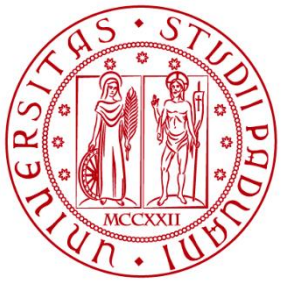
OF THE
DEPARTMENT OF
INFORMATION ENGINEERING



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Digital Systems

Shift Registers and Counters

Marta Bagatin, marta.bagatin@unipd.it

Degree Course in Information Engineering

Academic Year 2023-2024

Purpose of the Lesson

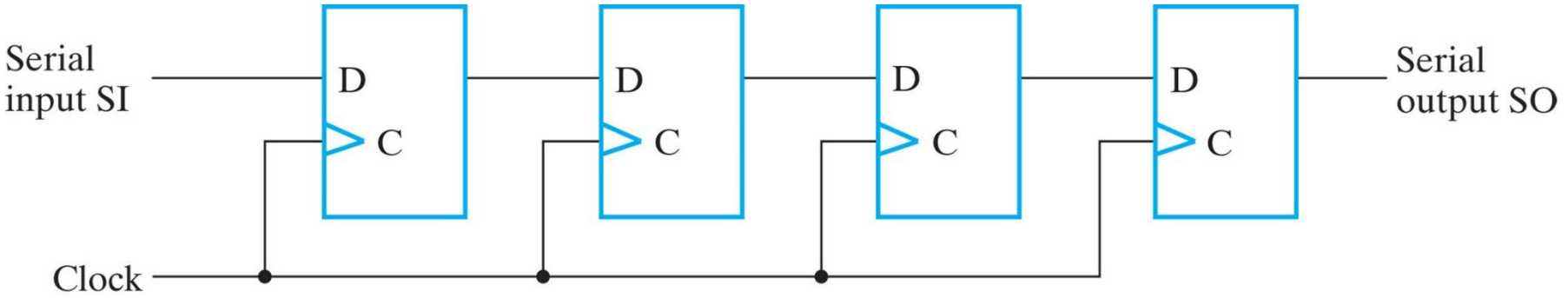
- Study the operation of different types of registers
  - Shift registers
  - Counters
    - Ripple counter
    - Synchronous counter

Shift Registers

Shift Register

- The shift register is a register that **shifts its bits laterally**, to one or to both directions
- It consists of a **chain of flip-flops**, with the output of each flip-flop connected to the input of the next one. All the flip-flops are controlled by the **same clock signal, which activates the shift**
- The simplest shift register consists only of flip-flops (see next slide)

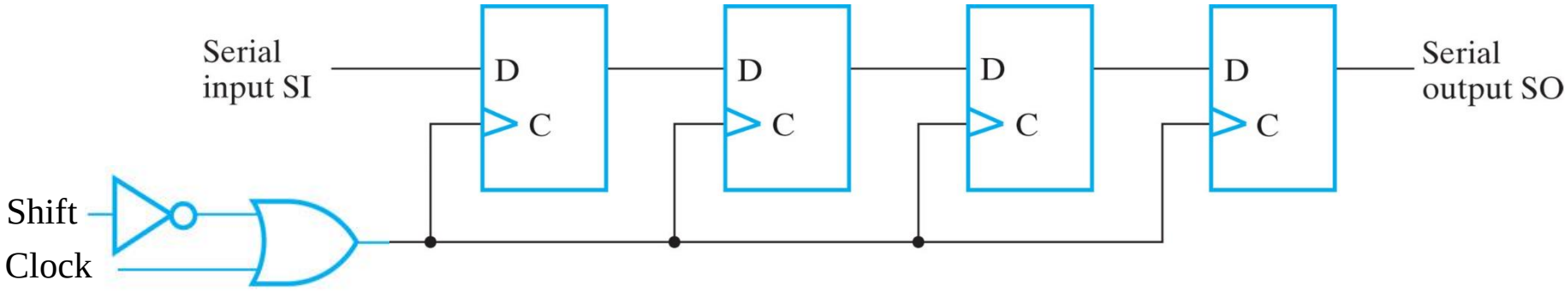
4-bit Shift Register



- **Serial input**: input of the leftmost flip-flop
- **Serial output**: output of the rightmost flip-flop
- The **clock** is in common and directly connected to all FFs: the shift occurs at each rising edge of the clock (Positive-Edge-Triggered D-FF)

Symbol

4-bit Shift Register with Shift Enable



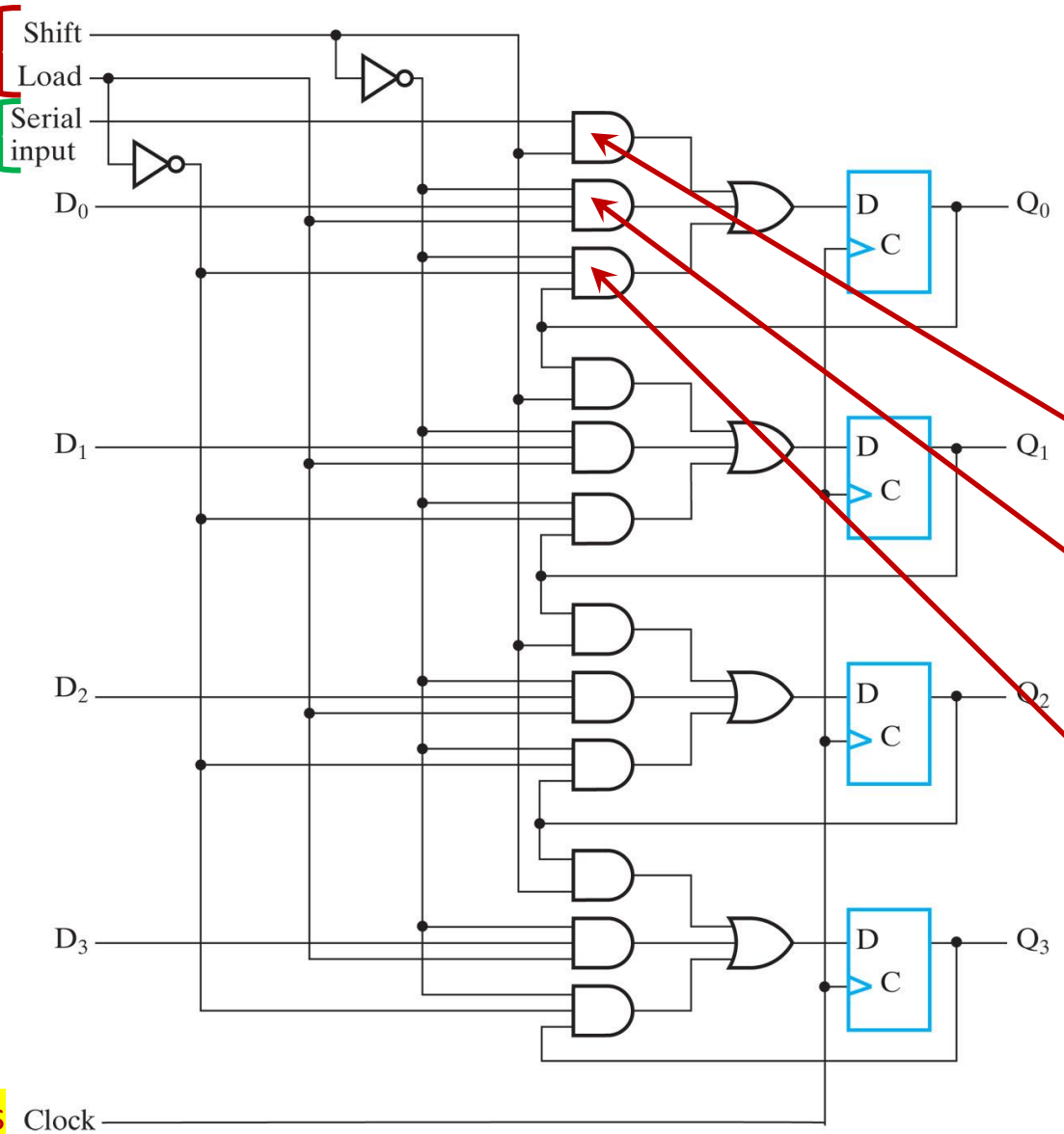
- The clock is connected to the FF through an **enabling circuit with a shift signal** (similar to Load signal, previously seen, to enable load of data into a register)
- In this way, we can enable the shift **only at selected time**
- This solution can lead to **clock skew issues**, so it is preferable to adopt a control on the input of the flip-flop rather than on the clock, as we will see in the following

Shift Register with Shift Enable and Parallel Load

- If we have **access to the input of all flip-flops** in the shift register (not only to the input of the first FF in the chain), we can **enter data in the register in parallel (parallel load)**
- Similarly, having access to the output of all FFs in the chain, we can take out in parallel the data entered serially in the register

$\Rightarrow$ A shift register with **access to the inputs and outputs of all flip-flops** can be used to **convert data entered in parallel to outgoing serial data and vice versa**

Shift Register with Shift Enable and Parallel Load



2 control signals (shift, load)

1 input signal

Each stage consists of
- 3 AND gates
- 1 OR gate
- 1 D-FF

Shift Enable

Enable load of data $D_0\,D_1\,D_2\,D_3$

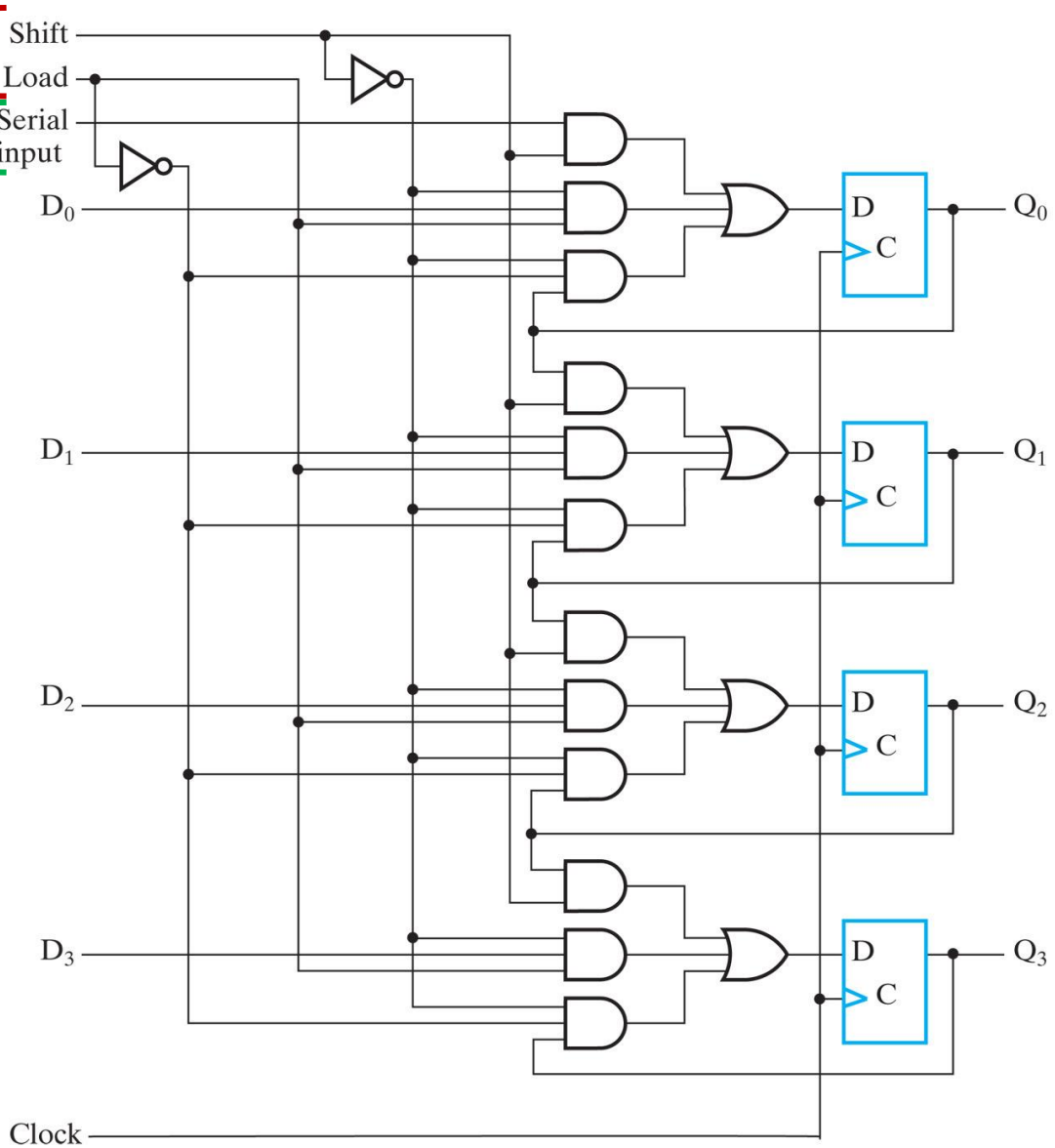
Keep current value (load FF output value into its input) if no change is required

Clock directly connected to FFs

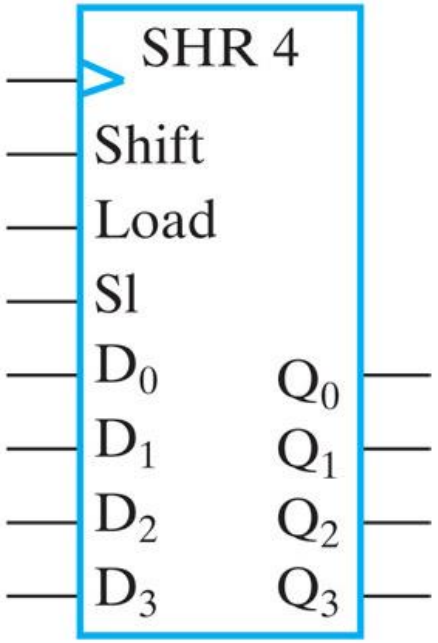
Shift Register with Shift Enable and Parallel Load

2 control signals (shift, load)

1 input signal



Clock directly connected to FFs



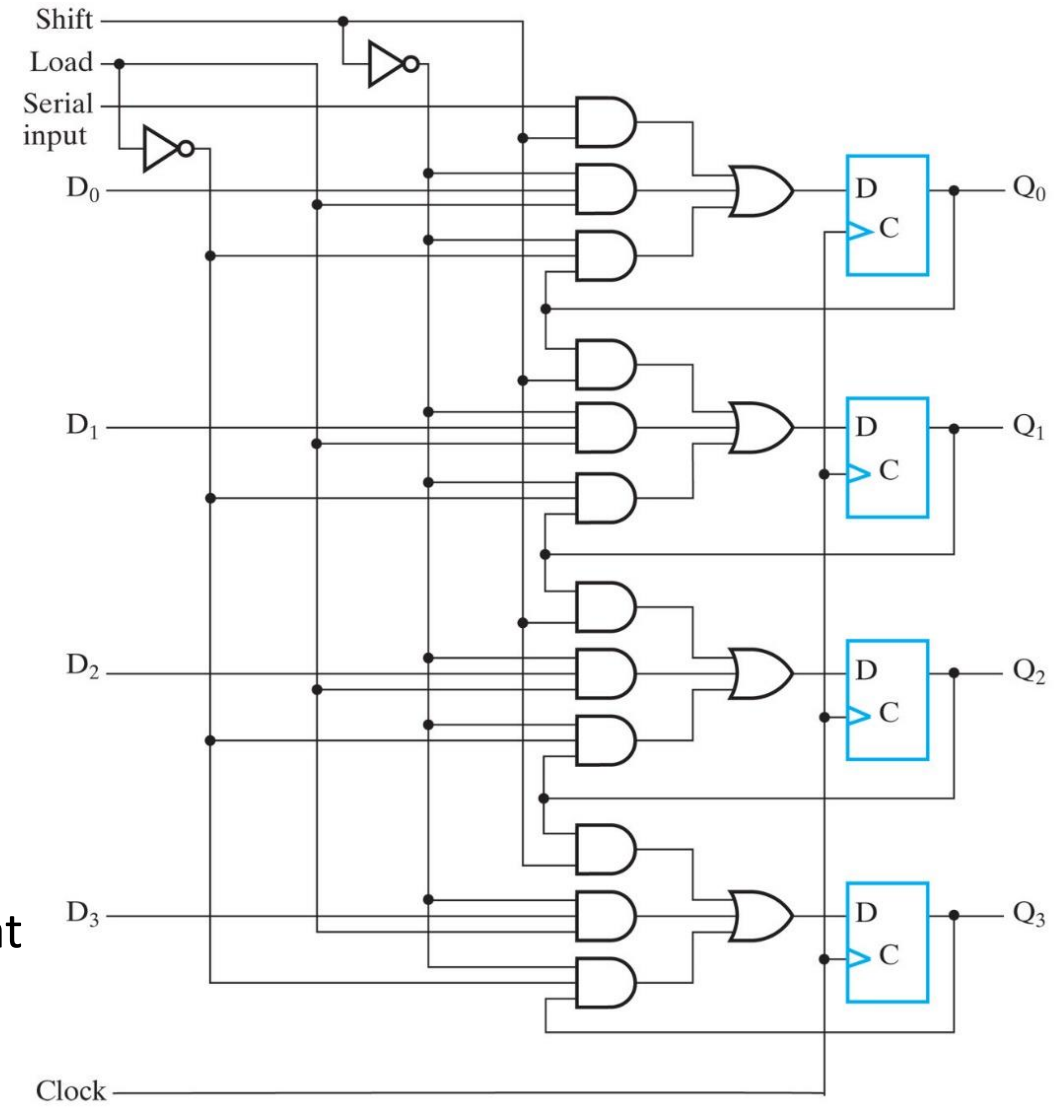
Shift Register with Shift Enable and Parallel Load

Shift	Load	Operation
0	0	No change (Hold)
0	1	Load parallel data
1	×	Shift left (down) from Q_0 to Q_3

$$\mathbf{Shift : Q \leftarrow sl\ Q}$$

$$\overline{\mathbf{Shift}} \cdot \mathbf{Load : Q \leftarrow D}$$

- if Shift = 1 (regardless of Load): at the next clk cycle, SI is transferred to Q_0, Q_0 to Q_1, Q_1 to Q_2, Q_2 to Q_3, i.e. the **bits are shifted to the left** (from the LSB Q_0 towards the MSB Q_3)
- if Shift = 0 and Load = 1: parallel load of data, the input D_i of each FF is transferred to the output Q_i at the next clock cycle
- If Shift = 0 and Load = 0: "hold" or "no change" operation, i.e. at the next cycle the outputs Q_i keep the same value as before (the output of each FF is applied to its own input)

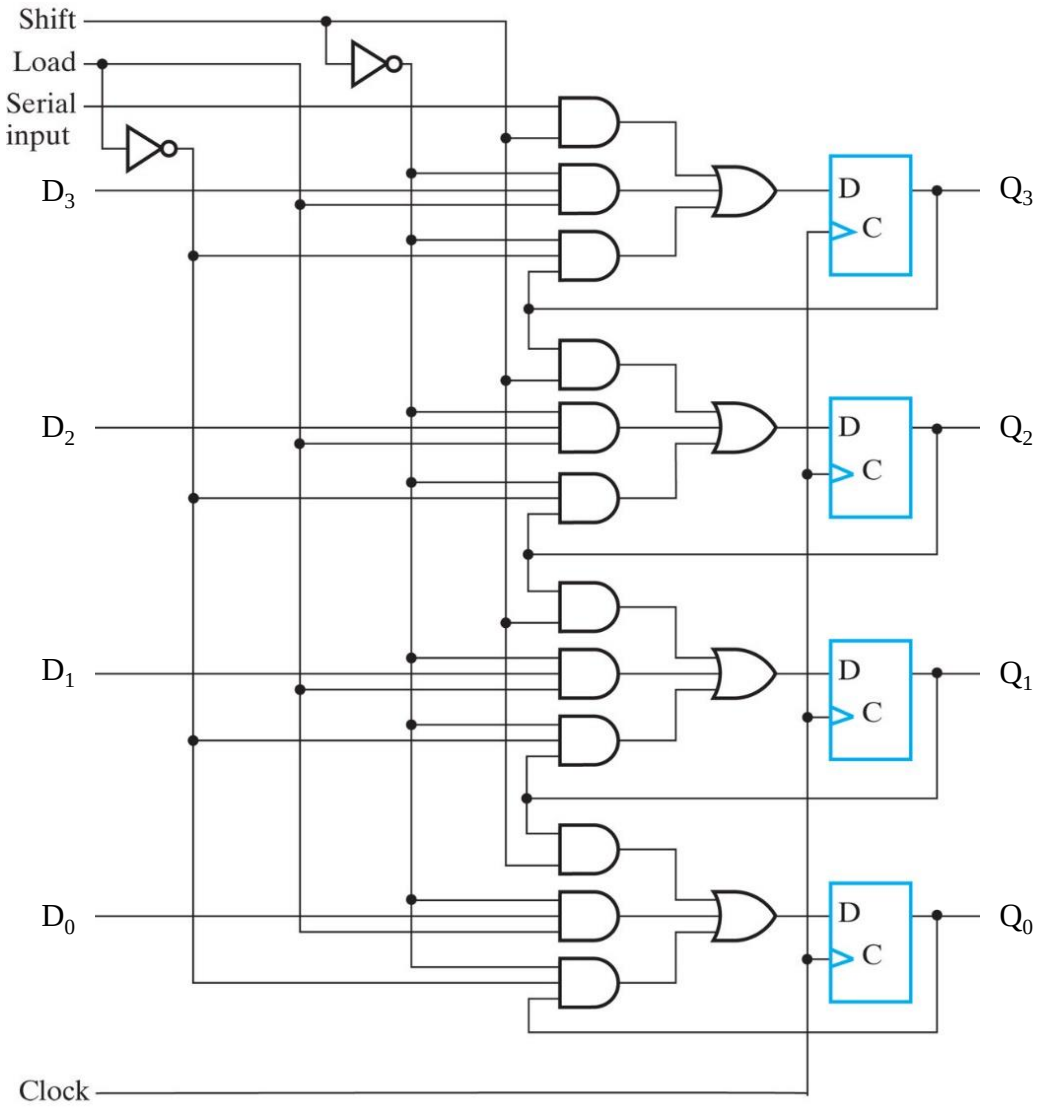


Shift Register with Shift Enable and Parallel Load

- To **shift data to the opposite direction** (to the right, i.e. from the MSB to the LSB) the circuit has the same structure, but, this time, D_i and Q_i are in the opposite order
- In RTL language:

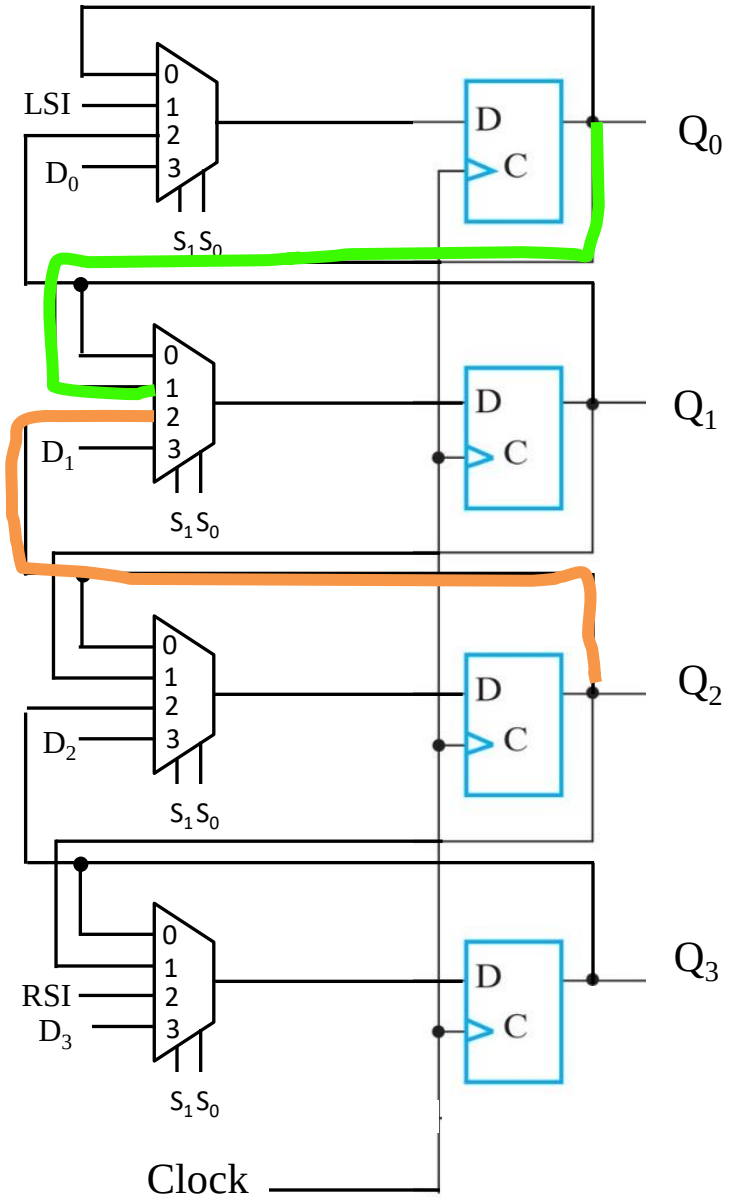
$$\mathbf{Shift} : \mathbf{Q} \leftarrow \mathbf{sr\ Q}$$

$$\overline{\mathbf{Shift}} \cdot \mathbf{Load} : \mathbf{Q} \leftarrow \mathbf{D}$$



Bidirectional Shift Register

- So far, we have seen unidirectional shift registers, i.e. SR able to shift to just one direction
- A **bidirectional shift register** can shift to both directions and can be obtained adding an AND gate (to enable shift to the other direction) to each stage
- This is equivalent to **add a 4-to-1 multiplexer to every stage**, with the output connected to the input of each flip-flop and selection inputs (S_0, S_1) controlling the type of operation of the register
  - **S_1=0, S_0=1: shift left**
  - **S_1=1, S_0=0: shift right**



Bidirectional Shift Register

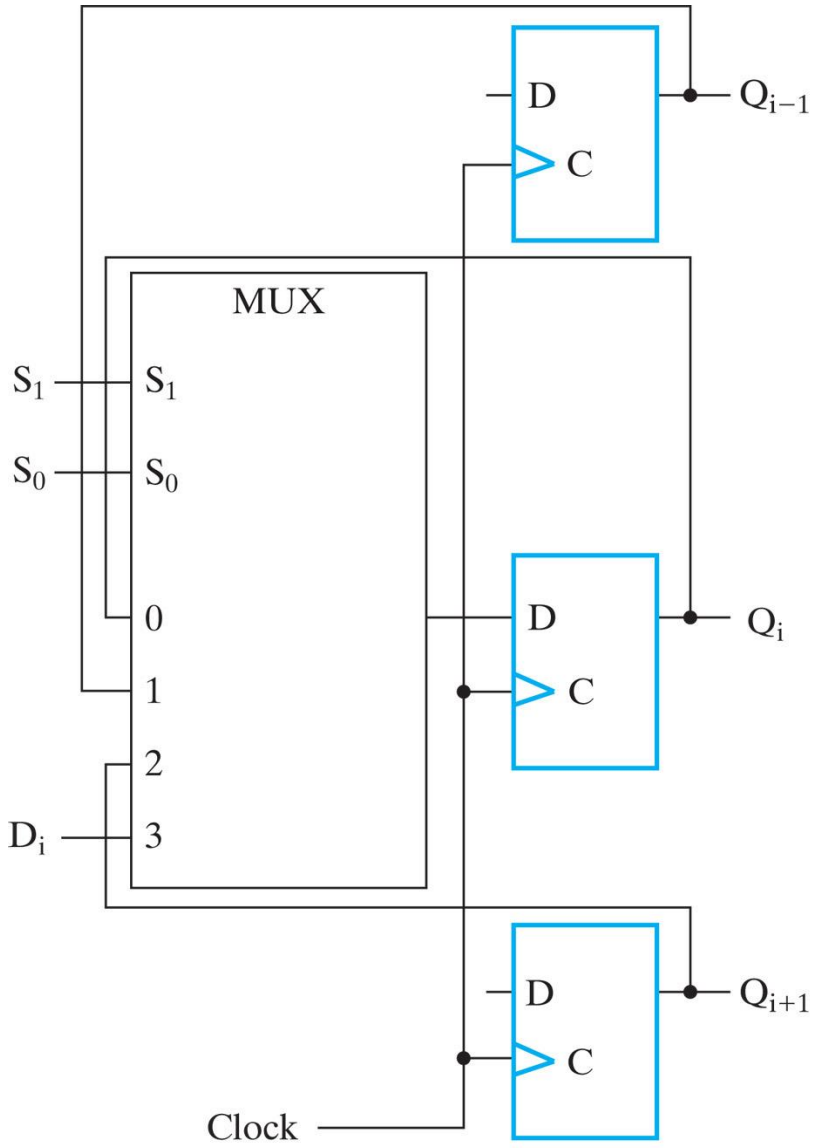
Logic diagram of each stage

- In RTL language:

$$S_1 \cdot \overline{S_0}: \; Q \leftarrow sr\ Q$$
$$\overline{S_1} \cdot S_0: \; Q \leftarrow sl\ Q$$
$$S_1 \cdot S_0: \; Q \leftarrow D$$

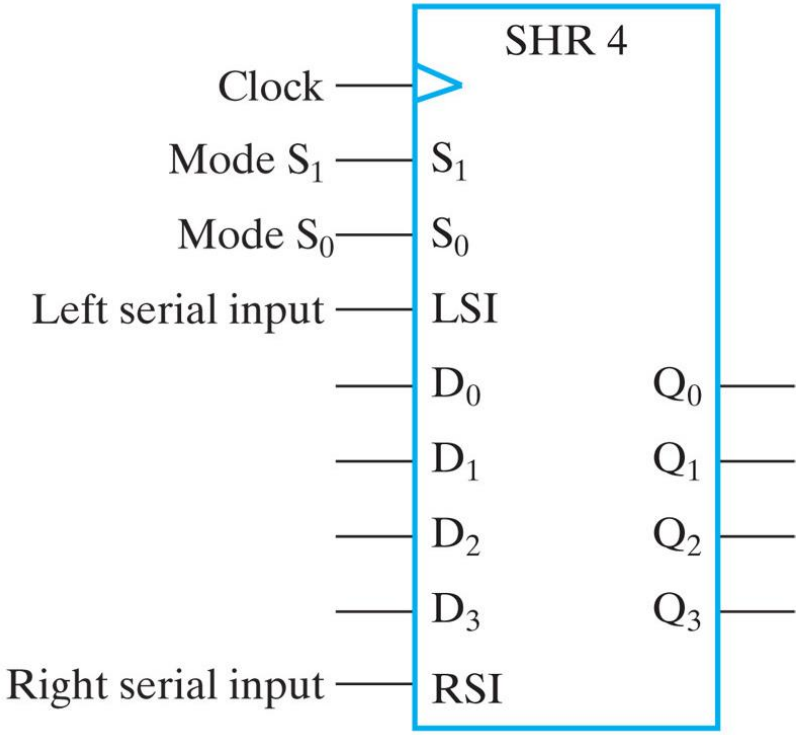
Note: the hold operation ($\overline{S_1} \cdot \overline{S_0}$) is implied when none of the above conditions is met

Mode Control		Register Operation
S_1	S_0	
0	0	No change (Hold)
0	1	Shift left
1	0	Shift right
1	1	Parallel load



Bidirectional Shift Register: Logic Symbol

The bidirectional shift register is also called **universal shift register**



LSI: Left Serial Input

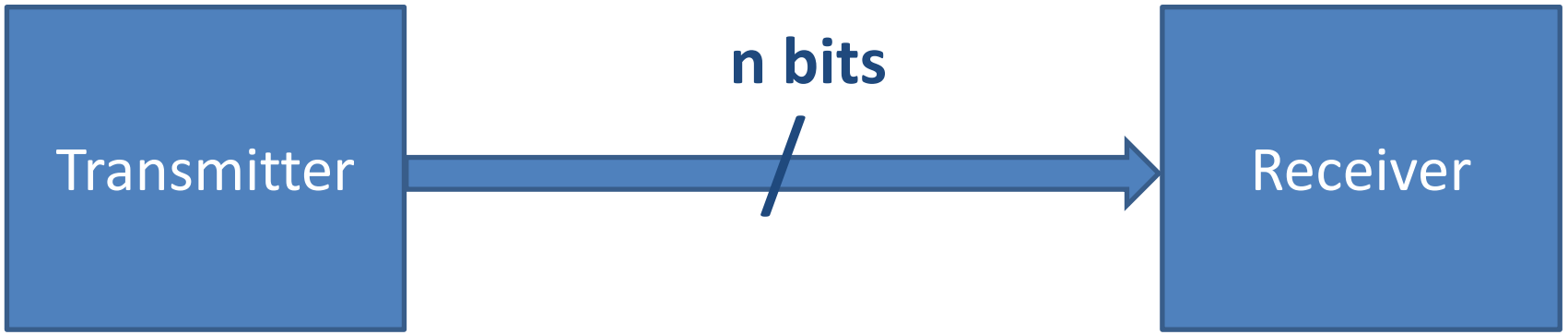
RSI: Right Serial Input

Q_0: serial output for Right Shift

Q_3: serial output for Left Shift

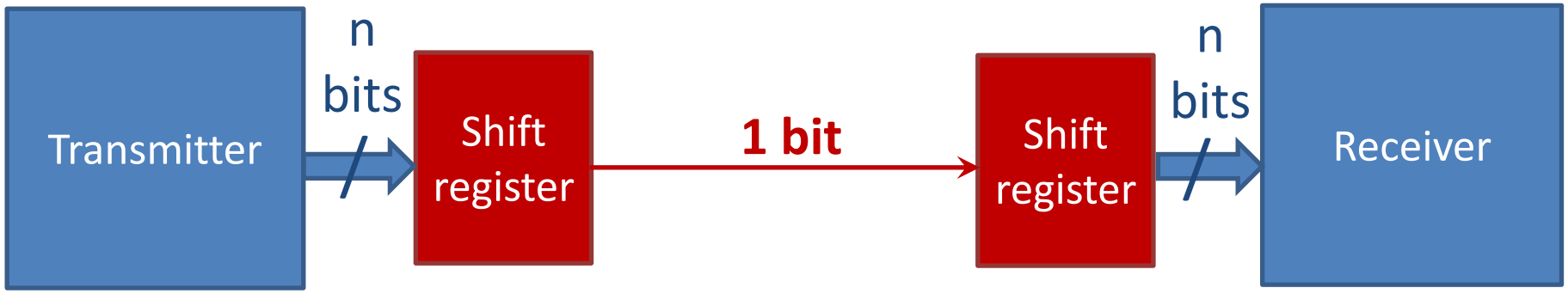
Where is the Shift Register used?

- One of the most common uses of shift registers is to realize **interfaces between different parts of a digital system**
- Suppose we need to transfer data (n bits) in parallel from one part of a circuit to another: using n data lines can be expensive, if the distance between the two parts is large



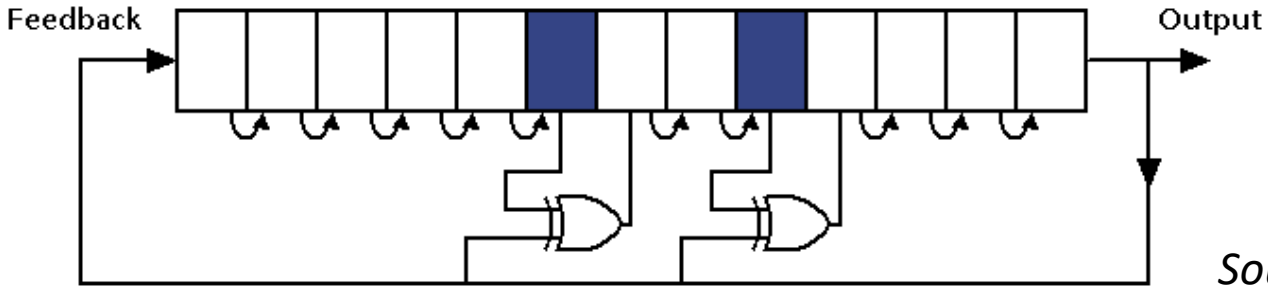
Where is the Shift Register used?

- With shift registers, we can use **a single line of data** (1 bit instead of n bits) to transmit data in a **serial way**
  - The transmitter loads the data in parallel into a shift register
  - The data are transmitted serially along the common line
  - Data are received serially from a shift register and they are available in parallel at the output of the shift register



Where is the Shift Register used?

- Shift registers are also used in other types of operations, such as:
  - **Multiplication**: sum of several partial products, properly shifted by a certain number of positions
  - **Pseudo-random number generator**: a sequence of pseudo-random numbers can be obtained if the input bit of the shift register is driven by the XOR of some bits of the overall shift register



Source: Wikipedia

  - **Delay line**: to transmit a signal to an element of the system with a certain delay compared to the moment in which it was generated (the delay can be varied by changing the clock frequency or by taking the signal on a different output of the register)

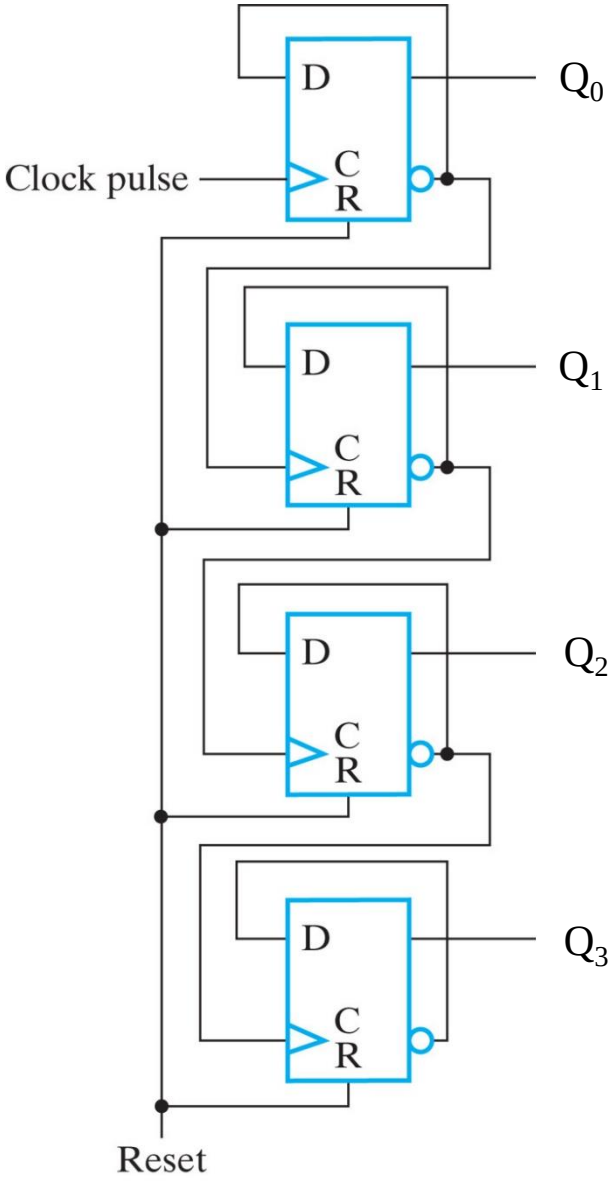
Counters

Definition and Categories of Counters

- The counter is a register which, upon the application of a series of input pulses, passes through a **predefined sequence of states** (binary numbers, random sequence, etc.)
- A counter that follows the binary number sequence is called a **binary counter**. An n-bit binary counter consists of n flip-flops and can count in binary **from 0 through 2^n - 1**
- There are two categories of counters
  - **Ripple counter**: the flip-flop output transitions provide the clock to other flip-flops in the system, i.e. flip-flop transitions serve as the sources to trigger the changes in other flip-flops. It is therefore an asynchronous counter
  - **Synchronous counter**: it is synchronized by a common clock, so each output transition (change of state) occurs at the edges of the clock

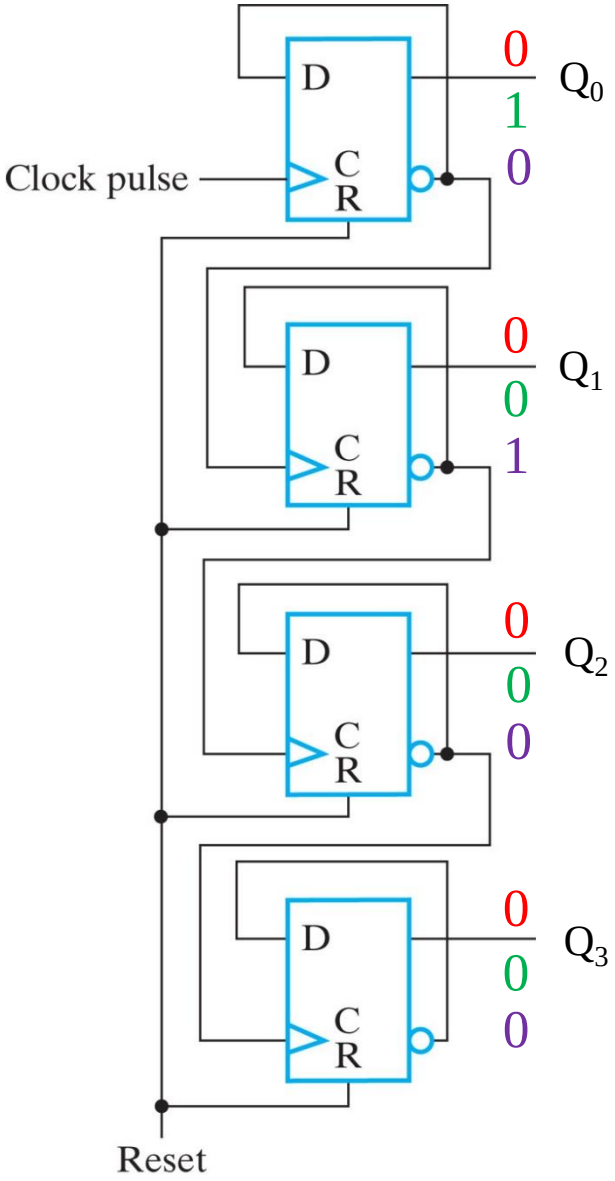
Ripple Counter

4-bit Binary Ripple Counter (Count Up)



- Chain of 4 D positive-edge-triggered flip-flops: the **outputs Q_0, Q_1, Q_2, Q_3 represent the counter** output
- The FF corresponding to the LSB Q_0 receives the system clock directly. Each subsequent FF receives the **complemented output of the previous FF as clock**
- The **input** of **each FF is connected to its complemented output**: at the rising edge of the corresponding C, each FF inverts its state
- A Reset signal can be used to clear the register to all zeros, asynchronously

4-bit Binary Ripple Counter (Count Up)

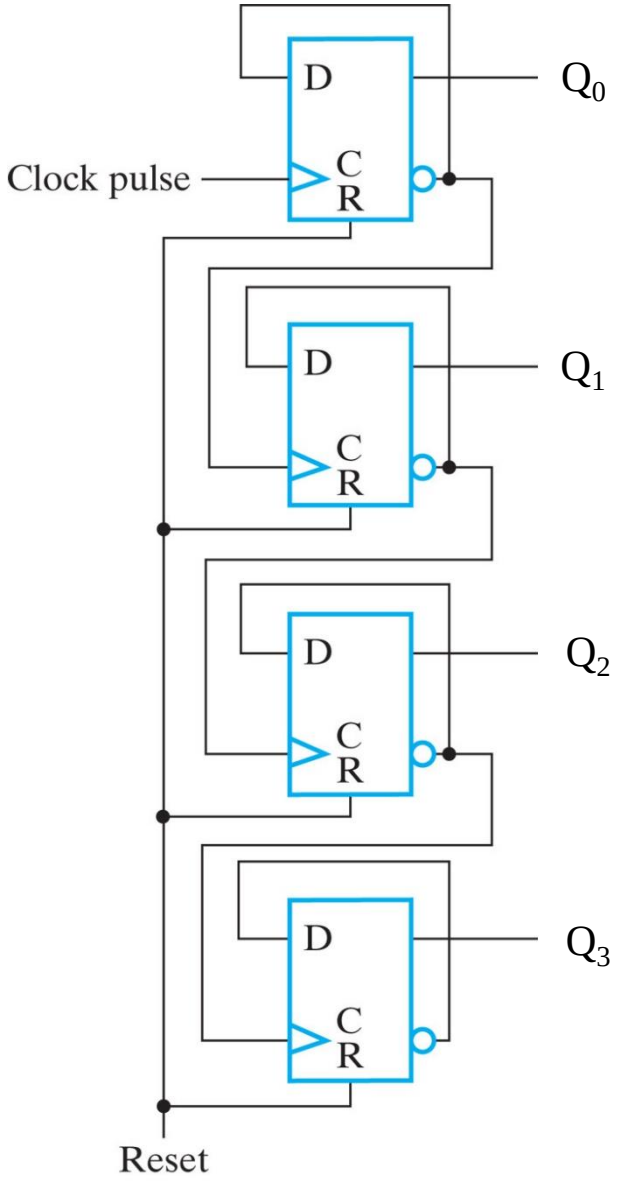


- Reset (Q_i set to '0'): the output (count) is cleared to "0000"
- At the first rising edge of the system clock:
  - FF_0 is triggered by the clock: Q_0 goes from 0 to 1, $\overline{Q_0}$ from 1 to 0
  - The remaining FFs are not triggered (no rising edge): Q_1,Q_2,Q_3 unchanged

  => the output (count) goes from "0000" to "0001"
- At the second rising edge of the system clock:
  - FF_0 is triggered by the clock: Q_0 goes from 1 to 0, $\overline{Q_0}$ from 0 to 1
  - FF_1 is triggered by $\overline{Q_0}$: Q_1 goes from 0 to 1
  - FF_2-FF_3 are not triggered (no rising edge on C input): Q_2,Q_3 unchanged

  => the output (count) goes from "0001" to "0010"
- When Q_i changes from 0 to 1, no change is induced in Q_{i+1}. When Q_i changes from 1 to 0, Q_{i+1} is complemented
- Transition time depends on the number of FFs in the chain (similar to ripple-carry adder)

4-bit Binary Ripple Counter (Count Up)



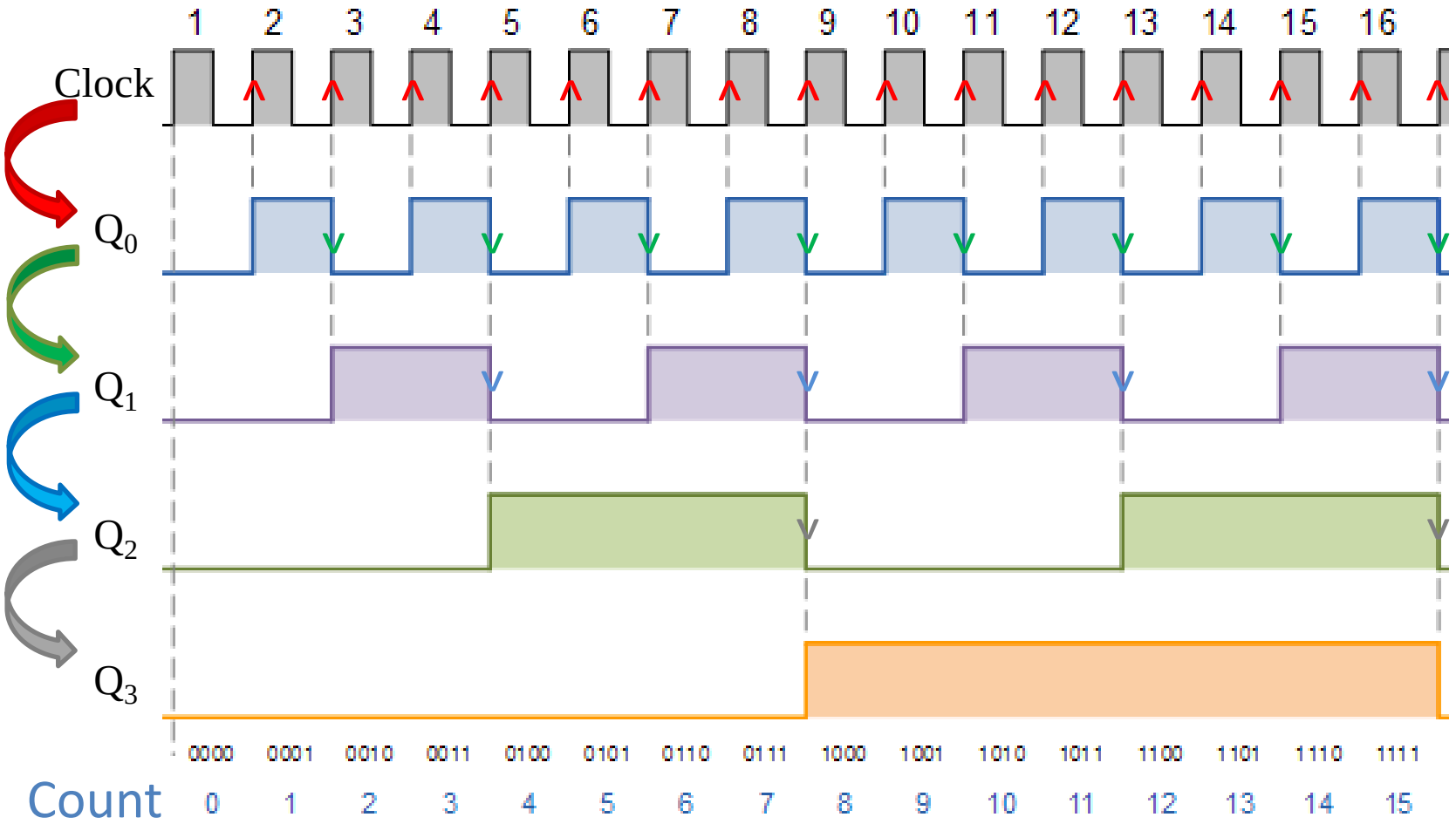
Upward Counting Sequence

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

Count up

Q_0 switches at each rising edge of the system clock, Q_1 switches if Q_0 goes from 1 to 0, Q_2 switches if Q_1 goes from 1 to 0, and so on ...

4-bit Binary Ripple Counter (Count Up): Time Diagram



4-bit Binary Ripple Counter (Count Down)

- If the **clock of each FF is connected to the direct output** (instead of complemented output) of the previous FF, we obtain a downward counting sequence
- Q_0 switches at each rising edge of the system clock, Q_1 switches if Q_0 goes from 0 to 1, Q_2 switches if Q_1 goes from 0 to 1, and so on ...

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

Count down

Ripple Counter: Pros and Cons

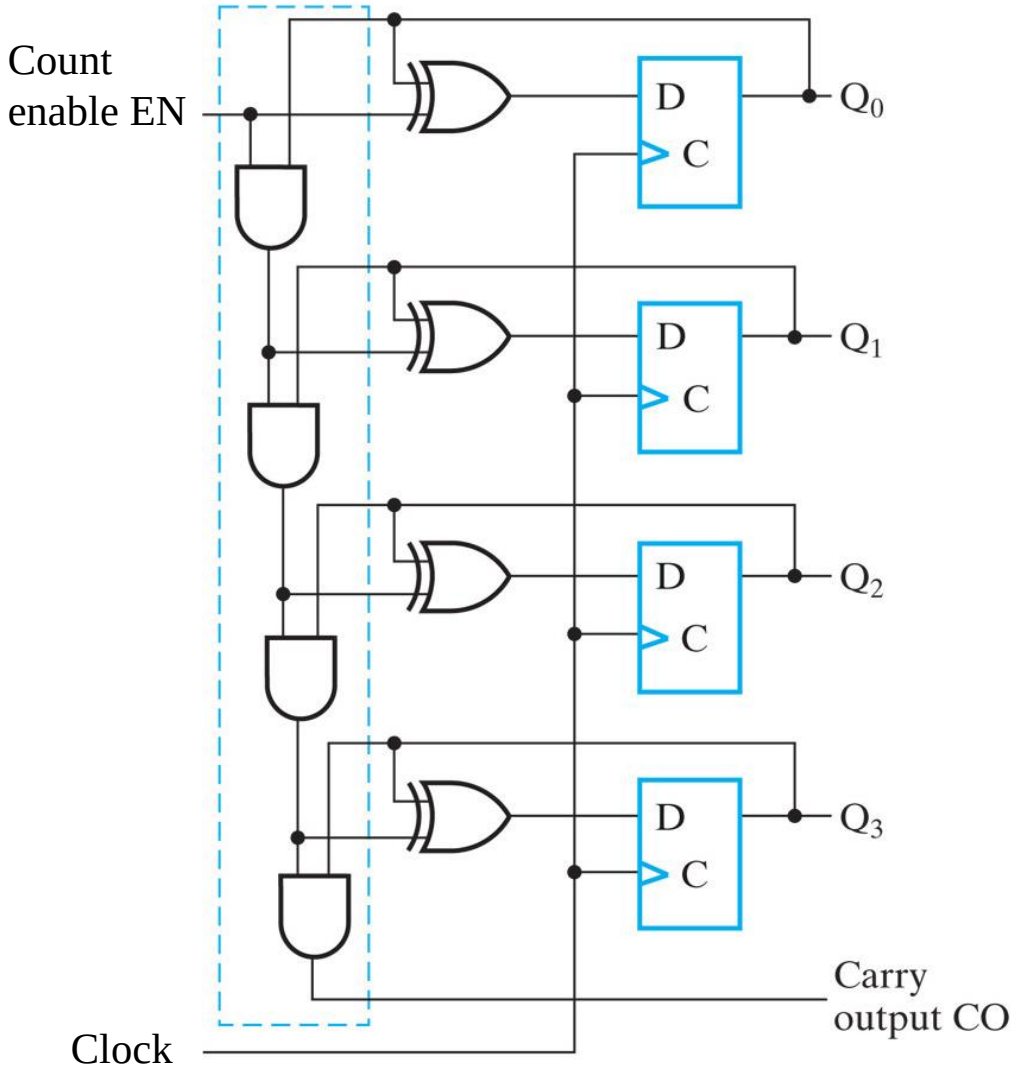
- Ripple counters have the advantage of having a **simple hardware**
- However, they are asynchronous circuits, so the output bits have **different delays** (especially if the counter contains a large number of bits), since the flip-flops change their state one at a time in sequence, being triggered each from the previous. This can lead to reliability issues
- Ripple counters can be used in circuits that do not have stringent constraints on the operating speed (i.e. the clock frequency is sufficiently low). In general, ripple counters are favored in low-power designs

Synchronous Binary Counters

- Unlike the ripple counter, in a synchronous counter all **flip-flops are regulated by the same clock signal**, so they are triggered all at the same time (and not one after the other, as in the ripple counter)

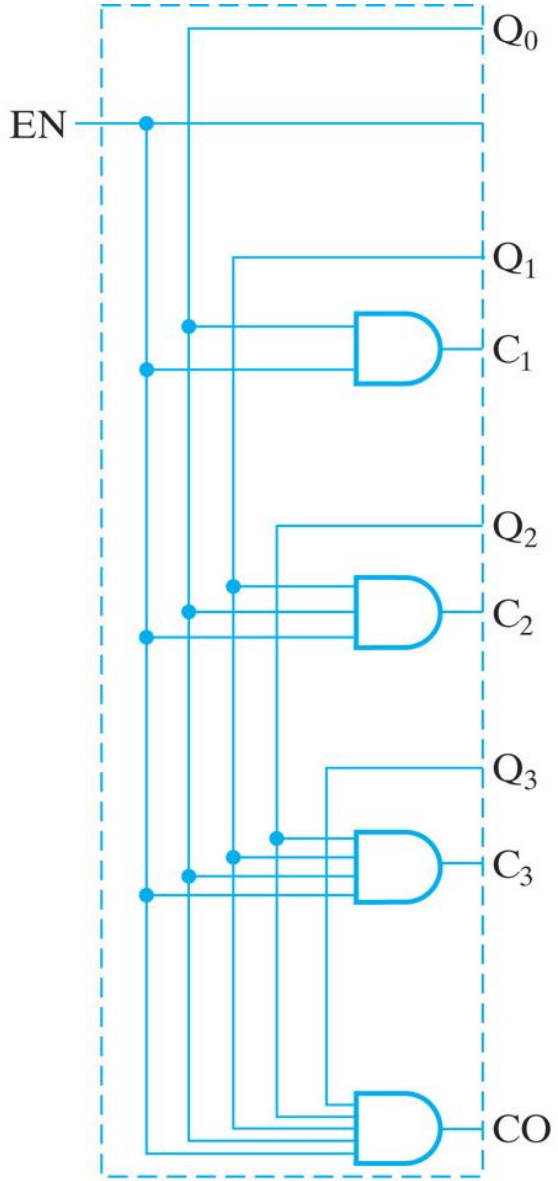
Synchronous Binary Counters

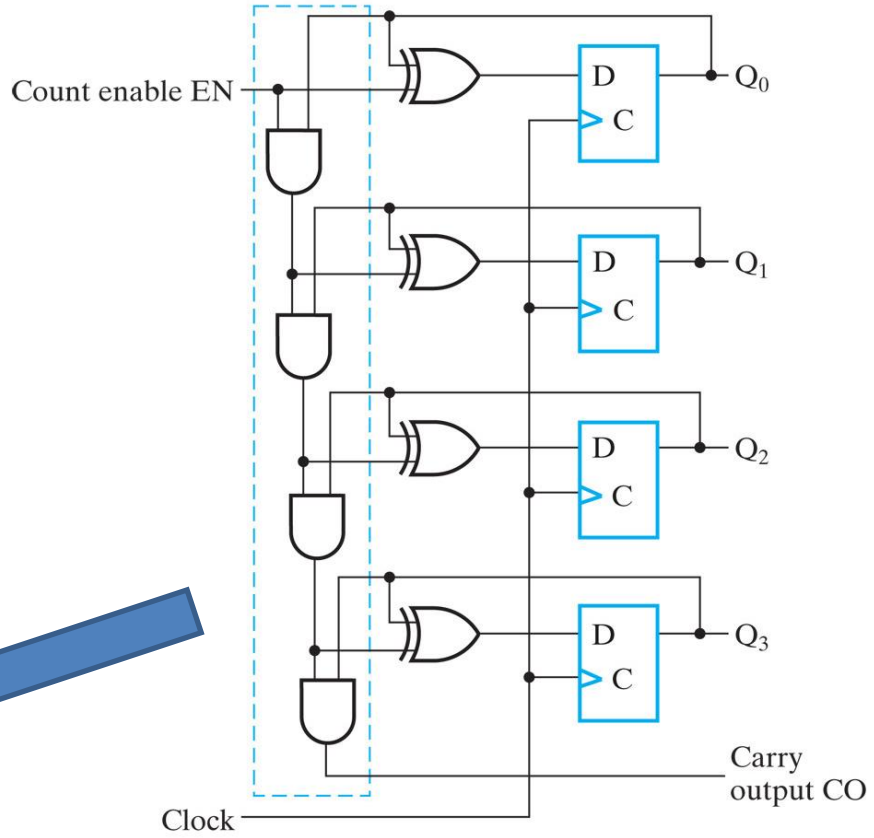
Binary Serial Synchronous Counter



- Clock is in common to all FFs
- EN (input): if EN = 1 **the count is enabled**
- If **EN = 1**: at each rising edge of the clock, the value stored in the register is increased by 1
- XOR gates perform the sum (increment by 1 the value stored in the register), the AND gates propagate the carry from one stage to the next
- CO (carry output): can be used to create multistage counters with a hierarchical structure
- In case of a 4-bit counter, a propagation delay of 4 AND gates is needed to generate CO, 3 AND + 1 XOR propagation delay for Q_3

Binary Parallel Synchronous Counter

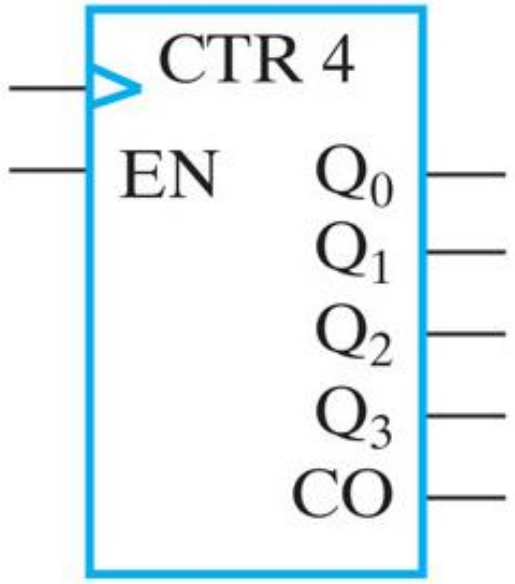




By replacing the chain of 2-input AND gates with the same number of multi-input AND gates in parallel, the counter will be faster

Binary Synchronous Counter

Symbol of the 4-bit binary synchronous counter



If we connect together two 4-bit binary synchronous counters (with the carry output CO of the first counter attached to the EN input of the second counter), we get an 8-bit counter

Summary

- We have seen possible implementations of two of the most used blocks in digital systems: **shift registers** and **counters**
- A shift register consists of a chain of flip-flops. It shifts the bits to the right or to the left (unidirectional shift register) or to both directions (bidirectional or universal)
- A counter consists of a chain of flip-flops, the outputs of which provide a count (up, down, etc.)
- A ripple counter is an asynchronous counter, where the clock of each flip-flop is provided by the previous flip-flop
- A synchronous counter consists of a chain of flip-flops, all controlled by the same clock signal
  - A synchronous counter has a slightly more complex structure than the ripple counter, but is faster and less prone to timing issues (i.e. more reliable)

Disclaimer

Figures from*Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano |Kime| Martin