Computers Fundamentals

Subject 4. Sequencial Circuits

Aims and objectives

- The study of basic sequential circuits
- The concept of timing diagram
- Basic operation of bistables: flip-flops and latches

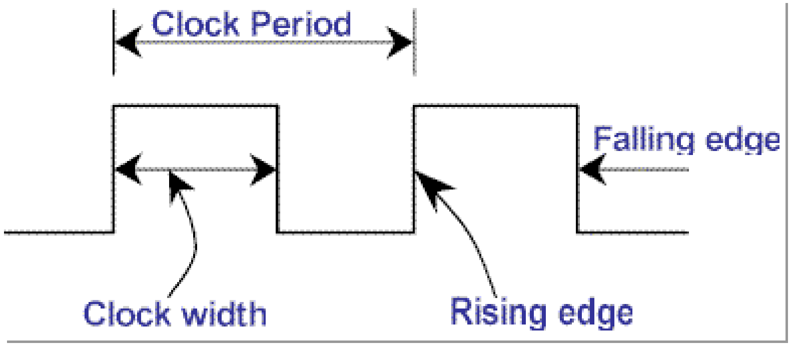
Outline

1. Introduction
   - Sequential circuits definition
   - Clock
   - timing diagram
   - logical symbols
2. Bistables
   - Latches (asynchronous) and Flip-Flops (synchronous)
   - D flip-flop
   - J-K Flip-flop
   - Flip-flops with asynchronous inputs
   - T Flip-flop

3. Basic sequential blocks
   - Storage registers
   - Memory Bank
   - Shift registers
   - Counters

Introduction

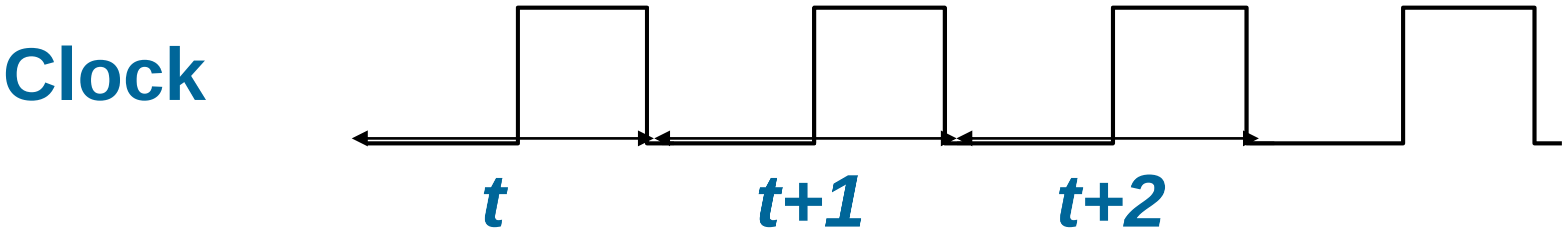
- **Sequential circuits** are divided into two main types:
  - **Synchronous** and **asynchronous (**Their classification depends on the timing of their signals)
- ***Synchronous sequential circuits*** change their states and output values at discrete instants of time, which are specified by the rising and falling edge of a free-running **clock signal**



- In ***Asynchronous sequential circuits,*** the transition from one state to another is initiated by the change in the primary inputs; there is no external synchronization.

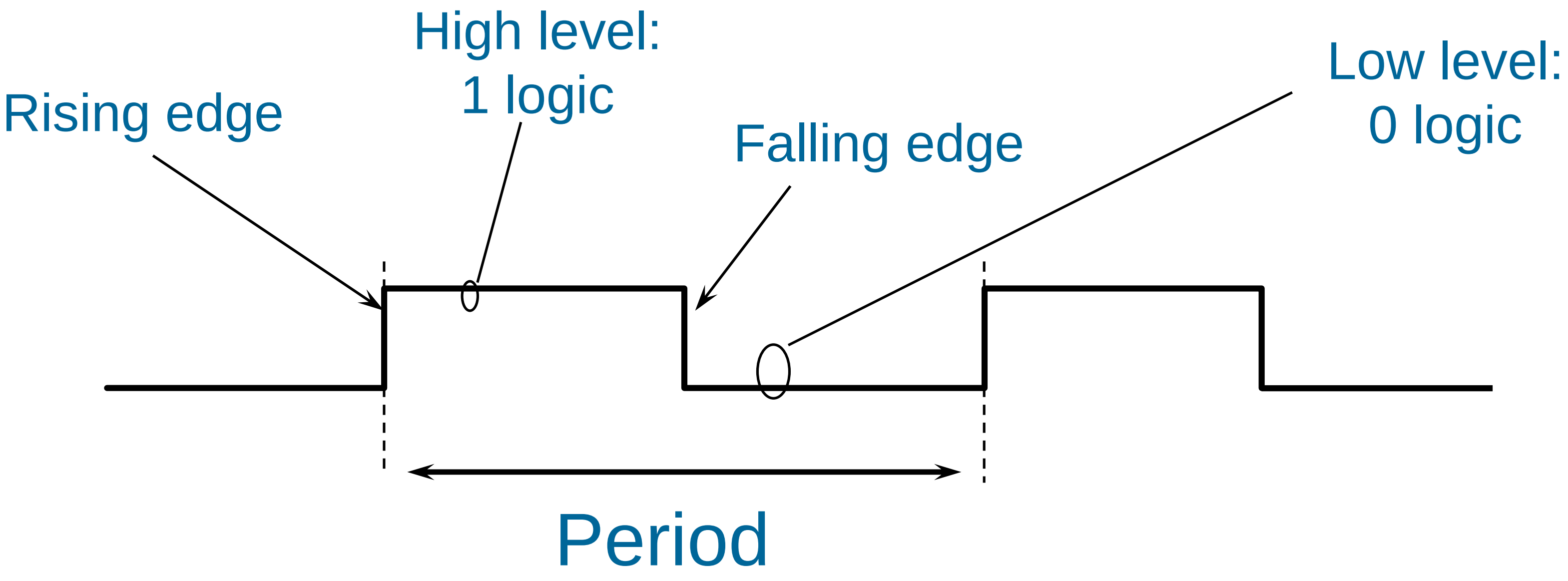
Introduction

- Sequential Circuits:
  - The circuit outputs at the current instant S(t) depends not only on the present value of the circuit inputs E(t), but also on its "memory" or "stored state" Q(t)
  - Are formed by
    - A combinational block
    - A block of memory elements to store the state Q
    - A clock input used to change the internal state of the circuit



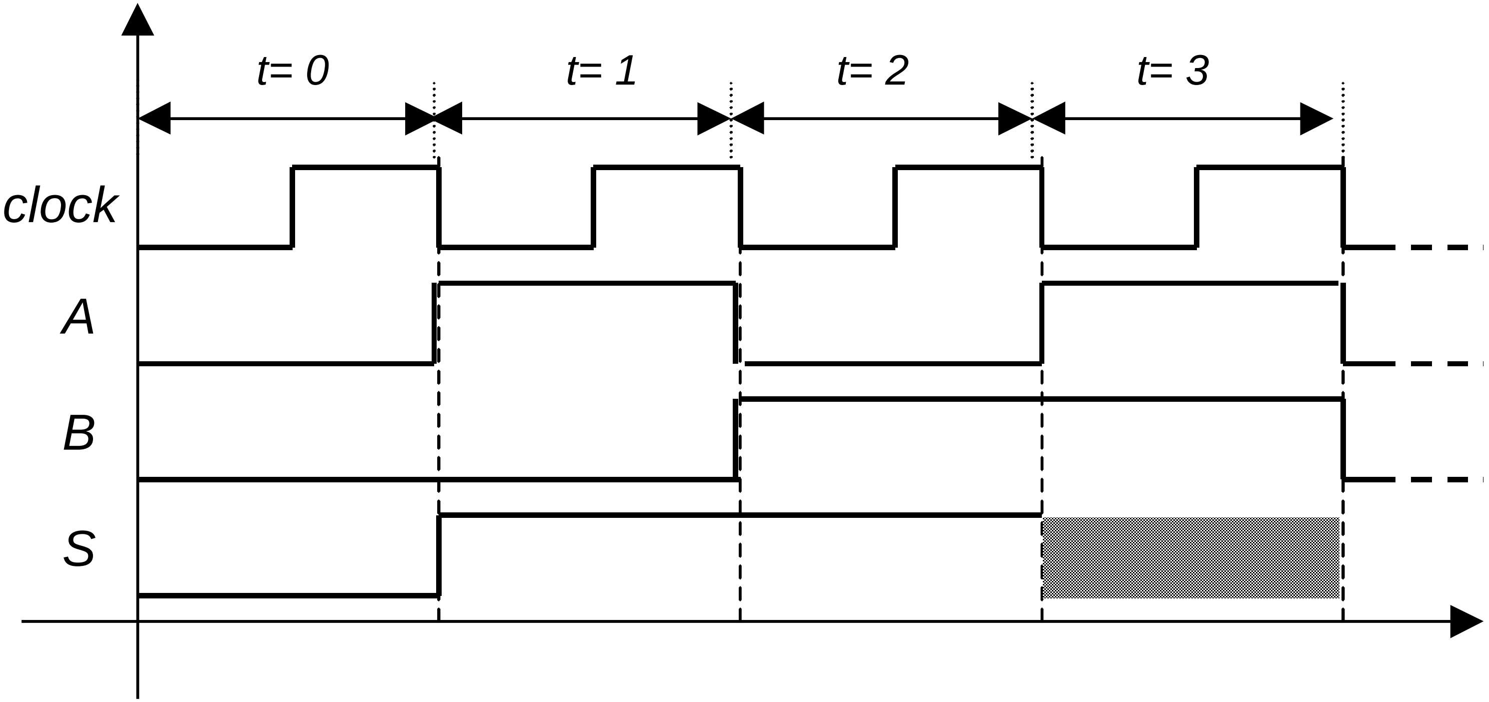
Introduction

- A bistable circuit has two stable states 'high' and 'low'. It can be switched from one stable state to another other stable by applying a signal–level or a trigger pulse.
- Clock signal is used to synchronize the time at which bistable circuits switch from one state to other (memory-elements store the state of the circuit)



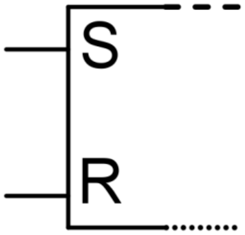
Introduction

- Timing diagram: Representation of the temporal evolution of the inputs and outputs of a circuit.
  - The undefined-value is shown shaded

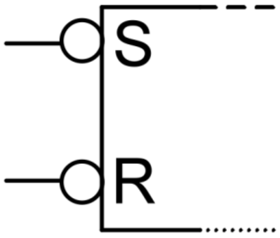


Introduction

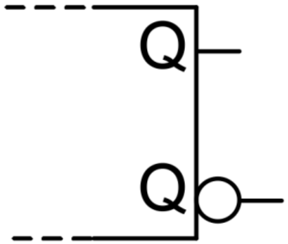
- Logical Symbols
  - Inputs y Outputs



Active high inputs

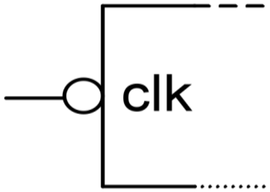


Low level active inputs

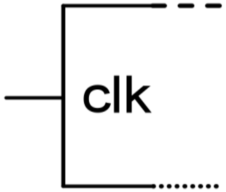


Outputs Q y /Q

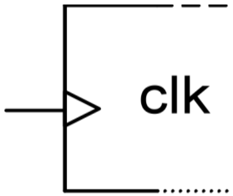
  - Clock



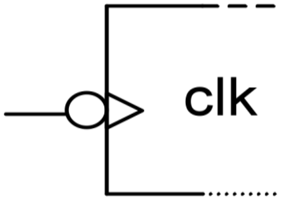
Active low



Active high



rising edge triggered

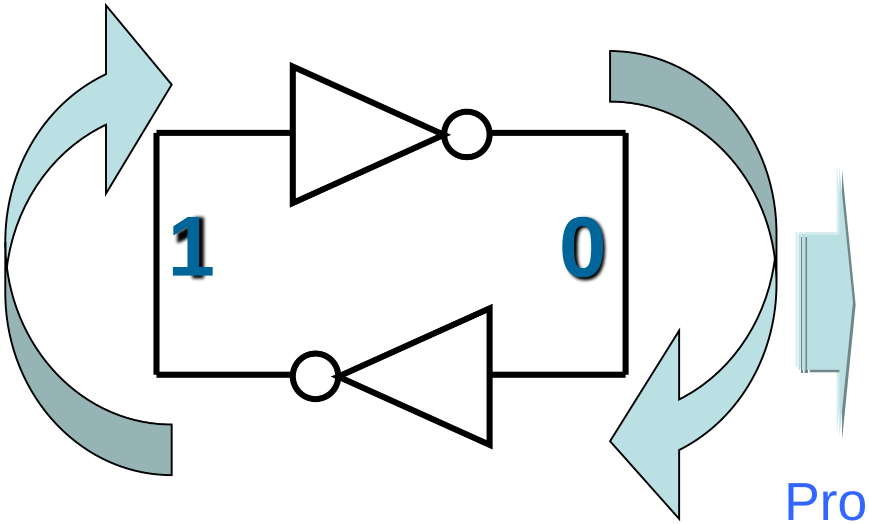


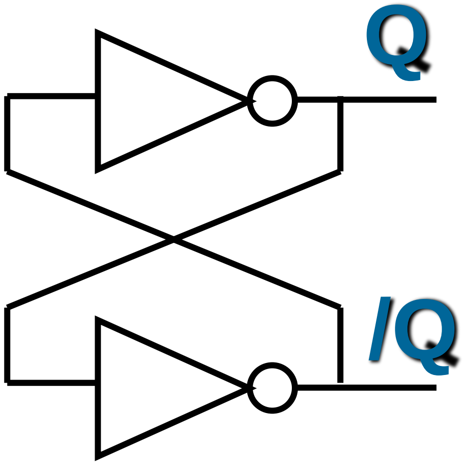
triggered by falling edge

Bistables

- Bistable: Sequential circuit with two stable states (0 and 1). It can be switched from one stable state to another other stable by applying a signal–level or a trigger pulse.

Doing that the information re-circulates indefinitely allowing us to store a bit in a circuit

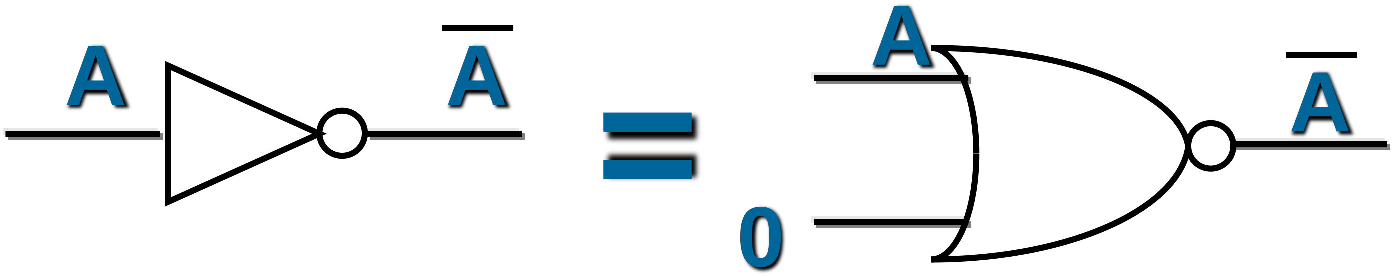


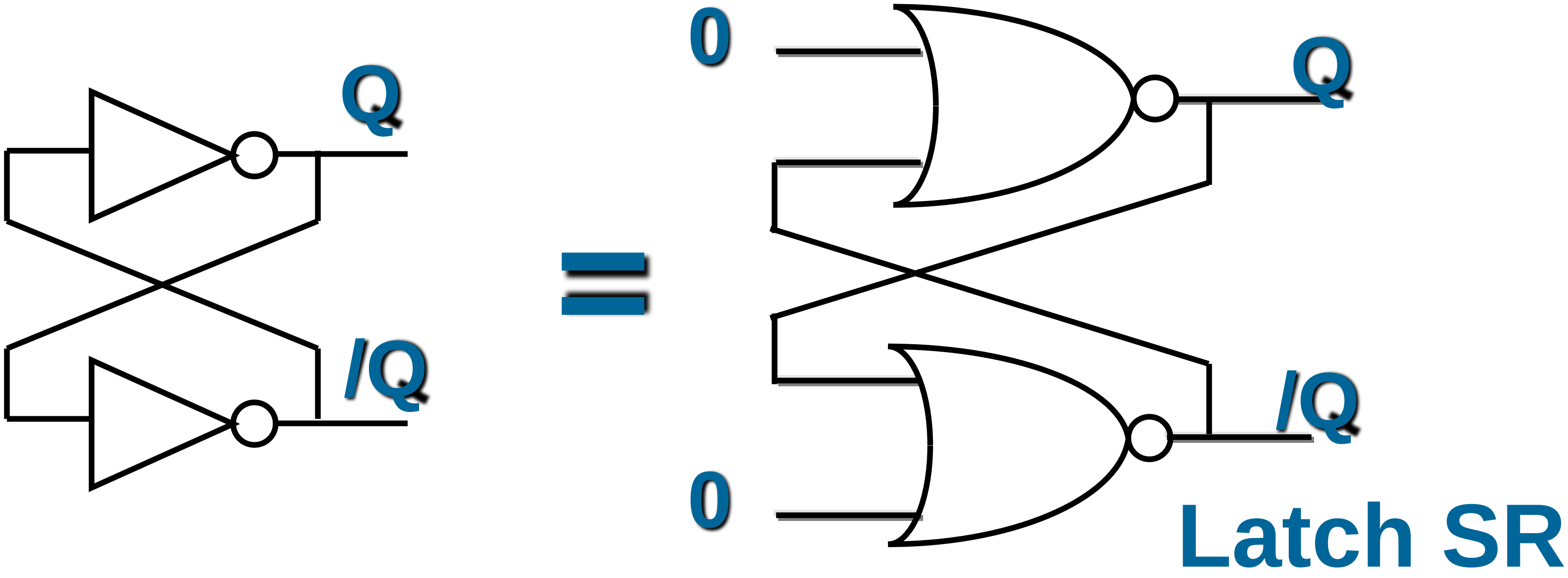


Problem: We can not change the value stored!

Asynchronous bistables: Latches

How to change the state of the bistable?





Latch SR

Latch S-R

- Analysis: $Q(t+1) = \overline{R + /Q(t)}$ y $/Q(t+1) = \overline{S + Q(t)}$

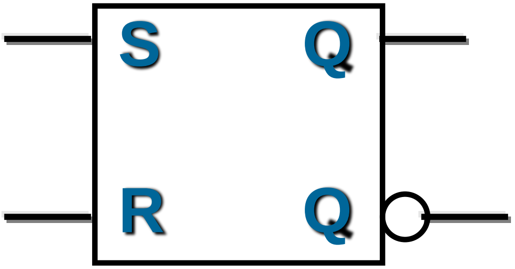
	S	R	Q(t)	/Q(t)	Q(t+1)	/Q(t+1)	
0	0	0	0	0	1	1	→ **3 (Oscillate)**
1	0	0	0	1	0	1	→ **1 (Stable)**
2	0	0	1	0	1	0	→ **2 (Stable)**
3	0	0	1	1	0	0	→ **0 (Oscillate)**
4	0	1	0	0	0	1	→ **5**
5	0	1	0	1	0	1	→ **5 (Stable)**
6	0	1	1	0	0	0	→ **4**
7	0	1	1	1	0	0	→ **4**
8	1	0	0	0	1	0	→ **10**
9	1	0	0	1	0	0	→ **8**
10	1	0	1	0	1	0	→ **10 (Stable)**
11	1	0	1	1	0	0	→ **8**
12	1	1	0	0	0	0	→ **12 (Stable)**
13	1	1	0	1	0	0	→ **12**
14	1	1	1	0	0	0	→ **12**
15	1	1	1	1	0	0	→ **12**

Q(t+1)	/Q(t+1)
Q(t)	/Q(t)
0	1
1	0
0*	0*

Asynchronous Bistables: SR-NOR-Latch

Latch SR with NOR

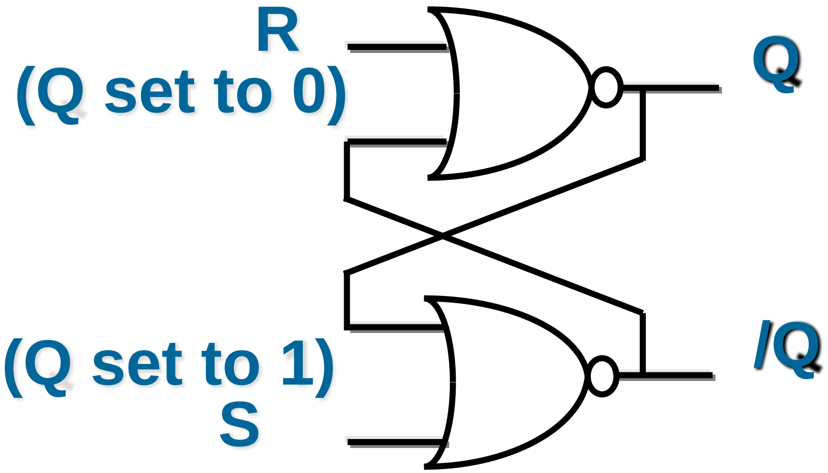
Symbol



Truth table
* =undesirable situation

S	R	Q(t)	Q(t+1)	/Q(t+1)
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0*	0*
1	1	1	0*	0*

Logic Diagram

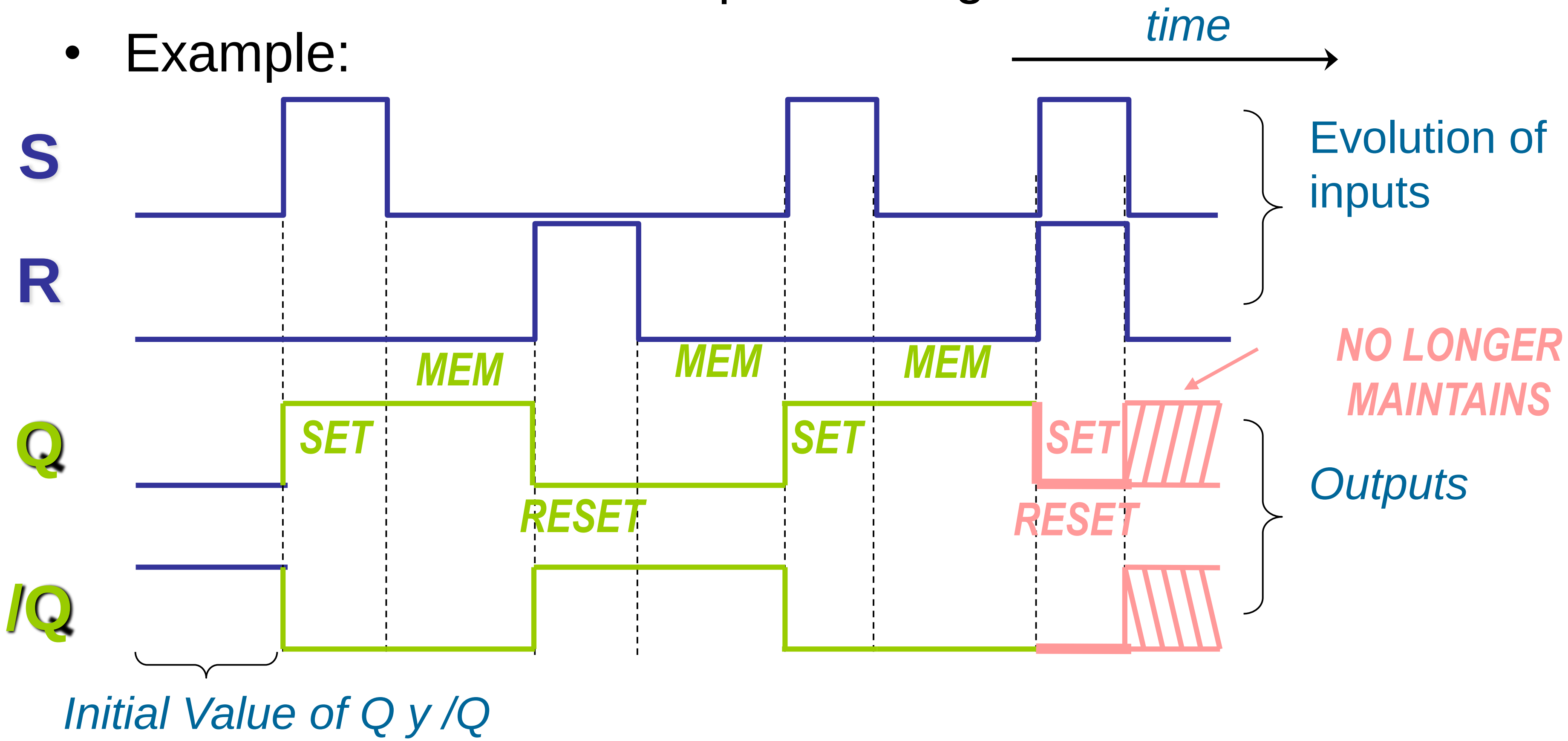


Reduced truth table
* =undesirable situation

S	R	Q(t+1)	/Q(t+1)
0	0	Q(t)	/Q(t)
0	1	0	1
1	0	1	0
1	1	0*	0*

Asynchronous Bistables

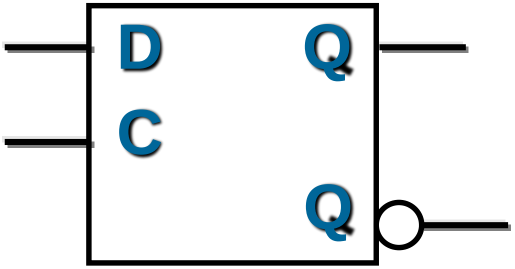
- The timing diagram is used to determine the temporal evolution of the state when their inputs change
- Example:



Asynchronous Bistables: D Latch

- Used to implement memory elements, whose purpose is to store one bit of information

Symbol



Truth table

C	D	Q(t+1)	/Q(t+1)
0	X	Q(t)	/Q(t)
1	1	1	0
1	0	0	1

Asynchronous Bistables: D Latch

- We can build a D Latch from a SR Latch.

C	D	Q(t+1)	/Q(t+1)	R	S
0	X	Q(t)	/Q(t)	0	0
1	1	1	0	0	1
1	0	0	1	1	0

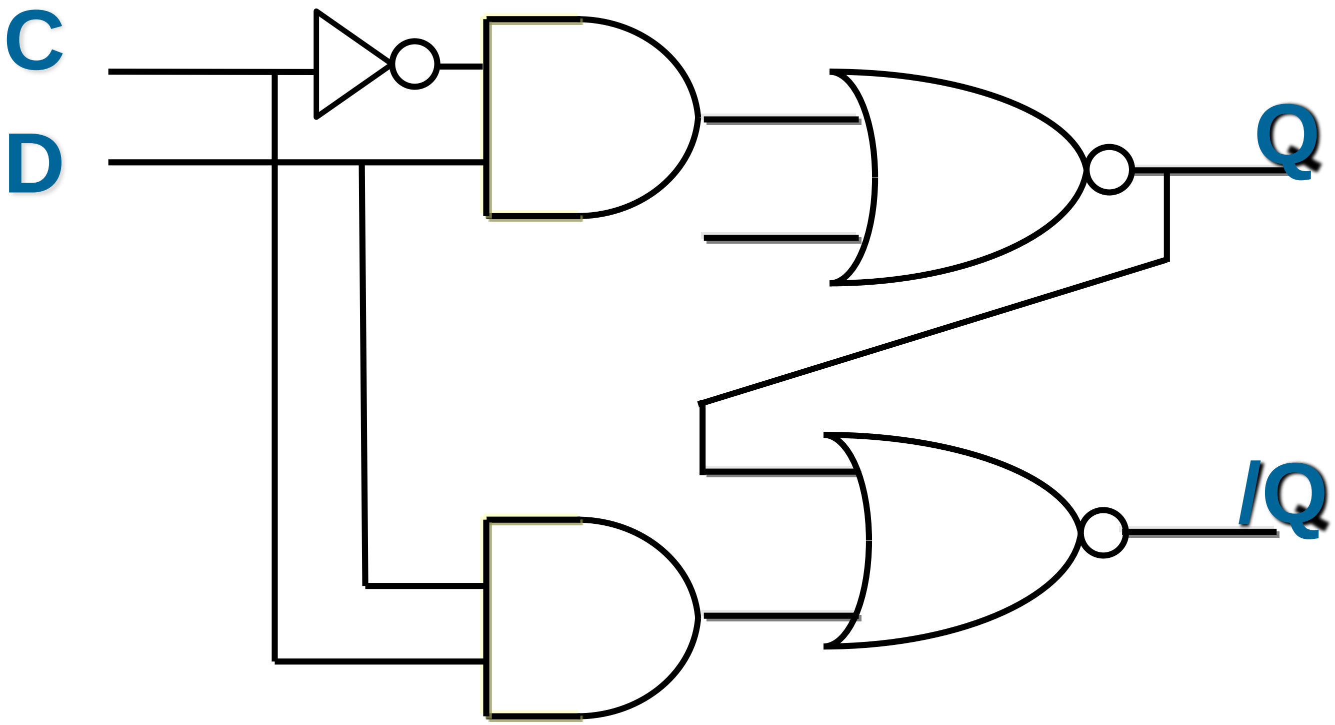
- The state is maintained when C = 0 (R = S = 0)
- Q(t+1)=1 if C=1 and D=1 (R=0 y S=1)
- Q(t+1)=0 if C=1 and D=0 (R=1 y S=0)

$R=CD$

$S=C\overline{D}$

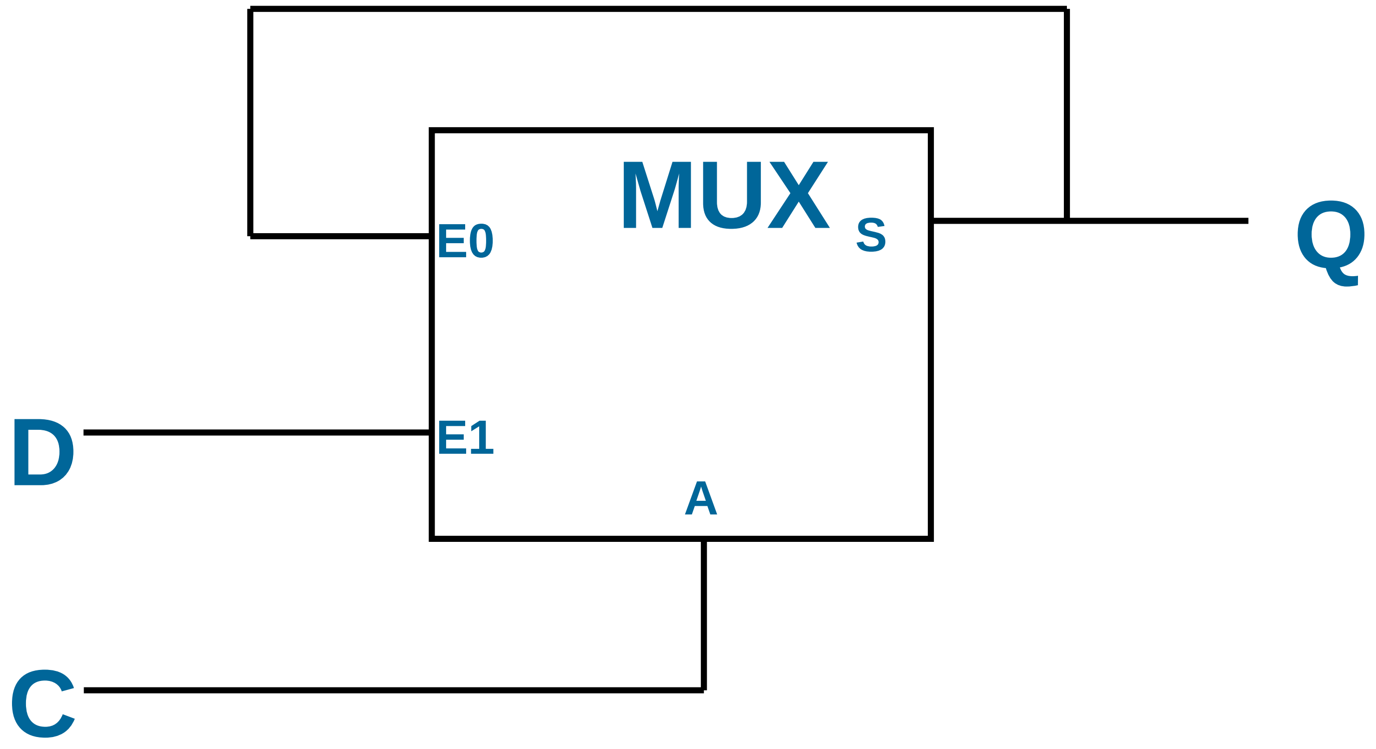
Asynchronous Bistables: D Latch

- One possible implementation of the D Latch:



Asynchronous Bistables: D Latch

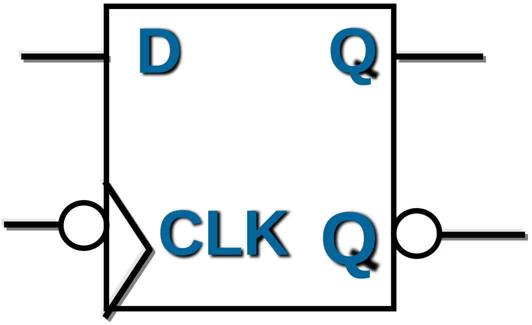
- Another possible implementation of the D Latch:



Synchronous Bistables: D Flip-Flop

- A pair of synchronous-bistables can be used to implement an edge-triggered bistable
- The configuration is known as master / slave

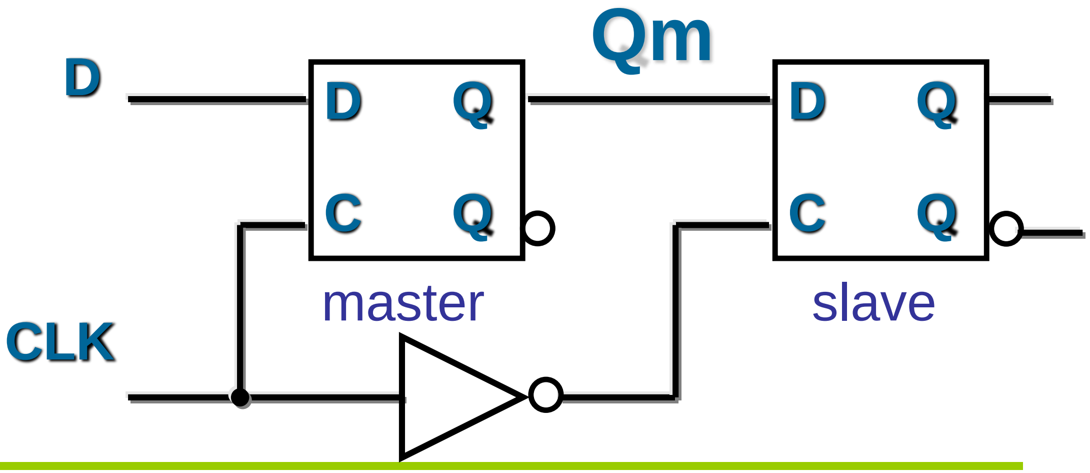
Symbol



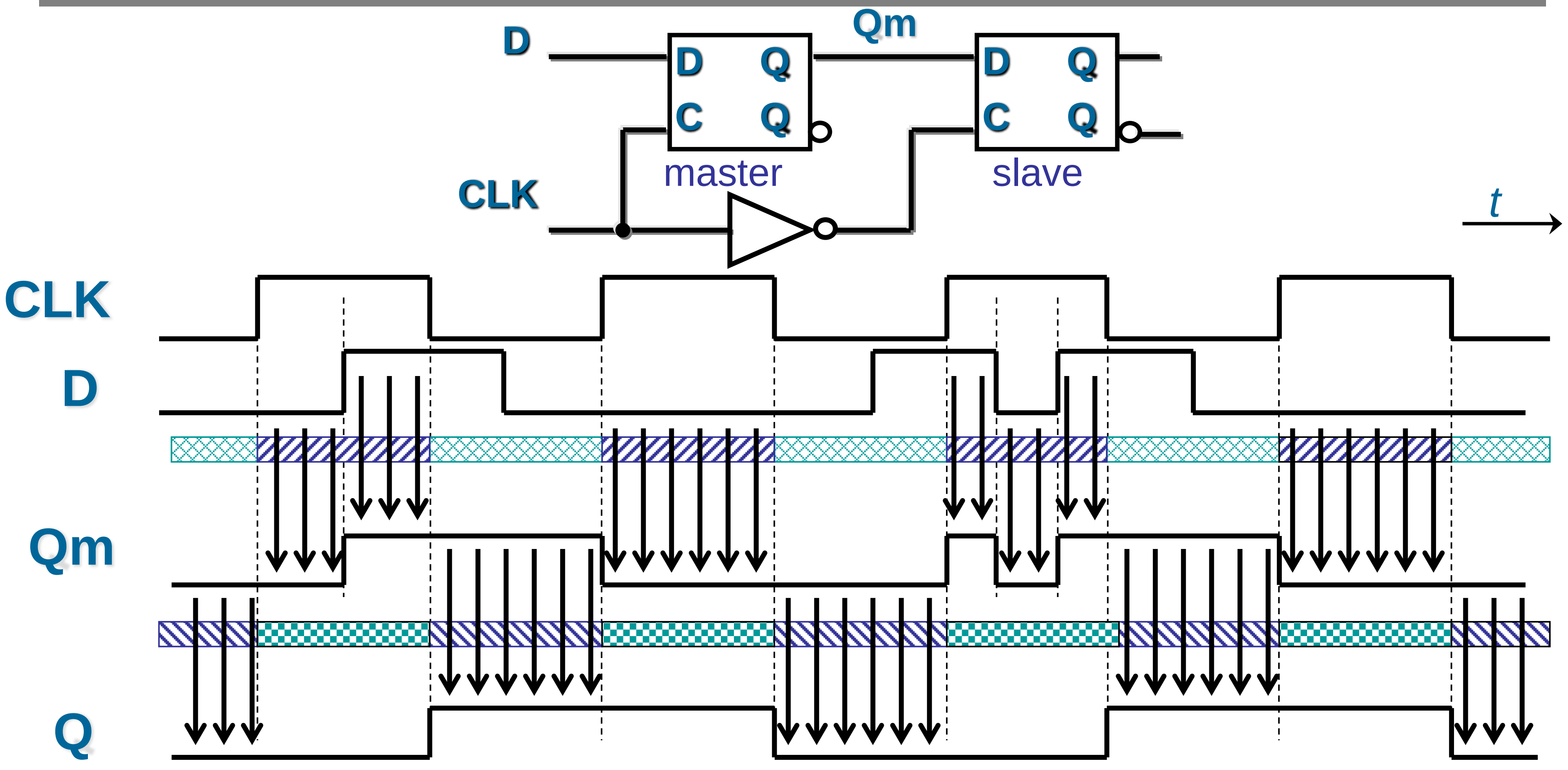
Truth table

CLK	D	Q(t+1)	/Q(t+1)
0	X	Q(t)	/Q(t)
1	X	Q(t)	/Q(t)
↓	1	1	0
↓	0	0	1

Master/Salve Configuration



D Flip-Flop falling-edge triggered



▨ *Master enabled Qm=D*

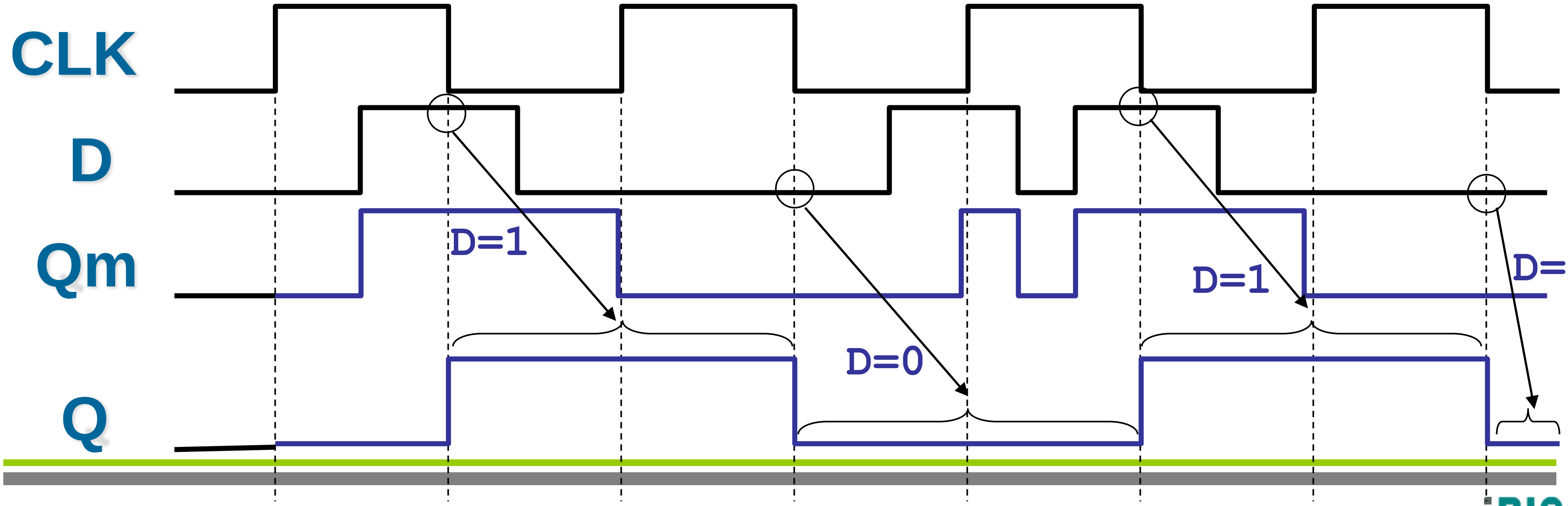
▨ *Slave enabled, Q=Qm*

▩ *Master disabled*

▩ *Slave disabled*

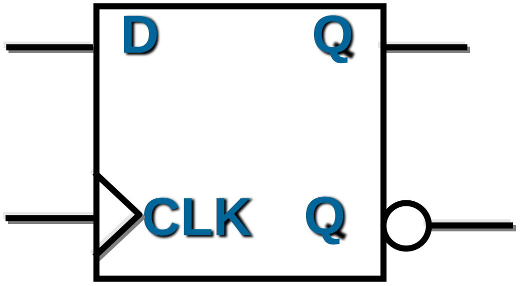
D Flip-Flop falling-edge triggered

- Note:
  - Qm Signal changes when CLK = 1, following the evolution of the input D.
  - Q output only changes with the falling-edges of CLK-signal

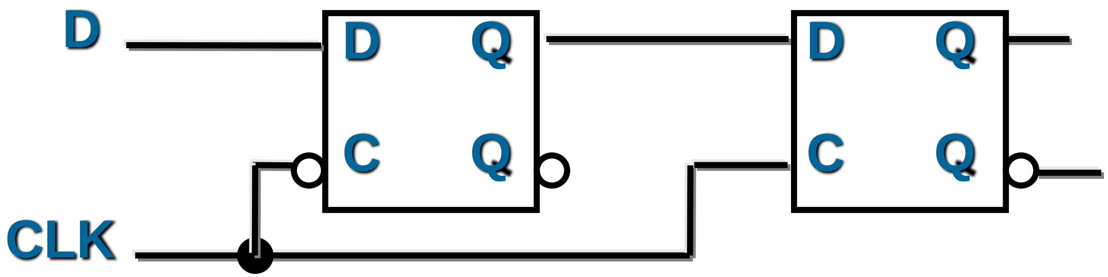


D Flip-Flop rising-edge triggered

Symbol



Implementation

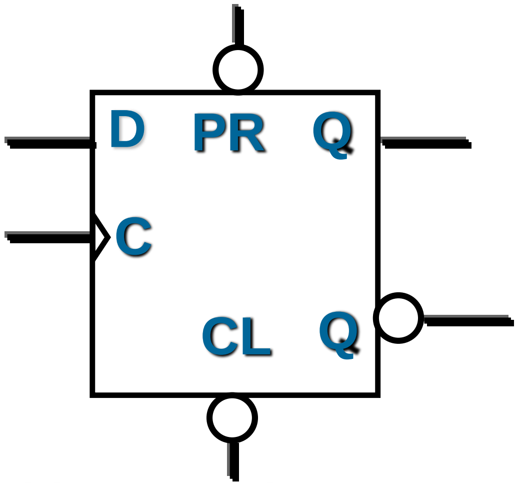


Truth table

CLK	D	Q(t+1)	/Q(t+1)
0	X	Q(t)	/Q(t)
1	X	Q(t)	/Q(t)
↑	1	1	0
↑	0	0	1

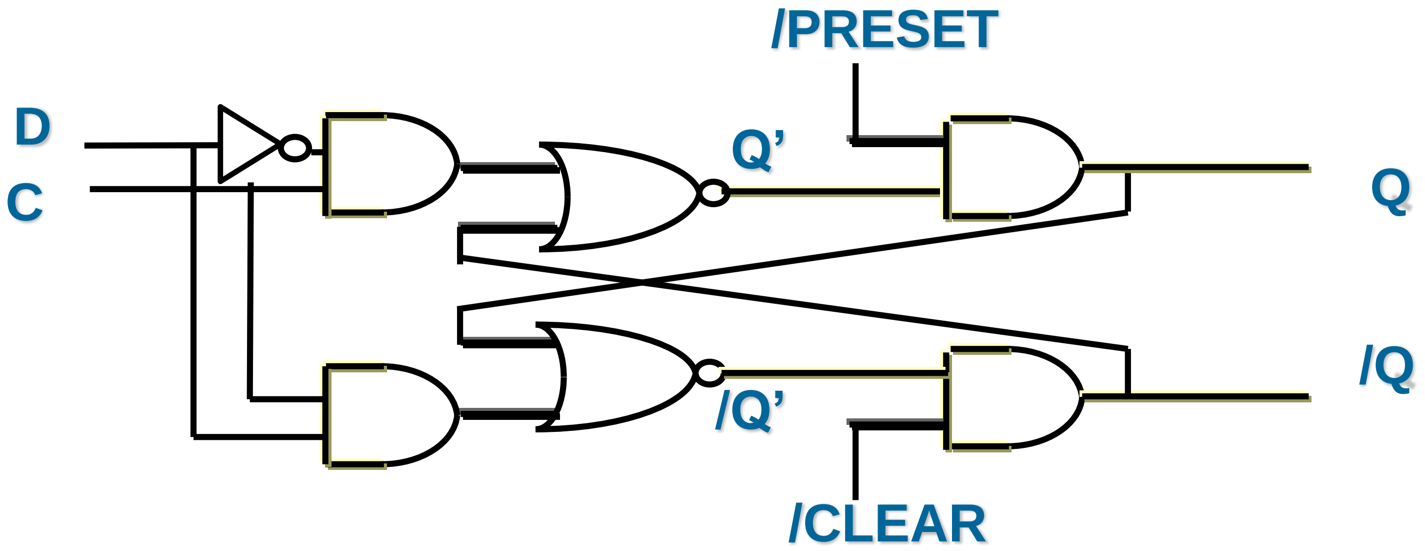
D Flip-Flop

- D Flip-Flop with asynchronous inputs for clear and reset operations
  - /CLEAR
  - /PRESET
- Asynchronous inputs have more priority than clock–input



Símbolo lógico

D LATCH



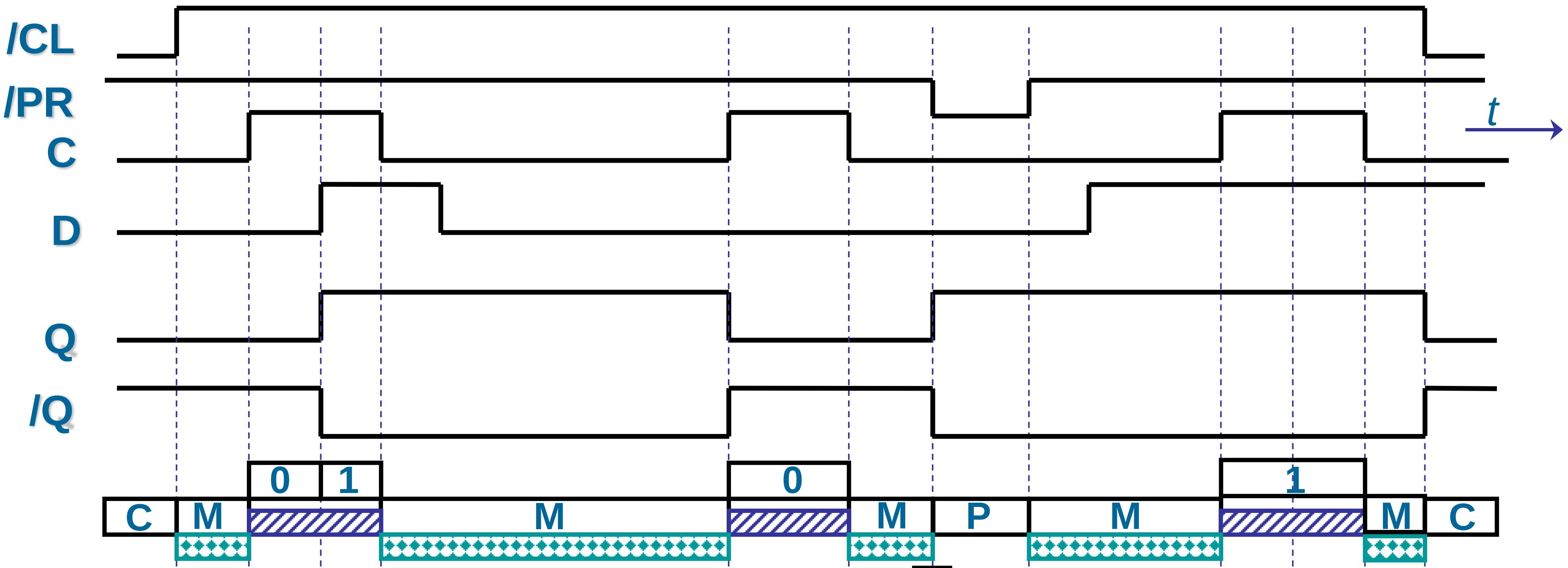
- /CLEAR=/PRESET=1 ⇒ Q = Q', /Q = /Q'
- /CLEAR=0,/PRESET=1 ⇒ Q = 0, /Q' = 1, /Q = 1
- /CLEAR=1,/PRESET=0 ⇒ /Q = 0, Q' = 1, Q = 1
- /CLEAR=0,/PRESET=0 ⇒ Q=/Q=0*, Q'=/Q'=1*

D Latch with asynchronous inputs

- Truth table

/PR	/CL	C	D	Q(t+1)	/Q(t+1)
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	0*	0*
1	1	1	1	1	0
1	1	1	0	0	1
1	1	0	X	Q(t)	/Q(t)

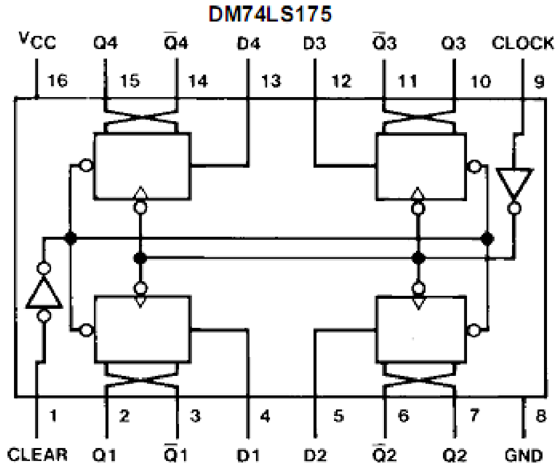
D Latch



*	*Not desired*			M	*Memory*
▨	*Enabled*	1	*Set*	P	*Preset: Asynchronous Set*
▩	*Disabled*	0	*Reset*	C	*Clear: Asynchronous Reset*

Commertial IC 74175

- Integrated circuit 74175
- 4 flip-flops rising-edge triggered
  with asynchronous inputs for set and reset

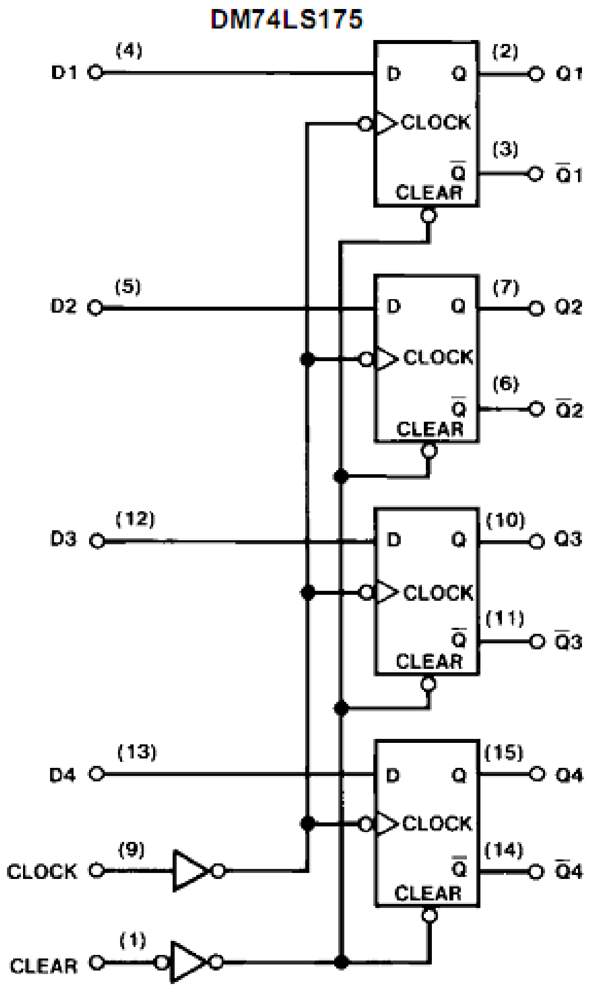


Function Table

(Each Flip-Flop)

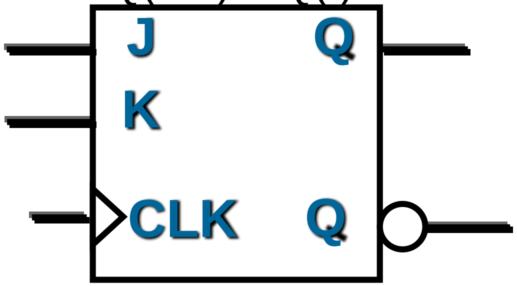
Inputs			Outputs	
Clear	Clock	D	Q	$\overline{Q}$ †
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\overline{Q}_0$

H = HIGH Level (steady state)
L = LOW Level (steady state)
X = Don't Care
↑ = Transition from LOW-to-HIGH level
Q_0 = The level of Q before the indicated steady-state input conditions were established.



JK Flip-flop

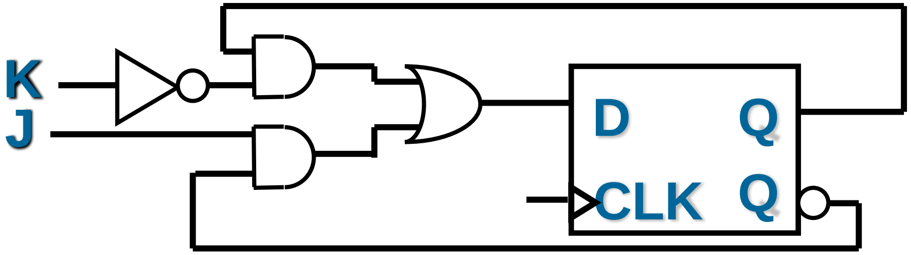
- The SR latch has problems when inputs S and R are activated at the same time
- Solution: design a flip-flop to avoid that problem. The behavior should be as follows
  - Input J (Set operation); when activated and there is a rising edge on CLK Q(t+1) = 1
  - Input K (Reset opertation) ; When activated and there is a rising edge on CLK Q(t+1) = 0
  - When inputs J and K are not activated the state must be maintained: Q(t+1) = Q(t)
  - When inputs J and K are activated and there is a rising edge on CLK (Toggle Op.): Q(t+1) = /Q(t)



Symbol

CLK	J	K	Q(t+1)	/Q(t+1)
0	X	X	Q(t)	/Q(t)
1	X	X	Q(t)	/Q(t)
↑	0	0	Q(t)	/Q(t)
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	/Q(t)	Q(t)

Truth table

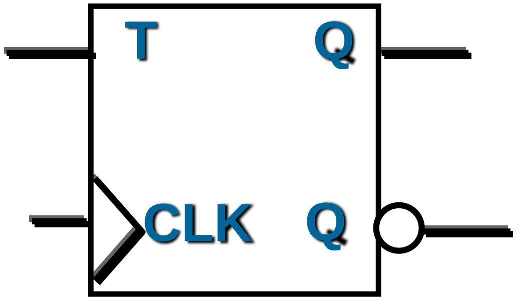


Logic diagram

T Flip-flop

- It has two inputs: T-input (*toggle*) and clock-input
- Memory (if T=0) or Toggle (if T=1) each time there is a clock-edge (in this case rising-edege)
- It is not commercialized, but it can be implemented from a D-Flip-Flop or from a JK Flip-flop

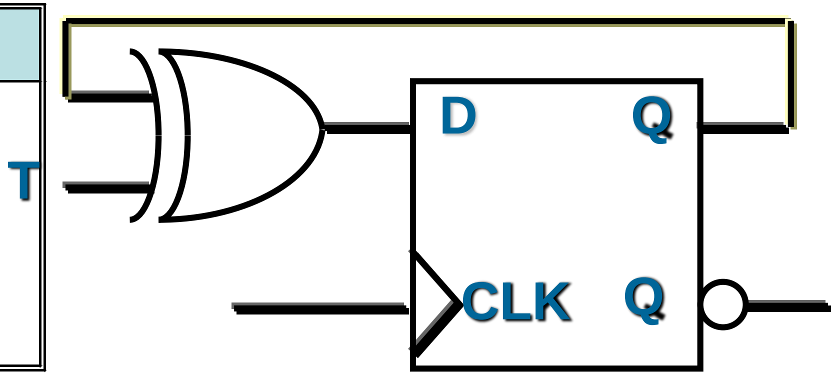
Symbol



Thruth table

T	Q(t)	D
0	0	0
0	1	1
1	0	1
1	1	0

Logic diagram



It is possible to build a T flip-flop falling-edge triggered in a similar way

Computers Fundamentals

Subject 4. Sequencial Circuits