Introduction to Digital Systems

Part I (4 lectures)

2023/2024

Introduction

Number Systems and Codes

Combinational Logic Design Principles

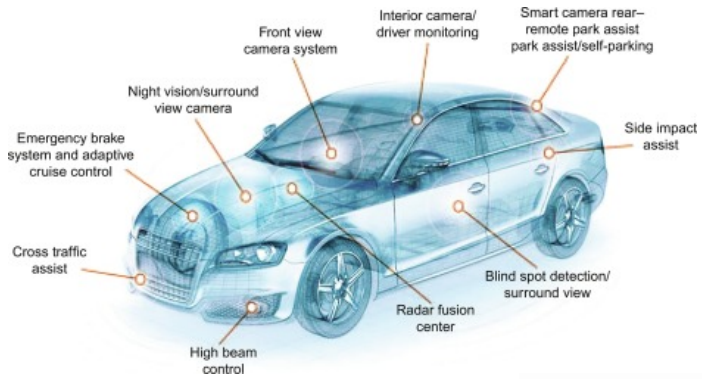
Arnaldo Oliveira, Augusto Silva, Iouliia Skliarova

Lecture 1 contents

- About digital design
- Number systems
  - Positional number systems
  - Binary, octal, and hexadecimal numbers
  - General positional-number-system conversions

Motivation

Electronics all around us: automotive, consumer products, communications, military and aerospace, medicine, etc.



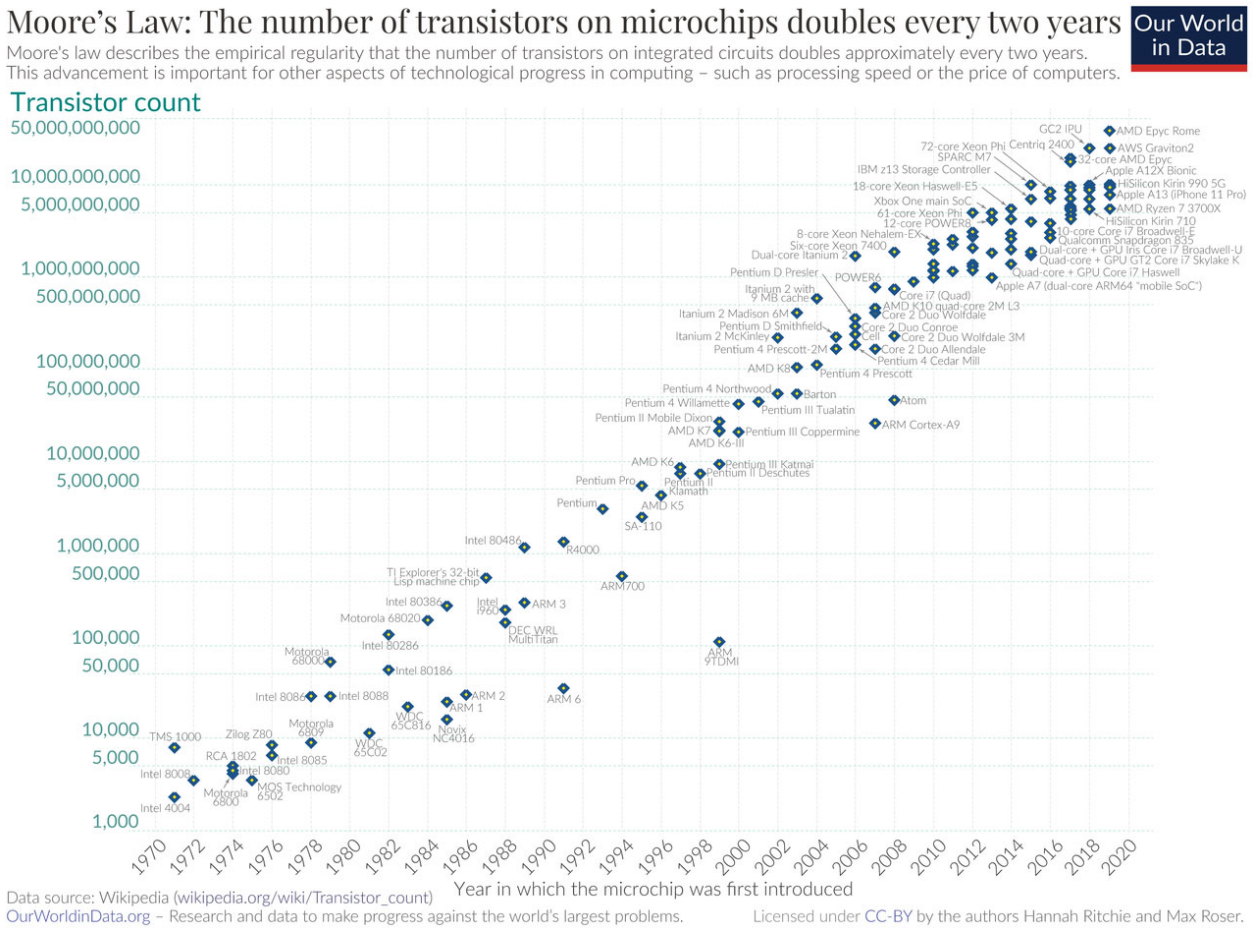
https://www.sciencedirect.com/topics/engineering/automotive-electronics
http://www.beadelectronics.com/military-and-aerospace-electronics-applications
https://electronicsforu.com/technology-trends/medical-electronics-key-concerns-opportunities
https://www.ittechnologynews24.com/tag/online-consumer-electronics-market-review/
http://www.indiaeducation.net/engineering/engineering-branch/electronics-engineering-versus-electronics-communication-engineering.html

Question

- A lei de Moore

Exponential Growth

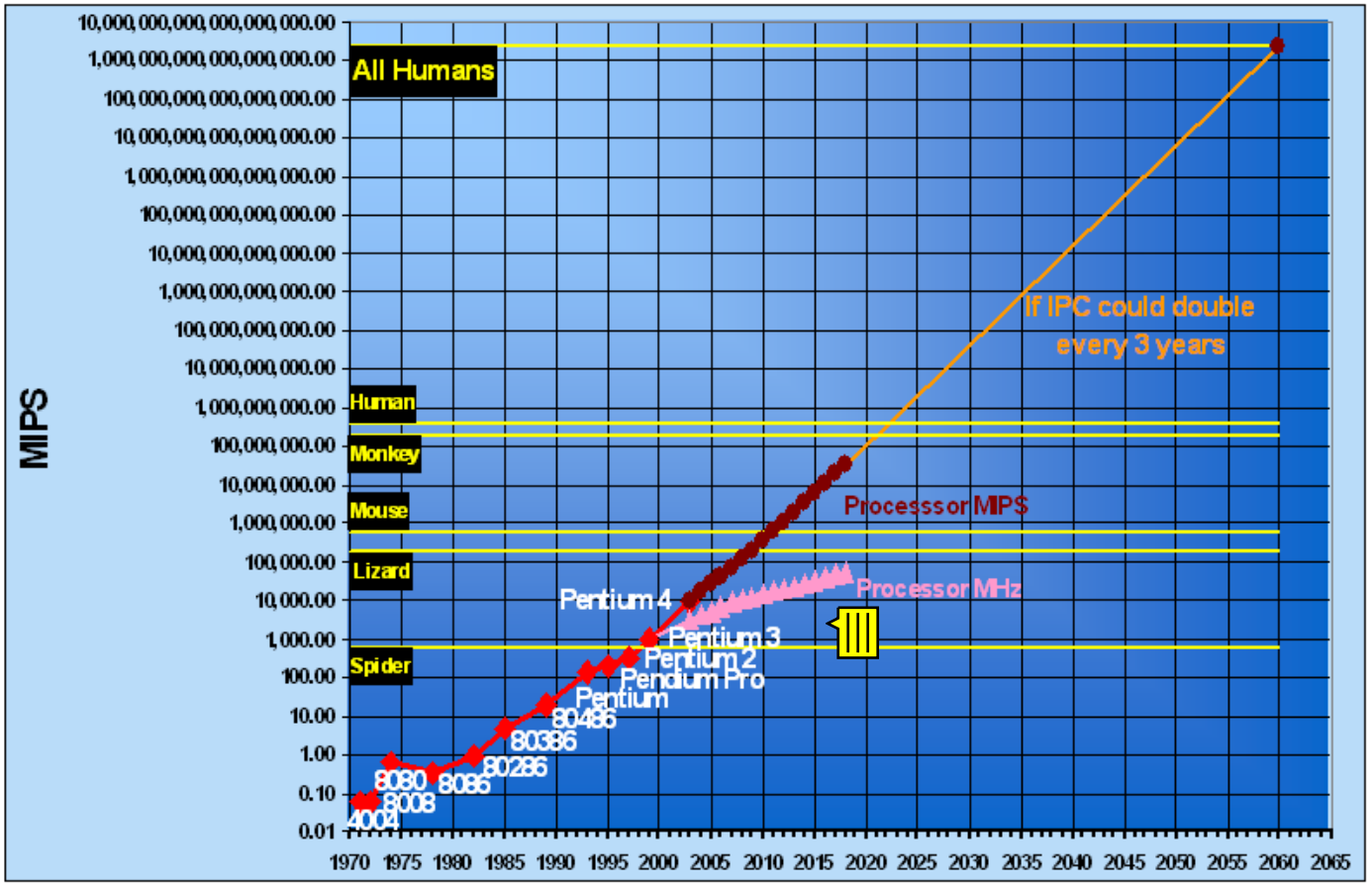
Moore's law (observed in 1965): the number of transistors on a computer chip doubles every 1.5-2 years



https://en.wikipedia.org/wiki/Moore%27s_law#/media/File:Moore's_Law_Transistor_Count_1970-2020.png

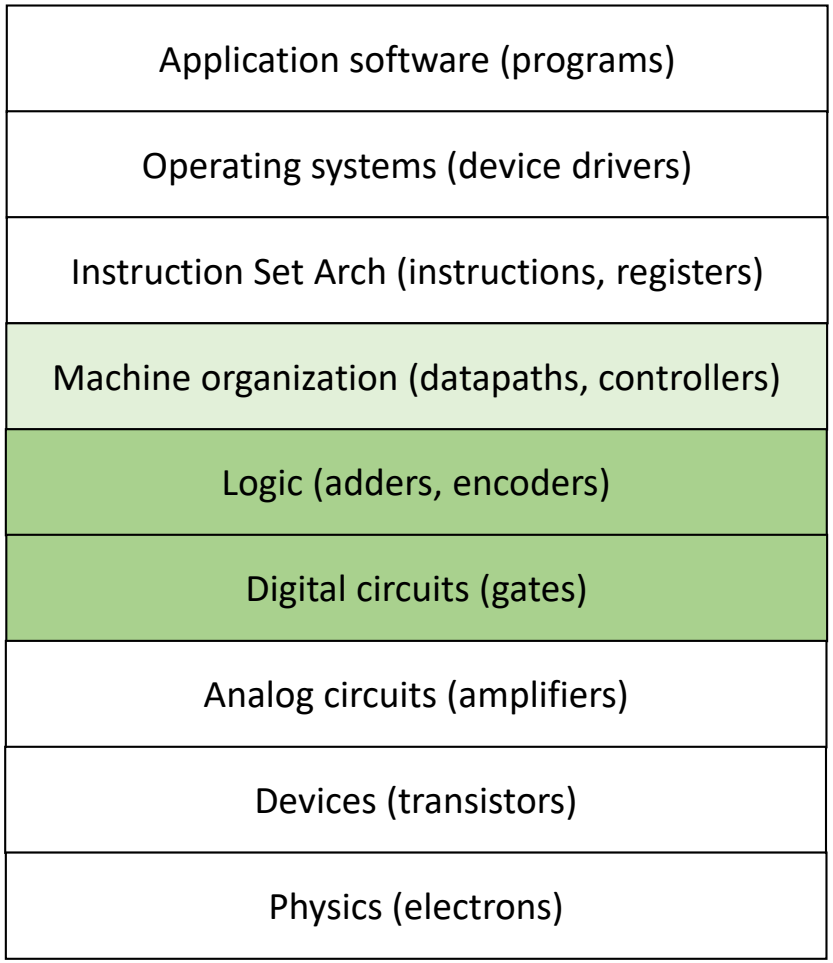
Exponential Growth

Processing power increases at about the same rate (~40%)



http://www.captec-group.com/

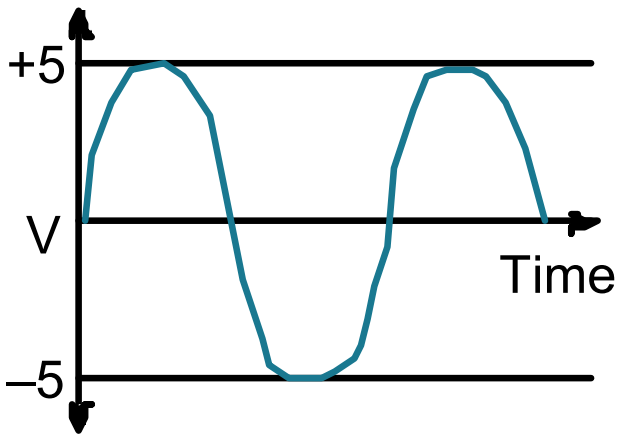
Course content



- Fundamentals of Boolean logic
- Encoding
- Combinational circuits
- Arithmetic units
- Synchronous circuits
- Finite state machines
- Timing and clocking
- Simulation

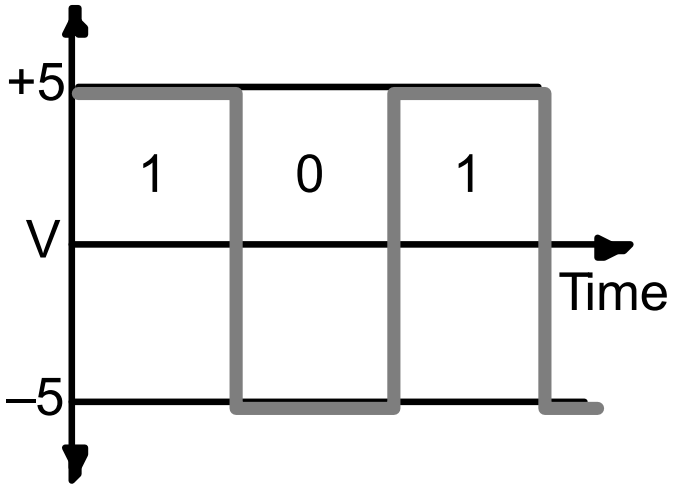
Digital abstraction

Analog: values vary over a broad range continuously



Digital: only assumes discrete values

- reproducibility of results
- ease of design
- programmability
- speed

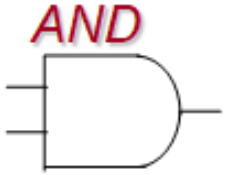


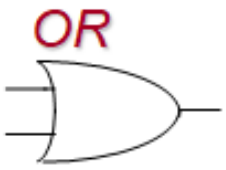
Boolean Algebra

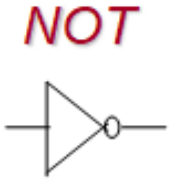
- Inputs and outputs can only have two discrete values:
  - physical domain (usually, voltages) (0V/5V)
  - mathematical domain : Boolean variables (true/ false, 0/1)
- **Boolean algebra** is used to analyze and describe the behavior of digital circuits
  - a symbolic variable, such as *x*, represents the condition of a logic signal
  - algebraic operators, such as AND, OR and NOT, represent **logic gates**
    - a **gate** is the most basic digital device and has one or more input and produces an output that is a function of the current input value(s)

Logic Gates

- AND – logical product
- OR – logical sum
- NOT - inversion







x	*y*	*x* and *y*
		xy
		$x \cdot y$
		$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

x	*y*	*x* or *y*
		$x+y$
		$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

x	not *x*
	$\bar{x}$
	x'
0	1
1	0

Number Systems

- Digital systems are built from circuits that process binary digits
- Very few real-life problems are based on binary numbers or any numbers at all
- Some correspondence must be established between the binary digits processed by digital circuits and real-life numbers, events, and conditions
  - How to represent familiar numeric quantities?
    - number systems: binary, octal, and hexadecimal
  - How to represent nonnumeric data?

Positional Number Systems

- In a positional number system, a number is represented by a string of digits, where each digit position has an associated weight.
- The value of a number is a weighted sum of the digits. For a decimal system with **radix** *r* = 10:
  - $1734_{10} = 1 \cdot 1000 + 7 \cdot 100 + 3 \cdot 10 + 4 \cdot 1$
  - $5185.68_{10} = 5 \cdot 1000 + 1 \cdot 100 + 8 \cdot 10 + 5 \cdot 1 + 6 \cdot 0.1 + 8 \cdot 0.01$

- In a general positional number system, the radix may be any integer $r \geq 2$, and a digit in position *i* has weight r^i.
- The general form of a number *D* in such a system is

$$D = d_{p-1}d_{p-2}...d_1d_0.d_{-1}d_{-2}...d_{-n} = \sum_{i=-n}^{p-1} d_i * r^i$$

, where there are *p* digits to the left of the point and *n* digits to the right of the point, called the **radix point**.

Radices and Sets of Symbols

number system	radix	symbols
binary	2	0,1
octal	8	0, 1,…, 7
decimal	10	0, 1,…, 9
hexadecimal	16	0, 1,…, 9, A, B, C, D, E, F

Examples:

Decimal

$2007_{10} = 2*1000 + 0*100 + 0*10 + 7*1$

$19.85_{10} = 1*10 + 9*1 + 8*0.1 + 5*0.01$

$$D = \sum_{i=-n}^{p-1} d_i * 10^i$$

$$D = \sum_{i=-n}^{p-1} d_i * 2^i$$

Binary

$1100110_2 = 1*2^6 + 1*2^5 + 1*2^2 + 1*2^1 = 64 + 32 + 4 + 2 = 102_{10}$

$101.0011_2 = 1*2^2 + 1*2^0 + 1*2^{-3} + 1*2^{-4} = 4 + 1 + 0.125 + 0.0625 = \ldots_{10}$

Most significant bit

Least significant bit

Number Systems Examples

Examples:

Octal

$$D = \sum_{i=-n}^{p-1} d_i * 8^i$$

$3577_8 = 3*8^3 + 5*8^2 + 7*8^1 + 7*8^0 = 1919_{10}$

$35.77_8 = 3*8^1 + 5*8^0 + 7*8^{-1} + 7*8^{-2}$

Hexadecimal

$$D = \sum_{i=-n}^{p-1} d_i * 16^i$$

$2007_{16} = 2*16^3 + 7*16^0 = 8199_{10}$

$7D7_{16} = 7*16^2 + 13*16^1 + 7*16^0 = 2007_{10}$

$A.2C_{16} = 10*16^0 + 2*16^{-1} + 12*16^{-2} = 10 + 0.125 + 0.046875 = \ldots_{10}$

Conversion from any Positional Number System to Decimal

- Radix 10 is important because we use it in everyday life.
- Radix 2 is important because binary numbers can be processed directly by digital circuits.
- Numbers in other radices are not often processed directly but may be important for documentation or other purposes.
- In particular, the radices 8 and 16 provide convenient shorthand representations for multibit numbers in a digital system.
- The value of the number expressed in radix *r* can be found by converting each digit of the number to its radix-10 equivalent and expanding the formula (D= $\sum_{i=-n}^{p-1} d_i \times r^i$) using radix-10 arithmetic.

Binary, Decimal, Octal , and Hexadecimal Numbers

binary	decimal	octal	hexadecimal
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	8	10	8
1001	9	11	9
1010	10	12	A
1011	11	13	B
1100	12	14	C
1101	13	15	D
1110	14	16	E
1111	15	17	F

Conversion from Decimal to any Positional Number System: Integral Part

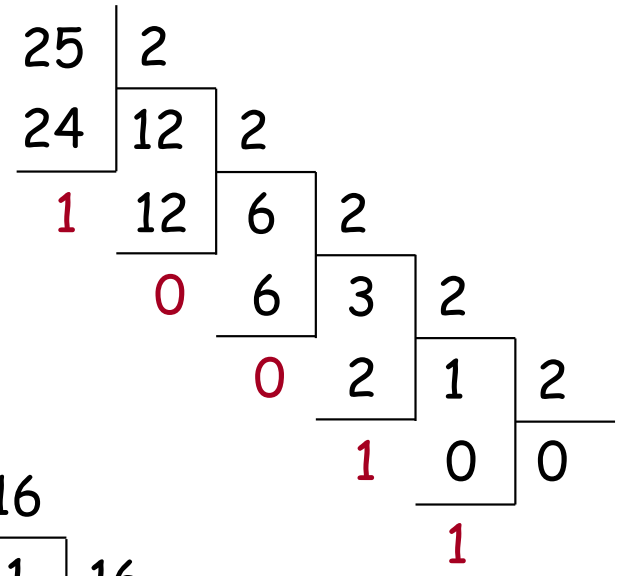
- The **integral part** of a number *D* expressed in decimal can be converted to any radix *r* ≥ 2 by **successful division of *D* by *r*** (using radix-10 arithmetic, until the result is 0) with **reverse recording** (in radix *r*) of all the obtained **remainders**.

Examples:

Conversion to binary

$25_{10} = ???_{2}$

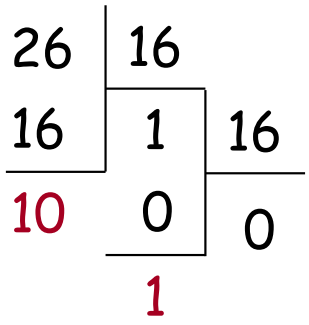
$25_{10} =$ $_{2}$



Conversion to hexadecimal

$26_{10} = ???_{16}$

$26_{10} =$ $_{16}$ A



Conversion from Decimal to any Positional Number System: Fractional Part

- The **fractional part** of a number *D* expressed in decimal can be converted to any radix $r \geq 2$ by **successful multiplication of *D* by *r*** (using radix-10 arithmetic, until the desired precision is reached) with **direct recording** (in radix *r*) of all the obtained **integral parts**.

Examples:

Conversion to binary

$0.6875_{10} = ???_{2}$

$0.6875_{10} = 0._{2}$

Conversion to hexadecimal

$0.25_{10} = ???_{16}$

$0.25_{10} = 0._{16}$

```
0.6875
     2
---------
0 .3750
     2
---------
0 .750
     2
---------
1 . 50
     2
---------
0 .00
```

```
0.25
  16
---------
4 .00
```

Rounding Error

- If the successive multiplication processes does not seem to be heading towards a final zero, the fractional number will have an infinite length. Try representing 0.33_{10} in binary.
- So the number of multiplication steps should be limited depending on the degree of accuracy required.
- In order to keep the accuracy of the original presentation, the following formula is applied:

$$n_2 = \lfloor n_1 * \log_{r_2} r_1 \rfloor$$

r_1 – initial radix
r_2 – final radix
n_1 – number of fractional digits in the original number, expressed in radix r_1
n_2 – number of fractional digits in the converted number, expressed in radix r_2

Examples:

$0.6875_{10} = ???_2$ $0.6875_{10} = 0.1011_2$ $0.6875_{10} = 0.1011000000000_2$

$A.2C_{16} = 10*16^0 + 2*16^{-1} + 12*16^{-2} = 10 + 0.125 + 0.046875 = 10.171875 = 10.17_{10}$

$101.0011_2 = 1*2^2 + 1*2^0 + 1*2^{-3} + 1*2^{-4} = 4 + 1 + 0.125 + 0.0625 = 5.1875 = 5.2_{10}$

Special Conversion Cases

- When a number is converted from radix r_1 to radix r_2 and $r_1 = r_2^x$, then each digit in radix r_1 can be **substituted directly** with x radix r_2 digits.
- The **octal** and **hexadecimal** number systems are useful for representing multibit numbers because their radices are powers of 2.
- Each **octal digit is represented with 3 binary digits** ($8 = 2^3$). Each **hexadecimal digit is represented with 4 binary digits** ($16 = 2^4$).

Examples:

Conversion from octal to binary

$r_1 = 8, r_2 = 2, 8 = 2^3$

$753.6_8 = 111\ 101\ 011\ .\ 110_2$

Conversion from hexadecimal to binary

$r_1 = 16, r_2 = 2, 16 = 2^4$

$A5.E_{16} = 1010\ 0101\ .\ 1110_2$

Special Conversion Cases (cont.)

- When a number is converted from radix r_1 to radix r_2 and $r_1^x = r_2$, then a sequence of x digits in radix r_1 can be **substituted directly** with one radix r_2 digit.
- To convert a **binary number to octal**, start at the binary point and work left, separating the bits into groups of three and replacing each group with the corresponding octal digit; then work right. Freely add zeroes on the left or right to make the total number of bits a multiple of 3.
- To convert a **binary number to hexadecimal**, start at the binary point and work left, separating the bits into groups of four and replacing each group with the corresponding hexadecimal digit; then work right. Freely add zeroes on the left or right to make the total number of bits a multiple of 4.

Examples:

Conversion from binary to octal $1\ 101\ .\ 010_2 = 15\ .\ 2_8$

$r_1 = 2, r_2 = 8, 2^3 = 8$

Conversion from binary to hexadecimal $110\ 0101\ 1100_2 = 65C_{16}$

$r_1 = 2, r_2 = 16, 2^4 = 16$

Summary of Conversion Methods

From	To	Method
Binary	Octal	Substitution (replace each group of 3 bits with the corresponding octal digit)
	Decimal	Summation (D= $\sum_{i=-n}^{p-1} d_i \times 2^i$)
	Hexadecimal	Substitution (replace each group of 4 bits with the corresponding hexadecimal digit)
Octal	Binary	Substitution (each octal digit is represented with 3 bits)
	Decimal	Summation (D= $\sum_{i=-n}^{p-1} d_i \times 8^i$)
	Hexadecimal	Convert to binary, then to hexadecimal
Decimal	Binary	Division by 2 for integral part, multiplication by 2 for fractional part
	Octal	Division by 8 for integral part, multiplication by 8 for fractional part
	Hexadecimal	Division by 16 for integral part, multiplication by 16 for fractional part
Hexadecimal	Binary	Substitution (each hexadecimal digit is represented with 4 bits)
	Octal	Convert to binary, then to octal
	Decimal	Summation (D= $\sum_{i=-n}^{p-1} d_i \times 16^i$)

Exercises

- Explain Moore's Law.
- What are the advantages of digital systems compared to analog systems?
- When an OR gate output is 0?
- When an AND gate output is 1?

Exercises (cont.)

- How to convert an integer octal number into hexadecimal?
- Consider that expression $1234_r < 1234_8$ is true. Is it possible to determine the value of *r*?
- Is it possible to convert the number 39_8 into decimal?
- What is the relationship (>, =, or <) between 34_8 and 34_{16}?
- Is the following affirmation true: $20=14_{16}$?

Exercises (cont.)

- Convert the following numbers into binary, octal, decimal, and hexadecimal:

10111011001_2 $= 2731_8 = 5D9_{16} = 1497_{10}$

1234_8 $= 001010011100_2 = 29C_{16} = 668_{10}$

$C0DE_{16}$ $= 1100000011011110_2 = 140336_8 = 49374_{10}$

108_{10} $= 1101100_2 = 154_8 = 6C_{16}$

15.46_{10} $= 1111.011101_2 = 17.35_8 = F.7_{16}$