CENG-232
Logic Design

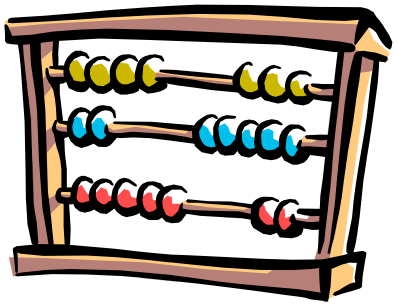
Lecture 7
Registers and Counters

Spring 2015 - Uluç Saranlı
saranli@ceng.metu.edu.tr

Acknowledgement: Most of the following slides are adapted from Prof. Kale's slides at UIUC, USA.

Registers and Counters

- Sequential circuits are classified based in their function, e.g., registers.
- Register is a group of flip-flops each storing one bit of information.
- Registers are built-up by using flip-flops and gates:
  - flip-flops hold the information, gates control how the information is transferred to the register.
- Counter is a register that can go through a predetermined sequence of states.



What good are registers?

- Flip-flops are limited in size because they can store only one bit.
  - We had to use two flip-flops for our two-bit counter examples.
  - Most computers work with integers and single-precision floating-point numbers that are 32-bits or 64-bits long.
- Registers are commonly used as temporary storage in a processor (PL variables are mapped to registers).
  - They are faster and more convenient than main memory.
  - More registers can help speed up complex calculations.

4-bit Register

- Loads in parallel
- Clear:
  - Cleans the output to all 0's.

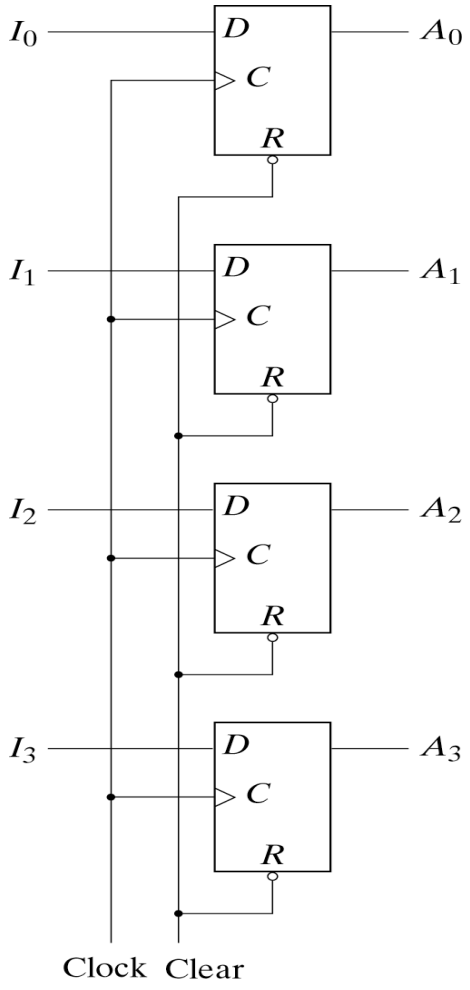


Fig. 6-1 4-Bit Register

Register with Parallel Load

- To fully synchronize, the system clock signals should arrive at the same time at all flip-flops.
  - Therefore we do not control the clock by gates.
- Load = 1, we load data ()
- Load = 0, register content does not change ()

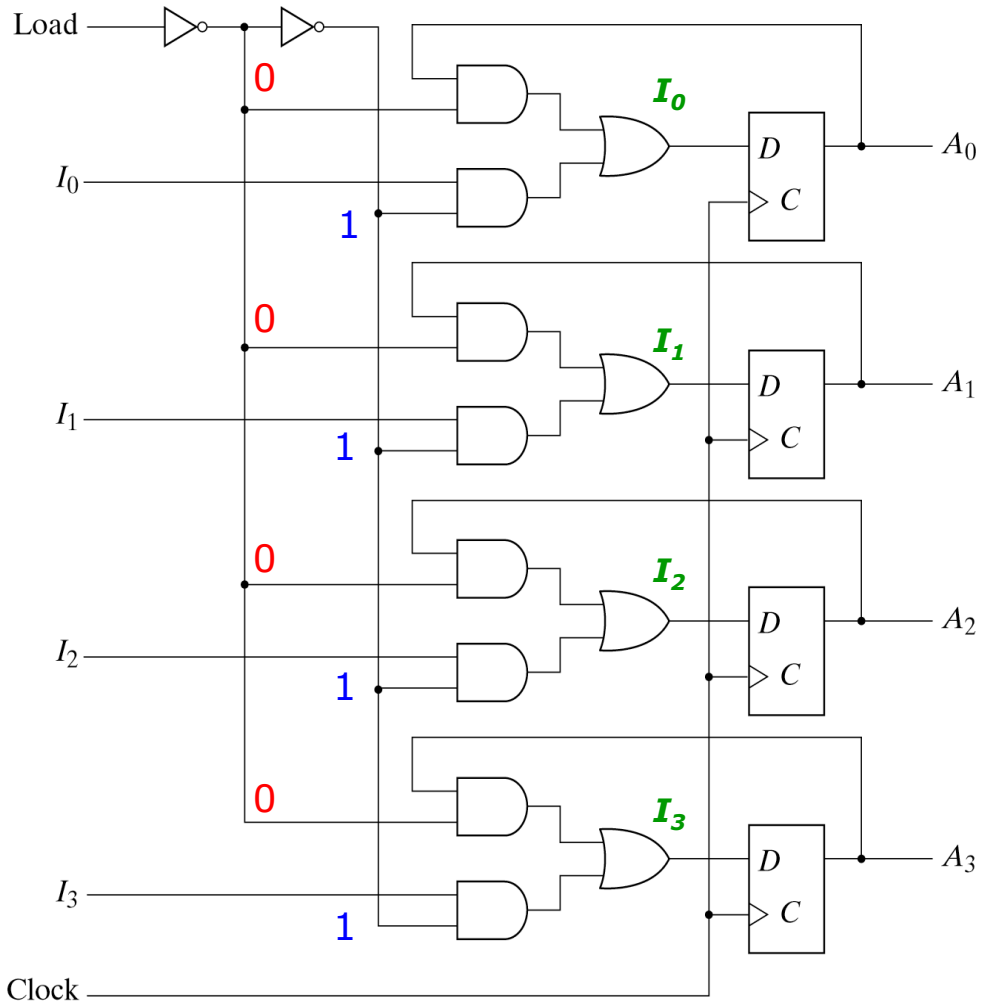
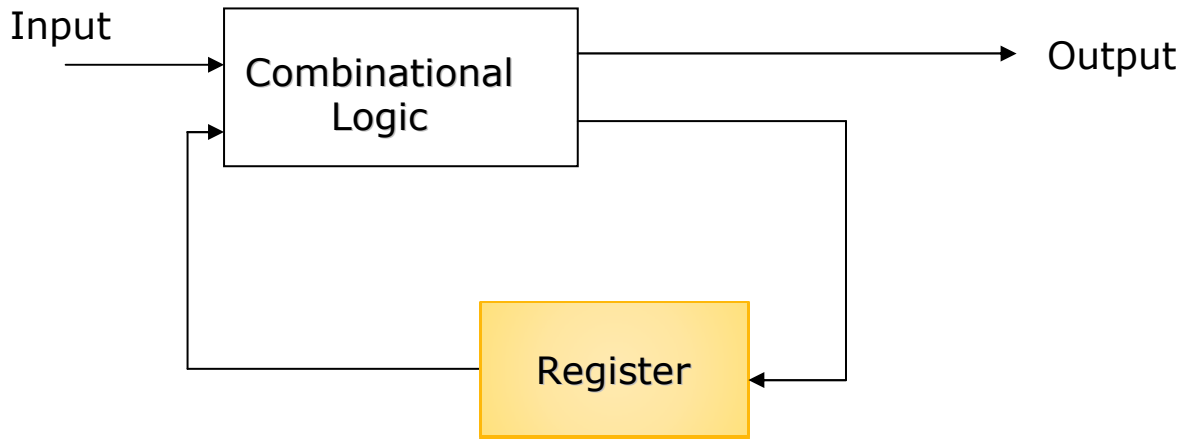


Fig. 6-2 4-Bit Register with Parallel Load

Implementing Sequential Circuits

- Using a register with parallel load capability, one can implement arbitrary sequential circuits.
- The register can act as the memory component of the sequential circuit.



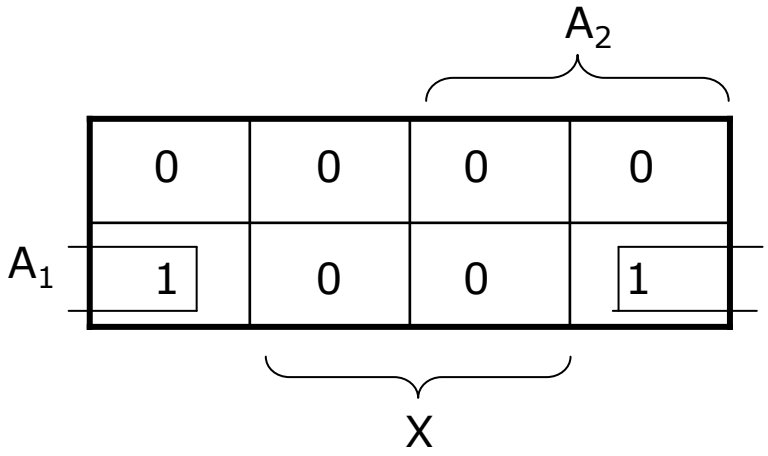
Sequential Circuit

Example

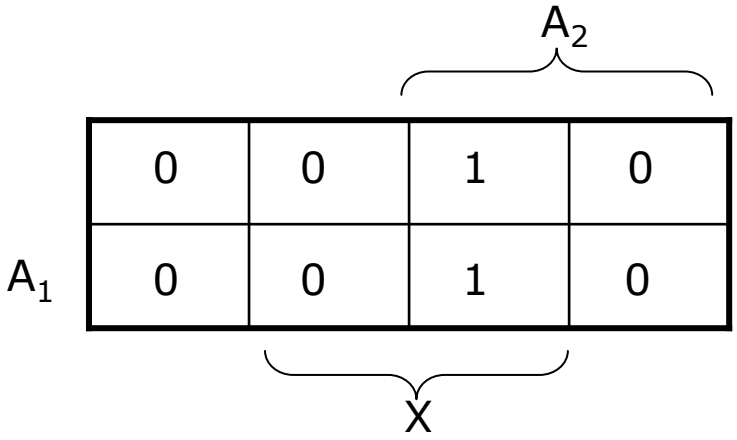
Present State		Input	Next State		Output
A_1	A_2	x	A_1	A_2	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	0	0	1

State table

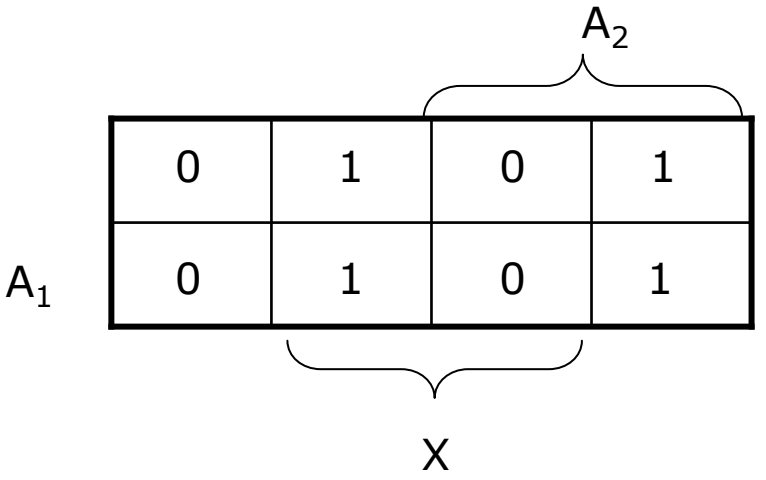
Example (Cont'd)



$A_1 = A_1 X'$

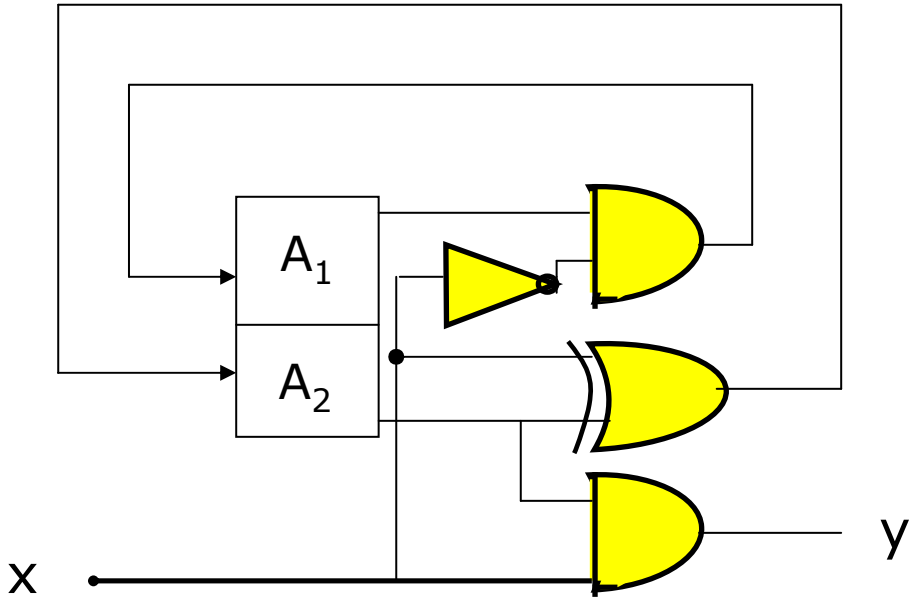


$Y = A_2 X$



$A_2 = A'_2 X + A_2 X$
$= A_2 \oplus X$

.. with a combinational circuit

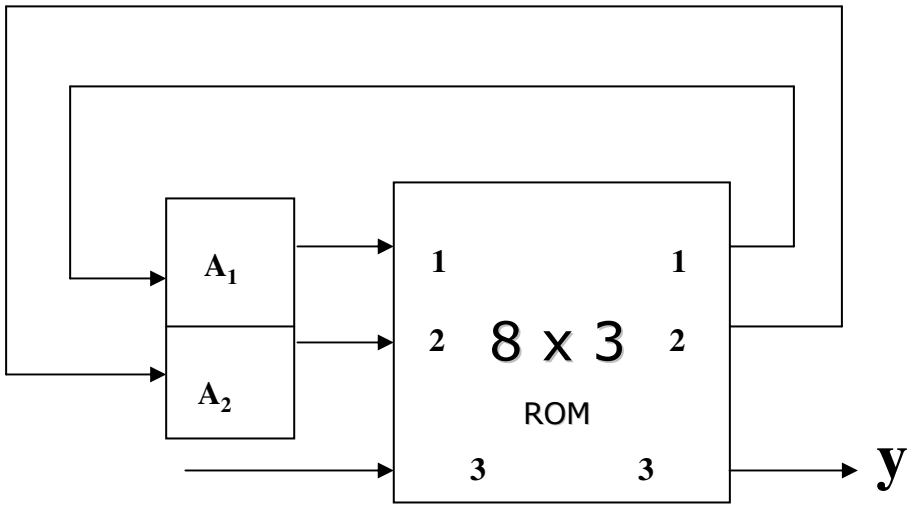


b) Logic Diagram

… with ROM

ROM truth table

Address			Outputs		
1	2	3	1	2	3
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	0	0	1



Shift Registers

- A register capable of shifting its binary information in one or both directions is called "*shift register*".

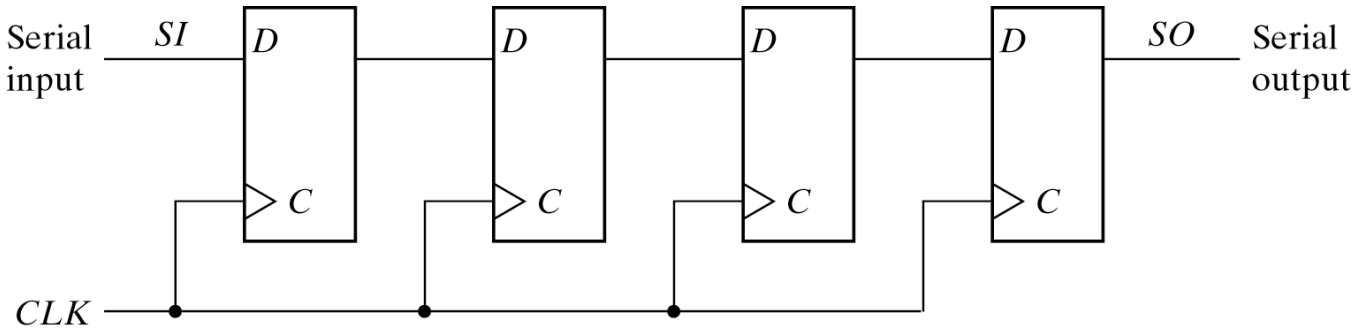


Fig. 6-3 4-Bit Shift Register

Serial Transfer

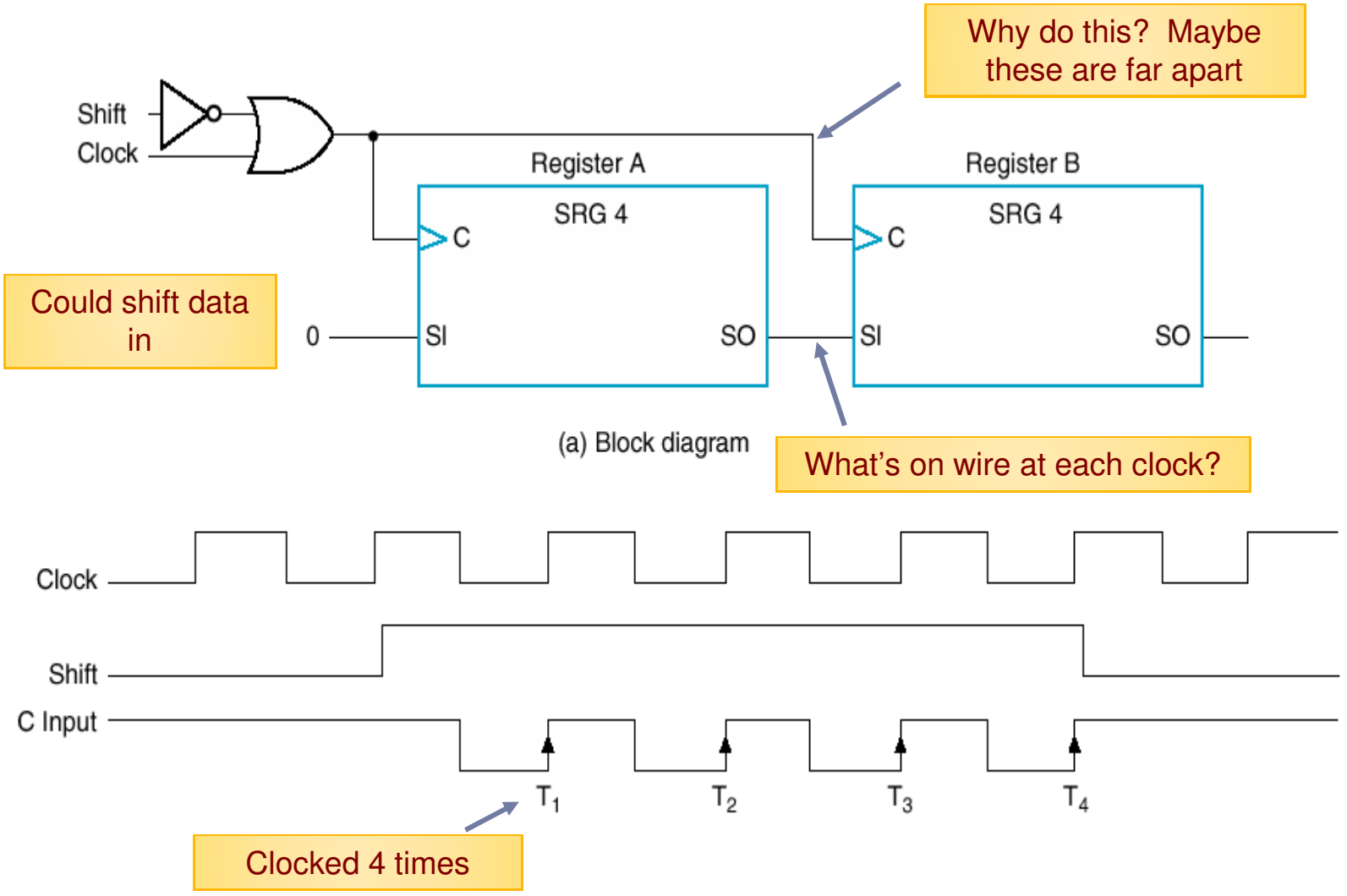


Table Showing Shift

Example of 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

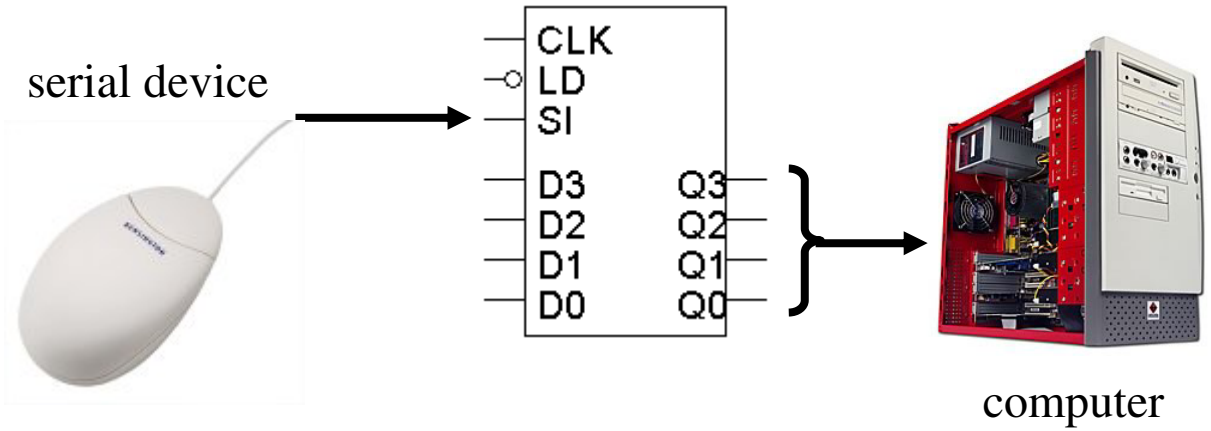
- A digital system is in the *serial mode* when information is processed one bit at a time (bit after bit).

Serial data transfer

- One application of shift registers is converting between “serial data” and “parallel data.”
- Computers typically work with multiple-bit quantities.
  - ASCII text characters are 8 bits long.
  - Integers, single-precision floating-point numbers, and screen pixels are up to 32 bits long.
- But sometimes it’s necessary to send or receive data serially, or one bit at a time. Some examples include:
  - Input devices such as keyboards and mice.
  - Output devices like printers.
  - Any serial port, USB or Firewire device transfers data serially.

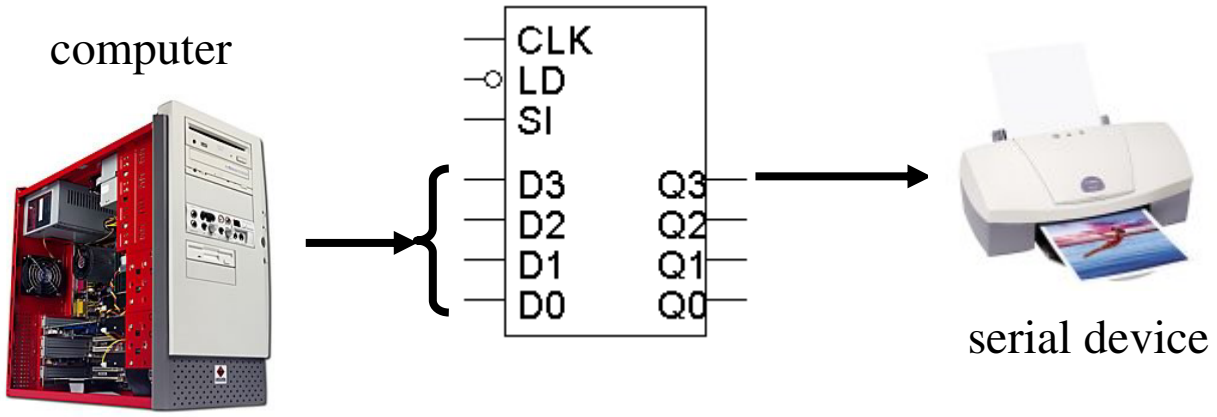
Receiving serial data

- To receive serial data using a shift register:
  - The serial device is connected to the register's SI input.
  - The shift register outputs Q3-Q0 are connected to the computer.
- The serial device transmits one bit of data per clock cycle.
  - These bits go into the SI input of the shift register.
  - After four clock cycles, the shift register will hold a four-bit word.
- The computer then reads all four bits at once from the Q3-Q0 outputs.



Sending data serially

- To send data serially with a shift register, you do the opposite:
  - The CPU is connected to the register's D inputs.
  - The shift output (Q3 in this case) is connected to the serial device.
- The computer first stores a four-bit word in the register, in one cycle.
- The serial device can then read the shift output.
  - One bit appears on Q3 on each clock cycle.
  - After four cycles, the entire four-bit word will have been sent.



Serial vs. Parallel

- Communication / Interfaces
  - RS-232 / Centronix
  - USB
  - SATA / PATA
  - PCI / PCI-X / PCI-e

Remember the 4-bit Parallel Adder

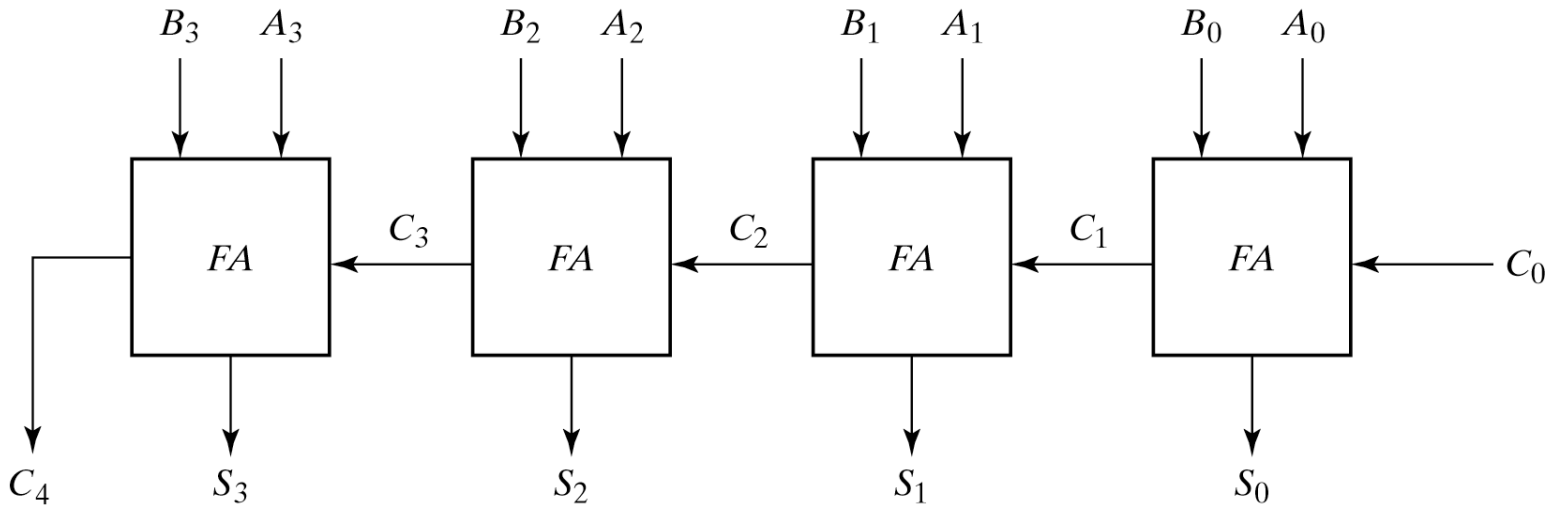


Fig. 4-9 4-Bit Adder

Serial Addition

- Slower compared to parallel addition, but uses less equipment.

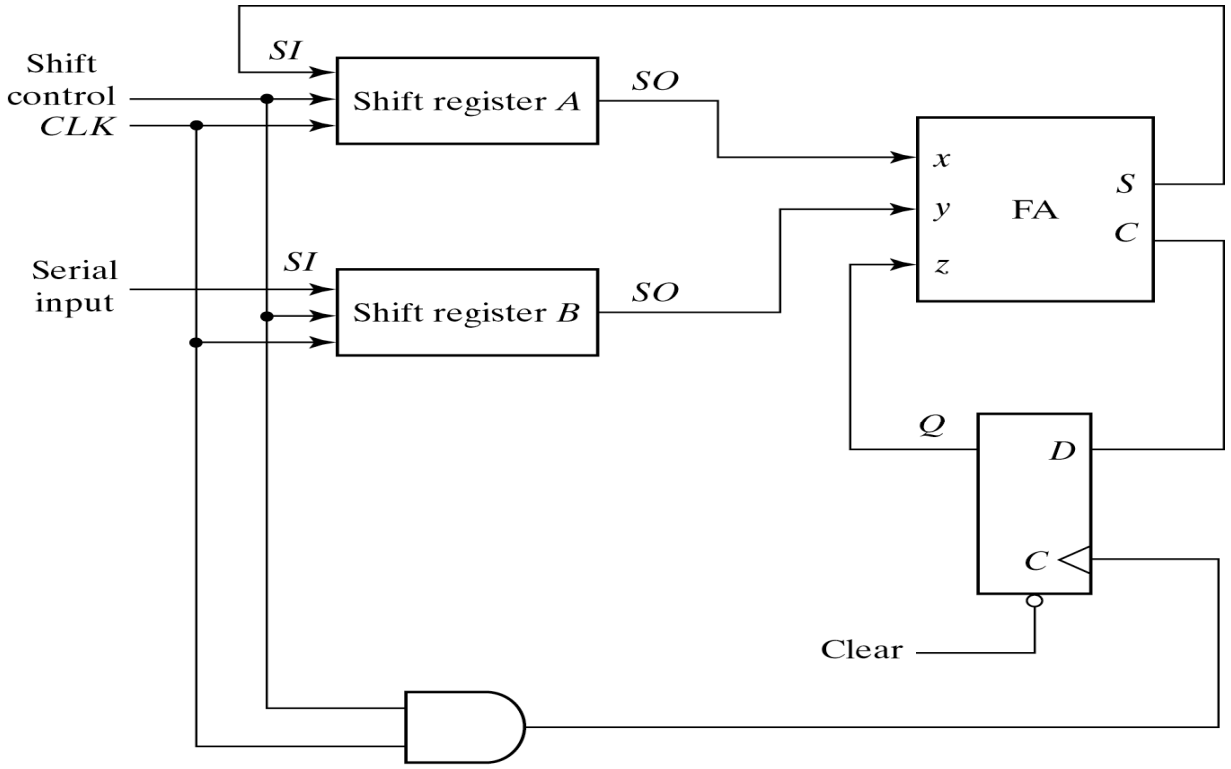


Fig. 6-5 Serial Adder

Serial Adder vs. Parallel Adder

- PA uses registers with parallel load, SA uses shift registers.
- PA uses more FAs compared to SA.
- Excluding the registers, PA is a combinational circuit, SA is sequential.

Serial Adder: Design Procedure

- State Table for a Serial Adder

Present State	Inputs		Next State	Output	Flip-Flop inputs	
Q	x	y	Q	S	J0	K0
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

J0 = xy K0 = x'y' = (x+y)' S = x XOR y XOR z

Serial 4-bit Parallel Adder Circuit

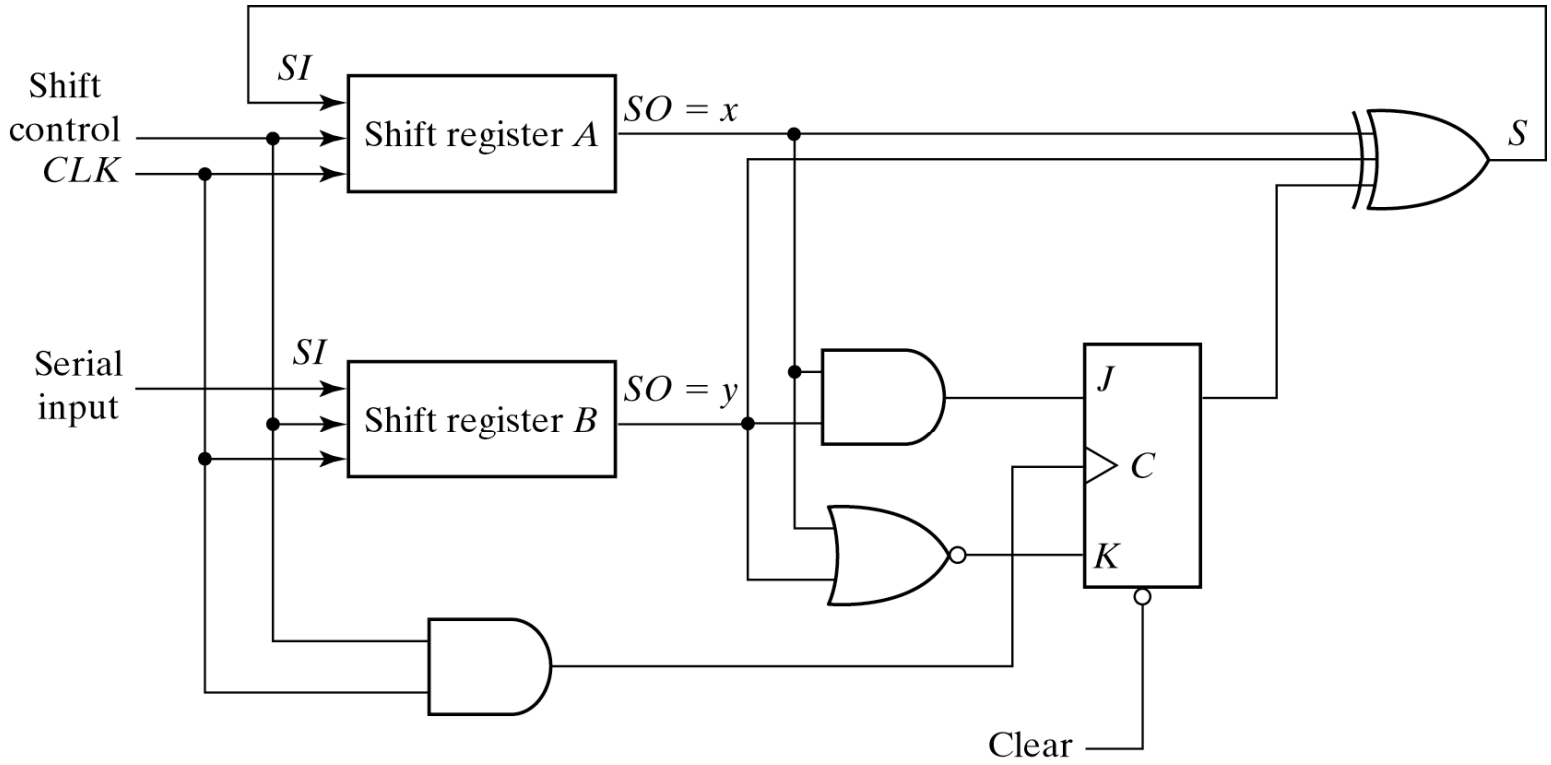


Fig. 6-6 Second form of Serial Adder

Universal Shift Register

- A register capable of shifting in both directions and loading in parallel.

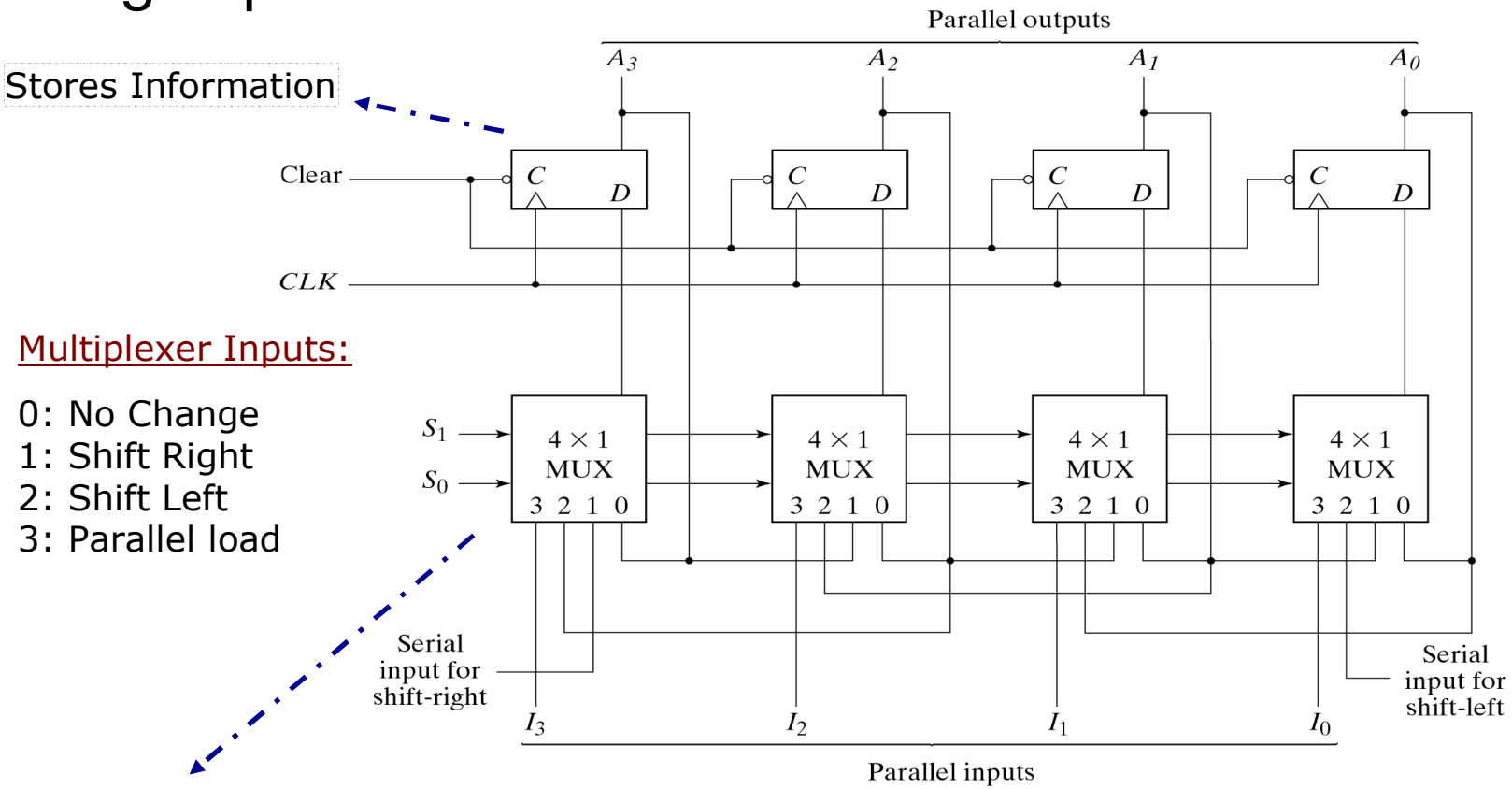


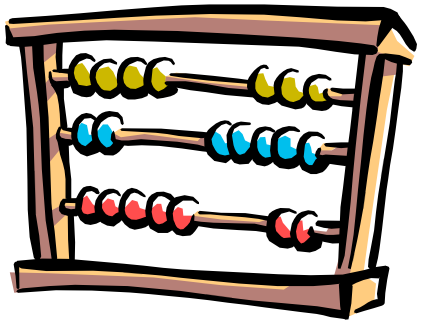
Fig. 6-7 4-Bit Universal Shift Register

Serial vs. Parallel: Pros and Cons

- Serial vs. Parallel adder
- One full adder vs. n adders
- Serial takes n units of time, parallel takes only one
- …

Counters

- A register that goes trough a prescribed sequence of states is called a counter.
- Binary counter
  - Counts through binary sequence
  - n bit counter counts from 0 to 2n
- There are two groups of counters: “Ripple counters” and “Synchronous counters”.
  - Ripple counters: The flip-flop output triggers other flip-flops in sequence.
  - Synchronous counters count the clock.



What good are counters?

- Counters can act as simple clocks to keep track of “time”.
  - Absolute/Relative timing
- You may need to record how many times something has happened.
  - How many bits/bytes have been sent or received?
  - How many steps have been performed in some computation?

- All processors contain a program counter, or PC.
  - Programs consist of a list of instructions that are to be executed one after the other (for the most part; except ?).
  - The PC keeps track of the instruction currently being executed.
  - The PC increments once on each instruction cycle, and the next program instruction is then executed.

Binary Ripple Counter

- A binary ripple counter consists of a series of complementing flip-flops, with the output of each flip-flop connected to the next higher order.
- Examples of complementing flip-flops are T and D (with the output complement connected to the input) flip-flop.
- Binary Count Sequence

A3	A2	A1	A0	
0	0	0	0	
0	0	0	1	A0 is complemented at each count pulse
0	0	1	0	A1 is complemented when A0 goes from 1 to 0
0	0	1	1	
0	1	0	0	A2 is complemented when A1 goes from 1 to 0
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	A3 is complemented when A2 goes from 1 to 0

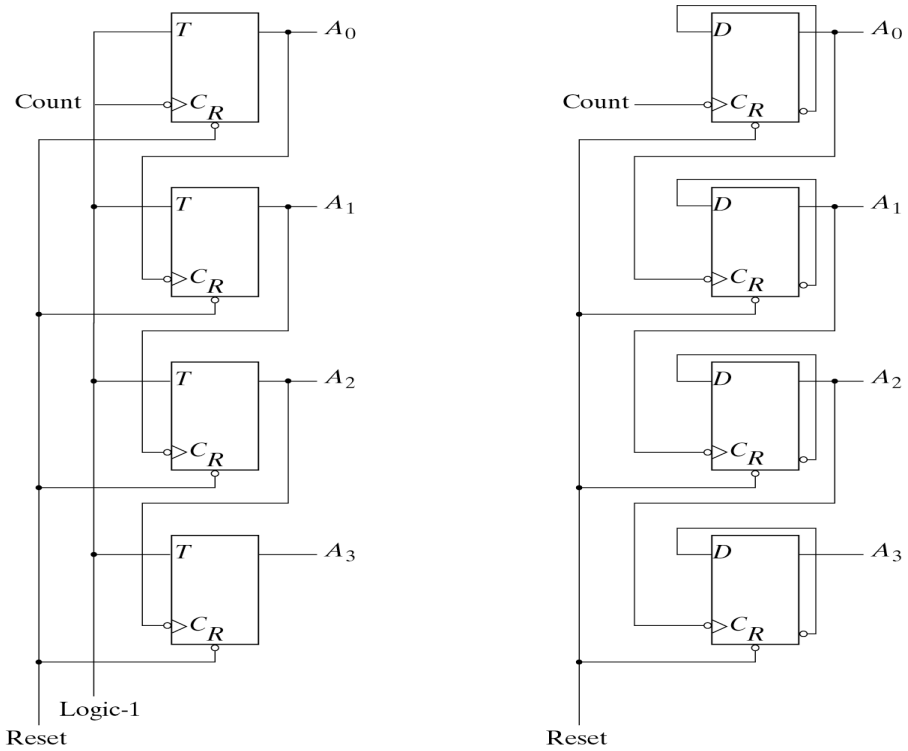
Binary Ripple Counter

- Count-down counter: A binary counter with reverse count. Starts from 15, goes down.

- In a count-down counter, the least significant bit is complemented with every count pulse. Any other bit is complemented if the previous bit goes from 0 to 1.
  - Therefore, we can use the same counter design with negative edge flip-flops to make a count-down flip-flop.

Examples of Ripple Counters

- No clocks!
- Why is the Count is negated at the clock input?



(a) With T flip-flops (b) With D flip-flops

Fig. 6-8 4-Bit Binary Ripple Counter

BCD Ripple Counter

- A BCD (Binary Coded Decimal) counter starts from 0 ends at 9.

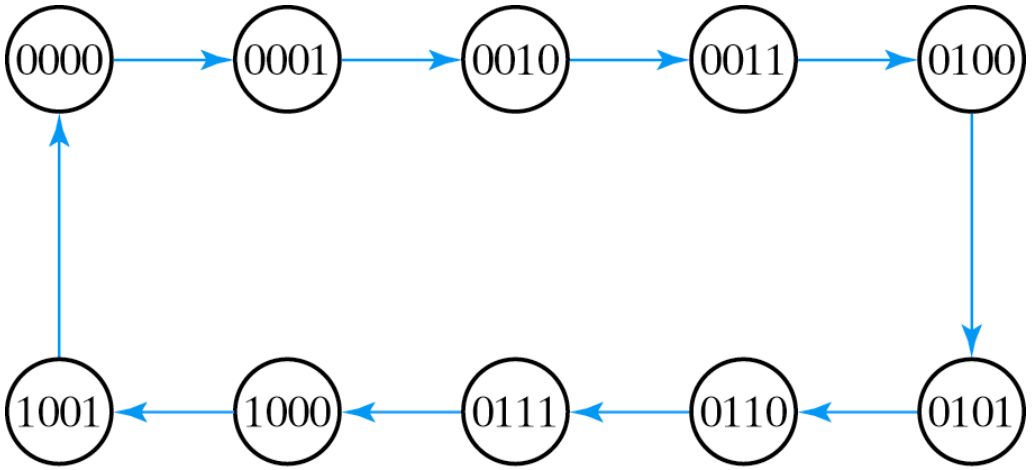


Fig. 6-9 State Diagram of a Decimal BCD-Counter

Logic Diagram of Ripple Counter

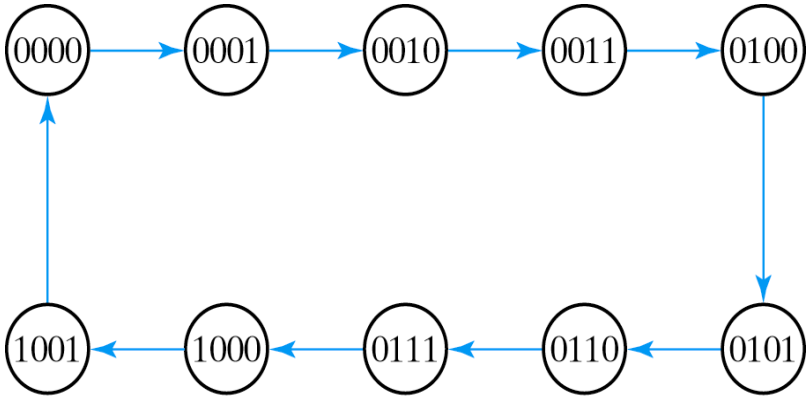


Fig. 6-9 State Diagram of a Decimal BCD-Counter

- Q1 is applied to the C inputs of Q2 and Q8 (?)
- Q2 is applied to the C input of Q4 (?)
- J and K are connected to either 1 or flip-flop outputs (?)
- *Study !!!*

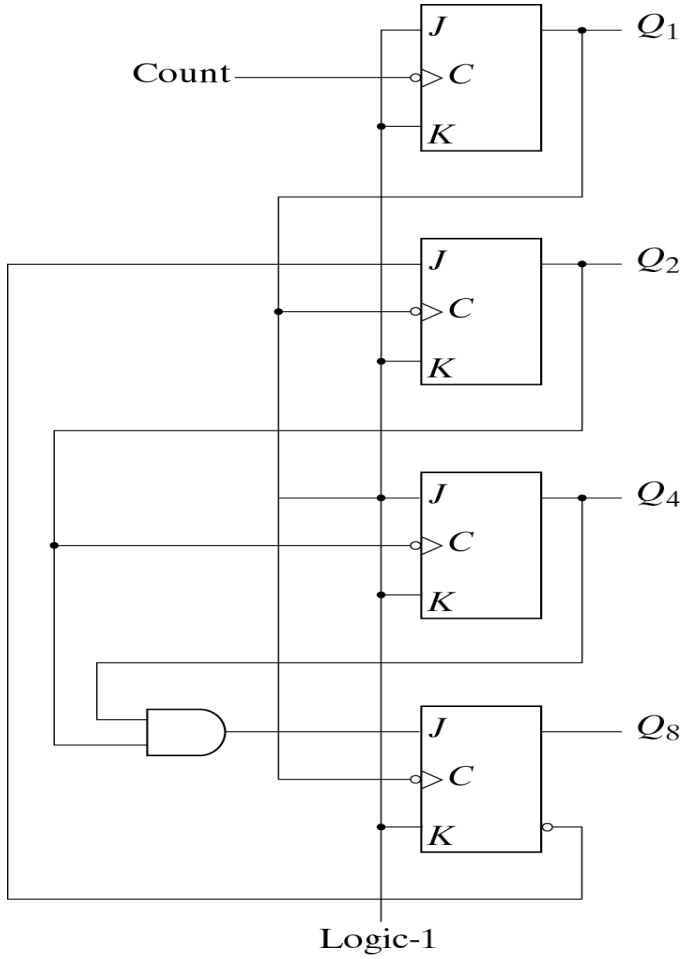


Fig. 6-10 BCD Ripple Counter

Logic Diagram of Ripple Counter

- Verification:
  - Does the circuit follow the states?
    - Q_1 is complemented with every count (J=K=1)
    - Q_2 complements if Q_1 goes from 1 to 0 and Q_8 is 0
    - Q_2 remains 0 if Q_8 becomes 1
    - Q_4 complements if Q_2 goes from 1 to 0
    - Q_8 remains 0 as long as Q_2 or Q_4 is 0
    - When Q_2 and Q_4 are 1, Q_8 complements when Q_1 goes from 1 to 0. Q_8 clears and the next Q_1 transition.

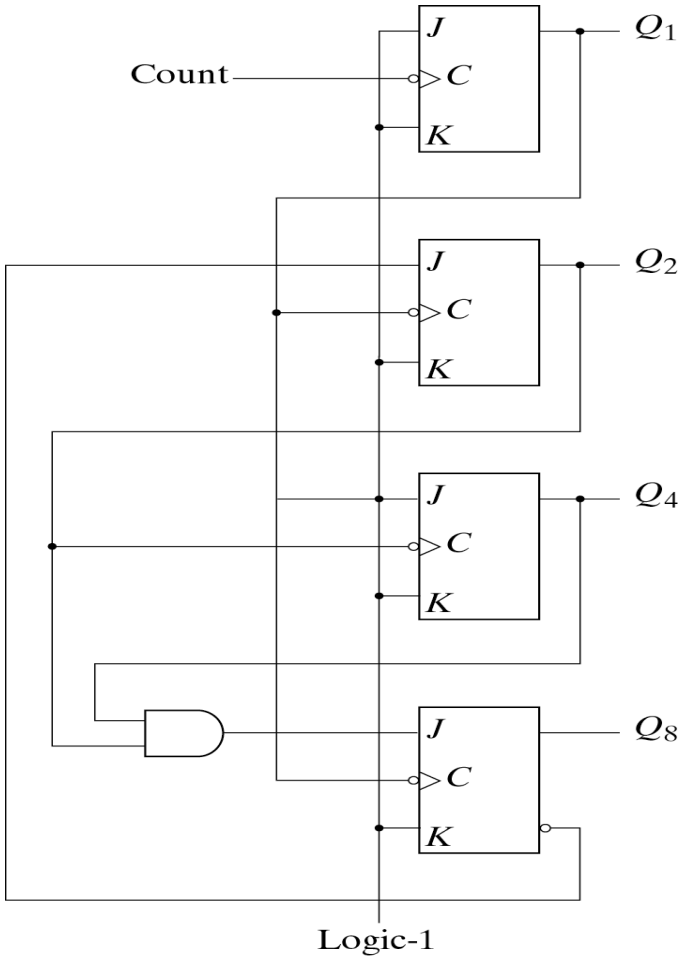


Fig. 6-10 BCD Ripple Counter

Three-Decade Decimal BCD Counter

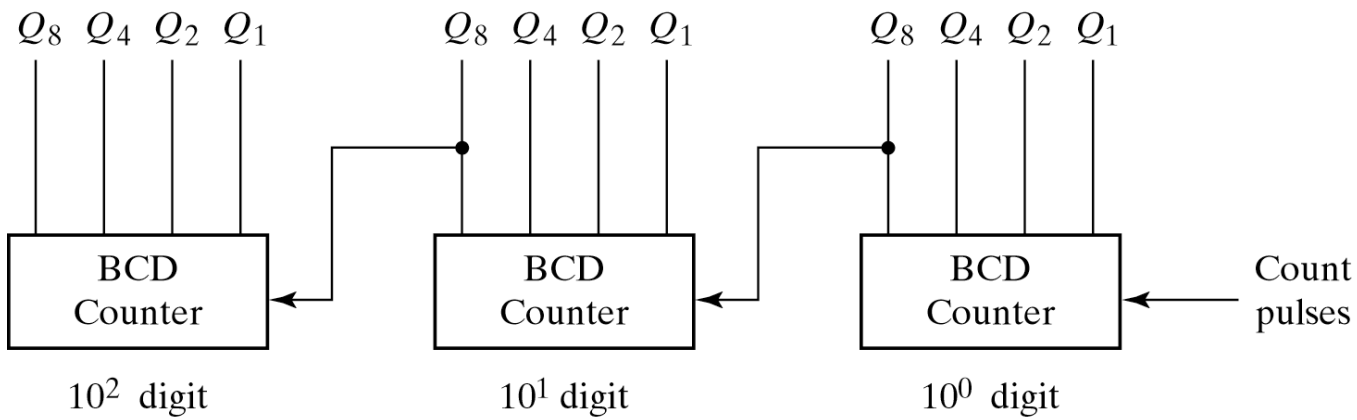


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

- Counts from 0 to 999: When Q_8 goes from 1 to 0 the next higher order decade is triggered
- 0 to 9 to 0 …

4-bit Synchronous Binary Counters

A flip-flop is complemented if all lower bits are 1.

A_3	A_2	A_1	A_0
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

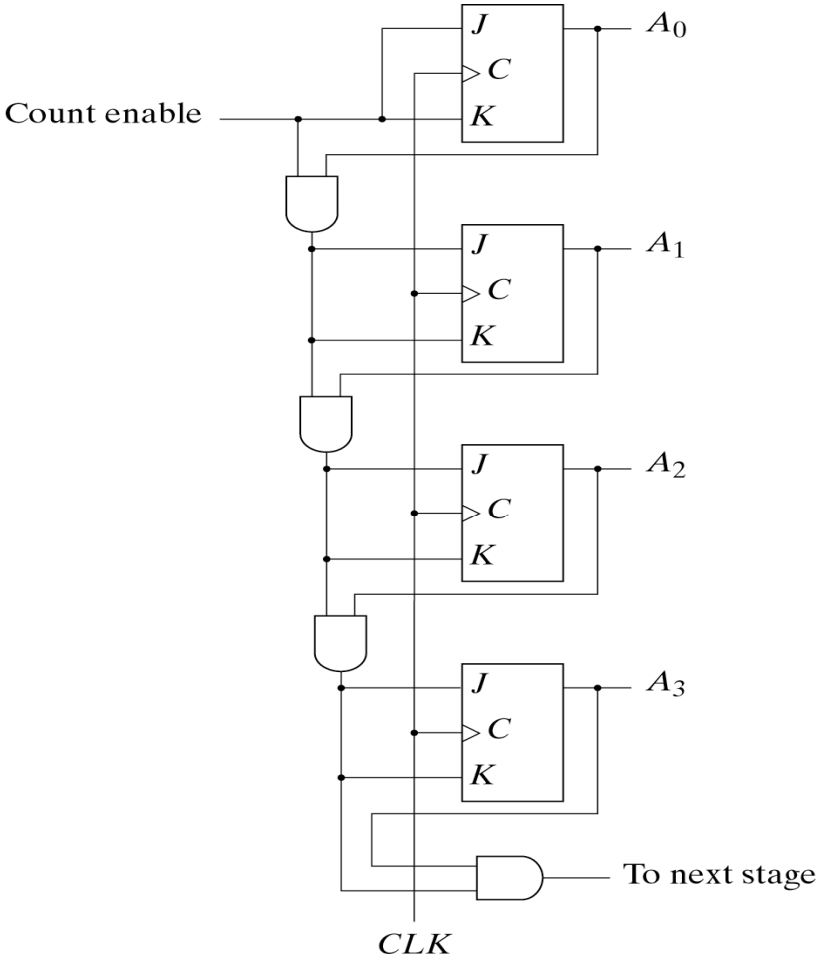


Fig. 6-12 4-Bit Synchronous Binary Counter

4-bit Up-Down Binary Counters

In a *down binary counter*,
- The least significant bit is always complemented,
- a bit is complemented if all lower bits are 0.

Change an up counter to a down counter: The AND gates should come from the complement outputs instead of the normal one

Up = 1, Down =0: Circuit counts up since input comes from Normal output

Up = 0, Down =1: Circuit counts down since input comes from Complemented output

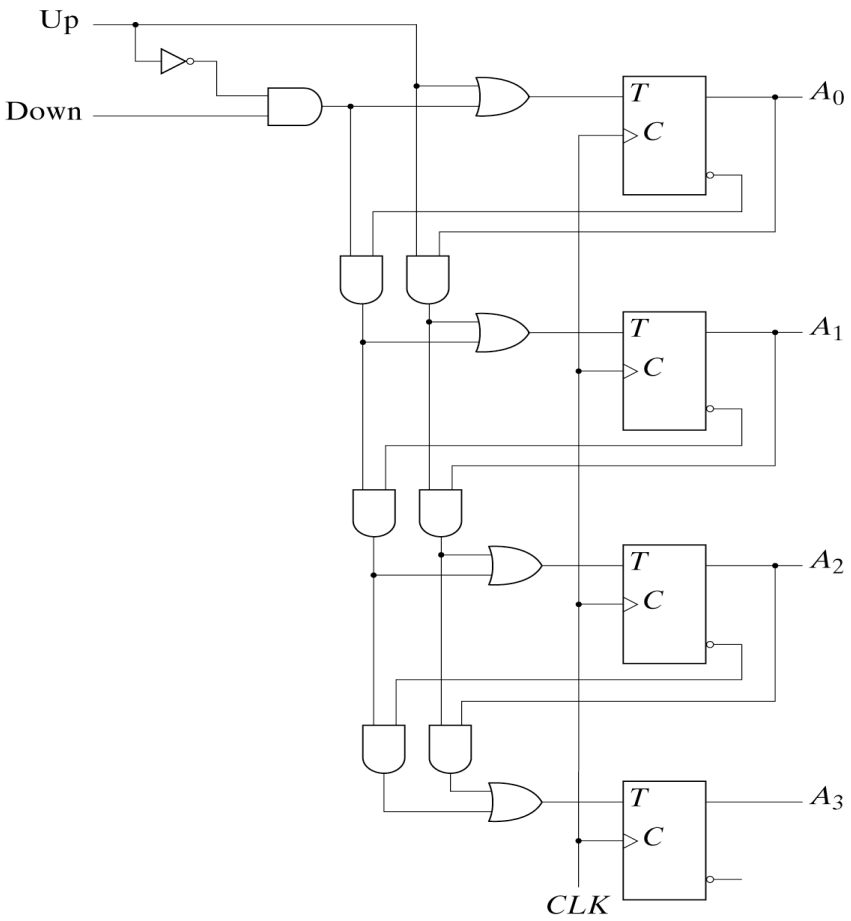


Fig. 6-13 4-Bit Up-Down Binary Counter

Binary Counter with Parallel Load

- Sometimes we need an initial value prior to counting
  - Initial value: I3 I2 I1 I0

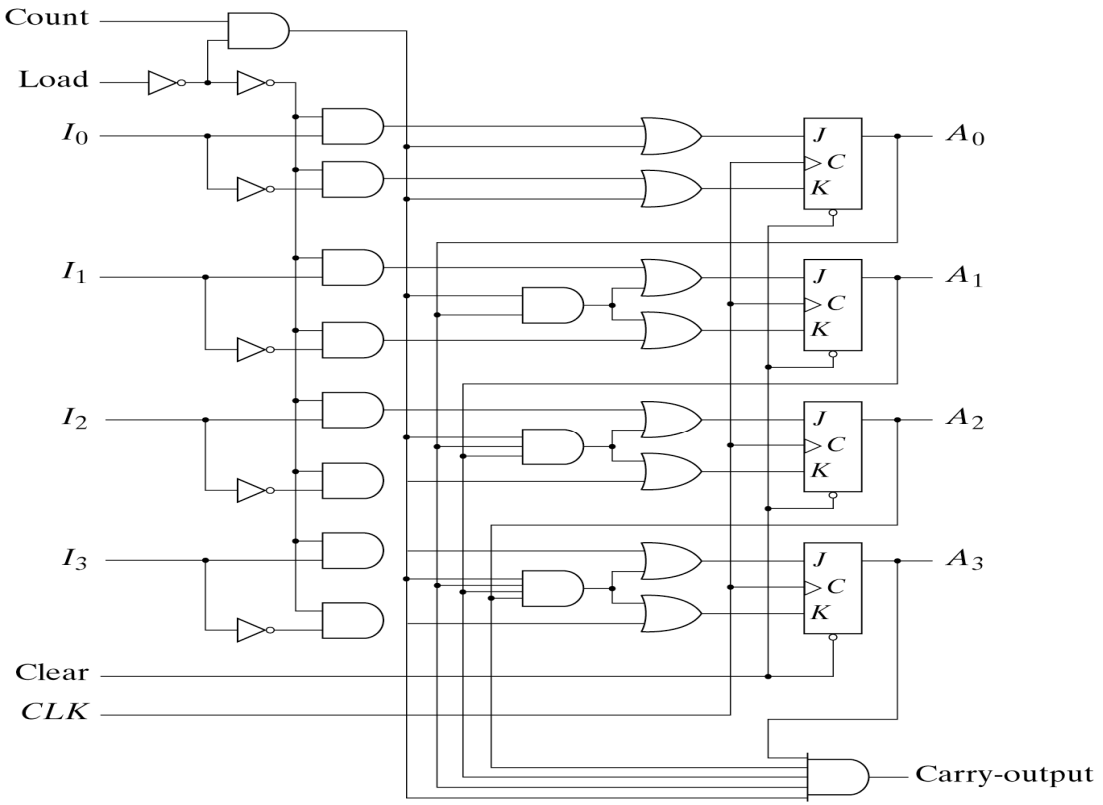


Fig. 6-14 4-Bit Binary Counter with Parallel Load

Binary Counter with Parallel Load

Count = 1, Load =0

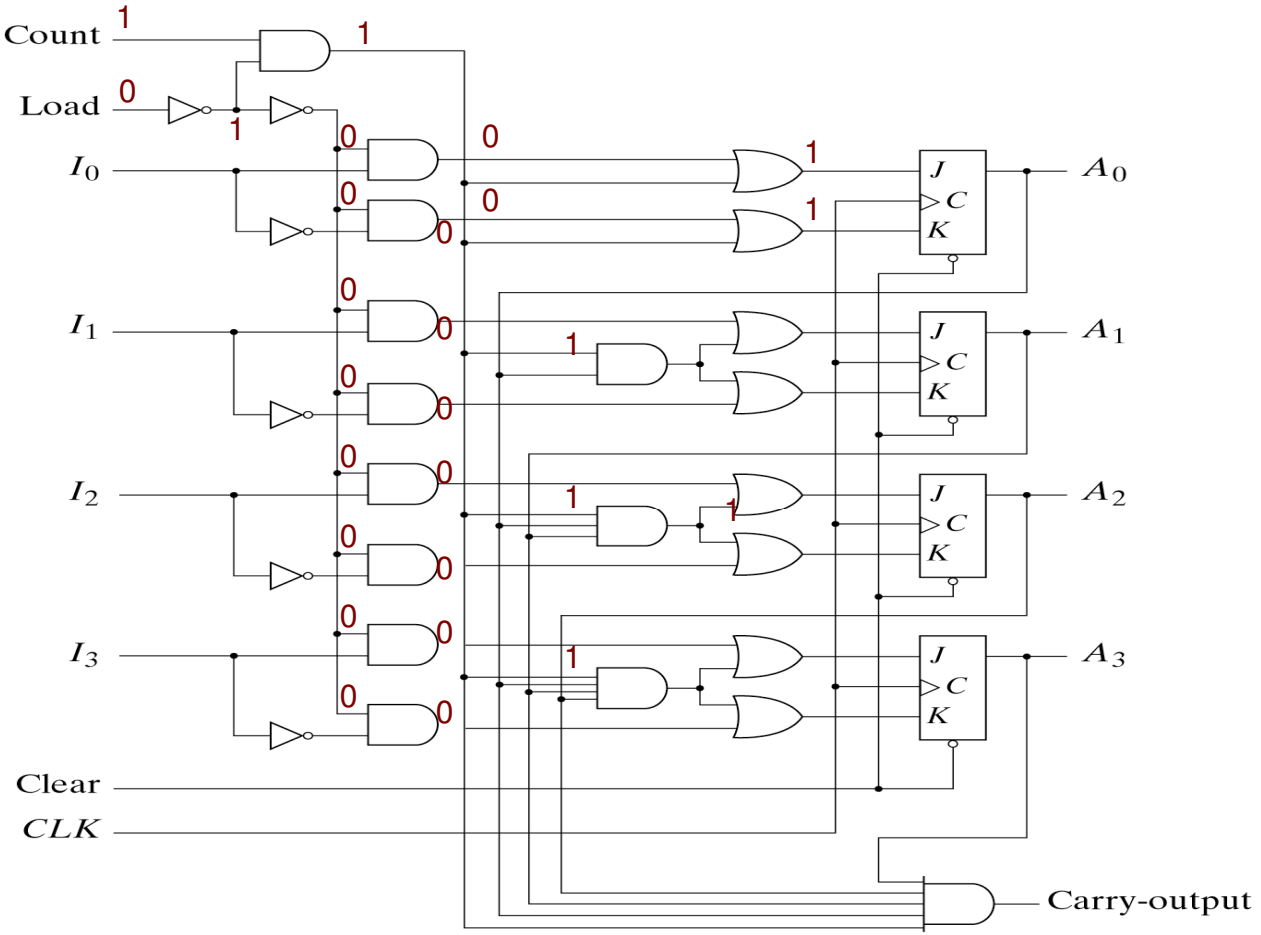


Fig. 6-14 4-Bit Binary Counter with Parallel Load

Binary Counter with Parallel Load

Count = 0, Load =1

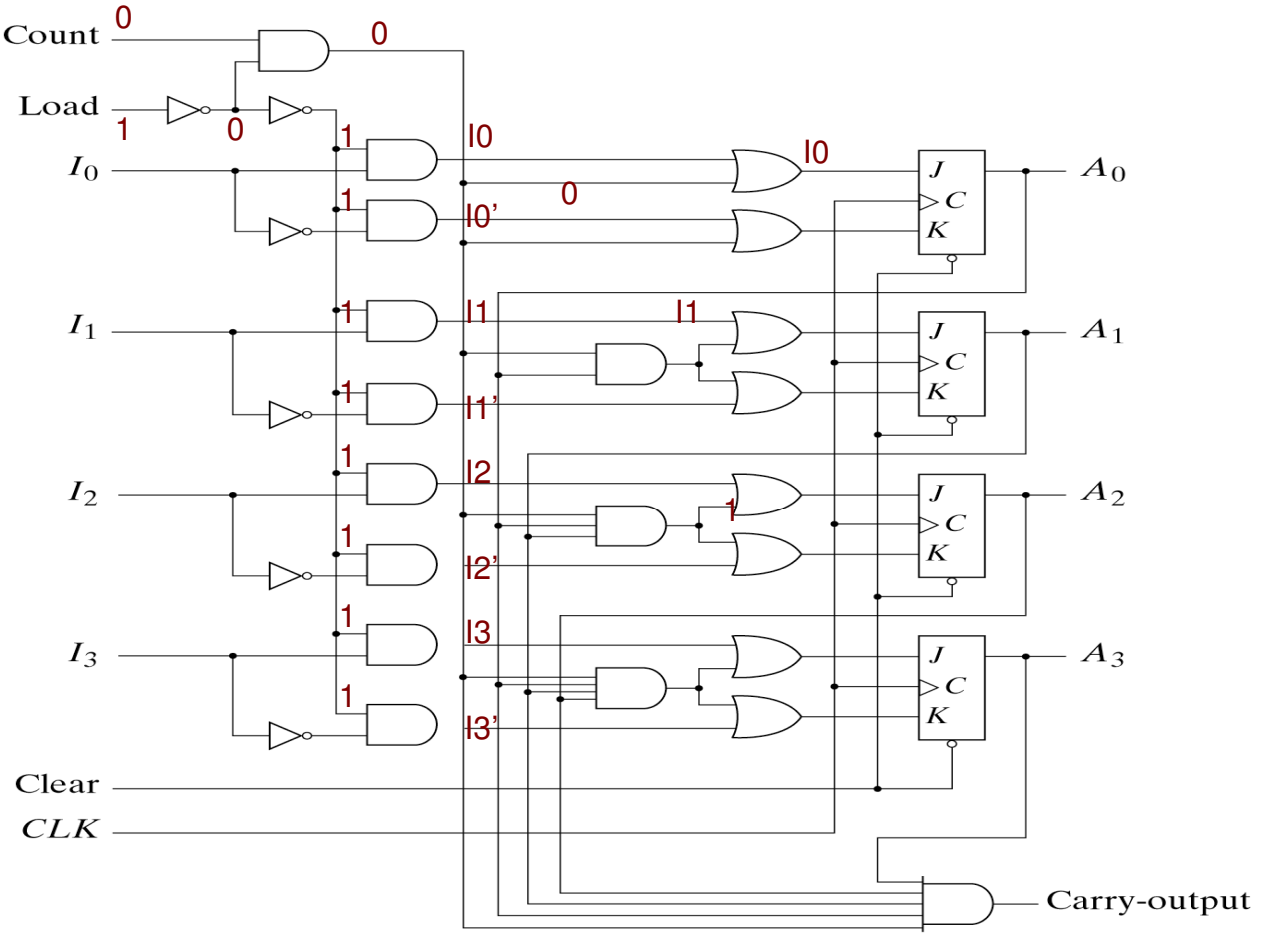


Fig. 6-14 4-Bit Binary Counter with Parallel Load

Binary Counter with Parallel Load

- In part (a), "1001" (9) is detected. In part (b), "1010" (10) is detected. Why?
- In part (a), LOAD is set to 1 and effective next cycle. (LOAD is a synchronous control input)
- In part (b), counter is immediately cleared. (Clear is an asynchronous control input)

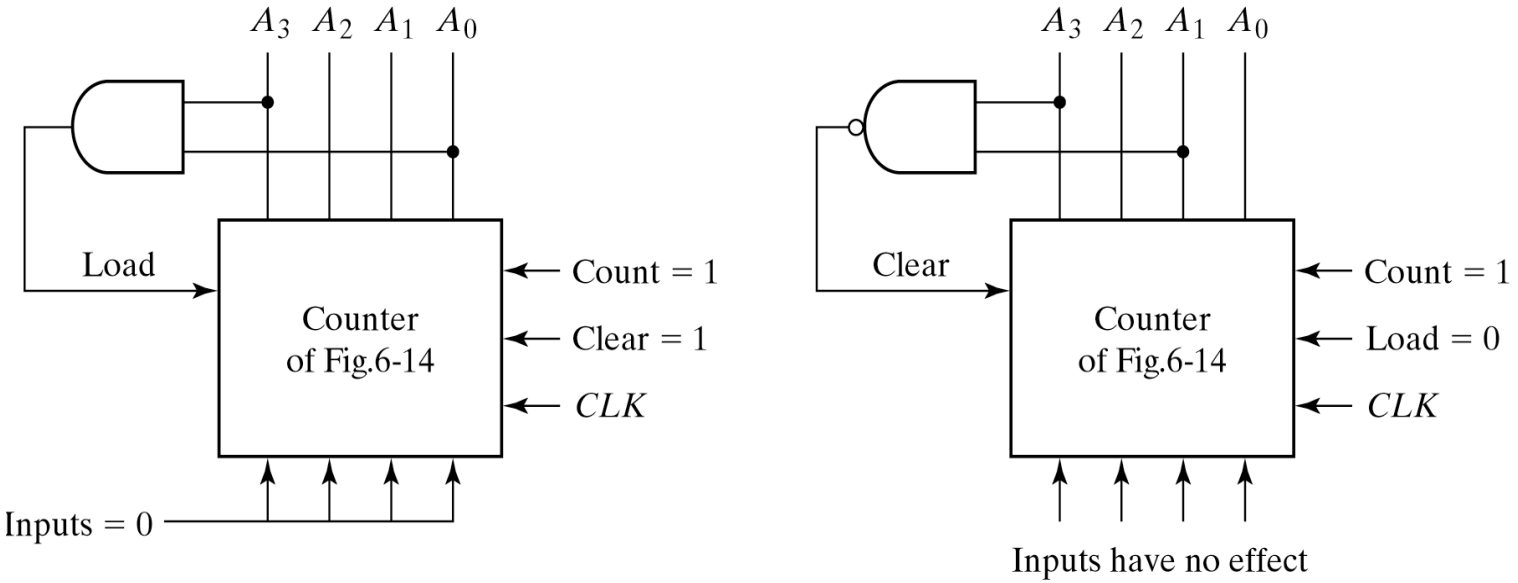
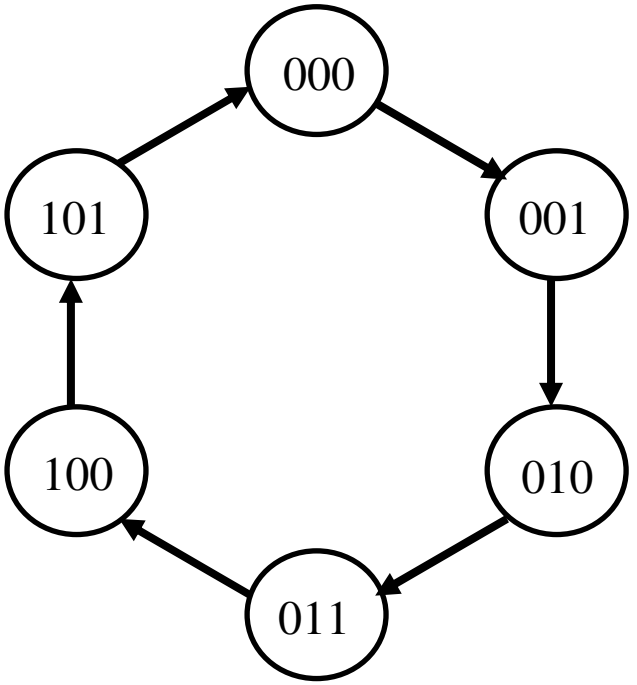


Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

Counters with unused states

- The examples shown so far have all had 2n states, and used n flip-flops. But sometimes you may have unused, leftover states.
- For example, here is a state table and diagram for a counter that repeatedly counts from 0 (000) to 5 (101).
- What should we put in the table for the two unused states?

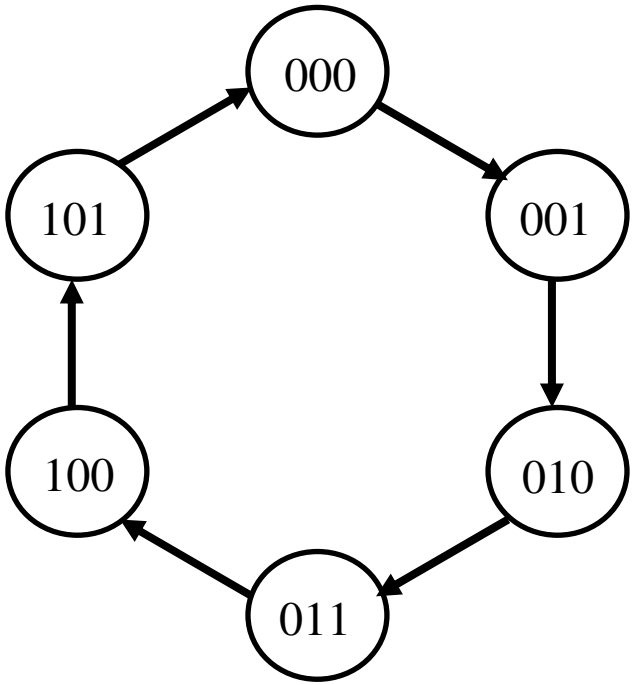
Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	?	?	?
1	1	1	?	?	?



Unused states can be don't cares…

- To get the simplest possible circuit, you can fill in don't cares for the next states. This will also result in don't cares for the flip-flop inputs, which can simplify the hardware.
- If the circuit "somehow" ends up in one of the unused states (110 or 111), its behavior will depend on exactly what the don't cares were filled in with.

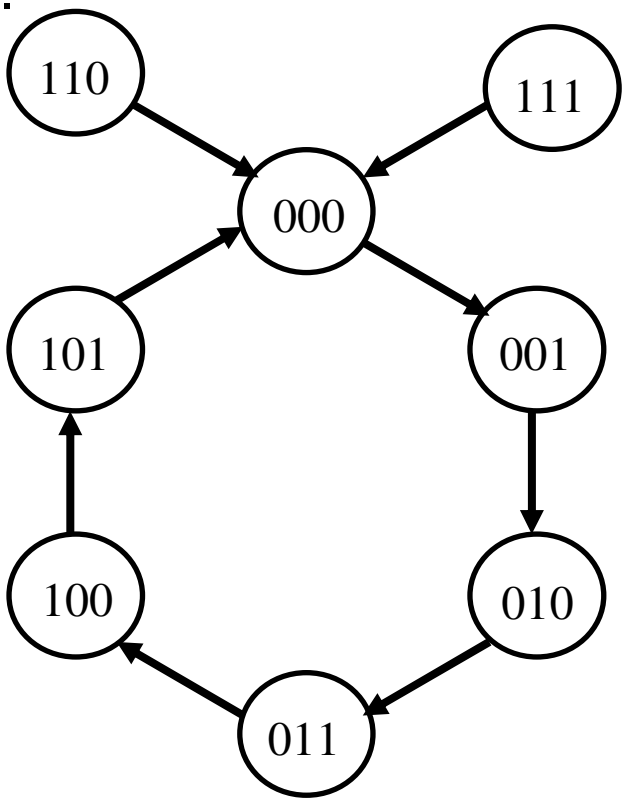
Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	x	x	x
1	1	1	x	x	x



…or maybe you do care

- To get the safest possible circuit, you can explicitly fill in next states for the unused states 110 and 111.
- This guarantees that even if the circuit somehow enters an unused state, it will eventually end up in a valid state.
- This is called a self-starting counter.

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0



Other Counters: Ring Counter

- A *ring counter* is a counter with ONLY one flip-flop set to 1 at any particular time, all others are cleared.

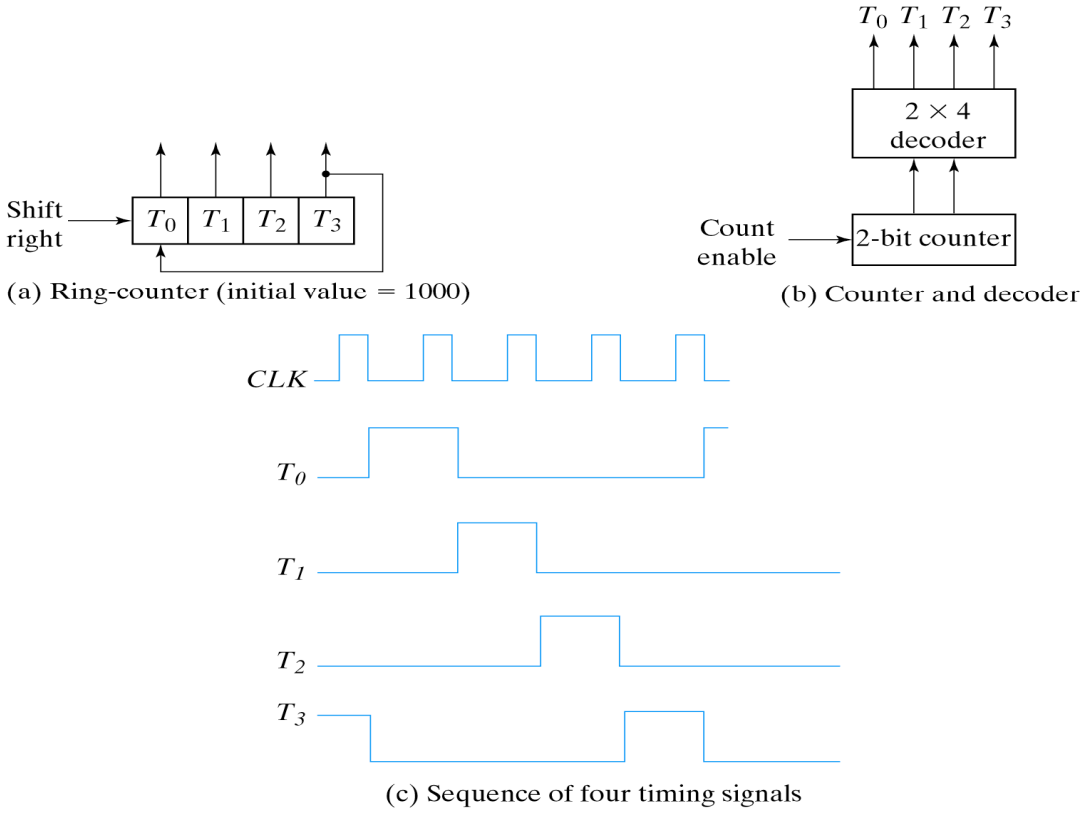
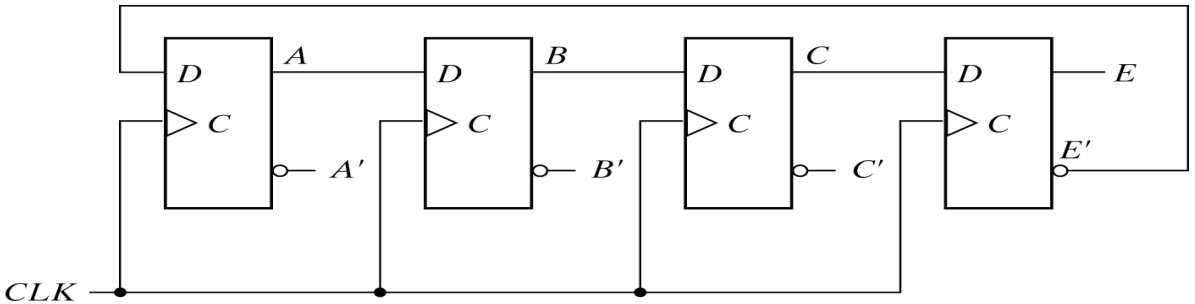


Fig. 6-17 Generation of Timing Signals

Other Counters: Johnson Counter

- A 4 flip-flop ring counter that produces 8 states (not 4).
- Where do we use it?



(a) Four-stage switch-tail ring counter

Sequence number	Flip-flop outputs				AND gate required for output
	A	B	C	E	
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

Fig. 6-18 Construction of a Johnson Counter

Registers - Summary

- A register is a special state machine that stores multiple bits of data.
- Several variations are possible:
  - Parallel loading to store data into the register.
  - Shifting the register contents either left or right.
  - Counters are considered a type of register too!
- One application of shift registers is converting between serial and parallel data.
- Registers are a central part of modern processors, as we will see in coming weeks.

Counters - Summary

- Counters serve many purposes in sequential logic design.
- There are lots of variations on the basic counter.
  - Some can increment or decrement.
  - An enable signal can be added.
  - The counter's value may be explicitly set.
- There are also several ways to make counters.
  - You can follow the sequential design principles from last week to build counters from scratch.
  - You could also modify or combine existing counter devices.