Gates and Circuits

Chapter Goals

- Identify the basic gates and describe the behavior of each
- Describe how gates are implemented using transistors
- Combine basic gates into circuits
- Describe the behavior of a gate or circuit using Boolean expressions, truth tables, and logic diagrams

Computers and Electricity

Gate

A device that performs a basic operation on electrical signals

Circuits

Gates combined to perform more complicated tasks

Computers and Electricity

How do we describe the behavior of gates and circuits?

Boolean expressions

Uses Boolean algebra, a mathematical notation for expressing two-valued logic

Logic diagrams

A graphical representation of a circuit; each gate has its own symbol

Truth tables

A table showing all possible input value and the associated output values

Gates

Six types of gates

- **NOT**
- **AND**
- **OR**
- **XOR**
- **NAND**
- **NOR**

Typically, logic diagrams are black and white with gates distinguished only by their shape

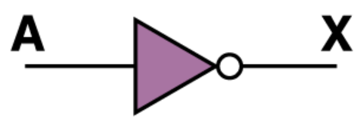
NOT Gate

A NOT gate accepts one input signal (0 or 1) and returns the opposite signal as output

Boolean Expression

$$X = A'$$

Logic Diagram Symbol



Truth Table

A	X
0	1
1	0

AND Gate

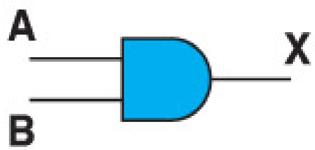
An AND gate accepts two input signals

If both are 1, the output is 1; otherwise, the output is 0

Boolean Expression

$$X = A \cdot B$$

Logic Diagram Symbol



Truth Table

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

OR Gate

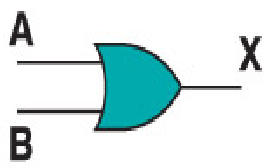
An OR gate accepts two input signals

If both are 0, the output is 0; otherwise, the output is 1

Boolean Expression

$X = A + B$

Logic Diagram Symbol



Truth Table

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

XOR Gate

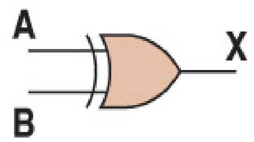
An XOR gate accepts two input signals

If both are the same, the output is 0; otherwise, the output is 1

Boolean Expression

$$X = A \oplus B$$

Logic Diagram Symbol



Truth Table

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

XOR Gate

Note the difference between the XOR gate and the OR gate; they differ only in one input situation

When both input signals are 1, the OR gate produces a 1 and the XOR produces a 0

XOR is called the *exclusive OR*

NAND Gate

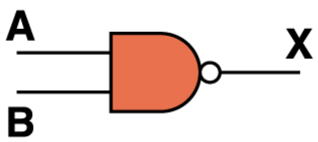
The NAND gate accepts two input signals

If both are 1, the output is 0; otherwise, the output is 1

Boolean Expression

$$X = (A \cdot B)'$$

Logic Diagram Symbol



Truth Table

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

Figure 4.5 Various representations of a NAND gate

NOR Gate

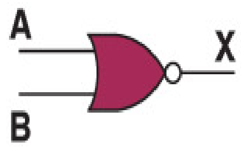
The NOR gate accepts two input signals

If both are 0, the output is 1; otherwise, the output is 0

Boolean Expression

$X = (A + B)'$

Logic Diagram Symbol



Truth Table

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

Review of Gate Processing

A NOT gate inverts its single input

An AND gate produces 1 if both input values are 1

An OR gate produces 0 if both input values are 0

An XOR gate produces 0 if input values are the same

A NAND gate produces 0 if both inputs are 1

A NOR gate produces a 1 if both inputs are 0

Gates with More Inputs

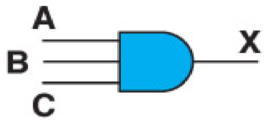
Gates can be designed to accept three or more input values

A three-input AND gate, for example, produces an output of 1 only if all input values are 1

Boolean Expression

$$X = A \cdot B \cdot C$$

Logic Diagram Symbol



Truth Table

A	B	C	X
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

Constructing Gates

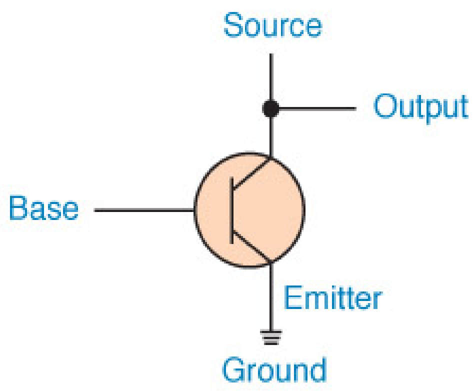


Figure 4.8 The connections of a transistor

A transistor has three terminals

- A source
- A base
- An emitter, typically connected to a ground wire

If the electrical signal is grounded, it is allowed to flow through an alternative route to the ground (literally) where it can do no harm

Constructing Gates

The easiest gates to create are the NOT, NAND, and NOR gates

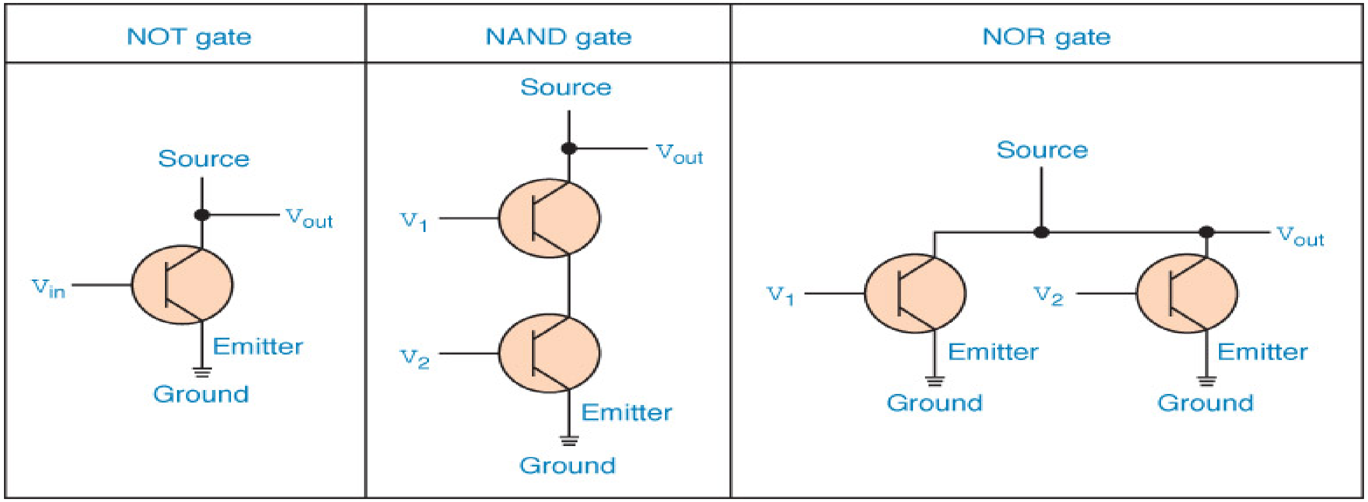


Figure 4.9 Constructing gates using transistors

Circuits

Combinational circuit

The input values explicitly determine the output

Sequential circuit

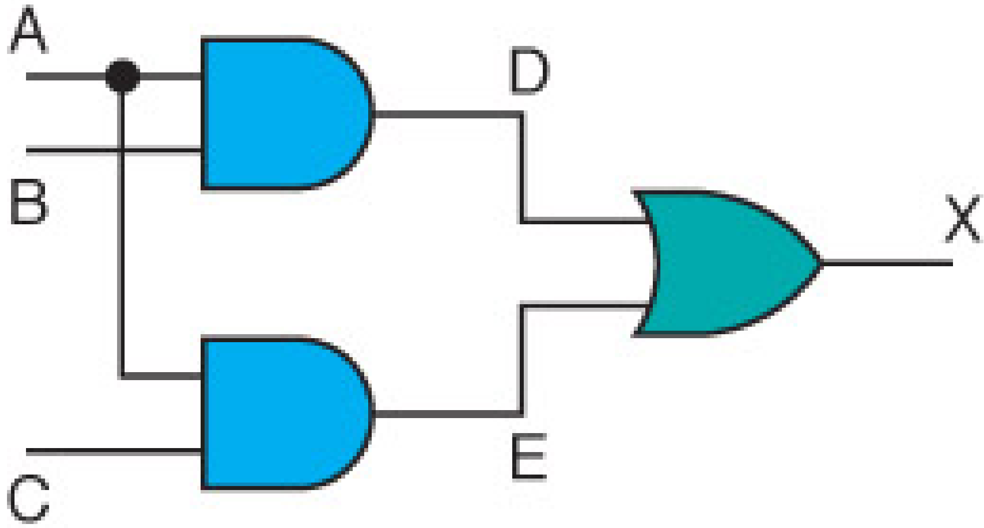
The output is a function of the input values and the existing state of the circuit

We describe the circuit operations using

- Boolean expressions
- Logic diagrams
- Truth tables

Combinational Circuits

Gates are combined into circuits by using the output of one gate as the input for another



Combinational Circuits

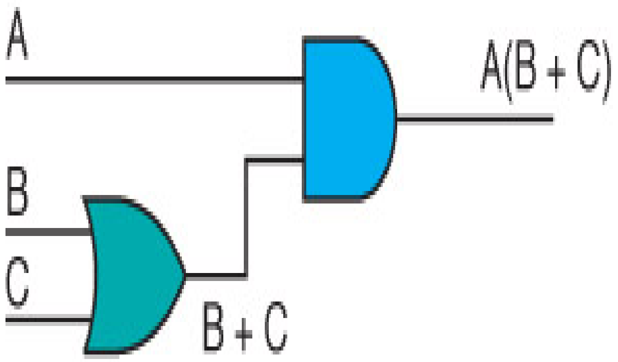
A	B	C	D	E	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Three inputs require eight rows to describe all possible input combinations

This same circuit using a Boolean expression is (AB + AC)

Combinational Circuits

Consider the following Boolean expression $A(B + C)$



A	B	C	B + C	A(B + C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Combinational Circuits

Circuit equivalence

Two circuits that produce the same output for identical input

Boolean algebra allows us to apply provable mathematical principles to help design circuits

A(B + C) = AB + BC (distributive law) so circuits must be equivalent

Properties of Boolean Algebra

Property	AND	OR
Commutative	AB = BA	A + B = B + A
Associative	(AB)C = A(BC)	(A + B) + C = A + (B + C)
Distributive	A(B + C) = (AB) + (AC)	A + (BC) = (A + B)(A + C)
Identity	A1 = A	A + 0 = A
Complement	A(A') = 0	A + (A') = 1
DeMorgan's law	(AB)' = A' OR B'	(A + B)' = A'B'

Adders

At the digital logic level, addition is performed in binary

Addition operations are carried out by special circuits called, appropriately, **adders**

Adders

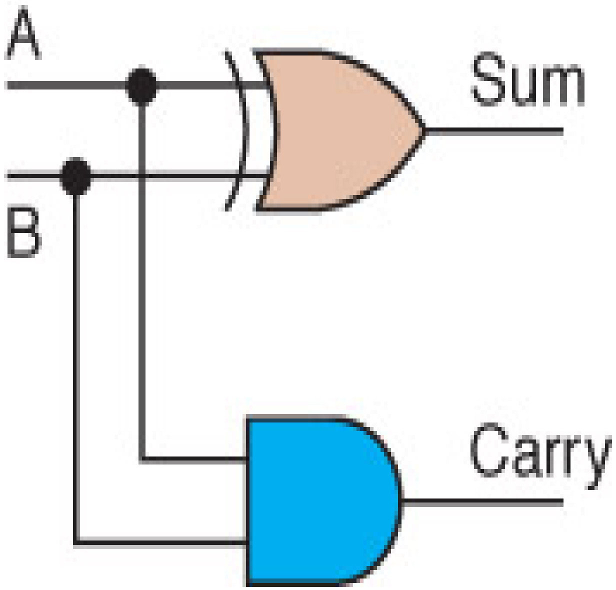
Half adder

A circuit that computes the sum of two bits and produces the correct carry bit

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

Truth table

Adders



Circuit diagram representing a half adder

Boolean expressions

$$sum = A \oplus B$$
$$carry = AB$$

Adders

Full adder

A circuit that takes the carry-in value into account

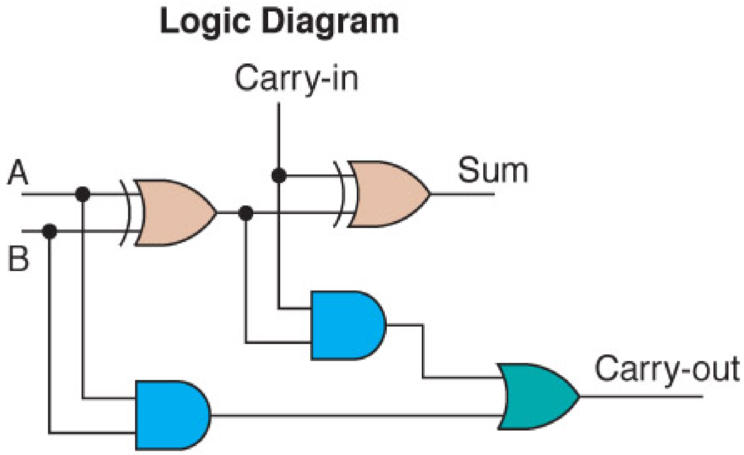


Figure 4.10 A full adder

Truth Table

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

Circuits as Memory

Digital circuits can be used to store information

These circuits form a sequential circuit, because the output of the circuit is also used as input to the circuit

Circuits as Memory

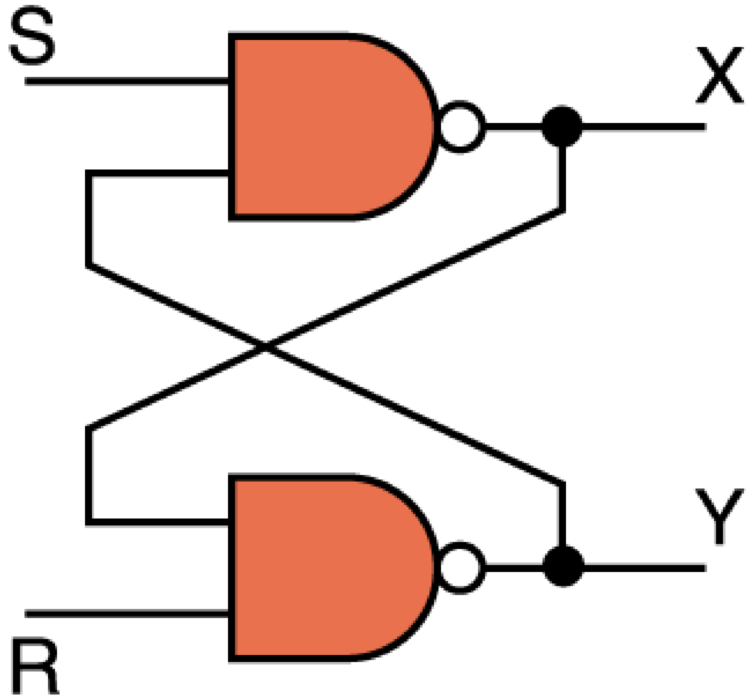


Figure 4.12 **An S-R latch**

An S-R latch stores a single binary digit (1 or 0)

There are several ways an S-R latch circuit can be designed using various kinds of gates

Circuits as Memory

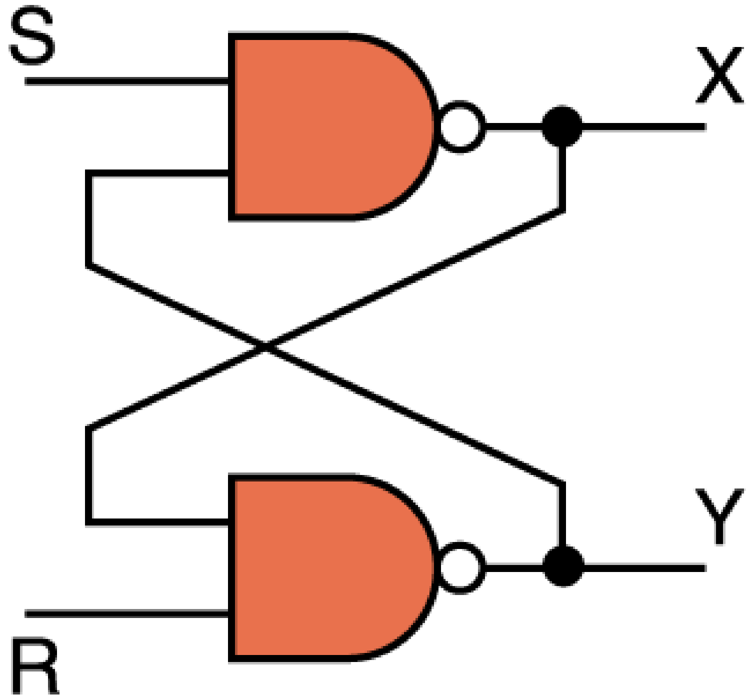
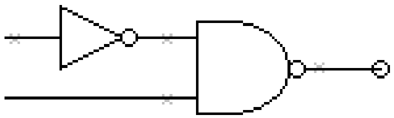


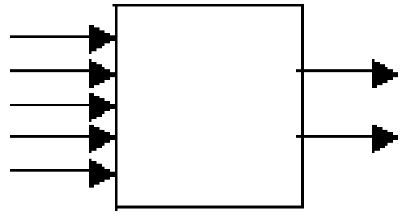
Figure 4.12 An S-R latch

The value of X at any point in time is considered to be the current state of the circuit

Therefore, if X is 1, the circuit is storing a 1; if X is 0, the circuit is storing a 0



Logic gates



Combinational logic circuits

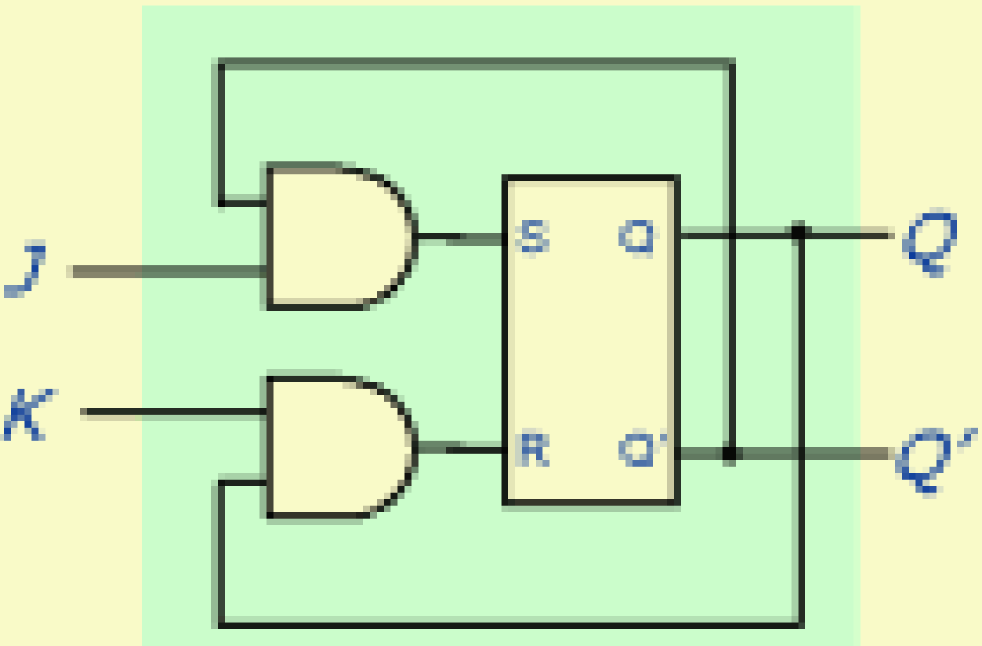
Sequential Logic

Acyclic connections
Composable blocks
Design:

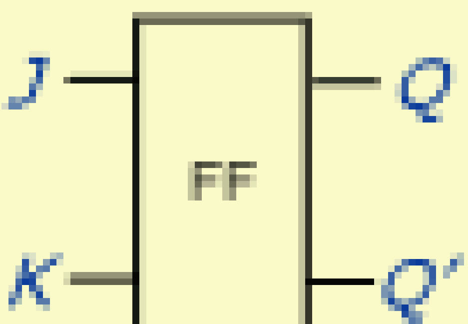
- truth tables
- sum-of-products
- simplification
- muxes, ROMs, PLAs

Storage & state
Dynamic discipline
Finite-state machines
Metastability
Throughput & latency
Pipelining

The JK Flip-Flop



J(t)	K(t)	Q(t)	Q(t+2δ)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$$Q(t+2\delta) = Q(t)K'(t) + Q'(t)J(t)$$

Integrated Circuits

Integrated circuit (also called a *chip*)

A piece of silicon on which multiple gates have been embedded

Silicon pieces are mounted on a plastic or ceramic package with pins along the edges that can be soldered onto circuit boards or inserted into appropriate sockets

Integrated Circuits

Integrated circuits (IC) are classified by the number of gates contained in them

Abbreviation	Name	Number of Gates
SSI	Small-Scale Integration	1 to 10
MSI	Medium-Scale Integration	10 to 100
LSI	Large-Scale Integration	100 to 100,000
VLSI	Very-Large-Scale Integration	more than 100,000

Integrated Circuits

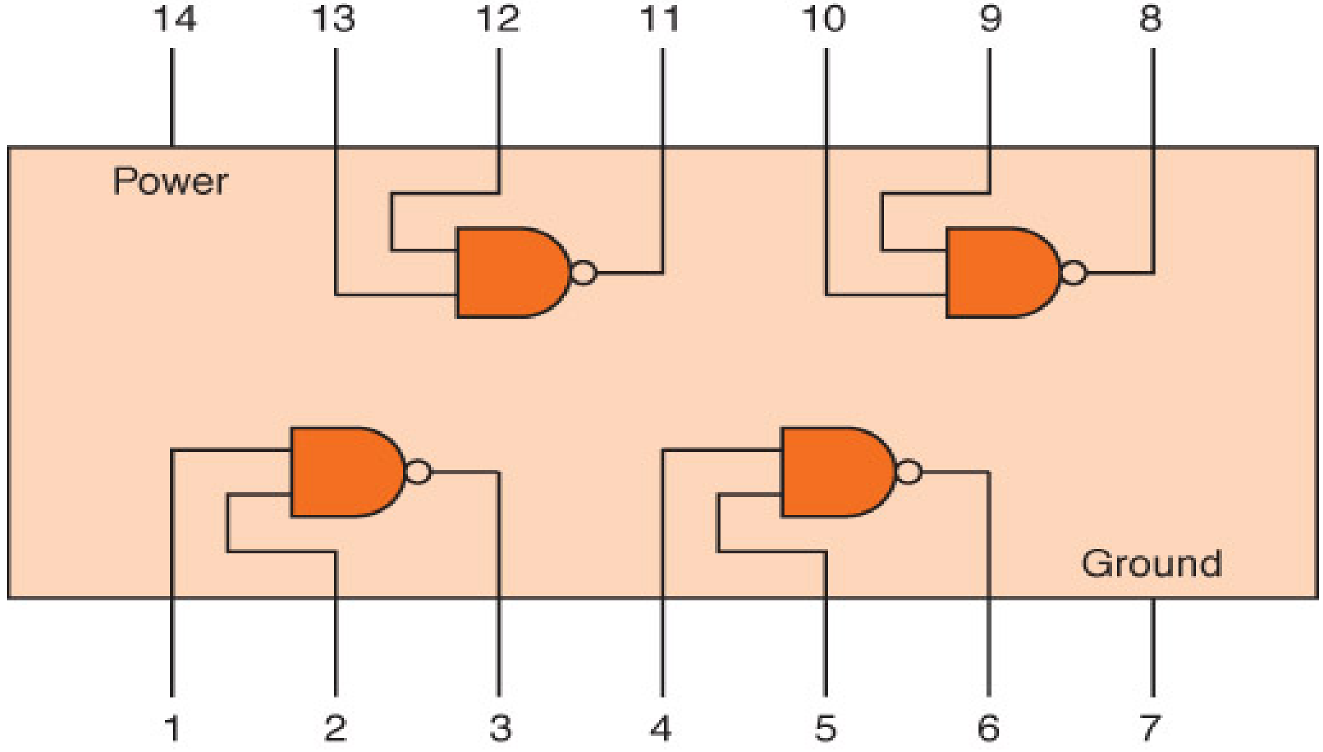


Figure 4.13 **An SSI chip contains independent NAND gates**

CPU Chips

The most important integrated circuit in any computer is the Central Processing Unit, or CPU

Each CPU chip has a large number of pins through which essentially all communication in a computer system occurs