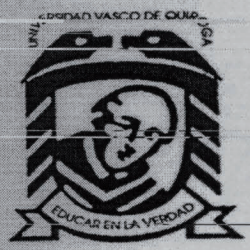
El diseño lógico en los sistemas de información

Autor: Cintia Lillian Ledesma Balvanera

Tesina presentada para obtener el título de:
Lic. En Sistemas Computarizados [sic]

Nombre del asesor:
Sergio Francisco Barraza Ibarra



UNIVERSIDAD VASCO DE QUIROGA

LICENCIATURA EN SISTEMAS COMPUTARIZADOS

"EL Diseño Lógico en los Sistemas de Información "

TESINA

Para obtener el Tìtulo de:
LICENCIADA EN SISTEMAS COMPUTARIZADOS

Presenta:
CINTIA LILLIAN LEDESMA BALVANERA

Asesor:
ING. Y M.A. SERGIO FRANCISCO BARRAZA IBARRA

CLAVE 16PSUOO14Q
ACUERDO 952006

Morelia, Mich. Mayo de 1999

Dedico este esfuerzo primeramente a Dios, que a estado conmigo en todo momento dandome su infinito amor y miscericordia.

En segundo lugar a mis Padres, los cuales me han dado la vida y me han apoyado para ser de mi una persona de bien.

En tercer lugar a mis Maestros, que gracias a sus enseñanzas nos han preparado para salir adelante, sobre todo en el ambito profesional.

Por último, dedico este esfuerzo a todos mis Amigos y Personas que de alguna manera colaboraron para ser esto posible.

" Gracias."

INDICE

1.0.- ***INTRODUCCION***

En este Trabajo de Investigación se mostrará la definición y los componentes básicos del diseño lógico. Se iniciará con las definición del álgebra booleana, los principales teoremas del algebra booleana. Mostraremos como definirlas en terminos de tablas de verdad, como obtener expresiones booleanas correspondientes.

Se comprenderá que una de las principales tareas del diseño lógico, es encontrar la expresión entre las posibles formas no normalizadas que tengan el menor número de operadores.

Veremos las puertas lógicas más sencillas que se usan para implementar operadores booleanos, así como las puertas más complejas que pueden implementar varios operadores booleanos a la vez. Las puertas lógicas sencillas y complejas constituyen las bibliotecas lógicas normalizadas usadas por los diseñadores del diseño lógico.

Se mostrará que lo más importante en el diseño lógico, es encontrar la implementación de una función booleana que mejor satisfaga los requisitos del coste y retarda que normalmente manifiesta con altas prestaciones. (retardo mínimo y coste mínimo). Se introduce la biblioteca lógica que es básicamente una implementación del álgebra boooleana y las técnicas del diseño lógico.

El presente trabajo esta dividido en siete capítulos, que son utilizados para describir el proceso de diseño lógico. El segundo capítulo contiene los antecedentes, donde explico la historia del diseño. En el tercer capítulo enumero el objetivo general y los objetivos específicos que pretendo obtener con éste trabajo de investigación.

En el cuarto capítulo abordo el tema del álgebra booleana, donde menciono su definición, sus princioales postulados y teoremas que proporcionan el sustento teórico-lógico para el diseño de expresiones.

El quinto capítulo es el diseño lógico, en donde explico la descripción de un biestable y la técnica del diseño lógico. En el sexto capítulo hago mención de los componentes de memorización, como el registro, las pilas de inserción / extracción, y las colas FIFO. Por ultimo en el séptimo capítulo me refiero a las conclusiones y recomendaciones, a las que conlleva este trabajo de investigación.

2.0.- ***ANTECEDENTES***

Con anterioridad las primeras computadoras electrónicas fueron sumamente complejas, y usaban el sistema numérico decimal, el cual requiere 10 distintos niveles en cada una de las posiciones de los números. El problema de definir y mantener esos 10 niveles provocó que el sistema decimal fuera sustituido por el sistema binario, el cual utiliza sólo dos niveles lógicos.(0,1)

En la aritmética binaria una cantidad puede existir o no existir, y éste tipo de decisiones pueden Implementar en forma relativamente sencilla usando circuitos de transistores donde un voltaje existe o no en la salida. Esta simplicidad, la economía del transistor y los circuitos integrados hacen de las computadoras digitales unas máquinas mucho más atractivas.

Los datos que alimentan a la computadora digital lo hacen en forma de pulsos eléctricos en dos niveles discretos de voltaje, y la información es representada por el número de pulsos y el tiempo que hay entre ellos. En la mayoría de las computadoras la información decimal se convierte a forma binaria para ser procesada en las diferentes operaciones convirtiendose posteriormente de binaria a decimal para presentar los resultados.

En el pasado todas estas tareas de diseño tenian que hacerse manualmente, ya que sólo una parte del proceso de diseño estaba automatizado. Sin embargo esto comenzó a cambiar, con la disponibilidad de herramientas CAD para el diseño físico, que se desarrollaron para ayudar a los diseñadores en la ubicación e interconexionado de los componentes.

Posteriormente se desarrollaron herramientas adicionales para la captura de esquemas a nivel de puerta y también para simulación a nivel de puerta, que hicieron posible automatizar más el proceso de diseño.

En los noventas, ha avanzado significativamente la automatización del proceso de diseño, introduciendo lenguajes de descripción Hardware estándares y herramientas de síntesis. Con los lenguajes de descripción Hardware se pueden generar ahora descrpciones de comportamiento que se simulan fácilmente y con las nuevas herramientas de síntesis se ha hecho mucho más fácil pasar de una representación estructural a una de comportamiento.

3.0.- ***OBJETIVO GENERAL***

Es reafirmar que el éxito de la tecnología de computadoras se basa principalmente en la sencillez de los circuitos lógicos que se diseñan y en la facilidad de su fabricación. Los circuitos digitales se componen de elementos básicos de procesamiento, denominados puertas y elementos básicos de memorización, conocidos como biestables. La sencillez del diseño de los circuitos digitales se debe al hecho de que las señales de entrada y salida de cada puerta o biestable pueden tomar sólo dos valores, 0 y1, y a que el cambio de los valores de las señales está regido por el álgebra booleana. Con los circuitos lógicos se hacen una serie de desiciones necesarias para obtener una respuesta lógica a problemas que tienen un conjunto de condiciones.

3.1.-*Objetivos específicos*

1.-*Las Bases del Diseño Lógico.-*

El diseño lógico se basa en el álgebra booleana, y esta a su vez se basa en los estatutos lógicos que puedan designarse como verdadero o falso, no permitiendo la asignación de valores intermedios. Las operaciones lógicas básicas del álgebra booleana son el AND, OR , XOR y sus bloques lógicos que son el NAND, NOR y XNOR. Este tipo de álgebra se encuentra regida por sus postulados y teoremas.

2.- *Aplicación del Diseño Lógico.-*

Este tipo de diseño se aplica en la creación de los circuitos digitales que procesan señales que pueden tomar sólo un pequeño número de valores fijos, normalmente dos, que hacen que sean más fuertes y faciles de diseñar, por ello los cálculos y procesamiento de la información se hacen con circuitos digitales.

3.-*Su utilidad en el proceso de actualización de las computadoras.*

Gracias al diseño lógico se han podido crear dispositivos a escala de integración muy grande (Tecnología VLSI), ejemplo de esto son las memorias grandes, microprocesadores, chips microcomputadores y subfunciones de sistemas como controladores gráficos y aceleradores.

El pequeño tamaño y bajo costo de los chip VLSI, han revolucionado la tecnología del diseño, ya que proporciona a los diseñadores, una oportunidad de crear circuitos integrados de uso específico a bajo costo y con altas prestaciones que pueden realizar una sola tárea de forma muy eficiente y adaptarlo para ciertas aplicaciones.

4.-*El Diseño Lógico en el procesamiento de datos.*

Por medio del diseño lógico las computadoras se han vuelto mucho más rápidas, y con esto ha influido en el proceso de datos de la siguiente manera:

-Capacidad

1.- Mayor velocidad de procesamiento:

Uso de la computadora para efectuar cálculos, ordenar, recuperar datos e información y efectuar repetidamente la misma tarea, con una mucho mayor velocidad que los seres humanos.

2.- Incremento en el volumen:

Proporcionar la capacidad necesaria para procesar una cantidad mayor de actividades, para aprovechar nuevas oportunidades de tipo comercial.

3.- Recuperación más rápida:

Localización y recuperación de información del sitio donde se encuentra almacenada. Llevar a cabo búsquedas complejas.

-Control

1.- mayor exactitud y mejora en la consistencia:

Se tiene un mejor control en la información que entra y sale. Se salvaguardan datos importantes y sensibles en una forma que sea accesible sólo al personal autorizado.

-Comunicación

1.-Mejoras en la comunicación:

Acelera el flujo de información y mensajes entre localidades, así como dentro de oficinas, incluyendo la trasmisión de documentos dentro de las oficinas.

4.0- ***ALGEBRA BOOLEANA***

4.1.- *Definición*

El álgebra booleana es el marco matemático en el que está basado el diseño lógico y se usa en la descripción, síntesis y análisis de las funciones lógicas binarias. EL álgebra booleana se basa en el concepto de que los estatutos lógicos que puedan designarse como verdadero o falso. No permitiéndose la asignación de valores intermedios.

También puede definirse como un conjunto de operaciones y una serie de postulados que se suponen ciertos sin necesidad de demostrarlos. Y un conjunto de elementos se define como cualquier colección de objetos que tengan una propiedad en común. Los postulados, son las suposiciones básicas a partir de las que se pueden deducir por medio de teoremas todas las demás propiedades del sistema.

El propósito del álgebra booleana es facilitar al análisis y diseño de los circuitos digitales. proporciona una herramienta para:

1.-Expresar en forma algebraica una relación de tablas de verdad entre las variables.

2.-Expresar en forma algebraica la relación entrada-salida de diagramas lógicos.

3.-Encontrar circuitos más simples para la misma función.

4.2.-*Operaciones Lógicas Básicas*

En las operaciones lógicas básicas que vamos a ver enseguida, hay que tomar en cuenta que el 0 y 1, son etiquetas que representan los valores falso o cierto, y que no hay que confundirlos con los valores númericos cero y uno.

Las operaciones lógicas básicas son:

Operación AND

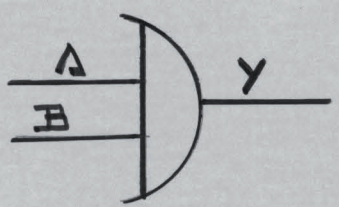
Una función AND es cierta(1) si y sólo si todas las variables involucradas son ciertas(1).
Ejemplo: Y= A . B

Aqui "Y" va ha ser cierta si y sólo sí, A y B son cierta

-Representación de su Tabla de Verdad:

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

-Compuerta AND



Operación OR

Una función OR es cierta(1), si al menos una de las variables involucradas es Cierta(1) o ambas.

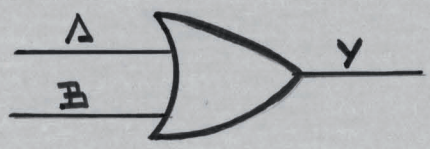
Ejemplo: y= A+B

Aqui "Y" va ha ser cierta si A o B son ciertas(1), o ambas.

-Representación de su tabla de verdad:

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

-Compuerta OR



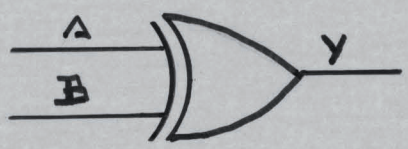
Operación XOR

Una función XOR va a ser cierta(1), si y sólo si una de las variables sea cierta, pero no ambas.
Ejemplo: Y= A + B

Aqui "Y" es cierta(1), si y sólo si A o B son ciertas(1), pero no ambas.
-Representación de su Tabla de Verdad

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

-Compuerta XOR



Operación NOT

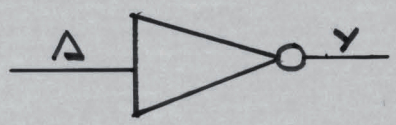
Es un negador o negación. Es una operación de inversión, convierte el estado o valor de una función o variable en su complemento. Al realizar la operación NOT sobre una función o variable cuyo valor es cierto(1), el resultado sera falso(0) y viceversa.

Ejemplo: Y= A'

Aqui "Y" es igual a la inversa de A.
-Representación de su Tabla de Verdad

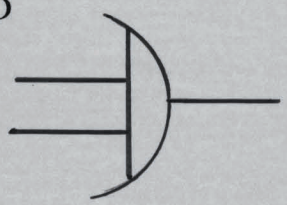
A	Y
0	1
1	0

-Compuerta NOT

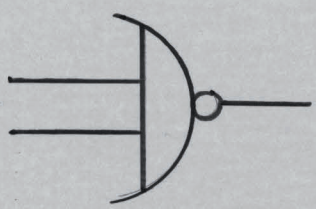


En las funciones anteriores si utilizamos el negador(NOT), las compuertas quedarían de la siguiente manera:

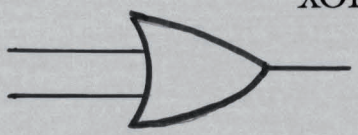
AND



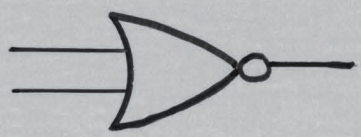
NAND



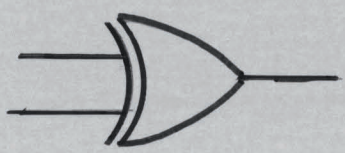
OR



XOR



XOR



XNOR

4.3.- *Postulados Del Algebra Booleana*

1.-Identidad-

Para OR $A+0 = A$
Para AND $A \cdot 1 = A$

2.-Conmutativa-

Para OR $A+B=B+A$
Para AND $A \cdot B=B \cdot A$

3.-Distributiva

Para OR $A+(B*C) =(A+B) \cdot (A+C)$
Para AND $A \cdot (B+C)=(A \cdot B) + (A \cdot C)$

4.-Idempotencia

Para OR $A + A= A$
Para AND $A \cdot A = A$

5.-Complementaria

Para OR $A + A' = 0$
Para AND $A \cdot A' = 1$

6.- Asociativa

Para OR $A+B+C =A +(B+C)= (A+B) +C$
Para AND $A \cdot B \cdot C = A \cdot (B \cdot C) = (A \cdot B) \cdot C$

7.-Dualidad

Esta propiedad establece que una expresión dual surge de otra a la que se le intercambian 0 por 1, y 1 por 0, junto con los operadores (+ y .) .

Ejemplo:

$a + 0 = a$ su dual será $a \cdot 1 = a$

4.4.- *Teoremas*

Teorema No. 1 (idempotencia)

Para OR X+X=X
Para AND X . X=X

Teorema No. 2

Para OR X+1= 1
Para AND X . 0= 0

Teorema No.4 (Involución)

(X')=X

Teorema No. 6 (Morgan)

Para OR A+B+C ... = A . B . C ...
Para AND A . B . C ... = A+B+C ...

Teorema No.3 (Absorción)

Para OR A+ A . B = A
Para AND A . (A+B)=A

5.0.- ***DISEÑO LOGICO***

5.1.-*Biestables o flip-flop*

El dispositivo básico de cálculo de una computadora digital está formado por compuertas y unos dispositivos llamados biestables y estos proporcionan la memoria.

El circuito básico para almecenar información en un circuito digital es un biestable. Existen varios tipos fundamentales y muchos diseños de circuitos, sin embargo hay dos características que son compartidas por todos los Biestables:

Figura 5.1

1.-*EL biestable, es un circuito que tiene sólo dos estados estables, que designaremos como estado 1 y estado 0. Este circuito puede recordar, almacenar, un bit de información binaria, debido a esta característica. El biestable responde a entradas, si una entrada lo lleva a tomar un estado 1, permanecerá en él, recordando el 1, hasta cuando alguna señal lo lleve a tomar un estado 0. De igual manera, una vez colocado en estado 0, permanecerá en el hasta cuando se le diga que vaya al estado 1. Esta habilidad del biestable de mantenerse en un estado, es la base del almacenamiento de información en la sección de operación o cálculo de un computador digital.

2.-*El Biestable tiene dos señales de salida, de las cuales una es el complemento de la otra.

a).-A cada Biestables se le da un nombre. Los nombres son generalemente letras como X, Y, A, o B, o combinaciones de letras y números como A1 o B2, o algunas veces a causa de la dificultad de colocar subíndices de las máquinas de escribir o de las impresoras, simplemente A1, o B2. En el diagrama el biestable se denomina X, y tiene dos salidas la X y X'. Las líneas de salida son siempre complementarias, esto significa que si la salida X tiene una señal 1, la línea de salida X' tendrá una señal 0 y si la línea de salida X tiene una señal 0, la línea de salida X' tendra una señal 1.

b).-El estado de biestable será el estado que tome la salida X. O sea si la línea de salida X tiene una señal 1, decimos que el biestable X está en el estado 1, de igual forma, si la línea X tiene una señal 0 decimos que el biestable está en el estado 0.
Hay dos líneas de entrada en el biestable, estas se usan para controlar el estado del biestable.

Las reglas son las siguientes:

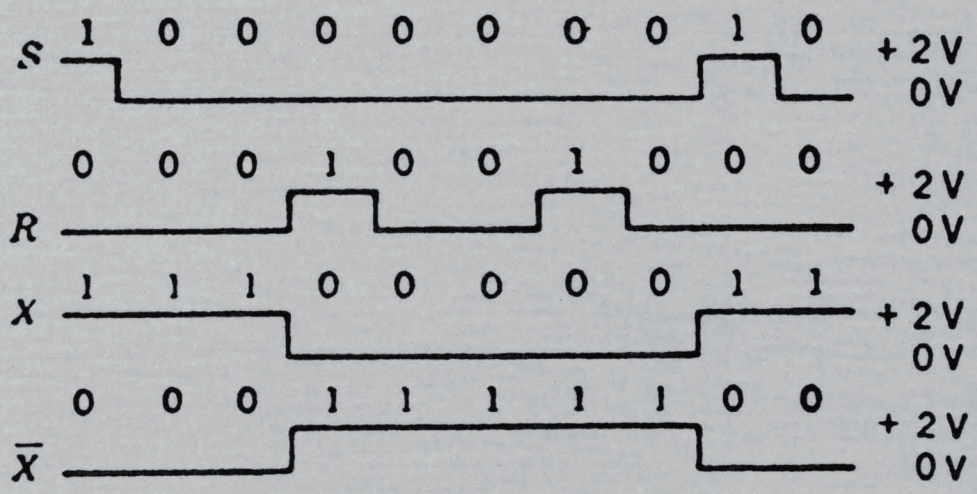
-Mientras las dos líneas de entrada S y R sean 0, el biestable permanece en el mismo estado, es decir no cambia de estado.

-Un 1 en la línea S y un 0 en la línea R, hacen que el biestable se coloque en el estado 1.

-Un 1 en la línea R y un 0 en la línea S, hacen que el biestable se coloque en el estado 0.
-Colocar un 1 en la línea S y un 1 en la línea R al mismo tiempo, está prohibido. Porque si esto ocurre el biestable no puede ir a ningún estado. Ejemplo de una secuencia posible de señales de entrada y el estado resultante de un biestable.

S	R	X	
			X es el estado del biestable después que las entradas S y R son aplicadas
1	0	1	
0	0	1	El biestable permanece en el mismo estado.
0	0	1	
0	1	0	El biestable va al estado 0.
0	0	0	
0	0	0	
0	1	0	Se le indica al biestable que vaya a 0, pero ya está.
0	0	0	
1	0	1	El biestable va al estado 1.
0	0	1	

figura 5.1.



Formas de onda de un flip-flop *SR*.

5.2- *Transferencia*

Aunque el biestable es simple en operación, El biestable básico es adecuado para todos lo casos. Analicemos la operación de este biestable en una configuración llamada circuito de transferancia. La función de esta configuración es transferir el estado, o contenido de X1-Y1 ,de X2-Y2, y de X3-Y3, bajo el comando de transferencia que corresponde a un 1 en la línea de transferencia.

En la figura 5.2, supongamos que Y1,Y2 y Y3 han sido colocados en un estado que deseamos recordar o almacenar en X1, X2 y X3, mientras que los biestables "Y" se usan para otros cálculos. Colocando un 1 en la línea de transferencia se realiza la transferencia de información deseada. Comprender esta transferencia de Y1 a X1 depende de que Y1 esté en estado 0, osea la línea de salida Y1 tiene un 0, y por lo tanto la línea de entrada conectada a la compuerta AND es 0; esta compuerta colocará un 0 en la línea de entrada S de X1 mientras que la salida Y1' tiene un 1, que hace que teniendo un 1 en la línea de transferencia, se tenga un 1 en la linea de entrada R de X1, mientras que la linea de transferencia es 0, ambas entradas del biestable X son 0 y el biestable permanecerá en el último estado que haya alcanzado.

Los conjuntos relacionados de biestables son llamados registros en las computadoras, o sea podemos llamar a los tres biestables Y1,Y2, y Y3, el registro "Y", y a los tres biestables X1, X2 y X3 el registro "X". Entonces un 1 en la línea de transferencia, trasmite el contenido del registro Y al registro X.

Figura 5.2

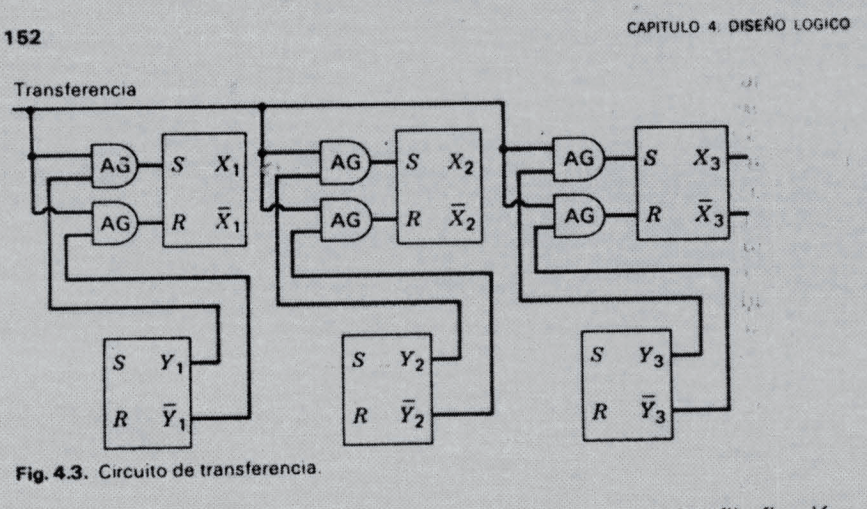


Fig. 4.3. Circuito de transferencia.

5.3.- *Registros De Desplazamiento*

Un registro de desplazamiento, es el resultado de utilizar un selector que permite desplazar el dato memorizado. Este tipo de registro desplaza su contenido en un bit, en una dirección específica, cuando la señal a desplazar(Shift) es 1.

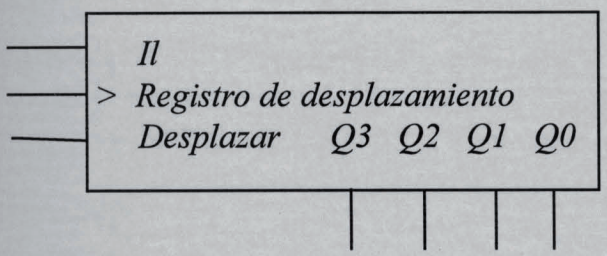
Ejemplo:

En la figura 5.3, un registro de desplazamiento de 4 bits. La entrada IL se utiliza para producir el nuevo dato de 1 bit en el biestable más a la izquierda del registro. El esquema del registro se podría utilizar para convertir una secuencia de datos serie, en otra de datos paralelo. Por ello se le denomina en ocaciones registro de desplazamiento de Entrada-Serie/ Salida-Paralelo. Para aumentar la versatilidad de un registro podríamos también usar un selector 4 a 1 que nos permite combinar las funciones de desplazamiento y de carga.

El registro podría desplazar su contenido o carga en paralelo un nuevo dato. Además, podría desplazar en un bit su contenido a la izquierda o a la derecha, introduciendo por la derecha o por la izquierda el bit de dato disponible en las entradas serie.

El registro de desplazamiento con carga en paralelo puede utilizarse para convertir una secuencia de datos serie a paralelo o una secuencia de datos paralelo a serie, con la opción de sacar el primero el Bit más significativo(MSB), o el menos significativo (LSB). Estos registros suelen utilizarse en la conversión de datos del computador para comunicarse en serie, y para captar datos trasmitidos en serie a fin de tratarlos en un procesador.

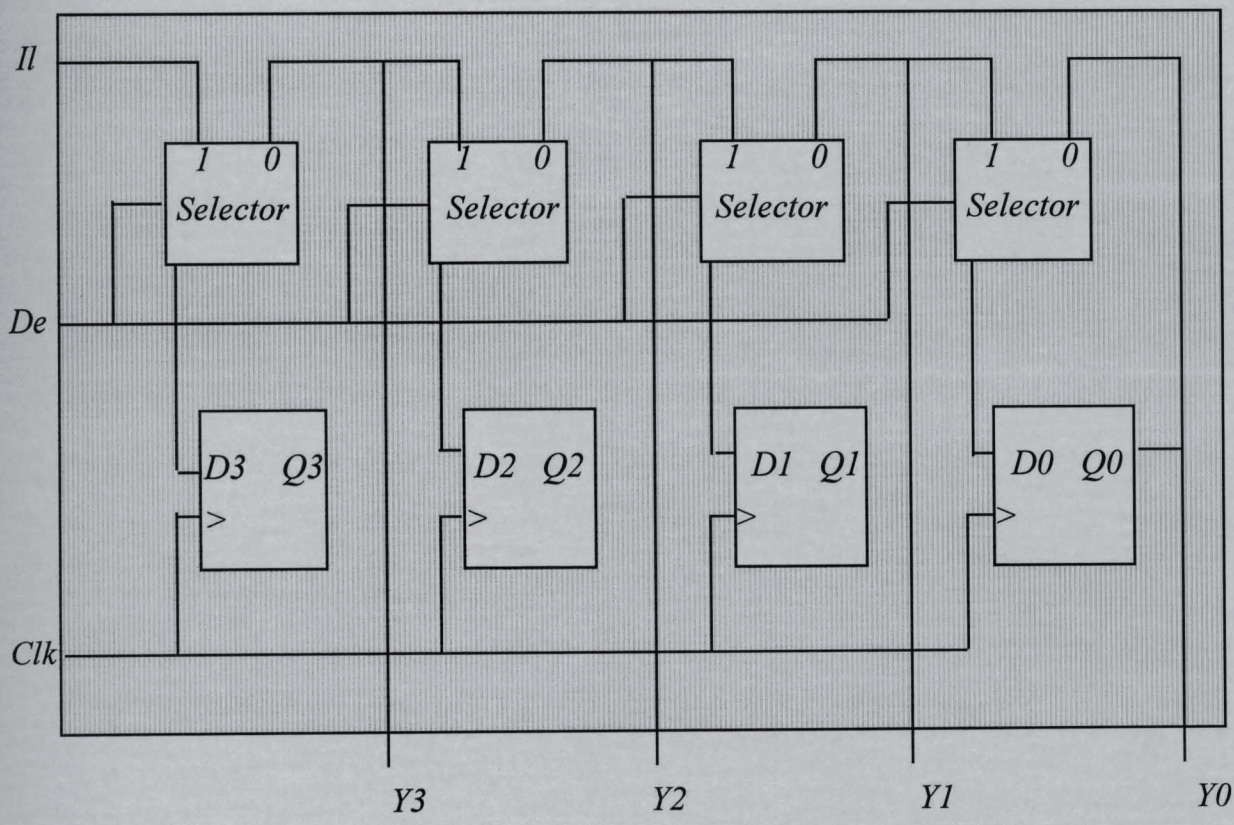
Figura 5.3.
"Diagrama de registro de desplazamiento"



a) Símbolo gráfico

Estado Actual	*Estado Siguiente*
Desplazar	*Q3 Q2 Q1 Q0*
0	*No cambia*
1	*Il Q3 Q2 Q1*

b) Tabla de operación



c) Esquema de registro

5.4.- *Contador Binario*

El contador binario es uno de los circuitos lógicos más importantes y existen varios tipos. El objetivo principal de un contador binario es registrar el número de eventos de alguna entrada.

En la figura 5.5. un contador binario registra el número de ocurrencias de borde de transición positiva (o pulsos) en la entrada. Es deseable que el contador comience con 0 en los tres biestables, para lo cual se ha adicionado una línea más a cada flip-flop, la línea puesta a CERO DC (RESET). Esta línea permanece normalmente en el nivel cero, cuando se hace positiva ó 1, coloca los biestables en cero. Cuando la línea de puesta a CERO DC va al estado 1, el biestable va a cero ignorando cualquiera de las otras entradas y la presencia o no del reloj. Es importante la entrada puesta a cero porque esta anula las otras entradas cuando está en 1, forzando al biestable a tomar el estado 0. Sin embargo un 0 en esta línea no afecta en forma alguna la operación del biestable.

Antes de iniciar el conteo, se pone en 1 la línea de puesta a cero del contador, y se llevan los tres biestables a 0, despues la línea de puesta a 0 del contador se coloca nuevamente en 0. Cuando se presenta el primer borde positivo del reloj, el biestable X1, va al estado 1. Esto se debe a que cuando X1 está en su estado cero, la salida X1' está en 1, colocando en al entrada S un 1. y la salida X1' está en cero colocando un cero en la entrada R, o sea se lleva un 1 al biestable X1.

El contador tiene ahora X3=0, X2=0, y X1=1 ó 001 en binario. El contador binario ha avanzado de 000 a 001. La aparición del segundo borde positivo del reloj hace que el biestable X1 vaya del estado 1 al estado 0. El razonamiento es el siguiente:

Cuando X1 está en 1 la salida X1' es 1, y esta conectada a la entrada R, y la salida X1 es 0 y esta conectada a la entrada S. Esto le indica al biestable que vaya a 0, y cuando aparece el segundo pulso de reloj, se produce la transición. Cuando un biestable está acoplado en forma cruzada, esto es cuando su salida sin complementación está conectada a la entrada R y su salida complementada a la entrada S, la aparición de un pulso de reloj siempre hará que éste se complemente o cambie de valor.

El cambio de un valor de 1 a 0 del biestable X1, hace que X2 cambie de 0 a 1. Esto se debe a que la salida X1' esta conectada a la entrada CL de X2 y ha cambiado de 0 a 1, que es un cambio positivo y como X2 está acoplado en forma cruzada se complementará (cambiará de valor), pasando de 0 a 1. Esto no afecta a X3 porque la entrda CL de X3 ha pasado de 1 a 0 que es un cambio negativo.

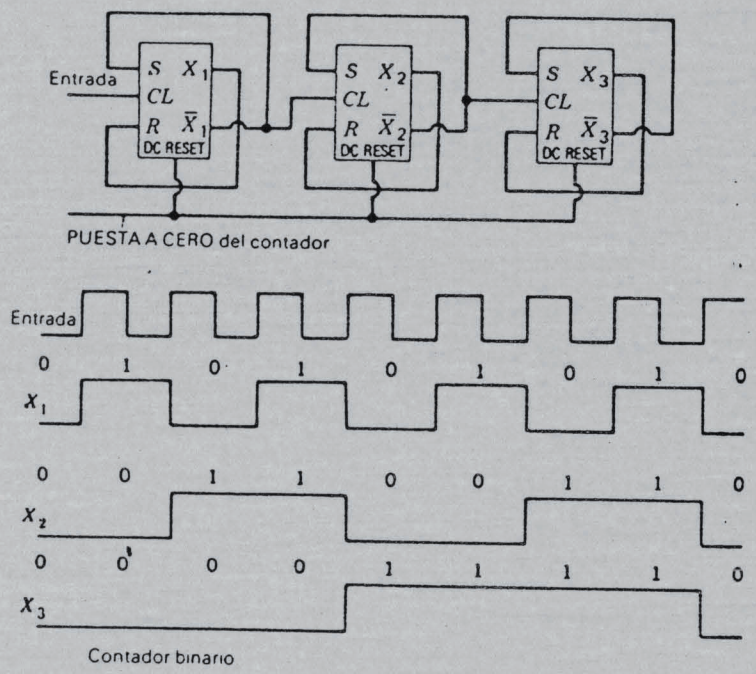
El contador ha avanzado a X1= 0, X2 =1, X3=0, es decir la secuencia de estado ha sido 000, 0001, 010. Un razonamiento de este tipo demostrará que la progresión de los estados del contador es como sigue:

X3	X2	X1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0
0	0	1
0	1	0

Esta es una lista de números binarios del 0 al 7 que se repite continuemente. Despues del quinto pulso de entrada el contador tendrá 101 ó 5 binario, despues del séptimo pulso tendrá 111, o 7 en binario. El número máximo que el contador puede manejar sin ambigüedad es 7. despues del octavo pulso, el contador tendrá 0, despues del noveno pulso 1. Este se llama contador módulo 8 o contador de tres etapas.

El contador puede extenderse con otro tipo de biestable X4 con acoplamiento cruzado y su entrada CL conectada a la salida del biestable X3. Este formaría un contador de 4 etapas o módulo 16, que puede manejar hasta 15 conteos, con un quinto biestable puede contar hasta 31, con un sexto biestable hasta 63.

figura 5.4


Entrada
S X1
CL
R X̄1
DC RESET
S X2
CL
R X̄2
DC RESET
S X3
CL
R X̄3
DC RESET
PUESTA A CERO del contador
Entrada
0 1 0 1 0 1 0 1 0
X1
0 0 1 1 0 0 1 1 0
X2
0 0 0 0 1 1 1 1 0
X3
Contador binario

5.5.- *Contador BCD*

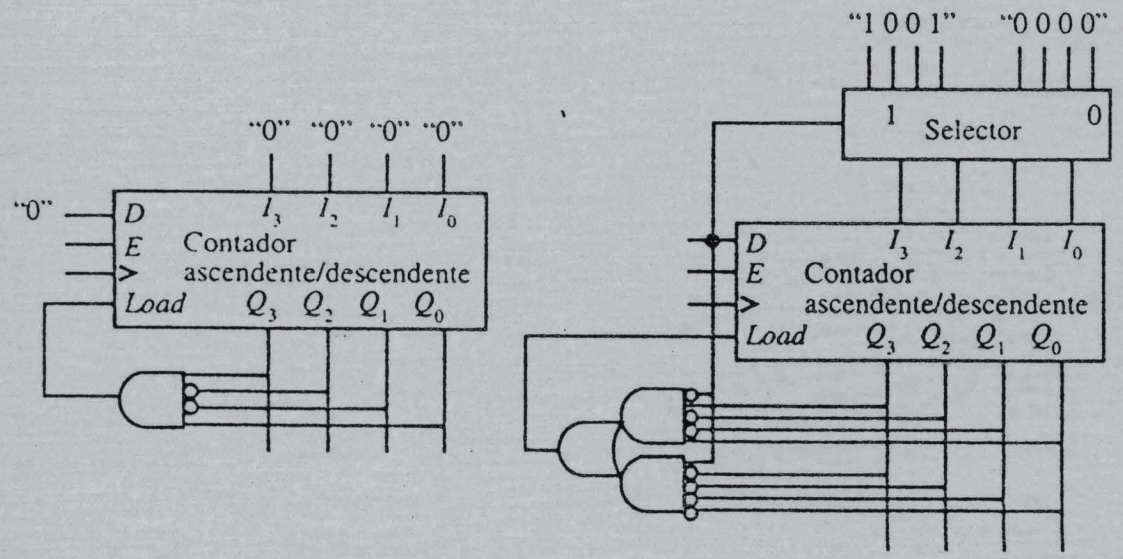
Los contadores BCD son los que cuentan siguiendo la secuencia 0,1,2,3,4,5,6,7,8,9,0..-

En la figura 5.5. (a), podemos construir un contador BCD detectando cuándo éste alcanza la cuenta de 9 y cargando 0 en lugar de 10 en el siguiente ciclo de reloj. La detección se realiza mediante una puerta AND cuya salida sea 1, si el contenido del contador es 1001. la salida de la puerta AND se conecta a la entrada de carga load del contador, lo que permite que el contador cargue 0 al llegar el siguiente flanco ascendente de la señal de reloj.

También en la figura 5.5.(b) de forma similar podemos construir contadores BCD Ascendente/ Descendentes. En la dirección de cuenta ascendente debemos cargar 0 en el contador cuando alcanza el valor 9, mientras que en la dirección descendente debemos cargar 9, cuando el contador alcanza la cuenta 0. Necesitamos un selector que decida entre cargar 0 o 9 cuando el contador debe reinicializarse, así como una puerta AND-OR que actúe sobre la entrada load cuando el contador alcance una cuenta 0 o de 9.

De manera similar, podemos construir contadores que comiencen en cualquier punto y cuenten en casi cualquier secuencia. Observese no obstante que cada salto en la secuencia de salida requiere una puerta adicional para la detección y una entrada adicional en el selector para la carga.

figura 5.5



(a) Contador BCD ascendente

(b) Contador BCD ascendente/descendente

Contadores BCD.

5.6.- *Técnica del Diseño Lógico*

Hasta ahora ya que se ha visto los principios básicos del álgebra de boole y se ha mostrado las diferentes funciones booleanas con diferentes puertas lógicas. En lo siguiente veremos la técnica de los mapas de karnaugh, esta técnica nos ayuda a optimizar el diseño lógico. A continuación ejemplificaremos las tablas de verdad para las siguientes expresiones algebraicas:

Expresión algebraica

a).- S = A + B

Tabla de Verdad

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

El resultado de la tabla de verdad de ésta expresión es una OR, porque con que sólo una entrada sea cierta o sea 1, la salida(S) es cierta.

Expresión algebraica

b).- S = A' + AB

Tabla de verdad

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

Expresión algebraica

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

Tabla de verdad

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

El resultado de la tabla de verdad de esta expresión es una OR, que equivale a S = A + B

Expresión algebraica

d).- A . (B + C)

Tabla de verdad

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

Nota:

El acomodo de las variables en la tablas de verdad se basa en en el sistema binario, y va de

-Solución de las siguientes expresiones utilizando los Postulados y Teoremas del Algebra Booleana.

Ejemplo 1.

S = xyz + xyz'	
xyz + xyz' = xy(z + z')	ditributiva
xy(z + z')= xy . (1)	complementaria
xy . (1) = xy	teorema 2

Ejemplo 2.

S = A'B + AB + AB'	
A'B + AB + AB' = A'B + A(B+ B')	distributiva
A'B + A(B + B') = A'B + (A . 1)	complementaria
A'B + (A . 1) = A'B + A	identidad
A'B + A = A+ B	teorema 3 (absorción)

Ejemplo 3.-

S = A+ AB' + AB + AB'C + C

A+ AB' + AB + AB'C + C = A + A(B' + B + B'C) + C	distributiva
A + A(B' + B + B'C) + C = A+ A(1 + B'C) + C	complementaria
A + A(1+ B'C) + C = A + A(1) + C	2ª de identidad
A +A(1) + C = A + A + C	identidad
A + A + C = A + C	idempotencia

-*Código Gray*

Este código se utiliza para transformar combinaciones binarias normales a combinaciones donde sólo hay un cambio.

Normal		Gray	
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Este código de gray es muy importante, porque lo vamos a utilizar para la ejemplificación de los mapas de karnaugh.

-*Mapas De Karnaugh*

Este método consiste en hacer una tabla de todos los posibles estados de las variables binarias y seleccionar dentro de un estado combinatorio las salidas que se requieran, observando que las variables permanecen constantes en las salidas escogidas y qué variables son indiferentes.

Instrucciónes:

Para la optimización de los mapas, se debe agrupar por renglones, por columnas o por ambas, en multiplos de dos (2, 4, 8, 16). No se pueden agrupar los que no sean múltiplos de 2, o que sean diagonales. Los mapas de karnaugh, es un mapa hemisférico, de tal manera que se pueden agrupar en el centro de norte a sur, de sur a norte, de este a oeste, y de oeste a este. Los grupos se deben buscar siempre que reunan el mayor número de unos posibles, y dichos grupos se pueden trasladar.-

-Nota: los unos deben estar adyacentes.

Ejercicio No. 1

A).- Expresión

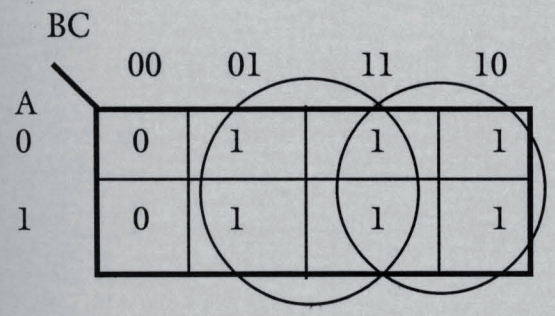
S = A'B'C + A'BC' + A'BC + AB'C + ABC' + ABC

B).- Tabla de verdad

A	B	C	S		
0	0	0	0	1	
0	0	1	1	2	
0	1	0	1	4	1
0	1	1	1	3	
1	0	0	0	1	
1	0	1	1	2	2
1	1	0	1	4	
1	1	1	1	3	

La numeración que se encuentra fuera de la tabla de verdad, es el orden en que va a colocar la columna S, dentro del mapa que va ha tener 4 columnas y dos renglones.

C).- Mapa



Como podemos observar en éste mapa, sólo seleccionamos dos grupos de unos.

Expresión resultante del mapa:

S = C + B

En la expresión resultante no participa A, porque es una sola columna.

Ejercicio No. 2

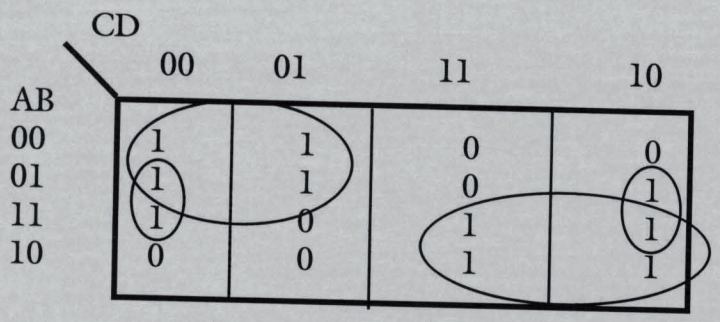
A).- En este ejercicio vamos a partir de una tabla de verdad de 4 variables que son A, B, C, y D. y con un supuesto resultado de la columna S.

B).-**Tabla de verdad**

A	B	C	D	S		
0	0	0	0	1	1	
0	0	0	1	1	2	1
0	0	1	0	0	4	
0	0	1	1	0	3	
0	1	0	0	1	1	
0	1	0	1	1	2	
0	1	1	0	1	4	2
0	1	1	1	0	3	
1	0	0	0	0	1	
1	0	0	1	0	2	4
1	0	1	0	1	4	
1	0	1	1	1	3	
1	1	0	0	1	1	
1	1	0	1	0	2	3
1	1	1	0	1	4	
1	1	1	1	1	3	

La numeración que va por fuera de la tabla es el orden en que va ir la columna S, en el mapa. El mapa va a constar de cuatro columnas, por cuatro renglones.

C).- **Mapa**



En este mapa tenemos tres grupos de cuatro unos, y de aquí sale la siguiente expresión:

$$S = A'C' + AC + BD'$$

6.0.- ***COMPONENTES DE MEMORIZACION***

6.1.- ***Registro***

El componente de memorización más sencillo es un registro, que se puede considerar como un biestable ampliado a múltiples bits. Cada registro consta de n biestables activados por una señal de reloj común. Cada biestable del registro almacena su propio dato en cada flanco ascendente de la señal de reloj. Un registro básico posee n entradas y n salidas además de la señal de reloj.

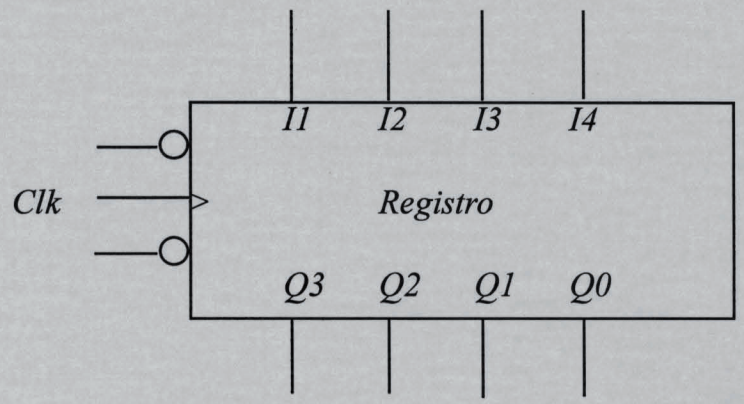
-El registro consta de 4 bits.
-El registro consta de 4 biestables D, conectados en paralelo.

La funcionalidad del registro básico puede mejorarse añadiendo diferentes señales de control. Si el registro debe ponerse a uno o a cero independientemente de la señal de reloj, cuando se conecta la alimentación o bien en respuesta a algún evento especial, podemos añadirle señales asíncronas de puesta a uno(preset) y de puesta a cero(clear). Esto se consigue reemplazando los biestables del registro básico, por otros con puesta a uno y a cero.

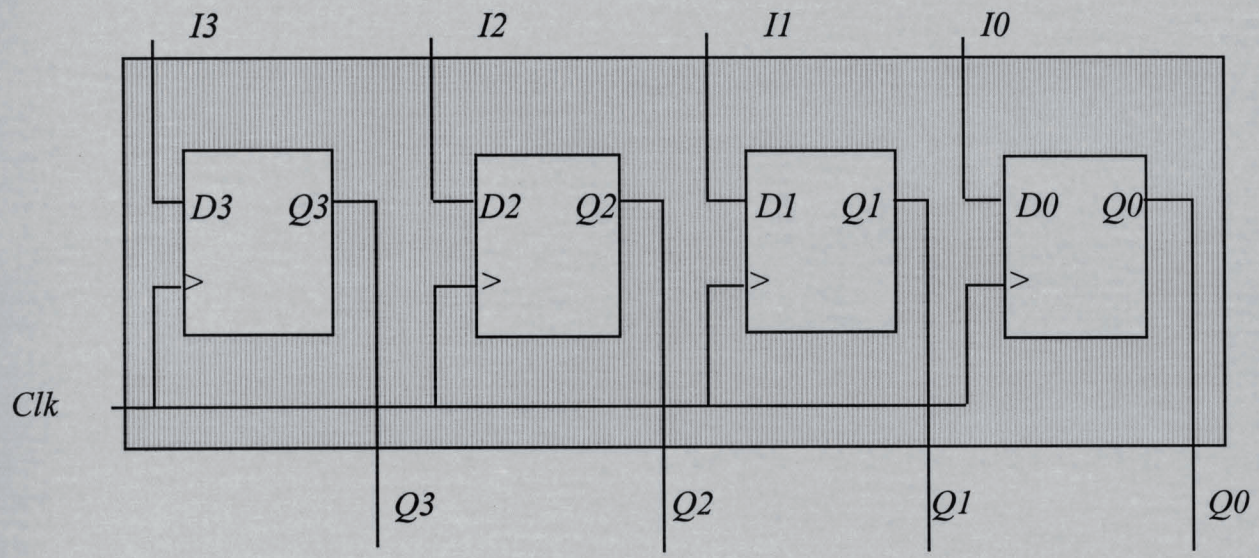
En la figura 6.1, un registro puede hacerse todo ceros poniendo a 0 la señal clear durante un corto período de tiempo. El contenido de registro puede hacerse uno, poniendo a cero la señal Preset. Estas señales de puesta a cero y puesta a uno, son independientes de la señal del reloj y tienen prioridad sobre ella. Significa que si el contenido de registro está a cero durante el flanco ascendente de la señal del reloj, se ignora la entrada y el registro es puesto a cero o a uno.

En el registro, se almacena automáticamente un nuevo dato por cada flanco ascendente de la señal de reloj. En la mayoría de los sistemas digitales un dato se memoriza durante varios ciclos de reloj antes de modificarlo. Por ello es útil poder controlar cuando ha de introducirse el dato en el registro. Esto se consigue con el uso de una señal de control normalmente denominada carga(load o habilitación(enable), que permite cargar un dato en el registro.

Figura 6.1
"Diagrama de Registro"



a).- Símbolo gráfico



b) Esquema de registro

6.2.- *Contadores*

Un contador es un tipo especial de registro que incorpora un incrementador, que permite contar de forma ascendente o descendente.

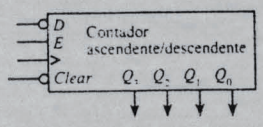
Ejemplo:

-Un contador ascendente que tiene dos señales de control:
-Una señal E de habilitación(Enable), que cuando vale 1 habilita la cuenta; y una señal clear para la puesta a cero del contador. Puede verse que el incrementador consta de una serie de semi-sumadores(HA). Mientras que la señal E sea 1, este contador realizará una cuenta ascendente, sumando 1 a su contenido por cada flanco de subida del reloj.

En la figura 6.2., un contador ascendente básico puede también ampliarse como contador ascendente /descendente, sustituyendo el semi-sumador por un semi-sumador/ Restador (Has), que pueda incrementar o decrementar bajo el control de una señal de dirección. La diferencia con un contador binario de 4 bits seria que tiene una tercera entrada de control D, y el objetivo de esta entrada adicional seria permitir contar de forma ascendente cuando su valor es 0 y de manera descendente cuando es 1.

En el esquema lógico de la figura 6.2., puede verse que cada HAS consta de una puerta XOR conectada a la entrada de los biestables, así como dos puertas AND y una OR que se utiliza para propagar el acarreo. Es importante observar que el contador ascendente/ descendente siempre comienza a contar desde cero. En muchas ocasiones es sin embargo más conveniente poner inicialmente el contador a un valor diferente para después contar de forma descendente o ascendente hasta que alcanza el valor 0. Para ello sólo necesita una puerta NOR que detecte cuando se alcanza el valor de cuenta 0. Para construir un contador con inicialización de este tipo combinaríamos un incrementador/decrementador con un registro con carga en paralelo. Este contador tendría tres señales de control E,D, y load. La señal E habilitaría la cuenta en la dirección especificada por la señal D, y la señal load cargará una nueva entrada e inhabilitará la cuenta siempre que su valor sea 1. Por otra parte, mientras la señal load sea 0, el contador se comportará exactamente igual que el ascendente / descendente.

figura 6.2



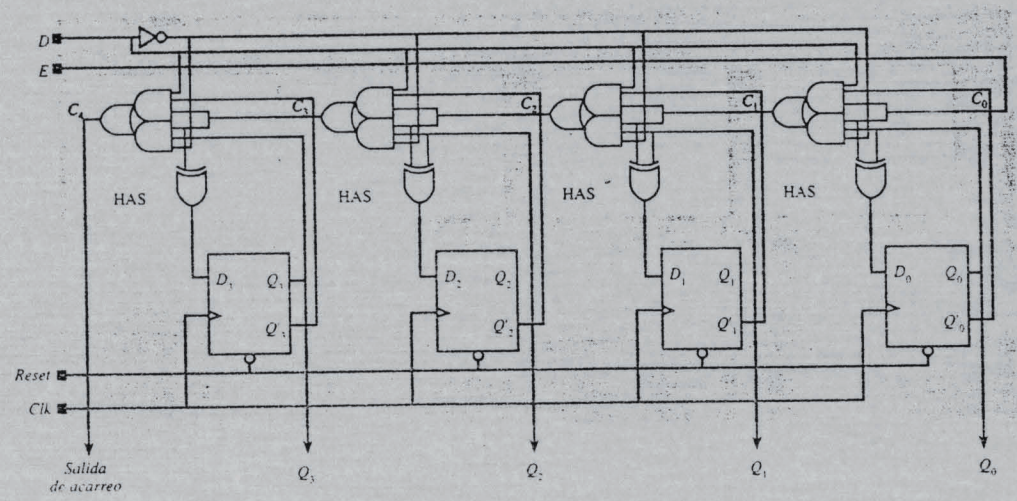
(a) Símbolo gráfico

E	D	OPERACIONES
0	X	No cambia
1	0	Cuenta ascendente
1	1	Cuenta descendente

(b) Tabla de operación

E	D	Q_i	C_i	C_{i+1}	D_i
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	1	0	0

(c) Tabla de verdad del HAS



(d) Esquema lógico

Contador binario ascendente/descendente de 4 bits.

6.3.- *Memorias De Acceso Aleatorio*

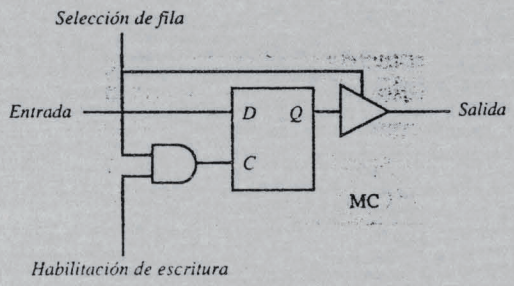
La memoria de acceso aleatorio (Random Access Memories), son grandes y lentas, pero muy apropiadas para memorización a largo plazo de programas y de los datos utilizados en los cálculos.

En la figura 6.3., la memoria RAM se clasifíca dependiendo de la forma de implementación de la celda de memoria (MC, Memory Cell), pueden ser estáticas o dínámicas. La memoria RAM estática, la celda se contruye entre cuatro y seis transistores, empleando dos inversores interconectados, e implementado con un transistor la puerta AND de entrada y con otro el adaptador de salida. Este tipo de celda de memoria retendrá su contenido indefinidamente en tanto no se reescriba y se mantenga conectada la alimentación.

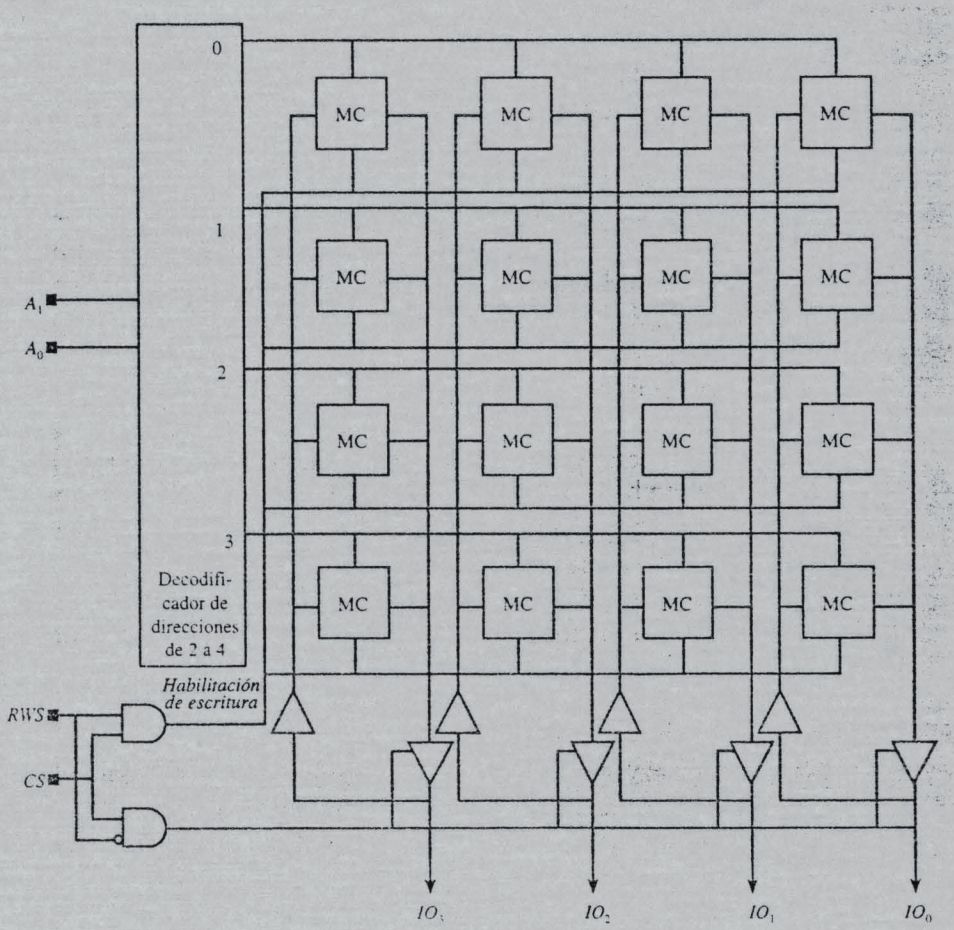
La memoria RAM dinámica, las celdas de memoria se implementan con sólo un transistor. El inconveniente es que es contenido de la celda se pierde con cada operación de lectura, tras la cual habrá que reescribirlo. Además ocurre que, debido a imperfecciones de fabricación el contenido de la celda se pierde transcurrido un tiempo. Para contrarrestar éste problema cada posición de memoria debe accederse con cierta frecuencia, o alternativamente, el contenido debe reescribirse o refrescarse periódicamente. Durante el refresco todas las demás operaciones de lectura o escritura deben suspenderse, lo cual puede resultar molesto. A persar de ello, la mayor densidad y coste reducido de la memoria RAM dinámica las hace muy populares para el diseño de productos electrónicos. La memoria RAM estática, por otro lado aunque más costosas, son más rápidas y por lo tanto apropiadas para tiradas más pequeñas y en los casos en que se requiera un acceso a memoria más rápido.

La memoria RAM dinámica, como la estática son memorias volátiles, ya que su contenido se pierde al desconectar la fuente de alimentación. Por este motivo equipos tales como teléfonos y contestadores automáticos incluyen baterías que evitan la pérdida de datos en memoria por un corte temporal del sumunistro eléctrico. La organización de la memoria RAM impone restricciones de temporización de las entradas y salidas para las operaciones de lectura y escritura.

figura 6.3



(a) Celda de memoria



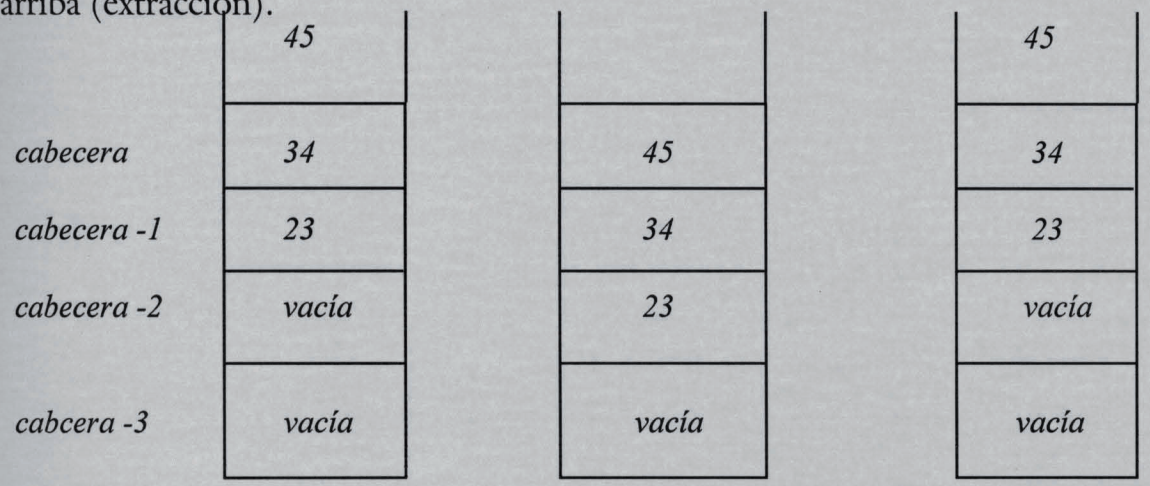
(b) Esquema de la memoria

Estructura de una RAM.

La memoria ROM(Memoria Sólo Lectura) y PROM, por el contrario son memorias no volátiles ya que preservan su contenido aunque se corte de la alimentación. No obstante el contenido de estas memorias no se puede leer hasta que se restablezca la alimentación. Esta memoria es utilizada para almacenar programas y tablas de constantes que no cambian el valor una vez que la producción del sistema microcomputador se ha completado. Los chips de memoria sólo para lectura contienen información importante que la computadora necesita para llevar a cabo funciones básicas, para correr programas interconstruidos, como el programa que utiliza para la computadora para arrancar. Este tipo de memoria es permanente y retiene la información registrada en el microcircuito aunque se apague la computadora. El tamaño de la memoria asignada a un microprocesador depende del programa y las palabras de datos necesarias para una aplicación particular.

6.4.- *Pilas de Inserción / Extracción*

Las pilas de inserción / extracción son de uso frecuente en diseños software y hardware. Por definición, una pila de inserción / extracción es un componente de memoria con un acceso limitado, en la que cualquier dato almacendo puede ser accedido en cualquier momento, a los datos memorizados, en una pila de inserción / extracción sólo se puede acceder a través de una posición concreta(la cabecera de la pila). Cuando se inserta un dato en la pila, éste se memoriza en la cabecera y todos lo demás datos se desplazan hacia el interior una posición de la pila(inserción). Por el contrario, cuando se extrae un dato de la pila, es eliminado de la cabecera y todos los demás datos se trasladan a una posición hacia arriba (extracción).



a).- Contenido de la pila antes de insertar el 45.

b).- Contenido de la pila tras insertar el 45.

c).- Contenido de la pila después de extraer el 45.

En la figura 6.4., cuando se diseña una pila de inserción / extracción, el punto más importante a tener en cuenta es que los datos almacenados se desplazan en una posición descendente o ascendente en las operaciones de inserción y extracción, respectivamente. Esta observación nos lleva a utilizar registros de desplazamiento para su implementación, así como un contador ascendente/ descendente para detectar cuándo está llena o vacía la pila. Estos componentes se emplean en el diseño de la pila de cuatro palabras de m bits. Esta pila tiene m líneas de entrada INi y m líneas de salida OUTi, también dispone de tres señales de control: inserción/extracción (push/pop), habilitación(enable), y puesta a cero (reset).

La señal push/pop controla la inserción y extracción de manera que, cuando es 0, se inserta un dato en la pila, y cuando es 1 se elimina el dato de la cabecera de la pila. La señal enable habilita el funcionamiento, y la señal reset, anula, cuando toma el valor 0, el contenido de los registros de desplazamiento y del contador. Las operaciones síncronas de la pila se indican de operaciones, observese que la pila tiene dos señales de salida, vacia y llena que indican el estado de la pila de la siguiente manera. Cuando vacía toma el valor 1 indica que la pila está vacia; por otro lado cuando llena es 1, la pila está llena.

Puede verse que la implementación de la pila incluye un registro de desplazamiento con carga en paralelo (RD conCP) por cada pareja entrada / salida(en total la pila contiene m registros de desplazamiento). Estos registros desplazarán a la derecha cuando se pida una operación de inserción y a la izquierda cuando sea necesaría una de extracción. Cada nuevo dato se inserta en la pila a través de la entrada IL del registro de desplazamiento, siendo memorizado en la salida Q3, que respresenta la cabecera de la pila. Ya que el contador contiene el número entradas a la pila, por cada flanco de subida del reloj, el registro desplazará a la derecha con la cuenta ascendente del contador(si push/pop=0 y enable=1), o desplazará a la izquierda con la cuenta descendente del contador (si push/pop=1 y enable=1).

La principal desventaja de la implementación con registros de desplazamiento es el número elevado de registros caros que se necesitan cuando se requiere una pila grande. Por esta razón las pilas de inserción/extracción grandes se implementan normalmente con RAM. Además, ya que las RAM no son capaces de desplazar su contenido, las operaciones de inserción y extracción en estas pilas deben implementarse de una manera ligeramente diferente; cambiando la posición de la cabecera de la pila. Al insertar datos en dicha pila, la dirección de la cabecera se incrementará por cada inserción y decrementará con cada extracción. Considérece una pila que está vacía, la cebecera está en la dirección cero. entonces a medida que van escribiendo datos en la cabcera de la pila, la dirección de dicha cabecera se incrementa con cada operación de inserción, de manera que la cabecera de la pila es siempre una posición vacía en la que se escribirá el nuevo dato en respuesta a la siguiente operación de inserción solicitada. Por otra parte cuando se solicite una operación de extracción, se leerá el dato de la posición justo debajo de la cabecera, cuya dirección es uno menos que la de la cabecera.

figura 6.4

PUSH/POP	ENABLE	OPERACIONES
X	0	No cambia
0	1	Inserción
1	1	Extracción

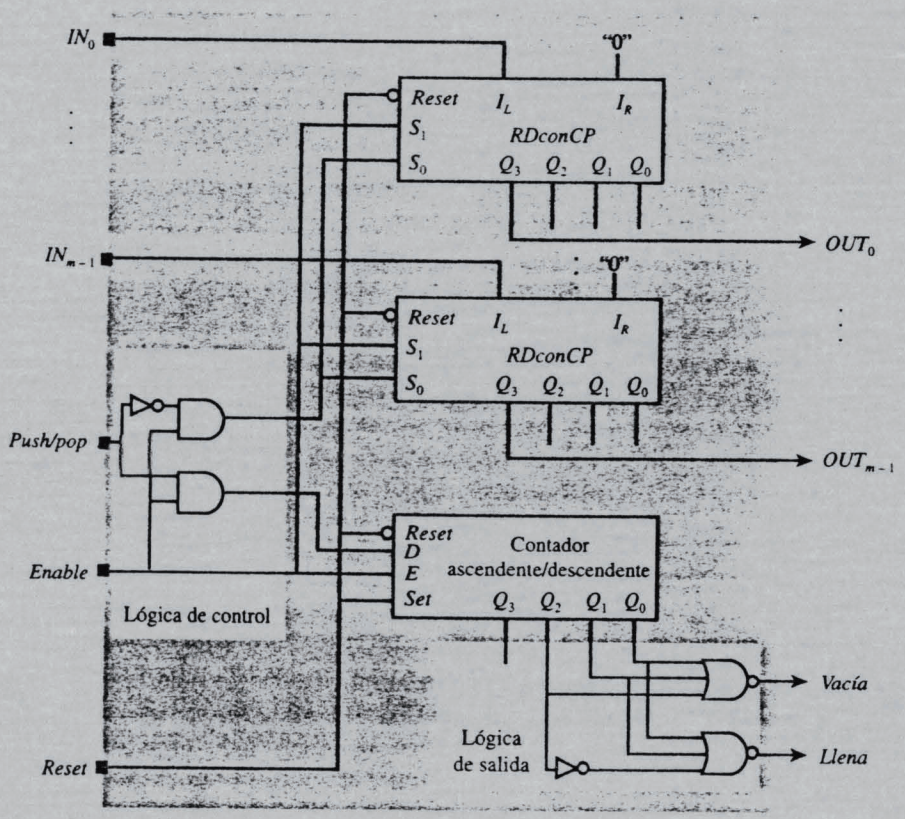
(a) Tabla de operación

		CONTROLES DE LOS REGISTROS DE DESPLAZAMIENTO		CONTROLES DEL CONTADOR	
PUSH/POP	ENABLE	S_1	S_0	D	E
X	0	0	0	X	0
0	1	1	1	0	1
1	1	1	0	1	1

(b) Tabla de control

SALIDAS DEL CONTADOR				
Q_2	Q_1	Q_0	VACÍA	LLENA
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1

(c) Tabla de salida



(d) Esquema de la pila

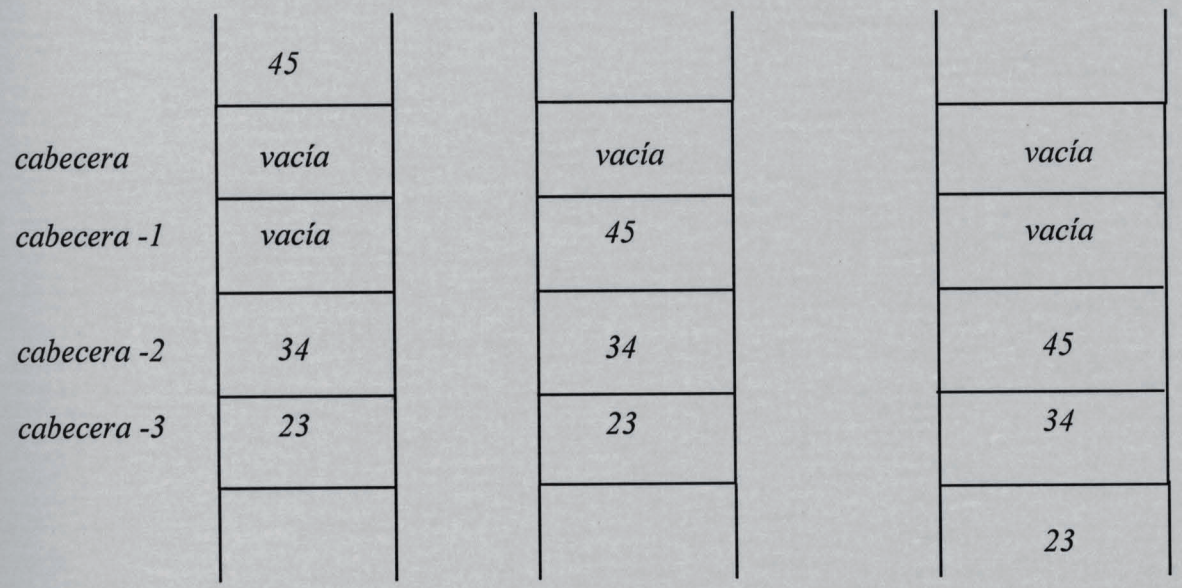
Pila de inserción/extracción de 4 palabras.

6.5.- *Colas primero en entrar, primero en salir.*

Las colas primero en entra, primero en salir(FIFO), se emplean con frecuencia para suavizar ráfagas en la peticiones de servicio. Como ejemplo la gente que hace cola para comprar entradas de cine, para entrar en un autobús, o para disfrutar de un viaje en un parque de atracciones, que debe esperar en fila hasta que le llegue el turno para recibir un servicio. Situaciones similares pueden surgir con diferentes procesadores, o cualesquiera elementos que intercambien datos entre si, en el sentido de que cuando la producción de datos exceda momentáneamente su consumo, debemos intercalar una cola FIFO entre el productor de datos y el consumidor de los mismos. En tales casos, la velocidad de producción no puede exceder indefinidamente a la del consumo, ya que se requeriría una cola infinita.

Por el contrario ambas velocidades deben ser en medida iguales. Sin embargo, ocasionalmente aparecerán ráfagas de producción y de consumo, y el tamaño de la cola determinará la longitud de ráfaga que se puede tolerar.

El objetivo de una cola FIFO es memorizar los datos en exceso, que serán eventualemente leídos de la cola en el mismo orden en el que fuerón escritos. Así pues, el primer dato memorizado es leído primero y así sucesivamente.



a).- Contenido de la cola antes de memorizar el número 45

b).- Contenido de la cola tras memorizar 45

c).- Contenido de la cola después de leer el 23

En la figura 6.5., una cola tiene m líneas de entrada INi, y m líneas de salida OUTi, dispone también de tres señales de control: lectura / escritura, habilitación (enable), y puesta a cero (reset). Cuando la señal lectura / escritura es 0, la cola dará como salida el dato que ha estado memorizado por más tiempo, tomandolo del principio de la cola. En consecuencia cuando la señal lectura / escritura sea 1, se añadira otro dato al final de la cola.

Las colas FIFO normalmente disponen de dos salidas de control que se emplean para controlar los circuitos productor y consumidor. Por ejemplo cuando la cola está llena, la señal llena tomará el valor 1, avisando al productor que puede perderse cualquier dato adicional que envíe. Cuando la cola está vacía , la señal vacía se hace 1, avisando al consumidor que no han llegado aún nuevos datos.

figura 6.5

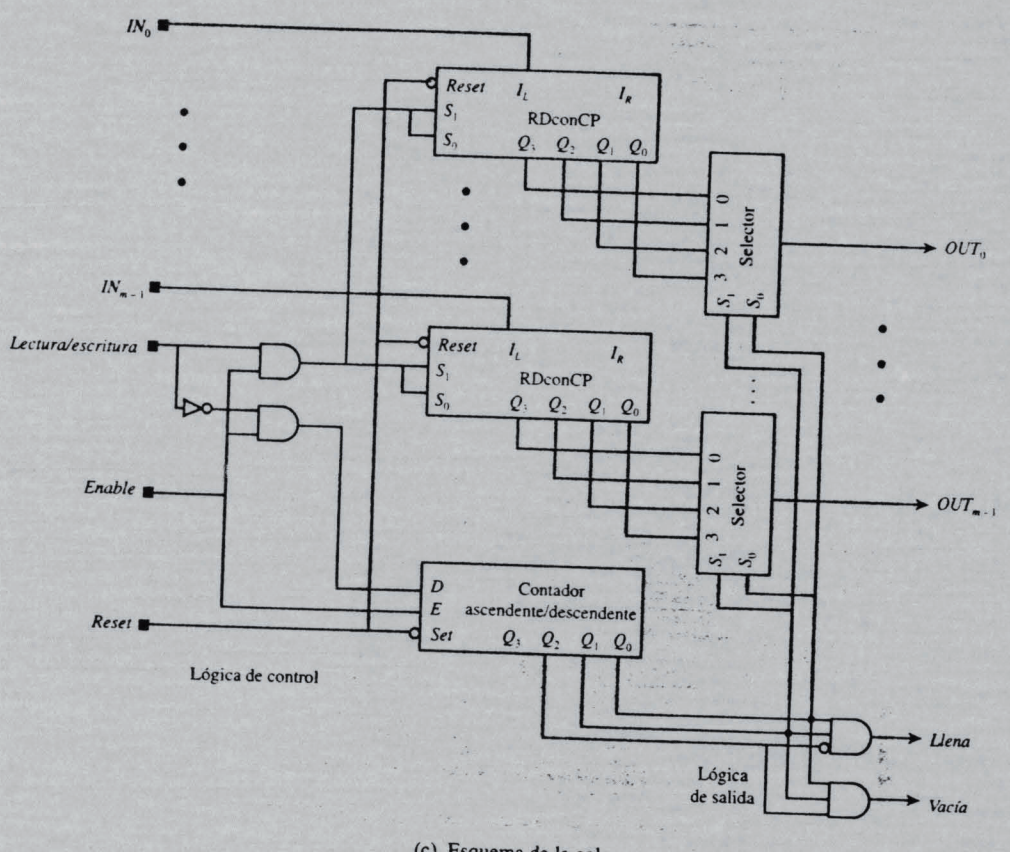
$S_0 = S_1 =$ *(Lectura/Escritura)Enable*
D = (Lectura/Escritura)'Enable
E = Enable

LECTURA/ESCRITURA	ENABLE	OPERACIONES
X	0	No cambia
0	1	*Lectura*
1	1	*Escritura*

(a) Tabla de operación

LECTURA/ESCRITURA	ENABLE	S_1	S_0	D	E
X	0	0	0	X	0
0	1	0	0	1	1
1	1	1	1	0	1

(b) Tabla de control



(c) Esquema de la cola

Cola FIFO de cuatro palabras.

7.0.- ***CONCLUSIONES Y RECOMENDACIONES***

Por lo anterior investigado podemos dejar claro, que el proceso de diseño abarca desde la manera de concebir su funcionamiento, hasta el desarrollo de un esquema de fabricación, muchas personas están implicadas desarrollando cada uno una tarea específica.

El diseño tiene cuatro respresentaciones que son las siguientes:

1.- Representación de comportamiento.
Esta representación describe el funcionamiento, pero no la implementación de un diseño dado.

2.-Una representación estructural.
Aqui se especifica la implementación del producto, sin hacer referencia a su funcionamiento.

3.-Representación física.
Aquí se especifica las características físicas, dando las dimensiones y situación en cada componente, o sea se describe el diseño después de haber sido fabricado.

Concluyo también,que el diseño de un sistema de información, produce los detalles y establcen la forma en la que el sistema cumplirá los requerimientos identificados durante la fase de análisis. Los especialistas en sistemas con frecuencia llaman a ésta etapa como diseño lógico. El proceso de diseño se comienza identificando los reporte y salidas que debe producir el sistema, y también los datos de entrada que serán calculados y los que deben ser alamacenados.

Por otra parte llego a la conclusión de que el diseño lógico es una herramienta que minimiza, actualiza y hace más eficiente la solución a los problemas lógicos.

Podemos decir que el diseño lógico, ha ayudado grandemente al avance tecnológico de las computadoras, sobre todo ha facilitado enormemente la creación de circuitos digitales al tomar sólo dos valores fijos.

Otro punto importante al que he llegado, es que las herramientas escenciales para el diseño de sistemas lógicos es el álgebra booleana, que permite expresar en téminos matemáticos el problema y trabajar con ecuaciones los conceptos lógicos para clasificar la aplicación de esta herramienta matemática.

Por otra parte puedo concluir que los mapas de karnaugh, son una respresentación gráfica de todas las combinaciones únicas de todas las variables involucradas. También es una técnica de minimización que sirve al diseñador de sistemas lógicos a reducir a la forma más simple y más económica, un sistema lógico.

La recomendación que hago a todo profesional dedicado al diseño de sistemas de información, es el estudio de el álgebra booleana, así como las técnicas especiales de minimización orientados a obtener un codigo pequeño y depurado, porque como he podido analizar, son la base del diseño lógico.

8.0.- ***BIBLIOGRAFIA***

1.- Fundamentos de Computadores Digitales
- Thomac. Batee
- McGrawHill

2.- Diseño Digital
- Morris Mano
- McGrawHill

3.- Diseño de Sistemas Digitales y Microprocesadores
- John P. Hayes
- McGrawHill

4.- Lógica Digital y Diseño de Computadoras
- M. Morris Mano
- Prentice Hall

5.- Fundamentos del Diseño Digital
- Cesar Rene de la Cruz Lazo

6.- Diseño Digital Principios y Prácticas
- John F. Wakerly
- Prentice Hall

7.- Principios de Diseño Digital
- Daniel D. Gajski
- Prentice Hall

8.- Teoría de Conmutación y Diseño Lógico
-Frederic T. Hill
-Limusa

9.- Introducción al Diseño lógico
-Sajjan G. Shiva
-Trillas

10.- Diseño Digital, Principios y Prácticas
-Jonh F. Wakerly
Prentice Hall

11.- Introducción al Diseño Lógico Digital
-Jonh P. Hayes
Adison Wesley

12.- Diseño Lógico de Circuitos
-Arturo Martínez
-Arturo Zepeda
-Marco A. Reyes
-Cuauhtémoc Utrera
-Francisco Rodríguez
Limusa

13.-Arquitectura de Computadores
-M. Morris Mano
Prentice Hall

14.-Diseño de Circuitos Integrados TTL
-Robert L. Morris
-John R. Miller
CECSA

15.-Lenguaje Ensamblador Para Microcomputadores IBM
-J. Terry Godfrey
Prentice Hall