Sequential Circuits

Instructor: Saraju P. Mohanty

- Sequential circuit analysis
- Sequential circuit design
- Designing with D Flip-Flop
- Designing with JK Flip-Flop
- Registers
- Counters

Note: The slides are from text or reference book authors or publishers.

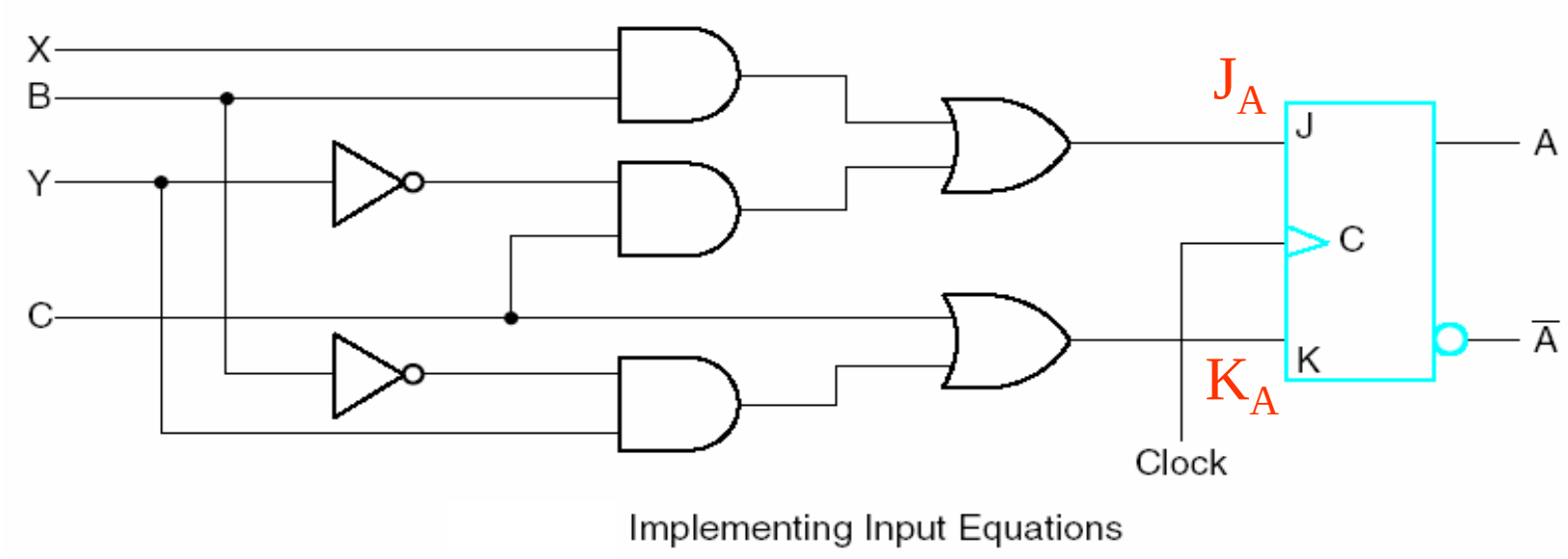
Sequential Circuit Analysis: basics

- Behavior is determined from input, output, and present state of the circuit.
- The output is dependent on the present input and the present states in case of a Mealy machine.
- In a Moore machine, the output is only a function of the present states.
- The next state functions are dependent on the present input and the present states.
- The input equations, state tables and state diagrams are three important tools to describe a sequential machine.

Sequential Circuit Analysis: Input Equations

The part of the combinatorial circuit that generates the signals for the inputs of flip-flops can be described by a set of Boolean expressions called flip-flop input equations.

For example: $J_A = (XB+Y'C)$, $K_A = (YB'+C)$



Implementing Input Equations

Convention: J and K are the input symbols of a FF and A is the output of the FF which is used as subscript.

Sequential Circuit Analysis: State Table

- State table specify the functional relationship between the inputs, outputs and FF states of a sequential circuits.
- The table consists of four sections, labeled present state, input, next state, and output.

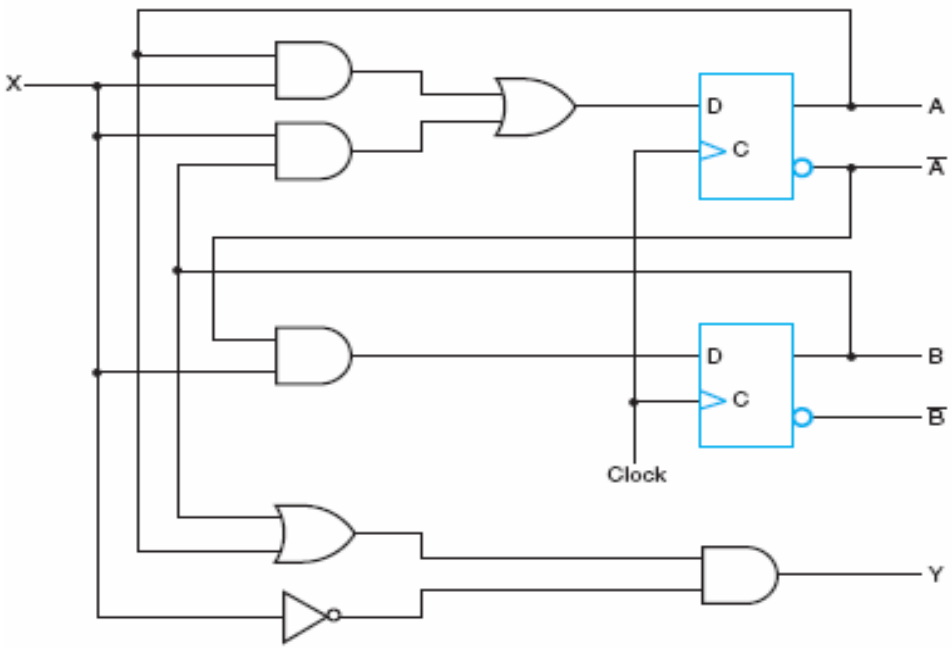
Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

State Table for Circuit

Sequential Circuit Analysis: State Table

- The derivation of a state table has following steps:
  - List all possible binary combinations of present state and inputs.
  - Determine next states from logic diagram or from flip-flop equations.
- In general, a sequential circuit with m FFs and n inputs needs 2^{m+n} rows in state table. The next state section has m columns, one for each FF.

Sequential Circuit Analysis: State Table



For Mealy circuits, two dimensional state tables can be used. Two dimensional, meaning the present state and input combination are separated.

Example given below

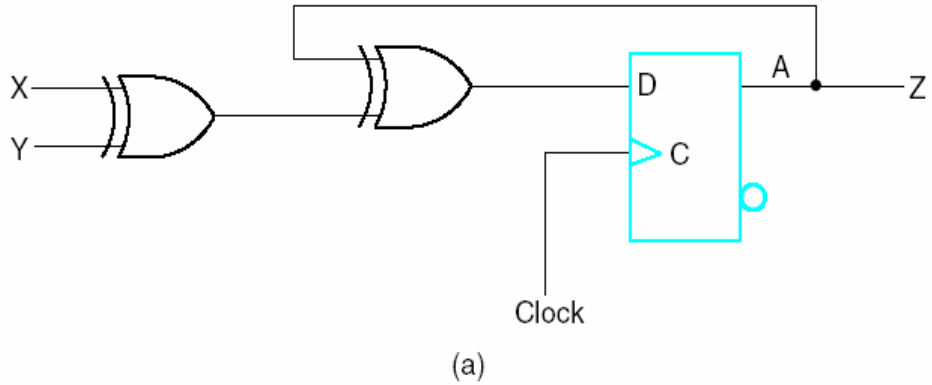
Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

State Table for Circuit

Present state		Next state				Output	
		X = 0		X = 1		X = 0	X = 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

Two-Dimensional State Table for the Circuit

Sequential Circuit Analysis: State Table



(a)

Present state	Inputs		Next state	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b) State table

Logic Diagram and State Table for $D_A = A \oplus X \oplus Y$

Example Moore Circuit

- The output is function of states only.
- The state table is given.
- The flip-flop input equations:

$$D_A = A \text{ XOR } X \text{ XOR } Y$$

- The output equation:

$$Z = A$$

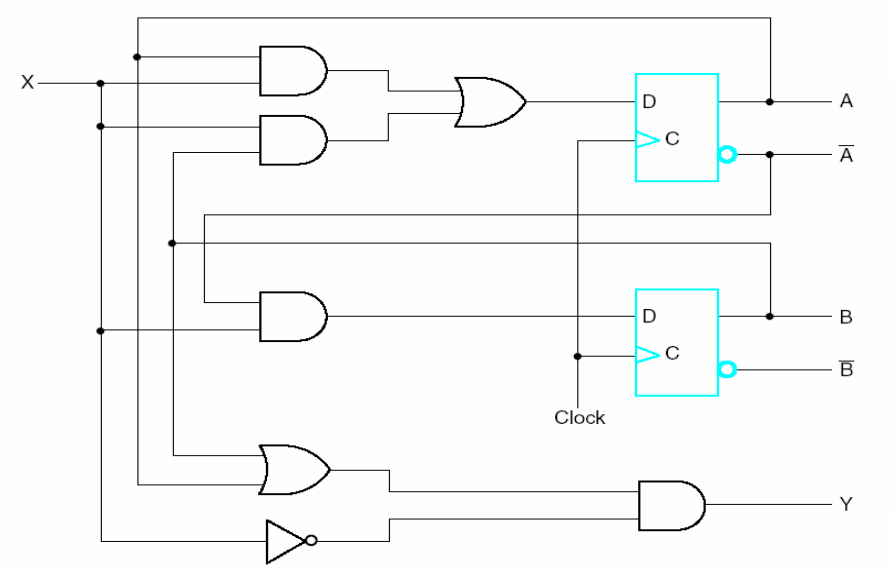
Sequential Circuit Analysis: State Table

- The analysis of circuits with other FFs can be performed similarly.
- For circuits with other types of FFs, such as JK, the next-state values are obtained by following a two-step procedure:
  - Obtain the binary values of each FF input equation in terms of the present-state and input variables
  - Use the corresponding FF characteristic to determine the next-state.

JK Flip-Flop

J	K	$Q(t+1)$	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

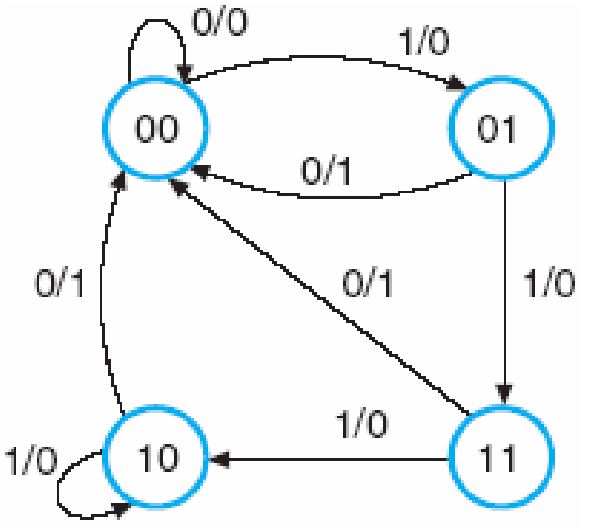
Sequential Circuit Analysis: State diagrams



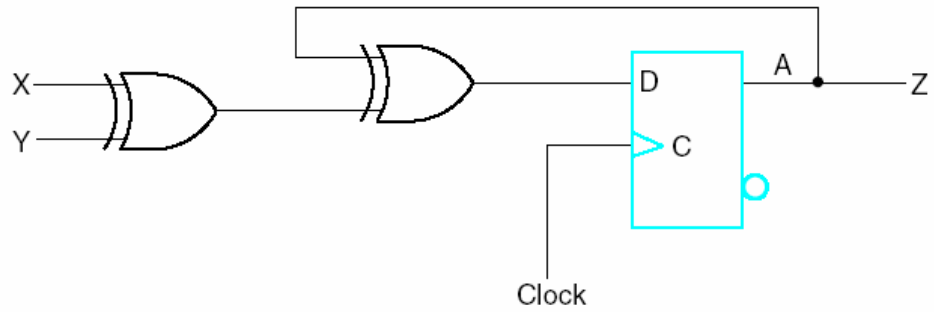
Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

State Table for Circuit

Binary number inside each circle identifies FF states. Directed lines are labeled with two binary numbers separated by /. The input value during the present state precedes slash, value following slash gives output value during present state with given input applied. Example: directed line from 00 to 01 is 1/0, meaning when circuit in present state 00 and input is 1, output is 0. After next clock transition, circuit goes to 01.



Sequential Circuit Analysis: State diagrams



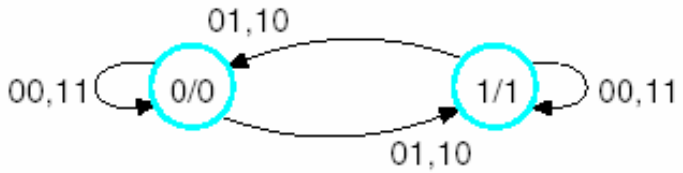
(a)

Present state	Inputs		Next state	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b) State table

Logic Diagram and State Table for $D_A = A \oplus X \oplus Y$

Only one FF with two states is needed. Two binary inputs and the output depends only on the state of the FF. Slash is not included, since outputs depend only on the state and not input values. Two input conditions for each state transition, separated by a comma.



(b)

Sequential Circuit Analysis: Few Points

- There is no difference between a state table and a state diagram.
- The state diagram is a graphical representation of the state table.
- The state table is easier to derive from a given logic diagram and input equations.
- The state diagram directly follows from the state table.
- The state diagram that gives pictorial view of state transitions is more suitable for human interpretation.
- In general, a Moore machine requires more states than a Mealy machine.

Sequential Circuit Design

1. Obtain either the state diagram or the state table from problem statement.
2. If only a state diagram is available from step 1, obtain state table.
3. Assign binary codes to the states.
4. Derive the FF input equations from the next-state entries in the encoded state table.
5. Derive output equations from the output entries in the state table.
6. Simplify the FF input and output equations.
7. Draw the logic diagram with D flip-flops and combinatorial gates as specified by the FF input and output equations.

Sequential Circuit Design: Example

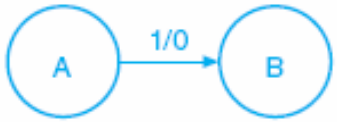
Problem statement: The circuit has one input X and one output Z. Recognize the occurrence of a sequence of 1101 on X by making Z equal to 1, when the previous three inputs to the circuits were 110 and current input is a 1. Otherwise Z equals 0.

Key Point

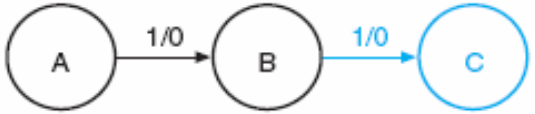
States are used to “remember” something about history of past inputs. For example, for sequence 1101, in order to be able to produce the output value 1 coincident with the final 1 in the sequence, the circuit must be in a state that remembers previous three inputs.

Sequential Circuit Design: Example …

- "A" is a state in which none of the first portion of the sequence to be recognized has occurred.
- If input is "1", since 1 is the first bit in the sequence, the event must be remembered and a new state is established. State B remembers "1".

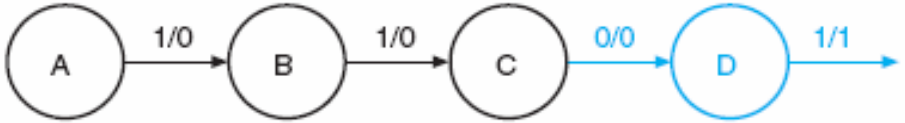


(a)



(b)

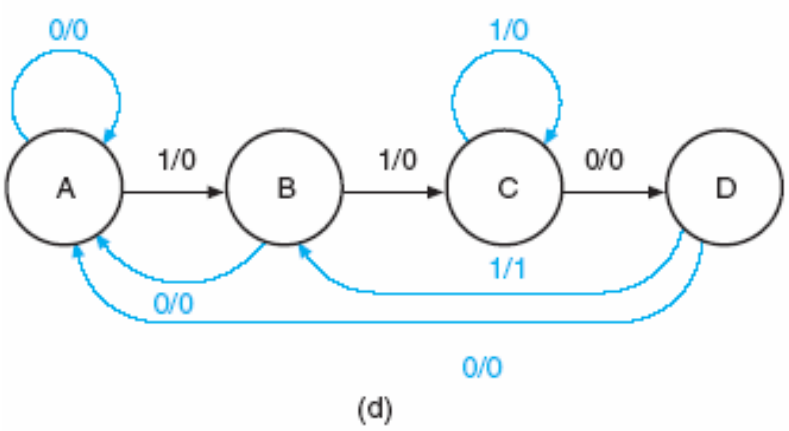
- The next bit of the sequence is 1 again. When this 1 occurs in state B, a state is needed to represent the occurrence of two 1's in a row on the input. Thus giving rise to state C.
- The next bit of the sequence is a 0. Thus a state is needed to represent two 1's followed by a 0. So, additional state D.



(c)

Sequential Circuit Design: Example …

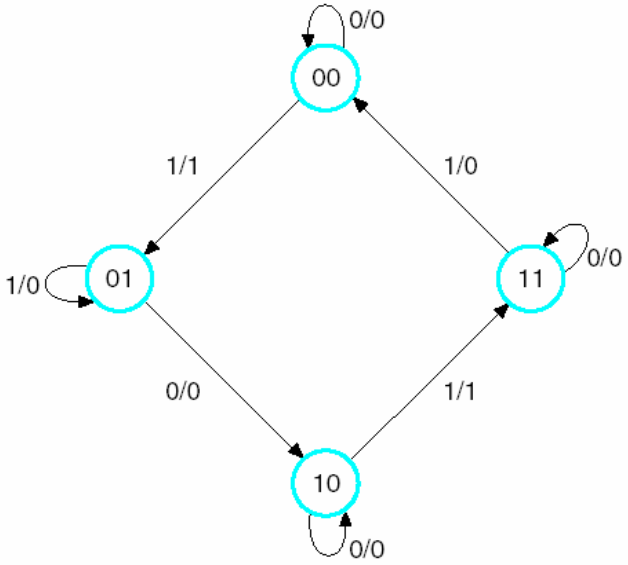
Since state D represents the occurrence of 110 as the previous three input bit values on X, the occurrence of a 1 in state D completes the sequence to be recognized. So the transition for the input value 1 from state D has output value of 1.



Complete state diagram.

Design with D flip-flops

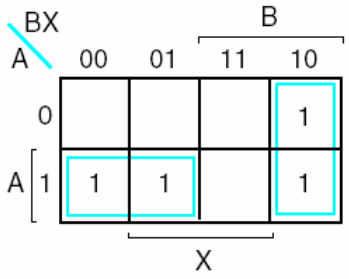
Remainder steps of sequential circuit design.



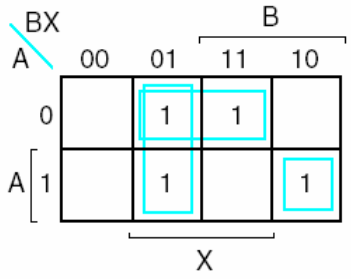
State Diagram for Design Example

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

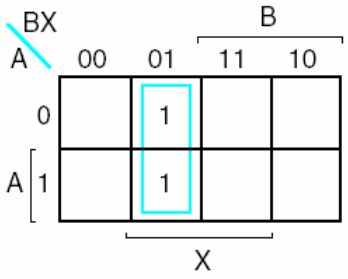
State Table for Design Example



$D_A = A\bar{B} + B\bar{X}$



$D_B = \bar{A}X + \bar{B}X + AB\bar{X}$



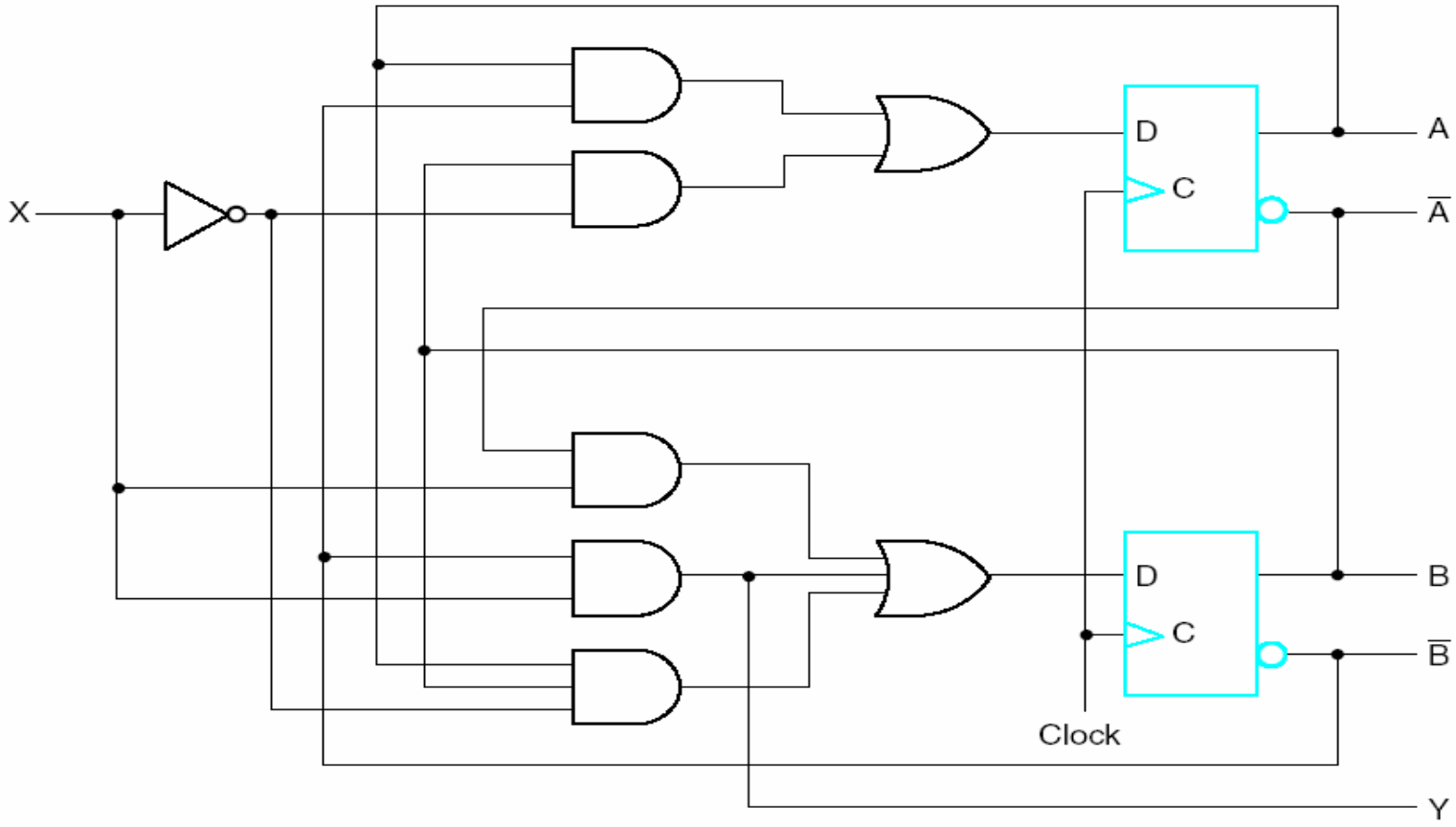
$Y = \bar{B}X$

Maps for Input Equations and Output *Y*

4 states, so 2 D FFs

Design with D flip-flops …

$D_A = AB'+BX'$, $D_B = A'X+B'X+ABX'$, $Y = B'X$



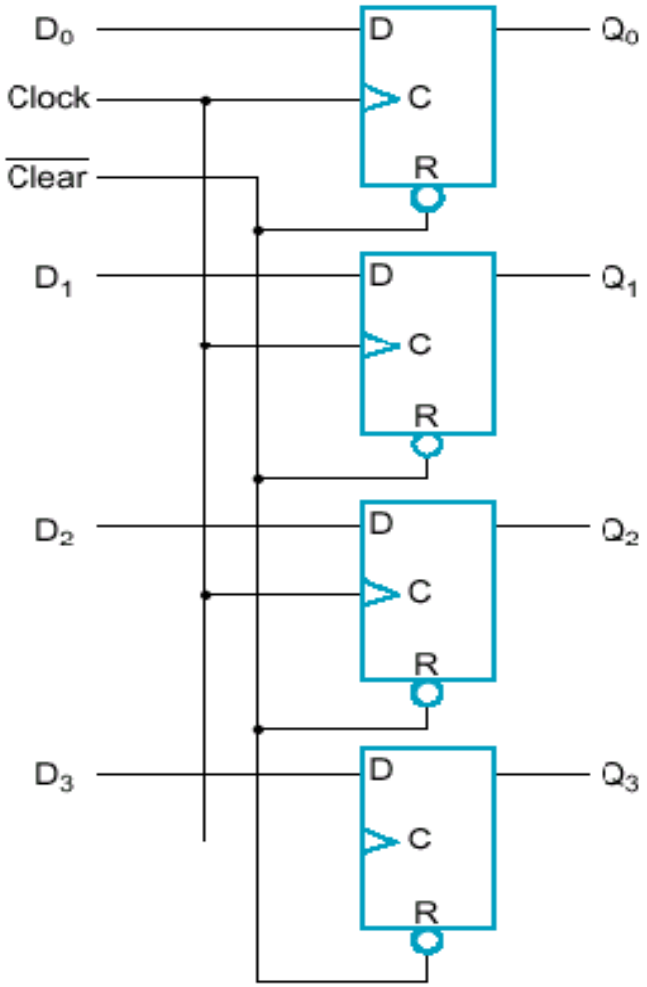
Logic Diagram for Sequential Circuit with *D* Flip-Flops

Similarly, the design with JK FFs can be done.

Registers and Counters: Basics

- Combinatorial and sequential circuits are brought together to result in sequential functional blocks, referred to as registers and counters.
- Registers are useful for storing and manipulating information.
- Counters are employed in circuits that sequence and control operations in a digital system.
- Register consists of a set of flip-flops, together with gates that implement their state transitions. Since each FF is capable of storing one bit information, an n-bit register constructed using n FFs can store n bits of binary information.
- Counter – A register that goes through a predetermined sequence of states upon the application of clock pulses.

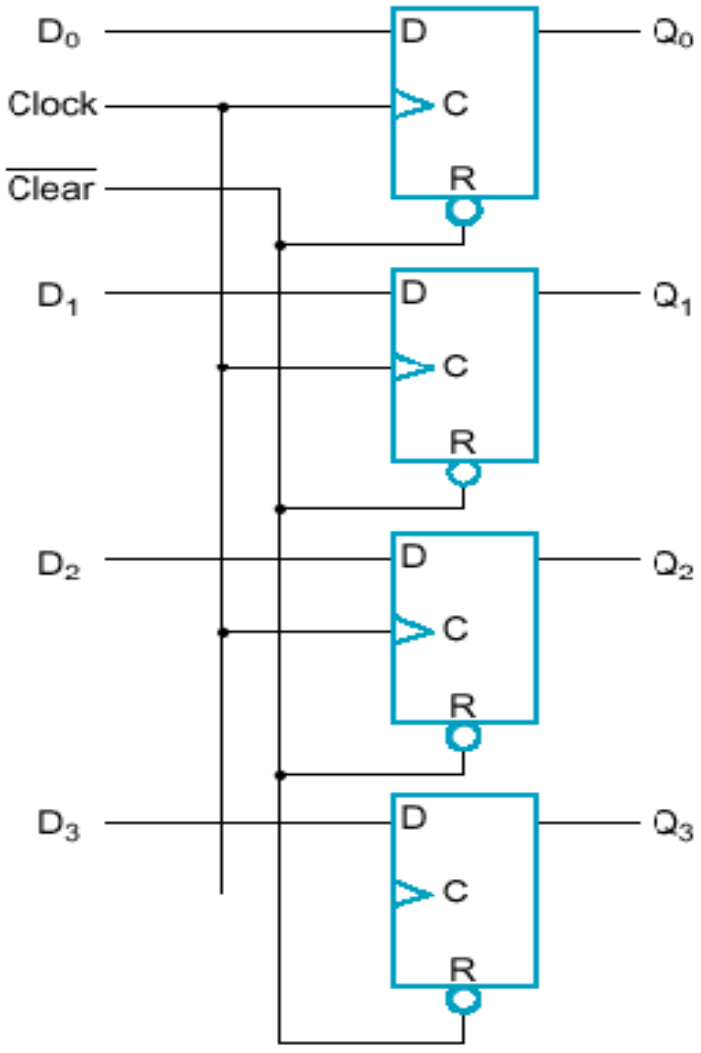
Registers in Simplest form



Register with 4 D-type FF

- Simplest register– A register that consists of only flip-flops without external gates.
- Common Clock input triggers all FFs on the rising edge of each pulse, and the binary data available at the four **D** inputs are transferred into the 4-bit register.
- The four Q outputs can be sampled to obtain the binary information stored in the register.

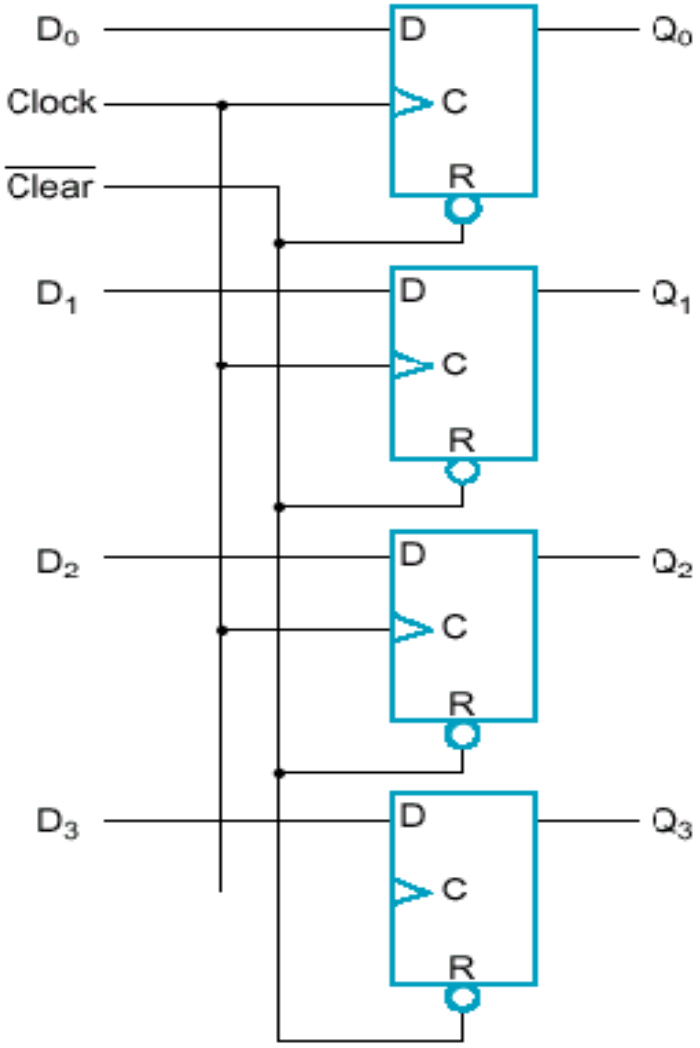
Registers in Simplest form ….



Register with 4 D-type FF

- The (Clear)' is useful for clearing the register to all 0's prior to its clocked operation. A 0 must be applied to it to cause all FF to reset asynchronously.
- Activation of the asynchronous R' inputs to FF during normal clocked operation can lead to circuit designs that are highly delay dependent, and thus malfunctioning. So, we maintain (Clear)' at logic 1 during normal clocked operation, allowing this to be 0 for system reset.

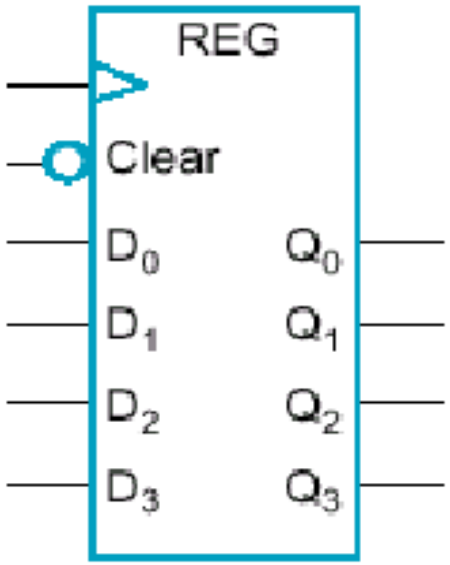
Registers in Simplest form ….



Register with 4 D-type FF

- Loading the register: Transfer of new information into a register.
- Parallel loading: Loading all the bits of the register simultaneously with a common clock pulse.
- Most digital systems have a master clock generator that supplies a continuous train of clock pulses, applied to all FF and registers.

Registers: Symbol



Register Symbol

- Symbol permits the use of register in a design hierarchy.
- Inputs are at the left and outputs are at the right.
- The clock input with a dynamic indicator represents positive edge-triggering.
- The clear symbol with a bubble indicate that logic 0 clears the FFs.
- If clear has to be written outside the symbol, then it should be written as (Clear)'.

Registers: Load and Clock

- The systems masters clock acts like a heart that supplies constant beat to all parts of the system.
- In order to preserve the contents of the registers this masters clock should not be given directly as the FF clock inputs.
- So, implementation with a load control input Load (L) combined with the clock as shown:

(c) Load control input

- The output of the OR gate is applied to the C inputs of the register FF.

C inputs = (Load)' + Clock

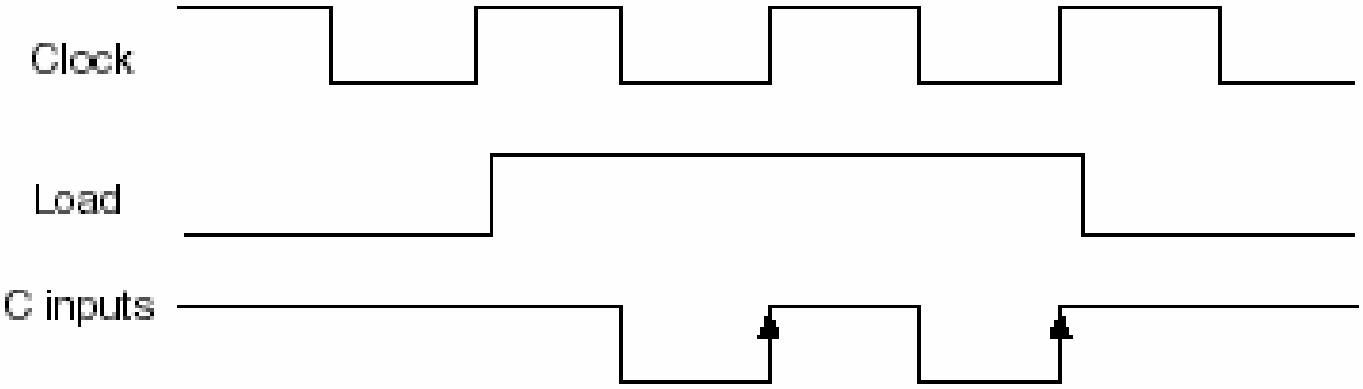
When Load = 1, C inputs = Clock, register is clocked normally, new info can be transferred into the register on the positive transitions of the clock. When Load = 0, C inputs = 1, no positive transitions so register contents remain unchanged.

Registers: Load and Clock ….

The output of the OR gate is applied to the C inputs of the register FF.

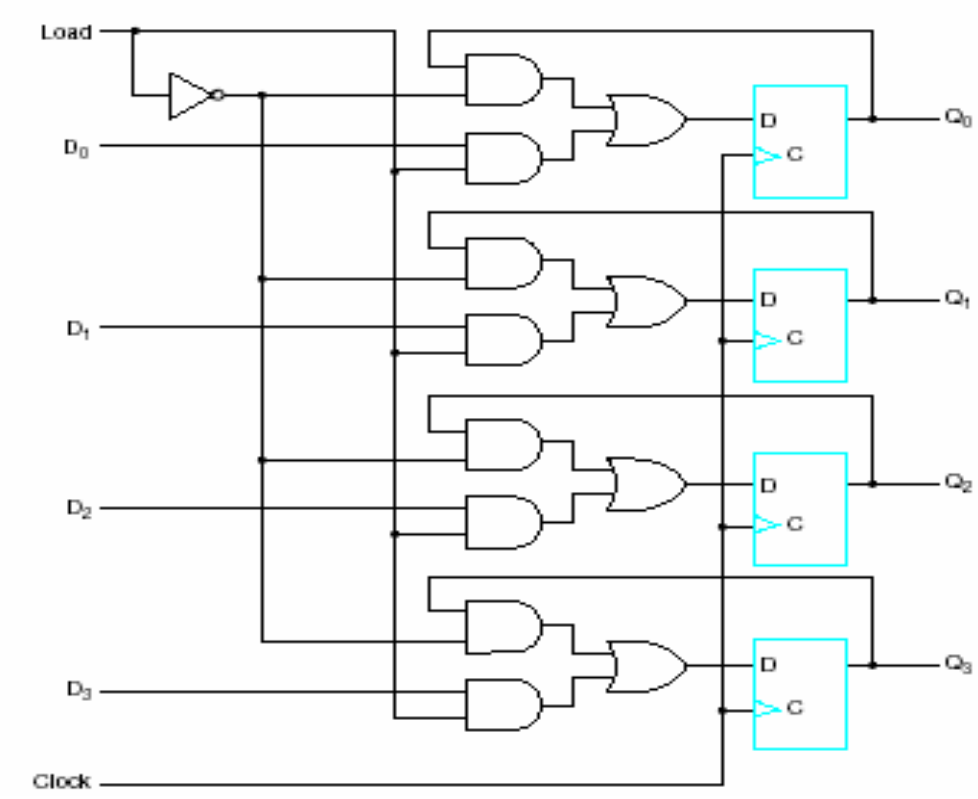
C inputs = (Load)' + Clock

When Load = 1, C inputs = Clock, register is clocked normally, new info can be transferred into the register on the positive transitions of the clock. When Load = 0, C inputs = 1, no positive transitions so register contents remain unchanged.



(d) Timing diagram

Register: with parallel load



With each clock pulse, the D input determines the next state of the output. To leave the output unchanged, it is necessary to make the D input equal to the present value of the output.

A 4-bit register with a control input Load directed through gates into the D inputs of the FF, instead of through clock gating on the C inputs. Load input determines the action to be taken.

When Load =1, data on the four inputs is transferred into the register with the next positive transition of the clock pulse.

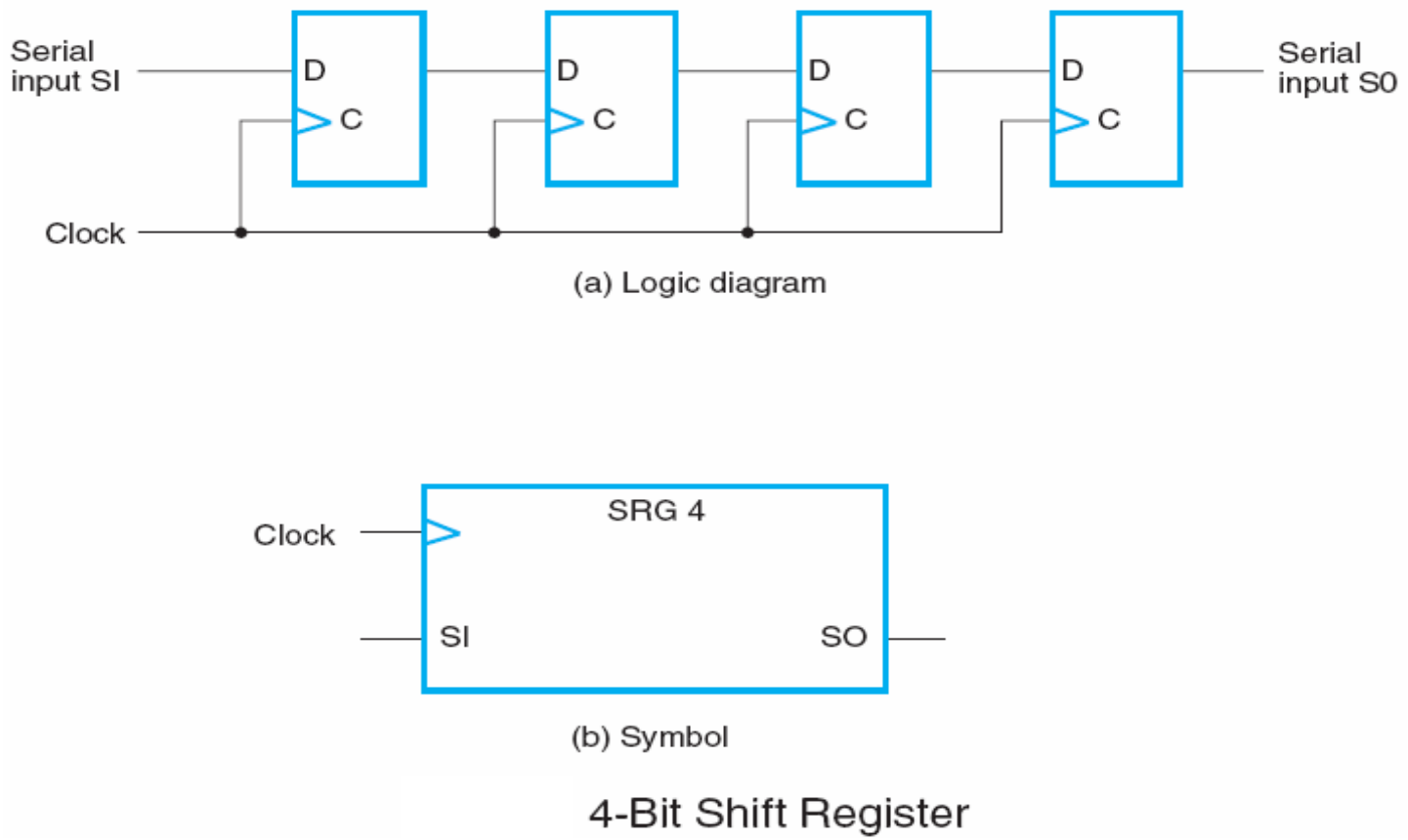
When Load =0, data inputs are blocked.

Feedback is necessary because the D FF does not have a “no change” input condition.

Shift Registers

- A register capable of shifting its stored bits laterally in one or both directions.
- Logical configuration consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop.
- All flip-flops receive a common clock pulse, which activates the shift from each stage to the next.
- Simplest possible shift register is the one that uses only FF.

Shift Registers …

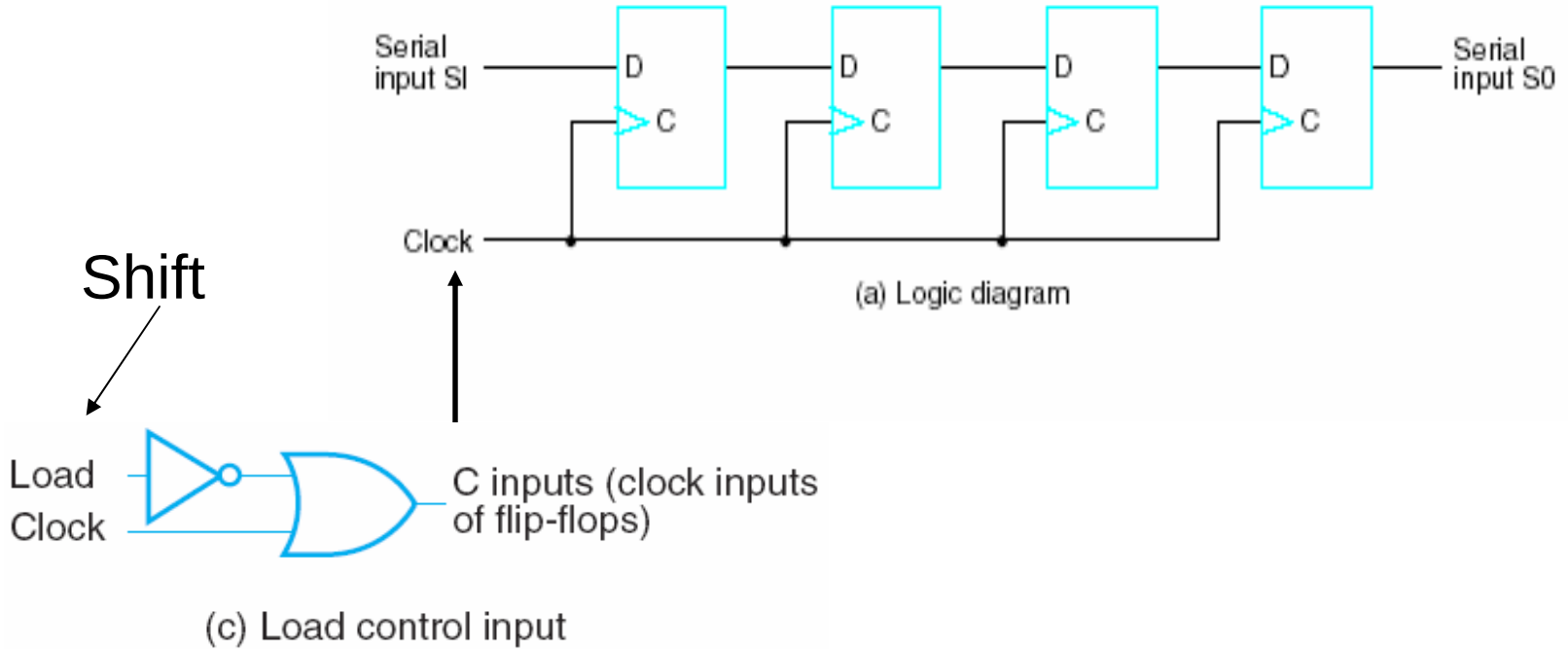


4-Bit Shift Register

- Output of a given FF is connected to the D input of the FF at its right. Common clock to all FF.
- Serial input SI is the input to the leftmost flip-flop during the shift.
- Serial output SO is taken from the output of the rightmost flip-flop.

Shift Registers …

Sometimes it is necessary to control the shift so that it occurs only with certain pulses, but not with others. In this case, shift can be controlled by connecting the clock through the logic shown, with Shift replacing Load.



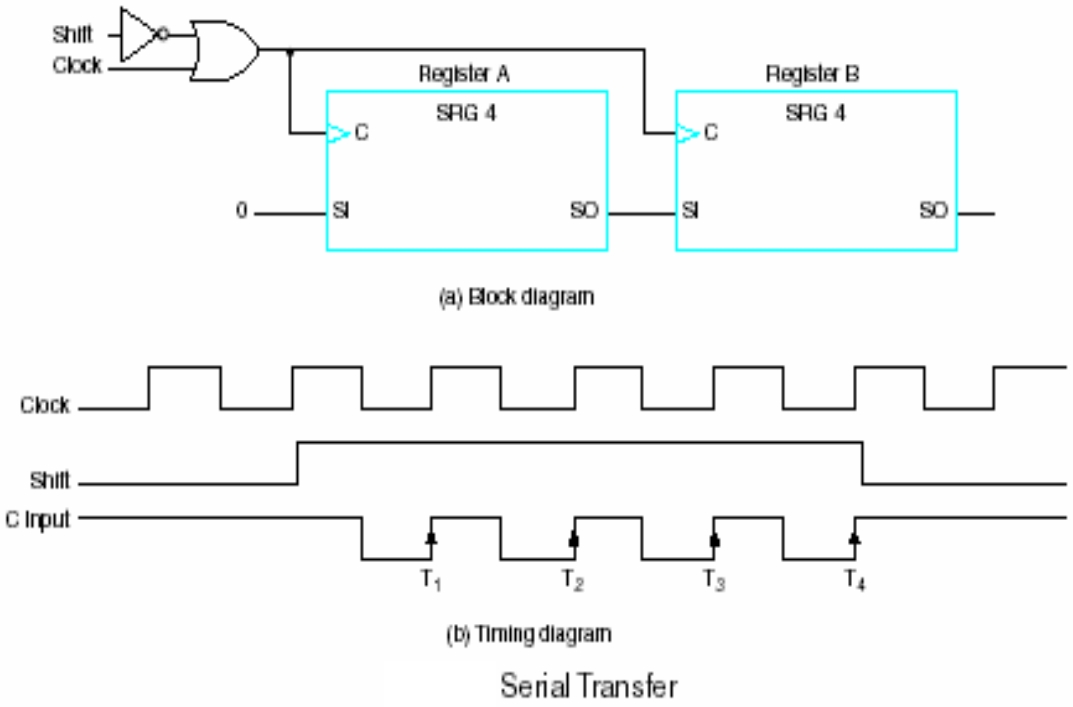
Shift Registers: Serial Transfer

- A digital system is said to operate in a serial mode when information in the system is transferred or manipulated one bit at a time.
- Transfer one bit at a time– Shift the bits out of one register and into a second register.
- Contrast with Parallel: Transfer all bits at the same time.
- Serial transfer of information from register A to B is done with shift registers.

Parallel Vs Serial

- Operations in digital computers are generally parallel, due to faster speed attainable.
- Serial operations are slower and require lesser hardware.

Shift Registers: Serial Transfer …



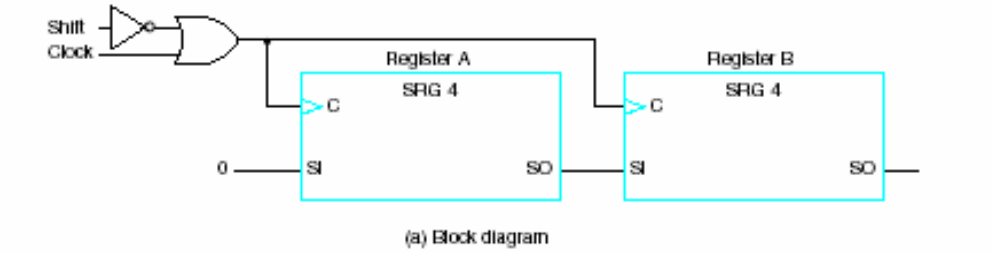
Serial Transfer

- Serial output of A connected to serial input of B.
- SI of A receives 0's while its data are transferred to B.
- Also possible for A to receive other binary information.
- Initial content of B is shifted out through its SO and it is lost, unless transferred back to A.
- By connecting A's serial output to its serial input, info can be circulated back into the register.

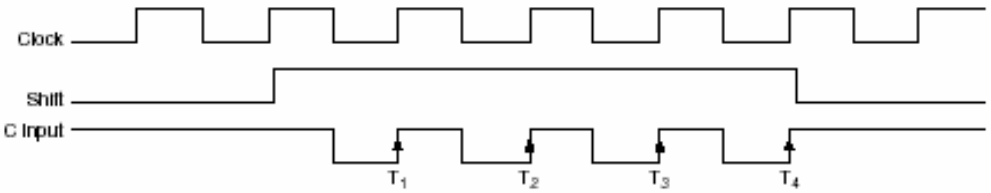
Shift determines when and how many times registers are shifted.

Clock pulses pass to the shift register clock inputs only when Shift = 1.

Shift Registers: Serial Transfer Example



(a) Block diagram



(b) Timing diagram

Serial Transfer

Timing pulse	Shift Register A				Shift Register B			
Initial value	1	0	1	1	0	0	1	0
After T_1	0	1	0	1	1	0	0	1
After T_2	0	0	1	0	1	1	0	0
After T_3	0	0	0	1	0	1	1	0
After T_4	0	0	0	0	1	0	1	1

Example of Serial Transfer

Each shift register has four stages.

Logic that supervises the transfer must be designed to enable shift registers through Shift for a fixed time of 4 clock pulses. 4 pulses find Shift in the active state, so that output of the logic connected to the clock inputs of the registers produces 4 pulses T_1 – T_4.

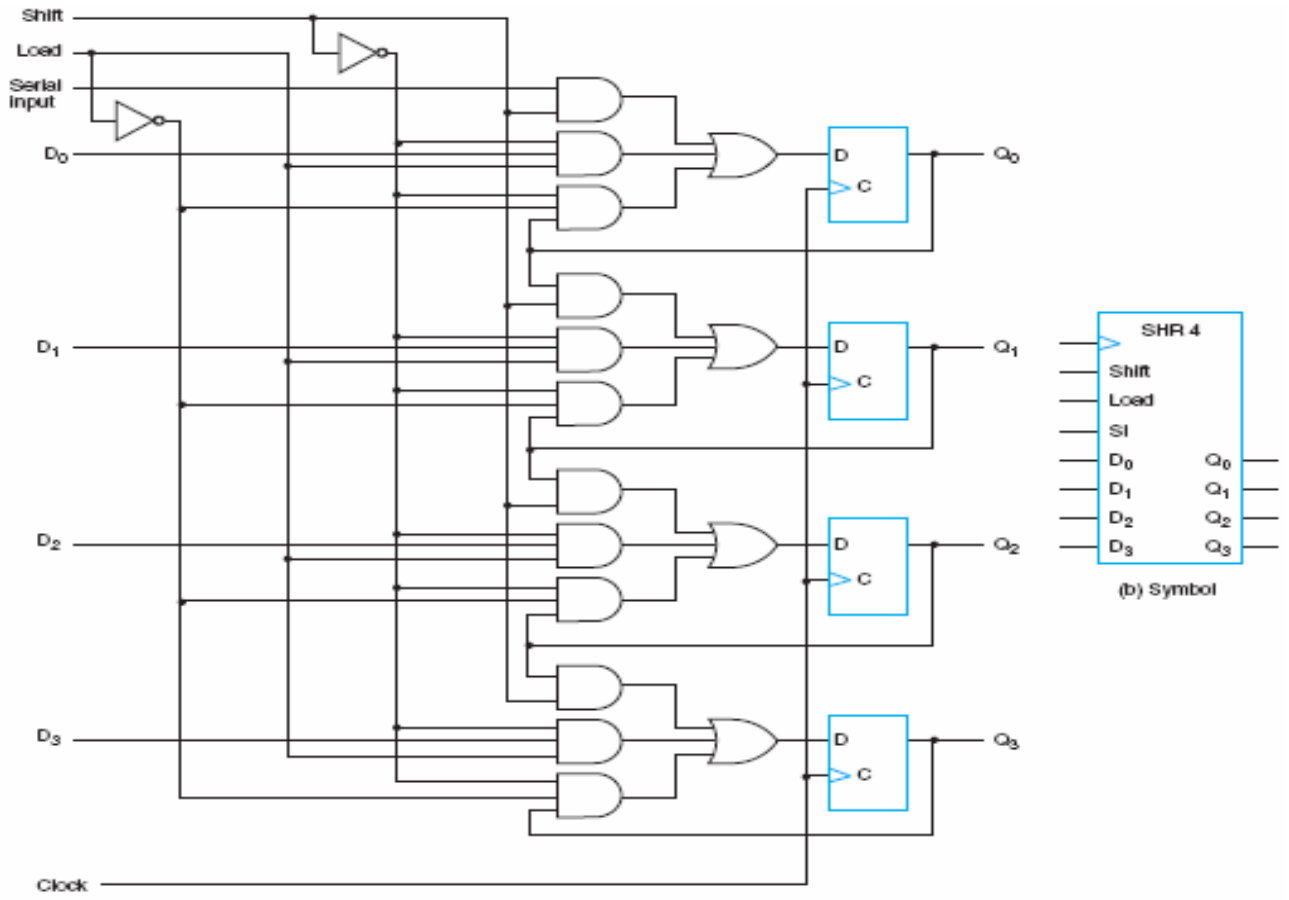
Each positive transition of these pulses causes a shift in both registers. After that Shift changes back to 0 and shift registers are disabled.

Example: A (1011), B(0010), SI of A is logic 0.

Shift Register: with Parallel Load

- If a parallel load capability is also added to shift register, then data entered in parallel can be taken out in serial fashion by shifting out the data in the register.
- Shift register with accessible flip-flop outputs and parallel load can be used for converting incoming parallel data to outgoing serial data and vice versa.
- Shift registers are often used to interface digital systems that are situated remotely from each other.

Shift Register: with Parallel Load …

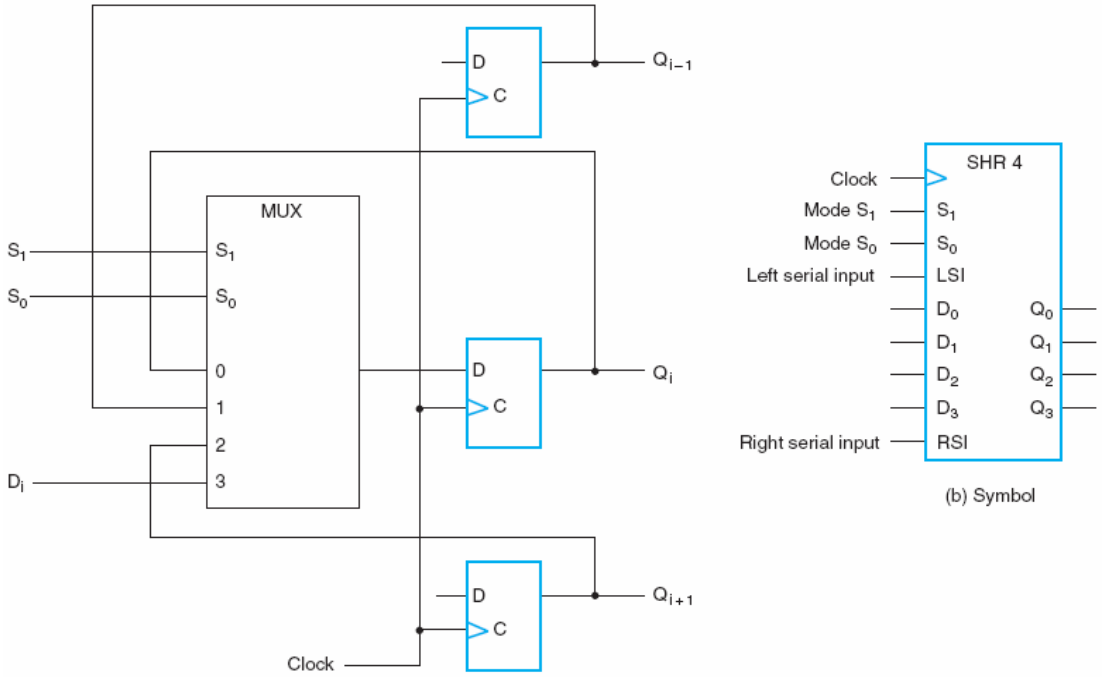


Shift	Load	Operation
0	0	No change
0	1	Load parallel data
1	×	Shift down from Q_0 to Q_3

Function Table for the Register

Shift Register: Bi-directional with Parallel Load

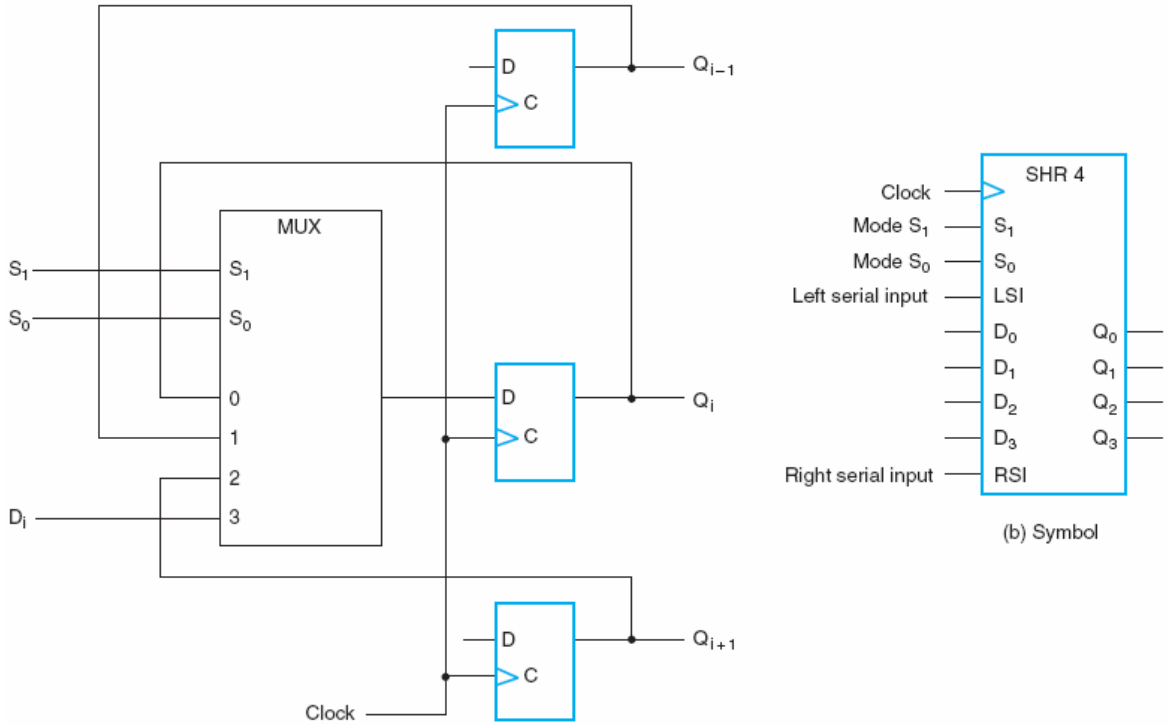
- Unidirectional shift register – A register capable of shifting in one direction.
- Bi-directional shift register – A register capable of shifting in both directions.



(a) Logic diagram of one typical stage

Bidirectional Shift Register with Parallel Load

Shift Register: Bi-directional …



(a) Logic diagram of one typical stage

(b) Symbol

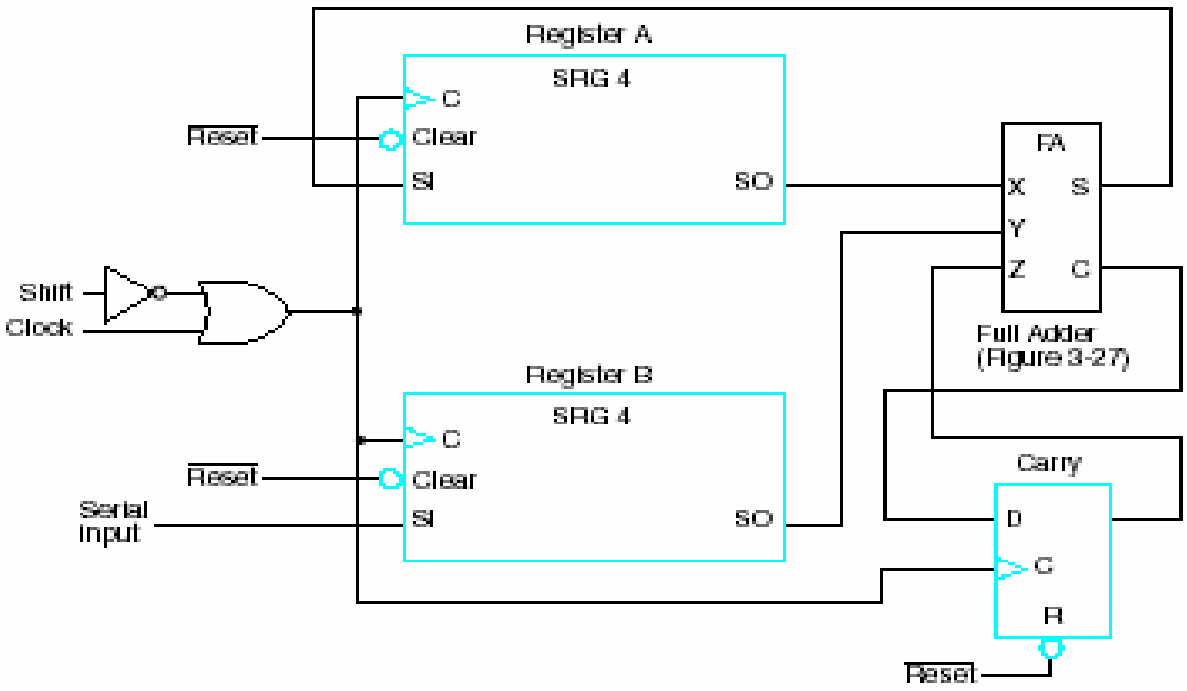
Mode control		Register Operation
S_1	S_0	
0	0	No change
0	1	Shift down
1	0	Shift up
1	1	Parallel load

Function Table for the Register

Applications of Shift Registers

- Use of shift registers are common in digital systems.
- Typical uses include, Registering and Shifting.
- Registering: The most common application for a shift register is to hold a series of bits or an operand for later processing.
- Shifting operations are frequently used for many type of operations including:
  - Shifting
  - Multiplication
  - Division
  - Scaling
  - Serial to Parallel Conversion
  - Parallel to Serial Conversion

Shift Registers: Serial Addition



- Register A holds the augend, register B holds the addend.
- The carry flip-flop is reset to 0.
- Serial outputs of A and B provide a pair of significant bits for the full adder at X and Y.
- Output of the carry flip-flop provides the carry input at Z.

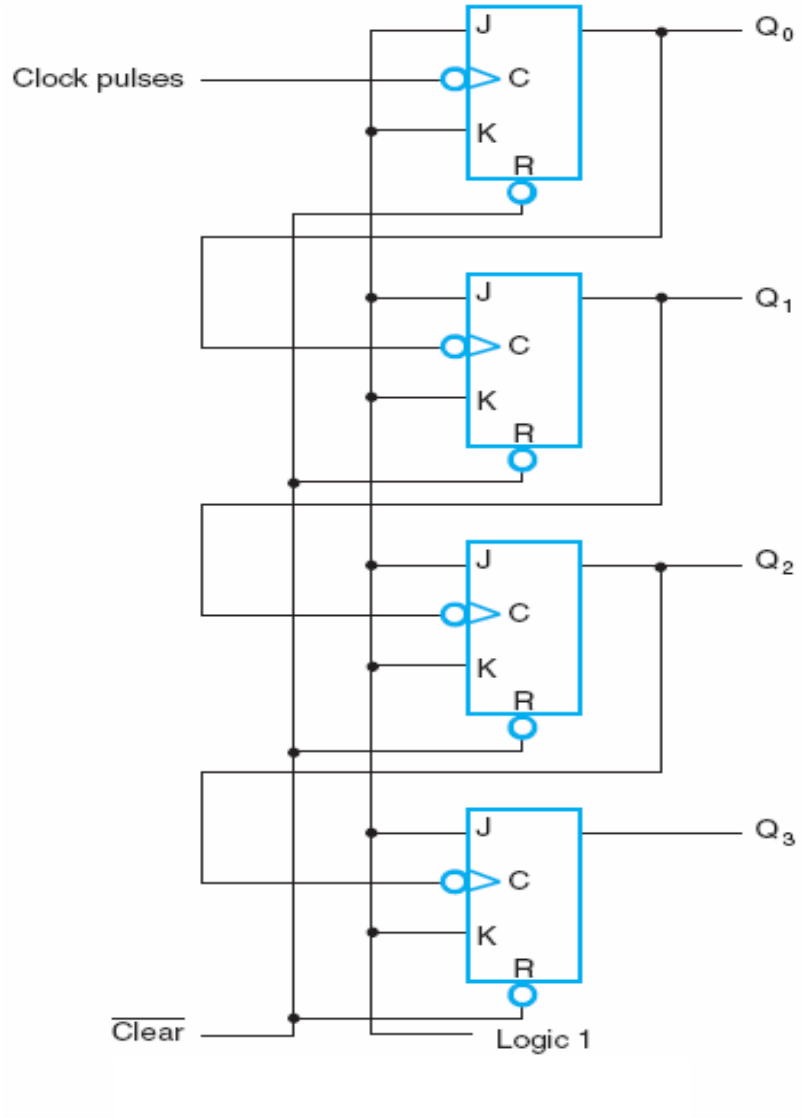
Counters: Basics

- A counter is a sequential machine that produces a specified count sequence. The count changes whenever the input clock is asserted.
- There is a great variety of counters based on its construction.
  - Clock: Synchronous or Asynchronous
  - Clock Trigger: Positive edged or Negative edged
  - Carry: Ripple or Parallel
  - Counts: Binary, Decade, Arbitrary
  - Count Direction: Up, Down, or Up/Down
  - Flip-flops: JK or T or D
- A Binary follows the binary number sequence. An n-bit binary counter consists of n flip-flops and can count in binary from 0 through $2^n - 1$.

Ripple Counter

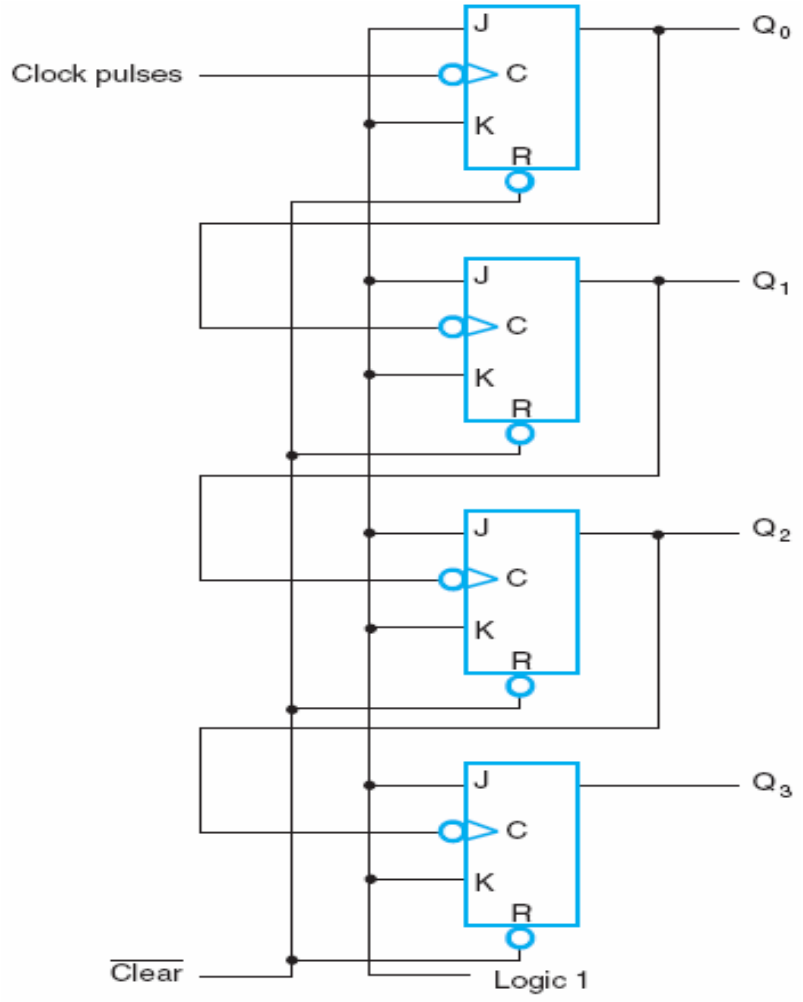
- Ripple Counter: Flip-flop output transition serves as a source for triggering other flip-flops, i.e. C input of some or all of the FFs is triggered not by common clock pulse, but by transition of other FFs, leading to Asynchronous behavior.
- Advantage: Simple hardware.
- Disadvantage: Asynchronous circuits, with added logic, can be unreliable and delay dependent.
- Counter constructed with FF capable of complementing their contents.
- Output of each FF is connected to the C input of the next ff in sequence.
- The FF holding the LSB receives incoming clock pulses.
- JK inputs of all FF connected to a permanent Logic 1.

Ripple Counter …



J and K's being connected to logic 1 and negative-edge triggering make each FF complement its value if the signal on its C input goes through a negative transition. Negative transition occurs when the output of the previous ff to which C is connected goes from 1 to 0. A 0-level signal on (Clear)' driving the R' inputs clears the register to 0 asynchronously.

Ripple Counter …



4-Bit Ripple Counter

Upward Counting Sequence				Downward Counting Sequence			
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	0
0	0	1	0	1	1	0	1
0	0	1	1	1	1	0	0
0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0

Counting Sequence of Binary Counter

Design of Synchronous binary Counters

Design procedure for a synchronous counter is the same as with any other synchronous sequential circuit. Counter may operate without external input, except for the clock pulses. Counter output taken from FF outputs without any additional outputs from gates. In the absence of inputs, counter state table will consist of columns for the present state and next state only.

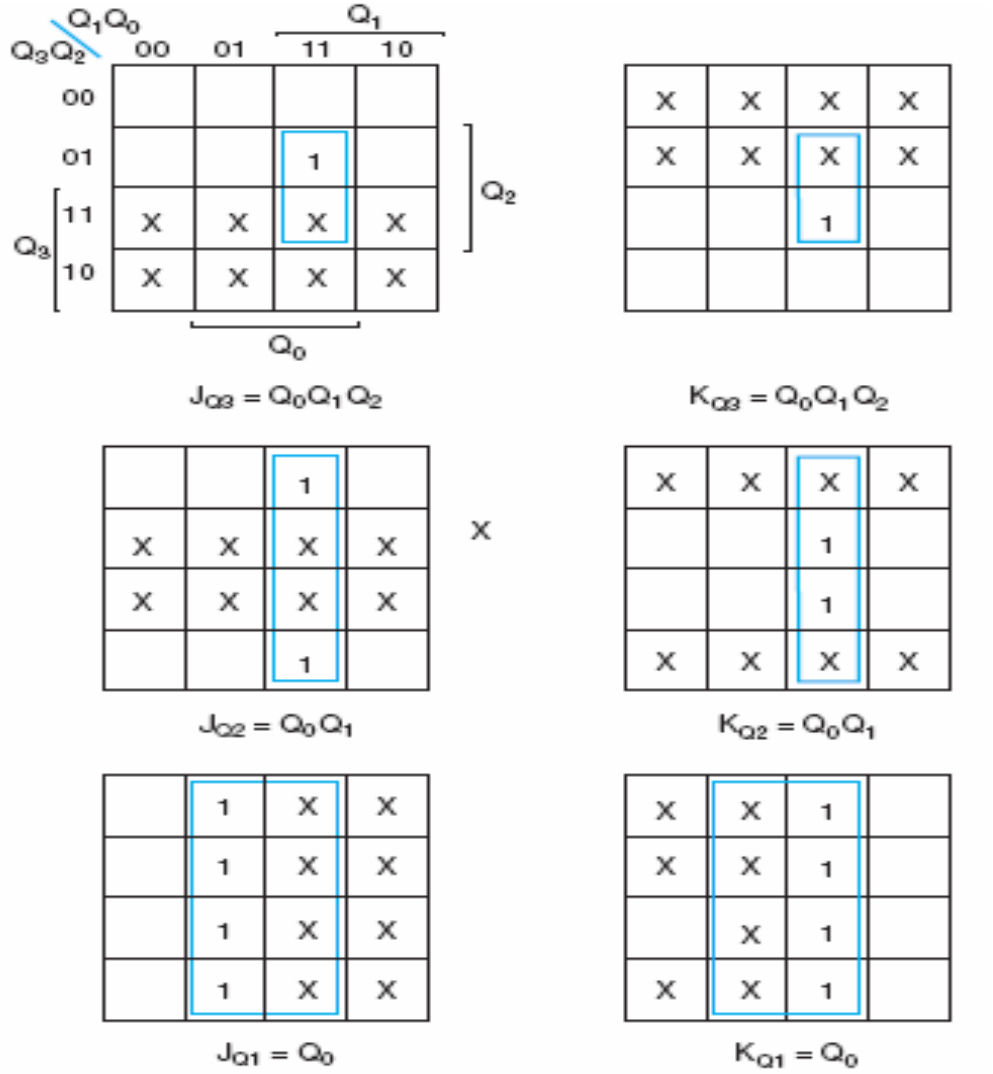
Outputs implicitly represented by the present state column.

Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J_{Q3}	K_{Q3}	J_{Q2}	K_{Q2}	J_{Q1}	K_{Q1}	J_{Q0}	K_{Q0}
0	0	0	0	0	0	0	1	0	×	0	×	0	×	1	×
0	0	0	1	0	0	1	0	0	×	0	×	1	×	×	1
0	0	1	0	0	0	1	1	0	×	0	×	×	0	1	×
0	0	1	1	0	1	0	0	0	×	1	×	×	1	×	1
0	1	0	0	0	1	0	1	0	×	×	0	0	×	1	×
0	1	0	1	0	1	1	0	0	×	×	0	1	×	×	1
0	1	1	0	0	1	1	1	0	×	×	0	×	0	1	×
0	1	1	1	1	0	0	0	1	×	×	1	×	1	×	1
1	0	0	0	1	0	0	1	×	0	0	×	0	×	1	×
1	0	0	1	1	0	1	0	×	0	0	×	1	×	×	1
1	0	1	0	1	0	1	1	×	0	0	×	×	0	1	×
1	0	1	1	1	1	0	0	×	0	1	×	×	1	×	1
1	1	0	0	1	1	0	1	×	0	×	0	0	×	1	×
1	1	0	1	1	1	1	0	×	0	×	0	1	×	×	1
1	1	1	0	1	1	1	1	×	0	×	0	×	0	1	×
1	1	1	1	0	0	0	0	×	1	×	1	×	1	×	1

State Table and Flip-Flop Inputs for Binary Counter

Next state = present state + one

Design of Synchronous binary Counters …



Maps for Input Equations of a Binary Counter

Introducing a enable variable EN the FF input equations can be written as:

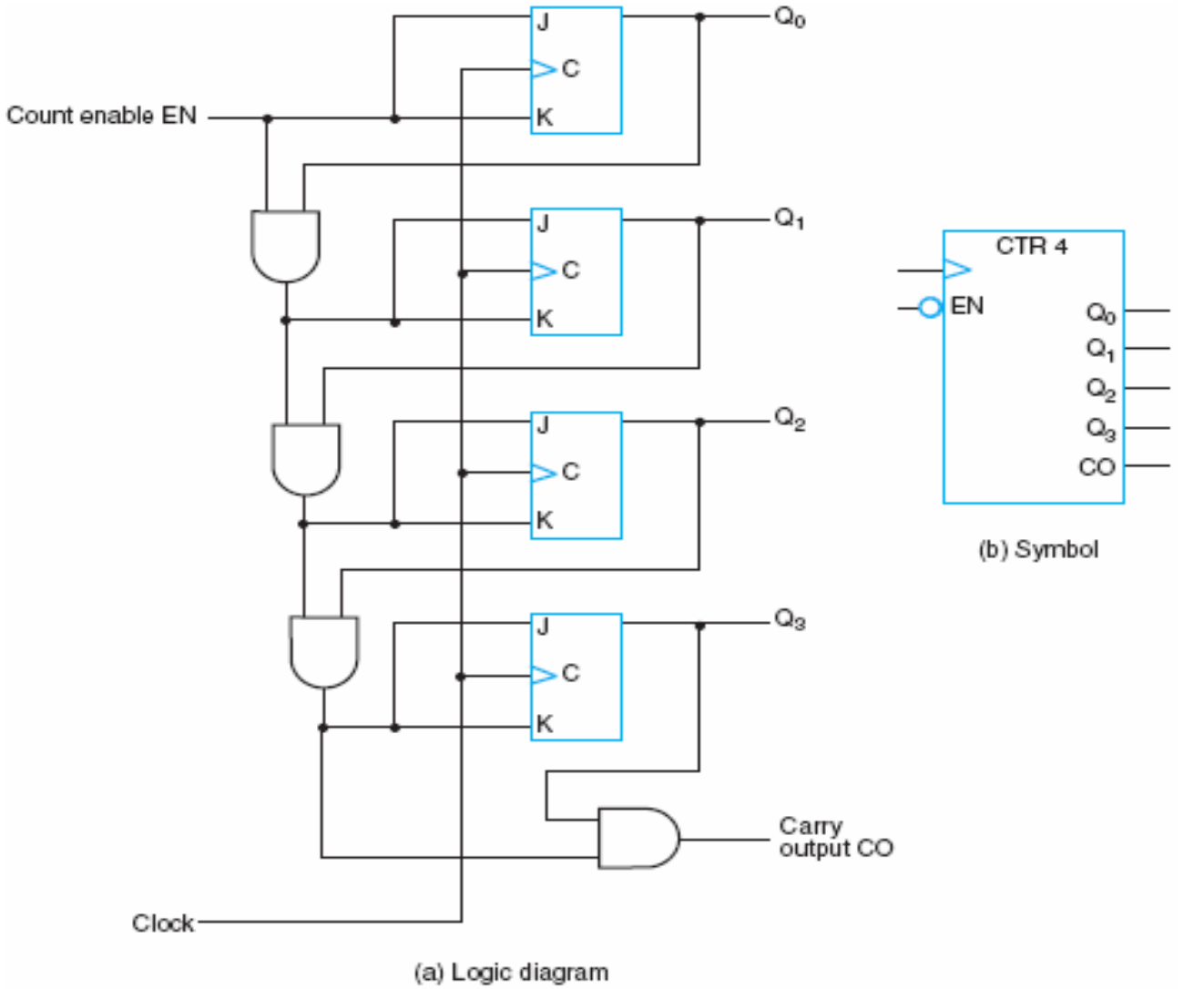
$J_{Q0} = K_{Q0} = EN$

$J_{Q1} = K_{Q1} = Q_0.EN$

$J_{Q2} = K_{Q2} = Q_0.Q_1.EN$

$J_{Q3} = K_{Q3} = Q_0.Q_1.Q_2.EN$

Design of Synchronous binary Counters …

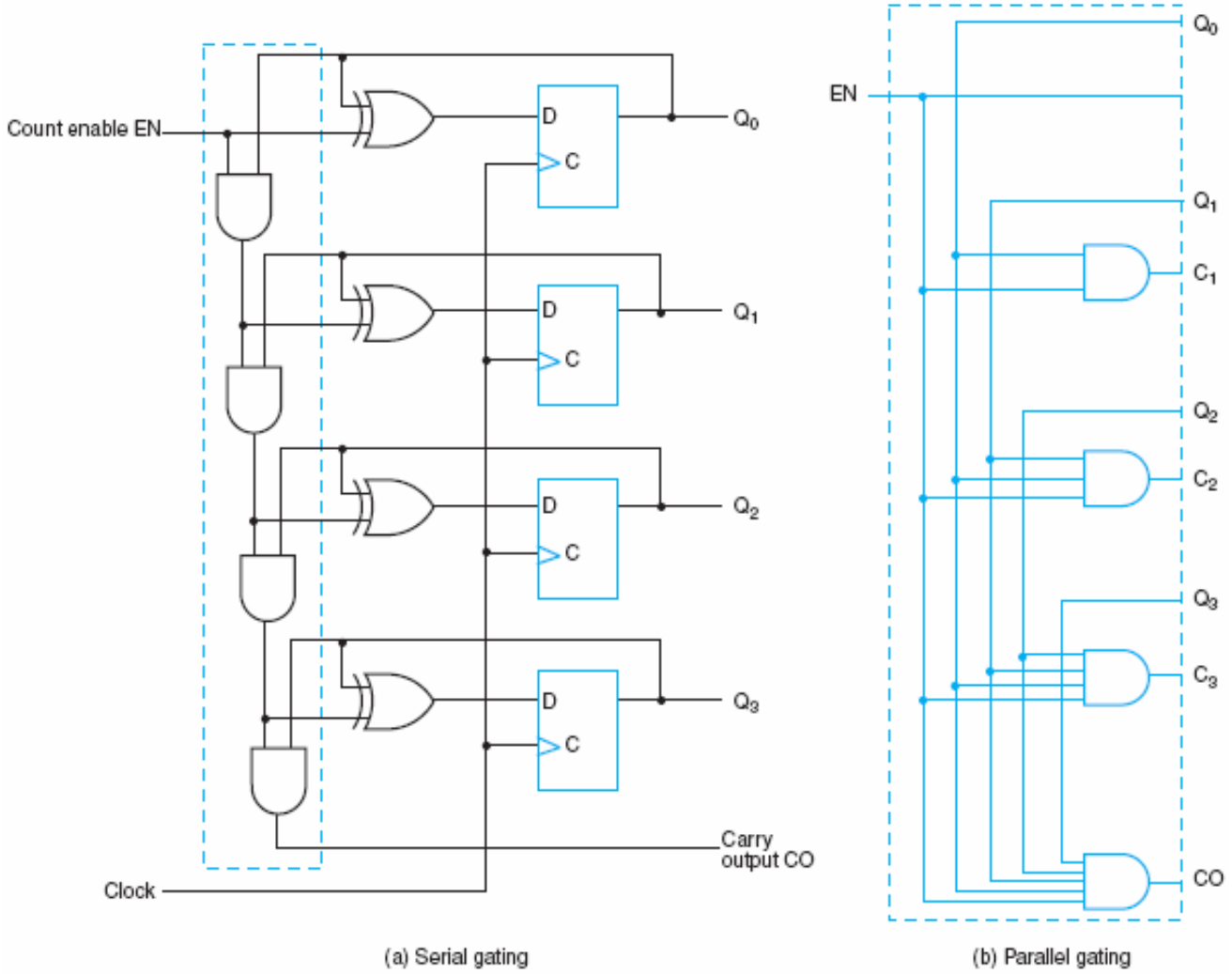


(b) Symbol

(a) Logic diagram

4-Bit Synchronous Binary Counter

Design of binary Counters (Similarly with D FF)



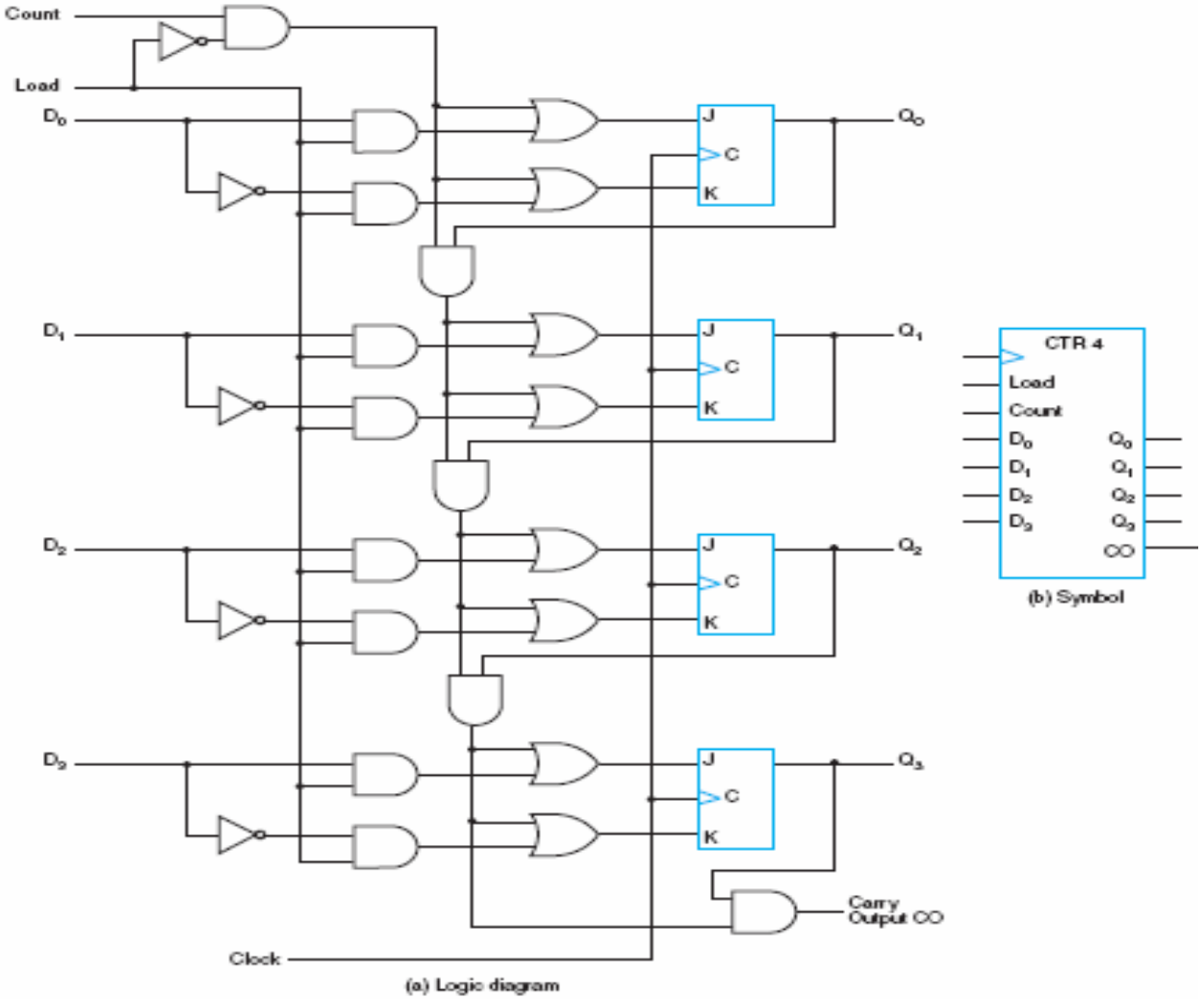
4-Bit Binary Counter with *D* Flip-Flops

Serial gating: A chain of 2-input AND gates used to provide information to a FF from a prior stage. (Analogous to ripple carry adder)

Parallel gating: (Analogous to carry lookahead adder)

Advantage: Delay reduction

Synchronous Binary Counter with Parallel Load



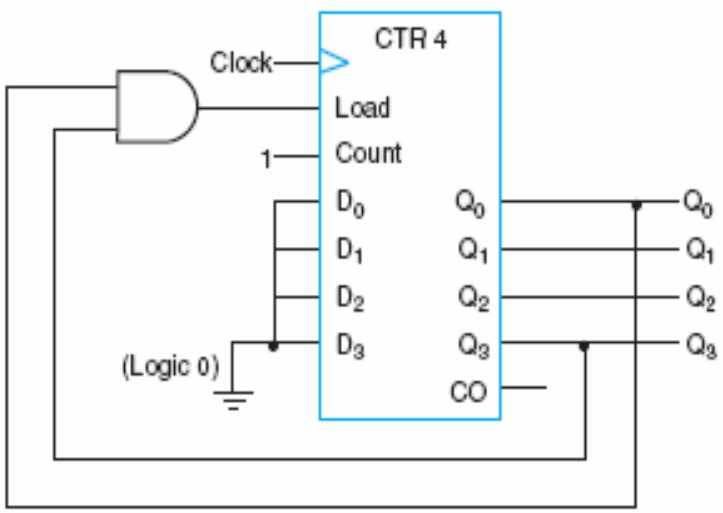
4-Bit Binary Counter with Parallel Load

Load=1, data transfer from input to FFs

Load=0 and Count = 1, works as a binary counter.

BCD Counter

A divide-by-N counter also known as a modulo-N counter is a counter that goes through a repeated sequence of N states. Designed using the same steps as in case of any sequential circuit.

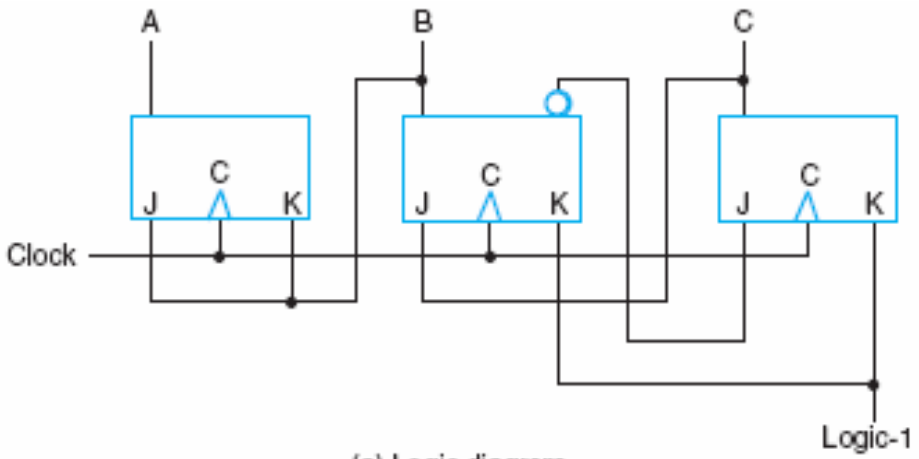


BCD Counter

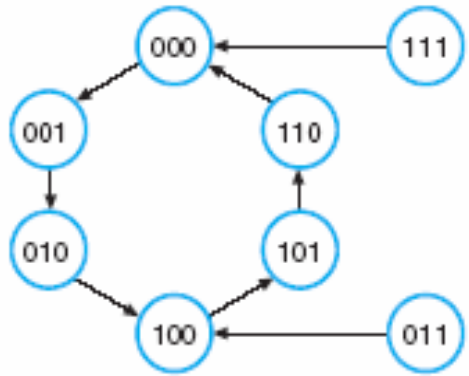
Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	Y	T_{Q8}	T_{Q4}	T_{Q2}	T_{Q1}
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

State Table and Flip-Flop Inputs for BCD Counter

Counters for Arbitrary Sequence



(a) Logic diagram



(b) State diagram

Counter with Arbitrary Count

Present State			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	×	0	×	1	×
0	0	1	0	1	0	0	×	1	×	×	1
0	1	0	1	0	0	1	×	×	1	0	×
1	0	0	1	0	1	×	0	0	×	1	×
1	0	1	1	1	0	×	0	1	×	×	1
1	1	0	0	0	0	×	1	×	1	0	×

State Table and Flip-Flop Inputs for Counter

The simplified FF input equations are:

$J_A = B, K_A = B$

$J_B = C, K_B = 1$

$J_C = B', K_C = 1$