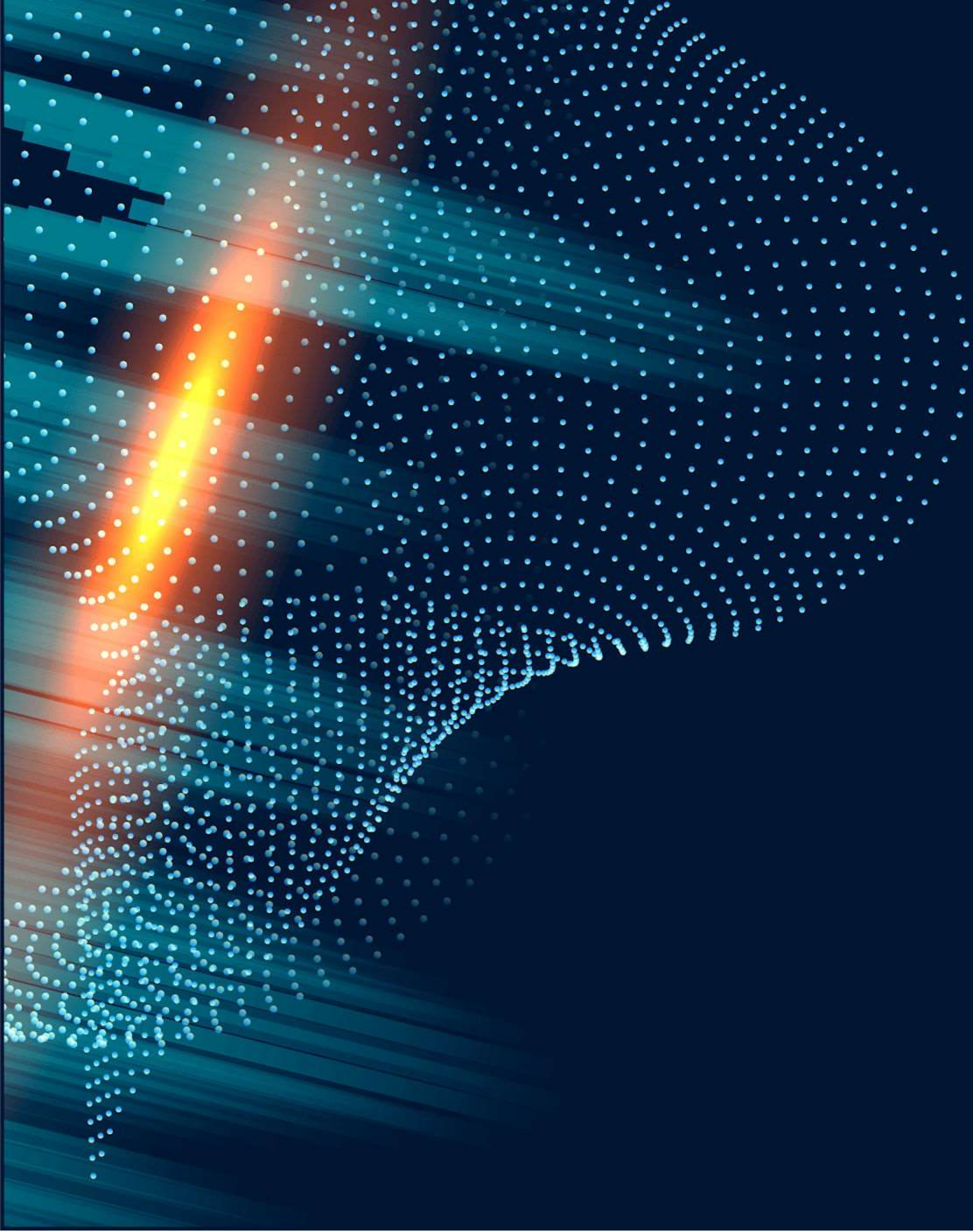
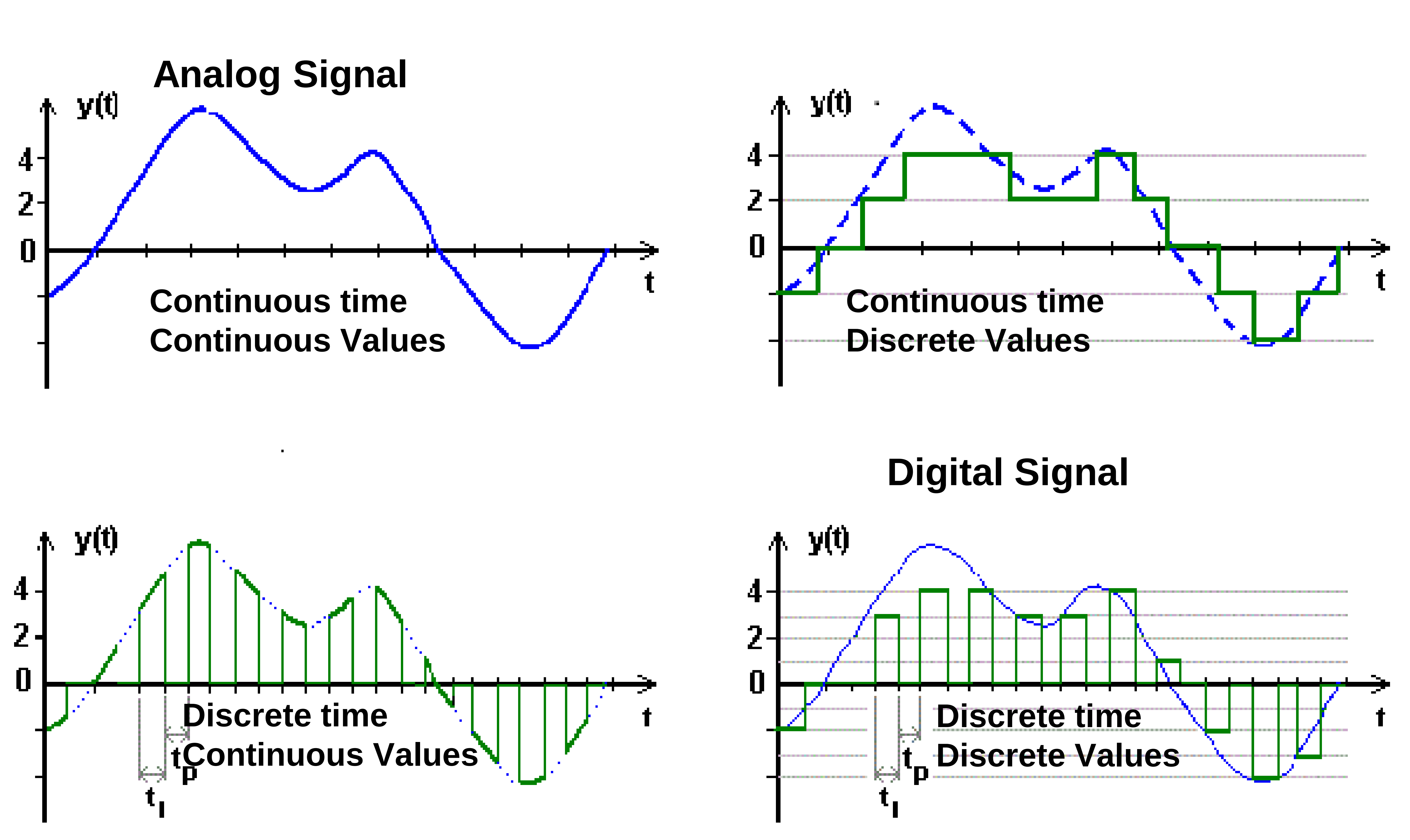
Digital Design

Introduction to digital electronics



Analog vs Digital

Analog vs Digital



Analog vs Digital

1. Analog Signals

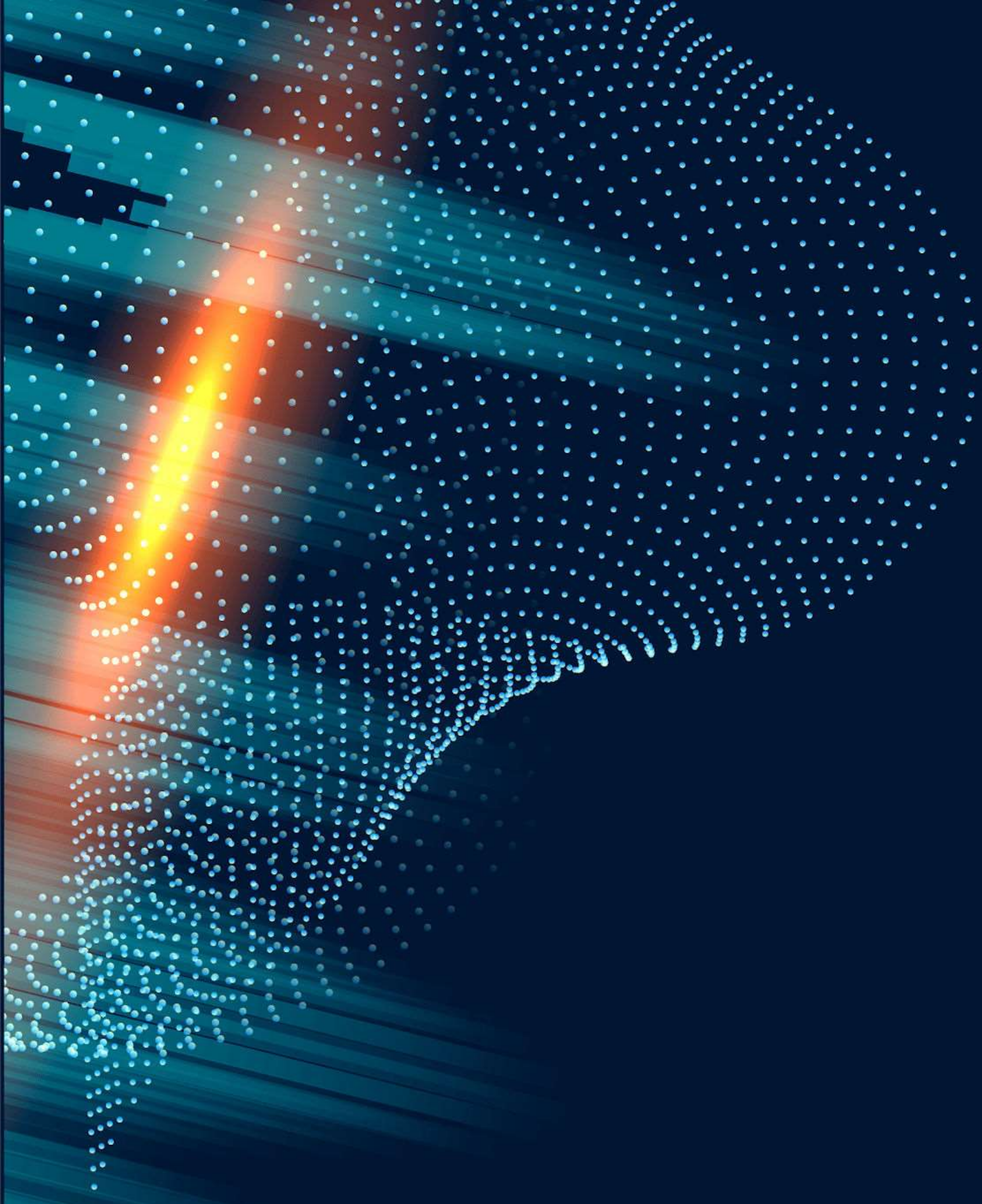
Definition: Analog signals are continuous signals that vary over time. They can take on any value within a given range.
Characteristics:
Represented by waveforms, such as sine waves.
Examples include sound waves, temperature readings, and voltage levels.
They can be affected by noise, which can distort the signal.

2. Discrete Signals

Definition: Discrete signals are signals that are defined at specific intervals or distinct points in time. Unlike analog signals, they do not take on values in between these points.
Characteristics:
Often represented as a series of separate values or samples.
Examples include digital representations of audio samples or data points in a digital chart.
Discrete signals can be derived from analog signals through a process called sampling.

3. Digital Signals

Definition: Digital signals are a type of discrete signal that represents information in binary form (0s and 1s). They are quantized and can only take on specific values.
Characteristics:
More resistant to noise and degradation compared to analog signals.
Examples include computer data, digital audio, and video files.
Digital signals can be easily stored, processed, and transmitted by digital devices.

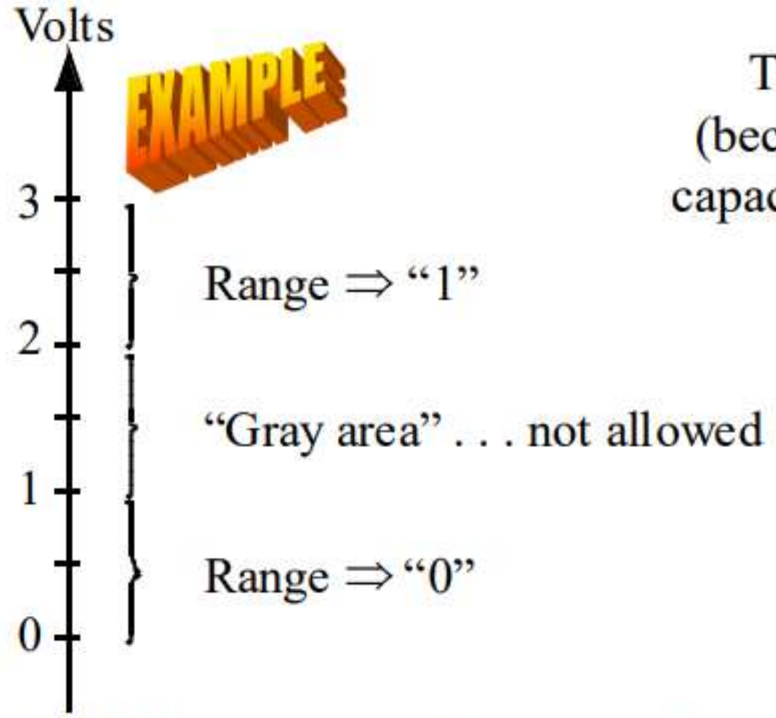


Why Digital ??

Logic gates are *electronic circuits* that process *electrical* signals

Most common signal for logic variable: voltage

Specific voltage ranges correspond to "0" or "1"

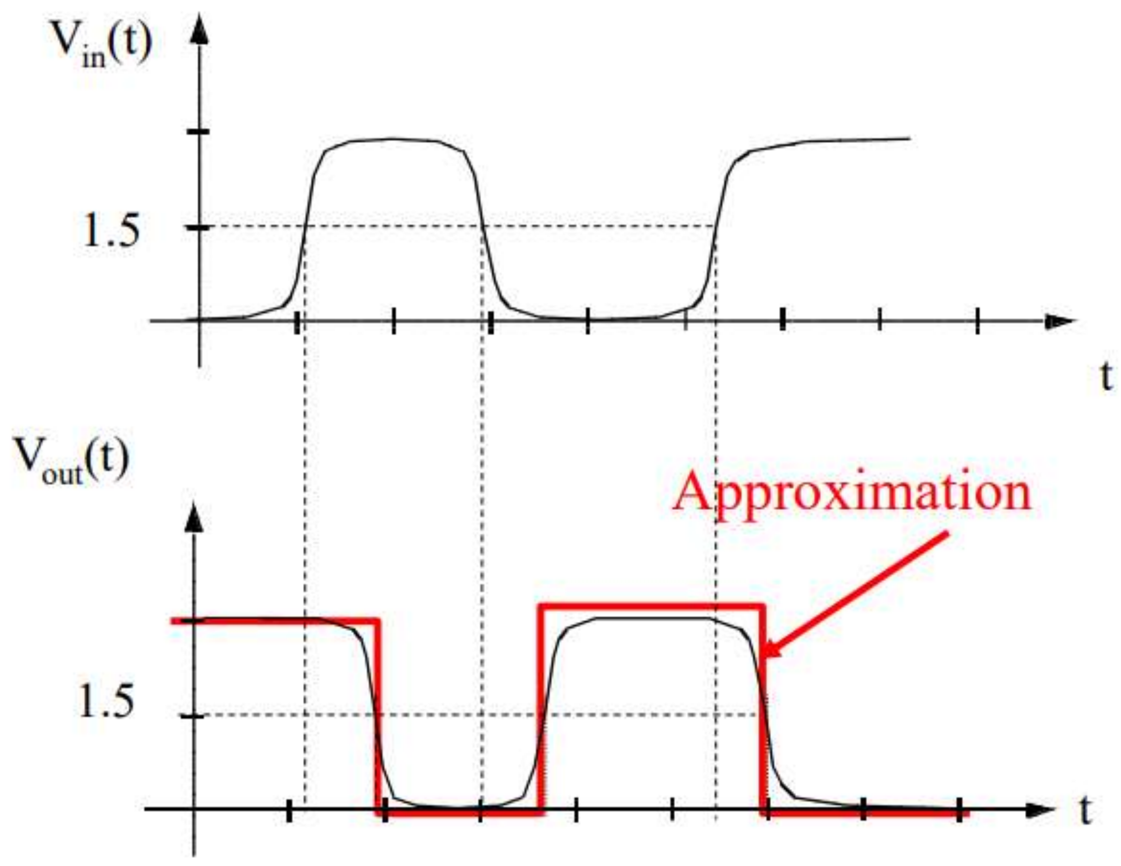


Thus delay in voltage rise or fall (because of delay in charging internal capacitances) will translate to a delay in signal timing

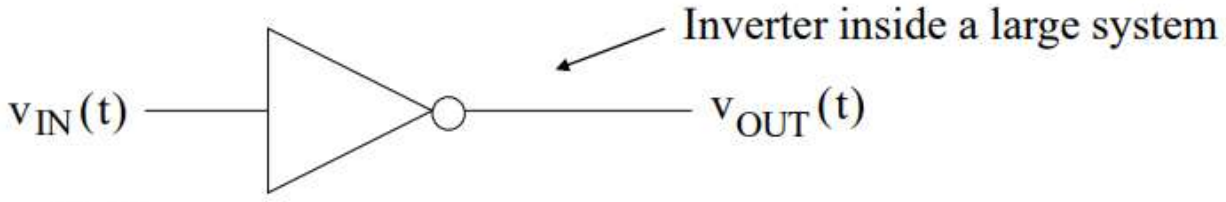
Note that the specific voltage range for 0 or 1 depends on "logic family," and in general decreases with succeeding logic generations

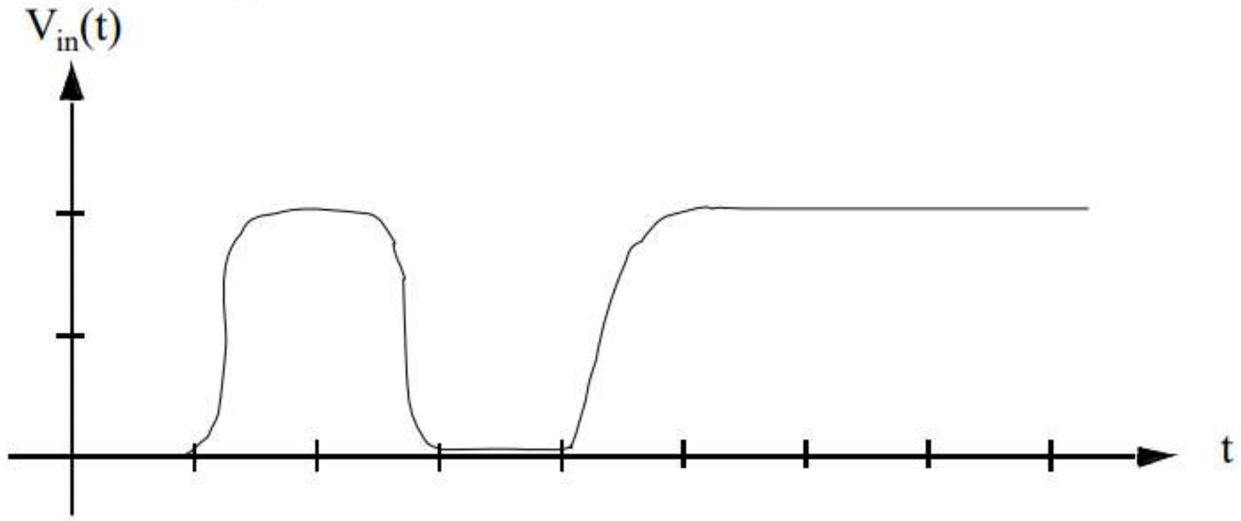
Define τ as the delay required for the output voltage to reach 50% of its final value.
In this example we will use 3V logic, so halfway point is 1.5V.

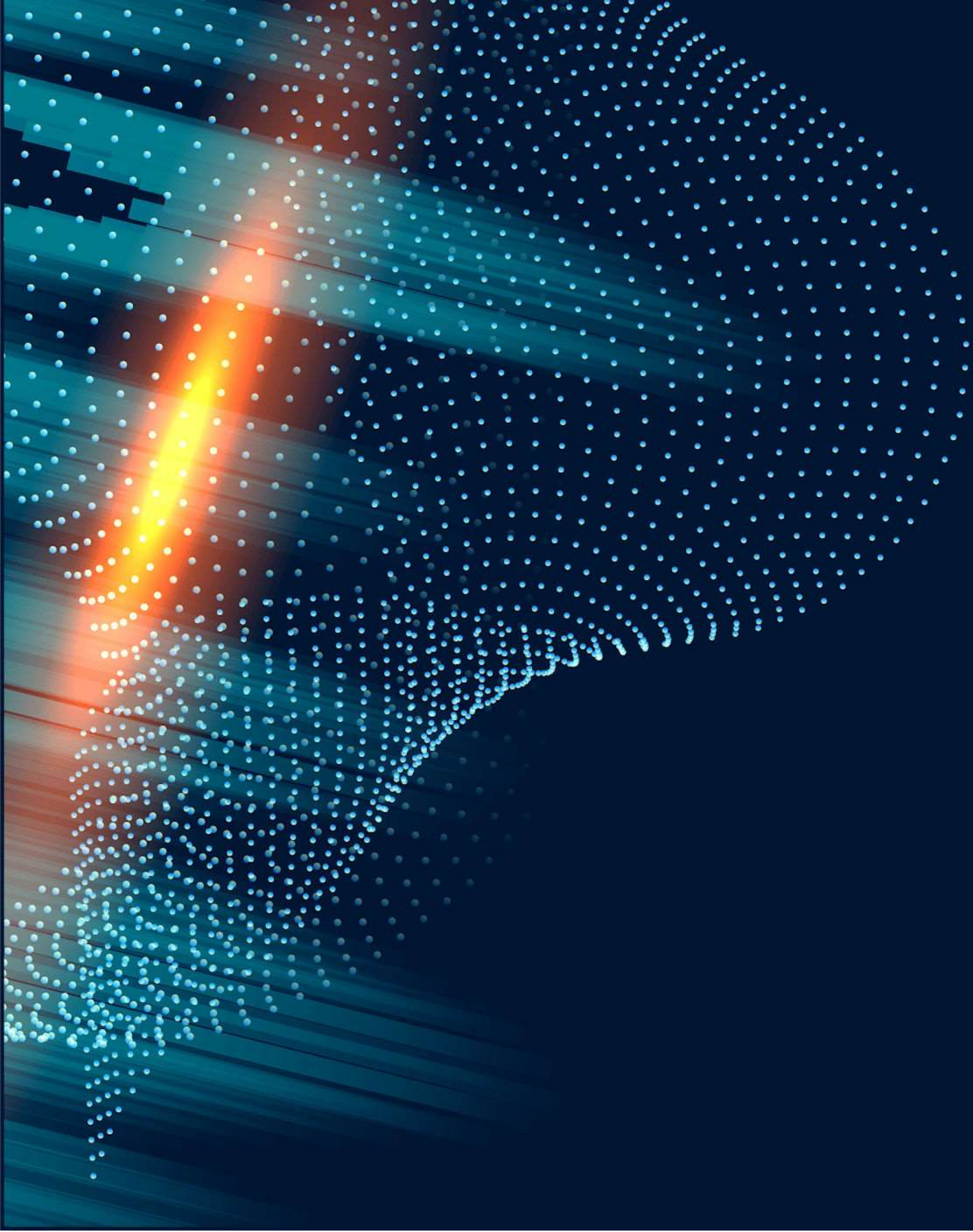
Inverters are designed so that the gate delay is symmetrical (rise and fall)



Inverter input is $v_{IN}(t)$, output is $v_{OUT}(t)$







Number System

Number System

Numbers

$$A_{n-1}A_{n-2}\dots A_1A_0.A_{-1}A_{-2}\dots A_{-m+1}A_{-m}$$

- The . is called the radix point
- A_{n-1} : most significant digit (msd)
- A_{-m} : least significant digit (lsd)

$$(724.5)_{10} = 724.5 = 7\times10^2 + 2\times10^1 + 4\times10^0 + 5\times10^{-1}$$

$$1620.375 = 1\times10^3 + 6\times10^2 + 2\times10^1 + 3\times10^{-1} + 7\times10^{-2} + 5\times10^{-3}$$

$$(312.4)_5 = 3\times5^2 + 1\times5^1 + 2\times5^0 + 4\times5^{-1} = (82.8)_{10}$$

In addition to decimal, three number systems are important: Binary, Octal, and Hexadecimal

Numbers

- Each number system is associated with a base or radix
  - The decimal number system is said to be of base or radix 10
- A number in *base r* contains r digits 0,1,2,...,r-1
  - Decimal (Base 10): 0,1,2,3,4,5,6,7,8,9
- Numbers are usually expressed in positional notation

$$A_{n-1}A_{n-2}\dots A_1A_0.A_{-1}A_{-2}\dots A_{-m+1}A_{-m}$$

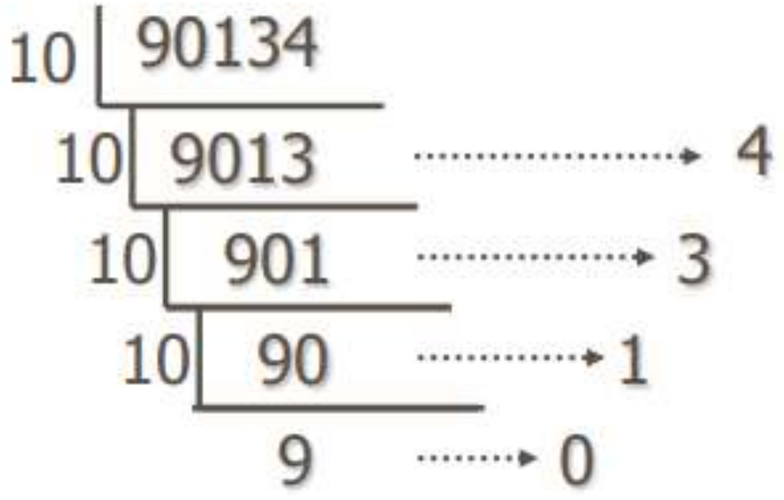
- A number is expressed as a power series in *r* with the general form

$$A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \dots + A_1r^1 + A_0r^0 + A_{-1}r^{-1} + A_{-2}r^{-2} + \dots + A_{-m+1}r^{-m+1}A_{-m}r^{-m}$$

Number System

Decimal Number Representation

- Example: 90134 (base-10, used by Homo Sapien)
  - $= 90000 + 0 + 100 + 30 + 4$
  - $= 9*10^4 + 0*10^3 + 1*10^2 + 3*10^1 + 4*10^0$
- How did we get it?



Generic Number Representation

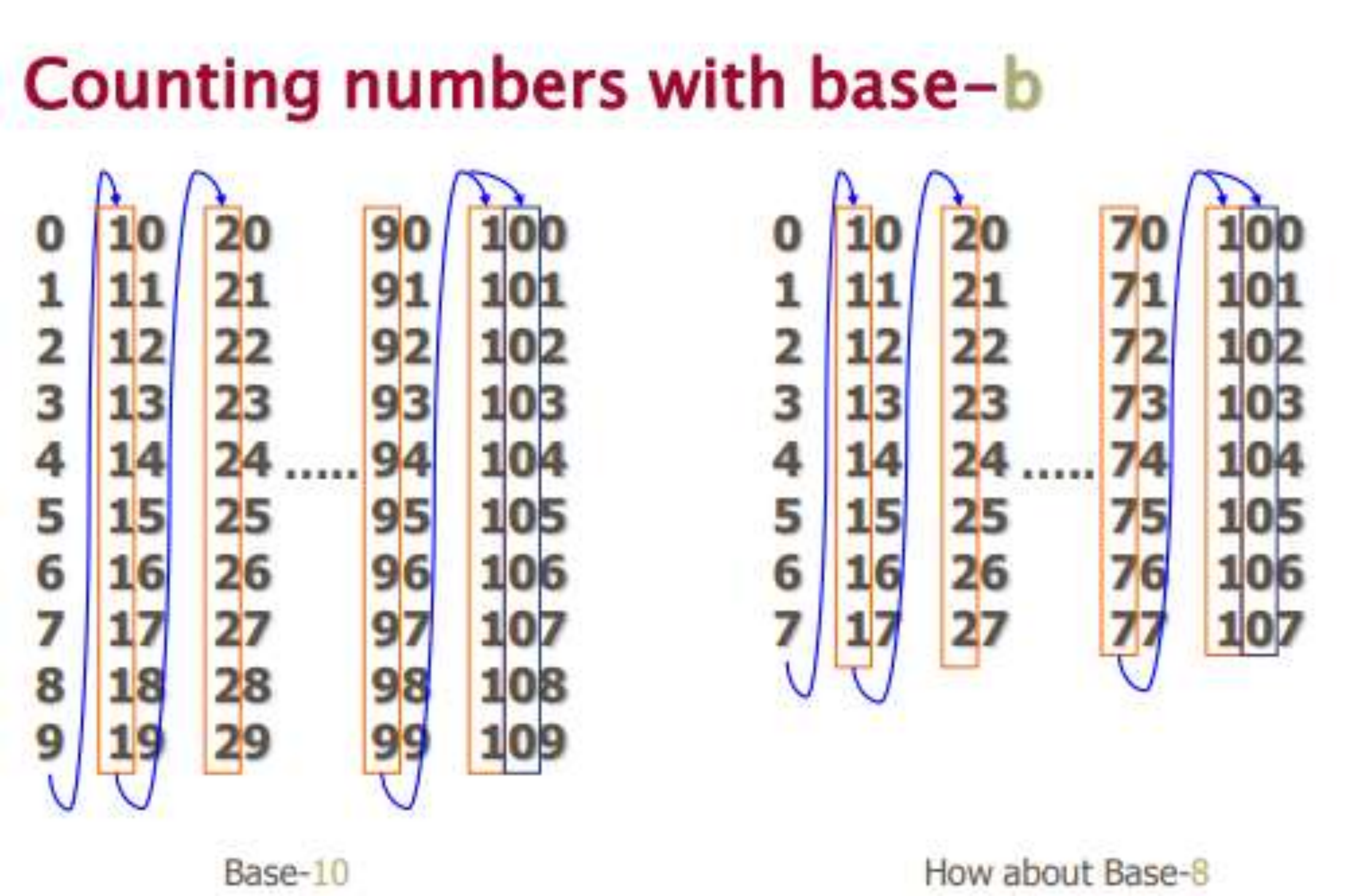
- 90134
  - $= 9*10^4 + 0*10^3 + 1*10^2 + 3*10^1 + 4*10^0$
- $A_4\ A_3\ A_2\ A_1\ A_0$ for base-10 (or radix-10)
  - $= A_4*10^4 + A_3*10^3 + A_2*10^2 + A_1*10^1 + A_0*10^0$
  - (A is coefficient; b is base)
- Generalize for a given number **N** w/ base-b

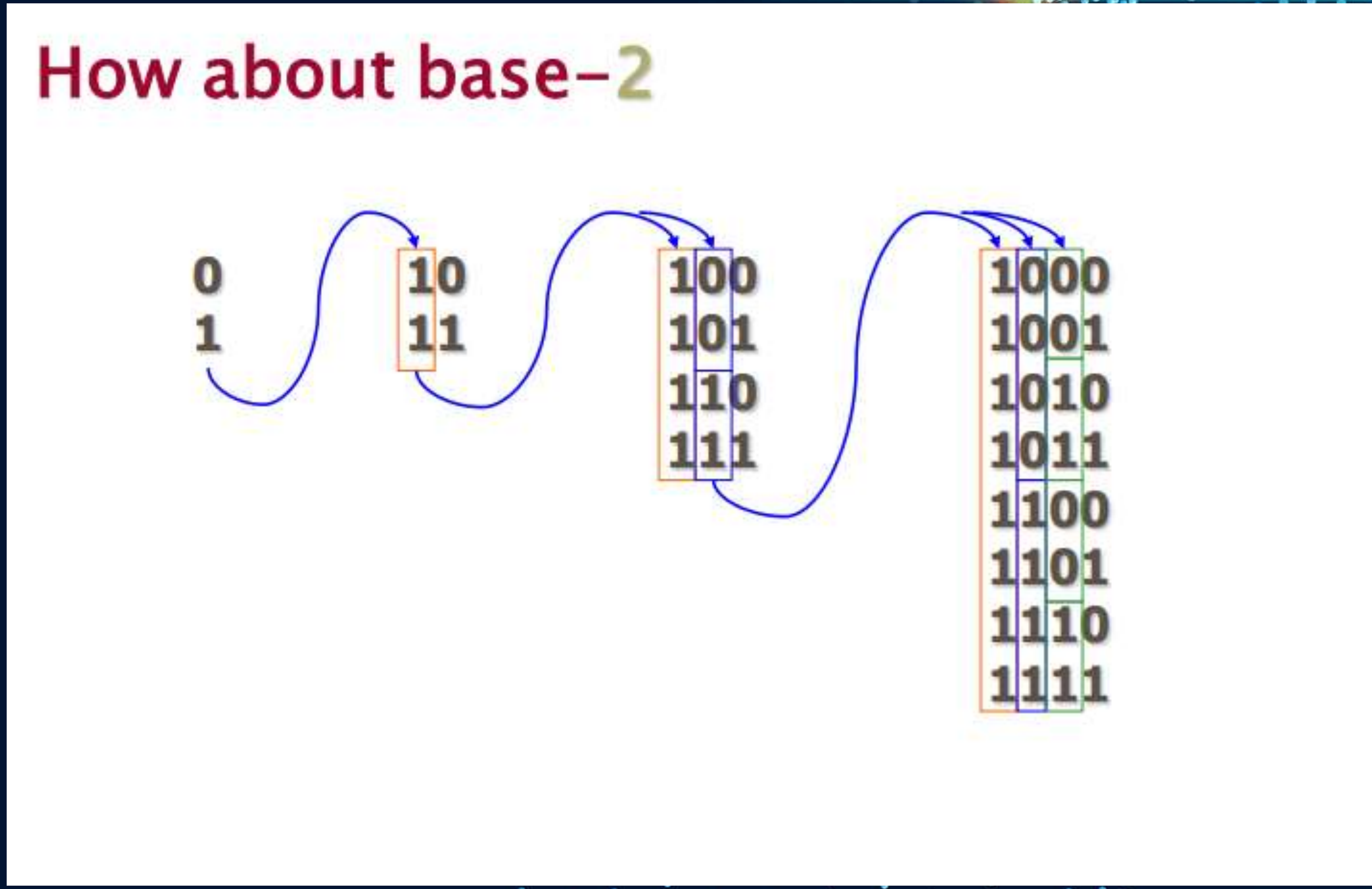
$$\mathbf{N} = A_{n-1}\ A_{n-2} \ldots A_1\ A_0$$

$$\mathbf{N} = A_{n-1}*b^{n-1} + A_{n-2}*b^{n-2} + \ldots + A_2*b^2 + A_0*b^0$$

**Note that $A < b$

Number System





Number System

How about base-2

0 = 0	10 = 2	100 = 4	1000 = 8
1 = 1	11 = 3	101 = 5	1001 = 9
		110 = 6	1010 = 10
		111 = 7	1011 = 11
			1100 = 12
			1101 = 13
			1110 = 14
			1111 = 15

Binary = Decimal

Number Examples with Different Bases

- Decimal (base-10)
  - $(982)_{10}$
- Binary (base-2)
  - $(01111010110)_{2}$
- Octal (base-8)
  - $(1726)_{8}$
- Hexadecimal (base-16)
  - $(3d6)_{16}$
- Others examples:
  - base-9 = $(1321)_{9}$
  - base-11 = $(813)_{11}$
  - base-17 = $(36d)_{17}$

Number System

Derive Numbers in Base-2

- Decimal (base-10)
  - $(25)_{10}$
- Binary (base-2)
  - $(11001)_2$
- Exercise

2	25		
2	12	→	1
2	6	→	0
2	3	→	0
	1	→	1

Convert between different bases

- Convert a number base-x to base-y, e.g. $(0100111)_2$ to $(?)_6$
  - First, convert from base-x to base-10 if $x \neq 10$
  - Then convert from base-10 to base-y

$$0100111 = 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 39$$

6	39		
6	6	→	3
	1	→	0

$$\therefore (0100111)_2 = (103)_6$$

Number System

Converting Binary to Decimal

- For example, here is 1101.01 in binary:

1	1	0	1	.	0	1	Bits
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	Weights (in base 10)

$$2^{10}: K(kilo);\quad 2^{20}: M(mega);\quad 2^{30}: G(giga)$$

- The decimal value is:

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) =$$

$$8 + 4 + 0 + 1 + 0 + 0.25 = 13.25$$

Converting Decimal to Binary

- To convert a decimal integer into binary, keep dividing by 2 until the quotient is 0. Collect the remainders in *reverse* order
- To convert a fraction, keep multiplying the fractional part by 2 until it becomes 0. Collect the integer parts in forward order
- Example: 162.375:

162 / 2 = 81 rem 0
81 / 2 = 40 rem 1
40 / 2 = 20 rem 0
20 / 2 = 10 rem 0
10 / 2 = 5 rem 0
5 / 2 = 2 rem 1
2 / 2 = 1 rem 0
1 / 2 = 0 rem 1

0.375 x 2 = 0.750
0.750 x 2 = 1.500
0.500 x 2 = 1.000

- So, $(162.375)_{10} = (10100010.011)_2$

Number System

Octal and Hexadecimal Numbers

- The octal number system: Base-8
  - Eight digits: 0,1,2,3,4,5,6,7

$$(127.4)_8 = 1\times8^2 + 2\times8^1 + 7\times8^0 + 4\times8^{-1} = (87.5)_{10}$$

- We use Base-16 (or Hex) a lot in computer world
  - Sixteen digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  - Ex: A 32-bit address can be written as
    - `0xfe8a7d20` (`0x` is an abbreviation of Hex)
    - Or in binary form `1111_1110_1000_1010_0111_1101_0010_0000`

$$(B65F)_{16} = 11\times16^3 + 6\times16^2 + 5\times16^1 + 15\times16^{-0} = (46687)_{10}$$

- For our purposes, base-8 and base-16 are most useful as a "shorthand" notation for binary numbers

Numbers with Different Bases

Decimal	Binary	Octal	Hex
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

You can convert between base-10 base-8 and base-16 using techniques like the ones we just showed for converting between decimal and binary

Number System

Binary and Octal Conversions

- Converting from octal to binary: Replace each octal digit with its equivalent 3-bit binary sequence

$$(673.12)_8 = 6\ \ 7\ \ 3\ .\ 1\ \ 2$$
$$= 110\ \ 111\ \ 011\ .\ 001\ \ 010$$
$$= (110111011.001010)_2$$

- Converting from binary to octal: Make groups of 3 bits,starting from binary point. Add 0s to ends of number if needed. Convert each bit group to its corresponding octal digit.

$$10110100.001011_2 = 010\ \ 110\ \ 100\ .\ 001\ \ 011_2$$
$$= 2\ \ 6\ \ 4\ .\ 1\ \ 3_8$$

Octal	Binary
0	000
1	001
2	010
3	011

Octal	Binary
4	100
5	101
6	110
7	111

Binary and Hex Conversions

- Converting from hex to binary: Replace each hex digit with its equivalent 4-bit binary sequence

$$261.35_{16} = 2\ \ 6\ \ 1\ .\ 3\ \ 5_{16}$$
$$= 0010\ \ 0110\ \ 0001\ .\ 0011\ \ 0101_2$$

- Converting from binary to hex: Make groups of 4 bits, starting from the binary point. Add 0s to the ends of the number if needed. Convert each bit group to its corresponding hex digit

$$10110100.001011_2 = 1011\ \ 0100\ .\ 0010\ \ 1100_2$$
$$= B\ \ 4\ .\ 2\ \ C_{16}$$

Hex	Binary
0	0000
1	0001
2	0010
3	0011

Hex	Binary
4	0100
5	0101
6	0110
7	0111

Hex	Binary
8	1000
9	1001
A	1010
B	1011

Hex	Binary
C	1100
D	1101
E	1110
F	1111

Number System

Information Representation (cont.)

- Various Codes used in Computer Industry
  - Number-only representation
    - BCD (4 bits per decimal number)
  - Alpha-numeric representation
    - ASCII (7 bits)
  - Unicode (16 bit)

Review of Numbers

- Computers are made to deal with numbers
- What can we represent in N bits?
  - Unsigned integers:

    0 to $2^N - 1$
  - Signed Integers (Two's Complement)

    $-2^{(N-1)}$ to $2^{(N-1)} - 1$

Signed Integers

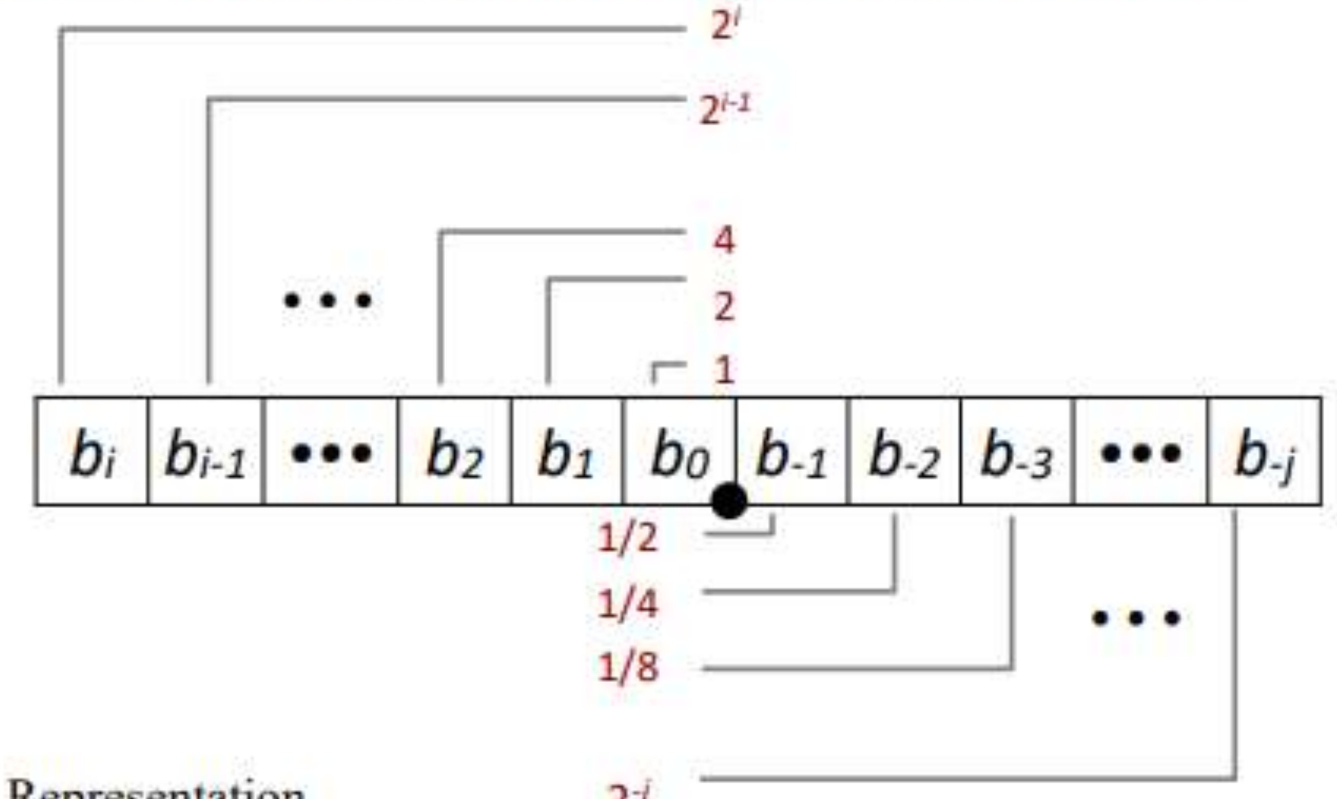
$-2^{(N-1)} - 1$ to $2^{(N-1)} - 1$

Number System

Other Numbers

- What about other numbers?
  - Very large numbers? (seconds/century)
    $3{,}155{,}760{,}000_{10}$ (3.15576_{10} x 10^9)
  - Very small numbers? (atomic diameter)
    0.00000001_{10} (1.0_{10} x 10^{-8})
  - Rationals (repeating pattern)
  - 2/3 (0.666666666. . .)
  - Irrationals
    $2^{1/2}$ (1.414213562373. . .)
  - Transcendentals
  - e (2.718...), π (3.141...)
- All represented in scientific notation

Fractional Binary Numbers



- Representation
  - Bits to right of "binary point" represent fractional powers of 2
  - Represents rational number: $\sum_{k=-j}^{i} b_k \times 2^k$

Number System

Fractional Binary Numbers: Examples

- **Value** / **Representation**

Value		Representation	
5 3/4	= 23/4	101.11_2	= 4 + 1 + 1/2 + 1/4
2 7/8	= 23/8	10.111_2	= 2 + 1/2 + 1/4 + 1/8
1 7/16	= 23/16	1.0111_2	= 1 + 1/4 + 1/8 + 1/16

- **Observations**
  - Divide by 2 by shifting right (unsigned)
  - Multiply by 2 by shifting left
  - Numbers of form $0.111111\ldots_2$ are just below 1.0
    - $1/2 + 1/4 + 1/8 + \ldots + 1/2^i + \ldots \rightarrow 1.0$
    - Use notation $1.0 - \varepsilon$

Representable Numbers

- Limitation #1
  - Can only exactly represent numbers of the form $x/2^k$
    - Other rational numbers have repeating bit representations

Value	Representation
1/3	$0.0101010101[01]\ldots_2$
1/5	$0.001100110011[0011]\ldots_2$
1/10	$0.0001100110011[0011]\ldots_2$

- Limitation #2
  - Just one setting of binary point within the *w* bits
    - Limited range of numbers (very small values? very large?)

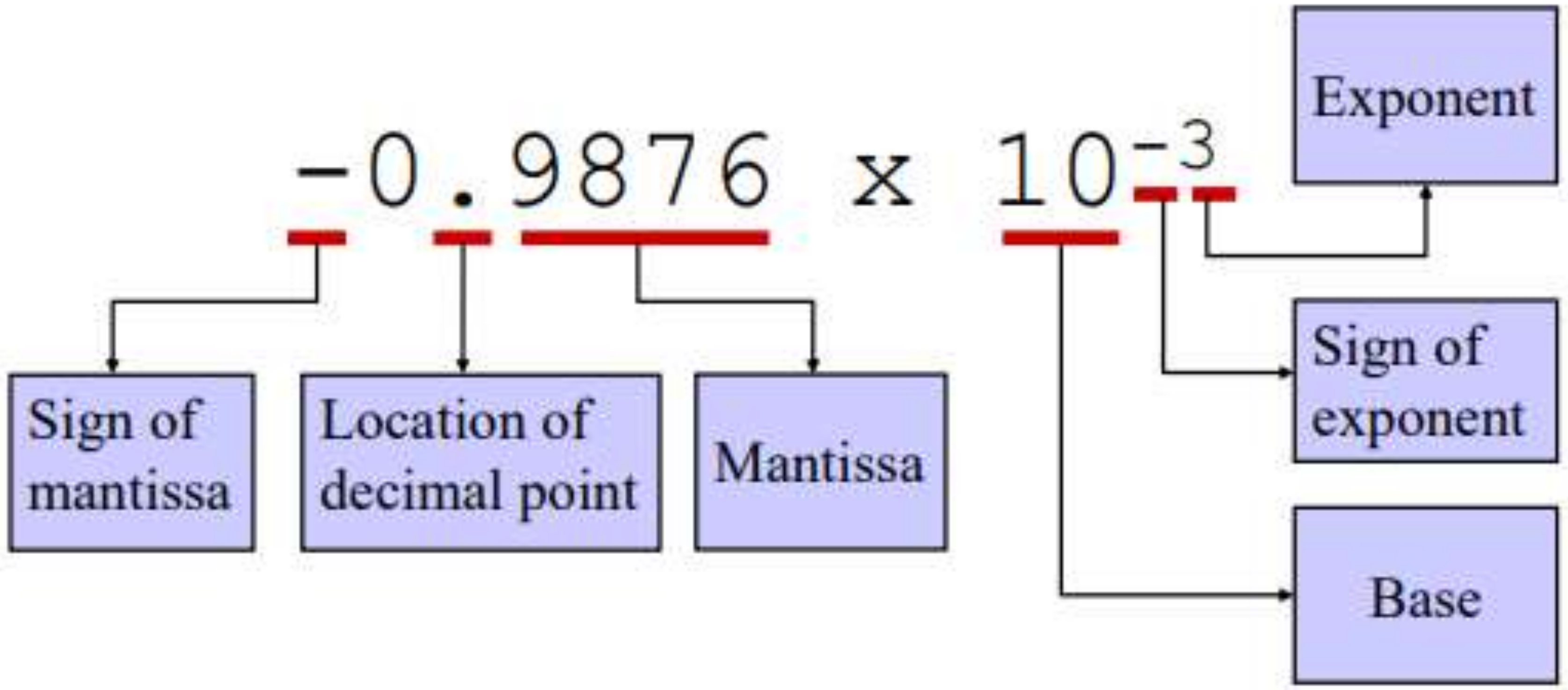
Number System

Floating Point Representation

Example:
$15213_{10} = (-1)^0 \times 1.1101101101101_2 \times 2^{13}$

- Numerical Form:
  $(-1)^s\, \boldsymbol{M}\ 2^{E}$
  - **Sign bit *s*** determines whether number is negative or positive
  - **Significand *M*** normally a fractional value in range [1.0,2.0).
  - **Exponent *E*** weights value by power of two
- Encoding
  - MSB s is sign bit ***s***
  - exp field encodes ***E*** (but is not equal to E)
  - frac field encodes ***M*** (but is not equal to M)

Parts of a Floating Point Number



Number System

[Check this out – Youtube video]

IEEE 754 Standard

- Most common standard for representing floating point numbers
- Single precision: 32 bits, consisting of...
  - Sign bit (1 bit)
  - Exponent (8 bits)
  - Mantissa (23 bits)
- Double precision: 64 bits, consisting of…
  - Sign bit (1 bit)
  - Exponent (11 bits)
  - Mantissa (52 bits)

Prof. Willian Kahan

Number System

Negative Number Representation

- Options
  - Sign–magnitude
  - One's Complement
  - Two's Complement (we use this in this course)

Sign–magnitude

- Use the most significant bit (MSB) to indicate the sign
  - 0: positive, 1: negative
- Problem
  - Representing zeros?
  - Do not work in computation

+0	000
+1	001
+2	010
+3	011
–3	111
–2	110
–1	101
–0	100

Number System

One's Complement

- Complement (flip) each bit in a binary number
- Problem
  - Representing zeros?
  - Do not always work in computation
    - Ex: 111 + 001 = 000 → Incorrect !

+0	000
+1	001
+2	010
+3	011
−3	100
−2	101
−1	110
0	111

Two's Complement

- Complement (flip) each bit in a binary number and add 1, with overflow ignored
- Work in computation perfectly

3 011
↓ One's complement
100
↓ Add 1
-3 101

-3 101
↓ One's complement
010
↓ Add 1
3 011

Number System

Two's Complement

- Complement (flip) each bit in a binary number and adding 1, with overflow ignored
- Work in computation perfectly
- We will use it in this course !

100
One's complement
011
Add 1
100

The same 100 represents both 4 and -4 which is no good

0	000
+1	001
−1	111
+2	010
−2	110
+3	011
−3	101
??	100

Two's Complement

- Complement (flip) each bit in a binary number and adding 1, with overflow ignored
- Work in computation perfectly
- We will use it in this course !

100
One's complement
011
Add 1
100

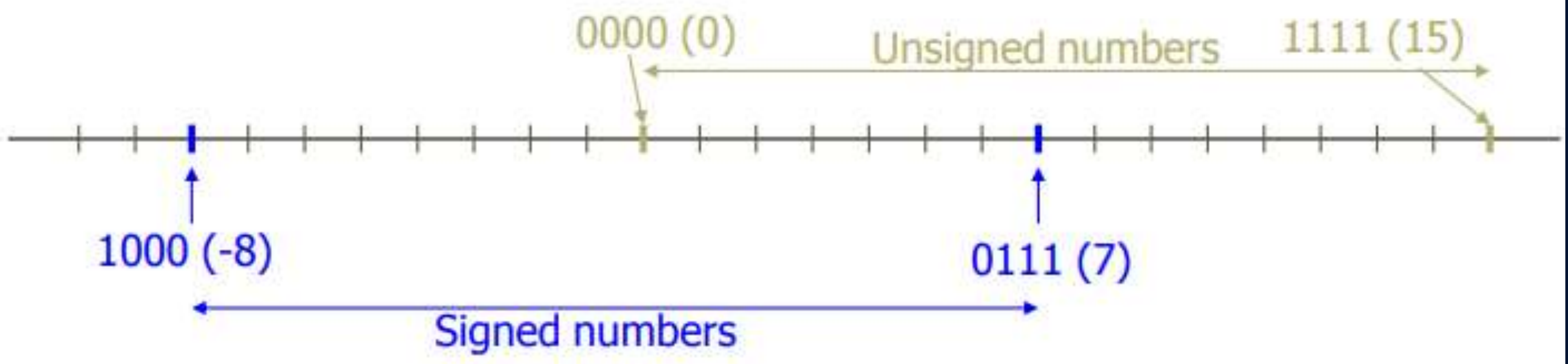
MSB = 1 for negative Number, thus 100 represents -4

0	000
+1	001
−1	111
+2	010
−2	110
+3	011
−3	101
−4	100

Number System

Range of Numbers

- An N–bit number
  - Unsigned: $0 \;..\; (2^{N}-1)$
  - Signed: $-2^{N-1} \;..\; (2^{N-1}-1)$
- Example: 4–bit



Binary Arithmetic

Sum	Carry	Difference	Borrow
$0+0=0$	0	$0-0=0$	0
$0+1=1$	0	$0-1=1$	1
$1+0=1$	0	$1-0=1$	0
$1+1=0$	1	$1-1=0$	0

Base 2

Carries:	*10011 11*	
	1001.011	$= (9.375)_{10}$
	1101.101	$=(13.625)_{10}$
	10111.000	$= (23)_{10}$ = Sum

Borrow:	*1 1*
	01011
Minuend	101000
Subtrahend	–011001
Difference	001111

Number System

Binary Arithmetic: Multiplication

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

```
  111.10
    10.1
--------
   11110
  00000x
 11110xx
--------
10010110
```

$(7.5)_{10} = (111.10)_2$ Q3.2
$(2.5)_{10} = (10.1)_2$ Q2.1
$(18.75)_{10} = (10010.110)_2$ Q5.3

Binary Computation

```
010001 (17=16+1)
001011 (11=8+2+1)
----------------
011100 (28=16+8+4)
```

Unsigned arithmetic
```
010001 (17=16+1)
101011 (43=32+8+2+1)
----------------
111100 (60=32+16+8+4)
```

Signed arithmetic (w/ 2's complement)
```
010001 (17=16+1)
101011 (-21: 2's complement=010101=21)
----------------
111100 (2's complement=000100=4, i.e. -4)
```

Number System

Binary Computation

The carry is discarded

Unsigned arithmetic

101111 (47)
011111 (31)

001110 (78?? Due to overflow, note that
78 cannot be represented
by a 6-bit unsigned number)

The carry is discarded

Signed arithmetic (w/ 2's complement)

101111 (-17 since 2's complement=010001)
011111 (31)

001110 (14)

Logic Gates

Logic Gates

Boolean Variables

- A multi-dimensional space spanned by a set of n Boolean variables is denoted by $\mathcal{B}^n$
- A literal is an instance (e.g. A) of a variable or its complement ($\bar{A}$)

What is Boolean Algebra

- An algebra dealing with
  - Binary variables by alphabetical letters
  - Logic operations: OR, AND, XOR, etc
- Consider the following Boolean equation

$$F(X,Y,Z) = \overline{X \cdot Y} + \overline{Y \cdot \bar{Z} + Z}$$

- A Boolean function can be represented by a truth table which list all combinations of 1's and 0's for each binary value

Boolean Algebra

- Algebra is a complete set of rules defined on some variables.
- Variables can be Real or Logical: This subject deals with Logical Variables
- A Logical Variable can take one of two values
- A Logical Function is represented by
  - Truth Tables
  - Boolean Expressions

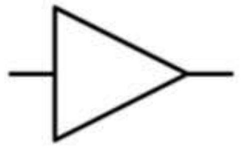
Logic Gates

Combinational Logic Circuits

- Logic Gates: Control the flow of information
- Represent Logical Operations (Functions)
  - Inputs are like arguments to a function
  - Outputs are like result of the function
  - Fundamental Set
    - AND
    - OR
    - NOT
    - Transmission Gate
- Truth Tables…

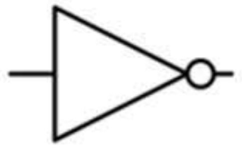
Logic Gates

Buffer



Input	Output
0	0
1	1

Inverter



Input	Output
0	1
1	0

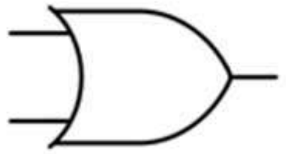
AND

A	B	Output
0	0	0
1	0	0
0	1	0
1	1	1

NAND

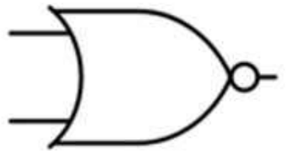
A	B	Output
0	0	1
1	0	1
0	1	1
1	1	0

OR



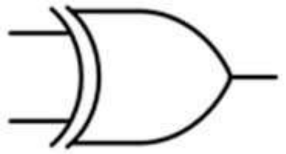
A	B	Output
0	0	0
1	0	1
0	1	1
1	1	1

NOR



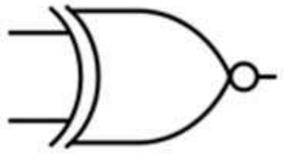
A	B	Output
0	0	1
1	0	0
0	1	0
1	1	0

XOR



A	B	Output
0	0	0
1	0	1
0	1	1
1	1	0

XNOR



A	B	Output
0	0	1
1	0	0
0	1	0
1	1	1

Logic Gates

Postulates of Boolean Algebra

S.No.	Name of the Postulates	Postulate Equation
1	Law of Identity	$A + 0 = 0 + A = A$ $A . 1 = 1 . A = A$
2	Commutative Law	$(A + B) = (B + A)$ $(A . B) = (B . A)$
3	Distributive Law	$A . (B + C) = (A . B) + (A . C)$ $A + (B . C) = (A + B) . (A + C)$
4	Associative Law	$A + (B + C) = (A + B) + C$ $(A . B) . C = A . (B . C)$
5	Complement Law	$A + A' = 1$ $A . A' = 0$

Logic Gates

Duality Principle

$X+0=X \qquad X\cdot 1=X$

$X+1=1 \qquad X\cdot 0=0$

$X+X=X \qquad X\cdot X=X$

$X+\overline{X}=1 \qquad X\cdot\overline{X}=0$

$X+Y=Y+X \qquad X\cdot Y=Y\cdot X$

$X(Y+Z)=XY+XZ \qquad X+Y\cdot Z=(X+Y)\cdot(X+Z)$

$\overline{X+Y}=\overline{X}\cdot\overline{Y} \qquad \overline{X\cdot Y}=\overline{X}+\overline{Y}$

$X+X\cdot Y=X \qquad X\cdot(X+Y)=X$

$X+\overline{X}\cdot Y=X+Y \qquad X\cdot(\overline{X}+Y)=X\cdot Y$

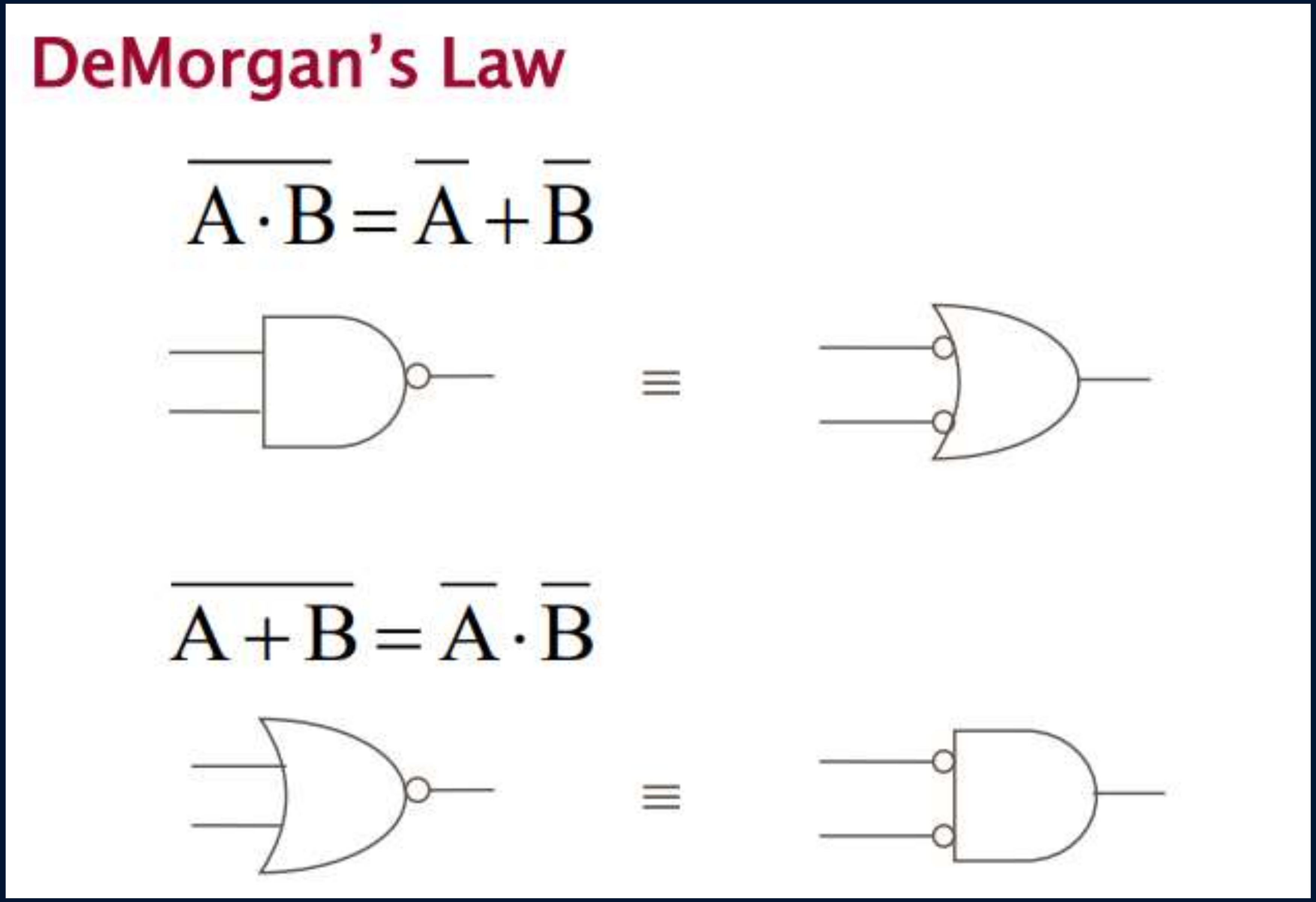
$XY+\overline{X}Z+YZ=XY+\overline{X}Z \qquad (X+Y)(\overline{X}+Z)(Y+Z)=(X+Y)(\overline{X}+Z)$

Theorem	Statement	Equations
Duality Theorem	A boolean relation can be derived from another boolean relation by changing OR sign to AND sign and vice versa and complementing the 0s and 1s.	A + A' = 1 and A . A' = 0 are the dual relations.

Duality Principle

- A Boolean equation remains valid if we take the dual of the expressions on both sides of the equals sign
- Dual of expressions
  - Interchange 1's and 0's
  - Interchange AND (•) and OR (+)

Logic Gates



DeMorgan's Theorem 1	Complement of a product is equal to the sum of its complement.	(A . B)' = A' + B'
DeMorgan's Theorem 2	Complement of a sum is equal to the product of the complement.	(A + B)' = A' . B'

Logic Gates

Idempotency Theorem	–	A + A = A A . A = A
Involution Theorem	–	A” = A
Absorption Theorem	–	A + (A . B) = A A . (A + B) = A
Associative Theorem	–	A + (B + C) = (A + B) + C A . (B . C) = (A . B) . C
Consensus Theorem	–	AB + A’C + BC = AB + A’C (A + B) + (A’ + C) + (B + C) = (A + B) + (A’ + C)
Uniting Theorem	–	AB + AB’ = A (A+B) + (A + B’) = A

Logic Gates

Derivation of Simplification

$$X + \overline{X}Y$$
$$= X \cdot (1 + Y) + \overline{X}Y$$
$$= X + XY + \overline{X}Y$$
$$= X + (X + \overline{X})Y$$
$$= X + Y$$
$$\therefore X + \overline{X}Y = X + Y$$

Derivation of Consensus Theorem

$$XY + \overline{X}Z + YZ$$
$$= XY + \overline{X}Z + YZ \cdot (X + \overline{X})$$
$$= XY + \overline{X}Z + XYZ + \overline{X}YZ$$
$$= XY(1 + Z) + \overline{X}Z(1 + Y)$$
$$= XY + \overline{X}Z$$
$$\therefore XY + \overline{X}Z + YZ = XY + \overline{X}Z$$

Logic Gates

Sum of Product (SOP) Form

- A product of literals is called a product term (e.g. $\bar{A}\cdot B\cdot C$ in $\mathcal{B}^3$, or $B\cdot C$ in $\mathcal{B}^3$)
- Sum–Of–Product (SOP) Form: OR of product terms is called SOP. e.g. $\bar{A}B+AC$
- A minterm is a product term in which every literal (or variable) appears in $\mathcal{B}^n$
  - $\bar{A}BC$ is a minterm in $\mathcal{B}^3$ but not in $\mathcal{B}^4$. ABCD is a minterm in $\mathcal{B}^4$.
- A canonical (or standard) SOP function:
  - A sum of minterms, corresponding to the input combination of the truth table, for which the function produces a "1" output.

Minterms in $\mathcal{B}^3$

			m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
A	B	C	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}B\bar{C}$	$\bar{A}BC$	$A\bar{B}\bar{C}$	$A\bar{B}C$	$AB\bar{C}$	ABC
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Logic Gates

Canonical (Standard) SOP Function

$$F(A,B,C) = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + A\overline{B}C$$
$$= m0 + m1 + m4 + m5$$

$$F(A,B,C) = \sum m(0,1,4,5) = one-set(0,1,4,5)$$

$$F(A,B,C,D) = \overline{A}B\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D + ABC\overline{D}$$
$$= m4 + m9 + m14$$

$$F(A,B,C,D) = \sum m(4,9,14) = one-set(4,9,14)$$

Product of Sums Design

Maxterms:

A maxterm is a NOT minterm

maxterm M0 = NOT minterm m0

```
If m0 = (!X.!Y)
M0 = !m0
   = !(!X . !Y)
   = !!X + !!Y
   = X + Y
```

Logic Gates

Product of Sum (POS) form (dual of SOP form)

- A sum of literals is called a sum term (e.g. $\bar{A}+B+C$ in $\mathcal{B}^3$, or $(B+C)$ in $\mathcal{B}^3$)
- Product-Of-Sum (POS) Form: AND of sum terms is called POS. e.g. $(\bar{A}+B)(A+C)$
- A maxterm is a sum term in which every literal (or variable) appears in $\mathcal{B}^n$
  - $(\bar{A}+B+C)$ is a maxterm in $\mathcal{B}^3$ but not in $\mathcal{B}^4$. $A+B+C+D$ is a maxterm in $\mathcal{B}^4$.
- A canonical (or standard) POS function:
  - A product (AND) of maxterms, corresponding to the input combination of the truth table, for which the function produces a "0" output.

Product of Sums Design

X	Y	minterms	maxterms
0	0	m0 = !X . !Y	M0 = !m0 = X + Y
0	1	m1 = !X . Y	M1 = !m1 = X + !Y
1	0	m2 = X . !Y	M2 = !m2 = !X + Y
1	1	m3 = X . Y	M3 = !m3 = !X + !Y

Logic Gates

Canonical (Standard) POS Function

$$F(A,B,C) = (\overline{A}+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)(A+\overline{B}+\overline{C})(A+\overline{B}+C)$$
$$= M7 \cdot M6 \cdot M3 \cdot M2$$

$$F(A,B,C) = \prod M(2,3,6,7) = \text{zero } \mathit{for} \text{ set}(2,3,6,7)$$

$$F(A,B,C,D) = (\overline{A}+B+\overline{C}+\overline{D})(A+\overline{B}+\overline{C}+D)(A+B+C+\overline{D})$$
$$= M11 \cdot M6 \cdot M1$$

$$F(A,B,C,D) = \prod M(1,6,11) = \text{zero } \mathit{for} \text{ set}(1,6,11)$$

Maxterms in $\mathcal{B}^3$

A	B	C	M_0 $A+B+C$	M_1 $A+B+\overline{C}$	M_2 $A+\overline{B}+C$	M_3 $A+\overline{B}+\overline{C}$	M_4 $\overline{A}+B+C$	M_5 $\overline{A}+B+\overline{C}$	M_6 $\overline{A}+\overline{B}+C$	M_7 $\overline{A}+\overline{B}+\overline{C}$
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Logic Gates

Convert a Boolean to Canonical SOP

- Expand the Boolean equation into a SOP
- Take each product term with a missing literal, say A, and "AND" (•) it with (A+Ā)

Convert a Boolean to Canonical SOP

$$F = \overline{A}\,\overline{B} + BC \text{ in } \mathcal{B}^3$$

$$\Rightarrow F(A,B,C) = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}BC + ABC$$

$$= \sum m(0,1,3,7)$$

	A	B	C	F	
$\overline{A}\,\overline{B}\,\overline{C}$	0	0	0	1	0
$\overline{A}\,\overline{B}C$	0	0	1	1	1
$\overline{A}B\overline{C}$	0	1	0	0	
$\overline{A}BC$	0	1	1	1	3
$A\overline{B}\,\overline{C}$	1	0	0	0	
$A\overline{B}C$	1	0	1	0	
$AB\overline{C}$	1	1	0	0	
ABC	1	1	1	1	7

Minterms listed as 1's

Logic Gates

Convert a Boolean to Canonical SOP

$$F = \overline{A}\,\overline{B} + BC \text{ in } \mathcal{B}^4$$

$$\begin{aligned}
\Rightarrow F(A,B,C,D) &= \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD \\
&\quad + \overline{A}BC\overline{D} + \overline{A}BCD + ABC\overline{D} + ABCD \\
&= \sum m(0,1,2,3,6,7,14,15)
\end{aligned}$$

$$F = AB + \overline{B}(\overline{A} + \overline{C}) \text{ in } \mathcal{B}^3$$

$$\begin{aligned}
\Rightarrow F(A,B,C) &= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + AB\overline{C} + ABC \\
&= \sum m(0,1,4,6,7)
\end{aligned}$$

Interchange Canonical SOP and POS

- For the same Boolean equation
  - Canonical SOP form is complementary to its canonical POS form
  - Use missing terms to interchange Σ and Π
- Examples
  - $F(A,B,C) = \Sigma\, m(0,1,4,6,7)$

  Can be re-expressed by

  - $F(A,B,C) = \Pi\, M(2,3,5)$

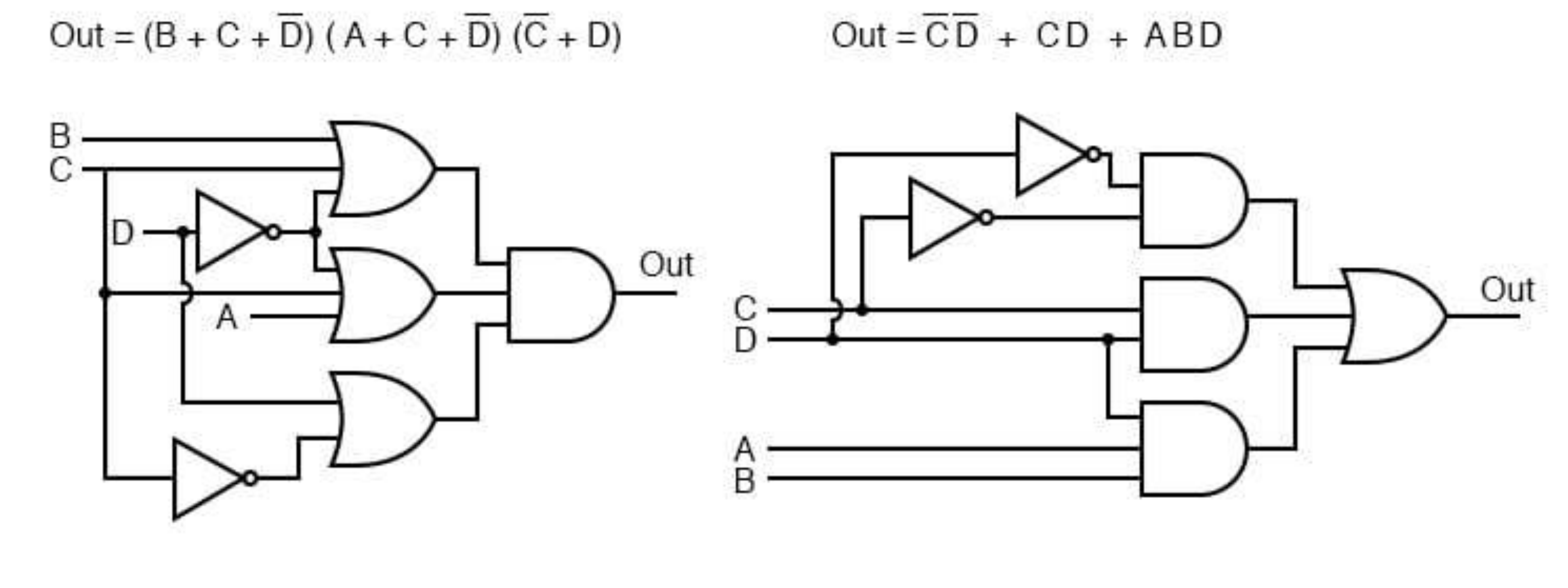
    Where 2, 3, 5 are the missing minterms in the canonical SOP form

Logic Gates

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \bar{x}_1\bar{x}_2x_3$	$M_1 = x_1 + x_2 + \bar{x}_3$
2	0	1	0	$m_2 = \bar{x}_1x_2\bar{x}_3$	$M_2 = x_1 + \bar{x}_2 + x_3$
3	0	1	1	$m_3 = \bar{x}_1x_2x_3$	$M_3 = x_1 + \bar{x}_2 + \bar{x}_3$
4	1	0	0	$m_4 = x_1\bar{x}_2\bar{x}_3$	$M_4 = \bar{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\bar{x}_2x_3$	$M_5 = \bar{x}_1 + x_2 + \bar{x}_3$
6	1	1	0	$m_6 = x_1x_2\bar{x}_3$	$M_6 = \bar{x}_1 + \bar{x}_2 + x_3$
7	1	1	1	$m_7 = x_1x_2x_3$	$M_7 = \bar{x}_1 + \bar{x}_2 + \bar{x}_3$

Logic Gates

Why Do We Need Boolean Algebra and SOP/POS Representations?



Logic Gates

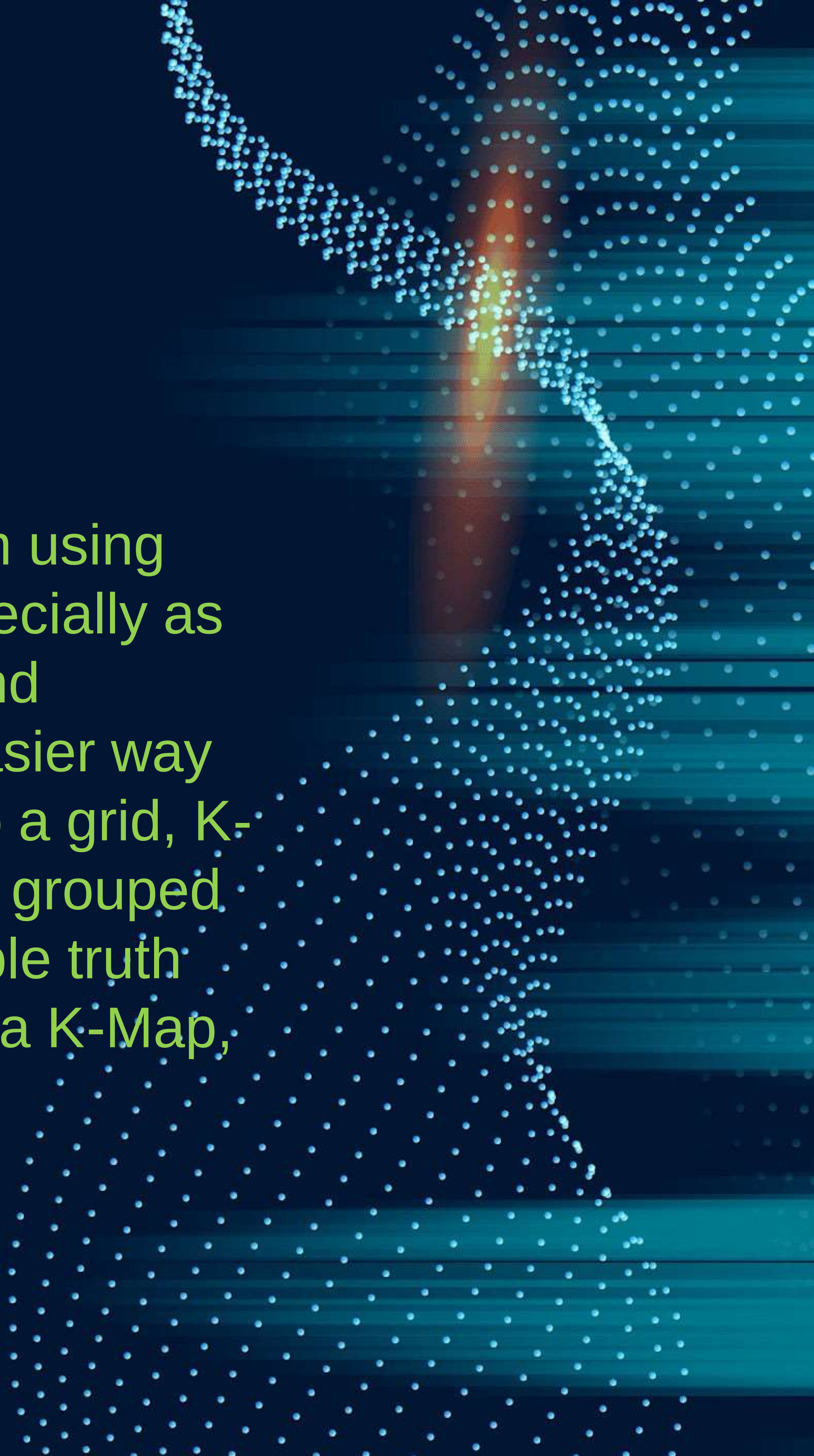
Why Do We Need Boolean Algebra and SOP/POS Representations?

Boolean algebra is essential because it helps simplify logical expressions, which directly impacts the design of efficient digital circuits. Without it, creating optimal circuits would be time-consuming and error-prone. SOP (Sum of Products) and POS (Product of Sums) representations are useful because they provide standardized formats that are easy to implement using logic gates. For example, SOP directly translates into OR gates fed by AND gates, while POS does the opposite.

Logic Gates

Introduction to Karnaugh Maps (K-Maps)

When dealing with larger Boolean expressions, simplifying them using Boolean algebra becomes very tricky and time-consuming, especially as the number of variables increases. This can lead to mistakes and inefficient circuits. Karnaugh Maps (K-Maps) provide a much easier way to simplify expressions visually. By arranging the truth table into a grid, K-Maps help identify patterns like adjacent 1s or 0s, which can be grouped to create simplified SOP or POS forms. For example, a 4-variable truth table might seem complicated to simplify algebraically, but with a K-Map, we can find the minimal solution much faster and without much guesswork.



Logic Gates

Steps to Solve Expression using K-map

1. Select the K-map according to the number of variables.

2. Identify minterms or maxterms as given in the problem.

3. For SOP put 1's in blocks of K-map respective to the minterms (0's elsewhere).

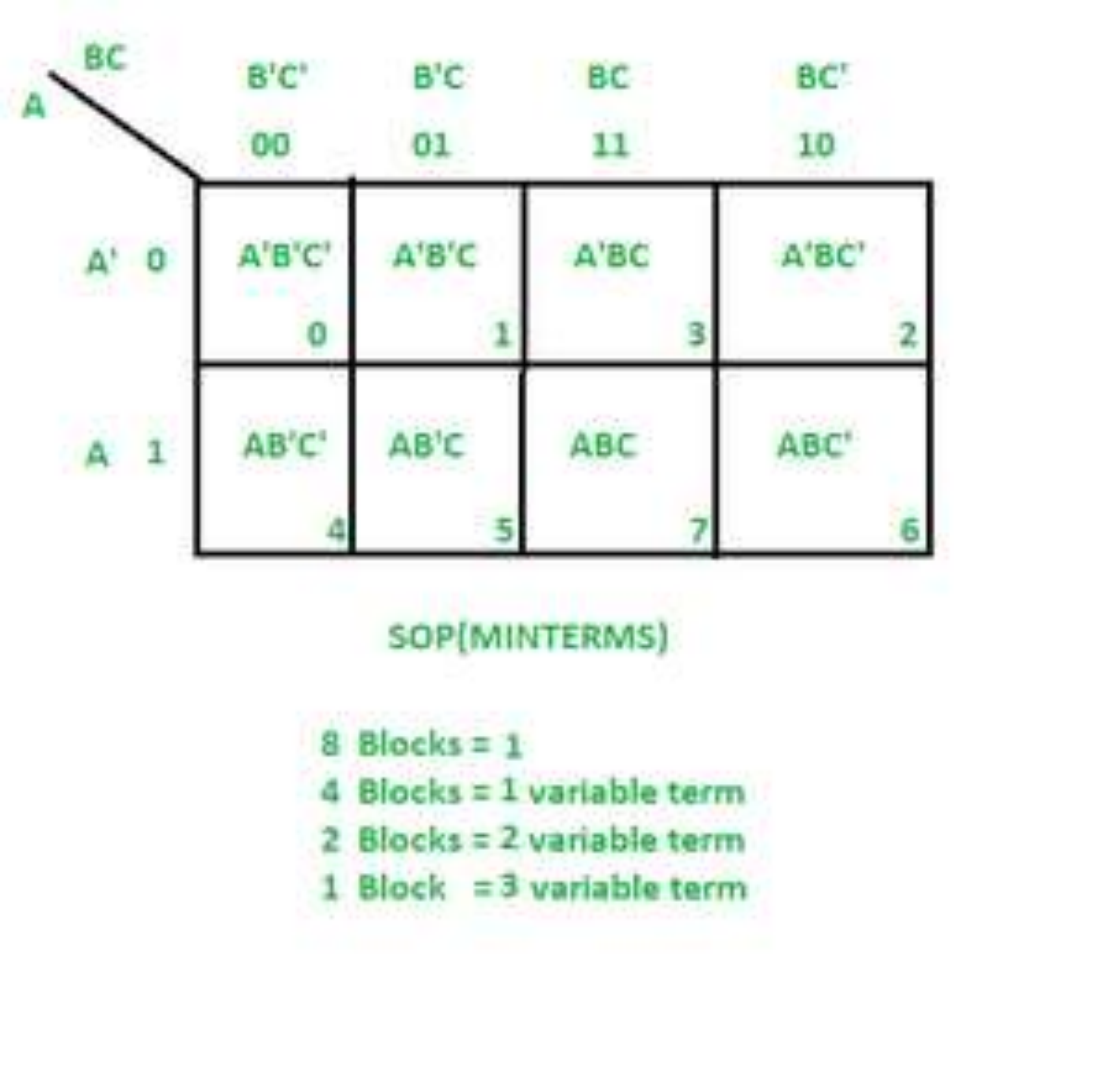
4. For POS put 0's in blocks of K-map respective to the max terms (1's elsewhere).

5. Make rectangular groups containing total terms in power of two like 2,4,8 ..(except 1) and try to cover as many elements as you can in one group.

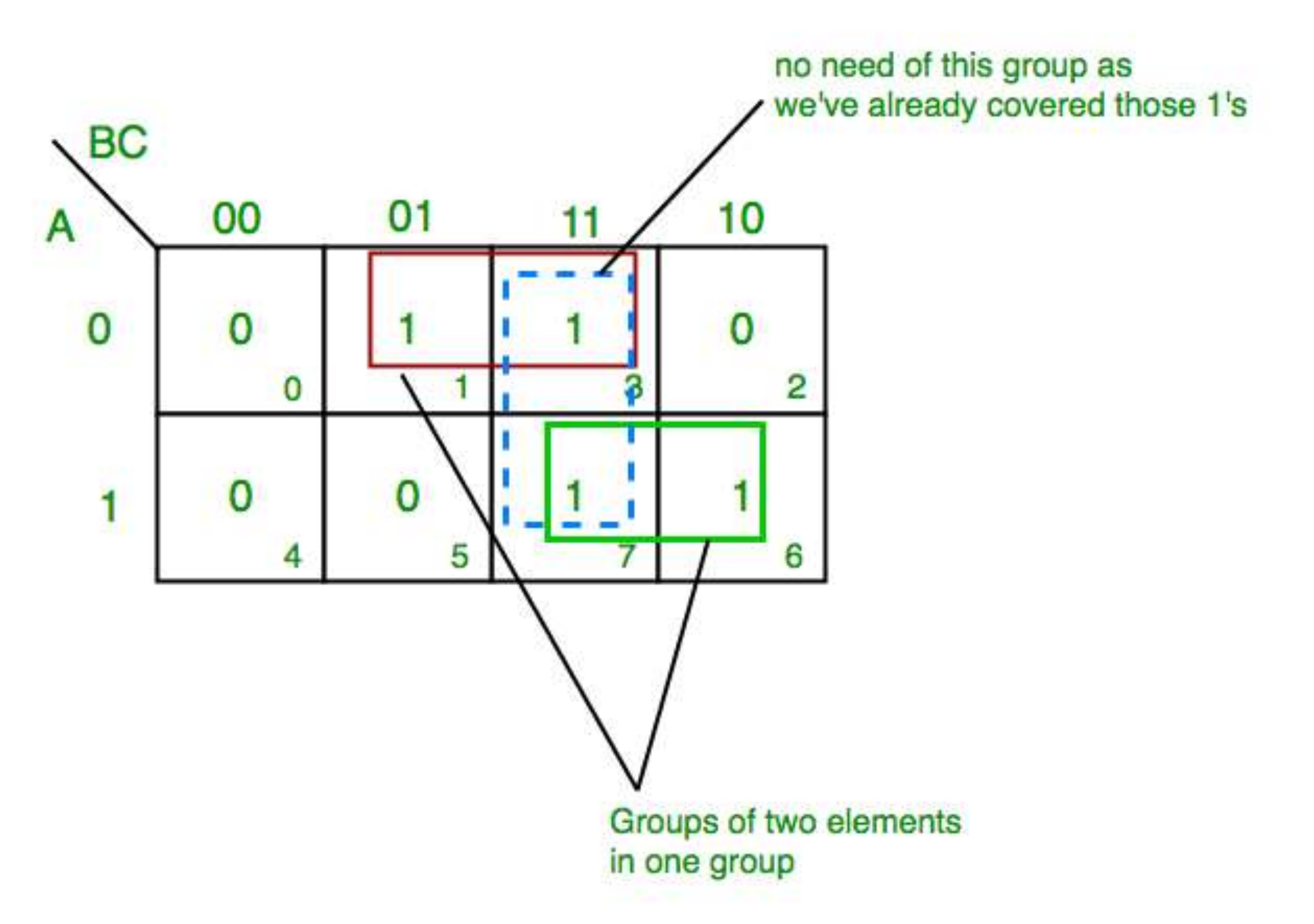
6. From the groups made in step 5 find the product terms and sum them up for SOP form.

Logic Gates

K-map of 3 variables - SOP



(A′C+AB)



Logic Gates

K-map of 4 variables - SOP

AB \ CD	C'D' 00	C'D 01	CD 11	CD' 10
A'B' 00	A'B'C'D' 0	A'B'C'D 1	A'B'CD 3	A'B'CD' 2
A'B 01	A'BC'D' 4	A'BC'D 5	A'BCD 7	A'BCD' 6
AB 11	ABC'D' 12	ABC'D 13	ABCD 15	ABCD' 14
AB' 10	AB'C'D' 8	AB'C'D 9	AB'CD 11	AB'CD' 10

SOP(MINTERMS)

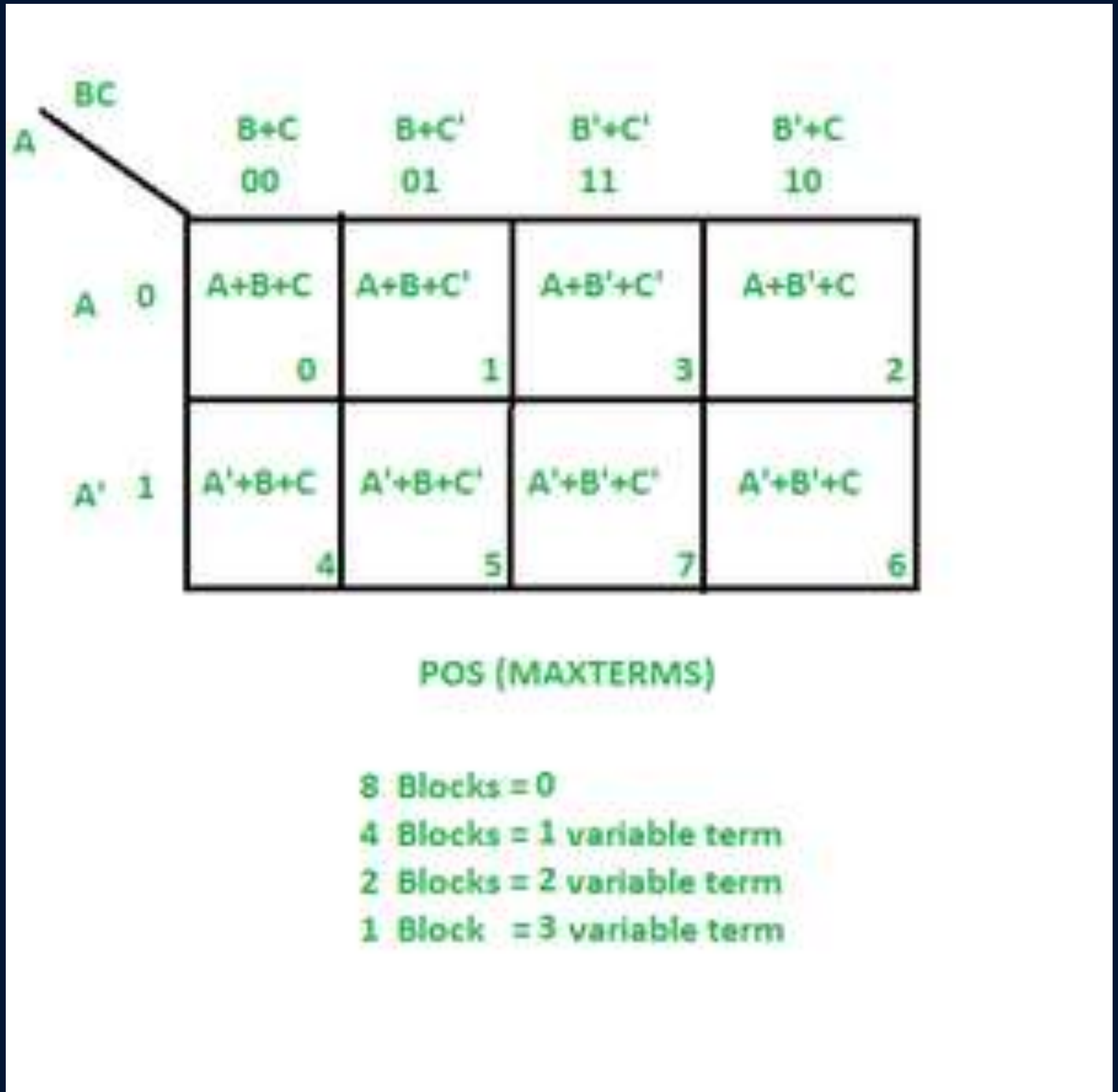
16 Blocks = 1
8 Blocks = 1 variable term
4 Blocks = 2 variable term
2 Blocks = 3 variable term
1 Block = 4 variable term

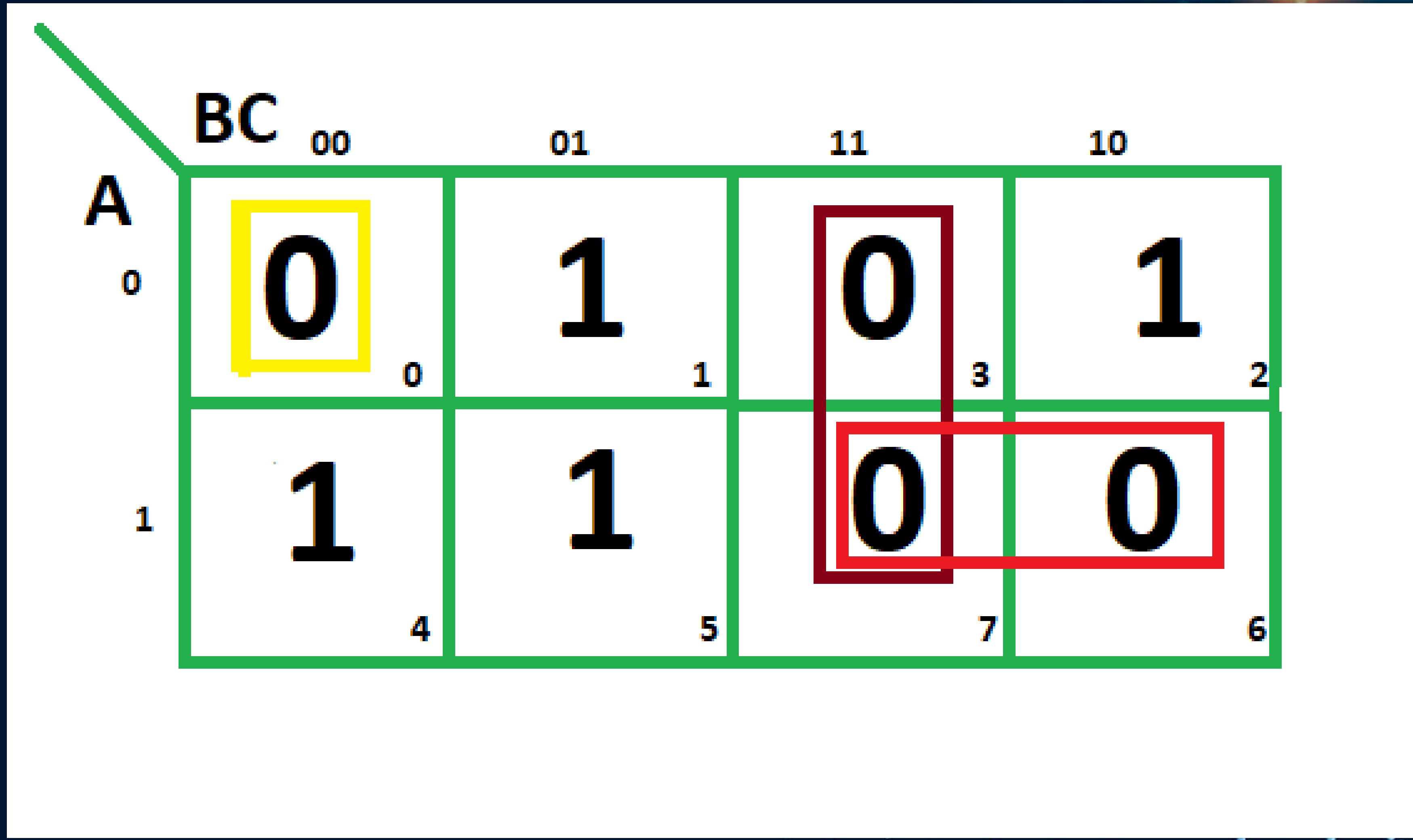
Logic Gates

(POS of F)=(SOP of F’)’

Logic Gates

K-map of 3 variables - POS

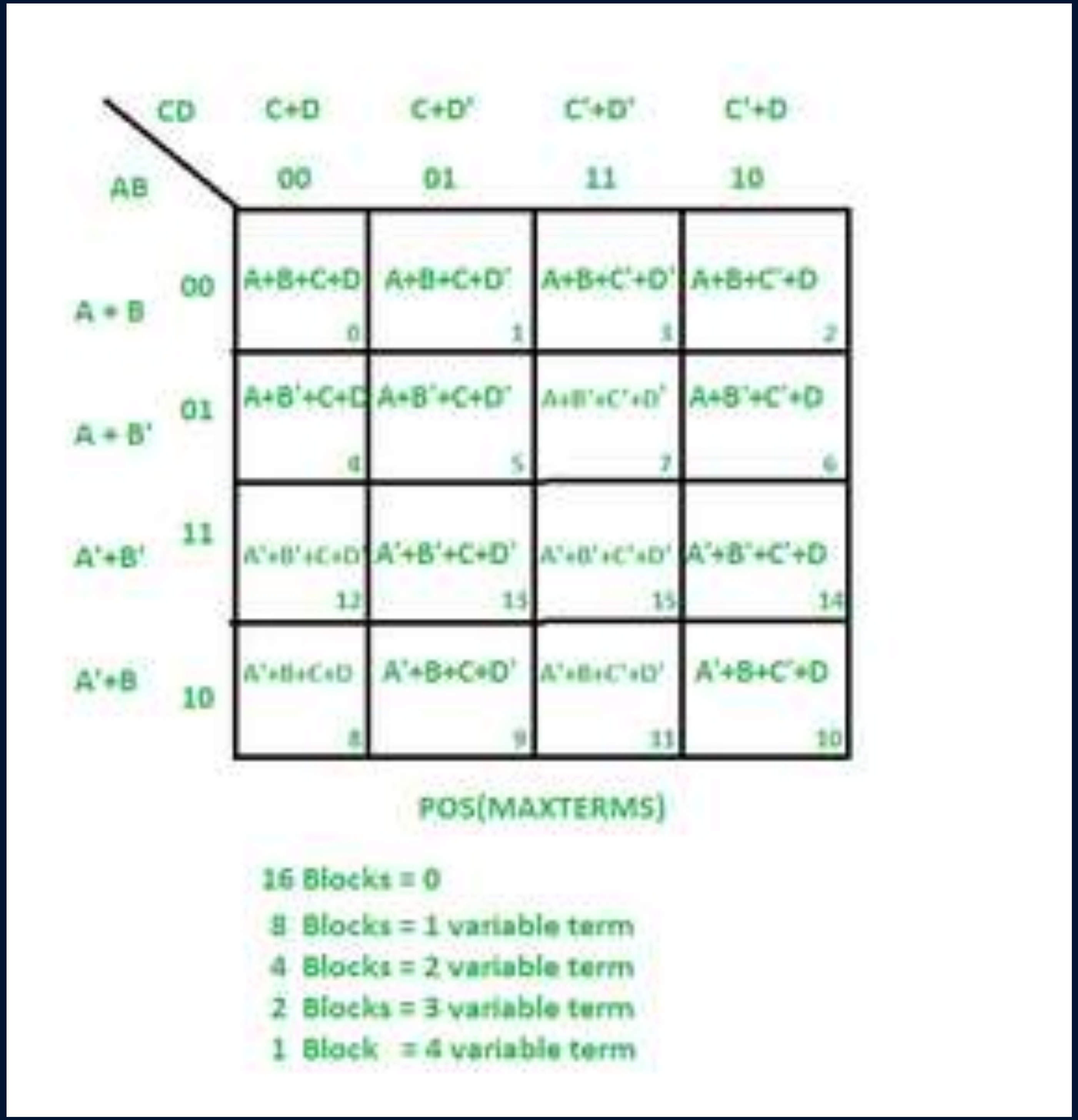


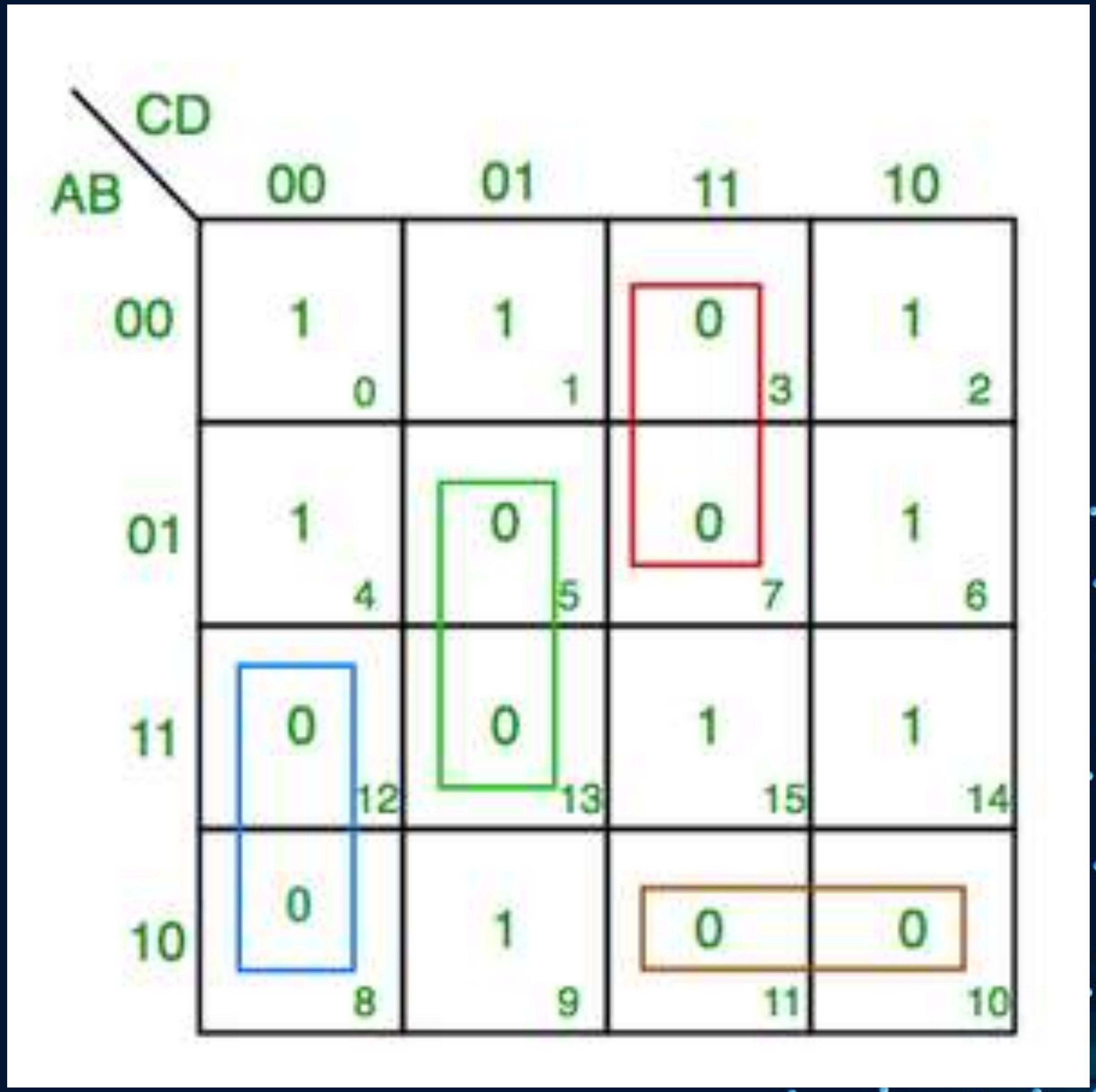


(A' + B’) (B’ + C’) (A + B + C)

Logic Gates

K-map of 4 variables - POS



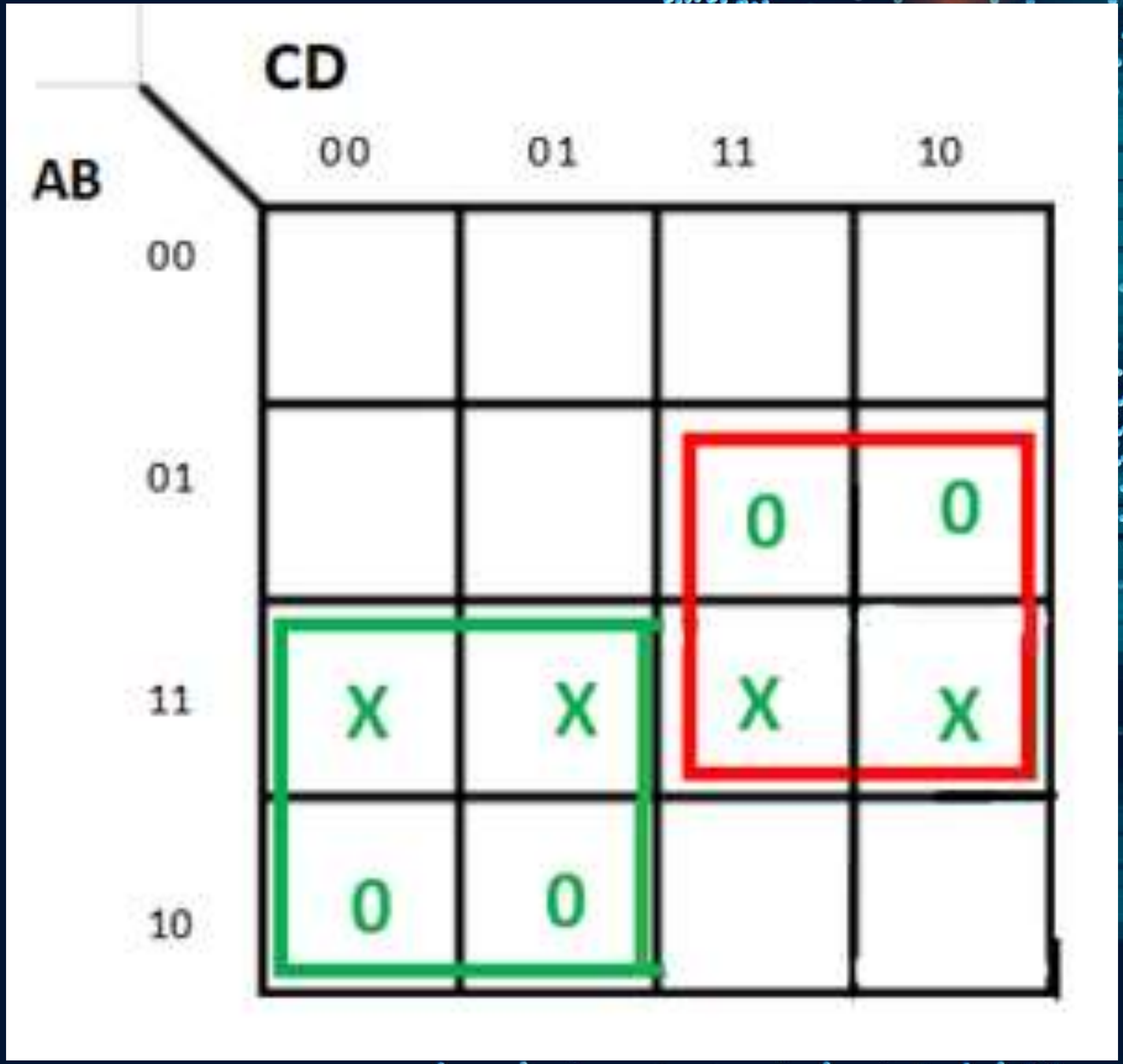


(C+D’+B’).(C’+D’+A).(A’+C+D).(A’+B+C’)

Logic Gates

Don't Cares in Digital Design

Don't cares are input conditions where the output doesn’t impact the circuit's functionality. Represented as 'X' in truth tables or K-Maps, they allow flexibility in simplification. Designers can group don't cares with 1s or 0s to create larger groups, resulting in simpler and more efficient circuits.



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

Thanks!

Instagram

Facebook

Linkedin