The basics of logic design: Combinational logic

Yipeng Huang

Rutgers University

December 2, 2025

Table of contents

Computer organization

Layer cake

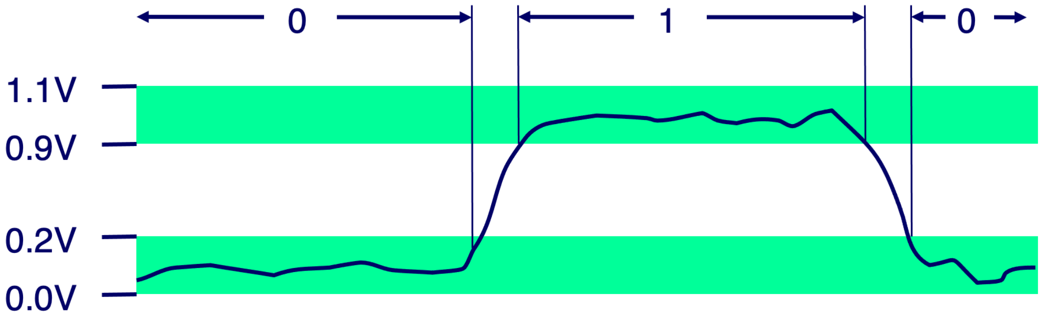
- Society
- Human beings
- Applications
- Algorithms
- High-level programming languages
- Interpreters
- Low-level programming languages
- Compilers
- Architectures
- Microarchitectures
- Sequential/combinational logic
- Transistors
- Semiconductors
- Materials science

logic: { combinational → no state
sequential → stateful.

Why binary

Everything is bits

- **Each bit is 0 or 1**
- **By encoding/interpreting sets of bits in various ways**
  - Computers determine what to do (instructions)
  - … and represent and manipulate numbers, sets, strings, etc…
- **Why bits? Electronic Implementation**
  - Easy to store with bistable elements
  - Reliably transmitted on noisy and inaccurate wires

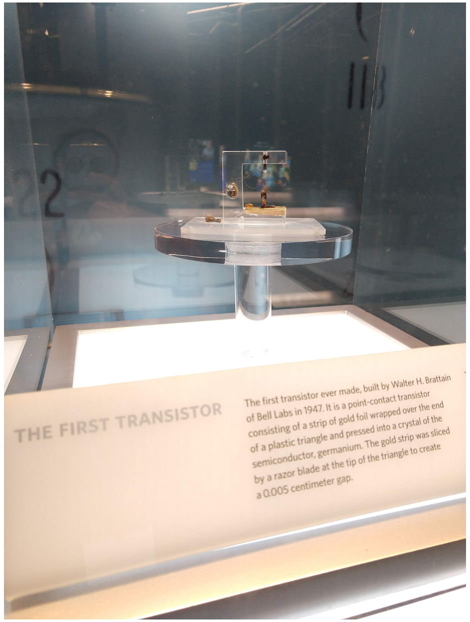


To build logic, we need switches

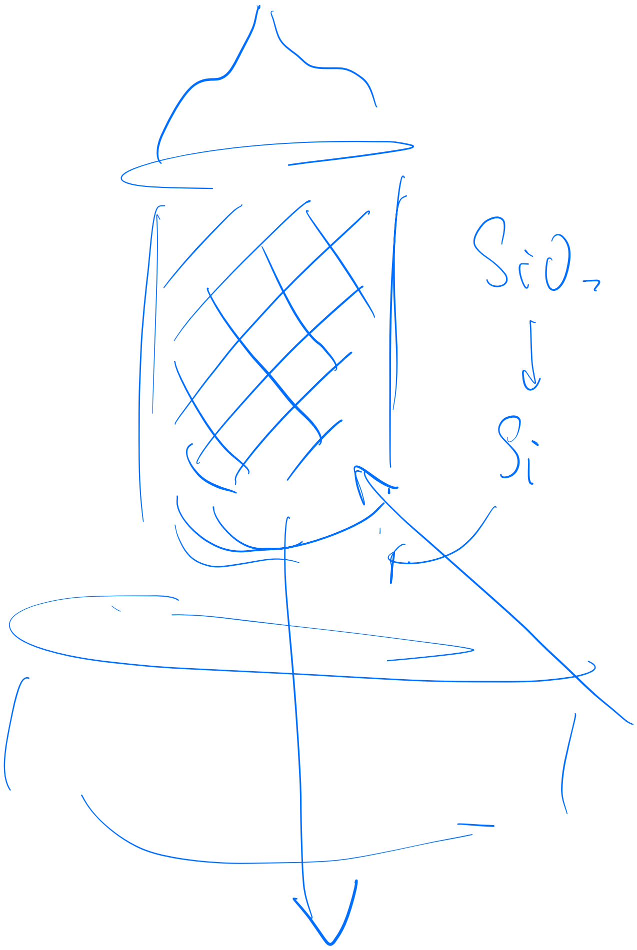
Vacuum tubes a.k.a. valves

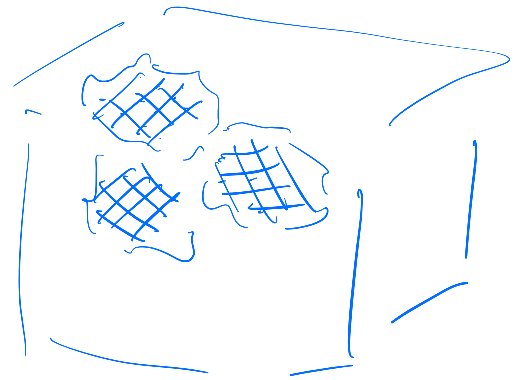
Figure: Source: By Stefan Riepl (Quark48) - Self-photographed, CC BY-SA 2.0 `https://commons.wikimedia.org/w/index.php?curid=14682022`

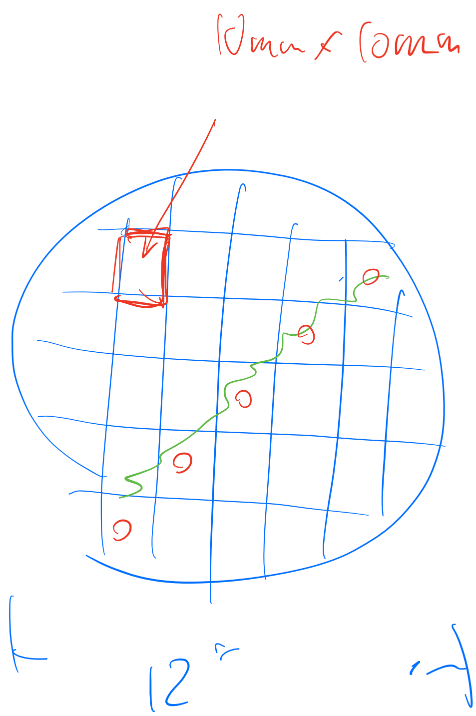
Transistors

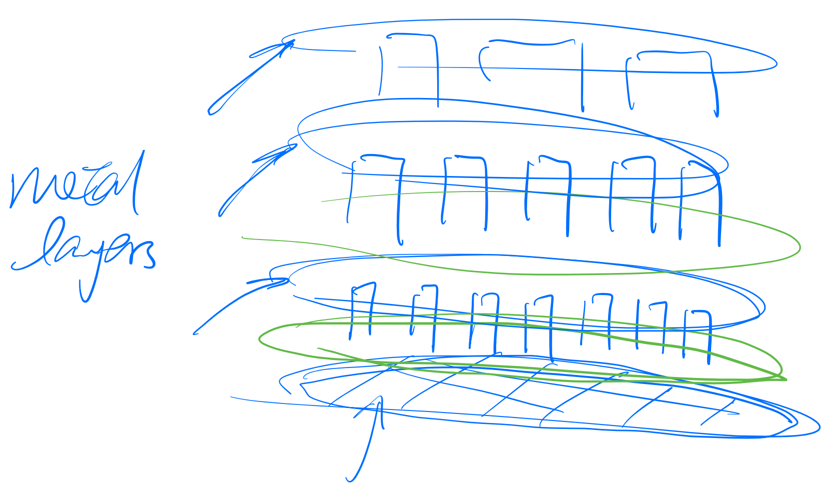


- The first transistor. Developed at Bell Labs, Murray Hill, New Jeresy
- `https://www.bell-labs.com/about/locations/`


SiO2
Si




10mm x 10mm
12"


metal layers

MOSFETs

diodes. Bipolar Junction transistors,

FETs.

MOS: Metal-oxide-semiconductor

- A sandwich of conductor-insulator-semiconductor.

FET: Field-effect transistor

- Gate exerts electric field that changes conductivity of semiconductor.

B C N

NMOS, PMOS, CMOS

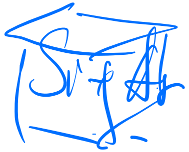
Al Si P

Ga Ge. As

PMOS: P-type MOS

- positive gate voltage, acts as open circuit (insulator)
- negative gate voltage, acts as short circuit (conductor)

NMOS: N-type MOS



- positive gate voltage, acts as short circuit (conductor)
- negative gate voltage, acts as open circuit (insulator)

CMOS: Complementary MOS

- A combination of NMOS and PMOS to build logical gates such as NOT, AND, OR.
- We'll go to slides posted in supplementary material to see how they work.

Combinational vs. sequential logic

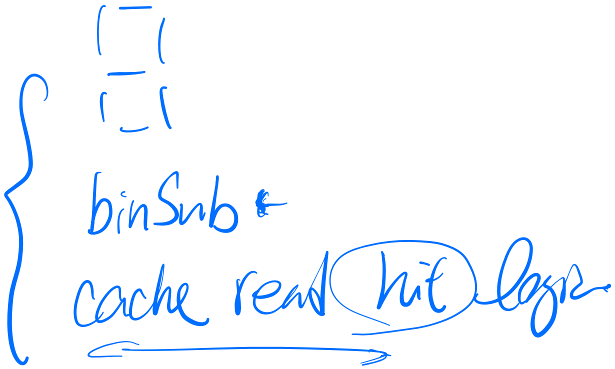
Combinational logic

- No internal state nor memory
- Output depends entirely on input
- Examples: NOT, AND, NAND, OR, NOR, XOR, XNOR gates, decoders, multiplexers.

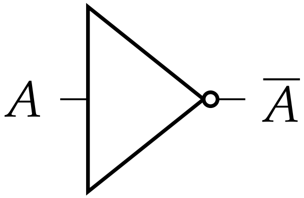
Sequential logic

- Has internal state (memory)
- Output depends on the inputs and also internal state
- Examples: latches, flip-flops, Mealy and Moore machines, registers, pipelines, SRAMs.

Table of contents



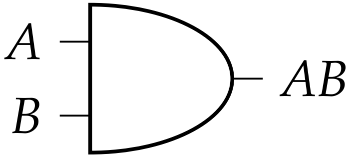
NOT gate



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

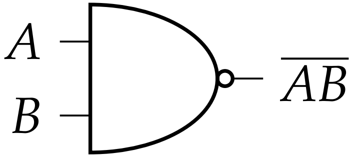
Table: Truth table for NOT gate

AND gate, NAND gate



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

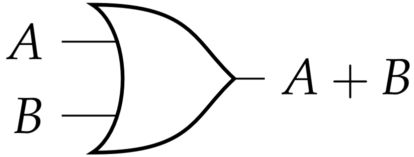
Table: Truth table for AND gate



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

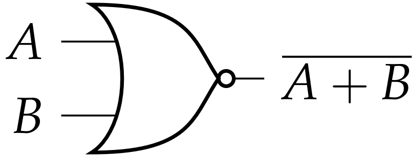
Table: Truth table for NAND gate

OR gate, NOR gate



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

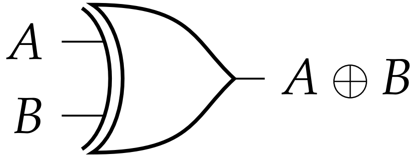
Table: Truth table for OR gate



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

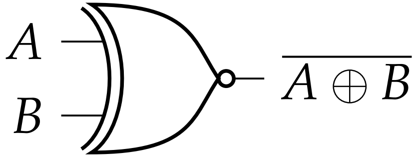
Table: Truth table for NOR gate

XOR gate, XNOR gate



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

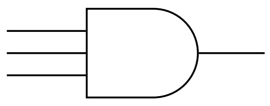
Table: Truth table for XOR gate

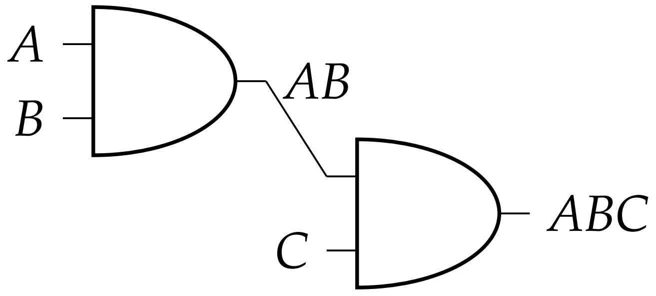


A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Table: Truth table for XNOR gate

More-than-2-input AND gate

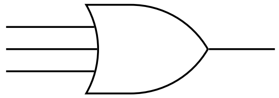


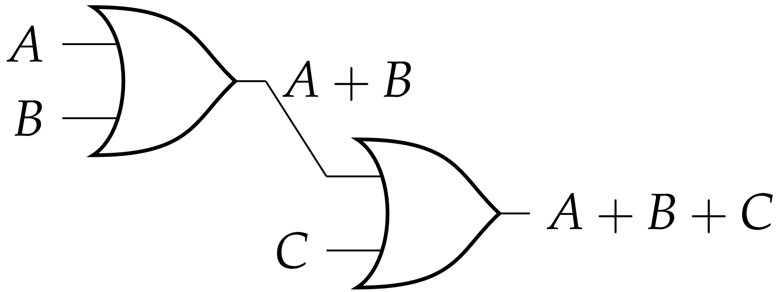


A	B	C	ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table: Truth table for three-input AND gate

More-than-2-input OR gate





A	B	C	$A+B+C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Table: Truth table for three-input OR gate

Table of contents

The set of logic gates {NOT, AND, OR} is universal

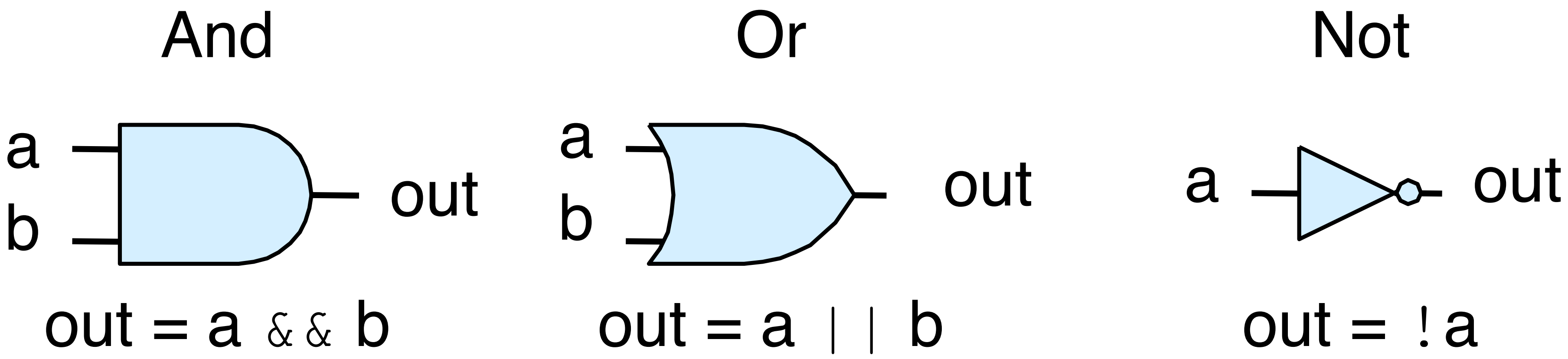


Figure: Source: CS:APP

Or-of ands
Sum-of-products

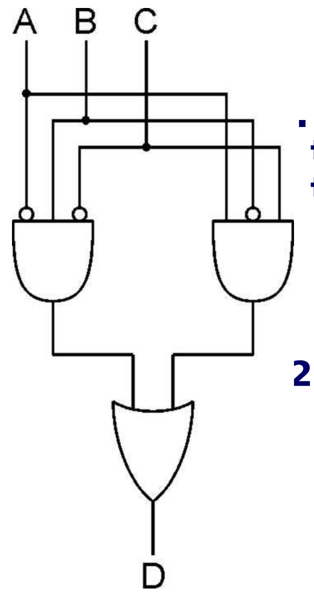
The set of logic gates {NOT, AND, OR} is universal

- Any truth table can be expressed as sum of products form.
- Write each row with output 1 as a product (minterm).
- Sum the products (minterm).
- Forms a disjunctive normal form (DNF).
- $D = \overline{A}B\overline{C} + A\overline{B}C$
- Always only needs NOT, AND, OR gates.
- Supplementary slides example...

Logical Completeness

Can implement ANY truth table with AND, OR, NOT.

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



. AND combinations that yield a "1" in the truth table.

2. OR the results of the AND gates.

Sum of products
OR of AND clauses

The set of logic gates {NOT, AND, OR} is universal

- Any truth table can be expressed as sum of products form.
- Write each row with output 1 as a product (minterm).
- Sum the products (minterm).
- Forms a disjunctive normal form (DNF).
- $D = \overline{A}B\overline{C} + A\overline{B}C$
- Always only needs NOT, AND, OR gates.
- Supplementary slides example...

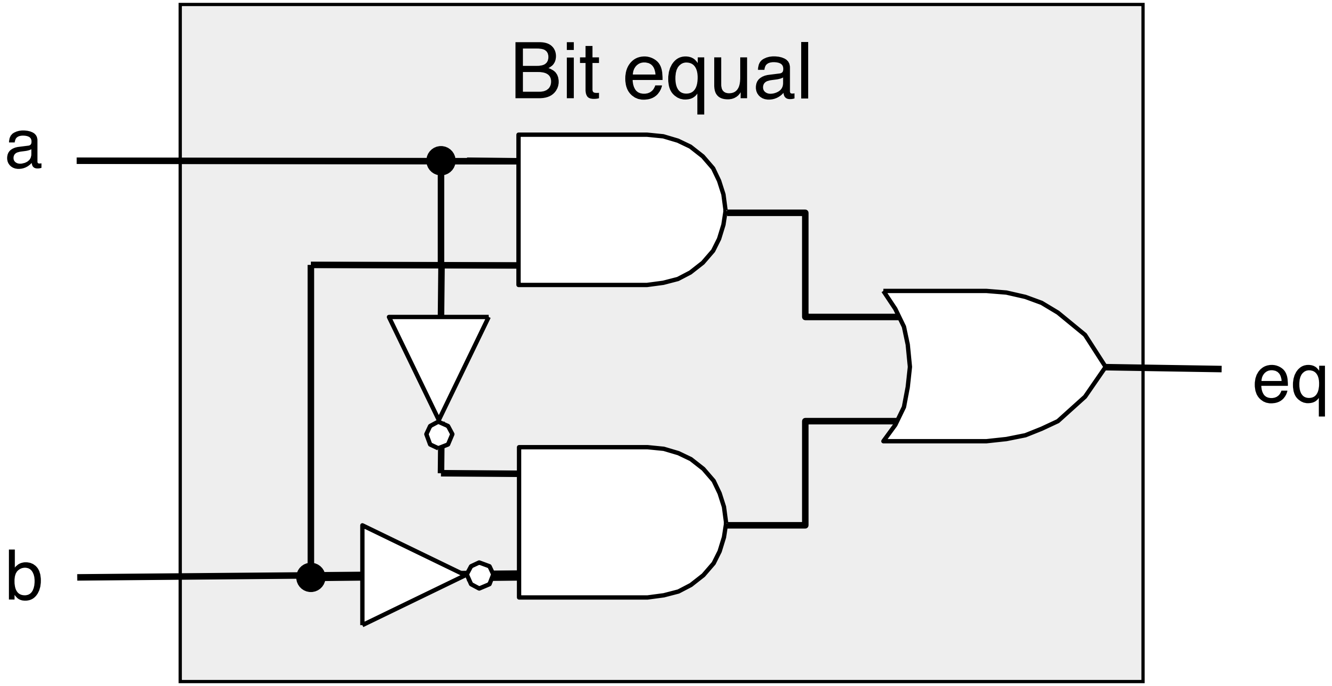
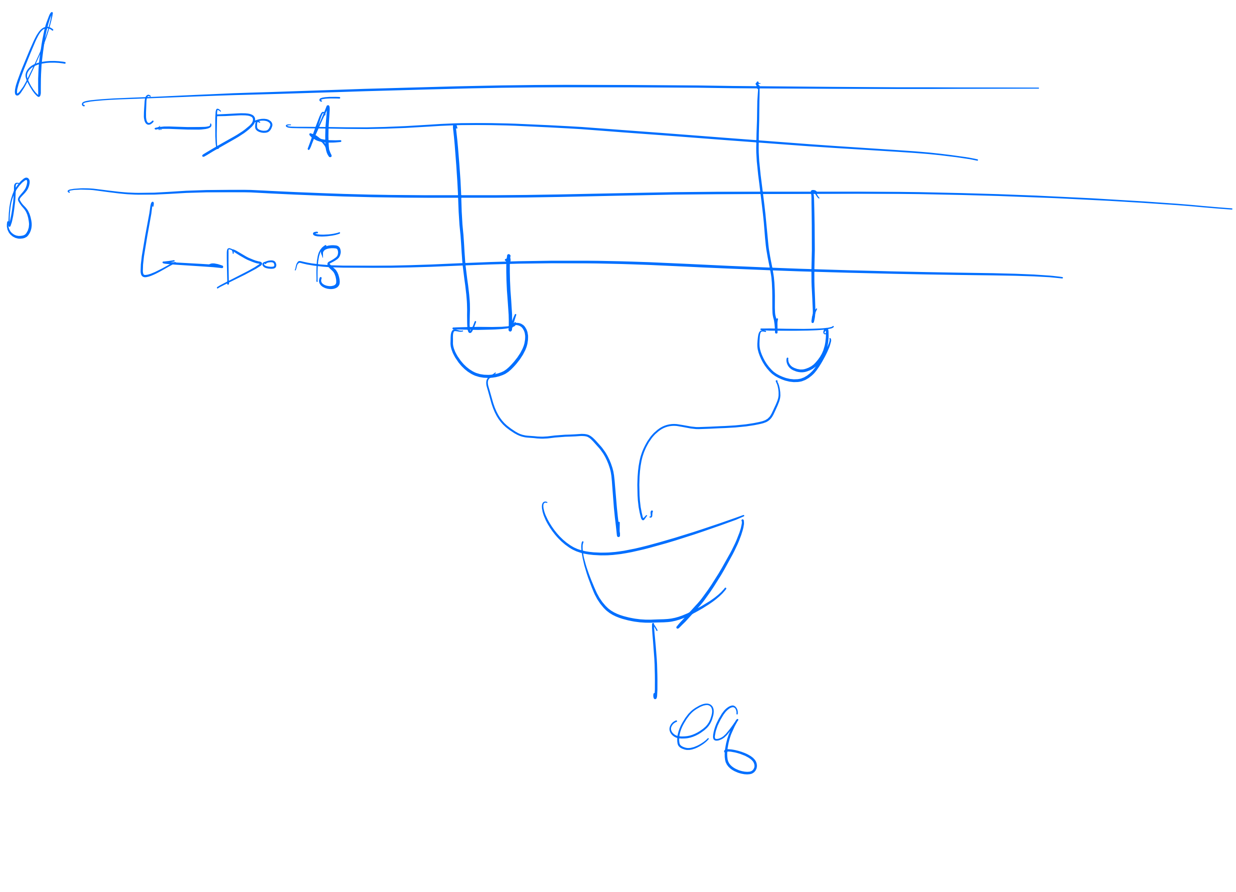


Figure: Source: CS:APP

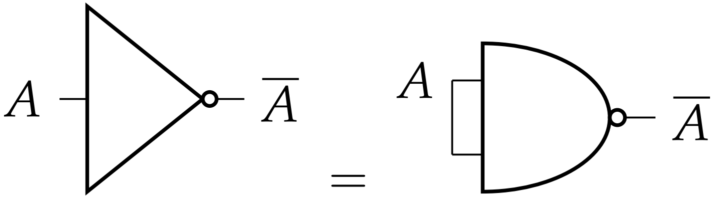
a	b	eq
0	0	1
0	1	0
1	0	0
1	1	1

$$eq = \bar{A}\,\bar{B} + AB$$



The NAND gate is universal

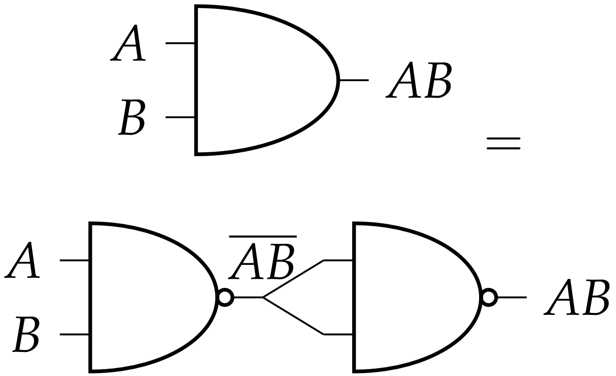
NOT gate as a single NAND gate



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

Table: $\overline{A} = \overline{AA}$

AND gate as two NAND gates



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

Table: $AB = \overline{\overline{AB}}$

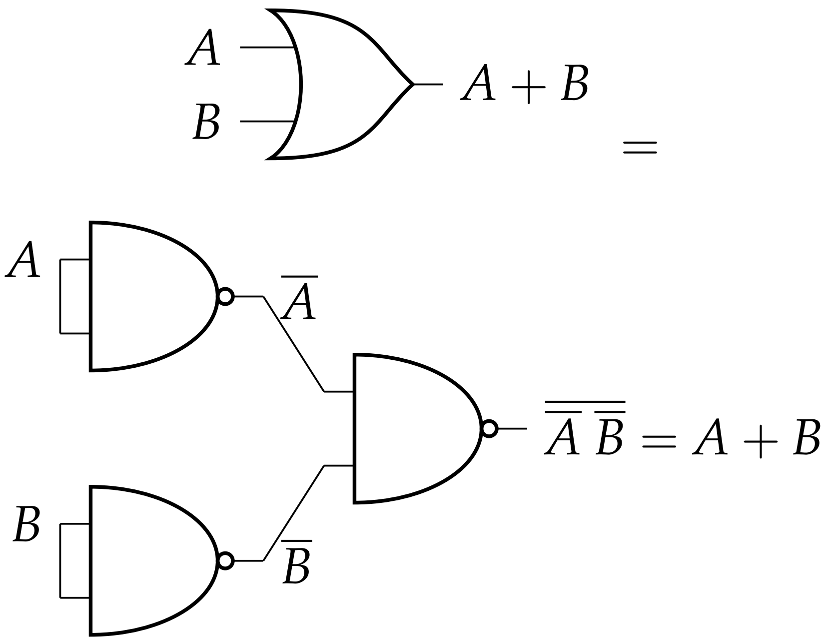
The NAND gate is universal

De Morgan's Law

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

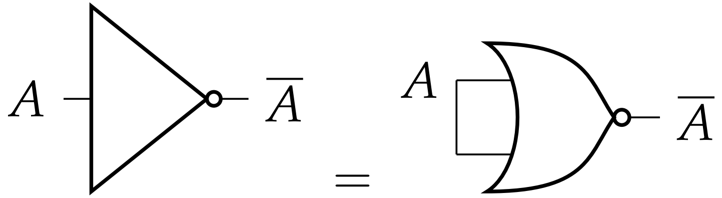
Table: $\overline{A}\,\overline{B} = \overline{A+B}$

OR gate as three NAND gates



The NOR gate is universal

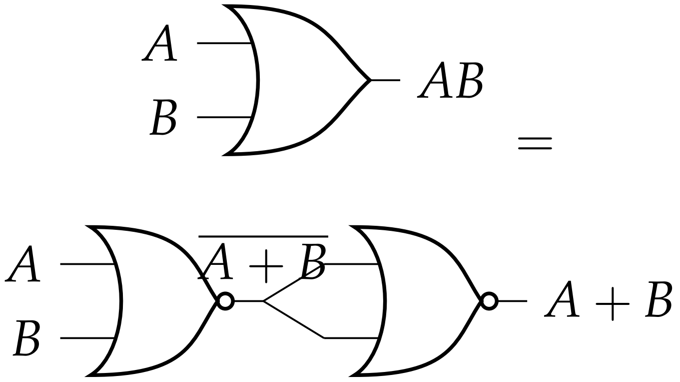
NOT gate as a single NOR gate



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

Table: $\overline{A} = \overline{A+A}$

OR gate as two NOR gates



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

Table: $A+B = \overline{\overline{A+B}}$

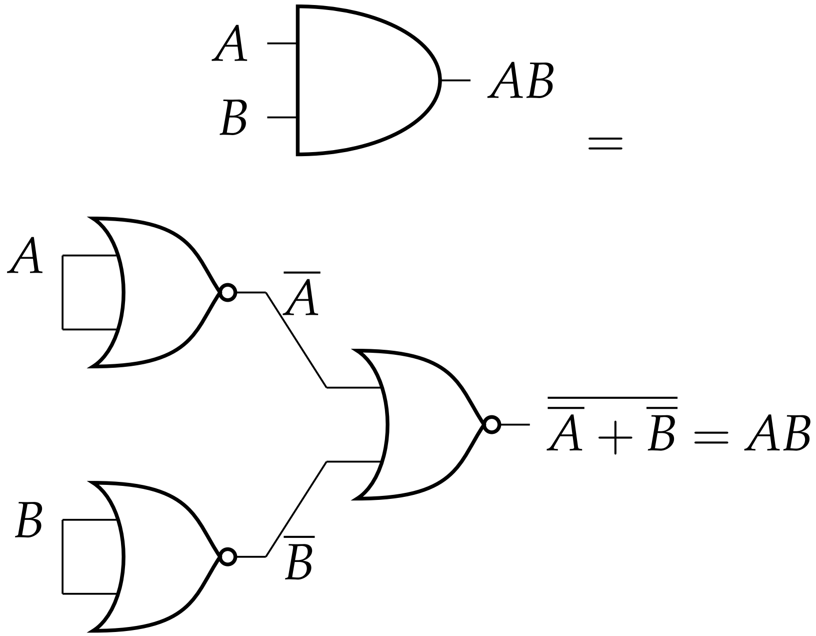
The NOR gate is universal

De Morgan's Law

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

Table: $\overline{A}+\overline{B}=\overline{AB}$

AND gate as three NOR gates



Combinational vs. sequential logic

Combinational logic

- No internal state nor memory
- Output depends entirely on input
- Examples: NOT, AND, NAND, OR, NOR, XOR, XNOR gates, decoders, multiplexers.

Sequential logic

- Has internal state (memory)
- Output depends on the inputs and also internal state
- Examples: latches, flip-flops, Mealy and Moore machines, registers, pipelines, SRAMs.

Table of contents

Decoders

Takes n-bit input, uses it as an index to enable exactly one of 2^n outputs

Internal design of 1:2 decoder

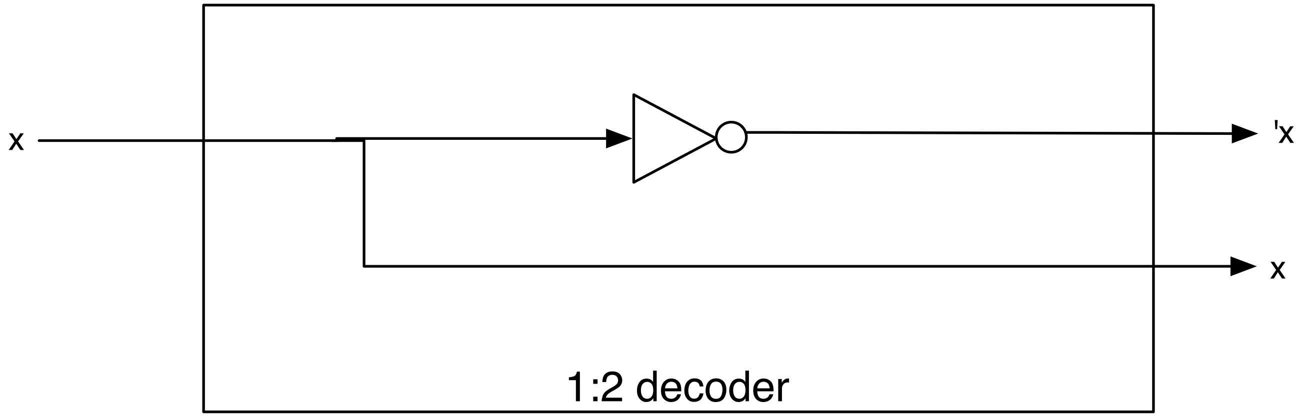


Figure: Source: Mano & Kime

Decoders

Hierarchical design of decoder (2:4 decoder)

Takes n-bit input, uses it as an index to enable exactly one of 2^n outputs

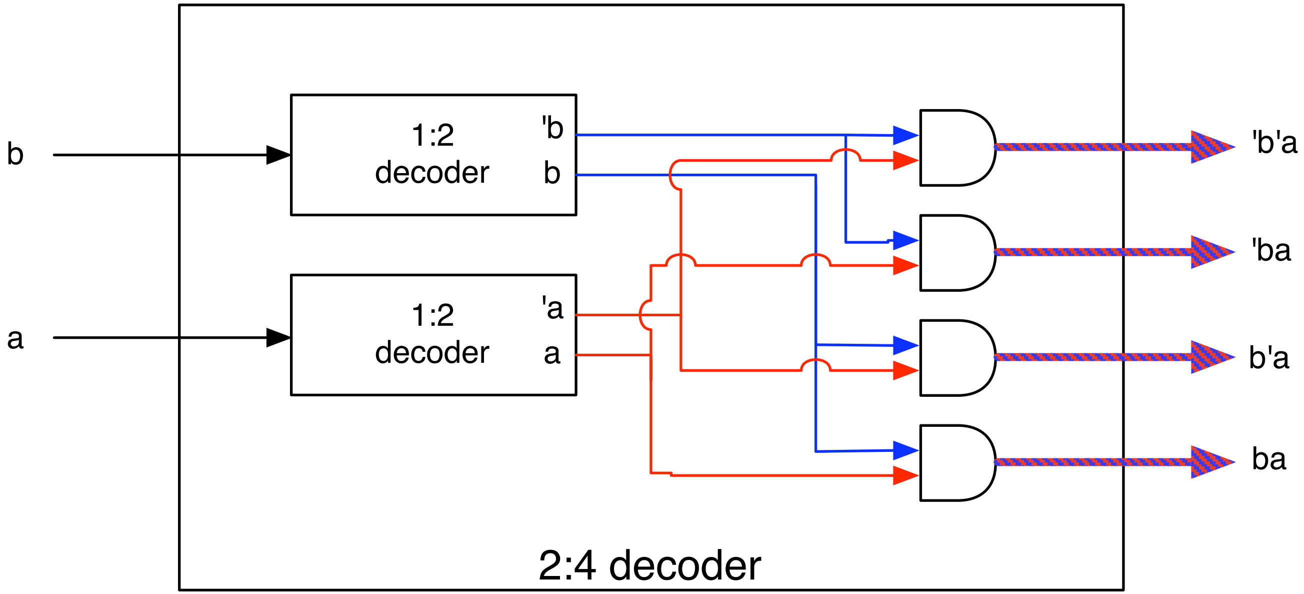
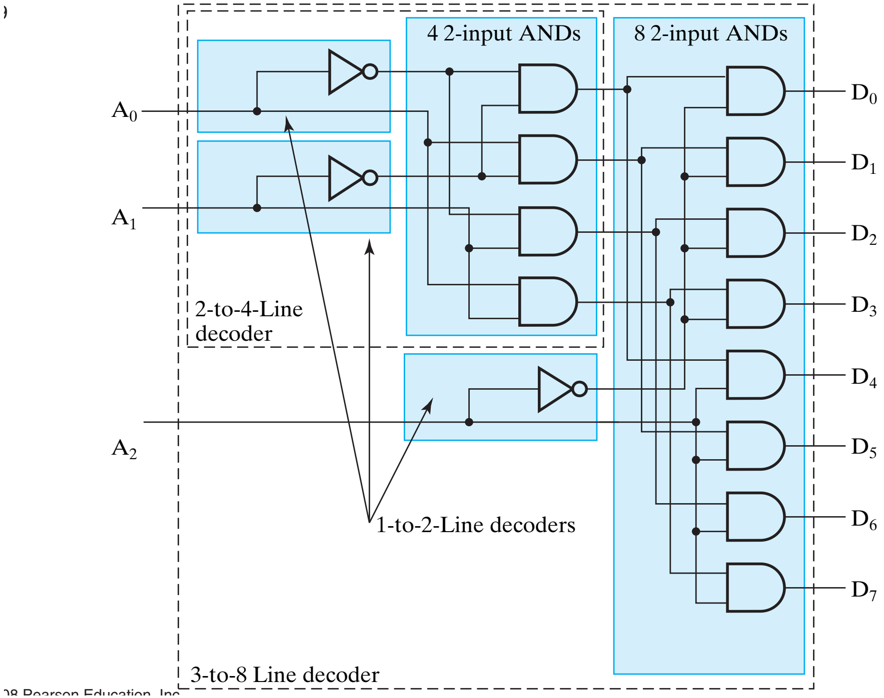


Figure: Source: Mano & Kime

Decoders

Decoder (3:8)

Hierarchical design: use small decoders to build bigger decoder



Takes n-bit input, uses it as an index to enable exactly one of 2^n outputs

Note: A_2 "selects" whether the 2-to-4 lin decoder is active in th top half (A_2=0) or the bottom (A_2=1)

Figure: Source: Mano & Kime

Multiplexers

Using n-bit selector input, select among one of 2^n choices

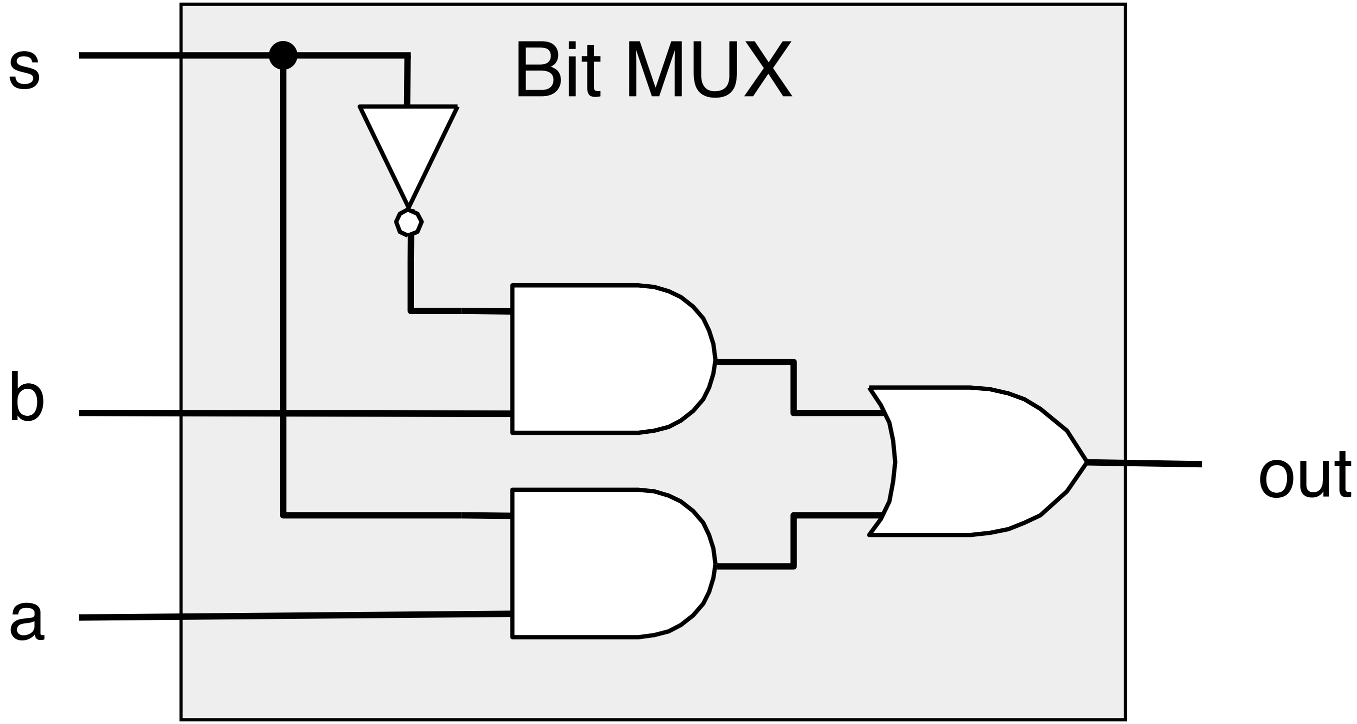


Figure: Source: CS:APP

Multiplexers

Using n-bit selector input, select among one of 2^n choices

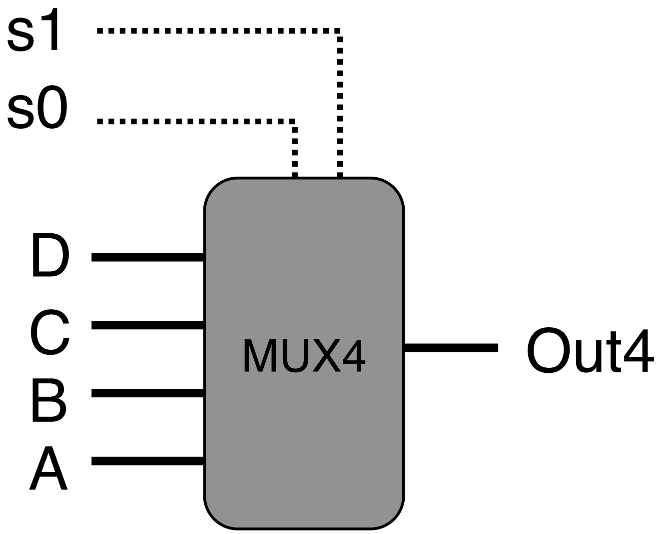


Figure: Source: CS:APP

Multiplexers

Using n-bit selector input, select among one of 2^n choices

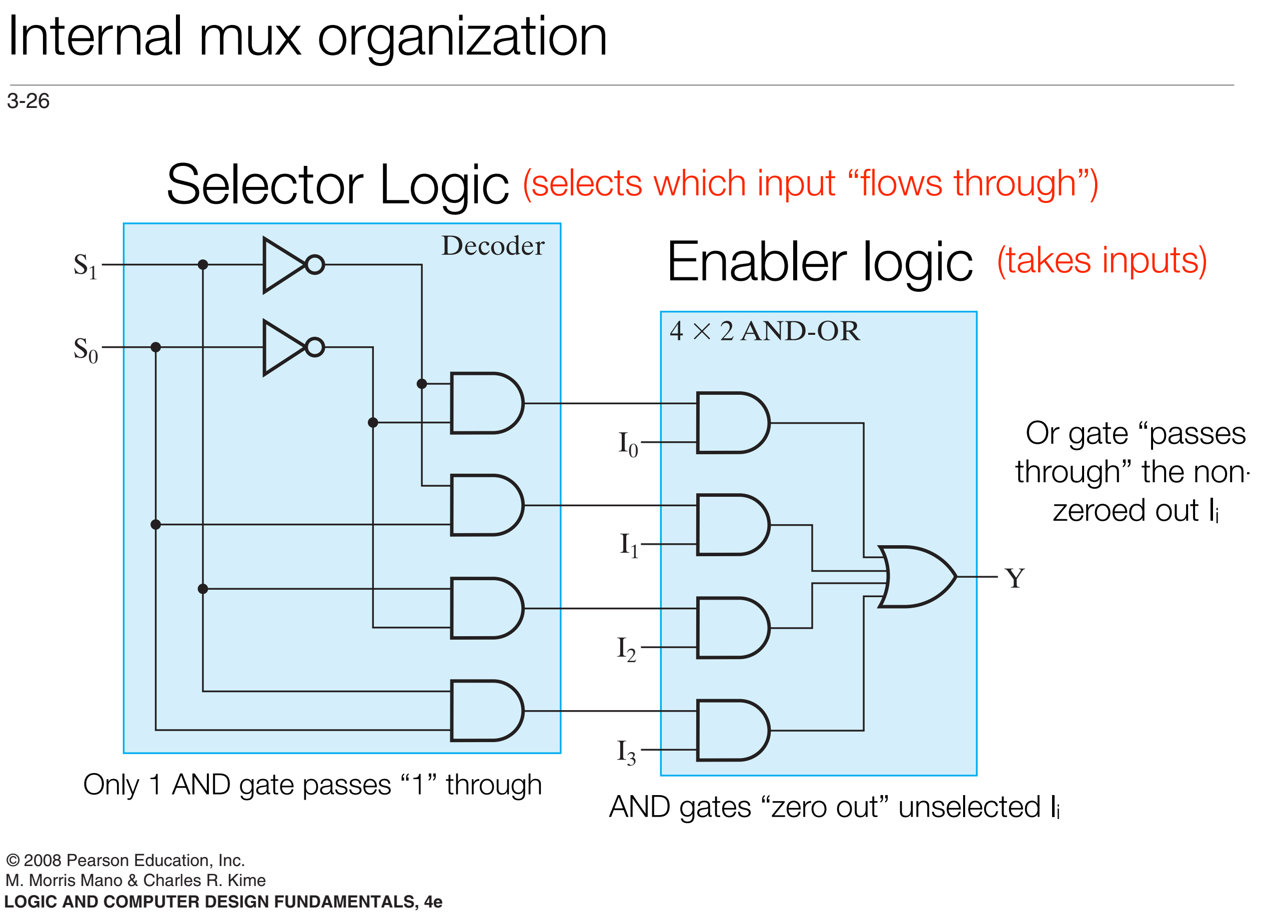


Figure: Source: Mano & Kime

Table of contents

directMapped read logic