

# Binary Adder & Boolean Logic

A conversation between Andrew and Claude — May 2026

## 1. What is Boolean Logic?

**Andrew:** What's Boolean?

**Claude:** Boolean — named after George Boole, a 19th century English mathematician. He developed a system of algebra where values are either TRUE or FALSE (1 or 0) and operations are AND, OR, NOT. Published in 1854 in 'The Laws of Thought.'

The genius is that it maps perfectly onto switches:

- Switch ON = 1 = TRUE
- Switch OFF = 0 = FALSE

So any physical switching mechanism — relay, mechanical plate, transistor — can implement Boolean logic.

## 2. XOR Gate — The Adder

XOR (Exclusive OR): the output is ON only when an **odd number** of inputs are ON. Think of a staircase switch — either switch toggles the light, but not both together.

**XOR Truth Table (A XOR B XOR Cin = Sum):**

A	B	Cin	Sum (XOR)	Notes
0	0	0	0	Zero inputs ON
1	0	0	1	One input ON (odd)
0	1	0	1	One input ON (odd)
0	0	1	1	One input ON (odd)
1	1	0	0	Two inputs ON (even)
1	0	1	0	Two inputs ON (even)
0	1	1	0	Two inputs ON (even)
1	1	1	1	Three inputs ON (odd)

## 3. AND Gate — The Carry

AND = two switches in **series**. Both must be ON for current to flow. In binary addition, AND detects overflow — when both A and B are 1, we need to carry.

A	B	Carry (AND)
0	0	0
0	1	0
1	0	0
1	1	1 ← carry!

## 4. Half Adder = XOR + AND

A half-adder adds two single bits (A and B) with no carry-in:

- **XOR** → gives the Sum
- **AND** → gives the Carry-out

Example:  $1 + 1 = 0$ , carry 1 (binary:  $10 = \text{decimal } 2$ )

A	B	Sum (XOR)	Carry (AND)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

## 5. Full Adder = Two Half Adders + OR

A full adder handles three inputs: A, B, and Carry-in (Cin).

**The two half-adders are connected in series** — the Sum output of half-adder 1 feeds directly into the input of half-adder 2. This is the data path.

### Structure:

- Half-adder 1:  $A + B \rightarrow \text{Sum1, Carry1}$
- Half-adder 2:  $\text{Sum1} + \text{Cin} \rightarrow \text{Final Sum, Carry2}$
- OR gate:  $\text{Carry1 OR Carry2} \rightarrow \text{Final Carry-out}$

**Key insight:** The two half-adders are in **series** (data path). Their carries meet in **parallel** through the OR gate. Carry1 and Carry2 are mutually exclusive — only one can ever be 1 at a time. The OR simply collects whichever carry exists.

**Full adder = 2× XOR + 2× AND + 1× OR**

## 6. The Complete Stack

Level	Component	Description
1	Switch	Stores 1 bit — ON or OFF. No processing.
2	AND / OR / XOR	Logic gates — relationships between switches.
3	Half-adder	XOR + AND. Adds 2 bits.
4	Full-adder	2× half-adder + OR. Adds 3 bits (with carry-in).
5	Ripple-carry chain	Full-adders in series. Adds any number of bits.
6	64-bit processor	64 full-adders. Runs SMF, Claude, everything.

## 7. Key Insights from the Conversation

## **XOR is a staircase switch**

A domestic staircase light switch IS an XOR gate — either switch toggles the light, but not both together. You've used XOR every day without knowing it.

## **Processing requires relationships**

A single switch stores 1 bit but cannot process. Processing begins the moment two switches are wired together — their combined state produces a new output.

## **Series and parallel are AND and OR**

AND = series (both must close). OR = parallel (either closes). XOR = cross-wired series/parallel (staircase wiring).

## **The full adder is series-parallel**

Data path is series (half-adder 1 feeds half-adder 2). Carry path is parallel (both carries meet at OR gate).

## **Same logic, different substrate**

Zuse's metal plates (1936) → telephone relays (Z3) → transistors → your i7 NUC. Same Boolean equations throughout. Just faster switching each time.

— Conversation captured May 2026 —