Conceptos de Organización de Computadoras

Curso de Ingreso 2014

Clase 2

Prof. Jorge M. Runco

1

- Debido a la construcción del sistema de cómputo, basada fundamentalmente en circuitos electrónicos digitales, utiliza un sistema *binario*, es decir la información está representada por unos y ceros .
- Esto obliga a transformar la representación de nuestra información, tanto numérica como alfanumérica (letras,+,/,*,etc) a una representación binaria para que la máquina sea capaz de procesarlos.

2

- Los dispositivos que procesan unos y ceros se denominan circuitos lógicos.
- Vamos a ver dispositivos (compuertas) que realizan funciones básicas.
- Dentro de un sistema de cómputo estas compuertas se conectan para realizar funciones más complejas, como por ejemplo un sumador.

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.
- 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:

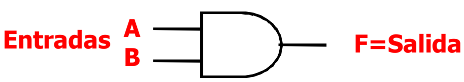
  => ***tabla de verdad***

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
- Estas funciones básicas son: AND, OR, NOT, NAND, NOR y XOR

5

AND – Y – Conjunción



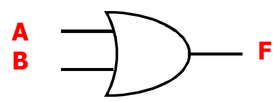
F = A . B = A ^ B

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

La salida F es 1 sólo cuando las dos entradas son 1 ó basta que una entrada valga 0 para que la salida sea 0.

6

OR – O – Disyunción



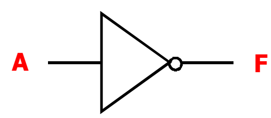
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

F = A + B = A v B

La salida F es 0 sólo cuando las dos entradas son 0 ó basta que una entrada valga 1 para que la salida sea 1.

7

NOT – Negación

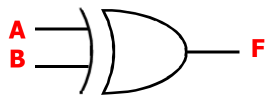


A	F
0	1
1	0

$F = \overline{A} = \sim A$

8

XOR (OR-Exclusive) (O-Exclusiva)



F = A ⊕ B

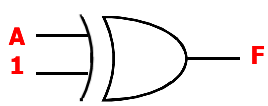
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

La salida F es 1 sólo cuando las dos entradas son distintas ó cuando una única entrada es uno. Es 0 cuando las dos entradas son iguales.

9

XOR

- Supongamos que una de las entradas (A ó B) permanece fija en el valor 1. ¿Cuánto valdrá la salida?



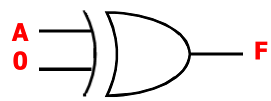
A	B	F
0	1	1
1	1	0

- Como B está fija en 1, vemos la tabla de verdad “reducida”. Si A=0, F=1. Si A=1, F=0. Hacer el XOR con 1 invierte la otra variable.

10

XOR

- Supongamos que una de las entradas (A ó B) permanece fija en el valor 0. ¿Cuánto valdrá la salida?

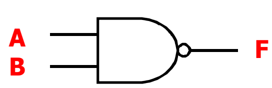


A	B	F
0	0	0
1	0	1

- Como B está fija en 0, vemos la tabla de verdad "reducida". Si A=0, F=0. Si A=1, F=1. Hacer el XOR con 0 "deja" a la otra variable sin cambios.

11

NAND – (NOT-AND)

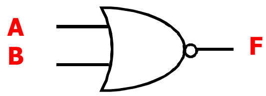


$F = \overline{A \cdot B} = \sim(A \cdot B)$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

12

NOR – (NOT-OR)



$F = \overline{A + B} = \sim(A + B)$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

13

Resumiendo

- Para que una variable lógica tome el valor 0, debo hacer un AND con 0.
- Para que una variable lógica tome el valor 1, debo hacer un OR con 1.
- Para invertir una variable lógica, debo hacer un XOR con 1.

14

Identidad	1.A=A	0+A=A
Doble negación	$\overline{\overline{A}} = A$	
Nula	0.A=0	1+A=1
Idempotencia	A.A=A	A+A=A
Inversa	A.A=0	A+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	$\overline{A.B}=\overline{A}+\overline{B}$	$\overline{A+B}=\overline{A}.\overline{B}$

Práctica 3: Ej.1

- 00010001 AND 01011100 =

```
        00010001
AND     01011100
       ----------
        00010000
```

- 01010101 AND 01010101=

```
        01010101
AND     01010101
       ----------
        01010101
```

Práctica 3: Ej.1

- 01010101 AND 10101010 =

```
      01010101
AND   10101010
      --------
      00000000
```

- 11110000 AND 11111111=

```
      11110000
AND   11111111
      --------
      11110000
```

17

Práctica 3: Ej.1

- 01010101 OR 01010101 =

```
      01010101
OR    01010101
      --------
      01010101
```

- 01010101 OR 10101010=

```
      01010101
OR    10101010
      --------
      11111111
```

18

Práctica 3: Ej.1

- 11110001 OR 11110010 =

```
       11110001
OR     11110010
       --------
       11110011
```

- 01010101 XOR 01010101 =

```
       01010101
XOR    01010101
       --------
       00000000
```


Práctica 3: Ej.1

- 01010101 XOR 10101010 =

```
       01010101
XOR    10101010
       --------
       11111111
```

- 00001111 XOR 00000000=

```
       00001111
XOR    00000000
       --------
       00001111
```

Práctica 3: Ej.1

- NOT 11111111 = 00000000
- NOT 01000000 = 10111111
- NOT 00001110 = 11110001

Práctica 3: Ej. 2)

- DATO Op. Lógica MASK = RESULTADO
- $D_7D_6D_5D_4D_3D_2D_1D_0$? ? = $D_7D_6D_5D_4 1\ D_2D_1D_0$

$$\begin{array}{rl} \text{OR} & D_7D_6D_5D_4D_3D_2D_1D_0 \\ & 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \\ \hline & D_7D_6D_5D_4 1\ D_2D_1D_0 \end{array}$$

- $D_7D_6D_5D_4D_3D_2D_1D_0$? ? = $0\ D_6D_5D_4D_3\ D_2D_1D_0$

$$\begin{array}{rl} \text{AND} & D_7D_6D_5D_4D_3D_2D_1D_0 \\ & 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ \hline & 0\ D_6D_5D_4D_3D_2D_1D_0 \end{array}$$

Práctica 3: Ej. 2)

- DATO Op. Lógica MASK = RESULTADO
- $D_7D_6D_5D_4D_3D_2D_1D_0$? ? = $D_7\overline{D_6}D_5\overline{D_4}D_3\ D_2D_1D_0$

$$
\begin{array}{ll}
 & D_7D_6D_5D_4D_3D_2D_1D_0 \\
\text{XOR} & 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0 \\
\hline
 & D_7\overline{D_6}D_5\overline{D_4}D_3\ D_2D_1D_0
\end{array}
$$

23

Práctica 3: Ej. 3)

- Sólo con NAND ó sólo con NOR, se pueden implementar el resto de las funciones lógicas básicas.
- Por esa razón a estas funciones se las llama "completas".
- Las leyes de De Morgan nos relacionan NAND con NOR. Observar que de un lado de la igualdad hay un signo +(.) y del otro lado .(+)

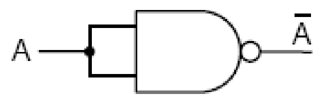
24

Práctica 3: Ej. 3)

- Ejemplo: construir un NOT con NAND

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

Idempotencia



25

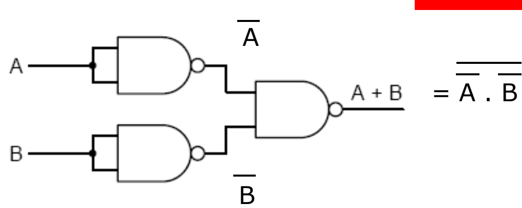
Práctica 3: Ej. 3)

- Ejemplo: construir un OR con NAND

Doble Negación

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

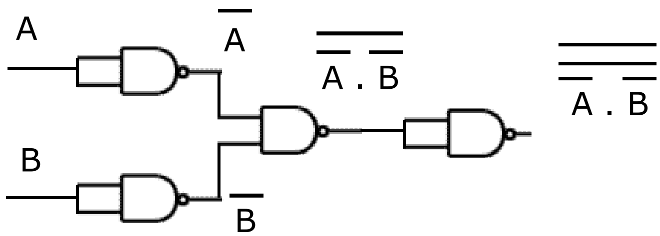
De Morgan



26

Práctica 3: Ej. 3)

- Ejemplo: construir un NOR con NAND



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

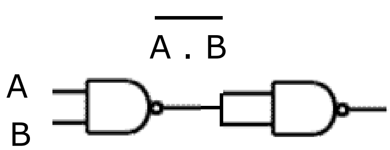
De Morgan

Doble Negación

27

Práctica 3: Ej. 3)

- Ejemplo: construir una AND con NAND

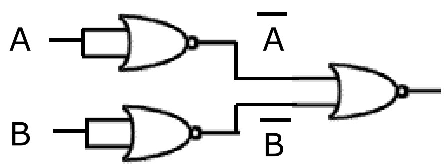


$$F = A . B = \overline{\overline{A . B}}$$

28

Práctica 3: Ej. 3)

- Ejemplo: construir una AND con NOR



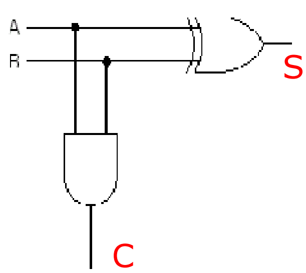
$$F = \overline{\overline{A} + \overline{B}}$$

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

De Morgan

Doble Negación

Práctica 3: Ej. 4)



$$\begin{array}{r} 0 \\ +\;0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\;1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\;0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\;1 \\ \hline 10 \end{array}$$

$$\begin{array}{r} A \\ +\;B \\ \hline C\;S \end{array}$$

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

- El circuito lógico anterior es un sumador de dos números de 1 bit c/u (A y B) y el resultado de la suma es (S) y en algunos casos se puede generar acarreo (C), que en la suma de todos los días llamamos "me llevo 1".
- Se llama semi-sumador ó sumador incompleto, porque sólo suma dos dígitos (A y B) y no permite sumar el acarreo del dígito anterior.

31

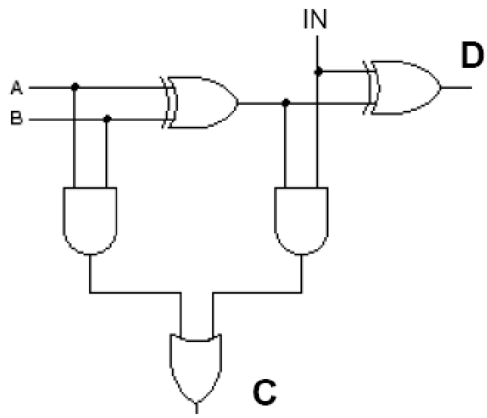
Veamos una suma

$$
\begin{array}{cccc}
C_2 & C_1 & C_0 & \\
 & A_2 & A_1 & A_0 \\
+ & B_2 & B_1 & B_0 \\
\hline
 & S_2 & S_1 & S_0
\end{array}
$$

En general, en una suma de dos dígitos (A y B) hay que sumar otro dígito más: "el que me llevo del dígito anterior". Nuestro sumador no tiene contemplado la posibilidad de sumar este dígito C.

32

Práctica 3: Ej. 4)



Este sumador que puede sumar ese dígito C (acá llamado IN), recibe el nombre de sumador completo. Este circuito lógico suma dos dígitos (bits) más el acarreo del bit anterior (me llevo). El resultado de la suma es D y el acarreo es C.

33