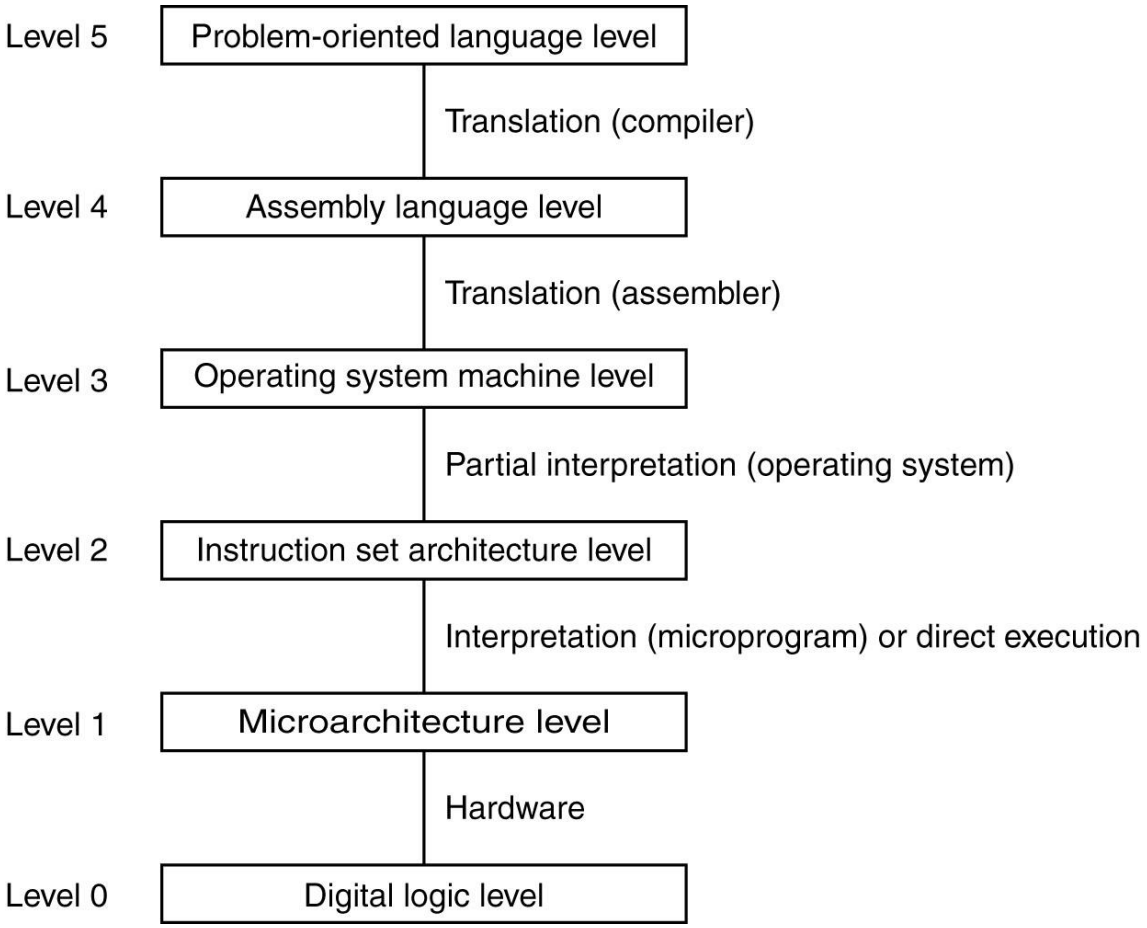
Topic 2: Digital Logic

ICT170: Foundations of Computer Systems

Recap



A six-level computer

The support method for each level is indicated below it

Overview

Topics:

- Logic Circuits
- Logic Operations
- Simple Logic Circuits
- Boolean Expressions
- Logic Operations
- Logic Circuit Examples

Objectives

In order to achieve the unit learning objectives, on successful completion of this topic, you should be able to:

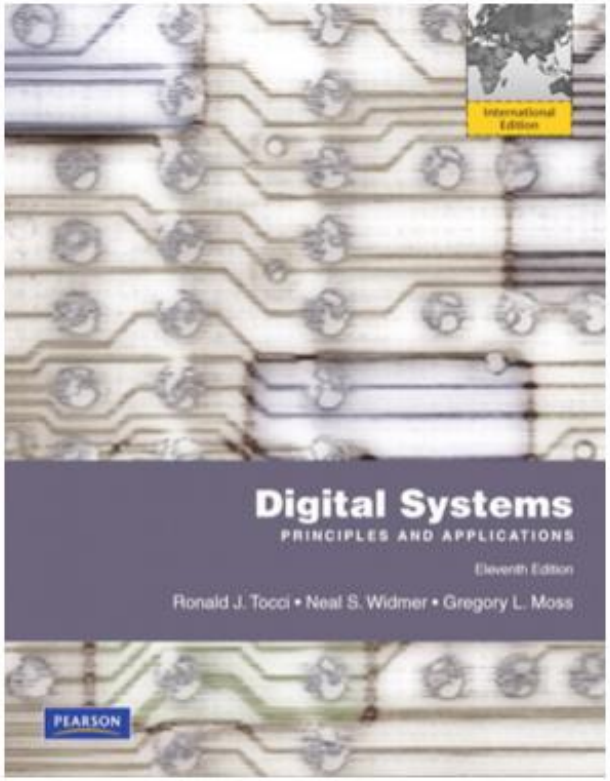
- Understand and describe the role of digital logic in building a modern computer.
- Understand the use of gates and Boolean algebra to perform simple calculations.
- Understand the common digital logic circuits.
- Understand the different types of memory at the digital logic level.
- Understand the variety of CPUs, buses and interfaces at the digital logic level.

Reading

Title: **Digital Systems: Principles And Applications (11th Edition)** Author: Ronald J. Tocci, Neal Widmer, Greg Moss,
Publisher: Pearson Higher Ed USA
Pages: 992
Published: 2010-08-07
ISBN-10: 0130387932
ISBN-13: 9780130387936
Category: Engineering, professional & Technical,
Binding: Hardcover (11)
Reading: Chapter 3 “Describing Logic Circuits”

Resources:

- The recorded lectures available on LMS.
- The lecture slides available on LMS.



Murdoch
UNIVERSITY

Digital Logic: Logic Circuits

Mur
UNIV

So? How do we build a computer?

Question?

How do we implement the parts we've talked about last week (CPU, memory, storage, I/O, buses)

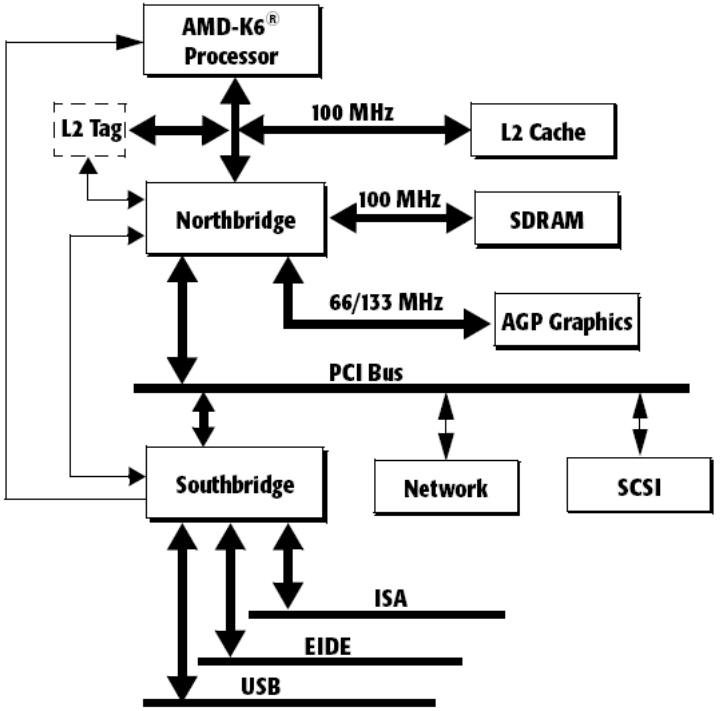
Answer:

Using basic Electronic components

First Vacuum tubes, then transistors and semiconductors

Recall that computers are just devices that shift bits around and perform simple calculations!

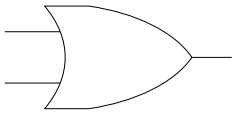
Simple computer circuits are called "Logic Circuits"



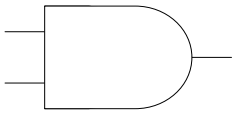
Boolean Algebra to Logic Gates

Logic circuits are built from components called logic gates.

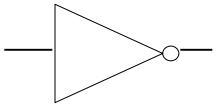
The logic gates correspond to Boolean operations +, *, '



OR
+



AND
*



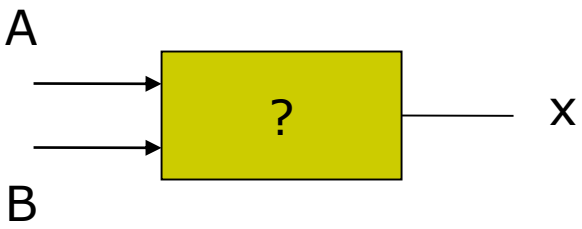
NOT
'

Binary operations have two inputs, unary has one

Truth Tables

- A truth table is a means for describing how a logic circuit's output depends on the logic levels present at the circuit's inputs.

Inputs		Output
A	B	x
0	0	1
0	1	0
1	0	1
1	1	0



Murdoch
UNIVERSITY

Logic Operations

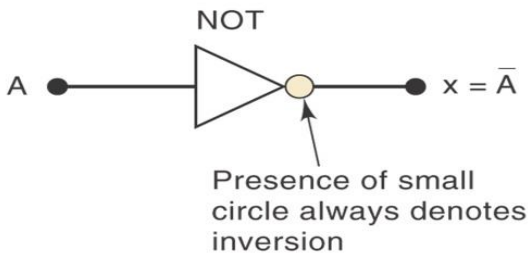
Logic Operations: NOT

- The NOT operation is a unary operation, taking only one input variable.
- Boolean expression for the NOT operation:
  $x = \bar{A}$
- The above expression is read as "x equals the inverse of A"
- Also known as inversion or complementation.
- Can also be expressed as: A'

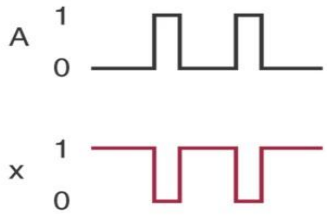
NOT

A	$x = \bar{A}$
0	1
1	0

(a)



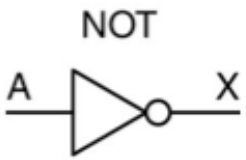
(b)



(c)

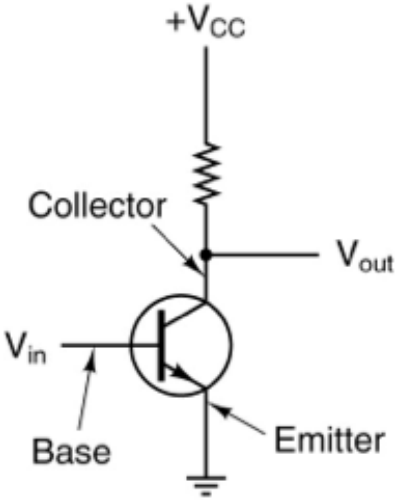
Logic Operations: NOT

Logical

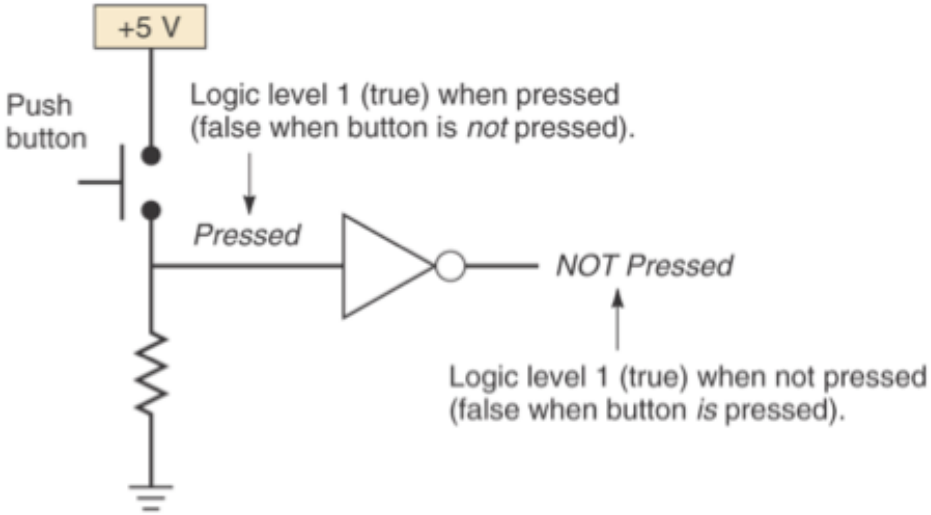


A	X
0	1
1	0

Electrical



Usage



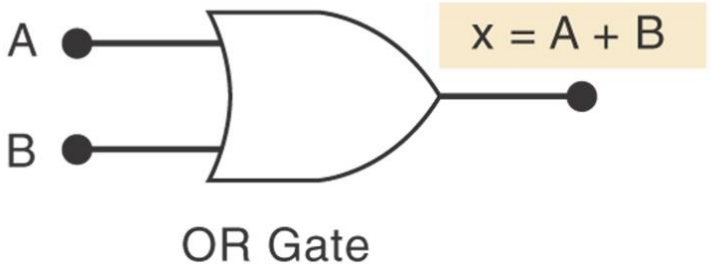
Logic Operations: OR

- Boolean expression for the OR operation:
  x =A + B
- The above expression is read as "x equals A OR B"

OR

A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1

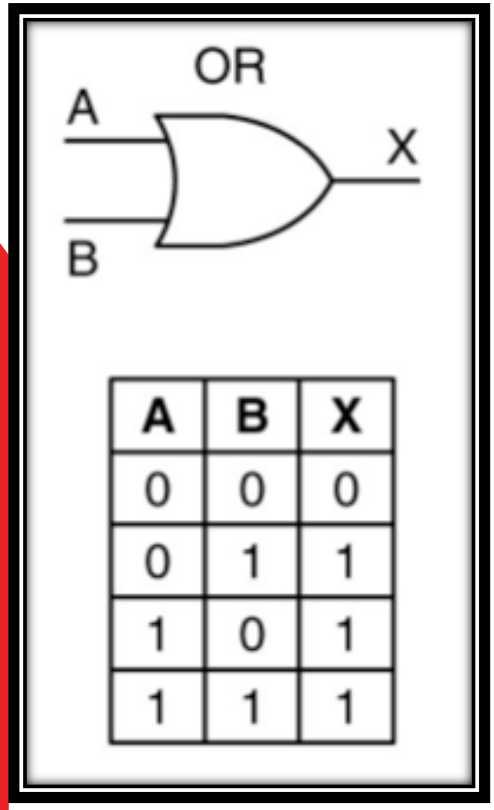
(a)



(b)

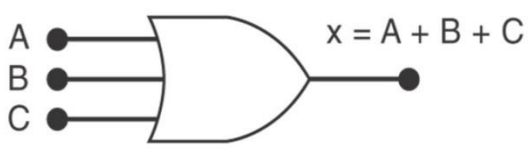
Logic Operations: OR

Example with 2 inputs



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

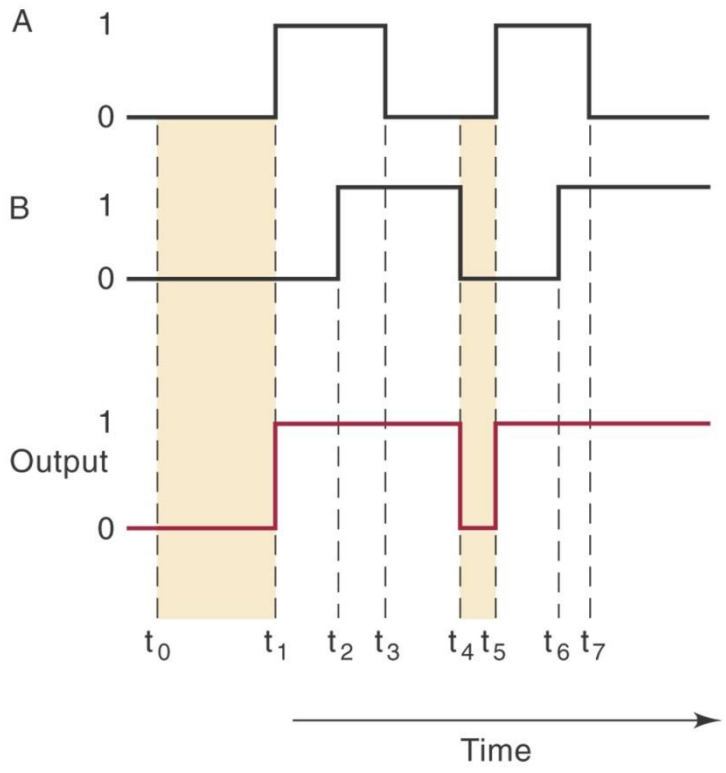
Example with 3 inputs

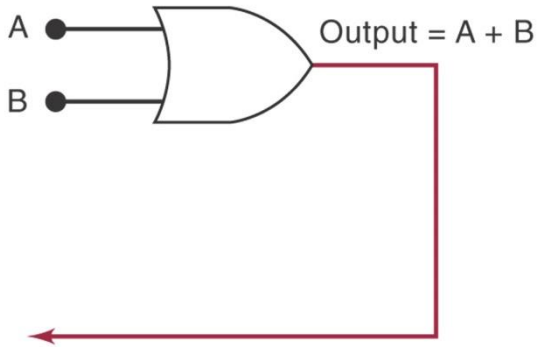


A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Logic Operations: OR

Timing of an OR gate





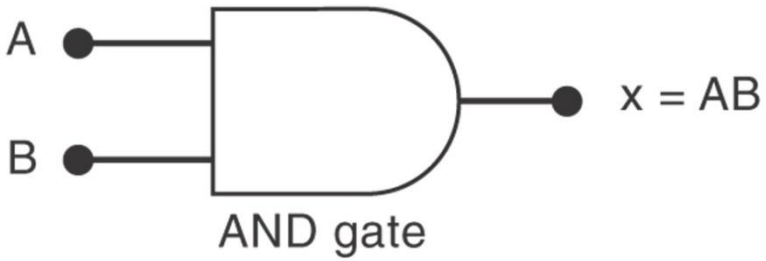
Logic Operations: AND

- Boolean expression for the AND operation:
  x =AB
- The above expression is read as "x equals A AND B"

AND

A	B	x = A • B
0	0	0
0	1	0
1	0	0
1	1	1

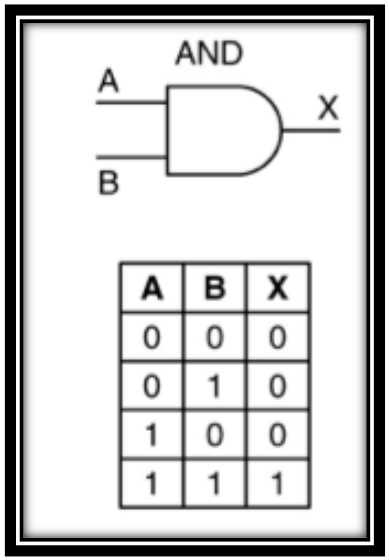
(a)



(b)

Logic Operations: AND

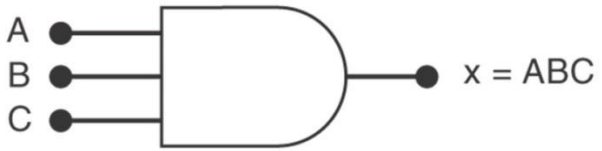
Example with 2 inputs



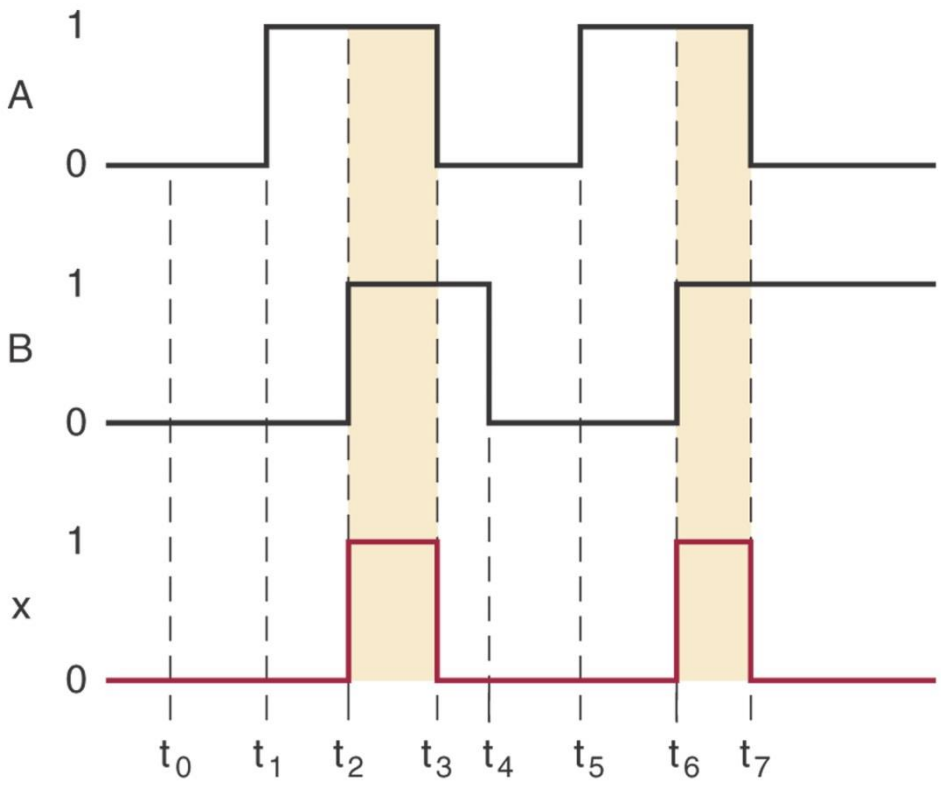
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

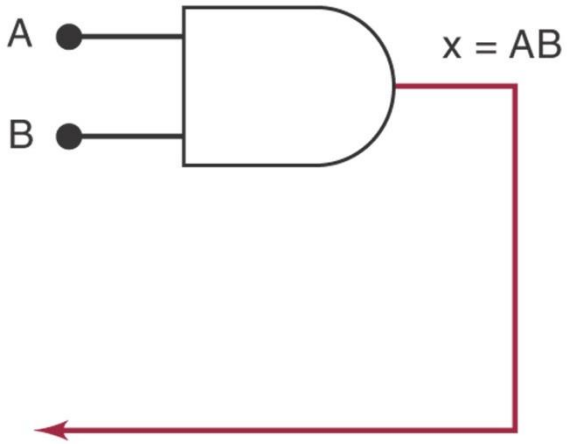
Example with 3 inputs

A	B	C	x = 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



Logic Operations: AND



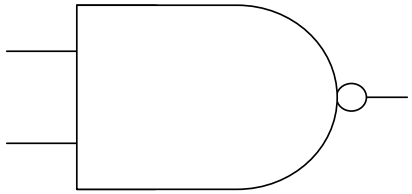


Timing of an AND gate

Logic Operations: NAND and NOR

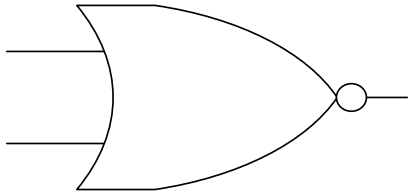
NAND and NOR gates can greatly simplify circuit diagrams. As we will see, you can use these gates wherever you could use AND, OR, and NOT.

NAND



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

NOR

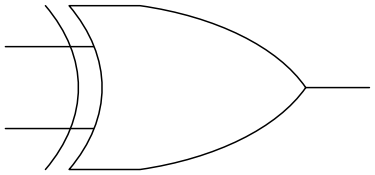


A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Logic Operations: XOR and XNOR

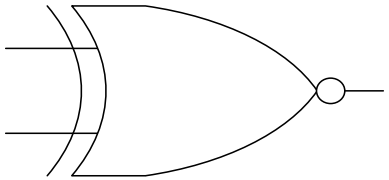
XOR is used to choose between two mutually exclusive inputs. Unlike OR, XOR is true only when one input or the other is true, not both.

XOR



A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

XNOR



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

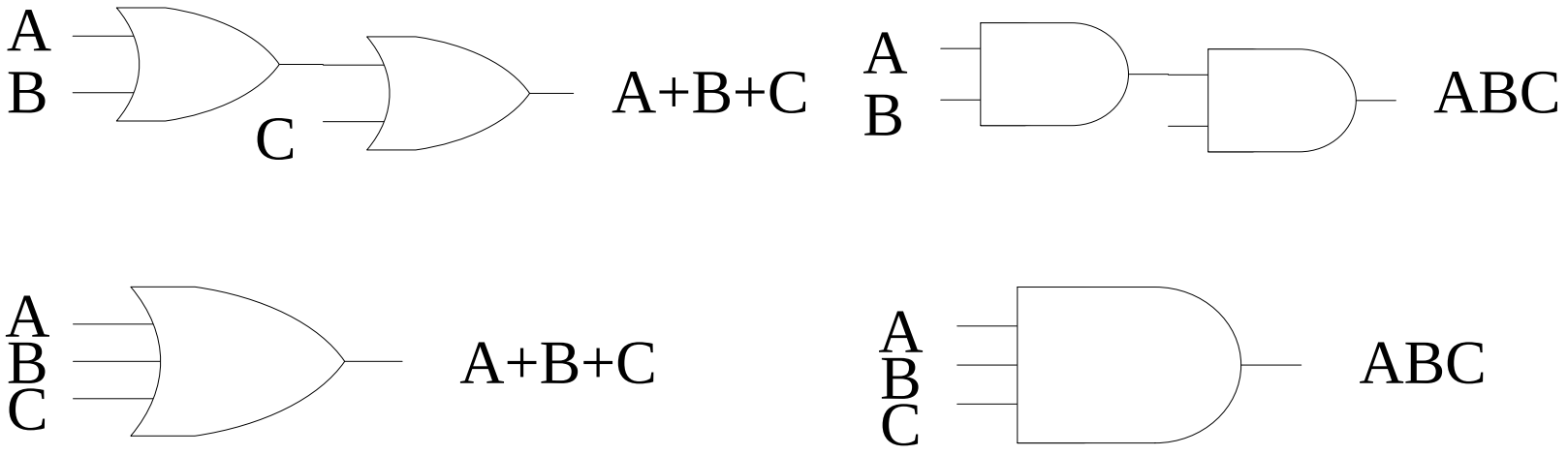
Murdoch
UNIVERSITY

Simple Logic Circuits

Mu
UNIV

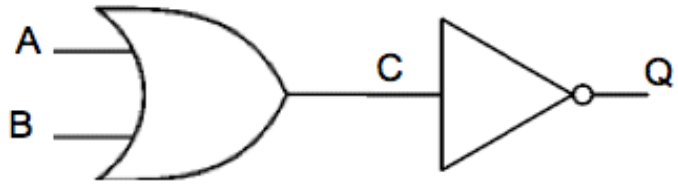
Simple Logic Circuits

- Because + and * are binary operations, they can be cascaded together to OR or AND multiple inputs.



Logic Circuits: Worksheet

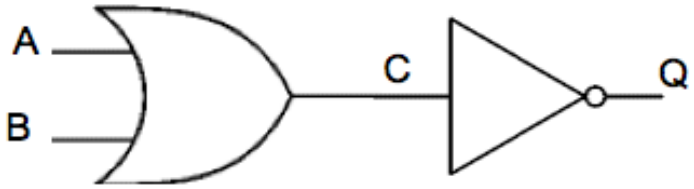
3. The figure below shows a logic circuit and its incomplete truth table. Complete the below truth table.



A	B	C	Q
0	0		
0	1		
1	0		
1	1		

Logic Circuits: Worksheet

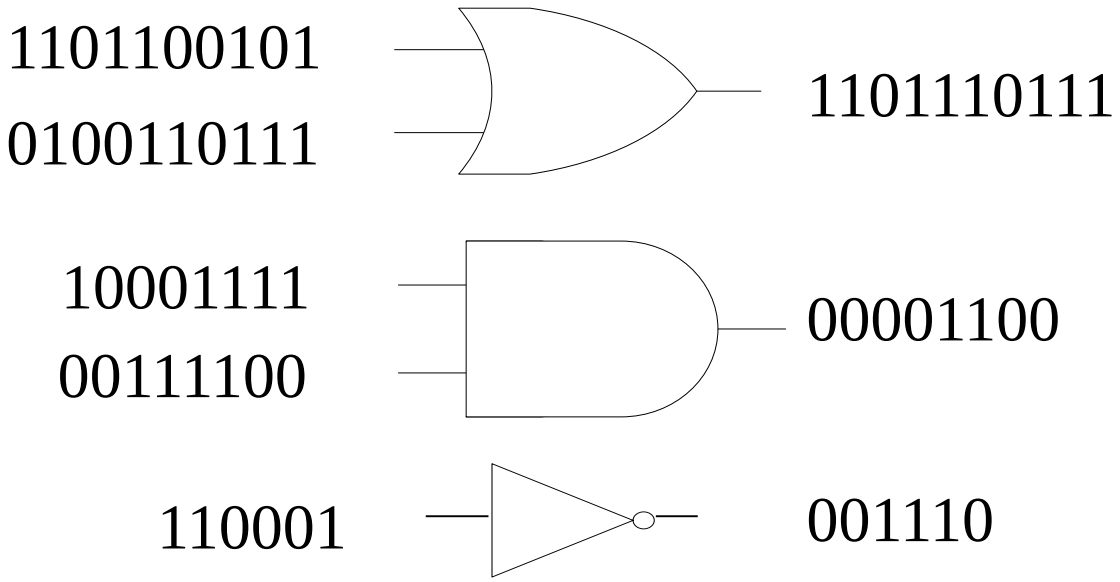
3. The figure below shows a logic circuit and its incomplete truth table. Complete the below truth table.



A	B	C	Q
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Simple Logic Circuits

- For convenience, it is sometimes useful to think of the logic gates processing *n*-bits at a time. This really refers to *n* instances of the logic gate, not a single logic date with *n*-inputs.



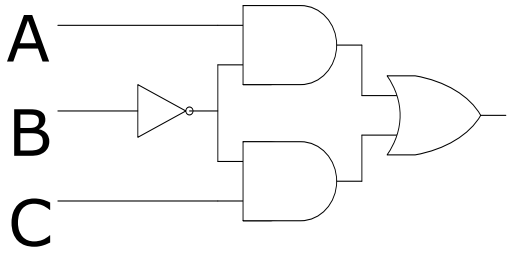
Murdoch
UNIVERSITY

Boolean Expressions

Mu
UNIV

Boolean Expressions

- Logic Circuits = Boolean Expressions
- All logic circuits are equivalent to Boolean expressions and any Boolean expression can be rendered as a logic circuit.
- AND-OR logic circuits are equivalent to sum-of-products form.
- Consider the following circuits:

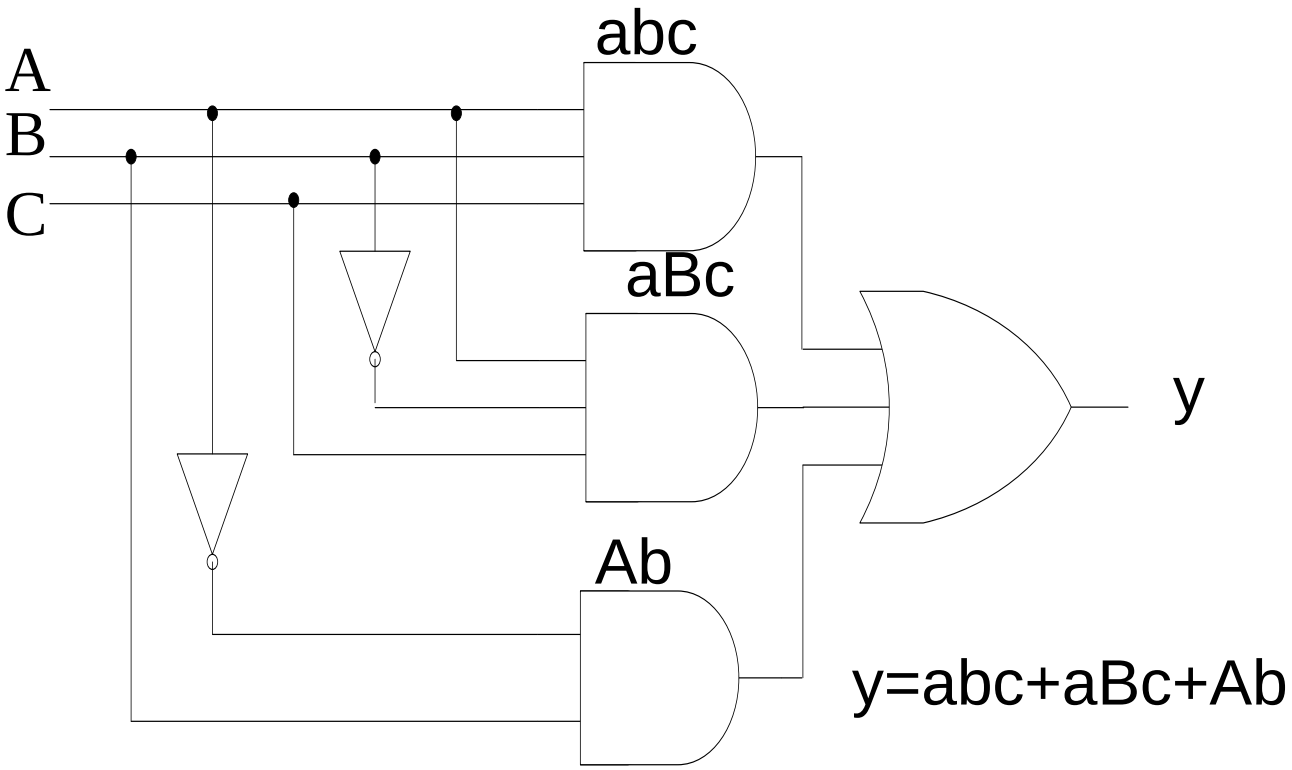


Y $y=A\overline{B}+\overline{B}C$

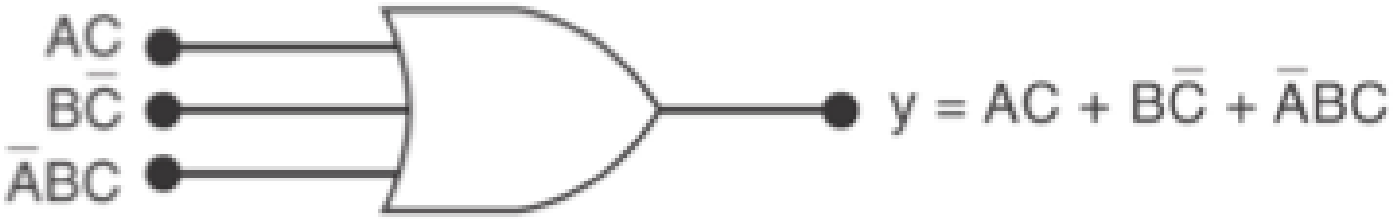
Sometimes written as, y=aB+Bc

Note the convention of CAPS and small letter

Boolean Expressions: examples



Boolean Expressions: examples

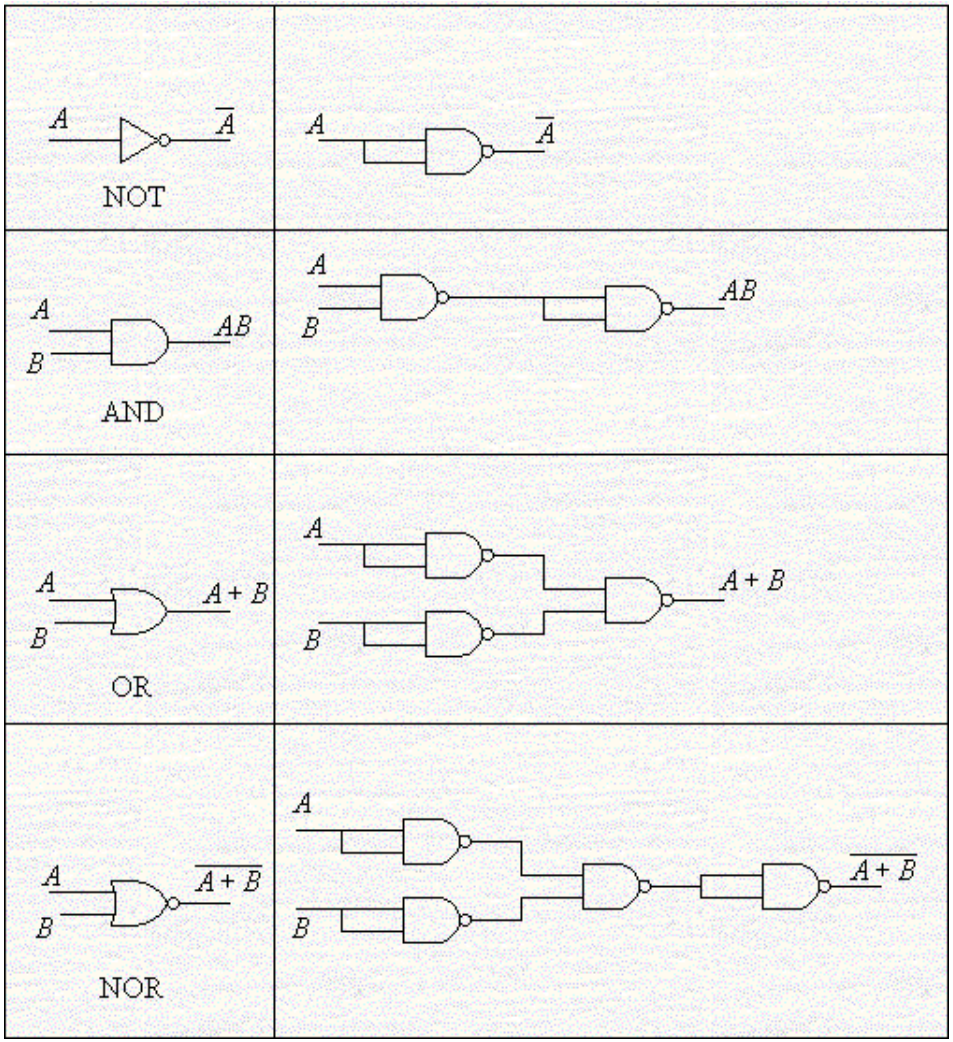


Murdoch
UNIVERSITY

Logic Operations

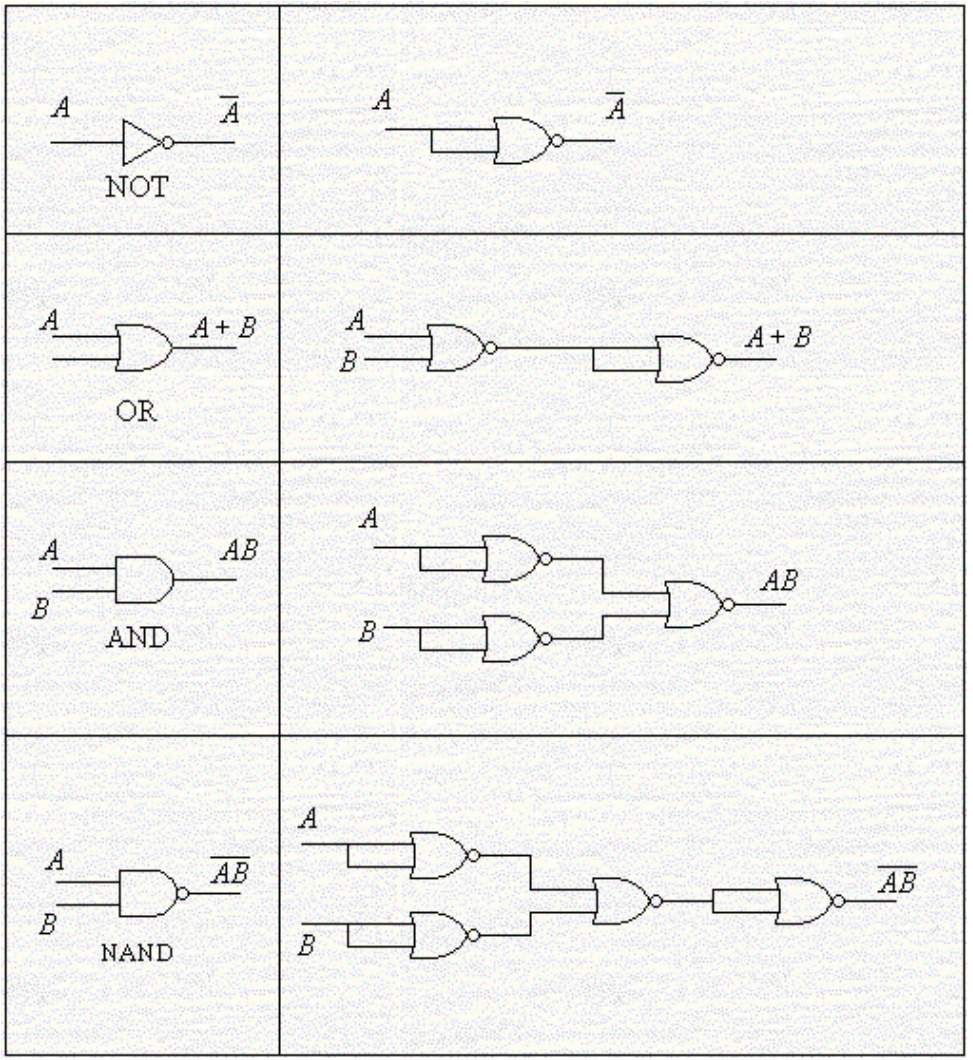
Universal gate: NAND

- Any logic circuit can be built using only NAND gates, or only NOR gates. They are the only logic gate needed.
- Here are the NAND equivalents:



Universal gate: NOR

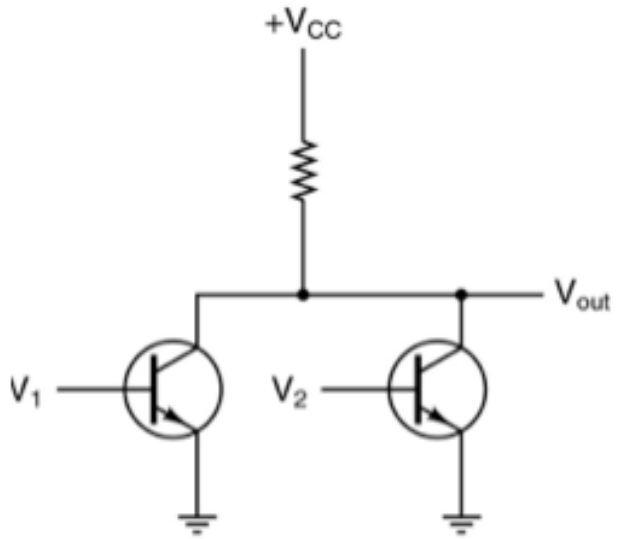
- Any logic circuit can be built using only NAND gates, or only NOR gates. They are the only logic gate needed.
- Here are the NOR equivalents:
- NAND and NOR can be used to reduce the number of required gates in a circuit.



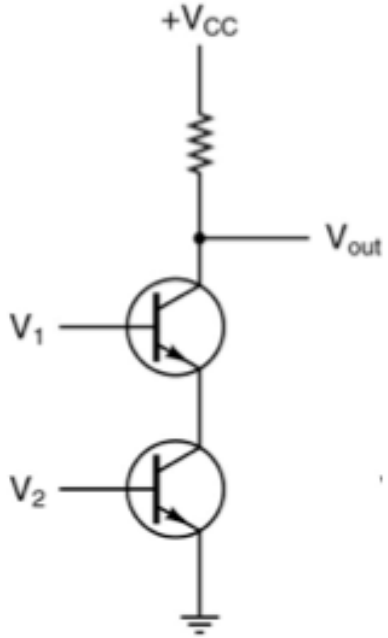
Universal gates

- So, if we can build any logic circuit out of only 1 type of gate, that makes building the logic circuit of any computer very easy
- Just use 1 type of gate…

NOR



NAND



Murdoch
UNIVERSITY

Logic Circuit Examples

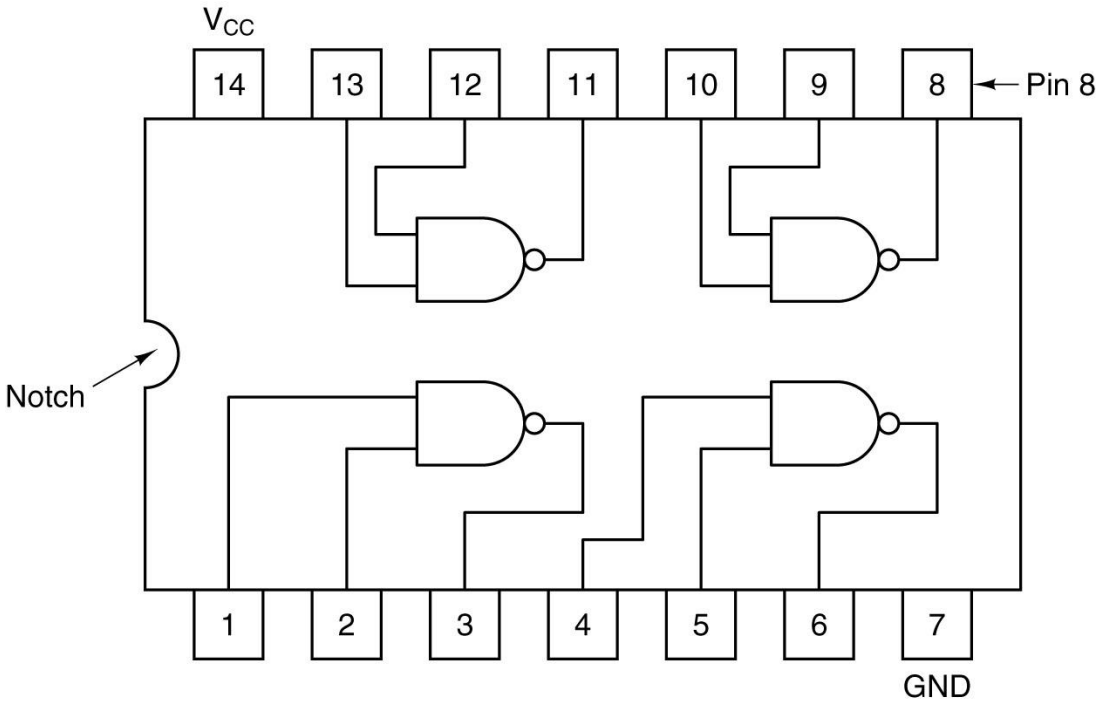
Mur
UNIV

What can we build with logic circuits?

- Integrated circuits for general use
- Multiplexers
- Decoders
- Comparators
- Programmable Logic Arrays
- Shifters
- Adders
- Arithmetic Logic Units
- Latches
- Flip-Flops and therefore Memory

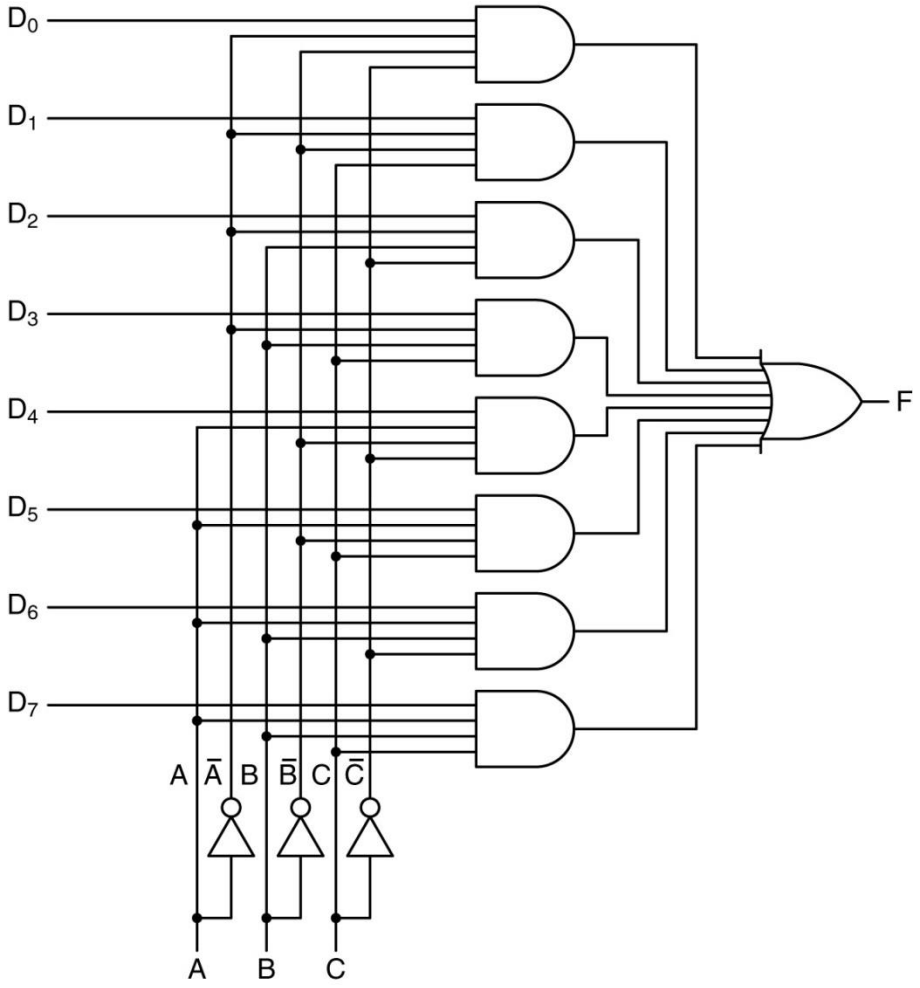
Integrated Circuits

An SSI chip containing four gates.



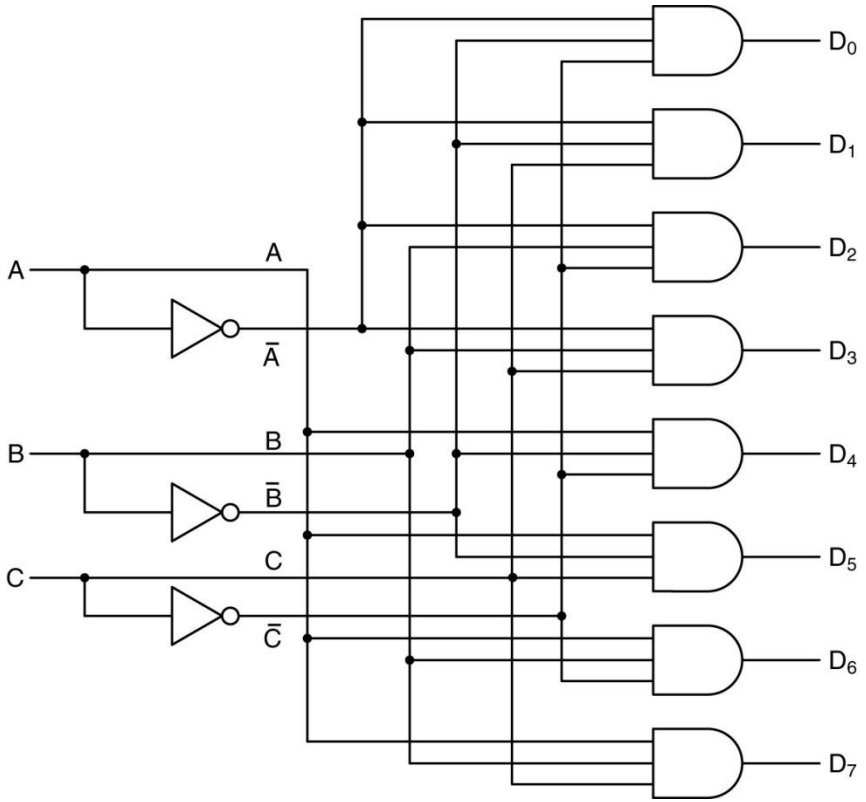
Multiplexers

An eight-input multiplexer circuit.



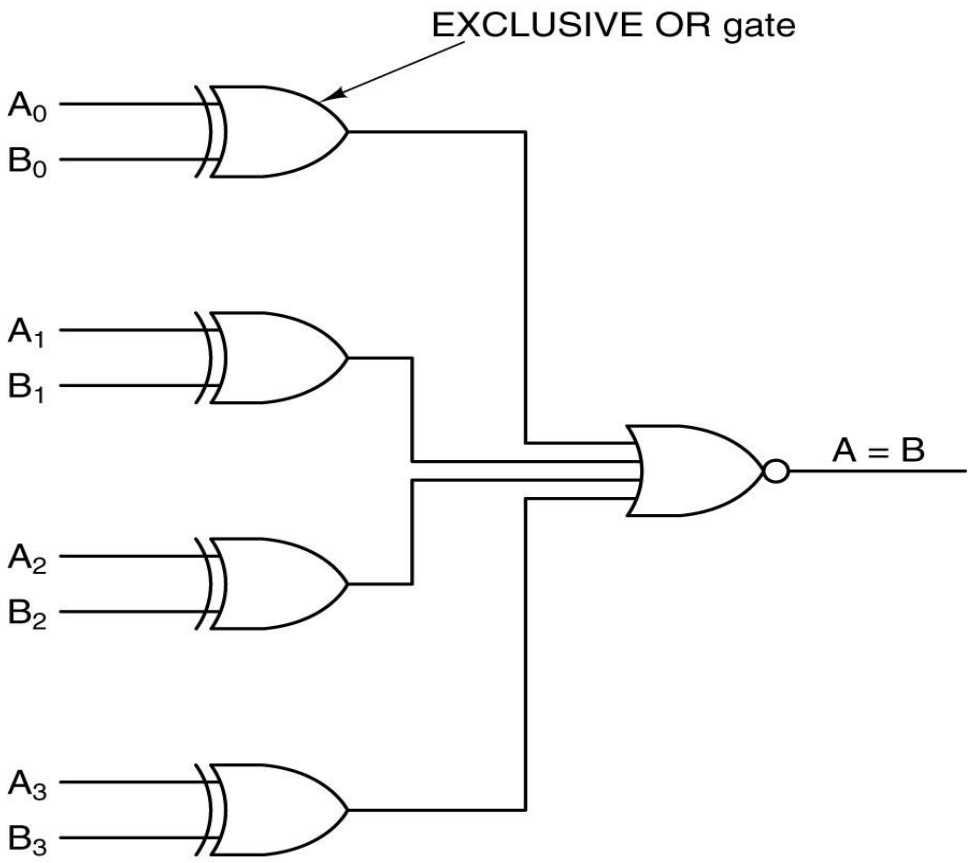
Decoders

A 3-to-8 decoder circuit.



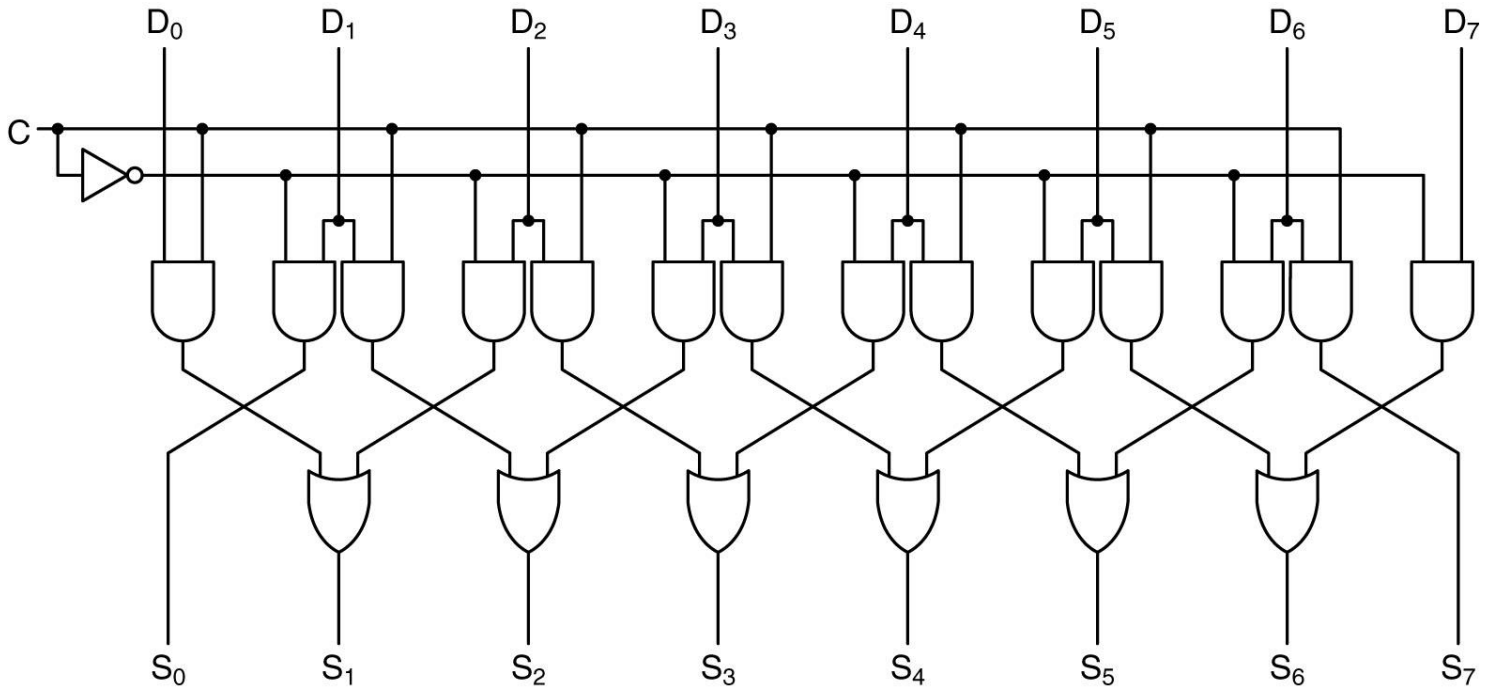
Comparators

A simple 4-bit comparator.



Shifters

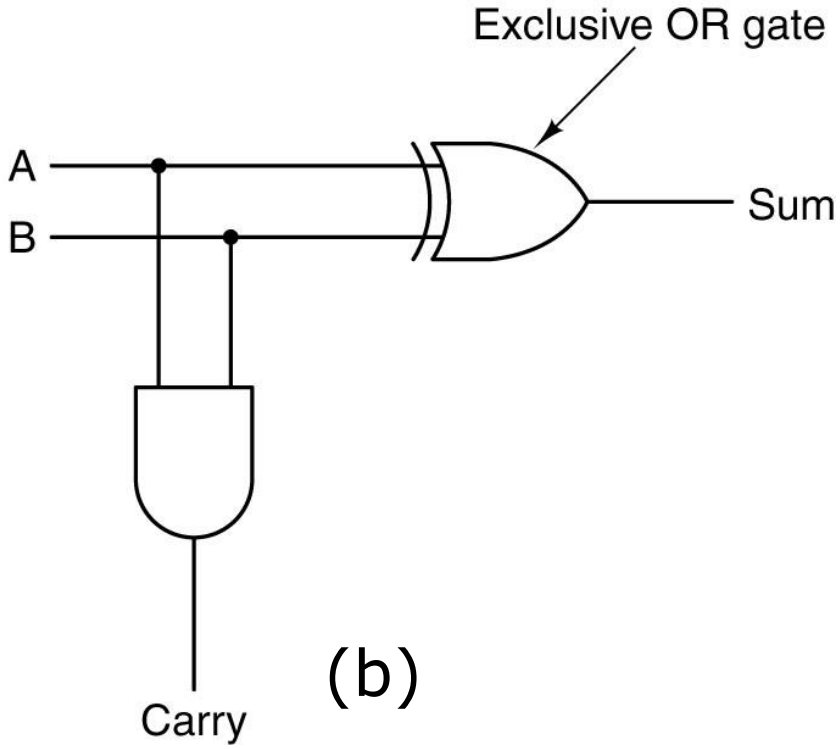
A 1-bit left/right shifter.



Adders (1): Half adder

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a)



(b)

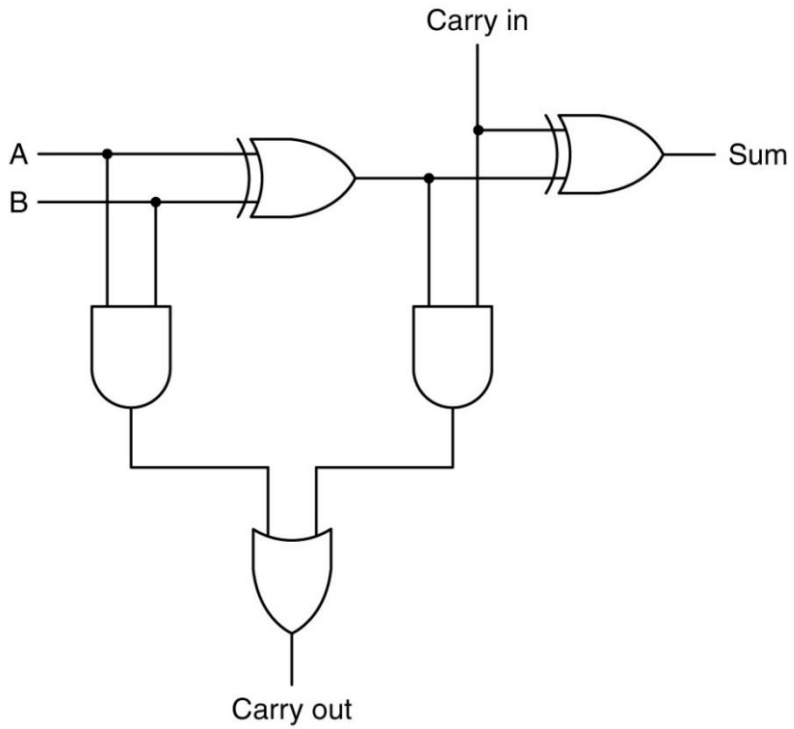
(a) A truth table for 1-bit addition.

(b) A circuit for a half adder.

Adders (2): full adder

A	B	Carry in	Sum	Carry out
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

(a)



(b)

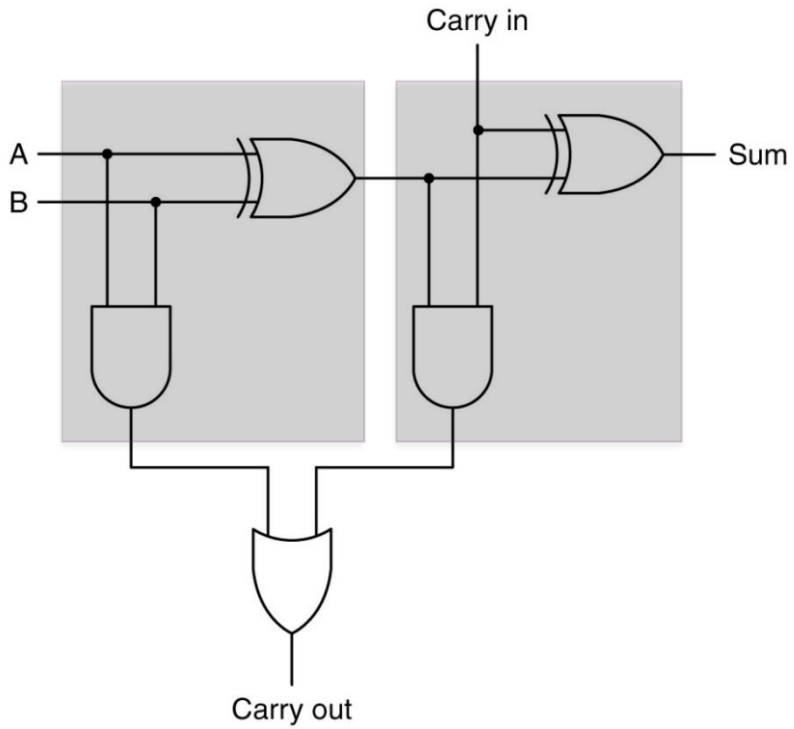
(a) Truth table for a full adder.

(b) Circuit for a full adder.

Adders (3): full adder

A	B	Carry in	Sum	Carry out
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

(a)



(b)

(a) Truth table for a full adder.

(b) Circuit for a full adder.

Arithmetic Logic Units

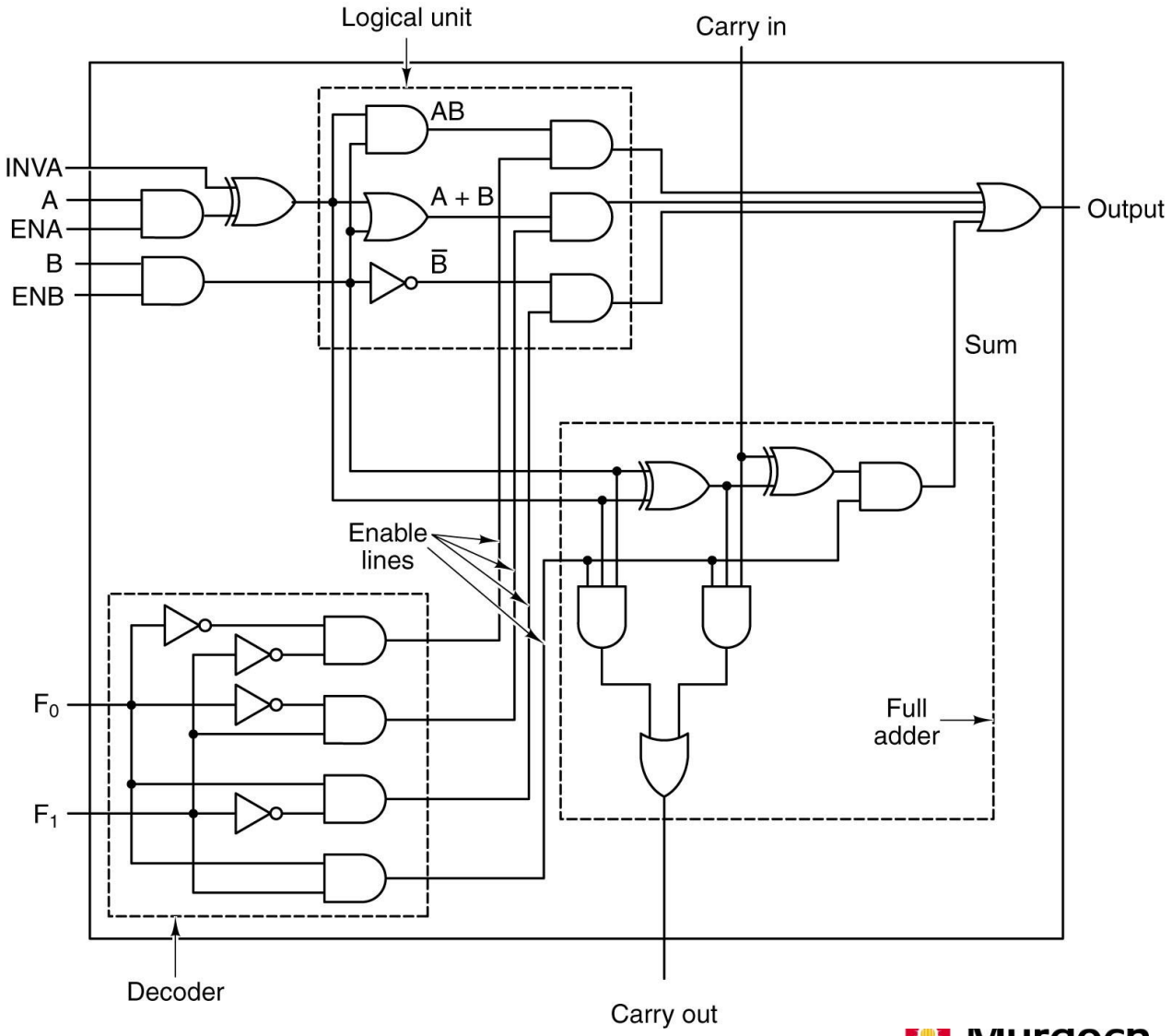
A 1-bit ALU.

Can do:
A AND B
A OR B
Inv B
A + B (full Adder)

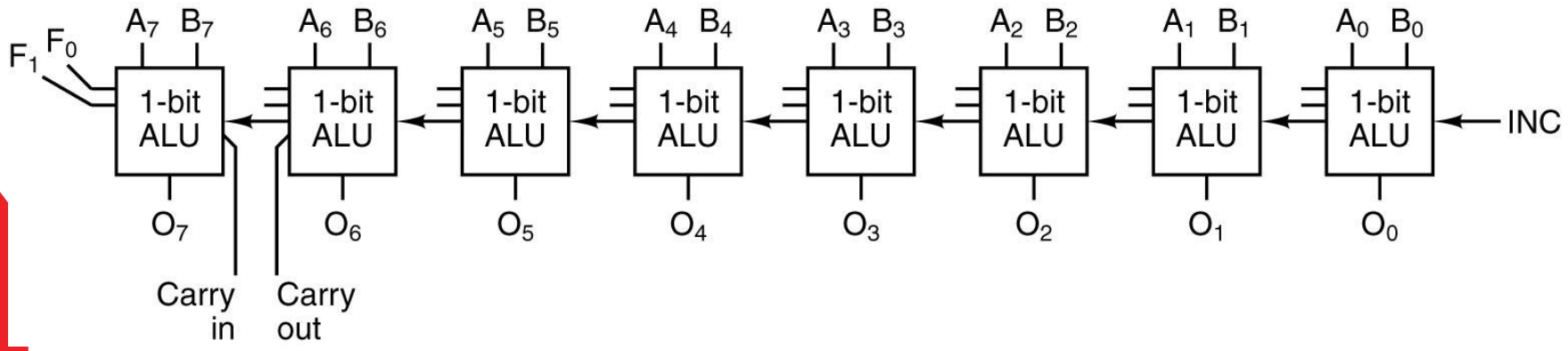
Selected from:
F0, F1:
00, 01, 10, 11

ENVA ENVB:
Forces A or B to 0.
(Normally 1)

INVA: Inv A



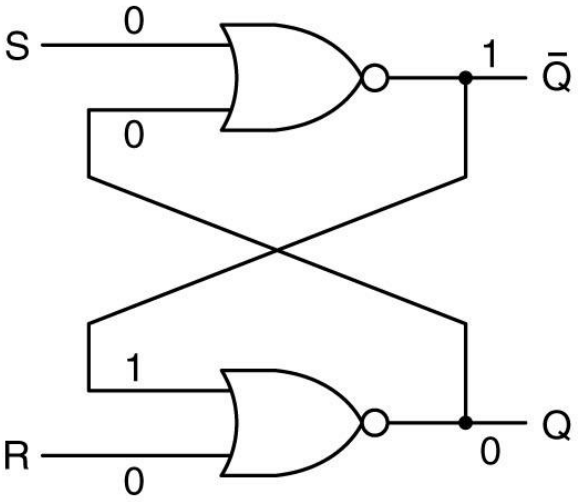
Arithmetic Logic Units



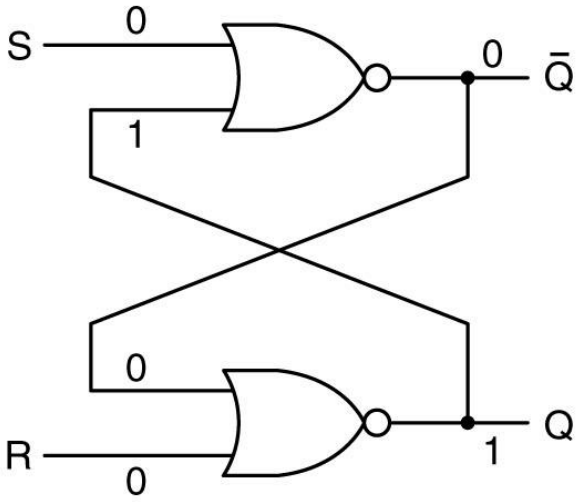
Eight 1-bit ALU slices connected to make an 8-bit ALU.

The enables and invert signals are not shown for simplicity.

Latches (1)



(a)



(b)

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

(c)

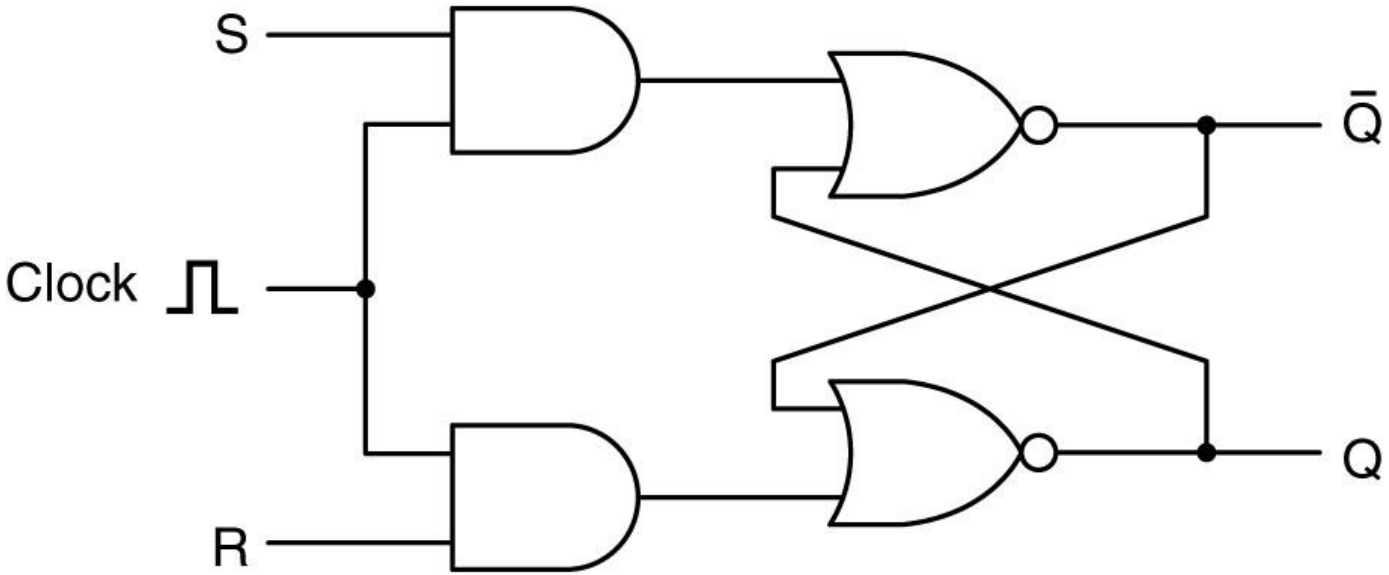
(a) NOR latch in state 0.

(b) NOR latch in state 1.

(c) Truth table for NOR.

Latches (2)

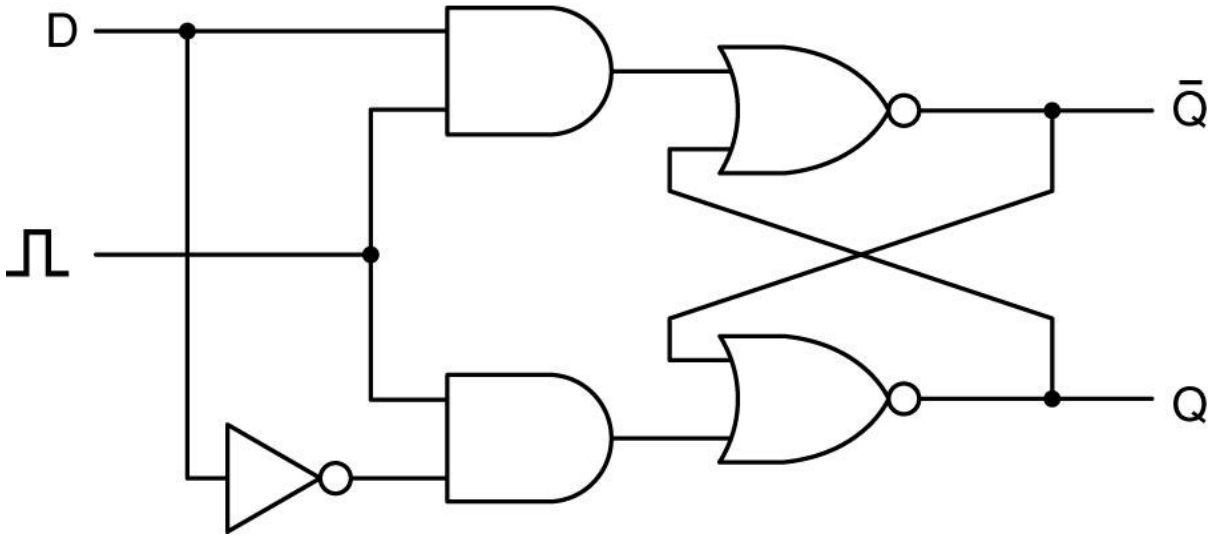
A clocked SR latch.



Only allows latch to change at certain times
Prevents changes when e.g. memory is being retrieved

Latches (3)

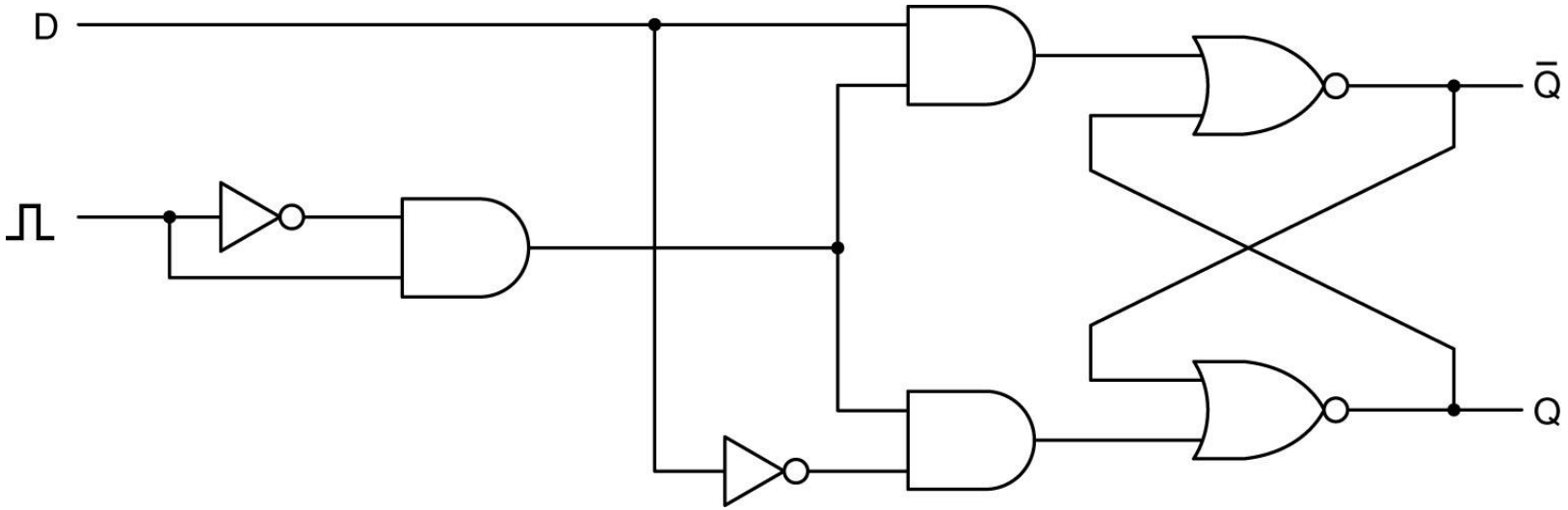
A clocked D latch.



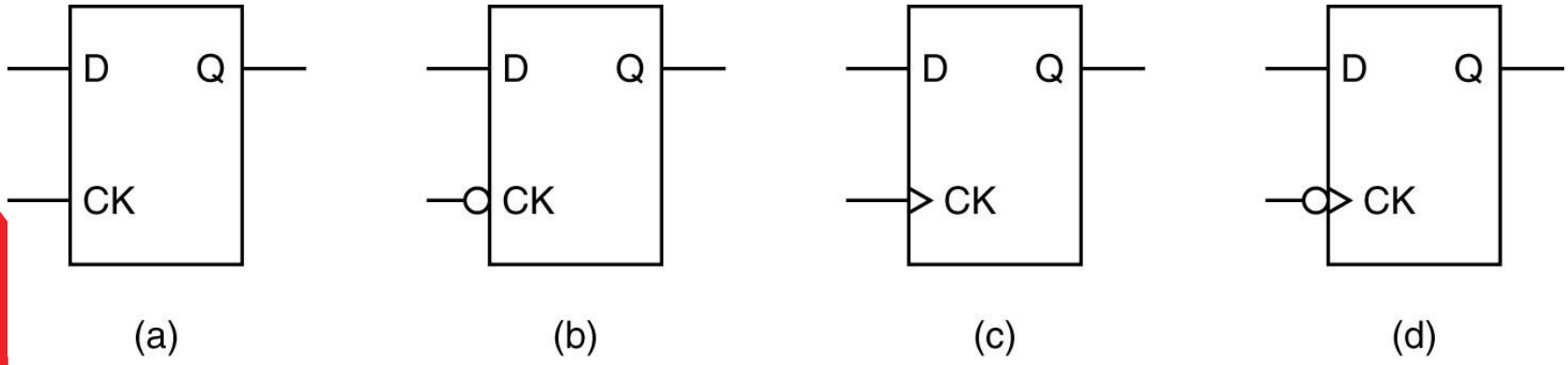
Loads the value of D into memory at a clock pulse.

Flip-Flops

A D flip-flop.

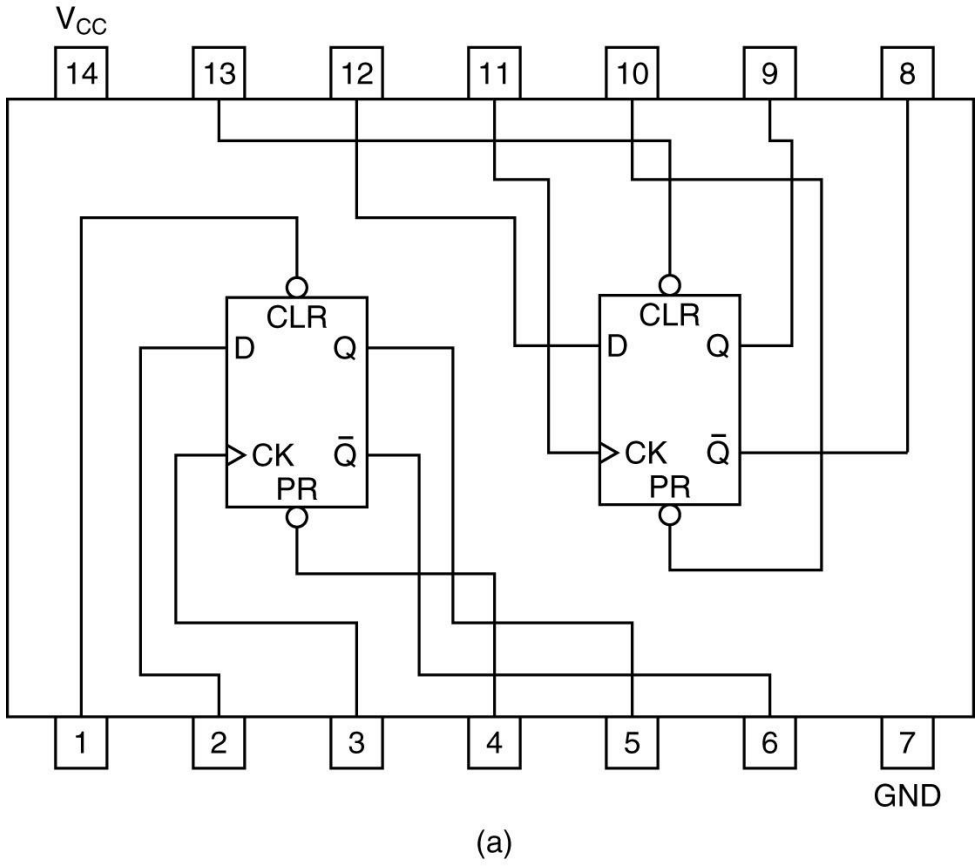


Flip-Flops



D latches and flip-flops.

Flip-Flops

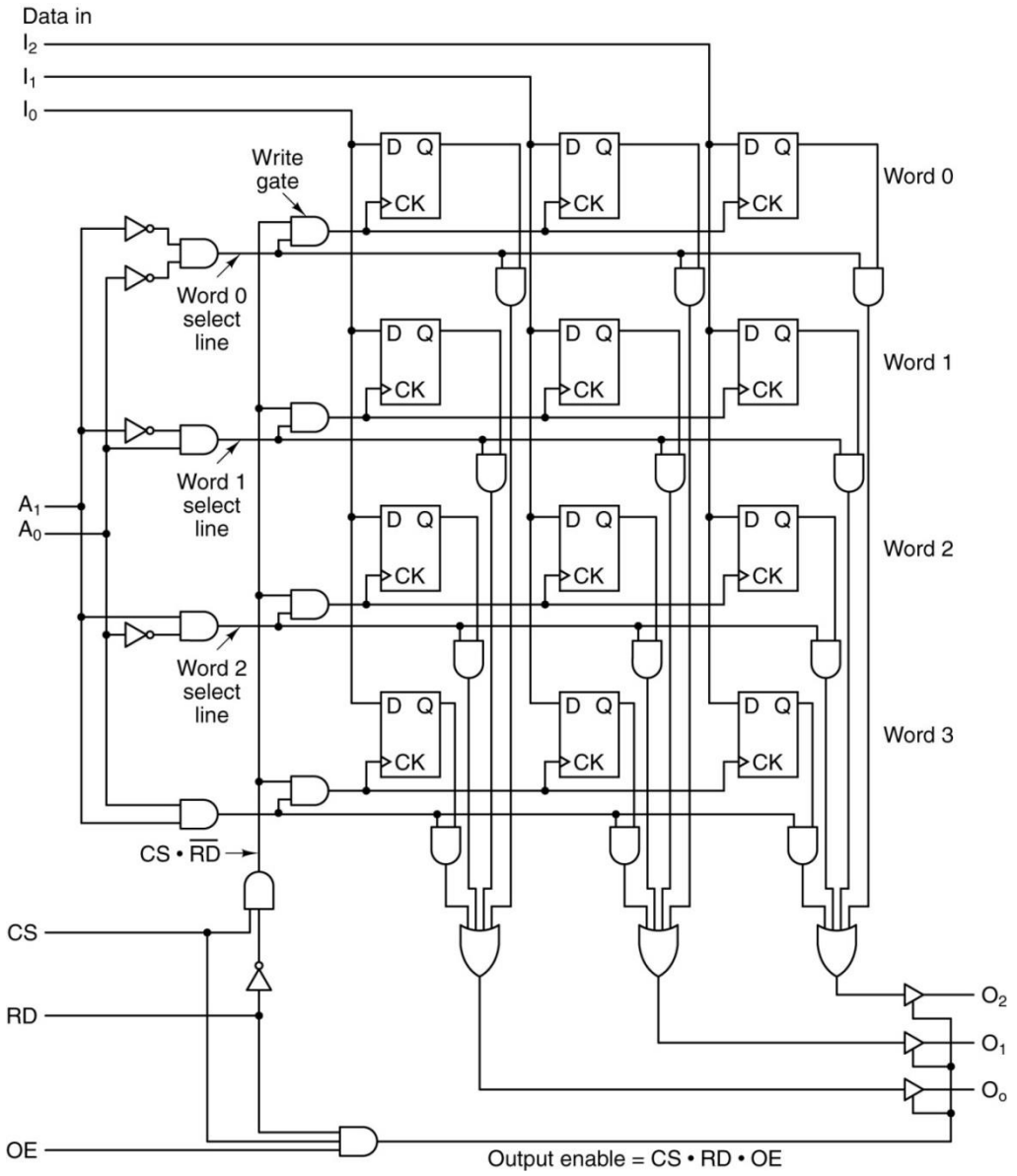


Dual D flip-flop.

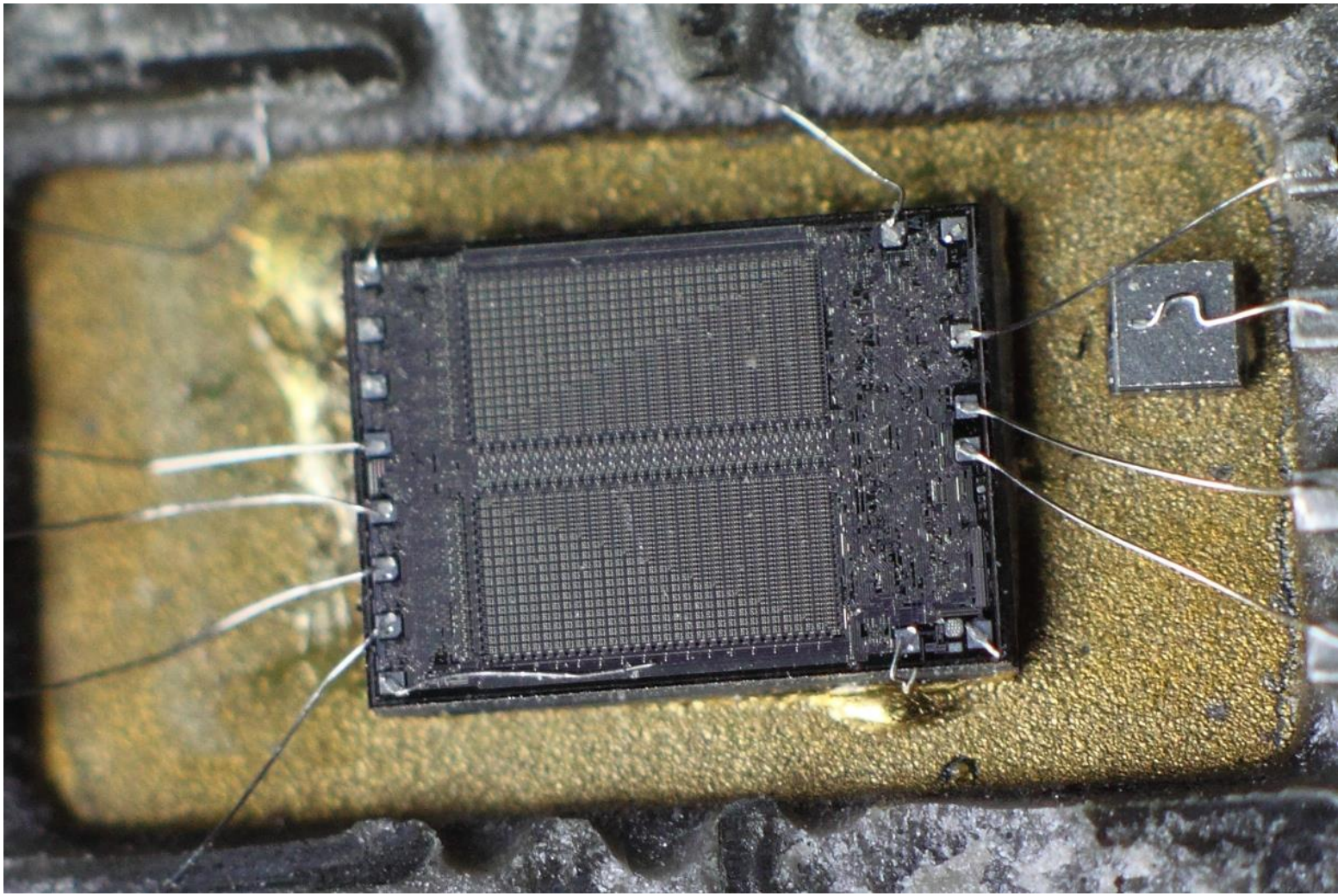
Memory Organization

Logic diagram for a 4 x 3 memory.

Each row is one of the four 3-bit words.



Pretty picture of a RAM chip?



Murdoch
UNIVERSITY

Summary

Mu
UNIV

Summary

- Logic Circuits
- Logic Operations
- Simple Logic Circuits
- Boolean Expressions
- Logic Operations
- Logic Circuit Examples