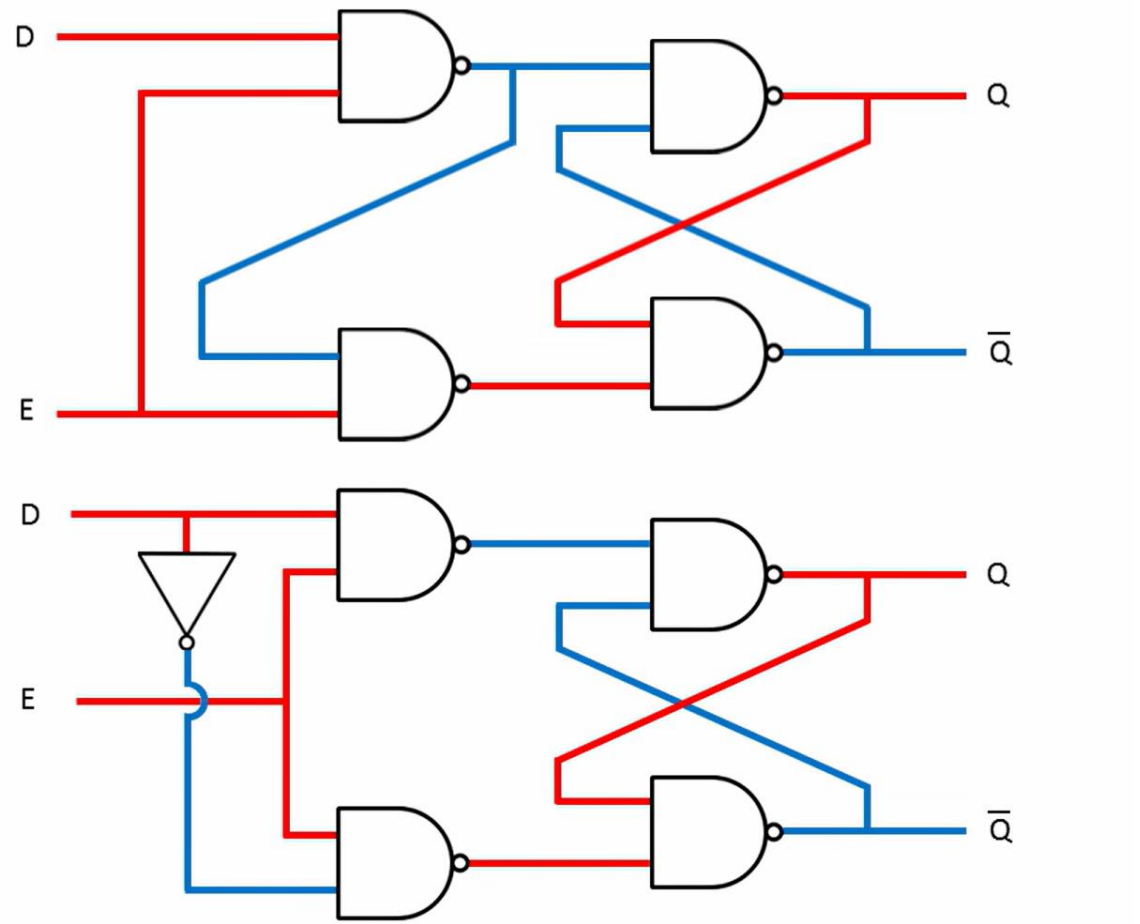
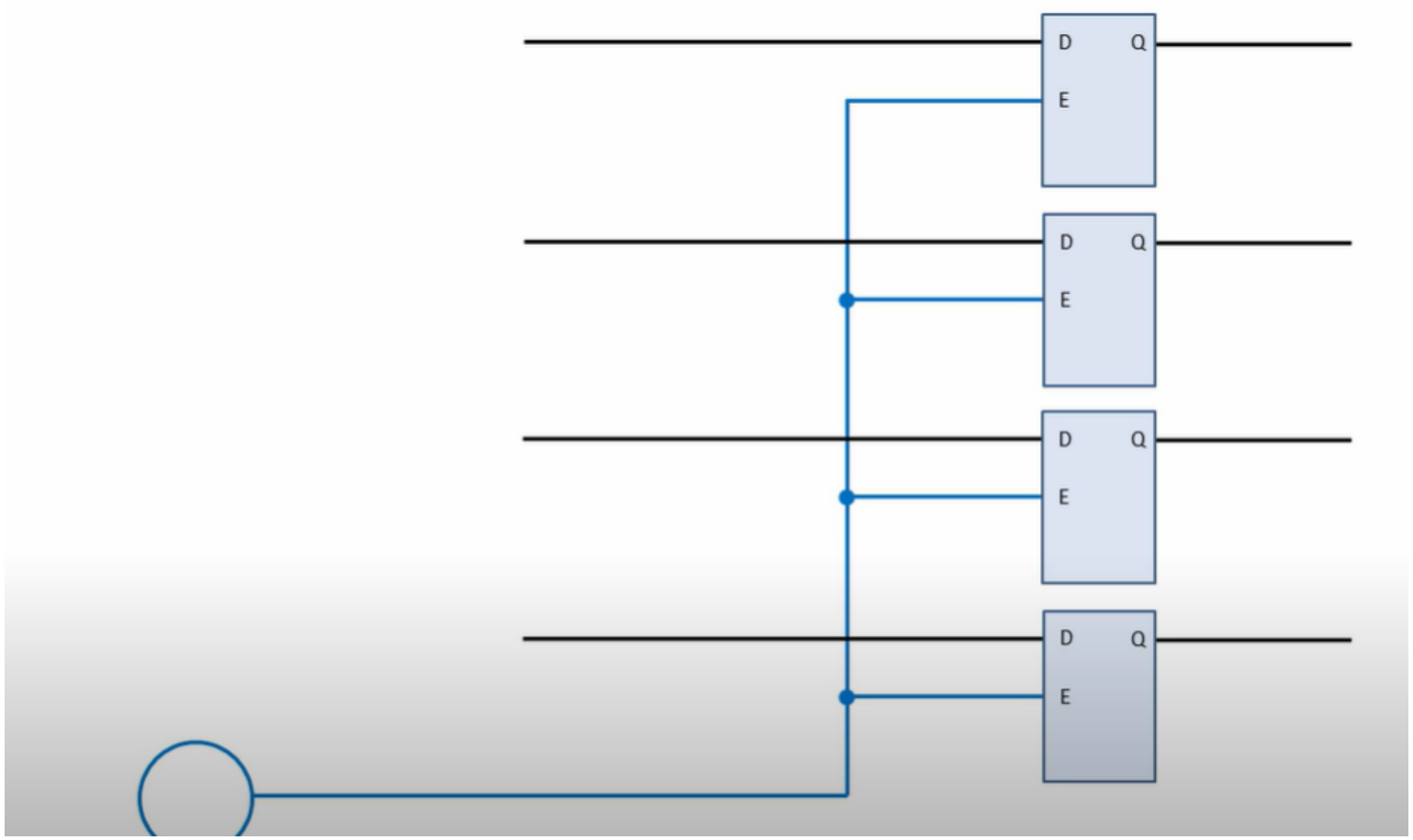
D Flip Flops

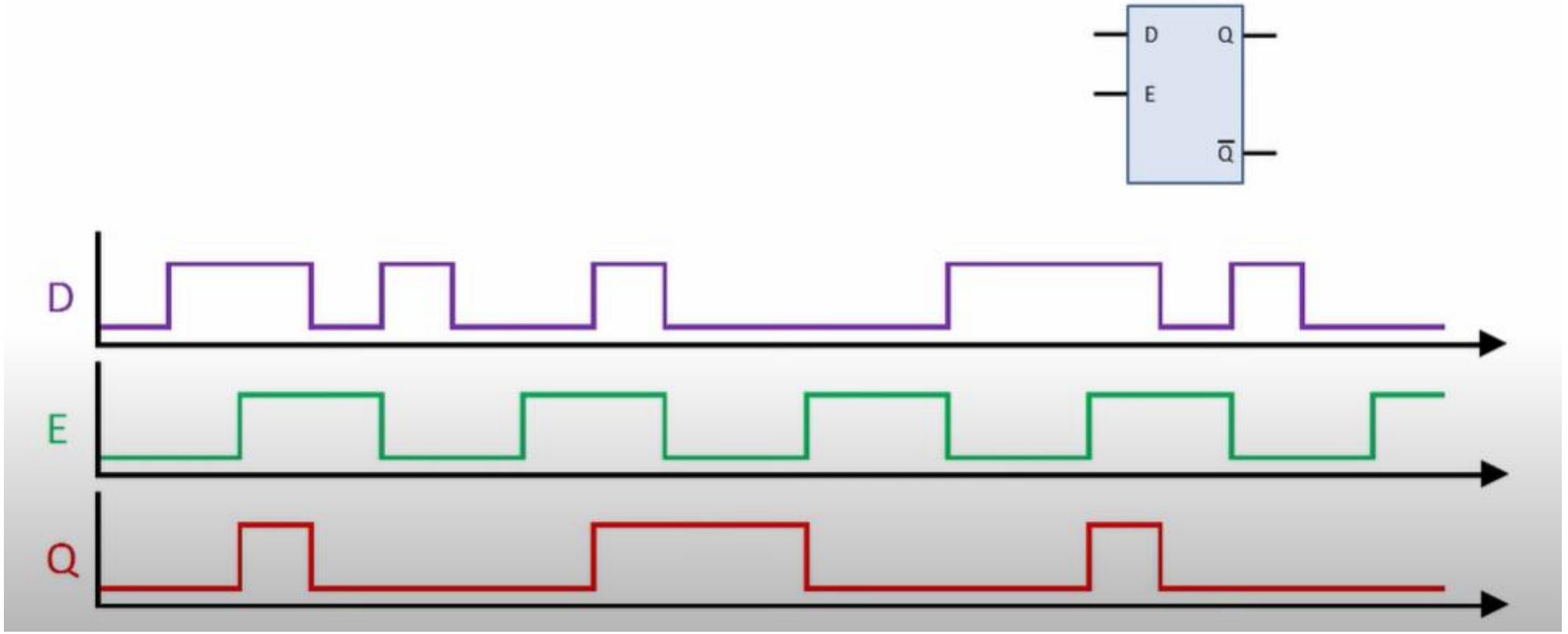
Homework: verify that these designs are equivalent



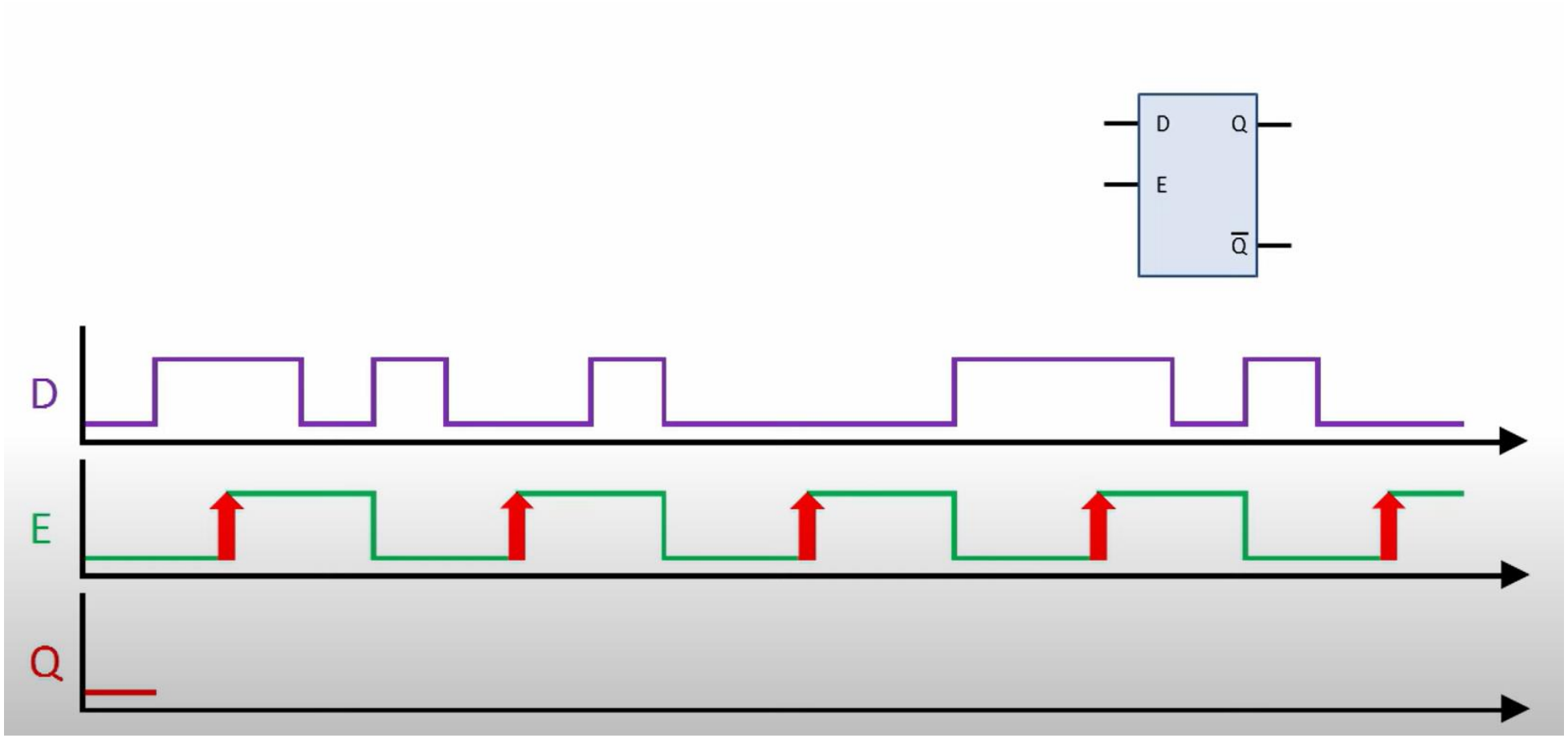
The problem of synchronization



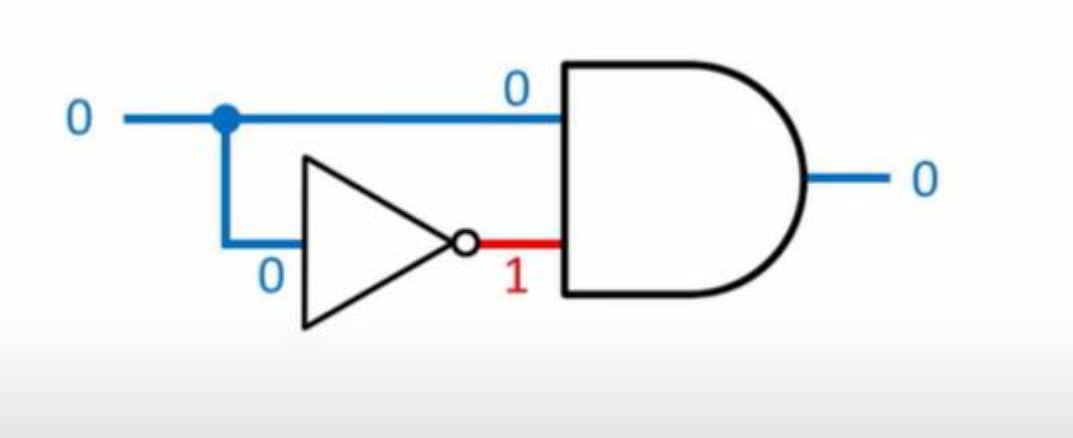
The D latch with a clock for enable

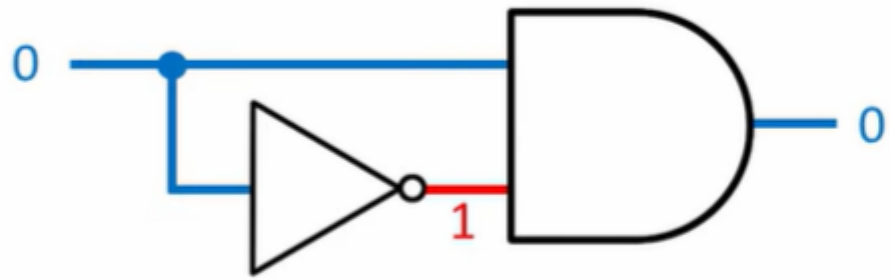


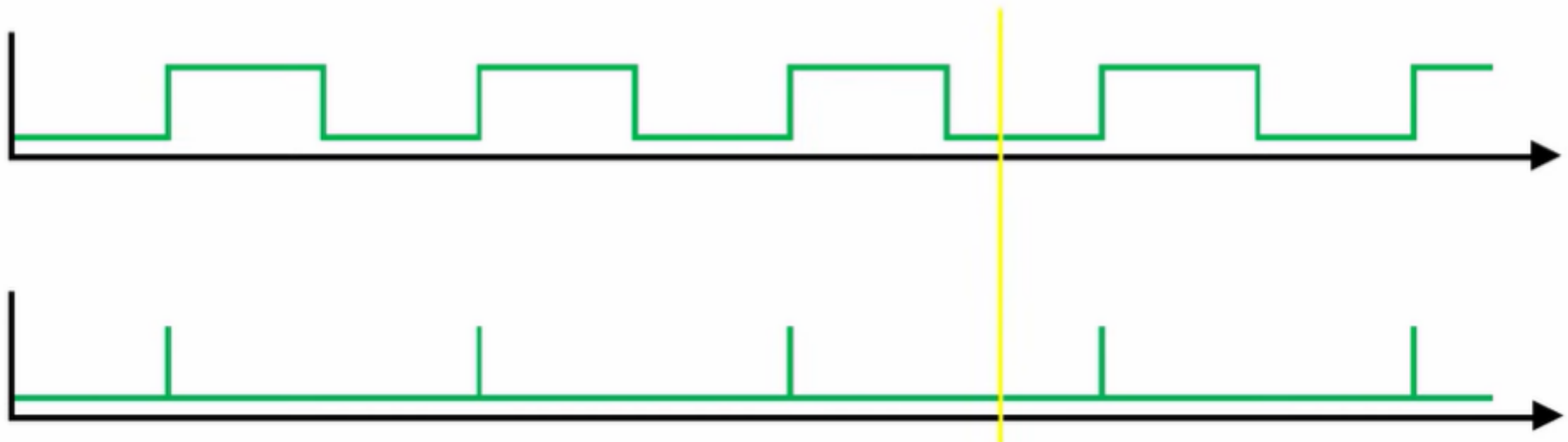
Clocked D Latch



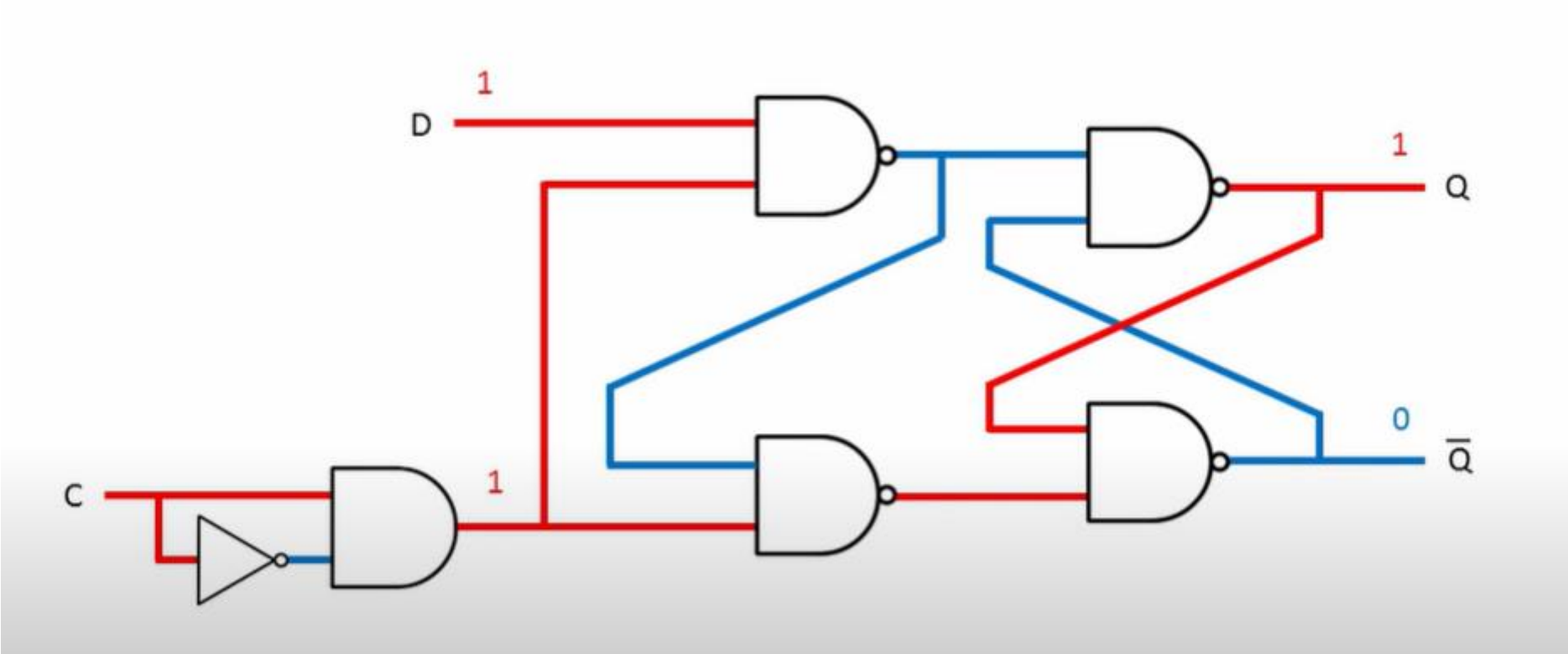
Edge detector

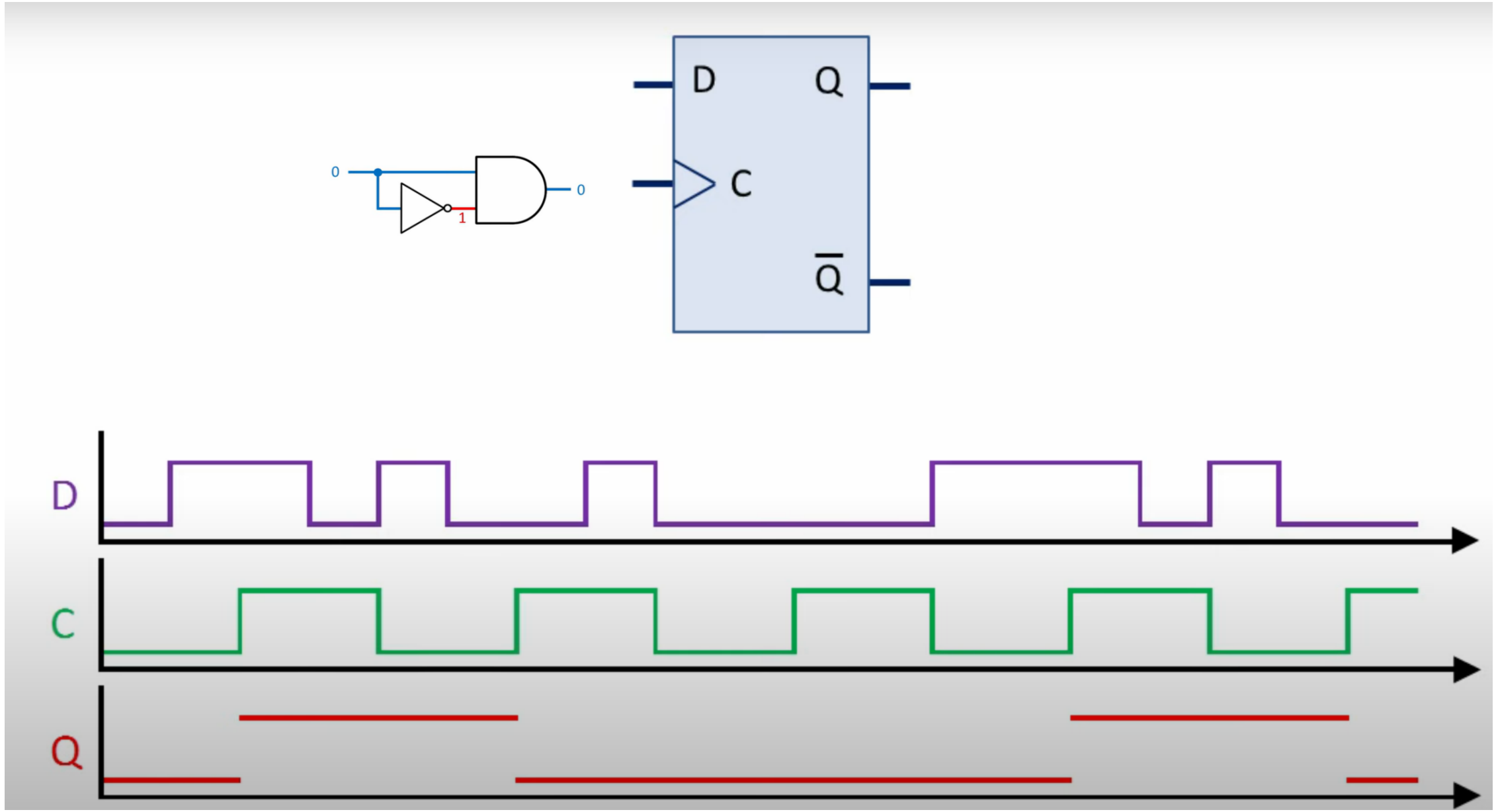



0
1
0

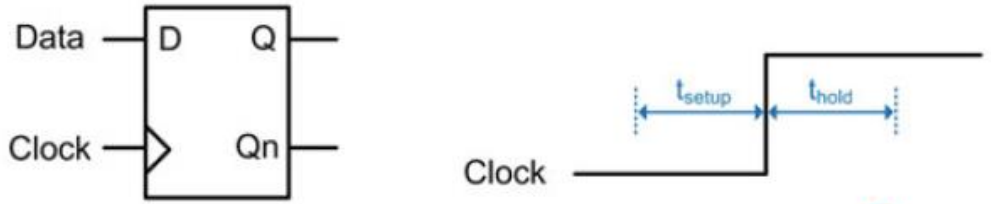


The Clocked D Latch

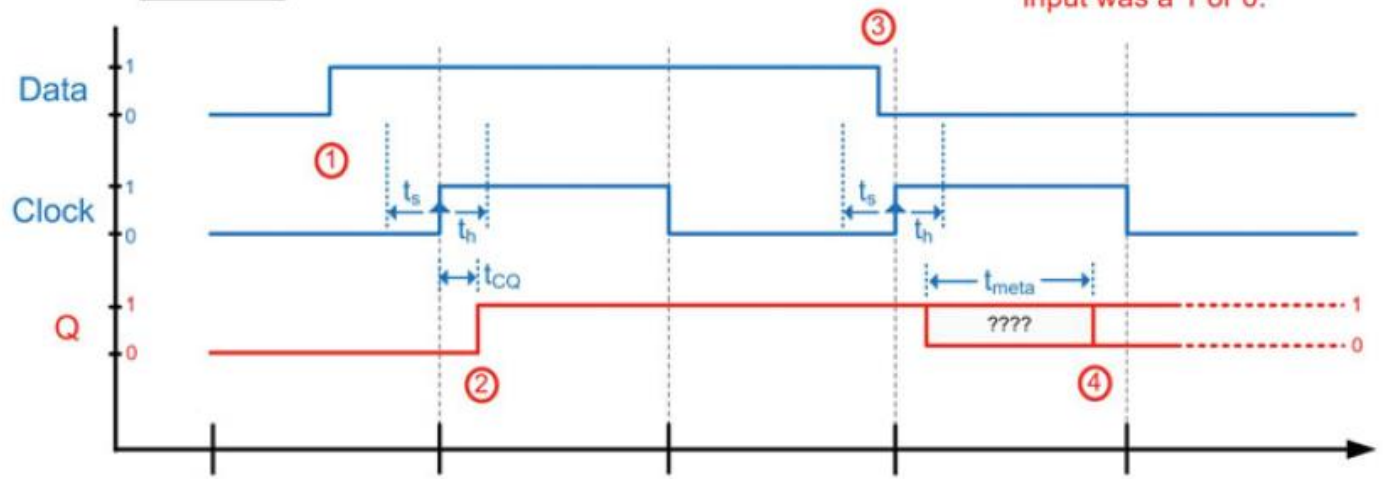



0
1
0
D
Q
C
$\overline{Q}$
D
C
Q

Sequential Storage Device Timing Specifications



The data cannot transition during the setup/hold timing specifications or the device will not be able to determine whether the input was a 1 or 0.



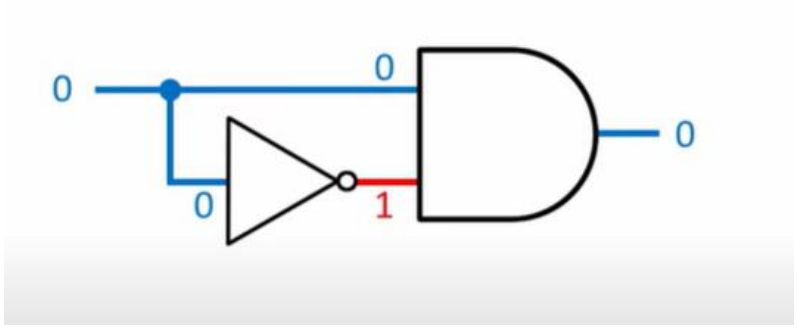
① The first transition on Data from a 0 to a 1 meets the setup/hold specifications for the D-Flip-Flop. This allows the device to successfully latch in the correct value.

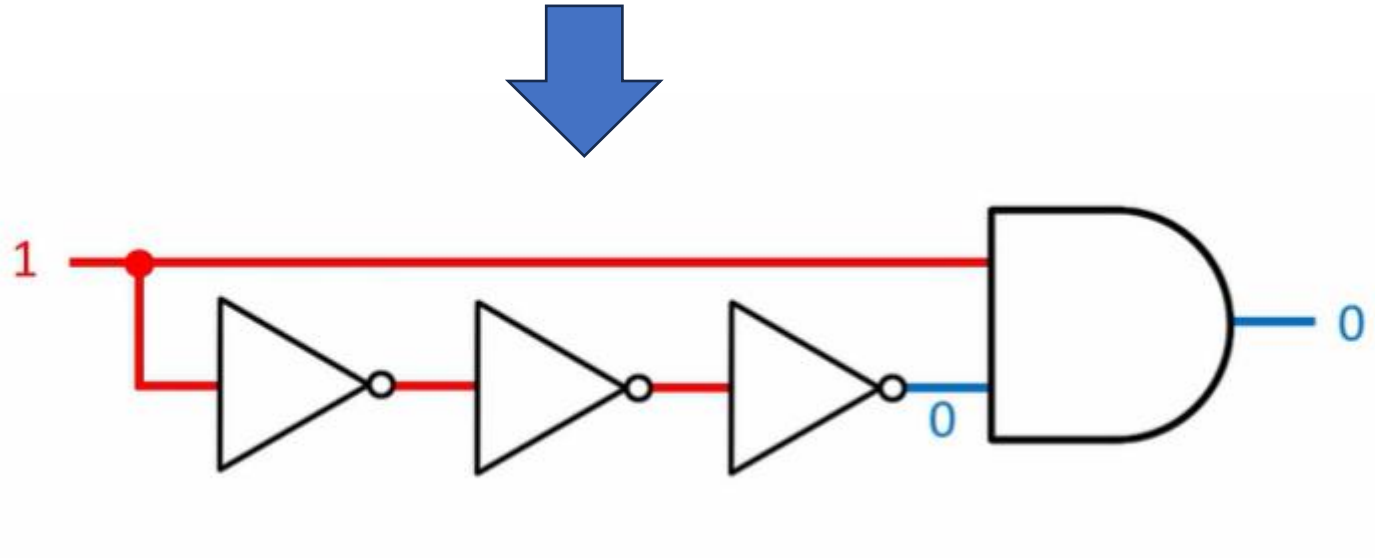
② The value of Data will show up on Q after the t_{CQ} delay of the D-Flip-Flop.

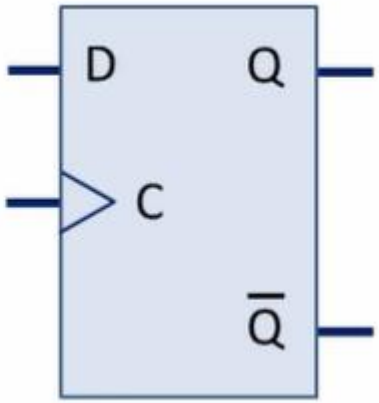
③ The second transition on Data from a 1 to a 0 violates the setup/hold specifications for the D-Flip-Flop. This sends the device into metastability. The D-Flip-Flop will remain metastable for t_{meta}. During this time, the value of the output is unknown. It may go to a steady state 1, a steady state 0 or toggle uncontrollably.

④ After coming out of its metastable state, the D-Flip-Flop output will go to one of two stable states, Q=0 or Q=1. The final resting state is random and unknown.

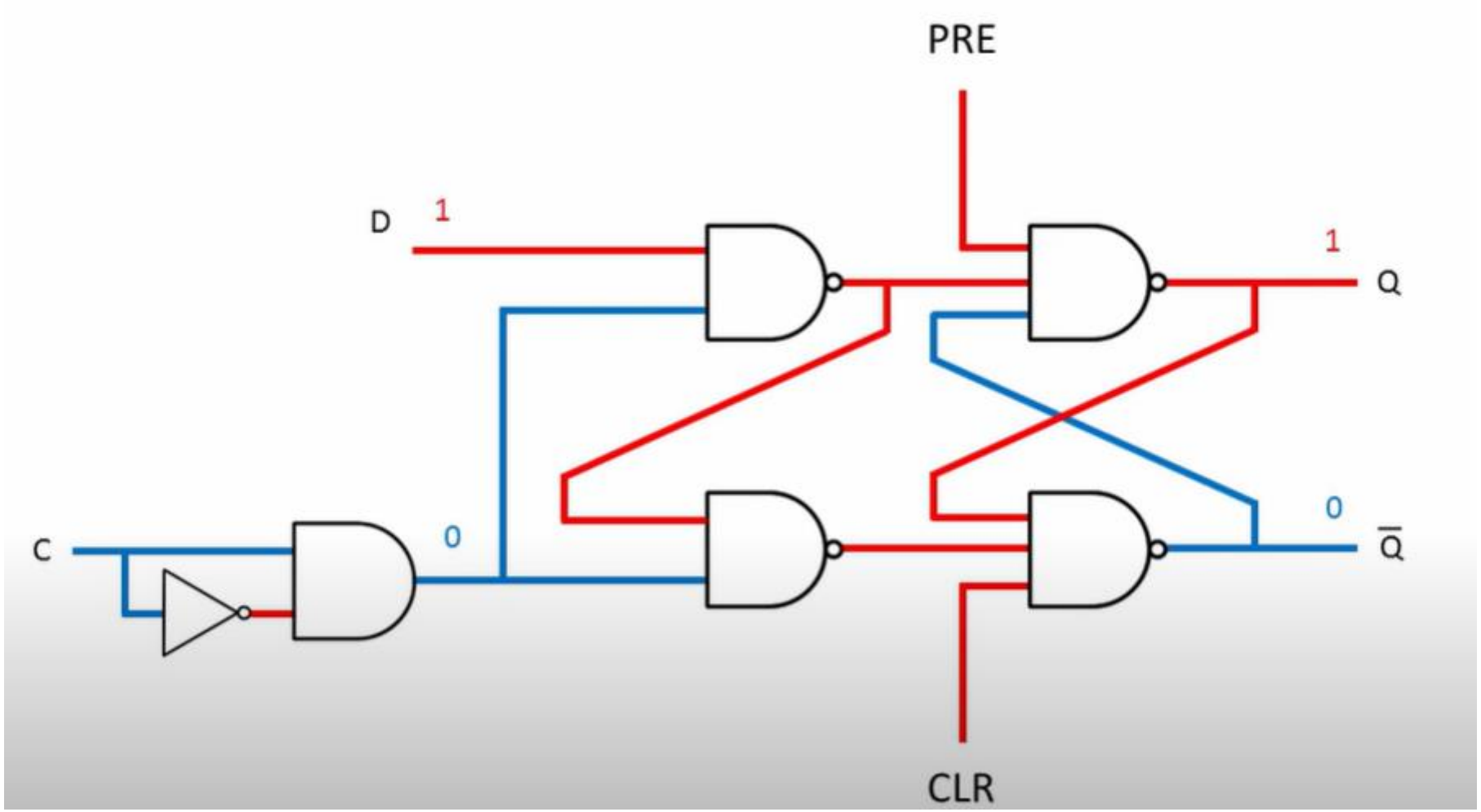
Improvisations



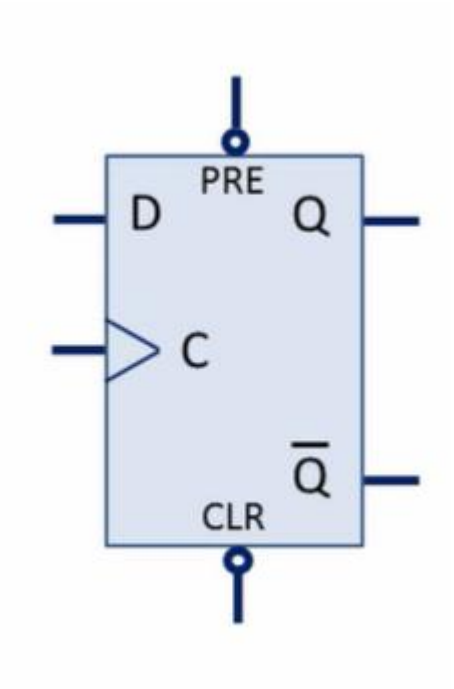




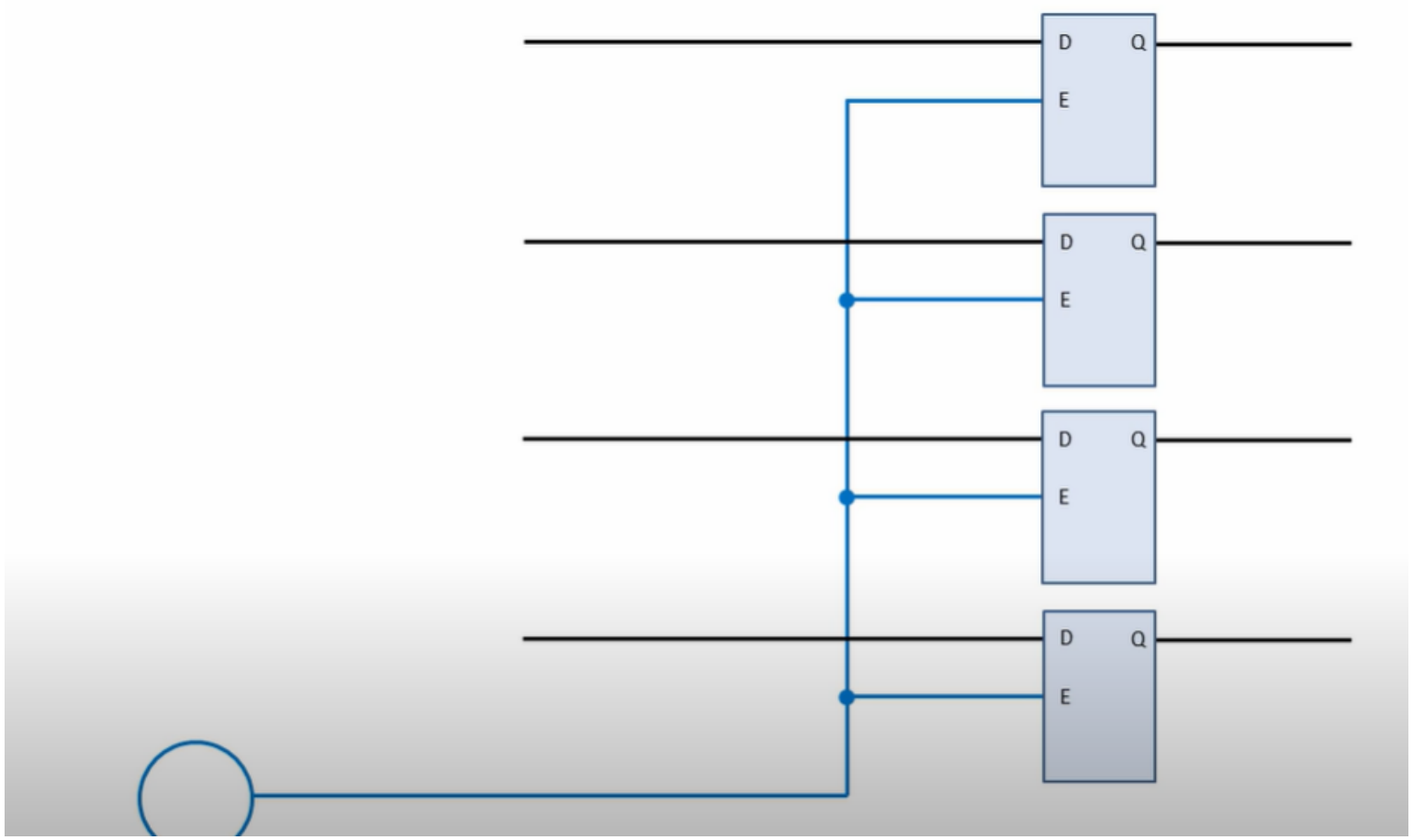
Improvisations : Overriding the clock

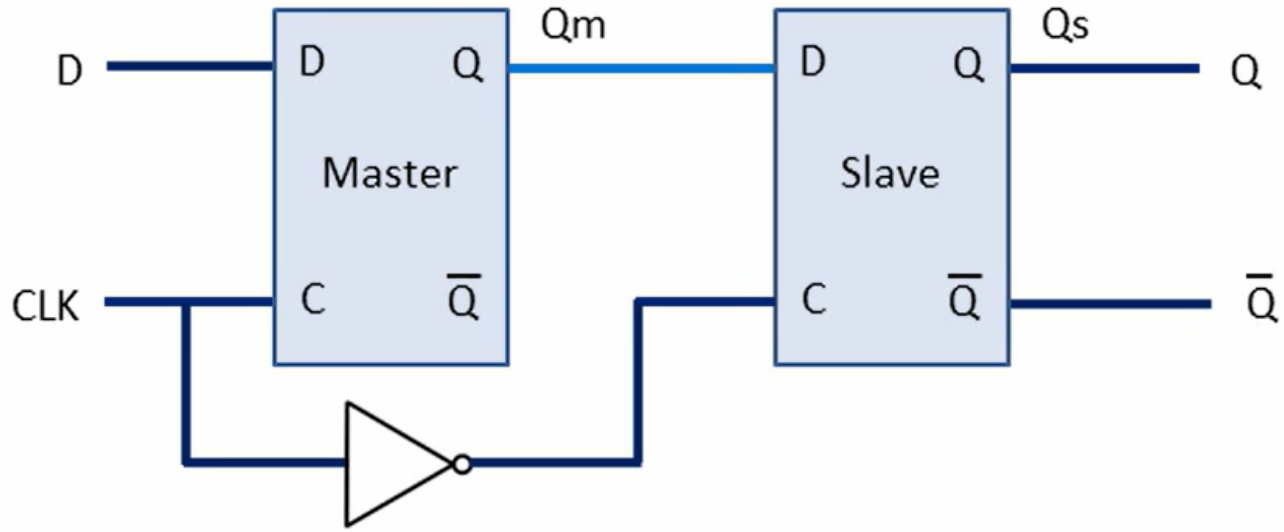


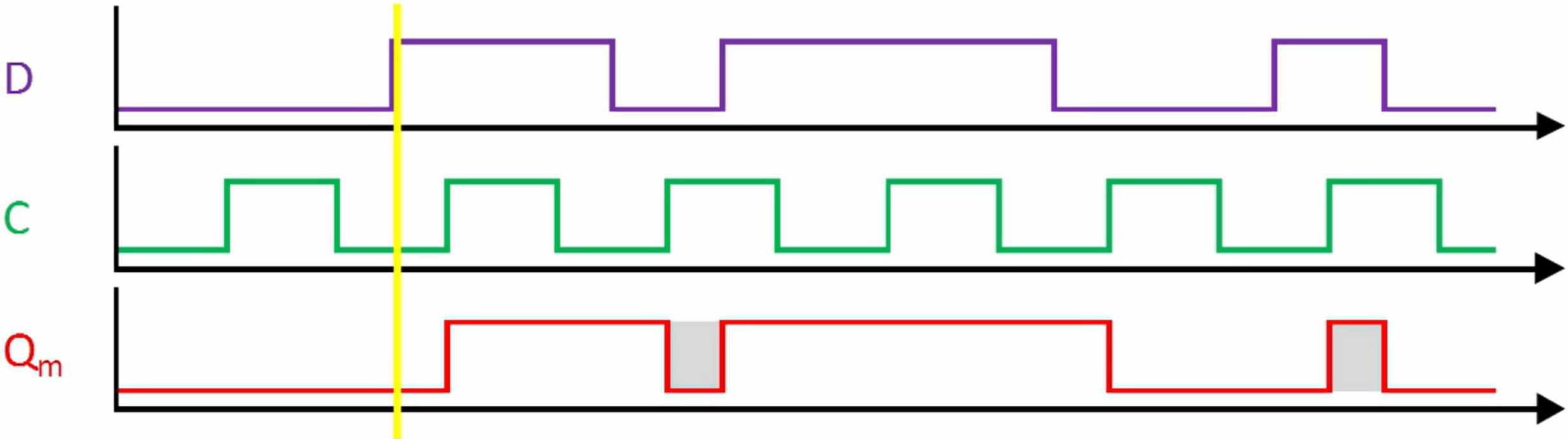
A Clocked D Latch with Preset and Clear

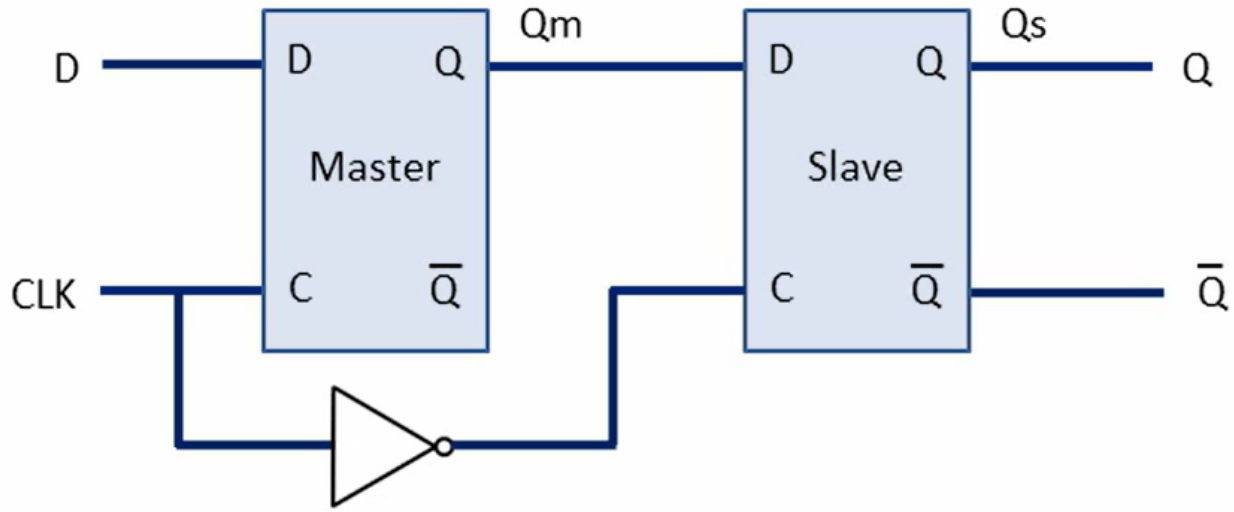


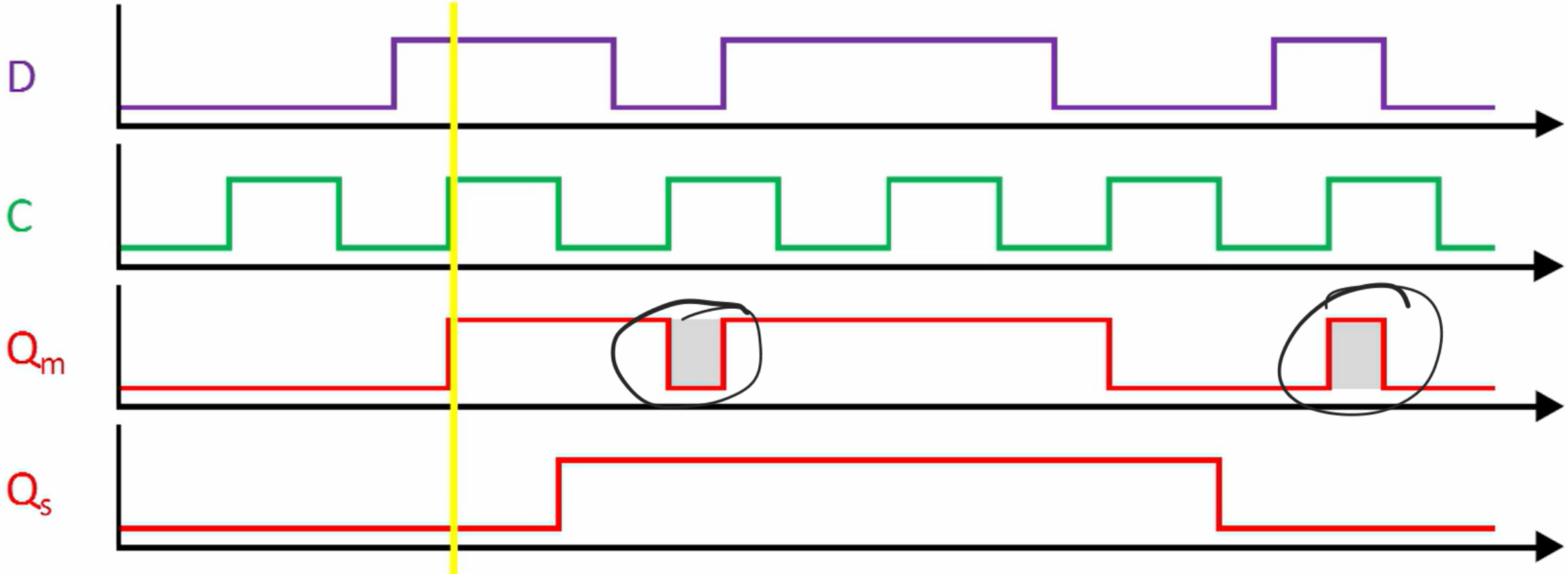
The problem of synchronization



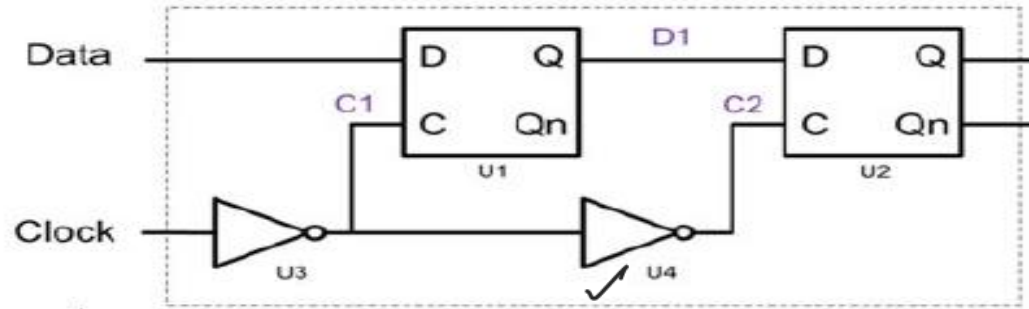

D
CLK
D
Q
Master
C
$\overline{Q}$
Qm
D
Q
Slave
C
$\overline{Q}$
Qs
Q
$\overline{Q}$

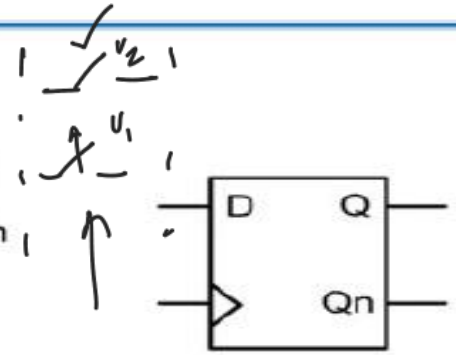

D
C
Q_m

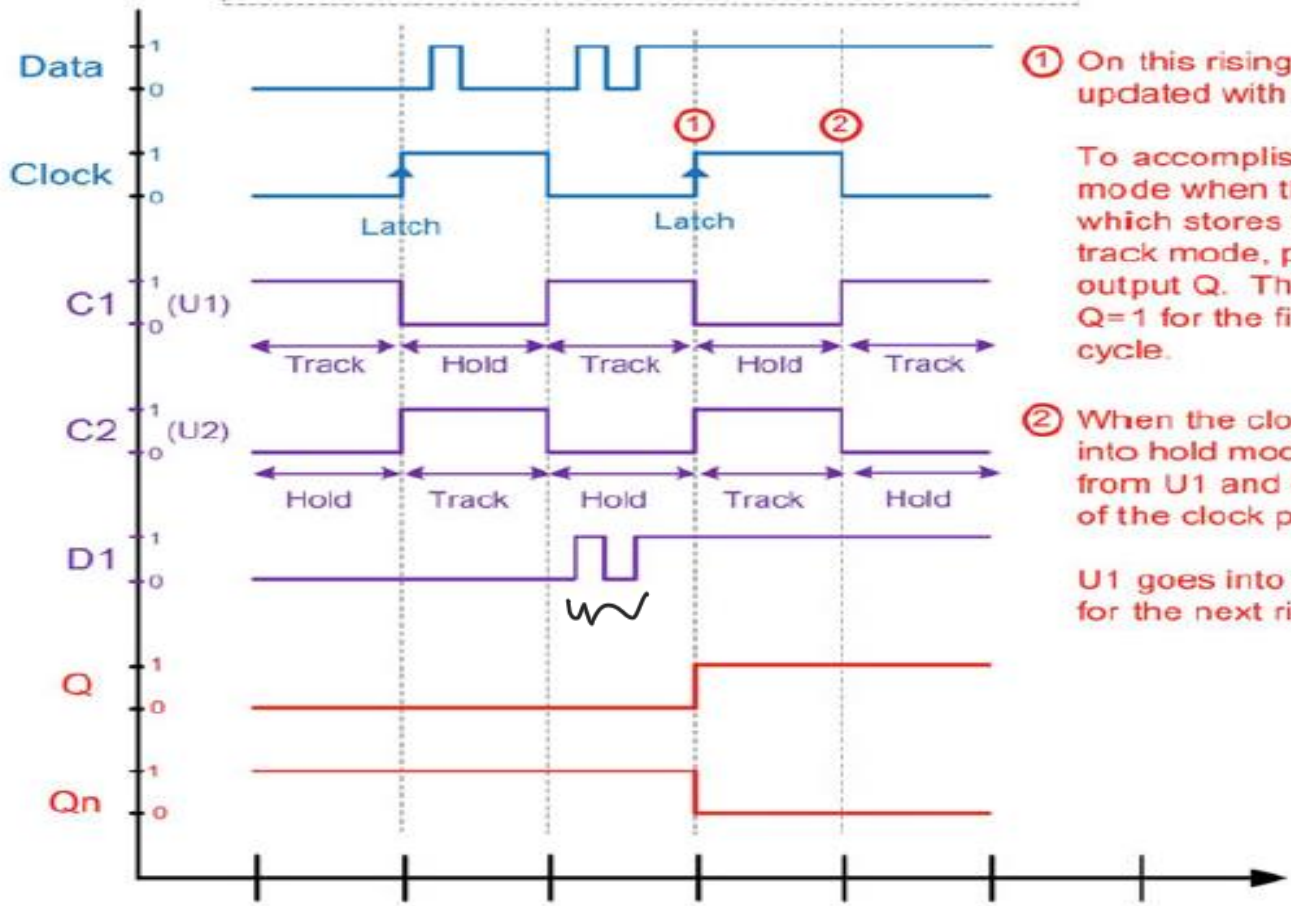

D
D
Q
Qm
Master
C
Q
CLK
D
Q
Qs
Slave
C
Q
Q
Q


D
C
Qm
Qs

D-Flip-Flop (Rising Edge Triggered) Timing Diagram







① On this rising edge of clock, Q is updated with the value of D (Q=1).

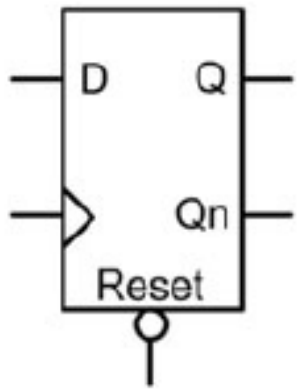
To accomplish this, U1 goes into hold mode when the clock goes HIGH, which stores the input 1. U2 goes into track mode, passing the 1 to the output Q. This configuration keeps Q=1 for the first part of the clock cycle.

② When the clock goes LOW, U2 goes into hold mode, which stores the 1 from U1 and drives Q=1 for the rest of the clock period.

U1 goes into track mode to get ready for the next rising edge of the clock.

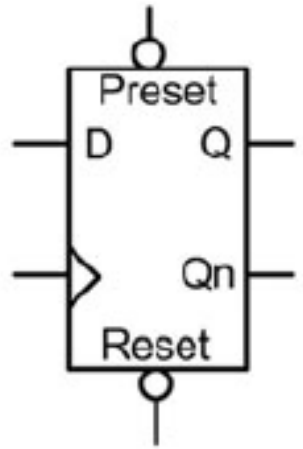
D-Flip-Flop with Asynchronous Reset and Preset

D-Flip-Flop with Active LOW Reset



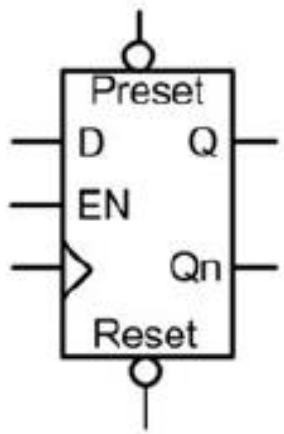
$\overline{R}$	Clk	D	Q	Qn	
0	X	X	0	1	Reset
1	0	X	Last Q	Last Qn	Store
1	1	X	Last Q	Last Qn	Store
1	↓	0	0	1	Update
1	↓	1	1	0	Update

D-Flip-Flop with Active LOW Reset and Active LOW Preset



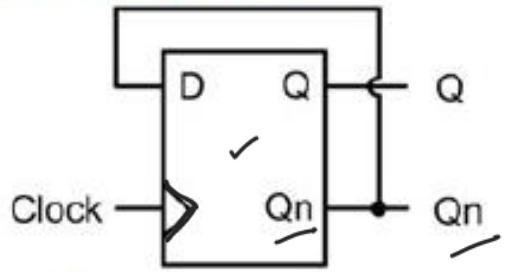
$\overline{R}$	$\overline{P}$	Clk	D	Q	Qn	
0	X	X	X	0	1	Reset
1	0	X	X	1	0	Preset
1	1	0	X	Last Q	Last Qn	Store
1	1	1	X	Last Q	Last Qn	Store
1	1	↓	0	0	1	Update
1	1	↓	1	1	0	Update

D-Flip-Flop with Synchronous Enable

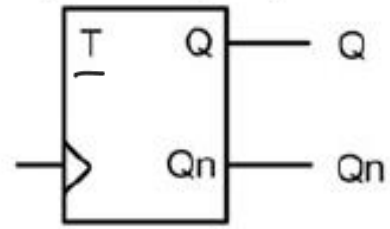


$\overline{R}$	$\overline{P}$	Clk	EN	D	Q	Qn	
0	X	X	X	X	0	1	Reset
1	0	X	X	X	1	0	Preset
1	1	0	X	X	Last Q	Last Qn	Store
1	1	1	X	X	Last Q	Last Qn	Store
1	1	↓	0	X	Last Q	Last Qn	Disabled (ignore clock)
1	1	↓	1	0	0	1	Update
1	1	↓	1	1	1	0	Update

Toggle Flop Clock Frequency Divider

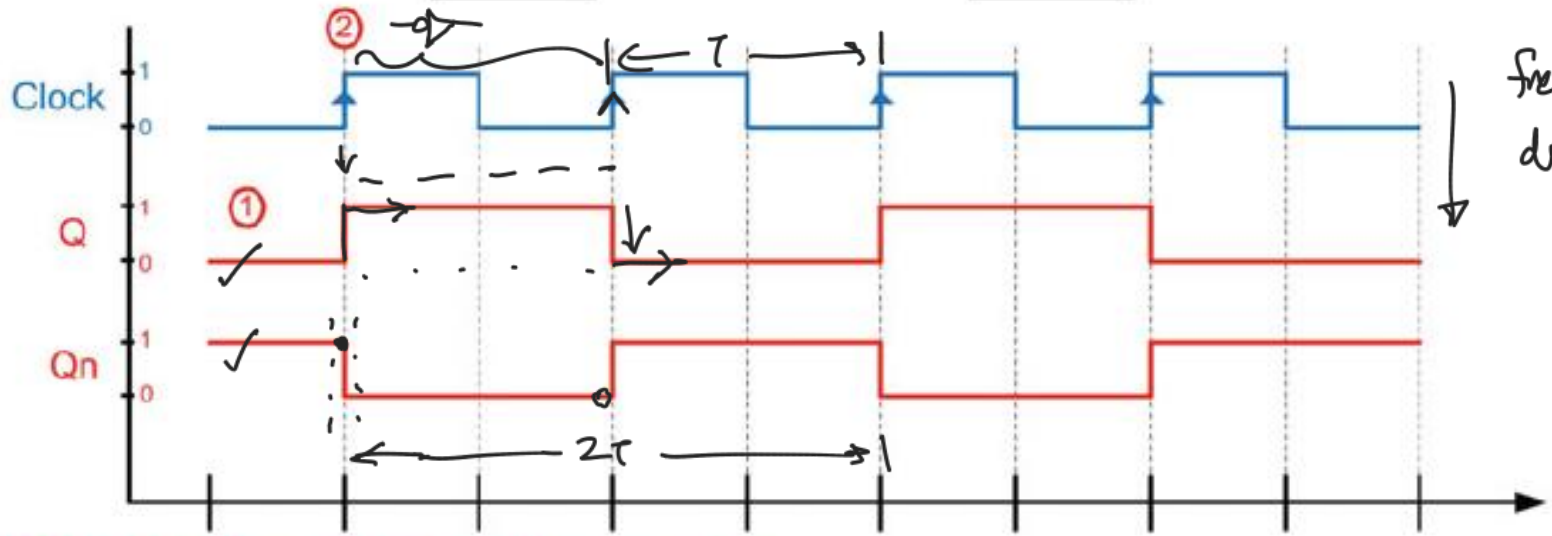


Optional Symbol for a Toggle-Flop or "T-Flip-Flop"



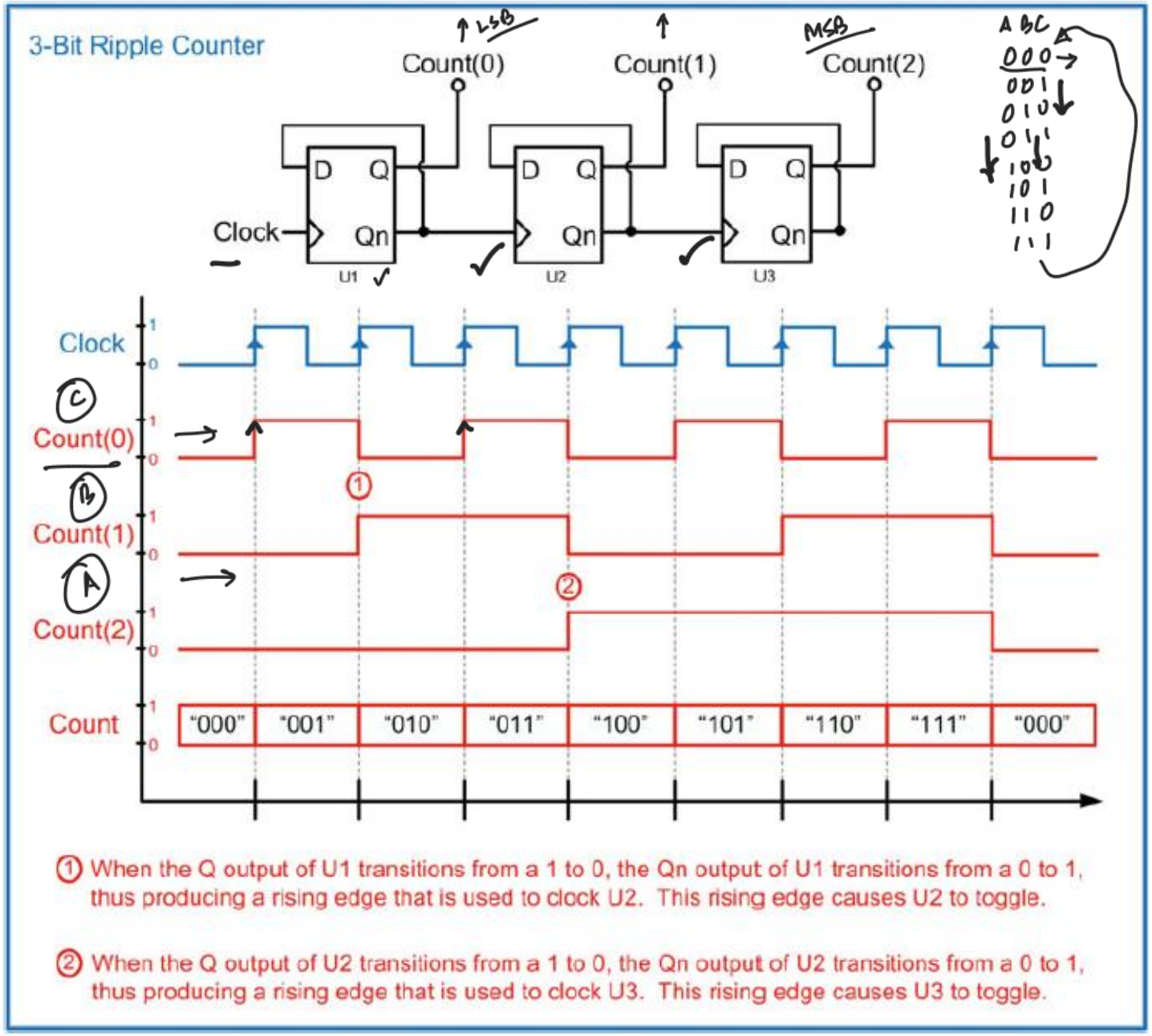
T-

$f = \frac{1}{T}$



① Q and Qn will always be at opposite logic values.

② When a rising edge of a clock occurs, Q will be updated with the value present on D. Since in this configuration the value on D will always be the opposite of the current value of Q, the outputs will toggle. The outputs will change, or toggle, every time there is a rising edge on the clock. This has the effect of creating a square wave on the output that has ½ the frequency of the clock.


3-Bit Ripple Counter
LSB
MSB
Count(0)
Count(1)
Count(2)
D
Q
Qn
Clock
U1
U2
U3
A BC
000
001
010
011
100
101
110
111
Clock
Count(0)
Count(1)
Count(2)
Count
"000"
"001"
"010"
"011"
"100"
"101"
"110"
"111"
"000"
① When the Q output of U1 transitions from a 1 to 0, the Qn output of U1 transitions from a 0 to 1, thus producing a rising edge that is used to clock U2. This rising edge causes U2 to toggle.
② When the Q output of U2 transitions from a 1 to 0, the Qn output of U2 transitions from a 0 to 1, thus producing a rising edge that is used to clock U3. This rising edge causes U3 to toggle.

Two Problems

- Analysis
- Design

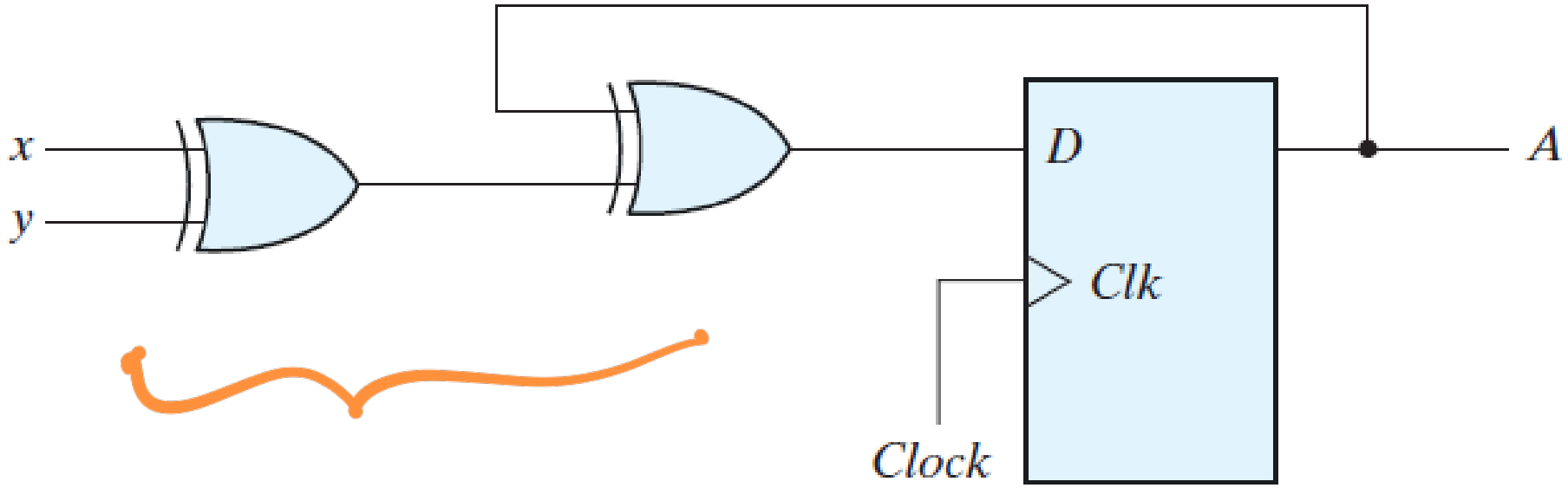
Ack for slides: Textbook by Mano and Ciletti

Analysis of Sequential Circuits

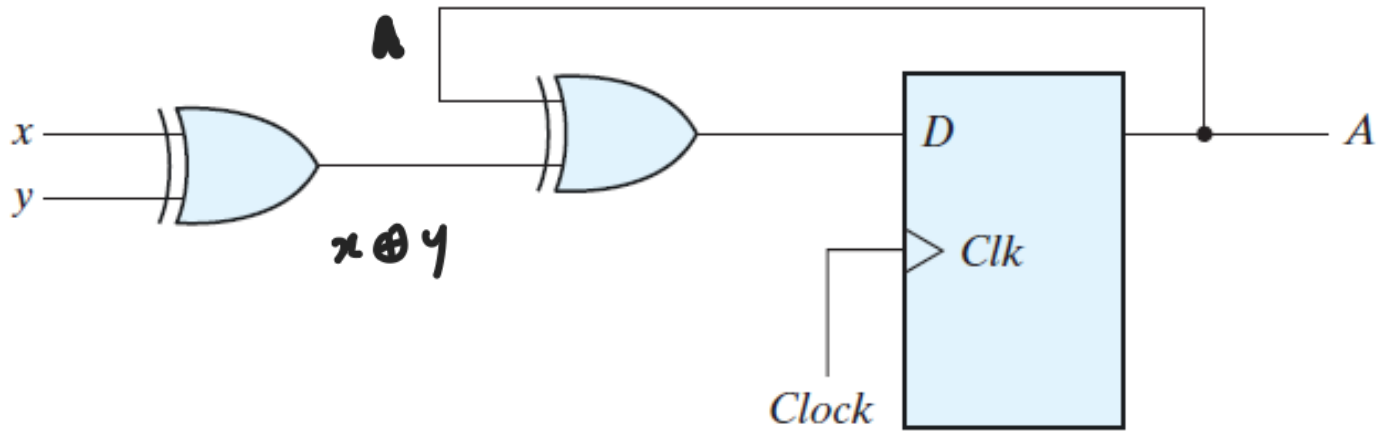
- Steps :

Circuit diagram $\rightarrow$ Equations – State table $\rightarrow$ State diagram

Step 1: Circuit



Step 2: Equations



Input Equation:

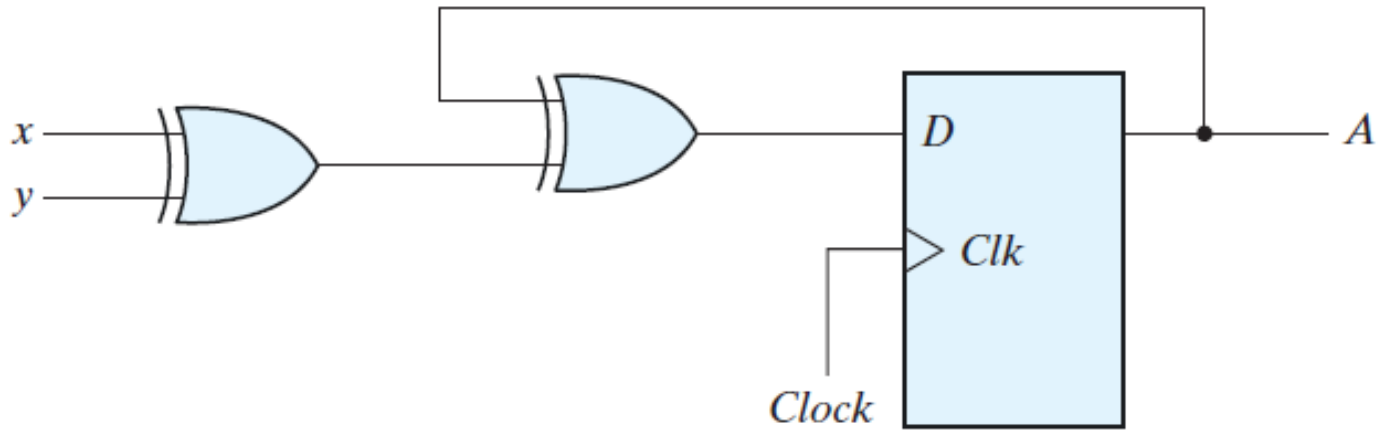
$$D_A = A \oplus x \oplus y$$

State Equation:

$$A(t+1) = A \oplus x \oplus y$$

A(t)

Step 3: State Transition Table



Present state	Inputs		Next state
A	x	y	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

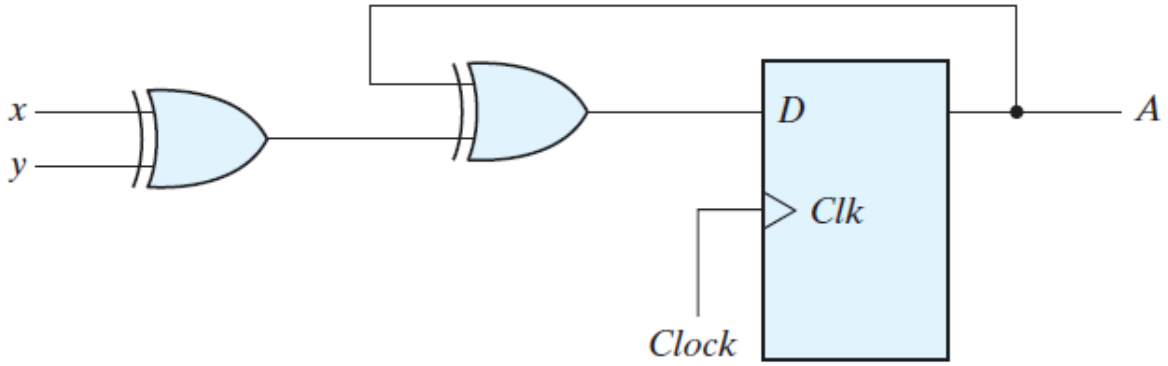
Input Equation:

$$D_A = A \oplus x \oplus y$$

State Equation:

$$A(t+1) = A \oplus x \oplus y$$

Step 4: State Transition Diagram



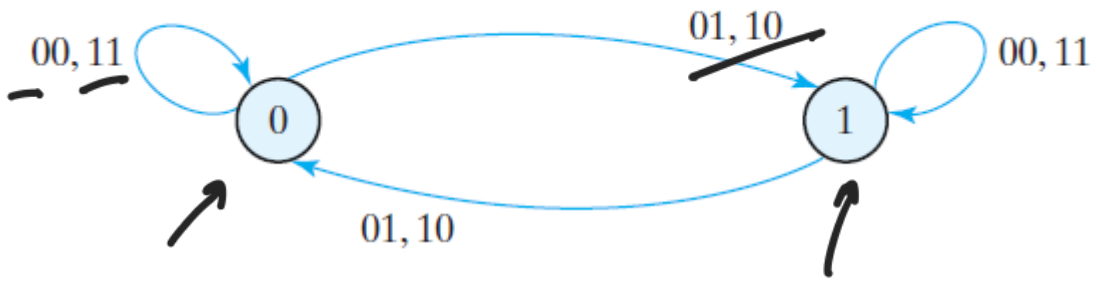
Input Equation:

$$D_A = A \oplus x \oplus y$$

State Equation:

$$A(t+1) = A \oplus x \oplus y$$

Present state	Inputs		Next state
A	x	y	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



Design Problem

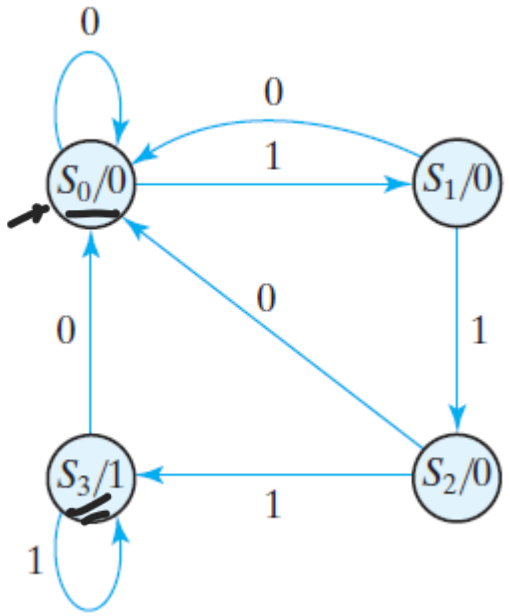
- Steps:
    - Word Description
    - State Transition Diagram
    - State Transition Table
    - K-maps for Sequence Detector
    - Logic/Circuit Diagram

Step 1: Word Description

We wish to design a circuit that detects a sequence of three or more consecutive 1's in a string of bits coming through an input line (i.e., the input is a *serial bit stream*).

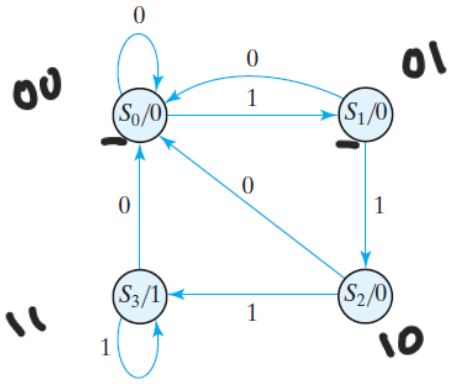
Step 2: State Transition Diagram

We wish to design a circuit that detects a sequence of three or more consecutive 1's in a string of bits coming through an input line (i.e., the input is a *serial bit stream*).



Step 3: State Transition Table

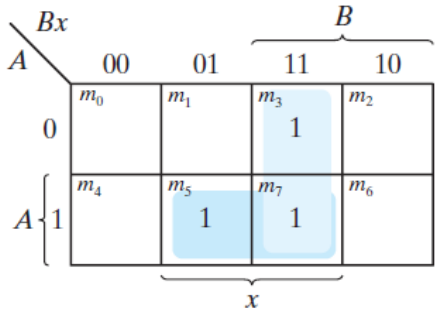
We wish to design a circuit that detects a sequence of three or more consecutive 1's in a string of bits coming through an input line (i.e., the input is a *serial bit stream*).



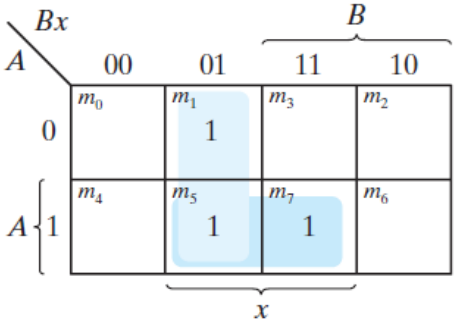
Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Step 4: K-Maps

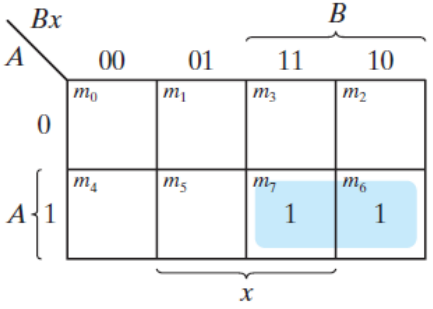
We wish to design a circuit that detects a sequence of three or more consecutive 1's in a string of bits coming through an input line (i.e., the input is a *serial bit stream*).



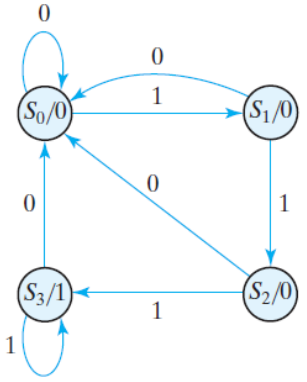
$D_A = Ax + Bx$



$D_B = Ax + B'x$



$y = AB$



Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Step 5: The Logic Diagram of The Sequence Detector

$$D_A = Ax + Bx$$
$$D_B = Ax + B'x$$
$$y = AB$$