

Hardware, part 1

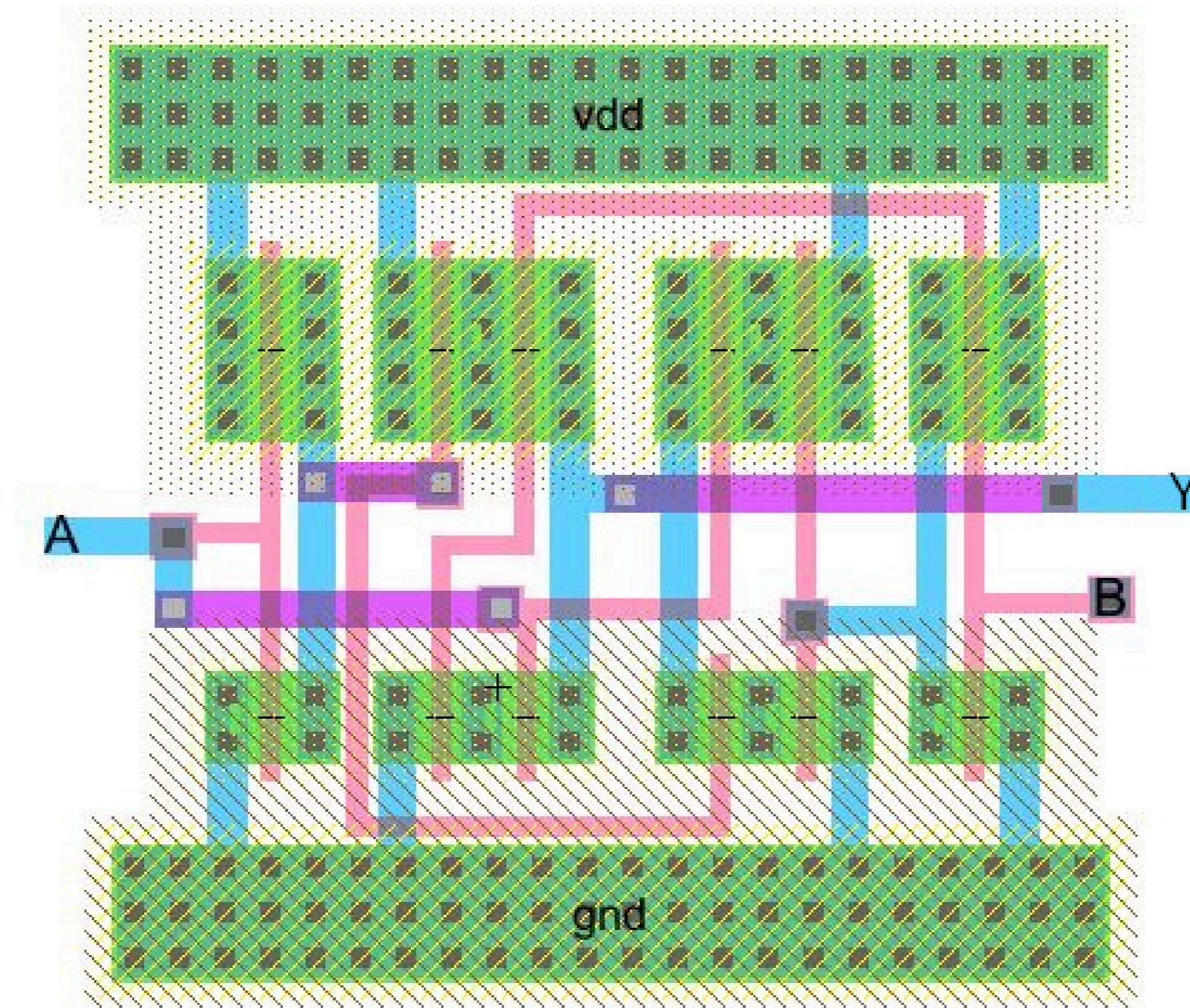
Levels of abstraction in IC design

Levels of abstraction

0. integrated circuit (IC) layout
1. transistors
2. logic gates
3. “intellectual property” (IP) block

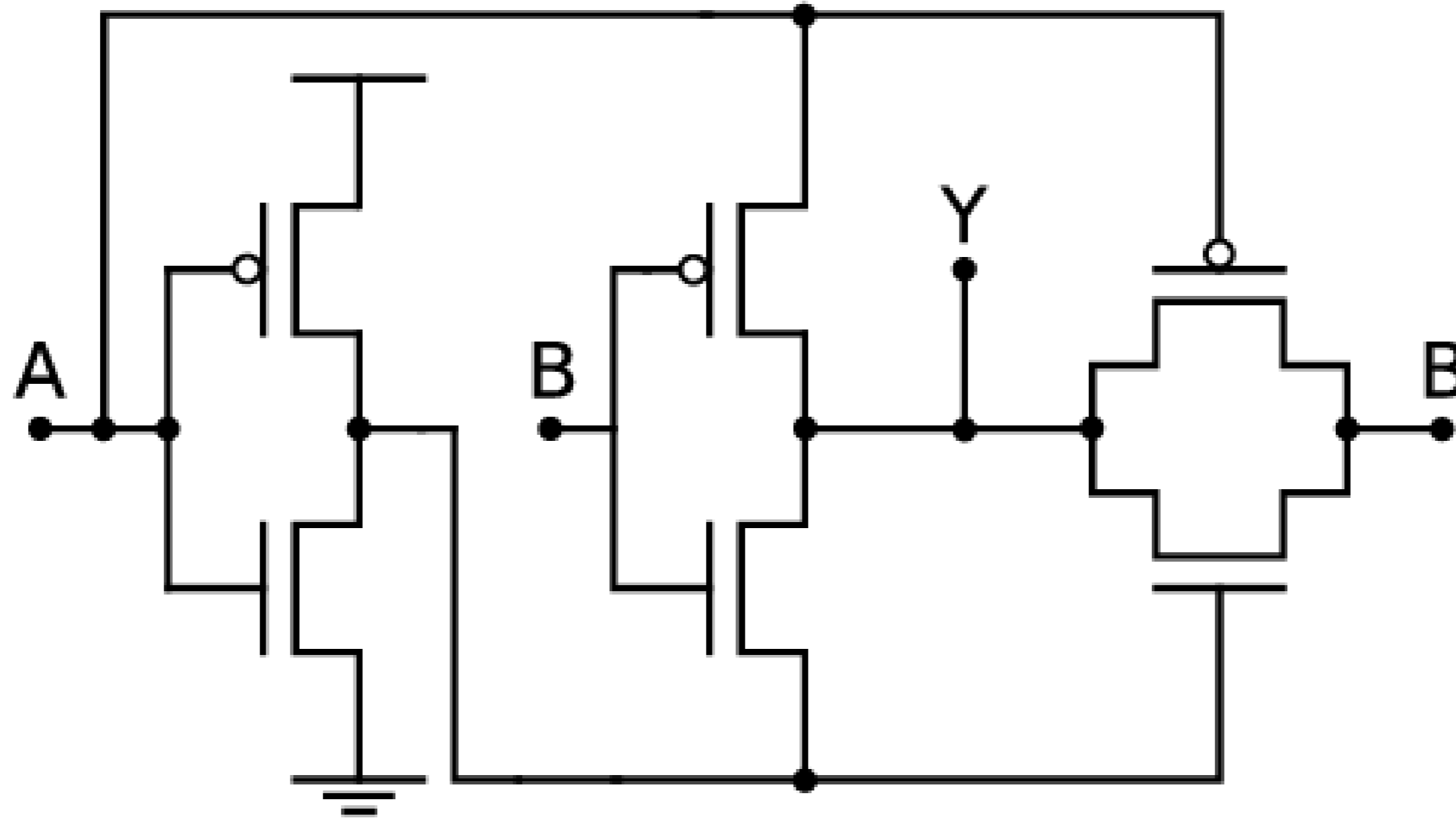


0. IC layout



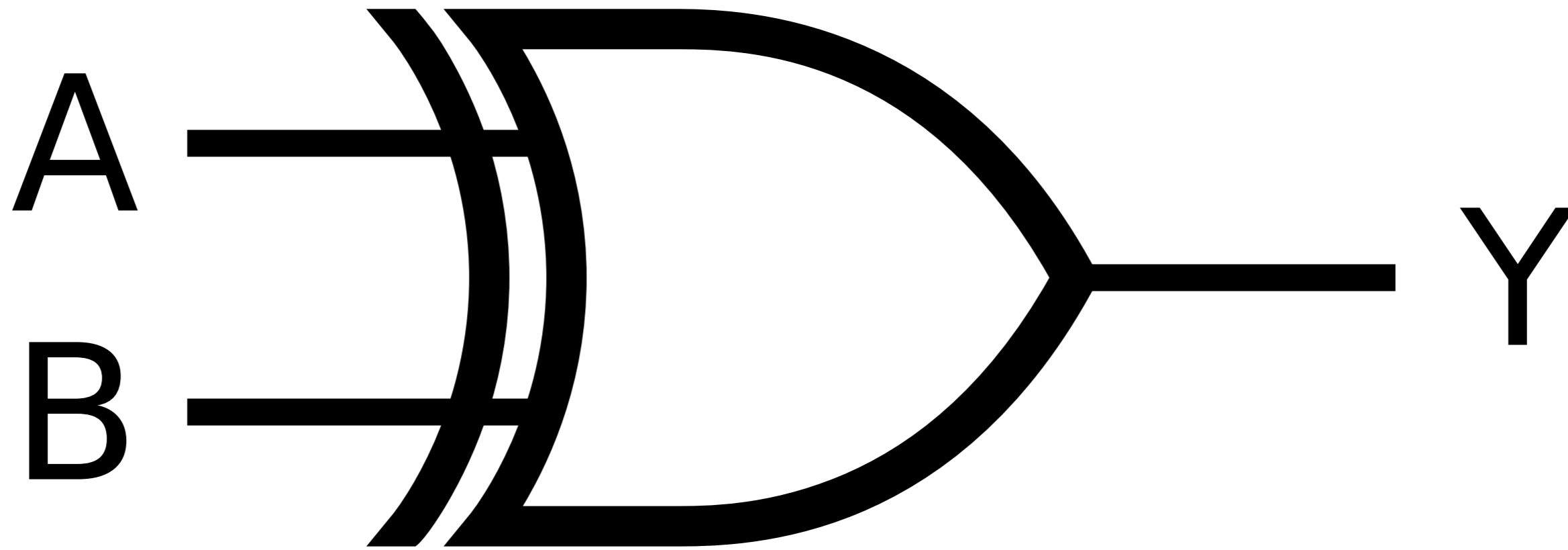
$Y := A \text{ xor } B$ (IC layout)

1. Transistors



$Y := A \text{ xor } B$ (transistors)

2. Logic gates



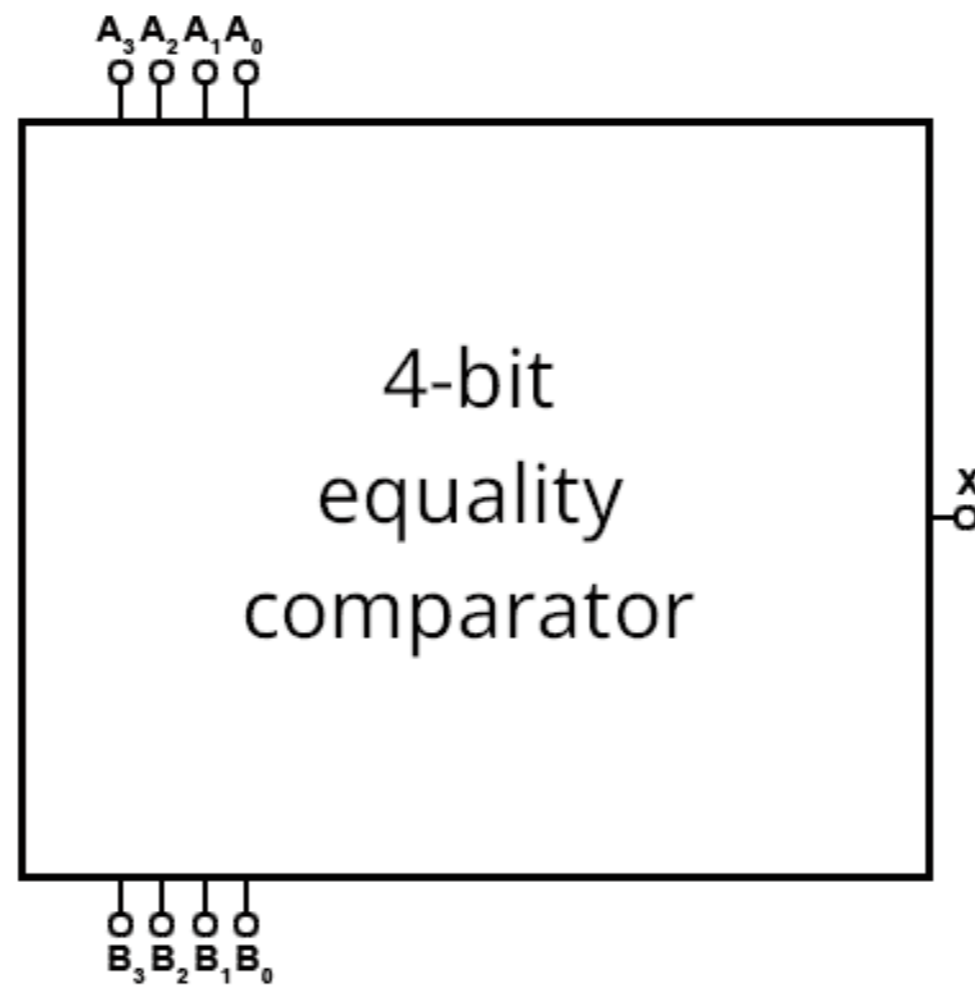
$Y := A \text{ xor } B$ (logic gate)

3. IP blocks

4-bit equality comparator:

$$\begin{aligned} \text{assuming } a &= A_3 \times 2^3 + A_2 \times 2^2 + A_1 \times 2^1 + A_0 \times 2^0 \\ \text{and } b &= B_3 \times 2^3 + B_2 \times 2^2 + B_1 \times 2^1 + B_0 \times 2^0 \end{aligned}$$

```
if a == b then x := 1
if a != b then x := 0
```

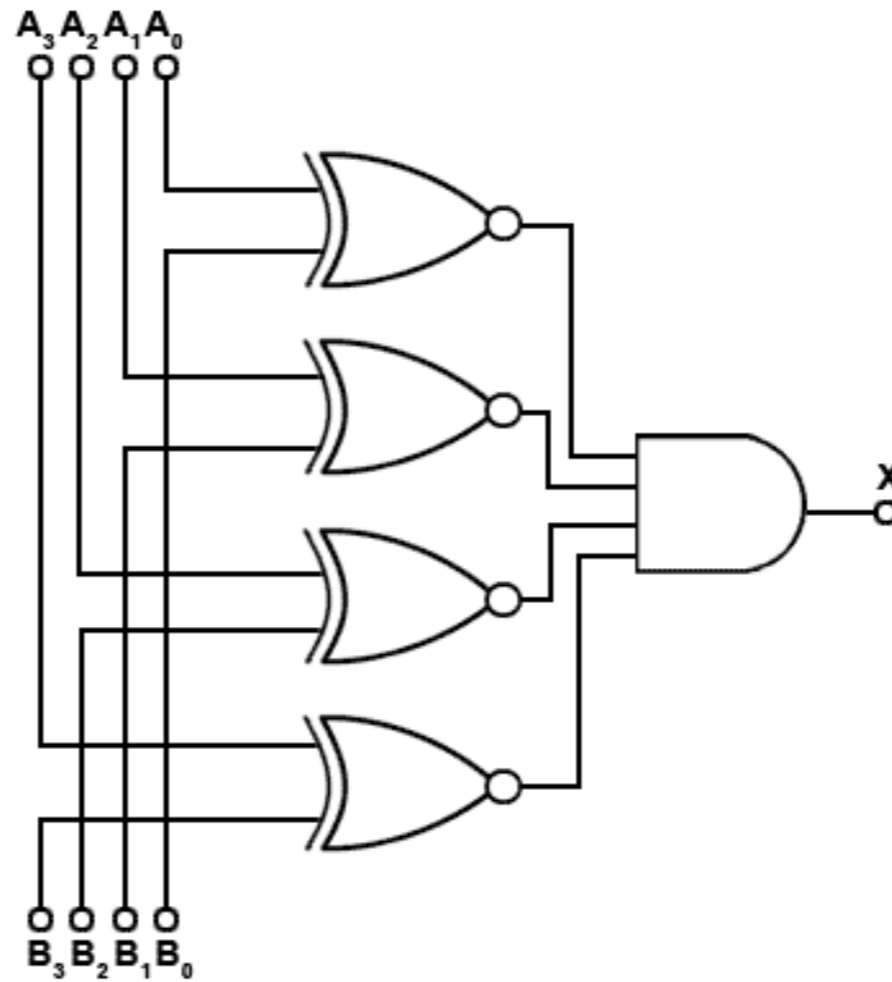


3. IP blocks

4-bit equality comparator:

$$\begin{aligned} \text{assuming } a &= A_3 \times 2^3 + A_2 \times 2^2 + A_1 \times 2^1 + A_0 \times 2^0 \\ \text{and } b &= B_3 \times 2^3 + B_2 \times 2^2 + B_1 \times 2^1 + B_0 \times 2^0 \end{aligned}$$

```
if a == b then x := 1
if a != b then x := 0
```



Example: n -bit addition

$$\begin{array}{r} \text{carry} \\ a \quad \dots \quad 0 \quad 1 \quad 1 \quad 0 \\ b \quad \dots \quad 0 \quad 1 \quad 1 \quad 1 \\ \hline a + b \end{array}$$

Example: n -bit addition

carry				0	
a	...	0	1	1	0
b	...	0	1	1	1
<hr/>					
a + b					1

Example: n -bit addition

carry			1	0	
a	...	0	1	1	0
b	...	0	1	1	1
<hr/>					
a + b				0	1

Example: n -bit addition

carry		1	1	0	
a	...	0	1	1	0
b	...	0	1	1	1
<hr/>					
a + b			1	0	1

Example: n -bit addition

carry	0	1	1	0	
a	...	0	1	1	0
b	...	0	1	1	1
<hr/>					
a + b	...	1	1	0	1

Example: n -bit addition

carry	0	1	1	0	
a	...	0	1	1	0
b	...	0	1	1	1
<hr/>					
a + b	...	1	1	0	1

The 1-bit “full adder”

- Input:

- 1 bit of carry: C_{in}
- 1 bit of a: A
- 1 bit of b: B

- Output:

- 1 bit of carry: C_{out}
- 1 bit of the sum $a + b$: S

- Operation: Compute C_{out} and S such that

$$A + B + C_{in} = C_{out} \times 2 + S$$

carry	...	1	1	0	
a	...	0	1	1	0
b	...	0	1	1	1
<hr/>					
a + b	...	1	1	0	1

The 1-bit “full adder”

- Input:

- 1 bit of carry: C_{in}
- 1 bit of a: A
- 1 bit of b: B

- Output:

- 1 bit of carry: C_{out}
- 1 bit of the sum $a + b$: S

- Operation: Compute C_{out} and S such that

$$A + B + C_{in} = C_{out} \times 2 + S$$

Truth table:

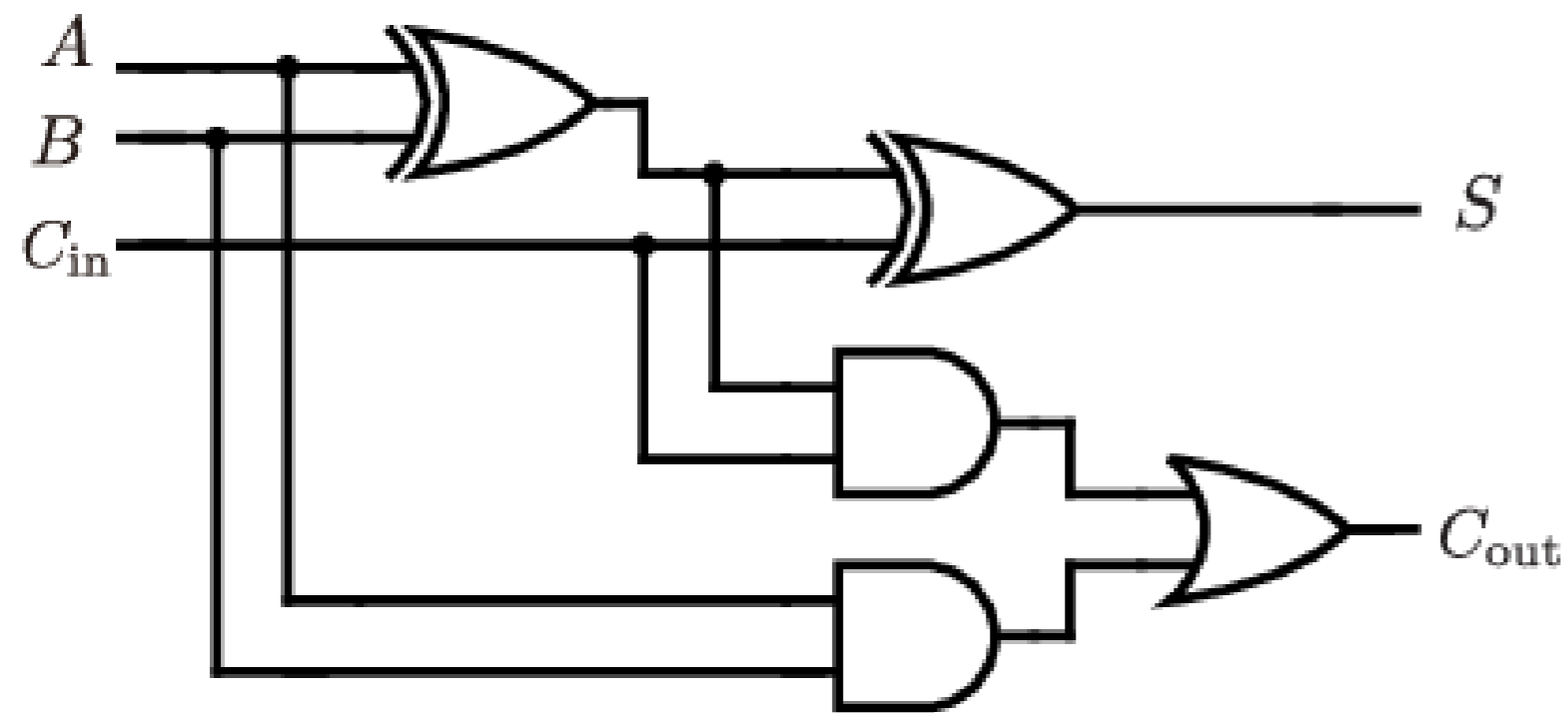
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The 1-bit “full adder”

Boolean expressions:

- $S := A \text{ xor } B \text{ xor } C_{in}$
- $C_{out} := (A \text{ and } B) \text{ or } ((A \text{ xor } B) \text{ and } C_{in})$

Logic diagram:



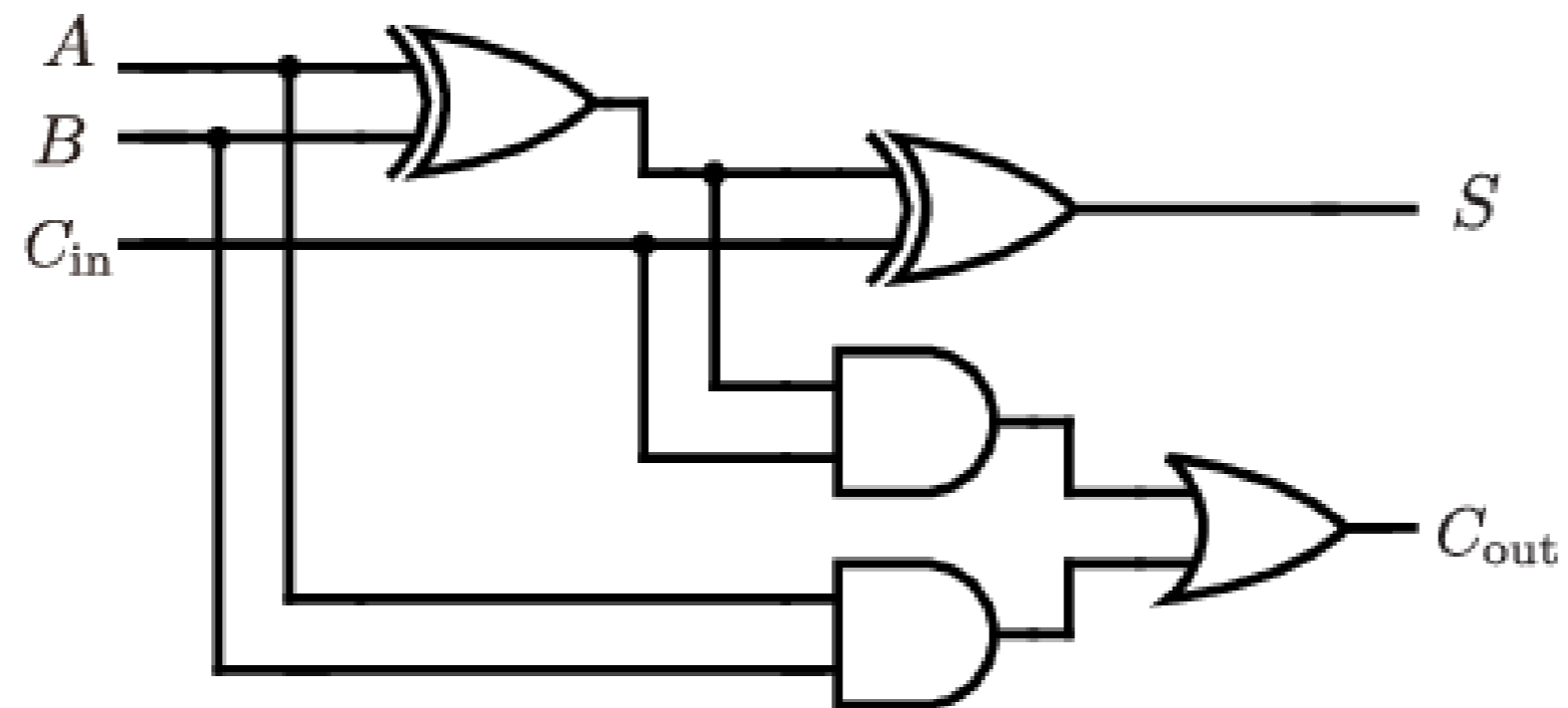
Truth table:

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

The 1-bit “full adder”

Notes:

- Called “full” in contrast to the “1-bit half adder” which has no C_{in} .
- There can be multiple valid Boolean expressions (and logic diagrams)



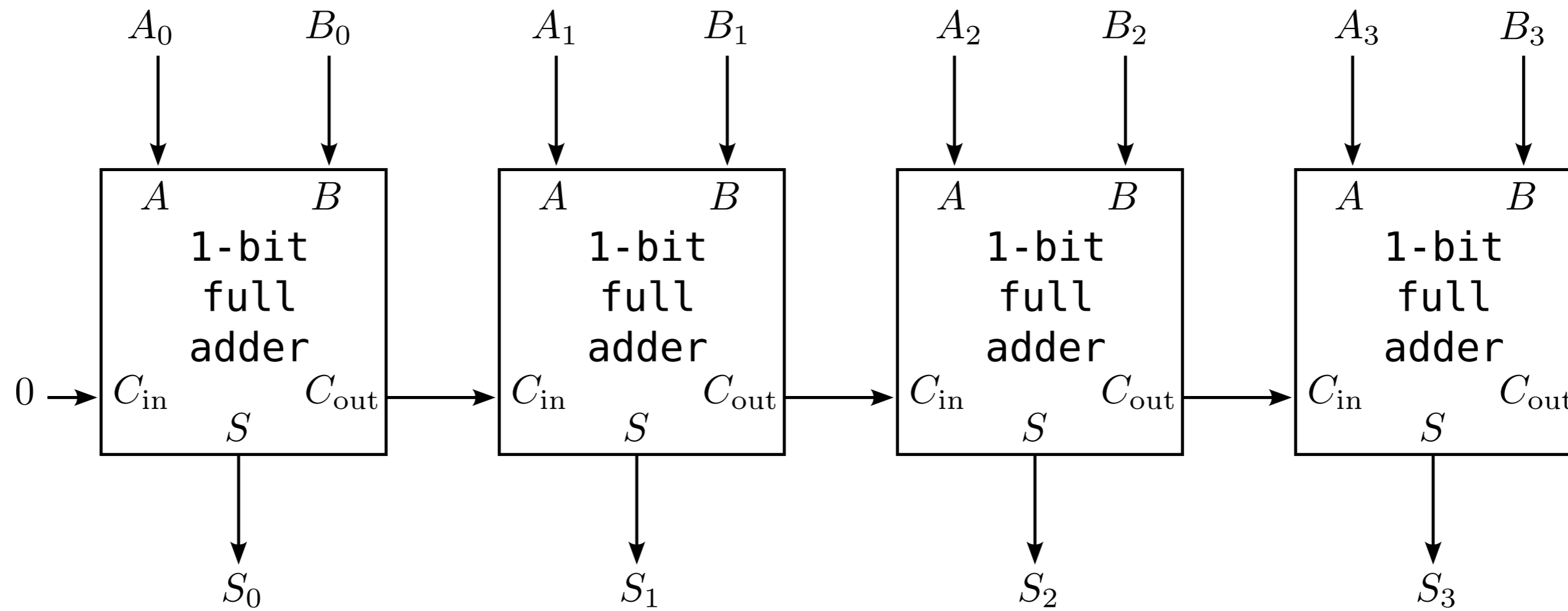
The 1-bit “full adder”

Notes:

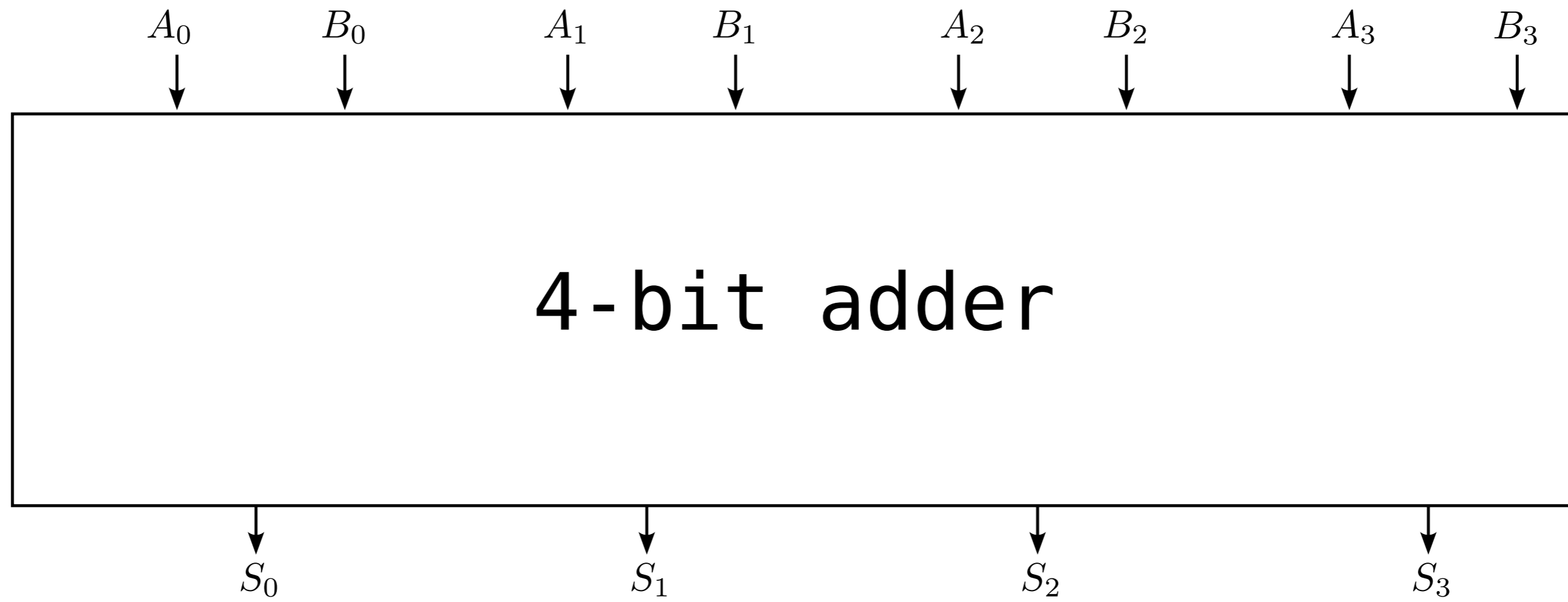
- Called “full” in contrast to the “1-bit half adder” which has no C_{in} .
- There can be multiple valid Boolean expressions (and logic diagrams)



4-bit adder



4-bit adder



How IP blocks are designed

IP blocks are designed (and combined) in hardware description languages (“HDL”):

Verilog, VHDL and derivatives

HDL is then translated into IC layouts by specialized tools.

```
interface adder_if();
    logic      rstn;
    logic [7:0] a;
    logic [7:0] b;
    logic [7:0] out;
    logic      carry;
endinterface

module adder(adder_if i);
    always_comb begin
        if (i.rstn) begin
            i.out <= 0;
            i.carry <= 0;
        end else begin
            {i.carry, i.out} <= i.a + i.b;
        end
    end
end
endmodule
```

SystemVerilog code for an 8-bit adder

The IC design and manufacturing industry

Foundries

- “**foundry**” or “fabrication plant” (“**fab**”): plant in which ICs are manufactured, and by extension, companies who own such plants (e.g. Intel, TSMC, Samsung).
- “**process node**”: marketing name given by foundries to a particular version of their manufacturing process – usually a “**feature size**”: the size of some parts of the transistors (e.g. 5nm, 7nm, etc.) – but not directly comparable across companies.

Smaller transistors means:

- more transistors (per unit of area)
- lower power consumption
- faster ICs (propagation delay↓, power consumption↓, heat dissipation↓)

Types of industry players

- **fabless**: company that does not own fabrication plants (e.g. ARM, AMD, Apple, nVidia).
Such companies either:
 - sell the designs of their IP blocks (ARM), or
 - subcontract foundries to manufacture their designs for them (AMD, Apple, nVidia).
- **“pure-play” foundries** (e.g. TSMC): foundries who manufacture other companies’ designs.
- **integrated device manufacturer (IDM)** (e.g. Intel): designs and manufactures its own ICs.

Recent history of the industry

- In the early 2000s, **Intel** (US), **AMD** (US) and **IBM** (US) have the best manufacturing technology. All three are IDMs – they design and manufacture in-house.
- Around 2008, **AMD** (US) spun off its foundry (as “**GlobalFoundries**”) and became fabless.
Note: first iPhone released in 2007 in the US
- In the 2010s, **TSMC** (Taiwan) emerges as a major foundry for **Apple** (US), **nVidia** (US) and **AMD**.
- In 2014, **IBM** sells its manufacturing business to **GlobalFoundries**.
- As of 2025, **GlobalFoundries** is still one of the largest pure-play foundries but it has fallen behind in terms of technology (by 5-10 years).

State of the industry

- Since ~2018, **TSMC** (Taiwan) has had the best process node, ahead of **Samsung** (Korea) and **Intel** (US).
- **TSMC**'s advantage in large part due to early bet on extreme ultra-violet (“EUV”) technology. Now, **Samsung** and **Intel** also use EUV tech. **ASML** (Netherlands) is currently the only supplier of EUV-capable machines.
- **Apple** (US), **AMD** (US), **nVidia** (US), **Qualcomm** (US) all mostly subcontract **TSMC** to fabricate their top-of-the-line ICs.