Organización de Computadoras
2009

Bibliografía

- ***Organización y Arquitectura de Computadoras – Diseño para optimizar prestaciones***, Stallings W., Editorial Prentice Hall (5ta edición).
- ***Organización de Computadoras***, Tanenbaum A., Editorial Prentice Hall (4ta edición).
- ***Estructura de Computadores y Periféricos***, Martinez Durá R. et al., Editorial Alfaomega, 2001.
- ***Arquitectura de Computadores - Un enfoque cuantitativo***, Hennessy & Patterson., Editorial Mc Graw Hill (1ra edición).

Repaso Curso de Ingreso

- Representación de Datos.
- Números sin signo. BCD.
- Lógica digital. Álgebra de Boole.

Representación de datos

- Las computadoras almacenan datos e instrucciones en memoria
- Para ello utilizan el sistema binario
- Razones :
  - el dispositivo se encuentra en uno de dos estados posibles (0 ó 1)
  - identificar el estado es más fácil si sólo hay dos

Representación de datos

- Ejemplo :
  - lámpara encendida ó apagada
  - lámpara encendida con 10 intensidades distintas
  - Es más fácil conocer el “estado” de la lámpara en el primer caso (encendida ó apagada), que determinar alguna de las 10 intensidades distintas

Tipos de datos

Las computadoras manejan 4 tipos básicos de datos binarios

- Números enteros sin/con signo
- Números reales con signo
- Números decimales codificados en binario (BCD)
- Caracteres

Representación de números enteros

- Sin signo
- Módulo y signo
- Complemento a uno (Ca1)
  Complemento a la base reducida
- Complemento a dos (Ca2)
  Complemento a la base
- Exceso

Números enteros sin signo

Si el número tiene n bits, puedo representar

- 2^n = números distintos

El rango va desde

- 0 a $(2^n – 1)$

Números enteros sin signo

Ejemplo: n = 3 bits

Decimal	Representación sin signo
0	000
1	001
2	010
..	…..
7	111

Números enteros sin signo

Ejemplo: n = 8 bits

0	00000000
..	....
128	10000000
..	....
254	11111110
255	11111111

Números enteros sin signo

- RECORDAR: la cantidad de representaciones distintas depende del número de bits

  Nºs distintos = 2^n

Sistemas Posicionales

Teorema Fundamental de la Numeración

$$N^{\circ} = \sum_{i=-m}^{n} (dígito)_i \times (base)^i$$

$$\ldots + x_4 \times B^4 + x_3 \times B^3 + x_2 \times B^2 + x_1 \times B^1 + x_0 \times B^0 + x_{-1} \times B^{-1} + x_{-2} \times B^{-2} + \ldots$$

N^0 es el valor decimal de una cantidad expresada en base B y con (n+1+m) dígitos en posiciones i.

Sistema Decimal

- Base 10.
- Dígitos {0,1,2,3,4,5,6,7,8,9}

  3574 = 3000 + 500 + 70 + 4

  $= 3 \times 10^3 + 5 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$

  - 3 unidades de mil + 5 centenas + 7 decenas + 4 unidades

$$3.1416_{(10} = 3\times10^{0} + 1\times10^{-1} + 4\times10^{-2} + 1\times10^{-3} + 6\times10^{-4}$$

  - 3 unidades + 1 décima + 4 centésimas + 1 milésima + 4 diezmilésimas

Sistema Binario

- Base 2.
- Dígitos {0,1}

$$1001{,}1_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$$

$$= 8 + 0 + 0 + 1 + 0{,}5$$

$$= 9{,}5_{10}$$

Sistema Hexadecimal

- Base 16.
- Dígitos {0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F}

  10,11,12,13,14,15

$$2CA,8_{16} = 2 \times 16^{2} + C \times 16^{1} + A \times 16^{0} + 8 \times 16^{-1}$$

$$= 512 + 192 + 10 + 0,5$$

$$= 714,5_{10}$$

Sistema hexadecimal codificado en binario (BCH)

- Los dígitos hexadecimales se convierten uno a uno en binario
- Para representar un dígito hexadecimal se utilizará siempre 4 bits
- Se asocia cada dígito con su valor en binario puro

BCH

Dígito hexadecimal	Código BCH
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Sistema decimal codificado en binario (BCD)

- Los dígitos decimales se convierten uno a uno en binario
- Para representar un dígito decimal se requerirán 4 bits
- Se asocia cada dígito con su valor en binario puro

BCD

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

BCD

- BCD tiene dos ámbitos de aplicación:

➢ E/S y periféricos, los números se codifican usando un byte por dígito. Se dice que el número está *desempaquetado.*

➢ En cálculo, se reservan 4 bits por dígito. Se dice que el número está *empaquetado.*

BCD

- Ejemplo: desempaquetado sin signo

  834 = 11111000 11110011 11110100

  = F8 F3 F4

- Por cada dígito se usan 8 bits, 4 para el binario puro y 4 se completan con “1”

BCD

- Desempaquetado con signo
- Con 4 bits hay 2^4=16 combinaciones posibles de unos y ceros :

➢ Diez usamos para los dígitos 0 al 9

➢ Nos quedan seis sin usar

➢ C_{16}= 1100 representa al signo +

➢ D_{16} = 1101 representa al signo -

BCD

- Ejemplo: desempaquetado con signo

  $+\ 834 = 11111000\ 11110011\ 11000100$

  $= \text{F8 F3 C4}$

- Los 4 bits que acompañan al último dígito son reemplazados por el signo.

BCD

Ejemplo:

- \- 834 = 11111000 11110011 11010100

  = F8 F3 D4

BCD

Ejemplo: empaquetado con signo

- + 834 = 10000011 01001100

  = 83 4C
- \- 34 = 00000011 01001101

  = 03 4D

Suma en BCD

- De las 16 representaciones posibles con 4 bits, usamos 10 para los dígitos 0 al 9
- Nos sobran 6 combinaciones de 4 bits
- Al sumar dos dígitos BCD, se nos presentan dos casos :
  - la suma es ≤ 9
  - la suma es > 9

Suma en BCD

- En el primer caso no hay problema

	41		0100 0001
+	22	+	0010 0010
	63		0110 0011

Suma en BCD

- En el segundo caso ¿Qué sucede ?

$$\begin{array}{r} 1 \\ 15 \\ +\ 27 \\ \hline 42 \end{array}$$

$$\begin{array}{rr} & 111 \\ 0001 & 0101 \\ +\ 0010 & 0111 \\ \hline \underbrace{0011}_{3} & \underbrace{1100}_{\text{no válido}} \end{array}$$

Suma en BCD

- Cuando la suma de los dos dígitos da >9 hay que generar el “acarreo” porque hay seis combinaciones no usadas

- Entonces: cuando la suma de los dígitos es > 9 hay que sumar 6 en ese dígito

Suma en BCD

```
   1
   15
+  27
 ------
   42
```

```
             111
   0001  0101
+  0010  0111
 ---------------
   111   1
   0011  1100
+        0110  ←  6
 ---------------
```

Resultado correcto   0100 0010

Suma en BCD

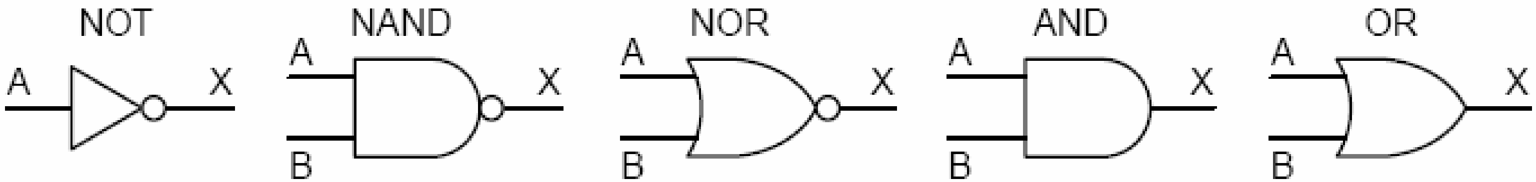
- Ejemplo

```
    1                111   1
  476          0100  0111  0110
+  55                0101  0101
-----          -----------------
  531          0100  1100  1011
+                    0110  0110
               -----------------
               0101  0011  0001
```

El nivel de lógica digital

- Un circuito digital es en el que están presentes dos valores lógicos
- Compuertas son dispositivos electrónicos que pueden realizar distintas funciones con estos dos valores lógicos
- Como vimos en el Ingreso las compuertas básicas son: AND, OR, NOT, NAND, NOR y XOR

Compuertas: símbolo y descripción funcional



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Algebra Booleana

- Para describir los circuitos que pueden construirse combinando compuertas, se requiere un nuevo tipo de álgebra, donde las variables y funciones sólo puedan adoptar valores 0 ó 1: álgebra booleana.

Algebra Booleana

- Puesto que una función booleana de n variables tiene 2^n combinaciones de los valores de entrada, la función puede describirse totalmente con una tabla de 2^n renglones, donde c/u indica un valor de la función (0 ó 1) para cada combinación distinta de las entradas:

=> ***tabla de verdad***

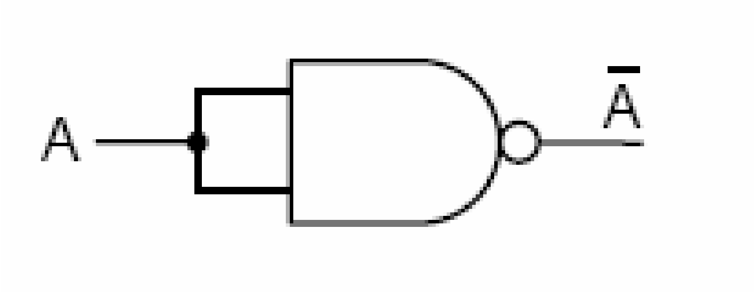
Recordemos algunas identidades del álgebra booleana

Identidad	$1.A=A$	$0+A=A$
Nula	$0.A=0$	$1+A=1$
Idempotencia	$A.A=A$	$A+A=A$
Inversa	$A.\overline{A}=0$	$A+\overline{A}=1$
Conmutativa	$A.B=B.A$	$A+B=B+A$
Asociativa	$(AB).C=A(BC)$	$(A+B)+C=A+(B+C)$
Distributiva	$A+B.C=(A+B).(A+C)$	$A.(B+C)=AB+AC$
Absorción	$A.(A+B)=A$	$A+A.B=A$
De Morgan	$\overline{A.B}=\overline{A}+\overline{B}$	$\overline{A+B}=\overline{A}.\overline{B}$

Leyes de De Morgan

- Ejemplo: construir un NOT con NAND

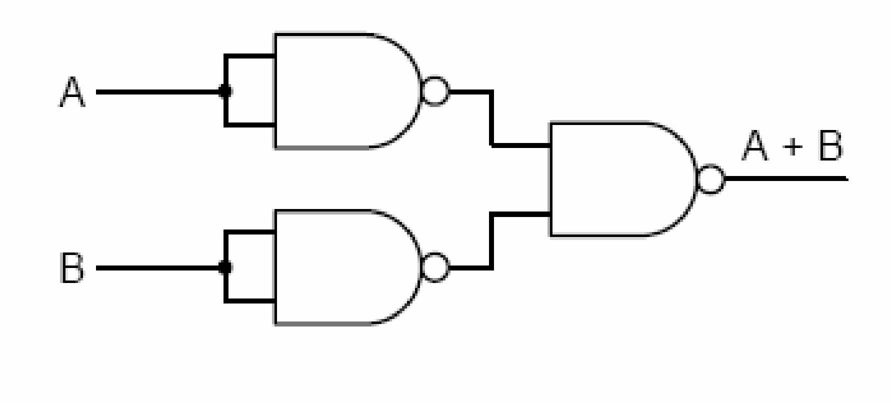
$$F=\overline{A.B}=\overline{A.A}=\overline{A}$$



Leyes de De Morgan

- Ejemplo: construir un OR con NAND

$$F = A + B = \overline{\overline{A + B}} = \overline{\overline{A}\ .\ \overline{B}}$$



Implementación de funciones booleanas

- Escribir la tabla de verdad para la función
- Dibujar una AND para cada término que tiene un 1 en la columna de resultado (con sus entradas apropiadas)
- Invertir las entradas necesarias
- Unir todas las AND a una OR

Implementación

Ejemplo: construir la tabla de verdad e implementar el circuito de una función booleana M, de tres entradas A, B y C, tal que M=1 cuando la cantidad de '1' en A, B y C es ≥ 2 y M=0 en otro caso.

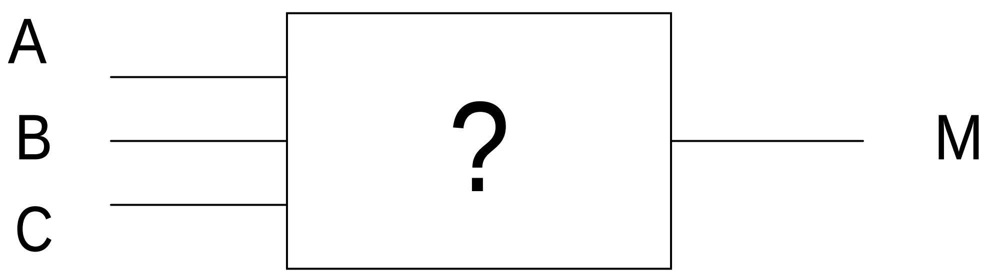


Tabla de verdad

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

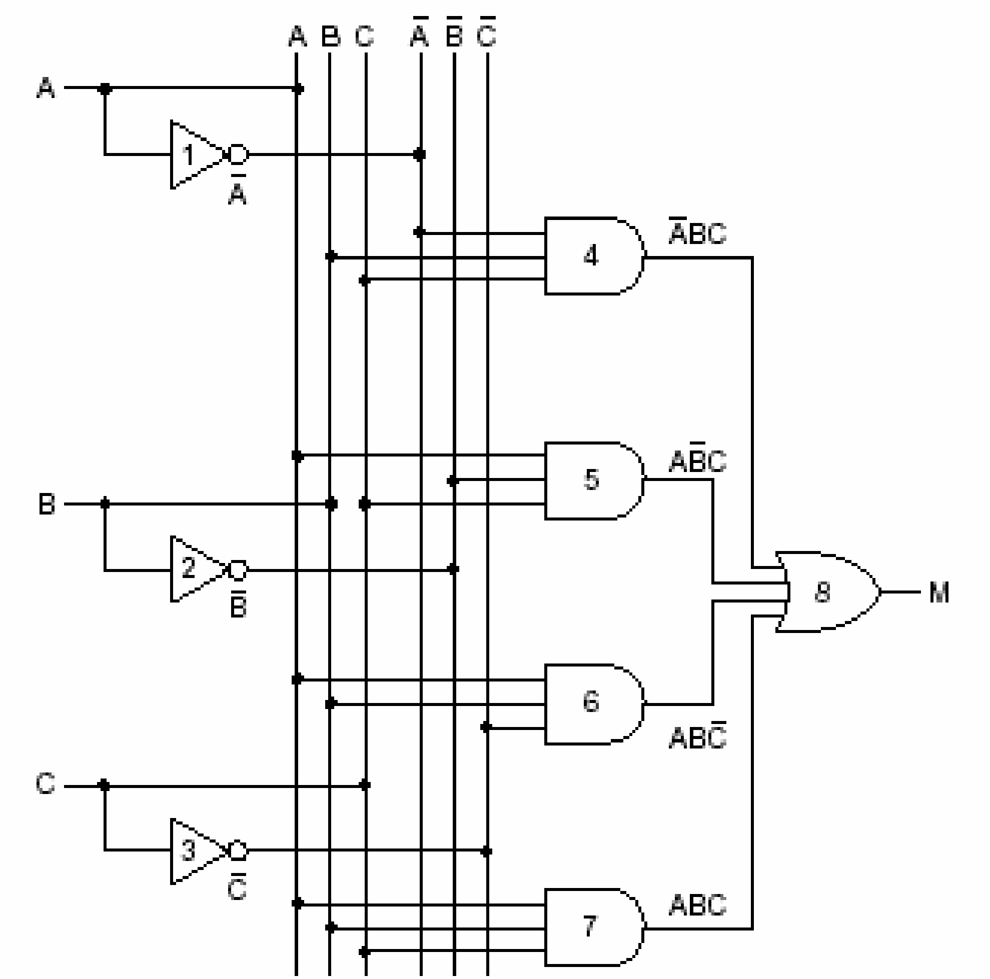
Función M

$$M=\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

- Hay tantos términos como 1s en la tabla
- Cada término vale 1 para una única combinación de A, B y C
- Las variables que valen 0 en la tabla aparecen aquí negadas

Función M (2)

$$M=\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$



Otro ejemplo

Supongamos la siguiente Tabla de Verdad

A	B	M
0	0	0
0	1	1
1	0	1
1	1	0

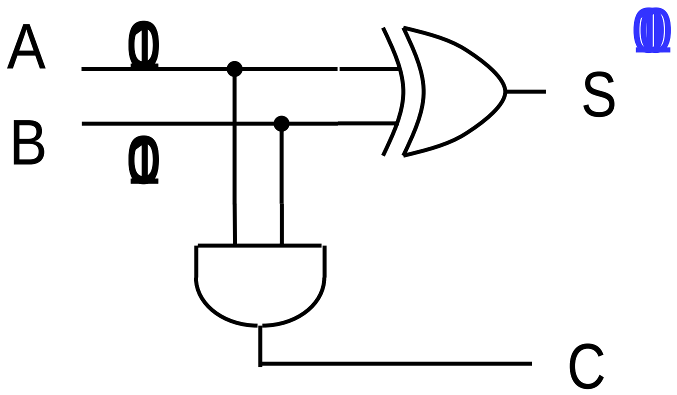
Función $\mathbf{M = \overline{A}B + A\overline{B}} \Rightarrow \mathbf{M = XOR}$

Recordemos

- ✓En un AND, basta que una de sus entradas sea 0 para que la función valga 0.
- ✓En un OR, basta que una de sus entradas sea 1 para que la función valga 1.
- ✓Hacer el XOR con 1 invierte el valor de la variable.
- ✓Hacer el XOR con 0 deja el valor de la variable como estaba.

Circuitos combinatorios

- Ejemplo



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

S representa la suma aritmética de 2 bits y **C** es el acarreo

Semi-sumador ó Half adder

mayor información ...

- Sistemas enteros y Punto fijo
  - Apunte 1 de Cátedra
- Operaciones lógicas
  - Apunte 3 de Cátedra
- Apéndice 8A: Sistemas de Numeración
  - Stallings, 5ta Ed.
- Apéndice A: Lógica digital (A.1., A.2.)
  - Stallings, 5ta Ed.
- Capítulo 3: Lógica digital y representación numérica
  - Apuntes COC - Ingreso 2009