UNIDAD 3: Circuitos lógicos y digitales

Introducción

Un **Sistema** es un conjunto de elementos que guardan una relación entre sí, a su vez un elemento del sistema puede ser otro sistema (**subsistema**). Los Sistemas se clasifican:

SISTEMAS

NATURALES

.

.

ARTIFICIALES

.

.

ELÉCTRICOS

.

.

ELECTRÓNICOS

ANALÓGICOS

DIGITALES

COMBINACIONALES

SECUENCIALES

Se concluye que un **sistema digital** es aquel cuyos elementos son digitales (sólo pueden adoptar valores discretos). En la Unidad 2 se llegó a la conclusión que la base 2, para la elección de un sistema de numeración, era la más adecuada desde el punto de vista de la confiabilidad y el costo. Por esta razón los Sistemas Digitales trabajan con **elementos binarios** (sólo pueden adoptar dos valores). Para poder realizar el estudio de los Sistemas Digitales se necesita estudiar una álgebra binaria. El **Álgebra de George Boole**, que data de 1854, es sin dudas la más apropiada para nuestro fin. Claude Shannon en 1938 adaptó esta álgebra para la aplicación en sistemas digitales.

Seguidamente se estudia el álgebra de Boole, las funciones booleanas, las compuertas lógicas, los Sistemas Combinacionales y, finalmente, los Sistemas Secuenciales.

Álgebra de Boole

Postulados y teoremas

Dentro de las álgebras de Boole, es de utilidad definir la **bivalente**, es decir compuesta por sólo **dos elementos**. Así, el álgebra es un conjunto de elementos binarios relacionados entre sí mediante las operaciones lógicas producto [**.**] y suma [+], que cumplen con los siguientes postulados (las letras a, b, c, etc., indican variables binarias):

1) Existe el **elemento identidad**

$a + 0 = a$
$a \cdot 1 = a$

2) Las dos operaciones cumplen con la **propiedad conmutativa**

$a + b = b + a$
$a \cdot b = b \cdot a$

3) Propiedad distributiva

$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
$a + (b \cdot c) = (a + b) \cdot (a + c)$

4) Complementación o **inversión lógica**

$a + a' = 1$
$a \cdot a' = 0$

Algunos teoremas importantes son:

1) **Dualidad**: Toda igualdad lógica sigue siendo válida si se intercambian los operadores (+ y .) y los elementos de identidad (0 y 1). La simetría de los postulados demuestra este teorema.

2) **El álgebra es un conjunto cerrado**; es decir, los resultados de aplicar las operaciones lógicas a las variables, pertenecen al álgebra.

3) En el álgebra se cumple que

a + 1 = 1
a . 0 = 0

4) Ley de Idempotencia

a + a = a
a . a = a

5) Ley de involución

(a')' = a

6) Las operaciones lógicas son asociativas

a + (b + a) = (a + b) + c
a . (b . c) = a . (b . c)

7) **Absorción**:

a = a + (a . b)
a = a . (a + b)

8) Leyes de De Morgan

(a + b + c + d + ...+ n)' = a' . b' . c' . d' .....n'
(a . b . c . d .....n)' = a' + b' + c' + d' + ...+ n'

Con excepción del teorema 1, siempre aparecen dos expresiones, obsérvese que la segunda es la dual de la primera. Se recomienda al alumno demostrar estos teoremas en forma algebraica basándose en los postulados.

Aún cuando las operaciones **+** y **.** son distributivas entre sí, de ahora en más prescindiremos de los paréntesis que encierran los productos lógicos. Además el símbolo del producto no se indicará en lo sucesivo. De esta forma, por ejemplo, la expresión

a + (b . c) . (d + e)

se escribirá

a + b c (d + e)

Funciones lógicas

Una **función lógica** es una variable binaria que depende de otras variables binarias relacionadas entre sí por las operaciones lógicas. Una función lógica se nota de la siguiente manera:

f(a ,b ,c ,....., n) = {expresión lógica que involucra a las variables a ,b ,c , d,....., n}

La función adoptará el valor 0 o 1 de acuerdo a la expresión y al valor determinado de las variables. Por ejemplo:

f(a ,b, c) = a b' + a c

Se trata de una función de tres variables a la cual le corresponde la siguiente **Tabla de Verdad,** ver figura 1. Puede decirse que la tabla de verdad es otra forma de expresar una función lógica.

C	B	A	F(a, b, c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Figura 1

Teoremas de funciones lógicas

8) En el Álgebra de Boole se cumple

F(a, b, c,.....n) = a f(1, b, c,....n) + a' f(0, b, c,.....n)

Para demostrar esta igualdad basta con reemplazar *a = 1* y *a = 0* en la expresión y verificar que la misma se cumple en ambos casos. También, considerando que la función en cuestión no tiene restricciones, se puede decir que también es válida su dual:

F(a, b, c,.....,n) = [a + f(0, b, c, ...n)] [a' + f(1, b, c,....n)]

Y se trata de una función cualquiera.

Este teorema posee corolarios muy útiles a la hora de simplificar (obtener una expresión más simple de la misma función) funciones (expresiones en general) lógicas. Se obtienen efectuando el producto miembro a miembro de la primera expresión por a o por a', como se indica a continuación:

a f(a, b, c, ...n) = a [a f(1, b, c, ...n) + a' f(0, b, c,.....n)]

aplicando propiedad distributiva al segundo miembro, se obtiene:

a f(a, b, c, ...n) = a f (1, b, c, ...n) **Primer Corolario**

a' f(a, b, c, ...n) = a' [a f(1, b, c, ...n) + a' f(0, b, c,.....n)]

aplicando propiedad distributiva al segundo miembro, se obtiene:

a' f(a, b, c, ...n) = a' f (0, b, c, ...n) **Segundo Corolario**

Aplicando dualidad a los corolarios, se obtienen:

a + f(a, b, c, ...n) = a + f(0, b, c, ...n) **Tercer Corolario**

y

a' + f(a, b, c, ...n) = a' + f(1, b, c, ...n) **Cuarto Corolario**

9) Toda función lógica puede expresarse en forma canónica, es decir:

- Como una **sumatoria de términos** en los cuales aparecen todas sus variables en forma de producto lógico (estos términos se llaman MINTERMS)
- O como una **productoria de términos** en los cuales aparecen todas sus variables en forma de suma lógica (estos términos se llaman MAXTERMS).

En ambos casos la función se dice expresada en forma canónica y sus términos (ya sean minterms o maxterms se llaman **términos canónicos**).

Se demostrará este teorema para una función de dos variables f(a, b), luego se hará extensivo para n variables.
Aplicando el teorema 1 a f(a, b), se tiene:

F(a, b) = a f(1, b) + a' f(0, b)

Aplicando nuevamente el teorema 1 a f(1, b) y a f(0, b), se tiene:

F(1, b) = b f(1, 1) + b' f(1, 0)
F(0, b) = b f(0, 1) + b' f(0, 0)

Reemplazando en la expresión inicial se obtiene

F(a, b) = a b f(1, 1) + a b' f(1, 0) + a' b f(0, 1) + a' b' f(0, 0)

Se observa entonces que toda función puede expresarse como una sumatoria de todos sus minterms, afectados cada uno de ellos por un coeficiente que consiste en el valor de la función (calculado reemplazando las variables por 1 o por 0 sí, en el minterm que acompaña, la variable correspondiente se encuentra directa o negada respectivamente).

Teniendo en cuenta que f(a, b) es una función cualquiera del álgebra de Boole, su dual también lo será, por lo tanto:

$$F(a, b) = [a + b + f(0, 0)]\ [a + b' + f(0,1)]\ [a' + b + f(1, 0)]\ [a' + b' + f(1, 1)]$$

Se observa entonces que toda función puede expresarse como una productoria de todos sus maxterms, afectados cada uno de ellos por un coeficiente que consiste en el valor de la función (calculado reemplazando las variables por 0 o por 1 sí, en el maxterm que acompaña, la variable correspondiente se encuentra directa o negada respectivamente).

La generalización de los resultados obtenidos para funciones de n variables, resulta evidente.

A fin de obtener una notación más sencilla de las funciones lógicas, se suele asignar a cada término canónico un número decimal que se obtiene dando pesos a las variables de acuerdo a sí las mismas se encuentran expresadas en forma directa o negada. El convenio es el siguiente:

VARIABLE	PESO
A	1
B	2
C	4
D	8
E	16
F	32
----	-----

Figura 2

Si la variable aparece en forma negada, el peso asignado es cero.
Según el convenio entonces, el término canónico cualquiera $a'\ b\ c'\ d$ correspondiente a un minterm de una función de cuatro variables, tendrá el número decimal 10.

El convenio mencionado permite una tercer forma, llamada compacta, de notar una función, a saber:

$$F(a, b, c,....n) = \sum_{(i=0,\ 2^n-1)} i\ f(i) = \prod_{(i=0,\ 2^n-1)} [(2^n - 1 - i) + f(i)]$$

De la expresión anterior se deduce una regla para pasar de una función canónica en minterms a una en maxterms y viceversa: *Se buscan los términos canónicos que no están en la expresión de la función, y se los complementa a $2^n - 1$. Estos serán los términos de la función buscada.*

Por ejemplo:

Sea la función de 4 variables

$$F(a, b, c, d) = \Sigma_4\ (0, 1, 3, 5, 6, 7, 10, 13, 14, 15)$$

Los términos canónicos que no están son: 2, 4, 8, 9, 11 y 12. Sus complementos a 15 son: 13, 11, 7, 6, 4 y 3. Por lo tanto la expresión canónica en maxterms de la función es:

$$F(a, b, c, d) = \Pi_4\ (3, 4, 6, 7, 11, 13)$$

Nótese que, a modo de verificación, la suma del número de minterms y maxterms de una función, siempre es igual a $2^n - 1$.

Minimización de funciones lógicas

Es importante obtener la mínima expresión posible de una función, esto es la menor cantidad de variables y operaciones involucradas. Los métodos de minimización se basan en los postulados del álgebra y a la conveniencia de agregar oportunamente términos en la expresión de la función.

Para aplicar los métodos es necesario que la función esté expresada en **forma canónica**. Como se vio en el punto anterior, toda función lógica es expresable en forma canónica, ya sea en minterms o maxterms.

Supóngase que una función canónica de 4 variables posee en su expresión los siguientes términos canónicos:

$$.....+ a' b c d' + a b c d' + ...$$

Se observa que puede sacarse factor común de la siguiente forma

$$.....+ b c d' (a' + a) + ...$$

Según el postulado 4, $a' + a = 1$, por lo tanto

$$.....+ b c d' 1 +..... = ...+ b c d' + ...$$

Se ha perdido la variable *a*.

Este procedimiento se sistematiza detectando todos los términos canónicos de la función que difieran en el estado (directo o negado) de sólo una variable, se saca factor común entre ellos y se van eliminando variables.

Sea el siguiente ejemplo:

$$F(a, b, c, d) = \Sigma_4 (0 , 4, 8, 12)$$

La expresión algebraica de la misma es:

$$F(a, b, c, d) = a' b' c' d' + a' b' c d' + a' b' c' d + a' b' c d$$

Se ve que los dos primeros son adyacentes, como así también los dos últimos. Puede sacarse factor común:

$$F(a, b, c, d) = a' b' d' (c' + c) + a' b' d (c' + c) = a' b' d' + a' b' d$$

Los dos términos que quedan, si bien no canónicos, son adyacentes, quedando finalmente:

$$F(a, b, c, d) = a' b' (d' + d) = a' b'$$

E. W. Veitch en 1952, propuso un método gráfico para la identificación de los términos adyacentes de una función. Posteriormente Maurice **Karnaugh** lo modificó tal como se conoce actualmente. Consiste en mapas aplicables a funciones de dos, tres, cuatro y cinco variables. Para funciones de más variables no resulta práctico este método gráfico, se usa un método numérico que no se estudia en este curso.

Los mapas Karnaugh son los siguientes:

Para funciones de 4 variables:

Los dos números binarios en las columnas y las filas, que siguen un código Gray de dos variables, se corresponden con las variables directas o negadas de cada cuadro, y los números decimales son los asignados a cada término canónico según la convención indicada con anterioridad. Esta tabla genérica puede particularizarse

c d \ *a b*	00	01	11	10
00	0	2	3	1
01	8	10	11	9
11	12	14	15	13
10	4	6	7	5

para una función determinada marcando en la misma con un 1 los términos canónicos que forman parte de la función. De esta forma es sencillo identificar los términos

Figura 3

canónicos adyacentes que serán los que limitan por los lados. Por ejemplo, el término canónico 14, posee cuatros términos adyacentes que son: 6; 10; 12 y 15.

Formar un grupo entre dos unos colindantes en el mapa se corresponde con sacar factor común

y perder la variable que cambia. Es de suponer la conveniencia de realizar los grupos que contengan mayor cantidad de unos en su interior. Pero esto debe seguir ciertas reglas.

Sea la función de 4 variables siguiente:

F(a, b, c, d) = S (0, 1, 2, 3, 6, 7, 8, 9, 10, 11, 14, 15)

El mapa que le corresponde es el indicado en la figura 4.

El grupo 0-2 corresponde a sacar factor común con pérdida de la variable *b*.

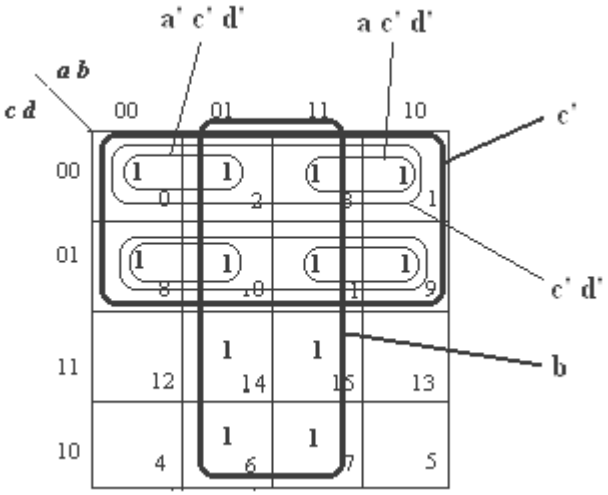


Figura 4

El grupo 3-1 pierde la variable *b*.

Se observa que estos dos grupos son adyacentes y se pueden juntar en un solo grupo 0-1-2-3 donde se pierden las variables a y b. El mismo razonamiento es válido para el grupo 8-9-10-11 que pierde las variables *a* y *b*.

Se observa que estos grupos son adyacentes y podría formarse un solo grupo 0-1-2-3-8-9-10-11 donde sólo queda la variable c'. Para el grupo vertical de 8 unos se ha seguido el mismo procedimiento, cabe aclarar que los términos canónicos 2, 3, 10 y 11 se han usado dos veces. Esto puede realizarse ya que según el teorema 5, un termino canónico podría repetirse cuantas veces se quiera sin alterar el valor de la función.

La función minimizada queda por lo tanto

F(a, b, c, d) = b + c'

Cabe aclarar que la última expresión es una suma porque la función inicial estaba en minterms, es decir era una sumatoria.

De lo visto pueden enunciarse la siguiente **regla de formación** de grupos:

a) Se agrupan la mayor cantidad de unos posible, siempre que sean una potencia de dos y el grupo resultante pueda subdividirse en grupos menores.

b) Se agrupan los unos restantes siguiendo la regla a), pudiendo usar (si es conveniente) un uno ya agrupado anteriormente

c) Se repite b) hasta realizar todos los unos.

Para el caso de funciones de tres y de dos variables las tablas son más pequeñas y la regla de formación de grupos es la misma. Se invita al alumno a sugerir cómo serías estas tablas y visitar el Práctico correspondiente resolviendo los ejercicios propuestos.

Compuertas lógicas

La realización práctica (implementación) de las funciones lógicas se hace por medio de las compuertas lógicas que son la base constructiva de la electrónica digital.

No todas las funciones lógicas presenta interés práctico. En la figura 5 se muestran las compuertas lógicas más comunes.

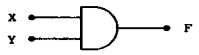
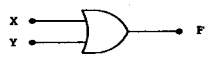
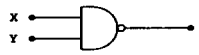
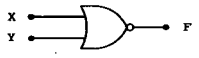
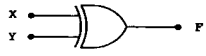
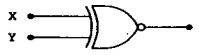
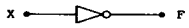
Nombre	Símbolo	Función	Tabla
AND		F=xy	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		F=x+y	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NAND		F=(xy)'	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		F=(x+y)'	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		F=xy'+x'y =x⊕y	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
XNOR		F=xy+x'y' =xOy	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1
NOT (Inversor)		F=x'	x | F 0 | 1 1 | 0
Buffer		F=x	x | F 0 | 0 1 | 1

Figura 5

En la figura aparecen compuertas de dos entradas. Existen compuertas de más entradas disponibles comercialmente en circuitos integrados (chips) en SSI. En función de la cantidad de compuertas por chip, se suele clasificar a los CI en escalas de integración:

- SSI, escala de integración pequeña, hasta 10 compuertas por CI
- MSI, escala de integración media, de 10 a 100 compuertas por CI
- LSI, escala de integración grande, de 100 a 1000 compuertas por CI
- VLSI, escala de integración muy grande, más de 1000 compuertas por CI.

A la hora de implementar una función lógica es cuando se torna importante la minimización. Por ejemplo, sea la función:

$F(x, y, z) = \Sigma_3 (2, 4, 5, 6)$

Si implementamos esta función sin minimizar, obtenemos el circuito de la figura 6

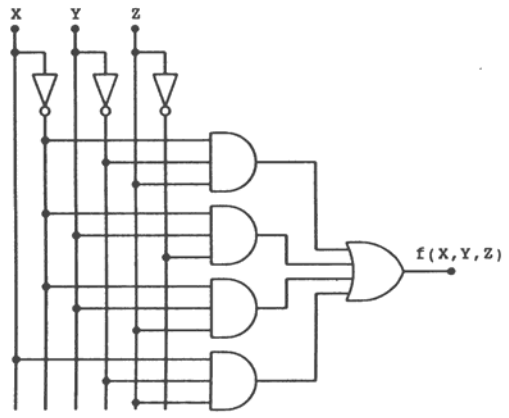


Figura 6

Se invita al alumno a minimizar la función y comparar los resultados obtenidos.

Sistemas digitales

Un sistema digital es un conjunto de elementos binarios relacionados entre si de alguna manera. Se distinguen dos tipos de variables en un sistema digital. Las variables de entrada y las variables de salida que dependen de las de entrada. Funcionalmente las variables de entrada se dividen en dos grupos: variables de proceso y variables de control. Ver figura 7

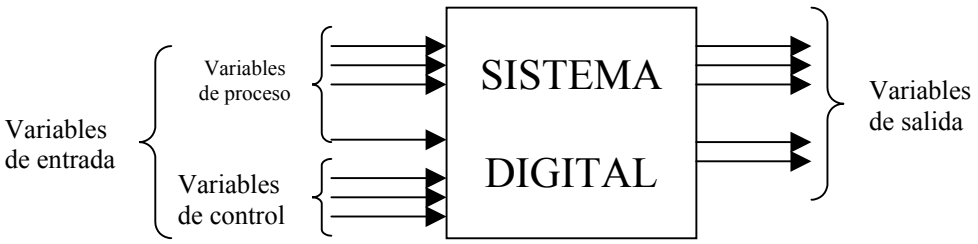


Figura 7

Cuando cada combinación de las variables (Vector de entrada) de entrada se corresponde con una y sólo una combinación de las variables de salida (Vector de salida), se trata de un sistema combinacional. Dicho de otra manera, siempre que se repita un conjunto de valores de las variables de entrada, se repetirá la salida. En la fig. 8 se muestran las correspondencias entre entradas y salidas de un sistema combinacional

Cuando a un mismo vector de entrada puede corresponder más de uno de salida, el sistema se llama secuencial. Dicho de otra manera cuando se repite un conjunto de valores de las variables de entrada, no necesariamente se repetirá la salida. Los sistemas secuenciales deben poseer memoria interna ya que sus salidas son consecuencia de la evolución anterior de sus entradas. En la figura 9 se muestra las correspondencias de un sistema secuencial.

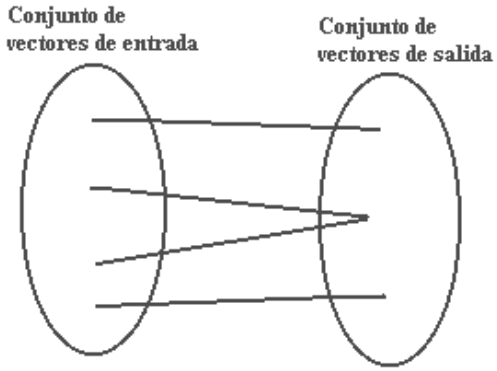


Figura 8. Correspondencias para un Sistema Combinacional

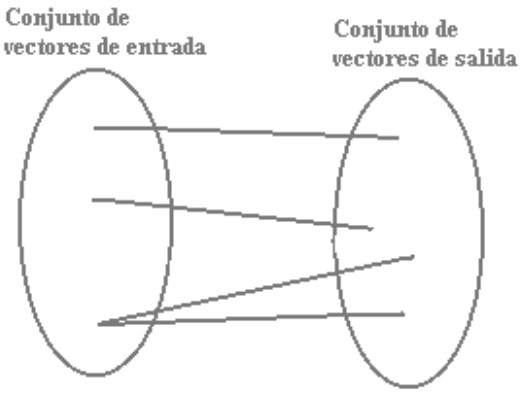


Figura 9. Correspondencias para un Sistema Secuencial

Sistemas Combinacionales

De lo definido en el punto anterior se concluye que en un Sistema combinacional las salidas no son otra cosa que funciones lógicas de las entradas.

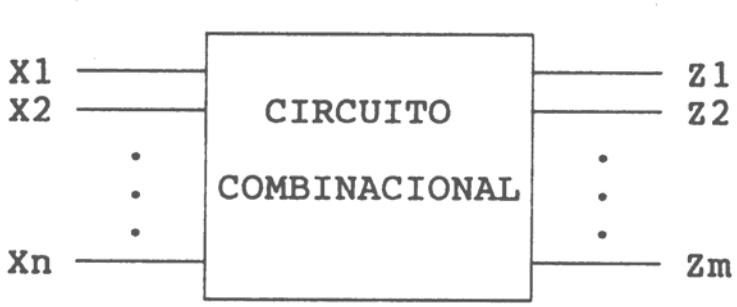


Figura 10

En la fig. 10 se ve el diagrama en bloque de un combinacional con n entradas y m salidas. Se puede escribir que:

$z_i = f_i (x_1, x_2, ... x_n)$

De lo anterior se deduce que para diseñar un circuito combinacional bastará con minimizar las funciones requeridas y finalmente implementar con compuertas lógicas.

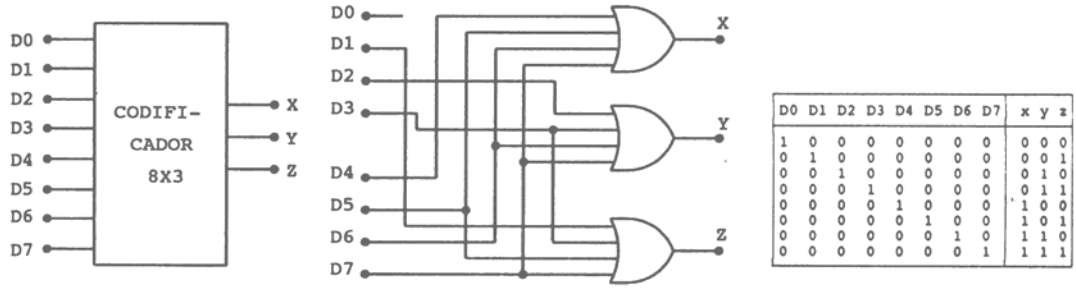
Circuitos Combinacionales MSI

Cuando las funciones lógicas son muy complejas no siempre el diseño basado en la minimización y posterior implementación con compuertas lógicas, es el más adecuado. Las técnicas de integración han permitido CI más complejos. Por ejemplo en MSI se dispone de CI de hasta 100 puertas. Estos bloques funcionales MSI, si bien a veces tienen fines específicos, pueden aplicarse a la implementación de funciones lógicas de muchas variables. Las ventajas caen en la disminución de los CI necesarios, del tiempo de diseño, del número de conexiones externas y facilita el mantenimiento.

A continuación se describen brevemente los Combinacionales MSI más comunes

Codificadores

Permiten codificar las líneas de entrada. Generalmente codifican en binario o BCD. En la figura 11 se



D0	D1	D2	D3	D4	D5	D6	D7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

muestra un codificador binario de 8 entradas y 3 salidas, su circuito interno y su tabla de verdad.

Figura 11

En este codificador se supone que sólo esta activa una entrad por vez. En caso de no ser así la salida debe calcularse como la función OR bit a bit de las salidas correspondientes a las entradas activadas independientemente. Estos decodificadores se llaman **sin prioridad**.

Si en la tabla de verdad de la fig. 11 se reemplazan con x los ceros a la izquierda de los unos de las entradas, se obtiene un codificador **con prioridad**. La entrada de mayor prioridad es la que define la salida.

Si ninguna entrada está activa las salidas son todas cero, igual que si estuviera activada la entrada D0. Para evitar este problema los codificadores cuentan con una salida adicional que indica la ausencia de activación de las entradas.

Por último los codificadores suelen contar con una entrada de habilitación. Cuando el chip está activado es válida la tabla de verdad, si no lo está el chip no funciona.

Decodificadores

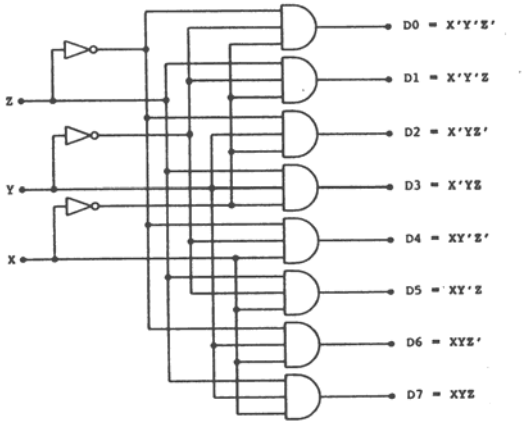
Son Combinacionales que poseen n entradas y m salidas. El orden adecuado de la salida se activa cuando la codificación correspondiente se inyecta ala entrada. Generalmente son binarios o BCD. En caso de un decodificador binario si tiene n entradas poseerá m = 2n salidas. Así un decodificador realiza lo opuesto a un codificador.

En la figura 12 se muestra un decodificador de 3 x 8 y su tabla de verdad.

Los decodificadores, además de usarse para decodificar, son útiles para implementar funciones lógicas. Cada una de sus salidas es un minterm de una función de n variables. Aprovechando la entrada de habilitación que suelen tener, es posible aumentar el número de variables.

En la figura 13 se usa un decodificador de 3 x 8 para implementar la función

$F(z, y, x) = \Sigma_3 (1,3,6,7)$



x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	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

Figura 12

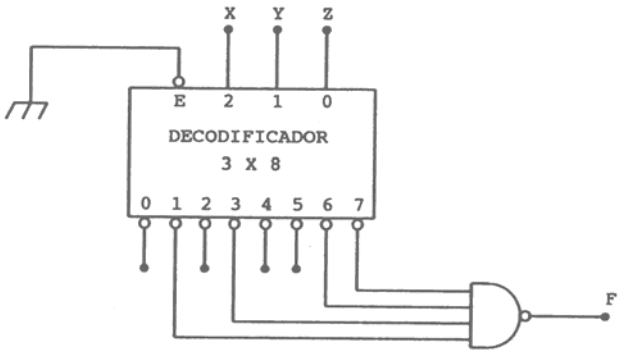


Figura 13

Se ve la entrada E de habilitación. Si E = 0 el decodificador está habilitado, si E = 1, cualesquiera sean los valores de x, y, o z, ninguna salida se activará.

La figura 14 muestra como obtener un decodificador de 4 x 16, partiendo de dos decodificadores 3 x 8.

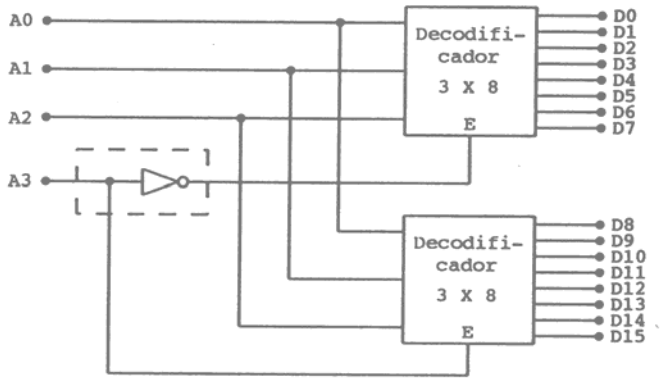


Figura 14

Multiplexores

Disponen de $m = 2^n$ líneas de entrada (canales), una línea de salida y n líneas de selección. En función de las líneas de selección determina qué entrada aparece en la salida. La figura 15 indica la función de un multiplexor y la figura 16 el circuito de un multiplexor de 4 canales.

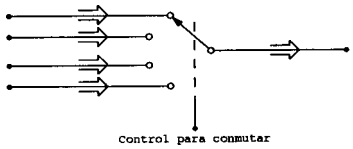


Figura 15

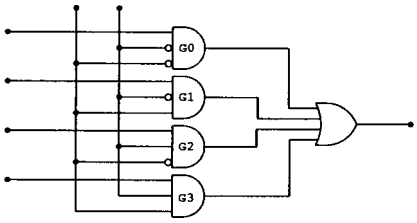


Figura 16

Los multiplexores , además de multiplexar, pueden usarse eficazmente para implementar funciones lógicas. Supongamos que la función a implementar sea:

$F(a, b, c) = \Sigma\ (0,1,5,6,7)$

Para implementar una función de 3 variables se necesita un multiplexor de 3 – 1 entradas de selección. Dos de las variables (por ejemplo a y b) se conectan a las líneas de selección. La tercer variable c, se conecta a los canales. A esta altura es conveniente contar con la tabla de verdad de la función, que en nuestro ejemplo es:

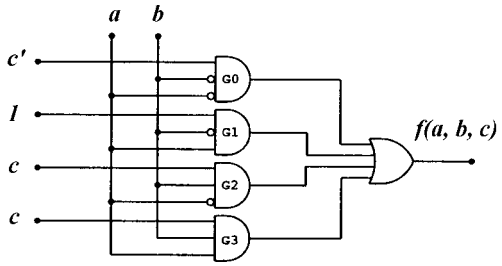
C	B	A	F(a, b, c)
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

De la tabla de verdad de la función, se construye la siguiente tabla auxiliar:

B	A	F(a, b, c)
0	0	C’
0	1	1
1	0	C
1	1	C

Esta tabla auxiliar se obtiene de la anterior verificando cuanto vale la función para las diferentes combinaciones de a y b, y permite determinar qué valores conectar a los canales del multiplexor. Ver Figura 17.

Figura 17



El procedimiento puede generalizarse para n variables, todas menos una se conectan a las líneas de selección del multiplexor. La restante a los canales de acuerdo a la tabla auxiliar.

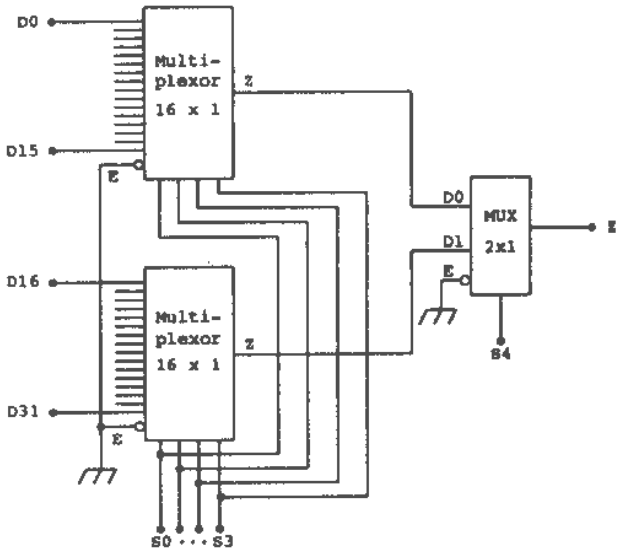


Figura 18

Para realizar multiplexores de muchos canales pueden combinarse diferentes multiplexores. Por ejemplo en la figura 18 de muestra un multiplexor de 32 canales a partir de dos de 16 canales y uno de dos canales. La entrada E (habilitación) es activa con un 0. Si E = 1, la salida del multiplexor es 0 independientemente del valor de las entradas.

Demultiplexores

Cumplen la función opuesta a los multiplexores. Tienen una entrada y m salidas y n entradas de selección. La salida seleccionada tendrá el valor de la entrada. En la figura 20 se muestra un demultiplexor de cuatro canales de salida.

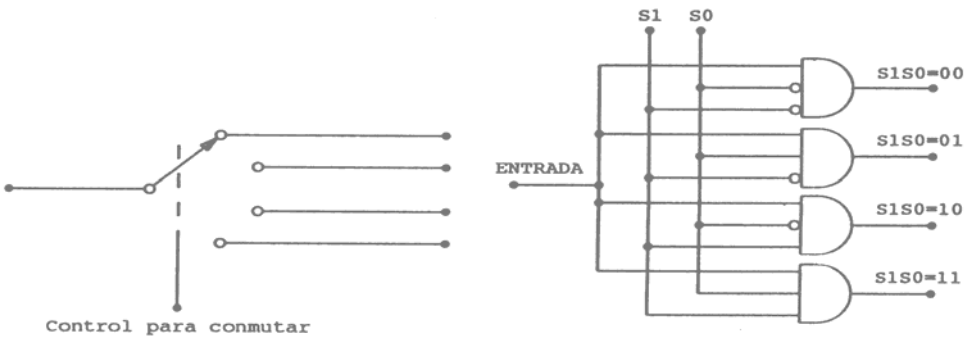


Figura 19

El circuito de un demultiplexor es coincidente con un decodificador que posea entrada de habilitación. Por esta razón no se encuentran demultiplexores específicos. En la figura 20 se indica como obtener un demultiplexor de cuatro canales desde un decodificador de 2 x 4 con entrada de habilitación.

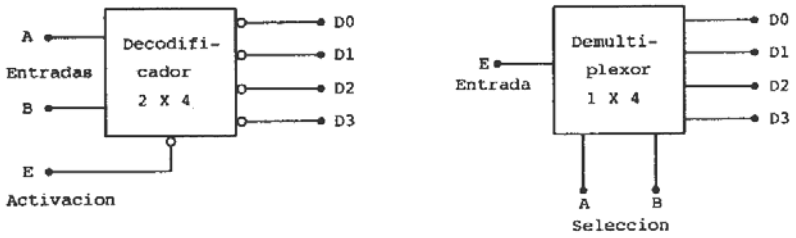


Figura 20

Es usual encontrar en algunas familias lógicas multiplexores/demultiplexores. Estos circuitos pueden cumplir ambas funciones.

Comparadores

Realizan la comparación entre dos números binarios de n bits. El circuito básico que realiza la comparación de 1 bit, se indica en la figura 21.

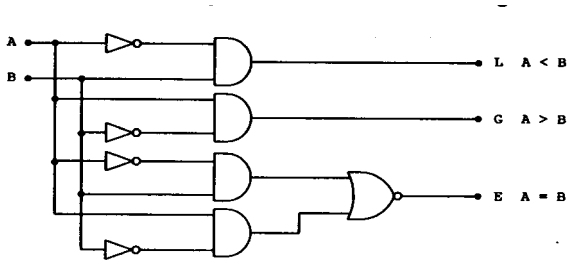
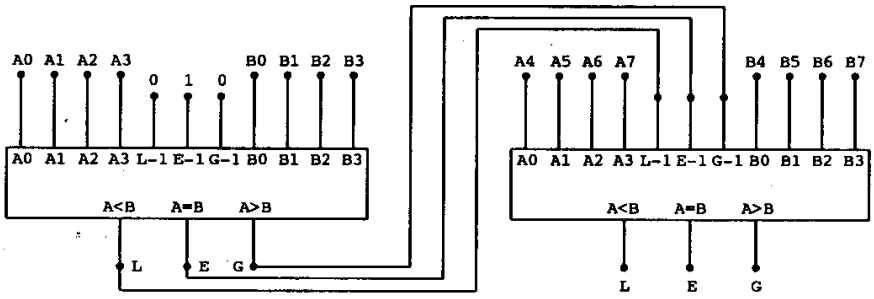


Figura 21

Este circuito responde a la siguiente tabla de verdad:



B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Figura 22

Comparadores de más bits se diseñan de la misma manera. Los Comparadores poseen además entradas por =, <, y >, esto permite realizar comparadores de elevado número de bits, partiendo de comparadores

menores. Por ejemplo, en la figura 22 se muestra un comparador de 8 bits, partiendo de 2 comparadores de 4 bits.

Detectores/generadores de paridad

Son CI capaces de generar/detectar la paridad de un conjunto de bits. En la figura 23 se muestra un generador/detector de paridad de 8 bits, su circuito, tabla de verdad y esquema del chip (se trata del 74180 de tecnología TTL)

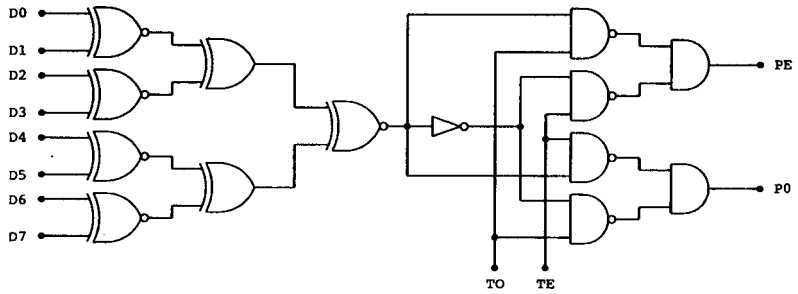


Figura 23

Las señales de control TO (paridad impar) y TE (paridad par) permiten seleccionar la paridad. Se recomienda al alumno obtener la tabla de verdad de este circuito y verificar su funcionamiento.

Sumadores

Son CI que realizan la suma aritmética de dos números de n bits. Antes de ver los sumadores disponibles en escala de integración MSI, estudiaremos la suma y resta binaria.

Suma binaria

Para indicar la suma aritmética utilizaremos el símbolo $+$, para diferenciarlo del + usado para la suma lógica.

Para sumar dos bits, se puede implementar el circuito de la figura 24, llamado Semisumador

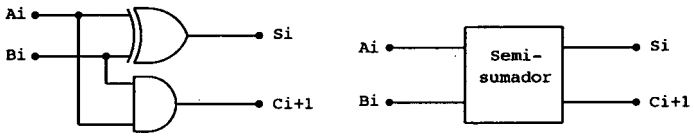


Figura 24

Cuya tabla de verdad es:

B_i	A_i	S_i	C_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Supóngase ahora que se desea sumar dos números binarios de cuatro bits A y B, entonces:

$$\begin{array}{rl} & c_4\ c_3\ c_2\ c_1\ c_0 \\ \boldsymbol{A} = & a_3\ a_2\ a_1\ a_0 \\ + & \\ \boldsymbol{B} = & b_3\ b_2\ b_1\ b_0 \\ \hline \boldsymbol{S} = & s_3\ s_2\ s_1\ s_0 \end{array}$$

Se observa que son necesarios cuatro circuitos, uno para cada columna, y cada uno debe ser capaz de sumar tres bits: a_i, b_i, y c_i. Se implementa entonces el circuito de la figura 25, llamado Sumador Total.

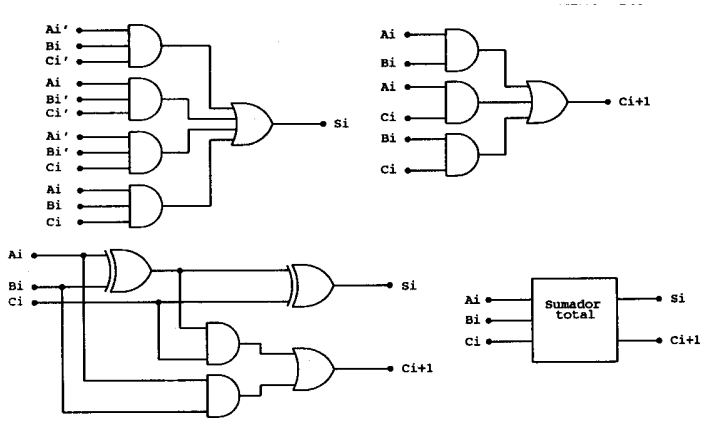


Figura 25

Cuya tabla de verdad es:

Ci	Bi	Ai	Si	Ci+1
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

La interconexión de cuatro **Sumadores Totales** permite obtener un **Cuádruple Sumador Total**, capaz de realizar la suma aritmética de dos números binarios de cuatro bits, ver figura 26

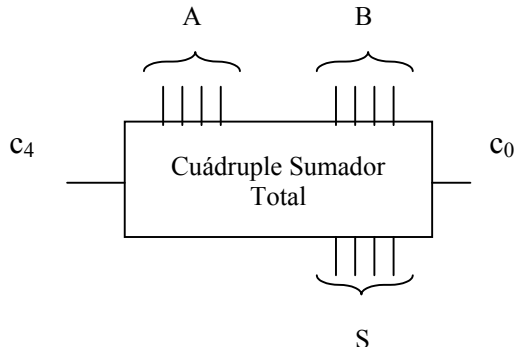


Figura 26

Resta binaria

Debe recordarse los convenios de representación de números negativos en binario.

Se podría implementar un circuito para realizar la resta como una nueva operación. Sin embargo se verá que es posible restar dos números realizando la suma de uno de ellos mas el complemento a dos (o a uno) del otro.

Para el caso de usar el convenio de complemento a dos.

Sean A y B dos números binarios signados en convenio de complemento a dos, véase el siguiente desarrollo:

$$A - B \,/\, A + C_2(B) = A + 2^n - B = 2^n + (A - B)$$

En la expresión se observa que el resultado obtenido difiere del buscado en el valor 2^n. Este resultado debe interpretarse como un acarreo a despreciar si el paréntesis (A - B) resulta positivo. En caso que el paréntesis resulte negativo el resultado estará expresado en complemento a dos.

Para el caso de usar complemento a uno

Sean A y B dos números binarios signados en convenio de complemento a uno, véase el siguiente desarrollo:

$$A - B \,/\, A + C_1(B) = A + 2^n - 1 - B = 2^n + (A - B) - 1$$

En la expresión se observa que el resultado obtenido difiere del buscado en el valor 2^n y además tiene un error en defecto de valor 1. Este resultado debe interpretarse como un acarreo que debe sumarse al resultado si el paréntesis (A - B) resulta positivo. En caso que el paréntesis resulte negativo el resultado estará expresado en complemento a uno.

Es conveniente que el alumno verifique el párrafo anterior par todos los casos, es decir

- $A > 0 \; ; \; B > 0$
- $A > 0 \; ; \; B < 0$
- $A < 0 \; ; \; B > 0$
- $A < 0 \; ; \; B < 0$

En la figura 27 se muestra un Sumador/restador en complemento a dos de 4 bits a partir de un Cuádruple sumador total

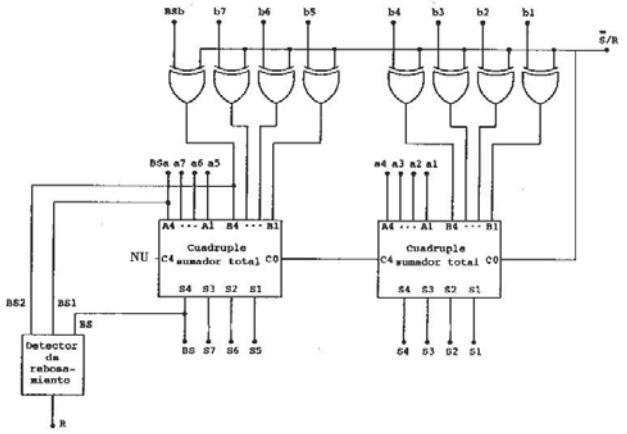


Figura 27

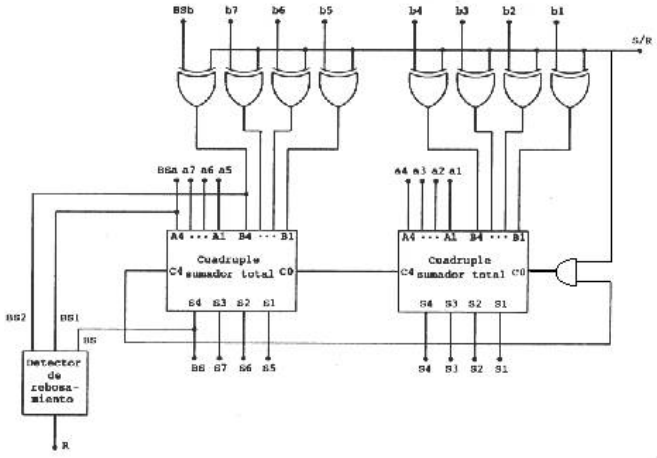


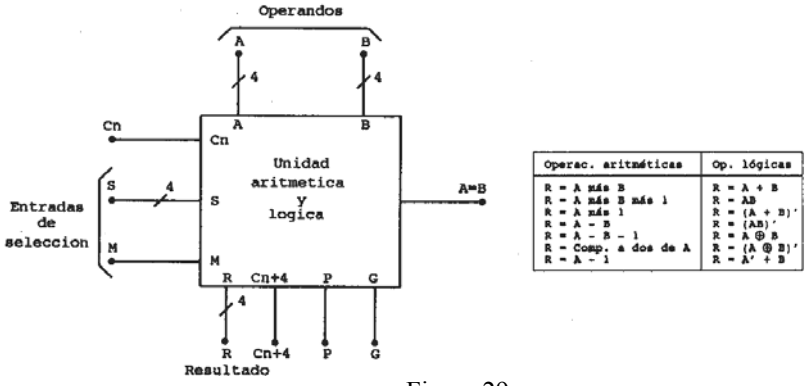
Figura 28

En la figura 28 se muestra un Sumador/Restador de 4 bits en complemento a uno a partir de un cuádruple sumador total.

En las dos últimas figuras aparece un circuito detecto de rebasamiento. Su salida será 1 si se ha rebasado al sumador y cero en caso contrario. Nótese que las entradas a este circuito son los bits de signo de los números de entrada y del resultado.

Unidad Lógico–Aritmética (ALU)

Son bloques funcionales en escala MSI que permiten realizar operaciones lógicas y aritméticas sobre números binarios (generalmente de 4 bits). La operación a realizar se selecciona colocando los valores adecuados en las líneas de selección. En la figura 29 se muestra una ALU típica de 4 bits.



Operac. aritméticas	Op. lógicas
R = A más B	R = A + B
R = A más B más 1	R = AB
R = A más 1	R = (A + B)'
R = A - B	R = (AB)'
R = A - B - 1	R = A ⊕ B
R = Comp. a dos de A	R = (A ⊕ B)'
R = A - 1	R = A' + B

Figura 29

Estos bloques funcionales pueden conectarse en cascada para realizar operaciones sobre números de mayor número de bits

Sistemas Secuenciales
Introducción

Son aquellos Sistemas Digitales cuyas salidas no sólo dependen de sus entradas en un momento dado, sino también de cómo han evolucionado estas anteriormente. El Sistema Secuencial tiene que ser capaz de memorizar la mencionada evolución. Puede decirse que las salidas de un Sistema Secuencial dependen de ellas mismas y de las entradas. Este concepto es equivalente al anterior y permite plantear un esquema general de Sistema Secuencial partiendo de un Sistema Combinacional realimentado (las entradas a este combinacional están formadas por variables independientes y además por una o más salidas del mismo. Lo mencionado puede verse en la figura 30.

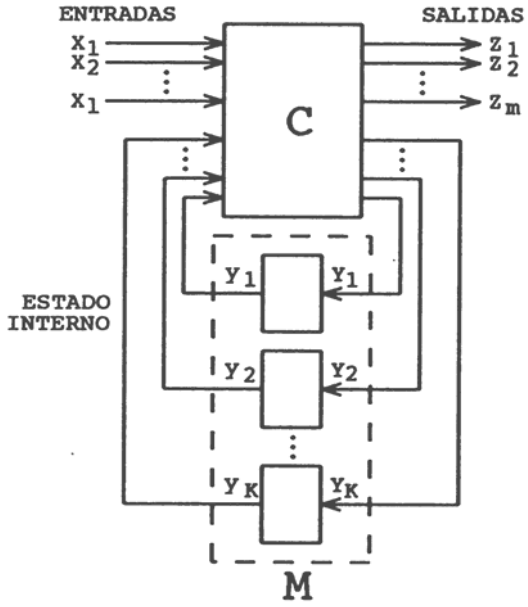


Figura 30 – Sistema Secuencial Asíncrono

Se observa un nuevo tipo de variables llamadas variables internas. El bloque M, indica un circuito capaz de mantener el estado de sus entradas en su valor, por un cierto tiempo. El alumno puede deducir que el sistema evolucionará entre distintos estados internos hasta arribar a un estado estable. Efectivamente, para un valor de las variables de entrada determinado, las salidas del combinacional adoptarán cierto estado, como algunas de ellas se realimentan, las salidas del combinacional cambiarán nuevamente. Este proceso (llamado evolución automática del sistema) se repetirá hasta tanto el valor de las variables internas coincida con el anterior, este es el **estado estable**.

Si a las variables internas se las deja pasar de izquierda a derecha sólo en ciertos momentos, se obtiene un Sistema Secuencial Síncrono como se muestra en la figura 31.

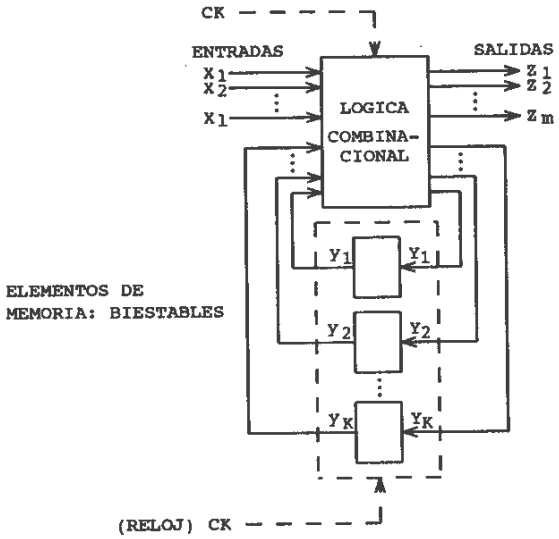


Figura 31 – Sistema Secuencial Síncrono

El diseño básico de estos sistemas consiste en plantear una tabla de verdad en la cual se tenga en cuenta el concepto **tiempo**. Ahora una función lógica no sólo depende de ciertas variables independientes sino que también depende de sí misma. Por ejemplo:

$p = f(a, b, c, ..., p, ...)$

Se observa que la función, indicada como p, aparece en ambos miembros de la expresión. Esto, para que no carezca de sentido, debe interpretarse de la siguiente manera:

$p_{t+1} = f_t\ (a, b, c, ..., p, ...)$

El subíndice ***t+1*** se interpreta como el valor que adoptará p para el conjunto de valores que tenían las variables de las cuales depende en el instante t.

A fin de aclarar los conceptos anteriores, sea resolver el siguiente problema.
Problema: Implementar un Sistema Digital para una alarma domiciliaria que posea:

- Dos variables de entrada:

Variable **S**: entrada de sensor, proviene por ejemplo del sensor de una puerta, etc.
Variable **R** entrada de inicialización.

- Una variable de salidas:

Variable **Q** salida a la sirena

El sistema deberá funcionar de la siguiente manera:

- Cuando la variable S tome el valor 1, se activará la salida (Q = 1)y si ya estaba activada permanecerá en esa condición. La salida quedará activada aún cuando la variable S pase a 0.
- Cuando la variable R toma el valor 1, se desactivará la salida (Q = 0) y si ya estaba desactivada permanecerá en esa condición. La salida quedará desactivada aún cuando la variable R pase a 0
- Nunca S y R podrán valer 1 simultáneamente.

Se propone la siguiente tabla de verdad con el concepto de tiempo ya explicado:

R	S	Qt	*Qt+1*
0	0	0	*0*
0	0	1	*1*
0	1	0	*1*
0	1	1	*1*
1	0	0	*0*
1	0	1	*0*
1	1	0	*X*
1	1	1	*X*

Figura 32

Las entradas al combinacional serán las tres variables de la izquierda, las X indican que nunca se produce esa combinación de las variables de entrada, por lo tanto no importa el valor que adopte Q t+1. Para implementar el circuito se realiza el mapa de Karnaught para la función. Figura 33a para la función en minterms y figura 33b para la función en maxterms:

$Q_{t+1} = \Sigma_3 (1, 2, 3, 6, 7)$ ó $Q_{t+1} = \Pi_3 (0, 1, 2, 3, 7)$

Las X corresponden a los minterms 6 y 7, y a los maxterms 0 y 1. Se ha decidido incluirlos en las expresión por cuanto podría obtenerse una minimización óptima.

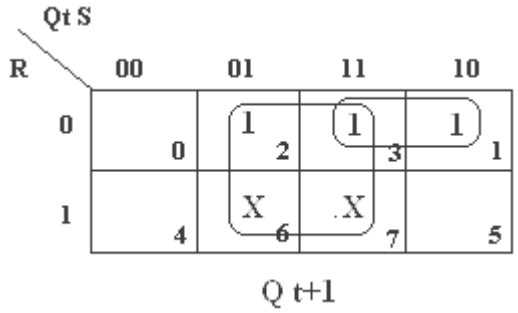


Figura 33-a

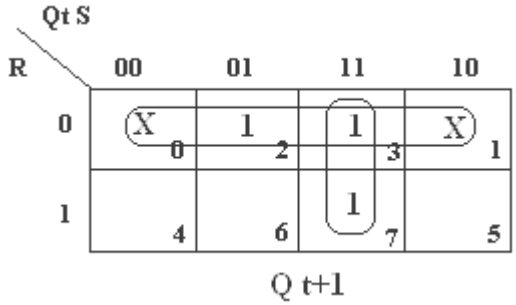


Figura 33-b

De la figura 33-a, resulta

$Qt+1 = S + Qt\ R' = (S + Qt\ R')'' = (S'\ (Qt\ R')')'$, cuyo circuito es el siguiente

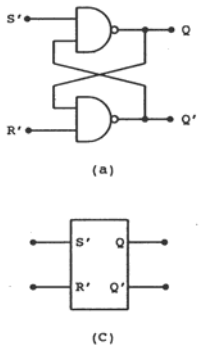


Figura 34 (a) Circuito – (c) Biestable SR (NAND)

De la figura 33-b, resulta

$Qt+1 = R'\ Qt + S = (R'\ Qt + S)'' = (R' + (Qt + S)')'$, cuyo circuito es el siguiente:

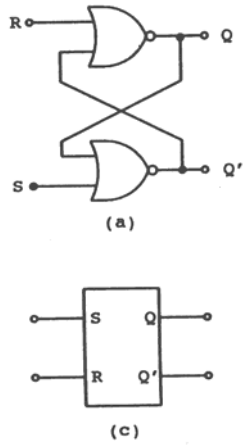


Figura 35 (a) Circuito – (c) Biestable SR (NOR)

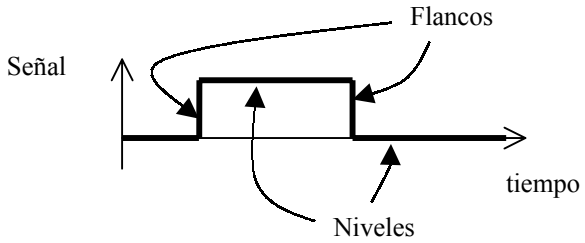
Dos conceptos útiles

I) Las compuertas lógicas reales se diferencian de las ideales en:

a) Poseen un tiempo de retardo, es decir: la señal lógica tarda un tiempo no nulo para atravesar la compuerta.
b) Disipan calor.

La característica a) es de especial importancia en los Secuenciales. Efectivamente, en la figura 30 aparecen unos elementos M necesarios para que el secuencial funcione. Si estos elementos no estuvieran, una misma línea lógica debería tener dos estados a la vez y esto no es posible. Sin embargo en los biestables de las figuras 34 o 35, estos elementos M no aparecen. La razón por la cual funcionan es que están construidos con compuertas reales y el retardo propio de las mismas cumple la función de los elementos M.

II) En una señal lógica se pueden indicar las siguientes partes:



Biestables

Como se vio en el problema anterior, los biestables son secuenciales que poseen dos estados estables, es decir que las variables internas pueden adoptar en este caso dos estados en los cuales permanecerán indefinidamente a menos que cambien las variables de entrada. Se trata entonces de los secuenciales más simples ya que poseen una sola variable interna. Los biestables representan los circuitos base para la construcción de secuenciales más complejos.

Se puede clasificar a los biestables de la siguiente manera:

Figura 37

- BIESTABLES
  - ASÍNCRONOS
  - SÍNCRONOS
    - ACTIVADOS POR NIVEL
    - MAESTRO - ESCLAVO
    - ACTIVADOS POR FLANCOS

Biestables asíncronos

Son aquellos en los cuales las entradas actúan directamente sobre el biestable. Son ejemplos de estos biestables los vistos en el problema de la alarma domiciliaria, figura 33 y figura 34 . Puede decirse que la tabla de verdad de la figura 32 es válida en todo momento.

Biestables Síncronos

Estos biestables cuentan con una entrada adicional: La entrada de sincronismo o reloj. De acuerdo a cómo actúa esta señal, los biestables síncronos se dividen en activados por nivel, maestro – esclavo y activados por flancos.:

Biestables Síncronos activados por nivel

Son aquellos biestables en los cuales la tabla de verdad es válida sólo en presencia de un nivel activo en la entrada de sincronismo. La figura 38 muestra un biestable SR síncrono por nivel

Se observa que parte de un RS asíncrono y se le agrega un circuito de disparo. La fig. 38 (a), muestra un SR activado con nivel 1, y la fig. 38 (b), un SR activado con nivel 0.

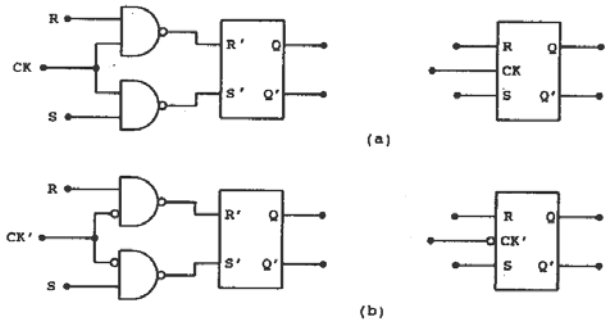


Figura 38

Biestables síncronos maestro – esclavo

Están formados por dos biestables activados por niivel. La fig. 39 muestra un biestable SR maestro esclavo.

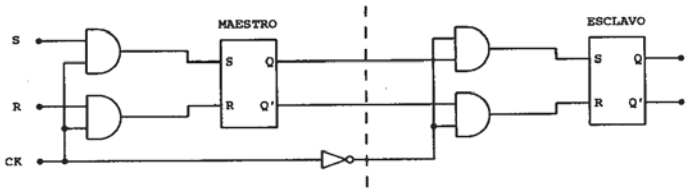


Figura 39

Se observa que mientras Ck = 1, se encuentra funcionando el primer biestable (maestro), en el momento que Ck = 0, la información del maestro pasa al esclavo. Este biestable actúa como si estuviera activado en el flanco de bajada de la señal de sincronismo, no obstante se diferencia de los activados por flancos en el hecho que las entradas actúan sobre el maestro durante el tiempo que Ck = 1.

Biestables activados por flancos

Es estos biestables las entradas actúan sólo en presencia de un flanco (de subida o bajada) en la entrada de sincronismo. La tabla de verdad será válida sólo en esos instantes. En la figura 40 se muestra un SR activado por flanco.

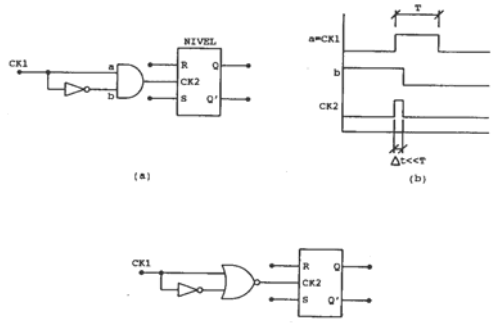


Figura 40

Se trata de un SR síncrono por nivel al cual se le agrega un circuito detecto de flancos.

En la figura 41, se muestra el símbolo utilizado para este tipo de biestable.

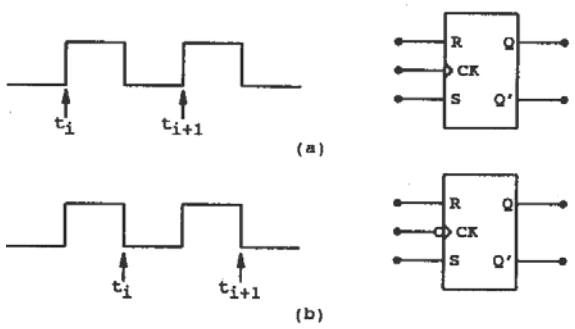


Figura 41

La fig. 41 (a), muestra un SR sincronizado por flanco de subida. La fig. 41 (b), muestra un SR sincronizado por flanco de bajada.

Todos los biestables SR tienen la misma tabla de verdad (figura 32), dependiendo de la clase de SR la tabla de verdad es válida siempre, durante el nivel activo o durante el flanco activo, según corresponda. Una forma reducida de indicar la tabla de verdad de un SR se indica en al figura 42.

R	S	*Qt+1*
0	0	*Qt*
0	1	*1*
1	0	*0*
1	1	*X*

Figura 42

Tipos de biestables

Existen otros tipos de biestables diferentes al SR. Ellos son el biestable JK, el biestable T. Y el biestable D. Estos no se encuentran disponibles en todas las clases (asíncronos, síncronos por nivel, etc.)

Las tablas de verdad son las indicadas en la figura 43 a, b y c

K	J	*Qt+1*
0	0	*Qt*
0	1	*1*
1	0	*0*
1	1	*Q't*

(a) Biestable JK

T	*Qt+1*
0	*Qt*
1	*Q't*

(b) Biestable T

D	*Qt+1*
0	*0*
1	*1*

(c) Biestable D

Figura 43 – JK, T y D

Biestable JK: En la Tabla 43 (a), para J = K = 1, $Q_{t+1} = Q'_t$, es decir la salida adopta el valor opuesto al anterior. Por esta razón sólo tienen aplicación práctica los biestables JK síncronos activados por flancos.
En la figura 44 se muestran algunos biestables JK.

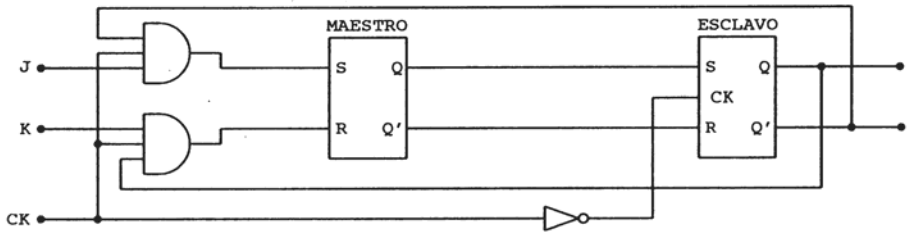


Figura 44 a – Biestable JK Maestro – Esclavo a partir de dos SR

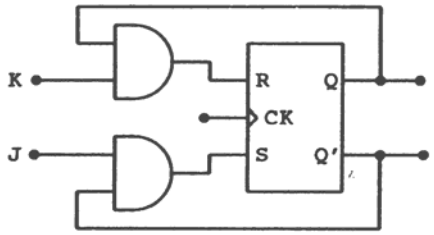


Figura 44 b – Biestable JK por flanco ascendente a partir de un SR

Biestables T: No están disponibles comercialmente. Se obtienen a partir de un biestable JK haciendo J=K=T. También pueden obtenerse a partir de un biestable D por flancos.

***Biestables D*:** En la figura 43 c se muestra la tabla de verdad de este biestable. Se concluye que carece de aplicación un biestable D asíncrono. Por lo tanto los biestables D se disponen comercialmente como síncronos, ya sea por nivel (D Latch), Maestro – Esclavo, o por flanco. Pueden obtenerse a partir de un SR síncrono, haciendo S = R' = D. También de un JK, haciendo J = K' = D. Ver figura 45 a y b.

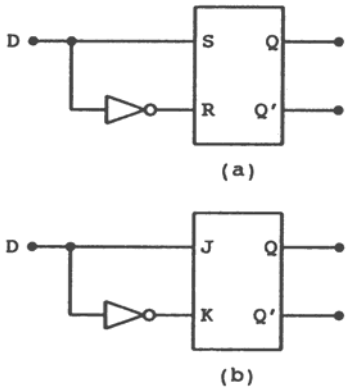


Figura 45

En cuanto al sincronismo, en la fig. 45 a, puede usarse un SR por nivel, maestro–esclavo o por flanco y resultare un D por nivel, maestro–esclavo o por flanco respectivamente.
En la fig. 45 b, se supone un JK maestro-esclavo o por flanco.

Aplicaciones de los biestables

Los biestables son secuenciales básicos capaces de memorizar un bit. Existen infinidad de aplicaciones, entre las principales se encuentran:

- Memorias, desarrolladas en la Unidad 4
- Registros
- Contadores

Registros

Es un secuencial síncrono que almacena varios bits de información. El formato de la información puede ser de dos tipos: serie (cuando los bits se transfieren uno después del otro por la misma línea) o paralelo (cuando se transfieren simultáneamente). Los registros pueden clasificarse de la siguiente manera

I) Registros de Desplazamiento

- Entrada serie, salida serie
- Entrada serie, salida paralela
- Entrada paralela, salida serie

II) Registros propiamente dichos (o sólo Registros)

- Entrada paralela, salida paralela

Registros de desplazamiento

Estos registros, si la cantidad de bits almacenados en grande (principalmente serie – serie), se incluyen en la Unidad 4 . Si se trata de algunos bits, ver las figuras 46 y 47.

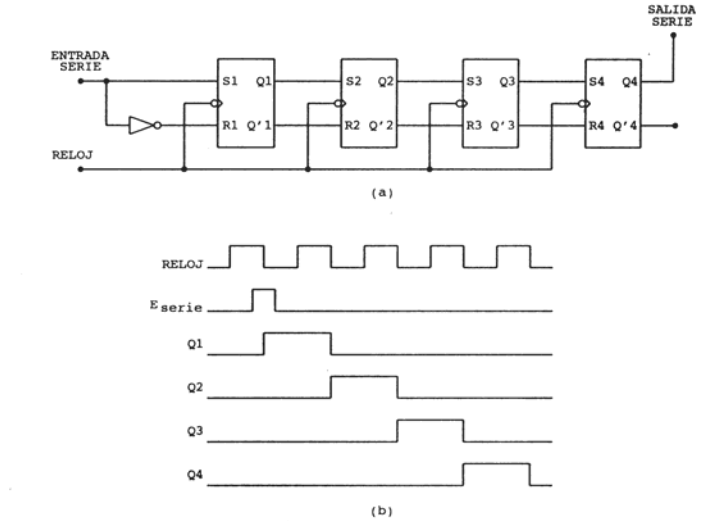


Figura 46 – Registro de desplazamiento de cuatro bits serie - serie

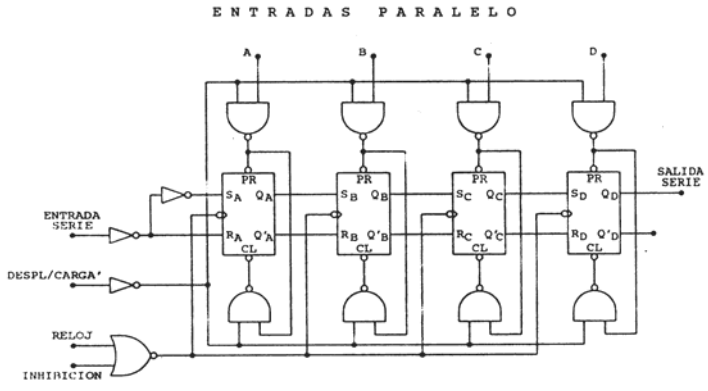


Figura 47 – Registro de desplazamiento paralelo – serie de cuatro bits

Registros propiamente dichos

Consisten en un conjunto de biestables sincronizados por nivel o por flancos cuyas entradas de sincronismo se encuentran unidas. Son de uso extensivo en cualquier sistema digital. En la figura 48 se

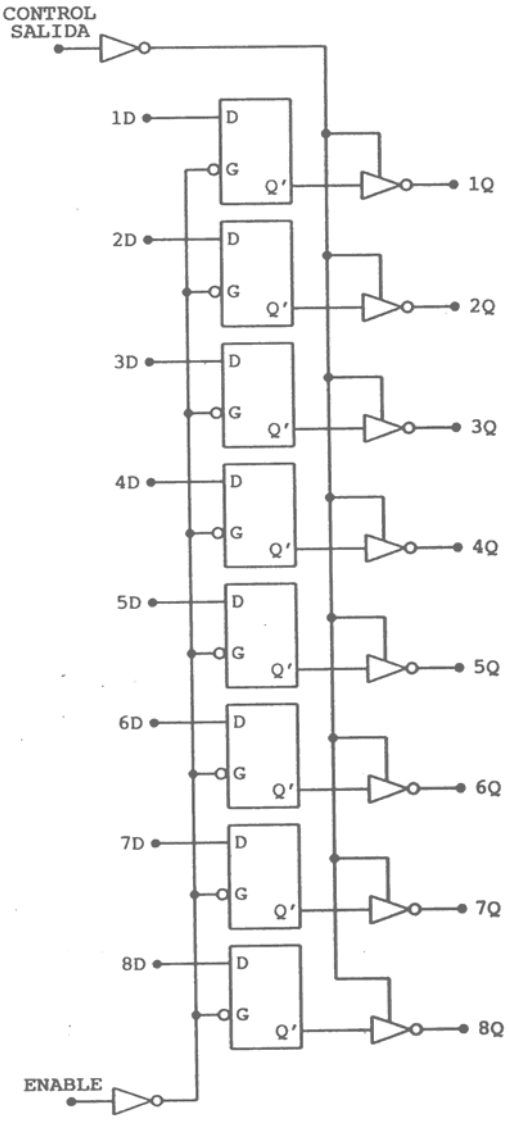


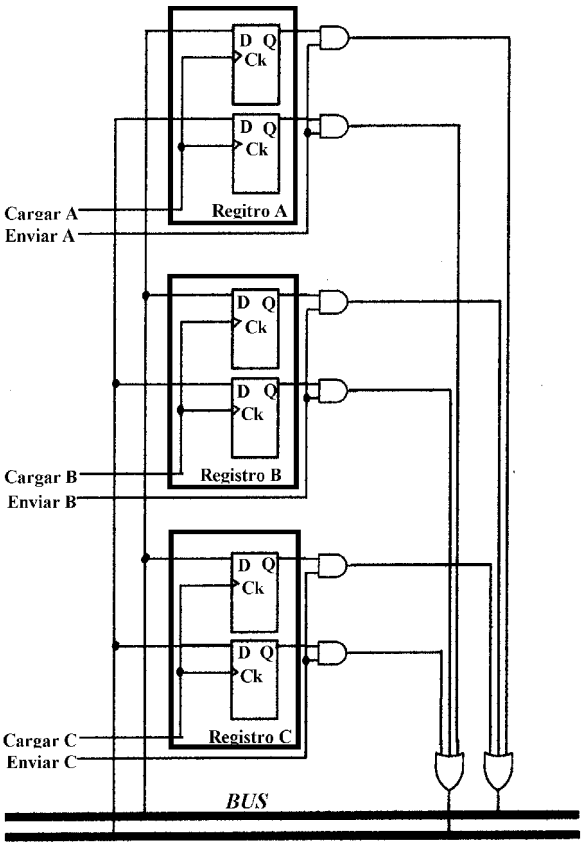
Figura 48 – Registro paralelo – paralelo de 8 bits

muestra un registro de 8 bits cuyas salidas están provistas de inversores tri-estado a fin de conectarse a un bus.

Transferencias entre registros

Buena parte de la actividad de un Sistema Digital es la transferencia de los contenidos entre distintos registros. En la Unidad 5 se presenta una computadora elemental y su funcionamiento se basa en la transferencia entre registros. Los Lenguajes de Programación de Hardware permiten diseñar sistemas digitales basándose en la transferencias entre registros. Es común ver estructuras en las cuales aparece un bus (conjunto de líneas lógicas que transporta información) del cual se encuentran "colgados" registros. Estos registros pueden actuar como elementos de interconexión entre el bus y distintas unidades funcionales, o bien ser registros de almacenamiento temporario de información exclusivamente.

En la figura 49a se presenta una forma de interconexión entre registros llamada BUS COMÚN, se trata de tres registros de dos bits cada uno. Las entradas de algunos de estos registros podrían estar conectadas a una Unidad Funcional (ALU por ejemplo) funcionando en este caso como registro de salida de la misma; o bien las salidas de alguno de los registros podrían estar conectadas a otra Unidad Funcional (Unidad de Memoria por ejemplo) funcionando en este caso como registro de entrada a la misma. En la figura 49b se indica cómo deberían ser las señales de control para llevar a cabo la transferencia entre registros. Estas señales de control son generadas en general por la Unidad de Control del Bus; y, de su eficiencia, depende en gran medida la perfomance (velocidad de procesamiento) del Sistema Digital.



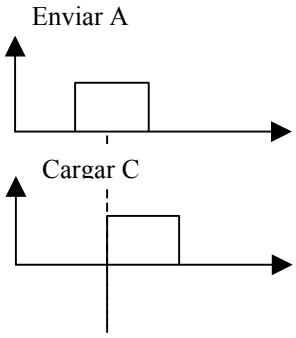


Figura 49b

Figura 49a

Otra forma de construir un sistema de interconexión entre registros es usando registros con salida tri-estado (como el indicado en la figura 48). Un ejemplo puede apreciarse en la figura 49c.

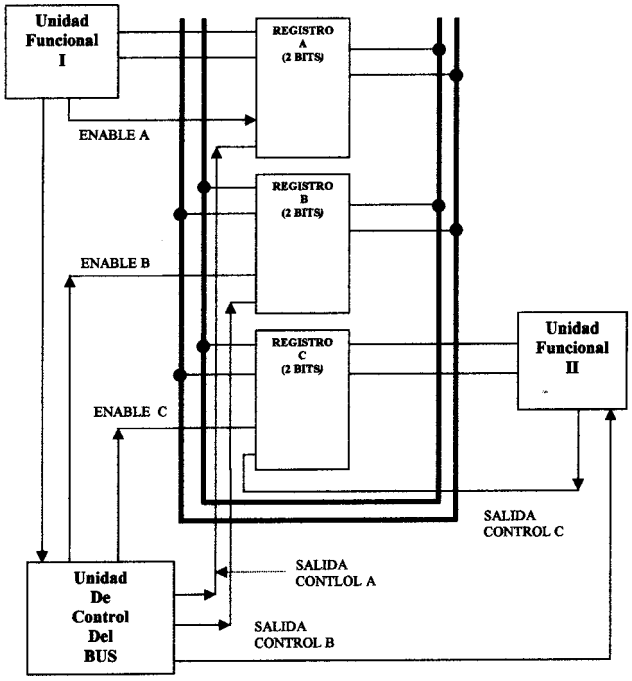


Figura 49c

En la figura se aprecian tres registros con salida tri-estado. Cada registro, además de dos entradas y dos salidas de información, poseen dos entradas de control: Enable y Control de salida. Estas son manejadas por la Unidad de Control del Bus y por las Unidades Funcionales. Supóngase por ejemplo que la Unidad Funcional I ya ha procesado una información y en necesario transferirla al Registro C para que,

finalmente, sea procesada por la Unidad Funcional II. Para este ejemplo la Unidad de Control del Bus deberá realizar lo siguiente:

- Recibir desde la Unidad Funcional I la señal de listo (La Unidad Funcional I una vez que dispone de la información, la carga en el Registro A con Enable A y luego envía la señal listo a la Unidad de Control del Bus).
- Activar la señal: Salida Control A, con lo cual vuelca al bus el contenido del Registro A.
- Activar la señal: Enable C, a fin de cargar el Registro C con la información presente en el Bus.
- Desactivar las señales: Salida Control A y Enable C.
- Indicar a la Unidad Funcional II que ,en el Registro C, existe información a procesar.

La Unidad Funcional II, al recibir la señal de la Unidad de Control del Bus, lee el contenido del Registro C mediante la señal Salida de Control C.

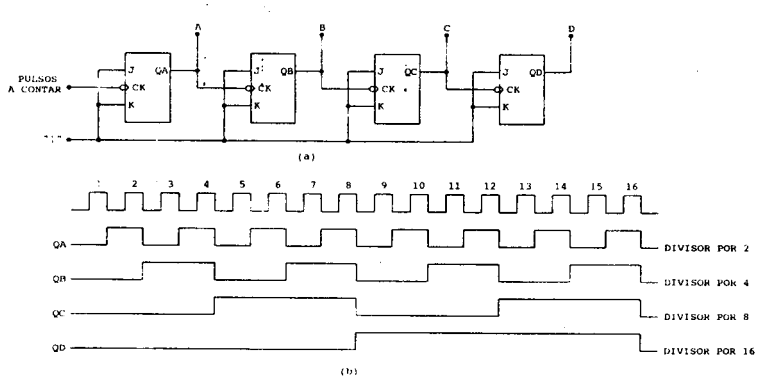
De lo visto se puede intuir la importancia de Bus en los Sistemas Digitales. En la Unidad 5 se desarrolla un Sistema Digital basado en la estructura de Von Newman en el que considera al Bus como una unidad en sí mismo.

Contadores

Es un sistema secuencial formado por biestables y lógica combinacional, capaz de almacenar en binario u otro código, la cantidad de impulsos recibidos por su entrada de cuenta. Puede aplicarse como divisor de frecuencia, control de tiempos, generador de direcciones en sistemas de memoria, secuenciador en unidades de control, etc.

Contadores Asíncronos

Son secuenciales síncronos formados por un conjunto de biestables síncronos por flancos. Su denominación de asíncrono no se refiere al tipo de secuencial sino al hecho que las entradas de sincronismo de sus biestables no están unidas entre sí. Por lo general la salida de un biestable sirve como entrada de sincronismo del siguiente. En la figura 50 se muestra un contador binario de 4 bits asíncrono, obsérvese que las salidas de los biestables se conectan a las entradas de sincronismo del siguiente. También puede verse el diagrama de tiempo



de este contador.

Figura 50 – Contador asíncrono de 4 bits. (a) Diagrama circuital (b) Diagrama de tiempo

Contadores Síncronos

Son similares al los anteriores sólo que comparten la misma señal de reloj. Son más rápidos y complejos que los asíncronos. En la figura 51 se muestra un contador binario natural síncrono de 4 bits. Nótese que, a diferencia de la figura 50, este contador tiene todas las entradas de sincronismo de los biestables unidas. Además es más complejo puesto que tiene más compuertas que el anterior.

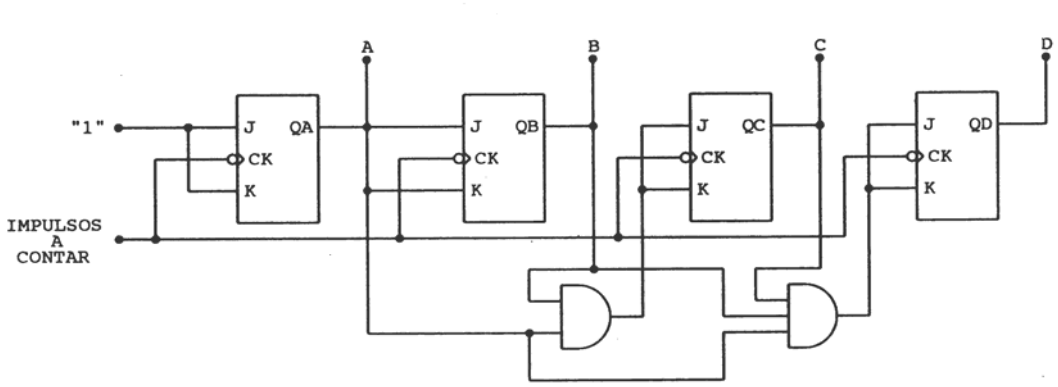


Figura 51 – Contador binario natural de 4 bits síncrono.

El diseño de contadores se realiza planteando una tabla de verdad temporal y luego de obtener las funciones correspondientes, se minimizan teniendo en cuenta el biestable elegido. Los biestables utilizados en los contadores como así también en los registros de desplazamiento, son biestables síncrono maestro – esclavo o activados por flancos.