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:

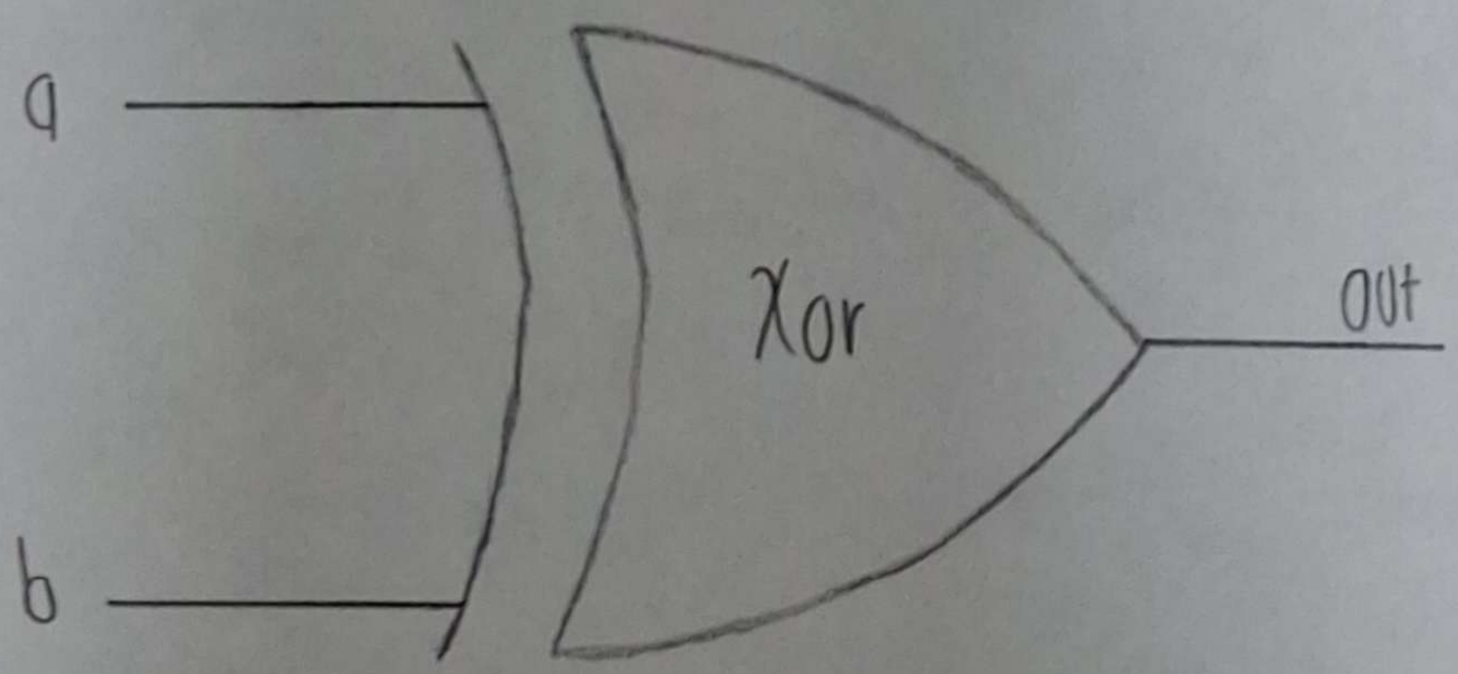
a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

→

→

Hallar la ecuación

* Compuerta Xor:



1.2 Ecuación booleana:

Utilizamos estas entradas

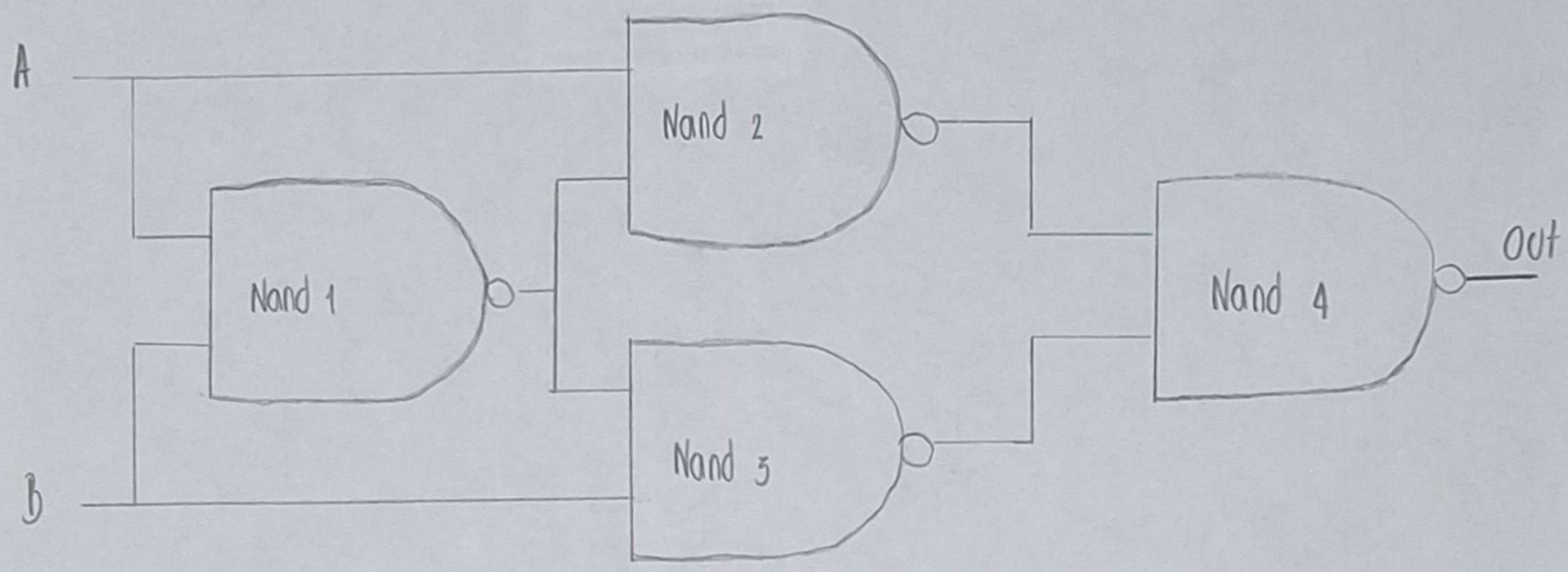
a	b	out
0	1	1
1	0	1

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

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

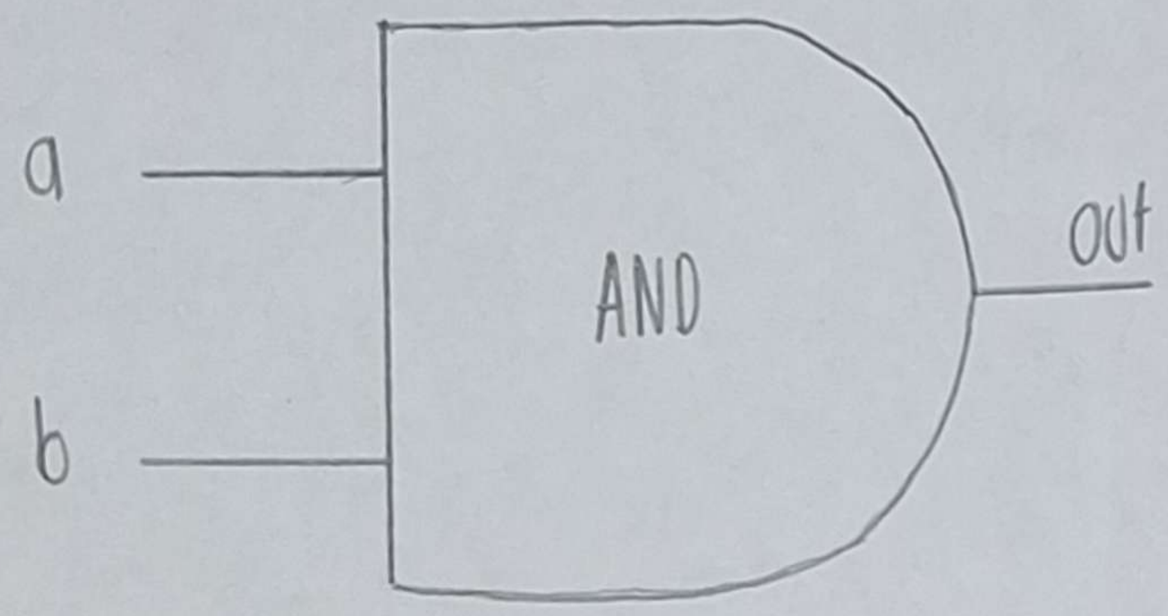


Tabla de verdad

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);
}
```

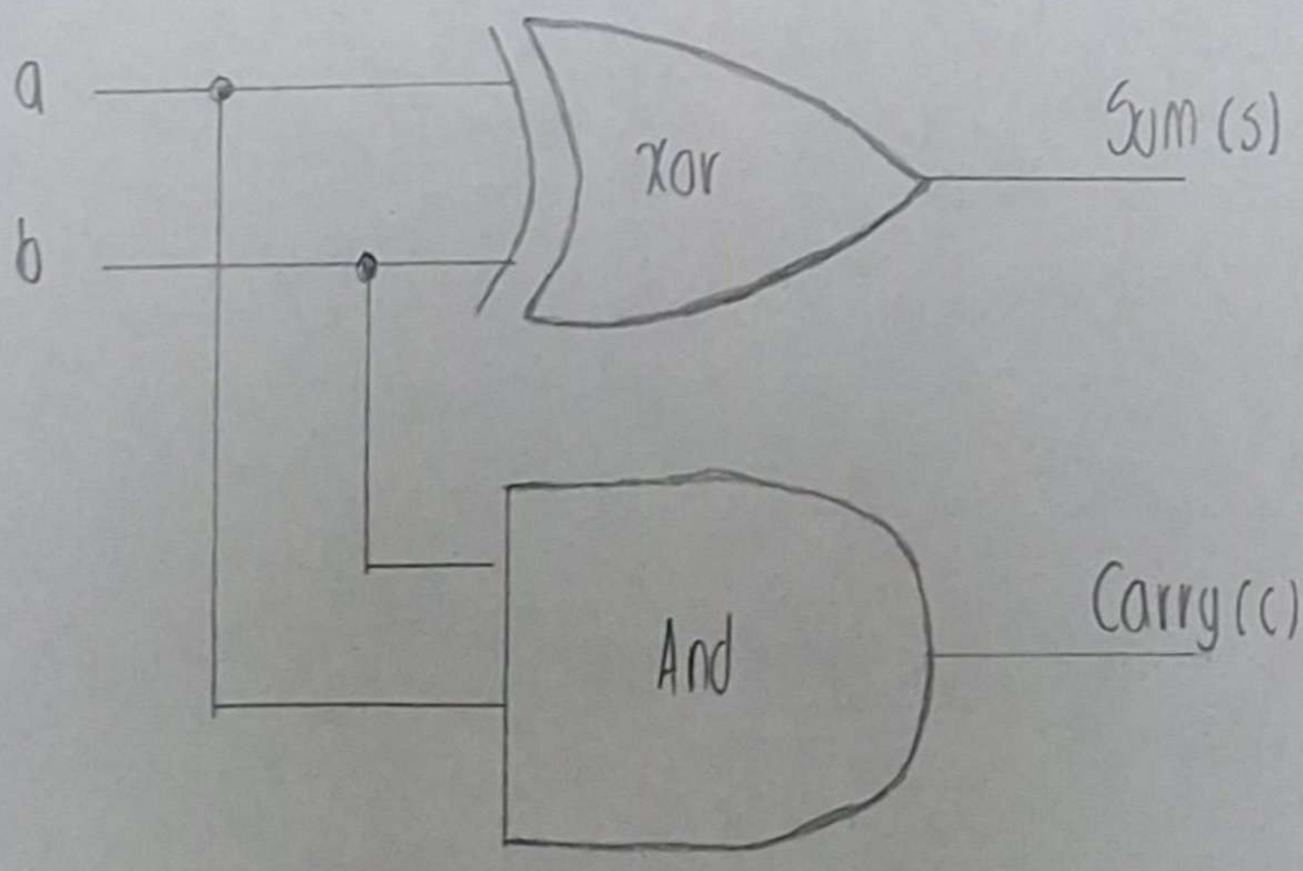
- Punto #3: Half adder

3.1 Tabla de verdad

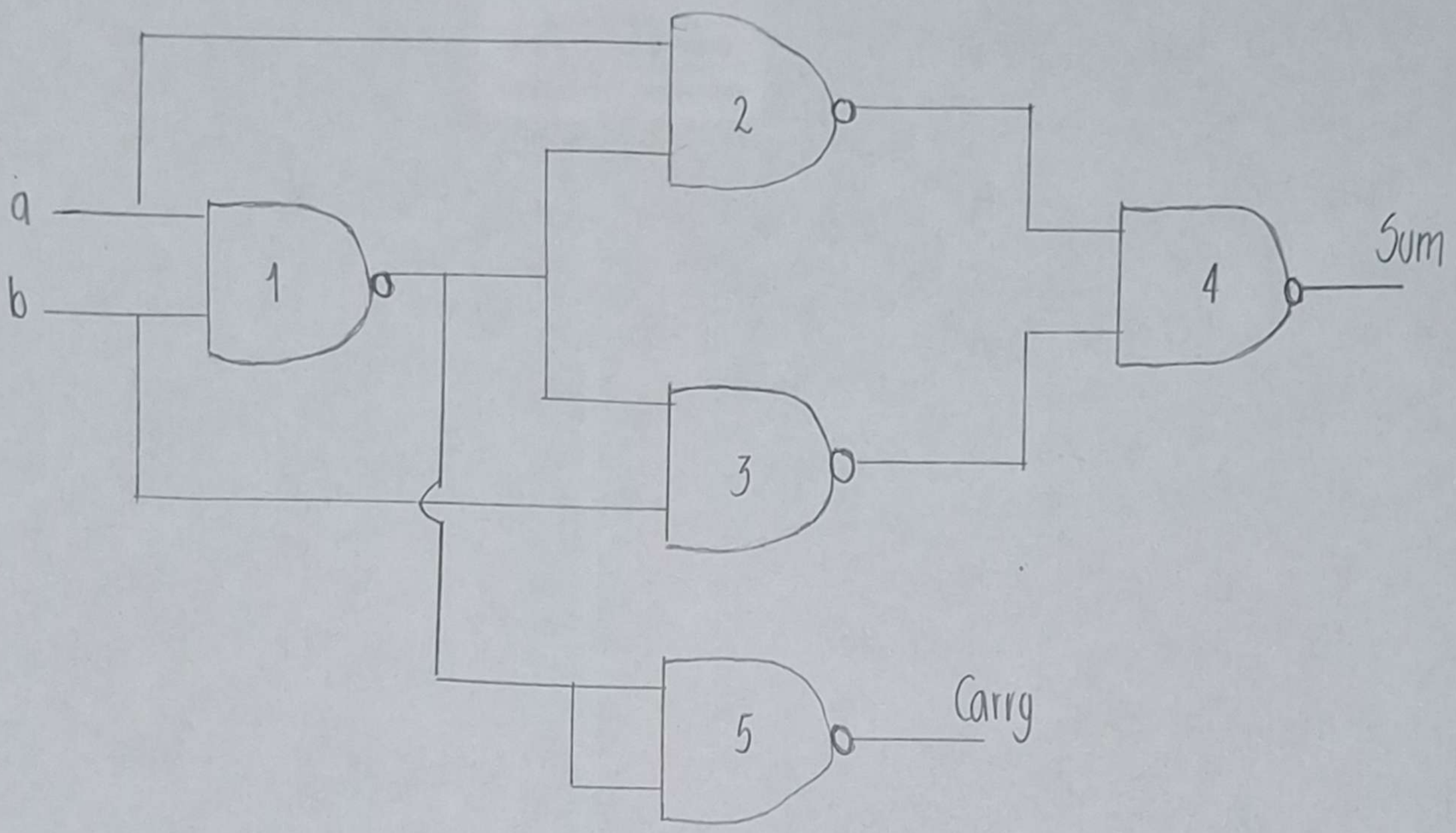
a	b	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

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

3.2 Diagrama (Compuertas basicas)



3.3 Diagrama (compuertas nand)



3.4 HDL Half adder:

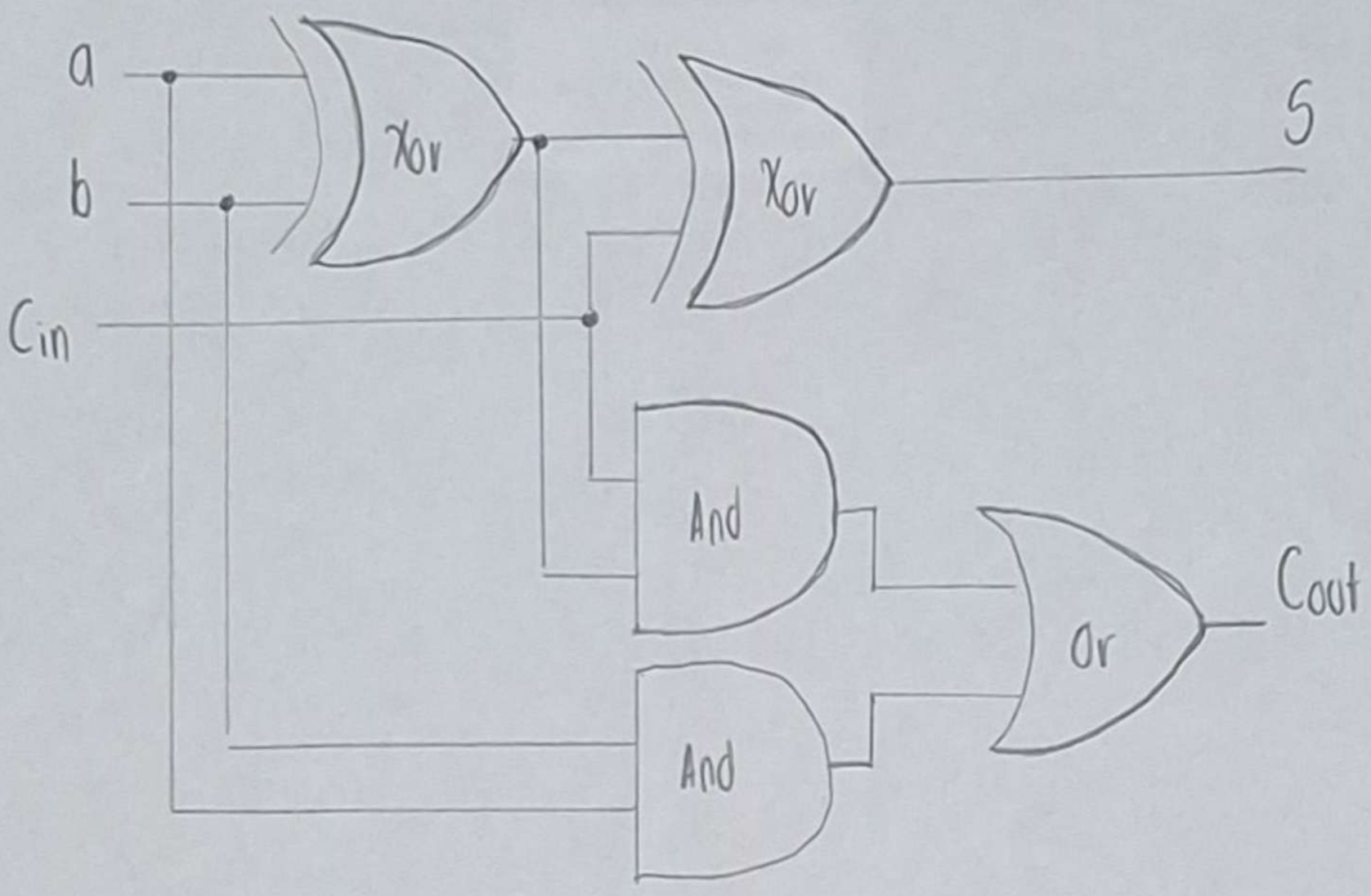
R/ Implementado en nand2tetris

- Punto #4 : Full adder

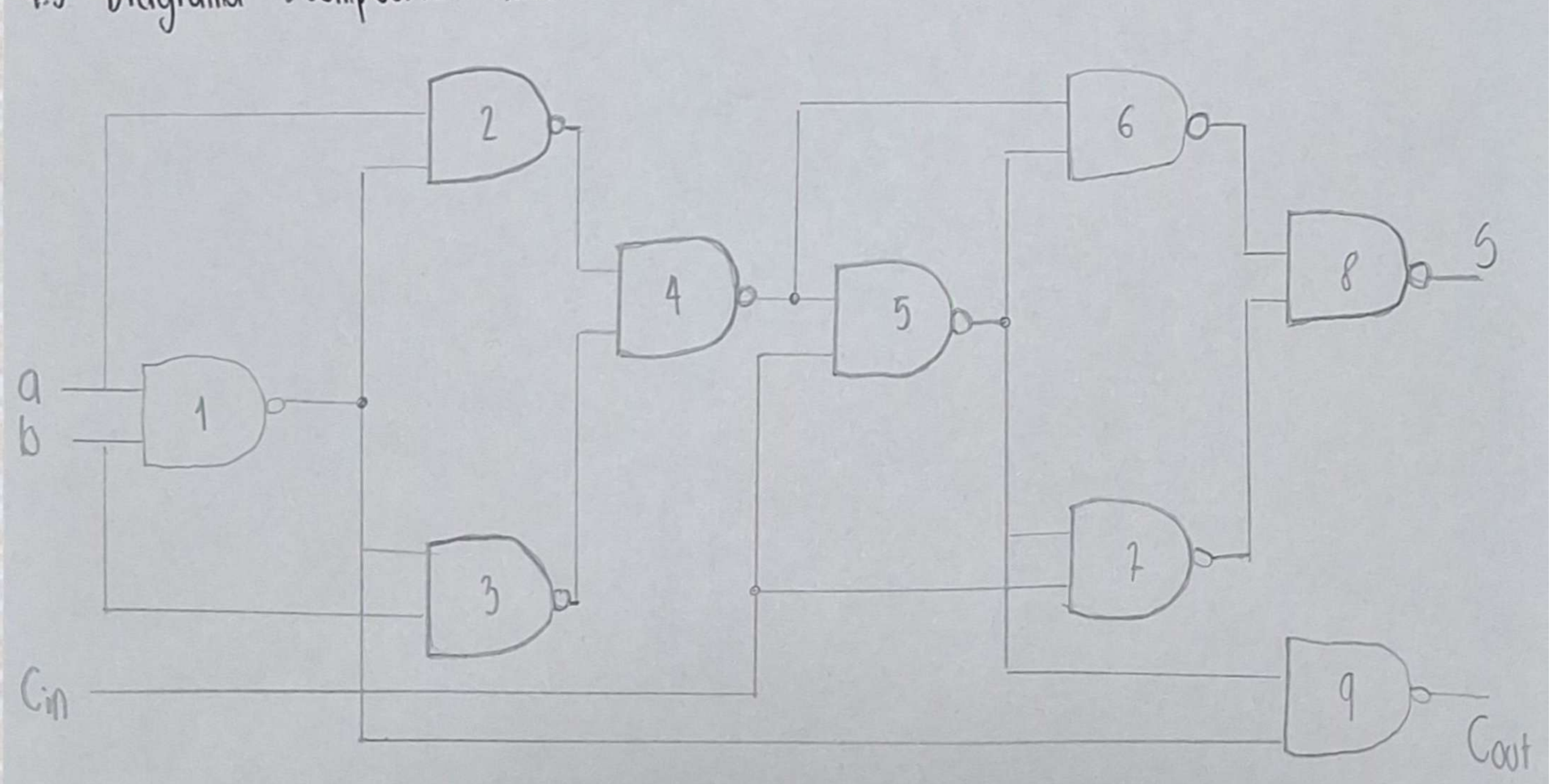
4.1 Tabla de verdad

a	b	Cin	Sum	Cout
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)

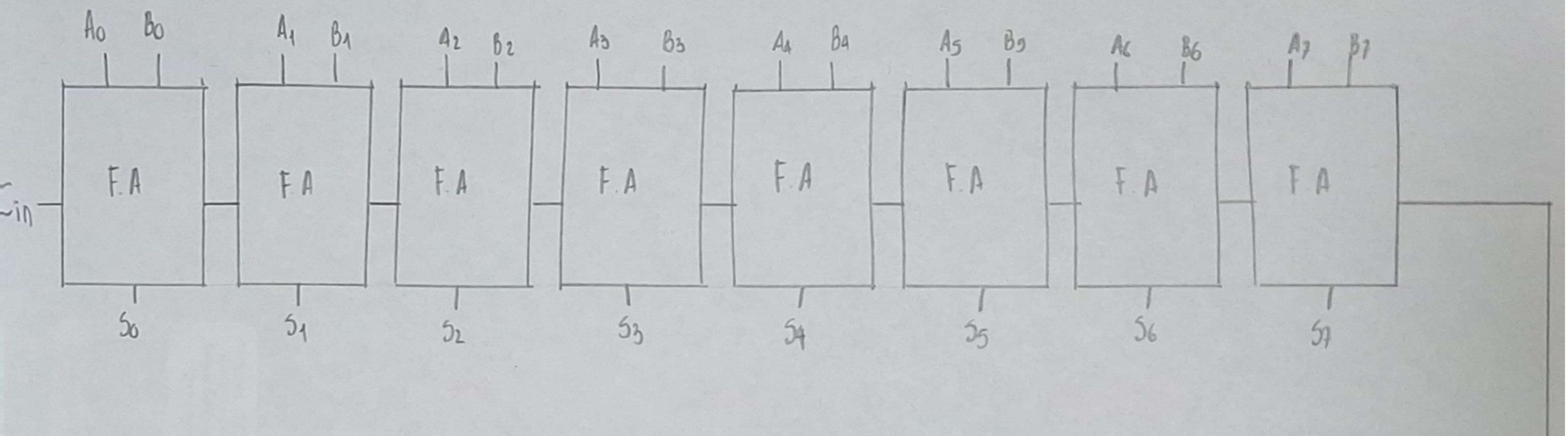


4.3 Diagrama (compuertas nand)



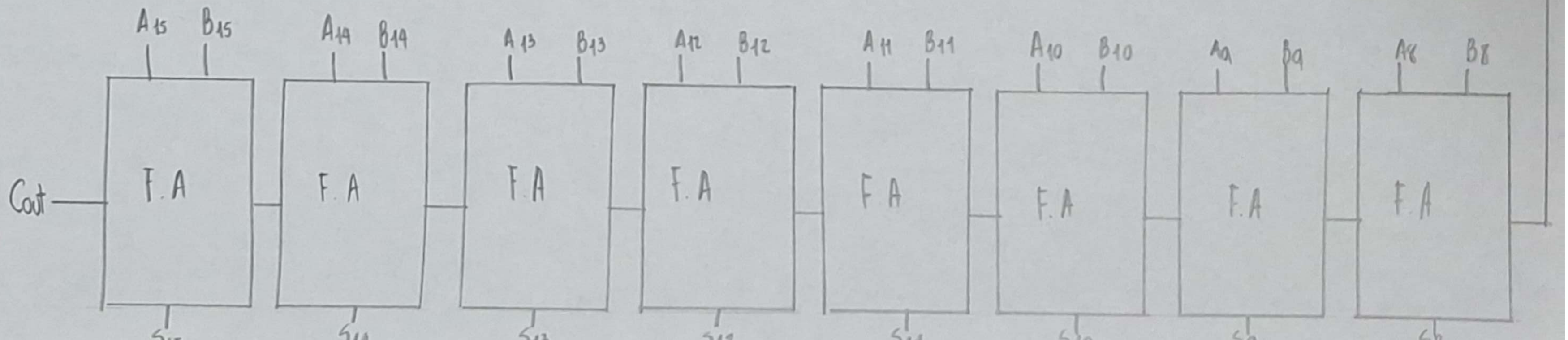
4.4 HDL Full adder

RI Implementado en nand2tetris

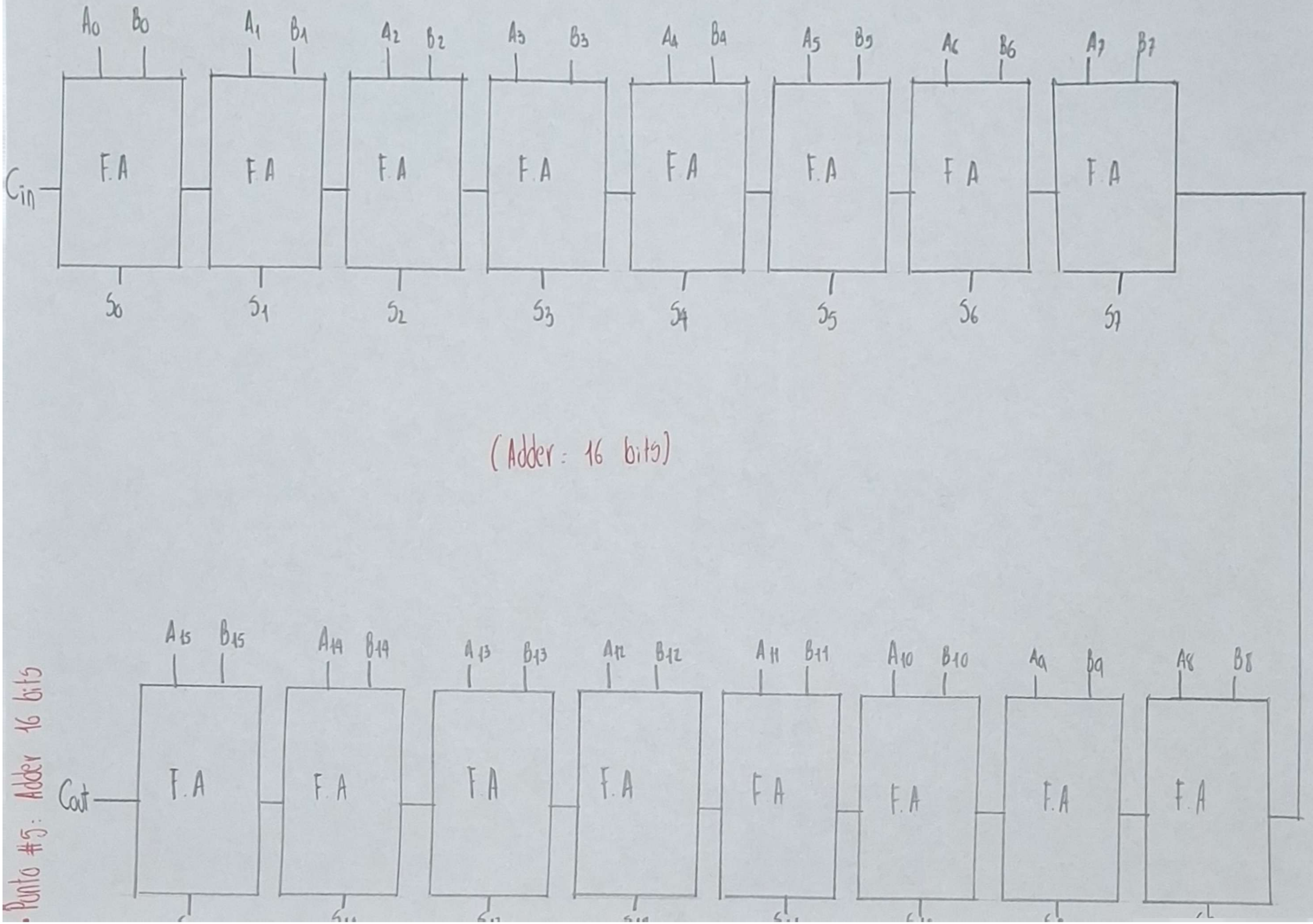


(Adder: 16 bits)

- Punto #5: 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 mas significativo de la salida indica si el numero 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, evalua el resultado: La salida out[16] es evaluada

- Se verifica el bit mas 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 almenos un bit en la cadena de bits es 1, Zr = 0

- Punto #9: ALU

- ¿Para que necesita un computador una ALU?

- Porque es el nucleo de procesamiento de cualquier CPU, esta se encarga de ejecutar las operaciones matematicas y logicas esenciales para el funcionamiento de cualquier sistema de cómputo; operaciones como suma, resta, AND, OR, NOT, XOR, etc...

Sin un ALU, el procesador no podria realizar dichas operaciones o cualquier otras operaciones que requiera manipulación de datos, lo que haria imposible la ejecución de los programas.

- ¿Por que no se hacen operaciones aritmeticas y logicas directamente en el procesador?

- No se hacen operaciones directamente en el procesador porque la ALU esta especificamente diseñada y optimizada para ejecutarlas de manera eficiente, mientras que el procesador se encarga de la gestión y control del flujo de instrucciones. Por otro lado, aquello ayuda a la optimización del hardware ya que la ALU es un bloque especializado que ejecuta las operaciones mucho mas rápido que si el procesador tuviera que hacerlas manualmente con instrucciones de bajo nivel.

Todo esto propicia una separación de funciones, en donde la CPU se compone de múltiples partes: Unidad de control, Memoria cache, ALU, etc, y cada una comple una Función.

- ¿Las ALUs de los computadores modernos procesan solamente dos entradas de 16 bits cada una?

- No, las alus de los computadores modernos pueden procesar diferentes tamaños de datos como 32, 64 o mas bits dependiendo de la arquitectura del procesador y el tipo de operación requerida

Ejemplo: los procesadores x86-64 usan una ALU de 64 bits para soportar operaciones avanzadas como instrucciones SIMD para calculos en paralelo.