Práctica 2: Lógica Digital - Combinatorios

Gustavo Hurovich

Organización del Computador I
DC - UBA

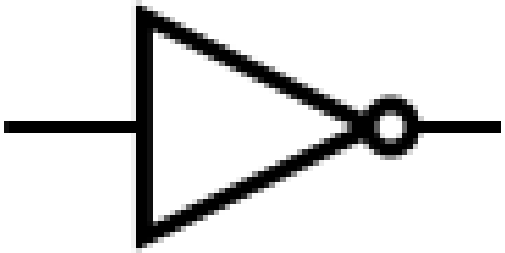
2017 - Segundo Cuatrimestre

Menú de hoy

Para hoy tenemos...

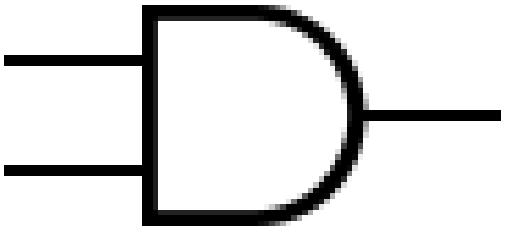
- Compuertas
- Tablas de Verdad
- Álgebra de Boole + Propiedades
- Ejercicios
- ¡Más ejercicios!

Compuertas



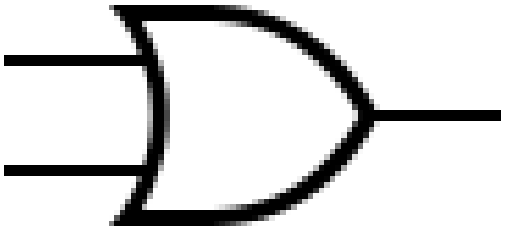
A	NOT A
0	1
1	0

Compuertas



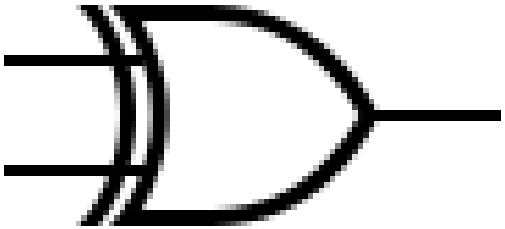
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Compuertas



A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Compuertas



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Notación

$$A + B \equiv A \ OR \ B$$

$$AB \equiv A.B \equiv A \ AND \ B$$

$$\overline{A} \equiv NOT \ A$$

Propiedades

Identidad	$1.A = A$	$0 + A = A$
Nulo	$0.A = 0$	$1 + A = 1$
Idempotencia	$A.A = A$	$A + A = A$
Inverso	$A.\overline{A} = 0$	$A + \overline{A} = 1$
Conmutatividad	$A.B = B.A$	$A + B = B + A$
Asociatividad	$(A.B).C = A.(B.C)$	$(A + B) + C = A + (B + C)$
Distributividad	$A + (B.C) = (A + B).(A + C)$	$A.(B + C) = A.B + A.C$
Absorción	$A.(A + B) = A$	$A + A.B = A$
De Morgan	$\overline{A.B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}.\overline{B}$

Tarea: ¡Demostrarlas!

Ejercicio I

Dada la siguiente tabla de verdad:

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

1. Escribir una función booleana a la que representa.
2. Implementar dicha función usando a lo sumo una compuerta binaria AND, una compuerta binaria OR y una compuerta NOT

Ejercicio I

Solución:

Como suma de productos:

$(\overline{A}.\overline{B}.C) + (\overline{A}.B.C) + (A.B.C)$

Ejercicio I

Solución:

$(\overline{A}.\overline{B}.C) + (\overline{A}.B.C) + (A.B.C) \longrightarrow$ Aplicamos la prop. distributiva

$((\overline{A}.\overline{B}) + (\overline{A}.B) + (A.B)).C \longrightarrow$ Distributiva

$((\overline{A}.\overline{B}) + (\overline{A} + A).B).C \longrightarrow$ Inverso

$((\overline{A}.\overline{B}) + 1.B).C \longrightarrow$ Identidad

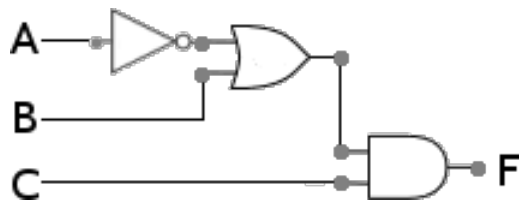
$((\overline{A}.\overline{B}) + B).C \longrightarrow$ Distributiva

$((\overline{A} + B).(\overline{B} + B)).C \longrightarrow$ Inverso

$((\overline{A} + B).1).C \longrightarrow$ Identidad

$(\overline{A} + B).C$

La implementación sería:



Ejercicio II - Shift

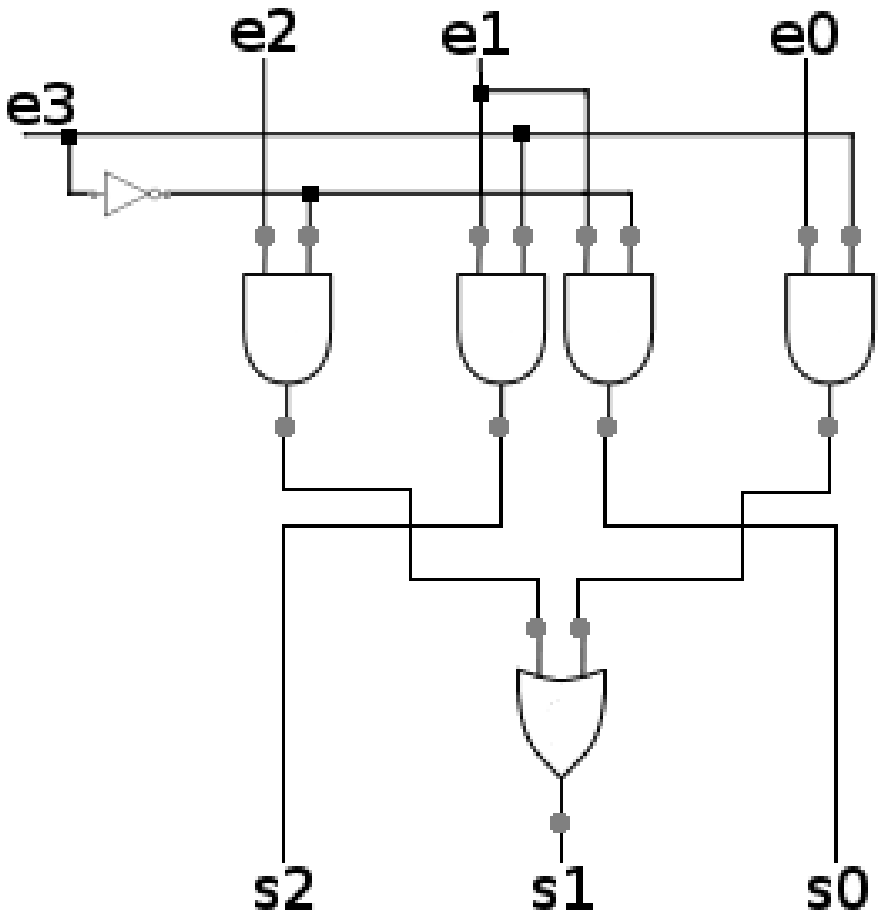
Armar un circuito de 3 bits. Este deberá mover a izquierda o a derecha los bits de entrada de acuerdo al valor de una de ellas que actúa como control. En otras palabras, un shift izq-der de k-bits es un circuito de $k+1$ entradas $(e_k, ..., e_0)$ y k salidas $(s_{k-1}, ..., s_0)$ que funciona del siguiente modo:

- Si $e_k = 1$, entonces $s_i = e_{i-1}$ para todo $0 < i < k$ y $s_0 = 0$
- Si $e_k = 0$, entonces $s_i = e_{i+1}$ para todo $0 \leq i < k-1$ y $s_{k-1} = 0$

Ejemplos:
shift_lr(1,011)= 110
shift_lr(0,011)= 001
shift_lr(1,100)= 000
shift_lr(1,101)= 010

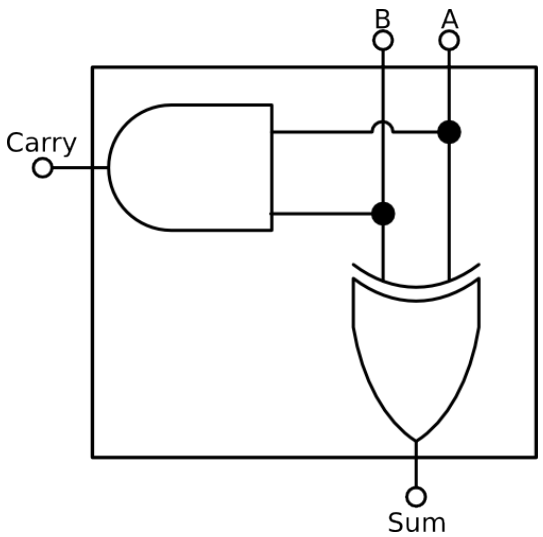
Ejercicio II - Solución

Solución:



Ejercicio III - Sumador Simple

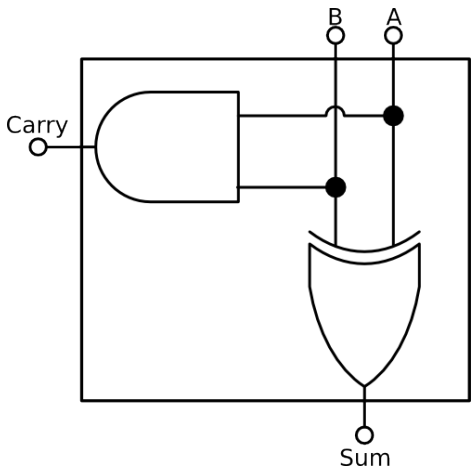
Armar un **sumador de 1 bit**. Tiene que tener dos entradas de un bit y dos salidas, una para el resultado y otra para indicar si hubo o no acarreo.
Solución:

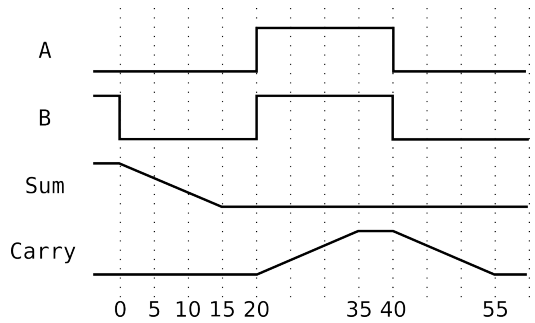


Ejercicio III - Sumador Simple - Segunda parte

En un **sumador de 1 bit** se observa inicialmente que la entrada A vale 0 y la B vale 1. En el tiempo cero, ambas pasan a valer 0; a los 20 ns ambas pasan a valer 1; 20 ns más tarde vuelven ambas a cero y así sucesivamente.

Sabiendo que las compuertas AND y XOR tienen un retardo de 15 ns, realizar un diagrama de tiempos del circuito.



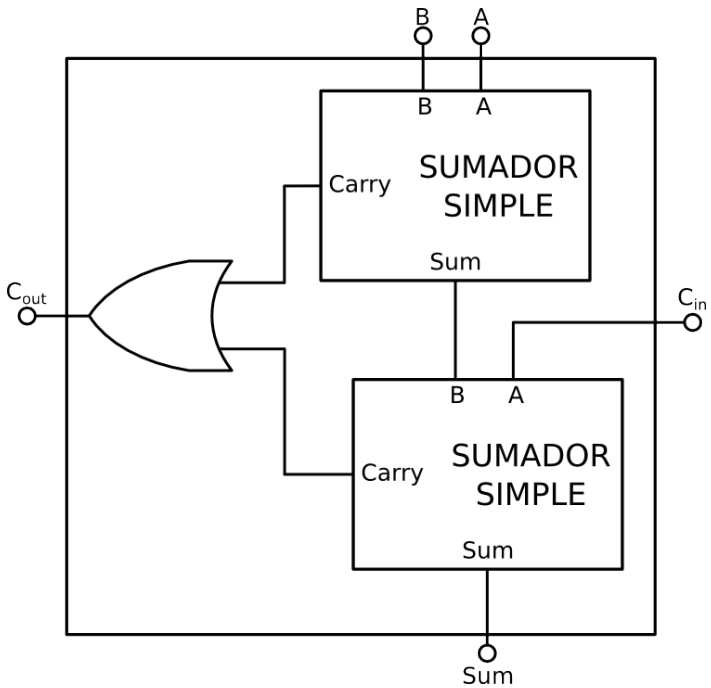


Moraleja: NO podemos considerar válido el output de un componente antes de que se cumpla su tiempo de propagación.

Ejercicio IV - Sumador Completo

Teniendo dos sumadores simples (de 1 bit) y sólo una compuerta a elección, arme un **sumador completo**. Tiene 2 entradas de 1 bit y una tercer entrada interpretada como C_{In}, tiene como salida C_{Out} y S.

Solución:



Ejercicio V - Sumador Completo de 3 bits

Armar un sumador completo de 3 bits.
Solución:

¡Tarea!

Más circuitos combinatorios!

Decodificador de n bits: Tiene n entradas y 2^n salidas. Sea k el número representado en binario en la entrada del decodificador, la salida e_k tendrá un uno lógico, mientras que para todas las demás señales de salida habrá un cero lógico.
Codificador de n bits: Tiene n entradas y $log_2(n)$ salidas. En la salida muestra en binario el número de la entrada que está levantada. De haber más de una o ninguna, el comportamiento del circuito dependerá de la implementación del fabricante.
Multiplexor de n entradas: Tienen n entradas, una salida y $log_2(n)$ señales de control. Mediante las señales de control se indica cuál entrada es requerida en la salida.
Demultiplexor de n salidas: Tienen n salidas, una entrada y $log_2(n)$ señales de control. Igual que el multiplexor, pero elijo mediante las señales de control por cuál señal de salida muestro la entrada.

La práctica...

- Con lo visto hoy pueden realizar toda la parte A de la práctica 2.
- Pueden usar el Logisim para probar sus circuitos.

Bibliografía

- Linda Null & Julia Lobur: The Essentials of Computer Organization and Architecture, Chapter 3
- Andrew S. Tanenbaum & Todd Austin: Structured Computer Organization, Chapter 3
- Curiosidad para chusmear: Charles Petzold: Code: The Hidden Language of Computer Hardware

¿Y ahora?

Lo que viene: **Martes 28 de agosto** a las **17hs** tendremos el primer **taller** de la materia. ¡Es **obligatorio**! (Y re divertido)

¡Eso es todo amigos!

¿Preguntas?