Course: Mathematics for IT Professionals

Lecture 2

Number Systems II: Arithmetic Computation

By

Solomon Mensah

Outline

The topics to be treated in this lecture are:

- Complements in Number Systems
- Overflow
- Addition of Signed Numbers
- Subtraction of Signed Numbers
- Multiplication of Signed Numbers
- Division of Signed Numbers
- Application of arithmetic computation

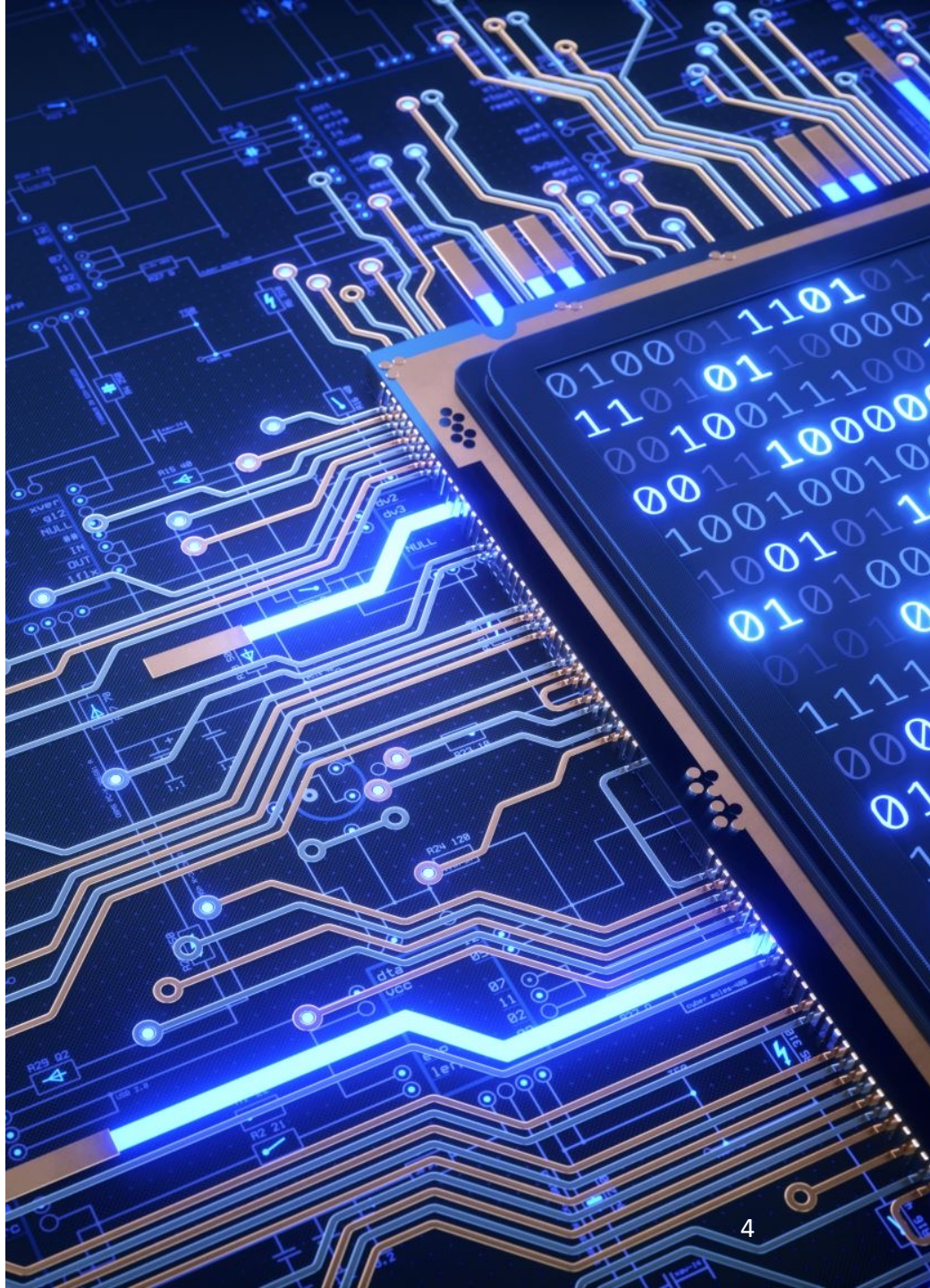
Lecture Learning Outcomes

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

- define the compliments in number systems
- understand overflow in number systems
- add two signed numbers
- subtract two signed numbers
- multiply two signed numbers
- divide two signed numbers
- have knowledge on applications of arithmetic computation

Introduction

- Complement is considered in digital computers for performing arithmetic computations.
  - It is used for representing the negative decimal number in binary form.
- Different formats of complement are possible of the binary number,
  - but 1's and 2's complements are mostly used for binary numbers.

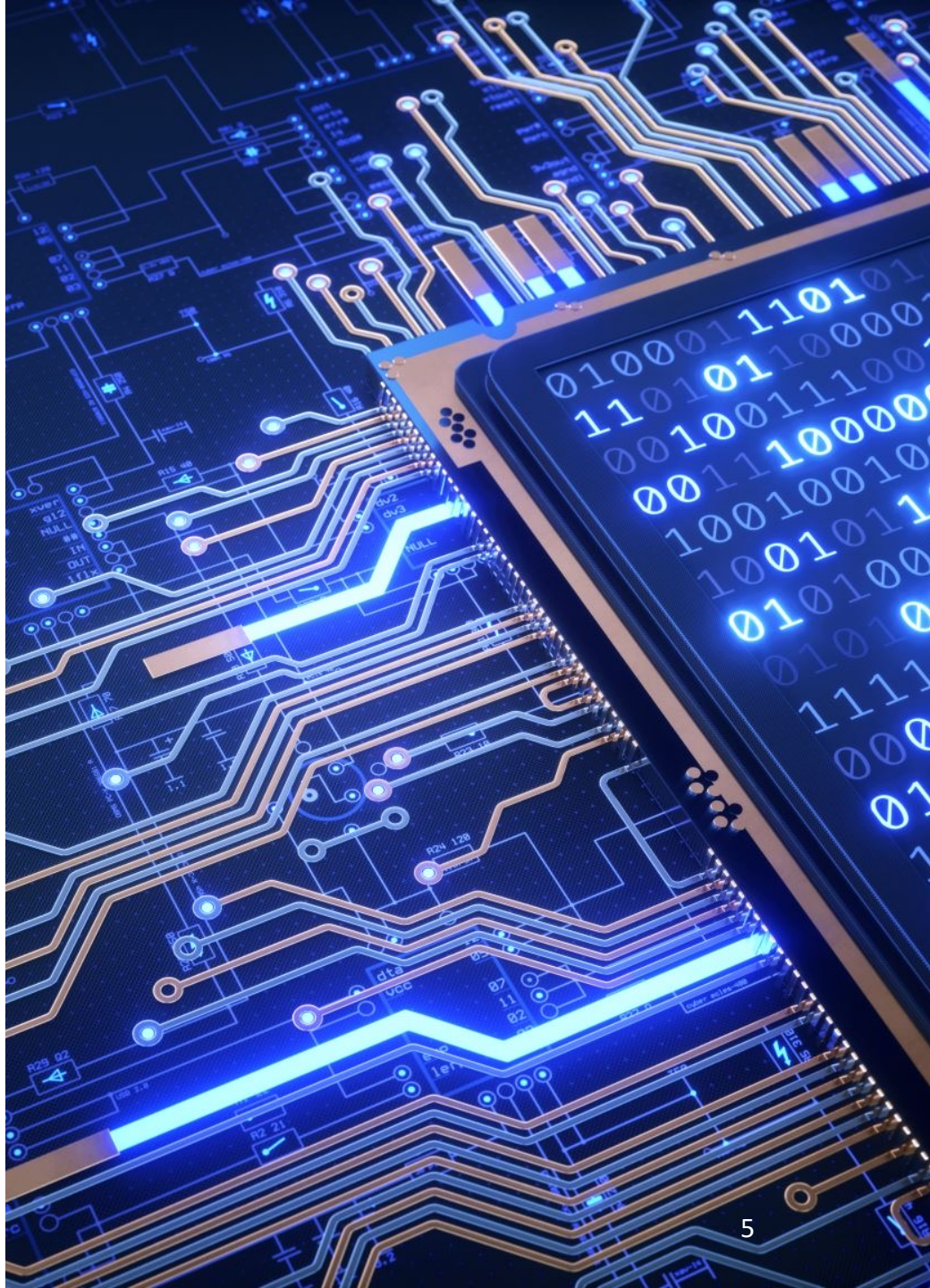


Maini, A. K. (2007). "Digital Electronics: Principles, Devices and Applications". *John Wiley & Sons, Ltd*. ISBN: 978-0-470-03214-5

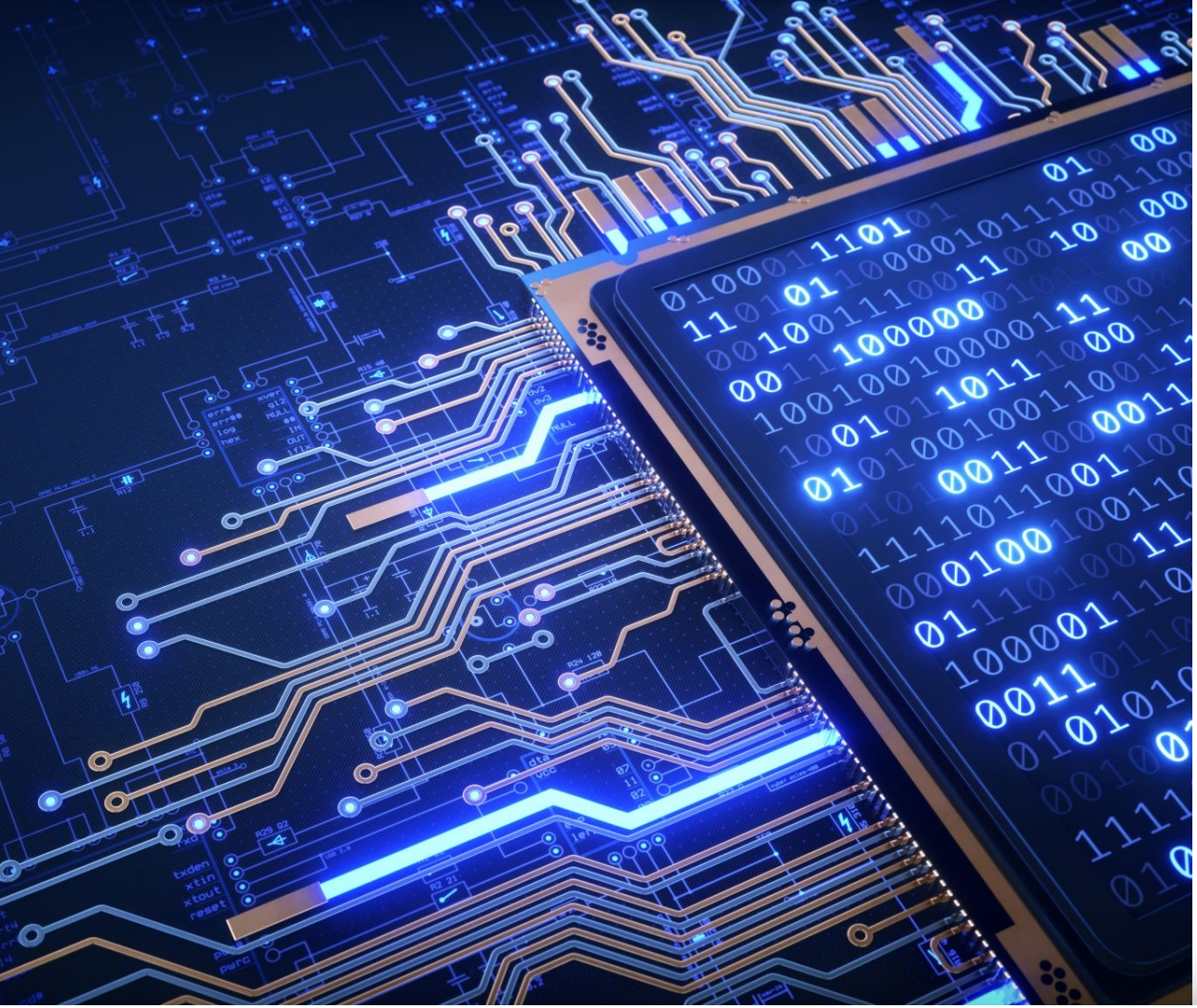
Number Systems: Complements

- Binary Number System
  - 1's complement achieved by inverting all its bits
  - 2's complement achieved by adding '1' to the LSB of 1's complement
  - E.g.: 0101 0110

Ans: 1010 1001 → 1's complement
1010 1010 → 2's complement



Maini, A. K. (2007). "Digital Electronics: Principles, Devices and Applications". *John Wiley & Sons, Ltd*. ISBN: 978-0-470-03214-5

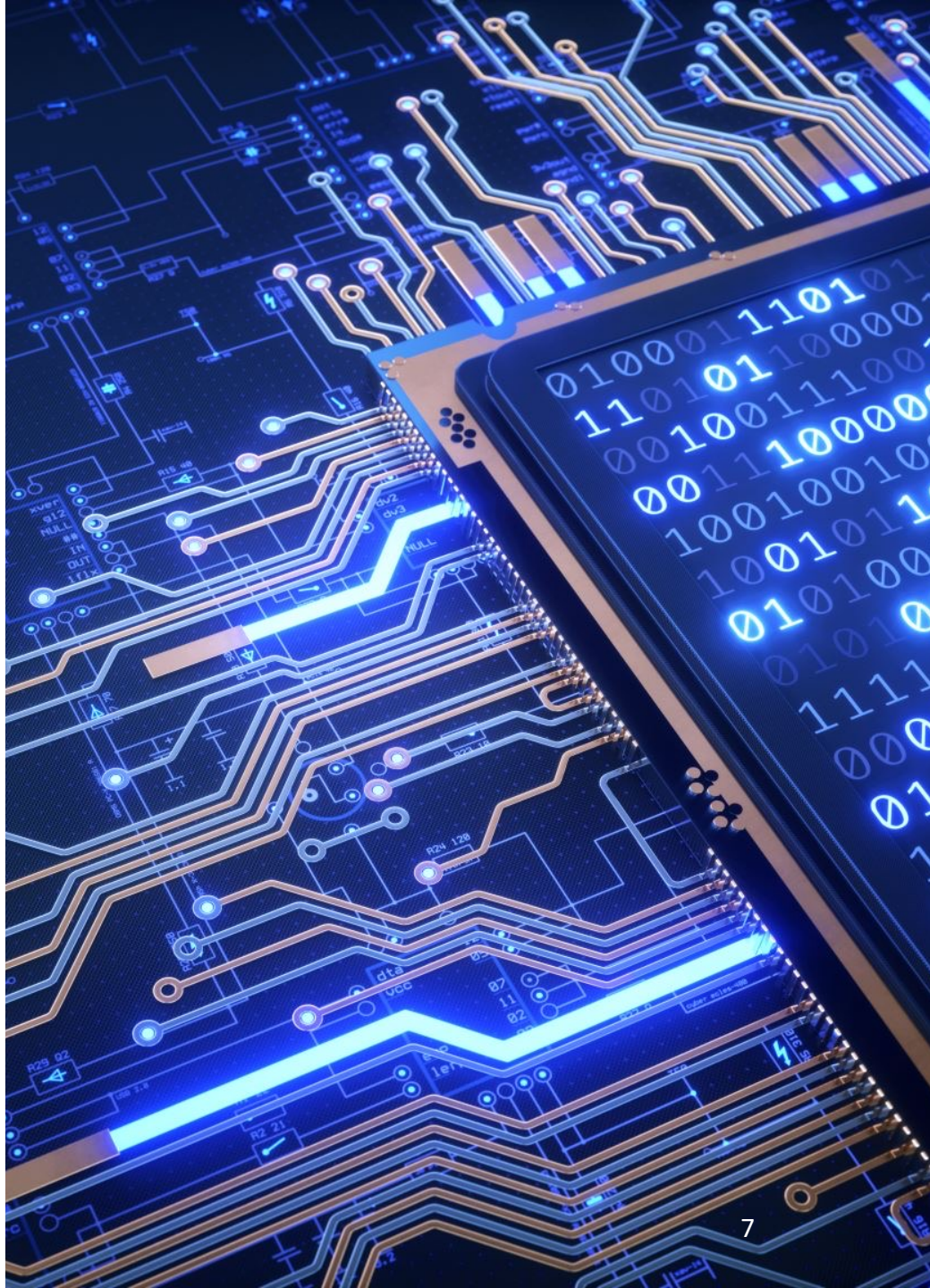


Binary Number	1's Complement
0000	1111
0001	1110
0010	1101
0011	1100
0100	1011
0101	1010
0110	1001
0111	1000
1000	0111
1001	0110
1010	0101
1011	0100
1100	0011
1101	0010
1110	0001
1111	0000

1’s Complement Table (4-bit Processor)

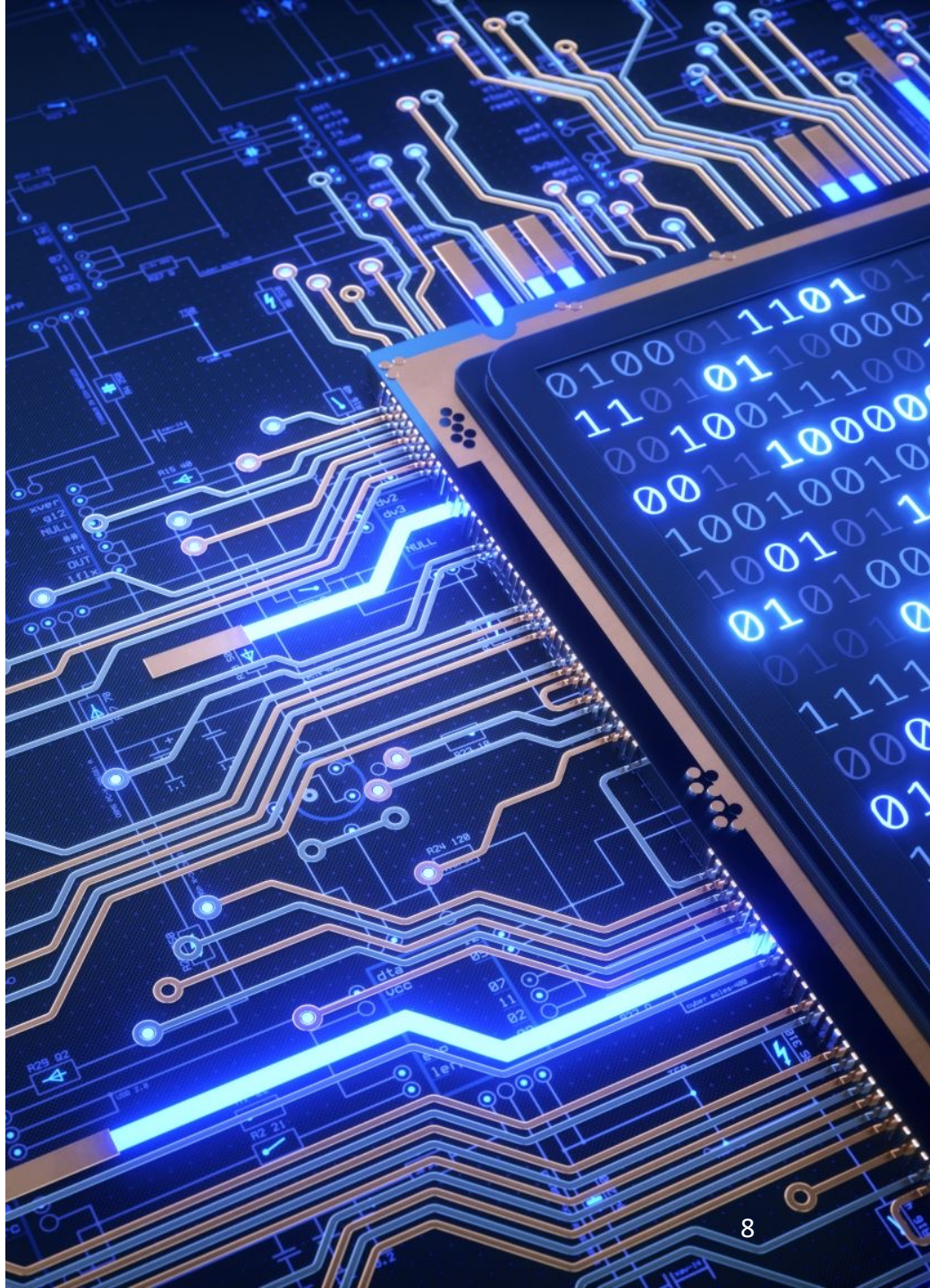
Number Systems: Complements

- Decimal Number System
  - 9’s complement achieved by subtracting each digit from 9
  - 10’s complement achieved by adding ‘1’ to the LSB of 9’s complement
  - E.g.: 2496
  - Ans: 7503 → 9’s complement
    7504 → 10’s complement



Number Systems: Complements

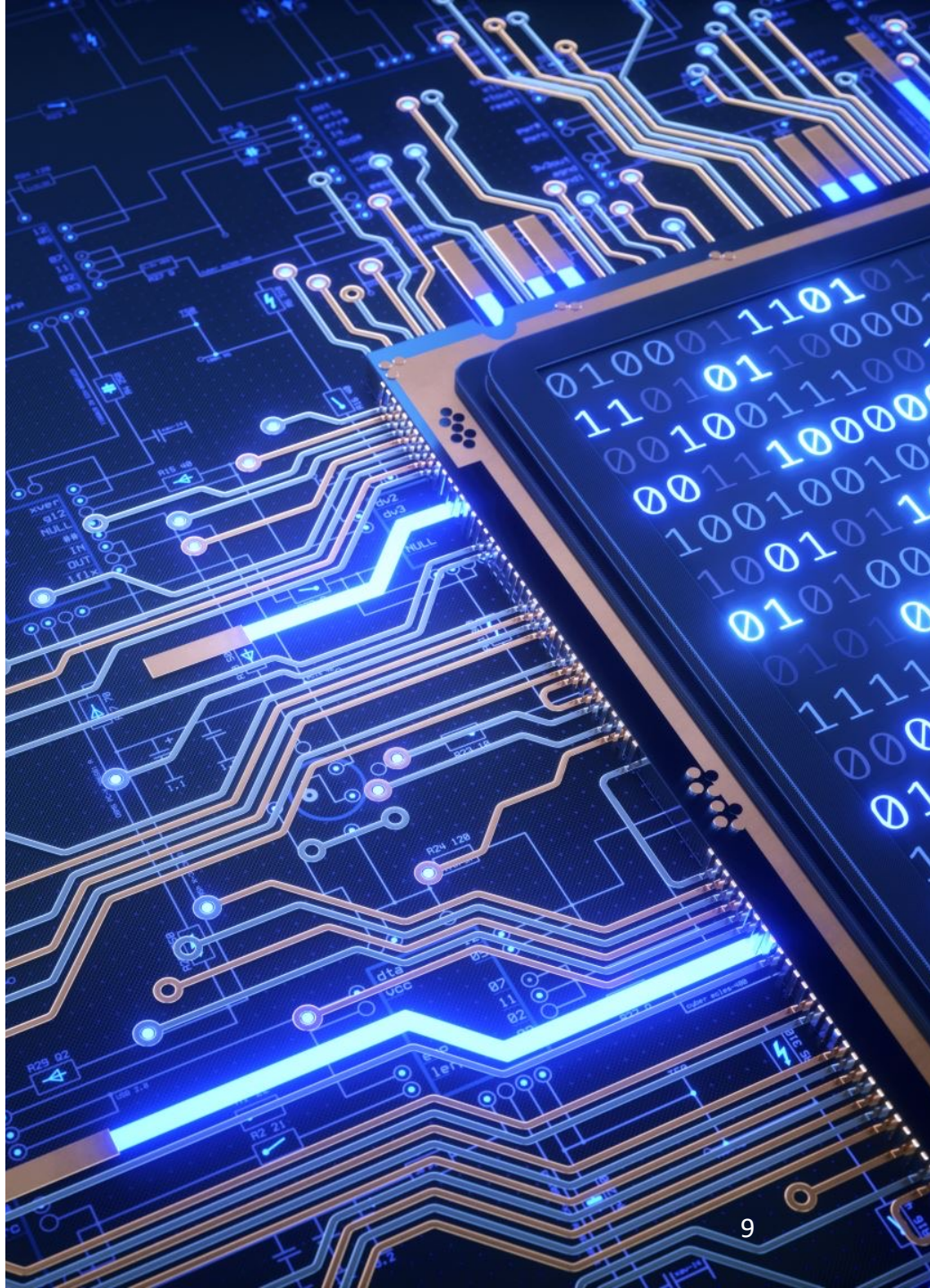
- Octal Number System
  - 7’s complement achieved by subtracting each digit from 7
  - 8’s complement achieved by adding ‘1’ to the LSB of 7’s complement
  - E.g.: 562
  - Ans: 215 → 7’s complement
    216 → 8’s complement



Number Systems: Complements

- Hexadecimal Number System
  - 15's complement achieved by subtracting each digit from 15
  - 16's complement achieved by adding '1' to the LSB of 15's complement
  - E.g.: 3BF

  - Ans: C40 → 15's complement
    C41 → 16's complement



Give this a try

The 7's complement of a certain octal number is 5264. Determine the binary and hexadecimal equivalents of that octal number.

Answer:

The 7's complement = 5264

Therefore, the octal number = $(2513)_8$

The binary equivalent = $(010\ 101\ 001\ 011)_2 = (10101001011)_2$

Also, $(10101001011)_2 = (101\ 0100\ 1011)_2 = (0101\ 0100\ 1011)_2 = (54B)_{16}$

Therefore, the hex equivalent of $(2513)_8 = (54B)_{16}$ and the binary equivalent of $(2513)_8 = (10101001011)_2$

Give this a try

Subtract $(1110.011)_2$ from $(11011.11)_2$ using basic rules of binary subtraction and verify the result by showing equivalent decimal subtraction.

Solution

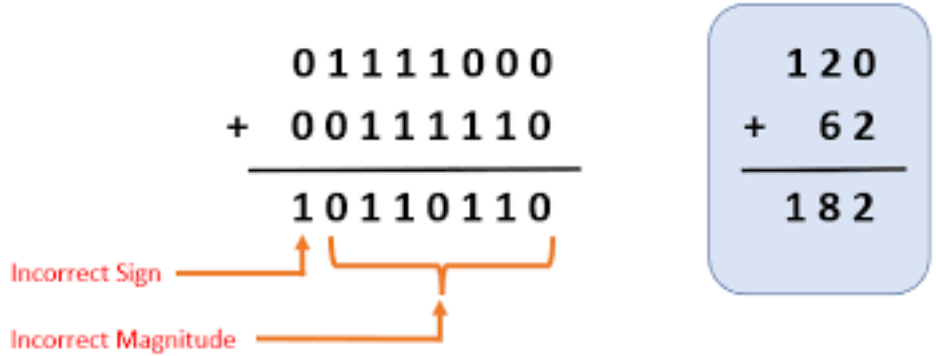
The minuend and subtrahend are first modified to have the same number of bits in the integer and fractional parts. The modified minuend and subtrahend are $(11011.110)_2$ and $(01110.011)_2$ respectively:

$$\begin{array}{r} 11011.110 \\ -\ 01110.011 \\ \hline 01101.011 \end{array}$$

The decimal equivalents of $(11011.110)_2$ and $(01110.011)_2$ are 27.75 and 14.375 respectively. Their difference is 13.375, which is the decimal equivalent of $(01101.011)_2$.

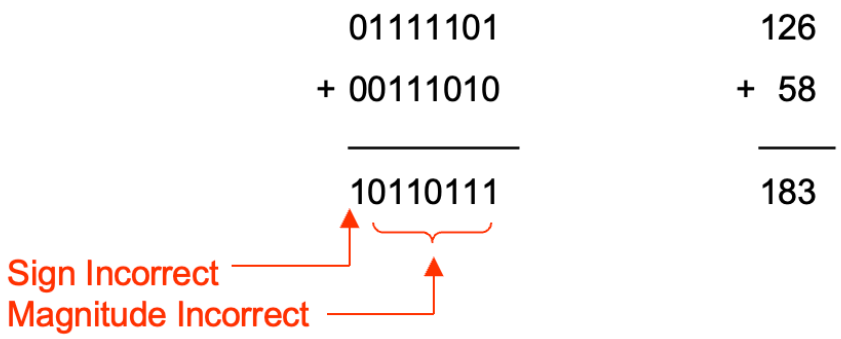
Overflow

- Note that, digital systems run on a fixed number of digits.
- If the result is too large to fit in the available digits, addition is said to overflow.
  - when a CPU's registers are unable to hold the sum of adding binary digits
  - The maximum number that can be stored in an 8-bit processor is 11111111 (or 255).



Overflow

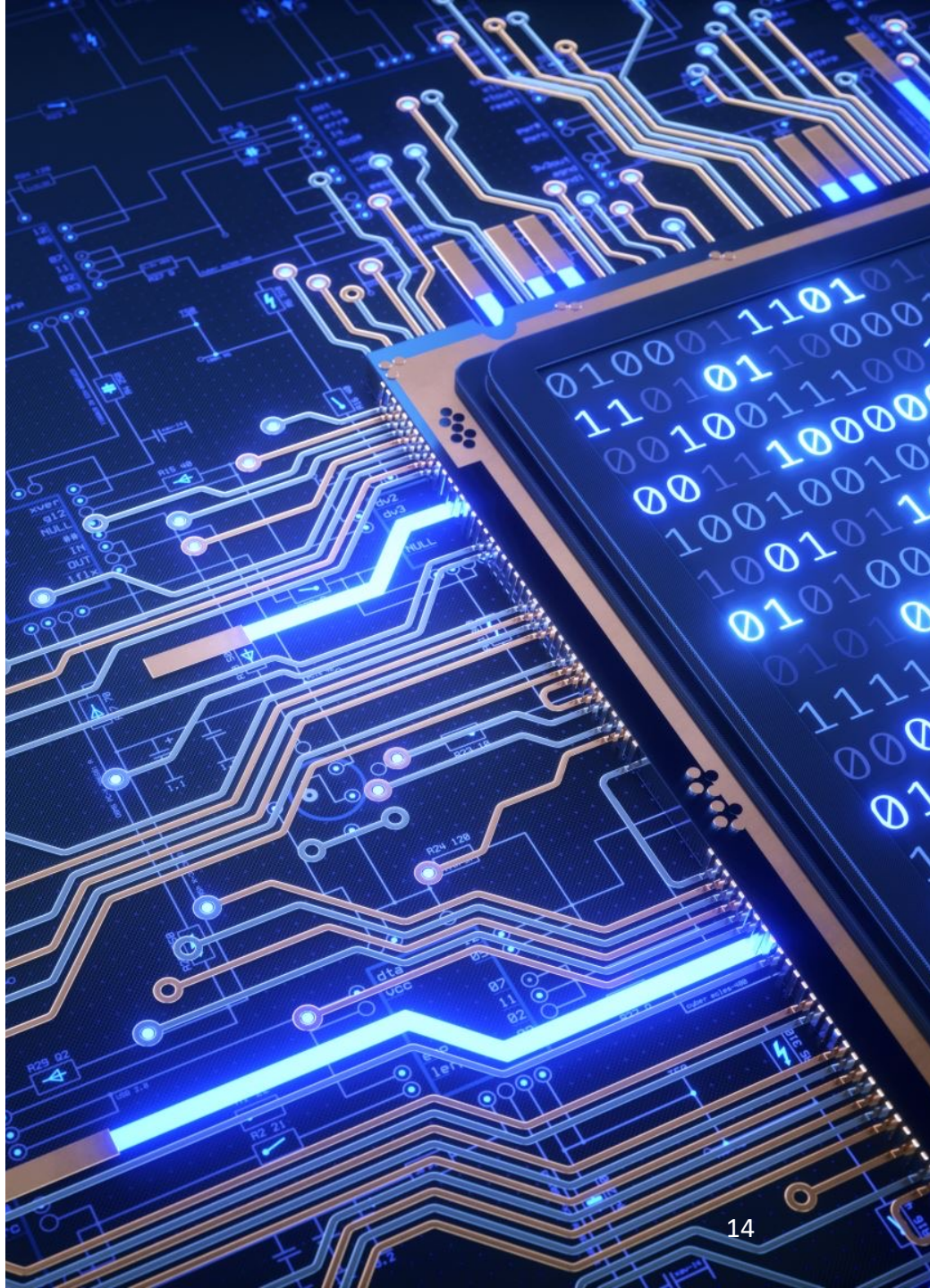
- When the number of bits in the sum exceeds the number of bits in the addend or augend, overflow occurs.
- Overflow is indicated by the wrong sign.
- Overflow occurs when
  - both numbers are positive or negative
  - two negative numbers are added, and an answer comes positive
  - two positive numbers are added, and an answer comes as negative



Signed Numbers: Addition

- The parts of an addition function are:
  - Augend - The first number
  - Addend - The second number
  - Sum - The result

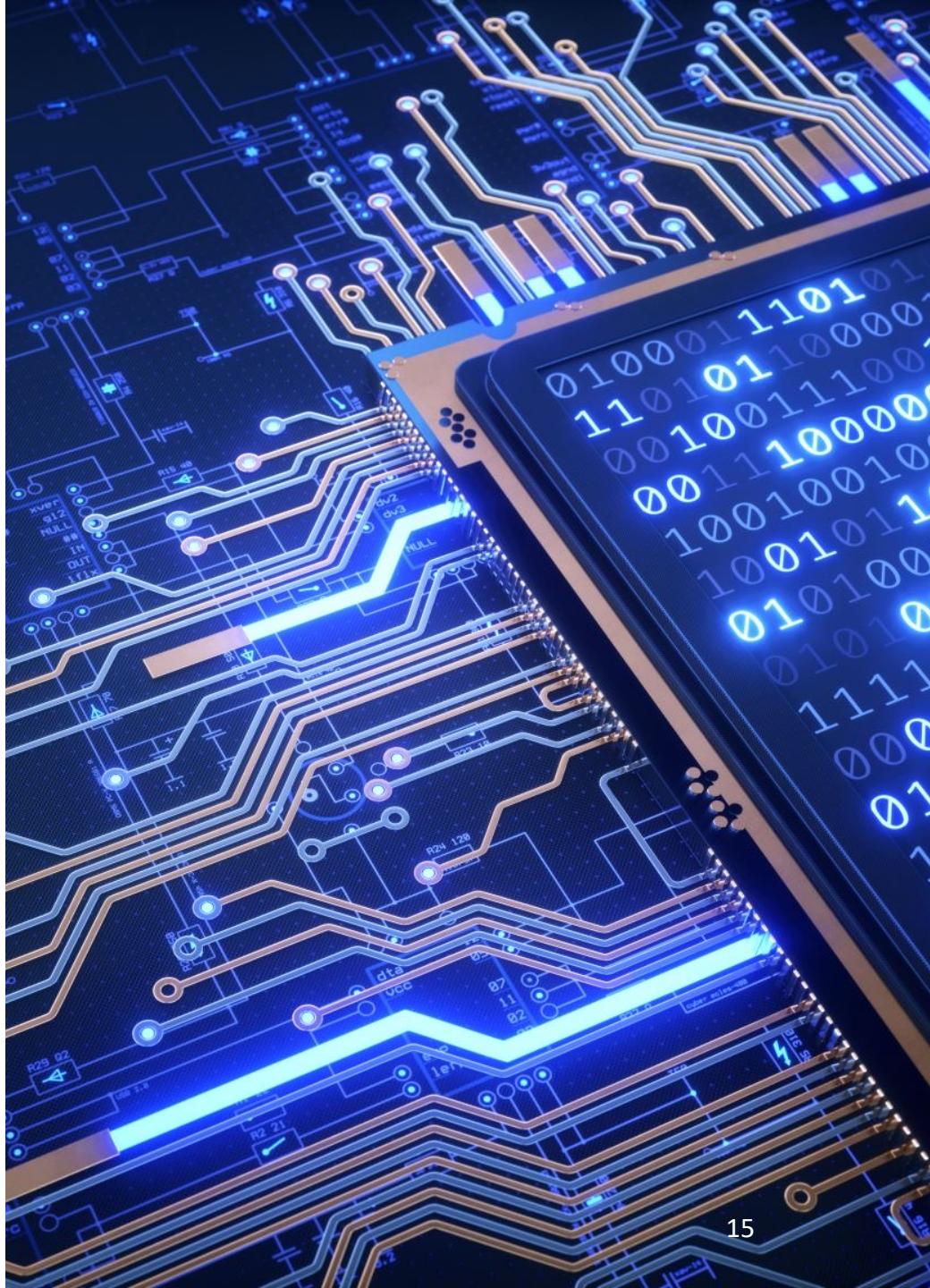
Numbers are added pairwise



Signed Numbers: Addition

Four conditions for adding numbers:

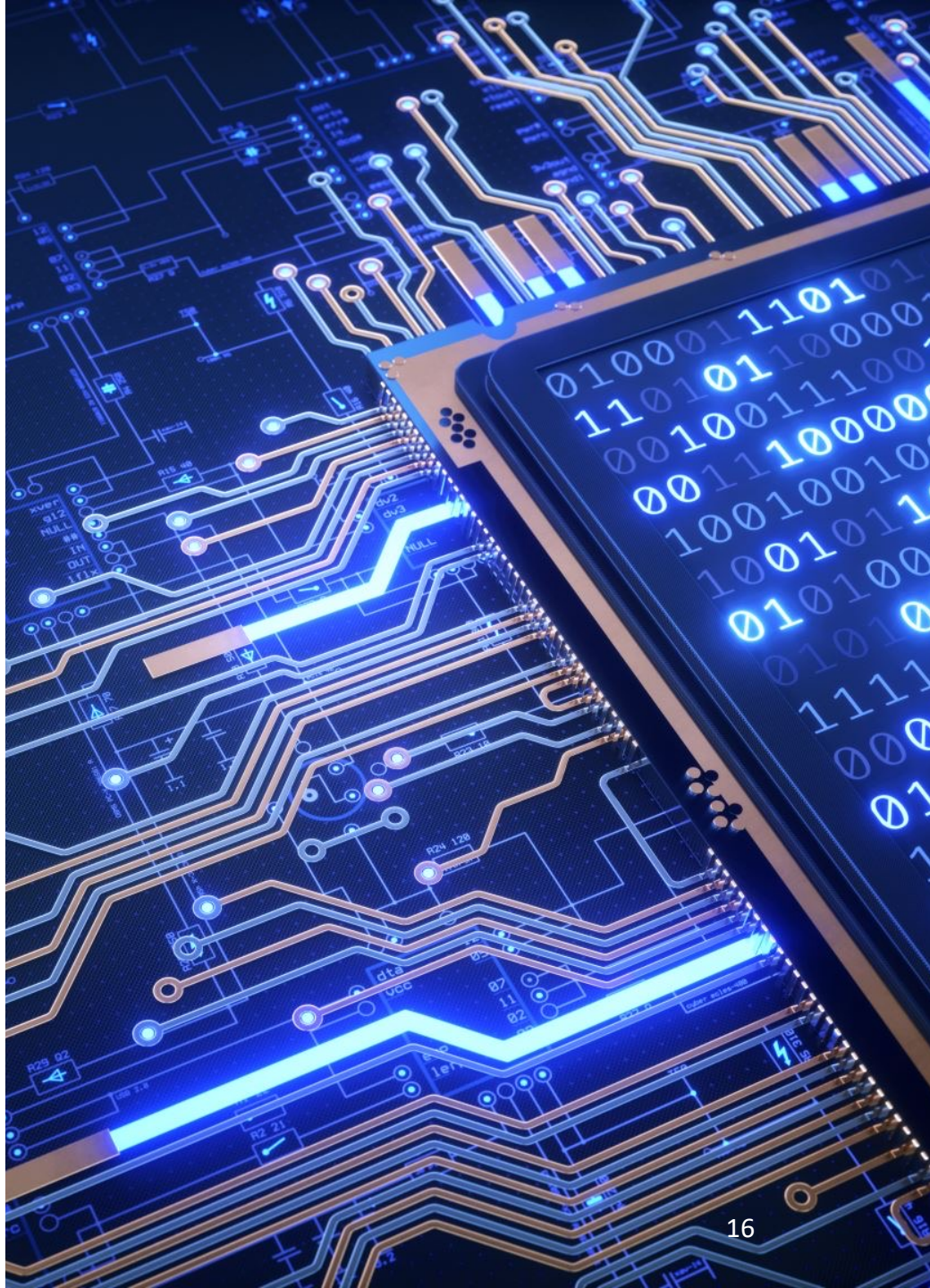
- Both numbers are positive.
- A positive number that is larger than a negative number.
- A negative number that is larger than a positive number.
- Both numbers are negative.



Signed Numbers: Addition

Signs for Addition

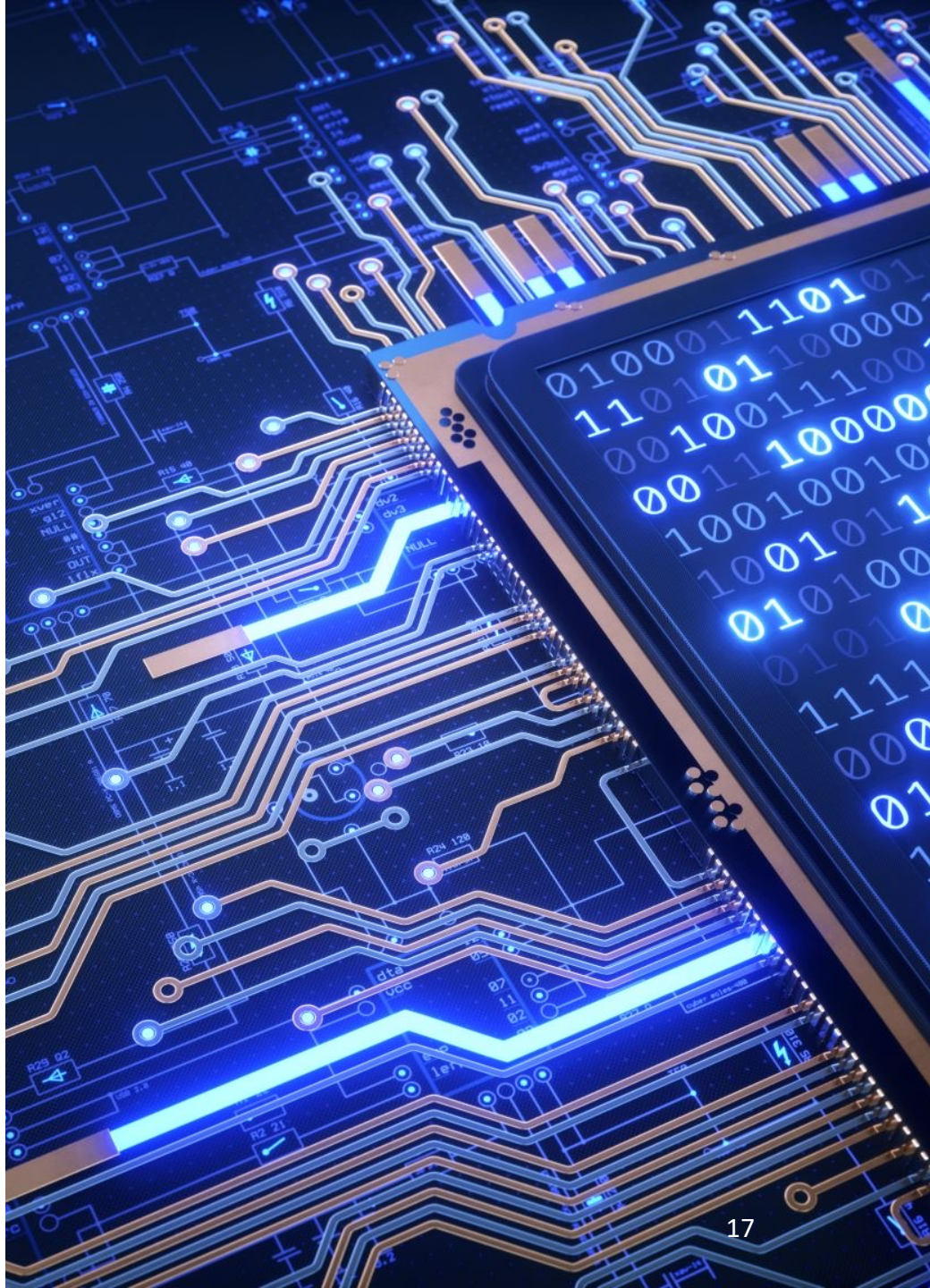
- When both numbers are positive, the sum is positive.
- When the larger number is positive and the smaller is negative, the sum is positive.
    - The carry is discarded.



Signed Numbers: Addition

Signs for Addition

- When the larger number is negative and the smaller is positive, the sum is negative (2's complement form).
- When both numbers are negative, the sum is negative (2's complement form).
  - The carry bit is discarded.



Example (using 8-bit processor)

- **Find the sum of positive 7 and positive 4**

```
 00000111      7
+00000100     + 4
 00001011     11
```

- **Find the sum of positive 15 and negative 6**

```
 00001111      15
+11111010     + -6
1 00001001      9
```

Discard carry bit →

- **Find the sum of positive 16 and negative 24**

```
 00010000      16
+11101000     + -24
 11111000      -8
```

Sign bit of sum is negative means sum is in 2's complement form

- **Find the sum of negative 5 and negative 9**

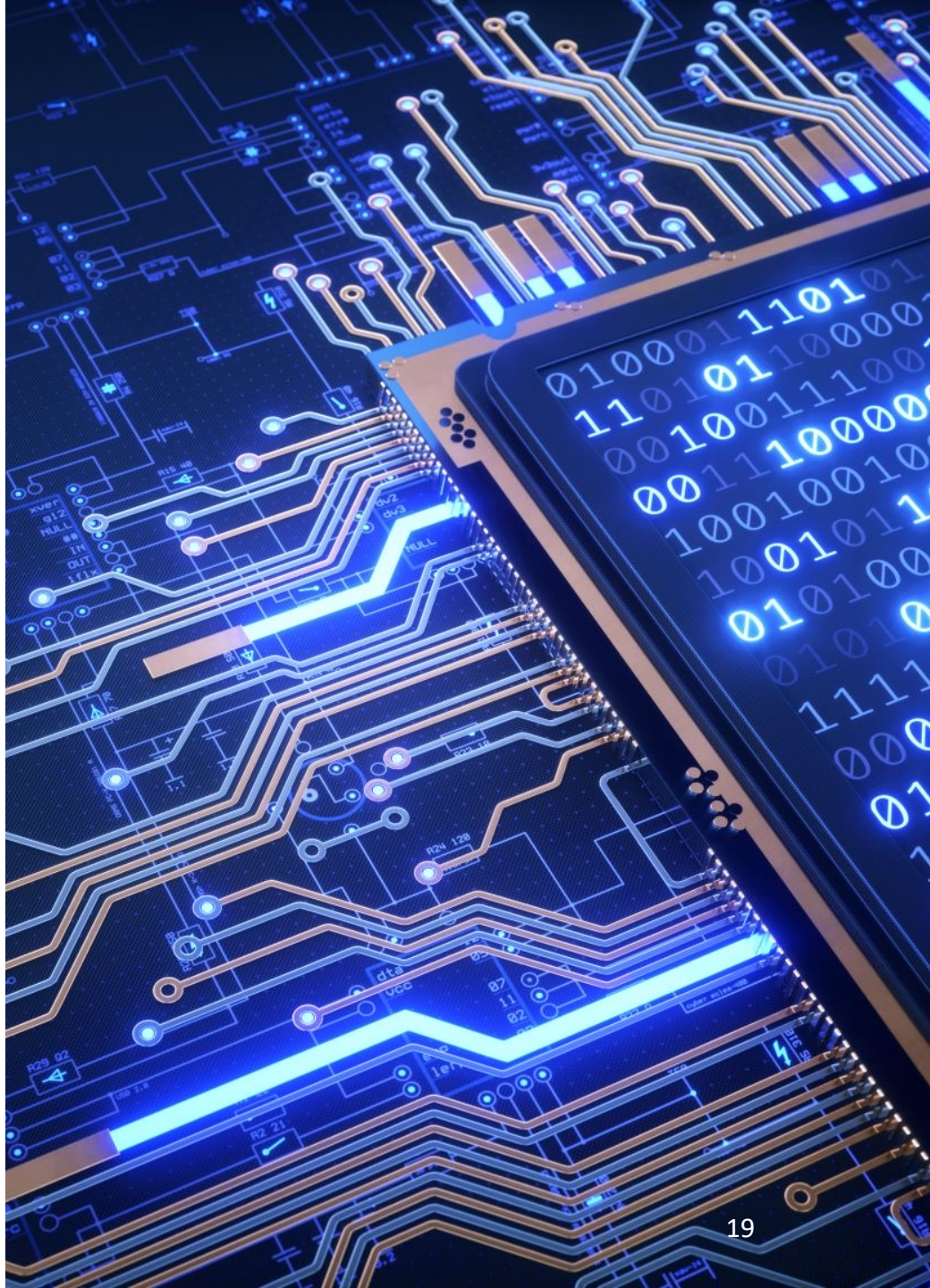
```
 11111011      -5
+11110111     + -9
1 11110010     -14
```

Discard carry bit →

Signed Numbers: Subtraction

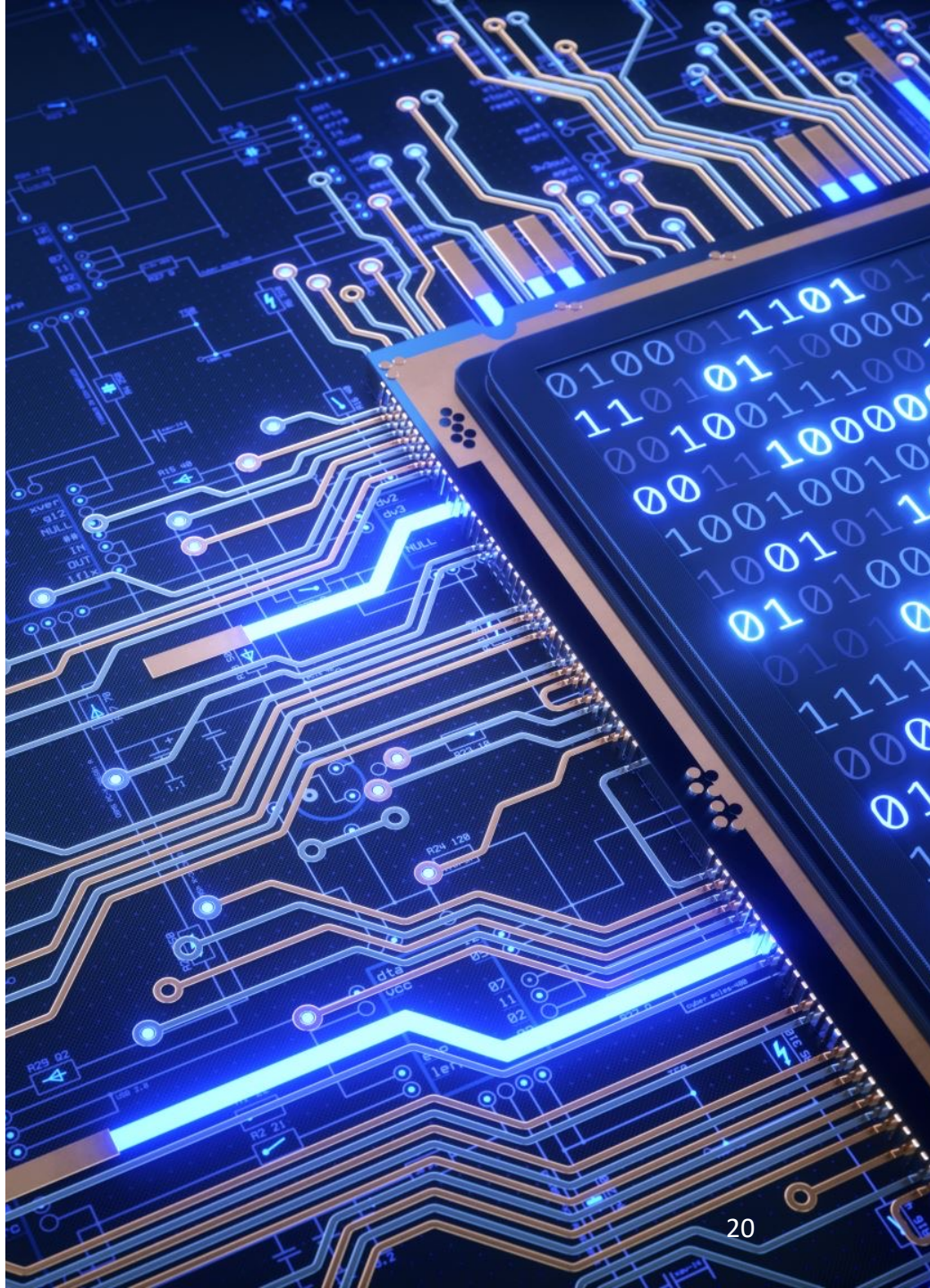
- The parts of a subtraction function are:
  - Minuend - The first number
  - Subtrahend - The second number
  - Difference - The result

Subtraction is a form of addition with the sign of the second number negative



Signed Numbers: Subtraction

- The sign of a positive or negative binary number is changed by taking its 2's complement
- To subtract two signed numbers, take the 2's complement of the subtrahend and add.
  - Discard any final carry bit.



Exercise (using 8-bit processor)

- Subtract 3 from 8
-

	00001000	8	Minuend
	+ 11111101	- 3	Subtrahend
Discard carry bit → 1	00000101	5	Difference

- Subtract negative 9 from 12

00001100	12	
+00001001	- -9	
00010101	21	

- Subtract 19 from negative 25

11100111	-25	
+11101101	- 19	
Discard carry bit → 1 11010100	-44	

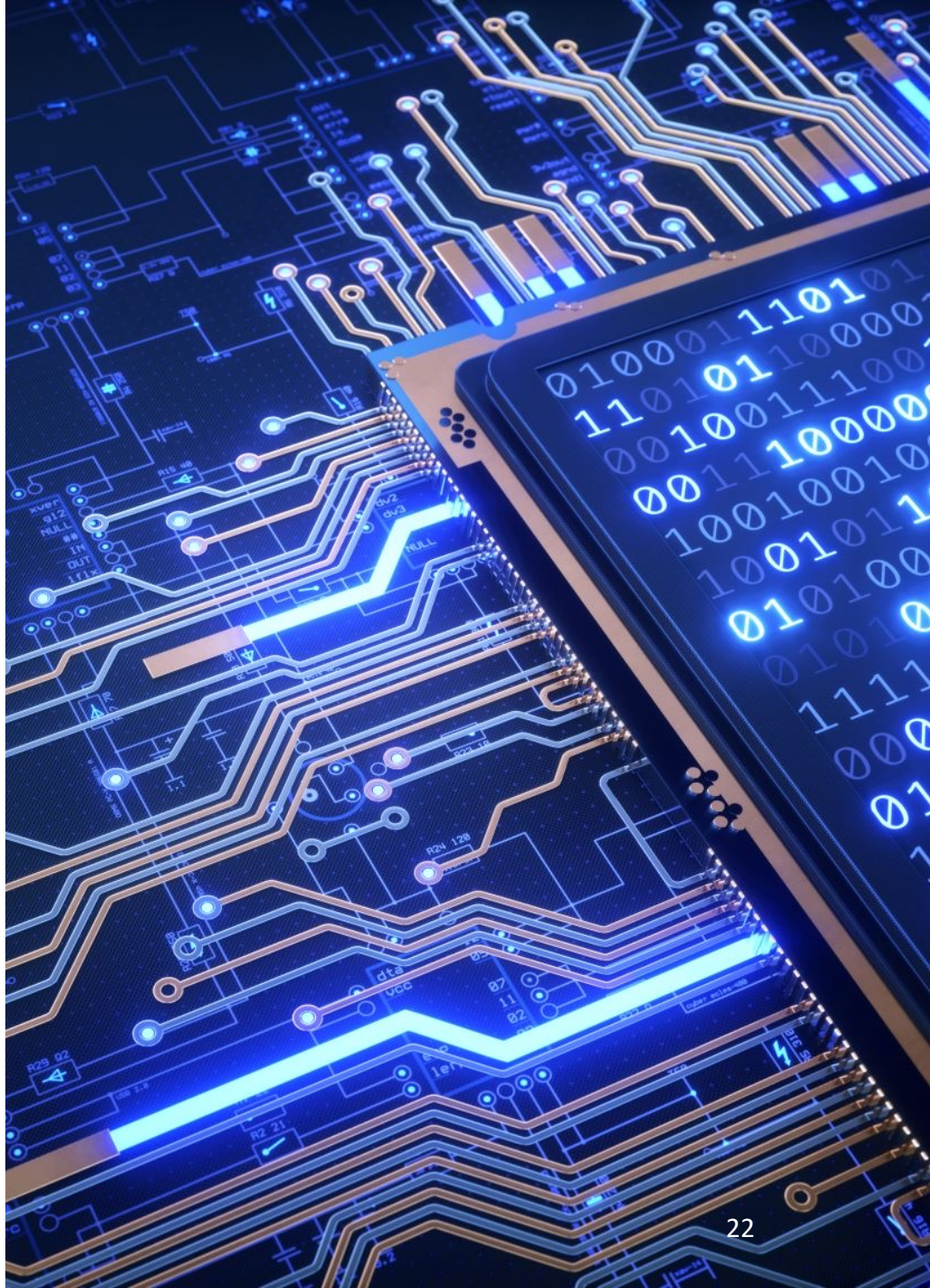
- Subtract negative 30 from negative 120

10001000	-120	
+00011110	- -30	
10100110	-90	

Signed Numbers: Multiplication

- The parts of a multiplication function are:
  - Multiplicand - First number
  - Multiplier - Second number
  - Product - Result

A number multiplied by the multiplier is equal to the number of times the original number is added to itself

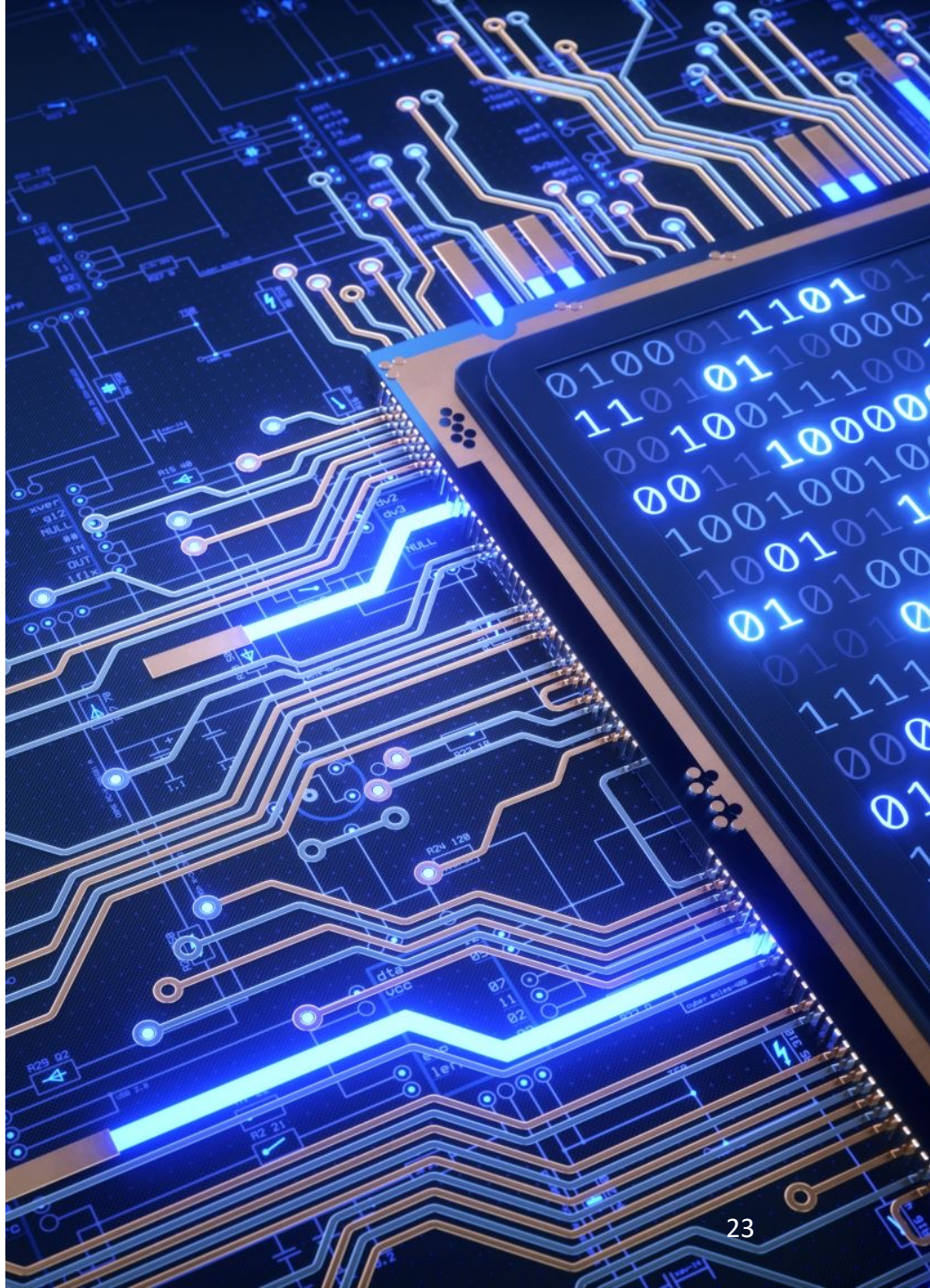


Signed Numbers: Multiplication

There are two methods for multiplication:

- Direct addition
  - add multiplicand multiple times equal to the multiplier
  - Can take a long time if multiplier is large
- Partial products
  - Similar to long hand multiplication
  - Repeated Left-Shift and Add Algorithm

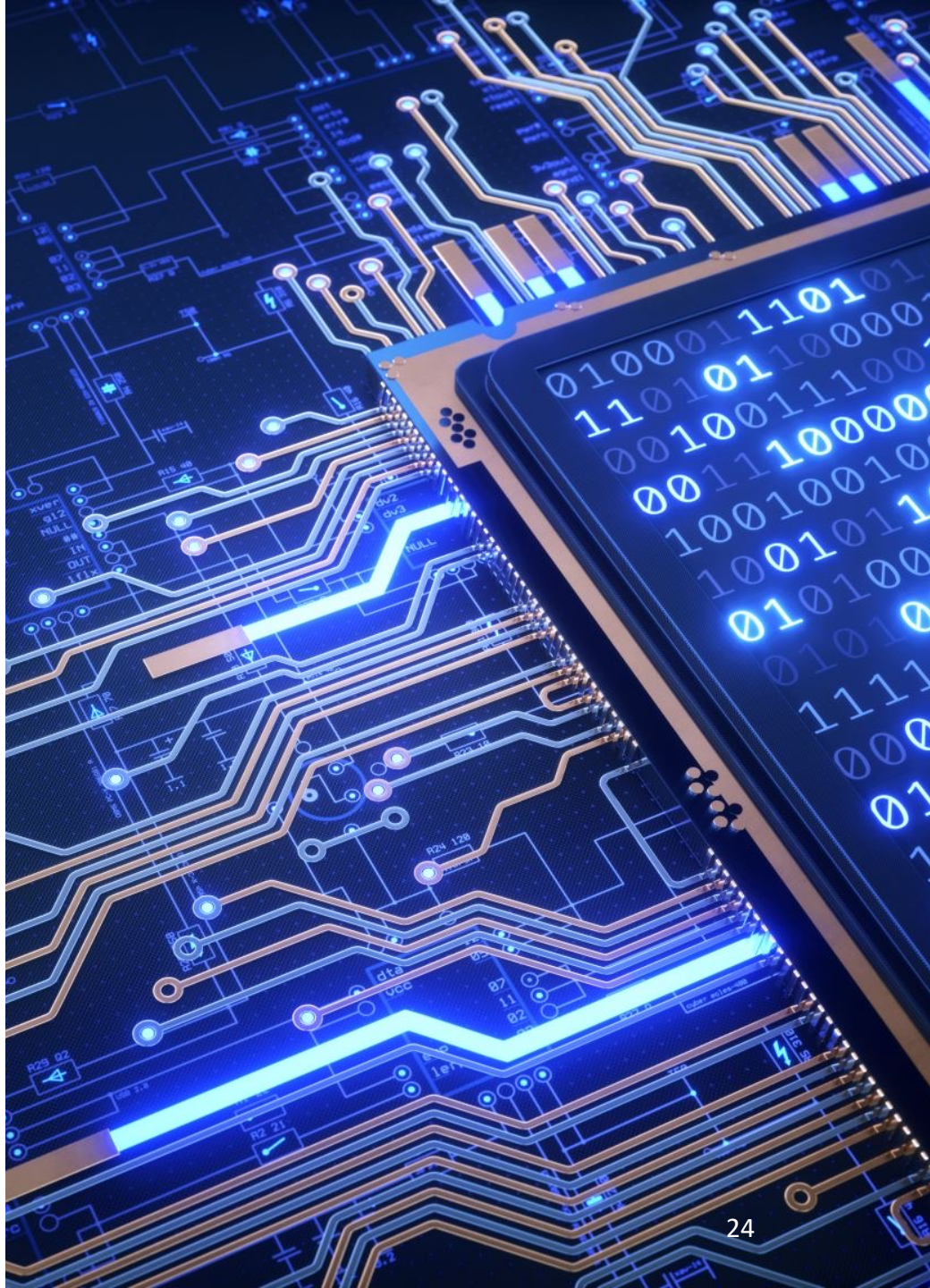
The most widely utilised approach is the partial products method



Signed Numbers: Multiplication

Multiplication of Signed Numbers

- If the signs are the same, the product is positive.
  (+ X + = + or - X - = +)
- If the signs are different, the product is negative.
  (+ X - = - or - X + = -)



Example (using 8-bit processor)

- Multiply 3 by negative 5
- Both numbers must be in uncomplemented form

Operands have opposite signs, so product will be negative

	00000011	Multiplicand
x	00000101	Multiplier
	00000011	First partial product
+	0000000	Second partial product
	00000011	Sum of 1st and 2nd
+	000011	Third partial product
	00001111	Sum and Final Product

Final result is negative, so take 2's complement
11110001 is the result which in decimal is -15

Exercise on Multiplication

- Multiply 100.01_2 by 10.1_2 using the 'repeated left-shift and add' algorithm. Show the equivalent decimal multiplication to validate the outcome.
- ***Solution:***

$$
\begin{array}{r}
1\ 0\ 0\ 0\ 1 \\
\times\ 1\ 0\ 1 \\
\hline
1\ 0\ 0\ 0\ 1 \\
0\ 0\ 0\ 0\ 0 \\
1\ 0\ 0\ 0\ 1 \\
\hline
1\ 0\ 1\ 0\ 1\ 0\ 1 \\
\hline
\end{array}
$$

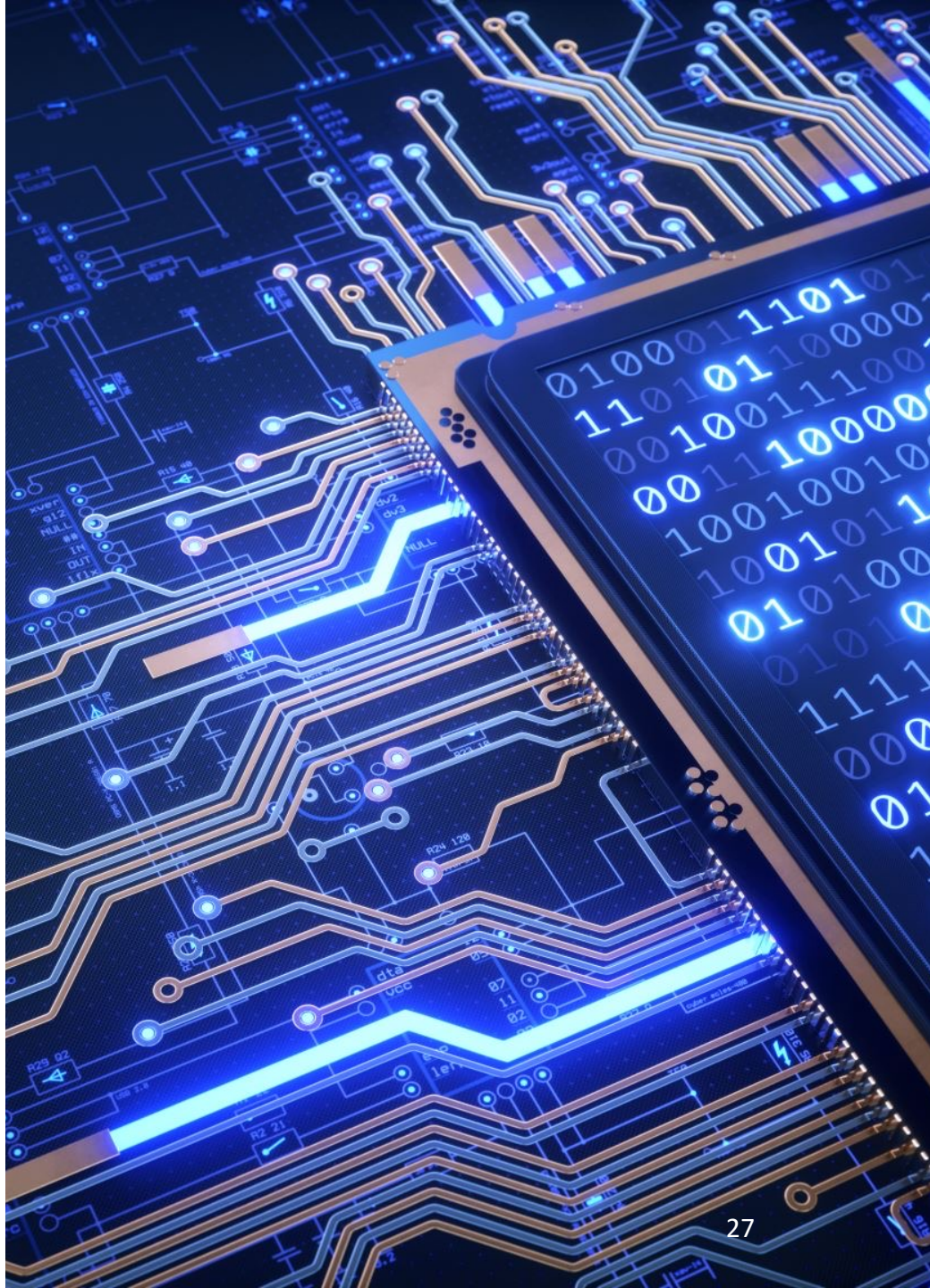
The multiplication result is then given by placing the binary point three bits after the LSB, which gives $(1010.101)_2$ as the final result. Also, $(100.01)_2 = (4.25)_{10}$ and $(10.1)_2 = (2.5)_{10}$. Moreover, $(4.25)_{10} \times (2.5)_{10} = (10.625)_{10}$ and $(1010.101)_2$ equals $(10.625)_{10}$, which verifies the result.

Signed Numbers: Division

- The parts of a division operation are:
  - Dividend
  - Divisor
  - Quotient

$$\frac{\text{dividend}}{\text{divisor}} = \text{quotient}$$

Division is the same as taking the divisor out of the dividend and multiplying the result by the quotient.



Signed Numbers: Division

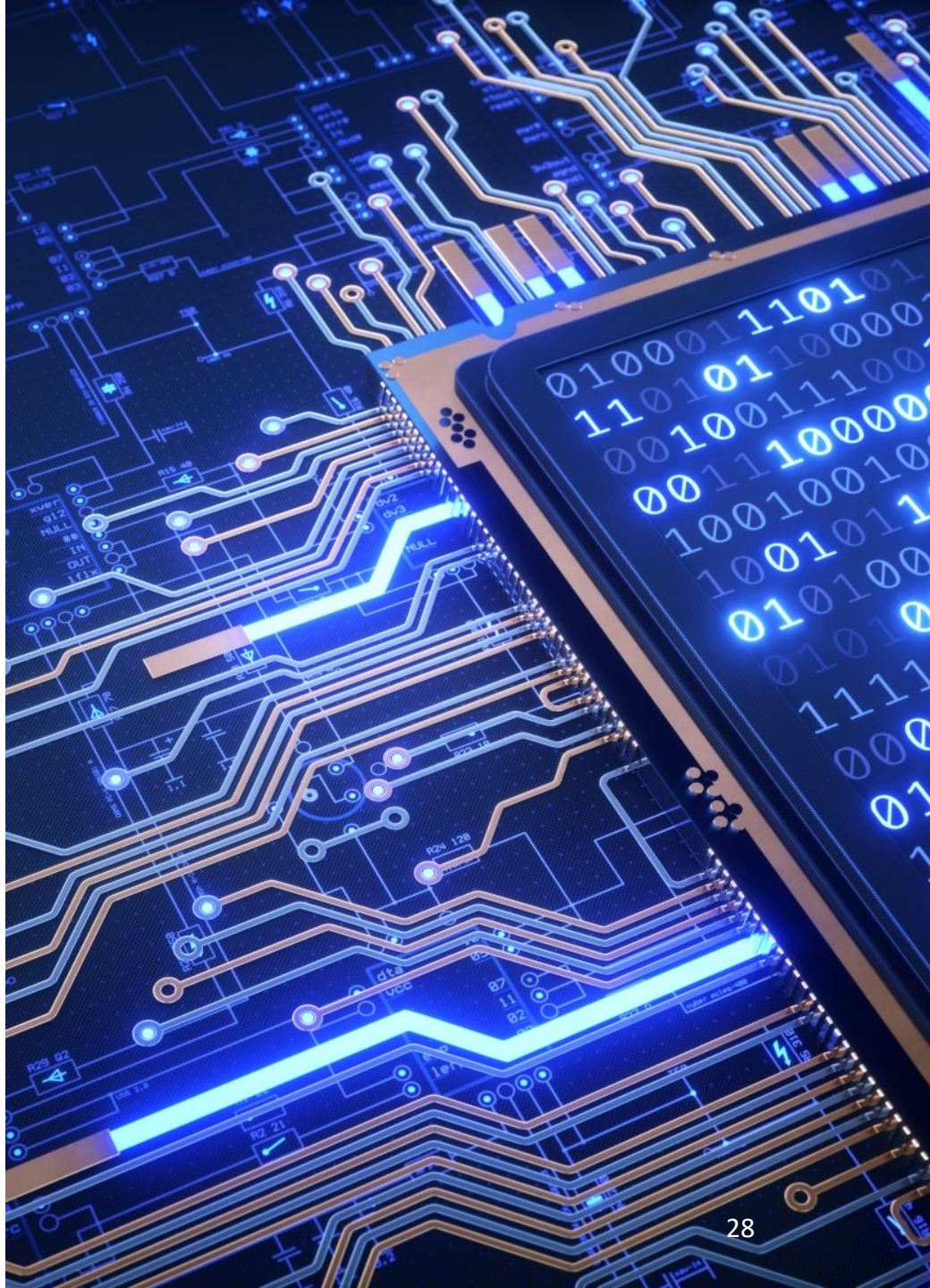
Division of Signed Numbers

- If the signs are the same, the quotient is positive

  (+ ÷ + = + or - ÷ - = +)

- If the signs are different, the quotient is negative

  (+ ÷ - = - or - ÷ + = -)



Let’s try this

- Divide 100 by 50
- Show your working based on how an an 8-bit processor will compute the arithmetic operation
- **Answer:**
- We know 100 divided by 50 is 2
- 100 → Dividend
- 50 → Divisor
- First convert both operands to binary
  - 100 → 01100100
  - 50 → 00110010
- Here, both the dividend and divisor must be uncomplemented prior to the division

Let's try this

Since the two operands are in the positive form, the quotient is also going to be positive. In that case, the quotient will be set to an initial value of zero.

quotient: 00000000

Take the divisor in binary from the dividend based on 2's complement (11001110).
Ignore the carry bit.

```
  01100100   Dividend
+ 11001110   Divisor in 2's complement
----------
1 00110010   1st Partial remainder
```

Since 1st subtraction is made, add 1 to quotient.

00000000 + 1 = 00000001

Subtract the divisor from the 1st partial remainder using 2's complement addition.

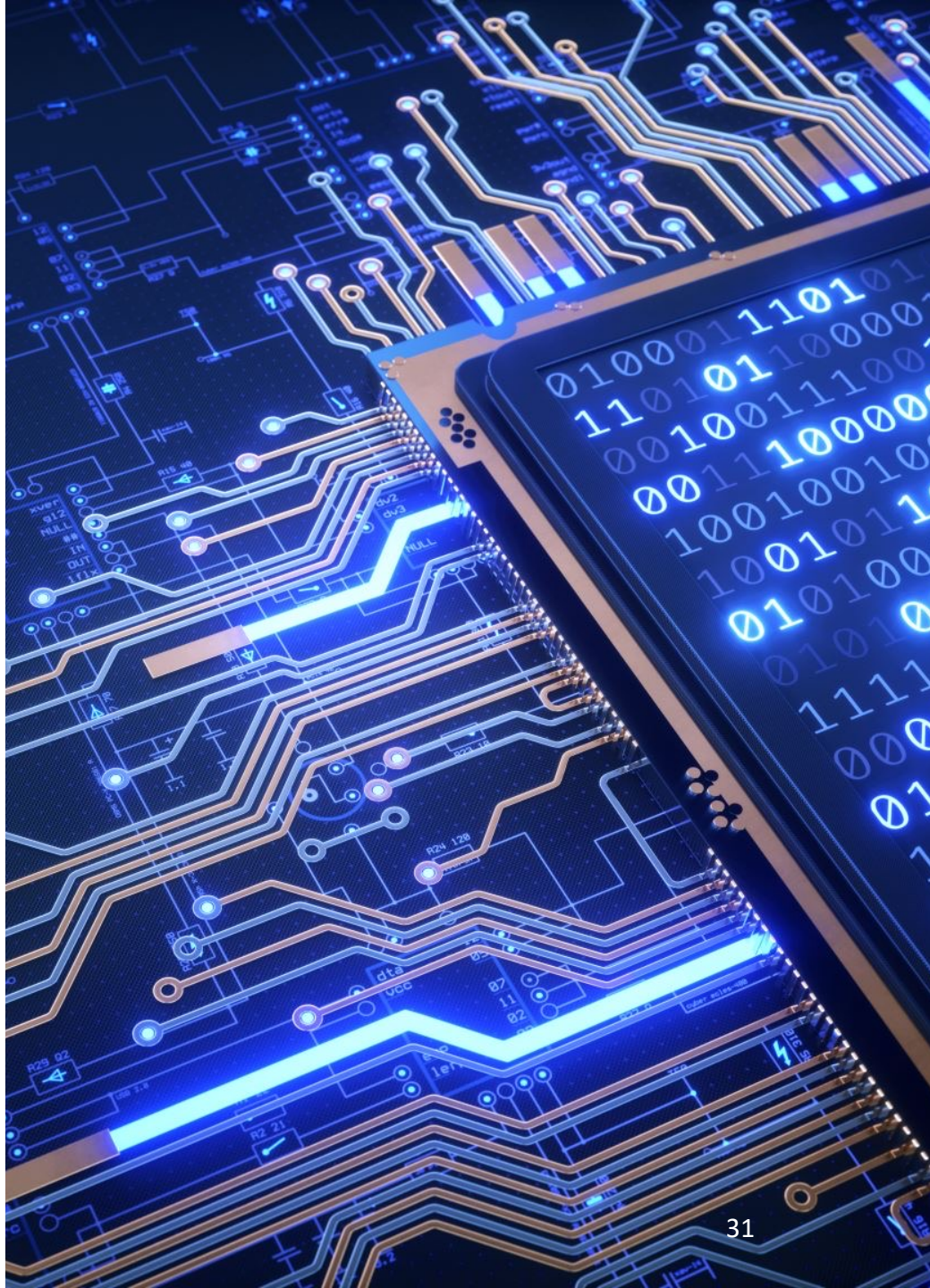
```
  00110010   1st Partial remainder
+ 11001110   Divisor in 2's complement
----------
1 00000000   zero remainder
```

Since 2nd subtraction is made, add 1 to quotient 00000001 + 1 = 00000010

The final result is 00000010 with a reminder of 00000000

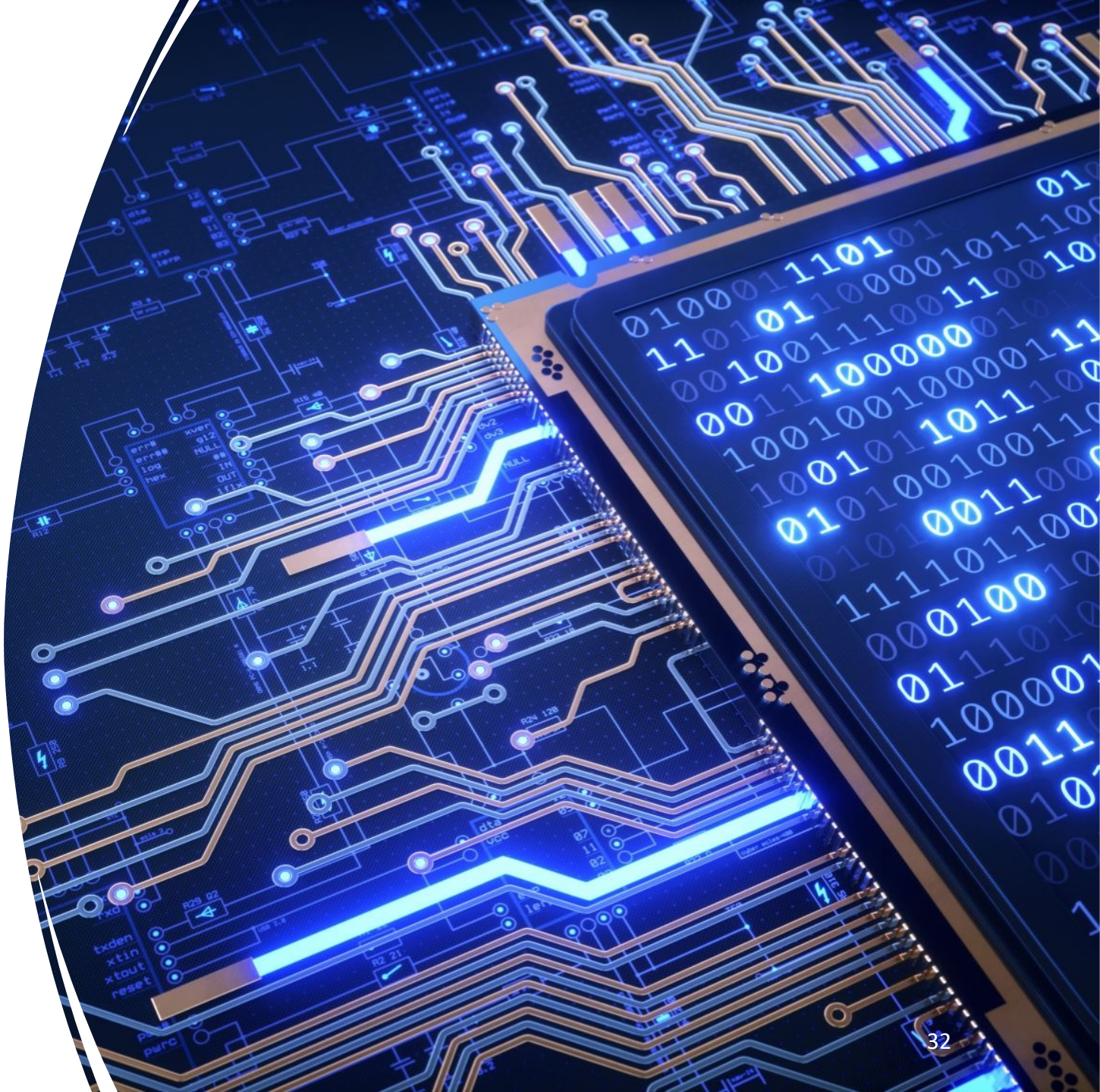
Application of Arithmetic Computation

- Computer Systems and Digital Electronics
- Digital Signal Processing
- Cryptography and Information Security
- Error Detection and Correction
- Computer Graphics and Image Processing
- Embedded Systems and IoT Devices
- Scientific Computing and Simulation
- Data Analysis and Machine Learning



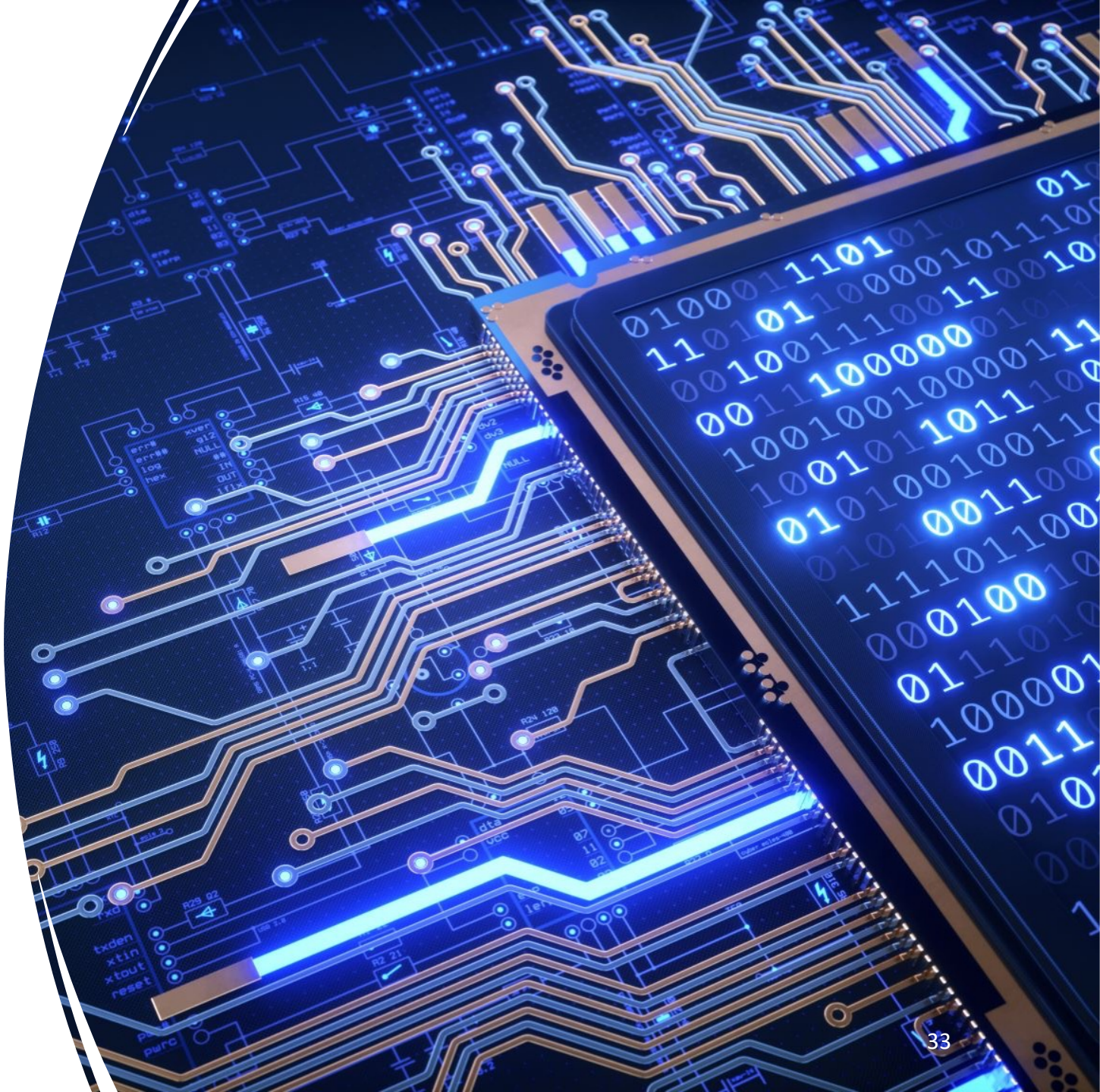
Summary

- Arithmetic operations
  - Addition
  - Subtraction
  - Multiplication
  - Division
- Complements provide avenue for negative arithmetic operations
- Applications of arithmetic computation



Reference

Maini, A. K. (2007). “Digital Electronics: Principles, Devices and Applications”. *John Wiley & Sons, Ltd*. ISBN: 978-0-470-03214-5
Chapter 3



See you next time!