

**Shift Registers**

**Serial Addition**

**Engr. Rimsha**

# Today's Lecture

- Registers
  - Register with Parallel load
- Shift Registers
  - Serial Transfer
  - Serial Addition
  - Universal shift register
- Serial Addition

# Flip Flop Conversion Procedure

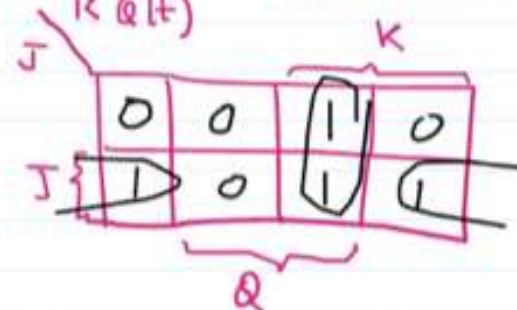
## T Flip Flop to JK Conversion

Step No 1 Make characteristic Table of JK Flip Flop Step 2

| J | K | Q(t) | Q(t+1) | T |
|---|---|------|--------|---|
| 0 | 0 | 0    | 0      | 0 |
| 0 | 0 | 1    | 1      | 0 |
| 0 | 1 | 0    | 0      | 0 |
| 0 | 1 | 1    | 0      | 1 |
| 1 | 0 | 0    | 1      | 1 |
| 1 | 0 | 1    | 1      | 0 |
| 1 | 1 | 0    | 1      | 1 |
| 1 | 1 | 1    | 0      | 1 |

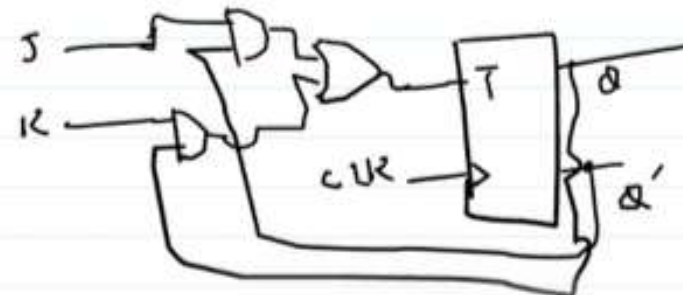
Step 2 Looking at change of state find T input accordingly

Step 3 Simplify T as output based on J, K and Q(t)



$$T = JQ' + KQ$$

Step 4 Given T Flip Flop draws combinational circuit at input of T.



# Flip Flop Conversion Procedure : Example

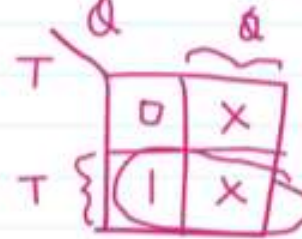
## JK Flip Flop to T FF Conversion

Step No 1 Make characteristic Table of T Flip Flop Step 2

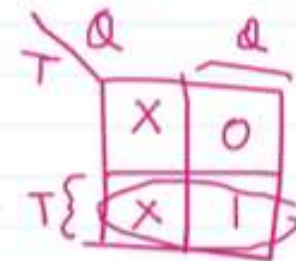
| T | Q(t) | Q(t+1) | J | K |
|---|------|--------|---|---|
| 0 | 0    | 0      | 0 | x |
| 0 | 1    | 1      | x | 0 |
| 1 | 0    | 1      | 1 | x |
| 1 | 1    | 0      | x | 1 |

Step 2 Looking at change of state find JK input accordingly

Step 3 Simplify J, K as outputs based on T and Q(t)

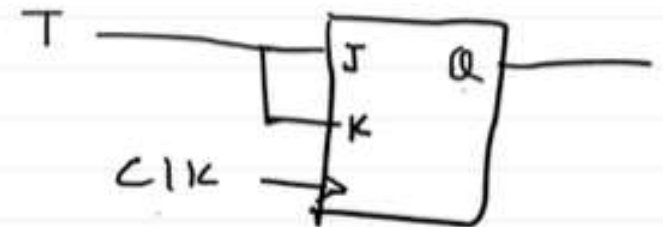


$$J = T$$



$$K = T$$

Step 4 Given JK Flip Flop draws combinational circuit at input of J & K



## Registers

- **Clocked sequential** circuits consist of:
  - a group of flip-flops and combinational gates connected to form a feedback path
- Flip-flops are essential because in their absence, the circuit reduces to a purely combinational circuit. (provided there is no feedback among the gates)
- Circuits that include flip-flops are usually classified by the function they perform rather than by the name of the sequential circuit
- Two such circuits are **registers** and **counters**

# Registers

- Register

- A register is a group of flip-flops, each flip flop capable of storing one bit of information
- A register may have combinational gates that determine how the information is transferred into the register
- An  $n$ -bit register consist of a group of  $n$  flip-flops capable of storing  $n$  bits of binary information. There are different types of registers available commercially

- Counter

- A counter is a special type of register that goes through a predetermined sequence of states
- The gates in the counter are connected in such a way to produce the prescribed sequence of binary states

## Simple Register

- The simplest **register** is one that consists of only **flip-flops** without any gates
- A n-bit register consists of n flip-flops capable of storing n bits of binary information
- Fig shows a 4-bit register with four D-type flip-flops. The information is loaded in parallel.
- The four outputs can be sampled at any time to obtain the binary information stored in the register
- The **clear** input goes to R (reset) input of all four flip flops
- When this input goes to zero all flip-flops are **reset** (to all 0's) asynchronously

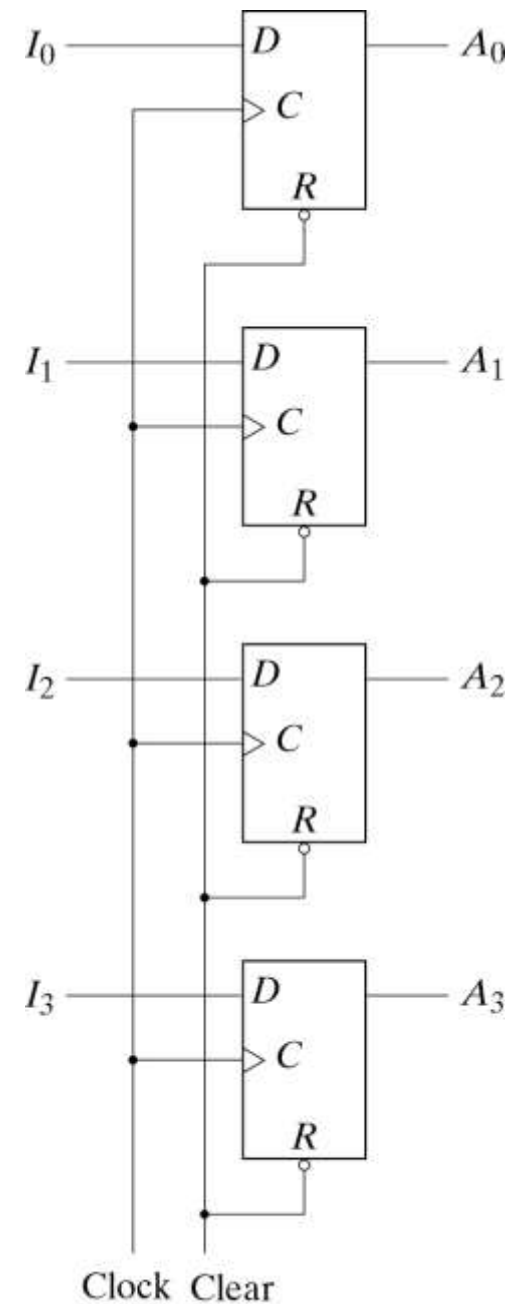


Fig. 6-1 4-Bit Register

# Register with Parallel Load

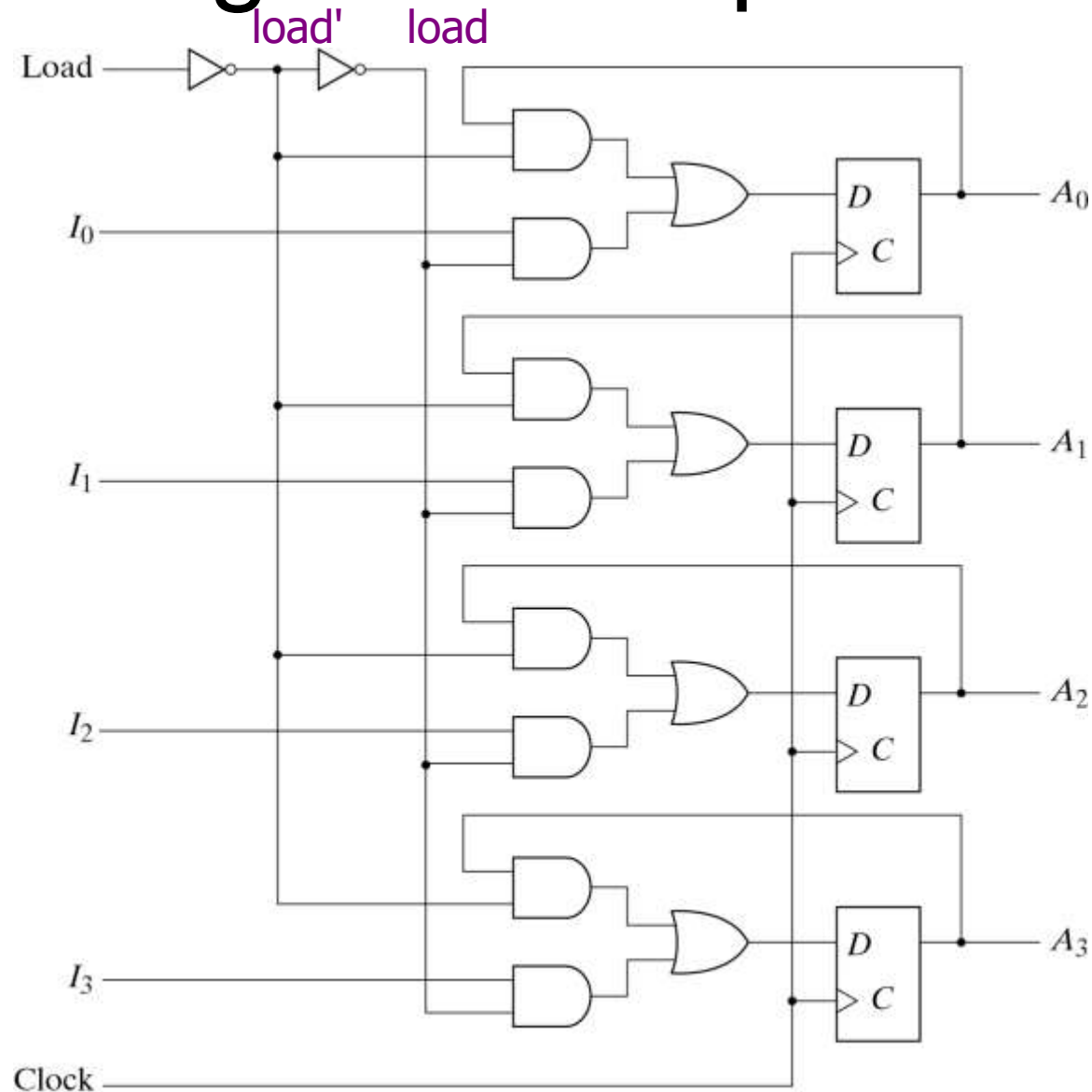
- **Synchronous** digital systems have a **master clock** generator that supplies a continuous train of clock pulses
- A separate **control** signal must be used to **decide** which **specific clock pulse** will have an effect on a particular register
- The **transfer** of new **information** into a register is referred to as **loading** the register
- If all the bits are loaded **simultaneously** with a common clock pulse we say that the loading is done in **parallel**
- A clock edge applied to C inputs of the register (Fig 6-1) will load all inputs in parallel
- If the content of the register be left **unchanged** then the **clock** must be **inhibited** from reaching the register



# Register with Parallel Load

- The clock can be inhibited from reaching the register by controlling the clock input signal with an enabling gate. Clock pulses perform the logic
- This insertion of **gates** produce **uneven propagation delays** between the master clock and the inputs of flip-flops
- To **synchronize** the system we must ensure that clock **pulses** must arrive at the **same time** anywhere in the system so that all flip-flops trigger simultaneously
- Performing logic with clock pulses, inserts variable delays and may cause the system to go out of synchronism
- For this reason it is advisable to **control** the operation of register with the **D inputs** rather than controlling the clock in the C inputs of flip flops

# 4-bit register with parallel load



# Register with Parallel Load

- A 4-bit register with a load control input that is directed through gates and into the D inputs of the flip-flops
- The load input to the register determines the action to be taken with each clock pulse
- When the **load** input is **1**, the data in the four inputs are **transferred** into the register with next **positive edge** of the clock
- When the **load** input is **0**, the outputs of the flip-flops are connected to their respective inputs
- The **feedback** connection from output to inputs is necessary because the D-type flip-flop doesn't have a "**no change**" condition
- With each clock edge the D input determines the next state of the register. To leave the **output unchanged** it is necessary to make the **D input** equal to **present value** of the **output**

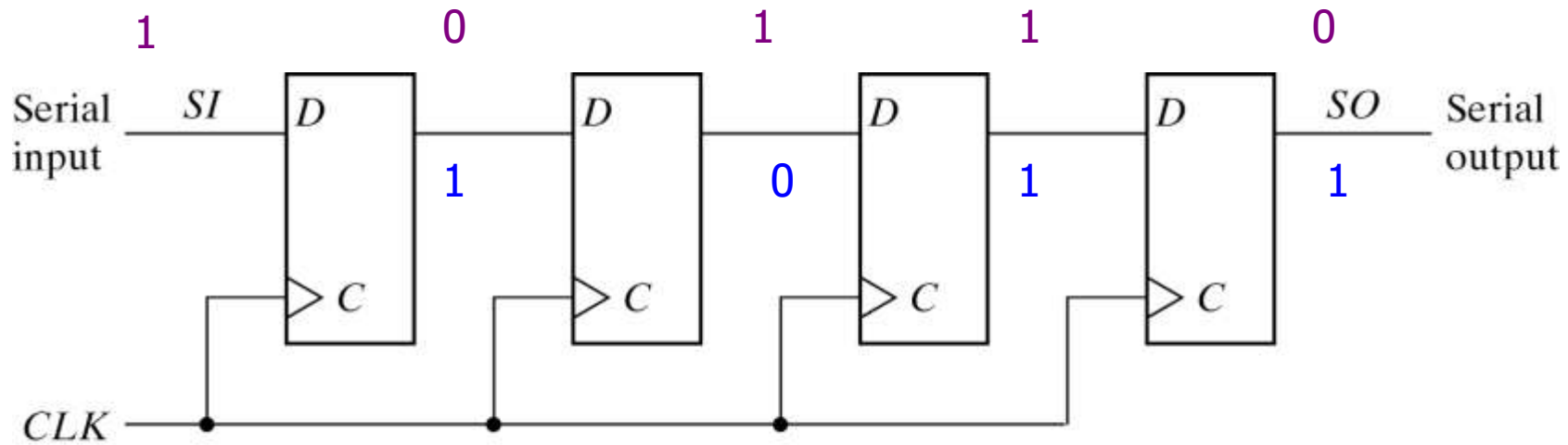
# Register with Parallel Load

- The clock pulses are applied to the C inputs at all times. The **load** input **determines** whether the next pulse will accept new information or leave the information in the register intact
- The transfer of information from the data inputs or the outputs of the register is done **simultaneously** with all four bits in response to clock edge

# Shift Registers

- A **Shift register** is a register capable of **shifting** its binary **information** in one or both directions
- It consists of a chain of **flip-flops** in **cascade**, with output of one flip-flop connected to the input of next flip-flop
- All flip-flops receive **common clock pulses** which activate the shift from one stage to the next
- A simplest possible shift register uses only flip flops and is shown in figure in next slide

# Shift Registers



# Shift Registers

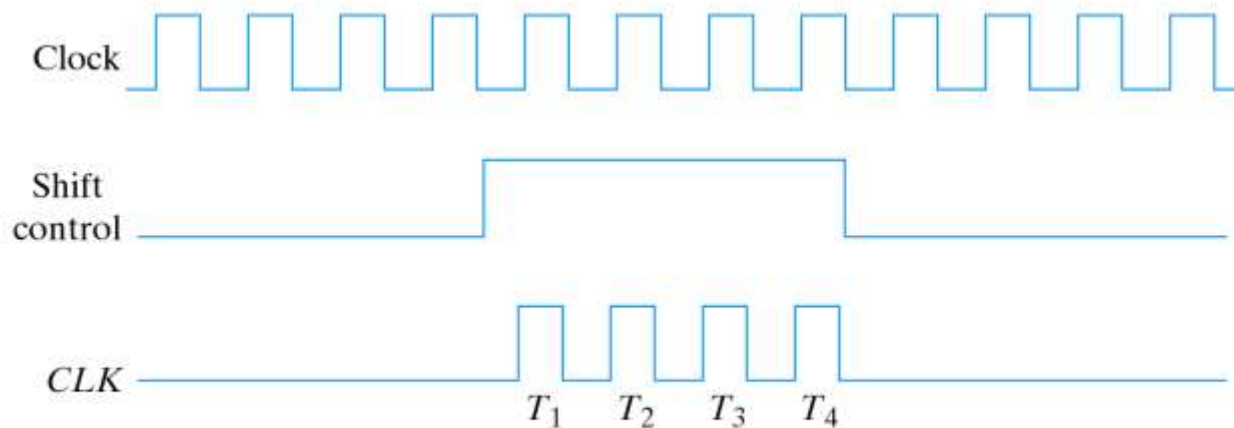
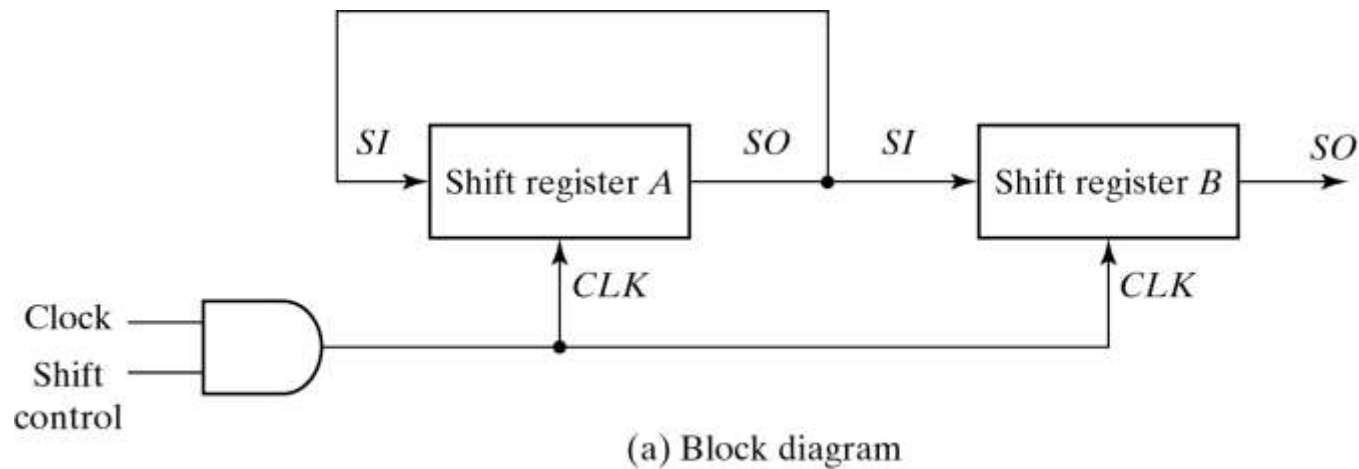
- The output of a given flip-flop is connected to the D input of the flip-flop at its right
- Each clock pulse **shifts** the contents of the register **one bit position** to the right
- The **serial input** determines what goes into the left-most flip-flop during the shift
- The **serial output** is taken from the output of the rightmost flip-flop
- If it is required to **control** the shift so that it occurs only with certain pulses but not with others then we **inhibit** the clock from the input of the register to prevent it from shifting. Shift is then controlled by connecting the clock through an AND gate with an input that controls the shift

# Serial Transfer

- A digital systems can work either in **serial** transfer mode or **parallel** transfer mode
  - Serial transfer
    - Information is transferred one bit at a time
    - shifts the bits out of the source register into the destination register
  - Parallel transfer:
    - All the bits of the register are transferred at the same time
- The serial transfer of information from register A to register B is done with shift registers as shown in figure on next slide
- The serial output (SO) of register A is connected to the serial input (SI) of register B
- To **prevent the loss** of information stored in the source register, the information in register A is made to **circulate** by connecting the serial output to its serial input



## Example: Serial Transfer from Register A to Register B



### Example: Serial Transfer from Register A to Register B

- The initial content of **register B** is shifted out through its serial output and is **lost** unless it is transferred to a third shift register
- The **shift control** input determines when and how many **times** the registers are **shifted**
- This is done with an AND gate that allows clock pulse to pass into the CLK terminals only when the shift **control** is **active**
- Suppose the shift register have **four bits** each. The control unit that supervises the transfer must be designed in such a way that it enables the shift register, through the shift control signal, for a fixed time of **four clock pulses**. (as shown in timing diagram in the figure)

## Serial Transfer from Register A to Register B-*continued*

- The **shift control** is **synchronized** with the **clock** and changes value just after the **negative edge** of the clock
- The next four clock pulses find the shift control signal in the active state so that the output of the AND gate connected to the CLK input produces four pulses  $T_1$ ,  $T_2$ ,  $T_3$  and  $T_4$ . Each **rising edge** of pulse causes a **shift** in **both registers**
- The fourth pulse changes the shift control to 0 and the shift registers are disabled
- Assume the binary contents of A before the shift is 1011 and that of B is 0010
- The serial transfer from A to B occurs in four steps as shown in table 6-1

## Serial Transfer from Register A to Register B-continued

- With the first pulse  $T_1$ , the **rightmost** of **A** is shifted into **leftmost** bit of **B** and is also **circulated** into the leftmost position of A
- At the same time all bits of A and B are shifted **one position** to the **right**
- The previous serial output from B in the rightmost position is lost and its value changes from 0 to 1
- The next three pulses perform identical operations, shifting the bits of A into B, one at a time
- After the fourth shift, the shift control goes to 0 and both registers A and B have the value 1011

**Table 6-1**  
*Serial-Transfer Example*

| Timing Pulse  | Shift Register A | Shift Register B |
|---------------|------------------|------------------|
| Initial value | 1 0 1 1          | 0 0 1 0          |
| After $T_1$   | 1 1 0 1          | 1 0 0 1          |
| After $T_2$   | 1 1 1 0          | 1 1 0 0          |
| After $T_3$   | 0 1 1 1          | 0 1 1 0          |
| After $T_4$   | 1 0 1 1          | 1 0 1 1          |

## Serial Transfer from Register A to Register B-*continued*

- The contents of **A** are **transferred** into **B**, while the contents of **A** remain **unchanged**
- This is a **serial transfer** where the registers have a single serial input and a single serial output. The information is transferred one bit at a time while the registers are shifted in the same direction

**Table 6-1**  
*Serial-Transfer Example*

| <b>Timing Pulse</b> | <b>Shift Register A</b> | <b>Shift Register B</b> |
|---------------------|-------------------------|-------------------------|
| Initial value       | 1 0 1 1                 | 0 0 1 0                 |
| After $T_1$         | 1 1 0 1                 | 1 0 0 1                 |
| After $T_2$         | 1 1 1 0                 | 1 1 0 0                 |
| After $T_3$         | 0 1 1 1                 | 0 1 1 0                 |
| After $T_4$         | 1 0 1 1                 | 1 0 1 1                 |

# Overview of Lecture

- Shift Registers
  - Shift left and right by one bit means?
    - Shift left mean multiply by 2
    - Shift right mean divide by 2
  - Serial transfer from one register to another register?
    - Needs four clock pulses to transfer four bits

## Today's Lecture

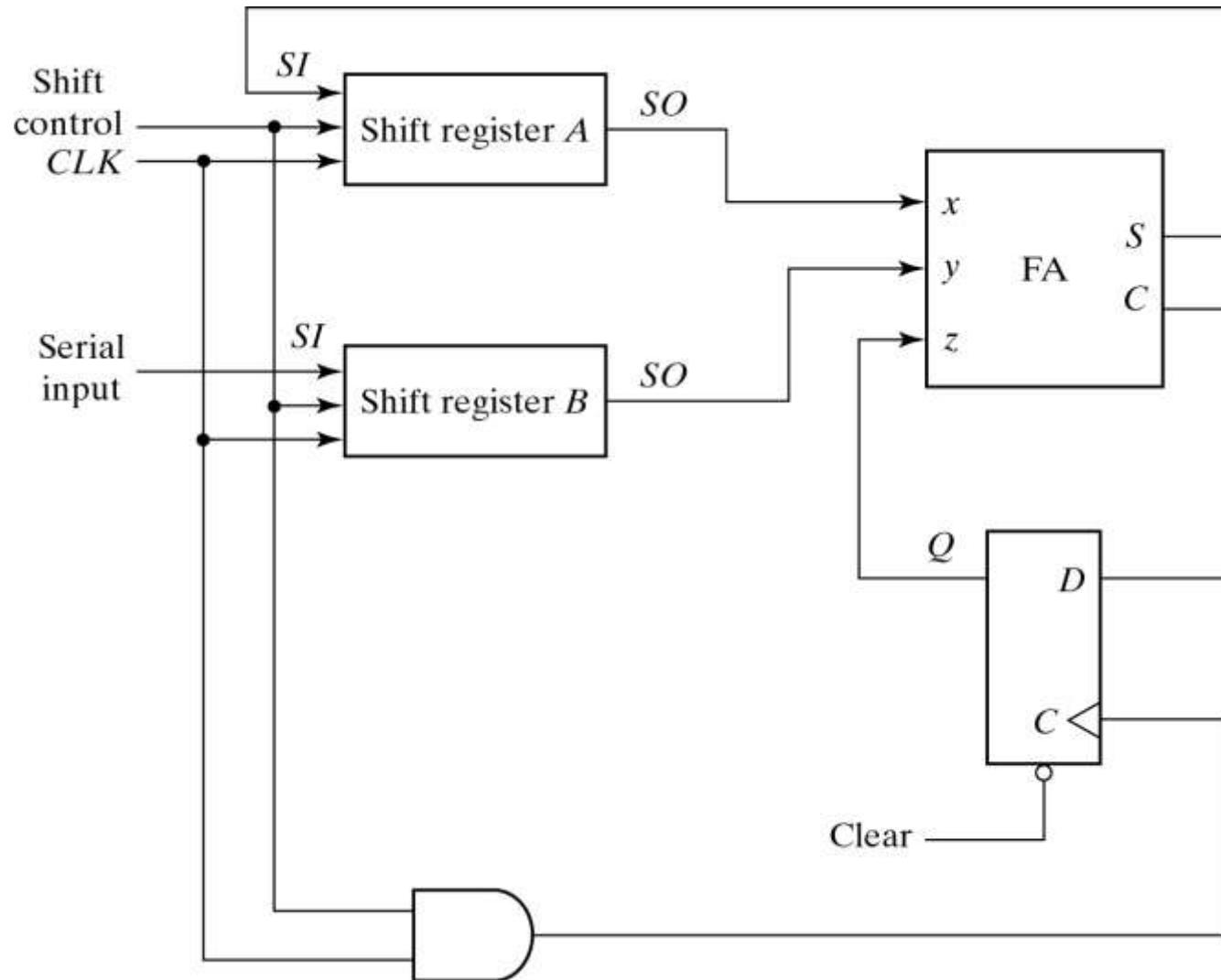
- Serial Addition
- Shift Registers
  - Universal shift register
- Ripple Counters
  - Binary Ripple Counter
  - BCD Ripple Counter
- Synchronous Counters
  - Binary counter
  - 4 bit up/down counter
  - BCD counter
  - Binary counter with parallel load

# Serial Addition

- Operation in digital computers are usually done in **parallel** because this is **faster** mode of operation
- **Serial** operations are **slower** but have the advantage of requiring **less equipment**
- **How to design a serial Adder?**
  - What resources are required?
  - How to connect them?



# Serial addition using D flip-flops



# Serial Addition

- The two binary numbers to be added serially are stored in two shift registers
- Bits are added one pair at a time through a single full adder (FA) circuit, as shown in Fig 6-5
- The carry out of the full adder is transferred to a D flip-flop
- The output of this flip-flop is then used as the carry input for the next pair of significant bits

# Serial Addition

- The sum bit from the S output of the full adder could be transferred into a third shift register
- By shifting the sum into A while the bits of A are shifted out, it is possible to use one register for storing both the augend and the sum bits
- The serial input of register B can be used to transfer a new binary number while the addend bits are shifted out during the addition

## Serial Adder using JK flip-flops

**Table 6-2**  
*State Table for Serial Adder*

| Present State | Inputs |   | Next State | Output | Flip-Flop Inputs |       |
|---------------|--------|---|------------|--------|------------------|-------|
|               |        |   |            |        | $J_Q$            | $K_Q$ |
| Q             | X      | y | Q          | S      |                  |       |
| 0             | 0      | 0 | 0          | 0      | 0                | X     |
| 0             | 0      | 1 | 0          | 1      | 0                | X     |
| 0             | 1      | 0 | 0          | 1      | 0                | X     |
| 0             | 1      | 1 | 1          | 0      | 1                | X     |
| 1             | 0      | 0 | 0          | 1      | X                | 1     |
| 1             | 0      | 1 | 1          | 0      | X                | 0     |
| 1             | 1      | 0 | 1          | 0      | X                | 0     |
| 1             | 1      | 1 | 1          | 1      | X                | 0     |

$$J_Q = x y$$

$$K_Q = x' y' = (x + y)'$$

$$S = x \oplus y \oplus Q$$

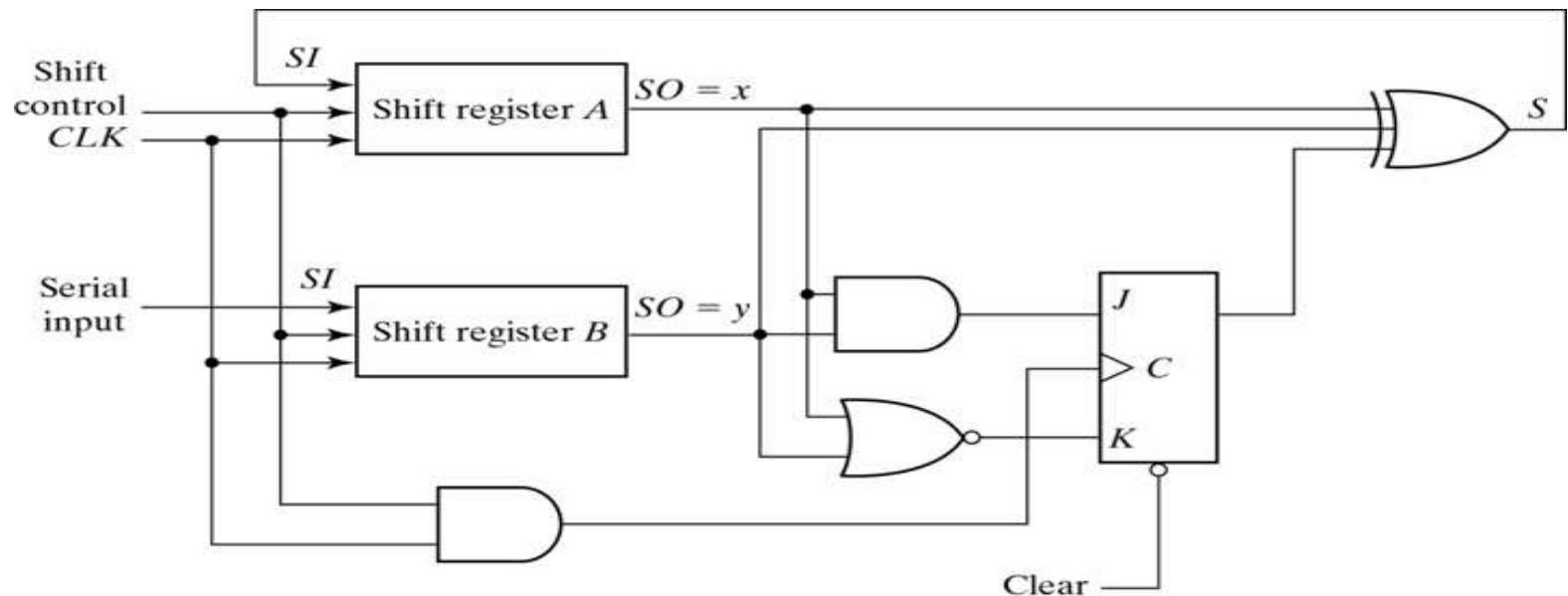
# Serial Adder using JK Flip-flops- continued

– Circuit diagram

$$J_Q = x y$$

$$K_Q = x' y' = (x + y)'$$

$$S = x \oplus y \oplus Q$$



THE END