

**Universidad Nacional Autónoma de
México**

Facultad de Ingeniería

Diseño Digital VLSI

Proyecto 1: 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**
- **Automatizar el funcionamiento de una ALU**

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 combinatoriales 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

<i>Suma binaria</i>	
$0 + 0 = 0$	
$0 + 1 = 1$	
$1 + 0 = 1$	
$1 + 1 = 0$	y acarreo 1

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:

$$\begin{array}{rcl} & 11 & \rightarrow \text{Acarreos} \\ & 1110 & \rightarrow 14 \\ + & 1111 & \\ = & 1|1101 & \rightarrow 13, \text{ con acarreo} \end{array}$$

Introducción

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

<i>Resta binaria</i>
$0 - 0 = 0$
$0 - 1 = 1$ y acarreo 1
$1 - 0 = 1$
$1 - 1 = 0$
© carlospes.com

Introduccion

- 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

Introduccion

- Operación NOT y OR

	<table><tr><th>A</th><th>B</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	B	0	1	1	0
A	B						
0	1						
1	0						
NOT							

	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	1	0	1	0	1	1	1	1	1
A	B	Y														
0	0	0														
1	0	1														
0	1	1														
1	1	1														
OR																

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:

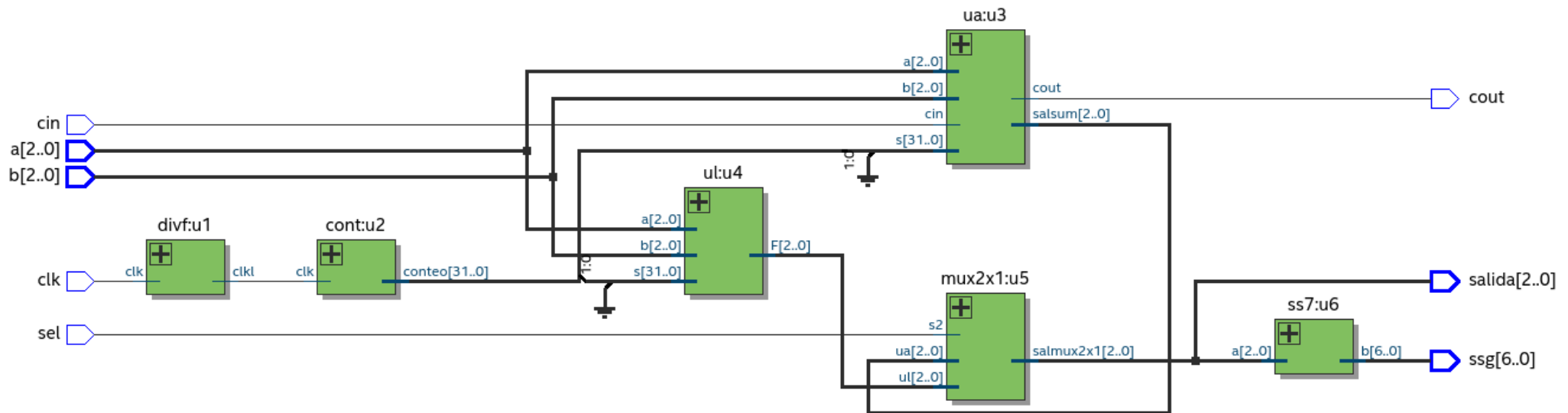
El mux2x1 se encargará de guiar la decisión de realizar una operación lógica o una operación aritmética

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

Desarrollo

También utilizaremos un divisor de frecuencia para controlar los tiempos de cambio de la ALU automatizada y de un contador, que irá registrando los cambios que ocurren para así cambiar entre operaciones de la ALU

Desarrollo



Desarrollo

Como observamos en la imagen anterior, vemos que el RTL consta de seis partes. Primero tenemos al divisor de frecuencias (divf) que controla los tiempos de cambios entre operaciones. Después un contador (cont), que cada que recibe un cambio del divisor, aumenta su tamaño y así cambia la operación.

Desarrollo

Contamos también con las partes de la Unidad Logica (UL) que se encarga de las operaciones and, or, xor not. Y la Unidad Aritmética que realizará operaciones como suma, resta, decremento e incremento.

Se ve además un multiplexor de 2x1, con el cuál podemos elegir entre cada una de las unidades

Desarrollo

Por último tenemos el encargado de transmitir la información al display de siete segmentos (ss7), del cual con el multiplexor asignamos la salida y según el resultado lo muestra en el display

Desarrollo

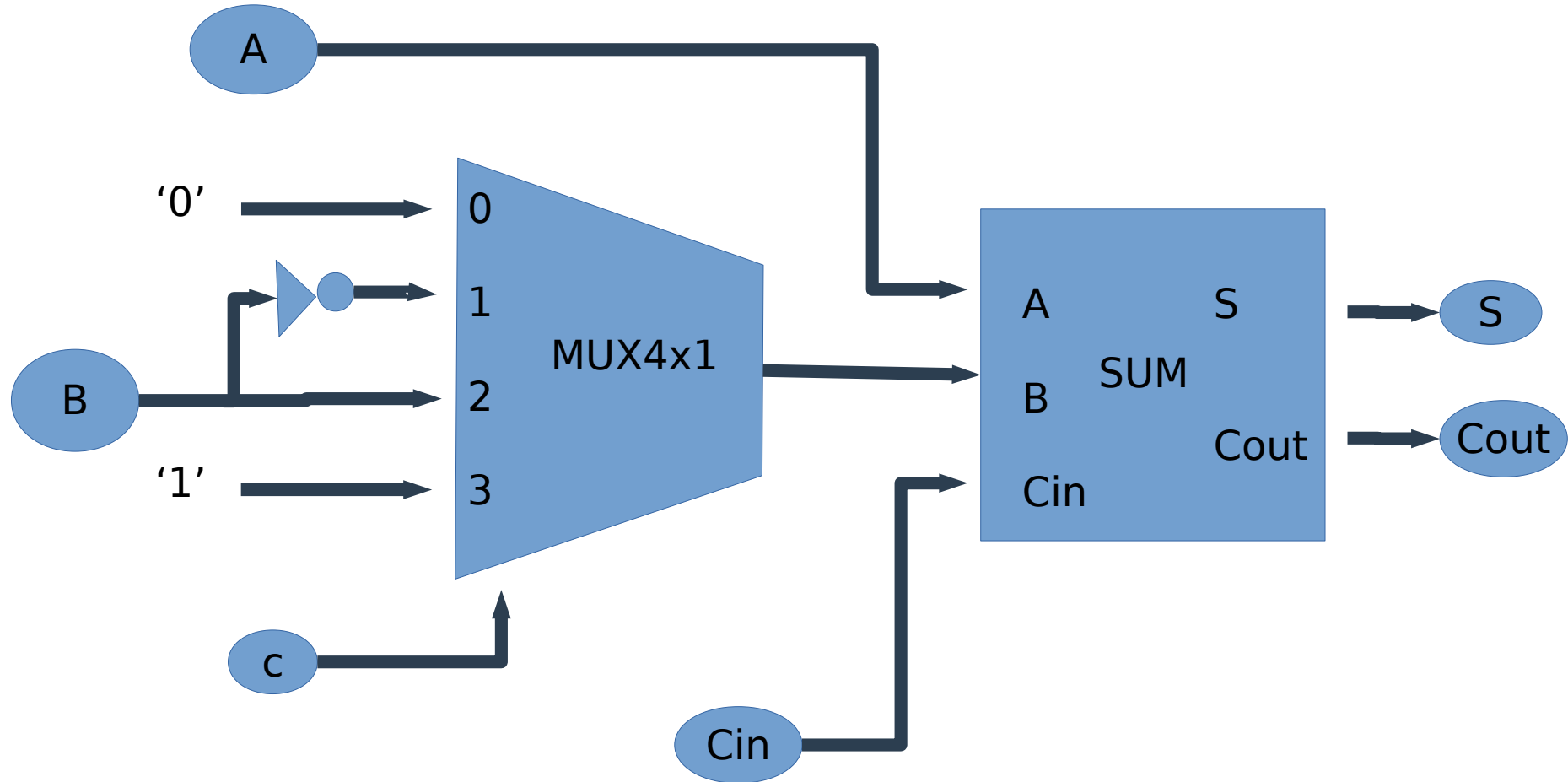
Unidad Lógica

```
ul.vhd    ua.vhd    ALU.vhd
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity ul is
5  port(
6    a,b: in std_logic_vector(2 downto 0);
7    s: in integer;
8    F: out std_logic_vector(2 downto 0)
9  );
10 end entity;
11
12 architecture arqUL of ul is
13   signal sand, sor, sxor, snot: std_logic_vector(2 downto 0);
14   begin
15     sand<= a and b;
16     sor <= a or b;
17     sxor<= a xor b;
18     snot<= not a;
19
20     with s select
21       F<=  sand when 0,
22            sor  when 1,
23            sxor when 2,
24            snot when 3,
25            (others=>'0') when others;
26   end architecture ;
```

S	Salida
0	A and B
1	A or B
2	A xor B
3	Not A

Desarrollo

Unidad Aritmética



Desarrollo

Unidad Aritmética

```
4
5 entity sum is
6 port(a, b: in std_logic_vector(2 downto 0);
7       cin: in std_logic;
8       salsum: out std_logic_vector(2 downto 0);
9       cout: out std_logic);
10 end entity sum;
11
12 architecture arqsum of sum is
13 signal mid: std_logic_vector(3 downto 0);
14 begin
15
16 mid<=('0' & a) + ('0' & b) + cin;
17 cout <=mid(3);
18 salsum <= mid(2 downto 0);
19 end architecture arqsum;
```

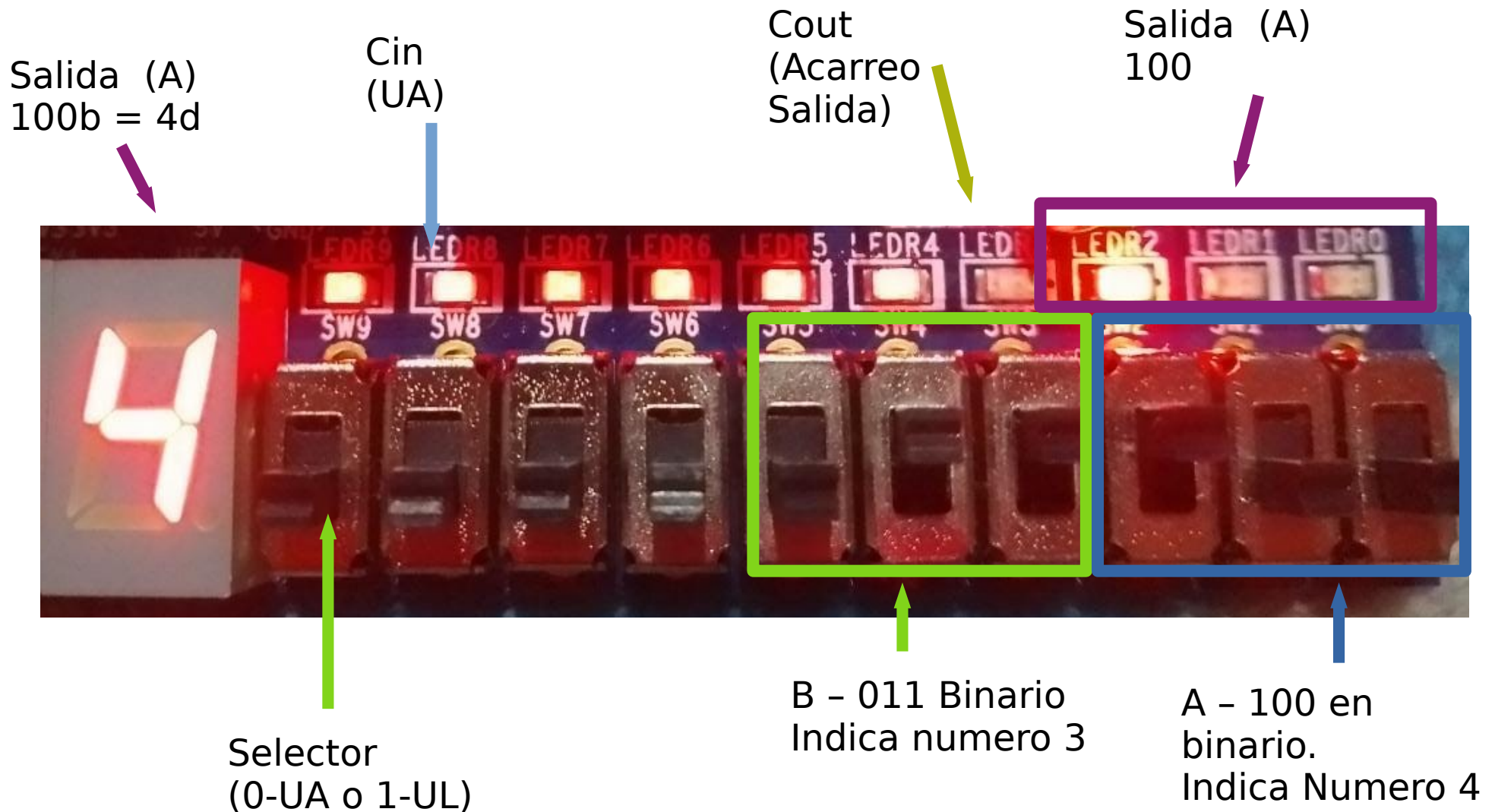


```
4 library ieee;
5 use ieee.std_logic_1164.all;
6 --unidad=>s(2),b(3),salmux4x1(3)
7 --del mux4 port map(s,b, salmux4x1)
8 entity mux4x1 is
9 port(
10 s: in integer;
11 b: in std_logic_vector(2 downto 0);
12 salmux4x1: out std_logic_vector(2 downto 0)
13 );
14 end entity mux4x1;
15 architecture arqmux4x1 of mux4x1 is
16 begin
17 with s select
18 salmux4x1 <=
19     (others=>'0') when 0,
20     b when 1,
21     not b when 2,
22     (others=>'1') when 3,
23     (others=>'0') when others;
24
25 end architecture arqmux4x1;
```

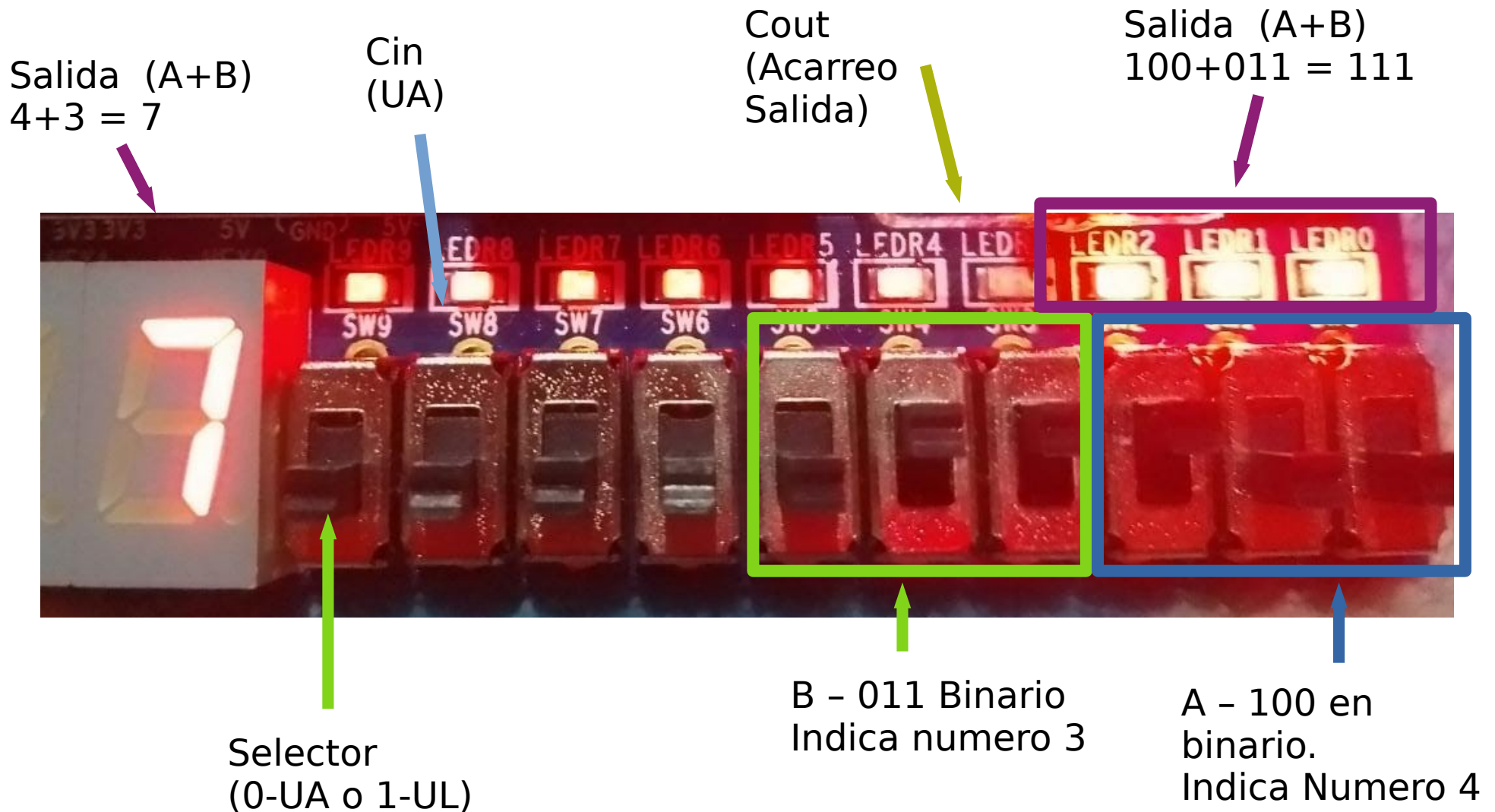
Cin	S	Salida
0	0	A
0	1	A + B'
0	2	A + B
0	3	A - 1
1	0	A + 1
1	1	A + B + 1
1	2	A - B
1	3	A

Simulación

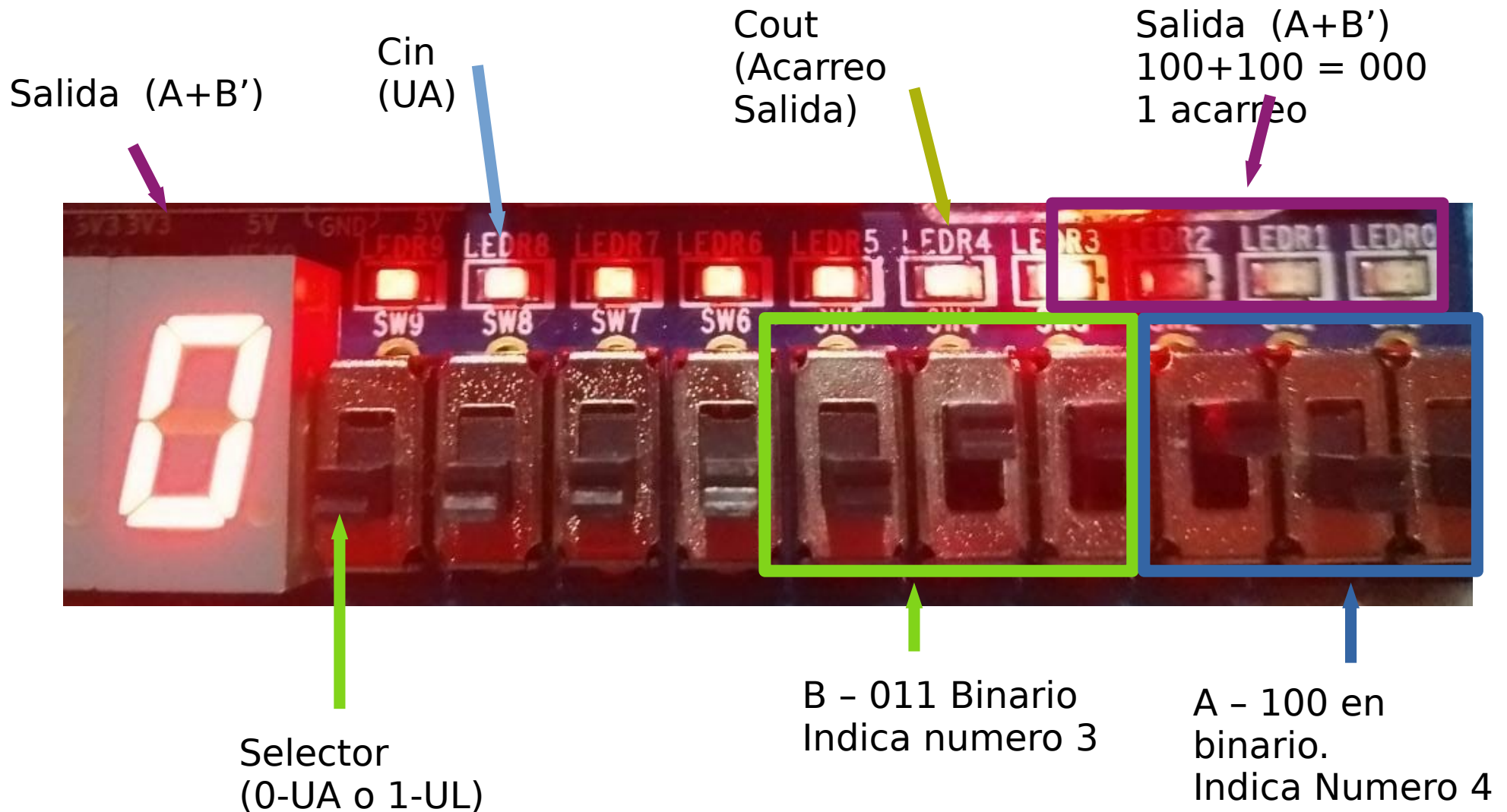
Pruebas en Tarjeta (UA - A sin acarreo)



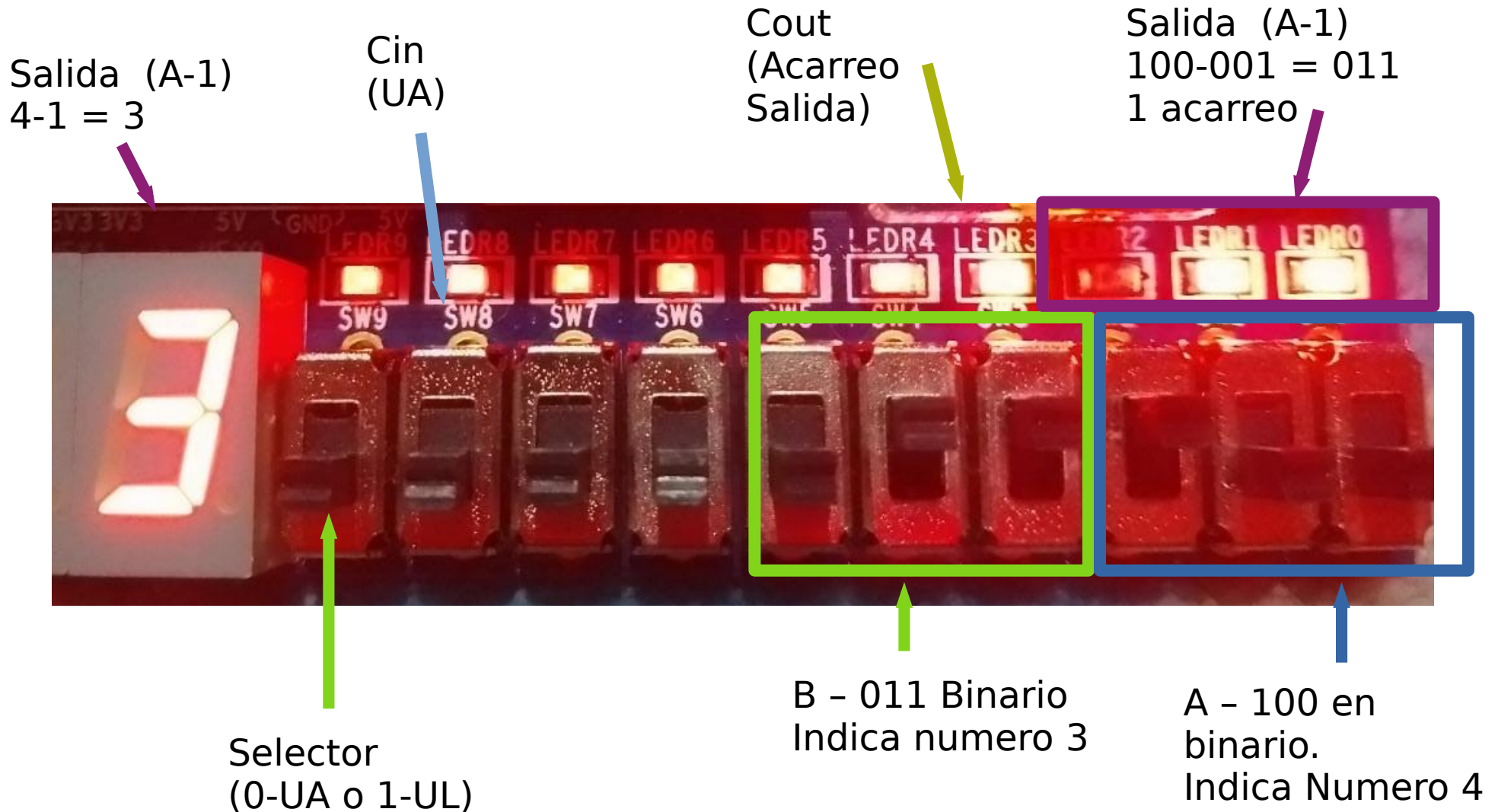
Pruebas en Tarjeta (UA - SUMA A+B)



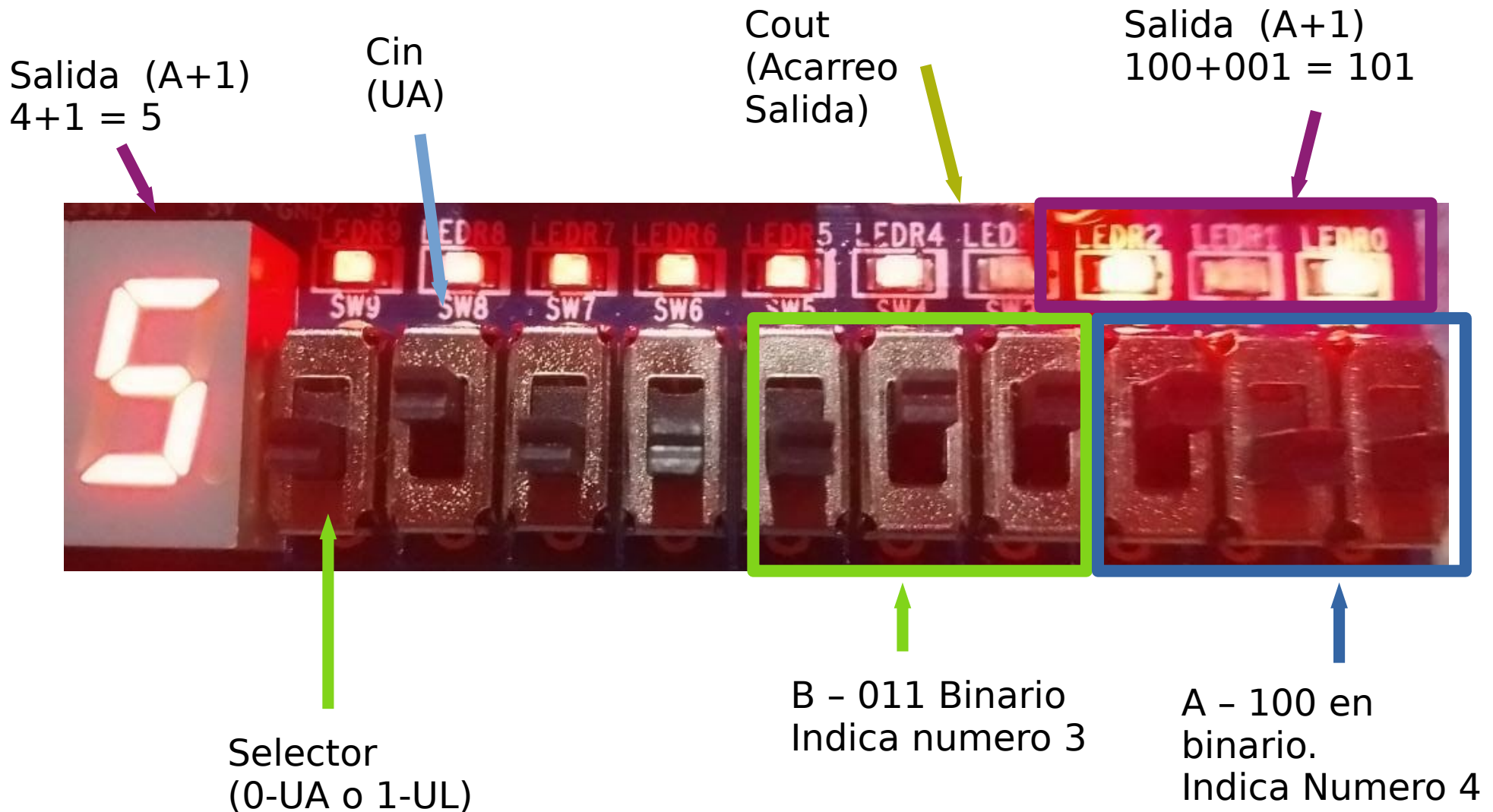
Pruebas en Tarjeta (UA - SUMA $A+B'$)



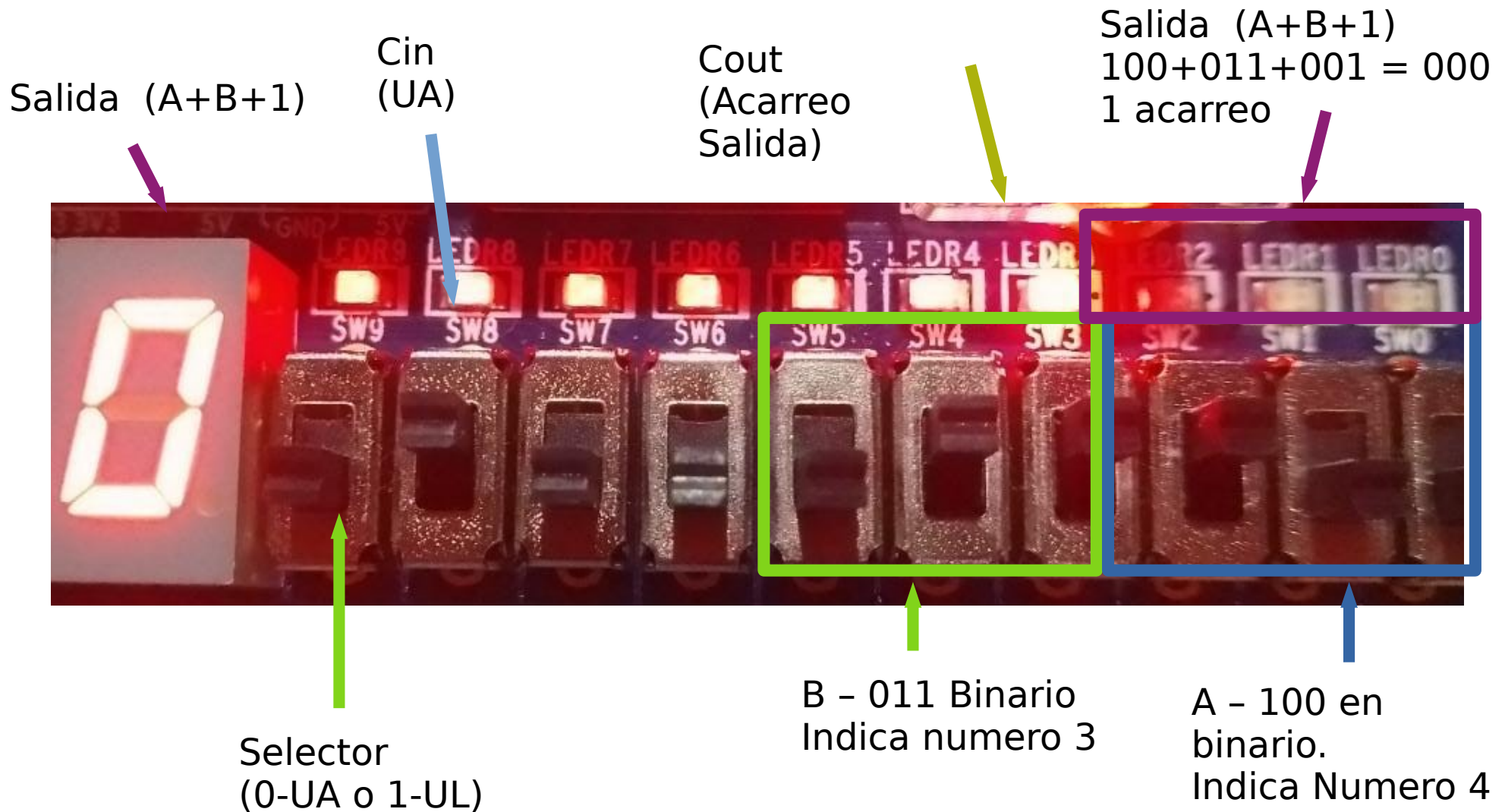
Pruebas en Tarjeta (UA - SUMA A-1)



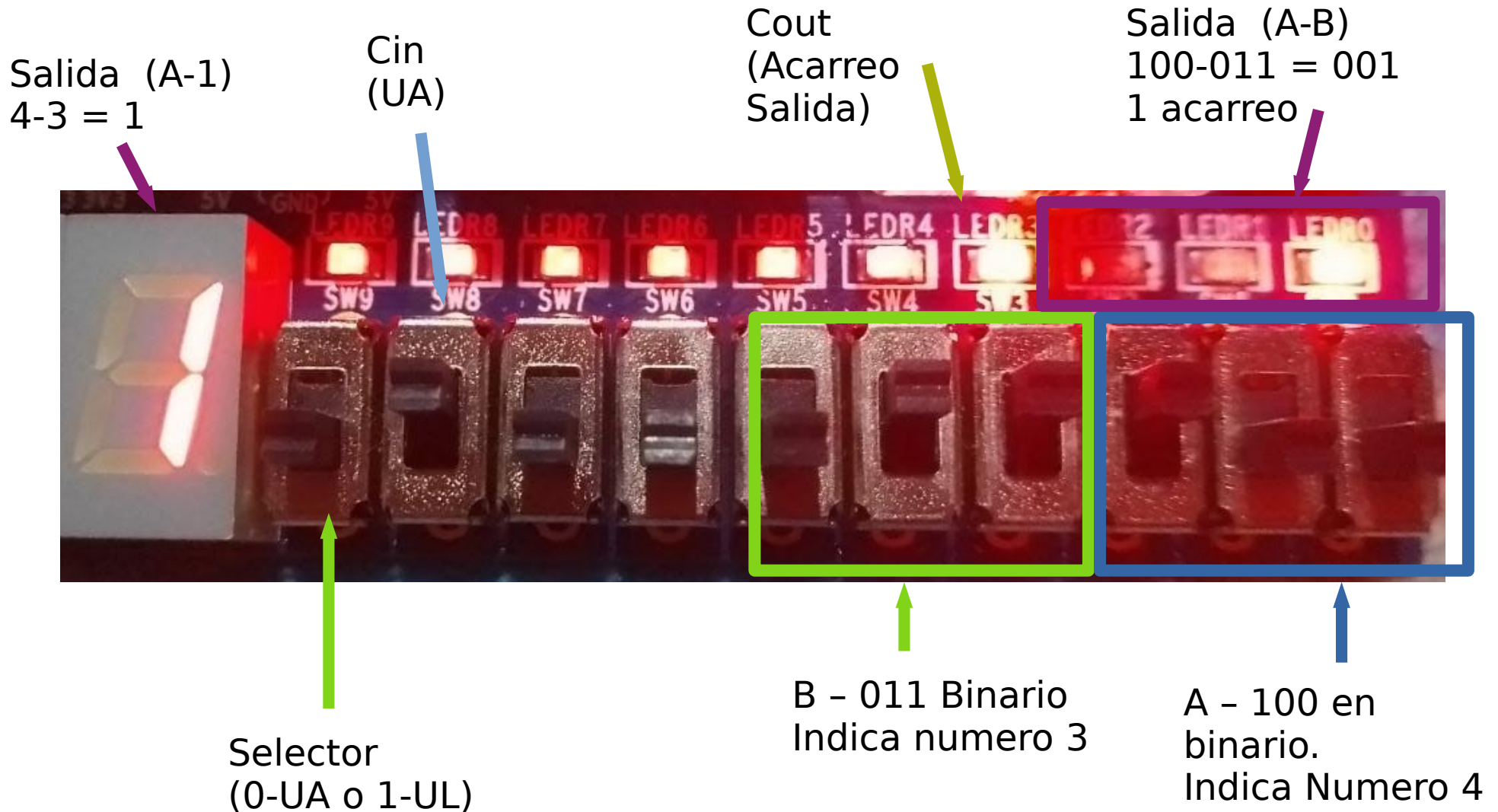
Pruebas en Tarjeta (UA - SUMA A+1)



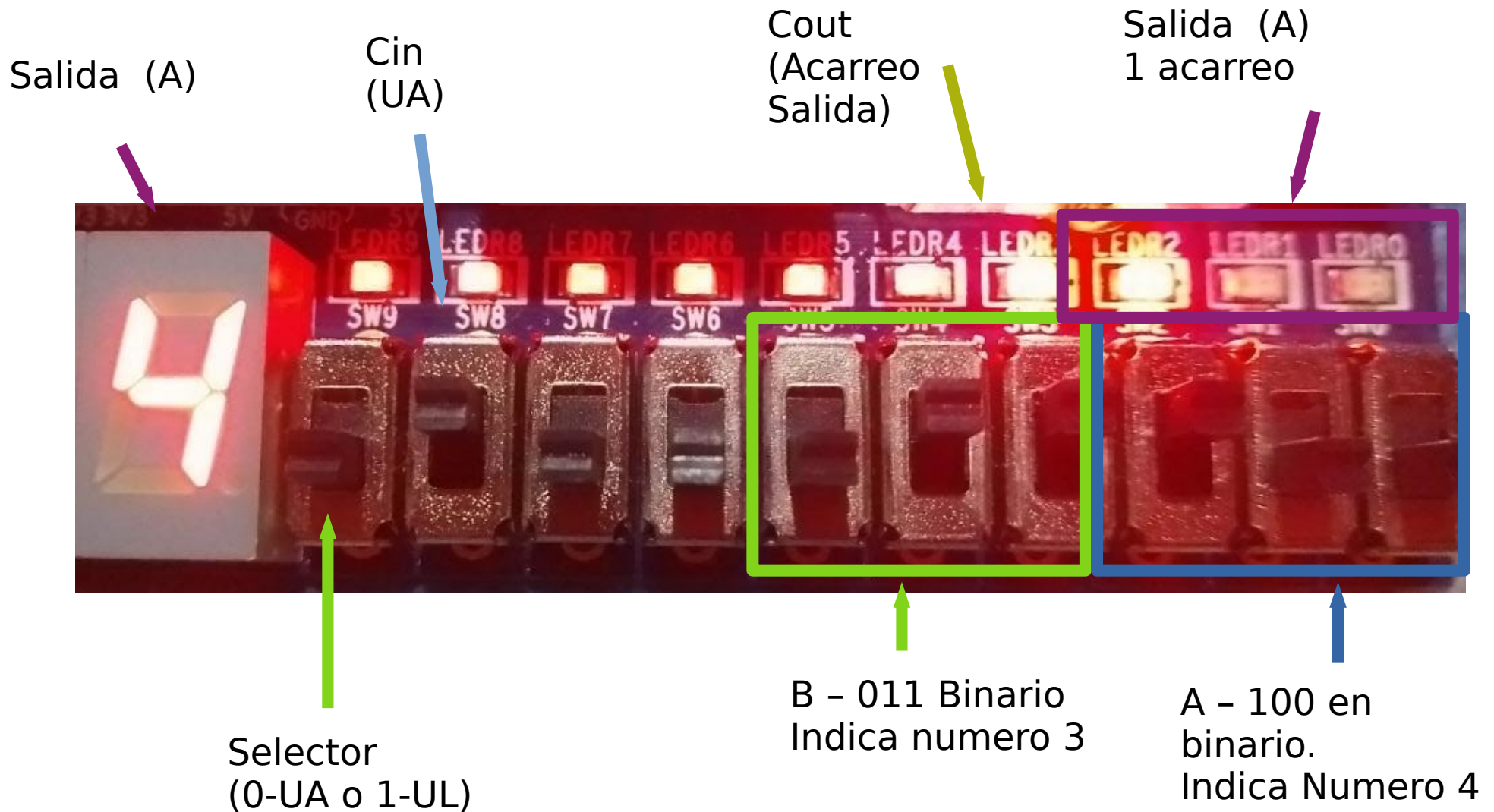
Pruebas en Tarjeta (UA - SUMA A+B+1)



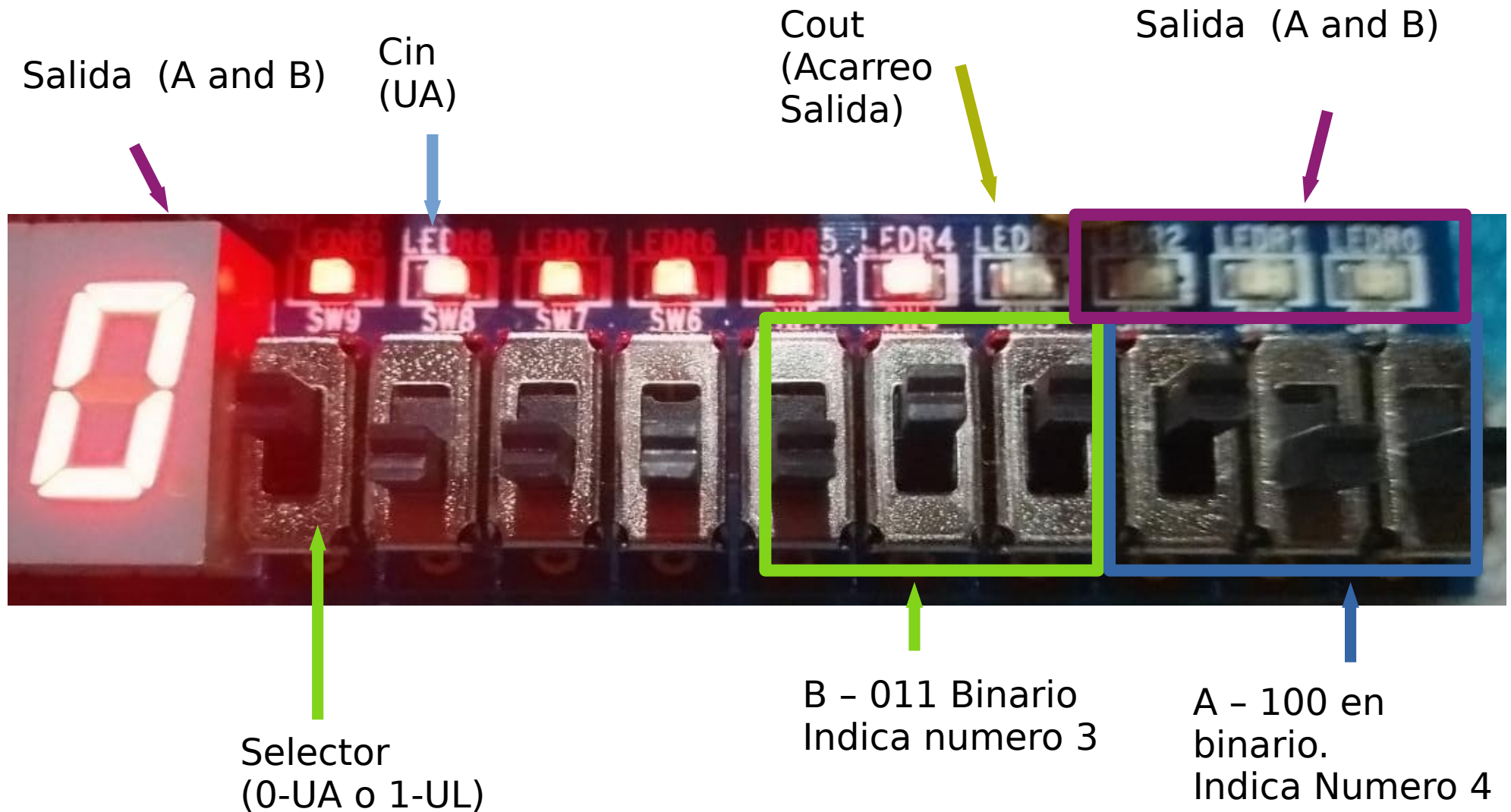
Pruebas en Tarjeta (UA - SUMA A-B)



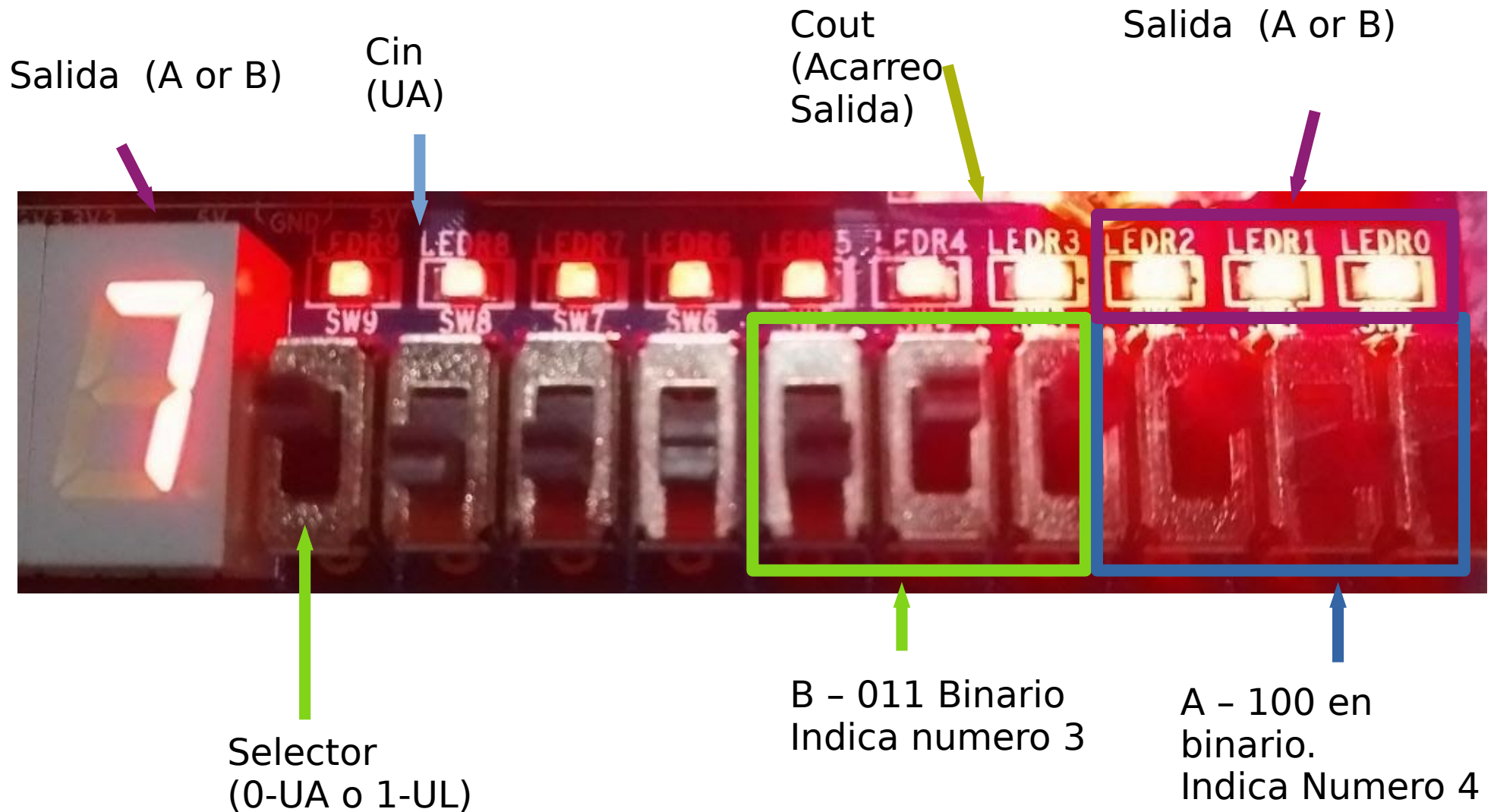
Pruebas en Tarjeta (UA - SUMA A acarreo)



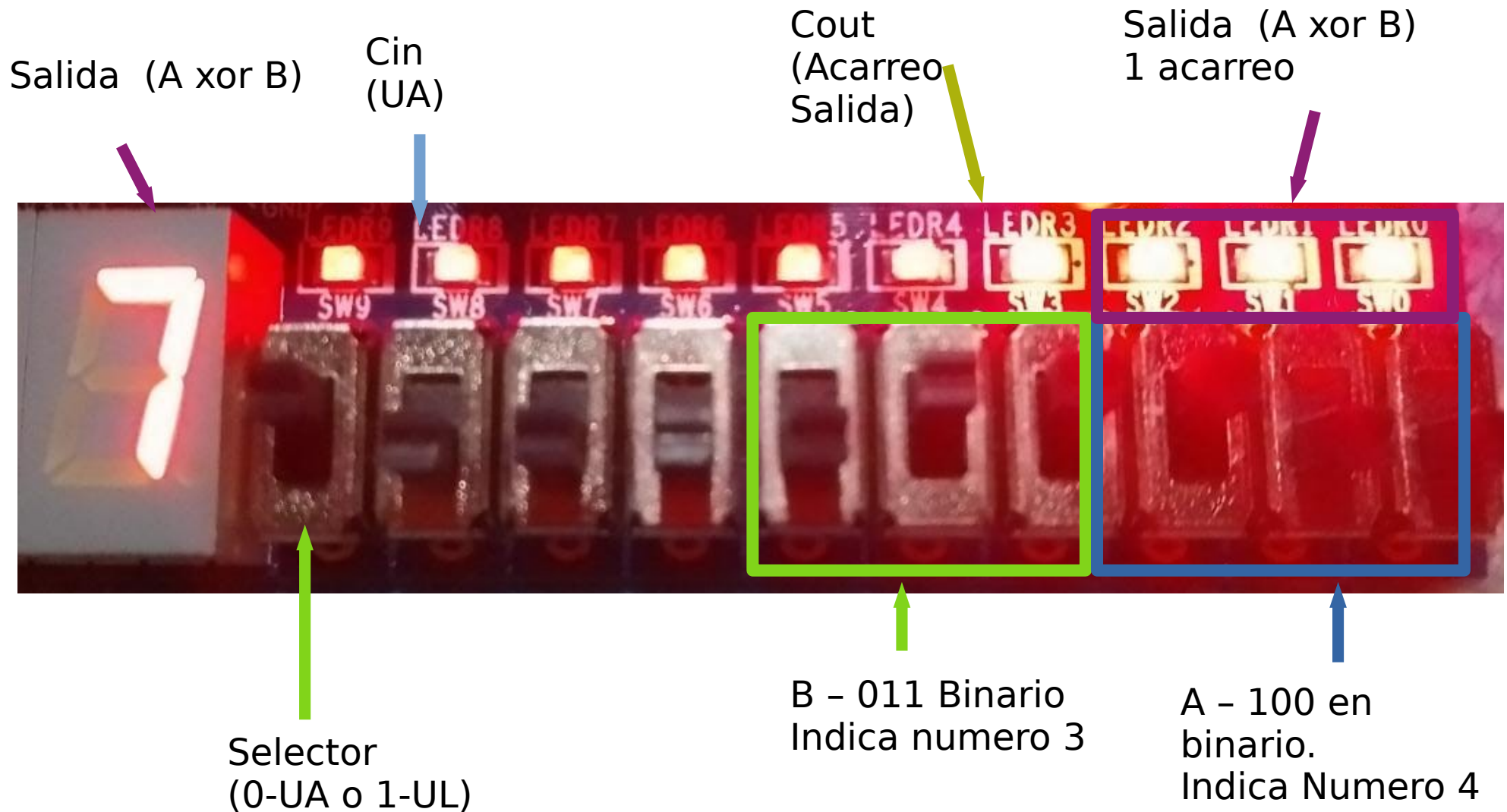
Pruebas en Tarjeta (UL - A and B)



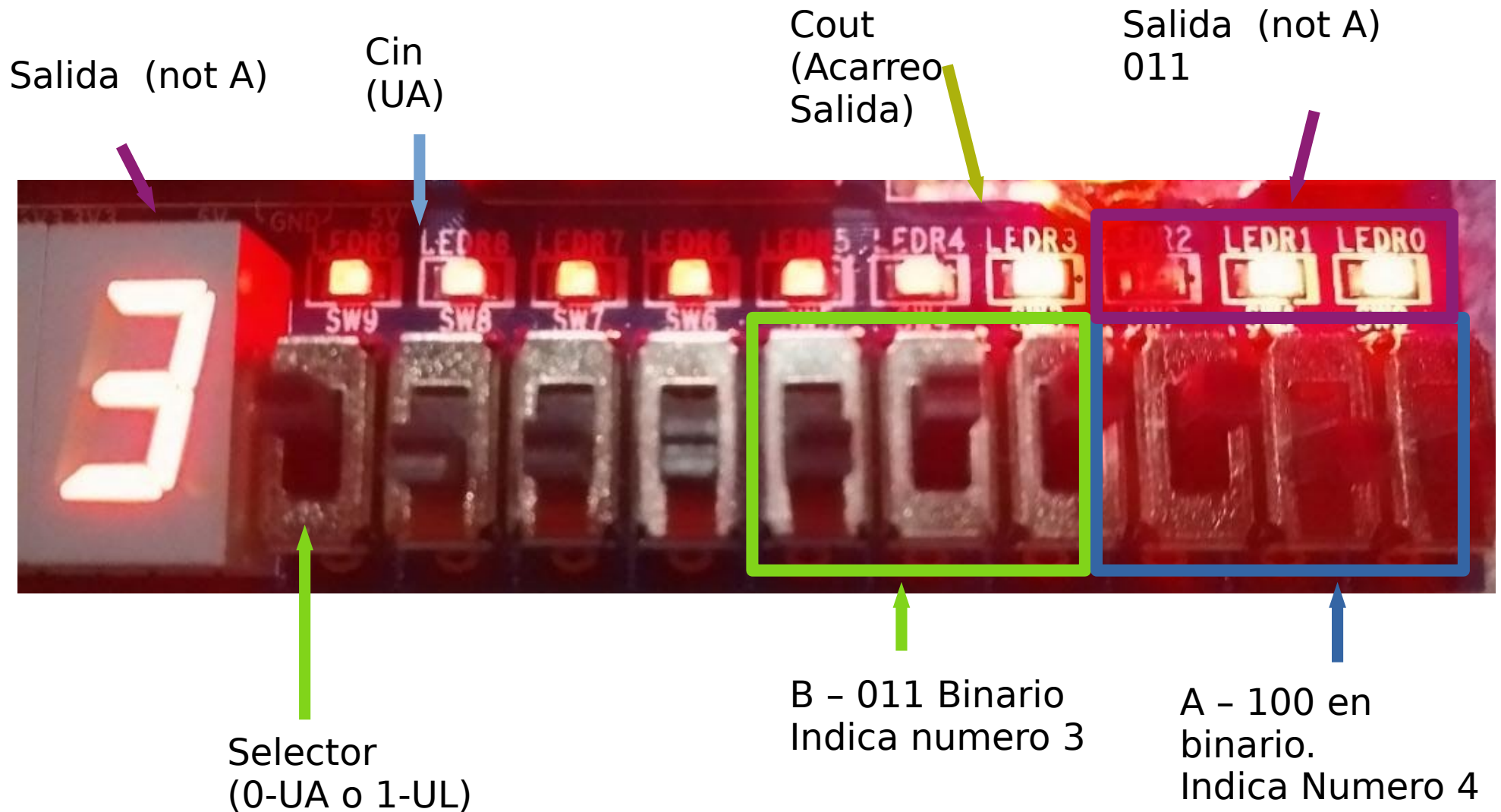
Pruebas en Tarjeta (UL - A or B)



Pruebas en Tarjeta (UL - A xor B)



Pruebas en Tarjeta (UL - not A)



Conclusiones

Observamos el comportamiento de una ALU y pudimos automatizar los cambios de las operaciones a partir de un divisor de frecuencia para cambiar los estados de manera automática. Y con la ayuda de multiplexores pudimos seleccionar entre la unidad lógica y la unidad aritmética

Referencias

- **Fonseca E.[Profesora Elizabeth Fonseca](2020) ULA. [Archivo de video] Youtube.**
<https://www.youtube.com/watch?v=8vszGhGa1RI&t=6s>