Chap 6
Registers

Hanwool Jeong

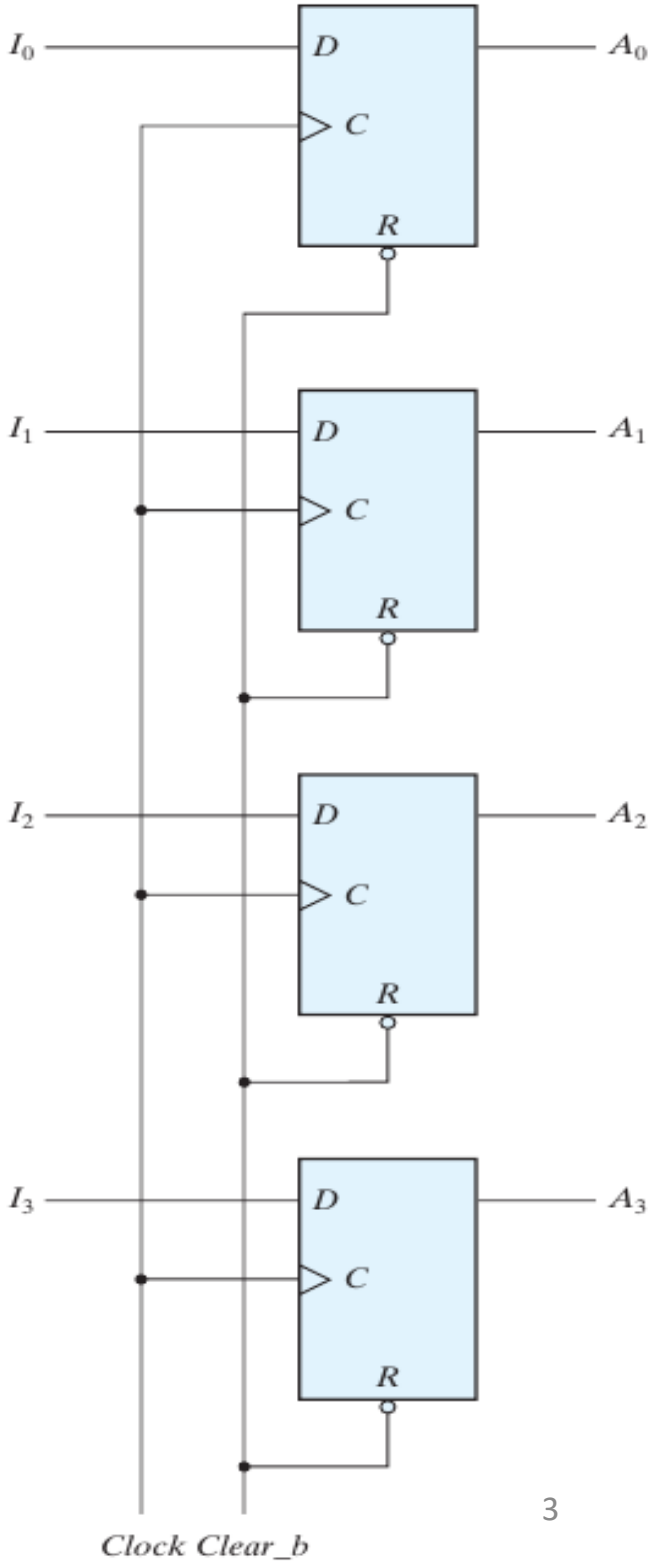
hwjeong@kw.ac.kr

Introduction

- A register is a group of FFs controlled by the common clock.
- N-bit register consists of n FFs capable of storing n bits.
- A register consists of a group of flip-flops together with gates defining how the information is transferred into the register.

Simplest Register Example

- 4-bit data storage register.
  Only FFs without any gate.
- (I_3 , I_2 , I_1 , I_0) immediately before CLK edge determines the value of (A_3, A_2 , A_1 , A_0) after CLK edge.
- Clear_b is asynchronous control signal

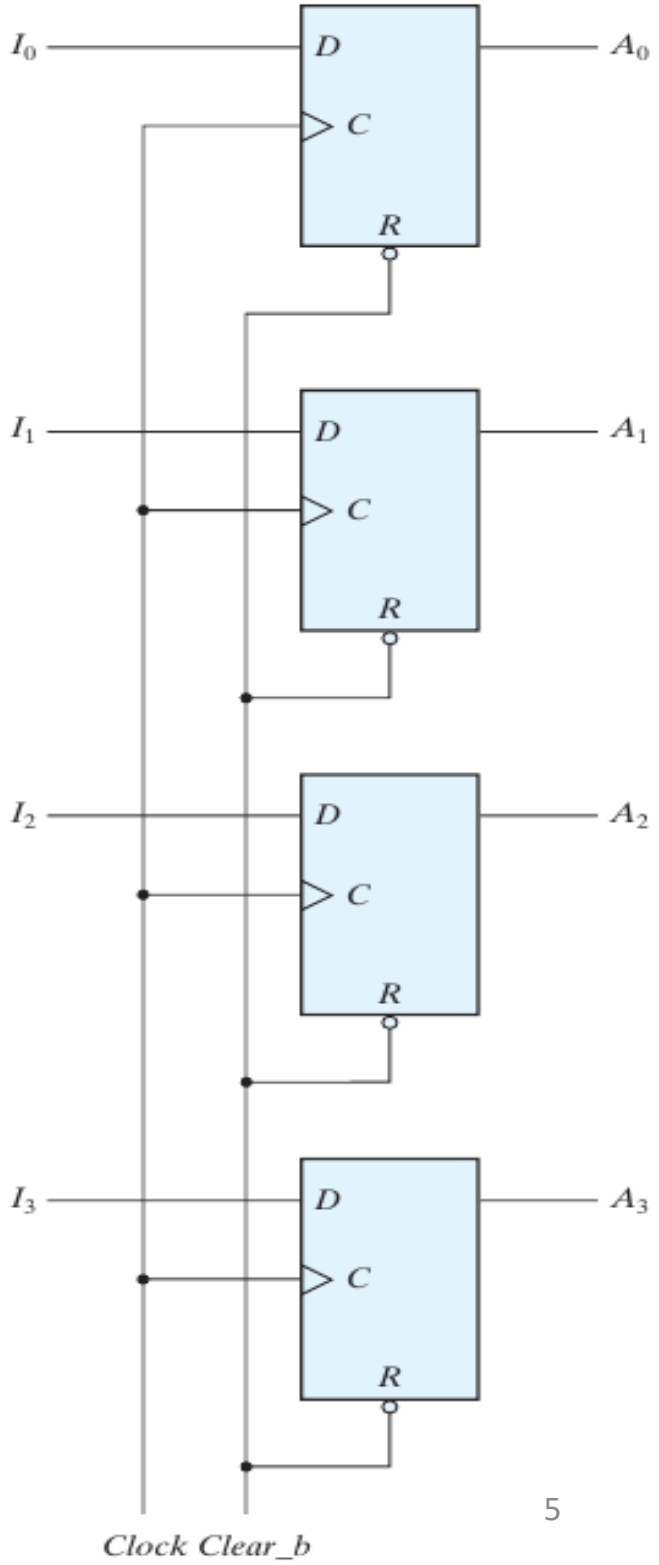


Problem of Clock Gating

- Suppose that you want to hold the data for several clock periods, that is, not update data. How can you realize this?
- CLK can be inhibited from reaching the register by controlling the clock input with an enabling gate ➔ **clock gating**
- However, it is ill advised because uneven propagation delays are produced between the master clock and the inputs of FFs.
- To fully synchronize the system, we must ensure that all clock pulses arrive at the same time anywhere in the system.
- Nevertheless, clock gating is sometimes used when it is necessary.

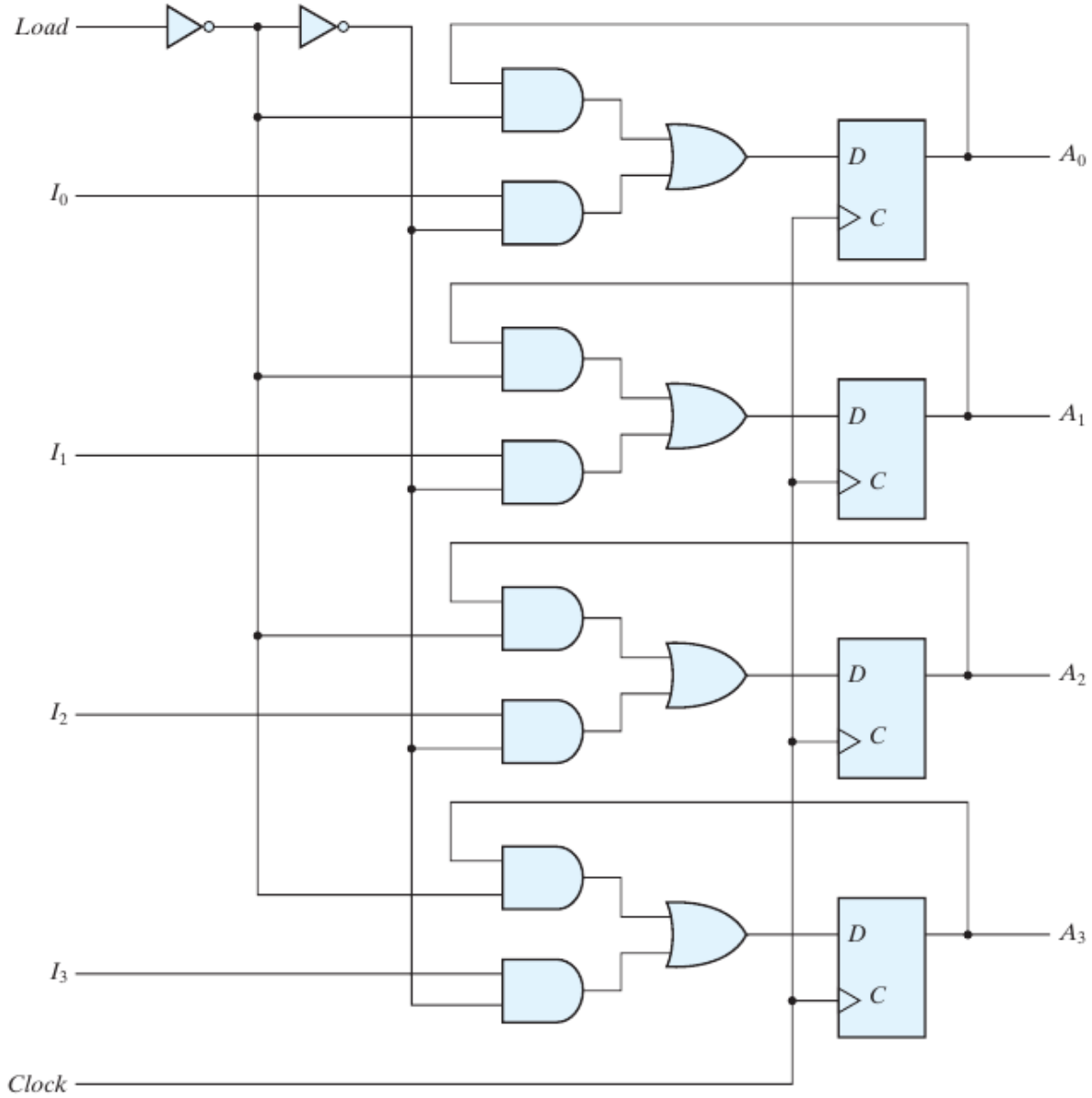
Try to Design a Circuit

- How can you prevent from the register being updated when required? Like,
- When “NoUpdate = 1” or “Hold = 1”, (A_3, A_2 , A_1 , A_0) is kept unchanged.



Four-bit register with parallel load

- The transfer of new information into a register is referred to as **loading** or **updating** the register.
- How can you keep the state of register unchanged?

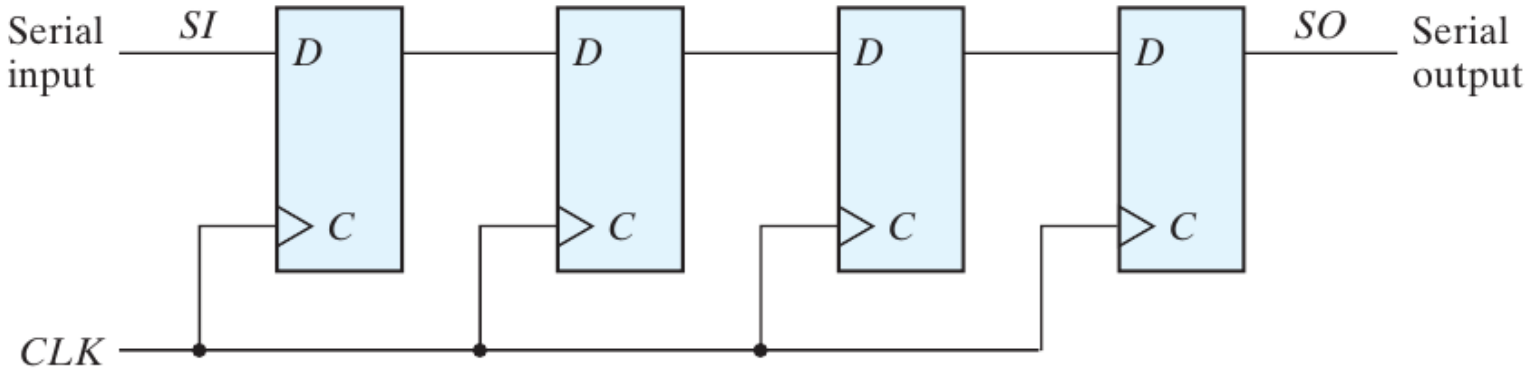


Limitation of Hardware

- Suppose that you want to multiply two 64 bit binary data. Thus, you fabricated chip including 64 bit multiplier.
- You have only 10 external pins due to space limit.
- In this condition, how can you apply two 64 bit inputs and monitor 128 bit output externally?

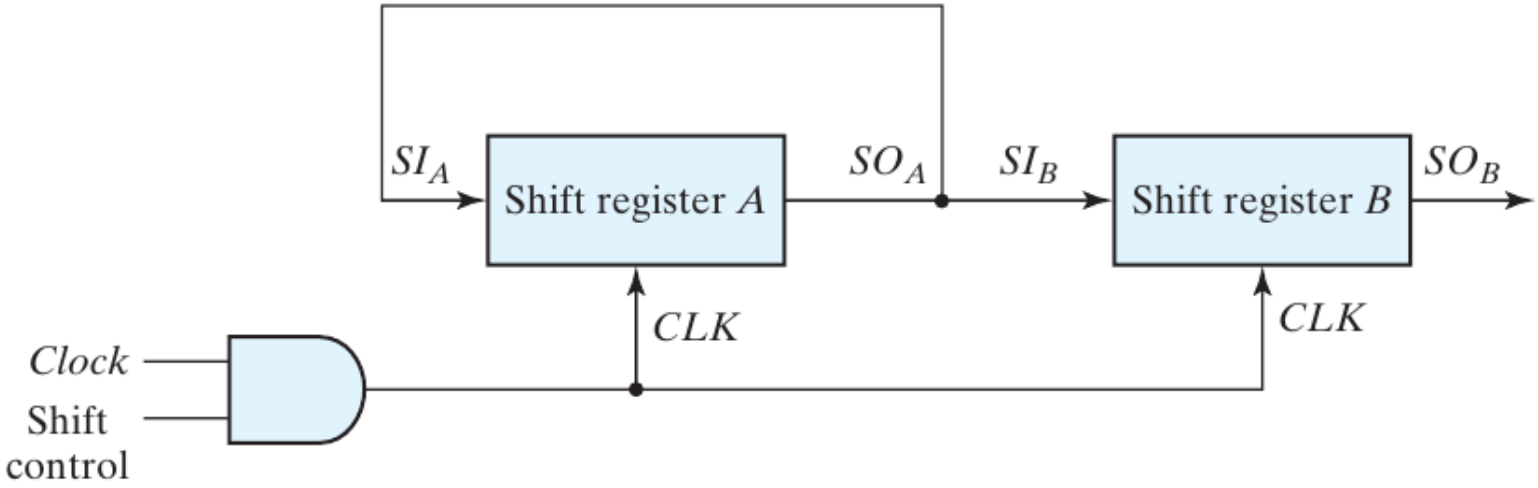
Shift Register

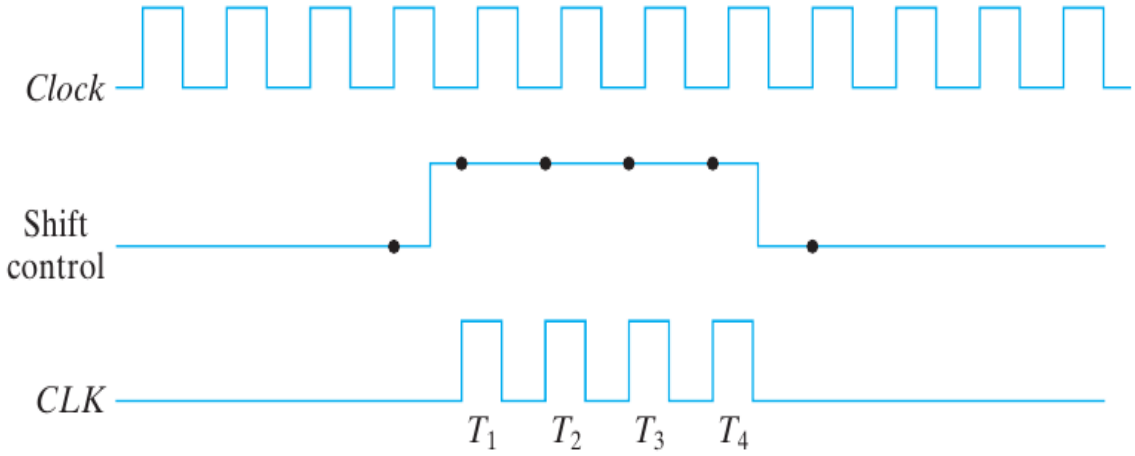
- Each clock pulse shifts the contents of the register one bit position to the right.



- Is there any way that we can suspend the shift action w/o gating clock? Or making it shift only when ShiftControl = 1.

Serial Transfer





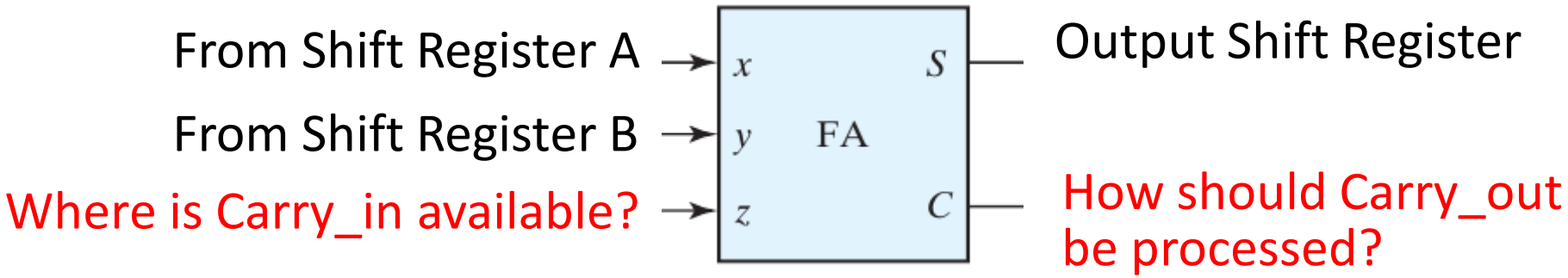
Timing Pulse	Shift Register *A*				Shift Register *B*			
Initial value	1	0	1	1	0	0	1	0
After T_1	1	1	0	1	1	0	0	1
After T_2	1	1	1	0	1	1	0	0
After T_3	0	1	1	1	0	1	1	0
After T_4	1	0	1	1	1	0	1	1

Serial Addition

- As previously mentioned, due to requirement for less silicon area on a chip, serial operation is often used.
- Not only number of external pins, but also the # of circuit components inside the chip can be minimized.
- For example, you want to add two 64bit data with only one FA. Can you imagine it?

Focusing on 1-bit FA

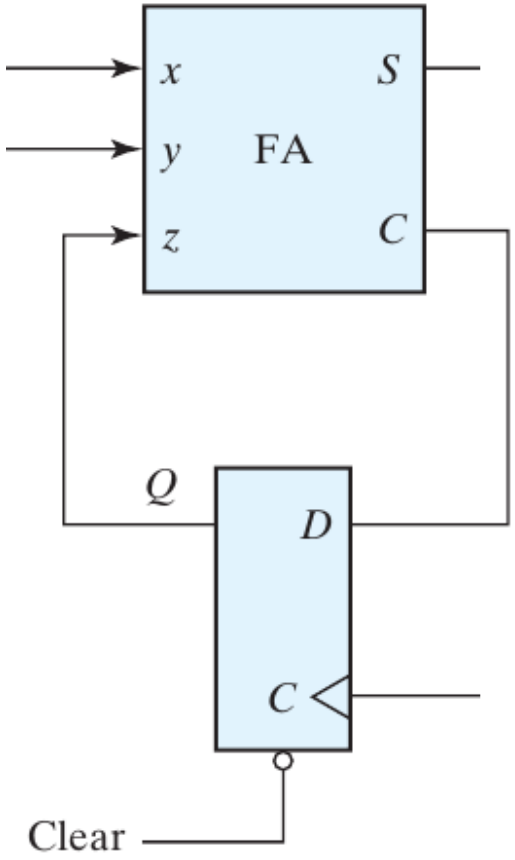
- With the assumption that the two data to be added will be supplied serially, we can focus on 1-bit FA as:



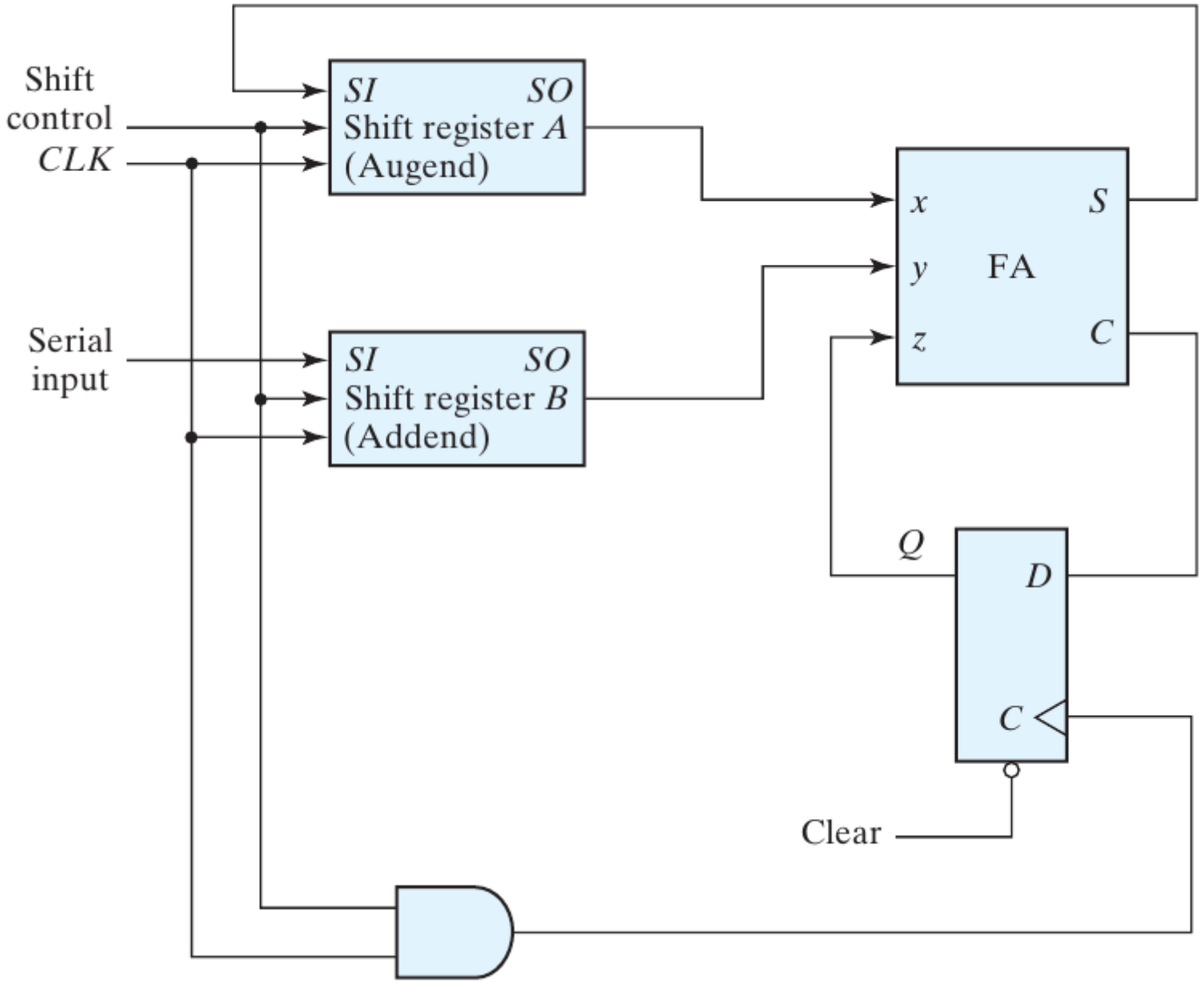
- Carry_out should be transferred to the next phase operation Carry_in ➔ we can use FF for storing “state”!

Employing FF for Carry Processing

- We do not need State table because it is obvious that the previous Carry_out becomes the next state of Carry_in



Present State	Inputs		Next State
Q	x	y	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1


Shift
control
CLK
SI
SO
Shift register A
(Augend)
Serial
input
SI
SO
Shift register B
(Addend)
x
y
z
FA
S
C
Q
D
C
Clear

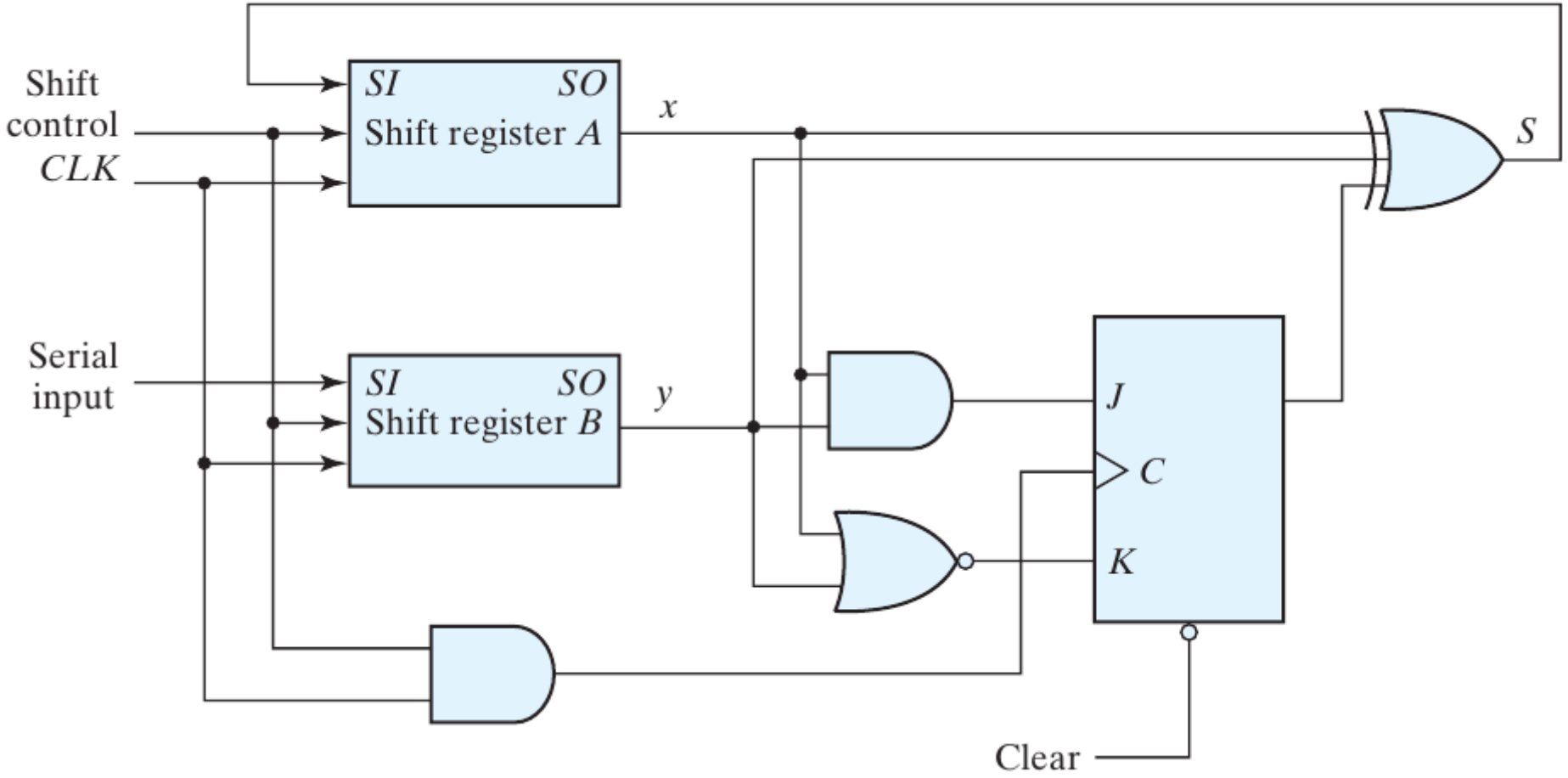
Can you Use JK FF?

- Referring JK-FF excitation table,

Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	x	y	Q	S	J_Q	K_Q
0	0	0	0	0		
0	0	1	0	1		
0	1	0	0	1		
0	1	1	1	0		
1	0	0	0	1		
1	0	1	1	0		
1	1	0	1	0		
1	1	1	1	1		

$$J_Q = xy$$
$$K_Q = x'y' = (x + y)'$$

$$S = x \oplus y \oplus Q$$


Shift
control
CLK
SI
SO
Shift register A
x
S
Serial
input
SI
SO
Shift register B
y
J
C
K
Clear

Some Requirement for Shift Register

- If the flip-flop outputs of a shift register are accessible, then information entered serially by shifting can be taken out in parallel from the outputs of the flip-flops.
- If a parallel load capability is added to a shift register, then data entered in parallel can be taken out in serial fashion by shifting the data stored in the register.
- Some shift registers provide the necessary input and output terminals for parallel transfer. They may also have both shift-right and shift-left capabilities.

The Most General Shift Register

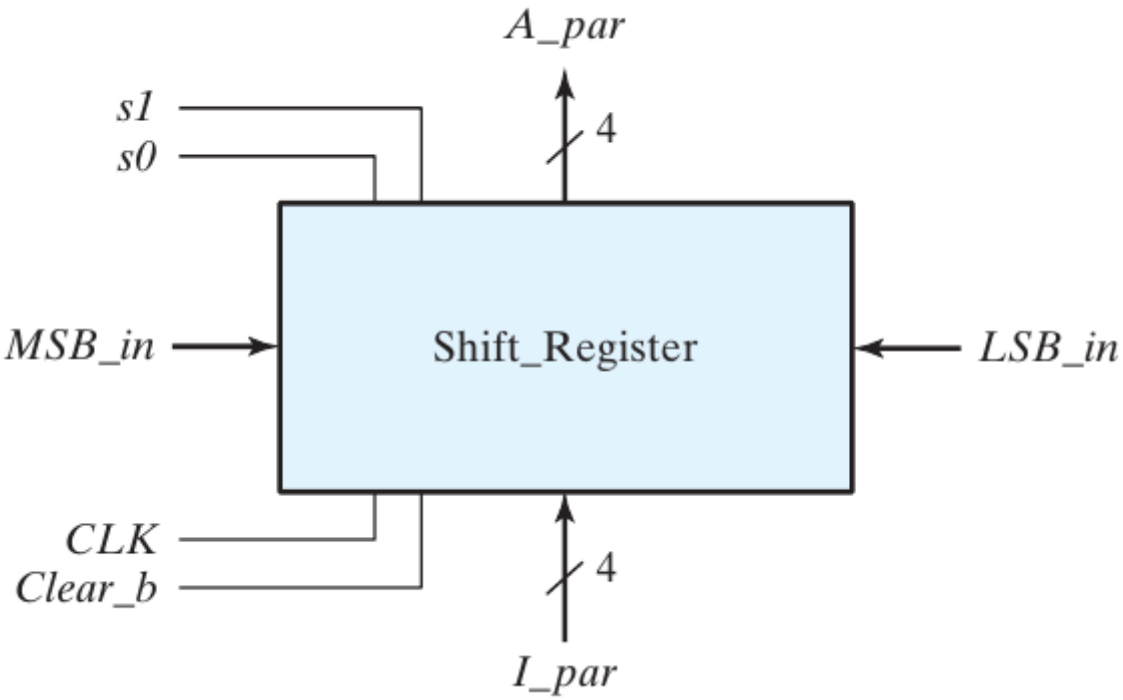
1. A clear control to clear the register to 0.
2. A clock input to synchronize the operations.
3. A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift right.
4. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift left.
5. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
6. *n* parallel output lines.
7. A control state that leaves the information in the register unchanged in response to the clock. Other shift registers may have only some of the preceding functions, with at least one shift operation.

Universal Shift Register

- A register capable of shifting in one direction only is a unidirectional shift register.
- One that can shift in both directions is a bidirectional shift register.
- If the register has both shifts and parallel-load capabilities, it is referred to as a **universal shift register**.

How Would be Look Like?

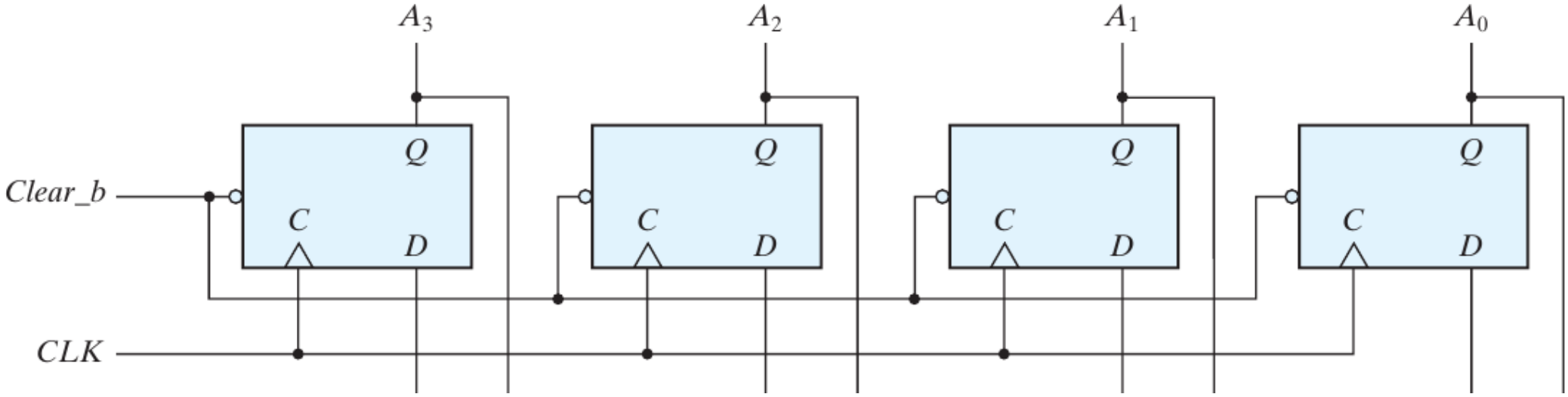
- There are various modes: s1, s0 are mode control signal



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

Designing Four-Bit Universal Shift Register; Q respect

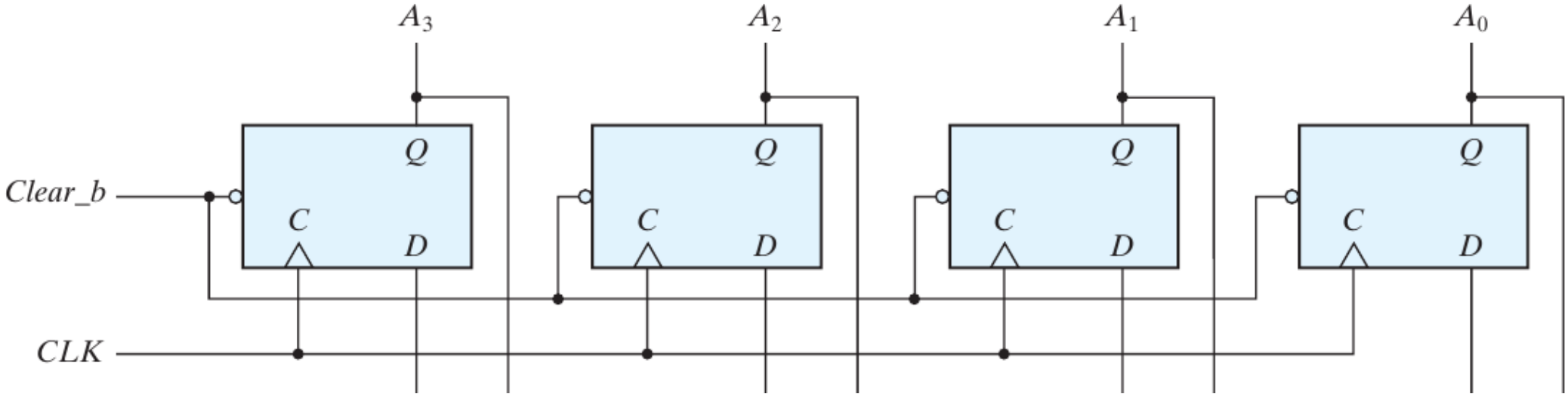
- You can start from:



- For each FF, Q should be connected to
  - External world (for what?)
  - Left FF (for what?)
  - Right FF (for what?)
  - Itself FF (for what?)

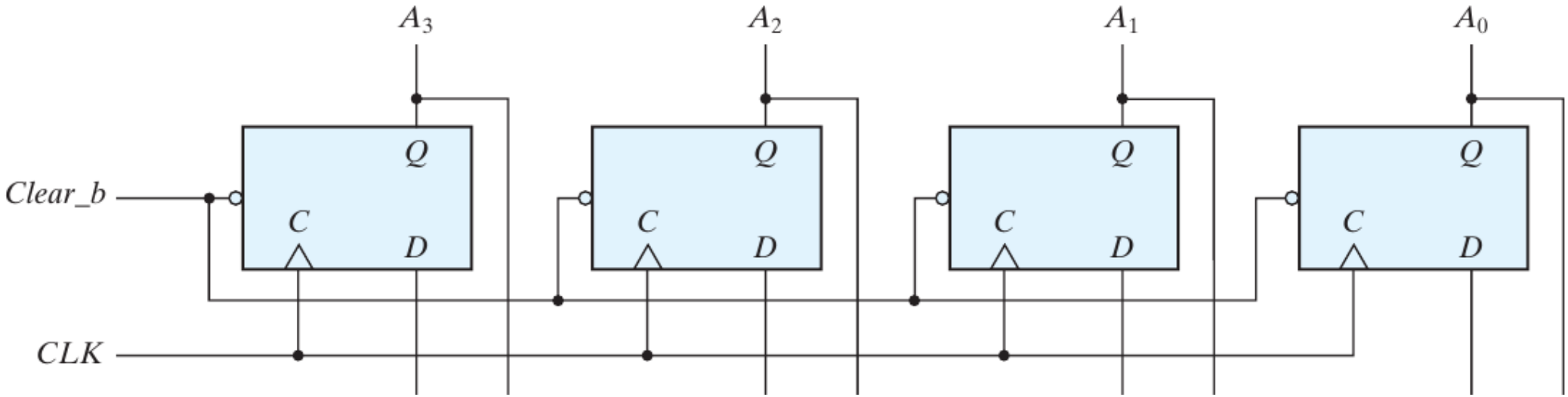
Designing Four-Bit Universal Shift Register; D respect

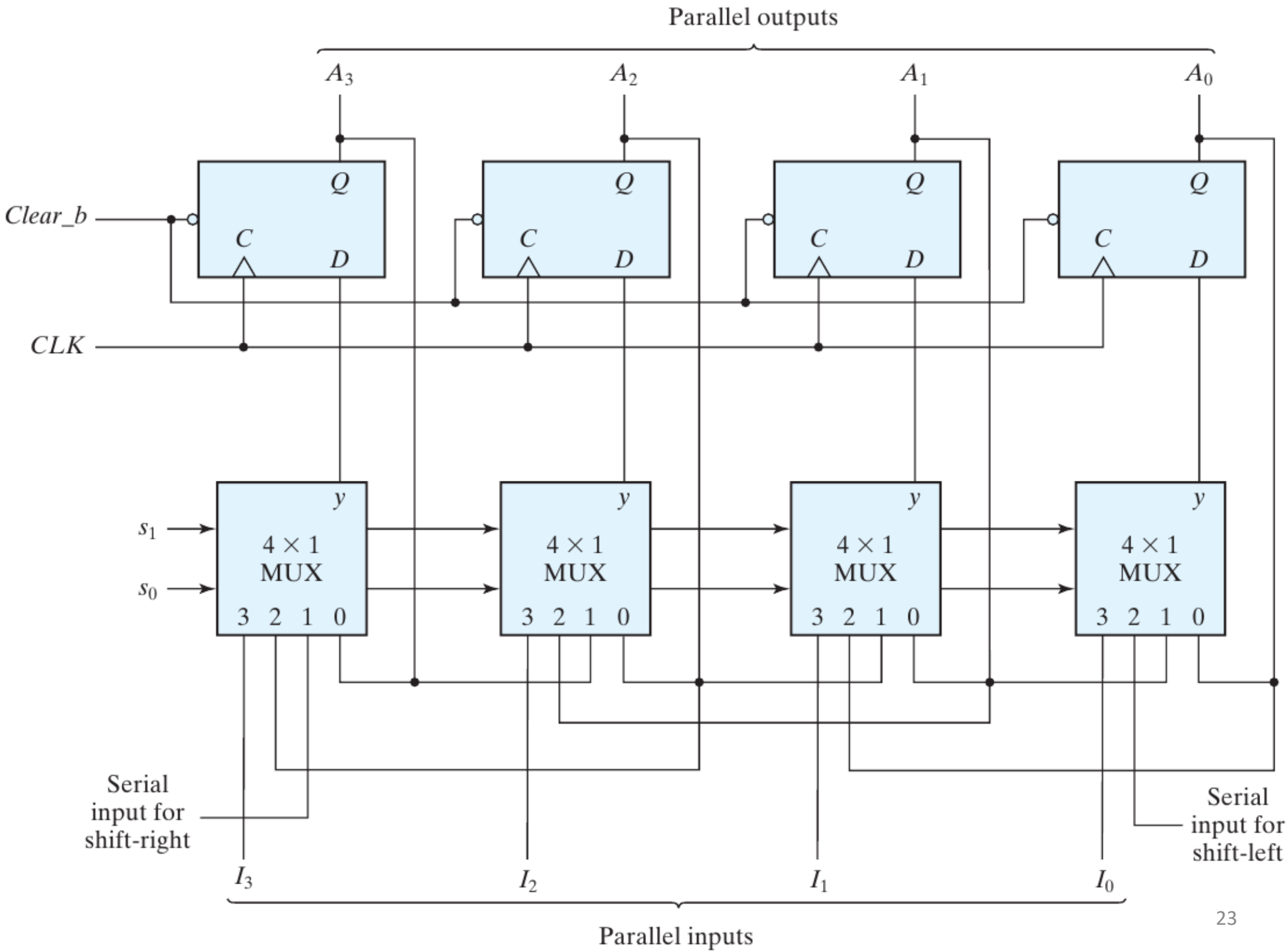
- You can start from:



- For each FF, D should be connected from
  - External world (for what?)
  - Left FF (for what?)
  - Right FF (for what?)
  - Itself FF (for what?)

How Can You Differentiate Connectivity or Configuration According to Modes?




Parallel outputs
A_3
A_2
A_1
A_0
Q
C
D
Clear_b
CLK
y
s_1
s_0
4 × 1
MUX
3 2 1 0
Serial
input for
shift-right
Serial
input for
shift-left
I_3
I_2
I_1
I_0
Parallel inputs