Digital Logic

Dov Kruger

Department of Electrical and Computer Engineering
Rutgers University

September 15, 2025

- Digital signals: 0 and 1
- Basis for all digital electronics
- Implemented using logic gates
- Combines signals to perform tasks

Why Digital?

- With an analog signal, difficult to distinguish signal and error
- With digital, a little noise can be ignored
- Digital signals are easier to store and process using computers
- Error detection and correction are more feasible with digital signals
- Digital systems are more scalable and compatible with modern technology

Combinatorial Logic Overview

- Output depends on input values
- No memory elements
- Built from basic gates
- Used in arithmetic and control circuits

Noise in Digital Signals

- Digital signals have noise
- Noise doesn't affect logic as long as it doesn't flip a bit
- Signal integrity depends on thresholds

Ringing and State in Digital Signals

- Digital is an abstraction
- Real-life signals have rise/fall time
- Ringing occurs after transitions
- Signal settles into a stable state

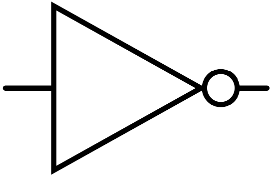
Trend in Voltage Levels

- Earliest digital: vaccum tubes (100V+)
- Early microprocessors: 5V
- Modern processors: 1.1V - 1.3V
- Power reduction with increased circuit density

Historical Voltage Levels

Computer	Year	Tech	Voltage	Size	Number of Devices	Power
ENIAC	1945	Tubes	150V - 300V	$5 - 10cm$	$18,000$ tubes	150kW
IBM 7090	1959	Transistors	6V - 12V	$1 - 2cm$	$5,200$	50kW
Cray-1	1976	IC	15V	$5\mu m$	$10,000$ ICs	115kW
Intel 4004	1971	PMOS	15V	$10\mu m$	$2,300$	1W
Intel 8086	1978	NMOS	5V	$3\mu m$	$29,000$	2W
Pentium	1993	CMOS	3.3V	$800nm$	$3.1x10^6$	10W
Core i7-920	2008	CMOS	1.1V - 1.2V	$45nm$	$731x10^6$	95W
Core i7-7700K	2017	CMOS	1.2V	$14nm$	$1.2x10^9$	91W
Core i9-9900K	2020	CMOS	1.2V	$14nm$	$19x10^9$	125W
Apple M1	2020	CMOS	0.8V - 1.1V	$5nm$	$16x10^9$	15W
AMD Ryzen 9	2023	CMOS	0.8V - 1.1V	$3nm$	$16x10^9$	170W
Apple M4	2024	CMOS	0.7V - 1.0V	$3nm$	$25x10^9$	12W

NOT Gate



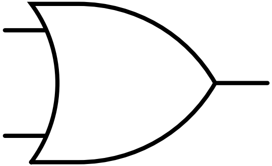
Boolean Algebra: $\overline{A}$

A	$\overline{A}$
0	1
1	0

It's hard to type the overline so there are other notations

$\overline{A} = A' = \tilde{}A$

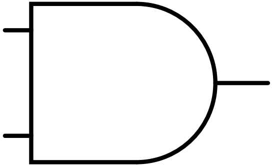
OR Gate



Boolean Algebra: $A + B$

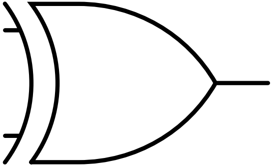
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

AND Gate



Boolean Algebra: $A \cdot B$

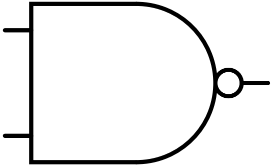
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1



Boolean Algebra: $A \oplus B$

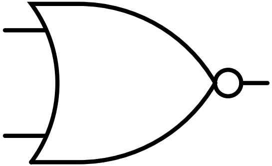
A	B	A $\oplus$ B
0	0	0
0	1	1
1	0	1
1	1	0

NAND Gate



Boolean Algebra: $\overline{A \cdot B}$

A	**B**	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

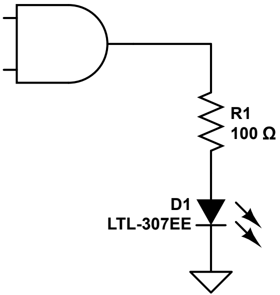


Boolean Algebra: $\overline{A + B}$

A	**B**	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

74LS Gate Sourcing Current

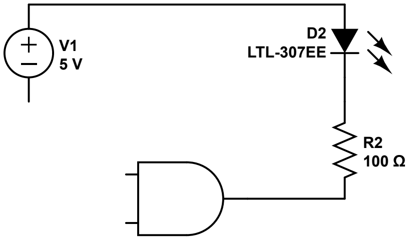
- 74LS gate can only source 0.4 mA
- Insufficient current for bright LED operation
- LED will barely glow due to limited current



Schematic showing 74LS gate sourcing an LED

74LS Gate Sinking Current

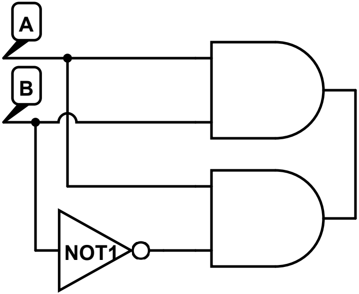
- 74LS gate can sink up to 16 mA
- Sufficient current to drive an LED partially



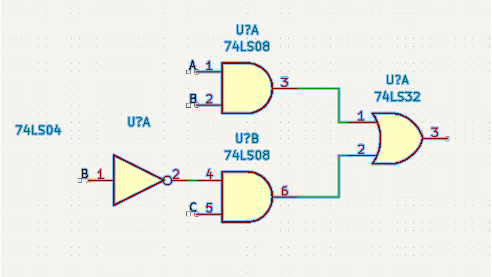
Schematic showing 74LS gate sinking an LED

Tying Outputs Together

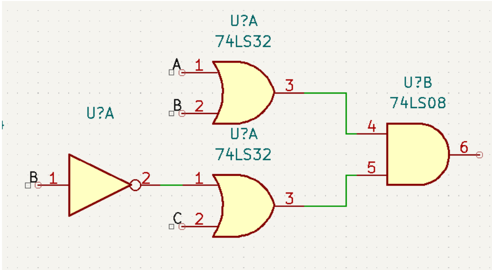
- You cannot wire together the outputs of two gates
- If the outputs are different values, it will create a short circuit
- This can damage the gates or other components



Sum of Products $F = AB + \overline{B}C$



Product of Sums $F = (A + B)(\overline{B} + C)$



Boolean Expression: $A + BC + A'C'$

Truth Table (Empty)

A	B	C	Output
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Boolean Expression Solution: $A + BC + A'C'$

Truth Table (Filled)

A	B	C	Output
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Verilog Bitwise Operations

- & : AND
- | : OR
- ^ : XOR
- ~ : Bitwise NOT

```
module bitwise_operations;
    reg a = 1;
    reg b = 0;

    reg r0 = a & b;
    reg r1 = a | b;
    reg r2 = a ^ b;
    reg r3 = ~a;
endmodule
```

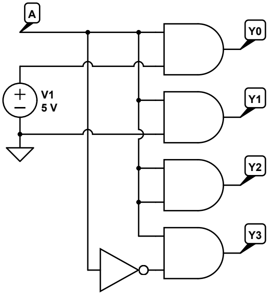
Verilog Multiple Bits

```
module bitwise_operations;
    reg [3:0] a = 4'b1010;
    reg [3:0] b = 4'b1100;

    reg [3:0] and_result = a & b;
    reg [3:0] or_result = a | b;
    reg [3:0] xor_result = a ^ b;
    reg [3:0] not_result = ~a;
endmodule
```

AND Gate Identities

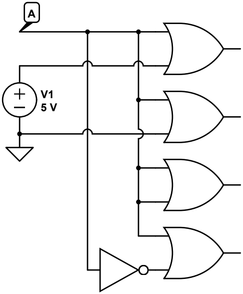
$$A \cdot 1 = A \quad A \cdot 0 = 0 \quad A \cdot A = A \quad A \cdot \overline{A} = 0$$



Schematic for AND Gate Identities

OR Gate Identities

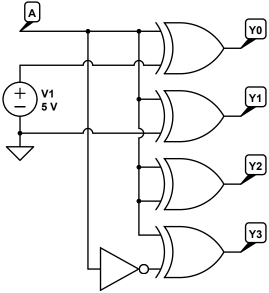
$$A + 1 = 1 \quad A + 0 = A \quad A + A = A \quad A + \overline{A} = 1$$



Schematic for OR Gate Identities

XOR Gate Identities

$$A \oplus 1 = \overline{A} \quad A \oplus 0 = A \quad A \oplus A = 0 \quad A \oplus \overline{A} = 1$$



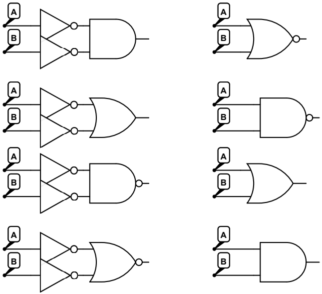
Schematic for XOR Gate Identities

- **DeMorgan's First Law**:

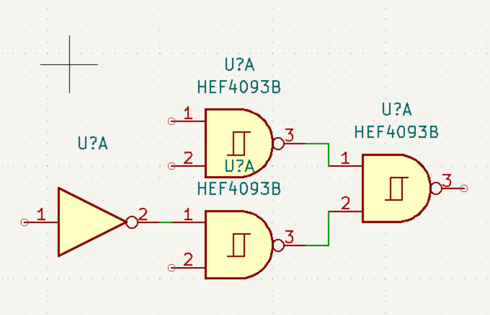
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

- **DeMorgan's Second Law**:

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

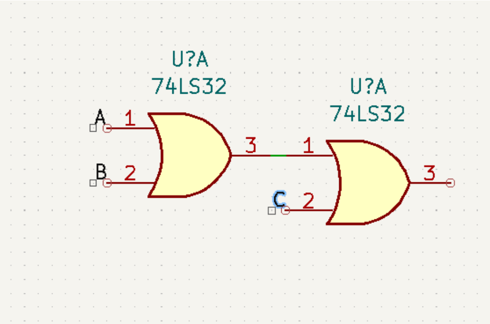


Any logic circuit can be implemented using NAND



Constructing a 3-input OR gate

A 3-input gate can be constructed from 2-input gates



A 3-input gate can be constructed from 2-input gates