ASIGNATURA: ARQUITECTURA DE COMPUTADORAS
PROFRA. ING. ROCÍO ROJAS MUÑOZ

Sistemas Numéricos

1.-Sistema Numérico.

a) **Definición:** Llamaremos sistema numéricos base "M" el conjunto de "M" símbolos que nos sirven para representar cantidades numéricas.

b) **Representación de un número en base "M":** La representación de un número de base "M" estará en función del número de posiciones disponibles que tenga en mí sistema. El mayor número "A" que puedo representar en un sistema base "M" con "N" posiciones será igual a:

$$A = M^N - 1$$

por otro lado para representar un número cualquier "A" en un sistema base "M" necesito obtener el número "N" de posiciones: N = Log (1+ A) / Log (M).

c) **Conversión de un número base "M" a base 10:**

$$(b_n\ b_{n-1}\ b_{n-2} \ldots\ldots\ldots\ldots b_0, b_{-1}\ b_{-2} \ldots\ldots\ldots b_{-p})_m = \sum_{i=-p}^{n} b_I\ m_i$$

Ejemplo: Convertir $(325.14)_6$ a base 10:

d) **Conversión de base 10 a base "M":**

Tenemos que:

i. Parte Entera:

M Ae		↑
Co	Ro	
C1	R1	
		lectura
Cn-1		
O	Rn = Cn-1	

C = cociente
R = residuo

ii. Parte Fraccionaria:

AF *Los E's determinan el número decimal.
x m.
E1.F1
x m.
E2.F2

Ejemplo:

- Convertir el número 426.78 a base 7.
- Convertir 0.75 a base 5.
- Convertir 0.33 a base 10.
- Convertir 0.3333 a base 10.

Podemos observar, que al convertir de un sistema numérico a otro podemos caer en errores que son inherentes, debido al número de posiciones que podemos manejar, cabe pensar que si tenemos un número infinito de posiciones no existirá error.

e) Conversión Octal, Hexadecimal, Binario.

$(1000)_{10} \longrightarrow (100)_{100}$

i. **Conversión de binario a Octal:** Para convertir un número binario a base 8, agrupamos de tres en tres bits a la izquierda y a la derecha del punto y obtenemos el correspondiente valor octal de cada grupo.

Ejemplo: Convertir $(11001101.11011)_2 \longrightarrow (315.66)_8$

ii. **Conversión de binario a Hexadecimal:** Se toman grupos de 4 en 4 de bits a la izquierda y a la derecha del punto y obtenemos el correspondiente valor octal en cada grupo.

Ejemplo: Convertir $(11001101.11011)_2 \longrightarrow (CD.D8)_{16}$

iii. **Conversión de octal a binario:** Para convertir un número octal a binario tenemos que representar cada "dígito" octal con su correspondiente binario utilizando tres posiciones (tres bits).

Ejemplo: Convertir el octal $(476.235)_8 \longrightarrow (100111110.001011101)_2$

iv. **Conversión de Hexadecimal a binario:** Representemos cada "dígito" Hexadecimal con su correspondiente binario utilizando cuatro posiciones.

$(AC3.D2F)_{16} \longrightarrow (101011000011.110100101111)_2$

2.-Representación de números signados:

a) Complemento: Es lo que le falta a ese número pasa llegar a la siguiente o siguientes potencias de la base, por ejemplo: Obtener el complemento de 37 en un sistema de 2 posiciones:

$1x10^2 = 100$

100

-37.

63 ⟶ Complemento

b) **Complemente a dos:** Es lo que le falta a un número binario para llegar a la siguiente o siguientes potencias del 2.

(Se convierten 0 a 1 y 1 a 0 y se les suma 1).

c) **Representación de números signados:** Para representar un número signado tenemos que:
   1. Sí el número es positivo (A>0): Representamos el número A utilizando n-1 bit (donde n es el número de bits del sistema), hacemos el MSB (Bit más significativo) igual acero.
   2. Sí el número es negativo (A<0): Representamos "A" utilizando n-1 bit, hacemos el MSB igual a cero y por último obtenemos el complemento a dos de los n bits.

Ejemplo: Representar el +10, -10 en un sistema de 5 bits.

3.-Operaciones Binarias:

a) **Suma y Resta Binaria:** No existe la resta, sólo existe la suma de símbolos signados. Para realizar una suma de números signados, basta con representar el número signado y sumarlos, pero hay que considerar la condición de overflow.
   i. **Condición de Overflow:** Sí entra y no sale o sí sale y no entra un bit de acarreo al bit de signo. (OV).

Ejemplo: Para un sistema de 4 bits realizar las operaciones correspondientes.

b) **Multiplicación:** La multiplicación de números en binario se realiza con los valores absolutos, sí los números eran de signos diferentes se realiza el complemento a dos del resultado utilizando un bit más.

4.-Códigos.

a) **Código.**

Codificar ≠ Convertir

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

9	1001

i. **Código exceso de 3:** Utiliza 4 bits para representar los dígitos decimales utilizando su representación de Binario comenzando por 3.

Decimal	Ex -3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Ejemplo: Codificar 327 a Ex -3:

0110 0101 1010

ii. **Código 2 de 5.** Consiste en utilizar 5 bits para representar cada dígito decimal en donde cada dígito tiene 2, 1's y tres 0's.

Decimal	2 en 5
0	00011
1	00101
2	00110
3	11000
4	01010
5	10010
6	10001
7	00110
8	01010
9	10100

Este código se utiliza en algunos sistemas de transmisión, para facilitar la detección de errores.

iii. **Código Gray:** Utiliza 4 bits para representar cada dígito decimal, sólo que en la tabla de código cada número consecutivo, varia en un solo bit.

Decimal	Gray	Gray reflejado
0	0000	0000
1	0001	0001

2	0101	0011
3	1001	0010
4	1010	0110
5	1100	0111
6	1110	0101
7	1101	0100
8	1011	1100
9	0111	1101

iv. Código Gray reflejado: Es igual que el anterior, pero este sí tiene una forma Standard de construcción:

0000
0001
0011
0010
0110
0111
0101
0100
1100
1101

---- Es un espejo que se refleja.

Podemos observar que sí seguimos reflejando obtendríamos cualquier número en Gray de tal forma que podríamos estar hablando de un sistema numérico Gray reflejado. Sin embargo la obtención de números muy grandes sería muy laboriosa. ¡Ah!, pero hay un truco, podemos utilizar un método alternativo:

- Para convertir un número a sistema Gray reflejado tenemos que:
  a) Obtener el número en binario.
  b) Agregar un 0 a la izquierda del número.
  c) Aplicar: $G0 = B0 \oplus B1$
     $G1 = B1 \oplus B2$
     $G2 = B2 \oplus B3$ $\oplus$ = XOR

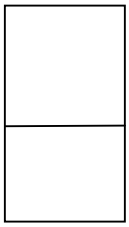
     $Gn = Bn \oplus Bn+1$ ------- Función XOR

x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

B ---- Representan los bits del número en binario.
G ---- Bits en Gray.

v. **Código de 7 segmentos:** Este código es un código visual binario de representación decimal (dígitos de un reloj) y esta formado por un arreglo de 7 segmentos.

Decimal	Segmento	F1	F2	F3	F4	F5	F6	F7
0	----------	x	x	x	x	x	x	-
1	----------	-	x	x	-	-	-	-
2	----------	x	x	-	x	x	-	x
3	----------	x	x	x	x	-	-	x
4	----------	-	x	x	-	-	x	x
5	----------	x	-	x	x	-	x	x
6	----------	x	-	x	x	x	x	x
7	----------	x	x	x	-	-	-	-
8	----------	x	x	x	x	x	x	X
9	----------	x	x	x	x	-	x	X



a) **Códigos Alfanuméricos:** Nos sirven para representar caracteres (letras, números, símbolos y en algunos casos, señales de control).

i. **Código ASCII:** Utiliza 7 bits y puede representar hasta 128 caracteres.
ii. **Código EBCDIC:** Utiliza 8 bits y puede representar hasta 286 caracteres.
iii. **Código Hollerith:** Utiliza 12 bits y puede representar 4096 caracteres.
iv. **Códigos internos de Máquinas:** Son de 6 bits y son utilizados en máquinas grandes y pueden representar 64 caracteres.

Ejemplo: Representar la palabra “deba” (DEBA).

1000100	1000101	1000010	1000001
D	E	B	A

b) **Códigos de detección de errores:** En un sistema de transmisión entre más larga y ruidosa sea la transmisión mayor será la probabilidad de error, por lo tanto deben de existir métodos de detección y corrección de errores.

i. **Método de la paridad:** Antes de mencionar en que consiste el método es necesario conocer que es el bit de paridad. El bit de paridad es un bit que se agrega a la izquierda de cada carácter codificado y puede valer 0 ó 1, dependiendo de la paridad.

Paridad: Consiste en transmitir por cada carácter codificado un número determinado de 1’s, y estos podrán ser: un número par ó un non dependiendo del sistema.

Ejemplo: Definir el bit de paridad para la palabra “DEBA” en un sistema de transmisión de paridad “par”.

01000100 **1**1000101 **0**1000010 **0**1000001

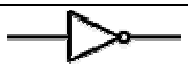
Bit de paridad

Este método reduce el 50% de error en transmisiones, después es posible aplicar métodos subsecuentes de detección y corrección hasta tener confiabilidad de más del 95%.

Tema III

“Álgebra Booleana y Compuertas Lógicas”

1.-Resumen de Compuertas Lógicas:

Nombre	Símbolo	Función	Tabla de Verdad
AND		F=(x)(y) F=xy F=x y	x y F 0 0 0 0 1 0 1 0 0 1 1 1
OR		F=x+y F=x y	x y F 0 0 0 0 1 1 1 0 1 1 1 1
NOT (inversión)		F=x F=x’	x y F 0 1 1 0
BUFFER		F=x	x F 0 0 1 1
NAND		F=(x)(y) F=x+y	x y F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		F=x+y F=(x)(y)	x y F 0 0 1

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>XOR (OR exclusivo)
(Función NON)</td><td>  </td><td>F=x´y+xy´
F=x⊕y</td><td>
<table>
<tr><td>x</td><td>y</td><td>F</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
*Sólo es válida cuando es 1’sde ellos.
</td></tr>
<tr><td>XNOR (Función PAR)</td><td>  </td><td>F=xy+x´y´
F=(x)(y)</td><td>
<table>
<tr><td>x</td><td>y</td><td>F</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

2.-Teoremas fundamentales del Álgebra de Boole:

1. a + 0 = a
2. a * 1 = a
3. a + 1 = 1
4. a * 0 = 0
5. a + a = a
6. a * a = a
7. a + a´ = 1
8. (a´) = a´
9. a * a´ = 0
10. a (a +b) = a
11. a + ab = a
12. a (a´+b) = ab
13. a + a´b = a + b
14. ab + ac = a (b+c)
15. (a+b) (a+c) = a + bc
16. ab + ac + a´c =ab + c
17. (a+b) (a´+c) = ac + a´b
18. (a´+b´+c´) = a´ * b´ * c´ } Teorema de
19. (a´*b´*c´) =a´ + b´ +c´ } Morgan
20. a´´ = a

Ley Conmutativa:
a+b=b+a; a*b=b*a

Ley Distributiva:
a+ (b*c)=(a+b)*(a+c)
a*(b+c)= (ab)+(ac)

+ Unión.
* Intersección.
a´´ = a Complemento.

1.- a+0=a
2.-a*1=a

3.-Álgebra de Boole.

a) **Definiciones:**
   i. **Función Booleana:** Se define una función Booleana como: K= (0,1).
   ii. **Variable Booleana:** Sean X, X_2, X_n símbolos denominados variables, las cuales pueden valer 0 ó 1.
   iii. **Función f(X_1, X_2,…X_n):** Se define como función de las variables booleanas X_1, X_2,…, X_n a la función booleana que puede valer 0 ó 1 dependiendo del valor de las variables y de la relación entre ellas.
   iv. **Implementación:** Cualquier función booleana puede ser implementada con dispositivos físicos.
   v. **Tabla de vedad:** Cualquier función booleana puede ser representada en una tabla de verdad en donde se expresen los valores de la función booleana dependiendo del valor de las variables.
   vi. **Formas algebraicas:** Cualquier función Booleana se puede representar de dos formas:

- Forma Standard.
- Forma Canónica.

***) Forma Standard:** La forma Standard puede tener dos notaciones.

- **Suma de productos (SP):** Las funciones SP están constituidas por OR's ó disyunciones de términos producto que a su vez están constituidos por AND's ó conjunciones de las variables Booleanas, las cuales pueden estar complementadas ó sin complementar.
- **Producto de sumas (PS):** Las funciones PS están constituidas por AND's ó conjunciones de términos suma que a su vez están constituidas por OR's ó disyunciones de las variables Booleanas, las cuales pueden estar complementadas ó no.

****) Forma Canónica:** La forma canónica puede tener dos notaciones:

- **Suma de Productos:** Una función Booleana expresada en forma canónica suma de productos es aquella en cada término producto contiene todas las variables de la función booleana ya sean complementadas ó sin complementar, a este término producto se le llama "Míntérmino".
- **Producto de Sumas:** Una función Bolean expresada en forma canónica producto de sumas es aquella en cada término suma contiene todas las variables de la función booleana ya sean complementadas o no, a este término se le llama Máxtermino.

   vii. **Obtención de la función booleana partiendo de la tabla de verdad.**

Sí una función booleana la queremos obtener en mintérminos tenemos que, de la tabla de verdad:

$f = \sum$ de los mintérminos para los cuales f =1.

y para una función de máxterminos, tenemos que de la tabla de verdad:

$f = \pi$ de los máxterminos, para las cuales f = 0. π =producto

En una tabla de verdad tenemos representados todos los mintérminos y máxterminos posibles, esto es, para un espacio Booleana de "n" dimensiones.

viii. **Reducción de funciones booleanas:** Reducir una función booleana es representarla en su mínima expresión. Para reducir una función booleana podemos utilizar álgebra de Boole.

ix. **Implementación de funciones booleanas:** Cualquier función booleana puede ser implementado por circuitos lógicos.

***) Implementación con interruptores.**

****) Implementación con compuertas.**

Tema IV

Minimización

1.-Introducción: Minimizar consiste en reducir una función booleana a su máxima expresión con el fin de optimizar el número de compuertas o de circuitos lógicos. Existen tres métodos para minimizar:

1. Álgebra de Boole: Que es un método 100% algebraico y sirve para minimizar funciones booleanas con cualquier número de variables.
2. Mapas de Karnaugh: Es un método gráfico y sirve para minimizar funciones booleanas con un número pequeño de variables (máximo 5 ó 6).
3. Quine Mcklusky: Es un método tabular y sirve para minimizar funciones booleanas con un gran número de variables; por ser un método interactivo es susceptible de ser computarizado.

2.-Mapas de Karnaugh:

a) **Introducción:** El mapa de Karnaugh es una representación gráfica de un espacio booleano, que contiene una expresión booleana y es equivalente a una tabla de verdad.

b) **Características del mapa de Karnaugh:** Un mapa de Karnaugh de "n" variables debe cumplir con:

i. Tener 2^n "cuadritos".

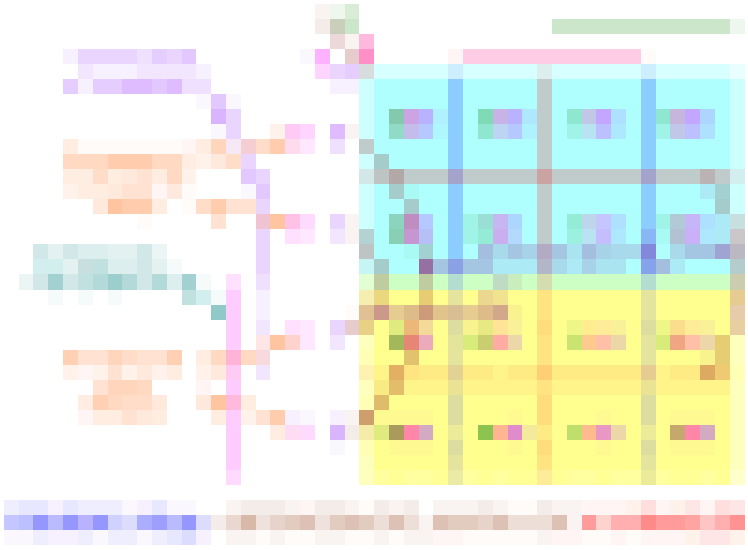
ii. Organizar esos "cuadritos" en renglones y columnas tratando de que sea lo mas cuadrado posible, puesto que el número de renglones y columnas serán potencia de 2.

iii. Puesto que el mapa de Karnaugh la representa un espacio booleano al definir la región de cada variable de generación los cuadritos mencionados de tal forma que se debe cumplir que cualquier par de cuadritos inmediatos deben corresponder a condiciones de combinaciones de variables

lógicamente adyacentes, es decir, que difieran en un solo “bit”.

*Cada cuadrito representa un Míntérmino.

Nota: Podemos observar que al definir los renglones de cada variable se generan los cuadritos, las cuales representan cada uno, una representación de las variables booleanas; pero como deben cumplir con adyacencia lógica (que varían en un bit) y se nos puede hacer la vida de cuadritos, el definir el dominio de cada variable basta en combinar renglones y columnas utilizando el código Grey reflejado.
Ejemplo: Construir un mapa de Karnaugh de cuatro variables:



c) Minimización de funciones booleanas usando mapas de Karnaugh (considerando minterminos).

Los pasos a seguir son:

i. Llenar el mapa de Karnaugh con la correspondiente función booleana.
ii. Considerar las condiciones “DON’T CARE” como (*).
iii. Agrupar los 1’s adyacentes en grupos de 1’s potencia de 2 (1, 2, 4, 8, 16, etc. 1’s).
iv. Si ayuda tomar (*) como 1’s con el fin de hacer grupos más de una vez los 1’s, siempre y cuando no existan conjuntos redundantes.

- Identificar la mínima cantidad de grupos que agrupen a todos los 1’s.
- La función minimizada será igual a la suma de los dominios de esos grupos.
- La minimización no es única.

Truco 1. Dividir el mapa, en mapas mas manejables (4x4) y obtener para cada mapa su superdominio; trabajar cada mapa independiente tratando de ser congruente con los grupos de los mapas, obtener la minimización de cada mapa premultiplicado por su súper dominio sumar las minimizaciones y aplicar algunos teoremas del álgebra de Boole.

Truco 2. Es similar al truco anterior pero consiste en doblar los mapas hasta tener un solo mapa de (4x4), todos los 1's o conjuntos queden traslapados en su totalidad serán adyacentes.

Truco 3. Consiste en tener mapas de Karnaugh que sean manejables independientemente del valor de la función, esto es, 0, 1 o una función o variable como valor de la función.

a) Reducción de Tablas de Verdad de Múltiples Variables a Tablas de Verdad con Variables o funciones de entrada.

Paso 1. Agruparlos expresiones de residuos U's que aparezcan en celdas adyacentes de la misma forma que se agrupan los 1's multiplicando por su prefijo a dominio correspondiente; de la misma forma se agrupan los residuos u's, y para facilitar el agrupamiento podemos considerar que:

$$1 = u + u'$$
$$* = u* + u'*$$

Paso 2. Una vez que hayamos los U's y u's necesitamos hacer una transformación del mapa.

Regla:

1. Transformar las u, u´, u* ,u´ * a cero.
2. Transformar las u´*+u, ,u*+u´ a * siempre y cuando al menos en términos sin * haya sido agrupado.
3. Transformar u´*+u, u*+u´a 1, cuando ninguno de los términos haya sido agrupado o sólo el término con *.
4. Transformar los 1 en * sí y sólo sí ambos términos de 1 hayan sido agrupados.

3.-Con las transformaciones tenemos un nuevo mapa con sólo 1, 0 y * y hay que minimizar.

4.-La minimización final será igual a lo obtenido en el paso 1 y en el 3.

d) **Minimización con mapas de Karnaugh con sistemas de salida múltiple.**

Regla. Se hacen tantos mapas como funciones de salida.

En un espacio booleano de n variable podemos representar 2_n funciones diferentes.

e) **Minimización de mapas de Karnaugh usando máxterminos.** Con un mapa de Karnaugh podemos minimizar una función booleana y expresarla como producto de sumas sí consideramos que el mapa del Karnaugh representa máxterminos podemos trabajar con el mapa como lo conocemos.

Nota: El dominio es la suma de las negaciones.

3.-Minimización por Quine Mcklusky.

a) Introducción: Como mencionamos este método es un método tabular que sirve para minimizar funciones booleanas con un gran número de variables y que por ser iterativo puede ser programado por computadora.

b) Metodología: Los pasos a seguir son:

1. Representar la función en minterminos.
2. Identificar el número de 1's de cada Míntermino y agruparlos por No. de 1's.
3. Obtener los implicantes primos, para ello.

*) Generar la tabla siguiente:

Índice	Función para las cuales f=1 ó *	Factor de peso

**) Agrupar mínterminos de grupos adyacentes de la siguiente forma:

i. La diferencia entre términos de grupos adyacentes debe de ser potencia de 2, es decir variar en un solo bit, el factor de peso debe ser igual.
ii. El Míntermino más pequeño debe de pertenecer al grupo con índice menor.
iii. El índice de la combinación debe de ser igual al índice menor.
iv. Indicar con un (paloma) los términos que sean padres de una combinación.
v. El factor de peso será igual a la diferencia decimal entre los dos términos combinados.

***) Los implicantes primos serán los términos sin (paloma) es decir que no hayan sido agrupados y se organizarán en una tabla de implicantes primos.

Implicantes primos	Mínterminos para los cuales F =1

4. Obtener los implicantes primos esenciales.

Para obtener los implicantes primos tenemos.

a) Con la tabla de implicantes primos, reducir de la siguiente forma:

- Buscar columnas que sólo contengan un solo Míntérmino y obtener así los implicantes primos esenciales.
- Absorber renglones y columnas dominados.
- Obtener los implicantes primos secundarios con los que se absorben todos los renglones y columnas.

5. La función minimizando será igual a la suma de los dominios de los implicantes primos esenciales, mas los secundarios.

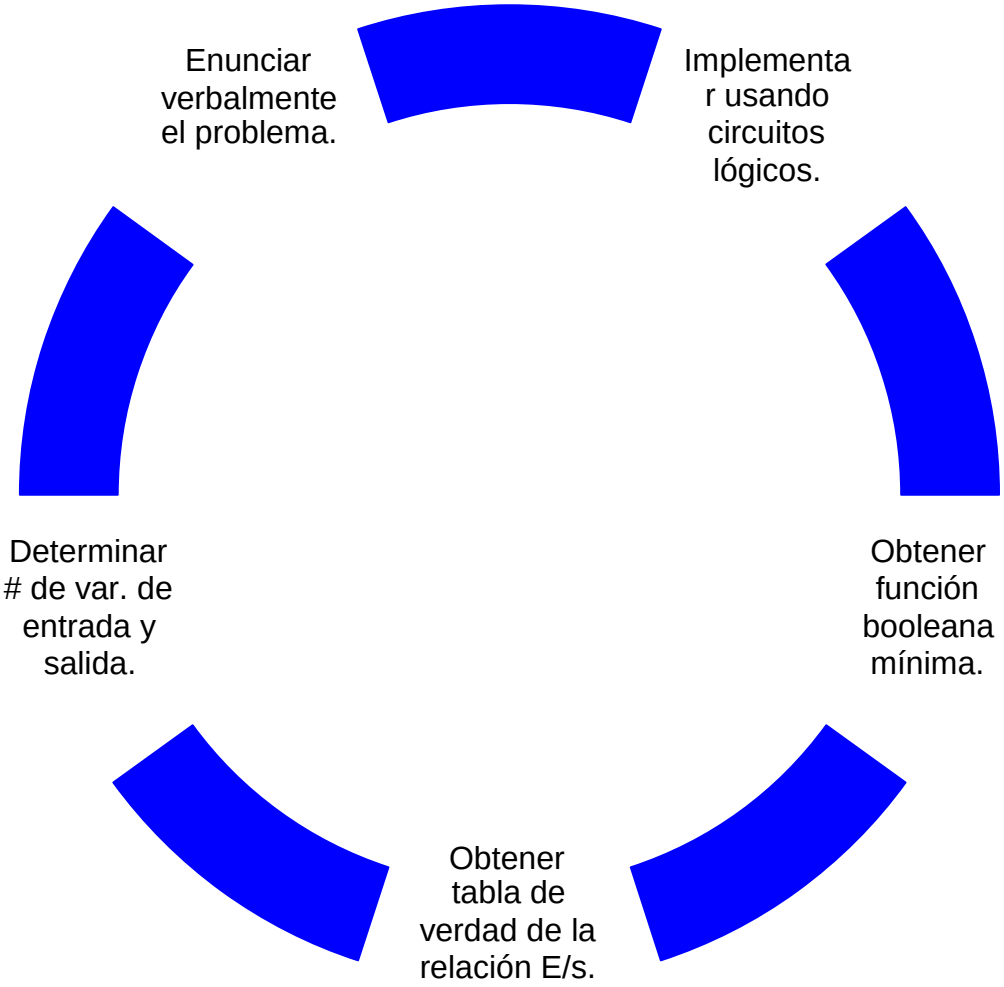
Tema V

"Circuitos Lógicos Combinacionales"

1.-Introducción: Un circuito combinacional es aquel que utiliza compuertas lógicas para su implementación, en donde la salida esta en función de las entradas y del arreglo de las compuertas.

"Todo circuito combinacional puede ser representado por una tabla de verdad".

a) Pasos de Diseño.

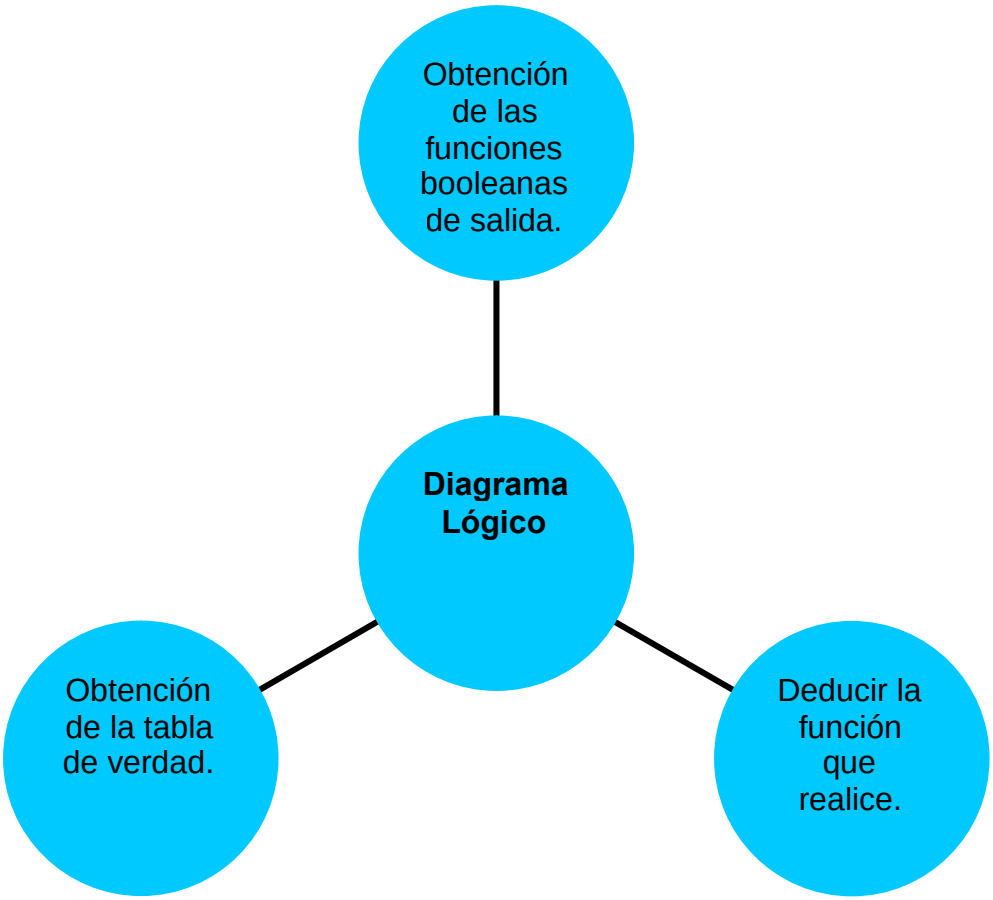


Nota: Cualquier circuito combinacional, y en general cualquier circuito lógico presenta retrasos en el tiempo desde el momento en que se presentan las entradas y el momento que aparece la salida; estos retrasos son debidos a:

1. Niveles de Implementación.
2. Tipos de compuertas lógicas.
3. Tipo de familia lógica con la cual se esta trabajando (TTL, CMOS, DTL, RTL, etc.).

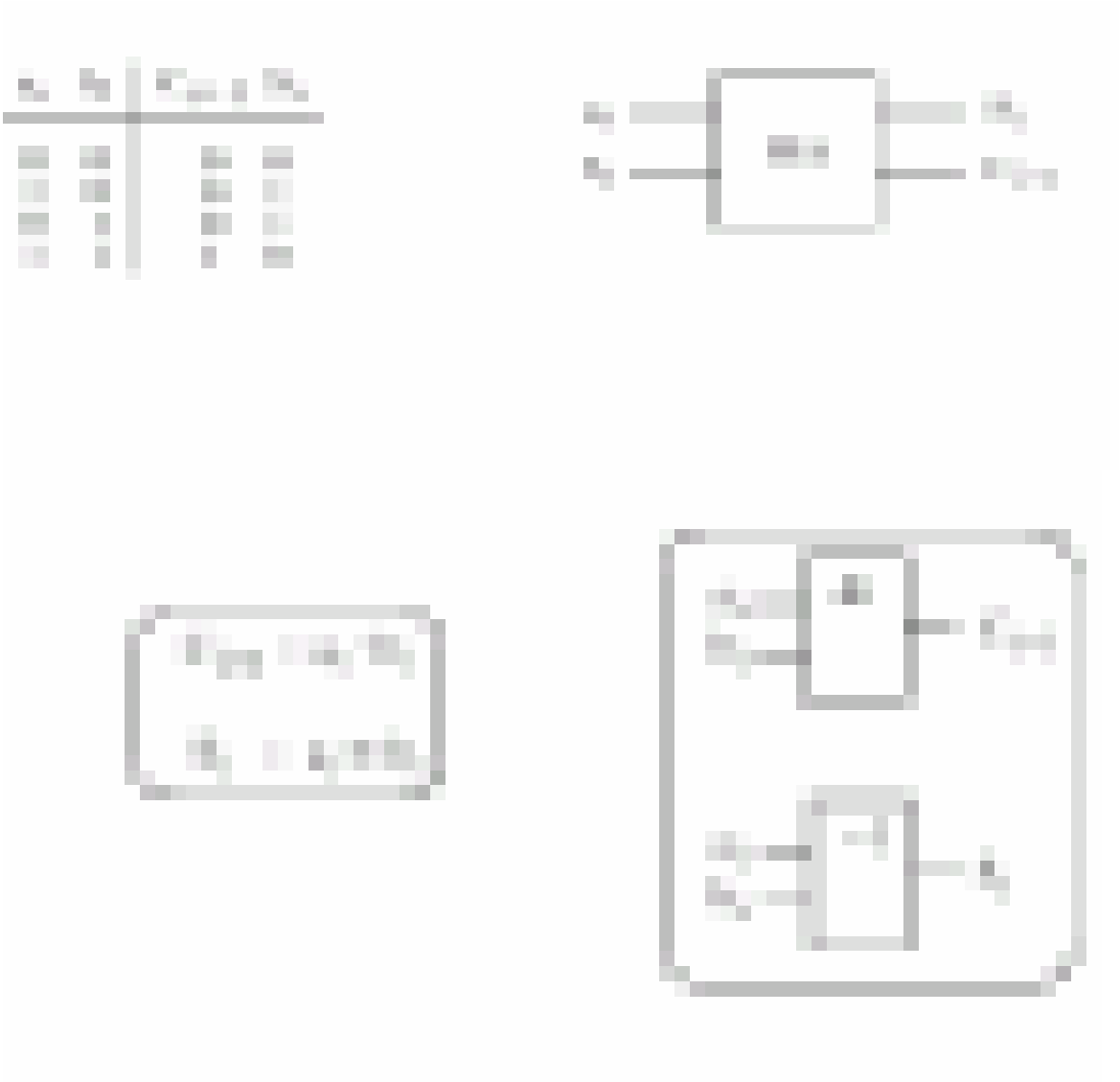
2.-Análisis de circuitos combinacionales.

Analizar un circuito combinacional consiste en dado un diagrama lógico. Obtener la función que realiza (No booleona).
Los pasos a seguir son:



3.- Diseño de Sumadores.

a) **El medio sumador (Half Hadder):** Es aquel sumador de 2 bits que no considera el CARRY anterior para efectuar la suma.



b) Sumador Completo (Foll-adder): Se toma el carry anterior. En forma general para sumar palabras de “n” bits necesitaré 2n-1 tiempos. Para esta implementación tengo que esperara el carry anterior para efectuar la suma en cada tapa, si nosotros conociéramos con anterioridad los “carry’s” podríamos efectuar la suma en cada etapa al mismo tiempo; a este método le llamamos “Carry loock-ahead”.