Práctica 1 - Explicación

Representación en Punto fijo (Parte 1): Números sin signo.

Lógica y compuertas (Parte 1): Funciones lógicas elementales. Puertas lógicas.

Objetivos de la práctica: que el alumno sea capaz de:

- ***Representar e interpretar números sin signo.***
- ***Realizar operaciones aritméticas e interpretar los flags de acarreo y cero.***
- ***Realizar operaciones lógicas***
- ***Usar máscaras y realizar equivalencias entre operaciones sucesivas.***
- ***Establecer la salida de circuitos combinatorios simples.***
- ***Confeccionar tablas de verdad.***
- ***Describir la relación entre entradas y salidas por ecuaciones.***

Bibliografía:

- ***"Organización y Arquitectura de Computadoras" de W. Stallings, capítulo 8.***
- ***Apunte 1 de la cátedra, "Sistemas de Numeración: Sistemas Enteros y Punto Fijo".***
- ***"Principios de Arquitectura de Computadoras" de Miles J. Murdocca, apéndice A, pág. 441.***
- ***Apunte 3 de la cátedra, "Sistemas de Numeración: Operaciones Lógicas".***

Representación de números en Punto Fijo. Operaciones Aritméticas

1. Represente los siguientes números en el sistema BSS restringido a 8 bits. En los casos que no se pueda representar, aclarar por qué.
   0; 1; 127; 128; 255; 256; -1; -8 ; -127; -128; -199; -256; 137; 35; 100; -100; 0,5; 1,25.

   Hay varios métodos, aquí mostraremos 2 de ellos:

   Ejemplo: representar en este sistema el valor 66:

   División:

66 / 2 = 33	Resto 0
33 / 2 = 16	Resto 1
16 / 2 = 8	Resto 0
8 / 2 = 4	Resto 0
4 / 2 = 2	Resto 0
2 / 2 = 1	Resto 0
1 / 2 = 0	Resto 1

   Tomamos los restos de atrás para adelante, rellenando con ceros a izquierda (hasta 8 bits): el número binario que representa al 66 decimal es: **01000010**

   Otro método: considerando potencias de 2, sin excederse del valor original:

   Ejemplo: representar en este sistema el valor 33

   De izquierda a derecha en 8 bits: el primer dígito puede ser 1? Es decir el número binario puede tener esta forma?:

   1XXXXXXX ? NO, porque para obtener el decimal aplicando el teorema de la numeración, sería

1 * 2 elevado a la 7: **128**, valor que excede al 33, por lo tanto el bit más significativo es 0 (cero).

Luego que pasa con el 2do bit? Puede ser 1, es decir el número binario puede tener esta forma?

01XXXXXX? NO

1 * 2 elevado a la 6: **64**, valor que excede al 33, por lo tanto el 2do bit más significativo es 0 (cero).

Luego que pasa con el 3er bit? Puede ser 1, es decir el número binario puede tener esta forma?

001XXXXX? SI

1 * 2 elevado a la 5: **32**, NO EXCEDE el valor 33, por lo tanto el 3er. bit más significativo es 1 (uno).

Se observa que para el resto de los bits excepto el menos significativo, no pueden valer 1, pues

1 * 2 elevado a la i es un valor > 1 para i = 4, 3, 2 o 1. ; y ese valor sumado a 32 excede el valor 33.

Entonces el formato seria 0010000X

Cuanto vale X? 1, pues

1 * 2 elevado a la 0: **1 y sumado al 32 obtenido antes llegamos al valor 33.**

Entonces el número binario es **00100001**

Cuando No se puede representar en el sistema BSS?

1) Si el número es negativo (el sistema BSS no permite representar negativos)
2) Si el número es mayor que el valor máximo representable en dicho sistema.
3) Si el número tiene decimales (el sistema dado no tiene bits para representar la fracción).

Enlaces asociados:

- Introducción al binario
- Conversión Decimal a Binario
- Ejemplo 1, ejemplo 2, ejemplo 3 y ejemplo 4

2. Interprete las siguientes cadenas de 8 bits en el sistema BSS.
   00000000; 01010101; 10000000; 11111110; 11111111; 10101010; 01111111; 01100110

Para resolverlo, aplicar el teorema fundamental de la numeración.

Enlaces asociados:

- Concepto básico: Teorema fundamental de numeración
- Ejemplo 1 y ejemplo 2

3. Calcule el rango y resolución de un sistema de punto fijo en BSS con 6 bits de parte entera y 4 de fraccionaria.

Rango: (valor mínimo representable, valor máximo representable)
Resolución: diferencia entre 2 números representables consecutivos.

Supongamos un sistema de punto fijo en BSS con 4 bits de parte entera y 2 de fraccionaria:
Los números representables tienen la forma siguiente:

XXXX.XX

El valor mínimo es en BSS: 0000.00 valor 0 en decimal.
El valor máximo es en BSS: 1111.11 que valor es?

Aplicando el teorema para la parte entera sería (1 * 2 elevado a la 0) + (1 * 2 elevado a la 1) +…..+ (1 * 2 elevado a la 3) = 1 +2 + 4+ 8 = 15

Aplicando el teorema, para la parte fraccionaria sería (1 * 2 elevado a la -1) + (1 * 2 elevado a la -2) = 0,5 + 0,25 = 0,75

Entonces el valor máximo es 15 + 0,75 = **15,75**

La resolución, es constante en todo el rango de números, pues la diferencia entre cualquier par de números representables consecutivos es siempre la misma: tomemos la resta de los 2 valores mínimos:

0000.01 – 0000.00 = 0000.01
Que valor es en decimal? Por el teorema, 1 * 2 elevado a la – 2 = 0,25

Entonces la resolución en este sistema es **0,25**

Enlaces asociados:

- Binario Sin Signo (BSS) y Binario Con Signo (BSS, BCS)
  - Rango BSS
  - Rango BCS
- Complemento a 1
  - Rango
- Complemento a 2
  - Rango
- Exceso
  - Rango

4. Represente los siguientes números en el sistema del ejercicio 3. Si no es posible obtener una representación exacta, indique cuál es la más próxima y calcule en ese caso el error cometido. Si el número a representar está fuera del rango del sistema, señale que ese número “NO SE PUEDE REPRESENTAR”.

3; 5,25; 1,2; 2,001; 23,125; 62,0625; 1,0625; 35

Ídem ejercicio 1 pero considerando el formato del sistema.

Tener en cuenta que en este formato que un número de valor periódico o irracional no se puede representar.

Ejemplo: 1,2

Parte entera: 000001

Parte fraccionaria: 001100110011.. (periódico)

Cuando se trata de pasar a binario un número fraccionario en lugar de dividir por 2 como se hace con los enteros, se multiplica por 2. Serán sucesivas multiplicaciones que se acotaran según la cantidad de bit con los que contemos para la representación. Las partes enteras de esas multiplicaciones parciales formarán al finalizar el número binario resultante. Las partes fraccionarias parciales son las que se irán multiplicando por 2 sucesivamente.

0,2 * 2 = **0**,4 me quedo con el entero 0 del resultado
0,4 * 2 = **0**,8 me quedo con el entero 0 del resultado
0,8 * 2 = **1**,6 me quedo con el entero 1 del resultado
0,6 * 2 = **1**,2 me quedo con el entero 1 del resultado
0,2 * 2 = **0**,4 …. Comienzan a repetirse los 4 cálculos anteriores

Entonces para la representación de la parte fraccionaria se toman los valores enteros de los resultados parciales, en el orden calculado:

.2 = .0011 0011 .. 0011 (periódico)

Como en el sistema dado tenemos 4 dígitos decimales el número representable más aproximado al 1,2 es:

000001.0011, el VALOR EXACTO 1,2 NO SE PUEDE REPRESENTAR EN EL SISTEMA DADO, se comete un error en la representación

Valor calculado para el sistema dado: 000001.0011 que en decimal es (1 * 2 elevado a la 0) + (1 * 2 elevado a la -8) + (1 * 2 elevado a la -16) = 1 + 1/8 + 1/16 = 1 + 0,125 + 0,0625 = **1.1875**

Error cometido: valor exacto - valor calculado: **1,2 - 1,1875** = **0,0125**

Enlaces asociados:
- Binario Sin Signo (BSS) y Binario Con Signo (BSS, BCS)
  - Rango BSS
  - Rango BCS
- Complemento a 1
  - Rango
- Complemento a 2
  - Rango
- Exceso
  - Rango

5. Interprete las siguientes cadenas en el sistema del ejercicio 3.
0000000000; 0101010101; 1000000000; 1111111110; 1111111111; 1010101010; 0111111111; 0110110110

Para resolverlo, aplicar el teorema fundamental de la numeración

Ejemplo:
0000101000: se asume el punto después de los primeros 6 dígitos más significativos:

000010.1000

Luego, por teorema: (1*2 elevado a la 1) + 1 * 2 elevado a la -1) = **2,5 en decimal**

Enlaces asociados:

- Introducción al binario
- Conversión Decimal a Binario
- Ejemplo 1, ejemplo 2, ejemplo 3 y ejemplo 4

6. Calcule el resultado de realizar las sumas (ADD) y restas (SUB) indicadas a continuación. Calcule el valor en el que quedarán los flags luego de realizada cada operación, de acuerdo a que haya habido acarreo (flag C, de Carry) o se haya producido borrow (flag B, es el mismo que C pero en la resta), o que el resultado sea cero en todos sus bits (flag Z, de Zero).

ADD 00011101 00011011; ADD 01110000 11110001; SUB 00011101 00011011; SUB 01110000 11110001
ADD 10011101 01110010; ADD 01001100 01110000; SUB 10011101 01110010; SUB 01001100 01110000
ADD 01110110 01110001; ADD 11001100 11110000; SUB 01110110 01110001; SUB 11001100 11110000
ADD 10111001 11100011; ADD 10000000 10000000; SUB 10111001 11100011; SUB 10000000 10000000
ADD 00111010 00001111; ADD 00000000 10000000; SUB 00111010 00001111; SUB 00000000 10000000

Recuerde que:

0+0=0 con C=0	1+0=1 con C=0	0-0=0 con B=0	1-1=0 con B=0
0+1=1 con C=0	1+1=0 con C=1	1-0=1 con B=0	0-1=1 con B=1

El acarreo se suma al bit siguiente, y en caso de ser el de mayor peso queda en el flag C:

Ejemplo de Suma:

```
  1111          acarreos
  01001111
+ 01111000
  11000111      Cómo quedan los flags?
```

El flag Zero se pone en 1 cuando el resultado de la operación es 0 (es decir, en este caso, los 8 bits del resultado quedan en 0). Como aquí no ocurrió esto , el flag Zero = 0

El flag de Carry se pone en 1 cuando se produjo “acarreo” en los bits más significativos (en la resta se pone en 1 si hay “borrow” en los bits más significativos. El flag de borrow se identifica con la letra N, de Negativo)

(si la suma de los bits más significativos de ambos números es 1 + 1, el resultado de esta suma dará 0 con un acarreo de un 1 al bit siguiente y entonces el flag de Carry se pondrá en 1)

Volviendo a nuestro ejemplo, no se produjo acarreo y el flag Carry queda en 0.

Enlaces asociados:

- Suma (Parte 1), suma (Parte 2)
- Resta (Parte 1), resta (Parte 2)
- El flag de carry
- flag de carry (continuación)
- El flag de Zero
- El flag de Signo (Negativo)
- El flag de Overflow
- El flag de Parity (adicional a la práctica)
- El flag de Adjust o Auxiliar Carry (adicional a la práctica)

7. Suponga que los operandos del ejercicio anterior (ej. 6) eran números representados en BSS. Determine si el resultado obtenido es correcto. Para eso, interprete en BSS tanto los operandos como el resultado de cada operación y luego compare con el resultado esperado trabajando en decimal. En caso de que la operación haya dado resultado incorrecto, indicar la/s posible/s cadena/s de bits que representa/n el resultado correcto.

1111	acarreos	Interpretación por teorema fundamental
01001111		□ 79 en decimal
+ 01111000		□120 en decimal
11000111		

01001111 por el teorema fundamental □ 79 en decimal
01111000 por el teorema fundamental□120 en decimal

Luego 79 + 120 = 199 que es equivalente al resultado obtenido antes en binario: 11000111, es decir

11000111 (resultado de la suma anterior) por el teorema fundamental□199 en decimal

Por lo tanto el resultado es correcto.

Enlaces asociados:
- Suma (Parte 1), suma (Parte 2)
- Resta (Parte 1), resta (Parte 2)

8. Referido al ejercicio 7 sobre la operación ADD: Observando cuáles resultados fueron correctos y cuáles fueron incorrectos y relacionándolos con los flags, describa una regla para determinar la correctitud de la operación ADD en el sistema BSS con la mera observación de los flags (sin verificar la operación pasando por el sistema decimal).

En el ejemplo anterior, Carry = 0 y Zero = 0

Al resolver los ejercicios, se observará que en los casos en que C = 1 el resultado es incorrecto, independientemente del resto de los flags. RECORDAR: esto en BSS. Indica condición fuera de rango.

Enlaces asociados:
- Suma (Parte 1), suma (Parte 2)
- Resta (Parte 1), resta (Parte 2)

9. Trabaje de forma similar al ejercicio 8 pero con la operación SUB.

1011	acarreos	Interpretación por teorema fundamental
01001111		+ 79 en decimal
- 01111000		+ 120 en decimal
11010111		

01001111 por el teorema fundamental +79 en decimal
01111000 por el teorema fundamental +120 en decimal

11010111 (resultado de la resta anterior) por el teorema fundamental +215 en decimal

Por lo tanto el resultado es incorrecto.

En el ejemplo , Borrow = 1 y Zero = 0

Al resolver los ejercicios, se observará que en los casos en que B = 1 el resultado es incorrecto, independientemente del resto de los flags. RECORDAR: esto en BSS. Indica condición fuera de rango.

ACLARACIÓN ADICIONAL:

Cuando en una operación de suma binaria se produce carry (C=1), hay un 1 que pasa hacia el bit siguiente al bit más significativo.
Cuando en una operación de resta binaria se produce borrow (B=1), hay un 1 que pasa desde el bit siguiente al bit más significativo a dicho bit más significativo.

Enlaces asociados:
- Suma (Parte 1), suma (Parte 2)
- Resta (Parte 1), resta (Parte 2)

10. Considere en el ejercicio 6, que el punto o coma fraccionaria se encuentra entre el bit 2 y el 3. Interprete el valor que tendrán las cadenas de bits que representan los operandos y los resultados como BSS. Observe los flags. ¿Qué concluye?

Enlaces asociados:
- Suma (Parte 1), suma (Parte 2)
- Resta (Parte 1), resta (Parte 2)

11. Escriba las cadenas de los sistemas BSS restringido a 4 bits. Considere el punto (o coma fraccionaria) fijo en cada una de todas las posibles posiciones (son 5 posibilidades en total, considerando que el punto fijo puede estar colocado a la izquierda del MSB y a la derecha del LSB) y obtenga el rango y resolución de cada uno de los sistemas de punto fijo resultantes. ¿Cuántas cadenas se pueden escribir en cada caso? ¿Cuántos números se pueden representar en los distintos sistemas?

Ejemplo: 3 dígitos enteros y 1 fraccionario
XXX.X

Rango: Valor mínimo: 000.0, equivale al 0 en decimal.
Valor máximo: 111.1 equivale al 7,5 en decimal
Entonces el rango es (0; 7,5)

Resolución: diferencia entre 2 números consecutivos, en este caso siempre vale : 000.1, equivale a 0,5.

Resolver cambiando la posición del punto decimal.

Cuantas cadenas se pueden escribir en cada caso? 2 elevado a la cantidad de bits; en 4 bits, 16 cadenas, independientemente de la posición del punto decimal.

Cuántos números se pueden representar? Los mismos que la cantidad de cadenas.

Enlaces asociados:

- Representación de binario con coma fraccionaria
- Rango
- Resolución

12. Represente los números **0, 1, 2, 9, 10, 11, 20, 34, 99, 100 y 1220** en los sistemas BCD y BCD empaquetado. Describa, con el mayor nivel de detalle posible, un procedimiento para calcular sumas en BCD. Sin considerar representación de signo, realice las siguientes operaciones en BCD: **20 + 34; 34 + 99; 1220 + 880**

Para representar una cadena en BCD, (Decimal Codificado Binario) cada dígito decimal tiene su equivalente en 4 dígitos binarios:

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

Las demás representaciones de 4 dígitos binarios no tienen equivalente en BCD (ellas son: 1010=A=10, 1011=B=11, 1100=C=12, 1101=D=13, 1110=E=14, 1111=F=15)

BCD empaquetado: se reemplaza cada dígito por los 4 dígitos binarios equivalentes

Ejemplo: 834: 1000 0011 0100 (BCD Empaquetado: 4 bits por dígito)

BCD empaquetado con signo: al valor en binario se le agregan 4 dígitos binarios que indican el signo del número:
C 1100 Valor positivo
D 1101 Valor negativo

Ejemplos:

+ 834: 1000 0011 0100 **1100** = 834C
- 834: 1000 0011 0100 **1101** = 834D

BCD desempaquetado: se reemplaza cada dígito por los 8 dígitos binarios, los 4 menos significativos corresponden al número binario equivalente, y los 4 más significativos son "unos":

Ejemplo: 834: 1111**1000** 1111**0011** 1111**0100** = F8F3F4 (BCD Desempaquetado: (8 bits por dígito)

BCD desempaquetado con signo: a diferencia del anterior, el signo del número se carga en lugar de los "unos" del rellenaban el byte menos significativo:

C 1100 Valor positivo
D 1101 Valor negativo

Ejemplos:

		(+/-)		
+ 834: 1111	**1000**	1111 **0011**	1100 **0100**	= F8F3C4
- 834: 1111	**1000**	1111 **0100**	1101 **0100**	= F8F3D4

Suma en BCD: para la suma en BCD se suma en binario, y puede ocurrir que se obtenga una representación no válida en BCD.

Ejemplo: 15 + 27; en decimal nos da 42.

Que pasa en BCD?

15	0001 0101
27	0010 0111
La suma da	0011 1100

Pero 1100 no es una representación válida en BCD; es una de las 6 combinaciones de 4 bits no válidas 1100=C=12)

La regla es la siguiente:

Si la suma de 2 dígitos BCD <=9 OK
Si la suma de 2 dígitos BCD >9 => Sumar 6 en ese dígito (o sea sumar 0110)

Entonces

15	0001 0101	
27	0010 0111	
La suma da	0011 1100	
+	0110	+ 6
	0100 0010	

El resultado es 42 (Correcto)

13. Escriba los números **21398, 2183, 972 y 89737** en los sistemas BCD, BCD empaquetado y BSS. Observe la cantidad de bits necesarios. ¿Qué conclusiones saca respecto de las ventajas y desventajas del sistema BCD sobre BSS?

14. Haga el pasaje de binario a hexadecimal y de hexadecimal a BCH en forma directa (sin utilizar sistema decimal). ¿Por qué cree que el sistema hexadecimal es muy utilizado?

Binario a Hexadecimal	
1001010010000	
11010010101011	
101011011001101	
1001111000100011	
1100101011101010	
11100101011011	

Hexadecimal a BCH	
1290	
34AB	
56CD	
E7F8	
8D71	
123B	

Para pasar de **Binario a Hexadecimal** sin utilizar el sistema decimal se toman de a 4 digitos binarios de derecha a izquierda y se reemplaza cada número formado por los 4 dígitos binarios por el número hexadecimal correspondiente:

Ejemplo: **101011011001101**

Separando de a 4 dígitos:

101 0110 1100 1101

Reemplazando los 4 dígitos binarios por el hexadecimal correspondiente:

101 0110 1100 1101 = 5 6 C D

Entonces el equivalente al número binario **101011011001101** en hexadecimal es **56CD**

Para pasar de **Hexadecimal** a **BCH (Hexadecimal codificado binario)** sin utilizar el sistema decimal se realiza la operación inversa a la anterior, es decir se reemplaza cada dígito hexadecimal por los 4 dígitos binarios que conforman su valor equivalente:

Ejemplo: **56CD**

Reemplazando cada dígito hexadecimal por los 4 dígitos binarios:

5 6 C D = 0101 0110 1100 1101 (completado con ceros a izquierda)

Luego **0101011011001101** es la representación en BCH del número hexadecimal 56CD

Enlaces asociados:

- Hexadecimal a decimal
- Hexadecimal a Binario
- Binario a Hexadecimal

Operaciones Lógicas

Enlaces asociados:

- Operación NOT
- Operación AND
- Operación OR
- Operación XOR
- Operaciones NAND, NOR y XNOR

15. Realizar las siguientes operaciones lógicas:

a.	**10101100 AND 11000101**	f.	**00100010 XOR 11111101**	k.	**10101110 NOR 11010101**
b.	**00100010 AND 11111101**	g.	**NOT 00101100**	l.	**00101010 NOR 11100101**
c.	**10101100 OR 11000101**	h.	**NOT 11000101**	m.	**10101100 XNOR 11000101**
d.	**00100010 OR 11111101**	i.	**10101011 NAND 11000101**	n.	**10101100 XNOR 11111101**
e.	**10101100 XOR 11000101**	j.	**00100010 NAND 11111101**		

Las operaciones lógicas se realizan bit a bit, es decir si tenemos la que hacer una operación lógica (OL) como:

$$\begin{array}{r} X_7\ X_6\ X_5\ X_4\ X_3\ X_2\ X_1\ X_0 \\ \text{OL}\ \underline{Y_7\ Y_6\ Y_5\ Y_4\ Y_3\ Y_2\ Y_1\ Y_0} \\ Z_7\ Z_6\ Z_5\ Z_4\ Z_3\ Z_2\ Z_1\ Z_0 \end{array}$$

A cada valor Xi se le aplica la operación lógica OL con el correspondiente Yi' obteniéndose un valor Zi

Luego las tablas de verdad para las operaciones lógicas básicas son:

X	Y	X and Y	X or Y	X xor Y
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

Ejemplo:

10101100
AND 11000101
10000100

16. Dado un byte **X=[X_7,X_6,X_5,X_4,X_3,X_2,X_1,X_0]** (indeterminado), ¿qué resultado obtendré al aplicarle una operación lógica junto a un valor predeterminado (máscara)?: Analice para cada operación cómo los bits de la 'máscara' condicionan el resultado que se obtendrá.
    a. **X OR 10101010**
    b. **X OR 11111000**
    c. **X AND 10101010**
    d. **X AND 10001111**
    e. **X XOR 10101010**
    f. **X XOR 00001111**
    g. **X OR 10000000**, al resultado **AND 11110000**, y al resultado **XOR 00011110**
    h. **X AND 10101111**, al resultado **OR 11110000**, y al resultado **XOR 00011110**
    i. **X XOR 10101010**, al resultado **AND 11110000**, y al resultado **OR 00011110**
    j. **X XNOR 10101010**, al resultado **NAND 11110000**, y al resultado **NOR 00011110**
    k. **X XOR 10101010**, al resultado **NAND 11110000**, y al resultado **NOR 00011110**

En los casos de más de una operación, obtenga el resultado y a ese resultado aplíquele la operación siguiente.

Las máscaras sirven para distintos acciones; algunas de ellas son:

1) Forzar un determinado bit a un valor 0 o 1;
2) Mantener un determinado bit con valor igual al actual
3) Invertir el valor de un determinado bit, de 0 a 1 o de 1 a 0, dependiendo del valor anterior.

Para cumplir esto, las máscaras se utilizan en conjunto con las operaciones lógicas.
Tengamos en cuenta las siguientes reglas:

Operación lógica AND

Xi AND 0 = 0, pues

Si Xi = 0 => 0 AND 0 = 0
Si Xi = 1 => 1 AND 0 = 0

Xi AND 1 = Xi, pues

Si Xi = 0 => 0 AND 1 = 0 = Xi
Si Xi = 1 => 1 AND 1 = 1 = Xi

Entonces con la operación lógica AND podemos forzar un valor a 0 o mantener el valor anterior.

Operación lógica OR

Xi OR 0 = Xi, pues

Si Xi = 0 => 0 OR 0 = 0 = Xi
Si Xi = 1 => 1 OR 0 = 1 = Xi

Xi OR 1 = 1, pues

Si Xi = 0 => 0 OR 1 = 1
Si Xi = 1 => 1 OR 1 = 1

Entonces con la operación lógica OR podemos forzar un valor a 1 o mantener el valor anterior.

Operación lógica XOR

Xi XOR 0 = Xi, pues

Si Xi = 0 => 0 XOR 0 = 0 = Xi
Si Xi = 1 => 1 XOR 0 = 1 = Xi

Xi XOR 1 = -Xi, (valor opuesto) pues

Si Xi = 0 => 0 XOR 1 = 1 = -Xi (valor contrario, pasa de 0 a 1)
Si Xi = 1 => 1 XOR 1 = 0 = -Xi (valor contrario, pasa de 1 a 0)

Entonces con la operación lógica XOR podemos forzar un valor al contrario o mantener el valor anterior.

Ejemplo:

X OR 11111000: **XXXXXXXX**
OR 1 1 1 1 1 0 0 0
1 1 1 1 1 XXX (En base a la explicación anterior)

Enlaces asociados:
- Introducción a las Máscaras
- Máscaras - Como forzar ceros
- Máscaras - Como forzar unos
- Máscaras - como forzar unos (continuación)
- Máscaras - Como forzar distintos (negados)
- Ejemplo
- Máscaras con NAND, NOR, XNOR

17. Complete con el operador lógico adecuado (AND, OR, XOR, NOT) las siguientes expresiones de modo tal que se cumpla la igualdad propuesta:
    a. **1000 ...¿op?... 1101 = 1101**
    b. **1111 ...¿op?... 0101 = 0101**
    c. **1101 ...¿op?... 1001 = 0100**
    d. **....¿op?... (1111 ...¿op?... 0011) = 1100**
    e. **X_3 X_2 X_1 X_0 ...¿op?... 1110 ..¿op?... 0101 ...¿op?... 0101 = X_3 0 X_1 0**

Se entiende que cada X es un bit que puede ser 1 o 0, debiendo obtenerse el resultado final combinando diferentes operaciones lógicas en un orden correcto.

Ejemplo:

$$\begin{array}{rr} & 1\ 0\ 0\ 0 \\ \text{¿op?} & 1\ 1\ 0\ 1 \\ \hline & 1\ 1\ 0\ 1 \end{array}$$

Cual es la operación lógica ¿op? Se observa que la operación está forzando valores a 1 y otros se mantienen igual => la op es OR

18. Dado un byte **X=[X_7,X_6,X_5,X_4,X_3,X_2,X_1,X_0]** (indeterminado), aplíquele operaciones lógicas (1 o más) con un byte MASK, que deberá también determinar, para lograr los siguientes efectos:

a. Poner a 1 los bits 0, 2 y 6 dejando los demás inalterados.
b. Poner a 1 los bits 4 y 7 dejando los demás inalterados.
c. Poner a 0 los bits 0, 2 y 6 dejando los demás inalterados.
d. Poner a 0 los bits 4 y 7 dejando los demás inalterados.
e. Cambiar los bits 0, 2 y 6 dejando los demás inalterados.
f. Cambiar los bits 4 y 7 dejando los demás inalterados.
g. Poner el bit 3 en 1, el bit 6 en 0, cambiar el bit 2 y dejar los demás inalterados.
h. Poner a 0 los bits 0, 3 y 7, cambiar el bit 2 y dejar los demás inalterados.

Ejemplo: Poner el bit 3 en 1, el bit 6 en 0 cambiar el bit 2 y dejar los demás inalterados

Para ello tenemos que hacer 3 operaciones lógicas:

1) Sabemos que la operación OR permite forzar un bit a 1, donde la máscara posea en dicho bit un 1, e inalterado donde la máscara posea en dicho bit un 0. Entonces para forzar el bit 3 a 1, le aplicamos la siguiente máscara.

$$\begin{array}{rr} & X_7\ X_6\ X_5\ X_4\ X_3\ X_2\ X_1\ X_0 \\ \text{OR} & 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0 \\ \hline & X_7\ X_6\ X_5\ X_4\ X_3\ \ 1\ X_1\ X_0 \end{array}$$

2) Al resultado anterior le tenemos que forzar el bit 6 a 0, para eso sabemos que debemos usar la operación lógica AND donde la máscara posea en el bit 6 un 0, y el resto de los bits en 1 para que el resultado de dichos bits quede inalterado.

$$\begin{array}{rr} & X_7\ X_6\ X_5\ X_4\ X_3\ \ 1\ X_1\ X_0 \\ \text{AND} & 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1 \\ \hline & X_7\ X_6\ 0\ X_4\ X_3\ \ 1\ X_1\ X_0 \end{array}$$

3) Finalmente tenemos que cambiar el valor del bit 2; para ello se utiliza la operación lógica XOR que con máscara 1 en dicho bit invierte su valor, y con valor 0 en los restantes bits de la máscara para que mantenga los demás valores del resultado inalterados:

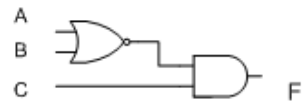
$$\begin{array}{rr} & X_7\ X_6\ 0\ X_4\ X_3\ \ 1\ X_1\ X_0 \\ \text{XOR} & 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0 \\ \hline & X_7\ X_6\ 0\ X_4\ X_3\ \ 1\ \text{-}X_1\ X_0 \end{array}$$

(-Xi ,indica valor opuesto a Xi)

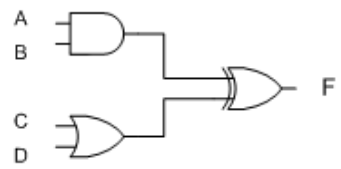
Circuitos Combinatorios

19. Construir la tabla de verdad de los siguientes circuitos. Especifique además la ecuación que describe la relación entre entradas-salidas.

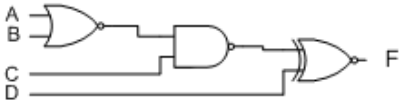
1)



2)



3)



Para identificar cada circuito por separado ver los videos correspondientes en la plataforma.

Primero debemos obtener la función F asociada al circuito; En el ejemplo 1, la salida F depende de 2 entradas al circuito AND a las que llamaremos X e Y, entonces:

$F = X \,.\, Y$

Luego debemos reemplazar X e Y por lo que ellos representan, X está representado por un circuito NOR con entradas A y B, e Y es la entrada C:

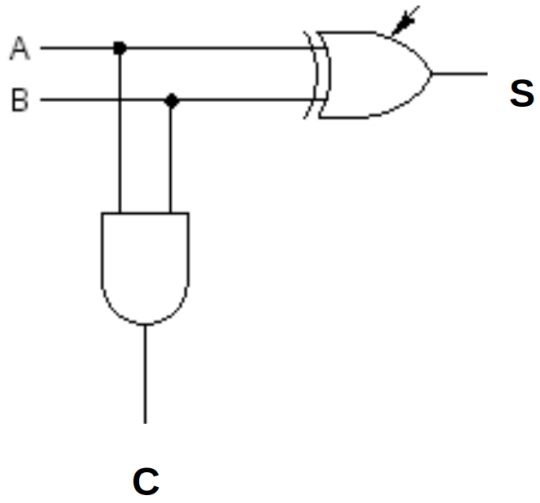
$X = \overline{A + B}$
$Y = C$

Reemplazando: $F = \overline{(A + B)} \,.\, C$

Los paréntesis permiten visualizar claramente el orden de precedencia de las operaciones lógicas. Hemos obtenido la función F, ahora podemos hacer la tabla de verdad:

A	B	C	A + B	- (A + B)	F
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

Recordar generar cada columna en base a las calculadas anteriormente. Porque son 8 filas de valores de entrada? Porque son 3 entradas con 2 valores posibles, las combinaciones son siempre 2 a la N, con N número de entradas.

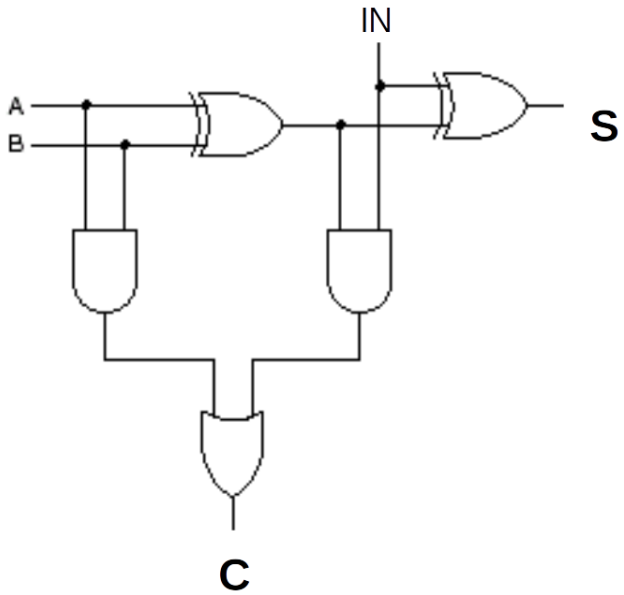


Este ejemplo representa un sumador de un bit. Las entradas son A y B, las salidas son: S y C.

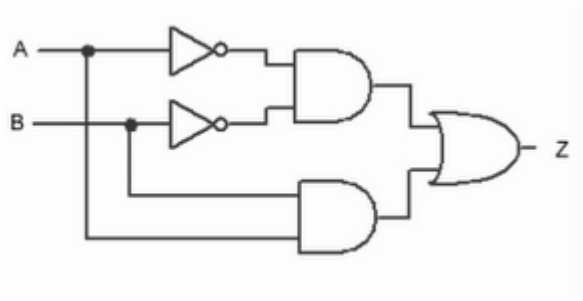
La salida S (XOR) "representa" la suma mientras que la salida C (AND) "representa" el Carry de este circuito sumador de 1 bit.

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

19.5)



19.6)



19.7)