Department of Electrical & Computer Engineering

# COEN 212:
# DIGITAL SYSTEMS DESIGN
## Lecture 1: Numbers

**Instructor:** Dr. Reza Soleymani, Office: EV-5.125,
Telephone: 848-2424 ext.: 4103.

# Lecture 1:
# Contents of this Lecture

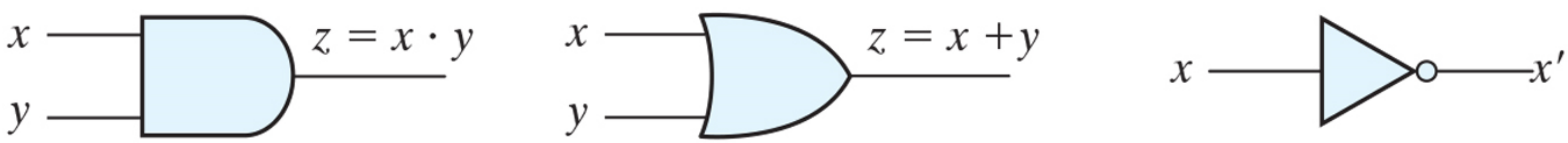
Department of Electrical & Computer Engineering

- Introduction to the course.
- Course outline.
- Number Systems.
- Binary Numbers and Binary Logic.

# Lecture 1: Objective of the Course



- In this course, we acquire an appreciation of the fact that all modern digital electronic devices, even those as complex as computers and smartphones are designed by a few basic blocks called logic gates.

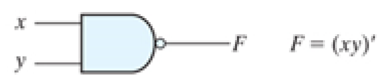



(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

# Lecture 1:
# Objective of the Course

Université Concordia University
Department of Electrical & Computer Engineering

- We will even show that technically speaking only one type of gate, namely a NAND gate is enough to build all the Digital Empire.
- Two-input NAND gate:




- We will see that, we can also exclusively use another type of gate called a NOR gate.
- Two-input NOR gate:

# Lecture 1: Course Content

Université Concordia University
Department of Electrical & Computer Engineering

- Mathematics of Digital Design.
- Combinational Circuits.
- Sequential Circuits.

**Note: In this course, we are not concerned with the implementation of gates. You will learn how the gates are made out of transistors in a digital electronics course.**

# Lecture 1:
# Course Content

Université Concordia University
Department of Electrical & Computer Engineering

1. Introduction, Number System, Binary Numbers: Chapter 1.
2. Boolean algebra and Functions: Chapter 2.
3. Canonical and Standard Forms: Chapter 2.
4. K-Map representation: Chapter 3.
5. K-Map minimization: Chapter 3.
6. 2-level, multilevel representation and minimization: Chapter 3
7. Introduction to HDL: Chapter 3.
8. Timing Analysis of combinational circuits: Chapter 3.
9. Analysis and design procedures: Chapter 4.
10. Popular arithmetic and logical combinational circuits: Chapter 4.
11. Decoders/Encoders and Multiplexers: Chapter 4.
12. Introduction to sequential circuits, Latches and Flip Flops: Chapter 5.
13. Analysis of sequential circuits: Chapter 5.
14. The state diagram: Chapter 5.
15. Synchronous circuit design: Chapter 5.
16. Registers and counters: Chapter 6
17. Memory and Programmable Logic Devices: Chapter 7.

# Lecture 1:
# Course Material

- **Textbook:**
  - **Digital** Design: With an Introduction to the Verilog HDL, VHDL, and SystemVerilog, by M. Morris R. Mano and Michael D. Ciletti, 6th Edition, Pearson, 2018.
- **Lab Manual.**

# Lecture 1: Grading Scheme

Concordia University
Department of Electrical & Computer Engineering

- Assignment: 5%
- Lab: 20%
- Midterm : 25%
- Final : 50%

Note 1: Failing to write the Midterm results in losing the 25% assigned to the test.
Note 2: In order to pass the course, you should get at least 50% in the final.

# Lecture 1:
# Reading for this Lecture



- Chapter 1 of Digital Design by M. Morris R. Mano and Michael D. Ciletti, 6th Edition, Pearson, 2018.

# Lecture 1: Numbers

Concordia University
Department of Electrical & Computer Engineering

- Integers and real numbers can be represented using a finite number of symbols called digits.
- For example, in decimal system the number 35987 is represented as:
  - 30000+5000+900+80+7 $= 3\times 10^4 + 5 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 7 \times 10^0$,
- and 275.69 is represented as:
  - $200+70+5+\frac{6}{10} + \frac{9}{100} = 2 \times 10^2 + 7 \times 10^1 + 5 \times 10^0 + 6 \times 10^{-1} + 9 \times 10^{-2}$
- Probably, we use base ten because we have 10 fingers and the first instances of counting was done with fingers. In fact the term digit means finger.
- Have we had four fingers (like the Simpsons), most likely, we have used base 8. Such system is called octal.
- The number 253 (in decimal system) will be represented as 375 in octal system:

$$3 \times 8^2 + 7 \times 8^1 + 5 \times 8^0 = 253.$$

- To translate from decimal (base 10) to octal (base 8) we do the following:

| Operation | Result | Remainder |
|---|---|---|
| Divide 253 by 8 | 31 | 5 |
| Divide 31 by 8 | 3 | 7 |
| Divide 3 by 8 | 0 | 3 |

Now read the remainders (the third column) from bottom to the top to get 375.

# Lecture 1:
# Numbers: Binary System

Université Concordia University
Department of Electrical & Computer Engineering

- Binary Numbers: A representation usually used in digital systems is binary.
- Binary system uses an alphabet consisting only of two symbols. You may call these two digits zero (0) and one (1).
- A number can be represented in binary (base 2) as:

  $a_n * 2^n + a_{n-1} * 2^{n-1} + \cdots + a_1 \times 2^1 + a_0 * 2^0 + a_{-1}2^{-1} + a_{-2} * 2^{-2} + \ldots$
- For example, the number 75 in our familiar decimal system will be transformed into binary using the following steps:

| Operation | Result | Remainder |
|---|---|---|
| Divide 75 by 2 | 37 | 1 |
| Divide 37 by 2 | 18 | 1 |
| Divide 18 by 2 | 9 | 0 |
| Divide 9 by 2 | 4 | 1 |
| Divide 4 by 2 | 2 | 0 |
| Divide 2 by 2 | 2 | 0 |
| Divide 1 by 2 | 0 | 1 |

- Now read the remainders (the third column) from bottom to the top to get $(1001011)_2$.

# Lecture 1:
# Numbers: *Binary to Octal conversion*

Department of Electrical & Computer Engineering

- We can divide binary digits of a binary number into groups of 3 and represent them by their octal value.
- Example, the number 1001011 can be divided into 001, 001 and 011. This will be 113 in base 8.
- Note that we inserted two zeros to the before the number to make the number of bits equal to 9.
- Let's check to see whether what we have done is correct by transforming 75 to octal directly:

| Operation | Result | Remainder |
|---|---|---|
| Divide 75 by 8 | 9 | 3 |
| Divide 9 by 8 | 1 | 1 |
| Divide 1 by 8 | 0 | 1 |

- Now read the remainders (the third column) from bottom to the top to get 113.

Université Concordia University
Department of Electrical & Computer Engineering

# Lecture 1: Numbers: *Hexadecimal*

- This is a representation in base (radix) 16. Here in addition to zero and 1 to 9 we use A, B, C, D, E, F as 10, 11, 12, 13, 14, 15, respectively.
- To change a binary number into hexa-decimal, we divide its bits into groups of 4.
- If the number of bits before the "decimal" (I avoid calling it binary point in order not to cause confusion) point is not a multiple of 4, we add extra 0's before the number.
- Also if the number of bits after the point is not a multiple of four we add zeros after the number.
- Example: 10110001101011.1111001 will be written as:

| 0010 | 1100 | 0110 | 1011 | . | 1111 | 0010 |
|---|---|---|---|---|---|---|
| 2 | C | 6 | B | | F | 2 |

- So, we get $(2C6B.F2)_{16}$.

# Lecture 1:
# Numbers: *Complements*

Université Concordia University
Department of Electrical & Computer Engineering

- **Radix Complement:**
- If we subtract a number X in radix r from $r^n$ we get $r^n - X$, the radix complement of X.
- For example, 10's complement of 3250 is $10^4 - 3250 = 6750$.
- Note that ten's complement can be formed by leaving the zeros at the end of the number intact, subtract the first non-zero digit after zeros from 10 and subtract the rest of the digits from 9.
- Putting any number of zeros before a number does not change its value. For example 003250 is no different from 3250. But 10's complement of 003250 is 996750. This shows that putting any number of 9's before a 10's complement does not change the value of the number.

- Two's complement of a number X (in binary system) is $2^n$-X.

- For example for X=1101100 we get 0010100 and for X=0110111 we get 1001001.

# Lecture 1:
# Numbers: *Complements*

Department of Electrical & Computer Engineering

- **Diminished Radix Complement:**
- If we subtract a number X in from $r^n - 1$ we get $r^n - 1 - X$, the r-1 complement of X.
- Note that $r^n - 1$ is an n digit number with all its digits equal to r-1. For example, the 9's complement of 546700 is,
- 999999 – 546700 $= 453299$.
- and 9's complement of 012398 is 987601. We see that the 9's complement of a decimal number is found by simply replacing each digit by the results of subtraction of that digita from 9.
- In binary case, 1's complement of a binary number is found by complementing each bit, i.e., subtracting it from 1. For example the 1's complement of 1011000 is 0100111 and the 1's complement of 0101101 is 1010010.

# Lecture 1: Numbers: *Subtraction*

Université Concordia University
Department of Electrical & Computer Engineering

- Subtraction in radix complement:
- To form M-N:
- Add M to r's complement of N to get: $M + r^n - N = r^n + M - N$.
- If M>N the result will be greater than $r^n$ and there will be a carry that can be discarded.
- If M<N, there would be no carry and the result will be $r^n$-(N-M), i.e., the r's complement of N-M. So, we take the r's complement of the result and add a minus sign.

- Example: Find X-Y and Y-X if X=1010100 and Y=1000011.

- To find X-Y: 1010100 + 0111101 = 10010001.
- Discarding the carry, we get 0010001.

- To find Y-X: 1000011 + 0101010 = 1101111.
- There is no carry. So, we find 2's complement to get 0010001 and put a minus sign in front of it to have -0010001.

# Lecture 1:
# Numbers: *Subtraction*

Department of Electrical & Computer Engineering

- **Subtraction in diminished radix complement**:
- If we use 1's complement, since the numbers are one less than 2's complement, we have to subtract one when there is a carry. When there is no carry, the final 1's complement takes care of the one.
- Example: Let X=1100101 and Y=1011010. Find X-Y and Y-X.
- For X-Y: 1100101 + 0100101 = 10001010. There is a carry so, we take 0001010 and add a one to get 0001011.
- For Y-X: 1011010 + 0011010 = 1110100. Just take 1's complement 0001011 and add a minus sign to get -0001011.
- **Signed Binary Numbers:**
- In digital (binary) system having an extra symbol to specify the sign of a number is not convenient. That is why usually the following convention is used:
- The most significant bit of a number is 1 if it is negative and 0 if it is positive.
- Of course, we need to specify the word length in advance in order to avoid confusion. For example 11100 can be either -12 or +28 depending on whether we deal with 5-bit signed number (1 bit sign and 4 bits magnitude) or a 5-bit unsigned number.

# Lecture 1:
# Numbers: *Signed Binary Numbers*

Concordia University
Department of Electrical & Computer Engineering

- **Signed Binary Numbers:**
- In digital (binary) system having an extra symbol to specify the sign of a number is not convenient. That is why usually the following convention is used:
- The most significant bit of a number is 1 if it is negative and 0 if it is positive.
- Of course, we need to specify the word length in advance in order to avoid confusion. For example 11100 can be either -12 or +28 depending on whether we deal with 5-bit signed number (1 bit sign and 4 bits magnitude) or a 5-bit unsigned number.

# Lecture 1:

## Basic building blocks of logical circuits

Department of Electrical & Computer Engineering

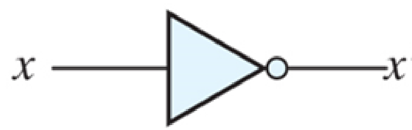
- A logical device, e.g., a computer, a digital phone, an IPOD, etc. consists of the following blocks:

1. Memory: to store information.
2. A processing unit: to operate on information.
3. A control unit: to specify the sequence of operations.
4. Input and output units: to input data to the device or output the result of processing: keyboard, scanners, monitors, and printers.

# Lecture 1:
## *Logic gates:NOT*

Université Concordia University
Department of Electrical & Computer Engineering

- We said that, we can represent any number with only two symbols, i.e., transform it to a binary format. In this format any processing task can be broken down into operation on binary digits (bits). There are three basic operations for bit manipulation. These are AND, OR, and NOT. NOT operates on a single bit, i.e., it has one input. The output of a NOT is 1 if its input is 0 and vice versa.




| $x$ | $x'$ |
|---|---|
| **0** | 1 |
| **1** | 0 |

# Lecture 1:
## *Logic gates: AND*

Université Concordia University
Department of Electrical & Computer Engineering

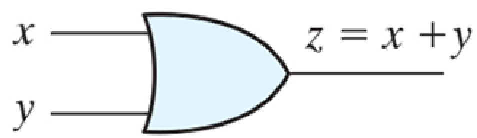
- NOT is a unitary operation (works on one input)
- AND and OR are binary operations.
- The output of an AND gate is 1 if both its inputs are 1 and 0 otherwise.


- Truth table for AND gate:

| $x$ | $y$ | $x.y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Lecture 1:
## *Logic gates: OR*

- The output of an OR gate is zero if both its inputs are zero. Otherwise its output is 1.




- Truth table for OR gate:

| $x$ | $y$ | $x + y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# Lecture 1:

## *De Morgan's Theorem*

Department of Electrical & Computer Engineering

- Note the similarity (duality) existing between the AND gate and the OR gate. If you exchange 0 and 1 in the truth table for one you get the other's truth table. That is if we negate the inputs to an OR gate (before feeding them to it) and also invert the output we get the operation of an AND gate. That is:
- $(X' + Y')' = XY$ or $(XY)' = X' + Y'$
- Similarly,
- $(X'Y')' = X + Y$ or $(X + Y)' = X'Y'$
- This is called De Morgan's theorem. This duality property makes the mathematics of binary systems (Boolean Algebra) different from the ordinary algebra.

# Lecture 1:

## *Knowledge Check*

Department of Electrical & Computer Engineering

1) $(XY + Z')'$ is:

a) $X'Z + Y'Z$, b) $(X'+Y')Z$ c) both a and b d) None

2) Truth table for $Z = X'Y + Y'X$ is:

a)

| x | y | z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

b)

| x | y | z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

c)

| x | y | z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

d)

| x | y | z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

3) For $x = y = 0$ and $z = 1$ find the output of: