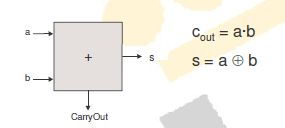
**APLICACION DE CIRCUITOS**

**Semisumador – Sumador Parcial**

El semisumador (half adder) es un circuito que suma dos bits de entrada a y b y devuelve un bit de resultado s y un bit de acarreo cout.



La suma de dos bits (2 variables de 2 estados cada una) se representa mediante la siguiente tabla de verdad.

|  |  |  |  |
| --- | --- | --- | --- |
| Sumandos | | Suma Binaria | Acarreo |
| a | b | S | c |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Se puede obtener la función suma, desarrollando la expresión donde la suma es válida. Esto es donde se hace 1, en este caso se representa por la forma canónica disyuntiva o por Minterm.

En el caso del acarreo es mismo y se representa algebraicamente.

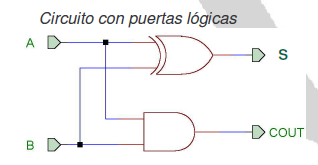
La función S(a,b) como canónica disyuntiva es: 

La función C (a,b) como canónica disyuntiva es: C ( a,b) = a.b

La construcción con compuertas lógicas (combinacional) del semisumador será:



La operación de suma se realiza a través de una compuerta O-exclusiva y el acarreo a través de una and.



Las entradas a ambas compuertas pueden obtenerse de la misma barra.

**Sumador completo**

El sumador completo (full adder) es un circuito que suma dos bits de entrada a y b más un acarreo de entrada cin y devuelve un bit de resultado s y un bit de acarreo cout.

|  |  |
| --- | --- |
|  |  |

Cuando uno desea sumar dos bits, que conforman parte de un número binario, es necesario sumar a ambos, el acarreo precedente de la suma de los bits de proceso inmediato inferior.

El semisumador no permite hacer esto y para ello se diseña un nuevo sistema combinacional, llamado sumador total, cuya tabla de verdad es como sigue.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cn | a | b | S | Cn+1 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

El acarreo procedente de la etapa anterior se lo identifica como Cn y al que se produce en la suma que estoy analizando se lo identifica como Cn+1



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cn\ab | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |

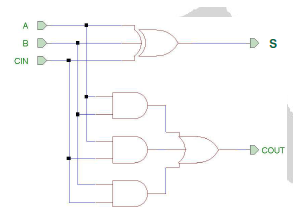
Si aplicamos Karnaugh nos queda:





Y la representación a través de compuertas será:



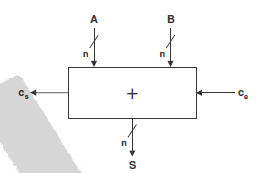


El sumador completo es de uso general, por lo que ha sido fabricado como un solo bloque.

Se representa como un bloque, indicando sus entradas y salidas.

Sumador Total paralelo de n bit.

Se construye asociando n sumadores elementales completos (full adder) que reciben y procesan todos ellos los datos en paralelo. El acarreo se propaga en serie de un sumador a otro.



El sumador total realiza simultáneamente la suma de dos números para n bits y para ello utiliza n sumadores completos.

La figura que sigue muestra la construcción de un sumador total para “n bits”, tomando de a dos.



Figura 5

**Decodificadores y Codificadores.**

Ya vimos en la Unidad I que el significado de un código era obtener la relación biunívoca entre dos símbolos, por ende para obtener esa relación es necesario generar una matriz de doble entrada, donde una de ellas son los símbolos del sistema A y la otra los del sistema B, donde la función sea cierta para ambas, se producirá la relación biunívoca.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SIMBOLOS DEL SISTEMA B | SIMBOLOS DEL SISTEMA A | | | | | | | | |
|  | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |
| B1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| B3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| B4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| B5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| B6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| B7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| B8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

En la tabla se observa que para el símbolo A1 del sistema A le corresponde uno y solo un símbolo del sistema B, en este caso B1, es decir que se ha desarrollado un código que relaciona los dos sistemas.

Esta relación entre los dos sistemas, que se denomina codificación, puede darse a su vez entre dos sistemas que ya sean códigos en sí mismo, como por ejemplo el sistema binario natural y el decimal, ambos son códigos del sistema de numeración.

Para este caso, si queremos convertir un número dado por el sistema de codificación binaria natural al sistema de codificación decimal, decimos que estamos decodificando el primer sistema ya que usamos como referencia el decimal.

**Codificadores.**

Los codificadores nos permiten compactar la información, generando un código de salida a partir de un código de entrada. Para explicar esto de una forma más clara utilizaremos un ejemplo.

Un ejemplo de codificador de 4 a 16

Ejemplo.

Imaginemos que estamos diseñando un circuito digital que se encuentra en el interior de una cadena de música. Este circuito controlará la cadena, haciendo que funcione correctamente.

Una de las cosas que hará este circuito de control será activar la radio, el CD, la cinta o el Disco según el botón que haya pulsado el usuario. Imaginemos que tenemos 4 botones en la cadena, de manera que cuando no están pulsados, generan un ’0’ y cuando se pulsa un ’1’ (Botones digitales). Los podríamos conectar directamente a nuestro circuito de control la cadena de música, como se muestra en la figura.



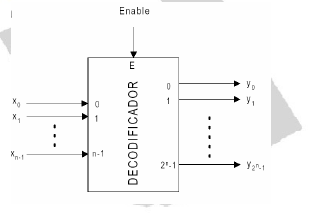
Sin embargo, a la hora de diseñar el circuito de control, nos resultaría más sencillo que cada botón tuviese asociado un número. Como en total hay 4 botones, necesitaríamos 2 bits para identificarlos. Para conseguir esta asociación utilizamos un codificador, que a partir del botón que se haya pulsado nos devolverá su número asociado:



**Decodificador**

Un decodificador (o decodificador de n a 2n) es un módulo combinacional con n entradas y 2n salidas, además de una señal de activación (Enable) de entrada.

El decodificador activa la salida i-ésima cuando se presenta la combinación binaria i en las entradas, siempre y cuando el módulo esté activo (enable=1): se activa la salida correspondiente al número binario codificado en la entrada.



Un decodificador es un circuito integrado por el que se introduce un número y se activa una y sólo una de las salidas, permaneciendo el resto desactivadas.

Ejemplo.

Queremos realizar un circuito de control para un semáforo. Los estados de un semáforo se pueden definir como verde, amarillo, rojo y azul. A cada uno de estos colores le asignamos un número como por ejemplo verde = 0, amarillo = 1, rojo = 2, Azul = 3.



Para controlar este semáforo podemos hacer un circuito que tenga 4 salidas, una para cada una de las luces. Cuando una de estas salidas este a “1”, la luz correspondiente estará encendida. Sin embargo tal como se menciono no pueden estar dos luces encendidas al mismo tiempo.



Si utilizamos un decodificador de 2 a 4, conseguiremos controlar el semáforo asegurándonos que sólo estará activa una luz en cada momento.

Tabla de Verdad del decodificador de 2 a 4.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| E0 | E1 | Q1 | Q2 | Q3 | Q4 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

**Ejemplo de un Decodificador Decimal.**

SI realizamos una tabla de la verdad, donde como entradas tenga al sistema binario y como salida el sistema decimal, podemos observar donde la función del número decimal se hace uno para cada número binario, es decir donde la función de números decimales es cierta para cada número binario.

Desarrollando los términos canónicos de estas funciones y construyendo los circuitos

Combinacionales que las representen obtendremos el circuito electrónico que representará el decodificador decimal.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A0 | A1 | A2 | A3 | f(0) | f(1) | f(2) | f(3) | f(4) | f(5) | f(6) | f(7) | f(8) | f(9) | Dec |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 5 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 6 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 7 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 8 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 9 |

Cada una de las funciones de salida obtenida puede desarrollarse como sigue:

|  |  |
| --- | --- |
| f(0) = | f(5) = |
| f(1) = | f(6) = |
| f(2) = | f(7) = |
| f(3) = | f(8) = |
| f(4) = | f(9) = |

De estas expresiones podemos construir el circuito combinacional que tendrá la siguiente forma:



Cada función fn está expresada como un término canónico disyuntivo, producto de la n variables por lo que las salidas de las conversión son llaves AND de 4 entradas.

De las señales se obtiene la variable negada a través de un inversor.

De esta forma se puede realizar un decodificador que cumpla con las expresiones planteadas.

Síntesis del uso de los Decodificadores.

Un decodificador permite materializar todos los minterms de una función de n variables.

Por lo tanto se puede usar para sintetizar cualquier función de n variables expresadas como suma de minterms sin más que usar un decodificador de n a 2n y una puerta OR con tantas entradas como sumandos tenga la expresión de la función.

**Multiplexores – Selectores de Datos.**

Un multiplexor (o multiplexor de 2n a 1) es un módulo combinacional con 2n entradas y 1 salida, además de una señal de activación y n señales de control.

La función que realiza un multiplex es la de seleccionar una de N entradas de datos y transmitir el dato seleccionado a un canal de información único.

Este es muy útil por ejemplo en comunicaciones de datos, cuando entre dos extremos disponemos de un único enlace físico y cada extremo a su vez está conectado a varios transmisores/receptores de datos.

Este se puede ser realizado a través de compuertas.



En este circuito se observa que las entradas de datos Xn van conectadas a una compuerta AND, a su vez en la misma compuerta una combinación de las entradas de selección A ó B y la habilitación I.

Entonces para que una salida de compuerta AND sea cierta, todas sus entradas deben estar en 1.

Como se observa en la figura, no hay ninguna combinación posible de las entradas de selección que permite que haya más de una AND habilitada por vez, por ende una Xn a la salida por vez.

Si en las entradas de selección se coloca un contador de dos bits, va a ir dando habilitación en forma secuencia a cada una de las entradas de datos, esto producirá una salía en forma de tren de datos donde estarán intercaladas las entradas de datos.

Si al otro extremo de la línea se coloca un sistema que realice la operación inversa, Demultiplexor, hemos logrado transmitir las señales que aparecen simultáneamente (en paralelo) en un extremo, sobre un único sistema físico (serie), para luego convertir esta cadena de información en varias salidas (paralelo).

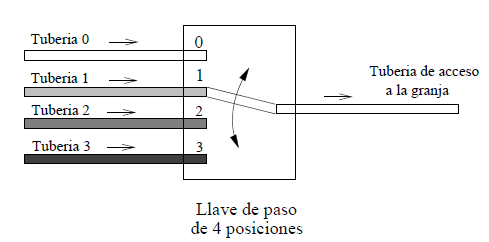
Estos sistemas e usan frecuentemente en enlace y transmisión de datos y también en telefonía.

Un único multiplexor de 2n a 1 permite materializar cualquier función deconmutación de n variables.

Ejemplo.

Imaginemos que hay dos tuberías (canales de datos) por el que circulan distintos fluidos (datos). Una transporta agua para regar y la otra agua potable.

Estas tuberías llegan a una granja, en la cual hay una única manguera por la que va a salir el agua (bien potable o bien para regar), según lo que seleccione el granjero posicionando la llave de paso en una u otra posición. En la figura se muestra un esquema. Las posiciones son la 0 para el agua potable y 1 para el agua de regar.



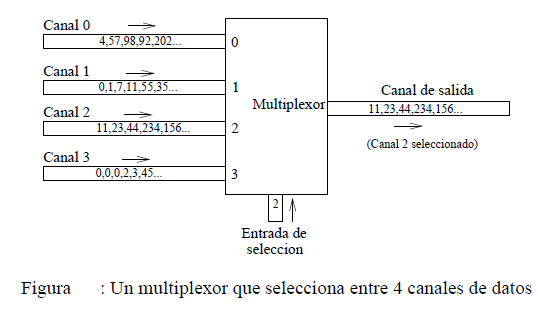
Moviendo la llave de paso, el granjero puede seleccionar si lo que quiere que salga por la manguera es agua potable, para dar de beber al ganado, o agua para regar los cultivos. Según cómo se posicione esta llave de paso, en la posición 0 ó en la 1, seleccionamos una tubería u otra.

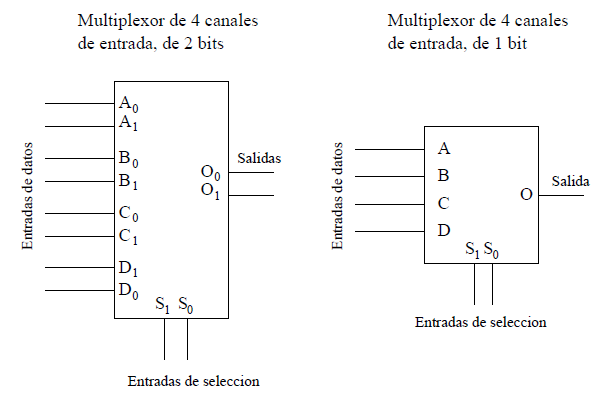
Pero ¿por qué sólo dos tuberías? Porque es un ejemplo. A la granja podrían llegar 4 tuberías.

En este caso el granjero tendría una llave de paso con 4 posiciones, como se muestra en la siguiente figura. Esta llave se podría poner en 4 posiciones distintas para dar paso a la tubería 0, 1, 2 ó 3. Obsérvese que sólo pasa una de las tuberías en cada momento, ¡y sólo una! Hasta que el granjero no vuelva a cambiar la llave de paso no se seleccionará otra tubería.

Con este ejemplo es muy fácil entender la idea de multiplexor. Es como una llave de paso, que sólo conecta uno de los canales de datos de entrada con el canal de datos de salida.

Ahora en vez de en tuberías, podemos pensar en canales de datos, y tener un esquema como el que se muestra en la figura a continuación, en la que hay 4 canales de datos, y sólo uno de ellos es seleccionado por el multiplexor para llegar a la salida. En general, en un multiplexor tenemos dos tipos de entradas:





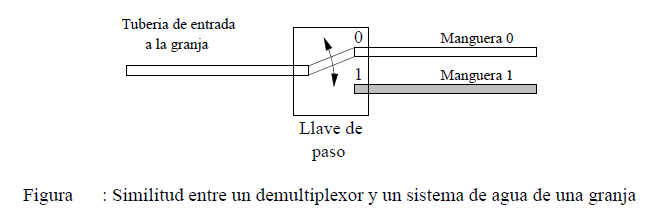
**Demultiplexores.**

El concepto de demultiplexor es similar al de multiplexor, viendo las entradas de datos como salidas y la salida como entradas. En un multiplexor hay varias entradas de datos, y sólo una de ellas se saca por el canal de salida. En los demultiplexores hay un único canal de entrada que se saca por una de las múltiples salidas.

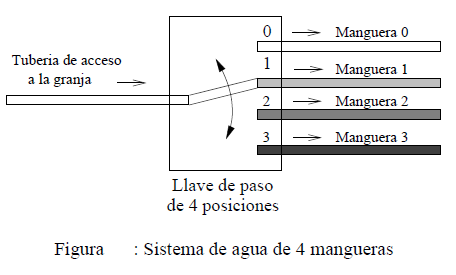
Si utilizamos el símil de la granja y las tuberías, podemos imaginar el siguiente escenario.

Supongamos que ahora a la granja le llega una única tubería con agua, pero en el interior de la granja hay varias mangueras, cada una para limpiar una zona del establo o dar de beber a los animales de esa zona. Cómo sólo hay un granjero, sólo podrá usar una de las mangueras cada vez (el granjero no podrá usar a la vez dos mangueras, porque están en sitios diferentes)

Para seleccionar qué manguera quiere usar en cada momento, hay una llave de paso, de manera que si la sitúa en una posición, el agua que viene por la entrada saldrá por la manguera 0, mientras que si la sitúa en la otra posición, el agua saldrá por la manguera 1 (ver figura)

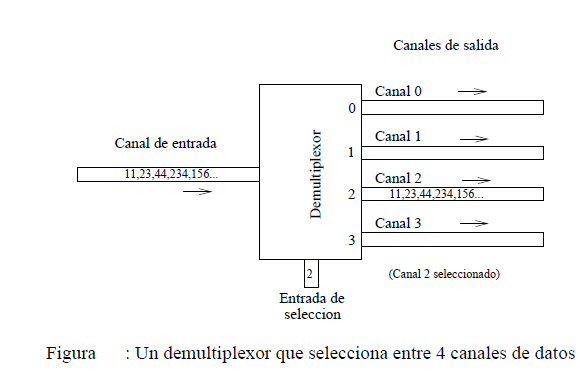


De la misma manera que en los multiplexores puede haber varias entradas, en los demultiplexores puede haber varias salidas. Por ejemplo en la figura 5.6 se muestra el mismo sistema de tuberías de la granja, pero ahora hay 4 mangueras, para llegar a 4 zonas distintas de la granja.



Ahora el granjero tendrá que posicionar la llave de paso en una de las 4 posiciones posibles, para que el agua salga por la manguera seleccionada.

Ya comprendemos cómo funcionan los demultiplexores. Si lo aplicamos al mundo de la electrónica, en vez de tuberías tendremos canales de datos. Habrá un único canal de entrada, por el que llegarán números, que saldrán sólo por uno de los canales de salida, el que tengamos seleccionado, como se muestra en la figura.



En general en un demultiplexor tendremos:

Una entrada de datos

Una entrada de selección: que indica a cuál de las salidas se manda la entrada

Varios canales de datos de salida. Sólo estará activo el que se haya seleccionado.

**Detectores y Generadores de Paridad**

Una operación aritmética que se emplea a menudo en los sistemas digitales, es la forma de determinar si la suma de bits binarios en una información es par o impar. La salida de una compuerta O-exclusiva es 1 si, y sólo sí, una entrada es 1 y otra es 0. Dicho de otra manera, la salida es 1 si la suma de los dígitos es 1.

El sistema de la figura no sólo es un comprobador de paridad, sino que también puede emplearse para generar un bit P que fije la paridad. Independientemente de la paridad de la señal de entrada de 4 bits, la paridad del código de cinco bit A, B, C, D es impar (par), entonces P es 0 (1), y en consecuencia la suma de A, B, C, D y P siempre es par.

El empleo de un código de paridad aumenta la seguridad de transmisión de una información binaria.

Lo indicado se muestra en la figura, donde se genera un bit de paridad P1 que se transmite conjuntamente con las señales de entrada de N bit. En el receptor se comprueba la paridad de la señal aumentada (N + 1) bit, y si la salida P2 de comprobador es 0 se puede suponer que no hay ningún error en la transmisión del mensaje, mientras que si P2 = 1 indica que (por ejemplo debido al ruido) hay un error en la recepción.

**Comparadores Digitales.**

A veces es necesario saber si un número binario A es mayor, menor o igual que otro número B.

El sistema para determinarlo, se llama comparador de valor digital (o binario).

Para obtener la función realizamos la tabla de verdad.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | A=B | A>B | A<B |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |

De estas tablas de la verdad se pueden obtener ecuaciones para las cuales las funciones son ciertas (1).

Luego de estas combinaciones se las representa mediante circuitos Combinacionales.

Las ecuaciones son:



**Acarreo en Serie y Acarreo Paralelo**

Los registros de desplazamiento son construidos a través de Flip-Flop.

Tiene características de memoria y función de desplazar datos. También son utilizados para convertir datos de:

Serie a Paralelo.

Paralelo a Serie.

Estos son identificables por la forma en la que leen los datos en la unidad de almacenamiento.

Existen cuatro categorías:

Entrada Serie – Salida Serie.

Entrada Serie – Salida Paralelo:

Entrada Paralelo – Salida Serie:

Entrada Paralelo – Salida Paralelo.

El tratamiento del acarreo puede realizarse de dos formas en serie y en paralelo.

**Generación del Acarreo en Serie.**

En este montaje cada sumador realiza la suma de dos bits y el acarreo procedente del sumador de los bits de peso inmediato inferior.

La Figura anterior representa el esquema de bloques de un sumador de n-bits de este tipo. En él se observa que el acarreo se propaga en serie de un sumador al siguiente y, por lo tanto, el tiempo necesario para que se realice la suma es igual a “n” veces el tiempo que tarda en generarse el acarreo en un sumando.

Este montaje es adecuado para la realización de sumadores en los que el tiempo de operación no tiene que ser reducido al mínimo.

**Generación del Acarreo en Paralelo**

En este montaje los acarreos de todas las etapas son generados simultáneamente.

La expresión del acarreo de la etapa n en función de la etapa anterior (n-1) es



De esta ecuación se deduce que Cn puede tomar el valor uno, si los dos bits de la etapa an y bn toman el valor uno simultáneamente (ver primer término de la ecuación) o bien si cualquiera de ellos es uno y a su vez, lo es también el acarreo de la etapa anterior (ver el segundo término de la ecuación)

El término anbn genera su acarreo en la propia etapa y se le denomina generador Gn (ver la tabla de verdad del semisumador).

El término (an + bn) permite la propagación del acarreo de la etapa anterior y se le llama Propagador Pn (ver circuito del sumador total).

Por lo tanto Cn se puede expresar como: 

Si denominamos C0 al acarreo que se aplica a la primera etapa, resultará que el acarreo de la

Primera etapa es:



Para la segunda etapa:



Para la tercera etapa:

Para la cuarta etapa:

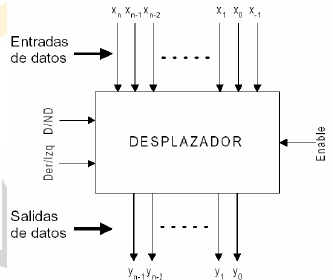
Observando estas ecuaciones podemos deducir, que si a un sumador total, del cual extraemos la función suma y la de acarreo propio de la etapa, la podemos extraer de los términos G1 y P1, con ellos y el acarreo que se le aplica a la primera etapa C0 se puede generar en forma simultánea todos los acarreos, por lo que las distintas etapas lo tendrán disponibles al mismo tiempo, esto configura un sumador en paralelo.

Los bloques donde se producen los acarreos paralelos, que en la figura anterior se representan con Bn, están construidos como circuitos integrados, construyendo un bloque funcional denominado generador de acarreos.

**Registros de Desplazamiento.**

Un desplazador (shifter) es un módulo combinacional con n+2 entradas de datos y n salidas, además de una señal de activación y señales de control.

El desplazador puede mover o no bits a derecha e izquierda en desplazamientos abiertos o cerrados (rotaciones) bajo las órdenes de las señales control.



Puesto que un Biestable es una memoria con un bit, con n FLIP-FLOP, cabe almacenar una información de n bits. Esta combinación toma el nombre de registrador. Los datos sucesivos de una información pueden leerse en un registrador serie, en el que la salida de cada FLIP-FLOP se conecta con la entrada del siguiente, esta configuración se denomina registro desplazamiento.

Cada FLIP-FLOP de la figura es del tipo S-R. Obsérvese que la etapa en la que se almacena el bit más significativo (MSB) se convierte en un Biestable tipo D conectado S y R a través de un inversor.

El registro desplazamiento de 5 bits indicado en la figura está disponible en un solo chip.

Su funcionamiento es el siguiente.

Suponiendo que se va a registrar la serie de datos 0001, el digito menos significativo es el que está más a la derecha del caso 1.

Los FLIP-FLOP se borran por la entrada de un 0 en la entrada de borrado y así cada Q0, Q1,……., Q4 es 0. Luego Cr se pone a 1 y Pr permanece constantemente igual a 1.

Se aplica entonces

el tren de datos de n serie y los impulsos del reloj sincronizados. El bit menos significativo (LSB) se introduce en el FF4 cuando CK cambia de 0 a 1. Después del impulso del reloj, Q4 = 1 mientras que las otras salidas permanecen en 0.

Al segundo impulso de reloj el estado de Q4 se transfiere al Biestable ordenador de FF3 por la acción de un FLIP-FLOP tipo SR. Simultáneamente el siguiente bit (un 1 en la información 01011) entra en el ordenador FF4, después del segundo impulso del reloj, el bit de cada ordenador se transfiere a su seguidor y Q4 = 1, Q3=1 y las otras permanecen en 0.

La siguiente tabla de las lecturas del registrador después de cada impulso.



Podemos seguir fácilmente el procedo y ver que registrando cada bit en el FLIP-FLOP MSB y pasándolo a la derecha para dejar sitio al próximo dígito, la información de entrada queda fijada en el registro al cabo del n-esimo impulso de reloj (para un código de n bit).

Naturalmente, los impulsos deben detenerse en el momento en el que la información quede registrada, cada salida queda disponible en una línea distinta y pueden leerse simultáneamente.

El convertidor se denomina de Serie a Paralelo debido a que la información de entra en serie y sale en paralelo. También puede llamarse de Entrada en Serie y Salida en Paralelo.

Un código temporal (los bits dispuestos en función del tiempo) se ha transformado en un código especial (información almacenada en una memoria estática)

**Registro de Desplazamiento Serie – Serie**

Podemos tomar la salida Q0 y leer el registrado en serie si aplicamos n impulsos de reloj, en el caso de una información de n bit. Después del n-ésimo impulso, cada FLIP-FLOP queda en 0. Obsérvese que la cadencia del reloj puede ser mayor o menor que la frecuencia de impulsos originales. Por lo tanto este sería un método de cambiar el espacio en tiempo de un código binario.