Chapter 6

Registers and Counters

Outline

- $\boxed{\text{Registers}}$
- Shift Registers
- Ripple Counters
- Synchronous Counters
- Other Counters

Registers

- A collection of flip-flops taken as an entity.
- Function: Hold information within a digital system so that it is available to the logic elements during the computing process.
- Each combination of stored information is known as the state or content of the register.
- Shift register: Registers that are capable of moving information upon the occurrence of a clock-signal.
  - Unidirectional
  - bidirectional

Registers

- Two basic ways in which information can be entered/outputted
  - Parallel: All 0/1 symbols handled simultaneously. Require as many lines as symbols being transferred.
  - Serial: Involves the symbol-by-symbol availability of information in a time sequence.
- Four possible ways registers can transfer information:
  - Serial-in/serial-out
  - Serial-in/parallel-out
  - Parallel-in/parallel-out
  - Parallel-in/serial-out

Serial-in, Serial-out, Unidirectional Shift Register

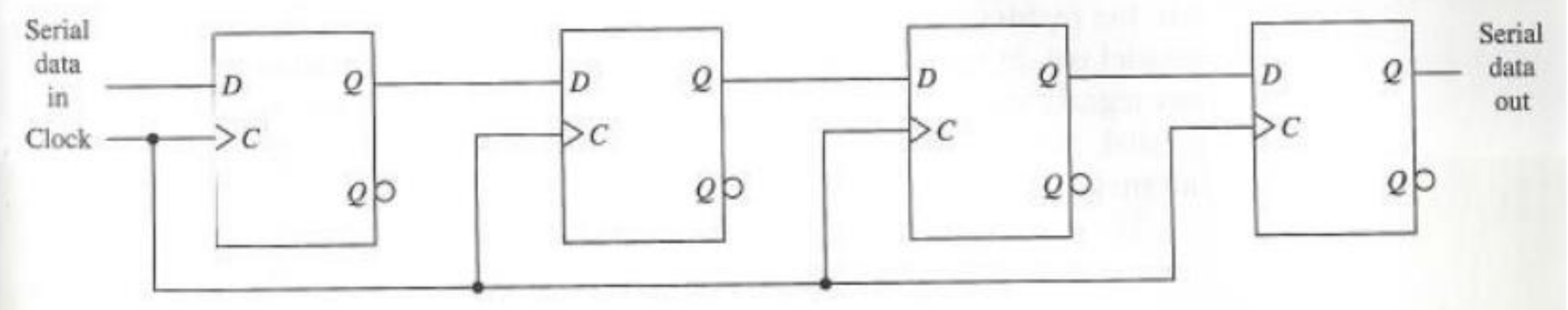


Figure 6.26 Serial-in, serial-out unidirectional shift register.

Serial-in, Parallel-out Unidirectional Shift Register

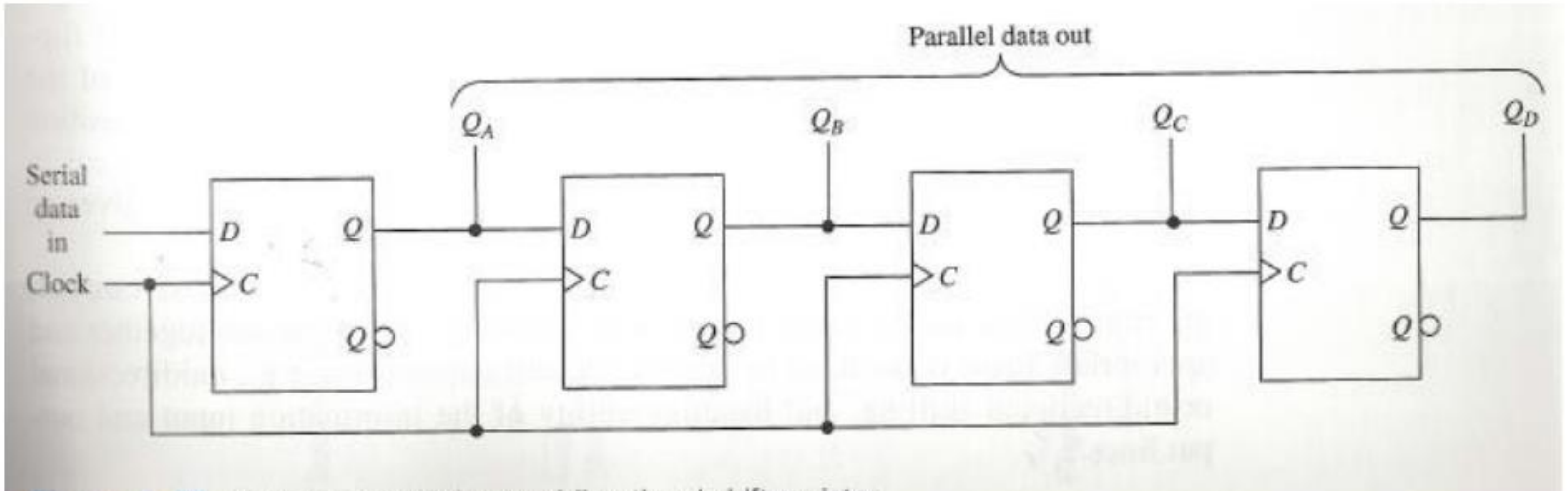


Figure 6.27 Serial-in, parallel-out unidirectional shift register.

Parallel-in, Parallel-out Unidirectional Shift Register

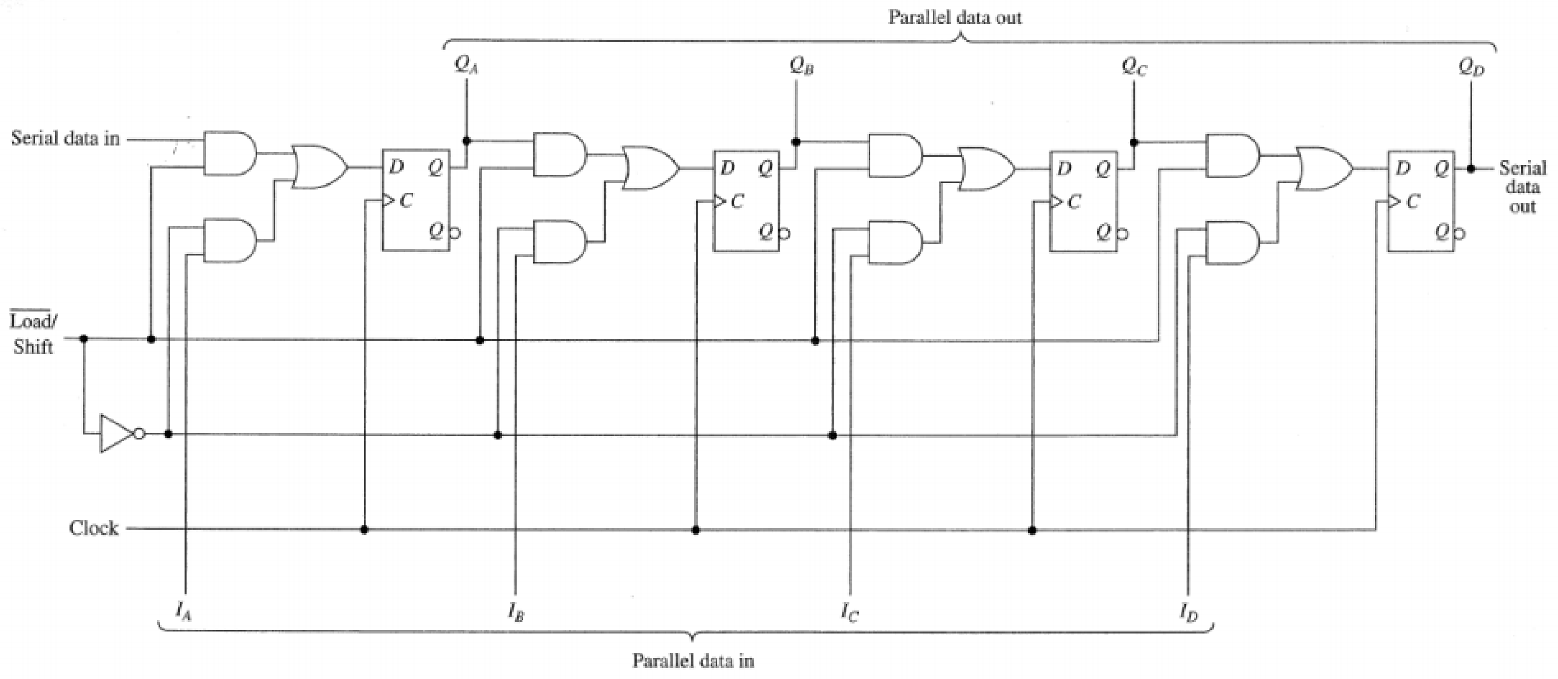


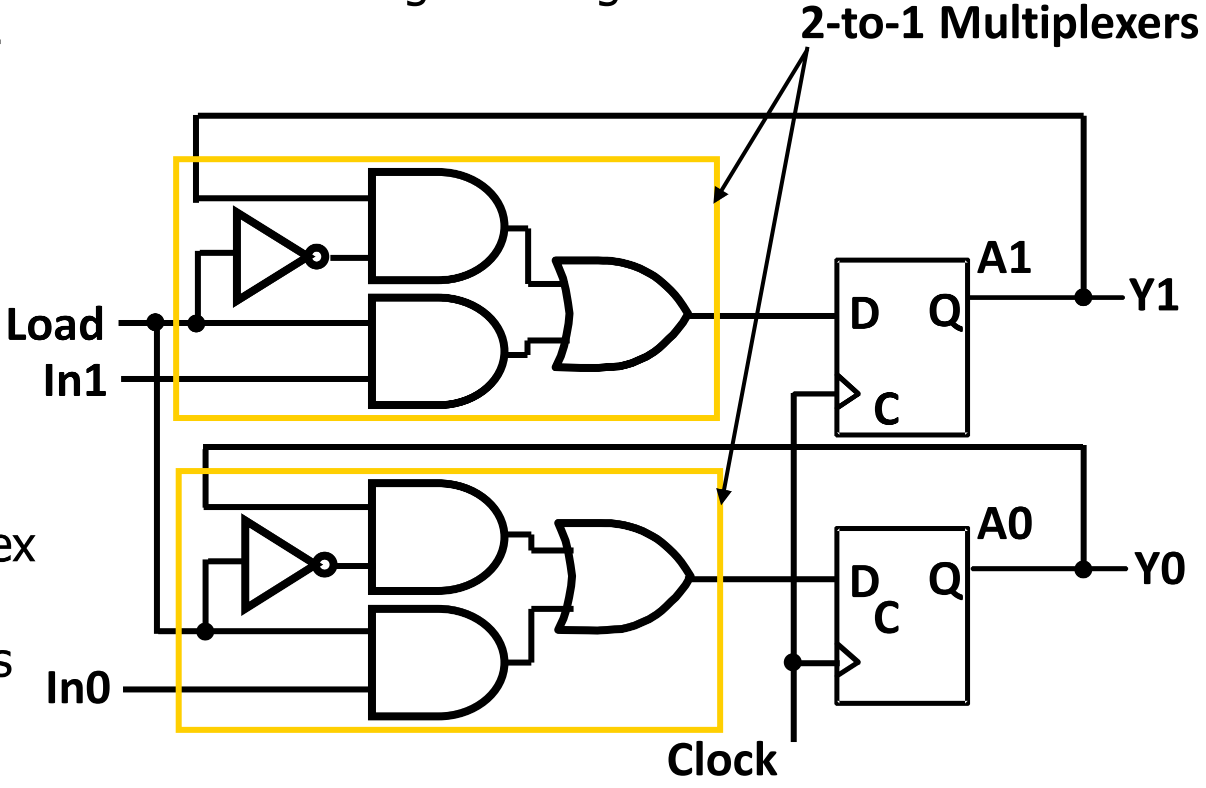
Figure 6.28 Parallel-in unidirectional shift register.

Register Storage

- Expectations:
  - A register can store information for multiple clock cycles
  - To “store” or “load” information should be controlled by a signal
- Reality:
  - A D flip-flop register loads information on every clock cycle
- Realizing expectations:
  - Use a signal to block the clock to the register,
  - Use a signal to control feedback of the output of the register back to its inputs, or
  - Use other SR or JK flip-flops, that for (0,0) applied, store their state
- Load is a frequent name for the signal that controls register storage and loading
  - Load = 1: Load the values on the data inputs
  - Load = 0: Store the values in the register

Registers with Load-Controlled Feedback

- A reliable way to selectively load a register:
  - Run the clock continuously, and
  - Selectively use a load control to change the register contents.
- Example: 2-bit register with Load Control:
- For Load = 0, loads register contents (hold current values)
- For Load = 1, loads input values (load new values)
- Hardware more complex than clock gating, but free of timing problems



Register with Parallel Load

- When *Load* = 1, all the bits of data inputs are transferred into the registers
  - Parallel loaded
- When *Load* =0, the outputs of the flip-flops are connected to their respective inputs
  - Keep no change

control data loading at the input side

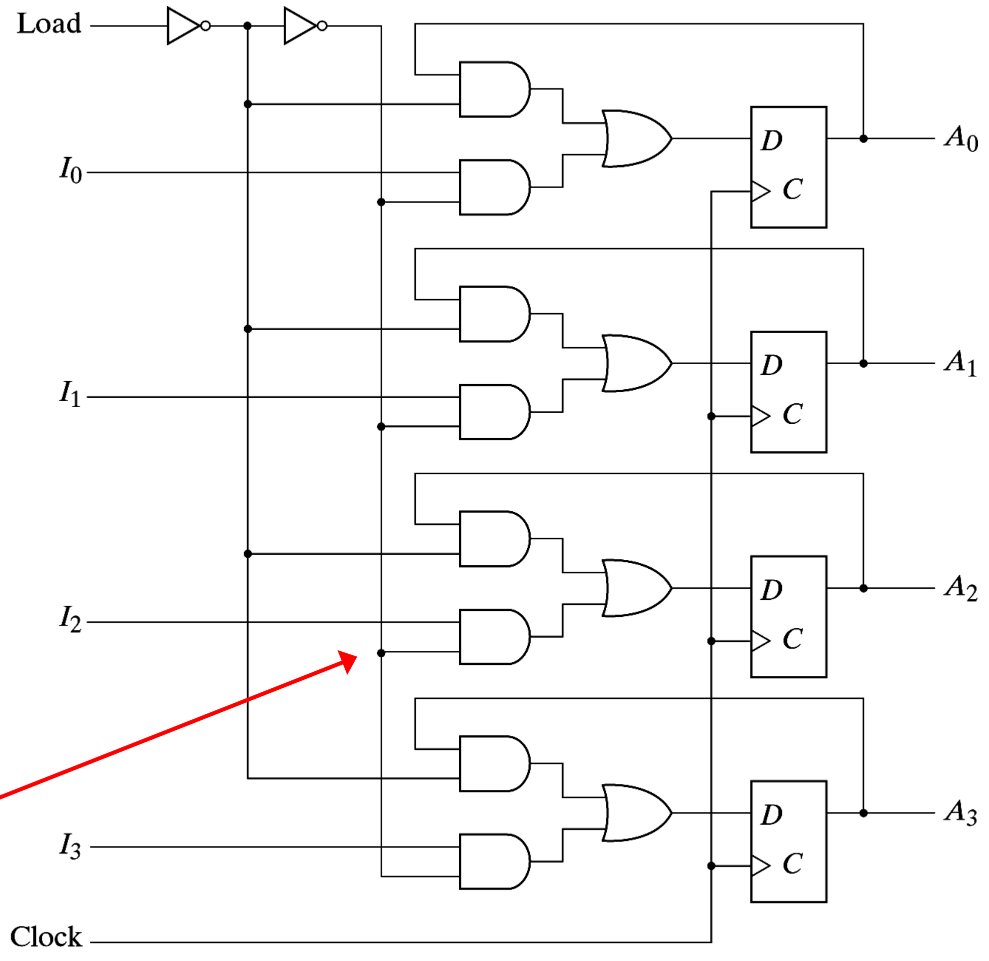


Fig. 6-2 4-Bit Register with Parallel Load

Outline

- Registers
- $\boxed{\text{Shift Registers}}$
- Ripple Counters
- Synchronous Counters
- Other Counters

The Simplest Shift Register

- Shift register: a register capable of shifting its binary information in one or both directions
- The simplest form: consist of only a chain of flip-flops in cascade

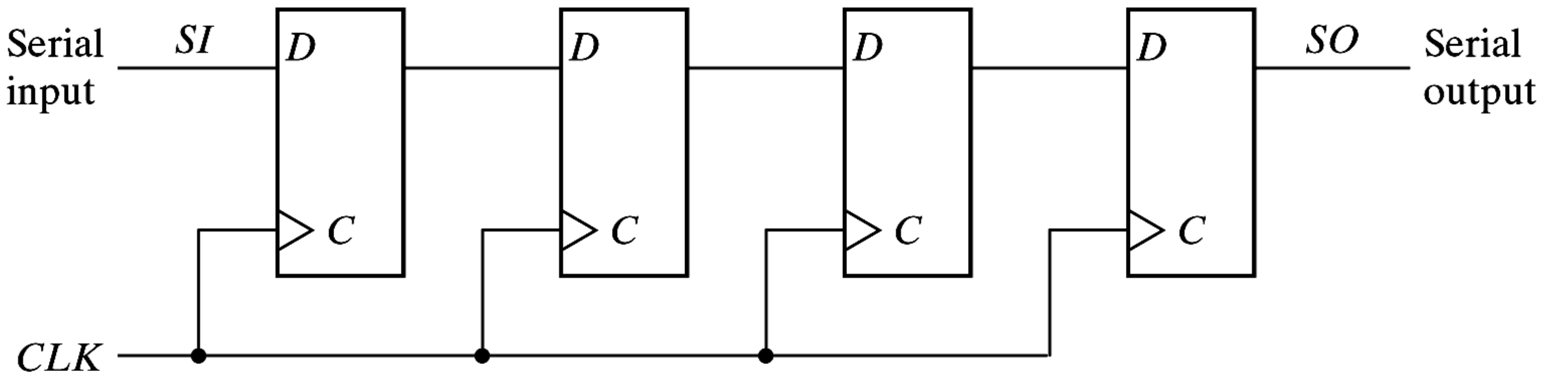
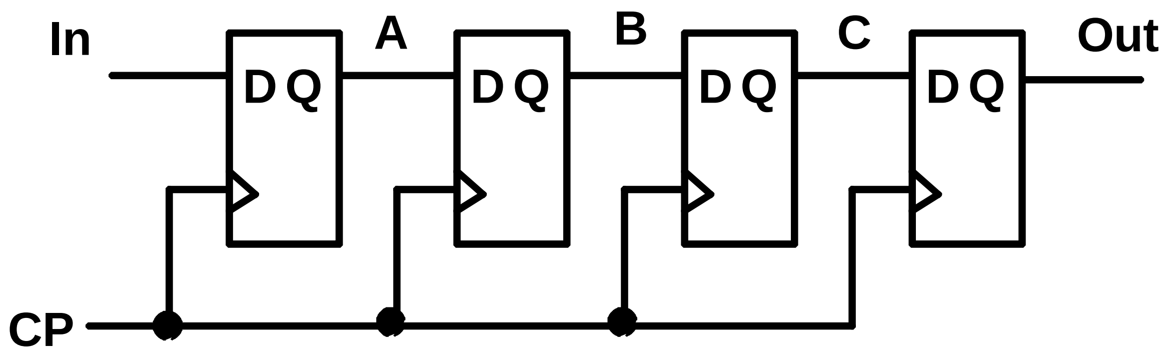


Fig. 6-3 4-Bit Shift Register

Shift Registers

- Shift Registers move data laterally within the register toward its MSB or LSB position
- In the simplest case, the shift register is simply a set of D flip-flops connected in a row like this:



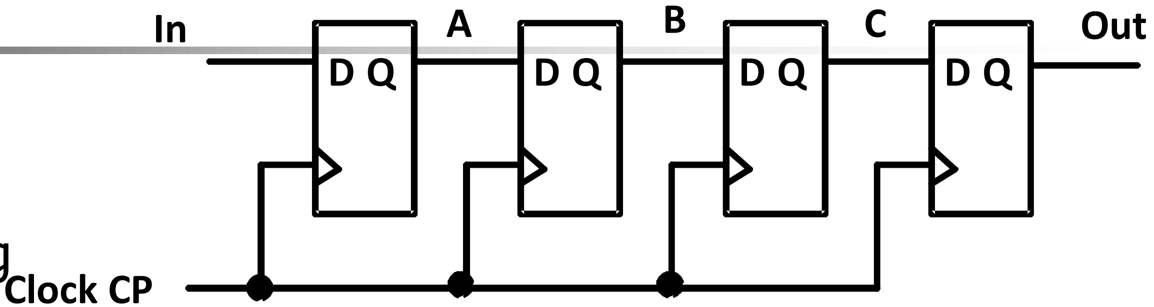
- Data input, In, is called a *serial input* or the *shift right input*.
- Data output, Out, is often called the *serial output*.
- The vector (A, B, C, Out) is called the *parallel output*.

Shift Registers

- Capability to shift bits
  - In one or both directions
- Why?
  - Part of standard CPU instruction set
  - Cheap multiplication/division
  - Serial communications
- Just a chain of flip-flops

Shift Registers (continued)

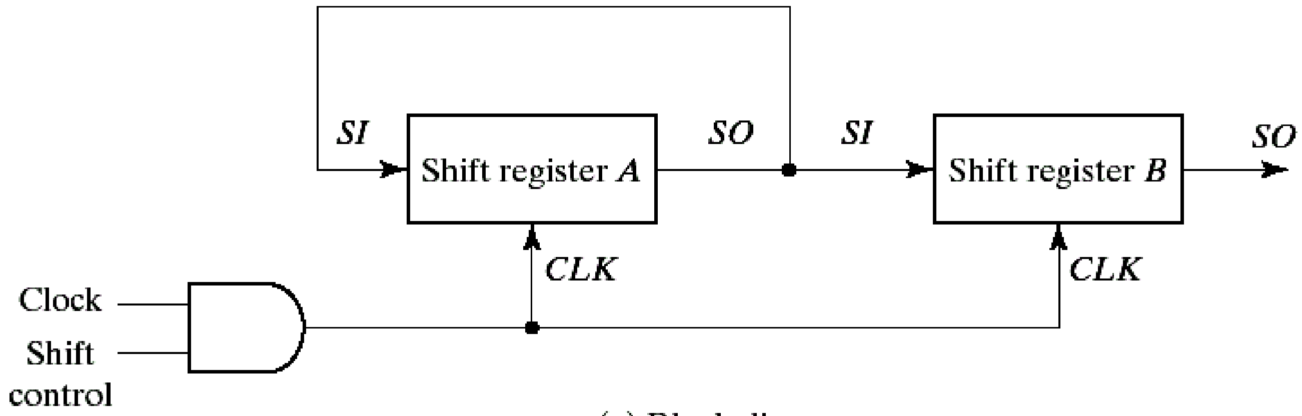
- The behavior of the serial shift register is given in the listing on the lower right
- T0 is the register state just before the first clock pulse occurs
- T1 is after the first pulse and before the second.
- Initially unknown states are denoted by "?"
- Complete the last three rows of the table



CP	In	A	B	C	Out
T0	0	?	?	?	?
T1	1	0	?	?	?
T2	1	1	0	?	?
T3	0	1	1	0	?
T4	1				
T5	1				
T6	1				

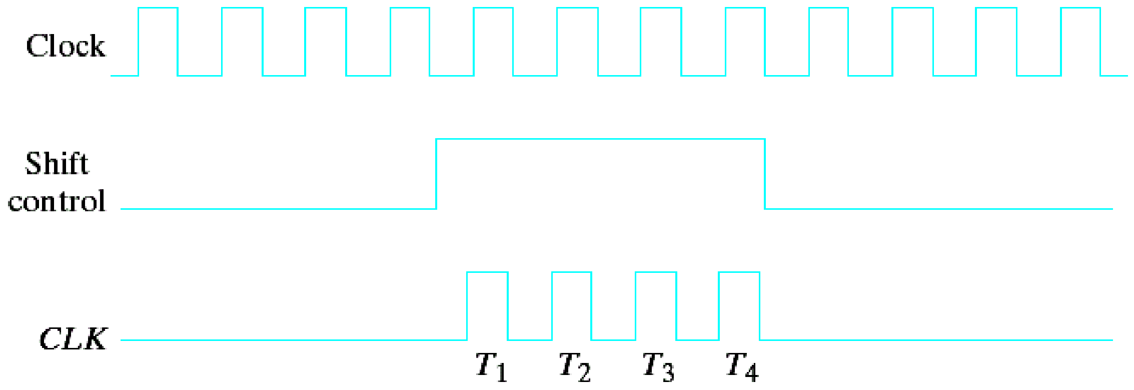
Serial Transfer

- Serial mode: information is transferred and manipulated one bit at a time
- Parallel mode: all the bits of the registers are transferred at the same time
- Shift control: determine when and how many times of the registers are shifted
- Register A is connected in circular mode in this circuit



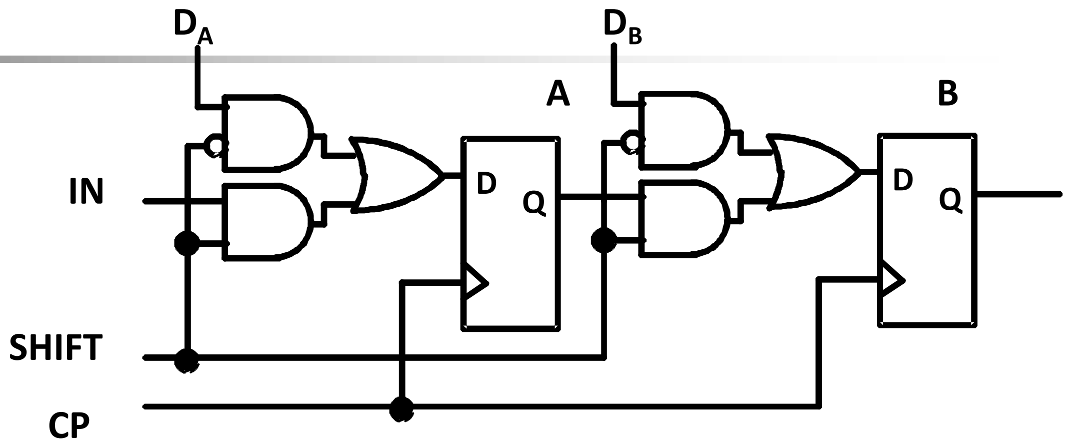
(a) Block diagram

Serial Transfer Example



Timing Pulse	Shift Register A	Shift Register B
Initial value	1 0 1 1	0 0 1 0
After T1	1 1 0 1	1 0 0 1
After T2	1 1 1 0	1 1 0 0
After T3	0 1 1 1	0 1 1 0
After T4	1 0 1 1	1 0 1 1

Parallel Load Shift Registers



- By adding a mux between each shift register stage, data can be shifted or loaded
- If SHIFT is low, A and B are replaced by the data on D_A and D_B lines, else data shifts right on each clock.
- By adding more bits, we can make *n*-bit parallel load shift registers.
- A parallel load shift register with an added "hold" operation that stores data unchanged is shown in textbook.

Shift Register with Parallel Load

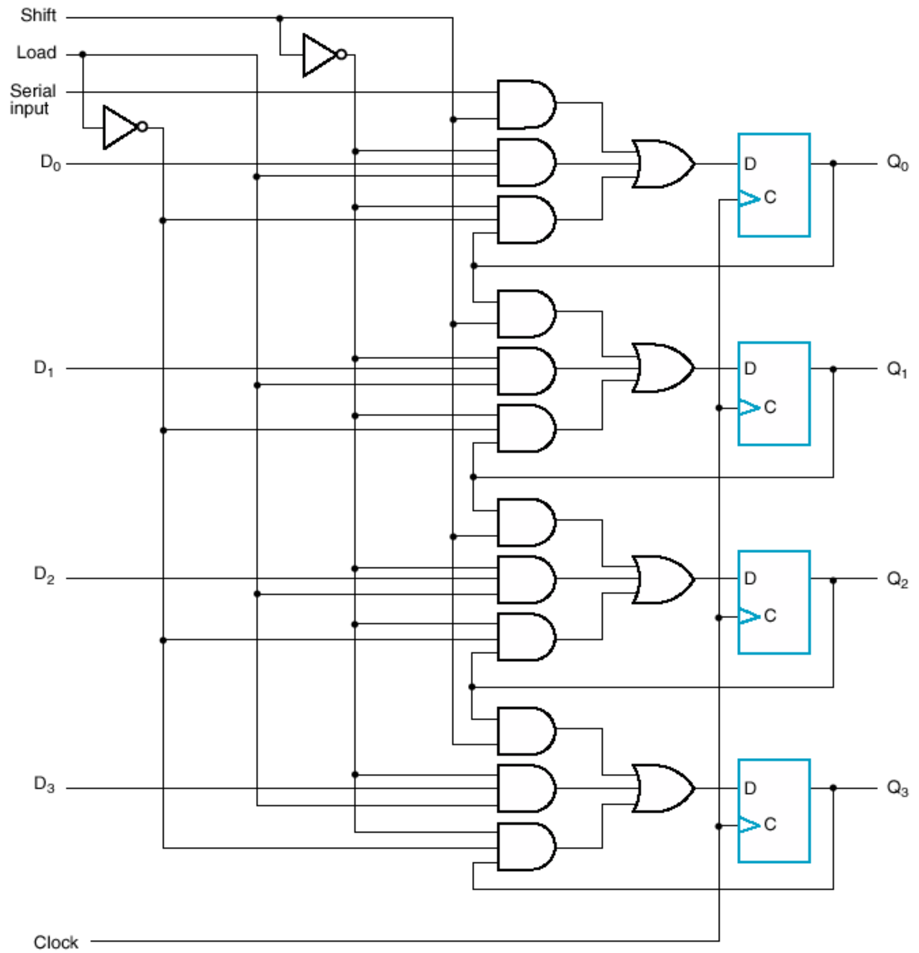
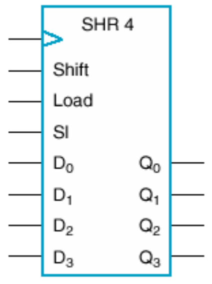


Fig. 5-6 Shift Register with Parallel Load



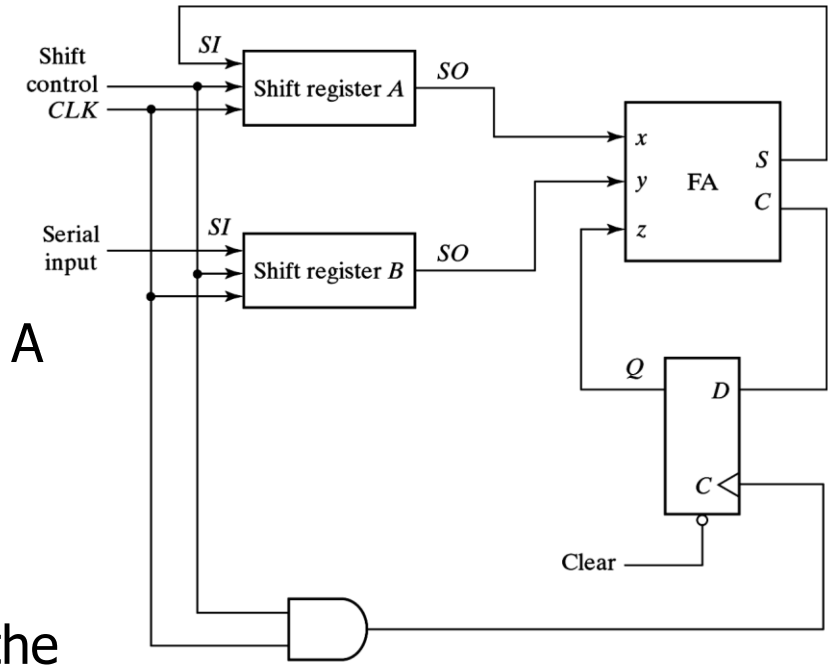
(b) Symbol

Function Table for the Register of Figure 5-6

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

Serial Adder

- The bits of two binary numbers are added one pair at a time through a single full adder (FA) circuit
- Initialization:
  - A = augend; B = addend
  - Carry flip-flop is cleared to 0
- For each clock pulse:
  - A new sum bit is transferred to A
  - A new carry is transferred to Q
  - Both registers are shifted right
- To add three or more numbers:
  - Shift in the next number from the serial input while B is shifted to the FA
  - A will accumulate their sum



Serial v.s. Parallel

- Serial adders:
  - Use shift registers
  - A sequential circuit
  - Require only one FA and a carry flip-flop
  - Slower but require less equipment

- Parallel adders:
  - Use registers with parallel load for sum
  - Basically a pure combinational circuit
  - ***n*** FAs are required
  - Faster

Shift Registers with Additional Functions - Universal Shift Register

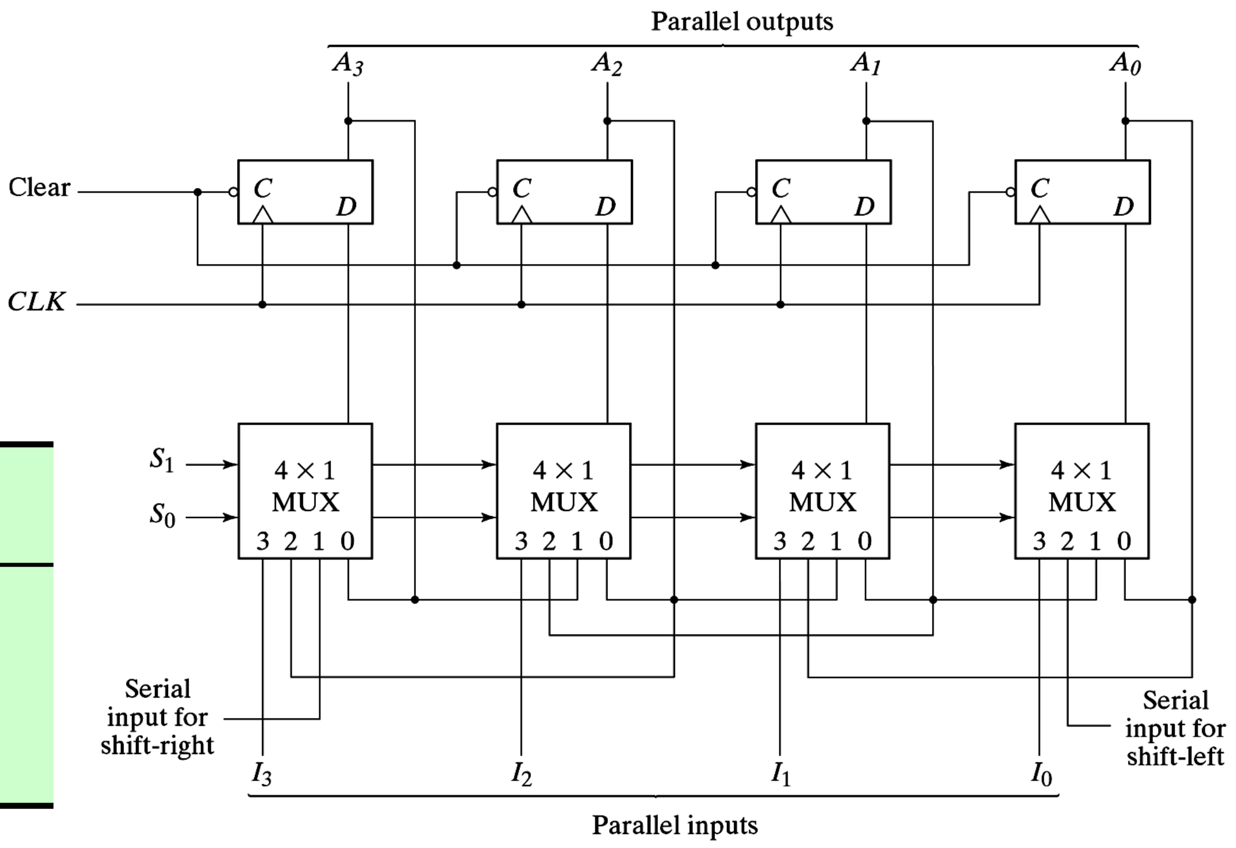
- By placing a 4-input multiplexer in front of each D flip-flop in a shift register, we can implement a circuit with shifts right, shifts left, parallel load, hold.
- Shift registers can also be designed to shift more than a single bit position right or left
- Shift registers can be designed to shift a variable number of bit positions specified by a variable called a *shift amount*.

Universal Shift Register

- The most general shift register has the following capabilities:
  - A *clear* control to clear the register to 0
  - A *clock* input to synchronize the operations
  - A *shift-right (left)* control to enable the shift right (left) operation and the *serial input* and *output* lines associated with the shift right (left)
  - A *parallel-load* control to enable a parallel transfer and the *n* input lines associated with the parallel transfer
  - *n* parallel output lines
  - A control state that leaves the information in the register unchanged in the presence of the clock

4-Bit Universal Shift Register

Mode Control		Register Operation
S1	S0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load



Outline

- Registers
- Shift Registers
- $\boxed{\text{Ripple Counters}}$
- Synchronous Counters
- Other Counters

Registers and Counters

- Register:
  - A set of flip-flops, possibly with added combinational gates, that perform data-processing tasks
  - Store and manipulate information in a digital system
- Counter:
  - A register that goes through a predetermined sequence of states
    - A special type of register
  - Employed in circuits to sequence and control operations

Counters

- An example of a register.
- Primary purpose is to produce a specified output pattern sequence.
  - Also called a pattern generator
- Each stored 0/1 combination is called the state of the counter.
- The total number of states is called its modulus.
  - If a counter has m distinct states then it is called a mod-m counter.
- The order in which states appear is referred to as its counting sequence.
  - Depicted by a directed graph called a state diagram.

State Diagram of a Counter

- S_i denotes one of the states of the counter.
- Arrows in the graph denote the order in which the states occur.

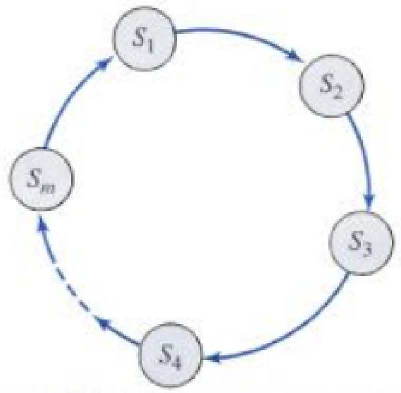
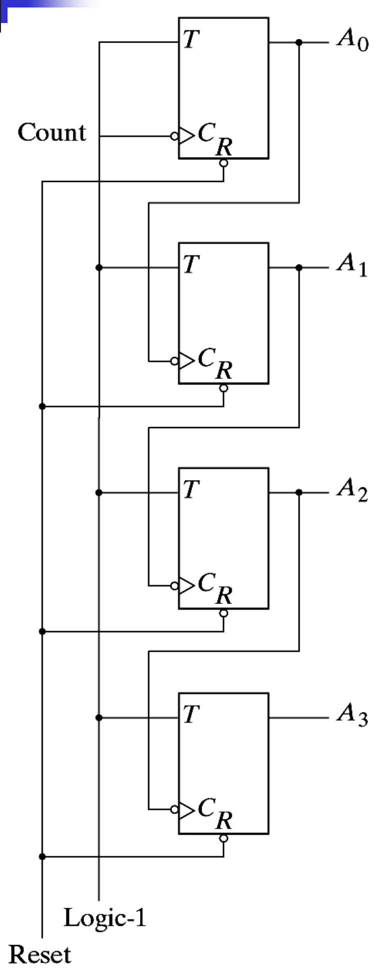


Figure 6.30 State diagram of a counter.

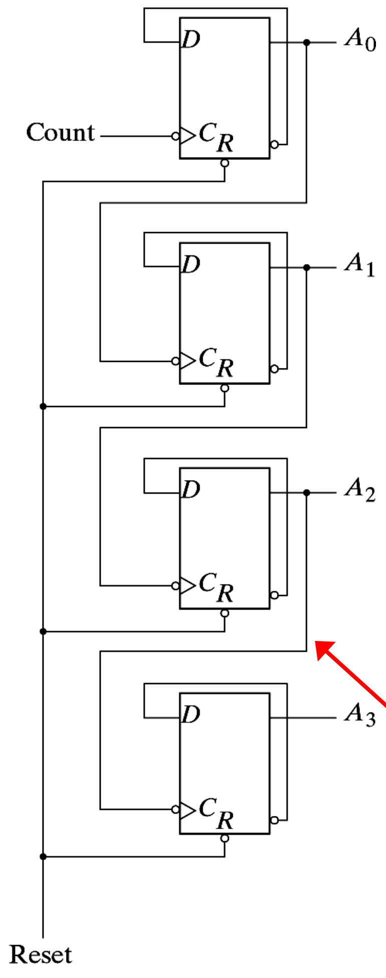
Binary Ripple Counters

- Counters whose counting sequence corresponds to that of the binary numbers are called binary counters.
- Modulus is 2^n, where n is the number of flip-flops in the counter.
- Binary up-counter, binary down-counter

Binary Ripple Counter



(a) With T flip-flops



(b) With D flip-flops

Binary Count Sequence

A3	A2	A1	A0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

the output transition triggers the next flip-flop

Outline

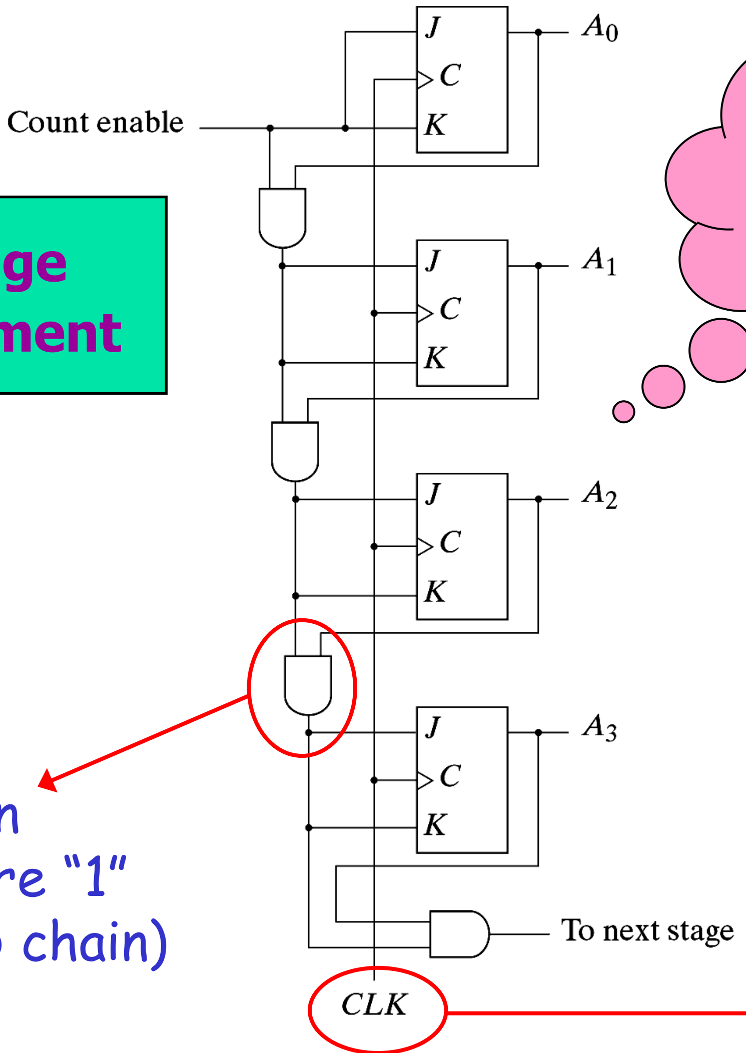
- Registers
- Shift Registers
- Ripple Counters
- $\boxed{\text{Synchronous Counters}}$
- Other Counters

Ripple v.s. Synchronous

- Ripple counters:
  - Flip-flops are triggered by the outputs of another flip-flops
    - Triggering source may not the same for each flip-flop
  - The flip-flops are changed serially
- Synchronous counters:
  - Flip-flops are triggered by common clock pulses
    - Triggering sources are the same for all flip-flops
  - All operations are performed simultaneously

Binary Counter

J=0,K=0: no change
J=1,K=1: complement

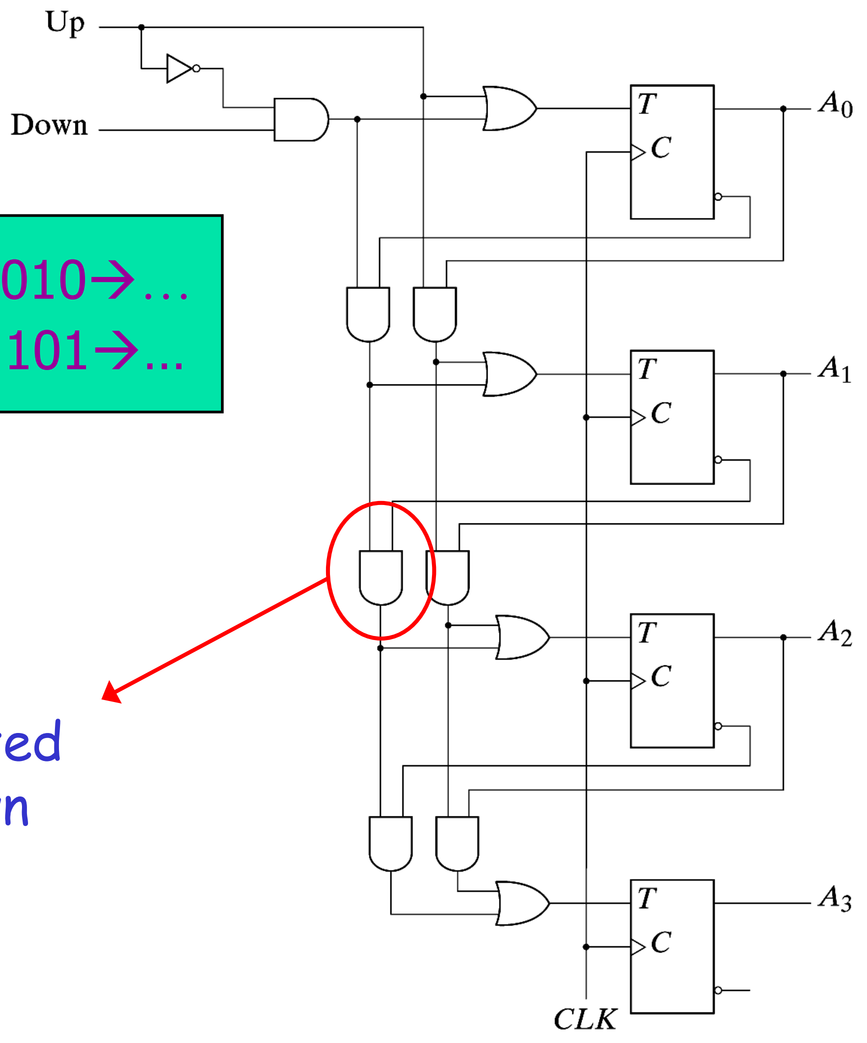


$0 \rightarrow 1 \rightarrow \ldots \rightarrow 15 \rightarrow 0 \rightarrow 1 \rightarrow \ldots$

complemented when
all the lower bits are "1"
(checked by a AND chain)

triggered by
positive clock edge

Up-Down Binary Counter



Up: 0000→0001→0010→…
Down: 1111→1110→1101→…

take the complemented values for count-down calculation

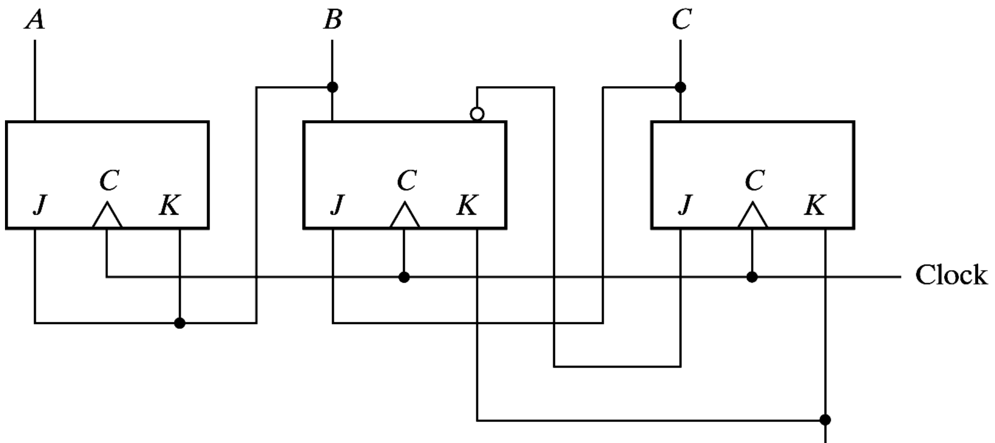
Outline

- Registers
- Shift Registers
- Ripple Counters
- Synchronous Counters
- $\boxed{\text{Other Counters}}$

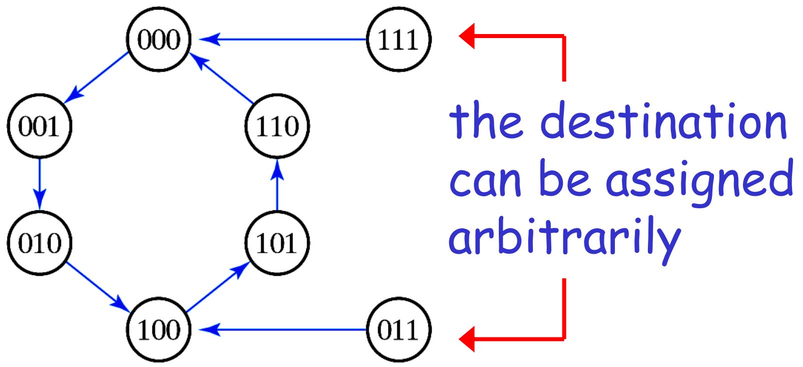
Counter with Unused States

- If the machine falls into the unused states, we have to bring it back !!
  - Cannot assign the used states as don't cares

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	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X



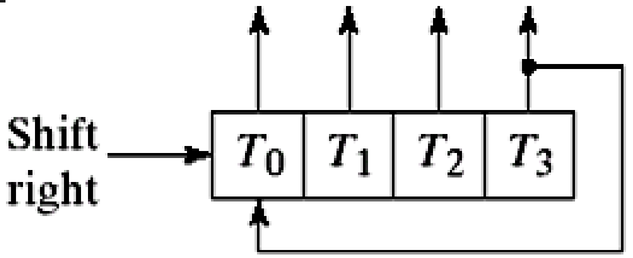
(a) Logic diagram



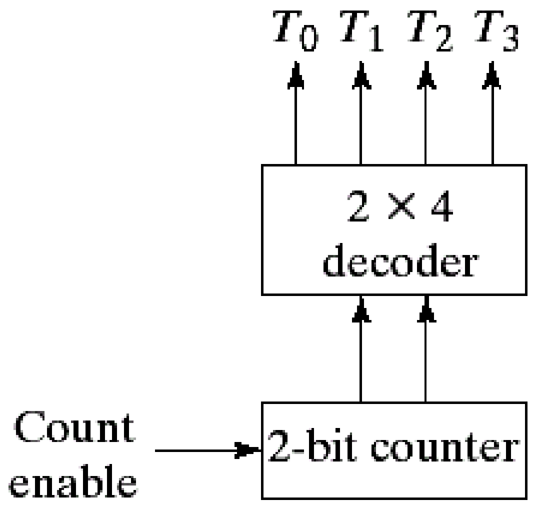
(b) State diagram

Ring Counter

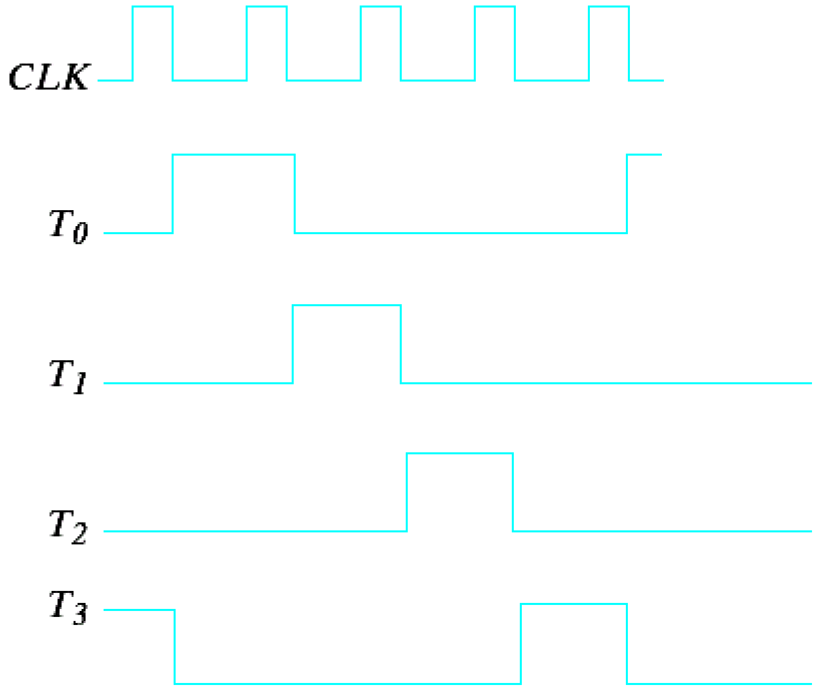
Ring counter: a circular shift register with only one flip-flop being set at any time



(a) Ring-counter (initial value = 1000)



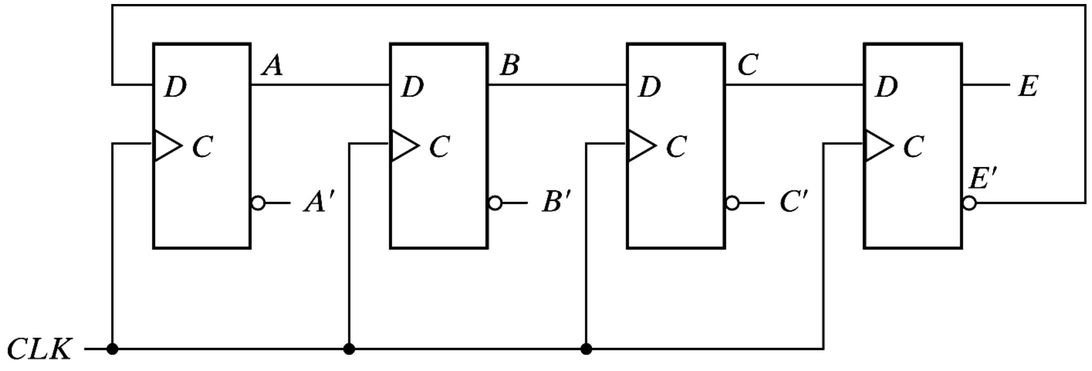
(b) Counter and decoder



(c) Sequence of four timing signals

Johnson Counter

- Double the number of states of a ring counter by switch-tail connection
- If this counter falls into an unused state, it will never come back to normal states !!
- To avoid this situation, use some gates to form the input equation of each flip-flop instead of connecting directly
  - Ex: Dc = (A + C) B



(a) Four-stage switch-tail ring counter

Sequence number	Flip-flop outputs A	B	C	E	AND gate required for output
1	0	0	0	0	$A'E'$
2	1	0	0	0	AB'
3	1	1	0	0	BC'
4	1	1	1	0	CE'
5	1	1	1	1	AE
6	0	1	1	1	$A'B$
7	0	0	1	1	$B'C$
8	0	0	0	1	$C'E$

(b) Count sequence and required decoding