# Práctica 2: Lógica digital - Combinatorios

Gabriel Budiño

Organización del computador I
DC - UBA

1er. cuatrimestre 2018

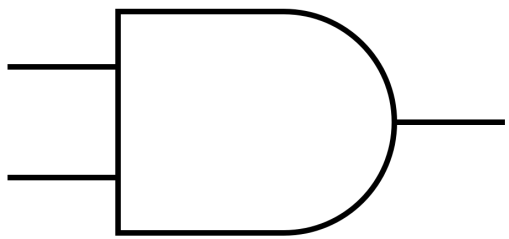
# Repaso

- Operadores lógicos:
  - NOT, OR, AND, XOR, NOR, NAND
  - Son descriptos por su tabla de verdad

- Expresiones booleanas:
  - Combinación de operadores lógicos y variables booleanas. Por ejemplo, $F(X, Y, Z) = X + Y \cdot Z$
  - Una tabla de verdad describe todas las combinaciones de valores de verdad para una función lógica determinada.
  - Dos expresiones son iguales sii tienen la misma tabla de verdad.

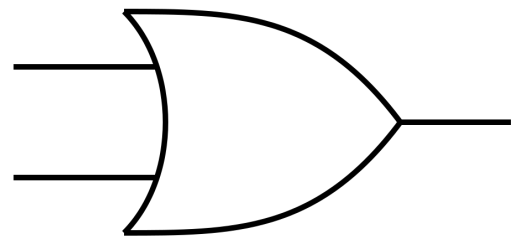
# Compuertas AND y OR

AND $(A \cdot B)$



| A | B | AND |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

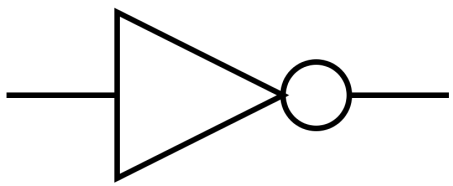
OR $(A + B)$



| A | B | OR |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

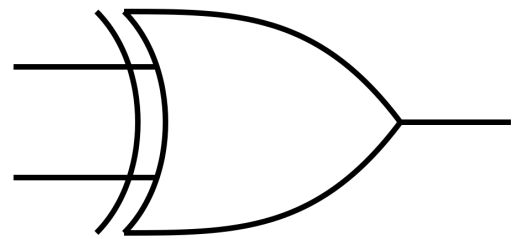
# Compuertas NOT y XOR

NOT ($\overline{A}$)



| A | NOT |
|---|---|
| 0 | 1 |
| 1 | 0 |

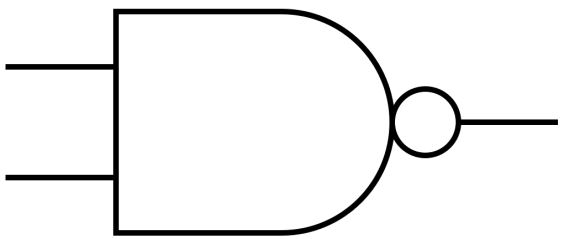
XOR ($A \oplus B$)



| A | B | XOR |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

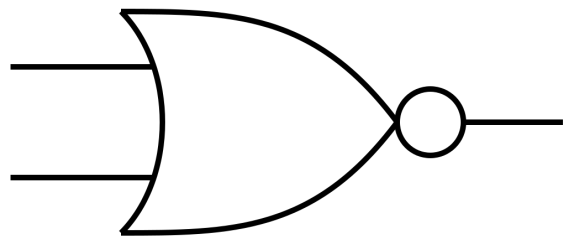
# Compuertas NAND y NOR

NAND $(A \mid B)$



| A | B | NAND |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOR $(A \downarrow B)$



| A | B | NOR |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

# Propiedades

Propiedades para las operaciones $(\cdot)$ y $(+)$

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

# Ejercicio 1

- Demostrar que las siguientes funciones booleanas son equivalentes:
  - $\overline{X + \overline{Y}}$
  - $\overline{X} \cdot Y \cdot Z + X \cdot \overline{Z} + \overline{Y + Z}$

# Ejercicio 1

- Demostrar que las siguientes funciones booleanas son equivalentes:
  - $\overline{X + \overline{Y}}$
  - $\overline{\overline{X} \cdot Y \cdot Z + X \cdot \overline{Z} + \overline{Y + Z}}$

**Solución:**

$$
\begin{array}{ll}
\overline{\overline{X} \cdot Y} \cdot Z + X \cdot \overline{Z} + \overline{Y + Z} & \leftarrow \text{De Morgan} \\
\overline{\overline{X} \cdot Y} \cdot Z + X \cdot \overline{Z} + \overline{Y} \cdot \overline{Z} & \leftarrow \text{Distributiva} \\
\overline{\overline{X} \cdot Y} \cdot Z + (X + \overline{Y}) \cdot \overline{Z} & \leftarrow \text{De Morgan} \\
(X + \overline{Y}) \cdot Z + (X + \overline{Y}) \cdot \overline{Z} & \leftarrow \text{Distributiva} \\
(X + \overline{Y}) \cdot (Z + \overline{Z}) & \leftarrow \text{Inverso} \\
(X + \overline{Y}) \cdot 1 & \leftarrow \text{Identidad} \\
(X + \overline{Y}) &
\end{array}
$$

# Formas canónicas de expresiones booleanas

- Suma de productos:

| A | B | F(A, B) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

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

- Producto de sumas:

| A | B | F(A, B) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

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

# Ejercicio 2

- Dada la siguiente tabla de verdad:
  1. Escribir la función booleana que representa
  2. Implementar la función usando a lo sumo, una compuerta binaria AND, una compuerta binaria OR y una compuerta NOT

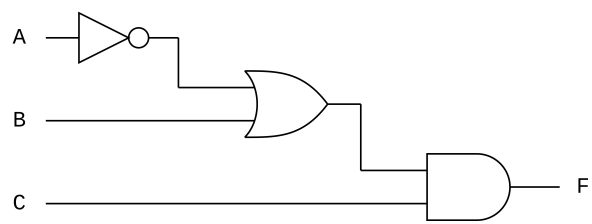
| A | B | C | F(A, B, C) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Solución ejercicio 2

- **Como suma de productos:**
  $(\overline{A} \cdot \overline{B} \cdot C) + (\overline{A} \cdot B \cdot C) + (A \cdot B \cdot C)$
- **Más compacto:**

$$
\begin{aligned}
&(\overline{A} \cdot \overline{B} \cdot C) + (\overline{A} \cdot B \cdot C) + (A \cdot B \cdot C) && \leftarrow \text{ Distributiva} \\
&((\overline{A} \cdot \overline{B}) + (\overline{A} \cdot B) + (A \cdot B)) \cdot C && \leftarrow \text{ Distributiva} \\
&((\overline{A} \cdot \overline{B}) + (\overline{A} + A) \cdot B) \cdot C && \leftarrow \text{ Inverso} \\
&((\overline{A} \cdot \overline{B}) + 1 \cdot B) \cdot C && \leftarrow \text{ Identidad} \\
&((\overline{A} \cdot \overline{B}) + B) \cdot C && \leftarrow \text{ Distributiva} \\
&((\overline{A} + B) \cdot (\overline{B} + B)) \cdot C && \leftarrow \text{ Inverso} \\
&((\overline{A} + B) \cdot 1) \cdot C && \leftarrow \text{ Identidad} \\
&(\overline{A} + B) \cdot C
\end{aligned}
$$

- **Implementación:**



# Ejercicio 3 - Shift

- Armar un circuito de 3 bits. Este deberá mover a izquierda o derecha los bits de entrada de acuerdo al valor de una de ellas que actúa como control. Es decir, un Shift Izq-Der de k bits es un circuito de k+1 entradas $(c, e_{k-1}, ..., e_0)$ y k salidas $(s_{k-1}, ..., s_0)$ que funciona del siguiente modo:
  - Si c = 1, $s_i = e_{i-1}$ para todo $0 < i < k$ y $s_0 = 0$
  - Si c = 0, $s_i = e_{i+1}$ para todo $0 \leq i < k - 1$ y $s_{k-1} = 0$

# Ejercicio 3 - Shift

- Armar un circuito de 3 bits. Este deberá mover a izquierda o derecha los bits de entrada de acuerdo al valor de una de ellas que actúa como control. Es decir, un Shift Izq-Der de k bits es un circuito de k+1 entradas $(c, e_{k-1}, ..., e_0)$ y k salidas $(s_{k-1}, ..., s_0)$ que funciona del siguiente modo:
  - Si c = 1, $s_i = e_{i-1}$ para todo $0 < i < k$ y $s_0 = 0$
  - Si c = 0, $s_i = e_{i+1}$ para todo $0 \leq i < k - 1$ y $s_{k-1} = 0$
- **Solución corta:**
  - Observar que cada $s_i$ toma dos posibles valores, y cuál de ellos toma depende del valor de $c$.
  - Así, llegar a las siguientes expresiones:
    - $s_0 = c \cdot 0 + \overline{c} \cdot e_1$
    - $s_1 = c \cdot e_0 + \overline{c} \cdot e_2$
    - $s_2 = c \cdot e_1 + \overline{c} \cdot 0$

# Ejercicio 3 - Solución larga

| $c$ | $e_2$ | $e_1$ | $e_0$ | $s_2$ | $s_1$ | $s_0$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

# Ejercicio 3 - Solución larga

$$
\begin{aligned}
s_0 &= \overline{c}.\overline{e_2}.e_1.\overline{e_0} + \overline{c}.\overline{e_2}.e_1.e_0 + \overline{c}.e_2.e_1.\overline{e_0} + \overline{c}.e_2.e_1.e_0 \\
s_0 &= \overline{c}.(\overline{e_2}.e_1.\overline{e_0} + \overline{e_2}.e_1.e_0 + e_2.e_1.\overline{e_0} + e_2.e_1.e_0) \\
s_0 &= \overline{c}.(e_1.(\overline{e_2}.\overline{e_0} + \overline{e_2}.e_0 + e_2.\overline{e_0} + e_2.e_0)) \\
s_0 &= \overline{c}.(e_1.(\overline{e_2}.(\overline{e_0} + e_0) + e_2.(\overline{e_0} + e_0))) \\
s_0 &= \overline{c}.(e_1.(\overline{e_2}. + e_2)) \\
s_0 &= \overline{c}.e_1
\end{aligned}
$$

$$
\begin{aligned}
s_2 &= c.\overline{e_2}.e_1.\overline{e_0} + c.\overline{e_2}.e_1.e_0 + c.e_2.e_1.\overline{e_0} + c.e_2.e_1.e_0 \\
s_2 &= c.(\overline{e_2}.e_1.\overline{e_0} + \overline{e_2}.e_1.e_0 + e_2.e_1.\overline{e_0} + e_2.e_1.e_0) \\
s_2 &= c.(e_1.(\overline{e_2}.\overline{e_0} + \overline{e_2}.e_0 + e_2.\overline{e_0} + e_2.e_0)) \\
s_2 &= c.(e_1.(\overline{e_2}.(\overline{e_0} + e_0) + e_2.(\overline{e_0} + e_0))) \\
s_2 &= c.(e_1.(\overline{e_2}. + e_2)) \\
s_2 &= c.e_1
\end{aligned}
$$

# Ejercicio 3 - Solución larga

$$
\begin{aligned}
s_1 = \quad & \overline{c}.e_2.\overline{e_1}.\overline{e_0} + \overline{c}.e_2.\overline{e_1}.e_0 + \overline{c}.e_2.e_1.\overline{e_0} + \overline{c}.e_2.e_1.e_0 + \\
& c.\overline{e_2}.\overline{e_1}.e_0 + c.\overline{e_2}.e_1.e_0 + c.e_2.\overline{e_1}.e_0 + c.e_2.e_1.e_0
\end{aligned}
$$

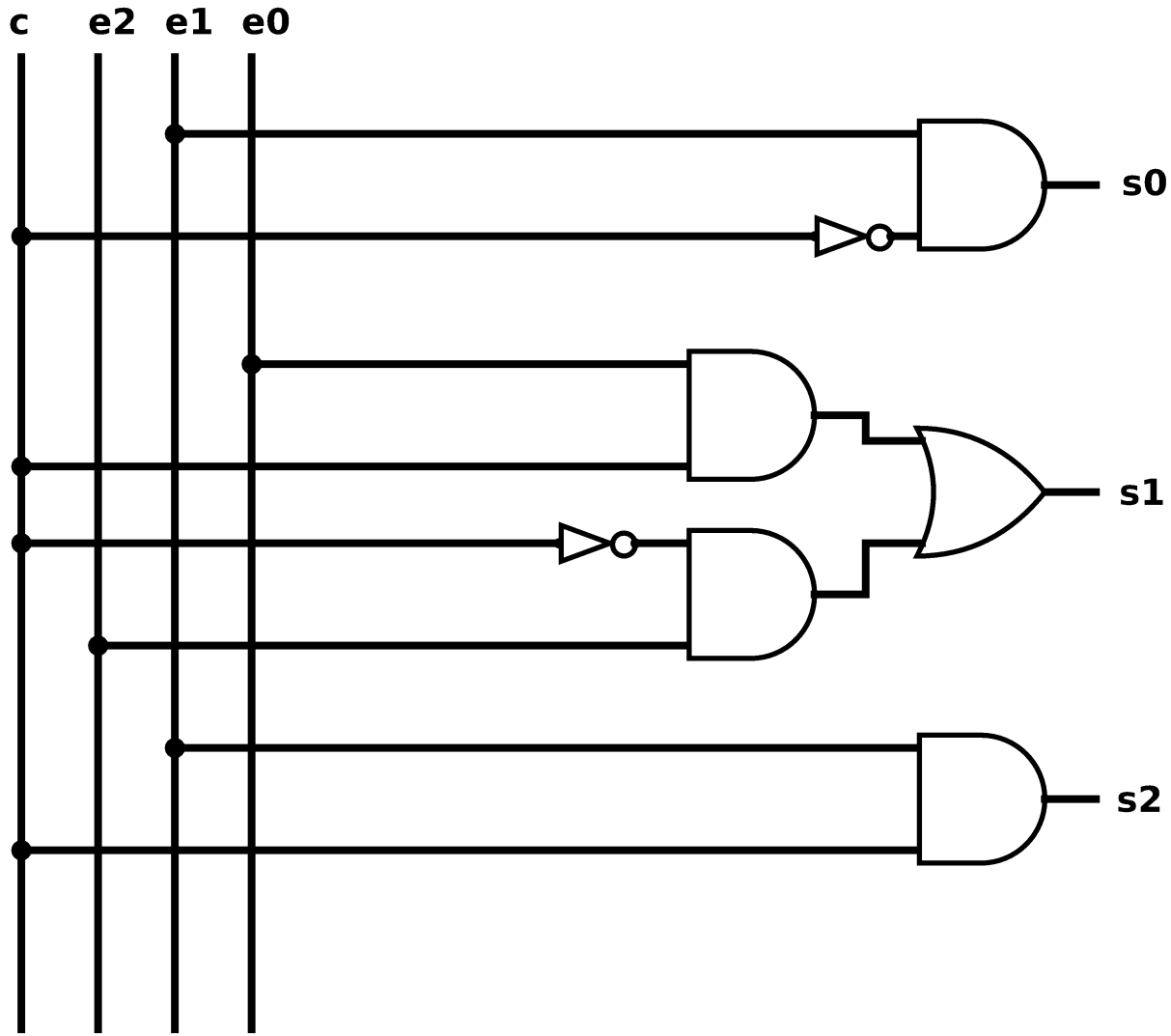
$$
\begin{aligned}
s_1 = \quad & \overline{c}.(e_2.\overline{e_1}.\overline{e_0} + e_2.\overline{e_1}.e_0 + e_2.e_1.\overline{e_0} + e_2.e_1.e_0) + \\
& c.(\overline{e_2}.\overline{e_1}.e_0 + \overline{e_2}.e_1.e_0 + e_2.\overline{e_1}.e_0 + e_2.e_1.e_0)
\end{aligned}
$$

$$
\begin{aligned}
s_1 = \quad & \overline{c}.(e_2.(\overline{e_1}.\overline{e_0} + \overline{e_1}.e_0 + e_1.\overline{e_0} + e_1.e_0)) + \\
& c.(e_0.(\overline{e_2}.\overline{e_1} + \overline{e_2}.e_1 + e_2.\overline{e_1} + e_2.e_1))
\end{aligned}
$$

$$
\begin{aligned}
s_1 = \quad & \overline{c}.(e_2.(\overline{e_1}.(\overline{e_0} + e_0) + e_1.(\overline{e_0} + e_0))) + \\
& c.(e_0.(\overline{e_2}.(\overline{e_1} + e_1) + e_2.(\overline{e_1} + e_1)))
\end{aligned}
$$

$$
\begin{aligned}
s_1 = \quad & \overline{c}.(e_2.(\overline{e_1} + e_1)) + c.(e_0.(\overline{e_2} + e_2)) \\
s_1 = \quad & \overline{c}.e_2 + c.e_0
\end{aligned}
$$

# Ejercicio 3 - Implementación



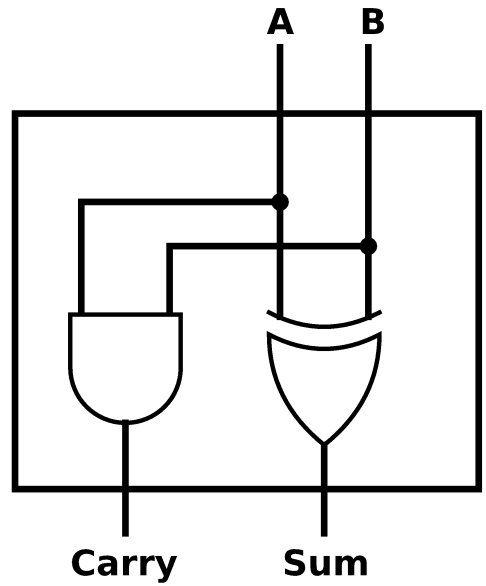
# Ejercicio 4

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

# Ejercicio 4

- Armar un **sumador de 1 bit**. Debe tener dos entradas de un bit y dos salidas, una para el resultado y otra para indicar si hubo o no acarreo.
- **Solución corta:**
  - Armar la tabla de verdad y observar que se compone de dos tablas conocidas: XOR y AND.

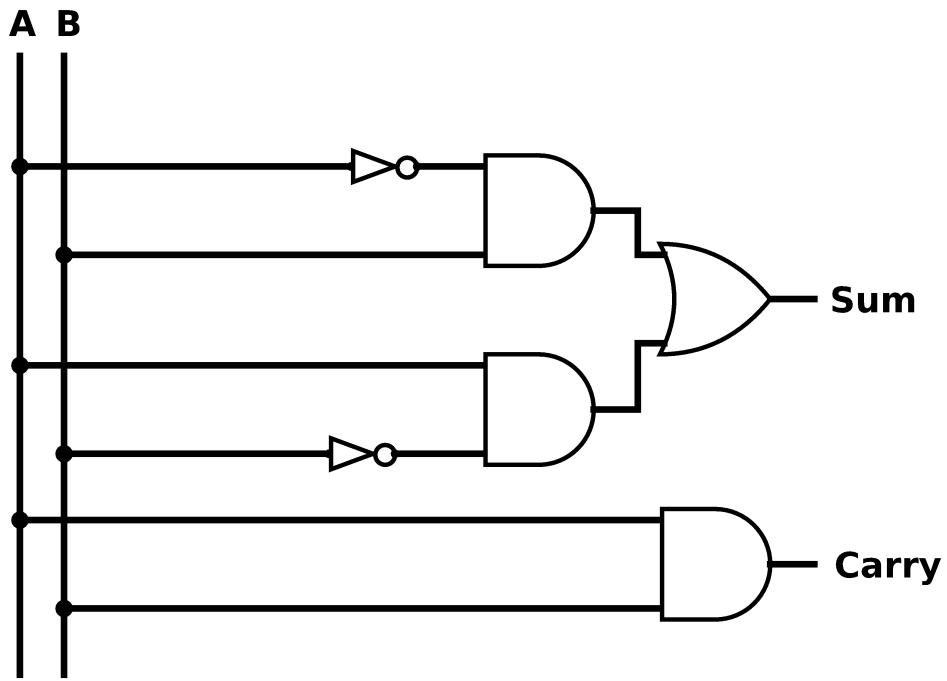
| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |



# Ejercicio 4 - Solución por suma de productos

| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$$Sum = \overline{A} \cdot B + A \cdot \overline{B}$$
$$Carry = A \cdot B$$

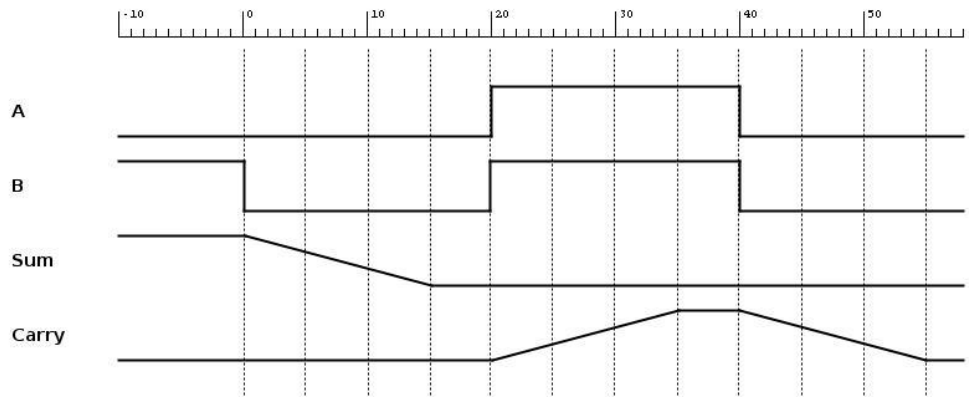


# Ejercicio 4 - Segunda parte

- Se tiene un **sumador simple de 1 bit** cuyas entradas A y B valen 0 y 1 respectivamente. En el tiempo cero, ambas pasan a valer 0; a los 20ns ambas pasan a valer 1; 20ns más tarde vuelven a valer 0 y así sucesivamente.
- Sabiendo que las compuertas AND y XOR tienen un retardo de 15ns, realizar el diagrama de tiempos del circuito.

# Ejercicio 4 - Segunda parte

- Se tiene un **sumador simple de 1 bit** cuyas entradas A y B valen 0 y 1 respectivamente. En el tiempo cero, ambas pasan a valer 0; a los 20ns ambas pasan a valer 1; 20ns más tarde vuelven a valer 0 y así sucesivamente.
- Sabiendo que las compuertas AND y XOR tienen un retardo de 15ns, realizar el diagrama de tiempos del circuito.
- **Solución:**



# Ejercicio 5

- Usando dos sumadores simples y una compuerta a elección, armar un sumador completo. Tiene dos entradas de 1 bit y una tercera intepretada como $C_{In}$, tiene como salida $C_{Out}$ y Sum.

# Ejercicio 5

- Usando dos sumadores simples y una compuerta a elección, armar un sumador completo. Tiene dos entradas de 1 bit y una tercera intepretada como $C_{In}$, tiene como salida $C_{Out}$ y Sum.
- **Solución:**

| $Cin$ | A | B | Sum | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Ejercicio 5 - Solución

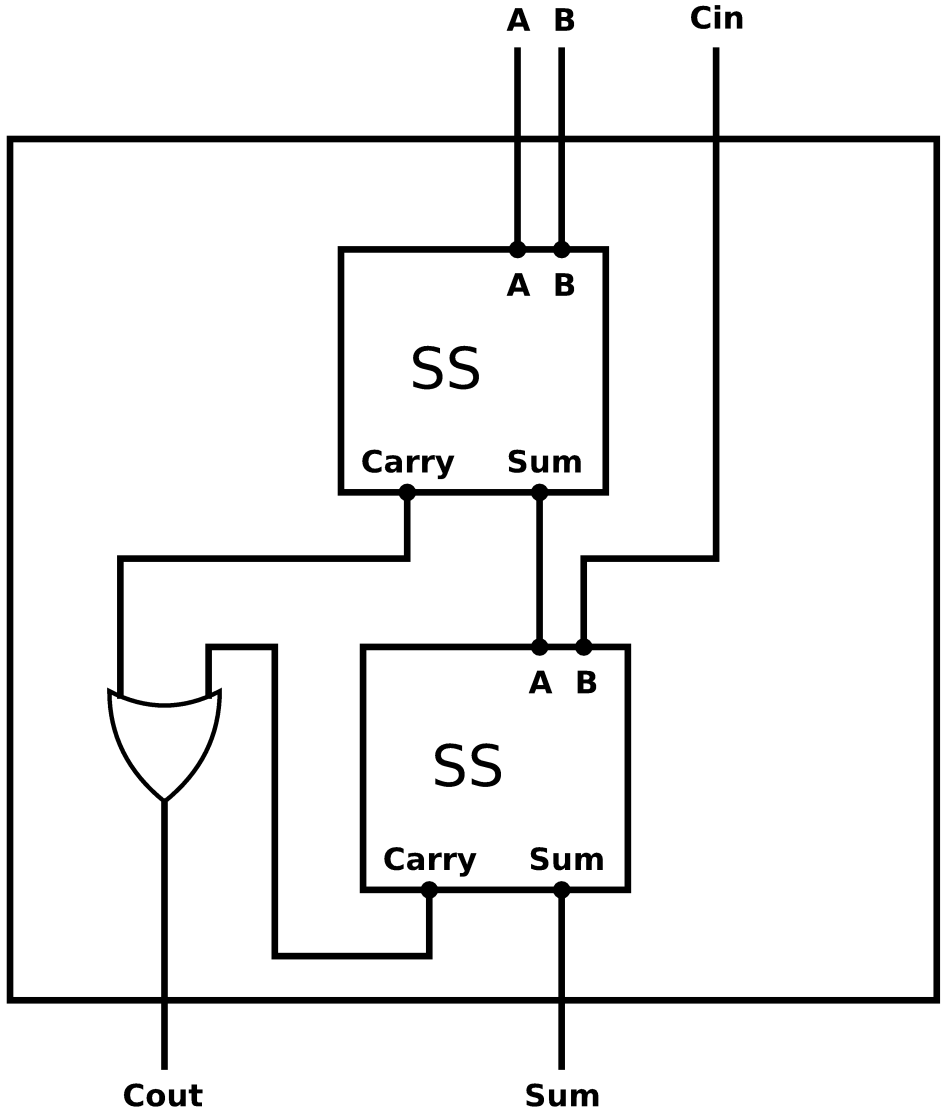
$$
\begin{aligned}
\text{Sum} &= \overline{C_{in}}.\overline{A}.B + \overline{C_{in}}.A.\overline{B} + C_{in}.\overline{A}.\overline{B} + C_{in}.A.B \\
\text{Sum} &= \overline{C_{in}}.(\overline{A}.B + A.\overline{B}) + C_{in}.(\overline{A}.\overline{B} + A.B) \\
\text{Sum} &= \overline{C_{in}}.(A \oplus B) + C_{in}.(\overline{A \oplus B}) \\
\text{Sum} &= C_{in} \oplus (A \oplus B)
\end{aligned}
$$

$$
\text{Sum} = Sum(C_{in} + Sum(A + B))
$$

$$
\begin{aligned}
C_{out} &= \overline{C_{in}}.A.B + C_{in}.\overline{A}.B + C_{in}.A.\overline{B} + C_{in}.A.B \\
C_{out} &= \overline{C_{in}}.A.B + C_{in}.(\overline{A}.B + A.\overline{B} + A.B) \\
C_{out} &= \overline{C_{in}}.A.B + C_{in}.(A \oplus B + A.B) \\
C_{out} &= \overline{C_{in}}.A.B + C_{in}.(A \oplus B) + C_{in}.A.B \\
C_{out} &= A.B + C_{in}.(A \oplus B)
\end{aligned}
$$

$$
C_{out} = Carry(A + B) + Carry(C_{in} + Sum(A + B))
$$

# Ejercicio 5 - Implementación



# ¿Qué sigue?

- **La práctica:**
  - Con lo visto pueden realizar la primera parte de la Práctica 2.
  - Pueden probar sus circuitos en Logisim.
    - sudo apt install logisim
    - O lo bajan desde http://www.cburch.com/logisim/

- **Taller:**
  - El martes siguiente tenemos el primer taller de la materia.
  - Será sobre lo visto hasta hoy inclusive.
  - Vamos a usar Logisim.