Álgebra de Boole y Circuitos Combinatorios

David Alejandro González Márquez

Clase disponible en: `https://github.com/fokerman/computingSystemsCourse`

Circuitos Combinatorios

Usando **compuertas** podemos construir un circuito equivalente a una **función booleana**.

A continuación vamos a construir circuitos para realizar **tareas** específicas.

1. Circuitos Codificadores y Decodificadores
2. Circuitos Multiplexores y Demultiplexores
3. Circuitos Aritméticos

Circuitos Decodificadores y Codificadores

Decodificador

Circuito que toma *n* entradas (e_0 a e_{n-1}) y genera 2^n salidas (s_0 a s_{2^n-1}).

Coloca un 1 en la salida codificada por la entrada.
El resto de las salidas quedan en 0.

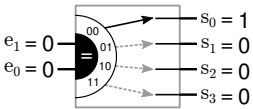
Circuitos Decodificadores y Codificadores

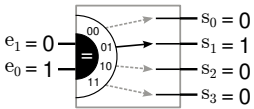
Decodificador

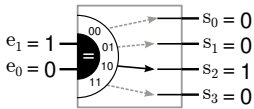
Circuito que toma *n* entradas (e_0 a e_{n-1}) y genera 2^n salidas (s_0 a s_{2^n-1}).

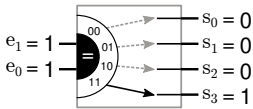
Coloca un 1 en la salida codificada por la entrada.
El resto de las salidas quedan en 0.

Ejemplo - Decodificador de 2 entradas



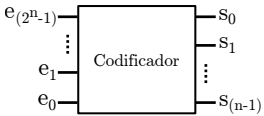






Circuitos Decodificadores y Codificadores

Codificador



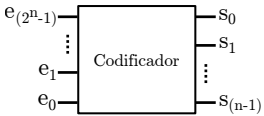
Circuito que toma 2^n entradas (e_0 a e_{2^n-1}) y genera *n* salidas (s_0 a s_{n-1}).

Expone en las salidas la codificación de la única entrada en 1.
Este circuito no permite que más de una entrada esté en 1.

Circuitos Decodificadores y Codificadores

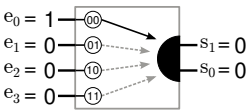
Codificador

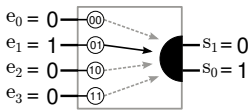
Circuito que toma 2^n entradas (e_0 a e_{2^n-1}) y genera *n* salidas (s_0 a s_{n-1}).

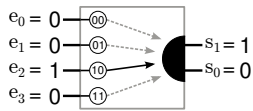


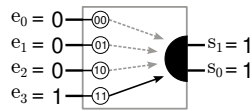
Expone en las salidas la codificación de la única entrada en 1.
Este circuito no permite que más de una entrada esté en 1.

Ejemplo - Codificador de 4 entradas





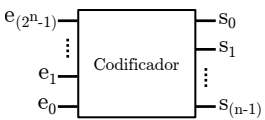




Circuitos Decodificadores y Codificadores

Decodificador

Codificador

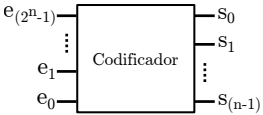


Circuitos Decodificadores y Codificadores

Decodificador

entradas		salida			
E_1	E_0	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

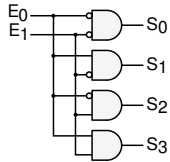
Codificador



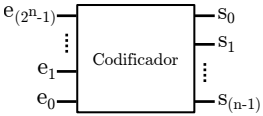
Circuitos Decodificadores y Codificadores

Decodificador

entradas		salida			
E_1	E_0	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



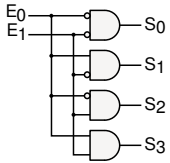
Codificador



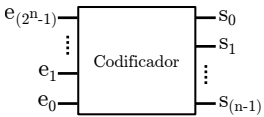
Circuitos Decodificadores y Codificadores

Decodificador

entradas		salida			
E_1	E_0	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Codificador

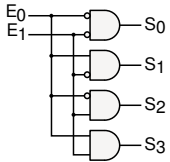


entradas				salidas	
E_0	E_1	E_2	E_3	S_1	S_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

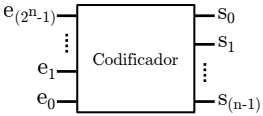
Circuitos Decodificadores y Codificadores

Decodificador

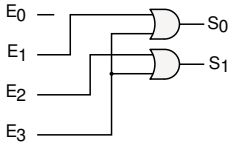
entradas		salida			
E_1	E_0	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Codificador

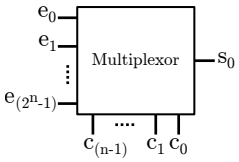


entradas				salidas	
E_0	E_1	E_2	E_3	S_1	S_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



Circuitos Multiplexores y Demultiplexores

Multiplexor

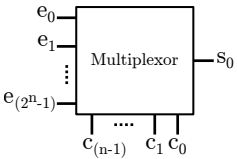


Circuito que toma *n* entradas de control (c_0 a c_{n-1}), 2^n entradas de datos (e_0 a e_{2^n-1}) y genera una salida s_0.

Dependiendo del valor de las entradas de control, selecciona una de las entradas de datos y la expone en la salida.

Circuitos Multiplexores y Demultiplexores

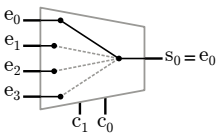
Multiplexor

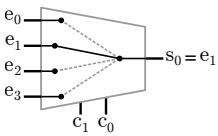


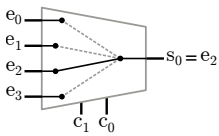
Circuito que toma *n* entradas de control (c_0 a c_{n-1}), 2^n entradas de datos (e_0 a e_{2^n-1}) y genera una salida s_0.

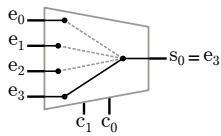
Dependiendo del valor de las entradas de control, selecciona una de las entradas de datos y la expone en la salida.

Ejemplo - Multiplexor de 4 entradas



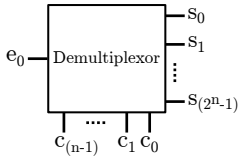






Circuitos Multiplexores y Demultiplexores

Demultiplexor

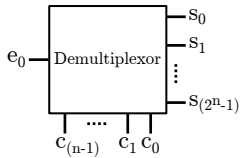


Circuito que toma *n* entradas de control (c_0 a c_{n-1}), una entrada de datos e_0 y genera 2^n salidas (s_0 a s_{2^n-1}).

Expone la entrada en una de las salidas, dependiendo del valor de las entradas de control.

Circuitos Multiplexores y Demultiplexores

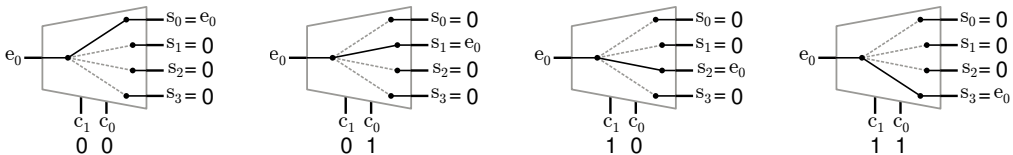
Demultiplexor



Circuito que toma *n* entradas de control (c_0 a c_{n-1}), una entrada de datos e_0 y genera 2^n salidas (s_0 a s_{2^n-1}).

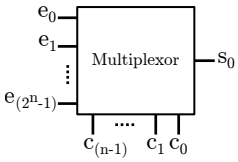
Expone la entrada en una de las salidas, dependiendo del valor de las entradas de control.

Ejemplo - Demultiplexor de 4 entradas

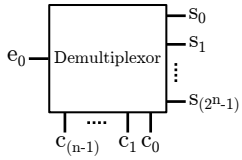


Circuitos Multiplexores y Demultiplexores

Multiplexor

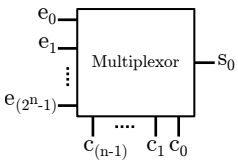


Demultiplexor



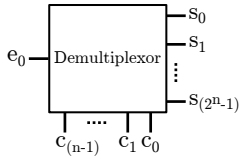
Circuitos Multiplexores y Demultiplexores

Multiplexor



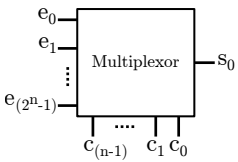
entradas						salida
E_0	E_1	E_2	E_3	C_1	C_0	S_0
e_0	e_1	e_2	e_3	0	0	e_0
e_0	e_1	e_2	e_3	0	1	e_1
e_0	e_1	e_2	e_3	1	0	e_2
e_0	e_1	e_2	e_3	1	1	e_3

Demultiplexor

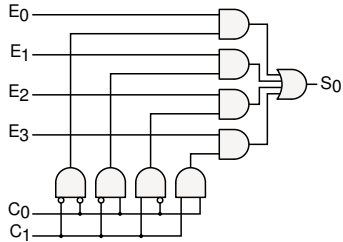


Circuitos Multiplexores y Demultiplexores

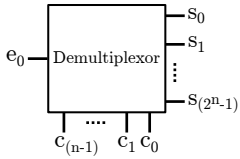
Multiplexor



entradas						salida
E_0	E_1	E_2	E_3	C_1	C_0	S_0
e_0	e_1	e_2	e_3	0	0	e_0
e_0	e_1	e_2	e_3	0	1	e_1
e_0	e_1	e_2	e_3	1	0	e_2
e_0	e_1	e_2	e_3	1	1	e_3

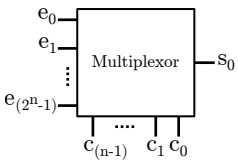


Demultiplexor

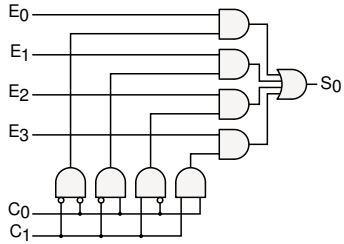


Circuitos Multiplexores y Demultiplexores

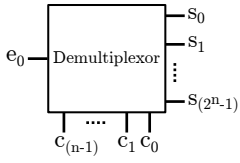
Multiplexor



entradas						salida
E_0	E_1	E_2	E_3	C_1	C_0	S_0
e_0	e_1	e_2	e_3	0	0	e_0
e_0	e_1	e_2	e_3	0	1	e_1
e_0	e_1	e_2	e_3	1	0	e_2
e_0	e_1	e_2	e_3	1	1	e_3



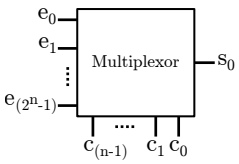
Demultiplexor



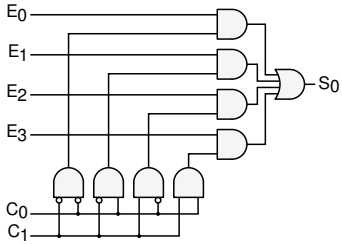
entradas			salidas			
E_0	C_1	C_0	S_0	S_1	S_2	S_3
e_0	0	0	e_0	0	0	0
e_0	0	1	0	e_0	0	0
e_0	1	0	0	0	e_0	0
e_0	1	1	0	0	0	e_0

Circuitos Multiplexores y Demultiplexores

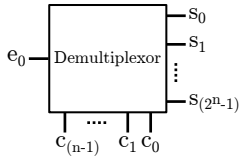
Multiplexor



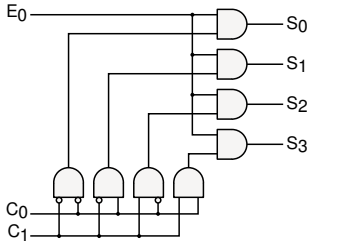
entradas						salida
E_0	E_1	E_2	E_3	C_1	C_0	S_0
e_0	e_1	e_2	e_3	0	0	e_0
e_0	e_1	e_2	e_3	0	1	e_1
e_0	e_1	e_2	e_3	1	0	e_2
e_0	e_1	e_2	e_3	1	1	e_3



Demultiplexor

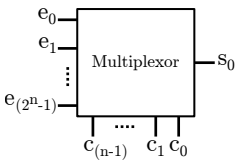


entradas			salidas			
E_0	C_1	C_0	S_0	S_1	S_2	S_3
e_0	0	0	e_0	0	0	0
e_0	0	1	0	e_0	0	0
e_0	1	0	0	0	e_0	0
e_0	1	1	0	0	0	e_0

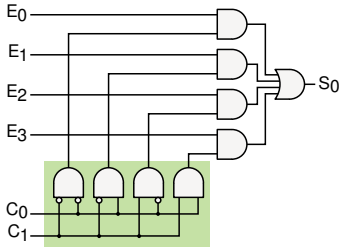


Circuitos Multiplexores y Demultiplexores

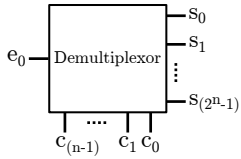
Multiplexor



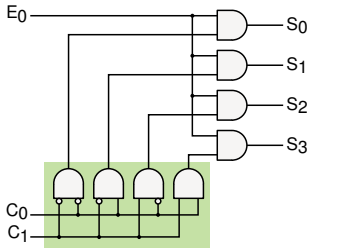
entradas						salida
E_0	E_1	E_2	E_3	C_1	C_0	S_0
e_0	e_1	e_2	e_3	0	0	e_0
e_0	e_1	e_2	e_3	0	1	e_1
e_0	e_1	e_2	e_3	1	0	e_2
e_0	e_1	e_2	e_3	1	1	e_3



Demultiplexor



entradas			salidas			
E_0	C_1	C_0	S_0	S_1	S_2	S_3
e_0	0	0	e_0	0	0	0
e_0	0	1	0	e_0	0	0
e_0	1	0	0	0	e_0	0
e_0	1	1	0	0	0	e_0



Circuitos Aritméticos

Conjunto de circuitos que nos permiten realizar **operaciones matemáticas**.

Circuitos Aritméticos

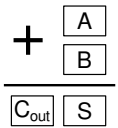
Conjunto de circuitos que nos permiten realizar **operaciones matemáticas**.

Sumador Simple

Circuitos Aritméticos

Conjunto de circuitos que nos permiten realizar **operaciones matemáticas**.

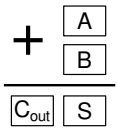
Sumador Simple



Circuitos Aritméticos

Conjunto de circuitos que nos permiten realizar **operaciones matemáticas**.

Sumador Simple

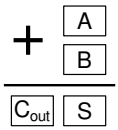


Sumador simple de 1 bit, nos permite sumar dos bits A y B.
El resultado se expresa como S y el carry como C_{out}.

Circuitos Aritméticos

Conjunto de circuitos que nos permiten realizar **operaciones matemáticas**.

Sumador Simple



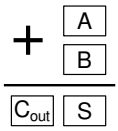
A	*B*	C_{out}	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Sumador simple de 1 bit, nos permite sumar dos bits A y B.
El resultado se expresa como S y el carry como C_{out}.

Circuitos Aritméticos

Conjunto de circuitos que nos permiten realizar **operaciones matemáticas**.

Sumador Simple



A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

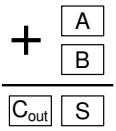
$$C_{out} = A \cdot B$$
$$S = (\overline{A} \cdot B) + (A \cdot \overline{B}) = A \oplus B$$

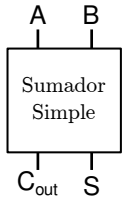
Sumador simple de 1 bit, nos permite sumar dos bits A y B.
El resultado se expresa como S y el carry como C_{out}.

Circuitos Aritméticos

Conjunto de circuitos que nos permiten realizar **operaciones matemáticas**.

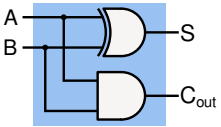
Sumador Simple

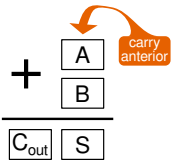




A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$C_{out} = A \cdot B$$
$$S = (\overline{A} \cdot B) + (A \cdot \overline{B}) = A \oplus B$$



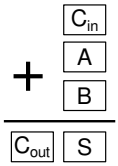


Sumador simple de 1 bit, nos permite sumar dos bits A y B.
El resultado se expresa como S y el carry como C_{out}.

Si queremos sumar más de un bit, necesitamos también sumar el carry de la operación anterior.

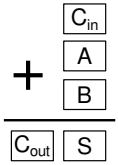
Circuitos Aritméticos

Sumador Completo

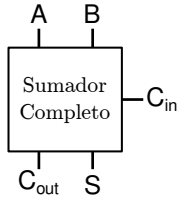


Circuitos Aritméticos

Sumador Completo

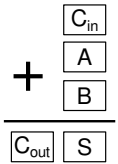


Un sumador completo de 1 bit, nos permite sumar dos bits A y B, y además el carry de la operación anterior C_{in}. El resultado se expresa como S y un carry C_{out}.

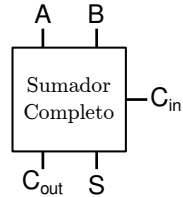


Circuitos Aritméticos

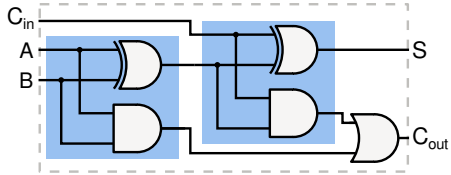
Sumador Completo



Un sumador completo de 1 bit, nos permite sumar dos bits A y B, y además el carry de la operación anterior C_{in}. El resultado se expresa como S y un carry C_{out}.



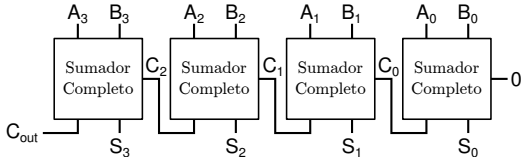
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Circuitos Aritméticos

Sumador de 4 bits

Utilizando 4 sumadores completos, podemos construir un sumador de 4 bits.



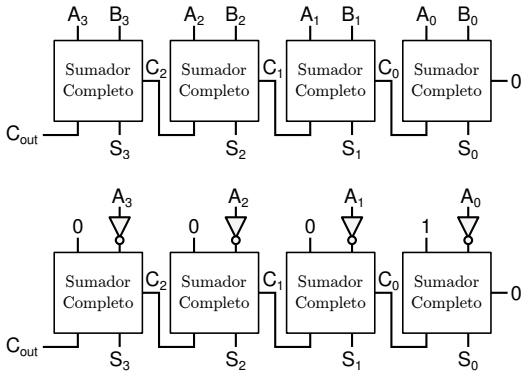
Circuitos Aritméticos

Sumador de 4 bits

Utilizando 4 sumadores completos, podemos construir un sumador de 4 bits.

Inversor aditivo de 4 bits

Si negamos la entrada y le sumamos uno. Obtenemos el inverso aditivo de un número en complemento a 2.



Circuitos Aritméticos

Sumador de 4 bits

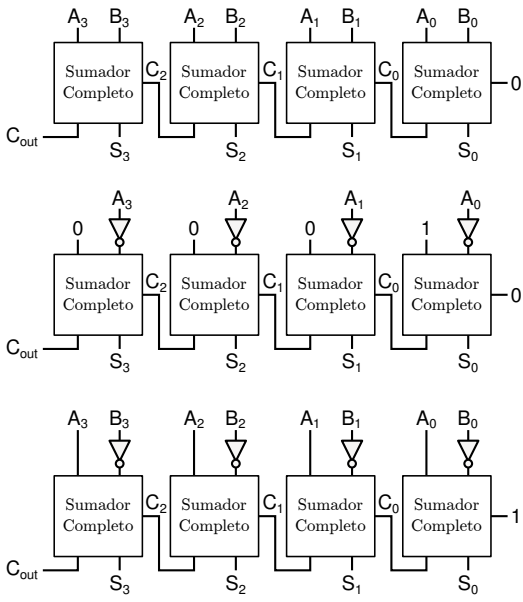
Utilizando 4 sumadores completos, podemos construir un sumador de 4 bits.

Inversor aditivo de 4 bits

Si negamos la entrada y le sumamos uno. Obtenemos el inverso aditivo de un número en complemento a 2.

Restador de 4 bits

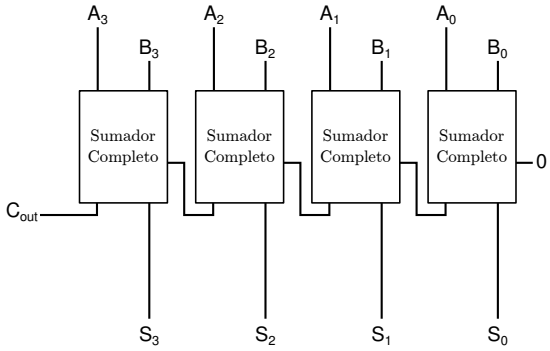
Si sumamos un número con el inverso aditivo de otro obtenemos un circuito restador.



Circuitos Aritméticos - Flags

Los circuitos aritméticos además de generar resultados de sus operaciones, también generan lo que se conoce como **palabra de estado**.

La palabra de estado contiene una serie de **Flags**.

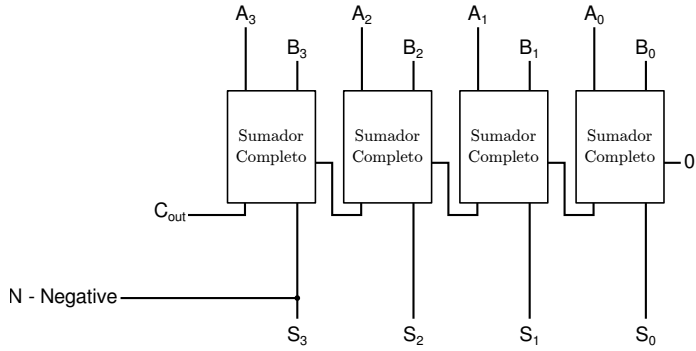


Circuitos Aritméticos - Flags

Los circuitos aritméticos además de generar resultados de sus operaciones, también generan lo que se conoce como **palabra de estado**.

La palabra de estado contiene una serie de **Flags**.

Vamos a ver algunos de ellos y cómo se construyen.



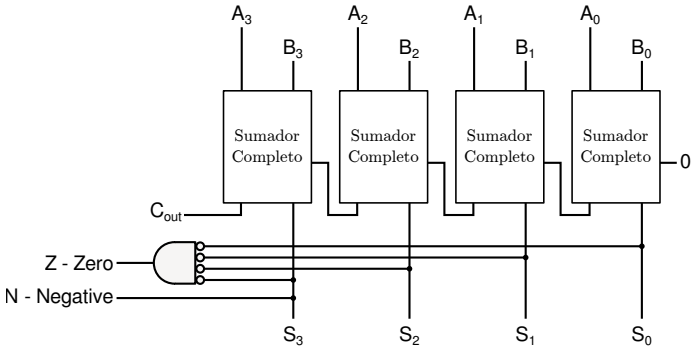
N	Negative	Indica si el número es negativo en complemento a 2.

Circuitos Aritméticos - Flags

Los circuitos aritméticos además de generar resultados de sus operaciones, también generan lo que se conoce como **palabra de estado**.

La palabra de estado contiene una serie de **Flags**.

Vamos a ver algunos de ellos y cómo se construyen.



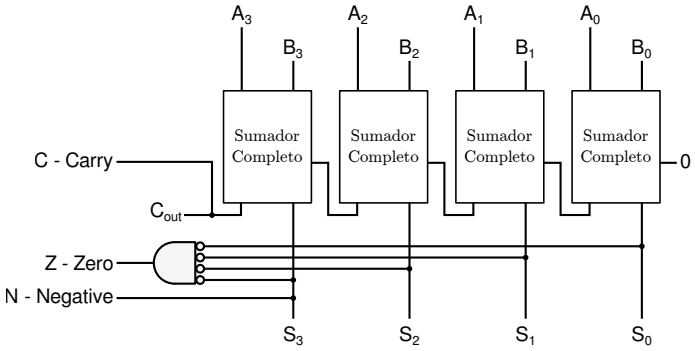
N	Negative	Indica si el número es negativo en complemento a 2.
Z	Zero	Indica si el número es cero en complemento a 2.

Circuitos Aritméticos - Flags

Los circuitos aritméticos además de generar resultados de sus operaciones, también generan lo que se conoce como **palabra de estado**.

La palabra de estado contiene una serie de **Flags**.

Vamos a ver algunos de ellos y cómo se construyen.



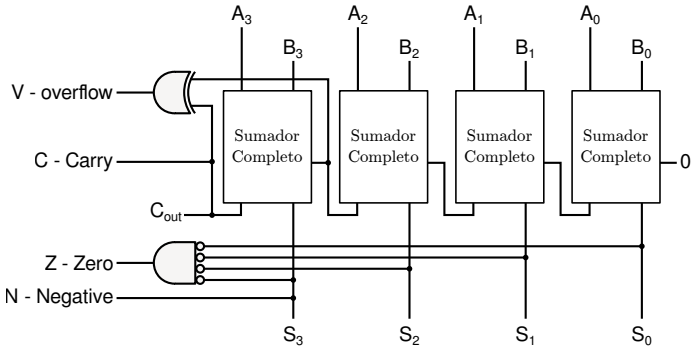
N	Negative	Indica si el número es negativo en complemento a 2.
Z	Zero	Indica si el número es cero en complemento a 2.
C	Carry	Indica si la operación en complemento a 2 genera acarreo.

Circuitos Aritméticos - Flags

Los circuitos aritméticos además de generar resultados de sus operaciones, también generan lo que se conoce como **palabra de estado**.

La palabra de estado contiene una serie de **Flags**.

Vamos a ver algunos de ellos y cómo se construyen.



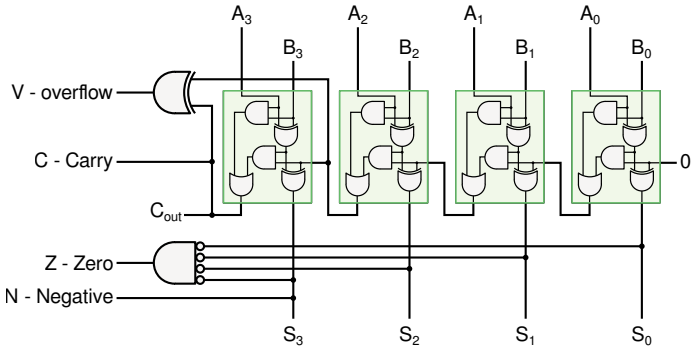
N	Negative	Indica si el número es negativo en complemento a 2.
Z	Zero	Indica si el número es cero en complemento a 2.
C	Carry	Indica si la operación en complemento a 2 genera acarreo.
V	Overflow	Indica si el resultado no es representable en complemento a 2.

Circuitos Aritméticos - Flags

Los circuitos aritméticos además de generar resultados de sus operaciones, también generan lo que se conoce como **palabra de estado**.

La palabra de estado contiene una serie de **Flags**.

Vamos a ver algunos de ellos y cómo se construyen.



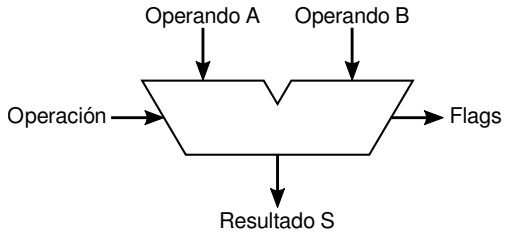
N	Negative	Indica si el número es negativo en complemento a 2.
Z	Zero	Indica si el número es cero en complemento a 2.
C	Carry	Indica si la operación en complemento a 2 genera acarreo.
V	Overflow	Indica si el resultado no es representable en complemento a 2.

Circuitos Aritméticos - ALU

La **ALU** (*Arithmetic logic unit*) es un circuito combinatorio que toma como entrada operandos e indicaciones de operaciones a realizar, y retorna el resultado de la operación junto con sus flags.

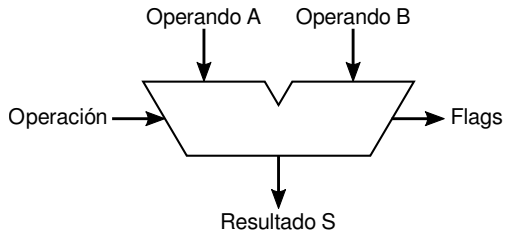
Circuitos Aritméticos - ALU

La **ALU** (*Arithmetic logic unit*) es un circuito combinatorio que toma como entrada operandos e indicaciones de operaciones a realizar, y retorna el resultado de la operación junto con sus flags.



Circuitos Aritméticos - ALU

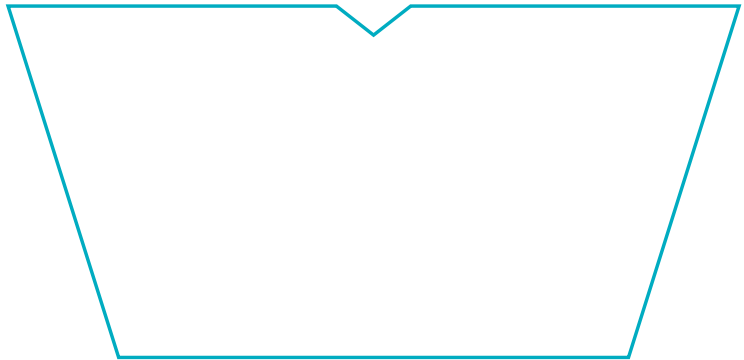
La **ALU** (*Arithmetic logic unit*) es un circuito combinatorio que toma como entrada operandos e indicaciones de operaciones a realizar, y retorna el resultado de la operación junto con sus flags.



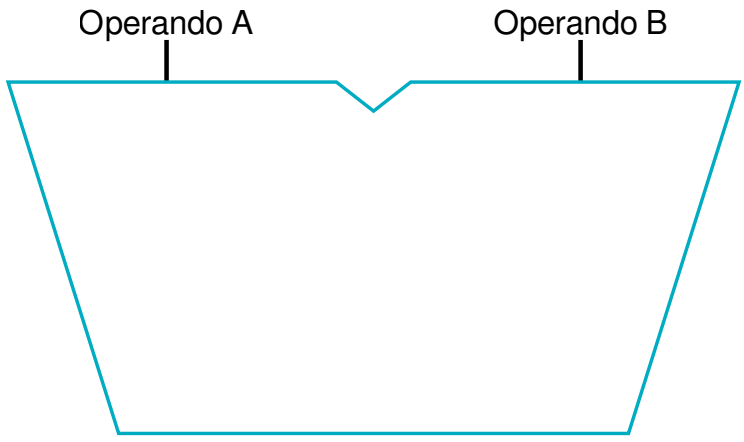
Ejemplo:

Operando A	Operando B	Operación	Resultado y Flags
A	B	$+$	$A + B$
A	B	*and*	A *and* B
A	B	*or*	A *or* B
A		*not*	$\overline{A}$

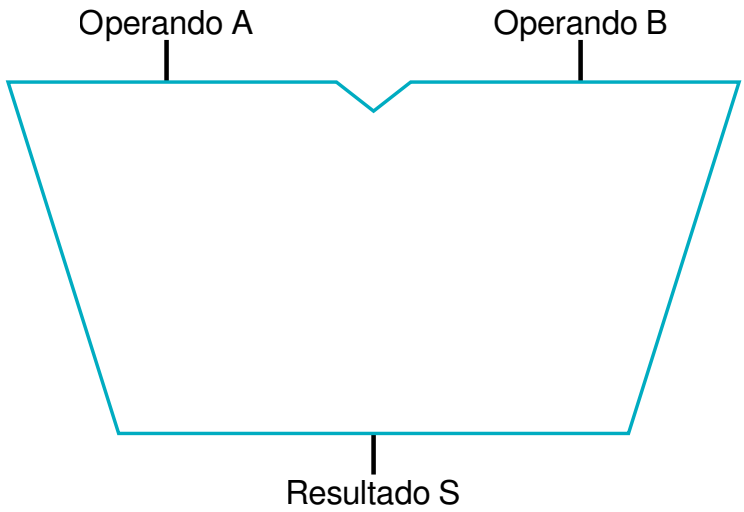
Circuitos Aritméticos - ALU de 1 bit



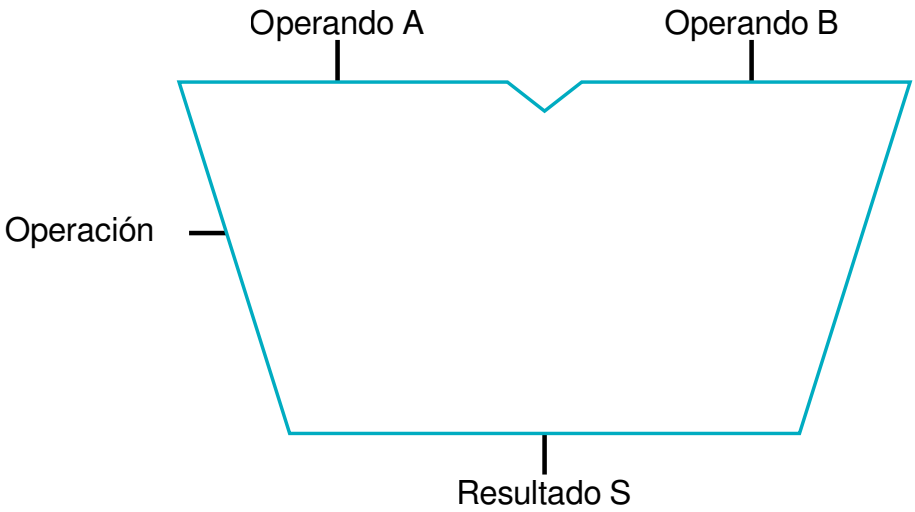
Circuitos Aritméticos - ALU de 1 bit



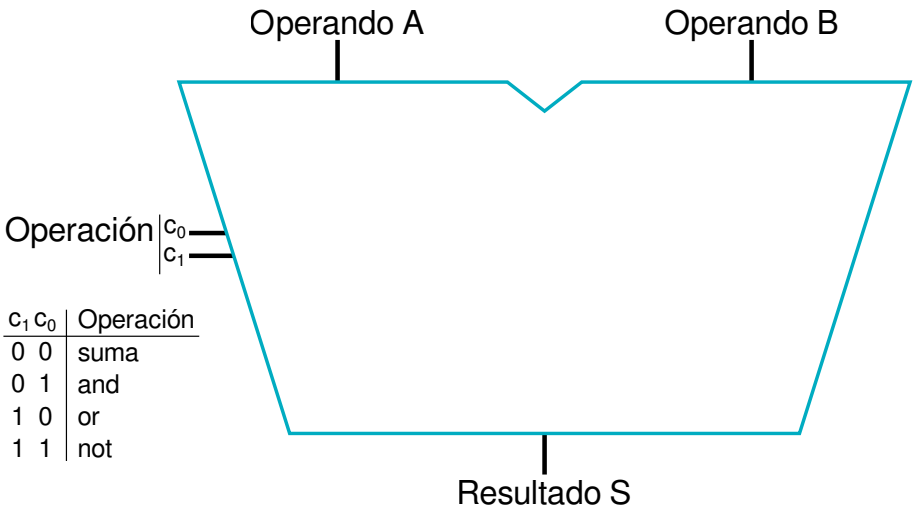
Circuitos Aritméticos - ALU de 1 bit



Circuitos Aritméticos - ALU de 1 bit

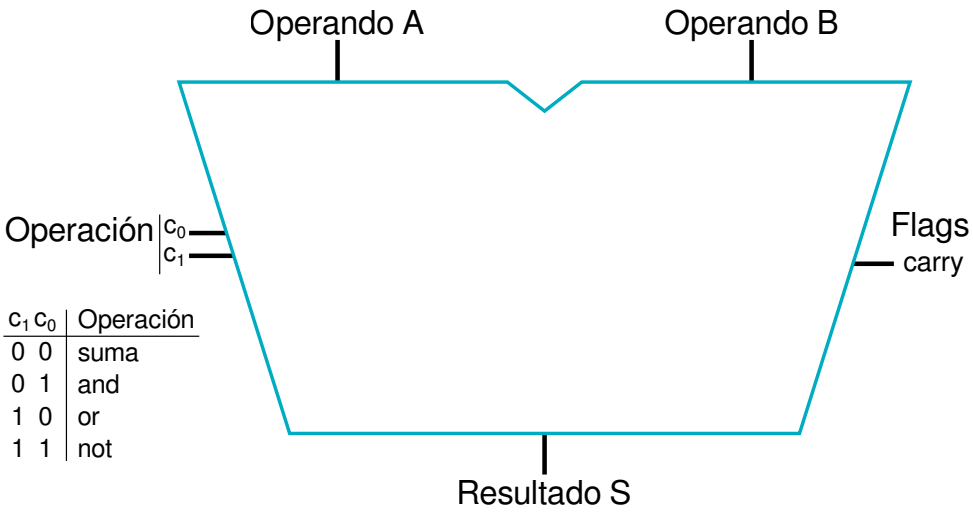


Circuitos Aritméticos - ALU de 1 bit



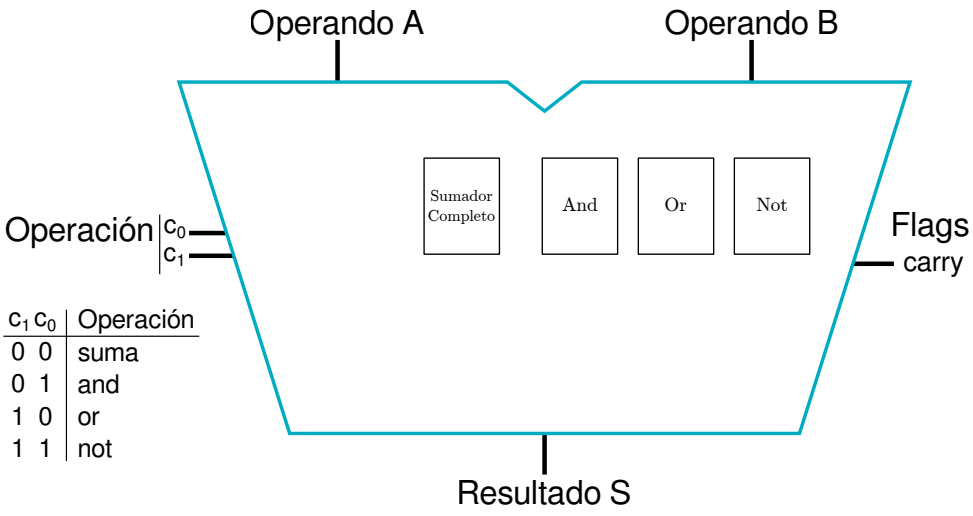
c_1	c_0	Operación
0	0	suma
0	1	and
1	0	or
1	1	not

Circuitos Aritméticos - ALU de 1 bit



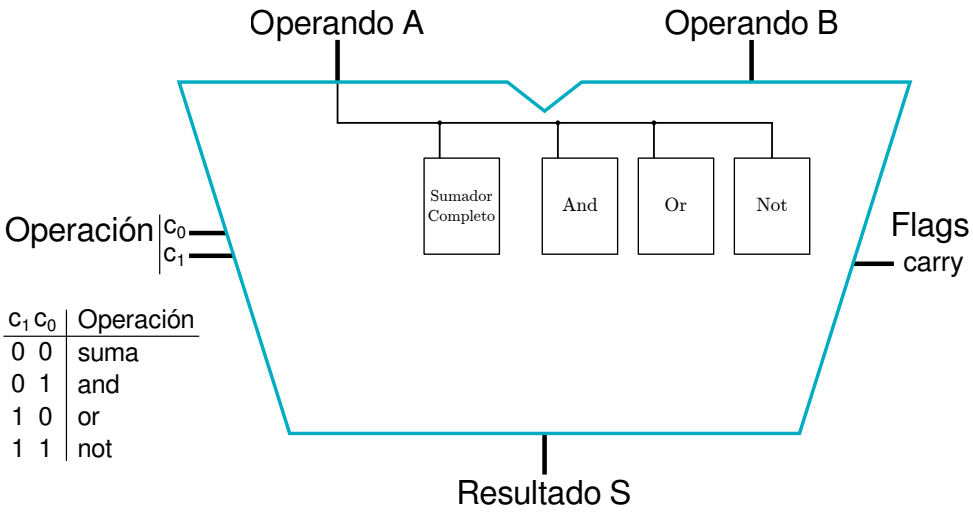
c_1	c_0	Operación
0	0	suma
0	1	and
1	0	or
1	1	not

Circuitos Aritméticos - ALU de 1 bit



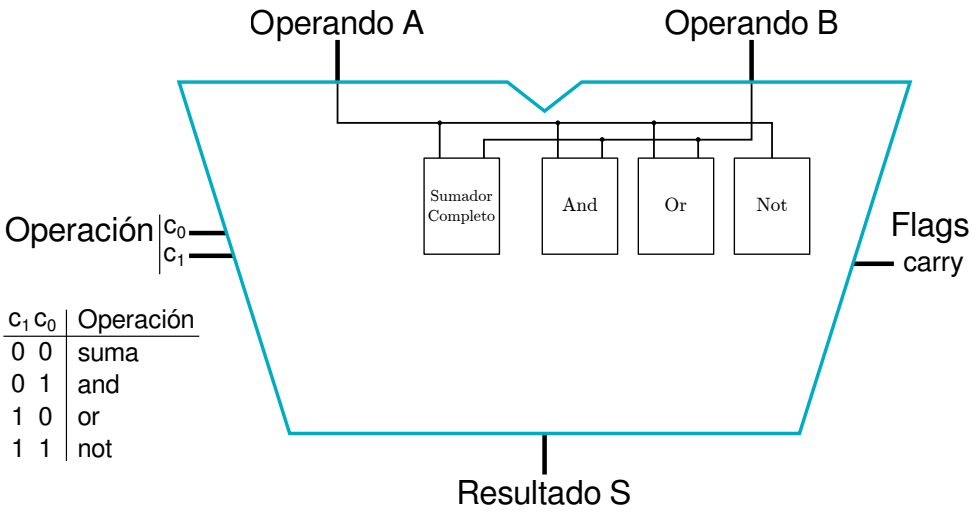
c_1	c_0	Operación
0	0	suma
0	1	and
1	0	or
1	1	not

Circuitos Aritméticos - ALU de 1 bit



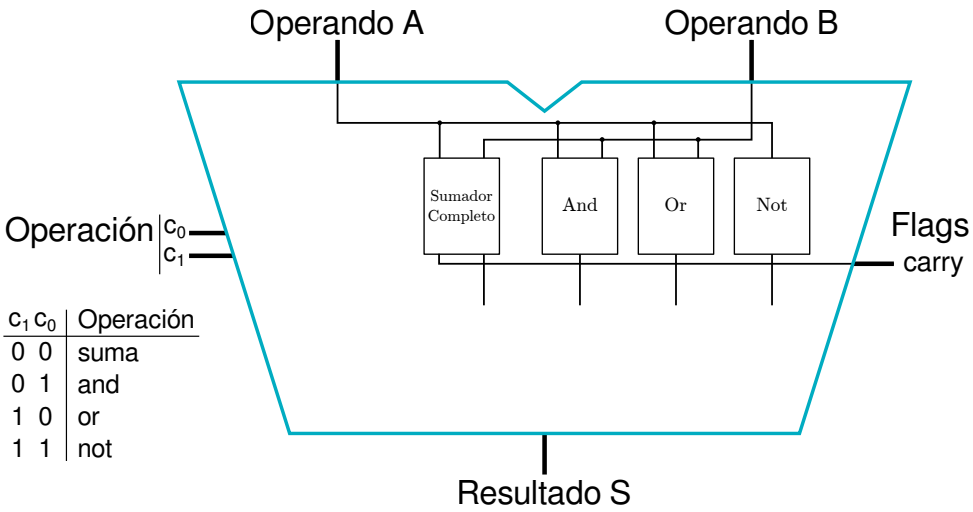
c_1	c_0	Operación
0	0	suma
0	1	and
1	0	or
1	1	not

Circuitos Aritméticos - ALU de 1 bit



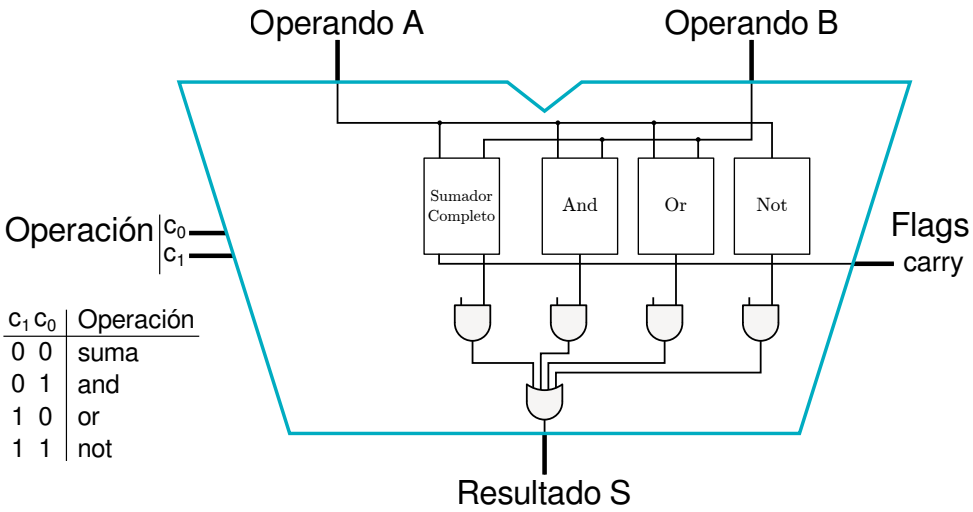
c_1	c_0	Operación
0	0	suma
0	1	and
1	0	or
1	1	not

Circuitos Aritméticos - ALU de 1 bit



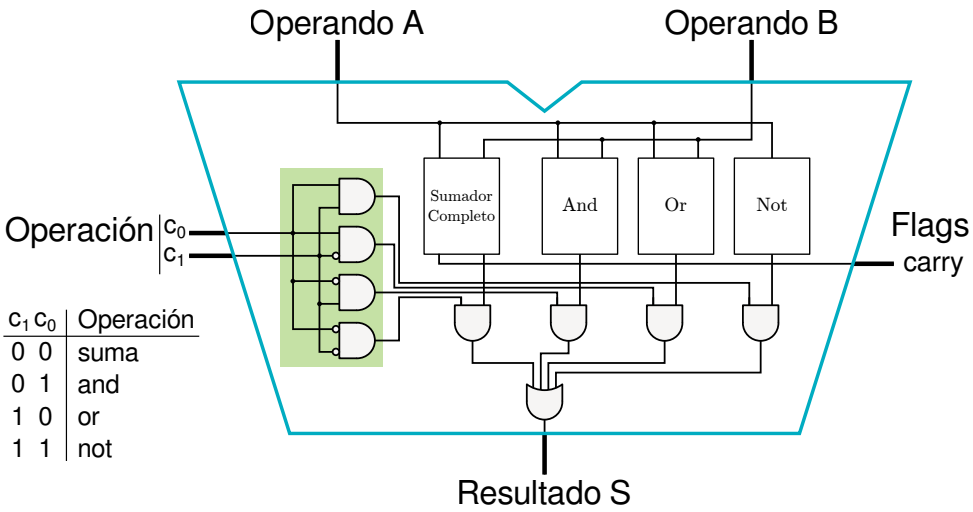
c_1 c_0	Operación
0 0	suma
0 1	and
1 0	or
1 1	not

Circuitos Aritméticos - ALU de 1 bit



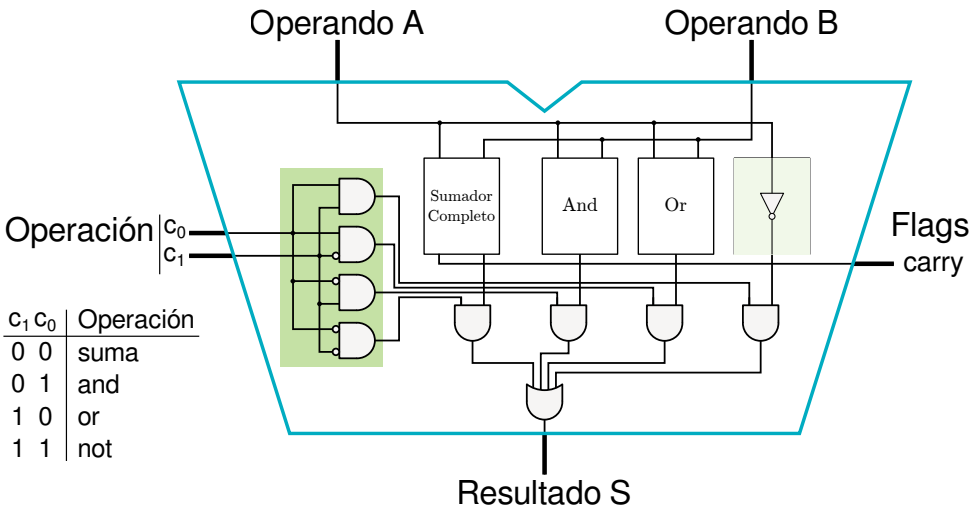
c_1	c_0	Operación
0	0	suma
0	1	and
1	0	or
1	1	not

Circuitos Aritméticos - ALU de 1 bit



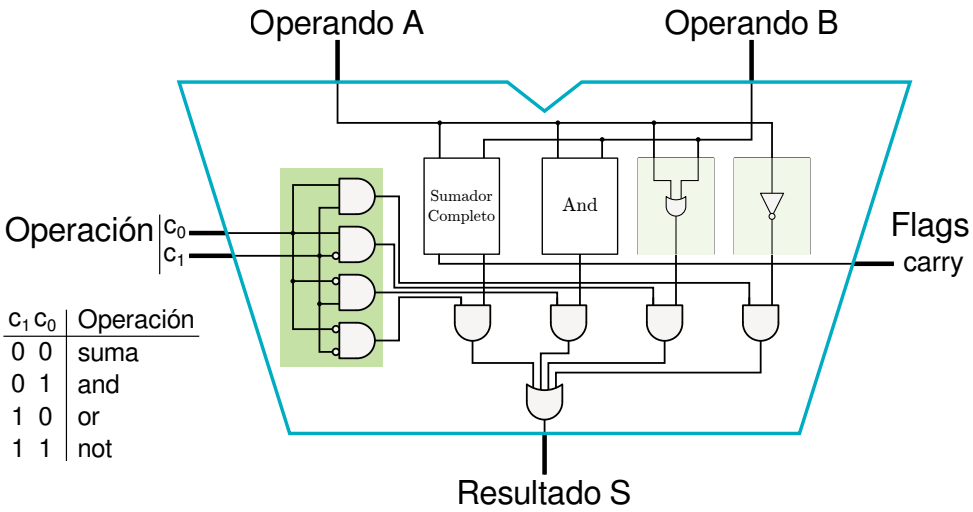
c_1 c_0	Operación
0 0	suma
0 1	and
1 0	or
1 1	not

Circuitos Aritméticos - ALU de 1 bit



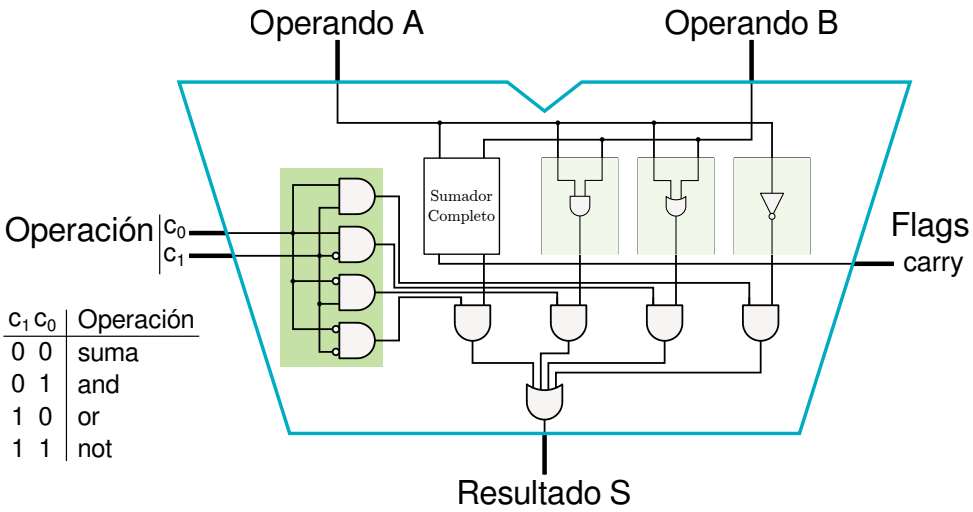
c_1 c_0	Operación
0 0	suma
0 1	and
1 0	or
1 1	not

Circuitos Aritméticos - ALU de 1 bit



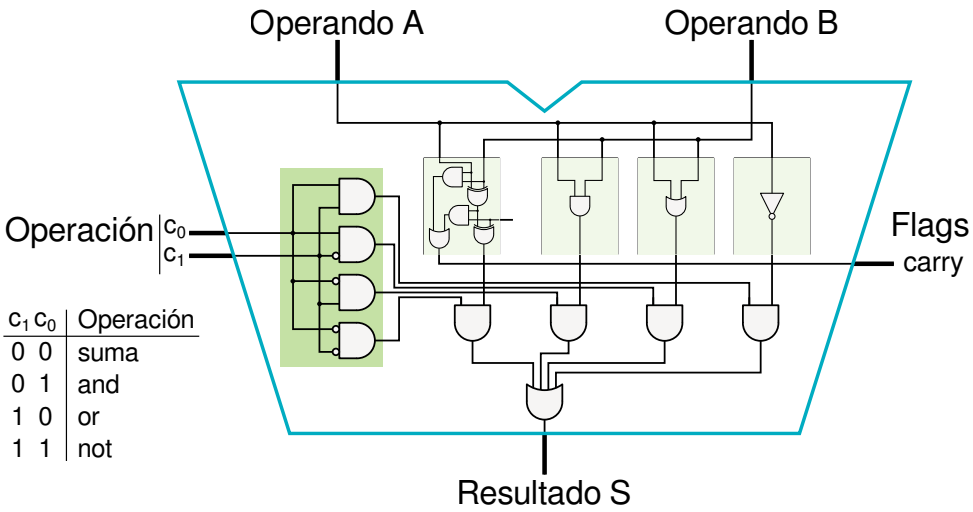
c_1 c_0	Operación
0 0	suma
0 1	and
1 0	or
1 1	not

Circuitos Aritméticos - ALU de 1 bit



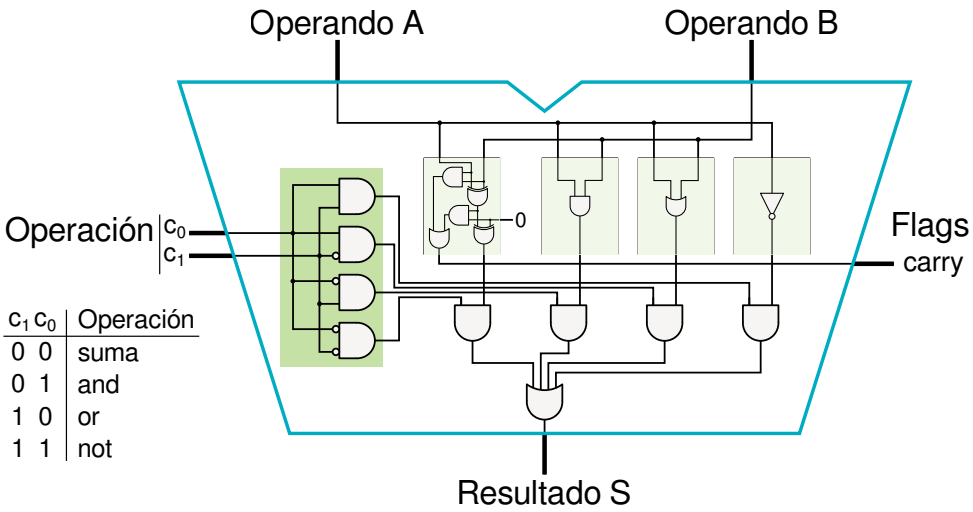
c_1 c_0	Operación
0 0	suma
0 1	and
1 0	or
1 1	not

Circuitos Aritméticos - ALU de 1 bit



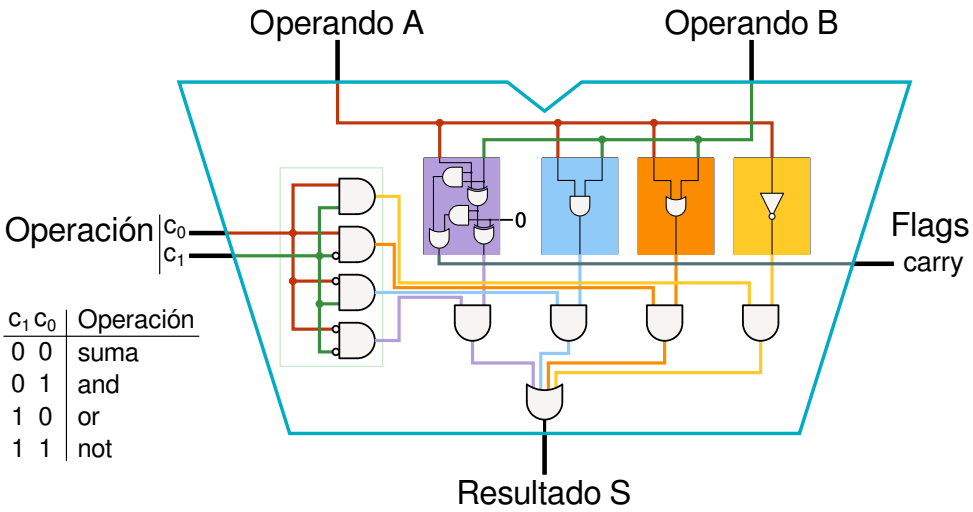
c_1 c_0	Operación
0 0	suma
0 1	and
1 0	or
1 1	not

Circuitos Aritméticos - ALU de 1 bit



c_1 c_0	Operación
0 0	suma
0 1	and
1 0	or
1 1	not

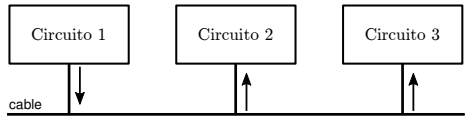
Circuitos Aritméticos - ALU de 1 bit



c_1 c_0	Operación
0 0	suma
0 1	and
1 0	or
1 1	not

Lógica de tres estados

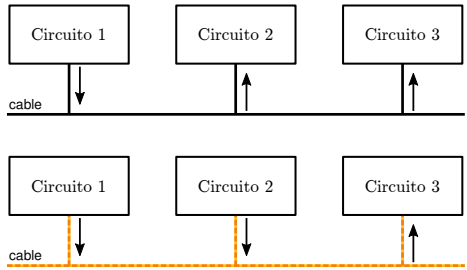
Supongamos tener un **cable** que conecta tres circuitos diferentes. Si un circuito escribe una señal, esta es leída por los otros.



Lógica de tres estados

Supongamos tener un **cable** que conecta tres circuitos diferentes. Si un circuito escribe una señal, esta es leída por los otros.

Ahora, no es posible que dos circuitos escriban **simultáneamente** una señal.



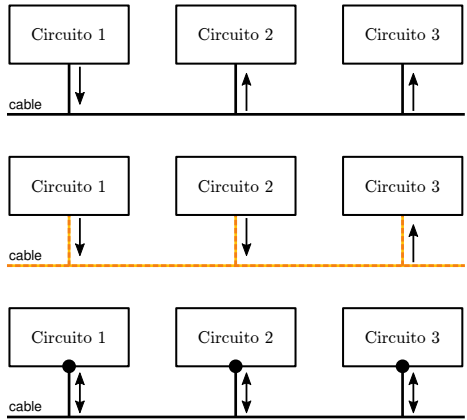
Lógica de tres estados

Supongamos tener un **cable** que conecta tres circuitos diferentes. Si un circuito escribe una señal, esta es leída por los otros.

Ahora, no es posible que dos circuitos escriban **simultáneamente** una señal.

A pesar de que los circuitos puedan acordar no escribir simultáneamente.

Necesitaríamos algún dispositivo que **permita decidir** si leemos o escribimos un cable.



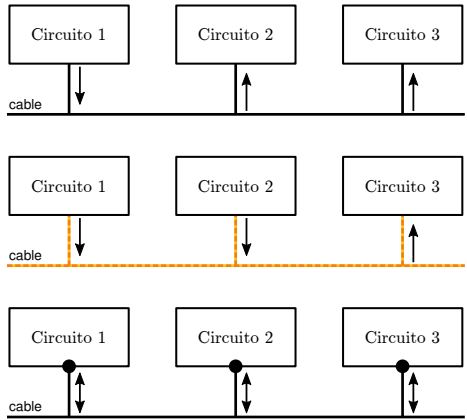
Lógica de tres estados

Supongamos tener un **cable** que conecta tres circuitos diferentes. Si un circuito escribe una señal, esta es leída por los otros.

Ahora, no es posible que dos circuitos escriban **simultáneamente** una señal.

A pesar de que los circuitos puedan acordar no escribir simultáneamente.

Necesitaríamos algún dispositivo que **permita decidir** si leemos o escribimos un cable.

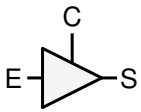


No podemos cambiar el cable para leer o escribir, pero sí podemos **desconectarlo**.

Lógica de tres estados

Un componente de 3 estados es un circuito electrónico que presenta a su salida tres estados posibles:

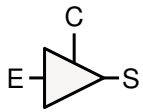
- `0` Estado lógico Cero.
- `1` Estado lógico Uno.
- `hi-Z` No estado. Desconectado (*Alta impedancia*).



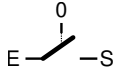
Lógica de tres estados

Un componente de 3 estados es un circuito electrónico que presenta a su salida tres estados posibles:

- `0` Estado lógico Cero.
- `1` Estado lógico Uno.
- `hi-Z` No estado. Desconectado (*Alta impedancia*).



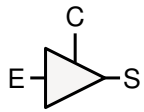
C	E	S
0	0	hi-Z
0	1	hi-Z
1	0	0
1	1	1



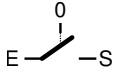
Lógica de tres estados

Un componente de 3 estados es un circuito electrónico que presenta a su salida tres estados posibles:

- `0` Estado lógico Cero.
- `1` Estado lógico Uno.
- `hi-Z` No estado. Desconectado (*Alta impedancia*).



C	E	S
0	0	hi-Z
0	1	hi-Z
1	0	0
1	1	1

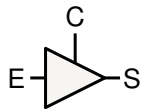


Este componente nos permite **desconectar** circuitos, para así decidir cuándo escribir o leer de un cable.

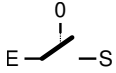
Lógica de tres estados

Un componente de 3 estados es un circuito electrónico que presenta a su salida tres estados posibles:

- `0` Estado lógico Cero.
- `1` Estado lógico Uno.
- `hi-Z` No estado. Desconectado (*Alta impedancia*).



C	E	S
0	0	hi-Z
0	1	hi-Z
1	0	0
1	1	1



Este componente nos permite **desconectar** circuitos, para así decidir cuándo escribir o leer de un cable.

Más adelante:

Vamos a construir registros bidireccionales y utilizar buses, donde usaremos este componente.

Bibliografía

- Tanenbaum, “Organización de Computadoras. Un Enfoque Estructurado”, 4ta Edición, 2000.
  - **Capítulo 3 - El nivel de lógica digital** - Páginas 128-139

- Null, “Essentials of Computer Organization and Architecture”, 5th Edition, 2018.
  - **Chapter 3 - Boolean Algebra and Digital Logic:**
    - 3.6 - Combinational Circuits

Ejercicios

Con lo visto, ya pueden resolver hasta el ejercicio 9 de la Guía de Lógica Digital.

¡Gracias!

Recuerden leer los comentarios adjuntos
en cada clase por aclaraciones.