



Course:
Mathematics for IT
Professionals



Lecture 3

Propositional Logic

By

Solomon Mensah



Outline

The topics to be treated in this lecture are:

- Logic
- Logical Operation
 - Negation
 - Conjunction
 - Disjunction
 - Implication
 - Bi-implication
- Precedence of Logical Operators
- Truth tables
- Bitwise Operations
- Applications
- Propositional Equivalence



Lecture Learning Outcomes

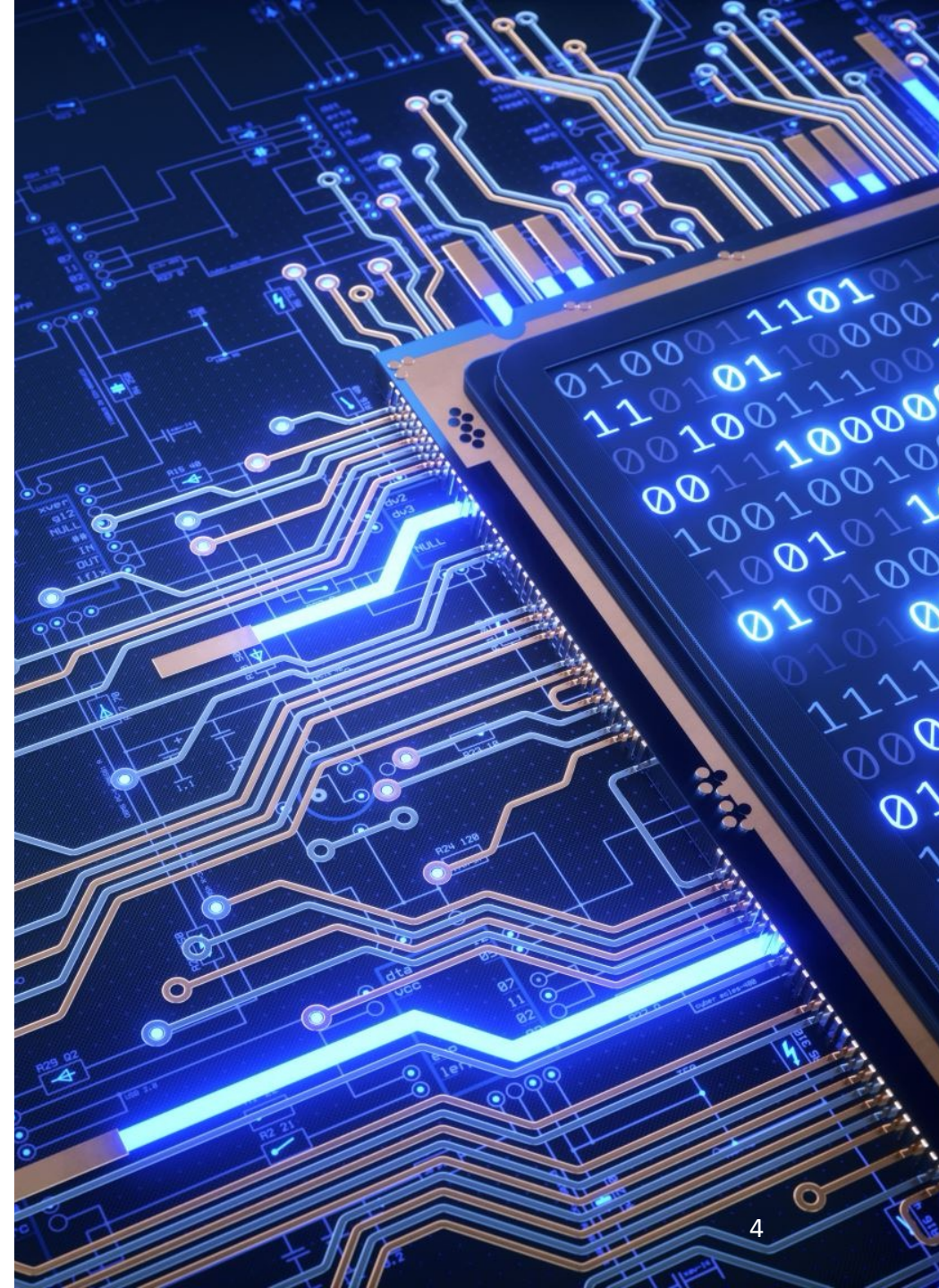
At the end of the session, you will be able to

- explain propositional logic
- determine the truth values of propositions
- understand the forms of logical operators
- construct truth tables based the logical operators
- perform bitwise operations
- have knowledge on applications of propositional logic

Introduction

- The rules of logic give precise meaning to mathematical statements
- Logic rules are used to distinguish between valid and invalid mathematical arguments
- Logic has numerous applications to computer science:
 - Design of computer circuits
 - Construction of computer programs
 - Verification of the correctness of a program
- Two types of logic:
 - Propositional logic
 - Predicate logic

Rosen, K. H. (2012). *Discrete mathematics and its applications* (7th Edition). McGraw-Hill.



What is a Proposition?

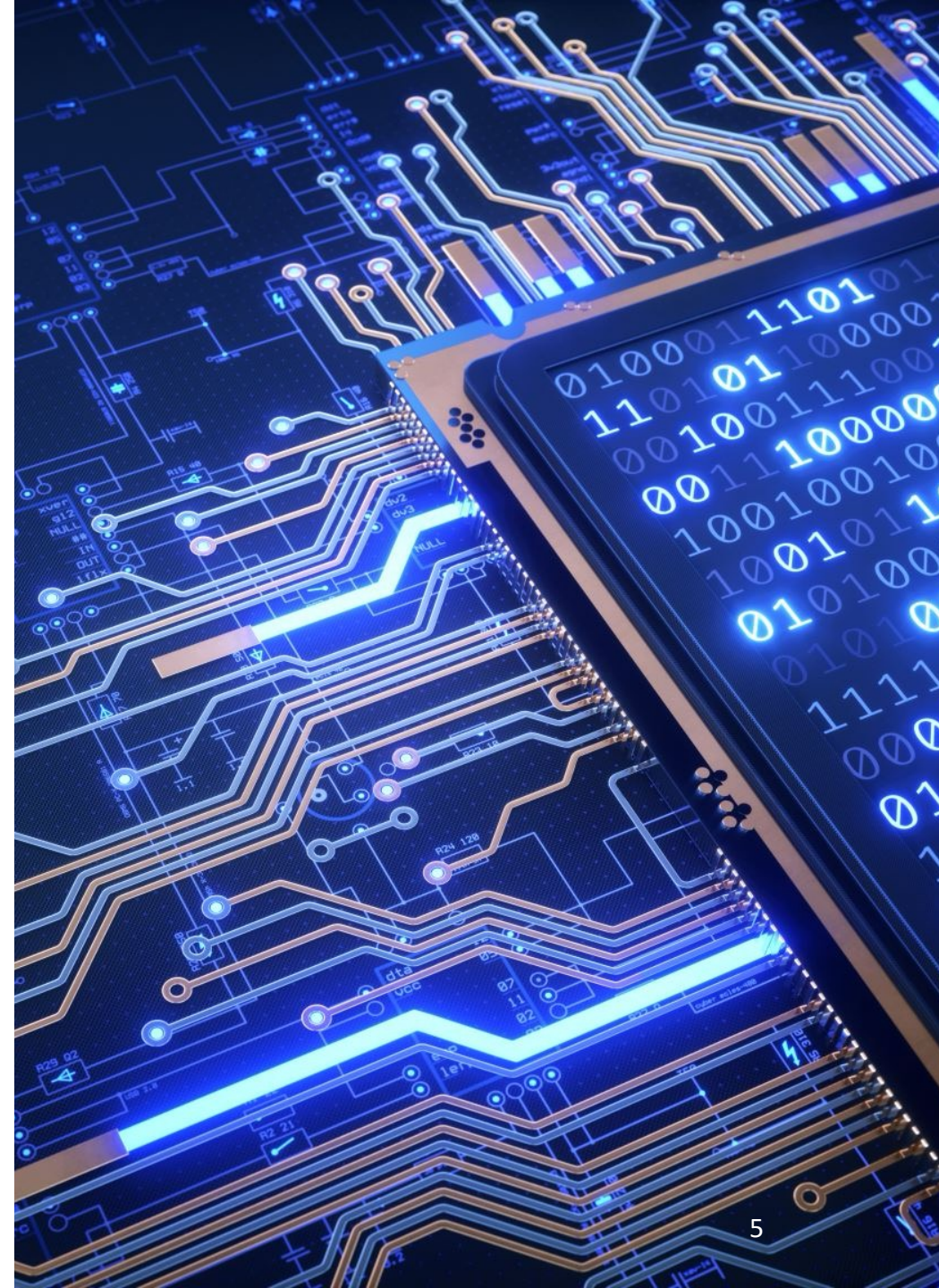
- A **proposition** is a declarative sentence that is either TRUE or FALSE, but not both
- Examples - **propositions**:

1. Washington, D.C., is the capital of the USA (TRUE)
2. Kumasi is the capital of Ghana (FALSE)
3. $1 + 1 = 2$ (TRUE)
4. $2 + 2 = 3$ (FALSE)

- Examples - **not propositions**:

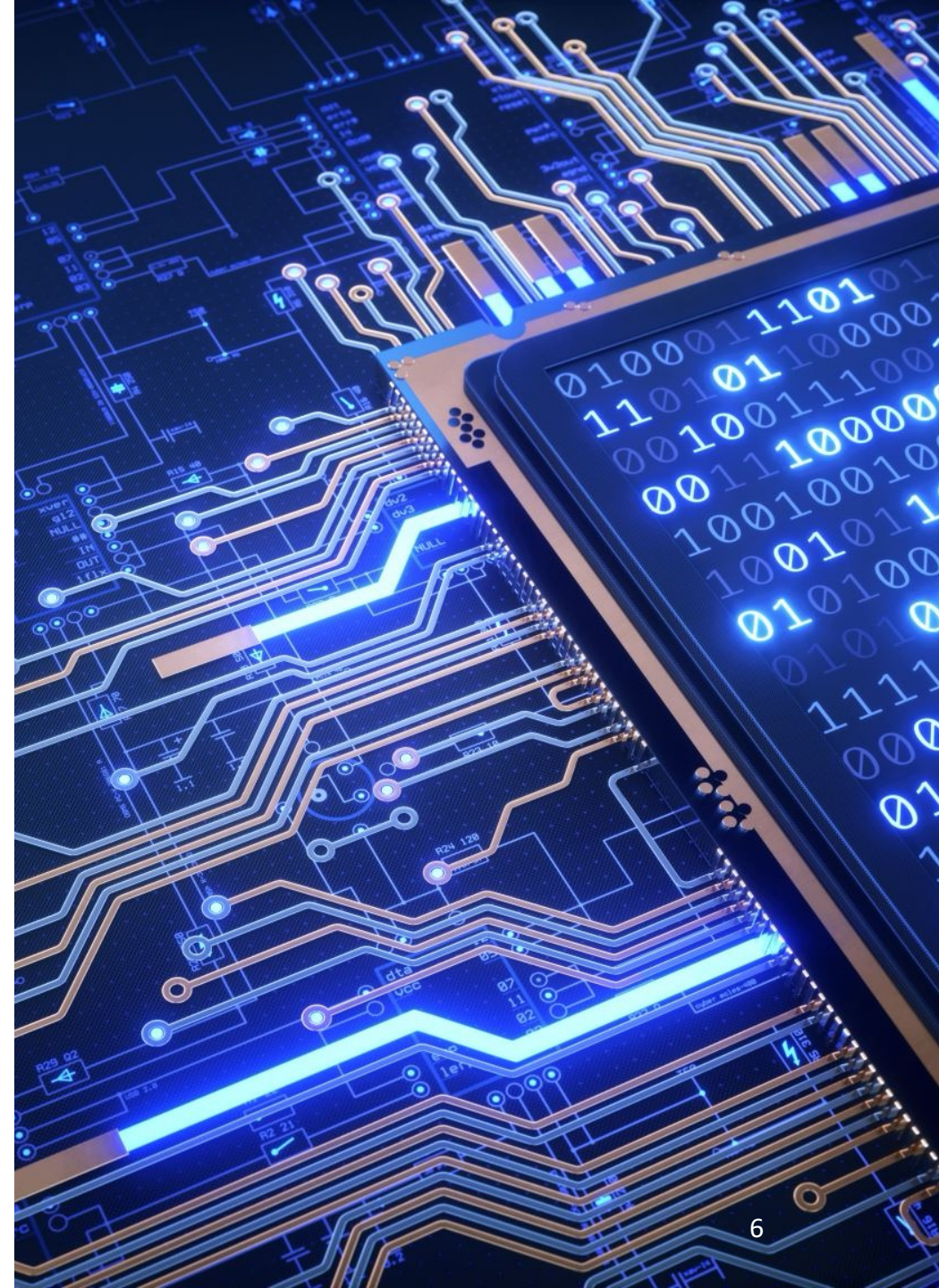
1. What time is it? (Not a Declarative Sentence)
2. Read this carefully (Not a Declarative Sentence)
3. $x + 1 = 2$ (Neither TRUE nor FALSE)
4. $x + y = z$ (Neither TRUE nor FALSE)

Rosen, K. H. (2012). *Discrete mathematics and its applications* (7th Edition). McGraw-Hill.



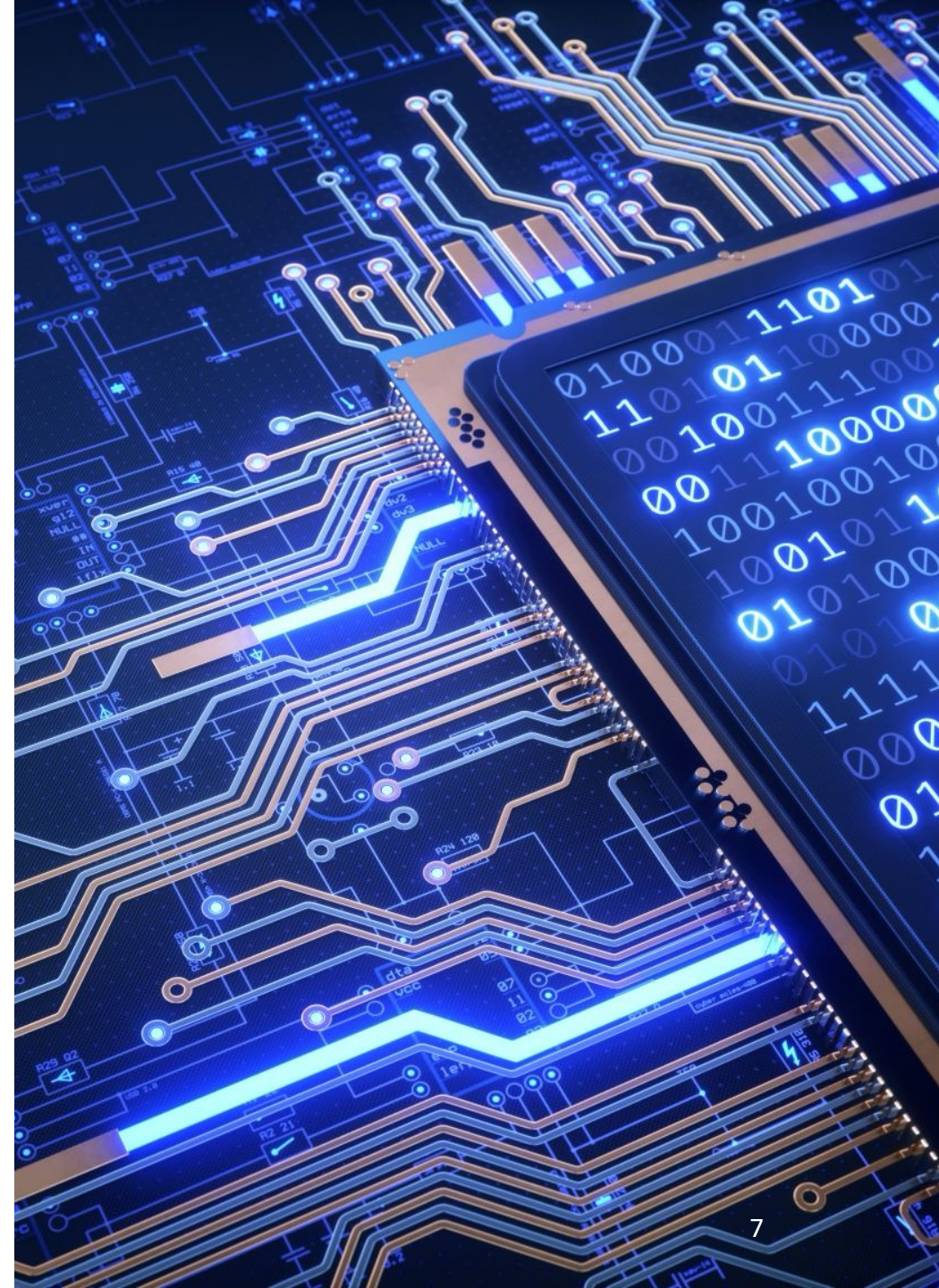
Preliminaries on Propositional Logic

- The area of the logic that deals with propositions is called *propositional logic* or *propositional calculus*
- **Notation:**
 - Propositional variables are denoted conventionally by the letters p, q, r, s, \dots
 - If the truth value of a proposition is TRUE, it is denoted by T
 - If the truth value of a proposition is FALSE, it is denoted by F



Compound Propositions

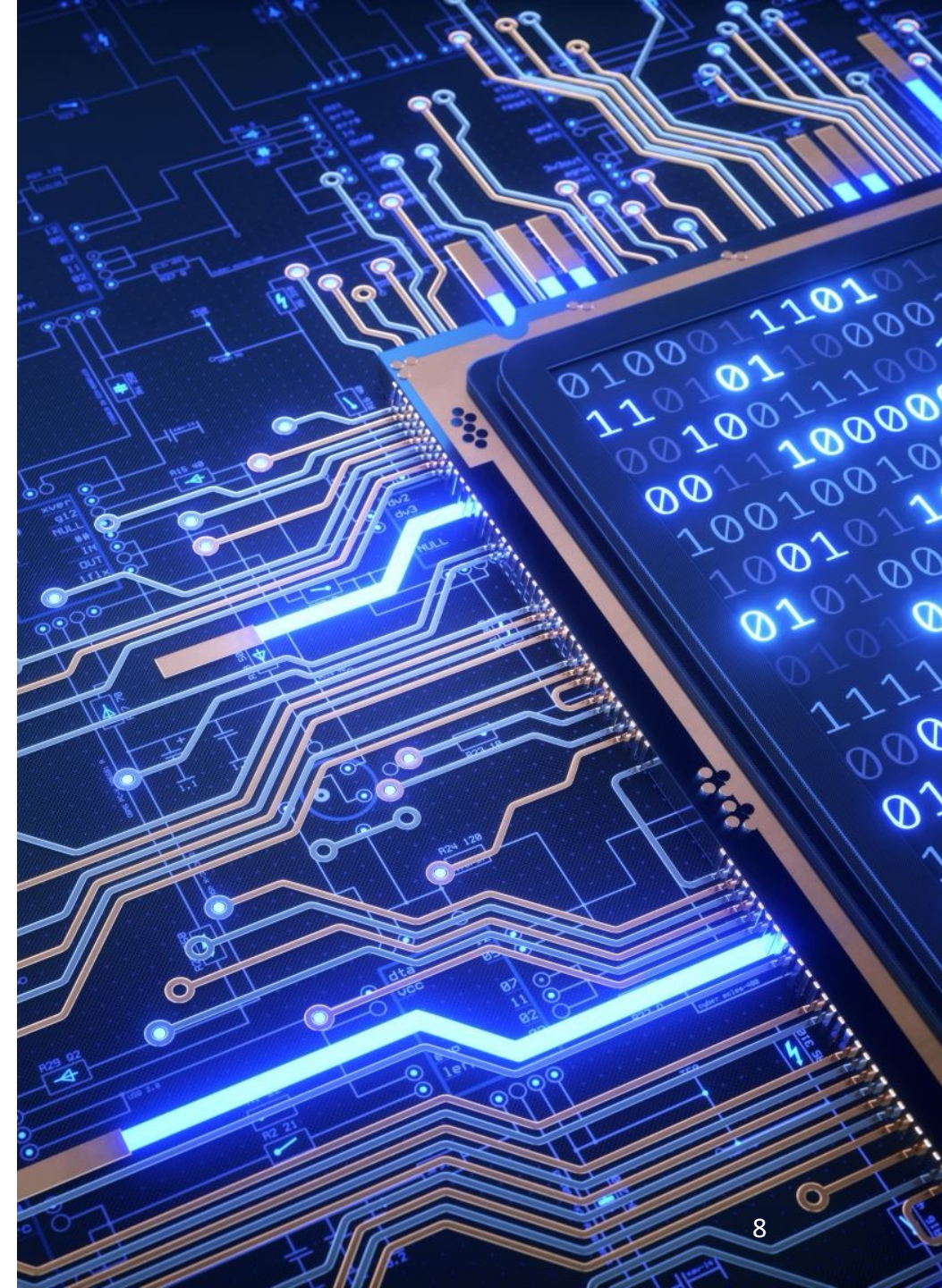
- These are the declarative statements that are constructed by combining one or more existing propositions using *logical operators*.
- The logical operators are:
 - Negation (NOT) denoted by the symbol, \neg
 - Conjunction (AND) denoted by the symbol, \wedge
 - Disjunction (OR) denoted by the symbol, \vee



Negation

- The *negation* of a proposition p , denoted by $\neg p$ is the statement “It is not the case that p ”, and is read as “**not** p .”
- The truth value of $\neg p$, is the opposite of the truth value of p .
- Example:
 p = Michael’s PC runs Linux
 $\neg p$ = It is not the case that Michael’s PC runs Linux
or
Michael’s PC does not run Linux
- Negation is applicable to only single proposition

Truth table for Negation	
P	$\neg p$
T	F
F	T



Conjunction

- The *conjunction* of two propositions p , q , denoted by $p \wedge q$ is the proposition “ **p and q** ”.
- The truth value of $p \wedge q$ is TRUE only when p , q are both TRUE and otherwise it is FALSE
- Conjunction is applicable to two or more propositions

- **Example:**

p = Kwame can read well

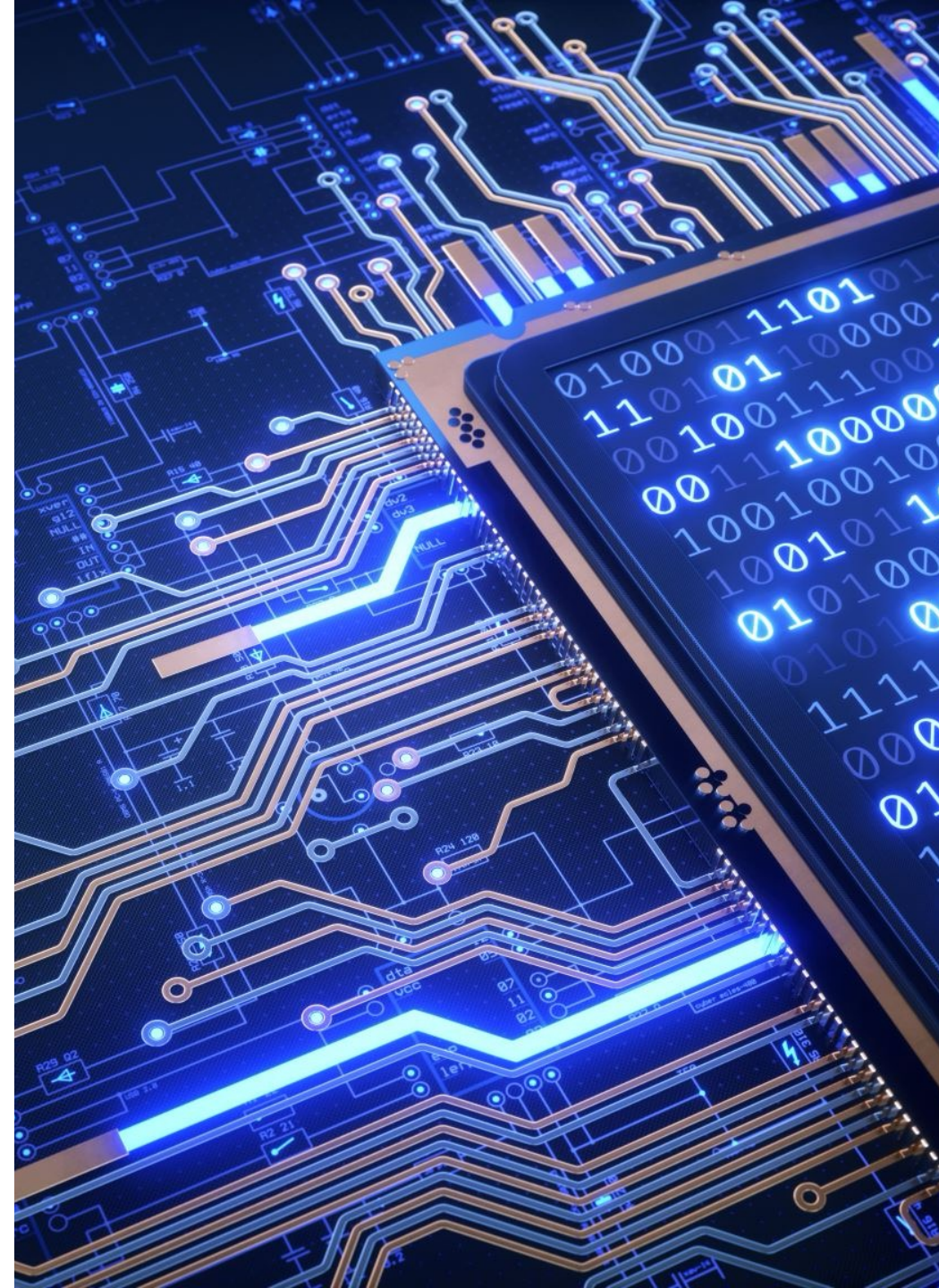
q = Kwame can write well

$p \wedge q$ = Kwame can read well and Kwame can write well

OR

Kwame can read and write well

Truth table for Conjunction		
P	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F



Disjunction

- The *disjunction* of two propositions p , q , denoted by $p \vee q$ is the proposition “ **p or q** ”.
- The truth value of $p \vee q$ is FALSE only when p , q are both FALSE and otherwise it is TRUE
- Disjunction is applicable to two or more propositions

- Example:

p = Kwame can read well

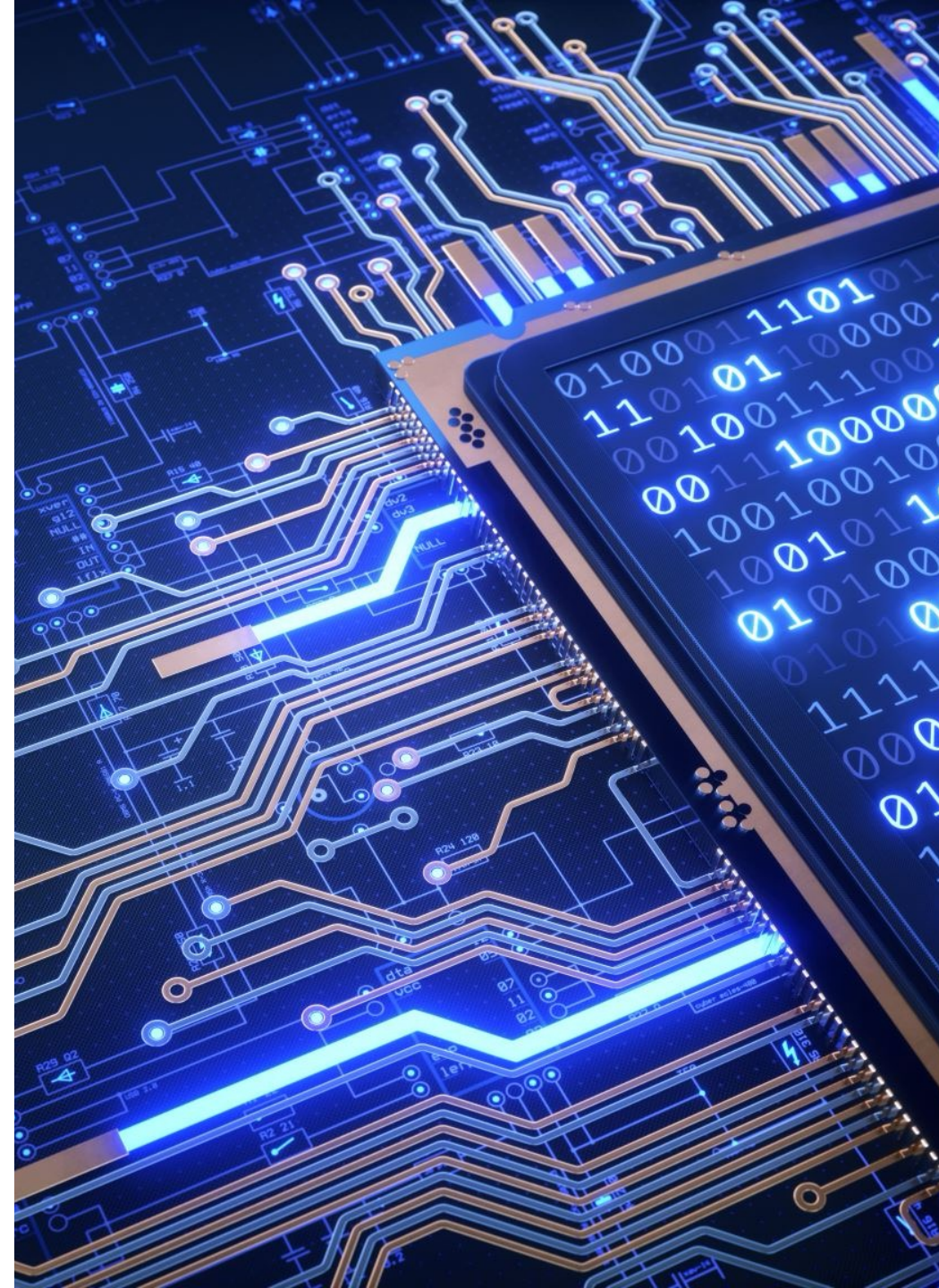
q = Kwame can write well

$p \vee q$ = Kwame can read well or Kwame can write well

OR

Kwame can read or write well

Truth table for Disjunction		
P	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F



Exclusive OR

- The connective "or" in a disjunction corresponds to inclusive OR
- The **exclusive OR** of two propositions p, q , denoted by $p \oplus q$ is the proposition " **p or q (but not both)**".
- The truth value of $p \oplus q$ is TRUE when exactly one of p, q is TRUE, and is FALSE otherwise

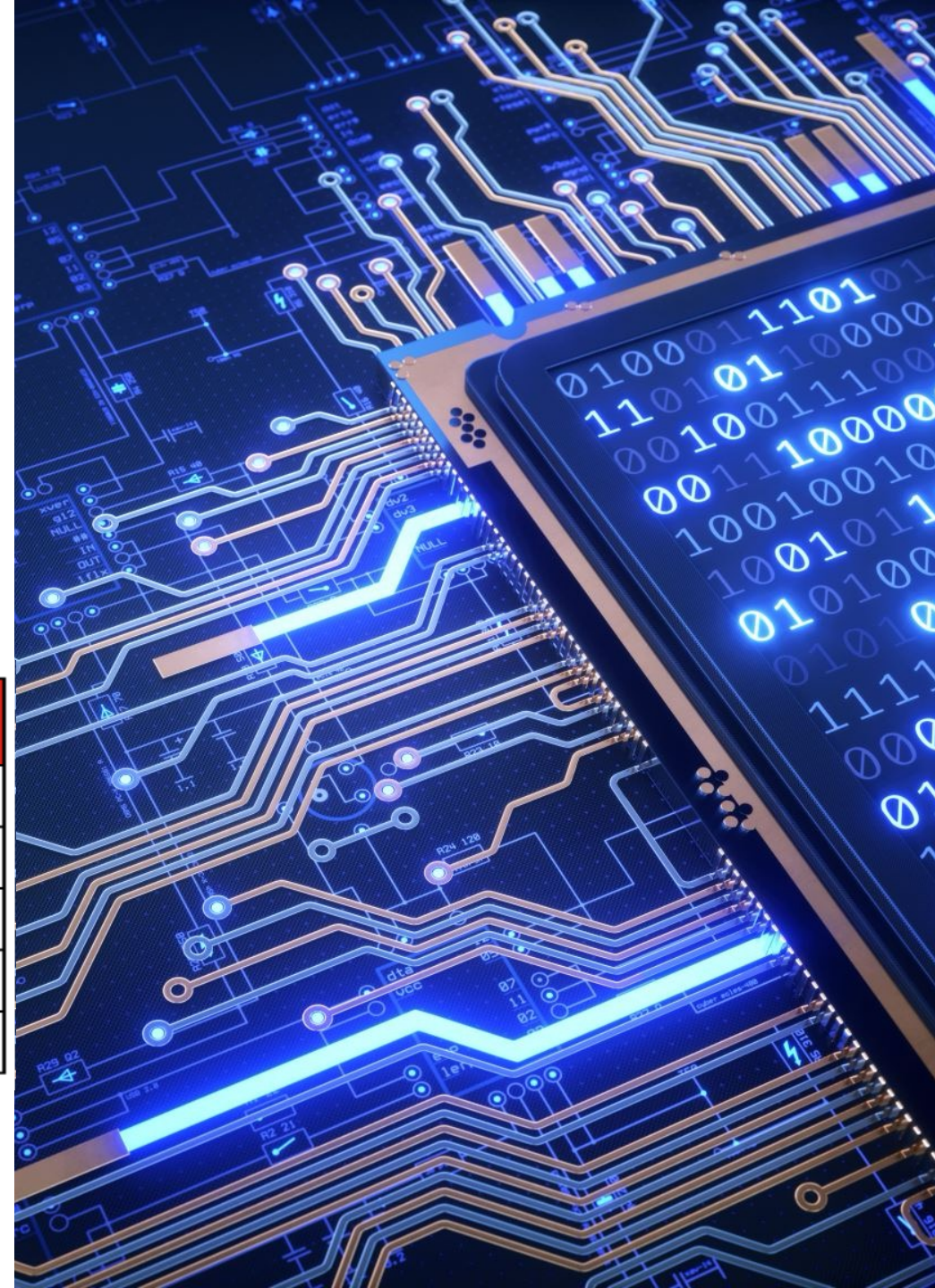
- 'Exclusive or' is applicable to two or more propositions

- Example:

p = Kwame can read well q = Kwame can write well

$p \oplus q$ = Kwame can read well or Kwame can write well or
Kwame can either read or write well but not both

Exclusive OR		
P	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

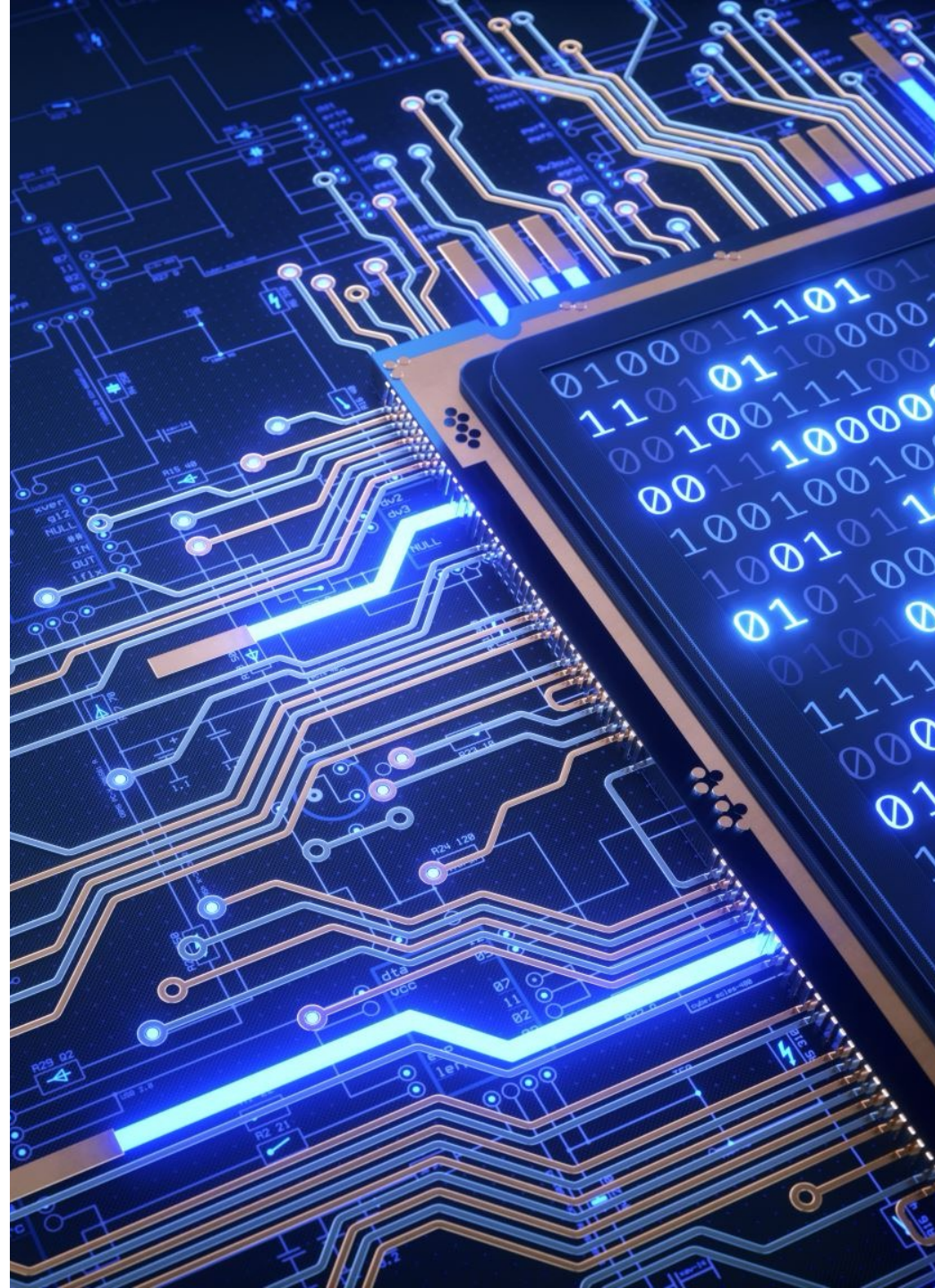


Implication

Conditional Statement (or Implication):

- The *conditional statement* $p \rightarrow q$ is the proposition “**if p , then q** ”, read as “ p implies q ”.
- For $p \rightarrow q$, the truth value is FALSE when p is TRUE and q is FALSE, otherwise it is TRUE
- In $p \rightarrow q$,
 p is called *hypothesis* or *antecedent* or *premise*
and
 q is called the *conclusion* or *consequence*

Truth table for Implication		
P	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T



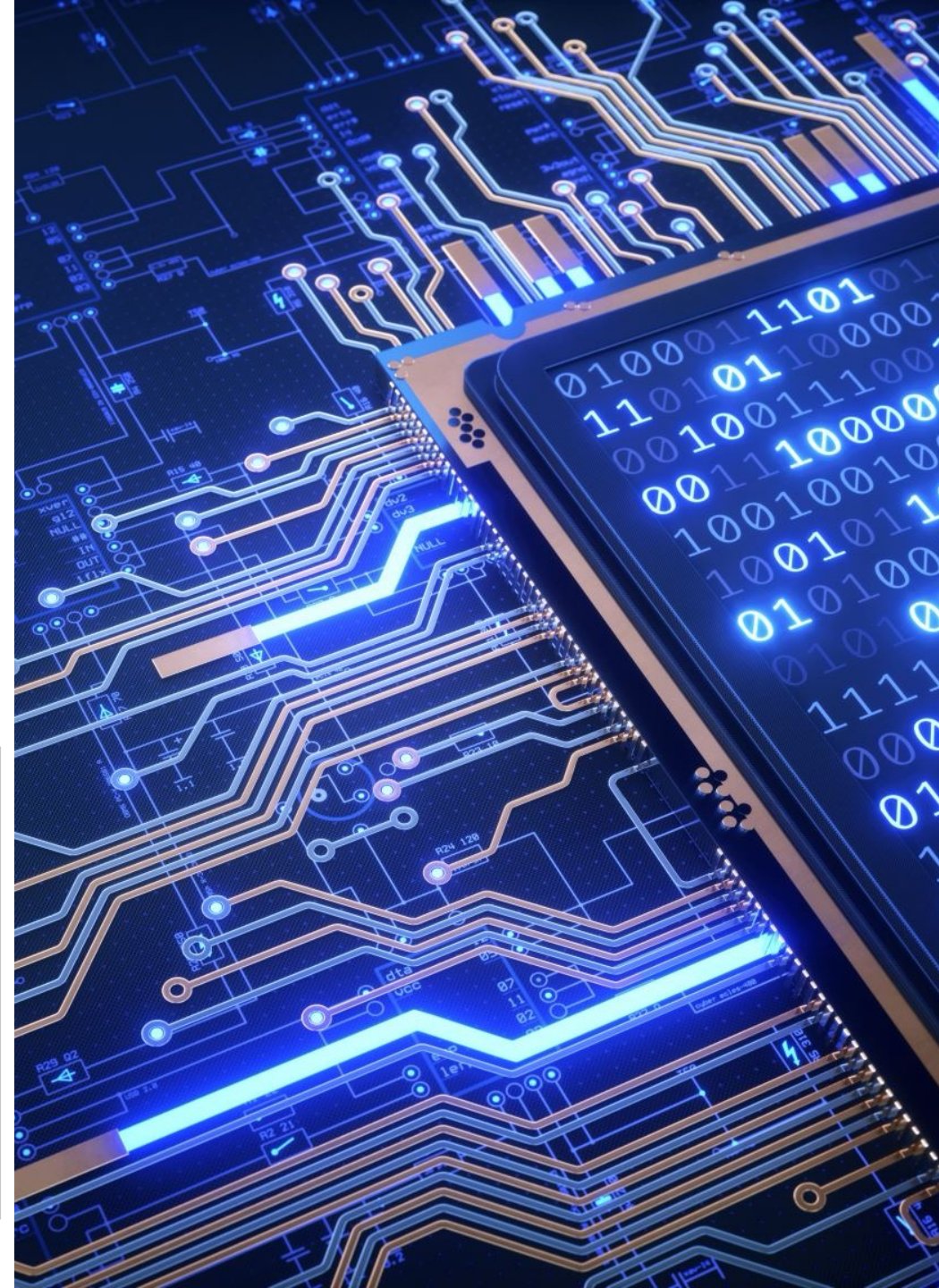
Implication

- Examples:

1) “If I am elected, then I will lower taxes” It is only when the politician is elected but does not lower taxes that voters can say that the politician has broken the campaign pledge.

2) “If you get 100%, then you will get an A” It is only when you get 100%, but the professor does not give you an A, you feel the statement is FALSE

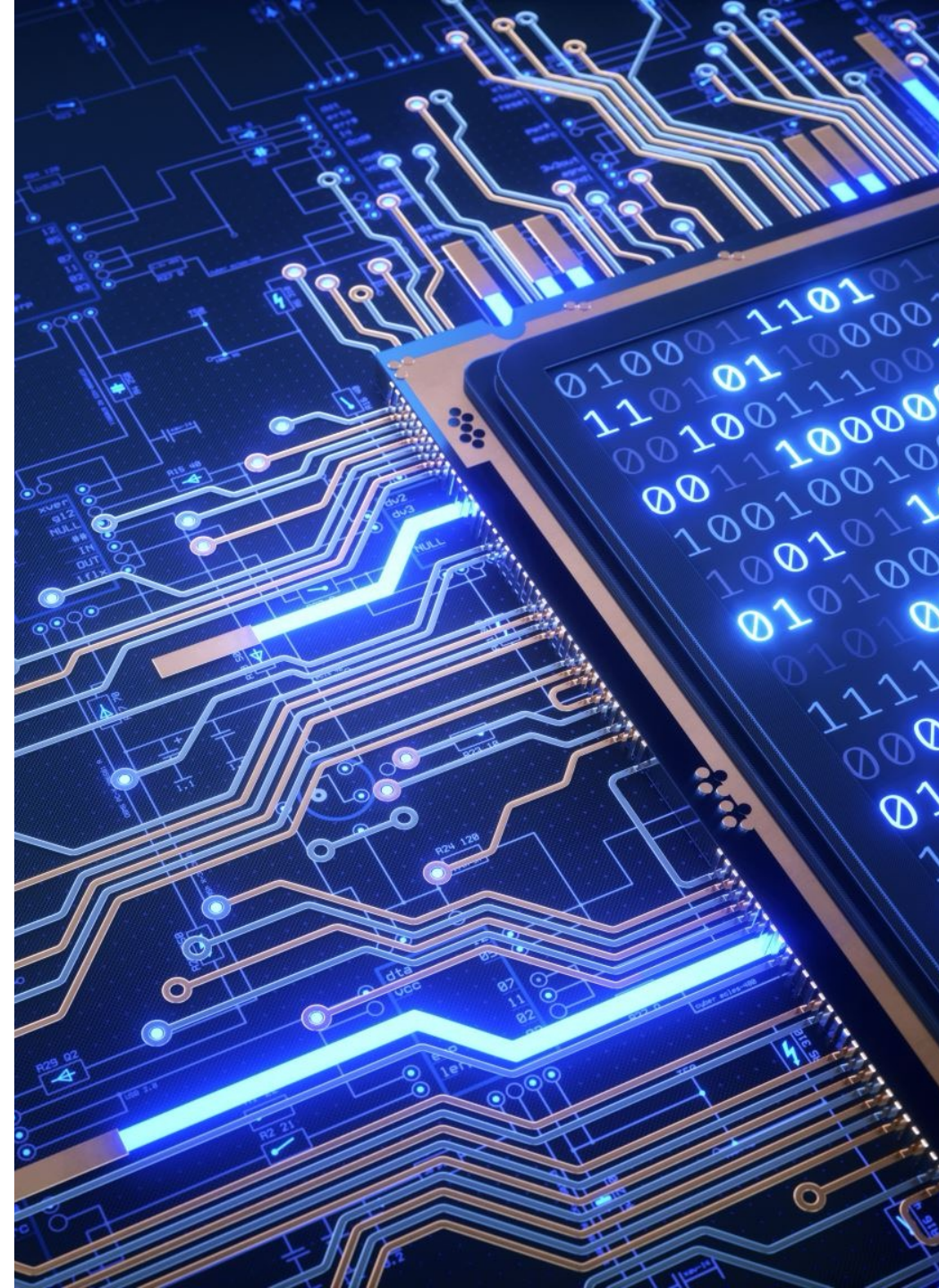
Truth table for Implication		
P	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T



Implication

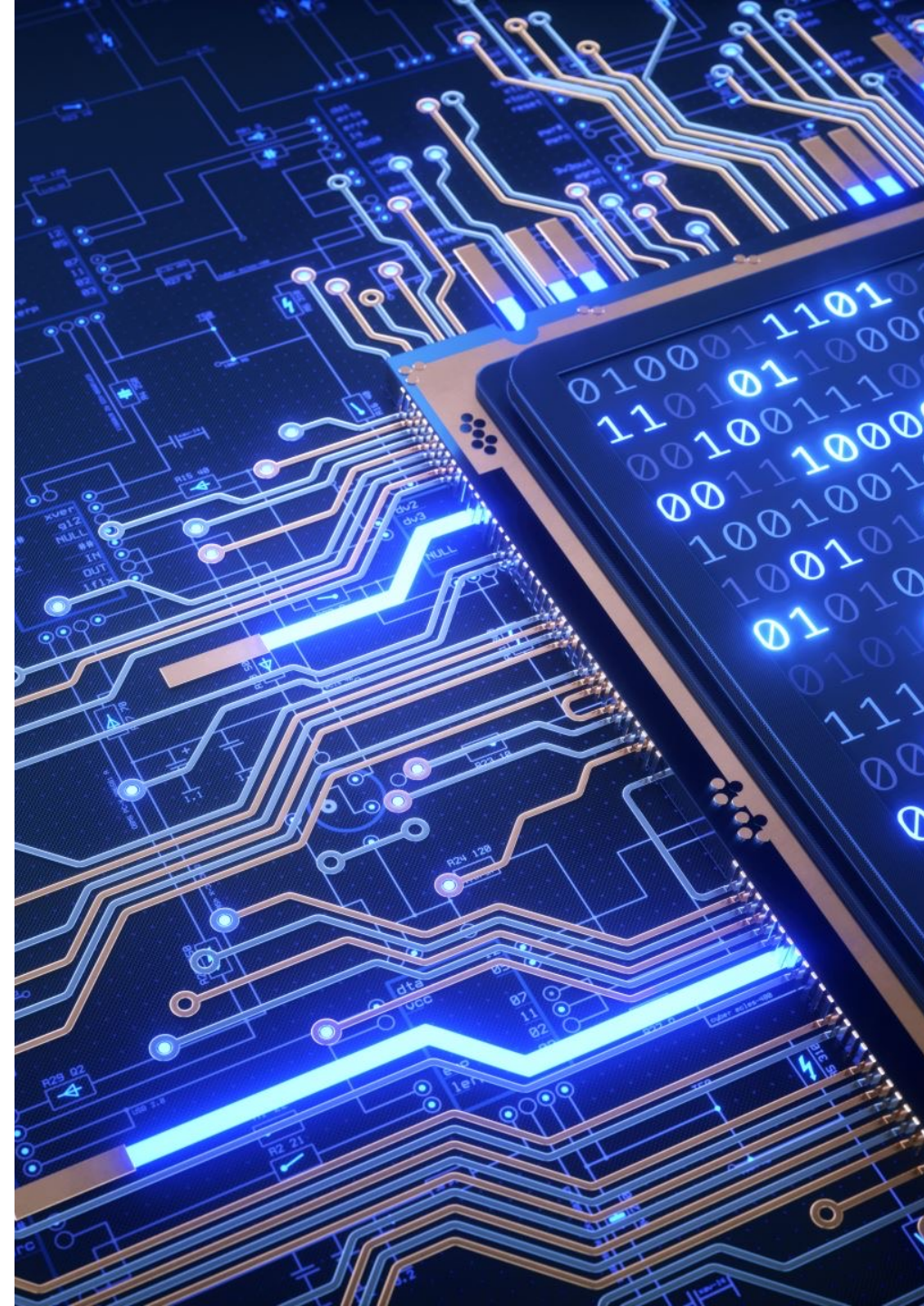
- Exercise (What did you observe?):
 - 1) “If Ama has a smartphone, then $2 + 3 = 5$ ”
 - 2) “If Ama has a smartphone, then $2 + 3 = 6$ ”
- The **if-then** construct used in programming is relatively different from the one used in propositional logic

Truth table for Implication		
P	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T



Implication Variants

- Given a conditional statement $p \rightarrow q$, we can form three new conditional statements, namely
 - The *converse* of $p \rightarrow q$, denoted by $q \rightarrow p$
 - The *contrapositive* of $p \rightarrow q$, denoted by $\neg q \rightarrow \neg p$
 - The *inverse* of $p \rightarrow q$, denoted by $\neg p \rightarrow \neg q$
- The contrapositive will have the same truth value as $p \rightarrow q$
- The converse and inverse will have same truth values



Implication variants: Equivalence

“When two compound propositions always have the same truth value we call them **equivalent**”

So, $p \rightarrow q$ and $\neg q \rightarrow \neg p$ are equivalent and
 $q \rightarrow p$ and $\neg p \rightarrow \neg q$ are equivalent

- Example: “The home team wins whenever it is raining”
(hint: Identify the Hypothesis and conclusion)
Contrapositive : “If the home team does not win, then it is not raining”
Converse : “If the home team wins, then it is raining”
Inverse : “If it is not raining, then the home team does not win.”



Bi-implication

Biconditional Statement (Bi-implication):

- The *biconditional statement*, denoted by $p \leftrightarrow q$ is the proposition “ **p if and only if q** ” (p iff q).
- $p \leftrightarrow q$ is TRUE when p and q have the same truth values, and is FALSE otherwise.
- $p \leftrightarrow q$ has exactly the same truth value as $(p \rightarrow q) \wedge (q \rightarrow p)$

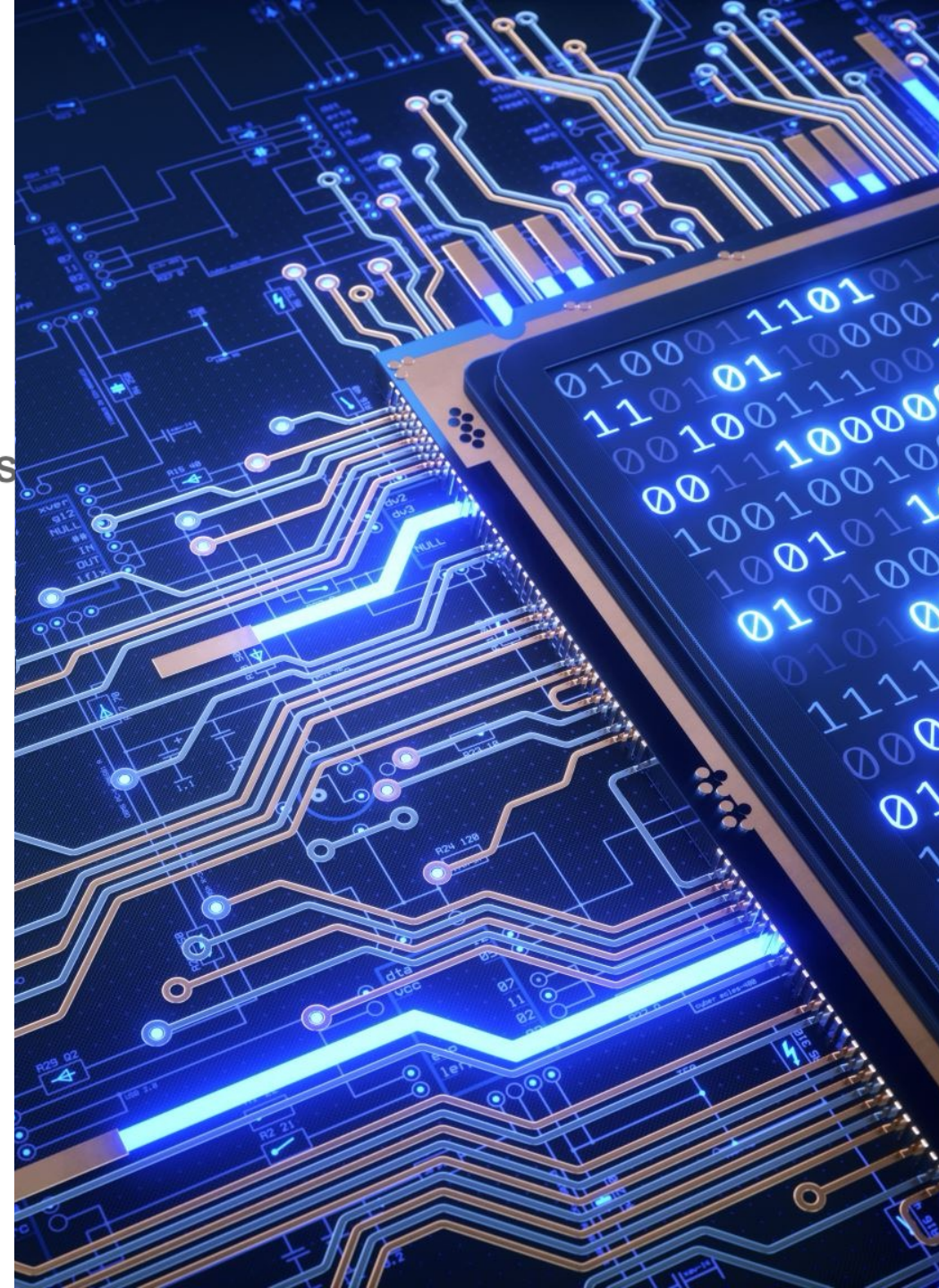
- Example:

p = “ You can take the flight”

q = “ You buy a ticket”

Bi-implication		
P	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

$p \leftrightarrow q$ = “You can take the flight if and only if you buy a ticket”



Exercise

Construct a truth table for the compound proposition

$$(p \vee \neg q) \rightarrow (p \wedge q)$$

Truth table for $(p \vee \neg q) \rightarrow (p \wedge q)$					
p	q	$\neg q$	$p \vee \neg q$	$p \wedge q$	$(p \vee \neg q) \rightarrow (p \wedge q)$
T	T	F	T	T	T
T	F	T	T	F	F
F	T	F	F	F	T
F	F	T	T	F	F

Precedence of Logical Operators

In the absence of proper parenthesization, compound propositions are evaluated according to the precedence of the operators

Examples:

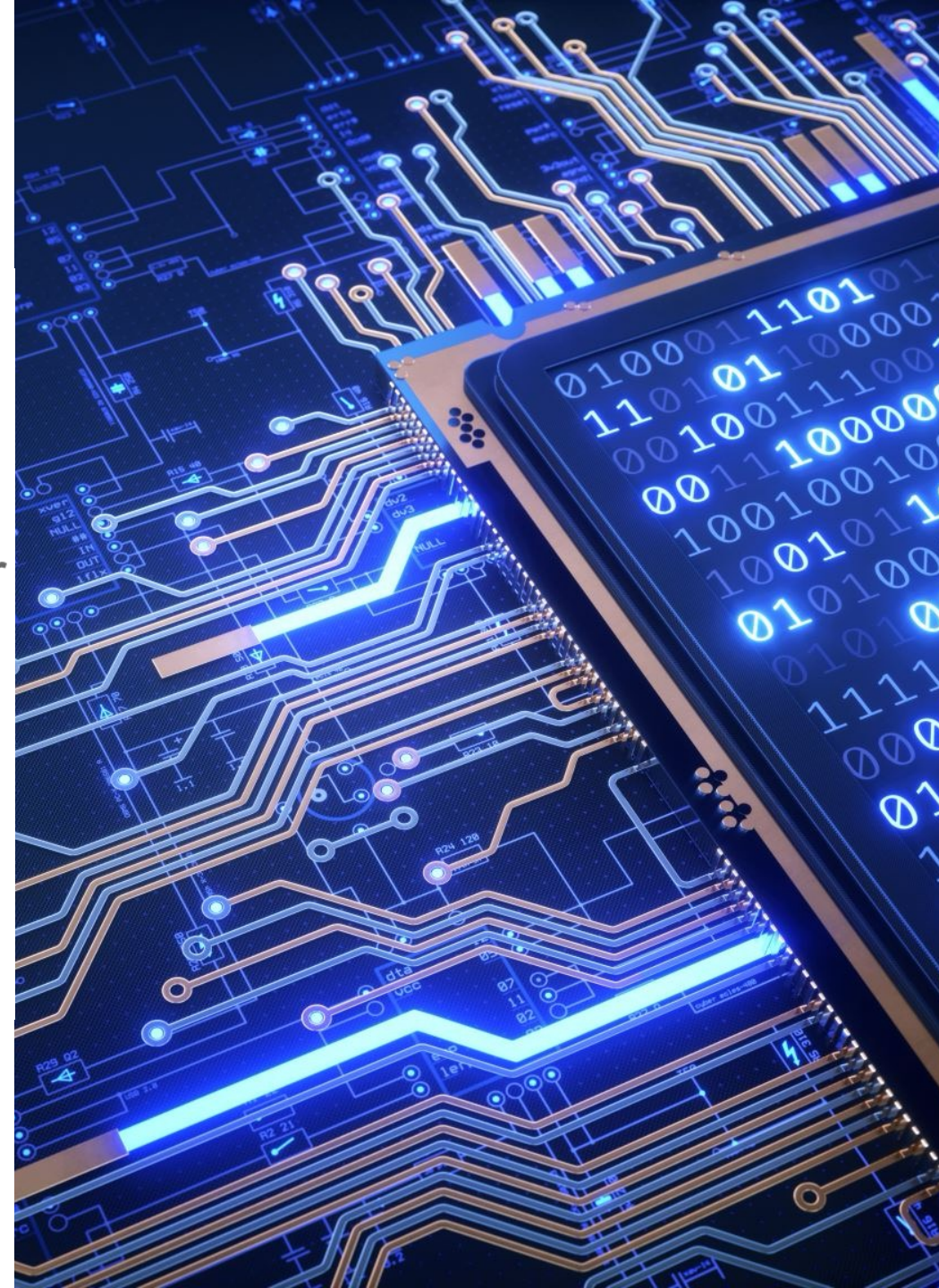
- $\neg p \wedge q$ is same as $(\neg p) \wedge q$
- $p \wedge q \vee r$ is same as $(p \wedge q) \vee r$
- $p \vee q \rightarrow r$ is same as $(p \vee q) \rightarrow r$

Precedence of logical operators	
Operator	Precedence
\neg	1
\wedge	2
\vee	3
\rightarrow	4
\leftrightarrow	5

Bit operations

- Computers represent information using **Bits**
- Bit means **B**inary **D**igit, which is either 0 or 1.
- Bits can be used to represent truth values with 1 for TRUE and 0 for FALSE
- **Boolean variable** is a variable having a value of either TRUE or FALSE, and hence *can be represented using a bit*

Truth value	Bit
T	1
F	0



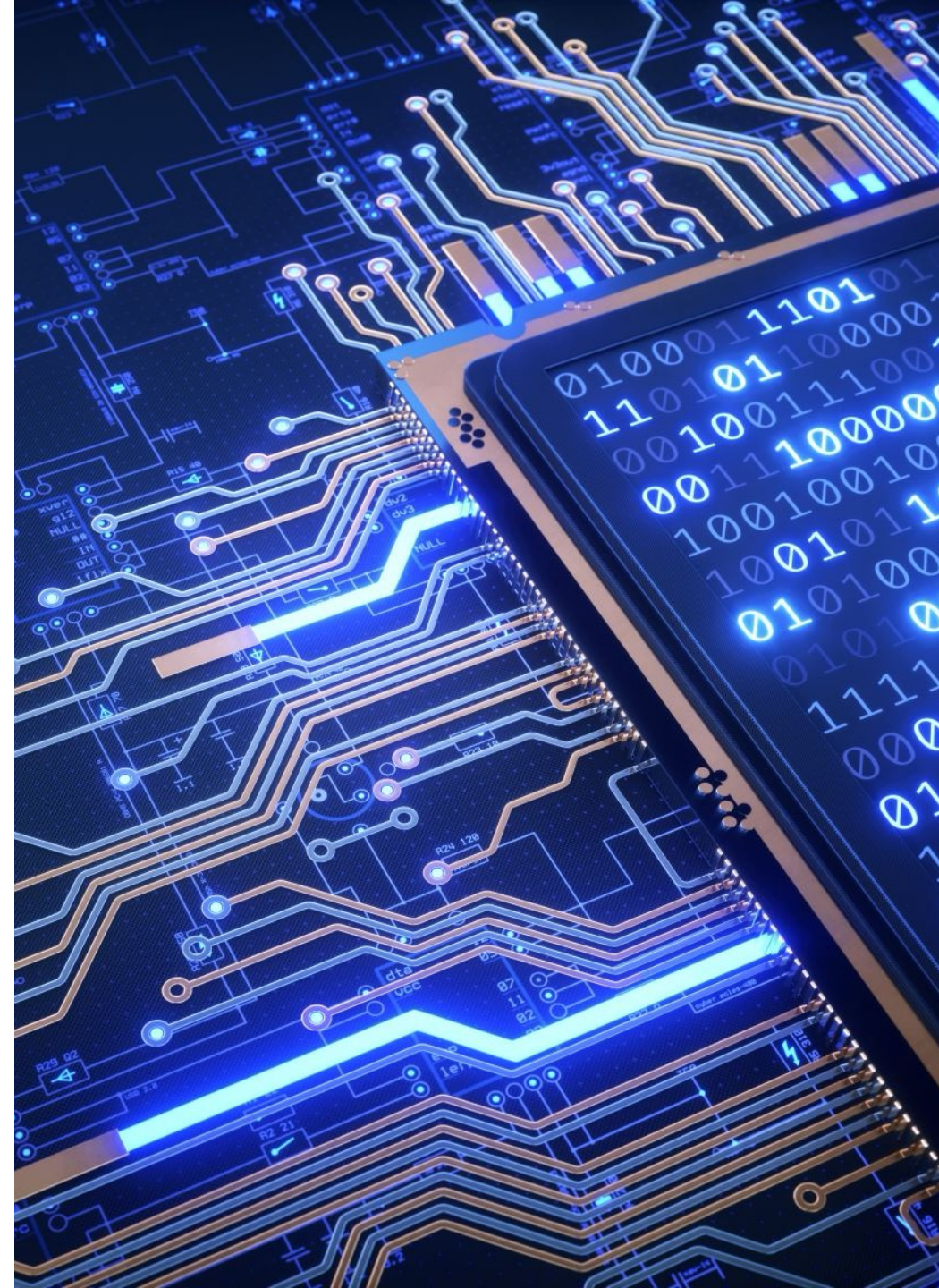
Exercise on Bit Operations

- Computer bit operations corresponding to the following logical operators:
 - \neg (NOT)
 - \wedge (AND)
 - \vee (OR)
 - \oplus (XOR)

Bit Operations						
x	y	$\neg x$	$\neg y$	$x \vee y$	$x \wedge y$	$x \oplus y$
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	1	0	1
1	1	0	0	1	1	0

Bit strings and Operations

- A bit string is a sequence of zero and/or one bits
- The length of a string is the number of bits in it
 - e.g.: 101010011 is a bit string of length nine
- Computers manipulate the information by performing operations on the bit strings
- The symbols \vee , \wedge and \oplus represent the **bitwise OR**, **bitwise AND**, and **bitwise XOR** operations, respectively
- Example: For the strings 01 1011 0110 and 11 0001 1101
 - bitwise OR gives 11 1011 1111
 - bitwise AND gives 01 0001 0100
 - bitwise XOR gives 10 1010 1011



Applications

Translating English sentences:

- Helps in removing ambiguity, analysing and determining the truth values and manipulating them using rules of inference

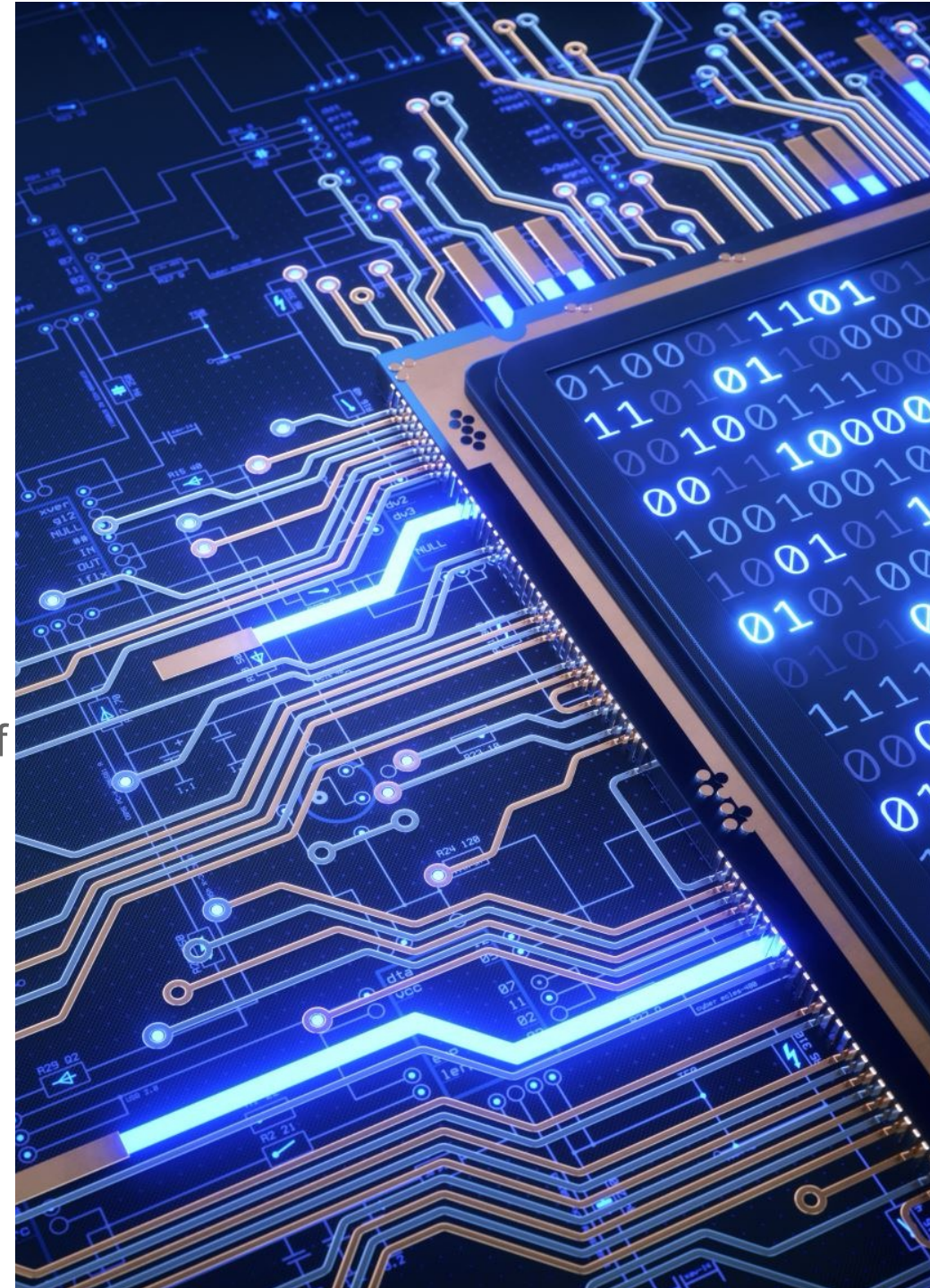
- Examples:

1) *English sentence*: “You can access the Internet from campus only if you are a computer science major or you are not a freshman”.

Logical expression: $(c \vee \neg f) \rightarrow a$

2) *English sentence*: “You cannot ride the roller coaster if you are under 4 feet tall and not older than 16 years old.”

Logical expression: $(r \wedge \neg s) \rightarrow \neg q$



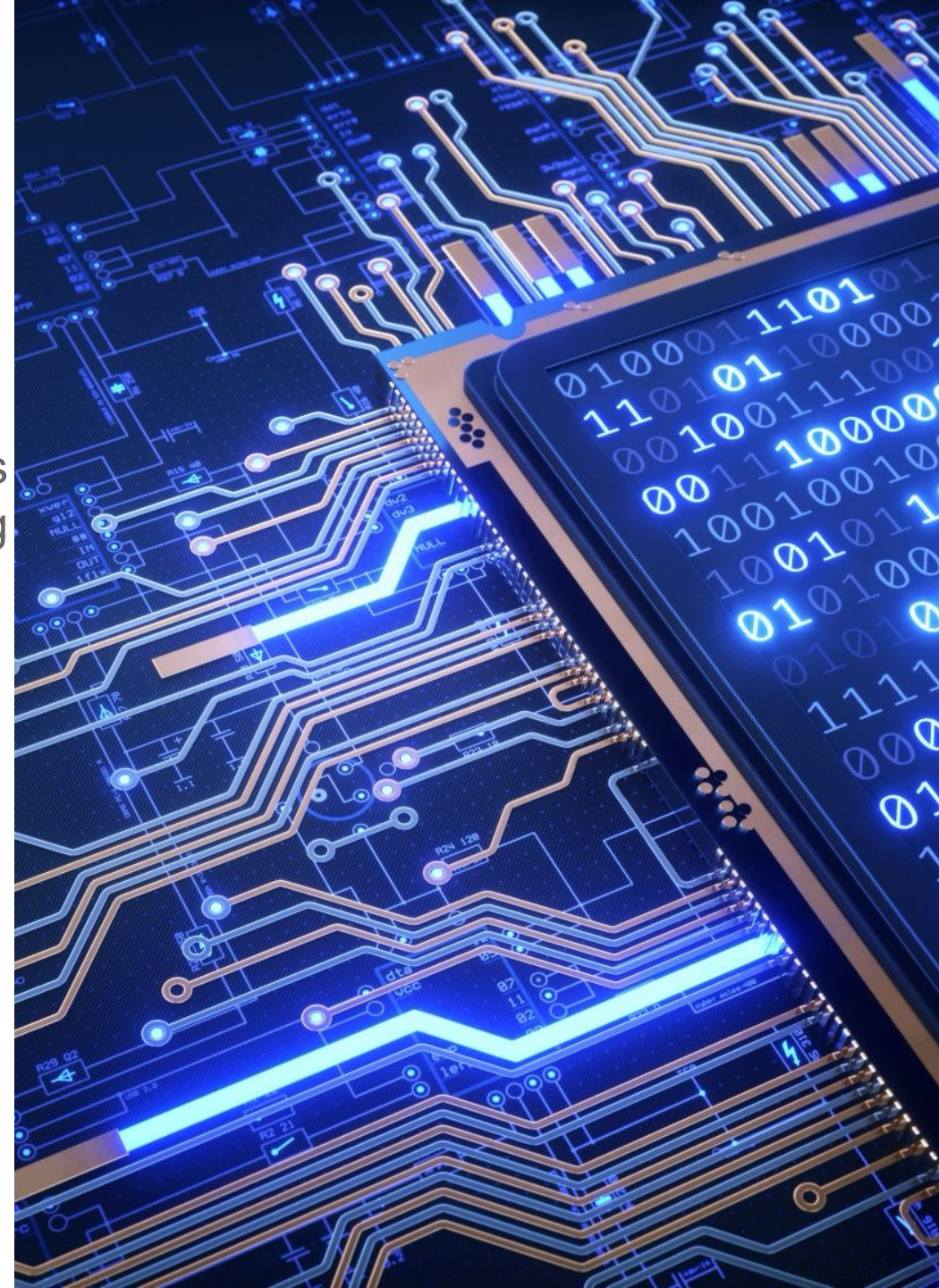
Applications

System Specification:

- System and software engineers create clear, concise specifications that can serve as the foundation for system development by taking requirements expressed in common language.
- Example:

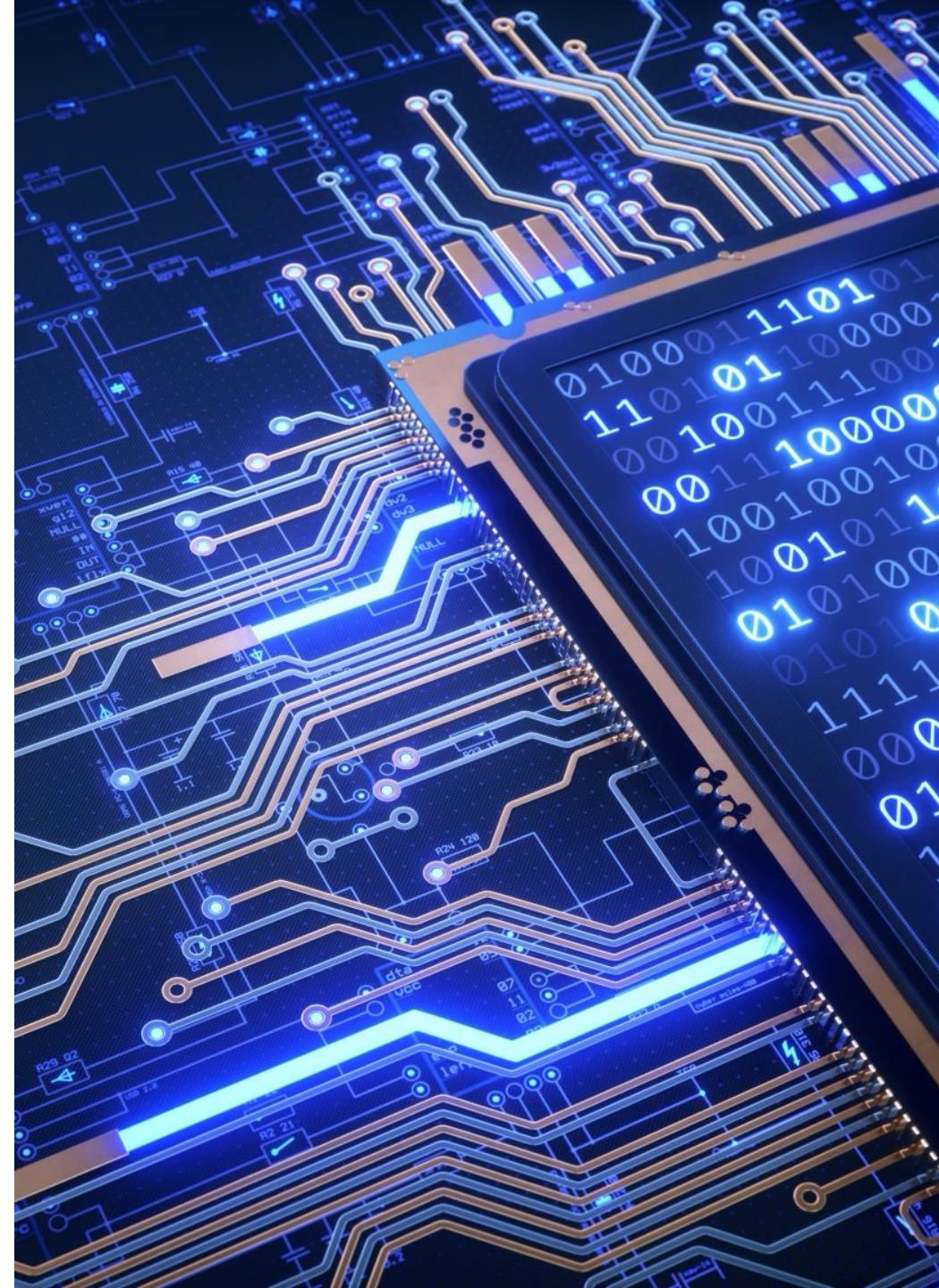
1) *Specification* : “The automated reply cannot be sent when the file system is full”

Logical expression: $q \rightarrow \neg p$



Applications

- **Boolean Searches:**
 - Search over large collections of information, for example Web search, employ techniques from propositional logic and hence called Boolean searches
 - The connective AND is used to match records that contain both search terms, the connective OR is used to match one or both search terms, and the connective NOT is used to exclude a particular search term
- Examples:
 - Search for web pages matching NEW AND MEXICO AND UNIVERSITIES Search for web pages matching (NEW AND MEXICO OR ARIZONA) AND UNIVERSITIES.



Applications

Logic Circuits:

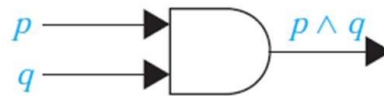
- Propositional logic can be applied to the design of computer hardware (*logic circuits* or *digital circuits*)
- A **logic circuit** (or **digital circuit**) receives input signals p_1, p_2, \dots, p_n , each bit [either 0 (OFF) or 1 (ON)], and produces output signals s_1, s_2, \dots, s_n , each bit respectively.
- Complicated digital circuits are constructed from the three basic circuits, called **gates**



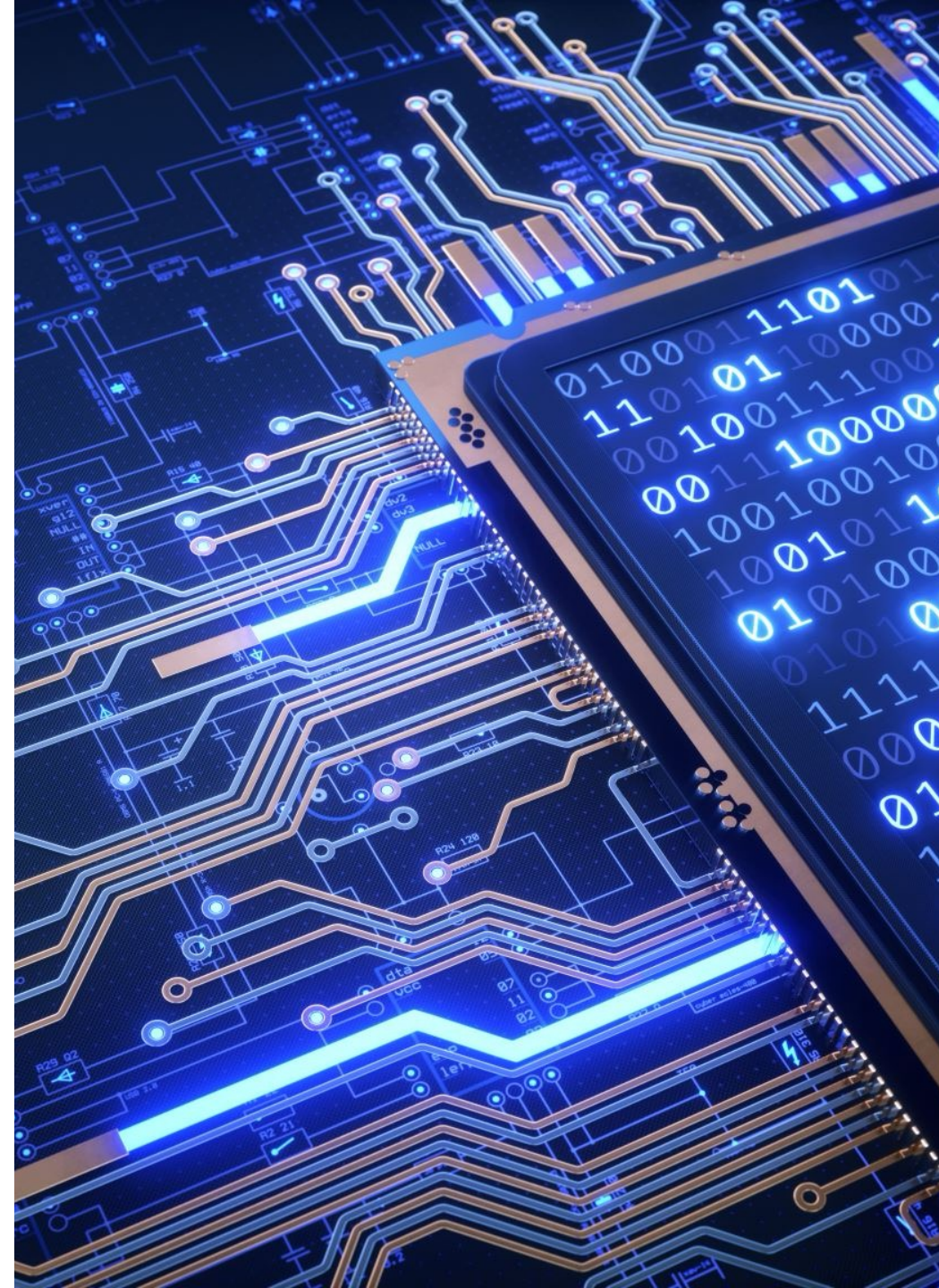
Inverter



OR gate

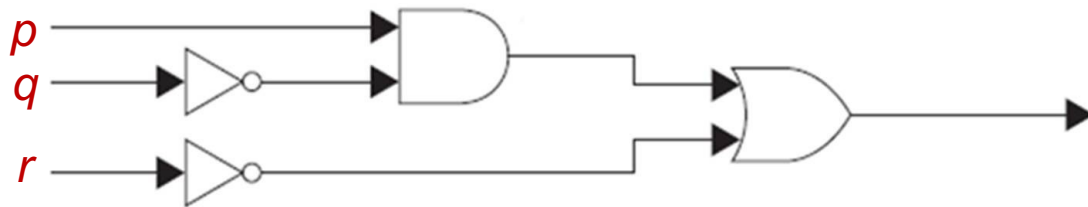


AND gate

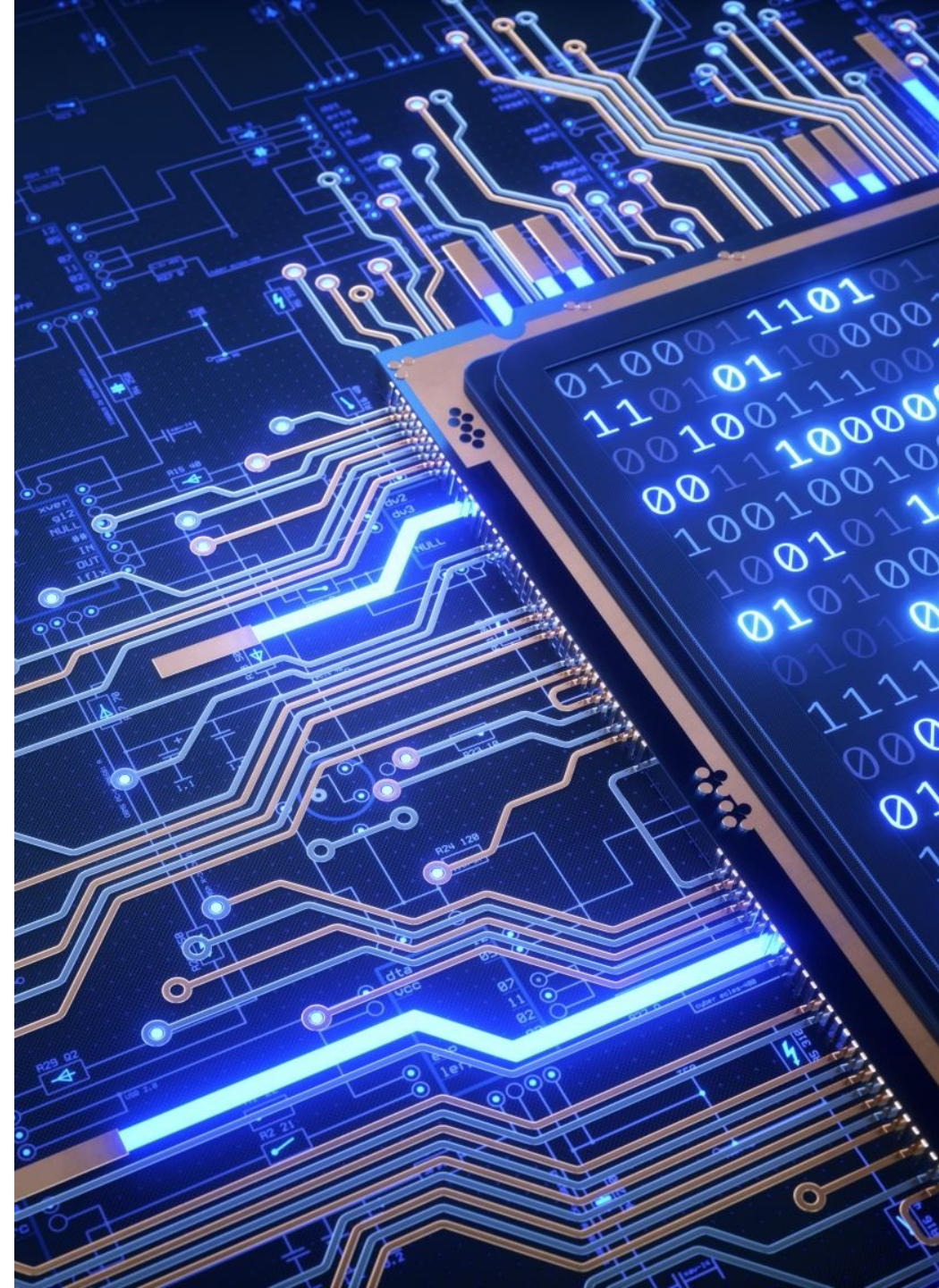


Exercise

Determine the output of the following digital circuit when $p = 1$, $q = 0$ and $r = 1$



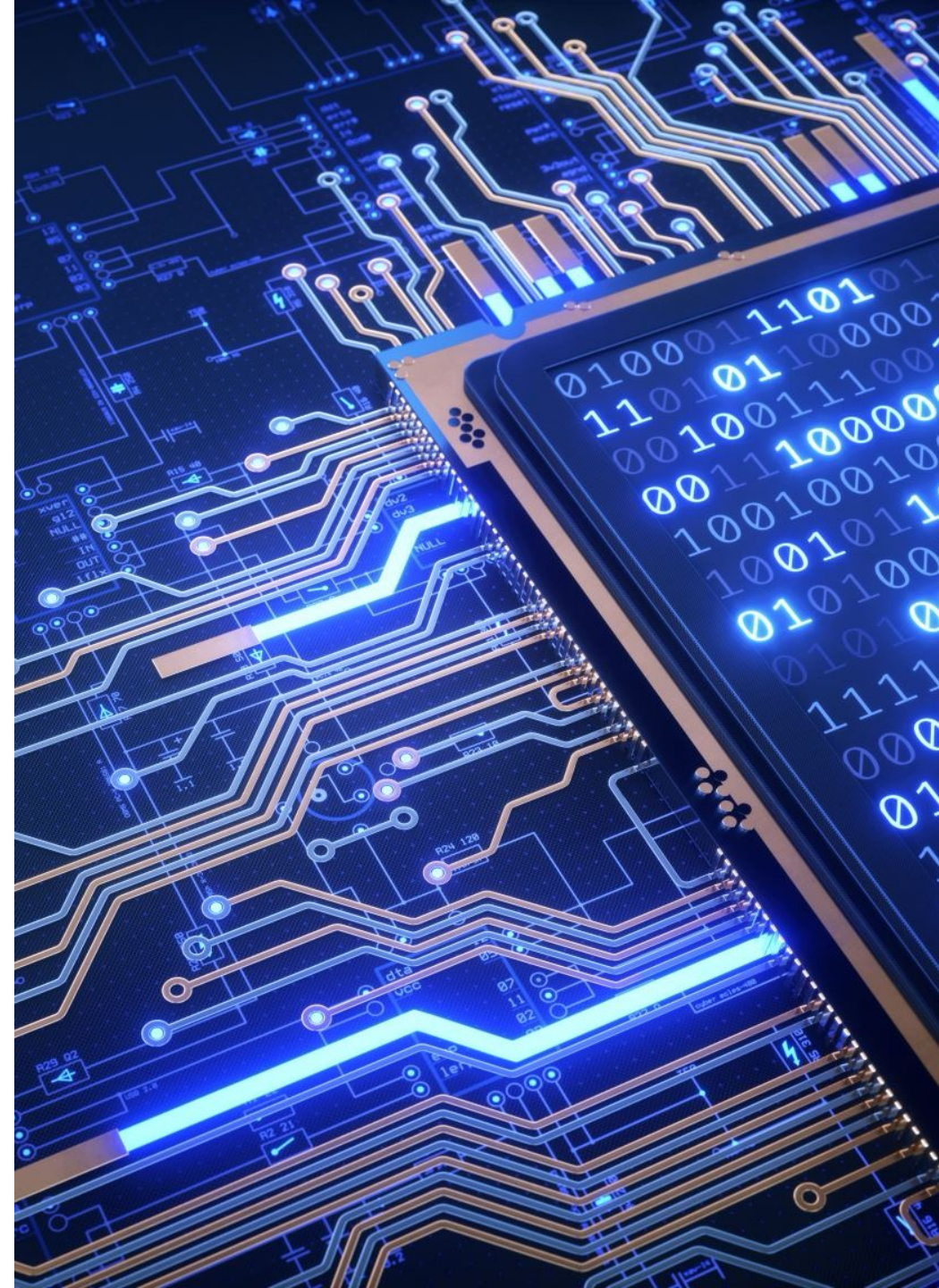
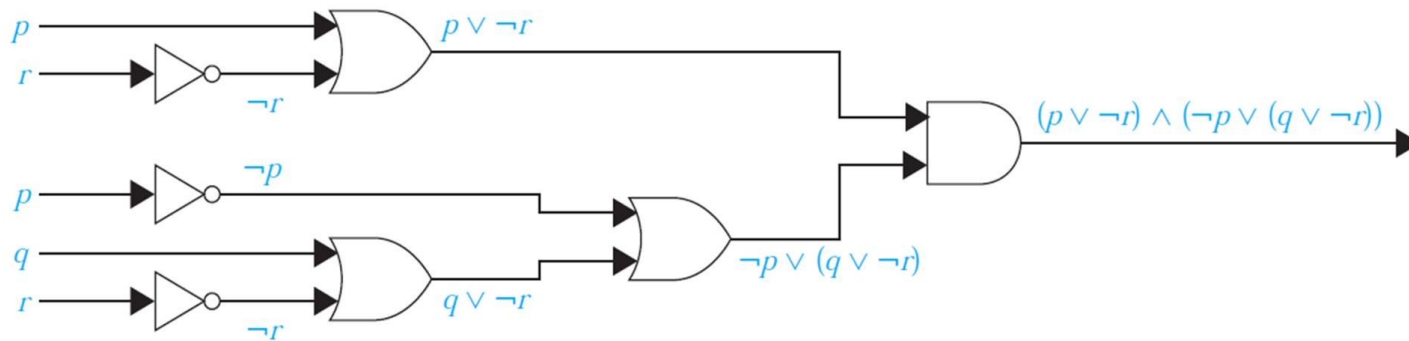
Answer: $(p \wedge \neg q) \vee \neg r = 1$



Exercise

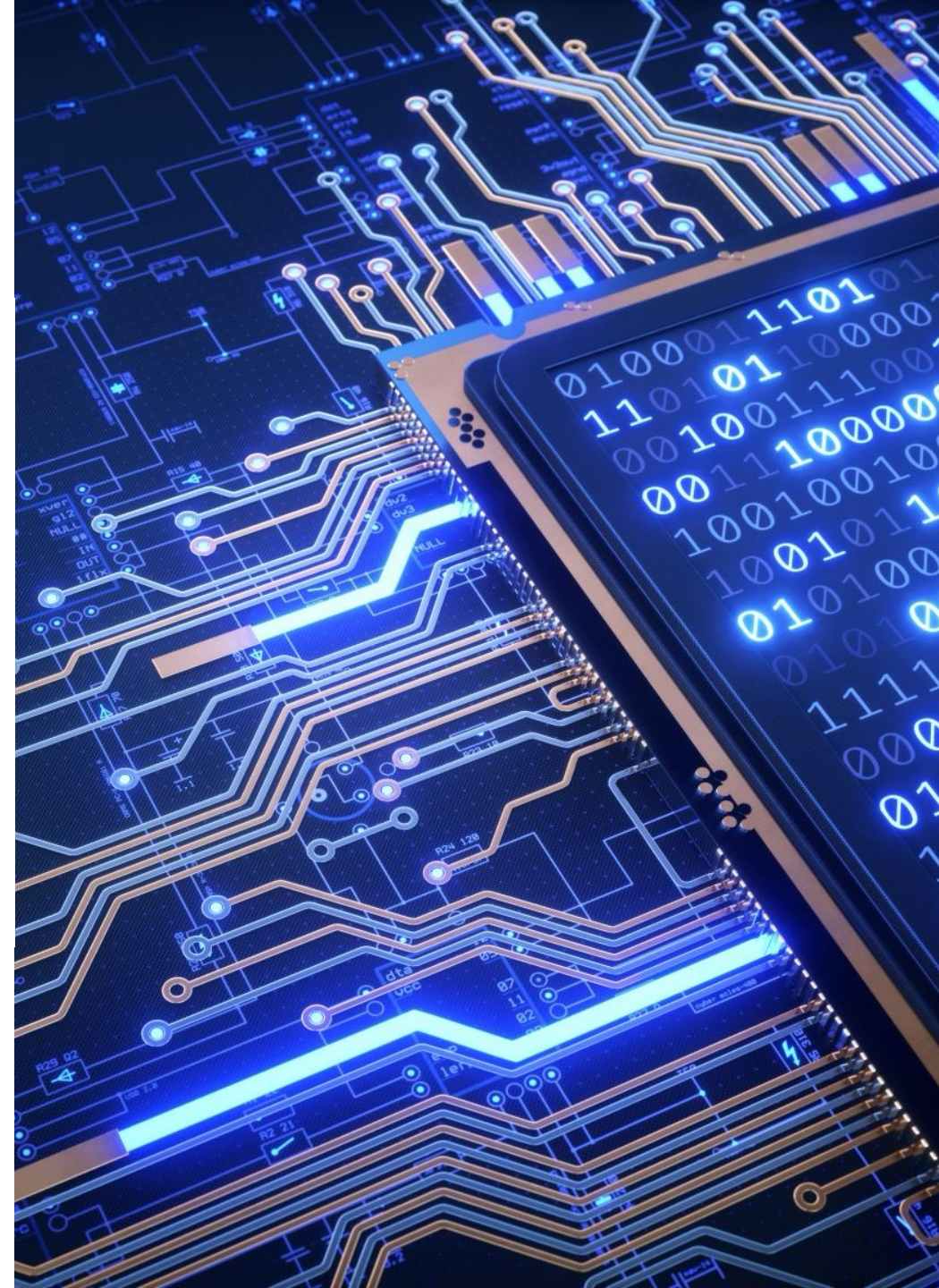
- Construct a digital circuit that results in the output $(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$ when given input bits p , q , and r

Answer:



Propositional Equivalence

- **Tautology:**
 - A *compound proposition that is always TRUE*, no matter what the truth values of the propositional variables that occur in it, is called a *tautology*.
- **Contradiction:**
 - A *compound proposition that is always FALSE*, no matter what the truth values of the propositional variables that occur in it, is called a *contradiction*.
- **Contingency:**
 - A compound proposition that is neither a tautology nor a contradiction is called a *contingency*.



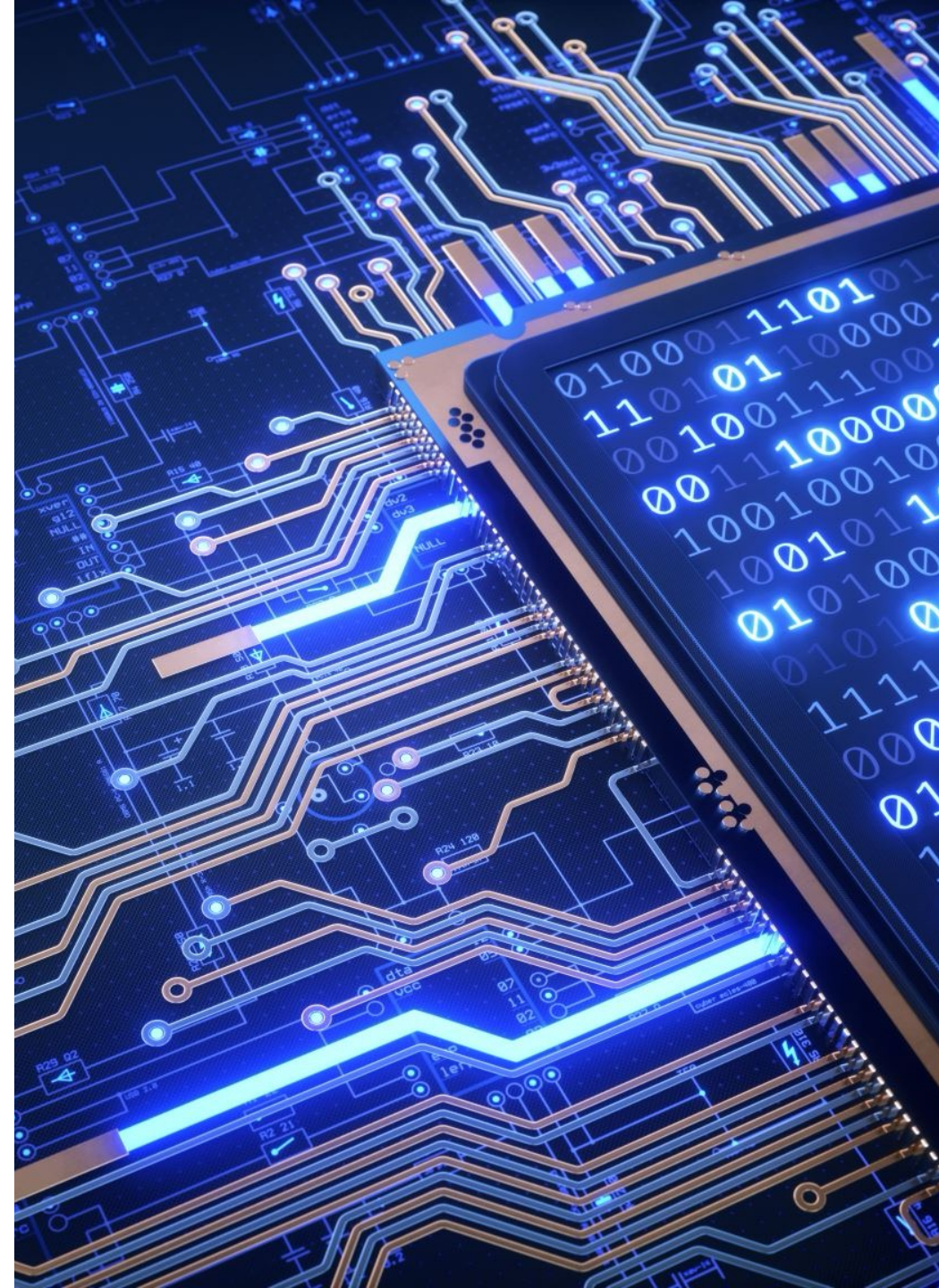
Propositional Equivalence

Logical equivalence:

- Compound propositions that have the same truth values in all possible cases are called **logically equivalent**.

or in other words,

- The compound propositions p and q are called *logically equivalent* if $p \leftrightarrow q$ is a tautology.
- The notation $p \equiv q$ (or sometimes $p \Leftrightarrow q$) is used to denote that
 p and q are logically equivalent.



Propositional Equivalence

Logical equivalence:

- The equivalence of two compound statements can be determined by constructing a truth table and checking whether the columns giving their truth values are same
- Example: Prove that the output from $\neg(p \vee q)$ is logically equivalent to $\neg p \wedge \neg q$

Truth Tables for $\neg(p \vee q)$ and $\neg p \wedge \neg q$.						
p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
T	T	T	F	F	F	F
T	F	T	F	F	T	F
F	T	T	F	T	F	F
F	F	F	T	T	T	T

- **Note:** $\neg(p \vee q) \equiv \neg p \wedge \neg q$ is one of the two **De Morgan's Laws**

Propositional Equivalence

- Exercise: Proof that $p \rightarrow q$ is logically equivalent to $\neg p \vee q$

Truth Tables for $\neg p \vee q$ and $p \rightarrow q$.				
p	q	$\neg p$	$\neg p \vee q$	$p \rightarrow q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Propositional Equivalence

- Exercise: Show that $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

A Demonstration That $p \vee (q \wedge r)$ and $(p \vee q) \wedge (p \vee r)$ Are Logically Equivalent.							
p	q	r	$q \wedge r$	$p \vee (q \wedge r)$	$p \vee q$	$p \vee r$	$(p \vee q) \wedge (p \vee r)$
T	T	T	T	T	T	T	T
T	T	F	F	T	T	T	T
T	F	T	F	T	T	T	T
T	F	F	F	T	T	T	T
F	T	T	T	T	T	T	T
F	T	F	F	F	T	F	F
F	F	T	F	F	F	T	F
F	F	F	F	F	F	F	F

Propositional Equivalence

De Morgan's laws can be extended to any number of propositions

For example,

$$\neg(p_1 \vee p_2 \vee \dots \vee p_n) \equiv (\neg p_1 \wedge \neg p_2 \wedge \dots \wedge \neg p_n)$$

<i>Equivalence</i>	<i>Name</i>
$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identity laws
$p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$	Domination laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent laws
$\neg(\neg p) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws
$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's laws
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv \mathbf{T}$ $p \wedge \neg p \equiv \mathbf{F}$	Negation laws

Propositional Equivalence

Logical Equivalences Involving Conditional Statements.

$$p \rightarrow q \equiv \neg p \vee q$$

$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$

$$p \vee q \equiv \neg p \rightarrow q$$

$$p \wedge q \equiv \neg(p \rightarrow \neg q)$$

$$\neg(p \rightarrow q) \equiv p \wedge \neg q$$

$$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$$

$$(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \vee q) \rightarrow r$$

$$(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$$

$$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$$

Logical Equivalences Involving Biconditional Statements.

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$

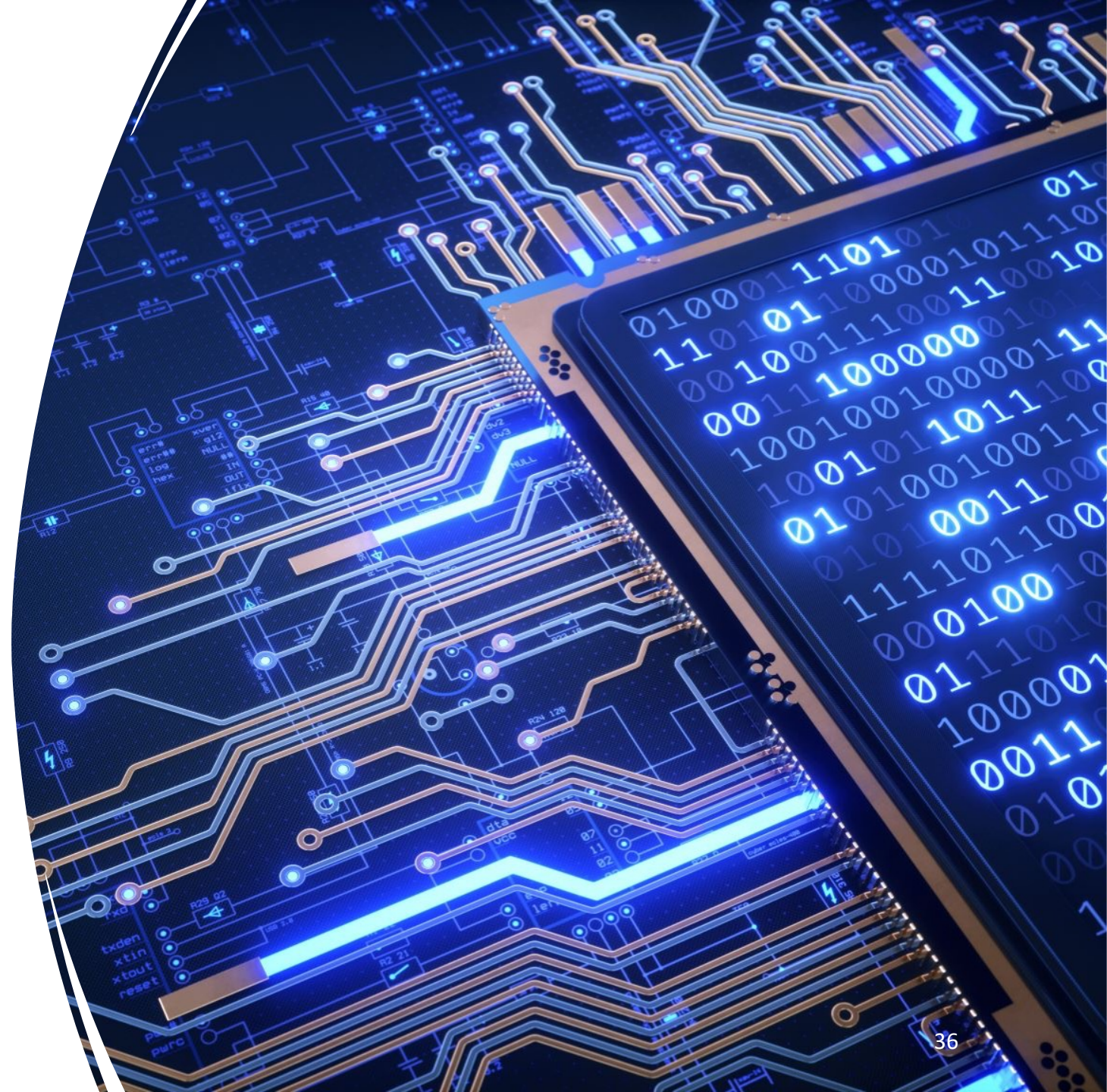
$$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$$

$$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$$

$$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$$

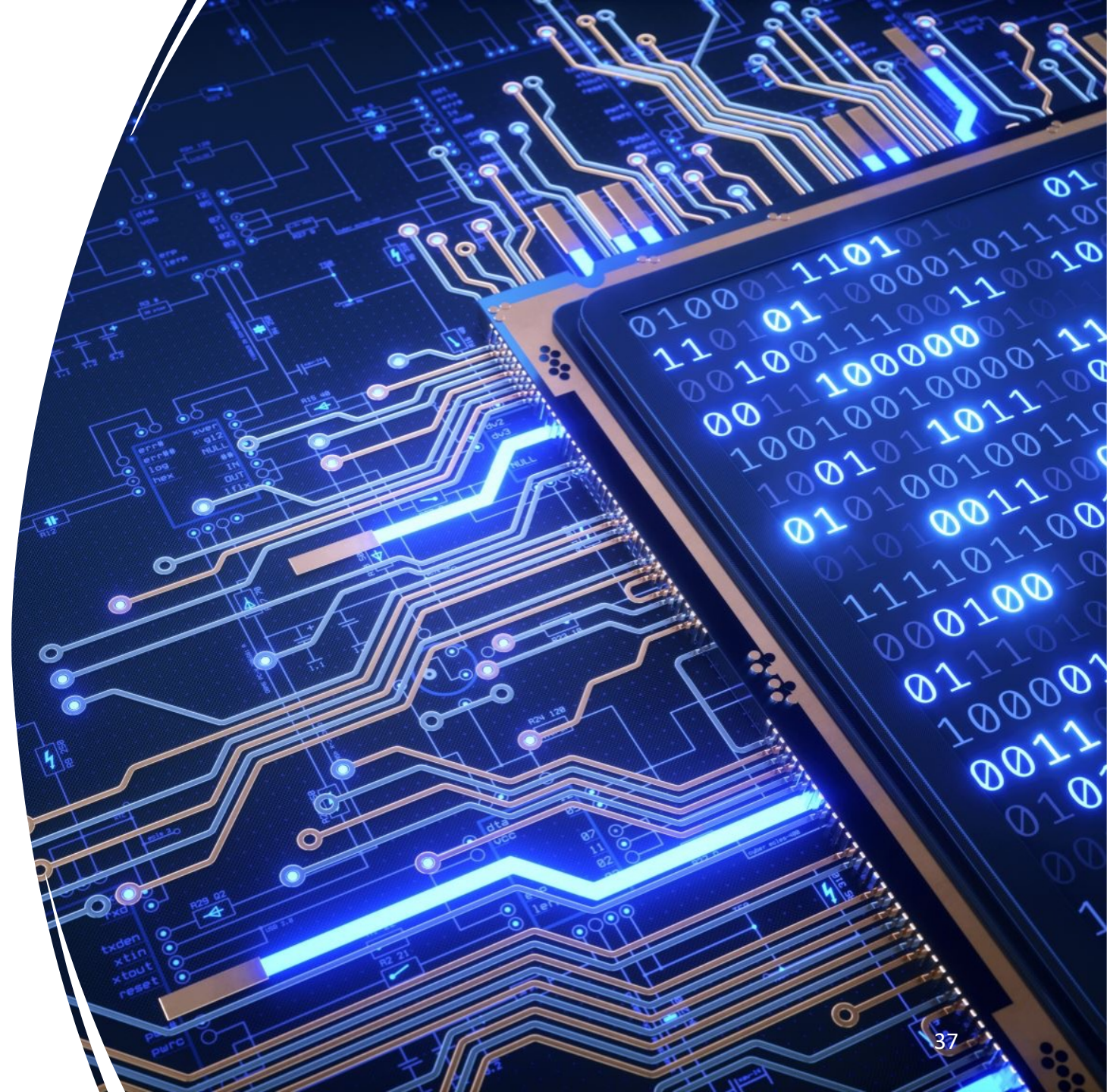
Summary

- Proposition is either true or false
- Logic operations
 - Negation
 - Conjunction
 - Disjunction
 - Implication
 - Bi-implication
- Construction of truth tables
- Bit string operations
- Propositional equivalence



Reference

Rosen, K. H. (2012). *Discrete mathematics and its applications (7th Edition)*. McGraw-Hill.
Chapter 1



See you next
time!

*Thank
you!*