REGISTERS -
Introduction to Registers
Shift Registers
Lecture 1

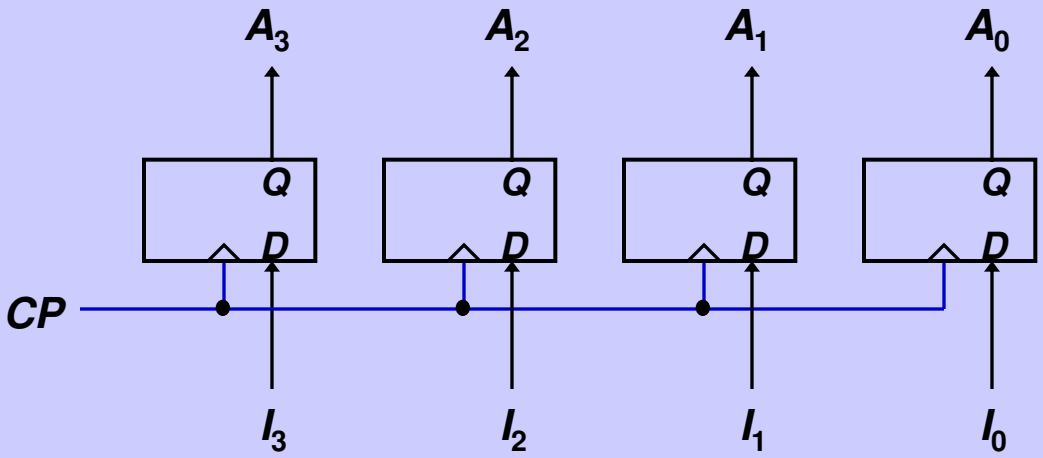
Dronacharya Group of Institutions

Introduction: Registers

- An *n*-bit register has a group of *n* flip-flops and some logic gates and is capable of storing *n* bits of information.
- The flip-flops store the information while the gates control when and how new information is transferred into the register.
- Some functions of register:
  - retrieve data from register
  - store/load new data into register (serial or parallel)
  - shift the data within register (left or right)

Simple Registers

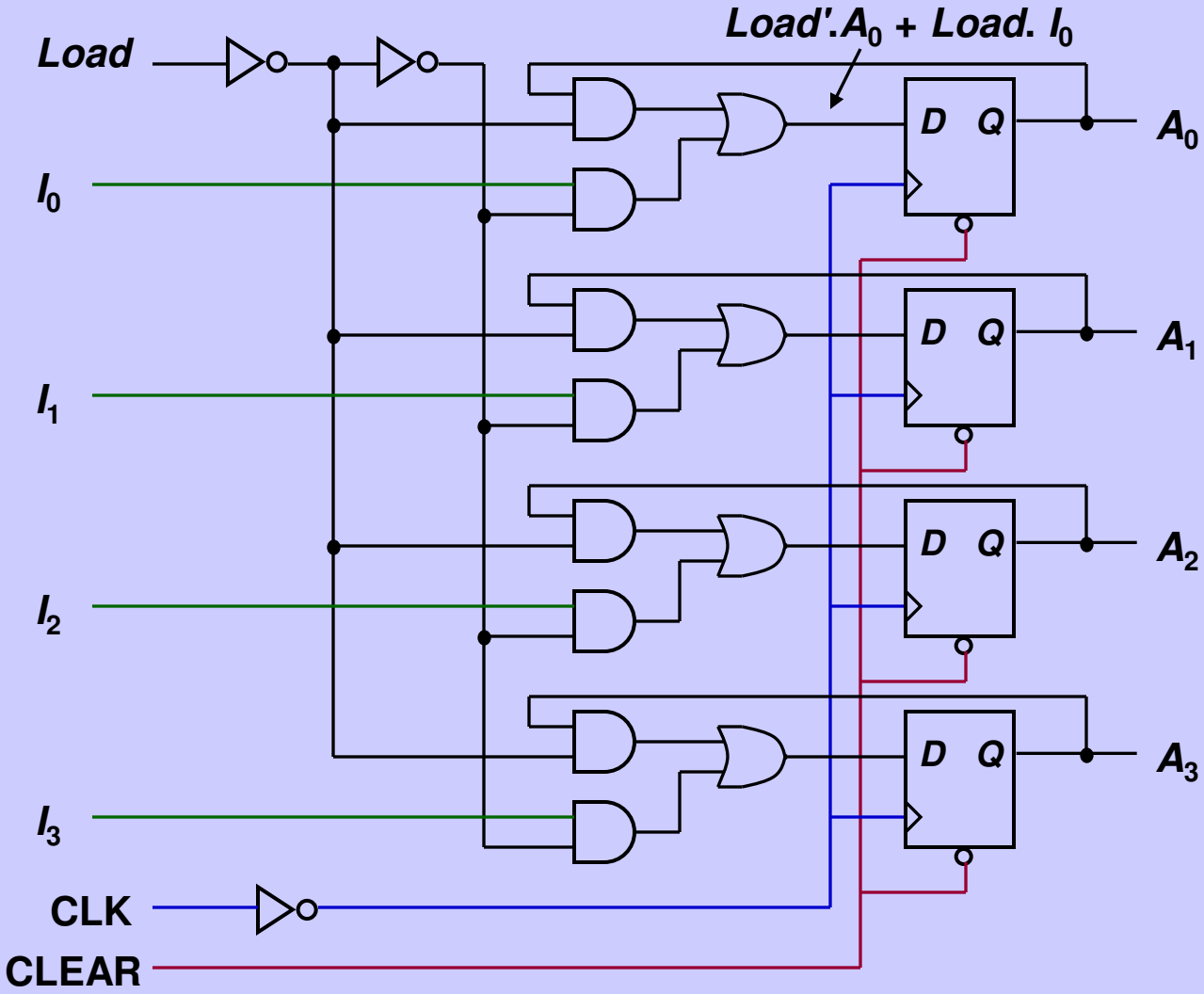
- No external gates.
- Example: A 4-bit register. A new 4-bit data is loaded every clock cycle.



Registers With Parallel Load

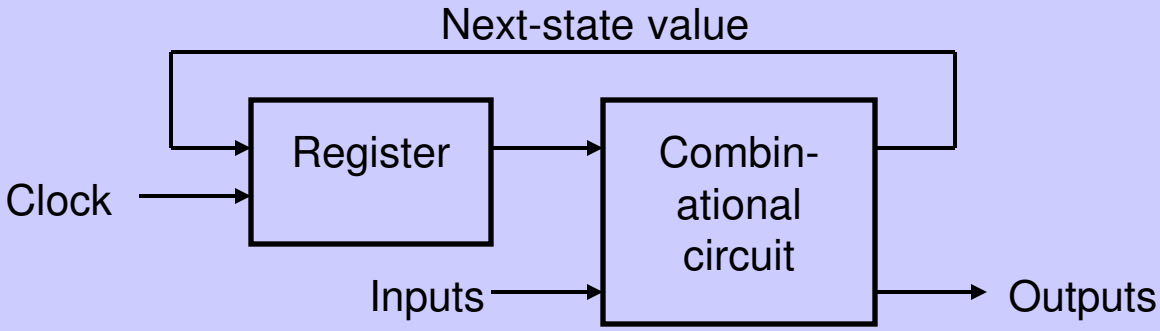
- Instead of loading the register at every clock pulse, we may want to control when to load.
- *Loading* a register: transfer new information into the register. Requires a *load* control input.
- *Parallel loading*: all bits are loaded simultaneously.

Registers With Parallel Load



Using Registers to implement Sequential Circuits

- A sequential circuit may consist of a *register* (memory) and a *combinational circuit*.



- The external inputs and present states of the register determine the next states of the register and the external outputs, through the combinational circuit.
- The combinational circuit may be implemented by any of the methods covered in *MSI components* and *Programmable Logic Devices*.

Using Registers to implement Sequential Circuits

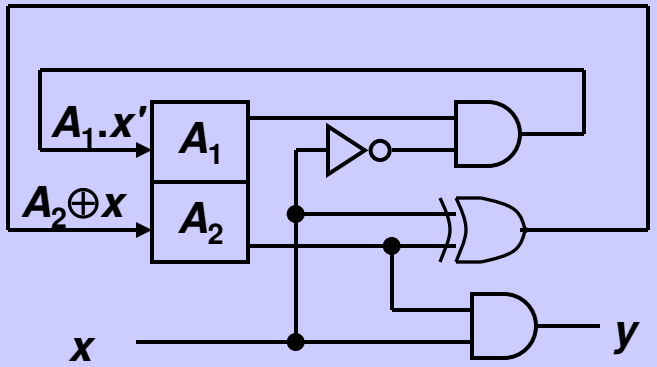
- Example 1:

$A_1^+ = \Sigma\, m(4,6) = A_1.x'$

$A_2^+ = \Sigma\, m(1,2,5,6) = A_2.x' + A_2'.x = A_2 \oplus x$

$y = \Sigma\, m(3,7) = A_2.x$

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	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

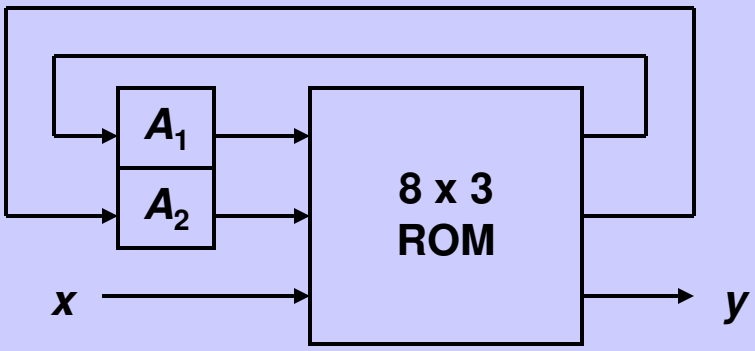


Using Registers to implement Sequential Circuits

- Example 2: Repeat example 1, but use a ROM.

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

ROM truth table

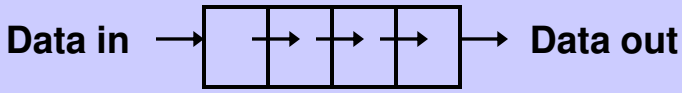


Shift Registers

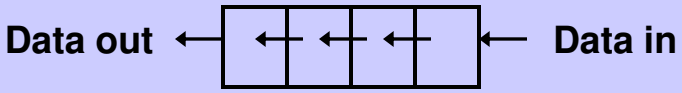
- Another function of a register, besides storage, is to provide for *data movements*.
- Each *stage* (flip-flop) in a shift register represents one bit of storage, and the shifting capability of a register permits the movement of data from stage to stage within the register, or into or out of the register upon application of clock pulses.

Shift Registers

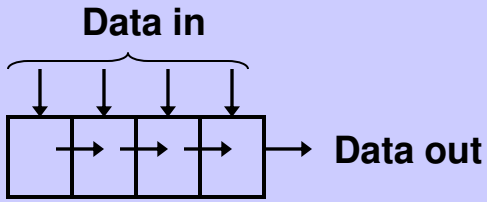
- Basic data movement in shift registers (four bits are used for illustration).



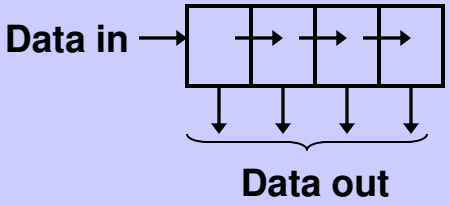
(a) Serial in/shift right/serial out



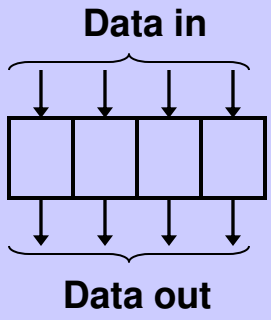
(b) Serial in/shift left/serial out



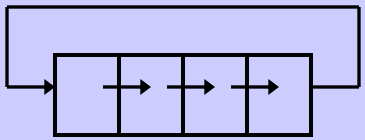
(c) Parallel in/serial out



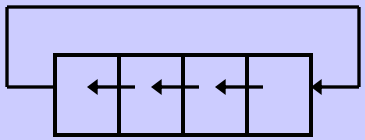
(d) Serial in/parallel out



(e) Parallel in / parallel out



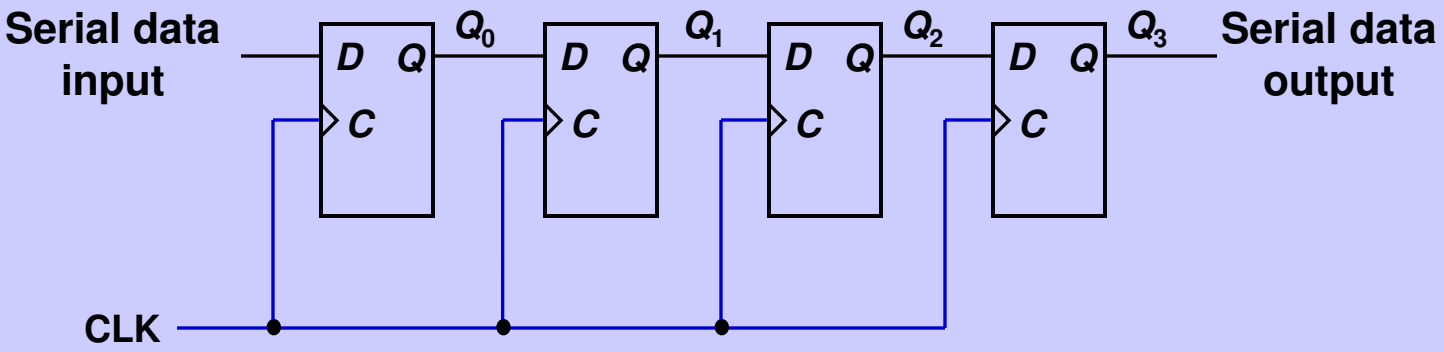
(f) Rotate right



(g) Rotate left

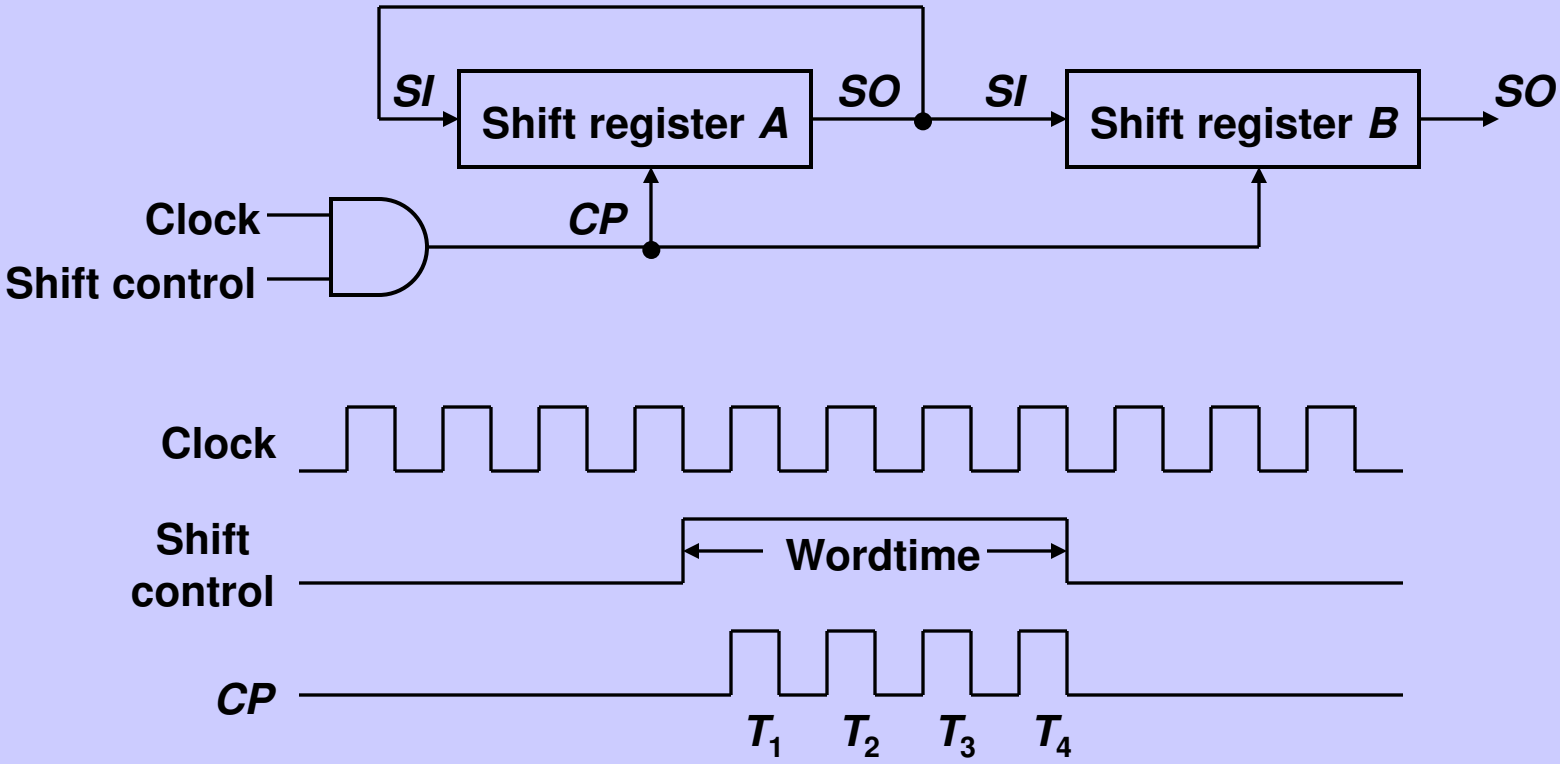
Serial In/Serial Out Shift Registers

- Accepts data serially – one bit at a time – and also produces output serially.



Serial In/Serial Out Shift Registers

- Application: Serial transfer of data from one register to another.



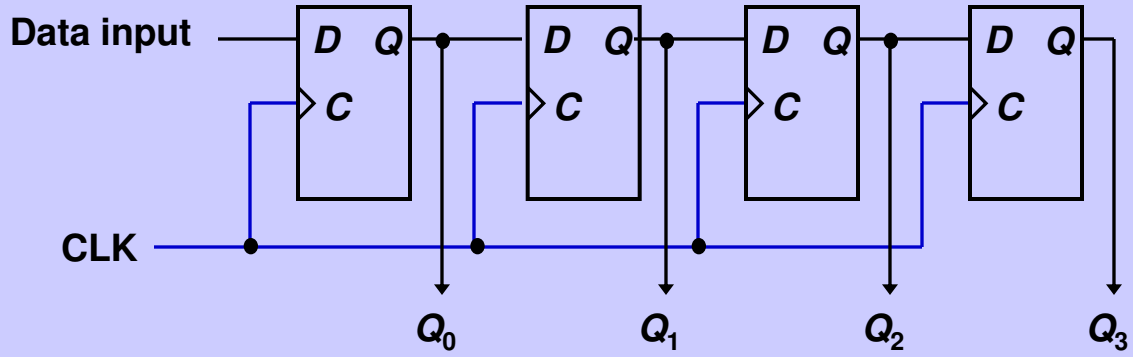
Serial In/Serial Out Shift Registers

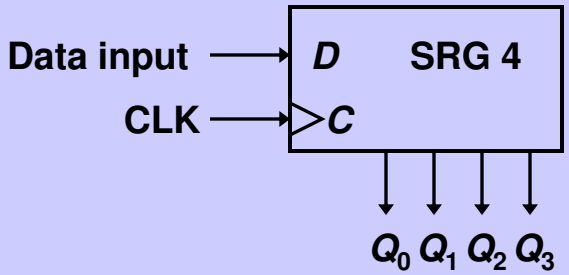
- Serial-transfer example.

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

Serial In/Parallel Out Shift Registers

- Accepts data serially.
- Outputs of all stages are available simultaneously.

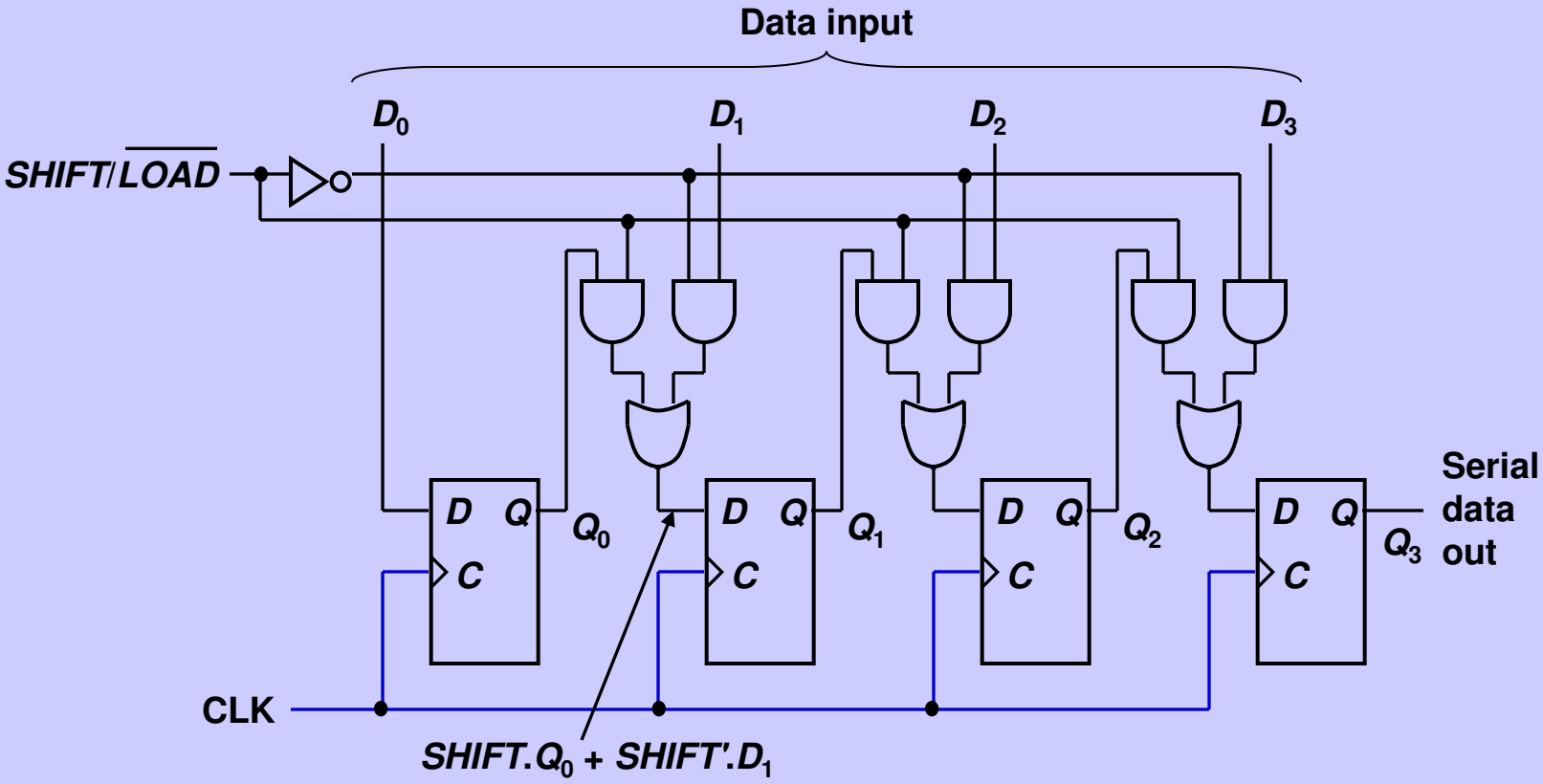




Logic symbol

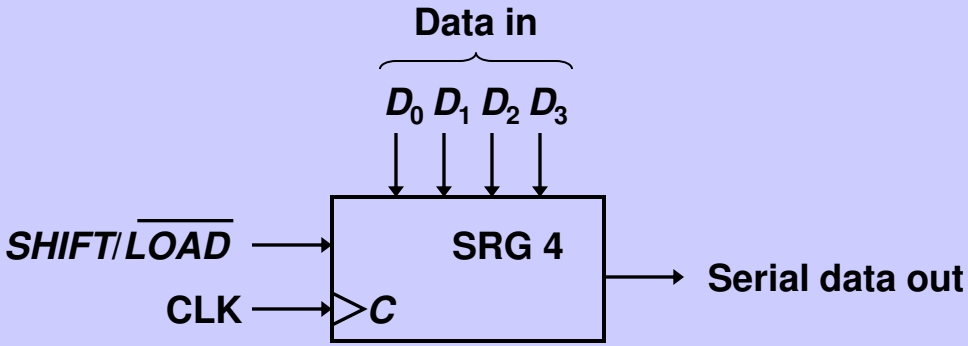
Parallel In/Serial Out Shift Registers

- Bits are entered simultaneously, but output is serial.



Parallel In/Serial Out Shift Registers

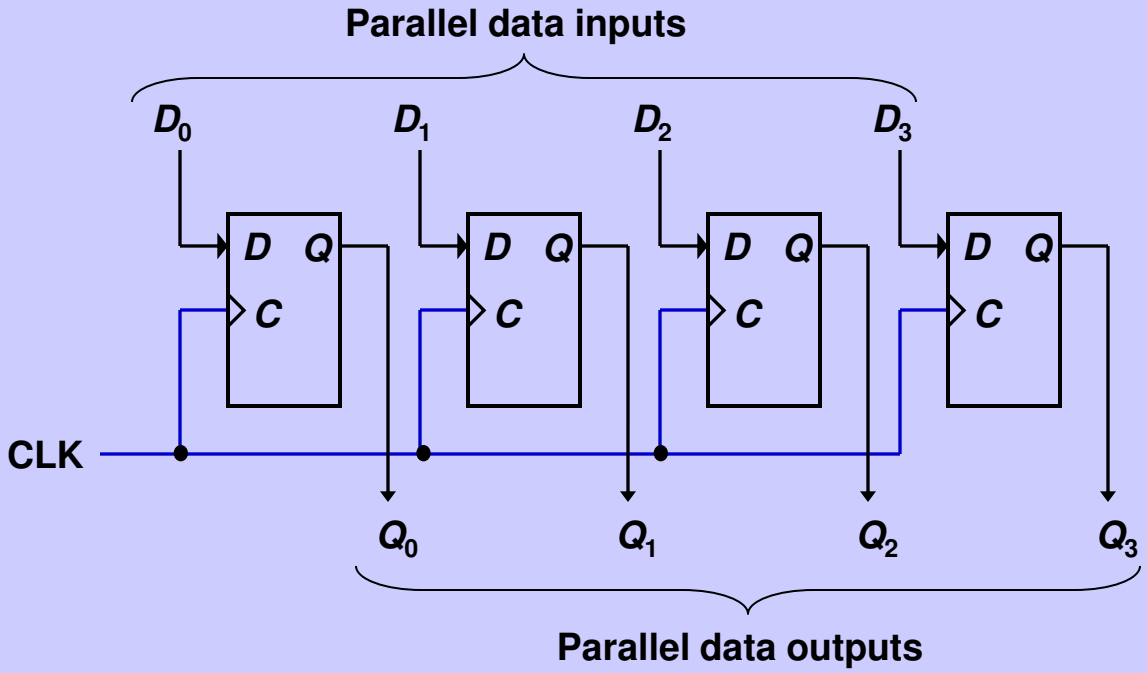
- Bits are entered simultaneously, but output is serial.



Logic symbol

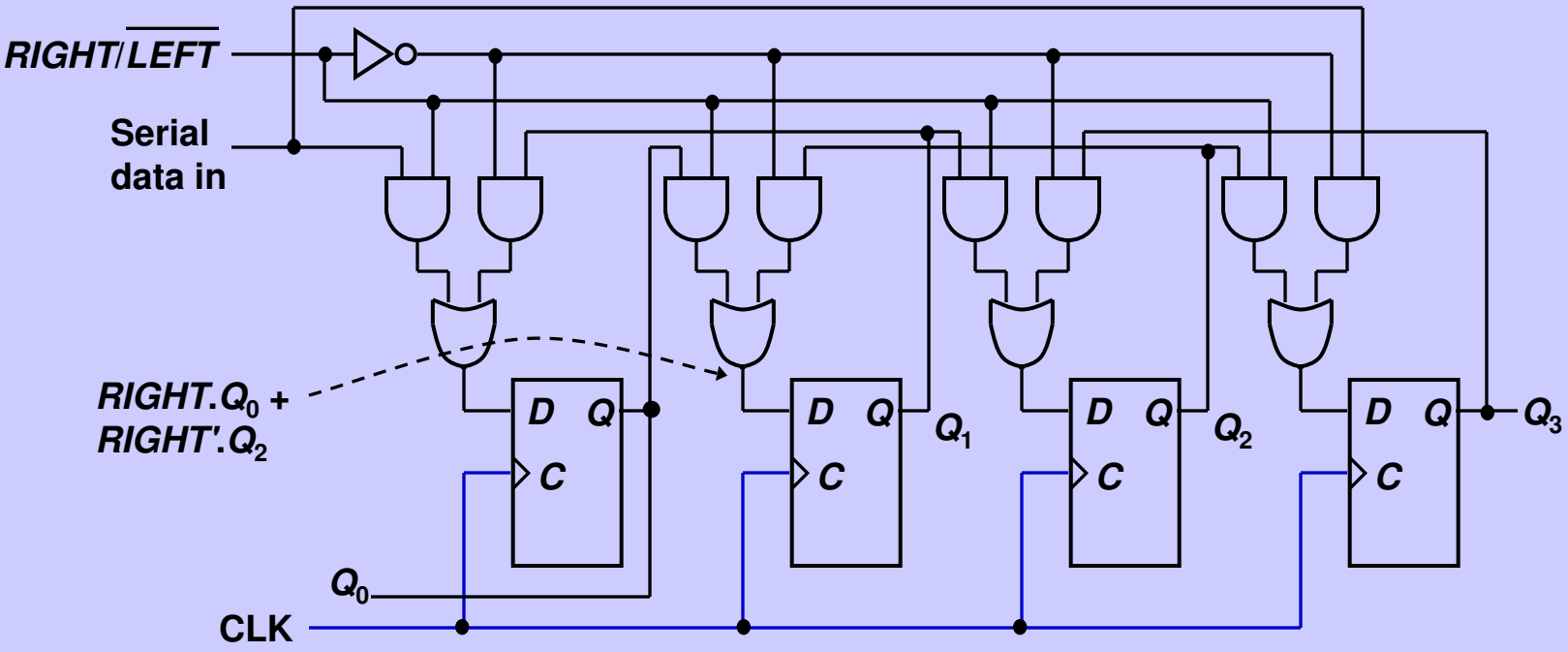
Parallel In/Parallel Out Shift Registers

- Simultaneous input and output of all data bits.



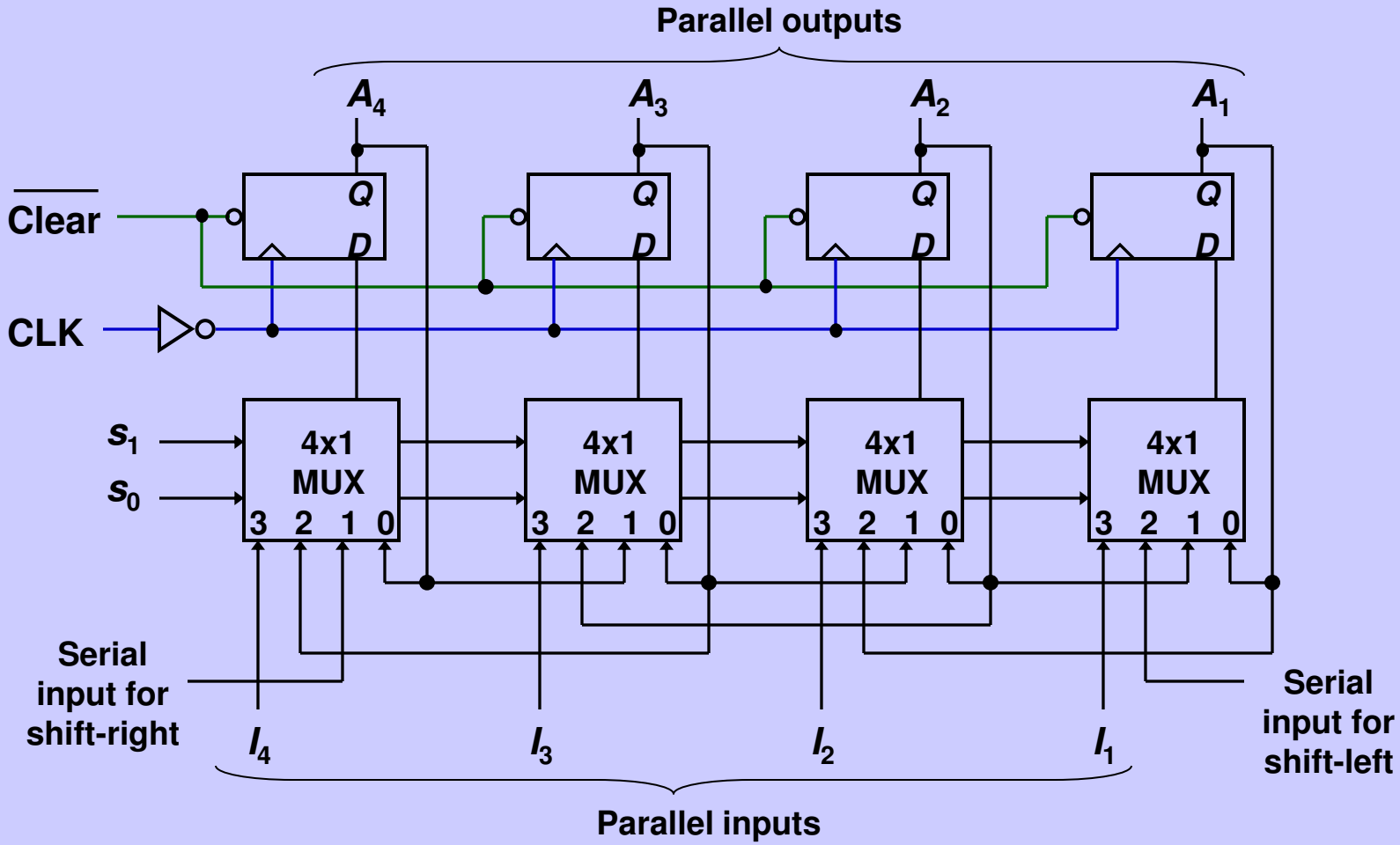
Bidirectional Shift Registers

- Data can be shifted either left or right, using a control line *RIGHT*/$\overline{LEFT}$ (or simply *RIGHT*) to indicate the direction.



Bidirectional Shift Registers

- 4-bit bidirectional shift register with parallel load.



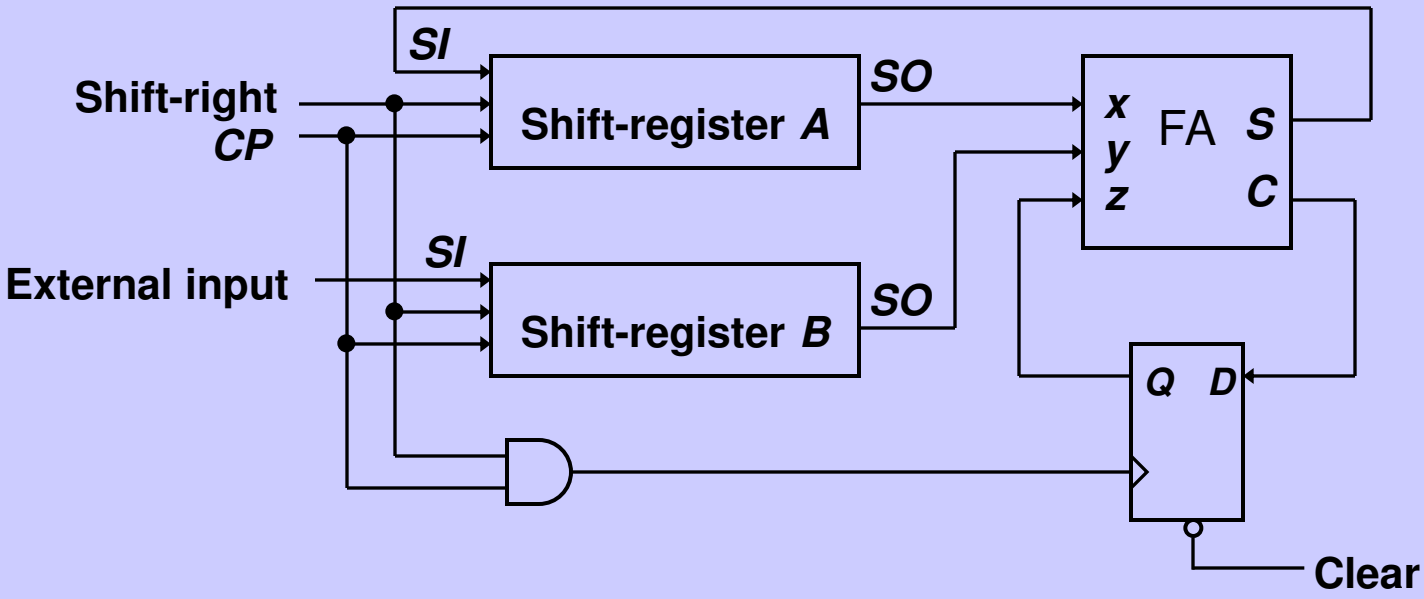
Bidirectional Shift Registers

- 4-bit bidirectional shift register with parallel load.

Mode Control		*Register Operation*
s_1	s_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

An Application – Serial Addition

- Most operations in digital computers are done in parallel. Serial operations are slower but require less equipment.
- A serial adder is shown below. $A \leftarrow A + B$.



An Application – Serial Addition

- $A = 0100$; $B = 0111$. $A + B = 1011$ is stored in A after 4 clock pulses.

Initial:		A: 0 1 0 0 B: 0 1 1 1	Q: 0
Step 1:	**0 + 1 + 0** **S = 1, C = 0**	A: 1 0 1 0 B: x 0 1 1	Q: 0
Step 2:	**0 + 1 + 0** **S = 1, C = 0**	A: 1 1 0 1 B: x x 0 1	Q: 0
Step 3:	**1 + 1 + 0** **S = 0, C = 1**	A: 0 1 1 0 B: x x x 0	Q: 1
Step 4:	**0 + 0 + 1** **S = 1, C = 0**	A: 1 0 1 1 B: x x x x	Q: 0