GATES AND CIRCUITS

The design and construction of electrical circuits to implement logical and arithmetical operations.

CHAPTER 4

Chapter Goals

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

Chapter Goals

- Compare and contrast a half adder and a full adder;
- Explain how an S-R latch operates;
- Describe the characteristics of the four generations of integrated circuits.

Definitions

Two valued Logic (F/T, 0/1)
Allows formal decision on logical expressions

Gate
A device that performs a basic operation on electrical signals.

Circuits
Gates combined to perform more complicated tasks.

Two Valued Logic

Statements:

A = *My grandmother is alive*

B = *My grandfather is alive*

NOT A = *My grandmother is dead*

A AND B = *Both my grandparents are alive*

A OR B = *At least one of my grandparents is alive*

Two Valued Logic

Statements:

A = *My grandmother is alive*

B = *My grandfather is alive*

A XOR B = *Exactly one of my grandparents is alive*

A NAND B = *My grandparents are not both alive*

A NOR B = *My grandparents are both dead*

Describing 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 values and the associated output values.

Gates

Six types of gates

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

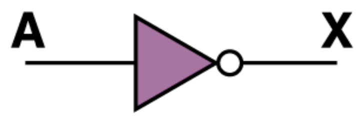
NOT Gate

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

Figure 4.1 Various representations of a NOT gate

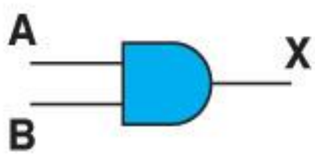
AND Gate

- 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

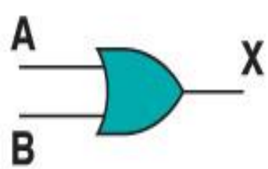
Figure 4.2 Various representations of an AND gate

OR Gate

An OR gate accepts two input signals

If both are 0, the output is 0;

otherwise, the output is 1

Boolean Expression	Logic Diagram Symbol
$X = A + B$	

Truth Table

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

Figure 4.3 Various representations of a OR gate

XOR Gate

An XOR gate accepts two input signals

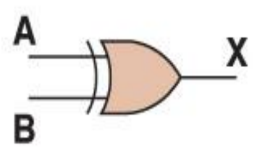
If both are the same, the output is 0;

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

Figure 4.4 Various representations of an XOR gate

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*

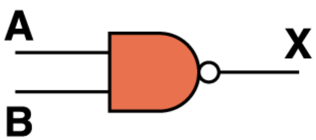
NAND Gate

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

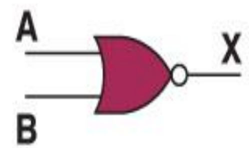
The NOR gate accepts two input signals

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

Figure 4.6 Various representations of a NOR gate

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

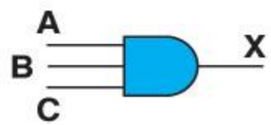
Gates with More Inputs

- 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

Figure 4.7 Various representations of a three-input AND gate

Constructing Gates

Transistor

A device that acts either as a wire that conducts electricity or as a resistor that blocks the flow of electricity, depending on the voltage level of an input signal

A transistor has no moving parts, yet acts like a switch

It is made of a semiconductor material, which is neither a particularly good conductor of electricity nor a particularly good insulator

Constructing Gates

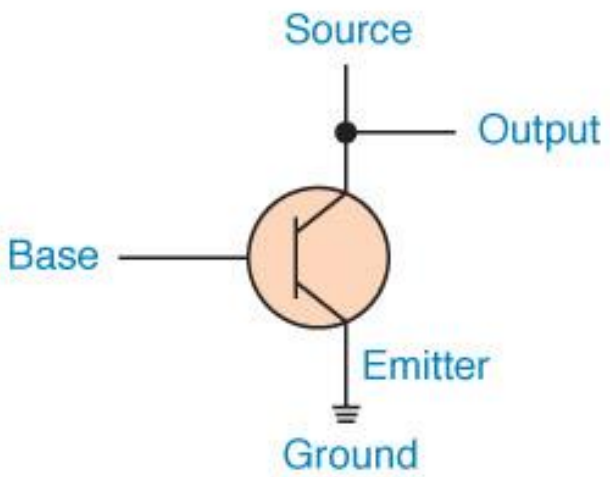


Figure 4.8 The connections of a transistor

A transistor has three terminals
- A source
- A base
- An emitter, typically connected to a ground wire

If the Base is “on”, then the transistor Emits, and is Earthed.

If the Base is “off”, then the transistor becomes an insulator and the Output is on.

Constructing Gates

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

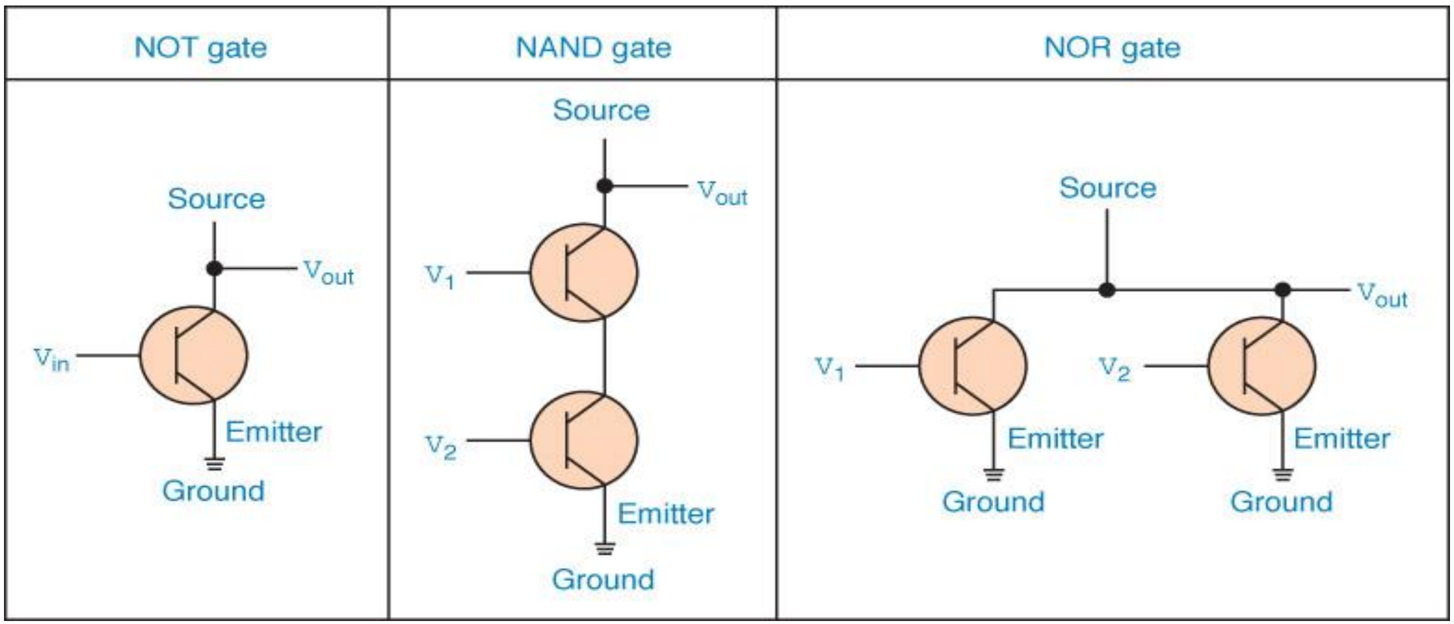


Figure 4.9 Constructing gates using transistors

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

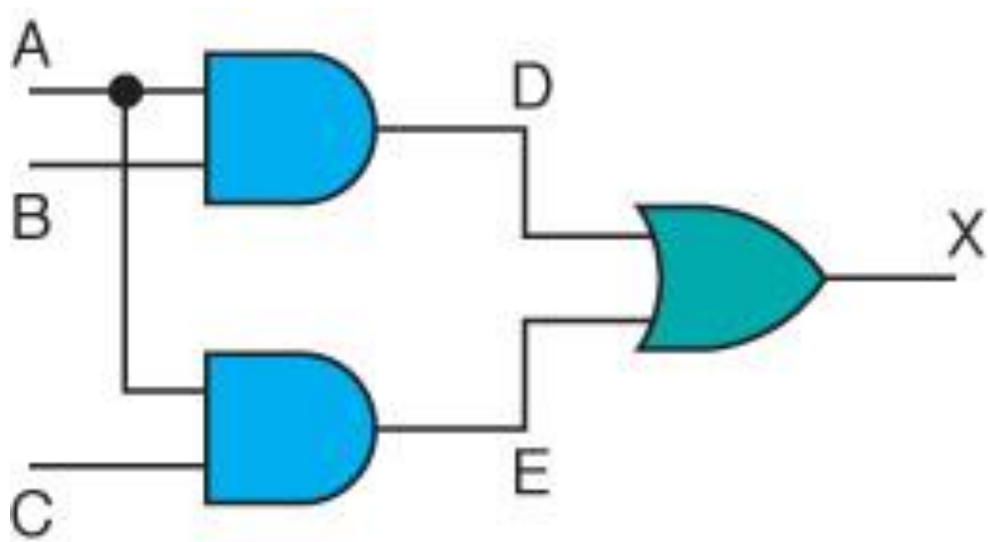
We describe the circuit operations using

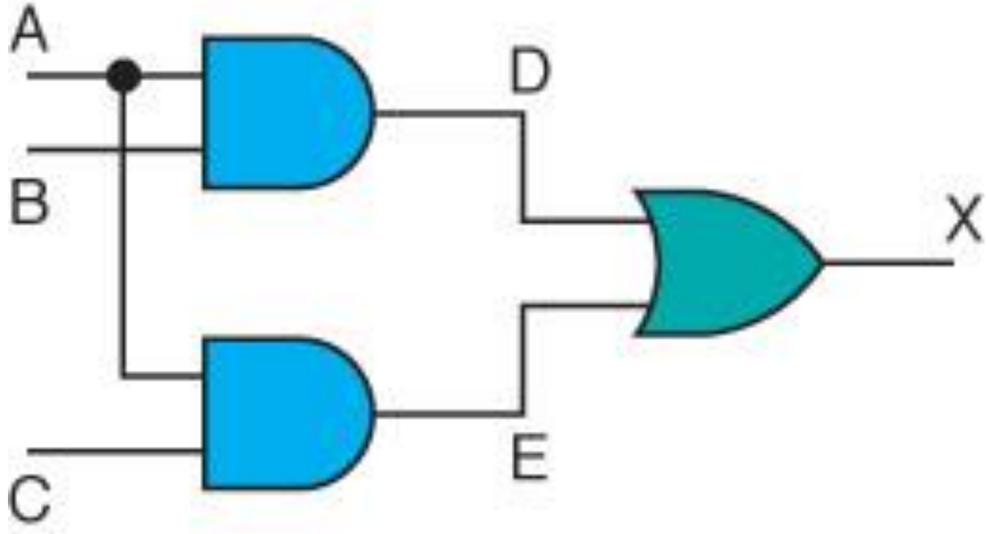
- Boolean expressions
- Logic diagrams
- Truth tables

Combinational Circuits

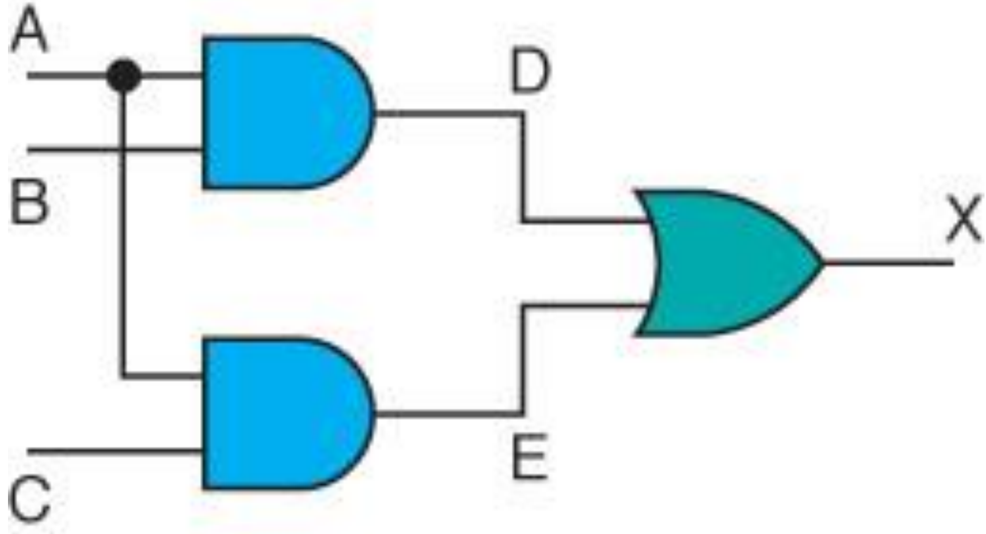
Gates are combined into circuits by using the output of one gate as the input for another

(A.B) + (A.C)

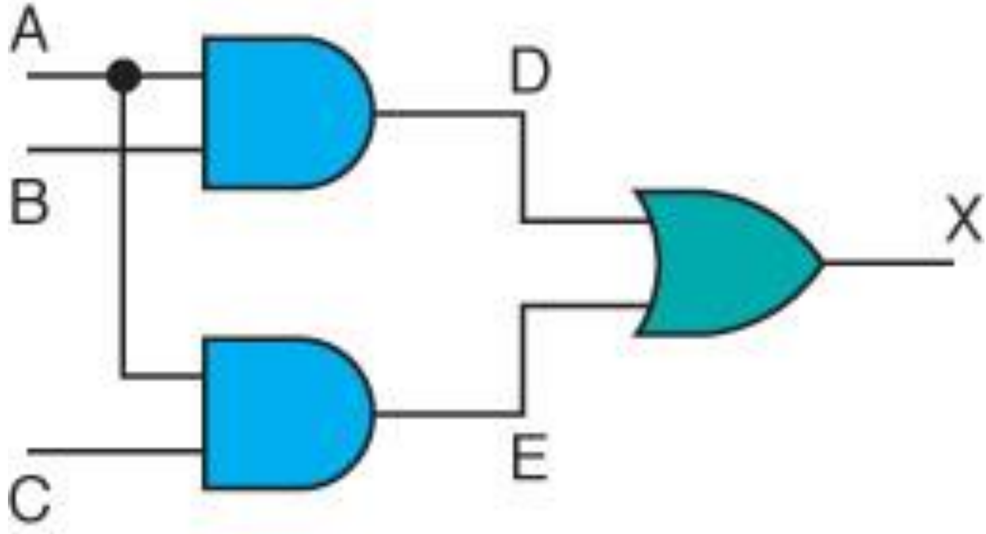




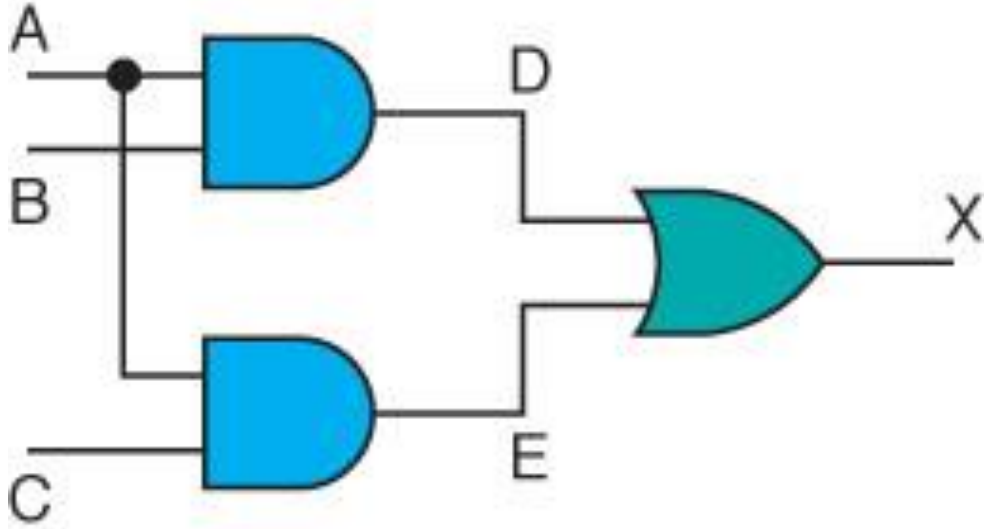
A	B	C	D	E	X
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			



A	B	C	D	E	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	1		
1	1	1	1		



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

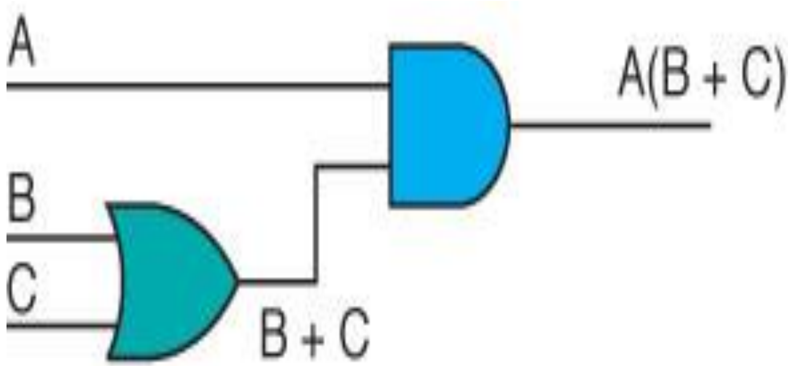


A	B	C	D	E	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

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

$A.(B + C) = A.B + A.C$ (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	$(AB)' = A' + B'$	$(A + B)' = A'B'$

Proving Boolean Relations

We can prove these rules by setting out all the values in a truth table and showing that the truth table for each side is the same.

For example take the Distributive Law:

$A+(BC) = (A+B).(A+C)$

$A+(BC) = (A+B).(A+C)$

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

A+(BC) = (A+B).(A+C)

A	B	C	A+B	A+C	(A+B).(A+C)	BC	A+(BC)
0	0	0	0				
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				

A+(BC) = (A+B).(A+C)

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

A+(BC) = (A+B).(A+C)

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

A+(BC) = (A+B).(A+C)

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

$A+(BC) = (A+B).(A+C)$

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

A+(BC) = (A+B).(A+C)

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

De Morgan’s Law (A+B)’=A’B’

A	B	A+B	(A+B)'	A'	B'	A'B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

De Morgan’s Law (A+B)’=A’B’

A	B	A+B	(A+B)'	A'	B'	A'B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Adders

At the digital logic level, addition is performed in binary

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

Adders

The result of adding two binary digits could produce a *carry value*

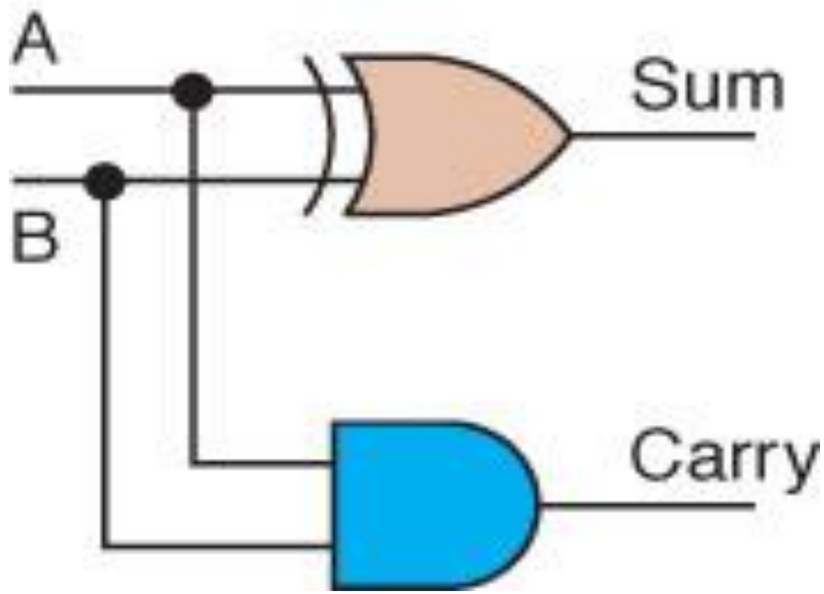
Recall that 1 + 1 = 10 in base two

Half adder

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

Half Adder

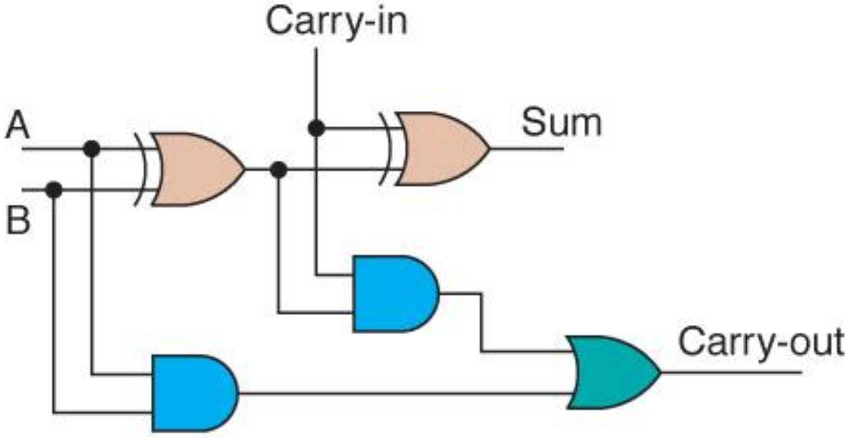
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$$\text{sum} = A \oplus B$$
$$\text{carry} = AB$$

Full Adder

Logic Diagram



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

Figure 4.10 A full adder

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

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

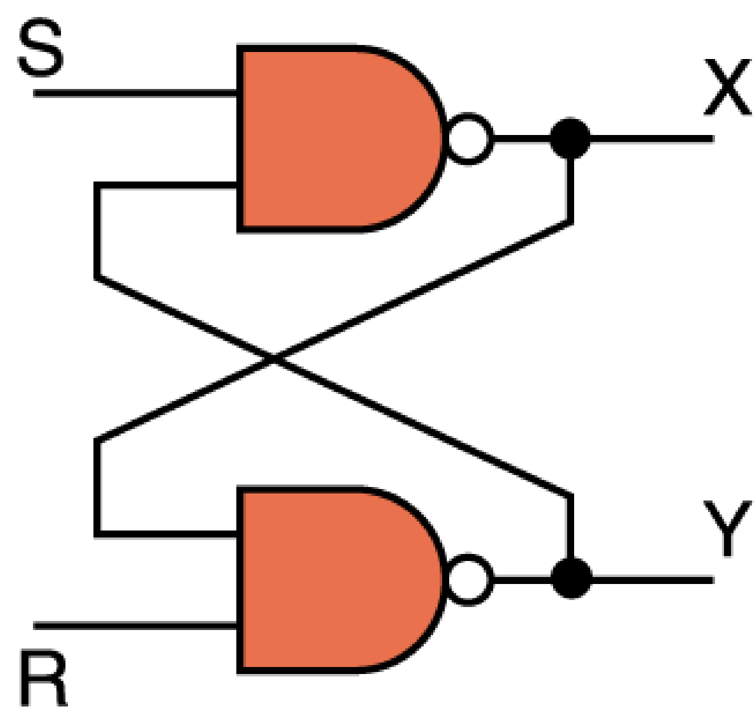


Figure 4.12 An S-R latch

Circuits as Memory

- The design of this circuit guarantees that the two outputs X and Y are always complements of each other
- 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

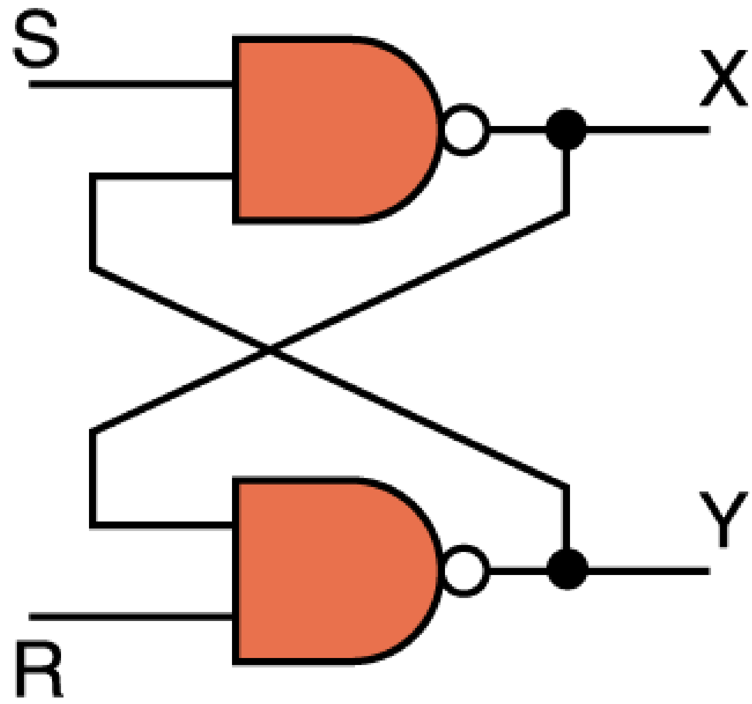
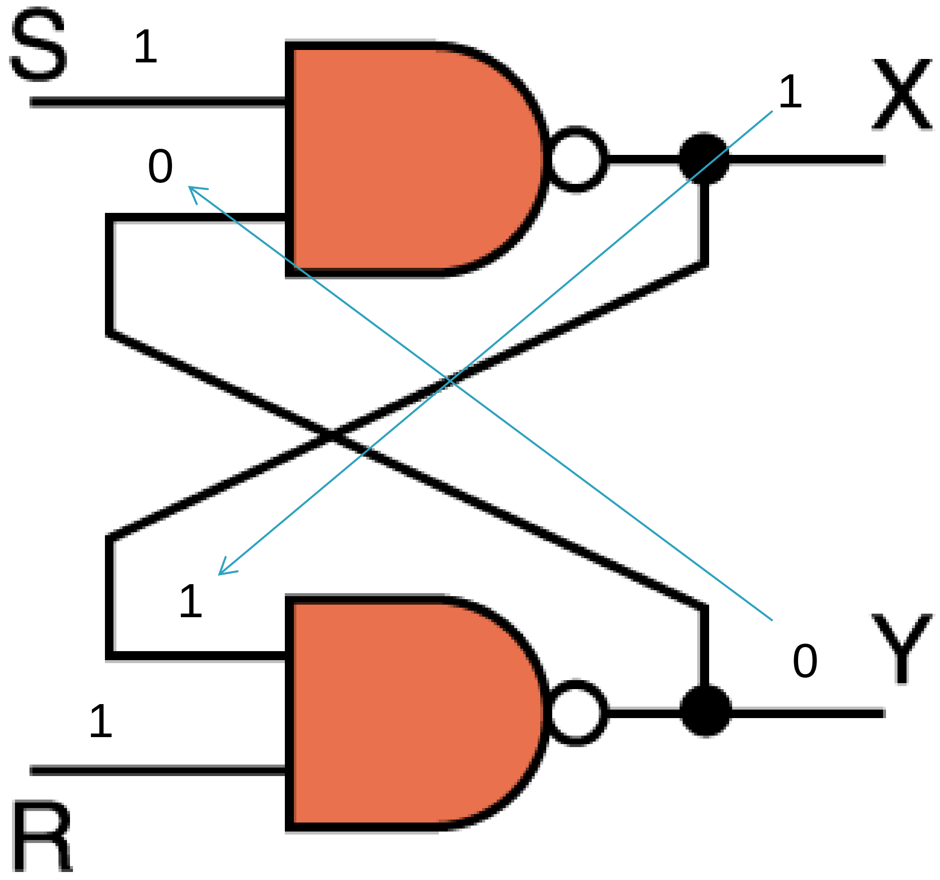
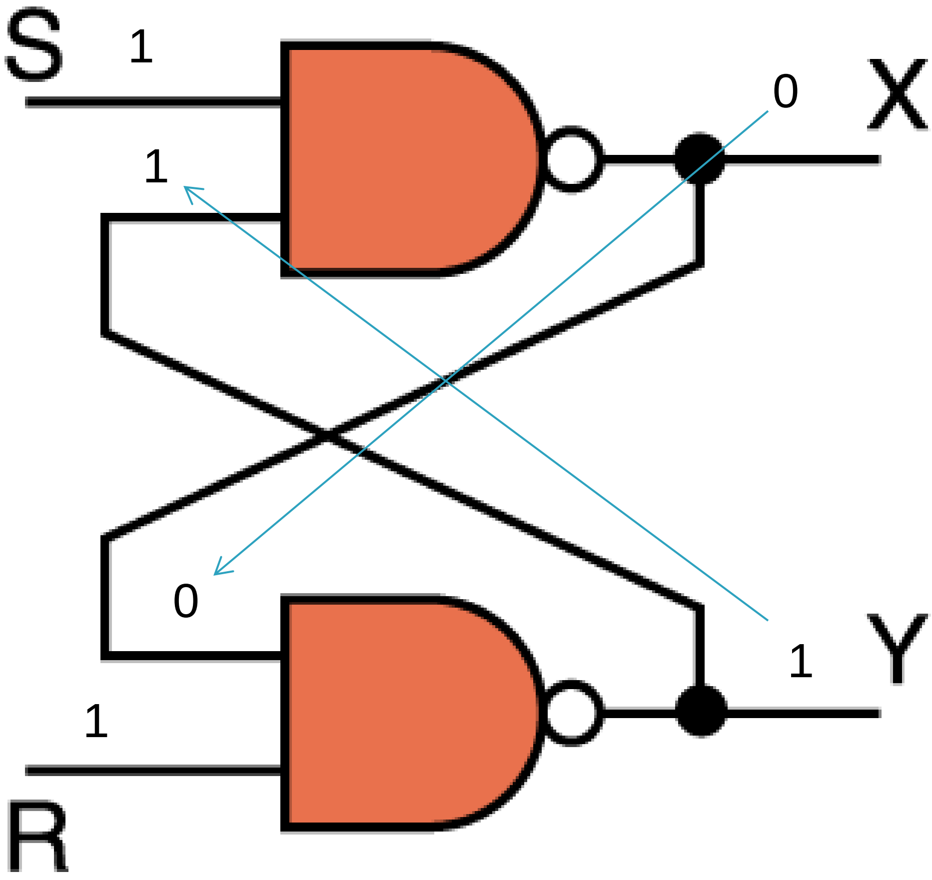


Figure 4.12 An S-R latch

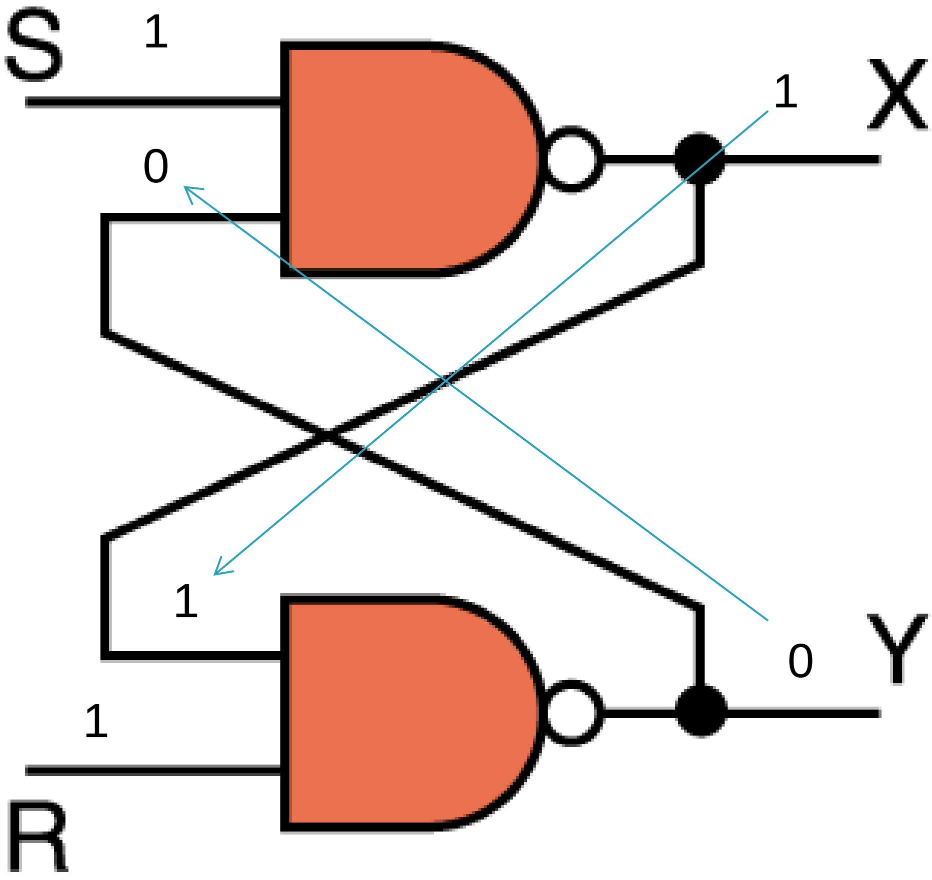
Storing a 1



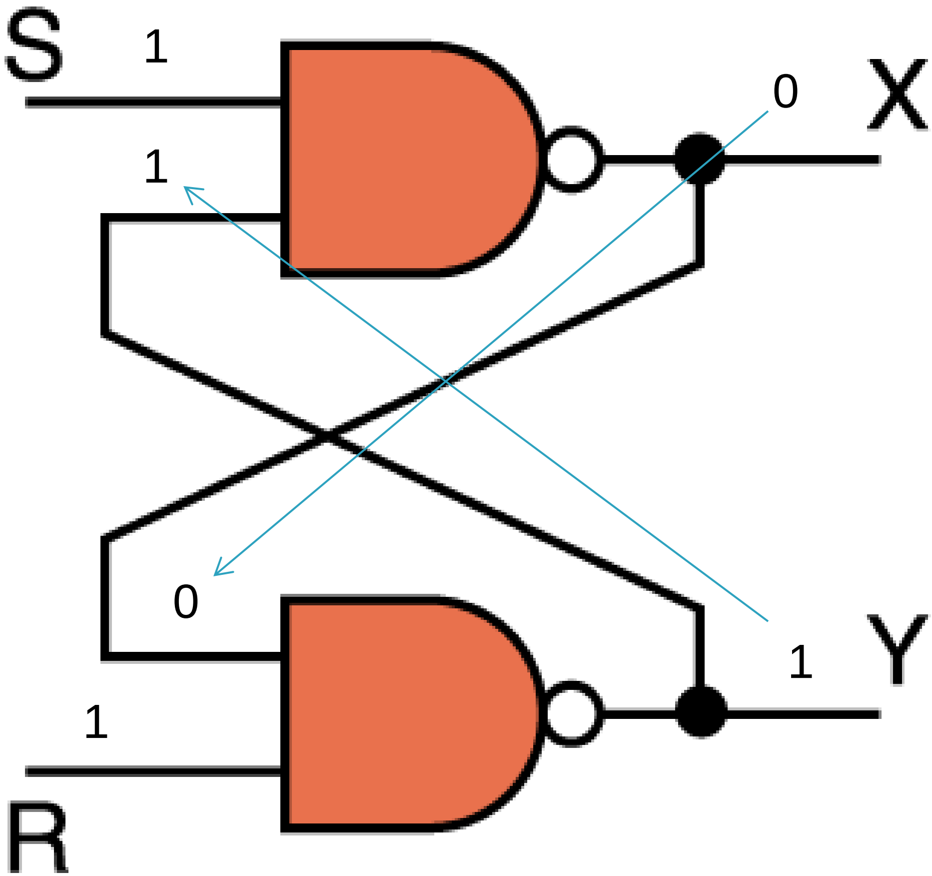
Storing a 0



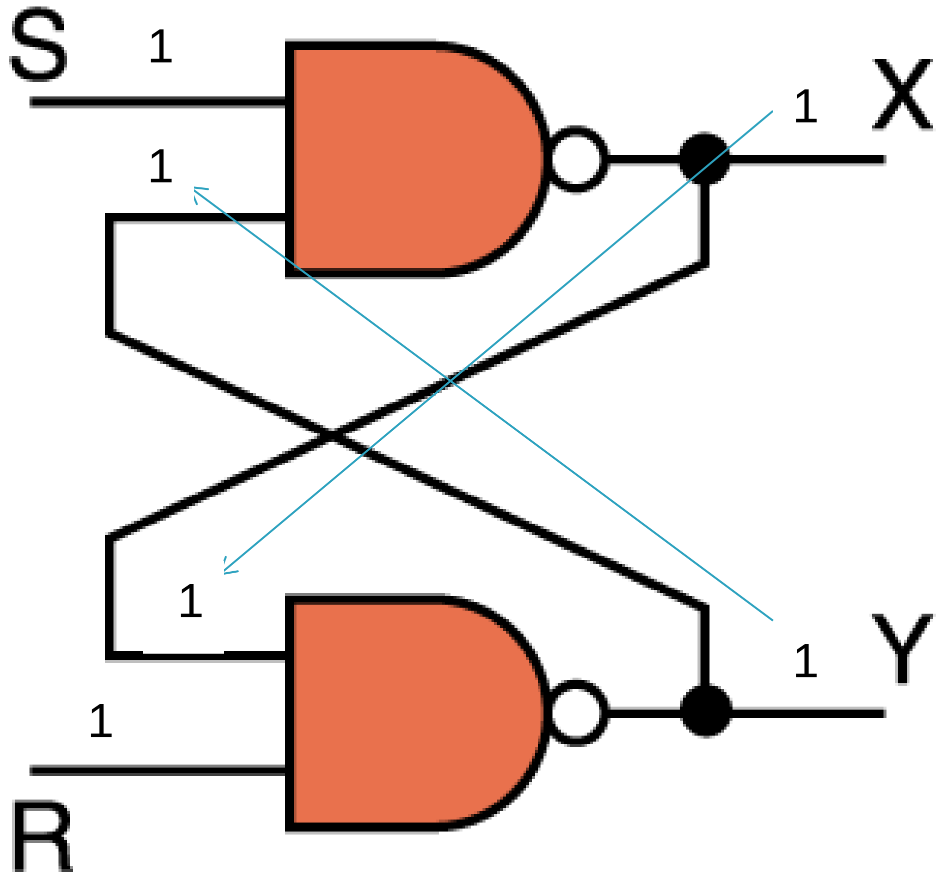
Writing a 1



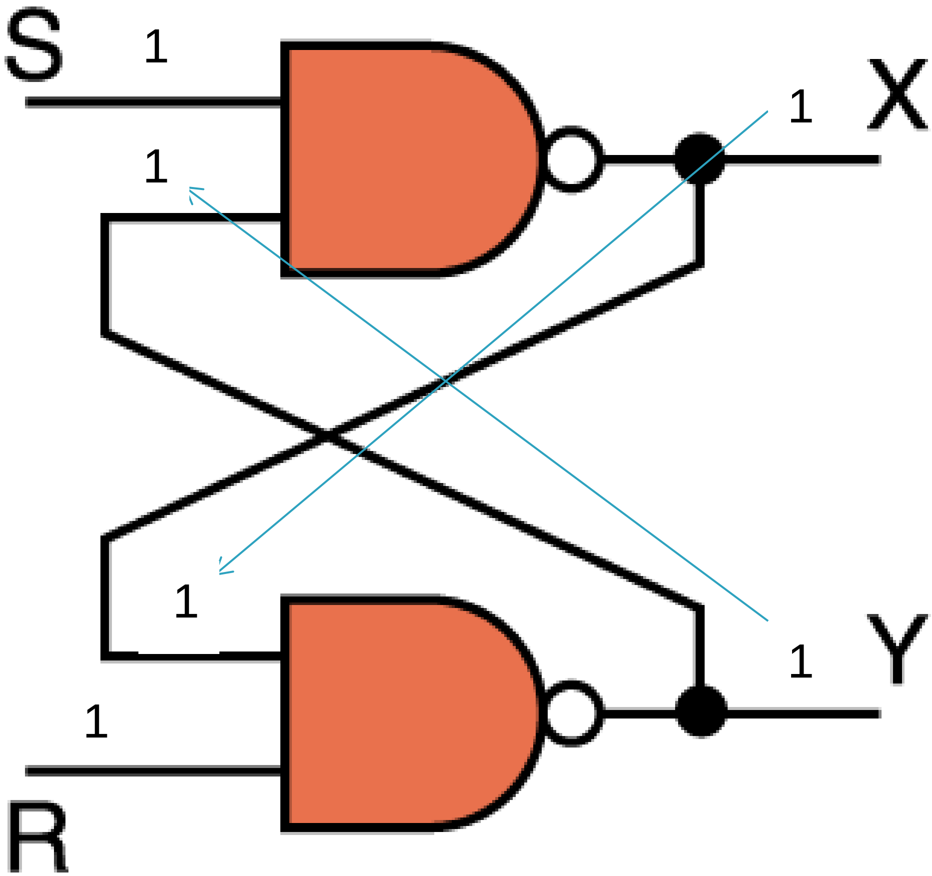
Writing a 0



But…



But…



Disjunctive Normal Form (DNF)

- It is straightforward to obtain a truth table from a circuit or Boolean Expression.
- It is also straightforward to obtain a circuit from a Boolean expression or conversely.
- It is not so obvious how to construct a Boolean expression (or circuit) from a complicated truth table.
- This can be done using DNF.

Disjunctive Normal Form (DNF)

1. Move down the truth table selecting all the rows where the output value is True (1);
2. Select a row;
3. Use the appropriate inputs or their negations to form the corresponding <u>conjunction</u> (this means that for each input A, say, use A if its value in the row is 1 or use A’ if its value in the row is 0, and combine them using AND);
4. Combine the conjuctions using OR.

Example

A	B	C	X		
0	0	0	0		
0	0	1	0		
0	1	0	0		
0	1	1	1	x	A’BC
1	0	0	0		
1	0	1	0		
1	1	0	1	x	ABC’
1	1	1	1	x	ABC

A’BC+ABC’+ABC

Notice that the three chosen rows give 1 in the expression by construction.

Notice that any input of A, B, C EXCEPT
the three chosen rows will give 0 in the expression.

Example

A	B	C	X		
0	0	0	0		
0	0	1	0		
0	1	0	0		
0	1	1	1	x	A’BC
1	0	0	0		
1	0	1	0		
1	1	0	1	x	ABC’
1	1	1	1	x	ABC

A’BC+ABC’+ABC = A’BC+AB(C’+C) = A’BC+AB = B(A’C+A) = B(A+C)

Full Adder

A	B	Carry in	Sum	Carry out		
0	0	0	0	0		
0	0	1	1	0	x	A’B’C
0	1	0	1	0	x	A’BC’
0	1	1	0	1		
1	0	0	1	0	x	AB’C’
1	0	1	0	1		
1	1	0	0	1		
1	1	1	1	1	x	ABC

$Sum = A'B'C + A'BC' + AB'C' + ABC$

$= (A'B' + AB)C + (A'B + AB')C'$

$= (A \oplus B)'C + (A \oplus B)C' = (A \oplus B) \oplus C$

Full Adder

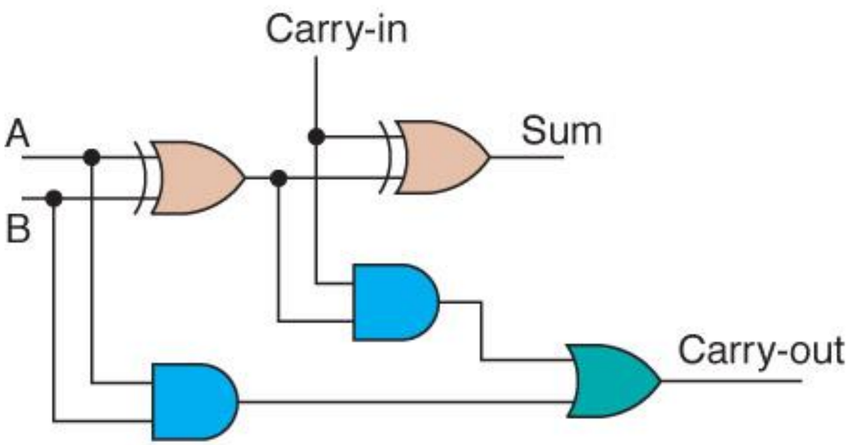
A	B	Carry in	Sum	Carry out		
0	0	0	0	0		
0	0	1	1	0		
0	1	0	1	0		
0	1	1	0	1	x	A’BC
1	0	0	1	0		
1	0	1	0	1	x	AB’C
1	1	0	0	1	x	ABC’
1	1	1	1	1	x	ABC

Carry out = A’BC + AB’C + ABC’ + ABC = (A’B+AB’)C + AB(C+C’)

$= (A \oplus B)\ C + AB$

Full Adder

Logic Diagram



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

Sum = $(A \oplus B) \oplus C$

Carry out = $(A \oplus B)\, C + AB$

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

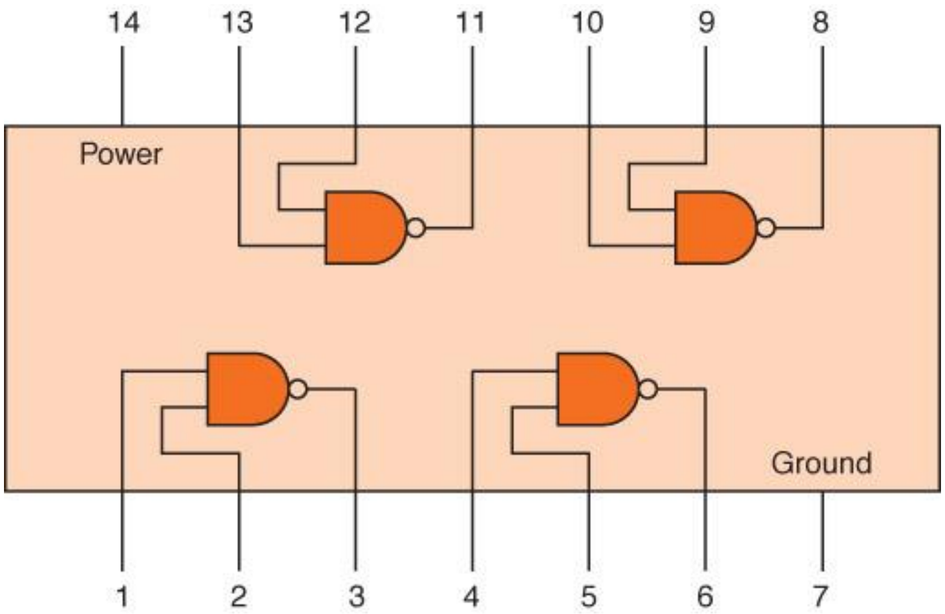


Figure 4.13 An SSI chip contains independent NAND gates

Integrated Circuits

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