UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

"NOCIONES BASICAS SOBRE EL DISEÑO DE CIRCUITOS DIGITALES"

T E S I S

PARA OBTENER EL TITULO DE

F I S I C O

P R E S E N T A :

LUZ MARIA GONZALEZ OSORIO

MEXICO 1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

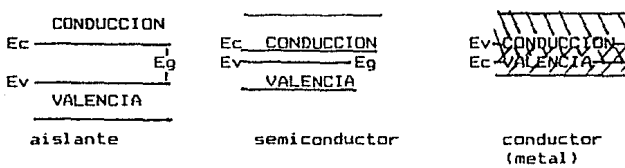
INTRODUCCION

interna de este tipo de materiales.

Estructura atómica de un semiconductor.- Antes que nada, recordaremos que la estructura atòmica de todo material se puede describir acorde a sus valores de energía, asociados cada uno de ellos a un electròn orbital. Entre mayor sea la distancia de un electròn el núcleo atòmico, mayor será la energía que èste posea, de modo que cualquier electròn libre poseerà siempre una energía mayor a la de cualquier electròn ligado.

A los electrones de la òrbita màs externa de un átomo se les denomina de valencia, mismo que serán los encargados de formar el enlace con otros àtomos, definiendo además las propiedades físicas y químicas del elemento así formado.

Cuando se forma una red de àtomos de un mismo material, se tiene que lo que eran niveles discretos de energía se convierten ahora en bandas de energía. Dos de estas bandas son de particular interès: la de valencia y la de conducción, ya que la cantidad de energía que se requiere para pasar un electròn de una banda a otra, será la característica definitoria de las propiedades elèctricas de un material. La siguiente figura muestra còmo es esto según el tipo de material de que se trate: conductor, semiconductor o aislante.



aislante semiconductor conductor (metal)

En la figura anterior, Ev es la màxima energía que pueda tener un electròn de valencia y Ec la energía mínima de conducción. Para pasar de la banda de valencia a la banda de conducción es necesario superar la banda prohibida de energía (Eg); esto, en el caso de un aislante es prácticamente imposible, ya que Eg resulta ser muy grande: en contraposición, en un conductor ambas bandas se traslapan y no es necesario proporcionar a los electrones energía alguna para que pasen a la banda de conducción, como sería el caso de un semiconductor, en el cual la banda prohibida no es tan ancha como la de un aislante, y basta con proporcionar por vía externa (luz, calor, etc) una cierta cantidad de energía para poder pasar de una banda a la otra.

Un àtomo de silicio posee 14 electrones orbitales, mientras que el germanio tiene 32; en ambos casos existen 4 electrones de valencia. El potencial que se necesita para extraer alguno

SEMICONDUCTORES Y CIRCUITOS INTEGRADOS.

El actual desarrollo en el campo de la microelectrònica se debe principalmente a la evoluciòn de los dispositivos semiconductores; es por todos conocido el hecho de que la mayoria de los equipos electrònicos usados, tanto en la investigaciòn como por la industria fabricante de relojes, calculadoras, etc., emplean extensamente elementos tales como diodos, transistores, circuitos integrados de diversos tipos, etc. A partir de este hecho se hace patente el interés que existe en diversos centros de investigaciòn por acelerar el desarrollo en este campo, el de los semiconductores. Sin embargo, y a pesar de que se ha difundido ampliamente el uso de los CI, no es mucha la gente que comprende las bases sobre las cuales se ha construido esta tecnologia. Es por ello que en las pàginas que siguen se expondràn, de la manera màs clara y breve, posible los conceptos bàsicos sobre semiconductores.

En general, se denomina semiconductor a todo aquel material cuyo comportamiento elèctrico es intermedio entre el de un aislante y el de un conductor. Ya a principios de siglo era usual el empleo de una gran variedad de cristales semiconductores, aunque la limitada tècnica de obtenciòn de los mismos impidiò su desarrollo.

En principio se creia que los materiales elèctricos se dividian en dos ùnicos grupos: aislantes y conductores. Esta situaciòn cambiò dràsticamente a principios del siglo pasado, cuando se observò que al hacer pasar una corriente elèctrica a travès de una barra de sulfuro de plata, ésta presenta una ligera conductividad, que aumenta al incrementarse la temperatura del material, con lo cual se llega a la conclusiòn de que deberia existir un grupo de materiales con caracterìsticas similares, caracterìsticas que dan nombre al grupo: semiconductores.

Ahora bien, la verdadera "explosiòn" de los semiconductores comienza a partir de la creaciòn del transistor en 1947.

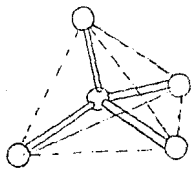
De entre los primeros semiconductores que se emplearon se muestran el silicio y el germanio, mismos que en estado puro y a bajas temperaturas se consideran aislantes, pero que al ser contaminados alteran de manera significativa su conductividad.

Otros materiales usados fueron el sulfuro de zinc (ZnS), el fosfuro de galio (GaP), el arseniuro de galio (GaAs), etc.

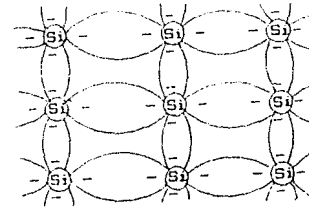
En la actualidad el que mayormente se emplea es el silicio, y en memor grado el germanio, siendo èste ùltimo extremadamente sensible a variaciones de temperatura.

Asì pues, como ya se habrà podido intuir, el comportamiento elèctrico de un semiconductor es diferente al de un conductor o un aislante. Con el objeto de comprender el por què de esta diferencia, a continuaciòn trataremos el tema de la estructura

de estos electrones es menor que el necesario para cualesquiera otro del conjunto. En la siguiente figura se puede ver, para el caso del silicio, còmo estos cuatro electrones establecen un enlace covalente con cuatro àtomos adyacentes; dada la dificultad que representa tratar de dibujar la red de estructuras tetraèdricas tridimensional que así se forma, se suele "aplanar" el diagrama y representar el enlace como un retículo cuadrado.



Estructura de un cristal de Si

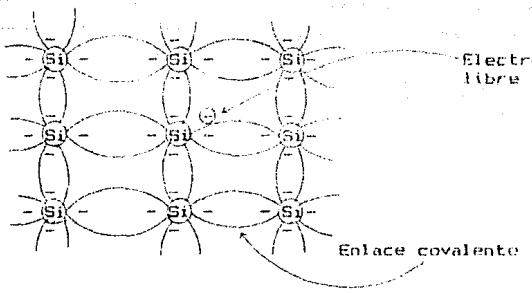


Enlace covalente del Si

Es precisamente este enlace covalente de los electrones de valencia el que se tiene que romper para que adquieran el estado libre, propio de la conducciòn. En principio, esto sucede en un porcentaje menor a partir de causas naturales, tales como la energía luminica y tèrmica de medio ambiente; en realidad a temperatura ambiente existe un promedio de $1.5x10^{10}$ portadores libres/cm^3 en el silicio, en tanto que hay 2.5 x 10^{13} p lib/cm^3 en el germanio, razòn por la cual presenta la ya mencionada sensibilidad a los cambios de temperatura.

Como ya antes se mencionò, el añadir impureza a un material semiconductor altera su conductividad. A los semiconductores en estado puro se les denomina intrìnsecos, y a los contaminados extrìnsecos; estos ùltimos se dividen en dos tipos, N y P, segùn la clase de impureza que se les añade.

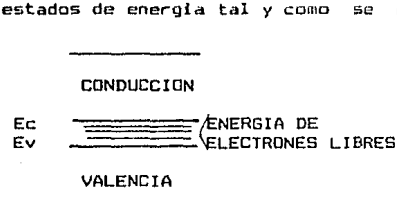
Semiconductor tipo N.- Tomemos como base para la explicaciòn el caso del silicio. Este tipo de semiconductor se forma añadiendo elementos cuyos àtomos sean pentavalentes, esto es, que posean 5 electrones de valencia, tales como el antimonio, el arsènico, etc. La siguiente figura muestra lo que sucede a nivel àtomico cuando al silicio (tetravalente) se le agrega un àtomo de antimonio (Sb).



Semiconductor tipo N

El silicio posee 4 electrones de valencia, en tanto que el Sb tiene 5, lo caul se traduce en la formaciòn de 4 enlaces covalentes del silicio con el antimonio, y un electròn relativamente libre de moverse dentro del material. Como se dice que la impureza ha donado un electròn a la estructura se le conoce como àtomo donante.

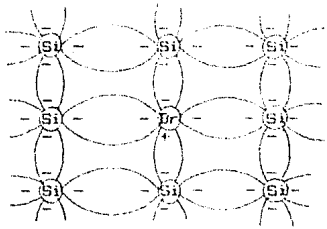
Nòtose bien que, aunque existen electrones libres, el material sigue siendo elèctricamente neutro; sin embargo, debido a que estos electrones "libres" poseen una energia mayor que los de valencia en el semiconductor, se establece un cambio en los estados de energia tal y como se muestra en seguida:



Niveles de energia en un semiconductor tipo N

Existe ahora un nivel de energia intermedio a las bandas de valencia y conducciòn, correspondientes a los àtomos donantes, que acorta notablemente la distancia a la banda de conducciòn (Eg es menor), tanto que, a temperatura ambiente, un material contaminado posee una gran cantidad de electrones en la banda de conducciòn.

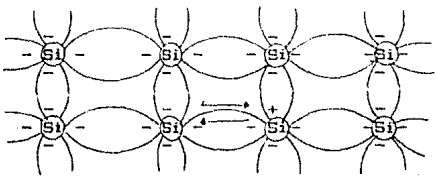
Semiconductores tipo P.- Se crean agregando átomos trivalentes como el boro, el galio, etc. La siguiente figura nos muestra cómo una impureza de boro afecta a una estructura de silicio.



Material semiconductor tipo P

En este caso sòlo se pueden formar tres enlaces covalentes, quedando un espacio vacío, que suele denominarse "hueco" y que se simboliza con un círculo o un signo "+". Como se dice que este hueco aceptará de imediato cualquier electròn circundante, se denomina a la impureza aceptora. En este caso el material también sigue siendo neutro y la conducciòn se lleva a cabo de la siguiente manera: si por alguna razòn un electròn llega a romper el enlace covalente que lo liga, pasará a ocupar presto el primer hueco que se encuentre, dejado a su vez un espacio vacío (hueco) en el lugar que ocupaba. De esta manera se tendrá que existe un flujo de electrones en un sentido, y de huecos hacia el otro.

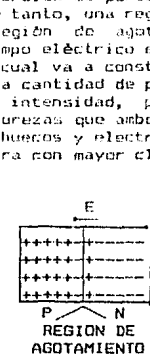
Convencionalmente se toma como dirección del flujo de una corriente elèctrica la que va de positivo a negativo, por lo que se dirá que en un semiconductor tipo P, la conducciòn se hace en el sentido del flujo de los huecos. Esto puede verse en la siguiente figura:



Flujo de corriente en un semiconductor tipo P

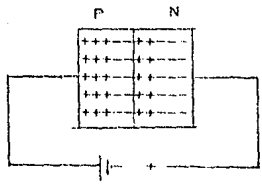
Unión PN.- Se establece cuando un material tipo P entra en contacto estrecho con una tipo N, y constituye la base del funcionamiento de una gran mayoría de los dispositivos semiconductores.

Veamos qué sucede cuando se establece esta unión. Se tiene que el tipo P realiza la conducción a través de huecos (portadores mayoritarios) y el N por electrones (portadores minoritarios); cuando ambos materiales entran en contacto, los electrones libres de N serán atraídos hacia los huecos de P, generando una pérdida de portadores en la región cercana a la unión, y por lo tanto, una región provista de iones positivos y negativos (región de agotamiento). Debido a esto se establece un campo eléctrico en la región de la unión (carga espacial), el cual va a constituir una barrera de potencial limitante de la cantidad de portadores que llegan de uno y otro lado, y su intensidad, por tanto, va a depender de la cantidad de impurezas que ambos materiales contengan (mayor o menor número de huecos y electrones existentes). La siguiente figura nos muestra con mayor claridad lo que se ha expuesto:



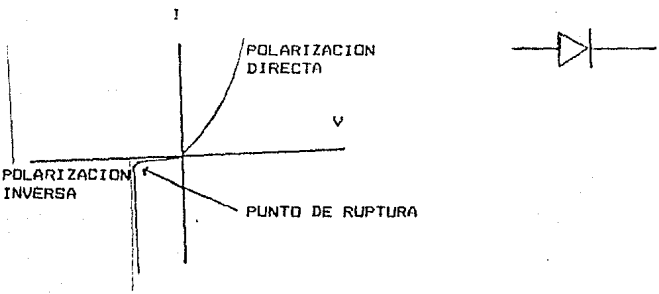
Unión PN sin potencial externo

Supongamos ahora que aplicamos una diferencia de potencial a esta unión. Puede suceder dos cosas dependiendo del sentido de la polarización: si se polariza en sentido directo, es decir, que el negativo se asocia al material tipo N, se tendrá que se estará proveyendo de electrones a esta parte, en tanto que a P se le suministran huecos, con lo que se tendrá que la barrera de potencial de la unión disminuirá su magnitud, facilitando el paso de electrones hacia P y de huecos hacia N, de tal manera que la unión presenta una resistenciamuy baja al flujo de corriente eléctrica; si por el contrario, se polariza la unión en sentido inverso (el negativo a P), se estarán extrayendo electrones de N y huecos de P. Consecuentemente, la magnitud de la barrera de potencial aumenta considerablemente, haciendo prácticamente imposible el flujo de portadores, ya sea de huecos hacia N o de electrones hacia p, y por lo tanto la unión presenta una impedancia (resistencia) muy alta al flujo de corriente electrónica.



Unión PN polarizada inversamente

Lo que acabamos de describir es lo que precisamente constituye la estructura bàsica del funcionamiento de un diodo como rectificador de corriente. La siguiente figura muestra el simbolo gràfico usado para representar al diodo y la gràfica que describe el comportamiento tipico del mismo:



En polarización directa la corriente crece de manera exponencial, en tanto que en inversa es casi nula (corriente de fuga), y si el voltaje crece demasiado se puede llegar a la ruptura de la unión, punto a partir del cual la corriente crece abruptamente (cabe mencionar que actualmente existen ya diodos tipo Zener que están especialmente diseñados para funcionar en polarización inversa).

BOSQUEJO HISTORICO

Es a partir de esta teoría que se introduce el transistor, conduciendo al surgimiento de lo que se designa como "revolución microelectrónica", misma que ha posibilitado el desarrollo de ciertos elementos, llamados circuitos integrados -analógicos y digitales- que poseen cada vez menores dimensiones y realizan funciones cada vez más complejas en un tiempo mínimo.

De alguna manera, una gran mayoría de los avances tecnológicos de nuestra era han dependido en gran medida de la microelectrónica: se han diseñado dispositivos de control y medición adaptables a condiciones ambientales variables, lo cual en su momento ha permitido que importantes investigaciones espaciales se llevaran a cabo con gran éxito; militarmente hablando, se han diseñado equipos de gran complejidad que han dado lugar a la creación de eficaces sistemas de defensa; la capacidad de procedimiento de información de una computadora se ha visto incrementada gracias al uso de los circuitos integrados (CI), lo cual a su vez ha tenido como consecuencia la reducción de las dimensiones de la misma.

Ahora bien, enunciar todas las consecuecias prácticas derivadas el uso de los CI sería muy largo, y además no forma parte de nuestro objetivo. En realidad, la finalidad de esta sección es mencionar algunos de los hechos clave que, de manera general, dieron lugar a la creación de los CI, haciendo mención especial de los digitales, tema central del presente texto.

Antes, sin embargo, es bueno hacer aquí algunas aclaraciones que se consideran importantes. Si bien suele hacerse unívocamente la relación computadoras-CI digitales, esto no es correcto, ya que aunque una computadora está constituida casi en su totalidad por elementos digitales, éstos no limitan su campo de acción a la informática; así mismo, y por confusiones similares, suele también creerse que en general los CI surgen exclusivamente como una necesidad para el desarrollo de las computadoras, lo cual, como veremos más adelante, no es totalmente cierto.

Así pues, para dar una idea clara del impacto que crea la intrusión de los CI en todos los campos donde se les aplica, es menester establecer lo que a grandes razgos define el escenario científico y tecnológico antes y después del momento de su surgimiento.

Por todos es sabido que los conocimientos que se tienen sobre los avences de la humanidad se remotan a algunos milenios antes de nuestra era; se ha podido establecer con certeza que desde sus inicios el hombre se ha procurado los elementos necesarios que le permitan dominar su medio, lo cual

Así pues, entre los primeros circuitos digitales que surgen, encontramos a los llamados compuertas simples, que pueden realizar operaciones lògicas sencillas como son la AND, OR, NOT, NOR, etc. Posteriormente, se fabrican circuitos que engloban en uno solo muchas de estas operaciones; se construyen contadores, decodificadores, etc., llegàndose a elaborar elementos tan complejos como son los microprocesadores actuales, que a pesar de sus mínimas proporciones, poseen la capacidad de efectuar una gran cantidad de funciones lògico-artimèticas.

La tabla 2 muestra de manera muy general y cronològica las diversas etapas del desarrollo de los CI, desde el tubo de vacío, primer antecedente, hasta las tècnicas actualmente utilizadas.

En la tabla 2 se muestra cierta clasificaciòn del tipo de tecnología (MSI, LSI, - etc.) que vá en funciòn de la complejidad de los circuitos, complejidad que depende de la cantidad de "funciones" que se integran en un ùnico circuito. Dicha clasificaciòn es la que se muestra a continuaciòn:

SSI- Small Scale Integration. Baja Escala de Integraciòn; de 1 a 10 componentes por CI.

MSI- Medium Scale Integration. Media Escala de Integraciòn; de 10 a 100 componentes por CI.

LSI- Large Scale Integration. Alta Escala de Integraciòn; de 100 a 100000 componentes por CI.

VLSI- Very Large Scale Integration. Muy Alta Escala de Integraciòn; màs de 100000 componentes por CI.

De esta menera, la complejidad (entiéndase integraciòn) es cada vez mayor, al grado de que se ha enunciado ya una sentencia que, llamada Ley de Moore, afirma que la complejidad de los CI se duplica cada año (esta complejidad, traducida en miniaturizaciòn, parece estar condicionada en mayor medida por factores econòmicos que físicos, por lo menos hoy en día).

lo llevò a la creaciòn de una gran variedad de instrumentos, tanto para facilitar tareas mecànicas, como para realizar con mayor eficacia cálculos astronòmicos; inventò sistemas numèricos y mètodos de cálculo, y con el tiempo, eficientes instrumentos de cálculo, tales como el àbaco o la regla de cálculo, mismos que actualmente han sido reemplazados por las calculadoras y/o computadoras, cuyos antecedentes màs directos se encuentran en la màquina calculadora diseñada por Pascal en el siglo 17; poco despuès, en ese mismo siglo, Leibnitz introduce el uso del sistema binario para las màquinas calculadoras.

Durante los siguientes siglos se realizan grandes avances en los campos cientifico y tecnològico; durante la època comprendida entre los años 1904 y 1947, el tubo de vacio, inventado por J.A. Fleming, pasa a ser el dispositivo electrònico de mayor interès, comenzando a ser usado en diversos campos, entre los cuales se encuentra el de la computaciòn (se construye la computadora ENIAC en 1946 en base a esta tecnologia). No obstante, durante la 2ª Guerra Mundial, se hacen evidentes las limitaciones del tubo de vacio: su tamaño, fragilidad, gran disipaciòn de potencia, etc., volvièndolo inadecuado para el equipo de radar y comunicaciones que en aquella època se estaba necesitando; se deseaban equipos portàtiles y autònomos, por lo que debia llegarse a la fabricaciòn de elementos de menores dimensiones y mayor eficiencia (menor consumo de energia, menor disipaciòn de potencia, mayor velocidad de operaciòn, etc.).

Las investigaciones sobre materiales semiconductores señalan entonces el camino a seguir. Los primeros resultados se obtienen al ser inventado en 1947 el transistor por W.H. Brattain y J. Bardien. Poco despuès se llega el diodo transistor y, en base a esto se logra a principios de los 60's la fabricaciòn de los primeros circuitos integrados.

Ahora bien, cuando se inventa el transistor, èste, habiendo probado su eficacia y mayores ventajas que el tubo de vacio, es inmediatamente adoptado para la fabricaciòn de màquinas calculadoras, optimizando los procesos que èstas realizan. En ese momento, la informàtica acapara la tecnologia de los CI cuando èsta surge al mercado, beneficiando asi su propio desarrollo (por ello, erròneamente se considera a la computadora como cuna de la microelectrònica). Dicho sea de otro modo, los ingenieros en computaciòn han ido adaptando los progresos de la microelectrònica a sus propios fines, con lo cual han logrado desarrollar diversos modelos de computadoras que poseen mínimas dimensiones y pueden ejecutar millones de operaciones por segundo, haciendo de ellas una herramienta bàsica en cualquier campo del conocimiento, dadas las màltiples capacidades con que han sido dotadas.

Ahora bien, hasta aquí hemos descrito de manera muy general cómo y bajo qué condiciones surgen los CI en el mundo de la electrónica; sin embargo, quisiéramos acentuar el hecho de que el mayor impulso lo recibieron en una época crítica de la Historia, es decir, durante la 2a Guerra Mundial, ya que, como dice Norbert Wiener 1, "una de las pocas ventajas del gran conflicto fue el rápido desarrollo de las invenciones, estimuladas por la necesidad y la disponibilidad de dinero".

En la actualidad, la tecnología de los CI, y en especial la de circuitos digitales, se ha ido desarrollando con extrema rapidez, llegando a infiltrarse, como en un principio se mencionó, en muchos y muy variados campos, tales como la industria (automatización de procesos industriales), la instrumentación (indicadores digitales de temperatura, presión, etc.), llegando inclusive a nuestra propia casa; basta observar los diversos indicadores digitales que poseen los aparatos electrodomésticos, o la gran variedad de relojes digitales de uso personal que actualmente se fabrica. Como dice el ya citado Wiener①: "...cuando se inventa algo, pasa mucho tiempo antes de comprender todas sus consecuencias". De cualquier modo que esto sea, "sabemos que nuestra época se caracteriza por un notable desarrollo de la ciencia; depende de nosotros utilizarla para ampliar y liberar nuestra cultura"②

---

① Norbet Wiener. Cibernética y Sociedad. Conacyt, 1981, p 128 y 129.

② Jacob Bronowski. El Sentido Común de la Ciencia, Península, 1978, p 161.

TABLA 2

FECHA DE APARICION	ELEMENTO
1904	Tubo de vacio
1947	1er transistor
1950	Transistor de juntura
1953	Tecnología FET
1958	Transistor planar
1959-1964	Compuertas simples
	Resistencia-transistor
	RTL
	MOSFET
	Tecnología SSI
1964-1969	Flip-Flop's
	Transistores DTL
	Elementos de lógica doble os 1 microprocesadores
	Tecnología MSI
1969-1974	Memorias de 64 KB
	Microprocesadores de 8 bits
	Tecnología LSI
1974...	Microprocesadores de 32 bits
	Memorias de gran
	capacidad (128 KB)
	Tecnología VLSI
	Arreglos lógicos programables

C A P I T U L O I

SISTEMAS DIGITALES

SISTEMAS DIGITALES.

En la vida diaria se observa que una gran variedad de fenómenos ocurren de manera aparentemente continua, es decir, a pesar de que las variaciones en el medio ambiente y en nosotros mismos consisten de un número infinito de pequeñísimos cambios que se suceden unos a otros en el tiempo, no somos capaces de percibirlos como tales, por el contrario, los percibimos como continuos. Es por esto que parece lógico que los instrumentos de medición trabajen con variables continuas, como sería el caso del termómetro de mercurio usual que mide cualquier temperatura dentro de un rango definido de operación, lo cual por ejemplo, permite medir las variaciones de temperatura en una habitación durante el día. Esta variación continua y puede ser representada en una gráfica como la de la figura 1.1.

De igual forma, variables tales como la presión o el volumen de un gas, intensidad de luz, etc., se comportan también como variables continuas.

Ahora bien, a todos los procesos que se realizan en forma continua se les suele denominar <u>analógicos</u> y, por consiguiente, se dice que una variable analógica es aquélla que puede

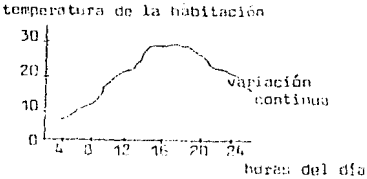


FIGURA 1.1. Gráfica de temperatura contra tiempo.

tomar cualquier valor posible dentro de un cierto intervalo (la temperatura en el caso del ejemplo de la figura 1 constituye una variable analógica).

Ejemplo común de un dispositivo analógico es el multímetro, que proporciona un rango continuo de medición de variables tales como voltaje, corriente, resistencia, etc.

Por otro lado, existe también un tipo de procesos -no analógicos- a los que se les denomina <u>digitales</u>, en los cuales, el cambio o transformación se realiza en pasos, es decir, de manera tal que es posible diferenciar tajantemente un estado del anterior, como ocurre, por ejemplo, con una lámpara, la cual puede estar prendida o apagada, pero no en algún estado intermedio entre estos dos (medio-encendida o

medio-apagada).

De esta forma, las variables que intervienen en un proceso discreto están restringidas a tomar sólo ciertos valores, contrario a lo que ocurre con las variables que se manejan en los procesos analógicos o continuos.

Para ejemplificar y aclarar lo anteriormente dicho, observe detenidamente las gráficas de las figuras 1.2.

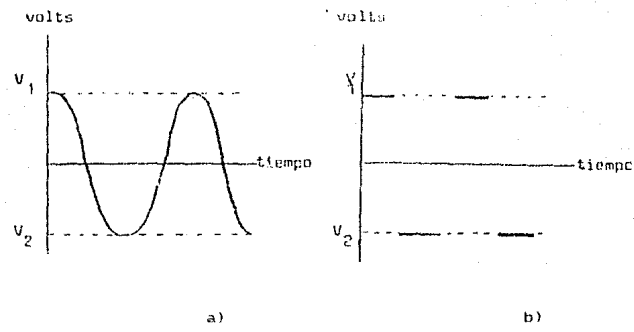


FIGURA 1.2 a) Tipos de señal Analógica (continua), b) Digital discreta.

Si se observa con cuidado cada una de las gráficas quedará claro que, mientras la variable analógica o continua puede tomar cualquier valor comprendido entre V1 y V2, la variable digital sólo puede tomar uno de los dos valores, V1 o V2.

Ahora bien, tanto las variables continuas como las discretas son manipulables por dispositivos electrónicos, pero dada la gran diferencia que existe entre unas y otras se han creado dos ramas de la electrónica para manejarlas por separado: la electrónica analógica y la electrónica digital. Esta última se aplica al diseño de sistemas que transmiten y/o procesan información representada como cantidades físicas discretas, a los que se les llama "sistemas digitales".

El número de posibles valores que puede tomar una variable discreta en un dispositivo electrónico puede fijarse de manera arbitraria, es decir, se podría asignar 2,5,7,14 ó más valores distintos, que corresponderían a diversos niveles de voltaje (véase la figura 1.2 b); sin embargo, hoy en día, la

tecnología digital se construye de manera que opere con sistemas y códigos binarios, es decir, manejando únicamente dos valores de voltaje, los cuales deben ser fijos y distintos entre sí. Esto no es casual, ya que la dificultad que presenta construir un circuito es mayor cuanto mayor es la cantidad de niveles de voltaje que se tengan que manipular, además, cualquier dispositivo electrónico requiere de una gran cantidad de elementos; si éstos sólo pueden estar en uno de los dos estados permitidos, pequeñas variaciones de los mismos no alterarán de modo significativo el funcionamiento del dispositivo, lo que se traduce en mayor confiabilidad para el diseño. En la tabla 1.1 se presenta una lista de los nombres más usuales que reciben los estados que ocupa una variable binaria, siendo los tres primeros los más empleados.

UN ESTADO DE LA VARIABLE BINARIA	OTRO ESTADO DE LA VARIABLE BINARIA
Verdadero	Falso
1	0
Alto	Bajo
Pulso	No-Pulso
Excitado	No-Excitado
Cerrado	Abierto

Tabla 1.1 Nombres usuales de los estados binarios que ocupa una variable binaria

De acuerdo con lo anterior, resulta lógico pensar que un sistema numérico binario va a permitir manejar eficientemente la información digital, ya que, al igual que en el sistema decimal, se cuenta con las reglas básicas para la realización de operaciones aritméticas binarias tales como la suma o la división.

A lo anterior hay que agregar una herramienta más que se conoce como Algebra de Boole, que explica el comportamiento lógico de un sistema digital, proporcionando los conocimientos básicos que habrán de permitir implementar diversas funciones con estos circuitos lógicos, como suele llamárseles.

Como posteriormente se verá, por medio del álgebra booleana se manipulan las variables booleanas y se desarrollan funciones que permitirán realizar una serie de interconexiones con los ya mencionados circuitos, de manera que en conjunto formen un sistema que ejecute una tarea determinada, como puede ser sumar o multiplicar dos cantidades binarias, o realizar un conteo a la manera de un reloj, etc.. Se espera entonces que cualquier tarea susceptible de reducirse a una serie de operaciones lógicas pueda ser implementada mediante un sistema digital.

C A P I T U L O I I

SISTEMAS NUMERICOS

SISTEMAS NUMERICOS.

Es una costumbre generalizada manejar un sistema numérico que consta de 10 dígitos (0,1,2,3,4,5,6,7,8,9), comúnmente llamado decimal. Así pues, se aprovechará la familiaridad con el sistema decimal y se utilizará como punto de referencia para el estudio de los sistemas numéricos en general.

Cuando se escribe un número cualquiera, como por, ejemplo:

847.6 (notación yuxtaposicional)

lo que se está haciendo es abreviar la expresión:

$$8 \times 10^{2} + 4 \times 10^{1} + 7 \times 10^{0} + 6 \times 10^{-1}$$ (notación polinomial)

Esto es, en el sistema decimal se tienen solamente diez dígitos, pero cada uno de ellos tiene un valor diferente según la posición que ocupe dentro del número, lo cual se conoce como valor posicional, y constituye una de las principales características de los sistemas numéricos.

El anterior es ejemplo de un número con parte fraccionaria, mismo que se identifica como tal gracias a que el sistema decimal cuenta con un elemento simbólico (el punto decimal) que permite diferenciar, en un número, la parte entera de la fraccionaria, cualidad que todo sistema debe poseer.

Ahora bien, expresar el número como una suma, en la que cada término es un múltiplo de una potencia de 10, da al sistema el nombre de decimal, conociéndose al número 10 como la base de dicho sistema.

Así, si se tuviera un número expresado como la suma de términos múltiplos de una potencia de 2, se estaría usando un sistema de numeración binario, es decir, de base 2.

De esta forma, usando diferentes bases, se puede crear tantos sistemas numéricos como se desee.

Para que un sistema numérico se considere completo, debe poseer sus propias reglas operacionales, es decir, su propia aritmética.

De todo lo dicho anteriormente se desprende que, en general, un sistema numérico está constituído por un conjunto ordenado de símbolos, llamados dígitos, que poseen reglas definidas para las operaciones aritméticas básicas; se dirá también que la base de un sistema numérico especifica el número de dígitos que incluye el conjunto ordenado y que, además, un sistema

numérico debe permitir a los números -colección de dígitos- tener una parte entera y otra fraccionaria separadas por un punto.

Se cuenta además con dos maneras de escribir un número: la polinomial, que es una suma cuyos términos se expresan como múltiplos de potencias de la base, y la yuxtaposicional, que es la usada comunmente, y que expresa solamente los coeficientes de la representación polinomial.

Ahora bien, el sistema numérico usado en el diseño de circuitos digitales es el binario, cuya base es 2, requiriendo únicamente de dos dígitos que son el 0 y el 1. La adopción de este sistema se debe al hecho de que dichos circuitos manejan señales binarias, es decir,tienen sólo dos posibles valores, uno de los cuales se asigna al 0 y el otro al 1.

También se usan frecuentemente otros sistemas numéricos, como son el octal (base 8) y el hexadecimal (base 16), mismos que resultan de gran utilidad para expresar cantidades muy grandes. En el caso del sistema hexadecimal se utilizan inclusive letras (A,B,C,D,E,F) para representar los valores más significativos, además de los diez dígitos usuales.

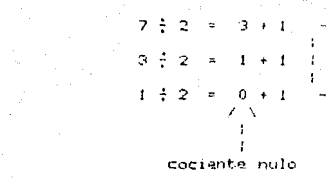
En la tabla 2.1 se muestran las equivalencias entre los sistemas decimal, octal, binario y hexadecimal.

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

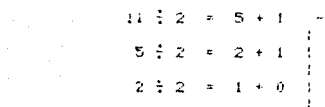
Tabla 2.1 Equivalencia entre 4 sistemas numéricos.

A continuación se explica con detalle la conversión de un número de una base dada a otra, que puede ser decimal, binaria, octal, etc. Este tipo de transformaciones son importantes dado que a menudo será necesario traducir información de un sistema numérico a otro.

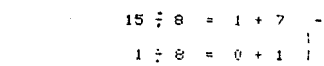
Dividimos primero hasta obtener un cociente nulo:



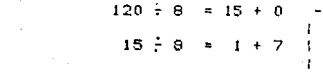
b) 11 ------> binario
10



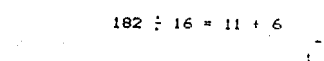
c) 15 ----------> octal
10



d) 120 ----------> octal
10



e) 182 --------> hexadecimal
10



CONVERSIONES NUMERICAS.

De base r a decimal.- Si se tiene un número N expresado en una base r#10, (Nr) es posible convertir este número a su equivalente decimal mediante un proceso muy sencillo: se expresa Nr en su forma polinomial y se efectuan los productos y sumas indicadas.

$$a)\ 1111_{2} = 1\times2^{3} + 1\times2^{2} + 1\times2^{1} + 1\times2^{0} = 15_{10}$$

número en base 2 — expresión polinomial — equivalente decimal

$$b)\ 1210_{3} = 1\times3^{3} + 2\times3^{2} + 1\times3^{1} + 0\times3^{0} = 48_{10}$$

$$c)\ 70.2_{8} = 7\times8^{1} + 0\times8^{0} + 2\times8^{-1} = 56.25_{10}$$

$$d)\ AC.\ 18_{16} = A\times16^{1} + C\times16^{0} + 1\times16^{-1} + 8\times16^{-2} = 172.09375_{10}$$

De base decimal a base r.- Como ya se ha visto, un número expresado en una determinada base consta de una parte entera y otra fraccionaria. La parte entera en una base equivale siempre a la parte entera en la otra base, y análogamente, la parte fraccionaria. Esto permite tratar por separado la conversión de cada una de las partes, ya que los métodos no son iguales.

Conversión de la parte entera.- El método es sencillo y consiste en dividir progresivamente el número N decimal por la base r a la que se quiere pasar. Se divide la primera vez y se obtienen un cociente y un residuo; el cociente se vuelve a dividir por la base, obteniéndose un nuevo cociente y otro residuo. La división del cociente por la base se continúa hasta obtener un cociente igual a cero. Se toman entonces los residuos para formar el número en base r.

Los siguientes ejemplos mostrarán con claridad cómo realizar este proceso:

a) Se tiene el número decimal N=7 y se desea expresarlo en la base r=2, i.e., 7_{10} -----> binario

f) 333 --------> hexadecimal
10

```
333 ÷ 16 = 20 + 13  ----------> 14D = 333
                    !                16    10
 20 ÷ 16 =  1 +  4  !
                    !
  1 ÷ 16 =  0 +  1  !
```

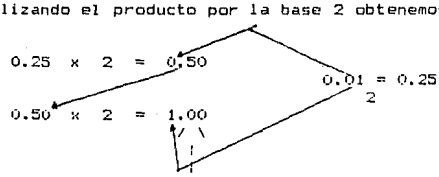
Conversión de la parte fraccionaria.- Sea N un decimal con parte fraccionaria. Para hacer la conversión de la fracción decimal a la base r se hace lo siguiente: se multiplica la fracción por la base r a la que se desea pasar. Como resultado de esta primera operación se obtiene un número N1 que posee parte entera y fraccionaria; la nueva parte fraccionaria se vuelve a multiplicar por la base r, con lo que se obtiene nuevamente un número N2 con partes entera y fraccionaria como resultado. La multiplicación de la parte fraccionaria por la base se continúa según se desee y, si es posible, deberá realizarse hasta obtener un resultado Nn (n=natural) con parte fraccionaria nula, se tendrá un equivalente exacto en la base r de la fracción decimal de N. Si por el contrario, la parte fraccionaria del resultado nunca llega a cero, se obtendrá tan sólo una aproximación.

Así pues, la fracción equivalente en la base r se obtiene escribiendo, en el orden en que se obtuvieron, las partes enteras del resultado de cada multiplicación.

Algunos ejemplos mostrarán más claramente cómo se realiza esta conversión:

a) Conviétase la fracción decimal n=0.25 en un número binario, i.e., 0.25 -----> binario
10

Realizando el producto por la base 2 obtenemos:



En este ejemplo se ha obtenido rapidamente un resultado exacto, es decir, un producto donde la fracción es nula. Vease por qué es exacto:

$$0.01_{2} = 0\times2^{-1} + 1\times2^{-2} = 0.25$$

Sin embargo, esto no siempre ocurre, como podrá verse en el siguiente ejemplo:

b) 0.3_{10} ------> binario

Realizando el producto por la base 2 se obtiene:

$$\begin{aligned}
0.3 \times 2 &= 0.6\\
0.6 \times 2 &= 1.2\\
0.2 \times 2 &= 0.4\\
0.4 \times 2 &= 0.8\\
0.8 \times 2 &= 1.6\\
0.6 \times 2 &= 1.2\\
0.2 \times 2 &= 0.4\\
&\vdots
\end{aligned}$$

El producto podría continuar realizándose indefinidamente y quizá no se lograría obtener jamás un resultado con parte fraccionaria nula. La equivalencia binaria que se obtiene es entonces una aproximación, la que evidentemente será mayor cuanto mayor sea el número de dígitos que se obtienen.

Se tiene entonces que $0.3_{10} \cong 0.0100110$

puesto que

$$0.0100110 = 1\times2^{-2} + 1\times2^{-5} + 1\times2^{-6} = 0.2980625 \cong 0.3$$

c) 0.125_{10} ------> octal

$$0.125 \times 8 = 1.000 \quad =====> \quad 0.125_{10} = 0.1_{8}$$

d) 0.52_{10} ------> octal

$0.52 \times 8 = 4.16$
$0.16 \times 8 = 1.28$
$0.28 \times 8 = 2.24$
$0.24 \times 8 = 1.92$
$0.92 \times 8 = 7.36$
$0.36 \times 8 = 2.88$
.
.
.

=======> $0.52_{10} \approx 0.412172_{8}$

e) 0.75_{10} ------> hexadecimal

$0.75 \times 16 = 12.0$ =====> $0.75_{10} = 0.C_{16}$

f) 0.32_{10} ------> hexadecimal

$0.32 \times 16 = 5.12$
$0.12 \times 16 = 1.92$
$0.92 \times 16 = 14.72$
$0.72 \times 16 = 11.52$
$0.52 \times 16 = 8.32$
$0.32 \times 16 = 5.12$
.
.
.

========> $0.32_{10} \approx 0.51EB85..._{16}$

Cabe aquí hacer notar que si se desea transformar un número de base r a base r', se realizaría, más fácilmente, convirtiendo primero de r a decimal y después de decimal a r'. excepto en algunos casos que se tratan a continuación.

Conversión octal-binario y binario-octal.- La mejor forma de explicar este tipo de conversiones es a través de un ejemplo, así que primero se va a ejemplificar la conversión binario-octal.

Supóngase que se tiene el número binario

1 1 1 0 1 1 0 1 1 . 0 1 0

Divídase en grupos de tres dígitos cada uno, a partir del

punto hacia la izquierda para la parte entera, y hacia la derecha para la fraccionaria, esto es:

111 011 011 . 010

Ahora bien, se sabe que con tres dígitos binarios se puede formar hasta ocho números (del 0 al 7), los ocho dígitos que constituyen el sistema octal, por lo que cada grupo o terna de dígitos puede hacerse corresponder con una cifra octal, esto es:

```
111 011 011 . 010
 |   |   |     |
 |   |   |     |
 V   V   V     V
 7   3   3     2
  8   8   8     8
```

Por lo que el número binario 111011011.010 se transforma en su equivalente octal 733.2

El proceso de conversión octal-binario es, como puede suponerse, el opuesto al anterior. Así, supongase que se tiene el número octal 714.02. La conversión a binario se hará dígito a dígito, es decir:

```
   7     1     4   .   0     2
   |     |     |       |     |
   V     V     V       V     V
  111   001   100     000   010
    2     2     2       2     2
```

y, por lo tanto,

$$714.02_{8} = 111001100.000010_{2}$$

A continuación se muestran dos ejemplos más:

a) 1101001.110_{2} -----> octal

```
   1   101   001 .  110
   |    |     |      |
   V    V     V      V
   1    5     1      6     =====> 1101001.110 = 151.6
                                              2       8
```

b) 1765 ------> binario
8

```
        1   7   6   5

        :   :   :   :
        V   V   V   V
       001 111 110 101      ======> 1765  = 001111110101
                                        8              2
```

Conversión binario-hexadecimal y hexadecimal-binario.- El procedimiento para realizar cada una de estas conversiones es similar al del caso octal/binario, con la salvedad de que se usan 4 dígitos binarios en lugar de 3. Véase esto a través de un ejemplo:

Supóngase que se tiene el número binario 101110010010.1001 . Como con cuatro dígitos se puede formar un máximo de dieciséis números (del 0 al 15), mismos que conforman al sistema hexadecimal, es posible hacer la correspondecia de un dígito hexadecimal con un grupo de cuatro dígitos binarios, es decir,

```
      1011    1001    0010   .    1001

      :       :       :           :
      V       V       V           V
      B       9       2           9
       16      16      16          16
```

por lo que

$$101110010010.1001 = B92.9_{2} \qquad _{16}$$

El proceso de conversión hexadecimal a binario resulta inverso al anterior: se hace corresponder a cada dígito hexadecimal un grupo de cuatro dígitos binarios. Así, si se tiene la cantidad hexadecimal A35.2 y se desea convertir a binaria, simplemente transformamos dígito a dígito, esto es:

```
      A       3       5      .    2

      :       :       :           :
      V       V       V           V
    1010    0011    0101        0010
```

y, por lo tanto,

$$A35.2 = 101000110101.0010_{2} \qquad _{16}$$

Véanse dos ejemplos más:

a) 111100111.0010 -----> hexadecimal

```
          1    1110    0111 .   0010

          ¦     ¦      ¦         ¦
          V     V      V         V
          1     E      7         2   ====> 111100111.0010 = 1E7.2
                                                          2        16
```

b) CF.34 -----> binario
16

```
          C     F     .     3     4

          ¦     ¦           ¦     ¦
          V     V           V     V
         1100  1111        0011  0100   ====> CF.34 = 11001111.00110100
```

Nótese que, para expresar cantidades muy grandes, resulta conveniente usar los sistemas octal y hexadecimal a fin de reducir la extensión de las espresiones numéricas.

ARITMETICA BINARIA

Las operaciones aritmeticas binarias se realizan en forma similar a como se efectúan en decimal y por ello es que en algunos casos se hace referencia a dichos procedimientos decimales a fin de aclarar algún concepto en particular.

Suma binaria.- Sumar significa representar en una sola la reunión de varias cantidades. Si se tienen, por ejemplo, los dos números decimales 28 y 17 y se suman se obtiene otra cantidad que representa la conjunción de los dos números: 28 + 17 = 45.

Lo importante es entender cómo se obtiene esta conjunción: Lo que se hace es sumar los dígitos de cada número que poseen el mismo valor posicional. Así, se suman unidades con unidades, decenas con decenas, centenas con centenas, etc. Sin embargo, puede ocurrir que la suma de dos dígitos exceda de 9 (dígito mayor del sistema decimal), lo cual se traduce en la existencia de un acarreo que se suma al siguiente dígito, esto es,

```
   1    acarreo
 + 2 8
   1 7
 -------
   4 5
```

En el sistema binario se cuenta sólo con los dos dígitos 0 y 1, de manera que el acarreo existe cuando la suma excede de 1. La suma binaria puede, entonces, definirse mediante las siguientes cuatro reglas:

```
0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 0 y existe un acarreo c=1
```

Véanse algunos ejemplos de suma binaria:

a) Suma de 001+010

```
   001
 + 010
 -----
   011
```

b)Suma de 11+01

```
   11  <---- acarreos
   11
 + 01
 -----
  100
```

c) Suma de .1+10+11.1

```
  111 <---- acarreos
   00.1
+  10.0
   11.1
 -------
  110.0
```

d) Suma de 001+010+011+100

```
  111 <---- acarreos
   001
   010
+  011
   100
 -------
  1010
```

Resta binaria.- La substraccion se entiende como el proceso inverso al de adición. La resta binaria puede definirse, al igual que la suma, a través de un conjunto de cuatro reglas, mismas que se presentan a continuación:

```
0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1   con un préstamo b=1
```

Existen otros métodos más sencillos para realizar la resta que el que aquí se muestra, los cuales serán tratados más adelante.

Por lo pronto, se verán algunos ejemplos de substracción binaria:

a) Resta de 101.1-100.1

```
   101.1
 - 100.1
 ---------
   001.0
```

b)resta de 1001-0111

```
    1001
  - 11 <--- préstamos
    0111        9
  -------     - 7
    0010      ------
                  2
```

c) Resta de 1111.1-0010.1

```
   1111.1
 - 0010.1
 ---------
   1101.0
```

d) Resta de 1010-0101

```
    1010
  - 1 1 <--- préstamos
    0101
 ---------
    0101
```

Multiplicación binaria.- Al igual que en el caso decimal, se puede definir esta operación según una "tabla" de multiplicación, que a continuación se presenta:

$$0 \times 0 = 0$$
$$1 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 1 = 1$$

Como se ve, las reglas son muy simples y el producto se puede efectuar en la forma usual con que se realiza el decimal.

Véanse algunos ejemplos:

a) 1111 x 10 = 11110

b) 10.01 x 0.01 = 0.1001

c) 110.1 x 10.11 = 10001.111

División binaria.- El procedimiento es sencillo y, al igual que en el sistema decimal, la división por cero no está definida, por lo que las únicas reglas que se tienen son las dos siguientes:

$$0 \div 1 = 0$$

$$1 \div 1 = 1$$

Se puede realizar la división según la forma convencional a la que estamos acostumbrados. Veamos esto en los siguientes ejemplos:

a)

```
          1 1 1 1.1
     ______________
  10 | 1 1 1 1 1
     - 1 0
       ---
       0 1 1
     -   1 0
     -------
         0 1 1
      -    1 0
      ---------
           0 1 1
         -   1 0 0
         ---------
             0 1 0
               1 0
             -----
                 0
```

b)

```
           1 0 1 . 0 1
      ________________
  1 0 | 1 0 1 0 . 1 0
        0 0·1 0
              0   1 0
                  0 0
```

c)

```
           0 . 0 1 0 1
      ________________
  1 1 | 0 1
      - 0 0
      -----
        0 1   0
      -   0   0
      ---------
          1   0 0
      -       1 1
      -----------
          0   0 1 0
      -         0 0
      -------------
                1 0 0
      -           1 1
      ---------------
              0 0 0 1
                    .
                    .
                    .
```

Complemento de un número.- Dado un número positivo N en una base r, con parte entera de n dígitos, el complemento a r de N se define como

$$N_r = r^n - N, \text{ para } N \neq 0 \qquad \ldots\ldots \text{(A)}$$

y

$$N_r = 0, \qquad \text{para} \quad N = 0$$

Así, el complemento a 10 de 9 resulta ser 1, puesto que

$$r^n - N = 10^1 - 9 = 1$$

El complemento a 10 de 99 también es 1 ya que

$$r^n - N = 10^2 - 99 = 1$$

El complemento a 2 de 10101 es

$$r^n - N = (2^5)_2 - 10101 = 100000_2 - 10101_2 = 1011_2$$

De los ejemplos se ve claramente que el complemento a la base de un número es la cantidad que "le falta" a éste para igualar a otro que es la siguiente potencia de la base.

Complemento a la base disminuída.- Dado un número N en base r, con parte entera de n dígitos, y parte fraccionaria de m dígitos, el complemento a la base disminuída se define como

$$N_{r-1} = r^n - r^{-m} - N \qquad \ldots\ldots (B)$$

Así por ejemplo, el complemento a 9 de 51600 es:

$$r^n - r^{-m} - N = 10^5 - 10^0 - 51600 = 48399$$

O bien, el complemento a 1 de 1011100_2 es:

$$(2^7) - 2^0 - 1011100 = 10000000 - 1 - 1011100 = 0100011$$

En el caso específico de complementos de números binarios, los procedimientos anteriores pueden simplificarse como sigue:

Complemento a 1 de un número binario.- Como se vió en el ejemplo anterior, el complemento a 1 de 1011100 es 0100011. Si se observa bien, se encontrará que se han intercambiado los 0's por 1's y viceversa, de modo que complementar a 1 un número binario se reduce a cambiar 0's por 1's y 1's por 0's. Así, el complemento a 1 de 1010 será 0101 y el de 1111101.01, 0000010.10.

Complemento a 2 de un número binario.- De las definiciones de complemento a r y a r-1 (ecs. A y B) se tiene que:

$$N_r = r^n - N \qquad y \qquad N_{r-1} = (r^n - N) - r^{-m}$$

de donde

$$N_{r-1} = N_r - r^{-m}$$

o bien

$$N_r = N_{r-1} + r^{-m} \qquad \ldots\ldots (C)$$

ecuación que facilita la obtención del complemento a r de un número N.

Dada entonces la simplicidad con que puede obtenerse el complemento a 1 de un número binario, la ecuación C proporciona una manera muy sencilla de obtener el complemento a 2.

Así, obténgase el complemento a 2 de 10111.111.

Por C se tiene que

$$N_2 = N_1 + (2^{-3}) = 01000.001$$

De igual manera, el complemento a 2 de 101010 es:

$$010101 + 2^0 = 010101 + 1 = 010110$$

Si se observa bien en los dos ejemplos anteriores, se encuentra que obtener el complemento a 2 de un número binario es equivalente a obtener su complemento a 1 y sumar un 1 en la posición menos significativa.

Lo anterior puede simplificarse aún más: se toma el número por la extrema derecha y, si existen ceros, éstos no se alteran, lo mismo que el primer uno que se encuentre; el resto de los dígitos se cambian los 1's por 0's y viceversa.

A continuación se muestran tres ejemplos, cada uno de los cuales ilustra uno de los métodos mencionados para complementar a 2 un número binario.

```
                                                    1er
a)      N  =  1 0 1 0 1 1 . 1 0 1  <-----  uno
              ! ! ! ! ! !   ! ! !
              V V V V V V   V V V
 ===>  N  =  0 1 0 1 0 0 . 0 1 1
        2
```

```
b)      N  =  1 0 0 0 1 0 0
```

$$\Longrightarrow \quad N_2 = (2^7)_2 - 1\,0\,0\,0\,1\,0\,0 = 00111100$$

```
c)    N = 1 1 1 1 1
```

$$\Longrightarrow \quad N_2 = 0\,0\,0\,0\,0 + 1 = 00001$$

Substracción por complemento a r.- Haber tratado el complemento de un número tiene su razón de ser, ya que ello facilitará la realización de una resta y, por tanto, si "se piensa en binario", se tendrá que se podrá usar la misma circuiteria para realizar tanto la suma como la resta.

Supóngase entonces que se tienen dos números M y N de la misma base r y se desea realizar la resta M - N. El modo de efectuar esta operación será el siguientes:

a) Se suma el minuendo (M) al complemento a r del sustraendo (N).

b) Si existe acarreo final como resultado de la suma, el número que se obtiene se considera ya un resultado definitivo, no tomando en cuenta el acarreo.

c) Si no existe acarreo, el resultado se tendrá que complementar a r. Esto indica que el minuendo es menor que el sustraendo, por lo cual se antepone un signo menos al resultado a fin de indicar que éste ha de considerarse negativo.

Los siguientes ejemplos muestran con claridad el método expuesto.

a)

Sean M = 489 y N = 037

$$M - N \Longrightarrow 489 + 963 \quad \text{pues} \quad N = 10^{3} - 37 = 963$$

Efectuando la suma se obtiene que

```
   489
 + 963
-------
 1 452
/     \
acarreo   resultado
```

b) Sean M = 037 y N = 489

$$\Longrightarrow N = 10^{3} - 489 = 511$$

$$M - N \Longrightarrow 037 + 511 = 01548$$

$$R = 10^{3} - 548 = 452$$

c = 0 ; R = resultado

que se denota como

$$M - N = -452$$

c) Sean $M = 101.0_{2}$ y $N = 011.1_{2}$

$$\Longrightarrow N = 100.1_{2} \quad \text{y} \quad M - N \Longrightarrow 101.0 + 100.1$$

```
 1 001.1
/       \
c = 1    resultado
```

d) Sean M = 0111 y N = 1010

```
===>  N  = 0110    y   M - N  ===>   0111
       2                          +  0110
                                    ---------
                                     0 1101   = R ==> R  = 0011
                                    /                    2
                                   /
                                c = 0            . .   M - N = -0011
```

Nótese que el acarreo tiene el valor de r^n, así, en el ejemplo "a" el acarreo es 10^3 y en el "c" es 2^4.

Se verá ahora el por qué de estar realizando la resta de esta manera.

Como se vio anteriormente $Nr = r^n - N$
entonces,

$$M + Nr = (M-N) + r^n$$

aquí se presentan dos casos:
Si existe acarreo al hacer la suma M+Nr se tiene que

$$M+Nr \geq r^n$$

entonces.

$$M+Nr-r^n \geq 0$$

y como

$$M+Nr-r^n = M-N \quad .. \quad M-N \geq 0$$

y, por tanto. $M \geq N$

Si, por el contrario, no existe acarreo

$$M+Nr < r^n$$

$$M-N < 0$$

y, por tanto

$$M \leq N$$

razón por la cual se considera negativo el resultado.

Substracción por complemento a $r - 1$.- Supóngase que se tienen dos números M y N de la misma base r. Para realizar la resta M - N por complemento a la base disminuída se realizan los siguientes pasos:

a) Sumar el minuendo (M) al complemento a $r-1$ del sustraendo (N).

b) Si existe un acarreo final como resultado de la suma, se suma un 1 al dígito menos significativo para obtener un resultado definitivo.

c) Si no existe acarreo final, se deberá obtener el complemento a la base disminuída r-1 del resultado. Se le antepone un signo menos para indicar que el resultado es negativo.

Veamos algunos ejemplos.

a) Sean M = 4853 y N=0384

```
                                    4853
                                +   9615
    ====> N  =  9615       M - N ===>  ---------
          9                           1 4468 ==> 4468
                                     /           +    1
                                    /            -------
                                c = 1              4469  = R
```

b) Sean M = 0384 y N = 4853

```
    ===> N  =  5146       y   M  -  N  ===>   0384
          9                                 +  5146
                                            -------- ==>
                                             0 5530
                                            /
                                           /
                                      c = 0

                                                  .
                                                . . R  = 4469
                                                     9
```

```
                                    .
                                  . . M - N = -4469
c) Sea M = 1010   y  N  =  0111
                2             2

   ===> N  = 1000        M-N  ===>    1010
         1                         +  1000
                                   ----------
                                     1 0010  ==>  0010
                                    /              +    1
                                   /               --------
                               c = 1                 0011  =R
```

d) Sea $M = 0111_2$ y $N = 1010_2$

$\Longrightarrow N_1 = 0101$ $\quad M - N \Longrightarrow$

$$\begin{array}{r} 0111 \\ +\ 0101 \\ \hline 0\ 1100 \end{array}$$

$c = 0$

$\Longrightarrow R = 1100 \Longrightarrow R_1 = 0011$

$\therefore M - N = -0011$

A todo esto, se puede decir que los complementos, como ya se ha visto, van a resultar muy útiles para la manipulación artimética en sistemas digitales. Sin embargo, existe la particularidad de que el complemento a 1 de números binarios es de suma utilidad en operaciones puramente lógicas ya que, como se verá después, el intercambio de 0's por 1's y de 1's por 0's, equivale a la "inversión lógica". El complemento a 2 tiene, por el contrario, aplicaciones puramente aritméticas.

CAPITILO III

CODIGOS

CODIGOS

En este capítulo se expondrá la forma en que la información ya sea numérica o alfabética puede representarse en términos de ceros y unos, lo que permitirá su manejo por medio de circuitos digitales.

El proceso mediante el cual se representa información, por medio de un grupo especial de símbolos, se conoce como codificación, en tanto que al citado grupo de símbolos como código.

Un ejemplo ampliamente conocido de lo que es un código lo constituye la clave Morse que se usa para el envío de telegramas y que representa cualquier mensaje mediante una serie de arreglos de dos símbolos: punto y raya.

En realidad, se pueden encontrar miles de códigos con diversas características; podría nombrarse, por ejemplo, el código de colores de las resistencias eléctricas. Lo importante es que toda clase de información puede ser codificada, ya sea con letras, números, colores, sonidos, etc., los cuales, a su vez, pueden hacerse corresponder con una o varias palabras codificadas en binario, es decir, por un conjunto de n bits (dígitos binarios).

Véase ahora, y a manera de ejemplo, cómo se puede crear un código cualquiera en binario que represente a cada número decimal del 0 al 15 (Tabla 3.1). Es importante observar que la asignación es arbitraria y no corresponde a la numeración binaria usual.

NUMERO DECIMAL	CODIGO BINARIO
0	0000
1	0010
2	0100
3	1000
4	0001
5	1001
6	1010
7	1100
8	1101
9	1011
10	0111
11	1110
12	0011
13	0101
14	1111
15	0110

Tabla 3.1 Ejemplo de código binario.

Como se podrá observar en la tabla, a cada número se le hace una asignación única, lo cual es importante ya que, de lo contrario, el código sería ambiguo y su interpretación representaría un problema.

Por otro lado, si se tiene cierta información ya codificada y se desea transformarla a su forma original, se realiza el proceso denominado decodificación que, obviamente, resulta ser el proceso inverso a la codificación.

En general, una gran mayoría de los esfuerzos que se realizan en este campo, se encaminan hacia la creación de códigos para la transmisión de información que no es del dominio público, lo cual hace necesario que éstos sean lo suficiente complejos como para que su decodificación sea casi imposible por personas no expertas.

Se han realizado también múltiples intentos por estandarizar los códigos más usuales. Una de tantas clasificaciones es la siguiente:

- Códigos Decimales Codificados en Binario (BCD)
- Códigos Reflejados
- Códigos de Distancia Unitaria
- Códigos de Detección de Errores
- Códigos Alfanuméricos

Además de los mencionados, los sistemas binario, octal, hexadecimal, etc., pueden ser considerados como códigos (mismos que ya han sido tratados anteriormente en este texto).

A continuación se describirán, de una manera muy general, los tipos de códigos mencionados en la lista anterior.

Códigos decimales codificados en binario (BCD).- Los sistemas digitales usan para su funcionamiento interno números binarios; no obstante, su interacción con el mundo exterior crea la necesidad de que estos sistemas cuenten con la capacidad de trabajar con números decimales. Tal sería el caso de multímetros y calculadoras, que deben desplegar sobre una pantalla cantidades decimales.

Realizar conversiones binario-decimal y viceversa de números grandes puede no ser sencillo, ya que el proceso resulta largo y tedioso. Debido a esto se han creado códigos que reúnen las características de ambos sistemas y que se conocen como Códigos Decimales Codificados en Binario o, abreviadamente, como códigos BCD.

Como en todo código binario, en los códigos BCD cada dígito decimal se representa por un conjunto de bits. El código BCD más usual es el NBCD (Natural Binary Code Decimal *), el cual asigna a cada dígito decimal su equivalente en binario.

Ahora bien, como se desea representar 10 valores mediante distintas combinaciones de 0's y 1's se requiere cuando menos de 4 bits. De esta manera, el código NBCD queda representado según la tabla 3.2.

Para ilustrar el uso de este código se representará el número 895.5 asignando directamente a cada dígito su equivalente NBCD, es decir:

```
   8    9    5  .  5
   :    :    :     :
   :    :    :     :
   V    V    V     V
1000 1001 0101 0101
```

o equivalente:

$$895.5 = 100010010101.0101_{NBCD}$$

NUMERO DECIMAL	NBCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Tabla 3.2 Código NBCD

Nótese que las combinaciones 1010, 1011, 1101, 1100, 1110 y 1111 no forman parte del código. Sólo se usan 10 de las 16 posibles combinaciones de los 4 bits son usadas.

---

* Debido a la frecuencia con que se emplea el código NBCD, en la mayoría de la bibliografía disponible se le conoce como BCD simplemente.

Debe quedar clara la diferencia que existe entre el código NBCD y el sistema binario. Si se tiene, por ejemplo el número 12, en binario se expresaría como

1100

mientras que en NBCD como

00010010

de lo cual se deduce que es muy diferente codificar un número en NBCD que convertirlo a binario.

Además del NBCD existe una gran variedad de códigos BCD, algunos de los cuales se muestran en la tabla 3.3.

DIGITO DECIMAL	NATURAL BCD 8421	EXCESO-3	84-2-1	2421	631-1	5421
0	0000	0011	00 0 0	0000	00 1 1	0000
1	0001	0100	01 1 1	0001	00 1 0	0001
2	0010	0101	01 1 0	0010	01 0 1	0010
3	0011	0110	01 0 1	0011	01 1 1	0011
4	0100	0111	01 0 0	0100	01 1 0	0100
5	0101	1000	10 1 1	1011	10 0 1	1000
6	0110	1001	10 1 0	1100	10 0 0	1001
7	0111	1010	10 0 1	1101	10 1 0	1010
8	1000	1011	10 0 0	1110	11 0 1	1011
9	1001	1100	11 1 1	1111	11 0 0	1100

Tabla 3.3 Códigos BCD.

La mayoría de estos códigos se forman asignando un factor de peso a cada posición dentro del conjunto de bits, razón por la cual son llamados códigos posicionales o ponderados. De esta forma, un número cualquiera puede decoficarse sumando los factores de peso de las posiciones donde se encuentra un 1. Así por ejemplo, el número 0111 en el código 84-2-1 representa al 1 decimal dado que

```
8   4   -2   -1
|   |    |    |
|   |    |    |
V   V    V    V
0   1    1    1 = 0x8 + 1x4 + 1(-2) + 1 (-1) = 1
```

Así mismo, el 8 decimal puede representarse como 1000 ya que

```
8  4  -2  -1
|  |   |   |
|  |   |   |
V  V   V   V
1  0   0   0   = 1(8) + 0(4) + 0 (-2) + 0 (-1) = 8
```

Así pues, asignando diversos factores de peso a cada uno de los bits de un arreglo se puede crear una infinidad de códigos, aunque algunos de ellos poseen ciertas características que los hacen preferibles para determinado tipo de aplicaciones. Así, por ejemplo, los códigos 631-1, 2421 y 84-2-1 de la tabla 3.5 son autocomplementarios, es decir, el complemento a 9 del número decimal representado mediante uno de estos códigos se obtienen cambiando los 0's por 1's y viceversa.

Si, por ejemplo, se tienen el número 25 en el código 84-2-1

0110 1011

cambiando 0's por 1's y 1's por 0's se obtiene el número

1001 0100

que representa al 74, el cual es precisamente el complemento a 9 de 25, ya que

74 + 25 = 99 (si hay duda a este respecto, remitirse a la sección de complementos)

Esta propiedad (autocomplementariedad) es muy útil si el sistema digital en cuestión realiza operaciones como la resta o la suma de números negativos, representados en estos códigos

En la tabla 3.3 aparece también el código Exceso-3, el cual, a diferencia de los demás, no se forma mediante factores de peso, i.e., no es un código posicional, sino que se forma simplemente sumando un 3 binario a la representación en el sistema binario usual de los 10 dígitos decimales. Así, si se tiene el número 4 en representación binaria

0100

sumándole 0011 (3_{10}) se obtiene su representación en el código Exceso-3

0100 + 0011 = 0111

es decir,

4_{10} = 0111 Exceso-3

Este es también un código autocomplementario y es usado por algunas calculadoras dado que simplifica las operaciones aritméticas internas de la misma.

Como ejemplo de las operaciones que se puedan realizar con este código, se verá a continuación la suma. Esta se realiza de manera muy sencilla ya que utiliza las mismas reglas que la adición binaria común, con las siguientes dos reglas adicionales:

1) Si la suma produce un acarreo, se le suma al resultado un 0011 (3).
10

2) Si la suma no produce acarreo, se le resta un 0011 al resultado.

Ambas operaciones se usan para que la suma quede correctamente expresada en el código Exceso-3.

Se verá un ejemplo sencillo de esto que se ha dicho:

Sumando 44 + 37 en representación Exceso-3

```
     0 1 1 1   0 1 1 1
+
     0 1 1 0   1 0 1 0   <--- representación Exceso-3
 ------------------------
     1 1 1 0   0 0 0 1
-          +
     0 0 1 1   0 0 1 1   <--- corrección
 -----------   ---------
     1 0 1 1   0 1 0 0   <--- resultado correcto
```

En la suma de los 4 bits de la derecha (0111+1010) se presenta un acarreo por lo que se suma 3 al resultado, en la suma de los 4 bits de la izquierda (0111+0110) no hay acarreo por lo que se resta 3 al resultado.

Como el código es autocomplementario, tiene la ventaja de que pueden usarse los complementos a 1 y a 2 para realizar la resta.

Códigos reflejados.- La principal característica que distingue a este tipo de códigos es la simetría casi perfecta de los mismos, respecto de su centro ya que, si imaginamos un espejo en su centro, entre el 4 y el 5 de uno de los códigos de la tabla 3.4, se encuentra que los números del 0 al 4 son un "reflejo" de los números del 5 al 9, excepto por un bit. Esta misma diferencia es la que permite que el complemento a 9 de un número decimal representado en este tipo de códigos se obtenga fácilmente al cambiar el valor de un determinado bit.

Véase un ejemplo de esto: el número 4, expresado por medio del código reflejado I de la tabla 3.4, se representa como el conjunto de bits 0100. Su reflejo o imagen con respecto al centro es el número 5, el cual se obtiene sencillamente cambiando el bit más significativo del número 4, es decir

```
 0  1  0  0  =  4
---
 ¦
 V
 1  1  0  0   =  5
---
```

Si se recuerda que el complemento a 9 de 4 es justamente 5, se notará que es muy fácil obtener dicho complemento con un código reflejado, particularidad que además lo hace muy útil en dispositivos digitales que realizan operaciones aritméticas como la substracción.

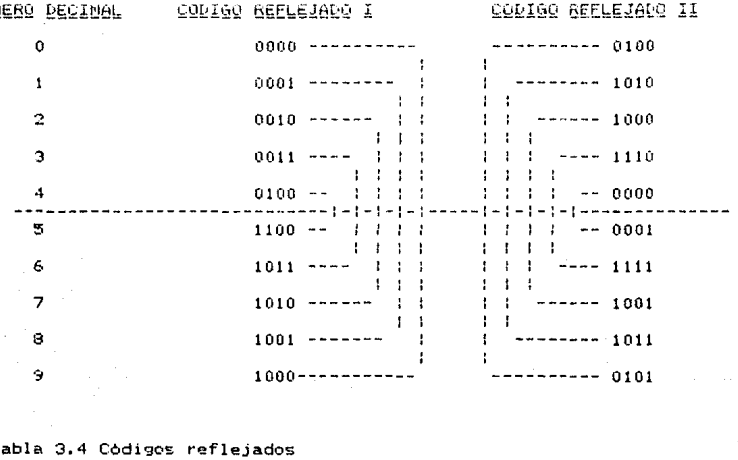


Tabla 3.4 Códigos reflejados

Códigos de distancia unitaria.- Estos códigos fueron creados para utilizarse en diversos dispositivos de entrada/salida, equipo periférico y convertidores analógico/digitales. No obstante, las ventajas que representan para dichas aplicaciones se convierten en una serie de inconvenientes para la realización de operaciones aritméticas.

No son códigos posicionales y su principal característica reside en el hecho de que cualquier número o conjunto de bits del código difiere de sus adyacentes (superior e inferior) en un solo bit.

Por ejemplo, el número 5 se expresa como 0111, el número siguiente, el 6, podría representarse como 1111.

Esta característica se transforma en cualidad principal cuando este tipo de códigos se usan en dispositivos de entrada/salida, ya que minimizan un tipo de errores conocidos como "transicionales" o de "reflejo".

La tabla (No. 3.5), muestra dos ejemplos de códigos de distancia unitaria, que abreviadamente se conoce como códigos UDC.

NUMERO DECIMAL	CODIGO I	CODIGO II
0	0000	0000
1	0001	1000
2	0011	1001
3	0010	0001
4	0110	0011
5	1110	0111
6	1111	1111
7	1101	1010
8	1100	0010
9	0100	0110

Tabla 3.5 Códigos de Distancia Unitaria.

Código gray.- Este tipo de código es considerado un caso especial, ya que reúne en sí tanto las características de los códigos reflejados como las de los UCD. No es un código posicional ni tampoco BCD.

En la tabla 3.6 se presenta un ejemplo de código gray.

Se puede hacer esta tabla tan larga como se desee usando la siguiente regla:

Primero se construye la columna menos significativa con un cero al principio y grupos alternados de dos unos y dos ceros. Las columnas restantes se construyen colocando en los primeros lugares un grupo de ceros cuyo número es el doble de los ceros iniciales de la columna anterior, a continuación se hacen grupos alternados de unos y ceros que tienen el doble de elementos que los grupos de la columna anterior.

alternados de unos y ceros que tienen el doble de elementos que los grupos de la columna anterior.

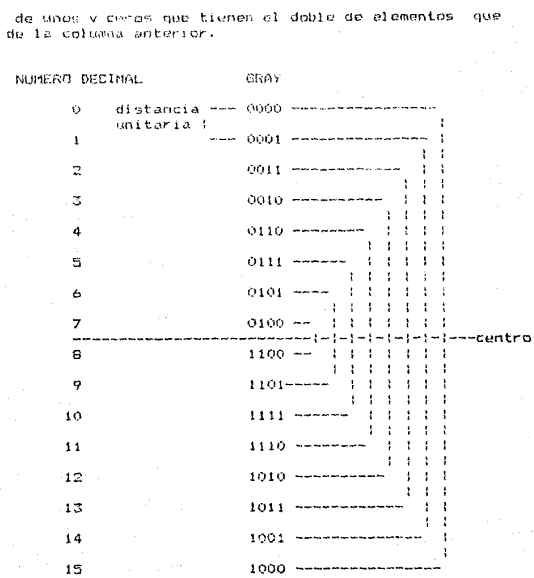


Tabla 3.6 Código Gray.

El código gray es muy práctico dada la sencillez con que se puede, a partir de él, pasar al sistema binario y viceversa. Veáse un ejemplo de esto:

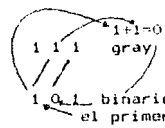
Supóngase que se tiene el número 101 binario.

Para pasar a código gray se hace lo siguiente: el primer dígito, de izquierda a derecha, no se altera, es decir, será el mismo en ambas representaciones; el siguiente dígito gray se obtiene sumando el primer dígito binario a su adyacente,

despreciándose el acarreo si es que existe. El proceso de sumar cada dígito a su adyacente se continúa hasta llegar el último par; el resultado de cada suma constituye un dígito gray. Así entonces:

```
1 0 1  binario
|
V
1 1 1  gray
```

El proceso inverso, gray-binario, es como sigue: el dígito más significativo no se altera. Para obtener los siguientes dígitos binarios se suma diagonalmente, es decir, el primer dígito binario de izquierda a derecha se suma al segundo dígito gray y el resultado será el segundo dígito binario, como se muestra a continuación:



Los acarreos se desprecian.

El código gray no es apto para operaciones aritméticas. Su utilidad es semejante a la de los códigos UDC: se usa ampliamente en dispositivos de entrada/salida, convertidores A/D y equipo periférico.

Códigos de detección de errores.- En un sistema digital se realiza siempre una transmisión de información, ya sea a distancias cortas, como en el interior de un circuito, o largas, como en el caso de redes de telecomunicación. Durante este proceso, la información puede sufrir cambios accidentales (inversión de 1's por 0's por ejemplo) ocasionados por ruido, falsos contactos, etc., lo cual puede llegar a alterarla de manera significativa.

Ahora bien, aunque la posibilidad de que un error de transmisión ocurra es pequeña, dada la eficiencia cada vez mayor de los equipos que se construyen, ha sido necesario crear códigos cuya finalidad sea permitir la fácil detección de dichas fallas. Códigos con estas características son por ejemplo, el llamado bit de paridad, el "shift-counter", etc., que a continuación se describen.

Bit de paridad.- El bit de paridad es un bit adicional que se agrega a la información en cuestión, sea cual fuere el código en el que ésta se encuentre representada. La finalidad de este bit extra se revela en las dos posibilidades que presenta y que se llaman paridad-impar y paridad-par, según el caso.

En el primer caso, esto es, cuando se usa la paridad par, se tiene que el valor que se le asigna a este bit de paridad (0 ó 1) se elige de tal manera que la cantidad de 1's que posee un grupo de bits cualquiera, incluyendo al mismo bit de paridad, sea par. Así por ejemplo, si se tiene cierta información representada por el conjunto de bits 1001001, es decir, por un conjunto que en total contiene tres 1's, el bit de paridad tomará el valor de 1 a fin de lograr que el total de 1's del conjunto sea par (i.e.,4), es decir:

```
   1      1 0 0 1 0 0 1
  ---   ---------------
   |          |
   V          V
  bit    conjunto de bits
  de     representan cierta
paridad  información
```

De la misma manera, si lo que se tiene es el conjunto 1000100, el bit de paridad tomará el valor de cero, pues el No de 1's es 2,i.e.:

```
   0     1 0 0 0 1 0 0
  ---   --------------
   |          |
   V          V
  bit     conjunto de bits
   de
paridad
```

Así, por ejemplo, si se está usando la paridad-par en una transmisión, y el receptor de ésta detecta un número impar de 1's, se hará entonces evidente la ocurrencia de una falla, equivalente a la alteración de por lo menos un bit. Claro está, si el número de errores que ocurre es par (2, 4, 16, etc.) el bit de paridad-par no resultará útil. Similarmente se usa la paridad-impar, en cuyo caso el error se detecta al encontrarse un total de 1's par.

Código de más de 4 bits.- Anteriormente se han mencionado códigos BCD que contienen 4 bits, sin embargo, también existen códigos con 5, 6 o más bits. Dichos códigos pertenecen también al grupo de aquéllos cuya finalidad consiste en facilitar la detección de errores en una transmisión. A manera de ejemplo, algunos de estos códigos de 5 bits se muestran en la tabla 3.7.

DECIMAL	2 OUT OF 5	SHIFT COUNTER	51111	
0	00011	00000	00000	
1	00101	00001	00001	
2	00110	00011	00011	
3	01001	00111	00111	
4	01010	01111	01111	Código BCD de 5 bits
5	01100	11111	10000	
6	10001	11110	11000	
7	10010	11100	11100	
8	10100	11000	11110	
9	11000	10000	11111	

Tabla 3.7 Códigos de más de 4 bits.

Cada uno de estos códigos posee características propias que lo hacen en menor o mayor medida eficiente, según, claro, las necesidades existentes. Por ejemplo, el código 2 out of 5 tiene la particularidad de poseer paridad par, determinada por el hecho de que cada conjunto de bits presenta únicamente dos 1's. Este código ha sido muy usado en comunicaciones, principalmente telefónicas.

Ahora bien, como ya se habrá notado, cuanto más bits posea un código, más eficiente será para la detección de errores. El por qué de esto es fácil de ver según se muestra en el siguiente ejemplo: supóngase que se tiene un código de 5 bits, de tal modo que se pueden formar 32 distintas combinaciones posibles. Si de ellas se usan solamente 10, entonces, en caso de que haya un error durante la transmisión, la probabilidad de que éste resulte ser una de las combinaciones "no válidas" es grande. De esta manera, un código de 10 bits resultaría muy útil, aunque, claro está, a costa de una mayor circuitería. En la tabla 3.8 se muestran algunos códigos de más de 5 bits.

DECIMAL	50 43210	9876543210	
0	01 00001	0000000001	
1	01 00010	0000000010	
2	01 00100	0000000100	
3	01 01000	0000001000	Codigos de
4	01 10000	0000010000	más de
5	10 00001	0000100000	5 bits
6	10 00010	0001000000	
7	10 00100	0010000000	
8	10 01000	0100000000	
9	10 10000	1000000000	

Tabla 3.8 Códigos de más de 5 bits.

Cada uno de estos códigos es un código posicional o ponderado, es decir, a cada número decimal le corresponde una única combinación de 1's y 0's, de modo que cualquier otra no será válida, aún si la suma de los factores de peso proporcionan el número correcto. Esto es equivalente a decir que, por ejemplo, si en el código 5043210 el 5 se representa como la combinación 1000001, la ocurrencia de error se hará evidente si en lugar de dicho arreglo aparece el siguiente: 0110010, que si bien representa un 5 tomando en cuenta los factores de peso, no está consederando en el codigo de la tabla 3.8. Al codigo 50 43210 se le denomina también Biquinario, y tiene la peculiaridad de dividirse en 2 grupos de bits: el primero con los factores de peso 5 y 0, que indican cuándo la cuenta es mayor que 5; y el segundo, con los factores de peso 43210, que lleva la cuenta propiamente dicha.

El otro código representado en la tabla, el 9876543210, resulta sencillo y muy útil, ya que cada conjuto de bits representado en este código posee un único 1, por lo que un error se detecta muy fácilmente.

Códigos alfanuméricos.- Hasta ahora se había usado la notación binaria para representar números únicamente; sin embargo, es usual encontrar que diversos sistemas digitales requieran manejar letras o caracteres especiales (+, %, $, ...). A los códigos que se usan para representar este tipo de información se les denomina alfanuméricos. El código más sencillo es el llamado código interno de 6 bits, que se usa para representar el alfabeto inglés en un solo tipo de letra (mayúsculas usualmente), y los 10 dígitos decimales.

Si además de esto se quiere representar letras minúsculas y diversos caracteres ortográficos como el acento, la coma, etc., es necesario usar al menos 7 bits, pues esto dará la posibilidad de representar 128 caracteres. Este objetivo lo cumple el llamado código ASCII (American Standard Code for Information Interchange).

Existen también códigos de 8 bits (el octavo es el de paridad) como el EBCDIC (Extended BCD Interchange Code).

En la tabla 3.9 se muestran los códigos mencionados.

A manera de comentario final sobre el tema de códigos, se quiere dejar claro que se pueden estructurar una infinidad de códigos, cada uno con características propias que lo hagan eficiente para la tarea específica para la que fue creado. Las necesidades que un código suple son las que determinan sus propiedades: paridad, extensión, cantidad de bits, combinaciones válidas, etc., de tal forma que un código creado para la detección de errores, difícilmente podrá ser usado con otros fines (como el de facilitar algún tipo de cálculo aritmético por ejemplo).

CARACTER	CODIGO INTERNO 6-bits	CODIGO ASCII 7-bits	CODIGO EBCDIC 8-bits
A	010 001	100 0001	1100 0001
B	010 010	100 0010	1100 0010
C	010 011	100 0011	1100 0011
D	010 100	100 0100	1100 0100
E	010 110	100 0101	1100 0101
F	010 110	100 0110	1100 0110
G	010 111	100 0111	1100 0111
H	011 000	100 1000	1100 1001
I	011 001	100 1001	1100 1001
J	100 001	100 1010	1100 0001
K	100 010	100 1011	1100 0010
L	100 011	100 1100	1100 0011
M	100 100	100 1101	1100 0100
N	100 101	100 1110	1100 0101
O	100 110	100 1111	1100 0110
P	100 111	100 0000	1100 0111
Q	101 000	100 0001	1100 1000
R	101 001	100 0010	1100 1001

Tabla 3.9 Códigos Alfanuméricos.

CARACTER	CODIGO INTERNO 6-bits	CODIGO ASCII 7-bits	CODIGO EBCDIC 8-bits
S	110 010	100 0011	1100 0010
T	110 011	100 0100	1100 0011
U	110 100	100 0101	1100 0100
V	110 101	100 0110	1100 0101
W	110 110	100 0111	1100 0110
X	110 111	100 1000	1100 0111
Y	111 000	100 1001	1100 1000
Z	111 0001	100 1010	1100 1001
0	000 000	011 0000	1111 0000
1	000 001	011 0001	1111 0001
2	000 010	011 0010	1111 0010
3	000 011	011 0011	1111 0011
4	000 100	011 0100	1111 0100
5	000 101	011 0101	1111 0101
6	000 110	011 0110	1111 0110
7	000 111	011 0111	1111 0111
8	001 000	011 1000	1111 1000
9	001 001	011 1001	1111 1001
blanco	110 000	010 0000	0100 0000
.	011 011	010 1110	0100 1011
(	111 100	010 1000	0100 1101
+	010 000	010 1011	0100 1110
$	101 011	010 0100	0101 1011
*	101 100	010 1010	0101 1100
)	011 100	010 1001	0101 1101
-	100 000	010 1101	0110 0000
/	110 001	010 1111	0110 0001
,	111 011	010 1100	0110 1011
=	001 011	011 1101	0111 1110

Tabla 3.9 Cont.

C A P I T U L O IV

ALGEBRA Y FUNCIONES BOOLEANAS

ALGEBRA Y FUNCIONES BOOLEANAS

En 1854, Gorge Boole introduce una lógica bivalente que sería la base para el desarrollo del álgebra booleana, herramienta básica en el diseño y análisis de circuitos digitales, que será motivo de estudio en las siguientes secciones.

Algebra de Boole.- Para definir el Algebra booleana es necesario definir un conjunto de elementos, los operadores que actúan sobre estos y las reglas básicas que los rigen.

Así pues, se define un conjunto B de elementos sobre los cuales actuan los operadores "+" (suma) y "." (producto)*, y que obedecen los siguientes postulados, enunciados en 1904 por Huntington:

1) Los elementos del Conjunto B están suscritos a una relación de equivalencia denotada con el simbolo "=", que satisface el principio de "sustitución", es decir, que si X, Y$\in$B, entonces X=Y implica que puede substituirse X por Y y viceversa en cualquier expresión álgebraica que las contenga.

2) Cerradura respecto de ambos operadores: si X, Y$\in$ B, entonces X + Y $\in$ B, y XY $\in$ B.

3) Conmutatividad de ambos operadores: si X, Y$\in$ B, entonces X + Y = Y + X y YX = YX.

4) Existe un elemento llamado identidad con respecto de cada uno de los operadores:

   Para la suma, la identidad es el 0 y es tal que

$$X + 0 = 0 + X = X$$

   Para el producto, la identidad es el 1 y es tal que

$$X\ 1 = 1X = X$$

5) Distributividad de un operador sobre el otro: Si X, Y, Z $\in$ B, entonces

$$X\ (Y + Z) = (XY) + (XZ)$$

y

$$X + (YZ) = (X + Y)\ (X + Z)$$

---

* En general, y por sencillez se suele omitir el punto, sobreentendiéndose el producto.

6) Para todo elemento $X \in B$, existe un elemento $\bar{X}$ (también denotado X') llamado su complemento tal que

$$X + \bar{X} = 1 \qquad y \qquad X\bar{X} = 0$$

7) Existen al menos 2 elementos en B, X, Y tales que $X \neq Y$.

Aunque el conjunto de postulados no lo contempla, los elementos de B aceptan reglas de asociatividad; no así, reglas para la división o la resta.

Ahora bien, pueden existir diversos conjuntos que satisfagan por completo los postulados anteriores, ya que no se especifica en ningún momento el tipo de elementos de dicho conjunto. De este modo, dado que el interès se centra en el tratamiento de circuitos que manejan señales de sólo dos posibles valores, a continuación se tratará el caso más simple del álgebra de Boole, donde el conjunto B está constituido por sólo dos elementos: el 0 y el 1.

Debe quedar claro que, si bien el conjunto B cuenta con dos elementos únicamente, la cantidad de variables booleanas es indefinida, o dicho de otro modo, cualquier variable booleana X, Y, Z, S, T, ..., puede tomar sólo los valores 0 ó 1, o bien un arreglo de ambos.

Principio de dualidad.- De entre las más importantes propiedades exhibidas por el álgebra de Boole existe una, llamada principio de dualidad, que establece que toda expresión algebraica obtenida a partir de los teoremas y postulados, sigue siendo cierta cuando se intercambia los operadores, "+" por "." , los 0's por 1's y viceversa, con lo cual se dice que se ha obtenido el dual de la expresión original. Así por ejemplo, si se tiene que $X + XY = X$, se obtendrá su dual cambiando los operadores como sigue:

$$\begin{array}{l} X + X\,Y = X \\ \quad\;\; \downarrow \quad \downarrow \\ \quad\;\; \vee \quad \vee \\ X\;(X + Y) = X \end{array}$$

A continuación se presenta una lista que resume los postulados y teoremas básicos del álgebra booleana.

POSTULADOS.

1) $X + 0 = X$

2) $X \cdot 1 = X$

3) $X \cdot \overline{X} = 0$

4) $X + \overline{X} = 1$

5) $X + Y = Y + X$ — conmutatividad

6) $X \cdot Y = X \cdot Y$ — conmutatividad

7) $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$ — distributividad

8) $X + Y \cdot Z = (X + Y) \cdot (X + Z)$ — distributividad

TEOREMAS.

1) $X + X = X$

2) $X \cdot X = X$

3) $X + 1 = 1$

4) $X \cdot 0 = 0$

5) $\overline{(\overline{X})} = X$ involución

6) $X + (Y + Z) = (X + Y) + Z$ — asociatividad

7) $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$ — asociatividad

8) $X + X \cdot Y = X$ — absorción

9) $X \cdot (X + Y) = X$ — absorción

10) $\overline{(X + Y)} = \overline{X} \cdot \overline{Y}$ — de De Morgan

11) $\overline{X \cdot Y} = \overline{X} + \overline{Y}$ — de De Morgan

Ahora bien, si se quiere estar seguro de la validez de cada uno de los teoremas anteriores, se puede probar haciendo uso de los postulados ya antes expuestos.

Así por ejemplo, se probará que $X + \bar{X} = Y$.

$$X + \bar{X} = (X + Y)\ 1$$
$$= (X + Y)(X + \bar{X})$$
$$= X + X\bar{X}$$
$$= X$$

$$\therefore\ X + \bar{X} = X$$

QED

O bien, probemos que $X + XY = X$:

$$X + XY = X\ 1 + XY$$
$$= X\ (Y + \bar{Y}) + XY$$
$$= XY + X\bar{Y} + XY$$
$$= XY + X\bar{Y}$$
$$= X\ (Y + \bar{Y}) = X\ 1$$
$$= X$$

$$\therefore\ X + XY = X$$

QED

Similarmente, se pueden probar los teoremas restantes; sin embargo, para comprobar los de De Morgan se evaluarán las expresiones algebraicas para cada uno de todos los casos posibles de valores de X, Y, verificando que en ninguno de ellos se llegue a una contradicción.

Se verá pues el primero de estos teoremas:

i) Caso en que $X = 0$, $Y = 0$.

Evaluando la expresión tenemos que:

$$\overline{(X + Y)} = \overline{(0 + 0)} = \bar{0} = 1 \quad y \quad \bar{X}\,\bar{Y} = \bar{0}\,\bar{0} = 11 = 1$$

y por tanto se cumple la igualdad.

ii) Caso X = 0, Y = 1.

$\overline{(X + Y)} = \overline{(0 + 1)} = \overline{1} = 0$ y $\overline{X}\,\overline{Y} = \overline{0}\,\overline{1} = 10 = 0$

y la igualdad se cumple.

iii) Caso X = 1, Y = 0.

$\overline{(X + Y)} = \overline{(1 + 0)} = \overline{1} = 0$ y $\overline{X}\,\overline{Y} = \overline{1}\,\overline{0} = 01 = 0$

y la igualdad se cumple.

iv) Caso X = 1, Y = 1.

$\overline{(X + Y)} = \overline{(1 + 1)} = \overline{1} = 0$ y $\overline{X}\,\overline{Y} = \overline{1}\,\overline{1} = 00 = 0$

y la igualdad se cumple.

Y por tanto, revisados todos los casos, puede afirmarse que

$\overline{(X + Y)} = \overline{X}\,\overline{Y}$, para el álgebra donde b=(0,1)

Como se observará, el proceso de verificar el teorema probando cada una de las posibilidades para X, Y es largo y puede resultar tedioso. Sin embargo, una vez que se ha hecho el trabajo se pueden resumir los resultados obtenidos en la tabla de la figura 4.1.

X Y	$\overline{X}\,\overline{Y}$	X + Y	$\overline{X + Y}$	$\overline{X}\,\overline{Y}$
0 0	1 1	0	1	1
0 1	1 0	1	0	0
1 0	0 1	1	0	0
1 1	0 0	1	0	0

Fig. 4.1 Tabla de verdad.

A la tabla de la Fig. 4.1, que muestra todos los posibles valores de las variables de la expresión booleana, se le denomina Tabla de Verdad.

En la tabla de la figura 4.1 se puede ver con claridad que

las últimas dos columnas, correspondientes a $\bar{X} + \bar{Y}$ y $\overline{XY}$, resultan ser iguales. Esto es indicativo de que se puede usar una tabla de verdad no solamente para mostrar toda la gama de posibilidades que existen para las variables y la expresión algebraica de las mismas, sino también como una comprobación de la veracidad de cualesquier expresión booleana. Así por ejemplo, se usará una tabla de verdad para comprobar el segundo teorema de De Morgan:

$$\overline{X\,Y} = \bar{X} + \bar{Y}$$

X	Y	$\bar{X}$	$\bar{Y}$	XY	$\bar{X} + \bar{Y}$	$\overline{X\,Y}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

y, dado que las dos últimas columnas son iguales, resulta inmediato establecer que:

$$\overline{XY} = \bar{X} + \bar{Y}$$

Cabe aquí mencionar que tanto los teoremas y postulados como las tablas de verdad, son útiles además en la simplificación de expresiones booleanas de una función, lo cual se verá en una sección posterior. Por lo pronto se presenta un último ejemplo de verificación, en el cual se ilustran los dos métodos mencionados.

Sea la expresión $X\,(X + Y) = X$.

a) Comprobación algebraica.

$$\begin{aligned} X\,(X + Y) &= X\,(X + Y)\,1 \\ &= X\,(X + Y)\,(X + \bar{X}) \\ &= (XX)X + (XX)Y + (X\bar{X})X + (X\bar{X})Y \\ &= X + XY + 0X + 0Y \\ &= X + XY \\ &= X \qquad \text{(absorción)} \end{aligned}$$

$$\therefore X\,(X + Y) = X$$

QED

b) Tabla de verdad.

X	Y	X + Y	X (X + Y)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

$\therefore X = X(X + Y)$

QED /

Ahora bien, quedan por mencionar, entre otros, algunos teoremas que se derivan de los 11 expuestos, y debido a su gran utilidad se muestran a continuación:

a) $X + \bar{X} Y = X + Y$

b) $X (\bar{X} + Y) = XY$

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

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

Comprobación de a) y d):

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

$= X Y + X \bar{Y} + \bar{X}YX + \bar{X}Y\bar{X}$

$= XY + X\bar{Y} + \bar{X}Y$

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

$= X + Y$

$\therefore X + \bar{X}Y = X + Y$

QED /

d)

X	Y	Z	$\bar{X}$	$\bar{X}$ Y	$\bar{X}$ Z	$\bar{X}$ + Y	X + Z	(X + Z) ($\bar{X}$ + Y)	XY + XZ
0	0	0	1	0	0	1	0	0	0
0	0	1	1	0	1	1	1	1	1
0	1	0	1	0	0	1	0	0	0
0	1	1	1	0	1	1	1	1	1
1	0	0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	0	0
1	1	0	0	1	0	1	1	1	1
1	1	1	0	1	0	1	1	1	1

Funciones booleanas.- Cuando se dice por ejemplo que:

> La alarma sonará si la puerta está abierta
> y alguna persona entra.

se está estableciendo una relación de dependencia tal, que el hecho de que suene la alarma se encuentra "en función" de que la puerta esté abierta y de que alguna persona entre, es decir, la proposición "la larma sonará", es verdadera si las otras dos lo son.

Este tipo de argumentación lógica es precisamente la que va a permitir definir una función booleana: si se tiene un número n (entero) cualquiera de variables binarias, que pueden o no estar completamente relacionadas entre sí mediante los operadores " + " y " . ", de modo que alguna de ellas dependa de las otras, se dice que se tiene una función booleana.

Si, por ejemplo, se tiene a las variables Y, X, T, tales que Y = XT, se dice que Y está en función de X y T, lo cual se puede indicar como Y = f(X, T), donde f(X,T) = XT.

En ese caso, la calidad de verdaddera o falsa de la función dependerá de cuándo sea verdadera o falsa la expresión boolenana XT.

Por ejemplo, si la variable X=1 indica que la puerta está abierta, T=1 que entre alguna persona y la función f(X,T)=1 que sonará la alarma, entonces la tabla de verdad de la figura 4.2 muestra su comportamiento.

X T	f(X, T) = XT
0 0	0
0 1	0
1 0	0
1 1	1

Fig. 4.2 Tabla de verdad para una alarma.

ii) Caso X = 0, Y = 1.

$$(\overline{X + Y}) = (\overline{0 + 1}) = \overline{1} = 0 \quad y \quad \overline{X}\,\overline{Y} = \overline{0}\,\overline{1} = 10 = 0$$

y la igualdad se cumple.

iii) Caso X = 1, Y = 0.

$$(\overline{X + Y}) = (\overline{1 + 0}) = \overline{1} = 0 \quad y \quad \overline{X}\,\overline{Y} = \overline{1}\,\overline{0} = 01 = 0$$

y la igualdad se cumple.

iv) Caso X = 1, Y = 1.

$$(\overline{X + Y}) = (\overline{1 + 1}) = \overline{1} = 0 \quad y \quad \overline{X}\,\overline{Y} = \overline{1}\,\overline{1} = 00 = 0$$

y la igualdad se cumple.

Y por tanto, revisados todos los casos, puede afirmarse que

$$(\overline{X + Y}) = \overline{X}\,\overline{Y}\ , \text{ para el Álgebra donde } b=(0,1)$$

Como se observará, el proceso de verificar el teorema probando cada una de las posibilidades para X, Y es largo y puede resultar tedioso. Sin embargo, una vez que se ha hecho el trabajo se pueden resumir los resultados obtenidos en la tabla de la figura 4.1.

X Y	$\overline{X}\,\overline{Y}$	X + Y	$\overline{X + Y}$	$\overline{X}\,\overline{Y}$
0 0	1 1	0	1	1
0 1	1 0	1	0	0
1 0	0 1	1	0	0
1 1	0 0	1	0	0

Fig. 4.1 Tabla de verdad.

A la tabla de la Fig. 4.1, que muestra todos los posibles valores de las variables de la expresión booleana, se le denomina Tabla de Verdad.

En la tabla de la figura 4.1 se puede ver con claridad que

las últimas dos columnas, correspondientes a $\bar{X} + \bar{Y}$ y $\overline{XY}$, resultan ser iguales. Esto es indicativo de que se puede usar una tabla de verdad no solamente para mostrar toda la gama de posibilidades que existen para las variables y la expresión algebraica de las mismas, sino también como una comprobación de la veracidad de cualesquier expresión booleana. Así por ejemplo, se usará una tabla de verdad para comprobar el segundo teorema de De Morgan:

$$\overline{X\,Y} = \bar{X} + \bar{Y}$$

X Y	$\bar{X}\,\bar{Y}$	XY	$\bar{X} + \bar{Y}$	$\overline{X\,Y}$
0 0	1 1	0	1	1
0 1	1 0	0	1	1
1 0	0 1	0	1	1
1 1	0 0	1	0	0

y, dado que las dos últimas columnas son iguales, resulta inmediato establecer que:

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

Cabe aquí mencionar que tanto los teoremas y postulados como las tablas de verdad, son útiles además en la simplificación de expresiones booleanas de una función, lo cual se verá en una sección posterior. Por lo pronto se presenta un último ejemplo de verificación, en el cual se ilustran los dos métodos mencionados.

Sea la expresión $X\,(X + Y) = X$.

a) Comprobación algebraica.

$$\begin{aligned} X\,(X + Y) &= X\,(X + Y)\,1 \\ &= X\,(X + Y)\,(X + \bar{X}) \\ &= (XX)X + (X\bar{X})Y + (X\bar{X})X + (X\bar{X})Y \\ &= X + XY + 0X + 0Y \\ &= X + XY \\ &= X \qquad \text{(absorción)} \end{aligned}$$

$$\therefore X\,(X + Y) = X$$

QED

b) Tabla de verdad.

X Y	X + Y	X (X + Y)
0 0	0	0
0 1	1	0
1 0	1	1
1 1	1	1

$$\therefore X = (X + Y)$$

QED

Ahora bien, quedan por mencionar, entre otros, algunos teoremas que se derivan de los 11 expuestos, y debido a su gran utilidad se muestran a continuación:

a) $X + \bar{X} Y = X + Y$

b) $X (\bar{X} + Y) = XY$

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

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

Comprobación de a) y d):

a) $$X + \bar{X}Y = X(Y + \bar{Y}) + \bar{X}Y(X + \bar{X})$$
$$= X Y + X \bar{Y} + \bar{X}YX + \bar{X}Y\bar{X}$$
$$= XY + X\bar{Y} + \bar{X}Y$$
$$= X(Y + \bar{Y}) + Y(X + \bar{X})$$
$$= X + Y$$

$$\therefore X + \bar{X}Y = X + Y$$

QED

d)

X	Y	Z	$\bar{X}$	$\bar{X}$Y	$\bar{X}$Z	$\bar{Y}$ + Y	Y + Z	($\bar{X}$ + Z)	($\bar{X}$ + Y)	$\bar{X}$Y + X$\bar{Z}$
0	0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	1	0	0	0
1	1	0	0	1	0	1	1	1	1	1
1	1	1	0	1	0	1	1	1	1	1

Funciones booleanas.- Cuando se dice por ejemplo que:

> La alarma sonará si la puerta está abierta
> y alguna persona entra.

se está estableciendo una relación de dependencia tal, que el hecho de que suene la alarma se encuentra "en función" de que la puerta esté abierta y de que alguna persona entre, es decir, la proposición "la larma sonará", es verdadera si las otras dos lo son.

Este tipo de argumentación lógica es precisamente la que va a permitir definir una función booleana: si se tiene un número n (entero) cualquiera de variables binarias, que pueden o no estar completamente relacionadas entre sí mediante los operadores " + " y " . ", de modo que alguna de ellas dependa de las otras, se dice que se tiene una función booleana.

Si, por ejemplo, se tiene a las variables Y, X, T, tales que Y = XT, se dice que Y está en función de X y T, lo cual se puede indicar como Y = f(X, T), donde f(X,T) = XT.

En ese caso, la calidad de verdaddera o falsa de la función dependerá de cuándo sea verdadera o falsa la expresión boolenana XT.

Por ejemplo, si la variable X=1 indica que la puerta está abierta, T=1 que entre alguna persona y la función f(X,T)=1 que sonará la alarma, entonces la tabla de verdad de la figura 4.2 muestra su comportamiento.

X	T	f(X, T) = XT
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 4.2 Tabla de verdad para una alarma.

como resulta evidente, f(X,T) = 1 cuando X = 1 y T = 1.

Así mismo, si se tiene por ejemplo la función $f(X,Y,Z) = XY + \bar{Z} + XZ$, se tendrá que, de la tabla de verdad,

X	Y	Z	$\bar{Z}$	XY	XZ	$f(X,Y,Z) = XY + \bar{Z} + XZ$
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	0	0	0
1	0	0	1	0	0	1
1	0	1	0	0	1	1
1	1	0	1	1	0	1
1	1	1	0	1	1	1

la función es verdadera cuando X=Y=1 ó X=Z=1 ó Z=0 .

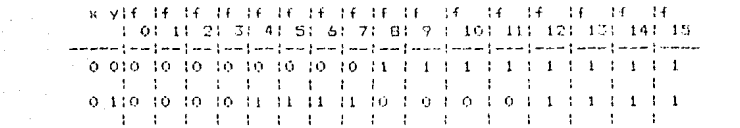
Ahora bien, cuando se tienen n variables se puede tener 2^{2^n} funciones distintas. Así, por ejemplo, si n=1 existen 4 funciones posibles"

x	f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1

o, lo que es lo mismo, para una sola variable, por ejemplo x, existen 4 funciones:

$$f_1(x) = 0, \quad f_2(x) = X, \quad f_3(x) = \bar{x}, \quad f_4(x) = 1$$

Igualmente se pueden encontrar las 2^{2^n} funciones para n=3 ($2^{2^3} = 256$), n=4 ($2^{2^4} = 65536$), etc., y de entre ellas, el caso n=2, ($2^{2^2} = 16$) resulta ser especial, ya que a las 16 funciones que existen se les suele dar un nombre y símbolo propios. Se muestra la tabla de verdad en la figura 4.3 que presenta estas 16 posibilidades.



f_0 = 0, función nula o antitautología.

f_1 = XY, AND (X y Y)

f_2 = $X\bar{Y}$ = X/Y, inhibición (X pero no Y)

f_3 = X

f_4 = $\bar{X}Y$ = Y/X, inhibición (Y pero no X)

f_5 = Y

f_6 = $\bar{X}Y + X\bar{Y}$ = X ⊕ Y, OR exclusivo

f_7 = X + Y, OR (X o Y)

f_8 = $\overline{(X + Y)}$ = X↓Y, NOR

f_9 = $\bar{X}\,\bar{Y} + XY$ = XOY, equivalencia o bicondicional (X = Y)

f_{10} = $\bar{Y}$, complemento (no Y)

f_{11} = $X + \bar{Y}$, X <== Y. implicación (si Y entonces X)

Fig. 4.3 Tabla de verdad y nombres para las posibles funciones de 2 variables.

f = $\bar{X}$, complemento (no X)
12

f = $\bar{X}$ + Y , X==> Y, implicación (si X entonces Y)
13

f = $(\overline{X\ Y})$ = X↑Y, NAND
14

f = 1, función identidad o tautología
15

Fig. 4.3 Cont.

Así,en el ejemplo representado en la figura 4.2 se tendrá que la función que describe el comportamiento de la alarma es:

$$f = XT$$

ó

$$f = X \text{ and } T$$

Complemento de una función.- Al igual que se complementa (niega) una variable, es posible complementar una función. Esto puede hacerse de dos maneras: se pueden intercambiar 1's por 0's y 0's por 1's en la columna correspondiente a la función en la tabla de verdad, o se puede complementar algebraicamente haciendo uso de los teoremas de De Morgan.

Así, por ejemplo, si tenemos la función f(x,y,z) = xy + xz + yz, podemos obtener su complemento, que se denota f'(x,y,z) como sigue:

a) Algebraicamente.

$$f'(x,y,z) = \overline{xy + xz + yz} = \overline{(xy)}\ \ \overline{(xz)}\ \overline{(yz)}$$

$$= (\bar{x} + \bar{y})\ (\bar{x} + \bar{z})\ (\bar{y} + \bar{z})$$

$$= (\bar{x}\,\bar{x} + \bar{x}\,\bar{z} + \bar{x}\,\bar{y} + \bar{y}\,\bar{z})\ (\bar{y} + \bar{z})$$

$$= (\bar{x} + \bar{x}\ (\bar{z} + \bar{y}) + \bar{y}\,\bar{z})\ (\bar{y} + \bar{z})$$

$$= (\bar{x} + \bar{y}\bar{z})\ (\bar{y} + \bar{z})$$

$$= \bar{x}\,\bar{y} + \bar{x}\,\bar{z} + \bar{y}\,\bar{z}$$

b) Mediante la tabla de verdad.

x	y	z	f	$\bar{f}$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

de donde $\bar{f}\ (x,y,z) = \bar{x}\ \bar{y} + \bar{y}\ \bar{z} + \bar{x}\ \bar{z}$.

C A P I T U L O V

IMPLEMENTACION DE FUNCIONES BOOLEANAS

LOGICA MIXTA

En este capitulo se abodarà el tema de implementaciòn de funciones desde dos puntos de vista: la " lògica mixta " y la "lògica positiva ". Actualmente no existe un criterio definitivo para establecer que alguna se ellas sea mejor que la otra, esto depende de la preferencia del diseñador. En la mayoria de los textos se trata la lògica positiva ampliamente, es por esto que aqui se ha dado màs èmfasis a la lògica mixta.

Se ha visto ya que las variables booleanas pueden tomar sòlo dos posibles valores: verdadero y falso (V y F ò, 1 y 0); y que los dispositivos fisicos que manejan señales digitales utilizan dos niveles de voltaje: estado alto (H) y estado bajo (L).

Lo que ahora se necesita es establecer una equivalencia entre los valores lògicos de una variable y los niveles de voltaje.

Existen dos posibles maneras de establecer dicha equivalencia: se puede convenir en asignar la calidad de verdadero al nivel de voltaje alto (V=H) -en cuyo caso se dirà que se està trabajando con lògica positiva- designando a las variables como verificadas altas; si por el contrario, se asigna la calidad de verdadero al nivel de voltaje bajo (V=L), se dirà que se està usando una lògica negativa, y en consecuencia a las variables se les llamarà verificadas bajas.

Anteriormente, un diseñador adoptaba, en general, una de estas lògicas -casi siempre la positiva- y manteniendo constante su convenciòn realizaba el diseño del dispositivo lògico. Por esto mismo es que la lògica positiva llegò a ser la pieza fundamental en el diseño de sistemas digitales.

Actualmente existe otra opciòn. la denominada lògica mixta, que permite "mezclar" el uso de las lògicas positiva y negativa en un mismo diseño. Este concepto, según algunos autores, constituye un mètodo poderoso y elegante para el diseño de circuitos. Proporcionando grandes ventajas en cuanto a la mayor facilidad con que con ella se puede crear, para un mismo diseño, diferentes configuraciones, sin tener la necesidad de enredarse en un tratamiento algebraico.

Notaciòn en lògica mixta.- La notaciòn en lògica mixta debe ser tal que cumpla con dos objetivos bàsicos:

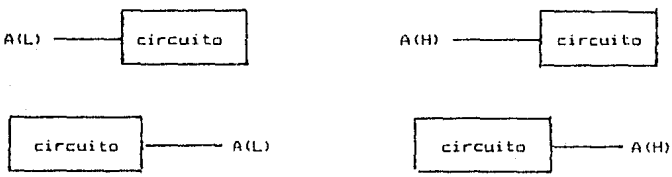
1) Que aunque el diagrama del diseño sea elaborado por un conocedor de lògica mixta, pueda también ser interpretado en tèrminos de la lògica positiva o negativa.
2) Debe quedar clara, en cualquier punto del circuito, la relaciòn entre los niveles de voltaje (H y L) y los valores lògicos de las variables (V y F).

Con el fin de satisfacer estos objetivos se debe establecer ciertas convenciones: si en algún punto del circuito se utiliza lògica negativa (V=L), se deberà dibujar un pequeño circulo en las entradas y salidas del simbolo lògico empleado: la ausencia de dicho circulo indicarà que en este punto las variables son verificadas altas (V=H). A este circulo se le suele denominar "indicador de nivel", "dot", "ninny" o "bubble" (burbuja, en inglès).

Ahora bien, cuando se da nombre a las entradas y salidas de un circuito, se debe especificar tambièn si la variable es verificada alta o baja. Para realizar esto existen diversas convenciones, de las cuales se habrà de utilizar una sola de ellas en el presente texto, misma que a continuaciòn se presenta:

Si la variable es verificada baja, se agrega al final del nombre de dicha variable la letra L dentro de parèntesis; en caso de que la variable sea verificada alta la letra entre parèntesis serà " H ".

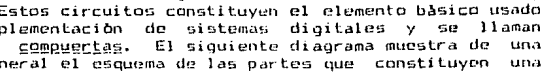
Asi entonces, si se tiene una variable denominada A, se podrà tener las siguientes posibilidades:



Cabe subrayar que H no significa que el voltaje esté en alto, significa, en cambio, que la variable A tomarà el valor lògico de verdadero si el voltaje es de un nivel alto. Lo mismo puede decirse en el caso de L.

DISPOSITIVOS FISICOS PARA LA IMPLEMENTACION DE FUNCIONES

Compuertas lògicas.- Existen diversos circuitos integrados que realizan funciones lògicas bàsicas tales como AND, OR, XOR, etc. Estos circuitos constituyen el elemento bàsico usado en la implementaciòn de sistemas digitales y se llaman comùnmente compuertas. El siguiente diagrama muestra de una manera general el esquema de las partes que constituyen una compuerta.



Las entradas se representan por variables booleanas, en tanto que cada salida por una funciòn; esto es, si a la entrada de la compuerta se tienen las variables x, y, a la salida se tendrà alguna funciòn f(x,y).

Si se tuviera, por ejemplo, la siguiente tabla de verdad:

x(H)	y(H)	f(x, y) = S(H)
0	0	0
0	1	1
1	0	1
1	1	0

Se implementara f(x,y) mediante alguna compuerta, no especificada por ahora. Se tendrìa una situaciòn similar a la mostrada en la Fig. 5.1.

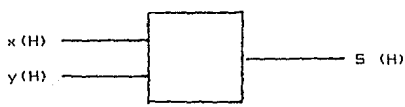


Fig.5.1 Compuerta lògica con salida S y entradas x,y.

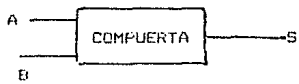
Es decir, se tendrìa que para todos los casos posibles de valores para x, y a la entrada de la compuerta, se obtiene siempre el valor de f(x,y) especificado en la tabla de verdad.

Por supuesto que se pueden conectar diversas compuertas entre sí, lo cual permite la creación de diseños más complejos que realizan funciones de mayor complicación.

A continuación se estudiarán algunas de las compuertas más comunes que están formadas por diversos tipos de circuitos integrados, cada uno de los cuales se identifica mediante un número y un nombre.

El comportamiento de estas compuertas se describe por medio de una tabla electrónica proporcionada por el fabricante, en la cual se indica la relación que existe entre el estado de las entradas y la salida. Un ejemplo de este tipo de tablas se presenta en la Fig. 5.2.

Entradas		Salida
A	B	S
L	L	L
L	H	L
H	L	L
H	H	H



L: Nivel de voltaje bajo

H: Nivel de voltaje alto

Fig.5.2 Ejemplo de tabla electrónica y compuerta.

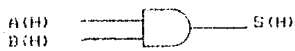
Se intentará ahora la interpretación del primer renglón de la tabla: se puede observar que cuando se presenta un nivel de voltaje bajo en A y B, se obtiene a su vez un nivel de voltaje bajo en la salida S.

Bien, ahora suponga que se ha decidido que todas las variables del diseño han de ser verificadas altas, es decir: A(H), B(H) y S(H), sustituyendo en la Fig. 5.2 los equivalente lógico 0 ó 1 se obtendrá la tabla de la fig. 5.3, que es la tabla de verdad de la función lógica AND, ya antes vista.

Para poder representar esta función diagramáticamente, se utiliza el símbolo de la Fig. 5.4, en donde sus entradas y su salida se consideran verificadas altas, según lo exige el ejemplo.

Entradas		Salida	
A(H)	B(H)	S(H)	
0	0	0	
0	1	0	
1	0	0	====> S(H) = A(H) B(H)
1	1	1	

Fig. 5.3 Tabla de verdad con variables verificadas altas.



representación diagamàtica de la compuerta AND.

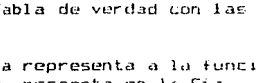
Fig.5.4 Representación de la función de la figura 5.3.

Se cambiará la convención y se supondrá que A, B y S son verificadas bajas, con lo cual se obtiene la tabla de verdad de la Fig.5.5, bajo la suposición de que L=1 y H=0:

Entradas		Salida	
A(L)	B(L)	S(L)	
1	1	1	
1	0	1	
0	1	1	====> S(L) = A(L) + B(L)
0	0	0	

Fig.5.5 Tabla de verdad con las variables verificadas bajas.

Esta tabla representa a la función lògica OR, cuyo simbolo gràfico se presenta en la Fig. 5.6, donde las entradas y salidas son verificadas bajas.



Representación gràfica de la Función OR.

Fig.5.6 Representación de la función de la tabla de la fig. 5.5.

Esta cualidad de la compuerta, la de poder realizar funciones lògicas AND y OR, es una caracterìstica general de todo tipo de compuerta.

La compuerta tratada en el ejemplo anterior se conoce en la familia TTL (Transistor-Transistor-Logic)con el número 7409 y se le denomina simplemente AND.

En la tabla de la figura 5.9 se presenta la tabla electrònica, número, nombre y representaciones AND, y OR de las compuertas màs usuales. Es importante advertir que el lector debe tener sumo cuidado en no confundir el nombre de las compuertas con las funciones lògicas que realizan, ya que èste ha sido adoptado como tal en virtud de la anterior preferencia hacia la lògica positiva.

Se verà ahora algunos ejemplos del uso de las compuertas de la tabla de la Fig.5.9.

EJEMPLO 1.

Sea ahora la función F(H)= c(L) + d(L), para implementar esta función se elige una representación OR para la cual las entradas sean verificadas bajas y la salida verificada alta. De esta manera se encuentra que la compuerta No. 7400 es la màs adecuada (NAND), y de ahì que el esquema del circuito sea el mostrado en la figura 5.7

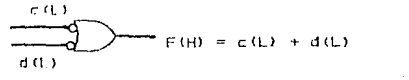
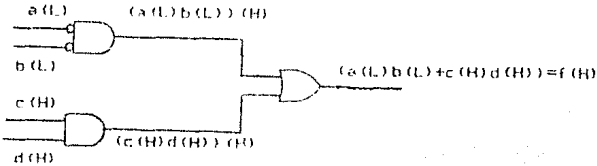


Fig.5.7 Implementación de F(H) = c(L) + d(L)

EJEMPLO 2

Se implementarà ahora la función f(H)=a(L)b(L) + c(H)d(H). En principio, dada la forma de la función, es evidente que se requieren dos compuertas con representación AND y una con representación OR; la que realiza la operación AND con a(L) y b(L) debe tener entradas verificadas bajas, en tanto la que la

realiza con c(H) y d(H) debe tenerlas verificadas altas. Como seguramente el lector ya lo habrá imaginado, no existe una única manera de representar este circuito; en la fig 5.8 se presentan dos posibles maneas de implementarlo.



o bien:

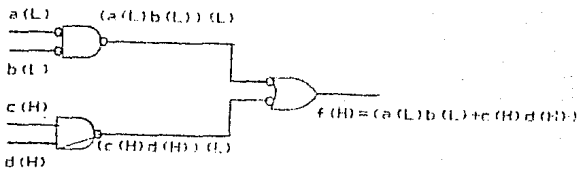
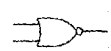
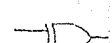


Fig.5.6 Implementación de f(H) =a(L)b(L) + c(H)d(H).

NOMBRE COMERCIAL	NUMERO	TABLA ELECTRONICA			REPRESENTACION AND	REPRESENTACION OR
		ENTRADAS		SALIDA	*	
		X	Y	S		
AND	7408	L	L	L		
		L	H	L		
		H	L	L		
		H	H	H		
OR	7432	X	Y	S		
		L	L	L		
		L	H	H		
		H	L	H		
		H	H	H		
NAND	7400	X	Y	S	*	
		L	L	H		
		L	H	H		
		H	L	H		
		H	H	L		
NOR	7402	X	Y	S		
		L	L	H		
		L	H	L		
		H	L	L		
		H	H	L		
XOR	7486	X	Y	S		
		L	L	L		
		L	H	H		
		H	L	H		
		H	H	L		

* Representación comercial.

Fig.5.9 Compuertas lógicas más usuales.

Función NOT.

Para implementar la función NOT, en lógica Mixta no se requiere de ningún implemento físico adicional a los ya conocidos, por muy extraño que pueda parecer, vease por que:

Supóngase que se tiene una señal denominada A, la cual puede asumir los dos valores de voltaje H y L. En la tabla de la Fig. 5.10 se muestra el comportamiento de la variable booleana asociada a dicha señal en función de su polaridad.

TABLA ELECTRONICA

A	A(H)	A(L)	$\overline{A(H)}$	$\overline{A(L)}$	$\bar{A}(H)$	$\bar{A}(L)$
L	0	1	1	0	1	0
H	1	0	0	1	0	1

Fig.5.10 Comportamiento de la variable booleana A.

Una observación detenida de la tabla anterior muestra que pueden deducirse las siguientes igualdades:

a) $A(H) = \bar{A}(L)$

b) $\overline{A(H)} = A(L)$

c) $\overline{A(H)} = \bar{A}(H)$

d) $A(L) = \bar{A}(H)$

e) $\overline{A(L)} = A(H)$

f) $\overline{A(L)} = \bar{A}(L)$

Con la práctica se verá que los incisos anteriores son de suma importancia dada su gran utilidad en el proceso de diseño con lógica mixta.

Supóngase que se tiene a la variable "A" verificada alta a la entrada de una línea, y que al otro extremo de la misma se encuentra un indicador de nivel, como se muestra en la Fig.5.11, en donde existe una incompatibilidad, ya que no coincide la polaridad de la variable en la línea con la de la entrada de la compuerta; esto es en la entrada se tiene una variable verificada alta mientras que la compuerta espera una verificada baja, lo que se resuelve sustituyendo $A(H)=\bar{A}(L)$ como se representa en la Fig. 5.12.

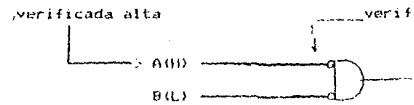


Fig.5.11. Circuito con incompatibilidad lógica.

Nótese que en este caso la lógica ha cambiado, aunque el nivel de voltaje es el mismo a lo largo de toda la línea.

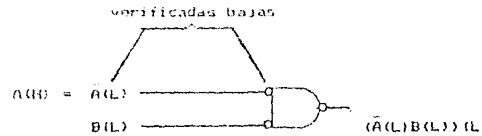


Fig.5.12. Circuito sin incompatibilidad

Para evitar errores en el diseño y/o interpretación de un circuito, se utiliza una diagonal sobre la línea de la variable que se está complementando, diagonal que puede ser interpretada como el símbolo gráfico de la operación NOT como se presenta en la Fig.5.13.

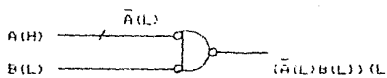


Fig.5.13 Circuito con la variable A negada.

Obviamente, esta operación (NOT=/) es de particular importancia cuando en la función que se va a implementar existen variables negadas.

EJEMPLO 1

Supóngase que se desea diseñar el circuito que a la salida obtenga la función s(H)=a(H)+b(H); se usará una compuerta 7400 y dado que se va a realizar una operación de suma, se adoptará la representación OR de dicha compuerta.

En la figura 5.14, la polaridad en la línea de entrada no es la misma que en la entrada de la compuerta, por lo que la variable a es interpretada como negada ($\bar{a}$).

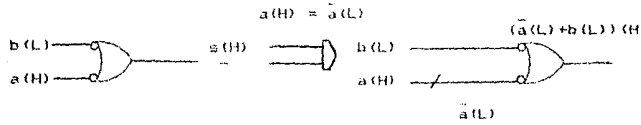


Fig.5.14. Implementación de la función $s(H) = \overline{a(L)}+b(L)$

EJEMPLO 2

Sea ahora el circuito de la figura 5.15, la compatibilidad se presenta a la salida del circuito, ya que de la compuerta sale una función verificada baja, pero se interpretará como verificada alta. Como S(L)=S(H), se sustituye en el circuito teniendo.

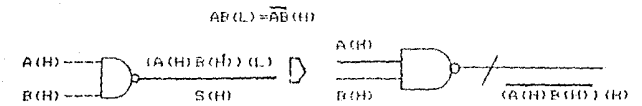


Fig.5.15 implementación de la función $S = \overline{A\bar{B}}$

Inversores de voltaje.- Adicionalmente a las ya mencionadas,existe una compuerta, denominada inversor que, como su nombre lo indica, invierte la polaridad de la variable, es decir, si es verificada alta la cambia a verificada baja y viceversa, posee una única entrada y una sola salida; su representación gráfica y tabla electrónica se presentan en las siguientes figuras.

Entrada	Salida
L	H
H	L

Fig.5.16 Inversor Tabla electrónica

Fig. 5.17 Representaciones gràficas de un inversor.

EJEMPLO 1

Supòngase que se desea implementar la funciòn a=b(H)c(L) y para ello se usarà una compuerta NOR(7402) (en su representaciòn and)

En principio se puede hacer lo siguiente:

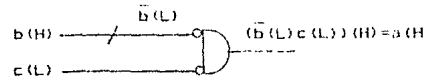


Fig.5.18 Implementaciòn de $\bar{b}c(H)$

Tal como està en la figura, lo que se tendria seria la funciòn $c\bar{b}(H)$, pero a b no se le quiere negada, no se desea cambiar su lògica, ùnicamente su polaridad, razòn por la cual se insertarà un inversor segùn se muestra en la Fig.5.19

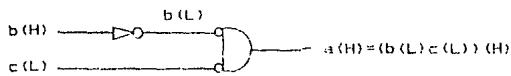


Fig. 5.19 Implementaciòn de a(H) = b(L)c(L)

EJEMPLO 2

Siguiendo el mismo razonamiento, la funciòn c(L)=a(H)+b(L) se puede implementar como se muestra en la Fig.5.20.

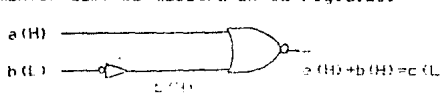


Fig.5.20 Implementaciòn de c(L) = a(H) + b(H).

Puede también darse el caso de que se haga necesario combinar los inversores con la operación NOT (/); como en el siguiente ejemplo.

EJEMPLO 3

En el diseño del circuito cuya salida sea s = $\bar{a}b$ (a(H), b(H) y s(L)), como se muestra en la Fig. 5.21.

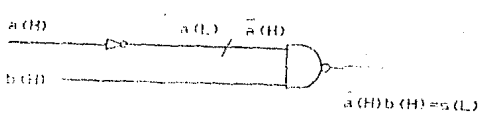
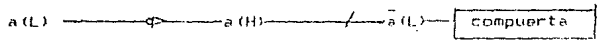


Fig.5.21 Circuito con el operador NOT e inversores.

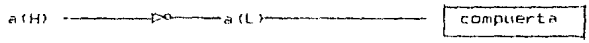
Ahora bien, revisado todos los ejemplos que se han visto, se puede sacar en claro que existen tres casos en los que es necesario incluir un inversor y/o un negador:

1) Complementación de una variable sin cambio de polaridad.

o bien

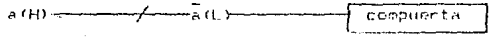


2) Cambio de polaridad sin alteración de la lógica.



o bien

3) Cambio de polaridad y negación de la variable.



IMPLEMENTACION DE FUNCIONES CON LOGICA MIXTA

Según el mismo Paul M. Kinter, quien fuera el primero en publicar formalmente el uso de la lògica mixta, el proceso para realiza un diseño por medio de èsta consta de dos etapas.

Una vez planteado el diseño en tèrminos de funciones lògicas AND y OR, y habiendo decidido què dispositivos fisicos se desean utilizar, se deben seguir los siguientes dos pasos:

1) Sustituir en las funciones AND y OR la representación de ellas que se considere màs conveniente según los dispositivos fisicos de los que se dispogan.

2) Considerar si es o no necesario la inclusión de inversores lògicos y/o negadores.

A continuación se presentan algunos ejemplos.

EJEMPLO 1

Se desea implementar la función $y(L)=a(L)\bar{b}(L)$.

Debido a que la función està planteada en tèrminos de una operación AND, se puede realizar un primer esbozo del diseño en tèrminos de èsta; como se muestra en la fig 5.22.

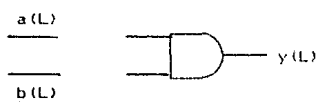


Fig. 5.22. Planteamiento del diseño en la implementación de $y(L)=a(L)\bar{b}(L)$

Se decide ahora usar compuertas NAND, eligiendo la representación AND de la misma por ser la que màs conviene (Fig 5.23)

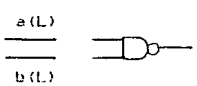


Fig.5.23 1er. Paso en la implementación con las compuertas NAND

Ahora bien, la expresiòn de la función indica que se debe complementar b y cambiarle de lògica; a la variable "a", en cambio, sòlo se necesita cambiar la lògica. De acuerdo a esto, lo que se tiene es el diagrama de la Fig. 5.24.

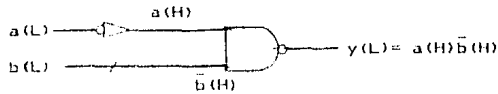


Fig. 5.24 2o. Paso en la implementaciòn de $y(L)=a(H)\bar{b}(H)$

Se verà ahora lo que sucederia si en vez de usar una compuerta NAND se usarà una NOR; en primera, y por conveniencia, se escoge la representación AND de la misma, esto se muestra en la fig. 5.25

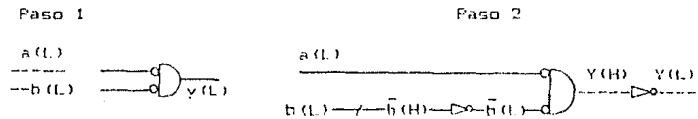


Fig. 5.25 Implementaciòn con la compuerta NOR $Y=A\bar{b}$

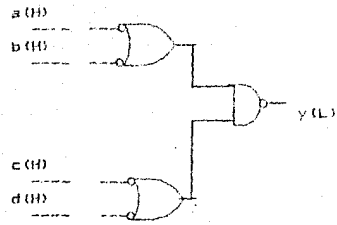
En este caso no se encuentra incompatibilidades y lo único que hay que hacer es complementar b sin cambiar su polaridad Fig.5.26

EJEMPLO 2

Este ejemplo consiste en representar el circuito que realice la función $y(L)=(\bar{a}(H)+\bar{b}(H))(c(H)+d(H))$ donde y es verificada baja.

En este caso se usarán compuertas NAND en sus dos representaciones AND y OR (fig 5.26)

1er. Paso



2o. Paso

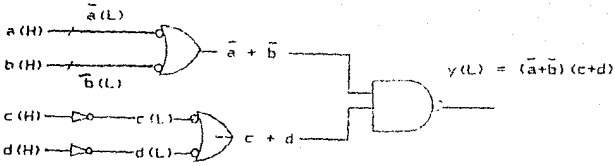


Fig. 5.26 Implementaciòn de $Y=(\bar{a}(L)+\bar{b}(L))(c(L)+d(L))$

Se verà ahora el aspecto que tiene el circuito que implementa esta misma funciòn con compuerta NOR:

1er Paso

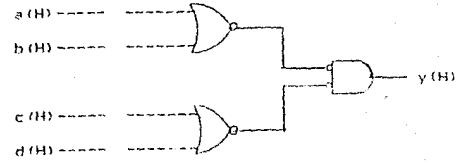


Fig. 5.27 a) Implementaciòn con compuertas NOR.

2o. Paso

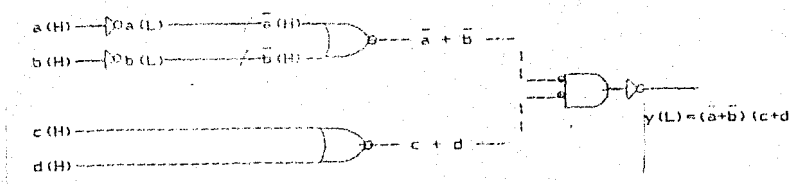


Fig. 5.27 b) Cont.

EJEMPLO 3.

Supóngase que se desea comparar 2 números de 2 bits cada uno A=A2A1 y B=B2B1 si A>B se activará la salida F(L). En la figura 5.28 se describe esta función por medio de su tabla de verdad, su expresión booleana y simplificación

$$F=\bar{A}_2A_1\bar{B}_2\bar{B}_1+A_2\bar{A}_1\bar{B}_2\bar{B}_1+A_2\bar{A}_1\bar{B}_2B_1+A_2A_1\bar{B}_2\bar{B}_1+A_2A_1\bar{B}_2B_1+A_2A_1B_2\bar{B}_1$$

$$F=A_1\bar{B}_2\bar{B}_1(\bar{A}_2+A_2)+A_2\bar{B}_2(\bar{A}_1\bar{B}_1+\bar{A}_1B_1+A_1\bar{B}_1+A_1B_1)+A_2A_1\bar{B}_1(\bar{B}_2+B_2)$$

$$F=A_1\bar{B}_2\bar{B}_1+A_2\bar{B}_2+A_2A_1\bar{B}_1$$

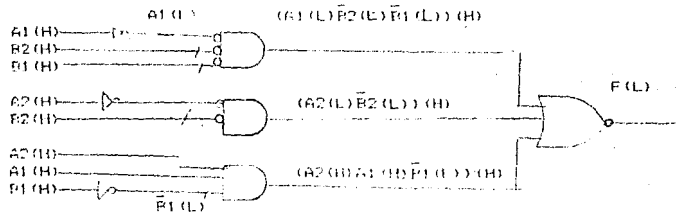


Fig. 5.28 a) Función booleana e implementación de F

A_2 (H)	A_1 (H)	B_2 (H)	B_1 (H)	f (L)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Fig 5.28 b) Cont Tabla de verdad

LOGICA POSITIVA

Como ya antes se ha mencionado, la lógica positiva es a menudo la más comúnmente usada, razón por la cual se considera indispensable tratarla en este texto.

Si se considera la lógica positiva como un caso particular de la lógica mixta, se tendrá que, las variables se toman siempre como verificadas altas, lo cual se traduce en el hecho de que las compuertas lógicas tengan ahora una única representación (señalada con un asterisco en la tabla de la figura 5.9. Se verá, por ejemplo, lo que sucede con la compuerta NAND de la figura 5.29.

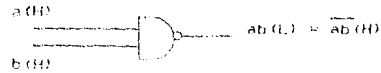


Fig. 5.29 Compuerta NAND

La salida, que comúnmente se había visto como verificada baja, se deberá interpretar ahora como verificada alta, lo cual resulta muy claro si se recuerda que para cualquier variable se cumple que x(L)=x(H).

Así pues, el circuito que representa a la función f=ab + cd, tiene un aspecto tan sencillo como el que se muestra en la fig. 5.30, donde se ha excluido la notación (H), ya que no es necesario porque todas las variables son siempre verificadas altas

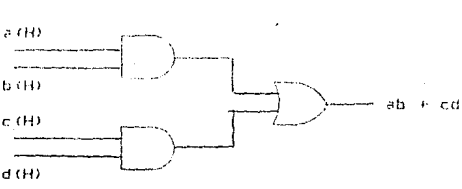


Fig. 5.30 Implementación en lógica positiva de ab+cd.

En este caso sin embargo, si se deseara encontrar algún otro circuito equivalente, el proceso de búsqueda tendría que ser forzosamente algebraico:

$$f = ab + cd$$

$$\bar{f} = \overline{ab + cd} = \bar{a}\bar{b}\,\bar{c}\bar{d}$$

$$f = \bar{\bar{f}} = \overline{(\bar{a}\bar{b})\ (\bar{c}\bar{d})}$$

que equivalente al siguiente circuito de la figura 5.31.

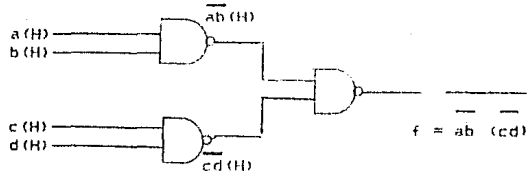
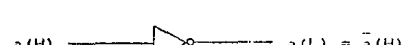


Fig. 5.31 Implementación de $f = \overline{\bar{a}\bar{b}\ \bar{c}\bar{d}}$

Inversores de voltaje.- En lógica positiva los inversores de voltaje son también inversores de lógica o negadores:



(aquí —▷○— = —○▷—)

De esta manera, el circuito que implementa la función $f=a\bar{b}$ es sencillamente el que se muestra en la figura 5.32.

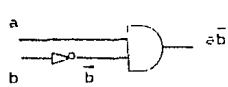


Fig. 5.32 Implementación de $f = a\bar{b}$

En la figura 5.33 se presentan ejemplos de funciones representadas e interpretadas con lógica positiva:

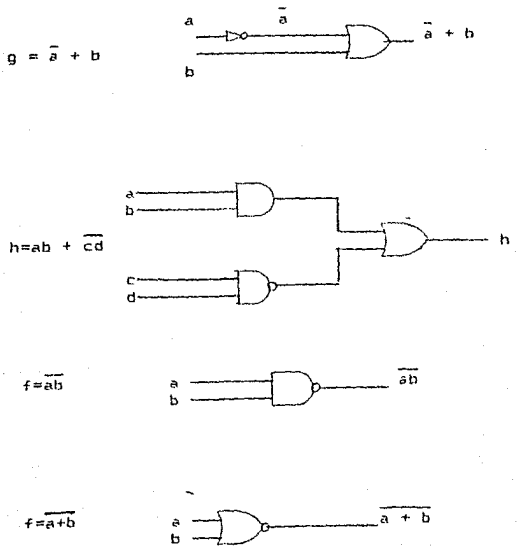


Fig. 5.33 Ejemplos de implementación con lógica positiva.

IMPLEMENTACION DE FUNCIONES CON LOGICA POSITIVA

A continuación se presenta un ejemplo en el que se implementará un probador de paridad con lógica positiva.

Supóngase que en cierta transmisión de información se desea probar la ocurrencia de errores por medio de un bit de paridad, teniendo además, la condición de que el 000 nunca será debe ser recibido, los datos transmitidos son de 4 bits, incluyendo el bit de paridad (par). en caso de presentarse algun error, deberá activarse la línea de salida F. La tabla de verdad de la figura 5.34 muestra esta función junto con la simplificación de la expresión booleana. transmisión

bit p	D_3	D_2	D_1	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Fig. 5.34 Tabla de verdad del probador de paridad

$$F = \bar{P}\,\bar{D}_3\bar{D}_2\bar{D}_1 + \bar{P}\,\bar{D}_3\bar{D}_2 D_1 + \bar{P}\,\bar{D}_3 D_2\bar{D}_1 + \bar{P}D_3\bar{D}_2\bar{D}_1 + \bar{P}D_3 D_2 D_1 + PD_3\bar{D}_2\bar{D}_1$$
$$+ PD_3 D_2 D_1 + PD_3\bar{D}_2 D_1 + PD_3 D_2\bar{D}_1$$

$$F = \bar{P}\,\bar{D}_3\bar{D}_2(\bar{D}_1 + D_1) + \bar{P}\,\bar{D}_2\bar{D}_1(\bar{D}_3 + D_3) + \bar{P}\,\bar{D}_3\bar{D}_1(\bar{D}_2 + D_2) + \bar{D}_3\bar{D}_2\bar{D}_1(\bar{P} + P) +$$
$$PD_3D_2D_1 + P\bar{D}_3D_2D_1 + PD_3\bar{D}_2D_1 + PD_3D_2\bar{D}_1$$

$$F = \bar{P}\,\bar{D}_3(\bar{D}_2 + \bar{D}_1) + \bar{D}_2\bar{D}_1(\bar{P} + \bar{D}_3) + D_2D_1(\bar{P}D_3 + P\bar{D}_3) + PD_3(\bar{D}_2D_1 + D_2\bar{D}_1)$$

$$F = \bar{P}\,\bar{D}_3(\bar{D}_2 + \bar{D}_1) + \bar{D}_2\bar{D}_1(\bar{P} + \bar{D}_3) + D_2D_1(P \oplus D_3) + PD_3(D_2 \oplus D_1)$$

Fig 5.34 Cont. Simplificación de la función booleana del probador de paridad

La implementación de esta función se representa en la figura 5.35.

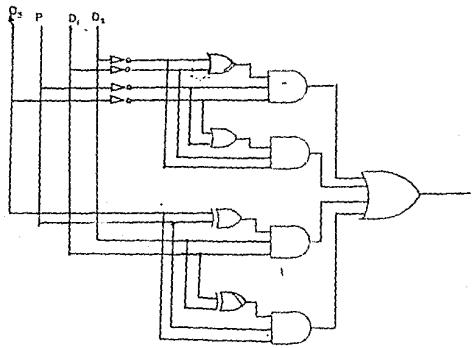


Fig. 5.35 Implementación de un probador de paridad.

C A P I T U L O V I

METODOS DE SIMPLIFICACION

En la tabla de la figura 6.1 se muestran los distintos minitèrminos y maxitèrminos que se pueden obtener con tres variables. Los primeros se designan con una letra m (minúscula) y los segundos con una M (mayúscula).

x	y	z	minitèrmino	maxitèrmino
0	0	0	$\bar{x}\,\bar{y}\,\bar{z} = m_0$	$x + y + z = M_0$
0	0	1	$\bar{x}\,\bar{y}\,z = m_1$	$x + y + \bar{z} = M_1$
0	1	0	$\bar{x}\,y\,\bar{z} = m_2$	$x + \bar{y} + z = M_2$
0	1	1	$\bar{x}\,y\,z = m_3$	$x + \bar{y} + \bar{z} = M_3$
1	0	0	$x\,\bar{y}\,\bar{z} = m_4$	$\bar{x} + y + z = M_4$
1	0	1	$x\,\bar{y}\,z = m_5$	$\bar{x} + y + \bar{z} = M_5$
1	1	0	$x\,y\,\bar{z} = m_6$	$\bar{x} + \bar{y} + z = M_6$
1	1	1	$x\,y\,z = m_7$	$\bar{x} + \bar{y} + \bar{z} = M_7$

Fig. 6.1 Minitèrminos y maxitèrminos de tres variables.

Los siguientes ejemplos mostrarán cómo, a partir de una expresión cualquiera de una función, se puede obtener alguna de las formas estándar de la misma y, en consecuencia la canónica correspondiente.

1) Sea la función $f(x,y,z) = \bar{x}\,\bar{y} + z + \bar{x}\,\bar{z}$. Se desea expresarla en forma canónica, para lo_ cual se hará uso de propiedades ya conocidas tales como que $x + x = 1$, $x + x = x$, $x+1 = 1$, etc..

Se tiene entonces que:

$$f(x,y,z) = \bar{x}\,\bar{y} + z + \bar{x}\,\bar{z}$$

$$= \bar{x}\,\bar{y}(z + \bar{z}) + z(x + \bar{x})(y + \bar{y}) + \bar{x}\,\bar{z}(y + \bar{y})$$

$$= \bar{x}\,\bar{y}z + \bar{x}\,\bar{y}\,\bar{z} + xyz + \bar{x}yz + x\bar{y}z + \bar{x}\,\bar{y}z + \bar{x}\,\bar{y}\,\bar{z} + x\bar{y}\,\bar{z}$$

$$= \bar{x}\,\bar{y}z + \bar{x}\,\bar{y}\,\bar{z} + xyz + x\bar{y}z + \bar{x}yz + x\bar{y}\bar{z}$$

METODOS DE SIMPLIFICACION

Cuando se diseña un sistema no sólo se espera que ejecute la función deseada, sino que además se busca siempre la mayor eficiencia, tomando en cuenta el espacio físico que ha de ocupar, el tiempo de ejecución, el consumo de potencia, etc., así como las diversas posibilidades de implementación con circuitos de baja, mediana, alta y muy-alta escala de integración. Es precisamente esta búsqueda la que llevará a considerar la minimización o simplificación de funciones booleanas como una de las etapas de mayor importancia en el diseño lógico.

En esta sección se tratará el método más usual para la simplificación de funciones: los mapas de Karnaugh y los mapas de variable introducida. Sin embargo, para poder proceder a describirlos con detalle, es necesario el conocimiento previo de las dos posibles formas en las que puede expresarse una función booleana, y que se conoce con el nombre de formas canónicas, mismas que nos resultarán imprescindibles a la hora de poner en práctica el método de simplificación mencionado.

Formas canónicas.- Existen dos maneras usuales de representar una función: como suma de productos, o como un producto de sumas. Un ejemplo de cada una de ellas se muestra a continuación:

1) $f(x,y,z) = \bar{x}\bar{y}\bar{z} + x\bar{y}z + xyz$ suma de productos

2) $f(a,b) = (a + b)(\bar{a} + \bar{b})$ producto de sumas

A todos aquellos términos que involucran a la totalidad de las variables, ya sean afirmadas o negadas, de las que depende la función expresada como una suma de productos, se les denomina productos estándar o, más comúnmente, minitérminos, en este caso los minitérminos son:

$$\bar{x}\bar{y}\bar{z} \quad x\bar{y}z \quad xyz$$

De forma similar, a los términos que forman expresiones del tipo producto de sumas y que involucran a todas las variables de las que depende la función, se les denomina sumas estándar, o sencillamente, maxitérminos. En el caso del inciso 2, los maxitérminos son:

$$(a + b) \text{ y } (\bar{a} + \bar{b}).$$

Ahora bien, cuando una función se expresa, ya sea como una suma de minitérminos o bien como un producto de maxitérminos, se dice que está expresada en forma canónica.

2) Sea $f(x,y,z) = xy + z\bar{y}$.

Expresándola primeramente en forma estándar como un producto de sumas, esto es:

$$
\begin{aligned}
f\ (x,\ y,\ z) &= xy + zy \\
&= (xy + z)\ (xy + \bar{y}) \\
&= (z + x)\ (z + y)\ (\bar{y} + x)\ (y + \bar{y}) \\
&= (z + x)\ (z + y)\ (\bar{y} + x)
\end{aligned}
$$

donde se ha utilizado una propiedad de distibutividad muy importante que dice que $x + yz = (x + y)(x + z)$.

Pasando ahora a la forma canónica se tendrá que:

$$
\begin{aligned}
f\ (x,\ y,\ z) &= (z + x + y\bar{y})(z + y + x\bar{x})(\bar{y} + x + z\bar{z}) \\
&= (z+x+y)\,(z+x+\bar{y})\,(z+x+y)\,(z+\bar{x}+y)\,(z+x+\bar{y})\,(\bar{z}+x+\bar{y}) \\
&= (z+x+y)\,(z+x+\bar{y})\,(z+\bar{x}+y)\,(\bar{z}+x+\bar{y})
\end{aligned}
$$

Ahora bien, según se vió en la tabla de la figura 6.1, existe una manera simplificada de expresar tanto a los minitérminos como a los maxitérminos, consistente en asociar a cada uno de ellos una sigla única. Véase cómo se realiza esta asociación prescindiendo de una tabla como la ya citada.

Supóngase que se tiene la función $f(x,y,z) = \bar{x}\ \bar{y}\ \bar{z} + xyz$. Para identificar a cada uno de los minitérminos que la conforman se hace lo siguiente: si se tiene la convención de que la variable sin testar es verdadera y la testada falsa, se tendrá que, interpretando cada minitérmino como un número binario, se obtendrá el subíndice correspondiente a la letra m que designa al minitérmino, esto es:

minitérmino	No. binario	sigla
$x\ y\ z$	$111 = 7_{10}$	m_7
$\bar{x}\ \bar{y}\ \bar{z}$	$000 = 0_{10}$	m_0

por lo que la función se puede expresar como

$$f(x,y,z) = m_0 + m_7$$

que suele escribirse

$$f(x,y,z) = \Sigma(0,7)$$

Con los maxitérminos, el procedimiento es similar, sólo que en este caso se toma la convención contraria, es decir, que ahora la variable testada será la verdadera y la no testada la falsa (esto es únicamente por conveniencia).

Así pues, supóngase que se tiene la función

$$f(x,y,z) = (x + y + z)(\bar{x} + \bar{y} + z)$$

en este caso se tendrá que:

maxitérmino	No. binario	sigla
$x + y + z$	$000 = 0_{10}$	M_0
$\bar{x} + \bar{y} + z$	$110 = 6_{10}$	M_6

y por tanto

$$f(x,y,z) = M_0 M_6$$

o bien

$$f(x,y,z) = \pi(0,6)$$

Ahora bien, no sólo es posible pasar de la forma estándar a la canónica, sino también de una forma canónica a otra. Esto, como se verá a continuación, es muy sencillo e inclusive puede ya haberse intuido.

Obsérvese nuevamente la tabla de la figura 6.1. Fijándose bien en alguno de los minitérminos expresados en ella, por ejemplo m_0, se verá que

$$m_0 = \bar{x}\,\bar{y}\,\bar{z}$$

Niéguese ahora esta expresión:

$$\bar{m}_0 = \bar{x}\,\bar{y}\,\bar{z} = x + y + z = M_0$$

lo cual es generalizable para una función cualquiera de n variables como la relación $\bar{m}_i = M_i$, donde i = 0, 1, 2, ...2n.

El siguiente ejemplo muestra ampliamente como transformar una función de una forma canónica a otra.

Sea la función.

$$f(x, y, z) = x\,y\,z + \bar{x}yz + x\bar{y}z + xy\bar{z}$$

$$= \Sigma\,(7, 5, 3, 0)$$

cuya tabla de verdad es la siguiente:

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Si se complementa a f, directamente de la tabla de verdad se obtiene que:

x	y	z	$\bar{f}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

de donde

$$f'(x, y, z) = m_0 + m_1 + m_2 + m_4$$

y por tanto, al volver a complementar, ahora algebraicamente, usando los teoremas de De Morgan, se obtiene que:

$$f''(x, y, z) = f(x, y, z) = \overline{m_0 + m_1 + m_2 + m_4}$$

$$= \bar{m}_0 \bar{m}_1 \bar{m}_2 \bar{m}_4$$

y dado que $\bar{m} = M$ se obtiene que

$$f(x, y, z) = M_0 M_1 M_2 M_4$$

Si se observa bien, se caerá en cuenta de que para pasar de una forma canónica a otra, basta con fijarse en los subíndices de los minitérminos que no están expresados en f (i.e., los que representan el complemento), y tomarlos para formar los maxitérminos.

De esta manera, se tiene que

$$f(x, y, z) = \Sigma (1, 3, 7, 4) = \Pi (2, 5, 6, 0)$$

Resumiendo lo dicho en esta sección, se ha encontrado que una función cualquiera f puede expresarse de dos maneras: como la suma de sus minitérminos ($f = \Sigma m_i$) o como el producto de sus maxitérminos ($f = \Pi M_i$).

MAPAS DE KARNAUGH.

En secciones anteriores se ha estudiado la minimización de expresiones booleanas por medio de métodos algebraicos, lo que en ocasiones puede resultar demasiado largo o complicado, además de que se carece de reglas específicas que indiquen el proceso que conduzca a la simplificación óptima

Los mapas de Karnaugh tienen la facultad de presentar de una manera esquemática todas las posibles alternativas que existen para expresar una función dada, de modo que pueda escogerse entre todas ellas a la más simple, siempre y cuando se tengan menos de seis variables, pues, de lo contrario se vuelve engorroso.

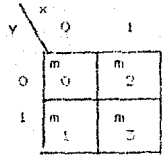
Los mapas están constituidos por cuadros o celdas, cada uno de los cuales representa un minitérmino. Dado que una función puede expresarse como la suma de sus minitérminos, se tiene en consecuencia que en un mapa-K, dicha función quedará representada por el área contenida por los cuadros correspondientes a los minitérminos que la conforman.

Supóngase que se tiene una función (x,y) -cuya forma explícita no interesará por ahora- que, como siempre, puede representarse en una tabla de verdad:

minitérmino	x	y	f
m_0	0	0	
m_1	0	1	
m_2	1	0	
m_3	1	1	

tabla en la que se ha agregado la columna del extremo izquierdo para indicar los minitérminos.

A continuación se dibujará el mapa-k correspondiente a esta tabla, para lo cual se habrá de necesitar cuatro cuadros, uno para cada minitérmino, como se presenta en la figura 6.2.



MAPA-K

minitèrmino	x	y	f
m 0	0	0	
m 1	0	1	
m 2	1	0	
m 3	1	1	

TABLA DE VERDAD

Fig. 6.2 Mapa-K asociada a una funciòn f.

Nòtese que el mapa-k es muy similar a un sistema coordenado; el cuadro que corresponde, por ejemplo, al minitèrmino m3, esto es, x=1, y=1, se localiza justamente en la intersecciòn de la columana x=1 y el reglòn y=1, como se muestra en la figura 6.3.

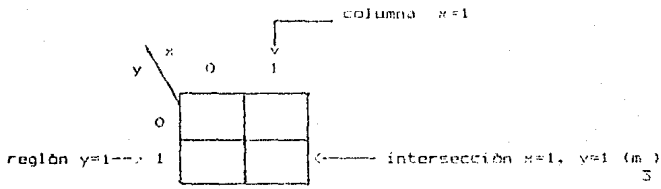


Fig. 6.3 Localizaciòn de m_3 en un mapa-K.

Es conveniente subrayar que en el presente texto se adoptarà la convenciòn de designar a las columnas por medio de las variables màs significativas y a los renglones por las menos significativas, convenciòn que varia segùn el autor, por lo que se deberà tener precauciòn en la interpretaciòn de los mapas cuando se recurra a otra fuente.

Veàse ahora un ejemplo de còmo se puede elaborar el mapa-K de una funciòn sencilla, como por ejemplo $f(x,y) = xy$ (Fig. 6.4).

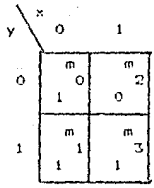
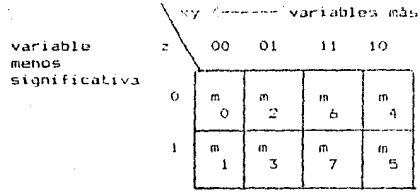


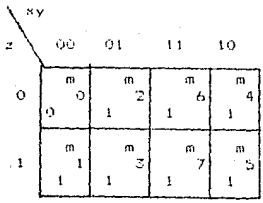
Fig. 6.6 Mapa-K de f(X,Y) = Y + Y + XY.

Supóngase ahora que nuestra función depende de tres variables, el orden en el que se numeran las celdas en el mapa-k es el correspondiente al código GRAY, por lo que se deberá tener la disposición siguiente:



Se verá un ejemplo de estos. Sea la función f(x, y, z) = x + y + z. cuya tabla de verdad se muestra en la figura 6.7 junto con su representación en un mapa-K.

minitérmino	x	y	z	f
m 0	0	0	0	0
m 1	0	0	1	1
m 2	0	1	0	1
m 3	0	1	1	1
m 4	1	0	0	1
m 5	1	0	1	1
m 6	1	1	0	1
m 7	1	1	1	1



f = (m , m , m , m , m , m , m)
1 2 3 4 5 6 7

Fig. 6.7 Mapa-K de f(X,Y,Z) = X + Y + Z.

mapa-K

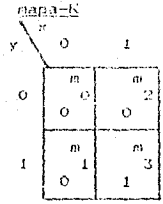


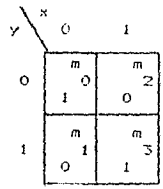
tabla de verdad

minitérmino	x	y	f
m 0	0	0	0
m 1	0	1	0
m 2	1	0	0
m 3	1	1	1

Fig. 6.4 Mapa-K de f(x,y) = xy.

Puede observarse, lo que se hace es colocar un 1 en la celda correspondiente al minitèrmino que representa a la función, colocàndose un 0 en las celdas restantes.

Ahora bien, cada cuadro en un mapa-K se relaciona con los demàs por medio de una operación OR lo_cual por ejemplo, lleva a representar la función f(x,y) =x y + xy como se expresa en la figura 6.5.



minitèrmino	x	y	f
m 0	0	0	1
m 1	0	1	0
m 2	1	0	0
m 3	1	1	1

Fig. 6.5 Mapa-K de $f(x,y) = \bar{x}\bar{y} + x\bar{y}$.

Vease otro ejemplo: Sea la función f(x,y,) = x + y + xy.

Como primer paso se obtiene su tabla de verdad:

x	y	f
0	0	0
0	1	1
1	0	1
1	1	1

Los minitèrminos que conforman a la función son m1 , m2 , y m3 , por lo que el mapa-K resultante es el de la figura 6.6.

En caso de que la función dependa de cuatro variables, el mapa-K presenta el aspecto de la figura 6.8.

zt \ xy	00	01	11	10
00	m 0	m 4	m 12	m 8
01	m 1	m 5	m 13	m 9
11	m 3	m 7	m 15	m 11
10	m 2	m 6	m 14	m 10

Fig. 6.8 Minitérminos en un mapa de 4 variables.

Como ejemplo de esto se presenta en la figura 6.9, la tabla de verdad y el mapa-K de la función f(x, y, z, t) = xy + zt.

Páginas atrás se mencionó el hecho de que los mapas-K eran una herramienta de suma utilidad en el proceso de simplificación de funciones; sin embargo, para poder usarlos con este fin, es necesario estudiar antes algunas de las características inherentes a la estructura de dichos mapas.

Primeramente, obsérvese que tanto las columnas como los renglones en un mapa-K se numeran acorde al código Gray, lo que da como resultado que dos cuadros o celdas adyacentes -horizontal o verticalmente- representen minitérminos que difieren entre sí únicamente en una de las variables, apareciendo complementada en alguno de los minitérminos y sin complementar en el otro. Característica que no se cumple cuando son adyacentes diagonalmente.

minitérmino	x	y	z	t	f
m_0	0	0	0	0	0
m_1	0	0	0	1	0
m_2	0	0	1	0	0
m_3	0	0	1	1	1
m_4	0	1	0	0	0
m_5	0	1	0	1	0
m_6	0	1	1	0	0
m_7	0	1	1	1	1
m_8	1	0	0	0	0
m_9	1	0	0	1	0
m_{10}	1	0	1	0	0
m_{11}	1	0	1	1	1
m_{12}	1	1	0	0	1
m_{13}	1	1	0	1	1
m_{14}	1	1	1	0	1
m_{15}	1	1	1	1	1

$f = (m_3, m_7, m_{11}, m_{12}, m_{13}, m_{14}, m_{15})$

zt \ xy	00	01	11	10
00	0 0	4 0	12 1	8 0
01	1 0	5 0	13 1	9 0
11	3 1	7 1	15 1	11 1
10	2 0	6 0	14 1	10 0

(el número en la esquina superior derecha de cada celda indica el minitérmino de que se trata).

Fig. 6.9 Mapa-K de $f(x,y,z,t) = xy + zt$

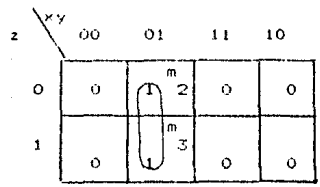
En la figura 6.10 se muestra un mapa-K para tres variables, presentando explícitamente cada uno de los minitérminos que lo conforman:

z \ xy	00	01	11	10
0	m_0 0 $\bar{x}\bar{y}\bar{z}$	m_2 2 $\bar{x}y\bar{z}$	m_6 6 $xy\bar{z}$	m_4 4 $x\bar{y}\bar{z}$
1	m_1 1 $\bar{x}\bar{y}z$	m_3 3 $\bar{x}yz$	m_7 7 xyz	m_5 5 $x\bar{y}z$

Fig. 6.10 Minitérminos en un mapa-K de 3 variables.

Se verà ahora mediante un sencillo ejemplo la minimizaciòn de una funciòn mediante un mapa-K.

Sea pues la funciòn $f(x,y,z) = \bar{x}y\bar{z} + \bar{x}yz$, cuyo mapa es el que se muestra en la figura 6.11.



$f = (m2, m3)$

Fig. 6.11 Reducciòn de $f(x,y,z) = \bar{x}\ y\ \bar{z} + z\ y\ \bar{x}$.

Al observarlo, nòtese que los minitèrminos implicados en la funciòn son adyacentes, lo cual indica que es posible realizar la siguiente reducciòn:

$$m_2 + m_3 = \bar{x}y\bar{z} + \bar{x}yz = \bar{x}y(z + \bar{z}) = \bar{x}y$$

con la cual se obtiene una expresiòn màs simple para la funciòn:

$$f(x,y,z) = \bar{x}y$$

De lo anterior se puede extraer una sencilla regla: cuando los minitèrminos involucrados en la representaciòn de una funciòn aparecen en el mapa-K de manera que se puedan formar grupos que contengan un nùmero de elementos que sea potencia de 2 , es posible simplificar las variables que son iguales en todos los minitèrminos, en tanto que las restantes, no iguales, pueden excluise.

Por regla general, y como ayuda visual para la simplificaciòn, el grupo de minitèrminos en cuestiòn se encierra en un òvalo.

Supòngase, ahora como ejemplo de lo expuesto, que se tiene la funciòn:

$$f(x,y,z) = \bar{x}\ \bar{y}\ \bar{z} + \bar{x}y\bar{z} + xy\bar{z} + x\bar{y}\ \bar{z} = \quad (m0, m2, m6, m4)$$

El mapa-K que le corresponde: se muestra en la figura 6.12.

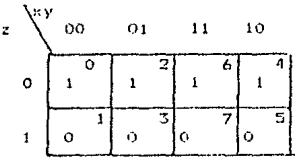


Fig. 6.12 Reducción de $f(X,Y,Z) = (m_0, m_2, m_0, m_4)$

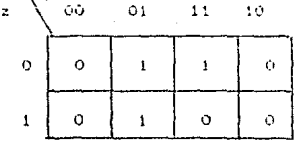
Como podrá observarse, la única variable que se presenta igual en todos los minitérminos es z, siendo entonces la única variable que se conserva. Por si hubiera alguna duda respecto de que si esto es o no válido, se justificará algebraicamente la omisión de las variables restantes:

$$f(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}yz + x\bar{y}z + xy\bar{z}$$

$$= \bar{z}(\bar{x}\,\bar{y} + \bar{x}y + x\bar{y}) + xy\bar{z}$$

$$= \bar{z}(\bar{x}(\bar{y} + y) + x(y + \bar{y})) = \bar{z}(\bar{x} + x) = \bar{z}$$

Se tratará ahora un caso diferente. Sea el mapa-K de la figura 6.13.



f = Σ (m2,m6,m3)

Fig. 6.13 Mapa-K de $f(X,Y,Z) = (m_2, m_6, m_3)$

En este caso lo que se hará será formar dos grupos de dos, ya que no es posible hacerlos de tres. De acuerdo a esto el mapa queda como se muestra en la figura 6.14.

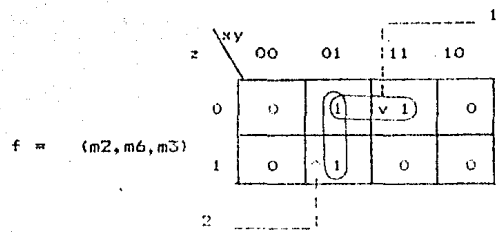


Fig. 6.14 Reducción de $f(X,Y,Z) = (m_2, m_6, m_3)$

En el grupo 1 las variables que coinciden son yz, en tanto que en el grupo 2 son xy, con lo que se obtiene la siguiente expresión simplificada para f:

$$f(x,y,z,) = \bar{x}y + y\bar{z}$$

Ahora bien, es posible que de vez en cuando se encuentren funciones en las que alguno de sus minitérminos no pueda ser agrupado con ningún otro, como por ejemplo, la que se presenta en el mapa-K de la figura 6.15.

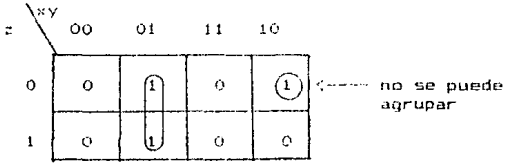


Fig. 6.15 Mapa-K de $f = (m_2, m_3, m_4)$

en cuyo caso la función se expresa simplemente como

$$f(x, y, z) = \bar{x}y + x\bar{y}\bar{z}$$

es decir, el minitérmino aislado se expresa sin alteración.

Considerando ahora la figura 6.16, para propósitos de simplificación, se deberá imaginar que ambas fronteras coinciden, que son la misma, como si estuvieran unidas:

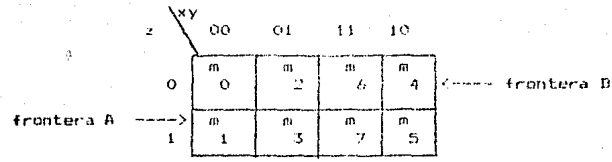


Fig. 6.16 Consideraciones de frontera.

De esta manera, m0 resulta adyacente a m4, del mismo modo que m1 con m5. Cabe hacer notar que un argumento similar resulta válido al considerar las fronteras horizontales. (Fig. 6.17).

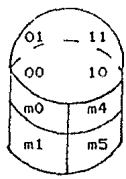
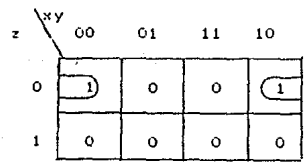


Fig. 6.17 Representación de mapa-K.

Vease ahora algunos ejemplos concretos sobre esto.

1) Sea $f(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + x\bar{y}\,\bar{z} =$ (m0, m4)

El mapa-K correspondiente es el de la figura 6.18.



de donde

$f(x,y,z) = \bar{y}\,\bar{z}$

Fig. 6.18 Reducción de $f = (m_0, m_4)$

2) Sea $f(x, y, z, t) = \bar{x}\bar{y}\bar{z}\bar{t} + xy\bar{z}\bar{t} + \bar{x}\bar{y}\bar{z}t + xyzt = \Sigma\ (m4, m12, m6, m14)$

z \ xy	00	01	11	10
00	0	1	1	0
01	0	0	0	0
11	0	0	0	0
10	0	1	1	0

de donde

$$f(x, y, z, t) = y\bar{t}$$

Fig. 6.19 Reducción de $f(X,Y,Z,T) = (m_4, m_{12}, m_{16}, m_{14})$

Funciones simplificadas expresadas como producto de sumas.- En ocasiones, se hace necesario expresar la función simplificada no como una suma de productos, que es lo usual, sino como un producto de sumas. Lo cual puede lograrse a partir de algunas variaciones aplicadas al método de los mapas-K que se ha estado usando.

Recuérdese que cuando se describieron los mapas-K, se estableció que los minitérminos representativos de una función dada se señalaban en él mediante un 1, lo cual lleva a la afirmación de que los restantes minitérminos, es decir, los señalados con un 0, son representativos de la función negada, i.e., de f'. De acuerdo a esta aseveración, y manejando el mismo método, se puede simplificar la función negada, intercambiando únicamente el papel de los 0's y los 1's.

Supóngase, como ejemplo, que se tiene el mapa-k de la figura 6.20.

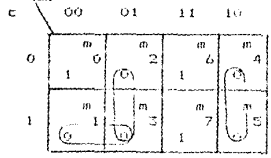


Fig. 6.20 Mapa-k de $f(a,b,c) = (M_1, M_2, M_3, M_4, M_5)$.

de donde se obtiene que

$$\bar{f}\ (a, b, c) = \bar{a}\bar{b} + a\bar{b} + \bar{a}c$$

y por tanto

$$f(a, b, c) = f''(a, b, c) = (a + \bar{b})\ (\bar{a} + b)\ (a + \bar{c})$$

Funciones incompletamente especificadas.- Cuando se elabora un mapa-K, el diseñador se basa en la información proporcionada por la tabla de verdad, observando en qué casos la función es verdadera (f=1) y en cuáles es falsa (f=0). Sin embargo, en ocasiones sucede que para ciertos casos la función no toma un valor determinado, lo cual se representa mediante una X en la tabla. En este caso se dice que la función no esta completamente definida y la función puede tomar cualquiera de los dos valores, 0 ó 1, según nuestra conveniencia al momento de la simplificación. Como ejemplo de esto se muestra la tabla siguiente:

a	b	f
0	0	1
0	1	x
1	0	x
1	1	1

Condiciones como éstas aparecen cuando las características propias del dispositivo representado por una función, establecen que dichos estados no habrán de representarse, es decir que los estados a=0,b=1 y a=1,b=0 de este ejemplo no son estados viables del diseño, razón por la cual no interesan.

Así pues, al llevar la información de la tabla de verdad al mapa-K, estos estados indefinidos se representan también con una X. En el caso del ejemplo anterior, el mapa-K tiene el aspecto de la figura 6.21.

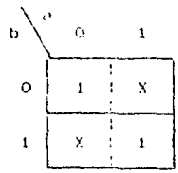


Fig. 6.21 Mapa-K con condiciones de "No Importa".

Se dijo ya que la X puede valer 0 ó 1 según convenga, lo cual implica diversas posibilidades de simplificación, algunas de ellas mejores que otras, como se puede juzgar a continuación a propósito de este mismo ejemplo (Fig 6.22).

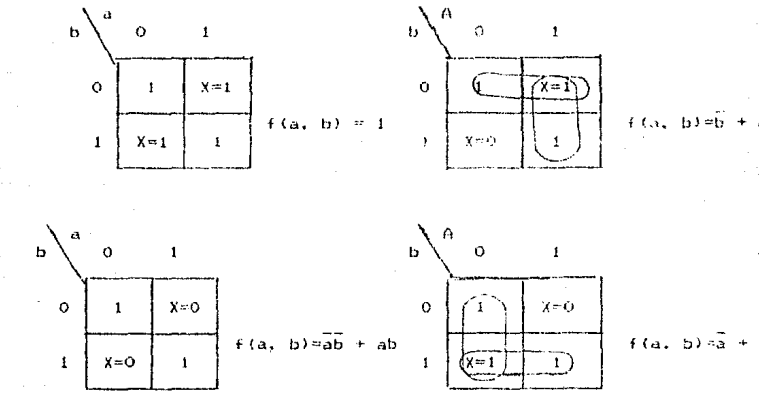


Fig. 6.22 Posibles simplificaciones para un mapa-K con condiciones de "No Importa".

Evidentemente, las cuatro funciones son diferentes, sin embargo todas satisfacen la tabla de verdad original, razón por la cual cualesquiera de ellas puede ser tomada como función representativa. Evidentemente también esto es aplicable al caso de tres o más variables.

Mapas de Karnaugh para funciones de más de 4 variables.- El procedimiento implicado en el uso y elaboración de mapas-K para cinco variables es, como puede preverse, algo más complicado; tan solo tómese en cuenta el hecho de que se habrá de necesitar 32 celdas, para tener una idea de ello. En la figura 6.23 se presenta este mapa.

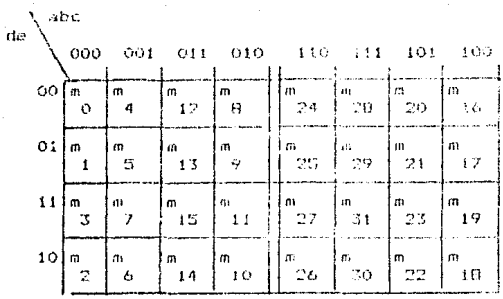


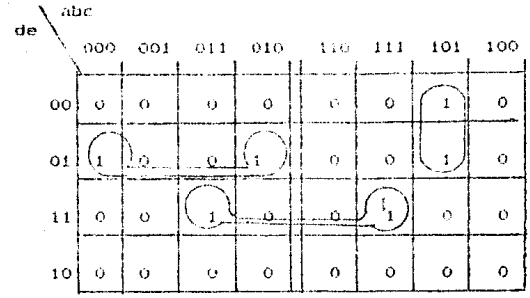
Fig. 6.23 Mapa-K de 5 variables.

Todas las reglas dadas anteriormente para el uso de los mapas-K son vàlidas para este caso, aunque resulta necesario modificar la idea de cuatro adyacente, ya que existen celdas que "fisicamente" no estàn juntas, pero puede considerarse que sí. Considèrese el mapa anterior como si estuviera formado por dos mapas de cuatro variables, delimitados entre sí por una linea doble. Visto asi, resulta claro, por ejemplo que las celdas correspondientes a m_{25} y m_{17} son adyacentes, lo mismo que m_3 y m_{11}.

Por otra parte, las fronteras de los mapas también son adyacentes, por lo que la celda m_9 resulta adyacente a la m_{25}, asi como la m_3 a la m_{19}.

Ademàs de las anteriores existe otro tipo de adyacencia: todo cuadro o celda es adyacente a su simètrico respecto de la linea doble divisora; las celdas m_{29} y m_{13} son adyacentes, asi como m_7 y m_{23}.

Como ejemplo veàse la figura 6.24 donde se simplifica la funciòn $f(a,b,c,d,e) = (m_1, m_9, m_{15}, m_{20}, m_{21}, m_{31})$.



$f(a,b,c,d,e) = \bar{a}\,\bar{c}\,\bar{d}\,e + b\,c\,d\,e + a\,\bar{b}\,c\,\bar{d}.$

Fig. 6.24 Simplificación de $f=\Sigma(m_1, m_9, m_{15}, m_{20}, m_{21}, m_{31})$.

Como puede verse, aunque el mètodo se ha complicado un poco, aún es posible reconocer patrones de cuadros adyacentes, cosa que también ocurre con mapas-K para 6 variables, mismos que se analizan considerandolos como si estuvieran formados por cuatro mapas-K de 4 variables. En la figura 6.25 se muestra la estructura de este mapa, en el cual por ejemplo, pueden considerarse adyacentes las celdas correspondientes a m_{15}, m_{47}, m_{43} y m_{11}, así como las que designan a m_{58}, m_{62}, m_{26} y m_{30}.

Para más de 6 variables el método se complica demasiado, por lo cual no serán tratados esos casos. Un ejemplo de simplificación de una función de 6 variables se presenta en la figura 6.26.

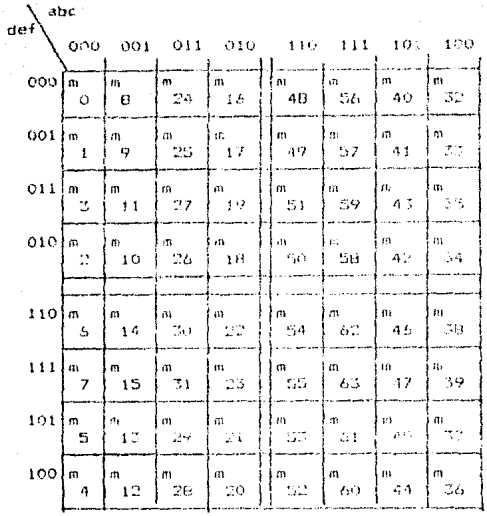


Fig. 6.25 Mapa de Karnaugh para 6 variables.

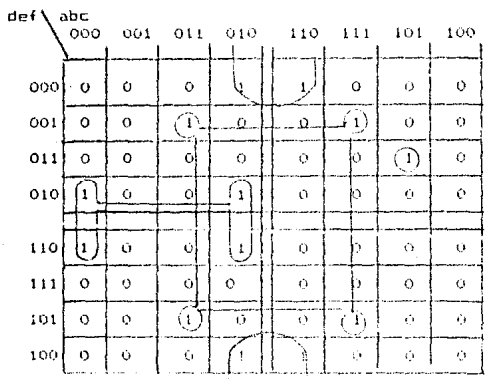


Fig. 6.26 Simplificación de:
f(a,b,c,d,e,f)= (m2, m6, m16, m18, m20, m22, m25, m29, m43, m49, m52, m57, m61)

MAPAS DE VARIABLE INTRODUCIDA

Este método constituye una potente herramienta empleada para la simplificación de funciones booleanas, y será enormemente útil en el tratamiento de los temas que abarcan las siguientes secciones, en especial, aquéllas que presentan problemas en las que el número de variables que se manejan (más de 6 ó 7), hacen de los mapas de Karnaugh un método complicado que entorpece la resolución del problema. Existe otro método muy reconocido para los casos de muchas variables conocido como "tabulación" que se tratado ampliamente en la mayoría de los textos, por lo que no se incluye aquí.

A diferencia de un mapa K, el mapa de variable introducida no solamente emplea las cantidades binarias, 0 y 1, sino que también permite la utilización de variables, o inclusive, de expresiones booleanas.

A fin de facilitar la comprensión del tema, éste se dividirá en dos partes: la primera se enfocará principalmente a la elaboración de los mapas de variable introducida y la segunda a lo que propiamente constituye la lectura o interpretación de dichos mapas.

I.1 Elaboración de Mapas de Variables Introducidas

Supóngase que se tiene una función F(A,B,C) expresada como la suma de sus minitérminos:

$$F(A,B,C) = \overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + A\overline{B}C + AB\overline{C} + ABC + A\overline{B}\overline{C} + A\overline{B}\overline{C} \quad \ldots (1)$$

Si se multiplica por F ambos miembros de la ecuación 1, se obtendrá que:

$$F + F.F = \overline{A}\overline{B}\overline{C}F + \overline{A}BCF + A\overline{B}\overline{C}F + A\overline{B}CF + AB\overline{C}F + ABCF + A\overline{B}\overline{C}F + A\overline{B}\overline{C}F \ldots (2)$$

Ahora bien, para generar una determinada función, únicamente es necesario substituir el valor de F en cada uno de los minitérminos que la conforman: así, si F=1, el minitérmino correspondiente será incluido en la expresión final, y si F=0 no lo será. Esto mismo se muestra en el ejemplo de la figura 6.??

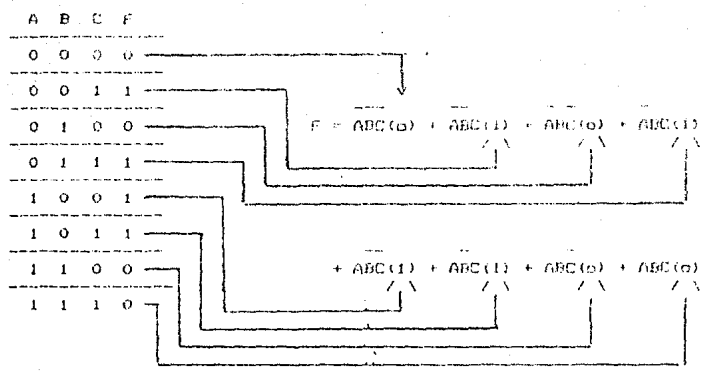


Figura 6.27 Tabla de verdad y expresiòn booleana de F.

Si ahora se toma la ecuaciòn 2 y se agrupan las variables C y F de cada tèrmino, se tendrà que:

$$F = F\,F = \bar{A}\bar{B}(CF+\bar{C}\bar{F}) + \bar{A}B(CF+\bar{C}\bar{F}) + A\bar{B}(CF+\bar{C}\bar{F}) + AB(CF+\bar{C}\bar{F}) \quad \ldots\ldots \quad (3)$$

y si ahora se substituyen en esta ecuaciòn los valores de la funciòn F, segùn la infomaciòn presentada en la figura 6.27, resultarà que:

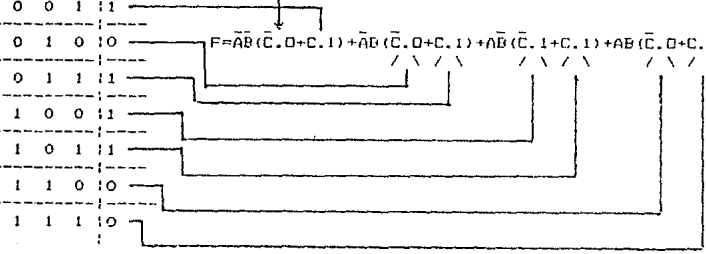


Fig. 6.28 Tabla de verdad y expresiòn booleana de F.

Simplificando la ecuación de la figura 6.28 se tiene que:

$$F = \bar{A}\,\bar{B}\,(C) + \bar{A}\,B\,(\bar{C}) + A\,\bar{B}\,(1) + A\,B\,(0)$$

de donde se obtiene la tabla de verdad reducida y su mapa correspondiente, como se muestra en la figura 6.29

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

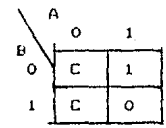


Fig. 6.29 Tabla de verdad y mapa de variable introducida de F..

De la misma manera se obtiene la ecuación para cuatro variables que se muestra a continuación

$$F(A,B,C,D) = A\bar{B}\,(\bar{C}\bar{D}+\bar{C}D+C\bar{D}+CD) + A\bar{B}(\bar{C}\bar{D}+\bar{C}D+C\bar{D}+CD)$$
$$+ \bar{A}B\,(\bar{C}\bar{D}+\bar{C}D+C\bar{D}+CD) + \bar{A}B(\bar{C}\bar{D}+\bar{C}D+C\bar{D}+CD)$$

De acuerdo con este resultado, se puede, por ejemplo, dar solución al problema planteado en la figura 6.30 que muestra una determinada función F, descrita a través de su tabla de verdad.

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

A B	F
0 0	0
0 1	C⊕D
1 0	C
1 1	D

B \ A	0	1
0	0	C
1	C⊕D	D

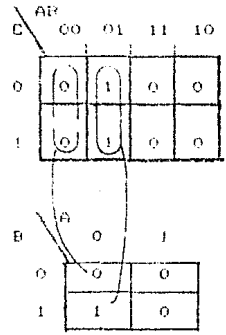
Fig. 6.30 Mapa de variable introducida para una función de 4 variables.

1.2) Equivalencia entre Mapas-k y Mapas de Variable Introducida

Con la finalidad de lograr una mayor comprensión de este tema, se compararáan la representación de una función en mapas-k y en mapas de variable introducida, por lo que se tomarán como referencia lo que ya se sabe sobre los mapas-k para comprender el funcionamiento de los mapas de variable introducida.

Así pues, supóngase que se tiene una función F descrita por la tabla de verdad de la figura 6.31 en donde también se muestran sus dos posibles representaciones: el mapa-k y el de variable introducida, observe, por ejemplo, que en el mapa-k en el minitérmino A=B=0 la función toma el valor de "0" para C=1 y C=0, esto equivale a la celda A=B=0 del mapa de variable introducida, es decir, cuando A=B=0 la función toma el valor de "1" sin importar el valor de C

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



$$F = \bar{A}\,\bar{B}(0) + \bar{A}B(1) + A\bar{B}(0) + AB(0)$$

Fig. 6.31 Equivalente entre mapas-k y de variable introducidas.

Supóngase ahora que se tiene el mapa de la figura 6.32.

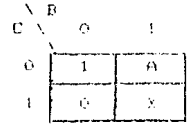


Fig. 6.32 Mapa de variable introducida.

del cual se puede deducir el mapa-K correspondiente de dicho mapa, el único caso que requiere de una explicación detallada, es el correspondiente al minitérmino 2 (B=1, C=0), ya que, en este caso, la variable introducida dentro de la celda cuá indicando que la función F toma el valor de la variable A, esto es:

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

lo cual se observa más claramente en la figura 6.33.

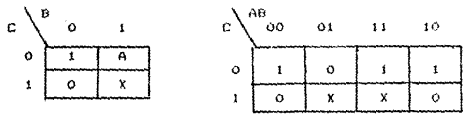


Fig. 6.33 Equivalencia entre mapas-K y de variable introducida.

II. Lectura de Mapas de Variable Introducida

En primer lugar se tratarán los llamados "casos básicos de la lectura", es decir, las posibles situaciones a interpretar que se pueden encontrar en un mapa de variable introducida.

i) Se puede encontrar un mapa contiene una única variable introducida y ceros, tal como se muestra en la figura 6.34.

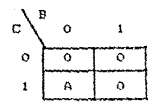


Fig. 6.34 Mapa de variable introducida.

En este caso, la expresión final a la que se llega es $F=\bar{B}CA$, esto es, la variable introducida multiplicada por el minitérmino donde se encuentra. Se comprende más fácilmente el porqué de esto si ahora se considera que el mapa de la figura 6.35 es el equivalente mapa-K del anterior.

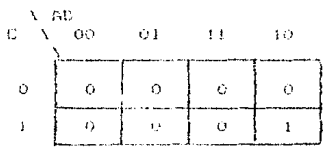


Fig. 6.35 Mapa-K equivalente al de la fig. 6.34.

2) Se puede también encontrar que en un mapa de variable introducida existe una condición de "no importa" acompañado a la variable introducida, tal y como se muestra en la figura 6.36

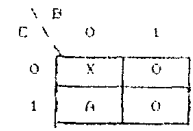


Fig. 6.36 Mapa de variable introducida.

En este caso, para obtener la ecuación booleana correspondiente, se agrupan la variable introducida y la condición de "no importa" (Fig. 6.37); se simplifica como ya es costumbre y, se multiplica al término resultante por la variable introducida, obteniéndose así el mapa y la ecuación.

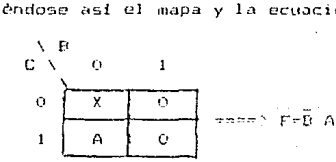


Fig. 6.37 (a) Simplificación de un mapa de variable introducida.

Lo cual es deducible precisamente del equivalente mapa-K de la figura 6.37 (b).

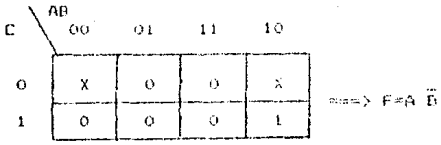


Fig. 6.37 (b) mapa-K equivalente al de la fig. 6.37 (a).

La figura 6.38 da cuenta de un ejemplo como el que se ha tratado.

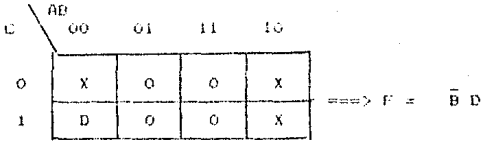


Fig. 6.38 Mapa de variable introducida y expresión correspondiente.

Comprendidos estos dos casos, se describirá ahora a un mètodo general que habrá de permitir leer cuales quiera mapa de variable introducida.

El mètodo en cuestión, consta de dos etapas: la primera consiste en leer todos los 1's existentes, tomando las variables introducidas como si fuesen 0's; debe intentarse la mayor simplificación posible, encerrando las cifras en óvalos como si se tratara de un mapa-K.

En la segunda etapa, todos los 1's se toman como condiciones de "no importa" y, con su ayuda se simplifican las variables introducidas.

El resultado final se obtiene al sumar los resultados parciales obtenidos en cada una de las etapas.

Para aclarar el procedimiento descrito, veánse los siguientes ejemplos:

Ej. 1) Simplificar el mapa de la figura 6.39.

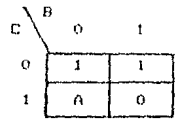
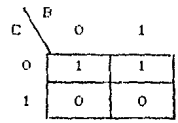


Fig. 6.39 mapa de variable introducida.

Se resolverá el problema por etapas, según se ha indicado:

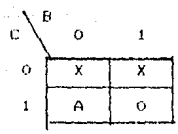
1a. etapa:



Las variables introducidas se toman como ceros.

$$\Longrightarrow F_1 = \bar{C}$$

2a. etapa:



$$\Longrightarrow F_2 = A\,\bar{B}$$

Se consideran los 1's como condiciones de "no importa".

Finalmente se suman lo resultados de cada etapa:

$$f_1 + f_2 = F = \bar{C} + A\,\bar{B}$$

Ej 2). Sea el mapa de la figura 6.40.

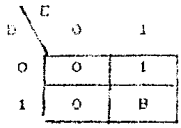
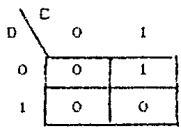


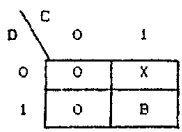
Fig. 6.40 Mapa de variable introducida.

1a. etapa:



$\Longrightarrow F_1 = C\,\bar{D}$

2a. etapa:



$\Longrightarrow F_2 = C\,B$

$$\therefore F = f_1 + f_2 = C\,\bar{D} + C\,B$$

A continuación se presenta un ejemplo más complejo donde se reduce, por medio de mapas de variable introducida, una función F de 5 variables (A, B, C, D y E) cuya tabla de verdad se muestra en la figura 6.41; en la columna F' se muestra la función con la variable introducida E, y en la columna F" se consideran como variables introducidas D y E. en las figuras 6.41 (b) y 6.41 (c) aparecen los mapas en sus 2 etapas y la función correspondiente.

A	B	C	D	E	F	F'	F"
0	0	0	0	0	0		
0	0	0	0	1	0	0	
0	0	0	1	0	1		D
0	0	0	1	1	1	1	
0	0	1	0	0	1		
0	0	1	0	1	1	1	
0	0	1	1	0	1		1
0	0	1	1	1	1	1	
0	1	0	0	0	0		
0	1	0	0	1	1	E	
0	1	0	1	0	0		E
0	1	0	1	1	1	E	
0	1	1	0	0	0		
0	1	1	0	1	1	E	
0	1	1	1	0	0		E
0	1	1	1	1	1	E	
1	0	0	0	0	0		
1	0	0	0	1	0	0	
1	0	0	1	0	0		DE
1	0	0	1	1	1	E	
1	0	1	0	0	1	1	
1	0	1	0	1	1		1
1	0	1	1	0	1	1	
1	0	1	1	1	1		
1	1	0	0	0	0		
1	1	0	0	1	1	E	
1	1	0	1	0	0		
1	1	0	1	1	1	E	E
1	1	1	0	0	1	1	
1	1	1	0	1	1		
1	1	1	1	0	0		$\bar{D}$
1	1	1	1	1	0	0	

Fig. 6.41 (a) Reducción por mapas de variable introducida.

F'

CD \ AB	00	01	11	10
00	0	E	E	0
01	1	E	E	E
11	1	E	0	1
10	1	E	1	1

1a. etapa.

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	0	0	1
10	1	0	1	1

$F' = \bar{B}C + AC\bar{D} + \bar{A}\bar{B}D$

2a. etapa.

CD \ AB	00	01	11	10
00	0	E	E	0
01	X	E	E	E
11	X	E	0	X
10	X	E	X	X

$F' = \bar{B}C + AC\bar{D} + \bar{A}\bar{B}D + \bar{A}BE + B\bar{C}E + \bar{B}DE$

1a. etapa

Fig. 6.41 (b) Cont.

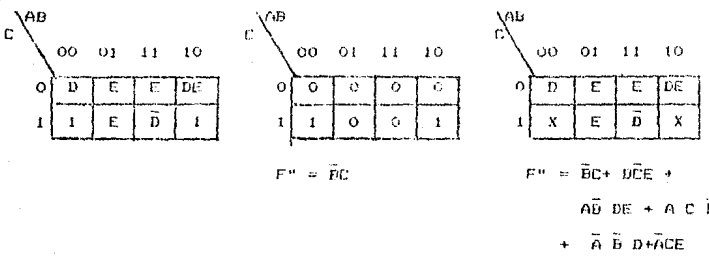


Fig. 6.41 (c) Cont.

Del ejemplo anterior se han obtenido dos expresiones booleanas igualmente complicadas, lo que muestra que el considerar más o menos variables introducidas no simplifica la función; el número de variables introducidas que serán utilizadas estará determinado por las condiciones de cada problema en particular, como se verá en capítulos posteriores.

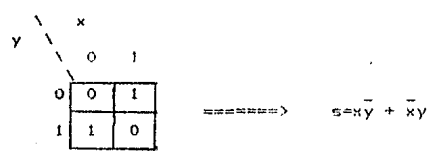
CAPITULO VII

IMPLEMENTACION DE FUNCIONES CON CIRCUITOS DE BAJA Y MEDIA ESCALA DE INTEGRACION

En esta tabla, los sumandos están representados por las variables x, y, el resultado de la suma por la variable s, y el acarreo por c.

Para obtener tanto la expresión de s como de c, se hará uso del método de los mapas de Karnaugh; como se muestra en la figura 7.2.

Para S



Para C

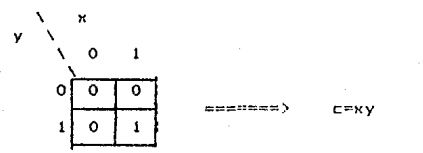


Fig. 7.2 Mapas-K para un sumador de 2 bits.

Si ahora se implementa a s y a c se tendrá el sumador, aunque el problema ahora es precisamente decidir qué tipo de compuertas usar. En la fig. 7.3 se presenta una de las posibles implementaciones de s.

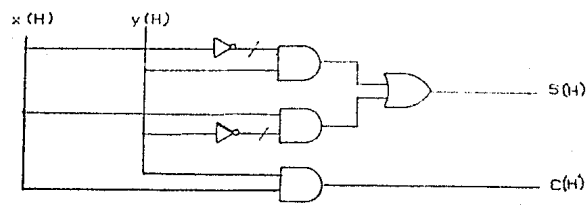


Fig. 7.3 Implementación de un sumador de 2 bits.

IMPLEMENTACION DE FUNCIONES CON CIRCUITOS DE BAJA Y MEDIA ESCALA DE INTEGRACION.

En esta sección se estudiarán algunos de los circuitos básicos que frecuentemente se requieren en el diseño de dispositivos lógicos complejos; se verá cómo se implementan estos circuitos a partir de elementos de baja escala de integración (compuertas AND, OR, XOR, etc.). Dada su gran utilidad, en el mercado se les puede encontrar ya implementados como circuitos de mediana escala de integración, es decir, como "chips", lo que hace necesario tratar las características de mayor importancia de dichos chips aunque para una descripción detallada se debe consultar el manual correspondiente.

Estos circuitos son manufacturados por diferentes compañías; las características que en este texto se mencionen se basan en las especificaciones propias de los diseños de la Texas Instruments Inc.

Sumadores.- La forma más simple de adición que se puede realizar es la de dos números de un solo bit, expresada mediante las siguientes cuatro posibilidades:

```
  0      0      1      1    ------------Sumando  X
+     +      +      +
  0      1      0      1    ------------Sumando  Y
-----  -----  -----  -----
  0      1      1     10    ------------Suma     S
                          |
                          |_____________Acarreo  C
```

Lo anterior puede resumirse en una tabla de verdad como la que sigue, en la que se han considerado a todas las variables como verificadas altas:

X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig. 7.1 Tabla de verdad de un sumador de 2 bits.

Así pues, a circuitos como éste, que satisfacen la tabla de verdad de la figira 7.1 se le denomina comúnmente como Half-Adders (H/A), o medio-sumadores, que en bloque se representan en la figura 7.4.

Fig. 7.4 Medio sumador (Half-Adder).

Véase ahora qué sucede cuando se intenta realizar una suma de tres números de un bit mediante un arreglo de medios sumadores.

Supóngase entonces que se suma X, Y, Z, obteniendo como resultado s_t, la suma total, y acarreo c_t :

$$\begin{array}{r} x \\ + \\ y \\ z \\ \hline C_t \quad S_t \end{array}$$

Divídase el problema en dos partes: primero súmese x con y usando un H/A, obteniendo una suma parcial s_p y un acarrero c_p

$$\begin{array}{r} x \\ + \\ y \\ \hline C_p \; S_p \end{array}$$

y ahora súmese s_p con z:

$$\begin{array}{r} s_p \\ + \; z \\ \hline C_{p+1} S_t \end{array}$$

donde $c_t = c_p + c_{p+1}$ será el acarreo total

En la implementación no será necesario usar otro H/A para sumar los acarreos, ya que nunca se presenta el caso de que $S_{n}=C_{n-1}=1$, y por tanto será suficiente con una compuerta OR. Tomando en cuenta todas estas consideraciones, el circuito de un sumador de tres bits sería el de la figura 7.5.

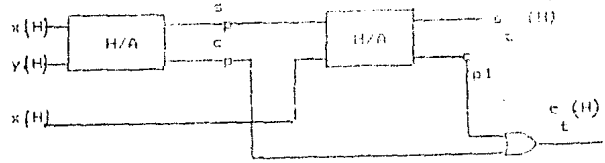


Fig. 7.5 Sumador de tres bits.

Otra manera de diseñar este circuito es a partir de su tabla de verdad, de la cual se obtiene la información para elaborar los mapas de Karnaugh para s y c, obteniendo así una expresión algebraica para cada una de estas variables (correspondientes a s y c).
 t t

Este proceso, así como la implementación que con ello se obtiene se muestra en la figuras 7.6(a) y 7.6(b).

X	Y	Z	S	C
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

TABLA DE VERDAD

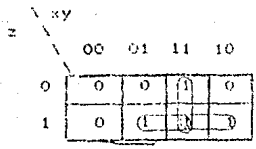
mapa-K para s

z \ xy	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$\Longrightarrow s = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$

Fig. 7.6 a) Implementación de un sumador completo

mapa-K para c



c = xy + zy + yz

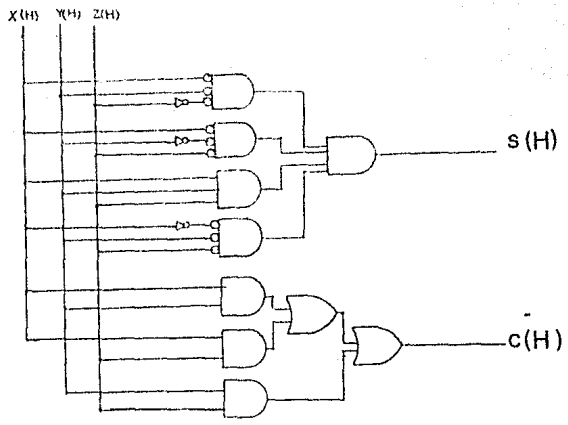


Fig. 7.6 (b) Diseño de un sumador completo.

A los circuitos de las figuras 7.5 y 7.6 se les denomina sumadores completos o " Full-Adders (F/A)" y se les representa normalmente en un bloque como el siguiente:

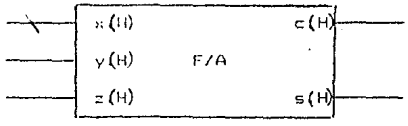


Fig. 7.7 Sumador completo.

Ambos, el medio sumador y el sumador completo, se convierten en útiles herramientas cuando, por ejemplo, se quiere sumar dos números, x, y, de 4 o más bits cada uno, como en el ejemplo que se presenta en la figura 7.8.

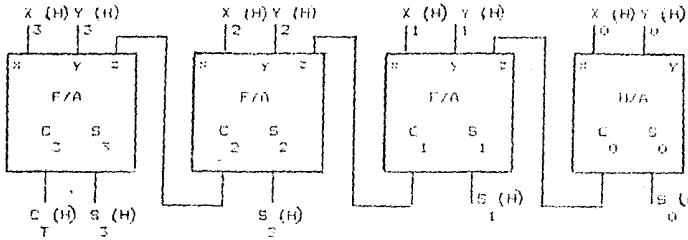
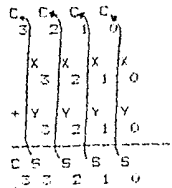


Fig. 7.8 Sumador de 2 números de 4 bits.

El diagrama muestra la implementación de un sumador que realiza la siguiente operación:



Ahora bien, como ya antes se mencionó, estos circuitos se pueden encontrar ya implementados en un chip. Según las necesidades y aplicaciones de los mismos existen diversas presentaciones, cada una ligeramente diferente de las otras. A continuación se presentan las características más importantes de los cuatro tipos de sumadores que se pueden encontrar usualmente en el mercado.

1) Sumadores completos de números de un sólo bit.

Este es un circuito que realiza la suma de dos números de un bit, A y B; C representa un posible acarreo inicial. El circuito cuenta con dos entradas complementadas A* y B*, y el resultado de la suma se representa por el símbolo Σ , siendo $\overline{\Sigma}$ su complemento; $\overline{C}_{n+1}$ corresponde al complemento del acarreo final.

En el manual "Texas Instrument TTL NSI "se le encuentra designado con los números SN5480 y SN7480. En la figura 7.9 se representa su tabla de verdad y la configuración de su diseño.

Entradas			Salidas		
C	B	A	$\overline{C}_{n+1}$	Σ	$\overline{\Sigma}$
L	L	L	H	H	L
L	L	H	H	L	H
L	H	L	H	L	H
L	H	H	L	H	L
H	L	L	H	L	H
H	L	H	L	H	L
H	H	L	L	H	L
H	H	H	L	L	H

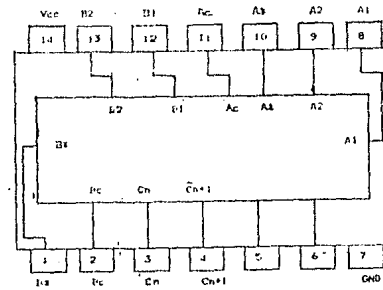
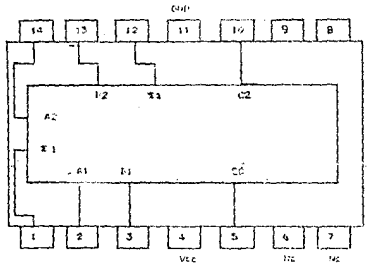


Fig. 7.9 sumador completo de 1 bit.

2) Sumadores completos de números de dos bits.

Este circuito cuenta con cinco entradas, una para el acarreo inicial C_0, y las cuatro restantes para los sumandos: A_1, A_2 ,B_1 y B_2 . El resultado de la suma se obtiene en las salidas Σ_1 y Σ_2 , en tanto que el acarreo final en C_2. Su clasificación en el manual corresponde a los números SN5482 y SN7482 (Fig. 7.10).

Entradas				Salidas							
				$C_0 = L$			$C_0 = H$				
B_2	A_2	B_1	A_1	C_2	Σ_2	Σ_1	C_2	Σ_2	Σ_1		
L	L	L	L	L	L	L	L	L	H		
L	L	L	H	L	L	H	L	H	L		
L	L	H	L	L	L	H	L	H	H		
L	L	H	H	L	H	L	H	L	L		
L	H	L	L	L	H	H	H	L	H		
L	H	L	H	L	H	L	H	H	L		
L	H	H	L	L	H	H	L	H	H		
L	H	H	H	H	L	L	L	H	L		
H	L	L	L	L	H	L	H	H	H		
H	L	L	H	L	H	H	H	H	L		
H	L	H	L	H	L	L	H	L	H		
H	L	H	H	H	L	H	H	L	L		
H	H	L	L	H	L	L	H	H	H		
H	H	L	H	H	L	H	H	H	L		
H	H	H	L	H	H	L	H	H	H		
H	H	H	H	H	H	H					



A2 A1
+ B2 B1
C2 Σ Σ
2 1

Fig. 7.10 Sumador completo de 2 bits.

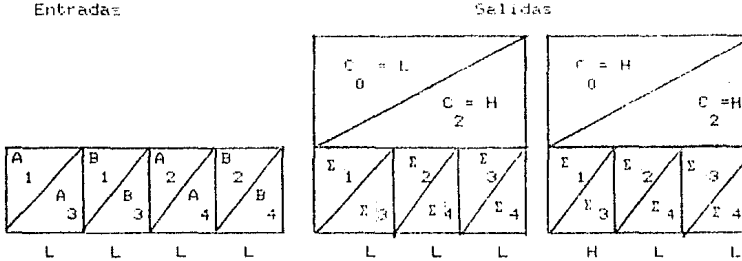
3)Sumadores completos de números de cuatro bits.

Este circuito realiza la suma de dos números A y B de cuatro bits cada uno A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4.

El resultado se obtiene a la salida de [illegible] . Cuenta además con una entrada para un acarreo inicial y una salida para el acarreo final (C_0 y C_4 respectivamente). Se encuentra catalogados bajo los números:

SN54LS83A	SN74LS83A
SN54LS283	SN74LS283
SN54S283	SN74S283
SN5483A	SN7483A
SN54283	SN74283

Como la configuración de cada uno de estos circuitos varía según el número, sólo se presentará la tabla de verdad que es común a todos ellos (Fig. 7.11).



L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

Fig. 7.11 Tabla de verdad del sumador completo de 4 bits.

4) Sumadores completos dobles de números de un bit .

Este circuito contiene dos medio sumadores para números de un mínimo de un bit; cada sumador cuenta con dos entradas A y B para los sumandos y una más para el acarreo final C_{n-1} . Se

encuentra catalogado con los números SN54H183 y SN74H183. Su configuración se muestra en la fig. 7.12 omitiéndose la tabla de verdad por ser ésta la misma que se presentó en el inciso 1.

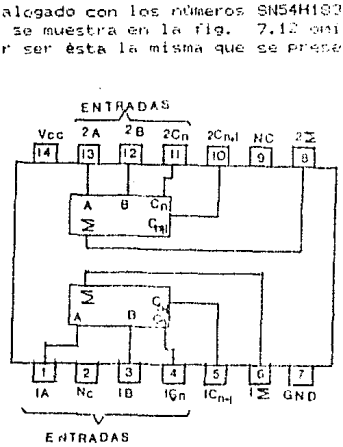


Fig. 7.12 Sumador completo de 1 bit, doble.

Comparadores.- Sucede con frecuencia que en el diseño lógico se presentan situaciones en las cuales existe la necesidad de un circuito comparador; éstos circuitos realizan la comparación de dos cantidades, digamos A y B, para lo cual cuentan con tres salidas que funcionan como banderas o indicadores, señalado así cada uno de los tres posibles resultados de la comparación: A>B, A<B o A=B.

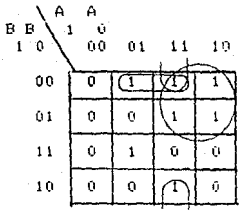
A continuación se muestra el diseño de un circuito comparador para números de dos bits: los números son $A=A_1A_0$ y $B=B_1B_0$ Como ya antes se ha hecho, es a partir de la tabla de verdad que se buscará la implementación del circuito, elaborando así los mapas-K que dan la forma explícita de las funciones de salida. Véase esto en la figura 7.13 (a) y (b)

TABLA DE VERDAD

Entradas				Salidas		
				A>B	A=B	A<B
A 1	A 0	B 1	B 0	S 2	S 1	S 0
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Mapas-K para cada una de las salidas

Para S (A>B)
2



=====> S = A $\overline{B}$ + A $\overline{B}$ $\overline{B}$ + A A $\overline{B}$
2 1 1 0 1 0 1 0 0

Fig 7.13 (a) Mapas-k de comparadores

Para S (A=B)
1

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

$$S = \bar{A}\,\bar{A}\,\bar{B}\,\bar{B} + \bar{A}\,A\,\bar{B}\,B + A\,A\,B\,B + A\,\bar{A}\,B\,\bar{B}$$
$$1 \quad 1\,0\,1\,0 \quad 1\,0\,1\,0 \quad 1\,0\,1\,0 \quad 1\,0\,1\,0$$

$$= \bar{A}\,\bar{B}\,(\bar{A}\,\bar{B} + A\,B) + A\,B\,(\bar{A}\,\bar{B} + A\,B)$$
$$1\,1 \quad 0\,0 \quad 0\,0 \quad 1\,1 \quad 0\,0$$

$$= (\bar{A}\,\bar{B} + A\,B)\,(\bar{A}\,\bar{B} + A\,B)$$
$$1\,1 \quad 1\,1 \quad 0\,0 \quad 0\,0$$

$$\therefore \quad S = (A \odot \bar{B})\,(A \odot \bar{B})$$
$$1 \quad 1 \quad 1 \quad 0 \quad 0$$

Para S (A < B
0

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

$$S = \bar{A}\,B + \bar{A}\,\bar{A}\,B + \bar{A}\,B\,B$$
$$0 \quad 1\,1 \quad 1\,0\,0 \quad 0\,1\,0$$

Fig. 7.13 (b) Cont.

De acuerdo con lo anterior, el circuito puede implementarse como se muestra en la figura 7.14.

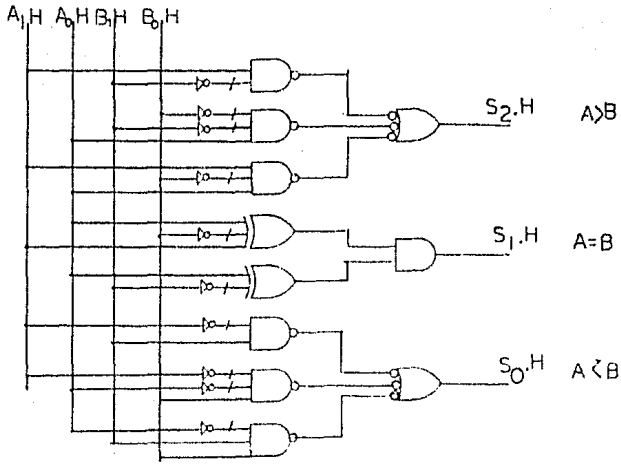
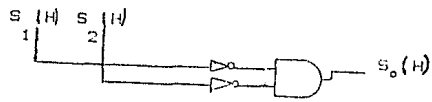


Fig. 7.14 Implementación de un comparador.

Es posible reducir un poco el diagrama si se considera que una vez que el circuito ha comparado los números, encontrando que A no es ni mayor ni igual a B, se puede evitar que vuelva a compararlos para decidir si A es menor que B implementando la función $S = \overline{S}_0 \; \overline{S}_1 \; \overline{S}_2$:



Ahora bien, con frecuencia lo que se necesita es comparar números de más de dos bits; diseñar el circuito para estos casos con el procedimiento que hasta ahora se ha seguido sería bastante complicado, razón por la cual lo que se hará es acoplar varios comparadores de dos bits, modificados ligeramente.

Sean pues A y B los números a comparar, cada uno de cuatro bits, considerando a cada número dividido en dos partes:

```
A  A      A  A
 3  2      1  0

B  B      B  B
 3  2      1  0
```

Compárense primero los bits menos significativos: (Fig 7.15)

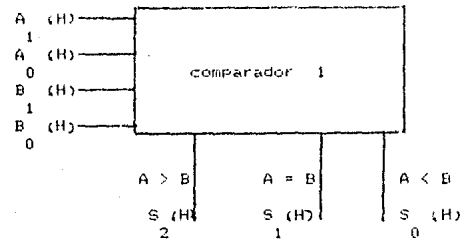


Fig. 7.15 Comparador de 2 números.

Compárense ahora las partes más significativas de los números usando un circuito que además de tener lo que el anterior, posee tres entradas extras, como el comparador 2 de la figura 7.16.

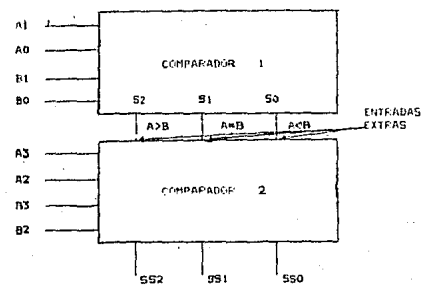


Fig. 7.16 Comparadores acoplados.

Estas entradas extras funcionan de tal manera que, si $A_1A_2 = B_1B_2$, el estado de las salidas SS_0, SS_1 y SS_2 , es el mismo que el de la entradas S_0, S_1 y S_2 respectivamente; por el contrario, si se tiene que $A_1A_2 \neq B_1B_2$, el circuito se comportará como el comparador 1.

Así, por ejemplo, si A = 0110 y B = 0100, primero se compara A_1A_0= 10 con B_1B_0 = 00, obteniéndose que A > B, es decir, S_2 = 1 y S_1=S_0= 0. A continuación se comparan las partes más significativas, pero como son iguales, se tiene que SS_0, SS_1 y SS_2 son un reflejo de S_0, S_1 y S_2 , esto es, SS_2 = 1,que implica que A>B.

Tómese otro caso. Sean A = 0010 y B = 0100. Al comparar las partes menos significativas se obtiene que A>B esto es. S_2=1: al pasar ahora a la parte más significativa se tiene que, como A<B, el comparador ignora sus entradas S_0, S_1 y S_2, para trabajar en la forma habitual, dándo como resultado que A<B.

De esta manera, se puede encontrar en el mercado comparadores para números de cuatro bits, que según las necesides, pueden conectarse en cascada para abarcar números más grandes. Estos circuitos poseen la particularidad de que la entrada denominada A=B correspondiente a la parte menos significativa de los números, debe ser puesta en el nivel de voltaje alto. Los números de clasificación de estos circuitos, así como de otros para números de más o menos bits, se representan en la siguiente lista:

SN54S85	SN74S85
SN5485	SN7485
SN54LS85	SN74LS85
SN54L85	SN74L85

En la figura 7.17 se muestan, como ejemplo, algunas de las tablas electrónicas de los comparadores de cuatro bits, la primera parte de la tabla es común a todos estos modelos, la segunda parte se anexa a la primera para los comparadores 'S85, 'LS85 y 'S85, en tanto que si es el 'L85 es la tercera parte de la tabla la que se anexa.

Entradas						Salidas Cascada			Salidas		
A B	A B	A B	A B			A>B	A<B	A=B	A>B	A<B	A=B
3 3	2 2	1 1	0 0								
A >B	X	X	X			X	X	X	H	L	L
3 3											
A <B	X	X	X			X	X	X	L	H	L
3 3											
A =B	A >B	X	X			X	X	X	H	L	L
3 3	2 2										
A =B	A <B	X	X			X	X	X	L	H	L
3 3	2 2										
A =B	A =B	A >B	X			X	X	X	H	L	L
3 3	2 2	1 1									
A =B	A =B	A <B	X			X	X	X	L	H	L
3 3	2 2	1 1									
A =B	A =B	A =B	A >B			X	X	X	H	L	L
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A <B			X	X	X	L	H	L
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			H	L	L	H	L	L
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			L	H	L	L	H	L
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			L	L	H	L	L	H
3 3	2 2	1 1	0 0								
'85, 'LS85, 'S85											
A =B	A =B	A =B	A =B			X	X	H	L	L	H
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			H	H	L	L	L	L
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			L	L	L	H	H	L
3 3	2 2	1 1	0 0								
'LS85											
A =B	A =B	A =B	A =B			L	H	H	L	H	H
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			H	L	H	H	L	H
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			H	H	H	H	H	H
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			H	H	L	H	H	L
3 3	2 2	1 1	0 0								
A =B	A =B	A =B	A =B			L	L	L	L	L	L
3 3	2 2	1 1	0 0								

Fig. 7.17 Tablas para comparadores de 4 bits.

Multiplexores.- El término multiplexar se refiere al hecho de transmitir un gran número de unidades de información a través de un número pequeño de canales o líneas; los circuitos que sirven a este propósito se denominan multiplexores o selectores de datos y poseen varias entradas y una salida, cuyo estado está determinado por un conjunto de líneas selectoras, por medio de las cuales se puede escoger o seleccionar una de las entradas cuyo estado se verá reflejado en la salida.

En los esquemas de la figura 7.18 se representa a un multiplexor 2 a 1 (2 entradas a 1 salida), los dos posibles estados que puede tener la línea selectora y cómo se comporta el multiplexor en cada caso.

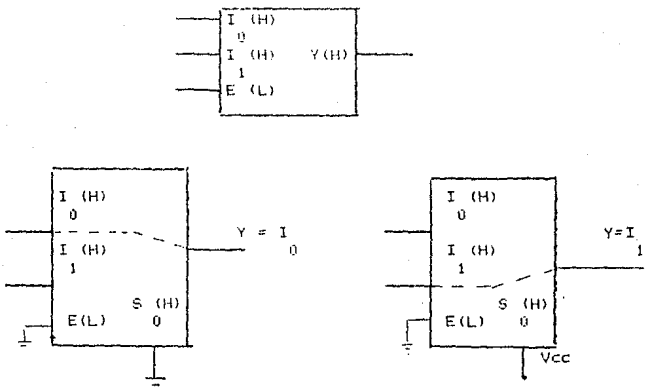


Fig. 7.18 Comportamiento de un multiplexor.

Este multiplexor está constituido internamente por el circuito de la figura 7.19.

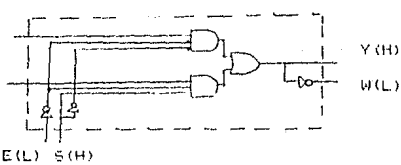


Fig. 7.19 Circuito Multiplexor.

Cuando la entrada E, llamada "enable" o "strobe" (habilitadora), está en un estado alto, la salida Y se coloca en un nivel de voltaje bajo, sin importar el estado de ninguna de las entradas. En el dibujo se muestra una salida extra, W, que es verificada baja.

Existen diversos tipos de multiplexores, de los cuales los más comunes son:

16 a 1 (16 entradas-una salida)
8 a 1 (8 entradas-una salida)
4 a 1 (4 entradas-una salida)
2 a 1 (2 entradas-una salida)

Modelos de multiplexores.- En el mercado se puede encontrar esencialmente cuatro tipos de multiplexores, cada uno de ellos en diversos modelos que se clasifican numéricamente según se ha visto con otros circuitos. Por ahora sólo se mencionarán las características principales de cada tipo y no se incluirán descripciones de un cierto tipo de multiplexores que cuentan con líneas que comunmente se denominan de tres estados.

1) Multiplexores 16 a 1.

Este circuito selecciona una de las 16 líneas de entrada por medio de cuatro líneas selectoras. Su clasificación corresponde a los números SN54150 y SN74150 y su configuración se muestra en la figura 7.20 (a) y (b)

ENTRADAS SELECTORAS				STROBE	SALIDA
D	C	B	A	S	W
X	X	X	X	H	H
L	L	L	L	L	E0
L	L	L	H	L	E1
L	L	H	L	L	E2
L	L	H	H	L	E3
L	H	L	L	L	E4
L	H	L	H	L	E5
L	H	H	L	L	E6
L	H	H	H	L	E7
H	L	L	L	L	E8
H	L	L	H	L	E9
H	L	H	L	L	E10
H	L	H	H	L	E11
H	H	L	L	L	E12
H	H	L	H	L	E13
H	H	H	L	L	E14
H	H	H	H	L	E15

Fig. 7.20 (a) Tabla de comportamiento de un multiplexor 16 a 1

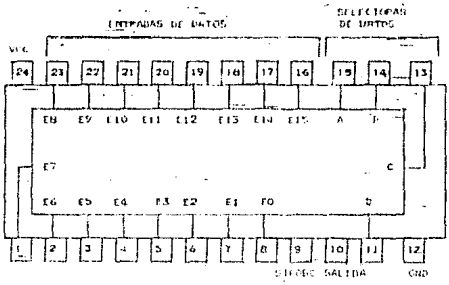


Fig. 7.20 (b) CI. de un multiplexor 16 a 1.

1) Multiplexores 8 a 1.

Este circuito selecciona una de las 8 líneas de entrada por medio de tres líneas selectoras de datos. Este tipo de multiplexor se presenta principalmente en dos formas distintas, los que cuentan con los números:

SN54S151	SN74S151	
SN54151A	SN7151A	
SN54LS151	SN74LS151	(a)

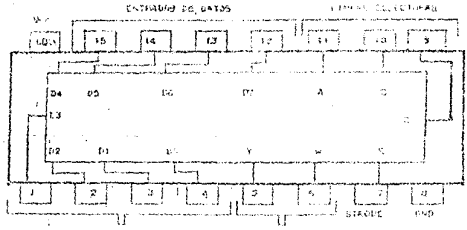
tienen una entrada de enable o strobe, mientras que los números

SN54S251	SN74S251	
SN54251	SN74251	
SN54LS251	SN74LS251	(b)
SN54152A	SN74LS152	

carecen de ella.

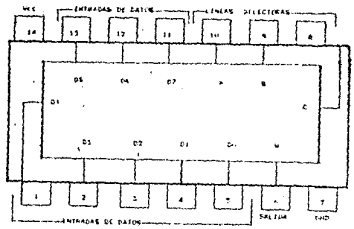
Ambos circuitos se muestran en la figura 7.21.

a)



ENTRADAS			STROBE	SALIDAS	
SELECTORAS					
C	B	A	S	Y(H)	W(L)
X	X	X	H	L	H
L	L	L	L	D0	D0
L	L	H	L	D1	D1
L	H	L	L	D2	D2
L	H	H	L	D3	D3
H	L	L	L	D4	D4
H	L	H	L	D5	D5
H	H	L	L	D6	D6
H	H	H	L	D7	D7

b)



ENTRADAS	
SELECTORAS	SALIDAS
C B A	W
L L L	D0
L L H	D1
L H L	D2
L H H	D3
H L L	D4
H L H	D5
H H L	D6
H H H	D7

Fig. 7.21 CI's Multiplexores de 8 a 1 a) con linea de strobe b) Sin linea de strobe.

3) Multiplexor doble, 4 a 1.

Cada chip contiene dos multiplexores con líneas de datos y strobe independientes, aunque las líneas selectoras están compartidas. Se le encuentra clasificado con los números.

SN54S153	SN74S154
SN54153	SN74153
SN54LS153	SN74LS153
SN54L153	SN74L153

En las figuras 7.22 (a) y (b) se muestran su tabla de verdad y su configuración.

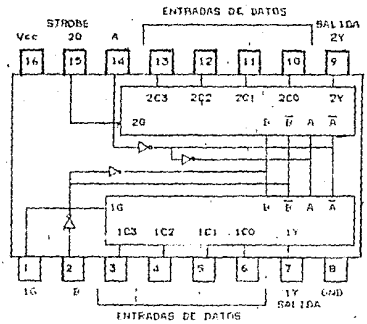


Fig. 7.22 (a) Multiplexor doble 4 a 1.

Lineas selectoras		Entradas de datos				Strobe	Salida
B	A	C 0	C 1	C 2	C 3	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	H
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

Fig 7.22 (b) Cont.

4)Multiplexor cuádruple, 2 a 1.

Este circuito consta de cuatro multiplexores de 2 a 1, todos ellos con una misma línea de strobe y de salida. Se les reconoce por los números

SN54S157	SN74S157
SN54LS158	SN74LS158
SN54LS157	SN74LS157
SN54157	SN74157
SN54L157	SN74L157

de los cuales, los terminados en 158 poseen salidas negadas.

Configuración y tabla de verdad de estos circuitos se muestran en la figura 7.23 (a) y (b)

Entradas				Salidas	
Strobe	Selectores	A	B	'157, 'L157 'LS157, 'S151	'LS158 'S158
H	X	X	X	L	H
L	L	L	X	L	H
L	L	H	X	H	H
L	H	X	L	L	H
L	H	X	L	H	L

Fig 7.23 (a) Multiplexores cuadruples de 2 a 1

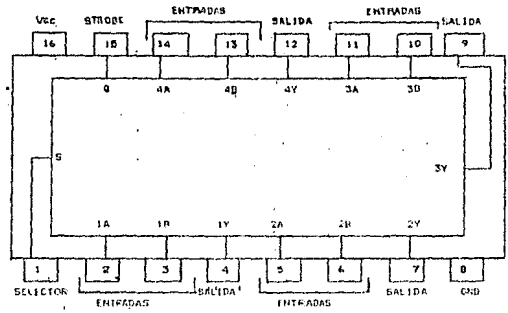
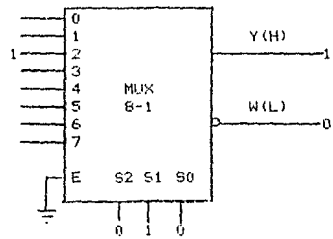


Fig. 7.23 (b) Cont.

Como ejemplo de qué se puede hacer con un multiplexor 8 a 1, Supóngase que se tienen ocho líneas y que se necesita enviar cierta información por sólo una de ellas:



Así pues, va a interesar el estado de la línea 2, para seleccionarla se hace S2=0, S1=1 y S0=0, de esta manera el estado de las línea Y es un reflejo de la línea 2 y W es su complemento.

IMPLEMENTACION DE FUNCIONES CON MULTIPLEXORES

Ahora bien, además de cumplir la función de transmitir información selectivamente, un multiplexor tiene otros usos, lo cual se verá a continuación.

Supongase que se tiene un multiplexor de 4 a 1, tal como el que se muestra en la figura 7.24 acompañado de la tabla electrónica que describe su comportamiento.

E	S1	S0	Y	W
1	X	X	0	1
0	0	0	I0	I0
0	0	1	I1	I1
0	1	0	I2	I2
0	1	1	I3	I3

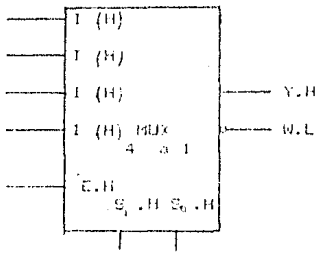


Fig. 7.24 Multiplexor 4 a 1.

Se pretende ahora implementar con este multiplexor la función descrita en la siguiente tabla de verdad.

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

Para lo cual se debera realizar el mapa de variable introducida mostrado en la figura 7.25.

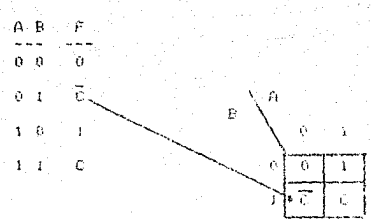


Fig. 7.25 Mapa para implementación con multiplexores.

Si ahora se conectan los bits más significativos (A y B) a las líneas selectoras S1 y S0 respectivamente, el mapa se modificará según se muestra en la figura 7.26.

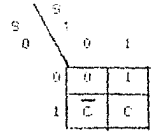


Fig. 7.26 Mapa para implementación con MUX.

De esta manera, si S_0=1 y S_1=0 se obtiene el minitérmino m_1, y la entrada seleccionada es I1, lo que implica que la salida y del multiplexor deberá tomar el valor de $\bar{C}$, que es el correspondiente al minitérmino m_1. Realizando este razonamiento para cada caso (minitérmino), se obtendrá la información que se encuentra especificada en la tabla de la fig. 7.27.

S 0	S 1	minitermino	Entrada seleccionada	Valor de la entrada selecionada
0	0	m_0	I_0	0
0	1	m_1	I_1	$\bar{C}$
1	0	m_2	I_2	1
1	1	m_3	I_3	C

Fig. 7.27 Comportamiento del multiplexor.

Lo que conduce al siguiente circuito de la figura 7.28.

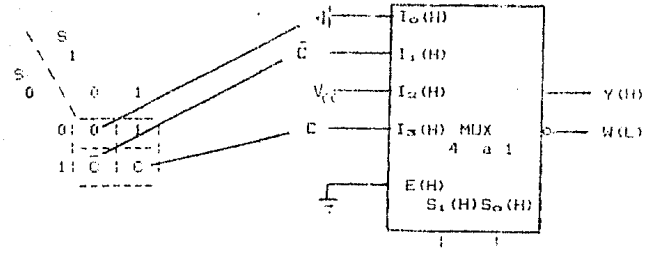


Fig. 7.28 Implementación con un multiplexor.

En la figura 7.29 se muestra otro ejemplo de implementación, pero ahora con un multiplexor 8 a 1.

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

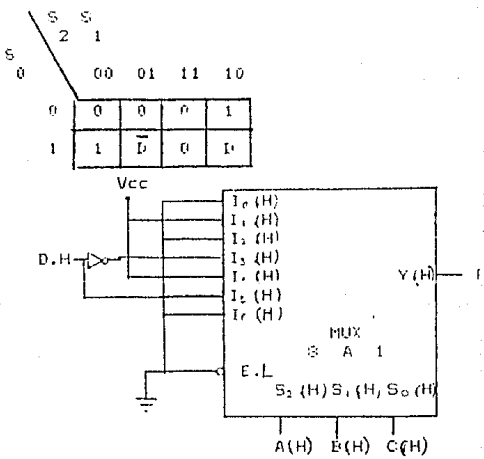


Fig. 7.29 Ejemplo de implementación con multiplexor.

Implementación de funciones con multiplexores usando lógica mixta.

En el diseño de los multiplexores se ha adoptado la convención de que tanto las entradas como las salidas sean verificadas altas; sin embargo, a pesar de esto es posible implementar con ellos funciones cuyas variables sean verificadas bajas. A continuación se verá cómo hacerlo.

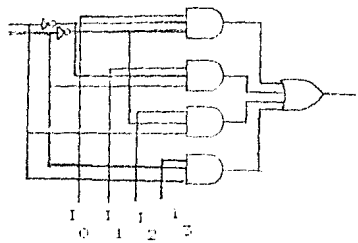
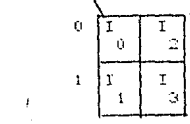


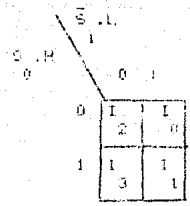
Fig. 7.30 Multiplexor 4 a 1.

Si se tiene el multiplexor de la figura 7.30 el mapa-K expresado en términos de sus entradas será el siguiente:



donde se ha supuesto que tanto S1 como S0 son verificadas altas.

Supóngase ahora que la variable que se conecta a la línea S es verificada baja, por lo que se supondrá por un momento que S1 también es verificada baja; analícese cómo sería el mapa-K en este caso revisando cada una de las cuatro posibilidades que hay, para lo cual recuérdese que S(L)=S(H) por lo que se sustituye en el mapa anterior. Para expresar el mapa en términos de S1(L) sólo se requiere intercambiar las columnas, obteniendo finalmente



Como se ve, se han intercambiado las columnas -donde las variables son verificadas bajas- respecto del mapa del caso ya tratado.

Este proceso es generalizable a cualquier número de variables, como se verá en el ejemplo siguiente.

Supóngase que se tiene la función de la figura 7.31, representada por su tabla de verdad y su mapa-K, donde A y B son verificadas bajas.

A(L)	B(L)	C(H)	D(H)	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

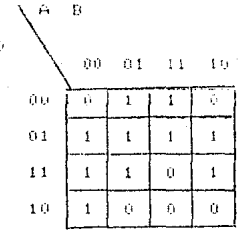


Fig. 7.31 Función a implementar con multiplexores.

Una vez que se tiene el mapa-K, el primer paso a seguir es intercambiar aquellas columnas -o renglones- en donde se presentan variables verificadas bajas, esto es:

la columna 00 se intercambia con la 11
" " 01 " " " " 10

obteniendo el mapa y la tabla de verdad de la figura 7.32.

CD \ AB	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	0	1	1	1
10	0	0	1	0

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

Fig. 7.32 Mapa con columnas intercambiadas.

y transformando el mapa-k en un mapa de variable introducida se obtiene

C \ AB	00	01	11	10
0	1	D	D	1
1	0	D	1	D

lo que nos lleva a la implementación del circuito de la figura 7.33.

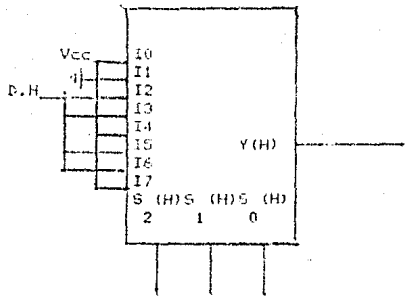


Fig. 7.33 Implementación con multiplexores.

Finalmente, y sólo por completez, se verael siguiente caso: supóngase que se tiene una función de tres variables A,B y C, misma que se encuentra representada por el siguiente mapa-K.

C \ AB	00	01	11	10
0	1	0	1	0
1	0	1	1	0

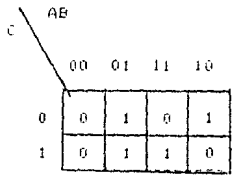
Si, por ejemplo, C es verificada baja, se deben intercambiar los renglones, esto es,

C \ AB	00	01	11	10
0	0	1	1	0
1	1	0	1	0

En cambio, si sólo A es verificada baja, las columnas se intercambian de la siguiente manera:

00 por 10
01 por 11

Nótese que el bit correspondiente a la variable verificada alta (B), no se altera al intercambiar las columnas, es decir,



Demultiplexores.- El término "demultiplexar" se refiere al proceso inverso de "multiplexar", esto es, recibir información de un pequeño número de canales y distribuirla sobre un número mayor. Estos circuitos poseen n líneas selectoras, 2n salidas y una sola entrada, la función de las líneas selectoras es, precisamente, seleccionar la salida en donde se reflejará el estado de la línea de entrada, la figura 7.34 muestra los esquemas de un demultiplexor con sus posibles estados.

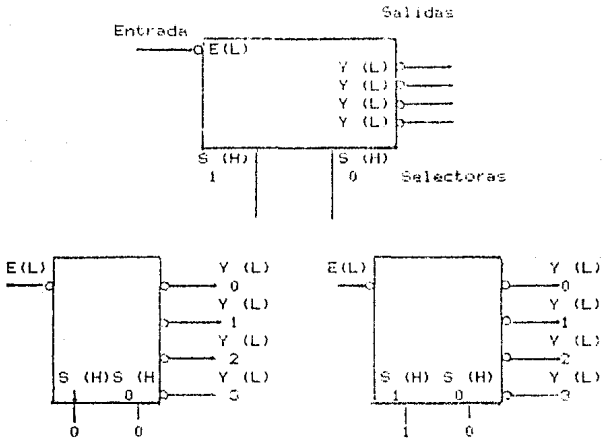


Fig. 7.34 Comportamiento de un demultiplexor.

Internamente, el demultiplexor del ejemplo se ve como se muestra en el circuito de la figura 7.35.

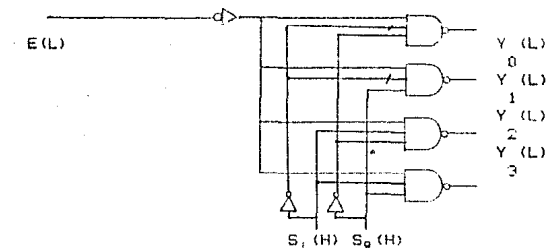


Fig. 7.35 Circuitería de un demultiplexor.

Algunos de los demultiplexores que a menudo son usados son:

1) 1 a 16 (1 entrada-16 salidas, 4 líneas selectoras)

2) 1 a 8 (1 entrada-8 salidas, 3 líneas selectoras)
3) 1 a 4 (1 entrada-4 salidas, 2 líneas selectoras)

A continuación se mencionan las características principales de cada uno de estos demultiplexores.

1) Demultiplexor de 1 a 16.
Este circuito contiene cuatro líneas selectoras A, B, C y D, por medio de las cuales se selecciona una de las 16 salidas; cuenta además con dos entradas G2 y G1, que deben permanecer en un nivel de voltaje bajo. Este demultiplexor se puede encontrar clasificado por los siguientes números:

SN54154 SN74154

SN54L154 SN74L154

La tabla de verdad y configuración correspondientes se muestran en la figura 7.36 (a) y (b) respectivamente.

Entradas						Salidas															
G2	G1	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

FIG 7.36 a) Tabla de electrónica del demultiplexor 1 a 16

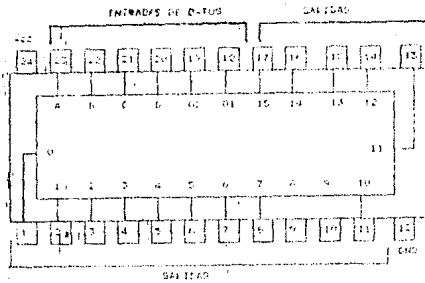


Fig. 7.36b Configuracion del demultiplexor 1 a 16.

2) Demultiplexor doble 1 a 4, con dos líneas selectoras
Este circuito está compuesto de dos demultiplexores individuales, cada uno con cuatro salidas, una entrada y dos

líneas selectoras; se le reconoce por los números
SN54LS139 SN74LS139

SN54S139 SN74S139
En las figuras 7.38 (a) y (b) se muestra su configuración y

tabla electrónica

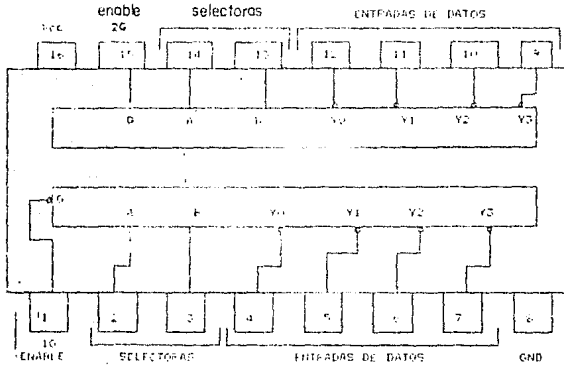


Fig 7.37 a) configuracion del demultiplexor 1 a 4

ENTRADAS			SALIDAS			
ENABLE	SELECTORAS					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

Fig. 7.37 b) Contiguración del demultiplexor 1-4.

3) Demultiplexor 1 a 4 (dual).

Este circuito está formado por dos demultiplexores 1 a 4, los cuales se encuentran relacionados de tal manera que se pueden utilizar independientemente o unidos para formar un demultiplexor de 1 a 8. Cada uno cuenta con una entrada de strobe individual, mientras que las líneas selectoras son comunes. Los strobes permiten desactivar alguna de las secciones si así se desea; las líneas 10 y 20 permiten a este dispositivo ser usado como un demultiplexor 1 a 8. Se puede identificar por los siguientes números:

SN54155	SN74155
SN54156	SN74156
SN54LS155	SN74LS155
SN54LS156	SN74LS156

En la figura 7.38 se muestra su configuración y su tabla de verdad, primero como dos demultiplexores individuales y después como uno sólo.

Entradas				Salidas				
Selector		atrobe	data					
B	A	1G	1C	1Y 0	1Y 1	1Y 2	1Y 3	
X	X	H	X	H	H	H	H	
L	L	L	H	L	H	H	H	
L	H	L	H	H	L	H	H	Sección 1
H	L	L	H	H	H	L	H	
H	H	L	H	H	H	H	L	
X	X	X	L	H	H	H	H	

Sección 2

Entradas				Salidas			
Selector		atrobe	data				
B	A	2G	2C	2Y 0	2Y 1	2Y 2	2Y 3
X	X	H	X	H	H	H	H
L	L	L	L	L	H	H	H
L	H	L	L	H	L	H	H
H	L	L	L	H	H	L	H
H	H	L	L	H	H	H	L
X	X	X	L	H	H	H	H

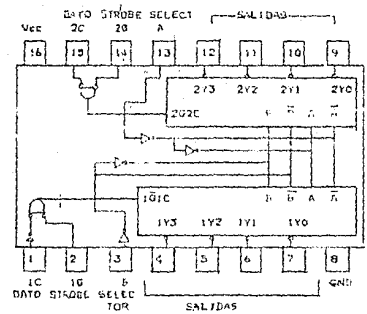


Fig. 7.38 (a) Demultiplexor dual 1 a 4

Cuando se tiene en las entradas selectores el minitérmino m0 es decir A=B=C=0, se seleccionará la salida Y0, la cual tendrá el mismo estado que la entrada E, esto es, un nivel de voltaje bajo (nótese que tanto E como Y0 son verificados bajas), lo que implica un valor lógico de 1 o verdadero que finalmente producirá un 1 lógico en la salida f.H. El mismo razonamiento se sigue para m3 y m7 , comprobándose así que el circuito en efecto representa a la función f.

Implementación de funciones con demultiplexores usando lógica mixta.- El procedimiento es el mismo que el utilizado con los multiplexores; supóngase que se tiene una función f(A(L),B(L),C(H))(H) representada por el mapa-K de la figura 7.41.

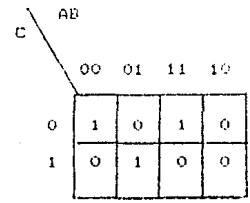


Fig. 7.40 Función "f" a implementar.

Ahora se intercambian las columnas

00 por 11

01 por 10

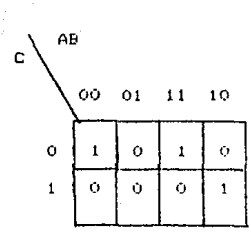


Fig. 7.41 Mapa de "f" con columnas intercambiadas.

modificándose el mapa como se muestra en la figura 7.41, que finalmente se implementa como en la figura 7.42.

Entradas				Salidas							
Selector			data								
'C	B	A	"G	(0) 2Y 0	(1) 2Y 1	(2) 2Y 2	(3) 2Y 3	(4) 1Y 0	(5) 1Y 1	(6) 1Y 2	(7) 1Y 3
X	X	X	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H
L	H	H	L	H	H	H	L	H	H	H	H
H	L	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H
H	H	L	L	H	H	H	H	H	H	L	H
H	H	H	L	H	H	H	H	H	H	H	L

'C = entradas 1C y 2C conectadas juntas

"G = entradas 1G y 2G conectadas juntas

Fig. 7.38 (b) cont.

Ahora bien, los demultiplexores también pueden ser usados en la implementación de funciones booleanas pero, a diferencia de lo que ocurre con un multiplexor, se necesitan algunas compuertas extra para realizarla. Vease ahora cómo es esto.

Supóngase que se tiene la función f= (m0, m3, m7), cuyas variables son A, B y C; su implementación con un demultiplexor 1 a 8 es la mostrada en la figura 7.39.

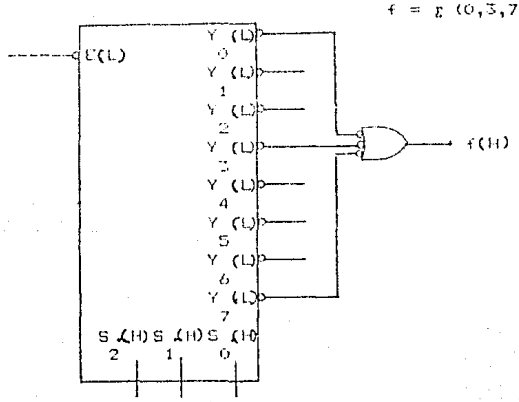


Figura 7.39.

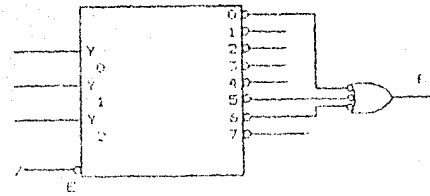
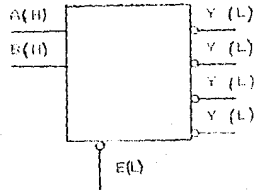


Fig. 7.42 Implementación de f.

Convertidores de códigos Los convertidores de código se dividen en 2 grandes grupos, cuando tienen n variables de entrada y m de salida tal que $m \leq 2^n$ se conocen como decodificadores, esto es cuando el número de entrada es menor que el de salidas, en cambio, cuando el convertidor tiene m entradas y n salidas, esto es, que tiene más entradas que salidas se les conoce como codificadores

Supóngase que se tiene un decodificador 2 a 4, en el cual la línea de "enable" siempre está activada, de manera que las líneas A y B seleccionen la salida que tenga un valor lógico verdadero según la tabla de la figura 7.43,



A	B	Y 3	Y 2	Y 1	Y 0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Fig.7.43 Tabla de verdad binario-decimal, decodificador 2 a 4

Lo que se tiene es un convertidor de binario a decimal, de manera similar se pueden encontrar convertidores BCD-decimal, exceso-3 a decimal, BCD-7 segmentos, etc.

Así, por ejemplo, los decodificadores clasificados con números cuyas terminaciones son '42A, 'L42 y L542, son convertidores de BCD a decimal; los de terminaciones '43A y 'L43 de exceso-3 a decimal, y los de terminación '44A y 'L44 de exceso-3-Gray-decimal.

Todos los circuitos mencionados poseen la configuración de la figura 7.44.

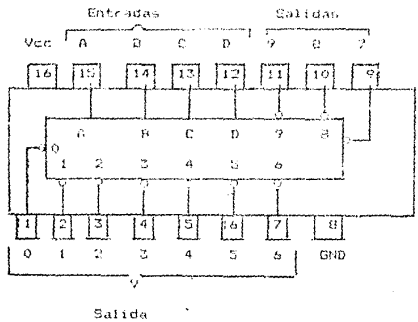


Fig. 7.44 Convertidor de códigos.

La tabla de verdad que describe su comportamiento es la presentada en la figura 7.45.

Otros convertidores de código frecuentemente usados son el BCD a 7 segmentos y el binario a 7 segmentos. Estos 7 segmentos se refieren al arreglo de "led's" de la figura 7.46, por medio del cual se visualiza un segmento de base 10.

	142A,L42	143A	144A	
	LS42	'L43	'L44	
No.	Entradas BCD	Entradas Exceso-3	Entradas Gray	Salidas Decimales (comunes)
	D C B A	D C B A	D C B A	0 1 2 3 4 5 6 7 8 9
0	L L L L	L L H H	L L H L	L H H H H H H H H H
1	L L L H	L H L L	L H H L	H L H H H H H H H H
2	L L H L	L H L H	L H H H	H H L H H H H H H H
3	L L H H	L H H L	L H L H	H H H L H H H H H H
4	L H L L	L H H H	L H L L	H H H H L H H H H H
5	L H L H	H L L L	H H L L	H H H H H L H H H H
6	L H H L	H L L H	H H L H	H H H H H H L H H H
7	L H H H	H L H L	H H H H	H H H H H H H L H H
8	H L L L	H L H H	H H H L	H H H H H H H H L H
9	H L L H	H H L L	H L H L	H H H H H H H H H L
NO VALIDAS	H L H L	H H L H	H L H H	H H H H H H H H H H
	H L H H	H H H L	H L L H	H H H H H H H H H H
	H H L L	H H H H	H L L L	H H H H H H H H H H
	H H L H	L L L L	L L L L	H H H H H H H H H H
	H H H L	L L L H	L L L H	H H H H H H H H H H
	H H H H	L L H L	L L H H	H H H H H H H H H H

Fig. 7.45 Tabla electrónica de convetidores a código decimal.

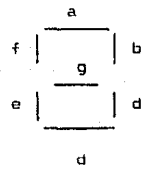


Fig. 7.46 Despliegue de 7 segmentos.

Un ejemplo de un convertidor BCD-a-7 segmentos son los circuitos con terminaciones 46A, 47A, L46, L47, LS47 cuyo comportamiento se describe en la tabla de la figura 7.47.

No.	Entradas				Salidas						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7		1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

Fig.7.47 Comportamiento de un convertidor BCD a 7 segmentos.

Supóngase, ahora que se tiene un teclado decimal, tal que cuando se presiona una de sus teclas las líneas de salida L3, L2, L1 y L0 toman el valor binario correspondiente (de 0 a 9); así, si se presiona la tecla 8 las salidas tomarán el siguiente valor L3L2L1L0=1000, y se desea visualizar este número por medio de un despliegue de 7 segmentos, que se conectará a las salidas del teclado como se muestra en la figura 7.48, y a la salida de este convertidor se conectará el despliegue de 7 segmentos; así, cada vez que se presione una tecla se podrá visualizar su valor.

Como último punto se hará notar que algunos demultiplexores pueden ser usados como decodificadores considerando las líneas selectoras del demultiplexor como entradas del decodificador y la línea de entrada del demultiplexor como enable del decodificador, tal es el caso del circuito demultiplexor 1-4 de la figura 7.38, el cual, puede ser usado como un decodificador 2 a 4

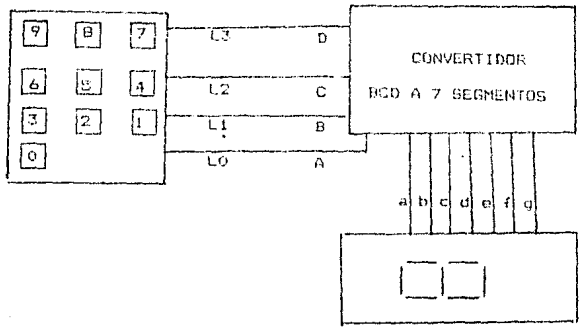


Fig 7.48 Ejemplo de circuito usando despliegues de 7 segmentos

ARREGLOS LOGICOS

Tradicionalmente los sistemas de diseño de circuitos digitales han sido particularmente eficientes en la solución de problemas que involucran pocas variables y un mayor número de términos (entendiéndose por términos funciones booleanas de las variables). Pero la mayoría de los problemas reales son exactamente al contrario, ya que involucran un gran número de variables y pocos términos, para resolver estos problemas se han desarrollado circuitos integrados muy flexibles llamados arreglos lógicos, los cuales son esencialmente de tres tipos:

Arreglos de Compuertas Programables (PGAs)

Arreglos Lógicos Programables (PLAs)

Secuenciales Lógicos Programables (PLSs)

En esta sección se expondrán en forma general los PGAs y PLAs.

Esta familia de dispositivos puede ser programada por el fabricante en caso de que el volumen de producción sea lo suficiente grande, se les conoce como "Gate Array" y llegan a tener una integración de hasta 80 000 compuertas. Para el desarrollo de prototipos existen versiones con menor integración que pueden ser programadas por el usuario, usando equipo especial, llamadas respectivamente FPGAs, FPLAs y FPLs.

Todos los arreglos lógicos tienen como estructura fundamental el circuito de la fig 7.49.

Los fusibles marcados en la figura 7.49a) conectan las líneas en las que se encuentran, lo que se indica en la figura 7.49b) con un punto; si se desea que estas líneas no estén conectadas se quema el fusible.

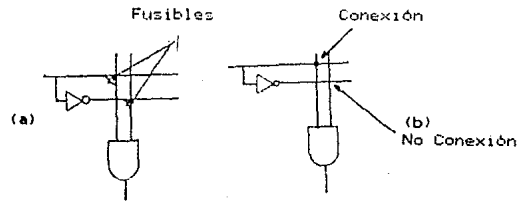


Fig. 7.49 Unidad Básica de Arreglo Lógico

Se tienen 4 posibilidades para este circuito que se muestran en la figura 7.50.

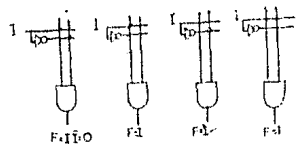


Fig. 7.50 Posibles conexiones.

Para facilitar el manejo de estos circuitos se introducirá la notación mostrada en la figura 7.51.

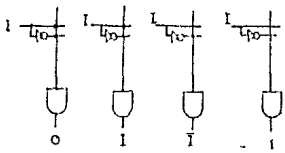


Fig. 7.51 Notación abreviada

Arreglos de Compuertas Programables (PGAs).

Estos dispositivos son arreglos de compuertas NAND como el mostrado en la figura 7.52.

El conjunto de compuertos XOR a la salida permite que ésta pueda ser manejada como verificada alta (f0 0 = f) ó verificada baja (f@ 1 = f).

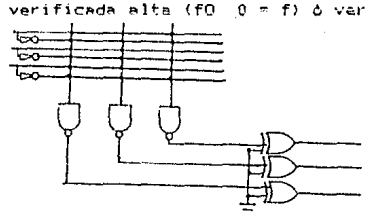


Fig. 7.52 Extructura básica de un GPAs

A continuación se verá un ejemplo de cómo implementar una función en estos dispositivos.

Supóngase que se va a implementar las siguientes funciones:

$F_0 = A\,\bar{B}\,C$	Verificada alta
$F_1 = A + B$	Verificada alta
$F_2 = ABC$	Verificada baja

El arreglo tendrá tantas salidas como funciones haya que implementar y tantas entradas como variables se tengan como se muestra en la figura 7.53.

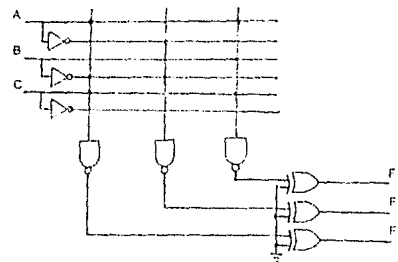


Fig. 7.53 Ejemplo de implementación con GPAs.

Quizá en este ejemplo no se vea claro la ventaja de estos arreglos pero basta con mencionar el circuito 82S101 que tiene 16 entradas, 8 salidas y 48 compuertas NAND, para tener una idea de la cantidad de variables que se pueden manejar de esta manera.

Arreglos Lógicos Programables (PLAs)

Estos dispositivos están formados por 2 arreglos, uno de compuertas AND similar a los PGAs y otro de compuertas OR como se muestra en la figura 7.54.

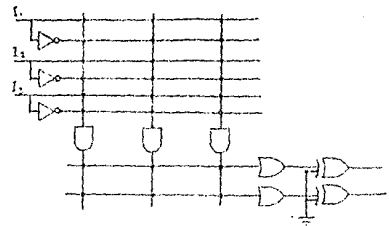


Fig. 7.54 Estructura de un PLAs.

Estos arreglos son especialmente eficientes para la implementación de funciones expresadas como la suma de miniterminos. Cabe recalcar que el que un circuito ocupe menos área dentro del PLAs no depende de la simplificación de la función, ya que todos los términos ocupan el mismo espacio sin importar cuántas variables incluyan; lo que determina el espacio utilizado es el número de términos que se presenten. Supóngase como ejemplo que se implementará un restador completo donde "X" es el minuendo, "Y", el sustraendo, "Z" el "debe" anterior, "D" la diferencia y "B" el "debe" final, todo esto descrito en la tabla de verdad de la figura 7.55.

X	Y	Z	D	B	Debe Anterior
0	0	0	0	0	
0	0	1	1	1	+ Z
0	1	0	1	1	
0	1	1	0	1	- Y
1	0	0	1	0	B D
1	0	1	0	0	
1	1	0	0	0	Debe final
1	1	1	1	1	

Fig. 7.55 Tabla de verdad de un restador.

La figura 7.56 muestra la implementación directa a partir de la tabla de verdad, mientras que en la figura 7.57 se implementan las funciones booleanas resultado de la simplificación por mapas-K.

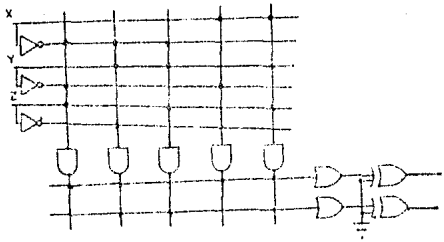
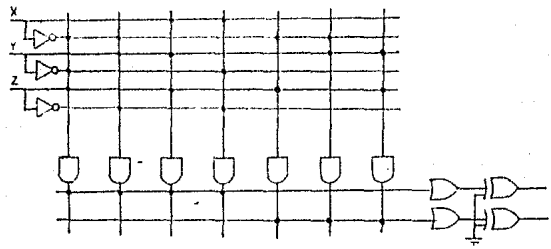


Fig. 7.56 Implementación directa de un restador.



$$D = \bar{X}\,\bar{Y}\,Z + \bar{X}\,Y\,\bar{Z} + X\,Y\,Z + X\,\bar{Y}\,\bar{Z}$$

$$B = \bar{X}\,Z + \bar{X}\,Y + Y\,Z$$

Fig.7.57 Implementación del restador minimizando las funciones.

Estos elementos al igual que los arreglos lógicos tienen versiones que sólo pueden ser programadas en el proceso de fabricación (ROM) y otras por el usuario (PROMS y EPROMS). Su capacidad varia desde 256 hasta 65536 bytes.

Constan esencialmente de 2 partes, la de dirección y la de memoria, que está formada por un arreglo de compuertas OR, mientras que la parte de direcciones está formada por un demultiplexor como se muestra en la figura 7.58.

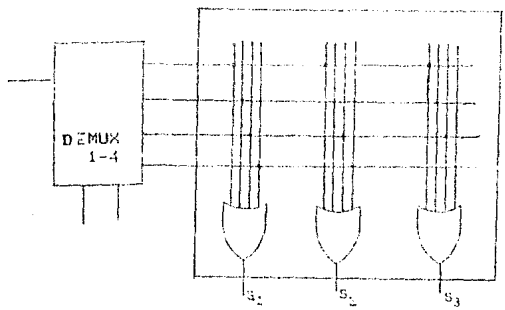


Fig. 7.58 Diagrama esquemático de una memoria ROM.

Por medio de éste se puede activar sólo una de las lineas li y si tiene conexión con alguna de las lineas li se activará alguna salida S. En la figura 7.59 se muestra un ejemplo.

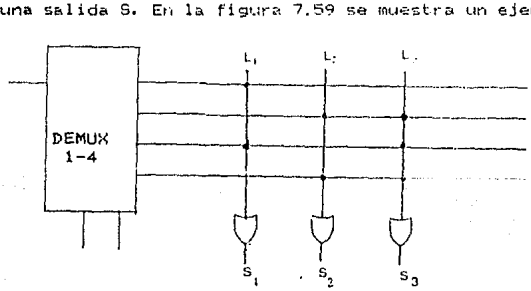


Fig. 7.59 Función implementada con una ROM.

Si por medio del demultiplexor se selecciona la línea I3., como ésta se conecta con L2 y L3 se activarán las salidas S2 y S3.

Direccion	Palabra
I 0	1 0 0
I 1	0 1 1
I 2	1 0 1
I 3	0 1 1

Fig. 7.60 Contenido de la memoria ROM.

Cada línea I es una dirección a la que le corresponde lo que se llama una palabra que es un conjunto de bits, en la figura 7.60 se muestra la palabra correspondiente a cada dirección.

Note que se tiene n estradas al multiplexor para accesar a 2n localidades de memoria.

Estas memorias proporcionan la forma más directa de implementar una función, esto es almacenando su tabla de verdad, como se muestra en el ejemplo de la figura 7.61, en donde cada renglón de la tabla de verdad es guardada en una dirección.

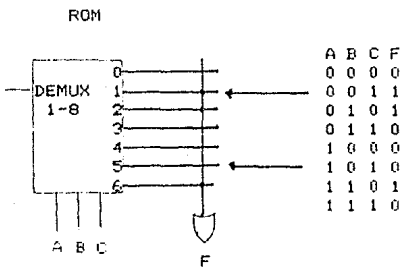


Fig. 7.61 Tabla de verdad almacenada en una ROM.

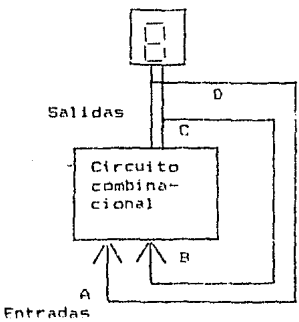
CAPITULO VIII

CIRCUITOS DECUENCIALES

LOGICA SECUENCIAL

Hasta ahora sólo se ha considerado un único tipo de circuitos digitales, denominados combinacionales, en los cuales la salida S de un circuito en un determinado momento es estrictamente una función de las entradas, i.e., depende del valor presente en éstas. Sin embargo, una gran mayoría de los sistemas digitales incluyen no sólo elementos combinacionales, sino que exigen además la presencia de algún tipo de "memoria". A los circuitos que se construyen con estos fines se les denomina secuenciales.

Por ejemplo si se supone que se quiere diseñar un circuito que realice el conteo del 0 al 3 cíclicamente, se podría pensar que basta con implementar un circuito como el que se muestra en la fig 8.1 el cual posee retroalimentación:



Entradas		Salidas	
A	B	C	D
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Fig. 8.1 Circuitos combinacional retroalimentado

Ahora se verá por què el circuito es insuficiente: cuando A=B=0, en las salidas aparecerá un 01, el cual se retroalimenta para producir una nueva salida 10, y asi secesivamente pero, ¿ cuàl de las dos salidas alcanza primero su valor final?, ¿còmo afecta esto a las entradas? Al parecer no se puede responder con certeza estas preguntas, pues se tiene un circuito cuyo comportamiento es imprevisible, ya que aún si ambas salidas se estabilizaran al mismo tiempo, no se puede controlar què tan ràpido se suceden cada uno de los estados (0,1,2,3), pues ello depende del tiempo que tardan en dar respuesta las compuertas.

Para superar estos problemas se requiere de un elemento de memoria como el que se muestra en la figura 8.2

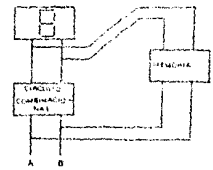
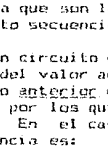


Fig. 8.2 Circuito combinacional con memoria y retroalimentación

El circuito posee la facultad de mantener un mismo estado durante el tiempo suficiente para que el circuito combinacional se estabilice y pueda dar paso al siguiente valor en el momento adecuado.

En este circuito existen tres características que es importante resaltar, ya que son las que en general definen la estructura de un circuito secuencial:

- A diferencia de un circuito combinacional, las salidas no sólo dependen del valor actual de las entradas, sino también del estado anterior del mismo, de tal manera que todos los estados por los que pasa un circuito conforman una secuencia . En el caso del ejemplo que se está tratando la secuencia es:



- La retroalimentación

- Los elementos de memoria

A manera de comentario, es importante hacer notar el cuidado que debe tenerse al definir un circuito secuencial, ya que si bien las condiciones arriba expuestas son necesarias, no resultan suficientes, y podría confundirse una máquina secuencial con una de estado finito. La primera pasa por una secuencia de estados determinados, secuencia que puede ser tan larga como sea posible, incluso infinita (como sería el caso de una computadora); una máquina de estado finito también realiza una secuencia, pero ésta debe ser cíclica y finita (como en un sistema de control).

Ahora bien, entre las características que se han mencionado se encuentran la de la necesidad de un elemento de memoria, pero, ¿qué es un elemento de memoria? Este se define como un dispositivo capaz de almacenar información durante un cierto tiempo.

Existen distintos tipos de memoria, pero todos basan su funcionamiento en algo que se conoce como celda binaria, que es simplemente un circuito con la capacidad de almacenar un bit de información.

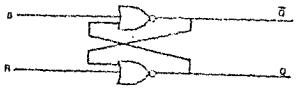


tabla electrónica de la compuerta NOR
(lógica positiva)

A	B	S
L	L	H
L	H	L
H	L	L
H	H	L

Fig. 8.3 Celda binaria

Un diseño típico para una celda de memoria es como el que se presenta en la figura 8.3 y cuyo comportamiento se describe a continuación:

1.- Supóngase que en un instante dado S es igual a H y R es igual a L, entonces, analizando el diagrama de la fig 8.3 (a) se tiene Q=H y Q=L

Fig. 8.3(a) Celda binaria con S=1 R=0

2.- Si ahora S cambia al estado bajo, la compuerta 1 tendrá en sus entradas los estados H y L respectivamente por lo que Q=L y Q=H. Así, aunque ha cambiado el estado de las entradas, las salidas han permanecido sin cambios, esto es, han almacenado un bit.

3.- Si R = H y S = L, se tendrá, a la salida de la compuerta 2, Q=L, por lo que las entradas de la compuerta 1 serán L y por ende $\bar{Q}$ = H.

Se le deja al lector verificar que si R cambia a L las salidas Q y Q no son alteradas.

4.- Si S = H y R = H entonces Q = L y Q = L, lo que se contrapone a la idea original de que Q es el complemento de Q, por lo que el estado S=R=H se considera no definido; existe otro inconveniente: si inicialmente S=R=L y posteriormente ambos cambian al estado alto, entonces el valor que tomen Q y Q dependerá de cual de las entradas cambie primero.

Considerando el análisis anterior se realiza la tabla de verdad mostrada en la fig 8.4 donde Qn se refiere a la salida Q en el instante n y Qn+1 en el instante n+1.

S	R	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	No
1	1	1	definidos

Tomando Qn como variable introducida.

S	R	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	No definido

Fig 8.4 Tabla de verdad de la celda de memoria SR.

Celda de memoria síncrona

Es usual encontrar a la celda de memoria como parte de un circuito más complejo, como se muestra en la fig 8.5.

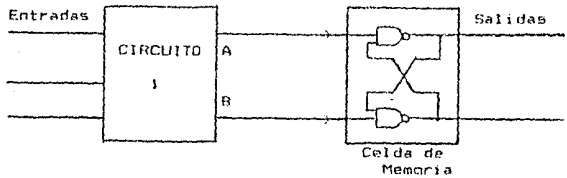


Fig 8.5. Circuito con celda de memoria.

Las salidas A y B variarán sus estados hasta estabilizarse, ocasionando que la celda de memoria actue aleatoriamente, por que lo que es necesario controlar el momento en que inicia su funcionamiento, lo cual se logra implementando una línea habilitadora tal como se muestra en la fig 8.6.

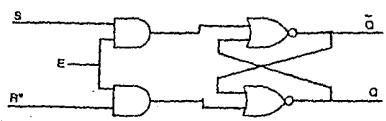
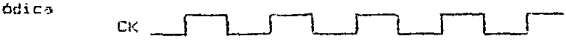


Fig 8.6. Celda de memoria con entrada habilitadora.

La entrada E se suele conectar a una señal periódica, en cuyo caso se le conoce como reloj (CK); cuando una línea de entrada depende de una señal de este tipo se le conoce como síncrona y cuando no es así como asíncrona

E	S	R	Qn	Qn+1	$\overline{Qn+1}$
L	X	X	L	L	
L	X	X	H	H	
H	L	L	L	L	
H	L	L	H	H	
H	L	H	L	L	
H	L	H	H	L	
H	H	L	L	H	
H	H	L	H	H	
H	H	H	L	No definida	
H	H	H	H	No definida	

Fig. 8.7. Tabla de verdad de la celda de memoria síncrona



Cuando E se encuentra en estado bajo, las entradas a la celda de memoria permanecen en estado bajo, esto es, sin cambio alguno; si por el contrario E está en estado alto, las entradas a la celda de memoria dependerán de S y R, según lo cual se obtiene la tabla de la fig 8.7.

Aquí la notación Qn y Qn+1 posee un significado más preciso ya que Qn designa ahora al estado de Q antes de la enésima entrada habilitadora (estado presente), y Qn+1 a la salida Q después de la enésima entrada (estado siguiente).

Al circuito que se acaba de tratar se le llama comúnmente flip-flop SR; de hecho, a los diversos circuitos que tienen una estructura similar se les denomina genéricamente flip-flop's.

Se verá ahora con mayor detalle las características del flip-flop SR: de la figura 8.7 se puede obtener un mapa K y, por consiguiente, una ecuación característica del flip-flop (suponiendo que todas las variables son verificadas altas).

Flip-Flop D.- Este tipo de flip-flop se presenta en dos modalidades que a menudo se confunden: el flip-flop D latch y el flip-flop D.

El flip-flop D latch está diseñado de manera tal que cuando la entrada habilitadora se encuentra en alto, la salida Q refleja el valor dado en la entrada D. Es sencillo implementar este flip-flop a partir de un SR; como se muestra en la fig. 8.10.

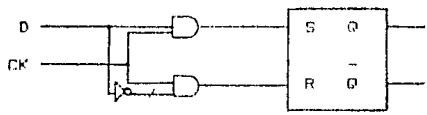
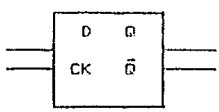
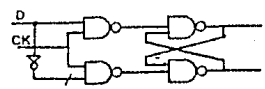


Fig. 8.10 Flip-Flop D latch.

Las características generales del flip-flop D latch se muestran en la figura 8.11.



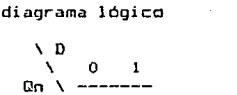
a) Símbolo lógico



b)diagrama lógico

Qn	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

c)Tabla característica



d) ecuación característica

Qn --->	Qn+1	D
0	0	0
0	1	1
1	0	0
1	1	1

e) Tabla de excitación.

Fig 8.11. Características generales del Flip-Flop D.

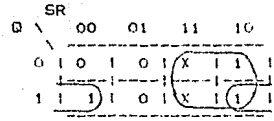


Fig. 8.8 Mapa-K del flip-flop SR

de donde

$$Qn+1 = S + \bar{R}Qn$$

y como S = R = 1 no está definido se tiene:

$$SR = 0$$

También de la fig. 8.7, denominada tabla característica (define las propiedades lógicas del circuito y "caracteriza" por completo su funcionamiento), se puede obtener una nueva representación llamada tabla de excitación mostrada en la fig. 8.9 la cual muestra cuáles son las condiciones que deben existir en las entradas del circuito a fin de lograr la transición de Qn a Qn+1 deseada; tanto la ecuación característica como la tabla de exitación del flip-flop serán elementos sumamente útiles en el diseño de circuitos secuenciales básicos, según se verá más adelante; por ahora se continuará el estudio de los distintos tipos de flip-flop's que existen.

Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Fig. 8.9 Tabla de exitación del Flip-Flop SR

En la actualidad, se puede encontrar a los flip-flop's como circuitos integrados, la mayoría de los cuales presenta dos posibilidades respecto a la entrada habilitadora E: unos se activan con la transición negativa y otros con la positiva de la señal habilitadora, a continuación se verá otros tipos de flip-flop's.

El flip-flop D latch es ampliamente usado como almacén temporal de información en diversos diseños, tales como contadores de frecuencia, voltímetros digitales, etc.

La figura 8.12 ilustra cómo se aprovecha esta idea de almacenamiento temporal.

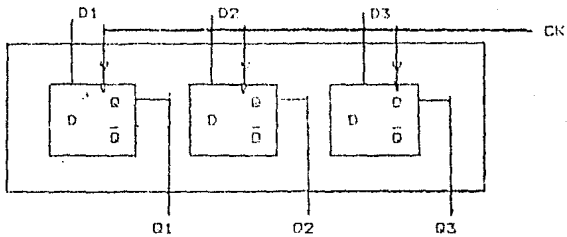


Fig. 8.12. Almacen temporal de información.

Cuando el reloj está en alto, los datos de la entrada son transferidos a la salida; cuando el reloj se va a bajo, la salida retiene los datos mientras la señal no cambie:

```
                    D       D       D    = 010
                    !1      !2      !3
reloj en alto:      !       !       !
se transfiere ---> V       V       V
la información      Q       Q       Q    = 010 <- mientras CK
                     1       2       3          esté en bajo,
                                                se almacena la
                                                información.
```

La figura 8.13 muestra importantes parámetros a considerar en un flip-flop D latch:

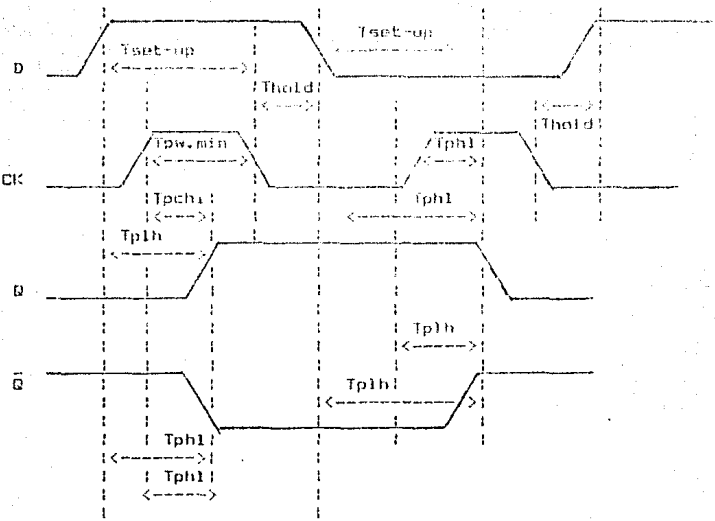


Fig. 8.13 Paràmetros temporales de un Flip-Flop D lath

Tpw,min: es el ancho mínimo que debe tener el pulso habilitador.

Tset-up, o tiempo de arranque, es el tiempo mínimo necesario durante el cual la señal de entrada debe permanecer estable, antes de la llegada del pulso activo de reloj, a fin de lograr un correcto funcionamiento del circuito.

Thold, o tiempo de relajación, es el lapso durante el cual la señal de entrada debe permanecer estable después de dado el pulso de reloj.

Tplh, o tiempo de propagación de abajo (low) a arriba (high), es el tiempo que transcurre desde que se da el pulso habilitador hasta que ocurre un cambio de estado en Q Tphl, es el tiempo de propagación arriba-abajo.

Las especificaciones de estos tiempos son proporcionados por el fabricante del circuito; si alguno de ellos no se ajusta a los márgenes permitido, se tendrá que el circuito presenta un comportamiento imprevisible.

Un diseño típico es el 7475, que consta de cuatro flip-flop's D latch, y que algunas veces es llamado "quad-bistable latch", en el que se especifica que Tset-up = 20 ns, Thold = 5 ns, Tplh = 30 ns como máximo y Tphl = 50 ns como máximo.

Ahora bien, como ya se mencionó al principio existe un circuito que a menudo es confundido con el Dlatch, y que es el flip-flop D. La diferencia entre ambos estriba en el hecho de que el flip-flop D transfiere la información presente en D sólo cuando existe una transición positiva del reloj. Esto puede verse, por ejemplo, en la tabla de funcionamiento del 7474.

Entradas				Salidas		
RESET	clear	CK	D	Q	$\bar{Q}$	
L	H	X	H	H	L	
H	L	X	X	L	H	
L	L	X	X	H	H	(NO ESTABLE)
H	H	↑	H	H	L	
H	H	↑	L	L	H	
H	H	L	X	Q	$\bar{Q}$	

Fig. 8.14 Funcionamiento del Flip-Flop D (7474)

La figura 8.15 muestra cómo las formas de onda de la salida difieren para un D latch y un D, con la misma entrada D y señal de reloj.

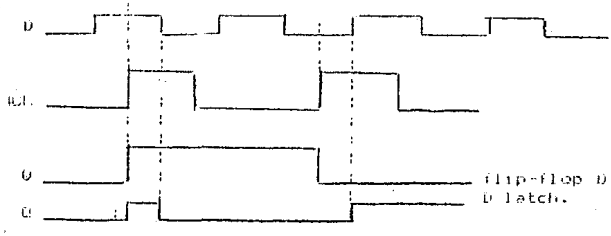
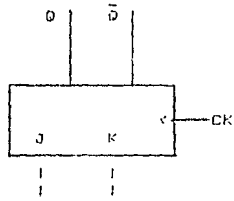


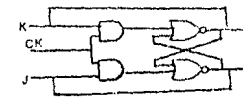
Fig. 8.15 Diagrama de tiempos comparativo de flip-flop D y latch.

Flip-Flop JK.- Este es una especie de refinamiento del Flip-Flop SR, ya que elimina el problema de los estados no definidos.

La fig. 8.16 muestra el símbolo lógico, tabla, ecuación característica, y tabla de excitación de este flip-flop.



a) Símbolo lógico.



b) Diagrama lógico.

Qn	J	K	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

b) Tabla característica

Q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$Qn+1=J\bar{Q}+\bar{K}Q$

c) Ecuación característica.

Qn	----->	Qn+1	J	K
0		0	0	X
0		1	1	X
1		1	X	1
1		1	X	0

e) Tabla de excitación.

Fig. 8.16 Características del Flip-Flop JK.

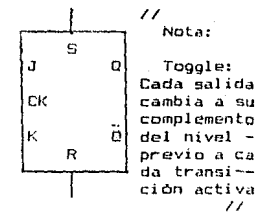
En la práctica se puede encontrar que se manufacturan tres tipos de flip-flop's JK: el que responde a transiciones positivas de reloj (positive-Edge-Triggred JK Flip-Flop), el que lo hace con transiciones negativas (Negative-Edge-Triggered JK Flip-Flop) y el que responde a un pulso, denominado maestro-esclavo (master-Slave JK flip-flop).

El siguiente es un ejemplo de "Positive-Edge-Triggred" y corresponde a la clasificación CMOS 4027:

4027

Entradas					Salidas	
S	R	CK	J	K	Q	$\bar{Q}$
L	H	X	X	X	0	1
H	L	X	X	X	1	0
H	H	X	X	X	H*	H*
L	L	↑	L	L	Qo	$\bar{Q}$o
L	L	↑	H	L	H	L
L	L	↑	L	H	L	H
L	L	↑	H	H	Toggle	

Tabla de Funcionamiento



Nota:

Toggle: Cada salida cambia a su complemento del nivel previo a cada transición activa.

Símbolo Lógico

Fig. 8.17 Flip-Flop JK "Positive-Edge-Troggred" (4027).

Las fechas ↑ indican que el cambio ocurre con la transición positiva del reloj (CK); las entradas directas S y R son idependientes del reloj y de las entradas J y K.

La flecha ↓ indica que el flip-flop responde a transiciones negativas del reloj.

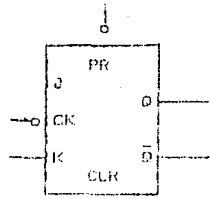
Otros ejemplos de este tipo de flip-flop son, por ejemplo, el 74103, 74113 y 74114.

La figura 8.18 corresponde al 74LS76 dual.

74LS76

Entradas					Salidas	
PR	CLR	CK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Qo	Qo
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	Toggle	
H	H	H	X	X	Qo	Qo

Tabla de funcionamiento



Simbolo lògico

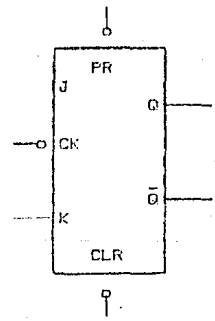
Fig. 8.18 Flip-Flop JK negative edge-Triggered (74LS76).

En la figura 8.19 se representa la tabla de funcionamiento del flip-flop 7476, maestro-esclavo:

7476, H76

Entradas					Salidas	
PR	CLR	CK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⎍⎍	L	L	Qo	Qo
H	H	⎍⎍	H	L	H	L
H	H	⎍⎍	L	H	L	H
H	H	⎍⎍	H	H	Toggle	
H	H	H	X	X	Qo	Qo

Tabla de funcionamiento



Simbolo Lògico

Fig. 8.19 Flip-Flop JK Maestro esclavo.

Como la mayoría de los diseños JK, este flip-flop consta de dos partes, un maestro y un esclavo. El circuito está diseñado para que cuando la señal de reloj esté en alto los datos (información) J y K entren al maestro, o porción de entrada; cuando la señal del reloj se va a bajo, los datos de J y K pasan al esclavo o porción de salida. El detalle consiste en que las entradas J y K no deben cambiar mientras la señal de reloj está en alto, o de lo contrario la salida será impredecible. Para evitar este problema debe usarse una señal de pulsos muy estrechos.

Flip-Flop T.- Se la denomina así a causa de su funcionamiento "Toggle", lo cual significa que cuando la entrada T está en alto, entonces existirá un cambio de estado cada vez que se presente un pulso de reloj.

Este flip-flop no existe como circuito integrado, ya que es fácilmente implementado con cualquiera de los otros tipos de flip-flop's, tal como se ve en la figura 8.20

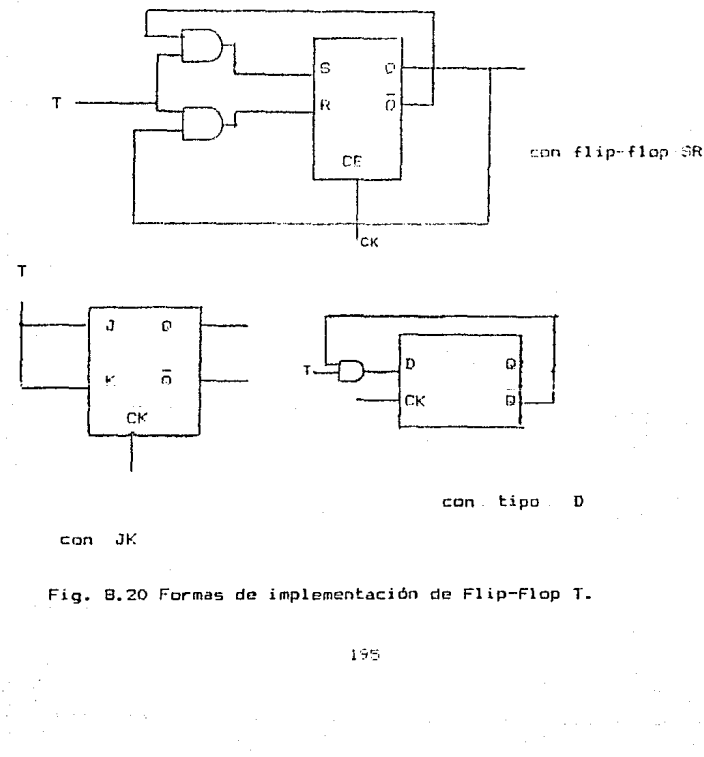


Fig. 8.20 Formas de implementación de Flip-Flop T.

Las formas de onda de la salida en cualquiera de los casos mostrados es como la que se muestra a continuación:

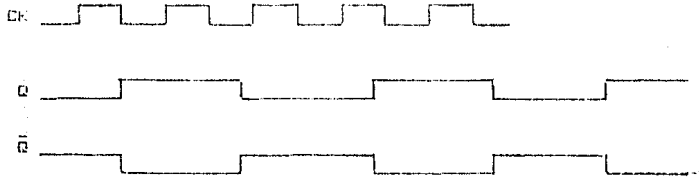
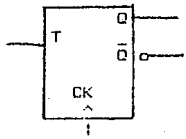


Fig. 8.21 Diagrama de tiempos de las salidas del flip-flop T.

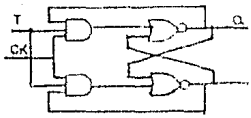
Como se puede ver, por cada dos pulsos de la señal de entrada, el flip-flop T produce un pulso como salida, esto es, la frecuencia de la señal de salida es la mitad de la frecuencia de entrada. Dos flip-flop, conectados en cascada producirán en consecuencia, una señal con una frecuencia que es un cuarto de la de entrada; añadiendo un tercero será un octavo y así sucesivamente, lo cual nos está mostrando una manera sencilla de implementar un divisor de frecuencia.

El símbolo lògico, la tabla y ecuación características, y la tabla de excitación de este flip-flop se muestrán en la figura 8.22 (a) y 8.22 (b) respectivamente.



a) Símbolo Lògico

Fig. 8.22 (a) Características del Flip-Flop T



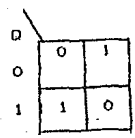
b) Diagrama lógico

Q	T	Q n+1
0	0	0
0	1	1
1	0	1
1	1	0

d) Tabla característica.

Qn ---> Qn+1	J
0 0	0
0 1	1
1 0	1
1 1	0

c)Tabla de excitación.



$Qn+1=T\bar{Q}+\bar{T}Q$

e) Ecuación característica.

Fig 8.22 (b) Cont.

REGISTROS Y RAM'S

Cuando se trató el tema de los Flip-Flop tipo D se mencionó que podría ser utilizados como un almacen temporal de información, a estos circuitos y a cualquier otro que sirva como memoria temporal se le denomina registro. Están constituidos por celdas de memoria, como la mostrada en la Fig. 8.23, en la que se puede almacenar un bit de información y posteriormente leerlo o modificarlo.

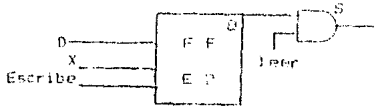


Fig 8.23 Celda tipica de memoria.

A este tipo de memoria se le conoce como de lectura-escritura.

X es la linea habilitadora de la celda, asi que si se desea acceder a ella hay que conectar esta entrada a voltaje alto, luego, si se desea escribir un bit se coloca la linea "escribe" en estado alto y se introduce la información por D. Cuando se requiere leer la información almacenada es suficiente con poner en alto la linea "leer" y el dato aparecerá en la salida S.

En este tipo de celdas los datos pueden ser almacenados indefinidamente, siempre y cuando el suministro de energia electrica sea adecuado, por lo que reciben el nombre de "memorias estáticas", por otro lado, existe otro tipo de memorias denominadas dinámicas en donde la información contenida en ellas en forma de carga eléctrica, decae a lo largo del tiempo (milisegundos), por lo que es necesario un circuito que restablezca constantemente la información, conocido como circuito de "refrescamiento". Este tipo de memoria ocupa menos espacio, es más veloz , disipa menos potencia que una memoria estática, y posee mayor capacidad de almacenamiento, pero requiere el circuito de refrescamiento junto con las señales extras que esto implica.

Un conjunto de Celdas de memoria forman un registro, como se muestra en la figura 8.24

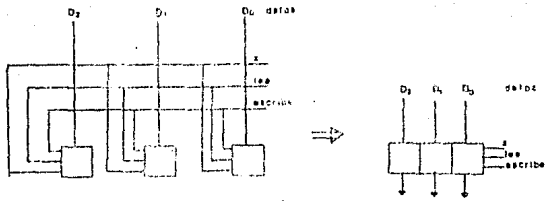


Fig. 8.24 Registro con celdas de memoria estàtica

Con el reciente desarrollo tecnológico, se han creado circuitos integrados capaces de almacenar temporalmente grandes cantidades de información a los que se les denomina Memoria de Acceso Aleatorio (Random Access Memori y RAM) que son bàsicamente arreglos de registros, en donde la información se puede accesar por medio de una dirección de igual manera que en los ROM; Un ejemplo de esto es el chip tms4161 MOS de Texas Instrument capaz de almacenar 65 536 bits o el vt23512 con una capacidad de 65 536 bytes.

Las memorias pueden estar organizadas por bit o por palabra, un ejemplo de èsta última es una pequeña memoria que almacena 16 palabras cada una de 4 bits, que consta esencialmente de 3 bloques: el arreglo de celdas de memoria, el módulo de direccionamiento (por medio del cual se selecciona una de las 16 palabras) y el módulo de salidas, como se muestra en la figura 8.25. Por medio de las lineas A0,A1,A2,A3 se selecciona la dirección deseada; para almacenar información en ella se pone en estado alto la linea de lectura/escritura (R/W) y se introduce la información por las lineas D0,D1,D2 y D3, si se desea leer esta información se pone en estado bajo la linea de lectura/escritura y se recibe la información por las lineas D0,D1,D2 y D3.

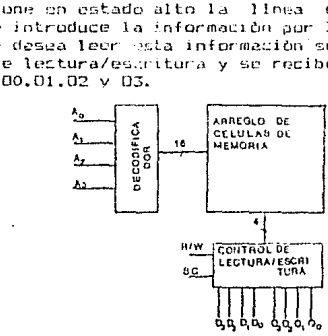


Fig 8.25 RAM de 16 palabras X 4 bits

Una memoria con la misma capacidad pero organizada en bits requiere 2 líneas de dirección más,esto es se crea una matriz en donde por medio de 4 líneas se selecciona el renglón y con otras 2 se selecciona la columna, como se muestra en la figura 8.26.

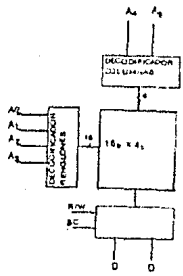


Fig. 8.26 RAM de 16 X 4 bits

Frecuentemente se requiere crear memorias de mayor capacidad a partir de memorias pequeñas, la figura 8.27 muestra como con 2 memorias de 16 palabras X 4 bits se puede crear una memoria de 16 palabras X 8 bits.

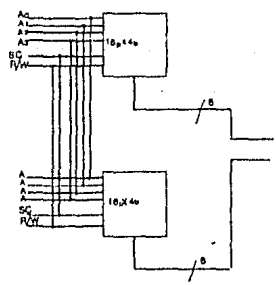


Fig. 8.27 expansion de memoria

El tema de memorias de lectura-escritura es muy rico pero excede el propósito de esta exposición por lo que no se ha tratado con mayor detalle.

CAPITULO IX

DISEÑO DE CIRCUITOS DIGITALES

DISEÑO DE CIRCUITOS SECUENCIALES.

En el año de 1972 Cristopher Clare introduce una metodología para el diseño de circuitos, cuya característica es la versatilidad respecto a los elementos físicos utilizados, respondiendo así a la necesidad de adaptarse al rápido desarrollo técnico.

Esta metodología se basa en la hipótesis de que cualquier circuito secuencial puede ser representado por un modelo general que consta de varios módulos que realizan operaciones básicas. Dicha metodología consta de las siguientes etapas:

1.- Definición. En donde se determinan los módulos que formarán el diseño.

2.- Descripción. Aquí se detalla la operación lógica de cada módulo.

3.- Síntesis. En donde se realiza la implementación física.

Para mostrar cómo se realiza cada uno de estos pasos, y buscando la claridad en el tratamiento, se explicará el tema a través de un ejemplo práctico.

Supóngase que se desea diseñar un instrumento contador de ciertos eventos tal que al ocurrir el cuarto de éstos se active una señal de salida y regrese a su estado inicial; la ocurrencia de cada evento se manifiesta por medio de la activación de una señal de entrada. Así, cuando ocurra el evento, el contador pasará al siguiente estado, y por el contrario, si esta entrada no es activada, el contador permanecerá en el mismo estado.

Se debe diseñar entonces un contador de 0 a 3, tal que al llegar a 3 se active una salida "Y" (para dar la señal de salida), además de tener una entrada "X" que incremente la cuenta al estar activada, es decir un contador condicional. La tabla de la figura 9.1 muestra el comportamiento de este circuito

En la tabla de la fig.9.1 según se puede observar, el llamado Estado Presente (cuya asignación binaria está dada por An,Bn) designa la situación de los flip-flop's antes de la

ocurrencia de un pulso de reloj, en tanto que el llamado Estado Siguiente (An+1,Bn+1) señala el estado posterior a dicho pulso.

	Nombre del Estado	Estado Presente		Entrada	Estado Siguiente		Salidas
		An	Bn	X	An+1	Bn+1	Y
X = 0 no cuenta	0	0	0	0	0	0	0
	1	0	1	0	0	1	0
	2	1	0	0	1	0	0
	3	1	1	0	1	1	0
X = 1 contando	0	0	0	1	0	1	0
	1	0	1	1	1	0	0
	2	1	0	1	1	1	0
	3	1	1	1	0	0	1

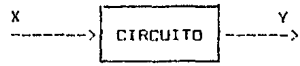


Fig. 9.1 Tabla de verdad y Esquema de bloques de un contador 0-3.

DEFINICION

En la etapa de definición se considera que el circuito puede ser representado por el modelo general de la maquina de estados que consiste en 2 módulos de transformación (circuitos combinacionales) y 1 módulo de memoria (flip-flop's, registros, RAM's, etc.) según se muestra en la figura 9.2.

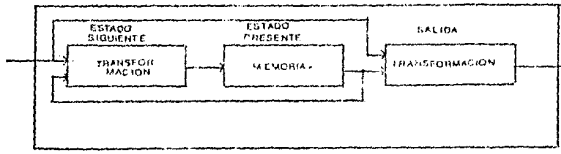


Fig.9.2 Modelo General de la Máquina de Estados

En la tabla de la figura 9.1 se observa que An+1 es función de An,Bn y X al igual que Bn+1 y la salida Yn+1

$$An+1=An+1(An,Bn,X)$$

$$Bn+1=Bn+1(An,Bn,X)$$

$$Y=Y(An,Bn,X)$$

a las variables An+1 y Bn+1 se les conoce como las funciones del estado siguiente y son implementadas mediante un módulo de transformación y otro de memoria (compárelas con las ecuaciones características del capitulo anterior).

La función de salida Y se implementa mediante un módulo de transformación, como un circuito combinacional común, así que en este caso el circuito queda representado como se muestra en al figura 9.3, con lo que se completa la primera etapa.

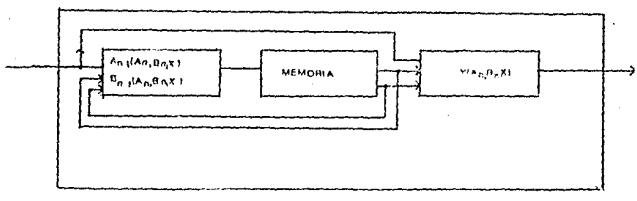


Fig 9.3. Máquina de Estados del contador condicional

DESCRIPCION

En la etapa de Descripción se detalla la operación del circuito para lo que existen 6 herramientas: tablas de verdad, funciones booleanas, mapas-K, mapa de variable introducida y dos formas más que se verán a continuación, llamadas diagramas de estados y diagramas ASM (Algorithmic State Machine).

Diagrama de estados

Para desentrañar el funcionamiento de un circuito se puede acudir al diagrama de estados del circuito, el cual viene a ser una representación gráfica del funcionamiento de la máquina y que contiene la misma información que la tabla de estados.

El diagrama de estados de la figura 9.4 correspondiente a la tabla de la figura 9.2 del ejemplo que se está tratando.

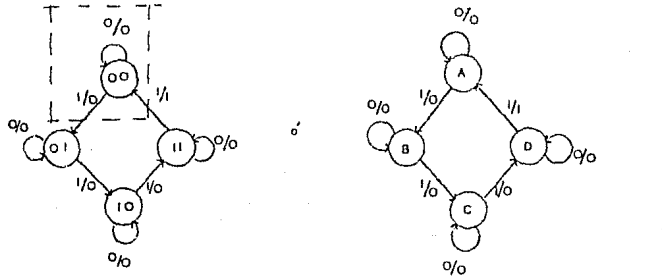


Fig. 9.4 Diagrama de estados del contador condicional.

Se puede ver que cada uno de los posibles estados del circuito se representa mediante un círculo, dentro del cual se halla inscrita la literal o cifra binaria a que corresponde. La transición de uno a otro estado se representa mediante una flecha conectora, que, sobre el mismo círculo indica que no hay cambio de estado, y de un círculo a otro, indica que sí existe transición. Junto a cada flecha se coloca una línea diagonal, en donde se especifica el valor de las entradas y salidas (entrada/salida).

Se analizará con más detalle lo expuesto, para lo cual se tiene solamente la parte encerrada en el cuadro de la fig. 9.4.

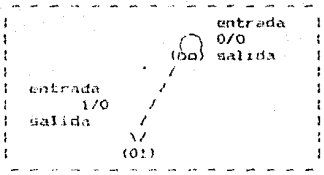


Fig. 9.5 Ampliación de la Fig. 9.4.

Cuando el estado presente es 00 y la entrada externa X es 0, lo cual se indica como 0/ , se ha establecido que el estado siguiente es el mismo, y la salida Y=0 (0/0). Si se tiene, en cambio, que X=1, se efectuará una transición al estado 01 y Y sigue siendo 0 (1/0).

Diagramas ASM

Usualmente, se presenta la necesidad de realizar diseños con varias variables, en donde los diagramas de estado resultan ineficientes, es aquí cuando los diagramas ASM muestran su utilidad.

Antes de comenzar propiamente con los diagramas ASM se verá una convención muy generalizada para denominar las líneas de entradas y salidas:

- A cada línea se le asigna un nombre de 3 ó 4 letras que haga alusión a la función que realiza, llamada mnenónico.

- Cuando las líneas son de salida se anteceden de la letra H si es verificada alta, y de L si es verificada baja.

- Cuando las líneas de entrada son verificadas altas se les antecede de la literal Y, y si son verificadas bajas de una N.

Siguiendo esta convención se le dará el nombre de YENT a la entrada X y HSAL a la salida Y.

Los diagramas ASM están formados por tres símbolos básicos que se muestran a continuación:

1) Caja de Estados. Como su nombre lo indica se utilizan para representar cada uno de los estados por los que

pasa la máquina su estructura es la representada en la figura 9.6.

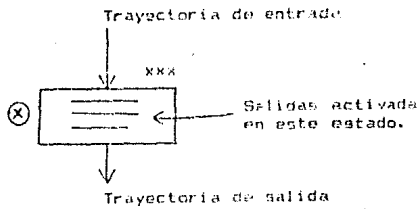


Fig. 9.6 Caja de Estados

2) Caja de Decisiones. En éstas se pregunta por la condición de falso o verdadera de una entrada dada, y dependiendo de ello se elige una de 2 trayectorias, su estructura se presenta en la figura 9.7.

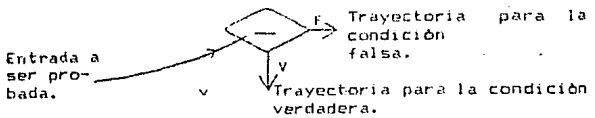


Fig. 9.7 Caja de Decisiones.

3) Caja de Salidas Condicionales. Estas siempre se colocan a continuación de una caja de decisiones, en su interior se expresan las salidas que se activarán para la condición dada, su representación se muestra en la fig. 9.8.

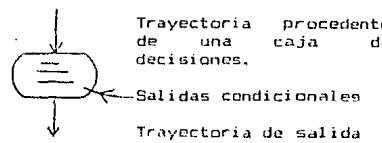


Fig. 9.8 Caja de salidas condicionales.

Como ejemplo se muestra en la figura 9.9 el diagrama ASM del instrumento descrito al principio del capítulo.

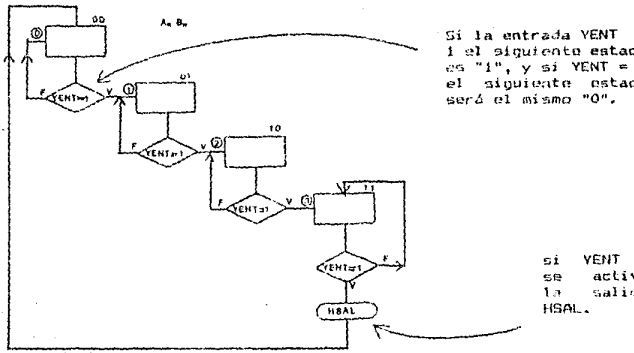


Fig. 9.9 Diagrama ASM.

Ahora se introducirá la idea de bloque en un diagrama ASM. Un bloque se indica por una caja de estados; si una de éstas está seguida de cajas de decisiones o salidas condicionales, todas forman parte de un bloque como se indica en la figura 9.10.

A las trayectorias que salen de un bloque se les conoce como trayectorias de ligamiento (link path).

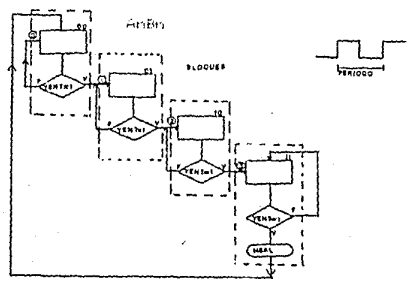


Fig. 9.10 Bloques en un diagrama ASM.

Cada caja de estados esté asociada a un ciclo de reloj, así que todas las acciones indicadas en un bloque se realizan en un período o ciclo de reloj. Por ejemplo, en el estado 3 se prueba la entrada YENT y se activa HOLD en el mismo ciclo de reloj.

Por último, se mencionarán algunas precauciones que se deben tomar al realizar un diagrama ASM.

1) Se debe estar seguro de que cada estado lleva a un único estado siguiente para las condiciones de entrada dadas, ya que la máquina no puede estar en dos estados distintos al mismo tiempo, un ejemplo se muestra en la figura 9.11.

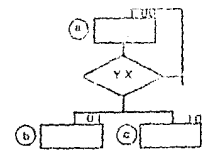


Fig. 9.11 Diagrama erróneo.

2) Los loop's siempre deben contener al menos una caja de estados, en la figura 9.12.a ésto no se cumple, mientras que en la 9.12.b se muestra la configuración correcta.

a) b)

Fig. 9.12 Configuraciones, a)inválida b)válida

3) Finalmente, se debe cuidar que los arreglos de cajas de decisiones no produzcan estados a los que sea imposible llegar, como se muestra en la fig. 9.13, ya que para llegar al estado "b" se debería seguir la trayectoria indicada con una línea punteada en dicha figura, para lo cual sería necesario que YX fuese falsa y verdadera simultaneamente.

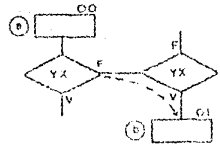
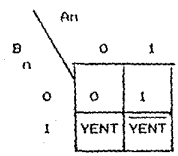


Fig. 9.13 Diagrama donde el estado b nunca podrá alcanzarse

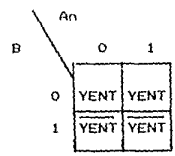
Estos errores no siempre son fáciles de reconocer, por lo que se deberá tener especial cuidado en el diseño.

Funciones Booleanas y Mapas de Variable Introducida

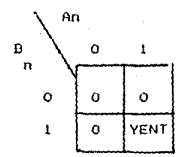
La función booleana que describe el circuito se puede obtener a partir de la tabla de verdad de la figura 9.1, simplificando por medio de un mapa de variable introducida para cada una de las variables de estado An+1, Bn+1 y la salida HSAL, como se muestra en la figura 9.14.



$An+1 = An\overline{Bn} + YENT\ \overline{An}Bn + An\overline{YENT}$



$Bn+1 = \overline{YENT}Bn + YENT\ \overline{Bn}$

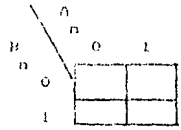


YENT = AnBn YENT

Fig. 9.14 Mapas VEM y Funciones Booleanas.

Estos mapas de variable introducida puede ser obtenidos, tambien a partir del diagrama ASM de la fig. 7.9, como se expondrá a continuación.

El primer paso es dibujar el mapa cuyas variables son siempre las variables de estado:



Para llenar la celda correspondiente a An = Bn = 0 se observa la caja correspondiente al estado "0" del diagrama ASM y se busca la trayectoria que lleve a un estado siguiente en el que An sea igual a 1, como en este caso no hay tal, se coloca un 0 en esta celda.

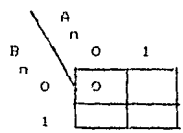


Fig. 9.15. Minitérmino correspondiente al estado "0".

A continuación se llenará la celda correspondiente a An=0, Bn = 1, para lo que se observa el estado respectivo en el diagrama ASM (1), en este caso cuando YENT = 1 la maquina pasa a un estado donde An = 1, esto es,

Nombre de estado	An	YENT	Nombre de estado siguiente	An + 1
1	0	0	1	0
1	0	1	2	1

===> An+1 = YENT

así que se escribe la variable introducida YENT en la celda correspondiente.

Cuando An = 1 y Bn = 0 se tiene que cualquiera de las dos posibles trayectorias llevan a la máquina a un estado donde An = 1, así que se coloca un "1" en esta celda.

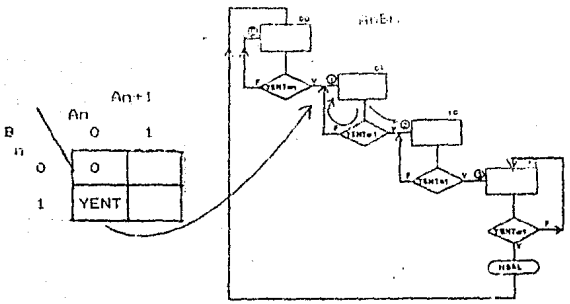


Fig. 9.16 Minitémino correspondiente al estado "1".

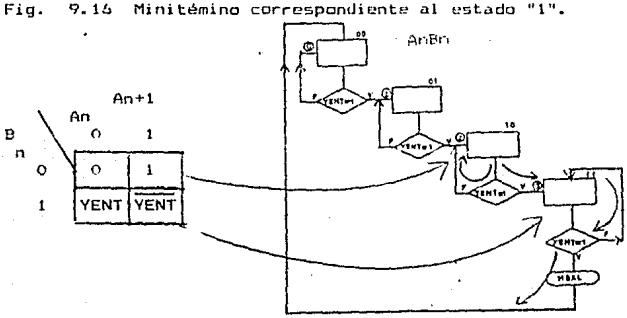


Fig. 9.17 Minitermino correspondiente al estado "2" y "3".

Finalmente a la celda $\overline{An} = \overline{Bn} = 1$ le corresponde la variable introducida $\overline{YENT}$

De manera análoga se obtiene el mapa para $\overline{Bn+1}$.

Para obtener la función de salida HSAL se realiza un proceso semejante, se verifica cuales salidas se activan en cada estado y la celda correspondiente se llena con el valor adecuado de igual manera que en el caso anterior. En la figura 9.18 se muestra el mapa de variable introducida y la booleana para la función de salida.

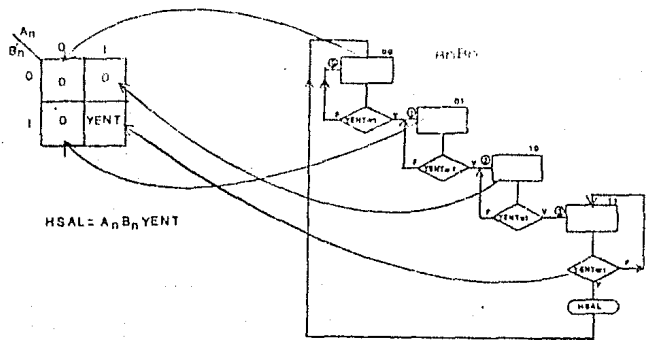


Fig. 9.18 Deducción de la función de salida HSAL.

SINTESIS

En esta etapa se eligen los dispositivos físicos con los que se implementará el diseño, para lo cual las opciones son ilimitadas ya que cada vez surgen nuevos dispositivos más versátiles y eficientes, en esta sección se verá sólo algunas opciones.

Siguiendo con el ejemplo del capítulo, se tienen las ecuaciones del estado siguiente y de salida

$$An+1 = An\,\overline{YENT} + An\overline{Bn} + \overline{An}Bn\,YENT$$
$$Bn+1 = YENT\,Bn + YENT\,\overline{Bn}$$
$$YENT = AnBn\,YENT$$

A continuación se implementarán An + 1 y Bn + 1, por medio de un módulo de memoria y otro de transformación. En el capítulo anterior se vieron cuatro tipos de flip-flop's que se muestran en la figura 9.19 junto con sus ecuaciones características.

Flip-Flop	Ec. característica
T	$Qn+1 = \overline{T}\,Qn + T\,\overline{Qn}$
D	$Qn+1 = D$
RS	$Qn+1 = S + Qn\,\overline{R}$
JK	$Qn+1 = J\,\overline{Qn} + \overline{K}\,Qn$

Fig. 9.19 Ecuaciones características de los F-F.

Cada variable del estado siguiente se implementa por medio de un Flip-Flop, si se elige el tipo D se debe hacer la análogia entre la ecuación de estado de la variable elegida y la ecuación característica de F-F como se muestra a continuación:

$$An+1 = An\,\overline{YENT} + An\overline{Bn} + \overline{An}Bn$$

$$Qn+1 = D$$

entonces

$$D = An\,\overline{YENT} + An\overline{Bn} + \overline{An}Bn$$

Si se elige un flip-flop JK se tiene:

$$An+1 = An\ \overline{YENT} + An\overline{Bn} + \overline{An}Bn\ YENT$$

$$Qn+1 = Qn\overline{K} + \overline{Qn}J$$

entonces

$$K = \overline{YENT} + \overline{Bn}\ YENT\ Bn$$
$$J = BnYENT$$

Con un flip-flop RS se tiene:

$$An+1 = An\ \overline{YENT} + An\overline{Bn} + \overline{An}Bn\ YENT$$

$$Qn+1 = Qn\overline{R} + S$$

entonces

$$S = AnBn\ YENT$$

$$R = \overline{(\overline{YENT} + \overline{Bn})} = YENT\ Bn$$

Si se elige un flip-flop tipo T se tiene:

$$An+1 = An\ \overline{YENT} + An\overline{Bn} + \overline{An}Bn\ YENT$$

$$Qn+1 = Qn\overline{T} + \overline{Qn}T$$

como

$$(\overline{YENT} + \overline{Bn}) = \overline{Bn\ YENT}$$

entonces

$$T = Bn\ YENT$$

De manera análoga se obtinen las ecuaciones siguientes:

Flip-flop	En+1
T	$T = YENT$
D	$D = \overline{Bn}\ YENT + Bn\ \overline{YENT}$
SR	$S = \overline{Bn}\ YENT \quad R = \overline{YENT}$
JK	$J = YENT \quad K = YENT$

Las implementaciones para cada tipo de flip-flop se muestra en la fig 9.20.

Obsérvese que se utiliza un flip-flop para cada variable de estado

En la fig. 9.21 se muestra la implementación completa con flip-flop T, esto es, junto con la función de salida.

Se puede también utilizar multiplexores para las funciones del estado siguiente y la salida. Como se muestra en la figura 9.22.

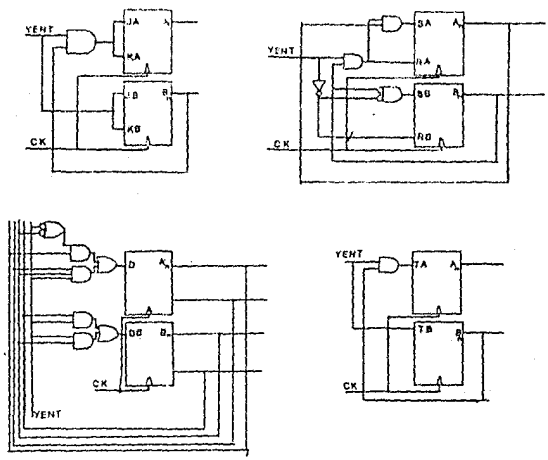


Fig. 9.20 Implementación del contador condicional con F-F tipo: a) T, b) D, c) JK y d) RS.

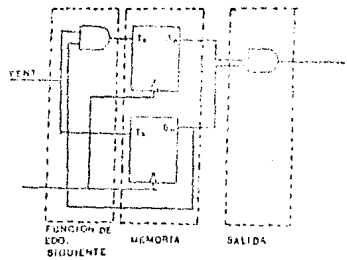


Fig. 9.21 Implementación completa del contador condicional.

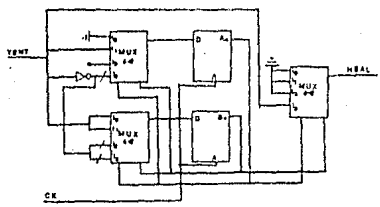


Fig. 9.22 Implementación del contador condicional por medio de multiplexores.

El reciente desarrollo de la tecnología de circuitos integrados ha permitido el uso de arreglos de compuertas para implementar funciones, cuestión que en otro tiempo se consideraba un derroche de circuitos.

Los arreglos que primeramente se usaron en la implementación ASM fueron las memorias ROM y su versión programable PROM. Estos dispositivos dan la oportunidad de implementar un circuito combinacional de la forma más directa posible: esto es almacenado la tabla de verdad de la función.

El uso de estos arreglos facilitan enormemente el diseño de circuitos digitales sobre todo cuando se manejan muchas variables, pero se debe ser cuidadoso porque esta no siempre es la mejor opción.

Se verá ahora cómo se pueden utilizar estos dispositivos en la implementación de un ASM, por medio del ejemplo antes tratado.

Se requiere un circuito ROM que posea 3 entradas y 3 salidas, 2 de las entradas se utilizarán para las variables de estado presente An y Bn y la tercera para la entrada YENT; por medio de estas 3 líneas se tiene acceso a 2 = 8 localidades de memoria, en cada una de las cuales se introducirá un renglón de la tabla de verdad de la figura 9.1. Dos de las salidas del circuito ROM serán utilizadas para las variables de estado siguiente An+1, Bn+1, y la restante se utiliza para la salida HSAL como se muestra en la Fig. 9.23 a.

La conexión entre el estado presente y el siguiente se realizará con flip-flop's tipo D, como se muestra en la figura 9.23 b.

Asi por ejemplo, supóngase que YENT = 0 y las variables del estado presente son A = 0 y B = 1.

Dirección			Contenido		
			PARTE DE LIGAMIENTO		PARTE DE INSTRUCCION
ENTRADA	ESTADO PRESENTE		ESTADO SIGUIENTE		SALIDA
YE	An	Bn	An+1	Bn+1	HSAL
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	0	1

Fig. 9.23 (a) Tabla de la memoria ROM condicional.

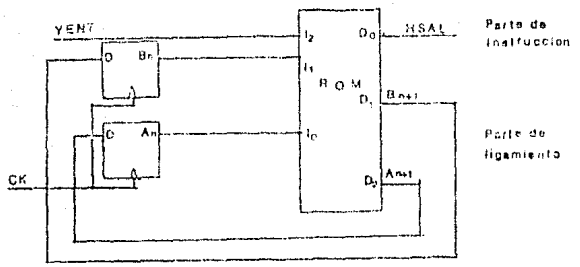


Fig 9.23 (b) Implementación del contador condicional

Al colocar estos valores en las entradas se está direccionando la localidad 01, la cual tiene el contenido 10, cada uno de estos bits aparece en una de las salidas D0 , D1 y D2. Los 2 bits mas significativos que son D1 y D2, que corresponden a An+1 y Bn+1, están conectados a los flip-flop's, por lo que en la siguiente transición de reloj se convertirán en el estado presente 10.

A la porción de la palabra de ROM que almacena las variables del estado siguiente se le llama "parte le ligamiento" (Link Part) y a la que corresponde a las salidas "parte de instrucciones" (instruction part).

A continuación se tratarán dos temas más que no han sido incluidos antes por no perder continuidad en el tema, pero debido a su importancia es indispensable su estudio.

IMPLEMENTACION A PARTIR DE LA TABLA DE EXCITACION

En ocasiones al realizar un diseño se conoce a priori el tipo de flip-flop que será utilizado y entonces el método anterior podría conducir a manipulaciones algebraicas engorrosas, por lo que es preferible diseñar a partir de la tabla de excitación del flip-flop de que se trate.

Se tomará nuevamente el ejemplo correspondiente a la tabla de la figura 9.1, suponiendo que se desea usar flip-flop's J-K para la implementación. Recuérdese la tabla de excitación de éste:

Qn --->	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Con esta información en mente, se elaborará una nueva tabla, similar a la tabla de la fig. 9.1, que se denomina tabla de excitación del circuito, mostrada en la fig. 9.24; para ello, lo que se hace es sustituir las asignaciones, esto es, si por ejemplo, el circuito se halla en el estado An=0 y el estado siguiente es también An+1=0, buscando en la tabla de excitación del flip-flop cuáles son las condiciones que deben existir para que dicha transición se realice, esto es, el valor de J y K, con lo que se encuentra que J debe ser 0 y K=X (no importa); esto se encuentra especificado en el primer reglón de la tabla de la fig.9.24; se puede observar que, como se están usando flip-flop's J-K, se necesitan dos columnas, una para J y otra para K, en cada variable involucrada, denominándose entonces como JA, JB, KA y KB a las columnas en cuestión. El resto de la tabla es igual al de la tabla original. (Fig. 9.1).

Ahora, se elaborarán los mapas K para cada variable y a partir de ellos se obtendrán las ecuaciones características mostradas en la Fig.9.25 que proporcionan, esta vez de manera directa, la implementación del circuito.

ESTADO PRESENTE		ENTRADA	ESTADO SIGUIENTE		ENTRADAS DE FLIP FLOP				SALIDA
An	Bn	X	A N+1	B N+1	JA	KA	JB	KB	Y
0	0	0	0	0	0	X	0	X	0
0	1	0	0	1	0	X	X	0	0
1	0	0	1	0	X	0	0	X	0
1	1	0	1	1	X	0	X	0	0
0	0	1	0	1	0	X	1	X	0
0	1	1	1	0	1	X	X	1	0
1	0	1	1	1	X	0	1	X	0
1	1	1	0	0	X	1	X	1	1

Fig. 9.24. Tabla de exitaciòn del diagrama de la Fig 2.

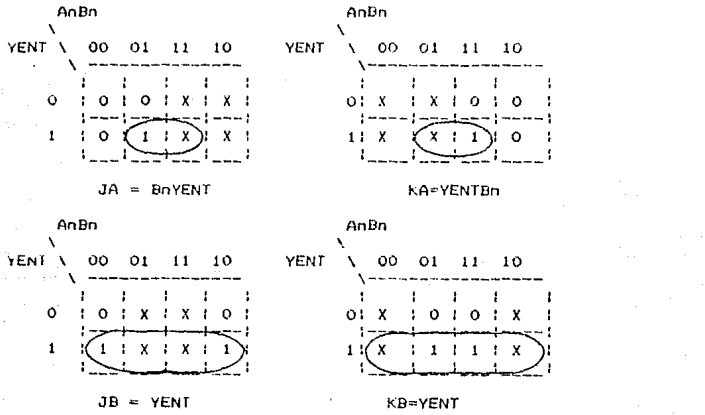


Fig. 9.25 Mapas-K e implementaciòn a partir de la tabla de exitaciòn.

Como puede verse, el resultado es el mismo que el anteriormente obtenido, expuestos en la Fig. 9.20

Reducción de Estados.

Aunque de manera general se han mostrado ya los pasos a seguir en la elaboración del diseño de un circuito secuencial, nos falta aún mencionar uno de los elementos básicos: uno de los tantos métodos para realizar la reducción de estados.

El método en cuestión busca minimizar el número de estados de un circuito, para así minimizar a su vez el número de elementos (flip-flops) usados en el diseño.

Como, ya es costumbre, se aplicará el método a partir de un ejemplo.

Supóngase entonces que se tiene necesidad de implementar un circuito que ha de pasar por ocho posibles estados; se cuenta con una entrada externa X mediante la cual se controla la secuencia en que se sucede dichos estados, y una salida Y que puede servir de indicador de algún estado en especial. El circuito puede ser, por ejemplo, un controlador de temperatura de un recinto cerrado, para el cual los ocho posibles estados pueden ser diferentes combinaciones de luz, ventilación, humedad, etc., que han de activarse en diferentes momentos; X puede servirnos para controlar directamente la temperatura del recinto, eligiendo, por ejemplo, que cuando X=1 la temperatura aumente por encima de los 0 C, y por el contrario, que cuando X=0, la temperatura descienda a menos de 0 C. Por supuesto que las acciones que se realicen (activar luz, ventilación, etc.) pueden ser escogidas según el caso, y por el momento no será de particular interés, ya que lo importante ahora es aprender a hacer la reducción de estados, proceso para el cual no es necesario conocer con precisión qué operaciones realiza el circuito, sino sólo por cuántos estados pasa, la secuencia que controla la sucesión de dichos estados (los valores de la variable externa X) y los valores de la variable de salida Y para cada estado.

Así entonces, se realizará la reducción de estados del circuito cuya tabla y diagrama de estados son los mostrados en la Fig. 9.26.

En el diagrama de la Fig. 9.26.b cada estado ha sido denotado mediante la literal minúscula correspondiente (a,b,c,...). Obsérvese que se pueden elegir numerosas secuencias de entrada al circuito de la variable externa X, tal como la 01010110100; con cada entrada X=0 ó 1, se obtiene una salida Y=0 ó 1, y un correspondiente cambio de estado. Tomando esto en cuenta podemos elaborar una tabla en la que se presenten el estado, la entrada y la salida; la primera columna tendrá a su inicio el estado presente inicial (en nuestro caso a) y, debajo, la entrada y la salida que marcan o indican el estado siguiente, el cual pasará a encabezar la siguiente columna. El resto de la tabla se trabaja de igual manera, y para aclarar cómo es esto, se muestra en la figura 9.27 la tabla de Entradas/Salidas correspondiente al diagrama

de estados ya mencionados.

ESTADO PRESENTE	ESTADO SIGUIENTE		SALIDA (Y)	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	h	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1
h	a	d	0	0

a)

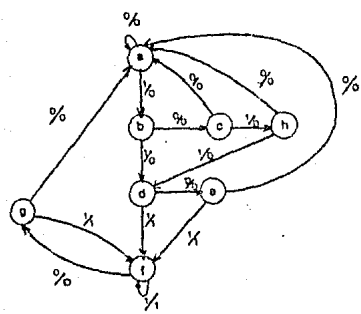


Fig. 26. a) tabla b) diagrama de estados

Estados : a a b c h d f g f g a

Entradas: 0 1 0 1 1 1 0 1 0 0 0 <- secuencia elegida

Salida : 0 0 0 0 0 1 0 1 0 0 0

Fig. 9.27. Secuencia de la màquina de Estados.

El problema ahora es encontrar un circuito equivalente que, sin alterar las relaciones de entrada/salida, presente un número menor de estados, para lo cual comenzaremos trabajando directamente con la tabla de estados (figura 9.28). La reducción de esta tabla se realiza según el siguiente enunciado: se dice que dos estados son equivalente si, por cada miembro del conjunto de entradas, proporcionan exactamente la misma salida, enviando al circuito al mismo estado (o su equivalente). En el caso de que dos estados sean equivalentes es posible suprimir uno de ellos sin que se afecten las relaciones entradas/salidas de las variables involucradas.

Se verá con más detalle cómo se realiza esto. De la tabla de la Fig. 9.28 se deduce que los estados e y g son equivalentes, ya que sus estados siguientes y salidas son las mismas; en este caso, el estado g -aunque pudiera ser el e- y en todo lugar se borra de la tabla donde aparezca se substituye por e (ver fig. 9.28).

ESTADO PRESENTE	ESTADO SIGUIENTE		SALIDA	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	h	0	0
d	e	f	0	1
e	a	f	0	1
f	g e	f	0	1
Estado Suprimido>				
h	a	d	0	0

Fig. 9.28 Reducción de Estado, 1ª Fase

Volvemos a reducir la tabla que nos queda según el procedimiento explicada; en esta ocasión los estados que son iguales son el d y el f (ver fig. 9.29).

ESTADO PRESENTE	ESTADO SIGUIENTE		SALIDA	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	b	0	0
d	e	f d	0	1
e	a	f d	0	1
h	a	d	0	0

Fig. 9.29. Reducción de Estado, 2ª Fase

de donde se obtiene el diagrama de estados de la Fig.30.

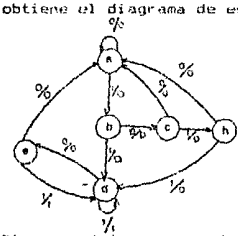


Fig. 9.30 Diagrama de estados reducido

Así entonces, se ha obtenido ya la tabla reducida de la cual se puede a su vez obtener una nueva tabla de entradas/salidas, en la que se verá que la secuencia de entradas que se había especificado sigue siendo la misma, lo cual además sirve como comprobación de que la reducción de estados se ha hecho correctamente ya que al ser comparada con la fig 9.27 resulta ser igual, considerando que d=f y e=g.

Estados :	a	a	b	c	h	a	d	e	d	e	a	
Entradas:	0	1	0	1	1	1	0	1	0	0	0	<-secuencia idèntica a la de la fig. 9.27
Salida :	0	0	0	0	0	1	0	1	0	0	0	

Fig 9.31 Secuencia de la màquina de Estados

CAPITULO X

APLICACIONES

CONTROLADOR DE SEMAFOROS

Se desea colocar un semáforo en el cruce de una avenida ancha con mucha circulación y una calle con circulación ocacional, como se muestra en la figura 10.1, se coloca un sensor en la calle angosta que se activará cuando haya algún automovil esperando.

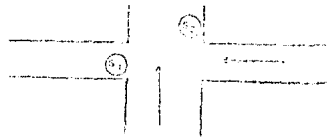


Fig 10.1 Ubicación de los semáforos

Hay un semáforo para cada calle, " S2 " para la avenida ancha y "S1" para la angosta. Si el sensor está activado, S2 permanecerá en luz verde por un máximo de 30 segundos, posteriormente pasará al ambar 10 seg., y finalmente al rojo 30 seg., lo que implica que S1 estrará 50 seg. en rojo, 20 seg. en verde y 10 en ambar. Mientras el sensor no esté activado S2 permanecerá en verde y S1 en rojo.

En la figura 10.2 se representa el control de semáforos por medio del modelo general de la maquina de estados.

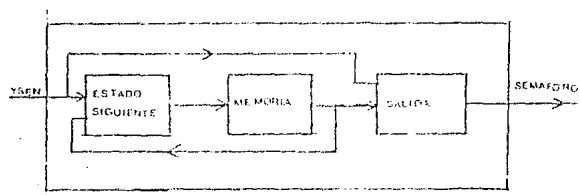


Fig 10.2 Modelo general de la máquina de estados para el controlador de semáforos

Las lineas de entrada y salida que intervienen en el diseño son las siguientes:

ENTRADA
YSEN Señal del sensor
SALIDAS
HS1V Pone el semáforo S1 en verde
HS1A Pone el semáforo S1 en ambar
HS1R Pone el semáforo S1 en rojo
HS2V Pone el semáforo S2 en verde

HS2A	Pone el semáforo S2 en ambar
HS2R	Pone el semáforo S2 en rojo

Y se tiene una señal de reloj de 10 seg.

El paso siguiente es describir mediante un diagrama ASM el funcionamiento del circuito (Fig 10.3)

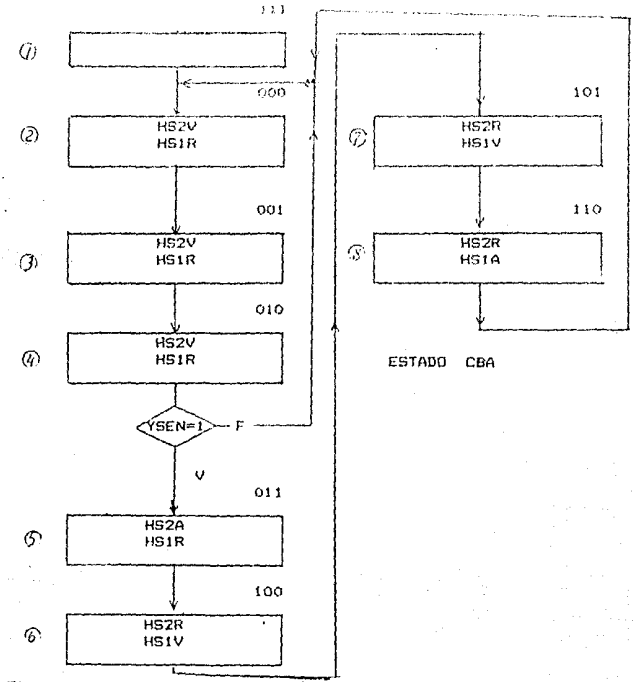


Fig 10.3 Diagrama ASM del controlador de semáforos

(si por accidente la máquina caé en el estado 111 y éste no se encontrara definido, el estado siguiente sería imprevisible, es así que debe ser incluido de tal manera que no active ninguna salidas pero que defina el estado siguiente.)

Las funciones de estado siguiente así como de las salidas se muestran en la figura 10.4, y la implementación completa en la figura 10.5

An \ CnBn	00	01	11	10
0	1	YSEN	0	1
1	0	0	0	0

$An+1=\bar{A}n\bar{B}n+YSEN\ \bar{A}n\bar{C}n$

An \ CnBn	00	01	11	10
0	0	YSEN	0	0
1	1	0	0	1

$Bn+1=\bar{B}nAn+YSEN\ \bar{C}nBn\bar{A}n$

An \ CnBn	00	01	11	10
0	0	0	0	1
1	0	1	0	1

$Cn+1=Cn\bar{B}n+\bar{C}nBnAn$

An \ CnBn	00	01	11	10
0	1	1	0	0
1	1	0	0	0

$HS2V=\bar{A}n\bar{C}n+\bar{C}n\bar{B}n$

An \ CnBn	00	01	11	10
0	0	0	0	0
1	0	1	0	0

$HS2A=\bar{C}nBnAn$

An \ CnBn	00	01	11	10
0	0	0	1	1
1	0	0	0	1

$HS2R=\bar{A}nCn+Cn\bar{B}n$

An \ CnBn	00	01	11	10
0	0	0	0	1
1	0	0	0	1

$HS1V=Cn\bar{B}n$

An \ CnBn	00	01	11	10
0	0	0	1	0
1	0	0	0	0

$HS1A=CnBn\bar{A}n$

An \ CnBn	00	01	11	10
0	1	1	0	0
1	1	1	0	0

$HS1R=\bar{C}n$

Fig.10.4 Determinación de las salidas y funciones de estado siguiente

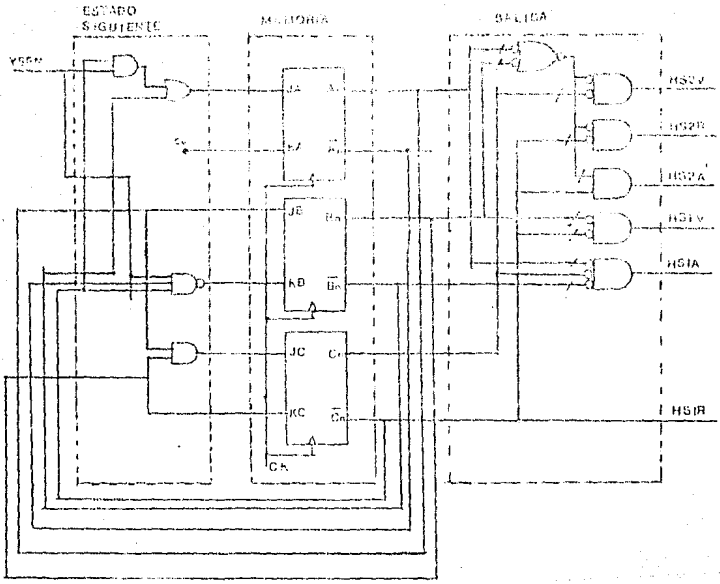


Fig. 10.5 Implementación del controlador de semáforos

REGISTROS

Frecuentemente se requieren almacenes temporales al realizar un diseño, como se verá en los ejemplo siguientes, por lo que resulta conveniente verlo en paréntesis para tratar este tema. En la figura 10.6 se muestra un almacén temporal de datos con una línea de habilitación (enable). Ya en capítulos anteriores se mencionó cómo los flip-flops tipo D son usados para estos almacenamientos temporales, dicha figura muestra la implementación cuando se trata de un solo bit; mientras la línea ENABLE esté activada el dato no llegará hasta la entrada del flip-flop, reflejándose en la salida Q a la siguiente transición de reloj permaneciendo así siempre que la entrada habilitadora esté activada

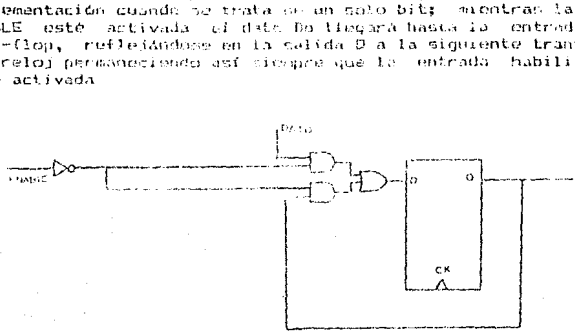


Fig 10.6 Registro con entrada habilitadora

A este circuito o a un conjunto de estos de les llama registros.

Existen registros que desempeñan otro tipo de actividades más complejas, por ejemplo los "registros de corrimientos" que, como su nombre lo indica, realizan un corrimiento a la izquierda o a la derecha de la información. La figura 10.7 muestra un registro de corrimiento a la derecha de 8 bits, TTL, catalogado con los números '165 y 'LS165; los datos se cargan en el registro en paralelo por medio de 8 entradas de datos independientes que son habilitadas cuando la línea de shift/load (recorre/carga)se encuentra en estado bajo, en cambio, la salida de la información se realiza en serie, es decir, por una sola línea, Qh.

Cuenta además con una entrada inhibidora de reloj, que cuando se encuentra en estado alto mantiene la información en el registro en el mismo lugar, esto es, sin que haya corrimiento. Si la línea shift/load se encuentra en estado alto y la inhibidora de reloj en bajo , entonces a cada transición activa de reloj habrá un corrimiento a la derecha. Note que

tambien existe la posibilidad de que la entrada de la información sea en serie utilizando la entrada serial.

Todo este funcionamiento se detalla en la tabla 10.1, mientras que la figura 10.7 muestra su implementación

El registro que se ha descrito es de entrada en paralelo y salida en serie pero en el mercado también se pueden encontrar como circuitos integrados registros de entradas y salidas en serie o en paralelo, entrada en serie y salida en paralelo, para palabras de distintos tamaños y con diversas variantes.

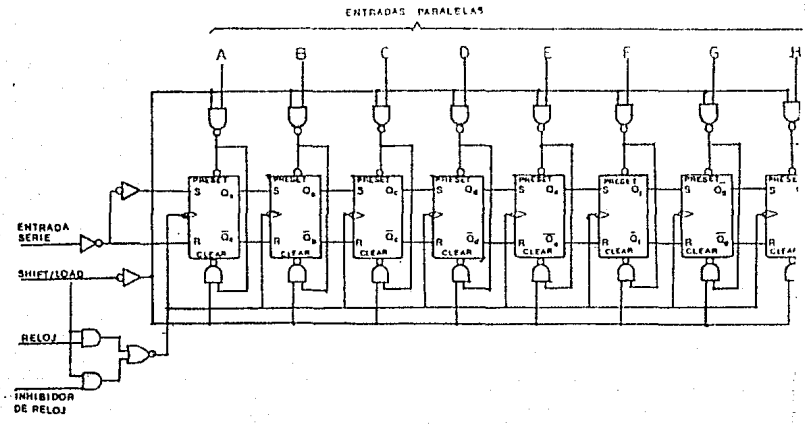


fig 10.7 Registro de corrimiento a la derecha de 8 bits

TRANSMISION DE INFORMACION

Este ejemplo muestra el diseño de un circuito que inicialmente tiene almacenada cierta información en una memoria organizada en 4 palabras de 4 bits; dicha información debe ser transmitida a través de una sola línea, así que es necesario convertir la salida en paralelo de la memoria a serie por medio de un registro de corrimiento.

Se utilizará un contador a 4 cuyas salidas D1 y D0 indican la localidad de memoria que dse está leyendo, dicho contador se incrementa cada vez que se activa la línea HENT y al llegar al cuarto estado activa la señal YSAL, de la misma manera que el contador de la figura 9.4, solo se añade una línea de RESET para inicializarlo.

El registro de corrimiento utilizado es el SN54165 descrito en la figura 10.7. A continuación se muestran las líneas de entrada y salida del módulo de control.

ENTRADAS

YSAL Entrada procedente del contador que se activa cuando la cuenta llega a 4.

YLIS1 Entrada procedente del receptor que inicia la transmisión.

YLIS2 Entrada procedente del receptor para indicar que está listo para recibir el siguiente bit

SALIDAS

HENV Indica al receptor que está transmitiendo un bit.

HINI Indica al receptor que se desea iniciar la transmisión.

LCLOK Activa corrimiento (inhibidor de reloj del registro de corrimiento).

LS/L Línea se Shift/load del registro de corrimiento, permite la carga de la localidad de memoria D1D0 en éste

LSC1 Selecciona el chip de memoria

HR/W Línea le lectura/escritura de la memoria

HENT Incrementa el contador

HRESET Inicializa el contador

En la figura 10.8 se muestra el diagrama de bloques del módulo de control:

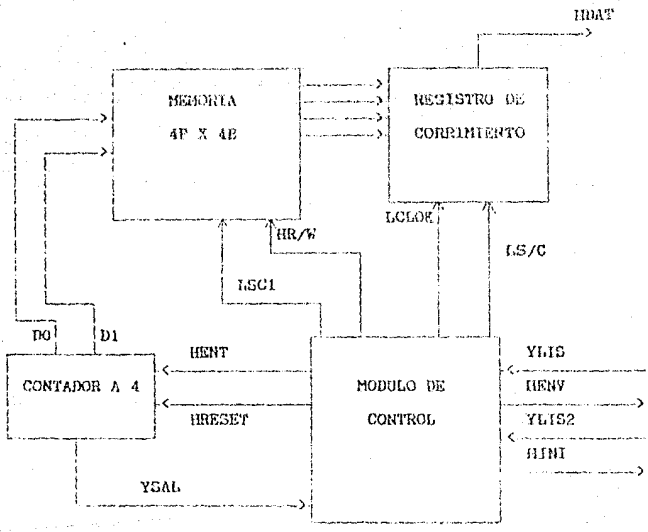


Fig. 10.8 Diagrama de bloques del emisor

Con la finalidad de aclarar el funcionamiento de este emisor se describirá de manera general el funcionamiento del receptor, éste tiene una estructura similar a la del emisor ya que está compuesto de un registro de corrimiento para con una entrada en serie y salida en paralelo, una memoria de 4p X 4b,

un contador a 4 y un módulo de control, como se muestra en la figura 10.9. Note que las líneas que son salidas para el emisor, son entradas para el receptor y viceversa.

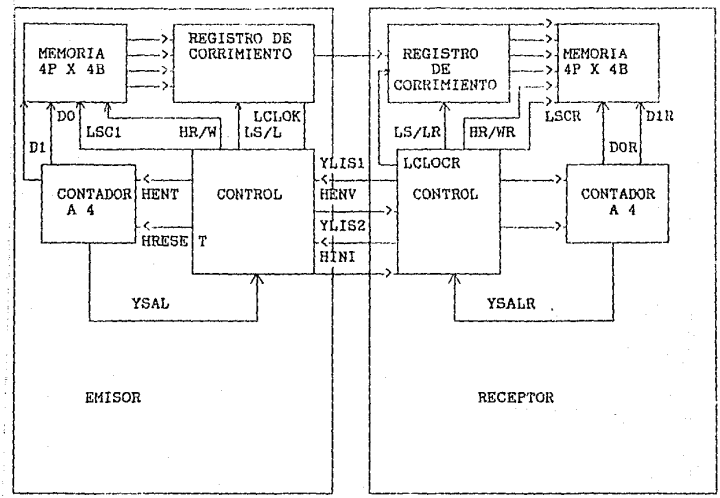


Fig. 10.9 Diagrama de bloques del sistema Emisor-Receptor

Inicialmente el receptor se encuentra en un estado de espera hasta que reciba la señal MINI.

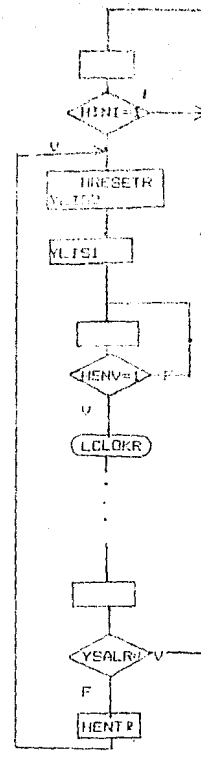
Direcciona la localidad de memoria inicial y avisa al emisor que está listo por medio de la línea YLIS2.

Activa la línea YLIS1 para indicar al emisor que está listo para recibir el primer bit de la palabra.

Se pregunta si ya ha sido enviado el dato (HENV), si es así se hace un corrimiento en el registro para recibir el siguiente bit.

Se repite este proceso hasta que los cuatro bits están en el registro de corrimiento.

Se pregunta si se ha escrito en todas las localidades de memoria (ie, si el contador ha llegado a 4), si no es así se direcciona la siguiente localidad de memoria, incrementando el contador y volviendo a iniciar la transmisión de una palabra, y si ya se ha terminado con todas las localidades se vuelve al estado inicial.



En la 10.10 el diagrama ASM ; la 10.11 los mapas de las variables de estado y las salidas y, finalmente en la figura 10.12 su implementación por medio de multiplexores.

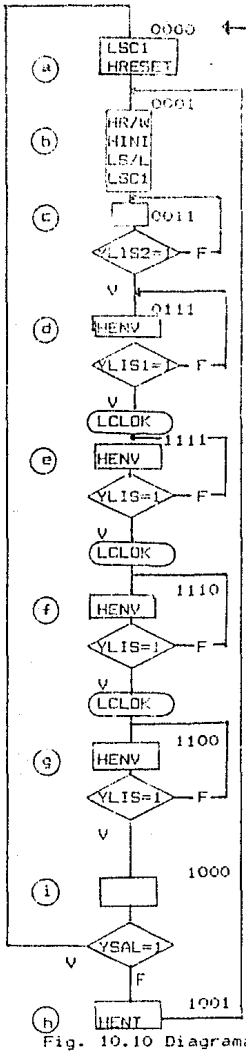


Fig. 10.10 Diagrama ASM del circuito emisor

BnAn \ DnCn	00	01	11	10
00	1	0	0	$\overline{YSAL}$
01	1	0	0	1
11	1	1	$\overline{YLIS1}$	0
10	0	0	0	0

An+1

BnAn \ DnCn	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	1	0
10	0	0	$\overline{YLIS1}$	0

Bn+1

BnAn \ DnCn	00	01	11	10
00	0	0	YLIS1	0
01	0	0	0	0
11	YLIS2	1	1	0
10	0	0	1	0

Cn+1

BnAn \ DnCn	00	01	11	10
00	0	0	1	$\overline{YSAL}$
01	0	0	0	0
11	0	YLIS1	1	0
10	0	0	1	0

Dn+1

BnAn \ DnCn	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	0	1	1	0
10	0	0	1	0

$HENV=CnBnAn+DnCn\bar{A}n$

BnAn \ DnCn	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	0	0

$HIN1=\bar{D}n\bar{C}n\bar{B}nAn$

Fig.10.11 Mapas VEN de las funciones de estado y salidas

BnAn \ DnCn	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	YLIS1	YLIS1	0
10	0	0	YLIS1	0

LCLOK=YLIS1 $\bar{C}$nBnAn+YLIS1 DnCnBn

BnAn \ DnCn	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	0	0

LS/L=$\bar{D}$n$\bar{C}$n$\bar{B}$nAn

BnAn \ DnCn	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	0	0

LSC1=$\bar{D}$n$\bar{C}$n$\bar{B}$n

BnAn \ DnCn	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	0	0

HR/W=$\bar{D}$n$\bar{C}$n$\bar{B}$nAn

BnAn \ DnCn	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

HRESET=$\bar{D}$n$\bar{C}$n$\bar{B}$n$\bar{A}$n

BnAn \ DnCn	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	0	0	0	0
10	0	0	0	0

HENT=Dn$\bar{C}$n$\bar{B}$nAn

Fig 10.11 Cont.

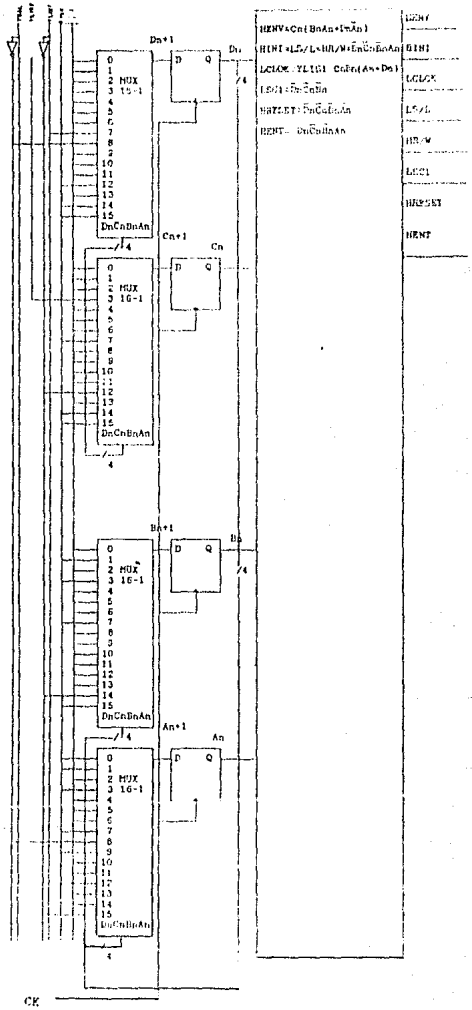


Fig 10.12 Implementación del emisor

Veamos ahora un último ejemplo se desea enviar un mensaje a traves de un sistema de comunicaciones ya establecido, en donde para indicar el principio de la transmisiòn deben aparecer tres estados de voltaje alto consecutivos por una línea R; los datos en esta línea se han sincronizado con una fuente de pulsos de reloj. Se debe diseñar un circuito secuencial que tenga una salida Z que se activarà cuando el tercer estado alto consecutivo haya sido detectado. El circuito servirá para advertirle al receptor sobre la iniciaciòn de un mensaje.

Entonces se tiene un sistema como el mostrado en la figura 10.13, junto con su diagrama de tiempos.

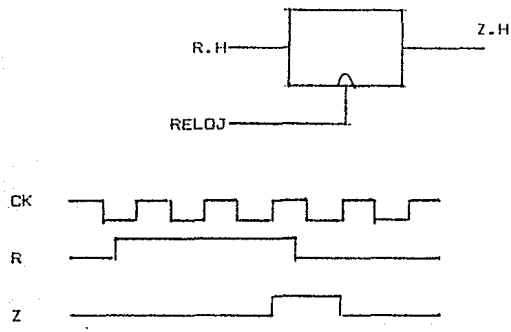


Fig 10.13 Diagrama modular y de tiempo

En la figura 10.14 se muestra el diagrama de estados del circuito; note que si los tres estados altos no se presentan consecutivamente el circuito vuelve a su estado inicial. Supòngase que por alguna razòn este circuito debe ser implementado con flip-flops JK; entonces es necesario realizar la tabla de exitaciòn del circuito de donde se obtendràn las funciones booleanas de las funciones de estado siguiente y la salida como se muestra en la figura 10.15, para finalmente implementarlo como se muestra en la figura 10.16

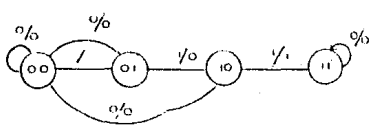
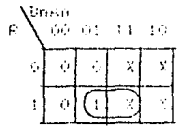
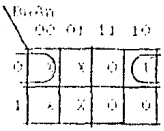


Fig 10.14 Diagrama de estados

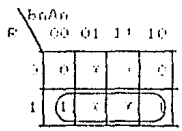
ESTADO PRESENTE		ENTRADA	ESTADO SIGUIENTE		SALIDA	ENTRADAS DE FLIP-FLOP			
Bn	An	P	Bn+1	An+1	Z	JB	KB	JA	KA
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	1	X
0	1	0	0	0	0	0	X	X	1
0	1	1	1	0	0	1	X	X	1
1	0	0	0	0	0	X	1	0	X
1	0	1	1	1	1	X	0	1	X
1	1	0	1	1	0	X	0	X	0
1	1	1	1	1	0	X	0	X	0



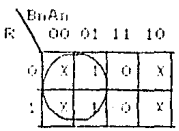
$JB=PA_n$



$KB=P\overline{A_n}$



$JA=P$



$KA=\overline{B_n}$

Y directamente de la tabla se tiene $Z=B_nA_nP$

Fig. 10.15 Tabla de excitación y reducción de funciones booleanas

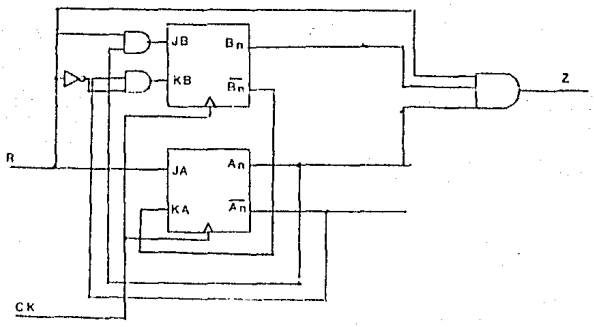


Fig 10.16 Implementación con flip-flop's JK

- Alonso Marcelo, Finn J. Edward
  FISICA III: FUNDAMENTOS CUANTICOS Y ESTADISTICOS
  Fondo Educativo Interamericano, México D.F., 1976

- Barton C. Thomas
  DIGITAL COMPUTER FUNDAMENTALS
  Mc Graw-Hill, 7a. Ed., Toiyo Japan, 1766

- Boylestad Robert, Nashelsky
  ELECTRONICA TEORIA DE CIRCUITOS
  Prentice-Hall, España, 1985

- Brophy J.J.
  ELECTRONICA FUNDAMENTAL PARA CIENTIFICOS
  Reverte, 2a. Ed., España, 1979

- Clare Christopher.
  DESIGNING LOGIC SYSTEMS USING STATE MACHINES
  Mc Graw-Hill, Maidenhead, 1972

- Dempsey A. John.
  ELECTRONICA DIGITAL BASICA
  Fondo Educativo Interamericano, México D.F., 1984

- Feyman P. Richard, Leighton B. Robert, Sands Matthew
  VOLUMEN III: MECANICA CUANTICA
  Fondo Educativo Interamericano, USA, 1971

- Fletcher I. William
  AN ENGINNERING APPROACH TO DIGITAL DESIGN
  PRENTICE-HALL, NEW JERSEY

- Green David
  MODERN LOGIC DESIGN
  Addison-Wesley Publishing, Great Britain, 1986

- Linter M. Paul
  MIXED LOGIC: A TOOL FOR DESIGN SIMPLIFICATION
  Computer Design, August 1971

- Malvino Albert Paul, Leach P. Donald
  DIGITAL PRINCIPLES AND APPLICATIONS
  Mc. Graw-Hill Co. [illegible] California, 1976

- Mano Morris
  COMPUTER LOGIC DESIGN
  PRENTICE-HALL, Englewood Cliffs N.J.

- [illegible] R. Paul
  [illegible] LOGIC: A TOOLS FOR DESIGN SIMPLIFICATION
  Computer Design, August 197[illegible]

- Taub Herbert, Schilling Donald
  DIGITAL INTEGRATED ELECTRONICS
  Mc Graw-Hill, [illegible], [illegible]

- Wiatrowski Claude A., House Charles H.
  LOGIC CIRCUITS AND MICROCOMPUTER SYSTEMS
  Mc Graw-Hill, Singapore, 198[illegible]

- Winkel David, Prosser Franklin
  DIGITAL DESIGN: AN INTRODUCTION TO TOPDOWN DESIGN

- Winkel David, Prosser Franklin
  THE ART OF DIGITAL DESIGN
  Prentice-Hall, Englewood Cliffs N.J.