ADC MODULE 3

Number systems

A number system is a format to represent numbers in a certain way.

- Base 10 (Decimal) — Represent any number using 10 digits [0–9]
  - The decimal number system is the most commonly used number system around the world which is easily understandable to people.
- Base 2 (Binary) — Represent any number using 2 digits [0–1]
  - The binary number system is used in computers and electronic systems to represent data and it consists of only two digits which are 0 and 1
- Base 8 (Octal) — Represent any number using 8 digits [0–7]
- Base 16(Hexadecimal) — Represent any number using 10 digits and 6 characters [0–9, A, B, C, D, E, F]

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
12	1101	D
14	1110	E
15	1111	F

CONVERSION

- Convert $(1110.011)_2$ into Decimal

$(1110.011)_2 = (?)_{10}$

Binary Number after Decimal Point

$= [\ (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)\ ] + [\ (0 \times 1/2) + (1 \times 1/4) + (1 \times 1/8)\ ]$

Binary Number before Decimal Point

$= 8 + 4 + 2 + 0 + 0 + 0.25 + 0.125$

$= 14.375$

Decimal to Binary

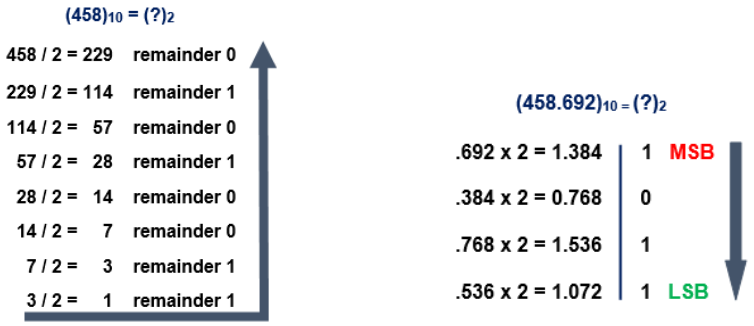
- Convert 160_{10} to binary Number

Divide by 2	Result	Remainder
160 ÷ 2	80	0
80 ÷ 2	40	0
40 ÷ 2	20	0
20 ÷ 2	10	0
10 ÷ 2	5	0
5 ÷ 2	2	1
2 ÷ 2	1	0
1 ÷ 2	0	1

Convert 294_{10} to binary Number

Divide by 2	Result	Remainder
294 ÷ 2	147	0
147 ÷ 2	73	1
73 ÷ 2	36	1
36 ÷ 2	18	0
18 ÷ 2	9	0
9 ÷ 2	4	1
4 ÷ 2	2	0
2 ÷ 2	1	0
1 ÷ 2	0	1

- Convert 458.692_{10} to binary Number



$(458.692)_{10} = (111001010.1011)_2$

Binary to Decimal

Binary to decimal conversion is done to convert a number given in the binary number system to its equivalent value in the decimal number system.

- Convert 1110_2, from binary to decimal

  $= (0 \times 2^0) + (1 \times 2^1) + (1 \times 2^2) + (1 \times 2^3)$
  $= (0 \times 2^0) + (1 \times 2^1) + (1 \times 2^2) + (1 \times 2^3)$
  $= 0 + 2 + 4 + 8$
  $= 14$

  Therefore, $1110_2 = 14_{10}$
- Find the decimal value of the binary number 11001011_2.

  $11001011_2 = (1 \times 2^0) + (1 \times 2^1) + (0 \times 2^2) + (1 \times 2^3) + (0 \times 2^4) + (0 \times 2^5) + (1 \times 2^6) + (1 \times 2^7)$
  $= (1 \times 1) + (1 \times 2) + (0 \times 4) + (1 \times 8) + (0 \times 16) + (0 \times 32) + (1 \times 64) + (1 \times 128)$
  $= 1 + 2 + 0 + 8 + 0 + 0 + 64 + 128 = 203$

  => $11001011_2 = 203_{10}$
- Convert the binary number $(1101)_2$ into a decimal number.

$$(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 8 + 4 + 0 + 1 = 13$$

Decimal to Hexadecimal

- Convert decimal number 540 into hexadecimal number.

Division	Remainder (R)
540 / 16 = 33	12 = C
33 / 16 = 2	1
2 / 16 = 0	2
0 / 16 = 0	0

- Convert decimal fractional number 0.06640625 into hexadecimal number.

Multiplication	Resultant integer part
0.06640625 x 16=1.0625	1
0.0625 x 16 =1.0	1
0 x 16=0.0	0

0.110 is equivalent hexadecimal number of decimal fractional 0.06640625.

Hexadecimal to Decimal

- Convert hexadecimal number ABCDEF into decimal number.

$$(ABCDEF)_{16} = (10x16^5+11x16^4+12x16^3+13x16^2+14x16^1+15x16^0)_{10}$$

$$= (10485760+720896+49152+3328+224+15)_{10}$$
$$= (11259375)_{10}$$

- Convert hexadecimal number 1F.01B into decimal number.

$(1F.01B)_{16} = (1x16^{1}+15x16^{0}+0x16^{-1}+1x16^{-2}+11x16^{-3})_{10}$

$= (31.0065918)_{10}$ which is answer.

BINARY CODED DECIMAL

- In binary-coded decimal, each digit in a decimal base 10 number is represented as a group of four binary digits, or bits.
- Each digit is encoded separately.
    - The decimal number is 234.

      In binary-coded decimal, 234 is written as the following:
        - 2 = 0010, 3 = 0011, 4 = 0100
        - Hence, 234 = 0010 0011 0100
    - Decimal number = 1764 — BCD = 0001 0111 0110 0100
    - Decimal number = 1895 — BCD = 0001 1000 1001 0101

Arithmetic Operations On Binary Numbers

Addition

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

0011010 + 001100 = 00100110

```
  1 1        carry
  0011010  = 26₁₀
+ 0001100  = 12₁₀
---------
  0100110  = 38₁₀
```

Subtraction

Case	A - B	Subtract	Borrow
1	0 - 0	0	0
2	1 - 0	1	0
3	1 - 1	0	0
4	0 - 1	0	1

0011010 - 001100 = 00001110

```
  1 1      borrow
  0011010  = 26₁₀
- 0001100  = 12₁₀
---------
  0001110  = 14₁₀
```

Multiplication

Case	A x B	Multiplication
1	0 x 0	0
2	0 x 1	0
3	1 x 0	0
4	1 x 1	1

Example:

0011010 x 001100 = 100111000

```
  0011010  = 26₁₀
x 0001100  = 12₁₀
---------
  0000000
 0000000
0011010
0011010
---------
0100111000  = 312₁₀
```

Division

Binary division is similar to decimal division

101010 / 000110 = 000111

```
          111    = 7₁₀
       ---------
000110 ) 101010  = 42₁₀
       - 110     = 6₁₀
       ------
         1001
       - 110
       ------
          110
        - 110
        -----
            0
```

QUESTIONS

- 111 + 101 (b) 100 – 011 c) 1101 x 10 d) 1100÷1 1

$$\begin{array}{r} {\scriptstyle (1)\,(1)} \\ 1\,1\,1 \\ +\;1\,0\,1 \\ \hline 1\,1\,0\,0 \end{array} \qquad \begin{array}{r} 100 \\ -\,011 \\ \hline 001 \end{array} \qquad \begin{array}{r} 11101 \\ \times\,1\,0 \\ \hline 0000 \\ 1101 \\ \hline 11010 \end{array}$$

$$\begin{array}{r|l} & 100 \\ \hline 11 & 11000 \\ & 11 \\ \hline & 0\,0 \\ & 0\,0 \\ \hline & \quad 0\,0 \\ & \quad 0\,0 \\ \hline & \quad\quad 0\,0 \end{array}$$

NEGATIVE NUMBER REPRESENTATION

- Negative numbers can be distinguishable with the help of extra bit or flag called sign bit or sign flag in Binary number representation system for signed numbers.
- It is not possible to add minus or plus symbol in front of a binary number because a binary number can have only two symbol either 0 or 1 for each position or bit.
- That's why we use this extra bit called sign bit or sign flag.
- The value of sign bit is 1 for negative binary numbers and 0 for positive numbers.
- When an integer binary number is positive, the sign is represented by 0 and the magnitude by a positive binary number.
- When the number is negative, the sign is represented by 1 but the rest of the number may be represented in any one of the following ways:
    - 1's Complement method
    - 2's complement method.

1's Complement Method:

- If the number is negative then it is represented using 1's complement.
- First represent the number with positive sign and then take 1's complement of that number.
- To get 1's complement of a binary number, simply invert the given number.
    - Find 1's complement of binary number 10101110. => 01010001
    - Find 1's complement of binary number 10001.001. =>01110.110

Two's Complement Method

- To get 2's complement of a binary number, simply invert the given number and add 1 to the least significant bit (LSB) of given result.
    - Find 2's complement of binary number 10101110.
        - Simply invert each bit of given binary number, which will be 01010001.
        - Then add 1 to the LSB of this result.
        - i.e., 01010001+1=01010010
    - Find 2's complement of binary number 10001.001.
        - Simply invert each bit of given binary number, which will be 01110.110
        - Then add 1 to the LSB of this result.
        - i.e., 01110.110+1=01110.111
- -5 is represented using the following steps:

(i) +5 = 0 0101

(ii) Take 2's complement of 0 0101 and that is 1 1011.

MSB is 1 which indicates that number is negative.

- MSB is always 1 in case of negative numbers.

Binary Subtraction by 2's Complement

The algorithm to subtract two binary number using 2's complement is explained as following below –

- Take 2's complement of the subtrahend
- Add with minuend
- If the result of above addition has carry bit 1, then it is dropped and this result will be positive number.
- If there is no carry bit 1, then take 2's complement of the result which will be negative

Example (Case-1: When Carry bit 1) –Evaluate 10101 - 00101

- According to above algorithm, take 2's complement of subtrahend 00101, which will be 11011.
- Then add both of these. So, 10101 + 11011 =1 10000.
- Since, there is carry bit 1, so dropped this carry bit 1.
- Take this result will be 10000 will be positive number.

Example (Case-2: When no Carry bit) –Evaluate 11001 - 11100

- According to above algorithm, take 2's complement of subtrahend 11100, which will be 00100.
- Then add both of these, So, 11001 + 00100 =11101.
- Since there is no carry bit 1, so take 2's complement of above result, which will be 00011, and this is negative number, i.e, 00011, which is the answer.

LOGIC GATES

- Logic gates are the basic building blocks of any digital system.
- It is an electronic circuit having one or more than one input and only one output.
- The relationship between the input and the output is based on a **certain logic**.
- Based on this, logic gates are named as AND gate, OR gate, NOT gate etc.

BASIC LOGIC GATES AND GATE

- In the AND gate, the output of an AND gate attains state 1 if and only if all the inputs are in state 1.
- The Boolean expression of AND gate is Y = A.B

Inputs		Output
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

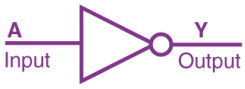
OR GATE

- In an OR gate, the output of an OR gate attains state 1 if one or more inputs attain state 1.
- The Boolean expression of the OR gate is Y = A + B, (read as Y equals A 'OR' B.)

Inputs		Output
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE

- In a NOT gate, the output of a NOT gate attains state 1 if and only if the input does not attain state 1.
- The Boolean expression is: $Y = A^{-}$ (read as Y equals NOT A.)



Inputs	Output
A	B
0	1
1	0

NAND GATE

- Combination of NOT and AND Gates.
- Universal gate.
- The Boolean expression of the NAND gate is:

$$Y = \overline{A.B}$$

Inputs		Output
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE

- Combination of NOT and OR Gates.
- Universal gate.
- The Boolean expression of NOR gate is:

$$Y = \overline{A + B}$$

Inputs		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

XOR GATE

- In an XOR gate, the output of a two-input XOR gate attains state 1 if one adds only input attains state 1.
- The Boolean expression of the XOR gate is:

$$A.\bar{B} + \bar{A}.B$$

or

$$Y = A \oplus B$$

Inputs		Output
A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Boolean algebra -important laws

There are six types of Boolean algebra laws. They are:

- Commutative law
- Associative law
- Distributive law
- AND law
- OR law
- Inversion law

Commutative Law

Any binary operation which satisfies the following expression is referred to as a commutative operation. Commutative law states that changing the sequence of the variables does not have any effect on the output of a

logic circuit.

A. B = B. A A + B = B + A

Associative Law

It states that the order in which the logic operations are performed is irrelevant as their effect is the same.

(A.B).C = A.(B.C) (A+B)+C = A+(B+C)

Distributive Law

Distributive law states the following conditions:

A .(B + C) = (A . B) + (A . C) A + (B . C) = (A + B) . (A + C)

AND Law

These laws use the AND operation. Therefore they are called AND laws.

A .0 = 0 A . 1 = A A. A = A

OR Law

These laws use the OR operation. Therefore they are called OR laws.

A + 0 = A A + 1 = 1 A + A = A

Inversion Law

In Boolean algebra, the inversion law states that double inversion of variable results in the original variable itself.

$$\bar{\bar{A}} = A$$

Demorgan's theorems

De Morgan's First Law

The first law states that the complement of the product of the variables is equal to the sum of their individual complements of a variable

$$\overline{A.B} = \overline{A}+\overline{B}.$$

DeMorgan's Second Theorem

According to the second theorem, the complement result of the OR operation is equal to the AND operation of the complement of that variable.

$$\overline{A+B} = \overline{A}.\overline{B},$$

SIMPLIFICATION OF EXPRESSIONS

Simplify: $A(A+\overline{B}C)+A(\overline{B}+C)$

Solution, $F= A(A+\overline{B}C)+A(\overline{B}+C)$

$= A.A+A\overline{B}C+A\overline{B}+AC$ — $A.A=A$

$= A+A\overline{B}C+A\overline{B}+AC$

$= A+A\overline{B}(C+1)+AC$ — $A+1=1$

$= A+A\overline{B}.1+AC$ — $A.1=A$

$= A+A\overline{B}+AC$

$= A(1+\overline{B})+AC$ — $A+1=1$

$= A.1+AC$ — $A.1=A$

$= A+AC$

$= A(1+C)$ — $A+1=1$

$= A.1$ — $A.1=A$

$= A$

Simplify: $ABC+A\overline{B}C+AB\overline{C}$

Solution, $F= ABC+A\overline{B}C+AB\overline{C}$

$= AC(B+\overline{B})+AB\overline{C}$

$= AC.1+AB\overline{C}$ — $A+\overline{A}=1$

$= AC+AB\overline{C}$ — $A.1=A$

$= A(C+B\overline{C})$

$= A(C+B)$ — $A+\overline{A}B=A+B$

Simplify: $A\overline{B}+\overline{(\overline{A}+\overline{B}+C.\overline{C})}$

Solution, $F= A\overline{B}+\overline{(\overline{A}+\overline{B}+C.\overline{C})}$

$= A\overline{B}+\overline{(\overline{A}+\overline{B}+0)}$ — $A.\overline{A}=0$

$= A\overline{B}+\overline{(\overline{A}+\overline{B})}$ — $A+0=A$

$= A\overline{B}+(\overline{\overline{A}}.\overline{\overline{B}})$ — $\overline{A+B}=\overline{A}.\overline{B}$

$= A\overline{B}+A.B$ — $\overline{\overline{A}}=A$

$= A(\overline{B}+B)$

$= A.1$ — $A+\overline{A}=1$

$= A$ — $A.1=A$

Simplify: $AB+\overline{A}B+A\overline{B}$

Solution, $F= AB+\overline{A}B+A\overline{B}$

$= B(A+\overline{A})+A\overline{B}$ — $A+\overline{A}=1$

$= B.1+A\overline{B}$ — $A.1=A$

$= B+A\overline{B}$ — $A+BC=(A+B)\ (A+C)$

$= (B+A)(B+\overline{B})$ — $A+\overline{A}=1$

$= (B+A).1$ — $A.1=A$

$= B+A$ — $A+B=B+A$

$= A+B$

Boolean Function Representations

- Digital circuits use digital signals to operate.
- These signals have binary values; they can be either one or zero.
- Zero represents false or low state whereas one represents true or high state.
- Boolean algebra helps to describe the binary numbers and binary variables.
- To be more specific, a Boolean function is an algebraic form of Boolean expression.
- It is also possible to simplify Boolean functions of digital circuits using Boolean laws and theorems
- SOP and POS are two methods of representing Boolean expressions.

SOP

- SOP stands for Sum of Product.
- SOP form is a set of product(AND) terms that are summed(OR) together.
- When an expression or term is represented in a sum of binary terms known as minterms and sum of products.

$$Y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

POS

- POS stands for product of sum.
- A technique of explaining a Boolean expression through a set of max terms or sum terms, is known as POS(product of sum).

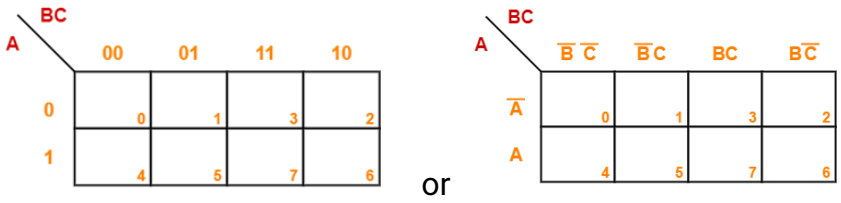
$$Y = (A+B+C)\left(A+B+\overline{C}\right)\left(A+\overline{B}+C\right)\left(\overline{A}+B+C\right)$$

K Map

- A Karnaugh map or a K-map refers to a pictorial method that is utilized to minimize various Boolean expressions without using the Boolean algebra theorems along with the equation manipulations.
- It is a method for simplification of Boolean functions in an easy way.
- It is a graphical method, which consists of 2^n cells for 'n' variables. The adjacent cells are differed only in single bit position.

3 Variable K-Map

- The number of cells in 3 variable K-map is eight, since the number of variables is three.
- The following figures show **3 variable K-Map**.



Solve using kmap

- Z= ∑A,B,C(1,3,6,7) F(A, B, C)= ∑m(0,4,5,7)