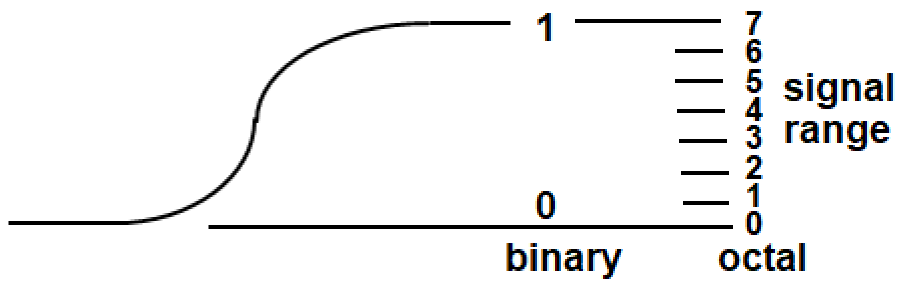
DIGITAL ELECTRONICS

LOGIC GATES

Digital Computers

- **Imply that the computer deals with digital information, i.e., it deals with the information that is represented by binary digits**
- **Why *BINARY* ? instead of Decimal or other number system ?**

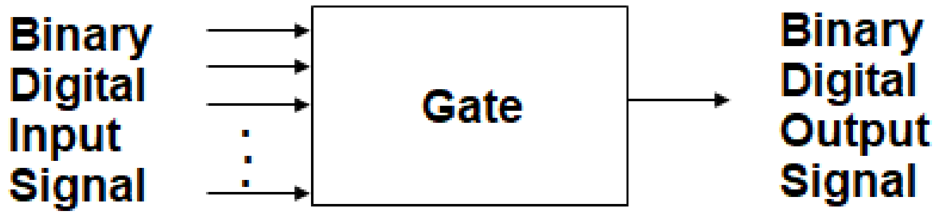
*** Consider electronic signal**



*** Consider the calculation cost - Add**

	0	1
0	0	1
1	1	10

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18



Types of Basic Logic Blocks

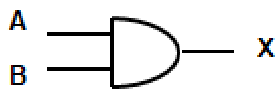
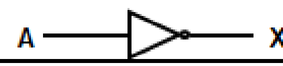
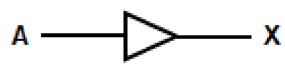
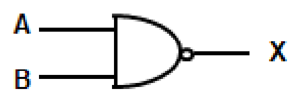
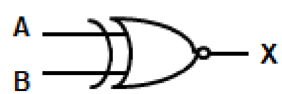
- Combinational Logic Block
  Logic Blocks whose output logic value depends only on the input logic values

- Sequential Logic Block
  Logic Blocks whose output logic value depends on the input values and the state (stored information) of the blocks

Functions of Gates can be described by

- Truth Table
- Boolean Function
- Karnaugh Map

COMBINATIONAL GATES

Name	Symbol	Function	Truth Table
AND		$X = A \cdot B$ or $X = AB$	A B | X 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$X = A + B$	A B | X 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
I		$X = A'$	A | X 0 | 1 1 | 0
Buffer		$X = A$	A | X 0 | 0 1 | 1
NAND		$X = (AB)'$	A B | X 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$X = (A + B)'$	A B | X 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR Exclusive OR		$X = A \oplus B$ or $X = A'B + AB'$	A B | X 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
XNOR Exclusive NOR or Equivalence		$X = (A \oplus B)'$ or $X = A'B' + AB$	A B | X 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

BOOLEAN ALGEBRA

Boolean Algebra

*** Algebra with Binary(Boolean) Variable and Logic Operations**
*** Boolean Algebra is useful in Analysis and Synthesis of Digital Logic Circuits**

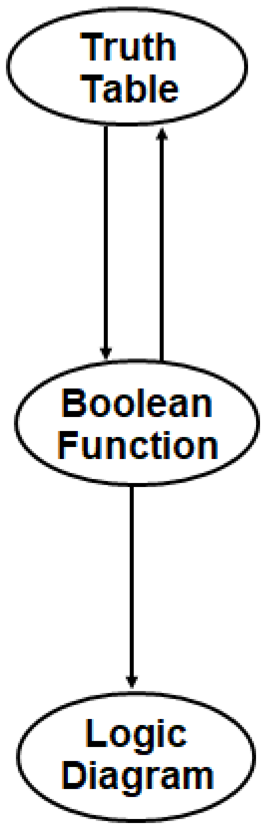
- **Input and Output signals can be represented by Boolean Variables, and**
- **Function of the Digital Logic Circuits can be represented by Logic Operations, i.e., Boolean Function(s)**
- **From a Boolean function, a logic diagram can be constructed using AND, OR, and I**

Truth Table

*** The most elementary specification of the function of a Digital Logic Circuit is the Truth Table**

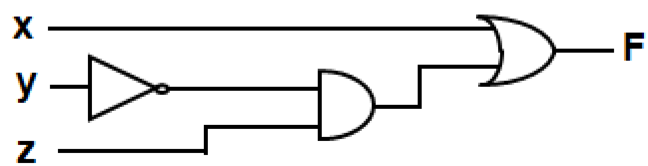
- **Table that describes the Output Values for all the combinations of the Input Values, called *MINTERMS***
- **n input variables → 2^n minterms**

LOGIC CIRCUIT DESIGN



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	1
1	1	1	1

$F = x + y'z$



COMPLEMENT OF FUNCTIONS

A Boolean function of a digital logic circuit is represented by only using logical variables and AND, OR, and Invert operators.

→ Complement of a Boolean function

- Replace all the variables and subexpressions in the parentheses appearing in the function expression with their respective complements

$$A,B,...,Z,a,b,...,z \Rightarrow A',B',...,Z',a',b',...,z'$$
$$(p + q) \Rightarrow (p + q)'$$

- Replace all the operators with their respective complementary operators

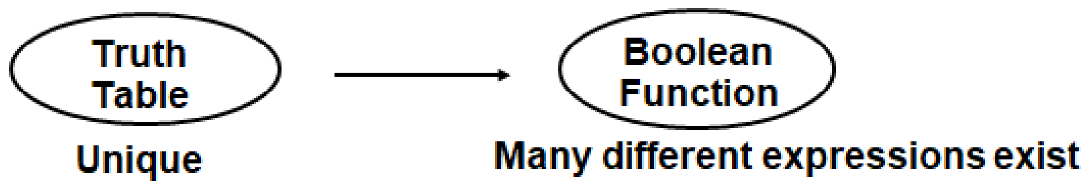
$$\text{AND} \Rightarrow \text{OR}$$
$$\text{OR} \Rightarrow \text{AND}$$

- Basically, extensive applications of the De Morgan's theorem

$$(x_1 + x_2 + ... + x_n)' \Rightarrow x_1'x_2'... x_n'$$

$$(x_1x_2 ... x_n)' \Rightarrow x_1' + x_2' +...+ x_n'$$

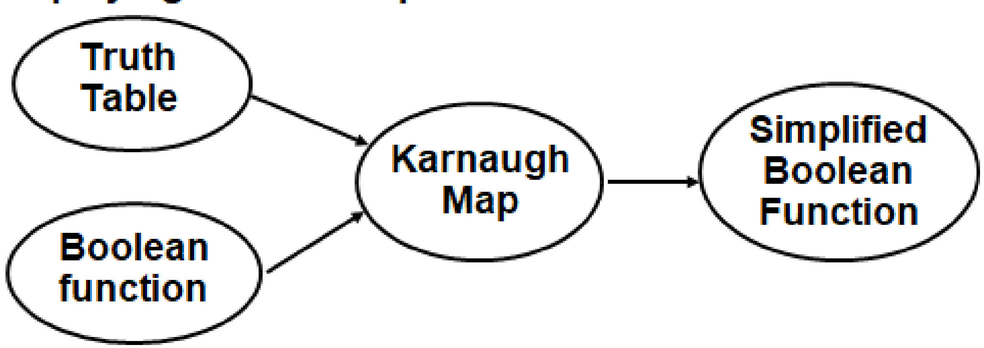
SIMPLIFICATION



Simplification from Boolean function

- Finding an equivalent expression that is least expensive to implement
- For a simple function, it is possible to obtain a simple expression for low cost implementation
- But, with complex functions, it is a very difficult task

Karnaugh Map (K-map) is a simple procedure for simplifying Boolean expressions.

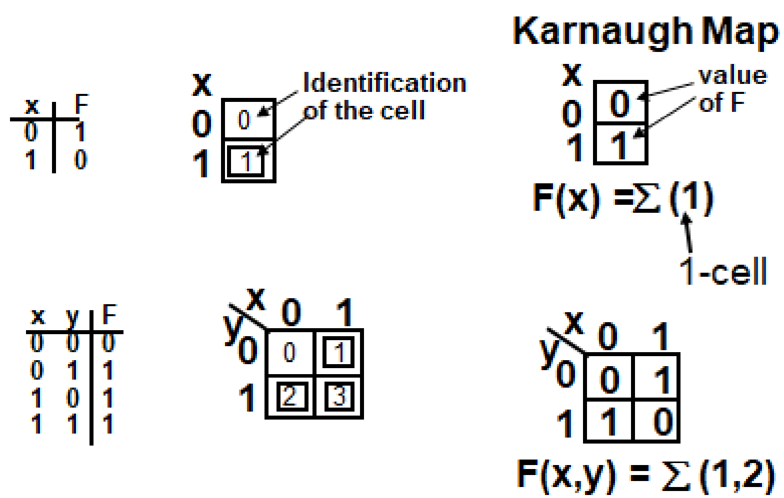


KARNAUGH MAP

Karnaugh Map for an n-input digital logic circuit (n-variable sum-of-products form of Boolean Function, or Truth Table) is

- **Rectangle divided into 2^n cells**
- **Each cell is associated with a *Minterm***
- **An output(function) value for each input value associated with a mintern is written in the cell representing the minterm**
  → 1-cell, 0-cell

Each Minterm is identified by a decimal number whose binary representation is identical to the binary interpretation of the input values of the minterm.



KARNAUGH MAP

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

x \ yz	00	01	11 (y)	10 (y)
0	0	1	3	2
x 1	4	5	7	6
		z	z	

x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	0	0

F(x,y,z) = $\sum$(1,2,4)

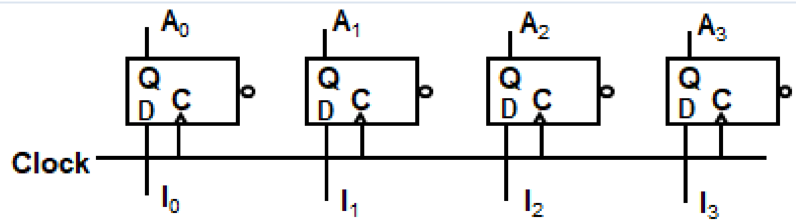
u	v	w	x	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

uv \ wx	00	01	11 (w)	10 (w)	
00	0	1	3	2	
01	4	5	7	6	v
u 11	12	13	15	14	v
u 10	8	9	11	10	
		x	x		

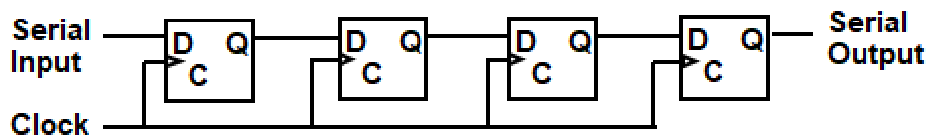
uv \ wx	00	01	11	10
00	0	1	1	0
01	0	0	0	1
11	0	0	0	1
10	1	1	1	0

F(u,v,w,x) = $\sum$(1,3,6,8,9,11,14)

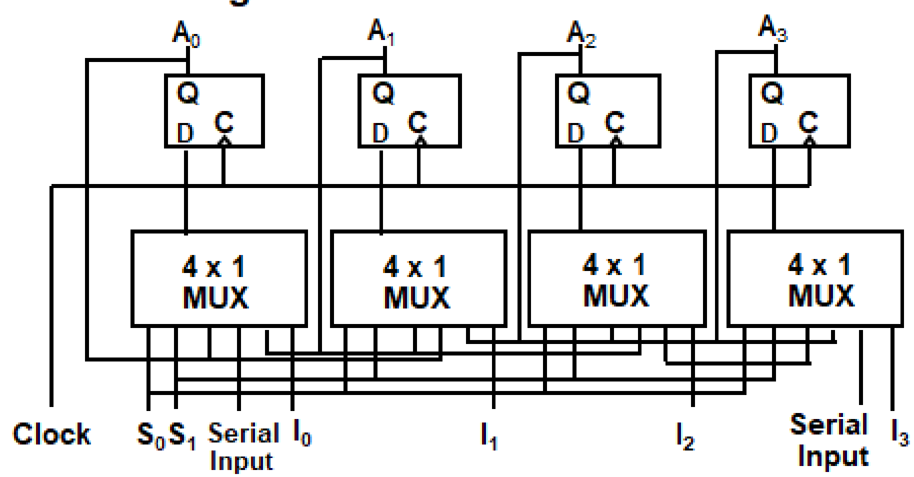
SEQUENTIAL CIRCUITS - Registers



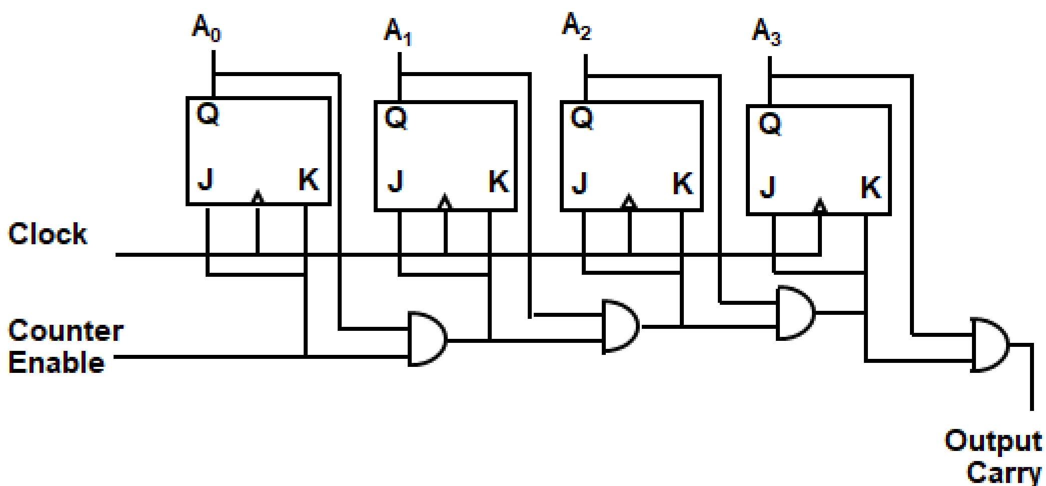
Shift Registers



Bidirectional Shift Register with Parallel Load

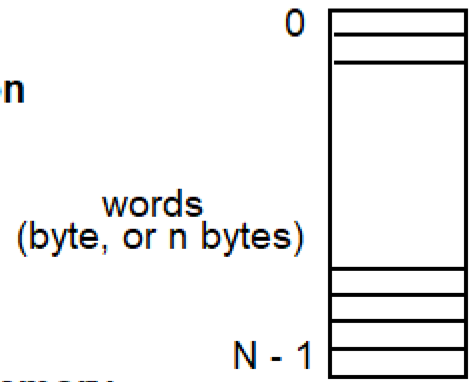


SEQUENTIUAL CIRCUITS - Counters



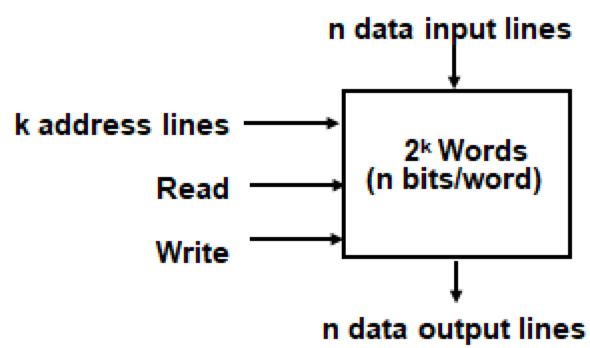
MEMORY COMPONENTS

Logical Organization



Random Access Memory

- Each word has a unique address
- Access to a word requires the same time independent of the location of the word
- Organization



TYPES OF ROM

ROM
- Store information (function) during production
- Mask is used in the production process
- Unalterable
- Low cost for large quantity production --> used in the final products

PROM (Programmable ROM)
- Store info electrically using PROM programmer at the user's site
- Unalterable
- Higher cost than ROM -> used in the system development phase
  -> Can be used in small quantity system

EPROM (Erasable PROM)
- Store info electrically using PROM programmer at the user's site
- Stored info is erasable (alterable) using UV light (electrically in some devices) and rewriteable
- Higher cost than PROM but reusable --> used in the system development phase. Not used in the system production due to eras ability

REFERENCES

- www. Wikipedia.com
- www.slideshare.net