UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
Escuela Nacional de Estudios Profesionales
" ARAGON "

DISEÑO LOGICO

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A:

JORGE SUASTEGUI NAVA

ENEP

ARAGON

Asesor: Ing. David J. Gonzalez Maxinez

San Juan Aragón, Edo. de México 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

D I S E N O L O G I C O

CONTENIDO :

INTRODUCCION

El diseño logico se ocupa del diseño de circuitos electronicos digitales. El tema se conoce tambien por otros nombres, como *diseño digital, circuitos conmutadores, logica digital y sistemas digitales*. Los circuitos digitales se emplean en el diseño de sistemas, por ejemplo computadoras digitales, calculadoras electronicas, dispositivos digitales de control, equipo de comunicacion digital y muchas otras aplicaciones que requieren hardware digital electronico.

Los presentes apuntes están enfocados al auxilio didáctico tanto de alumnos, como de profesores de la asignatura de Diseño Lógico correspondiente a la carrera de Ingeniería en Computación.

Uno de los objetivos del presente trabajo es proporcionar herramientas basicas que se emplean en el diseño de los circuitos digitales, además, proporcionar metodos y procedimientos adecuados para una variedad de aplicaciones del diseño logico.

El trabajo está formado por seis capítulos que proporcionan un conjunto coherente de temas adecuados para la asignatura de Diseño Logico.

El capitulo I, es la introducción al álgebra booleana junto con las diversas compuertas logicas que se emplean en la construccion de circuitos digitales.

El capitulo II, presenta los teoremas de reducción además de varios metodos para la minimizacion de una o varias funciones de salida.

El capitulo III, trata acerca de la electronica de los circuitos digitales y se presentan las familias logicas mas comunes. En este capitulo se supone cierto conocimiento de electronica analogica.

El capitulo IV, trata con los componentes de los circuitos combinacionales MSI y LSI. Se explican las funciones de uso frecuente como sumadores, restadores, multiplicadores, decodificadores, multiplexores, comparadores y verificadores de paridad.

El capitulo V, se compendian procedimientos para el analisis y diseño de circuitos secuenciales sincronos y asincronos.

El capitulo VI, es la introduccion a los elementos de memoria mostrando estructura, parametros y funciones de cada uno de ellos.

En la mayoria de los capitulos, se anexan ejercicios propuestos para que el lector ponga en practica lo aprendido en el capitulo correspondiente asi como en los capitulos anteriores.

Es preciso mencionar que los capitulos referentes a los sistemas de numeracion y codigos que se establecen en el temario de la asignatura de Diseño Logico, estan incluidos en el apendice del presente trabajo.

CAPITULO I : ALGEBRA BOOLEANA Y COMPUERTAS

CONTENIDO :

CAPITULO I : ALGEBRA BOOLEANA Y COMPUERTAS.

El algebra booleana es un sistema cerrado que consiste en un conjunto B de dos o mas elementos distintos, que designaremos por 0 y 1 y que estan relacionados por dos operaciones binarias (+ y .) y una operacion unaria (') denominadas suma, producto y negacion logica, las cuales satisfacen los axiomas que se mencionaran.

1.1. TABLAS DE VERDAD.

Los circuitos digitales (lógicos) operan en modo binario donde cada voltaje de entrada y salida son un 0 o bien un 1; las designaciones 0 y 1 representan intervalos predefinidos de voltaje.

Muchos circuitos logicos tienen mas de una entrada y solamente una salida. Una tabla de verdad muestra la forma en que la salida del circuito logico responda a las diversas combinaciones de niveles logicos en las entradas. El formato para tablas de verdad de 2, 3 y 4 entradas se presenta en la tabla 1.1.

En cada tabla de verdad las combinaciones posibles de niveles logicos 0 y 1 para las entradas (ABCD) se enlistan del lado izquierdo y el nivel logico resultante para la salida X se enlista a la derecha. Las salidas para X se muestan como "?" por ahora, debido a que estos valores seran diferentes en cada tipo de circuito logico.

Notese que hay cuatro valores en la tabla para la de 2 entradas, 8 para la tabla de verdad de 3 entradas y 16 valores para la de 4. El numero de combinaciones de entrada sera igual a 2 elevado a la N para una tabla de verdad de N entradas. Observese tambien que la lista de todas las posibles combinaciones de entrada sigue la secuencia de conteo binario, asi que resulta

sencillo expresar todas las combinaciones sin omitir una sola.

ENTRADAS A	B	SALIDA X
0	0	?
0	1	?
1	0	?
1	1	?

(a)

ENTRADAS A	B	C	SALIDA X
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

(b)

ENTRADAS A	B	C	D	SALIDA X
0	0	0	0	?
0	0	0	1	?
0	0	1	0	?
0	0	1	1	?
0	1	0	0	?
0	1	0	1	?
0	1	1	0	?
0	1	1	1	?
1	0	0	0	?
1	0	0	1	?
1	0	1	0	?
1	0	1	1	?
1	1	0	0	?
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

(c)

TABLA 1.1 TABLAS DE VERDAD.

1.2. COMPUERTAS LOGICAS Y BASICAS.

Las compuertas lógicas, son los circuitos lógicos más fundamentales, pueden combinarse para producir circuitos lógicos y como estos circuitos pueden ser descritos y analizados por medio del algebra booleana, fijaremos nuesta atencion en la operacion lógica que cada compuerta puede realizar; no nos interesaran los circuitos internos que permiten a la compuerta operan en la forma que lo hacen.

1.2.1. LA COMPUERTA OR.

En un circuito digital la compuerta OR es un circuito que tiene dos o mas entradas y cuya salida es igual a la suma OR de las entradas. La figura 1.1 muestra el simbolo correspondiente a la compuerta OR de 2 entradas. Las entradas A y B son niveles de voltaje lógicos y la salida X es un nivel de voltaje logico cuyo

Mediante el uso del lenguaje del algebra booleana, la salida X puede expresarse como X = A+B+C, donde una vez más debe hacerse hincapie en que el signo + significa adición OR. Por consiguiente, la salida de cualquier compuerta OR se puede expresar como la suma OR de todas sus entradas. Esto lo usaremos cuando se analicen circuitos logicos en forma subsecuente.

Ejemplo : *En el caso que representa la figura 1.3 determine la forma de onda en la salida de la compuerta OR.*

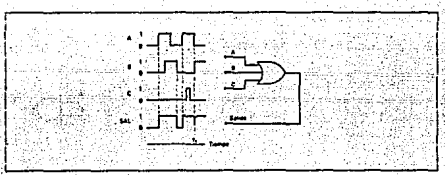


FIG. 1.3

Solución : Las 3 entradas de la compuerta OR, A, B y C son variantes, como lo muestran sus diagramas de formas de ondas. La salida de la compuerta se determina entendiendo que será alta cuando cualquiera de las 3 entradas esté en un nivel alto. Con este razonamiento, la onda de salida de la OR es como se muestra en la figura.

Debe prestarse especial atención a lo que sucede en el tiempo t1. El diagrama muestra que en ese instante de tiempo la entrada A pasa de alto a bajo en tanto que la entrada B pasa de bajo a alto. Ya que estas entradas efectuan sus transiciones en aproximadamente el mismo tiempo y debido a que dichas transiciones se llevan cierta cantidad de tiempo, hay un intervalo corto de tiempo cuando estas entradas de la compuerta OR se encuentran en el intervalo indefinido entre 0 y 1. Cuando esto sucede, la salida de la compuerta OR se convierte así mismo en un valor que se clasifica en este ambito, como lo indica el mal funcionamiento en la onda de salida t1. La aparición de este mal funcionamiento y su magnitud (amplitud y anchura) dependen de la velocidad con la cual se efectuen las transiciones de entrada. Debe observarse que si la entrada C hubiera estado en alto cuando A y B se encontraran cambiando, el mal funcionamiento no se presentaría debido a que el estado alto de C conservaría en este mismo estado la salida de OR.

valor es el resultado de la adicion OR de A y B; esto es, X = A+B.

En otras palabras, la compuerta OR opera en tal forma que su salida es ALTA (HIGH, nivel logico 1) si la entrada A, B o ambas están en un nivel logico 1. La salida de la compuerta OR sera BAJA (LOW, nivel logico 0) si todas sus entradas estan en el nivel logico 0.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Símbolo Americano

Símbolo Europeo

FIG. 1.1 COMPUERTA OR DE DOS ENTRADAS.

Esta misma idea puede ampliarse a más de dos entradas. La figura 1.2 muestra una compuerta OR de 3 entradas y su tabla de verdad. El analisis de esta tabla muestra una vez mas que la salida sera 1 en cualquier caso donde una o mas entradas sea 1. Este principio general es el mismo que rige para compuertas OR con cualquier numero de entradas.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Símbolo Americano

Símbolo Europeo

FIG 1.2 COMPUERTA OR DE TRES ENTRADAS.

Resumen de la operación OR :

1. La operación OR produce un resultado de 1 cuando cualquiera de las variables de entrada es 1.

2. La operación OR genera un resultado de 0 solamente cuando todas las variables de entrada son 0.

3. En la adición OR, 1 + 1 = 1, 1 + 1 + 1 = 1, etc.

1.2.2. LA COMPUERTA AND.

En la figura 1.4 se muestra en forma simbólica, una compuerta AND de 2 entradas junto con su tabla de verdad :

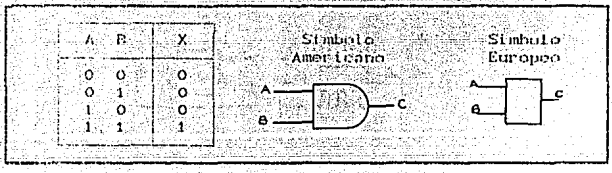


FIG. 1.4 COMPUERTA AND DE DOS ENTRADAS.

La salida de la compuerta AND es igual al producto AND de las entradas lógicas; es decir, X = AB. En otras palabras, la compuerta AND es un circuito que opera en forma tal que su salida sea alta solo cuando todas sus entradas sean altas. En todos los otros casos la salida de la compuerta AND es baja.

Esta misma operación es característica de las compuertas AND con mas de 2 entradas. Por ejemplo, una compuerta AND de 3 entradas y su tabla de verdad acompañando se muestran en la figura 1.5.

Una vez mas, notese que la salida de la compuerta es 1 solo en el caso cuando A = B = C = 1. La expresión para la salida es X=ABC. Para las compuerta AND de 4 entradas, la salida es X=ABCD, y así sucesivamente.

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

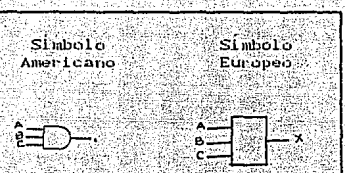


FIG. 1.5 COMPUERTA AND DE TRES ENTRADAS.

Ejemplo : *Determine la onda de salida de la compuerta AND que se muestra en la figura 1.6 :*

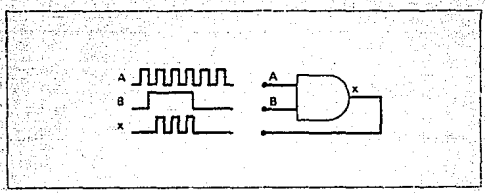


FIG. 1.6

Solución : La salida X será un 1 solo cuando A y B sean altas al mismo tiempo. Por este hecho la onda X se puede determinar como se muestra en la figura 1.6.

Obsérvese que la onda de X es 0 siempre que B es 0, independientemente de la señal en A. Obsérvese así mismo que la onda de X es la misma que A siempre que B es 1. Así, podemos pensar que la entrada B es una entrada de control cuyo nivel lógico determina si la onda de A se dirige hacia la onda X o no. En esta situación, la compuerta AND se utiliza como circuito inhibidor. Podemos decir que B = 0 es la condición de inhibición que produce una salida de A la inversa, B = 1 es la condición facilitativa, la cual hace que A llegue a la salida. La operación de inhibición es una importante aplicación de las compuertas AND.

Resumen de la operación AND :

1. La operación AND se ejecuta exactamente en la misma forma que la multiplicación ordinaria de 1's y 0's.

2. Una salida igual a 1 ocurre sólo en el único caso donde todas las entradas son 1.

3. La salida es 0 en cualquier caso donde una o mas entradas son 0.

1.2.3. EL CIRCUITO NOT (INVERSOR).

La figura 1.7 muestra el símbolo de un circuito NOT, al cual se le llama mas comunmente INVERSOR (INVERTER).

FIG. 1.7 CIRCUITO NOT.

Este circuito siempre tiene sólo una entrada y su nivel lógico de salida siempre es contrario al nivel lógico de esta entrada. La figura 1.8 muestra la forma en que el INVERSOR afecta a una señal de entrada. Invierte la señal de entrada en todos los puntos de la onda.

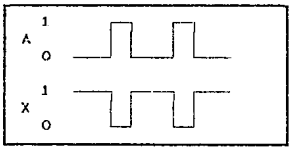


FIG. 1.8

1.2.4 CIRCUITOS QUE CONTIENEN INVERSOR.

Siempre que un inversor se encuentre presente en un diagrama de circuitos logicos, su expresion de salida es simplemente igual a la expresion de entrada con una barra o una comilla sobre ella. La presente figura 1.9 da dos ejemplos utilizando inversor. En el inciso a de la figura mencionada, la entrada A se alimenta a través de un inversor cuya salida es por lo tanto A negada. La salida del inversor se alimenta a una compuerta OR junto con B, de modo que la salida OR sea igual a A negada mas B. Notese que la comilla solo está encima de A, lo cual indica que A se invierte primero y luego se opera con OR con B.

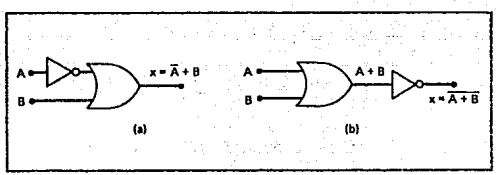


FIG. 1.9 CIRCUITOS QUE UTILIZAN INVERSORES.

En el inciso b de la anterior figura, la salida de la compuerta OR es igual a A + B y se alimenta a través de un INVERSOR. La salida del INVERSOR es por consiguiente igual a A + B negando toda esta salida, ya que invierte la expresion de entrada completa. Nótese que la barra cubre toda la expresion (A+B) esto es importante de distinguir.

1.2.5. COMPUERTAS NOR Y NAND.

Otros dos tipos de compuertas lógicas, NOR y NAND, se utilizan intensamente en los circuitos digitales. Estas compuertas en realidad combinan las operaciones basicas AND, OR y NOT, las cuales facilitan su descripcion mediante operaciones de algebra booleana.

1.2.5.1 LA COMPUERTA NOR

El simbolo de una compuerta NOR de 2 entradas se muestra

en la figura 1.10 :

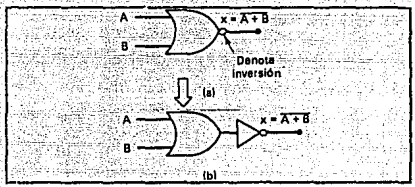


FIG. 1.10 COMPUERTA NOR.

Es el mismo que el simbolo de la compuerta OR excepto que tiene un circulo pequeño en la salida. Este simbolo representa la operacion de inversion. De este modo, la compuerta NOR opera como una compuerta OR seguida de un INVERSOR, de manera que los circuitos de la anterior figura y la que se muestra a continuación son equivalentes a la expresion de salida para la compuerta NOR.

		OR	NOR
A	B	A+B	(A+B)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

FIG. 1.11 TABLA DE VERDAD PARA NOR.

La tabla de verdad de la figura anterior muestra que la salida de la compuerta NOR es la inversa exacta de la salida de la compuerta OR en todas las posibles condiciones de entrada. En tanto que la salida de una compuerta OR se torna ALTA cuando cualquier entrada es ALTA, la salida de la compuerta NOR se torna BAJA cuando cualquier entrada es ALTA. Esta misma operacion se puede aplicar a las compuertas NOR con mas de 2 entradas.

Ejemplo : Determine la forma de onda en la salida de una compuerta NOR para las ondas de entrada que se muestran en la figura 1.12.

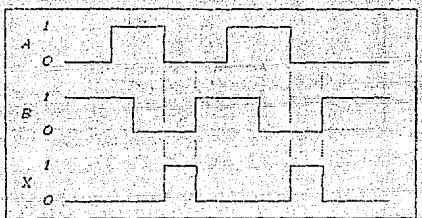


FIG 1.12

Solución : Existen varias maneras de determinar la onda de salida de la NOR. Una de ellas consiste primero en obtener la onda de salida de OR y luego de invertirla (cambiar todos los 1's por 0's y viceversa). Otra forma hace uso del hecho que la salida de una compuerta NOR será ALTA sólo cuando todas las entradas sean bajas. Así, uno puede examinar las ondas de entrada, hallar aquellos intervalos de tiempo donde todas sean BAJAS y hacer que la salida de la NOR se ALTA en esos intervalos. La salida de la NOR será BAJA en todos los otros intervalos de tiempo. La onda de salida resultante se muestra en la figura 1.12.

1.2.5.2. LA COMPUERTA NAND.

El símbolo correspondiente a una compuerta NAND de 2 entradas es el que se muestra en la figura 1.13.

Es el mismo que el de la compuerta AND, excepto por el pequeño círculo en su salida. Una vez mas, este círculo denota la operación de inversión. De este modo, la compuerta NAND opera igual que la AND seguida de un INVERSOR, de manera que los circuitos de la figura 1.13 y la figura 1.14 son equivalentes a la expresión de salida de la compuerta NAND.

AND NAND

A	B	AB	(AB)'
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

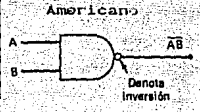


FIG. 1.13 COMPUERTA NAND DE DOS ENTRADAS.

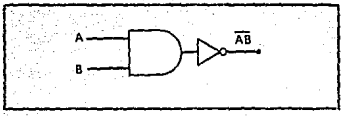


FIG. 1.14 CIRCUITO EQUIVALENTE.

La tabla de verdad de la compuerta NAND, muestra que la salida de esta compuerta es la inversa exacta de la compuerta AND en todas las posibles condiciones de entrada. La salida de AND se torna ALTA solo cuando todas las entradas son ALTAS, en tanto que la salida de NAND se vuelve BAJA solo cuando todas las entradas son ALTAS. Esta misma característica se observa en las compuertas NAND que tienen mas de 2 entradas.

Ejemplo : Determine la forma de onda de salida de una compuerta NAND que tiene las entradas que se muestran continuacion :

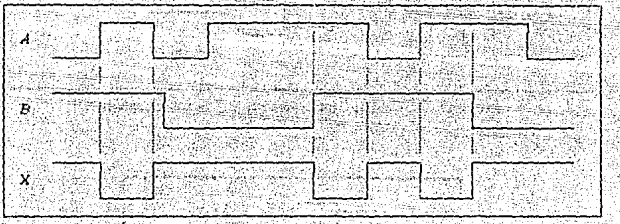


FIG. 1.15

Solución : La salida se puede determinar en varias formas. Una manera consiste en trazar primero la salida de una compuerta AND y luego invertirla. Otra manera aplica el hecho que una salida NAND será BAJA cuando todas las entradas sean ALTAS. De este modo, podemos determinar aquellos intervalos de tiempo durante los cuales todas las entradas sean ALTAS y hacer que la salida de NAND sea BAJA en esos intervalos. La salida será ALTA en el resto de la ocaciones.

1.3. FUNCIONES Y ALGEBRA BOOLEANA.

El algebra booleana difiere de manera importante del algebra ordinaria en que las constantes y variables booleanas solo pueden tener dos posible valores, 0 o bien 1. Una variable boolena es una cantidad que puede, en diferentes ocaciones, ser igual a 0 o a 1.

El algebra booleana se utiliza para expresar los efectos que los diversos circuitos digitales ejercen sobre las entradas logicas y para manipular variables logicas con el objeto de determinar el mejor metodo de ejecucion de cierta funcion de un circuito. En lo sucesivo emplearemos simbolos alfabeticos para representar las variables lonicas. Por ejemplo, A podria

representar cierta entrada o salida de un circuito digital y en cualquier instante debemos tener A = 0 o bien A = 1 ; si no es uno, entonces es cero.

Ya que solo puede haber dos valores, el álgebra booleana es relativamente facil de manejar en comparacion con la ordinaria. En el algebra booleana no hay fracciones, decimales, numeros negativos, raices cuadradas, raices cubicas, logaritmos, numeros imaginarios, etc. De hecho, en el algebra booleana solo existen tres funciones basicas :

1. Adición lógica, también llamada adición OR o simplemente operacion OR. El simbolo comun de esta operacion es el signo mas (+).

2. Multiplicación lógica, denominada así mismo multiplicación AND o simplemente operacion AND. El símbolo comun de esta operacion es el signo de multiplicación (.).

3. Complementación o inversión lógica, denominada también operacion NOT. El símbolo comun de esta operacion es la barra elevada (¯) o la comilla (').

Apartir de estas definiremos :

4. Funcion NOR.

5. Función NAND.

6. Funcion OR-EXCLUSIVA.

7. Función NOR-EXCLUSIVA.

Con las funciones anteriores se desarrolla una multitud de aplicaciones de circuitos lógicos, a los que se denomina COMBINATORIOS, existiendo otro tipo de circuitos logicos que tambien hacen uso de dichas funciones logicas agregando elementos de memoria (que pueden construirse con las funciones logicas basicas) y que se denominan circuitos logicos SECUENCIALES.

1.3.1. FUNCION OR.

Esta funcion tambien es conocida como suma logica, reunion, alternacion y es una funcion que se realiza entre dos variables logicas A y B, denotada como A+B, la cual existe cuando existe alguna de las variables de entrada o ambas.

La forma usual de definirla es mediante su tabla de verdad, la cual es una forma de representacion de una funcion logica, en la que se indica el valor "1" o "0" que toma la funcion para cada una de las combinaciones posibles de las cuales depende.

Asi, si se tienen dos variables A y B, se defino la funcion OR, denotada por A+B, como una funcion cuya tabla de verdad es :

A	B	A+B
1	1	1
1	0	1
0	1	1
0	0	0

TABLA 1.2 TABLA DE VERDAD DE LA FUNCION OR.

El siguiente circuito con interruptores, realiza la funcion OR, en el cual la lampara se enciende (L=1) cuando algunos de los interruptores esta cerrado, o bien los dos.

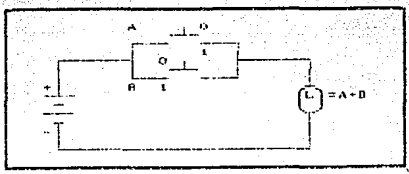


FIG. 1.10 CIRCUITO DE INTERRUPTORES EQUIVALENTE.

1.3.2. FUNCION AND.

Esta funcion es conocida con los nombres de funcion "y",

producto lógico, conjunto o intersección.

Su tabla de verdad esta dada por :

A	B	A.B
1	1	1
1	0	0
0	1	0
0	0	0

TABLA 1.3 TABLA DE VERDAD PARA LA FUNCION AND.

Observamos que la función existe solo cuando las dos variables A y B existen al mismo tiempo.

El circuito que realiza la función AND es el siguiente. La lampara encendida (L=1) cuando los interruptores A y B estan cerrados.

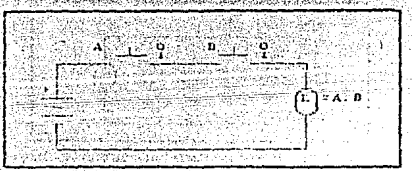


FIG. 1.17 CIRCUITO DE INTERRUPTORES EQUIVALENTE A LA FUNCION AND.

Al observar la tabla, se advierte que la multiplicación AND es exactamente lo misma que la multiplicacion ordinaria. Siembre que A y B sean cero, su producto es cero; cuando A y B son 1, su producto es 1.

La expresión X = A.B se lee "X es igual A y B". El signo de multiplicación se omite por lo general como en el aritmética ordinaria, de modo que la expresión se transforma en X = AB. Lo mas importante que debe recordarse es que la operacion AND es la misma que la operación ordinaria de la multiplicacion, donde las variables pueden ser 0 o bien 1.

1 . 3 . 3 . FUNCION NEGACION.

La funcion NEGACION se denomina complemento, funcion NO, inversor y es denotada por A' (se lee NO A) tal que su salida es el valor contrario a A. La tabla de verdad para la funcion negacion es :

A	A'
0	1
1	0

TABLA 1.4 TABLA DE VERDAD DE LA FUNCION NEGACION.

El circuito mas sencillo que realiza la negacion es el siguiente que emplea un interruptor normalmente cerrado, el cual cuando no se acciona (A=0) permite el paso de corriente a la lampara para la cual enciende (L=1) y si se acciona (A=1) se abre el circuito apagandose la lampara (L=0).

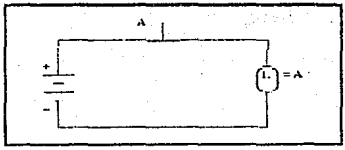


FIG. 1.10 CIRCUITO DE INTERRUPTORES EQUIVALENTE.

La operacion NO o NOT difiere de las operaciones OR y AND en que esta puede efectuarse con una sola variables de entrada.

1 . 3 . 4 . FUNCION NOR.

La funcion NOR, tambien llamada "o negada", es una funcion que se denota (A+B)' y cuya tabla de verdad esta dada por:

Observamos que la compuerta NOR es la negacion de la compuerta OR, la cual puede ser relizada con interruptores, segun se muestra.

A	B	A+B	(A+B)'
1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

TABLA 1.5 TABLA DE VERDAD PARA LA FUNCION NOR.

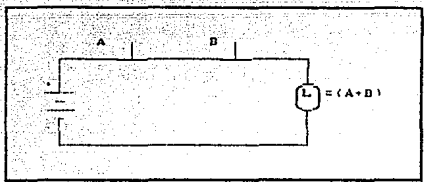


FIG. 1.19 CIRCUITO DE INTERRUPTORES EQUIVALENTE.

1.3.5. FUNCION NAND.

Esta función lógica está definida por la tabla de verdad siguiente :

		AND	NAND
A	B	A.B	(A.B)'
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

TABLA 1.6 TABLA DE VERDAD PARA LA FUNCION NAND.

Observamos que la función NAND es la negación de la compuerta AND.

La función NAND se puede realizar con interruptores de la siguiente manera :

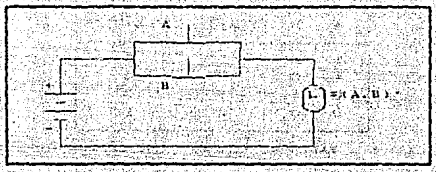


FIG. 1.20 CIRCUITO DE INTERRUPTORES EQUIVALENTE A LA FUNCION NAND.

1.3.6. FUNCION OR-EXCLUSIVA (XOR).

Esta función se denota por A⊕B, y existe cuando A existe o B existe, pero no ambas.

La tabla de verdad para esta función es :

A	B	A⊕B
1	1	0
1	0	1
0	1	1
0	0	0

TABLA 1.7 TABLA DE VERDAD PARA LA FUNCION XOR.

Observamos que si A y B son iguales la salida es falsa (0) y si son diferentes la salida es verdadera (1).

Su símbolo más conocido es el Americano, el cual se muestra en la figura siguiente:

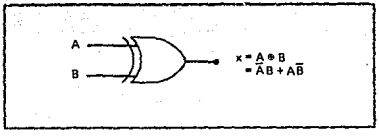


FIG. 1.21 COMPUERTA XOR.

Verificando por tabla lo anterior es verdad (es decir A⊕B = AB'+A'B.

Las caracteristicas de una compuerta XOR se resumen como sigue:

. Solo tiene dos entradas y su salida es:

$$X = AB'+A'B = A \oplus B$$

. Su salida es ALTA sólo cuando las dos entradas están en niveles diferentes.

1.3.7. FUNCION NOR-EXCLUSIVO.

La funcion NOR-EXCLUSIVO (que se abrevia EX-NOR) opera completamente al contrario que el XOR. La figura 1.22 muestra el circuito correspondiente a esta funcion junto con su respectiva tabla de verdad :

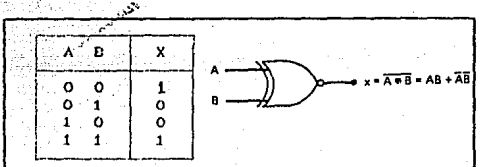


FIG. 1.22 FUNCION NOR-EXCLUSIVO.

La expresion de salida es X = AB + A'B' lo cual indica, junto con la tabla de verdad, que X sera uno en dos casos : A=B=1 (el termino AB) y A=B=0 (el termino A'B'). En otras palabras, este circuito produce una salida ALTA siempre que las dos entradas estan en el mismo nivel.

Debe estar claro que la salida del circuito EX-NOR es la inversa exacta del circuito XOR. El simbolo de la compuerta EX-NOR se obtiene simplemente agregando un pequeño circulo en la salida del simbolo XOR.

La compuerta EX-NOR se resume como sigue :

. Solo tiene dos entradas y su salida es :

$$X = AB + A'B' = (A \oplus B)'$$

. Su salida es ALTA sólo cuando las dos entradas están en el mismo nivel.

1.4. TEOREMAS FUNDAMENTALES DEL ALGEBRA BOOLEANA

Continuaremos nuestro estudio del algebra booleana investigando los diversos teoremas booleanos (reglas) que nos pueden servir para simplificar las expresiones y los circuitos logicos.

1.4.1. TEOREMAS CON UNA VARIABLE

El primer grupo de teoremas se muestra en la tabla 1.8.

En cada teorema, X es una variable logica que puede ser un 0 o bien un 1. Cada teorema se presenta con un diagrama de circuito logico que demuestra su validez.

El teorema (1) enuncia que, si cualquier variable se opera con AND con 0, el resultado tiene que ser 0. Esto es facil de recordar puesto que la operacion AND es como la multiplicacion ordinaria, donde sabemos que cualquier numero que se multiplica por 0 es 0. Tambien sabemos que la salida de una compuerta AND sera 0 siempre que cualquier entrada sea 0, independientemente del nivel de la otra entrada.

El teorema (2) es asi mismo obvio por la comparacion que se hace de el con la multiplicacion ordinaria.

El teorema (3) se puede demostrar ensayando cada caso. Si X=0, entonces 0.0 = 0; si X=1, entonces 1.1 = 1. Asi X.X = X.

El teorema (4) se puede probar en la misma forma. Sin embargo,

también puede razonarse que en cualquier instante, X o bien su inversa, tiene que estar en el nivel 0, de modo que su producto AND tiene siempre que ser 0.

El teorema (5) es directo ya que 0, sumado a cualquier numero, no altera su valor en la suma comun o en la adicion OR.

El teorema (6) afirma que, si cualquier variable se opera con OR con 1, el resultado siempre sera 1. Verificamos esto con ambos valores de X : 0+1 = 1 y 1+1 = 1. Equivalentemente, podemos recordar que la salida de una compuerta OR sera 1 cuando cualquier entrada sea 1, sin importar que valor tenga la otra.

El teorema (7) puede demostrarse verificando los dos valores de X : 0+0 = 0 y 1+1=1.

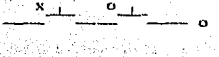
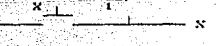
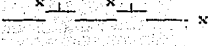
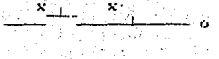
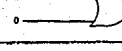
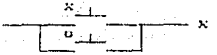
TEOREMA	CIRCUITO LOGICO	CIRCUITO CON INTERRUPTORES
(1) X.0=0		
(2) X.1=X		
(3) X.X=X		
(4) X.X'=0		
(5) X+0=X		

TABLA 1.8 TEOREMAS CON UNA VARIABLE.

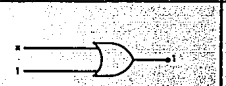
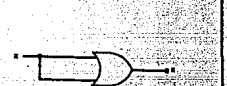
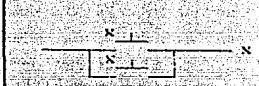
TEOREMA	CIRCUITO LOGICO	CIRCUITO CON INTERRUPTORES
(6) X+1=1		
(7) X+X=X		
(8) X+X'=1		

TABLA 1.8 TEOREMAS CON UNA VARIABLE (CONT.).

El teorema (8) se puede probar en forma similar o simplemente podemos razonar que en cualquier instante X o bien X inversa o negada tiene que estar en el nivel 1 de manera que siempre operemos con OR un 0 y un 1, que siempre da como resultado 1.

Antes de presentar mas teoremas, debemos indicar que al aplicar los teoremas (1) a (8), la variable X puede en realidad representar una expresion que contiene mas de una variable.

1.4.2. TEOREMAS CON MULTIPLES VARIABLES.

Los teoremas que se presentan en la tabla 1.9 implican mas de una variable.

Los teoremas (9) y (10) se denominan leyes conmutativas. Estas leyes indican que el orden en el cual operamos con OR y AND dos variables es intrascendente; el resultado es el mismo.

Los teoremas (11) y (12) son las leyes asociativas, las cuales afirman que podemos agrupar las variables en una expresion AND o en una OR en la forma que se desee.

El teorema (13) es la ley distributiva, la cual afirma que una

expresión puede aplicarse multiplicando término a término, como en el algebra ordinaria. Este teorema indica así mismo que podemos factorizar una expresión. Es decir, si tenemos una suma de dos o mas terminos, cada uno de los cuales contiene una variables común, esta se puede factorizar como en el algebra ordinaria. Por ejemplo, si tenemos la expresión AB'C + A'B'C', podemos factorizar la variable B' :

$$AB'C + A'B'C' = B'(AC + A'C')'$$

Para poner otro ejemplo, consideremos la expresión ABC + ABD. Aquí los dos terminos tiene las variables A y B en común, de manera que AB se puede factorizar en ambos terminos. Esto es,

$$ABC + ABD = AB(C + D)$$

(9)	X+Y=Y+X
(10)	XY=YX
(11)	X+(Y+Z)=(X+Y)+Z=X+Y+Z
(12)	X(YZ)=(XY)Z=XYZ
(13a)	X(Y+Z)=XY+XZ
(13b)	(W+X)(Y+Z)=WY+XY+WZ+XZ
(14)	X+XY=X
(15)	X+X'Y=X+Y

TABLA 1.9 TEOREMAS CON MULTIPLES VARIABLES.

Los teoremas (9) a (13) se pueden recordar facilmente y son de uso sencillo, ya que son identicos a los del algebra ordinaria. En cambio, los teoremas (14) y (15) no tiene equivalentes en el algebra ordinaria. Cada uno se puede demostrar ensayando todos los casos posibles para X y Y. Esto se ilustra para el teorema (14) como sigue :

Caso 1: Para X=0, Y=0,
X + XY = X
0 + 0.0 = 0
0 = 0

Caso 2 : Para X=0, Y=1,
X + XY = X
0 + 0.1 = 0
0 + 0 = 0
0 = 0

Caso 3 : Para X=1, Y=0,
X + XY = X
1 + 1.0 = 1
1 + 0 = 1
1 = 1

Caso 4 : Para X=1, Y=1,
X + XY = X
1 + 1.1 = 1
1 + 1 = 1
1 + 1 = 1

El teorema (14) también se puede demostrar factorizando y usando los teoremas (6) y (2) como sigue :

X + XY = X (1 + Y)
= X.1 (usando el teorema (6))
= X (usando el teorema (2))

Todos estos teoremas booleanos pueden ser de utilidad para simplificar una expresion logica; es decir, al reducir el numero de terminos de la expresion. Cuando se hace esto, la expresión reducida produce un circuito menos complejo que el que la expresion original habria generado. Los siguientes ejemplos sirviran para ilustrar la forma en que se pueden aplicar los teoremas booleanos.

Ejemplo : Simplifique la expresión $Y = AB'D + AB'D'$

Solución : Factorice las variables comunes AB' *utilizando el teorema (13) :*

$$Y = AB'(D' + D)$$

Utilizando el teorema (8), el término entre parentesis es equivalente a 1. De este modo,

$$Y = AB'.1$$
$$Y = AB' \quad \text{(usando el teorema (2))}$$

Ejemplo : Simplifique $Z = (A' + B)(A + B)$

Solucion : La expresión se puede desarrollar multiplicando los terminos (teorema (13)).

$$Z = A'A + A'B + BA + BB$$

Al invocar el teorema (4), el termino $A'A = 0$ asi mismo, $BB = B$ (teorema(3)).

$$Z = 0 + A'B + BA + B$$
$$Z = A'B + AB + B$$

Al factorizar la variable B (teorema(13)), tenemos

$$Z = B(A' + A + 1)$$

Finalmente, al usar el teorema (6), tenemos

$$Z = B$$

1.4.3 TEOREMAS DE MORGAN.

Dos de los teoremas más importantes del álgebra booleana fueron enunciados por el eminente matematico DeMorgan. Los teoremas de DeMorgan son de extrema utilidad en la simplificación de expresiones en las cuales se invierte un producto o suma de variables. Los dos teoremas son :

(16) $(X + Y)' = X' \cdot Y'$
(17) $(X \cdot Y)' = X' + Y'$

El teorema (16) afirma que cuando se invierte la suma OR de dos variables, esta inversion es la misma que la de cada variables en forma individual y luego la operacion con AND de estas variables invertidas. El teorema (17) expresa que, cuando se invierte el producto AND de dos variables, esto equivale a invertir cada variable en forma individual y luego operarlas con OR. Cada uno de los teoremas de DeMorgan se puede demostrar rapidamente verificando todas las posibles combinaciones de X y Y.

Aunque estos teoremas se han enunciado en terminos de variables sencillas X y Y, son igualmente validos en situaciones donde X y/o Y son expresiones que contienen mas de una variable. Por ejemplo, apliquemoslos a la expresion (AB' + C)' como se muestra a continuacion :

$$(AB' + C)' = (AB')'.C'$$

Nótese que tratamos a AB' como X y a C como Y. El resultado se puede simplificar todavia mas ya que tenemos un producto AB' que se invierte. Al utilizar el teorema (17), la expresion se transforma en :

$$(AB')'.C' = (A' + B'')'.C'$$

Nótese que podemos reemplazar B'' por B y asi, tenemos finalmente :

$$(A' + B).C' = A'C' + BC'$$

Este resultado final contiene solamente signos inversores que invierten una sola variable.

Ejemplo : *Simplifique la expresion*

$$Z=(A'+C).(B+D')'$$

Solución : Utilizando el teorema (17) podemos reescribir esto como

$$Z = (A' + C)' + (B + D')'$$

Podemos considerar dividir el signo INVERSOR mayor por la mitad y cambiar el signo AND (.) por un signo OR (+). Ahora bien, el termino (A' + C)' se puede simplificar aplicando el teorema (16). De igual manera, (B + D')' se puede simplificar.

$$Z = (A' + C)' + (B + D')'$$
$$= (A'C') + (B'D'')$$

Aqui hemos dividido los signos de inversion mayores a la mitad y sustituido el (+) por un (.). Al cancelar las inversiones dobles, tenemos por último

$$Z = AC' + B'D$$

El ejemplo anterior señala que, cuando se utilizan los teoremas de DeMorgan para reducir una expresion, un signo INVERSOR se puede dividir en cualquier punto de la expresion y el operador en el punto en la expresion se cambia por su contrario (+ se cambia por . y viceversa). Este procedimiento continua hasta que la expresion se reduce a una, en la cual solo se invierten variables individuales.

Los teoremas de DeMorgan se aplican fácilmente a mas de dos variables. Por ejemplo, se puede demostrar que :

$$(X + Y + Z)' = X'Y'Z'$$

$$(XYZ)' = X' + Y' + Z'$$

y de esta misma manera para mas variables. Una vez más, debemos entender que cualquiera de estas variables puede ser una expresion en vez de una sola variable.

1.5 FORMAS ESTANDAR DE LAS FUNCIONES BOOLEANAS.

Una variable binaria puede aparecer ya sea en forma normal (X) o en su forma complementaria (X'). Ahora consideremos dos variables binarias X y Y combinadas con operador AND. Ya que cada variable puede aparecer en cualquier forma, hay cuatro combinaciones posibles : X'Y', X'Y, XY' y XY. Cada uno de esos cuatro terminos AND representa un miniterminо o producto estandar. En forma semejante pueden combinarse N variables para formar 2 a la N miniterminos. Los 2 a la N miniterminos diferentes pueden determinarse por un método similar al que se muestra en la tabla 1.10 para tres variables.

X	Y	Z	MINITERMINOS TERMINO	DESIGNACION	MAXTERMINOS TERMINO	DESIGNACION
0	0	0	X'Y'Z'	m0	X+Y+Z	M0
0	0	1	X'Y'Z	m1	X+Y+Z'	M1
0	1	0	X'YZ'	m2	X+Y'+Z	M2
0	1	1	X'YZ	m3	X+Y'+Z'	M3
1	0	0	XY'Z'	m4	X'+Y+Z	M4
1	0	1	XY'Z	m5	X'+Y+Z'	M5
1	1	0	XYZ'	m6	X'+Y'+Z	M6
1	1	1	XYZ	m7	X'+Y'+Z'	M7

TABLA 1.10 MINITERMINOS Y MAXTERMINOS.

Los numeros binarios desde 0 a 2^N-1 se listan bajo las N variables. Cada minitermino se obtiene del termino AND de las N variables, con cada variables vuelta prima si el bit correspondiente del numero binario es un 0 y no prima si es un 1. En la tabla 1.10 tambien se muestra un simbolo para cada minitermino y esta en forma "m_j", donde j indica el equivalente decimal del numero binario del minitermino en cuestion.

De manera semejante, N variables forman un termino OR, con cada variable vuelta prima o no prima, proporcionando 2 a la N combinaciones posibles, denominadas maxterminos o sumas estandar. Los ocho maxterminos para tres variables, junto con su denotacion simbolica, se ilustran en la misma tabla mencionada. Cualesquiera 2 a la N maxterminos para N variables pueden determinarse en forma similar. Cada maxtermino se obtiene de un termino OR de las N variables, con cada variable no prima si el bit correspondiente es 0 y prima si es un 1. Observese que cada maxtermino es el complemento de su mintermino correspondiente y viceversa.

Una funcion booleana puede expresarse en forma algebraica mediante una tabla de verdad dada, formando un minitermino para cada combinacion de variables que producen un 1 en la funcion y, tomando entonces los OR de todos esos terminos. Por ejemplo, la funcion F1 en la tabla 1.11 se determina al expresar las combinaciones 001, 100 y 111 como X'Y'Z, XY'Z' y XYZ, respectivamente.

X	Y	Z	F1	F2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

TABLA 1.11

Ya que cada uno de estos minterminos resulta en F1 igual a 1, se debe tener :

$$F1 = X'Y'Z + XY'Z' + XYZ = m1 + m4 + m7$$

De manera semejante, puede verificarse con facilidad que :

F2 = X'YZ + XY'Z + XYZ' + XYZ = m3 + m5 + m6 + m7

Estos ejemplos muestran una propiedad importante del algebra booleana : cualquier funcion booleana puede expresarse como una suma de mintermnios.

Ahora considerese el complemento de una funcion booleana, a partir de la tabla de verdad puede leerse al formar un mintermino para cada combinacion que produce un 0 en la función y aplicando el operador OR a esos términos, el complemento de F1 se lee como :

F1' = X'Y'Z' + X'YZ' + X'YZ + XY'Z + XYZ'

si se toma el complemento de F1', se obtiene la funcion F1 :

F1 = (X+Y+Z)(X+Y'+Z)(X+Y'+Z')(X'+Y+Z')(X'+Y'+Z)
= M0 . M2 . M3 . M5 . M6

Este ejemplo demuestra una segunda propiedad importante del algebra booleana : cualquier funcion booleana puede expresarse como un producto de maxtermnios. El procedimiento para obtener el producto de los maxtermnios en forma directa de la tabla de verdad es como sigue : formese un maxtermino para cada combinacion de las variables que produce un 0 en la funcion, y entonces formese AND de todos los maxterminos. Las funciones booleanas expresadas como una suma de mintermnios o producto de maxtermnios se dice que estan en forma canonica.

Las dos formas canonicas del algebra booleana muy rara vez son las que tiene el menor numero de literales, debido a que cada mintermino y maxtermino debe contener, por definicion, todas las variables ya sea complementadas o sin complementar.

Otra forma de expresar las funciones booleanas es la forma estandar. En esta configuracion, los términos que forman la funcion pueden contener uno, dos o cualquier numero de literales. Hay dos tipos de formas estandar : la suma de productos y el producto de sumas.

La suma de productos es una expresión booleana que contiene

terminos AND, llamados terminos producto, de una o mas literales cada uno. La suma denota la operacion OR de esos terminos. Un ejemplo de una funcion expresada en suma de productos es :

$$F1 = Y' + XY + X'YZ'$$

La funcion tiene tres terminos producto de una, dos y tres literales cada uno, respectivamente. Su suma es, en efecto, una operacion OR.

Un producto de sumas es una expresion booleana que contiene terminos OR, llamados terminos suma. Cada termino puede tener cualquier numero de literales. El producto denota la operacion AND de esos terminos. Un ejemplo de una funcion expresada en producto de sumas es :

$$F2 = X(Y' + Z)(X' + Y + Z' + W)$$

Esta expresion tiene tres terminos de suma de una, dos, tres y cuatro literales cada uno. El producto es una operacion AND.

Una expresion booleana puede expresarse en una forma no estandar. Por ejemplo la funcion :

$$F3 = (AB + CD)(A'B' + C'D')$$

no es una suma de productos ni un producto de sumas. Puede cambiarse a una forma estandar usando la ley distributiva para eliminar los parentesis :

$$F3 = A'B'CD + ABC'D'$$

1.6. FORMAS DUALES.

El principio de dualidad establece que cada expresion algebraica deducida de los postulados del algebra booleana permanece valida si los operadores y los elementos identidad se intercambian. El principio de dualidad tiene muchas aplicaciones. Si se desea el dual de una expresion algebraica, simplemente se intercambian los operadores OR y AND y se reemplazan los 1's por

0's y los 0's por 1's.

El complemento de una función F es F' y se obtiene por el intercambio de numeros 0 a numeros 1 y de numeros 1 a numeros 0 en el valor de F. El complemento de una función puede derivarse en forma algebraica mediante el teorema de DeMorgan. Estos teoremas pueden ampliarse a tres o mas variables. La forma de tres variables de los teoremas de DeMorgan se muestran a continuación :

$$(A + B + C)' = A'B'C'$$

$$(ABC) = A' + B' + C'$$

La forma generalizada del teorema de DeMorgan enuncia que el complemento de una función se obtiene por el intercambio de los operadores AND y OR y complementando cada literal.

Ejemplo : Encuentre el complemento de las funciones F1 = X'YZ' + X'Y'Z y F2 = X(Y'Z' + YZ).

Solución : Se aplican los teoremas de DeMorgan cuantas veces sea necesario y se obtienen los complementos como sigue :

$$F1' = (X'YZ' + X'Y'Z)'$$
$$= (X'YZ')'(X'Y'Z)'$$
$$= (X + Y' + Z)(X + Y + Z')$$

$$F2' = (X(Y'Z' + YZ))$$
$$= X' + (Y'Z' + YZ)'$$
$$= X' + (Y'Z')'.(YZ)'$$
$$= X' + (Y + Z)(Y' + Z')$$

Un procedimiento mas simple para derivar el complemento de una funcion es tomar la dual de la funcion y complementar cada literal. Este metodo se sigue del teorema generalizado de DeMorgan. Recuerdese que la dualidad de una función se obtiene por el intercambio de los operadores AND y OR y los 1's y los 0's.

Ejemplo : Obtengase el complemento de las funciones del ejemplo anterior empleando el principio de dualidad

Solucion :

$F1 = X'YZ' + X'Y'Z$

La dual de F1 es (X' + Y + Z')(X' + Y' + Z)
Complemento de cada literal :

$F1' = (X + Y' + Z)(X + Y + Z')$

$F2 = X(Y'Z' + YZ)$

La dual de F2 es X + (Y' + Z')(Y + Z)
Complemento de cada literal :

$F2' = X' + (Y + Z)(Y' + Z')$

1.7. NIVELES DE VOLTAJE Y SU RELACION CON LAS VARIABLES LOGICAS.

En los sistemas digitales, la informacion que está siendo procesada por lo general se presenta en forma binaria. Las cantidades binarias se pueden representar por medio de cualquier dispositivo que solamente tenga dos estados de operacion o posibles condiciones. Por ejemplo, un interruptor tiene solo dos estados : abierto o cerrado. Arbitrariamente, podemos hacer que un interruptor abierto represente el 0 binario y que uno cerrado represente el 1 binario. Con esta asignacion podemos representar ahora cualquier numero binario como se ilustra en la figura 1.23, donde los estados de cinco interruptores representan la cantidad 10010.

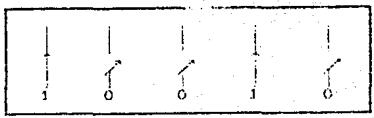


FIG. 1.23 USO DE INTERRUPTORES.

Otro ejemplo de lo anterior se muestra en la figura 1.24, donde se utilizan agujeros perforados en papel para representar números binarios. Un agujero perforado es un 1 binario y la ausencia de un agujero es un 0.

FIG. 1.24 USO DE CINTA DE PAPEL PERFORADA.

Hay muchos otros dispositivos que solamente tienen dos estados de operación o bien que pueden operarse en dos condiciones extremas, entre estos se encuentran : la bombilla eléctrica (encendida o apagada), el diodo (conductor o no conductor), relevador (energizado o desenergizado), etc.

En los circuitos electrónicos digitales, la información binaria se representa por medio de voltajes (o corrientes) que están presentes en las entradas o en las salidas de los diversos circuitos. Comunmente, el 0 y el 1 binarios se representan con dos niveles de voltajes nominales. Por ejemplo, cero volts (0 V) podría representar el 0 binario y +5 V, el 1 binario. En realidad, debido a las variaciones del circuito, el 0 y el 1 se representan con por medio de intervalos de voltaje. Esto se ilustra en la figura 1.25 :

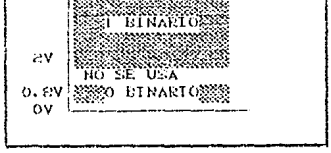


FIG. 1.25 ASIGNACIONES COMUNES DE VOLTAJE EN EL SISTEMA DIGITAL.

donde cualquier voltaje entre 0 y 0.8 V representa un 0 y

cualquier voltaje entre 2 y 5 V representa un 1. Todas las señales de entrada y salida se clasificaran en cualquiera de estos intervalos. La figura 1.26 muestra una forma de una señal digital comun cuando forma una secuencia a traves del valor binario 01010.

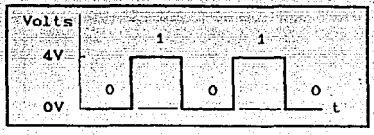


FIG. 1.26 SEÑAL DIGITAL COMUN.

Ahora podemos observar otra diferencia significativa entre los sistemas digitales y los analógicos. En los sistemas digitales, el valor exacto de un voltaje no es importante; por ejemplo, un voltaje de 3.6 V es los mismo que un voltaje de 4.3 V. En los sistemas analógicos, el valor exacto de un voltaje si es importante. Por ejemplo, si el voltaje analógico es proporcional a la temperatura registrada por un transductor, 3.6 V representaria una temperatura diferente de la que representaria 4.3 V. En otras palabras, el valor del voltaje conlleva informacion significativa. Esta caracteristica implica que el diseño de circuitos analogicos precisos es generalmente mas complicado que el de circuitos digitales.

Los circuitos digitales estan diseñados para producir voltajes de salida que se clasifican dentro de los intervalos de voltaje prescritos 0 y 1. De igual manera, los circuitos digitales se diseñan para responder predeciblemente a voltajes de entrada que se encuentran dentro de los intervalos definidos 0 y 1. Lo que esto significa es que un circuito digital respondera de la misma manera a todos los voltajes de entrada que se clasifiquen dentro del intervalo 0 admitido; en forma semejante, tampoco distinguira entre voltajes de entrada que entren en el intervalo 1 permitido.

Para ilustrar lo anterior, la figura 1.27 representa un circuito digital comun con entrada Vi y salida Vo. La salida corresponde a dos diferentes señales de entrada. Observese que Vo es igual en ambos casos debido a que las dos formas de onda de entrada, en tanto que difieren en sus niveles exactos de voltaje, estan en los mismos niveles binarios.

La forma en la cual un circuito digital responde a una entrada se conoce como logica del circuito. Cada tipo de circuito digital

obedece a cierto conjunto de reglas lógicas. Por esta razón, los circuitos digitales se denominan a sí mismos circuitos lógicos. Los dos términos se utilizarán indistintamente.

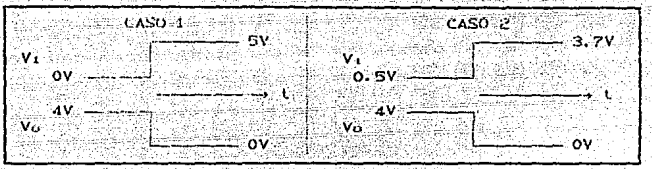


FIG. 1.22 UN CIRCUITO DIGITAL RESPONDE A UN NIVEL BINARIO DE ENTRADA Y NO A SU VOLTAJE REAL.

1.8. TRANSFORMACION DE EXPRESIONES BOOLEANAS A DIAGRAMAS LOGICOS.

Si la operación de un circuito se define por medio de una expresión booleana, un diagrama de circuito lógico se puede poner en práctica directamente a partir de esa expresión. Por ejemplo, si necesitáramos un circuito que se definiera por X = ABC, inmediatamente sabríamos que todo lo que se requeriría sería una compuerta AND de tres entradas. Si necesitáramos un circuito que se definiera por X = A + B', emplearíamos una compuerta OR de dos entradas con un INVERSOR en una de las entradas. El mismo razonamiento que se aplica en estos casos aislados se puede aplicar también a circuitos más complejos.

Supongamos que se desea construir un circuito cuya salida sea Y = AC + BC' + A'BC. Esta expresión booleana contiene tres términos, los cuales se operan todos con OR. Esto nos indica que se requiere una compuerta OR de tres entradas que sean iguales a AC, BC' y A'BC, respectivamente. Esto se ilustra en la figura 1.23, donde se traza una compuerta OR de tres entradas rotuladas como AC, BC' y A'BC.

Cada entrada de la compuerta OR es un término del producto AND, lo cual significa que una compuerta AND con entradas adecuadas se puede emplear para generar cada uno de estos términos. Esto se muestra en la figura 1.24, que es el diagrama final del circuito. Advierta el uso de INVERSORES para producir los términos A' y C' que se requieren en la expresión.

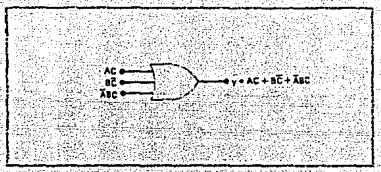


FIG. 1.28

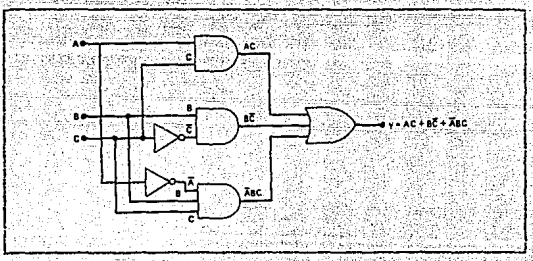


FIG. 1.29

Este mismo enfoque general se puede tomar siempre, aunque encontraremos que existen algunas técnicas más eficientes que pueden emplearse, sin embargo, por ahora se utilizará este método directo.

Ejemplo : Trace el diagrama de circuito que porga en acción la expresión $X = AB + B'C$.

Solución : Esta expresión indica que los términos AB y B'C son entradas en una compuerta OR y cada uno de estos términos es generado a partir de una compuerta AND aparte. El resultado se muestra en la siguiente figura:

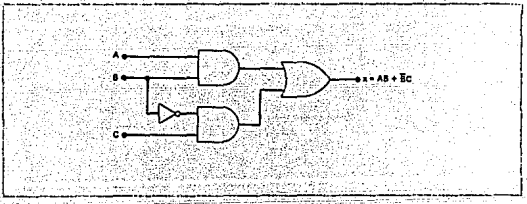


FIG. 1.30

Ejemplo : Ponga en practica el circuito logico que tiene la expresion X = C(AB).(C+D)'.D' utilizando unicamente compuertas NOR y NAND.

Solución : El termino (C+D)' es la expresión que corresponde a la salida de una compuerta NOR. Este termino se opera con AND con A y B y el resultado se invierte; esta, por supuesto, es la expresion NAND. Asi, el circuito se pone en accion como se muestra en la figura siguiente:

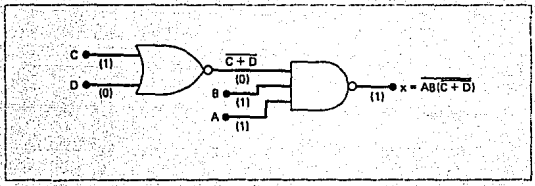


FIG. 1.31

Notese que la compuerta NAND primero opera con AND los terminos A,B y (C+D)' y luego invierte el resultado completo.

Ejemplo : Encuentre el circuito lógico correspondiente a la siguiente expresión :

$$X = CA \oplus CD'CB \oplus D2'$$

Solución : Los términos CA ⊕ CD' y CB ⊕ D2' son las salidas a compuertas EX-NOR, dichas salidas operan en una compuerta AND para obtener el circuito siguiente :

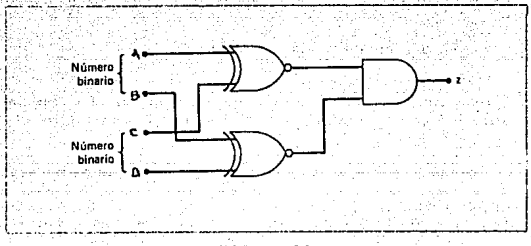
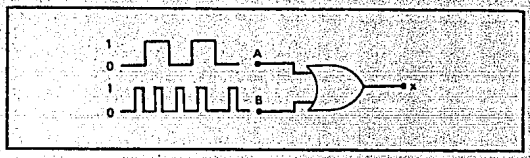


Fig. 1.32

Las formas que pueden ser utilizadas compuertas EX-NOR con inversores para después operar con la compuerta AND.

EJERCICIOS PROPUESTOS.

1.1 Trace la forma de onda de salida para el circuito siguiente:

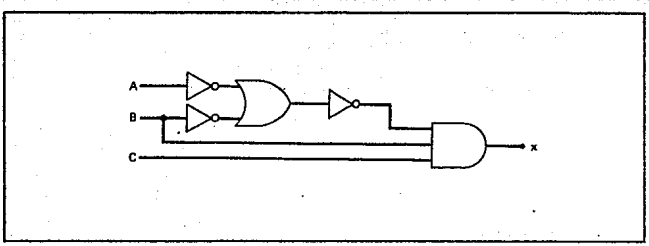


1.2 Cambie la compuerta OR de la figura anterior por una compuerta AND.

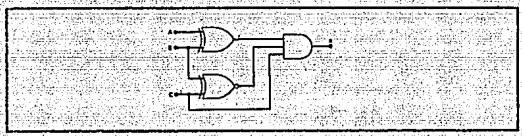
(a) Trace la forma de onda de salida.

(b) Trace la forma de onda de salida si la entrada A se mantiene permanentemente en el nivel mas bajo.

1.3 Escriba una expresión booleana para la salida X de la siguiente figura. Determine el valor de X en todas las posibles condiciones de entrada y enlístelas en una tabla de verdad.



1.4 Determine las condiciones de entrada que se necesitan para para producir X=1 en el circuito que se muestra a continuación:



1.5 Para cada una de las siguientes expresiones, construya el circuito logico correspondiente utilizando compuertas AND y OR, ademas de INVERSOR.

(a) X = (AB(CC + D))'

(b) Z = (A + B + C'DE')' + B'CD'

(c) Y = (M + N)' + P'Q

1.6 Simplifique la siguiente expresion utilizando los teoremas (13b), (3) y (4).

X = (M + N)(M' + P)(N' + P')

1.7 Simplifique la siguiente expresion utilizando los teoremas (13a), (3) y (6).

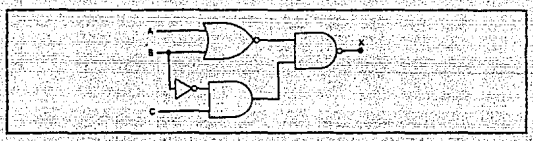
X = A'BC' + ABC' + BC'D

1.8 Simplifique cada una de las siguientes expresiones mediante el uso de los teoremas de DeMorgan.

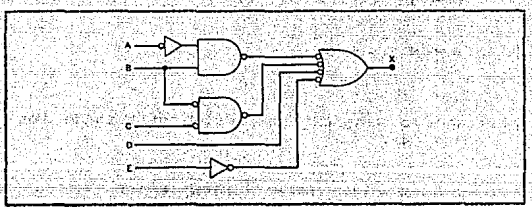
(a) (A'BC')' (b) (A' + B'C)'

(c) CABC(CD)')' (d) (A'B + C')'D'

1.9 Utilice teoremas de DeMorgan para simplificar la salida del siguiente circuito:



1.10 Determine las condiciones de entrada que se necesitan para ocasionar que la salida del siguiente circuito pase a su estado activo.



1.11 Exprese las siguientes funciones en suma de minitérminos y producto de maxtérminos:

(a) X = D(A' + B) + B'D

(b) X = (A' + B)(B' + C)

(c) X = (AB + C)(B + AC)

CAPITULO II : M I N I M I Z A C I O N .

CONTENIDO :

CAPITULO II : MINIMIZACION.

Una vez que la expresión para un circuito lógico se haya obtenido, podemos reducirla a una forma más simple que contenga menos términos o variables en uno o más términos. La nueva expresión puede utilizarse entonces para activar un circuito que sea equivalente al original pero que contenga menos compuertas y conexiones.

Cuando una función booleana se implanta con compuertas lógicas, cada literal en la función denota una entrada a una compuerta, y cada término se implanta con una compuerta. La minimización del número de literales y el número de términos resulta en un equipo con menos circuitos. No siempre es posible minimizar ambos en forma simultánea; por lo común, debe disponerse de más criterios. El número de literales de una función booleana puede minimizarse por manipulaciones algebraicas, desafortunadamente, no hay reglas específicas que seguir que garanticen la respuesta final.

El método de los diagramas OI o mapas de Karnaugh es el más apropiado para simplificar funciones lógicas de dos, tres y cuatro variables de entrada, aunque también puede utilizarse para funciones de cinco y seis variables. Cabe hacer notar que cuando tenemos un número considerable de variables, es conveniente utilizar el método de Quine-McCluskey (método tabular), debido a que es posible programarlo para que lo desarrolle una computadora.

En las siguientes secciones de este capítulo estudiaremos métodos de simplificación de circuitos lógicos. Uno de ellos consiste en utilizar los teoremas del álgebra booleana y, como observaremos depende de la inspiración y experiencia del diseñador. El método de los mapas de Karnaugh es un enfoque sistemático y sencillo y por último el método tabular el cual no es usado comunmente a nivel escolar.

2.1. TEOREMAS DE REDUCCION.

Los teoremas del álgebra booleana que se estudiaron en el capítulo 1 se pueden utilizar para ayudarnos a simplificar la expresión para un circuito lógico. Desafortunadamente, no siempre es obvio que teoremas deben aplicarse a fin de producir el resultado más simple. Además no existe una manera sencilla de indicar si la expresión simplificada se encuentra en su forma más simple o bien si pudiera haberse simplificado todavía más. Así, la simplificación algebraica con frecuencia se convierte en un proceso de ensayo y error. Sin embargo, con experiencia uno puede llegar a obtener resultados razonablemente exactos.

Los siguientes teoremas se emplean frecuentemente en la reducción de funciones lógicas y se pueden demostrar a través de las leyes del algebra booleana o tablas de verdad.

(1) $AB + AB' = A$

DEMOSTRACION :

USANDO TABLAS

A	B	B'	AB	AB'	AB + AB'
1	1	0	1	0	1
1	0	1	0	1	1
0	1	0	0	0	0
0	0	1	0	0	0

iguales

USANDO LEYES

$AB + AB' = A(B + B')$
$= A.1$
$= A$

(2) $(A+B)(A+B') = A$

DEMOSTRACION :

USANDO TABLAS

A	B	B'	A+B	A+B'	(A+B)(A+B')
1	1	0	1	1	1
1	0	1	1	1	1
0	1	0	1	0	0
0	0	1	0	1	0

USANDO LEYES

$$(A+B)(A+B') = A(A+B') + B(A+B')$$
$$= AA + AB' + AB + BB'$$
$$= A + A + 0$$
$$= A$$

(3) $A + A'B = A + B$

DEMOSTRACION :

USANDO TABLAS

A	B	A'	A'B	A+B	A+A'B
1	1	0	0	1	1
1	0	0	0	1	1
0	1	1	1	1	1
0	0	1	0	0	0

(iguales)

USANDO LEYES

$$A + B = A + B(A + A')$$
$$= A + AB + A'B$$
$$= A(1 + B) + A'B$$
$$= A + A'B$$

(4) $(A' + B) = AB$

DEMOSTRACION :

USANDO TABLAS

A	B	A'	A'+B	A(A'+B)	AB
1	1	0	1	1	1
1	0	0	0	0	0
0	1	1	1	0	0
0	0	1	1	0	0

(iguales)

USANDO LEYES

$$A(A' + B) = AA' + AB$$
$$= 0 + AB$$
$$= AB$$

(5) $AB + A'C + BC = AB + A'C$

DEMOSTRACION :

USANDO LEYES

$$AB + A'C + BC = ABC + ABC' + A'BC + A'B'C + ABC + A'BC$$
$$= ABC + ABC' + A'B'C + A'BC$$
$$= AB(C + C') + A'C(B + B')$$
$$= AB + A'C$$

Los ejemplos que siguen ilustrarán muchas de las maneras en que pueden aplicarse los teoremas booleanos al intentar simplificar una expresión. Se debe tomar en cuenta que estos ejemplos contienen dos etapas esenciales :

1. La expresión se pone en forma de suma de productos.

2. Una vez que está en esta forma, los términos del producto se verifican para ver si hay factores comunes y se realiza la factorización siempre que es posible. Con suerte, la factorización da como resultado la eliminación de uno o más términos.

Ejemplo : *simplifique la expresión* $Z = ABC + ABC' + AB'C$.

Solución :

$$
\begin{aligned}
Z &= ABC + ABC' + AB'C \\
&= AB(C + C') + AB'C && \text{FACTORIZANDO } AB \\
&= AB + AB'C \\
&= A(B + B'C) && \text{FACTORIZANDO } A \\
&= A(B + C) && \text{TEOREMA (3)}
\end{aligned}
$$

Ejemplo : *simplifique* $Z = A'C(A'BD)' + A'BC'D' + AB'C$

Solución :

$$
\begin{aligned}
Z &= A'C(A'BD)' + A'BC'D' + AB'C \\
&= A'C(A + B' + D') + A'BC'D' + AB'C && \text{T. DeMORGAN} \\
&= A'CA + A'CB' + A'CD' + A'BC'D' + AB'C \\
&= 0 + A'CB' + A'CD' + A'BC'D' + AB'C \\
&= B'C(A' + AD) + A'D'(C + BC) && \text{FAC. } B'C, A'D' \\
&= B'C + A'D'(C + B) && \text{TEOREMA (3)}
\end{aligned}
$$

Ejemplo : *simplifique* $X = CA' + B)CA + B + D)D'$

Solución :

$$
\begin{aligned}
X &= CA' + B)CA + B + D)D' \\
&= A'AD' + A'BD' + A'DD' + BAD' + BBD' + BDD' \\
&= 0 + A'BD' + 0 + BAD' + BD' + 0 \\
&= A'BD' + BAD' + BD' \\
&= BD'CA' + AD + BD' \qquad \textit{FACTORIZANDO } BD' \\
&= BD' + BD' \\
&= BD'
\end{aligned}
$$

2 . 2. MINIMIZACION PARA UNA FUNCION DE SALIDA.

Las funciones booleanas pueden simplificarse por métodos algebraicos como se vió en la sección anterior de este capítulo. Sin embargo, el procedimiento de minimización es difícil debido a que carece de reglas específicas para predecir cada paso sucesivo en el proceso de manipulación. El método de mapas proporciona un procedimiento simple y directo para minimizar las funciones booleanas. Este método puede considerarse ya sea como una forma grafica de una tabla de verdad o como una extensión del diagrama de Venn. El método de mapas, que Veitch fué el primero en proponer y que modificó ligeramente Karnaugh, también se conoce como el diagrama de Veitch o mapa de Karnaugh.

El mapa es un diagrama compuesto por cuadros. Cada cuadro representa un minitérmino. Ya que cualquier función booleana puede expresarse como una suma de minitérminos, se concluye que una función booleana se reconoce en forma gráfica en el mapa por el área encerrada en los cuadros cuyos minitérminos se incluyen en la función. De hecho, el mapa presenta un diagrama visual de todas las formas posibles en que puede expresarse una función en una manera estándar. Mediante el reconocimiento de diversos patrones, el usuario puede derivar expresiones algebraicas alternas para la misma función, de las cuales él puede seleccionar la más simple. Se supondrá que la expresión algebraica más simple es cualquiera en una suma de productos o producto de sumas que tiene un número mínimo de literales.

2.2.1. MAPAS DE KARNAUGH.

2.2.1.1. MAPAS DE DOS VARIABLES.

La figura 2.1 muestra un mapa de Karnaugh de dos variables. Hay cuatro miniterminos para dos variables; por lo tanto, el mapa consta de $2^2 = 4$ cuadros, uno para cada minitermino.

m_0	m_1
m_2	m_3

FIG. 2.1 MAPA DE DOS VARIABLES.

En la figura 2.2 se vuelve a dibujar el mapa para mostrar las relaciones entre los cuadros y las dos variables. Los numeros 0 y 1 que se marcan para cada renglon y cada columna designan los valores de las variables X y Y, respectivamente. Obsérvese que X aparece como prima en el renglon 0 y sin prima en el renglon 1. En forma similar, Y aparece prima en la columna 0 y sin prima en la columna 1.

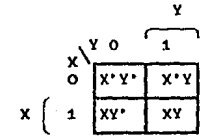


FIG. 2.2 MAPA DE DOS VARIABLES.

Si se marcan los cuadros cuyos miniterminos pertenecen a una funcion dada, el mapa de dos variables se convierte en otra forma util para representar cualquiera de las funciones booleanas de dos variables. Como ejemplo, la funcion XY se muestra en la fig. 2.3. Ya que XY = m_3, se coloca un 1 en el cuadro correspondiente a m_3.

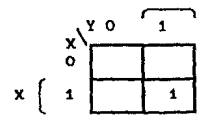


FIG. 2.3 REPRESENTACION DE XY EN EL MAPA.

En forma semejante, la función X+Y se representa en el mapa de la fig. 2.4 por tres cuadros marcados con 1. Estos cuadros se encuentran mediante los minitérminos de la función :

$$X+Y = X'Y + XY' + XY = m_1 + m_2 + m_3$$

Los tres cuadros pudieron haberse encontrado mediante la intersección de la variable X en el segundo renglón y la variable Y en la segunda columna, la cual encierra el area que pertenece a X o Y.

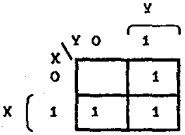


FIG. 2.4 REPRESENTACION DE X+Y EN EL MAPA.

2.2.1.2. MAPAS DE TRES VARIABLES.

La figura 2.5 muestra un mapa de Karnaugh de tres variables. Hay $2^3 = 8$ minitérminos para tres variables binarias, por lo tanto un mapa consta de ocho cuadros.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

FIG. 2.5 MAPA DE TRES VARIABLES.

El mapa de la fig. 2.6 se marca con números en cada renglón y cada columna para mostrar las relaciones entre los cuadros y las variables. Por ejemplo, el cuadro asignado a m_5 corresponde al renglón 1 y la columna 01, cuando estos dos números se concatenan, dan el número binario 101, cuyo equivalente decimal es 5. Obsérvese que hay cuatro cuadros donde cada variable es igual a 1 y cuatro cuadros donde cada variable es igual a 0.

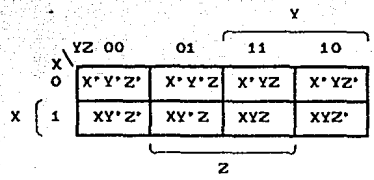


FIG. 2.6 MAPA DE TRES VARIABLES.

Para entender la utilidad del mapa y simplificar funciones booleanas, debe reconocerse la propiedad básica que poseen los cuadros adyacentes. Cualesquiera dos cuadros adyacentes en el mapa difieren solo en una variable que está en prima en un cuadro y sin prima en el otro. Por ejemplo, m_5 y m_7, caen en dos cuadros adyacentes. La variable Y tiene prima en m_5 y no tiene prima en m_7, en tanto que las otras dos variables son las mismas en ambos cuadros. La suma de dos minitérminos en cuadros adyacentes puede simplificarse a un solo término AND que consta solo de dos literales. Para aclarar esto, considérese la suma de dos cuadros adyacentes como m_5 y m_7 :

$$m_5 + m_7 = XY'Z + XYZ = XZ(Y' + Y) = XZ$$

Aquí los dos cuadros difieren por la variable Y, la cual puede eliminarse cuando se forma la suma de dos minitérminos. Por eso, cualesquiera dos minitérminos en cuadros adyacentes que se unen por el operador OR causarán una eliminación de la variable diferente. El siguiente ejemplo explica el procedimiento para minimizar una función booleana con un mapa.

Ejemplo : *simplifique la función siguiente :*

$$F = X'YZ + X'YZ' + XY'Z' + XY'Z$$

Solución : primero, se marca un 1 en cada cuadro como se necesite para representar la función como se muestra en la fig. 2.7. Esto puede llevarse a cabo de dos formas: ya sea por la conversión de cada minitérmino en un número binario y marcando entonces un 1 en el cuadro correspondiente o por la obtención de la coincidencia de

las variables en cada término. Por consiguiente, la función está representada por un área que contiene cuatro cuadros, cada uno marcado con un 1 como se muestra en la figura 2.7. El paso siguiente es subdividir el área dada en cuadros adyacentes. Esto se indica en el mapa por dos rectángulos, cada uno encierra dos 1. El rectángulo superior de la derecha representa el área encerrada por X'Y; el inferior de la izquierda el área encerrada por XY'. La suma de estos dos términos da la respuesta :

$$F = X'Y + XY'$$

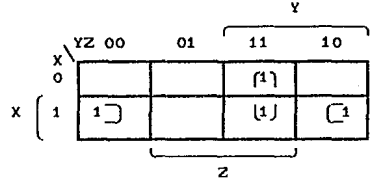


FIG. 2.7 MAPA DE TRES VARIABLES.

A continuación considérense los dos cuadros etiquetados m_0 y m_2 en la fig. 2.5 o X'Y'Z' y X'YZ' en la fig. 2.6. Estos dos miniterminos también difieren por una variable Y, y su suma puede simplificarse a una expresión de dos literales:

$$X'Y'Z' + X'YZ' = X'Z'$$

En consecuencia, debe modificarse la definición de cuadros adyacentes para incluir éste y otros casos similares. Esto se hace considerando el mapa como si estuviera dibujado en una superficie donde las orillas derecha e izquierda se tocan una con otra para formar cuadros adyacentes.

Considérese ahora cualquier combinación de cuatro cuadros adyacentes en el mapa de tres variables. Cualquiera de estas combinaciones representa la aplicación del operador OR a cuatro miniterminos adyacentes y resulta una expresión de una sola literal.

Ejemplo : *simplifique la función F = A'C+A'B+AB'C +BC*

Solución : el mapa que simplifica esta función se muestra en la figura 2.8. Algunos de los términos de la función tienen menos de tres literales y se representan en el mapa por más de un cuadro. Por ejemplo, para encontrar los cuadros correspondientes a A'C, se forma la coincidencia de A' (primer renglon) y C (dos columnas centrales) y se obtiene los cuadros 001 y 011. Obsérvese que cuando se marcan numeros 1 en los cuadros, es posible encontrar un 1 ya colocado ahí por un término precedente. En este ejemplo, el segundo término A'B tiene numeros 1 en los cuadros 011 y 010, pero el cuadro 011 es común al primer término A'C y sólo un 1 está marcado en él. En este ejemplo, la función tiene cinco miniterminos, como se indica por los cinco cuadros marcados con numeros 1. Se simplifica por la combinación de cuatro cuadros en el centro para dar la literal C. El único cuadro restante marcado con un 1 en 010 se combina con un cuadro adyacente que ya se ha usado una vez. Esto se permite y es inclusive deseable ya que la combinación de dos cuadros da el término A'B, en tanto que el único minitermino representado por el cuadro da el término de tres variables A'BC'. La función simplificada es :

$$F = C + A'B$$

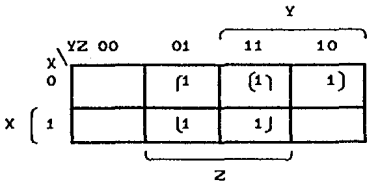


FIG. 2.8 MAPA DE TRES VARIABLES.

2.2.1.3. MAPAS DE CUATRO VARIABLES.

El mapa para las funciones booleanas de cuatro variables binarias se muestra en la fig. 2.9. Se listan los 16 miniterminos y los cuadros asignados a cada uno.

m0	m1	m3	m2
m4	m5	m7	m6
m12	m13	m15	m14
m8	m9	m11	m10

FIG. 2.9 MAPA DE CUATRO VARIABLES.

En la fig. 2.10 el mapa vuelve a dibujarse para mostrar las relaciones con las cuatro variables. Los renglones y columnas se numeran en una secuencia de codigo reflejado, con solo un dígito cambiando de valor entre dos renglones o columnas adyacentes. El minitérmino que corresponde a cada cuadro puede obtenerse por la concatenación del número de renglón con el número de columna.

La minimización por mapa de las funciones booleanas de cuatro variables es similar al metodo que se utiliza para minimizar las

		Y		
	YZ 00	01	11	10
WX				
00	W'X'Y'Z'	W'X'Y'Z	W'X'YZ	W'X'YZ'
01	W'XY'Z'	W'XY'Z	W'XYZ	W'XYZ'
W 11	WXY'Z'	WXY'Z	WXYZ	WXYZ'
10	WX'Y'Z'	WX'Y'Z	WX'YZ	WX'YZ'
		Z		X

FIG. 2.10 MAPA DE CUATRO VARIABLES.

funciones de tres variables. Se definen cuadros adyacentes para que sean cuadros juntos entre sí. Ademas, se considera que el mapa cae en una superficie en las orillas superior e inferior, al igual que en las orillas derecha e izquierda, tocandose uno a otro para formar cuadros adyacentes. La combinación de cuadros adyacentes que es util durante el proceso de simplificacion se determina con facilidad por la inspección del mapa de cuatro variables :

. Un cuadro representa un minitérmino, dando un término de

cuatro literales.

- Dos cuadros adyacentes representan un término de tres literales.

- Cuatro cuadros adyacentes representan un término de dos literales.

- Ocho cuadros adyacentes representan un término de una literal.

- dieciséis cuadros adyacentes representan la función igual a 1.

Ejemplo : *Simplifique la función :*

$F = W'X'Y'Z' + W'X'Y'Z + W'X'YZ' + W'XY'Z' + W'XY'Z + W'XYZ' + WX'Y'Z' + WX'Y'Z + WXY'Z' + WXY'Z + WXYZ'$

Solución : ya que la función tiene cuatro variables, debe usarse un mapa de cuatro variables. Los miniterminos se marcan con 1 en el mapa de la figura 2.11.

Ocho cuadros adyacentes marcados con números 1 pueden combinarse para formar un término de una literal Y'. Los tres 1 restantes a la derecha no pueden combinarse juntos para dar un término simplificado. Deben combinarse como dos o cuatro cuadros adyacentes.

Mientras mayor sea el número de cuadros combinados, menor será el número de literales en el término. En este ejemplo, los dos 1 de la parte superior a la derecha se combinan con los dos 1 de la parte superior a la izquierda para dar el término W'Z'. Obsérvese que se permite usar el mismo cuadro mas de una vez. Ahora queda un cuadro marcado con 1 en el tercer renglón y cuarta columna. En lugar de tomar este cuadro solo, se combina con cuadros que ya se han empleado para formar una área con cuatro cuadros adyacentes. Estos cuadros comprenden los dos renglones centrales y dos

columnas en los extremos, dando el término XZ'. La función simplificada es :

$$F = Y' + W'Z' + XZ'$$

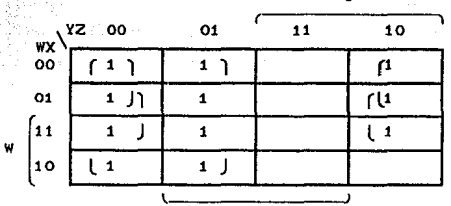


FIG. 2.11 MAPA DE CUATRO VARIABLES.

2.2.1.4. MAPAS DE CINCO Y SEIS VARIABLES.

Los mapas de más de cuatro variables no son de uso tan simple. El número de cuadros se vuelve en exceso grande y la geometría para combinar cuadros adyacentes se vuelve más complicada. El número de cuadros siempre es igual al número de miniterminos. Para mapas de cinco variables, se necesitan 32 cuadros; para mapas de seis variables se requieren 64 cuadros. Los mapas con siete o más variables necesitan tantos cuadros que no es practico usarlos. Los mapas de cinco o seis variables se muestran en las figuras 2.12 y 2.13, respectivamente. Los renglones y columnas se numeran en una secuencia de código reflejado; el minitérmino asignado a cada cuadro se lee mediante esos números. En esta forma, el cuadro en el tercer renglón (11) y la segunda columna (001), en el mapa de cinco variables, es el número 11001, el equivalente decimal es 25. Por tanto, este cuadro representa el minitérmino m25. El símbolo de letra de cada variable se marca junto a los cuadros donde el valor correspondiente de bit del número de código reflejado es un 1. Por ejemplo, en el mapa de cinco variables, la variable A es un 1 en los últimos dos renglones; B es un 1 en los dos renglones centrales. Los números reflejados en las columnas muestran las variables C con un 1 en las cuatro columnas más a la derecha, la variable D con un 1 en las cuatro columnas centrales y los 1 para

la variable E no son físicamente adyacentes pero se dividen en dos partes. La asignación de variables en el mapa de seis variables se determina de manera semejante.

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

C (110 111 101 100) · B (01, 11, 10) · A (11, 10) · D (011 010 110 111) · E (000 001 011 010) · E (110 111 101 100)

FIG. 2.12 MAPA DE CINCO VARIABLES.

La definición de cuadros adyacentes para los mapas de las figuras 2.12 y 2.13 debe modificarse de nuevo para tomar en cuenta el hecho de que algunas variables están divididas en dos partes.

ABC \ DEF	000	001	011	010	110	111	101	100
000	0	1	3	2	6	7	5	4
001	8	9	11	10	14	15	13	12
011	24	25	27	26	30	31	29	28
010	16	17	19	18	22	23	21	20
110	48	49	51	50	54	55	53	52
111	56	57	59	58	62	63	61	60
101	40	41	43	42	46	47	45	44
100	32	33	35	34	38	39	37	36

D (110 111 101 100) · C (001, 111) · B (011 010 110 101) · A (110 111 101 100) · E (001 011 111 010) · F (000 001 011 010) · F (110 111 101 100)

FIG. 2.19 MAPA DE SEIS VARIABLES.

Debe considerarse que el mapa de cinco variables consta de dos mapas de cuatro variables y que el mapa de seis variables consta de cuatro mapas de cuatro variables. Cada uno de estos mapas de cuatro variables se reconoce por las líneas dobles en el centro del mapa; cada uno tiene la adyacencia previamente definida cuando se toma de manera individual. Además, la doble línea en el centro debe considerarse como el centro de un libro, como si cada mitad del mapa fuera una página. Cuando se cierra el libro, dos cuadros adyacentes caen uno sobre el otro. En otras palabras, la línea doble del centro es como un espejo con cada cuadro que es adyacente, no sólo a sus cuatro cuadros vecinos, sino también a su imagen de espejo. Por ejemplo, el minitérmino 31 en el mapa de cinco variables es adyacente a los minitérminos 30, 15, 29, 23 y 27. El mismo minitérmino en el mapa de seis variables es adyacente a todos esos minitérminos más el minitérmino 63.

		D							
ABC \ DEF		000	001	011	010	110	111	101	100
	000	0	1	3	2	6	7	5	4
	001	8	9	11	10	14	15	13	12
	011	24	25	27	26	30	31	29	28
	010	16	17	19	18	22	23	21	20
	110	48	49	51	50	54	55	53	52
	111	56	57	59	58	62	63	61	60
	101	40	41	43	42	46	47	45	44
	100	32	33	35	34	38	39	37	36

C, B, C, A, F, E, F

FIG. 2.18 MAPA DE SEIS VARIABLES.

Mediante la inspección, y tomando en cuenta la nueva definición de cuadros adyacentes, es posible mostrar que cualesquiera cuadros adyacentes 2^k para $k=0,1,2,\ldots,n$, en un mapa de n variables, representarán un área que da un término de n-k literales. Para que lo mencionado tenga algún significado, n debe ser mayor que k. Cuando $n=k$, el área entera del mapa está combinada para dar la función de identidad. La tabla 2.1 muestra la relación entre el número de cuadros adyacentes con el número de literales en el

término.

	CUADROS ADYACENTES	LITERALES EN UN MAPA DE n VARIABLES					
k	2^k	n=2	n=3	n=4	n=5	n=6	n=7
0	1	2	3	4	5	6	7
1	2	1	2	3	4	5	6
2	4	0	1	2	3	4	5
3	8		0	1	2	3	4
4	16			0	1	2	3
5	32				0	1	2
6	64					0	1
							0

TABLA 2.1 RELACION ENTRE CUADROS ADYACENTES Y LITERALES EN EL TERMINMO.

Ejemplo : *simplifique la función :*

$$F(A,B,C,D) = \Sigma(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$$

Solución : el mapa de esta función se muestra en al fig. 2.14. Cada minitermino se convierte en su numero binario equivalente y los 1 se marcan en sus cuadros correspondientes. Ahora es necesario encontrar combinaciones de cuadros adyacentes que resulten en el area mas grande posible. Los cuatro cuadros en el centro de la mitad del mapa a la derecha se reflejan a traves de la linea doble y se combinan con los cuatro cuadros en el centro del mapa de la mitad izquierda, para dar ocho cuadros adyacentes disponibles equivalentes al término BE. Los dos 1 en el renglon inferior son reflejo uno de otro sobre la doble linea del centro.

Por la combinación de ellos con los otros dos cuadros adyacentes, se obtiene el termino AD'E. Los cuatro 1 en el renglon superior son todos adyacentes y pueden combinarse para dar el termino A'B'E'. Todos los 1 estan inclidos ahora. La función simplificada es :

$$F = BE + AD'E + A'B'E'$$

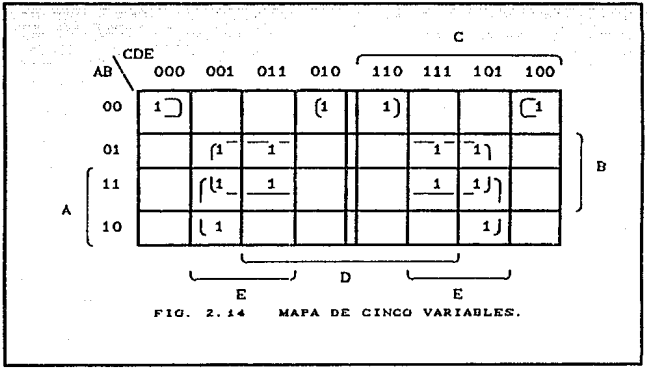


FIG. 2.14 MAPA DE CINCO VARIABLES.

2.2.2. METODO DE QUINE-McCLUSKEY.

El método de mapa de simplificación es conveniente en tanto que el número de variables no exceda cinco o seis. Conforme aumenta el número de variables, el número excesivo de cuadros evita una selección razonable de cuadros adyacentes. La desventaja obvia del mapa es que en escencia es un procedimiento de ensayo y error, que depende de la habilidad del usuario para reconocer ciertos patrones. Para funciones de seis o más variables, es difícil tener la seguridad de que se ha hecho la mejor seleccion.

El método de tabulación supera esta dificultad. Es un procedimiento específico de paso a paso que está garantizado para producir una expresión simplificada en forma estandar para una función. Puede aplicarse a problemas con muchas variables y tiene la ventaja de poder ser programado en una computadora. Sin embargo, es bastante tedioso para el uso humano y propenso a errores debido a su proceso rutinario y monótono. El método tabular lo formulo por primera vez Quine y lo mejoro posteriormente McCluskey. También se le conoce como el método de Quine-McCluskey.

Este método consta de dos partes. La primera es encontrar todos los términos que son candidatos para su inclusión en la función

simplificada. Estos términos se denominan implicantes primos. La segunda operación es escoger entre los implicantes primos los que dan una exxpresión con el menor numero de literales.

2.2.2.1. DETERMINACION DE LOS IMPLICANTES PRIMOS.

El punto de inicio del método de tabulación es la lista de los minitérminos que especifican la función. La primera operación tabular es encontrar los implicantes primos usando un proceso de comparación. Este proceso compara cada minitérmino con cada uno de los otros minitérminos. Si dos minitérminos difieren sólo en una variable, esta variable se elimina y se encuentra un término con una literal menos. Este proceso se repite para cada minitérmino hasta que se completa la busqueda. El ciclo del proceso de comparación se repite para los nuevos términos que acaban de encontrarse. Los ciclos tercero y posteriores continuan hasta que un paso único a través de un ciclo no rinde más eliminación de literales. Los términos restantes y todos los términos que no comparan durante el proceso comprenden los implicantes primos.

Ejemplo : *Simplifique la siguiente función booleana:*

$$F(W,X,Y,Z) = \Sigma(0,1,2,8,10,11,14,15)$$

Solución :

Paso 1 : Se hace la representación binaria de los minitérminos como se indica en la tabla 2.2, columna (a). Esto se hace agrupando los minitérminos en cinco secciones separadas por líneas horizontales. La primera sección contiene en numero que no contenga numeros 1. La segunda sección contiene los numeros que tienen un solo 1. La tercera, cuarta y quinta secciones contiene los numeros binarios con dos, tres y cuatro numeros 1 respectivamente. Los equivalentes decimales de los minitérminos también se llevan para identificación.

Paso 2 : Cualesquiera dos minitérminos que difieran uno del otro solo por una variable pueden combinarse, y la variable que no compara se elimina. Dos

minitérminos caen en esta categoría si ambos tiene el mismo valor de bits en todas las posiciones excepto una. Los minitermino sen una sección se comparan con los de la siguiente hacia abajo solamente, debido a que dos términos que difieren por más de un bit no pueden compararse. El minitermino en la primera sección se compara con cada uno de los tres minitérminos de la segunda sección. Si dos numeros cualesquiera son los mismos en cada posición excepto una, se coloca una marca a la derecha de ambos minitérminos para indicar que se han utilizado. El término resultante, junto con los equivalentes decimales, se listan en la columna (b) de la tabla. La variable eliminada durante la comparación se indica con un guión en su posición original.

(a)	W X Y Z		(b)	W X Y Z		(c)	W X Y Z
0	0 0 0 0	←	0,1	0 0 0 -	←	0,2,8,10	- 0 - 0
			0,2	0 0 - 0	←	0,8,2,10	- 0 - 0
1	0 0 0 1	←	0,8	- 0 0 0	←		
2	0 0 1 0	←				10,11,14,15	1 - 1 -
8	1 0 0 0	←	2,10	- 0 1 0	←	10,14,11,15	1 - 1 -
			8,10	1 0 - 0	←		
10	1 0 1 0	←					
			10,11	1 0 1 -	←		
11	1 0 1 1	←	10,14	1 - 1 0	←		
14	1 1 1 0	←					
			11,15	1 - 1 1	←		
15	1 1 1 1	←	14,15	1 1 1 -	←		

TABLA 2.2 DETERMINACION DE IMPLICANTES PRIMOS.

En este caso como m_0(0000) combina con m_1(0001) para formar (000-). Esta combinación es equivalente a la operación algebraica:

$$m_0 + m_1 = W'X'Y'Z' + W'X'Y'Z = W'X'Y'$$

El minitérmino m_0 también combina con m_2 para formar (00-0) y con m_8 para formar (-000). El resultado de esta comparación se coloca en la primera sección de la columna (b). Todas las otras secciones de (a) se

comparan en forma similar y se forman las secciones subsecuentes en (b). Este proceso de comparacion resulta en las cuatro secciones de (b).

Paso 3 : Los términos de la columna (b) tiene sólo tres variables. Un 1 bajo la variable indica que no tiene prima, y un 0 significa que si tiene prima, y un guion significa que la variable no se incluye en el termino. El proceso de busqueda y comparacion se repite para los terminos en la columna (b) para formar los terminos de dos variables de la columna (c). De nuevo, los términos en cada seccion necesitan compararse sólo si tienen guiones en la misma posición. Observe que el término (000-) no compara con cualquier otro termino. Por tanto, no tiene marca de verificacion a la derecha. El proceso de comparacion debe llevarse a cabo otra vez en la columna (c) y en las columnas subsecuentes, en tanto se encuentre una comparacion apropiada. En este ejemplo, la operacion se detiene en la tercera columna.

Paso 4 : Los términos que no están marcados en la tabla forman los implicantes primos. En este ejemplo se tiene el termino W'X'Y' (000-) en la columna (b), y los terminos X'Z' (-0-0) y WY (1-1-) en la columna (c). Observe que cada termino en la columna (c) aparece dos veces en la tabla, y en tanto el termino forma un implicante primo, es innecesario utilizar el término dos veces. La suma de los implicantes primos da una expresión simplificada de la función. Esto se debe a que cada termino marcado en la tabla lo ha tomado en cuenta una entrada en un término más simple en una columna subsecuente. Por tanto, las entradas no marcadas (implicantes primos) son los términos que se dejan para formular la función. Para este ejemplo, la suma de los implicantes primos de la función minimizada en suma de productos es :

$$F = W'X'Y' + X'Z' + WY$$

En la mayoría de los casos, la suma de implicantes primos no necesariamente forma la expresion con el numero mínimo de términos.

La manipulación tediosa que debe seguirse cuando se

utiliza el método de tabulación se reduce si se compara con la que se hace con números decimales en lugar de binarios. Se mostrará un método en el que usa sustracción de números decimales en lugar de comparar e igualar los números binarios. Obsérvese que cada 1 en un número binario representa el coeficiente multiplicado por una potencia de 2. Cuando dos miniterminos son los mismos en cada posición excepto uno, el miniterminо con el 1 adicional debe ser mayor que el número de otros miniterminos por una potencia de 2. Por tanto, dos miniterminos pueden combinarse si el número del primer miniterminо difiere en una potencia de 2 respecto a un segundo número más grande en la siguiente sección abajo en la tabla. Se ilustrará este procedimiento repitiendo el ejemplo anterior.

Como se muestra en la tabla 2.3, en la columna (a), los miniterminos se ordenan en secciones como antes, excepto que ahora sólo se listan los equivalentes decimales de los miniterminos. El proceso de comparar miniterminos es como sigue : Inspecciónese cada dos números decimales en secciones adyacentes de la tabla. Si el número en la sección abajo es mayor que el número de la sección de arriba en una potencia de 2 (esto es, 1,2,4,8,16,etc.), verifíquense ambos números para mostrar que se han usado y se escribe abajo en la columna (b). El par de números transferidos a la columna (b) incluye un tercer número entre paréntesis que designa la potencia de 2 por la cual difieren los números. El número entre paréntesis indica la posición del guión en la notación binaria. El resultado de todas las comparaciones de la columna (a) se muestra en la columna (b).

La comparación entre secciones adyacentes en la columna (b) se lleva a cabo de manera semejante, excepto que sólo se comparan los términos con el mismo número entre paréntesis. El par de números en una sección debe diferir por una potencia de 2 del par de números en la sección siguiente. Y los números en la sección siguiente abajo deben ser mayores para que tenga lugar la combinación. En la columna (c), se escriben todos los cuatro números decimales con los dos números entre paréntesis designando la posición de los guiones.

Los implicantes primos son los términos no señalados

en la tabla. Estos son los mismos de antes, excepto que están dados en notación decimal. Para convertir de notación decimal a binaria, conviértanse todos los números decimales en el término en binarios y entonces insértese un guión en las posiciones designadas por los números entre paréntesis. Por eso, 0,1(1) se convierte en binario como 0000, 0001; un guión en la primera posición de cualquier número da como resultado (000-).

(a)	(b)	(c)
0 ←	0,1 (1) ←	0,2,8,10 (2,8)
———	0,2 (2) ←	0,2,8,10 (2,8)
1 ←	0,8 (8) ←	———
2 ←	———	10,11,14,15 (1,4)
8 ←	2,10 (8) ←	10,11,14,15 (1,4)
———	8,10 (2) ←	———
10 ←	———	
———	10,11 (1) ←	
11 ←	10,14 (4) ←	
14 ←	———	
———	11,15 (4) ←	
15 ←	14,15 (1) ←	

TABLA 2.9 DETERMINACION DE IMPLICANTES PRIMOS.

2.2.2.2. SELECCION DE LOS IMPLICANTES PRIMOS.

La selección de implicantes primos que forman la función minimizada se realiza mediante una tabla de implicantes primos. En esta tabla, cada implicante primo se representa en un renglón cada minitérmino en una columna. Se marcan cruces en cada renglon para mostrar la composición de minitérminos que hacen los implicates primos. Un conjunto mínimo de aplicantes primos se elije entonces para que cubra todos los minitérminos en la función. Este procedimiento se ilustra en el ejemplo siguiente.

Ejemplo : *Minimice la funcion siguiente*

$$F(W,X,Y,Z) = \Sigma(1,4,6,7,8,9,10,11,15)$$

Solución : La tabla de implicantes primos para este ejemplo se muestra en las tablas 2.4, 2.5 y 2.6. Hay seis renglones, uno para cada implicante primo, y nueve columnas cada una representando un minitérmino de la función. Se marcan cruces en cada renglón para indicar los minitérminos contenidos en el implicante primo de este renglón. Por ejemplo, las dos cruces en el primer renglón indican

(a) W X Y Z		(b)		(c)	
1	0 0 0 1 ←	1,9	(8)	8,9,10,11	(1,2)
4	0 1 0 0 ←	4,6	(2)	8,9,10,11	(1,2)
8	1 0 0 0 ←	8,9	(1) ←		
		8,10	(2) ←		
6	0 1 1 0 ←	6,7	(1)		
9	1 0 0 1 ←	9,11	(2) ←		
10	1 0 1 0 ←	10,11	(1) ←		
7	0 1 1 1 ←	7,15	(8)		
11	1 0 1 1 ←	11,15	(4)		
15	1 1 1 1 ←				

TABLA 2.4 DETERMINACION DE IMPLICANTES PRIMOS.

que los minitérminos 1 y 9 están contenidos en el implicante primo X'Y'Z. Es aconsejable incluir la equivalente decimal del implicante primo en cada renglón, ya que en forma conveniente da los minitérminos contenidos en el. Después de marcar todas las cruces, proceda a seleccionar un número mínimo de implicantes primos.

DECIMAL	BINARIO W X Y Z	TERMINO
1,9 (8)	- 0 0 1	X'Y'Z
4,6 (2)	0 1 - 0	W'XZ'
6,7 (1)	0 1 1 -	W'XY
7,15 (8)	- 1 1 1	XYZ
11,15 (4)	1 - 1 1	WYZ
8,9,10,11 (1,2)	1 0 - -	WX'

TABLA 2.5 DETERMINACION DE IMPLICANTES PRIMOS.

			1	4	6	7	8	9	10	11	15
1,9	X'Y'Z	←	X					X			
4,6	W'XZ'	←		X	X						
6,7	W'XY				X	X					
7,15	XYZ					X					X
11,15	WYZ									X	X
8,9,10,11	WX'	←					X	X	X	X	
			←	←	←		←	←	←	←	

TABLA 2.6 DETERMINACION DE IMPLICANTES PRIMOS

La tabla completa de implicantes primos se inspecciona para buscar las columnas que contienen sólo una cruz. En este ejemplo, hay cuatro minitérminos cuyas columnas dan una sola cruz : 1,4,8 y 10. El minitermino 1, está cubierto por el implicante primo X'Y'Z, esto es, la selección del implicante primo X'Y'Z garantiza que el minitermino 1 esté incluido en la función. De manera semejante, el minitérmino 4 está cubierto por el implicante primo W'XZ' y los minitérminos 8 y 10 por el WX'. Los implicantes primos que cubren minitérminos con una sola cruz en su columna se denominan implicantes primos escenciales. Para permitir que la expresión simplificada final contenga todos los minitérminos, no se tiene otra alternativa que incluir los implicantes primos escenciales. En la tabla se coloca una marca de verificación junto a los implicantes primos escenciales para indicar que se han seleccionado.

A continuación se verifica cada columna cuyo miniterminо está cubierto por los implicantes primos escenciales que se seleccionaron. Por ejemplo, el implicante primo seleccionado X'Y'Z cubre los miniterminos 1 y 9. Se inserta una marca de verificación en la parte inferior de las columnas. En forma similar el implicante primo W'XZ' cubre los miniterminos 4 y 6, y WX' cubre los miniterminos 8,9,10 y 11. La inspección de la tabla de primo-implicandos muestra que la selección de los implicantes primos escenciales cubre todos los miniterminos de la función, excepto 7 y 15. Estos dos miniterminos deben incluirse por la selección de uno o mas primo-implicandos. En este ejemplo, es claro que el implicante primo XYZ cubre ambos miniterminos y por tanto, es el que se selecciona. Así se ha encontrado el conjunto mínimo de implicante primo cuya suma da la función minimizada requerida :

$$F(W,X,Y,Z) = X'Y'Z + W'XZ' + WX' + XYZ$$

Todas las expresiones simplificadas que se derivan en los ejemplos anteriores están en la forma de suma de productos. El método de tabulación puede adaptarse para dar una expresión simplificada en producto de sumas. Como en el método de mapa, se principia con el complemento de la función tomando los ceros como la lista inicial de miniterminos. Esta lista contiene los miniterminos que no se incluyen en la función original, los cuales son numéricamente iguales a los maxterminos de la función. El proceso de tabulación se lleva a cabo con los ceros de la función y termina con una expresión simplificada en suma de productos del complemento de la función. Tomando de nuevo el complemento, se obtiene la expresión simplificada en producto de sumas.

Una función con condiciones no importa puede simplificarse por el método de tabulación después de una ligera modificación. Los terminos no importa se incluyen en la lista de miniterminos cuando se determinan los implicantes primos. Esto permite la derivación de implicantes primos con el numero mas bajo de literales. Los terminos no importa no se incluyen en la lista de miniterminos cuando la tabla de implicantes primos se establece, porque los terminos no importa no tiene que cubrirse por los primo-implicandos seleccionados.

2.3. MINIMIZACION PARA VARIAS FUNCIONES DE SALIDA.

A continuación se tratará la minimización de varias funciones de salida para variables comunes, la implementación de este tipo de funciones a través de circuitos lógicos es muy común, por esto, la importancia de explicar este tema.

2.3.1. METODO DE QUINE-McCLUSKEY.

En el caso de circuitos combinacionales con salidas múltiples, los miniterminos son distinguidos, además, por las funciones a que pertenecen y se aplican las siguientes modificaciones a las reglas del método de Quine-McCluskey para una función de salida tratado anteriormente :

a) Para que puedan compararse implicantes pertenecientes a grupos adyacentes deben contener al menos una función en común.

b) En la comparación, un implicante se marca únicamente si el conjunto de funciones que lo distinguen está contenido completamente en el conjunto de funciones que distinguen al otro. En particular, si los conjuntos son idénticos, ambos implicantes se marcan.

c) El nuevo conjunto estará distinguido por las funciones que guarden en común las funciones que lo originen.

Para ampliar en más detalle el método de salidas múltiples, considerense el siguiente ejemplo.

Ejemplo : *minimizar las siguientes funciones salidas multiples.*

$$F_A(A,B,C,D) = \sum(0,1,3,4,5,7,9,11)$$

y

$$F_B(A,B,C,D) = \sum(1,2,3,5,6,10,11,13)$$

Solución : En la tabla 2.7 se enlistan los miniterminos por numero de unos que contienen y aparecen las funciones a las que

	A	0	←	A		0,1	(1)	←	A	0,1, 4, 5	(1,4)				
				A		0,4	(4)	←							
A	B	1	←						A	1,3, 5, 7	(4,2)				
	B	2	←	A	B	1,3	(2)		A	1,3, 9,11	(8,2)				
	A	4	←	A	B	1,5	(4)		B	2,3,10,11	(1,8)				
				A		1,9	(8)	←							
A	B	3	←	B		2,3	(1)	←							
A	B	5	←	B		2,6	(4)								
	B	6	←	B		2,10	(8)	←							
	A	9	←												
	B	10	←	A		4,5	(1)	←							
	A	1	←	A		3,7	(4)								
A	B	11	←	A	B	3,11	(8)								
	B	13	←		A	5,7	(2)	←							
					B	5,13	(8)								
					A	9,11	(2)	←							
					B	10,11	(1)	←							

TABLA 27 METODO DE UINE-McCLUSKEY PAR SALIDAS MULTIPLES.

pertenecen. Del primer grupo, el 0, que pertenece a la funcion A, se compara con el uno, que pertenece a AB. Como se tiene la funcion en comun y como cero para uno, es uno, entonces se anotan a la derecha con la funcion en comun y se marca el cero por estar su funcion completamente contenido en el conjunto de la otra. El cero perteneciente a A, con el dos, perteneciente a B, no se puede, ya que no tienen funcion comun. El cero, con A y el 4, con A tambien, si se puede, dando su diferencia 4, marcando ambos implicantes y anotando con la funcion comun, a la derecha, el implicante formado.

En la tabla 2.8 se pueden encontrar los implicantes primos escenciales, quedando al final el minitermino uno de B sin marcar. Para marcarlo solo se usa A o B y como ambos son de igual costo se toma cualquiera.

Finalmente, los productos son :

para: 1,3,9,11 (8,2) = 0001 = BD
1,3,5,7 (2,4) = 0001 = AD
0,1,4,5 (1,4) = 0000 = AC

para: $2,3,10,11 \quad (1,8) = 0010 = BC$
$5,13 \quad (8) = 0101 = BCD$
$6,6 \quad (4) = 0010 = ACD$

				A						B					
	0	1	3	4	5	7	9	11	1	2	3	5	6	10	11	13
B 2,3,10,11										←	←			←	←	
A 1,3,9,11		←	←				←	←								
A 1,3,5,7		←	←		←	←										
A 0,1,4,5	←	←		←	←											
B 5,13												←				←
AB 3,11			←					←			←				←	
B 2,6										←			←			
AB 1,5		←			←				←			←				
AB 1,3		←	←						←		←					
	←	←	←	←	←	←	←	←		←	←	←	←	←	←	←

TABLA 2.8 IDENTIFICACION DE ESCENCIALES PARA UNA FUNCION MULTIPLE.

y las funciones :

$$F_A(A,B,C,D) = BD + AB + AC$$
$$F_B(A,B,C,D) = BC + BCD + ACD$$

2.3.2. IMPLEMENTACION DE FUNCIONES BOOLEANAS EMPLEANDO COMPUERTAS NAND Y NOR.

Los circuitos digitales con más frecuencia se construyen mediante compuertas NAND y NOR que con compuertas AND u OR. Las compuertas NAND y NOR son más fáciles de fabricar con componentes electrónicos y son las compuertas básicas que se utilizan en todas

las familias IC de lógica digital. Debido a la preeminecia de las compuertas NAND y NOR en el diseño de circuitos digitales, se han desarrollado reglas y procedimientos para la conversion de las funciones booleanas dadas en terminos de AND, OR y NOT en diagramas lógicos equivalentes NAND o NOR.

Para facilitar la conversión a las lógicas NAND y NOR, es conveniente definir otros dos símbolos gráficos para esas compuertas. En la figura 2.15 se muestran dos símbolos equivalentes para la compuerta NAND. El símbolo AND invertido se ha definido con anterioridad y consta de un símbolo grafico AND seguido por un circulo pequeño. En lugar de esto, es posible representar una compuerta NOR con un símbolo gráfico OR precedido por circulos pequeños en todas las entradas. El símbolo OR invertido para la compuerta NAND es consecuencia del teorema de DeMorgan.

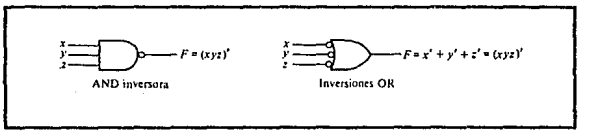


FIG. 2.15 SIMBOLOS GRAFICOS PARA LA COMPUERTA NAND.

De manera semejante, hay dos símbolos gráficos para la compuerta NOR como se muestra en la figura 2.16. El símbolo OR invertido es el convencional. La compuerta AND invertida es una alternativa conveniente que utiliza el teorema de DeMorgan.

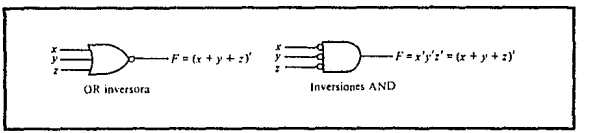


FIG. 2.16 SIMBOLOS GRAFICOS PARA LA COMPUERTA NOR.

Una compuerta de entrada NAND o NOR se comporta como una inversora. En consecuencia, una compuerta inversora puede dibujarse en tres formas diferentes como se muestra en la figura 2.17. Los circulos pequeños en todos los símbolos inversores pueden transferirse a la terminal de entrada sin cambiar la lógica de la

compuerta.

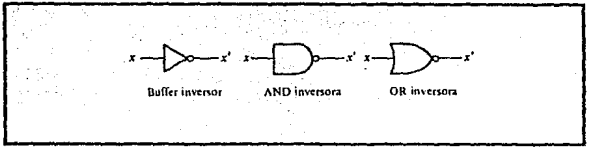


FIG. 2.17 SIMBOLOS GRAFICOS PARA INVERSORES.

2.3.2.1. IMPLEMENTACION CON NAND.

La implementación de una función boolenan con compuertas NAND requiere que la función se simplifique en la forma de suma de productos. Para ver la relación entre una expresión de suma de productos y su implentación equivalente NAND, considérense los diagramas lógicos que se dibujan en la figura 2.18. Todos los tres diagramas son equivalentes e implementan la función :

$$F = AB + CD + E$$

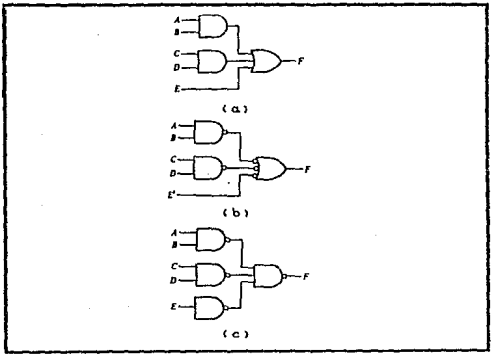


FIG. 2.18 FORMAS DE IMPLEMENTAR F=AB+CD+E.

La función está implantada en la figura 2.18(a) en la forma de suma de productos con compuertas AND y OR. En (b) las compuertas AND se reemplazan por compuertas NAND y la compuerta OR se sustituye por una compuerta NAND con un símbolo OR invertido. La variable única E se complementa y se aplica a la compuerta OR invertida del segundo nivel. El complemento de E pasa a través de un círculo pequeño que complementa la variable otra vez para producir el valor normal de E. La eliminación de los círculos pequeños en las compuertas de la figura 2.18(b) produce el circuito en (a). Por tanto, los dos diagramas implementan la función y son equivalentes.

En la figura 2.18(c), la compuerta NAND de salida se vuelve a dibujar con el símbolo ordinario. La compuerta NAND de una entrada complementa la variable E. Es posible eliminar esta inversora y aplicar E' directamente a la entrada de la compuerta NAND en el segundo nivel. El diagrama en (c) es equivalente al que se muestra en (b), el cual a su vez es equivalente al diagrama en (a). Obsérvese la similitid entre los diagramas en (a) y (c). Las compuertas AND y OR se han cambiado a compuertas NAND, pero se ha incluido una NAND adicional con la variable única E. Cuando se dibujan diagramas lógicos NAND, el circuito que se muestra ya sea en (b) o en (c) es aceptable. Sin embargo, el que se ilustra en (b) representa una relación más directa con la expresión booleana que implementa.

La implementación NAND en la figura 2.18(c) puede verificarse en forma algebraica. La función NAND que implanta puede convertirse con facilidad en una forma de suma de productos por la aplicación del teorema de DeMorgan :

$$F = [(AB)'(CD)'E']' = AB + CD + E$$

Mediante la transformación que se muestra en la figura 2.18 se concluye que una función boolena puede implementarse con dos niveles de compuertas NAND. La regla para obtener el diagrama lógico NAND mediante una función booleana es la siguiente :

a) Simplifíquese la función y exprésese en una suma de productos.

b) Dibújese una compuerta NAND para cada término producto de la función que tiene cuando menos dos literales. Las entradas a cada compuerta NAND son las literales del término. Esto constituye un grupo de compuertas de primer nivel.

c) Dibújese una sola compuerta NAND (utilizando el símbolo grafico AND invertido o bien OR invertido) en el segundo nivel, con entradas que vienen de las salidas de las compuertas del primer nivel.

d) Un término con una sola literal requiere un inversor en el primer nivel o puede complementarse y aplicarse como una entrada a la compuerta NAND en el segundo nivel.

Con compuertas NAND podemos impementar circuitos con salidas múltiples simplificando en gran medida el número de compuertas lógicas, ya que, los términos o variables que componen a las salidas se repiten en el circuito. Para implementar salidas multiples se siguen los siguientes pasos :

a) Simplificar las funciones de salidas y ponerlas en forma de suma de productos.

b) Identificar los productos comunes entre las salidas.

c) Implementar los productos que se repiten con el fin de no repetir compuertas con las mismas variables de entrada.

d) Armar las diferentes salidas.

Para mostrar mejor la aplicación de estos pasos veremos el siguiente ejemplo.

Ejemplo : *Implemente con NAND las siguientes funciones de salidas correspondientes a un circuito logico (estas salidas fueron simplificadas por el método de Quine-McCluskey):*

$$FA = BD + AD + AC$$
$$FB = BC + BCD + ACD$$

Solución : La función no requiere cambios pues ya está en forma de suma de productos.

Se realizan primero los productos que tengan dos variables para reutilizar estas salidas en los productos que tengan tres variables o más. Los primeros productos que se implementan son

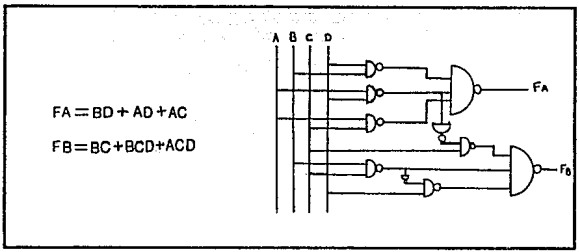


FIG. 2.19 IMPLEMENTACION CON NAND.

BD, AD, AC y BC, después se implementan los productos BCD empleando el producto BC implementado con anterioridad, luego el producto ACD es implementado empleando en producto AD o AC implentados con anterioridad. Hay que observar qué se utilizan inversores con compuertas NAND para poder negar las salidas de las NAND de los productos AD y BC, para utilizarlas como entradas a otras compuertas NAND y así formar el producto de tres variables lógicas.

2.3.2.1. IMPLEMENTACION CON NOR.

La función NOR es la dual de la función NAND. Por esta razón, todos los procedimientos y las reglas para la lógica NOR son la dual de los procedimientos y reglas correspondientes que se desarrollaron en la lógica NAND.

La implementación de una función booleana con compuertas NOR requiere que la función se simplifique en la forma de producto de

sumas. Una expresión en producto de sumas especifica un grupo de compuertas OR para los términos suma, seguidos por una compuerta AND para obtener el producto. La transformación del diagrama OR-AND al NOR-NOR se indica en la figura 2.20.

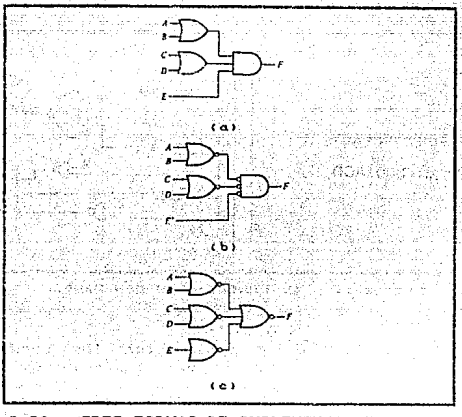


FIG. 2.20 TRES FORMAS DE IMPLEMENTAR F=(A+B)(C+D)E.

La regla para obtener el diagrama lógico NOR de una función booleana puede derivarse de esta transformación. Es similar a la regla NAND de tres pasos, excepto que la expresión simplificada debe ser el producto de sumas y los términos para el primer nivel de compuertas NOR son los términos suma. Un termino con una sola literal requiere una compuerta de una entrada NOR o inversora, o bien puede complementarse y aplicarse directamente el segundo nivel de compuerta NOR.

Las reglas para implementar salidas múltiples con compuertas NOR son las mismas que las utilizadas para implementar con compuertas NAND, sólo que emplearemos productos de sumas en lugar de suma de productos, y en lugar de implentar primero los productos se implentaran primero las sumas y hasta el final el producto de los términos suma con una NOR.

Para ilustrar mejor lo mencionado se realizará el mismo ejemplo implementado con NAND.

Ejemplo : *Implemente con NOR las siguientes funciones de salidas correspondientes a un circuito logico Estas salidas fueron simplificadas por el metodo de Quine-McCluskey):*

$$FA = BD + AD + AC$$
$$FB = BC + BCD + ACD$$

Solución : Primero tenemos que expresar las funciones en producto de sumas, esto lo haremos a través de los teoremas de DeMorgan.

$$\begin{aligned}
FA &= BD + AD + AC \\
&= ((BD)'(AD)'(AC)')' \\
&= ((B'+D')(A'+D')(A'+C'))' \\
&= ((A'B'+B'D'+A'D'+D')(A'+C'))' \\
&= ((A'B'+B'D'+D')(A'+C'))' \\
&= ((A'B'+D')(A'+C'))' \\
&= (A'B'+A'D'+A'B'C'+C'D')' \\
&= (A'B'+A'D'+C'D')' \\
&= ((A+B)'+(A+D)'+(C+D)')' \\
&= (A+B)(A+D)(C+D)
\end{aligned}$$

$$\begin{aligned}
FB &= BC + BCD + ACD \\
&= ((BC)'(BCD)'(ACD)')' \\
&= ((B'+C')(B'+C'+D')(A'+C'+D'))' \\
&= ((B'+B'C'+B'D'+B'C'+C'+C'D')(A'+C'+D'))' \\
&= ((B'+C')(A'+C'+D'))' \\
&= (A'B'+B'C'+B'D'+A'C'+C'+C'D') \\
&= (A'B'+B'D'+C') \\
&= ((A+B)'+(B+D)'+C')' \\
&= (A+B)(B+D)C
\end{aligned}$$

Pasamos a la implementación del circuito reutilizando las sumas de ambas salidas para no repetir terminos :

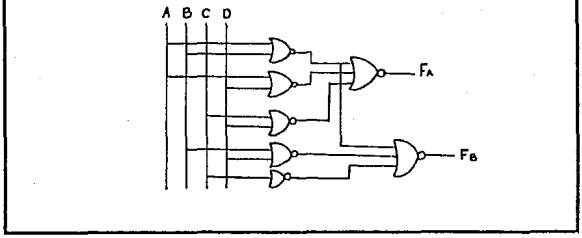


FIG. 2.21 IMPLEMENTACION CON NOR.

EJERCICIOS PROPUESTOS.

2.1 Simplifique las siguientes expresiones utilizando algebra booleana:

(a) X = A'B'C' + A'BC + ABC + AB'C' + AB'C

(b) X = (B + C')(B' + C) + (A' + B + C')'

(c) X = (C + D)' + A'CD' + AB'C' + A'B'CD + ACD'

2.2 Simplifique la expresión de problema 2.1(a) empleando mapas de Karnaugh.

2.3 Simplifique la expresión de problema 2.1(c) empleando mapas de Karnaugh.

2.4 Determine las expresiones mínimas para cada mapa de Karnaugh que se muestra a continuacion :

	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B	1	1	0	0
AB	0	0	0	1
AB'	0	1	1	0

(a)

	C'D'	C'D	CD	CD
A'B'	1	0	1	1
A'B	1	0	0	1
AB	0	0	0	0
AB'	1	0	1	1

(b)

	C'	C
A'B'	1	1
A'B	0	0
AB	1	0
AB'	1	1

(c)

2.5 Simplifique las funciones booleanas siguientes mediante el método de Quine-McCluskey.

(a) $F(A,B,C,D,E,F,G) = \sum (20,28,52,60)$

(b) $F(A,B,C,D,E,F,G) = \sum (20,28,38,39,52,60,102,103,127)$

(c) $F(A,B,C,D,E,F) = \sum (6,9,13,18,19,25,27,29,41,45,57,61)$

2.6 Simplifique por el método de Quine-McCluskey las siguientes funciones multiples :

(a) $F_{1A} = \sum (1,5,7,13,15)$
$F_{1B} = \sum (5,13)$

(b) $F_{2A} = \sum (5,7,13,15)$
$F_{2B} = \sum (1,3,5,7,9,11,13,15)$
$F_{2C} = \sum (4,5,6,7,12,13,14,15)$

(a) $F_{3A} = \sum (0,2,6,7,8,11,12,13)$
$F_{3B} = \sum (1,2,3,4,6,7,8,9,10,12,14)$

$F_{4A} = \sum (1,3,4,6,9,11,12,14,16,17,20,21,22,25,29,30,31)$
$F_{4B} = \sum (1,2,3,4,12,13,14,15,16,18,20,21,23,25,26,28,29,30,31)$

$F_{5A} = \sum (0,3,4,7,8,11,12,15)$
$F_{5B} = \sum (0,1,2,3,12,13,14,15)$
$F_{5C} = \sum (1,2,3,5,6,8,9,10,11,12)$

2.7 Simplifique cada una de las siguientes funciones e implantelas con funciones NAND. De dos alternativas.

(a) $F1 = AC' + ACE + ACE' + A'CD' + A'D'E'$
(b) $F2 = (B'+D')(A'+C'+D)(A+B'+C'+D)(A'+B+C'+D')$

2.8 Repita el problema 2.8 para implementaciones NOR.

CAPITULO III : FAMILIAS LOGICAS

CONTENIDO :

CAPITULO III : FAMILIAS LOGICAS.

3.1 ELEMENTOS BASICOS DE LOS CIRCUITOS INTEGRADOS.

Los circuitos integrados en forma invariable se construyen como circuitos integrados. Un circuito integrado (IC) es un cristal semiconductor pequeño de silicio, llamado pastilla, que contiene componentes electicos como transistores, diodos, resistores y capacitores. Los diversos componentes estan interconectados dentro de la pastilla para formar un circuito electronico. La pastilla se monta en un paquete de metal o plastico y se soldan conexiones a clavijas externas para formar el IC. Los circuitos integrados difieren de otros circuitos electronicos compuestos de componentes desprendibles en que los componenetes individuales de un IC no pueden separarse o desconectarse y el circuito en el interior del paquete es accesible solo a través de las clavijas externas.

3.2 CIRCUITOS INTEGRADOS.

Los circuitos integrados se obtienen en dos tipos de paquetes : el paquete plano y el paquete dual en línea (DIP) como se muestra en la figura 3.1. El paquete dual en línea es el tipo de mayor uso debido a su precio bajo y a su fácil instalacion en tableros para conexionar circuitos. La envolvente del paquete IC se hace de plastico o cerámica. La mayoria de los paquetes tienen tamaño estándar y el número de clavijas varía desde 8 hasta 64. Cada IC tiene una denominación numerica impresa en la superficie del paquete para su identificación. Cada vendedor publica un libro o catalogo con información que proporciona los datos necesarios que conciernen a los diversos productos.

El tamaño de los paquetes IC es muy pequeño. Por ejemplo, cuatro compuertas AND están encerradas dentro de un paquete dual en línea de 14 clavijas con dimensiones de 20x8x3 mm. Un microprocesador entero se encuentra dentro de un paquete dual en

línea de 40 clavijas con dimensiones de 50x15x4 mm.

Aparte de una reducción sustancial en tamaño, los IC ofrecen otras ventajas y beneficios en comparación con los circuitos electronicos hechos de componentes discretos. El costo de los IC es muy bajo, lo que los hace economicos para su utilización. Su consumo reducido de potencia hace que el sistema digital tenga una operación mas economica. Tiene una alta confiabilidad contra fallas, de modo que el sistema digital necesita menos reparaciones. La velocidad de operación es mas alta, lo cual los hace adecuados para operaciones de alta velocidad. El uso de los IC reduce el numero de conexiones de alambrado externas, debido a que muchas de las conexiones estan en el interior del paquete. Debido a todas estas ventajas, los sistemas digitales siempre se construyen con circuitos integrados.

Los circuitos integrados se clasifican en dos categorías generales, lineales y digitales. Los IC lineales operan con señales continuas para proporcionar funciones electronicas como amplificadores y comparadores de voltaje. Los IC digitales operan con señales binarias y están hecho de compuertas digitales interconectadas. Aquí el interes se centra sólo en los circuitos integrados digitales.

FIG. 3.1 PAQUETES DE CIRCUITOS ELECTRONICOS.

3.3 ESCALAS DE INTEGRACION PARA CIRCUITOS INTEGRADOS.

Conforme ha mejorado la tecnología de los IC, el numero de compuertas que pueden colocarse dentro de una sola pastilla de silicio ha aumentado en forma considerable. La diferenciación entre los IC que tienen unas cuantas compuertas internas y los que tiene decenas o cientos de compuertas, se hace por una referencia acostumbrada a que un paquete es un dispositivo de pequeña, mediana o gran escala de integración. Varias compuertas logicas en un solo paquete hacen un dispositivo con integración a pequeña escala (SSI). Para calificar como un dispositivo de integración a media escala (MSI), el IC debe realizar una función lógica

completa y tener una complejidad de 10 a 100 compuertas. Un dispositivo de integración a gran escala (LSI) lleva a cabo una función lógica con mas de 100 compuertas. También hay dispositivos de integración a muy alta escala (VLSI) que contiene miles de compuertas en una sola pastilla.

Muchos de los diagramas de circuitos digitales que se tratan en el desarrollo de la presente tesis muestran en detalle hasta las compuertas individuales y sus conexiones. Dichos diagramas son utiles para demostrar la construcción lógica de una función particular. Sin embargo, debe tenerse en cuenta que, en la práctica, la función puede obtenerse por un dispositivo MSI o LSI, y el usuario tiene acceso a las entradas y salidas externas pero no a las entradas o salidas de las compuertas intermedias. Por ejemplo, un diseñador que desea incorporar un registro en su sistema es más probable que escoja una función de esta clase de un circuito MSI disponible, en lugar de diseñarlo con circuitos digitales individuales como puede mostrarse en un diagrama.

3 . 4 FAMILIAS LOGICAS.

Las compuertas digitales IC se clasifican no sólo por su operación lógica, sino también por la familia de circuitos lógicos a las cuales pertenecen. Cada familia lógica tiene su propio circuito electrónico básico con el cual se desarrollan circuitos y funciones digitales más complejos.

3 . 4 . 1 TIPOS DE FAMILIAS LOGICAS.

El circuito básico de cada familia es una compuerta NAND o bien una compuerta NOR. Los componentes electrónicos que se emplean en la construcción del circuito básico por lo general se utilizan para nombrar la familia lógica. En el comercio se han introducido muchas familias lógicas diferentes de IC digitales, las cuales se listan en la tabla 3.1.

Las primeras dos, RTL y DTL, tienen sólo importancia histórica ya que rara vez se usan en los nuevos diseños. La RTL fue la primera familia comercial de uso extenso. Los circuitos DTL han sido reemplazados en forma gradual por los TTL. De hecho, la compuerta TTL es una modificación de la compuerta DTL. Las familias TTL, ECL y CMOS tiene un gran número de circuitos SSI, al igual que circuitos MSI y LSI. Las familias I^2L y MOS tiene mucho uso para construir funciones LSI.

RTL	Lógica de resistor-transistor.
DTL	Lógica diodo-transistor.
I^2L	Lógica integrada-inyección.
TTL	Lógica transistor-transistor.
ECL	Lógica emisor-acoplado.
MOS	Semiconductor metal-óxido.
CMOS	Semiconductor metal-óxido complementario.

TABLA 3.1 FAMILIAS LOGICAS.

La lógica TTL tiene una lista extensa de funciones digitales y hoy día es la familia lógica más popular. La lógica ECL se utiliza en sistemas que requieren operaciones de alta velocidad. Las MOS e I^2L se usan en circuitos que requieren alta densidad de componentes y la CMOS se emplea en sistemas que necesitan bajo consumo de potencia.

Debido a la alta densidad con la cual pueden fabricarse los transistores en MOS e I^2L, estas dos familias son las que más se utilizan para las funciones LSI. Las otras tres familias TTL, ECL, y CMOS, tienen dispositivos LSI y también un gran número de dispositivos MSI y SSI. Los dispositivos SSI son los que incluyen un pequeño número de compuertas o flip-flops en un paquete IC. El límite del número de circuitos en los dispositivos SSI es el número de clavijas en el paquete. Por ejemplo, un paquete de 14 clavijas puede acomodar sólo cuatro compuerta de dos entradas, debido a que cada compuerta requiere tres clavijas externas, dos para cada una de las entradas y una para la salida, con un total de 12 clavijas. Las dos clavijas restantes se necesitan para suministrar potencia a los circuitos.

Un ejemplo de circuitos típicos SSI se muestra en la figura 3.2. Este IC se encapsula en un paquete de 14 clavijas. Las clavijas se numeran a lo largo de los dos lados del paquete y especifican las conexiones que pueden hacerse. Las compuertas dibujadas dentro de los IC son solo para información y no pueden verse debido a que el paquete IC real aparece como lo muestra la figura 3.1.

Los IC de la familia TTL por lo común se distinguen por designaciones numéricas como las series 5400 y 7400. La primera tiene amplios margenes de temperatura de operación, adecuados para uso militar y, la segunda tiene margenes mas reducidos de temperatura, adecuados para uso industrial. La designación numérica de la serie 7400 significa que los paquetes IC están numerados como 7400, 7401, 7402, etc. Algunos proveedores ponen a la disposición IC de la familia TTL con denominaciones numéricas

diferentes, como la series 9000 u 8000.

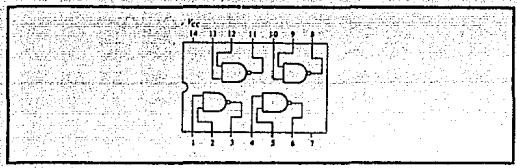


FIG. 3.2 CIRCUITO TTL (SSI TIPICO).

El tipo más común de ECL se designa como la serie 10000. En la figura 3.3 se muestran dos circuitos ECL. La serie 10102 proporciona compuertas NOR de dos entradas. Obsérvese que una compuerta ECL puede tener dos salidas, una para la función NOR y otra para la función O (clavija 9 del 10102 IC). El 10107 IC proporciona tres compuertas excluyentes OR. Aquí hay de nuevo dos salidas para cada compuerta; la otra salida de la función excluyente NOR o de equivalencia. Las compuertas ECL tiene tres terminales para suministro de potencia. V_{CC1} y V_{CC2} por lo común se conectan a tierra y V_{EE} a un suministro de -5.2 V.

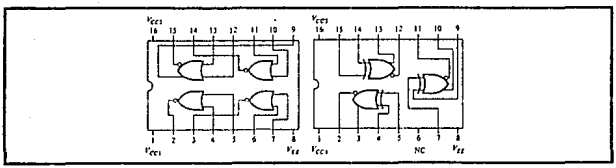


FIG. 3.3 CIRCUITO ECL.

Los circuitos CMOS de la serie 4000 se muestran en la figura 3.4. Sólo pueden acomodarse en el 4002 dos compuertas NOR de cuatro entradas, debido a la limitación de clavijas. El tipo 4059 proporciona seis compuestas buffer. Ambos ICs tienen dos terminales sin uso marcadas NC (no conexión). La terminal marcada V_{DD} requiere un voltaje en el suministro de potencia de 3 a 15 V, en tanto V_{SS} por lo común se conecta a tierra.

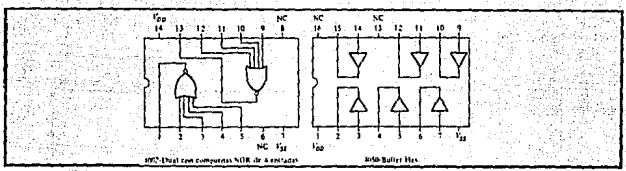


FIG. 3.4 CIRCUITO CMOS.

3.4.2 CARACTERISTICAS BASICAS DE LAS FAMILIAS LOGICAS.

Las características de las familias IC de lógica digital por lo común se comparan por el análisis de circuito de la compuerta, básica en cada familia. Los parámetros más importantes que se evaluan y comparan son la salida en abanico (multiplicidad de conexiones en la salida), disipación de potencia, retardo de propagación y margen de ruido. Se explicarán primero las propiedades de este parámero y después se utilizarán para comparar las familias IC lógicas.

El *abanico de salida* especifica el número de cargas estándar que pueden impulsar la salida de una compuerta sin menoscabar su operación normal. Una carga estándar por lo común se define como la cantidad de corriente necesaria por una entrada de otra compuerta en la misma familia IC. Algunas veces el término carga se usa en lugar de abanico de salida. Este término se deriva del hecho de que la salida de una compuerta puede suministrar una cantidad limitada de corriente, arriba de la cual cesa su operación apropiada y se dice que está sobrecargada. La salida de una compuerta por lo general se conecta a las entradas de otras compuertas similares. Cada entrada consume una cierta cantidad de potencia de la entrada de la compuerta, de modo que cada conexión adicional se agrega a la carga de la compuerta. Las "reglas de carga" por lo común se listan para una familia de circuitos digitales estándar. Estas reglas especifican la máxima cantidad de carga permitida para cada salida de cada circuito. El exceder la carga máxima especificada puede causar un mal funcionamiento debido a que el circuito no puede suministrar la potencia demandada de él. El abanico de salida es el número máximo de entradas (a otros circuitos) que pueden conectarse a la salida de una compuerta y se expresa por un número.

Las capacidades del abanico de salida de una compuerta pueden considerarse cuando se simplifican las funciones booleanas. Debe

tenerse cuidado de no desarrollar expresiones que resulten en una compuerta sobrecargada. Los amplificadores no inversores o buffer algunas veces se emplean para proporcionar capacidades adicionales de impulsión para cargas pesadas.

La *disipación de potencia* es la potencia suministrada requerida para operar la compuerta. Este parámetro se expresa en milliwatts (mW) y representa la potencia real disipada en la compuerta. El número que representa este parámetro no incluye la potencia suministrada por otra compuerta; más bien, representa la potencia suministrada a la compuerta por el suministro de potencia. Un IC con cuatro compuertas requerirá de su suministro de potencia, cuatro veces la potencia disipada por cada compuerta. En un sistema dado, puede haber muchos IC y, la potencia requerida por cada IC debe considerarse. La disipación total de potencia en un sistema es al suma total de la potencia disipada en todos los IC.

El *retardo de propagación* es el retardo de tiempo de transición promedio para que una señal se propague desde la entrada a la salida cuando la señal binaria cambia en valor. Las señales a través de una compuerta toman cierta cantidad de tiempo para propagarse desde las entradas a la salida. Este intervalo de tiempo se define como el retardo de propagación de la compuerta. El retardo de propagación se expresa en nanosegundos (ns) y, un ns es igual a 10^{-9} de un segundo.

Las señales que viajan de las entradas de un circuito digital a su salidas pasan a través de una serie de compuertas. La suma de los retardos de propagación a través de las compuertas es el retardo total de propagación del circuito. Cuando la velocidad de operación es importante, cada compuerta debe tener un pequeño retardo de propagación y el circuito digital debe tener un número mínimo de compuertas en serie entre las entradas y las salidas.

En la mayoría de los circuitos digitales las señales de entradas se aplican en forma simultánea a más de una compuerta. Todas las compuertas que reciben sus entradas exclusivamente desde las entradas externas, constituyen el primer nivel lógico del circuito. Las compuertas que reciben cuando menos una entrada de una salida de una compuerta del primer nivel lógico se considera que están en el segundo nivel lógico, y en forma semejante, para el tercer nivel y los más altos. El retardo total de propagación del circuito es igual al retardo de propagación de una compuerta multiplicado por el número de niveles lógicos en el circuito. Luego, una reducción en el número de niveles lógicos produce una reducción del retardo de señal y en circuitos más rápidos. La reducción del retardo de propagación en los circuitos puede ser

más importante que la reducción en el número total de compuertas si la velocidad de operación es un factor principal.

El *margen de ruido* es el máximo voltaje de ruido añadido a la señal de entrada de un circuito digital que no causa un cambio indeseable en la salida del circuito. Hay dos tipos de ruido que considerar : el ruido CC es causado por variaciones en los niveles de voltaje de una señal. El ruido CA es un pulso aleatorio que puede crearse por otras señales de interrupción. Por eso, el ruido es un termino que se utiliza para denominar una señal indeseable que está superpuesta sobre al señal normal de operación. La capacidad de los circuitos para operar en forma confiable en un ambiente de ruido es importante en muchas aplicaciones. El margen de ruido se expresa en volts (V) y representa la señal de ruido máximo que puede tolerarse por su compuerta.

El circuito básico de la familia lógica TTL es la compuerta NAND. Hay muchas versiones de la TTL y tres de ellas se listan en la tabla 3.2. En esta tabla se dan las características generales de las familias lógicas IC. Los valores que se listan son representativos en una base de comparación. Para cualquier familia o versión, los valores pueden tener cierta variación.

La compuerta estándar TTL fue la primera versión de la familia TTL. Conforme progreso la tecnología, se agregaron mejoras adicionales. La TTL Schottky es una ultima mejora que reduce el retardo de propagación, pero resulta en un aumento de la disipación de potencia. La versión TTL Schottky de baja potencia sacrifica cierta velocidad para reducir la disipación de potencia. Tiene el mismo retardo de propagación que la TTL estándar, pero la disipación de potencia se reduce en forma considerable. El abanico de salida de la TTL estándar es 10, pero la versión Schottky de baja potencia tiene un abanico de salida de 20. Bajo ciertas condiciones las otras versiones también pueden tener un abanico de salida de 20. El margen de ruido es mejor que 0.4 V, con un valor típico de 1 V.

El circuito básico de la familia ECL es la compuerta NOR. La ventaja especial de las compuertas ECL es su bajo retardo de propagación. Algunas versiones ECL pueden tener un retardo de propagación tan bajo como 0.5 ns. La disipación de potencia en las compuertas ECL es comparativamente alta y el margen de ruido bajo. Estos dos parametros imponen una desventaja cuando se elige la ECL sobre las otras familias logicas. Sin embargo, debido a su bajo retardo de propagación, la ECL ofrece la velocidad más alta entre todas las familias y es la elección final para sistemas muy rápidos.

FAMILIA LOGICA IC	ABANICO DE SALIDA	DISIPACION DE POTENCIA (mW)	RETARDO DE PROPAGACION (ns)	MARGEN DE RUIDO (V)
ESTANDAR TTL	10	10	10	0.4
SCHOTTKY TTL	10	22	3	0.4
BAJA POTENCIA SCHOTTKY TTL	20	2	10	0.4
ECL	25	25	2	0.2
CMOS	50	0.1	25	3

TABLA 3.2 CARACTERISTICAS TIPICAS DE LAS FAMILIAS LOGICAS.

El circuito más bajo de CMOS es el inversor por el cual ambas compuertas NAND y NOR pueden construirse. La ventaja especial del CMOS es su disipacion de potencia en extremo baja. Bajo condiciones estáticas, la disipación de potencia de la compuerta CMOS es despreciable, con promedio de cerca de 10 nW. Cuando la señal de la compuerta cambia de estado, hay una disipación dinamica de potencia que es proporcional a la frecuencia a la cual se ejerce el circuito. El numero que se lista en la tabla es un valor típico de la disipación dinamica de potencia en las compuertas CMOS.

Una desventaja principal de la compuerta CMOS es su alto retardo de propagación. Esto significa que no es práctica para utilizarse en sistemas que requieren operaciones a alta velocidad. Los parámetros característicos de la compuerta CMOS dependen del voltaje de suministro de potencia V_{DD} que se use. La disipación de potencia aumenta conforme aumenta el voltaje de suministro. El retardo de propagación disminuye con el incremento en el voltaje de suministro, y el margen de ruido se estima que es alrededor del 40 % del valor del voltaje de suministro.

3.5 TTL Y SUS CARACTERISTICAS.

En este momento, la familia: lógica-transistor (TTL) todavía disfruta de un extenso uso en aplicaciones que requieren de dispositivos SSI y MSI. El circuito lógico TTL básico es la

compuerta NAND. Su diagrama de circuito detallado, que se muestra en la figura 3.5 (a), tiene varias caracteristicas distintivas. Primero, notese que el transistor Q_1 tiene dos emisores; de este modo, tiene dos uniones con base en el emisor (E-B) que se puede utilizar para activar Q_1. Este transistor de entrada con emisores multiples puede tener hasta ocho emisores para una compuerta NAND de ocho entradas.

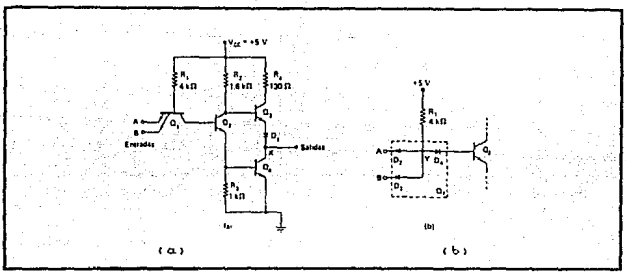


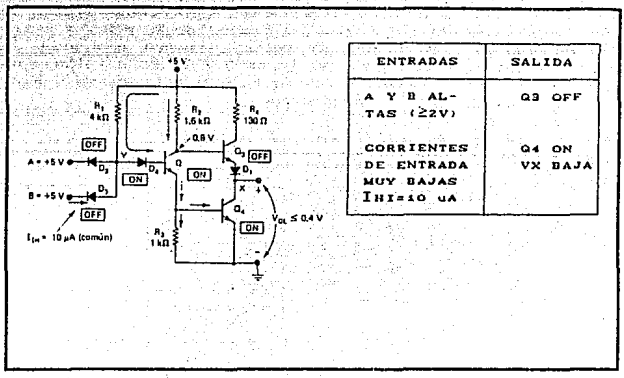
FIG. 3.5 (a) COMPUERTA NAND TTL BASICA; (b) DIODO EQUIVALENTE PARA Q_1.

Nótese así mismo que en la salida del circuito, los transistores Q_3 y Q_4 estan en disposicion con polo en forma de totem. Como se observara brevemente, en operacion normal Q_3 y Q_4 serán conductores, segun el estado logico de la salida.

Operación del circuito- Estado BAJO (LOW). Aunque este circuito parece extremadamente complejo, podemos simplificar un tanto su analisis utilizando el diodo equivalente del transistor Q_1 con multiples emisores como se muestra en la figura 3.5 (b). Los diodos D_2 y D_3 representan las dos uniones E-B de Q_1 y D_4 es la union con base en el colector (C-B). En el siguiente analisis utilizaremos esta representacion para Q_1.

Primero consideremos el caso donde la salida es BAJA. La figura 3.6 muestra esta situacion con las entradas A y B ambas en +5 V en los catodos de D_2 y D_3 desactivaran estos diodos y casi no conduciran corriente. El suministro de +5 V enviara corriente a través de R_1 y D_4 en la base de Q_2, que se activa. La corriente del emisor de Q_2 fluirá en la base de Q_4 y activara Q_4. Al mismo tiempo, el flujo de la corriente colectora de Q_2 produce una disminucion de voltaje a través de R_2 que reduce el voltaje del

colector de Q_2 a un valor bajo que es insuficiente para activar Q_3.



ENTRADAS	SALIDA
A Y B ALTAS (≥2V)	Q3 OFF
CORRIENTES DE ENTRADA MUY BAJAS IHI=10 uA	Q4 ON VX BAJA

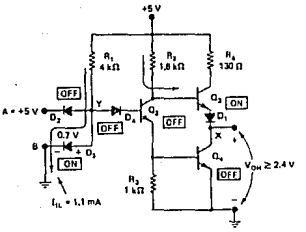
FIG. 3.6 COMPUERTA NAND TTL.

El voltaje presente en el colector de Q_2 se muestra aproximadamente como 0.8 V. Esto se debe a que el emisor debido a V_{CE} (sat). Este voltaje de 0.8 V en la base de Q_3 no es sufuciente para influenciar a futuro la union E-B y el diodo D_1 y Q_3. De hecho D_1 se necesita para mantener a Q_3 desactivado en esta situacion.

Con Q_4 activado, la terminal de salida, X, estará en un voltaje muy bajo, ya que la resistencia del estado activado (ON) de Q_4 será baja (1-25 Ω). En realidad el voltaje de salida, V_{OL}, dependera de cuanta corriente de colector conduzca Q_4. Con Q_3 desactivado, no hay corriente que emene de la terminal de +5 V a través de R_4. Como se observara, la corriente del colector de Q_4 emanará de las entradas TTL a las que está conectada la terminal X.

Es importante observar que las entradas ALTAS en A y B tendrán que suministrar solamente una pequeña corriente de dispersión de diodos. Comunmente, esta corriente I_{IH} está solo alrededor de 10 μA a la temperatura de la habitacion.

Operación del circuito-Estado ALTO (HIGH). La figura 3.7 muestra la situación donde la salida del circuito es alta. Esta situación puede ser producida conectada una u otra, o ambas entradas a BAJO. Aquí la entrada B se conecta a tierra. Esto influenciará a futuro a D_3 de manera que la corriente fluirá desde la terminal de suministro de +5 V, a través de R_1 y D_3, y a través de la terminal B a tierra. El voltaje futuro a través de D_3 mantendrá el punto Y aproximadamente en 0.7 V. Este voltaje no es suficiente para influenciar a futuro a D_4 y a la unión E-B de Q_2 lo necesario para lograr la conducción.



ENTRADAS	SALIDA
A o B o AMBAS BAJAS	Q4 OFF
LA CORRIENTE RETORNA A TIERRA A TRAVES DE LA TERMINAL DE ENTRADA BAJA IIL=1.1 mA	Q3 ACTUA COMO EMISOR-SEGUIDOR Y VOH≥2.4V GENERALMENTE 3.6 V.

FIG. 3.7 COMPUERTA NAND TTL.

Con Q_2 en el estado desactivado (OFF), no hay corriente de base para Q_4 y se desactiva. Ya que no hay corriente colectora Q_2, el voltaje en la base de Q_3 será lo suficientemente grande para influenciar a futuro a Q_3 y D_1, de modo que Q_3 conduzca. En realidad, Q_3 actúa como un emisor-seguidor, debido a que la terminal de salida X está esencialmente en su emisor. Sin una carga conectada del punto X a tierra, V_{OH} estará alrededor de 3.6 a 3.8 V, puesto que dos disminuciones de los diodos de 0.7 V (E-B de Q_3 y D_1) se restan de los 5 V aplicados a la base de Q_3. Este voltaje disminuirá en la carga debido a que la misma extraerá la corriente emisora de Q_3, que a su vez obtiene la corriente de la base a través de R_2, con lo cual se incrementa la disminución de voltaje a través de R_2.

Es importante observar que existe una corriente sustancial que fluye a través de la terminal de entrada B a tierra. Esta

corriente, I_{IL}, está comúnmente alrededor de 1.1 mA. La entrada B BAJA actua como un drenaje a tierra de esta corriente.

Acción de drenaje de la corriente. Los circuitos lógicos TTL son circuitos de drenaje de corriente de las entradas que conducen en el estado BAJO. La figura 3.8 muestra la salida de una compuerta TTL que impulsa la entrada de otra. Cuando la salida de la compuerta impulsora es BAJA, el transistor Q_4 está saturado y Q_3 está apagado (OFF). El voltaje BAJO en X influirá a futuro al emisor de Q_1 y la corriente fluye, como se muestra hacia Q_4. La corriente colectora de saturación de Q_4 es suministrada por la compuerta que conduce. Q_4 actúa como un drenaje de corriente.

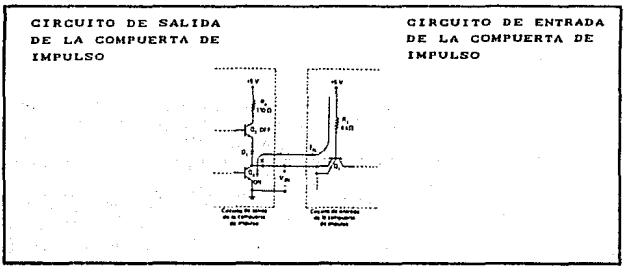


FIG. 3.8 ACCION DE DRENAJE DE CORRIENTE TTL.

3 . 5 . 1 . CARACTERISTICAS DE LA SERIE TTL ESTANDAR.

En el año de 1964 la Texas Instruments presentó la primera línea estándar de productos de circuitos TTL. La serie 5400/7400, como se le conoce, ha sido una de las familias de lógica de circuitos integrados más ampliamente utilizada. Simplemente nos referimos a ella como la serie 7400, ya que la única diferencia entre las versiones 5400 y 7400 es que la serie 5400 es para uso militar y puede operar a una temperatura e intervalo de suministro de energía mayores. Muchos fabricantes de circuitos integrados producen ahora la línea 7400 de circuitos integrados, aunque algunos utilizan sus números de identificación. Por ejemplo, Fairchild tiene una serie de circuitos integrados TTL que hace uso de números como 9N00, 9300, 9600, etc. Sin embargo, en el manual de especificación de Fairchild por lo general se indica el número equivalente de la serie 7400.

La serie 7400 opera con exactitud en el intervalo de temperatura de 0 a 70°C y con un voltaje de suministro (Vcc) de 4.75 a 5.25 V. La serie 5400 es un tanto mas flexible, ya que puede tolerar un intervalo de temperatura de -55 a +125°C y una variacion de voltaje de 4.5 a 5.5 V. Ambas series tienen comunmente un factor de carga (o fan-out) de 10, lo cual indica que puede impulsar con exactitud otras 10 entradas.

Niveles de voltaje de la 7400. La tabla 3.3 enlista los niveles de voltaje de entrada y salida para la serie 7400 estándar. Los valores mínimo y maximo se muestra en las peores condiciones de suministro de energía, temperatura y condiciones de carga. La inspección de la tabla revela una salida 0 lógica máxima garantizada V_{OL}=0.4 V, que es 400 mV menor que el voltaje 0 lógico que se necesita en la entrada V_{IL}=0.8 V. Esto significa que el margen de ruido dc en estado BAJO garantizado es 400 mV. Es decir,

$$V_{NL} = V_{IL(máx)} - V_{OL(máx)} = 0.8\ V - 0.4\ V = 400\ mV$$

Análogamente, la salida 1 lógica V_{OH} es un mínimo garantizado de 2.4 V, que es 400 mV mayor que el voltaje 1 lógico que se necesita en la entrada V_{IH}= 2.0 V. De este modo, el margen de ruido DC en el estado ALTO es 400 mV.

$$V_{NH} = V_{OH(mín)} - V_{IH(mín)} = 2.4\ V - 2.0\ V = 0.4\ V = 400\ mV$$

Así pues, los márgenes de ruido dc en el peor de los casos garantizados para la serie 7400 son ambos de 400 mV. En la operación real de los márgenes de ruido dc comunes son un tanto mayores (V_{NL} =1 V y V_{NH} =1.6 V).

	MINIMO	NORMAL	MAXIMO
V_{OL}	—	0.2	0.4
V_{OH}	2.4	3.6	—
V_{IL}	—	—	0.8
V_{IH}	2.0	—	—

TABLA 3.3 NIVELES DE VOLTAJE DE LA SERIE 7400 ESTANDARD.

3.5.2. OTRAS SERIES TTL.

Los circuitos integrados de la serie 7400 estándar ofrecen una combinación de velocidad y disipación de energía adecuadas a

muchas aplicaciones. Los circuitos integrados que se ofrecen en esta serie incluyen una amplia variedad de compuertas, biestables (flip-flops) y multivibrador monoestable en la línea de integracion a pequeña escala (SSI) y registros de corrimiento, contadores, decodificadores, memorias y circuitos aritmeticos en la línea de la integración a mediana escala (MSI).

Se han creado otras series TTL desde la introducción de la serie 7400 estandar. Estas otras series ofrecen una vasta alternativa de caracteristicas de velocidad y energía. Descubriremos estas series en los párrafos siguientes. Nótese que siempre que se usa el término "TTL", por lo general se hace referencia a la serie 7400 estándar.

TTL de baja energía, serie 74L00 (L-TTL). Los circuitos TTL de baja energia designados como la serie 74L00 tiene en esencia el mismo circuito básico que la serie 7400 estándar excepto que todos los valores de las resistencias se incrementan. Las resistencias mayores reducen los requisitos de suministro de energía pero a expensas de demoras mas largas en la propagación. Una compuerta NAND común de esta serie tiene una disipacion de energía en promedio de 1 mW y una demora en promedio en la propagación de 33 ns.

La serie 74L00 es ideal en aplicaciones en las cuales la disipación de la energia es mas crítica que la velocidad. Los circuitos de baja frecuencia operados con baterías como las calculadoras se adaptan bien a esta serie TTL. El término "L-TTL" se utilizará para hacer referencia a la serie TTL de baja energía.

TTL de alta velocidad, serie 74H00 (H-TTL). La serie 74H00 es una serie TTL de alta velocidad. Los circuitos básicos de esta serie son esencialmente los mismos que los de la serie 7400 estándar a excepción de que se emplean valores menores de resistencia y el transistor emisor-seguidor Q_4 es reemplazado por un par de Darlington. Estas diferencias producen una velocidad de cambio mucho mayor en una demora en promedio en la propagación de 6 ns. No obstante, la velocidad aumentada se logra a expensas de la disipación creciente de la energía. La compuerta NAND básica de esta serie tiene un P_D en promedio de 23 mW. Esta serie H-TTL se ha convertido esencialmente obsoleta debido a la creación de la serie TTL de Schottky que se describe en los párrafos siguientes.

TTL de Schottky, serie 74S00 (S-TTL). Las series TTL, L-TTL y H-TTL operan todas utilizando los cambios saturados en los cuales muchos de los transistores, cuando conducen, estaran en la condición saturada. Esta operación ocasiona una demora al tiempo del almacenamiento, t_s, cuando los transistores pasen de ENCENDIDO (ON) a APAGADO (OFF) limitan la velocidad de cambios en el

circuito.

La serie 74S00 reduce esta demora en el tiempo de almacenamiento no permitiendo al transistor profundizar tanto en la saturación. Logra esto, utilizando una barrera Schottky (diodo SBD) conectado entre la base y el colector de cada transistor como se muestra en la figura 3.9(a). El SBD tiene un voltaje de conducción de sólo 0.25 V. Así, cuando la unión C-B se encuentre con un voltaje de conducción hacia el nivel de la saturación, el SBD conducirá y desviará alguna de la corriente de entrada de la base. Esto reduce la corriente excesiva de la base y disminuye la demora en el tiempo de almacenamiento en el modo de APAGADO (OFF).

Como se muestra en la figura 3.9(a), la combinación del transistor y el SBD tiene un símbolo especial. Este símbolo se usa para todos los transistores del diagrama de circuito para la compuerta NAND 74S00 de la figura 3.9(b). Esta compuerta NAND 74S00 tiene una demora promedio en la propagación de sólo 3 ns, que es dos veces más rápido que el 74H00. Nótese la presencia de diodos de derivación D_1 y D_2 para limitar los voltajes negativos de entrada.

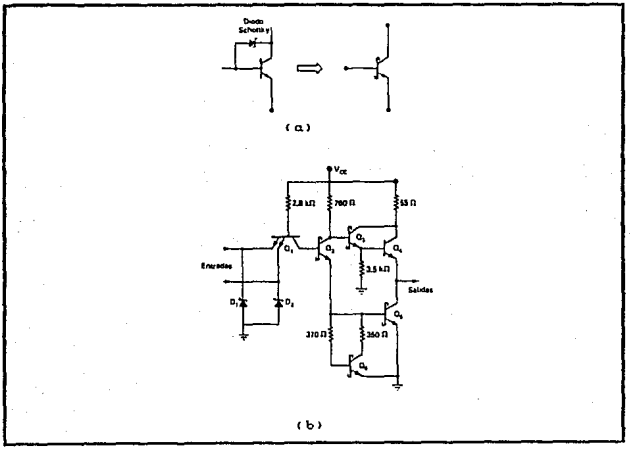


FIG. 3.9 (a) TRANSISTOR SCHOTTKY DE ACCION CONTROLADA; (b) COMPUERTA NAND BASICA EN LA SERIE LS-TTL.

Los circuitos de la serie S-TTL utilizan así mismo valores pequeños de resistencia para ayudar a depurar los tiempos de cambio. Esto incrementa la disipación de energía en promedio del circuito en cerca de 23 mW, misma que para la serie H-TTL. Los circuitos S-TTL utilizan también un par de Darlington (Q_3 y Q_4) para ofrecer un tiempo de elevación de salida mas rapido cuando se pase de ENCENDIDO (ON) a APAGADO (OFF).

De este modo, el S-TTL tiene dos veces mayor velocidad que la serie H-TTL con casi el mismo requisito de suministro de energía. Esta es la razon por la cual la serie H-TTL se vuelve obsoleta.

TTL Schottky de baja energía, serie 74LS00 (LS-TTL). Esta serie es una version de energía y velocidad menores de la serie 74S00. Utiliza el transistor de Schottky limitado, pero con valores mas grandes de resistencia que la serie S-TTL (vease la figura 3.10). Los valores mayores de resistencia reducen el requisito de energía del circuito, pero a expensas de un aumento de los tiempos de alternacion.

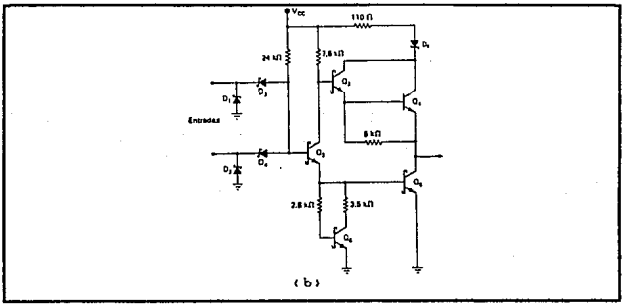


FIG. 3.10 COMPUERTA NAND 74LS00.

Una compuerta NAND de la serie LS-TTL comúnmente tendrá una demora promedio de la propagacion de 9.5 ns y una disipacion de energía en promedio de 2mW. Ya que tiene cerca de la misma velocidad de cambio que la serie TTL estándar con un requisito mucho menor de energía, la serie LS-TTL ha sustituido gradualmente a la serie 7400 en aquellas aplicaciones donde se requiere de una operación relativamente a alta velocidad con un mínimo consumo de energía. En otras palabras, la serie LS-TTL se convierte en el

"soporte principal" de la familia TTL y se puede uncontrar en casi todos los nuevos diseños TTL que no requieren la maxima velocidad (donde se usa la serie LS-TTL) o la minima disipacion de energia (donde se emplea la serie L-TTL).

Notese que la compuerta NAND 74LS00 de la figura 3.10 no hace uso del transistor de entrada con emisores multiples. Utiliza diodos de entrada (D_1 y D_2), pero la operacion basica del circuito de la misma que para las entradas con emisores multiples.

Comparacion de las series TTL. Algunas de las caracteristicas mas importantes de las diferentes series TTL se muestran en la tabla 3.4. Esta tabla da valores comunes que pueden variar ligeramente de un fabricante a otro. Los datos del fabricante deben consultarse siempre para conocer caracteristicas especificas de un IC determinado. Notese que todas las series TTL utilizan el mismo voltaje nominal de +5 V para V_{CC}, pero existe alguna variacion en los limites del voltaje de salida, V_{OH} (min) y V_{OL} (max).

La última entrada en la tabla representa la máxima frecuencia del cronometro en la cual puede operar con exactitud un FF J-K en promedio.

	ESTANDAR	L-TTL	H-TTL	S-TTL	LS-TTL
V_{CC}	← 5 V →				
$V_{OH(MIN)}$	2.4 V	2.4 V	2.4 V	2.7 V	2.7 V
$V_{IH(MIN)}$	2.0 V	2.0 V	2.0 V	2.0 V	2.0 V
$V_{OL(MAX)}$	0.4 V	0.3 V	0.4 V	0.5 V	0.5 V
$V_{IL(MAX)}$	0.8 V	0.8 V	0.8 V	0.8 V	0.8 V
DEMORA EN LA PROPAGACION	9 ns	33 ns	6 ns	3 ns	9.5 ns
ENERGIA REQUERIDA	10 mW	1 mW	23 mW	23 mW	2 mW
INTENSIDAD MAX. DE CRONOMETRO	25 MHz	3 MHz	40MHz	80MHz	30 MHz

TABLA 3.4 CARACTERISTICAS COMUNES DE LA SERIE TTL.

Ejemplo : Utilice la tabla 8.3 para calcular los márgenes de ruido DC para un circuito integrado LS-TTL comun. Como se comparan estos margenes con los margenes de ruido de la serie TTL estandar?

Solucion :

$$V_{NH} = V_{OH(min)} - V_{IH(min)}$$
$$= 2.7\ V - 2.0\ V$$
$$= 0.7\ V$$

en comparacion con $V_{NH} = 0.4\ V$ para la serie TTL estandar.

$$V_{NL} = V_{OL(max)} - V_{IL(max)}$$
$$= 0.8\ V - 0.5\ V$$
$$= 0.3\ V$$

en comparacion con $V_{NL} = 0.4\ V$ de la serie TTL.

3.5.3 REGLAS DE CARGA DEL TTL.

Al diseñar sistemas digitales con dispositivos TTL es importante saber como determinar y utilizar la capacidad del factor de carga o de impulso de cada circuito. La figura 3.11(a) muestra una salida TTL individual en el estado BAJO conectada a varias entradas TTL. El transistor Q_4 esta encendido y actua como devolucion de corriente para todas las corrientes (I_{IL}) que regresen en cada entrada. Aunque Q_4 está saturado, su resistencia en estado ENCENDIDO es algun valor distinto de cero, de manera que la corriente I_{OL} produce una disminucion del voltaje de salida V_{OL}. El valor de V_{OL} no debe exceder $V_{OL(max)}$ (tabla 8.3) y esto limita el valor de I_{OL} y de este modo el numero de cargas que se pueden impulsar.

La situación en estado ALTO se muestra en la figura 3.11(b). Aqui, Q_3 actua como un emisor-seguidor y abastece (suministra) de corriente a cada entrada TTL. Estas corrientes (I_{IH}) son simplemente corrientes de derivación con influencia invertida, ya que las uniones o diodos con base en el emisor de entrada TTL son

influenciados inversamente. Sin embargo, si se impulsan demasiadas cargas, la corriente total de salida I_{OH} se puede hacer demasiado grande, ocasionanado disminuciones mayores a través de R_2, Q_3 y D_1, con lo cual se lleva a V_{OH} debajo de $V_{OH(min)}$.

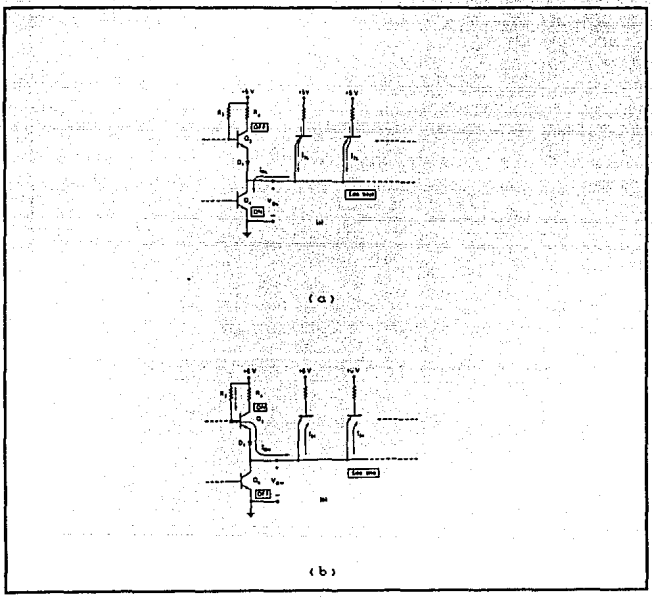


FIG. 3.11 CAPACIDADES DE IMPULSO DE SALIDAS TTL.

Cargas unitarias. A fin de simplificar el diseño con circuitos TTL, los fabricantes han establecido factores de carga de entrada y salida estandarizados en términos de la corriente. Estas corrientes se denominan cargas unitarias (UL) y se definen como sigue:

1 carga unitaria (UL) = 40 μA en el estado ALTO
1.6 mA en el estado BAJO

Estos factores de carga unitaria representan las corrientes máximas de entrada para la serie estándar. En otras palabras, la máxima corriente que fluye en la entrada TTL estándar en el estado ALTO es $I_{IH(max)}$ =40 μA y a la corriente máxima que fluye fuera de una entrada TTL estándar en el estado BAJO es $I_{IL(max)}$ =1.6 mA. Aunque los factores de carga unitaria se basan en la serie TTL estándar, se usa para expresar los requisitos de ingreso y las capacidades de impulso de salida en todas las series TTL.

La tabla 3.5 da los factores comunes de entrada y salida de las cinco series TTL. Estos valores son comunes, así que puede haber algunas variaciones, según el dispositivo o el fabricante en particular. Se puede consultar el manual de dispositivos para determinar los valores exactos.

Nótese que las series de más alta velocidad (74H00, 74S00) tiene factores de carga de entrada mayores y fan-outs que la serie 7400 estándar. Por el otro lado, las series de menor energía (74L00, 74LS00) tienen requisitos de carga de entrada menores y fan-outs menores.

SERIE TTL	CARGA DE ENTRADA ALTA	CARGA DE ENTRADA BAJA	FACTOR DE CARGA (UL) ALTA	FACTOR DE CARGA (UL) BAJA
7400	1 UL	1 UL	10 UL	10 UL
74H00	1.25 UL	1.25 UL	12.5 UL	12.5 UL
74L00	0.5 UL	0.1 UL	10 UL	2.5 UL
74S00	1.25 UL	1.24 UL	25 UL	12.5 UL
74LS00	0.5 UL	0.25 UL	10 UL	0.5 UL

TABLA 3.5 FACTORES COMUNES DE ENTRADA Y SALIDA DE LOS TTL.

3.5.4 OTRAS CARACTERISTICAS DEL TTL.

Propensión de las entradas TTL a 0. Ocasionalmente, surge la situación donde una entrada TTL debe conservarse normalmente en BAJO (LOW) y luego mandarla a ALTO (HIGH) por la acción de un interruptor mecánico, esto se ilustra en la figura 3.12 para la entrada en una emisión simple. La ES se activa en una transición positiva que ocurre cuando el interruptor se cierra momentáneamente. La resistencia R sirve para conservar la entrada T en BAJO mientras el interruptor está abierto. Debe tenerse precaución y conservar el valor de R lo suficientemente pequeño de manera que el voltaje creado o generado a través de él por la

corriente I_{IL} que fluye hacia afuera de la entrada de OS conectada a tierra no exceda $V_{IL(max)}$. De este modo, el valor de R esta dado por :

$$I_{IL} \times R_{max} = V_{IL(max)}$$

$$R_{max} = \frac{V_{IL(max)}}{I_{IL}}$$

R debe conservarse debajo de este valor para asegurar que la entrada de OS estará en un nivel BAJO aceptable en tanto que el interruptor esté abierto. El valor mínimo de R se determina por la corriente obtenida de la fuente de 5 V cuando el interruptor esta cerrado. En la practica, esta corriente debe ser minimizada conservando a R ligeramente debajo de R_{max}.

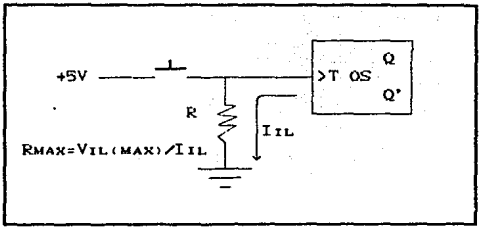


FIG. 3.12 PROPENSION DE LAS ENTRADAS TTL.

Ejemplo : *Determine el valor aceptable para R si OS es un IC TTL estandar con una clasificacion de entrada de 1 UL.*

Solución : A 1 UL, el valor de I_{IL} será un máximo de 1.6 mA. Este valor máximo debe utilizarse para calcular R_{max}. Para la lógica TTL estándar, $V_{IL(máx)}$ = 0.8 V. De este modo, se tiene :

$$R_{MAX} = \frac{0.8\ V}{1.6\ mA} = 500\ \Omega$$

Una elección adecuada aquí sería R =470 Ω, valor de resistencia estándar.

Ejemplo : *Repita el ejemplo anterior si OS pertenece a la serie LS-TTL con una clasificacion de entrada de 0.25 UL.*

Solucion : En 0.5 UL, tenemos $I_{IL(max)} = 0.4$ *mA. Para la serie LS-TTL,* $V_{IL\ (max)} = 0.8$ *V. De este modo,*

$$R_{MAX} = \frac{0.8\ V}{0.4\ mA} = 2\ K\Omega$$

Una eleccion eadecuada aqui seria 1.8 KΩ, valor estandar.

Tiempos de transicion. Las señales de entrada que impulsan circuitos TTL deben observar transiciones relativamente rapidas para lograr una operacion confiable. Los tiempos de elevacion y descenso que excedan de 50 ns pueden dar origen de oscilaciones en la salida de las compuertas logicas y en los INVERSORES (INVERTER), como se muestra en la figura 3.13(a), así como una activacion errónea de los FF y de la emision simple.

Una señal de transicion lenta puede ser acentuada pasándola a traves de un circuito Schmitt de disparo produce transiciones de salidas muy rapidas (comunmente de 10 ns) independientes de los tiempos de transicion de entrada. Varios circuitos integrados TTL incluyen circuitos Schmitt de disparo. Uno de ellos es el 7414, el cual es un integrado INVERSOR hexadecimal que contiene seis INVERSORES. Su operacion se ilustra en la figura 3.13(b). Nótese el simbolo del disparador Schmitt especial. Otro de ellos es el 7413 (integrado NAND de cuatro entradas doble).

Si se clasifica a un IC como uno que opera un disparador de Schmitt, este respondera confiablemente a señales de cambio lento; en caso contrario, no lo hara. Los diseñadores utilizan un IC disparador de Schmitt para convertir una señal de cambio lento en una de cambio rapido que pueda impulsar confiablemente cualquier circuito integrado.

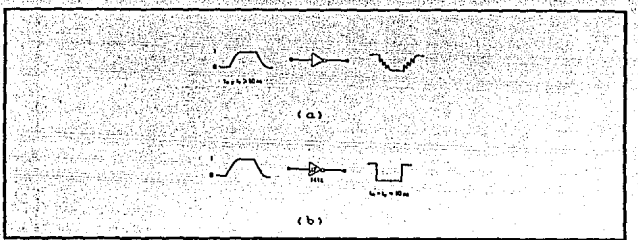


FIG. 3.13 (a) TR Y TF LENTOS HACEN QUE LAS SALIDAS TTL OSCILEN; (b) IC CON ACTIVADOR SCHMITT PARA REDUCIR TR Y TF.

Oscilaciones momentáneas de corriente. Los circuitos lógicos TTL sufren de fallas u oscilaciones momentáneas de corriente que se generan internamente debido a la estructura de salida con polo en forma de tótem. Cuando la salida cambia del estado BAJO al ALTO (figura 3.14), los dos transistores de salida cambian estados; Q_3 de APAGADO (OFF) a ENCENDIDO (ON) y Q_4 de ENCENDIDO (ON) a APAGADO (OFF). Ya que Q_4 cambia desde la condición saturada, le llevará más tiempo que a Q_3 en cambiar estados. Así, existe un intervalo de tiempo corto (cerca de 2 ns) durante la transición de cambios donde ambos transistores son conductores y un exceso relativamente grande de corriente (30-50 mA) es retirado de la fuente de +5 V. La duración de esta oscilación momentánea de corriente es ampliada por los efectos de cualquier capacitancia de carga en la salida del circuito. Esta capacitancia consta de la capacitancia de conexión en fuga y de la capacitancia de entrada de cualquier circuito de carga y debe cargarse con el voltaje de salida en estado ALTO. Este efecto integral se puede resumir como sigue: *Siempre que una salida TTL con polo en forma de tótem pasa de BAJO a ALTO se obtiene un transitorio de corriente de alta amplitud de la fuente de suministro Vcc.*

En un circuito o sistema digital complejo puede haber muchas salidas TTL cambiando estados al mismo tiempo, cada una generando un transitorio angosto de corriente de la fuente de energía. El efecto acumulativo de todas estas interrupciones de corriente tenderá a producir una falla de voltaje en la línea Vcc común, debido principalmente a la inductancia distribuida en la línea de suministro (recuérdese que $V = L\,(di/dt)$ para la inductancia y di/dt es muy grande para una interrupción de corriente de 2ns). Esta falla de voltaje puede ocasionar mal funcionamientos de gravedad durante las transiciones de cambios a menos que se utilice algún

tipo de filtrado. La técnica más común utiliza pequeños capacitores RF conectados de Vcc a TIERRA (GROUND) para "acortar" esencialmente estas fallas de alta frecuencia. A este proceso se le denomina desacoplamiento de suministro de energía.

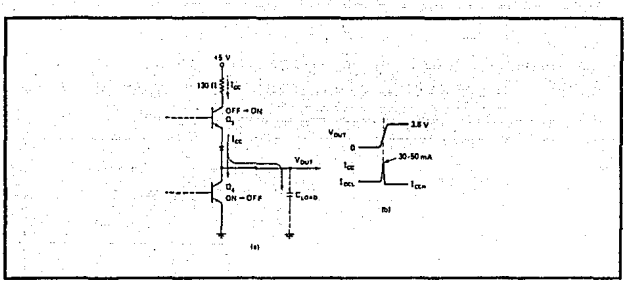


FIG. 3.14 UN TRANSITORIO GRANDE DE CORRIENTE SE DRENA DE Vcc CUANDO UNA SALIDA DE POLO EN FORMA DE TOTEM CAMBIA DE BAJO A ALTO.

Es una práctica normal conectar un capacitor de disco de baja inductancia de cerámica, de 0.01-μF o bien 0.1- μF entre Vcc y tierra cerca de cada IC TTL en un tablero de circuitos. Las terminales del capacitor se mantienen muy cortas para minimizar la inductancia.

Además es una práctica común conectar un solo capacitor grande (de 2 a 20 μF) entre Vcc y tierra en cada tablero para filtrar posibles variaciones en Vcc ocasionadas por las grandes variaciones en los niveles Icc cuando las salidas cambian estados.

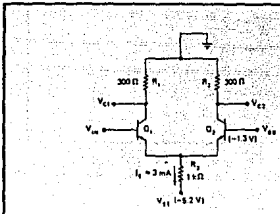
3.6 ECL Y SUS CARACTERISTICAS.

La familia TTL utiliza transistores que operan en el modo saturado. Como resultado, su velocidad de transición está limitada por la demora en el tiempo de almacenamiento asociada con un transistor que se conduce a saturación. Otra familia lógica bipolar ha sido creada y evita la saturación de transistores, con lo cual se incrementa la velocidad integral de transición. A esta familia lógica se le denomina lógica acoplada al emisor (ECL) y opera sobre el principio de la transición de corriente con lo cual

una corriente parcial fija menor que $I_{C(ON)}$ es cambiada del colector de un transistor a otro. Debido a esta operación en modo de corriente, esta forma lógica se conoce así mismo como lógica de modo de corriente (CML).

3.6.1 CIRCUITO ECL BASICO.

El circuito básico para la lógica acoplada al emisor es esencialmente la configuración de amplificador diferencial de la



V_{IN}	SALIDAS
-1.7 V (0 LOGICO)	V_{C1} = 0 V V_{C2} = -0.9 V
-0.8 V (1 LOGICO)	V_{C1} = -0.9 V V_{C2} = 0 V

FIG. 3.15 CIRCUITO ECL BASICO.

figura 3.15. La fuente de energía V_{EE} produce una corriente esencialmente fija I_E que permanece alrededor de 3mA durante la operación normal. Se permite que esta corriente fluya a través de Q_1 o de Q_2, según el voltaje en V_{IN}. En otras palabras, esta corriente cambiará entre el colector de Q_1 y el de Q_2 cuando V_{IN} varíe entre sus dos niveles lógicos de -1.7 V (0 lógico de ECL) y -0.8 V (1 lógico de ECL). La tabla de la figura 3.15 muestra los voltajes de salida resultantes en estas dos condiciones en V_{IN}. Deben observarse dos puntos importantes: (1) V_{C1} y V_{C2} son los complementos el uno del otro y (2) los niveles del voltaje de salida no son los mismos que los de los niveles lógicos de entrada.

El segundo punto observado antes se maneja fácilmente conectado V_{C1} y V_{C2} a las etapas seguidor-emisor (Q_3 y Q_4), como se muestra en la figura 3.16. Los seguidores-emisores desempeñan dos funciones: (1) restan aproximadamente 0.8 V de V_{C1} y V_{C2} para pasar los niveles lógicos ECL correctos y (2) ofrecen una impedancia de salida muy baja (comúnmente de 7 Ω), la cual

proporciona un factor de carga grande y un rápido abastecimiento de la capacitancia de carga. Este circuito produce dos salidas complementarias: V_{OUT1}, que es igual a V_{IN}' y V_{OUT2}, igual a V_{IN}.

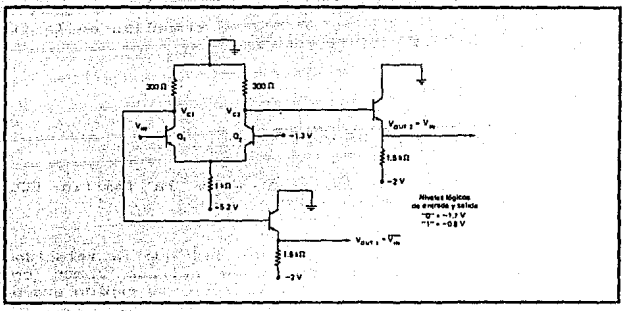


FIG. 3.16 CIRCUITO ECL BASICO CON ADICION DE EMISORES SEGUIDORES.

3.6.2 COMPUERTA OR Y NOR ECL.

El circuito ECL básico de la figura 3.16 se puede utilizar como un inversor si la salida se toma en V_{OUT1}. Este circuito básico se

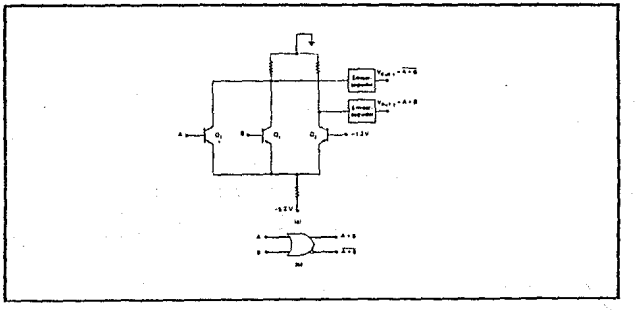


FIG. 3.17 (a) CIRCUITO NOR/OR ECL; (b) SIMBOLO LOGICO.

puede explicar con más de una entrada colocando en paralelo el transistor Q_1 con otros transistores para las otras entradas, como en la figura 3.17(a). Aquí, Q_1 o Q_2 pueden ocacionar que la corriente sea transferida fuera de Q_2, produciendo dos salidas, V_{OUT1} y V_{OUT2} que están en las operaciones lógicas OR y NOR, respectivamente. Esta compuerta OR/NOR se simboliza en la figura 3.17(b) y es la compuerta ECL fundamental.

3.6.3 CARACTERISTICAS DE ECL.

Las características más importantes de la familia ECL de circuitos lógicos son los siguientes:

1. Los transistores nunca se saturan, así que la velocidad de transición es muy alta. El tiempo de retraso común en la propagación es 2 ns, que hace del ECL un poco más rápido que el TTL Schottky (serie 74S00).

2. Los niveles lógicos son nominalmente -0.8 V y -1.70 V para el 1 y 0 lógicos, en forma respectiva.

3. Los márgenes de ruido de ECL en el peor de los casos son aproximadamente 250 mV. Estos márgenes de ruido bajos hacen del ECL un tanto inseguro para utilizarse en medios industriales de mucho trabajo.

4. Un bloque lógico ECL produce por lo general una salida y su complemento. Esto elimina la necesidad de inversores.

5. Los factores de carga (o fan-outs) se encuentran comúnmente alrededor de 25, debido a las salidas emisoras-seguidoras de baja impedancia.

6. La disipación básica de energía de una compuerta ECL básica es 25 mW, sólo ligeramente mayor que la TTL Schottky.

7. El flujo de corriente total en un circuito ECL permanece relativamente constante independientemente de su estado lógico. Esto ayuda a mantener un consumo de corriente invariante en el suministro de energía del circuito aun durante transiciones de cambio. De este modo, no se generarán ruidos internamente como

los producidos por los circuitos TTL con polo en forma de tótem.

La tabla 3.6 muestra la forma en que la familia ECL se compara con las familias lógicas TTL.

FAMILIA LOGICA	t_{pd} (ns)	P_D (mW)	MARGEN DE RUIDO EN EL PEOR DE LOS CASOS (mV)	INTENSIDAD MAX. DEL CRONOMETRO (MHz)
7400	9	10	400	25
74L00	33	1	400	3
74H00	6	23	400	40
74S00	3	23	300(V_{NL})	80
74LS00	9.5	2	300(V_{NL})	30
ECL	2	25	250	120

TABLA 3.6 COMPARACION CON LA FAMILIA TTL.

La familia ECL no se usa tan ampliamente como las familias TTL y MOS excepto en aplicaciones de muy alta frecuencia donde su velocidad es superior. Sus márgenes de ruido relativamente bajos y el alto o elevado consumo de energía son desventajas en comparación con las otras familias lógicas. Otra desventaja en su voltaje de suministro negativo y niveles lógicos, que no son compatibles con las otras familias lógicas; esto dificulta el uso de la familia ECL en conjunción con los circuitos TTL y MOS.

3.7 MOS Y CMOS Y SUS CARACTERISTICAS.

La tecnología MOS (semiconductor de óxido metálico) deriva su nombre de la estructura básica MOS de un electrodo metálico montado en un aislador de óxido sobre una base semiconductora. Los transistores de la tecnología MOS son transistores con efecto de campo denominados MOSFET. Muchos de los IC digitales MOS se construyen completamente con MOSFET y ninguna otra componente.

Las ventajas principales del MOSFET son que es relativamente simple y poco costoso de fabricar, es pequeño y consume muy poca energía. La fabricación de circuitos integrados MOS es aproximadamente un tercio tan compleja como la fabricación de circuitos integrados bipolares (TTL, ECL, etc.). Además los dispositivos MOS ocupan mucho menos espacio en un integrado que los transistores bipolares; comúnmente, un MOSFET requiere de un milésimo cuadrado del área de un IC integrado, pues los

transistores bipolares requieren cerca de 50 milésimos cuadrados. De mayor importancia es que los IC digitales MOS normalmente no utilizan los elementos de resistencia del IC, los cuales ocupan la mayor parte del area de los IC bipolares.

Todo esto significa que los circuitos integrados MOS pueden acomodar un numero mucho mayor de elementos del circuito en un solo integrado que los IC bipolares. Esta ventaja es reforzada por el hecho de que los IC MOS estan superando a los IC bipolares en el área de la integracion a gran escala (LSI). La alta densidad de empaquetamiento de los circuitos integrados MOS produce una mayor confiabilidad del sistema debido a la reduccion en el numero de conexiones externas que se necesitan.

Las desventajas principales de los ICMOS es su velocidad de operación relativanente lenta cuando se compara con la de las familias de IC bipolares. En muchas aplicaciones esta no es una consideración preponderante, de modo que la logica MOS ofrece una alternativa con frecuencia superior a la lógica bipolar.

3.7.1 CIRCUITOS MOSFET DIGITALES.

Los circuitos digitales que emplean MOSFET se dividen en tres categorías: (1) P-MOS, que utiliza sólo MOSFET de intensificación de canal P; (2) N-MOS, que utiliza solamente MOSFET de intensificación de canales N; y (4) CMOS (MOS complementaria), que usa dispositivos de canales P y N.

Los circuitos digitales P-MOS y N-MOS (integrados) tienen una mayor densidad de integración (mas transistores por integrado) y, por tanto, son más económicos que los CMOS. N-MOS tienen cerca de dos veces la densidad de integración de P-MOS. Además, N-MOS es así mismo casi dos veces más rapido que el P-MOS, debido al hecho de que los electrones libres son los transportadores de la corriente en N-MOS, en tanto que los agujeros (cargas positivas con un movimiento más lento) son los transportadores del P-MOS. El CMOS tiene la mayor complejidad y la menor densidad de integración de las familias MOS; pero posee las importantes ventajas de tener una mayor velocidad y mucho menos disipación de energía.

3.7.2 INVERSOR N-MOS.

La figura 3.18 muestra el circuito INVERSOR N-MOS básico. Este

contiene dos MOSFET de canal n: a Q_1 se le denomina MOSFET de carga y a Q_2 MOSFET de transición. Q_1 tiene su compuerta permanentemente conectada a +5 V, de modo, que siempre está en estado "ON" y actúa en esencia como una resistencia de carga del valor R_{ON}. Q_2 cambiará de "ON" a "OFF" en respuesta a V_{IN}. El MOSFET Q_1 está diseñado para tener un canal más estrecho que Q_2, de manera que R_{ON} de Q_1 es mucho mayor que el de Q_2. Comúnmente, R_{ON} de Q_1 es 100 kΩ y R_{ON} de Q_2 es 1 KΩ. R_{OFF} de Q_2 está por lo general alrededor de 10^{10} Ω.

Los dos estados del inversor se resumen en la figura 3.18(b). La mejor manera de analizar este circuito consiste en considerar a cada canal del MOSFET como una resistencia de manera que el voltaje de salida se tome de un divisor de voltajes formado por dos resistencias. Con V_{IN} =0 V, el transistor Q_2 está en "OFF", con una resistencia de 10^{10} Ω. Ya que Q_1 tiene R_{ON} =100 Ωk , la salida divisora del voltaje será esencialmente +5 V. Con V_{IN}=+5 V, Q_2 está en "ON", con R_{ON} =1 kΩ. El divisor del voltaje es ahora 100 kΩ y 1 kΩ, de modo que V_{OUT} =1/101 X(+5 V)=0.05 V.

El circuito funciona como un INVERSOR puesto que una entrada BAJA produce una salida ALTA y viceversa. Este INVERSOR básico puede ser modificado para formar compuertas lógicas NAND y NOR.

(a)

V_{IN}	Q_1	Q_2	$V_{OUT}=V_{IN}$'
0 V (0 LOGICO)	R_{ON}=100 KΩ	$R_{OFF}=10^{10}$ Ω	+5 V (1 LOGICO)
+5 V (0 LOGICO)	R_{ON}=100 KΩ	R_{OFF}=1 KΩ	+0.05 V (1 LOGICO)

(b)

FIG. 3.18 INVERSOR N-MOS.

3.7.3 COMPUERTA NAND N-MOS.

La operación NAND se efectúa por medio del circuito de la figura 3.19(a), donde Q_1 vuelve a actuar como una resistencia de carga mientras que Q_2 y Q_3 son interruptores controlados por los niveles de entrada A y B. Si A o B está en 0 V (0 lógico), el FET correspondiente está en "OFF", con lo cual se presenta una alta resistencia de la terminal de salida a tierra de manera que la salida X sea ALTA (+5 V). Cuando A y B son +5 V (1 lógico), Q_2 y Q_3 están en "ON", de modo que la salida X es BAJA. Con claridad, la salida es igual a la operación NAND de las entradas ($X=(AB)'$).

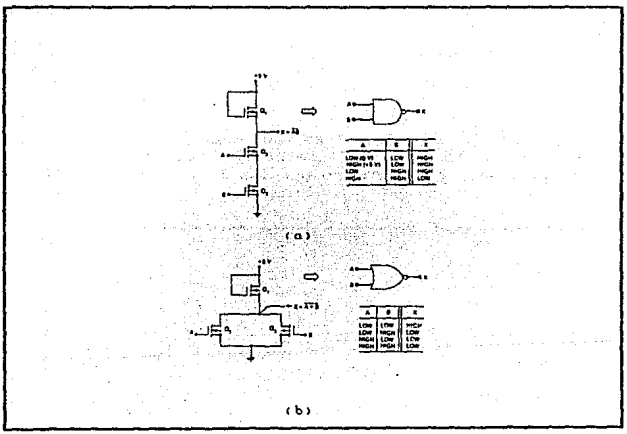


FIG. 3.19 (a) COMPUERTA NAND N-MOS; (b) COMPUERTA NOR.

3.7.4 COMPUERTA NOR N-MOS.

La compuerta NOR de la figura 3.19(b) utiliza Q_2 y Q_3 como interruptores paralelos con Q_1 que vuelve a actuar como una resistencia de carga. Cuando la entrada A o la B está en +5 V, el MOSFET correspondiente está en "ON", forzando la salida a ser BAJA. Cuando ambas entradas están en 0 V, Q_2 y Q_3 están en "OFF", de manera que la salida pasa a ALTA. Con claridad, ésta es la

operación NOR con X = (A+B)'.

Las compuertas OR N-MOS y las AND se forman fácilmente combinando la compuerta NOR o la NAND con inversores.

3.7.5 CARACTERISTICAS DE LA LOGICA MOS.

En comparación con las familias lógicas bipolares las familias lógicas MOS son más lentas en cuanto a la velocidad de operación, requieren de mucho menos energía, tienen un mejor margen de ruido, un mayor intervalo de suministro de voltaje y un factor de carga más elevado y, como se mencionó antes, requieren de mucho menos espacio (área en el integrado).

Velocidad de operación. Una compuerta NAND N-MOS común tiene un tiempo de retraso en la propagación de 50 ns. Esto se debe a dos factores: la resistencia de salida relativamente alta (100 kΩ) en el estado ALTO y la carga capacitativa representada por las entradas en los circuitos lógicos impulsados. Las entradas lógicas MOS tienen una muy alta resistencia de entrada ($>10^{12}\Omega$) y tienen una capacitancia de compuerta razonablemente alta (capacitor MOS), comúnmente de 2 a 5 picofarads. Esta combinación de R_{OUT} grande y C_{LOAD} grande sirve para incrementar el tiempo de transición.

Margen de ruido. Comúnmente los márgenes de ruido N-MOS están alrededor de 1.5 V cuando operan desde V_{DD} =5 V y serán proporcionalmente mayores para valores más grandes de V_{DD}.

Factor de carga o Fan-out. Debido a la resistencia de entrada extremadamente alta en cada entrada MOSFET, uno esperaría que las capacidades del factor de carga de la lógica MOS fueran virtualmente ilimitadas. Esto es esencialmente verídico para la operación dc o de baja frecuencia. Sin embargo, para frecuencias mayores que alrededor de 100 kHz, las capacitancias de entrada de la compuertaa ocasionan un deterioro en el tiempo de transición que se incrementa en proporción al número de cargas impulsadas. Aún así, la lógica MOS puede operar fácilmente en un factor de carga de 50, que es un tanto mejor que en las familias bipolares.

Consumo de energía. Los circuitos lógicos MOS consumen pequeñas cantidades de energía debido a las resistencias relativamente grandes que se utilizan. Para ilustrar esto, podemos calcular la disipación de energía del INVERSOR de la figura 3.18 en sus dos estados de operación.

1.- V_{IN} =0 V; $R_{ON(Q1)}$ =100 kΩ; $R_{OFF(Q2)}$ =10^{10}Ω. Por lo tanto, I_D, corriente de la fuente V_{DD}, = 0.05 nA, y P_D =5 V X 0.05 nA =0.25

2.- V_{IN} = +5 V; $R_{ON(Q1)}$ = 100 kΩ; $R_{ON(Q2)}$ = 1 kΩ. Por lo tanto, I_D = 5 V/101 kΩ = 50 μA y P_D = 5Vx50μA = 0.25 mW.

Esto da un promedio P_D de poco más de 0.1 mW para el INVERSOR. El bajo consumo de energía de la lógica MOS la hace adecuada para el LSI, donde muchas compuertas, biestables, etc., pueden caber en un IC integrados, sin ocasionar sobrecalentamiento que puede dañarla.

Complejidad del proceso. La lógica MOS es la familia lógica más simple de fabricar ya que utiliza sólo un elemento basico, un transistor N-MOS (o bien P-MOS). No requiere de otros elementos, como resistencias, diodos, etc. Esta característica, junto con su bajo P_D, la hace idealmente adecuada para LSI (memorias grandes, integrados de calculadora etc.), y aquí es donde la lógica MOS ha causado su más grande impacto en el campo digital. La velocidad de operación de N-MOS y P-MOS no es comparable con TTL, de manera que se ha hecho muy poco con ellos en aplicaciones SSI y MSI. De hecho, hay pocos circuitos lógicos MOS en las categorías SSI MSI (compuertas, biestables, contadores, etc.). Sin embargo, CMOS es competitiva en el área del MSI, que hasta ahora fue dominada por los TTL.

3.7.6 LOGICA MOS COMPLEMENTARIA.

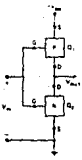
La familia lógica MOS complementaria (CMOS) utiliza los MOSFET de P y de N canales en el mismo circuito para obtener varias ventajas sobre las familias P-MOS y N-MOS. En términos generales, CMOS es más rápida y consume aun menos energía que las otras familias MOS. Estas ventajas son opacadas un poco por la elevada complejidad del proceso de fabricación de los circuitos integrados y una menor densidad de integración. De este modo, los CMOS todavía no pueden esperar competir con MOS en aplicaciones que requieren lo último en LSI.

Sin embargo, la lógica CMOS ha emprendido un crecimiento constante en el área de la MSI, principalmente a expensas del TTL, con la cual es directamente competitiva. El proceso de fabricación de CMOS es más simple que el TTL y tiene una mayor densidad de integración, con lo cual permite se tengan más circuitos en un área determinada y reduce el costo por función. Los CMOS utilizan

solamente una fracción de la energía que se necesita aun para la serie TTL de baja energía (74L00) y así se adapta en forma ideal a aplicaciones que utilizan la energía de una batería o con soporte en una batería. La velocidad de operacion de CMOS, no obstante, es comunmente de dos a cuatro veces más lenta que la serie LS-TTL y la TTL estandar.

3.7.6.1 INVERSORES CMOS.

Los circuitos del INVERSOR CMOS básico se muestra en la figura 3.20. Para este diagrama y los que siguen, los símbolos estandar de los MOSFET se han sustituido por bloques marcados como P y N para representar un P-MOSFET y un N-MOSFET, respectivamente. Esto se hace simplemente por conveniencia en el analisis de los circuitos. El inversor CMOS tiene dos MOSFET en serie tal que el dispositivo con canales P tiene su fuente conectadaa +VDD (un voltaje positivo) y el dispositivo de canales N tiene su fuente conectada a tierra. Las compuertas de los dos dispositivos se interconectan como una entrada común. Los consumos de los dispositivos se interconectan como la salida común.



(a)

VIN	Q1	Q2	VOUT
+VDD (1 LOGICO)	OFF $R_{OFF}=10^{10}\ \Omega$	ON $R_{ON} = 1\ K\Omega$	= 0 V
0 V (0 LOGICO)	ON $R_{ON}= 1\ K\Omega$	OFF $R_{OFF}=10^{10}\ \Omega$	= +VDD

VOUT = VIN'

(b)

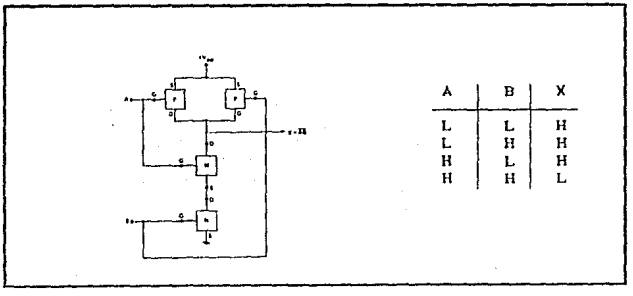
FIG. 3.20 INVERSOR CMOS BASICO.

Los niveles lógicos para CMOS son esencialmente +V_{DD} para 0 y 1 lógicos y 0 V para el 0 lógico. Consideremos primero el caso donde V_{IN} =+V_{DD}. En esta situación la compuerta de Q_1(canales P) está en 0 V en relación con la fuente de Q_1. De este modo, Q_1 estará en el estado "OFF" con R_{OFF} = $10^{10}\Omega$. La compuerta de Q_2 (canales N) estará en más V_{DD} en relación con su fuente. De este manera, Q_2 estará en "ON" comúnmente con R_{ON} =1 kΩ. El divisor de voltaje entre R_{OFF} de Q_1 y R_{ON} de Q_2 producirá V_{OUT}= 0 V.

A continuación, consideremos el caso donde V_{IN} =0 V. Q_1 tiene ahora su campuerta en un potencial negativo en relación con su fuente en tanto que Q_2 tiene V_{GS} =0 V. De este modo, Q_1 estará en "ON" con R_{ON} =1 kΩ y Q_2 en "OFF" con R_{OFF} =10^{10} Ω, produciendo un V_{OUT} de aproximadamente +V_{DD}. Estos dos estados de operación se resumen en la figura 3.20(b), donde se muestra que el circuito actúa como un INVERSOR lógico.

3.7.6.2 COMPUERTAS NAND CMOS.

Se pueden construir otras funciones lógicas modificando el INVERSOR básico. La figura 3.21 muestra una compuerta NAND formada por la edición de un MOSFET de canales P en paralelo y un MOSFET de canales N en serie al INVERSOR básico. Para analizar este circuito conviene recordar que una entrada de 0 V enciende el P-MOSFET y apaga el N-MOSFET correspondientes y viceversa para una entrada +V_{DD}. De este modo, puede observarse que la única vez que una salida BAJA ocurrirá es cuando las entradas A y B sean ambas



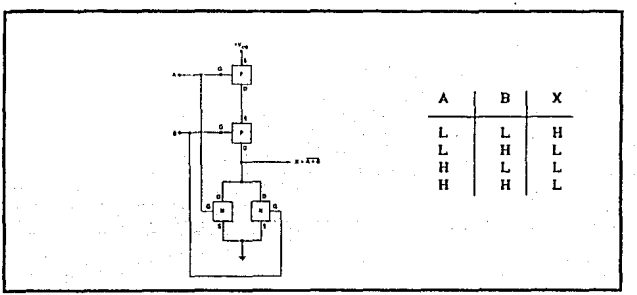
A	B	X
L	L	H
L	H	H
H	L	H
H	H	L

FIG. 3.21 COMPUERTA NAND CMOS.

ALTAS(+VDD) para encender ambos N-MOSFET, con lo cual ofrece una baja resistencia de la terminal de salida a tierra. En todas las otras condiciones de entrada, cuando menos un P-MOSFET estara en "ON" en tanto que al menos un N-MOSFET estara en "OFF". Esto produce una salida ALTA (HIGH).

3.7.6.3 COMPUERTAS NOR CMOS.

Una compuerta NOR CMOS se forma agregando un P-MOSFET en serie y un N-MOSFET en paralelo al inversor basico de la figura 3.22. Una vez más este circuito se puede analizar entendiendo que un



A	B	X
L	L	H
L	H	L
H	L	L
H	H	L

FIG. 3.22 COMPUERTA NOR CMOS.

estado BAJO (LOW) en cualquier entrada activa el, P-MOSFET y desactiva el N-MOSFET correspondientes y viceversa para una entrada ALTA. Se deja al lector verificar que este circuito opere como una compuerta NOR.

Las compuertas AND y OR CMOS se pueden formar combinando compuertas NAND y NOR con invertidores.

3.7.7 CARACTERISTICAS DE LA SERIE CMOS.

La primera serie lógica CMOS fue producida por RCA y se

denomina serie 4000; los circuitos integrados de esta serie se numeran 4000, 4001, etc. Algunos fabricantes utilizan este mismo sistema de numeración para sus dispositivos CMOS, mientras que otros han ideado sistemas de numeración íntimamente relacionados con éstos. Por ejemplo, los dispositivos CMOS de Motorola pertenecen a las series MC14000 y MC14500. Además, National Semiconductor ha creado la serie 74C00 que son equivalentes breves de la serie 7400 de dispositivos TTL. Por ejemplo, la serie 74C04 es un integrado INVERSOR hexadecimal que tiene las mismas asignaciones abreviadas que la 7404 y es lógicamente equivalente a la 7404.

Más recientemente, se ha creado una versión mejorada de la serie 4000. Denominada serie 400B, ofrece mejoras en velocidad de transición y conducción de salida sobre la serie 4000 original (que a menudo se conoce como la serie 4000A).

Los párrafos siguientes investigarán algunas de las características importantes que son comunes a muchos dispositivos CMOS.

Voltaje de suministro de energía. Muchos circuitos integrados CMOS de la serie 4000A operarán con voltajes V_{DD} que van de 3 a 15 V y muchos dispositivos de la seie 4000B pueden operar con valores V_{DD} de 3 a 18 V, de modo que la regulación del suministro de energía no constituya una consideración crítica. Cuando CMOS y TTL se utilizan en el mismo sistema, la fuente V_{DD} se hace por lo general igual a 5 V de manera que el suministro de 5 V pueda servir para ambos tipos de IC. En algunos casos los dispositivos CMOS serán operados en un voltaje de suministro mayor que 5 V. En estas situaciones, tienen que seguirse etapas especiales para permitir que los dispositivos CMOS y TTL trabajen juntos.

Niveles de voltaje. Los niveles de salida de un IC CMOS estarán muy próximos a 0 V en el estado BAJO y muy próximos a V_{DD} en el estado ALTO. La razón de esto es que la resistencia de una entrada CMOS es tan grande (10 ohms) que no carga una salida CMOS que la impulsa. Los requisitos de voltaje de entrada de los dos estados lógicos se especifican como un porcentaje del voltaje de suministro V_{DD} como se muestra en la tabla 3.7. El voltaje de entrada en estado BAJO mas elevado se especifica como el 30% de V_{DD} y el voltaje de entrada en estado ALTO más bajo se especifica como el 70% de V_{DD}. Por ejemplo, con V_{DD} =5 V, cualquier entrada CMOS menor que $V_{IL(max)}$ =1.5 V será aceptada como BAJO y cualquier entrada mayor que $V_{IH(mín)}$ 3.5 V será aceptada como ALTO.

	MINIMO	MAXIMO
V_{DD}	3 V	18 V
V_{OH}	V_{DD}	——
V_{IH}	70% V_{DD}	——
V_{OL}	——	0 V
V_{IL}	——	30% V_{DD}

TABLA 3.7 NIVELES DE VOLTAJE CMOS.

Margenes de ruido. Los márgenes de ruido de CMOS se pueden determinar a partir de la tabla 3.7 como sigue:

$$V_{NH} = V_{OH(mín)} - V_{IH(mín)}$$
$$= V_{DD} - 70\% V_{DD}$$
$$= 30\% V_{DD}$$

$$V_{NL} = V_{IL(máx)} - V_{OL(máx)}$$
$$= 30\% V_{DD} - 0$$
$$= 30\% V_{DD}$$

Los márgenes de ruido son los mismos en ambos estados y dependen de V_{DD}. En V_{DD} =5 V, los margenes de ruido son de 1.5 V. Esto es sustancialmente mejor que el TTL y ECL. Esto hace del CMOS una atractiva alternativa para aplicaciones que están expuestas a un medio con mucho ruido. Por supuesto, los márgenes de ruido pueden hacerse todavía mejores utilizando un valor mayor de V_{DD}. Esta mejora en inmunidad al ruido, no obstante, se obtendría a expensas de un mayor consumo de energía debido al mayor voltaje de suministro.

Disipación de energía. Cuando un circuito lógico CMOS se encuentra en estado estático (sin cambiar), su disipacion de energía es extremadamente baja. Podemos observar la razón de esto examinando cada uno de los circuitos que se muestran en las figuras 3.20 a 3.22. Nótese que independientemente del estado de la salida, siempre hay una muy alta resistencia entre la terminal V_{DD} y tierra debido a que siempre hay un MOSFET desactivado en la trayectoria de la corriente. Esto produce una disipación de energía de común del CMOS de solo 2.5 nW por compuerta cuando V_{DD}= 5 V; aun en V_{DD} =10 V esta energía aumenta a sólo 10 nW. Con estos valores de P_D, es facil observar por qué la familia CMOS se usa ampliamente en aplicaciones donde el consumo de energía es de interes primordial.

Factor de carga (fan-out). Al igual que N-MOS y P-MOS, las

entradas CMOS tienen una resistencia extremadamente grande (10 ohms) que esencialmente no drena corriente de la fuente emisora de señales. Sin embargo, cada entrada CMOS, representa comunmente una carga a tierra de 5 pF. Esta capacitancia de entrada limita el numero de entradas CMOS que una salida CMOS que puede impulsar (figura 3.23). La salida CMOS tiene que cargar y descargar la combinacion en paralelo de cada capacitancia de entrada, de manera que el tiempo de transicion de salida aumente en proporcion al numero de cargas impulsadas. Comunmente, cada carga CMOS aumenta la demora en la conduccion de la propagacion del circuito por 3 ns. Por ejemplo, la compuerta NAND 1 de la figura 3.23 podria tener un t_{PLH} de 25 ns si no impulsara cargas; esto incrementaria a 25 ns +20(3ns) =85 ns si condujera veinte cargas.

Así pues, el factor de carga de CMOS depende de la máxima demora permisible de la propagación. Comuunmente, las salidas CMOS son limitadas a un factor de carga de 50 para una operación en baja frecuencia (1 MHz). por supuesto, para una operación en frecuencia de más alta, el factor de carga debería ser menor.

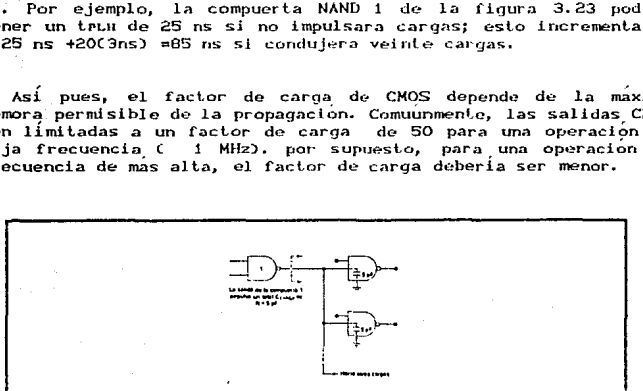


FIG. 3.23 CADA ENTRADA CMOS SUMA SU CONTENIDO A LA CAPACITANCIA TOTAL DE CARGA OBSERVADA POR LA SALIDA DE LA COMPUERTA DE IMPULSO.

Velocidad de transición. Aunque CMOS, al igual que N-MOS y P-MOS, tiene que impulsar capacitancias de carga relativamente grandes, su velocidad de transicion es mas rapida debido a su baja resistencia de salida en cada estado. Recordemos que una salida N-MOS tiene que cargar la capacitancia de carga a traves de una resistencia relativamente grande (100 kΩ). En el circuito CMOS, la resistencia de salida en el estado ALTO es el valor R_{ON} del P-MOSFET, el cual es generalmente de 1 kΩ o menor. Esto permite una carga mas rapida de la capacitancia de carga.

Una compuerta NAND CMOS tendrá por lo general un promedio t_{pd} de 50 ns en V_{DD} =5 V y 25 ns en V_{DD} =10 V. La razon de la mejora

en t_{pd} cuando V_{DD} es incrementado es que R_{ON} de los MOSFET decrece significativamente en voltajes de suministro mayores. De este modo, parece que V_{DD} debe hacerse tan grande como sea posible para operar en frecuencias más elevadas. Por supuesto, mientras más grande sea V_{DD} se producirá una mayor disipación de energía.

Entradas que no se usan. Las entradas CMOS nunca deben dejarse desconectadas. Todas las entradas CMOS tienen que estar conectadas a un nivel fijo de voltaje (0 V o bien V_{DD}) o bien otra entrada. Esta regla se aplica aun a las entradas de otras compuertas lógicas que no se utilizan en un integrado. Una entrada CMOS no conectada es susceptible al ruido y a cargas estáticas que pudieran fácilmente activar los canales MOSFET P y N en el estado conductor, produciendo una mayor disipación de energía y posible sobrecalentamiento.

Susceptibilidad a cargas estáticas. La resistencia alta de entrada de las entradas CMOS las hace especialmente propensas a una acumulación progresiva de cargas estáticas que puede producir voltajes lo suficientemente grandes para romper el aislamiento dieléctrico entre la compuerta y el canal del MOSFET. Esta carga estática puede generarse por un manejo inadecuado como introducir al integrado en un medio espumoso. Los circuitos integrados CMOS y MOS requieren de tomar precauciones en el manejo como en (1) su almacenamiento en espuma conductora o recipientes de metal y (2) el uso de un metal de soldadura conectado a tierra cuando se suelden en un circuito.

Muchos de los dispositivos de la serie CMOS están protegidos ahora contra daños por cargas estáticas mediante la inclusión de diodos protectores en cada entrada. Aun así, es mejor observar las reglas señaladas antes para evitar el daño por cargas estáticas.

Tiempos de transición de entrada. A semejanza de TTL, los dispositivos CMOS y MOS requieren transiciones razonablemente rápidas de las señales de entrada para una operación confiable, si bien ese requisito no es tan riguroso como en el caso de TTL. En general, los dispositivos CMOS funcionarán a nivel óptimo con las transiciones de señales de entrada que son menores que 15 μs cuando V_{DD} =5 V y menores que 4 μs cuando V_{DD} =10 V.

EJERCICIOS PROPUESTOS.

3.1 Dos circuitos lógicos tiene las siguientes características :

	CORRIENTE A	CORRIENTE B
V_{FUENTE}	6.0 V	5.0 V
V_{IH}	1.6 V	1.8 V
V_{IL}	0.9 V	0.7 V
V_{OH}	2.2 V	2.5 V
V_{OL}	0.4 V	0.3 V
t_{PLH}	10.0 ns	18.0 ns
t_{PHL}	8.0 ns	14.0 ns
P_D	16.0 mW	10.0 mW

(a) ¿Qué circuito tiene la mejor inmunidad al ruido dc en estado BAJO y en estado ALTO?

(b) ¿Qué circuito puede operar en frecuencias más altas?

(c) ¿Que circuito deriva la mayor cantidad de energía suministrada?

3.2 Determine los valores de $I_{IL(máx)}$, $I_{IH(máx)}$, $I_{OL(máx)}$ e $I_{OH(máx)}$ para cada una de las siguientes compuertas NAND de TTL :

(a) 7420 (b) 74S20 (c) 74H20 (d) 74LS20

3.3 Consulte el manual de información del FF J-K 74LS112.

(a) Determine el factor de carga de entrada en las entradas J y K.

(b) Determine el factor de carga de entrada en las entradas de cronometro y limpieza.

(c) ¿Cuántas otras unidades 74LS112 pueden impulsar la salida de una unidad 74LS112 en la entrada del cronometro?

3.4 La figura 3.24 muestra un circuito monoestable 74121 que es activado por el cierre del interruptor. ¿Que valor de R debe utilizarse para asegurar que la entrada B tiende a BAJO mientras el interruptor esta abierto.

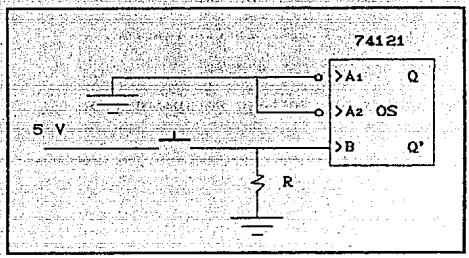


FIG. 3.24

3.5 ¿Cuáles de las siguientes son ventajas que ECL tiene sobre TTL convencional (serie 7400):

(a) Menor disipación de energía.

(b) t_{pd} más corto.

(c) Factor de carga mayor.

(d) Mayor inmunidad al ruido.

(e) Salidas complementarias.

(f) Sin fallas de corriente durante la transición.

3.6 El circuito de la figura 3.25 es una compuerta lógica N-MOS. Determine que tipo de compuerta es esta. Suponga que +16V = 1 lógico, 0V = 0 lógico.

3.7 ¿Cuáles de las siguientes son ventajas que la lógica P-MOS y N-MOS tiene sobre TTL?

(a) Mayor densidad de integración.

(b) Mayor velocidad de operación.

(c) Factor de carga mayor.

(d) Más adecuado para LSI.

(e) P_D inferior.

(f) Salidas complentarias.

(g) Mayor inmunidad al ruido.

(h) Proceso más simple de fabricación.

(i) Más funciones SSI y MSI.

(j) Utiliza menor voltaje de suministro.

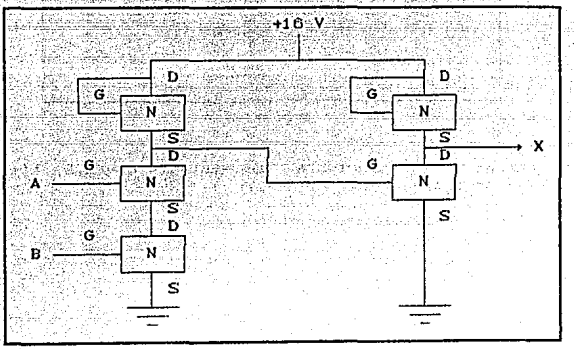


FIG. 3.25

3.8 ¿Cuáles de las siguientes son ventajas que CMOS tiene sobre P-MOS / N-MOS?

(a) Mayor densidad de integración.

(b) Mayor velocidad.

(c) Factor de carga mayor.

(d) Menor impedancia de salida.

(e) Proceso de fabricación más simple.

(f) Más adecuado para LSI.

(g) Menor P_D.

(h) Utiliza transistores como único elemento del circuito.

(i) Menor capacitancia de entrada.

3.9 Repita el problema 3.8 para las ventajas de los CMOS sobre los TTL.

3.10 ¿Cuáles son los margenes de ruido de la lógica CMOS que opera en V_{DD} = 12V?

CAITULO IV : CIRCUITOS COMBINACIONALES

CONTENIDO :

CAPITULO IV : CIRCUITOS COMBINACIONALES.

Los circuitos lógicos para sistemas digitales pueden ser combinacionales o secuenciales. Un circuito combinacional consta de compuertas logicas cuyas salidas en cualquier momento estan determinadas en forma directa por la combinacion presente de las entradas sin tomar en cuenta las entradas previas. Un circuito combinacional realiza una operacion especifica de procesamiento de información, especificada por completo en forma logica por un conjunto de funciones booleanas. Los circuitos secuenciales emplean elementos de memoria (celdas binarias) ademas de las compuertas logicas. Sus salidas son una funcion de las entradas y el estado de los elementos de memoria. El estado de los elementos de memoria, a su vez, es una funcion de las entradas previas. Como consecuencia, las salidas de un circuito secuencial dependen no solo de las entradas presentes, sino tambien de las entradas del pasado y, el comportamiento del circuito debe especificarse en una secuencia de tiempo de entradas y de estados internos.

Un circuito combinacional consta de variables de entrada, compuertas lógicas y variables de salida. Las compuertas logicas aceptan las señales de las entradas y generan señales a las salidas. Este proceso transforma la información binaria de los datos dados de entrada en los datos requeridos de salida. En forma obvia, tanto los datos de entrada y salida se representan por señales binarias, esto es, existen en dos valores posibles, uno representa la logica 1 y el otro la logica 0. En la figura 4.1 se muestra un diagrama de bloques de un circuito. Las n variables binarias de entrada provienen de una fuente externa; las m variables de salida van a un destino externo. En muchas aplicaciones, la fuente y/o destino son registros de almacenamiento localizados ya sea en la proximidad del circuito combinacional o en un dispositivo externo remoto. Por definicion, un registro externo no influencia el comportamiento del circuito combinacional ya que, si lo hace, el sistema total se vuelve un

circuito secuencial.

FIG. 4.1 DIAGRAMA DE BLOQUES DE UN CIRCUITO COMBINACIONAL.

Para las n variables de entrada, hay 2^n combinaciones posibles de los valores binarios de entrada. Para cada combinación posible de entrada, hay una y solo una combinacion posible de salida. Un circuito combinacional puede describirse por n funciones booleanas, una para cada variable de salida. Cada funcion de salida se expresa en terminos de las n variables de entrada.

Cada variable de entrada a un circuito combinacional puede tener uno o dos alambres. Cuando está disponible solo un alambre, puede representar la variable, ya sea en forma normal (sin prima) o en forma complementaria (con prima). Ya que una variable en una expresion booleana puede aparecer con prima y/o sin prima, es necesario proporcionar un inversor para cada literal que no esta disponible en el alambre de entrada. Por otra parte, una variable de entrada puede aparecer en dos alambres, suministrando las formas tanto normal como complentaria a la entrada del circuito. En este caso, no es necesario incluir inversores para las entradas. El tipo de celdas binarias utilizadas en la mayoría de los sistemas digitales son circuitos FLIP-FLOP, que tienen salidas para los valores tanto normal como complentario de la variable binaria almacenada.

4.1. SUMADORES.

Las computadoras y calculadoras realizan la operación de adición con dos numeros binarios a la vez, donde cada numero binario puede tener varias cifras decimales. La figura 4.2 ilustra la adicion de dos numeros de 5 bits.

El proceso de adición se inicia sumando los bits menos significativos (LSB) del cosumando y del sumando. Así, 1+1=10, lo cual significa que la suma para esa posicion es 0 con un corrimiento (acarreo) de 1.

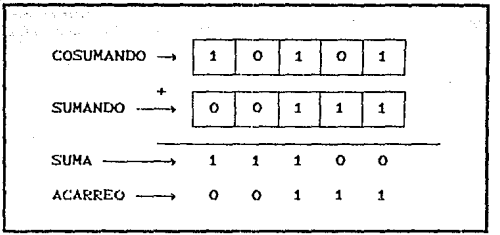


FIG. 4.2 PROCESO COMUN DE ADICION BINARIA.

Este corrimiento tiene que sumarse a la siguiente posición junto con el cosumando y el sumando de esta posicion. De este modo, en la segunda posicion, 1+0+1= 10, que es una vez mas una suma de 0 y un corrimiento de 1. Este corrimiento se suma a la siguiente posicion y así para las restantes posiciones.

En cada paso de este proceso de adición se efectúa la suma de 3 bits; el bit del cosumando, el bit del sumando y el bit del corrimiento de la posicion anterior. El resultado de la adicion de estos tres bits produce 2 bits: un bit de suma y uno de corrimiento, el cual se sumará a la siguiente posicion. Debe estar claro que se sigue el mismo proceso por cada posicion del bit. Como tal, si podemos diseñar un circuito lógico que pueda duplicar este proceso, entonces simplemente tenemos que emplear circuitos idénticos para cada una de las posiciones del bit. Esto se ilustra en la figura 4.3.

En este diagrama las variables A_4, A_3, A_2, A_1 y A_0 representan los bits del cosumando que están almacenados en el acumulador (que también se domina registro A). Las variables B_4, B_3, B_2, B_1 y B_0 son los bits del sumando almacenados en el registro B. Las variables C_4, C_3, C_2, C_1 y C_0 representan los bits del corrimiento que estan en las posiciones correspondientes. Las variables S_4,

S3, S2, S1 y So son los bits de salida de la suma por cada posicion. Los bits correspondientes del cosumando y del sumando se alimentan a un circuito logico llamado sumador total, junto con un bit de corrimiento (acarreo) de la posición anterior. Por ejemplo, los bits A1 y B1 se alimentan al sumador total 1 junto con C1, que es el bit del corrimiento producido por la adicion de los bits Ao y Bo. Los bits Ao y Bo se alimentan al sumador total 0 junto con Co. Ya que Ao y Bo son los LSB del cosumando y del sumando, parece ser que Co siempre tendra que ser 0, puesto que no puede haber corrimiento hacia esa posicion. Sin embargo, observaremos que habra situaciones donde Co tambien pueda ser 1.

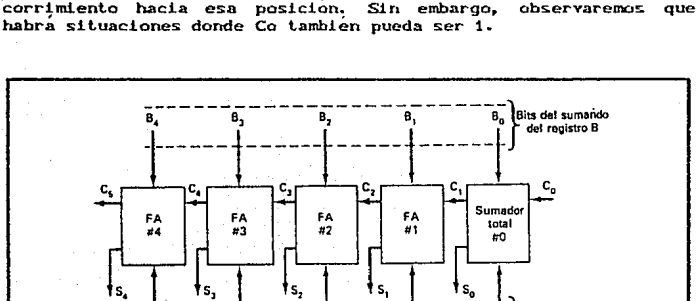


FIG. 4.3 DIAGRAMA DE BLOQUE DE UN CIRCUITO SUMADOREN PARALELO DE 5 BITS QUE EMPLEA SUMADORES TOTALES.

El circuito sumador total que se utiliza en cada posición tiene tres entradas: un bit A, un bit B y un bit C y produce dos salidas: un bit de suma y uno de corrimiento. Por ejemplo, el sumador total 0 tiene las entradas Ao, Bo y Co produce salidas So y C1. El sumador total 1 tiene como entradas A1, B1 y C1 y como salidas S1 y C2; y así sucesivamente. Esta disposicion se repite en tantas posiciones como haya en el cosumando. Aunque en esta ejemplificacion para numeros de 5 bits, en las computadoras modernas los numeros por lo general van de 8 a 64 bits.

La disposición de la figura 4.3 recibe el nombre del sumador en paralelo ya que todos los bits del cosumando están presentes y se alimentan a los circuitos sumadores simultaneamente. Esto significa que las adiciones en cada posición se llevan a cabo al mismo tiempo. Este procedimiento es distinto del que se sigue al sumar en papel, ya que se toma cada posición a la vez empezando con el LSB. Con toda claridad, la adición en paralelo es extremadamente rapida.

4 . 1 . 1 . MEDIO SUMADOR.

De la explicación verbal del medio sumador, se encuentra que este circuito necesita dos entradas binarias. Las variables de entrada designan los bits sumando y cosumando; las variables de salida producen la suma y el acarreo. Es necesario especificar dos variables de salida debido a que el resultado puede constar de dos dígitos binarios. Se asignan en forma arbitraria los símbolos A y B a las dos entradas y S (de suma) y C (para el acarreo) a las salidas.

Ahora que se han establecido el número y nombres de las variables de entrada y salida, ya puede formularse una tabla de verdad para identificar en forma exacta la función del medio sumador. Esta tabla de verdad se muestra a continuación:

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

TABLA 4.1 TABLA DE VERDAD PARA EL MEDIO SUMADOR.

El acarreo de salida es 0 a menos que ambas entradas sean 1. La salida S representa el bit menos significativo de la suma.

La función booleana simplificada de las dos salidas puede obtenerse de manera directa mediante la tabla de verdad. Las expresiones simplificadas en suma de productos son:

$$S = A'B + AB'$$

$$C = AB$$

El diagrama lógico para esta implementación se muestra en la figura 4.4(a), lo mismo que otras cuatro implementaciones para un medio sumador. Todas logran el mismo resultado en lo que respecta en el comportamiento de entrada-salida. Ilustra la flexibilidad de la que dispone el diseñador cuando implementa incluso una función lógica combinacional simple como ésta.

Como se mencionó antes, la figura 4.4(a) es la implementación del medio sumador en suma de productos. En la figura 4.4(b) se muestra la implementacion en producto de sumas:

$$S = (A+B)(A'+B')$$

$$C = AB$$

Para obtener la implementación de la figura 4.4(c), se observa que S es la OR excluyente de A y B :

$$S' = AB + A'B'$$

pero C = AB y, por lo tanto, tenemos:

$$S = (C + A'B')'$$

En la figura 4.4(d) se utiliza la implementación de producto de sumas con C derivada como sigue:

$$C = AB = (A' + B')'$$

el medio sumador puede implementarse con una compuerta OR excluyente y AND, como se muestra en la figura 4.4(e). Esta formula se usa posteriormente para mostrar que son necesarios dos circuitos medio sumadores para construir un circuito sumador completo.

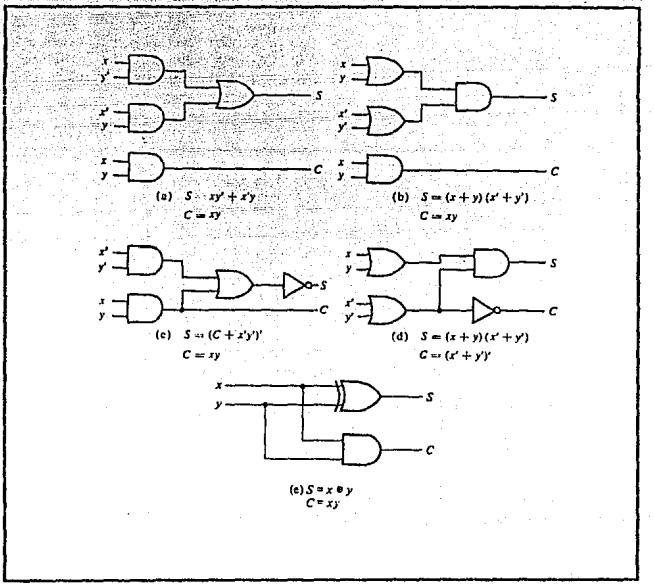


FIG. 4.4 VARIAS IMPLEMENTACIONES DE UN MEDIO SUMADOR.

4.1.2. DISEÑO DE UN SUMADOR COMPLETO (O TOTAL).

Ahora que conocemos la función del sumador completo, podemos proceder a diseñar un circuito lógico que realice esta función. Primero, debemos elaborar una tabla de verdad que muestre los diversos valores de entrada y salida en todos los casos posibles. La figura 4.5 muestra la tabla de verdad que tiene las tres entradas A, B y CIN y dos salidas, S y COUT. Hay ocho casos posibles para las tres entradas y en cada caso los valores de

salida que se buscan son enlistados. Por ejemplo, consideremos el caso A= 1, B= 0 y CIN= 1. El sumador total (que de aquí en adelante se abreviará FA) debe sumar estos bits para producir una suma (S) de 0 y un corrimiento (COUT) de 1. El lector debe verificar los otros casos para estar seguro que los ha comprendido.

Entrada del bit del cosumando A	Entrada del bit del sumando B	Entrada del bit de corrimiento C_{IN}	Salida del bit de la suma S	Salida del bit de corrimiento C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

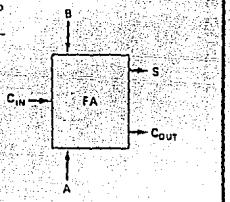


FIG. 4.5 TABLA DE VERDAD PARA UN CIRCUITO SUMADOR TOTAL.

Puesto que hay dos salidas, diseñaremos los circuitos para cada salida en forma individual, comenzando con la salida S. La tabla de verdad indica que hay cuatro casos donde S será 1. Utilizando el método, de la suma de productos, podemos escribir la expresión para S como :

$$S = A'B'C_{IN} + A'BC_{IN}' + AB'C_{IN}' + ABC_{IN}$$

Ahora, podemos intentar simplificar esta expresión factorizándola. Desafortunadamente, ninguno de los términos de la expresión tiene dos variables en común con alguno de los otros términos. Sin embargo, A', puede factorizarse en los dos primeros términos y A, en los dos últimos:

$$S = A'(B'C_{IN} + BC_{IN}') + A(B'C_{IN}' + BC_{IN})$$

El primer término entre paréntesis debe reconocerse como la combinación OR-exclusiva de B y C_{IN}, lo cual se puede escribir como $B \oplus C_{IN}$. El segundo término entre paréntesis debe reconocerse como el NOR-exclusivo de B y C_{IN}, que se puede escribir como $B \odot C_{IN}$. Así la expresión para S se transforma en :

$$S = A'(B \oplus C_{IN}) + A(B \odot C_{IN})$$

Si hacemos $X = B \oplus C_{IN}$, esto se puede escribir como :

$$S = A'X + AX' = A \oplus X$$

Que es simplemente el EX-OR de A y X. Al sustituir la expresión para X, se tiene :

$$S = A \oplus (B \oplus C_{IN})$$

Consideremos ahora la salida C_{OUT} en la tabla de verdad de la figura 4.5. Podemos escribir la expresión en suma de productos para C_{OUT} como sigue:

$$C_{OUT} = A'BC_{IN} + AB'C_{IN} + ABC_{IN}' + ABC_{IN}$$

Esta expresión puede simplificarse factorizándola. Utilizaremos tres veces el término ABC_{IN} ya que tiene factores comunes con cada uno de los otros términos. De aquí,

$$C_{OUT} = BC_{IN}(A' + A) + AC_{IN}(B' + B) + AB(C_{IN}' + C_{IN})$$

$$= BC_{IN} + AC_{IN} + AB$$

Esta expresión no puede simplificarse más.

Las expresiones finales para S y C_{OUT} pueden llevarse a la

práctica como se muestra en la figura 4.6. Existen otras alternativas que pueden servir para producir las mismas expresiones para S y Cout, ninguna de las cuales tiene ninguna ventaja en particular sobre las que se muestran. El circuito completo con las entradas A, B y Cin y las salidas S y Cout es el sumador total. Cada uno de los FA de la figura 4.3 contiene este mismo circuito (o bien su equivalente).

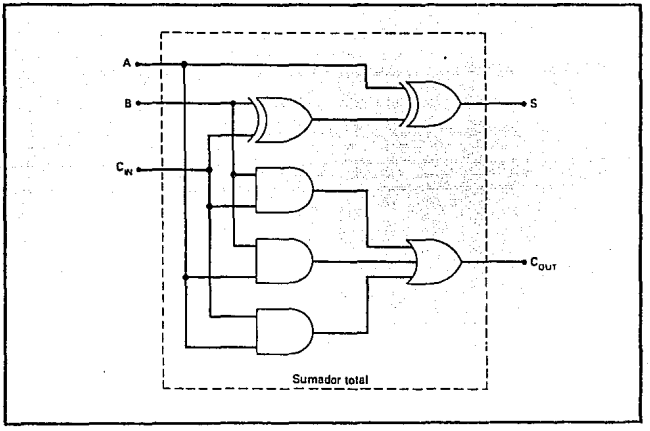


FIG. 4.6 CIRCUITOS COMPLETOS PARA UN SUMADOR TOTAL.

Simplificación con el mapa K. Simplificamos las expresiones para S y Cout utilizando métodos algebraicos. El método del mapa K tambien puede servir a este fin. La figura 4.7(a) muestra el mapa K para la salida S. Este mapa no tiene adyacentes, de manera que no hay ni pares ni cuadruples para repetir. De este modo, la expresion para S no puede ser simplificada mediante el uso del mapa K. Esto destaca una limitacion del método del mapa K en comparación con el metodo algebraico. Pudimos simplificar la expresion para S factorizando y atraves del uso de las operaciones EX-OR y EX-NOR.

El mapa K para la salida Cout se presenta en la figura 4.7(b). Los tres pares que se repiten producirán la misma expresion que se obtuvo por el metodo algebraico.

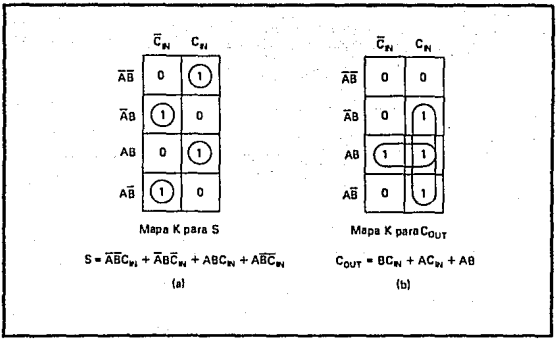


FIG. 4.7 MAPAS DE KARNAUGH PARA LAS SALIDAS DE UN SUMADOR TOTAL.

4.1.3. SUMADOR EN PARALELO CON CIRCUITOS INTEGRADOS.

Se dispone de varios circuitos sumadores en paralelo como circuitos integrados. Uno de los más comunes es un paquete sumador en paralelo de 4 bits que contiene cuatro FA interconectados y los circuitos de corrimiento al frente que se necesitan para operar a una alta velocidad. La version TTL de este IC es el 7483 (tambien el 74283) y la versión CMOS es el 4008.

La figura 4.8 muestra el símbolo lógico del sumador paralelo 7483 de 4 bits. Las entradas de este circuito integrado son dos numeros de 4 bits, A_3 A_2 A_1 A_0 y B_3 B_2 B_1 B_0 y el corrimiento, C_0, a la posicion del LSB. Las salidas son los cuatro bits de la suma S_3 S_2 S_1 S_0 y el corrimiento, C_4, del MSB.

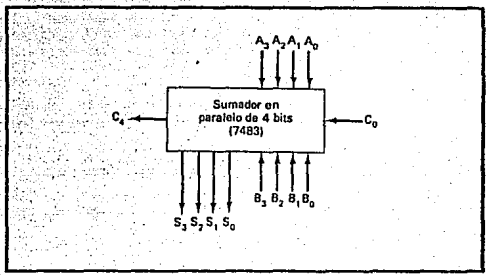


FIG. 4.8 SIMBOLO DE BLOQUE PARA EL SUMADOR DE 4 BITS 7483.

4.1.4. SUMADORES PARALELOS CON TRANSMISION EN CASCADA.

Dos o más bloques sumadores paralelos pueden conectarse en cascada para acomodar la adición de números binarios mayores. Para ilustrar esto, La figura 4.9 muestra la forma en que dos sumadores 7483 pueden conectarse para sumar dos números de 8 bits. El sumador de la derecha suma los cuatro dígitos menos significativos de los números. La salida C_4 de este sumador se conecta como el corrimiento de entrada a la primera posición del segundo sumador, el cual suma los cuatro bits más significativos de los números. Los ocho resultados de la suma resultante de dos números de 8 bits. C_8 es el corrimiento que sale de la última posición (MSB) del segundo sumador. C_8 se puede utilizar como un bit de sobrepaso o como un corrimiento en otra fase del sumador si se manejan números binarios mayores.

4.1.5. EL SUMADOR BCD.

El proceso de la adición BCD se muestra a continuación:

1. Sumar los grupos de código BCD para cada posición de dígito decimal; utilizar la adición binaria ordinaria.

2. Para aquellas posiciones donde la suma es 9 o menor, la suma se encuentra en forma BCD adecuada y no se necesita hacer correción.

3. Cuando la suma de dos cifras es mayor que 9, debe agregársele una corrección de 0110 para producir el resultado BCD indicado. Esto producirá un acarreo que se sumará a la siguiente posicion decimal.

Un circuito sumador BCD debe poder operar de acuerdo con los pasos citados antes. En otras palabras, el circuito debe poder realizar lo siguiente:

1. Sumar dos grupos de código BCD de 4 bits, utilizando la adición binaria directa.

2. Determinar si la suma de esta adición es mayor que 1001 (9 decimal); si lo es, sumar 0110 (6) a esta suma y generar un corrimiento a la siguiente posicion decimal.

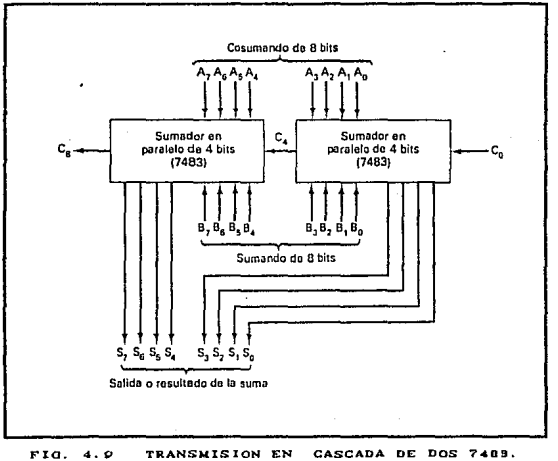


FIG. 4.9 TRANSMISION EN CASCADA DE DOS 7483.

El primer requisito se cumple fácilmente utilizando un sumador paralelo binario de 4 bits como el IC 7483. Por ejemplo, si los dos grupos de codigo BCD representados por A_8 A_2 A_1 Ao y B_8 B_2 B_1 Bo, respectivamente, se aplican a un sumador paralelo de 4 bits, el sumador realizara la siguiente operacion:

A_8 A_2 A_1 Ao	⟵ GRUPO DE CODIGO BCD
+ B_8 B_2 B_1 Bo	⟵ GRUPO DE CODIGO BCD
S_4 S_8 S_2 S_1 So	⟵ SUMA BINARIA DIRECTA

S_4 es en realidad C_4, el corrimiento que sale del MSB.

Las salidas de la suma S_4 S_8 S_2 S_1 So pueden variar de 00000 a 10010 (cuando ambos grupos de codigo BCD sean 1001 = 9). Los circuitos para un sumador BCD deben incluir la logica que se necesita para detectar siempre que las salidas S_4 S_8 S_2 S_1 So sean mayores que 01001, de manera que se pueda agregar la correcion. Estos casos donde la suma es mayor que 01001 se enlistan a continuacion:

Definamos a X como una salida lógica que pasará a ALTO (HIGH) sólo cuando la suma sea mayor que 01001 (es decir, en los casos que se enlistaron antes). Si examinamos estos casos, se puede razonar que X será ALTA en cualquiera de las condiciones siguientes:

1. Siempre que S_4= 1 (sumas mayores que 15).
2. Siempre que S_8= 1 y S_2 o bien S_1, o ambos, sean 1 (sumas 10-15).

S_4	S_8	S_2	S_1	So	
0	1	0	1	0	(10)
0	1	0	1	1	(11)
0	1	1	0	0	(12)
0	1	1	0	1	(13)
0	1	1	1	0	(14)
0	1	1	1	1	(15)
1	0	0	0	0	(16)
1	0	0	0	1	(17)
1	0	0	1	0	(18)

Esto se puede expresar como:

$$X = S_4 + S_3(S_2 + S_1)$$

Siempre que X = 1, se necesita sumar la correción 0110 a los bits de la suma y generar un corrimiento. La figura 4.10 muestra los circuitos completos de un sumador BCD, incluyendo la puesta de acción del circuito logico para X.

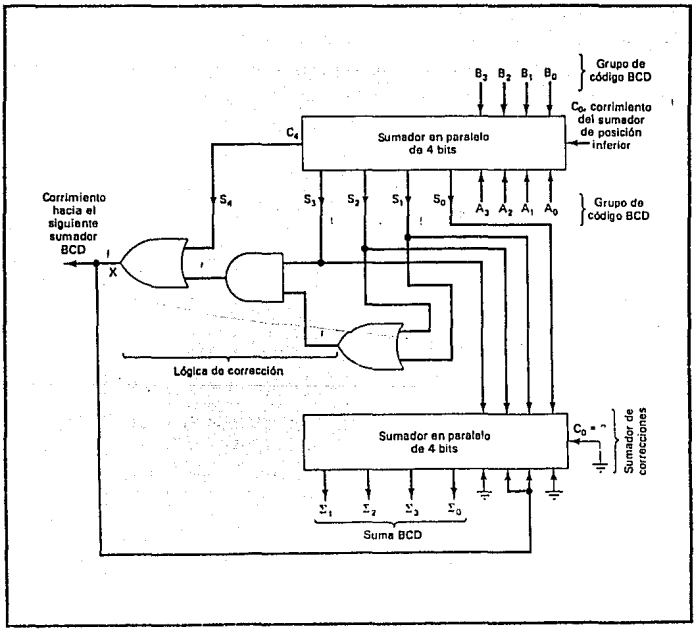


FIG. 4.10 UN SUMADOR BCD CONTIENE DOS SUMADORES DE 4 BITS Y UN CIRCUITO DETECTOR DE CORRECCIONES.

El circuito consta de tres partes básicas. Los dos grupos de código BCD A_3-A_0 y B_3-B_0 se suman en el sumador superior de 4 bits para producir la suma S_4 S_3 S_2 S_1 S_0. Las compuertas lógicas activan la expresión para X. El sumador inferior de 4 bits sumará la corrección 0110 a los bits de la suma solo cuando X = 1, produciendo la salida final de la suma BCD representada por $\sum_3$ $\sum_2$ $\sum_1$ $\sum_0$ (la letra griega , sigma, es un símbolo matemático común para suma). X es asimismo la salida del corrimiento que se produce cuando la suma es mayor que 01001. Desde luego, cuando X = 0, no hay corrimiento y tampoco adición de 0110. En tales casos, $\sum_3$ $\sum_2$ $\sum_1$ $\sum_0$ = S_3 S_2 S_1 S_0.

Para ayudar a entender la operación del sumador BCD, el lector debe ensayar varios casos siguiéndolos a través del circuito. Los siguientes casos serían particularmente instructivos:

Entradas :

(a) [A] = 0101, [B] = 0011, Co = 0
(b) [A] = 0111, [B] = 0110, Co = 0

Salidas :

(a) [S] = 01000, X = 0, [$\sum$] = 1000, CARRY = 0
(b) [S] = 01101, X = 1, [$\sum$] = 0011, CARRY = 1

4.1.5.1. SUMADORES BCD CON TRANSMISION EN CASCADA.

El circuito de la figura 4.10 se emplea para sumar dos cifras decimales, que se han codificado en código BCD. Cuando van ha sumarse números digitales con varias cifras, se necesita utilizar un sumador BCD aparte para la adición de dos números decimales de 3 cifras. El registro A contiene 12 bits, que son los tres grupos de código BCD para uno de los números decimales de tres cifras; en forma análoga, el registro B contiene la representación BCD del otro número decimal de tres cifras. Los grupos de código A_3-A_0 y B_3-B_0 que representan los dígitos menos significativos se alimentan al primer sumador BCD. Cada bloque sumador BCD se supone contiene los circuitos de la figura 4.10. Este primer sumador BCD produce las salidas de la suma $\sum_3$ $\sum_2$ $\sum_1$ $\sum_0$, que es el

código BCD del dígito menos significativo de la suma. También produce una salida del corrimiento que se envía al segundo sumador BCD, que suma A_7-A_4 y B_7-B_4, los grupos de código BCD para la segunda posición del dígito decimal. El segundo sumador BCD produce Σ_7 Σ_6 Σ_5 Σ_4, el código BCD para la segunda cifra de la suma, y así sucesivamente. Esta disposición puede, por supuesto, ampliarse a números decimales de cualquier tamaño.

4 . 2 . RESTADORES.

La sustracción de dos números binarios puede llevarse a cabo tomando el complemento del sustraendo y agregándolo al minuendo. Por este método, cada bit sustraendo del número se sustrae de su bit minuendo correspondiente significativo para formar un bit de diferencia. Si el bit minuendo es menor que el bit sustraendo , se toma 1 de la siguiente posición significativa. El hecho de que se ha tomado un 1 debe llevarse al siguiente par mas alto de bit mediante una señal binaria que llega de fuera (salida) de una etapa dada y va a (entrada) la siguiente etapa más alta. En forma precisa así como hay medio sumadores y sumadores completos, hay medio restadores y restadores completos.

4 . 2 . 1 . MEDIO RESTADOR.

Un medio restador es un circuito combinacional que sustrae dos bits y produce su diferencia. También tiene una salida para especificar si se ha tomado un 1. Se designa el bit minuendo por x y el bit sustraendo mediante y. Para llevar a cabo x-y, tienen que verificarse las magnitudes relativas de x y y. Si $x \geq y$, se tienen tres posibilidades; 0-0 = 0, 1-0 = 1 y, 1-1 = 0. El resultado se denomina bit de diferencia. Si $x < y$, tenemos 0-1 y es necesario tomar un 1 de la siguiente etapa mas alta. El 1 que se toma de la siguiente etapa mas alta añade 2 al bit minuendo, de la misma forma que en el sistema decimal lo que se toma añade 10 a un dígito minuendo. Con el minuendo igual a 2, la diferencia llega a ser 2-1 = 1. El medio restador requiere dos salidas. Una salida genera la diferencia y se denotara por el símbolo D. La salida, denotada B para lo que se toma, genera la señal binaria que informa a la siguiente etapa que se ha tomado un 1. La tabla de verdad para las relaciones de entrada-salida de un medio restador ahora puede derivarse como sigue:

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

TABLA 4.2 TABLA DE VERDAD PARA UN MEDIO RESTADOR.

La salida que toma B es un 0 en tanto que $x \geq y$. Es un 1 para $x = 0$ y $y = 1$. La salida D es el resultado de la operación aritmética $2B + x - y$.

Las funciones booleanas para las dos salidas del medio restador se derivan de manera directa de la tabla de verdad:

$$D = x'y + xy'$$
$$B = x'y$$

Es interesante observar que la lógica para D es exactamente la misma que la lógica para la salida S en el medio sumador.

4.2.2. RESTADOR COMPLETO.

Un restador completo es un circuito combinacional que lleva a cabo una sustracción entre dos bits, tomando en cuenta que un 1 se ha tomado por una etapa significativa más baja. Este circuito tiene tres entradas y dos salidas. Las tres entradas, x, y y z, denotan al minuendo, sustraendo y a la toma previa, respectivamente. Las dos salidas, D y B, representan la diferencia y la salida tomada, respectivamente. La tabla 4.3 representa la tabla de verdad para el circuito.

Los ocho renglones bajo las variables de entrada designan todas las combinaciones posibles de 1 y 0 que pueden tomar las variables binarias. Los 1 y 0 para las variables de salida están determinados por la sustracción de x-y-z. Las combinaciones que tienen salida de toma $z = 0$ se reducen a las mismas cuatro condiciones del medio sumador. Para $x = 0$, $y = 0$ y $z = 1$, tiene que tomarse un 1 de la siguiente etapa, lo cual hace $B = 1$ y añade

2 a x. Ya que 2-0-1= 1, D= 1. Para x = 0 y yz= 11, necesita tomarse otra vez, haciendo B= 1 y x= 2. Ya que 2-1-1 = 0, D = 0. Para x= 1 y yz= 01, se tiene x-y-z= 0, lo cual hace B = 0 y D = 0. Por último, para x= 1, y= 1, z= 1, tiene que tomarse 1, haciendo B= 1 y x= 3 y, 3-1-1= 1, haciendo D= 1.

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

TABLA 4.3 TABLA DE VERDAD PARA UN RESTADOR COMPLETO.

La función booleana simplificada para las dos salidas del restador completo se derivan en los mapas de la figura 4.11. Las funciones simplificadas de salida en suma de productos son:

$$D = x'y'z + x'yz' + xy'z' + xyz$$

$$B = x'y + x'z + yz$$

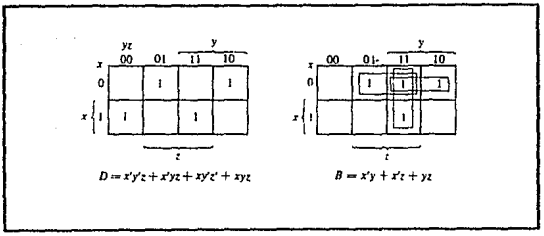


FIG. 4.11 MAPAS PARA UN RESTADOR COMPLETO.

De nuevo se observa que la función lógica para la salida D en el restador completo es exactamente la misma que para la salida S del sumador completo. Ademas, la salida B se asemeja a la función para C en el sumador completo, excepto que la variable de entrada x está complementada. Debido a estas similitudes, es posible convertir un sumador completo en un restador completo, complementando tan sólo la entrada x antes de su aplicación a las compuertas que forman la salida de acarreo.

4.2.3. SISTEMA DE COMPLEMENTO 2.

Las computadoras más modernas utilizan el sistema de complemento 2 para representar números negativos y para efectuar la operación de sustracción. Las operaciones de adición y sustracción (suma y resta) de números con signo se pueden efectuar usando solamente la operación de adición si se emplea la forma de complemento 2 para representar numeros negativos.

Adición. Los números positivos y negativos, incluyendo los bits de los signos, se pueden sumar en el circuito sumador paralelo básico cuando los numeros negativos están en forma de complemento 2. Esto se ilustra en la figura 4.12 para la adición de -3 y +6. El número -3 se representa en su forma de complemento 2 como 1101, donde el primer 1 es el bit del signo; el +6 se representa como 0110, con el primer 0 como bit del signo. Estos números se almacenan en sus registros correspondientes. El sumador paralelo de 4 bits produce resultados de la suma de 0011, que representa a +3. La salida C_4 es 1, pero se desprecia en el método de complemento 2.

Sustracción. Cuando se emplea el sistema de complemento 2, el número por restarse (el sustraendo) se complementa en 2 y luego se suma al minuendo (el número del cual se resta el sustraendo). Por ejemplo, podemos suponer que el minuendo ya está almacenado en el acumulador (registro A). El sustraendo se coloca entonces en el registro B (en una computadora se transferiría de aquí a la memoria) y se cambia a su forma de complemento 2 antes de que se sume al número en el registro A.

Las salidas del circuito sumador representan ahora la diferencia entre el minuendo y el sustraendo.

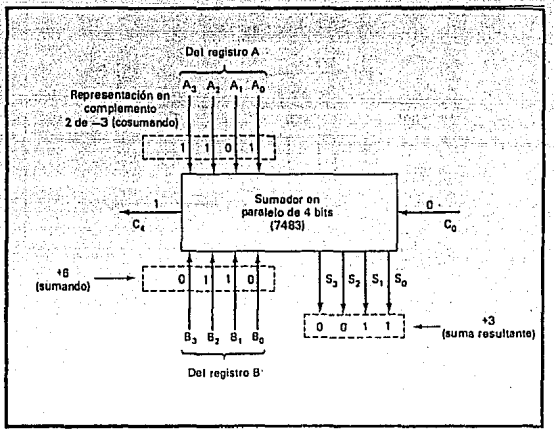


FIG. 4.12 SUMADOR EN PARALELO USADO PARA ADICIONAR DOS NUMEROS POSITIVOS Y NEGATIVOS EN EL SISTEMA DE COMPLEMENTO A 2.

El circuito sumador paralelo que se ha venido analizando se puede adaptar para efectuar la sustraccion que se describio antes si se da un medio para tomar el complemento 2 del numero del registro B. El complemento 2 de un numero binario se obtiene complementando (invirtiendo) cada bit y luego sumando 1 al LSB. La figura 4.13 muestra la forma en que esto se puede llevar a cabo. Las salidas invertidas del registro B se emplean en vez de las salidas normales; es decir, B_0', B_1', B_2' y B_3' se alimentan a las entradas del sumador (recuerdese que B_3 es el bit del signo). Este se ocupa de complementar cada bit del numero B.

Asimismo, C_0 se convierte en un 1 logico, de manera que suma otro 1 al LSB del sumador; esto ocasiona el mismo efecto que sumar 1 al LSB del registro B para formar el complemento 2.

Las salidas S_3- S_0 representan los resultados de la operación

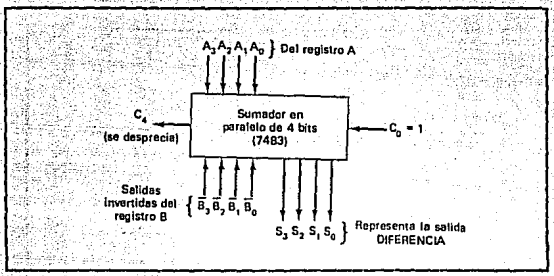


FIG. 4.19 SUMADOR EN PARALELO QUE SE EMPLEA PARA REALIZAR LA OPERACION DE SUSTRACCION CON EL SISTEMA DE COMPLEMENTO A 2.

de la sustracción. Por supuesto, S_3 es el bit del signo del resultado e indica si el resultado es positivo o negativo. La salida del acarreo C_4 se vuelve a despreciar.

Para ayudar a aclarar esta operación, estudiemos los siguientes pasos que se siguen para restar +6 de +4 :

1. +4 se almacena en el registro A como 0100.

2. +6 se almacena en el registro B como 0110.

3. Las salidas invertidas del registro B se alimentan al sumador (es decir, 1001).

4. El 1001 se suma a 0100 por medio de sumador paralelo junto con un 1 agragado a la posicion del LSB haciendo $C_0 = 1$. Esto produce los bits de salida de la suma 1110 y un $C_4 = 1$, que se desprecia. Este 1110 representa la diferencia que se pide. Ya que el bit del signo= 1, este es un resultado negativo y esta en forma de complemento 2. Podemos verificar que 1110 represente a -2_{10} complementandolo en 2 y obteniendo $+2_{10}$.

```
  1 1 1 0
  0 0 0 1
+       1
---------
  0 0 1 0 = +2₁₀
```

4.3. MULTIPLICADORES.

La multiplicación de dos números binarios se realiza con lápiz y papel efectuando adiciones sucesivas y corrimientos o alternaciones. Para ilustrar lo anterior, tenemos :

```
      1 0 1 1        multiplicando
      x 1 0 1        multiplicador
    -----------
      1 0 1 1        productos
    0 0 0 0          parciales
  1 0 1 1
-------------
  1 1 0 1 1 1        producto
```

Este proceso consiste en examinar los bits sucesivos del multiplicador, empezando con el LSB. Si el bit multiplicador es un 1, el multiplicando se transcribe abajo; si se trata de un 0, se escriben ceros abajo. Los números puestos en líneas sucesivas se corren una posición a la izquierda en relación con la línea anterior. Cuando todos los bits multiplicadores se han examinado, las diversas líneas se suman para producir el producto final.

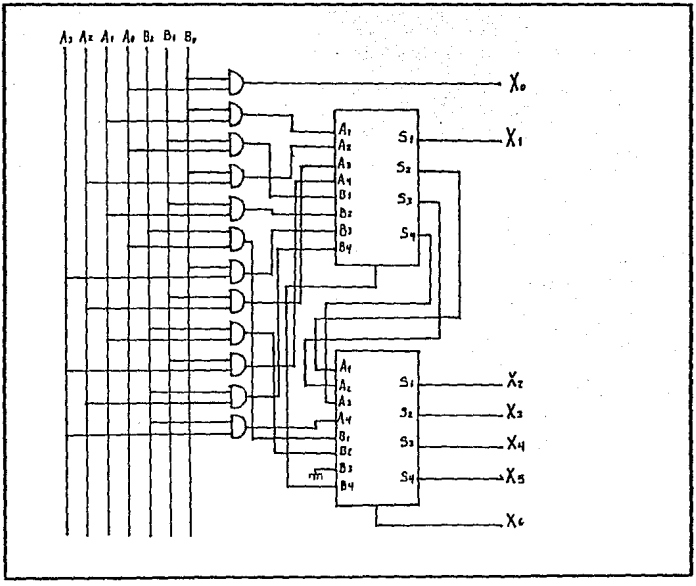
En las máquinas digitales este proceso se modifica, se utilizan sumadores binarios (CI 7483) para realizar esta operación.

El circuito multiplicador de dos números binarios se muestra en la figura anterior. Para explicar mejor el circuito considerense dos numeros binarios A y B designando a sus bits como $A_3A_2A_1A_0$ y $B_2B_1B_0$, respectivamente. Sustituyendo estos numeros en el proceso de multiplicacion se tiene :

```
                                  A3  A2  A1  Ao
                                  X   B2  B1  Bo
-------------------------------------------------
                      A3Bo A2Bo A1Bo AoBo
                  A3B1 A2B1 A1B1 AoB1
              A3B2 A2B2 A1B2 AoB2
-------------------------------------------------
Xo   X5   X4   X3   X2   X1   Xo
```

Donde :

Xo = BoAo
X1 = BoA1 + B1Ao
X2 = BoA2 + B1A1 + B2Ao
X3 = BoA3 + B1A2 + B2A1
X4 = B1A3 + B2A2
X5 = B2A3
Xo = POSIBLE ACARREO



MULTIPLICADOR BINARIO EMPLEANDO SUMADORES IC 7483.

Para gnerar los bits del producto total se sumarán los bits correspondientes de los productos parciales, puede ser posible que dicha suma genere un bit de acarreo que se tenga que sumar para generar el siguiente bit del producto total. Este problema se resuelve realizando la suma de los bits do los productos parciales en un sumador IC 7483, en este circuito el posible bit de acarreo se suma automaticamente al siguiente bit.

Como se puede observar en el circuoto, el primer bit del producto total (X_0), esta compuesto solo por el primer bit del primer producto parcial, por lo que no es necesario meterlo al sumador.

Los siguientes bits si son generados por la suma de dos o tres bits correspondientes a los productos parciales, por este motivo si se deben sumar dentro del circuiro IC 7483.

Para generar el bit X_1 (S_1 del primer sumador) se tiene que sumar B_0A_1 y B_1A_0, donde B_0A_1 se pone en la posición A_1 del arreglo A del sumador numero 1 y B_1A_0 en la posicion B_1 del arreglo B del mismo sumador, esto genera un posible acarreo el cual se sumara a B_0A_2 + B_1A_1 + B_2A_0 para generar el bit X_2. Para que este bit sea generado, B_0A_2 se posiciona en A_2 y B_1A_1 en la posicion B_2 del sumador numero 1 (generando un posible bit de acarreo para generar el siguiente bit del producto total), a la salida tenemo a S_2 el cual se suma con B_2A_0 en A_1 y B_1 del sumador numero 2 respectivamente generando otro posible acarreo y el bit X_2 del producto total, por esto, B_0A_3 y B_1A_2 se sumaran en A_3 y B_3 del primer sumador para tomar el posible acarreo generado por S_2 y el resultado S_3 se sumará con B_2A_1 en las posiciones A_2 y B_2 del sumador numero 2 respectivamente para sumarlo con el otro posible acarreo generado por X_2 y así generar X_3 (que sera la lalida S_2 del segundo sumador), este procedimiento continua hasta generar todos los bits del producto total.

4.4. DECODIFICADORES.

Un decodificador es un circuito lógico que convierte un código binario de entrada de N bits en M líneas de salida tal que cada una de estas líneas de salida sea activada solo para una posible combinación de entradas. La figura 4.14 muestra el diagrama general del decodificador con N entradas y M salidas. Ya que cada una de

las N entradas puede ser 0 o bien 1, hay 2^N posibles combinaciones o códigos de entrada. Para cada una de estas combinaciones de entrada sólo una de las M salidas será ALTA activa; todas las otras son BAJAS. Muchos decodificadores están diseñados para producir salidas BAJAS activas, donde solamente la salida seleccionada es BAJA, en tanto que todas las otras son ALTAS. Esto siempre lo indica la presencia de pequeños círculos en las líneas de salida del diagrama del decodificador.

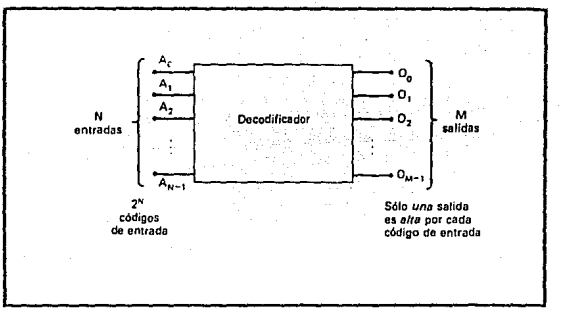
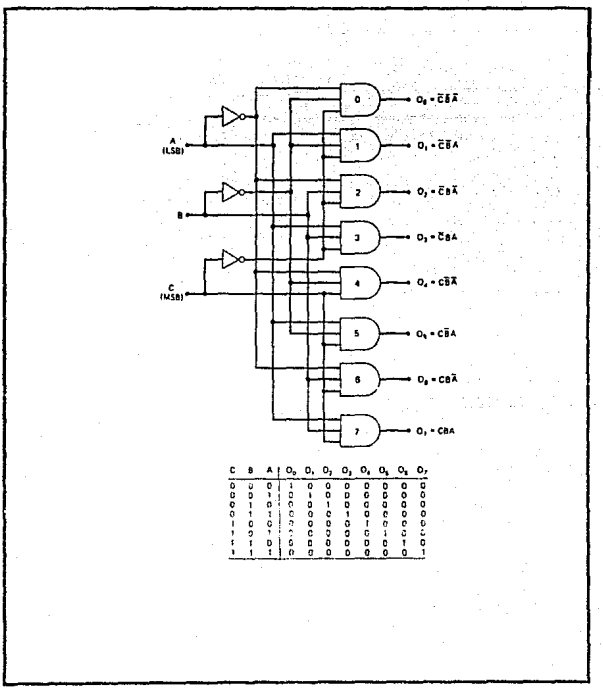


FIG. 4.14 DIAGRAMA GENERAL DEL DECODIFICADOR.

Algunos decodificadores no utilizan los 2^N posibles códigos de entrada, sino sólo unos cuantos. Por ejemplo, un decodificador de BCD a decimal tiene un código de entrada de 4 bits y diez líneas de salida que corresponden a diez grupos de código BCD, del 0000 al 1001. Los decodificadores de este tipo a menudo se diseñan de tal modo que si alguno de los códigos no utilizados se aplican a la entrada, ninguna de las salidas sea activada.

La figura 4.15 muestra los circuitos para un decodificador con tres entradas y $2^N = 8$ salidas. Este hace uso de todas las compuertas AND, de modo que las salidas son ALTAS activas. Para salidas BAJAS activas se utilizarán compuertas NAND. Nótese que para un código de entrada dado, la única salida que es activa (HIGH) es la que corresponde al decimal equivalente del código de entrada binario (por ejemplo, la salida O_6 pasa a ALTO cuando $CBA = 110_2 = 6_{10}$).

Este decodificador se puede denominar de varias maneras. Se le puede llanar decodificador de 3 a 8 líneas, ya que tiene tres líneas de entrada y ocho líneas de salida. También se le podría denominar decodificador de binario a octal o convertidor de binario a octal debido a que toma un código binario de entrada de 3 bits y activa una de las ocho salidas (octal) correspondiente a



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

FIG. 4.15 DECODIFICADOR DE 3 A 8 LINEAS (O BIEN 1 DE 8).

ese código. También se le conoce como decodificador 1 de 8, ya que solo 1 de las 8 se activa a la vez.

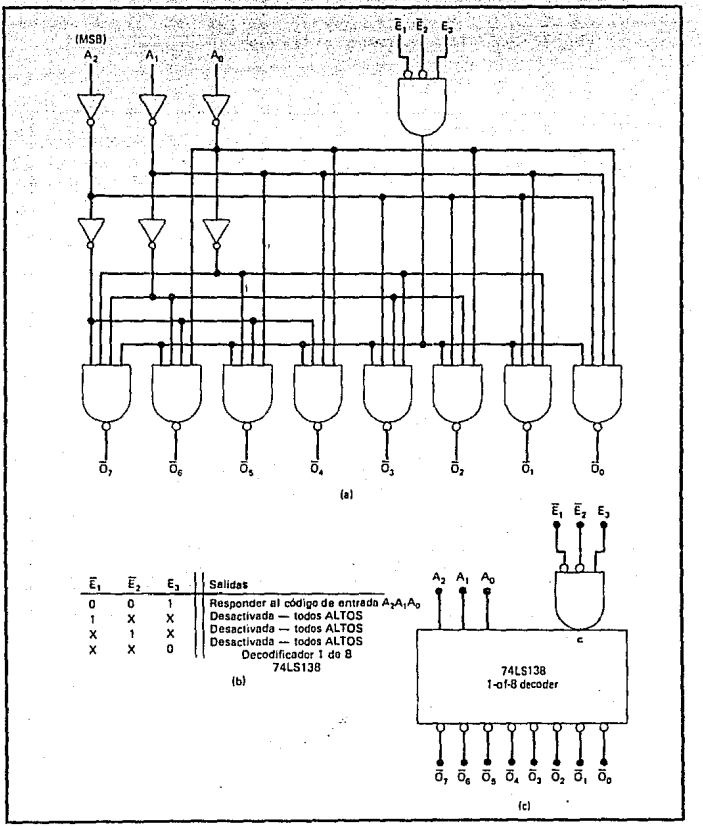
Entradas HABILITABLES (ENABLE). Algunos decodificadores tienen una o mas entradas HABILITABLES que se utilizan para controlar la operación del decodificador. Por ejemplo, diríjase al decodificador de la figura 4.15 y obsérvese que tiene una línea común HABILITABLE conectada a la cuarta entrada de cada compuerta. Con esta línea HABILITABLE mantenida en ALTO, el decodificador funcionará normalmente y el código de entrada A, B, C determinará que salida es ALTA. Con la HABILITACION sostenida en BAJO, no obstante, todas las salidas serán forzadas a estar en el estado BAJO independientemente de los niveles de entradas A, B, C. Así el decodificador es activado solamente si la HABILITACIÓN es ALTA.

La figura 4.16(a) muestra el diagrama lógico del decodificador 74LS138 tal y como aparece en el Manual de TTL de Fairchild. Si se examina este diagrama cuidadosamente podemos determinar exactamente la forma en que este decodificador funciona. Primero, obsérvese que tiene salidas de compuerta NAND, de modo que sus salidas son BAJAS activas. Otra indicación es la rotulación de las salidas como $\overline{O_7}$, $\overline{O_6}$, $\overline{O_5}$, etc.; la barra superpuesta de inversión indica que se trata de salidas BAJAS activas.

El código de entrada se aplica en A_2, A_1 y A_0, donde A_2 es el MSB. Con tres entradas y ocho salidas, éste es un decodificador de 3 a 8 o bien, equivalentemente un decodificador 1 de 8.

Las entradas E_1', E_2' y E_3 son entradas activadas separadas que se combinan en la compuerta AND. A fin de activar las compuertas NAND de salida para responder al código de entrada en $A_2A_1A_0$, esta salida de la compuerta AND tiene que ser ALTA. Esto ocurrirá solo cuando $E_1' = E_2' = 0$ y $E_3 = 1$. En otras palabras, E_1' y E_2' son BAJAS activas, E_3 es ALTA activa y los otros tres tienen que estar en sus estados activos para activar las salidas del decodificador. Si una o más de las entradas activadas se encuentran en su estado inactivo, la salida de AND será BAJA, lo cual forzará a todas las salidas de NAND a estar en su estado ALTO inactivo independiente del código de entrada. Esta operación se resume en la tabla de verdad de la figura 4.16(b); "x" representa la condición "no se preocupe".

El símbolo lógico del 74LS138 se muestra en la figura 4.16(c).



$\bar{E}_1$	$\bar{E}_2$	E_3	Salidas
0	0	1	Responder al código de entrada $A_2A_1A_0$
1	X	X	Desactivada — todos ALTOS
X	1	X	Desactivada — todos ALTOS
X	X	0	Desactivada — todos ALTOS

Decodificador 1 de 8
74LS138

(b)

FIG. 4.16 (a) DIAGRAMA LOGICO DEL DECODIFICADOR 74LS138; (b) TABLA DE VERDAD; (c) SIMBOLO LOGICO.

Nótese la forma en que las salidas BAJAS activas son representadas y como se representan las entradas activadas.

Ejemplo : *La figura 4.17 muestra la forma en que cuatro unidades 74LS138 y un INVERSOR se puede disponer para funcionar como un decodificador 1 de 32. Los decodificadores se rotulan Z1-Z4 para su facil referencia y las ocho salidas de cada uno se combinan en 32 salidas. Las salidas de Z1 son O_0'-O_7', las salidas de Z2 O_8'-O_{15}', respectivamente, las salidas de Z3 se denominan O_{16}'-O_{23}' y las de Z4 se denominan O_{24}'-O_{31}'. Un codigo de entrada de 5 bits A_4 A_3 A_2 A_1 A_0 activara solamente una de estas 32 salidas para cada uno de los 32 codigos de entrada. (a) Que salida sera activada para A_4 A_3 A_2 A_1 A_0= 01101? (b) Que intervalo de codigos de entrada activara el integrado Z4?*

SOLUCION : *(a) El código de entrada de 5 bits tiene dos porciones distintas. Los bits A_4 y A_3 determinan cual de los integrados del decodificador Z1-Z4 sera activada, en tanto que A_2 A_1 A_0 determina que salida del integrado activado se accionara. Con A_4 A_3= 01, solo Z2 tiene todas sus entradas de activacion accionadas. Así, Z2 responde al codigo A_2 A_1 A_0= 101 y activa su salida O_5', la cual se ha denominado O_{13}'. De este modo, el codigo de entrada 01101, que es el equivalente binario del 13 decimal, ocasionara que la salida O_{13}' pase a BAJO (LOW) en tanto que las otras permanecen en ALTO (HIGH). (b) Para activar Z4, tanto A_4 como A_3 tienen que ser ALTAS. Así, todos los codigos de entrada que van de $11000_{(24_{10})}$ a $11111_{(31_{10})}$ activaran Z4. Esto corresponde a las salidas O'_{24} a O'.*

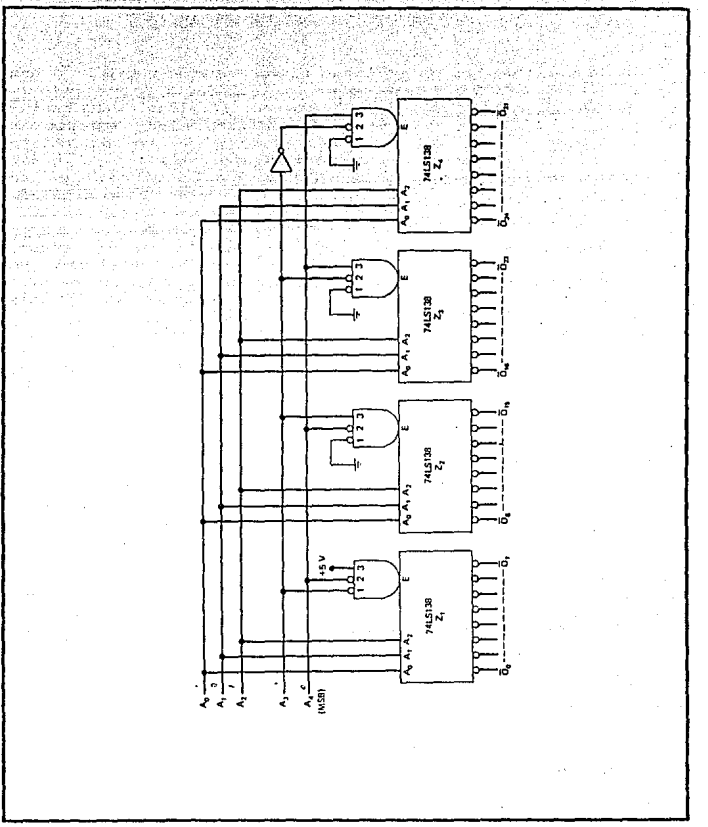


FIG. 4.17 CUATRO UNIDADES 74LS138 QUE FORMAN UN DECODIFICADOR 1 DE 32.

4.4.1. APLICACIONES DE LOS DECODIFICADORES.

Los decodificadores se utilizan siempre que una salida o grupo de salidas se activan solamente en la incidencia de un codigo de entrada unico. Cuando el codigo de entrada proviene de un contador que se cronometra a cierta intensidad, las salidas del decodificador se activan secuencialmente y se pueden utilizar señales de distribucion o secuenciacion para controlar la sucesion de eventos como el encendido de motores o calefactores, asi como su desactivacion, en un instrumento controlado en forma digital.

Los decodificadores son extremadamente valisos en el sistema de memoria de una computadora. Ahi, estos se utilizan para seleccionar un banco de integrados de memoria de entre muchos en respuesta a una entrada de codigos con direccion desde la unidad de control de la computadora.

Otra valiosa aplicacion de los decodificadores es la exhibicion de datos en formatos de lecturas decimales.

4.4.2. DECODIFICADORES DE BCD A DECIMAL.

La figura 4.18(a) muestra el diagrama logico del decodificador de BCD a decimal 7442. Cada salida pasa a BAJO cuando su entrada BCD correspondiente es aplicada. Por ejemplo, O_5' pasara a BAJA solo cuando las entradas DCBA= 0101 y O_8' pasara a BAJO solamente cuando DCBA= 1000. Para los codigos de entrada que no son BCD, ninguna de las salidas se activara. Este decodificador se denomina asimismo decodificador de 4 a 10 o bien decodificador 1 de 10. El simbolo logico del 7442 se muestra en la figura 4.18(b).

4.4.3. DECODIFICADOR / IMPULSOR DE BCD A DECIMAL.

Un decodificador/ impulsor de BCD a decimal tiene un transistor con colector abierto en cada salida de manera que pueda activar lamparas, relevadores, etc. Una de sus aplicaciones mas importantes es en la activacion de tubos de exhibicion de catodo frio (tubos nixie), los cuales exhiben los numerales decimales correspondientes al codigo de entrada BCD. La figura 4.19 muestra

una disposición común donde un decodificador/impulsor de BCD a decimal 74141 activa una lectura en tubo nixie. Las entradas DCBA generalmente provienen de un contador o de un registro de almacenamiento. Para cierto código BCD de entrada, una de las salidas Oo'-Oo' será BAJA (transistor conductor saturado). Este estado BAJO esencialmente pone a tierra el filamento adecuado del tubo nixie, de manera que la corriente fluya de la fuente de 170 V a través del tubo y hasta este filamento, ocacionando que resplandezca en la forma del dígito decimal correspondiente al código de entrada. En esta forma el tubo exhibe el dígito decimal que se presenta en las entradas del decodificador en forma BCD. Para cualquier código de entrada mayor de 1001, todas las salidas del decodificador serán ALTAS (transistor de impulsor abierto) y ninguno de los filamentos del tubo se quemará.

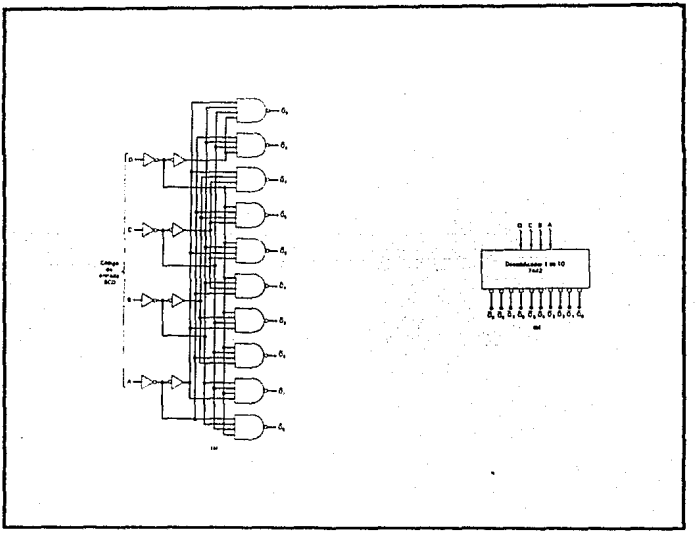


FIG. 4.18 (a) DIAGRAMA LOGICO DEL DECODIFICADOR DE BCD A DECIMAL; (b) SIMBOLO LOGICO.

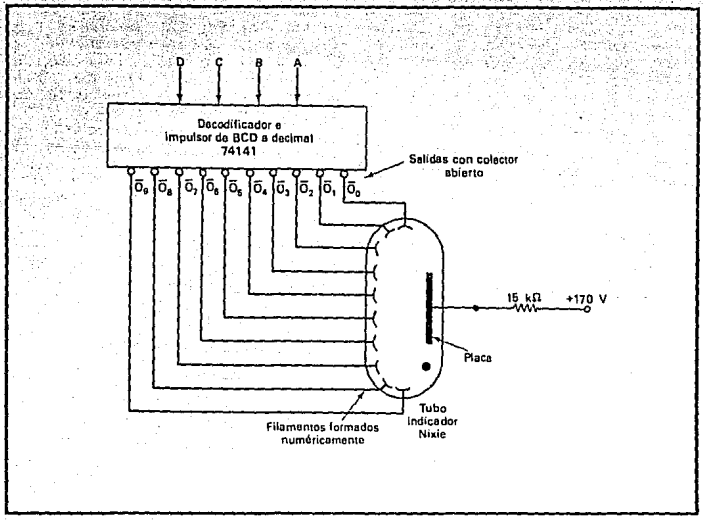


FIG. 4.19 APLICACION COMUN DE UN DECODIFICADOR E IMPULSOR DE BCD A DECIMAL CON EXHIBICION EN TUBO NIXIE.

4.4.4. DECODIFICADORES Y CONDUCTORES DE BCD A 7 SEGMENTOS.

Muchas exhibiciones numéricas utilizan una configuración de 7 segmentos (figura 4.20(a)) para producir los caracteres decimales 0-9 y algunas veces los caracteres hexadecimales A-F. Cada segmento está construido de un material que emite luz cuando se pasa corriente a través de él. Los materiales que se utilizan mas comunmente incluyen diodos emisores de luz (LED) y filamentos incandescentes. La figura 4.20(b) muestra los modelos de segmentos que sirven para exhibir los diversos dígitos. Por ejemplo, para

exhibir "6" los segmentos c, d, e, f y g son brillantes, en tanto que los segmentos a y b son oscuros.

Se utiliza un decodificador y condictor de BCD a 7 segmentos para tomar una entrada BCD de 4 bits y dar las salidas que pasaran corriente a traves de los segmentos indicados para exhibir el dígito decimal. La lógica de este decodificador es mas complicada que las que se analizaron anteriormente, debido a que cada salida es activada para mas de una combinacion de entradas. Por ejemplo, el segmento e debe ser activado para cualquiera de los dígitos 0, 2, 6 y 8, lo cual significa cuando quiera que cualquiera de los codigos 0000, 0110 o bien 1000 ocurre.

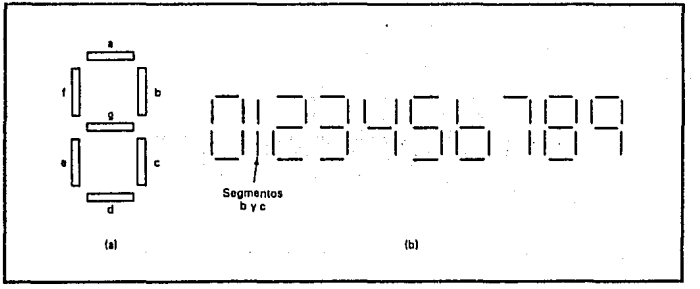


FIG. 4.20 (a) DISPOSICION DE 7 SEGMENTOS; (b) SEGMENTOS ACTIVOS PARA CADA DIGITO.

La figura 4.21(a) muestra un decodificador e impulsor de BCD a 7 segmentos (TTL 7446 o 7447) que se utiliza para impulsar una lectura LED de 7 segmentos. Cada segmento consta de uno o dos LED. Los ánodos de los LED están todos unidos a Vcc (+5 V). Los cátodos de los LED están conectados a través de resistencias limitadoras de corriente a las salidas adecuadas del decodificador e impulsor. El decodificador e impulsor tiene salidas BAJAS activas que son transistores conductores de colector abierto que pueden devolver una corriente bastante grande. Esto se debe a que las lecturas LED pueden requerir 10 mA a 40 mA por segmento, segun su tipo y tamaño.

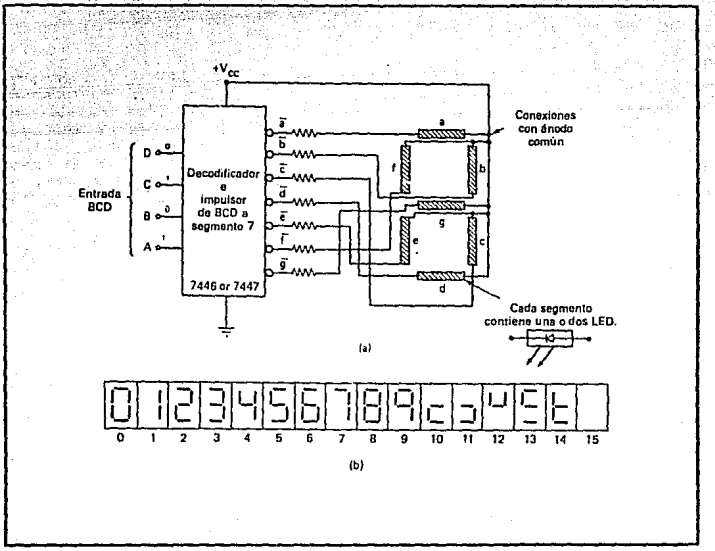


FIG. 4.21 (a) DECODIFICADOR E IMPULSOR DE BCD A SEGMENTO 7 QUE DA IMPULSO A UN EXHIBICION LED DE 7 SEGMENTOS CON ANODO COMUN; (b) MODELOS DE SEGMENTOS PARA TODOS LOS POSIBLES CODIGOS DE ENTRADA.

Ejemplo : *Cada segmento de una exhibición de 7 segmentos común está clasificado para operar en 20 mA a 2.8 V de brillantes normal. Calcule el valor de la resistencia de límite de corriente que necesita para producir aproximadamente 20 mA por segmento.*

Solucion : *Refiriéndonos a la figura 4.21, podemos apreciar que la resistencia en serie debera tener una reduccion de voltaje igual a la diferencia entre Vcc= 5 V y el voltaje del segmento de 2.8 V. Este voltaje de 2.2 V que atraviesa la resistencia debe producir una corriente de 20 mA. Por lo tanto se tiene*

$$Rs = \frac{2.2\ V}{20\ mA} = 110\ \Omega$$

Una resistencia de valor estándar en la proximidad de éste se puede utilizar. Una resistencia de 100 Ω sería una elección adecuada.

4 . 5 . CODIFICADORES.

Un decodificador acepta un código de entrada de N bits y produce un estado ALTO (o BAJO) en una y sólo una línea de salida. En otras palabras, podemos decir que un decodificador identifica, reconoce o bien detecta un código específico. El recíproco de este proceso de decodificación se denomina codificación y es realizado por un circuito lógico que se conoce como codificador. Un codificador tiene varias líneas de entrada, solo una de las cuales se activa en un momento dado, y produce un código de salida de N bits, según la entrada que se active. La fig. 4.22 es el diagrama general de un codificador con M entradas y N salidas. Aquí, las entradas son ALTAS activas, lo cual significa que normalmente son BAJAS.

Se observó que un decodificador de binario a octal acepta un código de entrada binario de 3 bits y activa una de las ocho líneas de salida. Un codificador de octal a binario opera en la forma contraria. Acepta ocho líneas de entrada y produce un código de salida binario de 3 bits. Sus circuitos se muestran en la figura 4.23. Se supone que solamente una de las líneas de entrada se hace ALTA en un momento dado; de manera que sólo hay ocho posibles condiciones de entrada. El circuito está diseñado de manera que cuando A0 sea ALTA, el código binario 000 sea generado

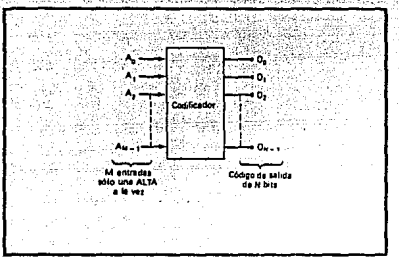


FIG. 4.22 DIAGRAMA GENERAL DE UN CODIFICADOR.

en la salida; cuando A_1 es ALTA, se genera el código binario 001; cuando A_2 es ALTA, se genera el código 010; y así sucesivamente (vease la tabla de verdad correspondiente). El diseño del circuito es muy simple, ya que sólo comprende la observación de cada bit de la salida y la determinación de en qué casos de entrada ese bit es ALTA y luego la operación con OR de los resultados. Por ejemplo, la tabla de verdad muestra que O_0 (LSB del código de salida) debe ser 1 siempre que las entradas A_1, A_3, A_5 o bien A_7 sean ALTAS. Por tanto, se tiene :

$$O_0 = A_1 + A_3 + A_5 + A_7$$

Las otras salidas se diseñan de acuerdo con este principio. Este diseño es simplificado porque sólo ocho del total de 2^8 posibles condiciones de entrada se utilizan. Si más de una entrada se hace ALTA en un instante dado, los resultados de salida serán erróneos. Si ninguna de las entradas es ALTA, las salidas se leerán 000. El codificador de la figura 4.23 se llama asimismo codificador de 8 a 3 líneas.

4.5.1. CODIFICADORES DE PRIORIDAD.

Como se dijo antes, el codificador de la figura 4.23 producirá resultados erróneos si se activan simultáneamente dos o más entradas. Por ejemplo, si seguimos a través de las compuertas lógicas de este circuito en el caso donde A_5 y A_3 son ambas

ALTAS, observamos que el codigo de salida sera 111, que es el correspondiente a la entrada A7.

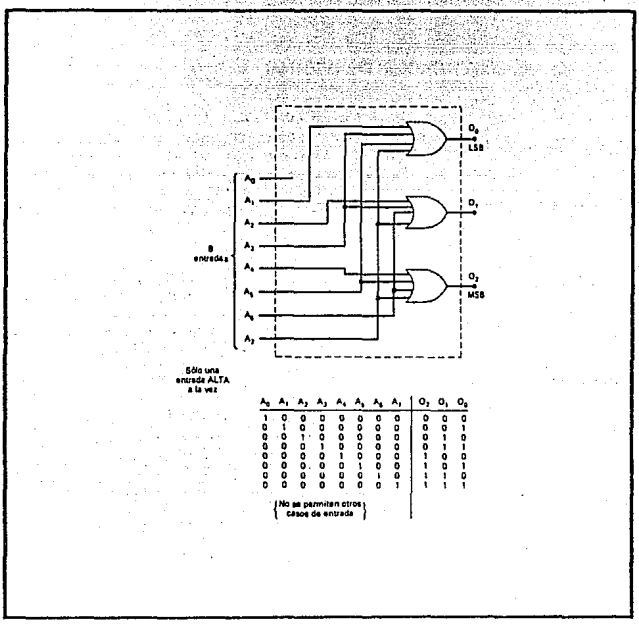


FIG. 4.29 CODIFICADOR DE OCTAL A BINARIO (DE 3 A 8 LINEAS).

Un codificador de prioridad es una versión modificada del circuito codificador basico que contiene los circuitos que se necesitan para asegurar que cuando activen dos o mas entradas, el código de salida corresponderá a la entrada con el número mayor. Por ejemplo, el 74148, que es un codificador de prioridad de 8 a 3

líneas, producirá el código de salida 101 cuando se activen A_5 y A_3. De igual manera, producirá el código de salida 110 para A_6 si las entradas A_0, A_2 y A_6 son activadas.

4.5.2. CODIFICADOR DE DECIMAL A BCD.

La figura 4.24 muestra el símbolo lógico del codificador de prioridad de decimal a BCD 74147. Tiene 9 entradas BAJAS activas que representan los dígitos decimales del 1 al 9 y produce el código BCD invertido que corresponde a la entrada A_4' es BAJA en tanto que todas las otras son ALTAS el código de salida será O_3 O'_2 O_1 O_0= 1011; este es el recíproco de 0100, que es el código BCD del número 4 decimal.

Las salidas del 7441 normalmente serán ALTAS, cuando no se active ninguna de las entradas. Esto corresponde a la condición de la entrada 0 decimal. En realidad no existe tal entrada A_0', ya que el codificador supone el estado de entrada 0 decimal cuando todas las otras entradas son ALTAS. Las salidas BCD invertidas del 74147 pueden convertirse en un código BCD normal al poner a cada una a través de un INVERSOR.

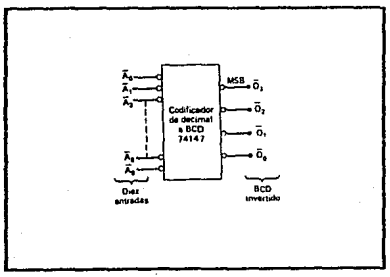


FIG. 4.24 CODIFICADOR DE PRIORIDAD DE DECIMAL A BCD 74147.

4.6. MULTIPLEXORES.

Un multiplexor de información o selector de datos es un circuito lógico que acepta varias entradas de datos y admite sólo una de ellas al momento de dirigirse hacia la salida. La dirección de la entrada de datos deseada hacia la salida es controlada por entradas "SELECT" (que algunas veces se conocen como entradas ADDRESS). La figura 4.25 muestra el símbolo de un transmisor masivo general (MUX). En este diagrama las entradas y salidas se trazan como flechas grandes para indicar que pueden ser una o mas líneas.

El transmisor masivo de información actúa como un interruptor de posiciones múltiples controlado digitalmente, donde el código digital que se aplica a las entradas SELECT controla que entradas de datos serán activadas en la salida. Por ejemplo, la salida Z será igual a la entrada de datos Zo de algún código de entrada SELECT determinado; Z será igual a I, para otro código de entrada SELECT específico; y así sucesivamente. Dicho de otra manera, un transmisor masivo de información selecciona 1 de N fuentes de datos seleccinados a un sólo canal de salida. A esto se le llama multiplexor de información.

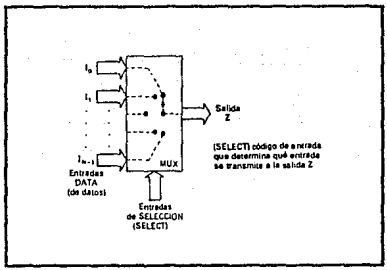


FIG. 4.25 SIMBOLO DE UN MULTIPLEXOR DIGITAL.

4 . 6 . 1 . MULTIPLEXOR DE DOS ENTRADAS.

La figura 4.26 muestra los circuitos lógicos de un multiplexor de información de dos entradas de datos Io e I1 y entrada SELECT S. El nivel lógico que se aplica a la entrada S determina que compuerta AND se activa de manera que su entrada de datos atraviese la compuerta OR hacia la salida Z. Observando esto desde otro punto de vista, la expresión booleana de la salida es:

$$Z = IoS' + I_1S$$

Con S= 0, esta expresión se convierte en

$$Z = IoI + I_10$$

$$= Io$$

lo cual indica que Z será idéntica a la señal de entrada Io, que puede ser un nivel lógico fijo o bien, una señal lógica que varia con el tiempo. Con S= 1, la expresión se transforma en

$$Z = Io0 + I_11 = I_1$$

lo cual muestra que la salida Z será idéntica a la señal de entrada I1.

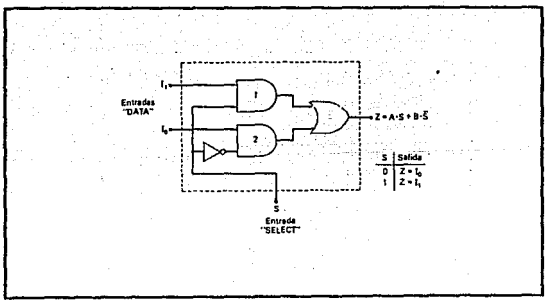


FIG. 4.26 MULTIPLEXOR DE DOS ENTRADAS.

Ejemplo : *Muestre cómo los multiplexores del tipo que se presenta en la figura 4.26 pueden usarse para tomar dos numeros binarios de 3 bits (X2, X1, X0 y Y2, Y1, Y0) y transmitir uno o el otro numero a las salidas Z2, Z1 y Z0, segun un nivel de seleccion de entrada.*

Solucion : *La figura 4.27 muestra 3 multiplexores de dos entradas que se usan para efectuar la operacion deseada. Notese que las entradas S de cada multiplexor estan interconectadas como una entrada de seleccion comun. Cuando S= 1, las entradas X de cada multiplexor individual se dirigen a las salidas Z. Cuando S= 0, las entradas Y se dirigen a traves de las salidas.*

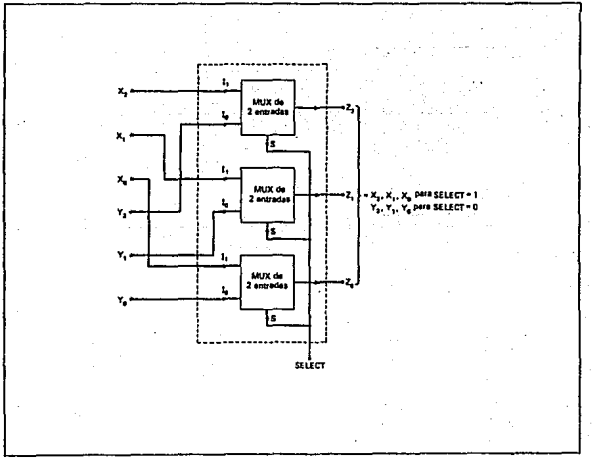


FIG 4.27 APLICACION DE LOS MULTIPLEXORES.

4.6.2. MULTIPLEXOR DE CUATRO ENTRADAS.

Se puede aplicar la misma teoría básica para formar el multiplexor de cuatro entradas que se muestra en la figura 4.28. Aquí se tienen cuatro entradas, que se transmiten en forma selectiva a la salida con base en las cuatro combinaciones posibles de las entradas de selección S_1 S_0. Cada entrada de datos se accesa con una diferente combinación de niveles de entrada de selección. Io se accesa con S_1' S_0' de manera que Io pase a través de su compuerta AND hacia la salida Z sólo cuando $S_1 = 0$. La tabla de la figura da las salidas de los otros tres códigos de selección de entrada.

En las familias lógicas TTL y CMOS se dispone regularmente de multiplexor de información de dos, cuatro, ocho y dieciséis entradas. Estos circuitos integrados (IC) básicos pueden ser combinados para multiplexor un gran número de entradas.

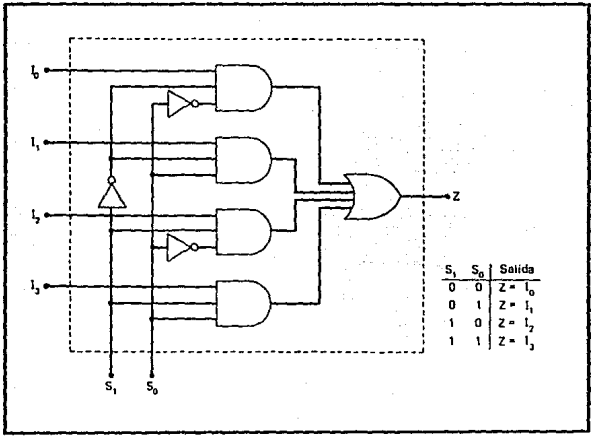


FIG. 4.28 MULTIPLEXOR DE CUATRO ENTRADAS.

4.6.3. MULTIPLEXOR DE OCHO ENTRADAS.

La figura 4.29(a) muestra el diagrama lógico del multiplexor de ocho entradas 74151. Este multiplexor tiene una entrada activadora, E'= 0, las entradas de selección S_2 S_1 S_0 seleccionaran una entrada de datos (I_0-I_7) para pasar hacia la salida Z. Cuando E'= 1, el multiplexor es desactivado de manera que Z= 0 independientemente del código de entrada de selección. Esta operación se resume en la figura 4.29(b), y el símbolo lógico 74151 se puede apreciar en la figura 4.29(c).

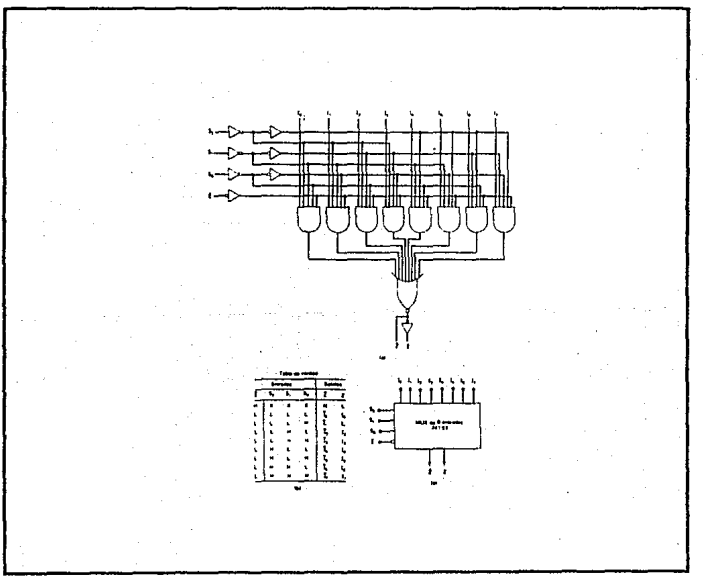


FIG. 4.29 (a) DIAGRAMA LOGICO DEL MULTIPLEXOR 74151; (b) TABLA DE VERDAD; (c) SIMBOLO LOGICO.

Ejemplo : *El circuito de la figura 4.30 emplea dos unidades 74151, un INVERSOR y una compuerta OR. Describa la operación de este circuito.*

Solución : *Este circuito tiene un total de 16 entradas de datos, ocho aplicadas a cada multiplexor se combinan en la compuerta OR para producir una sola salida X. El circuito funciona como un multiplexor de información de 16 entradas. Las cuatro entradas de selección* S_3 S_2 S_1 S_0 *definirán una de las 16 entradas para dirigirse hacia X.*

La entrada S_3 *determina qué multiplexor se activa. Cuando* $S_3 = 0$, *se activa el de la parte superior, y las entradas* S_2 S_1 S_0 *determinan cuáles de sus entradas de datos figurarán en su salida y atravesarán la compuerta OR para llegar a X. Cuando* $S_3 = 1$, *el multiplexor de la parte inferior es activado y las entradas* S_2 S_1 S_0 *seleccionan una de sus entradas de dotos para pasar hacia la salida X.*

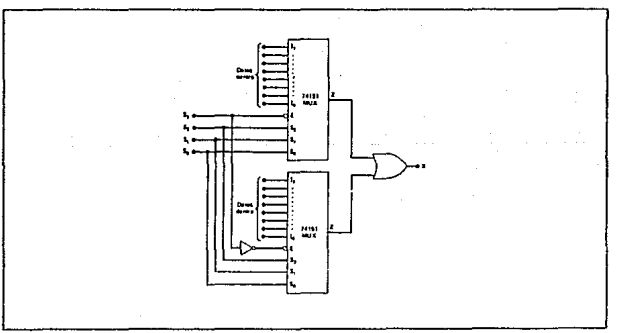


FIG. 4.30 APLICACION DE LOS MULTIPLEXORES.

4.6.4. MULTIPLEXOR CUADRUPLE DE DOS ENTRADAS.

Este es un IC de multiplexor muy útil que contiene cuatro multiplexores de dos entradas como el de la figura 4.26. El diagrama lógico del 74157 se muestra en la figura 4.31(a). Nótese la forma en que se rotulan las entradas y salidas de datos.

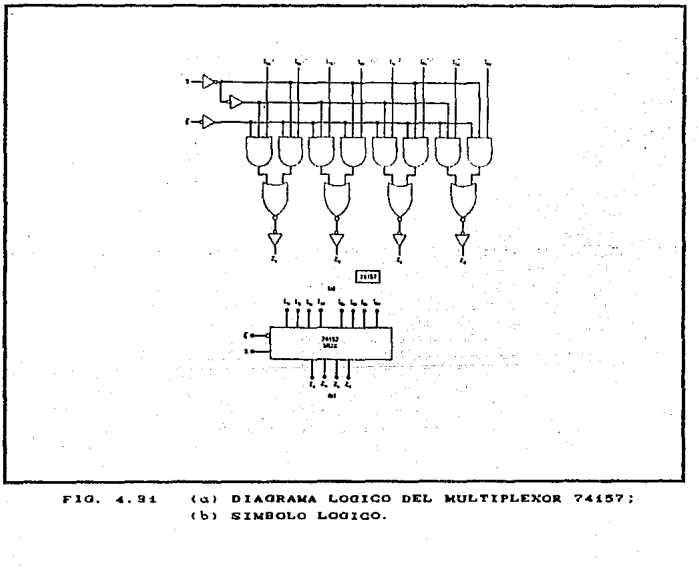


FIG. 4.31 (a) DIAGRAMA LOGICO DEL MULTIPLEXOR 74157; (b) SIMBOLO LOGICO.

4.6.5. APLICACIONES DEL MULTIPLEXOR DE INFORMACION.

Los circuitos del multiplexor hallan numerosas y variadas aplicaciones en sistemas digitales de todos los tipos. Estas aplicaciones incluyen seleccion de datos, destino, sucesion de

operaciones, conversión de paralelo en serie, generación de ondas y generación de funciones logicas.

Destino de los datos. Los multiplexores pueden dirigir los datos desde una de varias fuentes a un destino. Una aplicación común emplea transmisores masivos 74157 para seleccionar y exhibir el contenido de cualquiera de los dos contadores BCD utilizando un solo conjunto de decodificadores y conductores y exhibiciones LED. La disposición del circuito se muestra en la figura 4.32.

Cada contador consta de dos fases BCD en cascada y cada una es impulsada por su señal de cronometro. Cuando la linea CONTADOR SELECTOR (COUNTER SELECT) es ALTA, las salidas del contador 1 podran pasar a través de los multiplexores hacia el decodificador e impulsor para exhibirse en las lecturas LED. Cuando el CONTADOR SELECTOR = 0, las salidas del contador 2 pasaran a través de los multiplexores hacia las exhibiciones. En esta forma el contenido decimal de un contador o del otro será exhibido bajo el control de la entrada "COUNTER SELECT". Una situación comun donde podría usarse esto es en un reloj digital. Los circuitos del reloj digital. Los circuitos del reloj digital contienen muchos contadores y registros que se ocupan de los segundos, minutos, horas, días, meses, programación de la alarma, etcetera. Un esquema de selección como este permite se exhiban diferentes datos en el número limitados de lecturas decimales.

El objetivo de la técnica de seleccion de datos (multiplexación), tal y como se usa aquí, consiste en compartir el tiempo de los codificadores e impulsores y exhibir circuitos entre los dos contadores en vez de tener un conjunto aparte de decodificadores e impulsores y exhibiciones de cada contador. Esto da lugar a un ahorro significativo en el numero de conexiones alambradas, especialmente cuando se añaden más fases BCD a cada contador. Aun de mayor importancia es que representa una reducción considerable en consumo de energia, ya que los decodificadores e impulsores y lecturas LED por lo general atraen cantidades relativamente grandes de corriente de la fuente Vcc. Desde luego, esta técnica tiene la limitación de que solo un contenido del contador se puede exhibir a la vez. Sin embargo, en muchas aplicaciones esto no es una desventaja. Se podría utilizar una disposición de interrupción mecanica para realizar la función de interrumpir primero un contador y después el otro para los decodificadores e impulsores y exhibiciones; pero el número de contactos de interruptor que se requieren, la complejidad de la conexión y el tamaño físico podrían ser todos desventajas sobre el

método completamente lógico de la figura 4.32.

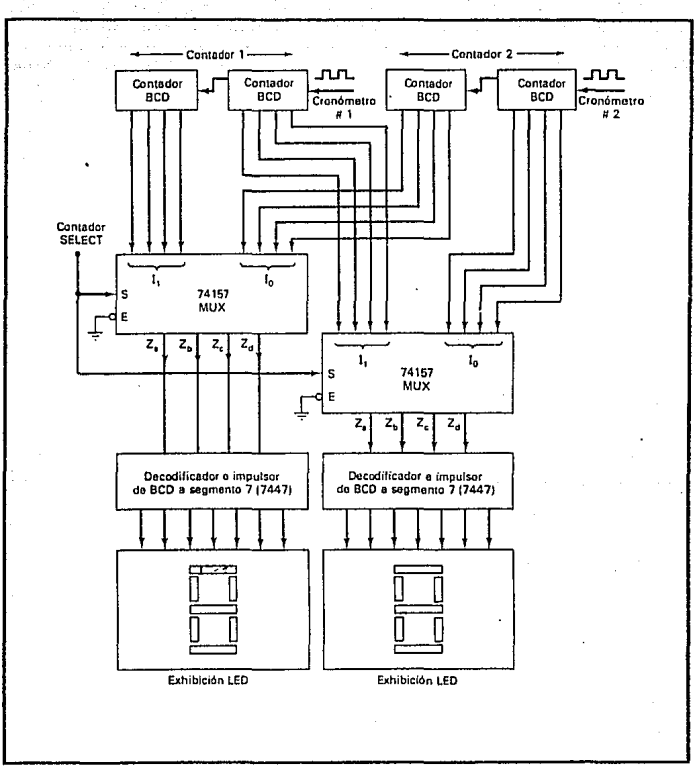


FIG. 4.32 SISTEMA PARA EXHIBIR DOS CONTADORES BCD DE DIGITOS MULTIPLES, UNO A LA VEZ.

Conversión de paralelo a serie. Muchos sistemas digitales procesan datos binarios en forma paralela (todos los bits simultaneamente) ya que es mas rapida. Sin embargo, cuando estos datos deben transmitirse en distancias relativamente largas, la disposición en paralelo es indeseable puesto que requiere un número considerable de líneas de transmisión. Por esta razón, los datos o información binaria que están en forma paralela a menudo se convierten en forma de serie antes de ser transmitidos a un destino remoto. Un método para efectuar esta conversión de paralelo a serie hace uso de un multiplexor, como se ilustra en la figura 4.33.

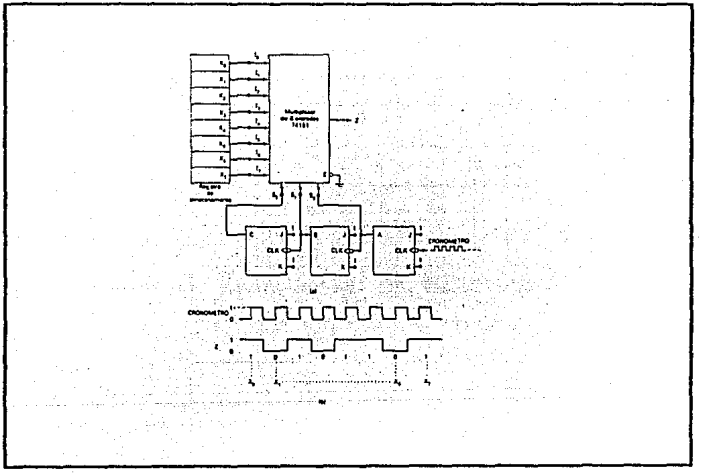


FIG. 4.33 (a) CONVERTIDOR DE PARALEO A SERIE; (b) FORMAS DE ONDA PARA X7X6X5X4X3X2X1X0 = 10110101.

Los datos figuran en forma paralela en las salidas del registro X y son alimentadas al multiplexor de ocho entradas. Se usa un

contador de 3 bits (MOD-8) para ofrecer los bits del código de selección S_2 S_1 S_0 de manera que entren en un ciclo de 000 a 111 cuando se apliquen pulsaciones del cronometro. En esta forma, la salida del multiplexor sera X_0 durante el primer periodo del reloj, X_1 en el segundo periodo del reloj, etcetera. La salida Z es una ondiforme que es una representación gradual de los datos de entrada en paralelo. Las ondas de la figura corresponden al caso donde $X_7X_6X_5X_4X_3X_2X_1X_0$ = 10110101. Este proceso de conversion emplea un total de ocho ciclos del cronometro. Nótese que X_0 (LSB) se transmite primero y X_7 (MSB) se transmite al final.

Generación de funciones lógicas. Los multiplexores se pueden utilizar para implantar funciones lógicas directamente desde una tabla de verdad sin necesitar simplificación. Cuando se usan con este fin las entradas de selección fungen como variables lógicas y cada entrada de datos se conecta permanentemente en ALTA o BAJA, según se necesite para satisfacer la tabla de verdad.

La figura 4.34 ilustra la forma en que un selector de datos de ocho entradas puede usarse para instrumentar el circuito lógico que cumpla con la tabla de verdad dada. Las variables de entrada A, B, C se conectan a S_0, S_1, S_2, respectivamente, de manera que los niveles en estas entradas determinen que entrada de datos aparece en la salida Z. De acuerdo con la tabla de verdad, se supone que Z es BAJA cuando CBA= 000. Por tanto, la entrada del multiplexor Z_0 debe conectarse a BAJA. De igual manera, se supone que Z es BAJA para CBA= 011, 100, 101 y 110, de modo que las entradas I_3, I_4, I_5, e I_6 deben estar conectadas tambien a BAJA. Los otros grupos de condiciones CBA deben producir Z= 1, de manera que las entradas del multiplexor I_1, I_2 e I_7 se conecten permanentemente a ALTA.

Es fácil observar que cualquier tabla de verdad de tres variables puede instrumentarse con este multiplexor de ocho entradas. Este método de implantación a menudo es mas efectivo que el uso de compuertas lógicas separadas. Por ejemplo, si escribimos la expresión de la suma de productos para la tabla de verdad de la figura 4.34, tenemos

$$Z = AB'C' + A'BC' + ABC$$

Esto no puede simplificarse ni algebraicamente ni por el método de Karnaugh K, de manera que su implantación de compuertas

requeriría tres inversores y cuatro compuertas NAND, para hacer un total de dos circuitos integrados (IC).

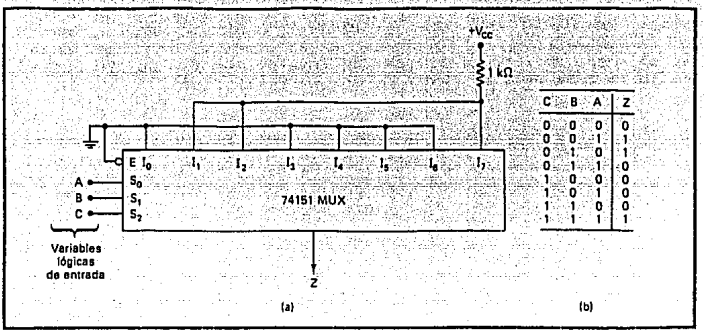


FIG. 4.34 MULTIPLEXOR QUE SE USA PARA IMPLEMENTAR UNA FUNCION LOGICA DESCRITA POR LA TABLA DE VERDAD.

4.7. DEMULTIPLEXORES.

Un multiplexor toma varias entradas y transmite una de ellas a la salida. Un demultiplexor efectua la operacion contraria; toma una sola entrada y la distribuye en varias salidas. La figura 4.35 muestra el diagrama general de un demultiplexor (DEMUX).

Las flechas grandes que corresponden a entradas y salidas pueden representar una o mas lineas. El codigo de entrada de seleccion determina hacia que salida se transmitira la entrada DATA. En otras palabras, el demultiplexor o distribuidor de datos toma una fuente de datos de entrada y la distribuye selectivamente a 1 de N canales de salida, igual que un interruptor de multiples posiciones.

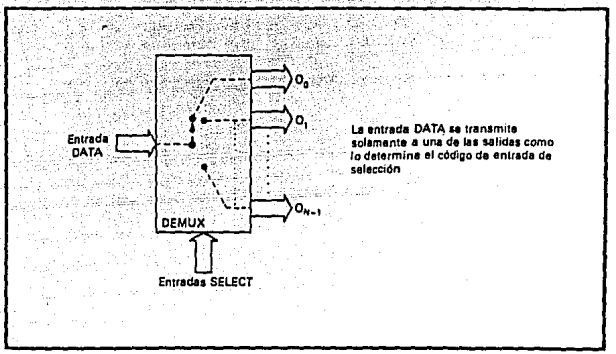


FIG. 4.35 DEMULTIPLEXOR GENERAL.

4.7.1. DEMULTIPLEXOR DE 1 A 8 LINEAS.

La figura 4.36 muestra el diagrama lógico de un distribuidor de datos que distribuye una línea de entrada a ocho líneas de salida. La línea de entrada de datos individual I se conecta a las ocho compuertas AND, pero solo una de estas compuertas será activada por las líneas de entrada SELECT. Por ejemplo, con S_2 S_1 S_0= 000, solamente la compuerta AND0 será activada, y la entrada de datos I figurará en la salida O_0. Otros codigos SELECT ocacionan que la entrada I llegue a las otras salidas. La tabla de verdad resume la operacion.

El circuito distribuidor de datos de la figura 4.36 es muy similar al circuito decodificador de 3 a 8 líneas de la figura 4.15 exepto que se a agregado una cuarta entrada (I) a cada compuerta. Ya antes se señalo que muchos decodificadores IC tienen una entrada ACTIVADA (ENABLE), que es una entrada extra que se añade a las compuertas del decodificador. Este tipo de circuito decodificador puede usarse por tanto como demultiplexor (demultiplexer), con las entradas de codigo binario (por ejemplo, A, B, C de la figura 4.15) que se sirven como las entradas SELECT; y la entrada ENABLE que sirve como la entrada de datos I. Por esta

razón, los fabricantes de circuitos impresos o integrados a menudo llaman a este tipo de dispositivo decodificador/demultiplexor, y se puede usar para desempeñar una función o la otra.

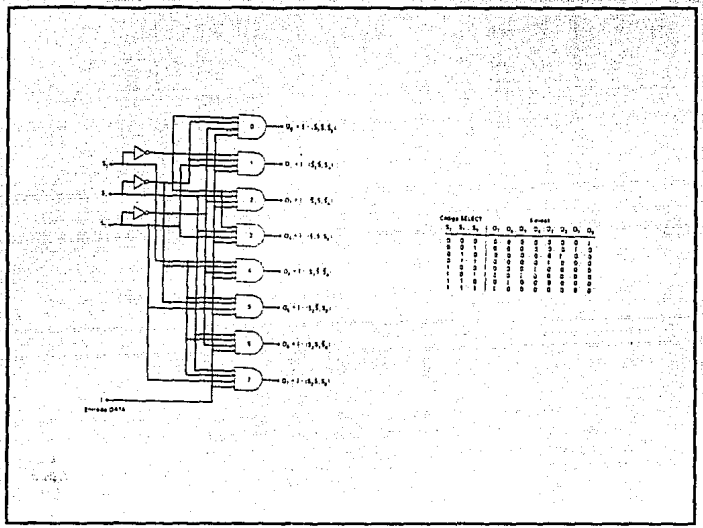


FIG. 4.36 DEMULTIPLEXOR DE 1 A 8 LINEAS.

Ya antes se observó la forma en que se utiliza el 74LS138 como decodificador 1 de 8. La figura 4.37 muestra como puede emplearse para que funcione como demultiplexor o distribuidor de datos. La entrada activada E_1' se usa como la entrada de datos I, en tanto que las otras dos entradas activadas se mantienen en sus estados activos. Las entradas A_2 A_1 A_0 sirven como codigo de seleccion. Para ilustrar la operacion, supongamos que las entradas de seleccion son 000. Con este codigo de entrada, la unica salida que puede activarse es O_0', en tanto que todas las otras salidas son ALTAS. O_0' pasara a BAJA solo si E_1' cambia a BAJA y sera ALTA si E_1' cambia a ALTA. Dicho de otra manera, O_0' seguira la señal en

Ei* (es decir, la entrada de datos, I) mientras todas las otras salidas permanecen ALTAS. En forma análoga, un código de selección diferente aplicado a A_2 A_1 A_0 ocacionará que la salida correspondiente siga la entrada de datos, I.

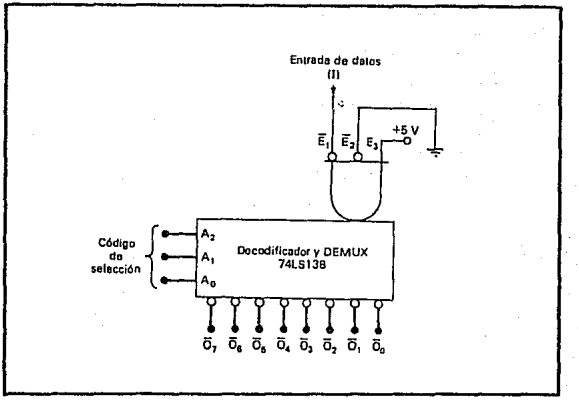


FIG. 4.37 DECODIFICADOR 74LS138 QUE SE USA COMO DEMULTIPLEXOR.

4.8. COMPARADOR DE MAGNITUD.

La comparación de dos números es una operación que determina si un numero es mayor que, menor que o igual a otro número. Un comparador de magnitud es un circuito combinacional que compara dos numeros, A y B y determina sus magnitides relativas. La salida de la comparcion se especifica por tres variables binarias que indican si $A > B$, $A = B$ o $A < B$.

El circuito para comparar dos números de n bits tiene 2^{2n} entradas en la tabla de verdad y llega a ser demasiado engorroso aun con $n = 3$. Por otra parte, como puede sospecharse, un circuito comprador posee cierto grado de regularidad. Las funciones

digitales que poseen una regularidad inherente bien definida por lo comun pueden diseñarse mediante un procedimiento algoritmico, si se encuentra que existe uno. Un algoritmo es un procedimiento que especifica un conjunto finito de pasos mediante los cuales, si se siguen, dan la solucion a un problema. Mostraremos este metodo derivando un algoritmo para el diseño de un comparador de magnitud de 4 bits.

El algoritmo es una aplicación directa del procedimiento que una persona utiliza para comparar las magnitudes relativas de dos numeros. Considerense dos numeros, A y B, con cuatro digitos cada uno. Se escriben los coeficientes de numeros con significacion decreciente como sigue :

$$A = A_3A_2A_1A_0$$

$$B = B_3B_2B_1B_0$$

en donde cada letra con subíndice representa uno de los dígitos en el numero. Los dos numeros son iguales si todos los pares de dígitos significativos son iguales, esto es, si $A_3=B_3$ y $A_2=B_2$ y $A_1=B_1$ y $A_0=B_0$. Cuando los numeros son binarios, los dígitos son 1 o 0 y la relacion de igualdad de cada par de bits puede expresarse en forma logica con una funcion de equivalencia :

$$X_i = A_iB_i + A_i'B_i' \qquad i = 0,1,2,3$$

en donde $X_i = 1$ sólo si el par de bits en la posición i son iguales, esto es, si ambos son 1 o ambos son 0.

La igualdad de dos números, A y B, se exhibe en un circuito combinacional por una salida de variable binaria que se designa con el símbolo (A=B). Esta variable es igual a 1 si los numeros de entrada, A y B, son iguales, y es 0 de otra manera. Para que exista la condicion de igualdad, todas las variables X_i deben ser iguales a 1. Esto dicta una operacion AND de todas las variables:

$$(A=B) = X_3X_2X_1X_0$$

la variable binaria (A=B) es igual a 1 sólosi todos los pares de

dígitos de los dos números son iguales.

Para determinar si A es mayor o menor que B, se inspeccionan las magnitudes relativas de pares de dígitos significativos iniciando desde la posion mas significativa. Si los dos digitos son iguales, el par de dígitos de la siguiente posicion significativa mas baja se comparan. Esta comparacion continua hasta que se alcanza un par de digitos desiguales. Si el digito correspondiente a A es 1 y el de B es 0, se concluye que A > B. Si el correspondiente dígito de A es 0 y el de B es 1, se tiene que A < B. La comparacion secuencial puede expresarse en forma logica por las siguientes dos funciones booleanas :

$$(A>B) = A_3B_3' + X_3A_2B_2' + X_3X_2A_1B_1' + X_3X_2X_1A_0B_0'$$

$$(A<B) = A_3'B_3 + X_3A_2'B_2 + X_3X_2A_1'B_1 + X_3X_2X_1A_0'B_0$$

los símbolos (A>B) y (A<B) son variables binarias de salida que son iguales a 1 cuando A > B o A < B, respectivamente.

La implementación de compuertas de las tres variables de salida que acaban de derivarse es mas simple de lo que parece, ya que implica cierta cantidad de repeticion. Las salidas "desiguales" pueden usar las mismas compuertas que se necesitan para generar la salida "igual". El diagrama logico del comparador de magnitud de 4 bits se muestra en la fig. 4.38. Las cuatro salidas X se generan con un circuito de equivalencia (NOR-excluyente) y se aplican a una compuerta AND para dar la variable binaria de salida (A=B). Las otras dos salidas usan las variables X para generar las funciones booleanas que se listaron antes. Esta es una implementacion de nivel multiple y, como se ve claramente, tiene un patron regular. El procedimiento para obtener circuitos comparadores de magnitud para numeros binariosde mas de 4 bits debe se obvio mediante este ejemplo. El mismo circuito puede utilizarse para comparar las magnitudes relativas de dos digitos BCD.

4 . 9 . VERIFICADOR DE PARIDAD.

Un bit de paridad es un esquema para detectar errores durante la transmision de informacion binaria. Un bit de paridad es un bit

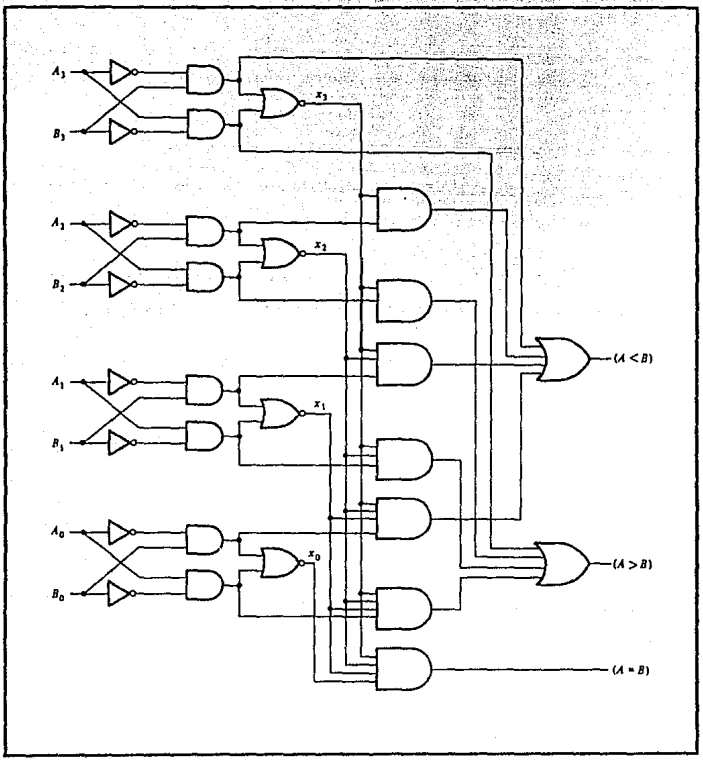


FIG. 4.98 COMPARADOR DE MAGNITUD DE 4 BITS.

adicional incluido con un mensaje binario para hacer que el número de los 1 sea impar o bien par. El mensaje que incluye el bit de

paridad, se transmite y entonces se verifica en la terminal receptora para buscar errores. Un error se detecta si la paridad verificada no corresponde con la transmitida. El circuito que genera el bit de paridad en el transmisor se conoce como "generador de paridad". El circuito que verifica la paridad en el receptor se denomina "verificador de paridad".

Como ejemplo, considérese un mensaje de tres bits que se transmite con un bit de paridad impar. En la tabla 4.4 se muestra la tabla de verdad para el generador de paridad. Los tres bits X, Y y Z constituyen el mensaje y son las entradas al circuito. El bit de paridad P es la salida. Para paridad impar, el bit P se genera de tal modo que el numero total de 1's sea impar (incluyendo P). Mediante la tabla de verdad, se ve que P = 1,

MENSAJE DE 3 BITS			BIT DE PARIDAD GENERADO
X	Y	Z	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

TABLA 4.4 GENERACION DE PARIDAD IMPAR.

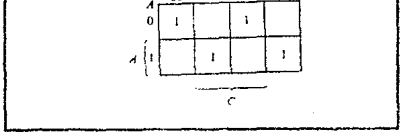


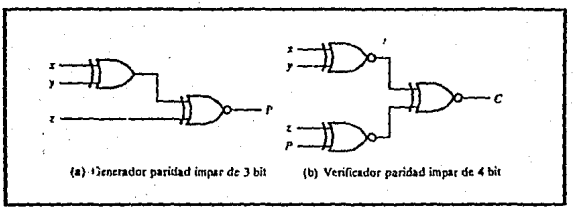
FIG. 4.30 MAPA PARA EL GENERADOR DE PARIDAD IMPAR.

cuando el número de 1's en X, Y y Z es par. Esto corresponde al mapa de la figura 4.39; de modo que la función P puede expresarse como sigue:

$$P = X \oplus Y \odot Z$$

El diagrama lógico para el generador de paridad se muestra en la figura 4.40 (a). Consta de una compuerta OR-Exclusiva de dos entradas y una compuerta de equivalencia de dos entradas. Las dos compuertas pueden intercambiarse y producir la misma función, ya que P es también igual a:

$$P = X \odot Y \oplus Z$$



(a) Generador paridad impar de 3 bit (b) Verificador paridad impar de 4 bit

FIG. 4.40 DIAGRAMAS LOGICOS PARA GENERACION Y VERIFICACION DE PARIDAD.

El mensaje de tres bits y el bit de paridad se transmiten a su destino, donde se aplica a un circuito verificador de paridad. Un error ocurre mediante la transmisión si la paridad de los cuatro bits es par, ya que la información binaria transmitida fue originalmente impar. La salida C del verificador de paridad debe ser 1 cuando ocurre un error, esto es cuando el número de 1's en las cuatro entradas es par. La tabla 4.5 es la tabla de verdad para el circuito verificador de paridad impar, mediante el cual se ve que la función para C consta de ocho miniterminos con valores numericos que tienen un número par de 0's. Esto corresponde al mapa de la figura 4.41; de modo que la función puede expresarse con operadores de equivalencia como sigue :

$$C = X \odot Y \odot X \odot P$$

CUATRO BITS RECIBIDOS				VERIFICACION DE ERROR DE PARIDAD
X	Y	Z	P	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

TABLA 4.5 VERIFICACION DE PARIDAD IMPAR.

El diagrama lógico para el verificador de paridad se muestra en la figura 4.40 (b) y consta de tres compuertas NOR-Exclusivas de dos entradas.

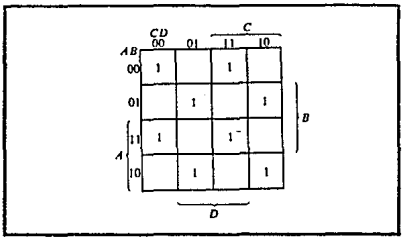


FIG. 4.41 MAPA PARA EL VERIFICADOR DE PARIDAD IMPAR.

Es de interés observar que el generador de paridad puede

implementarse con el circuito de la fig. 4.40(b), si la entrada P se mantiene en forma permanente a logica 0 y la salida se marca como P, en donde la ventaja es que el mismo circuito puede usarse para generación de paridad al igual que para verificación de paridad.

Del ejemplo anterior, es obvio, que los circuitos de generación de paridad y verificacion de paridad siempre tienen una función de salida, que incluye la mitad de los miniterminos cuyos valores numericos tienen ya sea un numero par o bien impar de 1's. Como consecuencia, deben implementarse con compuertas NOR-Exclusivas y/o OR-Exclusivas.

EJERCICIOS PROPUESTOS.

4.1 Un sumador total se puede activar en muchas formas diferentes. La figura 4.42 muestra como se puede construir uno a partir de dos sumadores medios. Elabore una tabla de verdad para esta disposicion y verifique que opera como un FA.

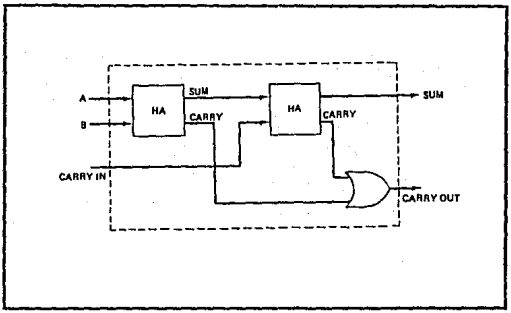


FIG. 4.42 SUMADOR TOTAL.

4.2 Implementar un restador completo con dos medios restadores y una compuesta OR.

4.3 Muestre cómo puede convertirse un sumador completo en un restador completo con la adición de un circuito inversor.

4.4 Es necesario multiplicar dos números binarios, cada uno de dos bits de longitud, con objetos de obtener su producto en binario. Los dos numeros se representan como a1a0 y b1b0, donde el subíndice 0 denota el bit menos significativo.

a) Determine el número de líneas de salida requerido.

b) Encuentre las expresiones booleanas simplificadas para cada salida.

4.5 Muestre la forma de utilizar varios 74LS138 para formar un decodificador de 1 a 16.

4.6 La figura 4.43 muestra la forma en que puede utilizarse un decodificador en la generacion de señales de control. Suponga que ha ocurrido una pulsacion "RESET" al tiempo to y determine la forma de onda de CONTROL en 32 pulsaciones del cronometro.

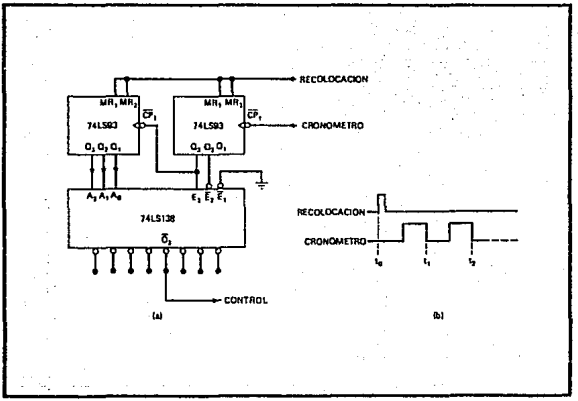


FIG. 4.43 APLICACION DEL DECODIFICADOR.

4.7 Modifique el circuito de la figura 4.43 para generar un forma de onda de control que vaya a LOW de tzo a t24. (La modificacion no requiere logica adicional).

4.8 El circuito de la figura 4.44 utiliza 3 multiplexores de dos entradas. Determine la funcion que realiza este circuito.

4.9 Emplee la idea del ejemplo anterior para acomodar varios multiplexores 1 de 8 de la serie 74151 para formar un multiplexor 1 de 64.

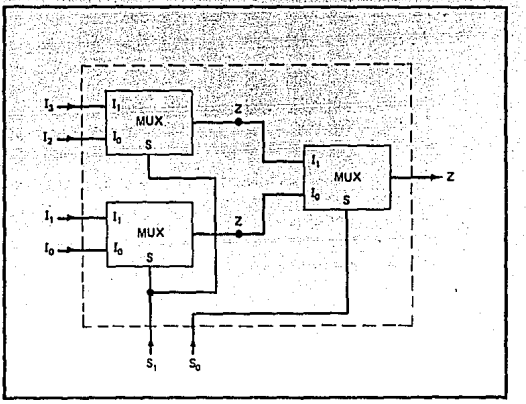


FIG. 4.44 APLICACIONM DE LOS MULTIPLEXORES.

4.10 Muestre la forma en que dos 74157 y un 74151 pueden disponerse para formar un multiplexor 1 de 16 sin otra logica que se necesite. Rotule las entradas Io-I15 para mostrar como corresponden al codigo de seleccion.

4.11 Muestre la forma en que se puede usar un 74151 para generar la funcion logica Z = AB + BC + AC.

CAPITULO V : CIRCUITOS SECUENCIALES

CONTENIDO :

CAPITULO V : CIRCUITOS SECUENCIALES

CAPITULO V: CIRCUITOS SECUENCIALES.

5.1 FLIP - FLOPS.

Los circuitos lógicos que se han considerado hasta ahora son circuitos combinatorios cuyos niveles de salida, en cualquier instante de tiempo, dependen de los niveles presentes de las entradas en ese momento. Cualquier condicion anterior al nivel de entrada no afecta a las entradas, ya que los circuitos logicos combinatorios no tienen memoria. Muchos sistemas digitales están constituidos por circuitos combinatorios y elementos de la memoria.

La figura 5.1 muestra un diagrama de bloque de un sistema digital general que conjuga compuertas lógicas combinatorias con dispositivos de memoria. La porcion combinatoria acepta señales lógicas de entradas externas y de salidas de los elementos de la memoria. El circuito lógico combinatorio opera sobre estas entradas a fin de producir diversas salidas, algunas de las cuales se utilizan para determinar los valores binarios que se almacenarán en los elementos de la memoria. Las salidas de algunos de los elementos de la memoria, a su vez, se dirigen hacia las entradas de compuertas lógicas en los circuitos combinatorios. Este proceso indica que las salidas externas de un sistema digital son función de sus entradas externas y de la información almacenada en los elementos de la memoria.

El elemento de la memoria que se utiliza más ampliamente es el multivibrador biestable o flip-flop, el cual se estudiara en todo este capítulo. El biestable (que se abrevia FF) es un circuito lógico de dos salidas, las cuales son inversas la una de la otra. La figura 5.2 indica estas salidas como Q y Q' (en realidad, se puede usar cualquier letra, pero Q es la mas común). La salida Q se denomina salida FF normal y Q' es la salida FF invertida.

Cuando se dice que un FF está en estado ALTO (1) o bien en estado BAJO (0), esta es la condición presente en la salida Q. Por supuesto, la salida Q' siempre es la inversa de Q.

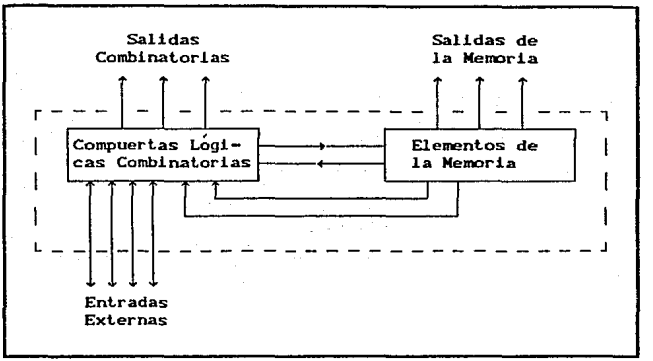


FIG. 5.1 DIAGRAMA DE UN SISTEMA DIGITAL GENERAL.

Hay dos posibles estados operativos para el FF: (1) Q=0, Q'=1; y (2) Q=1, Q'=0. El FF tiene una o más entradas, las cuales se emplean para definir que el FF alterne entre estos dos estados. Como observaremos, cuando se pulsa una entrada para enviar al FF a cierto estado, el FF permanecerá en ese estado aun después de que la entrada vuelva a ser normal. Esta es su característica de memoria.

El biestable, incidentalmente, se conoce por otros nombres, inclusive multibivibrador biestable, cerrojo y binario, pero en términos generales se utilizara biestable porque es la designación más común en el campo digital. Otros elementos de memoria, se usan así mismo en los sistemas digitales; pero los biestables son los más versátiles debido a su alta velocidad de operación, la facilidad con que se puede almacenar y extraer información de ellos y la facilidad con que se pueden interconectar con compuertas lógicas.

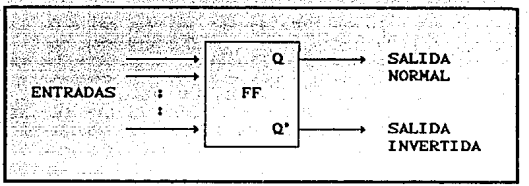


FIG. 5.2 SIMBOLO GENERAL DEL BIESTABLE.

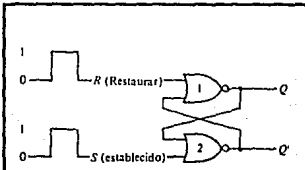
Un circuito flip-flop puede mantener un estado binario en forma indefinida (en tanto se suministre potencia al circuito) hasta que recibe la direccion de una señal de entrada para casmbiar estado. La diferencia principal entre los diversos tipos de flip-flops esta en el numero de entradas que poseen y en la manera en la cual las entradas afectan el estado binario. Los tipos mas comunes de flip-flop se exponen a continuacion.

5.1.1 FLIP - FLOP RS ASINCRONO.

Un circuito flip-flop puede construirse mediante dos compuertas NAND o dos compuertas NOR. Estas construcciones se muestran en los diagramas logicos de las Figs. 5.3 y 5.4. Cada circuito forma un flip-flop basico bajo el cual puede construirse otros tipos mas complicados. La conexion y acoplamiento cruzado mediante la salida de una compuerta a la entrada de otra constituye una trayectoria de retroalimentacion. Por esta razon, los circuitos se clasifican como circuitos secuenciales asincronos cada flip-flop tiene dos salidas, Q y Q', y dos entradas, ajustar (set) y restaurar (reset). Este tipo de flip-flop algunas veces se denomina flip-flop RS directamente acoplado o seguro (latch) SR. La R y S son las iniciales de los dos nombres de entrada (set y reset en ingles).

Para realizar la operacion del circuito en la figura 5.3, debe recordarse que la salida de una compuerta NOR es 0 si cualquier entrada es 1, y que la salida es 1 solo cuando todas las entradas son 0. Como punto de inicio, se supone que la entrada ajuste (set) es 1, y la entrada restaurar (reset) es 0. Ya que la compuerta 2 tiene una entrada de 1, su salida Q' debe ser 0, la

cual pone ambas entradas de la compuerta 1 en 0, de modo que la salida Q es 1. Cuando la entrada ajuste se regresa a 0, la salida permanece igual, debido a que la salida Q permanece en 1, dejando una entrada de la compuerta 2 en 1. Esto causa que la salida Q' permanezca en 0, lo cual deja ambas entradas de compuerta 1 en 0, de modo que la salida Q está en 1. En la misma forma es posible mostrar que un 1 en la entrada de restaurar cambia la salida Q a 0 y Q' a 1. Cuando la entrada de restaurar vuelve a 0, las salidas no cambian.



(A)

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(*)
0	1	0	1	
0	0	0	1	(**)
1	1	0	0	

(*) DESPUES QUE S=1, R=0
(**) DESPUES QUE S=0, R=1

(B)

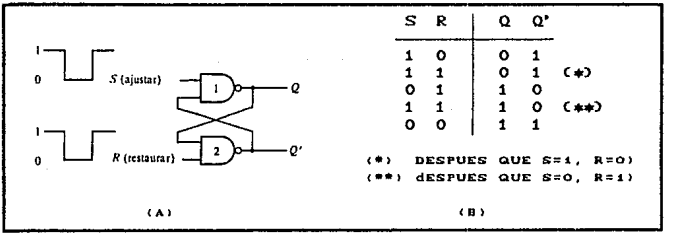
5.9 CIRCUITO FLIP-FLOP BASICO CON COMPUERTA NOR.
(A) DIAGRAMA BASICO, (B) TABLA DE VERDAD.

Cuando se aplica un 1 a ambas entradas de ajuste (set) y restaurar (reset), tanto la salida Q como la Q' van a 0. Esta condición viola el hecho de que las salidas Q y Q' son los complementos una de otra. En la operación normal esta condición debe evitarse al tener la seguridad de que los 1 no son aplicables en forma simultánea a ambas entradas.

Un flip-flop tiene dos estados útiles; cuando Q=1 y Q'=0, está en el estado ajuste (o estado 1). Cuando Q=0 y Q'=1, está en el estado despejado (o estado 0). Las salidas Q y Q' son complementarias una de otra y se refieren como las salidas normal y complementaria, respectivamente. El estado binario del flip-flop se toma para que sea el valor de la salida normal.

Bajo operación normal, ambas entradas permanecen en 0 a menos que tenga que cambiarse el estado del flip-flop. La aplicación de un 1 momentáneo a la entrada de ajuste provoca que el flip-flop pase al estado ajuste. La entrada ajuste debe volver a 0 antes de que un 1 se aplique a la entrada de restaurar. Un 1 momentáneo aplicado a la entrada de restaurar causa que el flip-flop vaya al

estado despejado. Cuando ambas entradas son inicialmente 0, un 1 aplicado a la entrada de puesto mientras el flip-flop esta en el estado ajuste o un 1 aplicado a la entrada de restaurar mientras el flip-flop está en el estado despejado deja las salidas sin cambio. Cuando se aplica un 1 a ambas entradas de ajuste y restaurar, ambas salidas pasan a 0. Este estado es indefinido y por lo común se evita. Si ambas entradas ahora van a 0, el estado de flip-flop es indeterminado y depende de cual entrada permanezca en 1 mas tiempo antes de la transición a 0.



5.4 CIRCUITO FLIP-FLOP BASICO CON COMPUERTA NAND.
(A) DIAGRAMA BASICO, (B) TABLA DE VERDAD.

El circuito flip-flop NAND básico en la figura 5.4 opera con ambas entradas normalmente en 1, a menos que el estado del flip-flop tenga que cambiarse. La aplicación de un 0 momentáneo a la entrada de ajuste causa que la salida Q vaya a 1 y Q' a 0, poniendo por tanto el flip-flop en el estado de ajuste. Despues de que la entrada de ajuste regresa a 1, un 0 momentáneo en la entrada de restaurar provoca una transición al estado despejado. Cuando ambas entradas van a 0, ambas salidas irán a 1, una condición que se evita en la operación normal del flip-flop.

5.1.2 FLIP - FLOP RS SINCRONO.

El flip-flop básico, tal como está, es un circuito secuencial asíncrono. Por la adición de compuertas a las entradas del circuito básico, puede hacerse que el flip-flop responda a niveles de entrada durante la ocurrencia de un pulso de reloj. El

flip-flop RS con reloj que se muestra en la figura 5.5(a) consta de un flip-flop básico NOR y dos compuertas AND. Las salidas de las dos compuertas AND permanecen en 0 en tanto que el pulso de reloj (abreviado CP, de las iniciales en inglés de clock pulse) sea 0, sin importar los valores de entrada S y R. Cuando el pulso de reloj va a 1, se permite que la información de las entradas S y R alcancen al flip-flop básico. El estado de ajuste

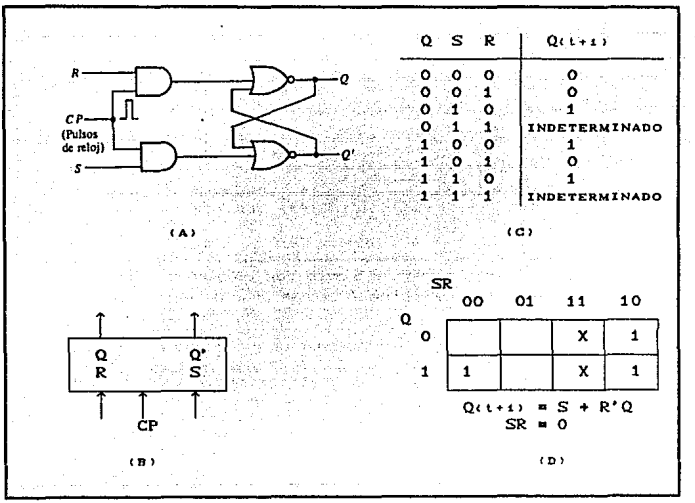


FIG. 5.5 FLIP-FLOP RS CON PULSOS DE RELOJ.
(A) DIAGRAMA LOGICO, (B) SIMBOLO GRAFICO
(C) TABLA CARACTERISTICA, (D) ECUACION CARACTERISTICA.

se alcanza con S=1, R=0 y CP=1. Para cambiar al estado despejado, las entradas deben ser S=0, R=1 y CP=1. Tanto con S=1 y R=1, la ocurrencia de un pulso de reloj provoca que ambas salidas momentáneamente a 0. Cuando se elimina el pulso, el estado del

flip-flop es indeterminado, esto es, puede resultar cualquier estado, dependiendo de si la entrada de ajuste o la de restaurar del circuito flip-flop basico permanezca en 1 durante un tiempo mas prolongado antes de la transicion a 0 al fin del pulso.

El símbolo gráfico para el flip-flop RS con reloj se muestra en la figura 5.5(b). Tiene tres entradas: S, R y CP. La entrada CP no esta indicada dentro de la caja, debido a que se reconoce por el triangulo pequeño marcado. El triangulo es un símbolo para un indicador dinamico y denota el hecho de que el flip-flop responde a una transicion de reloj en una señal de bajo nivel (binario 0) a un alto nivel (binario 1). Las salidas del flip-flop están marcadas con Q y Q' dentro de la caja. Puede asignarse al flip-flop una variable con nombre diferente aunque Q este escrita dentro de la caja. En ese caso, la letra que se elige para la variable del flip-flop se marca fuera de la caja junto a la línea de salida. El estado del flip-flop esta determinado por el valor de su salida normal Q. Si se desea obtener el complemento de la salida normal, no es necesario insertar un invertidor, ya que el valor complementado esta disponible directamente mediante la salida Q'.

La tabla característica para el flip-flop se muestra en la figura 5.5(c). En esta tabla se resume la operación del flip-flop en una forma tabular. Q es el estado binario del flip-flop en un momento dado (referido como estado presente), las columnas S y R dan los valores posibles de las entradas y Q(t + 1) es el estado del flip-flop despues de la ocurrencia de un pulso de reloj (referida como estado siguiente).

La ecuación característica del flip-flop se deriva en el mapa en la figura 5.5(d). Esta ecuacion especifica el valor del estado siguiente como una función del estado presente y las entradas. La ecuación característica es una expresión algebraica para la información binaria de la tabla característica. Los dos estados indeterminados están marcados con X en el mapa, ya que pueden resultar en 1 o bien en 0. Sin embargo, la relación SR=0 debe incluirse como parte de la ecuación característica para especificar que tanto S como R no pueden ser iguales a 1 en forma simultanea.

5.1.3 FLIP - FLOP TIPO D CON CRONOMETRO.

La figura 5.6(a) muestra el símbolo de un biestable D con cronometro que se activa en una TSP en la entrada CLK. La entrada D es sincrónica que controla el estado del FF de acuerdo con la tabla de verdad correspondiente. La operación es muy simple; Q pasará el mismo estado que está presente en la entrada D cuando ocurre una TSP en CLK. Esto se ilustra con las formas de onda de la figura 5.6(b).

Nótese que cada vez que una TSP ocurre en la entrada CLK, Q toma el valor presente en la entrada D. Las transiciones en sentido negativo en CLK no tiene efecto y la entrada D no tiene efecto excepto cuando ocurre una TSP.

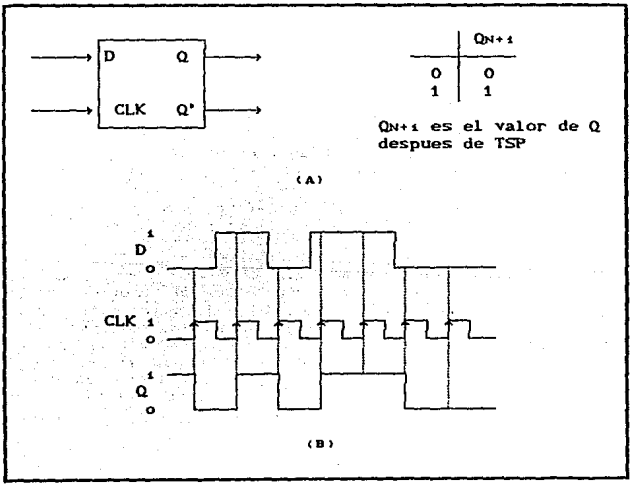


FIG. 5.6 (A) FF D QUE SE ACTIVA EN TRANSICIONES EN SENTIDO POSITIVO; (B) FORMAS DE ONDA.

Los multivibradores biestables D con transición negativa activada se encuentran disponibles también y operan exactamente en la misma forma, a excepción que responden sólo a TSN en la entrada CLK. El símbolo para estos FF tendrán un pequeño círculo en la entrada CLK.

5.1.3.1 PUESTA EN ACCION DEL FLIP - FLOP D.

Un FF D con transición activada se pone en práctica fácilmente agregando un INVERSOR sencillo al FF S-C con transición activada, como el que se muestra en la figura 5.7. Si se prueban ambos valores de D, debe observarse que $Q_{N+1} = D$.

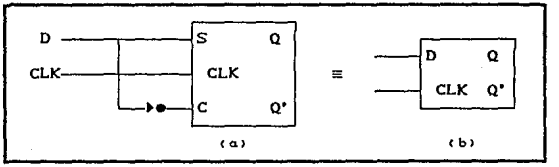


FIG. 5.7 UN FF D CON TRANSICION ACTIVADA PUESTO EN ACCION A PARTIR DEL FF S-C.

5.1.3.2 TRANSFERENCIA DE DATOS EN PARALELO.

En este punto quizá el lector se cuestione acerca del uso del FF D, ya que parece que la salida Q es la misma que la entrada D. No del todo; recordemos que Q toma el valor de D sólo en ciertos momentos y así, no es idéntico a D (por ejemplo, véanse las formas de onda de la figura 5.6).

En muchas aplicaciones del FF D, la salida Q debe tomar el valor de esta entrada D solamente en instantes de tiempo definidos con exactitud. Un ejemplo de esto se ilustra en la figura 5.9. Las salidas X, Y, Z del circuito lógico se transferirá a los FF $Q1$, $Q2$ y $Q3$ para su almacenamiento. Utilizando los FF D, los niveles presentes en X, Y y Z se

transferirán a Q_1, Q_2 y Q_3, respectivamente, hasta la aplicación de una pulsación de transferencia a las entradas comunes CLK. Los FF pueden almacenar estos valores para procesarse despues. Este es un ejemplo de la transferencia en paralelo de datos binarios; los bits X, Y y Z se transfieren simultaneamente.

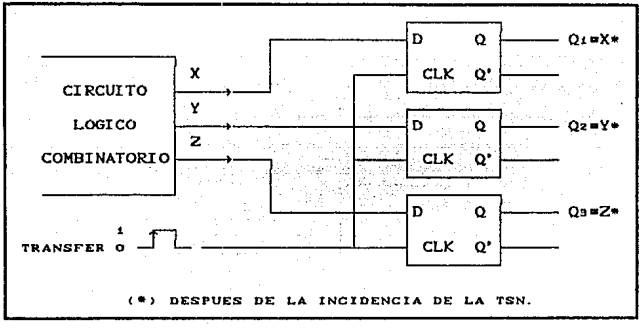


FIG. 5.9 TRANSFERENCIA EN PARALELO DE DATOS BINARIOS MEDIANTE EL USO DE BIESTABLES TIPO D.

5.1.4 FLIP - FLOP JK CON CRONOMETRO.

La figura 5.10(a) muestra un biestable J-K con cronometro que es activado por la transicion en sentido positivo de la señal del reloj. Las entradas J y K controlan el estado del FF en la misma forma que las entradas S y C lo hacen para el FF S-C con cronometro, excepto por una diferencia principal: la condicion J=K=1 no genera una salida ambigua. Para esta condicion 1,1, el FF siempre pasara a su estado opuesto cuando se efectue la transicion en sentido positivo de la señal del reloj. A este se le denomina modo articulado de operacion. En este modo, si J y K se dejan en el estado ALTO, el FF cambiara (se articulara) en cada pulsacion del cronometro.

La tabla de verdad de la figura 5.10(a) resume la forma en que

el FF J-K responde a la TSP por cada combinación de J y K. La condición J=K=1 produce $Q_{N+1}=Q'_N$, lo cual significa que el nuevo valor de Q será el inverso del valor que tenía antes de la realización de la TSP; esta es la operación de articulación.

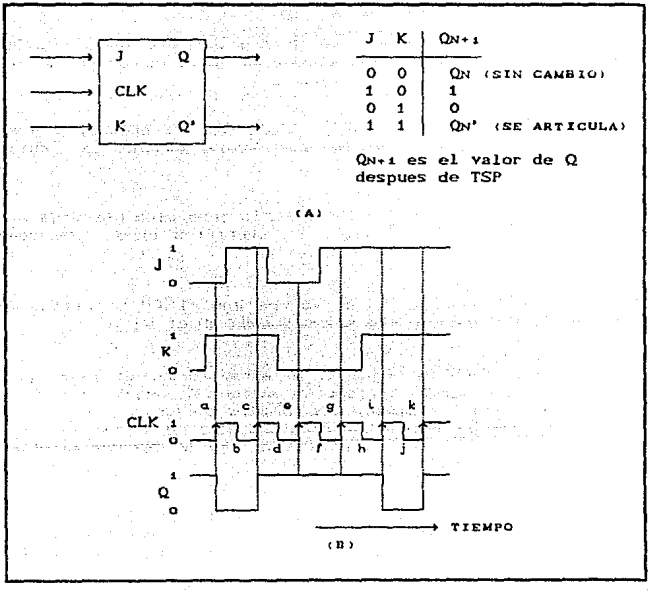


FIG. 5.10 (A) FF J-K QUE SE ACTIVA EN TRANSICIONES EN SENTIDO POSITIVO; (B) FORMAS DE ONDA.

La operación de este FF se ilustra por medio de las formas de onda de la figura 5.10(b). Una vez más se supone que los requisitos de tiempo de constitución y de contención se cumplen adecuadamente.

1. Inicialmente todas las entradas son 0 y la salida Q se supone como 1.

2. Cuando la transición en sentido positivo de la primera pulsación del cronometro ocurre (punto a) la condición J=0, K=1 existe. Así, el FF será anulado al estado Q=0.

3. La segunda pulsación del reloj encuentra que J=K=1 cuando realiza su transición positiva (punto c). Esto ocaciona que el FF se articule a su estado opuesto, Q=1.

4. En el punto e de la forma de onda del cronómetro, J y K son 0, de manera que el FF no cambia estados en esta transición.

5. En el punto g, J=1 y K=0. Esta es la condición que fija a Q en el estado 1. Sin embargo, se encuentra en 1, así que permanecerá ahí.

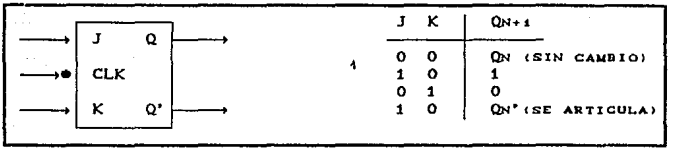
6. En el punto i, J=K=1, de manera que el FF se articula hacia ese estado opuesto. Lo mismo sucede en el punto k.

Debe observarse también, a partir de estas formas de onda, que el FF no es afectado por la transición en sentido negativo de las pulsaciones del reloj. Asimismo, los niveles de entrada J y K no tienen efecto excepto cuando aparece la TSP de la señal del cronometro. Las entradas J y K por sí mismas no pueden valerse del FF para cambiar estados.

La figura 5.11 muestra el símbolo correspondiente a un biestable J-K con cronómetro que se activa en las transiciones en sentido negativo de la señal del reloj. El círculo pequeño en la entrada CLK indica que este FF se activará cuando la entrada CLK pase de 1 a 0. Este FF opera en la misma forma que el FF con transición positiva de la figura 5.10, excepto que la salida puede cambiar estados sólo en transiciones en sentido negativo de señales de un cronometro (puntos b, d, f, h y j). Ambas polaridades de los FF J-K con arista activada se usan comunmente.

La condición J=K=1, la cual genera la operación de

articulación, se usa ampliamente en todos los tipos de contadores binarios. El FF J-K actualmente disfruta de un intenso uso en casi todos los sistemas digitales modernos.



5.11 UN FF J-K QUE SE ACTIVA EN TRANSICIONES EN SENTIDO NEGATIVO.

5.1.4.1 PUESTA EN ACCION EN NAND DEL FLIP - FLOP JK CON TRANSICION ACTIVA.

Una versión simplificada de los circuitos internos de un FF J-K con transición activada se muestra en la figura 5.12. Las

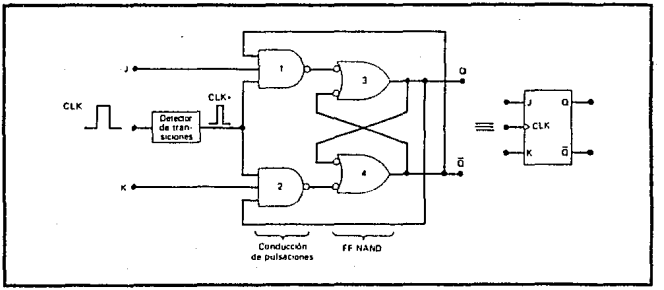


FIG. 5.12 PUESTA EN ACCION DE LA COMPUERTA NAND DEL FF J-K CON TRASICION ACTIVADA.

salidas Q y Q' se retroalimentan a las compuertas NAND de conducción de pulsaciones. Esta conexión retroalimentada es la que da al FF J-K su operacion de articulacion para la condicion J=K=1.

Ahora examinemos esta condición de articulación con más detalle suponiendo que J=K=1 y que Q reposa en el estado BAJO cuando ocurre una pulsación CLK. Con Q=0/Q'=1, la compuerta NAND 1 conducirá a CLK* (invertido) a la entrada SET de NAND FF para producir Q=1. Si suponemos que Q es HIGH cuando ocurre una pulsación CLK, la compuerta NAND 2 conducirá a CLR* invertido a la entrada "CLEAR" del FF NAND para producir Q=0. De este modo, Q siempre termina en el estado opuesto.

A fin de que la operación de articulación funcione como se describió antes, la pulsación CLK* debe ser muy breve. Tiene que retornar a 0 antes de que las salidas Q y Q' se articulen hacia sus nuevos valores; en caso contrario los nuevos valores de Q y Q' ocacionaran que la pulsación CLK* articule el FF NAND una vez mas.

EJEMPLO : *Como puede modificarse un FF J-K para que funcione como un FF D?*

SOLUCION : *La modificación, que se muestra en la figura 5.8 es la misma que se hizo al FF S-C de la figura 5.6.*

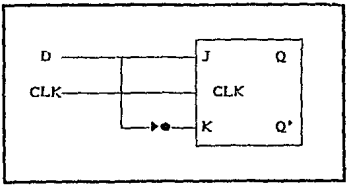


FIG. 5.8 UN FF D PUESTO EN ACCION A PARTIR DE UN FF J-K.

5.1.5 FLIP - FLOP TIPO T.

El flip-flop T es una versión de una sola entrada del flip-flop JK. Como se muestra en la figura 5.13(a), el flip-flop T se obtiene mediante un tipo JK si ambas entradas se ligan. La denominación T proviene de la capacidad del flip-flop para "conmutar" (de la inicial del término en inglés: toggle), o cambiar de estado. Sin importar el estado presente del flip-flop, asume el estado complementario cuando ocurre el pulso de reloj mientras la entrada T es lógica 1. El símbolo, la tabla característica y la ecuación característica del flip-flop T se muestra en la figura 5.13 partes (b), (c) y (d), respectivamente.

Los flip-flop que se introducen en esta sección son los tipos disponibles más comunes en el comercio. Los procedimientos de análisis y diseño que se desarrollan en este capítulo son aplicables para cualquier flip-flop temporizado una vez que se define su tabla característica.

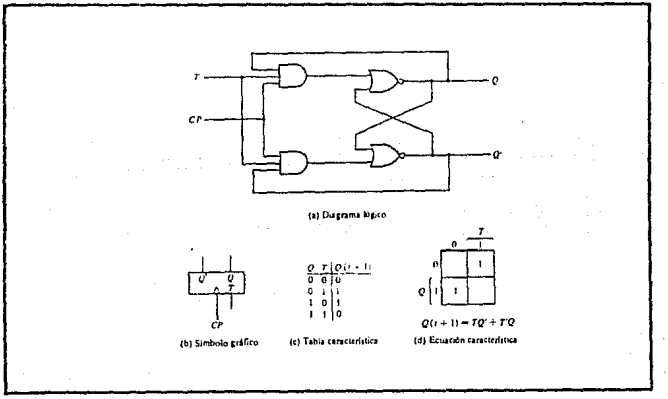


FIG. 5.13 FLIP-FLOP T CON PULSOS DE RELOJ.

5.1.6 TABLAS DE EXITACION FLIP - FLOP.

Una tabla característica define la propiedad lógica del flip-flop y caracteriza por completo su operación. Los circuitos integrados flip-flops algunas veces se definen por una tabla característica tabulada un poco diferente. Esta segunda forma de las tablas características para los flip-flops RS, JK, D y T se muestra en la tabla 5.1.

S	R	Qt+1
0	0	Qt
0	1	0
1	0	1
1	1	?

(a) RS

J	K	Qt+1
0	0	Qt
0	1	0
1	0	1
1	1	Q't

(b) JK

D	Qt+1
0	0
1	1

(c) D

T	Qt+1
0	Qt
1	Q't

(d) T

TABLA 5.1 TABLAS F-F CARACTERISTICAS.

En la Tabla 5.1 se define el estado de cada flip-flop como una función de sus entradas y su estado previo. Q(t) se refiere al estado presente y Q(t + 1) al siguiente estado después de la ocurrencia de un pulso de reloj. La tabla característica para los flip-flops RS muestra que el estado siguiente es igual al estado presente cuando tanto la entrada S como la R son 0. Cuando la entrada R es igual a 1, el siguiente pulso de reloj despeja el flip-flop. Cuando la entrada S es igual a 1, el siguiente pulso de reloj establece el flip-flop. El signo de interrogación para el siguiente estado cuando tanto S como R son iguales a 1 en forma simultánea designa un estado siguiente indeterminado.

La tabla para el flip-flop JK es la misma que para el RS, cuando J y K se reemplazan por S y R, respectivamente, exepto para el caso indeterminado. Cuando tanto J como K son iguales a 1, el estado siguiente es igual al complemento del estado presente, esto es, Q(t + 1)=Q'(t). El estado siguiente del flip-flop D depende por completo de la entrada D y es independiente del estado presente. El siguiente estado del flip-flop T es el mismo que el estado presente si T=0 y se complementa si T=1.

La tabla característica es útil para análisis y para definir la operación del flip-flop. Especifica el estado siguiente cuando las entradas del estado presente se conocen. Durante el proceso de diseño, por lo común se conoce la transición del estado presente al estado siguiente y se desea encontrar las condiciones de entrada del flip-flop, que provocaran la transición requerida. Por esta razón, se necesita una tabla que liste las entradas requeridas para el cambio dado de estado. Dicha lista se denomina tabla de excitación.

La tabla 5.2 se presentan las tablas de excitación para los cuatro flip-flops. Cada tabla consta de dos columnas, Q(t) y Q(t + 1), y una columna para cada entrada para mostrar como se logra la transición requerida. Hay cuatro transiciones posibles desde el estado presente al estado siguiente. Las condiciones requeridas de entrada para cada una de las cuatro transiciones se derivan mediante la información disponible en la tabla característica. El símbolo X en las tablas representa condiciones no importa, esto es, no importa si la entrada es 1 o 0.

Qt	Qt+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

(a) RS

Qt	Qt+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(b) JK

Qt	Qt+1	D
0	0	0
0	1	1
1	0	0
1	1	1

(c) D

Qt	Qt+1	T
0	0	0
0	1	1
1	0	1
1	1	0

(d) T

TABLA 5.2 TABLAS DE EXCITACION F-F.

5.1.6.1 TABLAS DE EXITACION FLIP - FLOP RS.

La tabla de excitación para el flip-flop RS se muestra en la Tabla 5.2(a). En el primer renglón se muestra el flip-flop en el estado 1 en el tiempo t. Se desea dejarlo en el estado 0 después de la ocurrencia del pulso. Mediante la tabla característica, se encuentra que si S al igual que R son 0, el flip-flop no cambiara de estado. En consecuencia, tanto la entrada S como la R deben 0. Sin embargo, en realidad no importa si R se hace un 1, cuando

ocurre un pulso, ya que resulta en que deja el flip-flop en estado, 0. Por eso R puede ser 1 o 0 y el flip-flop permanecerá en el estado 0 en t + 1. Así, la entrada bajo R se marca X como condición no importa.

Si el flip-flop está en el estado 0 y se desea que pase al estado 1, entonces mediante la tabla característica, se encuentra que la unica forma de hacer Q(t + 1) igual a 1 es hacer S=1 y R=0. Si el flip-flop va a tener una transición del estado 1 al estado 0, debe tenerse S=0 y R=1.

La última condición que puede ocurrir es para que el flip-flop este en el estado 1 y permanezca en el estado 1. Por supuesto R debe ser 0; no se desea despejar el flip-flop. Sin embargo, S puede ser ya sea un 0 o un 1. Si es 0, el flip-flop no cambia y permanece en el estado 1; si es un 1, se establece el flip-flop en el estado 1, como se desea. Así que, S se lista como una condición no importa.

5.1.6.2 TABLAS DE EXITACION FLIP - FLOP JK.

La tabla de excitación para el flip-flop JK se muestra en la Tabla 5.2(b). Cuando tanto el estado presente como el estado siguiente son 0, la entrada J deberá permanecer en 0 y la entrada K podrá ser 0 o bien 1. En forma similar, cuando tanto el estado presente como el siguiente son 1, la entrada K debe permanecer en 0 mientras la entrada J puede ser 0 o 1. Si el flip-flop va a tener una transición del estado 0 al estado 1, J debe ser igual a 1 ya que la entrada J establece el flip-flop. No obstante, la entrada K puede ser 0 o bien un 1. Si K=0, condición la J=1 establece el flip-flop cuando se requiere; si K=1 y J=1, el flip-flop está complementado y pasa del estado 0 al estado 1 cuando se requiere. En este caso, la entrada K se marca con una condición no importa para la transición de 0-a-1. Para una transición del estado 1 al estado 0, debe tenerse K=1, ya que la entrada K despeja el flip-flop. Sin embargo la entrada J puede ser 0 o bien 1, ya que J=0 no tiene efecto, J=1 junto con K=1 complementa el flip-flop con una transición resultante del estado 1 al estado 0.

La tabla de excitación para el flip-flop JK ilustra la ventaja de usar este tipo cuando se diseña en circuitos secuenciales. El hecho de que tiene muchas condiciones no importa indica que los

circuitos combinacionales para las funciones de entrada son susceptibles de ser más simples debido a que los términos no importa por lo común simplifican a una función.

5.1.6.3 TABLAS DE EXITACION FLIP - FLOP D.

La tabla de excitación para el flip-flop D se muestra en la Tabla 5.2(c). Mediante la tabla característica, Tabla 5.1(c), se observa que el estado siguiente siempre es igual a la entrada D y es independiente del estado presente. De este modo, D debe ser 0 si Q(t + 1) ha de ser 0, y 1 si Q(t + 1) tiene que ser 1, sin importar el valor de Q(t).

5.1.6.4 TABLAS DE EXITACION FLIP - FLOP T.

La tabla excitación para el flip-flop T se muestra en la Tabla 5.2(d). Mediante la tabla característica, Tabla 5.1(d), se encuentra cuando la entrada T=1, el estado del flip-flop se complementa; cuando T=0, el estado del flip-flop permanece sin cambio. En consecuencia, cuando el estado del flip-flop debe permanecer igual, el requisito es que T=0. Cuando el estado del flip-flop tiene que complementarse, T debe ser igual a 1.

5.1.7 FLIP - FLOP MAESTRO - ESCLAVO.

Un flip-flop maestro-esclavo se construye mediante dos flip-flops separados. Un circuito sirve como un maestro y el otro como un esclavo, y el circuito global se conoce como un flip-flop maestro-esclavo. El diagrama logico de un flip-flop maestro-esclavo RS se muestra en la figura 5.14. Consta de un flip-flop maestro, un flip-flop esclavo y un inversor. Cuando un pulso de reloj CP es 0, la salida del inversor es 1. Ya que la entrada de reloj del esclavo es 1, el flip-flop está habilitado si la salida Q es igual a Y, en tanto que Q' es igual a Y'. El flip-flop maestro se habilita porque CP=0. Cuando el pulso llega a 1, entonces la información en las entradas externas R y S se transmite al flip-flop maestro. Sin embargo, el flip-flop esclavo está aislado mientras el pulso esté en su nivel 1, ya que la salida del inversor es 0. Cuando el pulso regresa a 0, el

flip-flop maestro está aislado, lo cual evita que lo afecten las entradas externas. El flip-flop esclavo pasa entonces al mismo estado que el del flip-flop maestro.

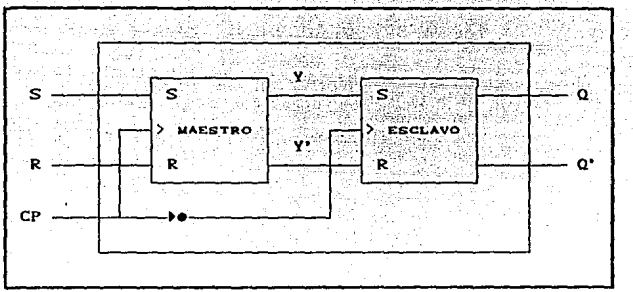


FIG. 5.14 DIAGRAMA LOGICO DEL F-F MAESTRO-ESCLAVO.

Las relaciones de temporizados que se muestran en la figura 5.15 ilustran la secuancia de eventos que ocurren en un flip-flop maestro-esclavo. Se supone que el flip-flop esta en estado despejado antes de la ocurrencia de un pulso, de modo que Y=0 y Q=0. Las condiciones de entrada son S=1, R=0, y el siguiente pulso de reloj cambiara el flip-flop al estado ajustar con Q=1. Mediante la transicion de un pulso de 0 a 1, el flip-flop maestro esta restaurado y cambia Y a 1. El flip-flop esclavo no es aceptado porque su entrada CP es 0. Ya que el flip-flop maestro es un circuito interno, su cambio de estado no es obvio en las salidas Q y Q'. Cuando el pulsor resgresa a 0, se permite que la informacion del maestro pase al esclavo, haciendo que la salida externa sea Q=1. Observese que la entrada externa S debe cambiarse al mismo tiempo que el pulso pasa a traves de su transicion de borde negativo. Esto se debe a que una vez que la entrada CP alcanza 0, el maestro esta inhabilitado y sus entradas R y S no tienen influencia hasta que ocurre el siguiente pulso de reloj. Por eso, en un flip-flop maestro-esclavo es posible cambiar la salida del flip-flop y su informacion de entrada con el mismo pulso de reloj. Debe tomarse en cuenta que la entrada S puede llegar mediante la salida de otro flip-flop maestro-esclavo que se cambio con el mismo pulso de reloj.

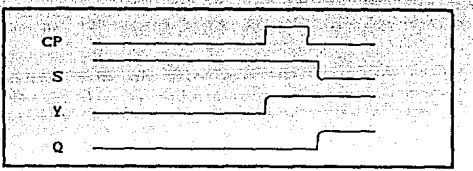


FIG. 5.15 RELACIONES DE TIEMPO EN UN F-F MAESTRO-ESCLAVO.

El comportamiento del flip-flop maestro-esclavo que acaba de describirse dicta que los cambios de estado en todos los flip-flops coincidan con la transicion de borde negativo del pulso. No obstante, algunos flip-flops maestro-esclavo IC cambian los estados de salida en la transicion de borde positivo de los pulsos de reloj. Esto sucede en flip-flops que tienen un inversor adicional entre la CP terminal y la entrada del maestro. Tales flip-flops se disparan con pulsos negativos, , de modo que el borde negativo del pulso afecte al esclavo y las terminales de salida.

La combinación maestro-esclavo puede construirse para cualquier tipo de flip-flop por la adicion de flip-flop RS temporizado con un reloj invertido para formar el esclavo. Un ejemplo de flip-flop JK maestro-esclavo construido con compuertas NAND se muestra en la figura 5.16. Consta de dos flip-flops; las compuertas 1 a la 4 forman el flip-flop maestro, a las compuertas 5 a la 8 forman el flip-flop esclavo. La informacion presente en las entradas J y K se transmite al flip-flop maestro en el borde positivo de un pulso de reloj y se sostiene hasta que ocurre el borde negativo del pulso de reloj, despues del cual permite que pase a través del flip-flop esclavo. La entrada de reloj normalmente es 0, lo cual mantiene las salidas de las compuertas 1 y 2 en el nivel. Esto evita que las entradas J y K afecten el flip-flop maestro. El flip-flop esclavo es un tipo RS temporizado, con el flip-flop sumunistrando las entradas y con la entrada de reloj invertida por la compuerta 9. Cuando el reloj es 0, la salida de la compuerta 9 es 1, de modo que la salida Q es igual a Y y Q' es igual a Y'. Cuando ocurre el borde positivo de un pulso de reloj, el flip-flop maestro se afecta y puede cambiar estados. El flip-flop esclavo está aislado mientras que el reloj esté en el nivel 1, ya que la salida de la compuerta 9 proporciona un 1 a ambas entradas de las compuertas 7 y 8 NAND flip-flop básico. Cuando la entrada de reloj regresa a 0, el

flip-flop maestro está aislado mediante las entradas J y K y el flip-flop esclavo pasa al mismo estado del flip-flop maestro.

Se considera ahora un sistema digital que contiene muchos flip-flops maestro-esclavo, con las salidas de algunos flip-flops que van a las entradas de otros flip-flops. Se supone que las entradas de pulso de reloj a todos los flip-flops están sincronizadas (ocurren al mismo tiempo). Al principio de cada pulso de reloj, algunos de los elementos maestro cambian estado, pero las salidas flip-flop permanecen en sus valores previos. Despues de que el pulso de reloj regresa a 0, algunas de las salidas cambian de estado, pero ninguno de los nuevos estados tiene efecto en cualquiera de los elementos maestro hasta el siguiente pulso de reloj. Así que, los estados de flip-flops en el sistema pueden cambiarse en forma simultánea durante el mismo pulso de reloj, aún cuando las salidas de los flip-flops estén conectadas a entradas de flip-flop. Esto es posible ya que el nuevoestado aparace en las terminales de salida solo despues de que el pulso de reloj ha regresado a 0. En consecuencia, el contenido binario de un flip-flop y el contenido del segundo transferirse al primero, y ambas transferencias pueden ocurrir durante el mismo pulso de reloj.

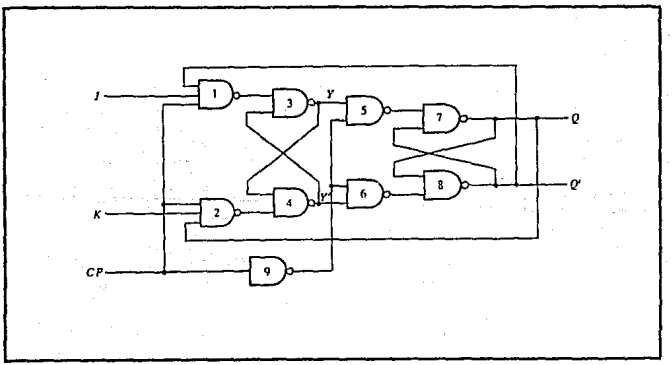


FIG. 5.16 F-F MAESTRO-ESCLAVO JK CON PULSOS DE RELOJ.

5.2 SISTEMAS SECUENCIALES SINCRONOS.

Los circuitos digitales que hasta ahora se han considerado han sido combinacionales, esto es, las salidas en cualquier momento dependen por completo de las entradas presentes en ese tiempo. Aunque cualquier sistema digital es susceptible de tener circuitos combinacionales, la mayoría de los sistemas que se encuentran en la practica también incluyen elemntos de memoria. los cuales requieren que el sistema se describa en términos de lógica secuencial.

Un diagrama de bloques de un circuito secuencial se muestra en la figura 5.17. Consta de un circuito combinacional al que se conectan elementos de memoria para formar una trayectoria de retroalimentecion. Los elementos de memoria son dispositivos capaces de almacenar dentro de ellos información binaria. La información binaria almacenada en los elementos de memoria en cualquier momento dado define el estado del circuito secuencial. El circuito secuencial recibe información binaria de entradas externas. Estas entradas, junto con el estado presente de los elementoa de memoria, determinan en valor binario en las terminales de salida. También determinan las condiciones para cambiar el estado en los elementos de memoria. El diagrama de bloque demuestra que las salidas externas en un circuito secuencial son funciones no solo de las entradas externas sino también del estado presente de los elementos de memoria. El siguiente estado de los elementos de memoria también es una función de las entradas externas y del estado presente. Por tanto, un circuito secuencial está especificado por una secuencia de tiempo de entradas, salidas y estados internos.

Hay dos tipos principales de circuitos secuenciales. Su clasificación depende del temporizado de sus señales. Un circuito secuencial síncrono es un sistema cuyo comportamiento puede definirse por el conocimiento de sus señales en instantes discretos de tiempo. El comportamiento de un circuito secuencial asíncrono depende del orden en el cual cambian sus señales de entrada y puede afectarse en cualquier instante de tiempo. Los elementos de memoria que por lo común se utilizan en los circuitos secuenciales asíncronos son dispositivos de retardo de tiempo. La capacidad de memoria de un dispositivo de retardo de tiempo se debe al hecho de que toma un tiempo finito para que la señal se propague a través del dispositivo. En la práctica, el retardo de propagación interno en las compuertas lógicas es de suficiente duración para producir el retardo necesario, de modo

que pueden ser innecesarias unidades físicas de retardo de tiempo. En los sistemas asíncronos de tipo de compuerta, los elementos de memoria en la figura 5.17 constan de compuertas lógicas cuyos retardos de propagación constituyen la memoria requerida. Por consiguiente, un circuito secuencial asíncrono puede considerarse como un circuito combinacional con retroalimentación. Debido a la retroalimentación entre compuertas lógicas, un circuito secuencial asíncrono a veces puede llegar a ser inestable. El problema de la inestabilidad le impone muchas dificultades al diseñador.

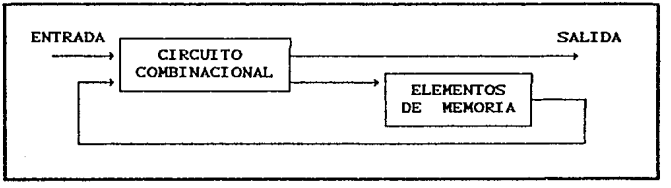


FIG. 5.17 DIAGRAMA DE BLOQUES DE UN CIRCUITO SECUENCIAL.

Un sistema lógico secuencial asíncrono, por definición, debe emplear señales que afecten los elementos de memoria solo en instantes discretos de tiempo. Una forma de lograr este objetivo es usar pulsos de duración limitada a través del sistema, de modo que una amplitud de pulso represente la lógica 1 y otra amplitud del pulso (o la ausencia de un pulso) represente la lógica 0. La dificultad con un sistema de pulsos es que cualesquiera dos pulsos que lleguen de fuentes independientes separadas a las entradas de la misma compuerta exhibirán retardos impredecibles, que separarán los pulsos ligeramente y resultarán en operación poco confiable.

Los sistemas lógicos secuenciales síncronos usan amplitudes fijas, como niveles de voltaje para las señales binarias. La sincronización se logra por un dispositivo temporizador, llamado reloj maestro generador, el cual genera un tren periódico de pulsos de reloj. Los pulsos de reloj se distribuyen a través del sistema en tal forma que los elementos de memoria están afectados sólo por la llegada del pulso de sincronización. En la práctica los pulsos de reloj se aplican a compuertas AND junto con las

señales que especifican el cambio requerido en los elementos de memoria. Las salidas de la compuerta AND pueden transmitir señales sólo a los instantes que coinsiden con la llegada de los pulsos de reloj. Los circuitos secuenciales síncronos que usan pulsos de reloj en las entradas de los elementos de memoria se denominan circuitos secuenciales de reloj. Los circuitos secuenciales de reloj son el tipo que se encuentra con más frecuencia. No manifiestan problemas de inestabilidad y su temporizado se desglosa fácilmente en pasos discretos independientes, cada uno de los cuales se considera por separado.

Los elementos de memoria que se usan en los circuitos secuenciales de reloj se llaman flip-flops. Estos circuitos son celdas binarias capases de almacenar un bit de información. Un circuito flip-flop tiene dos salidas, una para el valor normal y otra para el valor complementario del bit almacenado en él. La información binaria puede entrar a un flip-flop en una gran variedad de formas, hecho que da lugar a diferentes tipos de flip-flops. En la siguiente sección se examinarán los diversos tipos de flip-flops y se definiran sus propiedades lógicas.

5.2.1 TABLAS DE ESTADO.

La secuencia en tiempo de las entradas, salidas y estados de flip-flop pueden enumerarse en una tabla de estado. La tabla de estado para el circuito en la figura 5.18 se muestra en la Tabla 5.3. Consta de tres secciones etiquetadas estado presente, estado siguiente y salida. El estado presente indica los estados de los flip-flops antes de la ocurrencia del pulso de reloj. El estado siguiente muestra los estados de los flip-flops despues de la aplicación de un pulso de reloj, y la sección de salida lista los valores de las variables de salida durante el estado presente. Las secciones de estado siguiente al igual que la salida tienen dos columnas, una para x=0 y la otra para x=1.

La derivación de la tabla de estado principia desde un estado inicial supuesto. El estado inicial de la mayoría de los circuitos secuenciales prácticos se define como el estado con números 0 en todos los flip-flops. Algunos circuitos secuenciales tienen un estado inicial diferente y otros no tienen ninguno en absoluto. En cualquier caso, los análisis siempre pueden principiar desde cualquier estado advitrario. En este ejemplo se principia derivando la tabla de estado desde el estado inicial 00.

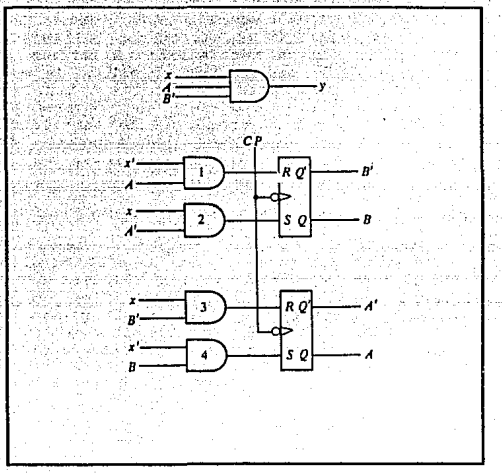


FIG. 5.18 EJEMPLO DE UN CIRCUITO SECUENCIAL CON PULSOS DE RELOJ.

ESTADO PRESENTE	ESTADO SIGUIENTE X=0	X=1	SALIDA X=0	X=1
AB	AB	AB	Y	Y
00	00	01	0	0
01	11	01	0	0
10	10	00	0	1
11	10	11	0	0

TABLA 5.3 TABLA DE ESTADOS PARA EL CIRCUITO DE LA FIG. 5.18.

Cuando el estado presente es 00, A=0 y B=0. Mediante el diagrama lógico, se ve que ambos flip-flops están despejados y x=0. Ninguna de las compuertas AND produce una señal lógica 1. Por lo tanto, el estado siguiente permanece sin cambio. Con AB=00 y x=1, la compuerta 2 produce una señal lógica 1 y la entrada S del flip-flop B y la compuerta 3 produce una señal lógica 1 a la entrada del flip-flop A. Cuando un pulso de reloj dispara los flip-flops, A se despeja y B está ajustado, haciendo que el estado siguiente sea 01. Esta información se lista en el primer renglon de la tabla de estado.

EN forma similar, puede derivarse el siguiente estado principiando desde los otros tres posibles estados presentes. En general, el estado siguiente es una función de las entradas, del estado presente y del tipo de flip-flop que se utilice. Con flip-flops RS, por ejemplo, debe recordarse que un 1 en la entrada S establece el flip-flop y un 1 en la entrada R despeja el flip-flop, sin importar su estado previo. Un 0 en las entradas S y R deja el flip-flop sin cambio, mientras que un tanto en la entrada S como en la R evidencia un mal diseño y una tabla de estado indeterminada.

Las entradas para la sección de salida son fáciles de derivar. En este ejemplo, la salida y es igual a 1 solo cuando X=1, A=1 y B=0. Por tanto, las columnas de salida se marcan con 0, excepto cuando el estado presente es 10 y la entrada x=1, por lo cual y se marca con un 1.

La tabla de estado de cualquier circuito secuencial se obtiene por el mismo procedimiento que se usa en el ejemplo. En general, un circuito secuencial con m flip-flops y n variables de entrada tendrá 2^m renglones, uno para cada estado. Cada una de las secciones de estado siguiente y salida tendrán 2^n columnas, una para cada combinación de entrada.

Las salidas externas de un circuito secuencial pueden tener procedencia de compuertas lógicas o de elementos de memoria. La sección de salida en la tabla de estado es necesaria sólo si hay salidas de compuertas lógicas. Cualquier salida externa que se toma en forma directa de un flip-flop ya esta listada en la columna de estado presente de la tabla de estado. Por lo tanto, la sección de salida de la tabla de estado puede excluirse si no hay salidas externas de compuertas lógicas.

5.2.2 DIAGRAMA DE ESTADO.

La información disponible en una tabla de estado puede representarse en forma gráfica en un diagrama de estado. En este diagrama, un estado se representa con un círculo y la transición entre estados se indica con líneas dirigidas que conectan los círculos. El diagrama de estados de un circuito secuencial en la fig. 5.18 se muestra en la fig. 5.19. El numero binario dentro de cada círculo identifica el estado que representa el círculo. Las líneas dirigidas están etiquetadas con dos números binarios separados por una /. El valor de entrada que provoca la transición de estado se etiqueta primero; el numero despues del símbolo / da el valor de la salida durante el estado presente. Por ejemplo, la línea dirigida desde el estado 00 al 01 se etiqueta 1/0, lo cual significa que el circuito secuencial está en un estado presente 00 mientras X=1 y Y=0, y que a la terminación del siguiente pulso de reloj, el circuito pasa al siguiente estado 01. Una línea dirigida que conecta un círculo con sí misma indica que no ocurre cambio de estado. El diagrama de estado proporciona la misma información que la tabla de estado y se obtiene en forma directa de la tabla 5.3.

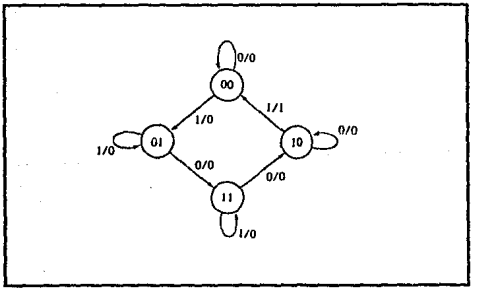


FIG. 5.19 DIAGRAMA DE ESTADO PARA EL CIRCUITO DE LA FIGURA 5.18.

No hay diferencia entre una tabla de estado y un diagrama de estado excepto en la forma de representación. La tabla de estado es mas fácil de derivar mediante un diagrama lógico dado y en el

diagrama de estado continúa en forma directa de una tabla de estado. El diagrama de estado da una imagen de las transiciones de estado y se encuentra en una forma adecuada para la interpretacion humana de la operacion del circuito. Un diagrama de estado se utiliza con frecuencia como la especificacion inicial de diseño de un circuito secuencial.

5.2.3 TECNICAS DE REDUCCION.

El análisis de los circuitos secuenciales principia madiante un diagrama de circuito y culmina enn una tabla de estado o diagrama. El diseño de un circuito secuencial se inicia mediante un conjunto de especificaciones y termina en un diagrama lógico. En esta seccion se exponen ciertas propiedades de los circuitos secuenciales que pueden usarse para reducir el número de compuertas y de flip-flops durante el diseño.

5.2.3.1 REDUCCION DE ESTADO.

En cualquier proceso de diseño debe considerarse el problema de minimizar el costo del circuito final. Las dos reducciones de el costo más obvias son las reducciones en el número de flip-flops y el número de compuertas. Debido a que estos dos detalles parecen los mas evidentes, se han estudiado e investigado extensamente. De hecho, una gran parte del tema de la teoría de conmutación se dedica a la búsqueda de algoritmos para minimizar el número de flip-flops y compuertas en los circuitos secuenciales.

La reducción del número de flip-flops en un circuito secuencial se conoce como el problema de reducción de estado. Los algoritmos de reducción de estado tratan con procedimientos para reducir el número de estados en una tabla de estado mientras se mantienen sin cambio los requisitos de entrada-salida externa. Ya que m flip-flops producen 2^m estados, una reducción en el número de estados puede (o no puede) dar por resultado una reducción en el número de flip-flops. Un efecto no predecible al reducir el número de flip-flops es que algunas veces el circuito equivalente (con menos flip-flops) puede requerir más compuertas combinacionales.

Se ilustrará la necesidad de la reducción de estado con un ejemplo. Se principia con un circuito secuencial cuya especificación esta dada en el diagrama de estado en la figura 5.20. En este ejemplo, solo son importantes las secuencias de entrada-salida; los estados internos solo se usan para proporcionar las secuencias requeridas. Por esta razon, los estados que se marcan dentro de los círculos se denotan por símbolos alfabéticos en lugar de sus valores binarios. Esto es en contraste a un contador binario, donde la secuencia de valores binarios de los estados por sí mismos se toman como las salidas.

Hay un número infinito de secuencias de entrada que pueden aplicarse al circuito, cada una conduce a una secuencia única de salidas. Como ejemplo, considérese la secuencia de entrada 01010110100 principiando desde el estado inicial A. Cada entrada de 0 o 1 produce una salida de 0 o 1 y causa que el circuito pase al estado siguiente. Mediante el diagrama de estado, se obtiene la secuencia de salida y estado para la secuencia dada de entrada como sigue: en el circuito en el estado inicial A, una entrada de 0 produce una salida de 0 y el circuito permanece en el estado a. Con el estado presente a y entrada de 1, la salida es 0 y el estado siguiente es b. Con el estado presente b y la entrada de 0, la salida es 0 y el estado siguiente es b. Continuando este proceso se encuentra la secuencia completa como sigue:

estado	a	a	b	c	d	e	f	f	g	f	g	a
entrada	0	1	0	1	0	1	1	0	1	0	0	
salida	0	0	0	0	0	1	1	0	1	0	0	

En cada columna, se tiene el estado presente, el valor de entada y el valor de salida. El estado siguiente se escribe en la parte superior de la siguiente columna. Es importante tomar en cuenta que en este circuito los estados por sí mismos son de importancia secundaria, ya que se tiene interés solo en las secuencias de salida que provocan las secuencias de entrada.

Ahora se supone que ha encontrado un circuito secuencial cuyo diagrama de estado tiene menos de siete estados y se desea compararlos con el circuito cuyo diagrama de estado está dado en la figura 5.20. Si se aplican secuencias de entrada idénticas para todas las secuencias de entrada, entonces se dice que los doscircuitos son equivalentes (en lo que respecta a la entrada-salida) y uno puede reemplazarse por el otro. El problema de reduccion de estado es encontrar formas de reducir el numero

de estados en un circuito secuencial y alterar las relaciones de entrada-salida.

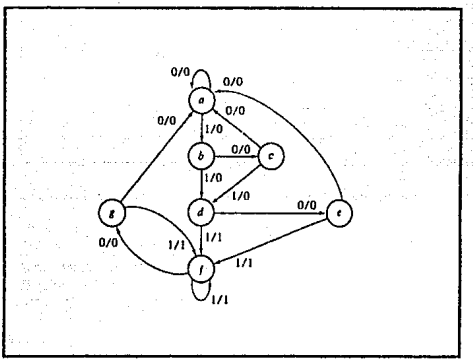


FIG. 5.20 DIAGRAMA DE ESTADO.

Ahora se procede a reducir el número de estados para este ejemplo. Primero, se necesita la tabla de estado; es más conveniente aplicar los procedimientos para reducción de estados aquí que en los diagramas de estado. La tabla de estado del circuito se lista en la Tabla 5.4 y se obtienen en forma directa mediante el diagrama de estado en la figura 5.20.

Aquí se presenta, sin prueba, un algoritmo para la reducción de estado de una tabla de estado por completo especificada: "Se dice que dos estados son equivalentes si, para cada miembro del conjunto de entradas, dan exactamente la misma salida y envían al circuito ya sea al mismo estado o a un estado equivalente. Cuando dos estados son equivalentes, uno de ellos puede eliminerse sin alterar las relaciones de entrada-salida".

Se aplica este algoritmo a la Tabla 5.4. Al pasar a través de la tabla de estado, se buscan dos estados presentes que vayan al mismo estado siguiente que tengan la misma salida para ambas

ESTADO PRESENTE	ESTADO SIGUIENTE X=0	X=1	SALIDA X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

TABLA 5.4 TABLA DE ESTADOS.

combinaciones de entrada. Los estados g y e son dos de dichos estados; ambos van a los estados a y f y tienen salidas de 0 y 1 para x=0 y x=1, respectivamente. Por eso, los estados g y e son equivalentes; puede eliminarse uno. El procedimiento de eliminar un estado y reemplazarlo por su equivalente se demuestra en la Tabla 5.5. El renglon con el estado presente g se cruza y el estado presente g se cruza y el estado g se reemplaza por el estado e cada vez que ocurre en las columnas de estados siguientes.

ESTADO PRESENTE	ESTADO SIGUIENTE X=0	X=1	SALIDA X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	~~f~~/d	0	1
e	a	~~f~~/d	0	1
~~f~~/	~~g~~/e	f	0	1
~~g~~/	a	f	0	1

TABLA 5.5 REDUCCION DE LA TABLA DE ESTADOS.

El estado presente f ahora tiene estados siguientes e y f y salidas 0 y 1 para x=0 y x=1, respectivamente. Los mismos estados siguientes y salidas aparecen en el renglón con el estado presente d. Por lo tanto, los estados f y g son equivalentes. El estado f puede eliminarse y reemplazarse por d. La tabla repetida final se muestra en la Tabla 5.6. El diagrama de estado para la tabla reducida consta solo de cinco estados y se muestra en la figura 5.21. Este diagrama de estado satisface las especificaciones originales de entrada-salida y producirá la secuencia referida de salida para cualquier secuencia dada de entrada. La siguiente lista que se deriva mediante el diagrama de estado en la figura 5.21 es para la secuencia de entrada que se utilizó con anterioridad. Se observa que resulta la misma secuencia de salida aunque la secuencia de estado sea diferente:

estado	a	a	b	c	d	e	d	d	e	d	e	a
entrada	0	1	0	1	0	1	1	0	1	0	0	
salida	0	0	0	0	0	1	1	0	1	0	0	

De hecho, esta secuencia es exactamente la misma que se obtuvo de la figura 5.20, y se reemplaza e por g y d por f.

ESTADO PRESENTE	ESTADO SIGUIENTE X=0	ESTADO SIGUIENTE X=1	SALIDA X=0	SALIDA X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

TABLA 5.6 TABLA REDUCIDA DE ESTADOS.

Vale la pena observar que la reducción en el número de estados de un circuito secuencial es posible si se tiene sólo interes en las relaciones externas de salida-entrada. Cuando se toman en forma directa salidas externas de los flip-flop, las salidas deben ser independientes del número de estados antes de que se apliquen algoritmos de reduccion de estado.

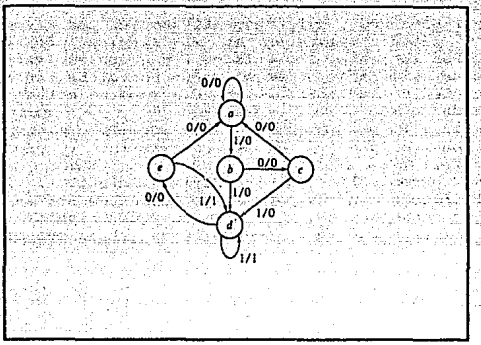


FIG. 5.21 DIAGRAMA REDUCIDO DE ESTADO.

El circuito secuencial de este ejemplo se redujo de siete a cinco estados. En cualquier caso, la representacion de los estados con componentes físicos requiere que se usen tres flip-flops, ya que m flip-flops pueden representar hasta 2^m estados distintos. Con tres flip-flops, pueden formularse hasta ocho estados binarios denotados por numeros binarios 000 hasta 111, con cada bit designando el estado de un flip-flop. Si se utiliza la tabla de estado en la Tabla 5.4, deben asignarse valores binarios a siete estados; el estado restante no se usa. Si se utiiiza la tabla de estado en la Tabla 5.6, solo cinco estados necesitan asignacion binaria, y quedan tres estados sin uso. Los estados sin uso se tratan como condiciones no importa durante el diseño del circuito. Ya que las condiciones no importa por lo comun ayudan a obtener funciones booleanas mas simples, es más probable que el circuito con cinco estados requiera menos compuertas combinacionales que el circuito con siete estados. En cualquier caso, la reduccion de siete a cinco estados no reduce el numero de flip-flops. En general, la reduccion del numero de compuertas en una tabla de estado es probable que resulte en un circuito con menos equipo. Sin embargo, el hecho de que una tabla de estado se ha reducido a menos estados no garantiza un ahorro en el numero de flip-flops o de compuertas.

5.2.3.2 ASIGNACION DE ESTADO.

El costo de un circuito combinacional parte de un circuito secuencial puede reducirse por el uso de metodos conocidos de simplificación para los circuitos combinacionales. Sin embargo, hay otro factor, conocido como el problema de asignacion de estado, que entra en juego al minimizar las compuertas combinacionales. Los procedimientos de asignación de estado se ocupan con métodos para asignar valores binarios a los estados, en tal forma que reducen el costo de un circuito combinacional que impulsa a los flip-flops. Esto es de particular ayuda cuando se considera un circuito secuencial desde sus terminales externas de entrada-salida. Tal circuito puede seguir una secuencia de estados internos, pero los valores binarios de los estados individuales puede no ser de secuencia mientras el circuito produzca la secuencia referida de salida para una secuencia dada de entradas. Esto no se aplica a circuitos cuyas salidas externas se aplican de manera directa mediante flip-flops con secuencias binarias especificadas por completo.

Las alternativas disponibles de asignación al estado binario pueden demostrarse junto con el circuito secuencial que se especifica en la Tabla 5.6. Recuerdese que, en este ejemplo, los valores de los estados son inmateriales mientras que su secuencia mantenga las relaciones apropiadas de entrada-salida. Por esta razon, cualquier asignación de número binario es satisfactoria en tanto que cada estado este asignado a un numero único. En la Tabla 5.7 se muestran tres ejemplos de asignaciones binarias posibles para los cinco estados de la tabla reducida. La asignacion 1 es una asignacion binaria directa para la secuencia de estados desde a hasta e. Las otras dos asignaciones se eligen en forma arbitraria. De hecho, hay 140 diferentes asignaciones distintas para este circuito.

La Tabla 5.8 es la tabla de estado reducida con asignación binaria 1 sustituida por los simbolos de letra de los cinco estados. Es obvio que una asignacion binaria diferente causara una tabla de estado con distintos valores binarios para los estados, en tanto las relaciones de entrada-salida permanezcan iguales. La forma binaria de la tabla de estado se utiliza para derivar el circuito combinacional parte del circuito secuencial. La complejidad del circuito combinacional depende de la asignacion binaria de estado que se escoja.

Se han sugerido diversos procedimientos que conducen a una

ESTADO PRESENTE	ASIGNACION 1	ASIGNACION 2	ASIGNACION 3
a	001	000	000
b	010	010	100
c	011	011	010
d	100	101	101
e	101	111	011

TABLA 5.7 ASIGNACION DE TRES ESTADOS POSIBLES.

asignación binaria particular de las muchas disponibles. El criterio mas común es que la asignación que se escoja debe producir un circuito combinacional simple para las entradas flip-flop. Sin embargo, a la fecha, no hay procedimientos de asignacion de estado que garanticen un circuito combinacional de mínimo costo. La asignacion de estado es uno de los problemas de reto de la teoría de conmutación. El lector interesado encontrará una rica y cada vez mas abundante literatura sobre este tema.

ESTADO PRESENTE	ESTADO SIGUIENTE X=0	ESTADO SIGUIENTE X=1	SALIDA X=0	SALIDA X=1
001	001	010	0	0
010	011	100	0	0
011	001	100	0	0
100	101	100	0	1
101	001	100	0	1

TABLA 5.8 TABLA REDUCIDA DE ESTADOS CON LA ASIGNACION BINARIA 1.

5.2.4 DISENO FORMAL

El diseño de un circuito secuencial temporizado principia mediante un conjunto de especificaciones y culmina en un diagrama lógico o en una lista de funciones booleanas, mediante las cuales puede obtenerse el diagrama lógico. En contraste con un circuito combinacional, que está especificado por completo por una tabla de verdad, un circuito secuencial requiere una tabla de estado para su especificación. El primer paso en el diseño de circuitos secuenciales es obtener una tabla de estado o una reprentación equivalente, como por ejemplo un diagrama de estado o ecuaciones de estado.

Un circuito secuencial síncrono está hecho de flip-flops y compuertas combinacionales. El diseño del circuito consiste en escoger los flip-flops y entonces encontrar una estructura de compuertas combinacionales que, junto con los flip-flops, producen un circuito que cumple con las especificaciones establecidas. El número de flip-flops se determina mediante el número de estados necesarios en el circuito. El circuito combinacional se deriva mediante la tabla de estado por métodos que se presentan en este capítulo. De hecho, una vez que se determina el tipo y número de flip-flops, el proceso de diseño implica una transformación del problema del circuito secuencial en un problema de circuito combinacional. En esta forma, pueden aplicarse las técnicas del circuito combinacional.

En esta sección se presenta un procedimiento para el diseño de circuitos secuenciales. Aunque se intenta que sirva como una guía para el principiante, este procedimiento puede reducirse con la experiencia. El procedimiento primero se resume en una lista de pasos consecutivos que se recomiendan como sigue:

1. La descripción verbal del comportamiento del circuito se establece. Esta descripción puede ir acompañada con un diagrama de estado, un diagrama de temporizado o bien otra información pertinente.

2. A partir de la información dada sobre el circuito, se obtiene la tabla de estado.

3. El número de estados puede reducirse por los métodos de

reducción de estado y el circuito secuencial puede caracterizarse por las relaciones entrada-salida independiente del numero de estados.

4. Se asignan valores binarios a cada estado obtenida en los pasos 2 o 3 contiene símbolos alfabeticas.

5. Se determina el número de flip-flops necesarios y se asigna un símbolo alfabetico a cada uno.

6. Se escoge el tipo de flip-flop que va a usarse.

7. Mediante la tabla de estado se derivan las tablas de excitación y salida del circuito.

8. Por el uso del método de mapa o cualquier otro método de simplificación, se derivan las funciones de salida y las de entrada del flip-flop.

9. Se dibuja el diagrama lógico.

La especificación verbal del comportamiento del circuito por lo comun supone que el lector esta familiarizado con la terminología de la logica digital. Es necesario que el diseñador utilice su intuicion y experiencia para llegar a la interpretación correcta de las especificaciones del circuito, debido a que las descripciones verbales pueden ser incompletas o inexactas. Sin embargo, una vez que se ha establecido dicha especificación y se ha obtenido la tabla de estado, es posible hacer uso del procedimiento formal para diseñar el circuito.

La redución del número de estados y la asignación de valores binarios a los estados se expuso en la sección anterior. En los ejemplos que siguen se supone que se conocen el numero de estado y la asignación binaria para los estados. Como consecuencia, los pasos 3 y 4 del diseño no se consideraran en las exposiciones subsecuentes.

Ya se ha mencionado que el número del flip-flops está

determinado por el número de estados. Un circuito puede tener estados binarios sin usar, si el numero total de estados es menor que 2^n. Los estados que no se usan se toman como condiciones no importa durante el diseño del circuito combinacional parte del circuito.

El tipo de flip-flop que va a usarse pueden incluirse en las especificaciones de diseño o puede depender de lo que este disponible para el diseñador. Muchos sistemas digitales se construyen por completo con flip- flops JK porque son el tipo mas versátil disponible. Cuando están disponibles muchos tipos de flip-flops, es aconsejable usar el flip-flop RS o D para aplicaciones que requieren transferencia de datos (como registadores con corrimiento), el tipo T para aplicaciones que aplican complementación (como contadores binarios) y el tipo JK para aplicaciones generales.

La información de la salida externa se especifica en la sección de salida en la tabla de estado. Mediante ella pueden derivarse las funciones de salida del circuito. La tabla de exitacion para el circuito es similar a la de los flip-flops individuales, excepto que las condiciones de entrada estan listadas por la información disponible en las columnas de estado presente y estado siguiente de la tabla de estado. El método para obtener la tabla de exitación y las funciones simplificadas de entrada flip-flops se ilustran mejor con un ejemplo.

Se desea diseñar el circuito secuencial temporizado cuyo diagrama de estado esta dado en la figura 5.22. El tipo de flip-flop que va a usarse es JK.

El diagrama de estado consta de cuatro estados con valores binarios ya asignados. Ya que las líneas dirigidas están marcadas con un solo dígito binario sin una /, se concluye que hay una variable de entrada y no hay variables de salida. (El estado de los flip-flops puede considerarse como las salidas del circuito). Los dos flip-flops necesarios para representar los cuatro estados se denotan A y B. La variable de estado se denota X.

La tabla de estado para este circuito, derivada mediante el diagrama de estado, se muestra en la Tabla 5.9. Obsérvese que no hay seccion de salida para este circuito. Ahora se mostrara el procedimiento para obtener la tabla de excitacion y la estructura

de compuertas combinacionales.

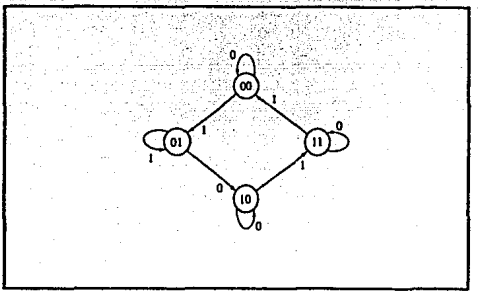


FIG. 5.22 DIAGRAMA DE ESTADO.

La derivación de la tabla de excitación se facilita si se ordena la tabla de estado en una forma diferente. Esta forma se muestra en la Tabla 5.10, donde el estado presente y las variables de entrada están ordenadas en la forma de una tabla de verdad. El valor del estado siguiente para cada estado presente y las condiciones de entrada se copian de la Tabla 5.9. La tabla de

ESTADO PRESENTE		ESTADO SIGUIENTE X=0		X=1	
A	B	A	B	A	B
0	0	0	0	0	1
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	1	0	0

TABLA 5.9 TABLA DE ESTADO.

excitación de un circuito es una lista de las condiciones de

entrada flip-flop que causarán las transiciones requeridas de estado y es una funcion del tipo de flip-flop que se utilice. Ya que este ejemplo especifica flip-flops JK, se necesitan columnas para las entradas J y K de los flip-flops A (denotadas por JA y KA) y B (denotadas por JB y KB).

ENTRADAS DEL CIRCUITO COMBINACIONAL			ESTADO SIGUIENTE		SALIDAS DEL CIRCUITO COMBINACIONAL			
ESTADO PRESENTE		ENTRADA			ENTRADAS FLIP-FLOP			
A	B	X	A	B	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

TABLA 5.10 TABLA DE EXCITACION.

La tabla de excitación para el flip-flop JK se derivó en la Tabla 5.2(b). Esta tabla se usa ahora para derivar la tabla de excitación del circuito . Por ejemplo, en el primer renglon de la Tabla 5.10 se tiene una transicion para el flip-flop A desde 0 en el estado presente hasta 0 en el estado siguiente. En la Tabla 5.2(b) se encuentra que una transicion de estado desde 0 hasta 0 requiere que la entrada J=0 y la entrada K=X, de modo que 0 y X se copian del primer renglon bajo JA y KA, respectivamente. Ya que el primer renglón tambien muestra una transicion para el flip-flop B desde 0 en el estado presente hasta 0 en el estado siguiente, 0 y X se copian en el primer renglón bajo JB y KB. El segundo renglon de la Tabla 5.10 muestra una transicion para el flip-flop B desde 0 en el estado presente hasta 1 en el estado siguiente. Mediante la Tabla 5.2(b) se encuentra que una transición desde 0 hasta 1 requiere que la entrada J=1 y la entrada K=X. De modo que 1 y X se copian en el segundo renglon bajo JB y KB, respectivamente. Este proceso se continua para

cada renglón de la tabla y para cada flip-flop, con las condiciones de entrada como se especifican en la Tabla 5.2(b) que se copian en el renglón apropiado del flip-flop particular que se este considerado.

Se hace ahora una pausa y se considera la información disponible en una tabla de excitación como la Tabla 5.10. Se sabe que un circuito secuencial consta de un número de flip-flops y un circuito combinacional. En la figura 5.23 se muestran los dos flip-flops JK necesarios para el circuito y una caja que representa el circuito combinacional. Mediante el diagrama de bloques, es claro que las salidas del circuito combinacional van a las entradas flip-flop y las salidas externas (si se especifica). Las entradas al circuito combinacional son las entradas externas y los valores de estado presentes de los flip-flops. Además, las funciones booleanas que especifican un

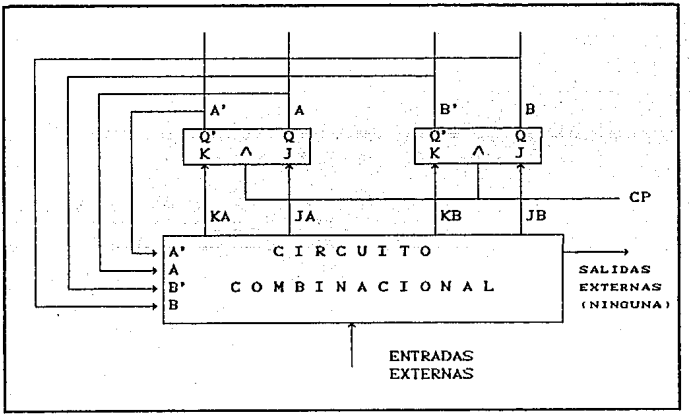


FIG. 5.23 DIAGRAMA DE BLOQUES DEL CIRCUITO SECUENCIAL.

circuito combinacional se derivan mediante una tabla de verdad que muestra las relaciones de entrada-salida del circuito. La

tabla de verdad que describe al circuito combinacional está disponible en la tabla de excitación. Las entradas al circuito combinacional se especifican bajo las columnas de estado presente y entrada, y las salidas del circuito combinacional se especifican bajo las columnas de entrada flip-flop. Por lo tanto, una tabla de excitación transforma un diagrama de estado en la tabla de verdad necesaria para el diseño del circuito combinacional parte del circuito secuencial.

Las funciones booleanas simplificadas para el circuito combinacional pueden derivarse ahora. Las entradas son las variables A, B y x; las salidas son las variables JA, KA, JB y KB. La información de la tabla de verdad se transfiere a los mapas en la figura 5.24, donde se derivan las cuatro funciones simplificadas de entrada flip-flop:

$$JA = Bx' \qquad KA = Bx$$

$$JB = x \qquad KB = A \odot x$$

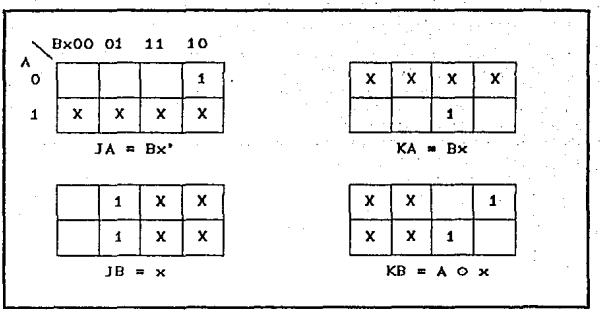


FIG. 5.24 MAPAS PARA EL CIRCUITO COMBINACIONAL.

El diagrama lógico se dibuja en la dibuja en la figura 5.25 y consta de dos flip-flop, dos compuertas AND, una compuerta de equivalencia y un inversor.

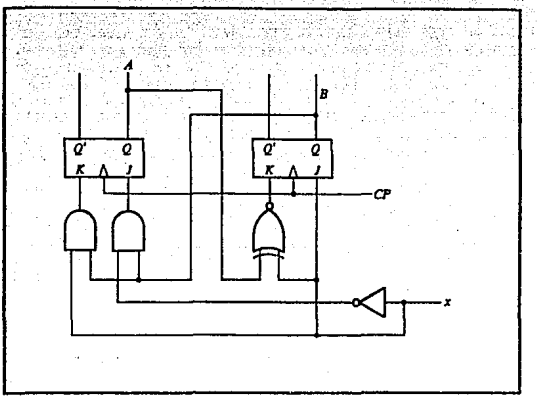


FIG. 5.25 DIAGRAMA LOGICO DEL CIRCUITO SECUENCIAL.

Con alguna experiencia, es posible reducir la cantidad de trabajo implicado en el diseño del circuito combinacional. Por ejemplo, es posible obtener la información de los mapas en la figura 5.24 en forma directa en la Tabla 5.9, sin tener que derivar la Tabla 5.10. Esto se hace por el paso sistemático a través de cada estado presente y combinación de entrada en la Tabla 5.9 y comparando con los valores binarios del estado siguiente correspondiente. Entonces, se determinan las condiciones requeridas de entrada como las especifica la excitación flip-flop en la Tabla 5.2. En lugar de insertar los 0, 1 o X obtenidos dentro de la tabla de excitación, pueden escribirse en forma directa en el cuadro adecuado del mapa apropiado.

La tabla de excitación de un circuito secuencial con m flip-flops, k entradas por flip-flop n impulsos externos está formada por m + n columnas para el estado presente y variables de entrada hasta 2^{m+n} renglones listados en alguna cuenta binaria conveniente. La sección de estado siguiente tiene m columnas, para cada flip-flop se listan en mk columnas, una para cada entrada de cada flip-flop. Si el circuito contiene j salidas, la

tabla debe incluir j columnas. La tabla de verdad del circuito combinacional se toma de la tabla de excitación considerando los m + n estados presentes y columnas de entrada como entradas y los mk + j valores de entrada flip-flop y las salidas externas como salidas.

5.2.4.1 DISEÑO CON ESTADOS SIN USO.

Un circuito con m flip-flops puede tener 2^m estados. Hay ocaciones en las que un circuito secuencial puede utilizar menos de este numero maximo de estado. Los estados que no se usan en la especificacion del circuito secuencial no se listan en la tabla de estado. Cuando se simplifican las funciones de entrada a los flip-flops, los estados sin uso pueden tratarse como condiciones no importa.

Ejemplo : *Complete el diseño del circuito secuencial que se presenta en la sección anterior. Use la tabla de estado reducida con asignacion 1 como se da en la Tabla 5.8. El circuito empleara flip-flops RS.*

Solucion : *La tabla de estados de la Tabla 5.8 vuelve a dibujarse en la Tabla 5.11 en la forma conveniente para obtener la tabla de excitación. Las condiciones de entrada flip-flop se derivan mediante las columnas de estado presente y estado siguiente de la tabla de estado. Ya que se usan flip-flops RS, se necesita consultar la Tabla 5.2(a) para las condiciones de excitación de este tipo de flip-flops. Los tres flip-flops reciben nombres variables A, B, y C. La variable de entrada es x y la variable de salida es y. La tabla de excitación del circuito proporciona toda la información necesaria para el diseño.*

Hay tres estados sin usar de este circuito: estados binarios 000, 110 y 111. Cuando se incluye una entrada de 0 o 1 con estos estados no usados, se obtienen seis

minitérminos no importa 0, 1, 12, 13, 14 y 15. Estas seis combinaciones binarias no se listan en la tabla bajo estado presente y entrada y se tratan como términos no importa.

El circuito combinacional parte del circuito secuencial se simplifica en los mapas de la figura 5.26. Hay siete mapas en el diagrama: seis mapas son para simplificar las funciones de entrada para los tres flip-flops RS. El septimo mapa es para simplificar la salida y. Cada mapa tiene seis X en los cuadros de los minitérminos no importa 0, 1, 2, 13, 14 y 15. Los otros términos no importa en el mapa provienen de las X en las columnas de entrada flip-flop de la tabla. Las funciones simplificadas se listan bajo cada mapa. El diagrama logico obtenido mediante esas funciones booleanas se dibuja en la figura 5.27.

ESTADO PRESENTE			ENTRADA	ESTADO SIGUIENTE			ENTRADAS FLIP-FLOP						SALIDA
A	B	C	X	A	B	C	SA	RA	SB	RB	SC	RC	Y
0	0	1	0	0	0	1	0	X	0	X	X	0	0
0	0	1	1	0	1	0	0	X	1	0	0	1	0
0	1	0	0	0	1	1	0	X	X	0	1	0	0
0	1	0	1	1	0	0	1	0	0	1	0	X	0
0	1	1	0	0	0	1	0	X	0	1	X	0	0
0	1	1	1	1	0	0	1	0	0	1	0	1	0
1	0	0	0	1	0	1	X	0	0	X	1	0	0
1	0	0	1	1	0	0	X	0	0	X	0	X	1
1	0	1	0	0	0	1	0	1	0	X	X	0	0
1	0	1	1	1	0	0	X	0	0	X	0	1	1

TABLA 5.11 TABLA DE EXCITACION.

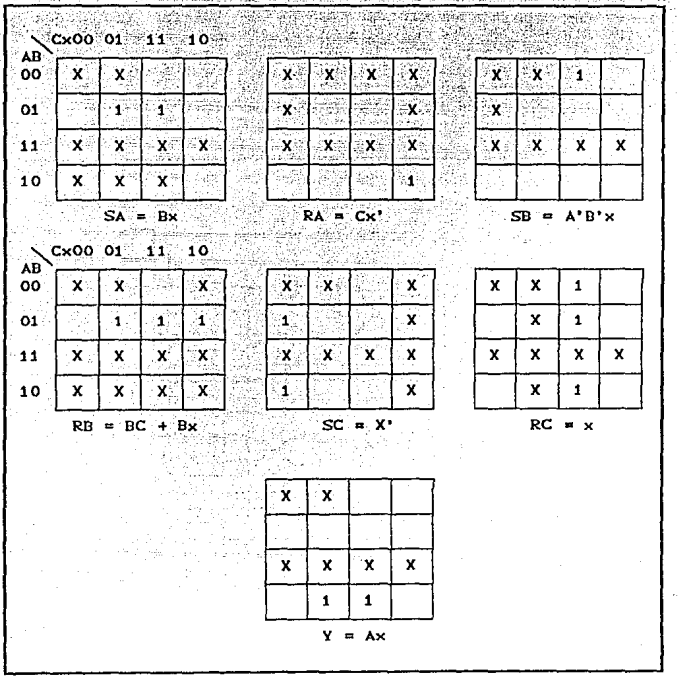


FIG. 5.26 MAPAS PARA SIMPLIFICAR EL CIRCUITO SECUENCIAL.

Un factor que hasta este punto no se ha considerado en el diseño es el estado inicial de un circuito secuencial. Cuando se conecta primero la potencia en un circuito digital, no se conoce en que estado se asentaran los flip-flops. Es costumbre proporcionar una entrada maestra de restaurar cuyo proposito es inicilliazar los estados de flip-flops en el sistema. En forma

típica, la señal maestra de restaurar se aplica a todos los flip-flops en forma síncrona antes de iniciar las operaciones temporizadas. En la mayoría de los casos los flip-flops se despejan en 0 por la señal maestra de restaurar pero algunos pueden ajustarse en 1. Por ejemplo, el circuito de la figura 5.27 puede restaurarse en forma inicial en un estado ABC = 001, ya que el estado 000 no es un estado válido para este circuito.

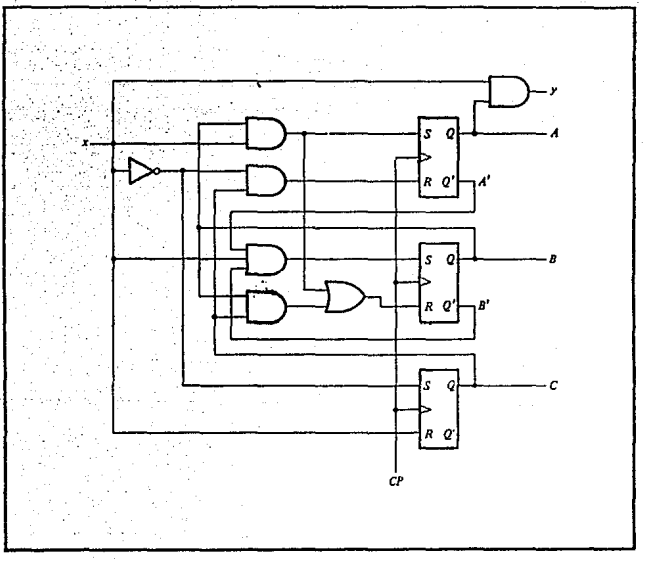


FIG. 5.27 DIAGRAMA LOGICO.

Pero qué pasa si un circuito no se restaura a un estado inicial valido? O peor aun, qué sucede si debido a una señal de ruido o cualquier otra razon no prevista el circuito mismo se encuentra en uno de sus estados invalidos? En este caso es

necesario asegurar que el circuito en forma eventual pasará a uno de los estados válidos de modo que pueda reasumir la operación normal. En otra forma, si el circuito secuencial circula entre estados inválidos, no habrá forma de devolverlo a su secuencia intentada de transiciones de estado. Aún cuando puede suponerse que esta condición indeseable no se supone que ocurra, un diseñador cuidadoso debe asegurarse de que esta situación no ocurra jamás.

Se estableció previamente que los estados sin usar en un circuito secuencial pueden tratarse como condiciones no importa. Una vez que se diseña el circuito, los m flip-flops en el sistema pueden estar en cualquiera de los 2^m estados posibles. Si alguno de los estados se toma como condiciones no importa, debe investigarse el circuito para determinar el efecto de estos estados sin usar. El siguiente estado de los estados inválidos puede determinarse mediante el análisis del circuito. En cualquier caso, siempre es prudente analizar un circuito obtenido mediante un diseño para asegurar que no se incurrió en errores durante el proceso de diseño.

Ejemplo : *Analice el circuito secuencial que se obtuvo en el Ejemplo 5.1 y determine el efecto de los estados sin usar.*

Solucion : *Los estados sin uso son 000, 110 y 111. Los mapas en la figura 5.26 puede ayudar en el analisis. Lo que se necesita aquí es principiar con el diagrama del circuito en la figura 5.27 y derivar la tabla de estado es idéntica a la Tabla 5.6(o a la tabla de estados que es parte de la Tabla 5.11), entonces se sabe que el diseño es correcto. Además, deben determinarse los estados siguientes mediante los estados sin uso 000, 110 y 111.*

Los mapas en la figura 5.26 pueden ayudar a encontrar el estado siguiente mediante cada uno de los estados sin uso. Tome, por ejemplo, el estado sin uso 000. Si el circuito, por alguna razón, está en el estado presente 000,

una entrada x = 0 transferirá el circuito a algún estado siguiente y una entrada x = 1 lo transferirá a otro (o el mismo) estado siguiente. Se investigará primero el mintérmino ABCx = 0000. Mediante los mapas, se ve que este mintérmino no está incluido en alguna función excepto para SC, esto es, la entrada establecida del flip-flop C. Por tanto, los flip-flops A y B no cambiarán pero el flip-flop C se establecerá en 1. Ya que el estado presente es ABC = 000, el estado siguiente será ABC = 001. Los mapas también muestran que el mintérmino ABCx = 0001 está incluido en las funciones para SB y RC. De este modo, B se establecerá y C se despejará. Al principiar con ABC = 000 y establecer B, se obtiene el estado siguiente ABC = 010 (C está despejada). La investigación en el mapa para la salida y muestra que ésta será 0 para estos dos mintérminos.

En el diagrama de estado en la figura 5.28 se muestra el resultado del procedimiento de análisis. El circuito opera como debe, mientras permanezca dentro de los estados 001, 010, 011, 100 y 101. Aún si se encuentra en uno de los estados inválidos 000, 110 o 111, pasa a uno de los estados válidos dentro de uno o dos pulsos de reloj. Así que, el circuito arranca y se corrige por sí mismo, ya que con el tiempo pasa a un estado válido desde el cual continúa operando como se requiere.

Una situación indeseable puede haber ocurrido si el estado siguiente de 110 para x = 1 resulta ser 111 y el siguiente estado de 111 para x = 0 o 1 resulta que es 110. Entonces, si el circuito principia desde 110 o 111, circulará y permanecerá entre esos dos estados para siempre. Deben evitarse los estados sin uso que provocan tal comportamiento indeseable; si se encuentran que existen, el circuito debe rediseñarse. Esto puede hacerse con mayor facilidad al especificar un estado siguiente válido para cualquier estado sin uso que se encuentre que circule entre los estados inválidos.

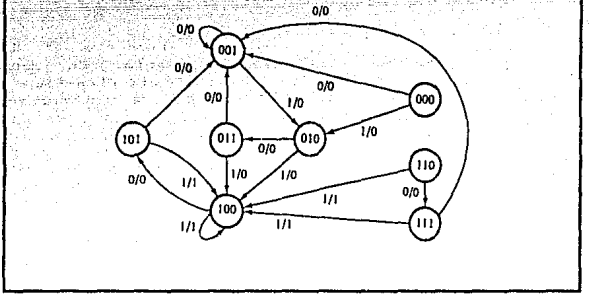


FIG. 5.28 DIAGRAMA DE ESTADOS PARA EL CIRCUITO DE LA FIGURA 5.27.

5.2.5 REGISTRO DE CORRIMIENTO.

Un circuito secuencial temporizado consta de un grupo de flip-flops y compuertas combinacionales conectadas para formar una trayectoria de retroalimentación. Los flip-flops son esenciales porque, cuando estan ausentes, el circuito se reduce a un circuito combinacional puro (siempre que no haya trayectoria de retroalimentación). Un circuito solo con flip-flops se considera un circuito secuencial incluso cuando estan ausentes las compuertas combinacionales.

Un circuito MSI que contiene celdas de almacenamiento en su interior es, por definicion un circuito secuencial. Los circuitos MSI que incluyen flip-flops u otras celdas de almacenamiento por lo comun se clasifican segun la funcion que realizan más que por

el nombre "circuito secuencial". Estos circuitos MSI se clasifican en una de tres categorías: registros, contadores o memorias de acceso aleatorio. En este capítulo se presentan varios registros y contadores disponibles en la forma IC y se explica su operación. También se presenta la organización de la memoria de acceso aleatorio.

Un registro es un grupo de celdas de almacenamiento binario adecuadas para mantener información binaria. Un grupo de flip-flops constituye un registro, ya que cada flip-flop es una celda binaria capaz de almacenar un bit de información. Un registro de n-bit tiene un grupo de n flip-flops y es capaz de almacenar cualquier información binaria que contenga n bits. Además de los flip-flops, un registro puede tener compuertas combinacionales que realicen ciertas tareas de procesamiento de datos. En su definición más amplia, un registro consta de un grupo de flip-flops y compuertas que efectúan su transición. Los flip-flops mantienen información binaria y las compuertas controlan cuándo y cómo se transfiere información nueva al registro.

Un registro capaz de correr su información binaria ya sea a la derecha o a la izquierda se denomina registro con corrimiento. La configuración lógica de un registro con corrimiento consta de una cadena de flip-flops conectados en cascada, con la salida de un flip-flop. Todos los flip-flops reciben un pulso común de reloj que causa el corrimiento de una etapa a la siguiente.

El registro con corrimiento más simple posible es uno que usa sólo flip-flops, como se muestra en la figura 5.29. La salida Q de un flip-flop dado se conecta a la entrada D del flip-flop a su derecha. Cada pulso de reloj corre el contenido del registrador una posición de bit a la derecha. La entrada serial determina qué pasa al flip-flop de la extrema izquierda durante el corrimiento. La salida serial se toma de la salida del flip-flop de la extrema derecha antes de la aplicación de un pulso. Aunque este registro corre su contenido a la derecha, si se gira la página de arriba hacia abajo, se encuentra que el registro corre su contenido a la izquierda. En consecuencia, un registro con corrimiento unidireccional puede funcionar ya sea como un registro de corrimiento a la derecha o con un registro de corrimiento a la izquierda.

El registro de la figura 5.29 corre su contenido con cada

pulso de reloj durante el borde negativo de la transición del pulso. (Esto se indica por el círculo pequeño asociado con la entrada de reloj en todos los flip-flops.) Si se desea controlar el corrimiento de modo que ocurra sólo con ciertos pulsos pero con otros, se debe controlar la entrada CP de registro. Sin embargo, si se usa el registro con corrimiento mostrado en la figura 5.29, el corrimiento puede controlarse con facilidad mediante una compuerta externa AND como se muestra a continuación.

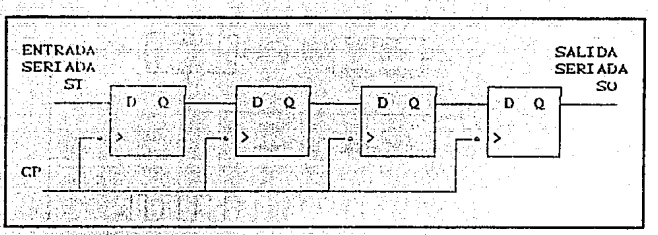


FIG. 5.29 REGISTRO DE CORRIMIENTO.

5.2.5.1 TRANSFERENCIA SERIAL.

Se dice que un sistema digital opera en modo serial cuando la información se transfiere y se manipula un bit a la vez. El contenido de un registro se transfiere a otro corriendo los bits de un registro a otro. La información se transfiere un bit a la vez corriendo los bits fuera del registro fuente al registro de destino.

La transferencia serial de información desde el registro A al registro B se hace con registros con corrimiento, como se muestra en el diagrama de bloque en la figura 5.30(a). La salida serial (SO) del registro A pasa a la entrada serial (SI) del registro B. Para evitar la pérdida de información almacenada en el registro fuente, el registro A se hace que circule su información por la conexión de salida serial a su terminal de entrada serial. El contenido inicial del registro B se corre saliendo a través de su salida serial y se pierde a menos que se transfiera a un tercer

registro con corrimiento. La entrada de control de corrimiento determina cuando y por cuantas veces se corren los registros. Esto se hace por la compuerta AND que permite que los pulsos de reloj pasen a las terminales CP solo cuando el control de corrimiento es 1.

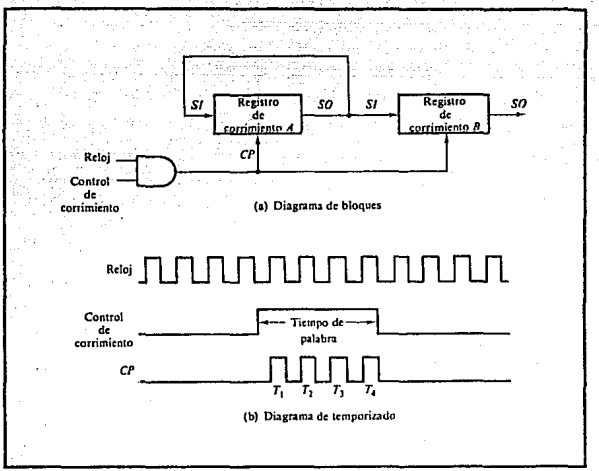


FIG. 5.30 TRANSFERENCIA SERIADA DEL REGISTRO A AL REGISTRO B.

Se supone que los registros con corrimiento tienen cuatro bits cada uno. La unidad de control que supervisa la transferencia debe diseñarse de tal forma que capacite los registros con corrimiento, a traves de la señal de control de corrimiento, por un tiempo de duración fija igual a cuatro pulsos de reloj. Esto se muestra en el diagrama de temporizado en la figura 5.30(b). La señal de control de corrimiento se sincroniza con el reloj y cambia de valor precisamente despues del borde negativo de un pulso de reloj. Los siguientes cuatro pulsos de reloj encuentran la señal de control de corrimiento en el estado

1, de modo que la salida de compuerta AND conectada a las terminales CP produce los cuatro pulsos T_1, T_2, T_3 y T_4. El cuarto pulso cambia el control de corrimiento a 0 y se inhabilitan los cuatro registros con corrimiento.

Se supone que el contenido binario de A antes del corrimiento es 1011 y el de B 0010. La transferencia serial desde A a B ocurrirá en cuatro pasos como se muestra en la Tabla 5.12. Después del primer pulso T_1, el bit de la extrema derecha de A se corre al bit de la extrema izquierda de B y, al mismo tiempo, este bit se circula a la posición de la extrema izquierda de A. Los otros bits de A y B se corren en lugar a la derecha. La salida serial previa desde B se pierde y su valor cambia de 0 a 1. Los siguientes tres pulsos realizan operaciones idénticas, corriendo los bits de A a B uno a la vez. Después del cuarto corrimiento, el control de corrimiento pasa a 0 y tanto el registro A como el B tienen el valor 1011. De este modo, el corrimiento de A se transfiere a B en tanto que el contenido a A permanece sin cambio.

PULSO TEMPORIZADOR	REG. A DE CORRIMIENTO	REG. B DE CORRIMIENTO	SALIDA EN SERIE DE B
VALOR INICIAL	1 0 1 1	0 0 1 0	0
DESPUES DE T_1	1 1 0 1	1 0 0 1	1
DESPUES DE T_2	1 1 1 0	1 1 0 0	0
DESPUES DE T_3	0 1 1 1	0 1 1 0	0
DESPUES DE T_4	1 0 1 1	1 0 1 1	1

TABLA 5.12 TRANSFERENCIA EN SERIE.

La diferencia entre los modos de operación serial y paralelo se hace aparente mediante este ejemplo. En el modo paralelo, está disponible la información de todos los bits de un registro y todos los bits pueden transferirse en forma simultánea durante un pulso de reloj. En el modo serial, los registros tienen una sóla entrada serial y una sóla salida serial. La información se transfiere un bit a la vez, mientras los registros están corriendo en la misma dirección.

Las computadoras pueden operar en un modo serial, un modo paralelo o en una combinación de ambos. Las operaciones seriales son mas lentas debido al tiempo que toma transferir la información de entrada y salida de los registros con corrimiento. Sin embargo, las computadoras seriales requieren menos hardware para realizar operaciones, porque un circuito comun puede usarse una y otra vez para manipular los bits que salen de los registros con corrimiento en una forma secuencial. El intervalo de tiempo entre los pulsos de reloj se denomina tiempo de bit, y el tiempo requerido para obtener el contenido completo de un registro con corrimiento se conoce como tiempo de palabra. Estas secuencias de tiempo se generan por las secciones de control del sistema. En una computadora en paralelo, las señales de control se habilitan durante un intervalo de pulso de reloj. Las transferencias en los registros son en paralelo y ocurren bajo la aplicacion de un solo pulso de reloj. En una computadora serial, las señales de control pueden mantenerse por un periodo igual a un tiempo de palabra. El pulso aplicado cada tiempo de bit transfiere el resultado de la operacion, uno a la vez, dentro de un registro con corrimiento. La mayoría de las computadoras operan en un modo paralelo ya que es un modo más rapido de operacion.

5.2.5.2 REGISTRO CON CORRIMIENTO BIDIRECCIONAL CON CARGA PARALELA.

Los registros con corrimiento pueden usarse para convertir datos seriales en datos en paralelo y viceversa. Si se tiene acceso a todas las salidas flip-flop de un registro con corrimiento, entonces la información que se introduce de manera serial por corrimiento puede tomarse en salida en paralelo mediante las salidas de los flip-flops. Si se agrega capacidad de carga en paralelo a un registro con corrimiento, entonces la información que se introduce en paralelo puede tomarse en salida en forma serial corriendo la información almacenada en el registro.

Algunos registros con corrimiento proporcionan las terminales necesarias de entrada y salida para la transferencia en paralelo. También pueden tener capacidades tanto de corrimiento a la derecha como de corrimiento a la izquierda. El registro con corrimiento mas general tiene todas las capacidades que se listan mas adelante. Otros pueden tener algunas de esas funciones, cuando menos con una operacion de corrimiento.

1. Un control de despeje para despejar el registro a 0.

2. Una entrada CP para los pulsos de reloj que sincronizan todas las operaciones.

3. Un control de corrimiento a la derecha para habilitar la operación de corrimiento a la derecha y a las líneas de entrada serial en líneas de salida asociadas con el corrimiento a la derecha.

4. Un control de corrimiento a la izquierda para habilitar la operación de corrimiento a la izquierda y las líneas de entrada serial y las líneas de salida asociadas con el corrimiento a la izqueirda.

5. Un control de carga en paralelo para habilitar una transferencia en paralelo de las n líneas de entrada asociadas con las transferencias en paralelo.

6. n líneas de salida en paralelo.

7. Un estado de control que deja la información sin cambio en el registro aun cuando se apliquen en forma continua pulsos de reloj.

Un registro capaz de correr tanto a la izquierda como a la derecha se denomina registro de corrimiento bidireccional. Uno que puede correr en una sola dirección se conoce como registro de corrimiento unidireccional. Si el registro tiene tanto capacidades de corrimiento como de carga en paralelo, se llama registro con corrimiento y carga paralela.

El diagrama de un registro con corrimiento que tiene todas las capacidades que se alistaron con anterioridad se muestra en la figura 5.31. Consta de cuatro flip-flops D, aunque pueden usarse flip-flops RS siempre que se inserte un inversor entre las terminales S y R. Los cuatro multiplexores (MUX) son parte del registro y aquí se dibujan en forma de diagrama de bloques. Los cuatro multiplexores tienen dos variables de selección comunes, s1 y s0. La entrada 0 en cada MUX se selecciona cuando s1 s0 = 00, la entrada 1 se selecciona cuando s1 s0 = 1 y en forma similar para las otras dos entradas de los multiplexores.

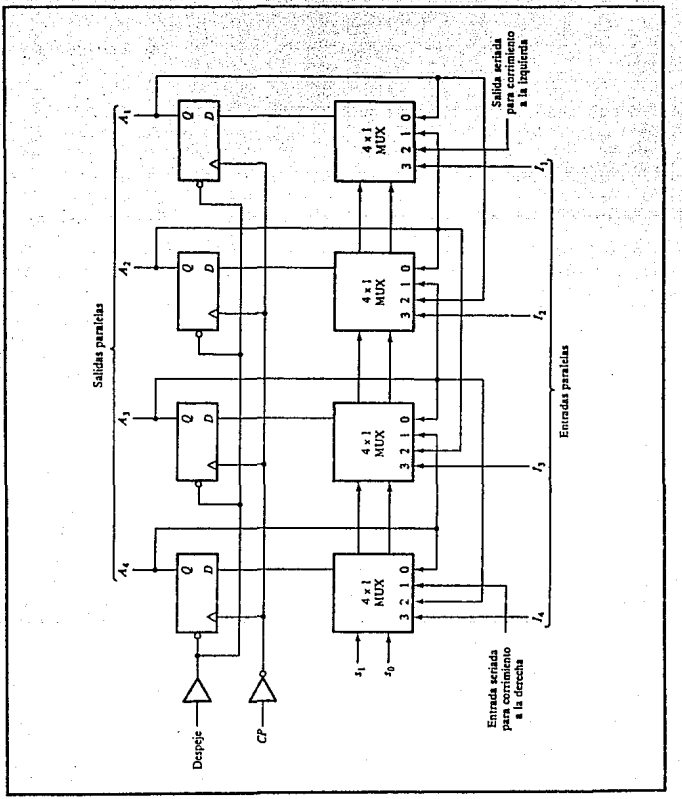


FIG. 5.91 REGISTRADOR DE CORRIMIENTO BIDIRECCIONAL DE 4 BITS CON CARGA PARALELA.

Las entradas s_1 y s_0 controlan el modo de operación del registro como se especifica en las funciones listadas en la Tabla 5.13. Cuando $s_1 s_0 = 00$, el valor presente del registro se aplica a las entradas D de los flip-flops. Esta condicion forma una trayectoria desde la salida de cada flip-flop hasta la entrada del mismo flip-flop. El siguiente pulso de reloj transfiere en cada flip-flop el valor binario que tenías previamente y no ocurre cambio de estado. Cuando $s_1 s_0 = 01$, las terminales 1 de las entradas del multiplexor tienen una trayectoria a las entradas D de los flip-flops. Esto causa una operación de corrimiento a la derecha, con la entrada serial transferida al flip-flop A_4. Cuando $s_1 s_0 = 10$, resulta una operación de corrimiento a la izquierda, con la otra entrada serial pasando al flip flop A_1. Por ultimo, cuando $s_1 s_0 = 11$, la información binaria en las líneas de entrada en paralelo se transfiere al registro en forma simultánea durante el siguiente pulso de reloj.

MODO DE CONTROL		OPERACION DE REGISTRO
S_1	S_0	
0	0	SIN CAMBIO
0	1	CORRIMIENTO A LA DER.
1	0	CORRIMIENTO A LA IZQ.
1	1	CARGA PARALELA

TABLA 5.13 TABLA DE FUNCION PARA EL REGISTRO DE LA FIG. 5.31

Un registro con corrimiento bidireccional con carga paralela es un registro de proposito general capaz de realizar tres operaciones: corrimiento a la izquierda, corrimiento a la derecha y carga paralela. No todos los registros con corrimiento disponibles en los circuitos MSI tienen todas estas capacidades. La aplicación particular determina la elección de un registro de corrimiento MSI con preferencia a otro.

5.2.6 DIVISORES DE FRECUENCIAS.

Consúltese la figura 5.32(a). Cada FF tiene sus entradas J y

K en el nivel 1, así que cambiará estados (se articulará) siempre que la señal en su entrada CLK pase de ALTO a BAJO. La sucesión de pulsaciones del cronómetro se aplica solamente a la entrada CLK del FF X_0. La salida X_0 se conecta a la entrada CLK del FF X_1 y a la salida X_1 se conecta a la entrada CLK del FF X_2. Las formas de onda de la figura 5.44(b) muestran la forma en que los FF cambian estados cuando se aplican las pulsaciones. Deben observarse los siguientes puntos importantes.

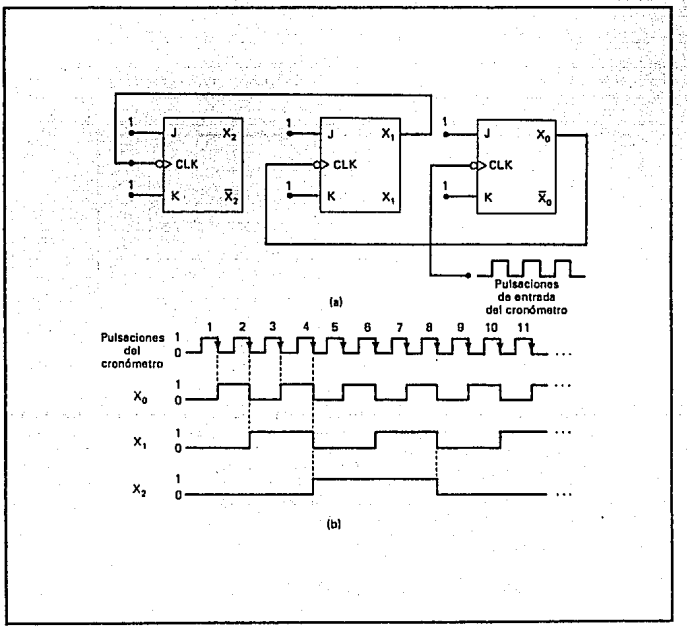


FIG. 5.32 BIESTABLE TIPO J-K CONECTADOS COMO UN CONTADOR BINARIO DE 3 BITS (MOD-8).

1. El FF Xo se articula en la transicion en sentido negativo de cada pulsacion de entrada del reloj. Asi, la salida Xo (forma de onda) tiene una frecuencia que es exactamente 1/2 de la frecuencia de la pulsación de reloj.

2. El FF X1 se articula cada vez que la salida Xo pasa de ALTO a BAJO. La ondiforme X1 tiene una frecuencia igual a exactamente 1/2 de la frecuencia de salida Xo y por lo tanto, 1/4 de la frecuencia del cronometro.

3. El FF X2 se articula cada vez que la salida X1 va de ALTO a BAJO. Así, la forma de onda X2 tiene 1/2 de la frecuencia de X1 y por tanto, 1/8 de la frecuencia de reloj.

4. Cada salida del FF es una onda cuadrada: (ciclo de utilidad al 50 por ciento).

Como se describio antes, cada FF divide la frecuencia de su entrada por 2. Asi, si se agregara un cuarto FF a la cadena, tendria una frecuencia igual a 1/16 de la frecuencia del reloj; y asi sucesivamente. Utilizando el numero adecuado de FF, este circuito pudiera dividir una frecuencia por cualquier potencia de 2. Especificamente, el uso de N flip-flops produciría una frecuencia de salida desde el ultimo FF que sea igual a $1/2^N$ de la frecuencia de entrada.

5.3 SISTEMAS SECUENCIALES ASINCRONOS.

Un circuito secuencial se especifica por una secuencia temporal de entradas, salidas y estados externos. En los circuitos secuenciales sincronos el cambio de estado interno ocurre como respuesta a los pulsos sincronizados de reloj. Los circuitos secuenciales asincronos no usan pulsos de reloj. El cambio interno ocurre cuando hay un cambio en las variables de entrada. Los elementos de memoria en los circuitos secuenciales sincronos son flip-flops controlados por reloj. Los elementos de memoria en los circuitos secuenciales asincronos son ya sea flip-flops sin reloj o elementos de retardo de tiempo. La capacidad de memoria de un dispositivo con retardo de tiempo se debe al hecho de que toma un tiempo finito para que la señal se propage a través de compuertas digitales. Un circuito secuencial asincrono con bastante frecuencia se asemeja a un circuito combinacional con retroalimentacion.

El diseño de los circuitos secuenciales asíncronos es más difícil que el de los circuitos síncronos debido a los problemas de temporizado implicados en la trayectoria de retroalimentación. En un sistema síncrono apropiadamente diseñado, los problemas de temporizado se eliminan por el disparo de todos los flip-flops con el borde del pulso. El cambio de un estado al siguiente ocurre durante el corto tiempo de la transición del pulso. Ya que el circuito asíncrono no usa un reloj, se permite que cambie el estado del sistema inmediatamente después de los cambios de entrada. Debe tenerse cuidado para asegurar que cada nuevo estado mantenga el circuito en una condición estable aun cuando existía una trayectoria de retroalimentación.

Los circuitos secuenciales asíncronos son útiles en una variedad de aplicaciones. Se usan cuando es importante la velocidad de operación, especialmente en los casos donde el sistema digital debe responder con rapidez sin tener que esperar un pulso de reloj. Son más económicos para utilizarse en sistemas independientes pequeños que requieren sólo unos cuantos componentes, donde puede no ser práctico incurrir en el gasto de proporcionar un circuito para generar pulsos de reloj. Los circuitos asíncronos son útiles en aplicaciones donde las señales de entrada al sistema pueden cambiar en cualquier momento, con independencia de un reloj interno. La comunicación entre dos unidades con cada unidad que tiene su propio reloj independiente debe hacerse con circuitos asíncronos. Los diseñadores digitales con frecuencia producen un sistema mixto donde alguna parte del sistema síncrono tiene las características de un circuito asíncrono. El conocimiento del comportamiento de la lógica secuencial asíncrona es de ayuda para verificar que el sistema digital total está operando en la forma apropiada.

En la figura 5.33 se muestra el diagrama de bloques de un circuito secuencial asíncrono. Consta de un circuito combinacional y elementos de retardo conectados para formar lazos de retroalimentación. Hay n variables de entrada, m variables de salida y k estados internos. Los elementos de retardo pueden visualizarse como proveedores de memoria a corto plazo para el circuito secuencial. En el circuito tipo compuerta, el retardo de propagación que existe en la trayectoria del circuito combinacional desde la entrada a la salida proporciona suficiente retardo junto con el lazo de retroalimentación, de modo que en realidad no se insertan elementos específicos de retardo en la trayectoria de retroalimentación. Las variables de estado presente y de estado siguiente en los circuitos secuenciales asíncronos por costumbre se denominan variables secundarias y

variables de excitación, respectivamente. Las variables de excitación no deben confundirse con la tabla de excitación que se usa en el diseño de los circuitos secuenciales controlados por reloj.

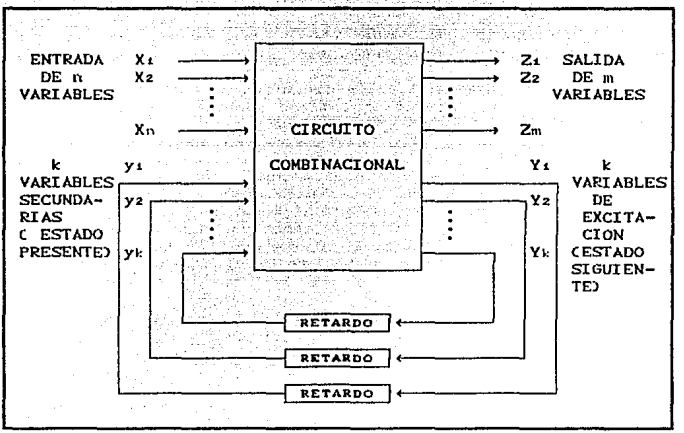


FIG. 5.39 DIAGRAMA DE BLOQUES DE UN CIRCUITO SECUENCIAL ASINCRONO.

Cuando una variable de entrada cambia de valor, las variables secundarias y no cambian en forma instantánea. Toma cierta cantidad de tiempo para que la señal se propague desde las terminales de entrada a través del circuito combinacional a las variables de excitación y donde se generan nuevos valores para el estado siguiente. Estos valores se propagan a través de los elementos de retardo y llegan a ser el nuevo estado presente para las variables secundarias. Obsérvese la distinción entre las y y las Y. En las condiciones de estado estacionario son las mismas, pero durante la transición no lo son. Para un valor dado de variables de entrada, el sistema es estable si el circuito alcanza una condición de estado estacionario con $y_i = Y_i$ para $i = 1, 2, \ldots, k$. De otra manera, el circuito está en transición

continua y se dice que está inestable. Es importante darse cuenta que una transicion de un estado estable a otro ocurre solo en respuesta a un cambio en una variable de entrada. Esto contrasta con los sistemas síncronos, donde las transiciones de estado ocurren en respuesta a la aplicacion de un pulso de reloj.

Para asegurar la operacion apropiada, los circuitos secuenciales asincronos necesitan alcanzar un estado estable antes de que la entrada se cambie a un nuevo valor. Debido a los retardos en el alambrado y los circuitos de compuerta, es posible tener dos o mas variables de entrada cambiando exactamente en el mismo instante sin una insertidumbre respecto a cual cambia primero. Por tanto, los cambios simultaneos de dos o más variables por lo comun se prohiben. Esta restriccion significa que solo una variable de entrada puede cambiar a la vez y el tiempo entre dos cambios de entrada debe ser mas largo que el tiempo que toma el circuito para alcanzar un estado estable. Este tipo de operacion se define como un modo fundamental. La operacion en modo fundamental supone que las señales de entrada cambian una a la vez y solo cuando el circuito esta en una condicion estable.

5 . 3 . 1 PROCEDIMIENTO DE ANALISIS.

El análisis de los circuitos secuenciales asíncronos consiste en obtener una tabla o un diagrama que describe la secuencia de los estados internos y las salidas como una funcion de los cambios en las variables de entrada. Un diagrama logico manifiesta un comportamiento del circuito secuencial asincrono si tiene uno o más lazos de retroalimentacion o si incluye flip-flops sin control de reloj. En cada seccion se investigara el comportamiento de los circuitos secuenciales asincronos que tienen trayectorias de retroalimentacion sin emplear flip-flops. Los flip-flops sin control de reloj se llaman seguros.

El procedimiento de análisis se presentará mediante tres ejemplos específicos. En el primer ejemplo se introduce la tabla de transicion. En el segundo ejemplo se define la tabla de flujo. En el tercer ejemplo se investiga la estabilidad de los circuitos secuenciales asincronos.

5.3.1.1 TABLA DE TRANSICION

En la figura 5.34 se muestra un ejemplo de un circuito secuencial asincrono solo con compuertas. En el diagrama se ilustran con claridad dos lazos de retroalimentacion, desde las salidas de la compuerta OR regresando a las entradas de la compuerta AND. El circuito consta de una variable de entrada, x, y dos estados internos. Los estados internos tienen dos variables de excitacion, Y_1 y Y_2, y dos variables secundarias, y_1 y y_2. El retardo asociado con cada lazo de retroalimentacion se obtiene mediante el retardo de propagacion en cada entrada y y su salida correspondiente Y. Cada compuerta logica en la trayectoria introduce un retardo de propagacion de 2 a 10 nanosegundos. Los alambres que conducen la señal electrica introducen aproximadamente un retardo de nanosegundo por cada pie de alambre. Asi, no son necesarios elementos externos de retardo adicionales cuando el circuito combinacional y los alambres en la trayectoria de retroalimentación proporcionan suficiente retardo.

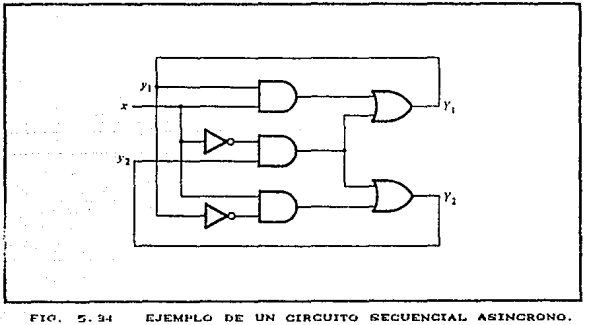


FIG. 5.34 EJEMPLO DE UN CIRCUITO SECUENCIAL ASINCRONO.

El analisis del circuito principia por considerar las variables de excitacion como salidas y las variables secundarias como entradas. Entonces se deriban las expresiones booleanas para las variables de excitacion como una funcion de la entrada y de las variables secundarias. Estas se obtienen con facilidad mediante el diagrama logico

$$Y_1 = xy_1 + x'y_2$$

$$Y_2 = xy'_1 + x'y_2$$

El siguiente paso es graficar las funciones Y_1 y Y_2 en un mapa como se muestra en la figura 5.35(a) y (b). Los valores binarios codificados de las variables y se usan para etiquetar los renglones, y la variable de entrada x se usa para designar las columnas. Esta configuracion resulta en un mapa de tres variables ligeramente diferente del que se utilizo en los capítulos anteriores. Sin embargo, sigue siendo un mapa valido, y este tipo de configuracion es mas conveniente cuando se trata de circuitos secuenciales asíncronos. Obsérvese que las variables que pertenecen a las casillas apropiadas no estan marcadas a los lados del mapa.

La tabla de transición que se muestra en la figura 5.35(c) se obtiene mediante los mapas combinando los valores binarios en las casillas correspondientes. La tabla de transicion muestra el valor de $Y = Y_1 Y_2$ dentro de cada casilla. El primer bit de Y se obtiene del valor de Y_1, y el segundo bit se obtiene del valor de Y_2 en la misma posicion de casilla. Para que un estado sea estable, el valor de Y debe ser el mismo que el de $y = y_1 y_2$. Las anotaciones en la tabla de transicion donde $Y = y$ se encierran dentro de un círculo para indicar una condición estable. Una anotacion sin círculo representa un estado inestable.

y_1y_2 \ x	0	1
00	0	0
01	1	0
11	1	1
10	0	1

(a) Mapa para $Y_1 = xy_1 + x'y_2$

y_1y_2 \ x	0	1
00	0	1
01	1	1
11	1	0
10	0	0

(b) Mapa para $Y_2 = x'y_1 + x'y_2$

y_1y_2 \ x	0	1
00	00	01
01	11	01
11	11	10
10	00	10

(c) Tabla de transicion

FIG. 5.35 MAPAS Y TABLA DE TRANSICION PAR EL CIRCUITO DE LA FIG. 5.34.

Ahora se considera el efecto de un cambio en la variable de

entrada. La casilla para x = 0 y y = 00 en la tabla de transición muestra que Y = 00. Ya que Y representa el valor siguiente de y, esta es una condición estable. Si X cambia de 0 a 1 cuando y = 00, el circuito cambia el valor de Y a 01. Esto representa una condición inestable temporal, porque Y no es igual al valor presente de y. Lo que sucede a continuación es que tan pronto como la señal se propaga para hacer Y = 01, la trayectoria de retroalimentación en el circuito causa un cambio en y a 01. Esto se manifiesta en la tabla de transición por una transición desde el primer renglón (y = 00) al segundo renglon, donde y = 01. Ahora que y = Y el circuito alcanza una condición estable con una entrada de x = 1. En general, si un cambio en la entrada lleva el circuito a un estado inestable, el valor de y cambiara (mientras x permanece sin cambio) hasta que alcanza un estado estable (encerrado dentro de un circuito). Usando este tipo de análisis para las casillas restantes de la tabla e transicion se encuentra que el circuito repite la secuencia de los estados 00, 01, 11, 10 cuando la entrada alterna en forma repetida entre 0 y 1.

Obsérvese la diferencia entre un circuito síncrono y uno secuencial asíncrono. En el sistema síncrono, el estado presente está totalmente especificado por los valores flip-flop y no cambia mientras el pulso de reloj está inactivo. En un circuito asíncrono el estado interno puede cambiar inmediatamente despues de un cambio en la entrada. Debido a esto, algunas veces es conveniente combinar el estado interno con el valor de entrada junto y llamarlo al estado total del circuito. El circuito cuya tabla de transicion se muestra en la figura 5.35(c) llené cuatro estados totales estables, $y_1\ y_2$ x = 000, 011, 110 y 101 y cuatro estados totales inestables, 001, 010, 111 y 100.

ESTADO PRESENTE		ESTADO SIGUIENTE			
		x = 0		x = 1	
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	1	1	1	0

TABLA 5.14 TABLA DE ESTADOS PARA EL CIRCUITO DE LA FIG. 5.34.

La tabla de transicion de los circuitos secuenciales asincronos es similar a la tabla de estado que se utiliza para los circuitos sincronos. Si se cinsideran las variables secundarias como el estado presente y las variables de excitacion como el estado siguiente, se obtiene la tabla de estado como se muestra en la Tabla 5.14. Esta tabla proporciona la misma informacion que la tabla de transicion. Hay una restriccion que se aplica al caso asincrono pero no se aplica al caso sincrono. En la tabla de transicion asincrona por lo comun hay cuando menos una anotacion de estado siguiente que es la misma que el valor de estado presente en cada renglon. En otra forma, todos los estados totales en ese renglon seran inestables.

El procedimiento para obtener una tabla de transicion mediante el diagrama de circuito secuencial asincrono es como sigue:

1. Determinese todos los lazos de retroalimentacion en el circuito.

2. Denotese la salida de cada lazo de retroalimentacion con la variable Y_i y su entrada correspondiente con y_i para $i = 1, 2, \ldots, k$ donde k es el numero de lazos de retroalimentacion en el circuito.

3. Derivense las funciones booleanas para todas las Y como una funcion de las entradas externas y las y.

4. Grafiquese cada funcion Y en un mapa usando las variables y para los renglones y las entradas externas para las columnas.

5. Combinense todos los mapas en una tabla mostrando el valor de $Y = Y_1 Y_2 \ldots Y_k$ dentro de cada casilla.

6. Encierrense dentro del circuito los valores de Y en cada casilla que son iguales al valor de $y = y_1 y_2 \ldots y_k$ en el mismo renglon.

Una vez que está disponible la tabla de transicion, el omportamiento del circuito puede analizarse por la observacion

de las transiciones de estado como una función de cambios en las variables de entrada.

5.3.1.2 TABLA DE FLUJO.

Durante el diseño de circuitos secuenciales asíncronos es más conveniente denotar los estados con símbolos alfabéticos sin hacer referencia específica a sus valores binarios. Dicha tabla se denomina tabla de flujo. Una tabla de flujo es similar a una tabla de transición excepto que los estados internos se simbolizan con letras en lugar de números binarios. La tabla de flujo también incluye los valores de salida del circuito para cada estado estable.

En la figura 5.36 se muestran ejemplos de tablas de flujo. La que se ilustra en la figura 5.36(a) tiene cuatro estados denotados con las letras a, b, c, d. Se reduce a la tabla de transición en la figura 5.35(c) si se asignan los siguientes valores binarios a los estados: a = 00, b = 01, c = 11 y d =10. La tabla en la figura 5.36(a) se conoce como tabla de flujo primitiva debido a que tiene sólo un estado estable en cada

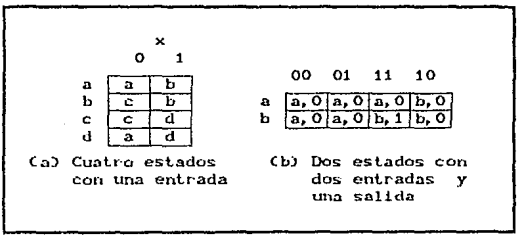


FIG. 5.36 EJEMPLOS DE TABLAS DE FLUJO.

renglón. En la figura 5.36(b) se muestra una tabla de flujo con más de un estado estable en el mismo renglón. Tiene dos estados a y b, dos entradas x_1 y x_2 y una salida z. El valor binario de la variable de salida se indica dentro de la casilla a continuación

del símbolo de estado y se separa con una coma. Mediante la tabla de flujo se observa el siguiente comportamiento del circuito. Si $x_1 = 0$, el circuito está en el estado a. Si x_1 pasa a 1 mientras x_2 es 0, el circuito pasa al estado b. Con las entradas $x_1\ x_2 = 11$, el circuito puede estar ya sea en el estado b. Si se encuentra en el estado a, la salida es 0, y si está en el estado b, la salida es 1. El estado b se mantiene si las entradas cambian desde 10 a 11. El circuito permanece en el estado a si las entradas cambian desde 01 a 11. Recuerdese que en el modo fundamental, dos variables de entrada no pueden cambiar en forma simultánea, y por tanto no se permite un cambio de entradas desde 00 a 11.

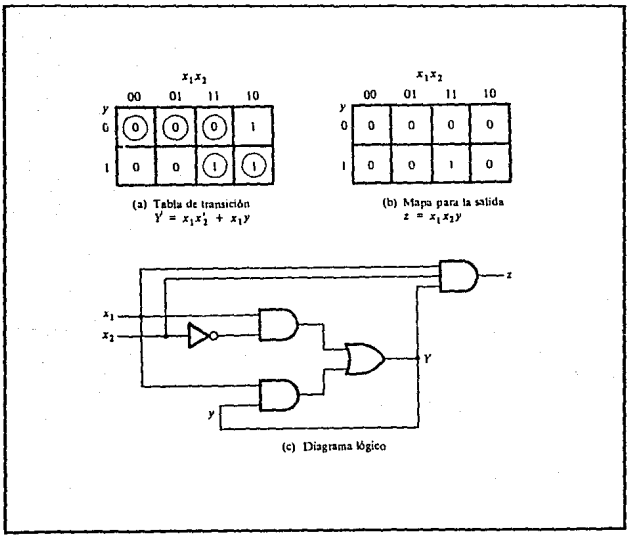


FIG. 5.37 DERIVACION DEL CIRCUITO ESPECIFICADO POR LA TABLA DE FLUJO DE LA FIG. 5.36(b).

Con objeto de obtener el circuito descrito por una tabla de flujo, es necesario asignar a cada estado un valor binario distinto. Esta asignacion convierte la tabla de flujo en una tabla de transicion mediante la cual puede derivarse el diagrama logico. Esto se ilustra en la figura 5.37 para la tabla de flujo en la figura 5.36(b). Se asigna el valor binario 0 al estado a y el valor binario 1 al estado b. El resultado es la tabla de transicion en la figura 5.37(a). El mapa de salida que se muestra en la figura 5.37(b) se obtiene de manera directa mediante los valores de salida en la tabla de flujo. La funcion de salida z se simplifica mediante los dos mapas. El diagrama logico de circuito se presenta en la figura 5.37(c).

Este ejemplo demuestra el procedimiento para obtener el diagrama logico mediante una tabla de flujo dada. Este procedimiento no siempre es tan simple como en este ejemplo. Hay varias dificultades asociadas con la asignacion de estado binario y con la salida asignada en los estados inestables. Estos problemas se exponen en detalle en las siguientes secciones.

5.3.1.3 CONDICIONES DE CARRERA.

Se dice que existe una condición de carrera en un circuito secuencial asincrono cuando dos o mas variables binarias de estado cambian de valor en respuesta a un cambio en una variable de entrada. Cuando se encuentran estados desiguales, una condicion de carrera puede provocar que las variables de estado cambien de una manera impredecible. Por ejemplo, si las variables de estado deben cambiar desde 00 a 11, la diferencia en retardos puede causar que la primera variable cambie mas rapido que la segunda, con el resultado de que las variables de estado cambian en secuencia desde 00 hasta 10 y entonces a 11. Si la segunda variable cambia más rapido que la primera, las variables de estado cambiaran desde 00 a 01 y entonces a 11. En consecuencia el orden en el cual cambian las variables puede no conocerse por anticipado. Si el estado final estable que alcanza el circuito no depende del orden en el cual cambian las variables se estado, la carrera se denomina carrera no critica. Si es posible terminar en dos o mas estados estables diferentes dependiendo del orden en el cual cambian las variables de estado, entonces es una carrera crítica. Para la operacion apropiada, deben evitarse las carreras criticas.

Los dos ejemplos que se muestran en la figura 5.38 ilustran carreras no criticas. Se principia con el estado total estable $y_1\ y_2\ x = 000$ y entonces cambia la entrada desde 0 a 1. Las variables de estado deben cambiar desde 00 a 11, lo cual define una condicion de carrera. Las trancisiones listadas bajo cada tabla muestran tres formas posibles en que las variables de estado pueden cambiar. Pueden cambiar ya sea en forma simultanea desde 00 a 11, o pueden cambiar en secuencia desde 00 a 01 y, entonces, a 11, o pueden cambiar en secuencia desde 00 a 10 y entonces a 11. En cualquier caso, el estado estable final es el mismo, lo cual resulta en una condicion de carrera no critica. En (a) el estado total final es $y_1\ y_2\ x = 111$ y en (b) es 011.

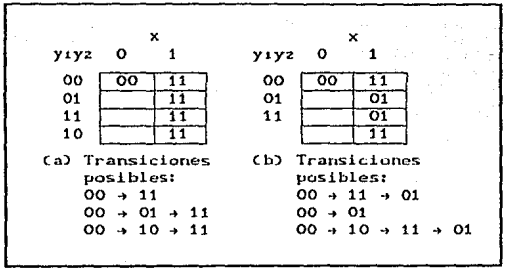


FIG. 5.38 EJEPLOS DE CARRERAS NO CRITICAS.

Las tablas de transicion en la figura 5.39 ilustran carreras criticas. De nuevo se principia con el estado total estable $y_1\ y_2\ x = 000$ y entonces cambia la entrada desde 0 a 1. Las variables de estado deben cambiar desde 00 a 11. Si cambian en forma simultanea, el estado total estable es 111. En la tabla de transicion de la parte (a), si Y_2 cambia a 1 antes de Y_1 debido al retardo desigual de propagación, entonces el circuito pasa al estado total estable 011 y permanece en él. Por otra parte, si Y_1 cambia primero, el estado interno llega a ser 10 y el circuito permanecera en el estado total estable 101. Por tanto, la carrera es crítica debido a que el circuito pasa a estados estables diferentes dependiendo del orden en el cual cambiaron las variables de estado. En la tabla de trancisiones en la figura 5.39(b) se ilustra otra carrera crítica donde resultan dos trancisiones posibles en un estado total final, pero la tercera

transicion posible pasa a un estado total diferente.

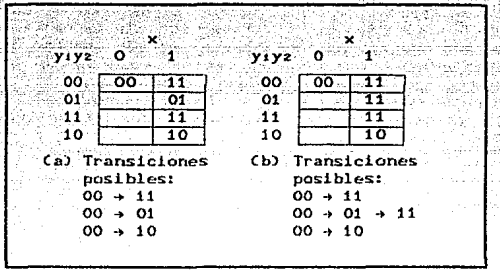


FIG. 5.39 EJEPLOS DE CARRERAS CRITICAS.

Pueden evitarse las carreras dando una asignación binaria apropiada a las variables de estado. Las variables de estado deben estar asignadas a números binarios en tal forma que solo una variable de estado pueda cambiar en cualquier momento cuando ocurre una transicion de estado en la tabla de flujo.

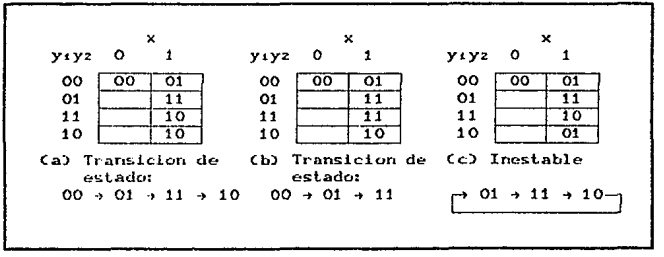


FIG. 5.40 EJEPLOS DE CICLOS.

Pueden evitarse las carreras si el circuito se dirige a través de estados intermedios inestables con un cambio unico de variable

de estado. Cuando un circuito pasa a través de una secuencia única de estados inestables, se dice que tiene un ciclo. En la figura 5.40 se ilustra la ocurrencia de ciclos. De nuevo se principia con $y_1\ y_2 = 00$ y entonces cambia la entrada desde 0 a 1. La tabla de transición de la parte (a) da una secuencia única que termina en un estado total estable 101. En la tabla en (b) se muestra que aun cuando las variables de estado cambian desde 00 a 11, el ciclo proporciona una transición única desde 00 a 01 y entonces a 11. Debe tenerse cuidado cuando se usa un ciclo que termina un estado estable. Si un ciclo no termina con un estado estable, el circuito seguirá pasando de un estado inestable a otro, haciendo inestable todo el circuito. Esto se demuestra en la Figura 5.40(c) y también en el ejemplo siguiente.

5.3.1.4 CONSIDERACIONES DE ESTABILIDAD.

Debido a la conexión de retroalimentación que existe en los circuitos secuenciales asíncronos, debe tenerse cuidado para asegurar que el circuito no llegará a ser inestable. Una condición inestable causará que el circuito oscile entre dos estados inestables. El método de análisis de la tabla de transición puede ser útil para detectar la ocurrencia de la inestabilidad.

Por ejemplo, considérese el circuito que se muestra en la figura 5.41(a). La función de excitación es:

$$Y = (x_1\ y)'x_2 = (x'_1 + y')x_2 = x'_1\ x_2 + x_2\ y'$$

La tabla de transición para el circuito se muestra en la figura 5.41(b). Los valores de Y que son iguales a y se encierran dentro de círculos y representan los estados estables. Las anotaciones sin círculo indican condiciones inestables. Obsérvese que la colomna 11 no tiene estados estables. Esto significa que con la entrada $x_1\ x_2$ fijada en 11, los valores de Y y y nunca serán los mismos. Si $y = 0$, entonces $Y = 1$; lo cual provoca una transición al segundo renglón de la tabla con $y = 1$ y $Y = 0$. Esto causa una transición de regreso al primer renglón, con el resultado de que la variable de estado alterna entre 0 y 1 en forma indefinida en tanto que la entrada sea 11.

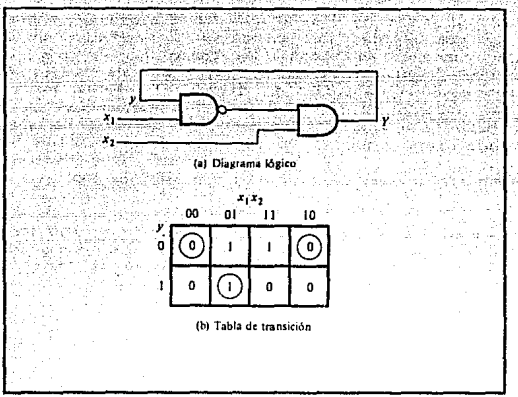


FIG. 5.41 EJEPLOS DE UN CIRCUITO INESTABLE.

La condición inestable puede detectarse en forma directa mediante el diagrama lógico. Sea $x_1 = 1$, $x_2 = 1$ y $y = 1$. La salida de la compuerta NAND es igual a 0, y la salida de la compuerta AND es igual a 0, haciendo Y igual a 0, con el resultado de que $Y \neq y$. Ahora si $y \neq 0$, la salida de la compuerta NAND es 1, la salida de la compuerta AND es 1, haciendo Y igual a 1, con el resultado de que $Y \neq y$. Si se supone que cada compuerta tiene un retardo de propagación de 5 ns (incluyendo los alambres) se encuentra que Y será 0 por 10 ns y 1 para los siguientes 10 ns. Esto resultará en una onda de forma cuadrada con un periodo de 20 ns. La frecuencia de oscilación es la inversa del periodo y es igual a 50 MHz. A menos que se está diseñando un generador de onda cuadrada, la inestabilidad que puede ocurrir en los circuitos secuenciales asíncronos es indeseable y debe evitarse.

5.3.2 PROCEDIMIENTO DE DISEÑO.

El diseño de un circuito secuencial asíncrono principia desde el planteamiento del problema y culmina en un diagrama lógico.

Hay un número de pasos de diseño que deben llevarse con objeto de minimizar la complejidad del circuito y producir un circuito estable sin carreras criticas. En forma breve, los pasos de diseño son como sigue. Se obtiene una tabla de flujo primitiva mediante las especificaciones de diseño. La tabla de flujo se reduce a un numero mínimo de estados. Entonces se da una asignacion binaria a los estados mediante la cual se obtiene la tabla de transicion. A partir de la tabla de transicion se deriva el diagrama logico como un circuito combinacional con retroalimentación o como un circuito con seguros SR.

El proceso de diseño se demostrará llevando a cabo un ejemplo especifico. Una vez que se domine este ejemplo, sera mas facil entender los pasos de diseño que se enumeran al final de esta sección.

5.3.2.1 EJEMPLO DE DISENO.

Es necesario diseñar un circuito de seguro con compuerta con dos entradas, G (compuerta) y D (datos), y una salida, Q. La informacion binaria presente en la entrada D se transfiere a la salida Q cuando G es igual a 1. La salida Q seguira las entradas D en tanto que G = 1. Cuando G pasa a 0, la informacion que estaba presente en la entrada D en el momento de la transicion ocurrida se retiene en la salida Q. El seguro con compuerta es un elemento de memoria que afecta el valor de D cuando G = 1 y retiene este valor despues de que G pasa a 0. Una vez que G = 0, un cambio en D no cambia el valor de la salida Q.

5.3.2.2 TABLA DE FLUJO PRIMITIVA.

Como se definió con anterioridad, una tabla de flujo primitiva es una tabla de flujo sólo con un estado total estable en cada renglón. Recuérdese que un estado total consiste del estado interno combinado con la entrada. La derivacion de la tabla de flujo primitiva puede facilitarse si primero se forma una tabla con todos los estados totales posibles en el sistema. Esto se muestra en la Tabla 5.15 para el seguro con una compuerta. Cada renglon en la Tabla especifica un estado total que consta en una designacion alfabetica para el estado interno y una combinación posible de entrada para D y G. La salida Q también se muestra

para cada estado total. Se principia con los dos estados totales que tienen G = 1. Mediante las especificaciones de diseño se conoce que Q = 0 si DG = 01 y Q = 1 si DG = 11 ya que D debe ser igual a Q cuando G = 1. Se asignan estas condiciones a los estados a y b. Cuando G pasa a 0, la salida depende del ultimo valor de D. Por tanto, si la transicion de DG es desde 01 a 00 a 10, entonces Q debe permanecer en 0, ya que D es 0 en el momento de la transicion desde 1 a 0 en G. Si la transicion de DG es desde 11 a 10 a 00, entonces Q debe permanecer en 1. Esta informacion resultan seis estados totales diferentes, como se muestra en la tabla. Obsérvese que las transiciones simultaneas de dos variables de entrada como desde 01 a 10 o desde 11 a 00 no se permiten en el modo fundamental de operacion.

ESTADO	ENTRADAS D	G	SALIDAS Q	C O M E N T A R I O S
a	0	1	0	D=Q PORQUE G=1
b	1	1	1	D=Q PORQUE G=1
c	0	0	0	DESPUES DEL a O DEL b
d	1	0	0	DESPUES DEL c
e	1	0	1	DESPUES DEL b O DEL f
f	0	0	1	DESPUES DEL e

TABLA 5.15 ESTADOS TOTALES DE SEGURO CON COMPUERTAS.

La tabla de flujo primitiva para el seguro con compuerta se muestra en la figura 5.42. Tiene un renglon para cada estado y una columna para cada combinacion de entrada. Primero se llena una casilla en cada renglon que pertenece al estado estable en ese renglón. Esas anotaciones se determinan mediante la Tabla 5.15. Por ejemplo, el estado a es estable y la salida es 0 cuando la entrada es 01. Esta informacion se anota en la tabla de flujo en el primer renglon y en la segunda columna. En forma similar, los otros cinco estados estables junto con su salida se anotan en las columnas de entrada correspondientes.

A continuación se observa que ya que no se permite que ambas entradas cambien en forma simultanea, pueden anotarse marcas con guión en cada renglón en las casillas que difieren en dos o mas

variables de las variables de entrada asociadas con el estado estable. Por ejemplo, el primer renglón en la tabla de flujo muestra un estado estable con una entrada de 01. Ya que solo puede cambiar una entrada en un momento dado, puede cambiar a 00 o 11 pero no a 10. Por tanto, se anotan dos guiones en al columna 10 del renglón a. Esto resultará a la larga en una condición no importa para el estado siguiente y la salida en esta casilla. Siguiendo este procedimiento se llena una segunda casilla en cada renglón de la tabla de flujo primitiva.

	D G 00	01	11	10
a	c, -	a, 0	b, -	-, -
b	-, -	a, -	b, 1	e, -
c	c, 0	a, -	-, -	d, -
d	c, -	-, -	b, -	d, 0
e	f, -	-, -	b, -	e, -
f	f, 1	a, -	-, -	e, -

FIG. 5.42 TABLA PRIMITIVA DE FLUJO.

A continuación es necesario encontrar valores para dos casillas más en cada renglón. Los comentarios anotados en la Tabla 5.15 pueden ayudar a derivar la información necesaria. Por ejemplo, el estado c está asociado con la entrada 00 y se alcanza después de que una entrada cambia del estado a o d. Por tanto, un estado inestable c se muestra en la columna 00 y los renglones a y d en la tabla de flujo. La salida se marca con un guion para indicar una condición no importa. La interpretación de esto es que si el circuito está en el estado estable a y la entrada cambia desde 01 a 00, el circuito pasa primero a un estado siguiente inestable c, el cual cambia el valor del estado presente desde a a c, causando una transición al tercer renglón y primera columna de la tabla de flujo. Los valores de estado inestable para las otras casillas se determinan en forma similar. Todas las salidas asociadas con los estados inestables se marcan

con un guión para indicar condiciones no importa. La asignación de los valores reales a las salidas se exponen en mas detalle después de que el ejemplo de diseño se completa.

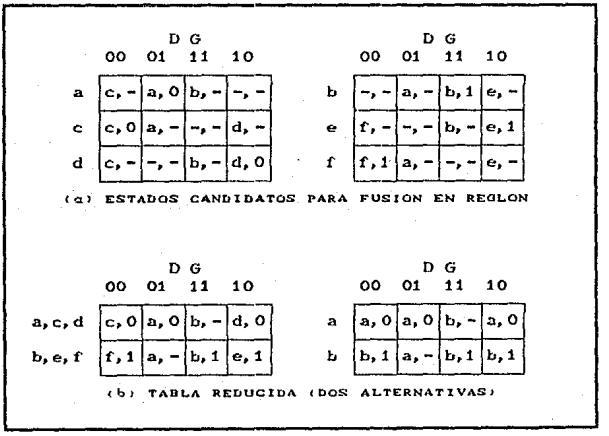
5.3.2.3 REDUCCION DE LA TABLA DE FLUJO PRIMITIVA.

La tabla de flujo primitiva tiene sólo un estado estable en cada renglón. La tabla puede reducirse a un numero menor de renglones y dos o mas estados estables se colocan en el mismo renglón de la tabla de flujo. La agrupación de los estados estables desde renglones separados a un renglón común se denomina fusión. La fusión de un numero de estados estables en el mismo renglón significa que la variable binaria de estado que se asigna en forma final al renglón fusionado no cambiara cuando la variable de entrada cambia. Esto se debe a que en una tabla de flujo primitiva la variable de estado cambia cada vez que cambia la entrada, pero una tabla de flujo reducida, un cambio de la entrada no provocara un cambio en la variable de estado si el siguiente estado estable está en el mismo renglón.

Con objeto de completar el ejemplo de diseño sin pasar a través del procedimiento formal, se aplicara el proceso de fusión usando una versión simplificada de las reglas de fusión. Pueden fusionarse dos o mas renglones en la tabla primitiva de flujo en un renglon si hay estado y salidas sin conflicto en cada una de las columnas. Siempre que un símbolo de estado y anotaciones no importa se encuentran en la misma columna, el estado se anota en el renglón fusionado. Ademas, si el estado se encientra dentro de un círculo en umo de los renglones, también esta dentro de un círculo en el renglón fusionado. El valor de salida se incluye con cada estado estable en el renglon fusionado.

Ahora se aplican estas reglas a la tabla de flujo primitiva en la figura 5.42. Para ver como se hace esto, la tabla de flujo primitiva se separa en dos partes de tres renglones cada una, como se muestra en la figura 5.43(a). Cada parte muestra tres estados estables que pueden fusionarse ya que no hay anotaciones en conflicto en cada una de las cuatro columnas. La primera columna ilustra el estado c en todos los renglones y 0 o un guión para la salida. Ya que un guión representa una condición no importa, puede asociarse con cualquier estado o salida. Los dos guiones en la primera columna pueden tomarse como salida 0 para hacer todos los tres renglones idénticos a un estado estable c

con una salida 0. La segunda columna muestra que los guiones pueden asignarse para corresponder a un estado estable a con una salida 0. Obsérvese que si el estado está dentro de un círculo en uno de los renglones, también está dentro de un círculo en el renglón fusionado. En forma similar, La tercera columna puede fusionarse en un estado inestable b con una salida despreocupada y la cuarta columna puede fusionarse en un estado estable d y una salida 0. Por lo tanto, los tres renglones a, c y d pueden fusionarse en un renglón con tres estados estables y un estado inestable como se muestra en el primer renglón en la figura 5.43(b). El segundo renglón de la tabla reducida resulta por la fusión de los renglones b, e y f de la tabla de flujo primitiva. Hay dos maneras de dibujar la tabla reducida. Los símbolos alfabéticos para los estados pueden retenerse para mostrar la relación entre las tablas de flujo primitiva y reducida. La otra alternativa es definir un símbolo común alfabético para todos los estados estables de los renglones fusionados. En consecuencia los estados c y d están reemplazados por el estado a, y los estados e y f se reemplazan por el estado b. Ambas alternativas se muestran en la figura 5.43(b).



	D G 00	01	11	10
a	c, -	a, 0	b, -	-, -
c	c, 0	a, -	-, -	d, -
d	c, -	-, -	b, -	d, 0

	D G 00	01	11	10
b	-, -	a, -	b, 1	e, -
e	f, -	-, -	b, -	e, 1
f	f, 1	a, -	-, -	e, -

(a) ESTADOS CANDIDATOS PARA FUSION EN REGLON

	D G 00	01	11	10
a, c, d	c, 0	a, 0	b, -	d, 0
b, e, f	f, 1	a, -	b, 1	e, 1

	D G 00	01	11	10
a	a, 0	a, 0	b, -	a, 0
b	b, 1	a, -	b, 1	b, 1

(b) TABLA REDUCIDA (DOS ALTERNATIVAS)

FIG. 5.43 REDUCCION DE LA TABLA PRIMITIVA DE FLUJO.

5.3.2.4 TABLA DE TRANSICION Y DIAGRAMA LOGICO.

Con objeto de obtener el circuito descrito por la tabla de flujo reducida, es necesario asignar a cada estado un valor binario indivudual. Esta asignacion convierte la tabla de flujo en una tabla de transicion. En el caso general, una asignacion binaria de estado debe hacerse para asegurar que el circuito quedara libre de carreras criticas. Por fortuna, no puede haber carreras criticas en una tabla de flujo de dos renglones, y por tanto puede terminarse el diseño del seguro con compuertas . Al asignar 0 al estado a y 1 al estado b en la tabla de flujo reducida en la figura 5.43(b), se obtiene la tabla de transicion que se muestra en la figura 5.44(a). La tabla de transicion en efecto es un mapa para la variable de excitacion Y. La funcion booleana simplificada para Y se obtiene entonces mediante el mapa.

$$Y = DG + G'y$$

	D G			
y	00	01	11	10
0	0	0	1	0
1	1	0	1	1

(a) Y = DG + G'y

	D G			
y	00	01	11	10
0	0	0	1	0
1	1	0	1	1

(b) Q = Y

FIG. 5.44 TABLA DE TRANSICION Y MAPA DE SALIDA DE UN SEGURO CON COMPUERTAS.

Hay dos salidas no importa en la tabla de flujo reducida final. Si se asignan valores a la salida como se muestra en la figura 5.44(b), es posible hacer la salida Q igual a la funcion de excitacion Y. Si se asignan los otros valores posibles a las salidas no importa, puede hacerse la salida Q igual a y. En cualquier caso, el diagrama logico del seguro con compuerta es como se ilustra en la figura 5.45.

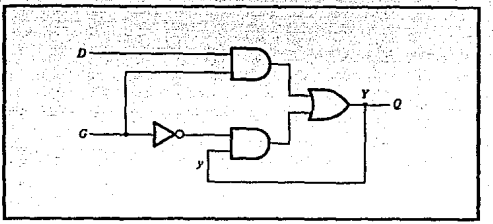


FIG. 5.45 DIAGRAMA LOGICO DE SEGURO CON COMPUERTAS.

5.3.2.5 ASIGNACION DE SALIDAS A LOS ESTADOS INESTABLES.

Los estados estables en una tabla de flujo tienen valores específicos de salida asociados con ellos. Los estados inestables tienen anotaciones de salida sin especificacion designados por un renglón. Los valores de salida para los estados inestables deben escogerse de modo que no ocurran salidas falsas momentáneas cuando el circuito cambia entre estados estables. Esto significa que si una variable de salida no se supone que cambie como resultado de una transición, estonces un estado inestable que es un estado transitorio entre dos estados estables debe tener el mismo valor de salida que los estados estables. Considerese, por ejemplo, la tabla de flujo en la figura 5.15(a). Una transicion del estado estable a al estado estable b pasa a traves del estado inestable b. Si la salida asignada al estado inestable b es un 1, entonces aparecera un corto pulso momentaneo en la salida conforme el circuito cambia desde una salida de 0 en el estado a a una salida de 1 para el inestable b y regresa a 0 cuando el circuito alcanza el estado estable b. Por consiguiente, la salida correspondiente al estado inestable b debe especificarse como 0 para evitar una salida momentanea falsa.

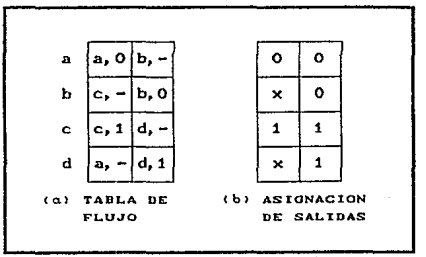
Si una variable de salida tiene que cambiar de valor como resultado de un cambio de estado, entonces esta variable se asigna a una condicion no importa. Por ejemplo, la transicion del estado estable b al estado estable c en la figura 5.46(a) cambia la salida desde 0 a 1. Si un 0 se introduce como el valor de

salida para el inestable c, entonces el cambio en la variable de salida no tendrá lugar hasta el final de la transición. Si se introduce un 1, el cambio tendrá lugar al inicio de la transición. Ya que no hay diferencia cuando ocurre el cambio de salida, se coloca una anotación no importa para la salida asociada con el estado inestable c. En la figura 5.46(b) se ilustra la asignación de salida para la tabla de flujo. Demuestra las cuatro combinaciones posibles en el cambio en la salida que puede ocurrir. El procedimiento para hacer la asignación a las salidas asociadas con los estados inestables puede resumirse como sigue:

1. Asígnese un 0 a una variable de salida asociada con un estado estable que es un estado transitorio entre dos estados estables que tienen un 0 en la variable de salida correspondiente.

2. Asígnese un 1 a un avariable de salida asociada con un estado inestable que es un estado transitorio entre dos estados estables que tienen un 1 en la variable de salida correspondiente.

3. Asígnese una condición no importa a una variable de salida asociada con un estado inestable que es un estado transitorio entre dos estados estables que tienen valores diferentes (0 y 1 o 1 y 0) en la variable de salida correspondiente.



a	a, 0	b, -
b	c, -	b, 0
c	c, 1	d, -
d	a, -	d, 1

(a) TABLA DE FLUJO

0	0
x	0
1	1
x	1

(b) ASIGNACION DE SALIDAS

FIG. 5.46 ASIGNACION DE VALORES DE SALIDA A ESTADOS INESTABLES.

5.3.2.6 RESUMEN DEL PROCEDIMIENTO DE DISENO.

El diseño de circuitos secuenciales asíncronos puede llevarse a cabo utilizando el procedimiento ilustrado en el ejemplo anterior. Algunos de los pasos de diseño requieren elaboracion adicional y se axplican en las siguientes secciones. Los pasos de pocedimiento son como sigue:

1. Obténgase una tabla de flujo primitiva mediante las especificaciones dadas de diseño. Esta es la parte mas difícil del diseño porque es necesario usar la intuición y la experiencia para llegar a la interpretación correcta de las especificaciones del problema.

2. Redúzcase la tabla de flujo fusionando renglones en la tabla de flujo primitiva.

3. Asígnese variables binarias de estado a cada ranglón de la tabla de flujo reducida para obtener la tabla de transicion.

4. Asígnese valores de salidas a los guiones asociados con los estados inestables para obtener los mapas de salida. Este procedimiento se explico con anterioridad.

5. Simplifíquense las funciones booleanas de las variables de excitación y la salida y se dibuja el diagrama lógico. El diagrama logico puede dibujarse usando seguros SR.

5.3.2.7 TABLA DE IMPLICACION.

El procedimiento para reducir el número de estados internos en el circuito secuencial asíncrono se asemeja al procedimiento que se utilizó para los circuitos síncronos.

El procedimiento de reducción de estado para tablas de estado completamente especificadas se basa en el algoritmo de que dos estados en una tabla de estado pueden combinarse en uno si puede

demostrarse que son equivalentes. Dos estados son equivalentes si para cada entrada posible dan exactamente la misma salida y pasan a los mismos estados siguientes o estados siguientes equivalentes. Hay ocaciones en que un par de estados no tienen los mismos estados siguientes pero sin embargo pasan a los estados siguientes equivalentes. Considerese, por ejemplo, la tabla de estado que se muestra en la Tabla 5.16. Los estados presentes a y b tienen la misma salida para la misma entrada. Los estados siguientes son c y d para x = 0 y b y a para x = 1. Si puede demostrarse que el par de estados (c, d) son equivalentes, entonces el par de estados (a, b) tambien serán equivalentes porque tienen los mismos o equievalentes estados siguientes. Cuando esta relación existe, se dice que (a, b) implica (c, d). En forma similar, mediante los ultimos dos renglones de la Tabla 5.16 se encuentra que el par de estados (c, d) implica el par de estados (a, b). La caracteristica de los estados equivalentes es que si (a, b) implica (c, d) y (c, d) implica (a, b), entonces ambos pares de estados son equivalentes; esto es, a y b son equivalentes lo mismo que c y d. Como consecuencia, los cuatro renglones de la Tabla 5.16 puede reducirse a dos renglones por la combinación de a y b en un estado y c y d en un segundo estado.

ESTADO PRESENTE	ESTADO SIGUIENTE		SALIDA	
	x = 0	x = 1	x = 0	x = 1
a	c	b	0	1
b	d	a	0	1
c	a	d	1	0
d	b	d	1	0

TABLA 5.16 TABLA DE ESTADOS PARA DEMOSTRAR LOS ESTADOS EQUIVALENTES.

La verificación de cada par de estados para equivalencia posible en una tabla con gran numero de estado puede hacerse en forma sistematica mediante una tabla de implicacion. La tabla de implicacion es una tabla que consta de casillas, una para cada par posible de estados, que proporciona espacios para listar cualesquiera estados implicados posibles. Por el uso juicioso de la tabla, es posible determinar todos los pares de estados equivalentes. La tabla de estados en la Tabla 5.17 se utilizara

para ilustrar este procedimiento. La tabla de implicación se muestra en la figura 5.47. En el lado izquierdo a lo largo de la vertical se listan todos los estados definidos en la tabla de estado, excepto el primero, y a través de la parte inferior en sentido horizontal se listan todos los estados excepto el ultimo. El resultado es un despliegue de todas las combinaciones posibles de dos estados con una casilla colocada en la inteseccion de un renglón y una culumna donde los dos estados pueden probarse para equivalencia.

ESTADO PRESENTE	ESTADO SIGUIENTE		SALIDA	
	x = 0	x = 1	x = 0	x = 1
a	d	b	0	0
b	e	a	0	0
c	g	f	0	1
d	a	d	1	0
e	a	d	1	0
f	c	b	0	0
g	a	e	1	0

TABLA 5.17 TABLA DE ESTADOS QUE SE REDUCE.

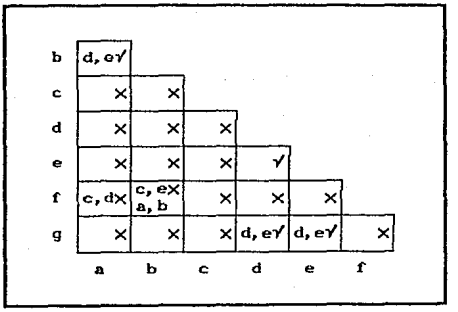


FIG. 5.47 TABLA DE IMPLICACION.

Dos estados que no son equivalentes se marcan con una cruz (x) en la casilla correspondiente, en tanto que su equivalencia se registra con una marca de verificación (✓). Algunas de las casillas tienen anotaciones de estados implicados que deben investigarse adicionalmente para determinar si son equivalentes o no. El procedimiento paso a paso de llenar las casillas es como sigue. Primero se coloca una cruz en cualquier estado correspondiente a un par de estados cuyas salidas no son iguales para cada entrada. En este caso el estado c tiene una salida diferente a cualquier otro estado, de modo que se coloca una cruz en las dos casillas del renglón c y las cuatro casillas de la columna c. Hay otras nueve casillas en esta categoría en la tabla de implicación.

A continuación, se anotan en las casillas restantes los pares de estados que están implicados por el par de estados que representan las casillas. Se hace esto principiando desde la casilla superior en la columna izquierda y se pasa abajo y entonces se puede con la siguiente columna a la derecha. Mediante la tabla de estado se ve que el par (a, b) implica (d, e) de modo que (d, e) se registra en la casilla definida por la columna a y el renglon b. Se procede de esta manera hasta que se completa la tabla entera. Obsérvese que los estados (d, e) son equivalentes ya que pasan al mismo estado siguiente y tienen la misma salida. Por tanto, se registra una marca de verificación en la casilla definida por la columna d y el renglón e, indicando que los dos estados son equivalentes e independientes de cualquier par implicado.

El siguiente paso es hacer repasos sucesivos a través de la tabla para determinar cuándo cualesquiera casillas adicionales deben marcarse con una cruz. Una casilla en la tabla se cruza si contiene cuando menos un par implicado que no es equivalente. Por ejemplo, la casilla definida por a y f se marca con una cruz contigua a c, d porque el par (c, d) define una casilla que contiene una cruz. Este procedimiento se repite hasta que no pueden cruzarse casillas adicionales. Por último, todas las casillas que no tienen cruces se registran con marcas de verificación. Estas casillas definen pares de estados equivalentes. En este ejemplo, los estados euquivalentes son:

(a, b) (d, e) (d, g) (e, g)

Ahora se combinan pares de estados en grupos más grandes de

estados equivalentes. Los últimos tres pares pueden combinarse en un conjunto de tres estados equivalentes (d, e, g) ya que cada uno de los estados en el grupo es equivalente a los otros dos. La aparicion final de los estados consta de los estados equivalentes encontrados mediante la tabla de implicación junto con todos los estados remanentes en la tabla de estado, los cuales no son equivalentes a cualquier otro estado.

(a, b) (c) (d, e, g) (f)

Esto significa que la Tabla 5.17 puede reducirse desde siete estados a cuatro estados, uno para cada miembro de la particion anterior. La tabla reducida se obtiene reemplazando el estado b por a y los estados e y g por d. La tabla de estado reducida se muestra en la Tabla 5.18.

ESTADO PRESENTE	ESTADO SIGUIENTE		SALIDA	
	x = 0	x = 1	x = 0	x = 1
a	d	a	0	0
c	d	f	0	1
d	a	d	1	0
f	c	a	0	0

TABLA 5.18 TABLA DE ESTADOS REDUCIDA.

5.3.2.8 FUSION DE LA TABLA DE FLUJO.

Hay ocasiones en las que la tabla de estado para un circuito secuencial esta especificada de manera incompleta. Esto sucede cuando ciertas combinaciones de entradas o secuencias de entrada no pueden ocurrir jamas por restricciones externas o internas. En tal caso, los estados siguientes y salidas que podrian haber ocurrido si todas las entradas fueran posibles nunca se alcanzan y se consideran como condiciones no importa. Aun cuando los circuitos secuenciales sincronos pueden representarse algunas veces con tablas de estado especificadas en forma incompleta, el

interés aquí es con los circuitos secuenciales asíncronos donde la tabla de flujo primitiva siempre esta especificada de manera incompleta.

Los estados especificados en forma incompleta pueden combinarse para reducir el numero de estados en la tabla de flujo. Dichos estados no pueden llamarse equivalentes porque la definicion formal de equilvalencia requiere que todas las salidas y los estados siguientes esten especificadas para todas las entradas. En lugar de esto, dos estados especificados enforma incompleta pueden combinarse y se dice que son compatibles. Dos estados son compatibles si para cada entrada posible tienen la misma salida siempre que se especifique y sus estados siguientes son compatibles siempre que esten especificados. Todas las condiciones no importa marcadas con guiones no tienen efecto cuando se buscan estados compatibles ya que representan condiciones no especificadas.

El proceso que debe aplicarse con objeto de encontrar un grupo adecuado de compatibles para el proposito de fusionar una tabla de flujo puede dividirse en tres pasos de procedimiento.

1. Se determinan todos los pares compatibles por el uso de la tabla de implicación.

2. Se obtienen los compatibles maximales usando un diagrama de fusión.

3. Se encuentra una colección mínima de compatibles que cubren todos los estados y es cerrada.

La colección mínima de compatibles se utiliza entonces para fusionar los renglones de la tabla de flujo. Ahora se procederá a mostrar y explicar los tres pasos de procedimiento usando la tabla de flujo primitiva del ejemplo de diseño en la seccion anterior.

5.3.2.9 PARES COMPATIBLES.

El procedimiento para encontrar pares compatibles se ilustra en la figura 5.48. La tabla de flujo primitiva (a) es la misma en

que se muestra en la figura 5.42. Las anotaciones en cada casilla representan el estado siguiente y la salida. Los guiones representan los estados especificados o salidas. La tabla de implicacion se usa para encontrar estados compatibles presisamente como se utiliza para encontrar estados equivalentes en el caso completamente especificado. La unica diferencia es que cuando se comparan los renglones, se tiene la libertad de ajustar los guiones para llegar a cualquier condicion deseada.

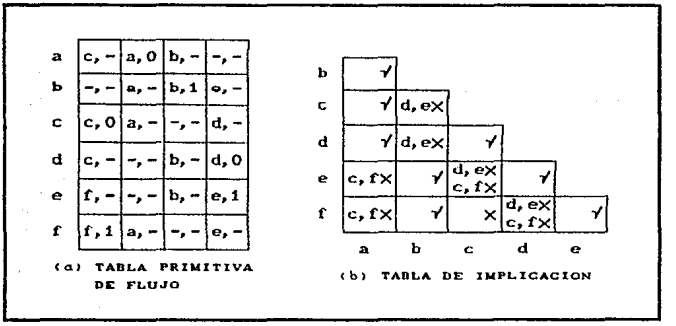


FIG. 5.40 TABLAS DE FLUJO E IMPLICACION.

Dos estados son compatibles si en cada columna de los renglones correspondientes en la tabla de flujo hay estados identicos o compatibles y si no hay conflicto en los valores de salida. Por ejemplo en la tabla de flujo se encuentra que los renglones a y b son compatibles, pero los renglones a y f seran compatibles sólo si c y f son compatibles. Sin embargo, los renglones c y f no son compatibles ya que tienen diferentes salidas en la primera columna. Esta informacion se registra en la tabla de implicacion. Una marca de verificacion designa una casilla cuyo par de estado son compatibles. Los estados que no son compatibles se marcan con una cruz. Las casillas remanentes se registran con los pares implicados que necesitan investigacion adicional.

Ya que se ha llenado la tabla inicial de implicación, se

explora otra vez para eliminar con cruces las casillas cuyos estados implicados no son compatibles. Las casillas restantes que contienen marcas de verificacion definen los pares compatibles. En el ejemplo en la figura 5.48 los pares compatibles son:

(a, b) (a, c) (a, d) (b, e) (b, f) (c, d) (e, f)

5.3.2.10 COMPATIBLES MAXIMALES.

Ya que se encontraron todos los pares compatibles, el paso siguiente es encontrar conjuntos más grandes de estados que son compatibles. El compatible maximal es un grupo de compatibles que copntienen todas las combinaciones posibles de los estados compatibles. El compatible maximal puede obtenerse mediante un diagrama de fusión como se muestra en la figura 5.49. El

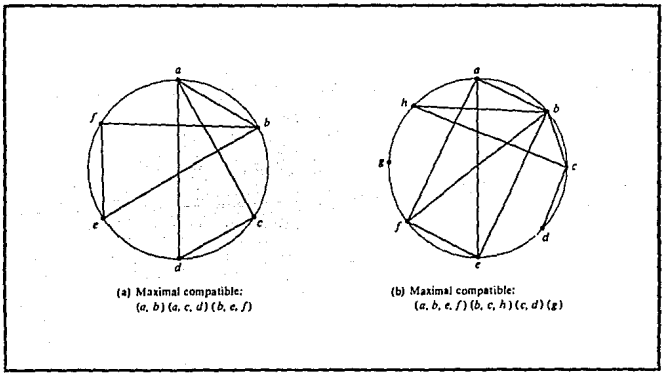


FIG. 5.49 DIAGRAMAS DE FUSION.

diagrama de fusión es una gráfica en la cual cada estado se representa por un punto colocado a lo largo de la circunferencia

de un círculo. Las líneas se dibujan entre dos puntos cualesquiera correspondientes que forman un par compatible. Todos los compatibles posibles pueden obtenerse mediante el diagrama de fusión observando los patrones geométricos en los cuales los estados están conectados uno con otro. Un punto aislado representa un estado que no es compatible con cualquier otro estado. Una línea representa un par compatible. Un triángulo consta de un compatible con tres estados. Un compatible de n estados se representa en el diagrama de fusión por un polígono de n lados con todas sus diagonales conectadas.

El diagrama de fusión en la figura 5.49(a) se obtiene mediante la lista de pares compatibles derivada mediante la tabla de implicación en la figura 5.48. Hay siete líneas rectas que conectan los puntos, para cada par compatible. Las líneas forman un patrón geométrico que consta de dos triángulos conectando (a, c, d) y (b, e, f) y una línea (a, b). Los compatibles maximales son:

(a,b) (a, c, d) (b, e, f)

En la figura 5.49(b) se muestra el diagrama de fusión para una tabla de flujo de ocho estados. Los patrones geométricos son un rectángulo con sus dos diagonales conectadas para formar el compatible de cuatro estados (a, b, e, f). Un triángulo (b, c, h), una línea (c, d) y un estado único g que no es compatible con cualquier otro estado. Los compatibles maximales son :

(a, b, e, f) (b, c, h) (c, d) (g)

El conjunto campatible maximal puede usarse para fusionar la tabla de flujo asignando un ranglón en la tabla reducida a cada miembro del conjunto. Sin embargo, con bastante frecuencia los compatibles maximales no constituyen necesariamente el conjunto de compatibles que es mínimo. En muchos casos, es posible encontrar una colección mas pequeña de compatibles que satisfaga la condición para fusionar renglones.

5 . 3 . 2 . 11 CONDICION DE COBERTURA CERRADA.

La condición que debe satisfacerse para la fusión de renglón es que el conjunto de compatibles elegido debe cubrir todos los

estados y debe ser cerrado. El conjunto cubrirá todos los estados si incluye todos los estados de la tabla de estado original. Se satisface la condición de cierre si no hay estados implicados o si los estados implicados se incluyen dentro del conjunto. Un conjunto cerrado de compatibles que cubre todos los estados se denomina cobertura cerrada. La condicion de cobertura cerrada se explicara mediante dos ejemplos.

Considérense los compatibles maximales en la figura 5.49(a). Si se elimina (a, b), queda un conjunto de dos campatibles:

(a, c, d) (b, e, f)

Todos los seis estados de la tabla de flujo en la figura 5.48 están incluidos en este conjunto. Esto satisface la condición de cobertura. No hay estados implicados para (a, c), (a, d), (c, d) (b, e), (b, f) y (e, f), como se ve mediante la tabla de implicación en la figura 5.48(b), de modo que la condición de cierre también se satisface. Por tanto, la tabla de flujo primitiva puede fusionarse en dos renglones, uno para cada uno de los compatibles. La construcción detallada de la tabla reducida para este ejemplo particular se hizo anteriormente y se ilustra en la figura 5.43(b).

El segundo ejemplo es de una tabla de flujo primitiva (que no se muestra) cuya tabla de implicación esta dada en la figura 5.50(a). Los pares compatibles derivados mediante la tabla de implicacion son:

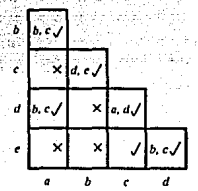
(a, b) (a, d) (b, c) (c, d) (c, e) (d, e)

Mediante el diagrama de fusión en la figura 5.50(b) se determinan los compatibles maximales:

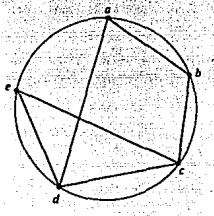
(a, b) (a, d) (b, c) (c, d, e)

Si se escogen los dos compatibles

(a, b) (c, d, e)



(a) Tabla de implicación



(b) Diagrama de fusión

Compatibles	(a, b)	(a, d)	(b, c)	(c, d, e)
Estados implicados	(b, c)	(b, c)	(d, e)	(a, d) (b, c)

(c) Tabla de cierre

FIG. 5.50 ELECCION DE UN CONJUNTO DE COMPATIBLES.

el conjunto cubrirá todos los cinco estados de la tabla original. La condición de cierre puede verificarse mediante una tabla de cierre como se muestra en la figura 5.50(c). Los pares implicados listados para cada compatible se toma en forma directa a partir de la tabla de implicación. Los estados implicados para (a, b) son (b, c). Pero (b, c) no se incluye en el conjunto elegido de (a, b) (c, d, e), de modo que el conjunto de compatibles no es cerrado. Un conjunto de compatibles no es cerrado. Un conjunto de compatibles que satisface la condición de cobertura cerrada es :

(a, d) (b, c) (c, d, e).

El conjunto está cobierto ya que contiene todos los cinco estados. Obsérvese que el mismo estado puede repetirse más de una vez. La condición de cierre se satisface porque los estados implicados son (b, c) (d, e) y (a, d), los cuales están incluidos en el conjunto. La tabla de flujo original (que no se muestra) puede reducirse desde cinco renglones a tres renglones por la fusión de los renglonesa y b, b y c, y c, d y e. Obsérvese que una elección satisfactoria alterna de compatibles de cobertura cerrada sería (a, b) (b, c) (d, e). En general, puede haber más de una forma posible de fusionar renglones cuando se reduce una tabla de flujo primitiva.

5.3.3 ASIGNACION DE ESTADO LIBRE DE CARRERAS.

Ya que se ha derivado una tabla de flujo para un circuito secuencial asíncrono, el paso siguiente en el diseño es asignar variables binarias a cada estado estable. Esta asignación resulta en la transformación de la tabla de flujo en su tabla de transición equivalente. El objeto primario al escoger una asignación binaria de estado apropiada es prevenir carreras críticas. El problema de las carreras críticas mostró anteriormente junto con la figura 5.39.

Las carreras críticas pueden evitarse haciendo una asignación binaria de estado en tal forma que sólo una variable cambie en cualquier momento dado cuando ocurre una transición de estado en la tabla de flujo. Para lograr esto, es necesario que los estados entre los cuales ocurren las transiciones reciban asignaciones adyacentes. Dos valores binarios se dice que son adyacentes si difieren sólo en una variable. Por ejemplo, 010 y 011 son adyacentes ya que sólo difieren en el tercer dígito.

Con el objeto de asegurar una tabla de transición no tenga carreras críticas, es necesario probar cada transición posible entre dos estados estables y verificar que las variables de estado cambien una a la vez. Esto es un proceso tedioso, especialmente cuando hay muchos renglones y columnas en la tabla. Para simplificar las cosas, se explicará primero el procedimiento de asignación binaria de estado pasando a través de ejemplos solo con tres o cuatro renglones en la tabla de flujo. Estos ejemplos demostrarán el procedimiento general que debe seguirse para asegurar una asignación de estado libre de carreras. El procedimiento puede aplicarse entonces a tablas de flujo con

cualquier número de renglones y columnas.

5.3.3.1 TABLA DE FLUJO CON TRES RENGLONES.

La asignación de una variable binaria única a una tabla de flujo con dos renglones no impone problemas de carrera crítica. Una tabla de flujo con tres renglones requiere una asignación de dos variables binarias. La asignación de valores binarios a los estados estables pueden provocar carreras críticas si no se hace de manera adecuada. Considérese, por ejemplo, la tabla de flujo reducida de la figura 5.51(a). Las salidas se han omitido en la tabla con objeto de simplificarla. La inspección del renglón a revela que hay una transición desde el estado a al estado b en la columna 01 y del estado a al estado c en la columna 11. Esta información se transfiere a un diagrama de transición, como se muestra en la figura 5.51(b). Las líneas dirigidas desde a a b y desde a a c representan las dos transiciones que acaban de mencionarse. En forma similar, las transiciones desde los otros dos renglones se representan por líneas dirigidas en el diagrama de transición. El diagrama de transición es una representación pictórica de todas las transiciones requeridas entre renglones.

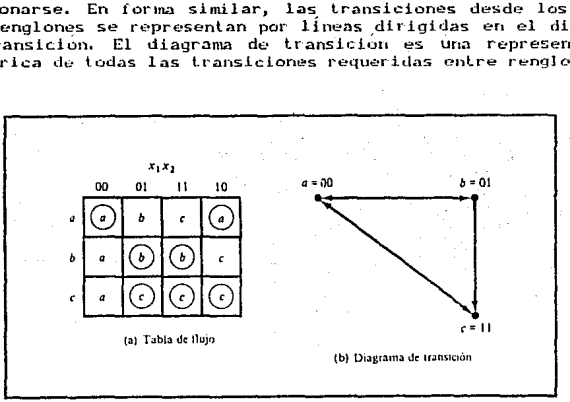
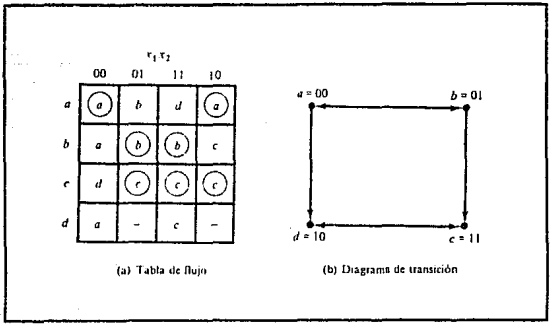


FIG. 5.51 EJEMPLO DE TABLA DE FLUJO CON TRES RENGLONES.

Para evitar carreras críticas, debe encontrarse una asignación de estado binario de modo que solo una variable binaria cambia

durante cada transición de estado. Un intento de encontrar dicha asignación se muestra en el diagrama de transición. Al estado a se la asigna el binario 00, y al estado c se le asigna el binario 11. Esta asignación provocará una carrera crítica durante la transición desde a hasta c, ya que hay dos cambios en las variables binarias de estado. Obsérvese que la transición desde c hasta a también causa una condición de carrera, pero no es crítica.

Una asignación libre de carrera puede obtenerse si se agrega un renglón adicional a la tabla de flujo. El uso de un cuarto renglon aumenta el número de variables de estado binarias, pero permite la formación de ciclos entre dos estados estables. Considérese la tabla de flujo modificada en la figura 5.52. Los primeros tres renglones representan las mismas condiciones en la tabla de tres renglones original. Al cuarto renglón, etiquetado d, se le asigna el valor binario de 10, el cual es adyacente tanto a a como a c. La transición desde a hasta c debe pasar ahora a través de d, con el resultado de que el cambio de variables binarias desde a = 00 a d = 10 a c = 11, evitando así una carrera crítica. Esto se lleva a cabo cambiando el renglon a, columna 11, a d y el renglon d, columna 11, a c. En forma similar, la transición desde c hasta a se muestra que va a través del estado inestable d aun cuando la columna 00 constituye una carrera no crítica.



(a) Tabla de flujo (b) Diagrama de transición

FIG. 5.52 TABLA DE FLUJO CON UN RENGLON ADICIONAL.

La tabla de transición correspondiente a la tabla de flujo con la asignación binaria de estado indicada se muestra en la figura 5.53. Los dos guiones en el renglón d representan estados no especificados que pueden considerarse como condiciones no importa. Sin embargo, debe tenerse cuidado de no asignar 10 a estas casillas con objeto de evitar la posibilidad de un estado estable no deseado establecido en el cuarto renglón.

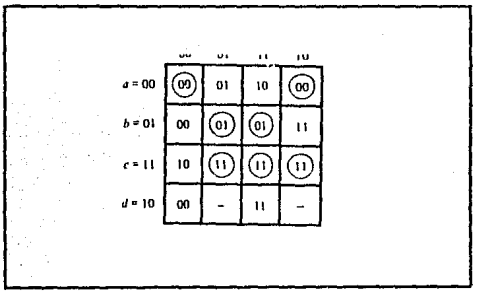


FIG. 5.53 TABLA DE TRANSICION.

Este ejemplo demuestra el uso de un renglón adicional en la tabla de flujo con el propósito de lograr una asignación libre de carrera. El renglón adicional no está asignado a ningún estado estable específico, pero en lugar de esto se usa para convertir una carrera crítica en un ciclo que pasa a través de transiciones adyacentes entre dos estados estables. Algunas veces, es posible que un renglón adicional solo no sea suficiente para evitar las carreras críticas, y puede ser necesario añadir dos o más renglones adicionales en la tabla de flujo. Esto se demuestra en el siguiente ejemplo.

5.3.3.2 TABLA DE FLUJO CON CUATRO RENGLONES.

Una tabla de flujo con cuatro renglones requiere un mínimo de dos variables de estado. Aun cuando la asignación libre de carrera algunas veces es posible con solo dos variables binarias

de estado, en muchos casos el requisito de renglones adicionales para evitar las carreras críticas dictará el uso de tres variables binarias de estado. Por ejemplo, considerese la tabla de flujo y su diagrama correspondiente de transición que se muestra en la figura 5.54. Si no hay transiciones en la dirección diagonal (desde b a d o desde c a a), sería posible encontrar una asignación adyacente para las cuatro transiciones remanentes. Con una o dos transiciones diagonales no hay forma de asignar dos variables binarias que satisfagan el requisito de adyacencia. Por eso, son necesarias cuando menos tres variables binarias de estado.

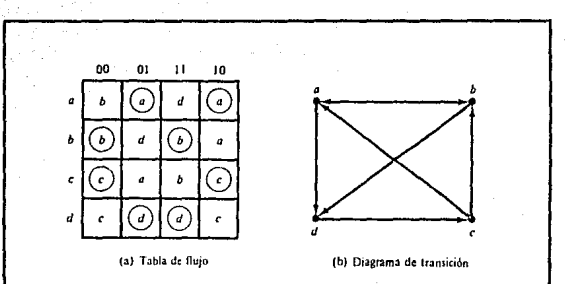


FIG. 5.54 EJEMPLO DE TABLA DE FLUJO CON CUATRO RENGLONES.

En la figura 5.55 se muestra un mapa de asignación de estado que es adecuado para cualquier tabla de flujo de cuatro renglones. Los estados a, b, c y d son los estados originales, y e, f y g son los estados adicionales. Los estados colocados en casillas adyacentes en el mapa tendrán asignaciones adyacentes. El estado b tiene la asignación binaria 001 y es adyacente a los otros tres estados originales. La transición desde a a d debe dirigirse a través del estado adicional e para producir un ciclo de modo que sólo una variable binaria cambie a la vez. En forma similar, la transición desde c a a se dirige a través de g y la transición de d a c pasa a través de f. Por el uso de la asignación dada por el mapa, la tabla de cuatro renglones puede expanderse a una tabla de siete renglones que está libre de carreras críticas, como se muestra en la figura 5.56. Obsérvese que aunque la tabla de flujo tiene siete renglones, hay sólo

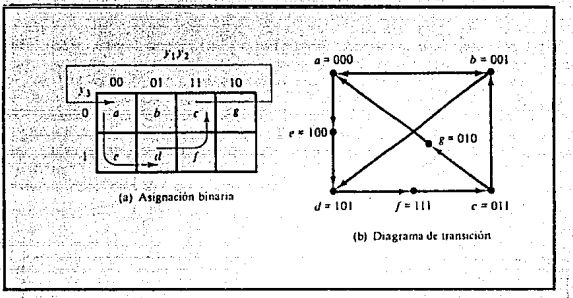


FIG. 5.55 ELECCION DE RENGLONES ADICIONALES PARA LA TABLA DE FLUJO.

	00	01	11	10
000 = a	b	(a)	e	(a)
001 = b	(b)	d	(b)	a
011 = c	(c)	g	b	(c)
010 = g	–	a	–	–
110	–	–	–	–
111 = f	c	–	–	c
101 = d	f	(d)	(d)	f
100 = e	–	–	d	–

FIG. 5.56 ASIGNACION DE ESTADOS A LA TABLA DE FLUJO MODIFICADA.

cuatro estados estables. Los estados que no están dentro de círculos en los tres renglones adicionales están ahí sólo para proporcionar una transición libre de carrera entre los estados estables.

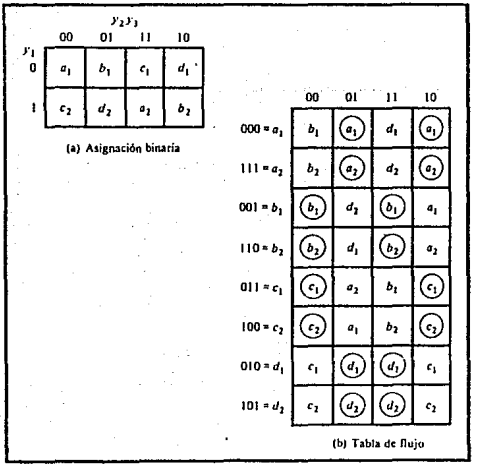
Este ejemplo demuestra una forma posible de seleccionar renglones adicionales en una tabla de flujo con objeto de lograr una asignación libre de carreras. Un mapa de asignación de estado similar al que se usa en la figura 5.55(a) puede ser de ayuda en la mayoría de los casos. Algunas veces es posible aprovechar anotaciones sin especificar en la tabla de flujo. En lugar de añadir renglones a la tabla, puede ser posible eliminar las carreras críticas al dirigir algunas de las transiciones de estado a través de anotaciones no importa. La asignación real se hace por prueba y error hasta que se encuentra una asignación satisfactoria que resuelve todas las carreras críticas.

5.3.3.3 METODO DE RENGLONES MULTIPLES

El método para lograr asignación de estado libre de carrera por la adición de renglones adicionales en la tabla de flujo, como se demostró en los ejemplos anteriores, algunas veces se conoce como el método de renglón compartido. Hay un segundo método que no es tan eficiente pero es más fácil de aplicar, llamado método de renglón múltiple. En la asignación de renglón múltiple, cada estado en la tabla de flujo original se reemplaza por dos o más combinaciones de variables de estado. El mapa de asignación de estado en la figura 5.57(a) muestra una asignación de renglón múltiple que puede usarse con cualquier tabla de flujo de cuatro renglones. Hay dos variables binarias de estado para cada estado estable, cada una siendo el complemento lógico una de otra. Por ejemplo, el estado original a se reemplaza con dos estados equivalentes $a_1 = 000$ y $a_2 = 111$. Los valores de salida, que no se muestran aquí, deben ser los mismos en a_1 y a_2. Obsérvese que a_1 es adyacente a b_1, c_2 y d_1, y a_2 es adyacente a c_1, b_2, y d_2, y en forma similar cada estado es adyacente a tres estados de diferente designación alfabética. El comportamiento del circuito es el mismo aunque el estado interno sea a_1 o a_2, y así sucesivamente para los otros estados.

En la figura 5.57(b) se muestra la asignación de renglones múltiples para la tabla de flujo original en la figura 5.54(a). La tabla expandida se forma reemplazando cada renglón de la tabla

original con dos renglones. El renglón b se reemplaza por los renglones b_1 y b_2 y el estado estable b se anota en las columnas 00 y 11 en los renglones b_1 al igual que b_2. Después de que se han anotado todos los estados estables, los estados inestables se llenan con referencia a la asignación especificada en el mapa de la parte (a). Cuando se escoge el estado siguiente para un estado presente dado, un estado adyacente al estado presente se selecciona del mapa. En la tabla original, los estados siguientes de b son a y d para las entradas 10 y 01, respectivamente. En la tabla con expansión, los estados siguientes para b_1 son a_1 y d_2 ya que esos son los estados adyacentes a b_1. En forma similar, los estados siguientes para b_2 son a_2 y d_1 porque son adyacentes a b_2.



(a) Asignación binaria

y_1 \ y_2y_3	00	01	11	10
0	a_1	b_1	c_1	d_1
1	c_2	d_2	a_2	b_2

(b) Tabla de flujo

	00	01	11	10
$000 = a_1$	b_1	(a_1)	d_1	(a_1)
$111 = a_2$	b_2	(a_2)	d_2	(a_2)
$001 = b_1$	(b_1)	d_2	(b_1)	a_1
$110 = b_2$	(b_2)	d_1	(b_2)	a_2
$011 = c_1$	(c_1)	a_2	b_1	(c_1)
$100 = c_2$	(c_2)	a_1	b_2	(c_2)
$010 = d_1$	c_1	(d_1)	(d_1)	c_1
$101 = d_2$	c_2	(d_2)	(d_2)	c_2

FIG. 5.57 ASIGNACION DE RENGLONES MULTIPLES.

En la asignación de renglones múltiples, el cambio de un estado variable a otro siempre provocará un cambio sólo de una variable binaria de estado. Cada estado estable tiene dos

asignaciones binarias exactamente con la misma salida. En cualquier momento, sólo una de las asignaciones esta en uso. Por ejemplo, si se principia con el estado a_1 y la entrada 01 y entonces se cambia la entrada a 11, 01, 00 y se regresa a 01, la secuencia de los estados internos sera a_1, d_1, c_1 y a_2. Aunque el circuito principia en el estado a_1 y termina en el estado a_2, en lo que respecta a la relación de entrada-salida, los dos estados a_1 y a_2 son equivalentes al estado a de la tabla de flujo original.

5.3.4 RIESGOS.

Cuando se diseñan circuitos secuenciales asíncronos, debe tenerse cuidado de cumplir con ciertas restricciones así como tomar precauciones para asegurar la operacción apropiada. El circuito debe operarse en el modo fundamental solo con una entrada cambiando en cualquier momento y debe estar libre de carreras críticas. Ademas, hay un fenomeno mas, denominado riesgo, que puede provocar que el circuito funcione mal. Los riesgos son transitorios indeseables con interrupciones que pueden aparecer a la salida de un circuito porque las diferentes trayectorias exhiben distintos retardos de propagación. Los riesgos ocurren en circuitos combinacionales, donde pueden causar un valor temporal falso de salida. Cuando ocurre esta condición en los circuitos secuenciales asíncronos, puede resultar en una transición a un estado estable equivocado. En consecuencia, es necesario verificar los peligros posibles y determinar cuando pueden ocasionar operaciones inadecuadas. Deben tomarse medidas para eleminar su efecto.

5.3.4.1 RIESGOS EN LOS CIRCUITOS COMBINACIONALES.

Un riesgo es una condición en la que un solo cambio de variable produce un cambio momentáneo en la salida cuando no debe ocurrir cambio en la salida. El circuito en la figura 5.58(a) demuestra la ocurrencia de un riesgo. Se supone que todas las tres entradas son iguales a 1 al inicio. Esto provoca que la salida de la compuerta 1 sea 1, que la de la compuerta 2 sea 0, y que la salida del circuito sea igual a 1. Ahora considerese un cambio de x2 desde 1 a 0. La salida de la compuerta 1 cambia a 0 y la de la compuerta 2 cambia a 1, dejando la salida en 1. Sin embargo, en forma momentánea la salida puede pasar a 0 si el

retardo de propagación a través del inversor se toma en consideración. El retardo en el inversor puede causar que la salida de la compuerta 1 cambie a 0 antes de que la salida de la compuerta 2 cambie a 1. En ese caso, ambas entradas de la compuerta 3 son momentaneamente iguales a 0, provocando que la salida pase a 0 por el corto intervalo de tiempo en que la señal de entrada desde x2 se retrasa mientras se propaga a través del circuito inversor.

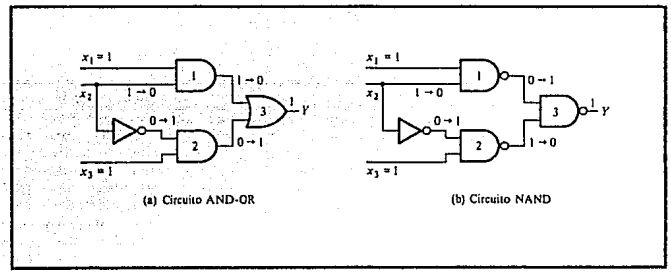


FIG. 5.58 CIRCUITOS CON RIESGOS.

El circuito que se muestra en la figura 5.58(b) es una implantación NAND de la misma función booleana. Tiene un peligro por la misma función booleana. Tiene un peligro por la misma razón. Ya que las compuertas 1 y 2 son compuertas NAND, sus salidas son el complemento de las salidas de las compuertas correspondiantes AND. Cuando x2 cambia desde 1 a 0, ambas entradas de la compuerta 3 pueden ser iguales a 1, causando que la salida se produzca un cambio momentaneo de 0 cuando debería permanecer en 1.

Los dos circuitos que se muestran en la figura 5.58 implementan la función booleana en suma de productos.

$$Y = x_1 x_2 + x'_2 x_3$$

El tipo de implementación puede provocar que la salida pase a

0 cuando debe permanecer a 1. Si el circuito se implementa en producto de sumas

$$Y = (x_1 + x'_2)(x_2 + x_3)$$

entonces la salida puede ir en forma momentánea a 1 cuando debería permanecer en 0. El primer caso se conoce como peligro estático 1 y el segundo caso peligro estático 0. Un tercer tipo de peligro conocido como peligro dinámico causa que la salida cambie tres o más veces cuando debe cambiar de 1 a 0, o de 0 a 1. En la figura 5.59 se demuestran los tres tipos de peligro. Cuando un tipo se implementa en suma de productos con compuertas AND-OR o con compuertas NAND, la remoción del riesgo estático 1 garantiza que no ocurran riesgos estáticos 0 o riesgos dinámicos.

FIG. 5.59 TIPOS DE RIESGOS.

La ocurrencia del riesgo puede detectarse mediante la inspección del mapa del circuito particular. Como ilustración, considérese el mapa en la figura 5.60(a), que es una gráfica de la función implementada en la figura 5.58. El cambio en x_2 desde 1 a 0 mueve el circuito desde el mintérmino 111 al mintérmino 101. El peligro existe porque el cambio de entrada resulta en un término producto diferente que cubre los dos mintérminos. El mintérmino 111 se cubre por el término producto implementado en la compuerta 1, y el mintérmino 101 se cubre por el término producto implementado en la compuerta 2 en la figura 5.58. Siempre que el circuito debe moverse desde un término producto a otro, hay una posibilidad de un intervalo momentáneo cuando ningún término es igual a 1, dando lugar a una salida indeseable 0.

El remedio para eliminar un riesgo es encerrar los dos mintérminos en cuestión con otro término producto que translape ambos agrupamientos. Esto se muestra en el mapa en la figura

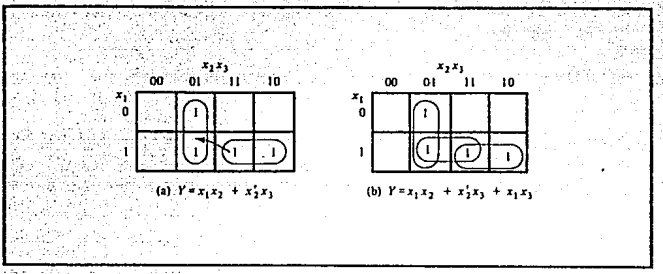


FIG. 5.60 MAPA PARA DEMOSTRAR UN RIESGO Y SU REMOCION.

5.60(b), donde los dos mintérminos que causan el peligro se combinan en un término producto. El circuito libre de riesgo obtenido por esta configuración se muestra en la figura 5.61. La compuerta adicional en el circuito genera el término producto x_1x_3. En general, los peligros en los circuitos combinacionales pueden eliminarse cubriendo cualesquiera dos mintérminos que puedan producir un peligro con un término producto común a ambos. La remoción de peligros requiere la condición de compuertas redundantes al circuito.

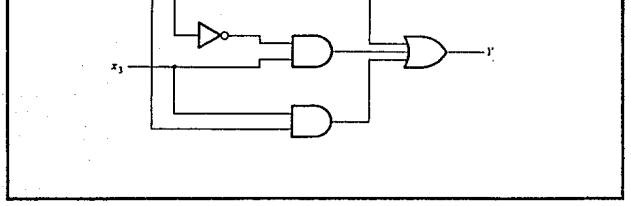


FIG. 5.61 CIRCUITO LIBRE DE RIESGO.

5.3.4.2 RIESGOS EN LOS CIRCUITOS SECUENCIALES.

En el diseño del circuito combinacional normal asociado con circuitos secuenciales síncronos, los riesgos no son de cuidado, ya que las señales momentáneas erróneas por lo general no causan dificultades. Sin embargo, si se retroalimenta una señal momentánea incorrecta en un circuito secuancial asíncrono, puede provocar que el circuito pase al estado estable equivocado. Esto se ilustra en el ejemplo que se muestra en la figura 5.62. Si el circuito está en el estado total estable $yx_1x_2 = 111$ y la entrada x_2 cambia desde 1 a 0, el siguiente estado total estable será 110. Sin embargo, debido al peligro, la salida Y puede pasar a 0 en forma momentánea. Si esta señal falsa se retroalimenta en la compuerta 2 antes que la salida del inversor pase a 1, la salida de la compuerta 2 permanecerá en 0 y el circuito cambiará al estado estable total incorrecto 010. Este mal funcionamiento puede eliminarse añadiendo una compuerta adicional como se hace en la figura 5.61.

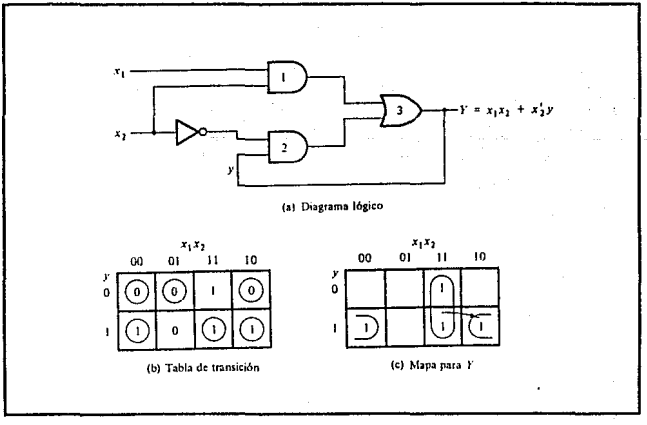


FIG. 5.62 RIESGO EN UN CIRCUITO SECUENCIAL ASINCRONO.

5.3.4.3 IMPLEMENTACION CON SEGUROS SR.

Otra forma de evitar riesgos estáticos en los circuitos secuenciales asíncronos es implementar el circuito con seguros SR. Una señal momentánea 0 aplicada a las entradas S o R de un seguro NOR no tendrá efecto en el estado del circuito. En forma similar, una señal momentánea 1 aplicada a las entradas S y R de un seguro NAND no tendrá efecto en el estado del seguro. En la figura 5.58(b) se observa que una suma en dos niveles de la expresión producto implementada con las compuertas NAND puede tener un peligro estático 1 si ambas entradas de la compuerta 3 pasan a 1, cambiando la salida desde 1 a 0 en forma momentánea. Pero si la compuerta 3 es parte de un seguro, la señal momentánea 1 no tendrá efecto en la salida porque llegará una tercera entrada a la compuerta desde el lado complementado del seguro, la cual será igual a 0 y por tanto mantendrá la salida en 1. Para aclarar esto, considerese un seguro SR NAND con las funciones booleanas siguientes para S y R.

$$S = AB + CD$$

$$R = A'C$$

Ya que este es un seguro NAND, deben aplicarse los valores complementados a las entradas.

$$S = (AB + CD)' = (AB)' (CD)'$$

$$R = (A'C)'$$

Esta implementación se muestra en la figura 5.63(a). S se genera con dos compuertas NAND y una AND. La función booleana para la salida Q es:

$$Q = (Q'S)' = (Q' (AB)' (CD)')'$$

La función se genera en la figura 5.63(b) con los dos niveles de compuertas NAND. Si la salida Q es igual a 1, entonces Q' es igual a 0. Si dos de las tres entradas pasan en forma momentánea a 1, la compuerta NAND asociada con la salida Q permanecerá en 1 ya que Q' se mantiene en 0.

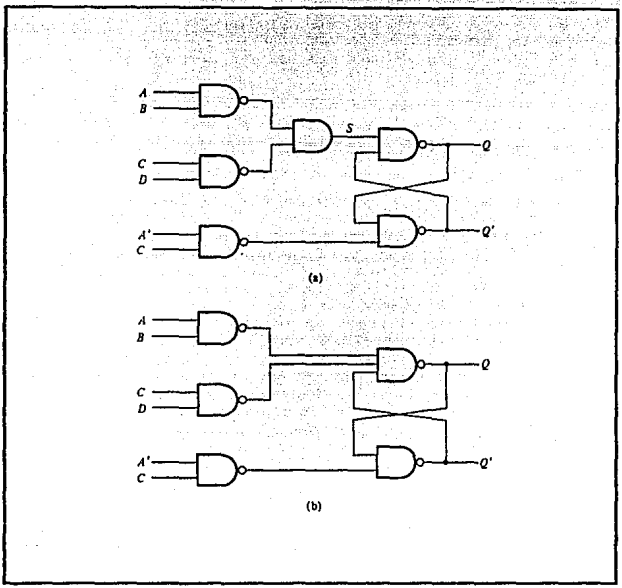


FIG. 5.63 IMPLEMENTACION DE SEGURO.

En la figura 5.63(b) se muestra un circuito típico que puede usarse para construir circuitos secuanciales asíncronos. Las dos compuertas NAND que componen el seguro en forma normal tienen dos entradas. Sin embargo, si las funciones S o R contienen dos o mas terminos producto cuando se expresan en suma de productos, entonces la compuerta NAND correspondiente del seguro SR tendra tres o mas entradas. Por tanto, los dos terminos en la expresion original de suma de productos para S son AB y CD y cada uno se implementa con una compuerta NAND cuya salida se aplica a la entrada del seguro NAND. En esta forma, cada variable de estado requiere un circuito de dos niveles de compuertas NAND. El primer

nivel consta de compuertas NAND que implantan cada término producto en la expresión booleana original de S y R. El segundo nivel forma la conexión en acoplamiento cruzando del seguro SR con entradas que vienen de las salidas de cada caompuerta NAND en el primer nivel.

5.3.4.4 RIESGOS ESENCIALES.

Hasta ahora se ha considerado lo que se conoce como peligro estático y dinámico. Hay otro tipo de peligro que es posible ocurra en los circuitos secuenciales asincronos, llamado riesgo esencial. El riesgo esencial esta causado por retardos desiguales en dos o más trayectorias que se originan desde la misma entrada. Un retardo excesivo a traves de un circuito inversor en comparación con el retardo asociado con la trayectoria de retroalimentación puede provocar dicho riesgo. Los riesgos esenciales no pueden corregirse por la adicion de compuertas redundantes como en el caso de los peligros estáticos. El problema que plantean puede corregirse ajustando la cantidad de retardo en la trayectoria afectada. Para evitar riesgos esenciales, cada lazo de retroalimentación debe manipularse con cuidado individual para asegurar que el retardo en la trayectoria de retroalimentación sea suficiente largo en comparación con los retardos de otras sañales que se originan desde las terminales de entrada. Este problema tiende a ser característica especial, ya que depende del circuito particular usado y la cantidad de retardo que se encuentran en sus diversas trayectorias.

5.3.5 DISENO DE SECUENCIALES ASINCRONOS.

Se ha llegado ahora a la posición de examinar un ejemplo de diseño completo de un circuito secuencial asincrono. Este ejemplo puede servir como referencia para el diseño de otros circuitos similares. Se demostrará el método de diseño al seguir los pasos de procedimiento siguiente :

1. Se establecen las especificaciones de diseño.
2. Se deriva una tabla de flujo primitiva.
3. Se reduce la tabla de flujo fusionando los renglones.

4. Se hace la asignación de estados binarios libre de carreras.

5. Se obtiene la tabla de transición y el mapa de salidas.

6. Se obtiene el diagrama lógico usando seguros SR.

5.3.5.1 ESPECIFICACIONES DE DISEÑO.

Es necesario diseñar un flip-flop T con disparo en el borde negativo. El circuito tiene dos entradas, T (lengüeta) y C (reloj), y una salida, Q. El estado de la salida se complementa si T = 1 y el reloj C cambia desde 1 a 0 (disparo en borde negativo). En otra forma, bajo cualquier otra condición de entrada, la salida Q permanece sin cambio. Aunque este circuito puede usarse como un flip-flop en circuitos secuenciales con reloj, el diseño interno del flip-flop (como es en el caso con todos los demas flip-flops) es un problema asíncrono.

5.3.5.2 TABLA DE FLUJO PRIMITIVA.

La derivación de la tabla de flujo primitiva puede facilitarse si se deriva primero una tabla que liste todos los estados totales posibles en el circuito. Esto se muestra en la Tabla 5.19. Se principia con la condicion de entrada TC = 11 y se le asigna el estado a. El circuito pasa al estado b y la salida Q se

ESTADO	ENTRADAS T	ENTRADAS C	SALIDAS Q	COMENTARIOS
a	1	1	0	SALIDA INICIAL ES 0
b	1	0	1	DESPUES DE a
c	1	1	1	SALIDA INICIAL ES 1
d	1	0	0	DESPUES DE c
e	0	0	0	DESPUES DE d O DE f
f	0	1	0	DESPUES DE e O DE a
g	0	0	1	DESPUES DE b O DE h
h	0	1	1	DESPUES DE g O DE c

TABLA 5.19 ESPECIFICACION DE ESTADOS TOTALES.

complementa desde 0 a 1 cuando C cambia desde 1 a 0 mientras T permanece en 1. Otro cambio en la salida ocurre cuando el circuito pasa desde el estado c al estado d. En este caso T = 1, C cambia desde 1 a 0 y la salida Q se complementa desde 1 a 0. Los otros cuatro estados en la tabla no cambian la salida, porque T es igual a 0. Si Q en forma inicial es 0, permanece en 0, si al inicio es 1, permanecerá en 1 aun cuando cambie la entrada del reloj. Esta información resulta en seis estados totales. Obsérvese que no se incluyen las transiciones simultaneas de las dos variables de estado, como desde 01 a 10, ya que violan la condición para la operación en modo fundamental.

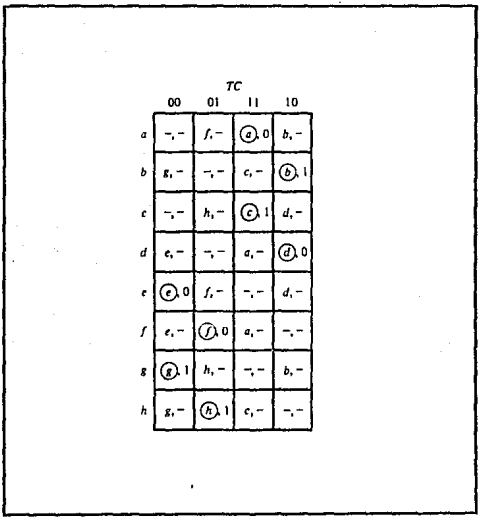


FIG. 5.64 TABLA PRIMITIVA DE FLUJO.

La tabla de flujo primitiva se muestra en la figura 5.64. La información para la tabla de flujo puede obtenerse en forma

directa mediante las condiciones que se listan en la Tabla 5.19. Se llena primero en una casilla en cada renglón que pertenece al estado estable en este renglón como se lista en la tabla. Entonces se anotan guiones en las casillas cuyas entradas difieren por dos variables de la entrada correspondiente al estado estable. Las condiciones inestables se deteminan entonces utilizando la información anotada como comentarios en la Tabla 5.19.

5.3.5.3 FUSION DE LA TABLA DE FLUJO.

Los renglones en la tabla de flujo primitiva se fusionan obteniendo primero todos los pares de estados compatibles. Esto se hace mediante la tabla de implicación que se muestra en la figura 5.65. Las casillas que contienen marcas de verificación definen los pares compatibles:

(a, f) (b, g) (b, h) (c, h) (d, e) (d, f) (e, f) (g, h)

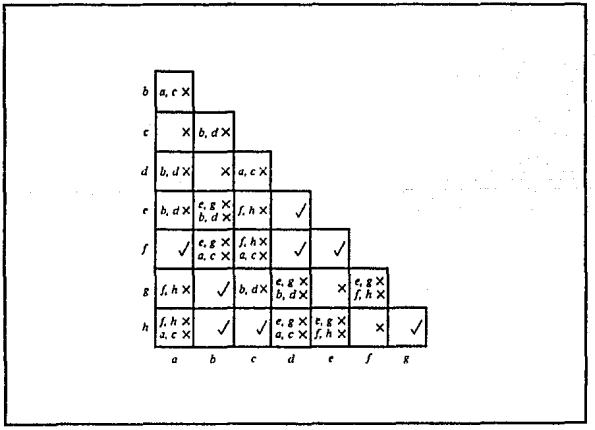


FIG. 5.65 TABLA DE IMPLICACION.

renglones, reteniendo los ocho símbolos alfabéticos originales. La otra alternativa para dibujar la tabla de flujo fusionada se muestra en la parte (b) de la figura. Aquí se asigna un símbolo común alfabético a todos los estados estables en cada renglón fusionado. Por lo tanto, el símbolo f se reemplaza por a, y g y h se reemplaza por b, y en forma similar para los otros dos renglones. La segunda alternativa muestra en forma clara una tabla de flujo de cuatro estados sólo con cuatro símbolos alfabéticos para los estados.

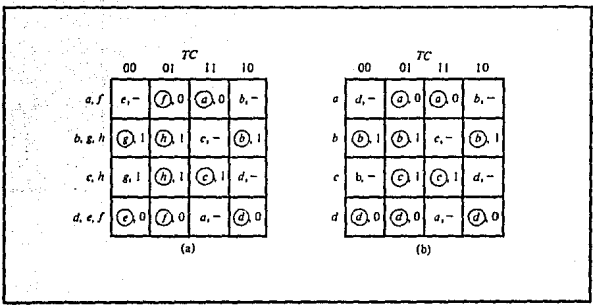


FIG. 5.67 TABLA DE FLUJO REDUCIDA.

5.3.5.4 ASIGNACION DE ESTADO Y TABLA DE TRANSICION.

El siguiente paso en el diseño es encontrar una asignación binaria libre de carrera para los cuatro estados estables en la tabla de flujo reducida. Con objeto de encontrar una asignación adyacente adecuada, se dibuja el diagrama de transición como se muestra en la figura 5.68. Para este ejemplo es posible obtener una asignación adyacente adecuada sin necesidad de estados adicionales. Esto se debe a que no hay líneas diagonales en el diagrama de transición.

Al sustituir la asignación binaria indicada en el diagrama de transición en la tabla de flujo reducida, se obtiene la tabla de

transición que se muestra en la figura 5.69. El mapa de salidas se obtiene mediante la tabla de flujo reducida. Los guiones en la sección de salida son valores asignados de acuerdo con las reglas establecidas anteriormente.

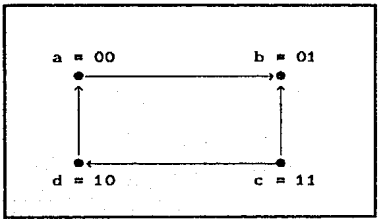


FIG. 5.68 DIAGRAMA DE TRANSICION.

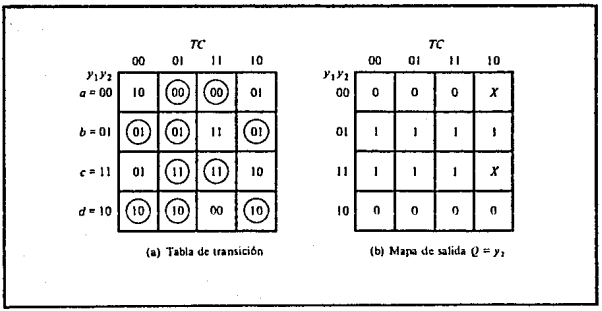


FIG. 5.69 TABLA DE TRANSICION Y MAPA DE SALIDA.

5.3.5.5 DIAGRAMA LOGICO.

El circuito que va a diseñarse tiene dos variables de estado,

Y_1 y Y_2, y una salida, Q. El mapa de salida en la figura 5.69 muestra que Q es igual a la variable de estado y_2. La immplementación del circuito requiere dos seguros SR, uno para cada variable de estado. Los mapas para las entradas S y R de los dos seguros se muestran en la figura 5.70. Las funciones booleanas simplificadas se anotan bajo cada mapa.

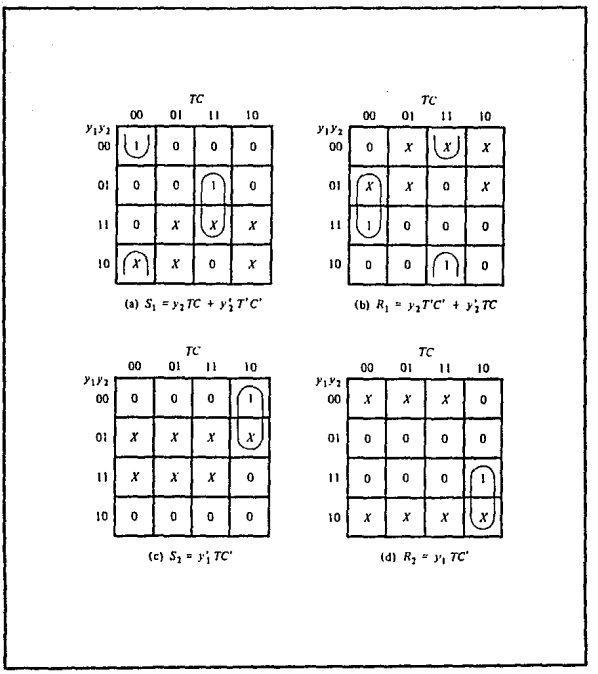


FIG. 5.70 MAPAS PARA ENTRADA DE SEGURO.

El diagrama lógico del circuito se muestra en la figura 5.71. Aquí se usan dos seguros NAND con dos o tres entradas en cada compuerta. Esta implementación está de acuerdo con el patrón establecido junto con la figura 5.63(b). Las funciones de entrada S y R requieren seis compuertas NAND para su implantación.

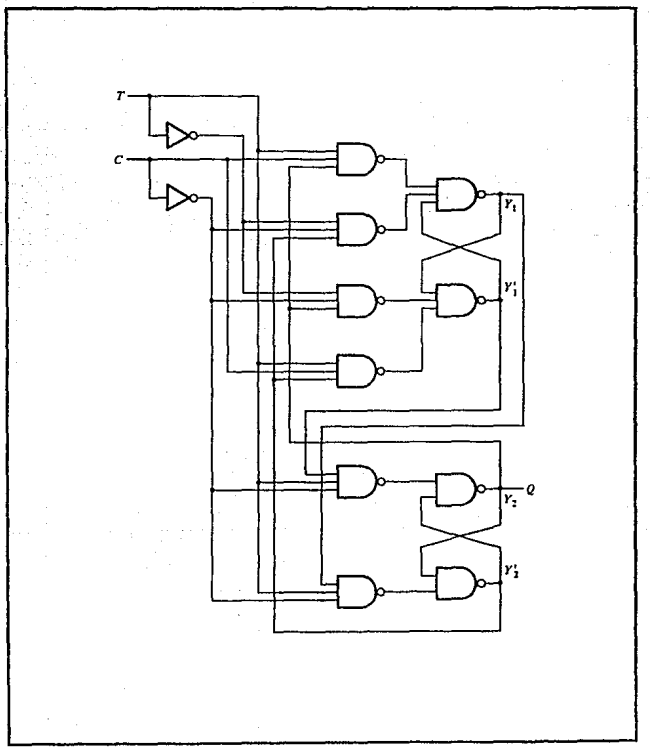


FIG. 5.71 DIAGRAMA LOGICO DE F-F CON DISPARO DE BORDE NEGATIVO.

Este ejemplo demuestra la complicación implicada al diseñar circuitos secuenciales asíncronos. Fue necesario pasar a través de varios diagramas con objeto de obtener el diagrama del circuito final. Aunque la mayor parte de los circuitos digitales son síncronos, hay ocaciones en que tiene que tratarse con el comportamiento asíncrono. Las propiedades básicas presentadas en este capítulo son esenciales para comprender por completo el comportamiento interno de los circuitos digitales.

EJERCICIOS PROPUESTOS.

5.1 Muestre el diagrama lógico de un flip-flop RS temporizado con cuatro compuertas NAND.

5.2 Muestre el diagrama lógico de un flip-flop D temporizado con compuertas AND y NOR.

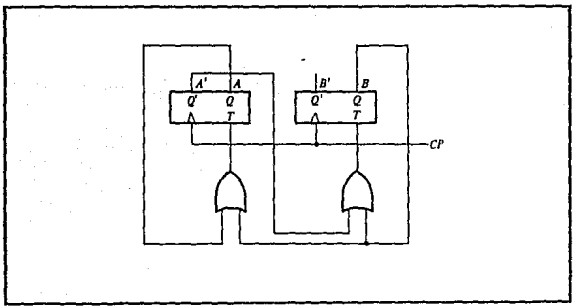
5.3 Dibuje el diagrama lógico (mostrando todas las compuertas) de un flip-flop D maestro-esclavo.

5.4 (a) Explique la diferencia entre los circuitos secuenciales asíncronos y los síncronos.

(b) Defina el modo fundamental de operación.

(c) Explique la diferencia entre los estados estable e inestable.

5.5 Derive la tabla de estado y el diagrama de estado del circuito secuencial de la figura siguiente. ¿Cúal es la función del circuito?.

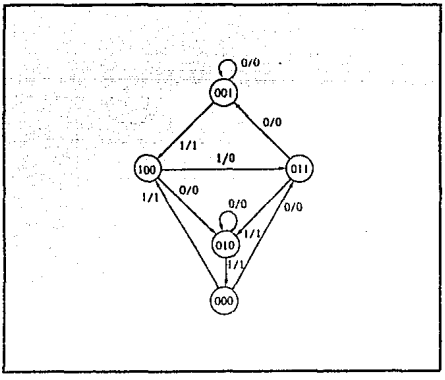


5.6 Reduzca el numero de estados en la siguiente tabla de estados y tabule la tabla de estados reducida.

ESTADO PRESENTE	ESTADO SIGUIENTE		SALIDA	
	x = 0	x = 1	x = 0	x = 1
a	f	b	0	0
b	d	c	0	0
c	f	e	0	0
d	g	a	1	0
e	d	c	0	0
f	f	b	1	1
g	g	h	0	1
h	g	a	1	0

5.7 Un circuito secuencial tiene una entrada y una salida. El diagrama de estado se muestra en la figura siguiente. Diseñe el circuito secuencial con :

(a) Flip-flops T.
(b) Flip-flops RS.
(c) Flip-flops JK.



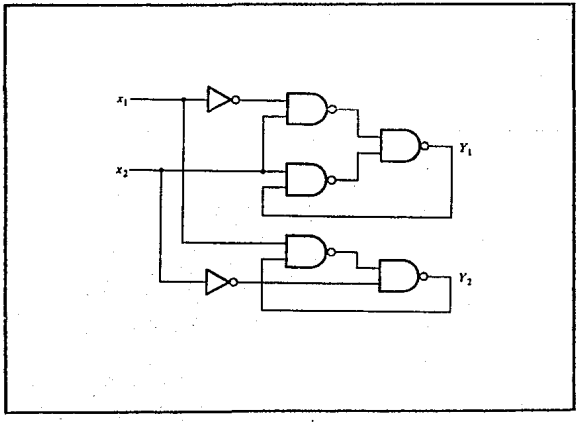
5.8 El contenido de un registro de corrimiento de 4 bits en forma inicial es 1101. El registro se corre seis veces a la derecha, con la entrada serial de 101101. ¿Cuál es el contenido del registro despues de cada corrimiento?.

5.9 ¿Cuál es la diferencia entre las transferencias serial y paralela? ¿Qué tipo de registro se emplea en cada caso?

5.10 El registro de corrimiento bidireccional de 4 bits que se muestra en la figura 5.31 está encerrado dentro de un paquete IC.

(a) Dibuje un diagrama de bloques del IC mostrando todas las entradas y salidas.
(b) Dibuje un diagrama de bloques usando tres IC para producir un registro de corrimiento bidireccional de 12 bits.

5.11 Derive la tabla de transición para el circuito secuencial asíncrono que se muestra en la siguiente figura. Determine la secuencia de los estados internos Y_1Y_2 para la siguiente secuencia de entradas x_1x_2 : 00, 10, 11, 01, 11, 10, 00.



5.12 Obtenga una tabla de flujo primitiva para un circuito con dos entradas, x_1 y x_2, y dos salidas, z_1 y z_2, que satisfacen las cuatro condiciones siguientes:

(a) Cuando $x_1x_2 = 00$, la salida es $z_1z_2 = 00$.
(b) Cuando $x_1 = 1$ y x_2 cambia desde 0 a 1, la salida es $z_1z_2 = 01$.
(c) Cuando $x_2 = 1$ y x_1 cambia desde 0 a 1, la salida es $z_1z_2 = 10$.
(d) De otra manera la salida no cambia.

5.13 Se instala un semáforo en un crucero de una vía de ferrocarril y una carretera. El semaforo está controlado por dos interruptores en los rieles colocados a una milla de distancia en cada lado del crucero. Un interruptor enciende cuando el tren está sobre él y se apaga si no sucede esto. La luz del semáforo cambia de verde (0) a rojo (1) cuando la parte delantera del tren está a una milla del crucero. La luz cambia regresando a verde cuando el extremo posterior del tren está a una milla del crucero. Se supone que la longitud del tren es menor a dos millas.

(a) Obtenga una tabla de flujo primitiva para el circuito.
(b) Muestre que la tabla de flujo puede reducirse a cuatro renglones.

5.14 Es necesario diseñar un circuito secuencial asíncrono con dos entradas, x_1 y x_2, y una salida, z. Al inicio tanto las entradas como la salidas son iguales a 0. Cuando x_1 o x_2 llega a ser 1, z llega a ser 1. Cuando la segunda entrada también llega a ser 1, la salida cambia a 0. La salida permanece en 0 hasta que el circuito regresa al estado inicial.

(a) Obtenga una tabla de flujo primitiva para el circuito y muestre que puede reducirce a la tabla de flujo siguiente.
(b) Complete el diseño del circuito.

	x_1x_2 00	01	11	10
a	(a), 0	(a), 1	b, –	(a), 1
b	a, –	(b), 0	(b), 0	(b), 0

CAPITULO VI : M E M O R I A S

CONTENIDO :

CAPITULO VI : MEMORIAS

Una ventaja importante de los sistemas digitales sobre los analógicos es la capacidad de almacenar fácilmente grandes cantidades de información digital por periodos de tiempo cortos o largos. Esta capacidad de memoria es la que hace de los sistemas digitales tan versátiles y adaptables a muchas situaciones. Por ejemplo, en una computadora digital la memoria central interna almacena instrucciones que indican a la computadora qué hacer en todas las circunstancias posibles, de manera que la computadora haga su trabajo con una mínima cantidad de intervención humana.

Ya nos hemos familiarizado con el multivibrador biestable (flip-flop), el cual es un dispositivo de memoria electrónico. Los registros de los FF son elementos de memoria de alta velocidad que se usan extensamente en las operaciones internas de una computadora digital, donde la información digital se desplaza en forma continua de una localidad a otra. Adelantos de la tecnología LSI y VLSI han hecho posible obtener grandes números de biestables en un solo integrado dispuestos en diversos formatos de arreglo de memoria. Estas memorias son los dispositivos especializados más veloces de que se dispone y su costa ha venido disminuyendo continuamente a medida que se mejora la tecnología de los LSI.

6.1. ESTRUCTURAS Y PARAMETROS DE LAS MEMORIAS ROM, EPROM, RAM Y PLA.

6.1.1. MEMORIAS ROM (READ ONLY MEMORY).

Este tipo de memoria se diseña con el fin de contener datos que sean permanentes o bien que no cambien frecuentemente.

Durante la operación normal, no pueden escribirse nuevos datos en una ROM pero sí puede leerse información de ella. Para algunas ROM los datos que están almacenados tienen que grabarse en el proceso de fabricación; para otras ROM los datos se pueden introducir en forma eléctrica. El proceso de grabar datos se conoce como programación de la ROM. Algunas ROM no puden alterar sus datos una vez que se hayan programado; otras pueden borrarse y reprogramarse con la frecuencia que se desee.

Las ROM se usan para almacenar datos e información que no cambiará durante la operación de un sistema. Un uso importante de las ROM es en el almacenamiento de programas en microcomputadoras. Ya que todas las ROM no son volátiles, estos programas no se pierden cuando la microcomputadora es desconectada, cuando se enciende la máquina, pueden empezar de inmediato a ejecutar el programa almacenado en ROM. Las ROM también se emplean para almacenar programas y datos en equipo controlado por microprocesadores como complejas cajas registradoras electrónicas.

6.1.1.1. DIAGRAMA DE BLOQUE DE UNA ROM

Un digrama de bloque común para una ROM se muestra en la figura 6.1. Tiene tres conjuntos de señales: entrada de dirección, entrada(s) de control y salidas de datos. Esta ROM almacena 16 palabras, ya que tiene 2^4=16 posibles direcciones y cada palabra contiene 8 bits, puesto que hay 8 salidas de datos. Por lo tanto, ésta es una ROM de 16 x 8. Otra manera de describir esta capacidad de la ROM consiste en decir que almacena 16 bytes de datos.

Las salidas de datos de muchos circuitos integrados de ROM son salidas con colector abierto o bien de estado triple para permitir la conexión de muchos circuitos ROM a la misma línea de datos para lograr la expansión de la memoria. Los números más comunes de salidas de datos para ROM son 4 y 8 bits, con palabras de 8 bits que son las más comunes.

La entrada CS significa selección de integrados. Esta es escencialmente una entrada activadora que acciona o desactiva las salidas ROM. La entrada CS que se muestra en la figura 6.1 es alta activa.

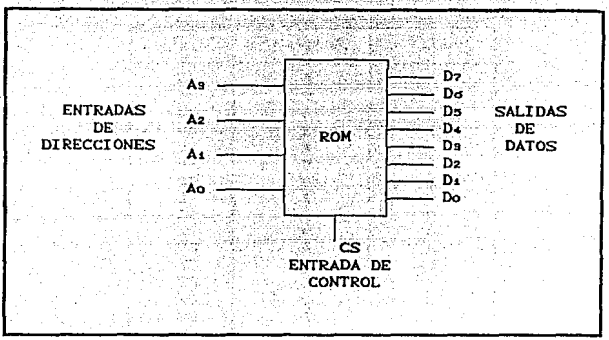


FIG. 6.1 DIAGRAMA DE BLOQUE DE LA ROM.

6.1.1.2. LA OPERACION DE LECTURA (READ).

Supóngase que la ROM ha sido programada con los datos que se muestran en la tabla 6.1. Dieciséis diferentes palabras de datos se almacenan en las 16 localidades de dirección distintas. Por ejemplo, la palabra de datos almacenada en la localidad 0011 es 10101111. Por supuesto, los datos se almacenan en binario dentro de la ROM, pero con mucha frecuencia se usa la notación hexadecimal para mostrar adecuadamente los datos programados. Esto se hace en la tabla 6.2.

A fin de leer una palabra de datos de la ROM, se necesitan hacer dos cosas : aplicar las entradas de dirección adecuadas y luego activar las entradas de control. Por ejemplo, si se desean leer los datos almacenados en la localidad 0111 de la ROM de la figura 6.1, tenemos que aplicar $A_3A_2A_1A_0$=0111 a las entradas de dirección y después aplicar un estado ALTO a CS. Las entradas de dirección se de decodificarán dentro de la ROM con el objeto de seleccionar la palabra de datos correcta, 11101101, que aparecerá en las salidas D_7-D_0. Si CS se mantiene en BAJO, las salidas de la ROM se desactivarán y estarán en el estado Hi-Z (Z ALTO).

PALABRA	$A_3A_2A_1A_0$	D_7-D_0
0	0	DE
1	1	3A
2	2	85
3	3	AF
4	4	19
5	5	7B
6	6	00
7	7	ED
8	8	3C
9	9	FF
10	A	B8
11	B	C7
12	C	27
13	D	6A
14	E	D2
15	F	5B

TABLA 6.1 DATOS PROGRAMADOS.

PALABRA	$A_3A_2A_1A_0$	$D_7D_6D_5D_4D_3D_2D_1D_0$
0	0 0 0 0	1 1 0 1 1 1 1 0
1	0 0 0 1	0 0 1 1 1 1 0 1 0
2	0 0 1 0	1 0 0 0 0 0 1 0 1
3	0 0 1 1	1 0 1 0 1 1 1 1
4	0 1 0 0	0 0 0 1 1 0 0 1
5	0 1 0 1	0 1 1 1 1 0 1 1
6	0 1 1 0	0 0 0 0 0 0 0 0
7	0 1 1 1	1 1 1 0 1 1 0 1
8	1 0 0 0	0 0 1 1 1 1 0 0
9	1 0 0 1	1 1 1 1 1 1 1 1
10	1 0 1 0	1 0 1 1 1 0 0 0
11	1 0 1 1	1 1 0 0 0 1 1 1
12	1 1 0 0	0 0 1 0 0 1 1 1
13	1 1 0 1	0 1 1 0 1 0 1 0
14	1 1 1 0	1 1 0 1 0 0 1 0
15	1 1 1 1	0 1 0 1 1 0 1 1

TABLA 6.2 DATOS HEXADECIMALES (HEX).

6.1.1.3. ARQUITECTURA DE LA ROM.

La arquitectura (estructura) interna de un circuito integrado ROM es muy compleja y no necesitamos conocer todos sus detalles. Sin embargo, es instructivo observar un diagrama simplificado de la arquitectura interna, como el que se muestra en la figura 6.2 de la ROM de 16 x 8. Existen cuatro partes básicas: decodificador de hilera, decodificador de columnas, arreglo de registros y separadores (buffers) de salida.

6.1.1.3.1. ARREGLO DE REGISTROS.

El arreglo de registros almacena los datos que han sido programados en la ROM. Cada registro contiene un número de celdas de memoria que es igual al tamaño de la palabra. En este caso, cada registro almacena una palabra de 8 bits. Los registros se disponen en un arreglo de matriz cuadriculada que es común a muchos integrados de memoria semiconductora. Podemos especificar la posición de cada registro como ubicada en una hilera y columna específicas. Por ejemplo, el registro 0 se encuentra en la hilera 0/columna 0 y el registro 9 está en la hilera 1/columna 2.

Las ocho salidas de datos de cada registro se conectan en una línea de datos interno que corre a través de todo el circuito. Cada registro tiene dos entradas activadas (E); ambas tienen que ser ALTAS a fin de que los datos del registro sean colocados en la línea.

6.1.1.3.2. DECODIFICADOR DE DIRECCIONES.

El código de dirección aplicado $A_3A_2A_1A_0$ determina qué registro del arreglo será desactivado para colocar su palabra de datos de 8 bits en la línea. Los bits de dirección A_1A_0 se alimentan a un decodificador 1 de 4 que activa una línea de selección de hilera y los bits de dirección A_3A_2 se alimentan a un segundo decodificador 1 de 4 que activa una línea de selección de columna. Solamente un registro estará en la hilera y columna seleccionadas por las entradas de dirección y estará activado.

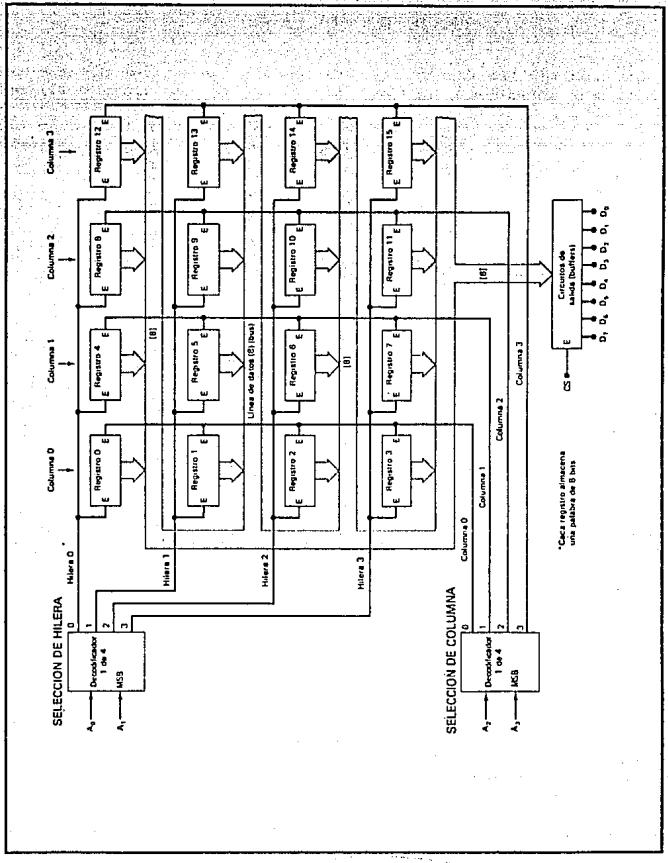


FIG. 6.2 ARQUITECTURA DE LA ROM DE 16 X 8.

Ejemplo :

Qué registro será activado por la dirección de entrada 1101 ?

Solución :

A_3A_2 = 11 ocacionará que el decodificador de columnas active la linea de selección de la columna 3 y A_1A_0 = 01 ocacionará que el decodificador de hileras active la linea de selección de la hilera 1. Esto colocará estados ALTOS en ambas entradas activadas del registro 13, con lo cual ocacionará que sus salidas de datos sean colocadas en la linea. Nótese que los otros registros en la columna 3 sólo tendrán una entrada activada; lo mismo sucederá con los otros registros de la Hilera 1.

6.1.1.3.3. CIRCUITOS (BUFFERS) DE SALIDA.

El registro que es activado por las entradas de dirección colocará sus datos en la línea de datos. Estos datos se alimentan a los circuitos de salida, los cuales transmitirán los datos a las salidas de datos externas, siempre que CS sea ALTA. Si CS es BAJA, los circuitos de salida están en el estado Hi-Z y D7-D0 estarán flotando.

La arquitectura que se muestra en al figura 6.2 es análoga a la de muchos IC de ROM. Según el número de palabras de datos almacenadas, los registros de algunas ROM no se dispondrán en un arreglo cuadrado.

6.1.1.4. DISTRIBUCION DE LA ROM.

Habrá una demora en la propagación entre la aplicación de las entradas de una ROM y la aparición de las salidas de datos durante una operación READ. Esta demora, denominada tiempo de acceso, t_{ACC}, es una medida de la velocidad de operación de la ROM. El tiempo de acceso se describe gráficamente por medio de las ondiformes de la figura 6.3.

La forma de onda de más arriba representa las entradas de dirección, la del medio es una selección del integrado LOW activa, CS', y la de más abajo representa las salidas de datos. Al tiempo t_0 las entradas de dirección están todas en algún nivel especificado, algunas en ALTO y algunas en BAJO. CS' es ALTA, de manera que las salidas de datos de la ROM se encuentran en su estado Hi-Z (representado por la linea punteada).

Antes de t_1 las entradas de dirección cambian a una nueva dirección para realizar una nueva operación READ. En t_1 la nueva dirección es válida; es decir, cada entrada de dirección está en un nivel lógico válido. En este punto los circuitos internos de la ROM empiezan a decodificar las entradas de dirección para seleccionar el registro que enviará sus datos a los circuitos de salida. En t_2 la entrada CS' es activada para los circuitos de salida. Finalmente, en t_3, las salidas cambian del estado Hi-Z a los datos válidos que representan los almacenados en la dirección especificada.

La demora entre t_1 y t_3, cuando la nueva dirección y las salidas de datos se vuelven válidas, es el tiempo de acceso t_{ACC}. Las ROM comunes tendrán tiempos de acceso en el orden de 30 a 90 ns.

Otro importante parámetro de distribución es el tiempo de activación de salida, t_{OE}, que es la demora entre la entrada CS' y salida de datos válida. Valores comunes de t_{OE} son 20 ns para ROM. Este parámetro de distribución es importante en situaciones donde las entradas de dirección están ya en sus nuevos valores, pero las salidas de la ROM no han sido acitivadas aún. Cuando CS' pasa a BAJO para activar las salidas, la demora será t_{OE}.

6.1.2. MEMORIA ROM PROGRAMABLE Y BORRABLE (EPROM).

Una EPROM puede ser programada por el usuario y también puede borrarse y reprogramarse tantas veces como se desee. Una vez programada, la EPROM es una memoria no volátil que contendrá sus datos almacenados indefinidamente. El proceso para programar una EPROM implica la aplicación de niveles de voltaje especiales (comúnmente en el orden de 25 a 50 V) a las entradas del circuito adecuadas en una cantidad de tiempo especificada (por lo general 50 ms por localidad de dirección). El proceso de programación usualmente es efectuado por un circuito especial de programación que está separado del circuito en la cual la EPROM trabajará por

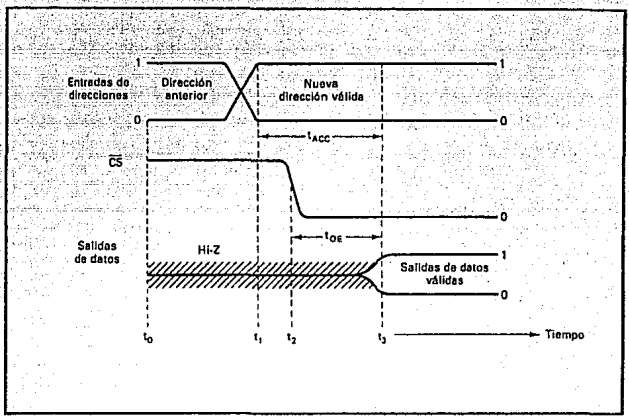


FIG. 6.3 DISTRIBUCION COMUN DE UNA OPERACION READ EN ROM.

último. El proceso de programación completo puede llevar hasta 7 minutos para una EPROM.

Las celdas de almacenamiento de una EPROM Oson transistores con efecto de campo con una compuerta de silicón que no tiene conexiones eléctricas (es decir, una compuerta flotante). Aplicando una pulsación de programación especial de alto voltaje al dispositivo, se inyectan electrones de alta energía en la región de la compuerta flotante, con lo cual se "enciende" el transistor. Estos electrones permanecen capturados en esta región una vez que la pulsación termina, ya que no hay trayectoria de descarga. Durante este proceso de programación, la dirección del circuito y las puntas de datos se usan para determinar qué celdas de memoria serán afectadas por la pulsación de programación.

Una vez que se ha programado una celda de memoria, puede ser borrada solamente exponiéndola a la luz ultravioleta (UV) aplicada a través de una ventana en el integrado. La luz UV produce un flujo de fotocorriente desde la puerta flotante en retroceso hacia el sustrato de silicón, con lo cual la compuerta se devuelve a su condición inicial. Nótese que no hay de manera

de exponer una sola celda a la luz UV sin exponer todas las celdas. Por consiguiente, la UV borrará toda la memoria. El proceso de borrado requiere comúnmente de 15 a 30 minutos de exposición a rayos ultravioleta.

Las EPROM están disponibles en una amplia selección con capacidades hasta de 32 K x 8 y tiempos de acceso de bajo 250 ns. Utilizaremos la popular Intel 2716 para ilustrar la operación regular de la EPROM. La 2716 es una EPROM de 2K x 8 que opera desde una sola fuente de +5 V, a diferencia de las primeras EPROM como la 2708 que necesitaban tres diferentes voltajes de fuente. El símbolo de bloque del 2716 se muestra en la figura 6.4. Nótese que tiene 11 entradas de dirección, ya que $2^{11} = 2048$ y 8 salidas de datos. Tiene dos entradas de fuente de energía Vcc y Vpp, las cuales requieren ambas +5 V durante una operación normal. Sin embargo, Vpp, tiene que fijarse en +25 V durante el proceso de programación.

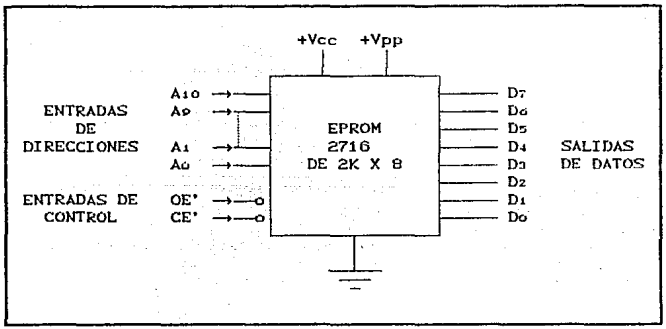


FIG. 6.4 SIMBOLO DE LA EPROM 2716.

Existen dos entradas de control comunes. OE es la activación de salida que controla los circuitos (buffers) de salida para determinar si los datos de la EPROM aparecen o no en las terminales de salida. CE es una entrada de activación del circuito que tiene dos funciones distintas. Durante una operación normal, actúa como una entrada de control de energía que determina si el integrado operará completamente o no. Para leer de la EPROM, CE tiene que ser BAJA a fin de que los circuitos

internos seleccionen los datos y los envíen a los circuitos de salida y OE tiene que ser BAJA para que los datos aparezcan en las puntas de salida. Cuando CE es ALTA, el circuito está en un modo de trasición donde consume mucho menos energía (132 mW) que en modo activo (525 mW).

CE' se usa también durante el proceso de programación donde tiene que ser pulsada de BAJA a ALTA durante 50 ms, cada vez que una nueva palabra vaya a ser programada en una de las 2048 localidades de dirección. La figura 6.5 muestra las condiciones que se necesitan para programar una palabra de datos en una localidad. Súpongase que la EPROM ha sido borrada anteriormente con luz UV de manera que es una EPROM "limpia" (todas las celdas son unos). Obsérvese que Vpp se conecta a +25 V. Las etapas que se requieren para programar cualquier localidad de dirección son como sigue:

1.- Aplicar la dirección deseada a las entradas de dirección.

2.- Aplicar la palabra de datos de 8 bits deseada a las terminales de datos D7-Do. Estas teminales de datos funcionarán como entradas en el modo de programa (Nótese que OE' es ALTA).

3.- Aplicar una pulsación de 50 ms BAJA a ALTA a CE'. En la terminación de esta pulsación, la localidad de dirección seleccionada debe almacenar la palabra de datos aplicada.

4.- A fin de verificar que la palabra de datos a sido programada adecuadamente, la localidad de la dirección debe ser leida. Esto se lleva a cabo aplicando condiciones BAJAS a CE' y OE' y leyendo los niveles en las terminales de salida de datos.
NOTA : Antes de esta etapa de verificación, las entradas de datos que se aplicaron durante las etapas de programación deben desconectarse de las terminales de salida de datos.

El proceso de programación cuando se efectúa manualmente puede tomar horas. Se dispone de numerosos programadores comerciales de EPROM que pueden programar y verificar una 2716 completa en menos de 2 minutos.

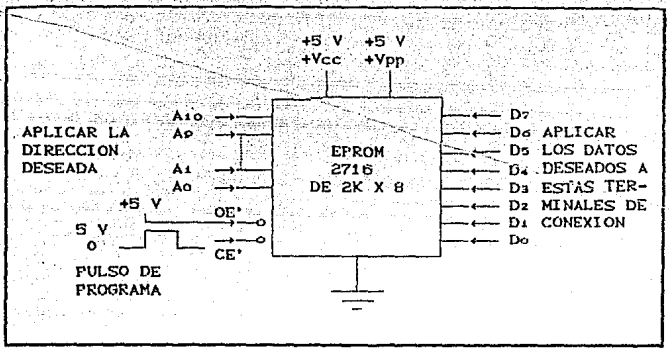


FIG. 6.5 PARA PROGRAMAR UNA 2716, LOS DATOS DESEADOS SE APLICAN A LAS TERMINALES DE DATOS Y SE APLICA UN PULSO DE PROGRAMA A CE'.

6.1.3. MEMORIA RAM (RANDOM ACCESS MEMORY).

El término RAM significa memoria con acceso aleatorio, lo cual quiere decir que cualquier localidad de dirección de memoria se puede accesar fácilmente como cualquier otra.

La RAM se emplea en las computadoras para el almacenamiento temporal de programas y datos. El contenido de muchas localidades de dirección en la RAM cambiará continuamente conforme la computadora ejecute un programa. Esto requiere tiempos de ciclo de lectura y escritura rápidos para la RAM de manera que no disminuya el tiempo de operación de la computadora.

Una desventaja importante de las RAM es que son volátiles y pierden toda la información almacenada en ellas si se interrumpe el suministro de energía o si se apaga la máquina. Sin embargo, algunas RAM emplean pequeñas cantidades de energía en modo de transición (sin efectuar operaciones de lectura o escritura) y pueden alimentarse con baterías siempre que se interrumpa la fuente de energía principal. Por supuesto, la ventaja principal de la RAM es que se puede escribir en ella y también se puede leer de ella muy rápidamente con la misma facilidad.

6.1.3.1. ARQUITECTURA DE LA RAM.

Como sucede con la ROM, es útil pensar que la RAM consta de varios registros, cada uno de los cuales almacena una sola parabra de datos y con una dirección única. Las RAM comúnmente vienen con capacidades de palabras de 1K, 4K, 8K, 16K o bien 64K y tamaños de palabra de 1, 4 u 8 bits.

La figura 6.6 muestra la arquitectura simplificada de una RAM que almacena 64 palabras de 4 bits cada una (es decir, una memoria de 64 x 4). Estas palabras tienen direcciones que van de

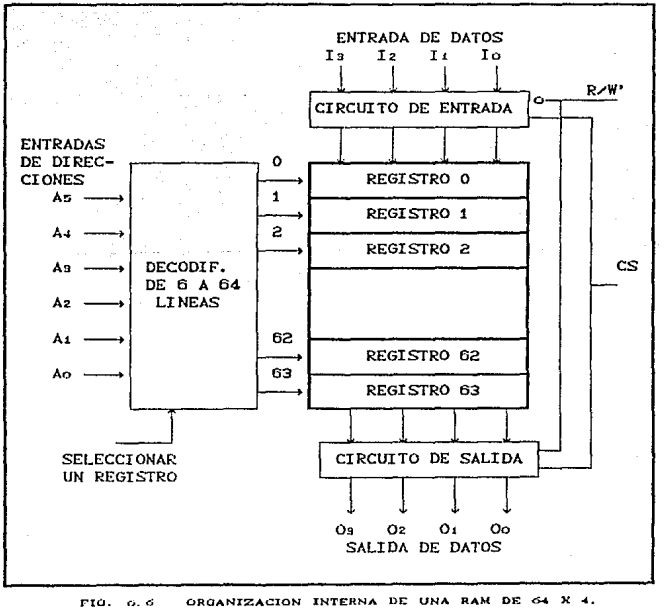


FIG. 6.6 ORGANIZACION INTERNA DE UNA RAM DE 64 X 4.

0 a 63_{10}. A fin de seleccionar una de las 64 localidades de dirección para leer o escribir, se aplica un código de dirección binario a un circuito decodificador. Ya que $64 = 2^6$, el decodificador requiere un código de entrada de 6 bits. Cada código de dirección activa una determinada salida del decodificador la cual, a su vez, activa su registro correspondiente. Por ejemplo, supóngase un código de dirección aplicado de :

$$A_5A_4A_3A_2A_1A_0 = 011010$$

Como $011010_2 = 26_{10}$, la salida del decodificador 26 pasará a estado ALTO, seleccionando el registro 26 para una operación de lectura o escritura.

6.1.3.1.1. OPERACION DE LECTURA.

El código de dirección selecciona un registro del circuito de memoria para leer o escribir. A fin de leer el contenido del registro seleccionado, la entrada READ/WRITE (R/W') debe ser 1. Además, la entrada CS (CHIP SELECT) deber ser activada (un 1 en este caso). La combinación de R/W'=1 y CS=1 activa los buffers de salida de manera que el contenido del registro seleccionado aparecerá en las cuatro salidas de datos. R/W'=1 también desactiva los circuitos de entrada de manera que las entradas de datos no afecten la memoria durante la operación de lectura.

6.1.3.1.2. OPERACION DE ESCRITURA.

Para escribir una nueva palabra de 4 bits en el registro seleccionado se requiere que R/W'=0 y CS=1. Esta combinación activa los circuitos de entrada de manera que la palabra de 4 bits aplicada a las entradas de datos se cargará en el registro seleccionado. R/W'=0 tambien desactiva los circuitos de salida, que están en estado triple, de manera que las salidas de datos se encuentren es estado de Hi-Z (alta impedancia) durante una operacion de escritura. La operación de escritura, desde luego, destruye la palabra que estaba almacenada antes en la dirección.

6.1.3.1.3. SELECCION DE CIRCUITO INTEGRADO.

Muchos circuitos de memoria tienen una o más entradas CS que se usan para activar o desactivar al circuito en su totalidad. En el modo "desactivado" todas las entradas y salidas de datos se desactivan (Hi-Z) de manera que no puedan realizarse operaciones de lectura ni de escritura. En este modo el contenido de la memoria no es afectado. La razón de tener entradas CS se hará importante cuando se combinen circuitos de memoria para obtener mayores memorias.

6.1.3.1.4. TERMINALES DE CONEXION COMUNES DE ENTRADA / SALIDA.

A fin de conservar terminales en un paquete de circuitos integrados (IC), los fabricantes a menudo combinan las funciones de entrada y salida de datos utilizando terminales de conexión comunes de entrada/salida. La entrada R/W' controla la función de estas terminales E/S. Durante una operación de lectura, las terminales E/S actúan como salidas de datos que reproducen el contenido de la localidad de dirección seleccionada. Durante una operación de escritura, las terminales E/S actúan como entradas de datos.

Podemos observar por qué se hace esto considerando al circuito de la figura 6.6. Con terminales de entrada y salida aparte, se requiere un total de 18 terminales de conexión (incluyendo tierra y fuente de energía). Con cuatro terminales comunes E/S, sólo se necesitan 14 terminales de conexión. El ahorro en el uso de conexiones se hace aún más significativo en circuitos con tamaño de palabra mayor.

La arquitectura que se ilustra en la figura 6.6 de una RAM de 64 x 4 será un tanto diferente para RAM de mayor capacidad. Los registros se dispondrán en una matriz como la que se muestra para la arquitectura de la ROM en la figura 6.2. Los decodificadores de direcciones seleccionarán la hilera y la columna del registro que esté siendo accesado para una operación de escritura o lectura. Esta arquitectura de matriz reduce el tamaño de los circuitos de decodificación que se requieren.

6.1.3.2. RAM ESTATICA.

La operación de la RAM que se ha venido analizando hasta ahora se aplica a una RAM estática (aquella que puede almacenar datos mientras se aplica energía al circuito). Las celdas de la memoria RAM estática son en esencia multivibradores biestables (flip-flops) que permanecerán en un estado determinado (almacenará un bit) indefinidamente, siempre y cuando no se interrumpa el suministro de energía al circuito. Más adelante describiremos las RAM dinámicas, que almacenan datos como cargas en capacitores. Con la RAM dinámica los datos almacenados desaparecerán gradualmente debido a la descarga del capacitor, de manera que se necesita refrescar en forma periódica los datos (o sea, cargar los capacitores).

6.1.3.2.1. DISTRIBUCION DE LA RAM ESTATICA.

Los circuitos integrados de la RAM son los que más frecuentemente se utilizan como una memoria interna de una computadora. La UCP (unidad central de procesamiento) efectúa en forma continua operaciones de lectura y escritura en su memoria a muy alta velocidad determinada por las limitaciones de la UCP. Los integrados de memoria que se sincronizan con la UCP tienen que ser lo suficientemente rápidos para responder a los comandos de lectura y escritura de la UCP y un diseñador de computadoras tiene que interesarse en las diversas características de la distribución de la RAM.

La nomenclatura de los diferentes parámetros de distribución variará de un fabriante a otro, pero el significado de cada parámetro es por lo general fácil de determinar a partir de los diagramas de distribución de la memoria en las hojas de datos de la RAM. La figura 6.7 muestra los diagramas de distribución de un ciclo completo de lectura y uno de escritura de un circuito RAM común.

Las formas de onda de la figura 6.7(a) muestra la forma en que las entradas de dirección, la entrada de selección de integrado, la entrada R/W' y las salidas de datos se comportan durante un ciclo de lectura. Nótese que al entrada R/W' permanece en ALTA en todo el ciclo de lectura. En muchos sistemas de meoria, R/W' normalmente se mantiene en ALTA y se lleva a BAJA sólo durante el ciclo de escritura. El ciclo de lectura empieza en to cuando las entradas de dirección cambia a la nueva dirección desde la cual

se leerán los datos; el ciclo de lectura termina en t3 cuando las entradas de dirección cambian a una dirección diferente para dar inicio al siguiente ciclo de lectura. Por lo tanto, el intervalo t0-t3 define el tiempo de ciclo de lectura, t_{RC}.

El tiempo de accesso, t_{ACC}, ocurre dentro del intervalo t_{RC} y representa el tiempo que se requiere para que el circuito de la memoria produzca salida de datos válidas. Por supuesto, la entrada CS' tiene que llevarse a BAJA. El parámetro de distribución t_{CO} es el tiempo que tardan las salidas de la memoria de ir de Hi-Z a datos válidos después de que CS' pasa a BAJA. El intervalo de tiempo t_{OD} es el tiempo que tardan las salidas en volver al estado desactivado (Hi-Z) después de que CS' regresa a ALTA. Por tanto, las salidas de la memoria contienen datos válidos entre t_1 y t_2, tiempo en el cual los datos se transfieren a otro circuito. Cualquier intento por transferir los datos de la memoria antes de t_1 producirá resultados inválidos. En muchos sistemas de microcomputadora la transición en sentido positivo de CS' se utiliza para cronometrar los datos en un registro de la UCP.

La figura 6.7(b) muestra un ciclo de escritura común. Las formas de onda incluyen las entradas de datos de memoria en lugar de las salidas de datos. El ciclo de escritura comienza en t_0, cuando las entradas de dirección cambian a la nueva dirección en la cual se escribirá. Este termina en t_4, de manera que el intervalo t_0-t_4 define el tiempo del ciclo de escritura, t_{WC}. La línea R/W' inicialente es ALTA y se lleva a BAJA sólo después de que la nueva dirección haya sido estable por un periodo de tiempo t_{AS}, llamado tiempo de preparación de las direcciones. Esto da tiempo a los decodificadores de direcciones de la RAM para responder a la nueva dirección.

La línea CS' se lleva a BAJA al mismo tiempo que R/W', ya que ambos se requieren para realizar una operación de escritura, CS' y R/W' deben mantenerse en BAJA en un tiempo t_W. Las entradas de datos en las que se escribirá la localidad de memoria direccionada se aplican en t_1. Los datos tienen que mantenerse estables (cuando menos) por un tiempo t_{DS}, tiempo de preparación de los datos, antes de que R/W' y CS' regresen a ALTA. Análogamente, las entradas de dirección tienen que mantenerse en un tiempo t_{AH}, tiempo de contención de la dirección. Si no se cumple alguno de estos requisitos de tiempo de preparación o contención, la operación de escritura no se efectuará adecuadamente.

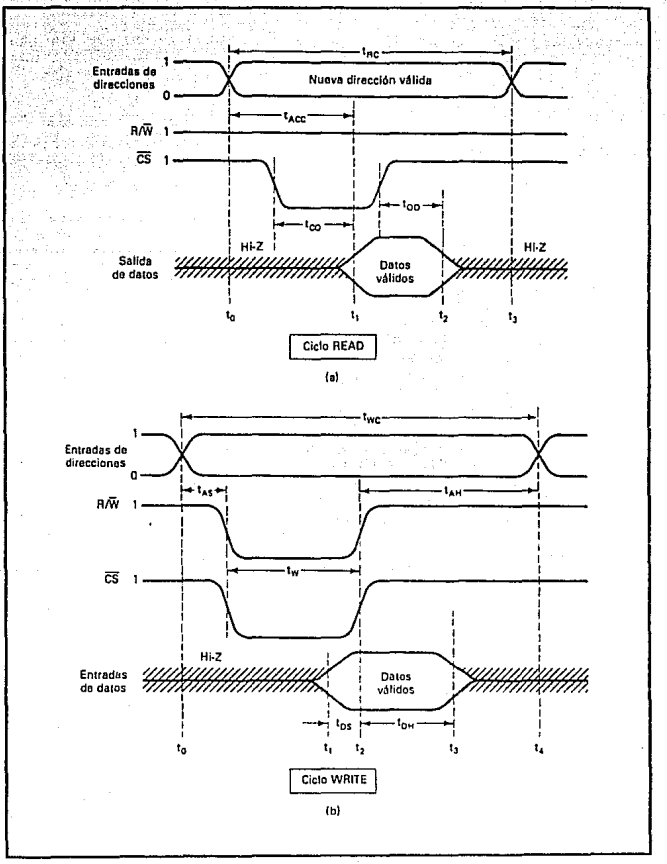


FIG. 6.7 DISTRIBUCION COMUN DE LA RAM ESTATICA: (A) CICLO READ; (B) CICLO WRITE.

6.1.3.3. RAM DINAMICA.

Una diferencia importante entre la RAM estática y la dinámica es que las celdas de memoria dinámica retienen datos sólo por un tiempo limitado (comúnmente 2ms, después del cual se pierden los datos). Esto requirere que los datos sean refrescados en intervalos regulares a fin de mantenerlos almacenados en la memoria. La necesidad de "refrescamiento" de los datos es una desventaja de la RAM dinámica. Sus ventajas son bajo consumo de energía y bajo costo (que son un resultado directo de la simplicidad de la celda de memoria). Una celda de memoria dinámica consta de un solo MOSFET y de un sólo capacitor MOS (comunmente de unos cuantos picofarads) para almacenar un solo bit (figura 6.8).

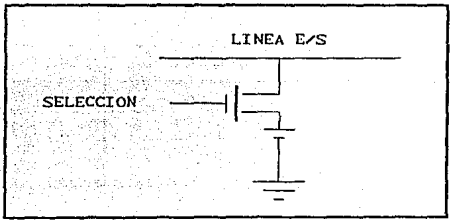


FIG. 6.8 CELDA DE MEMORIA RAM DINAMICA.

El MOSFET actua como un interruptor y un capacitor es el elemento de almacenamiento real. La operación de escritura básica se lleva a cabo aplicando un estado ALTO a la entrada SELECT para encender el MOSFET y completar la trayectoria entre la línea E/S y el capacitor. Durante una operación de escritura, la línea E/S se utiliza como línea de entrada de carga o descarga el capacitor para almacenar un 1 o bien un 0. Cuando la operación de escritura se completa, la entrada SELECT se hace BAJA y el MOSFET se apaga, abriendo esencialmente la trayectoria hacia el capacitor. En teoria, el capacitor contendrá su carga indefinidamente, pero en la práctica su carga desaparecerá después de 2ns.

Una operación de lectura se efectúa encendiendo el MOSFET con un estado ALTO en la entrada SELECT, después utilizando la línea E/S como línea de percepción de la salida. Ya que el capacitor está ahora esencialmente conectado a la línea E/S, su carga

almacenada determinará el voltaje que figura en la línea E/S. Este voltaje de salida se alimenta a un circuito amplificador que es parte del circuito de una RAM dinamica.

Debido a su estructura de celda simple, las RAM dinámicas tiene por lo general cuatro veces la densidad de las RAM estáticas. Esto permite cuatro veces la capacidad de memoria en una sola estructura o bien, alternativamente, requiere una cuarta parte del espacio en tablero para la misma cantidad de memoria. El costo por bit de almacenamiento en RAM dinámica es comunmente cuatro o cinco veces menor que para las RAM estáticas. Se logra todavía otro ahorro en costo debido a los requisitos menores de energía de una RAM dinámica, por lo general tres o seis veces menos que los de una RAM estática, que permiten al usuario emplear fuentes de energía menores y menos costosas. Estas ventajas de las RAM dinámicas se ilustran en la tabla 6.3, que compara dos integrados RAM mas modernos.

	2164 (DINAMICA)	2167 (ESTATICA)
Capacidad (bits)	64K	16K
Tiempo de acceso (ns)	150	70
Energía activa (mW)	300	625
Energía de transición (mW)	25	200
Costo por bit (milicentavos)	45	250

TABLA 6.3

Las RAM dinámicas tienen varias desventajas. Por lo general son más lentas que los dispositivos estáticos, como lo ejemplifica la comparación que se hace en la tabla 6.3. Algunas de las RAM dinámicas más antiguas requieren mas de un voltaje de suministro, en tanto que las estáticas utilizan solamente +5 V. Las RAM dinámicas más modernas emplean sólo +5 V.

6.1.3.3.1. ESTRUCTURA Y OPERACION DE LA RAM DINAMICA.

Muchas RAM dinámicas (DRAM) pueden visualizarse como una matriz de registros (celdas) de un solo bit como se ilustra en la figura 6.9. Aqui se tiene 4096 celdas dispuestas en una matriz de 64 x 64. Cada celda ocupa una hilera y una columna específicas en

la matriz. Se necesitan 12 entradas de dirección para seleccionar una de las celdas: los bits de dirección de orden inferior, Ao-A5 seleccionan la hilera y los bits de dirección de orden superior seleccionan la columna. Cada código de dirección de 12 bits selecciona una celda única para se leída o bien para escribirse en ella. La estructura que se muestra en la figura 6.9 es la de una DRAM de 4K x 1 (o sea, almacena 4096 palabras de un bit). Los circuitos DRAM se obtienen actualmente en capacidades de hasta 256K x 1. Ya que muchas DRAM tiene un tamaño de palabra de un bit, varios circuitos tienen que combinarse para producir palabras mayores.

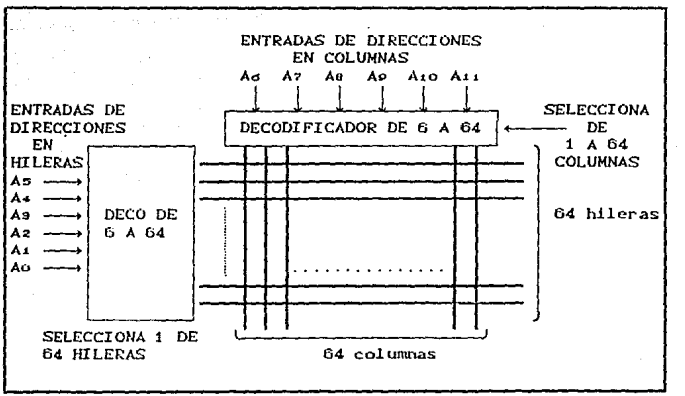


FIG. 6.9 DISPOSICION DE CELDAS DE LA RAM DINAMICA 4K X 1.

6.1.4. ARREGLO LOGICO PROGRAMABLE (PLA).

Ocasionalmente es posible que un circuito combinacional tenga condiciones no importa. Cuando se implementa con una ROM, una condición no importa se vuelve una entrada de dirección que nunca ocurrirá. Las palabras en las direcciones no importa no necesitan programarse y pueden dejarse en su estado original (todas en 0 o todas en 1). El resultado es que no se usan todos los patrones de bit disponibles en la ROM, lo cual puede considerarse como un

desperdicio de equipo disponible.

Para el caso donde el número de las condiciones no importa es excesivo, es más económico usar un segundo tipo de componente LSI llamado *arreglo lógico programable* o PLA. Un PLA es similar en concepto a una ROM; sin embargo, el PLA no proporciona la plena decodificación de las variables y no genera todos los minitérminos como en la ROM. En el PLA, el decodificador se reemplaza por un grupo de compuertas AND, cada una de las cuales puede programarse para generar un término producto de las variables de entrada. Las compuertas AND y OR dentro del PLA están fabricadas inicialmente con eslabones entre ellas. Las funciones booleanas específicas se implementan en la forma de suma de productos por la apertura de los eslabones apropiados y dejando las conexiones deseadas.

En la figura 6.10 se muestra un diagrama de bloques del PLA. Consta de n entradas, m salidas, k términos producto y m suma de términos. Los términos producto constituyen un grupo de k compuertas AND y los téminos suma constituyen un grupo de m compuertas OR. Los eslabones se insertan entre todas las n entradas y sus valores de complemento a cada una de las compuertas AND. Se proporcionan también eslabones entre las salidas de las compuertas AND y las entradas de las compuertas OR. Otro conjunto de eslabones en los inversores de salida permiten que se genere la función de salida ya sea en la forma AND-OR o en la forma AND-OR-inversa. Con el eslabón inversor en su lugar, el inversor se deriva, dando un AND-OR. Con el eslabón roto, el inversor se vuelve parte del circuito y se implanta la función en la forma AND-OR-inversa.

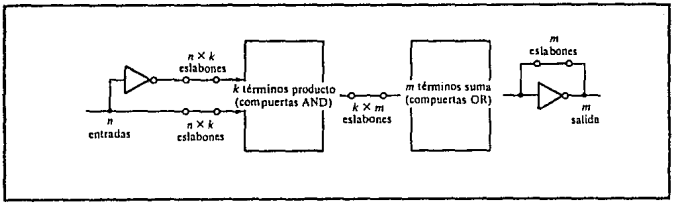


FIG. 6.10 DIAGRAMA DE BLOQUES DEL ARREGLO PLA.

El tamaño del PLA se especifica por el número de entradas. El número de términos producto y el número de salidas (el número de

los términos suma, es igual al número de salidas). Un PLA típico tiene 16 entradas, 48 términos producto y 8 salidas. El número de eslabones programado es $2^n xk+kxm+m$, tanto que el de una ROM es $2^n xm$.

En la figura 6.11 se muestra la construcción interna de un PLA específico. Tiene tres entradas, tres términos producto y dos salidas. Dicho PLA es demaciado pequeño para tener disponibilidad comercial; se presenta aquí sólo con fines de demostración. Cada entrada y su complemento se conectan a través de eslabones a las entradas de todas las compuertas AND. Las salidas de las compuertas AND se conectan a través de eslabones a cada entrada de las compuertas OR. Se proporcionan dos eslabones más con los inversores de salida. Mediante la rotura de eslabones selecionados y la colocación de otros en su lugar, es posible implantar funciones booleanas en su forma de suma de productos.

Como una ROM, el PLA puede ser programable por máscara o programable en campo. Con un PLA programable en mascara, el cliente debe someter una tabla de programa PLA al fabricante. Esta tabla la usa el vendedor para producir un PLA hecho sobre pedido que tenga las trayectorias internas requeridas entre entradas y salidas. Un segundo tipo de PLA disponible se conoce se conoce como arreglo lógico programable en campo o FPLA. El FPLA puede programarlo el usuario mediante ciertos procedimientos recomendados. Estan dsponibles unidades comerciales de hardware programable para usarse junto con ciertos FPLA.

6.1.4.1. TABLA DE PROGRAMA PLA.

El uso de un PLA debe considerarse para circuitos combinacionales que tengan un gran número de entradas y salidas. Es una superior a una ROM para circuitos que tienen un gran número de condiciones no importa. El ejemplo que se presenta a continuación demuestra como se programa un PLA. Cuando el lector vea el ejemplo debe tener en consideración que un circuito tan simple no requiere un PLA por que puede implementarse en forma más económica con compuertas SSI.

Considerese la tabla de verdad del circuito combinacional, que se muestra en la figura 6.12(a). Aunque una ROM implementa un circuito combinacional en la forma de suma de minitérminos, un PLA implementa las funciones en su forma de suma de productos. Cada producto término en la expresión requiere una compuerta AND.

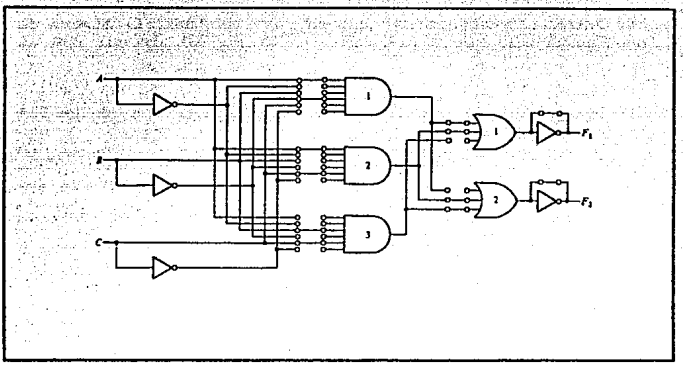


FIG. 6.11 ARREGLO PLA CON TRES ENTRADAS, TRES TERMINOS PRODUCTO Y DOS SALIDAS; IMPLEMENTA EL CIRCUITO COMBINACIONAL ESPECIFICADO EN LA FIGURA 6.12.

Ya que el número de compuertas AND en un PLA es finito, es necesario simplificar la función a un número mínimo de términos producto con objeto de minimizar el número de compuertas AND que se utilicen. Las funciones simplificadas en suma de productos se obtienen mediante los mapas en la figura 6.12(b):

$$F_1 = AB' + AC$$

$$F_2 = AC + BC$$

Hay tres distintos términos producto en este circuito combinacional: AB', AC y BC. El circuito tiene tres entradas y dos salidas; de modo que el PLA de la figura 6.11 puede usarse para implementar este circuito combinacional.

La programacón del PLA significa que se especifican las trayectorias en su patrón AND-OR-NOT. Una tabla típica de programa PLA se muestra en la figura 6.12(c). Consta de tres columnas. La primera columna lista los términos producto numéricamente. En la segunda columna se especifican las trayectoria requeridas entre las entradas y las compuertas AND.

La tercera columna especifica las trayectorias entre las compuertas AND y las compuertas OR. Bajo cada variable de entrada, se escribe una T (de la inicial en ingles de verdadero) si la salida inversora va a derivarse, y C (de la inicial de complemento) si la función va a complementarse con la salida inversora. Los términos booleanos que se listan a la izquierda no son parte de la tabla; se incluyen solo como referencia.

Para cada término producto, las entradas se marcan con 1, 0, o - (guión). Si una variable en el producto término aparece en su forma normal (sin '), la variable correspondiente de entrada se marca con un 1. Si aparece complementada (con '), la variable de entrada correspondiente se marca con un 0. Si la variable está ausente en el término producto se marca con un guión. Cada término producto se asocia con una compuerta AND. Las trayectorias entre las entradas y las compuertas AND se especifican bajo la columna con encabezado entradas. Un 1 en la columna de entrada especifica una trayectoria de la entrada correspondiente a la entrada de la compuerta AND que forma el término producto. Un 0 en la columna de entrada especifica una trayectoria desde la entrada complementada correspondiente a la entrada de la compuerta AND. Un guión especifica que no hay conexión. Los eslabones apropiados están rotos, y los que se dejan en su lugar forman las trayectorias deseadas, como se muestra en al figura 6.11. Se supone que las terminales abiertas en la compuerta AND se comportan como una entrada 1.

Las trayectorias entre las compuertas AND y OR se especifican bajo la columna con encabezado de salidas. Las variables de salida se marcan con 1 para todos los términos producto que formulan la función. En el ejemplo de la figura 6.12 se tiene:

$$F_1 = AB' + AC$$

de modo que F_1 está marcada con 1 para los términos producto 1 y 2 con un guión para el término producto 3. Cada término producto que tiene 1 en al columna de salidas requiere una trayectoria desde la compuerta correspondiente AND a la compuerta de salida OR. Los que están marcados con un guión especifican que no hay conexión. Por último, una salida T (verdad) determina que el eslabón a través de la salida inversora permanezca en su lugar, y una C (complemento) especifica que el eslabón correspondiente está roto. Las trayectorias internas del PLA para este circuito se muestra en la figura 6.11. Se supone que una terminal abierta en una compuerta OR se comporta como un 0 y que un corto circuito a través de la salida inversora no daña el circuito.

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

(a) TABLA DE VERDAD

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

$F_1 = AB' + AC$

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

$F_2 = AC + BC$

(b) SIMPLIFICACION DE MAPA

	TEMINO PRODUCTO	ENTRADAS A	B	C	SALIDAS F_1	F_2	
AB'	1	1	0	-	1	-	
AC	2	1	-	1	1	1	
BC	3	-	1	1	-	1	
					T	T	T/C

(c) TABLA DE PROGRAMA DEL PLA

FIG. 6.12 PASOS REQUERIDOS EN LA IMPLEMENTACION DEL PLA.

Cuando se diseña un sistema digital con un PLA no es necesario mostrar las conexiones internas de la unidad como se hizo en la figura 6.11. Todo lo que se necesita es una tabla de programa PLA mediante la cual puede programarse el PLA para sumunistrar las trayectorias apropiadas.

Cuando se implementa un circuito combinacional con PLA, debe emprenderse una investigacion cuidadosa con el objetos de reducir el numero total de terminos producto distintos, ya que un PLA dado pede tener un numero finito de términos AND. Esto puede

hacerse por la simplificacion de cada funcion a un numero minimo de terminos. El numero de literales en un termino no es importante ya que se tienen disponibles todas las variables de entrada. Tanto el valor verdadero como el complemento de la funcion deben simplificarse para ver cual puede expresarse con menos terminos producto y cual proporciona terminos producto que son comunes a otras funciones.

6.2. UNIDAD DE MEMORIA

Los registros en una computadora digital pueden clasificarse ya sea en el tipo operacional o de almacenamiento. Un registro operacional es capaz de almacenar informacion binaria en sus flip-flops y, ademas, tiene compuertas combinacionales capaces de realizar las tareas de procesamiento de datos. Un registro de almacenamiento se utiliza solo para almacenamiento temporal de la informacion binaria. Esta informacion no puede alterarse cuando se transfiere dentro y fuera del registro. Una *unidad de memoria* es una coleccion de registros de almacenamiento junto con los circuitos asociados necesarios para transferir la informacion dentro y fuera de los registros. Los registros de almacenamiento en una unidad de memoria se denominan registros de memoria.

La mayor parte de los registros en una computadora digital, son registros de memoria, a los cuales se transfiere informacion para su almacenamiento y de los cuales esta disponible para procesarla cuando sea necesario. En la unidad de procesamiento, en forma comparativa, se encuentran pocos registros operacionales. Cuando tiene lugar el procesamiento de datos, la informacion de registros seleccionados en la unidad de memoria se transfiere primero a los registros operacionales en la unidad de procesamiento. Los resultados intermedios y finales obtenidos en los registros operacionales se transfieren a registros de memoria seleccionados. En forma similar, la informacion binaria recibida de los dispositivos de entrada se almacena primero en los registros de memoria; la informacion transferida a los dispositivos de salida se toma de registros en la unidad de memoria.

El componente que forma las celdas binarias de los registros en una unidad de memoria debe tener ciertas propiedades basicas; las mas importantes de estas son:

(a) Debe tener una propiedad confiable de dos estados para la

representacion binaria.

(b) Debe tener tamaño pequeño.

(c) El costo por bit de almacenamiento debe ser bajo como sea posible.

(d) El tiempo de acceso a un registro de memoria debe ser razonablemente rapido.

Los componentes de unidades de memoria son núcleos magnéticos, circuitos integrados, superficies magneticas en cintas, discos, etc.

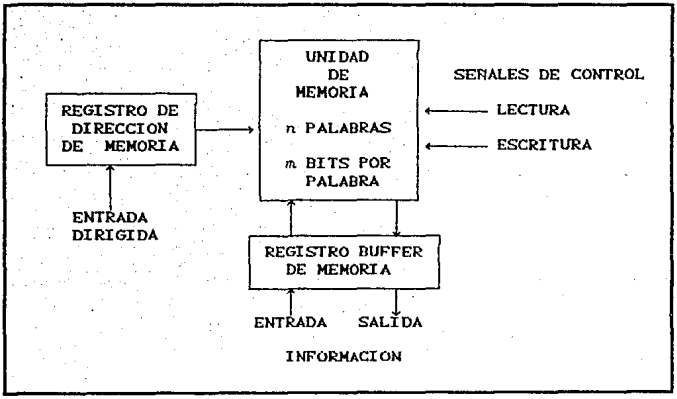


FIG. 6.13 DIAGRAMA DE BLOQUES DE UNA MEMORIA MOSTRANDO LA COMUNICACION CON SU MEDIO AMBIENTE.

Una unidad de memoria almacena información binaria en grupos llamados palabras y cada palabra se almacena en un registro de memoria. Una palabra en memoria es una entidad de n bits que se mueve hacia adentro y hacia afuera del almacen como una unidad.

Una unidad de memoria puede representar un operando, una instrucción, un grupo de caracteres alfanuméricos, o cualquier información codificada en binario. La comunicación entre la unidad de memoria y su medio ambiente se logra a través de dos señales de control y dos registros externos. Las señales de control especifican la dirección de la transferencia requerida, esto es, ya sea que una palabra se almacene en un registro de memoria o que una palabra previamente almacenada se transfiera fuera del registro de memoria. Un registro externo especifica el registro particular de memoria escogido entre los miles disponibles; el otro especifica la configuración particular de bits en la palabra en cuestión. Las señales de control y los registros se muestran en el diagrama de bloques de la figura 6.13.

6 . 3 . DIRECCIONAMIENTO.

El *registro de dirección* de memoria especifica la memoria de palabra seleccionada. Cada palabra en una memoria está asignada a un número de especificación que principia desde 0 hasta el número máximo de palabras disponibles. Para comunicarse con una palabra de memoria específica, su número de localización, o dirección, se transfiere al registro de dirección. Los circuitos internos de la unidad de memoria aceptan esta dirección del registro y abren las trayectorias necesarias para seleccionar la palabra llamada. Un registro de dirección con n bits puede especificar hasta 2^n palabras de memoria. Las unidades de memoria de computadora pueden variar desde 1024 palabras, que requieren un registro de dirección de 10 bits , a 1,048,576 = 2^{20} palabras, que requieren un registro de dirección de 20 bits.

Las dos señales de control aplicadas a la unidad de memoria se denominan lectura y escritura. Una señal de escritura especifica una función de transferencia interna; una señal de lectura especifica una función de transferencia externa. Cada una se referencia desde de la unidad de memoria. Al aceptar una de las señales de control, los circuitos interno de la unidad de memoria proporcionan la función deseada. Ciertos tipos de unidades de almacenamiento, debido a las características de su componente, destruyen la información almacenada en una celda cuendo el bit en esa celda se lee al exterior. Dicha unidad se dice que es una memoria de lectura destructiva, en contraposición a una memoria no destructiva donde la información permanece en la celda después de que se lee al exterior. En cualquier caso, la información anterior siempre se destruye cuando se escribe información nueva. La secuencia del control interno en una memoria de lectura

destructiva debe proporcional señales de control que provoquen que la palabra se restablezca en sus celdas binarias si la aplicacion exige una funcion no destructiva.

La transferencia de información hacia y desde los registros en memoria y el medio externo se comunica a través de un registro comun llamado *registro buffer* de memoria. Cuando la unidad de memoria recibe una señal de control para escritura, el control interno interpreta el contenido de registro buffer como la configuración de bits de la palabra que va a almacenarse en un registro de memoria. Con una señal de control de lectura, el control interno envía la palabra de un registro de memoria al registro buffer. En cada caso el contenido del registro de direccion especifica el registro particular de memoria referenciado para escritura o lectura.

C O N C L U S I O N E S

Al finalizar el presente trabajo, y dados los elementos y conocimientos necesarios para ser capaz de entender y analizar la mayor parte de problemas relacionados con la electrónica digital, se puede concluir lo siguiente:

El conocer y dominar la electrónica y el diseño digital contribuye al desarrollo profesional del ingeniero o técnico orientados a esta area, ya que con ello se puede diseñar cualquier elemento o controlador digital que puede ser aplicado en muchas tareas industriales y domesticas de uso cotidiano.

Desafortunadamente en México no existe mucho campo de desarrollo en esta área, sin embargo, el hecho de conocer el diseño digital nos da una gran vision sobre la lógica del diseño en muchas areas más. Personalmente, el conocimiento del diseño y la lógica digital me permitio entender más facilmente el diseño de sistemas de software.

Por último, es una necesidad imperiosa complementar todo este material con un laboratorio en donde se lleven a cabo practicas y proyectos.

A P E N D I C E .

AI . SISTEMAS DE NUMERACION

AI . 1 . REPRESENTACION DE LOS NUMEROS.

Es importante cuando se manejan diversas cantidades cuyos valores se pueden representar con eficiencia y exactitud. Existen basicamente dos maneras de representar el valor numérico de las cantidades : la "analogica" y la "digital".

AI . 1 . 1 . REPRESENTACIONES ANALÓGICAS.

En la representacion analógica, una cantidad se denota por medio de otra que es directamente proporcional a la primera. Un ejemplo de esto es el velocimetro de un automóvil, en el cual la deflexion de la aguja es proporcional a la velocidad a que se desplaza el auto. La posicion angular de la aguja representa la velocidad del auto y la aguja sigue cualquier cambio que ocurra conforme el automovil acelera o desacelera.

Las cantidades analogicas como las que se citaron antes tienen una caracteristica importante: pueden variar gradualmente sobre un intervalo continuo de valores. La velocidad del automovil puede tener cualquier valor entre cero y, por decir algo, 100 km/h.

AI . 1 . 2 . REPRESENTACIONES DIGIALES.

En la representacion digital las cantidades se denotan no por cantidades proporcionales, sino por simbolos denominados "digitos". Para poner un ejemplo, consideremos el reloj digital, el cual da la hora en forma de digitos decimales que representan horas, minutos y segundos. Como se sabe la hora varia continuamente, pero la lectura del reloj digital no cambia de esta misma manera; en su lugar, varia en etapas de uno por segundo. En otras palabras, esta representacion digital de la hora varia en etapas "discretas", en comparacion con la representacion analogica de la hora que da un reloj de pulso, donde la lectura del cuadrante varia continuamente.

La diferencia principal entre las cantidades analogicas y las digitales se pueden enunciar en forma simple de la manera siguiente :

analógico = continuo
digital = discreto

Debido a la naturaleza discreta de las representaciones digitales, no existe ambiguedad cuando se lee el valor de una cantidad digital, en tanto el valor de una cantidad analogica con frecuencia se presta a interpretacion.

AI . 2 . SISTEMAS NUMERICOS.

Un sistema numerico (S.N.) consiste de un conjunto ordenado de símbolos, llamados digitos con relaciones definidas para adicion, sustraccion, multiplicacion y division. La base del sistema numerico es el numero total de digitos permitidos en dicho sistema.

Como ejemplos de sistemas numericos, podemos mencionar los que se muestran en la fig. A.1.

Cuando el valor que representa cada digito depende de la posicion que ocupa dentro de la cantidad expresada decimos que el S.N. es un SISTEMA POSICIONAL o SISTEMA DE MULTIPLICACIONES POSICIONALES.

RAIZ	NOMBRE	DIGITOS USADOS EN EL SISTEMA DE NUMERACION
1	Unario	1
2	Binario	0,1
3	Terciario	0,1,2
4	Cuaternario	0,1,2,3
8	Octal	0,1,2,3,4,5,6,7
10	Decimal	0,1,2,3,4,5,6,7,8,9
12	Duodecimal	0,1,2,3,4,5,6,7,8,9,a,b
16	Hexadecimal	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

FIG. A.1 SISTEMAS NUMERICOS.

En cualquier sistema posicional, una cantidad N compuesta por una parte entera y una parte fraccionaria se escribe como :

$$N = (a_{n-1}\, a_{n-2}\, a_{n-3} \ldots a_1\, a_0 \,.\, a_{-1}\, a_{-2} \ldots a_{-m})_b$$

parte entera — parte fraccionaria

donde :

b = base del S.N.
ai = es cualquier dígito del S.N.
n = numero de dígitos enteros
m = numero de dígitos fraccionarios
an-1 = digito mas significativo
a-m = digito menos significativo

En un sistema posicional, el numero N también lo podemos escribir como :

$$N = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \ldots + a_1 b^1 + a_0 b^0 + a_{-1}b^{-1} + \ldots + a_{-m}b^{-m}$$

que es la llamada NOTACION POLINOMIAL, para indicar la ausencia de una potencia de la base, ponemos ai = 0.

A continuación mencionaremos algunos S.N. posicionales.

Sistema Unario : Base 1, unico digito, el símbolo 1. El sistema unario es el S.N. mas antiguo y el mas simple, era utilizado por el hombre primitivo para contar, la forma en que lo empleaba

consistía en marcar una rayita por cada unidad contada.

Sistema Decimal : Base 10, tiene los dígitos 0,1,2,3,4,5,6,7,8,9 el sistema decimal es obviamente el mas familiar y conocido por nosotros pues es el que empleamos normalmente en nuestra vida cotidiana.

Sistema Binario : Base 2, tiene los dígitos 0,1, es un sistema posicional. A los dígitos binarios se les acostumbra llamar BIT o bitio (del ingles Binary Digit).

Sistema Octal : Base 8, tiene los dígitos 0,1,2,3,4,5,6,7 es un sistema posicional.

Sistema Hexadecimal : Base 16, tiene los dígitos 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F, es un sistema posicional.

AI . 3 . CONVERSION DE BASE M A BASE 10.

Es importante saber convertir números de una base M a la base 10, porque esta última es la base a la cual pertenece nuestro sistema de numeración, y por lo tanto, con el que estamos mas familiarizados.

La tabla A.1 muestra las equivalencias entre los números de diferentes sistemas numéricos.

Podemos convertir el número N de base b al sistema decimal si escribimos al número N en la forma de notación polinomial y realizamos las operaciones indicadas

$$N = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \ldots + a_1b^1 + a_0b^0 + a_{-1}b^{-1} + \ldots + a_{-m}b^{-m}$$

donde todos los números del lado derecho de la igualdad pertenecen al sistema decimal.

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

TABLA A.1 EQUIVALENCIA ENTRE SISTEMAS NUMERICOS.

A1.3.1 CONVERSION DE BINARIO A DECIMAL.

Esta conversión se puede realizar usando la ecuación :

$$N = \sum_{i=-m}^{n-1} a_i (2)$$

Ejemplo : convertir 1111001.11011 a base 10 utilizando la descomposición polinomial.

$$Nio = 1x2^{6} + 1x2^{5} + 1x2^{4} + 1x2^{3} + 0x2^{2} + 0x2^{1} + 1x2^{0} + 1x2^{-1} + 1x2^{-2} + 0x2^{-3} + 1x2^{-4} + 1x2^{-5}$$

$$Nio = 64 + 32 + 16 + 8 + 1 + 0.5 + 0.25 + 0.0625 + 0.03125$$

$$Nio = 121.84375$$

AI . 3 . 2 CONVERSION DE OCTAL A DECIMAL

De la misma manera que el inciso anterior, la conversion sera realizada con base a la ecuacion :

$$N = \sum_{i=-m}^{n-1} ai\ (8)$$

Ejemplo : Dado el numero octal 427.21 convertir a su equivalente numero decimal.

$$N_{10} = 4x8^2 + 2x8^1 + 7x8^0 + 2x8^{-1} + 1x8^{-2}$$
$$= 256 + 16 + 7 + 0.25 + 0.015625$$
$$= 279.265625$$

AI . 3 . 3 CONVERSION DE HEXADECIMAL A BASE 10.

Para convertir en base 10 numeros en base 16 se tiene :

$$N = \sum_{i=-m}^{n-1} ai\ (16)$$

Ejemplo : convertir el numero hexadecimal 12AF a su equivalente en base 10.

$$N_{10} = 1x16^3 + 2x16^2 + Ax16^1 + Fx16^0$$

pero como debemos de tener todos los numeros del miembro derecho en el sistema decimal, recordamos que $(A)_{16} = (10)_{10}$ y que $(F)_{16} = (15)_{10}$, luego

$$N_{10} = 1x16^3 + 2x16^2 + 10x16^1 + 15x16^0$$
$$= 4096 + 512 + 160 + 15$$
$$= 4783$$

AI . 4 . CONVERSION DE BASE 10 A BASE M.

En este metodo, el número dado se divide sucesivamente por la base a la cuál se quiere convertir y el número se forma con los residuos de estas divisiones.

Supongase que se tiene que convertir un número decimal con parte entera y fraccional a binario.

Para convertir un numero N decimal, formado por parte entera y parte fraccionaria, a cualquier otra base, convertimos la parte entera a la base deseada y por separado convertimos la parte fraccionaria a la misma base y escribimos :

N = Nparte entera + Nparte fraccionaria

Primeramente tomamos la parte entera y la dividimos entre 2, así seguimos hasta que ya no sea posible realizar la division con numeros enteros, es decir, cuando encontremos un cociente igual a cero. Al ir efectuando estas divisiones vamos guardando los residuos encontrados y después los escribimos en el orden inverso al que los encontramos.

Para convertir la parte fraccionaria de una cantidad del sistema decimal a cualquier otro sistema se efectuan los siguientes pasos :

i) La parte fraccionaria se separa del número dado.

ii) Se multiplica por la base.

iii) Si el resultado es una parte entera y una fraccional se separa la parte entera y se procede de la misma manera, hasta obtener unicamente parte entera; de no ser posible, se realiza la mejor aproximación posible.

iv) El número estará formado por todas las partes enteras obtenidas.

Ejemplo : convertir (1025.25) a binario.

La parte entera es :

1025	2
512	1
256	0
128	0
64	0
32	0
16	0
8	0
4	0
2	0
1	0
0	1

Como habiamos visto, $1025_{10} = 10000000001_2$.

La parte fraccional es :

2	.25
0	.50
1	.00

Por lo tanto $1025.25_{10} = 10000000001.01_2$.

En base a los conceptos anteriores, ya podemos convertir un numero en base 10 a cualquier otra base, puesto que las reglas enunciadas anteriormente son las mismas.

Ejemplo : convertir 27.215_{10} a octal.

La parte entera es :

27	8
3	3
0	3

entonces $27_{10} = 33_8$

La parte fraccionaria es :

8	.215
1	.72
5	.76
6	.08
0	.64
5	.12
0	.96
7	.68
5	.44

entonces $.215_{10} = .15605075_8$

Por lo tanto $27.215_{10} = 33.15605075_8$

Ejemplo : convertir 842.25_{10} a hexadecimal.

La parte entera es :

842	16	
52	10	A
3	4	4
0	3	3

entonces $842_{10} = 34A_{16}$

La parte fraccionaria es :

16	.25
4	.00

entonces $.25_{10} = .4_{16}$

Por lo tanto $842.25_{10} = 34A.4_{16}$.

AI.5 CONVERSION DE BASE BINARIA A OCTAL Y VICEVERSA.

La base 8 (octal) y la base 16 (hexadecimal) son importantes para nosotros debido a su relacion con la base 2 (binaria). Cada digito octal corresponde a 3 bits y por cada 3 bits tenemos un digito octal. Analogamente cada digito hexadecimal corresponde a 4 bits y por cada 4 bits se tiene un digito hexadecimal. Debido a esta relacion podemos efectuar mas facilmente conversiones entre estos sistemas.

Conversión de octal a binario: La ventaja principal del sistema numerico octal es la facilidad con la cual puede realizarse la conversion entre números binarios y octales. La conversión de octal a binario se lleva a cabo convirtiendo cada digito octal en su equivalente binario de 3 bits. Los 8 digitos posibles se convierten como se indica en la siguente tabla :

Digito octal	0	1	2	3	4	5	6	7
Digito binario	000	001	010	011	100	101	110	111

TABLA A.2 EQUIVALENCIA OCTAL-BINARIO.

Por medio de estas conversiones cualquier digito octal se convierte en binario convirtiendolo de manera individual. Por ejemplo, podemos convertir 472_8 en binario de la siguiente manera:

4 7 2

↓ ↓ ↓

100 111 010

Por lo tanto $472_8 = 100111010_2$

Para convertir un número binario en número octal separamos el número binario en grupos de 3 bits a partir del punto binario (tanto a la derecha como a la izquierda del punto) en caso de no acompletarse grupos de 3 bits los acompletamos agregando los ceros necesarios y despues asignamos el digito octal correspondiente a

cada grupo.

Un ejemplo se muestra a continuación. El número binario contiene parte entera y parte fraccional.

001,010,011.110,110

Las comas indican separación de tercias, en tanto que los ceros mas pequeños indican el complemento a ceros; de esta manera cada tercia de bits representa un dígito en octal, tal como se ilustra :

```
 001  010  011 . 110  110
  ↓    ↓    ↓   .  ↓    ↓
  1    2    3   .  6    6
```

Por lo tanto $1010011.110110_2 = 123.66_8$

Para realizar el procedimiento inverso, es decir, pasar de octal a binario, cada dígito se descompone a su equivalente en binario, para integrar despues el numero en su conjunto.

Por ejemplo: sea el número $750.34_8 = 111101000.011100_2$ donde los ceros en la parte derecha del número fraccional pueden ser omitidos.

$$750.34_8 = 111101000.0111_2$$

AI . 6 . CONVERSION DE BASE BINARIA A HEXADECIMAL Y VICEVERSA.

Similarmente a lo expuesto en el apartado anterior, es posible realizar la conversion de base binaria a base hexadecimal. Sabemos que con 4 dígitos binarios es posible formar los 16 numeros que componen la base hexadecimal.

Ejemplo : sea el número 1010011111.1110012 obtener su equivalente hexadecimal.

```
0010 1001 1111 . 1110 0100
  ↓    ↓    ↓      ↓    ↓
  2    9    F  .   E    4
```

por lo tanto, $1010011111.11100100_2 = 29F.E4_{16}$

Por último veremos la conversión de base hexadecimal a base binaria. Para hacer esta conversión, cada letra o dígito que compone al número hexadecimal, se convierte a una cuarteta de bits.

Ejemplo : convertir $AF.16C_{16}$ a base binaria

```
  A    F  .  1    6    C
  ↓    ↓     ↓    ↓    ↓
1010 1111 . 0001 0110 1100
```

por lo tanto, $AF.16C_{16} = 10101111.000101101100_2$

AI.7. OPERACIONES ARITMETICAS CON NUMEROS NO SIGNADOS.

Como se había mencionado anteriormente un sistema numérico consiste de un conjunto ordenado de símbolos llamados dígitos para los cuales se tienen definidas ciertas reglas de asociación. Podemos asociar dos o mas dígitos mediante las operaciones aritméticas elementales, es decir en un sistema numérico se tienen definidas las operaciones de suma o adición, resta o diferencia, producto o multiplicación y división.

AI.7.1. SUMA BINARIA.

Esta operación es muy similar a la suma decimal.

Para sumar 2 números binarios, lo primero que tenemos que recordar es que en el sistema binario solo tenemos 2 dígitos el 0 y el 1 y después seguimos las siguientes reglas:

```
   0      0      1      1      primer sumando
 +      +      +      +
   0      1      0      1      segundo sumando
 ---    ---    ---    ---
   0      1      1     10      suma
                       ↑
                    carry, acarreo
```

En el caso de " 1 + 1 " como el resultado, ya no se puede representar solamente por medio de un dígito, es decir, en unidades binarias empleamos 2 dígitos, es decir, pasamos a las decenas binarias y escribimos 10 o también decimos que 1 + 1 = 0 y llevamos 1, a esto 1 le llamamos el acarreo o el transporte a las decenas (carry).

Cuando se presenta un acarreo, este debe colocarse en la columna de mayor peso mas cercana y sumarse, veamos un ejemplo :

```
  1 1 1 1 1    ← acarreos
  1 1 1 1 0 1
+       1 1 1
-------------
1 0 0 0 1 0 0 ← suma
```

De lo anterior cabe hacer notar, que el acarreo solo se presenta cuando la suma es igual o mayor a la raiz R del sistema de numeración empleado.

AI . 7 . 2. RESTA BINARIA.

Para restar 2 numeros en binario seguimos las siguientes reglas :

```
1 - 0 = 1
1 - 1 = 0
0 - 0 = 0
10  1 - 1 o 0 - 1 = 1 y borrow = 1
```

Otra manera de ver la substraccion : 0 - 1 es que como con el dígito 0 no "alcanza" para substraerle 1 entonces hay que pedir "prestado" (borrow) una unidad y así, 0 - 1 se convierte en 10-1,

donde 10 pasa a ser borrow; y como hay que pagar a la otra columna esto se hace con un "1" que fue el que se "pidió prestado" y el cual se denomina "pay back".

En forma mecanizada podemos decir que siempre que se tenga la diferencia 0 - 1, se colocará 1 en la columna correspondiente, y el pay back en la siguiente columna mas significativa.

El siguiente ejemplo ilustra este método :

Ejemplo :

```
  1 0 1 0 0 1 1      minuendo
- 1 1 1 1            pay back
    1 0 1 0 1        substraendo
-------------
  0 1 1 1 1 1 0      diferencia
```

AI . 7 . 3 . MULTIPLICACION BINARIA.

La multiplicación entre 2 números binarios la vamos a realizar en forma parecida a la multiplicación decimal, pero ahora seguimos las siguentes reglas :

```
0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1
```

Para la multiplicación binaria se tienen las mismas reglas que para la decimal, es decir, se multiplica cada digito del multiplicador con el multiplicando y luego se suman los productos parciales.

Ejemplo:

```
      1 0 1 1        Multiplicando
    x 1 0 1          Multiplicador
  -------------
      1 0 1 1        Producto parcial
    0 0 0 0          Producto parcial
  1 0 1 1            Producto parcial
  -------------

  1 1 0 1 1 1        Producto
```

AI.7.4. DIVISION BINARIA.

Para efectuar la división de dos numeros binarios lo primero que hacemos es colocarlos en la misma forma en que los colocamos cuando queremos hacer una división en decimal, despues separamos tantas cifras en el dividendo como cifras tenemos en el divisor, y vemos cuantas veces "cabe" el divisor en las cifras que separamos en el dividendo (obviamente cabe una vez o ninguna ya que solo tenemos dos opciones 1 o 0), el numero encontrado es el cociente, el cual se va a multiplicar por el divisor y se coloca debajo del dividendo, se efectua la resta de ellos y se "baja" el siguiente numero del dividendo, ahora nos preguntamos cuantas veces "cabe" el divisor en el numero que acabamos de encontrar, con lo cual obtenemos el siguiente número del cociente, etc., seguimos este procedimiento (en forma analoga que en el sistema decimal) hasta que encontramos un residuo igual a 0 ,o tambien que este se empiece a repetir lo cual indicara que tambien se repetira el cociente y entonces se tratara de un numero con fraccion periodica.

Podemos efectuar la comprobación de una división binaria mediante la conocida regla "DIVIDENDO es igual a DIVISOR por COCIENTE mas RESIDUO".

En la division binaria puede ser que no exista un resultado exacto. Es por ello, que se acostumbra expresar en estos casos el resultado como en cociente y el residuo.

Ejemplo: Encontrar el cociente entero de 100011010 entre 1101.

```
                 010101
            ------------
1101  | 100011010
        - 1101
        ------
         0010010
       -  1101
        -------
          0010110
        -  1101
         -------
           01001
```

comprobación :

```
    010101        cociente
  x 1101          divisor
 --------
   010101
  000000
 010101
010101
---------
100010001
+   01001       residuo
---------
100011010       dividendo
```

AI.8. SISTEMAS DE NUMERACION COMPLEMENTARIOS.

Dentro de ciertos sistemas electrónicos, como lo son las computadoras digitales es necesario ahorrar circuiteria y por lo tanto, espacio, esto es posible lograrlo cuando una unidad aritmética puede ser reducida. Esto es mediante el sistema complementario de numeracion. Como mas adelante veremos, el empleo de un sistema de numeracion complementario permite realizar las substracciones como sumas.

Existen varias formas de complementar un numero en cualquier base, pero por ser de mayor interes trabajaremos con los distintos complementos que se pueden hacer a la base binaria comenzando por el mas simple.

AI.8.1 COMPLEMENTO A 1's.

El complemento a 1's (N1') de un numero N teniendo n enteros y m bits fraccionales, pueder ser expresado como :

$$N1' = 2^n - \left(\sum_{i=-m}^{n-1} b_i 2^i \right) - 2^{-m}$$

donde b_i es el coeficiente binario (0 ó 1) del iésimo bit, como :

$$N = \left(\sum_{i=-m}^{n-1} b_i 2^i \right)$$

entonces,

$$N1' = 2^n - N_2 - 2^{-m}$$

Ejemplo : Hallar el complemento a 1's de $(25)_{10}$.

$(25)_{10} = 11001_2$

$$\begin{aligned} N1' &= 2^5 - 11001_2 - 1 \\ &= 100000 - 11001 - 1 \\ &= 00110 \end{aligned}$$

Como se observa del ejemplo anterior en el complemento a 1's de cualquier numero binario solamente se intercambian los 1's por 0's y viceversa.

AI.8.2. REPRESENTACION A COMPLEMENTO A 2's.

El complemento a 2's (N2') de un numero N, puede ser obtenido sumando uno al valor absoluto de un numero binario que es de orden inferior que el bit mas significativo (MSB).

$$N2' = 2^n - \sum_{i=m}^{n-1} bi\ 2^i$$

o simplemente,

$$N2' = 2^n - N2$$

Ejemplo : determinar el complemento a 2's de 10111010100

$$N2' = 2^{11} - 10111010100_2$$
$$= 100000000000 - 10111010100$$
$$= 01000101100$$

Otra forma de resolver el problema anterior, consiste en obtener el complemento a 1's de N, para despues sumarle 1 y obtener N2' :

$$N1' \rightarrow \begin{array}{r} 01000101011 \\ +\qquad\quad 1 \\ \hline \end{array}$$

$$N2' \rightarrow \overline{01000101100}$$

que obviamente es el mismo resultado que se obtuvo anteriormente.

AI . 9 . REPRESENTACION BINARIA DE NUMEROS SIGNADOS.

En las máquinas binarias, los números binarios se representan por medio de un conjunto de dispositivos de almacenamiento binario (por lo general multivibradores biestables). Cada dispositivo representa un bit. Ya que muchas computadoras y calculadoras digitales manejan numeros negativos y positivos, se necesita algun medio de representacion para el signo del numero (+ o -). Esto se lleva a cabo en general agregando otro bit al numero denominado bit del signo. En terminos generales, la convencion comun que se ha adoptado es que un 0 en el bit del signo representa un numero positivo y un 1 representa un numero negativo. Esto se ilustra a continuación.

```
  A6   A5   A4   A3   A2   A1   A0
  0    1    0    0    1    0    0    = +52
  ↓    └──────────────────────────┘
 Bit          magnitud
de signo

  A6   A5   A4   A3   A2   A1   A0
  1    0    1    1    1    1    1    = -31
  ↓    └──────────────────────────┘
 Bit          magnitud
de signo
```

De la anterior convención podemos deducir la tabla A.3.

Decimal signado	Binario magnitud signada	Decimal signado	Binario magnitud signada
+0	0,0000	-0	1,0000
+1	0,0001	-1	1,0001
+2	0,0010	-2	1,0010
+3	0,0011	-3	1,0011
+4	0,0100	-4	1,0100
+5	0,0101	-5	1,0101
+6	0,0110	-6	1,0110
+7	0,0111	-7	1,0111
+8	0,1000	-8	1,1000
+9	0,1001	-9	1,1001
+10	0,1010	-10	1,1010
+11	0,1011	-11	1,1011
+12	0,1100	-12	1,1100
+13	0,1101	-13	1,1101
+14	0,1110	-14	1,1110
+15	0,1111	-15	1,1111

TABLA A.3 NUMEROS SIGNADOS.

La coma indica "," tan sólo la separación de la magnitud y el signo.

De esta manera, un numero binario N de n enteros y m bits fraccionarios, es representado en la notacion de magnitud y signo por n + m + 1 bits como :

$$N_{2's} = b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i + \sum_{i=-m}^{-1} b_i 2^i$$

bit de signo — bits de la parte entera — bits de la parte fraccionaria

donde el bit bn es 0 para números positivos, y 1 para numeros negativos.

AI.9.1. FORMA DE MAGNITUD VERDADERA

Los numeros de la figura anterior contienen un bit de signo y seis bits de magnitud. Los bits de magnitud son el equivalente binario verdadero de los valores binarios que se representan. A esta se lo denomina forma de magnitud verdadera para representar numeros binarios con signo.

Aunque este sistema de magnitud verdadera es directo y facil de entender, no es de tanta utilidad como los otros dos sistemas para representar numeros binarios con signo. Estos sistemas utilizan la misma forma de magnitud verdadera para numeros positivos, pero utilizan una forma diferente para los negativos.

AI.9.2. FORMA DE COMPLEMENTO A 1

Cuando se representan numeros negativos en forma de complemento a 1, el bit del signo se convierte en 1 y la magnitud se transforma de forma binaria verdadera en su forma de complemento a 1. Para ilustrar lo antes dicho, el numero -57 se representaria de la manera siguiente :

bit de signo

-57 = 1 111001 (forma de magnitud real)
= 1 000110 (forma de complemento a 1)

Notese que el bit del signo no se complementa, sino que se conserva como 1 a fin de indicar un numero negativo.

A1.9.3. FORMA DE COMPLEMENTO A 2.

La forma de complemento a 2 de un numero binario se forma simplemente tomando el complemento a 1 del numero y sumando 1 en la posicion del bit menos significativo. El procedimiento se ilustra a continuacion para convertir 111001 (57 decimal) en forma de complemento a 2:

```
      1 1 1 0 0 1
      ↓ ↓ ↓ ↓ ↓ ↓
      0 0 0 1 1 0   complemento a 1's
    +           1
    -------------
      0 0 0 1 1 1   complemento a 2's
```

Asi pues, -57 se escribiria como 000111 en su representación de complemento a 2. Una vez mas, el MSB es el bit del signo. Los otros seis bits son la forma de complemento 2 de la magnitud. Como otro ejemplo considerar, el complemento a 2 de -14 se escribe como 10010.

Las tres formas de representacion de numeros con signo se resumen en la siguiente figura para +57 y -57:

	Sistema de magnitud verdadera		Sistema complemento a 1		Sistema complemento a 2	
+57	0	111001	0	111001	0	111001
-57	1	111001	1	000110	1	000111
	(bits de los signos)					

Recuerdese que en todos los sistemas de magnitud verdadera, complemento a 1 y complemento a 2, los numeros positivos siempre estan en forma binaria verdadera y tienen un bit de signo de 0. Las diferencias radican de la representacion de los numeros negativos.

La forma de complemento a 2 se utiliza debido a que, como se observara, su uso nos permite efectuar la operacion de substraccion utilizando en realidad solamente la operacion de adicion. Esto es significativo ya que se refiere a que una maquina digital puede

emplear los mismos circuitos para sumar y restar, con lo cual se obtiene un ahorro de sistema físico.

AI.9.4. CONVERSION DE NUMEROS COMPLEMENTARIOS A BINARIO.

Es una tarea relativamente simple tomar un numero que este en su forma de complemento a 1 o a 2 y de convertirlo en su valor binario verdadero. Para pasar de complemento 1 a binario verdadero simplente se necesita volver a complementar cada bit. En forma analoga, para pasar de complemento 2 a binario verdadero se requiere complementar cada bit y luego sumar 1 al LSB. En ambos casos la reconversion a binario es el mismo proceso que el que se empleo para producir el complemento con el cual se empezo.

Ejemplo : representar cada uno de los siguientes numeros decimales con signo como un numero binario con signo en el sistema de complemento a 2. Utilice un total de 5 bits incluyendo el del signo : a) +13 b) -9.

a) Ya que el numero es positivo, la magnitud (13) se representara en su forma de magnitud verdadera, es decir, $13_{10} = 1101_2$. se agrega el bit del signo de 0 y se tiene :

$$+13_{10} = 01101_2$$

bit del signo

b) Puesto que el numero es negativo, la magnitud (9) tiene que ser representada en forma de complemento a 2:

$$9_{10} = 1001_2$$

0110	complemento a 1's
+ 1	
0111	complemento a 2's

Cuando se agrega el bit del signo de 1, el numero complemento con signo se convierte en :

$$-9_{10} = 10111_2$$

Ejemplo : Cada uno de los numeros siguientes es un numero binario con signo en el sistema de complemento a 2. Determinar el valor decimal en cada caso : a) 01100 b) 11010.

a) El bit del signo es 0 de modo que el numero es positivo y los otro cuatro bits representan la magnitud verdadera del numero. Es decir, $1100_2 = 12_{10}$. De esta manera, el numero decimal es +12.

b) El bit del signo es 1 de modo que el numero es negativo y los otros cuatro bits representan el complemento a 2 de la magnitud. Para averiguar cual es esta magnitud, debemos volver a complementarlo a 2. Recordemos que la operación de complemento a 2 cambiara la polaridad de un numero con signo, así que si complentamos en 2 un numero negativo binario (incluyendo el bit de signo), obtendremos en numero positivo.

```
11010          numero negativo original

00101          complemento a 1
+   1
-----
00110  (+6)    complemento a 2
```

Ya que el resultado de la operación de complemento a 2 es +6, el numero original debe haber sido -6. Es decir,

$11010_2 = -6_{10}$

AI.9.5. CASO ESPECIAL EN LA REPRESENTACION DE COMPLEMENTO A 2.

Siempre que un numero con signo tiene un 1 en el bit del signo y todos los bits de magnitud son ceros, entonces su decimal equivalente es -2^N, donde N es el numero de bits que hay en la magnitud. Por ejemplo:

$1000_2 = -2^3 = -8$

$10000_2 = -2^4 = -16$

$100000_2 = -2^5 = -32$

y así sucesivamente.

De este modo, podemos decir que el intervalo completo de valores que se pueden representar en el sistema de complemento a 2 que tiene N bit de magnitudes :

$$-2^n \text{ a } +(2^n-1),$$

AI .9. OPERACIONES CON NUMEROS BINARIOS SIGNADOS.

Como mencionamos anteriormente, la substracción puede ser efectuada como una suma, esto se debe a que por ejemplo:

$$A - B = A + (-B)$$

Sin embargo, al sumar un número positivo a un número negativo, debemos de chear sus magnitudes absolutas y encontrar el signo correcto. De la misma manera, cuando se suman números signados del mismo signo, puede suceder, que la suma de los mismos exceda la capacidad de bits del circuito sumador.

A este hecho se le conoce como overflow. Al tener un overflow, nos podría producir un resultado incorrecto, excepto si se tuviera un circuito para monitorear cuando se tengan bits de exceso u overflow.

Es por esta razón que se empleó la ARITMETICA COMPLEMENTARIA.

Al realizar el anterior procedimiento de suma, puede presentarse la situación de que sea excedida la capacidad de bits, es decir, que se presente un bit extra o acarreo extra a la izquierda del MSB, en este caso, se suma este acarreo al LSB de la suma, para producir el resultado correcto.

Ejemplo : Dado un sumador con complemento a 1's y capacidad para 5 bits encontrar :

$$(-5)_{10} + (-3)_{10} = 1,1011 + 1,1101$$

```
27₁₀        = 0,11011   minuendo
15₁₀        = 0,01111   substraendo
-15₁₀       = 1,10000   complemento a 1's del
                        substraendo


    0 , 1 1 0 1 1
  + 1 , 1 0 0 0 0
  ---------------
C1) 0 , 0 1 0 1 1   → suma

bit de acarreo


    0 , 0 1 0 1 1
  *           1
  ---------------
    0 , 0 1 1 0 0   → resultado
```

por lo que $0,01100_2 = 12_{10}$.

De acuerdo a lo anterior, podemos decir que dada la diferencia $A_2 - B_2$, donde A y B son numeros binarios, la substraccion puede obtenerse como la suma de :

$$A + B_2' + \text{posible bit de overflow}$$

Como habiamos mencionado con anterioridad, la diferencia puede ser obtenida utilizando el complemento a 2's. El procedimiento es el siguiente :

Dados 2 numeros A y B en binario :

1. Sume $A + B_2'$.

2. Si se presenta un bit de acarreo, este se cancela y el resultado es $A + B_2'$.

3. Si no hubo bit de acarreo es producido, el resultado es $A + B_2'$ B_2, es decir, que al resultado de la suma $A + B_2'$ se le obtiene el complemento a 2's y se le cambia

de signo.

Ejemplo : Si se tiene un sumador con capacidad de 6 bits, obtenga, por medio de complemento a 2's :

$$22_{10} - 11_{10} = 0,10110_2 - 0,01011_2$$

Cuando vimos el complemento a 2's, se observó que es más sencillo obtener el complemento a 1's, para despues sumarle 1 al LSB.

```
                1 , 1 0 1 0 0      complemento a 1's
              +             1
              -----------------
sustraendo      1 , 1 0 1 0 1      complemento a 2's
minuendo      + 0 , 1 0 1 1 0
              -----------------
              1 0 , 0 1 0 1 1
              ↓
        bit de acarreo
          ignorado
```

por lo tanto el resultado es : $0,01011_2 = 11_{10}$.

Ejemplo : obtenga $11_{10} - 118_{10}$.

$$11_{10} - 118_{10} = 0,0000111 - 0,1110110$$

```
                1 , 0 0 0 1 0 0 1      complemento a 1's
              +                 1
              ---------------------
                1 , 0 0 0 1 0 1 0      complemento a 2's

minuendo        1 , 0 0 0 1 0 1 0      complemento a 2's
sustraendo    + 0 , 0 0 0 0 1 1 1
              ---------------------
                1 , 0 0 1 0 0 0 1
                ↓
          no hay bit de acarreo
```

```
  0 , 1 1 0 1 1 1 0  complemento a 1's del resultado
+                 1
-------------------
  0 , 1 1 0 1 1 1 1  complemento a 2's

  1 , 1 1 0 1 1 1 1  cambio de signo
```

por lo tanto el resultado es 1,1101111 = -107₁₀.

AII . C O D I G O S

AII .1 . BITS, BYTES, PALABRAS Y NIBBLES.

En una computadora, la unidad más elemental de información es el dígito binario (Bit). Un bits es un estado lógico que solo puede ser 0 o 1. Sin embargo, un solo bit puede brindar poca información. Por esta razón, la unidad primaria de información en una computadora es un grupo de bits que recibe el nombre de palabra en la computadora. El tamaño de la palabra es tan importante que a menudo se usa para describir una computadora.

Los bytes : A un grupo de 8 bits se le conoce como byte y representa una unidad universalmente utilizada en la industria de la computación. Por ejemplo, una microcomputadora con un tamaño de palabra de 8 bits se dice que tiene un tamaño de palabra de un byte.

Las microcomputadoras de 4 bits tiene un tamaño de palabra de medio byte. Esto se conoce comunmente como "nibble". Por consiguiente, cada palabra de una microcomputadora de 4 bits es un "nibble" y dos de estos constituyen un byte.

En resumen a partir de la definición del bit, tenemos definidos los siguientes conceptos :

```
1 Byte    = 8 bits
1 Nibble  = 4 bits
2 Nibbles = 1 byte = 8 bits
```

El concepto de palabra es el de una agrupacion mas grande de bits estando esta en funcion del sistema en consideracion.

La siguiente figura muestra un ejemplo de una palabra de instruccion de una sola direccion de una computadora hipotetica de 20 bits.

Los 20 bits de la palabra de instruccion se dividen en dos partes. La primera parte de la palabra (bits 16-19) contiene el codigo de operacion (codigo op, para abreviar). El codigo op de 4 bits representa la operacion que la computadora se le instruye efectuar, como la adicion o substraccion. La segunda parte (bits 0-15) es la direccion del operando que representa la localidad en memoria donde se almacena el operando.

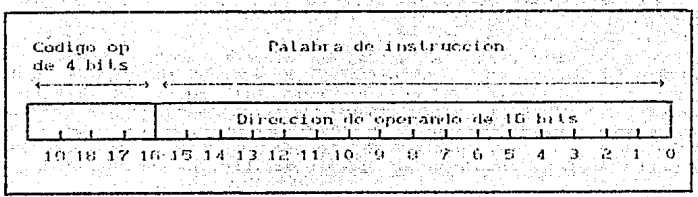


FIG A.2 PALABRA DE INSTRUCCION.

AII . 2 . CODIGOS BINARIOS. BCD. EXCESO 3. ALFANUMERICOS Y GRAY.

Los codigos binarios son todos aquellos codigos que nos permiten representar caracteres mediante el sistema binario, existen varios codigos de este tipo, a continuacion se definen algunos de ellos.

AII . 2 . 1. CODIGO BCD.

Si cada digito de un numero decimal se representa con su equivalente binario, esto produce un codigo llamado decimal

codificado en binario (en lo sucesivo se abreviará BCD). Ya que un dígito decimal puede ser tan grande como 9, se necesitan 4 bits para codificar cada dígito.

En la siguiente tabla se representa la conversión de Decimal a BCD.

DECIMAL	B C D
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

TABLA A.4 EQUIVALENCIA DECIMAL-BCD.

La razon principal del empleo de este código, es que ahorra espacio al representar un número decimal, razon por la cual es empleado en calculadoras. El empleo de este código permite trabajar en forma binaria con un mínimo de espacio.

Supongamos que se quiere convertir un número decimal de 3 dígitos a BCD, el paso de cada dígito decimal a BCD será como sigue :

800	400	200	100	80	40	20	10	8	4	2	1
Centenas BCD				Decenas BCD				Unidades BCD			
Centenas Decimal				Decenas Decimal				Unidades Decimal			

TABLA A.5 CONVERSION DECIMAL-BCD.

Para ilustrar el código BCD tomemos un número decimal como 7324.2006, cada dígito se cambia por su equivalente binario de la

manera siguiente :

```
    7    3    2    4  .  2    6    9
    ↓    ↓    ↓    ↓     ↓    ↓    ↓
 0111 0011 0010 0100 . 0010 0110 1001
```

Comparación de BCD y binario: Es importante entender que el BCD no es otro sistema numerico como el binario, el octal, el decimal y el hexadecimal. Es, de hecho, el sistema decimal con cada digito codificado en su equivalente binario. Tambien es importante comprender que el numero BCD no es el mismo que el numero binario directo. Un codigo binario directo toma un numero decimal completo y lo representa en binario; el codigo BCD convierte cada digito decimal en binario de manera individual.

La ventaja principal del codigo BCD es la relativa facilidad de conversion en y de decimal. Solo necesitan recordarse los grupos de codigo de 4 bits para los digitos decimales del 0 al 9. Esta facilidad de conversion es especialmente importante desde un punto de vista de hardware, ya que en un sistema digital son los circuitos lógicos los que efectuan las conversiones en y de decimal.

AII . 2 . 2 . CODIGO DE EXCESO 3.

El codigo de exceso 3 se relaciona con el BCD y algunas veces se utiliza en lugar de éste debido a que posee ventajas en ciertas operaciones aritmeticas. El codigo de exceso 3 para un numero decimal se efectua de la misma forma que el BCD excepto que se suma el numero 3 a cada dígito decimal antes de codificarlo en binario.

DECIMAL	B C D	EXCESO 3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

TABLA A.6 CODIGO EXCESO 3.

La tabla A.6 contiene las representaciones en código BCD y de exceso 3 para los dígitos decimales.

AII.2.3. CODIGO ALFANUMERICOS.

Este tipo de códigos son empleados cuando los sistemas de procesamiento de información utilizan, tanto caracteres numéricos, como alfabéticos y caracteres especiales.

A fin de ser de mucha utilidad, una computadora debe poder manejar información que no sea numérica. En otras palabras, una computadora debe poder reconocer códigos que representen números, letras y caracteres especiales. Estos códigos se clasifican como códigos alfanumericos.

Estos códigos tiene una longitud que va de 6 a 8 bits de longitud. Existen 2 códigos típicos e importantes, los cuales son el ASCII y el código EBCDIC. ASCII, son las siglas de American Standar Code for Information Interchange, es decir, código americano estandard para información e intercambio. Así mismo, ABCDIC son las siglas de Extended BCD Interchange Code, es decir, código de intercambio BCD extendido. Estos códigos se ilustran en la tabla A.7.

P es un bit llamado bit de paridad que mas adelante hablaremos de el. Este bit de paridad sera 0 si la suma decimal de los 1's restantes es par, entonces se dice que es bit de paridad par.

Ejemplo : Codifique la instrucción WRITE 23 en ASCII utilizando segmentos de 16 bits, considerandose P como un bit de paridad par.

	W	R	
Segmento 1 :	(1101 0111	1101 0010	)ASCII
	I	T	
Segmento 2 :	(1100 1001	1101 0100	)ASCII
	E	BLANCO	
Segmento 3 :	(1100 0101	1010 0000	)ASCII
	2	3	
Segmento 4 :	(1011 0010	0011 0011	)ASCII

Note que esta instrucción puede ser alamacenada en 4 registros

de memoria de 16 bits cada uno como :

REGISTRO 1 : 1101 0111 1101 0010
REGISTRO 2 : 1100 1001 1101 0100
REGISTRO 3 : 1100 0101 1010 0000
REGISTRO 4 : 1011 0010 0011 0011

CARACTER	CODIGO EBCDIC	CODIGO ASCII
BLANCO	0100 0000	P010 0000
.	0100 1011	P010 1110
(	0100 1101	P010 1000
+	0100 1110	P010 1011
$	0101 1011	P010 0100
*	0101 1100	P010 1010
)	0101 1101	P010 1001
-	0110 0000	P010 1101
/	0110 0001	P010 1111
,	0110 1011	P010 1100
'	0111 1101	P010 0111
=	0111 0001	P010 1101
0	1111 0000	P011 0000
1	1111 0001	P011 0001
2	1111 0010	P011 0010
3	1111 0011	P011 0011
4	1111 0100	P011 0100
5	1111 0101	P011 0101
6	1111 0110	P011 0110
7	1111 0111	P011 0111
8	1111 1000	P011 1000
9	0100 1011	P011 1001
A	1100 0001	P100 0001
B	1100 0010	P100 0010
C	1100 0011	P100 0011
D	1100 0100	P100 0100
E	1100 0101	P100 0101
F	1100 0110	P100 0110
G	1100 0111	P100 0111
H	1100 1000	P100 1000
I	1100 1001	P100 1001
J	1101 0001	P100 1010
K	1101 0010	P100 1011
L	1101 0011	P100 1100
M	1101 0100	P100 1101
N	1101 0101	P100 1110
O	1101 0110	P100 1111
P	1101 0111	P101 0000

TABLA A.7 CODIGO ASCII.

CARACTER	CODIGO EBCDIC	CODIGO ASCII
Q	1101 1000	P101 0001
R	1101 1001	P101 0010
S	1110 0001	P101 0011
T	1110 0010	P101 0100
U	1110 0011	P101 0101
V	1110 0101	P101 0110
W	1110 0110	P101 0111
X	1110 0111	P101 1000
Y	1110 1000	P101 1001
Z	1110 1001	P101 1010

TABLA A.7 CODIGO ASCII. (CONT.)

AII . 2 . 4 CODIGO GRAY.

El codigo Gray pertenece a una clase de códigos llamada codigos de cambio mínimo, en los cuales solo un bit del grupo de codigo cambia cuando pasa de una etapa a la siguiente. El codigo Gray es un codigo sin valor, es decir, las posiciones de los bits en los grupos de codigo no tienen ningun valor especifico asignado a ellos. Debido a esto el codigo no se ajusta a las operaciones aritmeticas sino que haya aplicacion en dispositivos de entrada y salida y algunos tipos de convertidores de analogico a digital.

La tabla A.8 muestra la representacion de codigo Gray de los numeros decimales del 0 al 15, junto con el codigo binario directo. Si examinamos los grupos en codigo de Gray para observar cada numero decimal, se puede observar que al pasar de numero decimal cualquiera al siguiente, solo un bit del codigo de Gray cambia.

El codigo Gray se usa en situaciones en las cuales otros codigos, como el binario, podrian producir resultados erroneos o ambiguos durante esas transiciones en las cuales mas de un bit del codigo cambia. Por ejemplo, al aplicar el codigo binario y pasar de 0111 a 1000 se supone que los 4 bits cambien simultaneamente, segun el dispositivo o circuito que genere los bits, puede haber una diferencia significativa en los tiempos de transicion de los diferentes bits. Si es asi, la transicion de 0111 a 1000 podria producir varios estados intermedios. Por ejemplo, si el bit mas significativo cambia mas rapido que el resto, ocurriran las

siguientes transiciones :

```
0111   →  decimal 7
↓
1111   →  codigo erroneo
↓↓↓↓
1000   →  decimal 8
```

La aparicion de 1111 solo es momentanea pero pudiera producir concebiblemente una operacion erronea de los elementos que son controlados por los bits. Con claridad, el uso del codigo Gray eliminaria este problema, ya que solo ocurre un cambio de bit por transicion y no puede haber "gerarquias" entre los bits.

DECIMAL	CODIGO BINARIO	CODIGO GRAY
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

TABLA A.8 CODIGO GRAY.

AII . 3 . PARIDAD Y DETECCION DE ERRORES.

El bit de paridad : Un bit de paridad es un bit extra que se agrega a un grupo de codigo que se transfiere de una localidad a otra. El bit de paridad es un 0 o un 1 segun el numero de 1's que haya en el grupo de codigo. Para esto se emplean 2 metodos

diferentes.

En el método de paridad par, el valor del bit de paridad se escoge de manera que el numero total de 1's que hay en el grupo de codigo (incluso el bit de paridad) sea un numero par. Por ejemplo, supongase que el grupo de codigo es 1000011. Este es el caracter C segun ASCII. El grupo de codigo tiene tres 1's, por tanto, sumaremos un bit de paridad de 1 para hacer que el numero total de 1's sea un numero par. El nuevo grupo de código, incluyendo al bit de paridad, se convierte de este modo en :

```
        11000011
           ↓
    Bit de paridad
    agregado
```

Si el grupo de codigo contiene un numero par de 1's para empezar, al bit de paridad se le asigna un valor de 0.

El método de paridad impar se utiliza en la misma forma en que se escoge el bit de paridad para que el numero total de 1's (incluyendo al bit de paridad) sea un numero impar. Por ejemplo, para el código de grupo 1000001, el bit de paridad asignado seria un 1. Para el grupo de codigo 100011, el bit de paridad seria un 0.

BIBLIOGRAFIA

. **Diseño Digital**
M. Morris Mano
Prentice-Hall
México. 1989.

. **Lógica Digital y Diseño de Computadoras**
M. Morris Mano
Prentice-Hall
México, 1988.

. **Arquitectura de Computadoras**
M. Morris Mano
Prentice-Hall
México. 1988.

. **Teoría de la Información y Codificación**
Norman Abramson
Paraninfo
México, 1985.

. **Matemáticas para Computación**
Symour Lipschutz
Mcgraw-Hill
Mexico. 1987.

. **Sistemas Digitales, Principios y Aplicaciones**
Ronald J. Tocci
Prentice Hall
México. 1987.

. Fundamentos de Computadoras Digitales
Earlee Thomas C.
McGraw-Hill
Mexico, 1984.

. Principio de Procesamiento de Datos
Davis Gordon B.
Trillas
Mexico, 1979.

. Diseño con Circuitos Integrados
Texas Instruments Inc.
CECSA
Mexico, 1980.

. Fundamentos de Diseño Digital
De la Cruz Laso C.
Trillas
Mexico, 1986.