8 - registers and counters

register

group of flip-flopss
each storing one bit of information

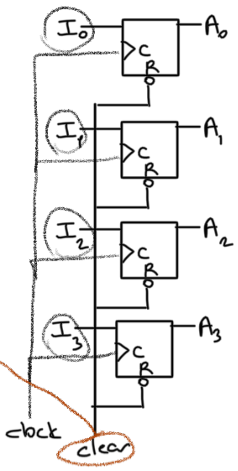
- registers are used as tenporary storage in a processor → faster than main memory

register with paralel load=

load=1 → we load the data

load=0 → register content does not change

4 bit register →



clear: when it is 1, all the flip-flops are reset to 0.

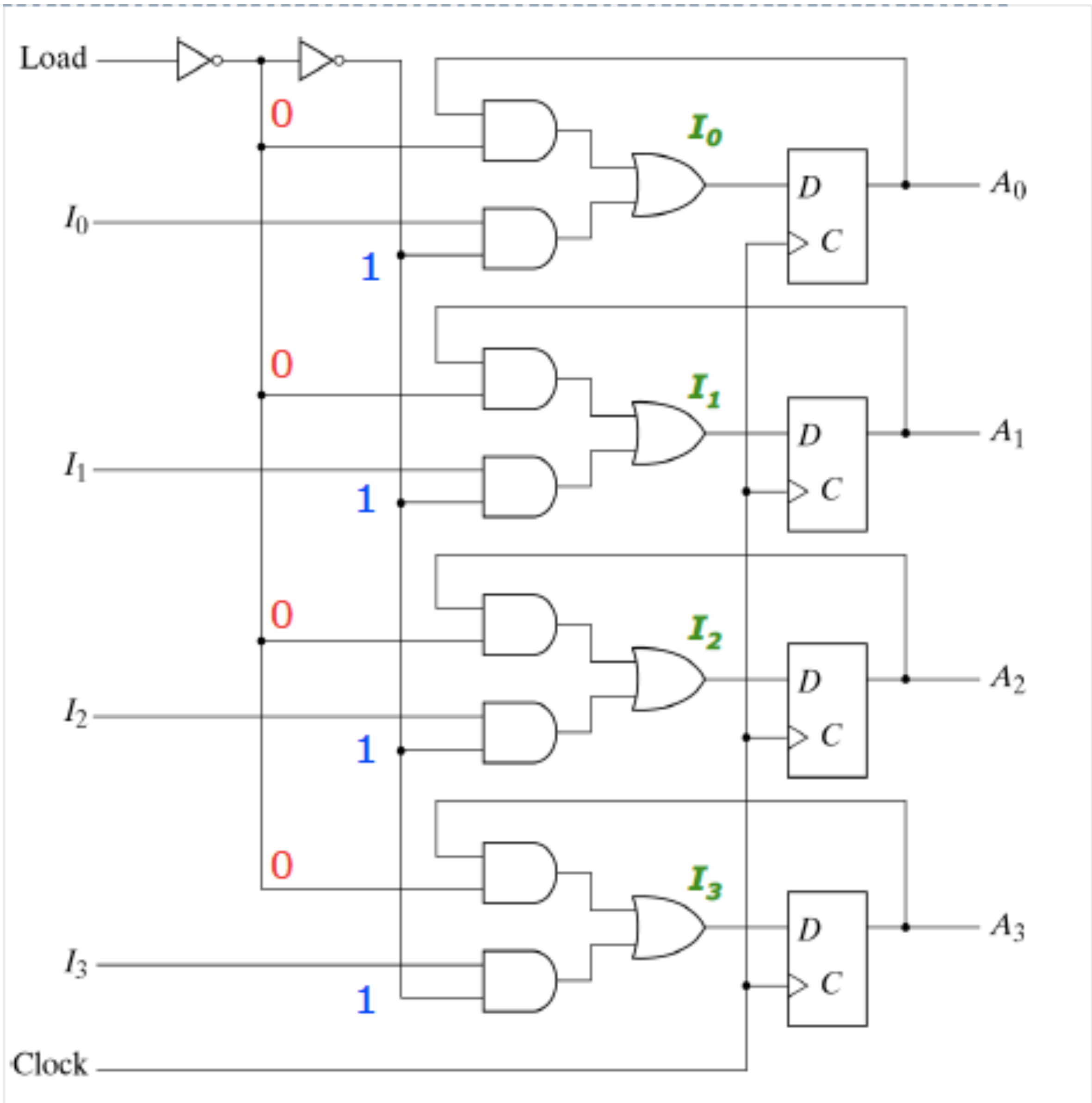
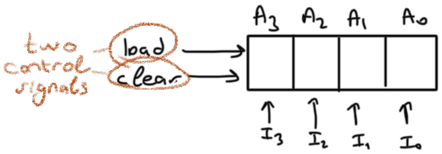


Fig. 6-2 4-Bit Register with Parallel Load



load =1 → register will receive data

clear =1 → register will be cleared to all 0

- register can act as the memory component of the sequential circuit

parallel registers = the input is loaded to the register in a single clock cycle
the number of bits that are loaded does not matter

shift (serial) registers = shifts its binary information in one or both direction

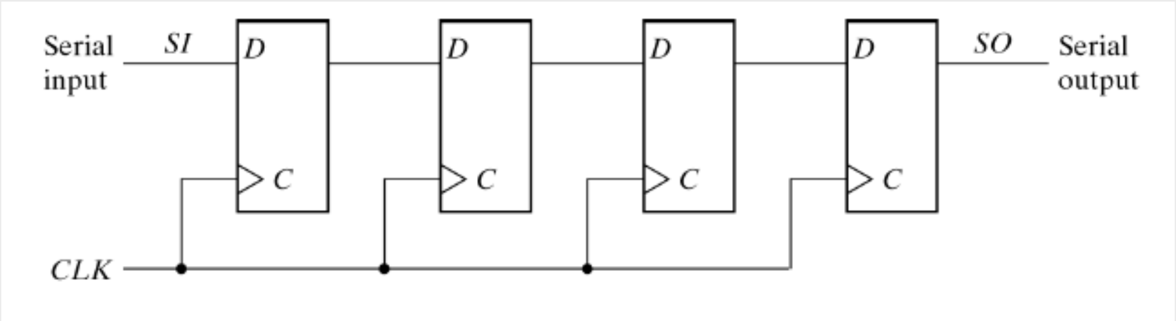
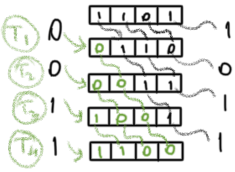


Fig. 6-3 4-Bit Shift Register

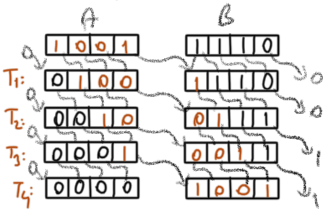
↳ right shift register

loading 1100 to 1101 register:



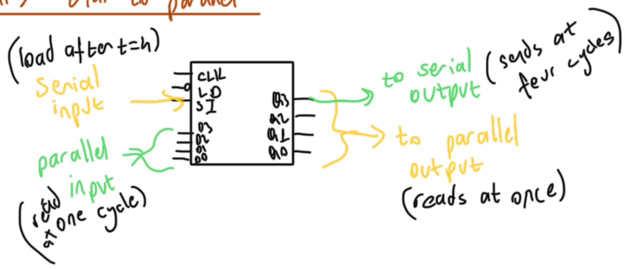
↳ 4 clock cycle

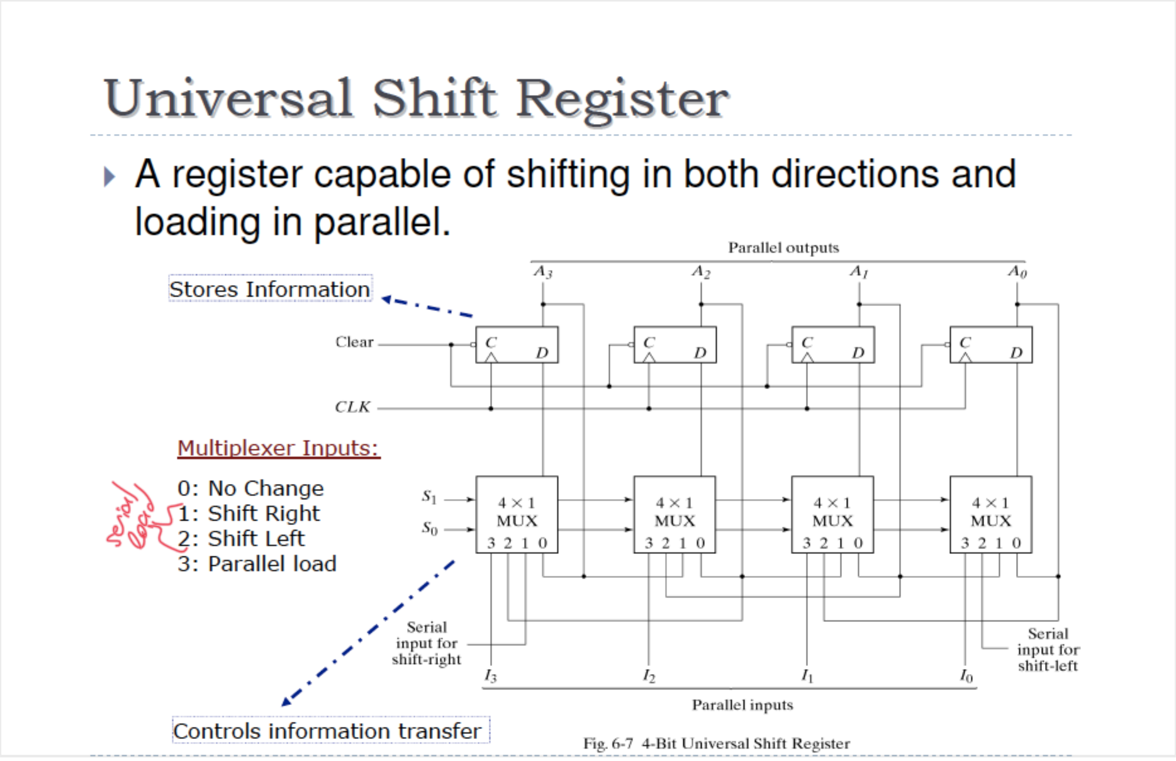
serial data transfer:



used in → keyboards, mice (input)
→ printers (output)
→ USB or Firewire (data transfer)
:

parallel to serial / serial to parallel





Universal Shift Register

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

Stores Information

Multiplexer Inputs:

0: No Change
1: Shift Right
2: Shift Left
3: Parallel load

serial load

Controls information transfer

Fig. 6-7 4-Bit Universal Shift Register

parallel adder

n adders

one unit of time

(combinational circuit)

serial adder

one full adder

n units of time

(sequential circuit)

counters

a register that can go through a predetermined sequence of states (basically counts)

binary counter = counts through binary sequence, n bit counter counts from 0 to 2^n

counters

(wave)
ripple conters

the flip-flop output triggers other flip-flops in sequence

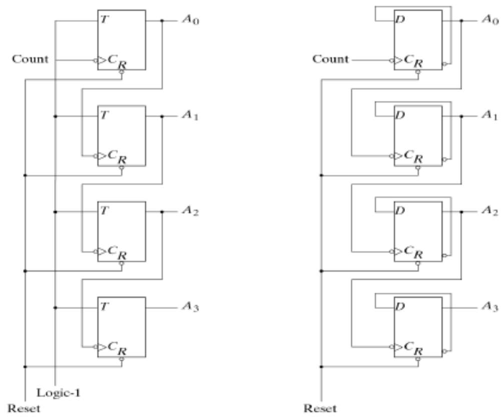
synchronous counters

count the clock

usage of counters:

- act as a simple clock → keep track of absolute/ relative timing
- record how many times something has happened

→ all processors contain a program counter, or PC → keeps track of instructions



(a) With T flip-flops

(b) With D flip-flops

Fig. 6-8 4-Bit Binary Ripple Counter

binary ripple counter: consists of a series of complementing flip-flops, with the output of each flip-flop connected to the next higher order

☆no clocks!

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
1	0	0	1

→ A_0 is complenented at each count pulse

→ A_1 is complemented when A_0 goes from 1 to 0

→ A_2 is complenented when A_1 goes from 1 to 0

→ A_3 is complenented when A_2 goes from 1 to 0

count-up counter

☆ in count-down counter, the bits change when previous bits go from 0 to 1

binary coded decimal →

0 → 0000
1 → 0001
⋮
9 → 1001

10 → 0001 0000
11 → 0001 0001
⋮

(down-count is complemented when all lower bits are 0)

Synchronous binary counter = when clock is 1 and all lower bits are one, it is complemented. (when JK=11 $Q_n \rightarrow Q_n'$)

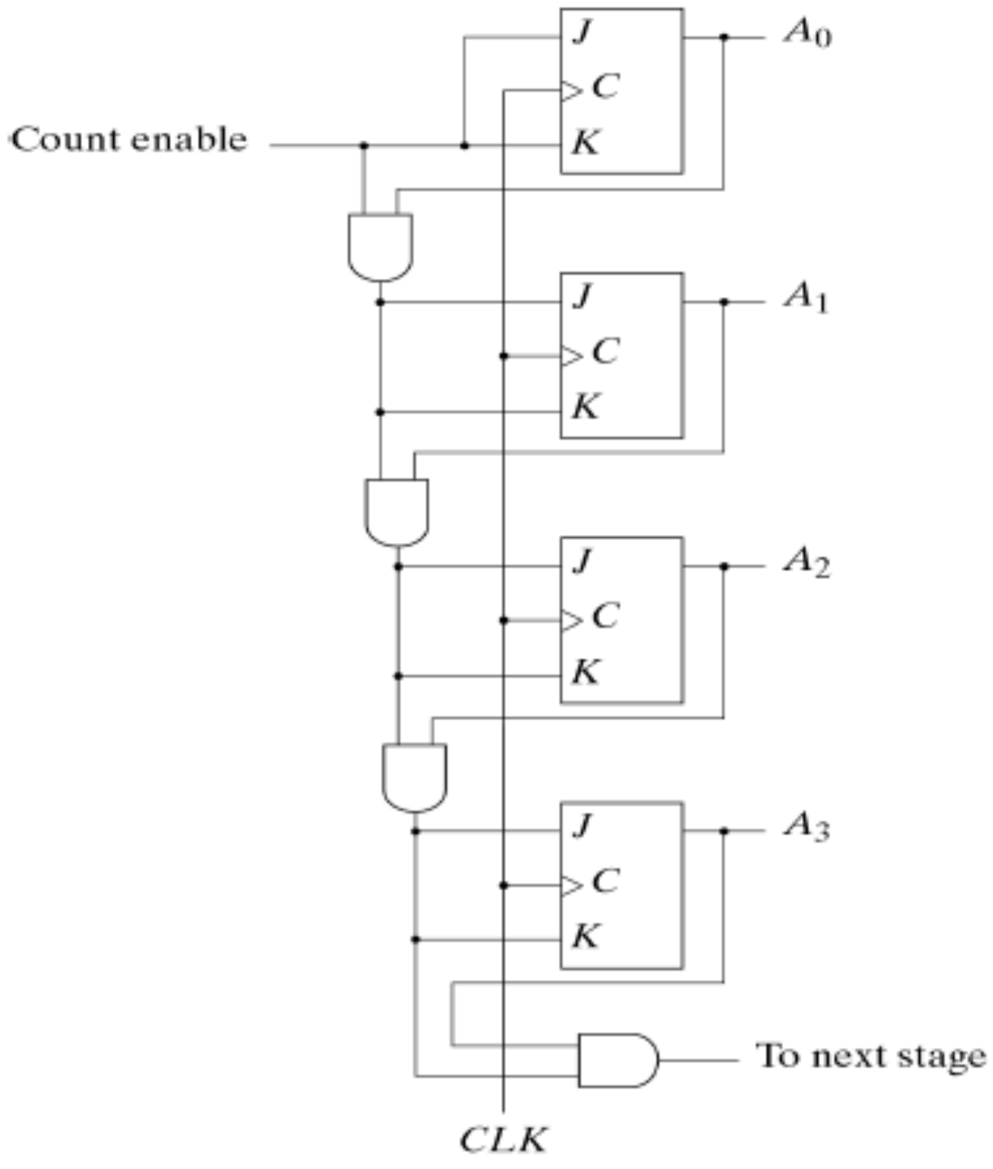


Fig. 6-12 4-Bit Synchronous Binary Counter

☆ if we need initial value to count, we can add parallel load feature

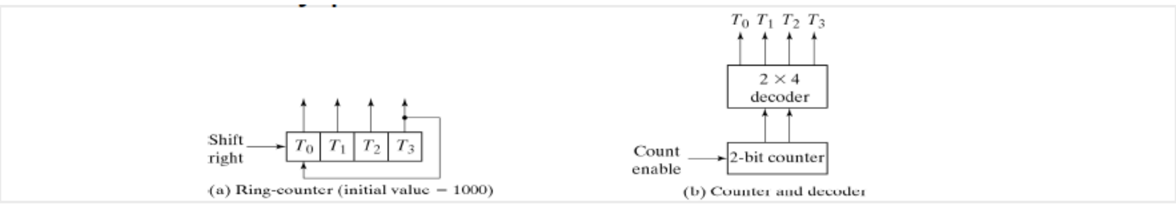
other counters:

↳ ring counter: 0001
0010
0100
1000

counts only one flip-flop set to 1

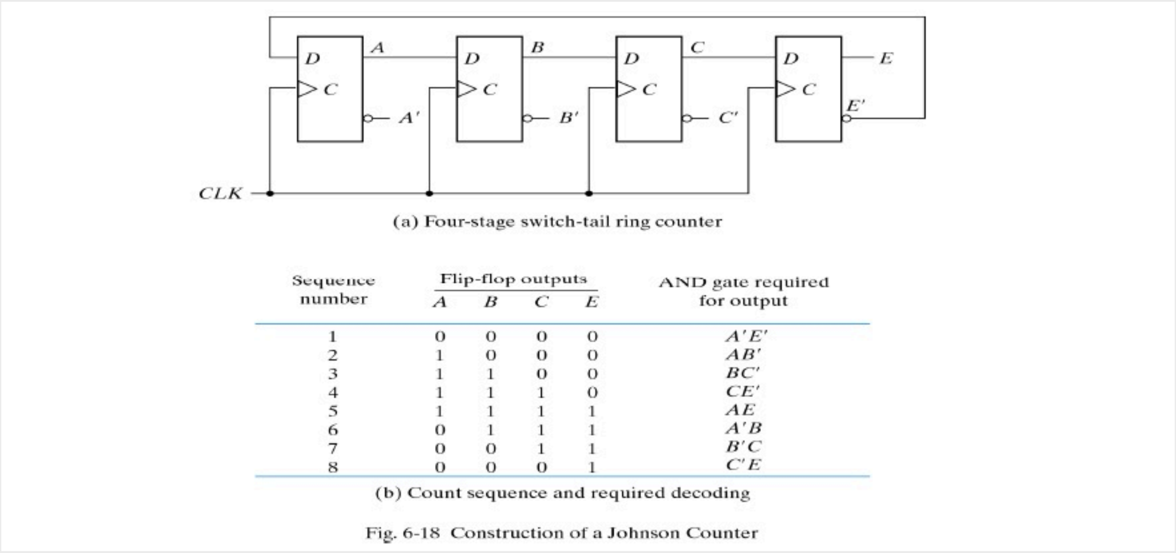
↳ Johnson counter: 0000
1000
1100
1110
1111
0111
0011
0001

they have same number of flip-flops (4) but, Johnson can count twice more then ring.



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

(b) Counter and decoder



(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

for ring counter:

(D = Q(next))

present				next							
Q_1	Q_2	Q_3	Q_4	Q_1	Q_2	Q_3	Q_4	D_3	D_2	D_1	D_0
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	0	1	0	0
0	1	0	0	1	0	0	0	$\boxed{1}$	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1

$D_3 = Q_1' . Q_2 . Q_3' . Q_4'$

Synchronous = there is a clock, all change at the same time

asynchronous = no clock, they change indepently of each other