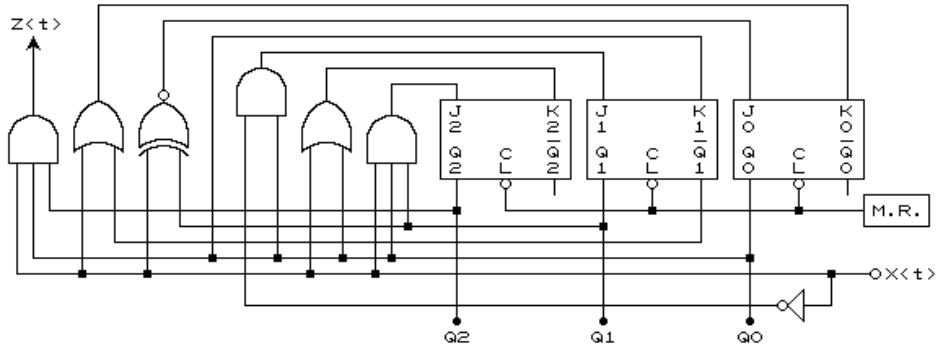


FIGURA 4.83

Otra vez, puede presentarse una variación al diseño anterior que sería si se desea que, una vez detectada

la secuencia, la salida Z<t> tome el valor 1 con el siguiente pulso de reloj y que además el circuito esté listo para detectar todas secuencias que lleguen a través de X<t>. El diagrama de estados resultante se muestra en la fig. 4.84.

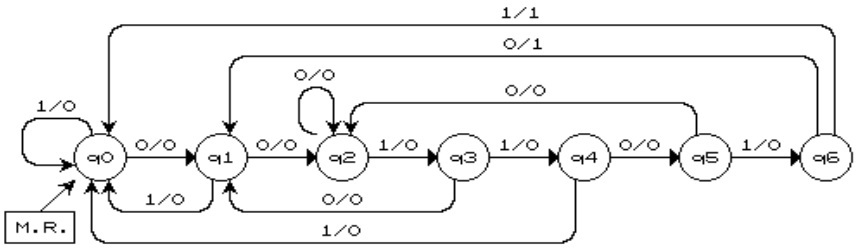


FIGURA 4.84

ESTADO \ ENTRADA ACTUAL	0	1
q0	q1/0	q0/0
q1	q2/0	q0/0
q2	q2/0	q3/0
q3	q1/0	q4/0
q4	q5/0	q0/0
q5	q2/0	q6/0
q6	q1/1	q0/1

Siguiendo los pasos de los problemas anteriores [asignación de estados, tabla de excitación y tabla del comportamiento de la red secuencial] se obtienen las ecuaciones de comando y de salida actuales para implementar la red lógica secuencial sincrónica que

se presenta en la fig. 4.85.

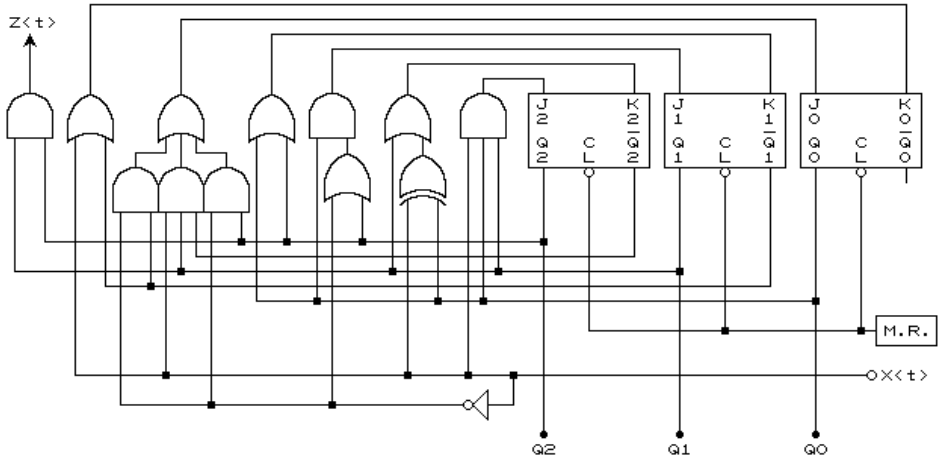


FIGURA 4.85

Ejemplo 6.–En base a un registro de desplazamiento diseñar un circuito que detecte la secuencia que se muestra a continuación. La salida debe ser 1_L cada vez que detecte la secuencia pedida.

t	0	1	2	3	4	5	6	7
X<t>	1	0	1	1	0	1	0	0

La fig. 4.86 muestra la solución del circuito secuencial pedido. El valor 1 que ingresa al tiempo t = 0, después de 8 pulsos de reloj ocupará la posición Q7, el 0 que ingresa al tiempo t = 1, después de 7 pulsos de reloj llegará a ocupar la posición Q6 y así sucesivamente, de modo que cuando ingrese la secuencia correcta, todas la entradas de la

compuerta AND tendrán el valor 1, y la salida Z<t>, será 1.

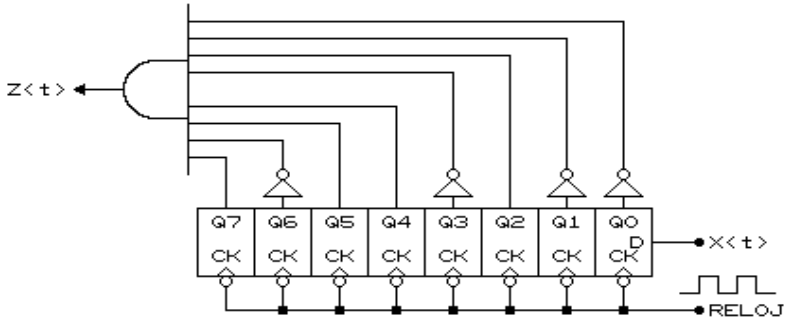


FIGURA 4.86

Se observa que cuado el valor de la secuencia es 1, esa salida va directamente a la compuerta AND y cuando es 0, se requiere un inversor para que pueda ingresar a la compuerta AND.

Ejemplo 7.- Diseñar un circuito secuencial sincrónico que permita detectar la siguiente secuencia [incluir un Master-Reset].

t	0	1	2	3
X_0<t>	1	0	0	1
X_1<t>	1	0	1	0

D:\RESPALDOS\SD-Cap04.wpd

Revisión: Junio - 2008