COC – PRÁCTICA 3

(Ejercicios resueltos)

1)

a) 00010001 **AND** 01011100 = 00010000
b) 01010101 **AND** 01010101 = 01010101
c) 01010101 **AND** 10101010 = 00000000
d) 11110000 **AND** 11111111 = 11110000
e) 01010101 **OR** 01010101 = 01010101
f) 01010101 **OR** 10101010 = 11111111
g) 11110001 **OR** 11110010 = 11110011
h) 01010101 **XOR** 01010101 = 00000000
i) 01010101 **XOR** 10101010 = 11111111
j) 00001111 **XOR** 00000000 = 00001111
k) **NOT** 11111111 = 00000000
l) **NOT** 01000000 = 10111111
m) **NOT** 00001110 = 11110001

Para realizar este ejercicio es necesario tener a mano las tablas de verdad de cada puerta lógica (se encuentran entre las páginas 44 y 48 del apunte de la materia) y tener en cuenta que la operación se hace de la forma:

Ej.: 00010001
AND 01011100
00010001

2)

DATO	OP. LÓGICA	MASK	=	RESULTADO
$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	OR	1 1 1 0 0 1 1 1	=	1 1 1 $D_4\ D_3$ 1 1 1
$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	OR	0 0 0 0 1 0 0 0	=	$D_7\ D_6\ D_5\ D_4$ 1 $D_2\ D_1\ D_0$
$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	AND	0 1 1 1 1 1 1 1	=	0 $D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$
$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	XOR	0 1 0 1 0 0 0 0	=	$\overline{D_7}\ \overline{D_6}\ D_5\ \overline{D_4}\ D_3\ D_2\ D_1\ D_0$

(La línea arriba del bit significa que está negado)

Para entender el ejercicio hay que pensarlo desde el lado de “qué pasa si me entra el valor 1/0 en un circuito AND/OR/XOR”...
Ej.: en un circuito OR sabemos que si una entrada “A” tiene el valor 1 no nos interesa saber qué valor tendrá la entrada “B” porque, sea 0 o 1 el valor de esa entrada “B”, el resultado va a ser de todas formas 1 (ya que basta con que una entrada sea 1 para que el resultado sea 1). Lo mismo pasa con el circuito AND, con que una entrada sea 0 basta para saber que la salida va a ser 0.

- Hay que tener en cuenta que en los circuitos XOR o XNOR no podemos determinar si la salida será 0 o 1 pero sí podemos determinar si va a ser el mismo valor de entrada o si será el valor de entrada negado.
  Ej.: en un circuito XOR tengo una entrada “A” que vale 1 y no sabemos el valor de nuestra entrada “B”, si analizan la tabla de verdad se van a dar cuenta que el resultado siempre va a ser la entrada “B” negada (ya que si nuestra entrada “B” es un 1 el resultado va a ser 0 y en caso de que sea un 0 el resultado va a ser 1).

3)

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

a)

Los resultados de estos circuitos se pueden asociar con un semisumador (half-adder), es decir un circuito que tiene dos bits de entrada y genera como salida:

- un bit que representa la suma de los dos bits de entrada
- otro bit que representa el acarreo generado por la suma.

Siendo la entrada “A” y “B” los sumandos, la salida “D” el resultado de la suma y la salida “C” el acarreo.

b)

A	B	IN	A AND B	A XOR B	(A XOR B) AND IN	D	C
0	0	0	0	0	0	0	0

0	0	1	0	0	0	1	0
0	1	0	0	1	0	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	1	0
1	0	1	0	1	1	0	1
1	1	0	1	0	0	0	1
1	1	1	1	0	0	1	1

Los resultados de estos circuitos se pueden asociar con un sumador completo (full-adder), que, a diferencia del semisumador, éste tiene bit que representa un acarreo de entrada.

Siendo la entrada “A” y “B” los sumandos, la entrada “IN” el acarreo de entrada”, la salida “D” el resultado de la suma y la salida “C” el acarreo de salida.

4) a) Si, es posible.

b)

- Para la NOT: hacer una puerta NAND donde sólo tendrá una entrada “A” que hará entrar el mismo valor para lograr la función NAND. Por lo que la tabla de verdad sería de esta forma:

A	A	A NAND A
0	0	1
1	1	0

  Como se ve en la tabla de verdad, el valor de entrada de “A” está negado a la salida, logrando la función de la NOT.

- Para la OR: hacer dos puertas NAND con una sola entrada cada una donde la salida de éstas serán la entrada de una tercera puerta NAND. La tabla de verdad sería esta:

A	A	B	B	A NAND A	B NAND B	(A NAND A) NAND (B NAND B)
0	0	0	0	1	1	0
0	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	1	1	0	0	1

  Como se ve en la tabla de verdad, sólo cuando el valor de entrada de “A” y de “B” es 0 la salida es 0, logrando la función de la OR.

- Para la AND: hacer una puerta NAND con una entrada “A” y una entrada “B” donde la salida de esta será la entrada de otra puerta NAND. La tabla de verdad:

A	B	A NAND B	A NAND B	(A NAND B) NAND (A NAND B)
0	0	1	1	0
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

Como se ve en la tabla de verdad, sólo cuando el valor de entrada de “A” y “B” es 1 la salida es 1, logrando la función de la AND.

5)

A	B	A XOR B	(A XOR B) NOR (A XOR B)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

(ESTA SERÍA LA SALIDA

F)

La salida “F” es igual a 1 sólo cuando sus valores de entrada son iguales.

8)

DECIMAL	BINARIO	HEXADECIMAL
27	11011	1B
54	110110	36
108	1101100	6C
542	1000011110	21E
1084	10000111100	43C
2013	11111011101	7DD
2168	100001111000	878

9)

a) $1000111101010_{(2}$ = $4586_{(10}$

b) $10100111001111000_{(2}$ = $85624_{(10}$

c) $FECB_{(16}$ = $65227_{(10}$

d) $1B2C_{(16}$ = $6956_{(10}$

10)

DECIMAL	BINARIO	HEXADECIMAL
5689	1011000111001	1639
896	1110000000	380
713	1011001001	2C9

11)

	RESULTADO	ZNVC	INTERPRETADOS COMO SIN SIGNO	OK?	INTERPRETADOS COMO CA2	OK?
a)	$10011111_{(2}$	0110	$112 + 47 = 159_{(10}$	SI	$112 + 47 = -97_{(10}$	NO
b)	$10000000_{(2}$	0110	$64 + 64 = 128_{(10}$	SI	$64 + 64 = -128_{(10}$	NO
c)	$00000000_{(2}$	1001	$255 + 1 = 0_{(10}$	NO	$-1 + 1 = 0_{(10}$	SI
d)	$10000000_{(2}$	0110	$127 + 1 = 128_{(10}$	SI	$127 + 1 = -128_{(10}$	NO
e)	$11111101_{(2}$	0101	$255 + 254 = 253_{(10}$	NO	$-1 + (-2) = -3_{(10}$	SI
f)	$00000011_{(2}$	0001	$192 + 67 = 3_{(10}$	NO	$-64 + 67 = 3_{(10}$	SI
g)	$10000000_{(2}$	0101	$192 + 192 = 128_{(10}$	NO	$-64 + (-64) = -128_{(10}$	SI
h)	$10001111_{(2}$	0101	$159 + 240 = 143_{(10}$	NO	$-97 + (-32) = -113_{(10}$	NO
i)	$10000000_{(2}$	0111	$127 - 255 = 128_{(10}$	NO	$127 - (-1) = -128_{(10}$	NO
j)	$00000000_{(2}$	1000	$143 - 143 = 0_{(10}$	SI	$-113 - (-113) = 0_{(10}$	SI
k)	$11111000_{(2}$	0101	$112 - 120 = 248_{(10}$	NO	$112 - 120 = -8_{(10}$	SI