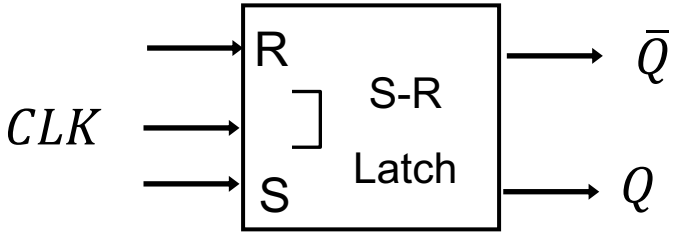
Digital Logic Design

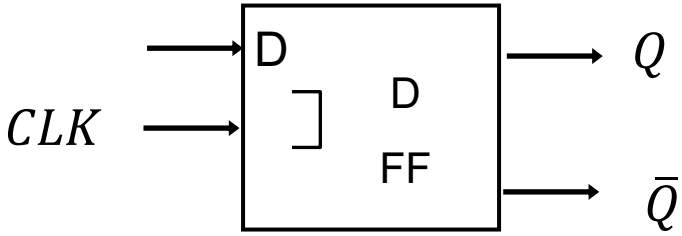
Hajar Falahati

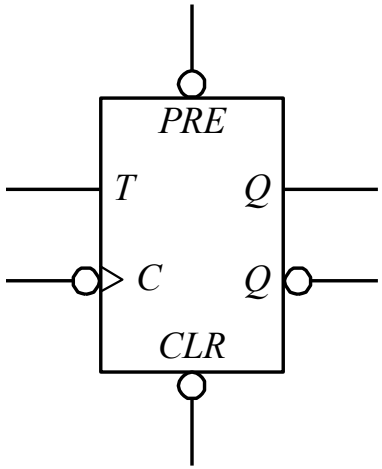
Department of Computer Engineering
IRAN University of Science and Technology

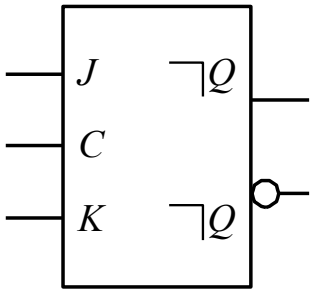
hfalahati@iust.ac.ir

Memory Unit









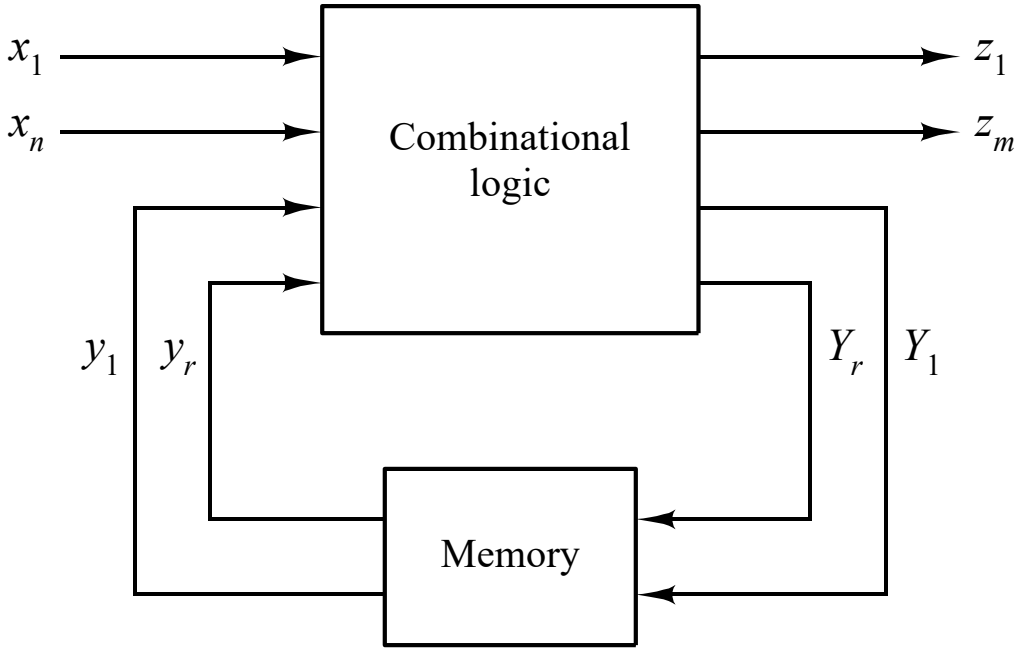
Outline

- Sequential Logics
  - Types
- Sequential Modules
  - Register
  - Counter

Modular Sequential Logics

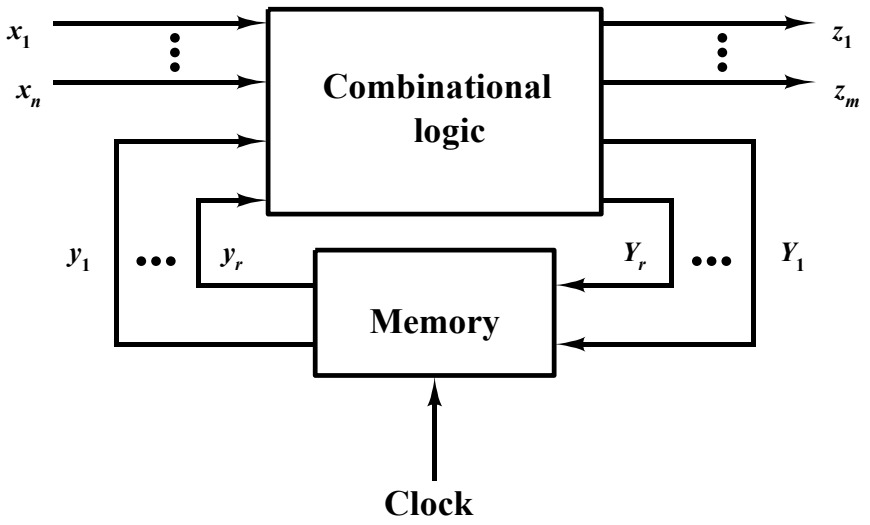
Sequential Logics

- Types
  - Synchronous
  - Asynchronous



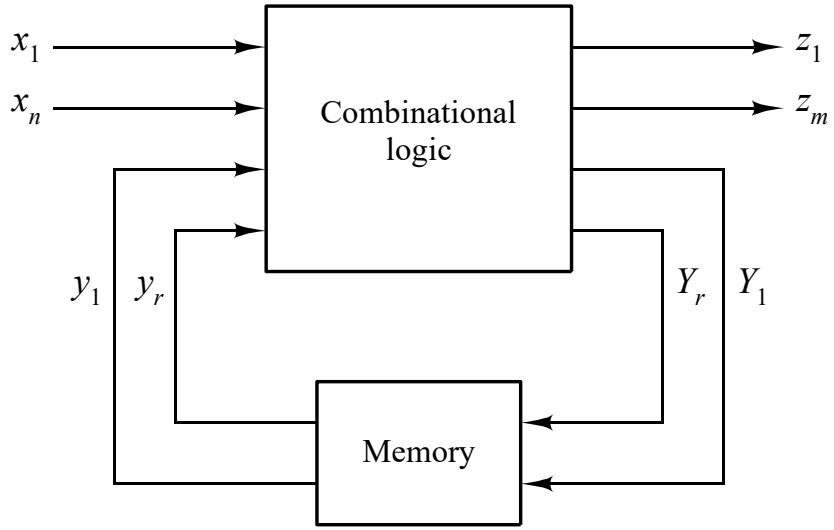
Synchronous Sequential Logics

- Circuit **behaves** at **discrete instance of time**
  - Determined by **CLK** signal
- **Storage** works only with the arrival of a clock pulse
- CLK
  - Avoids instability of the output
  - Clock Frequency = 1 / clock cycle time
    - Cycles per second, Hz



Asynchronous Sequential Logics

- Circuit **behaves** based on input changes
- Circuit **behaves** at any instant of time
  - Output may not be stable
- Not common



Sequential Modules

- Register
- Shift Register
- Counter

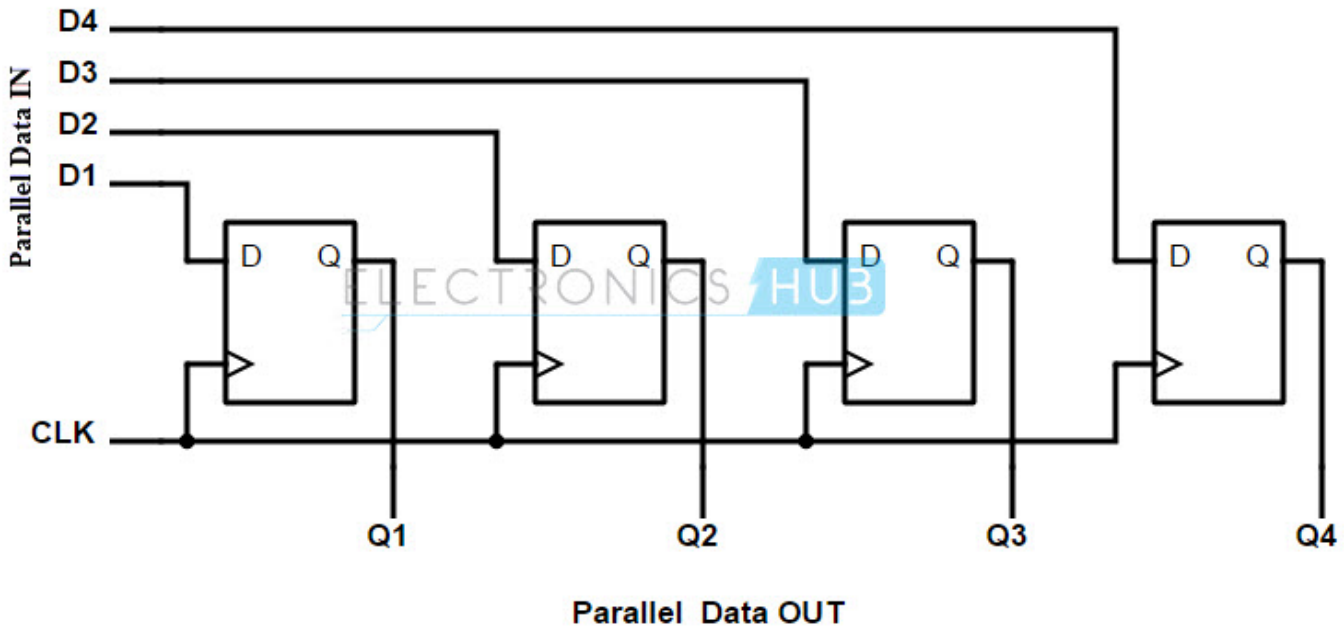
Register

Register

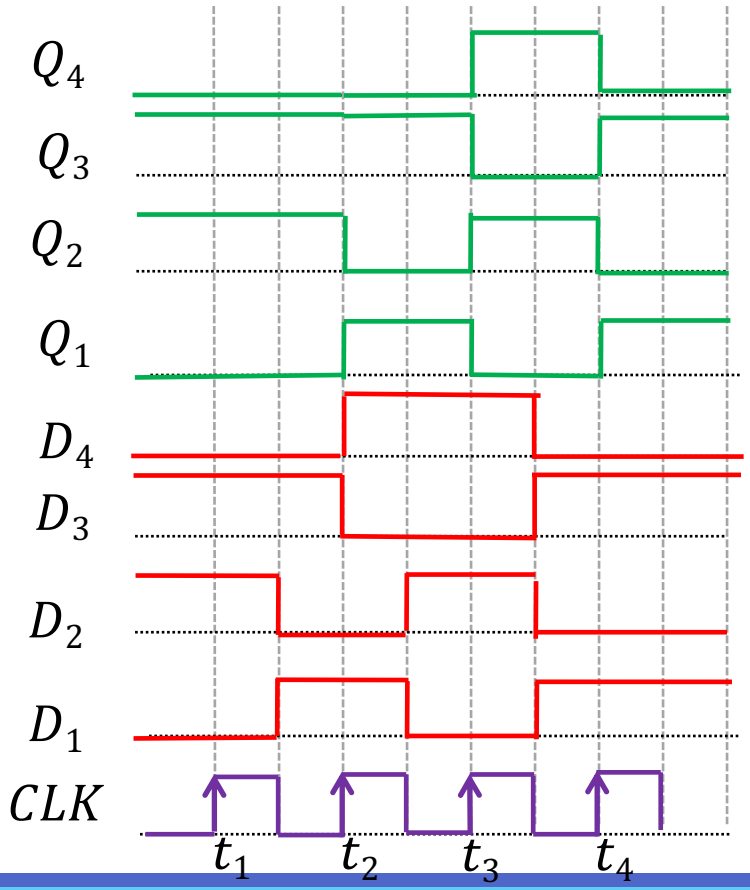
- A collection of binary storage elements
  - Store a vector of binary values
  - Perform simple data storage
  - Perform data movement

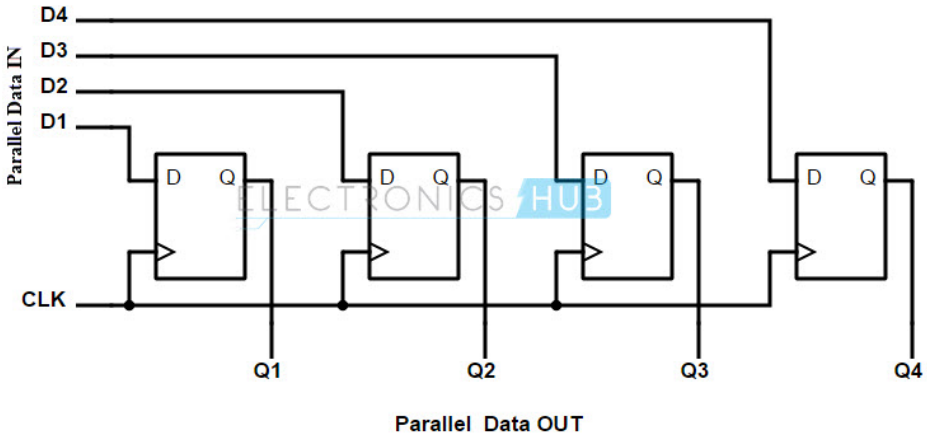
Register: Sample 1

- Synchronous register
- A collection of D FF



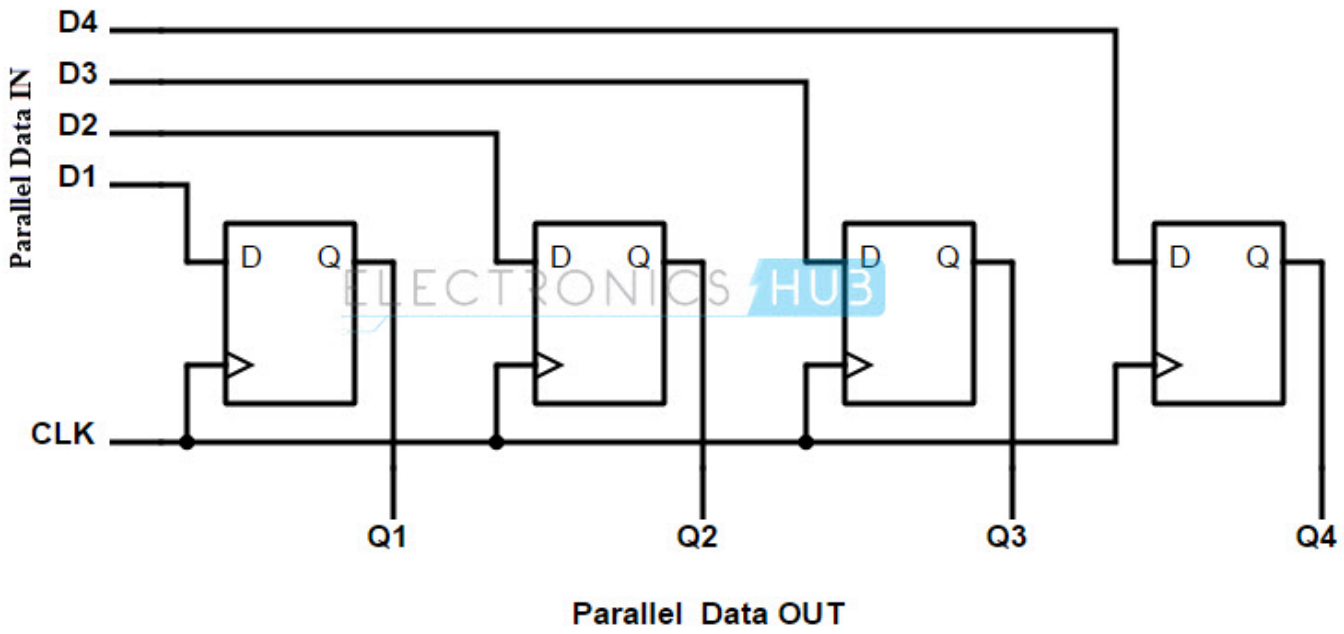
Register: Sample 1 (cont'd)





Register: Sample 1

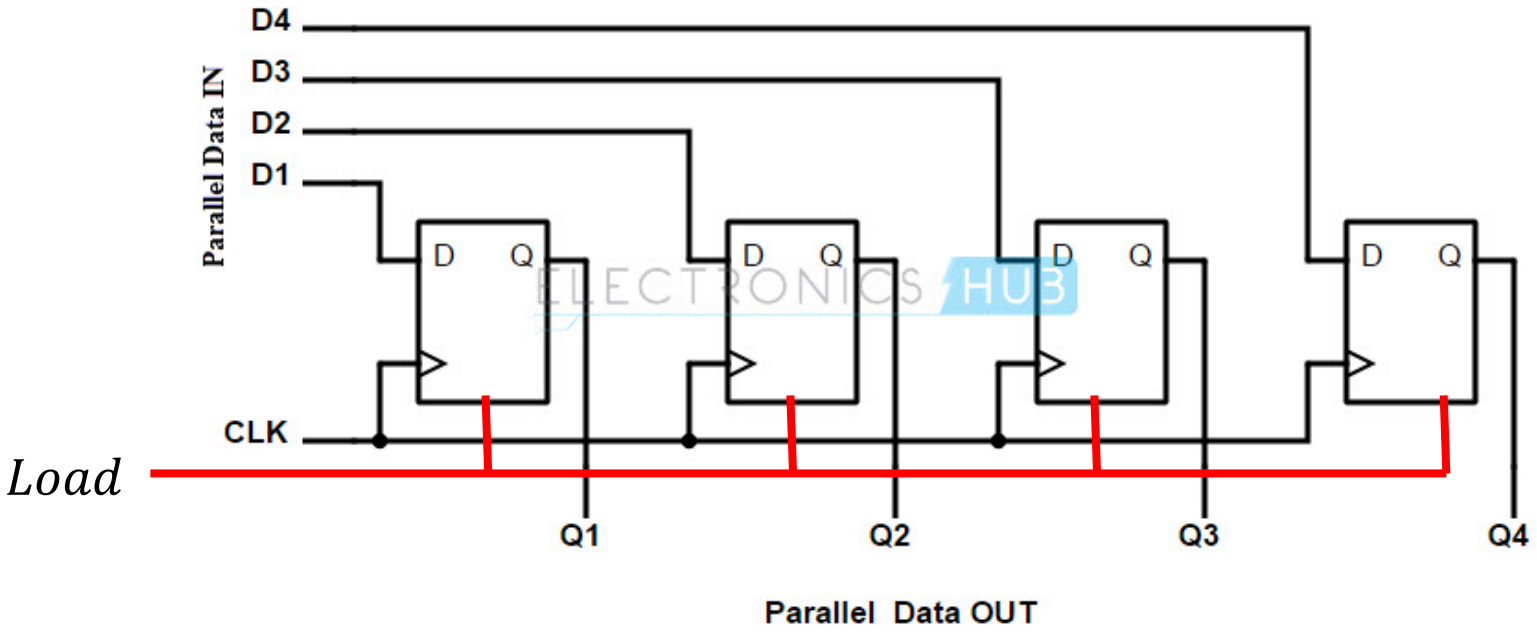
- Register with **parallel load**



Register with Parallel Load

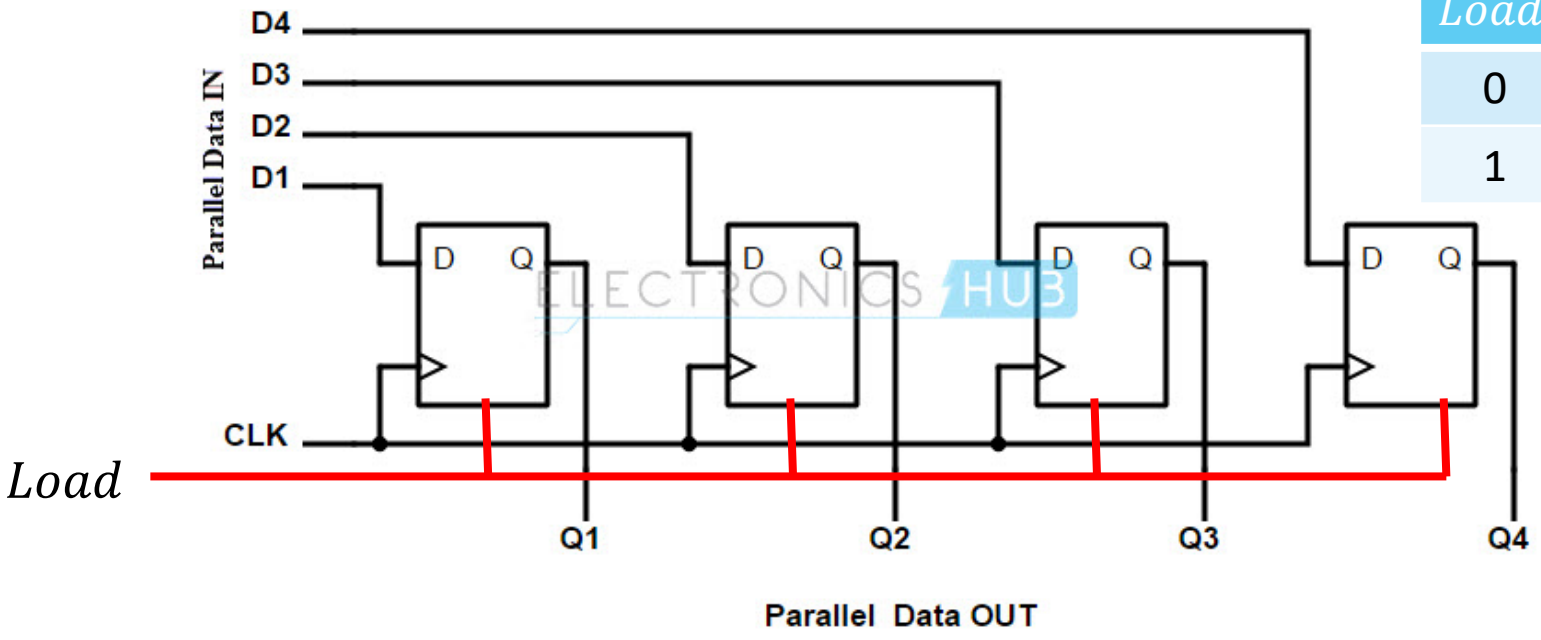
- Load (LD)
  - Control loading the new data to the register

$Load$	$Q(t+1)$
0	$Q(t)$
1	D



Register with Parallel Load (cont'd)

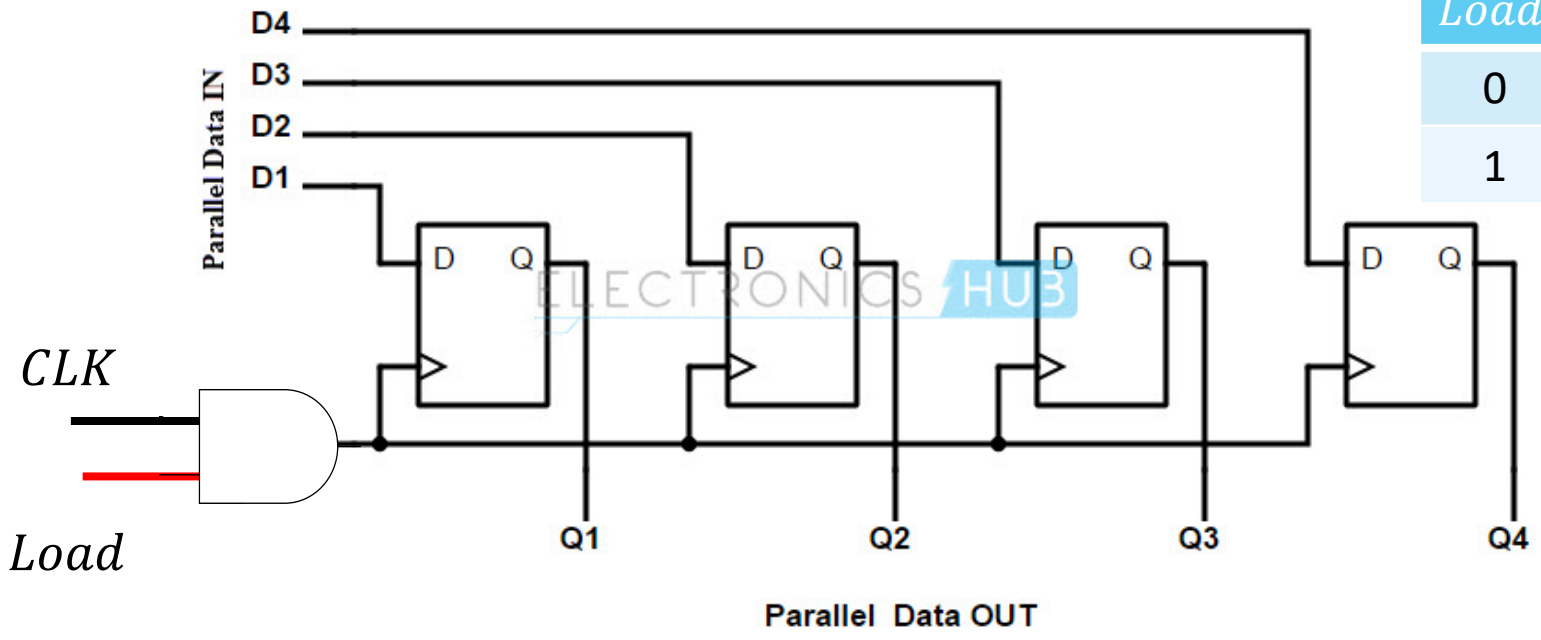
- How to insert a load signal to the register with parallel load?



$Load$	$Q(t+1)$
0	$Q(t)$
1	D

Register with Parallel Load (cont'd)

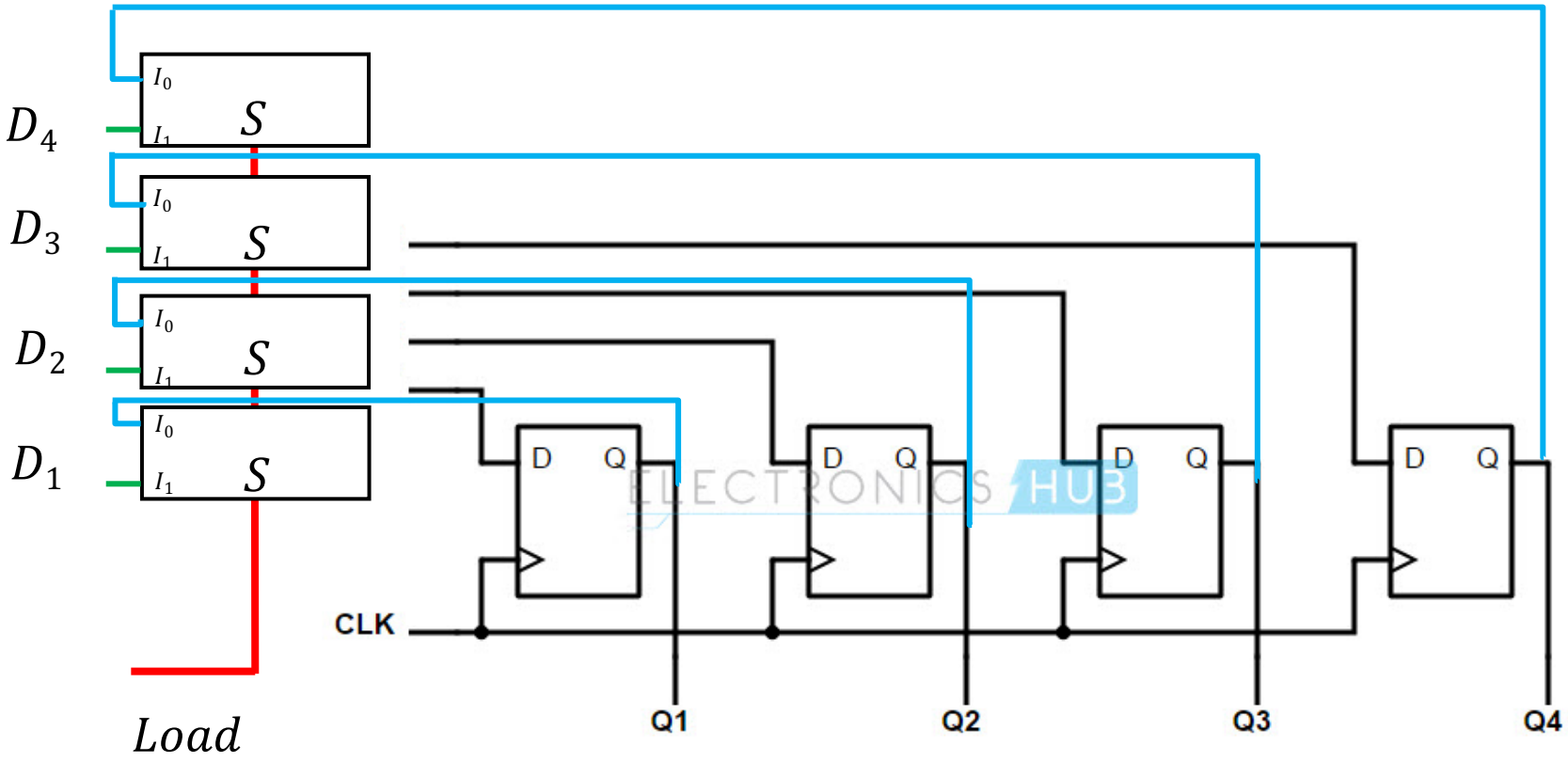
- And Load and CLK signal together
- Delays the clock



$Load$	$Q(t+1)$
0	$Q(t)$
1	D

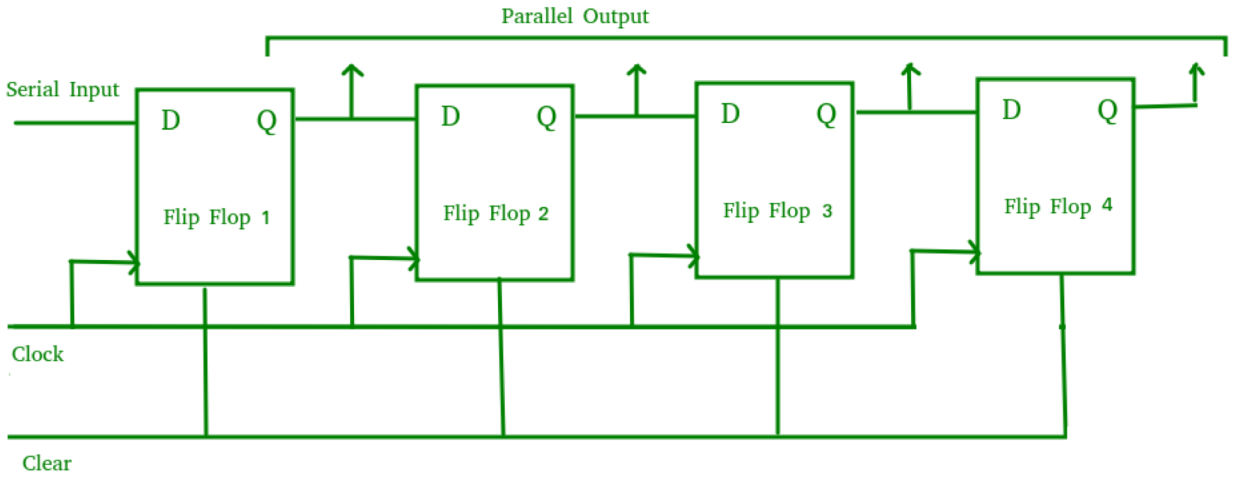
Register with Parallel Load (cont'd)

- Use MUX to select the new data by load signal

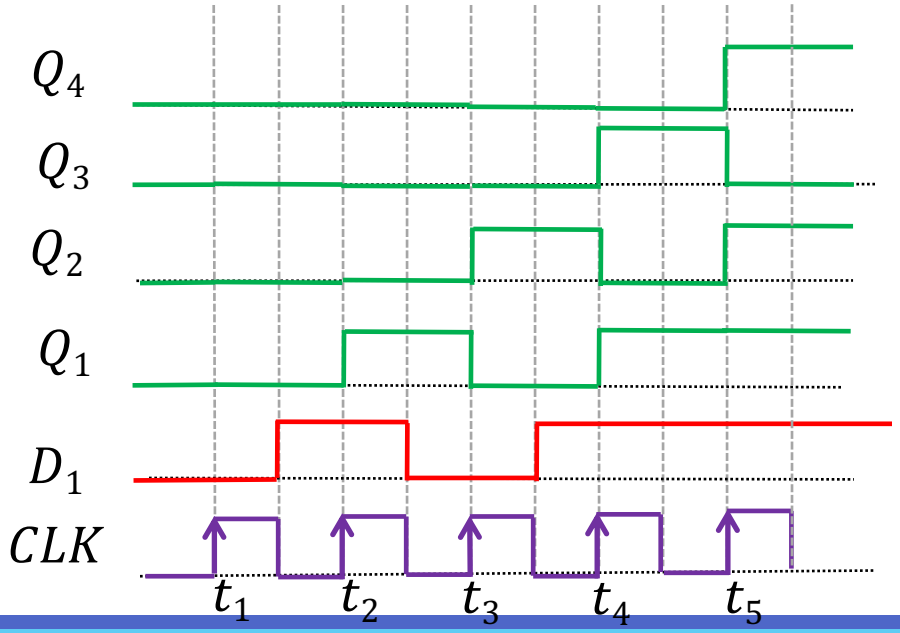


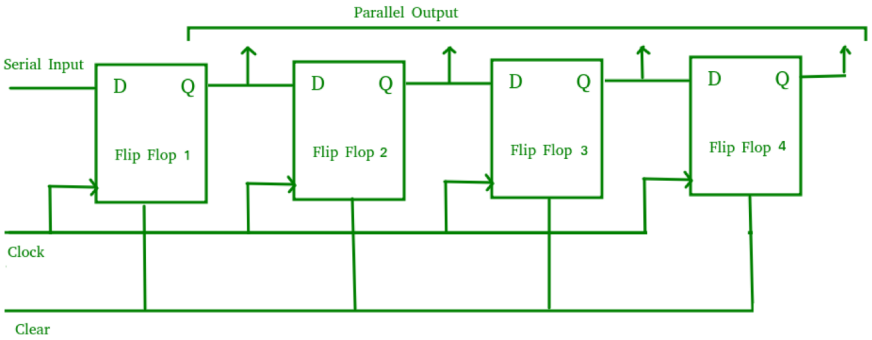
Register: Sample 2

- Synchronous register
- A collection of D FF



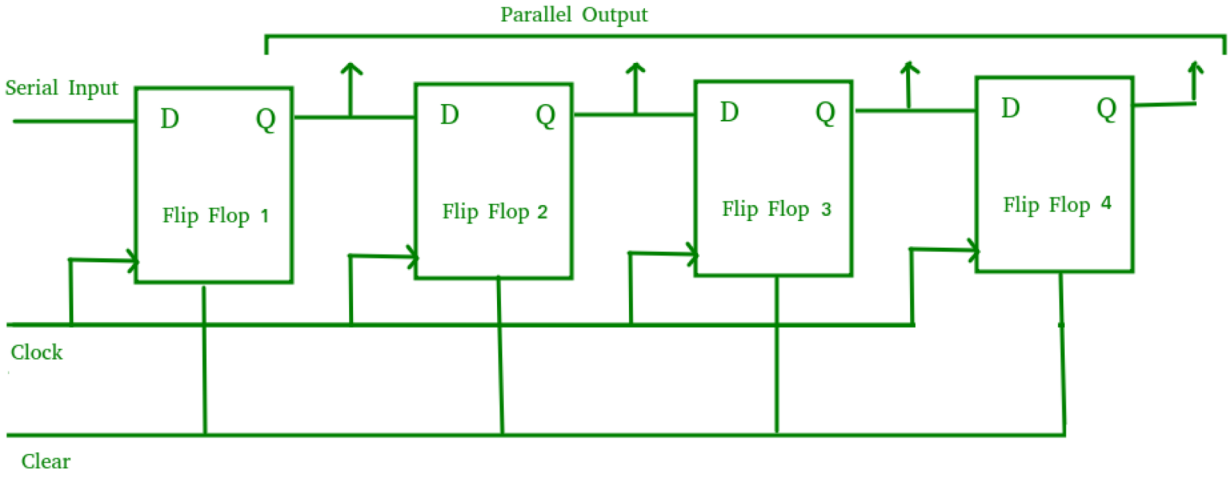
Register: Sample 1 (cont'd)





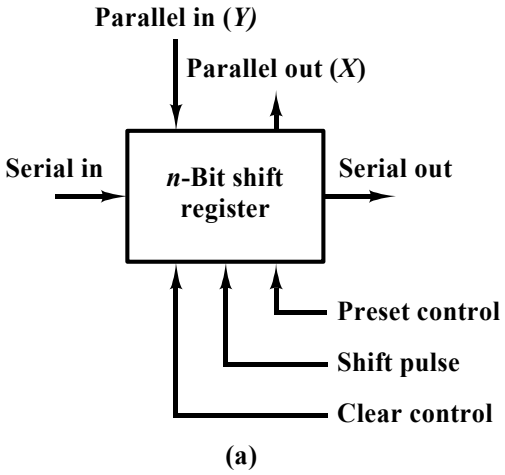
Shift Register

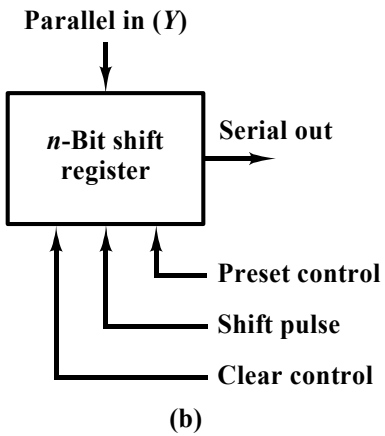
- 4-bit shift register

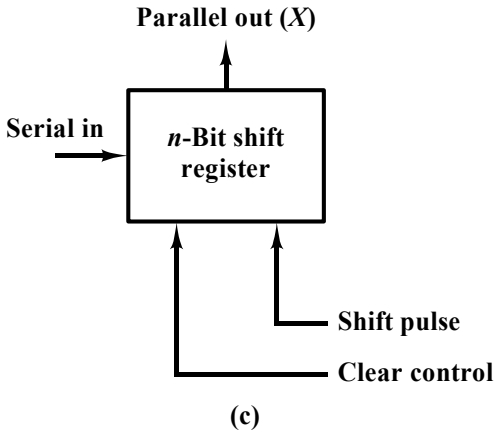


Shift Register: Types

- Parallel-in Parallel-out
- Serial-in Serial-out
- Serial-in Parallel-out
- Parallel-in Serial-out

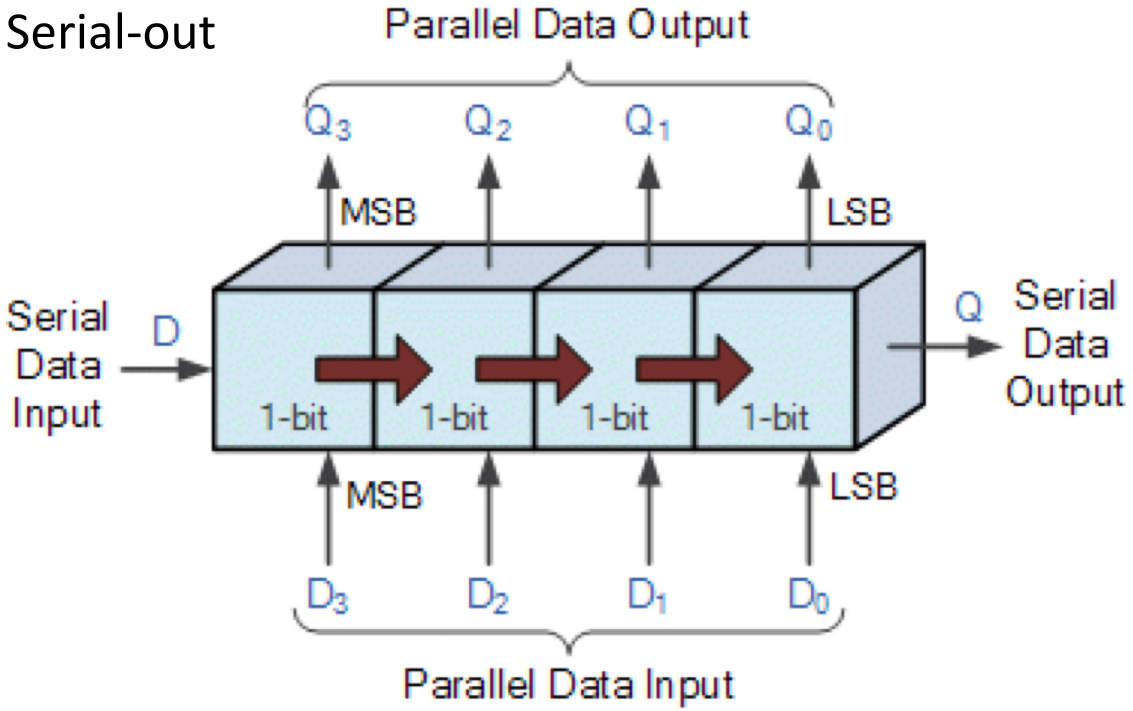






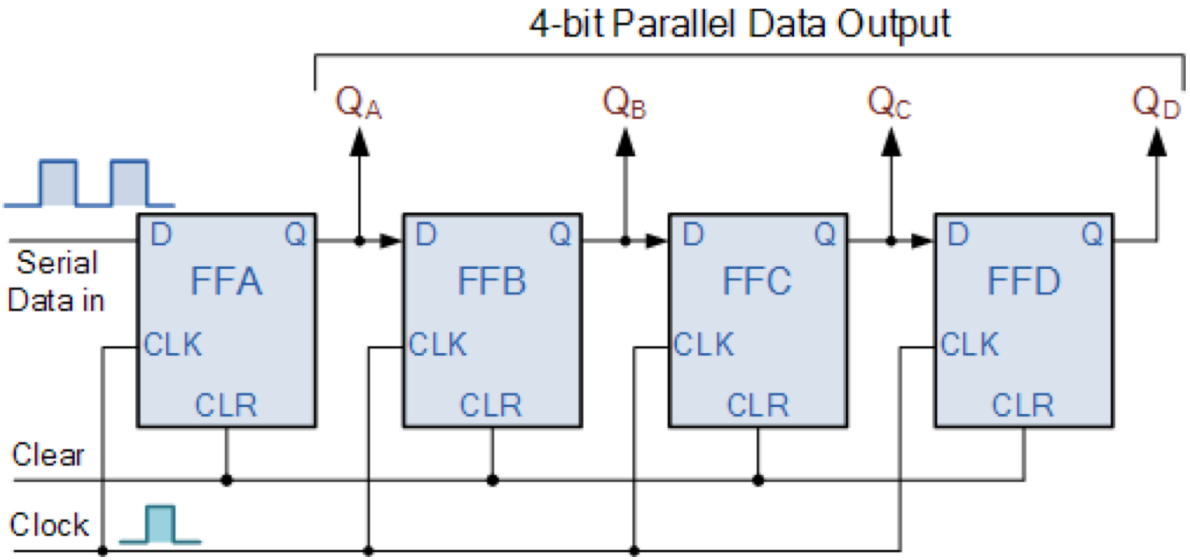
Shift Register: Types (cont'd)

- Parallel-in Parallel-out
- Serial-in Serial-out
- Serial-in Parallel-out
- Parallel-in Serial-out



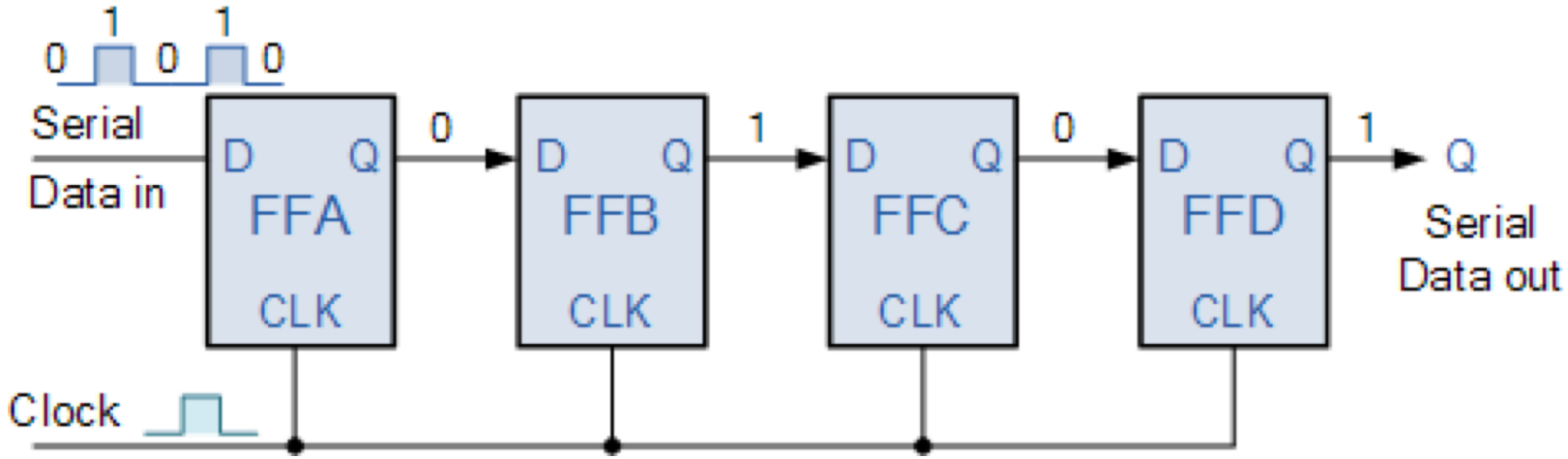
Serial-in Parallel-out Shift Register

- 4-bit shift register
  - Serial input
  - Parallel output



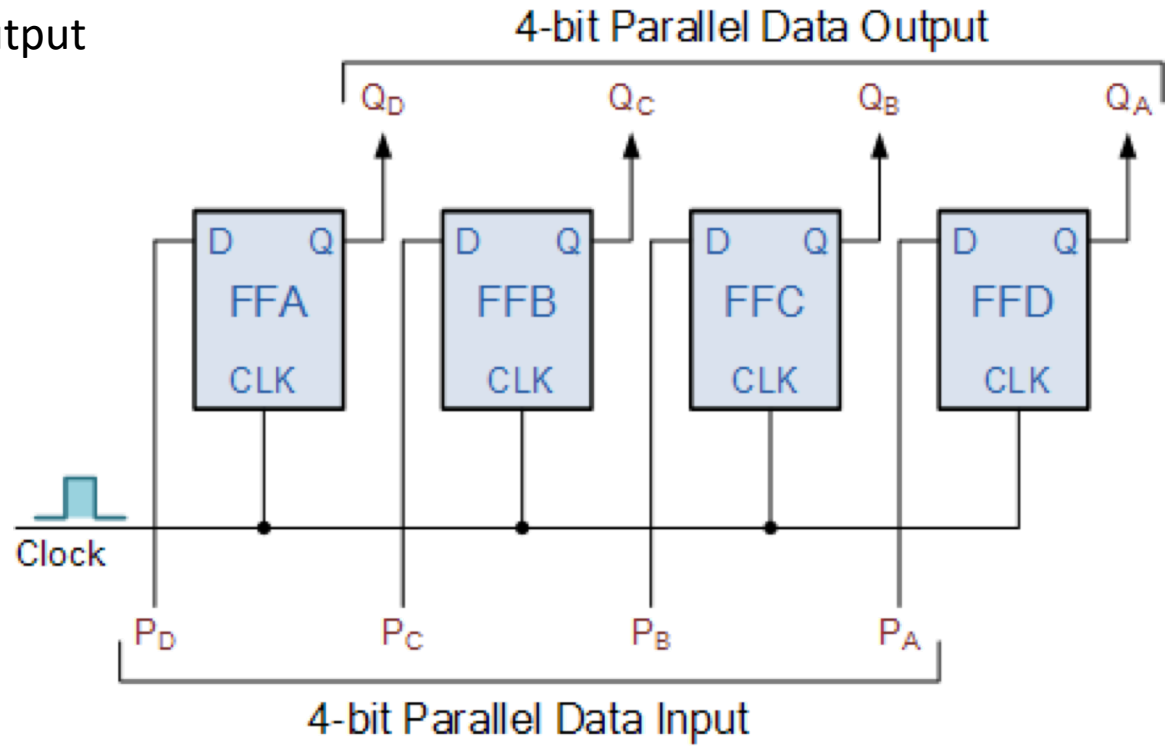
Serial-in Serial-out Shift Register

- 4-bit shift register
  - Parallel input
  - Serial output



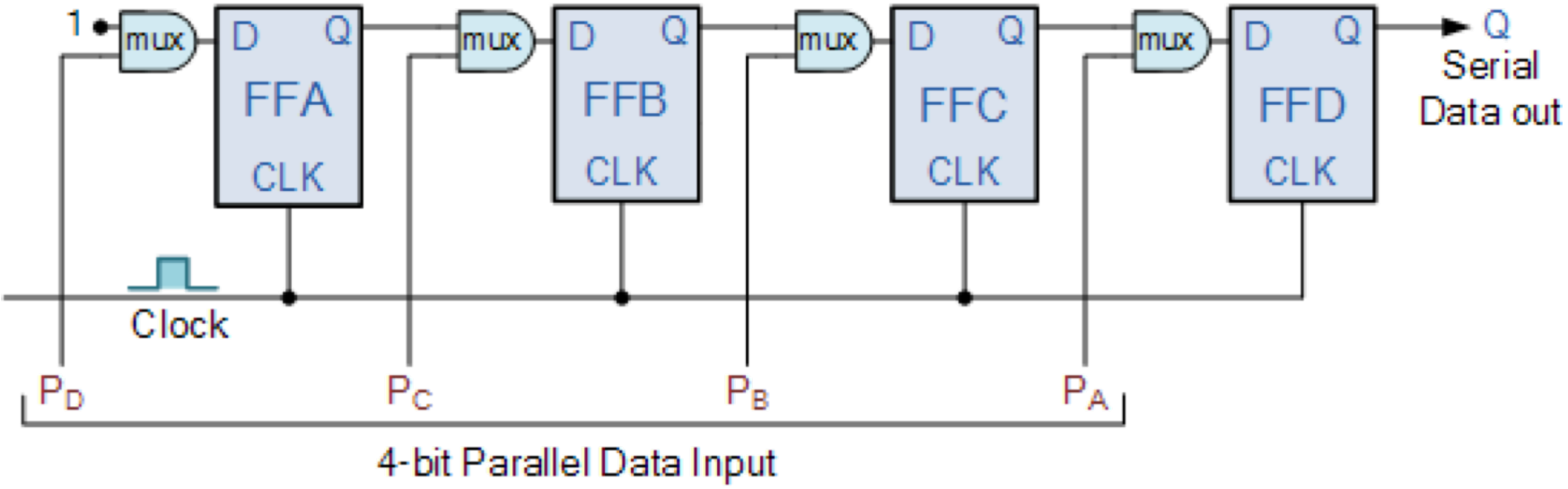
Parallel-in Parallel-out Shift Register

- 4-bit shift register
  - Parallel input
  - Serial output



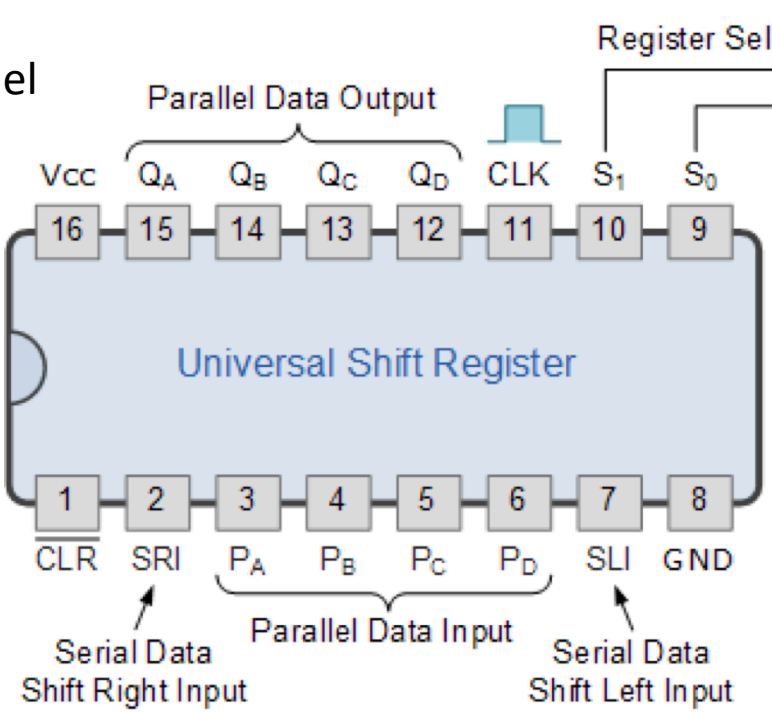
Parallel-in Serial-out Shift Register

- 4-bit shift register
  - Parallel input
  - Serial output



Universal Shift Register (USR)

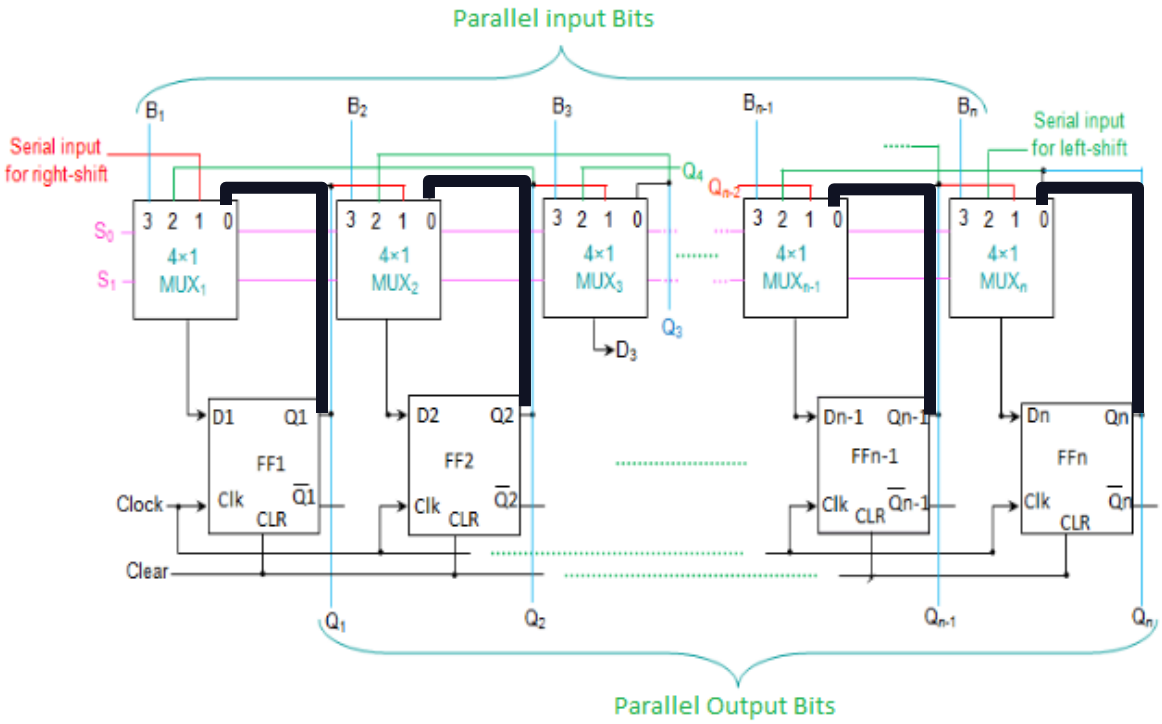
- Functionalities
  - Serial-to-serial
  - Serial-to-parallel
  - Parallel-to-serial
  - Parallel-to-parallel
  - Left shifting
  - Right shifting



Mode Control		Register Type
S_1	S_0	
0	0	Hold
0	1	Shift to right
1	0	Shift to left
1	1	Parallel Mode

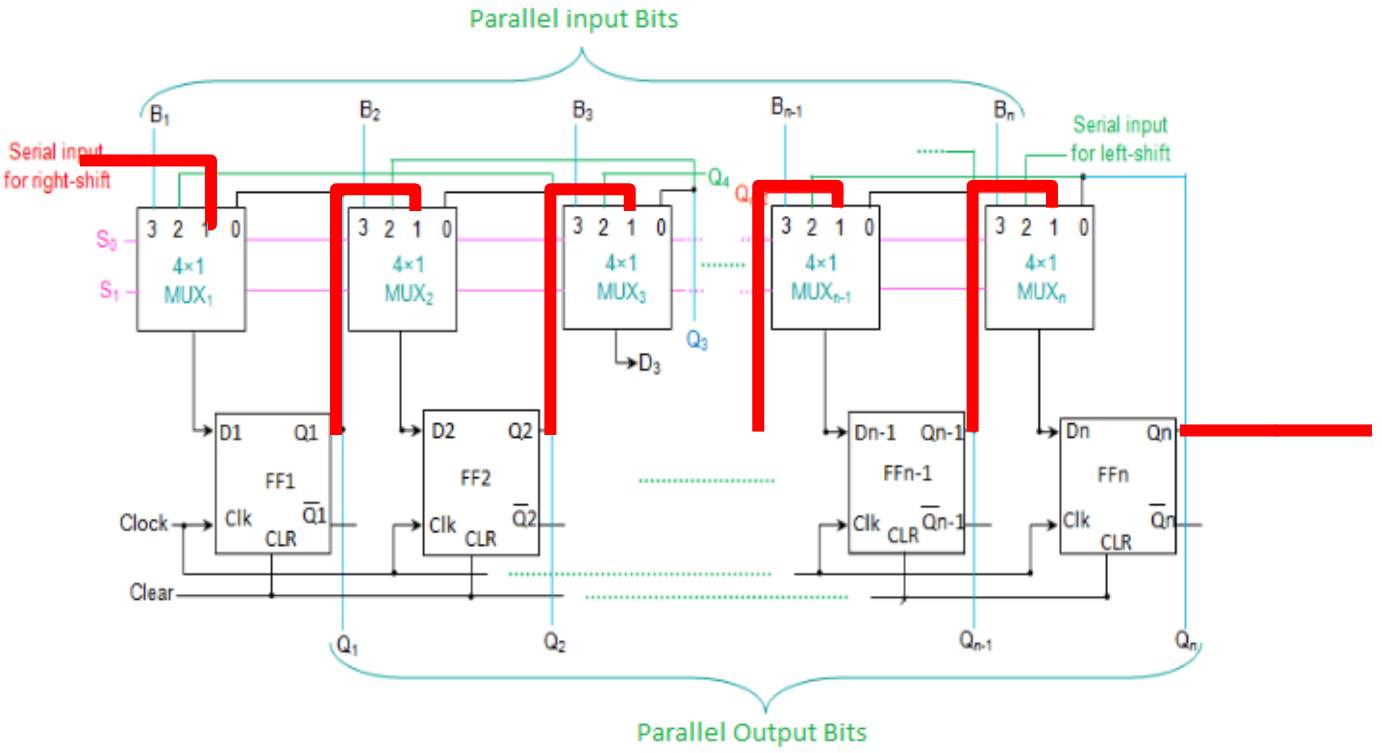
USR: Hold

- Hold



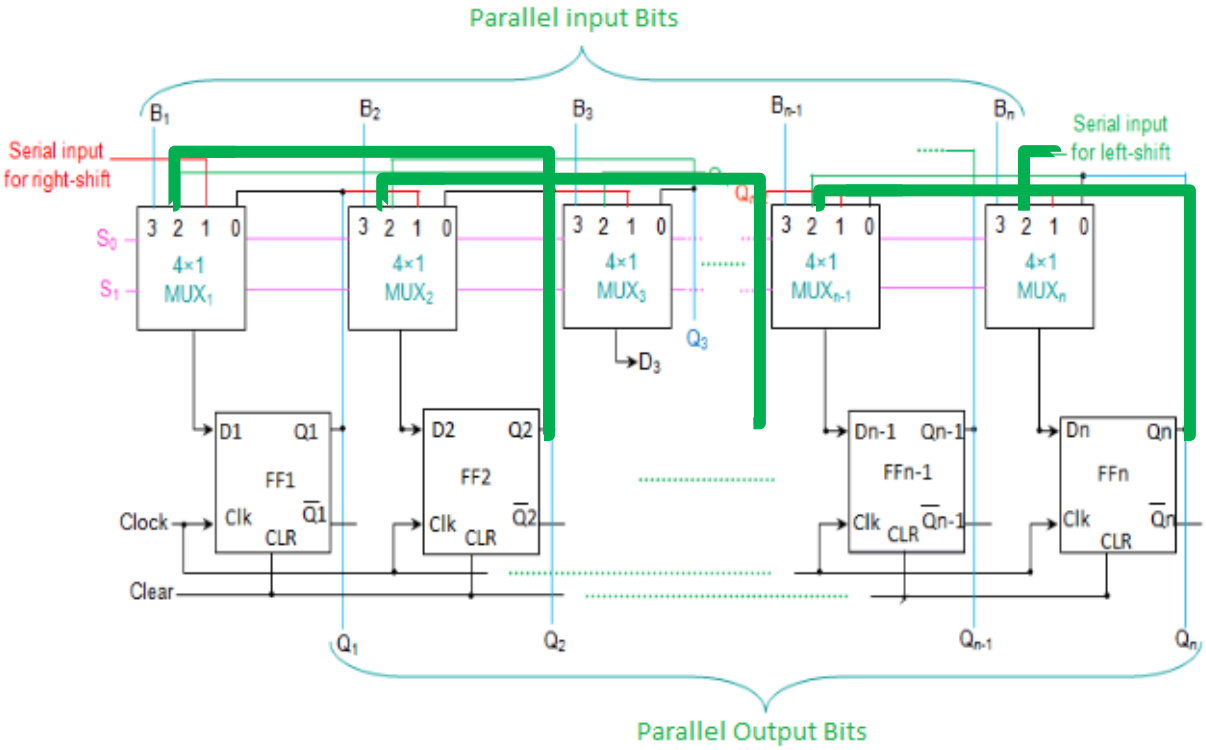
USR: Right Shift

- Right shift



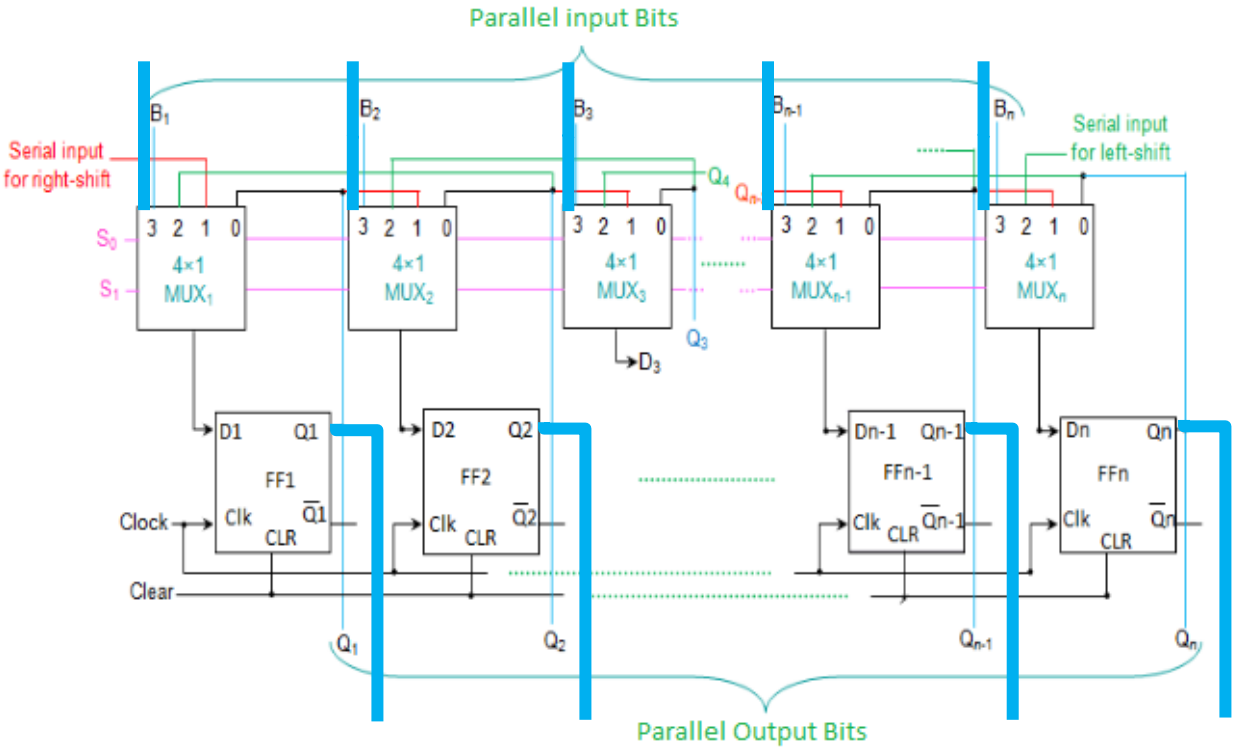
USR: Left Shift

- Left shift



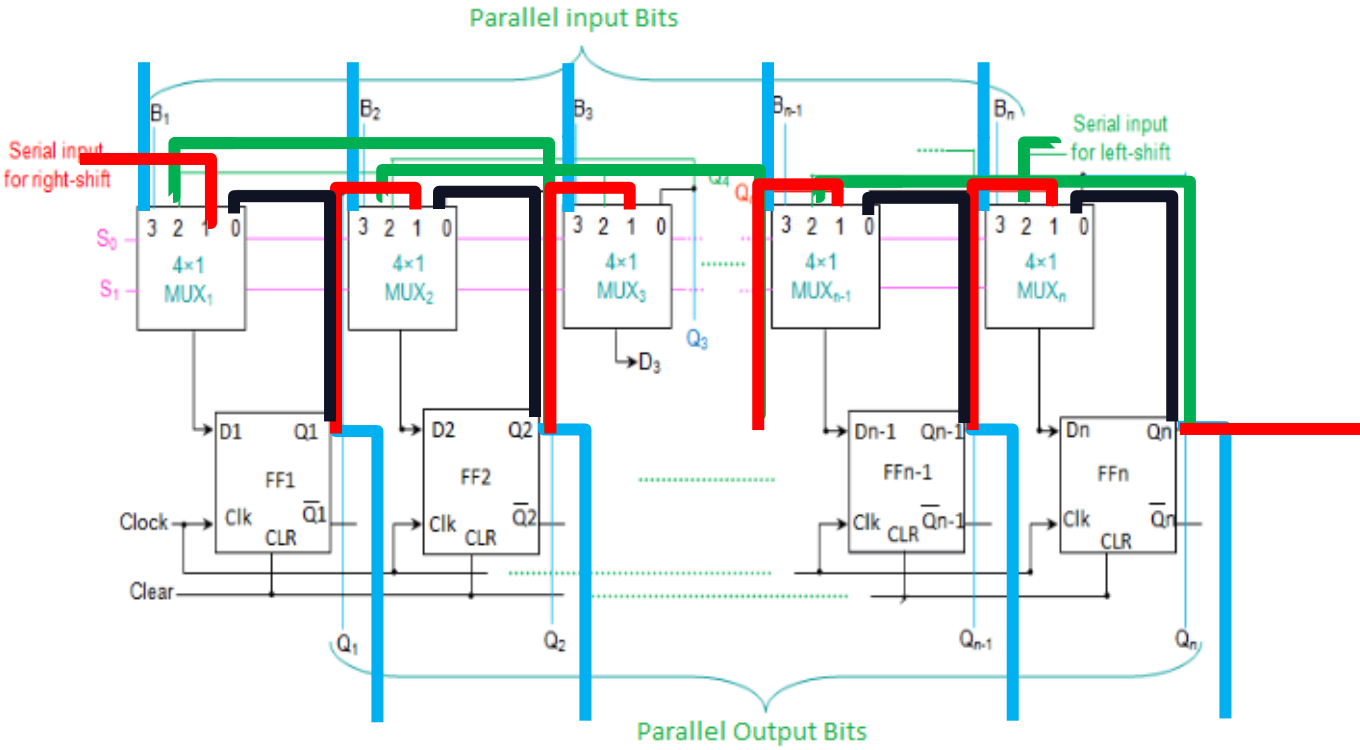
USR: Parallel Load

- Parallel Load



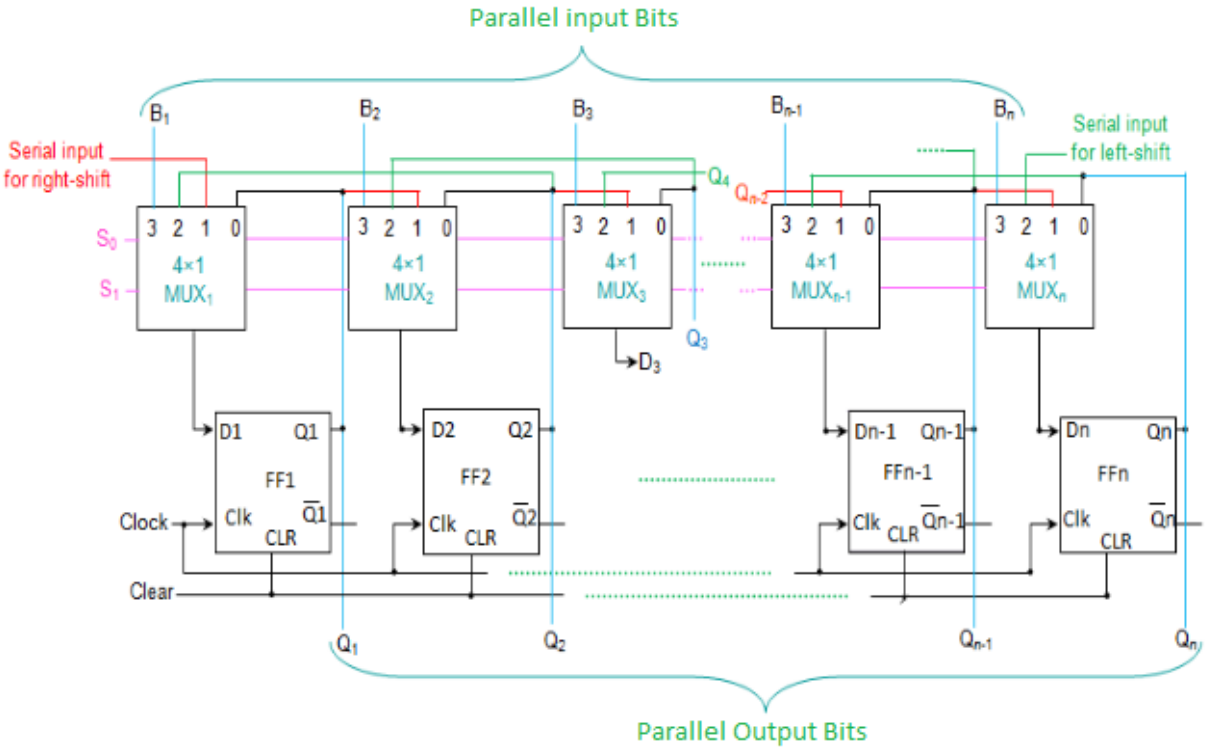
USR: All In One!

- Hold
- Right shift
- Left shift
- Parallel Load



USR: All In One!!

- Hold
- Right shift
- Left shift
- Parallel Load



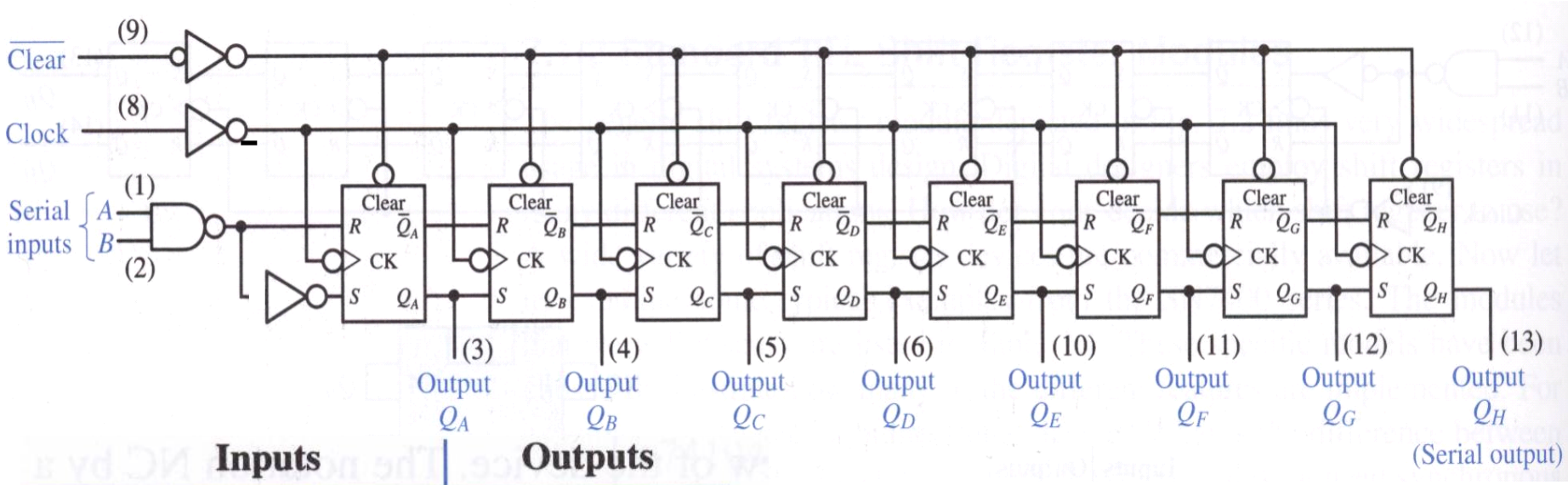
Shift Register Application

- **Send/Receive** data from computer/input device to output device /computer
- **Converting** between **serial** data and **parallel** data
  - Parallel data
    - Computers typically work with multiple-bit quantities
  - Serial data
    - Keyboards, mice, printers.
    - Serial port, USB, Firewire

Standard TTL Shift Registers

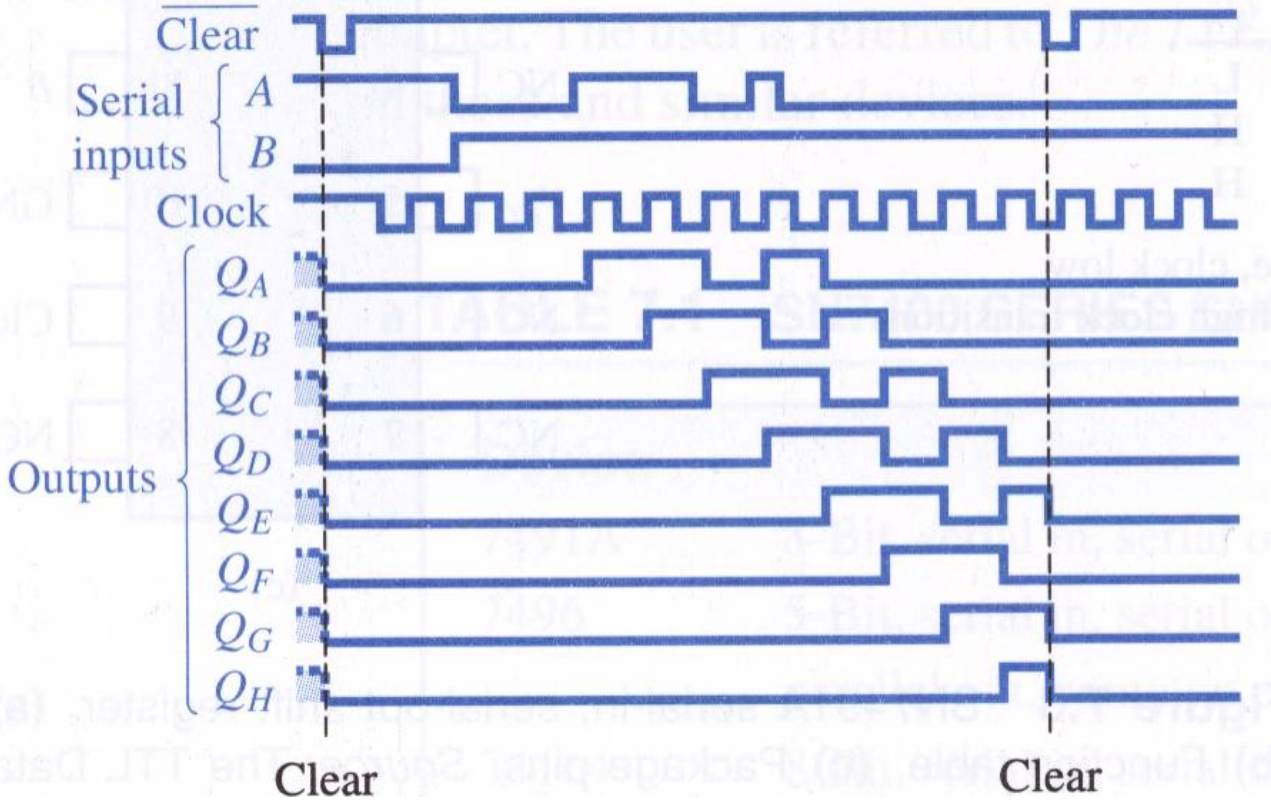
Device	Features
7491A	8-Bit, serial in, serial out
7496	5-Bit, serial in, serial out, asynchronous preset, parallel out, common clear
74164	8-Bit, serial in, serial out or parallel out, common clear
74165	8-Bit, serial in, serial out, asynchronous load, clock inhibit
74179	4-Bit, serial in, serial out, common clear, synchronous load, parallel out, synchronous data hold
74194	4-Bit, bidirectional, serial in, serial out, synchronous load, parallel out, clock inhibit, common clear

74164: 8-bit, Serial In, Serial/Parallel Out



Inputs				Outputs			
$\overline{\text{Clear}}$	Clock	A	B	Q_A	Q_B	…	Q_H
L	×	×	×	L	L		L
H	L	×	×	Q_{A0}	Q_{B0}		Q_{H0}
H	↑	H	H	H	Q_{An}		Q_{Gn}
H	↑	L	×	L	Q_{An}		Q_{Gn}
H	↑	×	L	L	Q_{An}		Q_{Gn}

74164 operation



74194: 4-bit, Serial In, Serial/Parallel out

- 74194: 4-bit, bidirectional, serial-in, serial/parallel out shift register
  - Asynchronous common clear
  - Synchronous load

Inputs				Internal signals ($i = A, B, C, D$)	
$\overline{\text{Clear}}$	$S0$	$S1$	Clock	S_i	Mode
L	×	×	×	×	Asynchronous clear
H	L	L	×	×	Clock inhibit (data hold)
H	L	H	↑	Q_{i+1}	Shift left
H	H	L	↑	Q_{i-1}	Shift right
H	H	H	↑	i	Parallel load

74194 operation

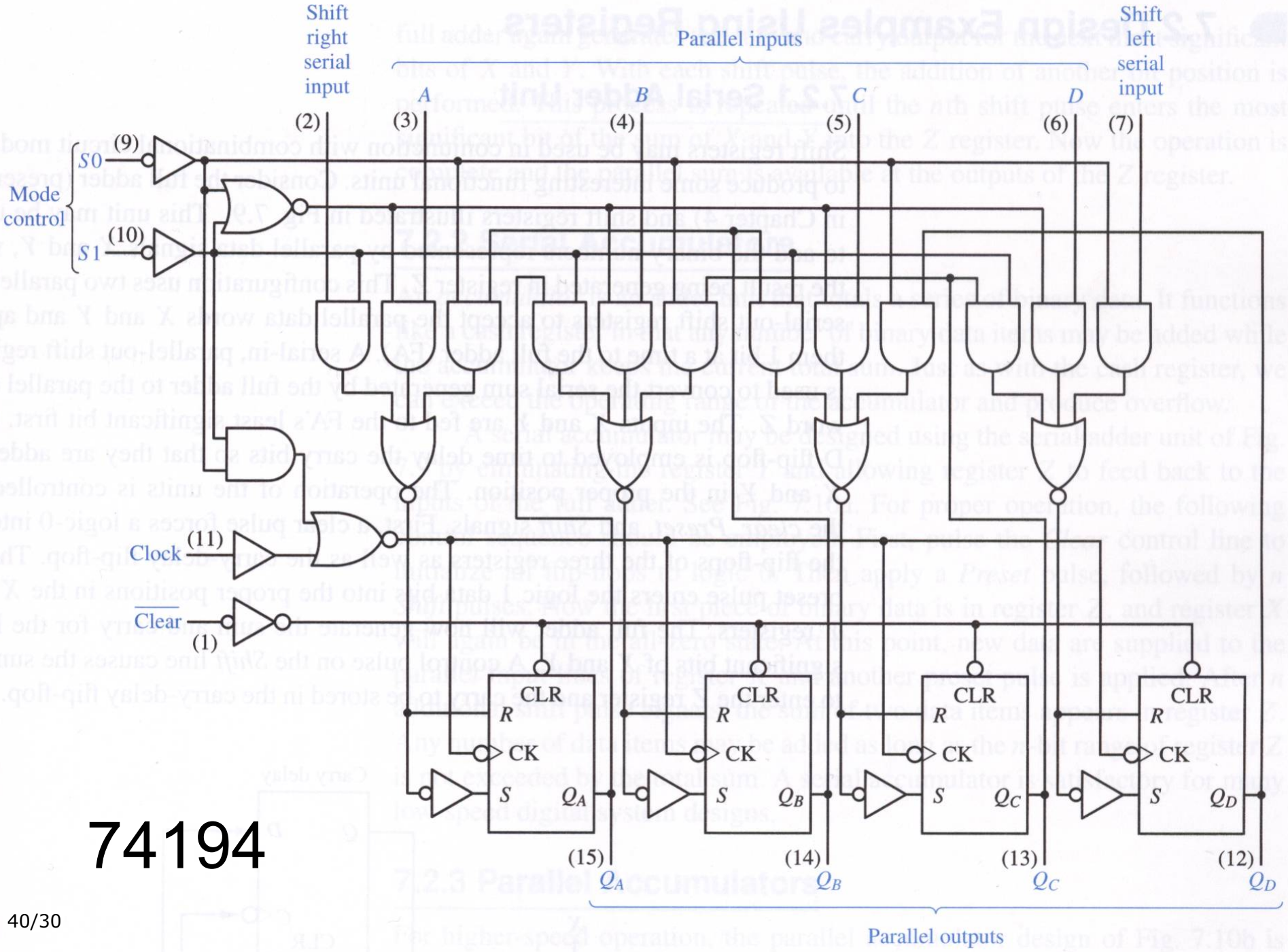
Internal clock CK = CLOCK + S0'•S1'

- When S0 and S1 are both 0, CK = 1; hold

Set input of internal FF (e.g. B) is

$$S_B = Q_C \bullet S0' + Q_A \bullet S1' + B \bullet S0 \bullet S1$$

- S1=1 & S0=0 $\Rightarrow$ $S_B = Q_C$; shift left
- S1=0 & S0=1 $\Rightarrow$ $S_B = Q_A$; shift right
- S1=1 & S0=1 $\Rightarrow$ $S_B = B$; external load
- Synchronous parallel load


Shift right serial input
Parallel inputs
Shift left serial input
A
B
C
D
(2)
(3)
(4)
(5)
(6)
(7)
(9)
S0
Mode control
(10)
S1
(11)
Clock
Clear
(1)
CLR
R
CK
S
Q_A
Q_B
Q_C
Q_D
(15)
(14)
(13)
(12)
Parallel outputs

74194

Thank You