

Parcial #1

Equipo:

Yasir Enrique Blandon Varela

Juan Pablo Rúa Cartagena

Docente Académico:

Alberto Mauricio Arias Correa

UNIVERSIDAD EAFIT

Departamento de Informática y Sistemas

Organización de computadores

Medellín – Antioquia

28 de febrero de 2025

Parcial #1 - Organización de computadores

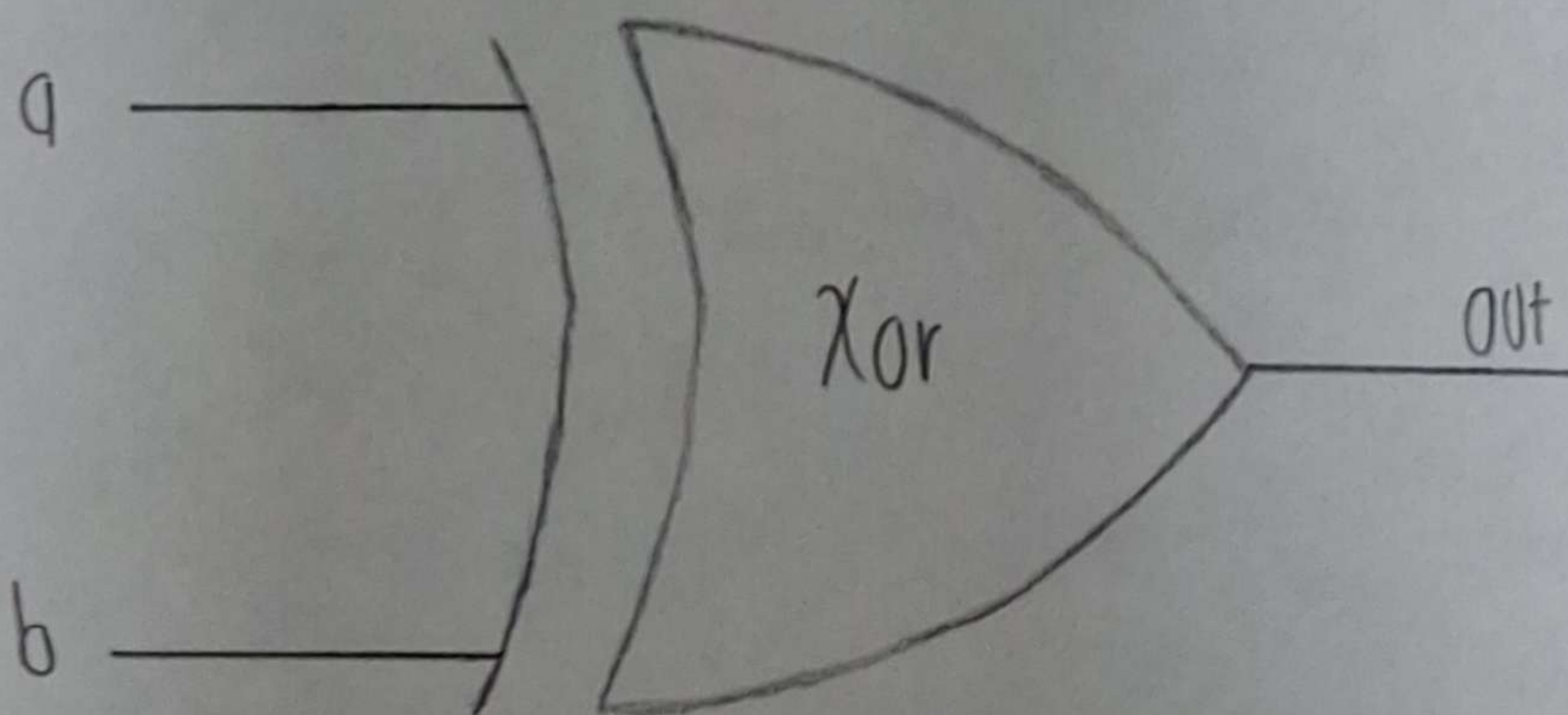
• Punto #1: Compuerta Xor

1.1 Tabla de verdad:

→
Hallar la ecuación

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

* Compuerta Xor:



1.2 Ecuación booleana:

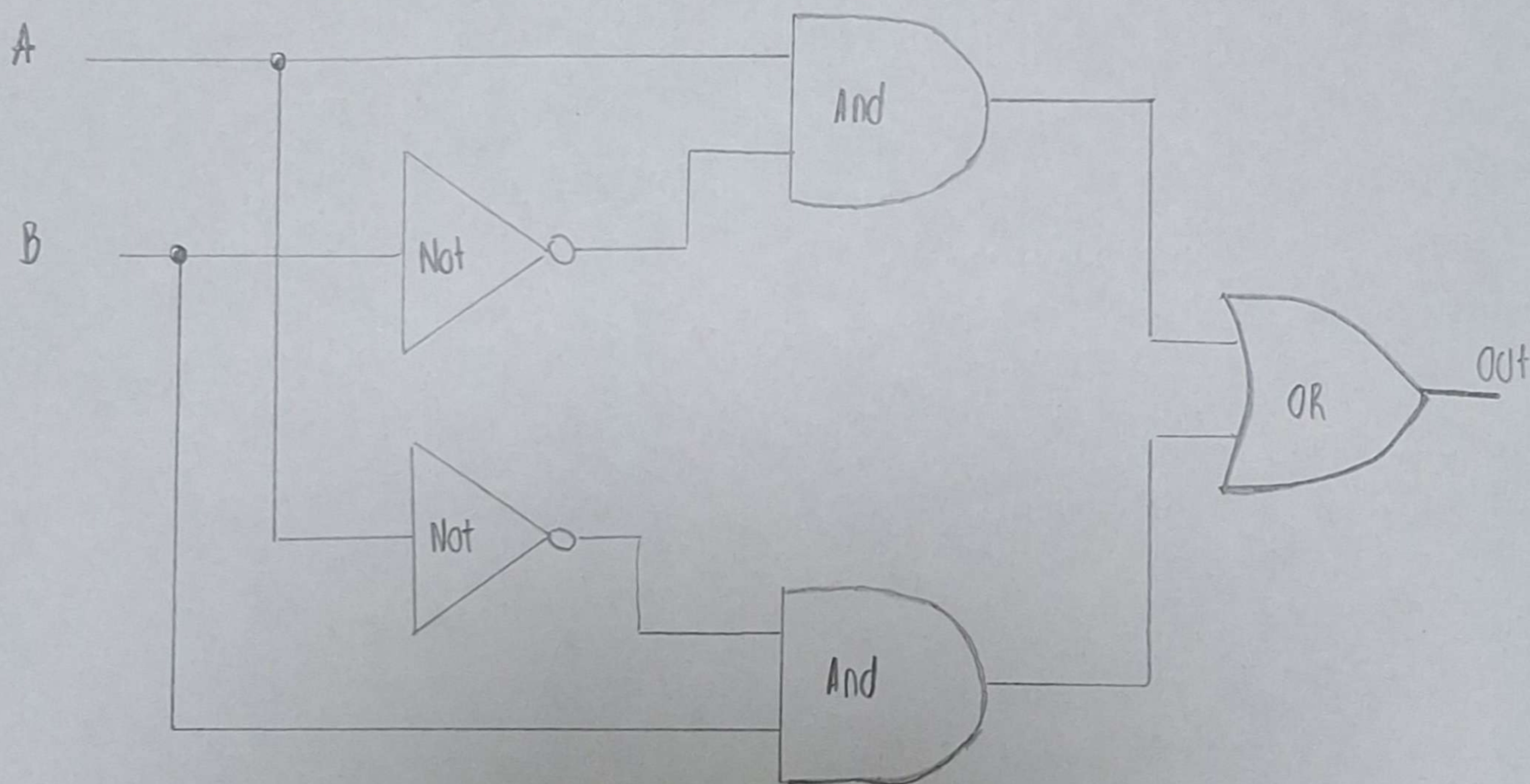
Utilizamos estas entradas

a	b	out
0	1	1
1	0	1

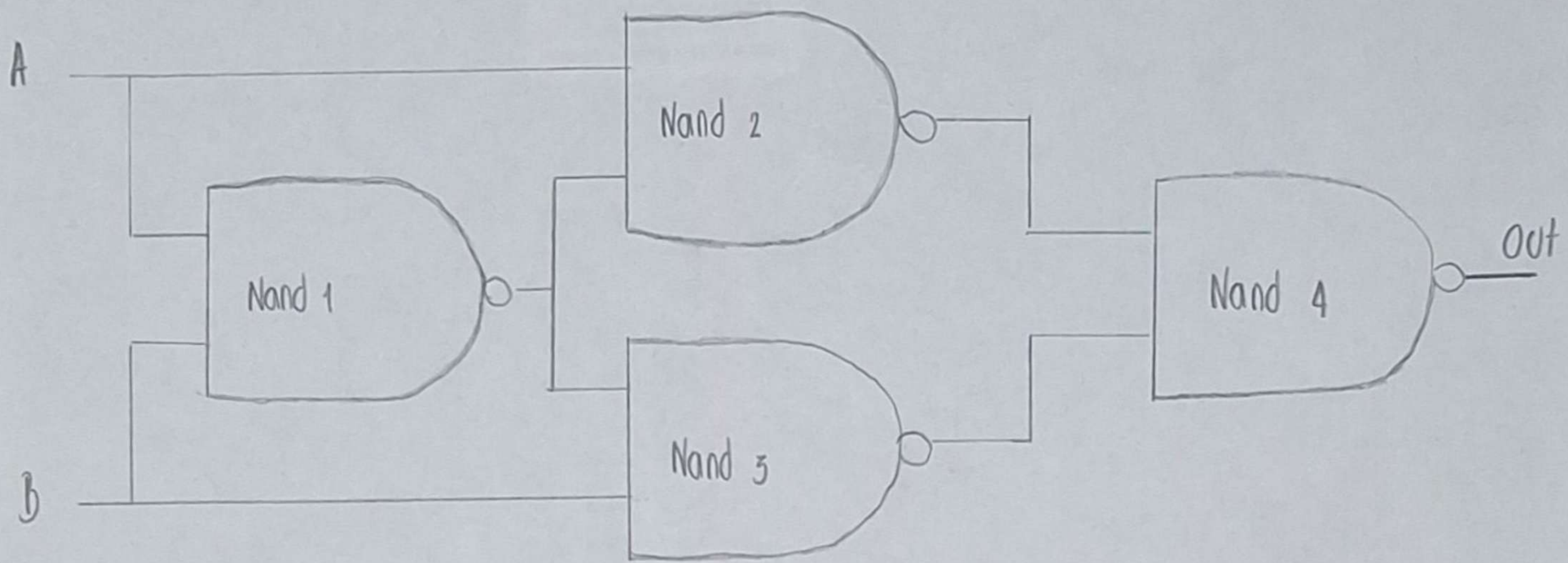
$$(\bar{a} \cdot b) + (a \cdot \bar{b})$$

$$a \oplus b = (a \cdot \bar{b}) + (\bar{a} \cdot b) \quad \text{Ecuación booleana}$$

* Diagrama (Compuertas basicas)



1.3 Diagrama (Compuertas nand):



1.4 HDL Xor

```
CHIP Xor {  
  IN a, b;  
  OUT out;
```

PARTS:

```
Nand (a = a, b = b, out = nandAB);  
Nand (a = a, b = nandAB, out = part1);  
Nand (a = b, b = nandAB, out = part2);  
Nand (a = part1, b = part2, out = out);
```

}

• Punto #2: Compuerta And

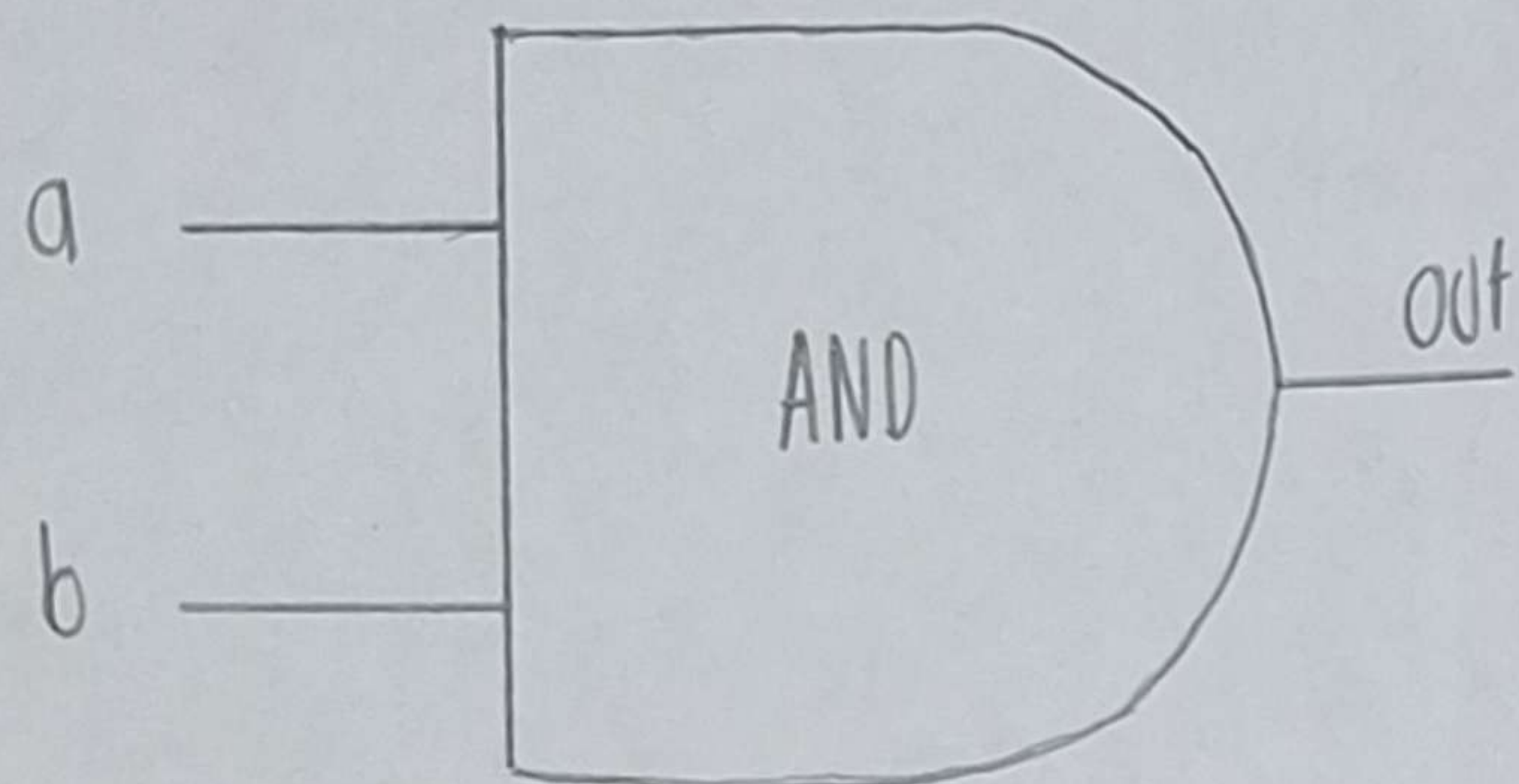


Tabla de verdad

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

2.1 HDL AND

CHIP And {

IN a,b;

OUT out;

PARTS:

Nand(a=a, b=b, out=nandOut);

Nand(a=nandOut, b=nandOut, out=out);

}

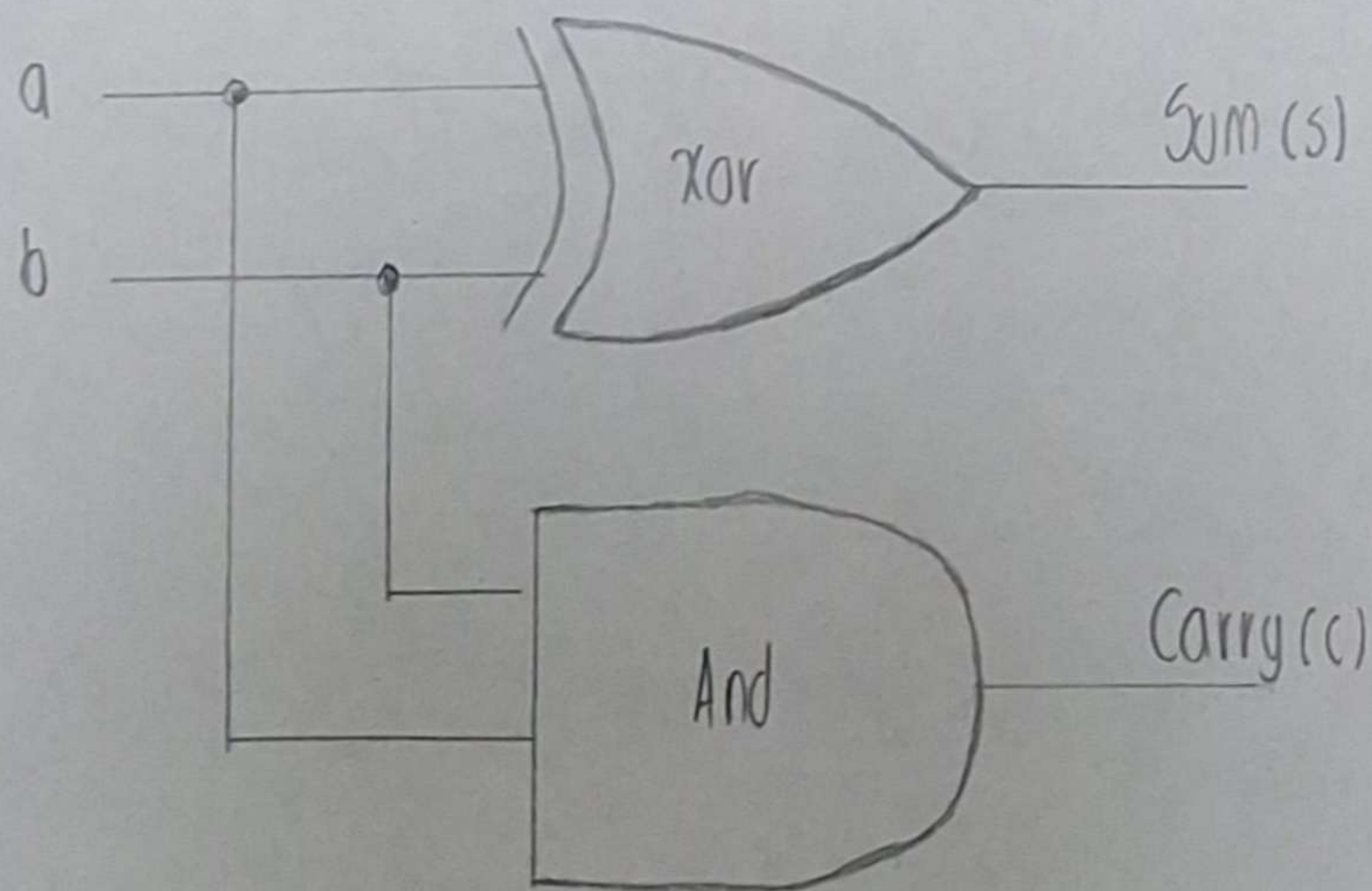
• Punto #3: Half adder

3.1 Tabla de verdad

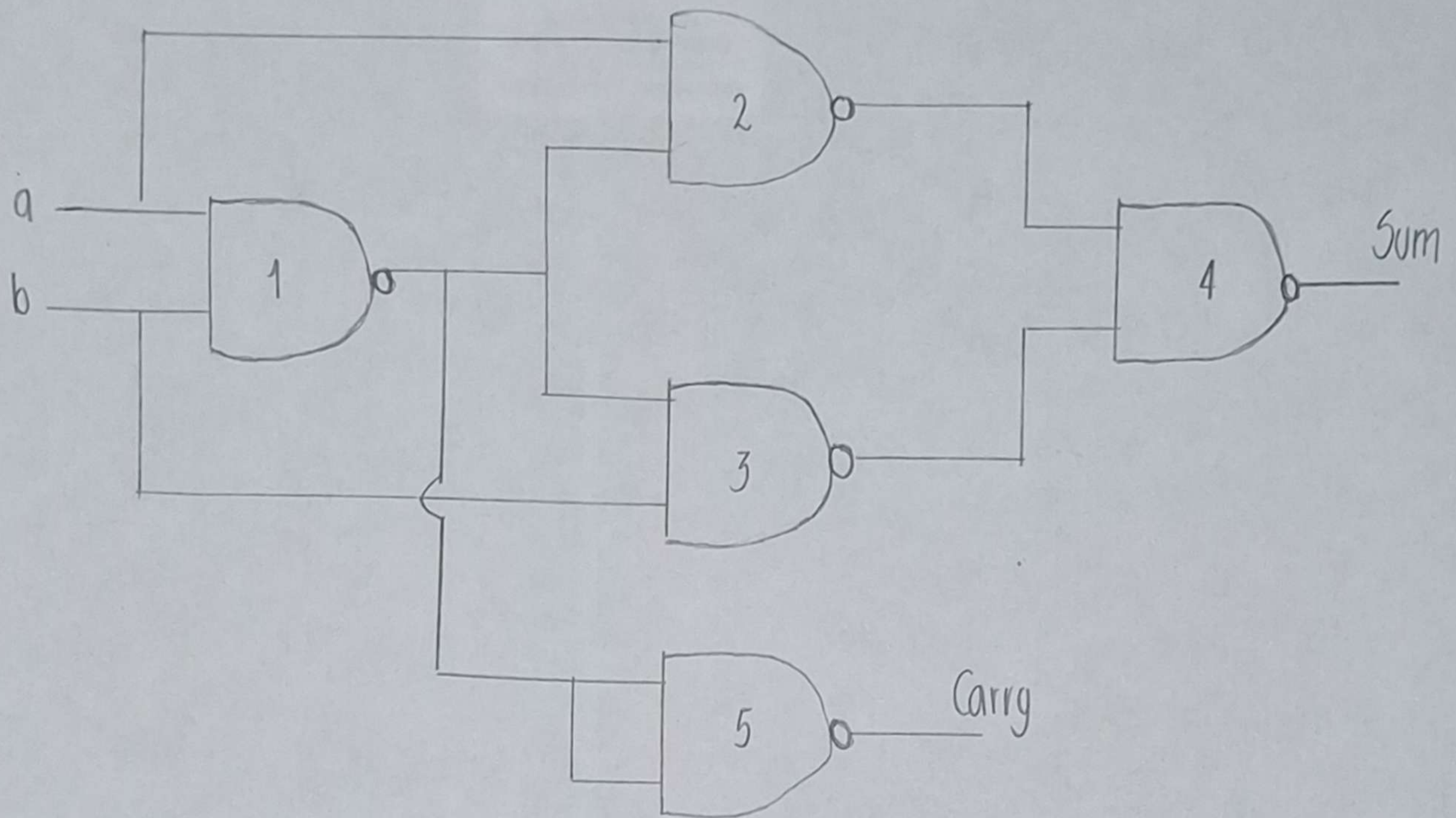
a	b	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = a \oplus b ; C = a \cdot b$$

3.2 Diagrama (compuertas basicas)



3.3 Diagrama (compuertas nand)



3.4 HDL Half adder:

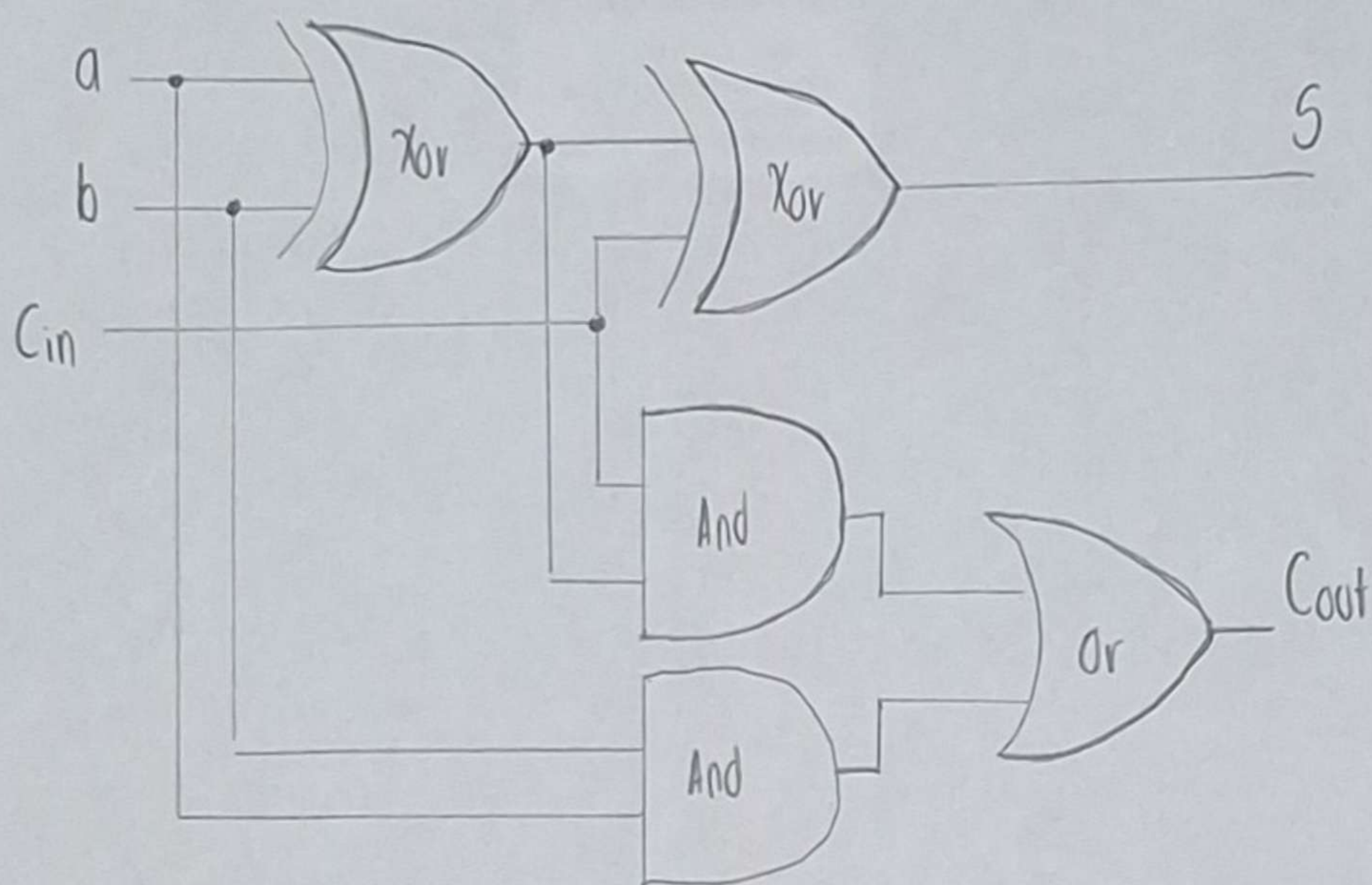
R/ Implementado en nand2tetris

• Punto # 4 = Full adder

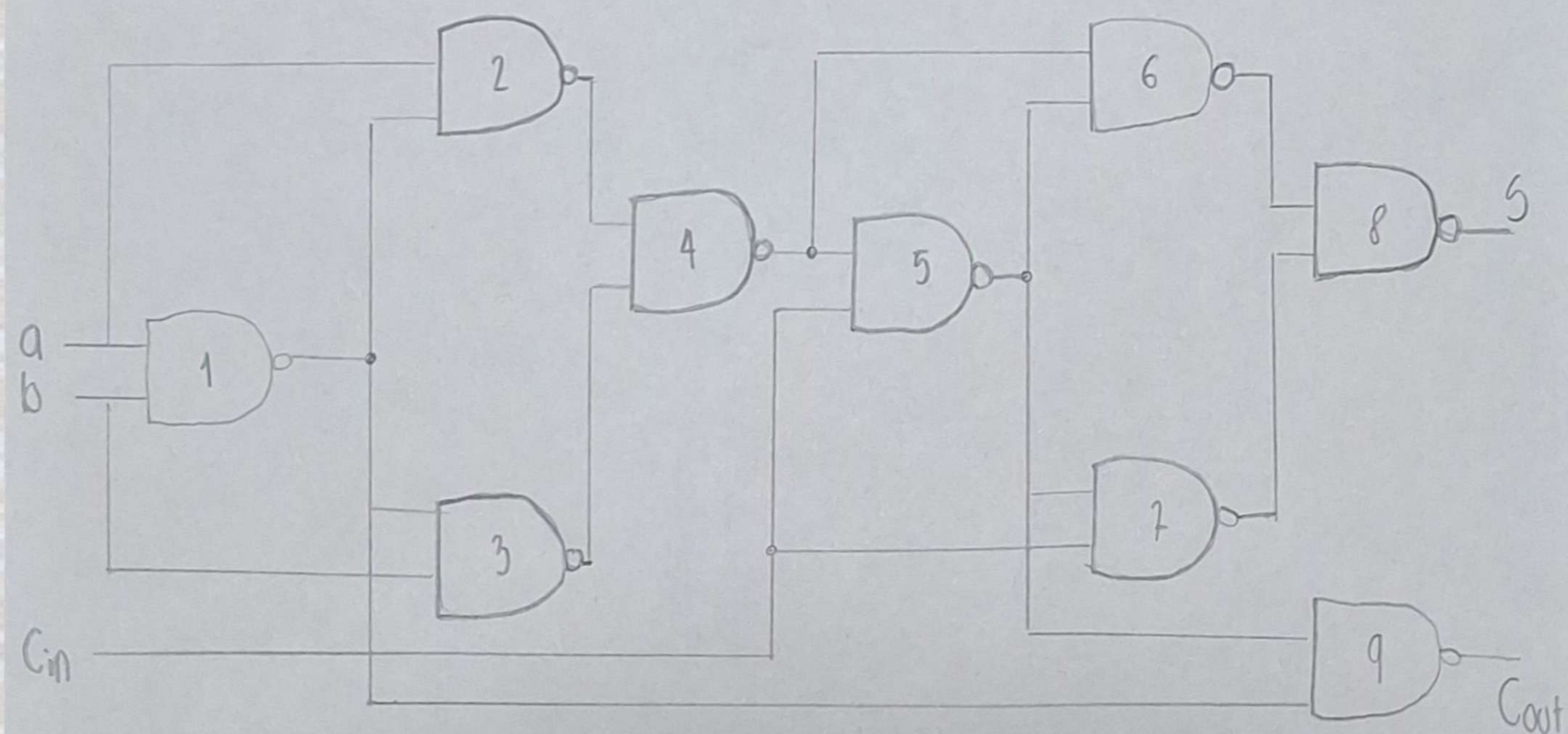
4.1 Tabla de verdad

a	b	Cin	Sum	Car
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

4.2 Diagrama (compuertas basicas)



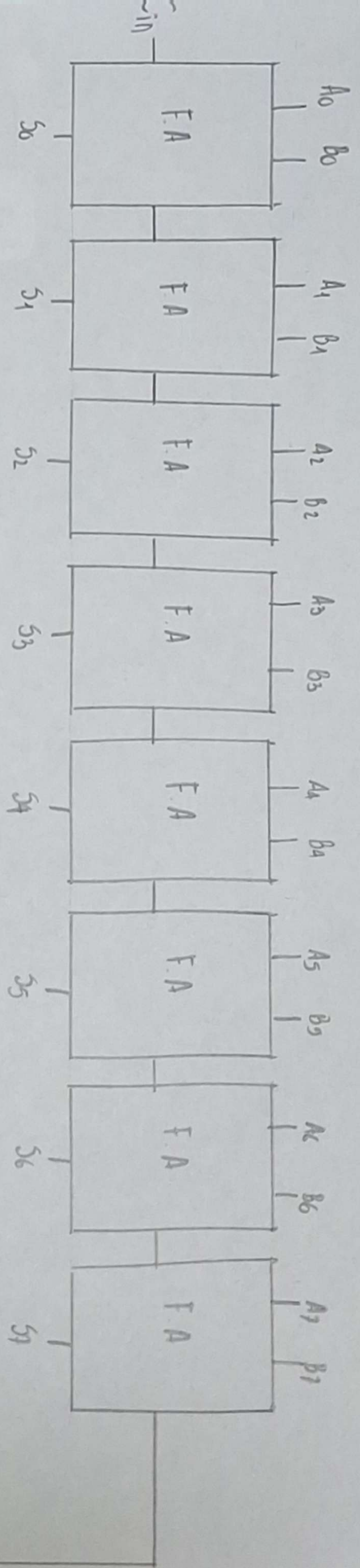
4.3 Diagrama (compuertas nand)



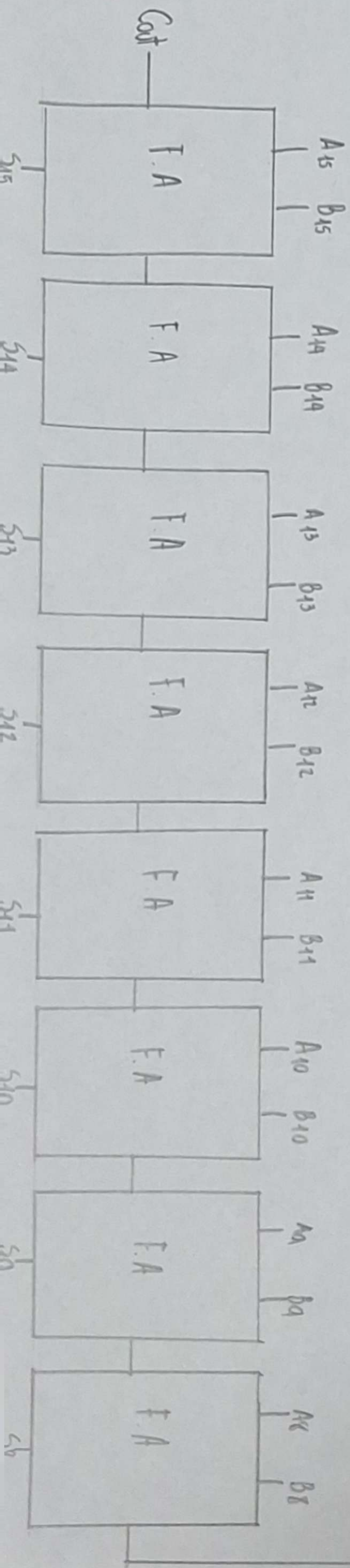
4.4 HDL full adder

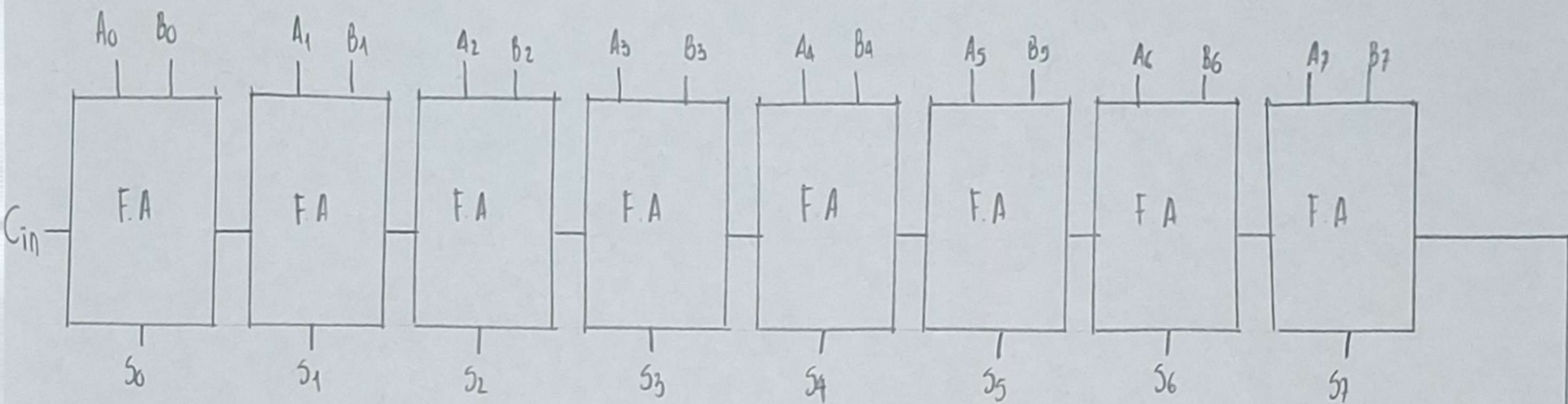
RI Implementado en nand2tetris

Punto #5: Adder 16 bits



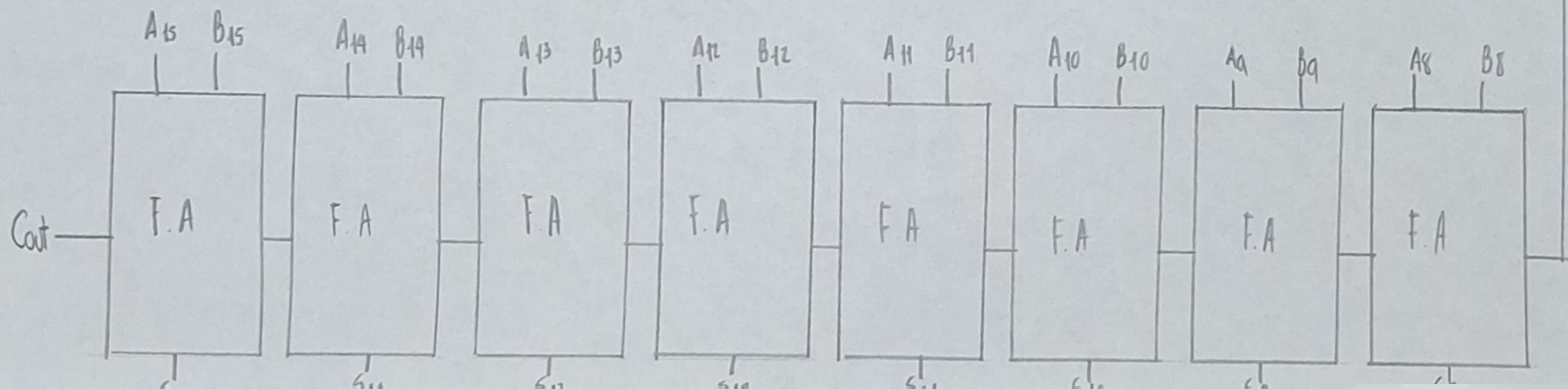
(Adder: 16 bits)





(Adder: 16 bits)

Punto #5: Adder 16 bits



• Punto #7: Funcionamiento de la ALU cuando el resultado es negativo

En una ALU, el bit más significativo de la salida indica si el número es negativo (MSB, Most significant bit)

- Si $MSB = 0$, el número es positivo o cero

- Si $MSB = 1$, el número es negativo

→ Proceso: La ALU toma 'x' y 'y' de 16 bits y aplica los bits de control

Luego, evalúa el resultado. La salida $out[16]$ es evaluada

- Se verifica el bit más significativo ($out[15]$):

- Si $out[15] == 1$; la ALU establece la bandera $ng = 1$

lo cual quiere decir que el resultado es un número negativo

- Si $out[15] == 0$; la bandera $ng = 0$ (valor positivo).

• Punto #8: Funcionamiento de la ALU cuando la salida es cero.

- La ALU necesita verificar si todos los bits de la salida son 0. Para ello, se usa la bandera Zr.

→ Proceso: La alu procesa 'x' y 'y' generando una salida $out[16]$

*Se usa una combinación de OR's para revisar si al menos un bit es 1

↳ Si $OR(out[0]...out[15]) == 0$, entonces todos los bits son 0 y $Zr = 1$

Si al menos un bit en la cadena de bits es 1, $Zr = 0$