School of Computing Sciences and Computer Engineering

Introduction to Computer Engineering

Evidence-Based Inclusive Teaching Practices

Zhaoxian Zhou*

November 12, 2024

* University of Southern Mississippi

Disclaimer

You can edit this page to suit your needs. For instance, here we have a no copyright statement, a colophon and some other information. This page is based on the corresponding page of Ken Arroyo Ohori's thesis, with minimal changes.

No copyright

Colophon

This document was typeset with the help of KOMA-Script and LaTeX using the kaobook class.
The source code of this book is available at:
`https://github.com/fmarotta/kaobook`
(You are welcome to contribute!)

Publisher

First printed in August 2023 by

In life sciences, we find a reasonable balance between men and women. In engineering and computer science, we have a major problem. A very small percentage of women will be in computer science.

– Freeman A. Hrabowski III.

Preface

1 Evidence-based instruction

Here are top 10 evidence based teaching strategies [1]

1. State Clear Learning Goals repeatedly, so students have a clear idea of where they are going and what it will look like when they get there. This is a practice that creates transparency in learning and teaching.
2. Share and Model concepts to explain and then demonstrate how students will do a task, whether a physical or thinking task. Sharing and modeling looks different in each discipline. For some, that may be “thinking out loud" to show students how experts process or it may be doing a physical demonstration.
3. Check for Student Understanding by asking for feedback from students in various ways, regularly. Research shows that this habit of asking for student feedback has more impact for learning than giving students feedback. Ask what students what they understand. and how the course and class sessions are structured helps them learn. Resources that you can import into Canvas are available for Clemson instructors. Search "clemson teaching" in the Commons.
4. Give Feedback to Students regularly - on work that is a low stakes grade - in a feedback loop. Think “homework", classwork and quizzes. Encourage students to evaluate their own assessment results (see below).
5. Record information in graphical ways by both instructor and students for visual learning and deeper processing. This area includes adding key, clear visuals to powerpoints (visuals are easier to remember), creating graphs, charts, and diagrams - and having students create their own.
6. Allow Repeat and Spaced Practice through assigned work out of class & work during class with opportunities for feedback to & from students. See number four above. Repeat practice spaced over time - such as having students recall information from earlier in the course - helps solidify their learning.
7. Create Opportunities for Peer-to-Peer Learning so that students assist each other in understanding concepts. The basis of excellent “group work" is work that is meaningful for students, in which they can all contribute to each others' learning. In large classes, this can be accomplished by “pair-share" questions they discuss with their immediate neighbors.
8. Build in Time to Succeed by allowing varying time per unit, in particular to account for learning difficult concepts. While difficult to accomplish “on the fly," instructors who have taught the content before can provide students more time on difficult concepts. Consider examining the “threshold concepts" in your content area.
9. Teach Strategies for Learning with general resources and techniques specific to a discipline. Encourage students to us resources from the Academic Success Center and the libraries and provide information on ways to learn in the particular content area that you are teaching. Students today often have gaps in their knowledge of study techniques, such as effective note-taking, approaches to time management, and test preparation.
10. Nurture Metacognition by prompting students to ask how they are thinking about a subject. This area is getting a lot of attention in higher education currently. Encouraging students to assess their own learning through activities and homework helps them take responsibility for their learning–and balances out the “teaching and learning" responsibilities.

2 Active learning

Active learning strategies in the college classroom refer to techniques that promote student engagement and active participation in the learning process. Rather than simply listening to lectures and taking notes, active learning strategies encourage students to interact with the material, think about what they are doing, ask questions, collaborate with peers, and apply their knowledge in real-world scenarios.

Examples of active learning strategies in the college classroom include:

- Group discussions and peer-to-peer learning
- Problem-based learning and case studies
- Interactive lectures and demonstrations
- Flipped classroom model, where students review materials before class and engage in discussions and activities in the classroom
- In-class exercises, such as debates, brainstorming sessions, and simulations
- Use of technology, such as online polls and quizzes, to encourage participation and engagement

The goal of active learning strategies is to create a dynamic learning environment that promotes critical thinking, problem-solving, and engagement. By encouraging students to actively participate in the learning process, active learning strategies can help improve learning outcomes and better prepare students for success in their future careers.

Active learning is a global trend, and many universities across the world are incorporating these methodologies into their teaching practices. The effectiveness of active learning often depends on the specific courses, instructors, and the overall institutional culture. Many universites, such MIT and standford Univeristy, support faculty development, pedagogical research, technology integration, and provide student support services.

3 Inclusive teaching

Inclusive teaching refers to the practice of creating a learning environment that values and respects the diversity of all students. It is a pedagogical approach that recognizes and addresses the unique backgrounds, experiences, and perspectives of students, with the goal of ensuring that all students feel supported, engaged, and empowered to learn.

Inclusive teaching strategies can include a variety of practices, such as:

- Creating a welcoming and respectful classroom environment that values diversity and encourages open discussion and active participation from all students.
- Using a variety of teaching methods and materials that reflect and respect the diversity of the student body.
- Providing multiple means of representation, expression, and engagement to accommodate different learning styles and abilities.
- Incorporating inclusive language and images in course materials, assignments, and discussions.
- Being aware of and addressing potential biases and stereotypes that may affect student learning and engagement.
- Encouraging and promoting collaborative and cooperative learning among students of different backgrounds and experiences.
- Providing opportunities for feedback and reflection to continually improve teaching practices and support student learning.

Overall, inclusive teaching seeks to create a learning environment where all students feel seen, heard, and valued, and where diversity is celebrated as a strength.

There are some resources that help inclusive teaching:

- The Association of College and University Educators (ACUE) [2]
- American Society for Engineering Education (ASEE) [3]

Although Computing Engineering might not be assumed to be inclusive traditionally, the proposed project tries to change it. It is expected that all students in CE 101 will be impacted by either improvement in their learning outcomes directly, or awareness of the necessity of diversity, equity, and inclusion, or both. It is also expected that underrepresented minority, first-generation, and woman students will benefit the most. The project aligns well with the School's mission and USM's value to support "an inclusive community that embraces the diversity of people and ideas".

Zhaoxian Zhou

Contents

List of Figures

List of Tables

1 Overview of Computer Engineering

1.1 What is Computer Engineering?

Computer engineering is an interdisciplinary field of study that combines principles from computer science and electrical engineering to design, develop, and test computer systems. It encompasses a wide range of technologies and applications, from microprocessors and embedded systems to computer networks and information systems. Computer engineering is a vital field, as it provides the foundation for the development of modern computing systems, including personal computers, smartphones, and the internet.

The field of computer engineering focuses on the design and analysis of digital systems, including microprocessors, memory systems, and communication networks. This involves understanding the underlying principles of computer hardware and software, as well as the algorithms and data structures that underlie modern computing systems. Computer engineers must also be familiar with the various programming languages and software tools used to develop and test computer systems.

1

One of the primary areas of focus in computer engineering is the development of new computer hardware. This includes designing and testing microprocessors, memory systems, and other components that form the building blocks of modern computers. Computer engineers may also work on the development of new types of computer hardware, such as wearable devices or specialized sensors.

Another area of focus in computer engineering is the design of computer networks. This involves developing protocols for transmitting data between different devices, as well as designing the hardware and software that

[1] List some computer engineering topics you know before enrolling in this class.

enable these networks to function. Computer engineers may work on the development of both local area networks (LANs) and wide area networks (WANs), which are used to connect computers and other devices over large distances.

In addition to hardware and network design, computer engineering also involves the development of software applications. Computer engineers may work on developing software that runs on a single device, such as a personal computer or smartphone, or they may work on the development of complex software systems that run on multiple devices and platforms. This requires a deep understanding of programming languages, software tools, and the various platforms on which software is developed and deployed.

Another important area of focus in computer engineering is user interface design. This involves designing the visual and interactive elements of software applications and other computer systems, including graphical user interfaces (GUIs), touchscreens, and other input devices. User interface design is critical to the success of computer systems, as it can greatly impact the user experience and usability of a system.

In addition to these specific areas of focus, computer engineering also involves a wide range of general skills and knowledge. Computer engineers must be able to work in a team environment, as they often collaborate with other engineers and professionals from different disciplines. They must also be able to communicate effectively with non-technical stakeholders, such as managers and clients. Computer engineers must also stay up-to-date with the latest technologies and trends in the field, as computer engineering is a rapidly evolving discipline.

[2]

Overall, computer engineering is a vital field that plays a critical role in the development of modern computing systems. It combines principles from computer science and electrical engineering to design, develop, and test computer systems, including hardware and software components. Computer engineers work in a wide range of industries, including technology, healthcare, finance, and entertainment, and play an essential role in driving innovation and progress in these industries. As the field of computing continues to evolve and expand, the need for skilled and knowledgeable computer engineers will only continue to grow.

1.2 History of Computer Engineering

The history of computer engineering can be traced back to the early days of computing, when the first digital computers were developed. During the 1940s and 1950s, computer systems were large, expensive, and highly specialized machines that were primarily used for scientific and military applications. These early computers were built using vacuum tubes and other primitive electronic components, and were programmed using punched cards and other mechanical devices.

[2] List some computer engineering topics you know after enrolling in this class.

In the 1960s and 1970s, the development of the integrated circuit (IC) revolutionized the field of computer engineering. The IC allowed for the miniaturization of electronic components, making it possible to build smaller and more powerful computer systems. The first commercially available ICs were developed by companies such as Texas Instruments and Fairchild Semiconductor, and were used in a wide range of electronic devices, including calculators, digital watches, and early personal computers.

During the 1980s and 1990s, computer engineering underwent a period of rapid growth and innovation. The development of the microprocessor, a single-chip CPU that integrated all of the functions of a computer's central processing unit onto a single chip, was a major milestone in the history of computer engineering. The microprocessor made it possible to build powerful and versatile computer systems that were affordable and widely available to consumers.

The rise of personal computers during the 1980s and 1990s also had a significant impact on the field of computer engineering. Companies such as IBM, Apple, and Microsoft developed new software and hardware technologies that made it possible for ordinary people to use computers for a wide range of applications, including word processing, gaming, and communication.

During the 2000s and 2010s, computer engineering continued to evolve and expand. The development of the internet and other network technologies enabled the creation of vast digital networks that connected people and computers around the world. The rise of mobile devices such as smartphones and tablets also had a significant impact on the field of computer engineering, as these devices required new hardware and software technologies to enable their small size, portability, and advanced features.

Today, computer engineering is a rapidly evolving field that continues to drive innovation and progress in a wide range of industries, including technology, healthcare, finance, and entertainment. Computer engineers work on a variety of projects, including developing new computer hardware, designing computer networks, creating software applications, and designing user interfaces. As the field of computing continues to evolve and expand, the need for skilled and knowledgeable computer engineers will only continue to grow.

1.3 Computer Science, Computer Engineering, and Electrical Engineering

[3]

Computer science, computer engineering, and electrical engineering are three closely related but distinct fields in the realm of technology. Some key similarities and differences between these fields are discussed below.

[4]

[3] Obtain a copy of the CEBS curriculum before hand. You can find it online, or request one from the office or the instructor.

[4] Examine the courses in the CEBS curriculum. Identify the courses in CS, EE, and CE, respectively.

Similarities:

- All three fields are involved in the design, development, and implementation of computer-based systems.
- They all involve the use of mathematics and logic to solve problems and create efficient solutions.
- They require an understanding of fundamental principles of computer hardware and software.
- They are all interdisciplinary fields, with some overlap in coursework and research areas.

Differences:

- Computer science focuses primarily on software and algorithms, including programming languages, data structures, algorithms, and computational theory. It involves the study of computer systems and their applications, but does not typically involve the design of hardware components.
- Computer engineering focuses on the design and development of computer hardware and systems, including microprocessors, circuit design, and embedded systems. It combines principles of computer science and electrical engineering.
- Electrical engineering involves the study and application of electricity, electronics, and electromagnetism. It encompasses a broad range of subfields, including power systems, signal processing, telecommunications, and control systems. It is not necessarily focused solely on computers, but can involve the design and implementation of electronic systems.

[5]

In summary, computer science, computer engineering, and electrical engineering share some similarities in their focus on technology and problem-solving, but differ in their specific areas of study and applications. Computer science focuses on software and algorithms, computer engineering focuses on hardware and systems, and electrical engineering has a broader range of applications in electronics and electromagnetism.

1.4 Fields and Subfields in Computer Engineering

Computer engineering is a field of study that involves the design, development, and maintenance of computer systems and their components. It is a broad field that encompasses a wide range of subfields, each of which focuses on specific aspects of computer engineering.

One of the primary subfields of computer engineering is computer hardware engineering. This subfield involves the design and development of computer hardware, such as central processing units (CPUs), memory devices, and input/output (I/O) devices. Computer hardware engineers also work on the design and development of computer systems, including desktop and laptop computers, servers, and supercomputers.

[5] List other courses you would like to see in our CEBS curriculum.

Another subfield of computer engineering is computer software engineering. This subfield focuses on the design, development, and maintenance of software applications and systems. Computer software engineers are responsible for creating programs and applications that run on computer systems, as well as developing software tools and platforms that other engineers can use to develop new applications.

Computer network engineering is another important subfield of computer engineering. This subfield focuses on the design and development of computer networks, including local area networks (LANs), wide area networks (WANs), and the internet. Computer network engineers are responsible for designing and implementing network architectures, protocols, and security measures to ensure that computer systems can communicate and share data securely and efficiently.

Other subfields of computer engineering include computer graphics and multimedia engineering, which involves the design and development of graphics and multimedia software and systems, and computer architecture engineering, which focuses on the design and development of computer systems and architectures.

In addition to these subfields, computer engineering also intersects with other fields, such as electrical engineering, software engineering, and computer science. As such, computer engineering is a multidisciplinary field that draws on a variety of different disciplines and technologies to create innovative and effective computer systems and applications.

Overall, computer engineering is a rapidly growing and evolving field that plays a crucial role in the development of modern technology. As technology continues to advance, the need for skilled computer engineers who can design, develop, and maintain computer systems will only continue to grow.

1.5 The Role of Computer Engineers in Society

Computer engineers play a crucial role in society today, as they are responsible for designing and developing the technology that powers many of the systems and applications we rely on every day. From smartphones and laptops to complex computer networks and supercomputers, computer engineers are the driving force behind much of the modern technology that enables us to work, communicate, and connect with one another.

One of the primary roles of computer engineers in society is to develop and maintain computer hardware and software systems. This includes designing and building computer systems and components, as well as creating and testing software applications and tools. Computer engineers also play a crucial role in ensuring that computer systems are secure and reliable, working to develop new security measures and protocols to protect against cyber threats and other forms of attack.

Computer engineers are also responsible for developing and implementing computer networks and communication systems. This includes designing and building the hardware and software systems that enable computers to communicate and share data with one another, as well as developing the protocols and standards that govern how information is transmitted and received over these networks.

In addition to these technical responsibilities, computer engineers also play an important role in society by using their expertise to help solve real-world problems. For example, computer engineers might work on developing new technologies and applications that can help address environmental challenges, such as improving energy efficiency or reducing waste. They might also work on developing new medical technologies, such as wearable devices or diagnostic tools, that can help improve health outcomes for patients.

Computer engineers also play an important role in education and research, working to develop new theories and methodologies that can help advance the field of computer engineering. This includes conducting research into new hardware and software technologies, as well as exploring new applications and use cases for existing technologies.

Overall, the role of computer engineers in society is multifaceted and complex. From designing and building computer systems and applications to developing new technologies and conducting research, computer engineers are responsible for driving much of the technological innovation that powers modern society. As such, the work of computer engineers is essential to the continued development and progress of our society, and will only become more important as technology continues to advance and evolve.

1.6 Ethical Issues

Ethics in computer engineering refers to the moral principles and values that guide the development and use of computer systems and technology. Here are some of the key ethical issues in computer engineering:

- Privacy and data security: As technology advances, collecting, storing, and analyzing data has become easier than ever. However, this has also led to concerns about data privacy and security. Computer engineers have a responsibility to design and implement systems that protect users' data and maintain their privacy.
- Bias and discrimination: Computer systems and algorithms can be influenced by the biases of their developers, resulting in discrimination against certain groups of people. Computer engineers should strive to create systems that are fair and unbiased, and take steps to identify and address biases in their work.
- Intellectual property: Intellectual property rights are an important consideration in computer engineering. Developers must respect the intellectual property rights of others and ensure that their own work is properly protected.

- Accessibility: Computer systems and technology should be accessible to everyone, regardless of their physical abilities. Computer engineers should design systems that are accessible to people with disabilities and ensure that their work complies with accessibility standards.
- Social responsibility: Computer engineers have a responsibility to consider the impact of their work on society as a whole. This includes designing systems that do not contribute to harmful practices, such as environmental damage or social inequality.

In summary, ethics in computer engineering involves considering the moral implications of one's work and making decisions that prioritize the well-being of users, society, and the environment. Computer engineers have a responsibility to create systems that are secure, unbiased, accessible, and socially responsible. By prioritizing ethics in their work, computer engineers can help ensure that technology is used for the greater good.

1.7 Diversity and Inclusion

Diversity and inclusion are crucial for creating a vibrant and innovative field in computer engineering. Here are some ways in which diversity and inclusion can be promoted in computer engineering:

- Increase diversity in computer engineering programs: Encouraging a diverse range of students to study computer engineering can help ensure that a variety of perspectives and ideas are brought to the field. This can be achieved through targeted recruitment efforts, scholarships and other financial incentives, and outreach to underrepresented communities.
  [6]
- Foster an inclusive culture: Creating a welcoming and supportive culture can help attract and retain a diverse range of individuals in computer engineering. This can involve providing mentorship and networking opportunities, promoting work-life balance, and addressing issues of bias and discrimination in the workplace.
  [7]
- Develop diverse teams: Creating teams with diverse backgrounds and experiences can lead to more innovative and creative solutions. This can be achieved through inclusive hiring practices, such as expanding recruitment to a wider range of sources, and providing training on unconscious bias to hiring managers.
  [8]
- Support professional development: Providing resources and opportunities for professional development can help ensure that individuals from underrepresented groups have the support they need to succeed in computer engineering. This can include access to training and mentoring programs, opportunities for networking and career advancement, and resources for continuing education.
- Promote diversity in leadership: Encouraging and supporting individuals from underrepresented groups to take on leadership roles in

[6] Discuss the diversity and inclusion of CEBS programs. Discuss ideas to increase diversity in our CEBS.

[7] Compared with other fields, how diverse and inclusive is computer engineering?

[8] Which companies are more diverse and inclusive?

computer engineering can help create a more inclusive and diverse field. This can involve providing leadership training and development opportunities, creating pathways for advancement, and actively seeking out and promoting diverse candidates for leadership positions.

[9]

Overall, promoting diversity and inclusion in computer engineering requires a concerted effort from individuals, organizations, and institutions. By fostering an inclusive culture and supporting a diverse range of individuals, the field can become more innovative, responsive, and effective in meeting the needs of society.

[9] Discuss any figures who promoted diversity and inclusion in engineering?

2 Binary Systems and Number-base Conversions

Some fundamental concepts in computer engineering, including bits, data types, operations, logic operations and elements,

2.1 Number Systems

1. Decimal : The decimal number system is a positional number system that uses ten digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) to represent numbers. Each digit has a place value that depends on its position in the number. The rightmost digit represents the units place, the next digit to the left represents the tens place, the next digit to the left represents the hundreds place, and so on.

   Example 2.1.1 The number 1234 in the decimal system represents

   $$1234.56 = 1\times10^3+2\times10^2+3\times10^1+4\times10^0+5\times10^{-1}+6\times10^{-2}.$$

   The decimal system is also known as the base-10 system because it uses ten digits. It is the most commonly used number system in the world and is used in everyday life for counting, arithmetic, and measuring quantities.
   To specify a number clearly in a base, we write the base as a subscript. For example: 1234.56_{10}.
   [1]

2. Binary : The binary system is a positional number system that uses only two digits, 0 and 1, to represent numbers. Each digit in the binary system has a place value that depends on its position in the number, just like in the decimal system.
   An example of binary number is 101101.101_2

[1] Write your student ID here as a decimal number N.

The binary system is used extensively in digital electronics and computing because it is easy to represent in electronic devices. In computing, binary digits are called bits, and groups of 8 bits are called bytes. All data in a computer is stored and processed in binary form.
[2]

3. Octal : The octal system is a positional number system that uses eight digits (0, 1, 2, 3, 4, 5, 6, and 7) to represent numbers. Each digit in the octal system has a place value that depends on its position in the number, just like in the decimal and binary systems.
   An example of octal number is 726_8.
   The octal system is sometimes used in computing, particularly in older computer systems. It was more popular in the past because it was easier to work with octal numbers when using hardware that had an 8-bit word length. However, with the advent of more advanced computing technology, the octal system is less commonly used today.
   [3]
4. Hexadecimal : The hexadecimal system is a positional number system that uses 16 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F) to represent numbers. Each digit in the hexadecimal system has a place value that depends on its position in the number, just like in the decimal, binary, and octal systems.
   An example of hexadecimal number is $AB2_{16}$.
   In the hexadecimal system, the letters A through F are used to represent the values 10 through 15, respectively. This is done because it is easier to represent and read hexadecimal numbers when they are written with fewer digits.
   The hexadecimal system is commonly used in computing because it is easy to convert between hexadecimal and binary numbers. Two hexadecimal digits can represent exactly one byte (8 bits) of data, making it easy to represent and manipulate binary data in a more compact form.
   [4]

2.2 Number Conversion

Numbers with different bases can be converted to each other. Here we discuss number conversions between decimal, binary, octal, and hexadecimal.

a) Binary, octal, and hexadecimal numbers converted to decimal
   There are two methods: positional notation method and doubling method.
   Positional notation method: the value of a digit in a number is determined by a weight based on its position. As in the decimal numbers, each digit in binary numbers has its bit values:

$$\cdots, 2^3, 2^2, 2^1, 2^0, 2^{-1}, 2^{-2}, \cdots$$

[2] Convert the decimal number N to binary number B here.

[3] Convert the decimal number N to Octal format O here.

[4] Convert the decimal number N to hexadecimal H here.

Similarly for octal numbers, the bit values are

$$\cdots, 8^3, 8^2, 8^1, 8^0, 8^{-1}, 8^{-2}, \cdots$$

For hexadecimal numbers, the bit values are

$$\cdots, 16^3, 16^2, 16^1, 16^0, 16^{-1}, 16^{-2}, \cdots$$

Doubling method : for binary numbers converted to decimal, start from the left-most digit of the binary number, double the previous number and add the current digit. Repeat until reaching the last digits.

Example 2.2.1 Positional notation method
The binary number 101101.101 converted to decimal:

$$101101.101_2 = 1\times2^5+0\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$

$$= 32 + 0 + 8 + 4 + 0 + 1 + 0.5 + 0 + 0.125 = 45.625_{10}$$

Example 2.2.2 Doubling method
The binary number 101 converted to decimal:

$$101_2 = (1 \times 2 + 0) \times 2 + 1 = 5_{10}$$

Example 2.2.3 Positional notation method
Octal numbers converted to decimal:

$$726_8 = 7 \times 8^2 + 2 \times 8^1 + 6 \times 8^0 = 448 + 16 + 6 = 470_{10}$$

Example 2.2.4 “Doubling" method (It no longer can be called doubling, but it is similar to that for binary numbers.)
Octal numbers converted to decimal:

$$726_8 = (7 \times 8 + 2) \times 8 + 6 = 470_{10}$$

Example 2.2.5 Positional notation method
Hexadecimal numbers converted to decimal:

$$AB2_{16} = 10\times16^2+11\times16^1+2\times16^0 = 2560+176+2 = 2738_{10}$$

Example 2.2.6 “Doubling" method
Hexadecimal numbers converted to decimal:

$$AB2_{16} = (10 \times 16 + 11) \times 16 + 2 = 2738_{10}$$

b) Decimal numbers converted to binary, octal, and hexadecimal numbers
To convert a decimal to binary, we simply divide the number by 2 recursively until we get to 0 and note down any remainder. All remainders in backward is the converted binary number.

Example 2.2.7 Decimal number converted to binary

divisor	*quotient*	*remainder*	*binary*
2	$\boxed{125}$		
2	62	1	↑
2	31	0	↑
2	15	1	↑
2	7	1	↑
2	3	1	↑
2	1	1	↑
	0	1	↑

The decimal number 125 to be converted is boxed for clarification only. All remainders in backward (in the direction of the arrows) is the converted binary number:

$$125_{10} = 1111101_2$$

Example 2.2.8 Decimal number converted to octal

divisor	*quotient*	*remainder*	*octal*
	$\boxed{470}$		
8	58	6	↑
8	7	2	↑
	0	7	↑

Write all remainders in backward (in the direction of the arrows) we get the converted octal number:

$$470_{10} = 726_8$$

Example 2.2.9 Decimal number converted to hexadecimal

divisor	*quotient*	*remainder*	*Hexadecimal*
16	$\boxed{2738}$		
16	171	2	↑
16	10	11	↑
	0	10	↑

Write all remainders in backward (in the direction of the arrows) we get the converted octal number:

$$2738_{10} = AB2_{16}$$

c) Conversions between binary, octal, and hexadecimal: we can convert one format into decimal and then to another format. However, there is an easier method through grouping .

Example 2.2.10 Binary to octal: break the binary number into groups of three digits, from right to left, then convert the binary number in each group into octal.

$$10001111111000010000_2 \xrightarrow[\text{of three}]{\text{groups}} 10, 001, 111, 111, 000, 010, 000_2$$

$$= 2177020_8$$

Example 2.2.11 Octal to binary: convert each octal digit into three binary digits ((except possibly the first octal digit).

$$2177020_8 \xrightarrow[\text{of three}]{\text{groups}} 10, 001, 111, 111, 000, 010, 000_2$$

[5]

Example 2.2.12 Binary to hexadecimal: break the binary number into groups of four, from right to left, then convert the binary number in each group into Hex. Hexadecimal to binary: each hexadecimal digit converts to four binary digits (except possibly the first hexadecimal digit).

$$10001111111000010000_2 \xrightarrow[\text{of four}]{\text{groups}} 1000, 1111, 1110, 0001, 0000_2$$

$$= 8FE10_{16}$$

[6]

d) Discussions: How to convert between octal and hexadecimal

To Do

1. Form a pair with one of your classmates.
2. Discuss your solutions.
3. Make sure all team members are able to perform number-base conversion.

To read large numbers easily, they can be written in groups of three or four digits, such as

$$101, 101, 101, \text{ or } 1, 0110, 1101$$

[5] Convert your binary number B to octal one here. Is it the same as your previous O? If not, double check.

[6] Convert the binary number B to hexadecimal here. Is it the same as the previous H? If not, ask for help.

[7] Write steps to convert octal to hexadecimal number.

2.3 Complements of Numbers

Complements of numbers are used in digital computers to simplify the subtraction operation and logical manipulation.

Radix complement is a way of representing the complement of a number with respect to a certain base or radix.

Radix complement also called r's complement. In a number system with a base or radix r, the radix complement of a non-negative integer N is defined as $r^n - N$, where n is the number of digits in the representation of N in the given radix system.

Diminished radix complement also called $(r-1)'s$ complement. The diminished radix complement is the complement of a number with respect to the base or radix minus one. The diminished radix complement of a number N in base b is equal to $r^n - N - 1$, where n is the number of digits in N.

2.3.1 Complements of Decimal Numbers

For decimal system, $r = 10$. The $9's$ complement of a $n-$digit decimal number N also has the n digits, with each origianl digit subtracted from 9 as the digit.

Example 2.3.1 The diminished radix $(9's)$ complement of the number 346 in base 10 is

$$10^3 - 1 - 346 = 999 - 346 = 653.$$

[8]

The $10's$ complement of a $n-$digit decimal number N is the $9's$ complement plus 1.

Example 2.3.2 The radix $(10's)$ complement of the number 346 in base 10 is

$$10^3 - 346 = 1000 - 346 = 654, \text{ which is } 653 + 1.$$

[9]

2.3.2 Complements of Binary Numbers

In computer science and computer engineering, we use binary system, where radix $r = 2$. The 1's complement and 2's complement are important because they provide a way to represent signed numbers in binary form that is both efficient and widely used in digital systems and computer hardware.

[10]

[8] Let decimal number N be your student ID, without leading zeros. Find the 9's complement of N.

[9] Let decimal number N be your student ID, without leading zeros. Find the 10's complement of N.

[10] Group discussion: how to find the 1's and 2's complements of a binary number?

The 1's and 2's complements simplify arithmetic operations in binary. For example, in the 1's complement system, addition and subtraction can be performed using the same circuits that are used for unsigned binary numbers. Similarly, in the 2's complement system, subtraction can be performed by simply taking the 2's complement of the subtrahend and adding it to the minuend.

In addition to simplifying arithmetic operations, both complements also have other advantages. For example, they can represent a wider range of numbers than sign-magnitude representation. They also allow for the representation of negative zero, which can be useful in some applications.

Example 2.3.3 The 1's complement of binary number $101,101_2$ is $010,010_2$

We compute 1's complement of a binary number by changing its 0s into 1s and changing its 1s into 0s.

Example 2.3.4 The 2's complement of binary number $101,101_2$ is $010,011_2$.

We compute 2's complement of a binary number by adding 1 to the last digit of its 1's complement.

Exercise 2.3.1 Find the 1's and 2's complements of your binary number B. Let your peer check your answer. If either of you get a wrong answer, try another one.

The concept of complement of numbers is important. It enables efficiency in hardware design by allowing addition and subtraction to be handled by the same circuit

2.4 Unsigned and Signed Binary Numbers

Variables such as integers can be represented in two ways, i.e., signed and unsigned.

2.4.1 Unsigned binary numbers

Unsigned binary numbers are a way of representing only non-negative integers using the binary number system. Unsigned binary numbers use a fixed number of bits to represent each integer, and each bit has a weight equal to a power of 2 (i.e., 1, 2, 4, 8, 16, etc.). The value of a binary number is found by adding up the weights of the bits that are set to 1.

Example 2.4.1 The binary number 1011 represents the unsigned integer 11, since $1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 = 11$.

2.4.2 Signed binary numbers

Signed binary numbers, on the other hand, are a way of representing both positive and negative integers using the binary number system. In signed binary, the leftmost bit, also called the most significant digit (MSD) , is used as a sign bit, with a value of 0 indicating a positive number and a value of 1 indicating a negative number.

There are three ways of representing signed numbers in a computer.

Sign and magnitude The sign-magnitude (SM) binary format is the simplest conceptual format.

> **Example 2.4.2** The binary number 0110 represents the positive integer 6, while the binary number 1110 represents the negative integer -6.

Note that both numbers have the same rightmost three bits but differ in their leftmost sign bit.

> **Exercise 2.4.1** Group Discussion: for a 8-bit integer variable N, what is its value range for unsigned and signed (SM) format, respectively?

If the word size is n bits, the range of numbers that can be represented is from $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$.
Table 2.1 shows the range of SM numbers that can be represented by a word size n.

Table 2.1: Ranges for Signed Magnitude Numbers

n	Range
4	-7 to $+7$
8	-127 to $+127$
16	-32767 to $+32767$
32	-2147483647 to $+2147483647$

Did you do the exercise above correctly? If not, check what happened.

One's complement One's complements is one of the methods of representing signed integers in the computer.

1. If the MSD is a 0, then the other bits indicate the magnitude (absolute value) of the number, and it is evaluated just as we would interpret any normal unsigned integer.
2. If the MSD is a 1, the number is negative, and then the other bits signify the 1's complement of the magnitude of the number.

For example,

1. a positive decimal number 5 is represented by 0101; because it is positive, the MSD is 0.
2. A negative decimal number -5 is represented by 1010, where MSD 1 represents negative, and the remaining bits 010 is the 1's complement of 101. We can also see that the whole number 1010 is the 1's complement of 0101.

Table 2.2 shows some signed decimal numbers and their equivalent in 1's complement notations, assuming a word size of n=4 bits.

Table 2.2: Signed Decimal Numbers and their 1's Complement Representation

Signed Decimal	1's Complement
$+6$	0110
-6	1001
$+0$	0000
-0	1111
$+7$	0111
-7	1000

> **Exercise 2.4.2** Group Discussion: review Table 2.1. Make a similar Table 2.3 showing the range of signed decimal numbers (size n) and their 1's complement representation. It should prove that if the

word size is n bits, the range of numbers that can be represented is from $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$.

Two's complement Two's complement is a mathematical operation used in digital circuits and computer systems to represent negative integers using binary numbers. In this system, the most significant bit of a binary number represents the sign of the number, where 0 indicates a positive number and 1 indicates a negative number.

Table 2.3: Ranges for Signed 1's Complement Numbers

n	Range
4	
8	
16	
32	

1. To convert a positive integer to its two's complement representation, we can simply represent the number in binary format, leaving the MSB as 0. For example, the number 7 in binary is 0111.
2. To convert a negative integer to its two's complement representation, we first need to represent the absolute value of the number in binary format. Then, we invert all the bits (0s become 1s and vice versa) and add 1 to the result. The resulting binary number is the two's complement representation of the negative integer. For example, to represent the number -7 in two's complement, we would start with the binary representation of 7 (0111), invert all the bits to get 1000, and then add 1 to get 1001.

Table 2.5 shows some signed decimal numbers and their equivalent in 2's complement notations, assuming a word size of n=4 bits.
For numbers with 8 bits, the range is from -128 to 127.

Table 2.4: Signed Decimal Numbers and their 2's Complement Representation

Signed Decimal	2's Complement
$+6$	0110
-6	1010
$+0$	0000
-0	0000
$+7$	0111
-7	1001

Exercise 2.4.3 Group Discussion: review Table 2.1. Make a similar Table 2.5 showing the range of signed decimal numbers (size n) and their 2's complement representation. It should prove that if the word size is n bits, the range of numbers that can be represented is from $-(2^{n-1})$ to $+(2^{n-1}-1)$. For example, for $n = 6$, the range should be from 100,000 to 011,111, which is from -32 to +31 in decimal.

2.4.3 How does a computer do arithmetic addition?

Table 2.5: Ranges for Signed 2's Complement Numbers

n	Range
4	
8	
16	
32	

First, we recall the situation for decimal numbers.

In signed-magnitude system, addition of two numbers follows ordinary arithmetic rules:

1. If the two numbers have same sign: add and then take the common sign.

Example 2.4.3

$$(-5) + (-6) = -11$$

Here we take the common sign - and use 6+5 as the magnitude.

2. If the two numbers have different signs, subtract the magnitude and then take the sign of the larger magnitude.

Example 2.4.4

$$(-5) + (+6) = +1$$

Here we take the sign + and use 6-5 as the magnitude.

Next, we consider for binary numbers. In binary system, it is easier: no comparison and subtraction are needed, but negative numbers need to be in signed 2's complement form.

The addition of two signed binary numbers (with negative numbers represented in signed 2's complement form) is just addition of the two numbers, including their sign bits. A carry out of the sign-bit position is discarded.

Example 2.4.5 Decimal 6+13=19.

In binary, it is then 000,110+001,101=010,011, which is +19.

Here we use 6 binary digits to represent signed numbers 6 and 13, with the MSD representing signs. If we use less, overflow will occur in the results. We can use more digits, but it will be less efficient and unnecessary.

[11]

Example 2.4.6 Decimal (-6)+13=7.

The 2's complement of -6 is 111,010, so

$$111,010 + 001,101 = 1,000,111.$$

Discarding the leading 1 (carry) gives 000,111, which is 7 in decimal.

Example 2.4.7 Decimal: 6+(-13)=-7.

The 2's complement of -13 is 110,011, so

$$000,110 + 110,011 = 111,001.$$

Because the MSD is 1, it is a negative number, and it is the 2's complement of 000,111 (please verify this).

We know 000,111 in binary is 7 in decimal and it is a negative number, therefore, the result is -7 in decimal format.

Example 2.4.8 Decimal: (-6)+(-13)=-19.

The 2's complement is -6 is 111,010, the 2's complement of -13 is 110,011, so

$$111,010 + 110,011 = \cancel{1},101,101.$$

Discarding the leading carry, we have 101,101.

[11] Can you figure out why we need to use 6 binary digits to represent signed numbers 6 and 13?

Because the MSD is 1, this is a negative number. It is the 2's complement of 010,011 (please verify this).

We know 010,011 in binary is 19 in decimal and it is a negative number, therefore, the result is -19 in decimal format.

2.4.4 How does a computer do arithmetic subtraction?

Computers can only to addition, not subtraction. Subtraction is carried out with addition of complement of subtrahend.

The subtraction of two signed binary numbers (with negative numbers represented in signed 2's complement form) is as follows:

1. Take the 2's complement of the subtrahend (including the sign bit).
2. Add it to the minuend (including the sign bit).
3. A carry out of the sign-bit position is discarded.
4. Keep in mind that if the leading digit is 1, it represents signed 2's complement, not the result itself.

Example 2.4.9 In decimal 50-37=13.

In binary,

$$50_{10} = 0,110,010_2; 37_{10} = 0,100,101_2$$

-37 in 2's complement form is $1,011,011_2$, so

$$50 - 37 = 0,110,010_2 + 1,011,011_2 = 10,001,101_2$$

Discarding the first bit gives the number 0,001,101. This is a positive result, and

$$001,101_2 = 13_{10}$$

Example 2.4.10 In decimal 37-50=-13.

In binary,

$$37_{10} = 0,100,101_2; 50_{10} = 0,110,010_2$$

-50 in 2's complement form is $1,001,110_2$, so

$$37 - 50 = 0,100,101_2 + 1,001,110_2 = 1,110,011_2$$

The sign digit is 1, so it is a negative number. It is the 2's complement of 0,001,101 (try to verify this). 0,001,101 is 13 in decimal. Therefore, the result is 13 with negative sign affix at the front. That means

$$37 - 50 = -13.$$

To Do

1. Form a pair with one of your classmates.
2. Obtain the two decimal IDs.

3. Each of you subtract your partner's ID from your ID.
4. Cross examine your results to make sure both are correct.
5. Your team together add up your results. Discuss your findings.

12

[12] Subtract your partner's ID from your ID in decimal.

To Do

1. Form a pair with another of your classmates.
2. Obtain the two binary IDs.
3. Each of you subtract your partner's ID from your ID.
4. Examine your partner's result to make sure it is correct.
5. Your team together add up your results. Discuss your findings.

13

[13] Subtract your partner's ID from your ID in binary.

Class Discussion

What are the differences between signed and unsigned binary numbers?

Subtraction is needed in computing the complements of decimal numbers. However this is different in computers where binary system is adopted. The 1's complement is done using NOT gates that invert each bit. The 2's complement is done by combining NOT gates for inversion and adder circuits to add 1 to the result.

2.5 Binary Codes

Binary code is a system of representing text, computer processor instructions, or any other data using a two-symbol system, for example, binary symbols 0 and 1. In binary code, each character or piece of data is represented using a combination of these two symbols. [14]

Computers and digital devices use binary code as their fundamental language because electronic circuits can easily distinguish between two states, such as high voltage, representing 1, and low voltage, representing 0. This binary representation forms the foundation of all digital information processing.

2.5.1 BCD code

BCD code stands for Binary Coded Decimal. It is a binary encoding scheme used to represent decimal numbers in a digital system. In BCD, each decimal digit is represented by a 4-bit binary number. For example, the decimal number 37 would be represented in BCD as 0011 0111. The first four bits represent the number 3, and the second four bits represent the number 7.

[14] We used two concepts here: digit and bit. Can you tell the difference between them?

Note: each decimal digit, not each decimal number, is represented by a BCD code.

The BCD codes for each decimal digit are listed in Table 2.6.

Table 2.6: BCD Codes for Decimal Digits

Decimal digit	BCD code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BCD is often used in digital systems where decimal arithmetic is required, such as in financial applications, in calculators, or in digital clocks. One advantage of BCD is that it is easy to convert to and from decimal, as each digit is represented by a separate set of four bits.

However, BCD requires more storage space than pure binary encoding since each decimal digit requires four bits instead of just one. Additionally, BCD operations can be more complex than binary operations, since the addition or subtraction of BCD numbers may result in a carry or borrow across multiple digits.

BCD code operations involve performing arithmetic operations on numbers represented in BCD format. The basic BCD arithmetic operations are addition and subtraction, which are performed using the same rules as decimal arithmetic.

BCD code Addition

To perform addition in BCD,

1. The two BCD numbers are added (decimal) digit by digit (in 4-bit groups), not the (binary) bit, starting from the least significant digit.
2. If the sum of two digits is less than 10 (in BCD), the result is the sum of the two digits.
3. If the sum is greater than or equal to 10, 6 (in BCD) is added to the sum. [15]
4. If there is a carry, it propagates.
5. Repeat until all digits have been added.

Example 2.5.1 Let's consider the addition of two decimal numbers in BCD format.

$$823 + 985 = 1808$$

1. We know that

$$823_D = 1000, 0010, 0011_{BCD};$$

$$985_D = 1001, 1000, 0101_{BCD}.$$

2. We start from the least significant (right most) digits, adding 3 and 5 first:

$$0011 + 0101 = 1000$$

which is less than 1010, so that is the correct result. There is no carry.
3. Next we add 2 and 8:

$$0010 + 1000 = 1010$$

[15] Can you figure out why it is 6 that should be added?

This is equal to 1010, so we add 0110 to it

$$1010 + 0110 = \underline{1}, 0000$$

We keep the last 4 bits and there is carry 1 (underlined above) forwarding to the next addition.

4. Add 8 and 9, with carry (underlined below)

$$1000 + 1001 + \underline{0001} = 1, 0010$$

This is greater than 1010, so we need to add 0110:

$$1, 0010 + 0110 = 1, 1000$$

5. Therefore, the final results is 1,1000,0000,1000.
6. Please verify that

$$1, 1000, 0000, 1000_{BCD} = 1808_D$$

BCD code Subtraction

Subtraction in BCD is performed by adding the ten's complement negative of the subtrahend to the minuend. Ten's complement negative of subtrahend is obtained by adding 1 to the nine's complement negative of the subtrahend.

Example 2.5.2 Let's consider the subtraction of two decimal numbers.

1. In decimal format:
For a human being, traditionally (compare to select the sign, then first, then subtract to calculate the magnitude),

$$985 - 823 = 162$$

To use complement method, a 4^{th} digit is needed for the sign. We know that the ten's complement negative of the subtrahend 0823 is 9177, then

$$0985 + 9177 = \cancel{1}0162$$

Discarding the first digit gives result of 0162, meaning a positive decimal number 162.

2. Now redo the subtraction in BCD format:

$$\begin{aligned} 0985_D + 9177_D &= 0000, 1001, 1000, 0101_{BCD} \\ &+ 1001, 0001, 0111, 0111_{BCD} \\ &= \cancel{1}, 0000, 0001, 0110, 0010_{BCD} \\ &= 0000, 0001, 0110, 0010_{BCD} \\ &= 0162_D \\ &= 162_D \end{aligned}$$

Note 1: when adding the two BCD numbers (the 3rd line above), do not forget the adding 0110 rule and the carry propagation.
Note 2: When interpreting the result, do not forget the carry discarding (the 4th line above).

Example 2.5.3 Let's consider another example of subtraction of two decimal numbers.

1. In decimal format:
   For a human being, traditionally (compare to select the sign, subtract to calculate the magnitude),

   $$823 - 985 = -162$$

   To use complement method, we know that the ten's complement negative of the subtrahend is 9015, then

   $$0823 + 9015 = 9838$$

   This is the ten's complement of negative number -162.
2. Now redo it in BCD format.

   $$\begin{aligned} 0823_D + 9015_D &= 0000, 1000, 0010, 0011_{BCD} \\ &+ 1001, 0000, 0001, 0101_{BCD} \\ &= 1001, 1000, 0011, 1000_{BCD} \\ &= 9838_D \end{aligned}$$

   This is the ten's complement of -162.

To Do

1. Form a pair with another of your classmates.
2. Obtain the two decimal IDs (the last four digits only).
3. Objective: In BCD format, each of you subtract your partner's ID from your ID by using the ten's complement negative of the subtrahend.
4. Examine your partner's result to make sure it is correct.
5. Your team together add up your results. Discuss your findings.

BCD arithmetic operations can be more complex than binary arithmetic operations, since carry and borrow propagation across digits is required. However, BCD allows for easy and efficient manipulation of decimal numbers using digital circuits.

The BCD code is also called 8421 code . Each decimal digit is represented by a unique combination of four bits. The four bits are weighted with values of 8, 4, 2, and 1, respectively, which are added together to give the decimal value of the digits.

2.5.2 Gray Code

Gray code , also known as reflected binary code or Gray binary code, is a binary numeral system where two consecutive values differ in only one bit position. It is named after Frank Gray, who patented the binary code in 1953.

In Gray code, the first two values are 0 and 1 (called 1-bit Gray code). The following values are obtained by reflecting the previous values and adding a 1 in front of each reflected value for odd positions. For even positions, a 0 is added in front of the reflected value. The following Table 2.7 shows the Gray code sequence for the first few numbers:

Table 2.7: Gray Code for Decimal Digits

Decimal	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

Generate Gray Code Recursively

Gray code can be generated recursively. Assume we want to generate n-bit Gray codes from $(n-1)$-bit Gray codes.

Steps:

1. Start with 1-bit Gray code:

$$\begin{array}{c}0\\1\end{array}$$

2. Generate n-bit Gray codes from $(n-1)$-bit Gray codes:

   a) Prefix ‘0‘ to the existing $(n-1)$-bit Gray codes.
   b) Prefix ‘1‘ to the reversed order of the existing $(n-1)$-bit Gray codes.
   c) Combine these two lists to get the n-bit Gray code.

   For example, starting with 1-bit Gray codes (0, 1):

   a) Prefix ‘0‘: 00, 01
   b) Prefix ‘1‘: 10, 11

   Resulting in 2-bit Gray codes:

$$\begin{array}{c}00\\01\\11\\10\end{array}$$

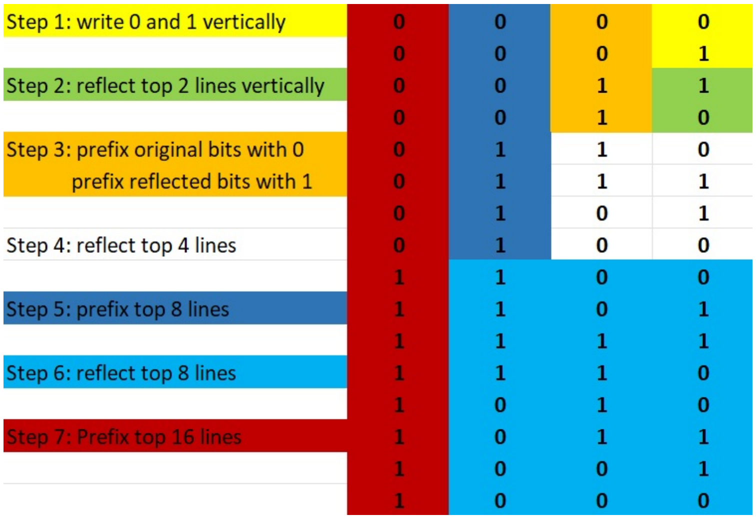
Starting with the 2-bit Gray codes (00, 01, 11, 10):

a) Prefix ‘0‘: 000, 001, 011, 010
b) Prefix ‘1‘: 110, 111, 101, 100

Resulting in 3-bit Gray codes:

000
001
011
010
110
111
101
100

The detailed steps to generate 4-bit Gray codes are illustrated in Figure 2.1.



Step 1: write 0 and 1 vertically	0	0	0	0
	0	0	0	1
Step 2: reflect top 2 lines vertically	0	0	1	1
	0	0	1	0
Step 3: prefix original bits with 0	0	1	1	0
prefix reflected bits with 1	0	1	1	1
	0	1	0	1
Step 4: reflect top 4 lines	0	1	0	0
	1	1	0	0
Step 5: prefix top 8 lines	1	1	0	1
	1	1	1	1
Step 6: reflect top 8 lines	1	1	1	0
	1	0	1	0
Step 7: Prefix top 16 lines	1	0	1	1
	1	0	0	1
	1	0	0	0

Figure 2.1: Generating 4-bit Gray Codes

Generate Gray Code Using Binary Reflected Gray Code Formula

For a given binary number b, the corresponding Gray code g can be computed using:

$$g = b \oplus (b \gg 1)$$

where $\oplus$ denotes the bitwise XOR operation and $\gg$ denotes the right shift operation.

For example, we want to convert decimal 5 (which is 0101 in binary) to Gray code.

Steps:

1. Binary: 0101
2. Right shift by 1 bit: 0010
3. XOR operation: $0101 \oplus 0010 = 0111$

So, the Gray code for decimal 5 or binary 0101 is 0111.

Applications of Gray Code

1. **Rotary encoders**: Gray code is used in rotary encoders to track the position of a rotating object, ensuring that only one bit changes at a time, which reduces errors during the transition between position states.
2. **Digital communication**: Gray code helps minimize errors in digital communication systems, where data might be transmitted over noisy channels.
3. **Error correction**: Gray code is used in error correction schemes where the goal is to detect and correct errors that might occur due to the simultaneous change of multiple bits.

Further investigation can be found in [4].

[4]: Doran (2007), 'The Gray Code'

2.5.3 ASCII Character Code

ASCII (American Standard Code for Information Interchange) is a character encoding standard used to represent text in computers and other devices. It assigns a unique numerical value to each character, including letters, numbers, punctuation marks, and control characters. In table 2.8 are the ASCII codes for some common characters:

Table 2.8: ASCII Codes for Some Characters

Character	ASCII Code in Decimal	ASCII Code in Binary
A	65	100,0001
B	66	100,0010
C	67	100,0011
a	97	110,0001
b	98	110,0010
c	99	110,0011
0	48	011,0000
1	49	011,0001
2	50	011,0010
!	33	010,0001
?	63	011,1111
,	44	010,1100
.	46	010,1110
space	32	010,0000
newline	10	000,1010

To represent a character in ASCII, we can use its decimal value and convert it to binary using 7 bits (which is the maximum number of bits needed to represent any ASCII character). For example, the ASCII code for the letter A is 65, which can be represented in binary as 01000001.

A student handout on ASCII code can be found at IBM [5].

[5]: Intel (2010), *Student Handout: ASCII Computer Code*

2.5.4 Error Detection Code

Error detection codes are used to ensure the integrity of data transmitted over a communication channel or stored in a storage medium. These codes add redundant information to the data, which can be used to detect errors that may have occurred during transmission or storage. There are many error detection codes and variations depending on the specific application and requirements. Some commonly used error detection codes are

Parity Check In a parity check code, a single bit (called the parity bit) is added to the data to make the total number of 1's in the data (including the parity bit) even or odd. During transmission or storage, the receiver can check the parity bit and compare it with the expected parity to detect whether any bit has been flipped.

Example 2.5.4 We want to transmit 6-bit data 101101. We can add a parity check code at the end.

1. For even parity , the code will be $101101\underline{0}$ to make the total number of 1s in the data (including the parity bit) even. If we receive a code whose number of 1s is odd, we know there must be at least one error during the transmission, but we do not know which bit is wrong, therefore, we cannot correct the error.
2. For odd parity check , the code will be $101101\underline{1}$ to make the total number of 1s in the data (including the parity bit) odd.

Checksum A checksum code adds a sequence of values (usually in binary) to the data and transmits or stores the sum. The receiver can then recalculate the sum of the data and compare it with the transmitted sum to detect whether any bits have been changed.

Example 2.5.5 ISBN (International Standard Book Number) is a type of checksum code. It uses a check digit to help detect errors when entering or transmitting the book's identification number.
An ISBN consists of 10 or 13 digits, depending on the edition of the book. The last digit of the ISBN is the check digit, which is calculated based on the other digits in the ISBN.
To calculate the check digit of a 10-digit ISBN, you can use the following algorithm:

1. Multiply each of the first 9 digits by a weight factor (from 10 to 2).
2. Add up the results of the multiplication.
3. Divide the sum by 11.
4. Subtract the remainder from 11 to get the check digit. If the remainder is 0, the check digit is 0.

Example 2.5.6 The check digit of the ISBN-10 of a textbook is covered by a microprocessor chip, as in Figure 2.2. Can we determine the check digit?
Answer: to calculate the check digit of a 10-digit ISBN, we follow the following steps:

1. Multiply each of the first 9 digits by a weight factor (from 10

to 2), add up the results of the multiplication, we have

$$10\times0+9\times1+8\times3+7\times4+6\times5+5\times4+4\times9+3\times8+2\times9 = 189$$

2. Divide the sum 189 by 11 we get a remainder 2.
3. Subtract the remainder 2 from 11 we get the check digit 9. Therefore, the check digit is 9. The ISBN-10 is 0134549899.

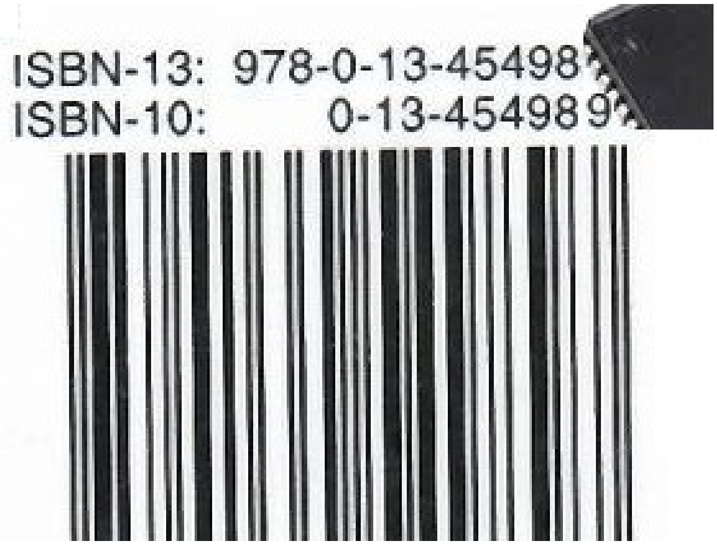


Figure 2.2: Example of ISBN

For a number to be a valid ISBN-10, one of the requirements is that the sum of the ten digits, each multiplied by its (integer) weight, descending from 10 to 1, is a multiple of 11. That is,

$$\sum_{i=1}^{10}(11-i)x_i \equiv 0 \pmod{11},$$

where x_i is the i^{th} digit.
To convert an ISBN-10 to ISBN-13, we just add 978 before in front of the ISBN-10, and then re-calculate the check digit.
For a number to be a valid ISBN-13, one of the requirements is that the weighted sum of all the digits, each multiplied by its (integer) weight, is a multiple of 10. That is,

$$(x_1+3x_2+x_3+3x_4+x_5+3x_6+x_7+3x_8+x_9+3x_{10}+x_{11}+3x_{12}+x_{13}) \equiv 0 \pmod{10},$$

where x_i is the i^{th} digit.
[16]

To Do

In Figure 2.2, the last two digits of the ISBN-13 are missing. Try to recover the two digits.

Cyclic Redundancy Check (CRC) A CRC code adds a fixed number of bits (called the checksum) to the data based on a mathematical function.

[16] Develop an algorithm, similar to the above example, to determine the check digit of an ISBN-13, given the first 12 digits. Steps:

1.
2.
3.
4.

The receiver can then apply the same function to the received data and compare the calculated checksum with the transmitted checksum to detect errors.

Hamming Code A Hamming code adds parity bits to the data in such a way that any single-bit error can be corrected, and any two-bit error can be detected.

2.5.5 Error Correction Code

Error correction codes (ECC) are codes that are used to detect and correct errors that may occur during data transmission or storage. These codes are commonly used in computing, digital communications, information theory, coding theory, and data storage systems, where data integrity is important.

One example of an error correction code is the linear block code.

Suppose we want to transmit k bits of data with n bits of code, which is called (n, k) code.

The data bits are $d_1, d_2, \cdots, d_k$. The code bits are $c_1, c_2, \cdots, c_n$.

In general case, the linear block codes are linear combinations of the data digits.

For a set of linear block codes to detect up to t bit of error, the minimum Hamming distance between any pair of codes should satisfy $d_{min} = 2t + 1$; to correct up to t bits of error, the minimum Hamming distance should satisfy $d_{min} = t + 1$.

A special case is the first k digits are data, the last $m = n - k$ digits are linear combinations of the data.

$$\begin{cases} c_1 & = & d_1 \\ c_2 & = & d_2 \\ \vdots & & \\ c_k & = & d_k \\ c_{k+1} & = & h_{11}d_1 \oplus h_{12}d_2 \oplus \cdots \oplus h_{1k}d_k \\ \cdots & & \\ c_n & = & h_{m1}d_1 \oplus h_{m2}d_2 \oplus \cdots \oplus h_{mk}d_k \end{cases}$$

For example, in a $(6, 3)$ code, assume the data are d_1, d_2 and d_3, the code are $c_1, c_2, \cdots, c_6$.

We assume the following generator rule:

$$\begin{cases} c_1 & = & d_1 \\ c_2 & = & d_2 \\ c_3 & = & d_3 \\ c_4 & = & d_1 \oplus d_3 \\ c_5 & = & d_2 \oplus d_3 \\ c_6 & = & d_1 \oplus d_2 \end{cases}$$

For all the possible data words from 000 to 111, the corresponding code words are:

Data Word **d**	Code word **c**	Data Word **d**	Code word **c**
000	000000	100	100101
001	001110	101	101011
010	010011	110	110110
011	011101	111	111000

The minimum Hamming distance between any two code words is $d_{min} = 3$, hence, the code can detect up to two bits of error and correct up to one bit of error.

We can summarize the generator rule into a generator matrix:

$$[G] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

In matrix form, the code is represented by the product of the data and generator matrix:

$$\mathbf{c} = \mathbf{dG} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Exercise 2.5.1 Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.6 Representation of data

Data refers to the symbols that represent people, events, things, and ideas. Data can be a name, a number, the colors in a photograph, or the notes in a musical composition.

Data Representation refers to the form in which data is stored, processed, and transmitted.

2.6.1 Data in Computers

At the lowest level, a computer is an electronic machine working by controlling the flow of electrons. Devices such as smartphones, iPods, and computers store data in digital formats that can be handled by electronic circuitry.

Computers use binary — the digits 0 and 1 — to store data. A binary digit, or bit, is the smallest unit of data in computing. It is represented by a 0 or a 1. Binary numbers are made up of binary digits (bits). For example, the binary number 1001. The circuits in a computer's processor are made up of billions of transistors. A transistor is a tiny switch that is activated by the electronic signals it receives. The transistor can be turned on to enable access to the capacitor, either to charge it up and store a 1, discharge it and store a 0, or read the stored charge. The digits 1 and 0 used in binary reflect the on and off states of a transistor.

The 0 and 1 numbers are logic levels (0 = logic 0, 1 = logic 1), which are created by voltages in a circuit. In positive logic, 0 is formed by a low voltage level, and 1 is formed by a high voltage level. In negative logic, 0 is formed by a high voltage level, and 1 is formed by a low voltage level. In general, for a low logic stage, acceptable input signal voltages range from 0 volts to 0.8 volts and for a high logic state, 2 volts to 5 volts.

Computer programs are sets of instructions. Each instruction is translated into machine code - simple binary codes that activate the CPU. Programmers write computer code, and this is converted by a translator into binary instructions that the processor can execute. All software, music, documents, and any other information that is processed by a computer, is also stored using binary.

2.6.2 Data on Hard Disks

When data can be stored on hard disks inside computers. Hard disks represent binary data using magnetic properties. The platters inside a hard disk drive are coated with a thin layer of magnetic material, usually a form of iron oxide. This magnetic coating allows the surface to be magnetized in different directions. The magnetic coating on the disk's surface is divided into tiny regions called magnetic domains. Each domain can be magnetized in different directions. The direction of magnetization of a domain is used to represent binary data. Typically, one direction of magnetization (e.g., north) represents a binary 1, while the opposite direction (e.g., south) represents a binary 0. When data is written to the disk, a current flows through the coil in the write head and generates a magnetic field that aligns the magnetic domains in the desired direction. By altering the direction of the electric current and thus the magnetic field, the write head sets the domains to represent either a 1 or a 0.To read data, the read head detects the magnetization direction of the domains on the disk's surface. To read the data from the hard disk, the read head uses a different mechanism called magnetoresistance to sense the magnetic fields of the domains. Changes in the magnetic field cause changes in electrical resistance, which the read head detects and converts into binary data.

2.6.3 Data in Wireless Communication Systems

In wireless communication systems, direct voltage signals like high (5 volts) and low (0 volt) cannot transmittd efficiently. Binary 1s and 0s are represented through a combination of encoding and modulation techniques. Encoding schemes prepare the binary data for transmission, while modulation techniques alter the carrier signal to convey the data over radio waves. For example, binary signals 1 and 0 by can be transmitted by turning on and off a sinusoidal signal. This is called amplitude modulation (AM) technique, where the sinusoidal signal is called a carrier. Binary signals can also be transmitted with a sinusoidal signal with two different frequencies. This is called frequency modulation (FM) technique.

3 Boolean Algebra and Logic Gates

3.1 Introduction

In computer engineering, the circuits implements binary logic in all computers and digital devices.

A mathematical method, called Boolean algebra can be used to simplify electric ciruits.

3.1.1 Two-valued Boolean Algebra

Two-valued Boolean algebra, also known as binary Boolean algebra, is a branch of mathematics that deals with the study of two possible values or states, commonly represented as 0 and 1. It is based on a set of operations that are defined on these values, which allow for the manipulation and evaluation of logical expressions.

Two-valued Boolean algebra has many practical applications, particularly in computer science and digital electronics, where it is used to design and analyze digital circuits and algorithms. It is also used in propositional logic, which is the branch of logic that deals with the manipulation of propositions using logical operations.

3.1.2 Logic Operations

In two-valued Boolean algebra, the two values, 0 and 1, are often used to represent the truth values of propositions in a logical system. For example, 0 may represent "false" and 1 may represent "true". The basic operations of two-valued Boolean algebra are:

AND The AND operation takes two values as input and produces 1 as output only if both inputs are 1. Otherwise, it produces 0. AND operation can be written as · in an expression.

The rules for AND operator are

A	B	F=A · B
0	0	0
0	1	0
1	0	0
1	1	1

OR The OR operation takes two values as input and produces 0 as output only if both inputs are 0. Otherwise, it produces 1. OR operation can be written as + in an expression.

The rules for OR operator are

A	B	F=A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT The NOT operation takes a single value as input and produces the opposite value as output. That is, if the input is 1, the output is 0, and if the input is 0, the output is 1. Essentially, the operator reverses the logical value associated with the expression on which it operates. NOT operation can be written as an apostrophe or a horizontal bar above a variable or an expression. For example,

$$F = \text{ NOT } A = A' = \overline{A}.$$

The rules for NOT operator are

A	F=A'
0	1
1	0

3.1.3 Logic Gates

Logic gates are fundamental building blocks in digital electronics. They perform basic logical functions that are essential for digital circuits. Each gate has a specific function based on Boolean algebra and performs basic logical operations on one or more binary inputs to produce a single binary output. Logic gates are used to create complex circuits, including arithmetic operations, memory, and processors.

The symbols of the three fundamental logic gates (AND, OR, and NOT) are in Figure 3.1.

Other logic gates include NAND, NOR, XOR, and XNOR.

Logic gates can be implemented in various ways, depending on the technology and context. We discuss below how to build fundamental logic gates with diodes and with transistors.

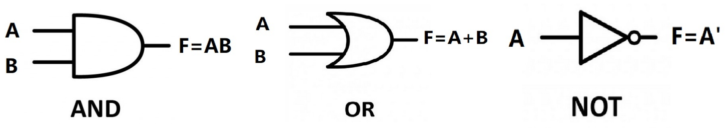


Figure 3.1: Logic Gates

Logic Gates with Diode

A diode as in Figure 3.2 is a semiconductor device with two terminals: a positive terminal, called the anode and a negative terminal, called the cathode. When a positve high voltage is applied to the anode, it is called forward-biased, and the current follws from positive to negative. If the diode is reverse-biased, almost no current flows.

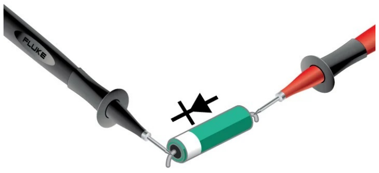


Figure 3.2: A diode and its Symbol

A diode acts as a one-way switch for current. When it is forward-biased, the normal voltage drop on the diode is about 0.5 V to 0.8 V. We can use this fact in a diode circuit, for example, to calculate the resistance to limit the current: if the current flowing through an LED is too large, the LED could be destroyed, but if the current is too small, the LED might not be bright enough.

Both AND and OR gates can be built with diode circuits, as showed in Figures 3.3 and 3.4.

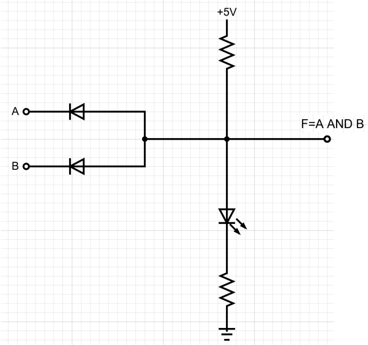


Figure 3.3: Diode Circuit for AND Gate

In Figure 3.3, when both input A and B are low (around 0 volt), the two diodes on the left are forward-biased. The top of the bottom diode is about 0.7 volts, and the diode does not conduct. There is no current flowing through the bottom diode, and the LCD is not lit. Therefore, the output is low. That is

$$0 \text{ AND } 0 = 0.$$

When any input to the two left diodes is low, we have the same result. This is

$$1 \text{ AND } 0 = 0; 0 \text{ AND } 1 = 0.$$

When both A and B are high (for example 5 volts), the left diodes do not conduct and the current flows through the bottom one. The light is on and the output is high. This is

$$1 \text{ AND } 1 = 1.$$

Exercise 3.1.1 Explain how the circuit in Figure 3.4 works as an OR function.

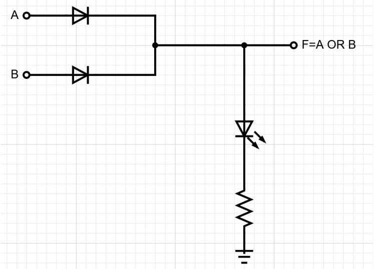


Figure 3.4: Diode Circuit for OR Gate

Logic Gates with Transistor

A transistor is a semiconductor device used to amplify or switch electronic signals and electrical power. It's a fundamental component in modern electronic devices, including computers, smartphones, and radios.

Transistors typically have three terminals: the emitter, the base, and the collector. There are different types of transistors, such as Bipolar Junction Transistors (BJTs) and Field-Effect Transistors (FETs), each with distinct characteristics.

NPN and PNP transistors as in Figure 3.5 are the two primary types of BJTs. Both serve similar purposes but have different configurations and operate in opposite ways.

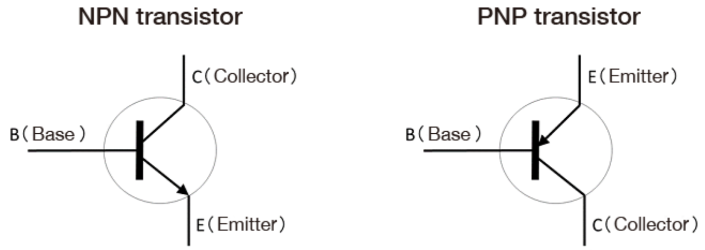


Figure 3.5: Transistors: NPN and PNP

Here we discuss NPN type only for example. Basically,

- When NPN type transitors are connected in series, they can implement AND or NAND gates;
- When connected in parallel, they can implement OR or NOR gates.

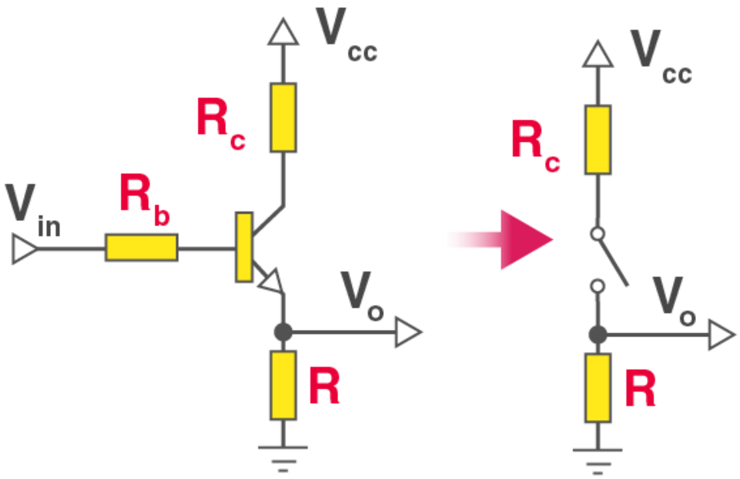


Figure 3.6: Transistor as a switch

In Figure 3.6, when the input voltage at the base is high, the transistor conducts and the current can flow from the collector to the emitter. When the input is low, the transistor does not conduct and the current cannot flow from the collector to the emitter. Therefore, the transistor serves as a switch with the control of input signal.

In Figure 3.7, both transistors are connected in series. When both input A and B are high (for example, 5 volts), both transistors conduct. The output is high. This is

$$1 \text{ AND } 1 = 1.$$

If either A or B or both are low, at least one transistor cannot conduct and no current flows from the voltage source (terminal VCC) to the ground (terminal GND). The output voltage will be low. This is

$$1 \text{ AND } 0 = 0; 0 \text{ AND } 1 = 0; 0 \text{ AND } 0 = 0.$$

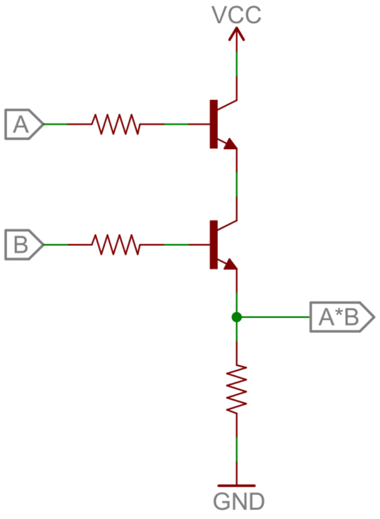


Figure 3.7: Transistor Circuit for AND Gate

Exercise 3.1.2 How do you modify Figure 3.7 to implement the NAND function?

In Figure 3.8, the two transistors are connected in parallel. When both input A and B are low (around 0 volt), both transistors do not conduct and thus there are no currents flowing out of the emitters. The output is low. This is

$$0 \text{ OR } 0 = 0.$$

If either A or B is high, one transistor conducts and there is current flowing out from the emitter so the output voltage is high. This is

$$1 \text{ AND } 0 = 1; 0 \text{ AND } 1 = 1.$$

If both A and B are high, both transistors conduct and the output is high. This is

$$1 \text{ AND } 1 = 1.$$

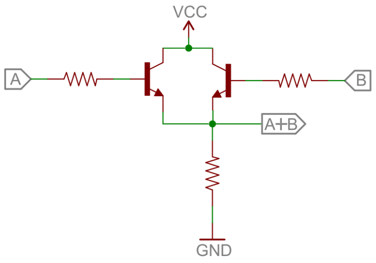


Figure 3.8: Transistor Circuit for OR Gate

Exercise 3.1.3 How do you modify Figure 3.8 to implement the NOR function?

In Figure 3.9, when input A is high, the transistor conducts and serves as a closed switch. The output voltage is low. That is

$$\text{NOT } 1 = 0.$$

when input A is low, the transistor does not conduct and serves as a opened switch. No current flows into the colector so the output voltage is high. That is

$$\text{NOT } 0 = 1.$$

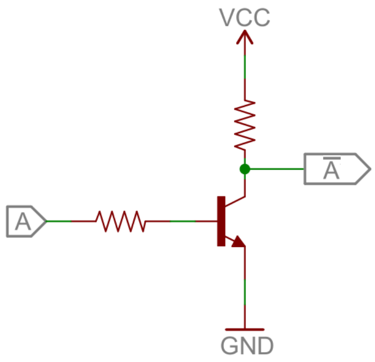


Figure 3.9: Transistor Circuit for NOT Gate

3.2 Boolean Functions

A Boolean function is a mathematical function that takes one or more Boolean variables as input and produces a single Boolean output value. Boolean functions are used in digital logic circuits, computer programming, and many other fields where logical operations are required.

In general, a Boolean function can have any number of input variables, but the output value can only be one of two possible values: 0 or 1. Boolean functions are often represented using truth tables or logic diagrams, which show the possible input values and the resulting output value for each combination of inputs.

There are many different types of Boolean functions, but some of the most common ones include:

NOT This is a unary function that takes a single Boolean input and produces the opposite Boolean value as output. For example, if the input is 0, the output is 1, and if the input is 1, the output is 0.

AND This is a binary function that takes two Boolean inputs and produces a Boolean value that is true (1) only if both inputs are true. For example, if the inputs are 1 and 1, the output is 1. If either input is 0, the output is 0.

OR This is a binary function that takes two Boolean inputs and produces a Boolean value that is true (1) if at least one of the inputs is true. For example, if the inputs are 0 and 1, the output is 1. If both inputs are 0, the output is 0.

XOR This is a binary function that takes two Boolean inputs and produces a Boolean value that is true (1) only if the inputs are different. For example, if the inputs are 0 and 1, the output is 1. If the inputs are the same (both 0 or both 1), the output is 0.
The rules for XOR operator are as in Table 3.1.

Table 3.1: Truth Table of XOR Function

A	B	F=A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

NAND A NAND gate (short for “not-AND") is a logic gate that produces a false (0) output only when all of its inputs are true (1). A NAND gate is the logical negation of an AND gate.
NAND gates are also used in computer memory systems, where they are used to implement dynamic random access memory (DRAM) cells. In this application, the NAND gate is used to ensure that the stored data is refreshed periodically to prevent it from decaying over time.
The rules for NAND operator are as in Table 3.2.

Table 3.2: Truth Table of NAND Function

A	B	F=(AB)′
0	0	1
0	1	1
1	0	1
1	1	0

NOR A NOR gate (short for “not-OR") is a logic gate that produces a true (1) output only when all of its inputs are false (0). In other words, a NOR gate is the logical negation of an OR gate.
NOR gates are also used in computer memory systems, where they are used to implement static random access memory (SRAM) cells. In

this application, the NOR gate is used to store the data in the memory cell and to control the read and write operations.

The rules for NOR operator are as in Table 3.3.

A	B	F=(A+B)′
0	0	1
0	1	0
1	0	0
1	1	0

Table 3.3: Truth Table of NOR Function

The binary variables and logic operations are used in Boolean algebra. The algebraic expression, which is known as Boolean Expression, consists of the constant value 1 and 0, logical operation symbols, and binary variables, and is used to describe more complicated Boolean Functions. One example is

$$F(A, B, C, D) = A + BC' + D$$

3.2.1 Represent Boolean Functions with Truth Table

A truth table is a table used to represent the relationship between the input and output of a logical expression. It lists all possible combinations of input values and their corresponding output values, based on the rules of the logical expression.

For example, truth tables for the logical operators “AND", “OR", and “NOT" are already given as operation rules in previous sections.

The truth table for XOR, NAND, NOR functions are in Tables 3.1, 3.2, and 3.3.

Class Activity

1. Examine the truth tables above.
2. Can you figure out how to develop a truth table for a given Boolean function?
3. For a Boolean function with 4 input varialbes (A, B, C, and D) and one output, how many rows and colums do you expect in the truth table?
4. Once you are done, discuss your result with your team members.
5. Genealize your conclusion.
6. Create a truth table for the following Boolean function and compare results.

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

[1]

[1] Write here the steps to develop a truth table:

1.
2.
3.
4.
5.

3.2.2 Represent Boolean Functions with Logic Circuits

Boolean functions can be represented using logic circuits, which are composed of logic gates. Here is anexample to show how to represent Boolean functions with logic circuits.

Example 3.2.1 Represent Boolean function $F_1 = x + y'z$ with logic gates.

Solution:

The logic circuit can be implemented as in Figure 3.10. Another example of an actual project is illustrated in Figure 3.11.

Can you imagine how to connect the circuit on the breadboard to implement the Boolean function above?

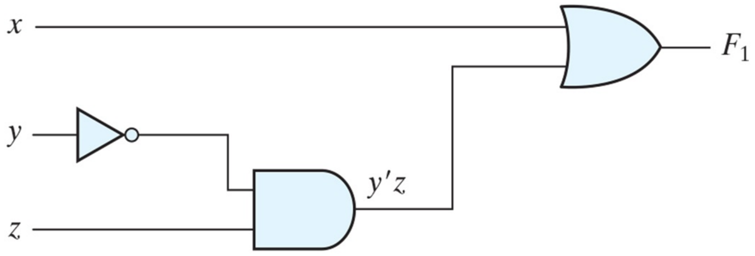


Figure 3.10: A Logic Circuit

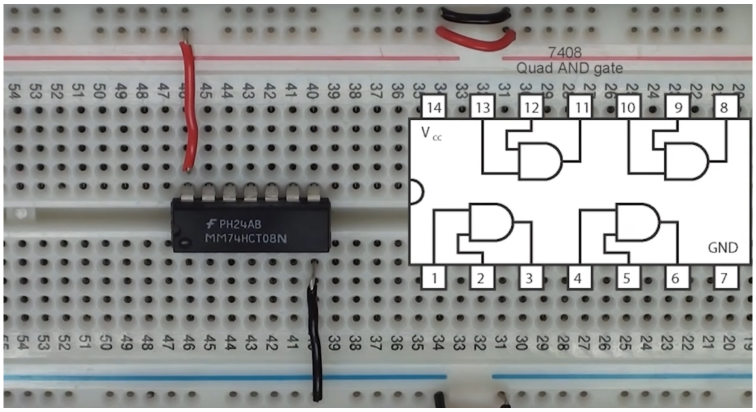


Figure 3.11: A Logic Circuit

Exercise 3.2.1 Represent the following Boolean function with logic gates. Compare your answer with your partner.

$$F = x'y'z + x'yz + xy'$$

Exercise 3.2.2 Given the following logic circuit in Figure 3.12, can you write its output as a Boolean function? Compare your answer with your partner.

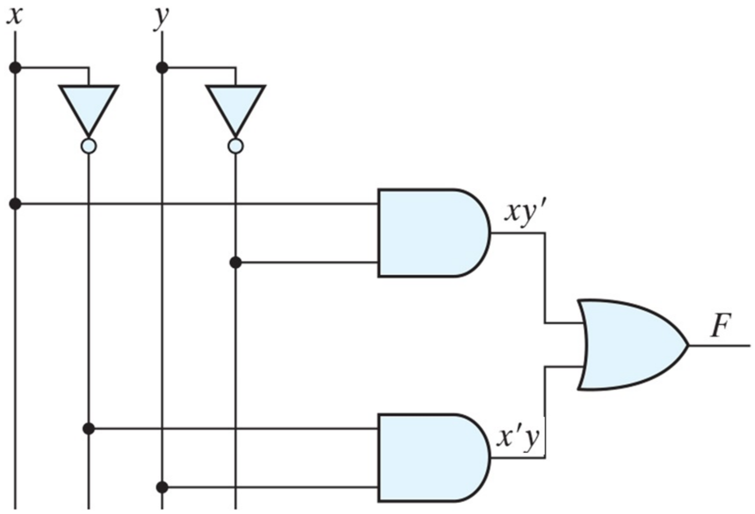


Figure 3.12: A Logic Circuit

3.3 Fundamentals in Boolean Algebra

3.3.1 Operator Precedence

In logic, operator precedence refers to the order in which logical operations are performed in a compound expression. The precedence of logical operators determines the grouping of terms in an expression and the order in which they are evaluated. Here are the precedence rules for the most common logical operators, in order from highest to lowest:

1. Negation (NOT)
2. Conjunction (AND)
3. Disjunction (OR)
4. Exclusive disjunction (XOR)

These rules dictate that operations of higher precedence are evaluated first. For example, in the expression "A AND B OR C", the AND operation is evaluated first because it has higher precedence than OR.

Parentheses can be used to override the default precedence rules and force a specific order of evaluation.

3.3.2 Basic Postulates and Theorems

The basic postulates and theorems in Boolean algebra form the foundation of the field and are essential for understanding and manipulating Boolean expressions. They provide a set of rules that can be used to simplify complex Boolean expressions, perform logical operations, and design digital circuits.

By using the basic postulates and theorems, Boolean expressions can be simplified and transformed into an equivalent form that is easier to understand and analyze. This simplification can greatly reduce the complexity of Boolean expressions, making them more manageable and easier to work with.

Some basic postulates and theorems in Boolean algebra are listed in Table 3.4[2]

Table 3.4: Basic Postulates and Theorems in Boolean Algebra

Postulate or Theorem	OR	AND
Postulate	$x+0=x$	$x \cdot 1 = x$
Postulate	$x+x'=1$	$x\,x' = 0$
Theorem	$x+x=x$	$x\,x = x$
Theorem	$x+1=1$	$x \cdot 0 = 0$
Involution	$(x')'=x$	
Commutative	$x+y=y+x$	$xy = yx$
Associative	$x+(y+z)=(x+y)+z$	$x(yz) = (xy)z$
Distributive	$x(y+z)=xy+xz$	$x+yz = (x+y)(x+z)$
DeMorgan	$(x+y)'=x'y'$	$(xy)' = x'+y'$
absorption	$x+xy=x$	$x(x+y) = x$

Class Activity

1. Form a pair with your neighbor.
2. Test each other the formulas in the table.
3. Make sure both of you have understood and memorized all formulas in the table.

3.3.3 Complement of Boolean Functions

The generalized form of the DeMorgan's theorems state that the complement of a function is obtained by interchanging AND and OR operators and complementing each literals.

1. Complement of OR:

$$(A + B + C + \cdots + F)' = A'B'C' \cdots F'$$

2. Complement of AND:

$$(ABC \cdots F)' = A' + B' + C' + \cdots + F'$$

Example 3.3.1 Find the complement of the following Boolean function

$$F = x'yz' + x'y'z$$

Solution:

1. The operator with the lowest preceedence is the OR (+) operator. Therefore, function F is the OR of two expressions. We use the rule for complement of OR

$$F' = (x'yz')'(x'y'z)'.$$

[2] Try to memorize this table.

2. Each of the two expressions above is a complement of AND, therefore,

$$(x'yz')' = x + y' + z; (x'y'z)' = x + y + z'.$$

3. The final result is

$$F' = (x + y' + z)(x + y + z').$$

Example 3.3.2 Find the complement of the following Boolean function

$$F = x(y'z' + yz)$$

Solution:

$$F' = x' + (y'z' + yz)' = x' + (y'z')'(yz)'$$
$$= x' + (y + z)(y' + z') = x' + yz' + y'z$$

Exercise 3.3.1 Find the complement of Boolean Function

$$F = x'y + xy'$$

Exercise 3.3.2 Find the complement of Boolean Function

$$F = (a + c)(a + b')(a' + b + c')$$

Exercise 3.3.3 Find the complement of Boolean Function

$$F = z + z'(v'w + xy)$$

4 Gate-Level Minimization

4.1 Introduction

Gate-level minimization is a process of reducing the number of logic gates and simplifying the logic circuit, while maintaining the same functionality. The goal of gate-level minimization is to reduce the complexity of a logic circuit, which can improve performance, reduce power consumption, and decrease the cost of production.

There are several techniques for gate-level minimization. In this class, we will discuss the two most fundamental methods: Boolean algebra and Karnaugh maps. Both techniques can be used to simplify a logic circuit by combining redundant logic gates, eliminating unnecessary logic gates, and reducing the number of inputs and outputs.

Boolean algebra is a fundamental tool for gate-level minimization. It involves manipulating logical expressions using Boolean operators such as AND, OR, and NOT. By applying Boolean algebraic rules, logical expressions can be simplified and expressed in a more compact form.

Karnaugh maps are another useful tool for gate-level minimization. They are graphical representations of Boolean functions that allow for easy identification of redundant logic gates. By grouping adjacent cells in a Karnaugh map that correspond to 1's in a Boolean function, the expression can be simplified.

Gate-level minimization is an important step in digital circuit design that can improve efficiency and reduce the cost of production. Designers often use a combination of these methods to achieve the best results.

4.2 Simplifying Boolean Functions with Boolean Algebra

Some of the most common theorems and postulates that can be used to simplify Boolean functions are

To Memorize

1. Identity theorem: The identity theorem states that any variable ANDed with 1 is itself, and any variable ORed with 0 is itself. This can be expressed as

$$A \cdot 1 = A \text{ and } A + 0 = A$$

2. Complement theorem: The complement theorem states that every variable has a complement that is its opposite. This can be expressed as

$$A + A' = 1 \text{ and } A \cdot A' = 0$$

3. Commutative theorem: The commutative theorem states that the order of the operands in an OR or an AND operation does not matter. This can be expressed as

$$A + B = B + A \text{ and } A \cdot B = B \cdot A$$

4. Associative theorem: The associative theorem states that the grouping of operands in an OR or an AND operation does not matter. This can be expressed as

$$A + (B + C) = (A + B) + C \text{ and } A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

5. Distributive theorem: The distributive theorem states that an OR operation over a set of operands ANDed with another variable is equivalent to the OR operation of the operands individually ANDed with the variable. Similarly, an AND operation over a set of operands ORed with another variable is equivalent to the AND operation of the operands individually ORed with the variable. This can be expressed as

$$A \cdot (B + C) = A \cdot B + A \cdot C \text{ and } A + (B \cdot C) = (A + B) \cdot (A + C)$$

6. Absorption laws to combine terms or eliminate redundant variables.

$$A \cdot (A + B) = A \text{ and } A + A \cdot B = A$$

7. De Morgan's theorem: De Morgan's theorem states that the complement of a logical expression that is ANDed or ORed is equivalent to the OR or AND of the complements of the operands. This can be expressed as

$$(A + B)' = A' \cdot B' \text{ and } (A \cdot B)' = A' + B'$$

Other Tricks to Memorize

1. Identify Common Sub-expressions: look for common sub-expressions within the expression. If you find repeating terms, factor them out. Example:

$$AB + ABCD = AB(1 + CD).$$

Here AB is the common sub-expression.

2. Use Idempotent Law:

$$A \cdot A = A \text{ and } A + A = A.$$

If you have repeating variables in the same operation, simplify them.

3. Apply Null Element: for AND operations, use the null element

$$A \cdot 0 = 0;$$

For OR operations, use the null element

$$A + 1 = 1.$$

4. Consider Double Negation: eliminate double negations by using the rule: NOT (NOT A) = A.
5. Repeat and Iterate: continue applying these rules and simplifying sub-expressions until you can't simplify the expression further.

Example 4.2.1 Using basic theorems and postulates to simplify Boolean expression

$$F = x'y'z + xyz + x'yz + xy'z$$

Solution: [1]

$$F = x'y'z + \underline{xyz + x'yz} + xy'z$$

$$= x'y'z + (x + x')yz + xy'z$$

$$= \underline{x'y'z} + yz + \underline{xy'z} = (x' + x)y'z + yz$$

$$= y'z + yz = (y' + y)z = z.$$

[2]

[1] Can you identify what property is used in each step? Discuss with your partner or ask the class.

[2] What is the significance of the above simplification? Hint: can you implement the Boolean function with logic circuit?

Example 4.2.2

$$F = xy + xy' = x(y + y') = x$$

Example 4.2.3

$$F = (x + y)(x + y') = xx + \underline{xy' + xy} + yy'$$

$$= x + x(y + y') = x + x = x$$

Example 4.2.4

$$F = xyz + x'y + xyz' = \underline{xyz} + x'y + \underline{xyz'}$$

$$= xy(z + z') + x'y = xy + x'y = y$$

Example 4.2.5

$$F = (x + y)'(x' + y')' = (x'y')(xy)$$

$$= x'y'\underline{xy} = x'\underline{(y'y)}x = 0$$

Or, consider using $A' \cdot B' = (A + B)'$,

$$F = (x + y)'(x' + y')' = (x + y + x' + y')' = (1 + 1)' = 0$$

Example 4.2.6

$$F = (a + b + c')(a'b' + c) = aa'b' + ac + ba'b' + bc + c'a'b' + c'c$$

$$= 0 + ac + 0 + bc + a'b'c' + 0 = ac + bc + a'b'c'$$

Example 4.2.7

$$F = a'bc + abc' + abc + a'bc' = \underline{a'bc} + \underline{\underline{abc'}} + \underline{abc} + \underline{\underline{a'bc'}}$$

$$= bc + bc' = b(c + c') = b$$

Example 4.2.8

$$F = xyz + x'y + xyz' = \underline{xyz} + x'y + \underline{xyz'} = xy + x'y = y$$

Example 4.2.9

$$F = x'yz + xz = \underline{(x'y + x)}z = \underline{(x + x')(x + y)}z$$

$$= (x + y)z$$

[3]

Example 4.2.10

$$F = (x + y)'(x' + y') = (x'y')(x' + y')$$

$$= x'y'x' + x'y'y' = x'y' + x'y' = x'y'$$

[3] Use distributive property on the two underlined terms.

Example 4.2.11

$$F = xy + x(wz + wz') = xy + xw(z + z') = xy + xw = x(y + w)$$

Example 4.2.12

$$F = (yz' + x'w)(xy' + zw') = yz'xy' + yz'zw' + x'wxy' + x'wzw'$$

$$= 0 + 0 + 0 + 0 = 0$$

Example 4.2.13

$$F = (x' + z')(x + y' + z') = \underline{x'x} + x'y' + x'z' + xz' + y'z' + \underline{z'z'}$$

$$= x'y' + x'z' + xz' + y'z' + z' = x'y' + \underline{x'z' + xz'} + y'z' + z'$$

$$= x'y' + z' + y'z' + z' = x'y' + \underline{y'z' + z'} = x'y' + z'(1 + y') = x'y' + z'$$

Example 4.2.14

$$x'z' + xyz + xz' = \underline{x'z'} + xyz + \underline{xz'} = z' + xyz = (z' + xy)(z' + z) = z' + xy$$

Pay special attention to the following [4]

$$z' + xyz = z' + xy$$

Example 4.2.15

$$(x'y' + z)' + z + xy + wz = \underline{(x + y)z' + z} + xy + wz$$

$$= (z + x + y)(z + z') + xy + wz$$

$$= \underline{z} + x + \underline{\underline{y + xy}} + \underline{wz} = x + y + z.$$

Example 4.2.16

$$w'x(\underline{z' + y'z}) + x(\underline{w + w'yz}) = w'x(z' + y')(z' + z) + x(w + w')(w + yz)$$

$$= x[\underline{w'(y' + z')} + (w + yz)] = x[(\underline{w + yz})' + (\underline{w + yz})] = x$$

[5]

Example 4.2.17

$$\underline{(w' + y)(w' + y')}(w + x + y'z) = w'(y + y')(w + x + y'z)$$

$$= w'(w + x + y'z) = w'(x + y'z)$$

[4] or in general, by using the distributive property, we have

$$A + A'BC = A + BC$$

This is very useful. Can you memorize it?

[5] Can you try it again with the following property? Might be easier than the above method.

$$A + A'BC = A + BC$$

Example 4.2.18

$$wxy'z + w'xz + wxyz = \underline{wxy'z} + w'xz + \underline{wxyz} = wxz + w'xz = xz$$

4.3 Minterms and Maxterms

4.3.1 Definitions

Minterms and maxterms are two important concepts that are used in the simplification of Boolean expressions.

A minterm is a product term in which each variable appears once, either in its complemented form or uncomplemented form. For example, in a two-variable system, there are four minterms: A'B', AB', A'B, and AB. A minterm is also known as a "canonical product term" because it represents a unique combination of input variables.

A maxterm, on the other hand, is a sum term in which each variable appears once, either in its complemented form or uncomplemented form. For example, in a two-variable system, there are four maxterms: A+B, A'+B, A+B', and A'+B'. A maxterm is also known as a "canonical sum term" because it represents a unique combination of input variables.

Both minterms and maxterms are used in the process of simplifying Boolean expressions using the laws of Boolean algebra. Minterms and maxterms are related to each other through a duality principle, which states that a minterm of a Boolean function is equal to the complement of the corresponding maxterm, and vice versa. This duality principle allows for the simplification of Boolean expressions using either minterms or maxterms, depending on which is easier or more convenient to use.

Class Activity

1. Assume three variables are A, B, and C.
2. List all minterms. How many are there?
3. List all maxterms. How many are there?
4. Once you are done, discuss your result with your team members.

6

4.3.2 Designations

Minterms and maxterms can be designated using different notations and symbols depending on the context and the specific application. The designation of minterms and maxterms is an important step in the simplification of Boolean expressions and their representation in digital logic circuits.

Minterms are usually designated using a subscripted binary number that represents the inputs for which the minterm evaluates to true.

6: For n variables,

- The total number of minterms is
- The total number of maxterms is

Example 4.3.1 In a three-variable system, the minterm A'B'C' would be designated as m_0, and the minterm ABC would be designated as m_7, because they correspond to the binary numbers 000 and 111, respectively.

Maxterms, on the other hand, are designated using a subscripted binary number that represents the inputs for which the maxterm evaluates to false. The maxterm A+B+C would be designated as M_0, and the maxterm A'+B'+C' would be designated as M_7, because they correspond to the binary numbers 000 and 111, respectively.

All minterm and maxterm designations for three variables are listed in Table 4.1:

Inputs			Minterms		Maxterms	
A	B	C	minterm	designation	maxterm	designation
0	0	0	A'B'C'	m_0	A+B+C	M_0
0	0	1	A'B'C	m_1	A+B+C'	M_1
0	1	0	A'BC'	m_2	A+B'+C	M_2
0	1	1	A'BC	m_3	A+B'+C'	M_3
1	0	0	AB'C'	m_4	A'+B+C	M_4
1	0	1	AB'C	m_5	A'+B+C'	M_5
1	1	0	ABC'	m_6	A'+B'+C	M_6
1	1	1	ABC	m_7	A'+B'+C'	M_7

Table 4.1: Designations of Minterms and Maxterms

Class Activity

1. Examine Table 4.1 carefully.
2. If you are given an input, say 011, can you write out the corresponding minterm? minterm designation? maxterm? maxterm designation?
3. Can you see the equivalence among the input, minterm, maxterm, and their designations?
4. Test each other with your teammates.
5. What relationship is there between a pair of corresponding minterm and maxterm? For example, m_3 and M_3.

4.4 Canonical Forms

In Boolean algebra, a canonical form is a unique expression that represents a Boolean function. There are two types of canonical forms: form 1 or the sum-of-products (SOP) and form 2 or the product-of-sums (POS) forms.

4.4.1 Canonical sum-of-products (SOP) form

A Boolean function can be expressed from a truth table by forming a *minterm* for each combination of the variables that produces a *1* in the function and taking the *OR* of all those terms.

Example 4.4.1 Three Boolean functions F_1, F_2 and F_3 are given by the following truth table:

x	y	z	F_1	F_2	F_3
0	0	0	0	①	0
0	0	1	①	0	0
0	1	0	0	①	1
0	1	1	0	①	1
1	0	0	①	0	0
1	0	1	①	0	0
1	1	0	0	①	1
1	1	1	①	0	1

Then function

$$F_1 = x'y'z + xy'z' + xy'z + xyz = m_1 + m_4 + m_5 + m_7$$

and function

$$F_2 = x'y'z' + x'yz' + x'yz + xyz' = m_0 + m_2 + m_3 + m_6$$

Exercise 4.4.1 From the above truth table, express Boolean function F_3 in the SOP form.

[7]

Example 4.4.2 Express Boolean function $F = A + B'C$ as a sum of minterms.

Solution 1: we can develop a truth table from the Boolean function and then find its SOP form by following the steps in the above example.

Solution 2: However, we have a second method here.

Hint: we need to have <u>all three</u> variables in each minterm.

The first term can be expressed as (by adding B and C variables)

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

The second term needs to add A variable only:

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

So

$$F = A + B'C = ABC + ABC' + \underline{AB'C} + AB'C' + \underline{AB'C} + A'B'C$$

$$= m_1 + m_4 + m_5 + m_6 + m_7$$

[7] Given Boolean function in its SOP form, can you develop its truth table?

Be reminded that there are two terms, underlined above, of $AB'C$, or m_5, but

$$m_5 + m_5 = m_5.$$

We can write the above result in another short form:

$$F(A, B, C) = \sum(1, 4, 5, 6, 7)$$

4.4.2 Canonical product-of-sums (POS) form

A Boolean function can be expressed from a truth table by forming a *maxterm* for each combination of the variables that produces a *0* in the function and taking the *AND* of all those terms.

Example 4.4.3 Three Boolean functions F_1, F_2 and F_3 are given by the following truth table:

x	y	z	F_1	F_2	F_3
0	0	0	⓪	1	0
0	0	1	1	⓪	0
0	1	0	⓪	1	1
0	1	1	⓪	1	1
1	0	0	1	⓪	0
1	0	1	1	⓪	0
1	1	0	⓪	1	1
1	1	1	1	⓪	1

Then function

$$\begin{aligned} F_1 &= (x + y + z)(x + y' + z)(x + y' + z')(x' + y' + z) \\ &= M_0M_2M_3M_6 = \prod(0, 2, 3, 6) \end{aligned}$$

and function

$$\begin{aligned} F_2 &= (x + y + z')(x' + y + z)(x' + y + z')(x' + y' + z') \\ &= M_1M_4M_5M_7 = \prod(1, 4, 5, 7) \end{aligned}$$

Exercise 4.4.2 From the above truth table, express Boolean function F_3 in the POS form.

[8]

Example 4.4.4 Express Boolean function $F = xy + x'z$ as a product of maxterms.

[8] Given Boolean function in its SOP form, can you develop its truth table?

Solution 1: we can develop a truth table first and then find its POS form by following steps in the above example.

Solution 2: However, we use a second method here.

Using distributive law $x + yz = (x + y)(x + z)$ repeatedly,

$$F = xy + x'z = (xy + x')(xy + z) = \cancelto{1}{(x' + x)}(x' + y)(x + z)(y + z)$$
$$= (x' + y)(x + z)(y + z)$$

Adding the missing variables for each of the three terms,

$$x' + y = x' + y + \underline{zz'} = (x' + y + z)(x' + y + z')$$
$$x + z = x + z + \underline{yy'} = (x + y + z)(x + y' + z)$$
$$y + z = y + z + \underline{xx'} = (x + y + z)(x' + y + z)$$

Combing all and removing repeating terms

$$F = (x + y + z)(x' + y + z)(x + y' + z)(x' + y + z')$$

or simply

$$= M_0M_2M_4M_5 = \prod(0, 2, 4, 5)$$

4.4.3 Conversion between SOP and POS forms

From the above sections we see that SOP represents a logical expression as the sum of several product terms, while POS represents a logical expression as the product of several sum terms. They can be converted to each other easily with the help of a truth table.

In Example 4.4.1 and Example 4.4.3, the truth tables for function F_1 are identical. This means, these two F_1 functions are the same. Its SOP form is

$$F_1 = m_1 + m_4 + m_5 + m_7 = \sum(1, 4, 5, 7),$$

and its POS form is

$$F_1 = M_0M_2M_3M_6 = \prod(0, 2, 3, 6).$$

By comparing the two, we see that

$$m_1 + m_4 + m_5 + m_7 = M_0M_2M_3M_6,$$

or

$$\sum(1, 4, 5, 7) = \prod(0, 2, 3, 6).$$

This gives us a hint about conversion between SOP and POS forms.

Steps of converting SOP form to POS form:

1. Create the truth table for the function.
2. Identify the rows in the truth table where the output is 0.

3. Write a product term for each row identified in step 2, where the product term includes all of the input variables that are set to 1 in that row.
4. Write the sum of all of the product terms from step 3 to get the POS expression.

Converting POS form to SOP form follows the same idea. [9]

[9] 9: Can you list here the steps of converting SOP form to POS form?

Example 4.4.5 Convert the following Boolean function from SOP form to POS form:

$$F = x'y'z + xy'z' + xy'z + xyz$$

Solution: we write

$$F = x'y'z + xy'z' + xy'z + xyz = \sum(1,4,5,7)$$

1. Develop its truth table as

x	y	z	F
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	0
1	1	1	1

   Make sure that the rows of F with value 1 correspond to indices 1, 4, 5, and 7.
2. Find the rows of function F with value 0. These are 0, 2, 3, and 6.
3. Their maxterms should be x+y+z, x+y′+z, x+y′+z′, and x′+y′+z.
4. The POS form of the function is then

$$F = (x+y+z)(x+y'+z)(x+y'+z')(x'+y'+z) = \prod(0,2,3,6)$$

10

[10] 10: In this example, the function

$$F = \sum(1,4,5,7) = \prod(0,2,3,6)$$

Do you find anything? Can you explain the reason?

Class Activity

1. To memorize: Boolean function F can be expressed as a sum of 1-minterms or as a product of 0-maxterms.
2. To test yourself: find another example and apply the skills we learned above.
3. To discuss: help each other to make sure all in your team can apply the skills.

4.5 Standard Forms and Non Standard Forms

Canonical forms are not economical, because minterms or maxterms must contain all variables.

Standard forms are simplified versions of canonical forms. In standard forms, each term in a function can contain one or more literals. There are still two types of standard forms:

- Sum of products: for example
$$F = y' + xy + x'y'z'$$
- Product of sums: for example
$$F = x(y' + z)(x' + y + z')$$

Boolean functions with standard forms can be implemented with two-level logic, which refers to a logic design that uses at most two logic gates between any input and output.

Example 4.5.1 Boolean function

$$F = y' + x'yz' + xy$$

can be implemented with a two-level logic as in Figure 4.1.

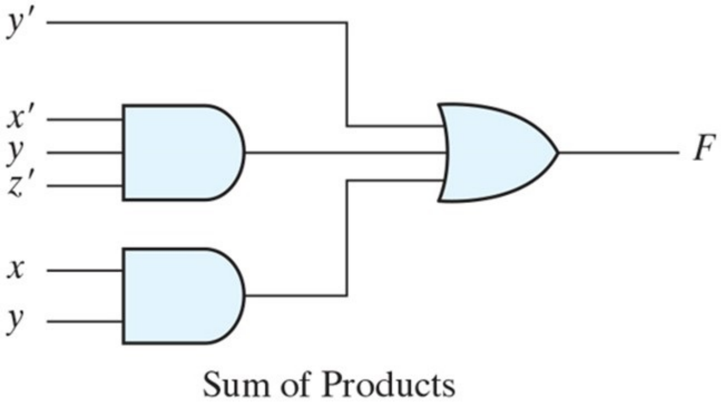


Figure 4.1: Two-Level Logic in SOP Form

Example 4.5.2 Boolean function

$$F = x(y' + z)(x' + y + z)$$

can be implemented with a two-level logic as in Figure 4.2.

Nonstandard forms are any forms that are not a sum of products nor a product of sums. For example

$$F = AB + C(D + E)$$

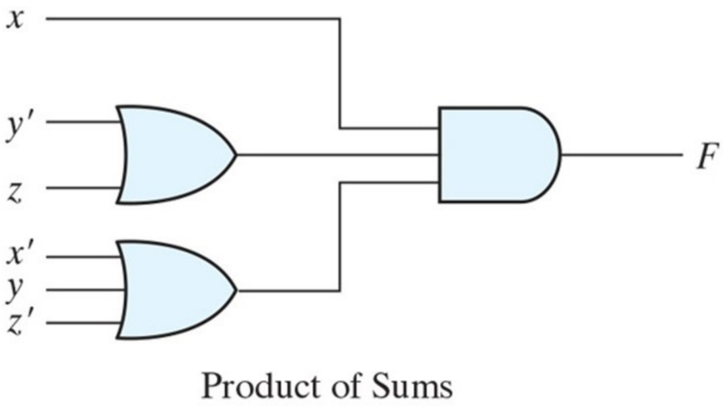


Figure 4.2: Two-Level Logic in POS Form

Nonstandard form can be converted into standard form. For example

$$F = AB + C(D + E) = AB + CD + CE$$

The non-standard form can be implemented with a multiple-level logic as in Figure 4.3.

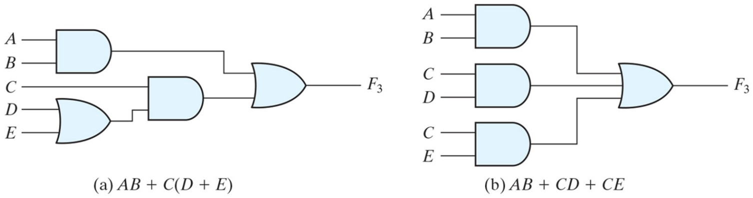


Figure 4.3: Three-Level Implementation

Exercise 4.5.1 Express Boolean function $F = A + B'C + AD$ as sum of minterms.

Exercise 4.5.2 Express Boolean function $F = x'y + xz$ as product of maxterms.

Exercise 4.5.3 Draw a two-level logic diagram to implement $F = BC' + AB + ACD$.

Advantages of Two-Level Implementation Compared to Three-Level Implementation:

1. Simplified Design: Two-level implementation reduces the complexity of the design process compared to three-level implementation, as there are fewer gates and interconnections to manage.
2. Improved Performance: Two-level implementation can result in faster and more efficient circuits than three-level implementation due to reduced propagation delays.
3. Reduced Cost: The reduction in gates and interconnections makes two-level implementation less expensive to produce and maintain than three-level implementation.

4. Flexibility: Two-level implementation can be more flexible than three-level implementation, as they can be easily modified or reconfigured to meet changing design requirements.
5. Simplified Testing: Two-level implementation is easier to test and debug than three-level implementation due to the simpler circuit design.

Disadvantages of Two-Level Implementation Compared to Three-Level Implementation:

1. Limited Expressiveness: Two-level logic circuits are limited in the types of logic functions they can implement compared to three-level logic circuits.
2. Size Limitations: Two-level implementation can become impractical for larger logic functions as the number of required gates and interconnections can quickly become unmanageable compared to three-level implementation.
3. Limited Scalability: Two-level implementation can be difficult to scale up to more complex circuits or designs compared to three-level implementation.
4. Reduced Flexibility: Three-level implementation can be more flexible than two-level implementation for some logic functions that require additional levels of logic gates.
5. Higher Power Consumption: Three-level implementation may consume more power than two-level implementation due to the additional circuitry and gates required.

Overall, the choice between two-level and three-level implementation of logic circuits depends on the specific design requirements and trade-offs between circuit complexity, performance, cost, and power consumption. Two-level implementation is often preferred for simpler designs with performance and cost considerations, while three-level implementation may be more appropriate for more complex designs with flexibility requirements.

4.6 Digital Logic Gates

In addition to the three types (AND, OR, and NOT) of digital logic gates discussed above, there are several other types, each with its own unique functionality:

- XOR gate: This gate has two inputs and produces a high output (1) if the inputs are different, and a low output (0) if the inputs are the same.
- NAND gate: This gate has two or more inputs and produces a low output (0) only if all of its inputs are high.
- NOR gate: This gate has two or more inputs and produces a low output (0) if any of its inputs are high.
- XNOR gate: This gate has two inputs and produces a high output (1) if the inputs are the same, and a low output (0) if the inputs are different.

These logic gates can be combined to form more complex circuits, such as adders, multiplexers, and flip-flops, which can perform a wide variety of functions in digital systems.

4.7 Karnaugh Maps

Gate-level minimization is to find an optimal gate-level implementation of the Boolean functions describing a digital circuit. However, we see that simplifying Boolean functions with Boolean algebra is not easy manually. Some computer-based logic synthesis tools can be used for simplification, but discussion of fundamentals in design of simple circuits helps understand complex modern design tools.

Another effective way is by using Karnaugh Maps, also known as K-Maps. K-Maps are a graphical method of simplifying Boolean algebra expressions. They are commonly used in digital electronics to simplify Boolean expressions and minimize the number of logic gates needed to implement a particular function.

K-Maps consist of a two-dimensional grid of cells, where each cell represents a unique combination of input variables. The cells are grouped together in a way that emphasizes common input patterns, such as adjacent cells that differ by only one input variable. By identifying these patterns, you can identify areas of the map that correspond to particular Boolean expressions and then combine these expressions to simplify the overall logic of the system.

K-Maps are a powerful tool for simplifying digital circuits, as they allow designers to quickly and easily visualize complex logic functions and identify areas where simplification is possible. They are often used in combination with other methods, such as Boolean algebra and truth tables, to fully optimize digital circuit design.

In K-maps,

- The order of the variables is important.
- Each cell represents a minterm or a maxterm of the Boolean function.
- Any two adjacent cells differ by only one variable: complemented in one cell and uncomplemented in the other.
- Multiple-cell areas correspond to standard terms.
- Because a function can be expressed as a sum of minterms, it is easy to see all possible expressions of a function.
- K-Map produces the simplest SOP or POS expressions (minimum expression).

4.7.1 Two-Variable K-Map

The Two-Variable K-Map consists of a grid of four cells, each representing a unique combination of two input variables. The cells are arranged in such a way that the inputs are arranged in a Gray code sequence, which ensures that adjacent cells differ by only one variable. A two-variable K-map is plotted in Figure 4.4.

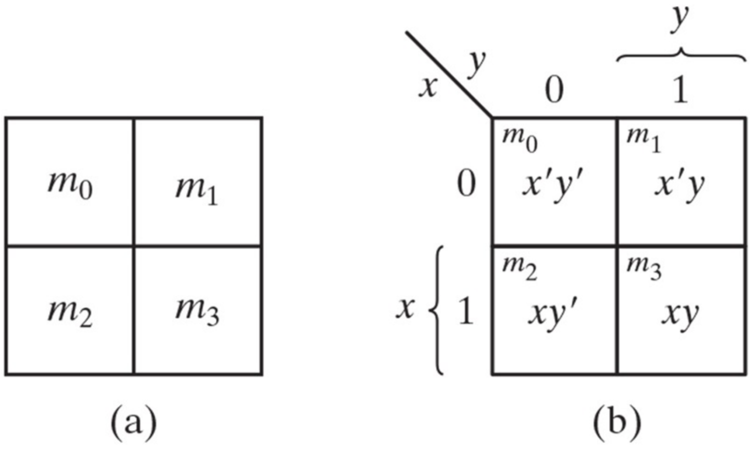


Figure 4.4: Two-Variable K-Map

To represent Boolean functions in K-Map, mark the cell (with 1) if a minterm is in the function.

Example 4.7.1 Boolean function $F = xy$ is marked in (a).

Example 4.7.2 Boolean function $F = x + y$ is marked in (b) because

$$F = m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$$

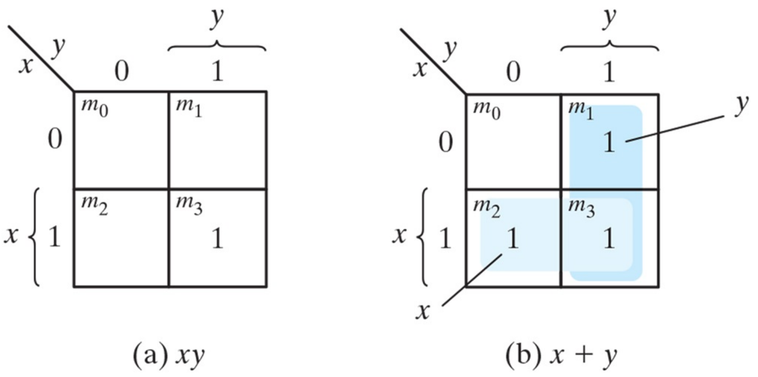


Figure 4.5: An exmaple

4.7.2 Three-Variable K-Map

A three-variable K-map is plotted in Figure 4.6.

Example 4.7.3 Simplify function with K-Map (Figure 4.7).

Solution: from the K-map, we see that the function $F(x, y, z) = \sum(2, 3, 4, 5)$.

Combine adjecent cells, it is simplified to: $F(x, y, z) = x'y + xy'$.

Example 4.7.4 Simplify function with K-Map (see Figure 4.8).

Solution:

Function $F(x, y, z) = \sum(3, 4, 6, 7)$ and simplified to $F(x, y, z) = yz + xz'$.

Example 4.7.5 Simplify function with K-Map (see Figure 4.9).

Solution:

Function $F(x, y, z) = \sum(0, 2, 4, 5, 6)$ simplified to $F(x, y, z) = z' + xy'$.

Example 4.7.6 Simplify function $F(A, B, C) = A'C + A'B + AB'C + BC$ with K-Map.

Solution:

First, implement the truth table for function $F(A, B, C) = A'C + A'B + AB'C + BC$, and then draw its K-map (see Figure 4.10. From the K-map, we see that $F(A, B, C) = \sum(1, 2, 3, 5, 7)$. After combining cells, it is simplified to: $F(A, B, C) = C + A'B$.

4.7.3 Four-Variable K-Map

A three-variable K-map is plotted in Figure 4.11.

In a four-variable Karnaugh Map,

- One cell $\Longleftrightarrow$ one minterm $\Longleftrightarrow$ a term with four literals
  eg. F=ABC'D
- Two adjacent cells $\Longleftrightarrow$ a term with three literals
  eg. F=A'BC
- Four adjacent cells $\Longleftrightarrow$ a term with two literals
  eg. F=AD
- Eight adjacent cells $\Longleftrightarrow$ a term with one literal
  eg. F=C'
- Sixteen adjacent cells $\Longleftrightarrow$ 1

Example 4.7.7 Simplify Functions with K-Map

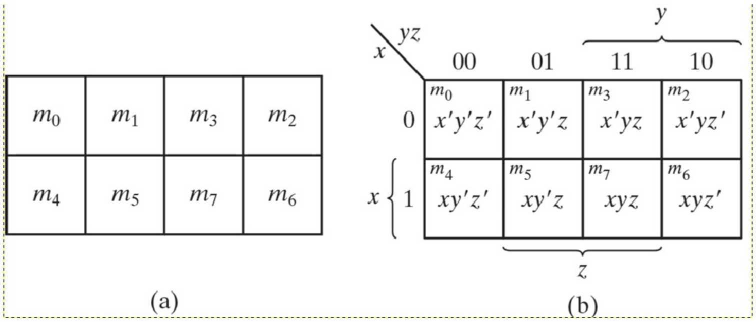


Figure 4.6: Three-Variable K-Map

Solution: Function: $F(w,x,y,z) = \sum(0,1,2,4,5,6,8,9,12,13,14)$

Simplified to: $F(w,x,y,z) = y' + w'z' + xz'$.

Example 4.7.8 Simplify Functions with K-Map

Solution: Function: $F(A,B,C,D) = A'B'C + B'CD' + A'BCD' + AB'C'$

Simplified to: $F(A,B,C,D) = B'D' + B'C' + A'CD'$.

Example 4.7.9 Product-of-Sums Simplification

Solution: Function: $F(A,B,C,D) = \sum(0,1,2,5,8,9,10)$

Simplified to: $F(A,B,C,D) = BD' + B'C + A'C'D$

While $F'(A,B,C,D) = AB + CD + BD'$ (by considering the cells with 0.)

That is $F(A,B,C,D) = (A' + B')(C' + D')(B' + D)$ (by Demorgan's).

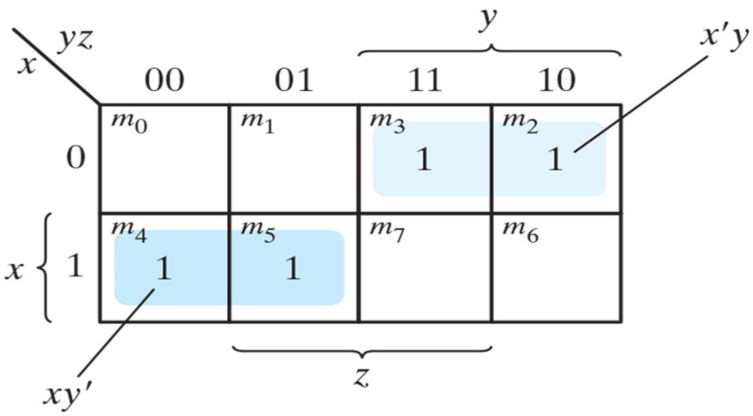


Figure 4.7: Example

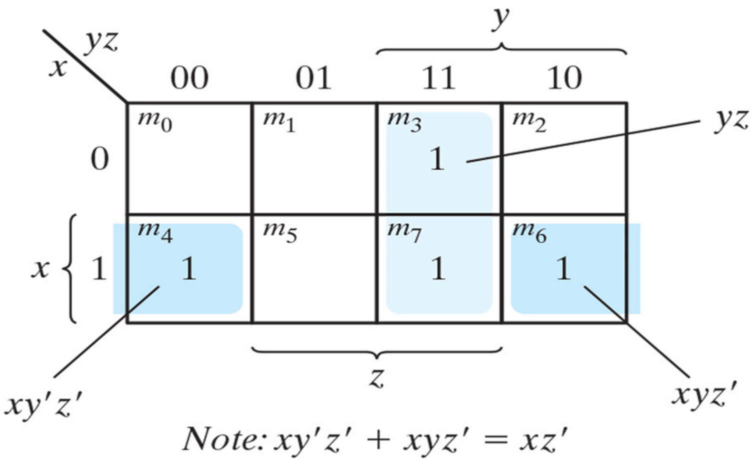


Figure 4.8: Example

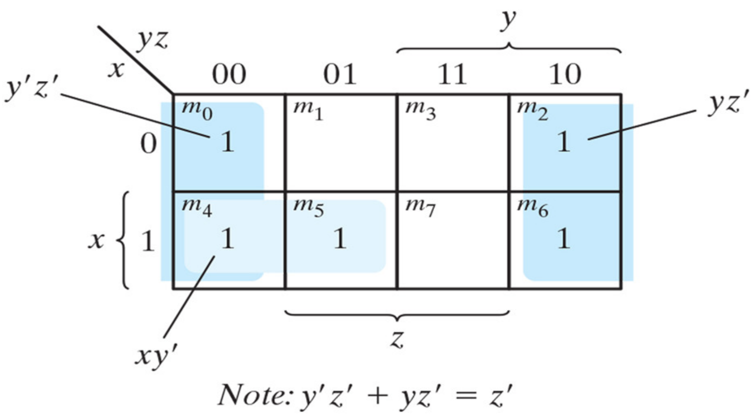


Figure 4.9: Example

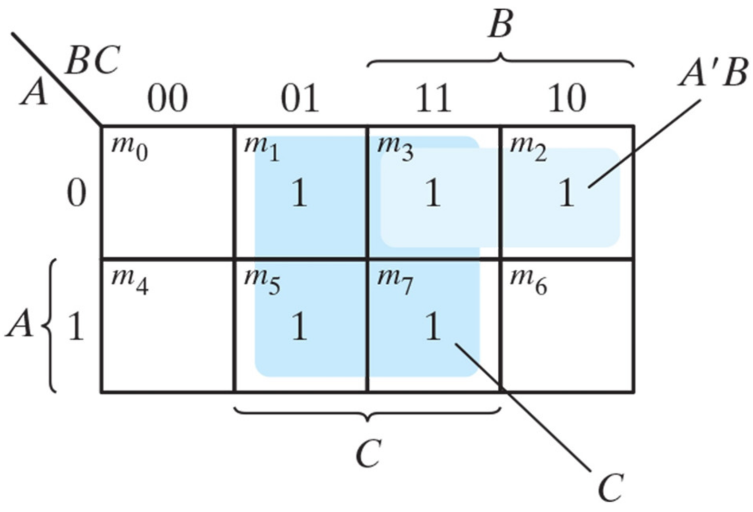


Figure 4.10: Example

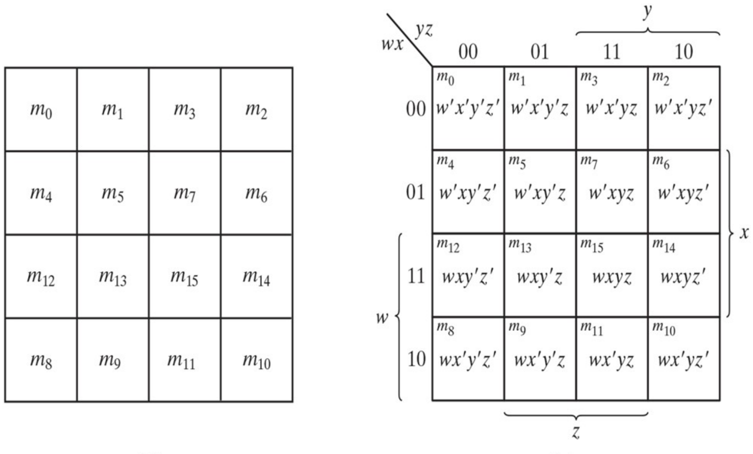
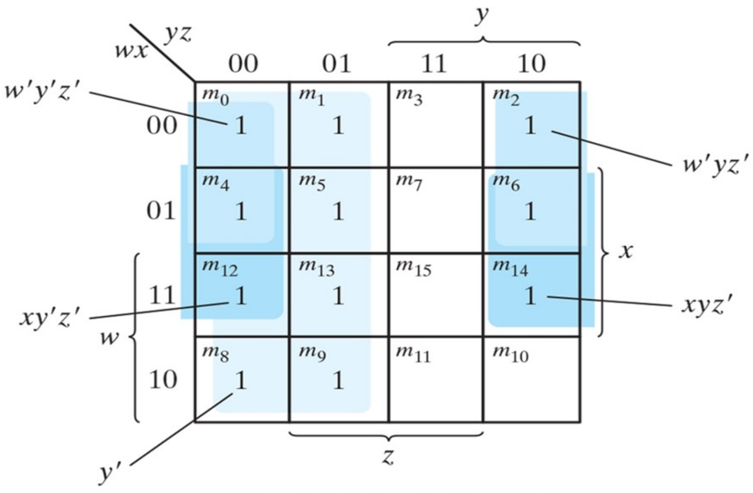

m0 m1 m3 m2
m4 m5 m7 m6
m12 m13 m15 m14
m8 m9 m11 m10
yz
wx
00 01 11 10
y
x
w
z
00 m0 w'x'y'z' m1 w'x'y'z m3 w'x'yz m2 w'x'yz'
01 m4 w'xy'z' m5 w'xy'z m7 w'xyz m6 w'xyz'
11 m12 wxy'z' m13 wxy'z m15 wxyz m14 wxyz'
10 m8 wx'y'z' m9 wx'y'z m11 wx'yz m10 wx'yz'

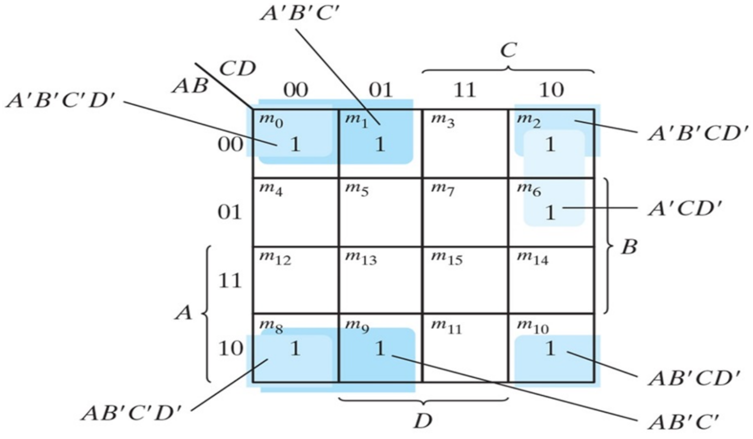
Figure 4.11: Four-Variable K-Map

(a) (b)



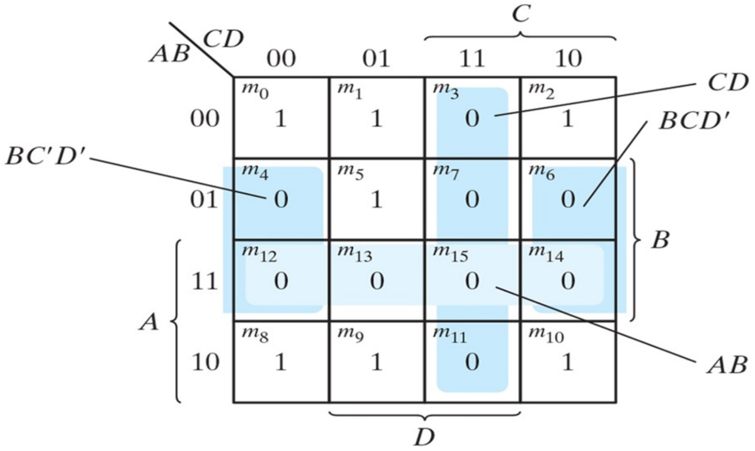
Note: $w'y'z' + w'yz' = w'z'$
$xy'z' + xyz' = xz'$

Figure 4.12: Example



Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$
$A'B'C' + AB'C' = B'C'$

Figure 4.13: Example



Note: $BC'D' + BCD' = BD'$

Figure 4.14: Example

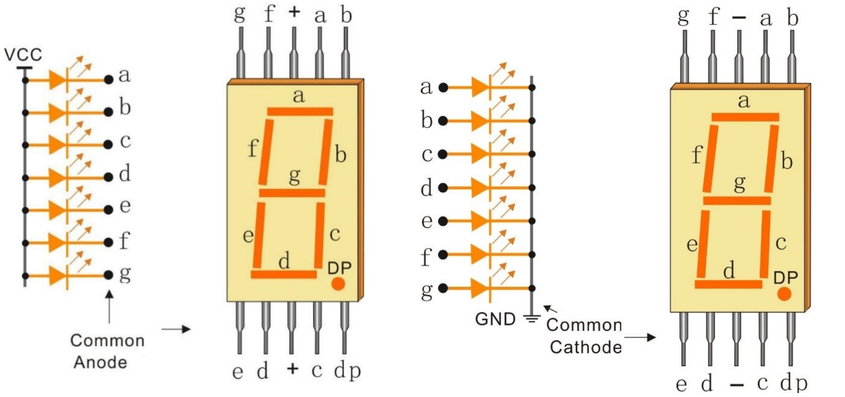
4.8 Design Example: a Clock with 7-Segment Display

A 7-segment display is a common electronic component used to display numerical digits and some basic alphanumeric characters. It consists of seven individual LED (Light Emitting Diode) or LCD (Liquid Crystal Display) segments arranged in a specific pattern to form each digit from 0 to 9 and a few additional characters like A, B, C, etc. The segments are typically labeled as a, b, c, d, e, f, and g.

7-segment displays are commonly used in a wide range of electronic devices like digital clocks, microwave ovens, calculators, and various digital indicators. They are simple, versatile, and easy to interface with microcontrollers and other electronic circuits, making them a popular choice for displaying numerical information.

There are several types of 7-segment displays, primarily categorized based on their technology, design, and purpose. The two most fundamental types are common anode and common cathode.

1. Common Anode (CA) 7-Segment Display: In this type, all the anodes of the seven segments are connected together and share a common positive voltage supply, while each segment is controlled by grounding its respective cathode. When a cathode is grounded, the corresponding segment lights up.
2. Common Cathode (CC) 7-Segment Display: opposite to CA type, all the cathodes of the segments are connected together and share a common ground, while each segment is controlled by applying a positive voltage to its respective anode.



(a) Common Anode **(b)** Common Cathode

Figure 4.15: Two Types of 7-Segment Display

For a commom cathode display, the 16 hexdecimal sybmbols are in Figure 4.16.

Assume we want to implement a clock that displays with a one digit hexdecimal number.

We use a binary counter IC chip. For example, 74HC393 is a 14 pin dual binary counter, as in Figure 4.17, Each counter contains a "Clock", a "Reset" and four outputs. The first counter involves pin 1-6, the second counter uses pin 8-13.

Our objective is to design a circuit using logic gates that takes A, B, C, and D as input variables and delivers 7 output values to control a common cathode 7-segment display pins a, b, c , d , e, f, and g. Below is our design process.

1. For all possible displayed symbols (1, 2, 3, ... E, F), we use the 4 bits ($Q_1, Q_2, Q3, Q_4$ in the IC chip) as the input variables A, B, C, and D of our Boolean functions for pins a, b, c, d, e, f, and g.
2. Implement a truth table. By examining what symbols light up pin a, we determine the truth table for output funtion for pin a. We see that for symbols 0, 2, 3, 5, 6, 7, 8, 9, A, C, E, and F, the segment a needs to be lit. Do the same with all other pins.
   The truth talbe is in Table 4.2.

Table 4.2: Truth Table for the Clock

number	A	B	C	D	a	b	c	d	e	f	g	symbol
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9
10	1	0	1	0	1	1	1	0	1	1	1	A
11	1	0	1	1	0	0	1	1	1	1	1	B
12	1	1	0	0	1	0	0	1	1	1	0	C
13	1	1	0	1	0	1	1	1	1	0	1	D
14	1	1	1	0	1	0	0	1	1	1	1	E
15	1	1	1	1	1	0	0	0	1	1	1	F

3. Determine the Boolean functions for all pins. For example, by reading the “a" column in the table, we see that the function for pin a is

$$\begin{aligned} a &= A'B'C'D' + A'B'CD' + A'B'CD + A'BC'D \\ &+ A'BCD' + A'BCD + AB'C'D' + AB'C'D \\ &+ AB'CD' + ABC'D' + ABCD' + ABCD \end{aligned}$$

   or simply in designations,

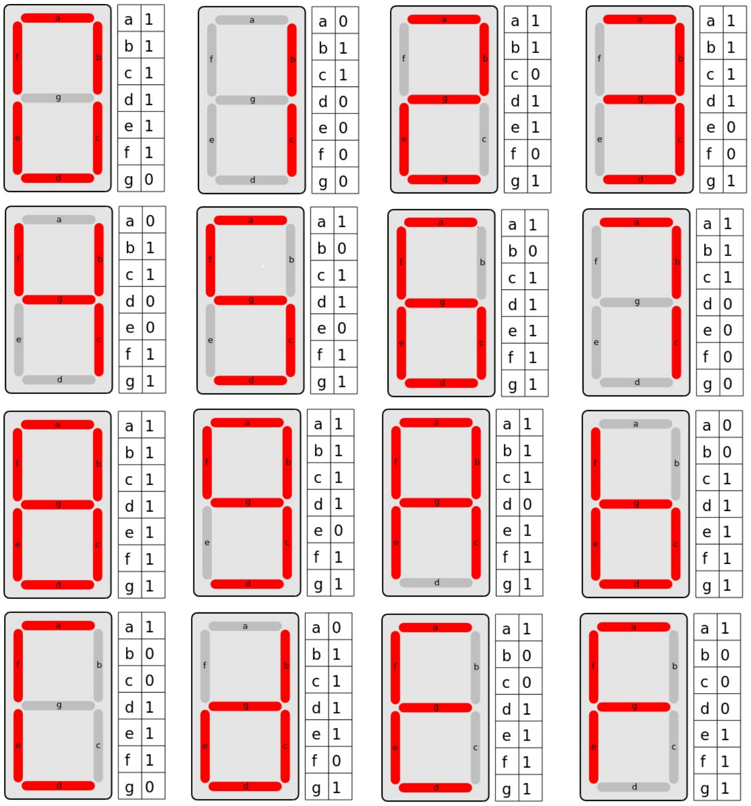
$$a = m_0 + m_2 + m_3 + m_5 + m_6 + m_7 + m_8 + m_9 + m_{10} + m_{12} + m_{14} + m_{15}.$$

   Determine the Boolean functions for other pins.
4. Simplfy all Boolean functions. For examle, we use K-map for function a as in Figure 4.18.
   The simplified function is then

$$a = A'C + BC + AD' + B'D'AB'C' + A'BD.$$

Simplify Boolean functions for other pins.

5. Use logic gates to implement all functions.



a	1	a	0	a	1	a	1
b	1	b	1	b	1	b	1
c	1	c	1	c	0	c	1
d	1	d	0	d	1	d	1
e	1	e	0	e	1	e	0
f	1	f	0	f	0	f	0
g	0	g	0	g	1	g	1

a	0	a	1	a	1	a	1
b	1	b	0	b	0	b	1
c	1	c	1	c	1	c	1
d	0	d	1	d	1	d	0
e	0	e	0	e	1	e	0
f	1	f	1	f	1	f	0
g	1	g	1	g	1	g	0

a	1	a	1	a	1	a	0
b	1	b	1	b	1	b	0
c	1	c	1	c	1	c	1
d	1	d	1	d	0	d	1
e	1	e	0	e	1	e	1
f	1	f	1	f	1	f	1
g	1	g	1	g	1	g	1

a	1	a	0	a	1	a	1
b	0	b	1	b	0	b	0
c	0	c	1	c	0	c	0
d	1	d	1	d	1	d	0
e	1	e	1	e	1	e	1
f	1	f	0	f	1	f	1
g	0	g	1	g	1	g	1

Figure 4.16: Hex Symbols

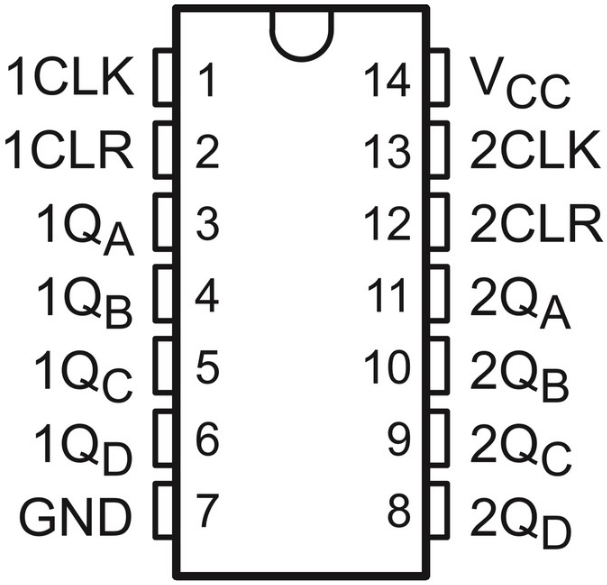
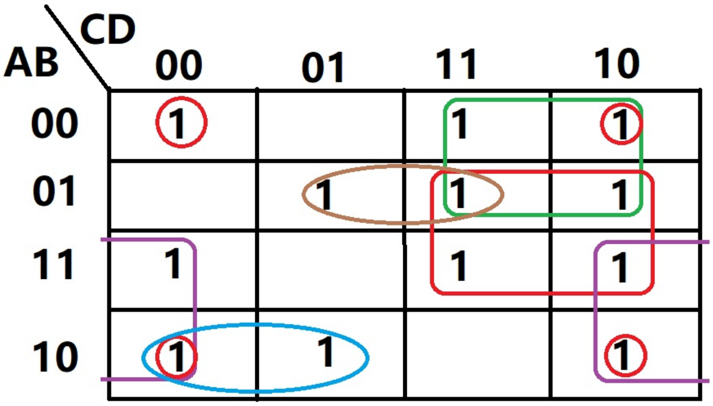


Figure 4.17: 74HC393



AB \ CD	00	01	11	10
00	1		1	1
01		1	1	1
11	1		1	1
10	1	1		1

Figure 4.18: K-map for Pin a

5 Hardware

A computer system is a complex and interconnected collection of components that work together to process data, perform tasks, and provide value to users.

Computer hardware refers to the physical components of a computer system that enable it to function. For a typical computer, its hardware includes the following physical parts: the case, central processing unit (CPU), random access memory (RAM), graphics card (GPU), computer data storage (hard disk), monitor, mouse, keyboard, sound card, speakers, and motherboard.

5.1 Computer Systems

Computer systems in computer engineering can be classified based on many different criteria. For example, based on size, computers can be classified into super computers[6, 7], mainframe computers[8], mini computers[9], and micro computers[10]. Based on functionality, computers can be classified into servers, workstations, information devices, and embedded computers.

Embedded vs. Standalone Systems

Embedded systems and standalone systems are two different types of computing systems that are designed for different purposes. Here's a brief overview of the differences between them:

Embedded Systems [1] Embedded systems are specialized computer systems that are designed to perform a specific task or function. They are typically found in devices such as cars, medical equipment, and

[6]: Hosch (2023), *supercomputer*

[7]: Fernandez et al. (2019), 'Supercomputers to improve the performance in higher education: A review of the literature'

[8]: IBM (2023), *What is a mainframe?*

[9]: Wikipedia contributors (2023), *Minicomputer — Wikipedia, The Free Encyclopedia*

[10]: Wikipedia contributors (2023), *Microcomputer — Wikipedia, The Free Encyclopedia*

[1] List and describe two examples of embedded systems.

home appliances. Embedded systems are usually small, low-power devices that are designed to be integrated into a larger system or product.

Embedded systems often have dedicated hardware and software designed specifically for their task. They may not have a user interface or require user interaction, and may not be easily upgradeable or modifiable. Examples of embedded systems include GPS systems, industrial control systems, and smart home devices.

Standalone Systems [2] Standalone systems, on the other hand, are general-purpose computing systems that are designed to perform a wide variety of tasks. They are typically larger and more powerful than embedded systems and are designed to be used by a user. Standalone systems usually have a graphical user interface (GUI) and can run a variety of software applications.

Standalone systems can be customized or upgraded by the user, and can be used for a wide variety of purposes such as office productivity, gaming, or multimedia. Examples of standalone systems include desktop computers, laptops, and tablets.

In summary, embedded systems are specialized, task-specific computing systems that are designed to be integrated into a larger system or product, while standalone systems are general-purpose computing systems designed for use by a user.

Real-Time vs. Non-Real-Time Systems

Real-time and non-real-time systems are two different types of computing systems that are designed for different purposes. Here's a brief overview of the differences between them:

Real-Time Systems [3] Real-time systems[11] are computing systems that must respond to events or stimuli within a specified time frame. These systems are designed to respond to input or events in real-time or near real-time, meaning they must process data and provide a response within a specific time limit.

Real-time systems are often used in applications where timing is critical, such as aviation, medical devices, and industrial automation. Examples of real-time systems include air traffic control systems, pacemakers, and robotic manufacturing systems.

Non-Real-Time Systems [4] Non-real-time systems[12] are computing systems that do not have specific timing requirements. These systems are designed to perform tasks without any specific time constraints, and can take as long as necessary to complete a task.

Non-real-time systems are often used in applications where timing is not critical, such as office productivity, gaming, and multimedia. Examples of non-real-time systems include desktop computers, laptops, and mobile devices.

In summary, real-time systems are designed to respond to input or events within a specific time frame, while non-real-time systems do not have

[2] List and describe two examples of standalone systems.

[3] List and describe two examples of real-time systems.

[11]: Intel (2023), *Real-Time Systems Overview and Examples*

[4] List and describe two examples of non-real-time systems.

[12]: Dalkiran et al. (2021), 'Automated integration of real-time and non-real-time defense systems'

specific timing requirements and can take as long as necessary to complete a task.

Control Systems vs. Information Systems

Control systems and information systems are two different types of computing systems that are designed for different purposes. Here's a brief overview of the differences between them:

Control Systems [5] Control systems[13] are computing systems that are designed to manage and control physical processes or systems. They use sensors and actuators to measure and manipulate physical variables, such as temperature, pressure, or speed, to achieve a desired outcome.

Control systems are often used in industrial automation, robotics, and manufacturing applications. Examples of control systems include HVAC (heating, ventilation, and air conditioning) systems, process control systems, and flight control systems.

Information Systems [6] Information systems[14] are computing systems that are designed to manage and process data and information. They are used to collect, store, process, and retrieve data and information, and to support decision-making and business operations.

Information systems are often used in business, government, and educational applications. Examples of information systems include customer relationship management (CRM) systems, human resource information systems (HRIS), and financial management systems.

In summary, control systems are designed to manage and control physical processes or systems, while information systems are designed to manage and process data and information.

Distributed vs. Centralized Systems

Distributed and centralized systems are two different types of computing systems that are designed for different purposes. Here's a brief overview of the differences between them:

Centralized Systems [7] Centralized systems[15] are computing systems that are designed to have a single point of control or authority. In a centralized system, all processing and decision-making occurs at a central location, and all communication between components occurs through this central point.

Centralized systems are often used in situations where there is a clear hierarchy of control, such as in a company or government organization. Examples of centralized systems include mainframe computers, centralized databases, and client-server architectures.

Distributed Systems [8] Distributed systems[16] are computing systems that are designed to distribute processing and decision-making across multiple locations or components. In a distributed system, there is no

[5] List and describe two examples of control systems.

[13]: Wikipedia contributors (2023), *Control system — Wikipedia, The Free Encyclopedia*

[6] List and describe two examples of information systems.

[14]: Wikipedia contributors (2023), *Information system — Wikipedia, The Free Encyclopedia*

[7] List and describe two examples of centralized systems.

[15]: Veetil (2017), *Coordination in Centralized and Decentralized Systems*

[8] List and describe two examples of distributed systems.

[16]: van Steen et al. (2017), *Distributed Systems*

central point of control or authority, and components communicate and collaborate with each other to achieve a desired outcome.
Distributed systems are often used in situations where there is a need for flexibility, scalability, and fault-tolerance. Examples of distributed systems include peer-to-peer networks, distributed databases, and cloud computing.

In summary, centralized systems are designed to have a single point of control or authority, while distributed systems are designed to distribute processing and decision-making across multiple locations or components.

Open vs. Closed Systems

Open and closed systems are two different types of computing systems that are designed for different purposes. Here's a brief overview of the differences between them:

Open Systems [9] Open systems[17] are computing systems that are designed to be interoperable with other systems and components. In an open system, components can be added, removed, or replaced with little or no impact on the overall system. Open systems are often designed to adhere to industry standards and specifications, and use open protocols and interfaces.
Open systems are often used in situations where there is a need for interoperability, flexibility, and vendor neutrality. Examples of open systems include the Internet, Linux operating system, and open-source software.

Closed Systems [10] Closed systems are computing systems that are designed to be self-contained and tightly controlled. In a closed system, components are typically proprietary and cannot be easily added, removed, or replaced. Closed systems often use proprietary protocols and interfaces, and may be designed to prevent or limit interoperability with other systems.
Closed systems are often used in situations where there is a need for security, control, and stability. Examples of closed systems include Apple's iOS operating system, some gaming consoles, and some industrial control systems.

In summary, open systems are designed to be interoperable with other systems and components, while closed systems are designed to be self-contained and tightly controlled.

For each type of system, the hardware can be classified differently.

Select one system classification, list and explain the hardware components.

5.2 Hardware in Computer Architecture

Engineers in computer architecture will design and develop computer systems and components, including processors, memory, and input/output systems.

[9] List and describe two examples of open systems.

[17]: Wikipedia contributors (2021), *Open system (computing) — Wikipedia, The Free Encyclopedia*

[10] List and describe two examples of closed systems.

Central Processing Unit (CPU) The CPU is the "brain" of the computer that performs arithmetic and logical operations on data. Further readings can be found at [18].

Random Access Memory (RAM) RAM is a type of memory that stores data and instructions temporarily while the computer is running. Further readings can be found at [19].

Hard Disk Drive (HDD) or Solid State Drive (SSD) HDDs and SSDs are storage devices that store data on a permanent basis.

Motherboard The motherboard is the main circuit board of a computer that connects all of the other components together.

Power Supply Unit (PSU) The PSU is responsible for supplying power to all of the other components in a computer.

Graphics Processing Unit (GPU) The GPU is a specialized processor that is responsible for rendering graphics and accelerating video playback.

Input/Output Devices These include devices such as the keyboard, mouse, monitor, printer, sound card, and video card, which allow the user to input information into the computer and receive output.

Central Processing Unit

The Central Processing Unit (CPU) is responsible for executing instructions and controlling the operations of the computer's hardware and software.

Here's a simplified explanation of how a CPU works:

- Fetch: The CPU retrieves an instruction from memory, which contains the operation to be performed and the data on which to perform it.
- Decode: The CPU decodes the instruction and determines what operation to perform and what data to use.
- Execute: The CPU performs the operation, such as adding two numbers, and stores the result in a register.
- Repeat: The CPU repeats this cycle for each instruction in the program.

The block diagram of a basic uniprocessor-CPU computer is illustratd in Figure 5.1, where black lines indicate data flow; red lines indicate control flow; and arrows indicate flow directions [18].

The CPU has several components that work together to execute instructions:

- Control Unit: The control unit manages the flow of instructions and data between the CPU and other components, such as memory and input/output devices.
- Arithmetic Logic Unit (ALU)[20]: The ALU performs arithmetic and logic operations, such as addition, subtraction, multiplication, and division. It also performs logical operations, such as AND, OR, and NOT. It receives data from registers within the CPU, processes it based on the instructions from the control unit, and produces the result.
- Registers[21]: The registers are high-speed storage areas within the CPU that hold data and instructions during processing. Registers are faster than other memory levels like RAM or cache memory. The types

[18]: Wikipedia contributors (2023), *Central processing unit — Wikipedia, The Free Encyclopedia*

[19]: Wikipedia contributors (2023), *Random-access memory — Wikipedia, The Free Encyclopedia*

[18]: Wikipedia contributors (2023), *Central processing unit — Wikipedia, The Free Encyclopedia*

[20]: Wikipedia contributors (2023), *Arithmetic logic unit — Wikipedia, The Free Encyclopedia*

[21]: Wikipedia contributors (2023), *Processor register — Wikipedia, The Free Encyclopedia*

of registers include general-purpose registers that hold operational data, instruction registers that hold the current instruction being processed, and program counter that holds the memory address of the next instruction to be fetched.

- Cache[22]: The cache is a small amount of high-speed memory that stores frequently used data and instructions to speed up processing. It improves computer performance by minimizing the frequency of slower main memory accesses and reduceing latency for data access. There are three levels of Cache. L1 Cache is the smallest and fastest, located directly on the CPU chip. It typically stores the most frequently used data. L2 Cache is larger than L1 but slightly slower. It can be on the CPU chip or close to it. L3 Cache is even larger and slower, usually shared among multiple CPU cores.
- Bus Interface Unit (BIU): The BIU is responsible for communicating with other devices in the computer, such as memory and input/output devices. More details can be found in [23].
- Clock: The Clock is a component that synchronizes the operations of the CPU. It provides a regular signal that controls the timing of the CPU's operations.

[22]: Wikipedia contributors (2023), *Cache (computing) — Wikipedia, The Free Encyclopedia*

[23]: Rushby (2001), *A Comparison of Bus Architectures for Safety-Critical Embedded Systems*

The performance of a CPU is determined by its clock speed, which is the rate at which it executes instructions, and the number of cores, which allow it to perform multiple tasks simultaneously. CPU design is an ongoing area of research and development, with new technologies such as multicore processors, hyper-threading, and cache hierarchies being developed to improve performance and efficiency.

Random Access Memory

Random Access Memory, commonly known as RAM, is a type of computer memory that allows data to be accessed in any order, without having to read and discard the data that comes before it. RAM is a volatile memory,

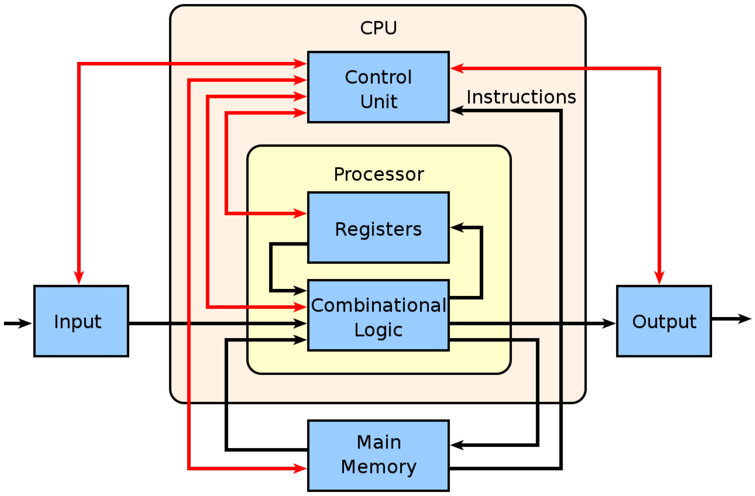


Figure 5.1: Block diagram of a basic CPU

which means that it loses its contents when the power to the computer is turned off.

RAM is an important component of a computer, as it provides a fast and efficient way for the computer to access data that it needs to run programs and operate the operating system. The amount of RAM in a computer affects its performance, as more RAM allows the computer to run more programs simultaneously and access data faster.

There are different types of RAM, including dynamic random access memory (DRAM) and static random access memory (SRAM).

DRAM is the most common type of RAM used in computers, and it is less expensive to produce and therefore is used for main memory. DRAM stores a bit of data using a transistor and capacitor pair (typically a MOSFET and MOS capacitor, respectively),which together comprise a DRAM cell, as in Figure 5.2. The capacitor holds a high or low charge (1 or 0, respectively), and the transistor acts as a switch that lets the control circuitry on the chip read the capacitor's state of charge or change it.

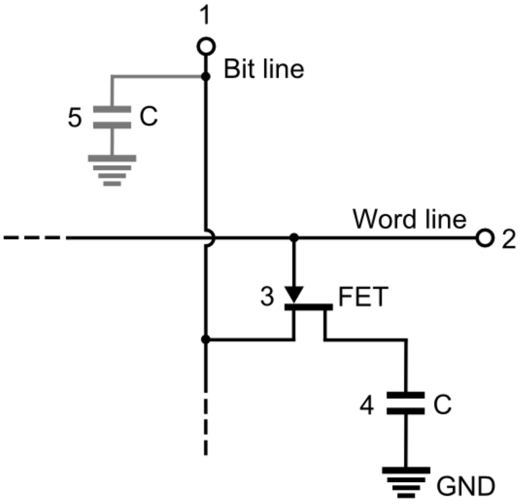


Figure 5.2: DRAM cell

In SRAM, a bit of data is stored using the state of a six-transistor memory cell, typically using six MOSFETs, as in Figure 5.3. SRAM is more expensive to produce, but is generally faster and requires less dynamic power than DRAM. SRAM is used for cache memory, which provides faster access to data that is frequently used by the processor [19].

[19]: Wikipedia contributors (2023), *Random-access memory — Wikipedia, The Free Encyclopedia*

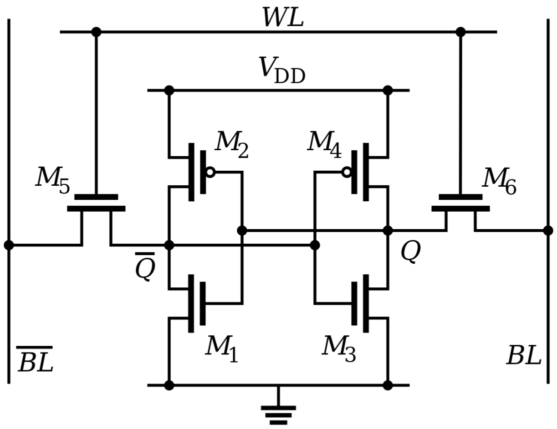


Figure 5.3: SRAM cell

5.3 Hardware in Computer Networks and Cyber Security

Computer networks are composed of hardware devices that enable the communication and transfer of data between different devices connected to the network.

A basic firewall network diagram is as in Figure 5.4.

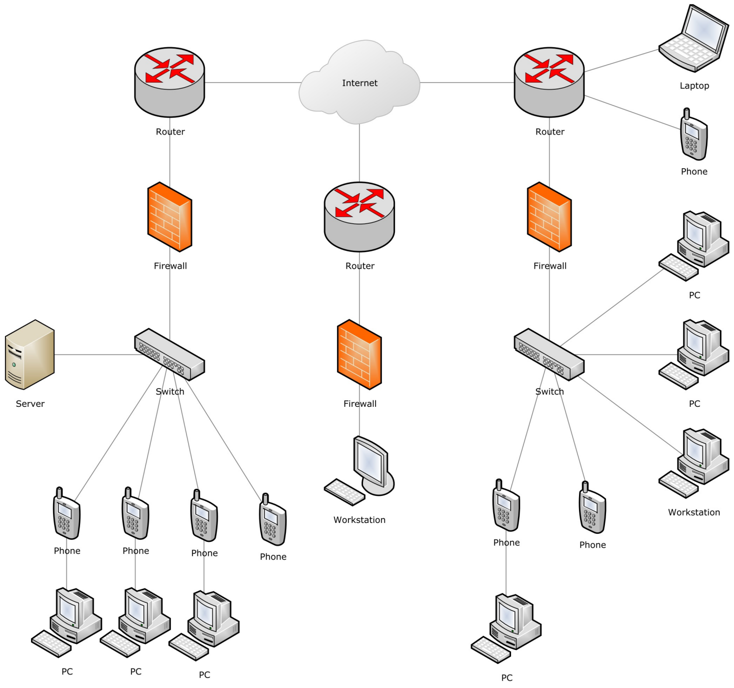


Figure 5.4: A Firewall Network Diagram

Network Interface Cards (NICs) NICs are used to connect devices to the network. They enable communication between the device and the network by translating the electrical signals from the device into a format that can be transmitted over the network. NICs can be wired (Ethernet) or wireless (Wi-Fi).

Switches Switches are used to connect multiple devices to the network. They enable communication between devices by directing network traffic between them. Switches can be either managed or unmanaged, and they vary in the number of ports they have.

Routers Routers are used to connect networks together. They enable communication between devices on different networks by directing network traffic between them. Routers use protocols such as IP and ARP to route data between networks.

Modems Modems are used to connect a computer network to the internet. They convert digital signals from the computer network into analog signals that can be transmitted over phone lines or cable lines.

Firewalls Firewalls are used to protect computer networks from unauthorized access. They act as a barrier between the network and the internet and control the flow of network traffic to and from the network. Firewalls can be either software or hardware-based.

Network Attached Storage (NAS) NAS devices are used to store data on the network. They provide a centralized storage location for files and can be accessed by multiple devices on the network.

Hubs Hubs are used to connect multiple devices to the network. They function as a repeater, transmitting network signals to all devices connected to the hub. Hubs are less common now as switches have largely replaced them.

Repeaters Repeaters are used to extend the reach of the network by boosting network signals. They receive weak signals and amplify them before retransmitting them to the network.

These hardware components work together to create a computer network that enables communication and data transfer between devices. They are connected to each other using cables or wireless signals and are controlled by software that manages network traffic and enables communication between devices.

In addition to the hardware components commonly found in computer networks, there are also specific hardware components used for cyber security purposes. Here are some examples of hardware used in computer networks and cyber security:

Intrusion Detection Systems (IDS) IDS hardware is used to monitor network traffic for potential threats. IDS hardware can be placed at the network edge or on individual devices and can detect and alert network administrators to potential attacks.

Intrusion Prevention Systems (IPS) IPS hardware is similar to IDS hardware, but it also has the ability to block or stop potential threats. IPS hardware can be placed at the network edge or on individual devices and can automatically block malicious traffic.

Firewall Appliances Firewall appliances are hardware devices that act as a barrier between the network and the internet. They monitor and control network traffic and can block unauthorized access to the network.

VPN Concentrators VPN concentrators are hardware devices used to manage Virtual Private Networks (VPNs). They encrypt and decrypt network traffic between remote devices and the network, providing a secure connection.

Secure Routers Secure routers are hardware devices that are designed with built-in security features. They can encrypt network traffic and have the ability to block unauthorized access to the network.

Encryption Devices Encryption devices are hardware devices that encrypt and decrypt network traffic. They are used to protect sensitive data that is transmitted over the network.

Two-Factor Authentication Devices Two-factor authentication devices are hardware devices used to provide an additional layer of security to network logins. They require a user to provide a password and a physical token such as a USB key or smart card.

These hardware components work together to provide cyber security for computer networks. They can be used in combination with software-based security measures such as anti-virus software and access controls to create a comprehensive cyber security strategy.

Firewall

One of the most important hardware in computer networks and cyber security is firewall.

Firewalls work by monitoring and controlling the traffic that flows between an organization's internal network and the public internet. The firewall can be implemented as hardware, software, or a combination of both.

Here's a simplified overview of how a firewall typically works:

- Packet inspection: When data is sent over a network, it's broken down into smaller packets. The firewall examines each packet of data to determine whether it should be allowed or blocked based on the rules that have been set.
- Filtering: Firewalls can filter traffic based on various criteria, such as the source and destination IP address, port number, and protocol type. For example, the firewall can be configured to block traffic from specific IP addresses or to only allow traffic on specific ports.
- Access control: Firewalls can also be used to control access to certain resources within the network. For example, the firewall can be configured to allow employees to access certain websites but block access to social media sites.
- Logging and reporting: Firewalls can log all traffic that passes through them, providing administrators with a record of all network activity. This information can be used to identify security threats and investigate any potential security incidents.

Firewalls are an important part of network security and are used to protect against various types of cyber attacks, such as malware, viruses, and unauthorized access.

Router

Another critical component of computer networks is router. It connects multiple networks together and directs traffic between them. It determines the most efficient path for data to travel and can filter and block certain types of traffic based on specific rules. Routers are also used in conjunction with VPNs to provide secure remote access to a network.

Here's a simplified overview of how a router works:

- Packet forwarding: When a packet of data is received by the router, the router examines the destination IP address in the packet header to determine where the packet should be forwarded. If the destination is within the same network, the router forwards the packet to the appropriate device. If the destination is in a different network, the router forwards the packet to the next router in the path towards the destination.

- Routing table: Routers use a routing table to determine the best path for data to travel. The routing table contains information about the network topology, including the IP addresses of other routers and the networks they connect to. Routers use this information to determine the most efficient path for data to travel.
- Routing protocols: Routers use routing protocols to exchange information with other routers and update their routing tables. There are several different routing protocols, including OSPF, BGP, and RIP.
- Security: Routers can also be used to enhance network security. For example, routers can be configured to filter traffic based on certain criteria, such as source IP address, destination IP address, or protocol type. Routers can also be used to implement virtual private networks (VPNs), which encrypt traffic and provide secure remote access to a network.

Routers are an essential component of computer networks and are used to direct data between networks, optimize network performance, and enhance network security.

Intrusion Detection Systems (IDS)

An Intrusion Detection System (IDS) is a security technology that monitors network traffic for signs of malicious activity or policy violations. IDS can be deployed as hardware or software, and can be implemented as a network-based or host-based system.

The primary function of an IDS is to detect potential security breaches by analyzing traffic for indicators of an attack. IDS works by analyzing network traffic and comparing it to a set of rules or signatures. If the traffic matches a known attack signature or pattern, the IDS generates an alert.

Here are some common types of IDS:

- Network-based IDS (NIDS): A network-based IDS monitors network traffic for signs of attacks. It analyzes packets as they pass through the network and can detect attacks that originate from outside the network.
- Host-based IDS (HIDS): A host-based IDS monitors activity on a specific host or server. It analyzes logs and system events to detect potential intrusions or policy violations.
- Signature-based IDS: A signature-based IDS uses a database of known attack signatures to detect malicious activity. It compares network traffic against this database and generates alerts if a match is found.
- Behavioral-based IDS: A behavioral-based IDS analyzes network traffic and system activity to establish a baseline of normal behavior. It then alerts administrators if it detects activity that deviates from the established baseline.

IDS is an important part of network security and can help organizations detect and respond to potential security threats. However, IDS is not foolproof and should be used in conjunction with other security technologies, such as firewalls and antivirus software, to provide comprehensive protection.

5.4 Hardware in Embedded Systems and Robotics

Hardware in embedded systems and robotics is a crucial component that enables the system to function effectively. Embedded systems are designed to perform a specific task, and the hardware is tailored to meet the requirements of that task. Here are some of the hardware components commonly found in embedded systems:

Microcontrollers Microcontrollers are small computing devices that are the heart of most embedded systems and are the brains of the robot. They are designed to handle input and output operations, process data, and control the system's peripherals. Popular microcontroller families include Arduino, Raspberry Pi, PIC, and STM32.

Sensors Sensors are used to detect and measure physical or environmental conditions such as temperature, humidity, pressure, and light. Robots use sensors to measure various physical parameters such as position, velocity, and force. These measurements are used by the system to make decisions or to adjust the system's behavior. Examples of sensors used in robotics include cameras, accelerometers, gyroscopes, and force sensors.

Actuators Actuators are used to control mechanical components of the system. Examples of actuators include motors, solenoids, and relays. Actuators are used to move things or to control the flow of electricity or other physical variables.

Communication Interfaces Communication interfaces are used to enable communication between the embedded system and other devices or systems. Examples include serial communication interfaces such as UART, SPI, and I2C, Ethernet, and wireless interfaces such as Wi-Fi and Bluetooth.

Memory Embedded systems require memory to store program code and data. Memory can be either internal, such as flash memory, or external, such as SD cards.

Power Management Embedded systems are often powered by batteries or other low-power sources, so power management is essential. This includes hardware components such as voltage regulators and power switches, as well as software techniques such as power saving modes and sleep modes.

Real-time Clock Real-time clocks are used to keep track of time in embedded systems. They are used to timestamp data, schedule tasks, and control the system's behavior over time.

Display Display components are used to provide visual feedback to the user. Examples include LED displays, LCD displays, and OLED displays.

Audio Audio components are used to provide audible feedback or to play audio files. Examples include speakers, microphones, and audio codecs.

Frame and Chassis The frame and chassis provide the structure and support for the robot. They are designed to be durable and rigid to withstand the forces and stresses of the robot's movements. The frame and chassis should also be lightweight to minimize the overall weight of the robot.

Grippers Grippers are used in robotics to grasp and manipulate objects. Grippers come in a variety of shapes and sizes and can be designed to handle different types of objects.

Wheels and Tracks Wheels and tracks are used to provide mobility to the robot. They enable the robot to move on different types of surfaces and terrain.

These hardware components are usually interconnected via a printed circuit board (PCB) and communicate with each other through a variety of protocols and interfaces.

Microcontroller

A microcontroller is a small computer on a single integrated circuit chip. It contains a processor, memory, and input/output peripherals, all on a single chip. Microcontrollers are designed to perform specific functions and are often used in embedded systems, such as appliances, automobiles, and medical devices.

Characteristics of Microcontrollers

Here are some common features and characteristics of microcontrollers:

- Low power consumption: Microcontrollers are designed to operate on low power and can run on batteries or other low-power sources.
- Real-time processing: Microcontrollers can process data in real-time, making them suitable for time-critical applications, such as control systems or robotics.
- Small size: Microcontrollers are designed to be small and compact, making them suitable for use in devices with limited space.
- Integrated peripherals: Microcontrollers contain a range of built-in peripherals, such as timers, serial ports, and analog-to-digital converters, that are essential for many applications.
- Programmability: Microcontrollers can be programmed to perform specific functions and can be reprogrammed if needed.

Microcontrollers are used in a wide range of applications, including automotive systems, home appliances, medical devices, and consumer electronics. They are often used to perform specific functions, such as controlling a motor, measuring temperature, or processing sensor data. Microcontrollers can be programmed in a variety of programming languages, including C and Assembly, and there are many development tools available to help developers create and test microcontroller-based applications.

Types of Microcontrollers

There are many different types of microcontrollers available, each with its own features and capabilities. Here are some of the most common types of microcontrollers:

- 8-bit microcontrollers: 8-bit microcontrollers are some of the simplest and most widely used microcontrollers. They are often used in simple applications, such as controlling a motor or monitoring a sensor.
- 16-bit microcontrollers: 16-bit microcontrollers are more powerful than 8-bit microcontrollers and are often used in applications that require more processing power or memory.
- 32-bit microcontrollers: 32-bit microcontrollers are even more powerful than 16-bit microcontrollers and are used in more complex applications, such as automotive systems or medical devices.
- ARM-based microcontrollers: ARM-based microcontrollers use the ARM architecture, which is widely used in mobile devices and other embedded systems. They are often used in applications that require high processing power and low power consumption.
- PIC microcontrollers: PIC microcontrollers are a family of microcontrollers developed by Microchip Technology. They are widely used in industrial control systems, automotive systems, and consumer electronics.
- AVR microcontrollers: AVR microcontrollers are a family of microcontrollers developed by Atmel. They are widely used in robotics, home automation, and other embedded systems.
- Arduino microcontrollers: Arduino microcontrollers are a type of microcontroller that is widely used in hobbyist and educational projects. They are designed to be easy to use and program, making them a popular choice for beginners.

There are many other types of microcontrollers available, each with its own strengths and weaknesses. When choosing a microcontroller, it's important to consider the specific requirements of the application and choose a microcontroller that can meet those requirements.

Sensor

A sensor is a device that detects and measures physical, chemical, or biological quantities and converts them into an electrical or digital signal. Sensors are used in a wide range of applications, from detecting temperature and humidity to monitoring the levels of pollutants in the environment.

Here are some common types of sensors:

- Temperature sensors: Temperature sensors measure the temperature of a particular object or environment. They are commonly used in thermostats, refrigerators, and industrial applications.
  One example of a temperature sensor is a thermocouple. A thermocouple is a type of temperature sensor that consists of two dissimilar metals that are joined together at one end. When the joined end is

exposed to a temperature change, it generates a small voltage that can be measured and used to calculate the temperature.

Thermocouples are widely used in industrial applications to measure temperature in harsh environments, such as in furnaces or industrial ovens. They are also used in home appliances, such as stoves and refrigerators, to regulate temperature and ensure proper operation.

Another example of a temperature sensor is a resistance temperature detector (RTD). An RTD is a type of temperature sensor that measures temperature by changing resistance as temperature changes. RTDs are commonly used in laboratory settings, as well as in industrial applications that require high accuracy and stability.

In addition to these examples, there are many other types of temperature sensors, including thermistors, infrared sensors, and bimetallic sensors. The specific type of temperature sensor used will depend on the requirements of the application, such as the temperature range, accuracy, and response time needed.

- Pressure sensors: Pressure sensors measure the pressure of a fluid or gas. They are used in a variety of applications, including monitoring tire pressure in cars, measuring blood pressure, and controlling hydraulic systems.
- Light sensors: Light sensors detect the presence and intensity of light. They are used in cameras, security systems, and automatic lighting systems.
- Motion sensors: Motion sensors detect movement in a particular area. They are used in security systems, automatic doors, and video game controllers.
- Proximity sensors: Proximity sensors detect the presence of an object or person in close proximity. They are used in parking sensors, automatic faucets, and robotics.
- Humidity sensors: Humidity sensors measure the moisture content in the air. They are used in weather monitoring, home humidifiers, and industrial applications.
- Gas sensors: Gas sensors detect the presence of gases, such as carbon monoxide, methane, or oxygen. They are used in industrial safety systems, home carbon monoxide detectors, and environmental monitoring.

Sensors are essential components in many modern technologies and are used to collect data and enable automation. The data collected by sensors can be analyzed and used to make informed decisions, such as adjusting temperature settings, detecting security threats, or monitoring environmental conditions.

6 Software

In computer engineering, software refers to the programs, applications, and systems that run on computers and other digital devices. These programs can be developed to perform a wide range of tasks, from basic functions like word processing and web browsing, to complex operations like data analysis, artificial intelligence, and virtual reality. Some common types of software in computer engineering include operating systems, application software, programming software, database software, security software, and AI-related software.

6.1 Operating Systems

Operating systems (OS) are an essential component of computer engineering. They are software programs that manage computer hardware and provide services for computer programs. The following are some key aspects of operating systems in computer engineering:

Resource Management

An OS manages computer resources such as memory, CPU, input/output devices, and storage devices. It allocates these resources to different programs based on their needs and priorities.

Process Management

An OS manages the execution of multiple programs (processes) on a computer system. It schedules processes and provides mechanisms for communication and synchronization between them.

Memory Management

An OS manages the computer's memory, which is a limited resource. It allocates memory to processes, tracks memory usage, and provides mechanisms for memory protection and sharing.

File Management

An OS manages files and directories on the computer's storage devices. It provides mechanisms for creating, deleting, and modifying files, as well as for accessing and sharing files between processes.

Security

An OS provides mechanisms for protecting computer systems and data from unauthorized access and malicious software. It includes features such as authentication, encryption, and access control.

Examples of popular operating systems used in computer engineering include Windows, macOS, Linux, Android, and iOS. Different types of operating systems are used for different types of devices, such as desktop computers, mobile devices, servers, and embedded systems. Overall, operating systems are critical to the functioning of computer engineering systems, and their design and implementation are a fundamental aspect of computer engineering.

6.2 Application Software

Application software refers to the programs or software applications that are designed to perform specific tasks for end-users. In computer engineering, application software is developed to meet various needs, from personal computing to business and industry-specific needs. Here are some examples of application software used in computer engineering:

Word Processors

Word processors like Microsoft Word, Google Docs, and OpenOffice are used for creating and editing documents, such as reports, resumes, and letters.

Spreadsheet Applications

Spreadsheet applications like Microsoft Excel, Google Sheets, and OpenOffice Calc are used for creating and managing spreadsheets, performing calculations, and analyzing data.

Multimedia Applications

Multimedia applications like Adobe Photoshop, Premiere, and InDesign are used for creating and editing images, videos, and other multimedia content.

Web Browsers

Web browsers like Google Chrome, Mozilla Firefox, and Safari are used for accessing the internet and browsing websites.

Email Applications

Email applications like Microsoft Outlook, Gmail, and Apple Mail are used for managing email communication.

Computer-Aided Design (CAD) software

CAD software like AutoCAD, SolidWorks, and Fusion 360 are used in engineering, architecture, and product design to create 2D and 3D models.

Virtualization software

Virtualization software like VMware, VirtualBox, and Hyper-V is used for creating virtual machines, which allow multiple operating systems to run on a single physical computer.

Scrum software

Scrum is an agile framework that is used to manage complex projects. It was initially developed for software development, but it can be applied to any project where there is a need to deliver value quickly and adapt to changing requirements. Scrum is based on a set of values, principles, and practices that encourage collaboration, communication, and transparency among team members.

In Scrum, the project is divided into short iterations called sprints, typically lasting two to four weeks. At the beginning of each sprint, the team meets to plan the work to be done during the sprint, and at the end of the sprint, the team meets to review the work that was completed and plan for the next sprint. The team works together to deliver a potentially releasable product increment at the end of each sprint.

Scrum emphasizes self-organization and cross-functional teams, with a product owner responsible for prioritizing and managing the product backlog, and a Scrum Master responsible for ensuring the team follows the Scrum process and removing any impediments that may arise. The team uses daily stand-up meetings to communicate progress, identify any obstacles, and plan the day’s work.

Scrum is a popular framework for project management, and it has been adopted by many organizations around the world.

There are many software tools available to support Scrum processes and help teams manage their work more effectively. Some popular Scrum software tools include

1. Jira: A project management tool that allows teams to plan and track their work using Scrum or other agile methodologies. It includes features for backlog management, sprint planning, sprint boards, and reporting.
2. Trello: A visual collaboration tool that helps teams organize their work using boards, lists, and cards. It can be used to manage Scrum boards, with features for backlog management, sprint planning, and team communication.
3. Asana: A project management tool that allows teams to track their work using lists and tasks. It includes features for sprint planning, backlog management, and team communication.
4. Monday.com: A team management tool that allows teams to track their work using boards and timelines. It includes features for backlog management, sprint planning, and team communication.
5. Agilefant: A web-based tool that provides Scrum support for project management. It includes features for backlog management, sprint planning, and reporting.
6. Scrumwise: A web-based tool that provides Scrum support for project management. It includes features for backlog management, sprint planning, and team communication.
7. VersionOne: A tool that provides a centralized platform for agile project management, including Scrum support. It includes features for backlog management, sprint planning, and reporting.
8. Agile Central: A cloud-based tool that provides agile project management support, including Scrum. It includes features for backlog management, sprint planning, and team collaboration.
9. Pivotal Tracker: A tool that provides project management support for agile teams, including Scrum. It includes features for backlog management, sprint planning, and team collaboration.
10. Targetprocess: A tool that provides agile project management support, including Scrum. It includes features for backlog management, sprint planning, and team collaboration.
11. SprintGround: A tool that provides Scrum support for project management. It includes features for backlog management, sprint planning, and team communication.
12. ScrumDo: A web-based tool that provides Scrum support for project management. It includes features for backlog management, sprint planning, and team collaboration.

6.3 Programming Software

Programming software is a set of tools and software applications that are used by software developers to create, test, and debug computer programs. In computer engineering, programming software is a critical component of the software development process. Here are some examples of programming software used in computer engineering:

Integrated Development Environments (IDEs)

IDEs are software applications that provide a complete development environment for software developers. Examples include Visual Studio, Eclipse, and NetBeans. IDEs include tools for writing code, debugging, version control, and project management.

Text Editors

Text editors are lightweight software applications that allow developers to write and edit code. Examples include Sublime Text, Atom, and Notepad++. Text editors are often used in conjunction with other tools like version control software and compilers.

Compilers

Compilers are software programs that convert high-level programming language code into machine language code that can be executed by a computer. Examples include GCC, Clang, and Visual C++.

Debuggers

Debuggers are software tools used to find and fix errors or bugs in software programs. Examples include GDB, Visual Studio Debugger, and LLDB.

Code Libraries

Code libraries are collections of pre-written code that developers can use in their own programs. Examples include Boost, jQuery, and NumPy.

Testing Frameworks

Testing frameworks are software tools that automate the testing of software programs. Examples include JUnit, NUnit, and PyTest.

Overall, programming software is an essential part of computer engineering, providing developers with the tools they need to create and test software programs efficiently and effectively. The development of programming software requires expertise in software engineering, computer science, and programming languages.

6.4 Database Software

Database software is used in computer engineering to manage and organize large amounts of data. Databases are used to store, retrieve, and manipulate data for various purposes, such as inventory management, customer relationship management, and financial transactions. Here are some examples of database software used in computer engineering:

Relational Database Management Systems (RDBMS)

RDBMS is a type of database software that organizes data into tables and uses SQL (Structured Query Language) to access and manipulate the data. Examples include MySQL, Oracle Database, and Microsoft SQL Server.

NoSQL Databases

NoSQL databases are used for managing unstructured or semi-structured data, such as social media data or data from the internet of things (IoT). Examples include MongoDB, Cassandra, and Couchbase.

Object-Oriented Databases

Object-oriented databases are used to store data in object-oriented programming (OOP) languages, such as Java or C++. Examples include db4o, Versant Object Database, and ObjectStore.

Cloud Databases

Cloud databases are databases that are hosted on cloud platforms, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud. Cloud databases provide scalability, reliability, and accessibility.

In-Memory Databases

In-memory databases store data in computer memory instead of on disk. They are designed to provide fast access to data for real-time applications, such as financial trading or fraud detection. Examples include SAP HANA, Oracle TimesTen, and Redis.

Overall, database software is an essential part of computer engineering, providing a mechanism for organizing and managing data. The development of database software requires expertise in database design, database management systems, and database programming languages.

6.5 Security Software

Security software is an important part of computer engineering, as it is used to protect computer systems and networks from various threats, such as viruses, malware, and hacking attempts. Here are some examples of security software used in computer engineering:

Antivirus software

Antivirus software is used to detect and remove viruses, spyware, and other malware from computer systems. Examples include Norton, McAfee, and Avast.

Firewalls

Firewalls are used to monitor and control incoming and outgoing network traffic to prevent unauthorized access to a computer system or network. Examples include Windows Firewall, Cisco ASA, and Fortinet FortiGate.

Intrusion Detection and Prevention Systems (IDPS)

IDPS is used to detect and prevent unauthorized access to computer systems and networks. Examples include Snort, Suricata, and McAfee Network Security Platform.

Virtual Private Networks (VPNs)

VPNs are used to create a secure connection between a computer system and a remote network over the internet. VPNs use encryption to protect data from interception and hacking attempts. Examples include NordVPN, ExpressVPN, and OpenVPN.

Password Managers

Password managers are used to securely store and manage passwords for multiple online accounts. Password managers use encryption to protect passwords from unauthorized access. Examples include LastPass, 1Password, and Dashlane.

Encryption Software

Encryption software is used to encrypt data to protect it from unauthorized access. Examples include VeraCrypt, BitLocker, and OpenSSL.

Summary

Overall, security software is an essential part of computer engineering, providing protection against various threats to computer systems and networks. The development of security software requires expertise in security protocols, encryption algorithms, network architecture, and software engineering.

6.6 AI-related Software

AI-related software is adaptive, learning from data to improve over time. It automates decision-making, processes natural language, recognizes images and speech, and solves complex problems. Scalable and capable of real-time processing, it emphasizes explainability, fairness, privacy, and security. AI software often includes self-learning, correction, and reasoning, enhancing its performance across diverse applications. AI-related software covers a broad range of applications and tools, from machine learning frameworks to AI-driven platforms. Some of such software are included below:

Machine Learning (ML) Frameworks

These are libraries and tools that help in building, training, and deploying AI models. Some examples are:

- **TensorFlow** [24] : An open-source library. developed by Google for deep learning and machine learning applications.
- **PyTorch** [25]: Developed by Facebook, it's another popular deep learning framework known for its flexibility and ease of use.
- **Keras** [26]: A high-level neural networks API, written in Python, capable of running on top of TensorFlow.
- **Scikit-learn** [27]: A Python library for classical machine learning algorithms like classification, regression, and clustering.

Natural Language Processing (NLP) Tools

NLP focuses on the interaction between computers and human language.

- **SpaCy** [28]: An open-source NLP library in Python for tasks like tokenization, part-of-speech tagging, and named entity recognition.
- **NLTK (Natural Language Toolkit)** [29]: A suite of libraries and programs for symbolic and statistical NLP for English.
- **Hugging Face Transformers** [30]: A library for state-of-the-art NLP models like BERT, GPT, and T5.
- **OpenAI GPT** [31]: This family of language models (e.g., GPT-3, GPT-4) is used for text generation, question-answering, and language understanding.

Computer Vision Libraries

Computer vision focuses on the visual perception of computers.

- **OpenCV** [32]: A popular open-source computer vision library for real-time image processing and computer vision tasks.
- **TensorFlow Vision** [33]: Part of TensorFlow, it's specialized for deep learning models in image classification, detection, and segmentation.
- **Detectron2** [34]: A Facebook AI Research (FAIR) project for object detection and segmentation.
- **YOLO (You Only Look Once)** [35]: A real-time object detection system.

Robotic Process Automation (RPA)

RPA automates repetitive tasks.

- **UiPath** [36]: One of the leading platforms for RPA, it allows automating repetitive tasks in business processes.
- **Automation Anywhere** [37]: Another major RPA tool used for automating business processes across multiple systems.
- **Blue Prism** [38]: A software platform for robotic process automation that emphasizes scalability and security.

AI Development Platforms

These platforms offer integrated environments to build and deploy AI solutions.

- **Google AI Platform**: A suite of cloud-based tools from Google for building AI models, including AutoML for automating the model development process.
- **AWS SageMaker**: Amazon's platform for building, training, and deploying machine learning models in the cloud.
- **Microsoft Azure AI**: Microsoft's cloud service offering AI and machine learning tools, including Azure Machine Learning.

- **IBM Watson**: An AI platform that provides natural language processing, machine learning, and computer vision tools.

AI-driven Analytics Tools

These tools use AI to automate data analysis.

- **Tableau with AI Extensions**: Tableau integrates AI-driven insights into data visualization through tools like Einstein Analytics.
- **DataRobot**: A platform that automates machine learning model building and deployment.
- **H2O.ai**: A machine learning platform that allows users to build models using various algorithms, supporting automatic model selection.

Reinforcement Learning (RL) Libraries

RL is used to train agents to make a sequence of decisions.

- **OpenAI Gym**: A toolkit for developing and comparing reinforcement learning algorithms.
- **Stable Baselines**: A set of implementations of reinforcement learning algorithms built on OpenAI Gym and TensorFlow.
- **RLlib**: A scalable RL library built on top of Ray, a framework for distributed applications.

AI in Edge Computing

These software tools enable AI models to run on edge devices (e.g., IoT devices).

- **TensorFlow Lite**: TensorFlow's lightweight version for mobile and IoT devices.
- **AWS IoT Greengrass**: Allows local execution of machine learning inference at the edge.
- **Edge Impulse**: A platform focused on creating ML models for embedded devices.

Each of these software tools plays a crucial role in the AI ecosystem, enabling a wide range of applications from image recognition to language understanding and automation.

7 Network

A computer networking is a process of connecting multiple computers and/or computing devices with the purpose to communicate: exchange data and share resources with each other.

Internet is one example where different computer systems, especially at different locations, are connected through a system of rules, called communications protocols, to transmit information over physical or wireless technologies.

The basic components of computer networking are bodes and links. Many devices such as computers, modems, or switchs are considered as nodes. Between any nodes are transmission media that are called links. Long links usually use low-loss optical fibers, while short links usually use metal cables or free space as in wireless networks.

Class Activity

1. Draw a computer network and label all nodes and links.
2. Compare your network with your team members.
3. Redraw the network with with new types of nodes and links.

[1]

Networking is an important area in computer engineering that deals with the design, implementation, and management of computer networks. Here are some topics related to networking in computer engineering.

7.1 Network Architecture

Network architecture refers to the design and structure of a computer network. It includes the physical layout of network devices, such as routers, switches, and servers, as well as the logical structure of the network, such as

[1] List here

- Types of network nodes:
- Types of network links:

the addressing scheme and routing protocols that enable communication and data exchange between devices.

The most common types of network architecture include:

- Client-server architecture: In this architecture, one or more central servers provide resources or services to client devices that request them over the network. Examples of client-server architectures include web servers and email servers.
- Peer-to-peer architecture: In this architecture, all devices on the network are considered equal and can share resources and communicate with each other directly. Peer-to-peer architectures are often used for file sharing and collaboration.
- Hybrid architecture: This architecture combines elements of both client-server and peer-to-peer architectures. For example, a hybrid network might use client-server architecture for centralized management and control, but also allow peer-to-peer communication for certain tasks.

[2]

Network architecture is a critical aspect of computer engineering, as it determines how efficiently and effectively data can be transmitted between devices. Different network architectures are better suited for different types of applications and use cases, and must be carefully designed and implemented to ensure optimal performance and security.

7.2 Network Protocols

Network protocols are a set of rules and standards that govern the exchange of data between network devices. These protocols define the format and structure of data packets, as well as the sequence of actions that must be taken by devices to transmit and receive data.

There are many different network protocols that are used in computer engineering. Some of the most common ones include:

- Transmission Control Protocol (TCP): TCP is a connection-oriented protocol that provides reliable, ordered data transmission between devices. It ensures that data is received in the correct order and retransmits lost packets.
- User Datagram Protocol (UDP): UDP is a connectionless protocol that provides unreliable, unordered data transmission between devices. It is faster than TCP but does not guarantee the delivery of all packets.
- Internet Protocol (IP): IP is a protocol that provides the routing and addressing functions necessary for data transmission over the internet. It defines the format and structure of IP packets, which are used to transmit data between devices on different networks.
- Simple Mail Transfer Protocol (SMTP): SMTP is a protocol that is used for sending email messages between devices on a network. It defines the format and structure of email messages and specifies the rules for transferring them between email servers.

[2] By online searching or other methods,

- Identify a new network architecture that is not listed above.
- Explain it briefly.

- File Transfer Protocol (FTP): FTP is a protocol that is used for transferring files between devices on a network. It defines the format and structure of file transfer commands and data, and specifies the rules for transferring files between FTP servers and clients.

3

3:

- List a network protol that you are familiar with the most.
- Explain it briefly.

Network protocols are essential for enabling communication and data exchange between devices on a computer network. Different protocols are designed for different types of applications and use cases, and must be carefully selected and configured to ensure optimal network performance and security.

7.3 Wireless Networking

Wireless networking is a technology that enables devices to connect to a computer network without the need for physical cables or wires. Wireless networks use radio waves to transmit data between devices, allowing users to access network resources and the internet from anywhere within range of the network.

There are several types of wireless networking technologies used in computer engineering:

- Wi-Fi: Wi-Fi is the most common type of wireless networking technology, and is used for connecting devices to local area networks (LANs) and the internet. Wi-Fi networks use radio waves to transmit data between devices, and typically operate within a range of a few hundred feet.
- Bluetooth: Bluetooth is a wireless technology that is used for connecting devices within a short range, typically up to 30 feet. Bluetooth is commonly used for connecting devices such as smartphones, headphones, and speakers.
- Zigbee: Zigbee is a low-power wireless networking technology that is used for connecting devices in a network. Zigbee networks are commonly used in home automation and Internet of Things (IoT) applications.
- Cellular: Cellular networks use wireless technology to provide internet connectivity to devices over large geographic areas. Cellular networks are commonly used for smartphones, tablets, and other mobile devices.

4

4:

- Do you see any challenges for the future of wireless networking?
- Elaborate with some details.

Wireless networking is a critical component of modern computer engineering, as it enables users to access network resources and the internet from anywhere within range of a wireless network. However, wireless networks also present security challenges, as data transmitted over a wireless network can be intercepted by unauthorized users. As such, it is important to implement appropriate security measures, such as encryption and strong passwords, to protect wireless networks and the devices that connect to them.

7.4 Network Security

Network security is the practice of implementing measures to protect computer networks from unauthorized access, theft, damage, or disruption. Network security is a critical component of computer engineering, as networks are often vulnerable to a wide range of threats, including malware, viruses, hacking, and cyber attacks.

There are several techniques used in network security in computer engineering:

- Firewalls: Firewalls are hardware or software systems that control access to a network, and prevent unauthorized access from external networks. Firewalls can be configured to block certain types of traffic, and can be used to monitor network traffic for suspicious activity.
- Intrusion Detection and Prevention Systems (IDPS): IDPS are systems that monitor network traffic for signs of a security breach, and can be configured to automatically respond to detected threats. IDPS systems can detect a range of attacks, including malware, viruses, and hacking attempts.
- Virtual Private Networks (VPN): VPNs are a technology that enables users to access a network securely over the internet. VPNs encrypt network traffic, making it difficult for unauthorized users to intercept and access data transmitted over the network.
- Encryption: Encryption is a technique used to protect data by converting it into an unreadable format. Encryption can be used to protect sensitive data transmitted over a network, making it difficult for unauthorized users to access or read.
- User Authentication: User authentication is a technique used to verify the identity of users accessing a network. User authentication can be implemented through the use of passwords, biometric identification, or other techniques.

Network security is a critical aspect of computer engineering, and must be carefully designed and implemented to protect networks from a wide range of threats. It is important to regularly update network security measures and stay up-to-date on emerging threats and vulnerabilities to ensure optimal network security.

7.5 Network Management

Network management is the process of overseeing and controlling the operation of a computer network. Network management encompasses a wide range of activities, including network design, deployment, maintenance, monitoring, and troubleshooting.

Some of the key tasks involved in network management in computer engineering include:

- Network planning and design: Network managers are responsible for planning and designing computer networks that meet the needs of the organization. This involves assessing the organization's requirements for network bandwidth, security, and reliability, and designing a network architecture that meets those requirements.
- Network deployment and configuration: Once a network has been designed, network managers are responsible for deploying and configuring the network hardware and software. This includes setting up network devices such as routers, switches, and firewalls, and configuring network protocols and services such as DHCP, DNS, and NAT.
- Network monitoring and optimization: Network managers must monitor network performance to ensure that it is meeting the organization's needs. This involves monitoring network traffic, analyzing network performance data, and identifying and resolving performance issues.
- Network security: Network managers are responsible for ensuring that the network is secure and protected from unauthorized access and other security threats. This involves implementing security measures such as firewalls, intrusion detection systems, and user authentication protocols.
- Network maintenance and troubleshooting: Network managers must maintain the network hardware and software, and troubleshoot and resolve any issues that arise. This includes updating network devices with the latest firmware and software patches, replacing faulty hardware, and diagnosing and resolving network connectivity issues.

Effective network management is critical to the success of any organization that relies on a computer network. By ensuring that the network is designed, deployed, and maintained to meet the organization's needs, network managers can help to ensure that the network is reliable, secure, and performing optimally.

7.6 Cloud Networking

Cloud networking refers to the use of cloud computing technology to deliver network services over the internet. It allows organizations to connect their on-premises networks to cloud-based networks or services. Cloud networking enables organizations to extend their network infrastructure into the cloud, and to take advantage of the scalability, flexibility, and cost savings offered by cloud computing. It is typically achieved through the use of virtual private networks (VPNs) or direct connections to cloud service providers.

Technologies include cloud-based network management, software-defined networking (SDN), and network function virtualization (NFV).

Some of the key benefits of cloud networking include:

- Scalability: Cloud networking allows organizations to easily scale their network infrastructure up or down to meet changing demands. With cloud networking, organizations can quickly provision new resources and services as needed, without the need for costly hardware upgrades or reconfigurations.
- Cost savings: Cloud networking can help organizations reduce their network infrastructure costs by eliminating the need for expensive hardware and reducing the need for in-house network management and maintenance.
- Flexibility: Cloud networking allows organizations to access a wide range of cloud-based services and resources, and to easily integrate these services with their existing on-premises network infrastructure.
- Reliability: Cloud networking services are typically highly reliable, with robust network redundancy and failover capabilities that help ensure network uptime and availability.
- Security: Cloud networking services typically provide advanced security features such as firewalls, intrusion detection and prevention, and user authentication and access controls, which can help organizations protect their network infrastructure from security threats.

Cloud networking is a rapidly evolving technology, and new services and capabilities are being introduced all the time. By leveraging cloud networking technologies, organizations can take advantage of the scalability, flexibility, and cost savings offered by cloud computing, while also ensuring that their network infrastructure remains secure and reliable.

7.7 Summary

Overall, networking is a critical area in computer engineering, enabling the communication and exchange of data between devices and users. The development of networking technologies requires expertise in computer science, electrical engineering, and telecommunications.

8 Embedded Systems

8.1 Introduction

Embedded systems are specialized computer systems that are designed to perform specific functions within a larger system or device. They are often hidden from view, embedded within other products or systems, and are not typically intended for use as standalone devices. Embedded systems can be found in a wide range of products and applications, especially applications with size, weight, power, and cost (SWaP-C) constraints, such as consumer electronics, automotive systems, medical devices, and industrial control systems. They are designed to perform a specific set of tasks or functions and are often optimized for performance, power consumption, and cost-effectiveness.

8.2 History of Embedded Systems

The origins of embedded systems can be traced back to the early days of computing, when electronic systems were used to control industrial processes and military equipment. In the 1960s, the development of integrated circuits made it possible to design smaller and more powerful electronic systems, and the first embedded systems began to appear in consumer products such as calculators and digital watches.

The 1970s saw the emergence of microcontrollers, which combined a microprocessor with memory, input/output (I/O) ports, and other peripherals on a single chip. This made it possible to design highly integrated embedded systems that were smaller, faster, and more cost-effective than earlier designs.

The 1980s and 1990s saw a rapid expansion in the use of embedded systems, as they became increasingly important in a wide range of industries and applications. The development of real-time operating systems (RTOS) and software development tools made it easier to design and test complex embedded systems, and the rise of the Internet of Things (IoT) has led to a new wave of innovation in this field.

More detailed description can be found in books [39] and [40].

[39]: Jiménez et al. (2013), *Introduction to embedded systems*

[40]: Lee et al. (2013), *Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition*

8.3 Characteristics of Embedded Systems

Embedded systems have several characteristics that distinguish them from other types of computer systems. These include:

- Real-time operation: Embedded systems are often designed to operate in real-time, meaning that they must respond to external events or stimuli within a specific time frame. This is especially important in safety-critical applications such as medical devices or automotive systems, where delays or failures can have serious consequences.
- Limited resources: Embedded systems are often designed to operate with limited resources, including memory, processing power, and power consumption. This requires careful optimization of software and hardware components to ensure that the system can perform its intended function without exceeding these limits.
- Dedicated functions: Embedded systems are typically designed to perform a specific set of functions or tasks, rather than being general-purpose computing devices. This allows for more efficient use of resources and can lead to faster and more reliable performance.
- Integration: Embedded systems are often integrated into larger systems or devices, such as an engine control unit in a car or a heart monitor in a hospital. This requires careful coordination with other hardware and software components to ensure that the system functions correctly and reliably.

A tutorial on the characteristics of embedded systems can be found on [41].

[41]: Williams (2023), *Embedded Systems Tutorial: What is, History & Characteristics*

8.4 Design Principles of Embedded Systems

Designing embedded systems requires a combination of hardware and software engineering skills, as well as an understanding of the system's intended use and environment. Some of the key design principles for embedded systems include:

- System requirements: The first step in designing an embedded system is to define the system requirements, including the intended function, performance, power consumption, and cost.

- Hardware design: The hardware design involves selecting appropriate microcontrollers or microprocessors, designing and testing circuits, and integrating them with other hardware components such as sensors, actuators, and communication interfaces.
- Software design: The software design involves programming the system to perform the desired function, often using low-level programming languages such as C or assembly language. This requires an understanding of the underlying hardware architecture and the ability to optimize code for performance and efficiency.
- Testing and verification: Once the hardware and software have been designed, the system must be thoroughly tested and verified to ensure that it meets the system requirements and operates reliably in the intended environment. This may involve simulation, emulation, and hardware testing, as well as software testing and verification.
- Maintenance and updates: Embedded systems are often designed to operate for many years without significant maintenance or updates. However, as technology evolves and new requirements arise, it may be necessary to update the system to ensure continued performance and compatibility.

More details can be found in book [42]

[42]: Murti (2021), *Design Principles for Embedded Systems*

8.5 Applications of Embedded Systems

Embedded systems are used in a wide range of applications, from consumer electronics to industrial automation. Some of the most common applications include:

- Automotive systems: Embedded systems are used extensively in modern automobiles to control various functions, such as engine control units (ECUs) to infotainment systems and advanced driver assistance systems (ADAS), anti-lock braking systems (ABS), airbags, and navigation. These systems require real-time operation, high reliability, and integration with other systems in the vehicle.
- Medical devices: Many medical devices, such as patient monitors, insulin pumps, and pacemakers, use embedded systems to control their operation. These systems require real-time operation, high reliability, and strict adherence to safety standards.
- Consumer electronics: Embedded systems are used in a variety of consumer electronics devices, such as digital cameras, smart TVs, mobile phones, and smart home devices. These systems often rely on real-time operation and require low power consumption to ensure long battery life.
- Industrial control systems: Embedded systems are used in industrial automation and control systems, such as programmable logic controllers (PLCs), process control systems, robotics, and sensors, to control manufacturing processes and other industrial applications. These systems require real-time operation, high reliability, and integration with other systems in the manufacturing process.

- Aerospace and defense: Embedded systems are used in aerospace and defense applications, such as avionics systems, missile guidance systems, and satellite control systems.
- Robotics: Embedded systems are used extensively in robotics applications, such as in the control of robot arms, sensor systems, and autonomous vehicles.
- Home appliances: Embedded systems are used in home appliances, such as refrigerators, washing machines, and dishwashers, to control their operation and provide user interfaces.
- Security systems: Embedded systems are used in security systems, such as access control systems, surveillance cameras, and alarms, to control and monitor access to buildings and properties.

8.6 Challenges and Opportunities

The rapid evolution of technology and the increasing complexity of embedded systems present both challenges and opportunities in this field. Some of the key challenges include:

- Security: As embedded systems become more connected and integrated with other systems, they become more vulnerable to cyber attacks and other security threats. Ensuring the security of these systems is a critical challenge for designers and developers.
- Complexity: Embedded systems are becoming increasingly complex, with more powerful processors, larger memory, and more sophisticated software. This complexity can make it more difficult to design, test, and maintain these systems.
- Standards: As the use of embedded systems grows, the need for standards and interoperability becomes more important. Developing and adhering to standards can help ensure that embedded systems work together reliably and efficiently.

At the same time, there are also many opportunities in the field of embedded systems. Some of the key opportunities include:

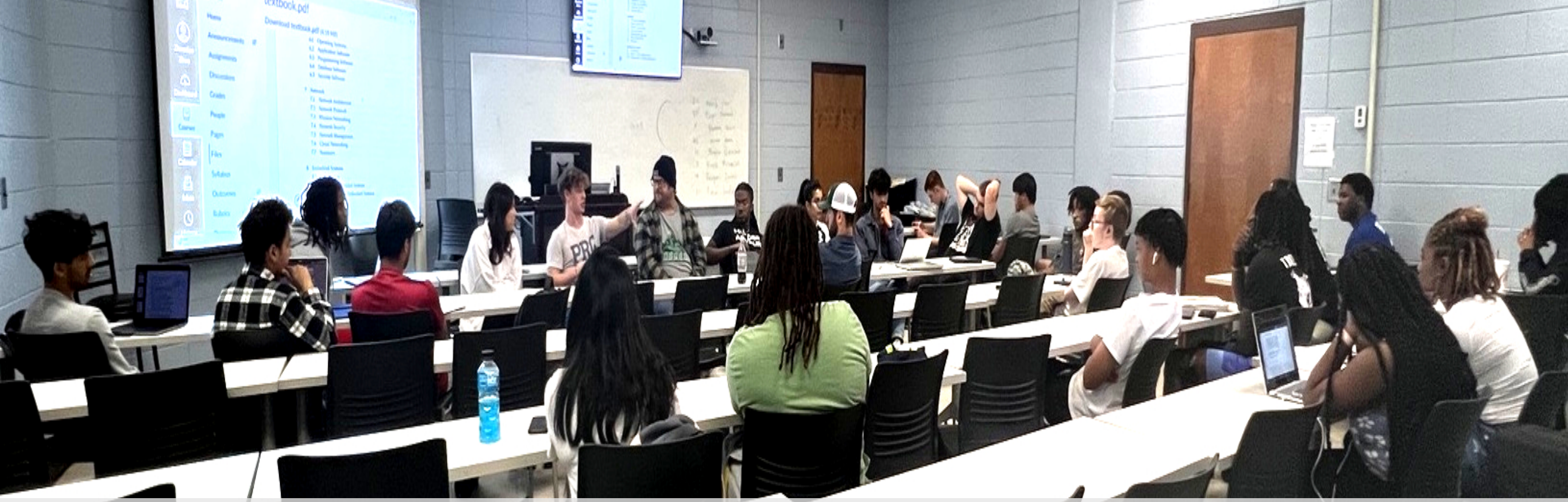
- IoT: The rise of the Internet of Things (IoT) has led to a new wave of innovation in embedded systems. As more devices become connected, there is a growing need for embedded systems that can collect, analyze, and act on data in real-time.
- Machine learning: The growing availability of machine learning tools and algorithms is opening up new opportunities for embedded systems, such as predictive maintenance and autonomous decision-making.
- Sustainability: Embedded systems can play an important role in promoting sustainability, by optimizing energy consumption, reducing waste, and improving efficiency in a wide range of applications.

8.7 Summary

Embedded systems are an important and rapidly evolving area of computer science and engineering, with applications in a wide range of industries and domains. They are designed to perform specific functions within a larger system or device, and are often optimized for performance, power consumption, and cost-effectiveness. As technology continues to evolve, the opportunities and challenges in this field will continue to grow, making it an exciting and dynamic area for research and innovation.

Further readings can be found at the follows [43, 44].

[43]: Keller (2023), *The future of high-performance embedded computing*
[44]: Beningo (2023), *Will AI take embedded software jobs?*



9 Artificial Intelligence

9.1 Introduction

Artificial Intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think and act like humans. It is a branch of computer science that focuses on creating intelligent machines that can perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and language translation.

Although the groundwork for AI can be traced much earlier back, the biggest breakthrough weren't made until the 1950s. The most commenly agreed start of artifical intelligce is 1950, when Alan Turing published "Computer Machinery and Intelligence" [45]. The Turing Test has since then been used to measure computer intelligence.

AI can be classified into two categories: narrow or weak AI, and general or strong AI. Narrow AI is designed to perform a specific task or set of tasks, such as image recognition or speech recognition. General AI, on the other hand, is capable of performing any intellectual task that a human can do.

There are several subfields of AI, including machine learning, natural language processing, computer vision, and robotics. Machine learning is the process of training algorithms to recognize patterns in data, while natural language processing involves analyzing and understanding human language. Computer vision involves teaching machines to interpret and understand images and video, and robotics involves designing and building machines that can perform tasks autonomously.

AI has the potential to revolutionize many industries and fields, from healthcare and transportation to finance and education. AI has become an increasingly important topic in the field of computer engineering in recent years. This is because AI has the potential to revolutionize the way computer systems are designed, developed, and maintained. Bill Gates

[45]: Turing (1950), 'Computing Machinery and Intelligence'

noted that “AI is about to completely change how you use computers" [46].

[46]: Gates (2023), *AI is about to completely change how you use computers*

Artificial intelligence is changing the business landscape and redefining the workforce as well. In the near future, many human workers need to help computers to get the job done. Further discussions on the potential impact of large language models (LLMs) can be found at [47, 48].

[47]: Felten et al. (2023), *How will Language Modelers like ChatGPT Affect Occupations and Industries?*
[48]: Eloundou et al. (2023), *GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models*

In this chapter, we will explore the various applications of AI in computer engineering, including design optimization, quality control, predictive maintenance, cybersecurity, and intelligent automation.

9.2 AI Algorithms

There are many common AI algorithms that are used in machine learning and other applications of artificial intelligence.

There are different ways to classify AI algorithms. For exmaple, they can be

- Supervised learning: from labeled data, to learn patterns and relationships between input and output data.
- Unsupervised learning: with unlabeled data, to learn patterns and insights between input and output without explicit guidance.
- Reinforcement learning: interacts with the environment, without labelled data pairs, to learn optimal actions based on rewarding or punishing behaviors.

They can also be

- Classification algorithms: to categorize data into a class or category.
- Regression algorithms: to predict numerical values based on input data.
- Clustering algorithms: to group objects into clusters where objects are more similar in one cluster but are not similar between clusters.

Some algorithms can be both supervised and unsupervised, or ensemble learning algorithm, which are a combination of several models in a single problem.

Here are a few examples:

Linear Regression A statistical technique that is used to find the best fit line that describes the relationship between two or more continuoous variables. It is considered one of the simplest AI algoritms.
Figure 9.1 is a graphical explanation of linear regression algorithm. A more detailed explanation of linear regression with Python can be found on web [49].

[49]: Stojiljkovic (2023), *Linear Regression in Python*

Example 9.2.1 Suppose we have a dataset of house prices and the corresponding square footage of each house.
Objective: we want to use this data to create a model that can predict

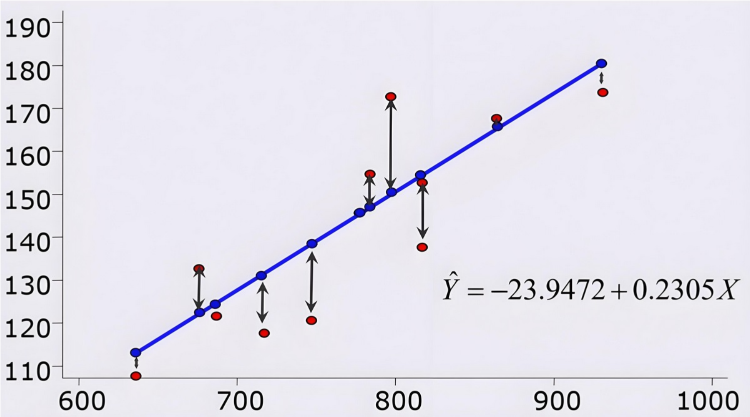


Figure 9.1: Linear Regression

the price of a house given its square footage.
Procedure: to do this,

1. We use linear regression to find the best-fit line that describes the relationship between house prices and square footage. This line will be of the form:

$$y = mx + b$$

where y is the predicted house price, x is the square footage, m is the slope of the line, and b is the intercept.
Using the dataset, the linear regression algorithm will determine the values of m and b that minimize the difference between the predicted values and the actual values in the dataset.

2. Once the model is trained, we can use it to predict the price of a new house based on its square footage.
For example, if the model determines that the best-fit line is:

$$y = 100x + 50$$

then a house with a square footage of 1500 would have a predicted price of

$$y = 100 \times 1500 + 50 = \$150,050$$

[1]

Linear regression can be used in computer engineering:

- Software development effort estimation [50]: Linear regression can be used to estimate the amount of effort required for software development tasks. By using historical data on software development tasks and their associated effort, a linear regression model can be trained to predict the effort required for new tasks based on their characteristics.
- Network traffic prediction [51]: Linear regression can be used to predict network traffic, which can help in network capacity planning and management. By using historical data on network traffic and associated factors such as time of day and day of the

[1]: Have you ever used linear regression algorithm before you even know AI?

[50]: Sharma et al. (2020), 'Linear Regression Model for Agile Software Development Effort Estimation'

[51]: Ulanowicz et al. (2023), 'Combining Random Forest and Linear Regression to Improve Network Traffic Prediction'

week, a linear regression model can be trained to predict future traffic levels.

- Hardware performance analysis [52]: Linear regression can be used to analyze the performance of hardware components, such as processors and storage devices. By using data on hardware specifications and performance benchmarks, a linear regression model can be trained to predict the performance of new hardware components.
- Fault detection in industrial systems [53]: Linear regression can be used to detect faults in industrial systems, such as manufacturing plants and power grids. By using sensor data and historical fault data, a linear regression model can be trained to detect deviations from normal operating conditions, indicating the presence of faults.

[52]: Joseph et al. (2006), 'Construction and use of linear regression models for processor performance analysis'

[53]: Latocha (2018), 'Robust fault detection, location, and recovery of damaged data using linear regression and mathematical models'

Logistic Regression A statistical method used to model the probability of a certain event or outcome based on input variables.

Compared to linear regression, which are used for regression problems, logistic regression algorithms are used for classification problems. The output is continuous for linear regression while it is discreet for logistic regressions.

Figure 9.2 is a graphical explanation of logistic regression algorithm.

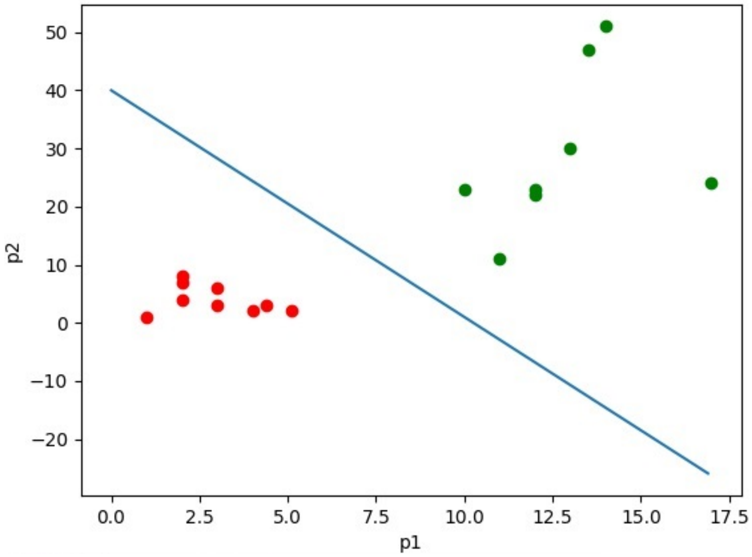


Figure 9.2: Logistic Regression

2

As logistic regression is widly used, we can easily find many references of logistic regression in literature. A detailed and useful explanation with Python coding can be found on web [54].

2: Can you summarize the difference between linear regression and logistic regression?

[54]: Banoula (2023), *An Introduction to Logistic Regression in Python*

Example 9.2.2 Suppose we have a dataset of medical records.
Objective: we want to create a model that can predict whether a patient has a certain medical condition based on their age and other health metrics.

Proceedure: to do this,

1. We can use logistic regression to model the probability of the patient having the condition as a function of their age and other health metrics. The logistic regression model will output a probability value between 0 and 1, where values closer to 1 indicate a higher probability of the patient having the condition.
[3]
2. Once the model is trained, we can use it to predict the probability of a new patient having the condition based on their age and health metrics. If the predicted probability is above a certain threshold, we can classify the patient as having the condition.
3. For example, suppose the logistic regression model determines that the probability of a patient having the condition is:

$$p = \frac{1}{1 + e^{-z}},$$

where z is a linear combination of the input variables. If the threshold for classifying a patient as having the condition is 0.5, then we can classify a patient as having the condition if:

$$p \geq 0.5$$

Overall, logistic regression is a useful AI algorithm for predicting binary outcomes and can be applied in a wide range of applications, such as healthcare, finance, and marketing.

Logistic regression can be used in computer engineering:

- Spam detection [55]: Logistic regression can be used to classify emails as spam or not spam. By using features such as the email content, sender information, and email headers, a logistic regression model can be trained to predict whether an email is spam or not.
- Image classification [56]: Logistic regression can be used to classify images into different categories. By using features such as pixel values and image attributes, a logistic regression model can be trained to predict the category of an image.
- Credit risk analysis [57]: Logistic regression can be used to predict credit risk, which can help in making lending decisions. By using historical data on credit applications and associated risk factors, a logistic regression model can be trained to predict the likelihood of default for new credit applications.
- Medical diagnosis [58]: Logistic regression can be used to diagnose medical conditions based on patient symptoms and other factors. By using data on patient symptoms and medical history, a logistic regression model can be trained to predict the likelihood of different medical conditions.

Decision Trees A tree-like model of decisions and their possible consequences, used to generate a classification or regression model.

Figure 9.3 is a graphical explanation of decision tree algorithm.

3: Logistic Regression is a statistical method used for binary classification, which is the task of categorizing items into two classes. It uses the logistic function (also known as the sigmoid function) to model the probability of a binary outcome. Can you explain how this works?

[55]: Dedeturk et al. (2020), 'Spam filtering using a logistic regression model trained by an artificial bee colony algorithm'

[56]: Song et al. (2023), 'Doubly robust logistic regression for image classification'

[57]: Yang (2022), 'Prediction of Credit Risk Based on Logistic Regression and Random Forest Technique'

[58]: Kazemi-Arpanahi et al. (2022), 'Using logistic regression to develop a diagnostic model for COVID-19: A single-center study'

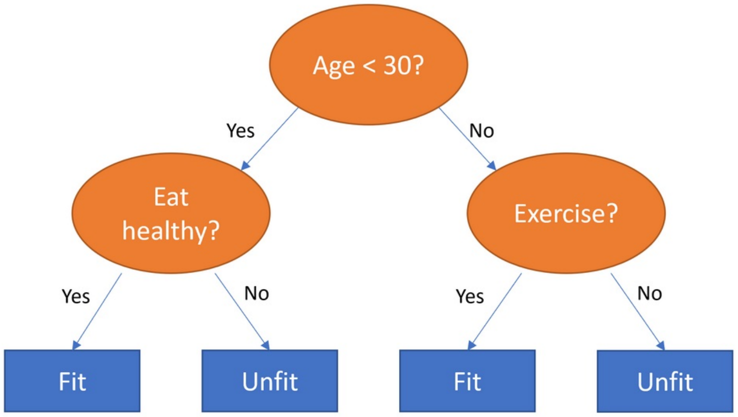


Figure 9.3: Decision Tree

4

4: Can you list the problem domains where Decision Trees are most suitable?

Example 9.2.3 Suppose we have a dataset of customer information, including their age, income, and purchase history.
Objective: We want to create a model that can predict whether a new customer will purchase a certain product based on their age and income.
Proceedure: to do this,

1. We use a decision tree to model the decision-making process of a customer. The decision tree will split the data into branches based on the most important factors that influence the customer's decision to purchase the product.
2. For example, the decision tree might first split the data based on the customer's age, with one branch for customers under 40 and another branch for customers 40 and over. Each branch of the tree will then continue to split the data based on additional factors, such as income or purchase history, until a prediction is made.
3. Once the decision tree is trained, we can use it to predict whether a new customer will purchase the product based on their age and income. To make a prediction, we start at the root node of the decision tree and follow the branches down until we reach a leaf node, which contains the predicted outcome.
4. For example, if the decision tree determines that customers under 40 with an income over $50,000 are likely to purchase the product, then a new customer who is 35 years old and has an income of $60,000 would be predicted to purchase the product.

Decision tree algorithm can be used in various ways in computer engineering. Here are some examples:

- Fault diagnosis [59]: Decision trees can be used to diagnose faults in computer hardware or software. The decision tree can be trained on a dataset of known faults and their symptoms.

[59]: Tiwari et al. (2022), 'A Decision Tree-Based Algorithm for Fault Detection and Section Identification of DC Microgrid'

Once trained, the decision tree can be used to predict the fault based on the symptoms observed.

- Software testing: Decision trees can be used to guide software testing. The decision tree can be trained on a dataset of inputs and expected outputs. Once trained, the decision tree can be used to guide testing by suggesting inputs that are likely to reveal faults.
- Resource allocation [60]: Decision trees can be used to allocate resources in computer systems. For example, a decision tree can be trained to allocate processing power or memory based on the current load and other factors.
- Malware detection [61]: Decision trees can be used to detect malware on a computer. The decision tree can be trained on a dataset of known malware and their characteristics. Once trained, the decision tree can be used to identify new malware based on its characteristics.
- Prediction of system performance [62]: Decision trees can be used to predict the performance of a computer system based on its hardware and software configuration. The decision tree can be trained on a dataset of system configurations and their performance characteristics. Once trained, the decision tree can be used to predict the performance of new configurations.

[60]: Khanli et al. (2011), 'Active rule learning using decision tree for resource management in Grid computing'

[61]: Ullah et al. (2020), 'Modified Decision Tree Technique for Ransomware Detection at Runtime through API Calls'

[62]: Matzavela et al. (2021), 'Decision tree learning through a Predictive Model for Student Academic Performance in Intelligent M-Learning environments'

Random Forest An ensemble learning method that constructs multiple decision trees and combines their predictions to produce a final result.
5

Figure 9.4 is a graphical explanation of random forrest algorithm.

5: Why do we prefer a forest (collection of trees) rather than a single tree to make predictions?

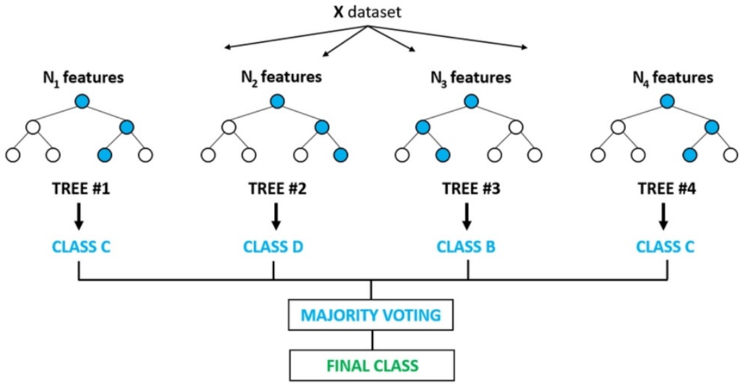


Figure 9.4: Random Forrest

Example 9.2.4 Suppose we have a dataset of customer information, including their age, income, purchase history, and other features. *Objective:* we want to create a model that can predict whether a new customer will purchase a certain product based on these features. *Proceedure:* to do this,

1. We use random forest to build multiple decision trees, where each decision tree is trained on a different subset of the data

and a random subset of the features. This helps to reduce overfitting and improve the accuracy of the model.

2. Once the random forest is trained, we can use it to predict whether a new customer will purchase the product based on their features. To make a prediction, we input the customer's features into each decision tree in the random forest and aggregate the results. The final prediction is then based on the majority vote of the decision trees.
3. For example, suppose the random forest contains 100 decision trees, and 70 of the trees predict that a new customer will purchase the product. Then the final prediction would be that the customer is likely to purchase the product.

The random forest algorithm can be applied to a wide range of problems in computer engineering where accurate predictions or classifications are required. Here are some examples:

- Image recognition [63]: Random forest algorithm can be used to classify images by training on a large dataset of labeled images. Once trained, the algorithm can classify new images with high accuracy.
- Anomaly detection [64, 65]: Random forest algorithm can be used to detect anomalies in system logs, network traffic, or other data sources. The algorithm can be trained on a dataset of normal behavior and can identify unusual patterns or outliers.
- Feature selection [66]: Random forest algorithm can be used to select the most important features in a large dataset. The algorithm ranks features based on their importance in predicting the target variable, allowing engineers to focus on the most relevant features for a particular application.
- Recommender systems [67]: Random forest algorithm can be used to build personalized recommender systems by training on a large dataset of user preferences and item characteristics. Once trained, the algorithm can recommend items to users based on their preferences.
- Predictive maintenance [68]: Random forest algorithm can be used to predict failures in hardware components or systems. The algorithm can be trained on a dataset of sensor data and failure events to identify patterns that lead to failure and predict when failures are likely to occur.

Support Vector Machines (SVMs) A set of algorithms used for classification, regression, and outlier detection. SVMs are particularly useful in high-dimensional spaces and can handle non-linear data.

Figures 9.5 and 9.6 are graphical explanations of support vector machines for two-dimensional and three-dimensional data sets.

Example 9.2.5 Suppose we have a dataset of customer information, including their age, income, and purchase history.
Objective: we want to create a model that can predict whether a new customer will purchase a certain product based on their age and income.

[63]: Bosch et al. (2007), 'Image Classification using Random Forests and Ferns'

[64]: Prashanth et al. (2008), 'Using Random Forests for Network-based Anomaly detection at Active routers'

[65]: Kopp et al. (2020), 'Anomaly explanation with random forests'

[66]: Chen et al. (2020), 'Selecting critical features for data classification based on machine learning methods'

[67]: Zhang et al. (2016), 'Three-way recommender systems based on random forests'

[68]: Kizito et al. (2018), 'The Application of Random Forest to Predictive Maintenance'

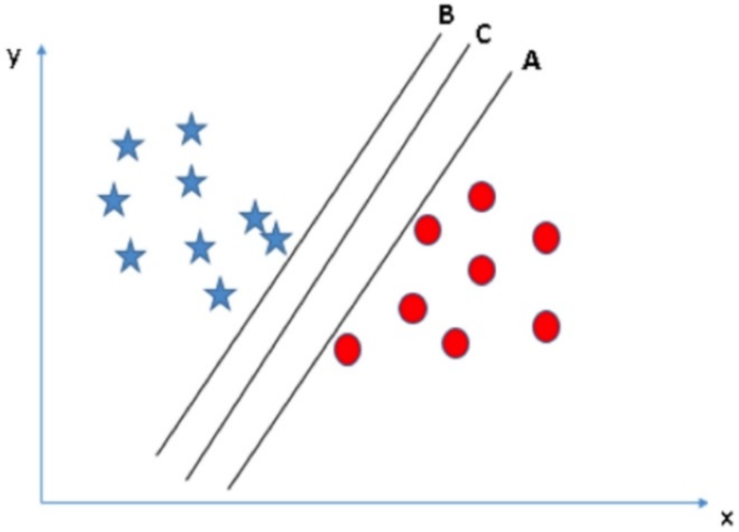


Figure 9.5: Support Vector Machine for 2D Data Set

Proceedure: to do this,

1. We use SVMs to find a hyperplane that maximally separates the data into two classes: customers who are likely to purchase the product and customers who are not likely to purchase the product. The hyperplane is defined by a set of weights that are learned during training.
2. Once the SVM is trained, we can use it to predict whether a new customer will purchase the product based on their age and income. To make a prediction, we input the customer's age and income into the SVM and calculate the hyperplane's distance to the customer's data point. If the distance is positive, the customer is classified as likely to purchase the product. If the distance is negative, the customer is classified as not likely to purchase the product.
3. For example, suppose the SVM determines that customers with an income above $50,000 and age below 40 are likely to purchase the product. Then a new customer who is 35 years old and has an income of $60,000 would be predicted to purchase the product.

SVM is a versatile algorithm that can be applied to a wide range of classification and regression tasks in computer engineering, where accurate predictions or classifications are required. Here are some examples of how SVM can be applied in computer engineering:

- Image classification [69]: SVM can be used for image classification tasks, such as identifying objects or recognizing faces. The algorithm can be trained on a large dataset of images, where each image is labeled with the object it contains or the person it depicts. Once trained, the SVM can classify new images with high accuracy.
- Spam filtering [70]: SVM can be used to filter spam emails by training on a dataset of labeled emails. The algorithm can classify incoming emails as spam or not spam based on their

[69]: Chandra et al. (2021), 'Survey on SVM and their application in imageclassification'

[70]: Torabi et al. (2015), 'Efficient Support Vector Machines for Spam Detection: A Survey'

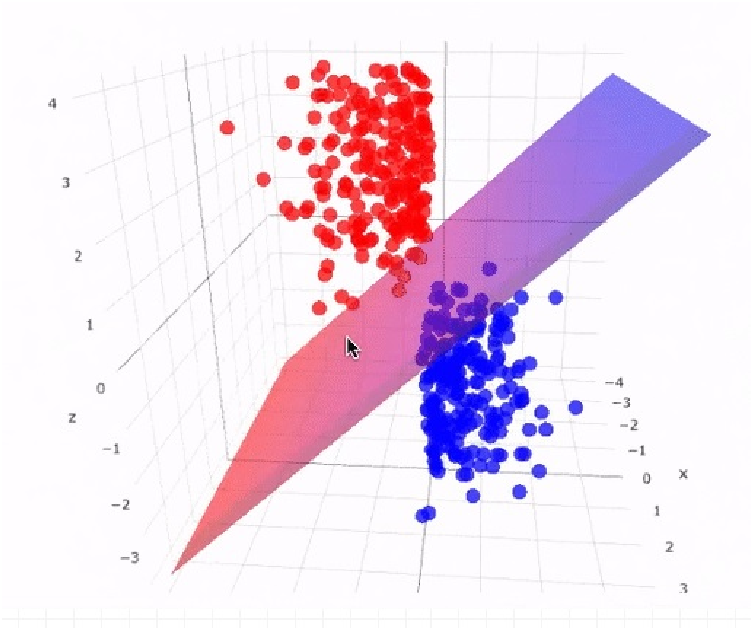


Figure 9.6: Support Vector Machine for 3D Data Set

content, subject, and other features.

- Intrusion detection [71]: SVM can be used to detect intrusions in computer networks by training on a dataset of network traffic. The algorithm can identify patterns in the traffic that indicate a potential intrusion and alert the system administrator.
- Handwriting recognition [72]: SVM can be used for handwriting recognition tasks, such as recognizing handwritten digits or letters. The algorithm can be trained on a large dataset of labeled handwriting samples and can recognize new samples with high accuracy.
- Predictive maintenance [73]: SVM can be used for predictive maintenance tasks, such as predicting when a hardware component or system is likely to fail. The algorithm can be trained on a dataset of sensor data and failure events to identify patterns that lead to failure and predict when failures are likely to occur.

[71]: Li et al. (2012), 'An efficient intrusion detection system based on support vector machines and gradually feature removal method'

[72]: Ahlawat et al. (2020), 'Hybrid CNN-SVM Classifier for Handwritten Digit Recognition'

[73]: Ran et al. (2019), *A Survey of Predictive Maintenance: Systems, Purposes and Approaches*

Neural Networks A set of algorithms that are modeled after the structure of the human brain and can be used for tasks such as image recognition, speech recognition, and natural language processing.

> **Example 9.2.6** Suppose we have a dataset of images of handwritten digits, such as the MNIST dataset. Each image is a 28×28 pixel grayscale image of a handwritten digit (0-9).
> *Objective:* we want to create a model that can classify the images into their respective digits.
> *Proceedure:* to do this,
>
> 1. We use a neural network with multiple layers of artificial neurons. Each neuron in the network receives inputs from the neurons in the previous layer, performs a computation,

and passes its output to the neurons in the next layer. The neurons in the final layer of the network represent the output classes (0-9), and the values of these neurons represent the predicted probabilities of each class.

2. During training, the neural network adjusts the weights and biases of each neuron to minimize the difference between the predicted outputs and the true labels of the training data. This is done using a loss function and an optimization algorithm, such as stochastic gradient descent.
3. Once the neural network is trained, we can use it to classify new images of handwritten digits. To make a prediction, we input the image into the neural network, and the output of the final layer represents the predicted probabilities of each digit class. The class with the highest probability is then selected as the predicted digit.
4. For example, if the neural network predicts that an image of a handwritten digit has a high probability of being a 5, then the digit is classified as a 5.

NN are a versatile tool for solving a wide range of problems in computer engineering, especially when the input data is high-dimensional, noisy, and complex. NN are capable of learning complex relationships between input and output variables and can be used to make accurate predictions and classifications.

- Computer Vision [74]: NN can be used for object recognition, face recognition, image and video processing, and other computer vision tasks. Convolutional Neural Networks (CNN) are a popular type of NN for computer vision tasks.
  Figure 9.7 is a graphical explanation of Convolutional Neural Networks.

[74]: Shastri (2020), *5 Neural network architectures you must know for Computer Vision*

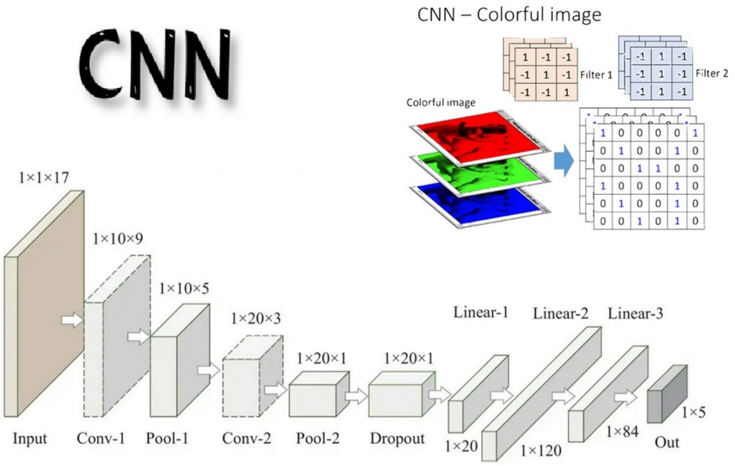


Figure 9.7: Convolutional Neural Network

- Natural Language Processing [75]: NN can be used for natural language processing tasks, such as language translation, text summarization, and sentiment analysis. Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks are popular types of NN for natural language processing tasks.

[75]: Davydova (2017), *7 types of Artificial Neural Networks for Natural Language Processing*

- Speech Recognition [76]: NN can be used for speech recognition tasks, such as speech-to-text conversion and speaker identification. Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are popular types of NN for speech recognition tasks.
- Robotics [77]: NN can be used for robot control, navigation, and vision tasks. NN can learn to recognize objects, navigate through environments, and perform tasks in a way that is similar to human-like decision making.
- Recommender Systems [78]: NN can be used for building personalized recommender systems by training on a large dataset of user preferences and item characteristics. Once trained, the algorithm can recommend items to users based on their preferences.
- Predictive Maintenance [79]: NN can be used to predict when a hardware component or system is likely to fail. NN can be trained on a dataset of sensor data and failure events to identify patterns that lead to failure and predict when failures are likely to occur.

[76]: Papastratis (2021), *Speech Recognition: a review of the different deep learning approaches*

[77]: Pierson et al. (2017), 'Deep learning in robotics: a review of recent research'

[78]: Hassan et al. (2017), 'A Neural Networks Approach for Improving the Accuracy of Multi-Criteria Recommender Systems'

[79]: Bampoula et al. (2021), 'A Deep Learning Model for Predictive Maintenance in Cyber-Physical Production Systems Using LSTM Autoencoders'

K-Nearest Neighbors (KNN) A classification algorithm that compares an unknown data point to the k-nearest known data points to determine its classification.

Figure 9.8 is a graphical explanation of K-Nearest Neighbors algorithm.

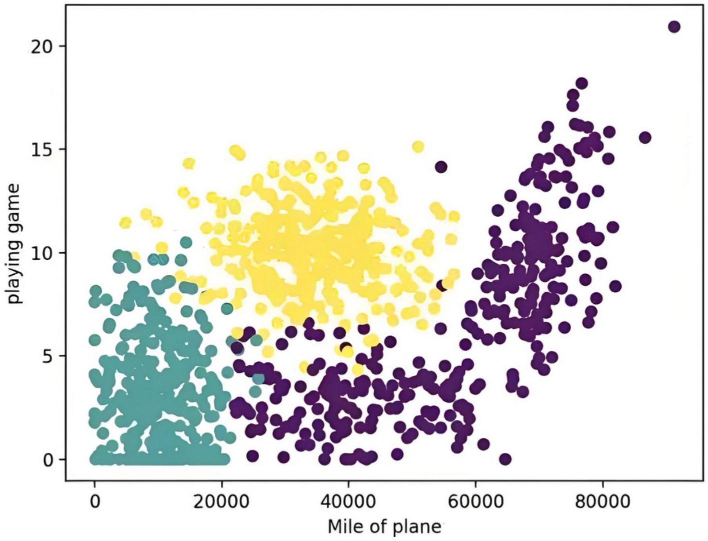


Figure 9.8: K-Nearest Neighbors

Example 9.2.7 Suppose we have a dataset of customer information, including their age, income, and purchase history.
Objective: we want to create a model that can predict whether a new customer will purchase a certain product based on their age and income.
Proceedure: to do this,

1. We use KNN to find the K nearest neighbors of the new customer in the training data, where K is a user-defined parameter. The nearest neighbors are determined by calculating the Euclidean distance between the new customer's data point and the data points in the training data.
2. Once the nearest neighbors are found, we can use their labels (i.e., whether they purchased the product or not) to predict the label of the new customer. One common approach is to use majority voting, where the label with the most occurrences among the K nearest neighbors is selected as the predicted label for the new customer.
3. For example, suppose the KNN algorithm determines that the 5 nearest neighbors of a new customer are all customers who have purchased the product in the past. Then the KNN algorithm would predict that the new customer is likely to purchase the product.

KNN can be used in computer engineering for a variety of tasks, such as:

- Anomaly detection [80]: KNN can be used to detect anomalies in computer systems, such as network traffic or system logs. By comparing the new data points with the K nearest neighbors in the training data, KNN can detect anomalies that deviate significantly from the normal patterns.
- Fault diagnosis [81]: KNN can be used to diagnose faults in computer hardware or software systems. By analyzing the patterns in the data, KNN can identify the root cause of the faults and suggest possible solutions.
- Predictive maintenance [82]: KNN can be used to predict when computer systems are likely to fail, based on their historical performance data. By identifying the K nearest neighbors that have similar performance patterns, KNN can predict when a system is likely to fail and alert the maintenance team.
- Resource allocation [83]: KNN can be used to allocate resources in computer systems, such as CPU time or memory. By analyzing the workload patterns of the K nearest neighbors, KNN can optimize the resource allocation to minimize the response time or maximize the throughput.

Clustering A family of algorithms used to group similar data points together based on certain criteria.

One example is K means. It is an unsupervised machine learning algorithm used for clustering data points. K means organizes the data into clusters with similar characteristics without making predictions. Figure 9.9 is a graphical explanation of K means algorithm.

Example 9.2.8 Suppose we have a dataset of customer transactions, including the amount spent, the date of the transaction, and the type of product purchased.

Objective: we want to group the customers into clusters based on

[80]: Wang et al. (2020), 'A Log-Based Anomaly Detection Method with Efficient Neighbor Searching and Automatic K Neighbor Selection'

[81]: Elshenawy et al. (2022), 'Fault detection and diagnosis strategy based on k-nearest neighbors and fuzzy C-means clustering algorithm for industrial processes'

[82]: Mazzuto et al. (2021), 'Health Indicator for Predictive Maintenance Based on Fuzzy Cognitive Maps, Grey Wolf, and K-Nearest Neighbors Algorithms'

[83]: Chatzigeorgakidis et al. (2018), 'FML-kNN: scalable machine learning on Big Data using k-nearest neighbor joins'

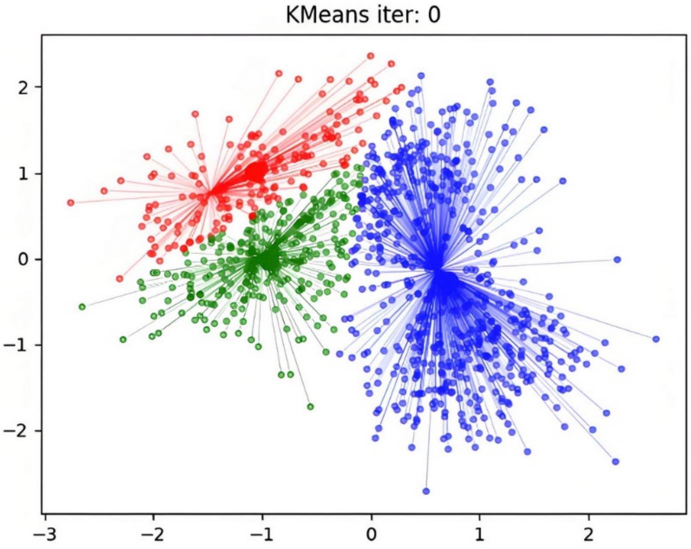


Figure 9.9: K Means

their purchasing behavior, so that we can target them with more personalized marketing campaigns.
Proceedure: to do this,

1. We use clustering algorithms such as K-Means or Hierarchical Clustering. These algorithms work by iteratively grouping data points together based on their similarity, until the clusters are formed.
2. In the case of customer transactions, we can use clustering to group customers based on their purchasing behavior, such as those who purchase similar products or those who purchase at similar times. By clustering the customers, we can identify patterns and segment them into different groups, each with a distinct profile and behavior.
3. For example, we might find that one cluster consists of customers who purchase mostly high-end products, while another cluster consists of customers who purchase mostly low-end products. we can then target each cluster with different marketing campaigns that appeal to their specific interests and preferences.

Clustering can be used in various ways in computer engineering, including:

- Network traffic analysis [84]: Clustering can be used to identify patterns and anomalies in network traffic. By clustering network traffic based on its flow features, such as source IP address, destination IP address, port numbers, and protocols, it is possible to identify network clusters that share similar communication patterns. This can help in detecting network attacks, traffic congestion, and performance issues.
- Hardware and software profiling [85]: Clustering can be used to identify the usage patterns of hardware and software systems. By clustering system data, such as CPU usage, memory usage,

[84]: Zhang et al. (2022), 'Cluster analysis of day-to-day traffic data in networks'

[85]: Panapakidis et al. (2018), 'Optimal Selection of Clustering Algorithm via Multi-Criteria Decision Analysis (MCDA) for Load Profiling Applications'

disk I/O, and network traffic, it is possible to identify clusters that share similar performance characteristics. This can help in optimizing system resource usage, diagnosing performance issues, and predicting system failures.

- Image and video analysis [86]: Clustering can be used to segment images and videos into meaningful regions. By clustering pixels or regions based on their color, texture, and spatial features, it is possible to identify objects, scenes, and activities in images and videos. This can help in tasks such as object recognition, image segmentation, and video summarization.
- Software fault localization [87]: Clustering can be used to identify the root cause of software faults. By clustering program execution traces based on their features, such as function calls, variable values, and execution paths, it is possible to identify clusters of traces that share similar error patterns. This can help in localizing the fault to specific parts of the program and suggesting possible fixes.

Gradient Boosting A machine learning technique that involves combining multiple weak models to create a single, more accurate predictive model.

Figure 9.10 is a graphical explanation of gradient boosting algorithm.

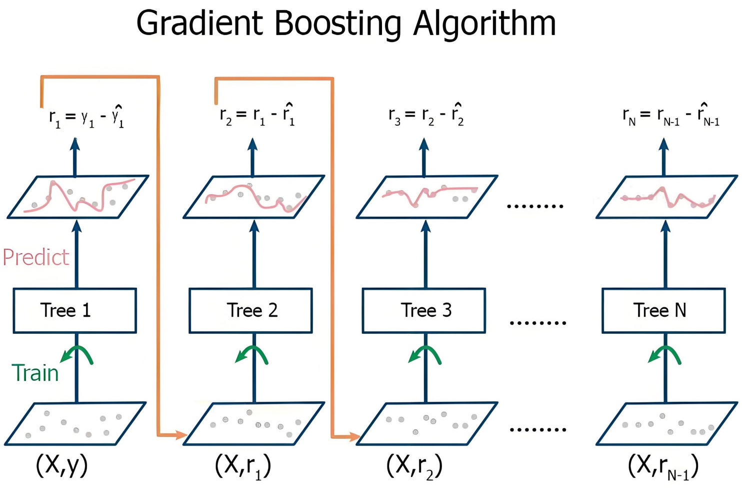


Figure 9.10: Gradient Boosting

Example 9.2.9 Suppose we have a dataset of customer transactions, including the amount spent, the date of the transaction, and the type of product purchased.
Objective: we want to predict which customers are likely to make a repeat purchase in the next month, based on their past transaction history.
Proceedure: to do this,

1. We use Gradient Boosting algorithms such as XGBoost [88] or LightGBM [89]. These algorithms work by iteratively adding decision trees to the model, each of which predicts the residual

[86]: Mittal et al. (2022), 'A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets'

[87]: Wu et al. (2020), 'FATOC: Bug Isolation Based Multi-Fault Localization by Using OPTICS Clustering'

[88]: Wikipedia contributors (2023), *XGBoost — Wikipedia, The Free Encyclopedia*

[89]: Wikipedia contributors (2023), *LightGBM — Wikipedia, The Free Encyclopedia*

errors of the previous tree.

2. In the case of customer transactions, we can use Gradient Boosting to create a model that predicts the likelihood of a customer making a repeat purchase based on their past transaction history. By using features such as the amount spent, the date of the transaction, and the type of product purchased, the model can learn patterns in the data and make accurate predictions.
3. For example, the model might learn that customers who made a purchase within the last week are more likely to make a repeat purchase in the next month, or that customers who purchased high-end products are more loyal than those who purchased low-end products.

Gradient Boosting can be used in various ways in computer engineering, including:

- Anomaly detection in system logs [90]: Gradient Boosting can be used to detect anomalies in system log data, such as event logs and error logs. By using Gradient Boosting to learn the patterns in the log data, it is possible to identify anomalous log entries that indicate system failures, security breaches, or other issues.
- Predictive maintenance in industrial equipment [91]: Gradient Boosting can be used to predict when industrial equipment is likely to fail. By using Gradient Boosting to learn the patterns in sensor data from the equipment, it is possible to predict when maintenance is needed, reducing downtime and maintenance costs.
- Network intrusion detection [92]: Gradient Boosting can be used to detect network intrusions by learning the patterns of normal network traffic and identifying deviations from those patterns. By using Gradient Boosting to analyze network traffic data, it is possible to detect anomalies that indicate potential intrusions.
- Image and speech recognition [93]: Gradient Boosting can be used to recognize images and speech. By using Gradient Boosting to learn the patterns in images or speech data, it is possible to recognize objects, faces, and speech patterns.

These are just a few examples of the many AI algorithms that are used in various applications of artificial intelligence. Different algorithms are suitable for different tasks and datasets, and researchers and developers continue to explore new algorithms and techniques to improve the capabilities of AI systems.

9.3 Design Optimization

One of the primary applications of AI in computer engineering is design optimization. Design optimization is the process of finding the most efficient design for a particular component or system. This is an important process

[90]: Louk et al. (2022), 'Revisiting Gradient Boosting-Based Approaches for Learning Imbalanced Data: A Case of Anomaly Detection on Power Grids'

[91]: Roque et al. (2022), 'An analysis of machine learning algorithms in rotating machines maintenance'

[92]: Douiba et al. (2022), 'Anomaly detection model based on gradient boosting and decision tree for IoT environments security'

[93]: Dash et al. (2022), 'Gradient boosting machine and efficient combination of features for speech-based detection of COVID-19'

because it can significantly reduce the cost and time required to develop new computer systems.

AI can be used to optimize the design of computer chips and other hardware components. Machine learning algorithms can be used to analyze large amounts of data to identify the most efficient designs. These algorithms can learn from previous designs and identify patterns that indicate areas where improvements can be made.

For example, AI can be used to optimize the design of computer processors. By analyzing the performance of different processor architectures and identifying patterns in the data, machine learning algorithms can identify the most efficient design. This can lead to significant improvements in performance, power consumption, and cost.

Here are some ways AI is used in design optimization in computer engineering:

- Parameter tuning [94]: AI algorithms can be used to automatically adjust the values of system parameters, such as clock speeds or memory timings, to optimize performance or reduce power consumption.
- Prediction and simulation [95]: AI can be used to predict the behavior of complex systems, such as integrated circuits or networks, and to simulate the effects of different design choices.
- Design synthesis [96]: AI can be used to generate new design solutions based on performance criteria and design constraints.
- Failure prediction [97]: AI can be used to predict potential system failures based on patterns in data from sensors or other sources.
- System optimization [98]: AI can be used to optimize the overall system design, taking into account interactions between different components and subsystems.

[94]: Brown (2023), *AI for Circuit Design Quality, Productivity, and Advanced-Node Mapping*

[95]: Jenis et al. (2023), 'Engineering Applications of Artificial Intelligence in Mechanical Design and Optimization'

[96]: Castro Pena et al. (2021), 'Artificial intelligence applied to conceptual design. A review of its use in architecture'

[97]: Meyes et al. (2021), 'Transparent and Interpretable Failure Prediction of Sensor Time Series Data with Convolutional Neural Networks'

[98]: Habeeb et al. (2023), 'Design Optimization Method Based on Artificial Intelligence (Hybrid Method) for Repair and Restoration Using Additive Manufacturing Technology'

9.4 Quality Control

Another important application of AI in computer engineering is quality control. Quality control is the process of ensuring that computer components and systems are free of defects or other quality issues. This is important because defects or quality issues can lead to system failures or other problems.

AI can be used to monitor the manufacturing process for computer components and identify defects or other quality issues. Machine learning algorithms can analyze data from sensors and other sources to detect patterns that indicate problems. This can help engineers identify the root cause of quality issues and take corrective action to address them.

For example, AI can be used to monitor the manufacturing process for computer chips. By analyzing data from sensors that monitor temperature, humidity, and other factors, machine learning algorithms can detect patterns that indicate defects in the manufacturing process. This can help engineers identify the root cause of defects and take corrective action to prevent them from occurring in the future.

Here are some ways in which AI is commonly used in quality control:

- Automated Visual Inspection [99]: AI-powered computer vision systems can analyze images and videos to identify defects or anomalies in products. This is particularly useful in industries like manufacturing, where products are visually inspected for flaws.
- Defect Detection and Classification [100]: AI algorithms can be trained to detect and classify defects in products based on visual inspection. This includes identifying scratches, dents, color variations, and other imperfections.
- Predictive Maintenance [101]: AI can analyze data from sensors and equipment to predict when machines or components are likely to fail. This proactive approach helps in scheduling maintenance activities before a failure occurs, reducing downtime and improving overall product quality.
- Statistical Process Control (SPC) [102]: AI can be applied to analyze data from various stages of the production process, helping to identify patterns and trends that may indicate potential quality issues. This enables real-time adjustments to maintain consistent quality.
- Natural Language Processing (NLP) for Text Analysis [103]: In industries where quality control involves analyzing textual data, such as customer feedback or product specifications, NLP can be used to extract valuable insights and identify areas for improvement.
- Machine Learning for Process Optimization [104]: Machine learning algorithms can analyze historical data to optimize manufacturing processes for better quality outcomes. This includes adjusting parameters to minimize defects and enhance overall efficiency.
- Supply Chain Monitoring [105]: AI can be used to monitor and assess the quality of raw materials and components in the supply chain. This ensures that only high-quality inputs are used in the manufacturing process.
- Automated Testing [106]: AI-driven automated testing processes can rapidly and accurately test software applications, ensuring that they meet quality standards and perform as expected.
- Robotics in Quality Inspection [107, 108]: Robots equipped with AI systems can perform precise and repetitive inspection tasks, ensuring that products meet specific quality criteria without human error.
- Data Analytics for Quality Improvement [109]: AI tools can analyze large datasets to identify patterns and correlations that may not be immediately apparent. This information can be used to continuously improve processes and enhance product quality.

9.5 Predictive Maintenance

Predictive maintenance is another important application of AI in computer engineering. Predictive maintenance is the process of predicting when computer systems are likely to fail or need maintenance. This is important because it can help prevent system failures and reduce downtime.

[99]: Trajkova et al. (2021), ‘Active Learning for Automated Visual Inspection of Manufactured Products’

[100]: Westphal et al. (2021), ‘A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks’

[101]: Kamel (2022), ‘Artificial intelligence for predictive maintenance’

[102]: Boaventura et al. (2022), ‘On flexible Statistical Process Control with Artificial Intelligence: Classification control charts’

[103]: Dataquest (2022), *Using Machine Learning and Natural Language Processing Tools for Text Analysis*

[104]: Walas Mateo et al. (2021), ‘Artificial Intelligence and Machine Learning as a Process Optimization driver under Industry 4.0 framework, the role of the people in the process’

[105]: Pournader et al. (2021), ‘Artificial intelligence applications in supply chain management’

[106]: Fatima et al. (2022), ‘Automated Testing with Machine Learning Frameworks: A Critical Analysis’

[107]: Hussain et al. (2023), ‘Robotics and Automation with Artificial Intelligence: Improving Efficiency and Quality’

[108]: Rayhan (2023), ‘Artificial Intelligence In Robotics: From Automation To Autonomous Systems’

[109]: Ciora et al. (2016), ‘Quality Improvement Based on Big Data Analysis’

AI can be used to predict when computer systems are likely to fail or need maintenance. Machine learning algorithms can analyze data from sensors and other sources to identify patterns that indicate potential problems. This can help technicians take preventive action before a failure occurs.

For example, AI can be used to predict when hard drives are likely to fail. By analyzing data from sensors that monitor temperature, vibration, and other factors, machine learning algorithms can identify patterns that indicate when a hard drive is likely to fail. This can help technicians replace the hard drive before it fails, reducing the risk of data loss and downtime.

9.6 Cybersecurity

AI is also an important tool for cybersecurity in computer engineering. Cybersecurity is the process of protecting computer systems from cyber threats, such as malware, viruses, and hackers. This is important because cyber threats can lead to data breaches, system failures, and other problems.

AI can be used to detect and respond to cyber threats. Machine learning algorithms can analyze network traffic, identify patterns that indicate malicious activity, and take action to block or contain the threat. This can help prevent cyber attacks and reduce the impact of those that do occur.

Below are some research topics that can be found in literature:

- Threat Detection and Prevention, including anomaly detection (identify anomalies that may indicate a potential security threat), signature-based detection (recognize known malware and malicious patterns in code or network traffic), and heuristic-based detection (recognize new and previously unknown threats by identifying patterns that deviate from normal behavior).
- Behavioral Analysis: to monitor and analyze user and entity behavior to identify deviations from normal patterns, helping to detect insider threats and compromised accounts.
- Endpoint Security: to identify and block malicious activities on individual devices, offering real-time protection against malware and other threats.
- Phishing Detection: to analyze emails and messages to detect phishing attempts by identifying suspicious links, content, or patterns in communication.
- Vulnerability Management: to scan systems for vulnerabilities and prioritizing them based on the level of risk, helping organizations address critical issues first.
- Incident Response and Forensics: to automate and improve incident response by quickly analyzing vast amounts of data to identify the source and extent of a security incident.
- User Authentication: to enhance user authentication processes through biometrics, behavioral analysis, and multi-factor authentication, making it more difficult for unauthorized users to gain access.

- Security Analytics: to process and analyze large datasets to identify patterns, trends, and potential security threats that may be challenging for humans to discern.
- Security Information and Event Management (SIEM): to enhance SIEM systems by automating the analysis of security alerts, helping organizations respond to incidents more efficiently.
- Network Security: to identify and respond to abnormal network traffic, helping to prevent and mitigate cyberattacks.
- Machine Learning in Antivirus Software: to adapt and learn from new threats over time, improving their effectiveness.
- Automated Threat Hunting: to continuously scan networks for potential threats and vulnerabilities.
- Deep Packet Inspection: to analyze network traffic at a granular level, helping to identify and block malicious activities.

9.7 Intelligent Automation

Finally, AI can be used to automate many routine tasks in computer engineering. Intelligent automation is the process of automating routine tasks using AI and machine learning. This is important because it can reduce the workload on engineers and technicians and allow them to focus on more complex tasks.

AI can be used to automate many routine tasks in computer engineering, such as software testing and system monitoring. Machine learning algorithms can analyze data from sensors and other sources to identify patterns that indicate potential problems or opportunities for optimization. This can help engineers and technicians focus on more complex tasks and improve the efficiency and reliability of computer systems.

For example, AI can be used to automate software testing. Machine learning algorithms can analyze data from previous software tests to identify patterns that indicate potential problems. This can help engineers identify potential problems before they occur and take corrective action to prevent them from occurring in the future.

The use of AI seems unlimited. A generative model for protein design has been published [110] recently.

[110]: Ingraham et al. (2023), 'Illuminating protein space with a programmable generative model'

9.8 AI in Education

ALthough there are ethical considerations and data privacy concerns, AI is able to enhance learning experience for students, support instructors, and improve overall educational outcomes. AI has the potential to revolutionize education in the following areas: personalized learning [111], adaptive learning platforms [112], intelligent tutoring systems, automated grading and assessment, language processing and translation, virtual reality (VR) and augmented reality (AR), predictive analytics, automated administrative tasks, education chatbots, and professional development for educators.

[111]: Bhutoria (2022), 'Personalized education and Artificial Intelligence in the United States, China, and India: A systematic review using a Human-In-The-Loop model'

[112]: Small (2023), *The Top 10 Adaptive Learning Platforms of 2023*

For further reading, here are some interesting references, including books [113], articles [114, 115], report [116], and websites [117, 118].

9.9 Summary

In conclusion, AI has become an increasingly important tool for computer engineering. It has the potential to revolutionize the way computer systems are designed, developed, and maintained. The various applications of AI in computer engineering, including design optimization, quality control, predictive maintenance, cybersecurity, and intelligent automation, can help engineers and technicians work more efficiently and effectively. As AI continues to develop and evolve, it is likely that it will become an even more integral part of the field of computer engineering.

[113]: Kosslyn (2023), *Active Learning with AI: A Practical Guide*

[114]: Ghai et al. (2021), 'Explainable Active Learning (XAL): Toward AI Explanations as Interfaces for Machine Teachers'

[115]: Agarwal et al. (2021), *Addressing practical challenges in Active Learning via a hybrid query strategy*

[116]: Cardona et al. (2023), *Artificial Intelligence and the Future of Teaching and Learning*

[117]: Gifford (2023), *Educators Roundtable: Demystify AI With Transparency and Practice*

[118]: Keelor (2023), *The power of AI in the classroom*

10 Cybersecurity

10.1 Introduction

Cybersecurity is a complex and rapidly evolving field that encompasses various aspects of protecting computer systems, networks, and electronic devices from unauthorized access, attack, or damage. The importance of cybersecurity has become more critical than ever before as technology continues to advance, and cybercriminals become increasingly sophisticated in their attacks. In this chapter, we will discuss the different types of cyber threats, their impact on individuals and organizations, the measures that can be taken to prevent them, and the challenges faced in the cybersecurity field.

10.2 Types of Cyber Threats

The National Institute of Standards and Technology (NIST) defines cyber threats in its NIST Secial Publication 1800-15 [119] as “any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability".

Cyber threats can be classified into various types, including the following:

1. Malware: malicious software designed to damage, disrupt, or gain unauthorized access to a computer system or network. Malware includes viruses, worms, Trojan horses, and ransomware. Malware is usually spread through email attachments, infected websites, or software downloads. Explore more details, including vieos, at Cisco [120].

[119]: Dodson et al. (2021), *Securing Small-Business and Home Internet of Things (IoT) Devices: Mitigating Network-Based Attacks Using Manufacturer Usage Description (MUD)*

[120]: Cisco (2023), *What is malware?*

2. Phishing: a type of cyber-attack that involves tricking individuals into revealing sensitive information such as usernames, passwords, or credit card numbers. Phishing attacks usually involve an email or a text message that appears to be from a legitimate source, but in reality, it is a fake message designed to steal information. Explore more details at Phishing.org [121].
3. DoS attack: an attempt to overwhelm a computer system or network with traffic, rendering it inaccessible to users. The attacker achieves this by flooding the network with traffic from multiple sources, thereby causing the system to crash or become unavailable. Read the Denial of Service (DoS) guidance from National Cyber Security Centre[122].
4. SQL injection attacks: attacks that exploit vulnerabilities in web applications to gain unauthorized access to databases. The attacker enters malicious code into a web form or query string, which the application processes and executes, allowing the attacker to extract data from the database or modify it. Further reading can be found on Cisco Security [123].
5. Social engineering: a technique used by cybercriminals to trick individuals into revealing sensitive information or performing actions that are not in their best interests. Social engineering attacks can involve phishing, pretexting, baiting, or tailgating. Find more reading from IBM [124].
6. APTs: attacks that involve a long-term and targeted approach to gain unauthorized access to a computer system or network. APTs are usually carried out by sophisticated attackers who aim to steal sensitive information or intellectual property. Refer to Cybersecurity & Infrastructure Security Agency (CISA) [125] for further understanding.

10.3 Impact of Cyber Threats

Cyber threats have a significant impact on individuals and organizations. Some of the impacts of cyber threats include:

- Financial Losses
  Cybercriminals can steal money from individuals or organizations through fraudulent activities, such as phishing or hacking into bank accounts.
- Damage to Reputation
  A cyber-attack can damage an organization's reputation, resulting in a loss of trust from customers or stakeholders.
- Data Loss or Theft
  Data loss or theft can occur when cybercriminals gain unauthorized access to a computer system or network, resulting in the loss of sensitive information.
- Disruption of Business Operations
  A cyber-attack can disrupt an organization's business operations, leading to loss of productivity, revenue, or customer trust.

[121]: Phishing.org (2023), *What Is Phishing?*

[122]: NCSC (2023), *Denial of Service (DoS) guidance*

[123]: Cisco (2023), *Understanding SQL Injection*

[124]: IBM (2023), *What is Social Engineering?*

[125]: CISA (2023), *Advanced Persistent Threats and Nation-State Actors*

- Legal Liability
  An organization can be held legally liable for any data breaches or cyber-attacks that occur, resulting in legal fees, fines, or penalties.

10.4 Preventing Cyber Threats

Preventing cyber threats requires a combination of technical and non-technical measures. Some of the measures that can be taken to prevent cyber threats include:

- Regular Software Updates
  Regular software updates can help protect against known vulnerabilities and reduce the risk of cyber-attacks.
- Use of Strong Passwords
  Strong passwords that are difficult to guess or crack can help protect against unauthorized access to computer systems or networks.
- Use of Encryption Encryption is the process of converting plain text into ciphertext, which is unreadable without a key or password. Encryption can help protect sensitive information from being accessed or intercepted by unauthorized parties.
- Implementation of Firewalls and Antivirus Software
  Firewalls and antivirus software can help detect and prevent malicious activities from entering a computer system or network.
- Training and Awareness Programs
  Training and awareness programs can help individuals and organizations recognize and respond to cyber threats effectively.
- Implementing Multi-Factor Authentication
  Multi-factor authentication (MFA) adds an additional layer of security by requiring users to provide two or more forms of identification before gaining access to a computer system or network.

10.5 Challenges in Cybersecurity

The cybersecurity field faces various challenges that make it difficult to protect against cyber threats effectively. Some of the challenges include:

- Rapidly Evolving Threat Landscape
  The threat landscape is constantly evolving, and cybercriminals are becoming more sophisticated in their attacks. This makes it challenging to keep up with the latest threats and implement effective security measures.
- Shortage of Skilled Cybersecurity Professionals
  There is a shortage of skilled cybersecurity professionals, making it challenging for organizations to find and hire individuals with the necessary skills and expertise to protect against cyber threats.

- Lack of Standardization
  There is a lack of standardization in the cybersecurity field, making it challenging to ensure that security measures are implemented consistently across different organizations and industries.
- Legacy Systems
  Many organizations continue to use legacy systems that are not designed to handle modern security threats. Upgrading these systems can be expensive and time-consuming, making it challenging for organizations to keep up with the latest security standards.
- Human Error
  Human error remains one of the most significant challenges in cybersecurity. Employees may inadvertently click on malicious links or disclose sensitive information, leaving organizations vulnerable to cyber-attacks.

10.6 Summary

In conclusion, cybersecurity is an essential aspect of protecting computer systems, networks, and electronic devices from cyber threats. The different types of cyber threats include malware, phishing, denial of service attacks, SQL injection attacks, social engineering, and advanced persistent threats. Cyber threats have a significant impact on individuals and organizations, including financial losses, damage to reputation, data loss or theft, disruption of business operations, and legal liability. Preventing cyber threats requires a combination of technical and non-technical measures, including regular software updates, use of strong passwords, implementation of firewalls and antivirus software, training and awareness programs, and multi-factor authentication. However, the cybersecurity field faces various challenges, including a rapidly evolving threat landscape, shortage of skilled cybersecurity professionals, lack of standardization, legacy systems, and human error. Addressing these challenges will be crucial in ensuring effective cybersecurity measures are implemented to protect against cyber threats.

11 Engineering Standards and Constraints

11.1 Introduction

Students in computer engineering must demonstrate that they have considered various constraints in the design portion of the curriculum. These include, in part, the following:

- Engineering Codes and Standards
- Economic Factors
- Environmental Effects
- Sustainability
- Manufacturability (Constructability)
- Ethical Considerations
- Health and Safety Issues
- Social Ramifications
- Political Factors
- Legal Issues

11.2 Engineering Standards

Engineering standards are a set of guidelines and specifications that define best practices, quality requirements, and safety measures for engineering activities. These standards are developed by international, national, and regional organizations, and they provide a framework for engineers to design, build, and operate safe and reliable products and systems. Here are some examples of engineering standards:

- IEEE Standards: The Institute of Electrical and Electronics Engineers (IEEE) develops and publishes standards for electrical and electronics engineering, including standards for software engineering, telecommunications, and power systems.

- IEC standards: The International Electrotechnical Commission (IEC) develops and publishes international standards for a wide range of topics related to electrical, electronic, and related technologies. These standards are used by engineers, manufacturers, and regulators around the world to ensure that products and systems are safe, reliable, and effective.
- ISO Standards: The International Organization for Standardization (ISO) develops and publishes international standards for a wide range of industries, including engineering. ISO standards cover topics such as quality management, environmental management, and occupational health and safety.
- ANSI Standards: The American National Standards Institute (ANSI) develops and publishes national standards for various industries, including engineering. ANSI standards cover topics such as electrical safety, building codes, and engineering drawings.
- ASME Standards: The American Society of Mechanical Engineers (ASME) develops and publishes standards for mechanical engineering, including codes and standards for pressure vessels, boilers, and piping systems.
- ASTM Standards: The American Society for Testing and Materials (ASTM) develops and publishes standards for materials testing and engineering, including standards for metals, plastics, and construction materials.

Adhering to engineering standards is important for ensuring the safety, reliability, and quality of engineering products and systems. Standards provide a common language and framework for engineers to communicate and collaborate, and they help to promote innovation and improve efficiency in engineering practices.

Engineering standards play an important role in computer engineering by providing guidelines for the design, implementation, and testing of hardware and software systems. These standards help to ensure that computer systems are safe, reliable, and effective, and that they can be used in a wide range of applications and environments. There are several engineering standards that are relevant to computer engineering. Here are a few examples:

- IEEE 488 - General Purpose Interface Bus (GPIB): This standard specifies the electrical and mechanical characteristics of a digital communications bus that allows communication between various electronic devices. GPIB is commonly used in laboratory and industrial environments for controlling instrumentation and measurement equipment.
- IEEE 802 - LAN/MAN Standards: This family of standards covers a wide range of topics related to local area networks (LANs) and metropolitan area networks (MANs), including wireless LANs, Ethernet, and broadband access. These standards provide guidelines for the design, implementation, and interoperability of network equipment and protocols.
- IEEE 754: This standard defines the floating-point arithmetic used by most modern computers. It specifies how numbers are represented

in binary and how arithmetic operations should be performed.

- IEEE 1394: This standard, also known as FireWire, specifies a high-speed serial bus for connecting devices such as cameras and hard drives to computers.
- IEEE 1588: This standard defines a protocol for synchronizing clocks in a computer network. It is commonly used in industrial automation and control systems.
- IEEE 1687: This standard, also known as the Internal JTAG (IEEE 1149.1) Extension for Embedded Cores, defines a standard for accessing and testing embedded cores in integrated circuits.
- IEEE 1800: This standard defines the SystemVerilog hardware description language, which is widely used for designing and verifying digital systems.
- IEEE 1901: This standard defines a high-speed powerline communication (PLC) system for home networking.
- ISO/IEC 27001 - Information Security Management: This standard specifies a framework for establishing, implementing, maintaining, and continually improving an information security management system. It provides a systematic approach to managing sensitive company information and mitigating security risks.
- ISO/IEC 12207 - Software Lifecycle Processes: This standard defines a framework for software development processes. It provides a systematic approach to managing the development, testing, and maintenance of software systems.
- IEC 62304 - Medical Device Software: This standard specifies requirements for the software lifecycle processes of medical devices. It provides a framework for developing safe and effective software for medical devices, including requirements for design, testing, and documentation.
- IEC 60601 - Medical electrical equipment: This standard specifies general safety and performance requirements for medical electrical equipment.
- IEC 61850 - Communication networks and systems for power utility automation: This standard specifies the communication protocols and data models for power utility automation systems.
- IEC 61508 - Functional safety of electrical, electronic, programmable electronic safety-related systems. This standard specifies the requirements for ensuring the functional safety of electrical, electronic, and programmable electronic safety-related systems.
- NIST SP 800-53: This is a standard for information security that provides a catalog of security controls for federal information systems and organizations.
- NIST SP 800-171: This is a set of guidelines for protecting Controlled Unclassified Information (CUI) in nonfederal information systems and organizations.
- NIST SP 800-88: This is a standard for media sanitization that provides guidelines for securely erasing data from storage devices such as hard drives and flash drives.
- NIST SP 800-30: This is a standard for risk management that provides guidelines for identifying, assessing, and mitigating risks to

information and information systems.

- NIST SP 800-37: This is a standard for the Risk Management Framework (RMF) that provides guidelines for managing cybersecurity risk in federal information systems.
- NIST SP 800-171A: This is a companion document to NIST SP 800-171 that provides guidelines for assessing compliance with the security requirements outlined in that standard.
- NIST SP 800-63: This is a set of guidelines for digital identity and authentication that provides recommendations for identity proofing, registration, and authentication.
- PCI DSS (Payment Card Industry Data Security Standard): This standard specifies the requirements for securing credit card data. It is intended to protect the privacy and security of cardholder information and ensure that merchants and service providers comply with industry standards.
- Common Criteria: This is an international standard for evaluating and certifying the security of information technology products. It provides a framework for evaluating the security features of software and hardware products.
- DO-178C: This is a software engineering standard for the development of safety-critical avionics systems. It specifies the requirements for software development, verification, and validation for systems that are critical to flight safety.

11.3 Contraints

Design constraints in computer engineering projects are limitations or requirements that affect the design and development of the project. They may include technical, financial, environmental, or social constraints that must be considered during the design process.

Design constraints play a critical role in shaping the design and development of computer engineering projects. Computer engineers must carefully consider these constraints and balance them against the project's goals and objectives to ensure a successful outcome.

Here are some common design constraints in computer engineering projects:

- Technical constraints: These are limitations related to the technology used in the project, such as hardware and software limitations, compatibility issues, and performance requirements.
- Time constraints: These are limitations related to the timeline of the project, such as deadlines for completion, project milestones, and delivery schedules.
- Budget constraints: These are limitations related to the financial resources available for the project, such as the cost of materials, equipment, and labor.

- Environmental constraints: These are limitations related to the physical environment in which the project will be used, such as temperature, humidity, and electromagnetic interference.
- Physical constraints: These include factors such as the size, weight, and power consumption of the product, as well as the available space for installation.
- Scalability constraints: These include factors such as the ability to expand and scale the system as the needs of the user or market change.
- Legal and regulatory constraints: These are limitations related to laws and regulations that govern the design and use of the project, such as safety standards, privacy regulations, and intellectual property laws.
- Usability constraints: These are limitations related to the user experience and user interface of the project, such as ease of use, accessibility, and user feedback.
- Social and cultural constraints: These are limitations related to the social and cultural context in which the project will be used, such as language barriers, cultural norms, and ethical considerations.

The constraints can be very versatile. For example, in a particular computer engineering senior design project, the floowing constraints might be applied: potential design constraints that could apply to a computer engineering project:

- Processor speed and performance requirements
- Memory and storage limitations
- Compatibility with existing systems and protocols
- Required input/output interfaces and connectivity options
- User interface design and usability requirements
- Power consumption and battery life limitations
- Temperature and environmental operating range constraints
- Regulatory and safety compliance requirements
- Physical size, weight, and form factor limitations
- Cost of components and materials
- Budget constraints and resource availability
- Development timeline and time-to-market requirements
- Software and firmware development constraints
- Network bandwidth and latency limitations
- Security and encryption requirements
- Availability and reliability requirements
- Durability and resistance to physical damage or environmental stress
- Scalability and future-proofing considerations
- Noise and electromagnetic interference requirements
- Localization and language support requirements

12 Future of Computer Engineering

In general, it is likely that computer engineering will continue to grow and evolve as technology advances. Here are some specific areas that may see significant developments in the coming years:

Artificial intelligence: AI is already transforming many industries, and it is likely to continue to do so in the future. Computer engineers will play a critical role in developing and improving AI algorithms, as well as building the hardware and software infrastructure needed to support AI systems.

Quantum computing: Quantum computing is a rapidly developing field that has the potential to revolutionize computing by solving problems that are currently intractable with classical computers. Computer engineers will be instrumental in developing the hardware and software needed to build practical quantum computers.

Internet of Things (IoT): The IoT refers to the growing network of interconnected devices, sensors, and machines that are becoming increasingly prevalent in our homes, workplaces, and communities. Computer engineers will be needed to design and develop the hardware and software necessary to connect and manage these devices.

Cybersecurity: As more of our lives move online, cybersecurity will become an increasingly critical concern. Computer engineers will play a vital role in developing secure systems and technologies to protect against cyber threats.

Overall, the future of computer engineering is likely to be exciting and full of opportunities for innovation and growth.

Appendix

A

About AAAA

A.1 aaaa

A.2 bbbb

B

About BBBB

B.1 aaaa

B.2 bbbb

Bibliography

Here are the references in citation order.

[1] Clemson University. *Evidence Based Teaching Strategies*. [Online; accessed 08-April-2023]. 2023. URL: https://www.clemson.edu/otei/evidence-based.html (cited on page v).

[2] ACUE. *Inclusive Online Teaching Webinars*. [Online; accessed 13-November-2023]. 2023. URL: https://acue.org/inclusive-online-teaching-webinars/ (cited on page vii).

[3] ASEE. *Webinar Series*. [Online; accessed 13-November-2023]. 2023. URL: https://edge.asee.org/webinar-series/ (cited on page vii).

[4] Robert W. Doran. 'The Gray Code'. In: *J. Univers. Comput. Sci.* 13 (2007), pp. 1573–1597 (cited on page 26).

[5] Intel. *Student Handout: ASCII Computer Code*. [Online; accessed 30-November-2023]. 2010. URL: https://www.intel.com/content/dam/www/program/education/us/en/documents/the-journery-inside/digital/tji-digital-info-handout4.pdf (cited on page 26).

[6] William L. Hosch. *supercomputer*. [Online; accessed 21-November-2023]. Oct. 2023. URL: https://www.britannica.com/technology/supercomputer (cited on page 71).

[7] Alvaro Fernandez et al. 'Supercomputers to improve the performance in higher education: A review of the literature'. In: *Computers & Education* 128 (2019), pp. 353–364. DOI: https://doi.org/10.1016/j.compedu.2018.10.004 (cited on page 71).

[8] IBM. *What is a mainframe?* [Online; accessed 21-November-2023]. 2023. URL: https://www.ibm.com/topics/mainframe (cited on page 71).

[9] Wikipedia contributors. *Minicomputer — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Minicomputer&oldid=1184910564 (cited on page 71).

[10] Wikipedia contributors. *Microcomputer — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Microcomputer&oldid=1181658852 (cited on page 71).

[11] Intel. *Real-Time Systems Overview and Examples*. [Online; accessed 21-November-2023]. 2023. URL: https://www.intel.com/content/www/us/en/robotics/real-time-systems.html (cited on page 72).

[12] Emre Dalkiran et al. 'Automated integration of real-time and non-real-time defense systems'. In: *Defence Technology* 17.2 (2021), pp. 657–670. DOI: https://doi.org/10.1016/j.dt.2020.01.005 (cited on page 72).

[13] Wikipedia contributors. *Control system — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Control_system&oldid=1144002131 (cited on page 73).

[14] Wikipedia contributors. *Information system — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Information_system&oldid=1181776236 (cited on page 73).

[15] Vipin P. Veetil. *Coordination in Centralized and Decentralized Systems*. Mar. 2017. URL: https://en.wikipedia.org/w/index.php?title=Information_system&oldid=1181776236 (cited on page 73).

[16] Martinus Richardus van Steen and Andrew S. Tanenbaum. *Distributed Systems*. English. 3rd. Self-published, open publication. Feb. 2017 (cited on page 73).

[17] Wikipedia contributors. *Open system (computing) — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-November-2023]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Open_system_(computing) (cited on page 74).

[18] Wikipedia contributors. *Central processing unit — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Central_processing_unit. [Online; accessed 23-November-2023]. 2023 (cited on page 75).

[19] Wikipedia contributors. *Random-access memory — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Random-access_memory (cited on pages 75, 77).

[20] Wikipedia contributors. *Arithmetic logic unit — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Arithmetic_logic_unit (cited on page 75).

[21] Wikipedia contributors. *Processor register — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Processor_register (cited on page 75).

[22] Wikipedia contributors. *Cache (computing) — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Cache_(computing) (cited on page 76).

[23] John Rushby. *A Comparison of Bus Architectures for Safety-Critical Embedded Systems*. [Online; accessed October, 03, 2024]. 2001. URL: https://www.csl.sri.com/papers/buscompare/buscompare.pdf (cited on page 76).

[24] *TensorFlow*. Accessed: 2024-10-07. URL: https://www.tensorflow.org/ (cited on page 94).

[25] *PyTorch*. Accessed: 2024-10-07. URL: https://pytorch.org/ (cited on page 94).

[26] *Keras*. Accessed: 2024-10-07. URL: https://keras.io/ (cited on page 94).

[27] *scikit-learn*. Accessed: 2024-10-07. URL: https://scikit-learn.org/stable/ (cited on page 94).

[28] *SpaCy*. Accessed: 2024-10-07. URL: https://spacy.io/ (cited on page 95).

[29] *NLTK*. Accessed: 2024-10-07. URL: https://www.nltk.org/ (cited on page 95).

[30] *Hugging Face*. Accessed: 2024-10-07. URL: https://huggingface.co/ (cited on page 95).

[31] *ChatGPT*. Accessed: 2024-10-07. URL: https://openai.com/chatgpt/ (cited on page 95).

[32] *OpenCV*. Accessed: 2024-10-07. URL: https://opencv.org/ (cited on page 95).

[33] *TensorFlow Vision*. Accessed: 2024-10-07. URL: https://www.tensorflow.org/tutorials/images (cited on page 95).

[34] *Detectron2*. Accessed: 2024-10-07. URL: https://github.com/facebookresearch/detectron2 (cited on page 95).

[35] *YOLO*. Accessed: 2024-10-07. URL: https://yolov8.com/ (cited on page 95).

[36] *UiPath*. Accessed: 2024-10-07. URL: https://www.uipath.com/ (cited on page 95).

[37] *Automation Anywhere*. Accessed: 2024-10-07. URL: https://www.automationanywhere.com/ (cited on page 95).

[38] *blueprism*. Accessed: 2024-10-07. URL: https://www.blueprism.com/ (cited on page 95).

[39] Manuel Jiménez, Rogelio Palomera, and Isidoro Couvertier. *Introduction to embedded systems*. Springer, 2013 (cited on page 104).

[40] Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition*. MIT Press, 2013 (cited on page 104).

[41] Lawrence Williams. *Embedded Systems Tutorial: What is, History & Characteristics*. [Online; accessed 12-November-2023]. 2023. URL: https://www.guru99.com/embedded-systems-tutorial.html (cited on page 104).

[42] KCS Murti. *Design Principles for Embedded Systems*. Springer Nature Singapore, 2021 (cited on page 105).

[43] John Keller. *The future of high-performance embedded computing*. [Online; accessed 15-November-2023]. Oct. 2023. URL: https://www.militaryaerospace.com/computers/article/14299053/the-future-of-highperformance-embedded-computing (cited on page 107).

[44] Jacob Beningo. *Will AI take embedded software jobs?* [Online; accessed 15-November-2023]. Oct. 2023. URL: https://www.embedded.com/will-ai-take-embedded-software-jobs/ (cited on page 107).

[45] A. M. Turing. 'Computing Machinery and Intelligence'. In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. DOI: 10.1093/mind/LIX.236.433 (cited on page 109).

[46] Bill Gates. *AI is about to completely change how you use computers*. [Online; accessed 13-November-2023]. Nov. 2023. URL: https://www.gatesnotes.com/AI-agents (cited on page 110).

[47] Edward W. Felten, Manav Raj, and Robert Seamans. *How will Language Modelers like ChatGPT Affect Occupations and Industries?* Mar. 2023. DOI: http://dx.doi.org/10.2139/ssrn.4375268. URL: https://ssrn.com/abstract=4375268 (cited on page 110).

[48] Tyna Eloundou et al. *GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models*. 2023. URL: https://arxiv.org/abs/2303.10130 (cited on page 110).

[49] Mirko Stojiljkovic. *Linear Regression in Python*. [Online; accessed 13-November-2023]. 2023. URL: https://realpython.com/linear-regression-in-python/ (cited on page 110).

[50] Amrita Sharma and Neha Chaudhary. 'Linear Regression Model for Agile Software Development Effort Estimation'. In: *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*. 2020, pp. 1–4. DOI: 10.1109/ICRAIE51050.2020.9358309 (cited on page 111).

[51] Blazej Ulanowicz et al. 'Combining Random Forest and Linear Regression to Improve Network Traffic Prediction'. In: *2023 23rd International Conference on Transparent Optical Networks (ICTON)*. 2023, pp. 1–4. DOI: 10.1109/ICTON59386.2023.10207506 (cited on page 111).

[52] P.J. Joseph, Kapil Vaswani, and M.J. Thazhuthaveetil. 'Construction and use of linear regression models for processor performance analysis'. In: *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.* 2006, pp. 99–108. DOI: 10.1109/HPCA.2006.1598116 (cited on page 112).

[53] A. Latocha. 'Robust fault detection, location, and recovery of damaged data using linear regression and mathematical models'. In: *IFAC-PapersOnLine* 51.24 (2018). 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018, pp. 300–306. DOI: https://doi.org/10.1016/j.ifacol.2018.09.593 (cited on page 112).

[54] Mayank Banoula. *An Introduction to Logistic Regression in Python*. [Online; accessed 13-November-2023]. 2023. URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/logistic-regression-in-python (cited on page 112).

[55] Bilge Kagan Dedeturk and Bahriye Akay. 'Spam filtering using a logistic regression model trained by an artificial bee colony algorithm'. In: *Applied Soft Computing* 91 (2020), p. 106229. DOI: https://doi.org/10.1016/j.asoc.2020.106229 (cited on page 113).

[56] Zihao Song et al. 'Doubly robust logistic regression for image classification'. In: *Applied Mathematical Modelling* 123 (2023), pp. 430–446. DOI: https://doi.org/10.1016/j.apm.2023.06.039 (cited on page 113).

[57] Xinyi Yang. 'Prediction of Credit Risk Based on Logistic Regression and Random Forest Technique'. In: ICCSIE '22. Brisbane, QLD, Australia: Association for Computing Machinery, 2022, pp. 531–535. DOI: 10.1145/3558819.3565138 (cited on page 113).

[58] Hadi Kazemi-Arpanahi, Raoof Nopour, and Mostafa Shanbehzadeh. 'Using logistic regression to develop a diagnostic model for COVID-19: A single-center study'. en. In: *J. Educ. Health Promot.* 11.1 (2022), p. 153 (cited on page 113).

[59] Shankarshan Prasad Tiwari and Ebha Koley. 'A Decision Tree-Based Algorithm for Fault Detection and Section Identification of DC Microgrid'. In: *DC Microgrids*. John Wiley & Sons, Ltd, 2022. Chap. 13, pp. 397–420. doi: https://doi.org/10.1002/9781119777618.ch13 (cited on page 114).

[60] Leyli Mohammad Khanli, Farnaz Mahan, and Ayaz Isazadeh. 'Active rule learning using decision tree for resource management in Grid computing'. In: *Future Generation Computer Systems* 27.6 (2011), pp. 703–710. doi: https://doi.org/10.1016/j.future.2010.12.016 (cited on page 115).

[61] Faizan Ullah et al. 'Modified Decision Tree Technique for Ransomware Detection at Runtime through API Calls'. In: *Scientific Programming* 2020 (Aug. 2020), p. 8845833. doi: 10.1155/2020/8845833 (cited on page 115).

[62] Vasiliki Matzavela and Efthimios Alepis. 'Decision tree learning through a Predictive Model for Student Academic Performance in Intelligent M-Learning environments'. In: *Computers and Education: Artificial Intelligence* 2 (2021), p. 100035. doi: https://doi.org/10.1016/j.caeai.2021.100035 (cited on page 115).

[63] Anna Bosch, Andrew Zisserman, and Xavier Munoz. 'Image Classification using Random Forests and Ferns'. In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–8. doi: 10.1109/ICCV.2007.4409066 (cited on page 116).

[64] G. Prashanth et al. 'Using Random Forests for Network-based Anomaly detection at Active routers'. In: Feb. 2008, pp. 93–96. doi: 10.1109/ICSCN.2008.4447167 (cited on page 116).

[65] Martin Kopp, Tomas Pevny, and Martin Holena. 'Anomaly explanation with random forests'. In: *Expert Systems with Applications* 149 (2020), p. 113187. doi: https://doi.org/10.1016/j.eswa.2020.113187 (cited on page 116).

[66] Rung-Ching Chen et al. 'Selecting critical features for data classification based on machine learning methods'. In: *Journal of Big Data* 7.1 (July 2020), p. 52. doi: 10.1186/s40537-020-00327-4 (cited on page 116).

[67] Heng-Ru Zhang and Fan Min. 'Three-way recommender systems based on random forests'. In: *Knowledge-Based Systems* 91 (2016). Three-way Decisions and Granular Computing, pp. 275–286. doi: https://doi.org/10.1016/j.knosys.2015.06.019 (cited on page 116).

[68] Rodney Kizito et al. 'The Application of Random Forest to Predictive Maintenance'. In: May 2018 (cited on page 116).

[69] Mayank Arya Chandra and S. S. Bedi. 'Survey on SVM and their application in imageclassification'. In: *International Journal of Information Technology* 13.5 (Oct. 2021), pp. 1–11. doi: 10.1007/s41870-017-0080-1 (cited on page 117).

[70] Zahra Torabi, Mohammad H. Nadimi-Shahraki, and Akbar Nabiollahi. 'Efficient Support Vector Machines for Spam Detection: A Survey'. In: *(IJCSIS) International Journal of Computer Science and Information Security, Vol. 13, No. 1, January 2015* 13 (Jan. 2015) (cited on page 117).

[71] Yinhui Li et al. 'An efficient intrusion detection system based on support vector machines and gradually feature removal method'. In: *Expert Syst. Appl.* 39 (2012), pp. 424–430 (cited on page 118).

[72] Savita Ahlawat and Amit Choudhary. 'Hybrid CNN-SVM Classifier for Handwritten Digit Recognition'. In: *Procedia Computer Science* 167 (2020). International Conference on Computational Intelligence and Data Science, pp. 2554–2560. doi: https://doi.org/10.1016/j.procs.2020.03.309 (cited on page 118).

[73] Yongyi Ran et al. *A Survey of Predictive Maintenance: Systems, Purposes and Approaches*. 2019 (cited on page 118).

[74] Akash Shastri. *5 Neural network architectures you must know for Computer Vision*. [Online; accessed 19-November-2023]. Nov. 2020. URL: https://towardsdatascience.com/5-neural-network-architectures-you-must-know-for-computer-vision-31d2991fe24e (cited on page 119).

[75] Olga Davydova. *7 types of Artificial Neural Networks for Natural Language Processing*. [Online; accessed 19-November-2023]. Sept. 2017. URL: https://medium.com/datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2 (cited on page 119).

[76] Ilias Papastratis. *Speech Recognition: a review of the different deep learning approaches*. [Online; accessed 19-November-2023]. July 2021. URL: https://theaisummer.com/speech-recognition/ (cited on page 120).

[77] Harry A. Pierson and Michael S. Gashler. 'Deep learning in robotics: a review of recent research'. In: *Advanced Robotics* 31.16 (2017), pp. 821–835. DOI: 10.1080/01691864.2017.1365009 (cited on page 120).

[78] Mohammed Hassan and Mohamed Hamada. 'A Neural Networks Approach for Improving the Accuracy of Multi-Criteria Recommender Systems'. In: *Applied Sciences* 7.9 (2017). DOI: 10.3390/app7090868 (cited on page 120).

[79] Xanthi Bampoula et al. 'A Deep Learning Model for Predictive Maintenance in Cyber-Physical Production Systems Using LSTM Autoencoders'. In: *Sensors (Basel)* 21.3 (Feb. 2021) (cited on page 120).

[80] Bingming Wang, Shi Ying, and Zhe Yang. 'A Log-Based Anomaly Detection Method with Efficient Neighbor Searching and Automatic K Neighbor Selection'. In: *Scientific Programming* 2020 (June 2020), p. 4365356. DOI: 10.1155/2020/4365356 (cited on page 121).

[81] Lamiaa M. Elshenawy, Chouaib Chakour, and Tarek A. Mahmoud. 'Fault detection and diagnosis strategy based on k-nearest neighbors and fuzzy C-means clustering algorithm for industrial processes'. In: *Journal of the Franklin Institute* 359.13 (2022), pp. 7115–7139. DOI: https://doi.org/10.1016/j.jfranklin.2022.06.022 (cited on page 121).

[82] G. Mazzuto et al. 'Health Indicator for Predictive Maintenance Based on Fuzzy Cognitive Maps, Grey Wolf, and K-Nearest Neighbors Algorithms'. In: *Mathematical Problems in Engineering* 2021 (Feb. 2021), p. 8832011. DOI: 10.1155/2021/8832011 (cited on page 121).

[83] Georgios Chatzigeorgakidis et al. 'FML-kNN: scalable machine learning on Big Data using k-nearest neighbor joins'. In: *Journal of Big Data* 5.1 (Feb. 2018), p. 4. DOI: 10.1186/s40537-018-0115-x (cited on page 121).

[84] Pengji Zhang, Wei Ma, and Sean Qian. 'Cluster analysis of day-to-day traffic data in networks'. In: *Transportation Research Part C: Emerging Technologies* 144 (2022), p. 103882. DOI: https://doi.org/10.1016/j.trc.2022.103882 (cited on page 122).

[85] Ioannis P. Panapakidis and Georgios C. Christoforidis. 'Optimal Selection of Clustering Algorithm via Multi-Criteria Decision Analysis (MCDA) for Load Profiling Applications'. In: *Applied Sciences* 8.2 (2018). DOI: 10.3390/app8020237 (cited on page 122).

[86] Himanshu Mittal et al. 'A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets'. In: *Multimedia Tools and Applications* 81.24 (Oct. 2022), pp. 35001–35026. DOI: 10.1007/s11042-021-10594-9 (cited on page 123).

[87] Yong-Hao Wu et al. 'FATOC: Bug Isolation Based Multi-Fault Localization by Using OPTICS Clustering'. In: *Journal of Computer Science and Technology* 35.5 (Oct. 2020), pp. 979–998. DOI: 10.1007/s11390-020-0549-4 (cited on page 123).

[88] Wikipedia contributors. *XGBoost — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=XGBoost (cited on page 123).

[89] Wikipedia contributors. *LightGBM — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-November-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=LightGBM (cited on page 123).

[90] Maya Hilda Lestari Louk and Bayu Adhi Tama. 'Revisiting Gradient Boosting-Based Approaches for Learning Imbalanced Data: A Case of Anomaly Detection on Power Grids'. In: *Big Data and Cognitive Computing* 6.2 (2022). DOI: 10.3390/bdcc6020041 (cited on page 124).

[91] Alexandre S. Roque et al. 'An analysis of machine learning algorithms in rotating machines maintenance'. In: *IFAC-PapersOnLine* 55.2 (2022). 14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022, pp. 252–257. DOI: https://doi.org/10.1016/j.ifacol.2022.04.202 (cited on page 124).

[92] Maryam Douiba et al. 'Anomaly detection model based on gradient boosting and decision tree for IoT environments security'. In: *Journal of Reliable Intelligent Environments* 9 (July 2022). DOI: 10.1007/s40860-022-00184-3 (cited on page 124).

[93] Tusar Kanti Dash et al. 'Gradient boosting machine and efficient combination of features for speech-based detection of COVID-19'. In: *IEEE J. Biomed. Health Inform.* 26.11 (Nov. 2022), pp. 5364–5371 (cited on page 124).

[94] Steve Brown. *AI for Circuit Design Quality, Productivity, and Advanced-Node Mapping*. [Online; accessed 17-November-2023]. Oct. 2023. URL: https://community.cadence.com/cadence_blogs_8/b/artificial-intelligence/posts/ai-for-circuit-design-quality-productivity-and-advanced-node-mapping (cited on page 125).

[95] Jozef Jenis et al. 'Engineering Applications of Artificial Intelligence in Mechanical Design and Optimization'. In: *Machines* 11.6 (2023). DOI: 10.3390/machines11060577 (cited on page 125).

[96] M. Luz Castro Pena et al. 'Artificial intelligence applied to conceptual design. A review of its use in architecture'. In: *Automation in Construction* 124 (2021), p. 103550. DOI: https://doi.org/10.1016/j.autcon.2021.103550 (cited on page 125).

[97] Richard Meyes, Nils Hutten, and Tobias Meisen. 'Transparent and Interpretable Failure Prediction of Sensor Time Series Data with Convolutional Neural Networks'. In: *Procedia CIRP* 104 (2021). 54th CIRP CMS 2021 - Towards Digitalized Manufacturing 4.0, pp. 1446–1451. DOI: https://doi.org/10.1016/j.procir.2021.11.244 (cited on page 125).

[98] Hiyam Adil Habeeb et al. 'Design Optimization Method Based on Artificial Intelligence (Hybrid Method) for Repair and Restoration Using Additive Manufacturing Technology'. In: *Metals* 13.3 (2023). DOI: 10.3390/met13030490 (cited on page 125).

[99] Elena Trajkova et al. 'Active Learning for Automated Visual Inspection of Manufactured Products'. In: *CoRR* abs/2109.02469 (2021) (cited on page 126).

[100] Erik Westphal and Hermann Seitz. 'A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks'. In: *Additive Manufacturing* 41 (2021), p. 101965. DOI: https://doi.org/10.1016/j.addma.2021.101965 (cited on page 126).

[101] H Kamel. 'Artificial intelligence for predictive maintenance'. In: *Journal of Physics: Conference Series* 2299.1 (2022), p. 012001. DOI: 10.1088/1742-6596/2299/1/012001 (cited on page 126).

[102] Laion Lima Boaventura, Paulo Henrique Ferreira, and Rosemeire Leovigildo Fiaccone. 'On flexible Statistical Process Control with Artificial Intelligence: Classification control charts'. In: *Expert Systems with Applications* 194 (2022), p. 116492. DOI: https://doi.org/10.1016/j.eswa.2021.116492 (cited on page 126).

[103] Dataquest. *Using Machine Learning and Natural Language Processing Tools for Text Analysis*. [Online; accessed 19-November-2023]. Feb. 2022. URL: https://www.dataquest.io/blog/using-machine-learning-and-natural-language-processing-tools-for-text-analysis/ (cited on page 126).

[104] Federico Walas Mateo and Andres Redchuk. 'Artificial Intelligence and Machine Learning as a Process Optimization driver under Industry 4.0 framework, the role of the people in the process'. In: June 2021 (cited on page 126).

[105] Mehrdokht Pournader et al. 'Artificial intelligence applications in supply chain management'. In: *International Journal of Production Economics* 241 (2021), p. 108250. DOI: https://doi.org/10.1016/j.ijpe.2021.108250 (cited on page 126).

[106] Sana Fatima et al. 'Automated Testing with Machine Learning Frameworks: A Critical Analysis'. In: *Engineering Proceedings* 20.1 (2022). DOI: 10.3390/engproc2022020012 (cited on page 126).

[107] Nazim Hussain and Greian Pangilinan. 'Robotics and Automation with Artificial Intelligence: Improving Efficiency and Quality'. In: *Aptisi Transactions on Technopreneurship (ATT)* 5 (May 2023), pp. 176–189. DOI: 10.34306/att.v5i2.252 (cited on page 126).

[108] Abu Rayhan. 'Artificial Intelligence In Robotics: From Automation To Autonomous Systems'. PhD thesis. July 2023. DOI: 10.13140/RG.2.2.15540.42889 (cited on page 126).

[109] Radu Ciora, Carmen Simion, and Marius Cioca. 'Quality Improvement Based on Big Data Analysis'. In: Apr. 2016, pp. 101–109. DOI: 10.1007/978-3-319-32942-0_7 (cited on page 126).

[110] John B. Ingraham et al. 'Illuminating protein space with a programmable generative model'. In: *Nature* (Nov. 2023). DOI: 10.1038/s41586-023-06728-8 (cited on page 128).

[111] Aditi Bhutoria. 'Personalized education and Artificial Intelligence in the United States, China, and India: A systematic review using a Human-In-The-Loop model'. In: *Computers and Education: Artificial Intelligence* 3 (2022), p. 100068. DOI: https://doi.org/10.1016/j.caeai.2022.100068 (cited on page 128).

[112] Gavoy Small. *The Top 10 Adaptive Learning Platforms of 2023*. [Online; accessed 13-November-2023]. Aug. 2023. URL: https://www.edapp.com/blog/adaptive-learning-platforms/ (cited on page 128).

[113] Stephen Kosslyn. *Active Learning with AI: A Practical Guide*. Alinea Learning, Nov. 2023 (cited on page 129).

[114] Bhavya Ghai et al. 'Explainable Active Learning (XAL): Toward AI Explanations as Interfaces for Machine Teachers'. In: *Proc. ACM Hum.-Comput. Interact.* 4.CSCW3 (2021). DOI: 10.1145/3432934 (cited on page 129).

[115] Deepesh Agarwal et al. *Addressing practical challenges in Active Learning via a hybrid query strategy*. 2021 (cited on page 129).

[116] Miguel Cardona, Roberto Rodriguez, and Kristina Ishmael. *Artificial Intelligence and the Future of Teaching and Learning*. [Online; accessed 13-November-2023]. May 2023. URL: https://www2.ed.gov/documents/ai-report/ai-report.pdf (cited on page 129).

[117] Aaron Gifford. *Educators Roundtable: Demystify AI With Transparency and Practice*. [Online; accessed 12-November-2023]. Sept. 2023. URL: https://www.govtech.com/education/higher-ed/educators-roundtable-demystify-ai-with-transparency-and-practice (cited on page 129).

[118] Josette Keelor. *The power of AI in the classroom*. [Online; accessed 12-November-2023]. Nov. 2023. URL: https://www.jmu.edu/news/2023/10/24-ai-in-the-classroom.shtml (cited on page 129).

[119] Donna Dodson et al. *Securing Small-Business and Home Internet of Things (IoT) Devices: Mitigating Network-Based Attacks Using Manufacturer Usage Description (MUD)*. Accessed: 11-20-2023. May 2021. DOI: https://doi.org/10.6028/NIST.SP.1800-15 (cited on page 131).

[120] Cisco. *What is malware?* [Online; accessed 21-November-2023]. 2023. URL: https://www.cisco.com/site/us/en/learn/topics/security/what-is-malware.html (cited on page 131).

[121] Phishing.org. *What Is Phishing?* [Online; accessed 21-November-2023]. 2023. URL: https://www.phishing.org/what-is-phishing (cited on page 132).

[122] NCSC. *Denial of Service (DoS) guidance*. [Online; accessed 21-November-2023]. 2023. URL: https://www.ncsc.gov.uk/collection/denial-service-dos-guidance-collection (cited on page 132).

[123] Cisco. *Understanding SQL Injection*. [Online; accessed 21-November-2023]. 2023. URL: https://sec.cloudapps.cisco.com/security/center/resources/sql_injection.html (cited on page 132).

[124] IBM. *What is Social Engineering?* [Online; accessed 21-November-2023]. 2023. URL: https://www.ibm.com/topics/social-engineering (cited on page 132).

[125] CISA. *Advanced Persistent Threats and Nation-State Actors*. [Online; accessed 21-November-2023]. 2023. URL: https://www.cisa.gov/topics/cyber-threats-and-advisories/advanced-persistent-threats-and-nation-state-actors (cited on page 132).

Alphabetical Index