Jorge Aliaga – Verano 2013

# Si – No- Si – Si- No

Parece raro que alguien se pudiera comunicar con solo dos palabras. Es lo que hacemos con todos los dispositivos digitales que usan el código binario ( 0 y 1 ) o dos estados lógicos (falso y verdadero). La práctica 5 es muy simple, si conoce la base del sistema binario y los conceptos fundamentales de lógica.

## Sistema de numeración binario

El sistema de numeración binario es similar conceptualmente al decimal, con la diferencia que tiene solamente dos caracteres (en lugar de 10) y cada dígito indica el número por el que se debe multiplicar la potencia respectiva de 2, es decir 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, … (en lugar de 10), obteniendo el número final al sumar todos los dígitos.

Por ejemplo

$137_{10} = (1*100 + 3*10 + 7)_{10} = 10001001_2 = (1*10000000 + 1*1000 + 1)_2$

Además del código binario existen otros códigos que usan 0 y 1, que tienen diversas propiedades. El código de Gray tiene una particularidad llamativa: entre un número y el siguiente solamente cambia un dígito.

Otros códigos, como el BCD es como el binario, pero que asigna 4 dígitos binarios a cada número decimal, el exceso 3 (XS3) se obtiene de sumar 3 al correspondiente número en BCD.

¿Cómo se suman y restan dos números binarios?

```
  1 1 0 1 1          1 1 0 1 1
+   1 1 0 1        -   1 1 0 1
-------------      -------------
  1 0 1 0 0 0          1 1 1 0
```

(27 + 13 = 40)  (27 – 13 = 14)

| **BIN** | **GRAY** |
|---|---|
| 0000 | 0000 |
| 0001 | 0001 |
| 0010 | 0011 |
| 0011 | 0010 |
| 0100 | 0110 |
| 0101 | 0111 |
| 0110 | 0101 |
| 0111 | 0100 |
| 1000 | 1100 |
| 1001 | 1101 |
| 1010 | 1111 |
| 1011 | 1110 |
| 1100 | 1010 |
| 1101 | 1011 |
| 1110 | 1001 |
| 1111 | 1000 |

$7_{10} = 0111_2 = 0100_{GRAY}$

$8_{10} = 1000_2 = 1100_{GRAY}$

| **BCD** | **XS3** |
|---|---|
| 0000 | 0011 |
| 0001 | 0100 |
| 0010 | 0101 |
| 0011 | 0110 |
| 0100 | 0111 |
| 0101 | 1000 |
| 0110 | 1001 |
| 0111 | 1010 |
| 1000 | 1011 |
| 1001 | 1100 |

$12_{10} = 0001\ 0010_{BCB} = 0100\ 0101_{XS3}$

La particularidad del código XS3 es que el complemento a 9 de un número se obtiene cambiando 0 a 1 y 1 a 0. Ej: El complemento a 9 de 1 ($0100_{XS3}$) es 8 ($1011_{XS3}$).

Si se expresa un número y en base **r**:

$Y = a_0 + a_1\, r + a_2\, r^2 + \ldots + a_n\, r^n$

El complemento a r-1 del número y será

$Y' = (r\text{-}1\text{-}a_{0)} + (r\text{-}1\text{-}a_1)\, r + (r\text{-}1\text{-}a_2)\, r^2 + \ldots + (r\text{-}1\text{-}a_n)\, r^n = (r\text{-}1_{)} + (r\text{-}1)\, r + (r\text{-}1)\, r^2 + \ldots + (r\text{-}1)\, r^n - Y = r^{(n+1)} - 1 - Y$

Por lo tanto

$X - Y = X + Y' + 1 - r^{(n+1)}$

Es decir, salvo por una potencia (n+1) de la base, puedo transformar una resta en una suma si paso al complemento. En el caso del XS3 entonces, es particularmente fácil construir el complemento a nueve, y por lo tanto puedo restar números BCD pasándolos a XS3, tomando el complemento del sustraendo y sumándole 1.

Algebra de Boole

Se puede definir una estructura algebraica con las siguientes propiedades:

1. Propiedad conmutativa: $x + y = y + x$ ; $x \cdot y = y \cdot x$

2. Propiedad asociativa: $x + (y + z) = (x + y) + z$ ; $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

3. Propiedad distributiva: $x + (y \cdot z) = (x + y) \cdot (x + z)$ ; $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$

4. Propiedad de los neutros. Existen 0 y 1 tales que: $x + 0 = x$ ; $x \cdot 1 = x$

5. Propiedad de los opuestos. Existe $\bar{x}$ tal que: $x + \bar{x} = 1$ ; $x \cdot \bar{x} = 0$

Si X e Y son números binarios 0 y 1 (falso y verdadero) estas propiedades se verifican si:

**X + Y = q** cumple

| **X** | **Y** | **X + Y = Q** | $\overline{X + Y} = \overline{Q}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

y **X . Y = R**

| **X** | **Y** | **X . Y = R** | $\overline{X \cdot Y} = \overline{R}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

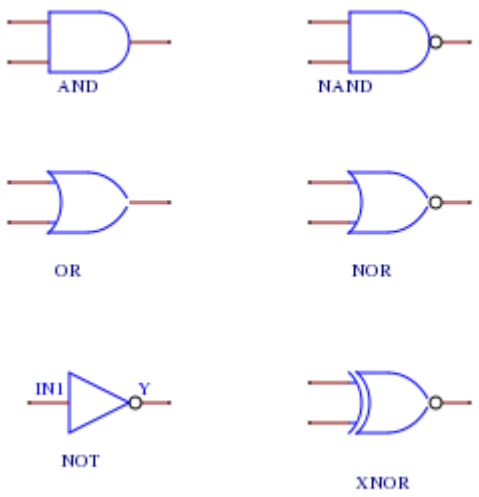
Si se piensa en términos de funciones lógicas, "**X + Y**" es la función "O" o "**OR**" mientras que "**X . Y**" es la función "Y" o "**AND**". La compuerta "AND" con la salida negada se denomina "NAND" mientras que la compuerta "OR" con la salida negada se llama "NOR".

Entre las propiedades que se verifican, están las de **De Morgan**

$$\overline{X + Y} = \overline{X} \cdot \overline{Y}$$

$$\overline{X \cdot Y} = \overline{X} + \overline{Y}$$

Los símbolos que representan funciones con compuertas lógicas son:




Por una cuestión histórica, cuando se desarrollaron los circuitos integrados de compuertas surgieron dos "familias", los basados en transistores bipolares, llamada TTL, y los basados en transistores de efecto de campo, llamados CMOS. Mientras que los TTL se identificaron por números del tipo 74XX, los CMOS tomaron los números 40XX, donde las XX indicaban que función cumplía el circuito.

De esa forma un chip con 4 compuertas NAND de dos entradas es un 7400 o un 4011 según si es TTL o CMOS. Un chip con 4 compuertas NOR de dos entradas es un 7402 o un 4001 según si es TTL o CMOS. Un chip con 6 inversores es un 7404 o un 4069 según si es TTL o CMOS.

Mientras que TODOS los TTL se alimentan con +5V y 0V los CMOS se pueden alimentar con una tensión positiva de entre 3V y 15V, y 0V. Originalmente los TTL eran más rápidos mientras que los CMOS consumían menos potencia.

Para que dos circuitos con compuertas lógicas se puedan interconectar, y los resultados obtenidos sean correctos, deben "entender" a un CERO y a un UNO de la misma forma. Si bien en un TTL, que se alimenta con 5V y 0V el UNO es el estado de 5V y el CERO el de 0V, uno se podría preguntar ¿Qué pasa si la entrada de una compuerta es 4V? ¿Y si fuera 3V, 2V? En algún momento pasará a tomarla como un CERO. Tanto porque tienen alimentaciones distintas como porque tienen rangos de UNO y CERO distintos, las compuertas TTL y CMOS no se podían interconectar. Es decir, uno debía optar, en función de cual fuera mejor para una aplicación, por una u otra familia.

A lo largo de su historia, de casi 50 años. Se fueron logrando mejoras, que se indicaron mediante el agregado de letras. Las características de cada uno eran:

| **TTL** | **CMOS** |
|---|---|
| 74XX ORIGINAL | 40XX ORIGINAL |
| 74HXX ALTA VELOCIDAD | 40XXB MAYOR VELOCIDAD |
| 74LXX BAJO CONSUMO | 74CXX PINOUT TTL |
| 74SXX ALTA VELOCIDAD (SCHOTTKY) | 74HCXX IGUAL VELOCIDAD QUE 74LSXX COMP TTL |
| 74LSXX SCHOTTKY BAJOS CONSUMO | 74ACXX IGUAL QUE 74FXX |
| 74FXX FAST | 74HCT ALTA VELOCIDAD COMP TTL |
| 74ASXX ADVANCED SHOTTKY | 74ACT ADVANCED CMOS COMP TTL |
| 74ALSXX ADVANCED LOW POWER SHOTTKY | |

En las últimas versiones de CMOS, ya son compatibles con las compuertas TTL, y por lo tanto las familias pueden converger. Los marcados en rojo son aquellos que hoy se usan. Mucho de los otros son históricos.

Con todo esto uno puede tomar la cantidad de entradas digitales que quiera y escribir la función lógica entre ellas que quiera. Esto se puede expresar tanto por ecuaciones como a través de una tabla de verdad. Por ejemplo, si A, B y C son entradas, y Q está definido por la siguiente tabla de verdad:

| **A** | **B** | **C** | **Q** |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Q = $\bar{A}\ \bar{B}$ C + A $\bar{B}\ \bar{C}$ + A $\bar{B}$ C + A B C

Porque **Q** vale **UNO** (es verdadero) cuando **A** es **CERO** Y **B** es **CERO** Y **C** es **UNO** "**O**" **A** es **UNO** y **B** es **CERO** y **C** es **CERO "O"** …

Esta ecuación no es única, en el sentido de que se pueden usar propiedades del álgebra para simplificarla. Un método gráfico de hacer esto es usar los **DIAGRAMAS DE KARNAUGH**. Para este método se agrupan las entradas, ordenándolas según el código gray, y se buscan bloques de UNOS. En el ejemplo anterior, será:

| AB<br>C | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

Obteniéndose Q = $A\,\bar{B}$ + $A\,C$ + $\bar{B}\,C$

La práctica 5 propone algunos usos de compuertas lógicas actuando en función de la COMBINACION de las entradas, denominada **lógica combinacional**. Uno de estos usos es probar un Multiplexor, que es un circuito integrado que permite seleccionar electrónicamente (con un número que representa una "dirección") cuál de las 8 entradas se conecta a la salida.

El circuito de la Figura 5.3 usa 4 entradas para "contar" de 1 a 12, representando los doce meses del año. Tres de las entradas ($A_2$=**A**, $A_1$=**B** y $A_0$=**C**) cumplen la función de seleccionar cuál de las 8 entradas, $D_i$, se conecta a la salida. La entrada $A_3$ se usa también para fijar los niveles de las entradas, ya sea directamente o a través de un inversor. El circuito "funciona", es decir pone un "1" en la salida Q en los meses que tienen 31 días, porque se han seleccionado de manera adecuada cuales de las entradas se conecta a $A_3$ (grisado en la tabla) y cuáles a $\overline{A3}$ .

| C | B | A | | $D_i$ | Q | Mes |
|---|---|---|---|---|---|---|
| $A_0$ | $A_1$ | $A_2$ | $A_3$ | i | | |
| 0 | 0 | 0 | 0 | | | |
| 0 | 0 | 0 | 1 | 0 | 1 | 1-E |
| 0 | 0 | 1 | 0 | 1 | 0 | 2- F |
| 0 | 0 | 1 | 1 | 1 | 1 | 3- M |
| 0 | 1 | 0 | 0 | 2 | 0 | 4- A |
| 0 | 1 | 0 | 1 | 2 | 1 | 5- M |
| 0 | 1 | 1 | 0 | 3 | 0 | 6- J |
| 0 | 1 | 1 | 1 | 3 | 1 | 7- J |
| 1 | 0 | 0 | 0 | 4 | 1 | 8- A |
| 1 | 0 | 0 | 1 | 4 | 0 | 9- S |
| 1 | 0 | 1 | 0 | 5 | 1 | 10- O |
| 1 | 0 | 1 | 1 | 5 | 0 | 11- N |
| 1 | 1 | 0 | 0 | 6 | 1 | 12- D |

Luego se prueba un 7483, que funciona como SUMADOR de dos números de 4 bits.

| | | | | |
|---|---|---|---|---|
| | | | | $C_0$ |
| | $A_4$ | $A_3$ | $A_2$ | $A_1$ |
| + | $B_4$ | $B_3$ | $B_2$ | $B_1$ |
| $C_4$ | $\Sigma_4$ | $\Sigma_3$ | $\Sigma_2$ | $\Sigma_1$ |

$C_4$ es el bit de “acarreo” (el “me llevo”) de la suma, y obviamente es una SALIDA. En el caso que los números a sumar tengan mas de 4 bits, los agrupo de a 4, empezando por los menos significativos, y uso tantos 7483 como agrupaciones tenga. $C_0$ es una ENTRADA, y se utiliza para conectar la salida $C_4$ que sumó los bits menos significativos anteriores. Obviamente el primer 7483 tendrá $C_0$ = 0, dado que no hay en ese caso bits menos significativos.

Finalmente se usa un 7485, que es un COMPARADOR de dos números de 4 bits. La comparación del número A($A_3$ $A_2$ $A_1$ $A_0$) con el número B($B_3$ $B_2$ $B_1$ $B_0$), puede dar 3 resultados: **<**, **=**, **>**. El circuito tiene 3 entradas adicionales, de manera que si los números A y B tienen mas bits, por ej A($A_7$ $A_6$ $A_5$ $A_4$ $A_3$ $A_2$ $A_1$ $A_0$), B($B_7$ $B_6$ $B_5$ $B_4$ $B_3$ $B_2$ $B_1$ $B_0$), se compara con un 7485 $A_7$ $A_6$ $A_5$ $A_4$ con $B_7$ $B_6$ $B_5$ $B_4$ y con otro 7485 $A_3$ $A_2$ $A_1$ $A_0$ con $B_3$ $B_2$ $B_1$ $B_0$, y el resultado de esta última comparación, que podrá ser **<**, **=**, **>**, se ingresa como ENTRADA a la comparación de los bits mas significativos. Obviamente esta ENTRADA, en el comparador de los bits MENOS significativos, tiene que estar puesta en **=**, dado que justamente no existen bits menores que puedan dar un resultado distinto.

ATENCION: RECUERDE QUE LOS CIRCIUTOS CON COMPUERTAS ESTÁN HECHOS CON DIODOS Y TRANSISTORES. TODOS LOS CIRCUITOS DEBEN SER **ALIMENTADOS**, Y CON LA TENSIÓN CORRECTA EN LA PATA CORRECTA, SEGÚN INDICA CADA HOJA DE DATOS.

A modo de ejemplo, algunos circuitos que se comportan como compuertas lógicas, solo para que vean que no son “mágicas”, y que con lo que ya vieron los pueden entender.