

EE2000 Logic Circuit Design

Sequential Functional Blocks:
Registers and Counters

Outline

9.1 Registers

- Registers

 - Parallel load, Bidirectional shift

 - Serial Adder, Accumulator

9.2 Counters

- Asynchronous Counters

 - Ripple counters

- Synchronous Counters

 - Up-down counters,

 - Modulo counters,

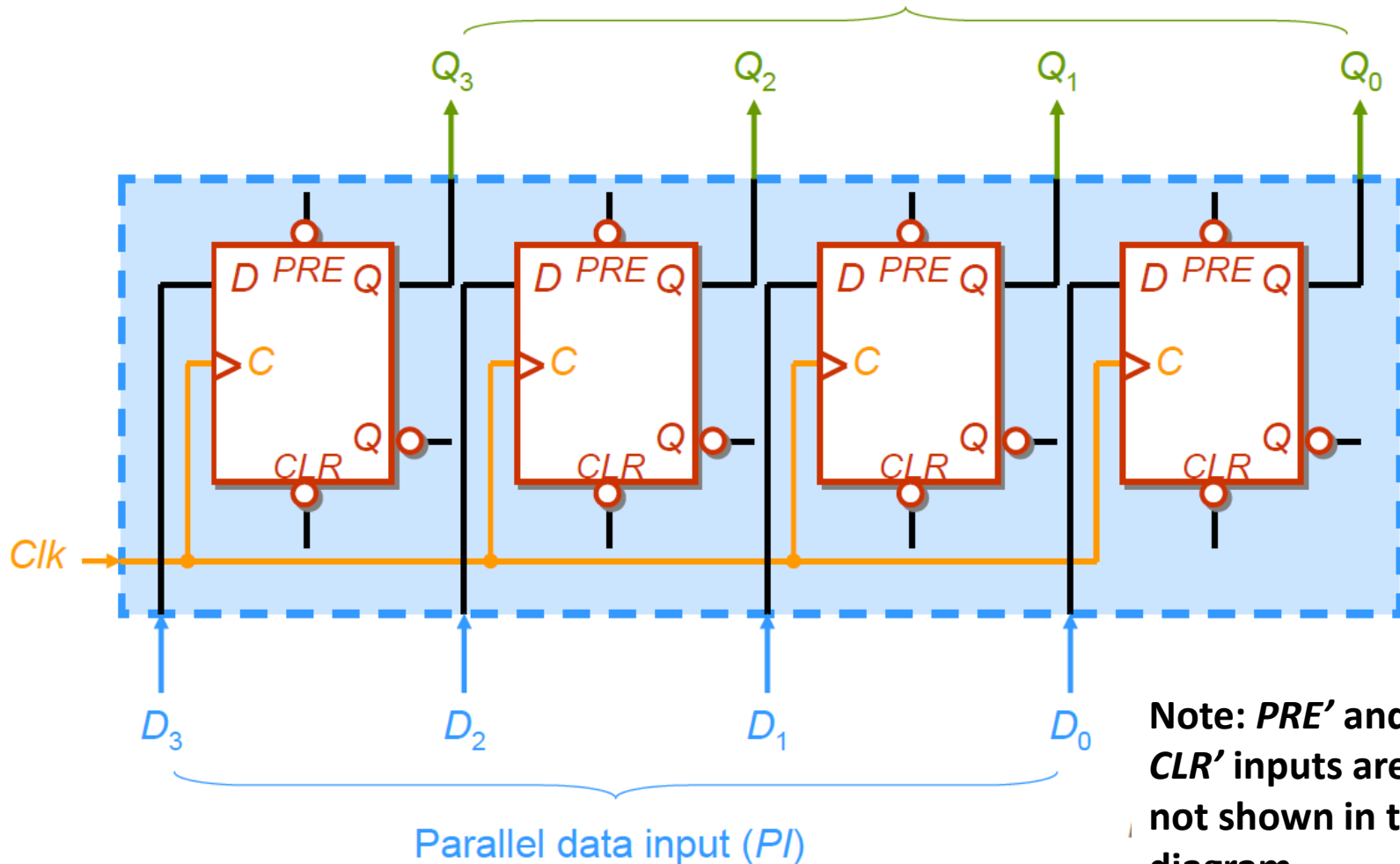
 - Ring counters, Twist Ring counters

Registers

- Memory used for storing (binary) information during the processing of data
- To store n -bit data, require n flip-flops
- n -bit register
 - Composed of a set of n edge-triggered flip-flops
 - The clock pulse is the enable signal
 - Data will be latched to the flip-flops during the clock active period (pos. edge or neg. edge)
- Two designs
 - Parallel, or Serial

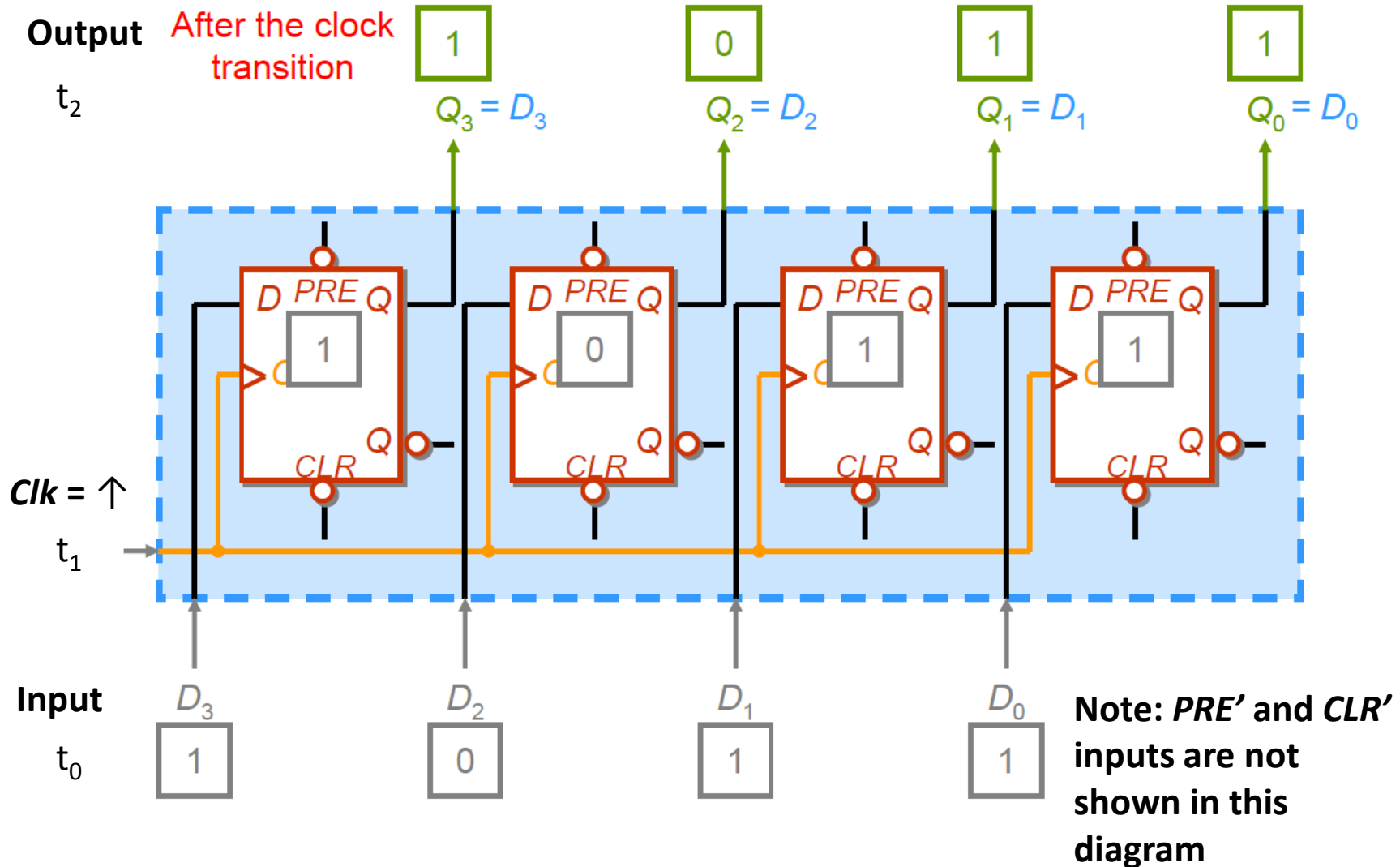
4-bit Register

Parallel data output (*PO*)



Note: *PRE'* and *CLR'* inputs are not shown in this diagram

Example



PET *D* Flip-Flops with Enable

■ The truth table

Inputs			Outputs	
<i>D</i>	<i>Clk</i>	<i>Load</i>	Next <i>Q</i>	Operation
x	↓, 0, 1	X	<i>Q</i>	Disable
x	↑	0	<i>Q</i>	Hold
x	↑	1	<i>D</i>	Parallel Load

■ The full truth table is

■ $D_A(Q_A, D, Load)$

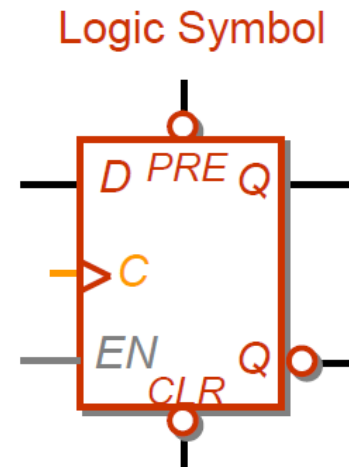
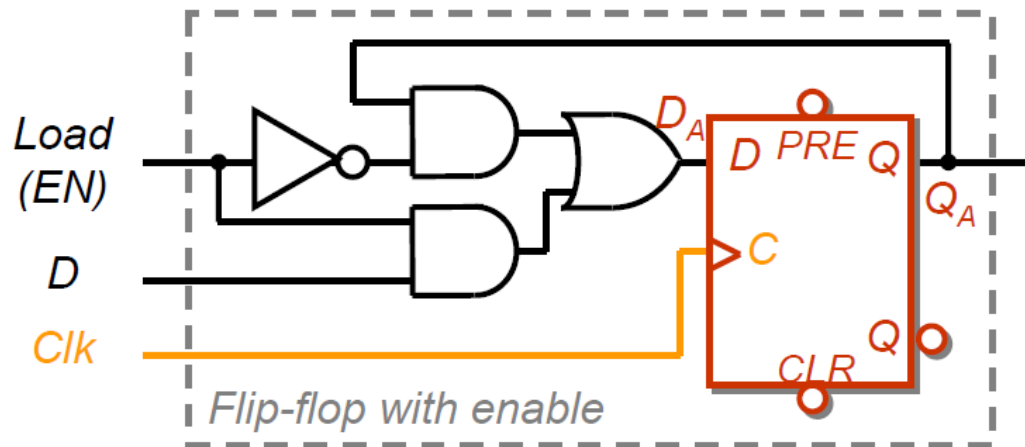
■ $= \sum m(3, 4, 6, 7)$

■ $= Load' \cdot Q_A + Load \cdot D$

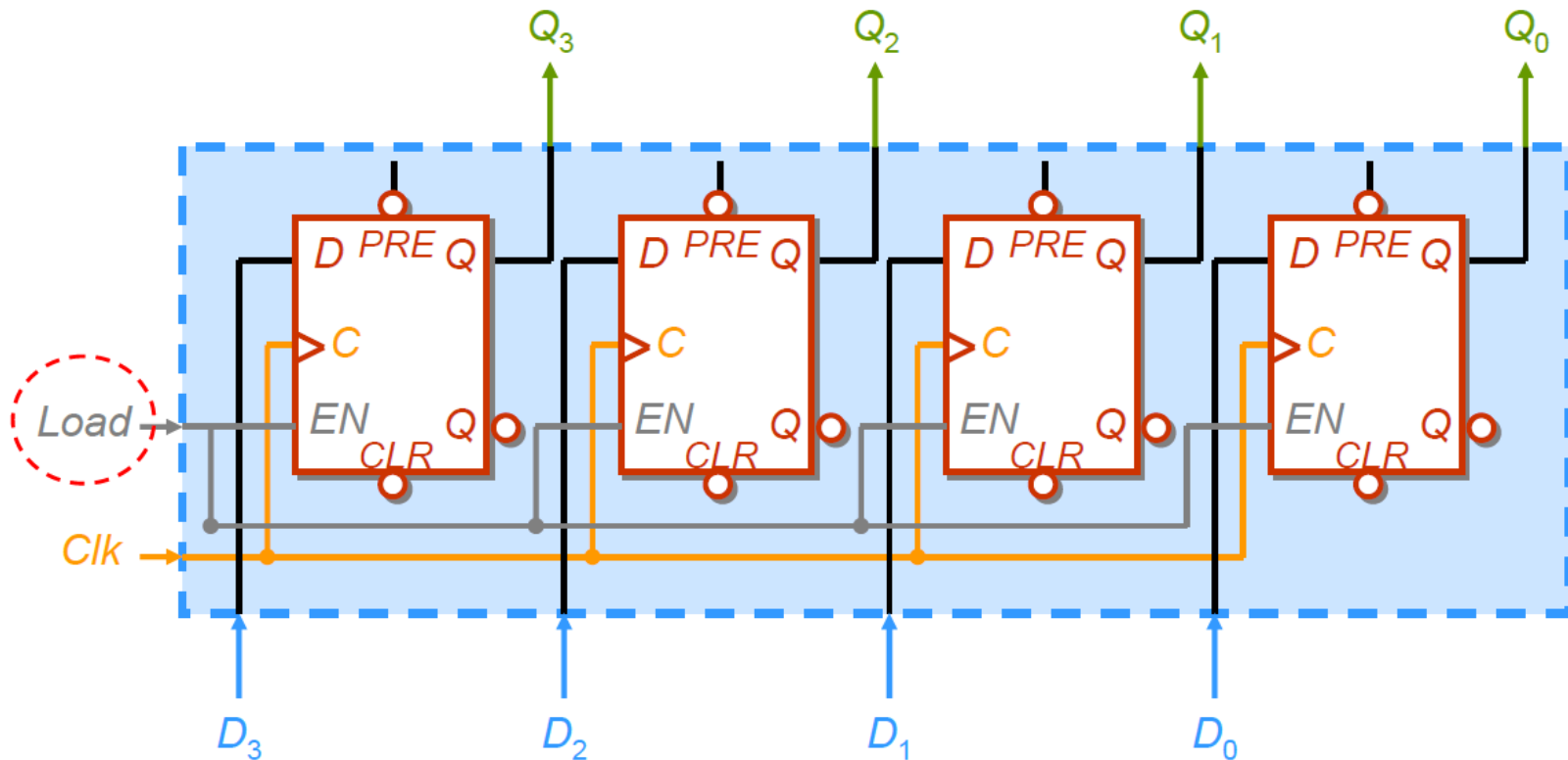
PS	Inputs		NS	FF
<i>Q_A</i>	<i>D</i>	<i>Load</i>	<i>Q_A</i>	<i>D_A</i>
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

PET *D* Flip-Flops with Enable

■ $D_A = Load' \cdot Q_A + Load \cdot D$



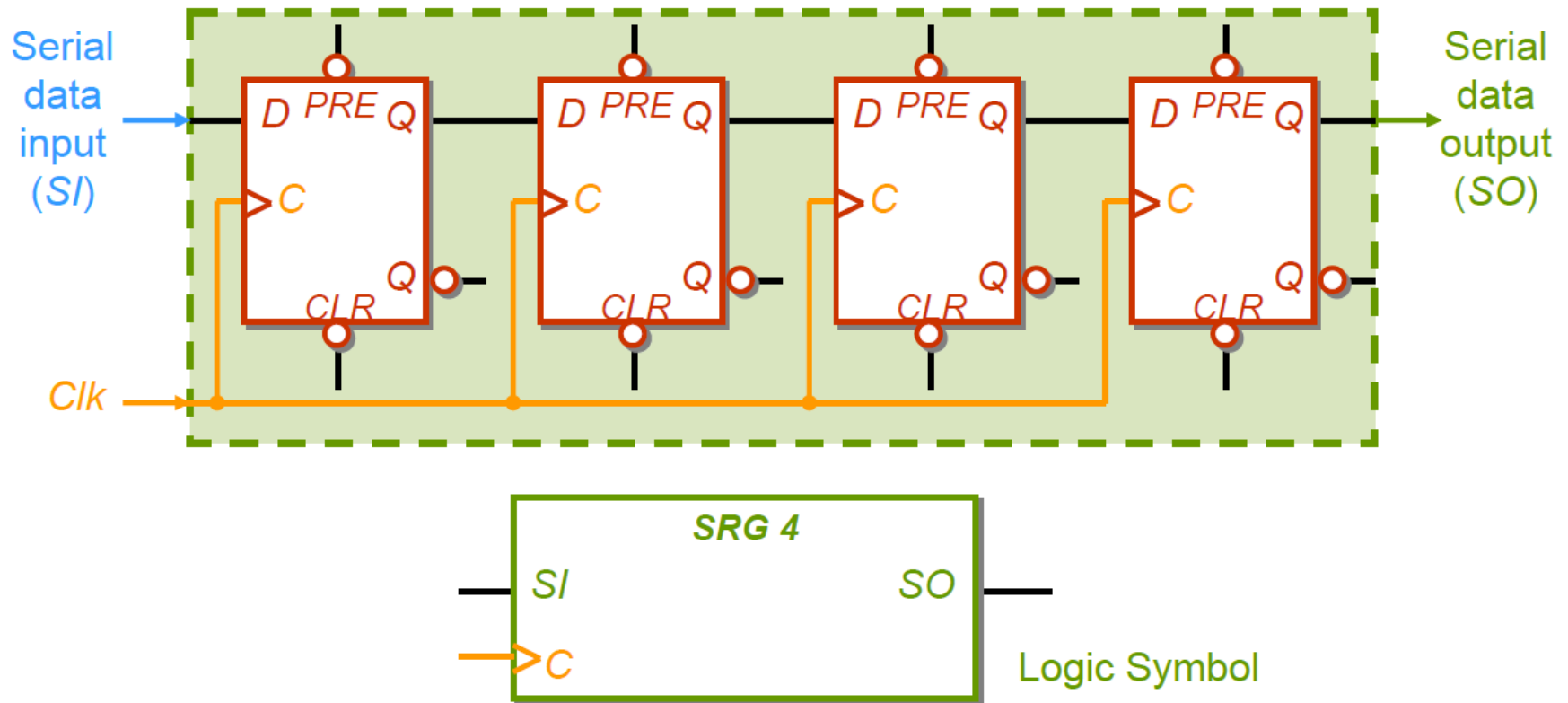
Register with Parallel Load



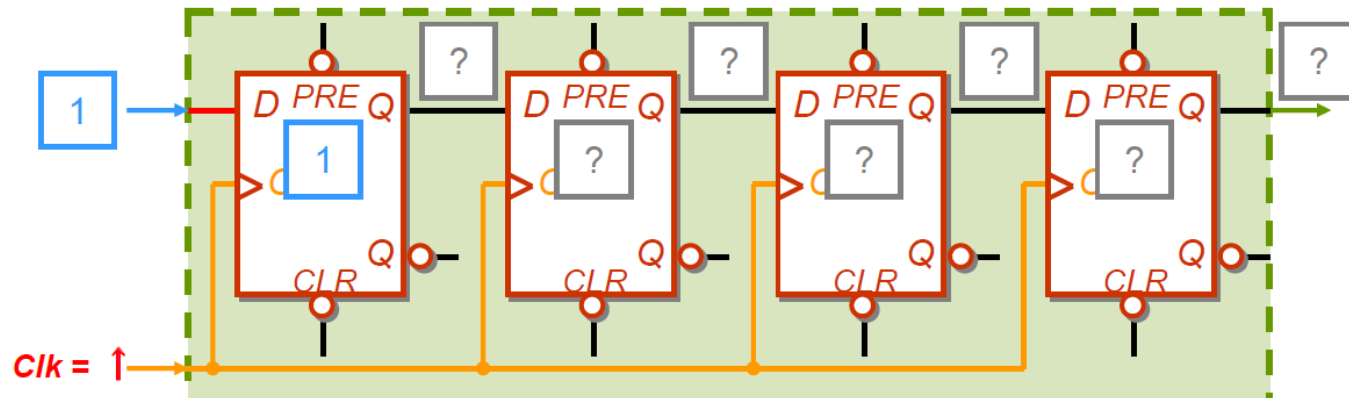
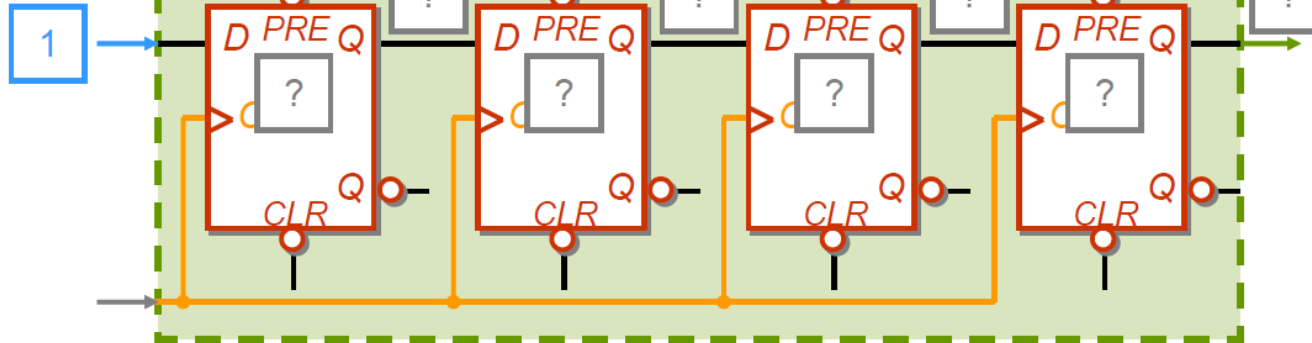
The register will load the inputs into the flip-flops only when Clk is at the rising edge AND $Load = 1$

Shift Registers

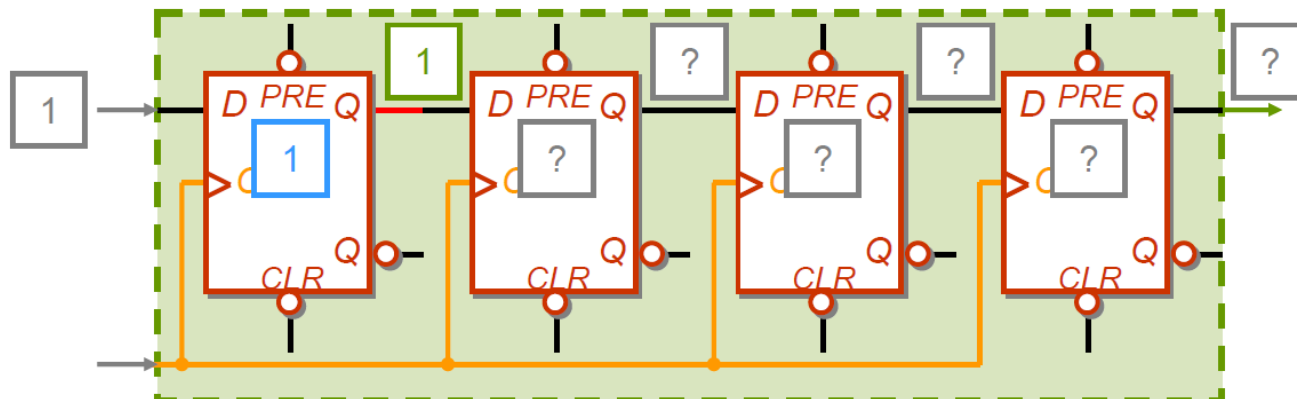
■ A 4-bit Serial-In-Serial-Out Register



The first input bit

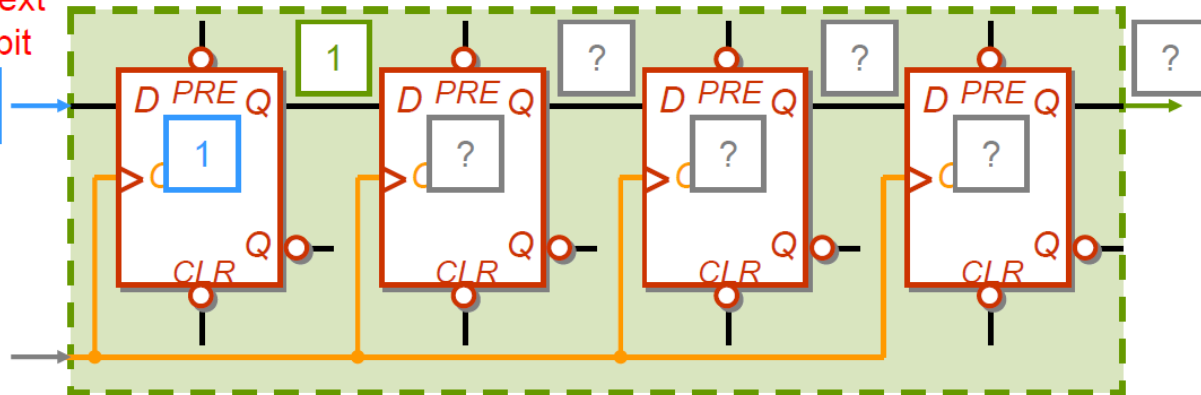


After the clock transition



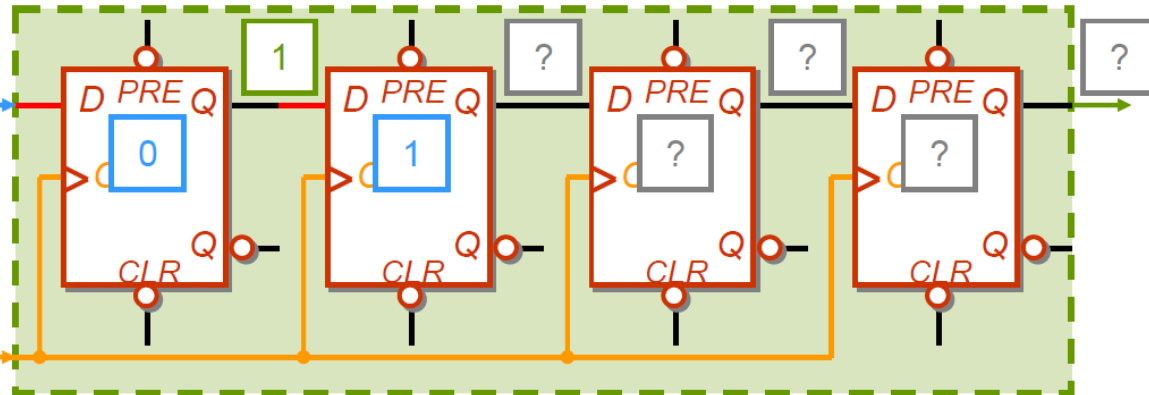
The next input bit

0

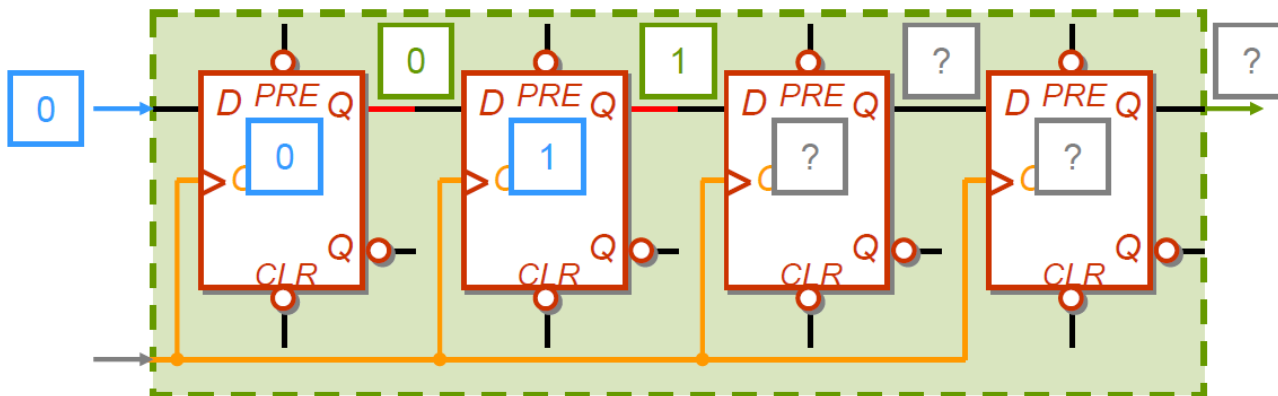


0

Clk = ↑



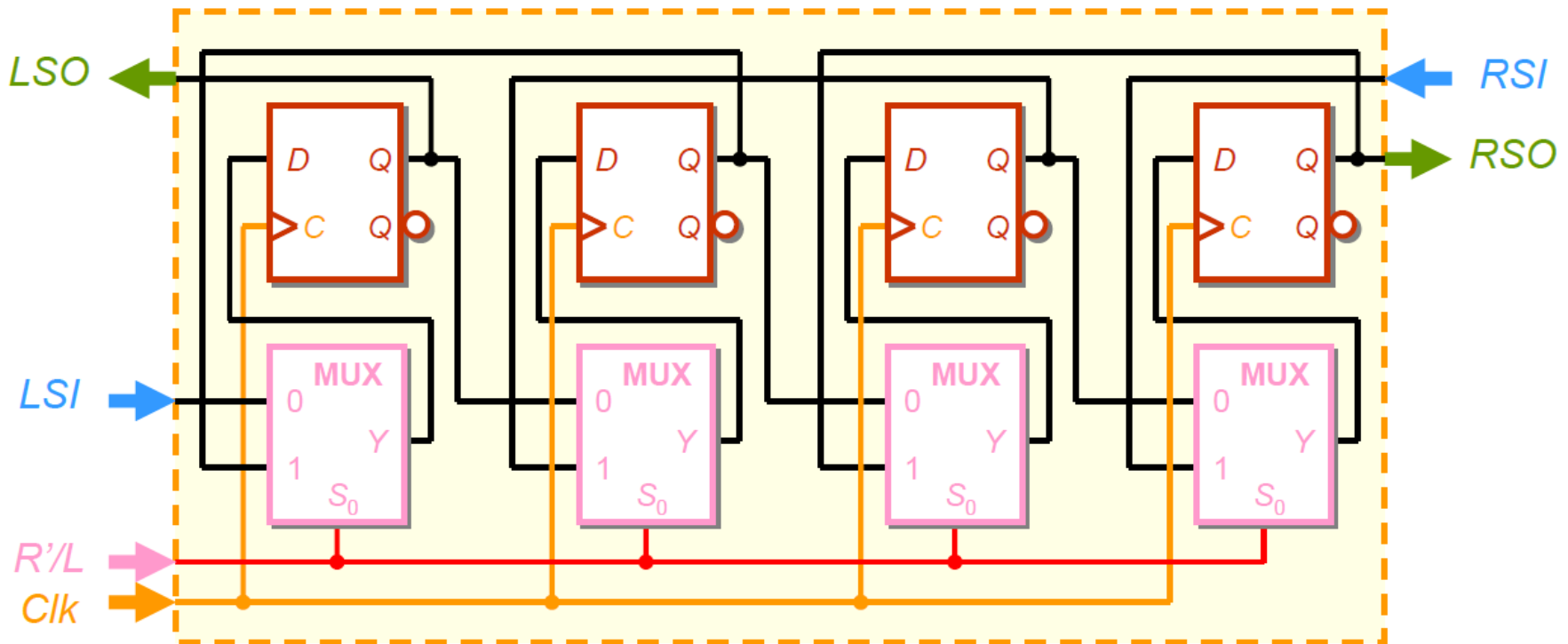
After the clock transition



Shift Registers

- A chain of flip-flops
- When the flip-flops are triggered, the stored bits will **shift** to the next flip-flops
- A first-in-first-out (FIFO) structure
- How about shift register + two shifting directions?
 - Bidirectional Shift Register

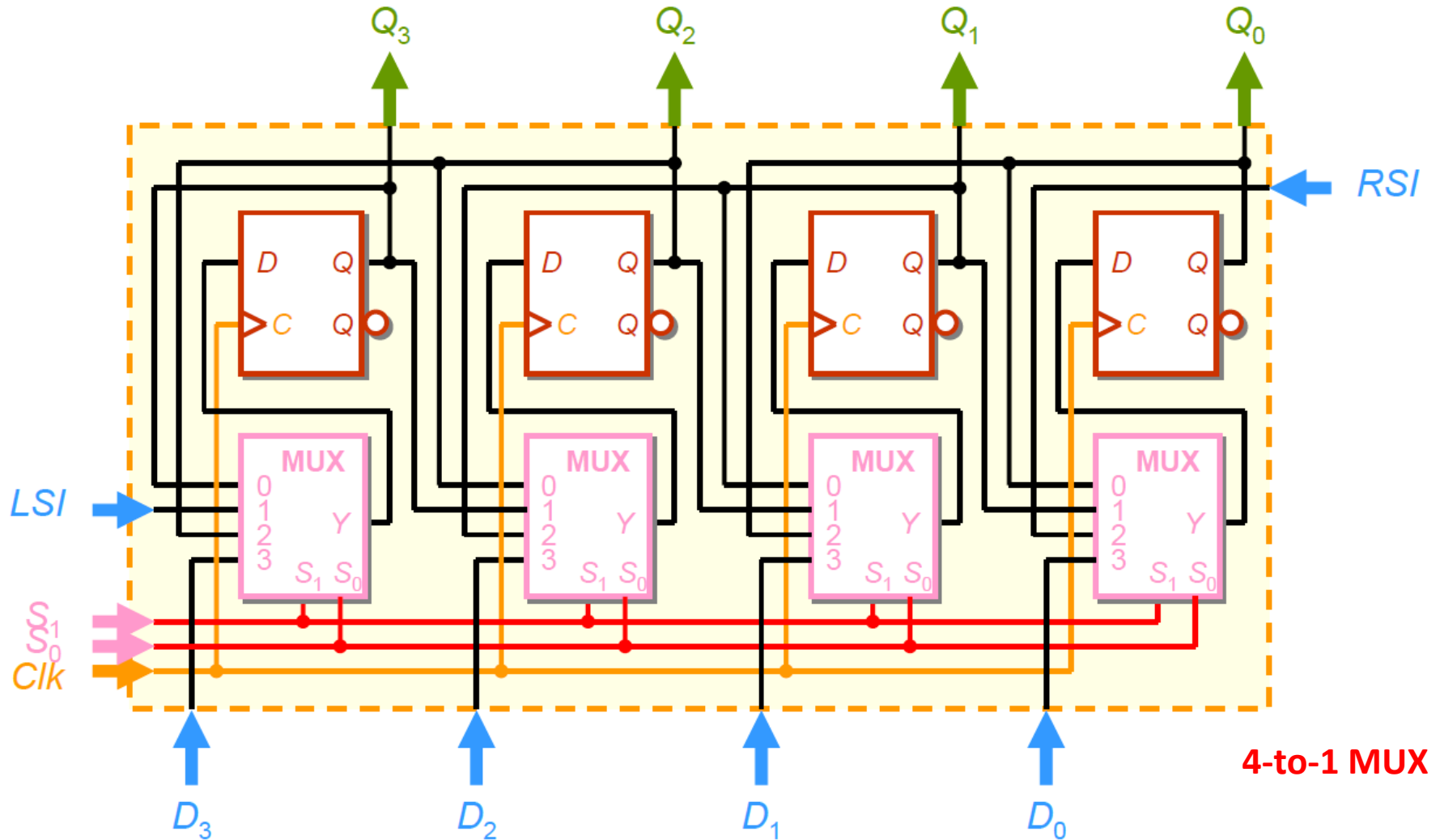
Bidirectional Shift Registers



- When $R' = 0$, the stored bits shift to right
- When $R' = 1$, the stored bits shift to left

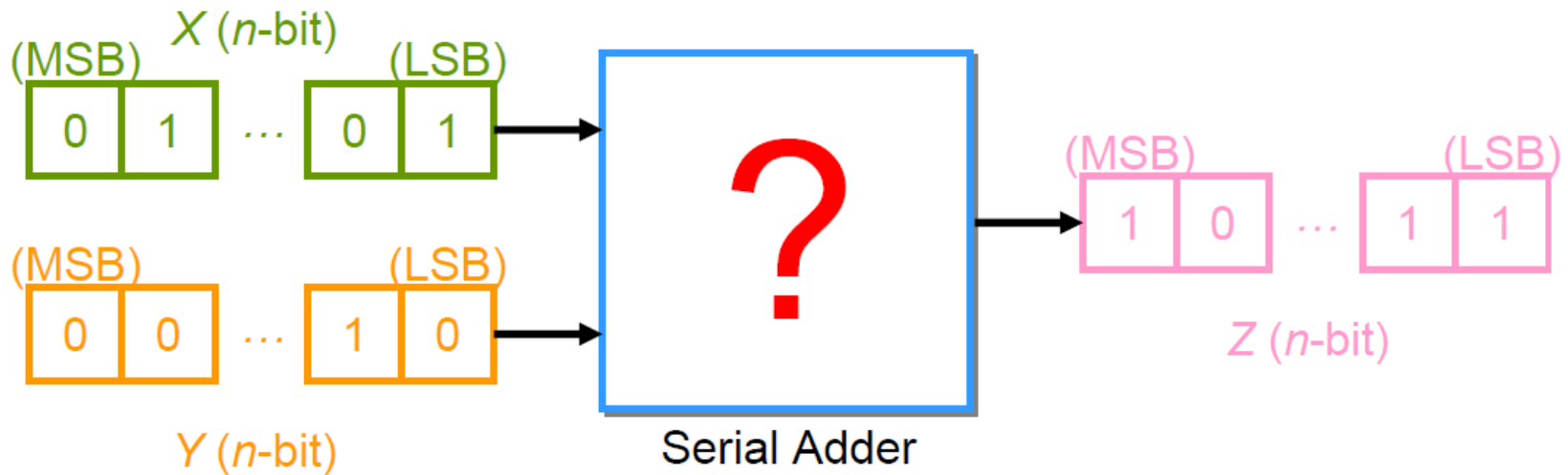
2-to-1 MUX

Bidirectional Shift Register with Parallel Load



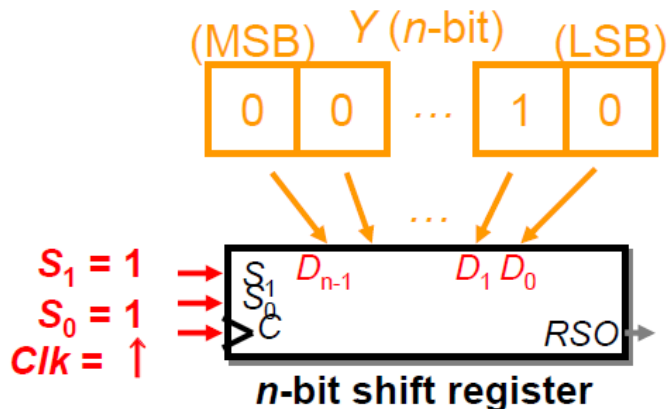
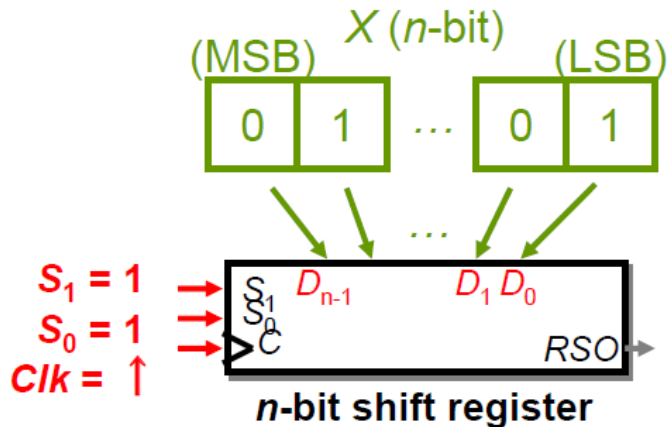
Serial Binary Adder

- Perform addition
- Strings of inputs and output



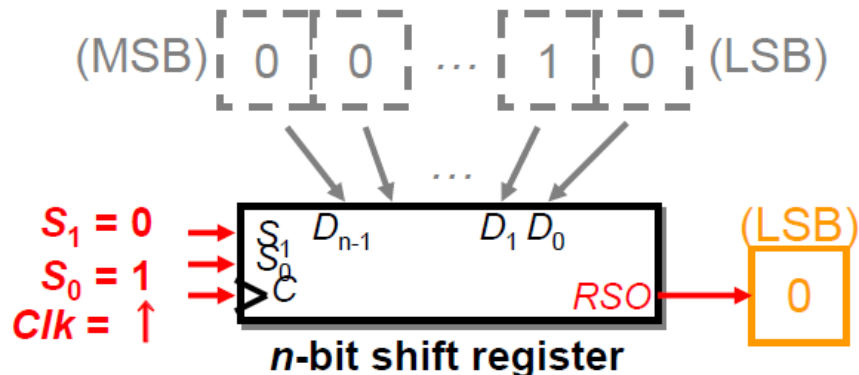
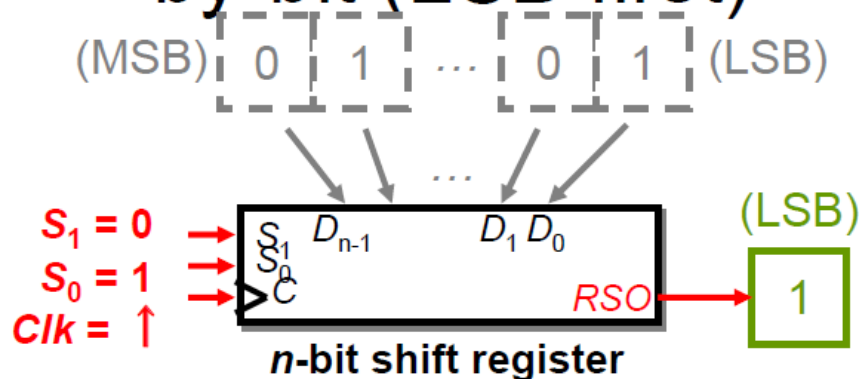
Serial Binary Addder

- 1) Parallel load the binary number X and Y



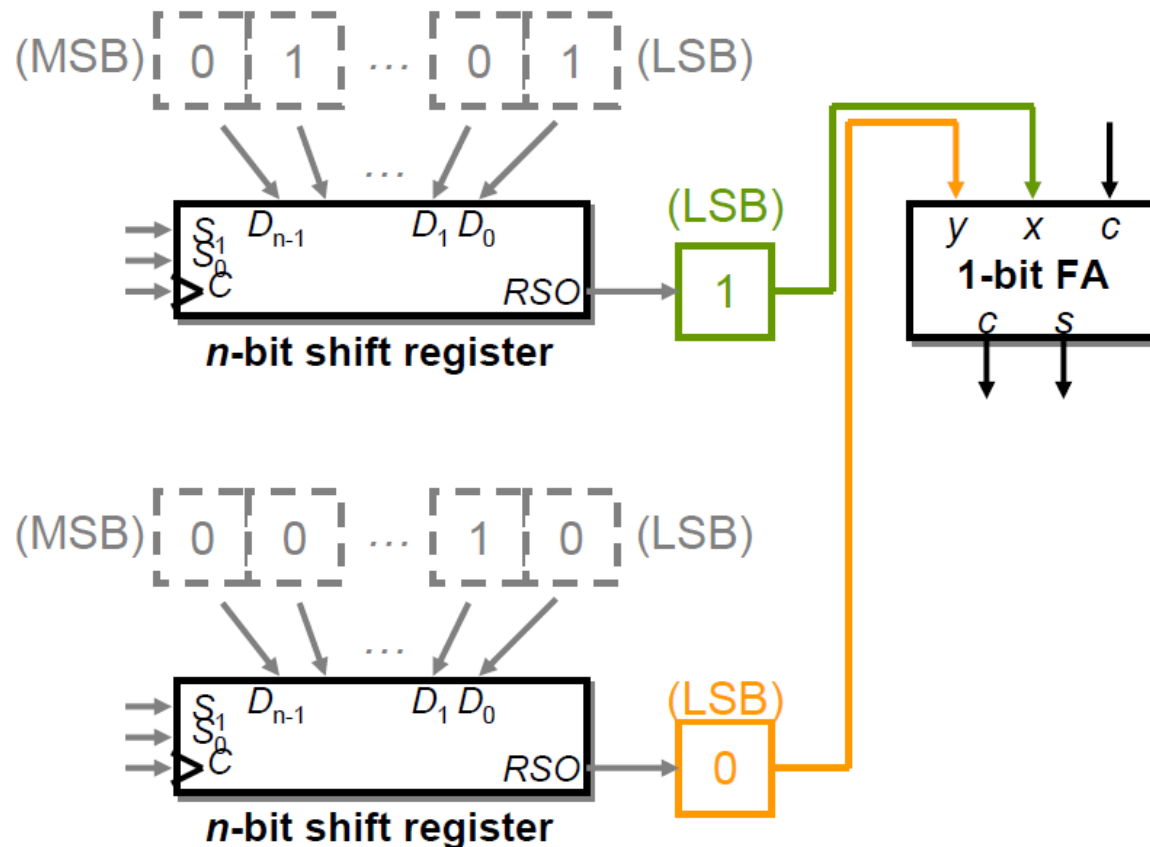
Serial Binary Addder

- 2) Squeeze the binary number X and Y bit-by-bit (LSB first)



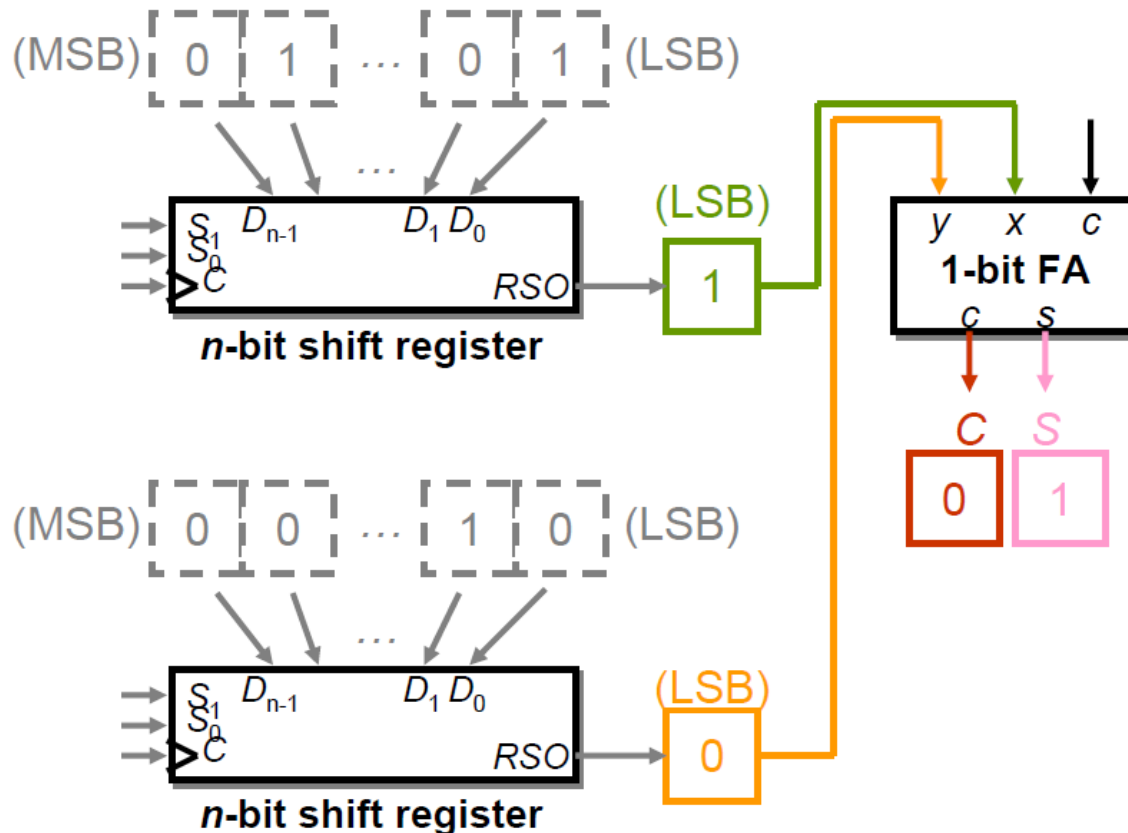
Serial Binary Addder

- 3) Forward the two binary bits to full adder



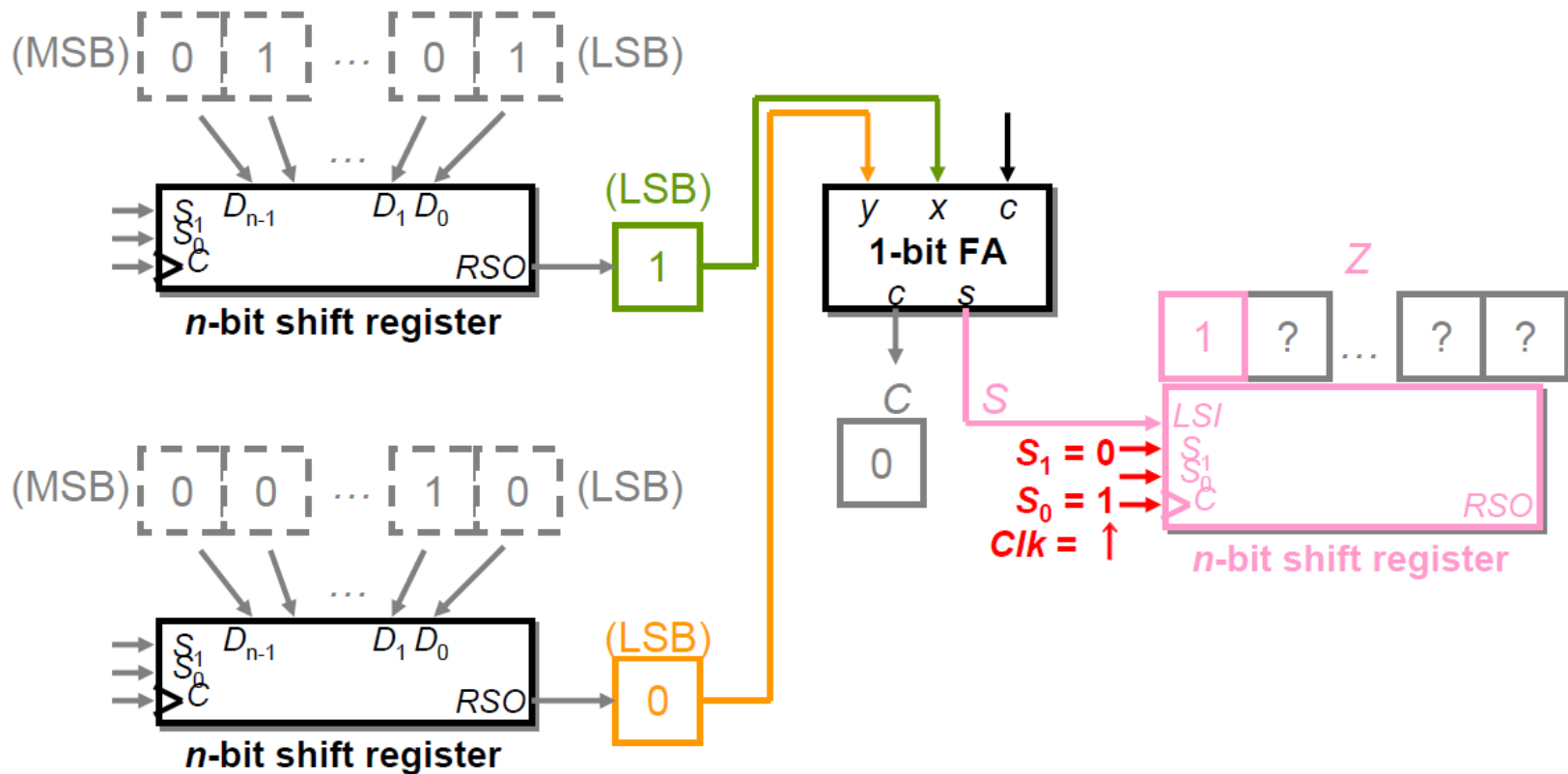
Serial Binary Adder

■ 4) Perform 1-bit addition



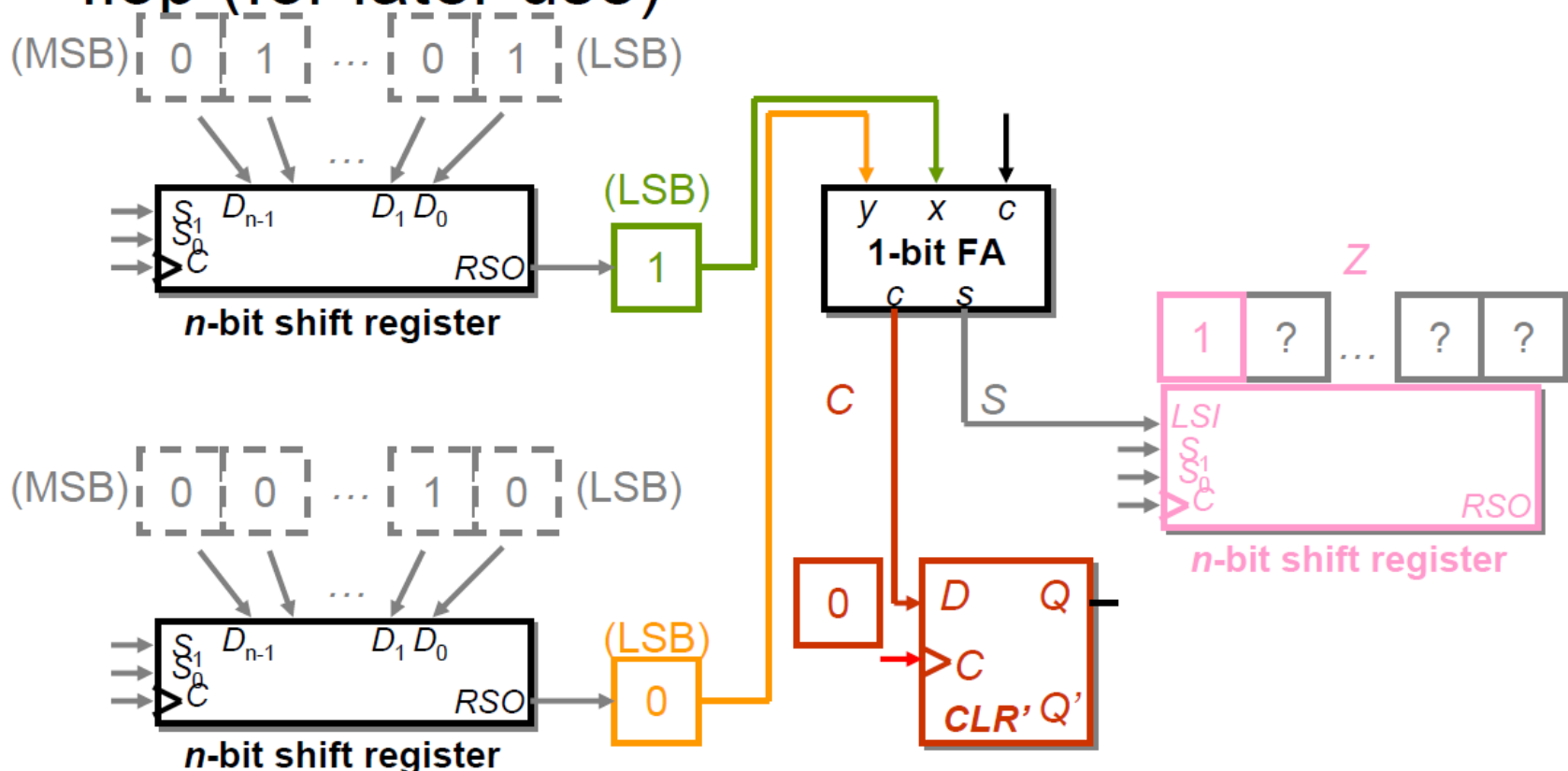
Serial Binary Adder

- 5) Feed the 1-bit output sum (S) to Z 's LSI



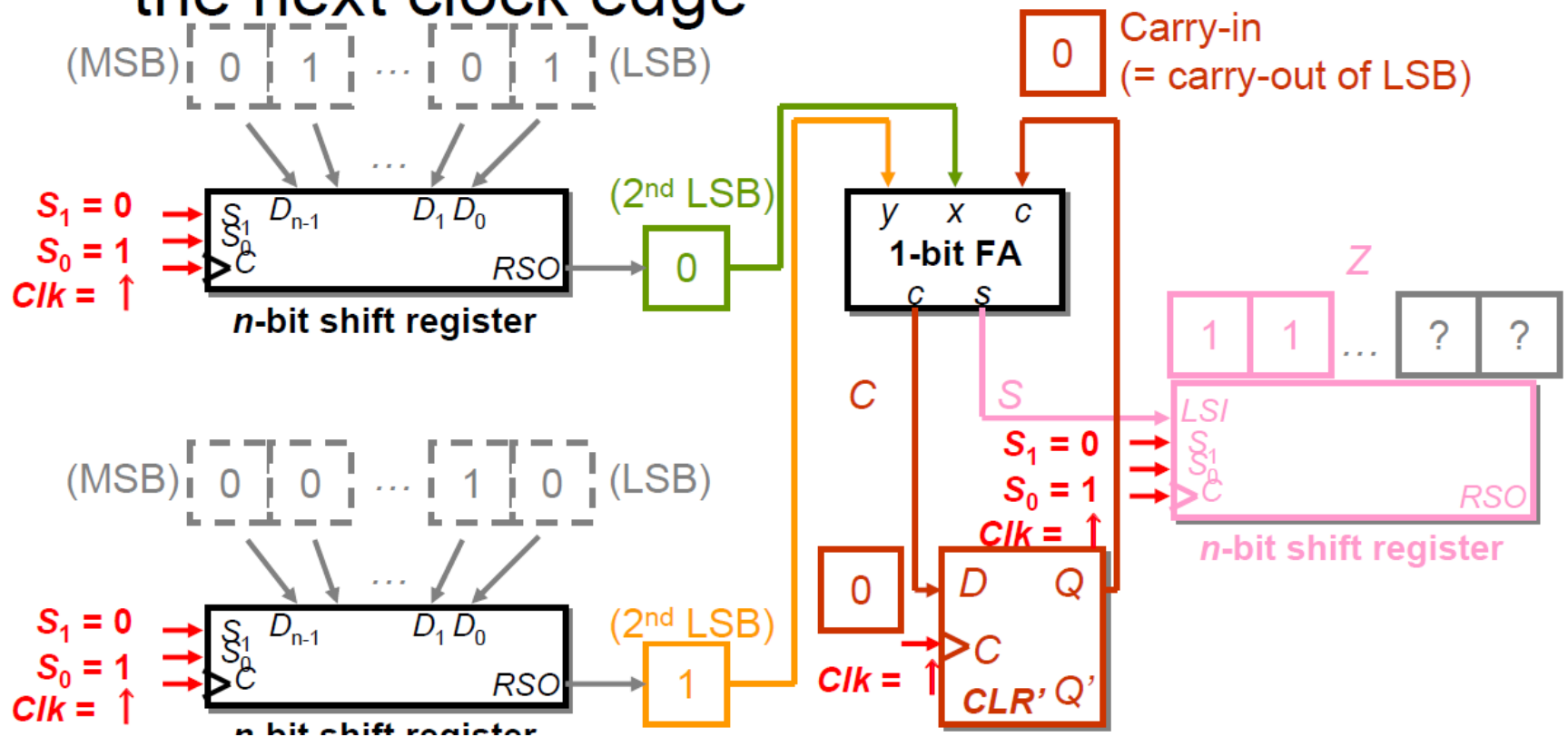
Serial Binary Adder

- 6) Store the 1-bit carry output (C) in a flip-flop (for later use)



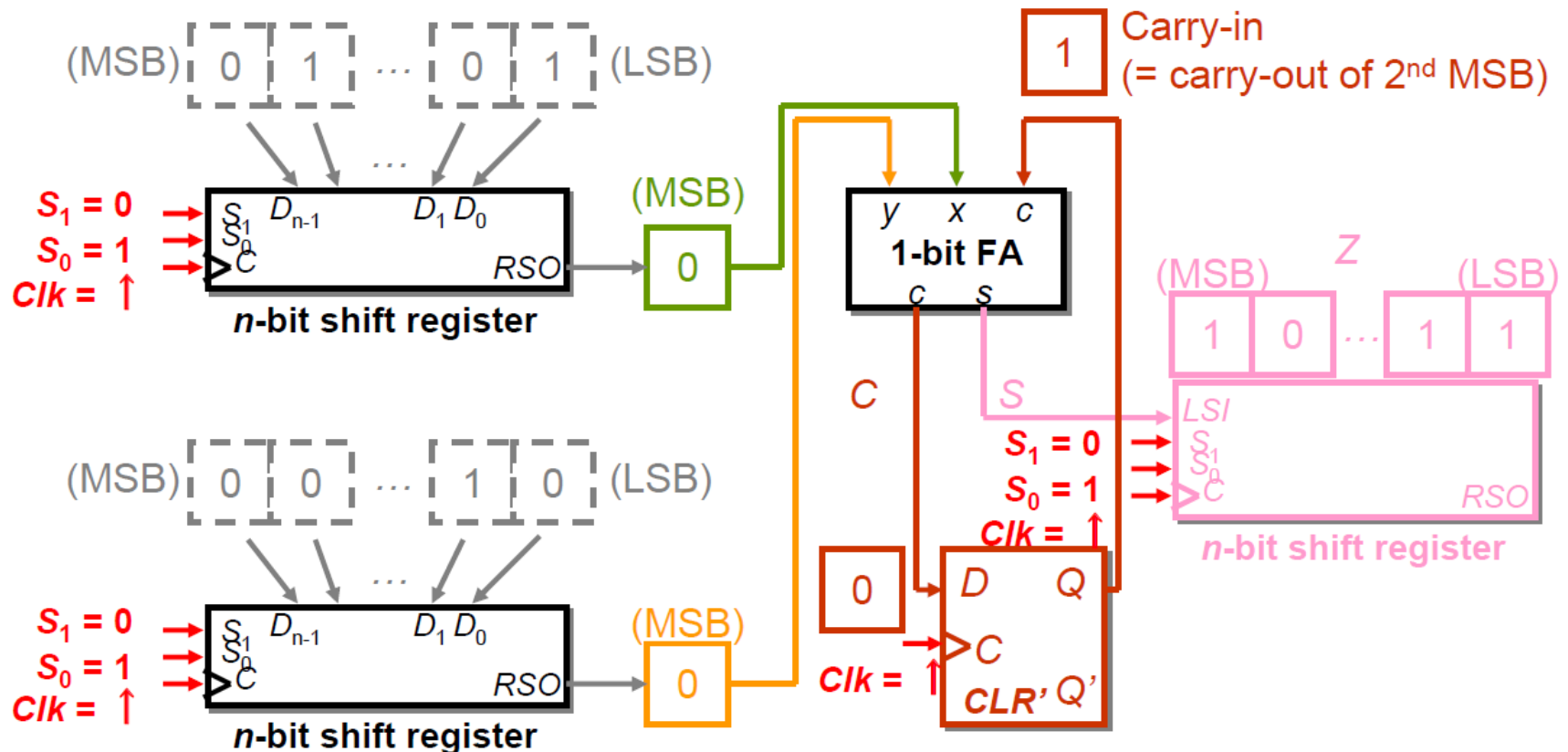
Serial Binary Adder

- 7) Calculate the 2nd least significant bit at the next clock edge

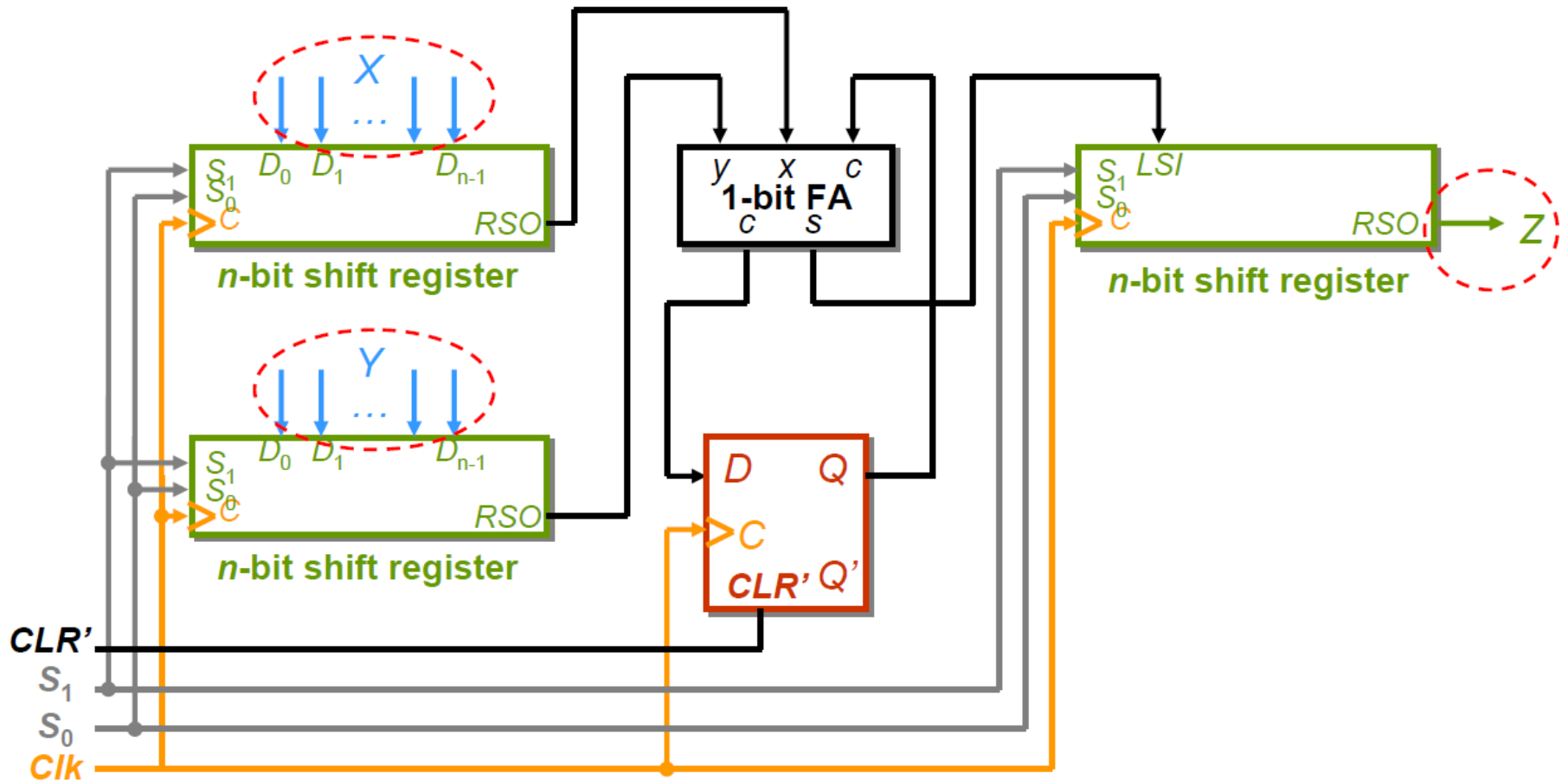


Serial Binary Adder

- Finally) Calculate the most significant bit



Serial Adder (Full Circuit)



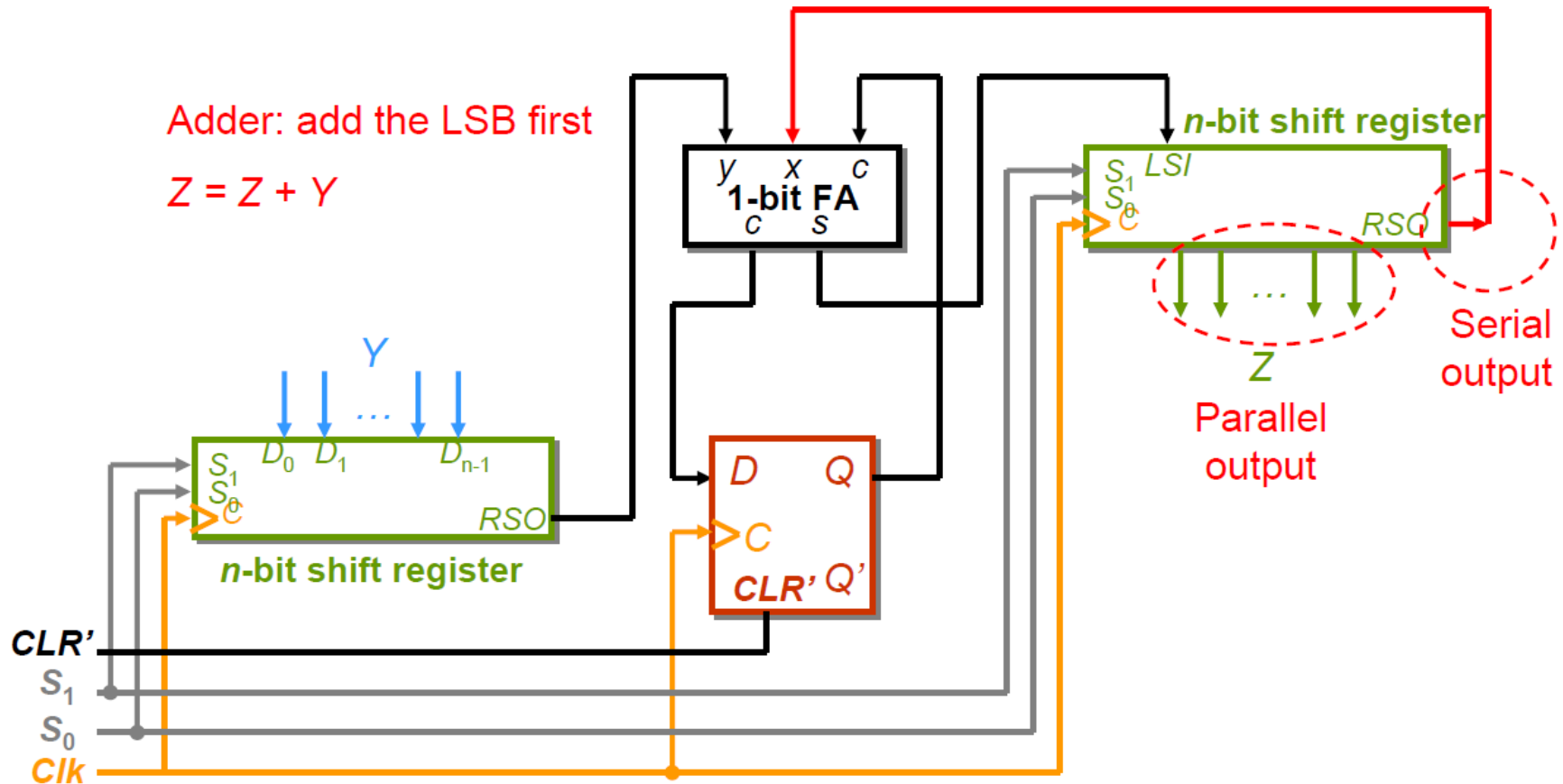
Input: X, Y (n -bit parallel load)

Adder: add the LSB first

Output: Z (n -bit serial out, LSB first)

$Z = X + Y$

Serial Accumulator



Summary of 9.1

- Registers used to store a group of binary bits (known as **word**)
- A n -bit register can store a n -bit word
- Functions of shift registers
 - Hold
 - Parallel load
 - Serial input / output, left / right shift

Counters

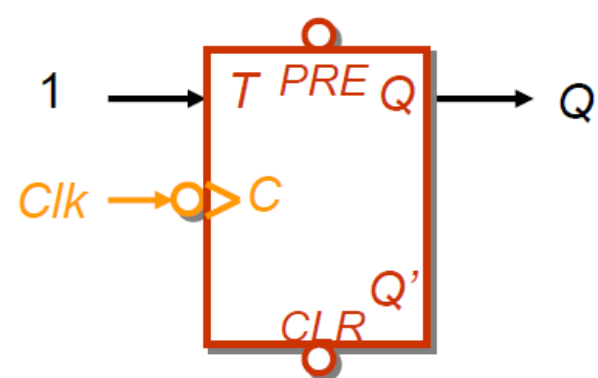
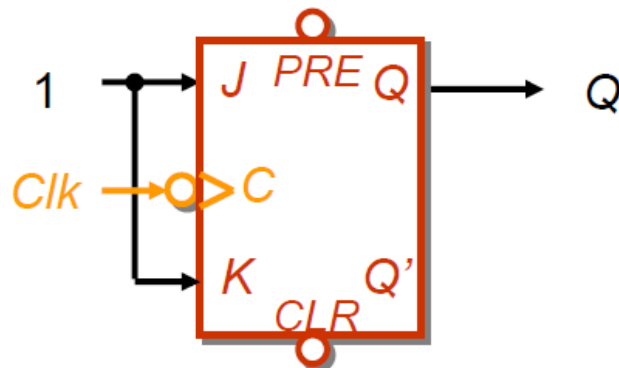
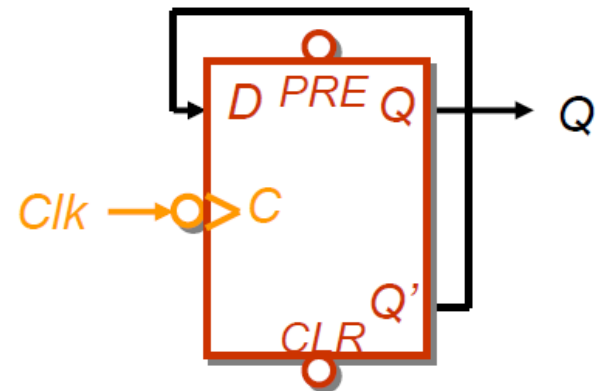
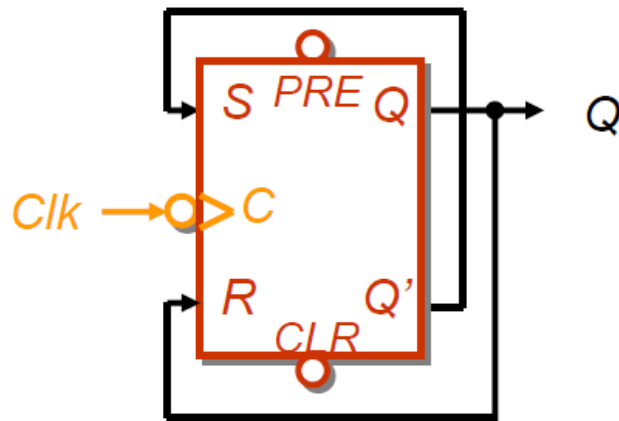
- Register are useful for storing and manipulating information
- What is a counter?
 - A special type of register
 - Goes through a predetermined sequence of states
 - Used to sequence and control operations in a circuit

Counters

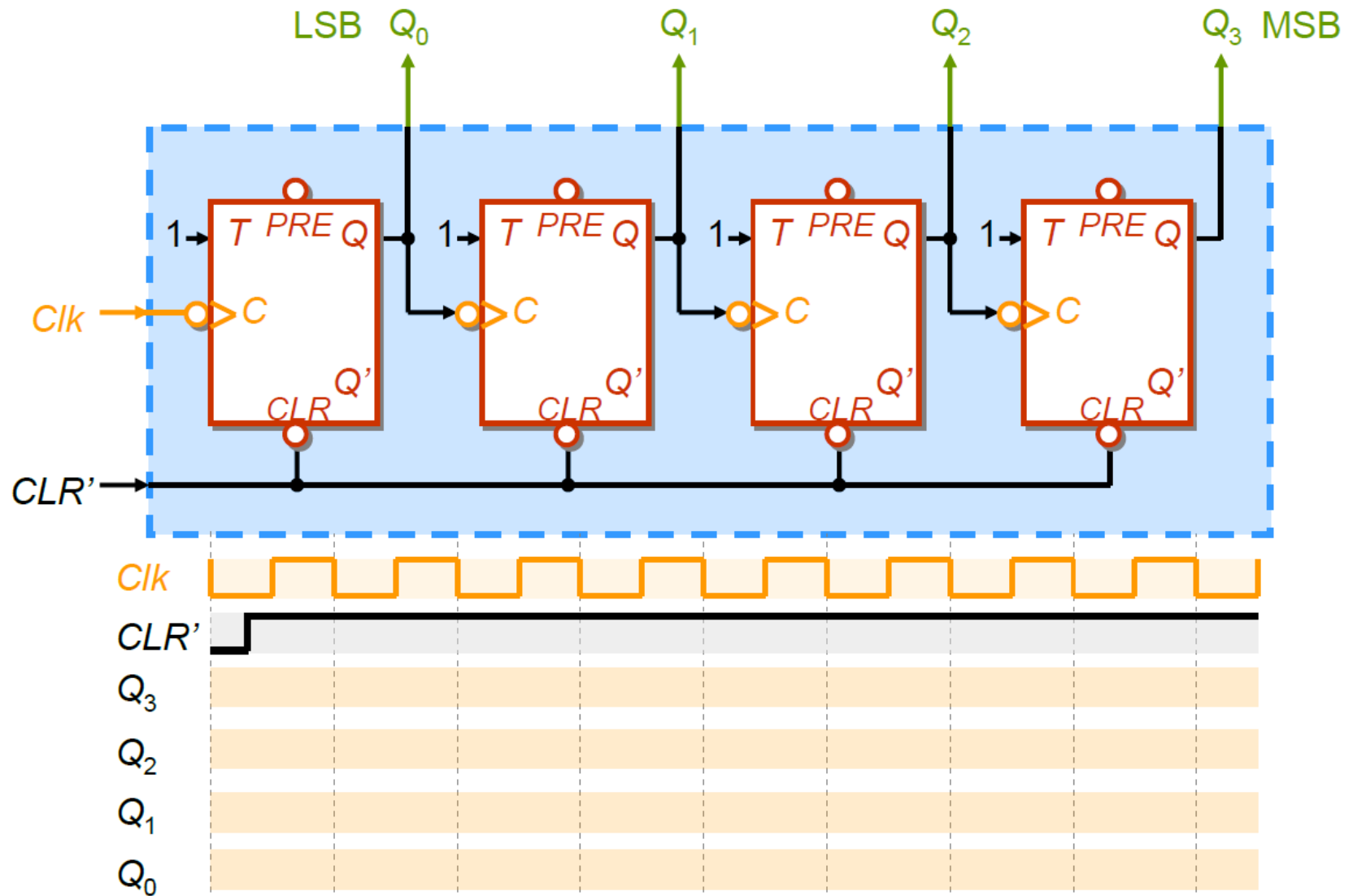
- Two classes of counters
 - Asynchronous (binary) counters
 - Synchronous (binary) counters
- Asynchronous counters
 - e.g. ripple counters: the flip-flops are triggered by the **transition of other flip-flops**
- Synchronous Counters
 - All flip-flops are triggered by the **common clock**

Asynchronous Counters

- Review the following flip-flop circuits:



4-bit Ripple Counter (Async.)



4-bit Ripple Counter (Async.)

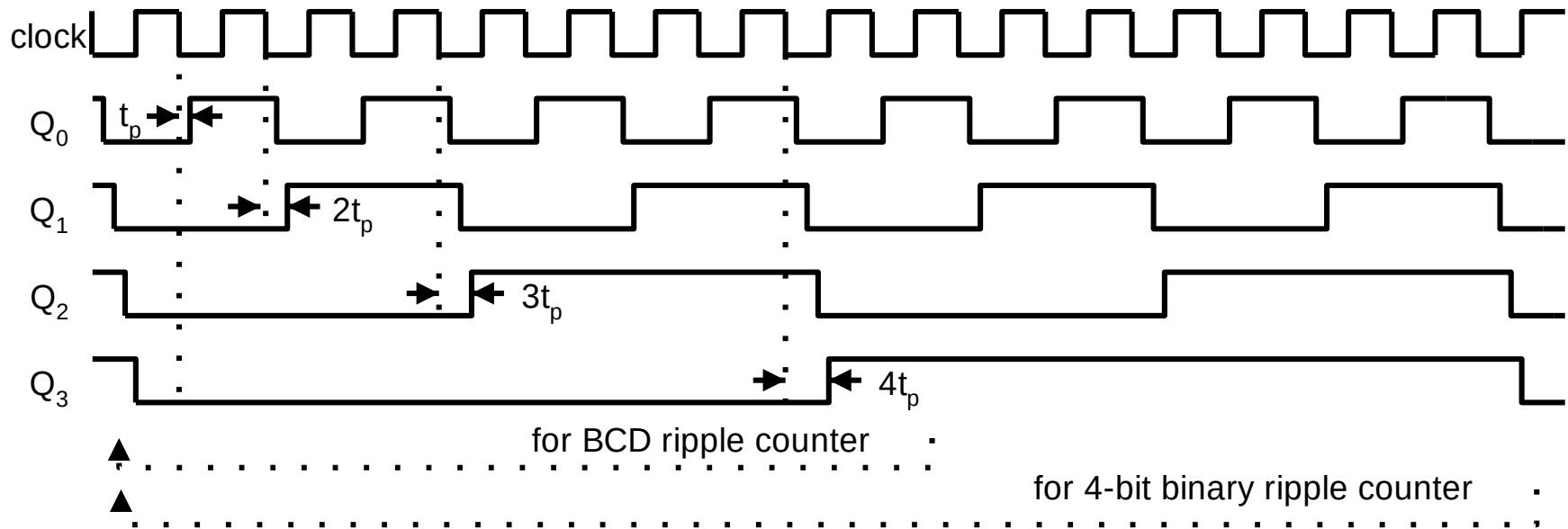
Upward Counting Sequence			
Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

repeat

$CLR' = 0$

Triggered at the falling edge (i.e. the moment that previous FF's Q falling from 1 to 0)

4-bit Ripple Counter (Async.)

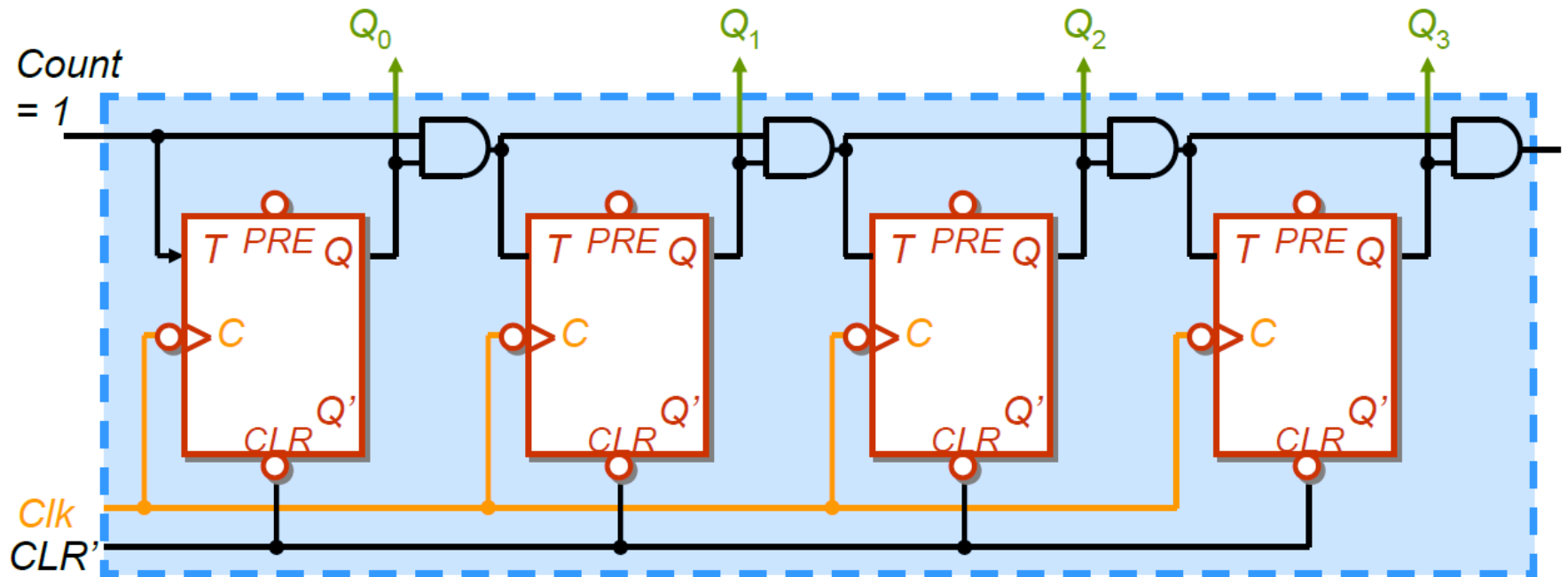


- The change of states does not occur at the same time
- Propagation delay of the flip-flops will ripple through the “clock input” of FFs
- The Q_i of the last flip-flop has the largest delay

Asynchronous binary counter

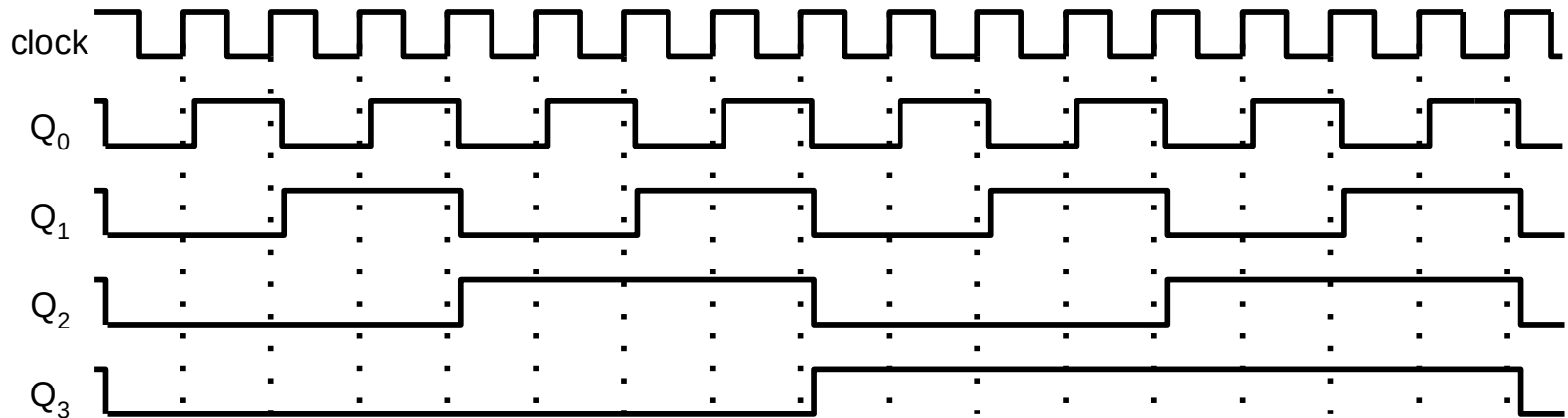
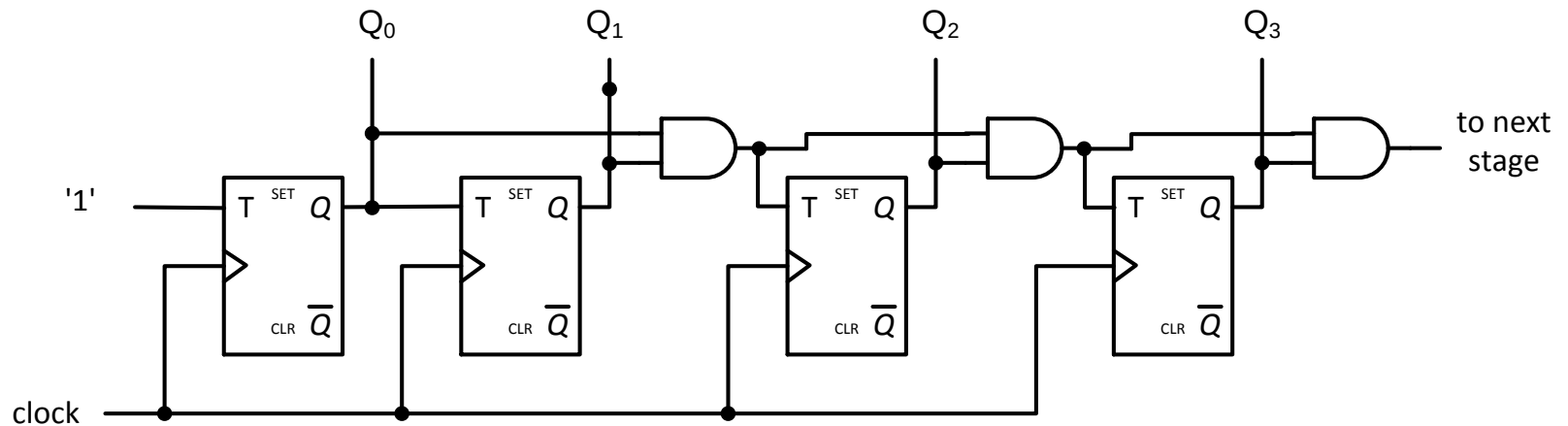
- An asynchronous counter is also called ripple counter.
- In a ripple counter, the FF output transition serves as a source for triggering other FFs.
- In other words, the clock inputs of all FFs (except the first) are triggered not by the incoming pulse, but rather by the transition that occurs in other FFs.

Synchronous Counter



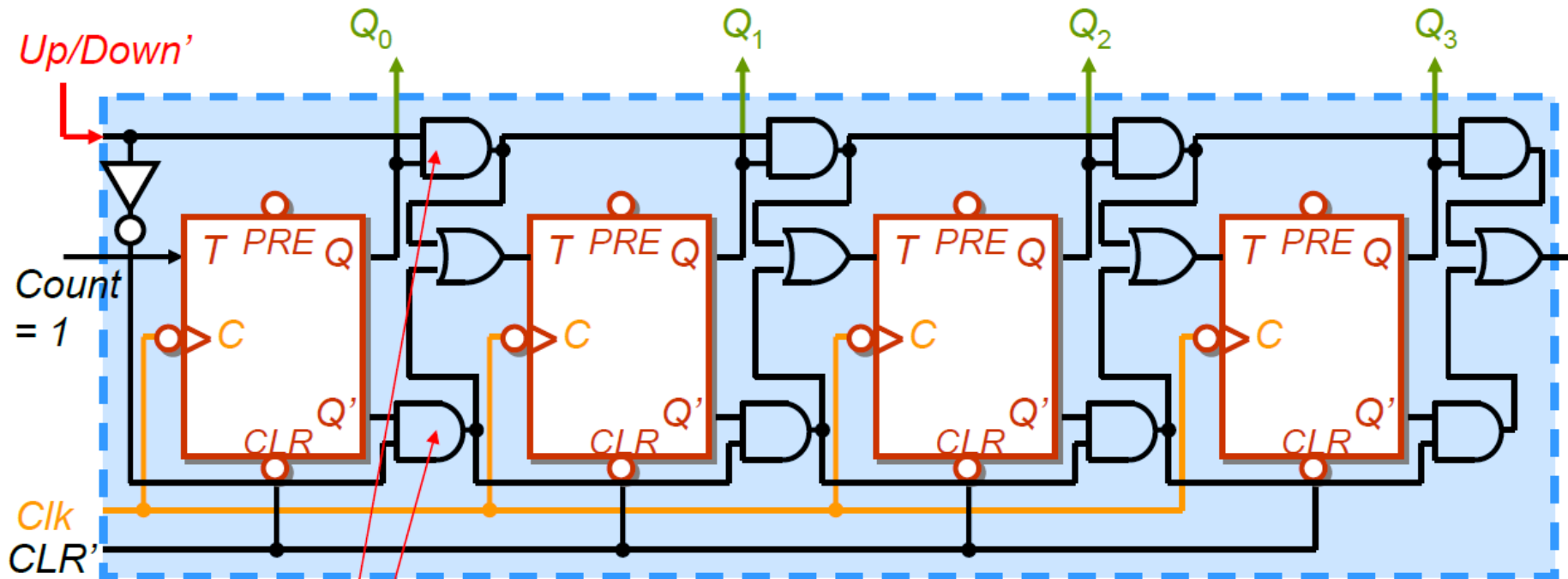
- A synchronous binary counter is a simple counter for which all the FFs are driven by a common clock, thus all the **change of states will occur at the same time** without further delay.
- It can also be used to **provide the frequency division function**.

Synchronous Counter



The delays of all FFs are the propagation delays of their own.

Sync. Up-Down Counter



Enable circuit

Up/Down' used to select feeding either Q or Q' to next T

If $Up/Down' = 1$, upward counting (the same as previous counter). If $Up/Down' = 0$, downward counting (feeding Q' to T)

Sync. Up-Down Counter

Upward Counting Sequence			
Q_3	Q_2	Q_1	Q_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
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Downward Counting Sequence			
Q_3	Q_2	Q_1	Q_0
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0

Synchronous Vs. Asynchronous

■ Synchronous Counters

- The change of states of the flip-flops will occur at the same time, as they are driven by the common clock

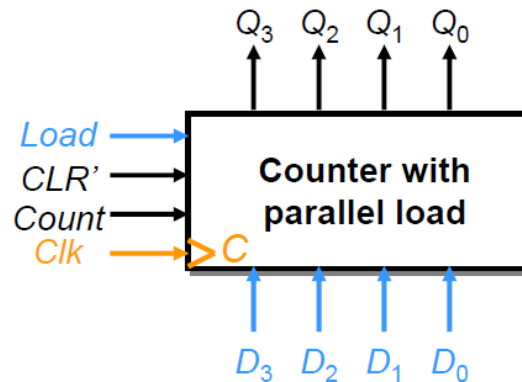
■ Asynchronous Counters

- The change of states are not occur at the same time
- Propagation delay of the flip-flops
- The last flip-flop has the largest delay

Modulo counter

- A modulo- N (abbreviated mod- N) counter is a counter that goes through a repeated sequence of N counts.

■ Counters + parallel load

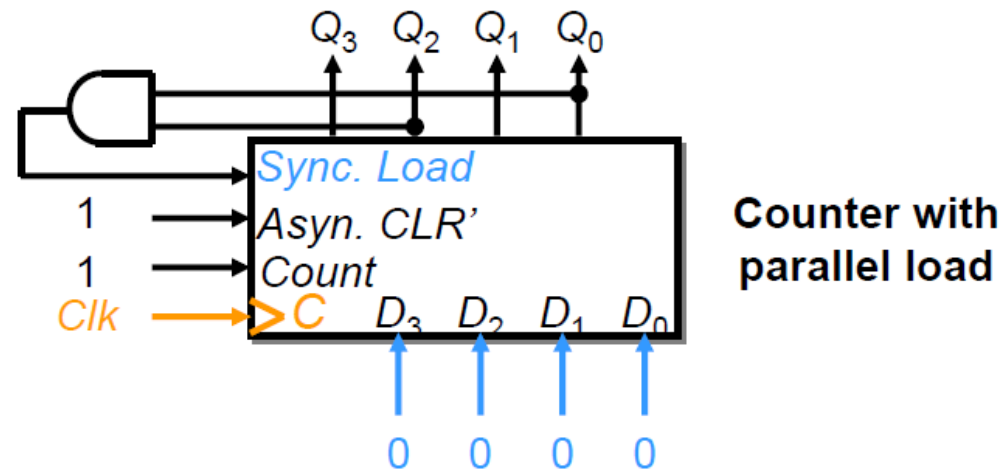


Parallel load the data into the counter when $Load = 1$ and Clk is in active transition

- Used to initialize the counter to a specific value

Modulo Counter Example

- Counting from 0, 1, 2, 3, 4, 5 and repeats (why?)

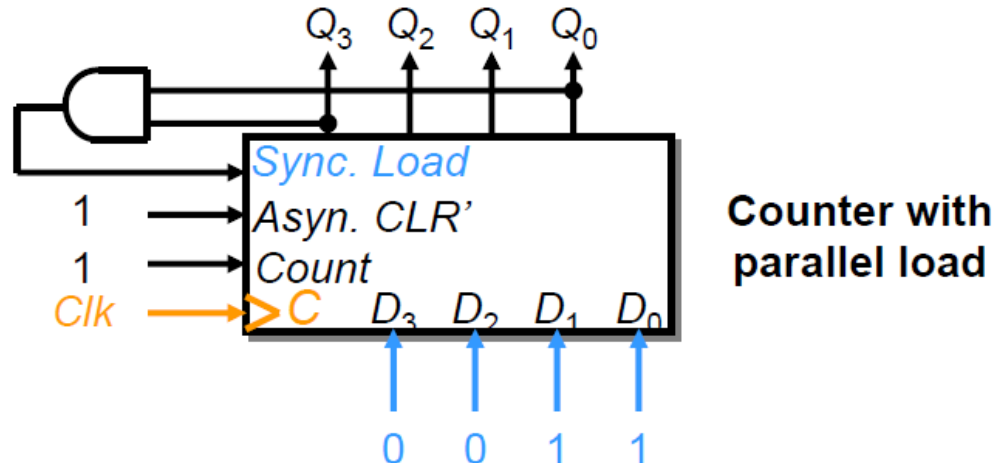


Then it counts 0000 → 0001 → 0010 → 0011 → 0100 → 0101 (= x1x1)

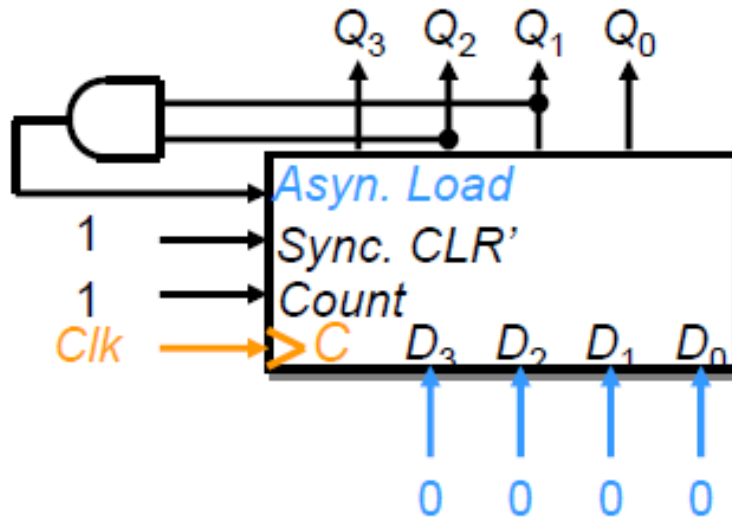
It will trigger the parallel load when the $Q_i = x1x1$ (e.g. 0101, 0111, 1101, 1111)

Modulo Counter Examples

- Can perform specific sequence generator
- e.g. counting from 3, 4, 5, 6, 7, 8, 9 and repeats



Modulo Counter Example

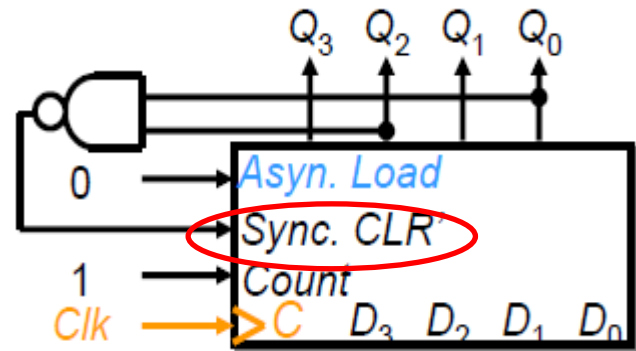
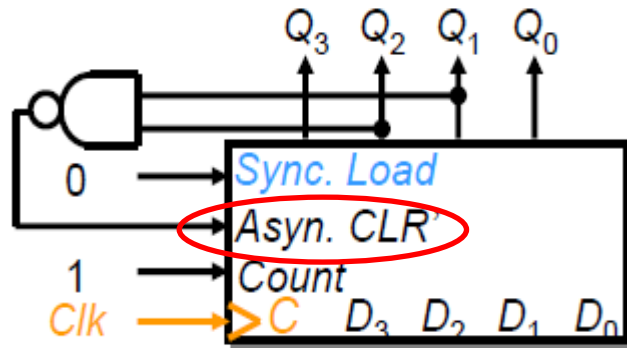


With *Async Load* control, its counts

0000 $\rightarrow$ 0001 $\rightarrow$ 0010 $\rightarrow$ 0011 $\rightarrow$ 0100 $\rightarrow$ 0101 $\rightarrow$ (0110/0000)

When 0110 appears, $D_3D_2D_1D_0$ (0000) will immediately be loaded to $Q_3Q_2Q_1Q_0$. **0110 will only appear for a negligible period.** The counter output becomes 0000 and the counter repeat its normal operation.

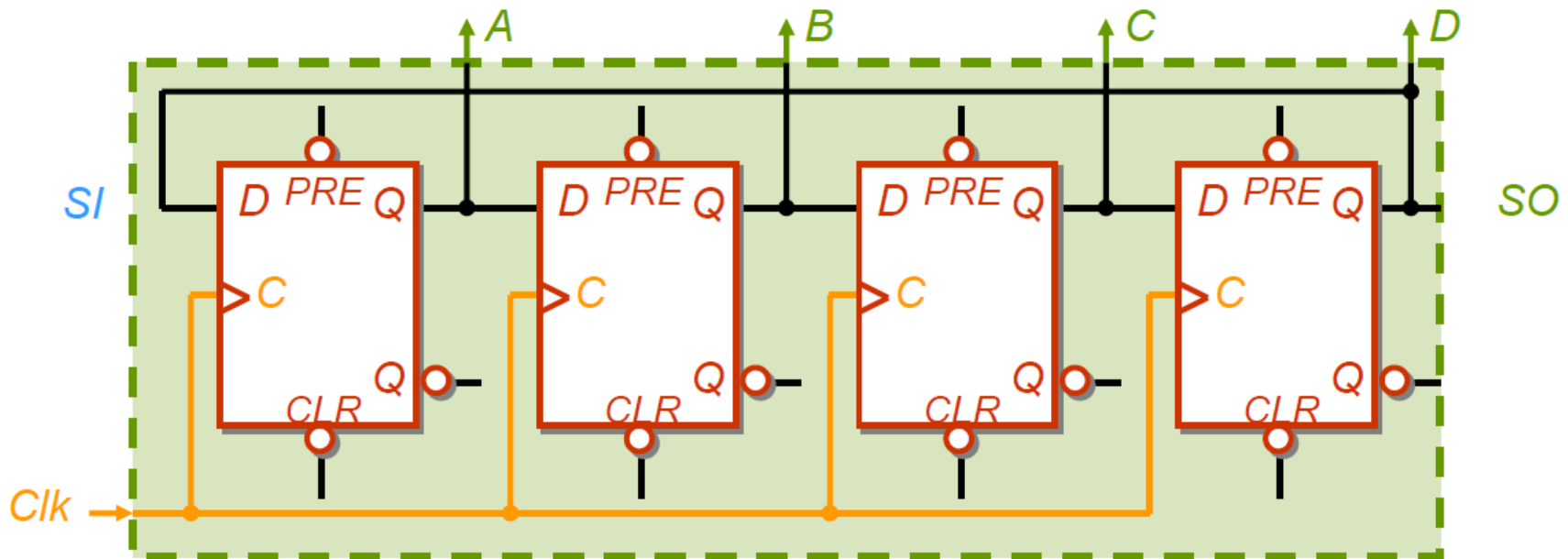
Modulo Counter Example



With **Sync & Sync CLR** control, the clear conditions are different. This design is simpler since the **connections to inputs are not required**.

Ring Counters

- Special class of synchronous counters
- Basic Ring Counters
 - Circular shift register with only 1 flip-flops being **set** (i.e. = 1) at all time



Basic Ring Counters

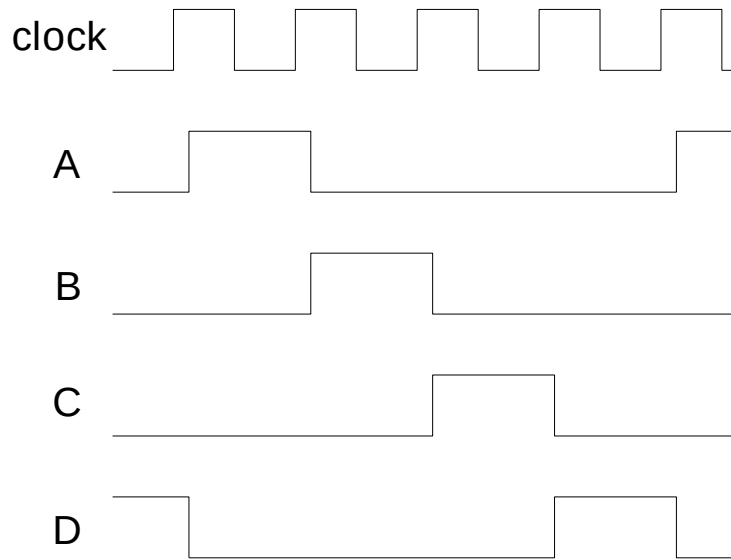
- Initially set one flip-flop and clear all other flip-flops

- The “1” will be rotated through the registers

Time	A	B	C	D	Decoded Output
1 st clock period	1	0	0	0	A
2 nd clock period	0	1	0	0	B
3 rd clock period	0	0	1	0	C
4 th clock period	0	0	0	1	D
5 th clock period	1	0	0	0	A
...	...	...	...	...	...

- Particularly useful in time division multiplexing (TDM) applications
 - Enable only one Q_i at a time

Basic Ring Counters



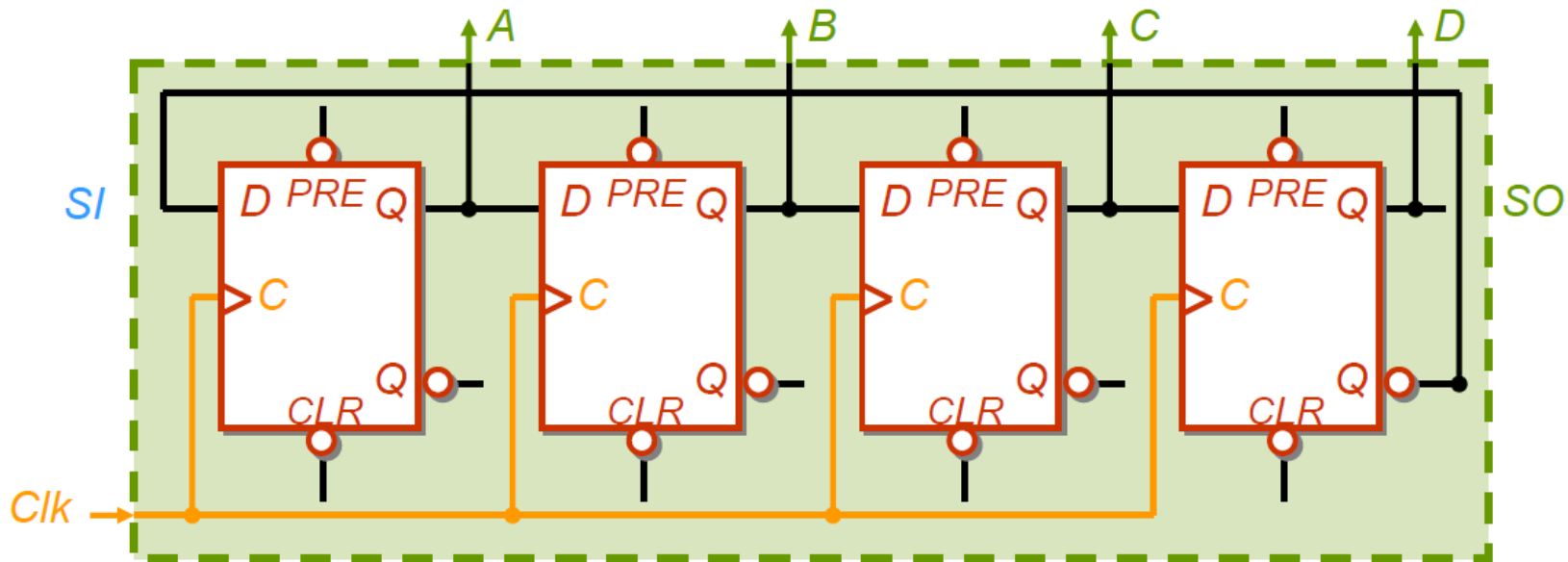
A	B	C	D	Decoded Output
1	0	0	0	A
0	1	0	0	B
0	0	1	0	C
0	0	0	1	D

- N-bit counter with N-state
- Particularly useful in time division multiplexing (TDM) applications
 - Enable only one Q_i (only one user) at a time

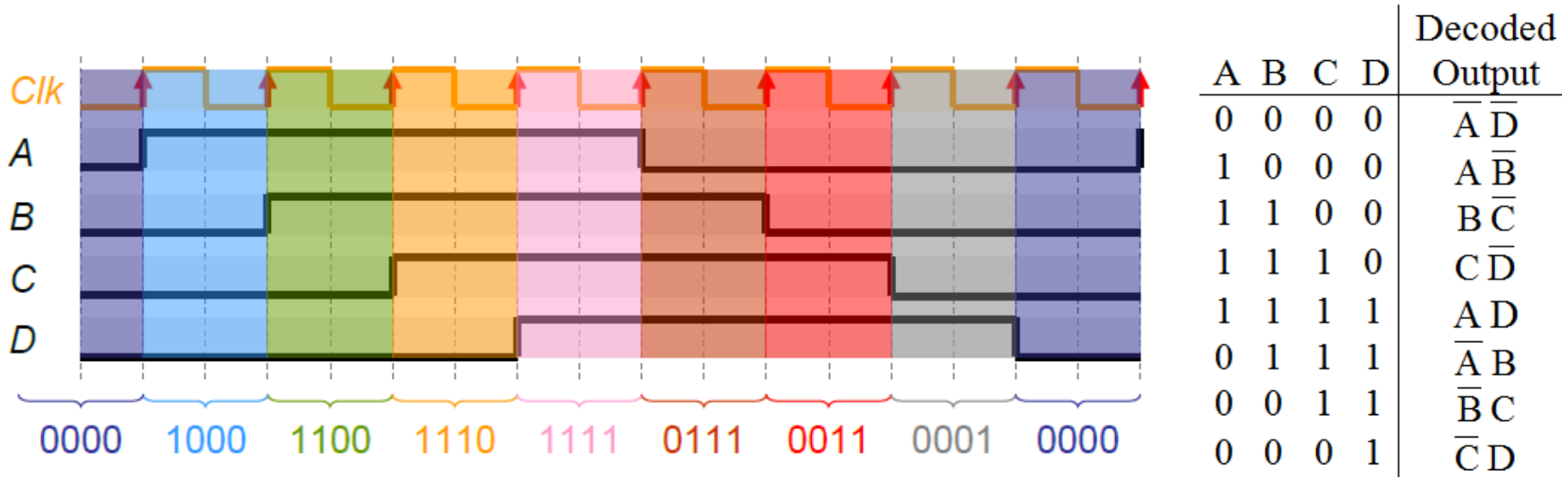
Twist Ring Counters

(Johnson counter)

- The twisted ring counter is a circular shift register with the complement output of the last stage connected to the input of its first stage.



Twist Ring Counters



- The number of states for a n -bit twisted ring counter is $2n$.
- Only one output bit changes between consecutive count states.
- The count sequences can be decoded with just 2 bits from each count state, no matter how many stages are in the counter.

Summary of 9.2

- Synchronous Counters
 - Outputs (Q_i) appear at the same time
- Asynchronous Counters
 - Outputs (Q_i) do not appear at the same time
 - The final flip-flop has the largest propagation delay
- Counters, modulo counters, ring counters, twisted ring counters
 - Simple design to construct frequency divisor, irregular frequency divisor, time division multiplexer, period shifter respectively