Lecture
0_7.1

Latches & Flip-Flops



Paolo PRINETTO
Politecnico di Torino (Italy)
University of Illinois at Chicago, IL (USA)

Paolo.Prinetto@polito.it
prinetto@uic.edu

www.testgroup.polito.it
www.comitato-girotondo.org

License Information

This work is licensed under the Creative Commons BY-NC License



To view a copy of the license, visit:
http://creativecommons.org/licenses/by-nc/3.0/legalcode

© Prinetto - 2018

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Rel. 05.06.2018
© Prinetto - 2018

Goal

– The lecture aims at presenting the set of functional sequential blocks that are usually considered to be “elementary” or “basic” at the logic level.

Rel. 05.06.2018

© Prinetto - 2018

Prerequisites

- Track 0_4 - System & Device Modeling

Rel. 05.06.2018
© Prinetto - 2018

Homework

- None

© Prinetto - 2018

Further readings

- Any additional info on the covered topics
- Students interested in making a reference to a text book on the arguments covered in this lecture can refer, for instance, to:
  - M. Morris Mano, C.R.Kime: "*Logic and Computer Design Fundamentals*," 2nd edition updated Prentice Hall, Upple Saddle River, NJ (USA), 2001, (chapter 4, pp. 182-201)

Rel. 05.06.2018
© Prinetto - 2018

Further readings

or to:

- **J. P. Hayes:**
  "*Introduction to Digital Logic Design*,"
  Addison Wesley, Reading, MA (USA), 1994,
  (chapter 6, pp. 390-453)

Rel. 05.06.2018
© Prinetto - 2018

Outline

- Latches vs. Flip-Flops
- Latches
- Flip-Flops
- Scannable Flip-Flops

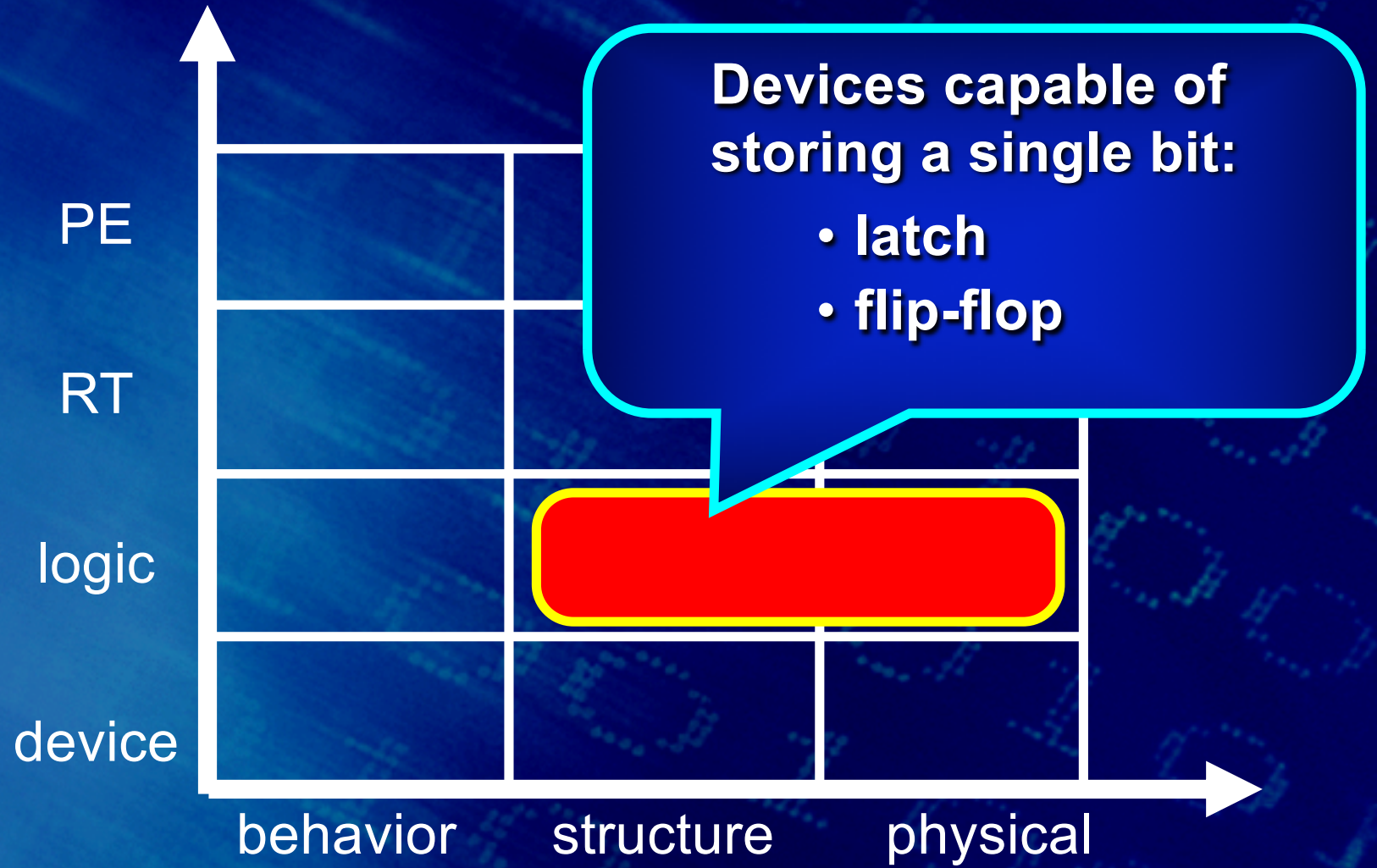
Rel. 05.06.2018
© Prinetto - 2018

Outline

- Latches vs. Flip-Flops
- Latches
- Flip-Flops
- Scannable Flip-Flops

© Prinetto - 2018

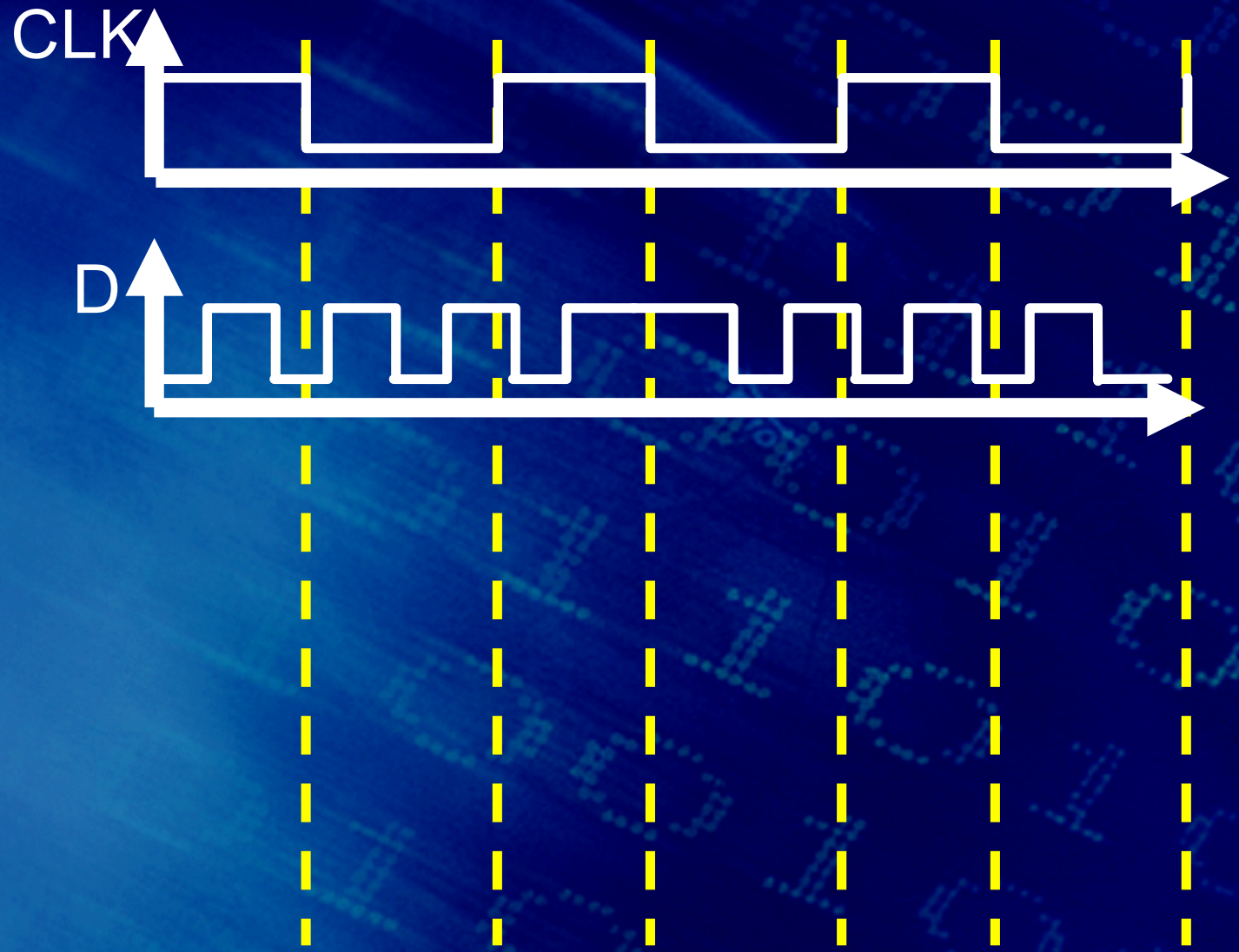
Logic level elementary sequential blocks



Rel. 05.06.2018

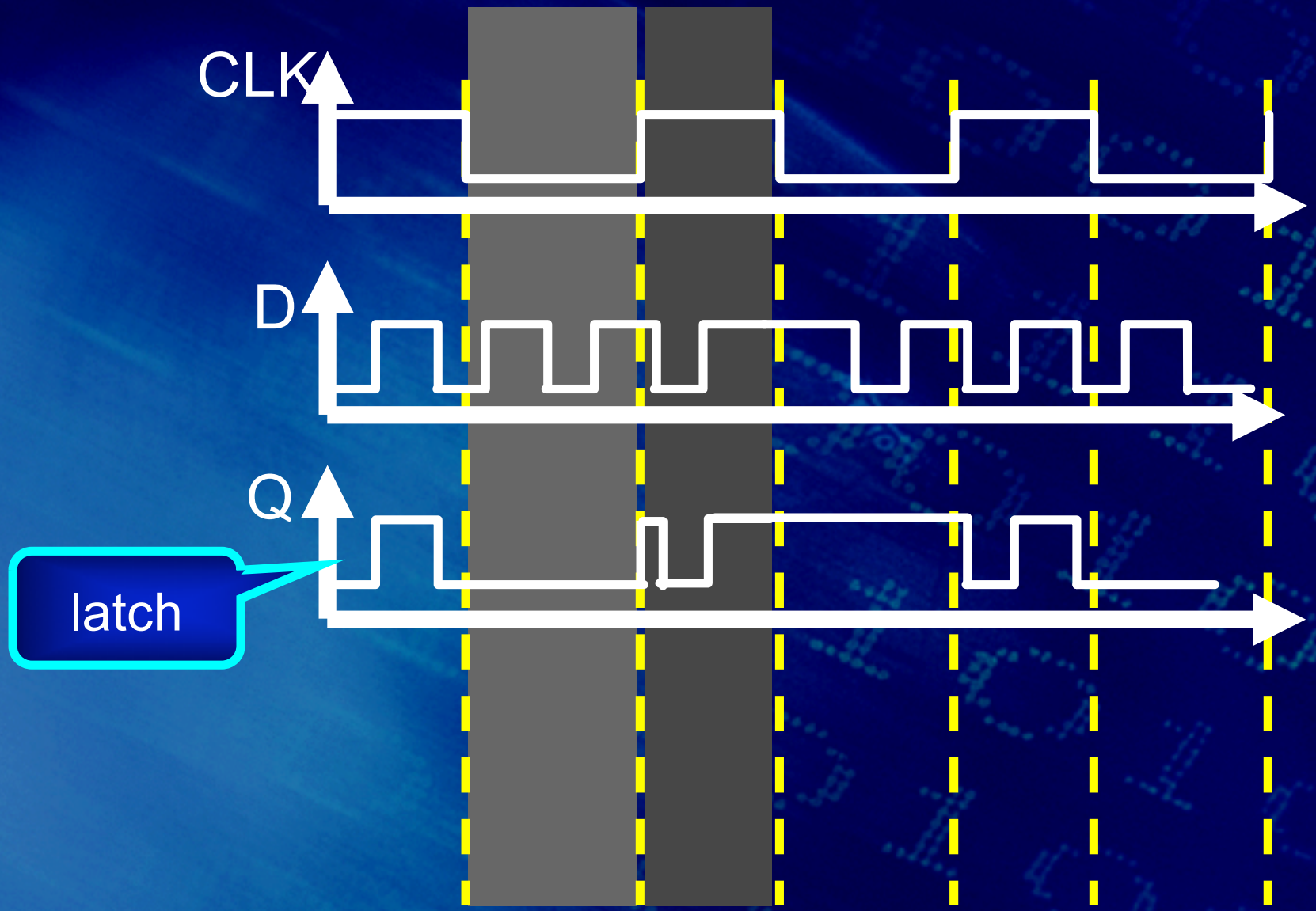
© Prinetto - 2018

Latch vs. Flip-flop



© Prinetto - 2018

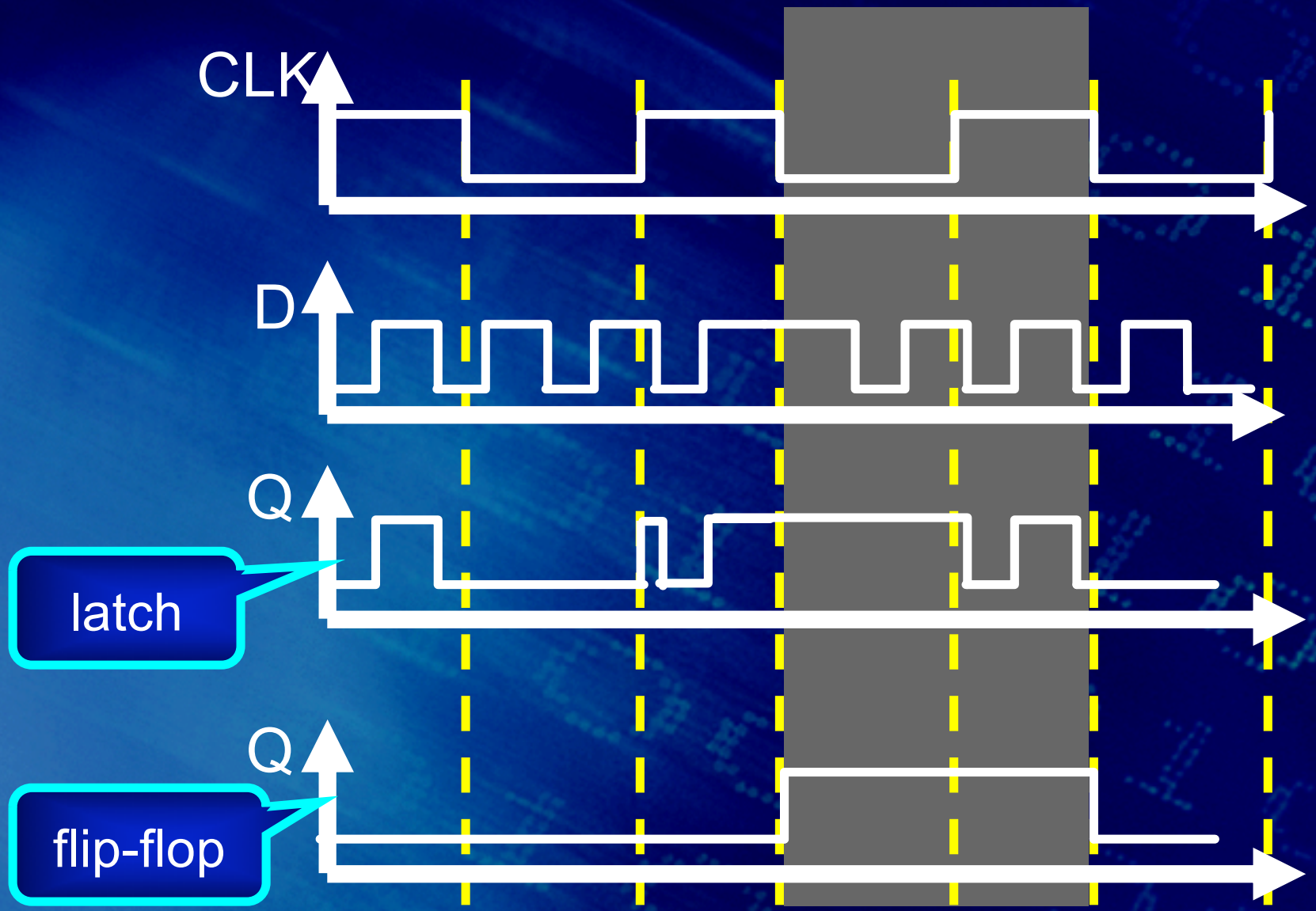
Latch vs. Flip-flop



Rel. 05.06.2018

© Prinetto - 2018

Latch vs. Flip-flop



Rel. 05.06.2018

© Prinetto - 2018

Pro's & Con's

- A latch is a Mealy machine
- A flip-flop is a Moore machine
- Flip-flops improve testability

© Prinetto - 2018

Outline

- Latches vs. Flip-Flops
- Latches
- Flip-Flops
- Scannable Flip-Flops

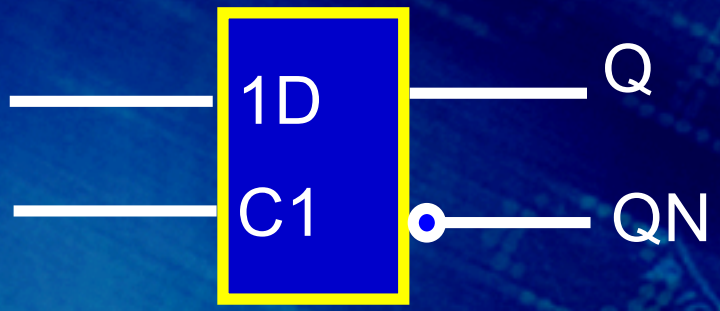
© Prinetto - 2018

Latch types

- **Several latch types exist, including:**
  - ***D latch***
  - ***IBM LSSD D latch***
  - ***SR latch***
  - ***Gated SR latch***

Rel. 05.06.2018
© Prinetto - 2018

D latch

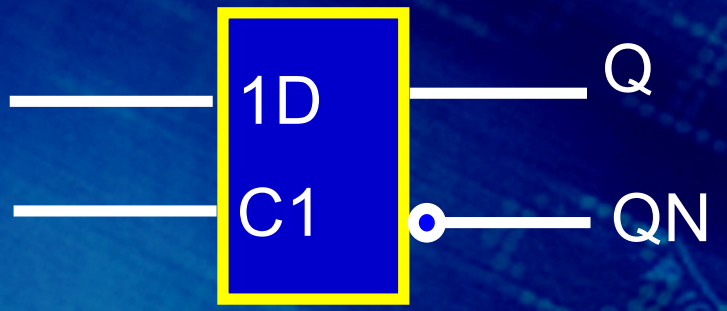


IEEE Symbol

Rel. 05.06.2018

© Prinetto - 2018

D latch



IEEE Symbol

Characteristic Table

C	D	Q	QN
0	-	Q_{-1}	QN_{-1}
1	0	0	1
1	1	1	0

Rel. 05.06.2018

© Prinetto - 2018

D latch

Describes the logical properties of the latch by describing its operations in tabular form



IEEE Symbol

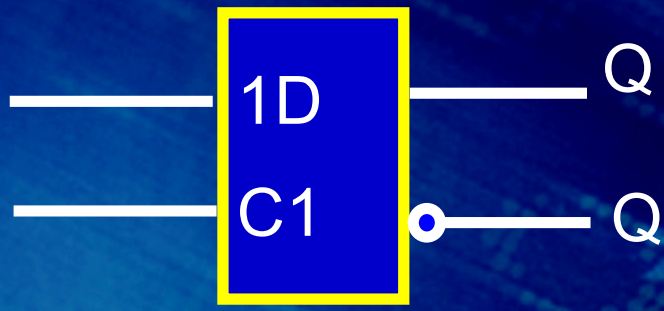
Characteristic Table

C	D	Q	QN
0	-	Q_{-1}	QN_{-1}
1	0	0	1
1	1	1	0

Rel. 05.06.2018

© Prinetto - 2018

D latch



IEEE Symbol

C	D	Q	QN
0	-	Q_{-1}	QN_{-1}
1	0	0	1
1	1	1	0

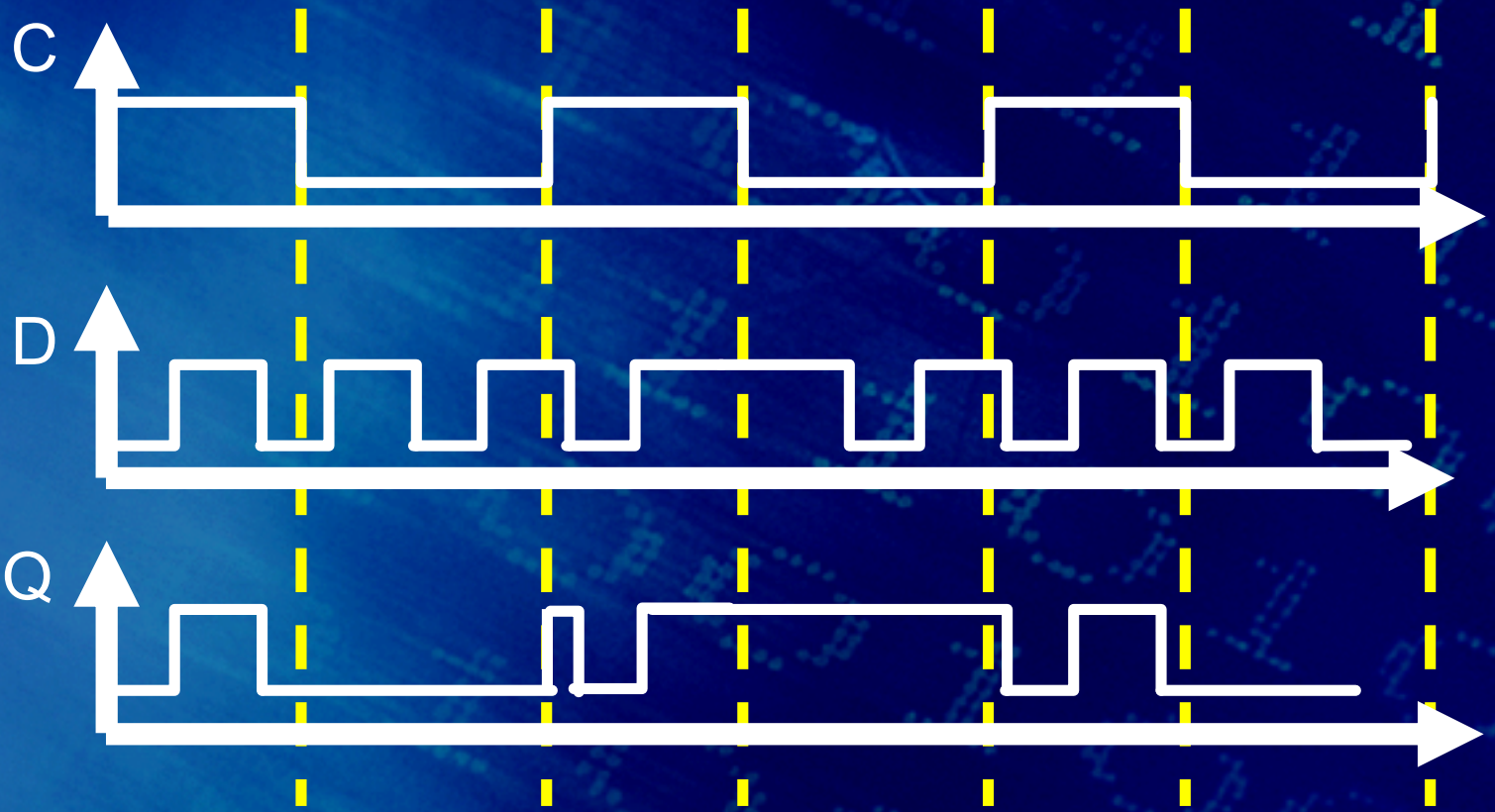
Previous value of Q

Rel. 05.06.2018

© Prinetto - 2018

Waveforms

C	D	Q	QN
0	-	Q_{-1}	QN_{-1}
1	0	0	1
1	1	1	0



Rel. 05.06.2018

© Prinetto - 2018

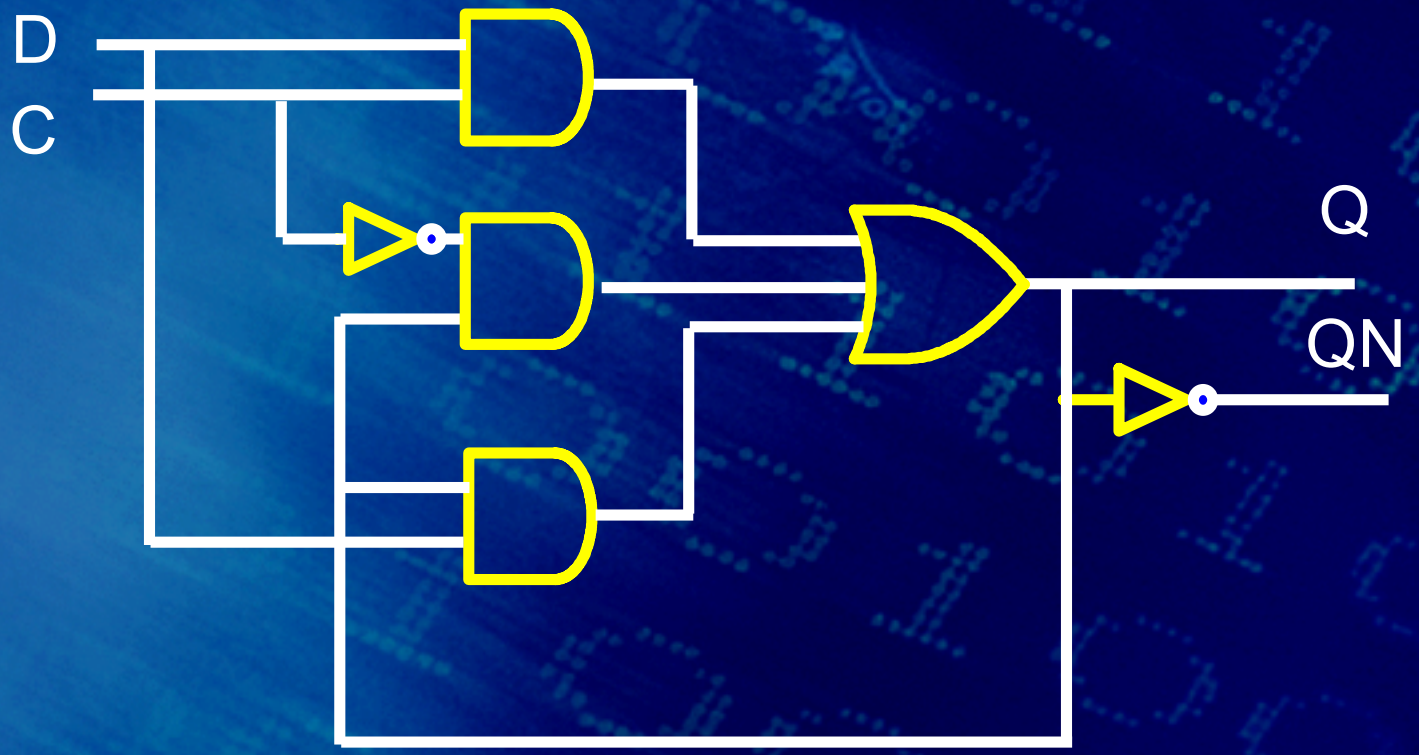
Latch types

- Several latch types exist, including:
  - *D latch*
  - *IBM LSSD D latch*
  - *SR latch*
  - *Gated SR latch*

Rel. 05.06.2018
© Prinetto - 2018

IBM LSSD D latch

- **Within the in-house developed *LSSD design methodology*, IBM uses the following hazard-free polarity hold D latch implementation:**



© Prinetto - 2018

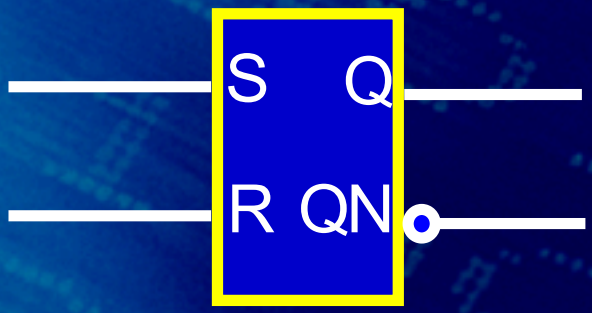
Latch types

- Several latch types exist, including:
  - *D latch*
  - *IBM LSSD D latch*
  - *SR latch*
  - *Gated SR latch*

Rel. 05.06.2018
© Prinetto - 2018

Set-Reset latch

- The ***Set-Reset latch*** can be considered the ***elementary building block*** for the implementation of all synchronous sequential circuits.
- It has:
  - 2 inputs: ***S*** and ***R***
  - 2 outputs: ***Q*** and ***QN***, such that ***QN*** = ***Q'***



Rel. 05.06.2018

© Prinetto - 2018

– The ***Set-***
elementa
impleme
sequentia

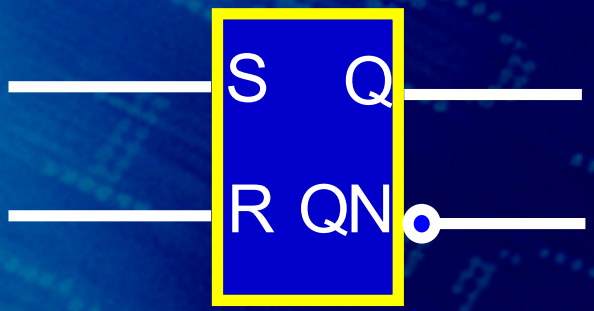
– It has:

. 2 inputs: ***S*** and ***R***

. 2 outputs: ***Q*** and ***QN***, such that ***QN*** = ***Q'***

Of the 4 combinations of S and R:

- One forces Q QN = 0 1 (reset)
- One forces Q QN = 1 0 (set)
- One lets Q QN unchanged
- One is forbidden



Rel. 05.06.2018

© Prinetto - 2018

– The ***Set-***
elementa
impleme
sequentia
– It has:

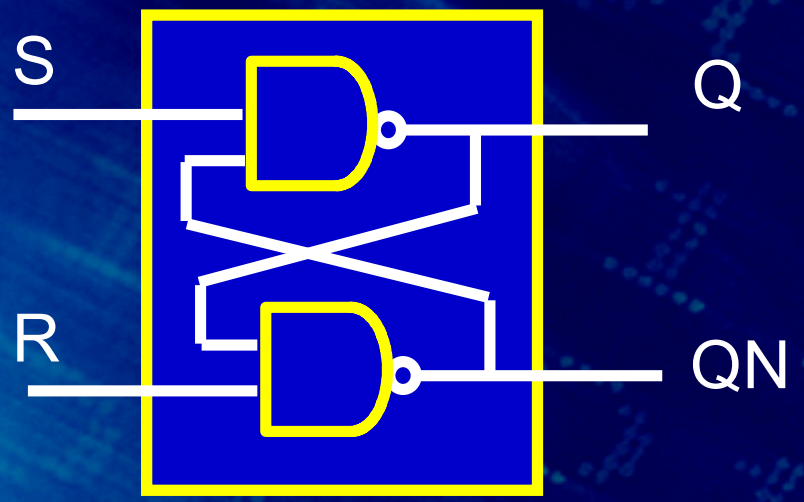
Of the 4 combinations of S and R:

- One forces Q QN = 0 1 (reset)
- One forces Q QN = 1 0 (set)
- One lets Q QN unchanged
- One is forbidden

The correspondence
input combination ⇔ performed operation
is technology and implementation dependent

© Prinetto - 2018

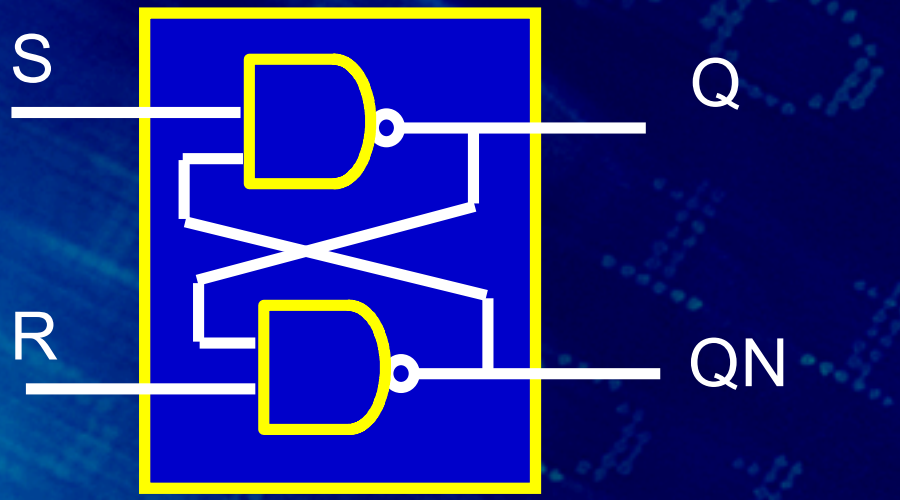
Set-Reset latch (nand implementation)

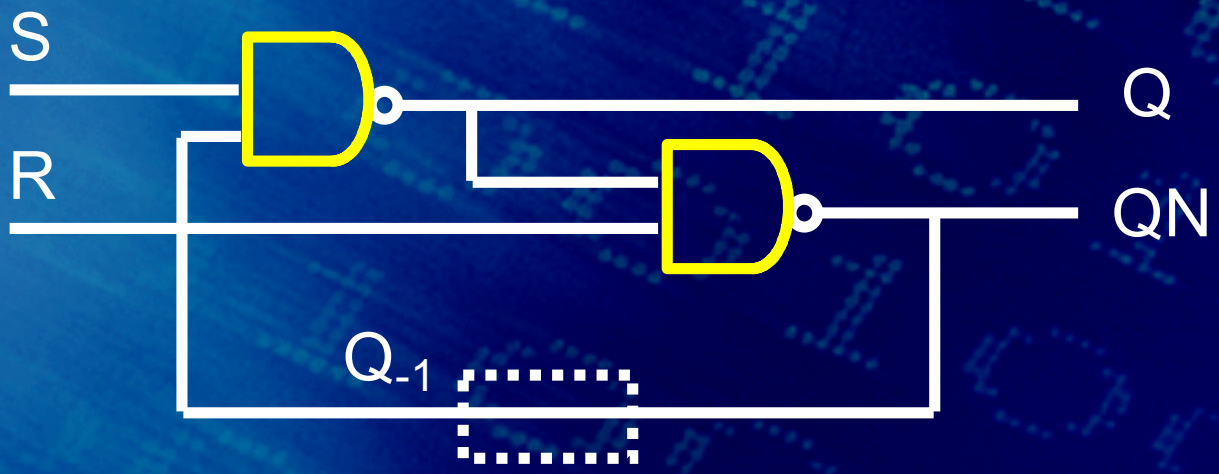


Rel. 05.06.2018

© Prinetto - 2018

Set-Reset latch (nand implementation)



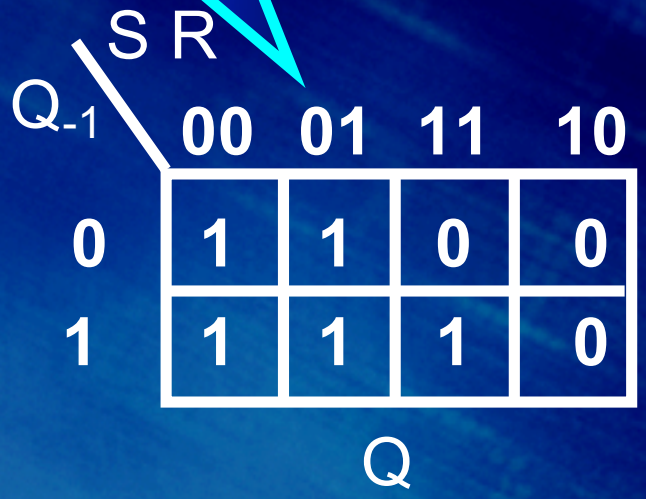


Rel. 05.06.2018

© Prinetto - 2018

Karnaugh Map

Set-Reset latch (nand implementation)



Rel. 05.06.2018

© Prinetto - 2018

Set-Reset latch (nand implementation)

Q_{-1} \ S R	00	01	11	10
0	1	1	0	0
1	1	1	1	0

Q

Characteristic Table

S	R	Q	QN
0	0	~~1~~	~~1~~
0	1	1	0
1	0	0	1
1	1	Q_{-1}	QN_{-1}

Rel. 05.06.2018

© Prinetto - 2018

Set-Reset latch (nand implementation)



S	R	Q	QN
0	0	1	1
			0

This combination is forbidden since:

- it forces both outputs to 1, thus preventing QN = Q'
- a transition

$$S\ R = 00 \rightarrow S\ R = 11$$

forces the circuit to enter an unpredictable state, whose value depends from internal delays

Rel. 05.06.2018
© Prinetto - 2018

Set-Reset latch (nand implementation)

Q_{-1} \ S R	00	01	11	10
0	1	1	0	0
1	1	1	1	0

Q

S	R	Q	QN
0	0	~~1~~	~~1~~
0	1	1	0
1	0	0	1
1	1	Q_{-1}	QN_{-1}

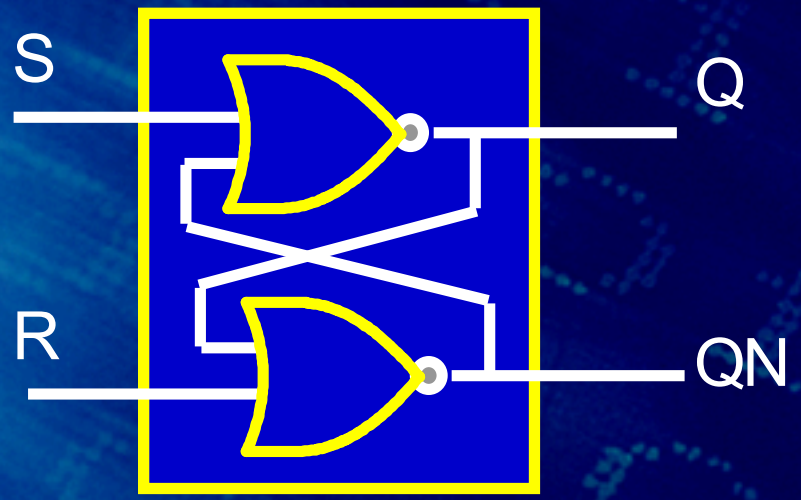
$Q_{-1} \rightarrow$	Q	S	R
0	→ 0	1	-
0	→ 1	0	1
1	→ 0	1	0
1	→ 1	-	1

Transition Table

Rel. 05.06.2018

© Prinetto - 2018

Set-Reset latch (nor implementation)



Rel. 05.06.2018

© Prinetto - 2018

Set-Reset latch (nor implementation)

Karnaugh Map

Q_{-1} \ S R	00	01	11	10
0	0	1	0	0
1	1	1	0	0

Q

Characteristic Table

S	R	Q	QN
0	0	Q_{-1}	QN_{-1}
0	1	1	0
1	0	0	1
~~1~~	~~1~~	0	0

Transition Table

Q_{-1}	→	Q	S	R
0	→	0	-	0
0	→	1	0	1
1	→	0	1	0
1	→	1	0	-

Rel. 05.06.2018

© Prinetto - 2018

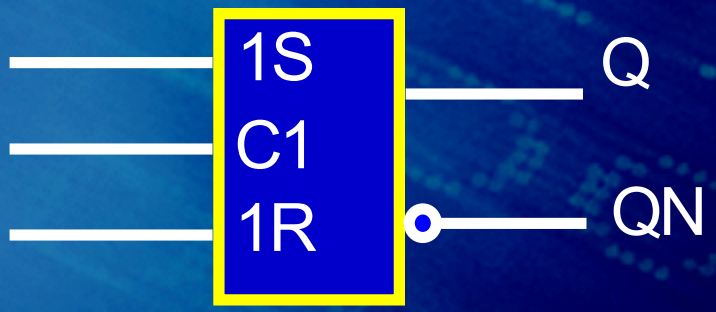
Latch types

- Several latch types exist, including:
  - *D latch*
  - *IBM LSSD D latch*
  - *SR latch*
  - *Gated SR latch*

Rel. 05.06.2018
© Prinetto - 2018

Gated Set-Reset latch

- It's a variation of the Set-Reset latch, in which data are allowed to enter the latch when a given control signal C is asserted, only.

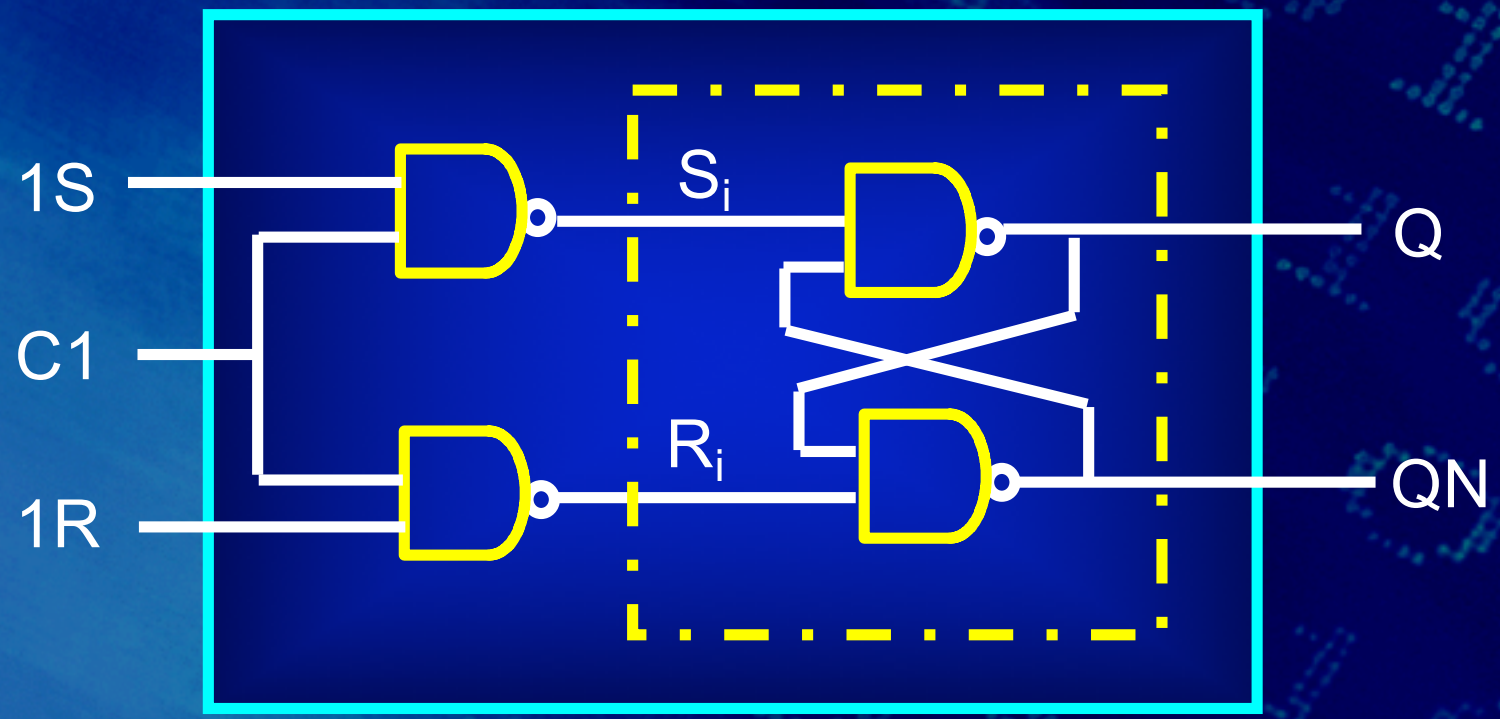


Symbol IEEE

Rel. 05.06.2018

© Prinetto - 2018

Possible implementation



Rel. 05.06.2018

© Prinetto - 2018

Characteristic table

C1	1S	1R	S_i	R_i	Q	QN
0	-	-	1	1	Q_{-1}	QN_{-1}
1	0	0	1	1	Q_{-1}	QN_{-1}
1	0	1	1	0	0	1
1	1	0	0	1	1	0
1	1	1	0	0	~~X~~	~~X~~

Rel. 05.06.2018

© Prinetto - 2018

Outline

- Latches vs. Flip-Flops
- Latches
- Flip-Flops
- Scannable Flip-Flops

Rel. 05.06.2018
© Prinetto - 2018

Flip-Flop

- **A *flip-flop* is a single-bit storage device implemented as a *Moore machine*.**

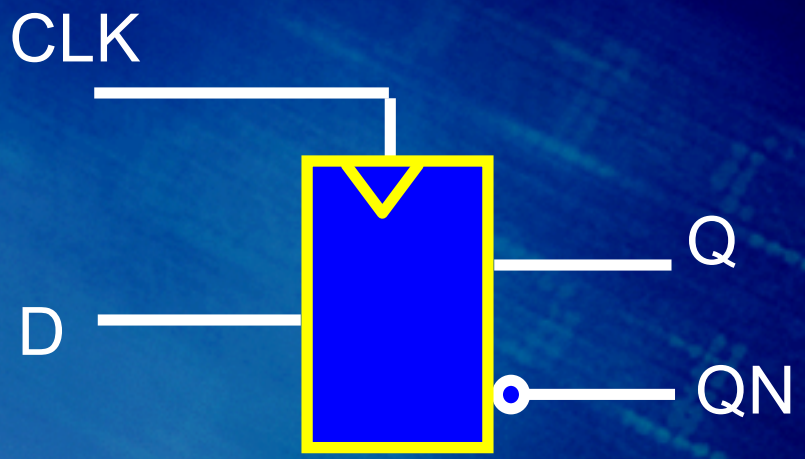
Rel. 05.06.2018
© Prinetto - 2018

FF classification w.r.t. data inputs

- **According to their behavior w.r.t. data inputs, flip-flops are usually classified as:**
  - ***D***
  - ***SR***
  - ***JK***
  - ***T***

Rel. 05.06.2018
© Prinetto - 2018

D behavior



D	Q
0	0
1	1

Characteristic Table

Rel. 05.06.2018

© Prinetto - 2018

Transition Table

D	Q
0	0
1	1

Characteristic Table



Q_{-1}	→	Q	D
0	→	0	0
0	→	1	1
1	→	0	0
1	→	1	1
-	→	0	0
-	→	1	1

Rel. 05.06.2018

© Prinetto - 2018

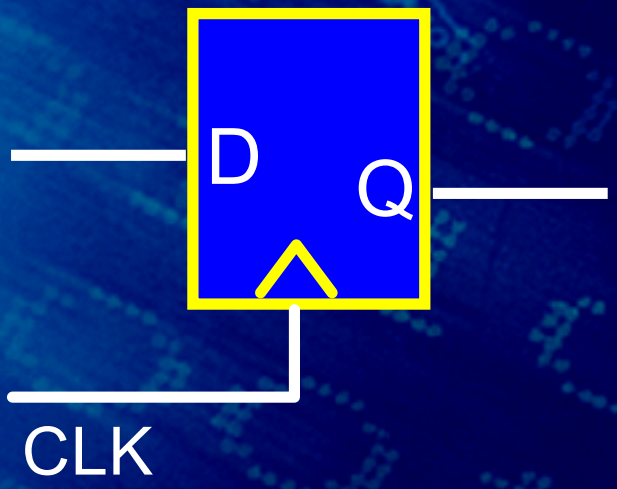
Caveat

- **Distinction between data inputs and control inputs is context and application dependent.**

Rel. 05.06.2018
© Prinetto - 2018

Example (D = data input)

- **At any CLK, it stores the input data**



© Prinetto - 2018

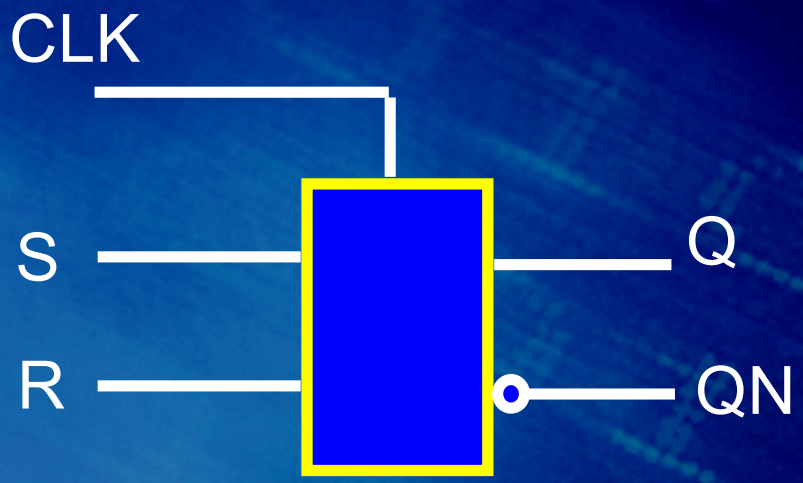
Example (D = control input)

- **If (D = 0)**
  then force Q=0
  else force Q=1



Rel. 05.06.2018
© Prinetto - 2018

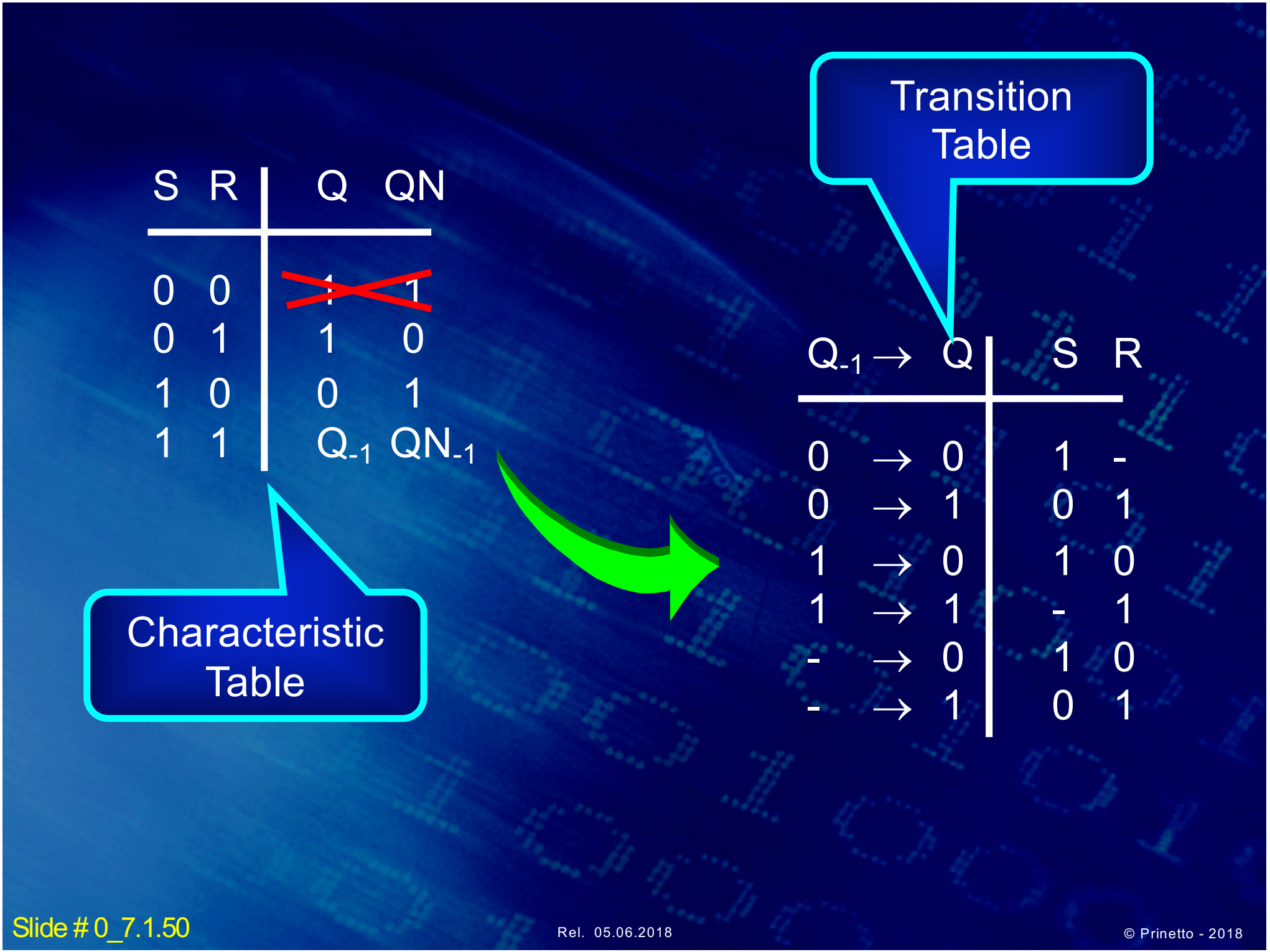
Set-Reset behavior



S	R	Q	QN
0	0	~~1~~	~~1~~
0	1	1	0
1	0	0	1
1	1	Q_{-1}	QN_{-1}

Rel. 05.06.2018

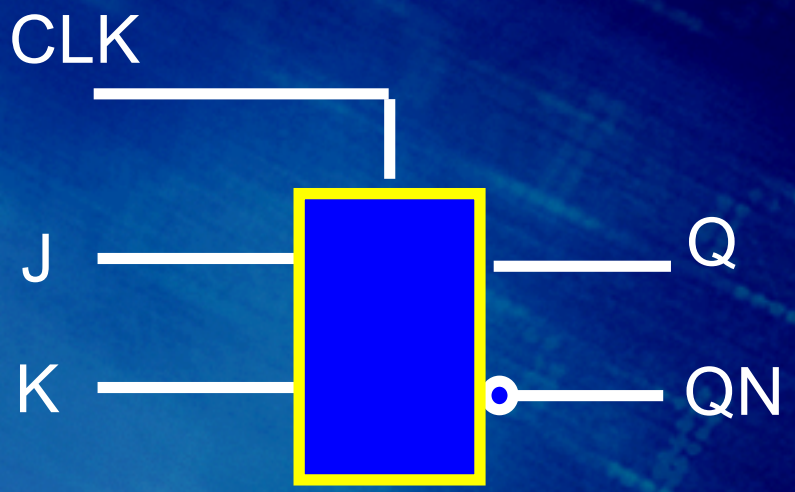
© Prinetto - 2018



Rel. 05.06.2018

© Prinetto - 2018

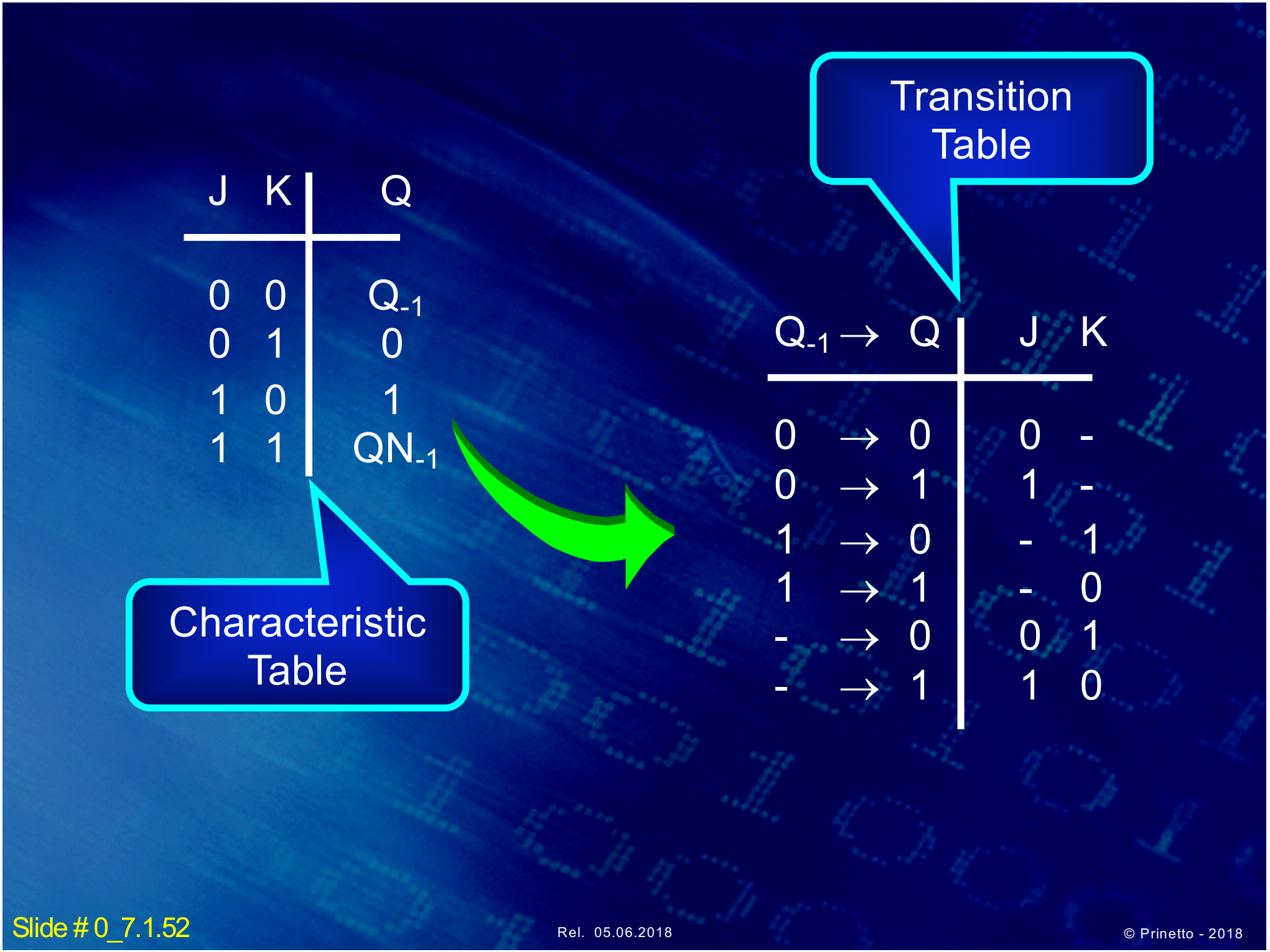
JK behavior



J	K	Q
0	0	Q_{-1}
0	1	0
1	0	1
1	1	QN_{-1}

Rel. 05.06.2018

© Prinetto - 2018



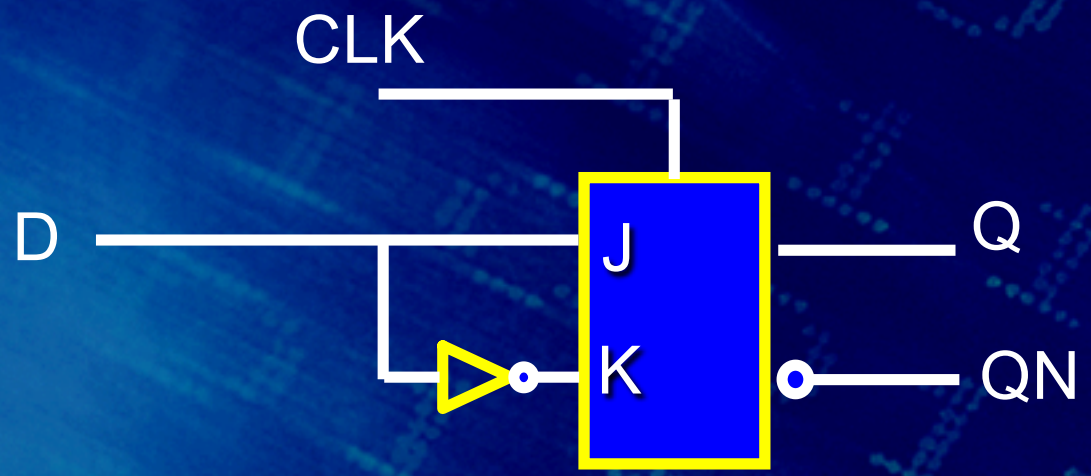
Rel. 05.06.2018

© Prinetto - 2018

D FF implemented by a JK FF

© Prinetto - 2018

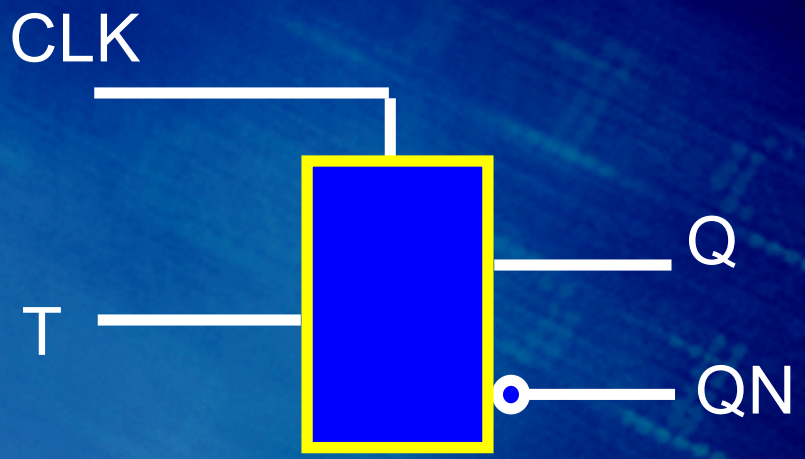
D FF implemented by a JK FF



Rel. 05.06.2018

© Prinetto - 2018

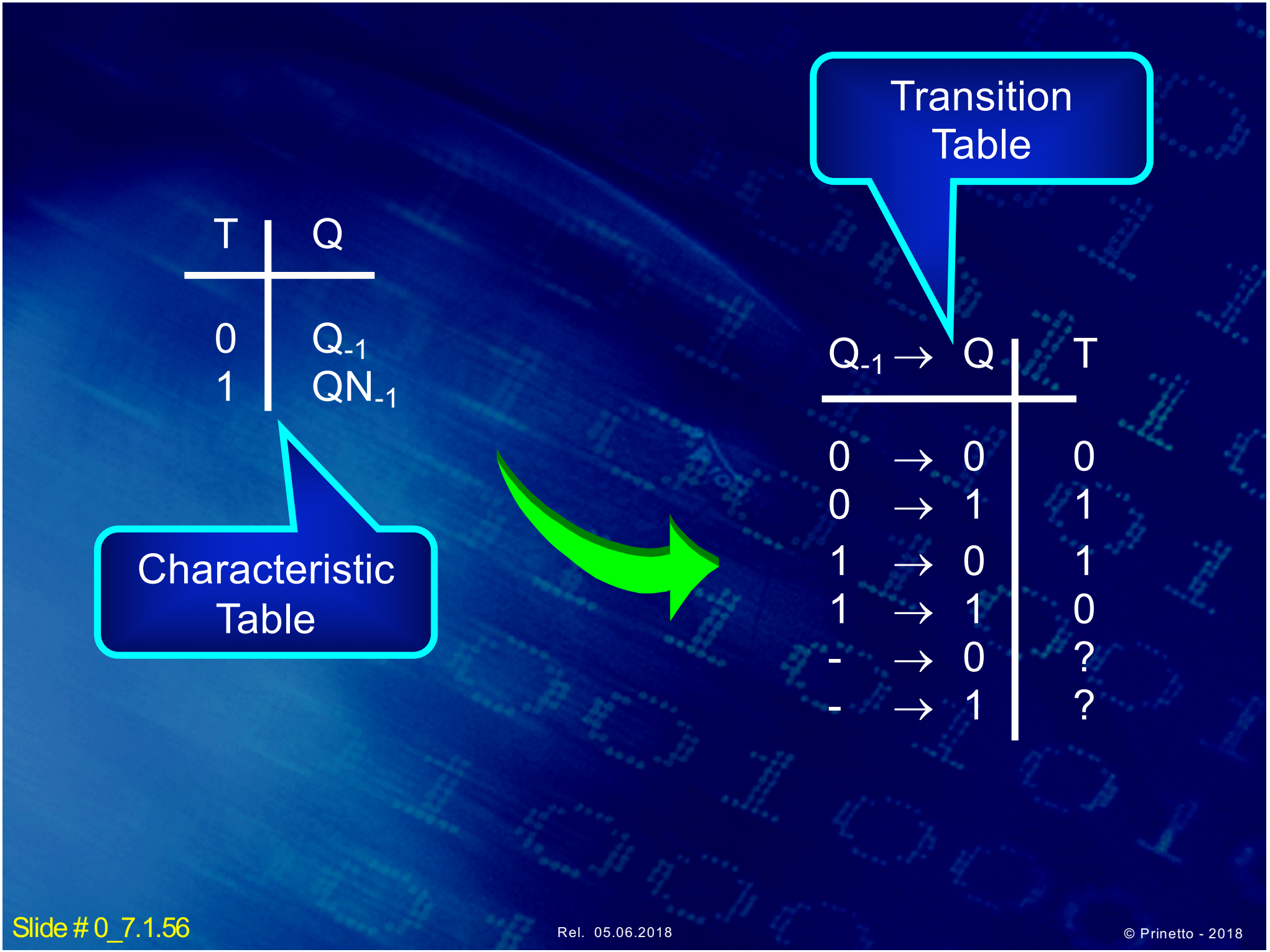
T behavior



T	Q
0	Q_{-1}
1	QN_{-1}

Rel. 05.06.2018

© Prinetto - 2018



Rel. 05.06.2018

© Prinetto - 2018

Caveat

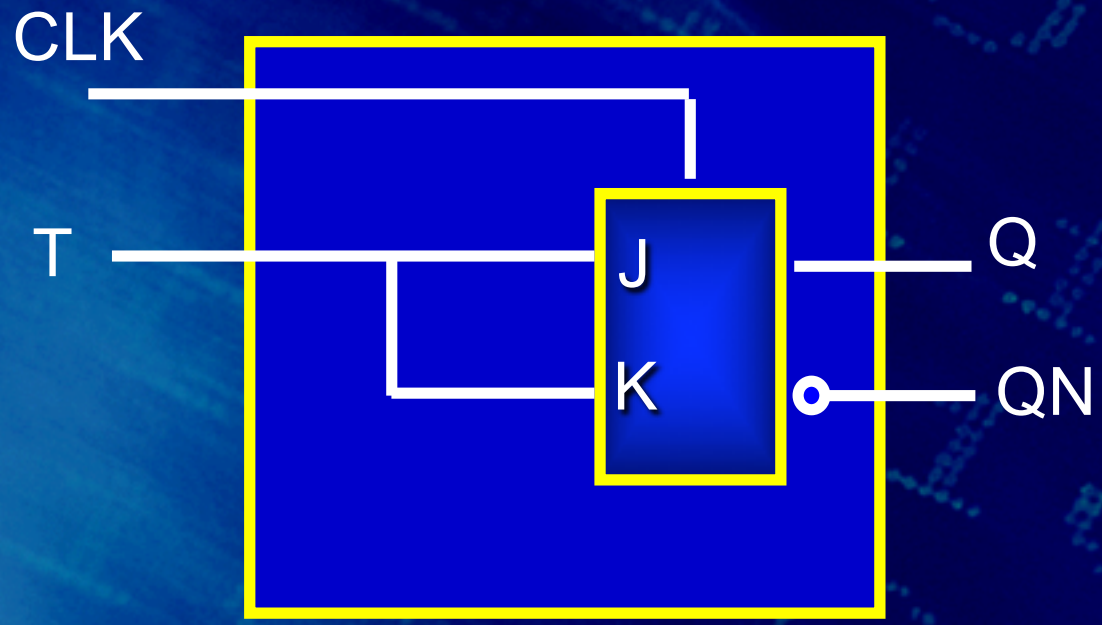
- **A T type flip-flop cannot be initialized: if you don’t know the value of the output, acting only on the input T, you cannot state the value of the output.**

© Prinetto - 2018

T FF implemented by a JK FF

© Prinetto - 2018

T FF implemented by a JK FF



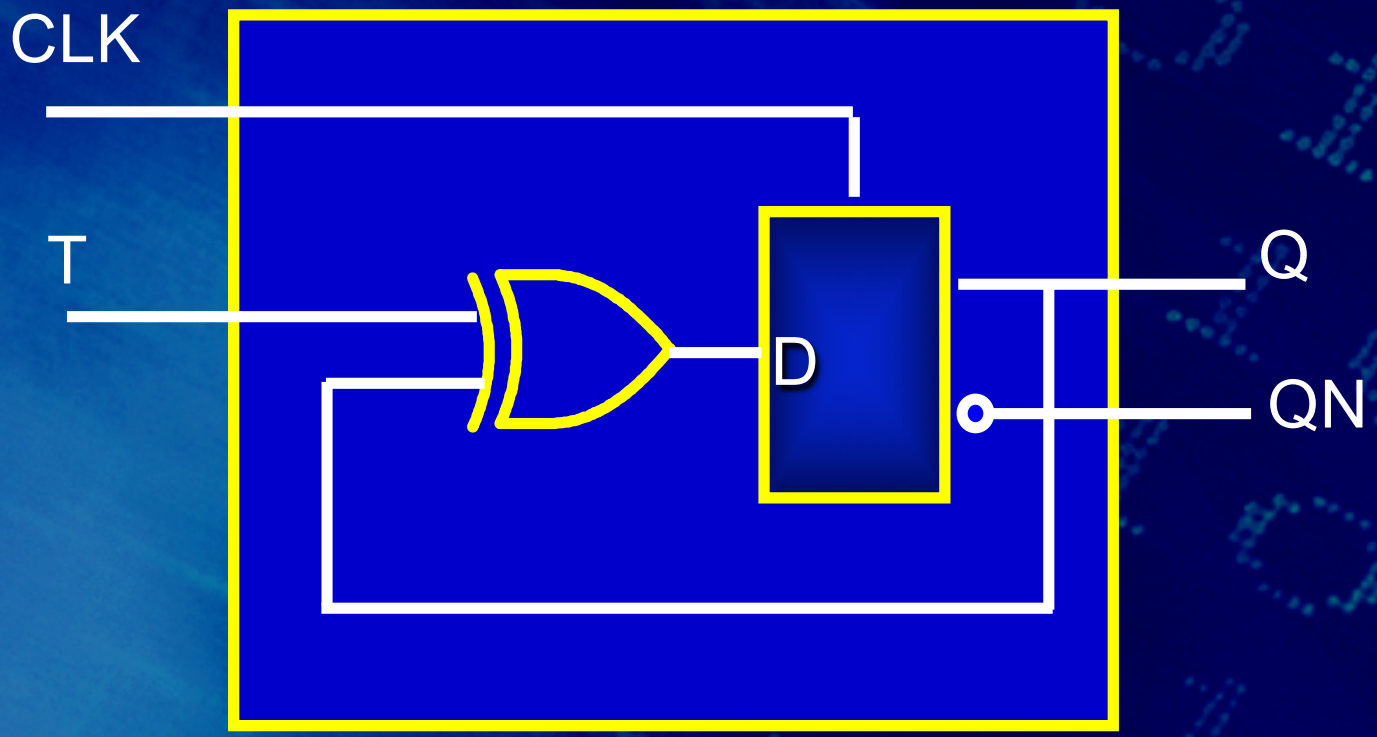
Rel. 05.06.2018

© Prinetto - 2018

T FF implemented by a D FF

© Prinetto - 2018

T FF implemented by a D FF

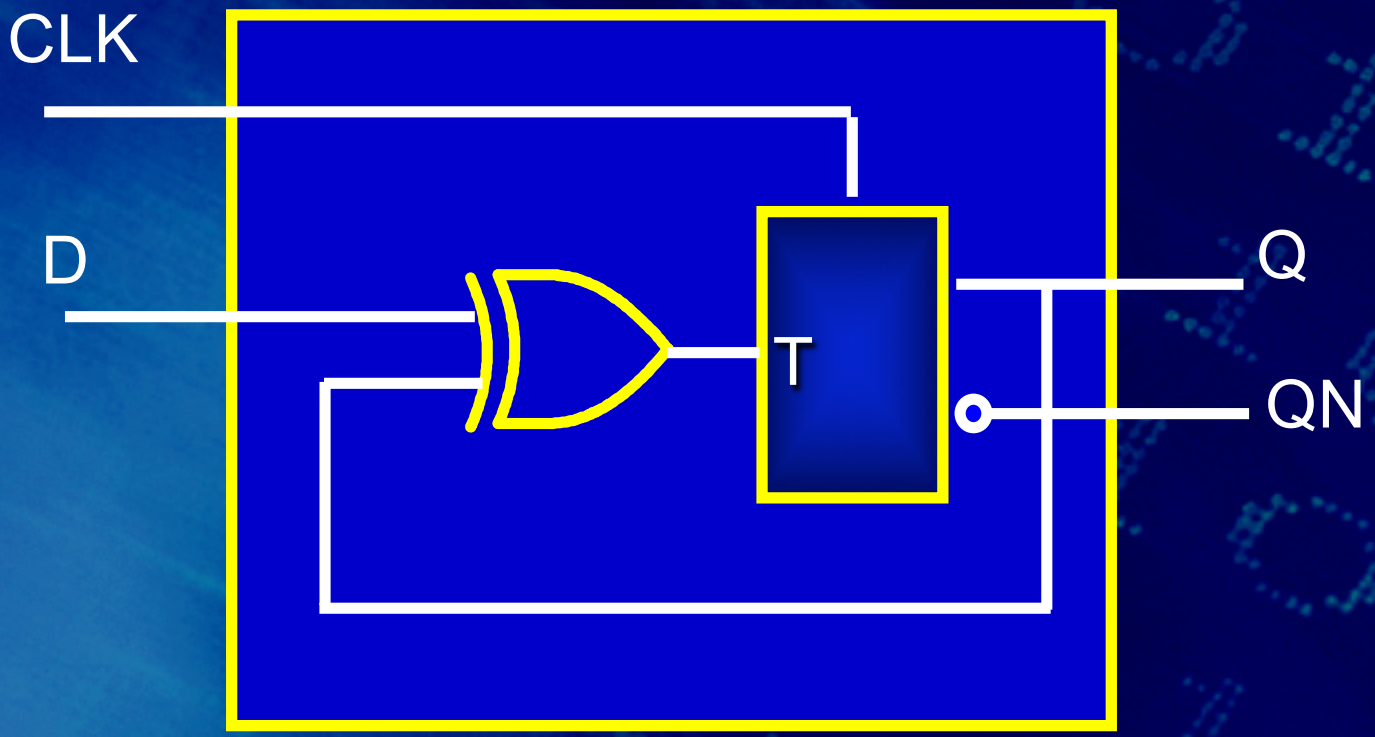


© Prinetto - 2018

D FF implemented by a T FF

© Prinetto - 2018

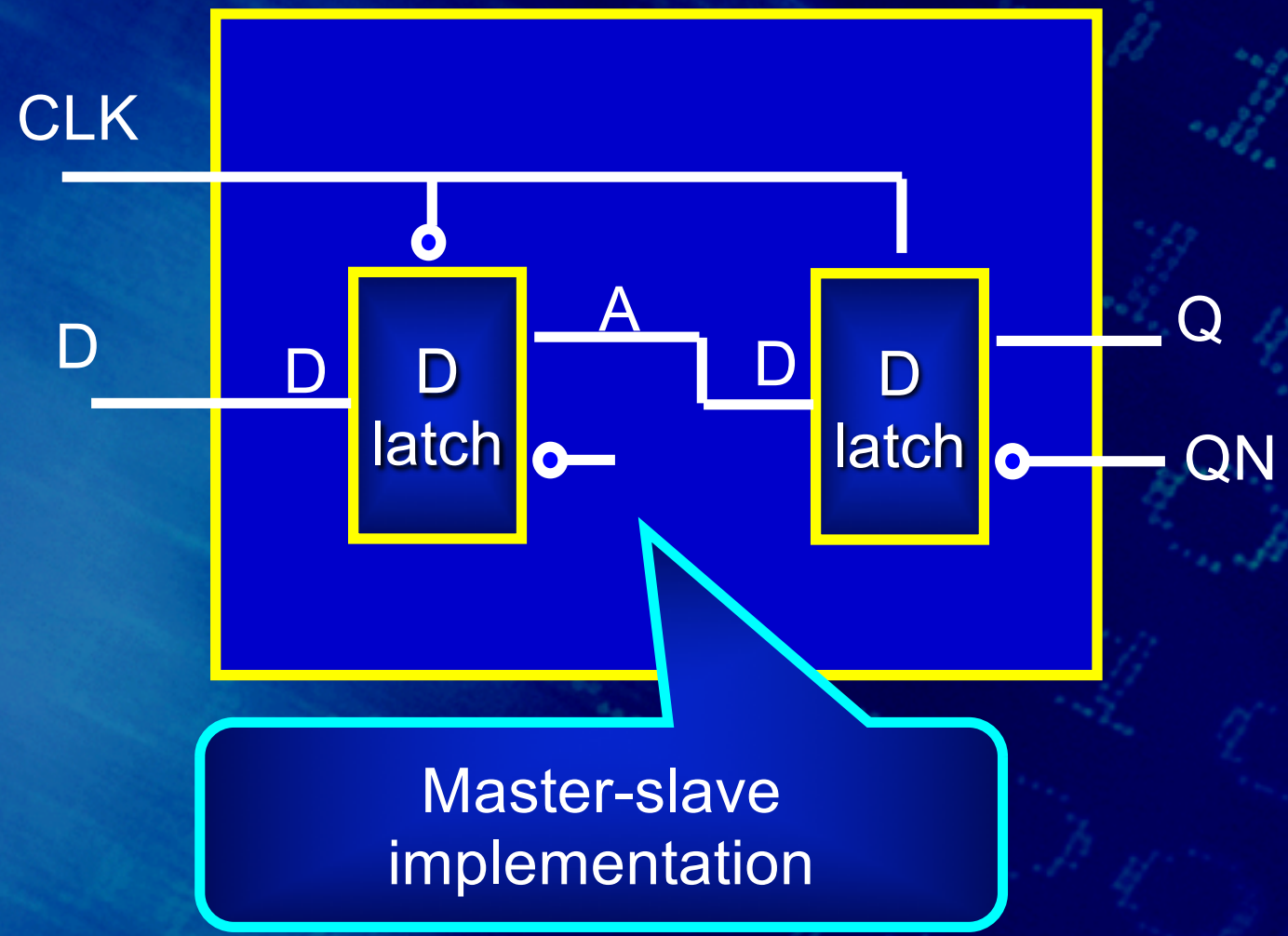
D FF implemented by a T FF



Rel. 05.06.2018

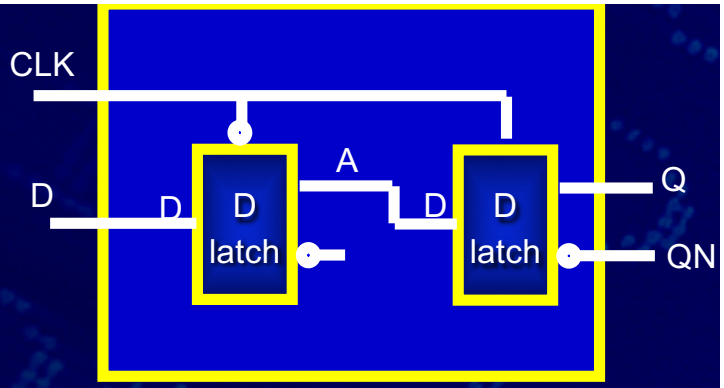
© Prinetto - 2018

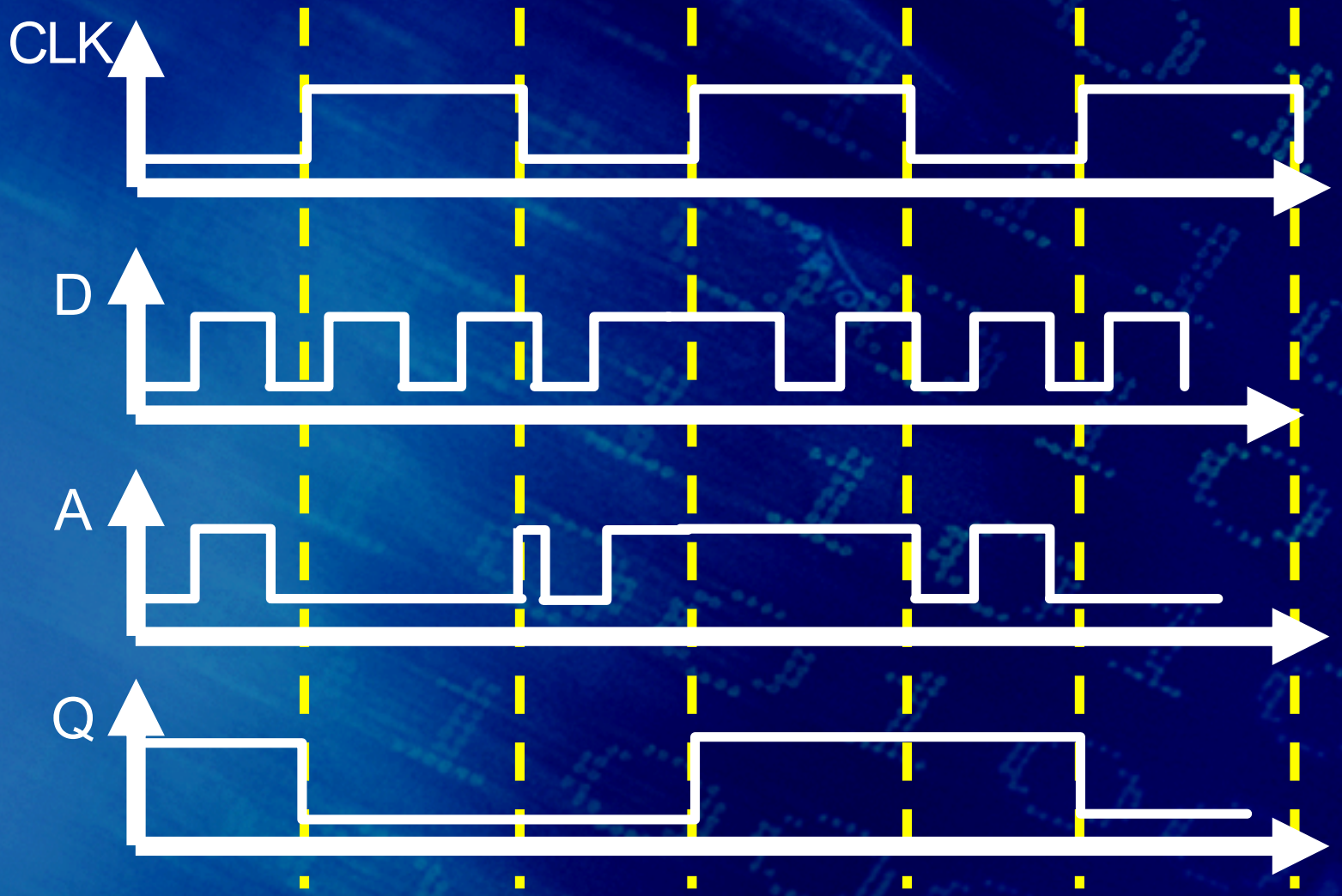
D FF positive edge triggered implemented by latches



© Prinetto - 2018

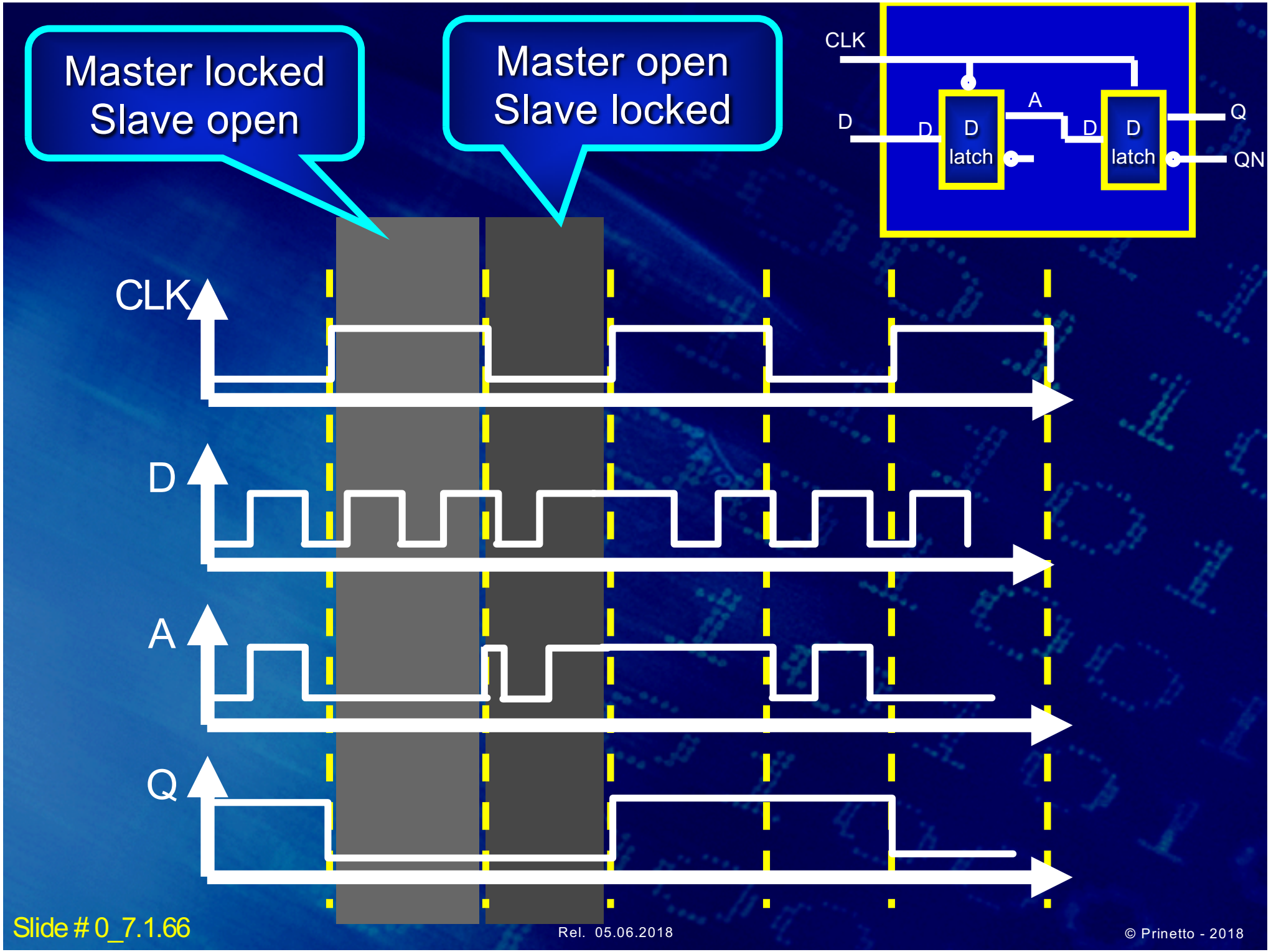
Waveforms





Rel. 05.06.2018

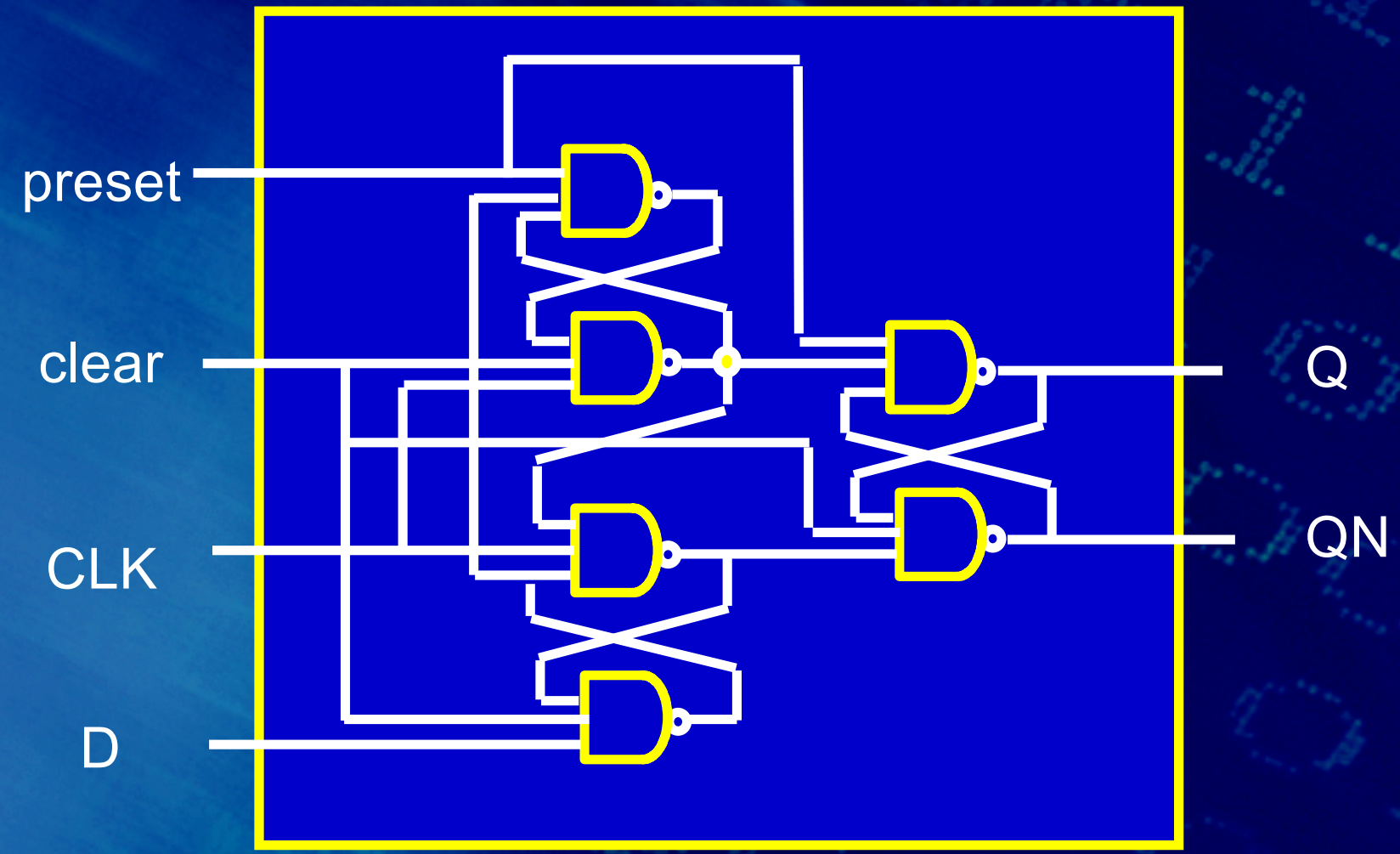
© Prinetto - 2018



Rel. 05.06.2018

© Prinetto - 2018

D FF positive edge triggered implemented in TTL logic (‘74)



Rel. 05.06.2018

© Prinetto - 2018

The D FF as a library cell

- A *D flip flop* is a sequential functional block able to perform the following operations:
  - *asynchronous clear*
  - *synchronous clear*
  - *synchronous preset*
  - *data parallel load*
  - *stored data hold*

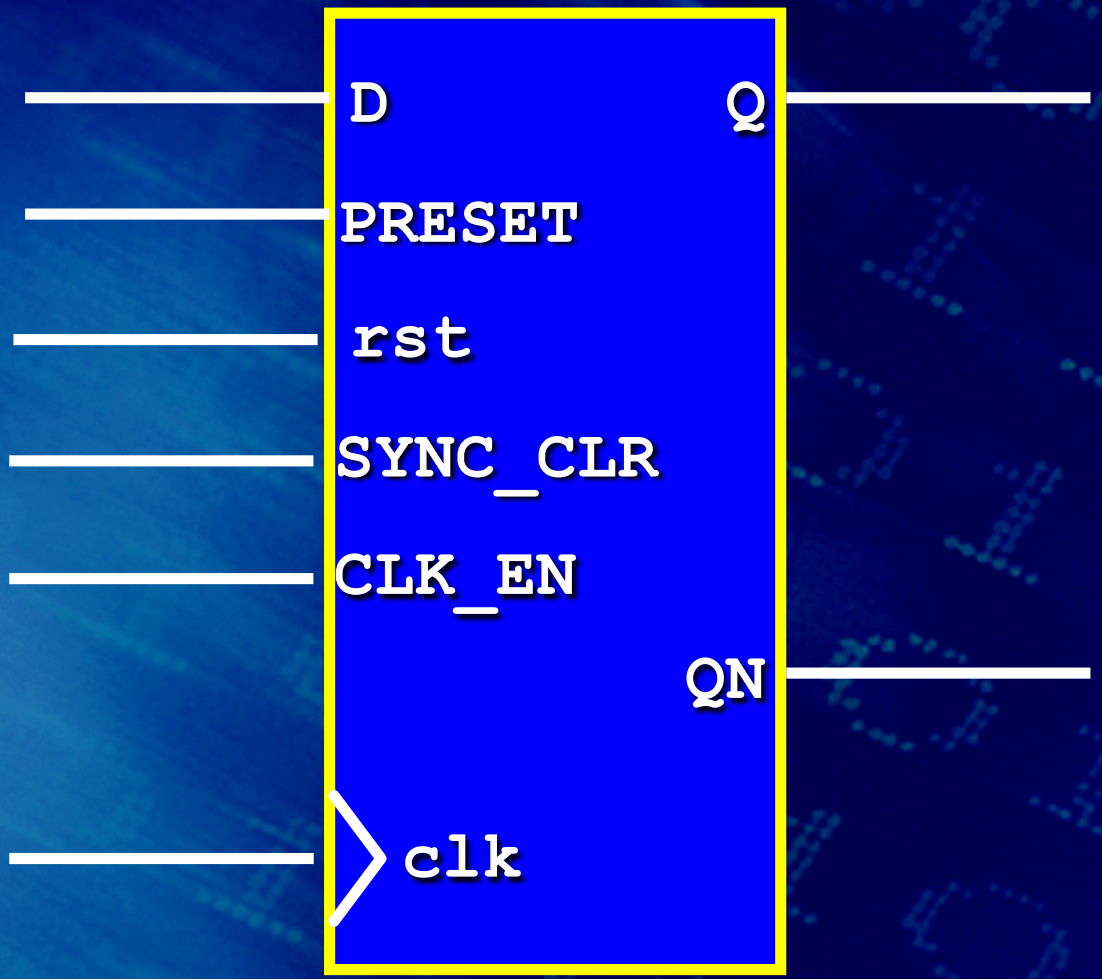
Rel. 05.06.2018
© Prinetto - 2018

Flip flop D: I/O signals

- Input Data signals : `D`
- Output Data signals : `Q,`
- `QN`
- Clock signal : `clk`
- Reset signal : `rst`
- Input Control signals : `CLK_EN`
  `SYNC_CLR`
  `PRESET`
- Output Control signals : none

Rel. 05.06.2018
© Prinetto - 2018

Flip flop D: symbol



Rel. 05.06.2018

© Prinetto - 2018

VHDL Behavioral description

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity pdt_ffd is
port(
     D: in std_logic;
     rst, SYN_CLR, CLK_EN: in std_logic;
     clk: in std_logic;
     Q,NQ: out std_logic );
end PdT_FFD;
```


© Prinetto - 2018

```
architecture pdt_ffd_arch of pdt_ffd is
  type state_set is (s0, s1);
  signal state : state_set;
  begin
    process(rst, clk)          -- to compute FS
      begin
      if rst ='1' then
        state<=s0;             -- Q<='0';
      elsif  clk' event and clk='1' then
      if CLK_EN='1' then
        if  SYN_CLR ='1' then
          state<=s0;           -- Q<='0';
        elsif D='1' then
          state<=s1;           -- Q<='1';
        else
          state<=s0;           -- Q<='0';
        end if;
      end if;
    end if;
  end process;
```


© Prinetto - 2018

VHDL Behavioral description *(cont'd)*

```
process(state)          -- to compute PO
  begin
  if state=s0 then
     Q<='0';
     NQ<='1';
  else
     Q<='1';
     NQ<='0';
  end if;
end process;
end pdt_ffd_arch;
```


© Prinetto - 2018

Outline

- Latches vs. Flip-Flops
- Latches
- Flip-Flops
- Scannable Flip-Flops

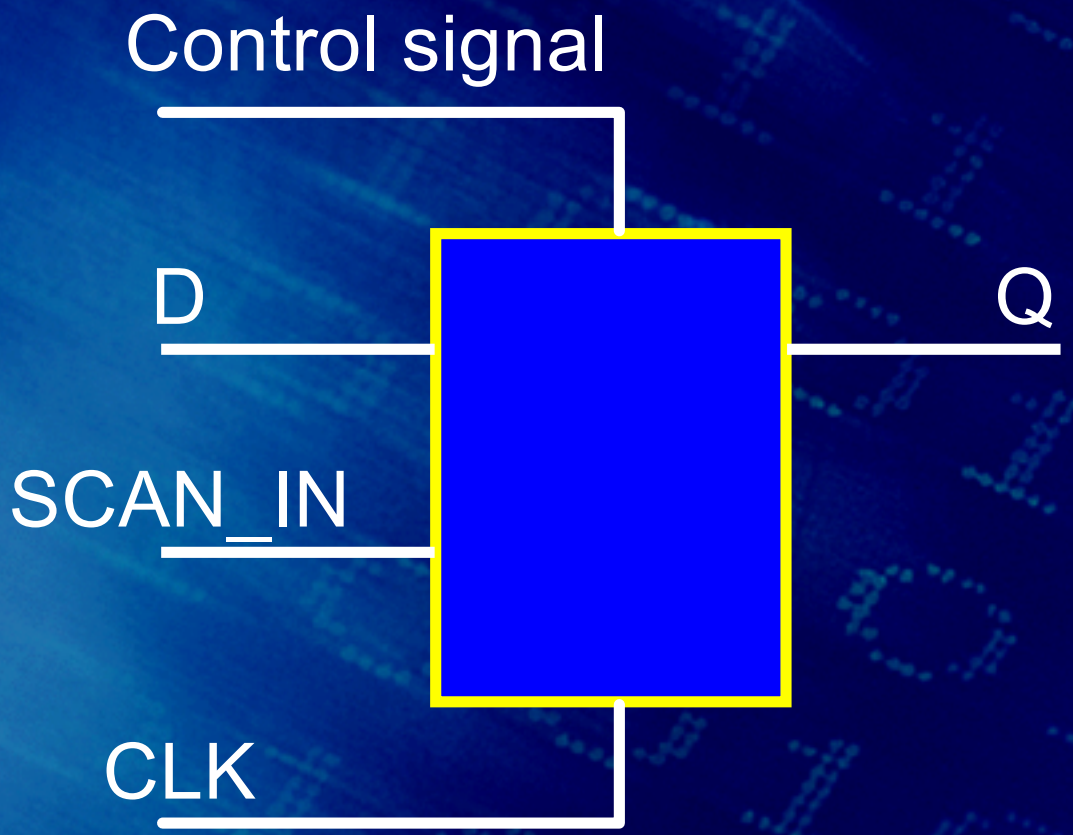
© Prinetto - 2018

Scannable Flip-flops

- **Most cell libraries include particular FF families, particularly suited to improve testability, resorting to the so called *Scan chains*.**

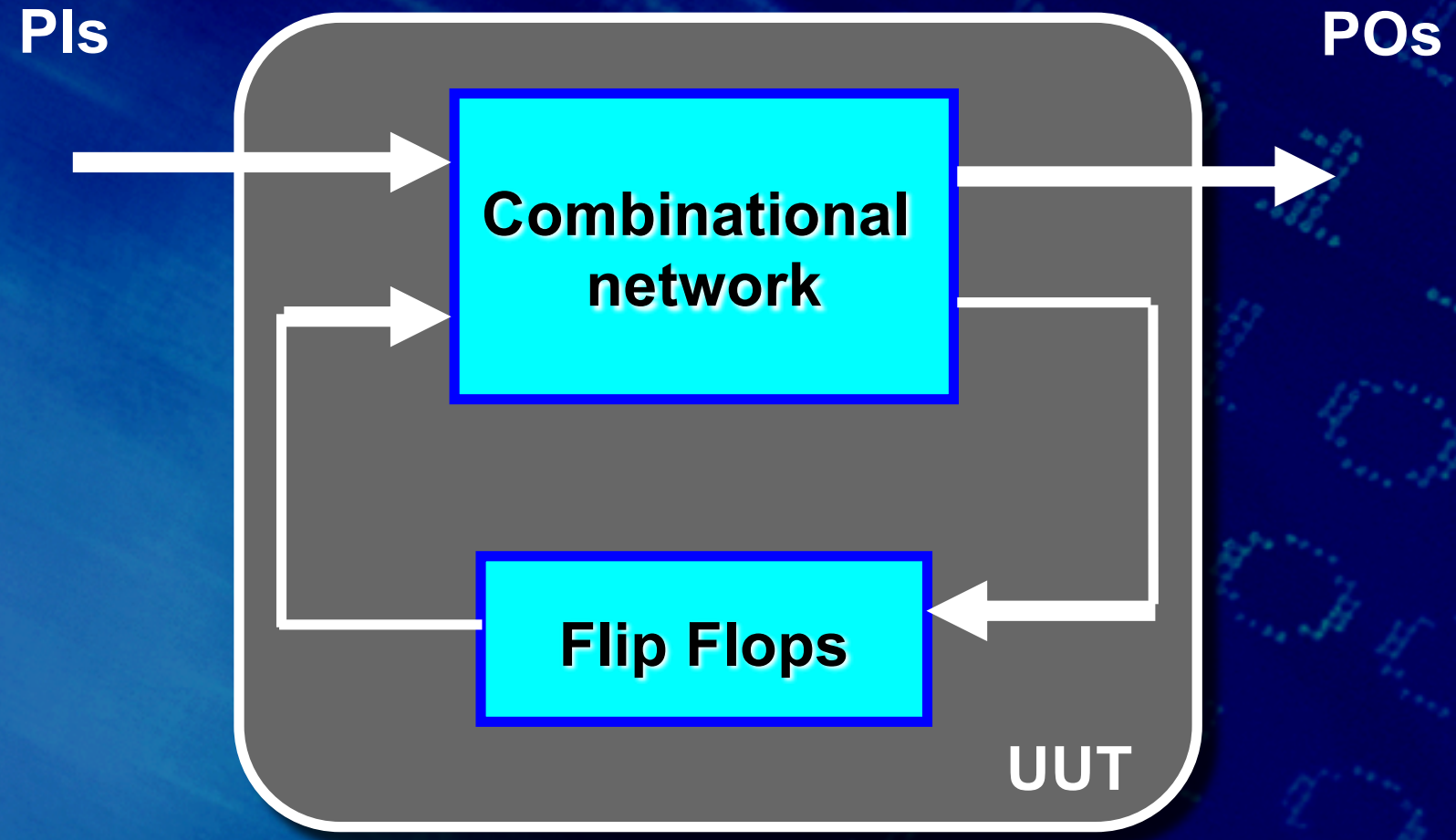
© Prinetto - 2018

Flip-flop scannable



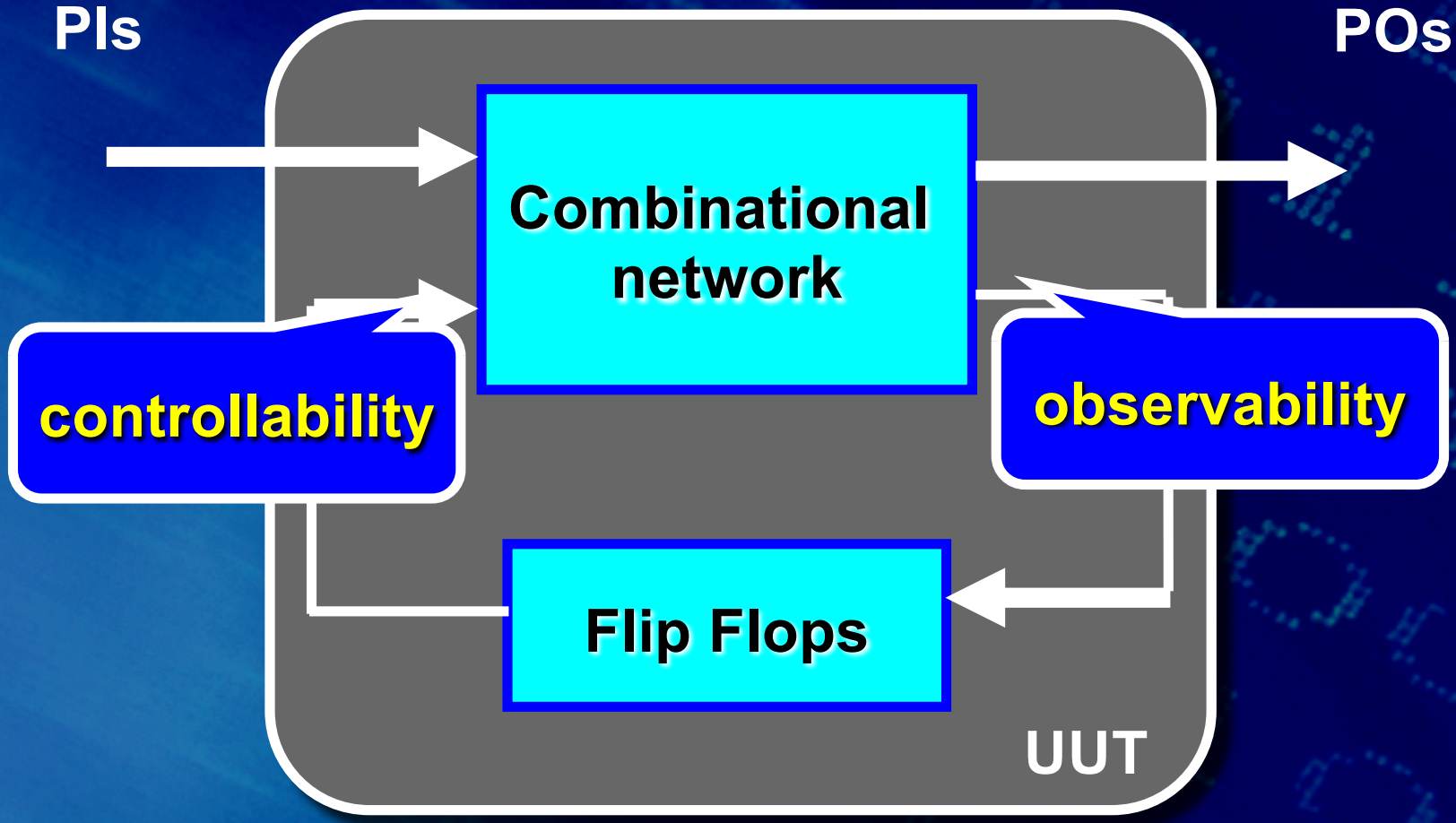
© Prinetto - 2018

The issue



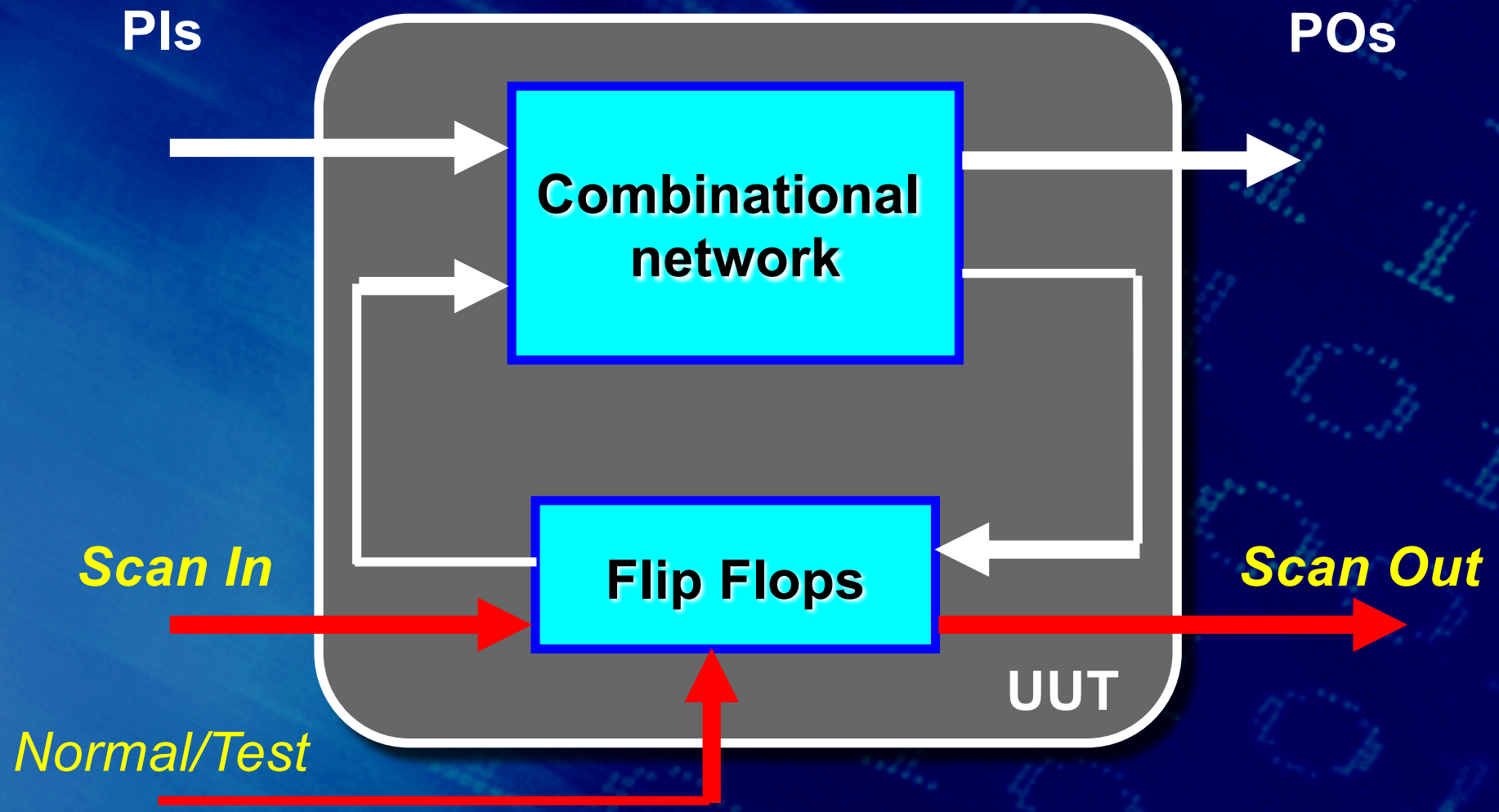
© Prinetto - 2018

The issue



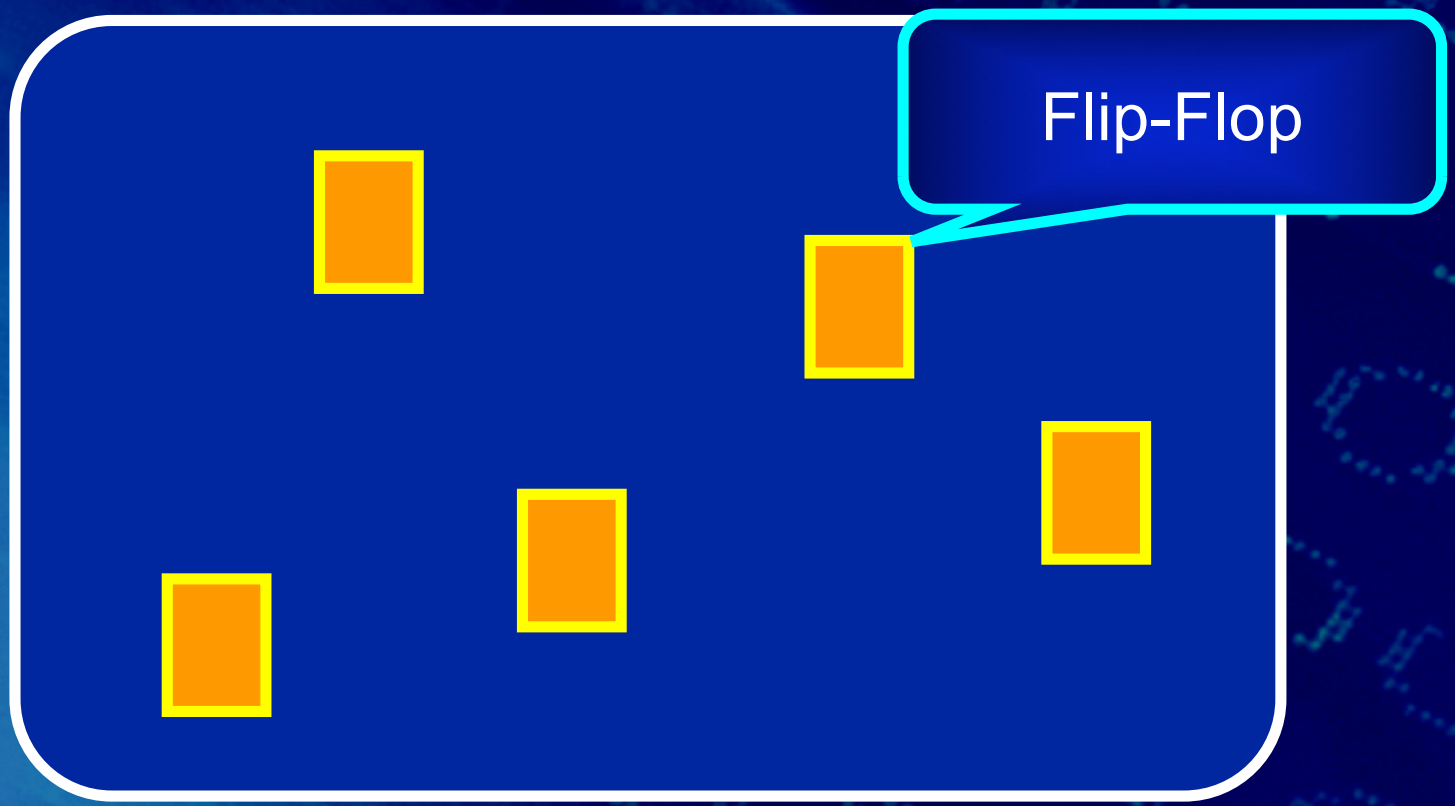
© Prinetto - 2018

The solution



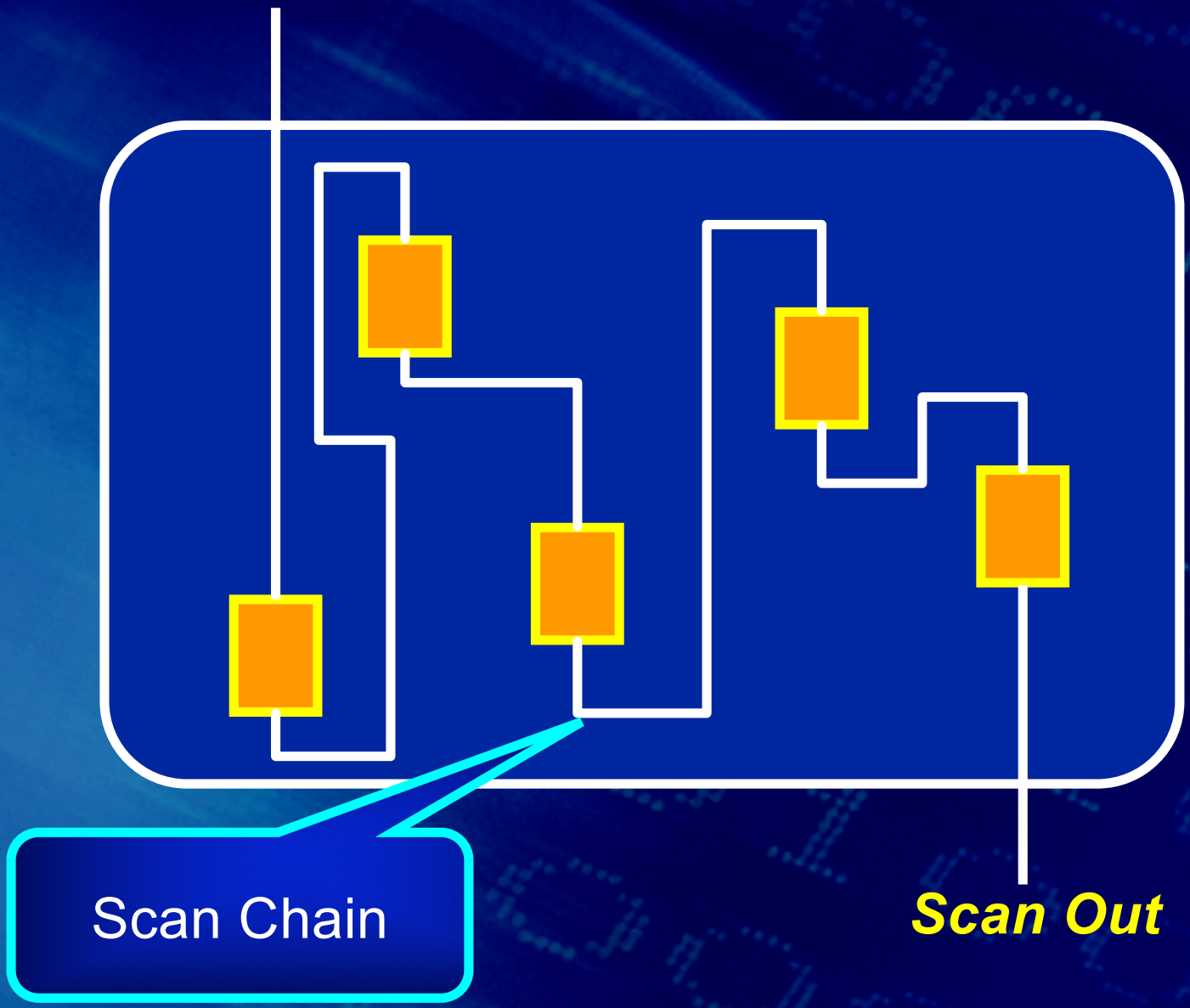
© Prinetto - 2018

The solution



© Prinetto - 2018

The solution



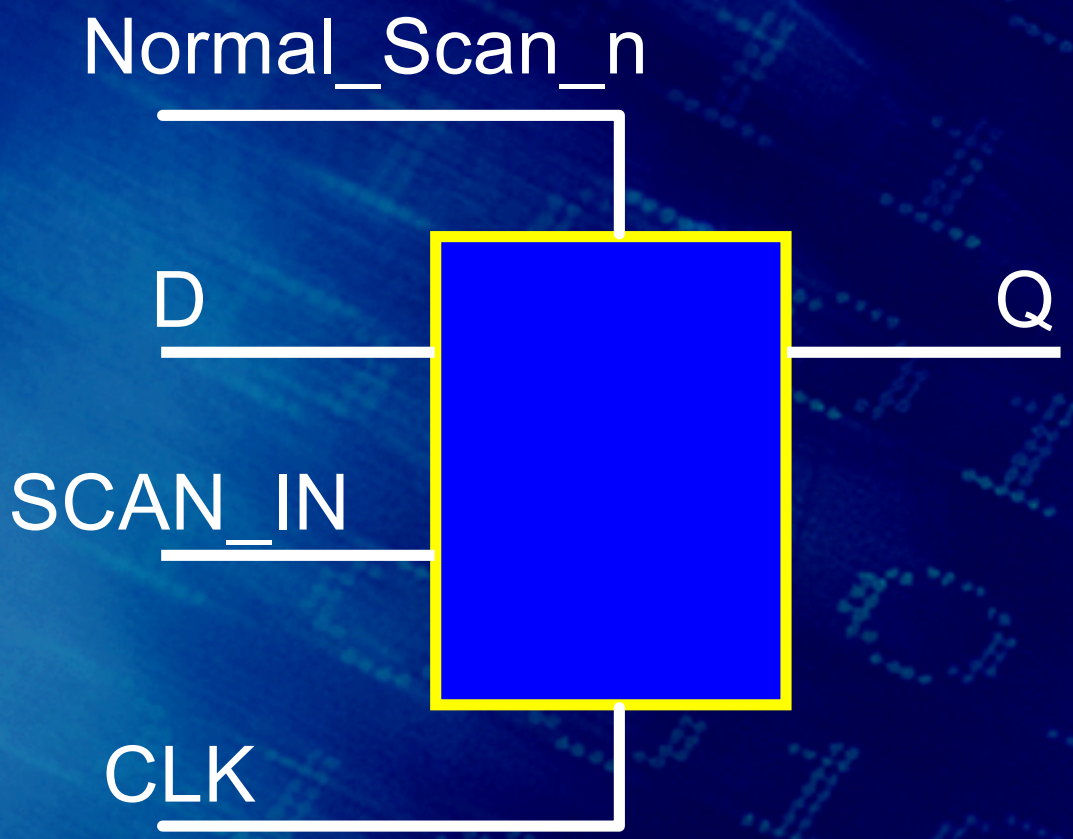
© Prinetto - 2018

Implementations

- The ***control signal*** states the value to be stored in the FF, i.e., the D or the SCAN_IN, respectively.
- Two implementations are usually adopted:
  - ***mux-scan***
  - ***clock-scan***

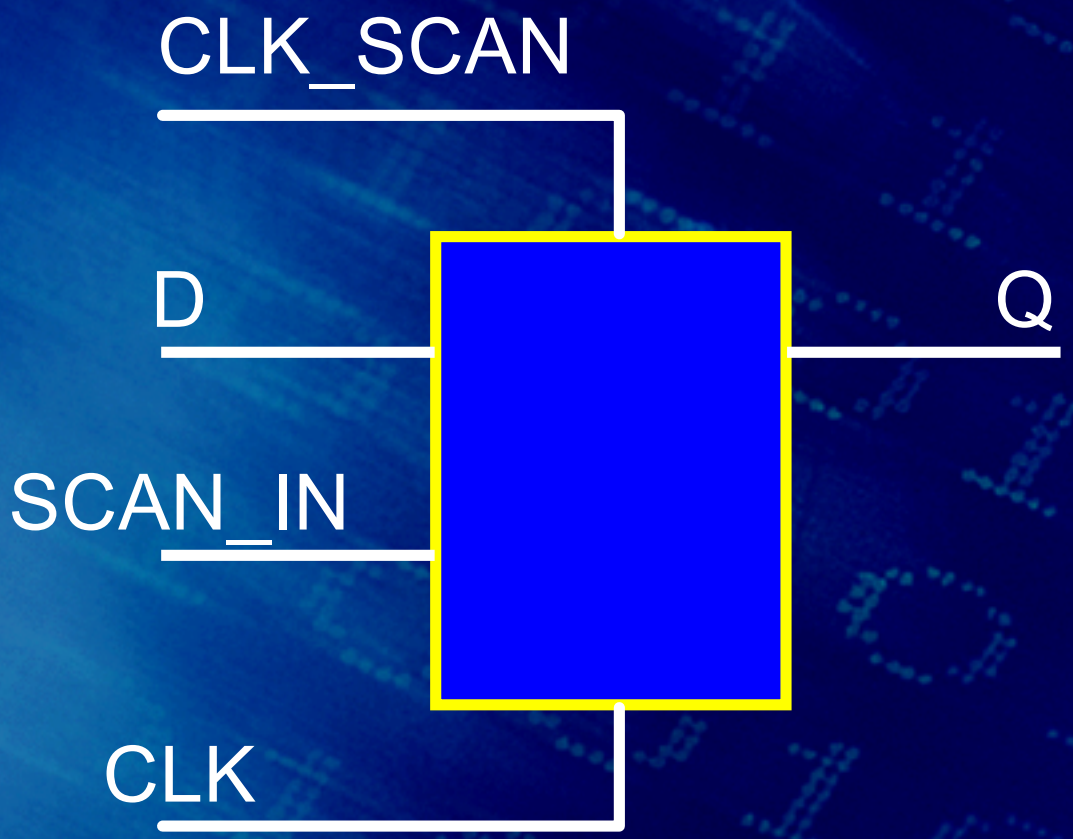
Rel. 05.06.2018
© Prinetto - 2018

Mux-scan approach

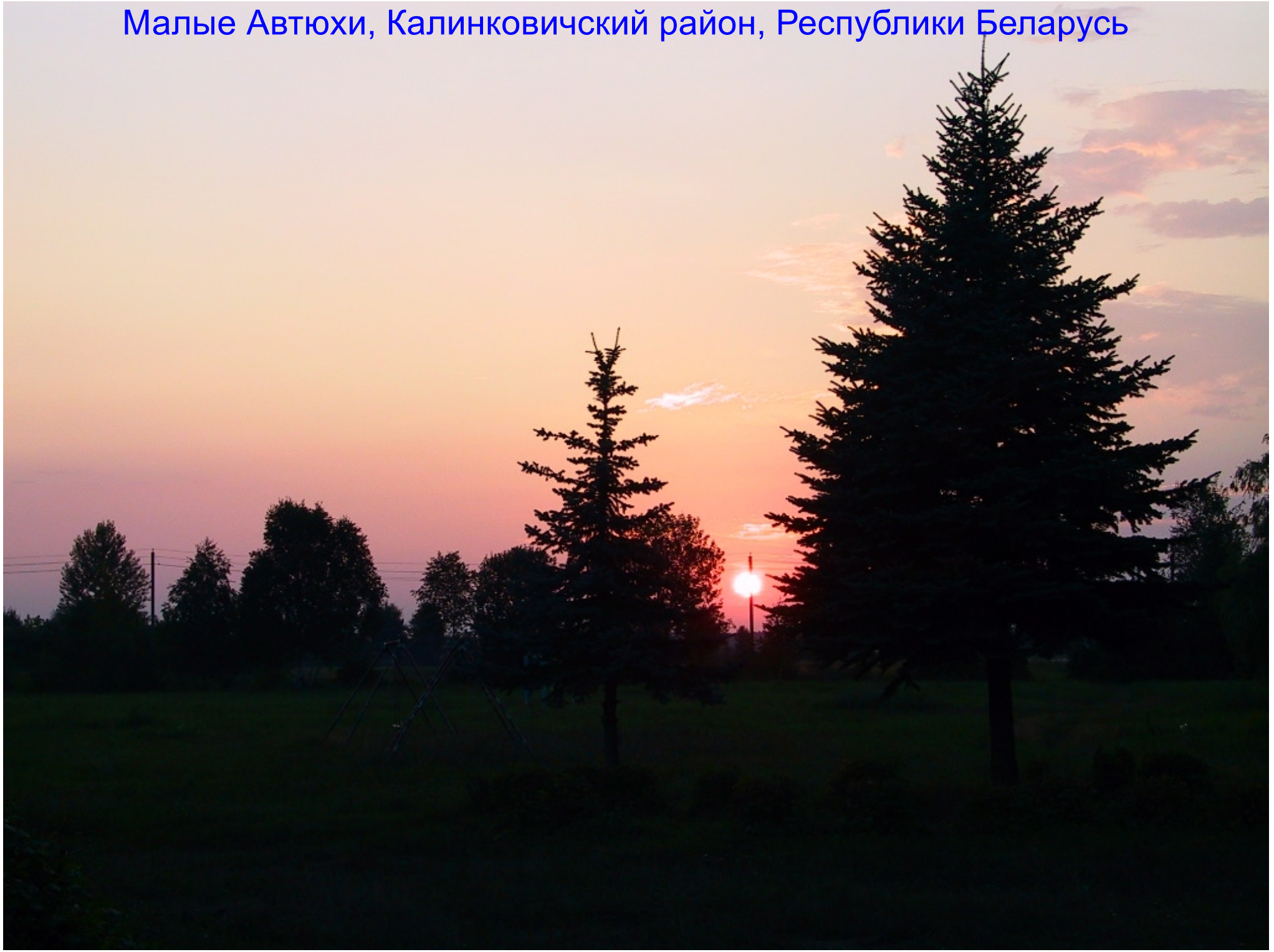


© Prinetto - 2018

Clock-scan approach



© Prinetto - 2018


Малые Автюхи, Калинковичский район, Республики Беларусь