

**Universidad Nacional Autónoma de
México**

Facultad de Ingeniería

Diseño Digital Moderno

Proyecto 2: ALU

Profesora: Elizabeth Fonseca Chavez

Alumno: Quintanar Ramírez Luis Enrique

Objetivos

- **Analizar, diseñar, simular e implementar arquitecturas aritméticas y lógicas.**
- **Implementar multiplexores en una tarjeta lógico-programable**

Introducción

- **¿Qué es una ALU?**

Unidad Lógico-Aritmética (ALU, por sus siglas en inglés) es un circuito digital que calcula operaciones aritméticas (suma, resta, multiplicación, etc) y operaciones lógicas (and, or, not, etc)

Introducción

- **¿Qué es un Multiplexor?**

Son circuitos combinacionales con dos tipos de entradas (datos y control) y una salida. Las entradas de control seleccionan una, y solo una, entrada de datos para permitir la salida

Introducción

- **Suma Binaria**

Para realizar una suma binaria, consideramos

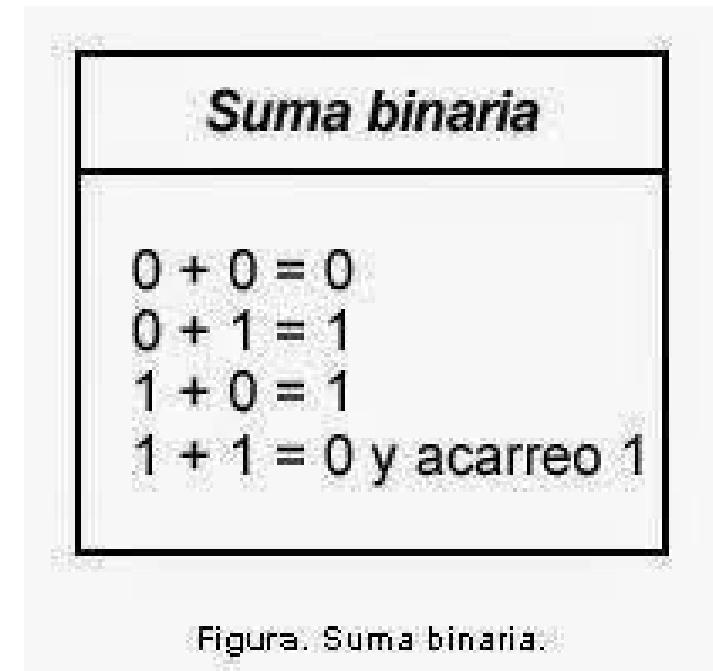


Figura. Suma binaria.

Introducción

- **Resta Binaria (N-1)**

Para realizar la resta en esta tarjeta, realizaremos una suma, como sólo queremos obtener N-1 (el número menos uno), al número dado, le sumaremos solo unos, es decir:

11 → Acarreos

1110 → 14

+ 1111

= 1| 1101 → 13, con acarreo

Introducción

- **Resta Binaria (A-B)**
- Para realizar una resta binaria, consideramos

Resta binaria

$$0 - 0 = 0$$

$$0 - 1 = 1 \text{ y acarreo } 1$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

© carlospes.com

Introducción

- Operación AND y XOR



a	b	out
0	0	0
0	1	0
1	0	0
1	1	1



a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

Desarrollo

Para realizar el diseño, será necesario dividir el problema en dos partes, una Unidad Lógica (UL) y otra Unidad Aritmética (UA), la unidad Lógica se encargará de realizar la operación AND y XOR, y la Unidad Aritmética realizará la operación Suma y Resta

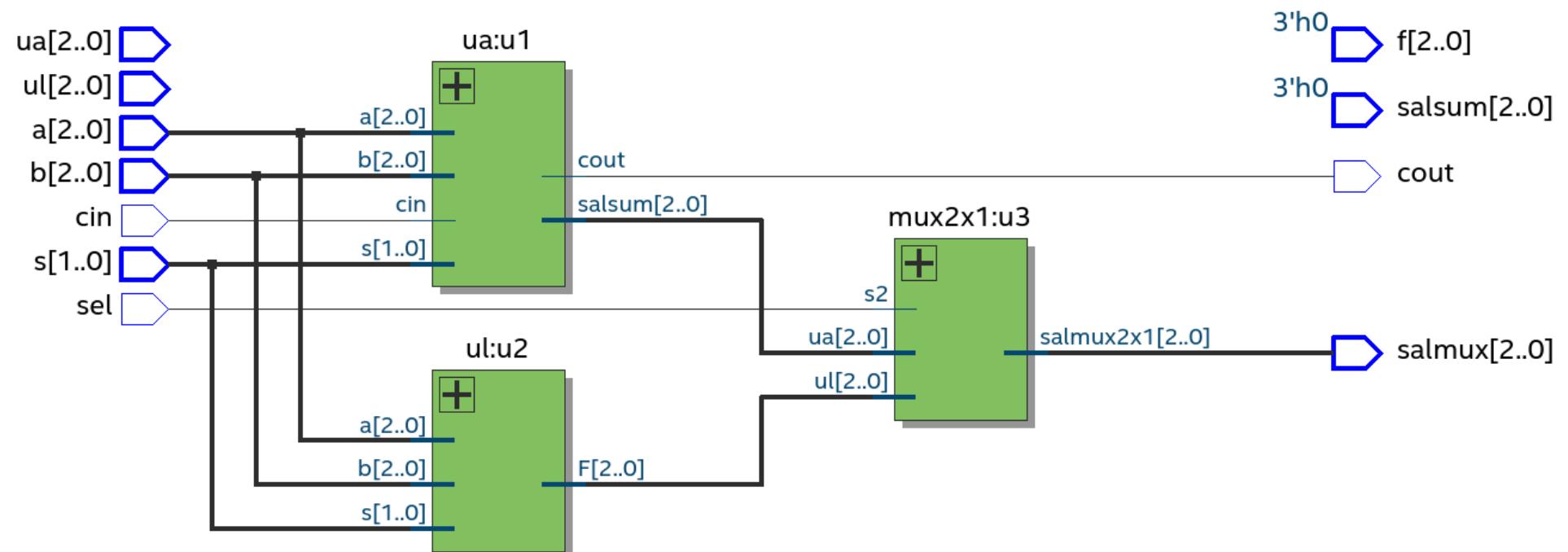
Desarrollo

Necesitaremos, además, dos multiplexores, uno de 2 a 1 y otro de 4 a 1:

El mux2x1 se encargará de guiar la decisión de realizar una operación lógica o una operación aritmética

Mientras que el mux4x1 se encargará del control del segundo número, esto dentro de la unidad aritmética

Desarrollo



Desarrollo

Como observamos en la imagen anterior, vemos que el RTL consta de dos partes, una lógica (UL) y una aritmética (UA).

La UA consta de un multiplexor que nos permitirá realizar la suma y la resta, además de otras operaciones que serán irrelevantes

La UL, a partir de la entrada de dos datos, nos dará una salida según un selector que realizará las operaciones AND y XOR entre otras operaciones

Desarrollo

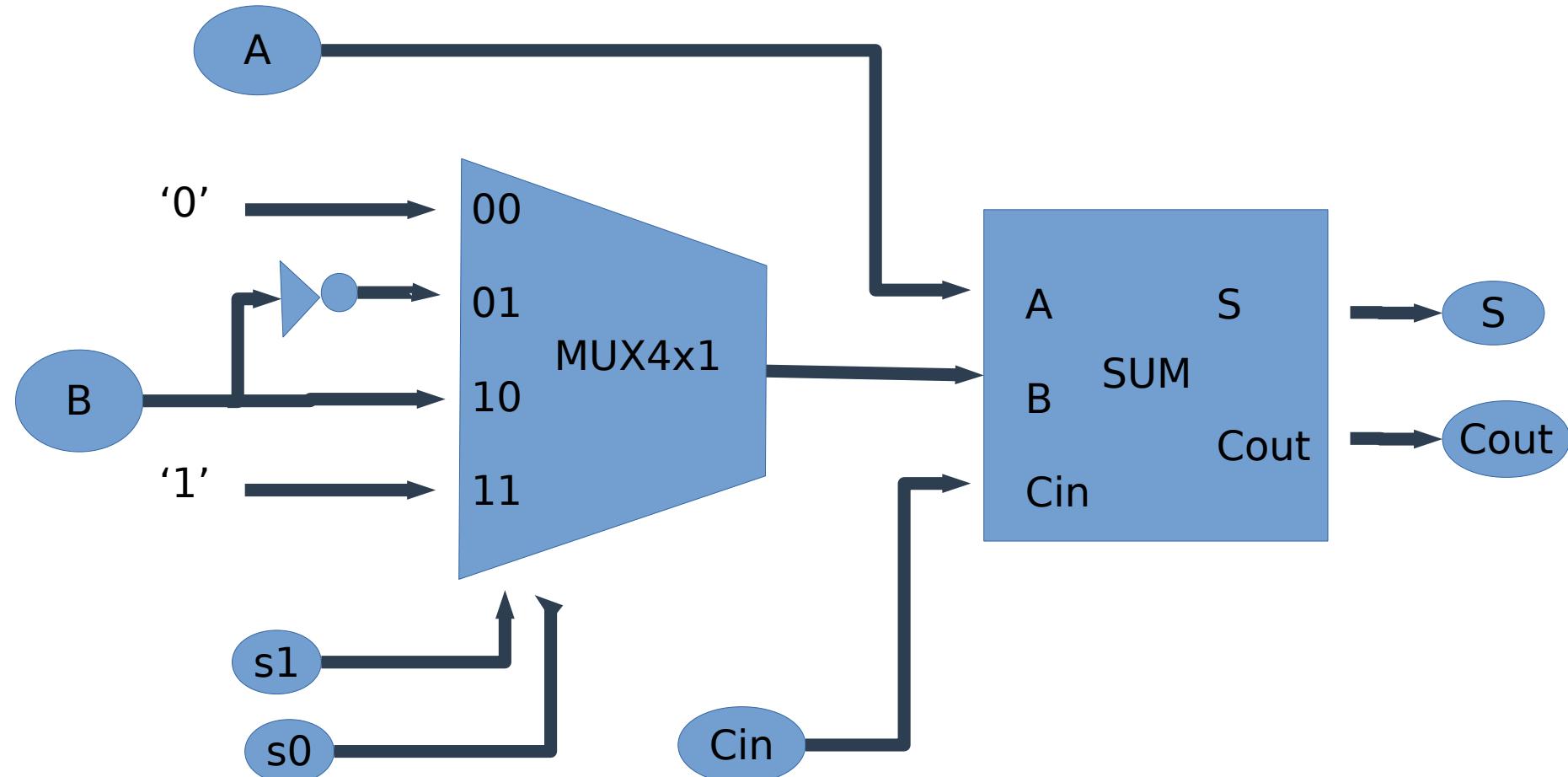
Unidad Lógica

```
entity ul is
  port(
    a,b: in std_logic_vector(3 downto 0);
    s: in std_logic_vector(1 downto 0);
    F: out std_logic_vector(3 downto 0)
  );
end entity ul;
architecture arqUL of ul is
  signal sand, sor, sxor,snot: std_logic_vector(3 downto 0);
begin
  sand<= a and b;
  sor <= a or b;
  sxor<= a xor b;
  snot<= not a;
  with s select
    F<=
      sand when "00",
      sor when "01",
      sxor when "10",
      snot when "11",
      (others=>'0') when others;
end architecture arqUL;
```

S	Salida
00	A and B
01	A or B
10	A xor B
11	Not A

Desarrollo

Unidad Aritmética



Desarrollo

Unidad Aritmética

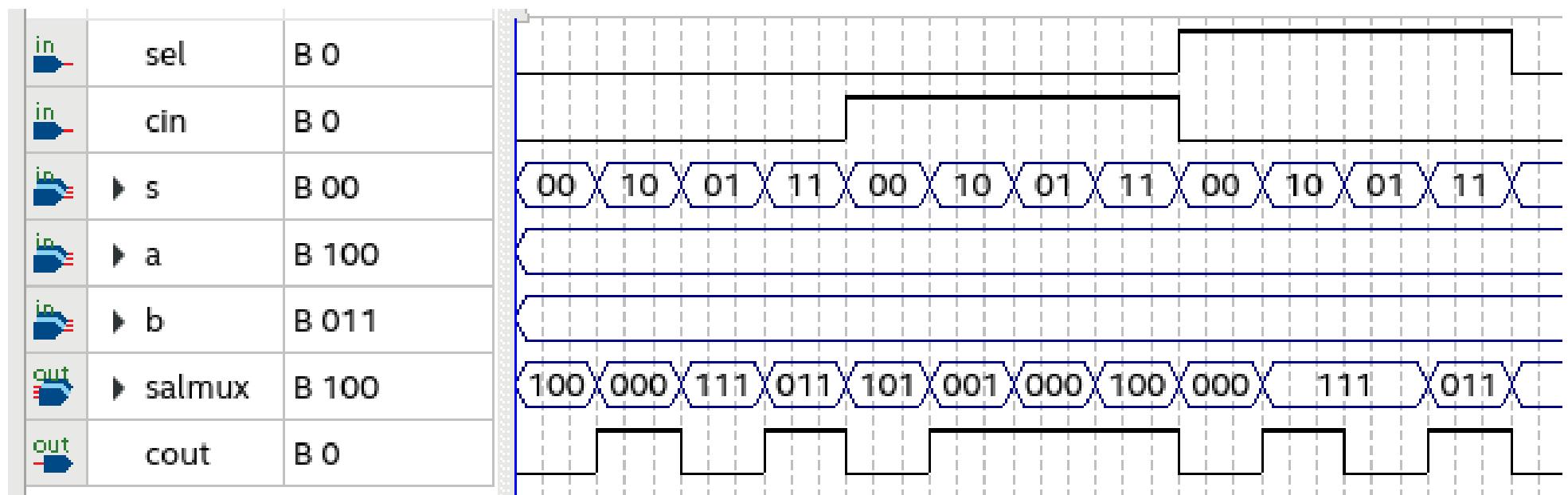
```
entity sum is
port(a, b: in std_logic_vector(3 downto 0);
      cin: in std_logic;
      salsum: out std_logic_vector(3 downto 0);
      cout: out std_logic);
end entity sum;

architecture arqsum of sum is
signal mid: std_logic_vector(4 downto 0);--un bit que a y b
begin
mid<= ('0' & a) + ('0' & b) + cin;
cout <= mid(4);
salsum <= mid(3 downto 0);

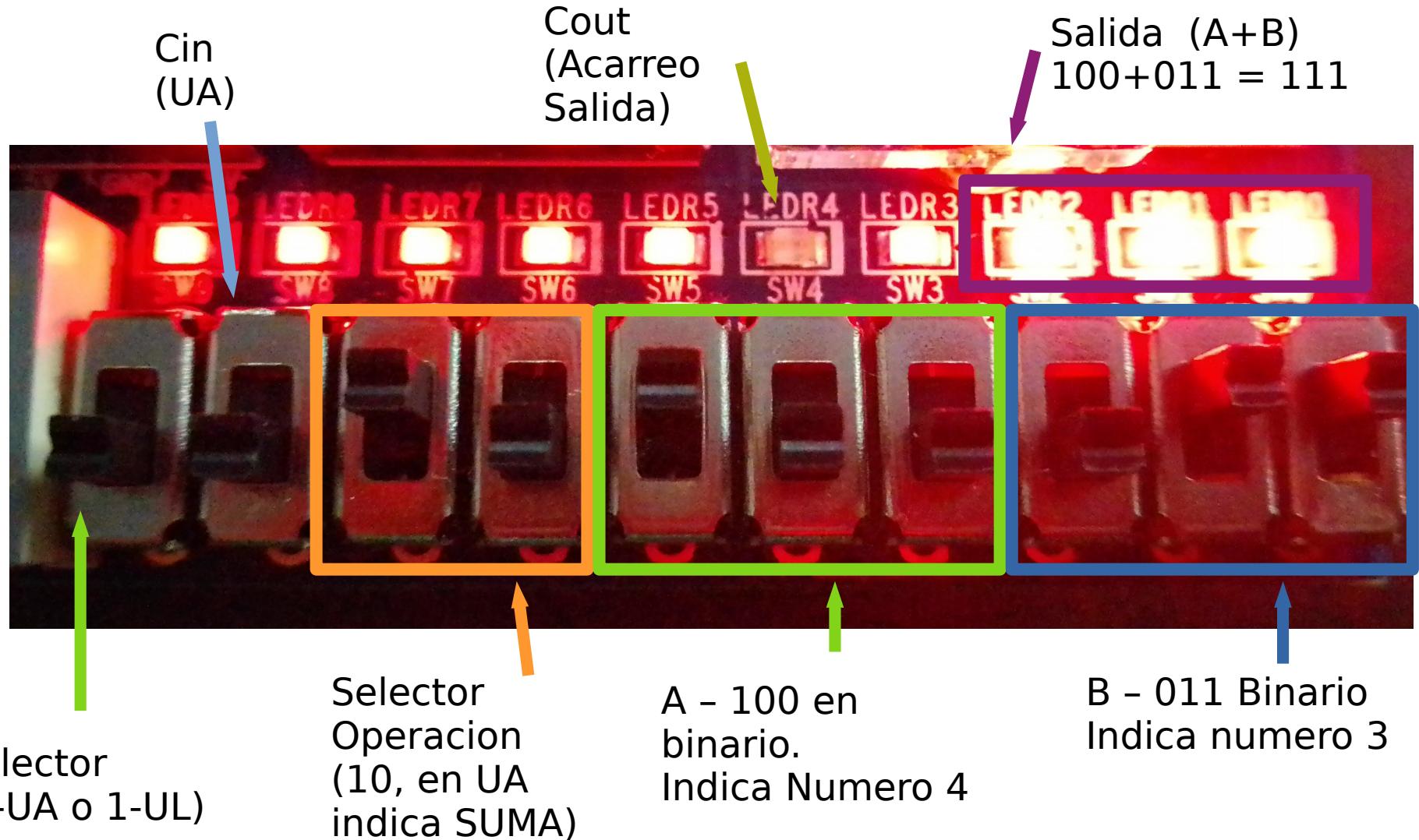
entity mux4x1 is
port(
  s: in std_logic_vector(1 downto 0); --s(1) y s(0)
  b: in std_logic_vector(3 downto 0); --000 001 010 011  100 101 110 111
  salmux4x1: out std_logic_vector(3 downto 0)
);
end entity mux4x1;
architecture arqmux4x1 of mux4x1 is
begin
with s select
  salmux4x1 <=
    (others=>'0') when "00",
    b when "01",
    not b when "10",
    (others=>'1') when "11",
    (others=>'0') when others;
end architecture arqmux4x1;
```

Cin	S1	S0	Salida
0	0	0	A
0	0	1	A + B'
0	1	0	A + B
0	1	1	A - 1
1	0	0	A + 1
1	0	1	A - B
1	1	0	A + B +1
1	1	1	A

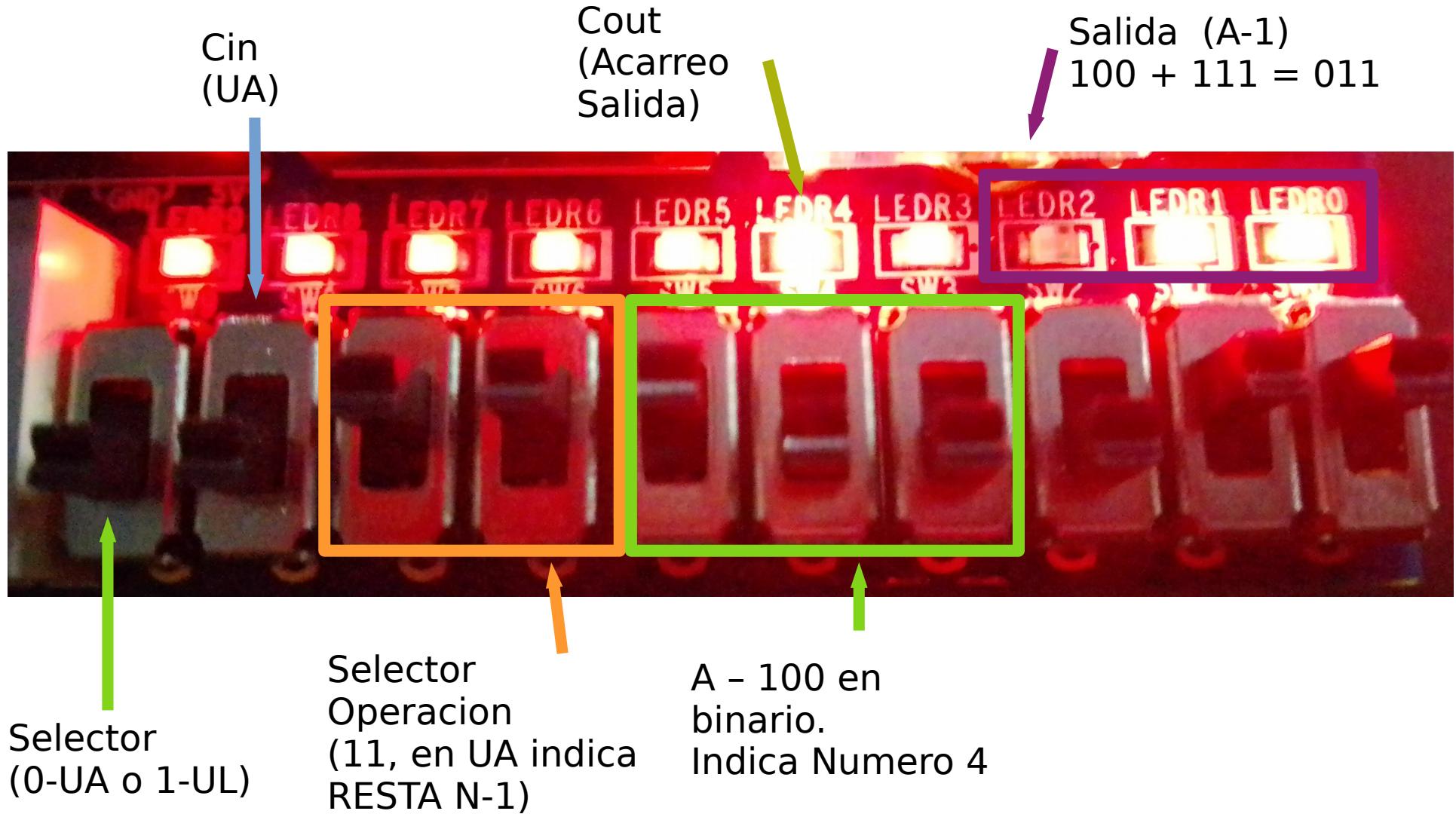
Simulación



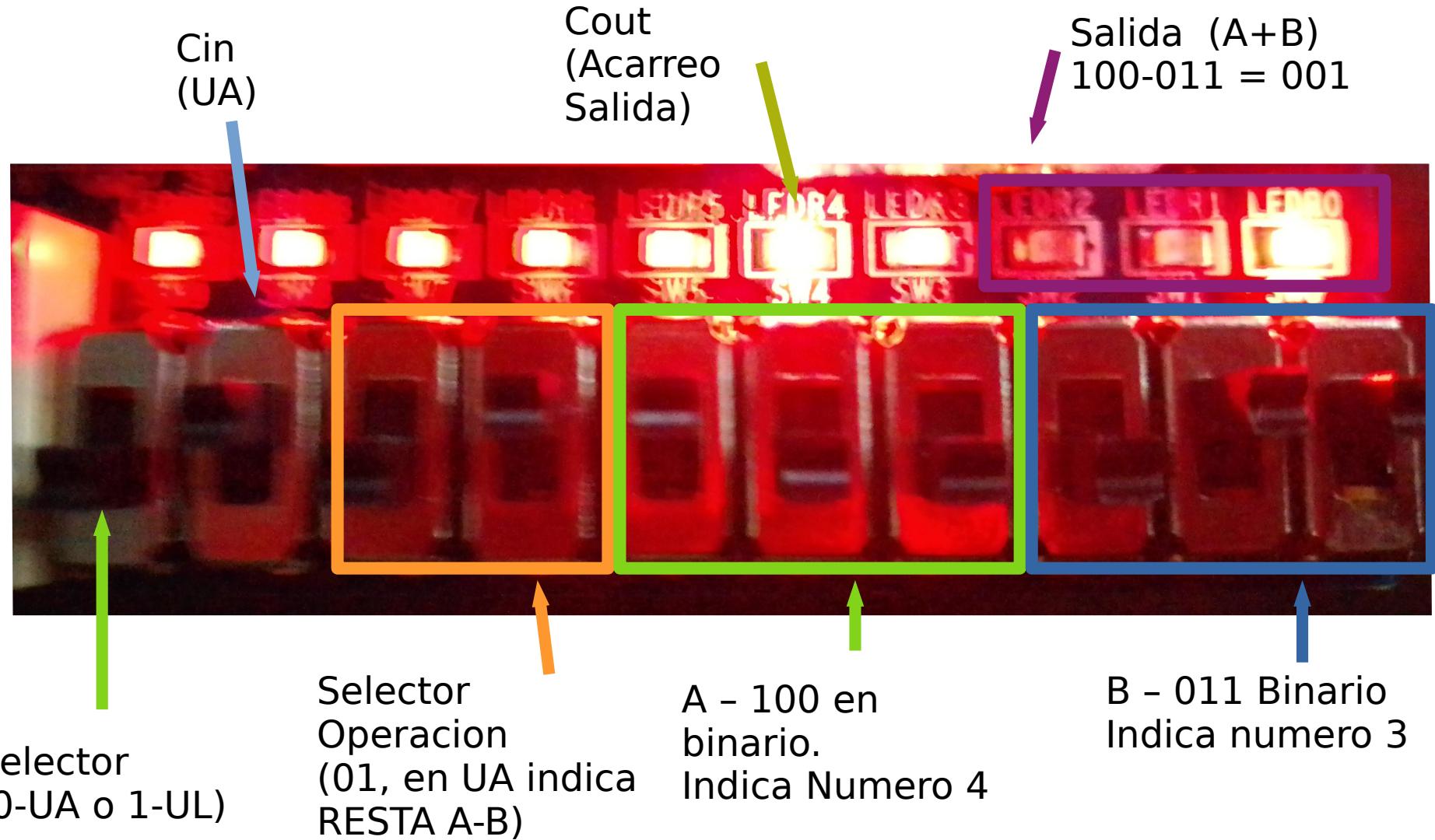
Pruebas en Tarjeta (UA - SUMA A+B)



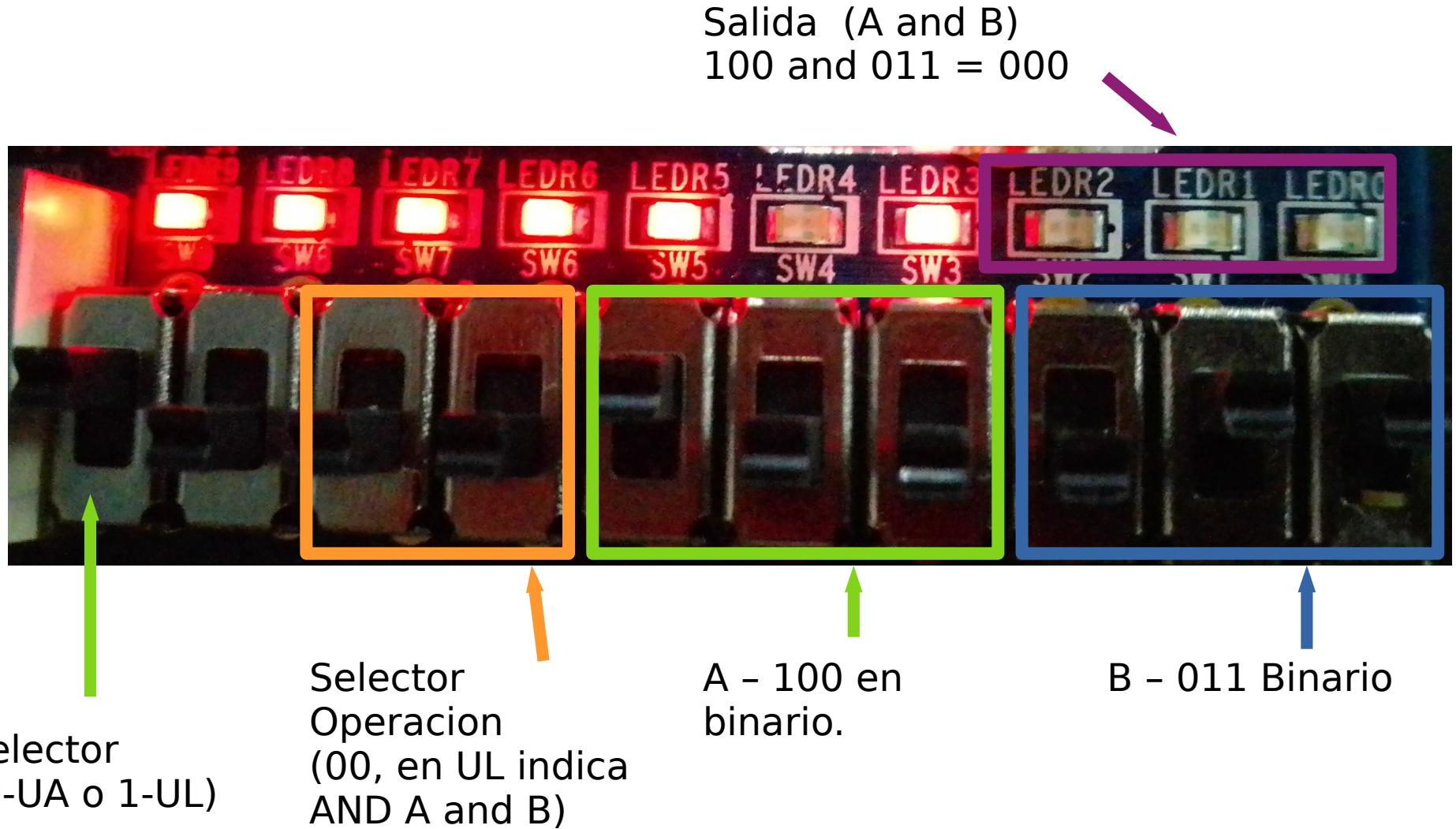
Pruebas en Tarjeta (UA - RESTA A-1)



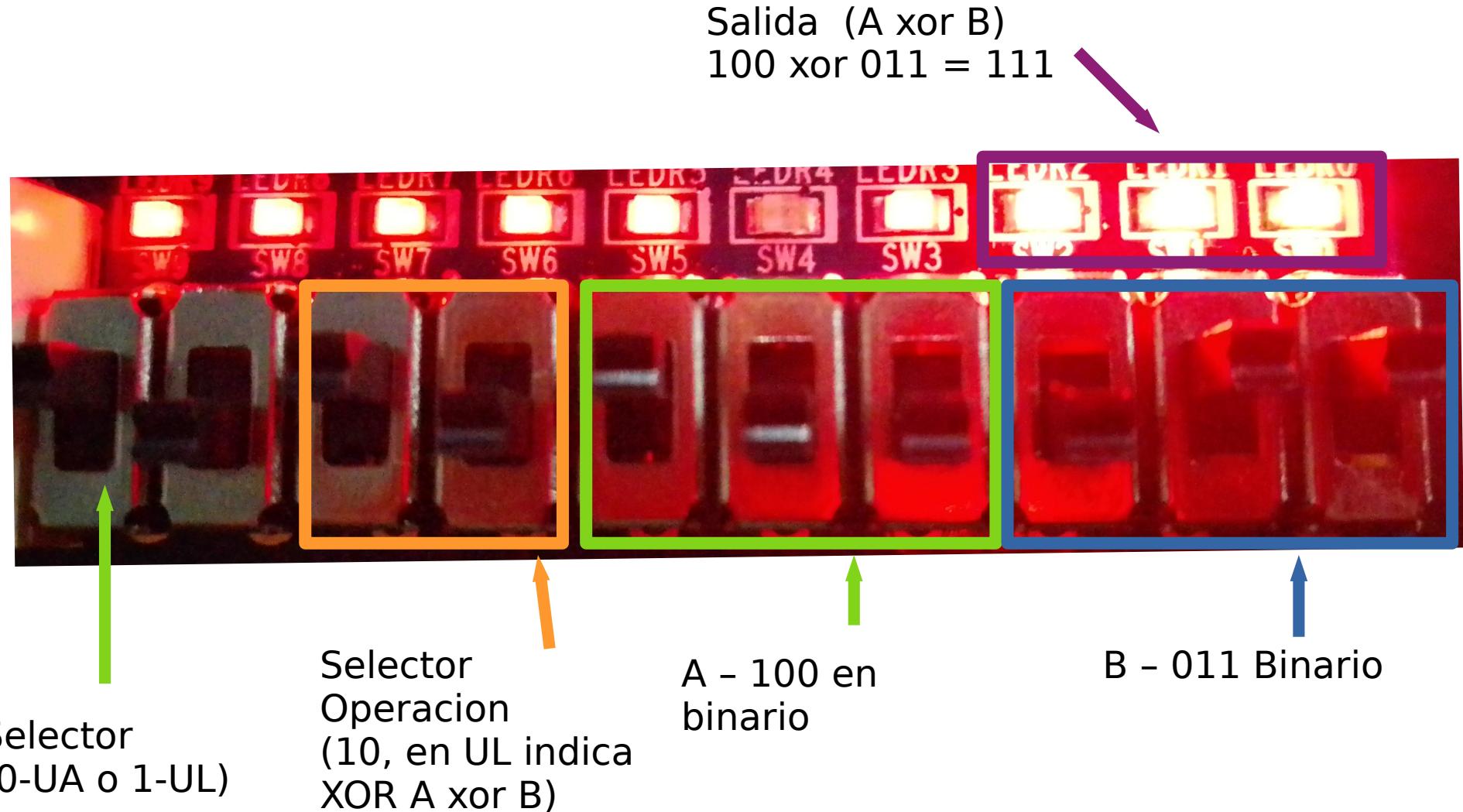
Pruebas en Tarjeta (UA - SUMA A-B)



Pruebas en Tarjeta (UL - A AND B)



Pruebas en Tarjeta (UL - A XOR B)



Conclusiones

Pudimos observar el comportamiento de una ALU. Implementamos, además, multiplexores para hacer varias tareas cambiando un selector y también pudimos diseñar todo el problema como si fuera una caja negra. Aunque nosotros sabemos cómo funciona, algún otro usuario no necesitaría conocer el funcionamiento, sólo ocuparía los resultados dados

Referencias

- Chávez, Elba (s.f) Multiplexores. Recuperada el 19 de octubre de 2021 de http://profesores.fi-b.unam.mx/normaelva/multiplexores_binarios.pdf
- Fonseca E.[Profesora Elizabeth Fonseca](2020) ULA. [Archivo de video] Youtube.
<https://www.youtube.com/watch?v=8vszGhGa1RI&t=6s>
-