Prefacio

El presente documento forma parte del programa de estudios de la carrera Electrónica ofrecida a sus estudiantes en el Instituto Universitario de Tecnología para la informática – Iutepi. Sirve de apoyo complementario bajo la modalidad de autoaprendizaje publicado en su campus virtual, a todos los alumnos que cursan la materia.

Tener el conocimiento básico es necesario para la comprensión y desenvolvimiento en el área no solo de la electrónica sino también de los sistemas y las señales digitales.

En el texto se ha reunido un material académico que permite introducir al estudiante en los principios y fundamentos para desarrollarse en el campo de la lógica digital, el contenido esta hecho en forma modular para permitir al estudiante ir adquiriendo los conocimientos y debida comprensión de los temas conforme avanza en el contenido.

El termino digital deriva en la forma en que las computadoras realizan operaciones usando dígitos, hoy dia la tecnología digital tiene aplicación en un gran rango de área, como la telefonía, televisión, comunicaciones, instrumentos médicos entre otros.

El sistema de numeración binario y los códigos digitales son la base de los computadores y en general de la electrónica. En el texto estudiaremos el sistema de numeración binario y su relación con otros sistemas de numeración como el sistema decimal, el sistema octal y el sistema hexadecimal. Se hace el análisis y estudio de las operaciones aritméticas básicas (Suma-Resta-Multiplicación) en números binarios, así como el análisis de los códigos digitales como el BCD y el GRAY.

Haremos un estudio particular en el funcionamiento y manejo de compuertas lógicas (AND-OR-NOT) así como el análisis de la tabla de verdad de cada compuerta o de un circuito conformado por la combinación de estas. Se hará uso del algebra de Boole que permitirá el análisis sistemático haciendo uso de mapas de Karnaugh, de los circuitos conformados por compuertas lógicas y analizaremos los circuitos combinacionales así como los circuitos secuenciales que son la base de los equipos de cómputo y almacenamiento como microprocesadores y microcontroladores y las memorias.

Al Estudiante:

Casi todo está digitalizado en la actualidad, desde electrodomésticos de uso cotidiano hasta nuestro teléfono celular, los avances en tecnología como Internet de las cosas (IoT) y Blockchain, crean la necesidad de conocer y manejar las bases y fundamentos de las tecnologías digitales, para así poder ingresar al campo de trabajo con conocimientos y una alta capacitación.

La lógica digital programable está presente y no desaparecerá, seguirá creciendo y mejorando por lo que es fundamental contar con futuros profesionales preparados en el área.

Contenido del programa de estudios

Sistemas de Numeración: Decimal, Binaria, Octal y Hexadecimal. Conversiones.

En los sistemas digitales, la información está codificada en ceros y unos, ya que utilizan voltajes digitales que sólo pueden ser de valor alto y bajo (High/Low). Para entender cómo se hace, empezaremos por estudiar la representación numérica. Y para ello vamos a hacer uso de la forma en que habitualmente manejamos los números enteros. Este método se llama sistema de numeración posicional, ya que el principio de este sistema es que cada cifra que escribimos tiene un valor en función de la posición que ocupa en el número, esta posición va ordenada de derecha a izquierda, entendiendo que los dígitos más a la izquierda en un numero son de mayor valor o peso que aquel que se encuentra a su derecha.

Sistema DECIMAL:

El sistema decimal es el utilizado por todos de manera habitual, está compuesto por diez dígitos que van desde el cero (0) hasta el nueve (9), su base es el número diez (10) y la forma en que los números tienen su valor o peso, depende de la posición en la cual esté se encuentre ubicado el digito, por tanto para un primer digito decimal, ubicado en la posición de la unidad (posición cero), su valor o peso será de diez elevado a cero y multiplicado por el digito en dicha posición.

Por ejemplo, un único digito, el número tres (3) en la posición de unidad, tendría el valor decimal, diez elevado a cero, multiplicado por tres, lo cual resulta en tres como valor decimal.

$$3 = 3x10^0$$

Ahora agreguemos un segundo digito ubicado en la posición decimal conocida como decena (posición uno), el digito ubicado en esta posición tendrá un valor o peso de diez elevado a uno multiplicado por el digito en esa posición.

Por ejemplo, el número cuarenta y tres (43) decimal, está formado por el digito tres en la posición cero y el cuatro en la posición uno, así que, el valor o peso de este número es:

$$43 = 4x10^1 + 3x10^0$$

En la medida que se van agrupando más dígitos en diferentes posiciones (el número aumenta de valor), la posición de cada digito se convierte en el exponente numérico por el cual la base se multiplicara igual número de veces, posteriormente la suma total será igual al número que forman dichos dígitos.

Ejemplo: Numero decimal 325

$$325 = 3x10^2 + 2x10^1 + 5x10^0$$

$$325 = 3x10x10 + 2x10 + 5$$

$$325 = 300 + 20 + 5$$

Sistema OCTAL:

El sistema octal, está compuesto por un total de ocho dígitos, que van desde el digito cero (0) hasta el siete (7), su base es el número ocho (8). Al igual que en sistema decimal, la posición de cada digito octal estará asociada a una multiplicación de dicho digito por la base ocho elevado a la posición del digito.

Por ejemplo el número octal $505_{(8)}$, puede ser escrito en decimal, observando que el digito 5 está en la posición cero, el 0 en la posición uno y 5 en la posición dos; por tanto:

$$505_{(8)} = 5x8^2 + 0x8^1 + 5x8^0$$

$$505_{(8)} = 5x64 + 0x8 + 5x1$$

$$505_{(8)} = 320 + 0 + 5$$

$$505_{(8)} = 325_{(10)}$$

De esta forma observamos que el número quinientos cinco base ocho ($505_{(8)}$) es el numero trescientos veinticinco base diez ($325_{(10)}$), implícitamente se puede observar que esta forma permite convertir un numero octal en un numero decimal.

Sistema Hexadecimal:

El sistema Hexadecimal, su base es el 16, así que se compone de un total de 16 dígitos, que van desde el 0 hasta 15 decimal, sin embargo para diferenciarlo del sistema decimal, los dígitos que continúan después del nueve decimal, se representan con las primeras letras del abecedario en mayúscula, así que el digito 10 se representa con la letra A, el 11 con la letra B y así hasta la letra F que representa el digito 15. De esta manera tenemos que los 16 dígitos Hexadecimales se representan de esta forma: 1 2 3 4 5 6 7 8 9 A B C D E F

En la actualidad, las direcciones IP versión 6 (IPv6), están basadas en números Hexadecimales, esto permite un gran números de direcciones que facilitan el crecimiento de la interconexión en la redes de comunicaciones e informática. Al igual que los sistemas de numeración anteriormente descritos, la posición de un digito en sistema hexadecimal, tendrá un valor equivalente a ese digito multiplicado por la base elevada a la potencia según la posición.

Por ejemplo el número hexadecimal $145_{(16)}$, puede ser escrito en forma decimal, observando que el digito cinco está en la posición 0, el cuatro en la posición 1 y el uno en la posición 2 por tanto:

$$145_{(16)} = 1x16^2 + 4x16^1 + 5x16^0$$

$$145_{(16)} = 1x256 + 4x16 + 5x1$$

$$145_{(16)} = 256 + 64 + 5$$

$$145_{(16)} = 325_{(10)}$$

De esta forma observamos que el número ciento cuarenta y cinco hexadecimal ($145_{(16)}$) es el numero trescientos veinticinco decimal ($325_{(10)}$), implícitamente se puede observar que esta forma permite convertir un numero hexadecimal en un numero decimal.

Sistema BINARIO:

El sistema binario o sistema de base dos, es el sistema de numeración que entienden los sistemas digitales y de computación, está conformado por solo dos dígitos, cero (0) y uno (1), lo cual permite asociarlos en la electrónica y sistemas digitales a los valores de alto y bajo voltaje. Debido a que el sistema binario solo cuenta con dos dígitos para expresar valores o números, la representación de dichos números puede ser extensa.

Instituto Universitario de Tecnología para la Informática – Iutepi

Por ejemplo el numero $101000101_{(2)}$ puede ser escrito en forma decimal observando que existen nueves dígitos binarios, por tanto tendremos un total de nueve sumas para representar el número en decimal.

$$101000101_{(2)} = 1x2^8 + 0x2^7 + 1x2^6 + 0x2^5 + 0x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0$$

$$101000101_{(2)} = 1x256 + 0x128 + 1x64 + 0x32 + 0x16 + 0x8 + 1x4 + 0x2 + 1x1$$

$$101000101_{(2)} = 256 + 0 + 64 + 0 + 0 + 0 + 4 + 0 + 1$$

$$101000101_{(2)} = 325_{(10)}$$

De esta forma observamos que el número de nueve dígitos es el numero trescientos veinticinco decimal ($325_{(10)}$), que solo tiene tres dígitos para ser representados, implícitamente se puede observar que esta forma permite convertir un numero binarios en un numero decimal.

Conversión de números en sistema Decimal a otros sistemas de numeración:

Así como existen una conversión al sistema Decimal desde el sistema Octal, Hexadecimal y Binario, también es posible obtener el equivalente de un número en estos sistemas de numeración desde el sistema Decimal.

La forma de convertir un número a Octal, Hexadecimal y Binario, desde el decimal, es dividiendo el numero decimal por la base del sistema al cual se quiera convertir, la división se hace hasta que el número a dividir sea menor a la base por la cual se está dividiendo, y posteriormente se toman los residuos de cada sub-división en sentido inverso al cual se obtuvieron, de esta forma se construye el número.

Consideremos convertir el $325_{(10)}$ en octal. Se harán varias divisiones por el número 8 (base de sistema octal) hasta que el número sea menor a ocho.

325 / 8 = 40, residuo 5

40/8 = 5, residuo 0

5/8 => 5 es menor que 8, no se divide más, el 5 es residuo.

Si tomamos en sentido inverso (del ultimo al primero) los residuos de esta división, encontramos el número 5, que no pudo ser dividido, luego el 0, y por último el primer residuo 5, esto forma el 505, este número en octal es equivalente a 325 en decimal, lo cual se puede apreciar en la sección de Sistema Octal.

Convirtamos el $325_{(10)}$ en Hexadecimal, esto implica dividir 325 por 16, que es base hexadecimal, se hará la división hasta que numero sea menor a 16.

325/16 = 20, residuo 5

20/16 = 1, residuo 4

1/16 => 1 es menor a 16, no se divide más, el 1 es residuo.

Si tomamos los residuos en forma inversa, tenemos el número 145 en hexadecimal, el cual es equivalente a 325 en decimal, esto se puede observar en el ejemplo de la sección sistema Hexadecimal.

Convirtamos el $325_{(10)}$ en binario, ya que la base por la cual dividiremos el numero decimal es 2, la división suele ser bastante extensa a medida que el número a convertir sea alto.

325/2 = 162, residuo 1

162/2 = 81, residuo 0

81/2 = 40, residuo 1

40/2 = 20, residuo 0

20/2 = 10, residuo 0

10/2 = 5, residuo 0

5/2 = 2, residuo 1

2/2 = 1, residuo 0

1/2 => 1 es menor a 2, no se divide más, el 1 es residuo

Si tomamos los residuos, desde el último al primero, tenemos el número 101000101, el cual en binario representa al número 325 en decimal, esto puede observarse en el ejemplo de la sección sistema Binario.

Autoevaluación

Números Decimales

- ¿Cuál es el peso del dígito 6 en cada uno de los siguientes números decimales?

(a) 1386 (b) 54,692 (c) 671,920

- Hallar el valor de cada dígito en cada uno de los siguientes números decimales:

(a) 471 (b) 9.356 (c) 125.000

Números binarios

- Convertir a decimal los siguientes números binarios:

(a) 11 (b) 100 (c) 111 (d) 1000

- ¿Cuántos bits se requieren para representar los siguientes números decimales?

(a) 17 (b) 35 (c) 49 (d) 68

Conversión Decimal-Binario

- Convertir a binario cada uno de los números decimales indicados usando el método de la suma de pesos:

(a) 10 (b) 17 (c) 24 (d) 48

- Convertir a binario cada uno de los números decimales indicados usando el método de la división sucesiva por 2:

(a) 15 (b) 21 (c) 28 (d) 34

Operaciones aritméticas en base: Binaria

Las operaciones aritméticas están presentes en la cotidianidad, tanto en el ámbito laboral como en el educativo, en esta sección haremos especial énfasis en la operación en binario, debido a que son la base de cálculo en los sistemas digitales y de informática.

Al igual que en el sistema decimal, las operaciones de sumar restar multiplicar y dividir son posibles en el sistema binario. Las operaciones aritméticas lógicas se hacen del mismo modo que en sistema decimal.

Suma en Binario:

La suma en binario muy sencilla, esto se debe a que solo existe dos números que pueden ser sumados, el cero (0) y el uno (1).

Estos dígitos solo pueden sumarse en cuatro combinaciones posibles: 0 + 0 , 0 + 1 , 1 + 0 , 1 + 1 . Las sumas de 0 + 0 , 0 + 1 , 1 + 0 son evidentes, resultan igual que sumar en el sistema decimal, sin embargo la suma de 1 + 1, resulta en 0 más un digito(bit) que se debe colocar en la siguiente posición. Este digito se conoce como acarreo (bit de acarreo).

En la suma decimal el acarreo se presenta cuando el resultado de la suma es igual o superior a la base del sistema, en este caso cuando es mayor o igual a 10, es decir, la suma de 9 + 1 resulta en 10, esto es colocar el primer digito en cero y agregar una unidad por acarreo al siguiente digito o posición.

En el caso de sumar en binario 1 + 1, el resultado es igual a la base, así que el resultado es 0 en el primer digito (posición cero) y en el siguiente digito se coloca el acarreo que es 1.

+	0	1
0	0	1
1	0	0, y se tiene acarreo

Tabla 1.- Cuadro para sumar en Binario

Veamos un ejemplo de suma en binario partiendo del conocimiento de sumar en decimal, sumemos 17 más 4, que en decimal es 21.

$17_{(10)} = 10001_{(2)}$; $4_{(10)} = 100_{(2)}$, la suma viene dada por $10001_{(2)} + 100_{(2)}$, sumando los dígitos en la misma posición tenemos como resultado $10101_{(2)}$, realizando la conversión de este número a decimal se obtiene como resultado 21.

Resta en Binario:

La resta en binario se realiza igual que en decimal, al tener solo dos dígitos, tendremos al igual que en la suma cuatro combinaciones posibles; 0 – 0 , 1 – 0 , 1 – 1 y 0 – 1, esta última combinación implica la sustracción de un número menor de una mayor, por lo cual se deberá quitar una unidad prestada al digito siguiente, al igual que se hace en la resta en decimal. Esta unidad de préstamo debe devolverse agregando (sumando) esa unidad al siguiente digito.

-	0	1
0	0	1, se toma prestado
1	1	0

Tabla 2.- Cuadro para restar en Binario

Instituto Universitario de Tecnología para la Informática – Iutepi

Veamos el siguiente ejemplo que nos permitirá ilustrar la resta en binario. Restemos los números decimales 17 menos 4, cuyo resultado es 13.

$17_{(10)} = 10001_{(2)}$; $4_{(10)} = 100_{(2)}$, la resta viene dada por $10001_{(2)}$ - $100_{(2)}$, restando los bits por posición tenemos: 1 – 0 = 0 , 0 – 0 = 0, 0 – 1 = tomo prestado, el resultado es 1 y devuelvo el préstamo al siguiente digito/posición, 0 – 0 = 0 pero se tiene un acarreo por préstamo, el resultado es uno, el ultimo bit del número 17 resulta ser 0 debido a que fue el bit de préstamo, así que se termina restando 0 – 0 = 0; el resultado de la resta es : $01101_{(2)}$, este número convertido a decimal es 13.

Multiplicación en Binario:

La multiplicación en binario resulta ser muy sencilla, ya que los números a multiplicar son 0 y 1, los resultados de esa multiplicación solo pueden ser 0 o 1. El procedimiento de multiplicar en binario es igual que en el sistema decimal, se realiza por medio de la suma repetida.

X	0	1
0	0	0
1	0	1

Tabla 3.- Cuadro para Multiplicar en Binario

Veamos el siguiente ejemplo, multipliquemos 17 x 4, cuyo resultado en decimal es 68.

```
   10001
   x100
  -------
   00000
  00000
 +10001
  -------
 1000100
```

El resultado $1000100_{(2)}$ convertido a decimal equivale a 68.

Autoevaluación

Aritmética binaria

- Sumar los números binarios:

(a) 11 + 01 (b) 10 + 10 (c) 101 + 11 (d) 111 + 110

- Realizar las siguientes multiplicaciones binarias:

(a) 11 × 11 (b) 100 × 10 (c) 111 × 101 (d) 1001 × 110

Instituto Universitario de Tecnología para la Informática – Iutepi

Operaciones aritméticas en complemento a 1 y a 2

Aunque el proceso de restar dos números es algo sencillo en decimal, en el sistema binario puede generar ciertas confusiones debido al préstamo de una unidad o bit en el momento de restar 0 – 1, para simplificar o tratar de disminuir los errores al restar en binario haremos uso de dos técnicas, desarrollaremos el procedimiento de resta en binario como una suma haciendo uso del complemento a uno de un numero binario y de igual forma restaremos haciendo una suma con el complemento a dos de un numero binario.

Resta usando complemento a uno (C1):

El complemento a uno (C1) de un numero en binario, no es más que intercambiar los 1 por los 0 y viceversa, es decir, el numero $10001_{(2)}$ tiene su C1 igual a $01110_{(C1)}$, con esta técnica podemos convertir una resta en binario como una suma, para eso debemos convertir al sustraendo de la resta (número a restar) en su complemento a uno, y posteriormente realizar la suma, el resultado de esa suma tiene dos vertientes:

- Si el resultado de la suma con C1 genera un digito de acarreo, se deberá sumar ese uno de acarreo al resultado, esto genera el resultado final de la resta y adicionalmente indica que la resta es un número positivo.
- Si el resultado de la suma con C1 no genera digito de acarreo, esto significa que el resultado deberá convertirse en C1 para obtener el resultado final de la resta y adicionalmente indica que la resta es un número negativo.

Veamos el siguiente ejemplo de resta usando complemento a uno (C1), tomemos los números decimales 23 ($10111_{(2)}$) y 17 ($10001_{(2)}$) para realizar dos restas, la primera resta será 23 – 17 que es igual a 6 positivo, la siguiente resta será 17 – 23 que es igual a 6 negativo.

- Restemos $10111_{(2)} – 10001_{(2)}$, aplicando C1 al sustraendo tenemos $10111_{(2)} + 01110_{(2)}$ el resultado de esta suma es $100101_{(2)}$, cabe notar que el resultado es de seis bit, por lo tanto se generó un digito de acarreo, en esta situación ese digito debe sumarse a ese resultado para obtener el resultado final de la resta: $100101_{(2)} + 1_{(2)} = 00110_{(2)}$ este resultado ya es el definitivo de la resta y es equivalente a 6 decimal.
- Ahora restemos $10001_{(2)} – 10111_{(2)}$ aplicando C1 al sustraendo tenemos $10001_{(2)} + 01000_{(2)}$ el resultado de esta suma es $11001_{(2)}$, tiene la misma cantidad de dígitos, por lo que no se generó acarreo, esto significa que el resultado debe convertirse a C1 para obtener el resultado final de la resta, aplicando C1 a 11001 tenemos como resultado 00110, este número binario es 6 decimal, pero el resultado de la resta es negativo.

Resta usando Complemento a dos (C2):

El complemento a dos (C2) de un numero binario es el complemento a uno sumado con uno, C2 = C1 + 1. Sea el número 10001, el complemento a dos (C2) es: 01110 + 1 resultado en 01111 en C2.

La resta haciendo uso del C2 es similar a restar usando C1, convertimos el sustraendo a C2 y procedemos a realizar la suma de ambos números. Al igual que resta con C1 en C2 tenemos dos vertientes:

.- Si la resta con C2 genera un digito de acarreo, este digito se desprecia y los dígitos restantes son el resultado final de la resta, adicionalmente esto indica que el resultado de la resta es positivo.

.- Si la resta con C2 no genera digito de acarreo, el resultado debe ser convertido a C2, para obtener el resultado final de la resta, adicionalmente esto significa que el resultado de la resta es un número negativo.

Instituto Universitario de Tecnología para la Informática – Iutepi

Código BCD:

El código BCD (Binary Coded Decimal), es un código en el cual cada digito de un numero decimal, se representa por cuatro dígitos (bits) en binario, es decir, si tenemos un numero decimal de dos dígitos, cada digito será convertido a su respectivo numero binario de un total de 4 bits, por lo tanto, dos dígitos en decimal son ocho dígitos en BCD.

Veamos el siguiente ejemplo, el numero decimal 273, este número tiene tres dígitos, así que su representación en BCD será de 12 bits (dígitos).

$$273_{(10)} = 0010\ \ 0111\ \ 0011\ _{(BCD)}$$

Suma BCD:

La suma en BCD se realiza bit a bit; si en un grupo de cuatro bits el resultado es mayor que nueve decimal, ese grupo de bits debe corregirse sumando seis en BCD (0110). Tomemos la siguiente suma decimal como ejemplo:

23 + 57 = 80

23: 0010 0011

57: 0101 0111 +

Al sumar las primeras columnas tenemos 0011 + 0111, el resultado de esa suma binaria es 1010 que es 10 en decimal, al ser mayor que nueve, se debe corregir sumando seis (0110).

Al sumar 1010 + 0110 por corrección, se obtiene 1 0000, así que tenemos como resultado 0000 en esa columna y un bit de acarreo que deberá agregarse a la siguiente columna.

Sumando la segunda columna tenemos: 0010 + 0101 + 1 acá sumamos el bit de acarreo proveniente de la suma de la columna anterior, esto arroja como resultado: 1000, que es el número ocho en decimal, y al no ser mayor a nueve, no debe corregirse.

Como resultado final tenemos: $1000\ \ 0000_{(BCD)}$ que es: 80 en decimal.

NOTA: El sumar seis BCD (0110) se debe a que existe seis números que no pueden pertenecer al resultado de una suma BCD por ser mayores a nueve, los resultados posibles 10, 11, 12, 13, 14, 15, en una suma BCD, están prohibidos y cuando se obtienen se deben corregir.

Resta BCD:

Para realizar una resta en BCD, se debe utilizar la técnica de complemento a diez (C10), el complemento a diez no es más que la diferencia del número que está en el sustraendo con cien (100), esto hace que la resta BCD se convierta en una suma con dos vertientes posibles:

- Si el resultado final, arroja bit de acarreo luego de la suma de la última columna, significa que es un número positivo que ya está en BCD
- Si el resultado final, no arroja bit de acarreo luego de la suma de la última columna, significa que el resultado está en C10, debe aplicarse C10 para convertirlo a BCD y adicionalmente es una resta de valor negativo.

Instituto Universitario de Tecnología para la Informática – Iutepi

Veamos el siguiente ejemplo:

23 - 17 = 06

Aplicando C10 al sustraendo: 100 – 17 = 83, ahora convertimos la resta en una suma BCD de 23 + 83 , lo cual arroja como resultado 106, el digito uno ubicado en la tercera posición (centena) es el bit de acarreo, el cual indica que el resultado de la resta es 06 y positivo.

Código GRAY:

También conocido como código reflejado, es un sistema de numeración binario en el que dos números consecutivos difieren solo en uno de sus dígitos. El código Gray no es un código analítico.

Conversión Binario a Gray.

Para la conversión de un número binario a gray, usaremos números de tres dígitos binarios (0-7) para ejemplificar la conversión y enumeraremos los pasos a seguir para dicha conversión:

1.- Se copia el primer bit (de izquierda a derecha) del número binario como el primer bit en gray

2.- Se suma el primer bit en binario con el segundo, eso arroja el segundo bit en gray

3.- Se suma el segundo bit en binario con el tercer bit, eso arroja el tercer bit en gray.

4.- Si existe acarreo, este se desprecia.

NOTA: Si el numero en binario es mucho mayor (más de tres dígitos o bits) la suma se continua de forma sucesiva hasta sumar el ultimo bit.

Realicemos la conversión a código gray de los números en binario del 0 al 3:

Decimal	Binario	Procedimiento/ Conversión	Gray
0	000	0 + 0 = 0 , 0 + 0 = 0	000
1	001	0 + 0 = 0 , 0 + 1 = 1	001
2	010	0 + 1 = 1 , 1 + 0 = 1	011
3	011	0 + 1 = 1 , 1 + 1 = 0	010

Tabla 4.- Proceso de conversión número Binario a código Gray

Conversión Gray a Binario:

Para la conversión de un número gray a binario, usaremos tres dígitos binarios (0-7) para ejemplificar la conversión y enumeraremos los pasos a seguir para dicha conversión:

1.- Se copia el primer bit código gray igual

2.- Se suma el primer bit con el segundo bit de código gray, así se obtiene el segundo bit binario.

3.- El resultado de la suma anterior se debe sumar al tercer bit del código gray, así se obtiene el tercer bit binario.

Instituto Universitario de Tecnología para la Informática – Iutepi

4.- Se existe acarreo, este se desprecia.

NOTA: Si el código gray es de más de tres bits, la suma deberá continuarse hasta sumar el último bit.

Realicemos la conversión a código gray de los números en binario del 0 al 3:

Gray	Procedimiento/ Conversión	Binario	Decimal
000	0 + 0 = 0 , 0 + 0 = 0	000	0
001	0 + 0 = 0 , 0 + 1 = 1	001	1
011	0 + 1 = 1 , 1 + 1 = 0	010	2
010	0 + 1 = 1 , 1 + 0 = 1	011	3

Tabla 5.- Proceso de Conversión de código Gray a Numero Binario

Autoevaluación

Complemento a 1 de números binarios

- Determinar el complemento a 1 de los siguientes números binarios:

(a) 101 (b) 110 (c) 1010 (d) 11010111

- Determinar el complemento a 2 de cada uno de los siguientes números binarios:

(a) 00010110 (b) 11111100 (c) 10010001

Números hexadecimales

- Convertir a binario los siguientes números hexadecimales:

(a) 38_{16} (b) 59_{16} (c) $A14_{16}$ (d) $5C8_{16}$

- Convertir a decimal los siguientes números hexadecimales:

(a) 23_{16} (b) 92_{16} (c) $1A_{16}$ (d) $8D_{16}$

Números Octales

- Convertir a decimal los siguientes números octales:

12_8 (b) 27_8 (c) 56_8 (d) 64_8

- Convertir a binario los siguientes números octales:

(a) 13_8 (b) 57_8 (c) 101_8 (d) 321_8

Código Decimal Binario (BCD)

- Convertir los siguientes números decimales a BCD 8421:

(a) 10 (b) 13 (c) 18 (d) 21

Instituto Universitario de Tecnología para la Informática – Iutepi

- Convertir a BCD los siguientes números decimales:

(a) 104 (b) 128 (c) 132 (d) 150

- Sumar los siguientes números BCD:

(a) 1000 + 0110 (b) 0111 + 0101

Código Gray

- Convertir a código Gray los números binarios:

(a) 11011 (b) 1001010 (c) 1111011101110

- Convertir a binario los números en código Gray:

(a) 1010 (b) 00010 (c) 11000010001

Algebra de Boole:

El álgebra booleana fue inventada en el año 1854 por el matemático inglés George Boole. El álgebra de Boole es un método para simplificar los circuitos lógicos (o a veces llamados circuitos de conmutación lógica) en electrónica digital.

El álgebra de Boole es una estructura algebraica que consiste de un conjunto**,** de dos elementos, y dos operaciones binarias; tales que se cumplen los axiomas de clausura, conmutatividad, asociatividad, distributividad, identidad y complementariedad

Dualidad:

Dada una ecuación lógica, la ecuación dual se obtiene: reemplazando los operadores: + por · y · por +; reemplazando las constantes: 1 por 0 y 0 por 1; y dejando las variables sin cambios. En los siguientes postulados y teoremas se observaran los duales de cada ecuación planteada. El principio de dualidad indica que si una ecuación booleana es válida, también lo es su dual.

Postulados:

El álgebra de Boole es un sistema cerrado, en los postulados se hacen uso de los operadores lógicos suma (OR) y multiplicación (AND), y se considera a K como el conjunto formado por 1 y 0 (k = {1,0})

- Operaciones AND y OR: Para todo a,b perteneciente a k, tenemos que a.b pertenece a k y a+b pertenece a k
- Elemento Nulo: Existe el par 1 y 0 tal que a + 0 = a y a.1 = a
- Conmutatividad AND y OR: Para to a,b perteneciente a k, tenemos que a + b = b +a y que a.b = b.a
- Asociatividad AND y OR: Para todo a,b,c perteneciente a k tenemos:

  $$a + (b + c) = (a + b) + c$$
  $$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$
- Distributividad AND y OR: Para todo a,b,c perteneciente a k tenemos:

  $$a + (b \cdot c) = (a+b) \cdot (a + c)$$
  $$a \cdot (b + c) = (a \cdot b) + (b.c)$$
- Complemento o Inverso: Para todo a perteneciente a k, existe un único elemento $\bar{a}$, complemento de a en k, por lo tanto $a + \bar{a} = 1$; $a \cdot \bar{a} = 0$

Teoremas:

- Idempotencia: $a + a = a$; $a \cdot a = a$
- Elemento Nulo: $a + 1 = 1$; $a \cdot 0 = 0$; $a \cdot 1 = a$
- Involución: $\bar{\bar{a}} = a$ (complemento del complemento)
- Absorción: $a + a.b = a$; $a.(a + b) = a$
- Absorción del complemento:

  $$a + \bar{a}.b = a + b$$
  $$a \cdot (\bar{a} + b) = a \cdot b$$
- Fusion:

  $$a.\bar{b} + a.b = a$$

  $$(a + b) \cdot (a + \bar{b}) = a$$

Instituto Universitario de Tecnología para la Informática – Iutepi

- DE MORGAN:

$$\overline{a + b} = \bar{a} \cdot \bar{b}$$

$$\overline{a \cdot b} = \bar{a} + \bar{b}$$

- Consenso:

$$a.b + \bar{a}.c + b.c = a.b + \bar{a}.c$$
$$(a + b) \cdot (\bar{a} + c) \cdot (b + c) = (a + b) \cdot (\bar{a} + c)$$

Tabla de Verdad:

Una tabla de verdad es la representación lógica en unos y ceros de un sistema o función en binario. La tabla de verdad es única para ese sistema o función. Sea $f(A) = X_0A + X_1\bar{A}$, donde X_0 y X_1 pertenecen al conjunto k = {1,0}, la correspondiente tabla de verdad de la función f(A) es la siguiente:

X_0	X_1	f(A)
0	0	0
0	1	$\bar{A}$
1	0	A
1	1	$A + \bar{A} = 1$

Tabla 6.- Tabla de Verdad de la función f(A)

Compuertas Lógicas:

A continuación trataremos el funcionamiento lógico de las puertas lógicas, así como la respectiva tabla de verdad de cada una de ellas.

El Inversor: Negado de un bit

El inversor (circuito NOT) realiza la operación denominada inversión o complementación. El inversor cambia un nivel lógico al nivel opuesto. En términos de bits, cambia un 1 por un 0, y un 0 por 1.

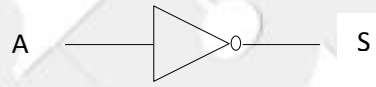


Figura 1.- Símbolo Lógico compuerta NOT.

A	S ($\bar{A}$)
0	1
1	0

Tabla 7.- Tabla de Verdad para Compuerta NOT

Compuerta Producto Lógico: AND

La puerta AND es una de las puertas básicas con la que se construyen todas las funciones lógicas. Una puerta AND puede tener dos o más entradas y realiza la operación que se conoce como multiplicación lógica.

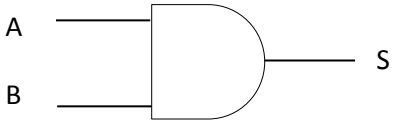


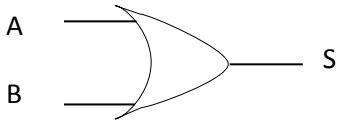
Figura 2.- Símbolo Lógico compuerta AND.

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

Tabla 8.- Tabla de Verdad para Compuerta AND

Compuerta Suma Lógica: OR

La puerta OR es otra de las puertas básicas con las que se construyen todas las funciones lógicas. Una puerta OR puede tener dos o más entradas y realiza la operación que se conoce como suma lógica.

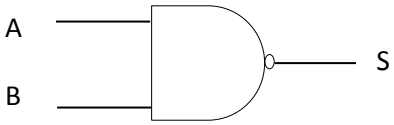


A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Figura 3.- Símbolo Lógico Compuerta OR. Tabla 9.- Tabla de Verdad para Compuerta OR

Compuerta Negada AND: NAND

El término NAND es una contracción de NOT–AND, e implica una función AND con la salida complementada (negada). La puerta NAND es un elemento lógico popular, debido a que se puede utilizar como una puerta universal, es decir, las puertas NAND se pueden combinar para implementar las operaciones de las puertas AND, OR y NOT.

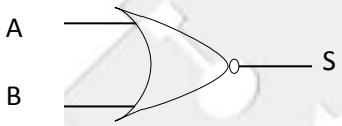


A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Figura 4.- Símbolo Lógico Compuerta NAND. Tabla 10.- Tabla de Verdad para Compuerta NAND

Compuerta Negada OR: NOR

El término NOR es una contracción de NOT–OR e implica una función OR con la salida invertida (complementada). La puerta NOR, al igual que la puerta NAND, es un útil elemento lógico porque también se puede emplear como una puerta universal; es decir, las puertas NOR se pueden usar en combinación para implementar las operaciones AND, OR y NOT.

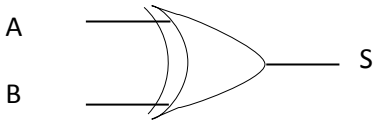


A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

Figura 5.- Símbolo Lógico Compuerta NOR. Tabla 11.- Tabla de Verdad para Compuerta NOR

Compuerta OR Exclusiva: XOR

Debido a su importancia este circuito se considera como una puerta lógica con su propio símbolo distintivo, realmente es una combinación de dos puertas AND, una puerta OR y dos inversores.

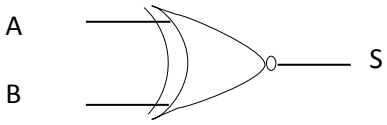


A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Figura 6.- Símbolo Lógico Compuerta XOR. Tabla 12.- Tabla de Verdad para Compuerta XOR

Compuerta NOR Exclusiva: XNOR

La función NOR-exclusiva es el complemento de la función OR-exclusiva. La función NOR-exclusiva puede implementarse invirtiendo la salida de un circuito OR-exclusiva.



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

Figura 7.- Símbolo Lógico Compuerta XNOR. Tabla 13.- Tabla de Verdad para Compuerta XNOR

Autoevaluación

Compuerta AND

- Determine la salida S de la compuerta AND de tres entradas, para las variables A, B y C de la tabla

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

Compuerta OR

- Determine la salida S para la compuerta OR de 4 entradas, para las variables A, B, C y D de tabla

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

Compuerta NAND

- Repita el ejercicio 21 para una compuerta NAND de 4 entradas

Compuerta NOR

- Repita el ejercicio 20 para una compuerta NOR de 3 entradas

Operaciones y Expresiones Booleanas

- Utilizando la notación booleana, escribir una expresión que sea 1 siempre que una o más de sus variables (A, B, C y D) sean 1.
- Escribir una expresión que sea 1 cuando una o más variables (A, B y C) son 0.
- Hallar los valores de las variables que hacen que cada término producto sea 1 y que cada suma sea 0.

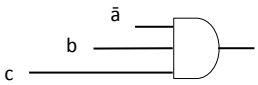
  (a) AB (b) A + B (c) ABC

Instituto Universitario de Tecnología para la Informática – Iutepi

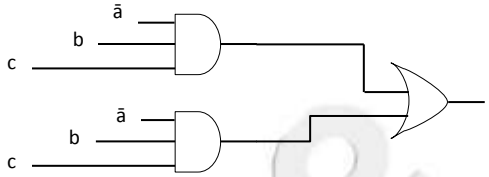
Análisis de Circuitos Lógicos con Algebra de Boole

- Escribir la expresión booleana para cada puerta lógica de la Figuras.

(a)



(b)



- Dibujar el circuito lógico representado por cada una de las siguientes expresiones.

  (a) A + B + C (b) AB + C

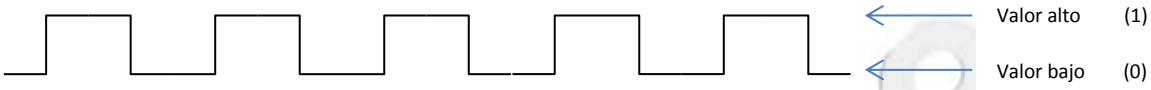
Expresiones Booleanas y Tabla de Verdad

- Para la tabla de verdad, obtener una expresión suma de productos estándar y un producto de sumas estándar.

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

Funcionamiento con trenes de impulso:

Un tren de impulsos es una señal que tiene niveles altos y bajos de tensión (voltaje) en el tiempo. Un impulso posee dos flancos, uno ascendente (subida) y otro descendente (bajada). Los trenes de impulso son periódicos y no periódicos. Un tren de impulso periódico puede ser utilizado para sincronizar un sistema digital en el tiempo, este tipo de tren de impulso se conoce como reloj.



Tren de impulsos no periódico:

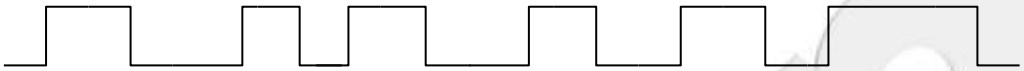


Figura 8.- Imagen de tren de Impulsos periódico y no periódico

Compuertas lógicas y tren de impulsos:

En esta sección aplicaremos a las entadas de las compuertas lógicas, valores altos y bajos de tensión, haciendo analogía a un tren de impulsos aplicado y observando el comportamiento de la salida de dicha compuerta la cual está regida por la tabla de verdad.

Funcionamiento de compuerta AND con trenes de impulso

En la mayoría de las aplicaciones, las entradas a una puerta no son niveles estacionarios sino tensiones que cambian frecuentemente entre los niveles lógicos ALTO y BAJO. Ahora vamos a ver el funcionamiento de las puertas AND con entradas que son señales digitales (tren de impulsos), teniendo en mente que una puerta AND obedece a su tabla de verdad, independientemente de que sus entradas sean niveles constantes o señales que varíen de un nivel a otro.

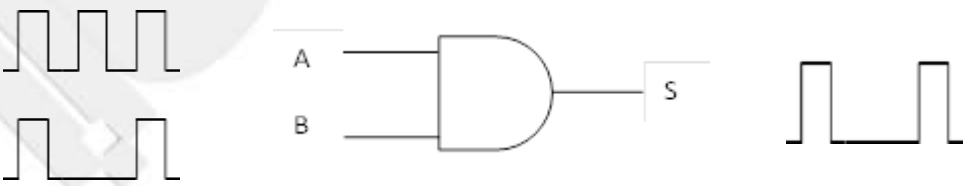


Figura 9.- Tren de Impulsos aplicado a Compuerta AND

En la imagen observamos dos entradas diferentes a la compuerta AND, por A entra una señal cuadrada en forma de tren de impulso, por B se aplica una señal de entrada cuadrada diferente a la señal entrante por A, claramente se puede apreciar que la señal entrante por A tiene el comportamiento lógico 10101 y en ese mismo periodo de tiempo

la señal entrante por B tiene los valores lógicos 10001, estos no permite obtener la salida S conociendo la tabla de verdad de la compuerta AND:

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

Tabla 14.- Tabla de Verdad de tren de impulso aplicado a una Compuerta AND

Funcionamiento de compuerta OR con trenes de impulso

Ahora vamos a ver el funcionamiento de una puerta OR con trenes de impulsos como entradas, teniendo en mente su modo de operación lógico. De nuevo, lo importante en el análisis del funcionamiento de la puerta con trenes de impulsos en las entradas es la relación de tiempos de todas las señales implicadas.

Figura 10.- Tren de Impulsos aplicado a Compuerta OR

En la imagen se hizo uso de las mismas entrada para la compuerta AND anteriormente descrita, la entrada A tiene valores lógicos 10101 y la entrada B 10001, esto hace que la salida S lógica sea 10101, obedeciendo a la tabla de verdad de la compuerta OR.

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

Tabla 15.- Tabla de Verdad de tren de impulso aplicado a una Compuerta OR

Instituto Universitario de Tecnología para la Informática – Iutepi

Familia Lógica TTL:

TTL, cuyo significado es Transistor Transistor Logic (lógica Transistor Transistor), es una tecnología de construcción de circuitos electrónicos digitales. En los componentes fabricados con tecnología TTL los elementos de entrada y salida del dispositivo son transistores bipolares (BJT).

FAMILIA TTL ESTÁNDAR

La familia lógica TTL-Estándar es una familia "saturante" de TTL caracterizada fundamentalmente por su rapidez. Es saturante porque la mayor parte de los transistores que la forman trabajan en corte-saturación (switch). Estos transistores conducen tan pronto como la corriente de base sea suficiente para hacer que la intensidad de colector sea la de saturación.

FAMILIA TTL-SCHOTTKY

Uno de los principales problemas que existen en la familia TTL-Estándar es la pérdida de velocidad en la conmutación (cambio de estado), debido a que la mayoría de los transistores trabaja en corte-saturación y es difícil evacuar el exceso de carga almacenada en la región de base durante la saturación. Este problema tiene solución con la aparición de la familia TTL-Schottky. Se trata de evitar que los transistores alcancen el estado de saturación. De esta manera se reduce el exceso de carga en la región de base, de forma que se tardará menos en evacuarla cuando el transistor intente pasar al corte, lo que se traduce en un aumento considerable de la velocidad.

FAMILIA TTL-LS

Con la familia TTL-S se obtiene un gran aumento de velocidad de conmutación con respecto a la TTL-Estándar, pero también se había aumentado la corriente que circulaba por la puerta y, por lo tanto, su consumo de potencia. A partir de la TTL-S se obtuvo la familia TTL-LS; TTL Schottky de baja potencia (TTL-Low Power Schottky). Con esta familia se obtiene un consumo menor de potencia, pero se reduce la velocidad de conmutación. A pesar de esto, la velocidad que se ha obtenido es muy parecida a la de la TTL-Estándar pero el consumo de potencia se ha reducido. La familia TTL-LS, como descendiente de la familia TTL-S, sigue utilizando el diodo Schottky.

FAMILIA TTL-ALS

La familia "Schottky de baja potencia avanzada" (Advanced Low-power Schottky, ALS) es una de las más avanzadas de la familia TTL. En ésta se aumenta dos veces la eficiencia de conducción y se ofrece una reducción de potencia en comparación con la familia TTL-LS. Con esta familia se mejora el producto Potencia-Velocidad. El producto Potencia-Velocidad (powerspeed) es un sistema de medida (cuya unidad es el picojulio) utilizado en los circuitos donde la velocidad y la potencia son factores muy importantes. En los circuitos digitales vistos hasta ahora, hemos observado que siempre se intenta reducir el consumo del circuito y aumentar la velocidad de conmutación (con lo cual la información será transmitida más rápidamente).

FAMILIA FAST

La familia FAST, donde FAST proviene de TTL Schottky Avanzada de FAIRCHILD (FAIRCHILD Advanced Schottky TTL) es el último paso en TTL. Fue creada en la década de los 80 y debido a su alta velocidad de conmutación puede trabajar en áreas hasta ahora reservadas para la lógica "ECL IOK" utilizando los diseños TTL básicos y una única alimentación de 5V. La alta impedancia de entrada de la familia FAST permite la interconexión directa con los circuitos de las familias TTL-LS, TTL-ALS en un mismo sistema. Los circuitos FAST reducen la potencia que disipan con respecto a la familia TTL-S ya sea trabajando a nivel alto o bajo mejorando además el producto Potencia-Velocidad.

La mayoría de los sistemas diseñados con circuitos TTL-S, pueden funcionar reemplazando estos circuitos por sus equivalentes de la familia FAST.

Características de las Familias TTL

La familia de circuitos integrados TTL tiene las siguientes características:

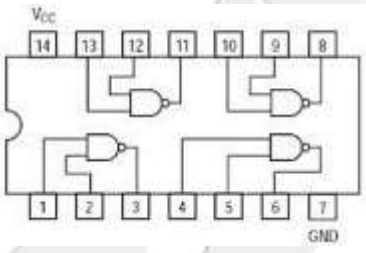
- La tensión o voltaje de alimentación es de + 5 Voltios, con Vmin = 4.75 Voltios y Vmax = 5.25 Voltios.
- Niveles lógicos: entre 0,2 V y 0,8 V para el nivel bajo (L) y entre 2 V y 5 V para el nivel alto (H), ya que estos chips son activados por altos y bajos, o también llamados 0 y 1.
- Su fabricación es con transistores bipolares.
- La velocidad de transmisión entre los estados lógicos es su mejor ventaja, ciertamente esta característica le hacer aumentar su consumo.
- Su compuerta básica es la NAND (7400)
- Código identificador: el 74 para los comerciales y el 54 para los de diseño militar

Designaciones de Componente

Los circuitos integrados de la familia lógica TTL, tienen una designación de pieza formada por un número de cuatro a cinco dígitos. Con la incorporación de otros tipos de construcciones de dispositivos, se añadieron letras al centro de la numeración, para recordar al usuario que no se está utilizando el chip básico TTL. Los números de dispositivos que empiezan con un prefijo corresponden a su serie, seguida por otro número que identifica el chip individual.

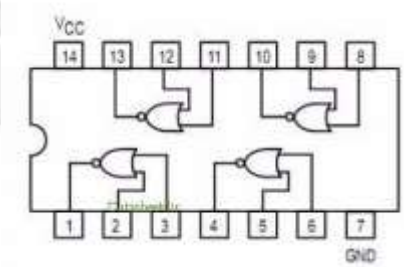
Componentes TTL Serie 74XX

Por serie 74XX se conoce a los circuitos integrados digitales fabricados en tecnología TTL, dentro del campo de la electrónica digital. A continuación observamos algunos integrados de la familia TTL 74XX:



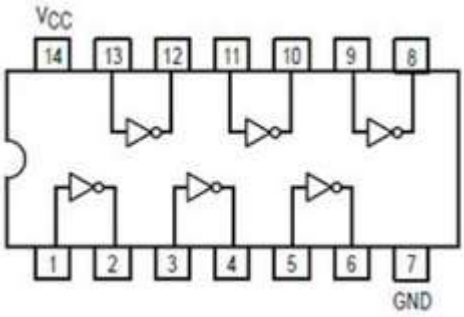
NAND 7400

Compuerta NAND (Fabricante: Texas instruments)



NOR 7402

Compuerta NOR (Fabricante: Hitachi Ltd.)



NOT 7404

Compuerta NOT (Fabricante: Texas Instruments)

Figura 11. – Diagrama de construcción interno e Integrado para Compuertas NAND-NOR-NOT

Convenios Lógicos de Voltaje:

En electrónica y los circuitos digitales es común referiste a las entradas y salidas que tienen las compuertas lógicas como altos o bajos de tensión. A un valor alto se le asocia un "1" y a un valor bajo un "0". Según la tecnología con la cual está construida la compuerta podemos tener diferentes valores de tensión que se pueden tomar como un 1 o 0 lógico; el siguiente cuadro indica los valores alto y bajo de tensión con los que algunos fabricantes asocian los valores lógicos 1 y 0.

Nivel de Tensión/Lógico	Tecnología TTL	Tecnología CMOS	Tecnología HC
Bajo (0)	0v – 0,8v	0v – 1,5v	0v – 1v
Alto (1)	2v – 5v	3,5v – 5v	3,5v – 5v

Tabla 16.- Convenios de voltaje lógico, por tecnología.

Se puede observar que los valores de cero lógicos en TTL corresponden a valores de tensión menor a 1v, mientras que se puede considerar un uno lógico cuando la tensión está por encima de los 2v.

Teorema de las leyes de DeMorgan

DeMorgan, matemático que conoció a Boole, propuso dos teoremas que constituyen una parte muy importante del álgebra de Boole. En términos prácticos, los teoremas de DeMorgan proporcionan una verificación matemática de la equivalencia entre las puertas NAND y negativa-OR, y las puertas NOR y negativa-AND.

El primer teorema de DeMorgan se enuncia de la siguiente forma: El complemento de un producto de variables es igual a la suma de los complementos de las variables. O dicho de otra manera El complemento de dos o más variables a las que se aplica la operación AND es equivalente a aplicar la operación OR a los complementos de cada variable. La fórmula para expresar este teorema para dos variables es:

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

Instituto Universitario de Tecnología para la Informática – Iutepi

El segundo teorema de DeMorgan se enuncia como sigue:

El complemento de una suma de variables es igual al producto de los complementos de las variables. O dicho de otra manera, el complemento de dos o más variables a las que se aplica la operación OR es equivalente a aplicar la operación AND a los complementos de cada variable. La fórmula para expresar este teorema es:

$$\overline{a + b} = \bar{a} \cdot \bar{b}$$

Expresiones Booleanas – Forma SOP y POS

Las expresiones del algebra de Boole contenidas en postulados y teoremas, así como las leyes de De Morgan, permiten construir funciones booleanas, vamos a definir funciones booleanas, que son exactamente iguales a las funciones matemáticas a las que estamos habituados pero con la particularidad de que las variables son booleanas y que los valores devueltos por la función también son booleanos, es decir, una función booleana sólo puede tomar los valores cero y uno.

Por ejemplo, desarrollemos la función booleana de una variable F(a) = ā, es claro que los valores que esta función acepta son a = 0 y a = 1, si evaluamos dicha función obtendremos la tabla de verdad de la misma:

a	F(a)	F
0	a = 0 , F(0) = $\bar{0}$ = 1	1
1	a = 1 , F(1) = $\bar{1}$ = 0	0

Tabla 17.- Tabla de Verdad de la función F(a).

Ahora desarrollemos una función que contenga dos variables booleanas, por ejemplo F(a,b) = ā + b, al igual que la función anterior de una variable, esta función solo aceptara valores de a y b entre 0 y 1.

Evaluemos la función para determinar la tabla de verdad:

a	b	F(a,b)	F
0	0	F(0,0) = $\bar{0}$ + 0 = 1 + 0 = 1	1
0	1	F(0,1) = $\bar{0}$ + 1 = 1 + 1 = 1	1
1	0	F(1,0) = $\bar{1}$ + 0 = 0 + 0 = 0	0
1	1	F(1,1) = $\bar{1}$ + 1 = 0 + 1 = 1	1

Tabla 18.- Tabla de Verdad de la función F(a,b)

Acá se debe acotar que para una función de una variable, se obtienen dos valores para función, en el caso de una función de dos variables, la función tienen como resultado cuatro valores, esto define de cierta forma el tamaño de la tabla de verdad de una función booleana en base a la cantidad de variables que estas posea, si la función tienen n variables, la tabla de verdad tendrá 2^n valores o resultados. Esto significa que para una función de tres variables se tendrán ocho valores de la función ($2^3 = 8$).

Instituto Universitario de Tecnología para la Informática – Iutepi

Forma SOP o Suma de Productos:

La suma de productos (Sum of Products) son expresiones booleanas donde las funciones están expresadas como el producto entre las variables sumadas entre sí, estás expresiones están escritas en forma canónica, significa que cada término de la función, posee todas las variables de la misma negadas o no negadas.

Observemos como ejemplo la siguiente función en forma SOP: F(a,b) = ā.b + a.b, esta función al ser booleana solo podrá aceptar en sus variables valores entre cero y uno, así que procederemos a evaluar la función para así obtener su tabla de verdad:

a	b	F(a,b)	F
0	0	F(0,0) = $\bar{0}$.1 + 0.0 = 0	0
0	1	F(0,1) = $\bar{0}$.1 + 0.1 = 1	1
1	0	F(1,0) = $\bar{1}$.0 + 1.0 = 0	0
1	1	F(1,1) = $\bar{1}$.1 + 1.1 = 1	1

Tabla 19.- Tabla de Verdad de la función F(a,b) forma SOP

La forma SOP también es conocida como “unos de la función” o minterms, ya que los resultados donde la función tiene valor uno, permiten generar la expresión booleana en su forma canónica.

Forma POS o Productos de Suma:

El productos de sumas (Product of Sums) son expresiones booleanas donde las están expresadas como la multiplicación de términos que están sumándose entre sí, estas expresiones también están escritas en su forma canónica, incluyendo cada termino a todas la variables de la funciones negadas o no negadas.

Veamos la siguiente función escrita en forma POS como ejemplo: F(a,b) = (ā+b).(a+b), esta función es booleana así que solo aceptara valores para a y b entre cero y uno. Evaluemos la función para determinar la tabla de verdad:

a	b	F(a,b)	F
0	0	F(0,0) = ($\bar{0}$+0).(0+0) = 0	0
0	1	F(0,1) = ($\bar{0}$+1).(0+1) = 1	1
1	0	F(1,0) = ($\bar{1}$+0).(1+0) = 0	0
1	1	F(1,1) = ($\bar{1}$+1).(1+1) = 1	1

Tabla 20.- tabla de Verdad de la funcio F(a,b) forma POS

La forma POS también es conocida como “los ceros de la función” o maxterms, esto se debe a que los resultados donde la función tiene valor cero, permiten generar la función booleana en su forma canónica.

Instituto Universitario de Tecnología para la Informática – Iutepi

Simplificación de funciones Booleanas:

Cuando se diseñan circuitos digitales, se utilizan funciones booleanas para describirlos, y antes de implementarlos como componentes electrónicos (compuertas lógicas) tenemos que simplificar al máximo la función que lo describe. No basta con realizar un circuito, sino que hay que hacerlo con el menor número posible de componentes electrónicos. Y esto es lo que conseguimos si trabajamos con funciones simplificadas.

Existen dos formas que permiten reducir o simplificar una función a una expresión de menor cantidad de términos, las funciones booleanas en su forma canónica no son las expresiones más simplificadas que se pueden obtener.

Como primer método tenemos el analítico que hace uso de los postulados y teoremas de del algebra de Boole para simplificar funciones y el método de simplificación por mapas de Karnaugh, el cual toma la tabla de verdad de la función para reducir la expresión.

Veamos el siguiente ejemplo de una función en forma SOP canónica:

$$F(a,b,c) = \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + ab\bar{c}$$

Para simplificar esta función a una expresión de menores términos haremos uso del algebra de Boole, tomaremos el primer y tercer término para factor común de $\bar{a}c$ y el segundo y cuarto termino para factor común de $b\bar{c}$:

$$\bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + ab\bar{c}$$

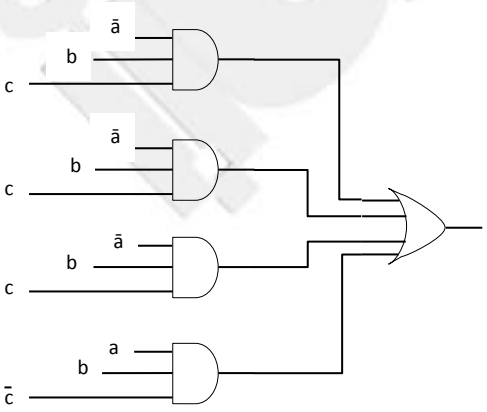
$$\bar{a}c.(\bar{b}+b) + b\bar{c}.(\bar{a}+a)$$

$$\bar{a}c.1 + b\bar{c}.1$$

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

De esta forma, una función de cuatro términos; queda reducida o simplificada a otra de solo dos términos no canónica, ambas funciones son equivalentes, por lo que tienen la misma tabla de verdad, esta simplificación permite una reducción de compuertas y cableado (pistas) al momento de implementar el circuito combinacional.

Función F implementada sin reducir
(Circuito Combinacional)



Función F simplificada e implementada
(Circuito Combinacional)

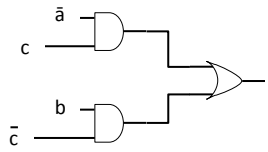


Figura 12.- Circuito SOP implementado y simplificado

Mapas de Karnaugh

Un mapa de Karnaugh proporciona un método sistemático de simplificación de expresiones booleanas y, si se aplica adecuadamente, genera las expresiones suma de productos y producto de sumas más simples posibles, conocidas como expresiones mínimas. La razón por la que se construye un mapa de Karnaugh es debido a la forma práctica y rápida en la cual se puede simplificar una función booleana, ya hemos visto que la simplificación reduce considerablemente la implementación de un circuito lógico con compuertas para un sistema digital.

Un mapa de Karnaugh es similar a una tabla de verdad, ya que muestra todos los valores posibles de las variables de entrada y la salida resultante para cada valor. En lugar de organizar en filas y columnas como una tabla de verdad, el mapa de Karnaugh es una matriz de celdas en la que cada celda representa un valor binario de las variables de entrada. Las celdas se organizan de manera que la simplificación de una determinada expresión consiste en agrupar adecuadamente las celdas. Los mapas de Karnaugh se pueden utilizar para expresiones de dos, tres, cuatro y cinco variables.

El método con mapas de Karnaugh es un método gráfico. Se usan unas tablas llamadas tablas o diagramas de Karnaugh. Dichas tablas tienen una casilla por cada combinación de variables de la función, de forma que para 3 variables (A,B,C) tendremos $2^3 = 8$ casillas, para cuatro variables(A,B,C,D) tendremos $2^4 = 16$ casillas.

Mapa de Karnaugh de tres variables

El mapa de Karnaugh de tres variables es una matriz de ocho celdas. En este caso, A, B y C se emplean para denominar a las variables, aunque podían haberse usado cualesquiera otras letras. Los valores binarios de A se encuentran en el lado izquierdo y los valores de B y C se colocan en la parte superior. El valor de una determinada celda es el valor binario de A, en la parte izquierda de la misma fila combinado con el valor de B y C en la parte superior de la misma columna.

A \ BC	00	01	11	10
0				
1				

Tabla 21.- Mapa de Karnaugh de tres variables

Mapa de Karnaugh de cuatro variables

El mapa de Karnaugh de cuatro variables es una matriz de dieciséis celdas. Los valores binarios de A y B se encuentran en el lado izquierdo y los valores de C y D se colocan en la parte superior. El valor de una determinada celda es el valor binario de A y B, en la parte izquierda de la misma fila combinado con los valores binarios de C y D en la parte superior de la misma columna

AB \ CD	00	01	11	10
00				
01				
11				
10				

Tabla 22.- Mapa de Karnaugh de cuatro variables

Mapa de Karnaugh de una suma de productos estándar (SOP)

Por cada término de la expresión suma de productos, se coloca un 1 en el mapa de Kanaugh en la celda correspondiente al valor del producto. Se coloca un 1 en la celda correspondiente al valor de un término producto. Cuando una expresión suma de productos se ha reflejado por completo en el mapa de Karnaugh, en dicho mapa habrá tantos 1 como términos producto tenga la suma de productos estándar. Las celdas que no contienen un 1 son aquellas para las que la expresión es igual a 0. Normalmente, cuando se trabaja con una expresión suma de productos, los 0 no se incluyen en el mapa. Veamos la siguiente tabla de verdad representada en un mapa de Karnaugh de tres variables usando la forma SOP:

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

A \ BC	00	01	11	10
0		1		1
1	1			1

Tabla 23.- Tabla de Verdad y Mapa de Karnaugh para función de tres variables.

El proceso que genera una expresión que contiene el menor número posible de términos con el mínimo número de variables posibles se denomina minimización. Después de haber obtenido el mapa de Karnaugh de una suma de productos, la expresión suma de productos mínima se obtiene agrupando los 1 y determinando la expresión suma de productos mínima a partir del mapa.

Podemos agrupar los unos del mapa de Karnaugh de acuerdo con las reglas siguientes, rodeando las celdas adyacentes que contengan unos. La finalidad es maximizar el tamaño de los grupos y minimizar el número de estos grupos.

1. Un grupo tiene que contener 1, 2, 4, 8 ó 16 celdas, valores que se corresponden con las potencias de2. En el caso de un mapa de Karnaugh de 3 variables, el grupo máximo puede contener $2^3 = 8$ celdas.

2. Cada celda de un grupo tiene que ser adyacente a una o más celdas del mismo grupo, pero no todas las celdas del grupo tienen que ser adyacentes entre sí.

3. Incluir siempre en cada grupo el mayor número posible de 1 de acuerdo a la regla número 1.

4. Cada 1 del mapa tiene que estar incluido en al menos un grupo. Los 1 que ya pertenezcan a un grupo pueden estar incluidos en otro, siempre que los grupos que se solapen contengan 1 no comunes.

A \ BC	00	01	11	10
0		1		1
1	1			1

Tabla 24.- Grupos de unos en el MK

Cuando todos los 1 que representan los términos productos estándar de una expresión se han trasladado al mapa y se han agrupado adecuadamente, comienza el proceso de obtención de la suma de productos mínima. Para encontrar los términos mínimos y la expresión suma de productos mínima se aplican las siguientes reglas:

1. Agrupar las celdas que contienen 1. Cada grupo de celdas que contiene 1 da lugar a un término producto compuesto por todas las variables que aparecen en el grupo en sólo una forma (no complementada o complementada). Las variables que aparecen complementadas y sin complementar dentro del mismo grupo se eliminan. A éstas se les denomina variables contradictorias.

2. Determinar la operación producto mínima para cada grupo.

(a) Para un mapa de 3 variables:

- Un grupo formado por 1 celda da lugar a un término producto de 3 variables.
- Un grupo formado por 2 celdas da lugar a un término producto de 2 variables.
- Un grupo formado por 4 celdas da lugar a un término de 1 variable.
- Un grupo formado por 8 celdas indica que la expresión vale 1.

(b) Para un mapa de 4 variables:

- Un grupo formado por 1 celda da lugar a un término producto de 4 variables.
- Un grupo formado por 2 celdas da lugar a un término producto de 3 variables.
- Un grupo formado por 4 celdas da lugar a un término producto de 2 variables.
- Un grupo formado por 8 celdas da lugar a un término de 1 variable.
- Un grupo formado por 16 celdas indica que la expresión vale 1.

3. Cuando se han obtenido todos los términos producto mínimos a partir del mapa de Karnaugh, se suman para obtener la expresión suma de productos mínima.

Aplicando estos pasos a nuestro ejemplo obtenemos la siguiente función simplificada en forma SOP:

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

El grupo que forma un único 1 en la posición 001 genera la expresión de tres variables $\bar{A}\bar{B}C$, el grupo que se forma con una pareja de 1 entre 100 y 110 genera una expresión de dos variables $A\bar{C}$, y por último el grupo que se forma entre 010 y 110 genera la expresión de dos variables BC.

Mapa de Karnaugh de un Producto de sumas estándar (POS)

Para un producto de sumas en forma estándar, se introduce un 0 en el mapa de Karnaugh por cada término suma de la expresión. Cada 0 se sitúa en la celda correspondiente al valor de un término suma. Cuando un producto de sumas se ha trasladado por completo al mapa, habrá tantos 0 en el mapa de Karnaugh, como términos suma en la expresión del producto de sumas estándar. Las celdas que no contienen un 0 son aquellas para las que la expresión vale 1.

Generalmente, cuando se trabaja con productos de sumas, los 1 no se escriben. El proceso de minimización de un producto de sumas es básicamente el mismo que para una expresión suma de productos, excepto que ahora hay que agrupar los ceros para generar el mínimo número de términos suma, en lugar de los 1 para obtener el número

mínimo de términos producto. Las reglas para agrupar los 0 son las mismas que para agrupar los 1. Colocando los 0 en nuestro mapa de tres variables del ejemplo de tabla de verdad tenemos, obtenemos los siguientes grupos:

A \ BC	00	01	11	10
0	0		0	
1		0	0	

Tabla 25.- Grupos de ceros en el MK

Aplicando los pasos de simplificación a nuestro ejemplo obtenemos la siguiente función reducida en forma POS:

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

El grupo que forma el único 0 en la posición 000, genera la expresión de tres variables A+B+C, el grupo que se forma con una pareja de 0 en la posición 101 y 111 genera la expresión de dos variables $\bar{A}+\bar{C}$, y por último el grupo de dos 0 que se forma en la posición 011 y 111 genera la expresión de dos variables $\bar{B}+\bar{C}$.

Circuitos Combinacionales:

Después de introducir y trabajar con el Algebra de Boole, vamos a volver a los circuitos digitales. Recordemos que son circuitos electrónicos que trabajan con números, y que con la tecnología con la que están realizados, estos números están representados en binario.

Si tomamos un circuito genérico y miramos en su interior, podemos ver que está constituido por otros circuitos más simples, interconectados entre sí. Estos subcircuitos se pueden clasificar en dos tipos:

- Circuitos combinacionales
- Circuitos secuenciales

Las expresiones suma de productos se implementan con una puerta AND para cada término producto y una puerta OR para sumar todos los términos producto. Esta implementación de la suma de productos se llama lógica AND-OR y es la forma básica de un circuito combinado para realizar las funciones estándar booleanas. En esta sección, se estudian las combinaciones (circuitos) AND-OR y AND-OR-NOT.

Circuito AND-OR

Consideremos la siguiente función Booleana F(A,B,C,D) = AB + CD, dicha función de cuatro variables tendrá 16 (2^4) resultados para la función F, estos resultados se pueden plasmar en una tabla de verdad, sin embargo el circuito que resulta de implementar dicha función (sin simplificar) es una combinación de una compuerta AND de dos entradas para el productos AB, una segunda compuerta AND de dos entradas para el producto CD y una compuerta OR que permita sumar los productos AB y CD; el circuito resulta de esta forma:

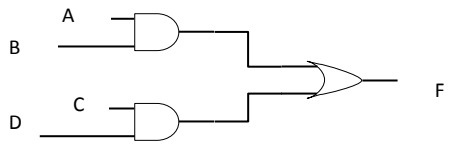


Figura 13.- Circuito combinacional de compuertas AND y OR

CIRCUITO AND-OR-NOT

Cuando se complementa (invierte) la salida de un circuito AND-OR, se obtiene el circuito AND-OR-NOT. Recuerde que el circuito AND-OR implementa directamente la suma de productos. El producto de sumas puede implementarse con un circuito lógico AND-OR-NOT.

Consideremos la función Booleana F(A,B,C,D) = AB + CD ; esta función es similar a la descrita en el circuito anterior AND-OR, es evidente que la diferencia está en que el resultado de la función debe ser negado (invertido) para cumplir con ella (tabla de verdad), es decir, usaremos una compuerta AND de dos entradas para el producto AB, repetiremos dicha compuerta para el producto CD, colocaremos una compuerta OR de dos entradas que realice la suma AB + CD, y al final agregamos una compuerta NOT que invierta la salida proveniente de la OR, el circuito resulta de esta forma:

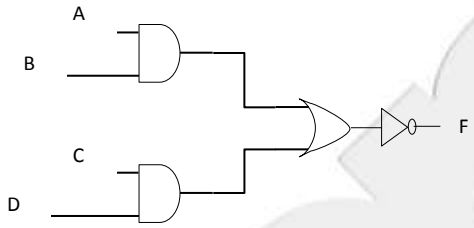


Figura 14.- Circuito combinacional de compuertas AND, OR y NOT

Diseño de circuitos combinacionales

En Circuitos Digitales se entiende por diseñar, al proceso por el cual se obtiene el objeto pedido a partir de unas especificaciones iniciales. Cuando diseñamos circuitos combinacionales, estamos haciendo lo mismo. Partimos de unas especificaciones iniciales y obtenemos un esquema, o plano, que indica qué puertas básicas u otros elementos hay que utilizar así como la interconexión que hay entre ellos.

Los pasos que seguiremos para el diseño son los siguientes:

1. Estudio de las especificaciones iniciales, para entender realmente qué es lo que hay que diseñar. Este punto puede parecer una trivialidad, sobre todo en el entorno académico donde las especificaciones son muy claras. Sin embargo, en la realidad, es muy difícil llegar a comprender o entender qué es lo que hay que diseñar.

2. Obtención de las tablas de verdad y expresiones booleanas necesarias. En el entorno académico este suele ser el punto de partida. Nos describen qué función es la que se quiere implementar y lo hacemos.

3. Simplificación de las funciones booleanas. No basta con implementar una función, hay que implementar la mejor función, de manera que obtengamos el mejor diseño posible, reduciendo el número de puertas lógicas empleadas, el número de circuitos integrados o minimizando el retraso entre la entrada y la salida.

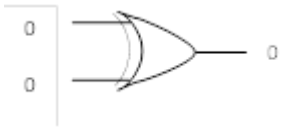
4. Implementación de las funciones booleanas utilizando puertas lógicas. Aquí podemos tener restricciones, como veremos. Puede ser que por especificaciones del diseño sólo se dispongan de puertas tipo NAND o puede ser que sólo podamos utilizar puertas lógicas con el mínimo número de entradas. En ese caso habrá que tomar la función más simplificada y modificarla para adaptarla a este tipo de puertas. El resultado de esto es la obtención de un esquema o plano del circuito.

Instituto Universitario de Tecnología para la Informática – Iutepi

Comparadores

La función básica de un comparador consiste en comparar las magnitudes de dos cantidades binarias para determinar su relación. En su forma más sencilla, un circuito comparador determina si dos números son iguales.

La puerta OR-exclusiva (XOR) se puede emplear como un comparador básico, ya que su salida es 1 si sus dos bits de entrada son diferentes y 0 si son iguales.



Los bits de entrada son iguales

Los bits de entrada son diferentes

Figura 15.- Comparación de bits iguales y diferentes en una compuerta XOR

Además de disponer de una salida que indica si los dos números son iguales, muchos circuitos integrados comparadores tienen salidas adicionales que indican cuál de los dos números que se comparan es el mayor. Esto significa que existe una salida que indica cuándo el número A es mayor que el número B (A > B) y otra salida que indica cuándo A es menor que B (A < B).

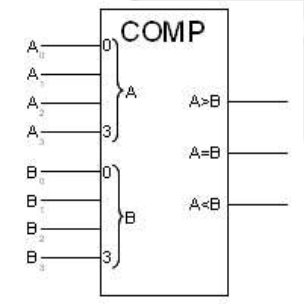


Figura 16.- Símbolo de comparador de dos números de cuatro bits

Para determinar una desigualdad entre los números binarios A y B, en primer lugar se examina el bit de mayor orden de cada número. Las posibles condiciones son las siguientes:

- Si $A_3 = 1$ y $B_3 = 0$, entonces A es mayor que B.
- Si $A_3 = 0$ y $B_3 = 1$, entonces A es menor que B.
- Si $A_3 = B_3$, entonces tenemos que examinar los siguientes bits de orden inmediatamente inferior.

Estas tres operaciones son válidas para cada posición que ocupen los bits dentro del número. El procedimiento general utilizado en un comparador consiste en comprobar una desigualdad en cualquier posición de bit, comenzando por los bits más significativos (MSB). Cuando se encuentra una desigualdad, la relación entre ambos números queda establecida y cualquier otra desigualdad entre bits con posiciones de orden menor debe ignorarse, ya que podrían indicar una relación entre los números completamente opuesta. La relación de más alto orden es la que tiene prioridad.

Sumador – Restador

Los sumadores son muy importantes no solamente en las computadoras, sino en muchos tipos de sistemas digitales en los que se procesan datos numéricos. Comprender el funcionamiento de un sumador básico es fundamental en el estudio de los sistemas digitales. En esta sección se presentan el semisumador y el sumador completo.

Semi-Sumador o Medio Sumador:

Recordemos las reglas básicas de la suma binaria:

+	0	1
0	0	1
1	0	0, 1(acarreo)

Tabla 26.- Suma en Binario

Un semi-sumador admite dos dígitos binarios en sus entradas y genera dos dígitos binarios en sus salidas: un bit de suma y un bit de acarreo.

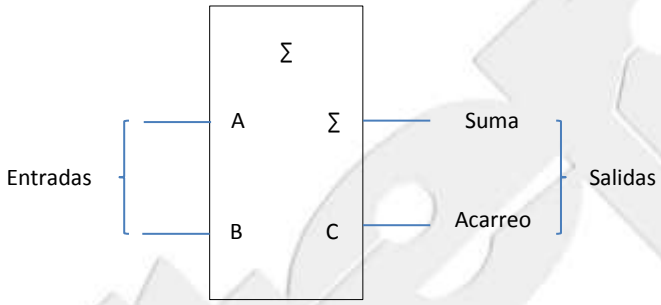


Figura 17.- Símbolo de sumador de dos bits

A partir del funcionamiento lógico de un semi-sumador, las expresiones correspondientes a la suma y al acarreo de salida se pueden obtener como funciones de las entradas. Observe que la salida de acarreo (C) es 1 sólo cuando A y B son 1; por tanto, C puede expresarse como una operación AND de las variables de entrada, C = A.B.

Observe ahora que la salida correspondiente a la suma (Σ) es 1 sólo si las variables A y B son distintas. Por tanto, la suma puede expresarse como una operación XOR de las variables de entrada, $\Sigma = A \oplus B$

A partir de las expresiones anteriores, se puede desarrollar la implementación lógica del funcionamiento de un semi-sumador. La salida de acarreo se produce mediante una puerta AND, siendo A y B sus dos entradas, y la salida de la suma se obtiene mediante una puerta XOR.

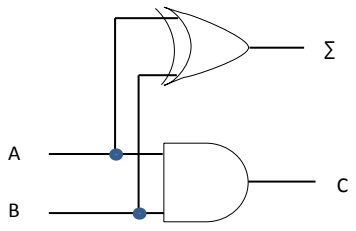


Figura 18.- Circuito sumador de dos bits con acarreo

Sumador Completo

Un sumador completo tiene un acarreo de entrada mientras que el semi-sumador no, es decir, el sumador completo es un semi-sumador que trabaja con dos bit de acarreo.

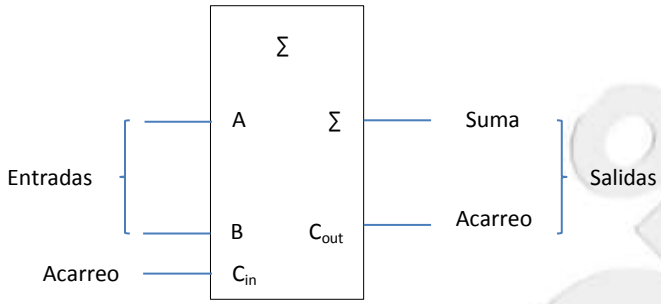


Figura 19.- Símbolo sumador completo de 2 bits

El sumador completo tiene que sumar dos bits de entrada y un acarreo de entrada. Del semi-sumador sabemos que la suma de los bits de entrada A y B es la operación XOR de esas dos variables, A ⊕ B. Para sumar el acarreo de entrada (C_{in}) a los bits de entrada, hay que aplicar de nuevo la operación XOR, obteniéndose la siguiente expresión para la salida de suma del sumador completo: $\Sigma = (A \oplus B) \oplus C_{in}$.

Esto significa que para implementar la función del sumador completo se pueden utilizar dos puertas XOR de 2 entradas. La primera tiene que generar el término A ⊕ B, y la segunda tiene como entradas la salida de la primera puerta XOR y el acarreo de entrada.

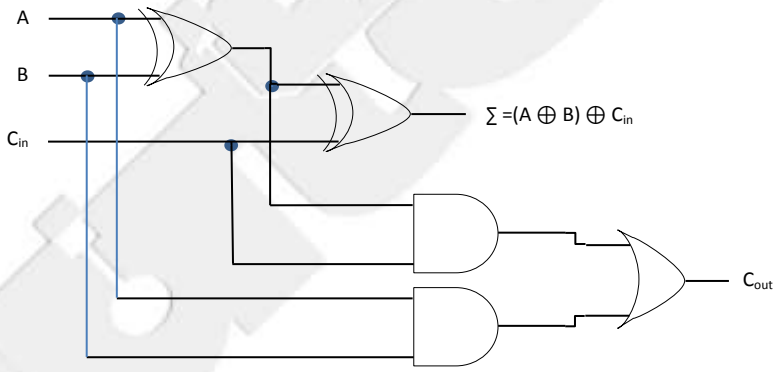


Figura 20.- Circuito sumador completo de dos bits con acarreo

El acarreo de salida es 1 cuando las dos entradas de la primera puerta XOR están a 1 o cuando las dos entradas de la segunda puerta XOR están a 1. El acarreo de salida del sumador completo se obtiene por tanto del producto lógico (AND) de la entradas A y B, y del producto lógico de A ⊕ B y C_{in}, es decir; $C_{out} = AB + (A \oplus B) C_{in}$. Después se aplica la operación OR a estos dos términos. Esta función se implementa y se combina con la lógica de la suma para formar un circuito sumador completo.

Observe que, existen dos semi-sumadores conectados, como se muestra en el diagrama de bloques, cuyos acarreos de salida se aplican a una puerta OR.

Sumador de dos números de 4 bits

Un sumador básico en paralelo de 4 bits se implementa mediante cuatro sumadores completos. Los bits menos significativos (A_0 y B_0) de cada número que se suman, se introducen en el sumador completo que está más a la derecha; los bits de orden más alto se introducen sucesivamente en los siguientes sumadores, aplicando los bits más significativos de cada número (A_3 y B_3) al sumador que está más a la izquierda. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del siguiente sumador de orden superior. Estos acarreos se denominan acarreos internos.

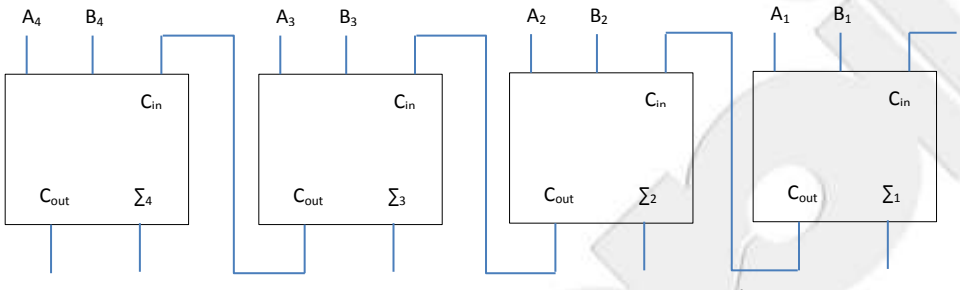


Figura 21.- Diagrama de bloques de sumador completo de dos números de 4 bits

En la mayoría de las hojas de características suministradas por los fabricantes, se denomina C_0 al acarreo de entrada del sumador del bit menos significativo; C_4, en el caso de cuatro bits, sería el acarreo de salida del sumador del bit más significativo; Σ_1 hasta Σ_4 son las sumas de salida.

A continuación se muestra la imagen y diagrama interno del sumador con lógica TTL 7483, el cual puede sumar dos números de 4 bits.

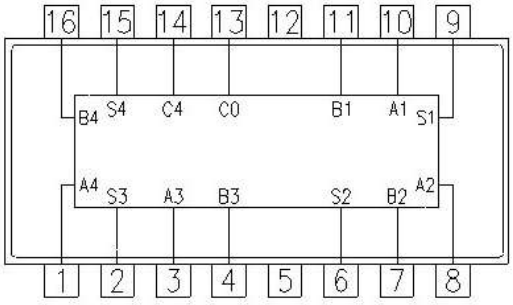
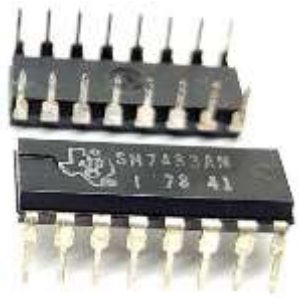


Diagrama Interno Sumador 7483



Circuito Sumador 7483 (Fabricante: Texas Instruments)

Figura 22.- Diagrama interno de sumador completo de 4 bits e Integrado TTL 7483

Restador

Sabemos que la operación resta en binario se realiza haciendo un cambio al sustraendo para realizar una operación de suma, haciendo uso del bloque sumador y el complemento a dos (C2) se puede hacer del sumador de dos números de 4 bits un restador de dos números de 4 bits, comencemos por el sustraendo, que se cambia a complemento a uno (C1) usando una compuerta XOR en la entrada del sumador y además se añade un 1 por la entrada de acarreo, para obtener el complemento a dos (C2 = C1 + 1), veamos el siguiente bloque que ilustra nuestra descripción:

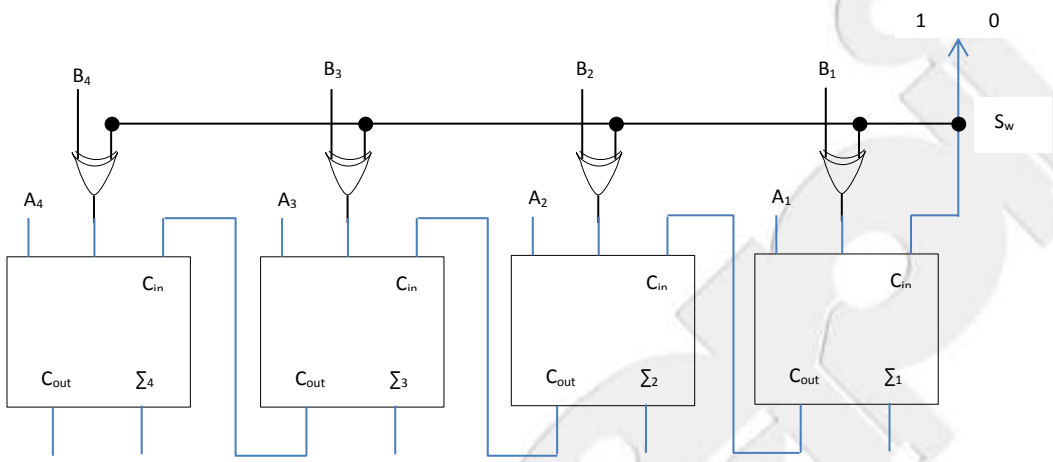


Figura 23.- Diagrama de bloques restador de números de 4 bits

Se puede apreciar que existe un switche (Sw) que está posicionado para seleccionar un uno o cero lógico, esto permite escoger entre sumar (Sw = 0) o restar (Sw = 1). En el caso de una resta, la compuerta XOR está recibiendo los bits del número B (B_1,B_2,B_3,B_4) más el único lógico cuando Sw = 1; esto convierte a B en su complemento a uno (C1), y posteriormente esa misma entrada Sw = 1; está conectado a C_{in} del primer bloque del sumador, lo que hace que B que ya está en C1 pase a C2 y se proceda a realizar la resta a partir de una suma.

Multiplexores – Árbol de Mux

Un multiplexor (Mux) es un circuito combinacional que permite dirigir la información digital procedente de diversas fuentes a una única línea para ser transmitida a través de dicha línea a un destino común. El multiplexor básico posee varias líneas de entrada de datos y una única línea de salida. También posee entradas de selección de datos, que permiten conmutar los datos digitales provenientes de cualquier entrada hacia la línea de salida. A los multiplexores también se les conoce como selectores de datos.

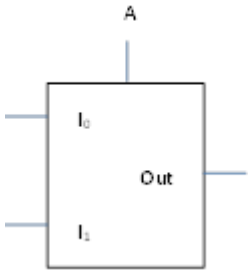
Básicamente un Mux es un circuito combinacional de varias entradas y una única salida, la salida dependerá de las variables que controlan el circuito.

Una aplicación muy útil de los multiplexores/selectores de datos consiste en la generación de funciones lógicas combinacionales en forma de suma de productos. Cuando se emplea de esta manera, este dispositivo puede reemplazar puertas lógicas discretas, puede reducir significativamente el número de circuitos integrados y permite que los cambios en el diseño sean mucho más sencillos.

Instituto Universitario de Tecnología para la Informática – Iutepi

La cantidad de puertos de entrada en un Mux depende de la cantidad de variables de control n, en una relación donde: puertos de entrada = 2^n. Para un Mux de una variable de control, se tendrán dos puertos de entradas de datos ($2^1 = 2$), y así, para un Mux de dos variables de control se tendrán cuatro puertos de entradas de datos, para cuatro variables de control se tienen 16 entradas de datos.

La salida de un Mux obedece a la tabla de verdad de las variables de control, a continuación se describe brevemente un Mux de una variable de control con su tabla de verdad correspondiente:



Multiplexor (Mux), dos entradas de datos (I_0,I_1) una variable de control (A) una salida (Out)

A	Out
0	I_0
1	I_1

Tabla de verdad Mux cuatro entradas

Figura 24.- Símbolo de Mux 2:1. Tabla 27.- Tabla de verdad para Mux 2:1

La interpretación es muy sencilla, para la posición A = 0, la salida del Mux será la entrada I_0, esto significa que, lo que está en ese primer puerto pasa directamente a la salida.

El circuito combinacional de un Mux 2:1, dos entradas de datos y una variable de control es el siguiente:

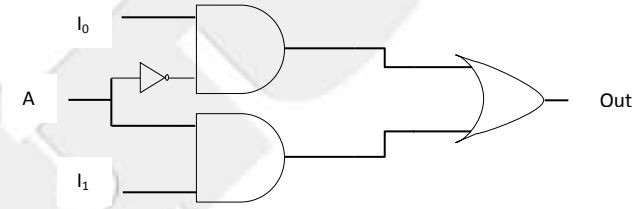
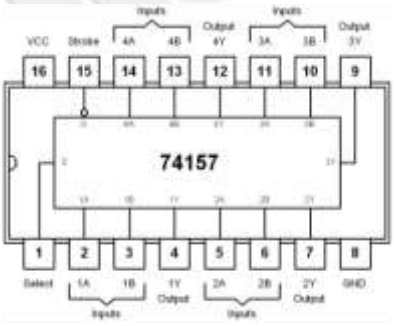


Figura 25.- Circuito combinacional AND-OR-NOT de un Mux 2:1



Multiplexor 4 bloques 2:1 (Fabricante: Texas Instruments)

Figura 26.- Diagrama interno de cuatro Mux 2:1. Circuito integrado 74157

Árbol de MUX

El árbol de Mux, no es más que la unión de varios multiplexores, de manera que, se logren obtener la cantidad de entrada de datos necesarios para un determinado proceso, manifestándose la respuesta en una única salida, por ejemplo, construyamos un Mux 8:1, a partir de otros multiplexores, para esto construiremos un árbol de Mux de dos formas:

- Hacemos uso de dos multiplexores 4:1 y un multiplexor de 2:1, los primeros dos multiplexores de cuatro entradas se encargaran de recibir los ocho datos, la salida de cada Mux 4:1 va a la entrada del Mux 2:1, este último multiplexor tiene una única salida, de esta forma logramos tener un Mux de 8:1 a partir de otros tres multiplexores.
- Se hace uso de cuatro multiplexores 2:1 y un multiplexor 4:1, los primeros multiplexores reciben dos entradas de datos cada uno para un total de ocho, posteriormente cada salida de esos Mux va a la entrada del Mux 4:1, el cual termina entregando una salida, de esta forma también se logra obtener un multiplexor 8:1 a parir de otros cinco multiplexores.

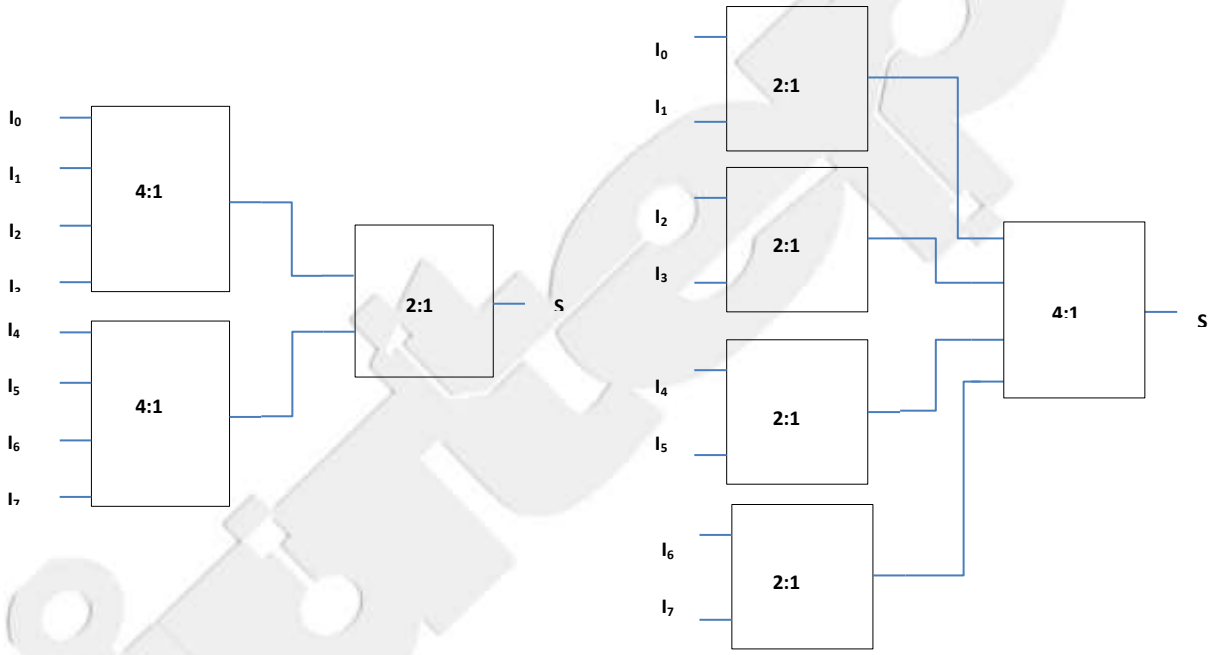
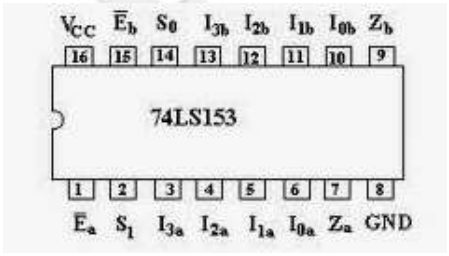


Figura 27.- Diagrama de bloques de árbol de Mux 8:1



Multiplexor 2 bloques 4:1 (Fabricante: Texas Instruments)

Figura 28.- Diagrama Interno de dos bloques Mux 4:1. Circuito Integrado 74153

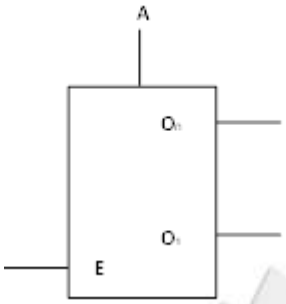
Demultiplexores – Árbol de Demux

Un demultiplexor (DEMUX) realiza la función contraria a la del multiplexor. Toma datos de una línea y los distribuye a un determinado número de líneas de salida. Por este motivo, el demultiplexor se conoce también como distribuidor de datos.

Básicamente un Demux es un circuito combinacional de una entrada y una varias salidas, la salida dependerá de las variables que controlan el circuito.

La cantidad de puertos de salida en un Demux depende de la cantidad de variables de control n, en una relación donde: puertos de salida = 2^n. Para un Demux de una variable de control, se tendrán dos puertos de salida de datos ($2^1 = 2$), y así, para un Deux de dos variables de control se tendrán cuatro puertos de salida de datos.

La salida de un Demux obedece a la tabla de verdad de las variables de control, a continuación se describe brevemente un Demux de una variable de control con su tabla de verdad correspondiente:



A	O_0	O_1
0	Dato	0
1	0	dato

Figura 29.- Simbolo de Demux 1:2. Tabla 28.- Tabla de Verdad Demux 1:2

Aunque el Demux es contrario al Mux, tiene una interpretación de su tabla de verdad muy similar, es claro que el puerto dato (E) pasa directamente a la salida (O_0,O_1), según lo indique la variable de control (A)

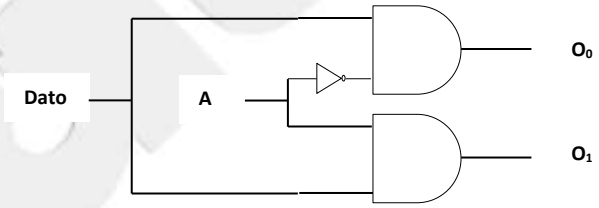


Figura 30.- Circuito combinacional Demux 1:2

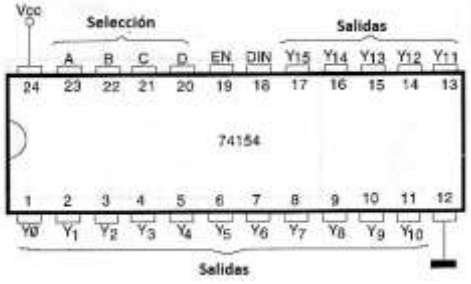


Figura 31.- Integrado 74154 Demultiplexor 1:16

Árbol de Demux

Un árbol de demultiplexores, es una combinación de varios demux para obtener un circuito con el equivalente a un solo demux, es similar a un árbol de Mux pero invertido, donde se combinan una única entrada, para obtener varias salidas. Desarrollemos el siguiente ejemplo, en el cual tenemos la necesidad de construir un circuito para el cual es necesario un Demux de tres variables de control, esto haría que la selección del Demux fuese de 1:8, así las tres variables de control (A,B,C) manejarían las ocho salidas ($2^3 = 8$) O_0 hasta O_7, pero solo poseemos Demux de menor tamaño, por tanto debemos construir el circuito como un árbol de Demux.

Para este ejemplo podemos hacer uso de un Demux 1:2, el cual sería controlado por una variable, y se completaría el árbol utilizando dos Demux 1:4, controlados ambos por las mismas dos variables de control restantes, estos ultimo dos Demux entregarían las ocho salidas necesarias. El circuito de bloques quedaría de la siguiente forma:

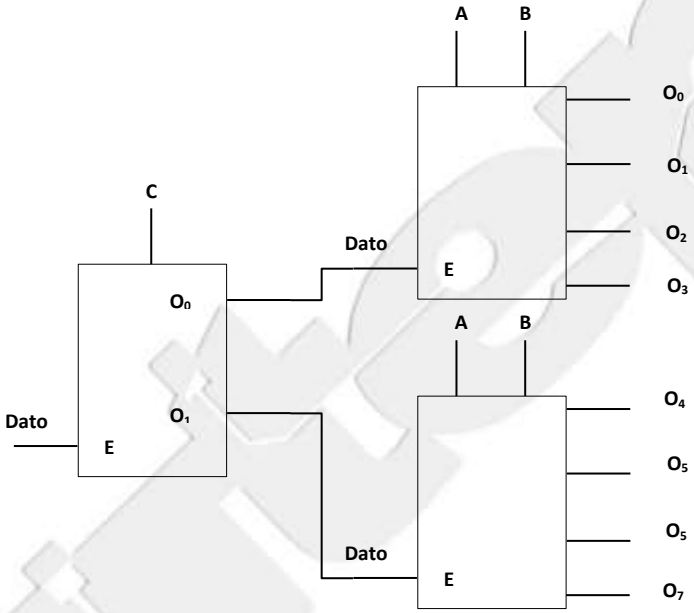


Figura 32.- Diagrama de bloques de árbol de Demux

Autoevaluación

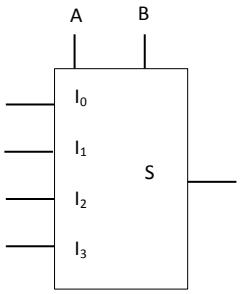
Sumador

- Para un sumador completo de dos bits, determinar el estado lógico (1 o 0) a la salida de cada puerta para las siguientes entradas:

(a) A = 1, B = 1, C_{in} = 1 (b) A = 0, B = 1, C_{in} = 1

Multiplexores

- En el demultiplexor de la Figura 6.85, determinar la salida para los siguientes estados de entrada: $I_0 = 0$, $I_1 = 1$, $I_2 = 1$, $I_3 = 0$, A = 1, B = 0



Introducción a los Circuitos Secuenciales

Los circuitos lógicos que hemos trabajado hasta este punto han sido combinaciones entre ellos, de allí la definición de circuitos combinacionales, sin embargo esto no es suficiente al momento de necesitar almacenar información por un periodo de tiempo prolongado, ya que los circuitos combinacionales solo actúan en un instante de tiempo particular y sus salidas depende de la entrada, sus cambios no están sujetos a la variable tiempo.

Un diagrama a bloques de un circuito secuencial consta de un circuito combinacional al cual se le conectan elementos de memoria que almacenan información binaria (1,0), este almacenaje de información se define como estado del circuito secuencial.

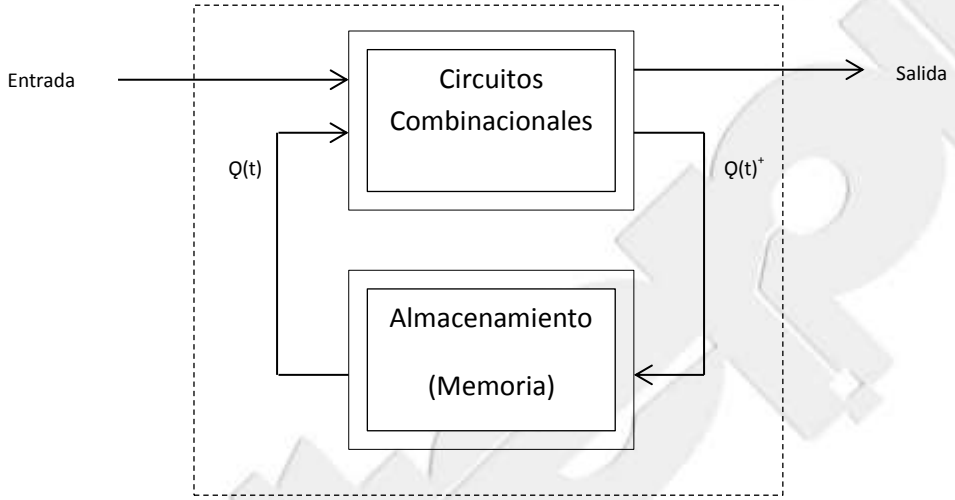


Figura 33.- Diagrama de bloques de un circuito secuencial

El circuito secuencial recibe información binaria de su ambiente a través de la entrada, las cuales, en combinación con el estado actual Q(t) (almacenado), determinan el valor binario de la salida y el siguiente estado $Q(t)^+$.

La mayoría de los sistemas digitales requieren la función de almacenamiento, cuyo objetivo es mantener los datos binarios durante un periodo de tiempo. Algunos dispositivos de almacenamiento se usan para almacenamiento temporal y otros para almacenamiento permanente y pueden memorizar un bit o un grupo de bits.

Un circuito o sistema secuencial queda definido por dos funciones lógicas, llamadas funciones de transición:

- Función de salida: nos indica cómo depende la salida o salidas, de las entradas actuales y del estado actual.
- Función de transición de estado: nos indica como depende el nuevo estado del estado anterior y de las entradas al sistema.

Ejemplos de circuitos secuenciales hay muchos, en especial de la vida cotidiana, si consideramos el funcionamiento de un semáforo, este está basado en un circuito secuencial, el tiempo en el cual las diferentes luces se mantienen encendidas o incluso apagadas, el almacenamiento del estado actual o el estado anterior, que permite cambiar entre una luz roja o verde.

Concepto de Biestable

Un circuito de dos estado so biestable, conocido por su nombre en inglés como FLIP-FLOP, es un circuito que puede mantener dos estados diferentes por un tiempo indefinido, al ser dos y solo dos estados, podemos catalogar un circuito biestable como un circuito binario.

El componente elemental de un circuito secuencial es el Flip-Flop, estos se caracterizan por mantener por un tiempo definido o indefinido, el estado lógico de cero o uno según sea necesario, el mantener un uno lógico o cero lógico por un periodo de tiempo es la base fundamental de una memoria.

Esto significa que un circuito biestable o flip-flip es capaz de memorizar un uno lógico por un periodo de tiempo, pero también puede cambiar su estado y almacenar un cero lógico por un mismo periodo de tiempo u otro diferente, estos cambias de estado se conocen como transición del biestable.

Tipos de biestables

Deben distinguirse tres aspectos en las señales de entrada que producen la transición de un estado a otro:

1. La lógica de disparo, que determinará que el biestable cambie de estado cuando en sus entradas se dé una cierta combinación de señales.
2. El tipo de disparo, que determinará la forma en que las excitaciones de entrada afectan al estado del biestable.
3. El sincronismo en el disparo, que determinará si el funcionamiento del biestable se hará de acuerdo con la presencia de una señal adicional a las entradas, y que se denomina señal de reloj (tren de impulsos).

Podemos clasificar los biestables según estos criterios:

- 1. Atendiendo a la lógica de disparo (modo de funcionamiento):
  - Biestables R-S
  - Biestables J-K
  - Biestables D
  - Biestables T
- 2. Atendiendo al sincronismo en el disparo y tipo de disparo:
  - Asíncronos (latches): funcionan sin señal de reloj.
  - Síncronos (flip-flops): funcionan con señal de reloj.
    - Disparo por nivel de tensión: alto ("1") o bajo ("0")
      - Nivel alto ("1"): El biestable podrá cambiar de estado cuando la señal de reloj esté a "1".
      - Nivel bajo ("0"): El biestable podrá cambiar de estado cuando la señal de reloj esté a "0".
    - Disparo por flanco: de subida o bajada
      - Flanco de subida: El biestable podrá cambiar de estado en el instante en que la señal de reloj pase de "0" a "1".
      - Flanco de bajada: El biestable podrá cambiar de estado en el instante en que la señal de reloj pase de "1" a "0".

FLIP FLOP - LATCHES

El latch (cerrojo) es un tipo de dispositivo de almacenamiento temporal de dos estados (biestable), que se suele agrupar en una categoría diferente a la de los flip-flops. Básicamente, los latches son similares a los flip-flops, ya que son también dispositivos de dos estados que pueden permanecer en cualquiera de sus dos estados gracias a su capacidad de realimentación, lo que consiste en conectar (realimentar) cada una de las salidas a la entrada opuesta. La diferencia principal entre ambos tipos de dispositivos está en el método empleado para cambiar de estado.

Latch S-R (Set-Reset)

Un latch es un tipo de dispositivo lógico biestable o multivibrador. Un latch S-R (Set-Reset) con entrada activa a nivel ALTO se compone de dos puertas NOR interconectadas; un latch con entrada activa a nivel BAJO está formado por dos puertas NAND interconectadas. Observe que la salida de cada puerta se conecta a la entrada de la puerta opuesta. Esto origina la realimentación (feedback) regenerativa característica de todos los latches y flip-flops.

Figura 34.- Flip S-R, nivel bajo NOR, nivel alto NAND

El latch S-R es el único biestable que tiene sentido como asíncrono, aunque también funciona de forma síncrona, los demás biestables requieren reloj para un correcto funcionamiento.

El latch S-R tiene dos entradas:

- R (Reset): permite poner a 0 el estado del biestable, es decir, la salida Q vale 0.
- S (Set): permite poner a 1 el estado del biestable, es decir, la salida Q vale 1.

Tiene dos salidas complementarias: Q y Q̄. Para analizar la tabla de transición basta con que nos fijemos en Q. Para explicar el funcionamiento del latch, vamos a utilizar el latch de puertas NAND.

Asumimos que las dos entradas y la salida Q están a nivel ALTO. Dado que la salida Q se realimenta a una entrada de la puerta NAND inferior y que la entrada está a nivel ALTO, la salida de la NAND inferior tiene que ser un nivel BAJO. Esta salida a nivel BAJO está acoplada de nuevo a una entrada de la NAND superior, asegurando así que su salida sea un nivel ALTO.

Cuando la salida Q está a nivel ALTO, el latch se encuentra en estado SET y permanecerá indefinidamente en él hasta que se aplique un nivel BAJO a la entrada. Si tenemos un nivel BAJO en la entrada **R** y un nivel ALTO en **S**, la salida de la puerta NAND inferior se pone forzosamente a nivel ALTO. Este nivel ALTO en la salida **Q̄** se realimenta a una de las entradas de la puerta NAND superior y, dado que la entrada **S** está a nivel ALTO, la salida de la puerta NAND superior se pone a nivel BAJO. Este nivel BAJO en la salida **Q** se realimenta a una de las entradas de la puerta NAND inferior, asegurando que la salida **Q̄** permanezca a nivel ALTO incluso cuando se elimine el nivel BAJO de la entrada **R**.

Instituto Universitario de Tecnología para la Informática – Iutepi

Cuando la salida **Q** es un nivel BAJO, el latch se encuentra en estado RESET. Ahora el latch permanece indefinidamente en este estado hasta que se aplique un nivel BAJO en la entrada **S**.

Se produce una condición de funcionamiento no válida en un latch **S** - **R** con entradas activas a nivel BAJO, cuando se aplican simultáneamente niveles bajos a las dos entradas, **S** y **R**. Mientras que se mantengan las dos entradas a nivel BAJO, las dos salidas **Q** y **Q̄** deberían forzosamente estar a nivel ALTO, lo que viola la condición de complementariedad de las salidas. Además, si se eliminan simultáneamente los niveles BAJOS, las dos salidas van a tender al nivel BAJO y, dado que siempre va a existir un cierto retraso de propagación de la señal eléctrica a través de las puertas, una de las puertas dominará en la transición a nivel BAJO. Esto hará que la salida de la puerta más lenta permanezca a nivel ALTO. Cuando se produce esta situación, no se puede predecir el siguiente estado del latch.

La tabla de transición es la siguiente, en forma normal y forma compacta:

S	R	**Q**	**Q̄**	Comentario
0	0	NV	NV	Estado No Valido
0	1	1	0	SET
1	0	0	1	RESET
1	1	Q	Q̄	No Cambia

Tabla 29.- Tabla de verdad y transición Latch S-R nivel Bajo (NAND)

Para un latch R –S nivel alto la tabla de verdad es la siguiente

S	R	**Q**	**Q̄**	Comentario
0	0	Q	Q̄	No cambia
0	1	1	0	RESET
1	0	0	1	SET
1	1	NV	NV	Estado No Valido

Tabla 30.- Tabla de verdad Latch S-R nivel Alto (NOR)

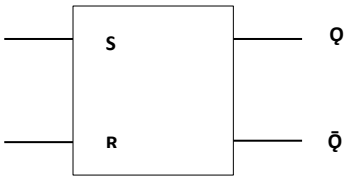


Figura 35.- Símbolo para Latch S – R (NAND)

Instituto Universitario de Tecnología para la Informática - Iutepi

Flip – Flop SR

El latch S – R el cual es asíncrono, puede convertirse en un flip-flop añadiendo una entrada de trenes de impulso, esta entrada es una señal de reloj que hará trabajar el latch en función del tiempo convirtiéndolo en flip-flop. De esta forma, el flip-flop S – R tiene las mismas compuertas que el latch S – R, puede trabajar el nivel bajo con compuertas NAND o altas con las compuertas NOR. A continuación mostramos el esquema de flip-flop S – R, nivel bajo (NAND):

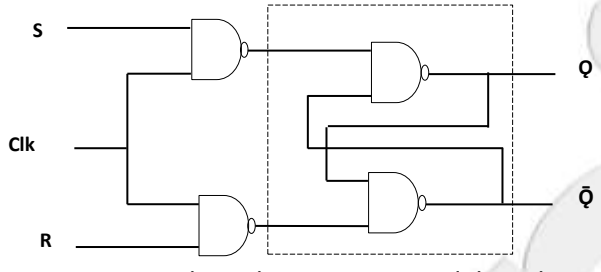


Figura 36. – Fliop-Fliop S-R a partir del Latch S-R

En la figura anterior podemos apreciar en líneas punteadas al latch S – R, al cual se le esta añadiendo una entrada de reloj (Clk) adicional a las entradas S – R, esto conveirte al latch en un FLIP-FLOP.

Flip-Flop D

Este flip-flop tiene la capacidad de transferir datos. En forma básica es un flip-flop S-R con un inversor en la entrada, reduciendo las entradas de datos a solo una (D), la cual pasa directamente a la salida, adicionalmente tiene una entrada de reloj (FF-Síncrono). El flip-flop D también se conoce como flip-flop de retardo (Delay). La figura a continuación muestra el circuito de un FF-D.

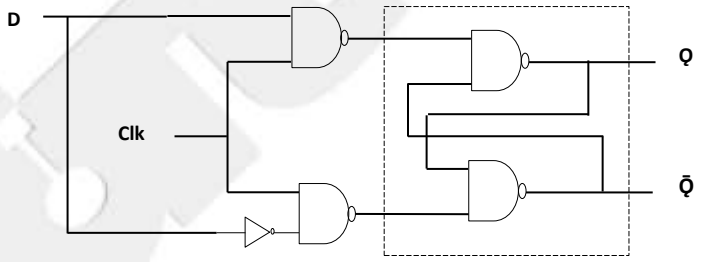


Figura 37.- Flip-Flop D – a partir del Latch S-R NAND

A continuación se muestra la tabla de verdad o transición para un flip-flop D basado en FF-SR NAND

Clk	D	Q	Q̄	Comentario
1	0	0	1	RESET
1	1	1	0	SET
0	x	Q	Q̄	Sin cambio

Tabla 31.- tabla de Verdad y transición del Flip-Flop D

Flip-Flop JK

El flip-flop J-K es uno de los más ampliamente utilizados, en algunos textos llamado flip-flop universal. Las denominaciones J y K de sus entradas no tienen ningún significado conocido, excepto el hecho de que son dos letras consecutivas del alfabeto, en algunos libros lo definen como Jump-keep.

Este flip-flop es similar al FF R-S, pero se diferencia en que la indeterminación que se presenta cuando las dos entradas son 1, no existe. La entrada J es la equivalente a la entrada S de un flip-flop R-S y la entrada K, al equivalente a la entrada R. En este caso, para la combinación J = 1 y K = 1; el estado cambia de valor, es decir, si tenía el valor 0 pasa a valor 1 y viceversa, a este modo de funcionamiento se le denomina modo de basculación.

A continuación se muestra el circuito de un FF-JK con base a un FF-SR de nivel bajo (NAND)

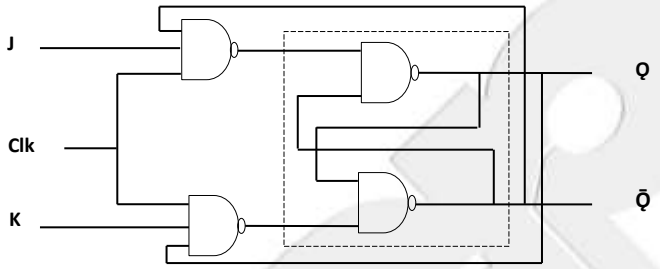


Figura 38.- Flip-Flop J-K , a partir del Latch S-R NAND

La tabla de verdad o de transición del FF-JK se muestra a continuación:

Clk	J	K	Q	Q̄	Comentarios
1	0	0	Q	Q̄	Sin Cambio
1	0	1	0	1	RESET
1	1	0	1	0	SET
1	1	1	Q̄	Q	Bascula / Toggle

Tabla 32.- Tabla de verdad y transición del Flip-Flop J-K

Autoevaluación

- Describir la principal diferencia entre un latch S-R con entrada de habilitación y un flip-flop S-R disparado por flanco.
- ¿Cuál es la diferencia en el funcionamiento básico entre un flip-flop J-K y un flipflop S-R?
- ¿Cómo se tiene que conectar un flip-flop J-K para funcionar como un dispositivo divisor por 2?
- ¿Cuántos flip-flops son necesarios para obtener un dispositivo divisor por 64?
- Si un latch S-R tiene un 1 en la entrada S y un 0 en la entrada R y a continuación la entrada S pasa a 0, el latch estará en:

(a) Estado SET (b) Estado RESET (c) condición no válida (d) borrado

- El estado no válido de un latch S-R se produce cuando

(a) S = 1, R =0 (b) S = 0, R =1 (c) S = 1, R =1 (d) S = 0, R =0

Instituto Universitario de Tecnología para la Informática – Iutepi

Contadores y Tipos

La función de contar es muy importante en los sistemas digitales. Existen muchos tipos de contadores digitales, pero su objetivo básico es contar sucesos representados por cambios de nivel o impulsos, o generar una secuencia de códigos particular. Para contar, el contador debe recordar el número actual, con el fin de poder pasar correctamente al siguiente valor de la secuencia. Por tanto la capacidad de almacenamiento es una característica importante en todos los contadores, por lo que generalmente se utilizan flip-flop para su implementación.

Los contadores producen un dato binario que va cambiando a cada flanco de reloj, de una secuencia prefijada. La secuencia suele ser la binaria ascendente o descendente, pero puede ser también BCD. Los contadores se llaman síncronos si todos los bits del dato producido cambian al mismo tiempo. Si no ocurre así, se llaman asíncronos.

Dado que el estado del contador debe cambiar cuando llega un flanco de la señal de entrada, se utilizan Flip-Flop J-K para construir los contadores, conectando la línea con los impulsos a contar a la entrada de reloj.

Tipos de contadores

Contador es asíncrono: Cuando la salida del biestable es la entrada de reloj del biestable siguiente. Estos contadores llevan una secuencia (ascendente o descendente) que se repite indefinidamente.

Contador es síncrono: Cuando la señal de reloj se conecta a la entrada de reloj de cada uno de los biestables. Se utiliza cuando los estados por los que pasa (secuencia) no son correlativos.

Características de algunos contadores

Dependiendo de las necesidades del circuito, la forma de contar puede ser diferente, describiremos estos tipos de forma de contar de la siguiente manera:

1. Atendiendo al código que cuentan
- Binario (natural)
- BCD
- En anillo
- En Gray
- Johnson

2. Atendiendo al sentido de conteo
- Contador hacia arriba (ascendentes)
- Contador hacia abajo (descenentes)
- Contador en ambos sentidos, no simultáneos

3. Atendiendo a la posibilidad de preselección
- Contador con carga en paralelo
- Contador con puesta a cero inicial solamente

4. Atendiendo a la forma de propagarse la señal de reloj internamente
- Contador asíncrono (contadores con propagación)
- Contador síncrono con acarreo serie
- Contador síncrono con acarreo paralelo

Contador Asíncrono Binario de dos Bits

La siguiente figura presenta un contador de 2 bits conectado para que funcione en modo asíncrono. Observe que el reloj (CLK) está conectado únicamente a la entrada de reloj (C) del primer flip-flop, FF0. El segundo flip-flop, FF1, se dispara mediante la salida $\bar{Q}_0$ de FF0. FF0 cambia de estado durante el flanco positivo de cada impulso de reloj, pero FF1 sólo cambia cuando es disparado por una transición positiva de la salida $\bar{Q}_0$ de FF0. Debido al retardo de propagación inherente al paso de las señales por un flip-flop, las transiciones de los impulsos de entrada del reloj y de la salida $\bar{Q}_0$ de FF0 no pueden ocurrir nunca al mismo tiempo. Por tanto, los dos flip-flops nunca se disparan de forma simultánea, por lo que el modo de funcionamiento de este contador es asíncrono.

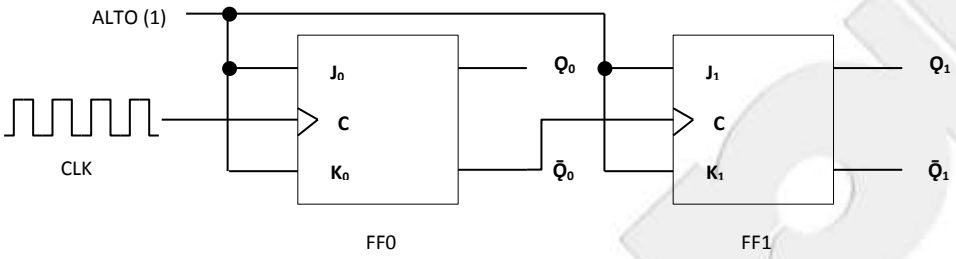


Figura 39.- Contador Asíncrono binario de 2 bits

Vamos a examinar el funcionamiento básico del contador asíncrono de la Figura, aplicando cuatro impulsos de reloj a FF0 y observando la salida Q de cada flip-flop. Ambos flip-flops están conectados en modo de basculación (J = 1, K = 1) y se presupone que, inicialmente, están en estado RESET (Q a nivel BAJO). El flanco positivo de CLK1 (impulso de reloj 1) hace que la salida Q_0 de FF0 pase a nivel ALTO. Al mismo tiempo, la salida pasa a nivel BAJO, pero esto no afecta a FF1, ya que tiene que ser una transición positiva la que le dispare. Después del flanco anterior de CLK1, Q_0 = 1 y Q_1 = 0. El flanco positivo de CLK2 hace que Q_0 pase a nivel BAJO. La salida se pone a nivel ALTO y dispara FF1, haciendo que Q_1 pase a nivel ALTO. Tras el flanco anterior de CLK2, Q_0 = 0 y Q_1 = 1. El flanco positivo de CLK3 hace que Q_0 pase a nivel ALTO de nuevo. La salida se pone a nivel BAJO y no afecta al estado de FF1. Por tanto, tras el flanco anterior de CLK3, Q_0 = 1 y Q_1 = 1. El flanco positivo de CLK4 hace que Q_0 pase a nivel BAJO, mientras que se pone a nivel ALTO y dispara FF1, haciendo que Q_1 pase a nivel BAJO. Después del flanco anterior de CLK4, Q_0 = 0 y Q_1 =0. El contador ha vuelto a su estado original (los dos flip-flops se encuentran en estado RESET).

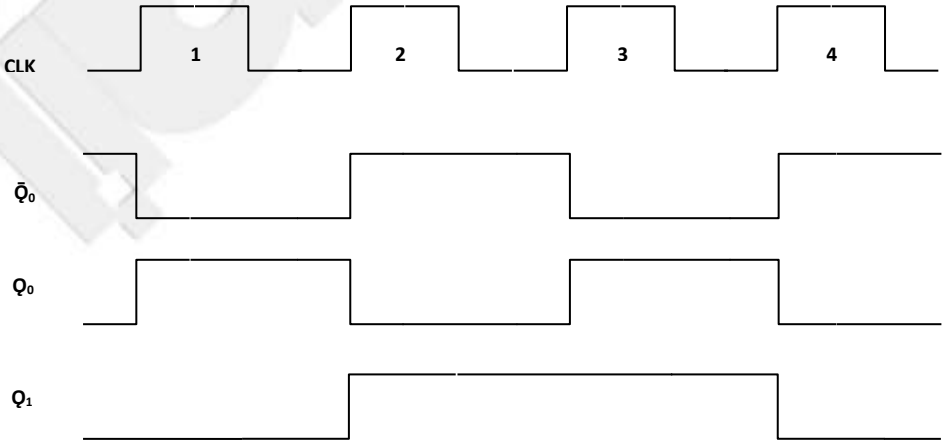


Figura 40.- Diagrama de tiempos y salidas del contador del contador asíncrono binario

Contador Síncrono Binario de dos bits

La Figura 41, muestra un contador binario síncrono de 2 bits. Observe que debe utilizarse una disposición distinta a la del contador asíncrono para las entradas J1 y K1 de FF1, con el fin de poder conseguir una secuencia binaria.

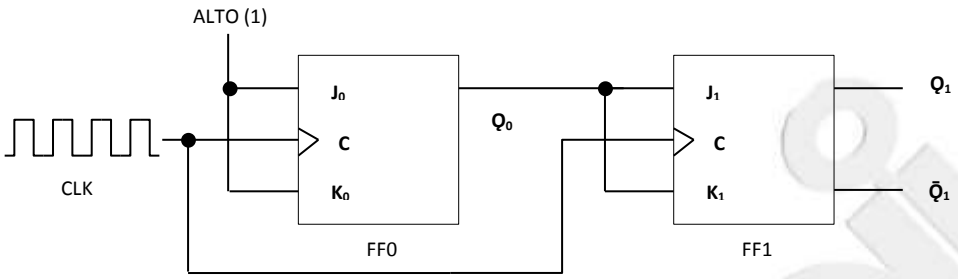


Figura 41.- Contador Síncrono Binario de 2 Bits

El funcionamiento de este contador síncrono es el siguiente: en primer lugar, se supone que el contador se encuentra inicialmente en el estado binario 0; es decir, los dos flip-flop's se encuentran en estado RESET.

Cuando se aplica el flanco positivo del primer impulso de reloj, FF0 bascula, por lo que Q_0 se pone a nivel ALTO. ¿Qué le ocurre a FF1 en el flanco positivo de CLK1? Para averiguarlo, vamos a fijarnos en las condiciones de entrada de FF1.

Las entradas J1 y K1 están ambas a nivel BAJO, ya que están conectadas a Q_0, y ésta todavía no se ha puesto a nivel ALTO. Recuerde que existe un retardo de propagación desde el flanco de disparo del impulso de reloj hasta que, realmente, se realiza la transición en la salida Q.

Por tanto, J = 0 y K = 0 cuando se aplica el flanco anterior del primer impulso de reloj. Ésta es una condición de no cambio y, por tanto, FF1 no cambia de estado. Después de CLK1, $Q_0 = 1$ y $Q_1 = 0$ (que corresponde al estado binario 1).

Cuando se produce el flanco anterior de CLK2, FF0 bascula y Q_0 se pone a nivel BAJO. Puesto que FF1 tiene un nivel ALTO ($Q_0 = 1$) en sus entradas J1 y K1 durante el flanco de disparo del impulso de reloj, el flip-flop bascula y Q_1 pasa a nivel ALTO. Por lo que, después de CLK2, $Q_0 = 0$ y $Q_1 = 1$ (que corresponde al estado binario 2).

Cuando se produce el flanco anterior de CLK3, FF0 bascula de nuevo al estado SET ($Q_0 = 1$) y FF1 permanece en estado SET ($Q_1 = 1$), ya que sus entradas J1 y K1 están ambas a nivel BAJO ($Q_0 = 0$). Tras este flanco de disparo, $Q_0 = 1$ y $Q_1 = 1$ (que corresponde al estado binario 3).

Finalmente, durante el flanco anterior de CLK4, Q_0 y Q_1 se ponen a nivel BAJO, dado que ambos flip-flop están en modo de basculación debido al valor presente en sus entradas J y K. El contador acaba de iniciar un nuevo ciclo a partir de su estado original, 0 binario.

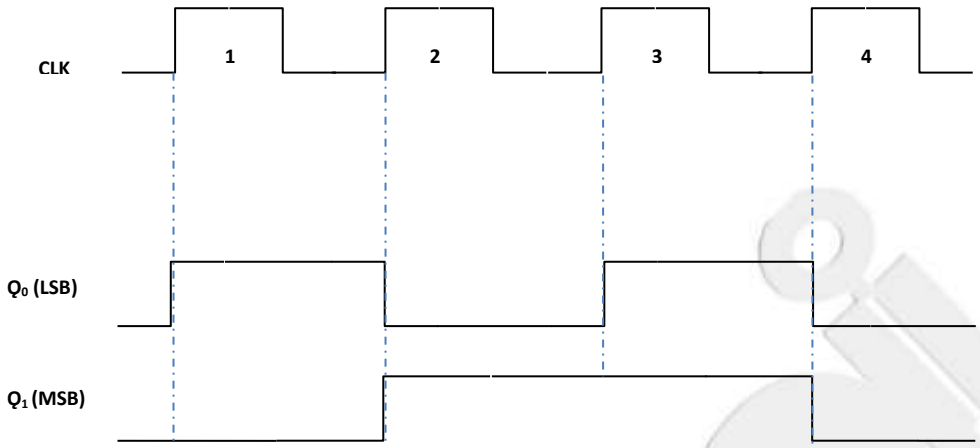


Figura 42.- Diagrama de tiempos del contador síncrono de 2 bits

El diagrama de tiempos completo del contador se muestra en la Figura 42, observe que todas las transiciones de las señales son coincidentes; es decir, no se indican los retardos de propagación.

Aunque los retardos son un factor importante en el funcionamiento de un contador síncrono, se suelen omitir para simplificar los diagramas de tiempos generales.

Si no se muestran los pequeños retardos y las diferencias de temporización, se puede conseguir relacionar mejor las señales resultantes de un circuito lógico.

Sin embargo, en circuitos digitales de alta velocidad, estos pequeños retardos son una consideración importante en el diseño y la localización de averías.

Autoevaluación

- ¿En qué se diferencia un contador síncrono de uno asíncrono?
- Un contador binario de 4 bits ascendente/descendente se encuentra en modo descendente y en el estado 1010. En el siguiente impulso de reloj, ¿a qué estado pasa?

Instituto Universitario de Tecnología para la Informática – Iutepi

Conclusiones

Al terminar esta asignatura el estudiante debe estar familiarizado con el sistema de numeración binario y su importancia no solo en la electrónica sino también en la informática y más aún en los sistemas digitales.

El manejo de las compuertas lógicas TTL como AND, OR y NOT que son la base del entendimiento de los circuitos combinados o combinacionales. Circuitos que debido a razones de diseño deberán ser simplificados por medio del método de Karnaugh o mapas de Karnaugh.

El estudiante deberá poder comprender como se manejan los comparadores usando compuertas lógicas en combinación y como agregar al diseño la variable tiempo (reloj) la cual es fundamental en los circuitos de estados no estables (aestables) o de un estado estable (monoestable) e incluso de dos estados controlable (biestables) por medio de los flip-flop.

Tener la capacidad de manejar compuertas lógicas y de diseñar circuitos que controlen o manejen ciertos sistemas de control, como semáforos o la apertura y cierra de puertas automáticas, el objetivo de esta asignatura.

Bibliografía

1. Floyd, Thomas. “Fundamentos de Sistemas Digitales”. Pearson Educación S.A., Madrid 2.006
2. Malik, Norbert. “Circuitos Electrónicos, Análisis, Simulación y Diseño”. Prentice Hall, Madrid 1.998
3. Tokheim, Roger. “Principios Digitales”. Mc Graw Hill. Serie Shaum.