

EE204: Computer Architecture

Lecture 1

Instructor Profile (1)

2

- Dr Aamir Shafi
- Email: aamir.shafi@nu.edu.pk
- Office: C140 (Civil Block)
- Office Hours:
 - ◆ Monday: 11:00 AM to 12:00 PM
 - ◆ Wednesday: 11:00 to 12:00 PM

Course Information

3

- Credit Hours: 3
- Course Type: Core
- Pre-requisites: Digital Logic Design
- Class Venue: CS-4

Section #	Lecture 1	Lecture 2
Section A	Tuesday (8:00 – 9:20)	Thursday (8:00 – 9:20)
Section B	Tuesday (9:30 – 10:50)	Thursday (9:30 – 10:50)

- The main objective of this course is to **provide a profound understanding of the architectural design and internal working of a microprocessor** which will allow computer science students to appreciate concepts like **optimization and hardware level performance issues**.
- This course also introduces advanced concepts like **pipelining and superscalar architecture** and techniques like **microprogramming**.
- **Multi-Core processors** and issues related to multicore processors are introduced.

- **Course Textbook:**

- ◆ M. Morris Mano, *Computer System Architecture* 3rd Edition 1993, Prentice Hall
- ◆ David A. Patterson, John L. Hennessy, *Computer Organization and Design: The hardware/software interface*, 5th Edition

- **Additional references:**

- ◆ Modern Processor Design: Fundamentals of Superscalar Processors by John Paul Shen and Mikko H. Lipasti

Lecture-wise Topics

6

Lect #	Topics to be covered
1	Introduction to basic hardware components and devices a) Digital Logic Design Review i) Boolean Algebra , ii) Combinational Circuits, iii) Sequential Circuits
2	Frequently used components in computers a) Adders, b) Decoders, c) Multiplexers, d) Registers (Parallel load with shifts), e) Counters Number Systems and Binary Representations a) Integer Representations, b) Floating point representation
3	Logic Operations Shift Operations Combined Arithmetic, Logic and Shift Unit.
4	Additional Arithmetic Operations: a) Integer Multiplication, Division b) Floating Point Operations (Addition)
5	Complete ALU and FPU design Introduction to Machine Language Central processing unit in detail a) General register organization, b) Addressing modes, c) CISC Vs RISC
6	Computer Instructions and instruction codes Introduction to the MIPS Assembly Language Instruction Types 1) Arithmetic Instructions, 2) Memory Reference, 3) Branch Instructions

Lect #	Topics to be covered
7	Design of a Single Cycle Machine 1) Register File, 2) ALU 3) Instruction and Data Memory 4) Control Unit 5) Integration of Units
8	Single Cycle Machine Design Continued, Control Unit (As a hardwired combinational circuit), Exceptions and Interrupts
9	Revision for Mid 1
10	Multi-cycle implementation Motivation and Hardware Differences
11	Enhancing Performance with Pipelining a) Introduction to pipelining b) A pipelined data-path c) Pipelined Control d) Shallow and deep pipelining
12	Multi-cycle Continued, Control Unit State for each instruction type, Pipelined Machine: Hardware and Control Unit
13	Multi-cycle Continued, Control Unit State Machine combined, Exception Handling in Multi-Cycle machine
14	Data Hazards a) RAW (covered in detail) b) WAR c) WAW
15	Control Hazards a) Branch Prediction, Performance of pipelined systems
16	Understanding Performance a) CPU Performance b) Benchmarks c) Evaluating Performance
17	Superscalar Processors. Basic concepts of superscalar processors and instruction independence
18	Step by step execution of instructions in superscalar processors.
19	Code Rescheduling Loop Unrolling

Lect #	Topics to be covered
21	Cache Design: Memory Hierarchy basic concepts Basics and Direct mapped caches
22	Cache Design: Associative Caches and Miss Rates
23	Cache Design: Processor performance evaluation with cache / Multi-level Caches.
24	Virtual Memory
25	Multicores, Multiprocessors, and Clusters Multi-processor Systems basics. Multi-processor system types Parallel Processing Programs
26	Intro to memory sharing models and cache coherence problem Clusters and Other Message-Passing Multiprocessors
27	Hardware multithreading SISD,MIMD,SIMD,SPMD
28	GPUs
29	NVIDIA GPU architecture
30	Revision for Final

Grading Criteria

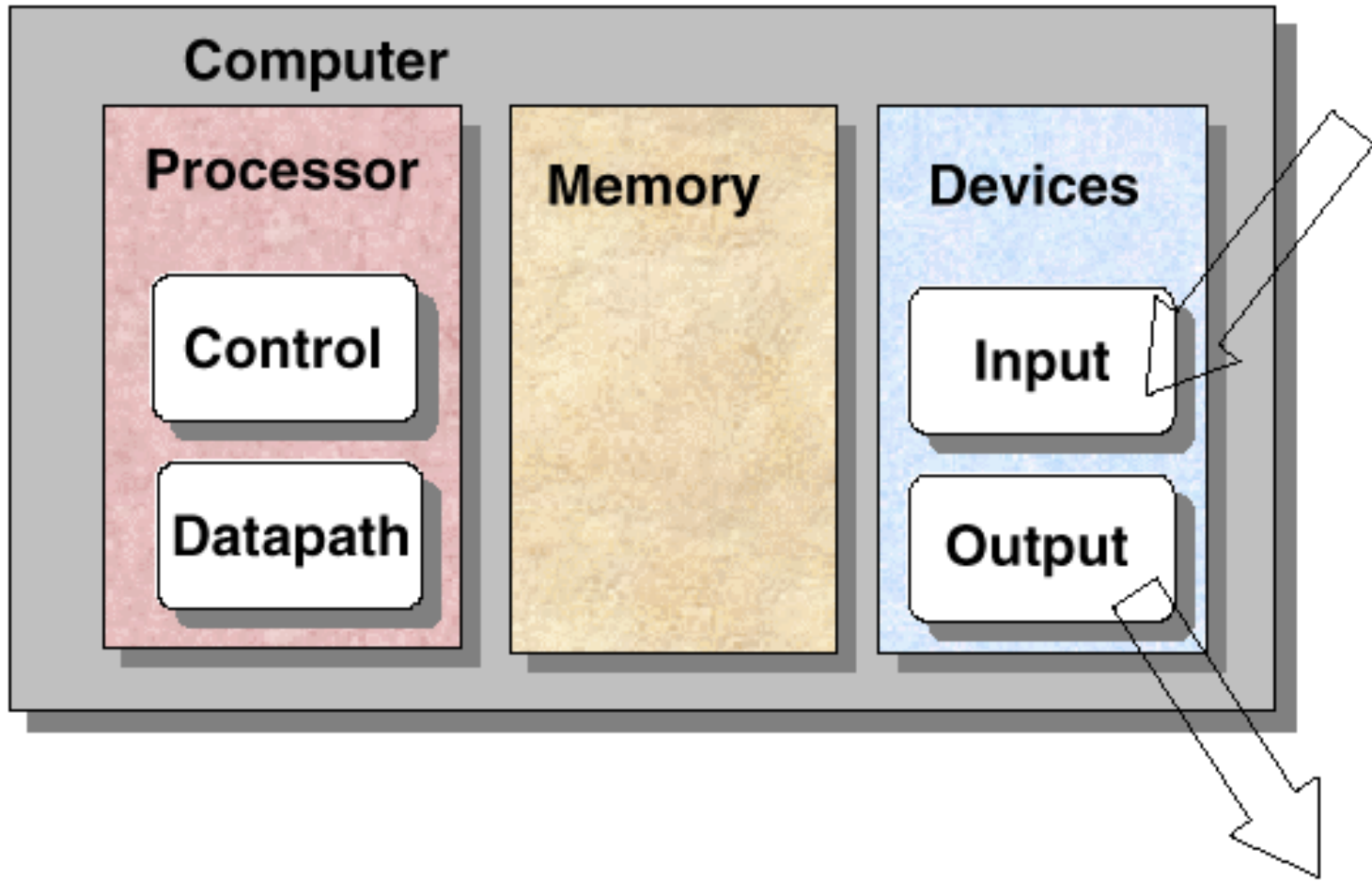
9

- 3-4 Assignments (10%)
 - 4-5 Quizzes (15%)
 - 2 Midterm Exam(s) (30%)
 - Final Exam (45%)
-
- There will be no “best-of” policy.

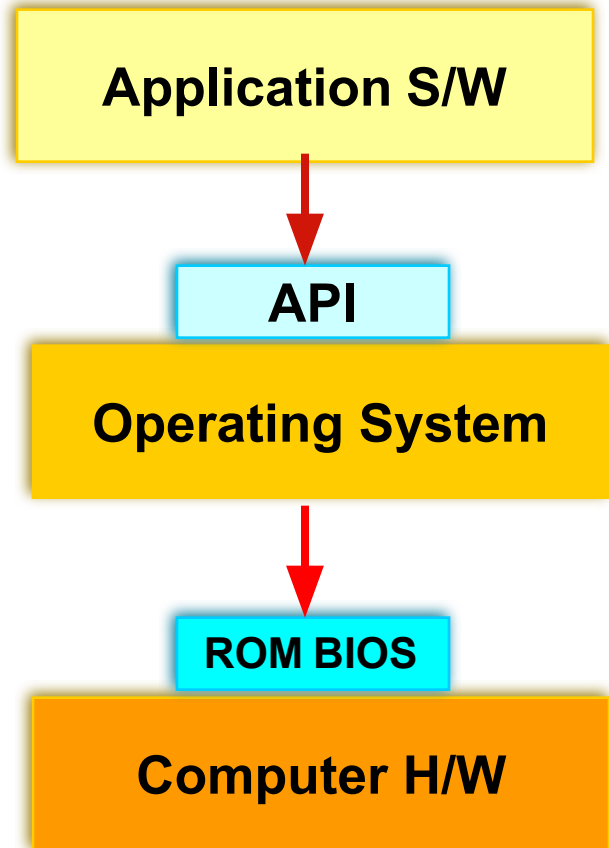
- Quizzes may be un-announced.
- No makeup for missed quiz or assignment.
- 80% attendance
- Academic integrity is expected of all the students. Plagiarism or cheating in any assessment will result in negative marking in that assessment (as per its weightage) OR more severe penalties.

A Simple Picture

11



- **Digital** – A limited number of discrete value
- **Bit** – A Binary Digit
- **Program** – A Sequence of instructions



- Computer Hardware
 - ◆ CPU
 - ◆ Memory
 - Program Memory(ROM)
 - Data Memory(RAM)
 - ◆ I/O Device
 - Input Device: Keyboard, Mouse, Scanner
 - Output Device: Printer, Plotter, Display
 - Storage Device(I/O): FDD, HDD, MOD

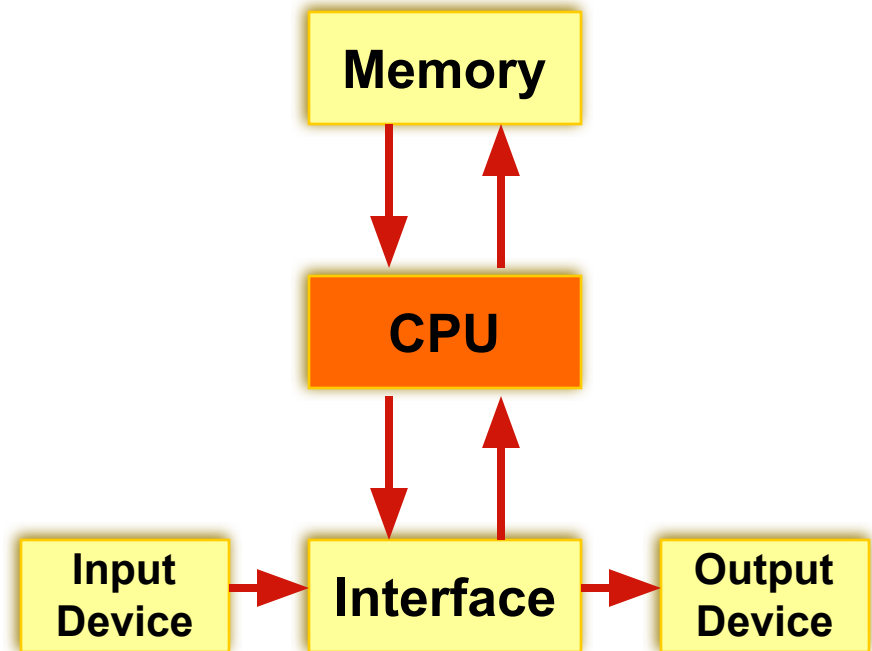
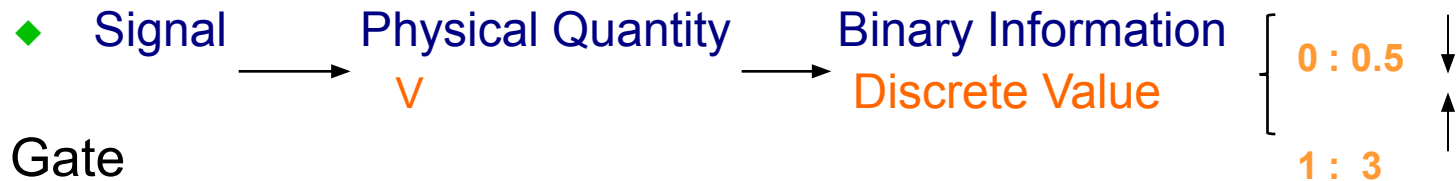


Figure 1-1 Block Diagram of a digital Computer

- What is “Computer Architecture”?
 - Hennessy and Patterson, *Computer Organization and Design*(1990)
- ◆ Computer Architecture
 - Instruction Set Architecture (ISA)
 - Machine Organization
- “ISA”?
 - ◆ Instructions
 - ◆ Addressing modes
 - ◆ Instruction and data formats
 - ◆ Register
- “Machine Organization”?
 - ◆ CPU(Control, Data path), Memory, Input, Output
 - ◆ How these components are arranged?

- ADC(Analog to Digital Conversion)



- Gate


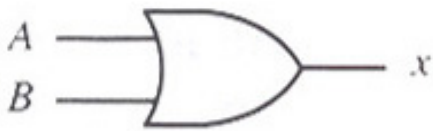
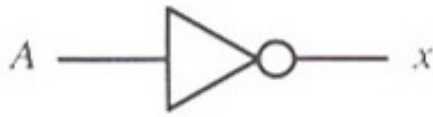
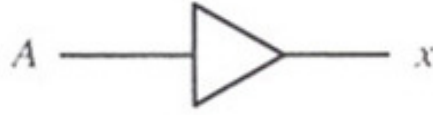
- ◆ The manipulation of binary information is done by logic circuit called "gate".

- *Fig. 1-2 Digital Logic Gates*

- ◆ AND, OR, INVERTER, BUFFER, NAND, NOR, XOR, XNOR

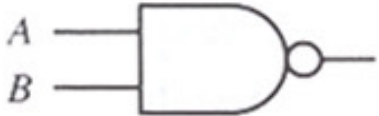
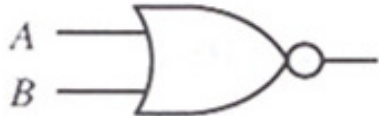


1-2 Logic Gates

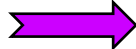
16

Name	Graphic symbol	Algebraic function	Truth table															
AND		$x = A \cdot B$ or $x = AB$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table>	A	B	x	0	0		0	1		1	0		1	1	
A	B	x																
0	0																	
0	1																	
1	0																	
1	1																	
OR		$x = A + B$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table>	A	B	x	0	0		0	1		1	0		1	1	
A	B	x																
0	0																	
0	1																	
1	0																	
1	1																	
Inverter		$x = A'$	<table><tr><th>A</th><th>x</th></tr><tr><td>0</td><td></td></tr><tr><td>1</td><td></td></tr></table>	A	x	0		1										
A	x																	
0																		
1																		
Buffer		$x = A$	<table><tr><th>A</th><th>x</th></tr><tr><td>0</td><td></td></tr><tr><td>1</td><td></td></tr></table>	A	x	0		1										
A	x																	
0																		
1																		

1-2 Logic Gates

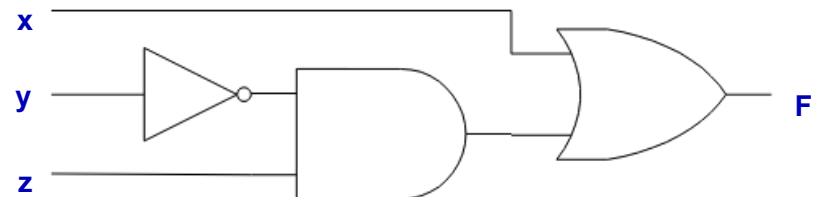
17

NAND	 $x = (AB)'$	<table> <tr> <th>A</th><th>B</th><th>x</th></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </table>	A	B	x	0	0		0	1		1	0		1	1	
A	B	x															
0	0																
0	1																
1	0																
1	1																
NOR	 $x = (A + B)'$	<table> <tr> <th>A</th><th>B</th><th>x</th></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </table>	A	B	x	0	0		0	1		1	0		1	1	
A	B	x															
0	0																
0	1																
1	0																
1	1																
Exclusive-OR (XOR)	 $x = A \oplus B$ or $x = A'B + AB'$	<table> <tr> <th>A</th><th>B</th><th>x</th></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </table>	A	B	x	0	0		0	1		1	0		1	1	
A	B	x															
0	0																
0	1																
1	0																
1	1																
Exclusive-NOR or equivalence	 $x = (A \oplus B)'$ or $x = A'B' + AB$	<table> <tr> <th>A</th><th>B</th><th>x</th></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </table>	A	B	x	0	0		0	1		1	0		1	1	
A	B	x															
0	0																
0	1																
1	0																
1	1																

- Boolean Algebra
 - ◆ Deals with binary variable(A, B, x, y: T/F or 1/0) + logic operation(AND, OR, NOT...)
- Boolean Function: variable + operation
 - ◆ $F(x, y, z) = x + y'z$
- Truth Table: *Fig. 1-3(a)*
Relationship between a function and variable
- Logic Diagram: *Fig. 1-3(b)*
Algebraic Expression 
Logic Diagram(gates)

**2ⁿ Combination
Variable n = 3**

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



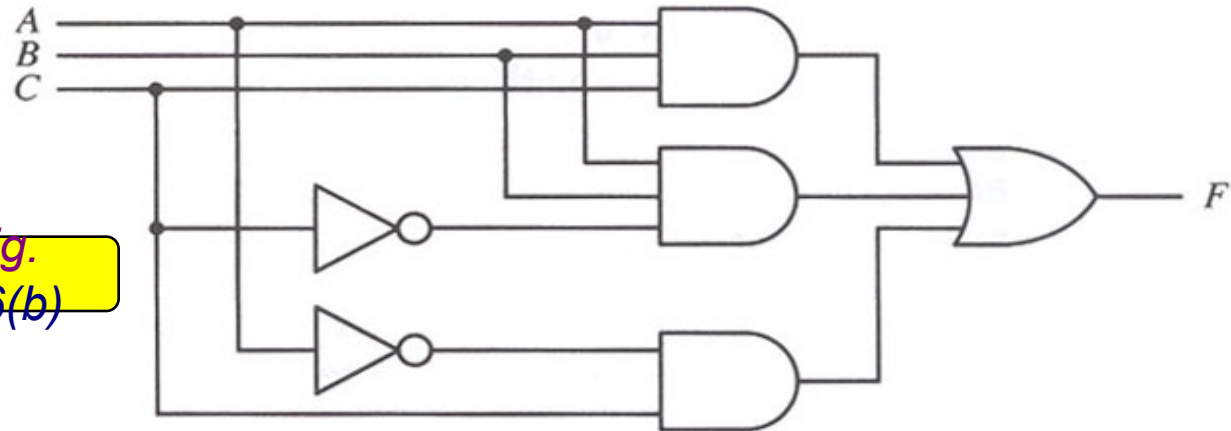
- Purpose of Boolean Algebra
 - ◆ To facilitate the analysis and design of digital circuit
- Convenient Tools
 - ◆ Truth table : relationship between binary variables
 - ◆ Logic diagram : input-output relationship
 - ◆ Find simpler circuits for the same function
- Boolean Algebra Rule : Tab. 1-1
 - **Operation with 0 and 1:** $x + 0 = x$, $x + 1 = 1$, $x \cdot 1 = x$, $x \cdot 0 = 0$
 - **Idempotent Law:** $x + x = x$, $x \cdot x = x$
 - **Complementary Law:** $x + x' = 1$, $x \cdot x' = 0$
 - **Commutative Law:** $x + y = y + x$, $x \cdot y = y \cdot x$
 - **Associative Law:** $x + (y + z) = (x + y) + z$, $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
 - **Distributive Law:** $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$, $x + (y \cdot z) = (x + y) \cdot (x + z)$
 - **DeMorgan's Law:** $(x + y)' = x' \cdot y'$, $(x \cdot y)' = x' + y'$
 - General Form:** $(x_1 + x_2 + x_3 + \dots x_n)' = x_1' \cdot x_2' \cdot x_3' \cdot \dots x_n'$
 $(x_1 \cdot x_2 \cdot x_3 \cdot \dots x_n)' = x_1' + x_2' + x_3' + \dots x_n'$

Fig.
1-6(a)

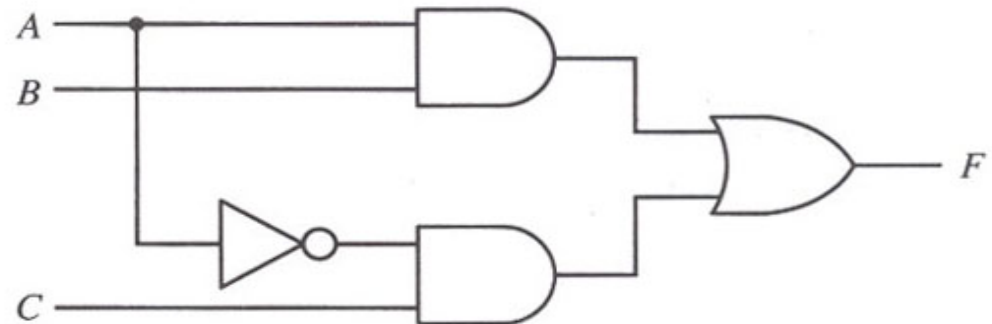
• [Example 2]

◆ $F = ABC + ABC' + A'C$
 $= AB(C + C') + A'C$
 $= AB + A'C$

Fig.
1-6(b)



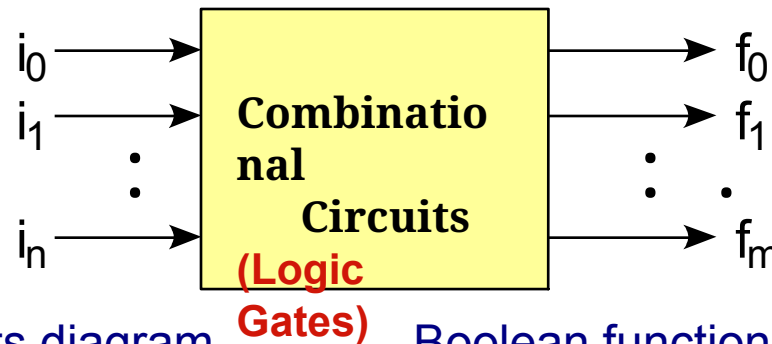
(a) $F = ABC + ABC' + A'C$



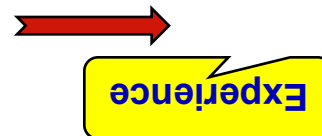
(B) $F = AB + A'C$

Figure 1-6 Two logic diagrams for the same Boolean function.

- Combinational Circuits
 - ◆ A connected arrangement of *logic gates* with a set of inputs and outputs
 - ◆ *Fig. 1-15 Block diagram of a combinational circuit*



- Analysis
 - ◆ Logic circuits diagram Boolean function or Truth table



- Design Example : Half Adder

- 1. Half adder is a combinational circuits that forms the arithmetic sum of two input bit
- 2. 2 Input(x, y), 2 Output(S: sum, C: carry)
- 3. Truth Table

Input		Output	
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- 4. Simplification

0	1
2	3

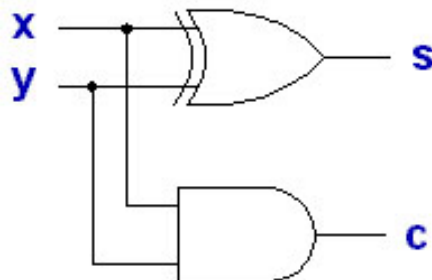
$$C = xy$$

0	1
2	3

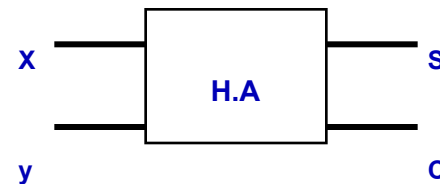
$$S = x'y + xy'$$

$$= x \oplus y$$

- 5. Logic circuit diagram



- 6. Block diagram



Design Example : Full Adder

- 1. Full adder is a combinational circuits that forms the arithmetic sum of three input bit(Carry considered)
- 2. 3 Input(x, y, z), 2 Output(S: sum, C: carry)
- 3. Truth Table

Input			Output	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

4. Simplification

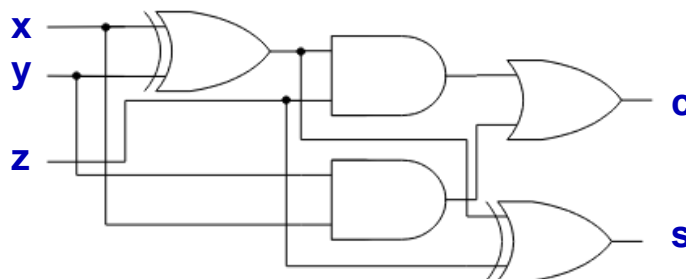
		y	
		0	1
x {	0	0	1
	1	1	0
		z	

$$\begin{aligned}
 C &= xy'z + x'y'z + xy \\
 &= z(xy' + x'y) + xy \\
 &= z(x \oplus y) + xy
 \end{aligned}$$

		y	
		0	1
x {	0	1	0
	1	0	1
		z	

$$\begin{aligned}
 S &= xy'z' + x'y'z + xy'z + x'y'z' \\
 &= z'(xy' + x'y) + z(xy' + xy) \\
 &= z'(x \oplus y) + z(x \oplus y) \\
 &= a'b + ab' \text{ (let } a=z, b=x \oplus y) \\
 &= x \oplus y \oplus z
 \end{aligned}$$

5. Logic circuit diagram



$$\begin{aligned}
 (x \oplus y)' &= (xy' + x'y)' \\
 &= (x' + y)(x + y') \\
 &= x'x + x'y' + xy + yy' \\
 &= x'y' + xy
 \end{aligned}$$

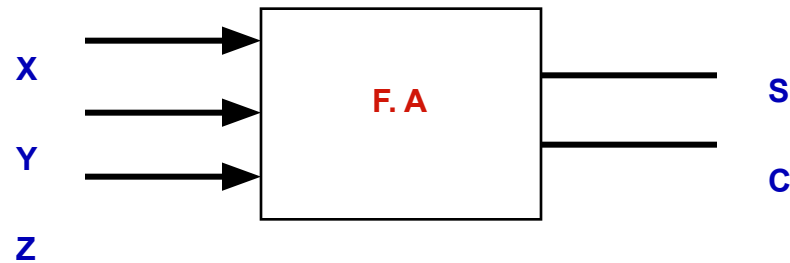
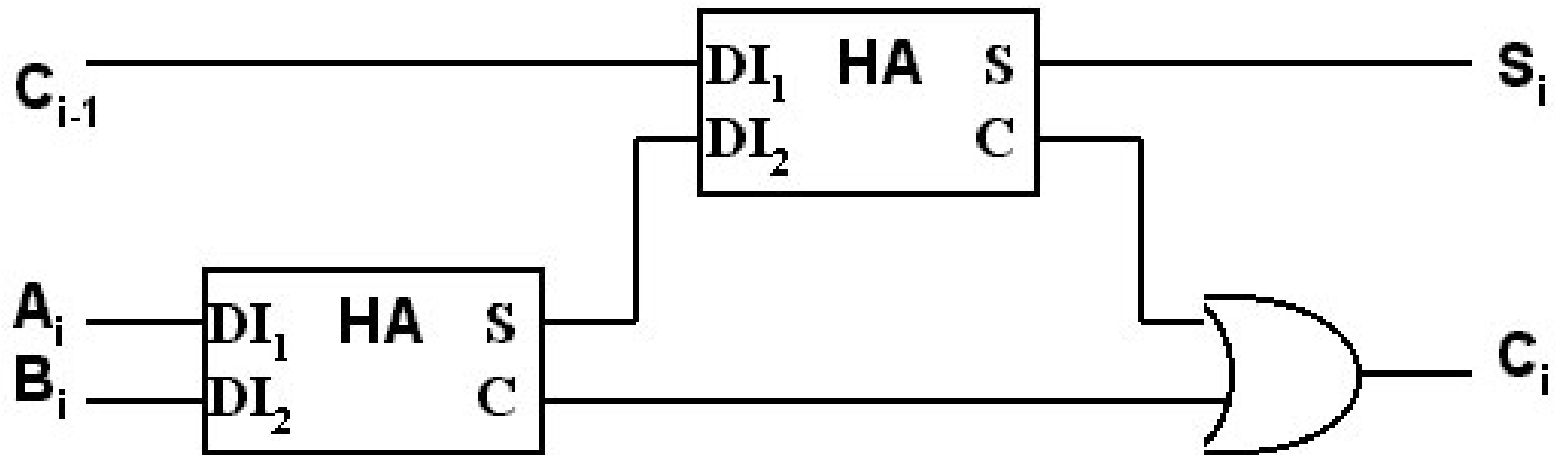


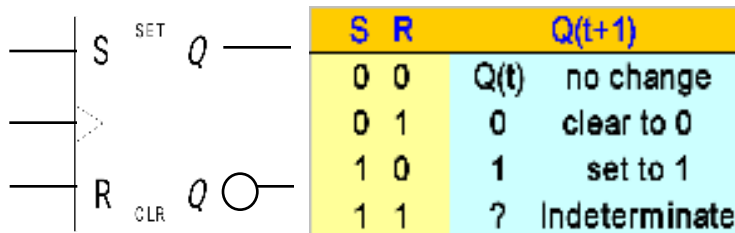
Fig 1-18. Block diagram

1-6 Flip-Flops

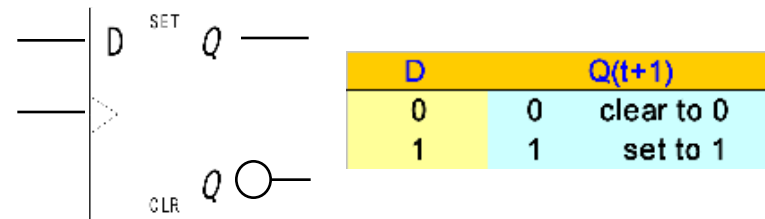
25

Combinational Circuit = Gate
Sequential Circuit = Gate + F/
F

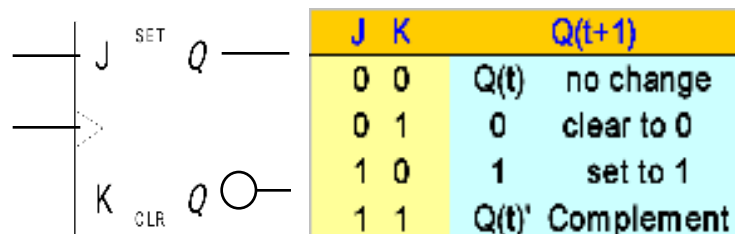
- Flip-Flop
 - ◆ The *storage elements* employed in clocked *sequential circuit*
 - ◆ A binary cell capable of storing one bit of information
- SR(*Set/Reset*) F/F



- D(*Data*) F/F

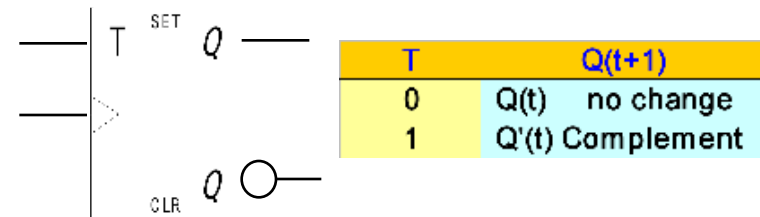


- JK(*Jack/King*) F/F



- ◆ JK F/F is a refinement of the SR F/F
- ◆ The indeterminate condition of the SR type is defined in complement

- T(*Toggle*) F/F

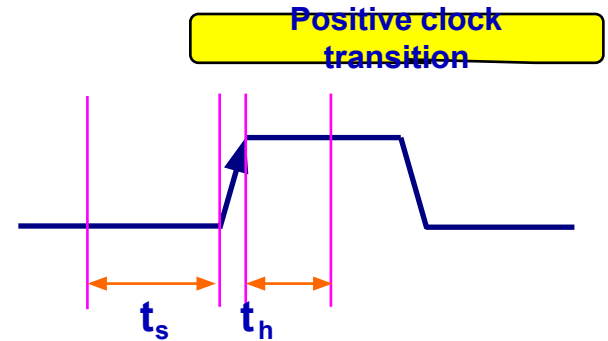


- ◆ $T=1(J=K=1)$, $T=0(J=K=0)$
- ◆ $Q(t+1) = Q(t) \oplus T$

- Edge-Triggered F/F

- ◆ State Change : *Clock Pulse*

- Rising Edge(positive-edge transition)
- Falling Edge(negative-edge transition)



- Excitation Table
 - Required input combinations for a given change of state
 - Present State $Q(t)$, Next State $Q(t+1)$

SR F/F			
$Q(t)$	$Q(t+1)$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	1

JK F/F			
$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

D F/F		
$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

T F/F		
$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

Don't
Care

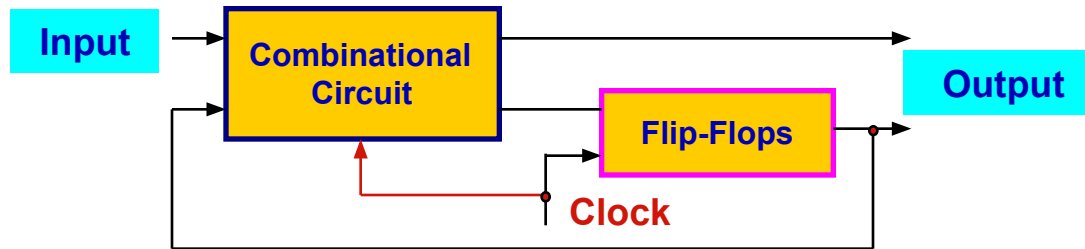
1 : Set to 1
0 : Complement

1 : Clear to 0
0 : No change

1-7 Sequential Circuits

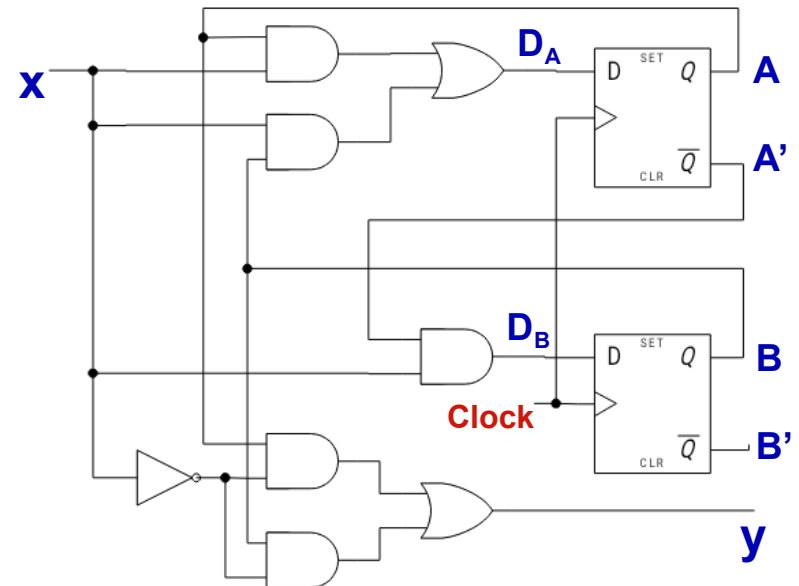
28

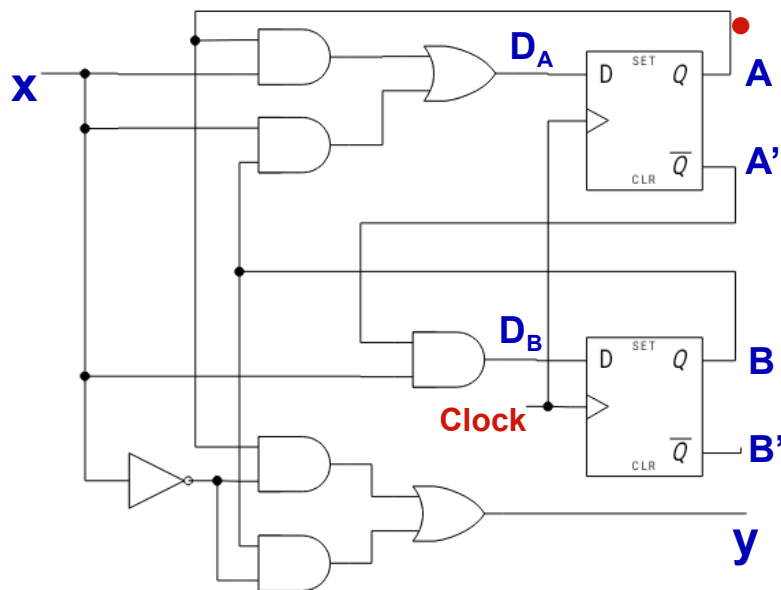
- A sequential circuit is an interconnection of F/F and Gate
- Clocked synchronous sequential circuit



Combinational Circuit = Gate
Sequential Circuit = Gate + F/
F

- Flip-Flop Input Equation
 - ◆ Boolean expression for F/F input
 - ◆ Input Equation
 - $D_A = Ax + Bx$, $D_B = A'x$
 - ◆ Output Equation
 - $y = Ax' + Bx'$
 - ◆ *Fig. 1-25 Example of a sequential circuit*

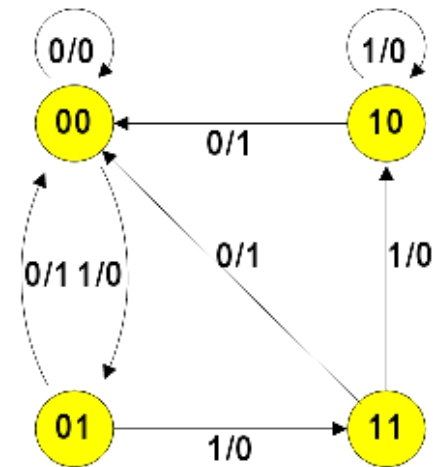




State Diagram

◆ Graphical representation of state table

- Circle(state), Line(transition), I/O(input/output)



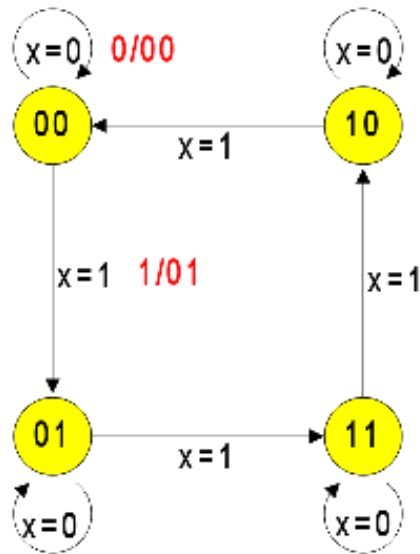
- State Table
Present state, input, next state, output

Present State		Input			Input Equ.		Next State		Output
A	B	x	Ax	Bx	D _A	D _B	A	B	y
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	1	0
0	1	0	0	0	0	0	0	0	1
0	1	1	0	1	1	1	1	1	0
1	0	0	0	0	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	1	0	0	0	0	0	0	0	1
1	1	1	1	1	1	0	1	0	0

Design Example: Binary Counter

- ◆ $x=1$: 00, 01, 10, 11, 00, 01,
- $x=0$: no change

- ◆ **State Diagram:**
4 state(00, 01, 10, 11)



Next State =
Output

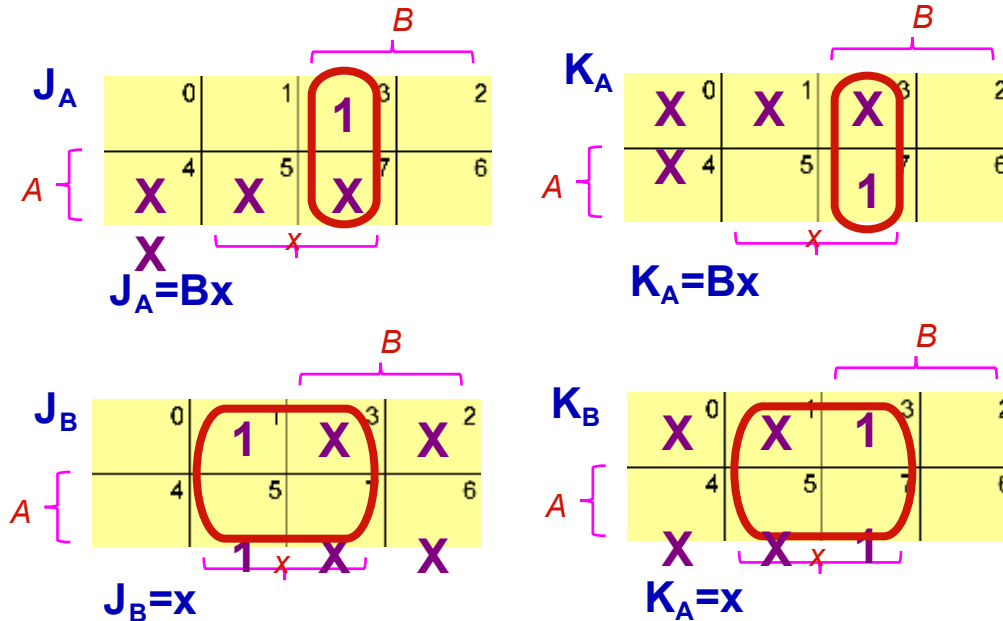
JK F/F			
Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

- ◆ **Excitation Table(2 bit counter = 2 F/F)**

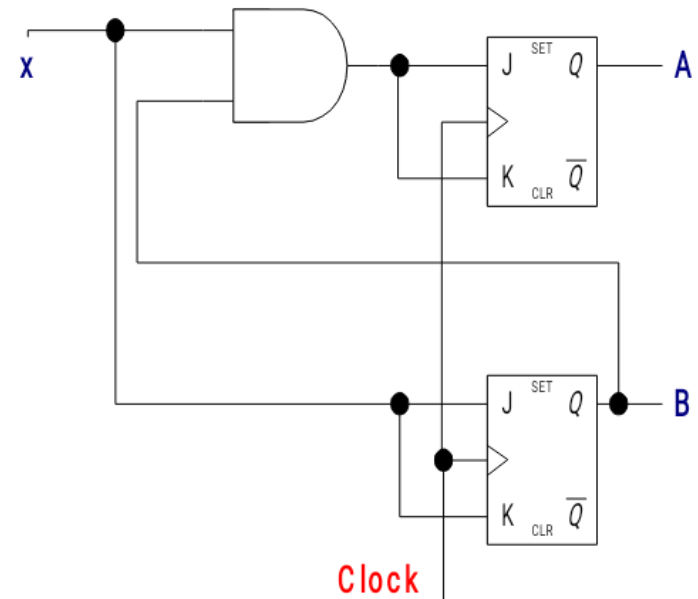
Present State		Input	Next State		F/F Input			
A	B	x	A	B	J _A	K _A	J _B	K _B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	0	1	x	x	1
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

Map for simplification

- Input variable: A, B, x



Logic Diagram



Sequential Circuit Design Procedure

1. The Problem is stated
2. I/O variables are assigned
3. Truth table(I/O relation)
4. Simplified Boolean Function
5. Logic circuit diagram