

Ministry of Higher Education and Scientific Research
University of Larbi Ben M'Hidi, Oum El Bouaghi
Faculty of Exact Sciences and Natural and Life Sciences
Department of Mathematics and Computer Science

Computer Structure 2



Presented by: **Dr. NASRI/A**
nasri.ahlem1988@gmail.com
nasri.ahlem@univ-oeb.dz

2023-2024

Target audience

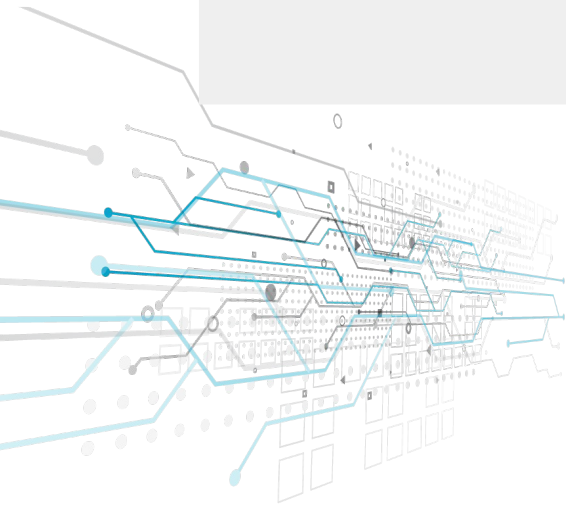
1st year mathematics student + 1st year Common core Mathematics and computer science students .

Prerequisite

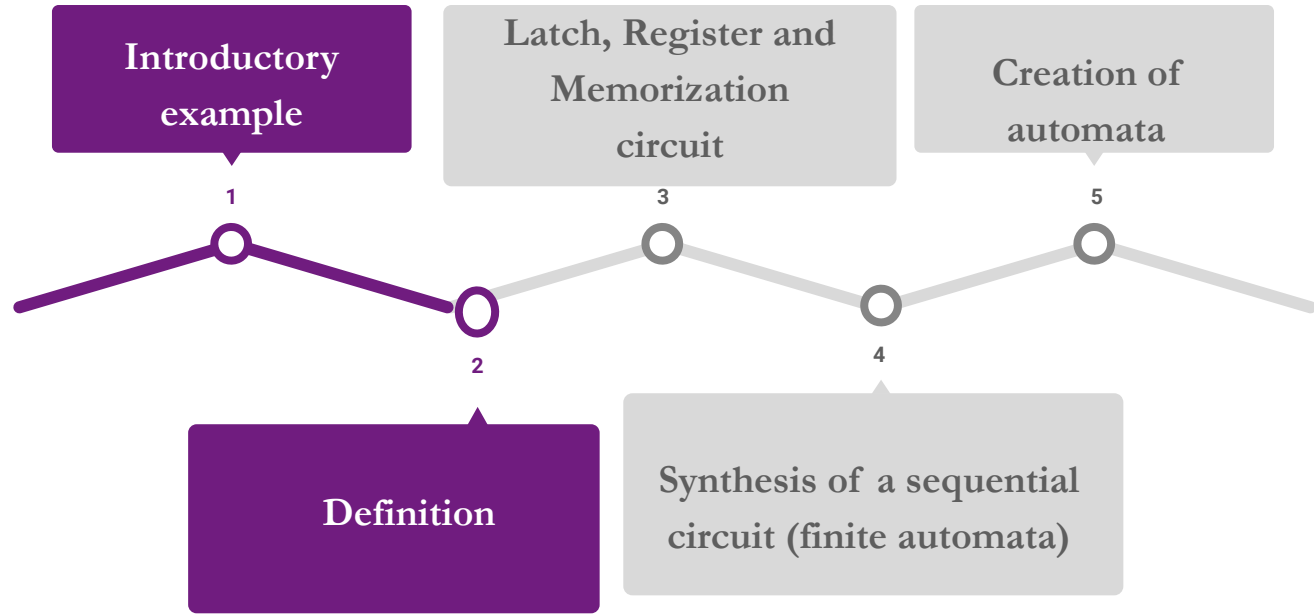
Students must have basic computer skills.

Assessment method

Assessment method: Exam (60%), Continuous assessment (40%)

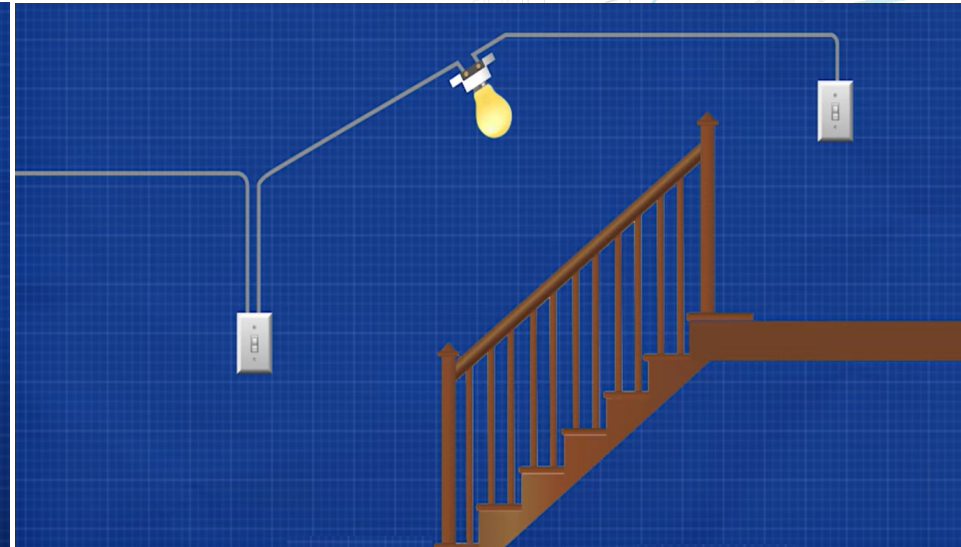
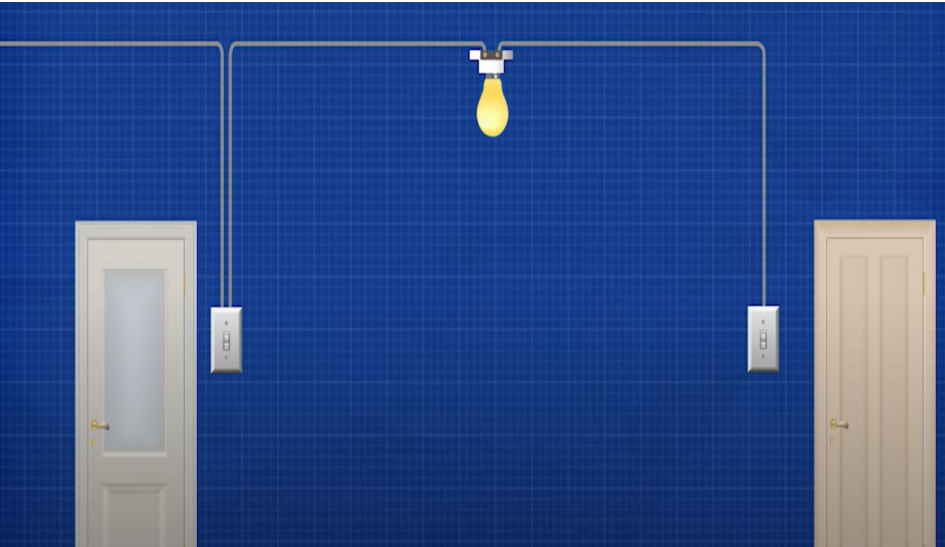


Chapter 2: Sequential logic



A two-button light switch controlling a single bulb

Let's imagine you have switches (A and B) at each end of a corridor or a staircase and a lamp on the ceiling.



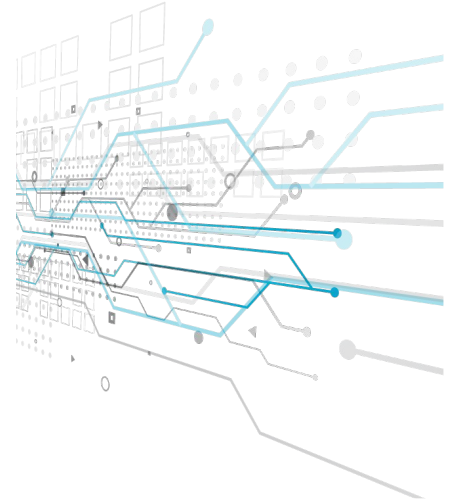
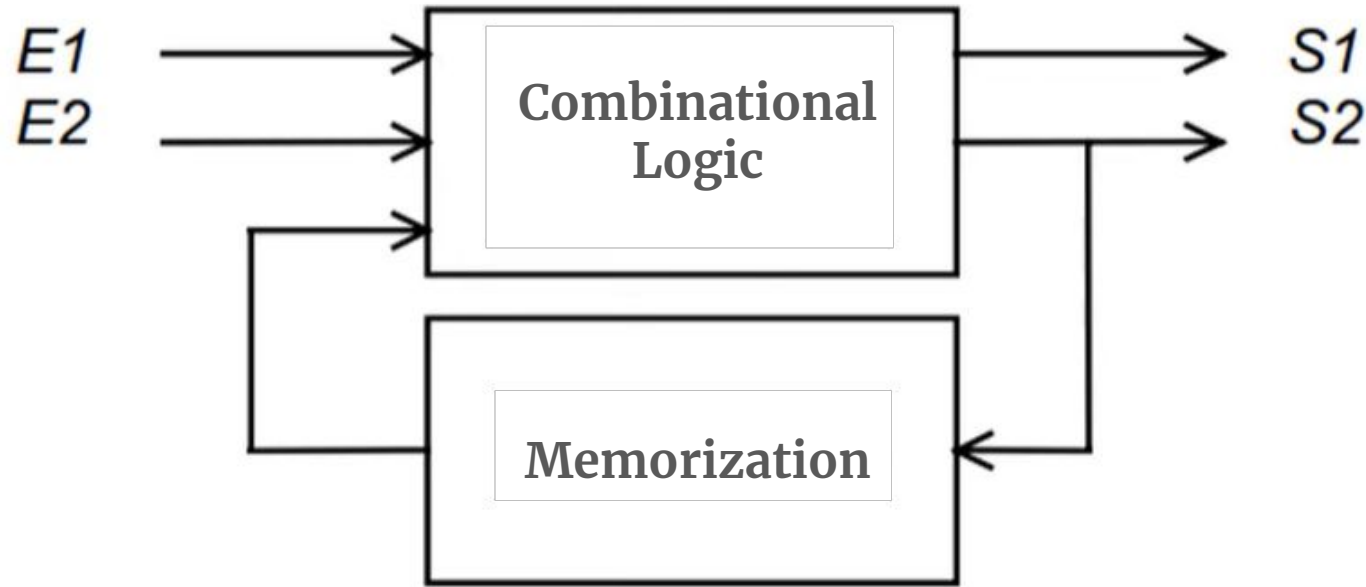
A two-button light switch controlling a single bulb

Here's how it works:

- When switch **A** is in the off position and switch **B** is in the off position, the light remains off.
- If you press switch **A**, the light turns on. Now, regardless of the position of switch **B**, the light remains on.
- If you press switch **B** while the light is on, the light turns off. Regardless of the position of switch **A** at this point, the light remains off.

"Could you propose an implementation of a circuit that controls this lamp?"

A two-button light switch controlling a single bulb



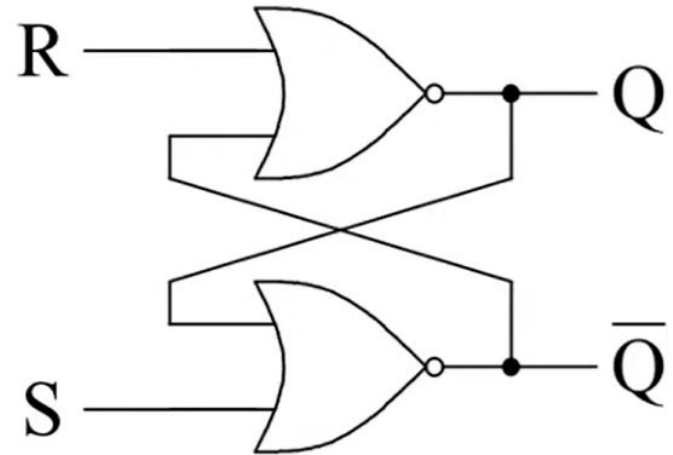
A sequential circuit: A sequential circuit is a circuit whose output state depends **not only** on the inputs but also on the previous state of the outputs.

Therefore, these circuits must be capable of **memorizing**.

→ It may depend also on **the past history of inputs**.

→ Sequential circuits are commonly used in applications where past history or context is important, such as in counters, registers, and finite state machines.

Example



Sequential Logic

Basic sequential functions

Memorization

Counting

Shifting

Fondamental sequential circuits

Latching (3 types)

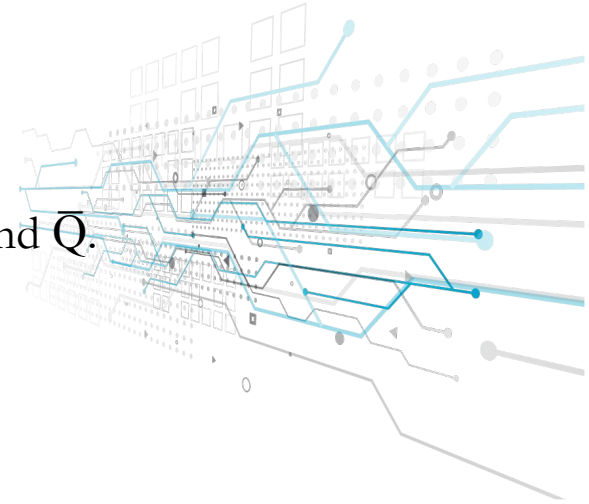
Counters /Decounters

Registers

RAM(Random Access
Memory)

Latch circuit: is the basic element of sequential logic; latching refers to the ability of a circuit to retain its output state after the input signals have been removed or changed.

- Once a latching circuit has been set to a certain state, it will maintain that state until it receives a specific signal to change it.
- Consisting essentially of :
 - A set of **inputs** : $I_1, I_1, I_1, \dots I_n$
 - **One output Q** or **two complementary outputs Q** and \bar{Q} .



There are two primary types of operations associated with latches:

- ➔ **Asynchronous mode:** In asynchronous operation, the outputs of the latch can change immediately in response to changes in the input signals, without any requirement for a clock signal.

This type is sensitive to input changes at any time, regardless of the clock signal.

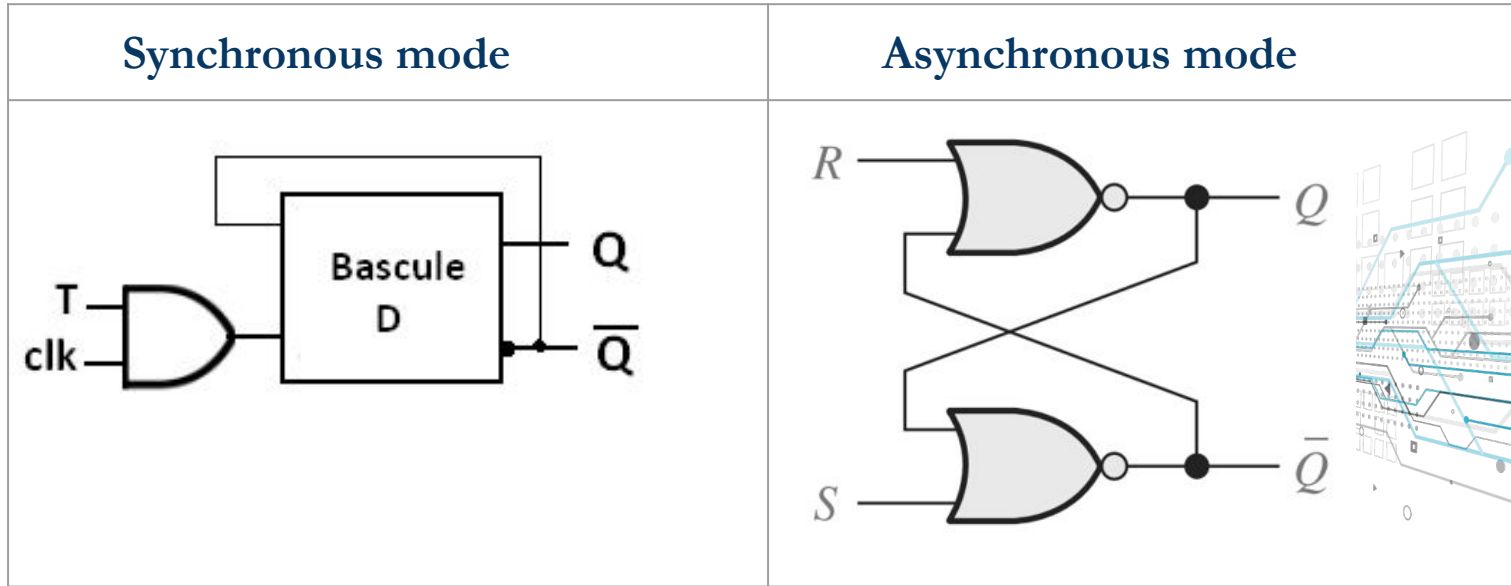
Ex: SR (Set-Reset) latches, can change their output states as soon as the input signals change.

- ➔ **Synchronous mode:** In synchronous operation, the changes in the output of the latch are synchronized to a **clock signal**.

The outputs of synchronous latches only change on specific edges of the clock signal,

Ex: D (Data) latches or flip-flops, have dedicated clock inputs that control when the inputs are sampled and when the outputs are updated.

There are two primary types of operations associated with latches:



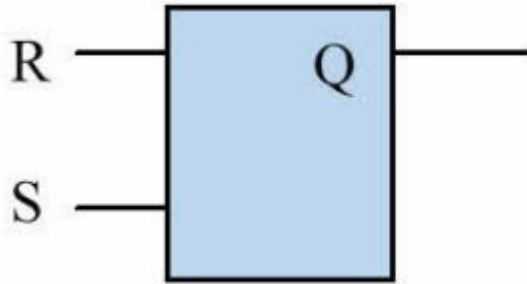
NB: For Synchronous mode we have three subtypes: **level synchronization**, **edge synchronization** and **pulse synchronization**.

Level synchronization		Edge synchronization		Pulse synchronization
Hight level	Low Level	Raising Edge	Descending edge	
<div> <div>CLK</div> <div>→</div> <div></div> </div>	<div> <div></div> </div>	<div> <div>CLK</div> <div>→</div> <div></div> </div>	<div> <div>CLK</div> <div>●→</div> <div></div> </div>	<div> </div>

Asynchronous Latches

→ **Latch SR (Set-Reset):** An SR latch, or Set-Reset latch, is a basic memory element in sequential logic.

It has two inputs, S (Set) and R (Reset), one for **setting to 0** and the other for **setting to 1** of the output Q (so as output we will have Q and \bar{Q}).



R_t	S_t	Q_{t+}	The operation
0	0	Q_t	Memory
0	1	1	Setting to 1
1	0	0	Setting to 0
1	1		Not allowed

Operation table of the SR circuit


Asynchronous Latches


→ Latch SR (Set-Reset):

According to the operating table we distinguish three operating modes of an SR circuit:

- **Memory Mode:** $R=S=0$, the output remains in the state it was in (0 or 1)
- **Setting to 1 of the output:** $S=1$ and $R=0$
- **Setting to 0 of the output:** $S=0$ and $R=1$

NB: the case $S=1$ and $R=1$ is not usable because it leads to both situations setting to 1 and setting to 0 in the same time which is not allowed

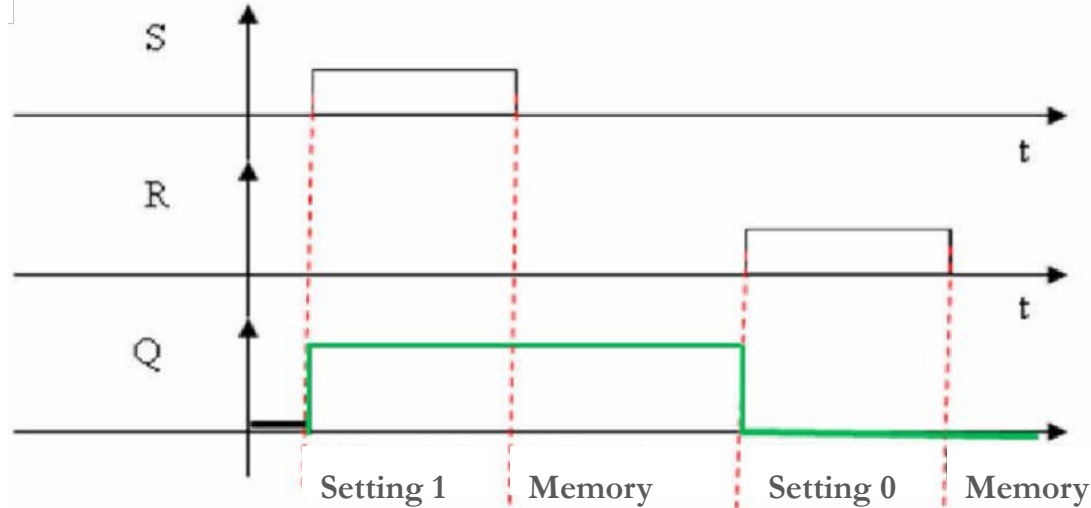


R_t	S_t	Q_{t+}	The operation
0	0	Q_t	Memory
0	1	1	Setting to 1
1	0	0	Setting to 0
1	1		Not allowed

Operation table of the SR circuit

Asynchronous Latches

- Latch SR (Set-Reset):
Example:



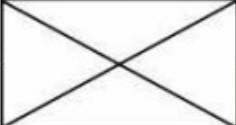
Example of a chronogram of an SR Circuit



Asynchronous Latches

→ Latch SR (Set-Reset):

we can draw up the operating table in another way

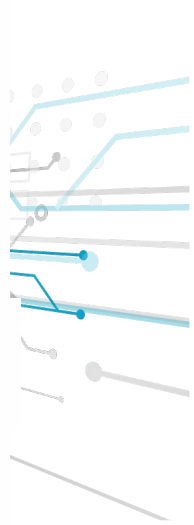
R	S	Q_t	Q_{t+}	
0	0	0	0	Memory
0	0	1	1	
0	1	0	1	Setting to 1
0	1	1	1	
1	0	0	0	Setting to 0
1	0	1	0	
1	1	0		Not allowed
1	1	1		

$$Q_{t+} = f(R, S, Q_t)$$

Avec:

Q_{t+} : present state

Q_t : past state



A two-button light switch controlling a single bulb

Here's how it works:

- When switch **A** is in the off position and switch **B** is in the off position, the light remains off.
- If you press switch **A**, the light turns on. Now, regardless of the position of switch **B**, the light remains on.
- If you press switch **B** while the light is on, the light turns off. Regardless of the position of switch **A** at this point, the light remains off.

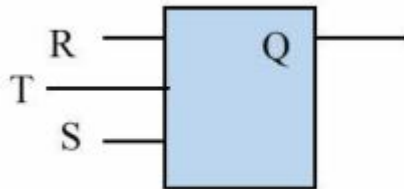
"Could you propose an implementation of a circuit that controls this lamp, Now?"

Synchronous Latches

→ Synchronous Latch Set-Reset (SRT):

The term "**synchronous**" in **Synchronous Latch Set-Reset** refers to the fact that the latch operates based on a **clock signal T**. It has two distinct operations:

- **T=0** → The output does not change regardless of the R and S inputs; it is a memory operation (the latch is not synchronized)
- **T=1** → The latch is then synchronized, the output respects the operating table of the RS Latch.

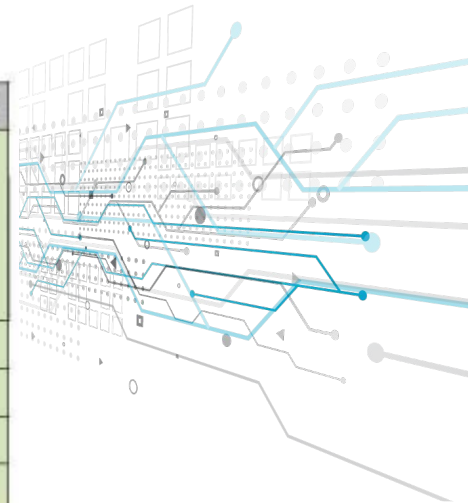


Synchronous Latches

→ Synchronous Latch Set-Reset (SRT):

The term "synchronous" in **Synchronous Latch Set-Reset** refers to the fact that the latch operates based on a **clock signal T**. It has two distinct operations:

T	R	S	Q_{t+}	Operation
0	0	0	Q_t	Memory
0	0	1	Q_t	
0	1	0	Q_t	
0	1	1	Q_t	
1	0	0	Q_t	Memory
1	0	1	1	Set to 1
1	1	0	0	Set to 0
1	1	1	x	Not allowed



Operation table of the SRT circuit

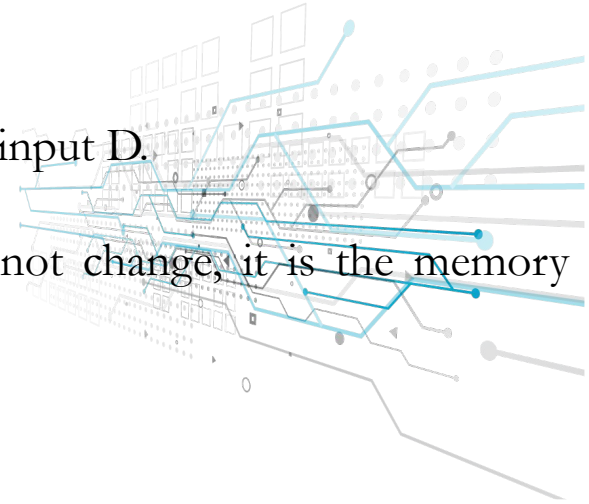
Synchronous Latches

→ D Latch:

A D Latch has a **single data input** (usually denoted as **D**), a **clock input** (often labeled as **CLK**), and an **output** (typically referred to as **Q**).

With a single input we can only have two operating modes:

- If the synchronization signal is active, the output copies the input D.
- If the synchronization signal is inactive, the output does not change, it is the memory operating mode.

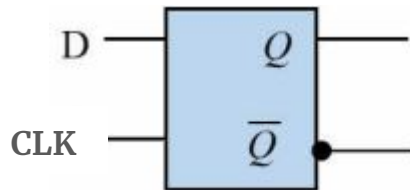


Synchronous Latches

→ D Latch:

A D Latch has a **single data input** (usually denoted as **D**), a **clock input** (often labeled as **CLK**), and an **output** (typically referred to as **Q**).

With a single input we can only have two operating modes:



D	CK	Q_t	Q_{t+}	Operation
0	0	0	0	memorize Q_t
0	0	1	1	
0	1	0	0	Copy D
0	1	1	0	
1	0	0	0	memorize Q_t
1	0	1	1	
1	1	0	1	Copy D
1	1	1	1	

Synchronous Latches

→ D Latch:

D Latch output equation:

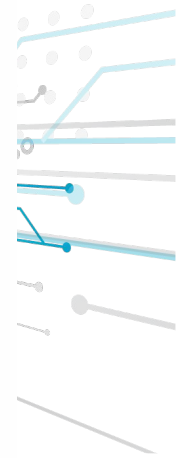
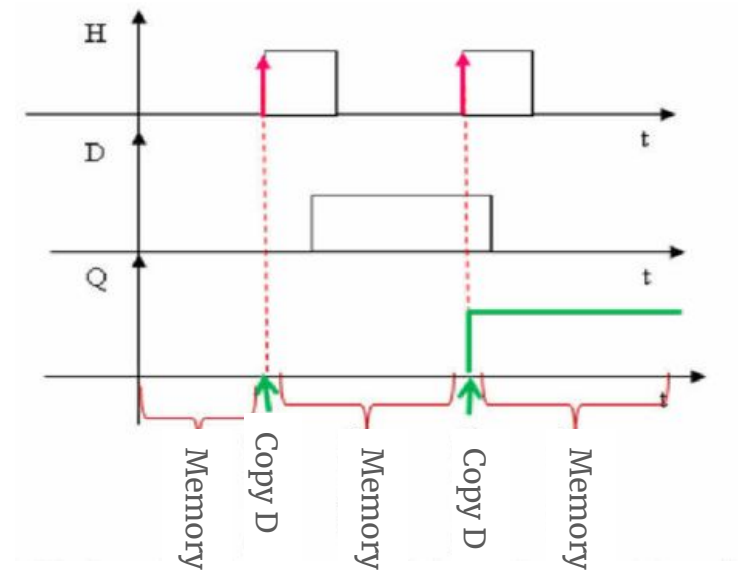
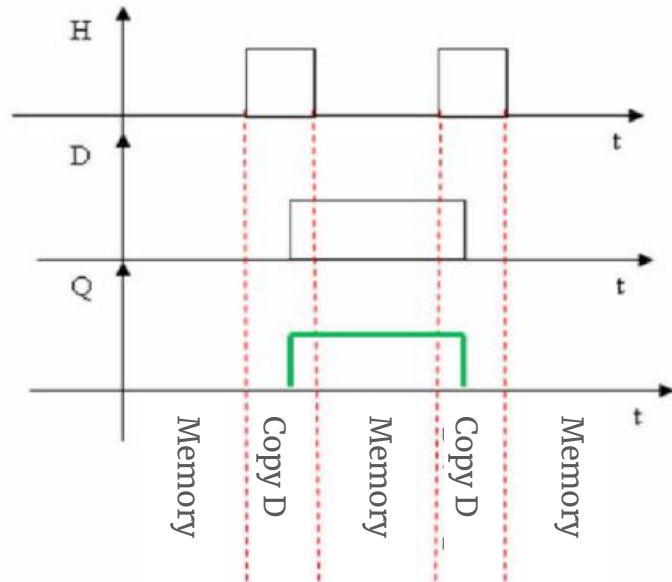
the D latch is produced by the RST latch where the R and S inputs are linked by the relationship $D=S=\overline{R}$

- In the phase where the clock is inactive we have $Q_{t+} = Q_t$ (memory function)
- In the phase where the clock is active we have $Q_t = D_t$ (copy function)

NB: The transition to memory position means that the D latch memorizes the last copied value.

Synchronous Latches

→ D Latch:
Examples:

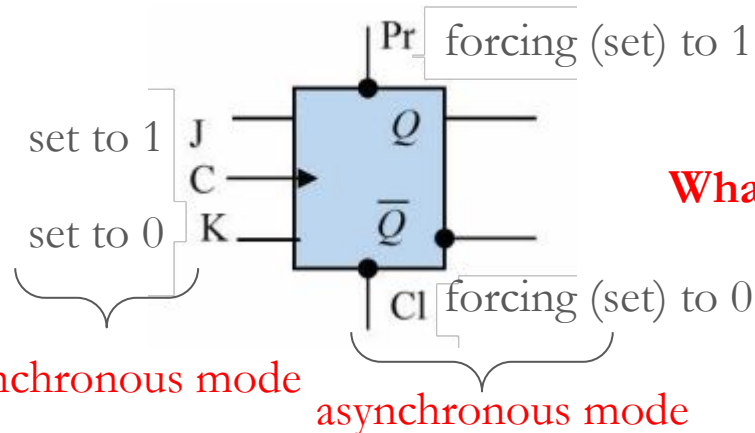


Synchronous Latches

→ JK Latch:

A JK latch has two main inputs: **J** (for Set to 1), **K** (Reset to 0), and **CLK** (clock).

It also has an output **Q** and its complement \bar{Q} .



What is the difference between RS latch and a JK latch?



Synchronous Latches

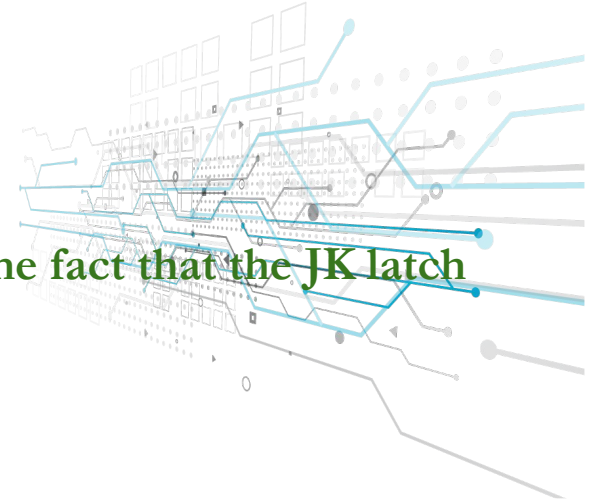
→ JK Latch:

A JK latch has two main inputs: **J** (for Set to 1), **K** (Reset to 0), and **CLK** (clock).

It also has an output **Q** and its complement \bar{Q} .

What is the difference between RS latch and a JK latch?

The main difference between an RS and a JK latch lies in the fact that the JK latch does not have a prohibited state.



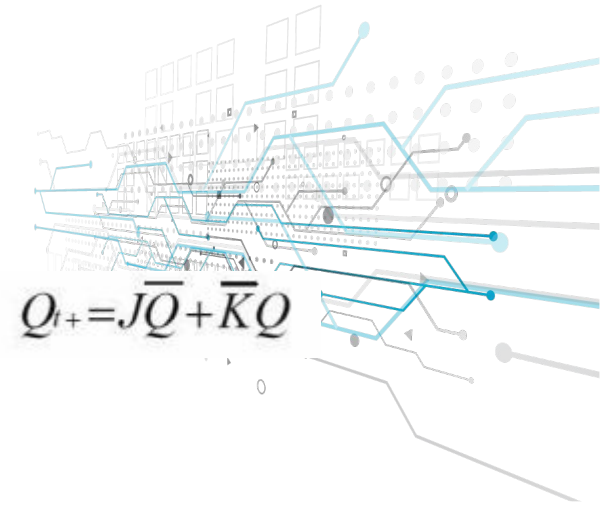
Synchronous Latches

→ JK Latch:

A JK latch has two main inputs: **J** (for Set to 1), **K** (Reset to 0), and **CLK** (clock).

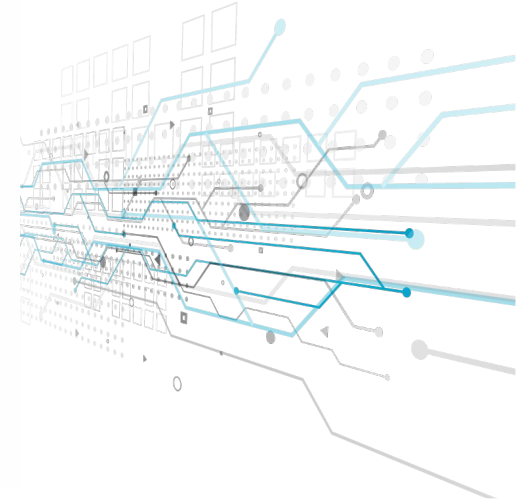
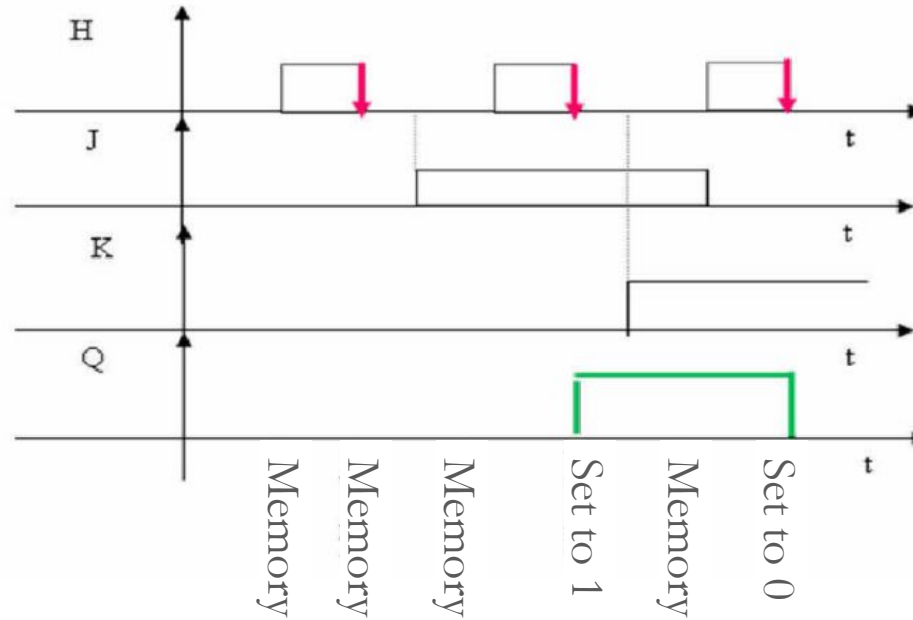
It also has an output Q and its complement \bar{Q} .

CK	J	K	Q_{t+}	Operation
↑	0	0	Q_t	Memory
↑	0	1	0	Set to 0
↑	1	0	1	Set to 1
↑	1	1	\bar{Q}_t	Toggle



Synchronous Latches

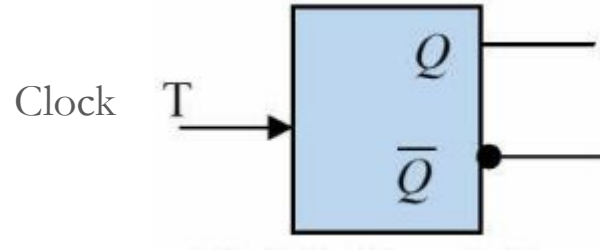
- JK Latch:
- Example:



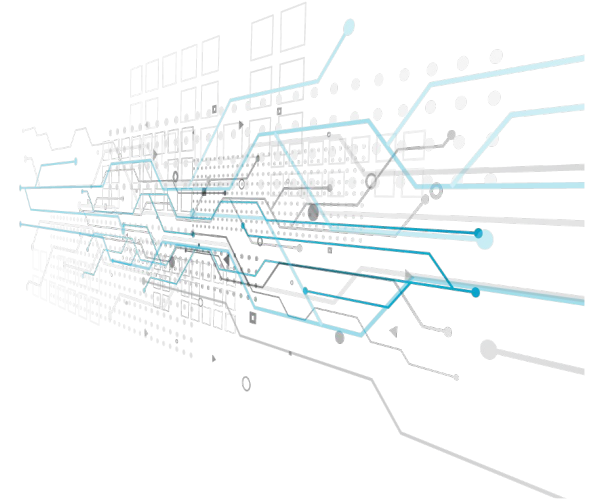
Synchronous Latches

→ Toggle Latch (T latch):

Its principal function is to toggle its output state each time it receives a clock signal, while holding the current state if no signal is present.

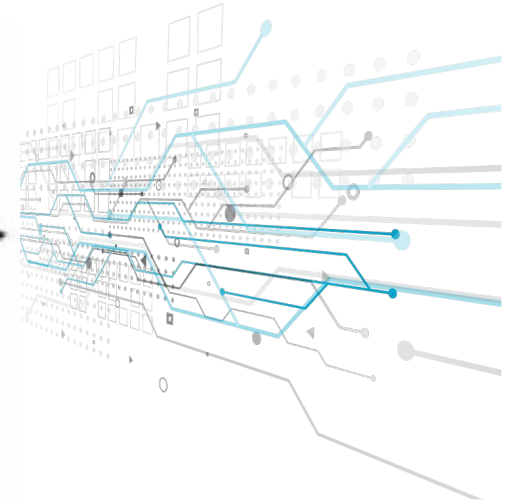
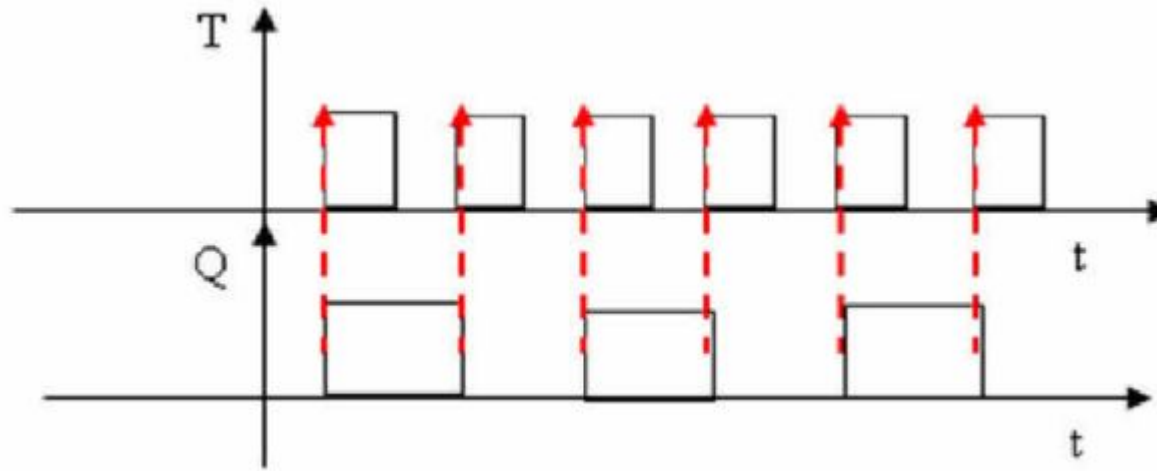


$$Q_{t+} = \overline{Q_t}$$



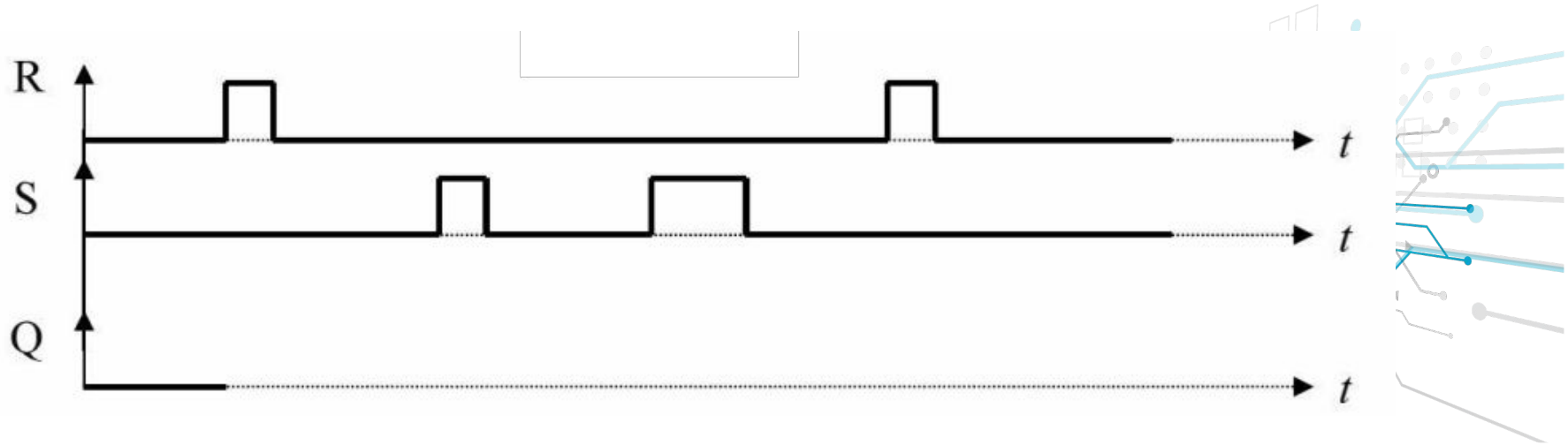
Synchronous Latches

- Toggle Latch (T latch):
- Example



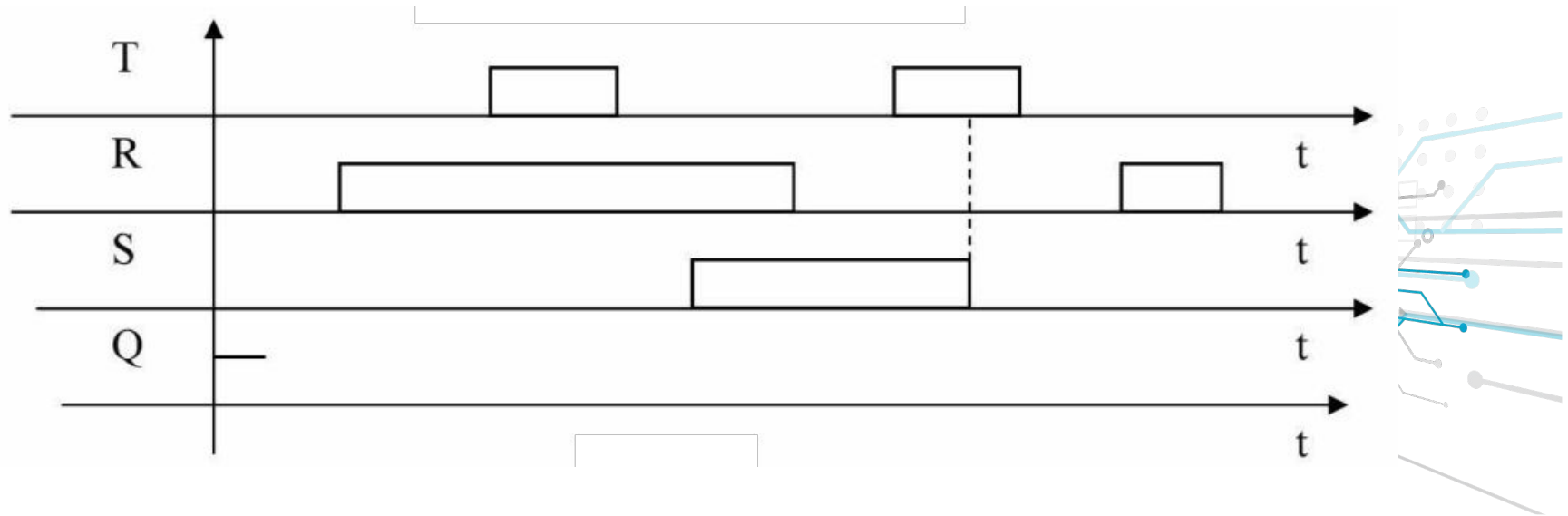
Exercise :

- Complete the chronogram (timing diagram) of the RS latch.

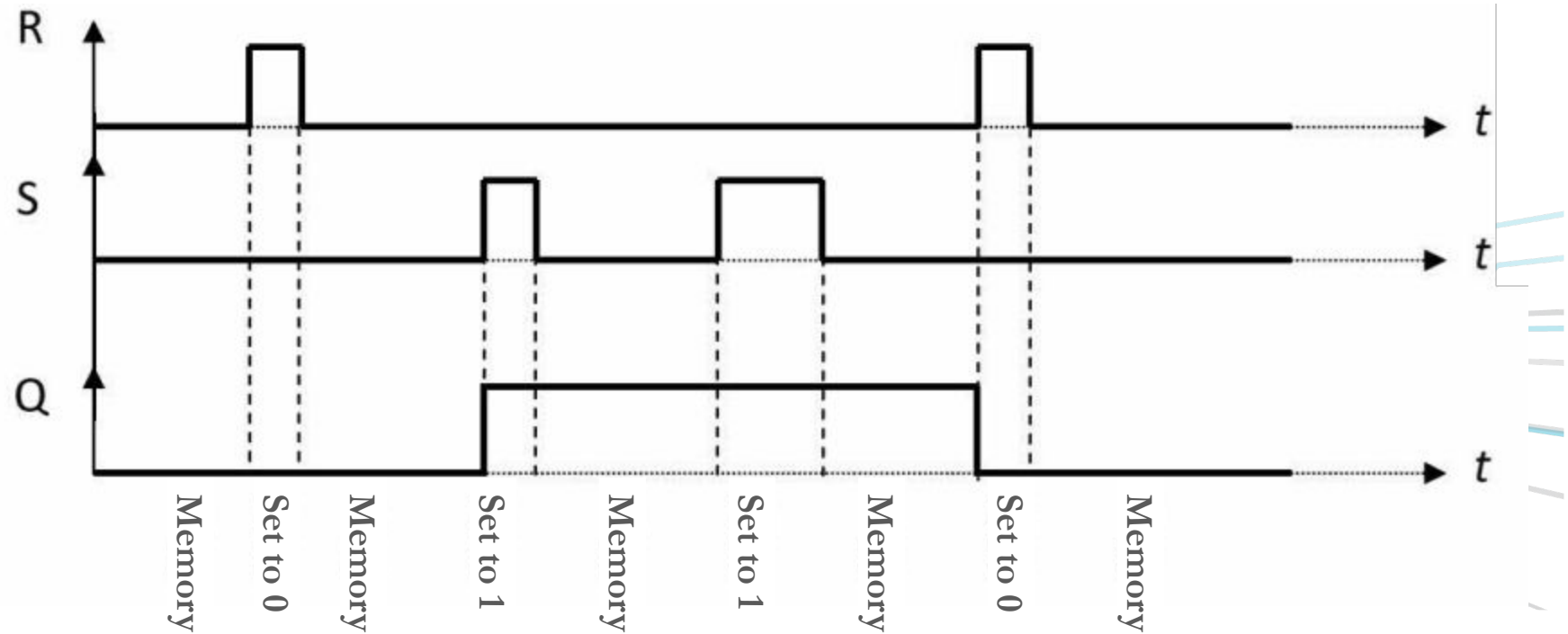


Exercise :

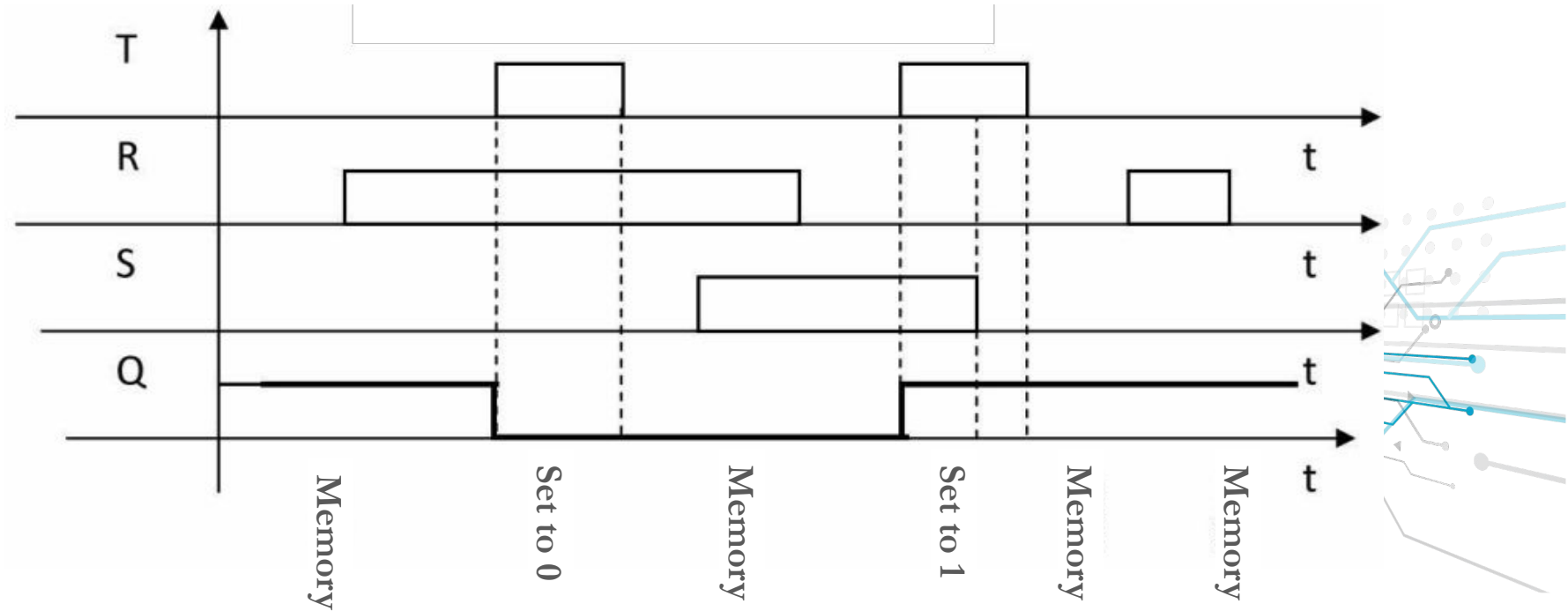
- Complete the chronogram of the SRT latch synchronized on the high level of T



Solution:



Solution:

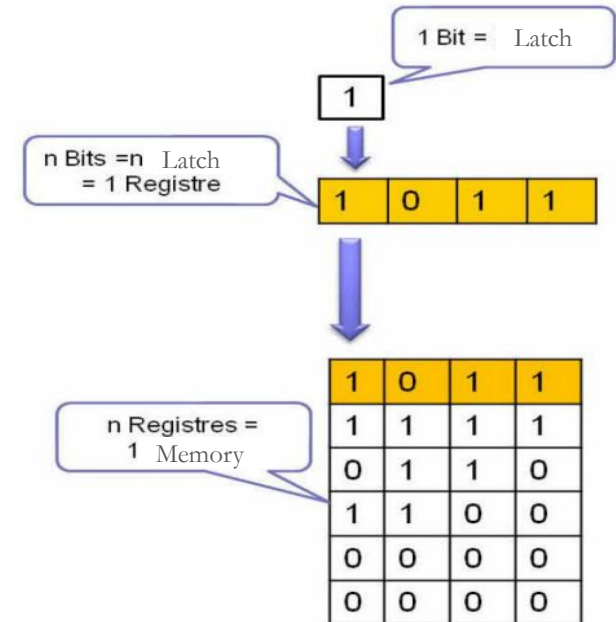


Register: First of all, a register is a set of boxes or memory cells capable of storing information.

- In the binary system, a memory cell is defined using a latch circuit. a register is therefore an ordered set of latches.

Functioning:

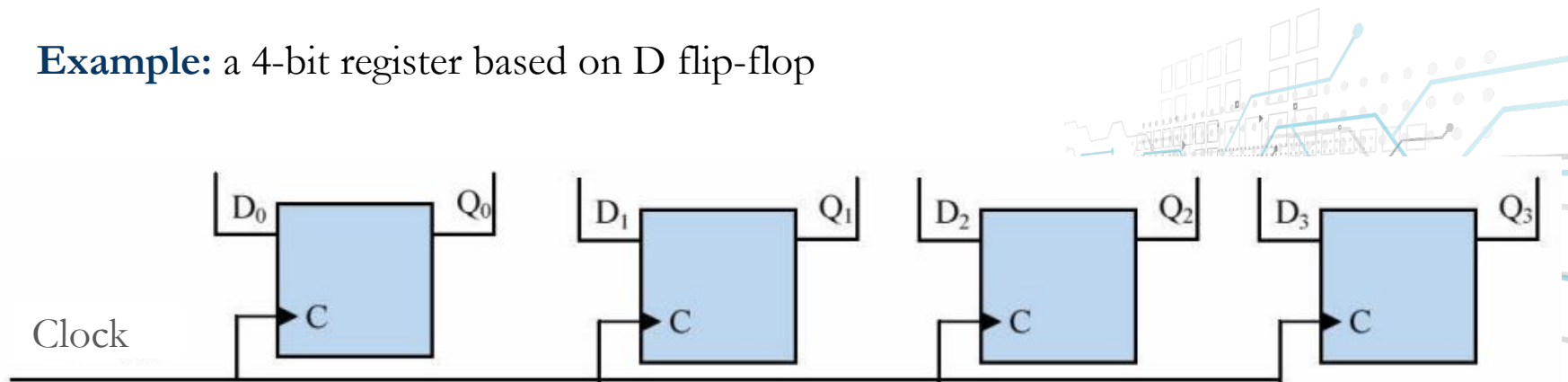
- A register set to memorize a word or a binary number.
- The diagram of such a system includes as many D Latches as there are binary elements to be memorized.
- All Latches are controlled by the same clock signal.



Register: First of all, a register is a set of boxes or memory cells capable of storing information.

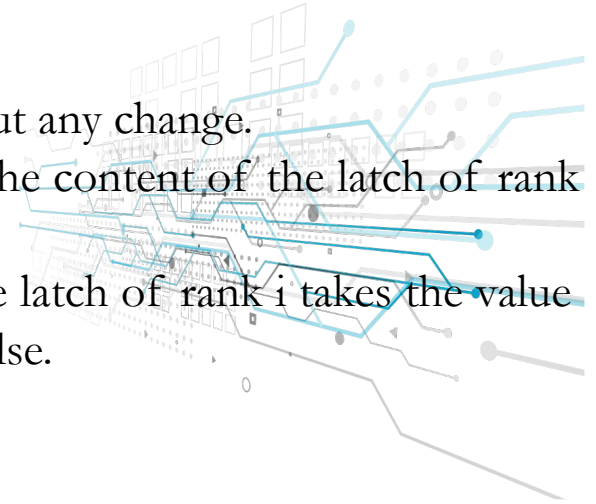
→ In the binary system, a memory cell is defined using a latch circuit. a register is therefore an ordered set of latches.

→ **Example:** a 4-bit register based on D flip-flop



Register: First of all, a register is a set of boxes or memory cells capable of storing information.

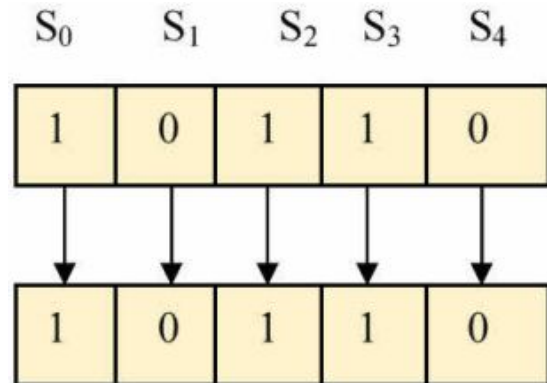
- In the binary system, a memory cell is defined using a latch circuit. a register is therefore an ordered set of latches.
- the main functions of a register are:
 - **Memorization:** memorizing information as it is without any change.
 - **The right shift:** shift information from left to right. The content of the latch of rank i is transmitted to that of rank $i+1$ at each clock pulse.
 - **The left shift:** shift information from right to left. The latch of rank i takes the value of the output of the latch of rank $i+1$ at each clock pulse.



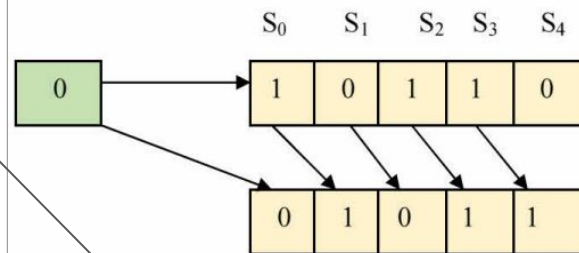
Register: First of all, a register is a set of boxes or memory cells capable of storing information.

- In the binary system, a memory cell is defined using a latch circuit. a register is therefore an ordered set of latches.
- The main functions of a register are:

Memorization



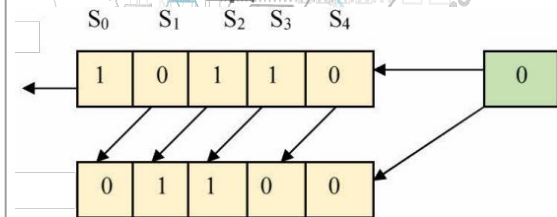
Right shift



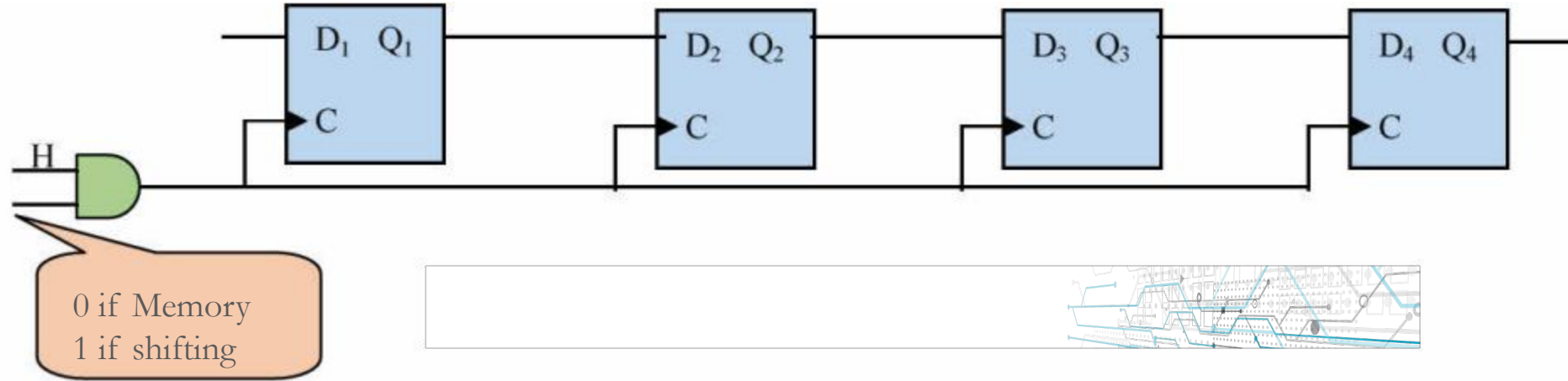
initial state

after a clock pulsestate

Left shift

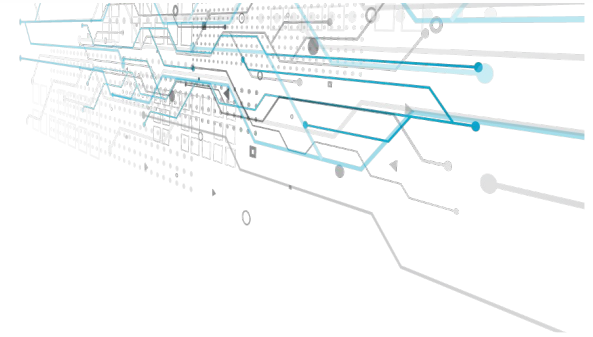
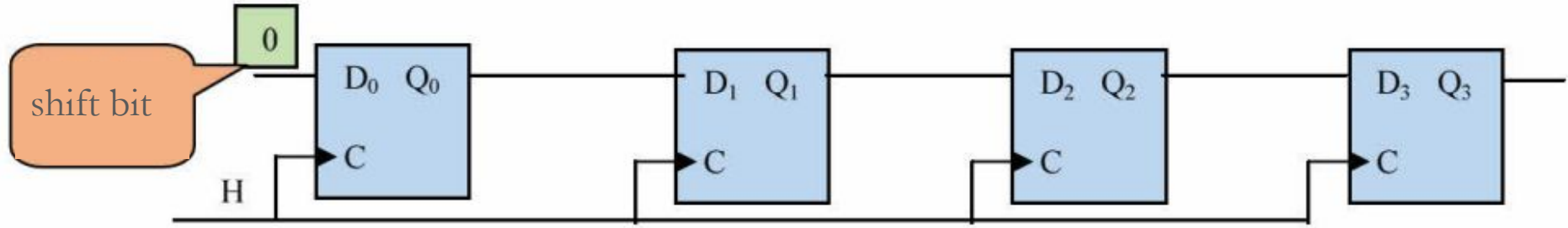


Example : Storage register

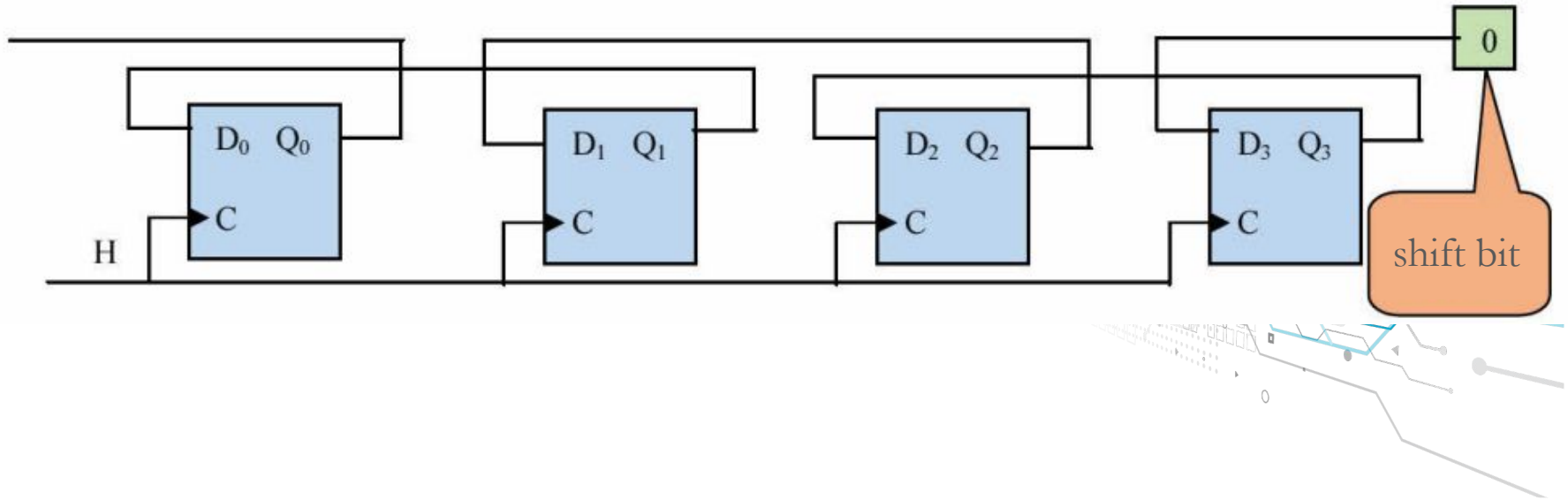


Prohibits clock action by inserting a serial AND gate

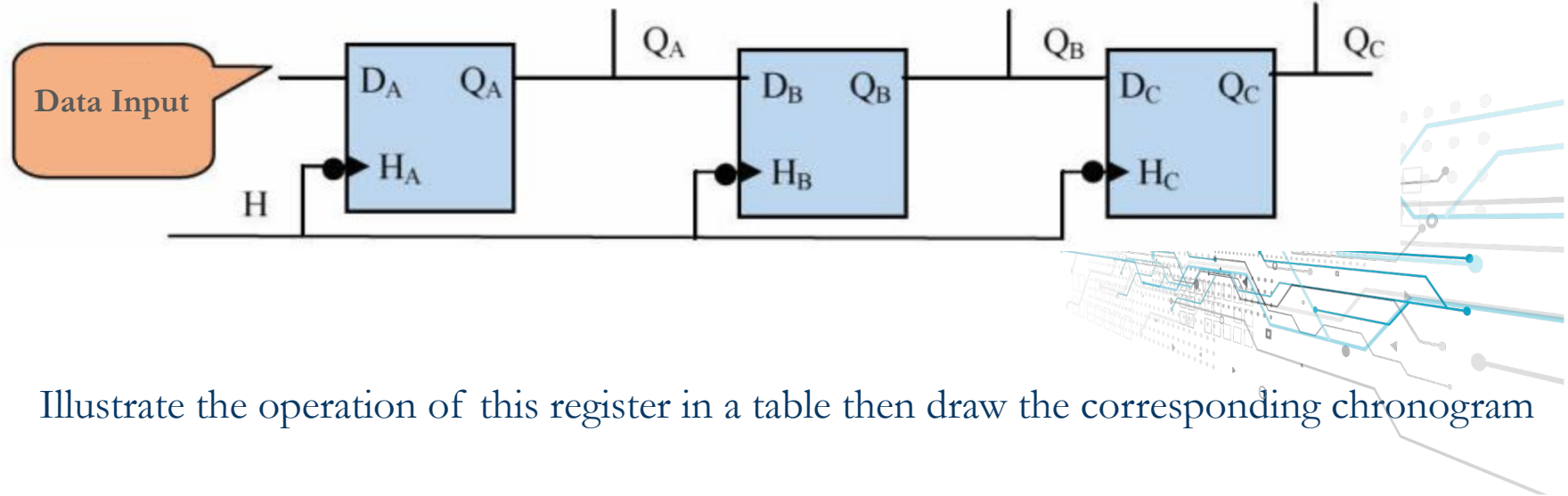
Example : Right shifting



Example : left shifting



Example: Consider the shifting register made up of 3 following D latches:



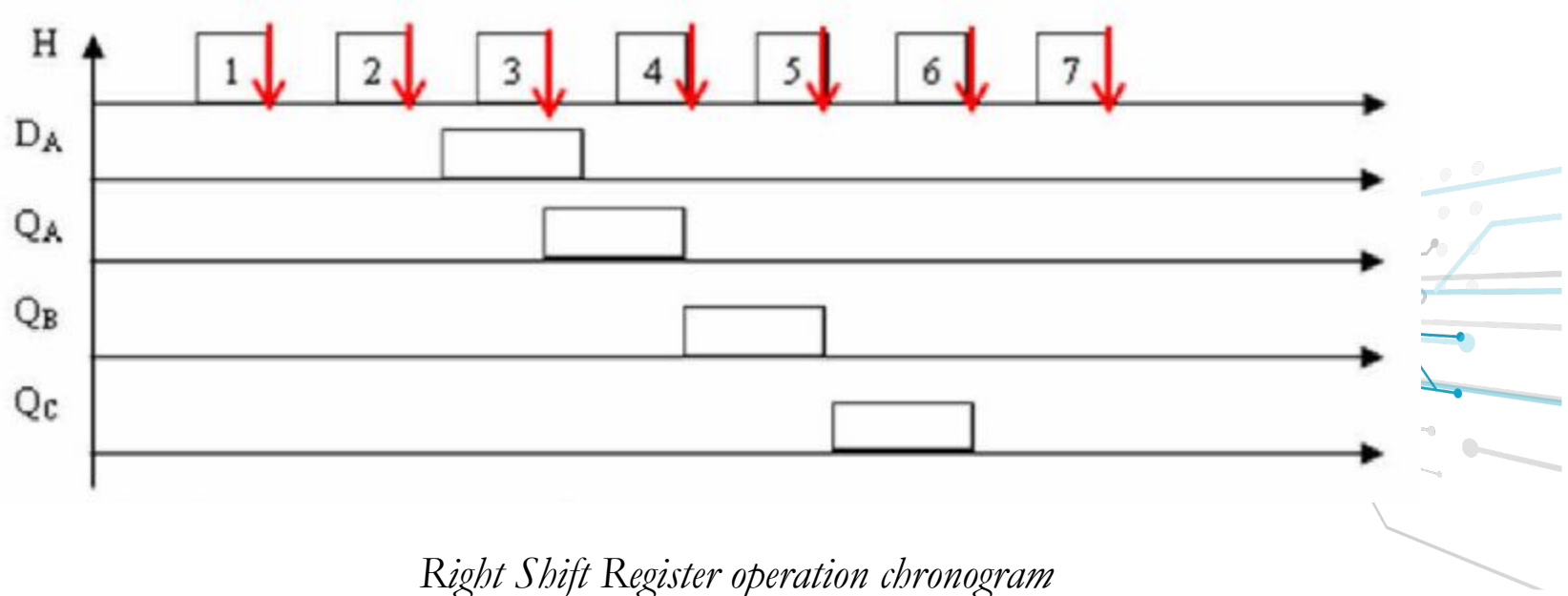
- Illustrate the operation of this register in a table then draw the corresponding chronogram

Solution:

Clock: N pulse	1	2	3	4	5	6	7
Input D_A	0	0	1	0	0	0	0
Q_A	0	0	0	1	0	0	0
Q_B	0	0	0	0	1	0	0
Q_C	0	0	0	0	0	1	0

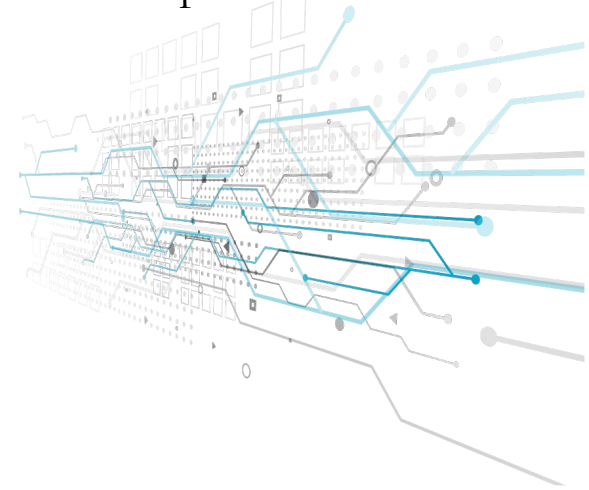
Illustration of shifting 1 in right shift register

Solution:



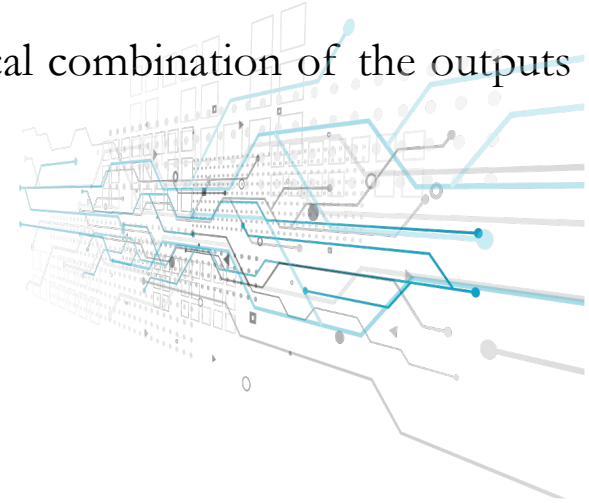
Counters: The essential element of counters, like registers, is latch.

- The state of the counter is defined by the binary number formed with the set of outputs of the latch.
- Counters are classified into two categories according to the mode of operation:
 - Asynchronous or serial counters
 - Synchronous or parallel counters



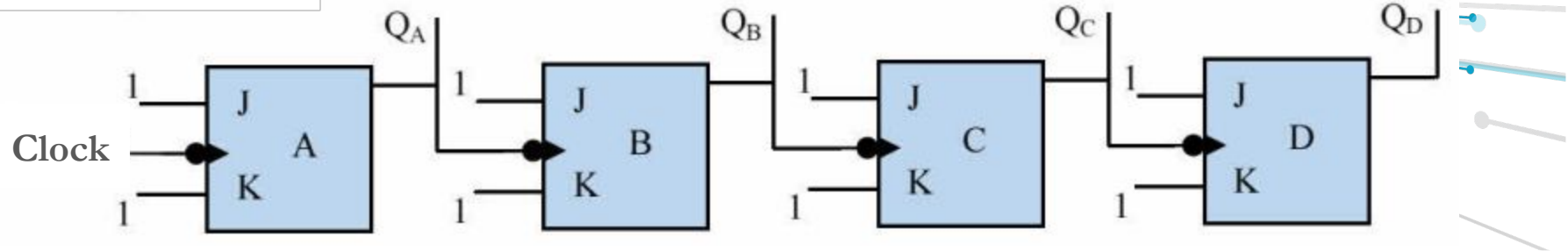
Asynchronous Counters:

- The main characteristic of these counters is the cascade propagation of the state change order of the latches.
- In an asynchronous counter, the clock triggers latch B1, whose output serves as the clock signal for latch B2.
- More generally, the signal of latch B_i is derived from a logical combination of the outputs of latch B_j (where $j \leq i$).



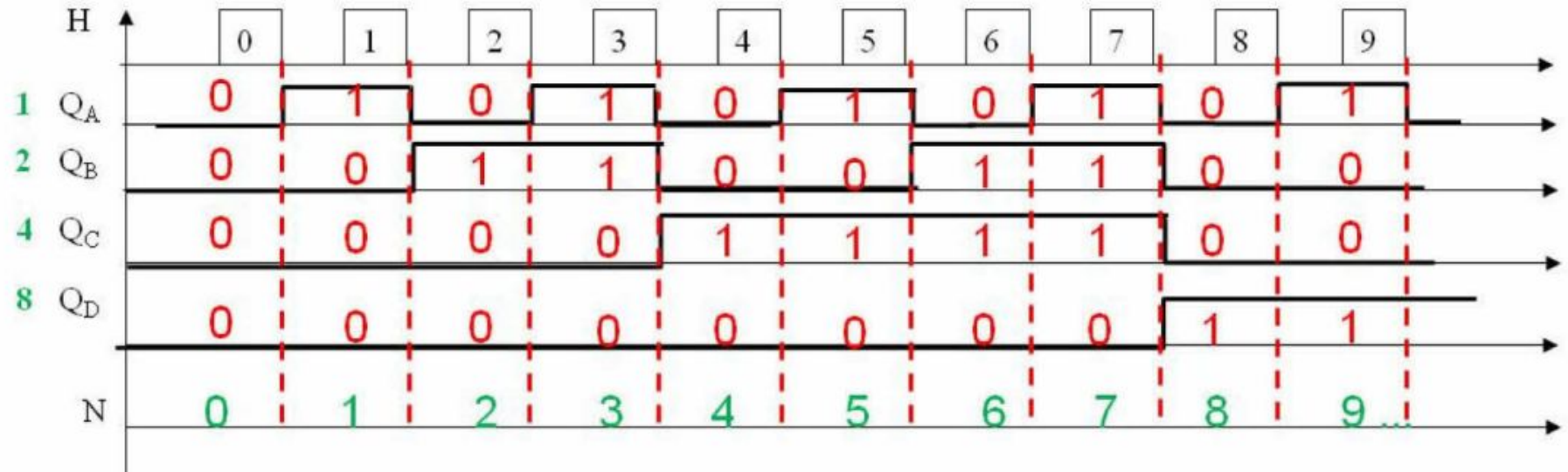
Asynchronous Counters:

- **Binary counters:** This is the simplest asynchronous counter: the outputs of the latches that compose it evolve with the clock in a manner to represent the increasing sequence of numbers expressed in base 2.
- **Example:** Consider the binary counter based on JK latches as follows:
we set $J=K=1$ so that the latch operate as a 'T' latch, thus functioning as an asynchronous counter.



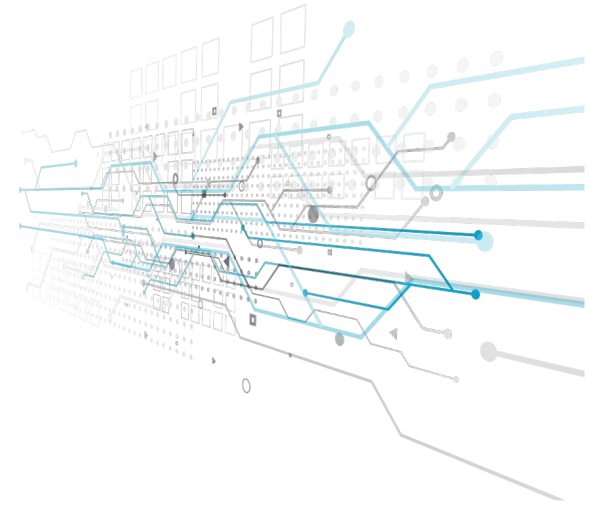
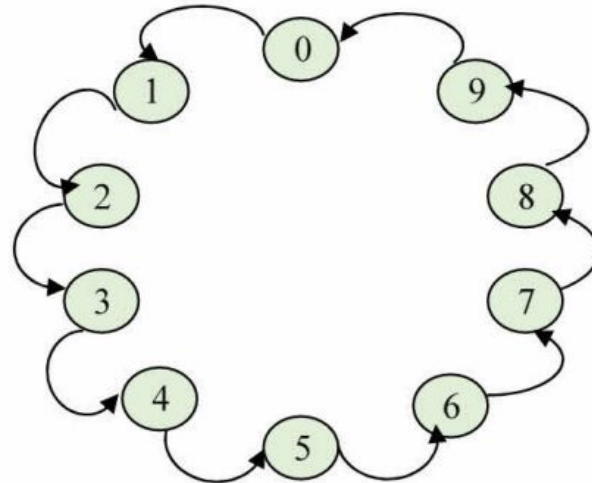
Asynchronous Counters:

- **Example:** Consider the binary counter based on JK latches as follows:
 we set $J=K=1$ so that the latch operate as a T latch, thus functioning as an asynchronous counter.



Modulo N Counters:

- A modulo-N counter is a counter that describes the succession of binary numbers between 0 and $N-1$, meaning the sequence of digits in an N -base translated into binary.
- For example, a modulo-10 counter ($N=10$) will count from 0 to 9 before resetting to 0.
- A modulo-N counter includes N clock pulse.



Modulo N Counters:

Example:

We want to design a modulo-10 counter, meaning it counts 10 pulses and loops (from 0 to 9, then restarts counting when it reaches 9)

