Registers and Counters

Jee-Hwan Ryu

School of Mechanical Engineering
Korea University of Technology and Education

Registers and Counters

- Circuits that include flip-flops are usually classified by the function they perform
  - Registers
  - Counters
- Register is a group of flip-flops.
- Each flip-flop is capable of storing one bit of information.
- An n-bit register consists of a group of n flip-flops.
- Register is a group of binary cells suitable for holding binary information.
- A counter is essentially a register that goes through a predetermined sequence of states.

Registers

- Clock= 0 to 1 ; Input information is transferred to output : I –> A
- Clock = 0 and 1; Output unchanged
- Clear = 0 ; Clearing the register to all 0's prior to its clocked operation.
- Clear : asynchronous input.

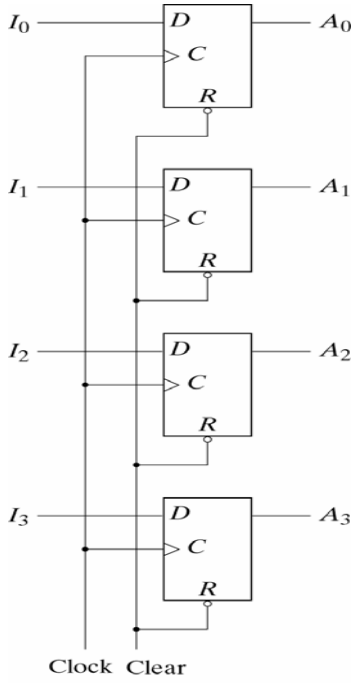


Fig. 6-1 4-Bit Register

Register with Parallel Load

- Synchronous digital systems have a master clock generator that supplies a continuous train of clock pulses.
- The transfer of new information into a register is referred to as loading the register.
- If all the bits of the register are loaded simultaneously with a common clock pulse, we say that the loading is done in parallel.
- The load input determines whether the next pulse will accept new information or leave the information in the register intact

Register with Parallel Load

- Load = 1 ; the I inputs are transferred into the register
- Load = 0 ; maintain the content of the register
- Because the D flip-flop does not have a “no change”

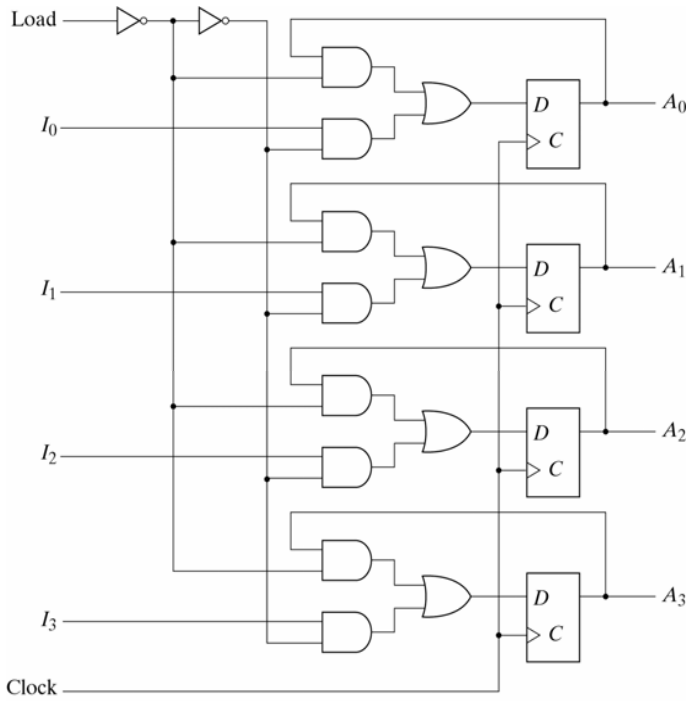


Fig. 6-2 4-Bit Register with Parallel Load

Shift Registers

- Capable of shifting its binary information in one or both directions

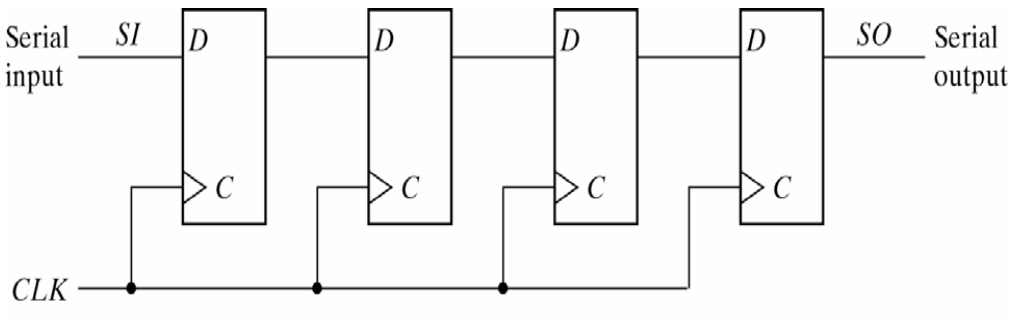


Fig. 6-3 4-Bit Shift Register

The simplest shift register

Shift Registers: Serial Transfer

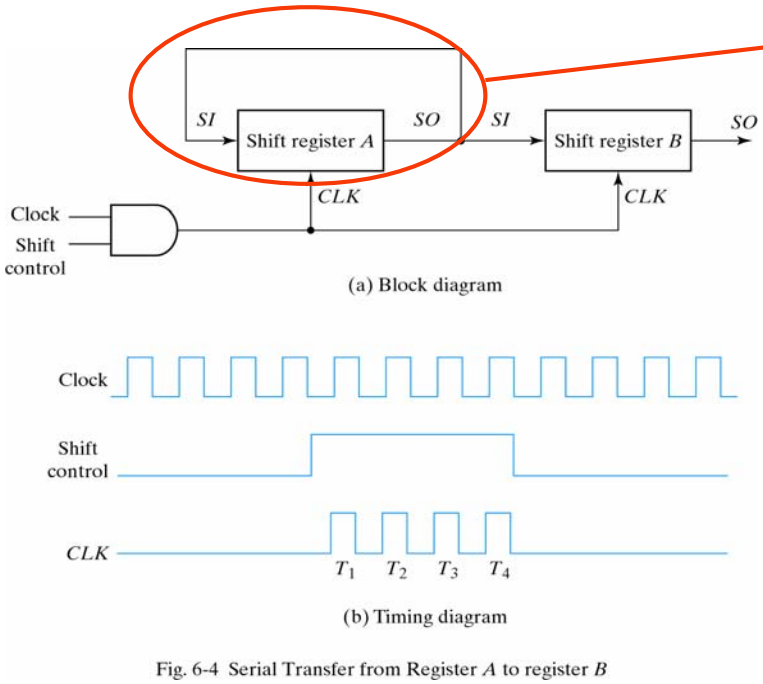


Fig. 6-4 Serial Transfer from Register *A* to register *B*

To prevent the loss of information stored in the source register

Serial-Transfer Example

Table 6-1: Serial-Transfer Example

Timing pulse	Shift register A	Shift register B	Serial output of B
Initial value	1 0 1 1	0 0 1 0	
After T_1	1 1 0 1	1 0 0 1	0
After T_2	1 1 1 0	1 1 0 0	1
After T_3	0 1 1 1	0 1 1 0	0
After T_4	1 0 1 1	1 0 1 1	0

A is transferred into B, while the content of A remains unchanged

Serial Addition

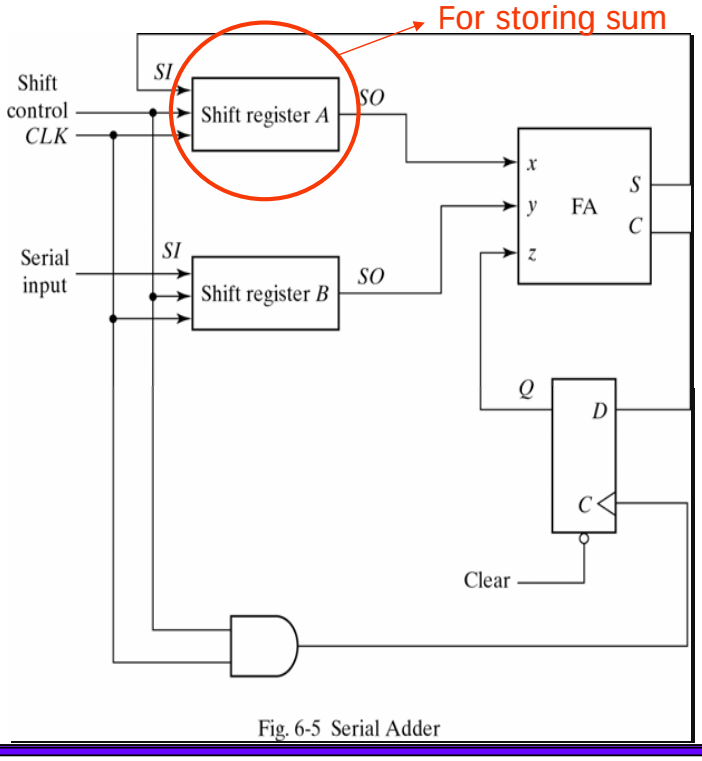


Fig. 6-5 Serial Adder

- Operation
  - The A register ->augend
  - The B register ->addend
  - Carry ->0
- The SO of A and B provide a pair of significant bits for the FA
- Output Q gives the input carry at z
- The shift-right control enables both registers and the carry flip-flop.
- The sum bit from S enters the leftmost flip-flop of A
- Parallel adder needs more circuits than serial adder

State Table for Serial Adder

Present value of carry

Output carry

Table 6-2
State Table for Serial Adder

Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	X	y	Q	S	J_Q	K_Q
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

$$J_Q = xy$$
$$K_Q = x'y' = (x + y)'$$
$$S = x \oplus y \oplus Q$$

By k-map

Second form of Serial Adder

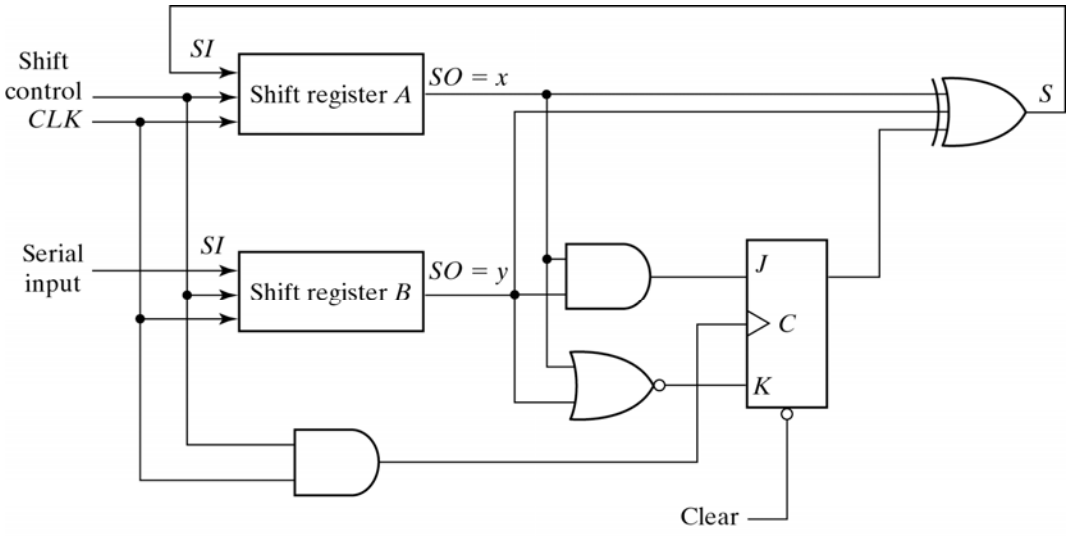


Fig. 6-6 Second form of Serial Adder

Universal Shift Register

If the register has both shifts and parallel load capabilities, it is referred to as a universal shift register.

- A clear control to clear the register to 0.
- A clock input to synchronize the operations.
- A shift-right control to enable the shift right operation and the serial input and output lines associated with the shift right.
- A shift-left control to enable the shift left operation and the serial input and output lines associated with the shift 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.

Universal Shift Register

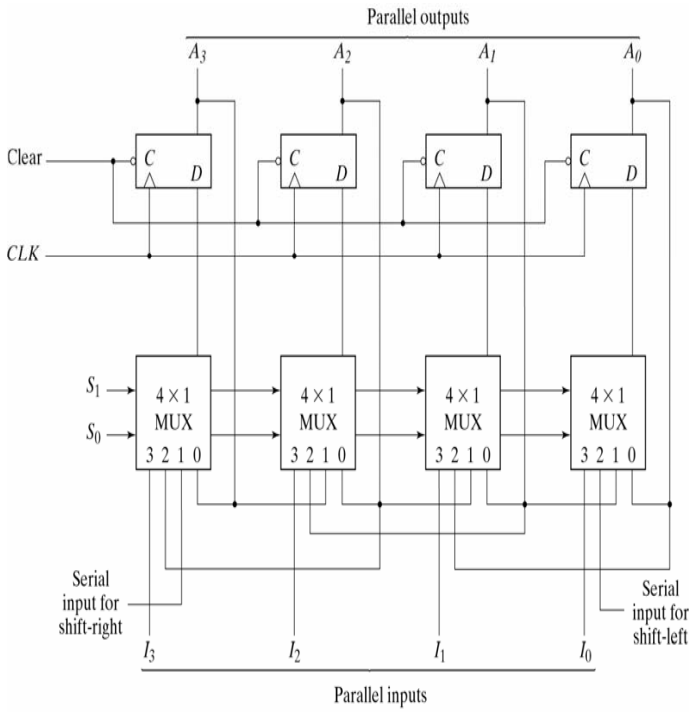


Fig. 6-7 4-Bit Universal Shift Register

- S_1, S_0 -> 0, 0 ;No change
- S_1, S_0 -> 0, 1 ; Shift right, The serial input for shift-right is transferred to the A3.
- S_1, S_0 -> 1, 0 ; Shift left, The serial input for shift-left is transferred to the A0.
- S_1, S_0 -> 1, 1 ;Parallel load

Universal Shift Register

- Shift register are often used to interface digital systems.
- Suppose it is necessary to transmit an n-bit quantity between two points.
  - It will be expensive to use n lines to transmit the n bits in parallel.
  - It is more economical to use a single line and transmit the information serially, one bit at a time.
  - The transmitter accepts the n-bit data in parallel into a shift register and then transmits the data serially along the common line.
  - The receiver accepts the data serially into a shift register.
  - When all n-bits are received, they can be taken from the outputs of the register in parallel.
- The transmitter: a parallel-to-serial conversion of data, the receiver: a serial-to-parallel conversion.

Counters

- A register that goes through a prescribed sequence of states upon the application of input pulses is called a counter.
- The input pulses may be clock pulses or they may originated from some external source and may occur at a fixed interval of time or at random.
- A counter that follows the binary number sequence is called a binary counter.
- An n-bit binary counter consists of n flip-flops and can count in binary from 0 through 2^n -1.

Counters

- Counters in two categories
  - Ripple counters
  - Synchronous counters
- Ripple counters
  - The flip-flop output transition serves as a source for triggering other flip-flops.
  - The C input some or all flip-flops are triggered not by the common clock pulses, but rather by the transition that occurs in other flip-flop outputs.
- Synchronous counters
  - The C inputs of all flip-flops receive the common clock .

Binary Ripple Counters

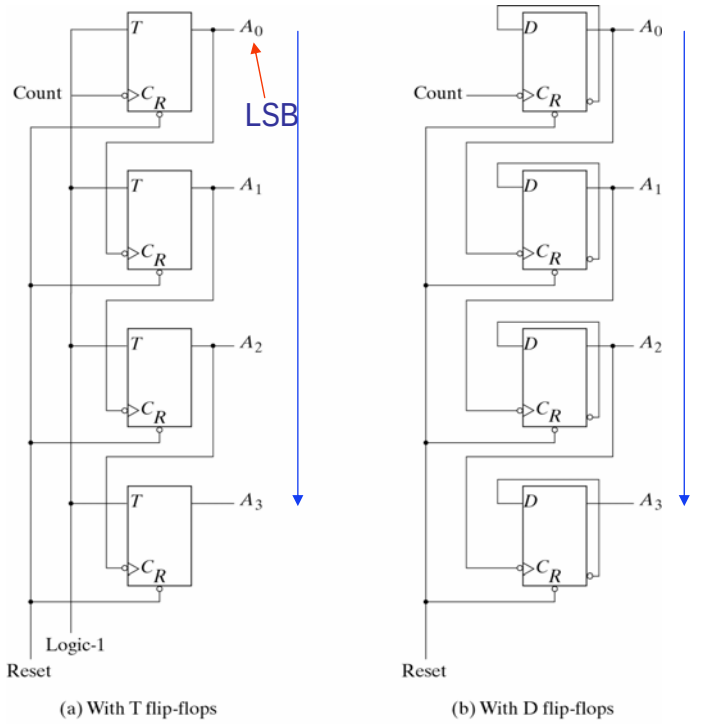


Fig. 6-8 4-Bit Binary Ripple Counter

Series connection of Complementing flip-flops

How about for Cont-down ?
-Positive edge triggered
-Or connected to comp. out

Count Sequence for a Binary Ripple Counter

Count sequence A_3 A_2 A_1 A_0		Conditions for Complementing
0 0 0 0	Complement A_0	
0 0 0 1	Complement A_0	A_0 will go from 1 to 0 and complement A_1
0 0 1 0	Complement A_0	
0 0 1 1	Complement A_0	A_0 will go from 1 to 0 and complement A_1 ;
0 1 0 0	Complement A_0	A_1will go from 1 to 0 and complement A_2
0 1 0 1	Complement A_0	
0 1 1 0	Complement A_0	A_0 will go from 1 to 0 and complement A_1
0 1 1 1	Complement A_0	
.....		
1 0 0 0	and so on...	

BCD Ripple Counter

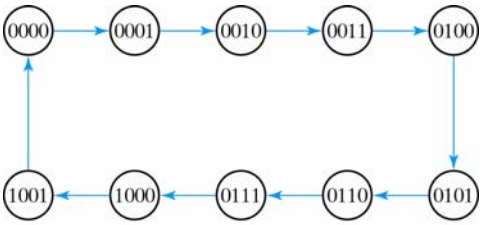


Fig. 6-9 State Diagram of a Decimal BCD-Counter

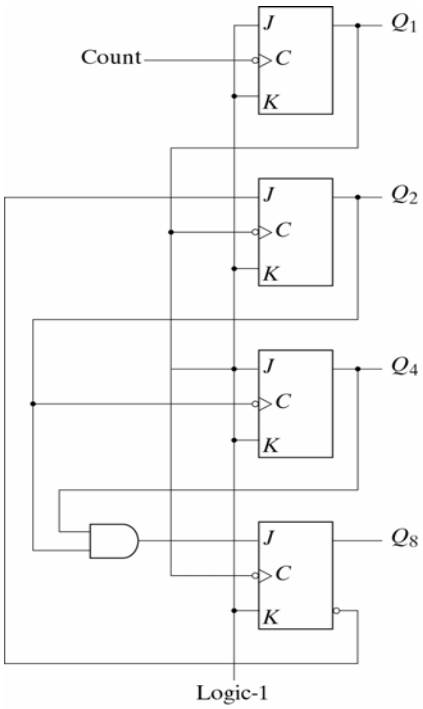


Fig. 6-10 BCD Ripple Counter

BCD Ripple Counter

1. Q1 is complemented on the negative edge of every count pulse.
2. Q2 is complemented if Q8=0 and Q1 goes from 1 to 0. Q2 is cleared if Q8=1 and Q1 goes from 1 to 0.
3. Q4 is complemented when Q2 goes from 1 to 0.
4. Q8 is complemented when Q4Q2=11 and Q1 goes from 1 to 0. Q8 is cleared if either Q4 or Q2 is 0 and Q1 goes from 1 to 0

Three-Decade Decimal BCD Counter

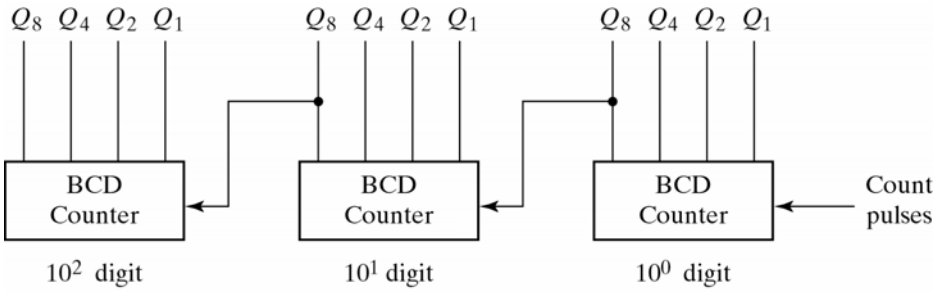


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

- To count from 0 to 999, We need a three-decade counter.
- When Q8 in one decade goes from 1 to 0, it triggers the count

Synchronous Counters

- Clock pulses are applied to the inputs of all flip-flops
- A common clock triggers all flip-flops simultaneously rather than one at a time as in a ripple counter

Synchronous Binary Counter

- The first stage A0 has its J and K equal to 1 if the counter is enabled .
- The other J and K inputs are equal to 1 if all previous low-order bits are equal to 1 and the count is enabled.
- A flip-flop in any other position is complemented when all the bits in the lower significant positions are equal to 1
  - A3A2A1A0=0011 -> 0100

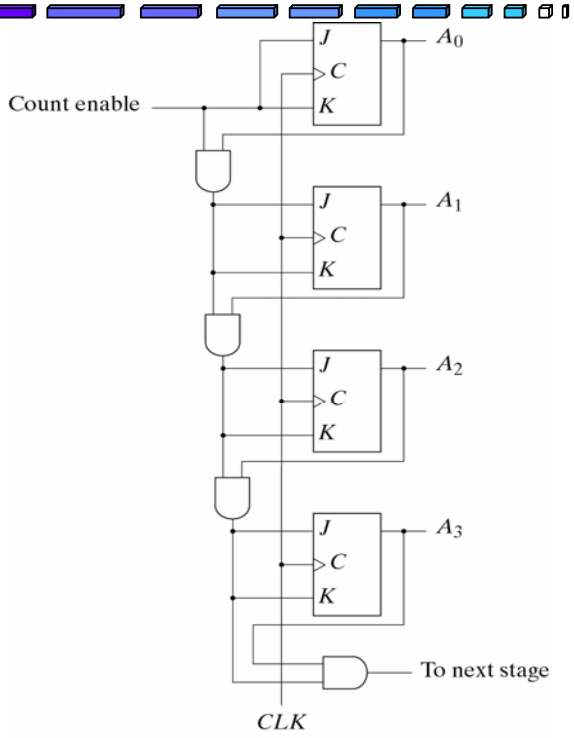


Fig. 6-12 4-Bit Synchronous Binary Counter

Up-Down Binary Counter

- Up input control=1 ;count up (the T inputs receive their signals from the values of the previous normal outputs of the flip-flops.)
- Down input control=1, up input control=0 ; count down
- Up=down=0 ;unchanged state
- Up=down=1 ;count up

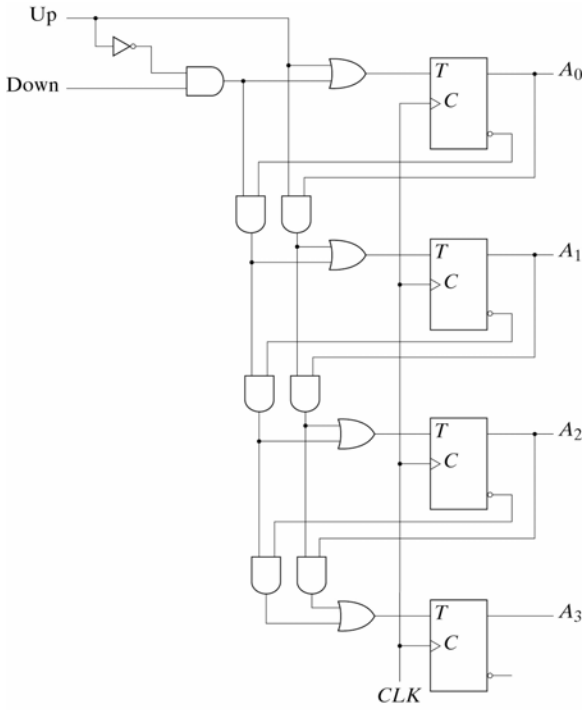


Fig. 6-13 4-Bit Up-Down Binary Counter

BCD Counter

Table 6-5
State Table for 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	TQ_8	TQ_4	TQ_2	TQ_1
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

$$T_{Q1} = 1$$
$$T_{Q2} = Q'_8 Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

Example

- Design a synchronous BCD counter with J-K flip-flops.

Binary Counter with Parallel Load

- Parallel load for initial number
- Input **load** control=1; disables the count sequence, data transfer
- **Load** =0 and **count**=1 ;count
- **Load**=0 and **count**=0 ;unchanged
- **Carry out**=1 (all flip-flop=1 and Count=1)

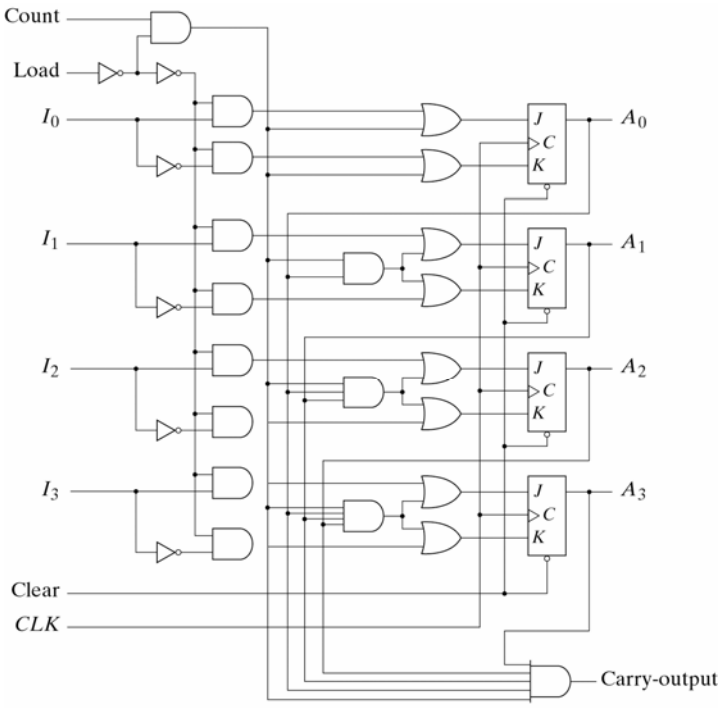


Fig. 6-14 4-Bit Binary Counter with Parallel Load

BCD Counter using Binary Counter with Parallel Load

- The **AND** gate detects the occurrence of state **1001(9)** in the output. In this state, the load input is enabled and all-**0**'s input is loaded into register.
- The **NAND** gate detects the count of **1010(10)** , as soon as this count occurs the register is **cleared.**
- A momentary **spike** occurs in output A2 as the count goes from **1001** to **1010** and immediately to **0000**

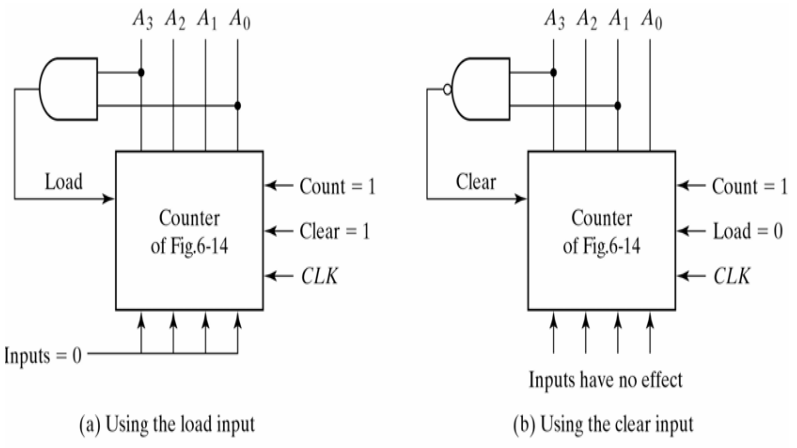
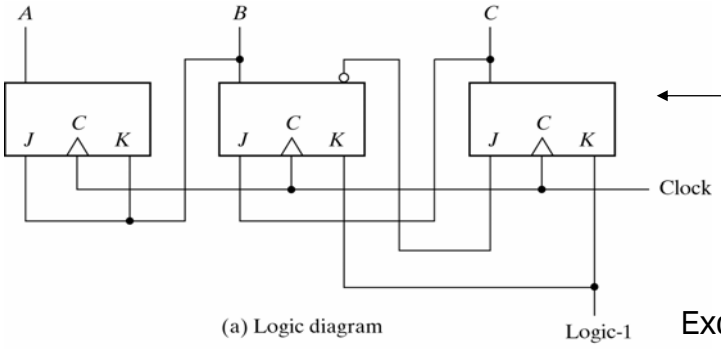


Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

Other Counters



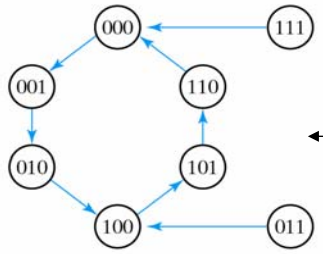
(a) Logic diagram

$J_A = B$ $K_A = B$

$J_B = C$ $K_B = 1$

$J_C = B'$ $K_C = 1$

Except **011 ,111**



(b) State diagram

Fig. 6-16 Counter with Unused States

Table 6-7
State Table for Counter

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

Example

- Design a counter with the following repeated binary sequence: 0, 1, 2, 4, 6. Use D flip-flops

Ring Counter

- For generating timing signal that control the sequence of operations
- A circular shift register with only one flip-flop being set at any particular time; all others are cleared.
- The single bit is shifted from one flip-flop to the other.
- 2^n timing signals need 2^n flip-flops

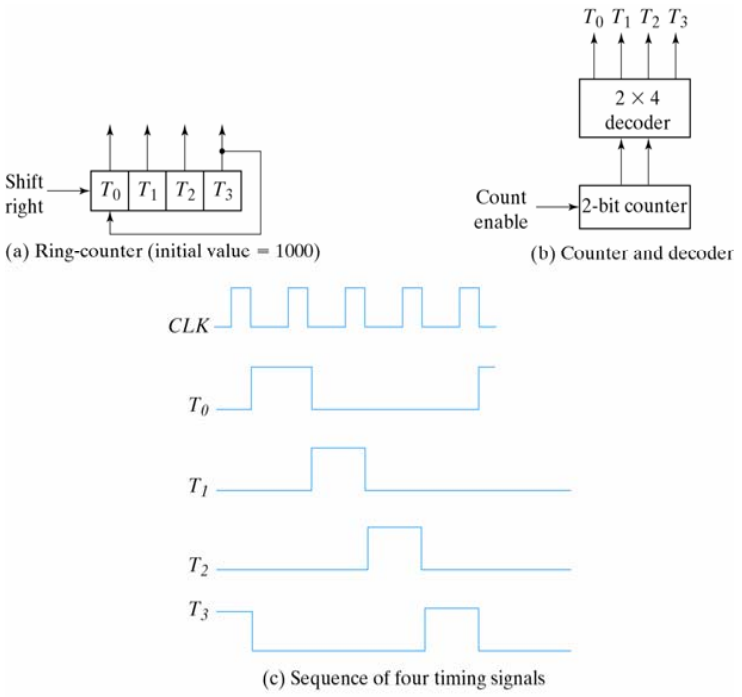
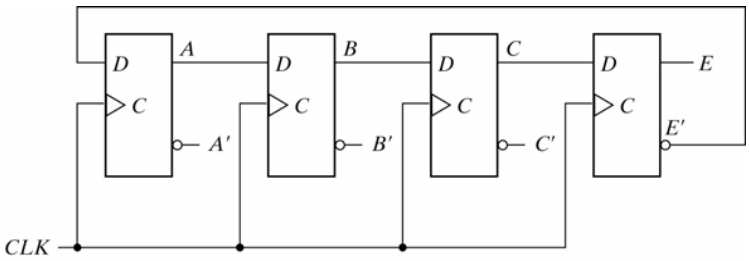


Fig. 6-17 Generation of Timing Signals

Johnson Counter

- A circular shift register with the complement output of the last flip-flop connected to the input of the first flip-flop.
- A k-bit switch-tail ring counter will go through a sequence of 2k states.
- A Johnson counter is a k-bit switch-tail ring counter with 2k decoding gates to provide outputs for 2k timing signals.



(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

Fig. 6-18 Construction of a Johnson Counter

Memory Decoding

- The equivalent logic of a binary cell that stores one bit of information
- The binary cell stores one bit in its internal flip-flop
- It has three inputs and one output. The read/write input determines the cell operation when it is selected.

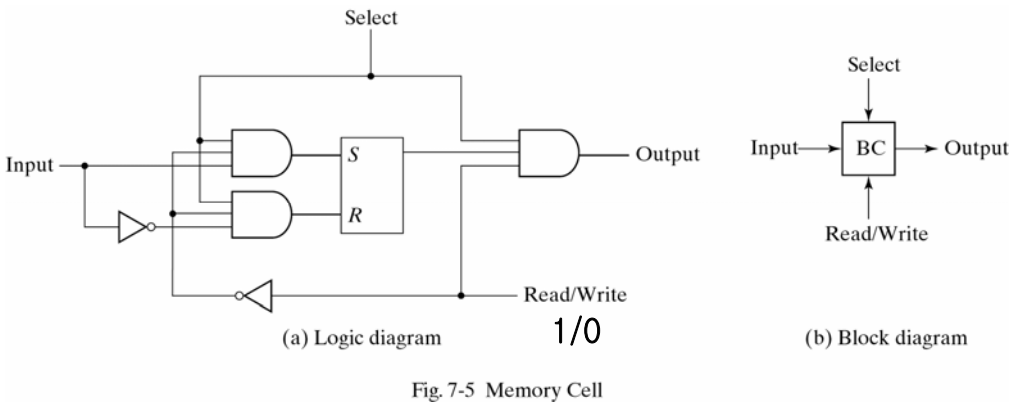
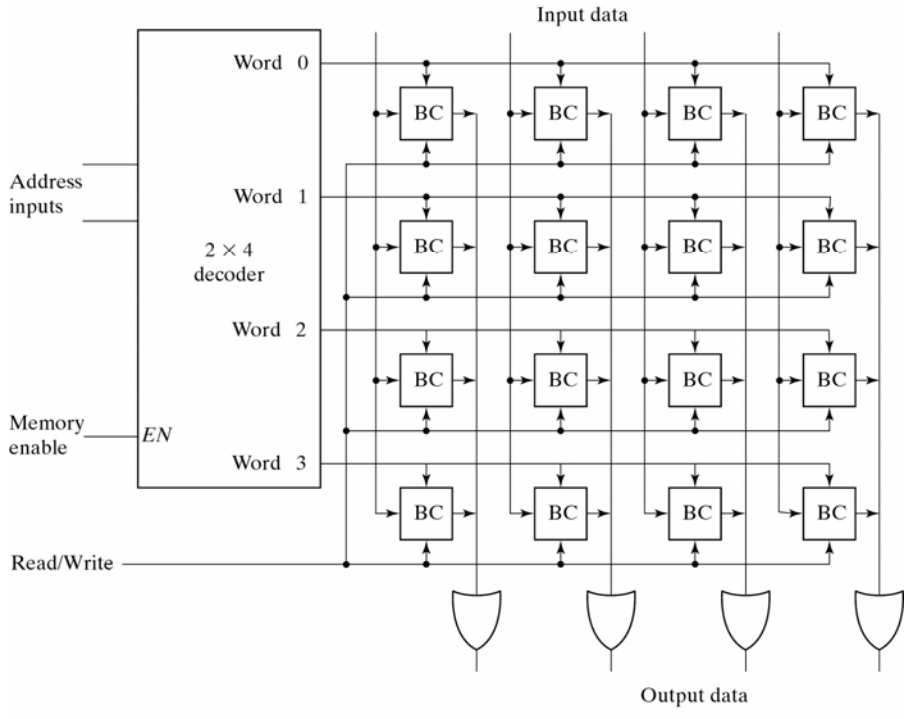


Fig. 7-5 Memory Cell

Internal Construction



4 words of 4 bits
=16 binary cells

Fig. 7-6 Diagram of a 4 × 4 RAM

Read-Only Memory

ROM = AND gates connected as a decoder + a number of OR gates

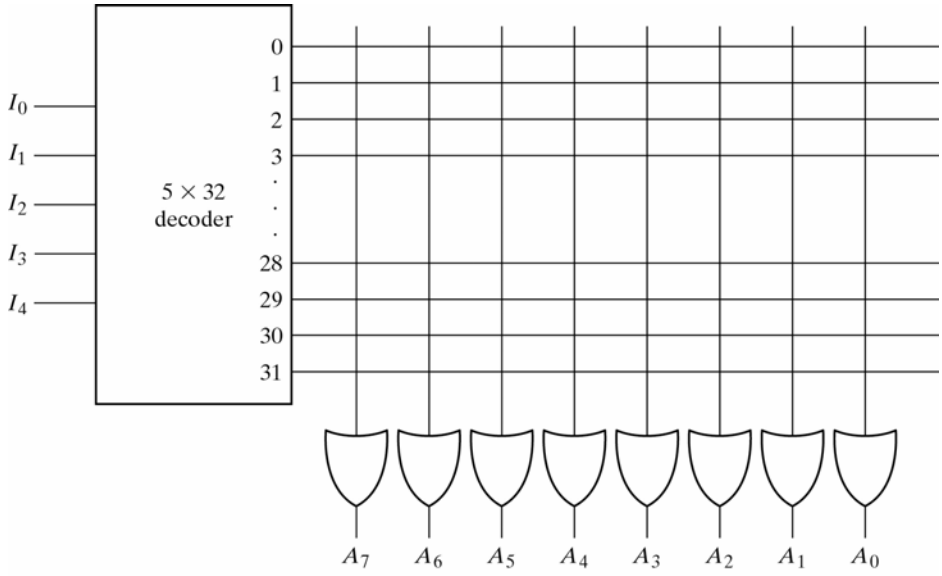


Fig. 7-10 Internal Logic of a 32 × 8 ROM

ROM Truth Table (Partial)

Table 7-3
ROM Truth Table (Partial)

Inputs					Outputs							
I4	**I3**	**I2**	**I1**	**I0**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮					⋮					
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

Combinational Circuit Implementation

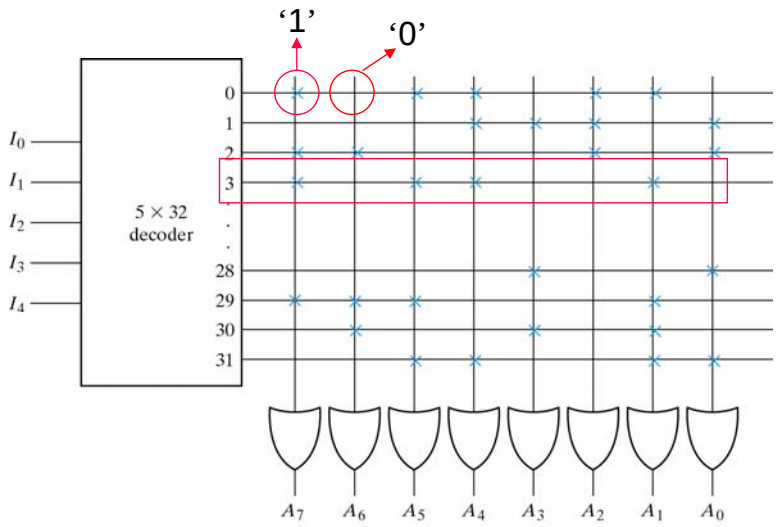


Fig. 7-11 Programming the ROM According to Table 7-3

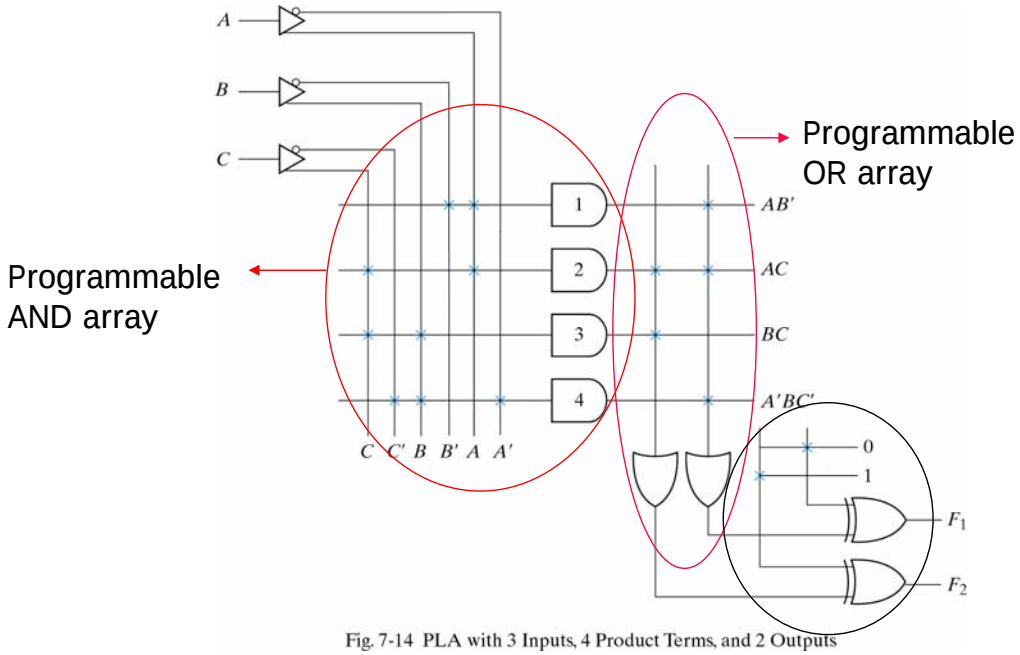
$A_7(I_4,I_3,I_2,I_0)$=Sum of minterms(0,2,3,…,29)

Input->00011(3)

Others-> all '0'

Output->10110010

Programmable LOGIC ARRAY



Programmable OR array

Programmable AND array

Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs