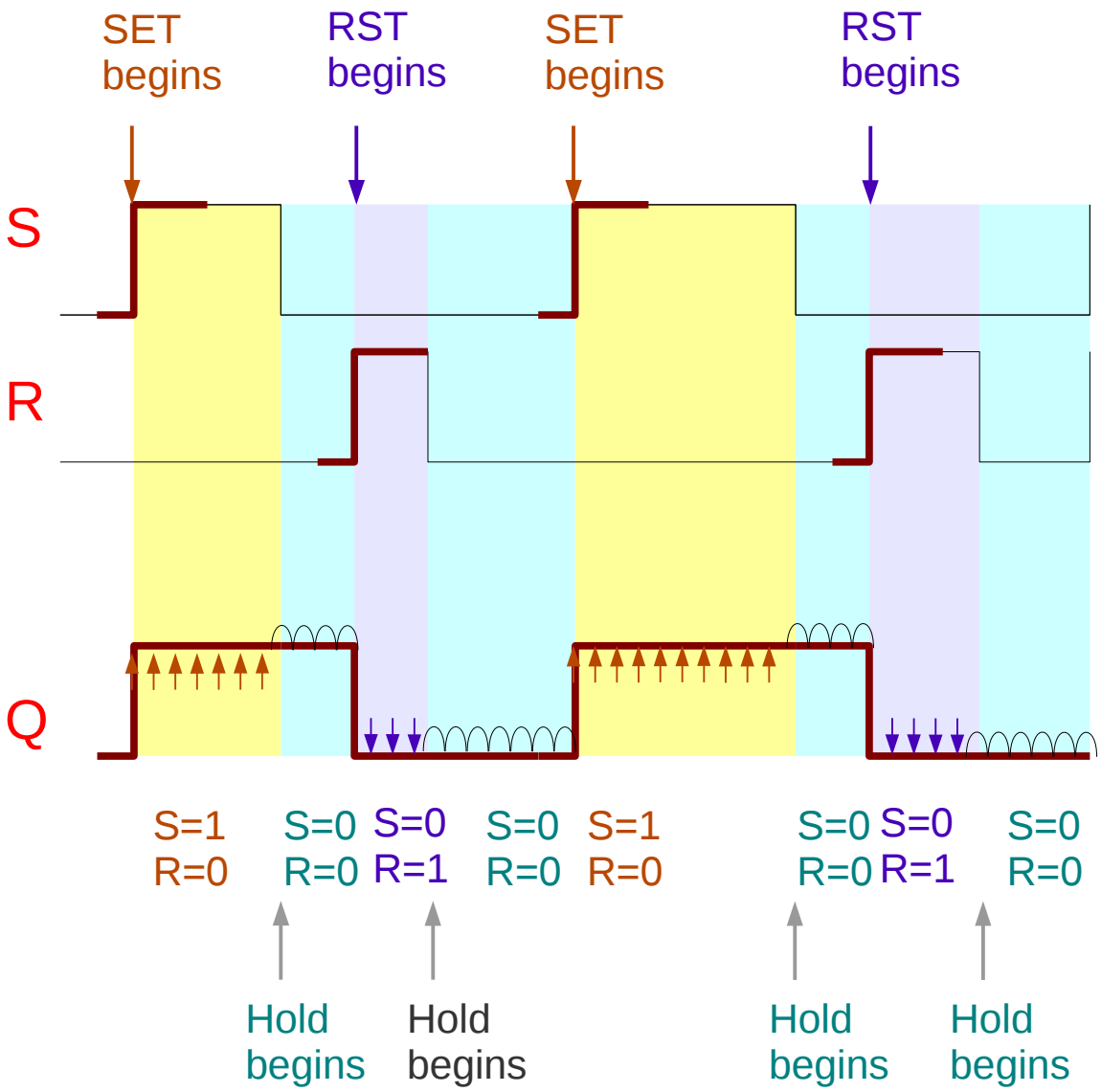
Sequential Circuit Background

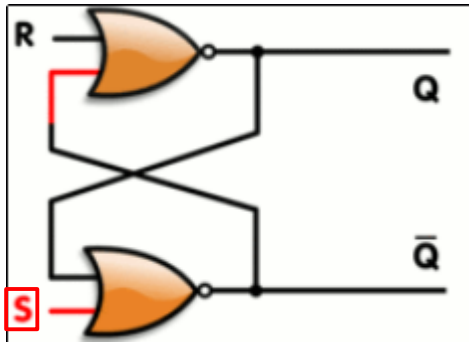
Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Latches and FF's

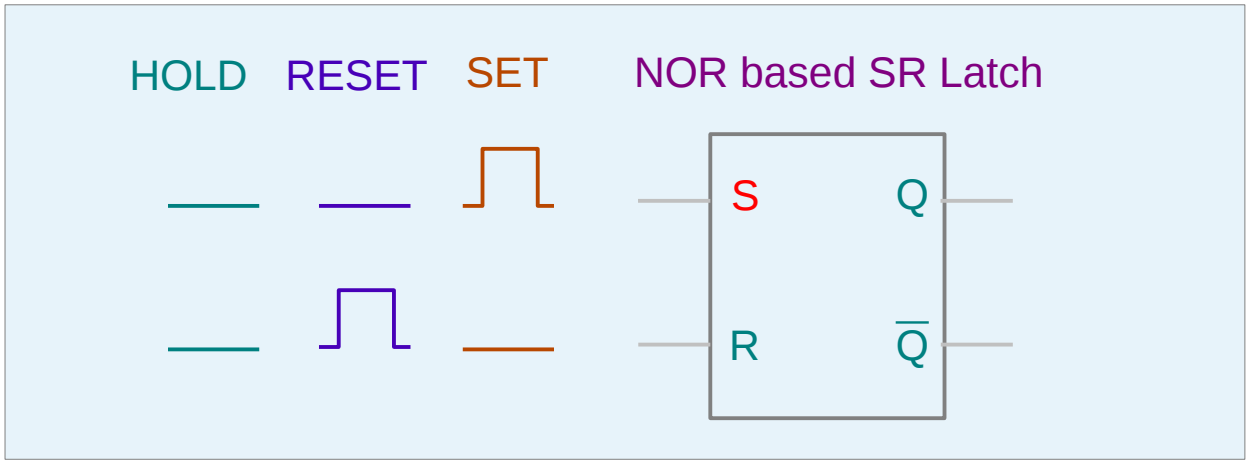
NOR-based SR Latch

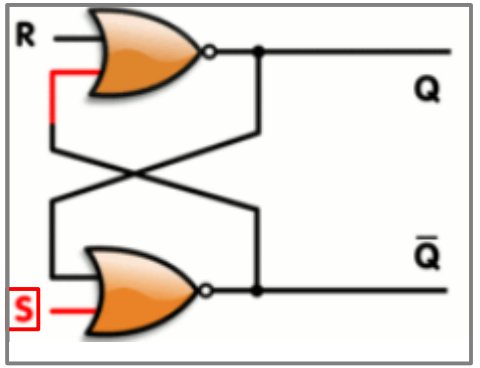


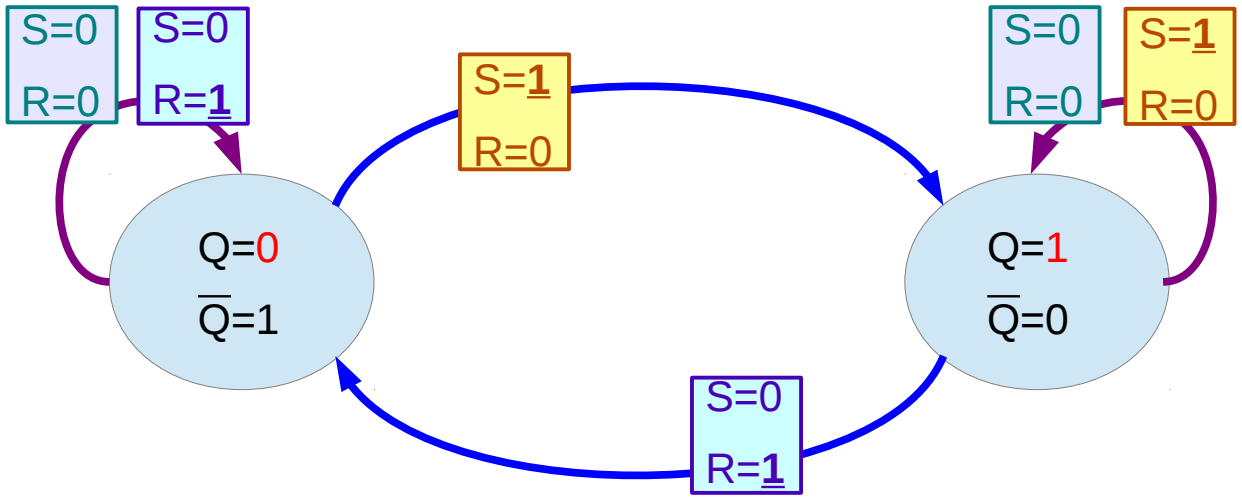


SET	S=1 R=0	Q=1 $\bar{Q}$=0
RESET	S=0 R=1	Q=0 $\bar{Q}$=1
HOLD	S=0 R=0	Q=old Q $\bar{Q}$=old $\bar{Q}$

NOR-based SR Latch States

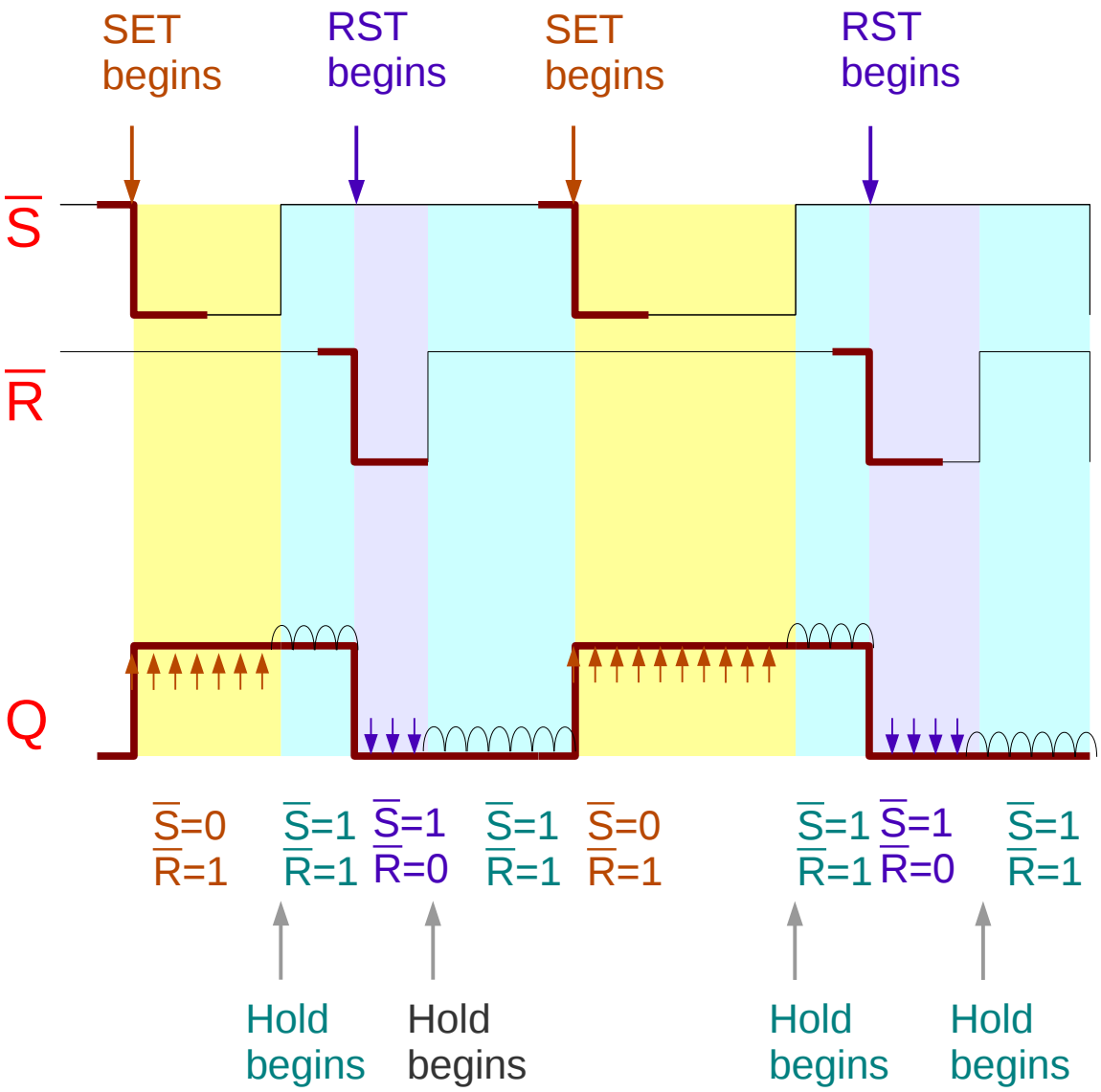


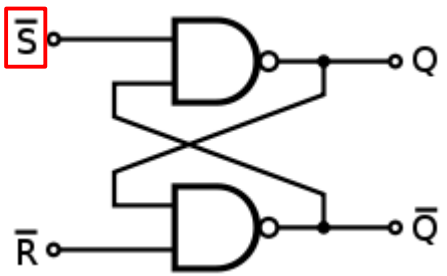




SET	S=1 R=0	Q=1 $\overline{Q}$=0
RESET	S=0 R=1	Q=0 $\overline{Q}$=1
HOLD	S=0 R=0	Q=old Q $\overline{Q}$=old $\overline{Q}$

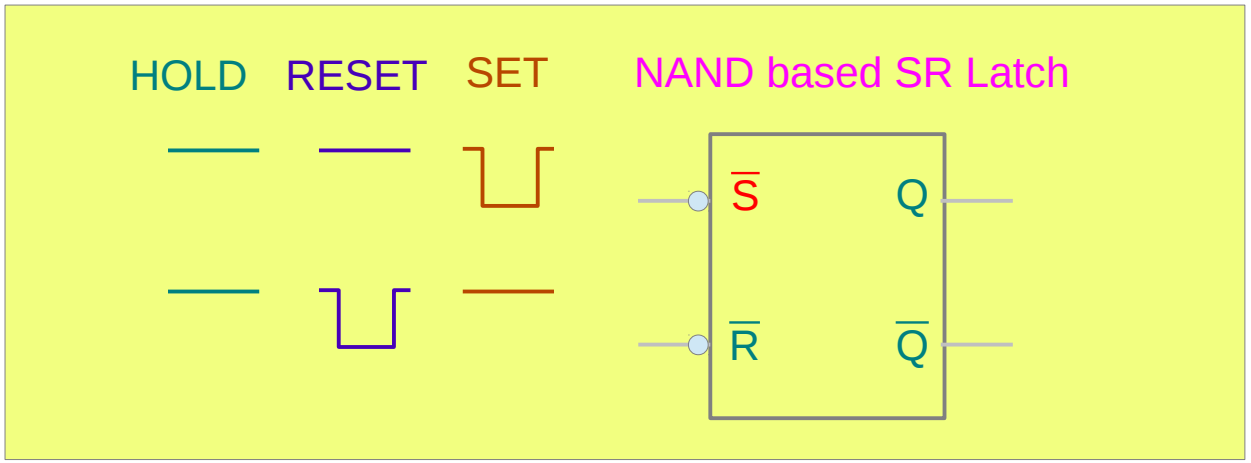
NAND-based SR Latch

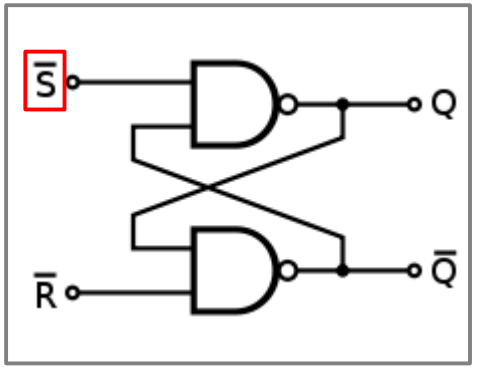


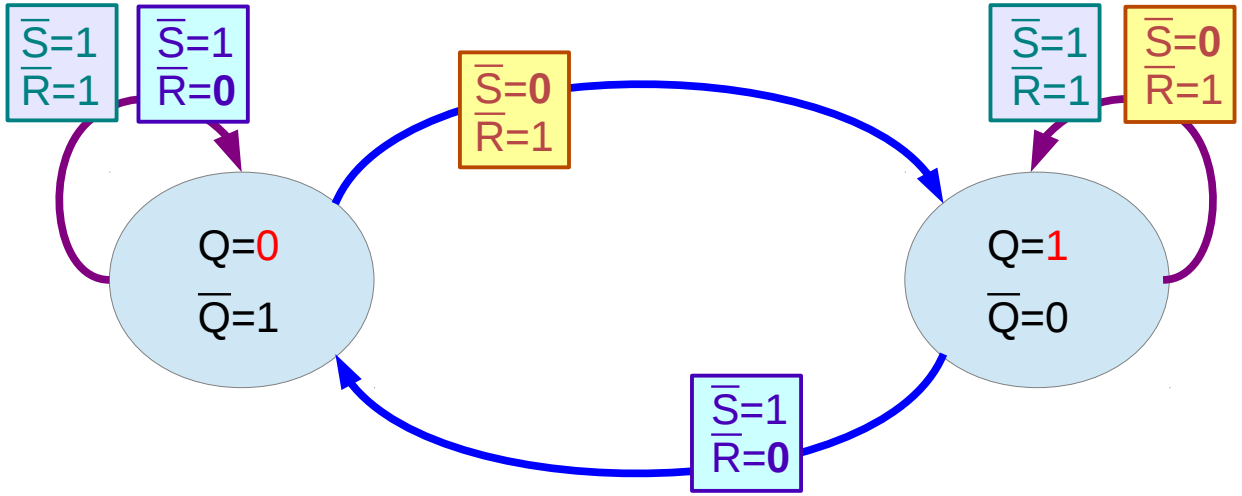


SET	$\overline{S}=0$ $\overline{R}=1$	Q=1 $\overline{Q}=0$
RESET	$\overline{S}=1$ $\overline{R}=0$	Q=0 $\overline{Q}=1$
HOLD	$\overline{S}=1$ $\overline{R}=1$	Q=old Q $\overline{Q}$=old $\overline{Q}$

NAND-based SR Latch States

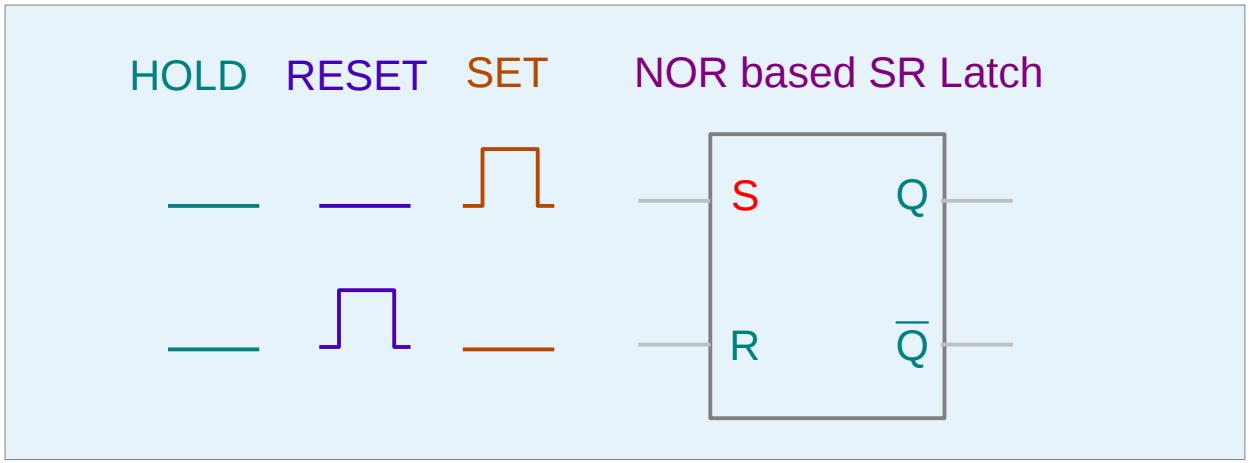


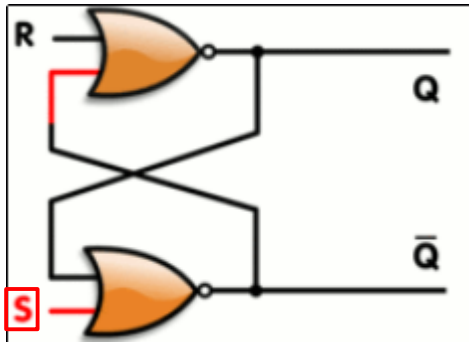


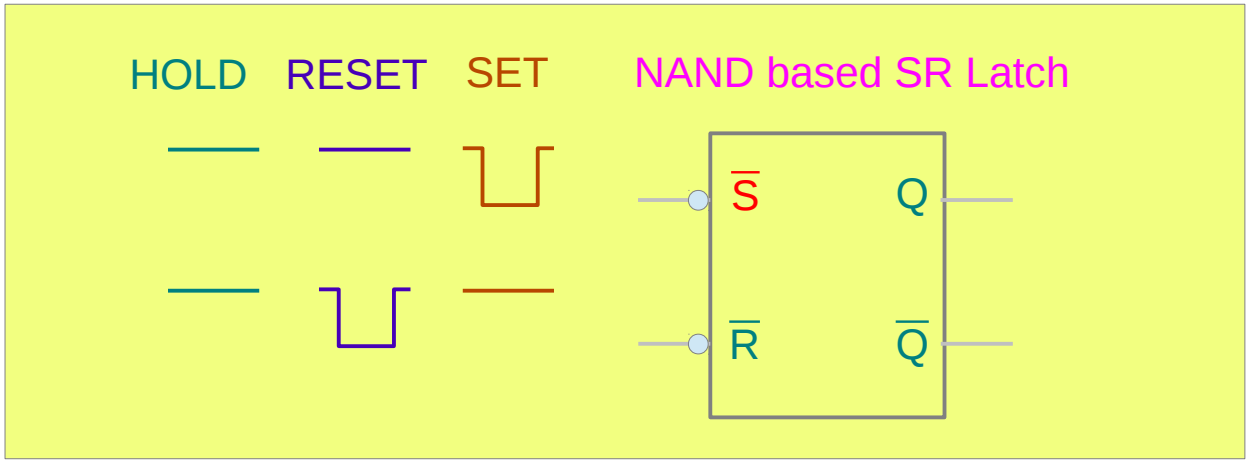


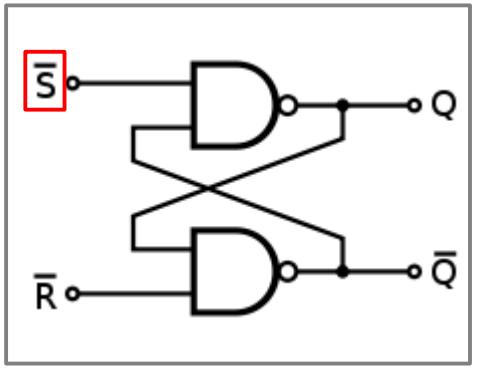
SET	$\overline{S}=0$ $\overline{R}=1$	Q=1 $\overline{Q}=0$
RESET	$\overline{S}=1$ $\overline{R}=0$	Q=0 $\overline{Q}=1$
HOLD	$\overline{S}=1$ $\overline{R}=1$	Q=old Q $\overline{Q}$=old $\overline{Q}$

SR Latch Symbols



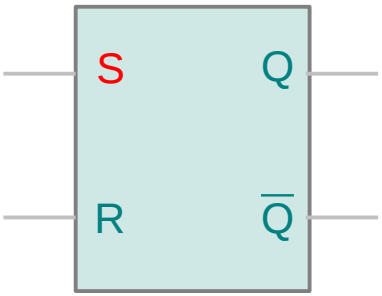




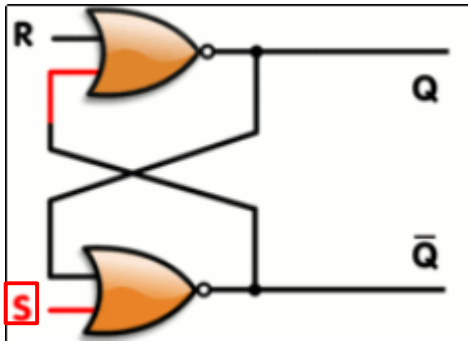


Active High and Low Inputs

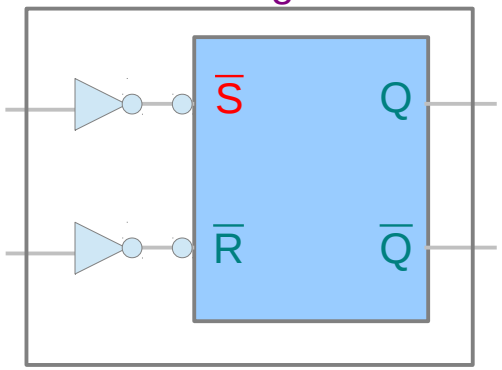
Active High



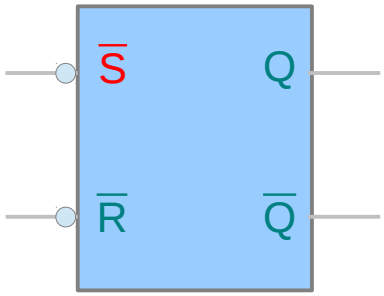
Active High



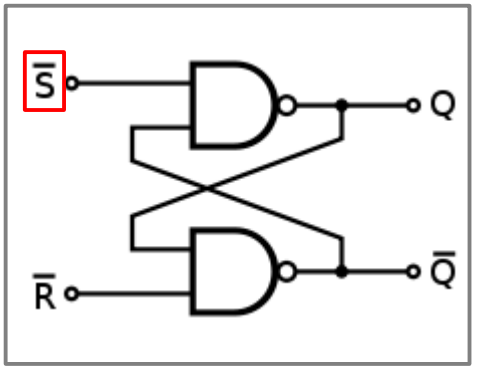
Active High



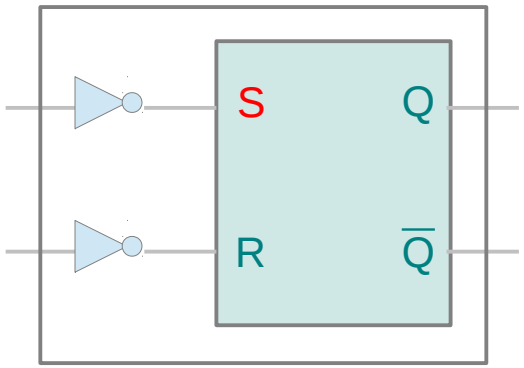
Active Low



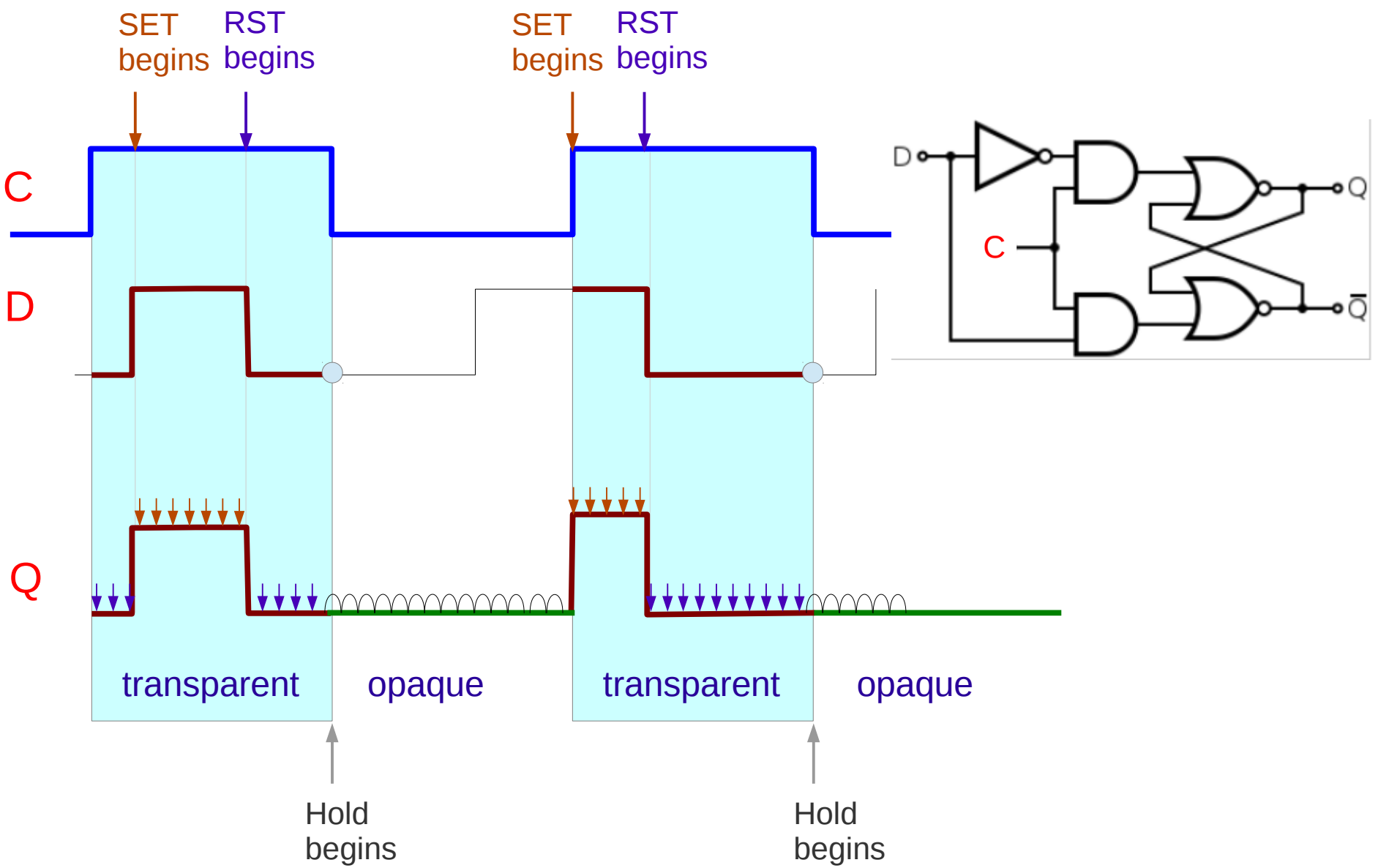
Active Low



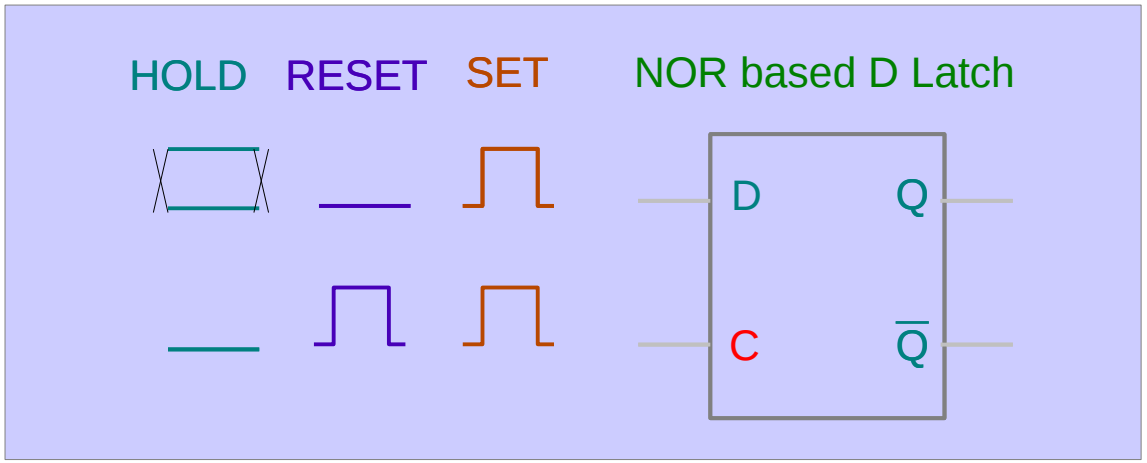
Active Low

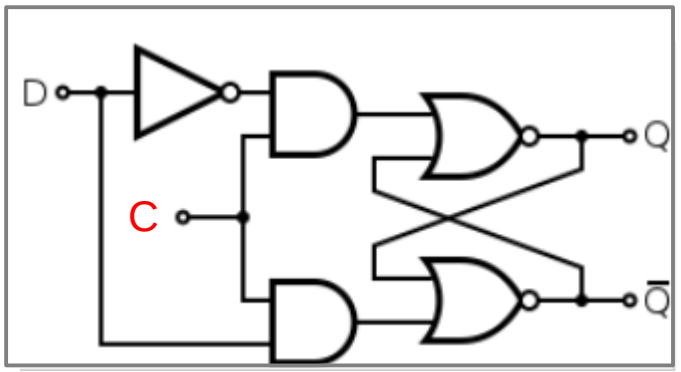


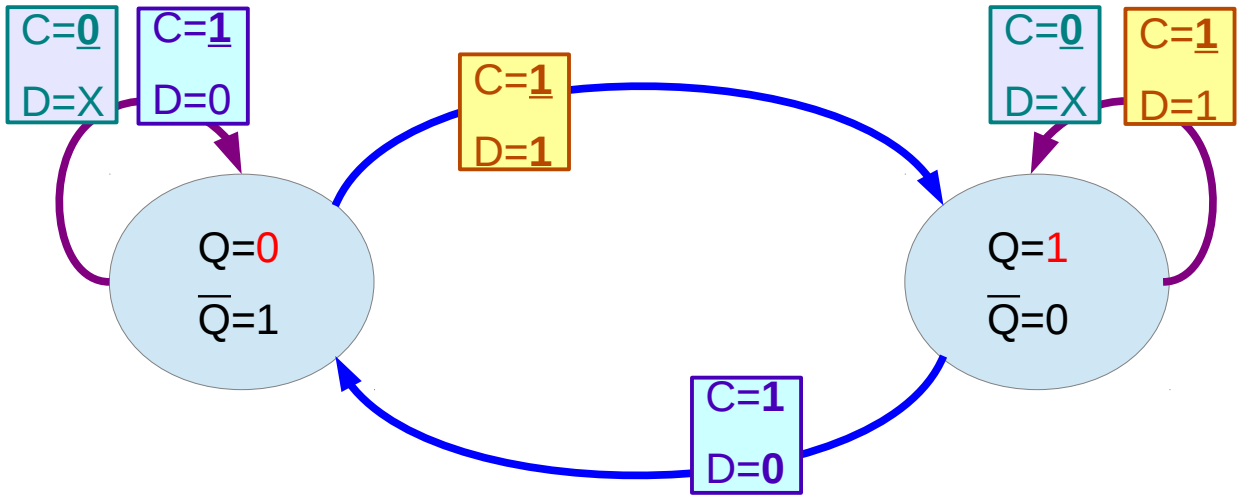
NOR-based D Latch



NOR-based D Latch

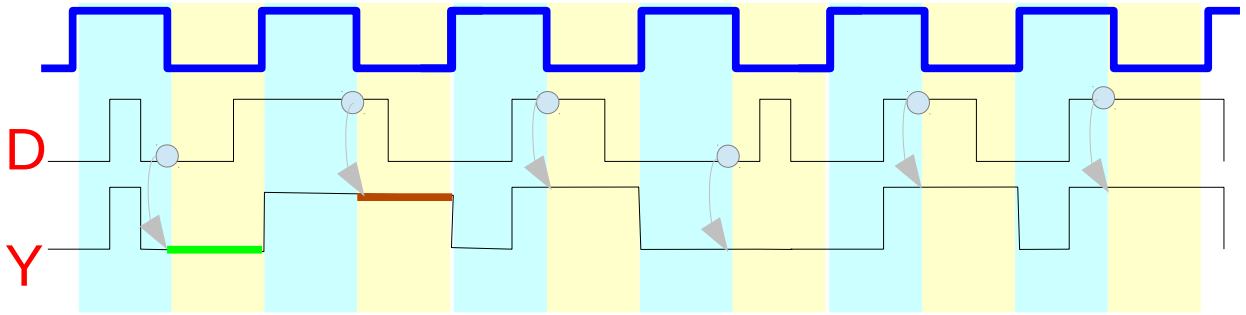


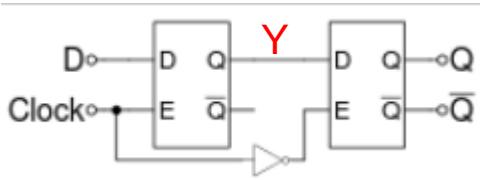




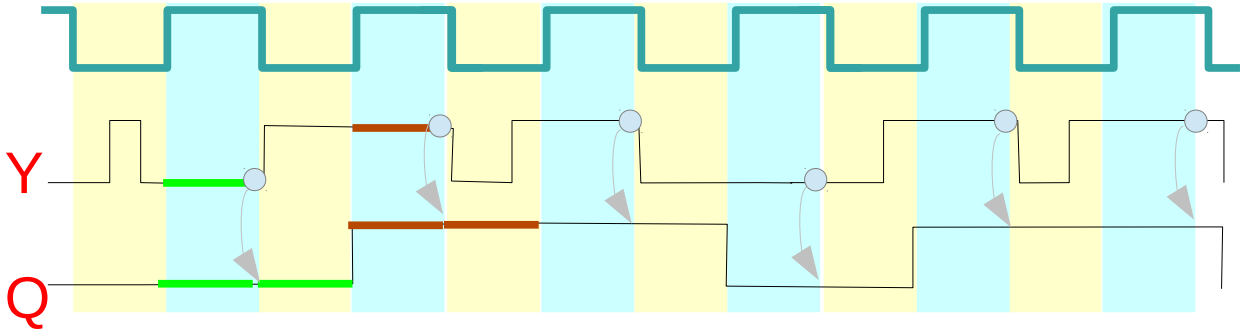
Master-Slave D FlipFop

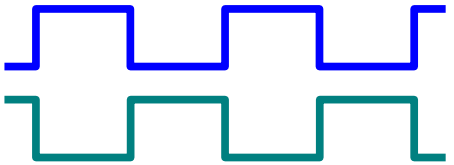
Master D Latch

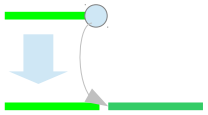




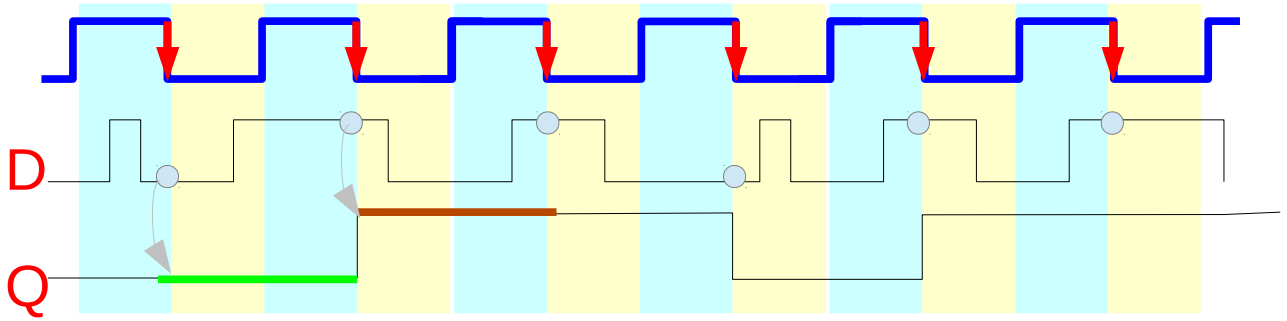
Slave D Latch







Master-Slave D F/F

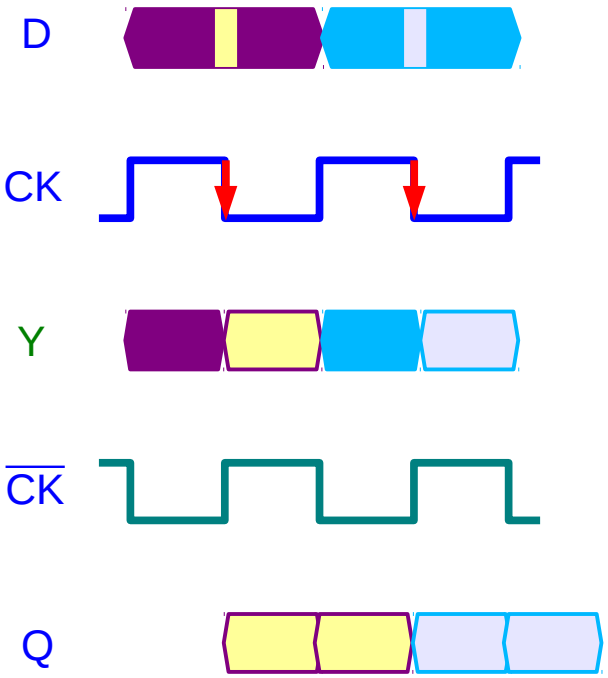


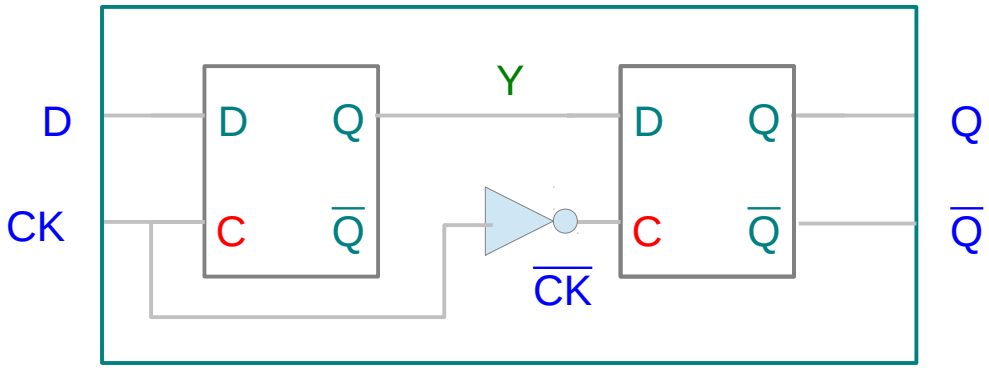
the hold output of the master is transparently reaches the output of the slave

this value is held for another half period

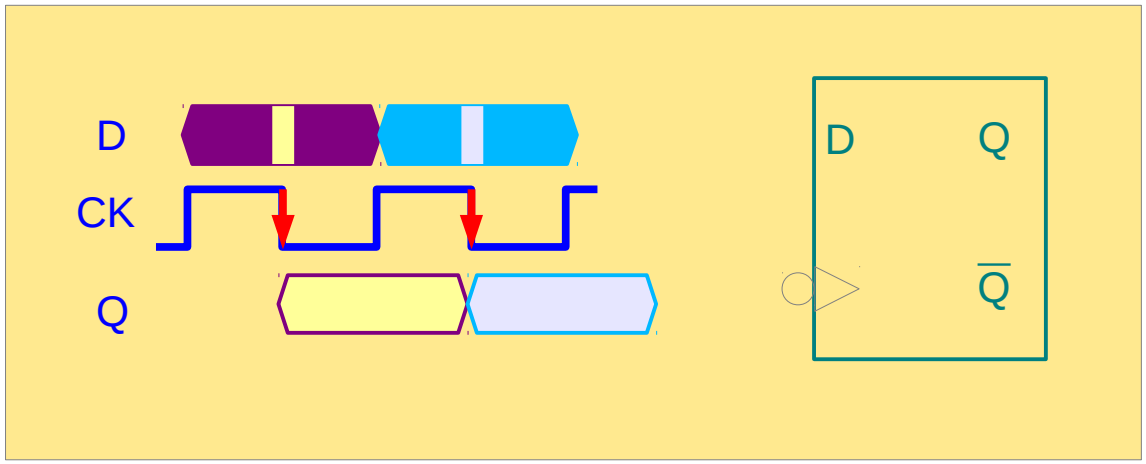
Master-Slave D FlipFlop – Falling Edge

Master D Latch



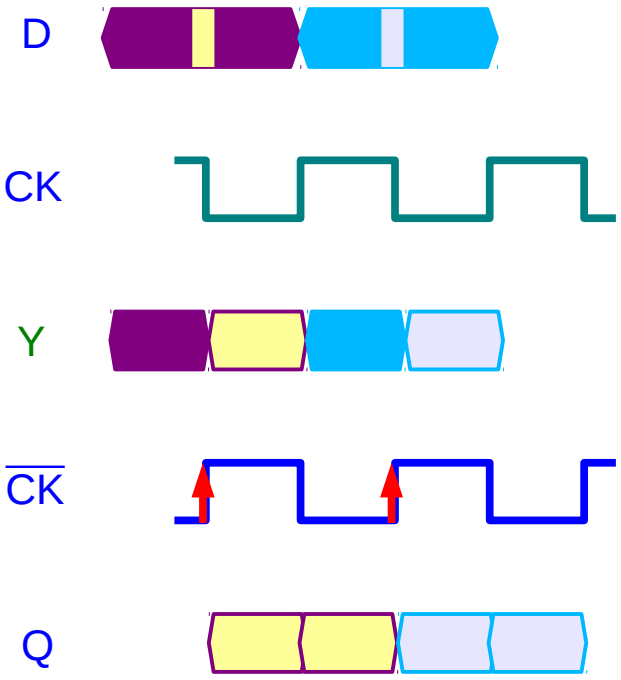


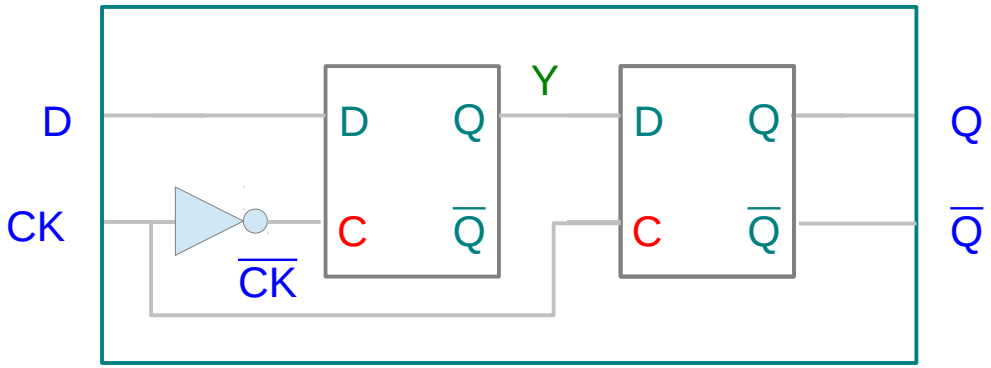
Slave D Latch



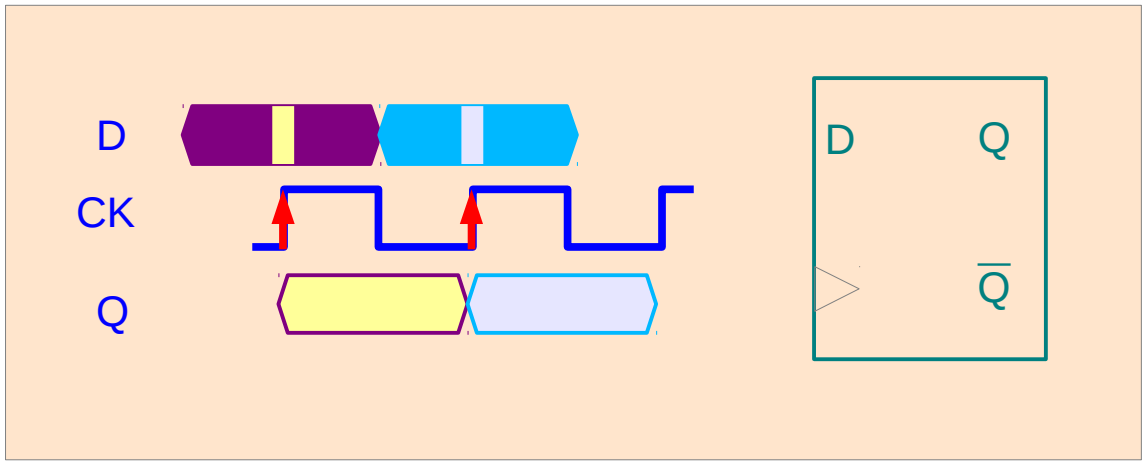
Master-Slave D FlipFlop – Rising Edge

Master D Latch





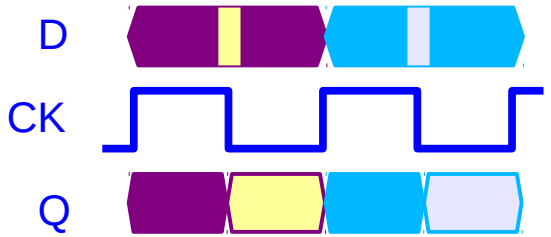
Slave D Latch

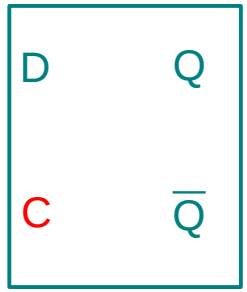


D Latch & D FlipFlop

Level Sensitive D Latch

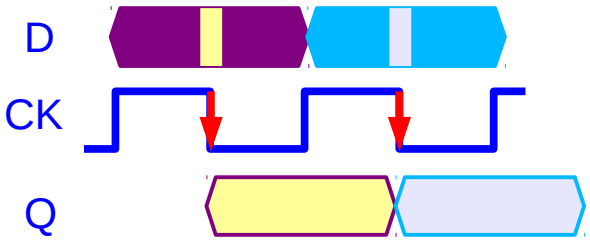
CK=1 transparent
CK=0 opaque

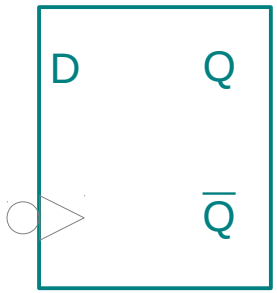




Edge Sensitive D FlipFlop

CK=1→0 transparent
else opaque





Advantages of Latches over FFs

Flipflop designs are very **easy to verify timing**

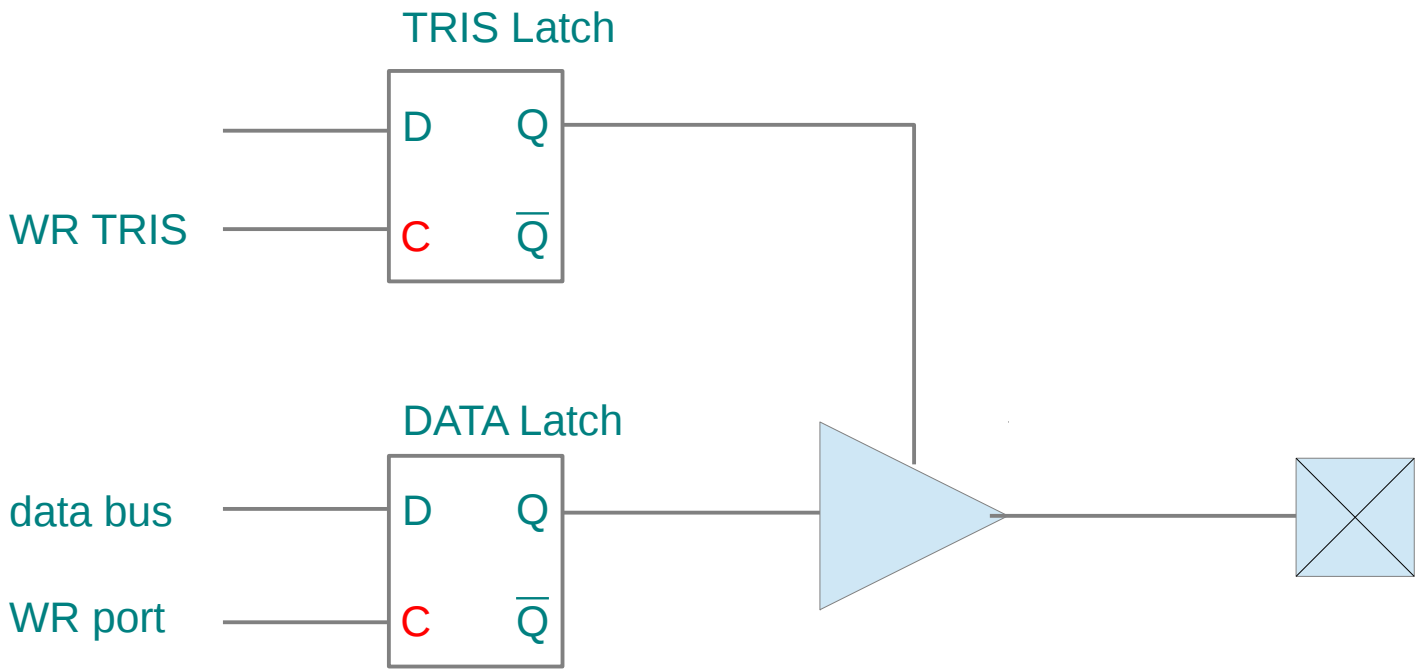
- Each path between flip-flops must be less than the clock period
- Tools check for skew, setup, and hold time violations
- Short paths are padded
  (buffers are added to slow down the signals)
- Skew in flip-flop based systems affects the critical path

Most designs in industry are based on flip-flops

Latch designs are **more flexible** than a flip-flop design

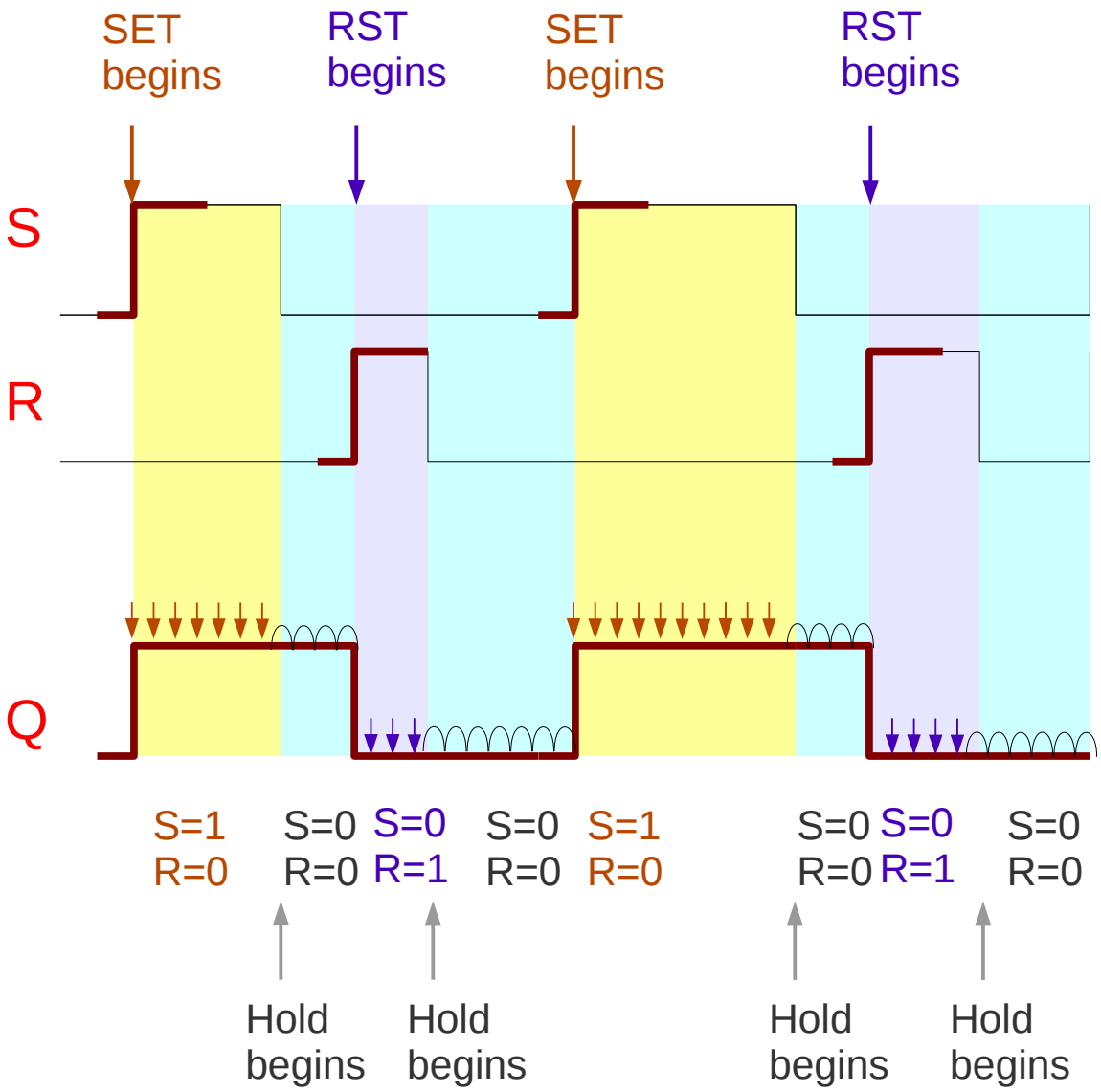
- Need to CAD tools to make sure it works
- Can borrow time to allow a path to be longer than clock period
- Can tolerate clock skew
  -- skew does not directly add to cycle time
- Less silicon area

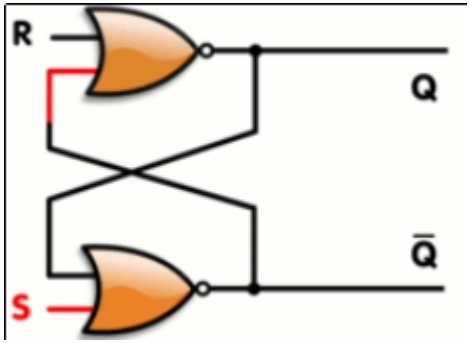
Latches at the output ports



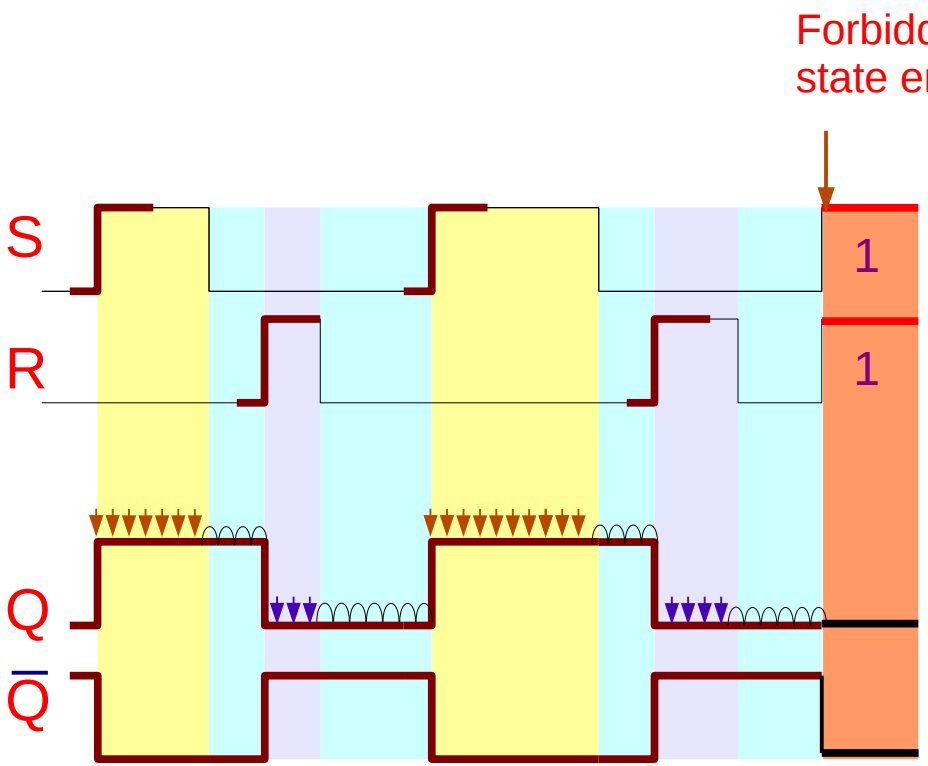
Forbidden State

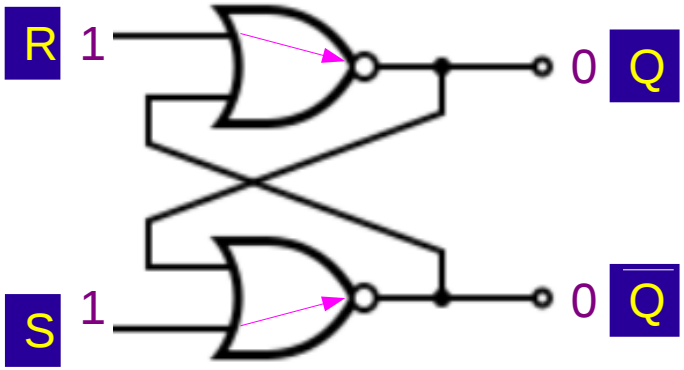
NOR-based SR Latch



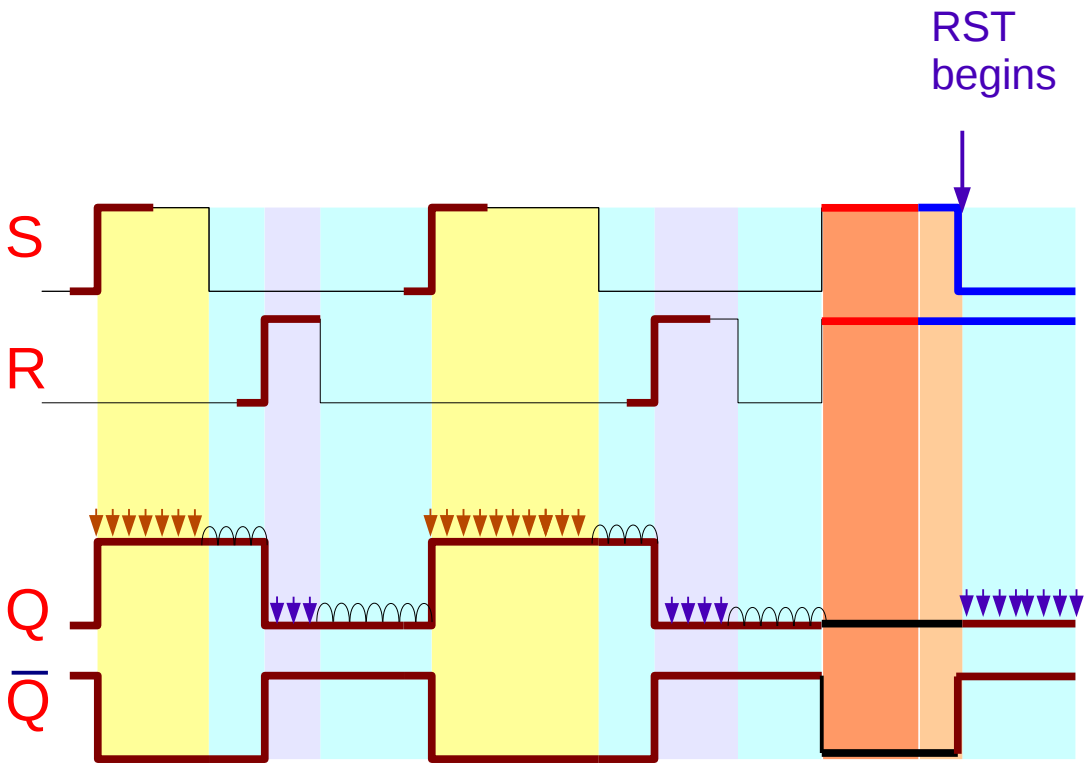


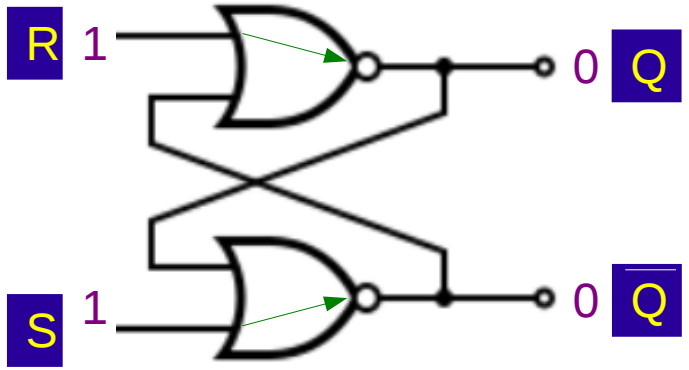
Gated SR Latch (SR=11)

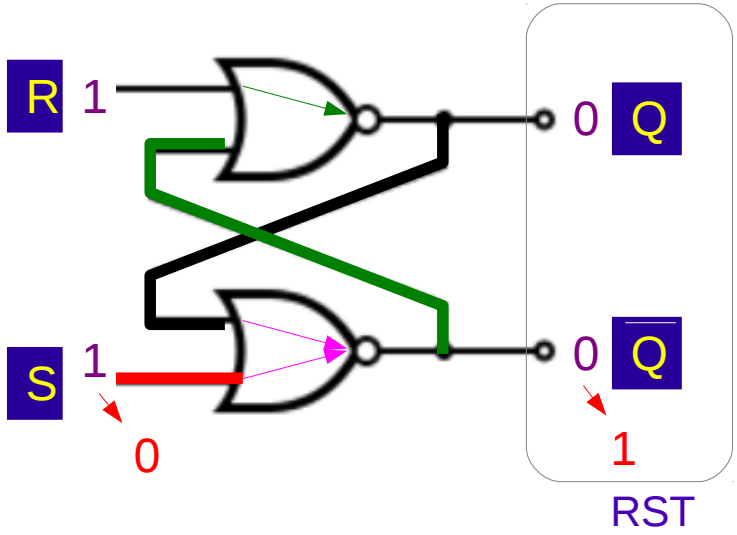




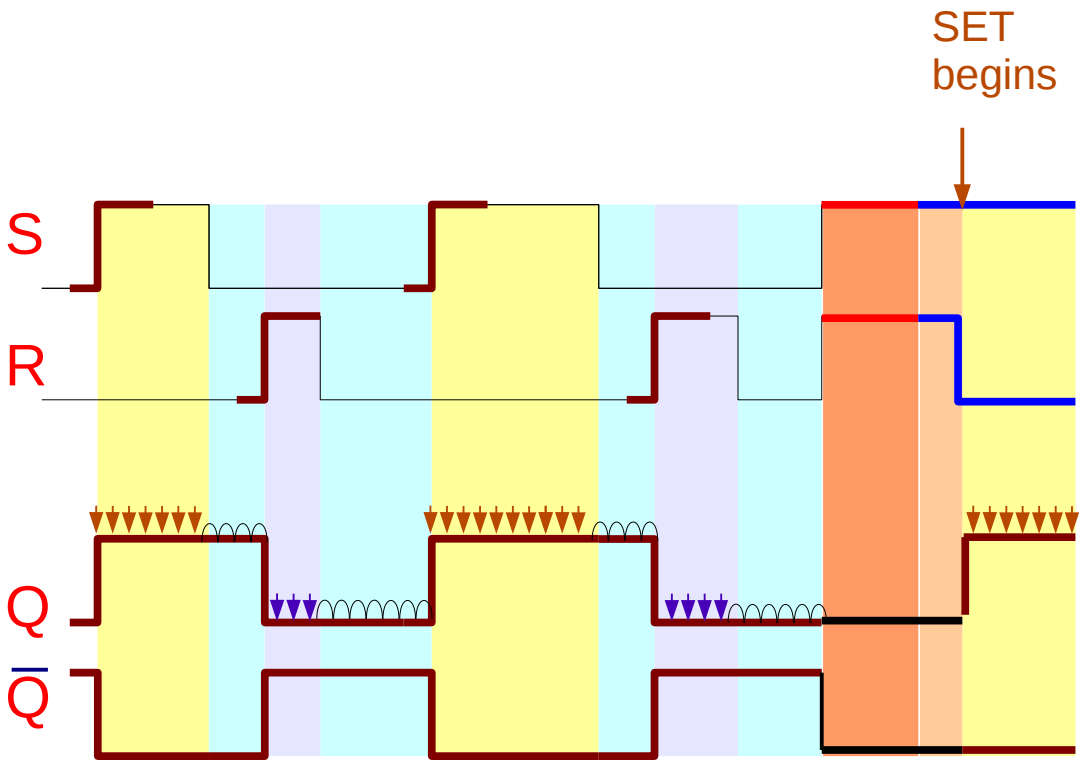
Gated SR Latch (SR=11 to 01)

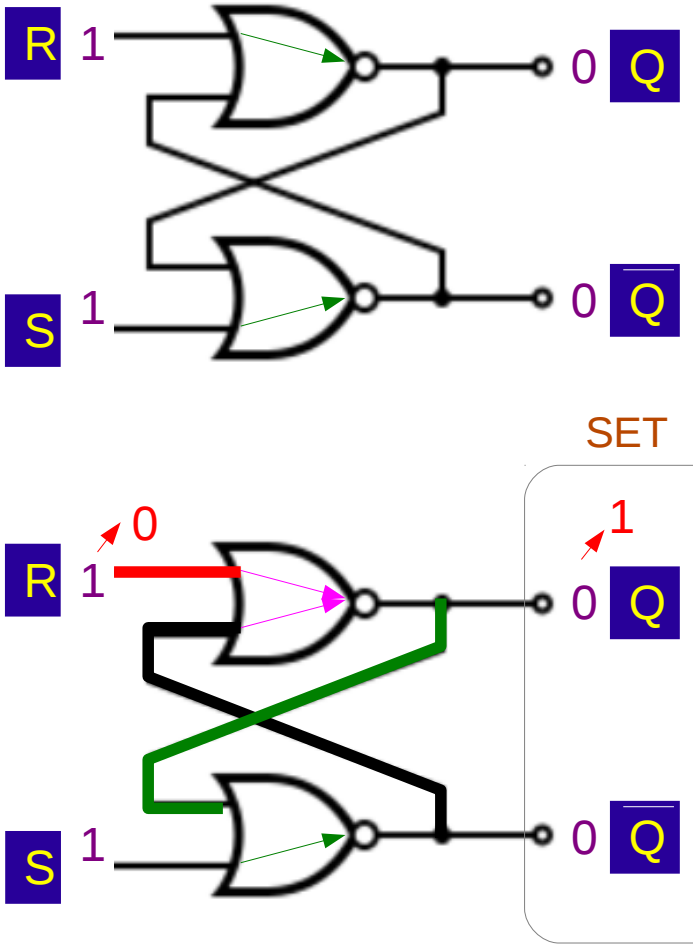




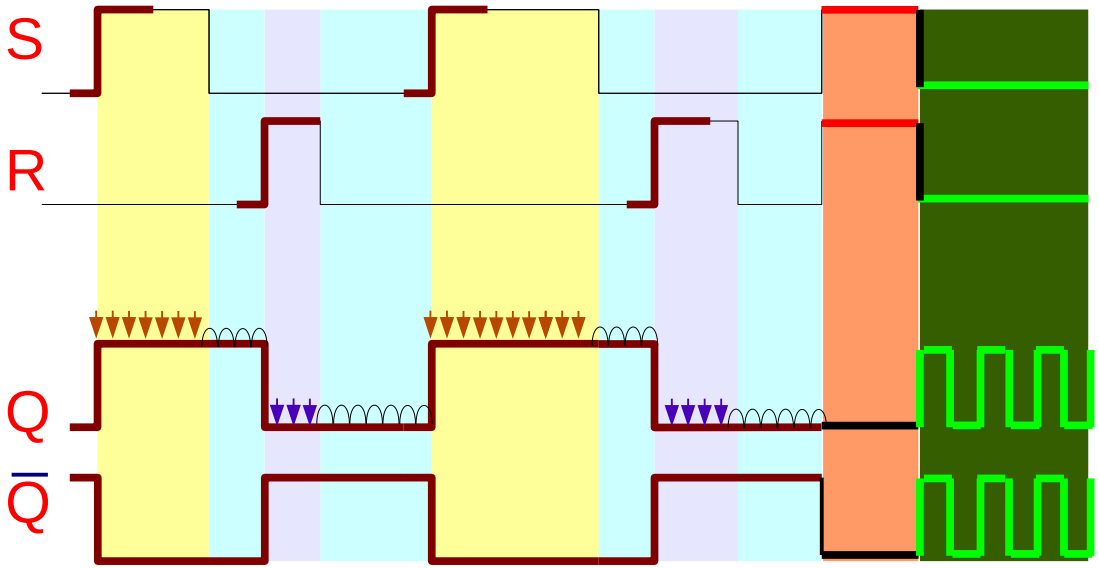


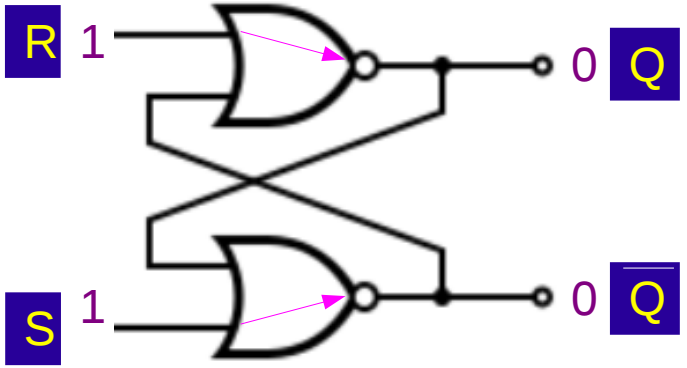
Gated SR Latch (SR=11 to 10)

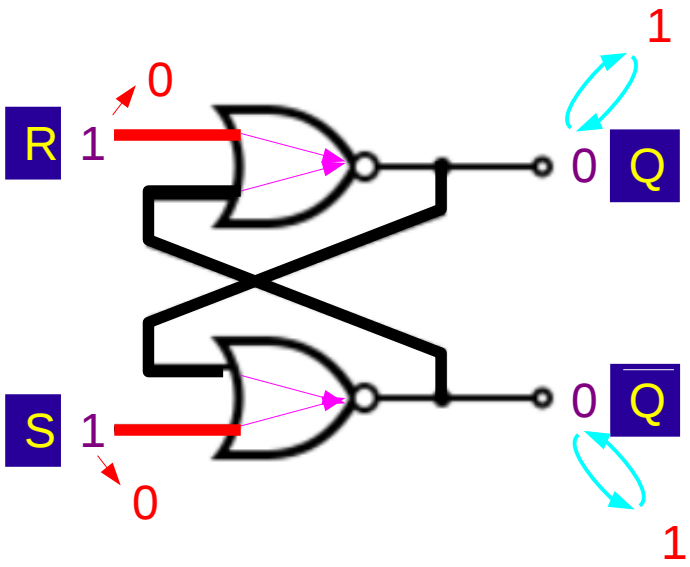




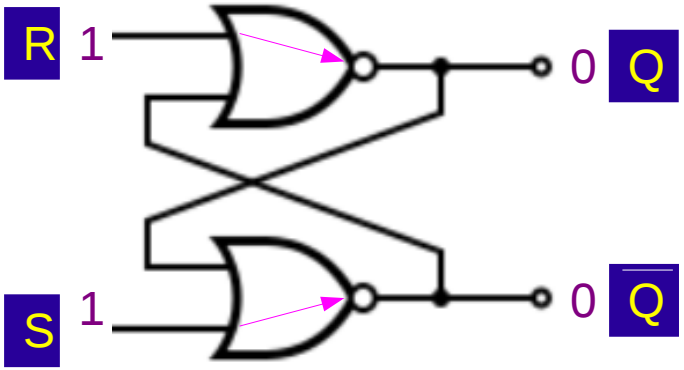
Gated SR Latch (SR=11 to 00)

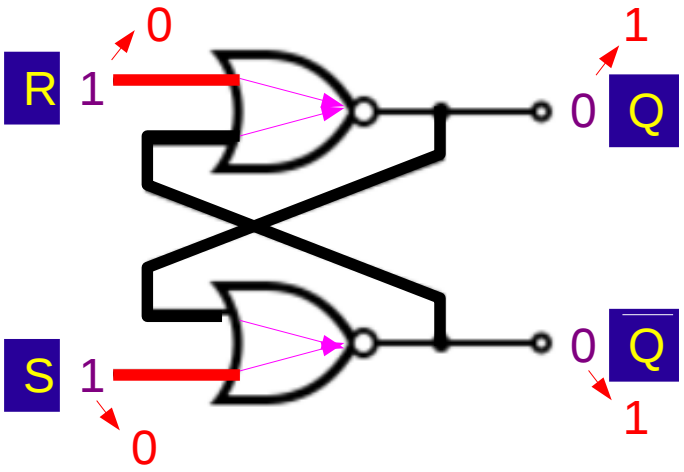


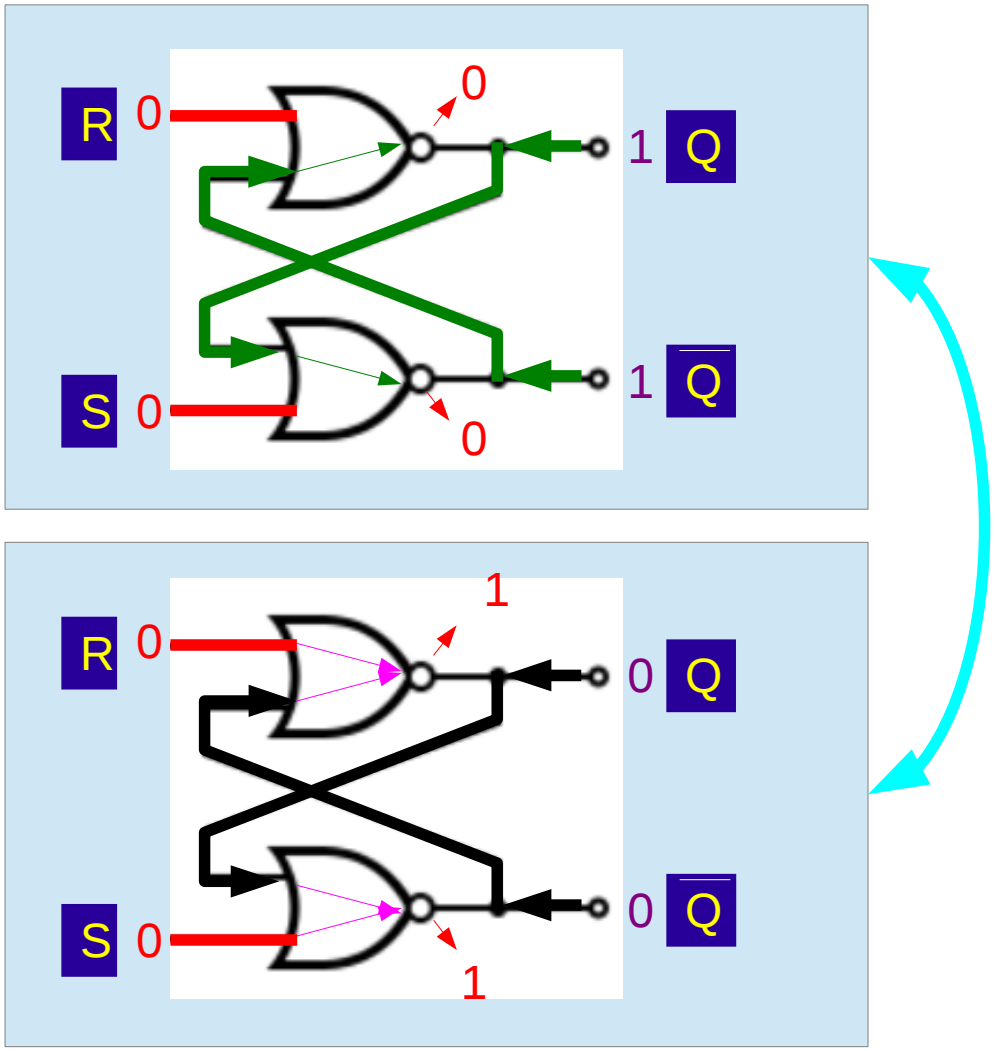




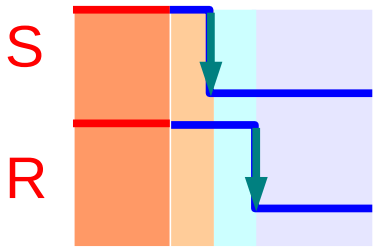
Gated SR Latch Oscillation State



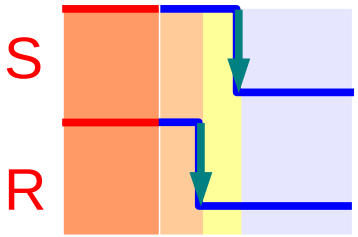




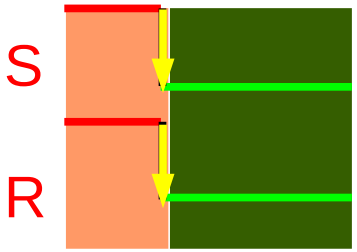
Forbidden State Transition



Forbidden St → RESET, Hold



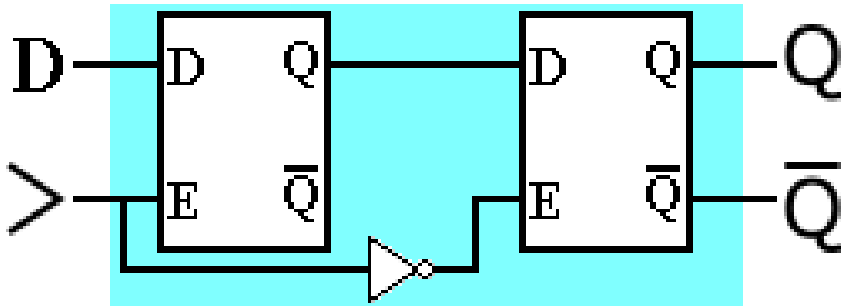
Forbidden St → SET, Hold

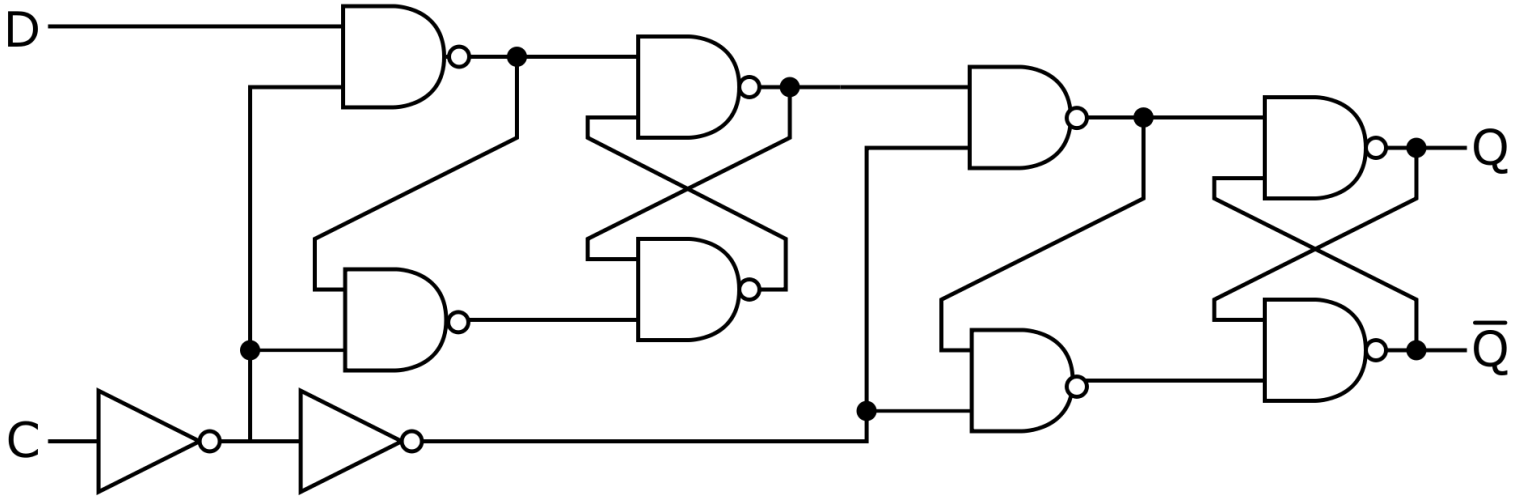


Forbidden St → Oscillation

Classical SR Latch Design

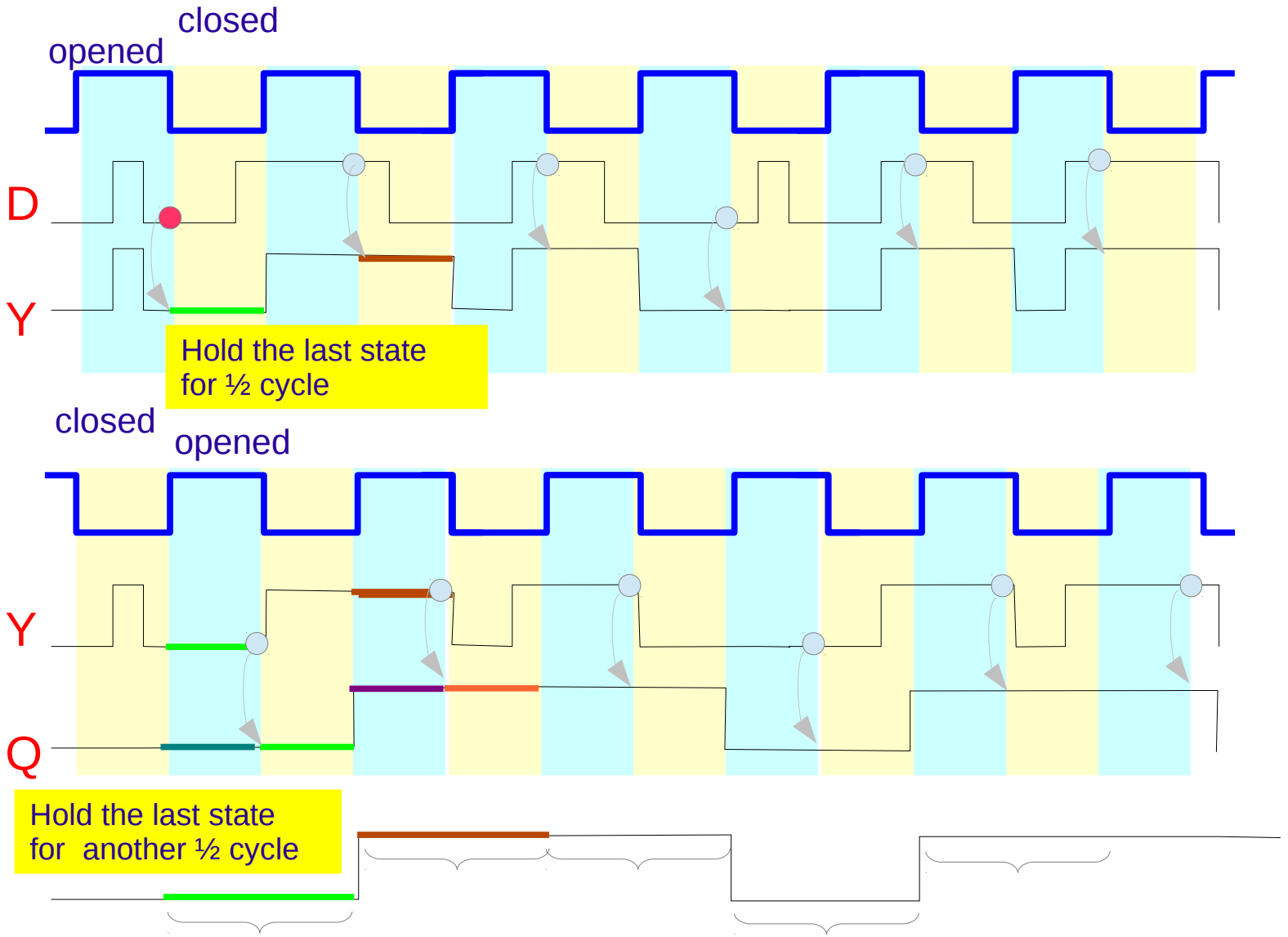
Master-Slave FF



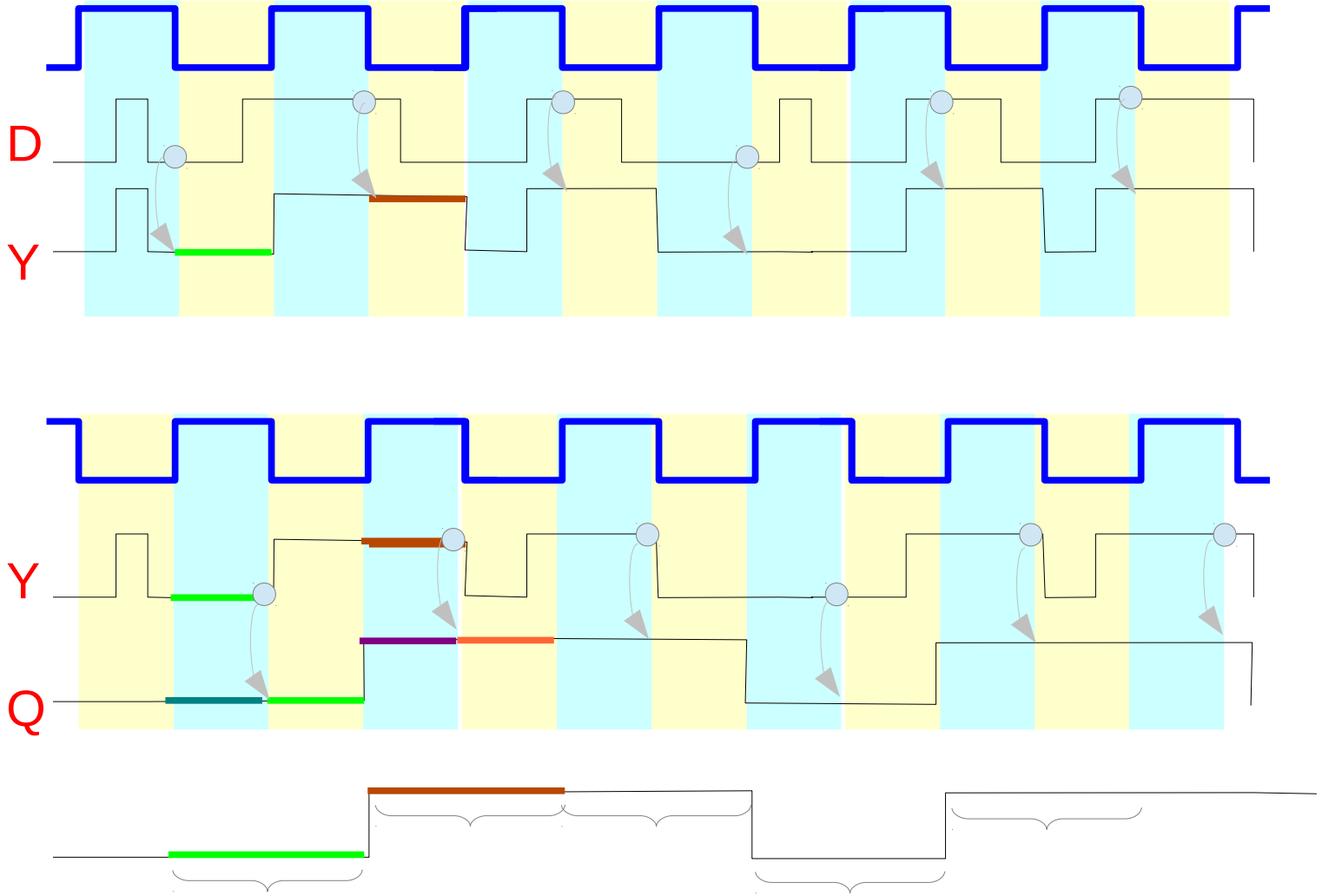


* figures from wikipedia.org

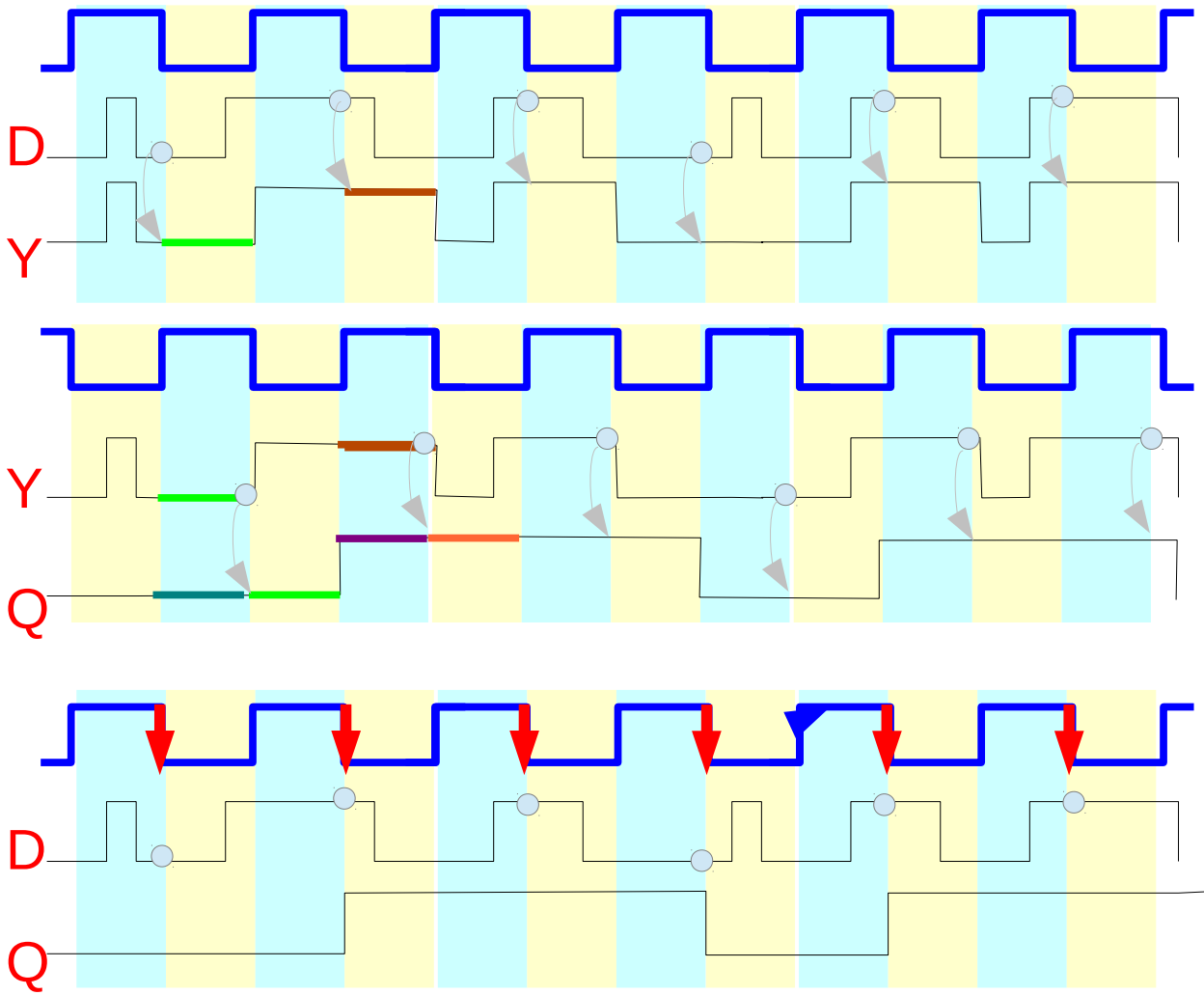
Master-Slave D FF



Master-Slave FF



Master-Slave FF

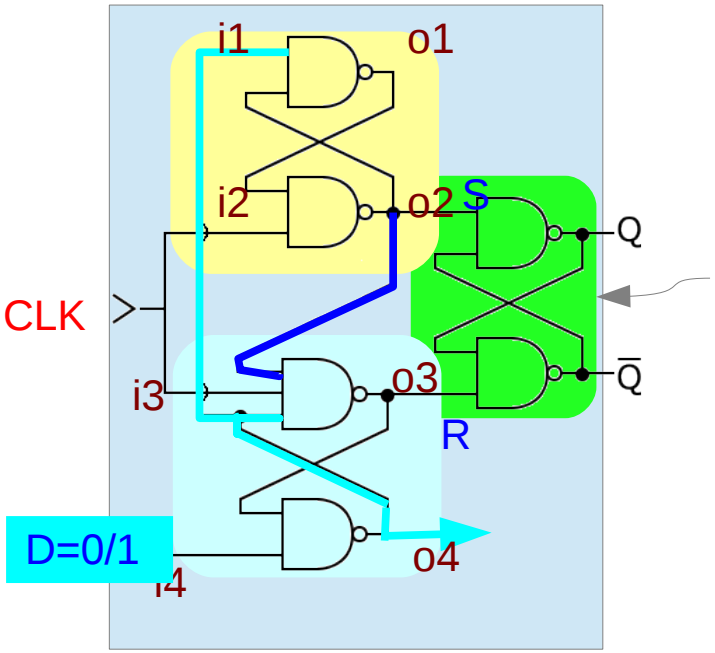


Master D Latch

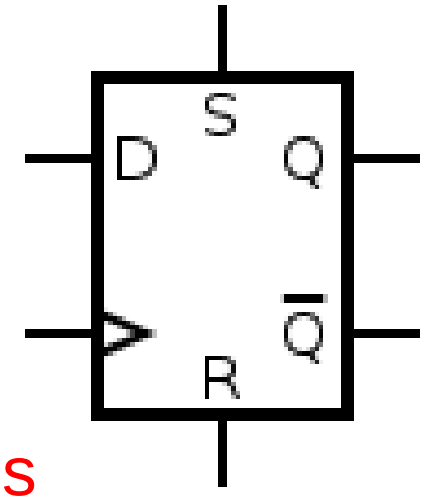
Slave D Latch

Master-Slave D F/F

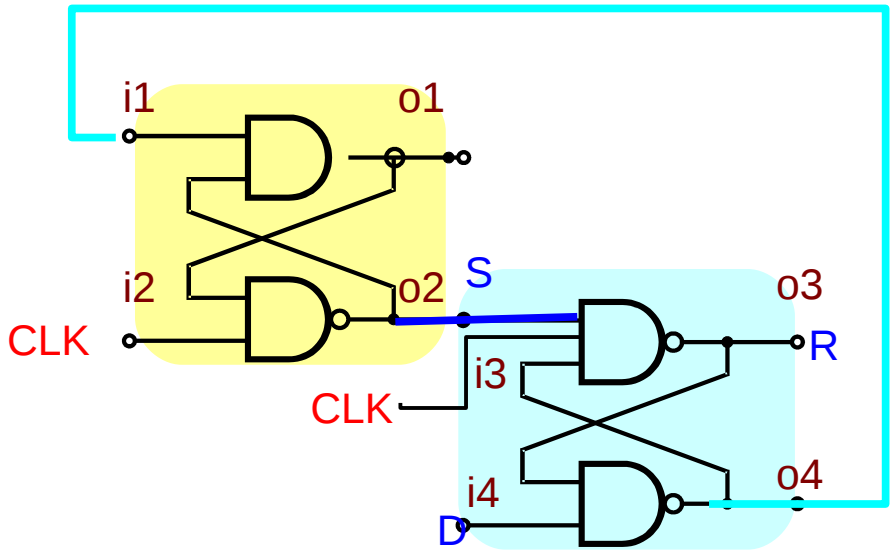
Classical Edge Triggered FF



Output Stage Latch

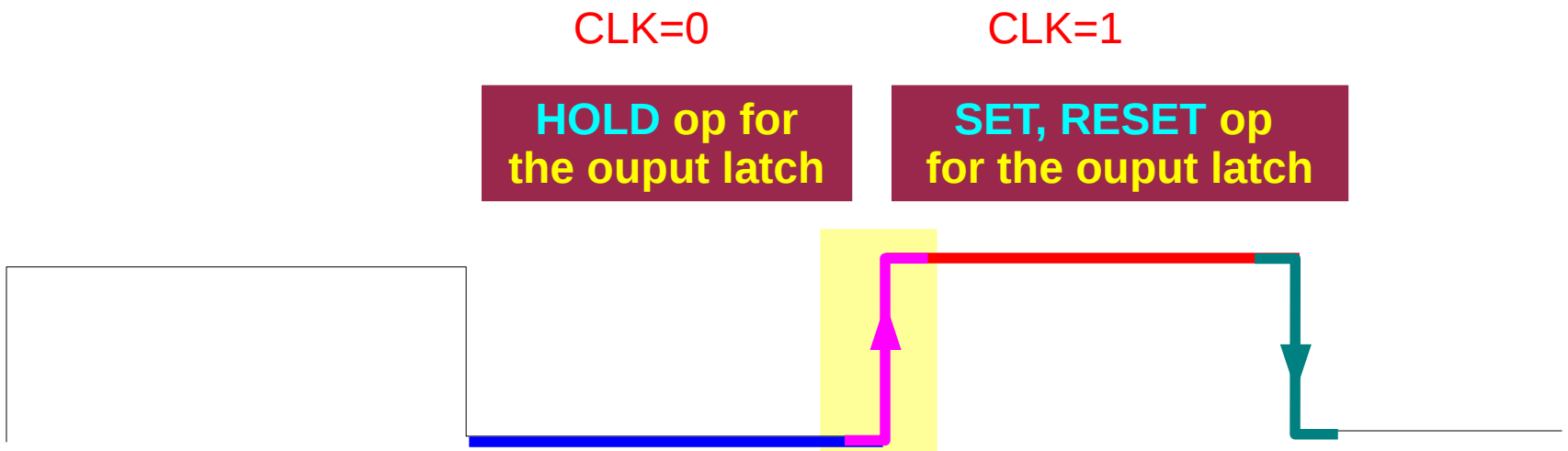


Input Stage Latches

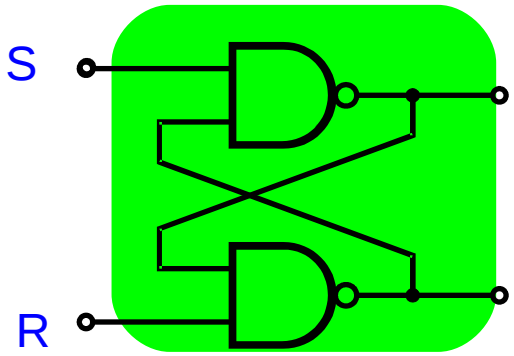


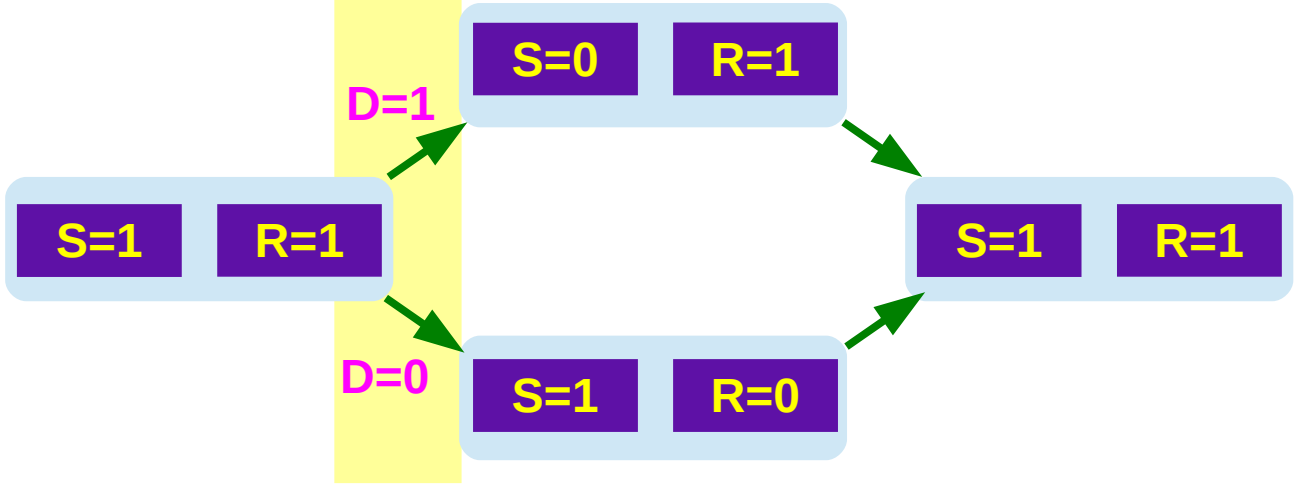
* figures from
wikipedia.org

Classical Edge Triggered FF

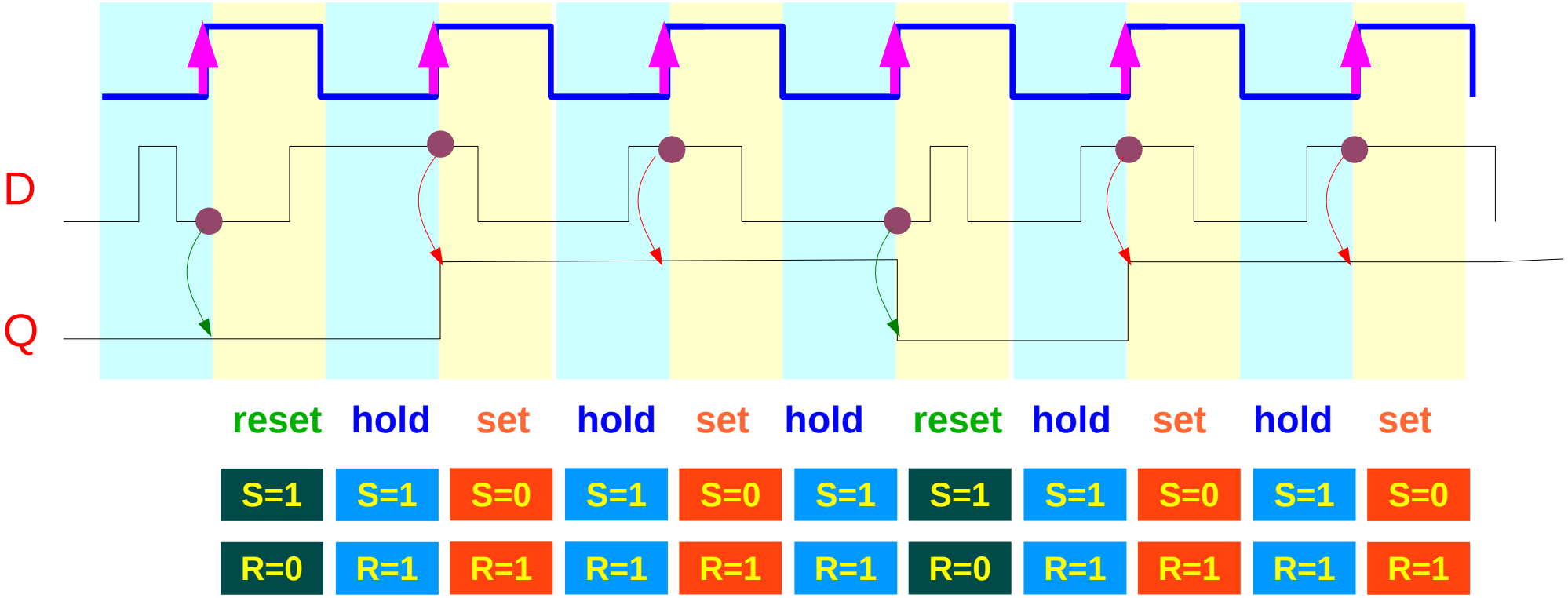


Output Stage Latch

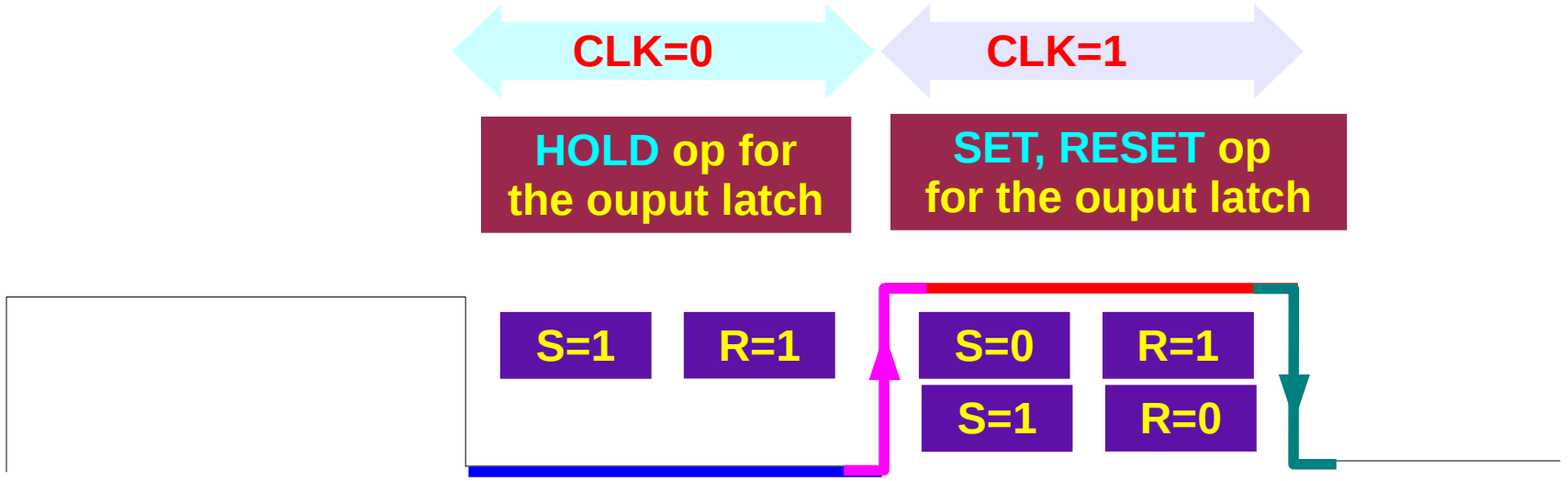




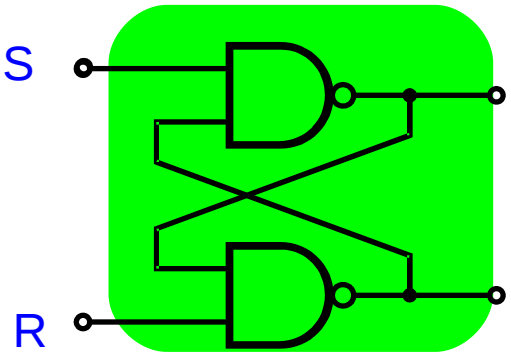
Classical Edge Triggered FF



Classical Edge Triggered FF



Output Stage Latch



CLK=0 → 1

When D=0
HOLD → RESET

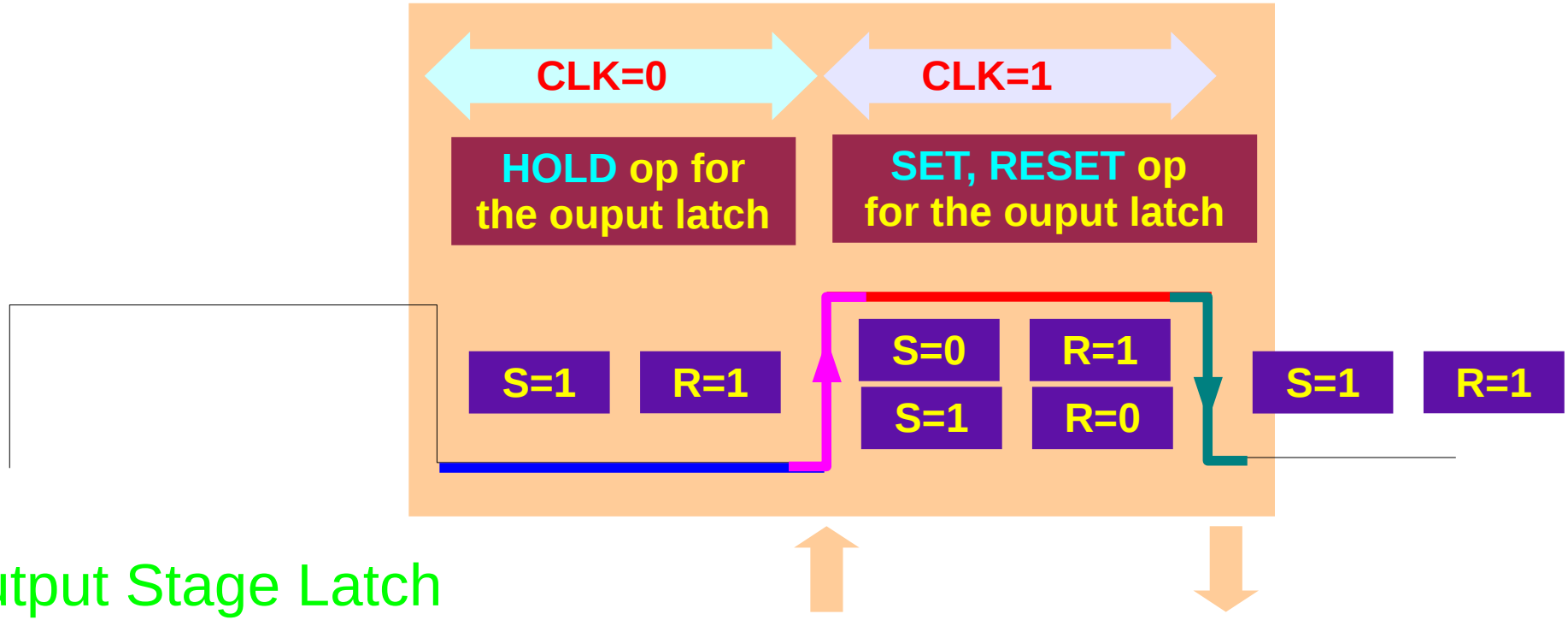
When D=1
HOLD → SET

CLK=1 → 0

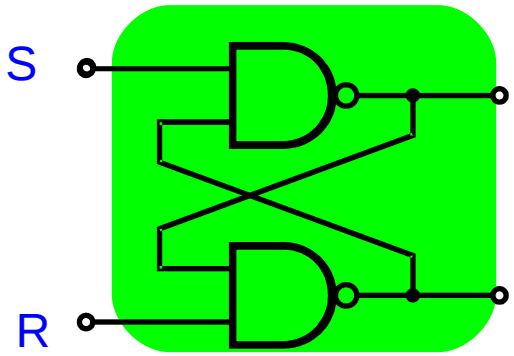
RESET → HOLD

SET → HOLD

Output Latch Operation (Classical Edge Triggered FF)



Output Stage Latch



CLK=0 → 1

When D=0
HOLD → RESET

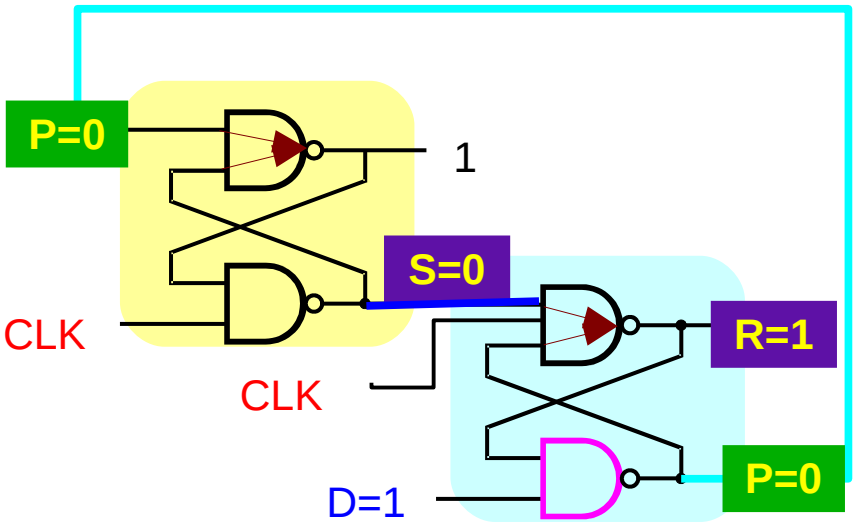
When D=1
HOLD → SET

CLK=1 → 0

RESET → HOLD

SET → HOLD

Set Operation (Classical Edge Triggered FF)



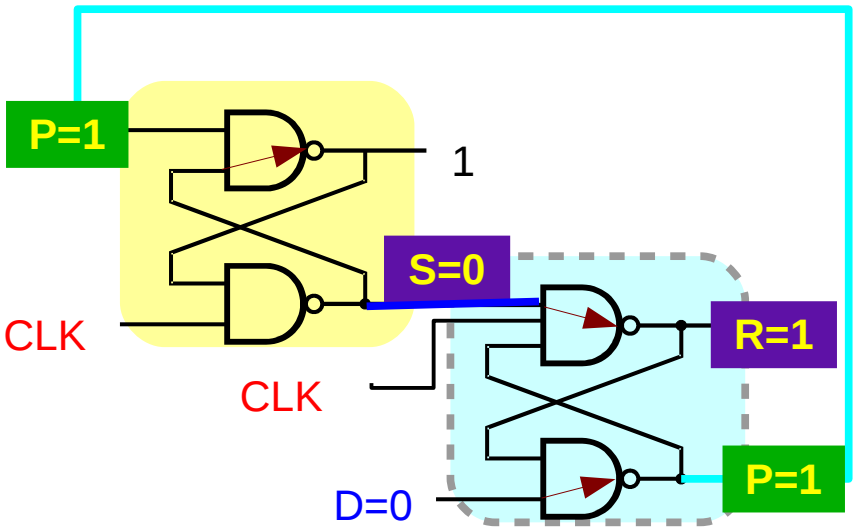
SET op for the ouput latch

When CLK=1, regardless of D

S=0 **R=1**

CLK=0 → S=1 ⇨ CLK=1

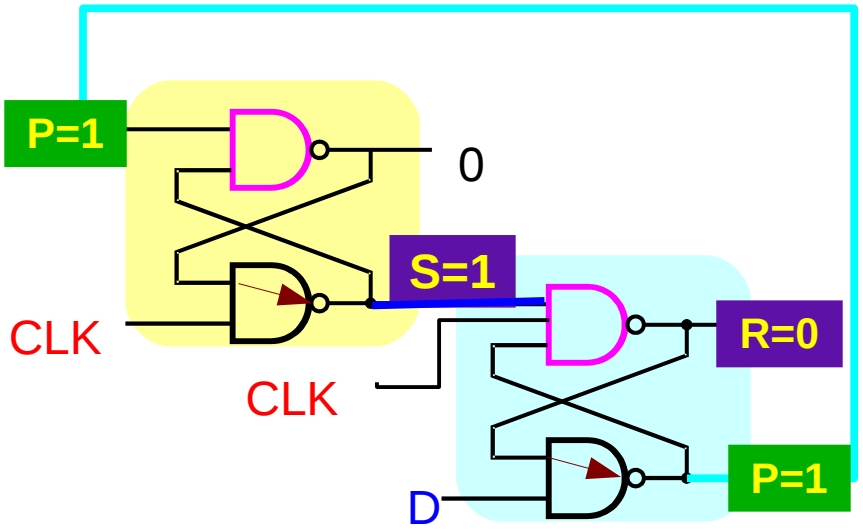
D=1 → P=0



CLK=0 → S=1 ⇨ CLK=1

D=0 → P=1

Reset Operation (Classical Edge Triggered FF)



RESET op for the ouput latch

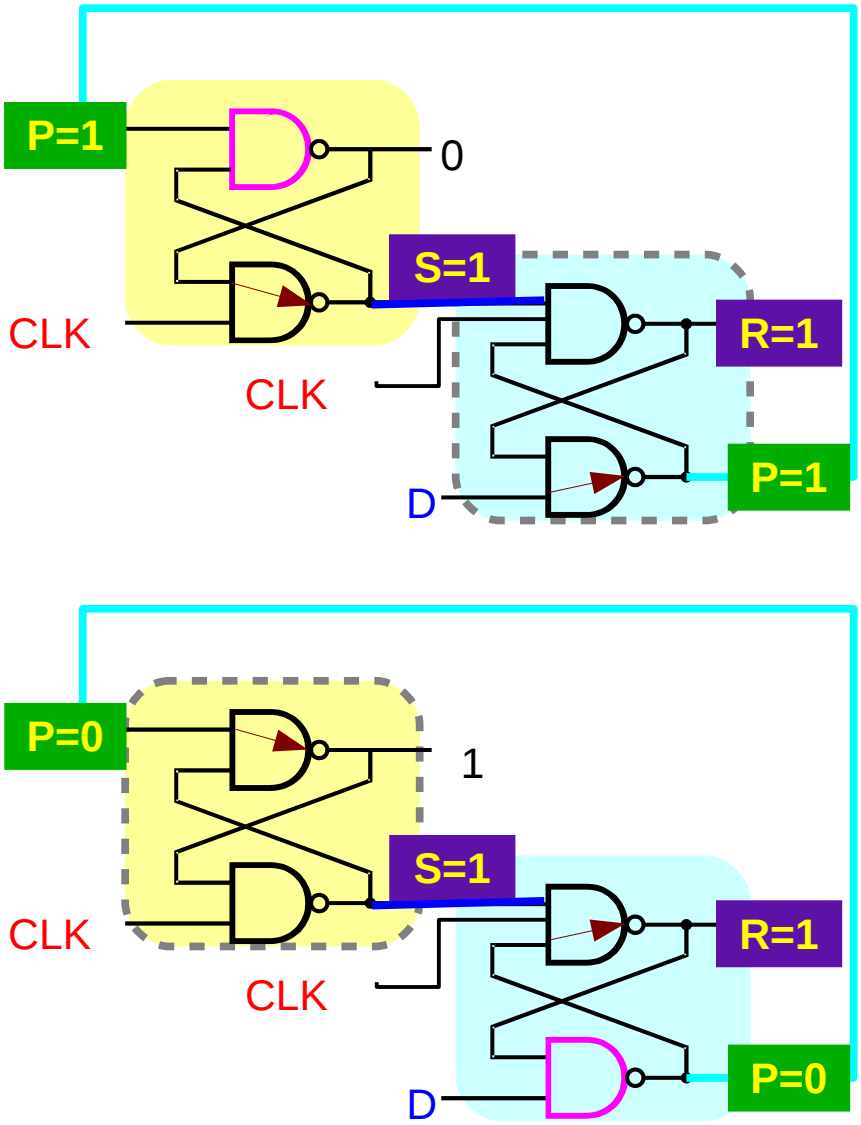
When CLK=1, regardless of D

S=1 **R=0**

CLK=0 → ~~R=1~~ ⇨ CLK=1

D=0 / 1

Hold Operation (Classical Edge Triggered FF)



HOLD op for the ouput latch

When CLK=0, regardless of D

S=1 **R=1**

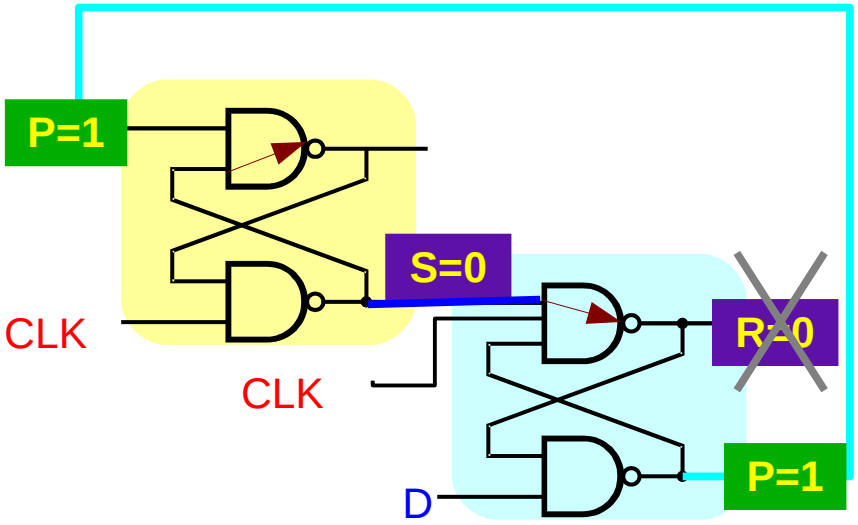
CLK=1 → R=0 ⇨ CLK=0

D=0 → P=1

CLK=1 → S=0 ⇨ CLK=0

D=1 → P=1

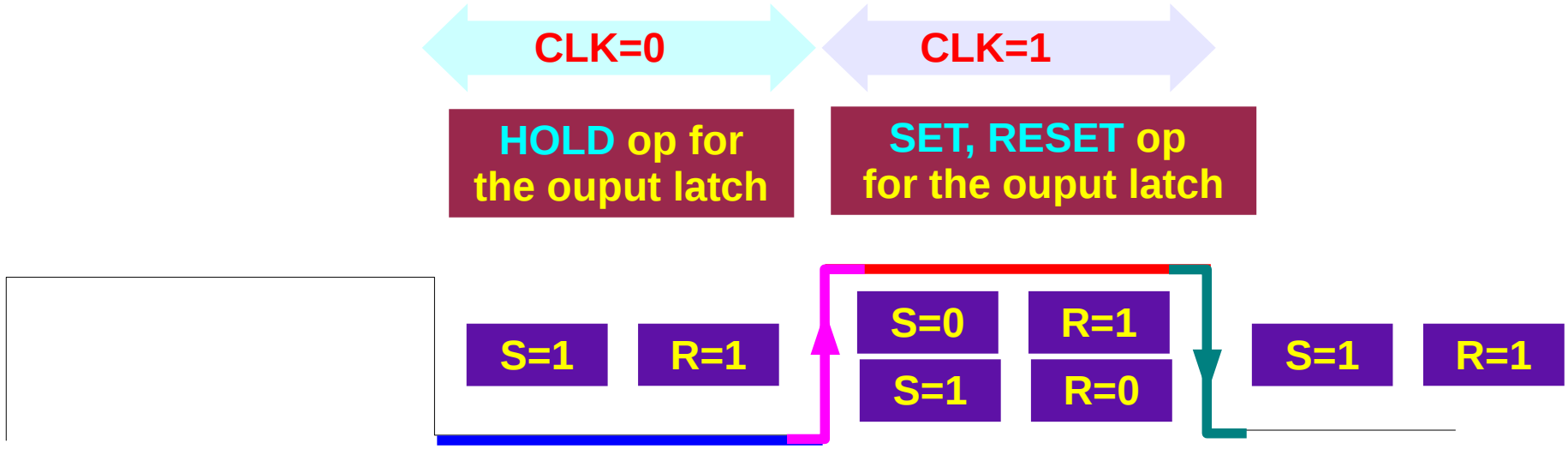
Forbidden Operation (Classical Edge Triggered FF)



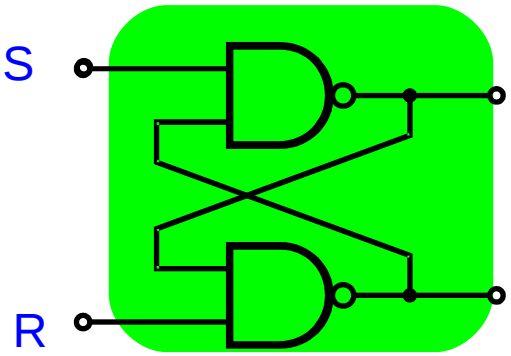
Forbidden for the ouput latch

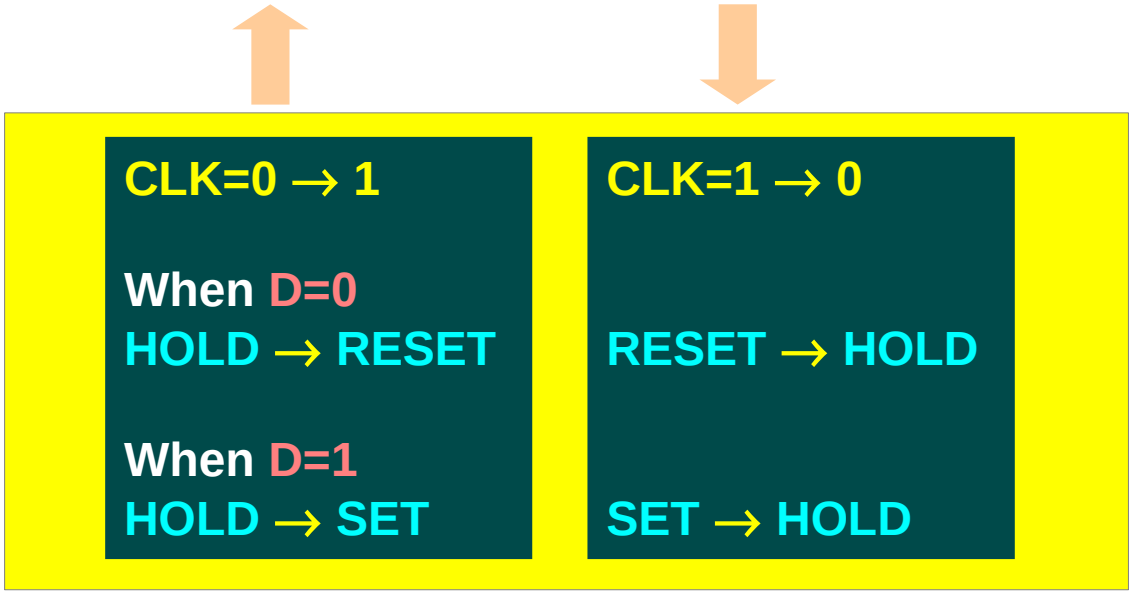
Impossible to enter

Clock Transition (Classical Edge Triggered FF)

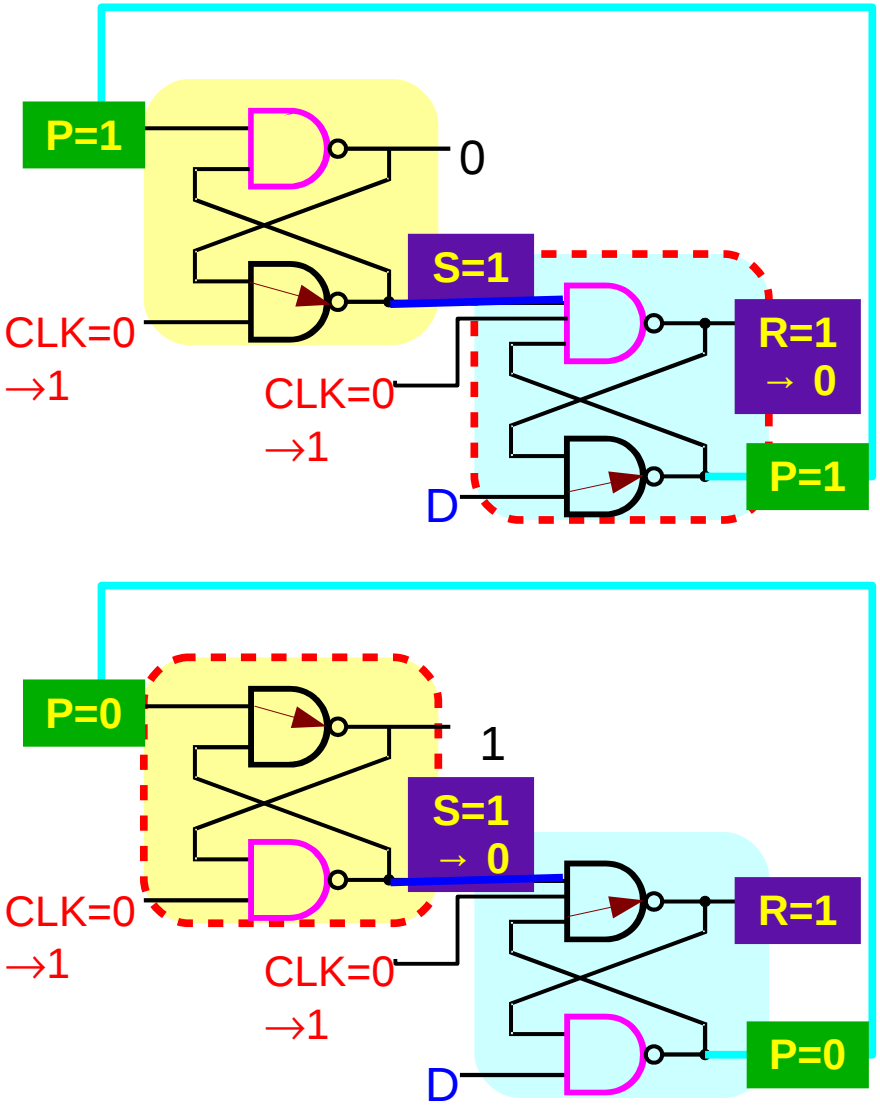


Output Stage Latch





Rising Edge Transition (Classical Edge Triggered FF)



Rising edge CLK=0→1

HOLD → RESET / SET

If D=0	SR=11	⇨	SR=10
If D=1	SR=11	⇨	SR=01

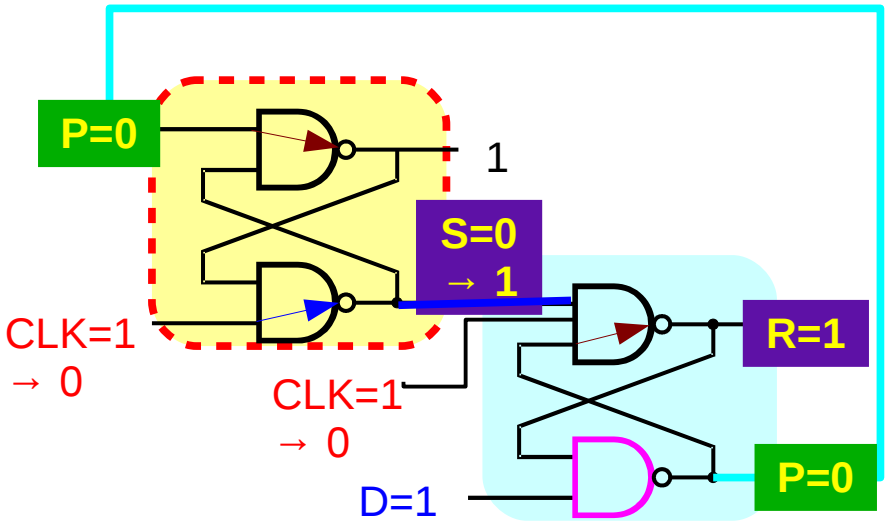
CLK=1 → R=0

D=0 → P=1

CLK=1 → S=0

D=1 → P=1

Falling Edge Transition (1) (Classical Edge Triggered FF)



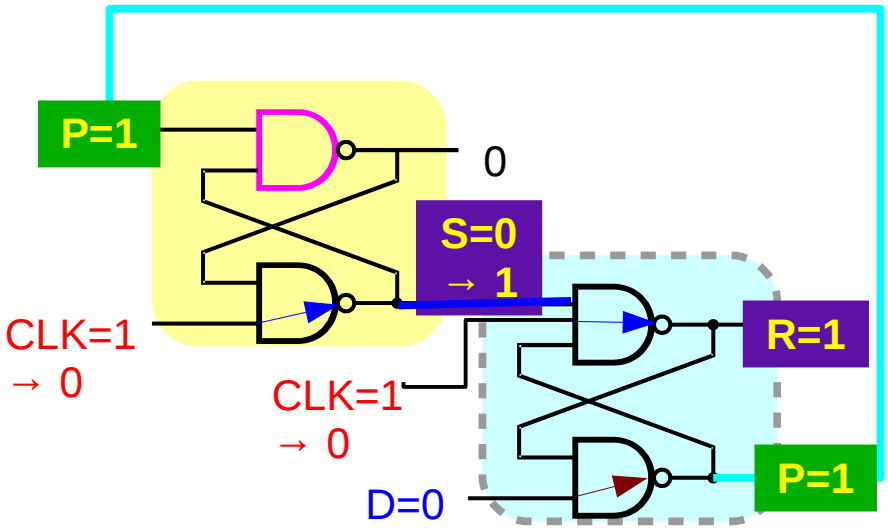
Falling edge CLK=1→0

SET → HOLD

SR=01 ⇨ **SR=11**

CLK=0 → S=1

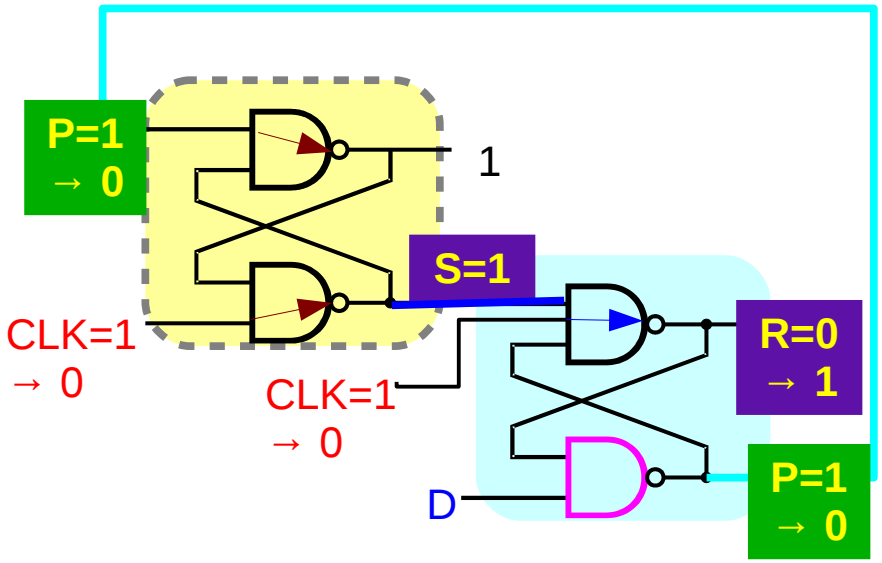
D=1 → P=0



CLK=0 → S=1

D=0 → P=1

Falling Edge Transition (2) (Classical Edge Triggered FF)



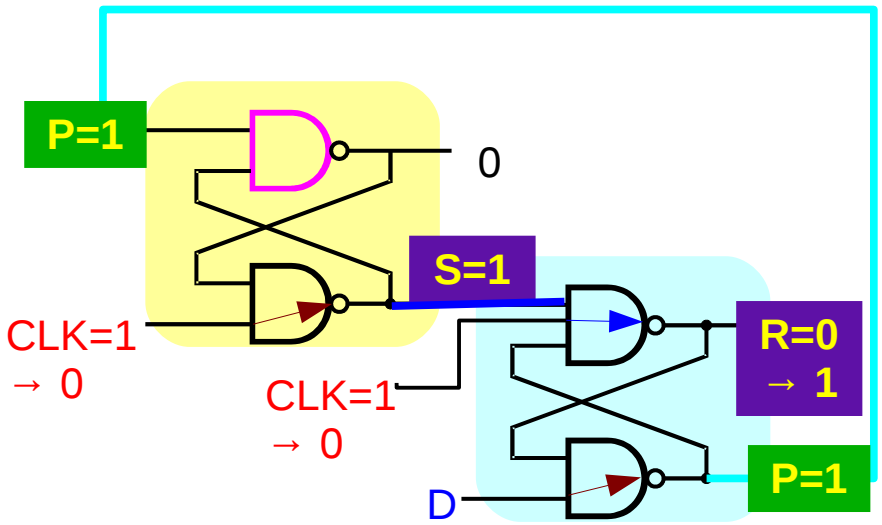
Falling edge CLK=1→0

RESET → HOLD

SR=10 ⇨ **SR=11**

CLK=0 → R=1

D= 1

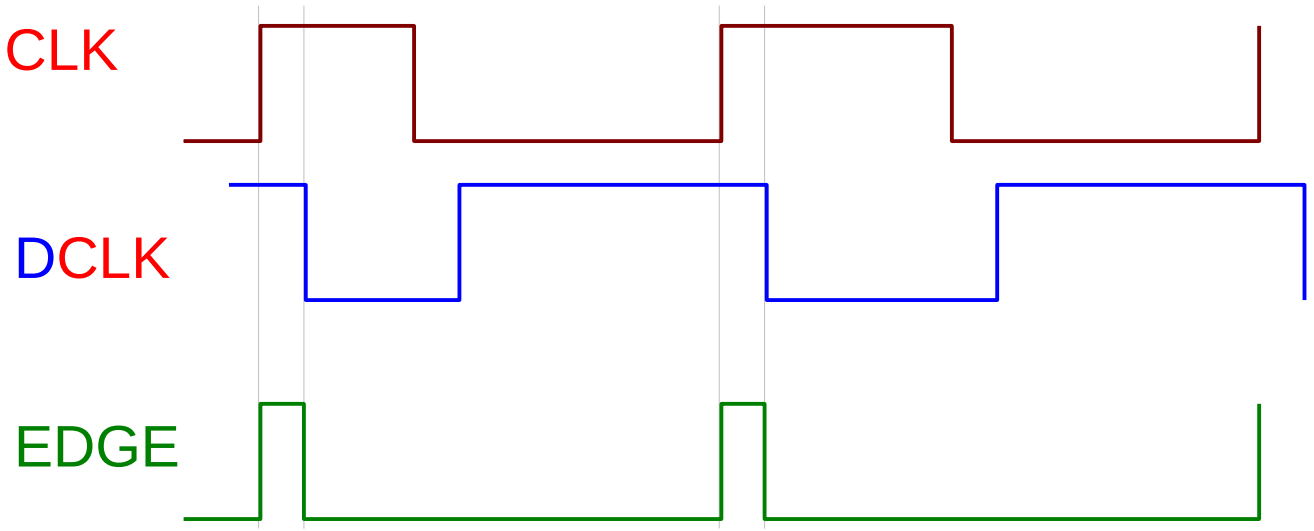


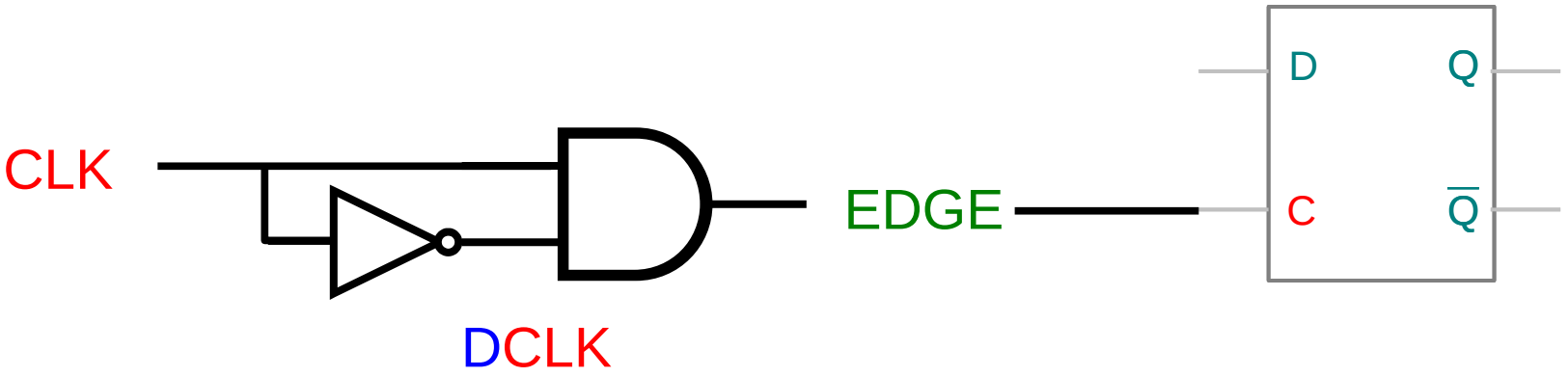
CLK=0 → R=1

D= 0

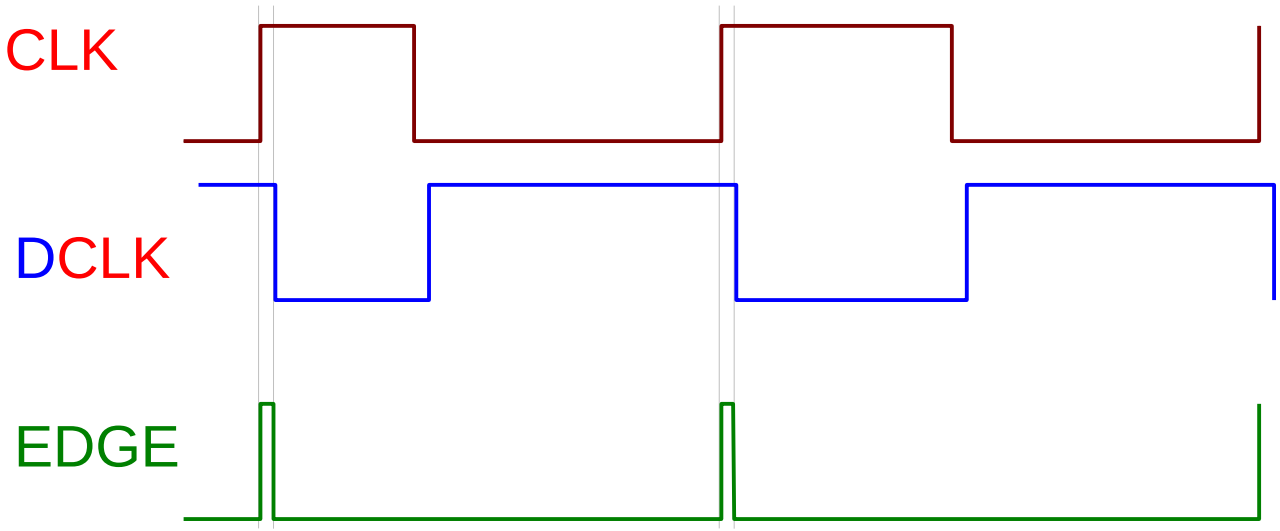
Modern Latch Design

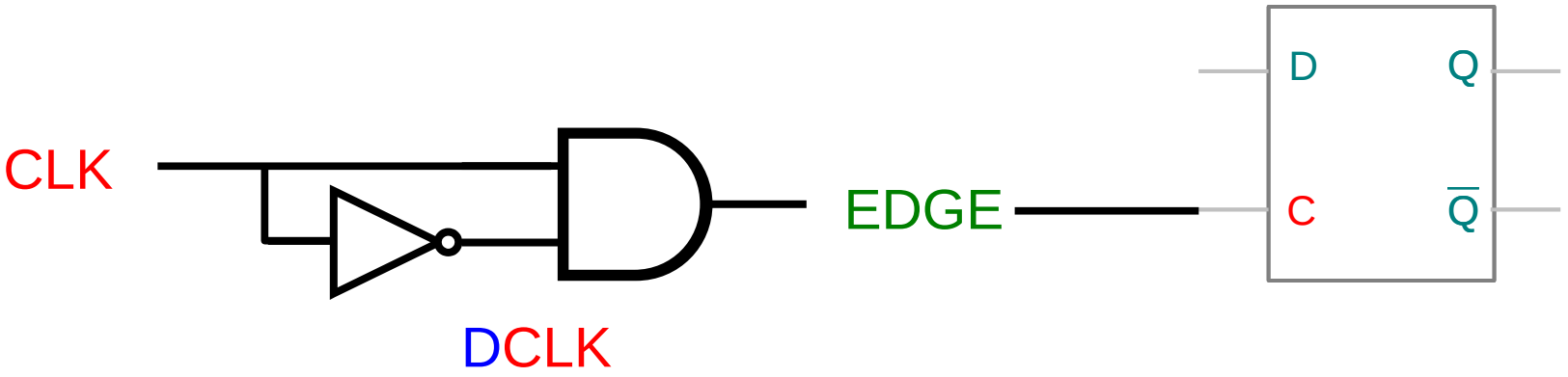
Edge Detector (1)



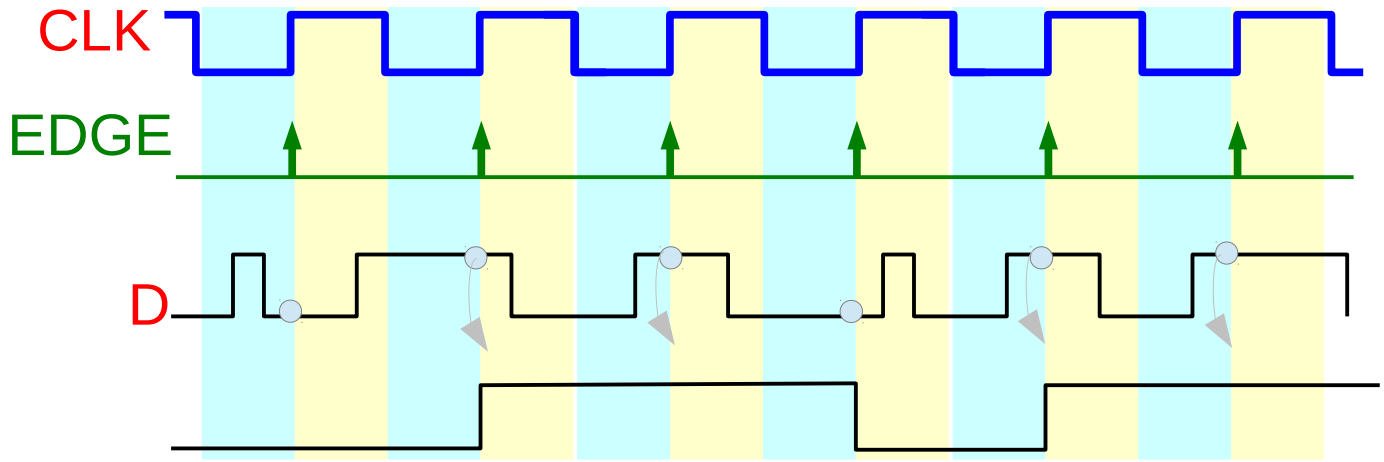


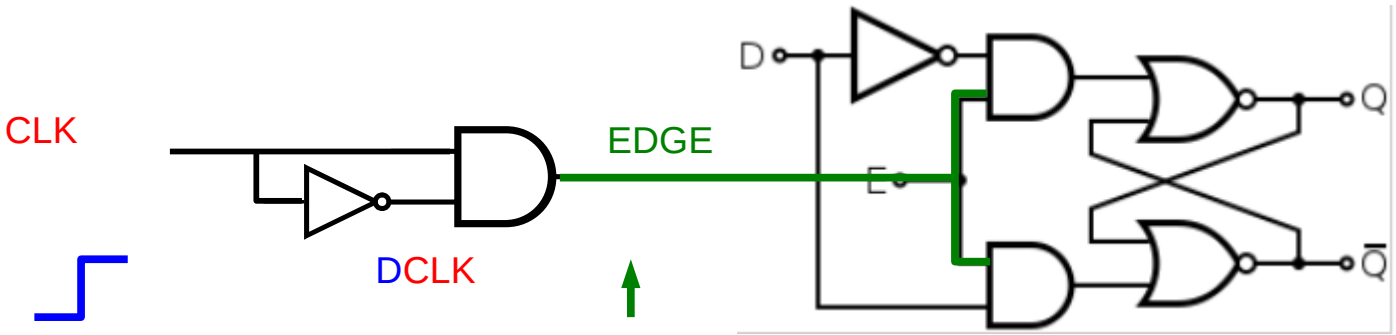
Edge Detector (2)



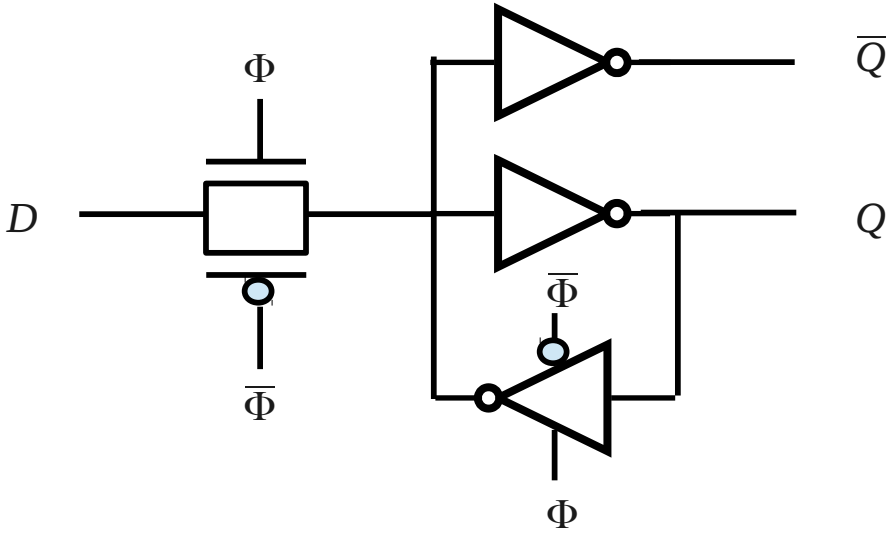


Edge Detector + D Latch = D FlipFlop

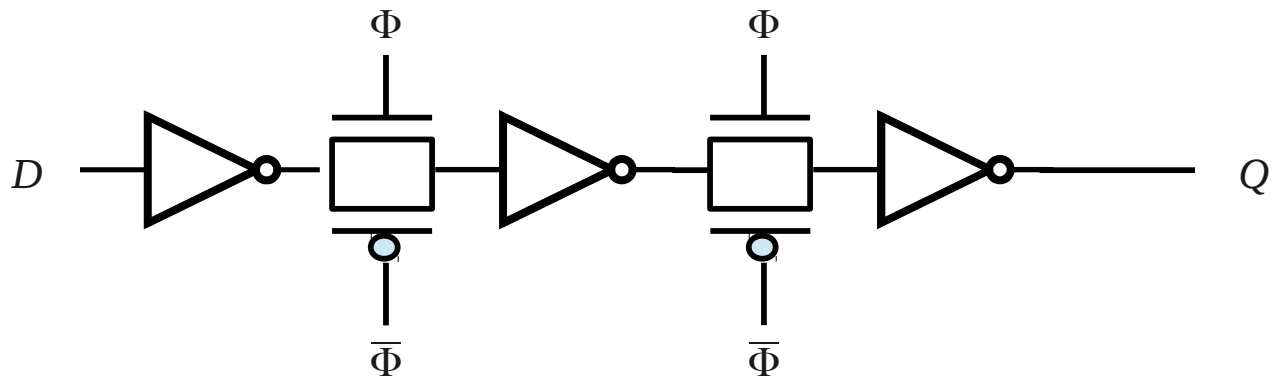


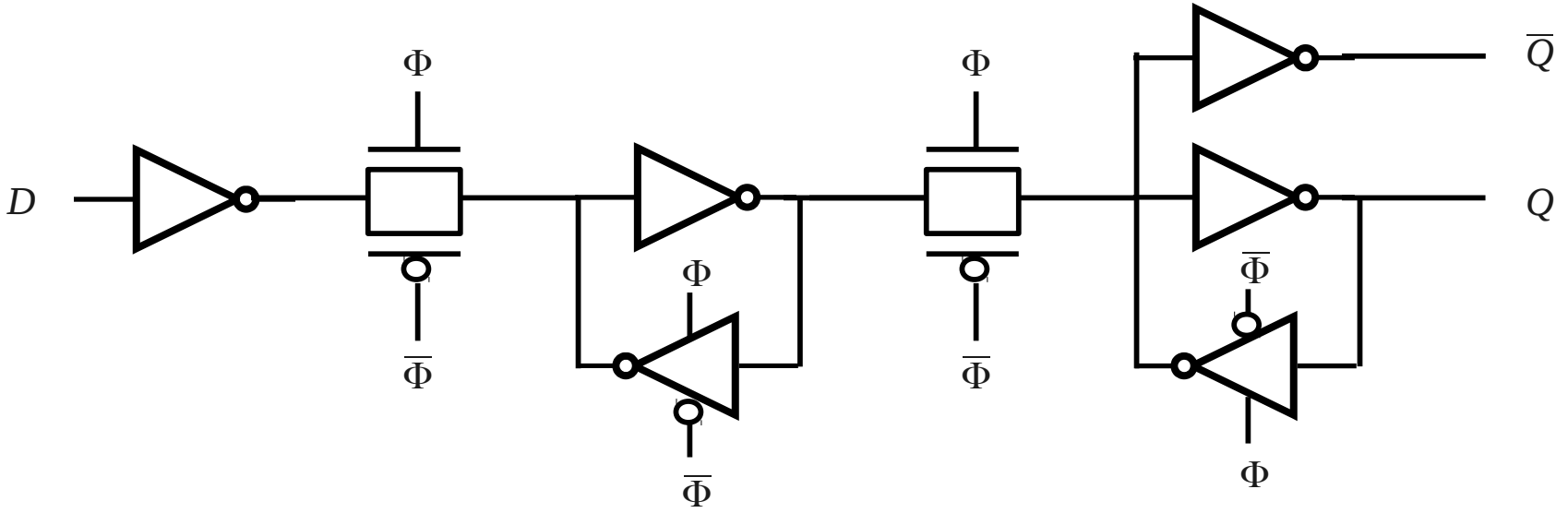


CMOS Latch Design



CMOS FlipFlop Design





Modern Latch Design

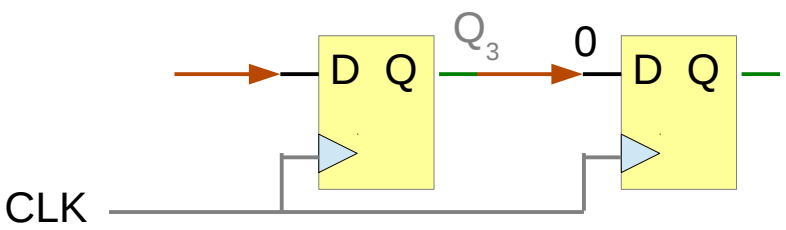
Registers

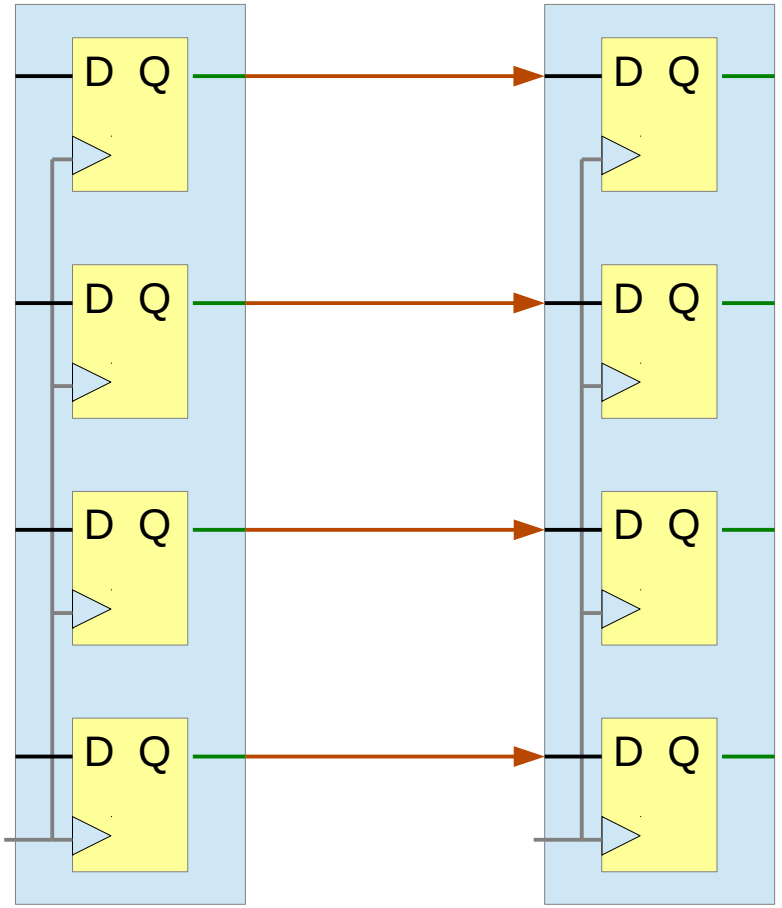
Toggling Input

Shift Register
Feedback Flip Flop
- JK Flip Flip
- T Flip Flop
- Toggling D Flip Flop

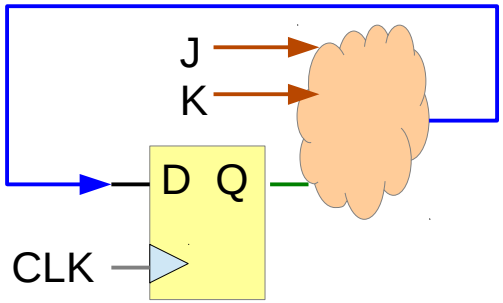
Pipeline Stage Register
Feedback Register
- FSM State Register
- Counter Register
-

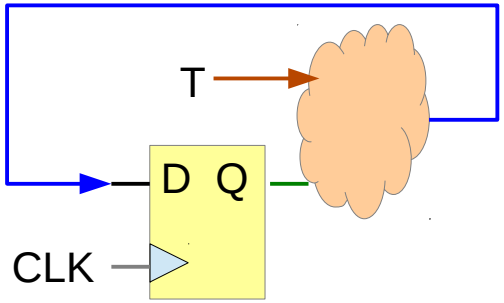
Shift Register v.s. Pipeline Stage Register

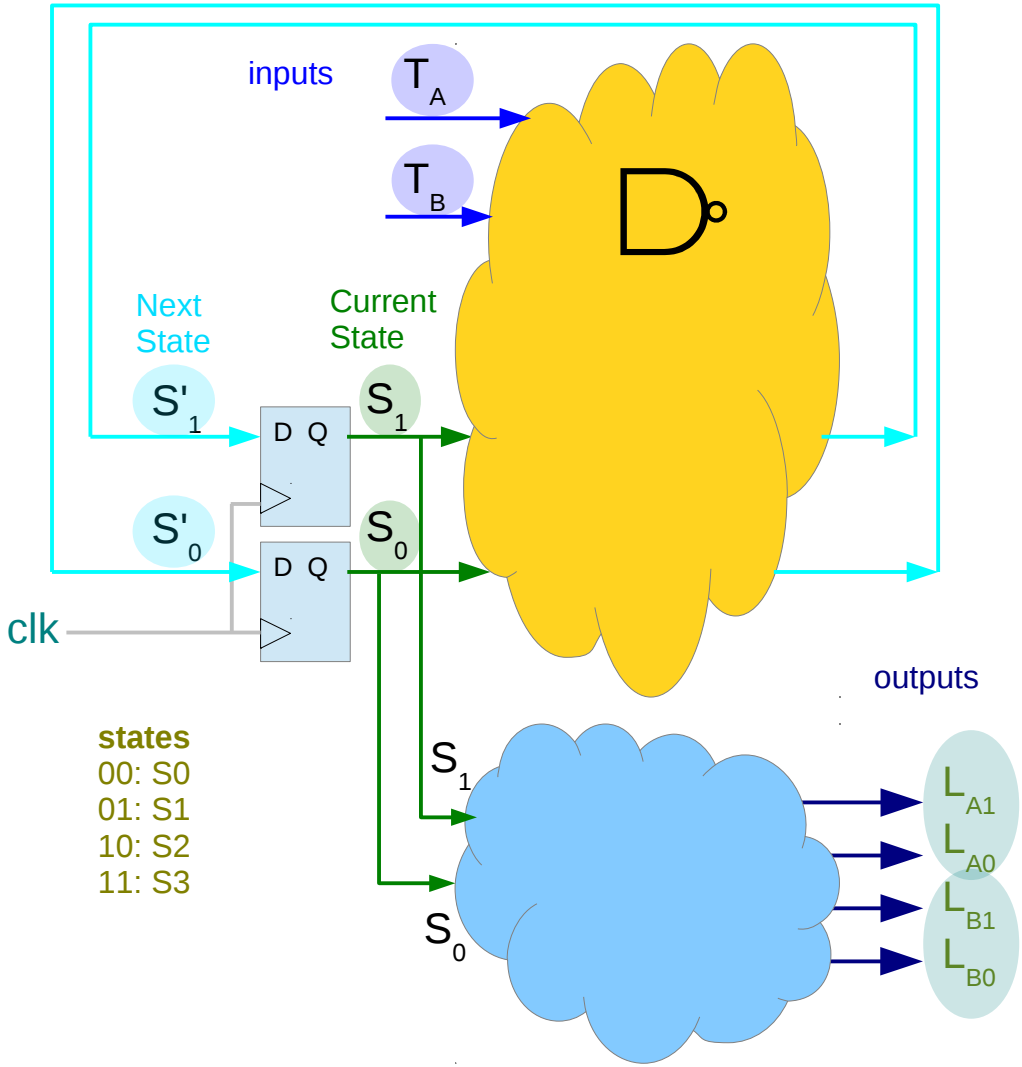




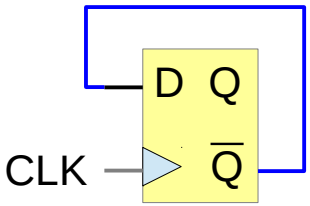
JKFF & TFF v.s. FSM

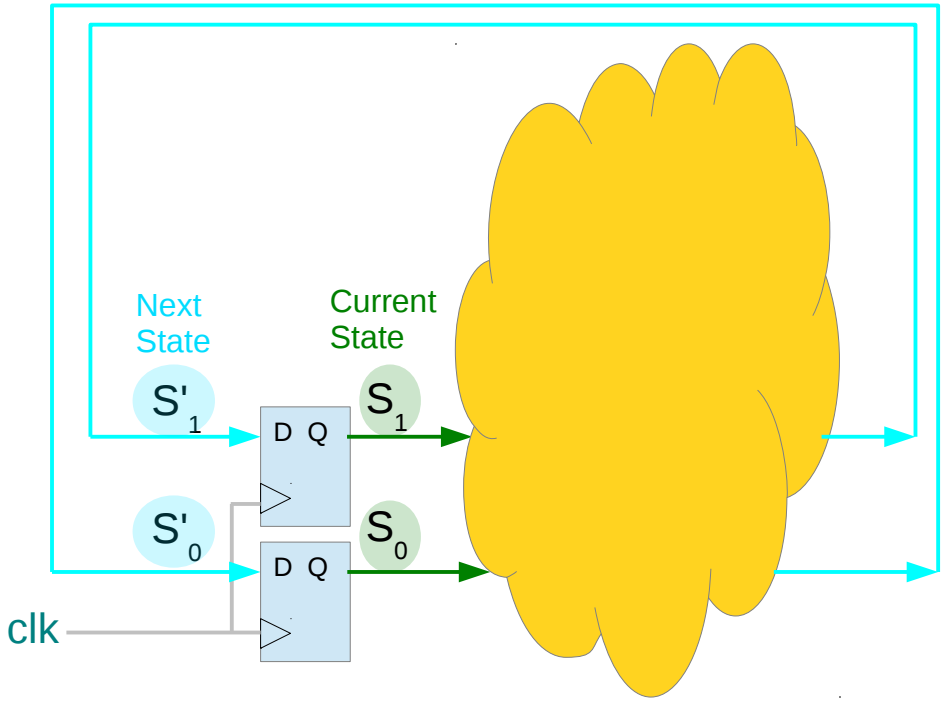




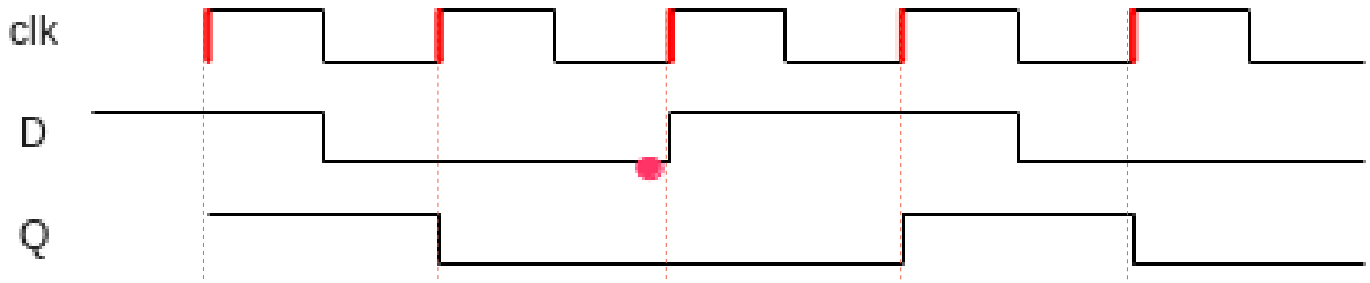


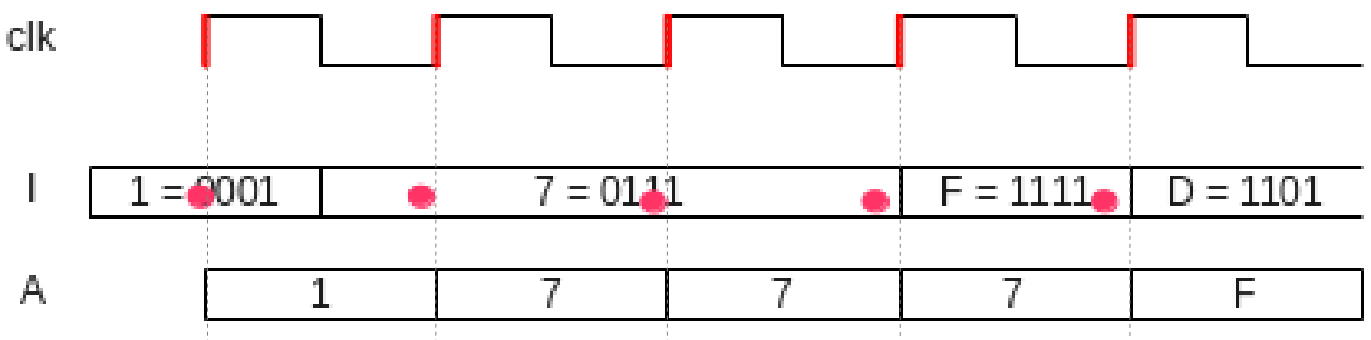
Toggling DFF v.s. Counter

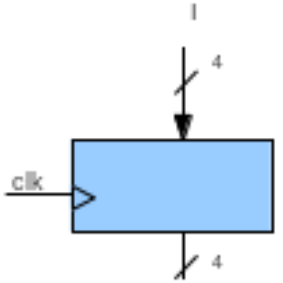




FF and Register Timing

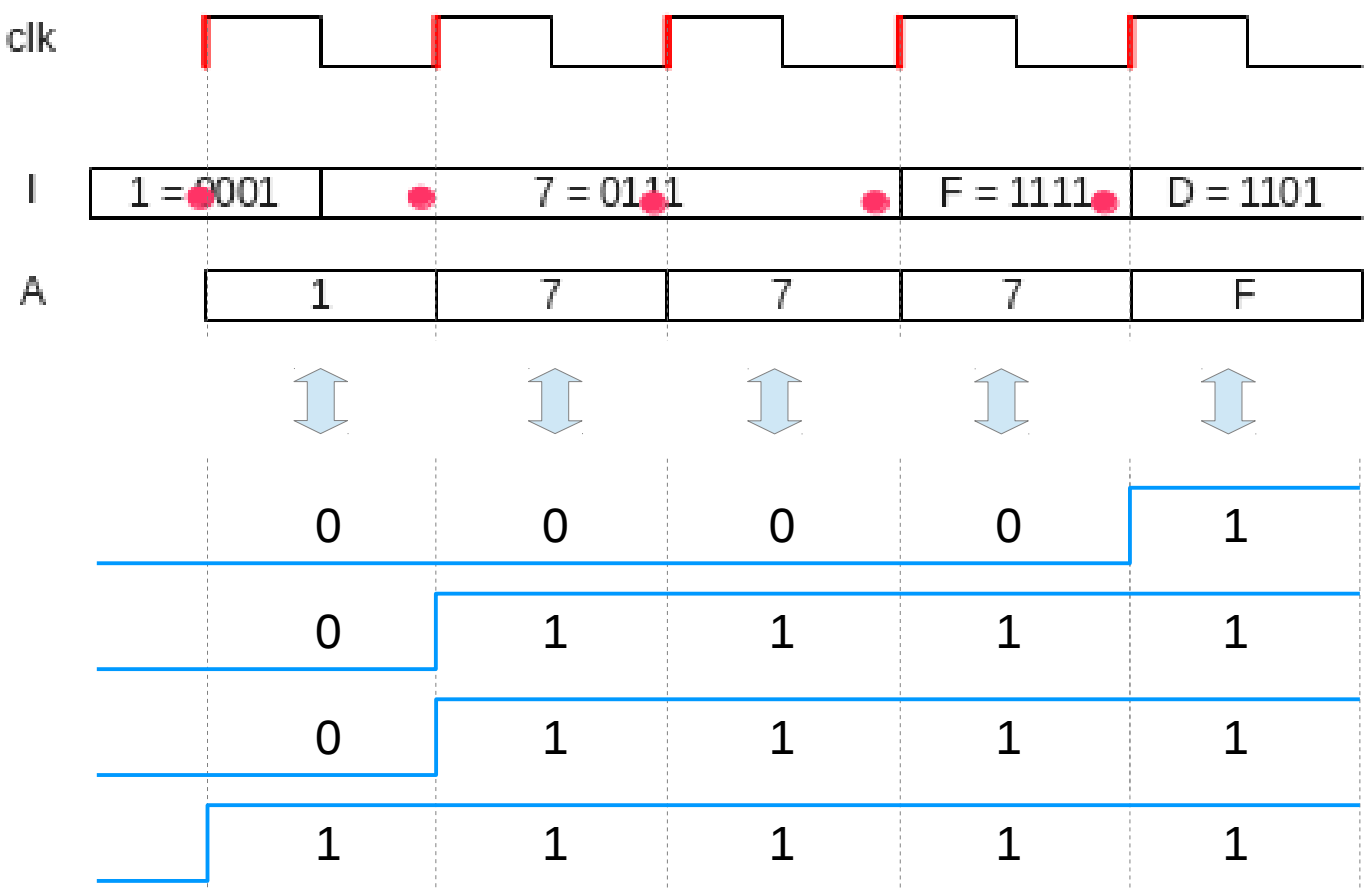


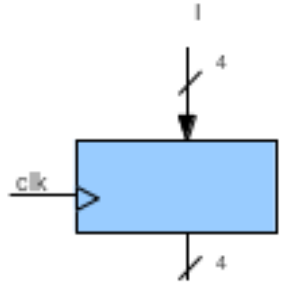




Decimal

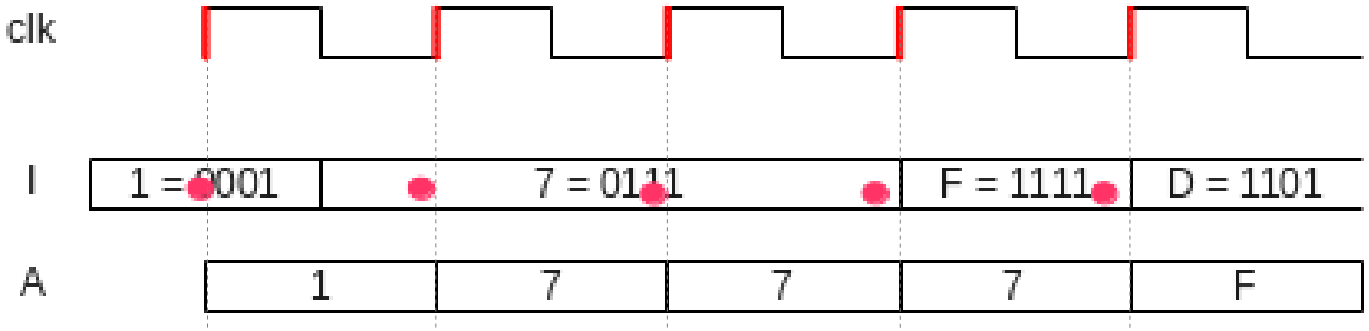
Bus Notation

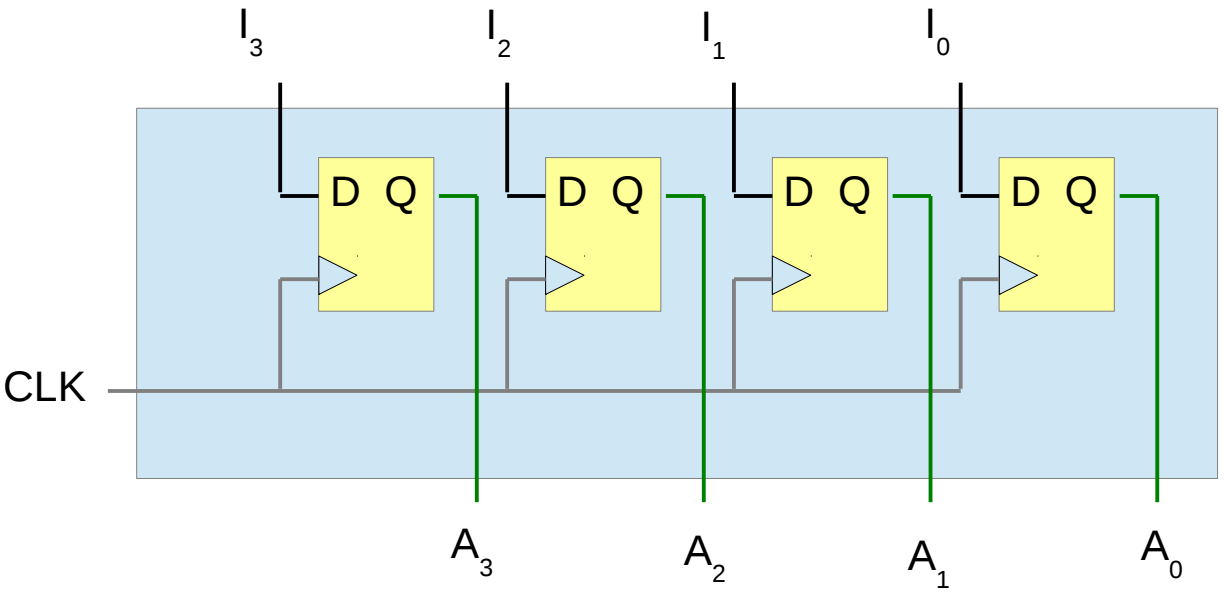


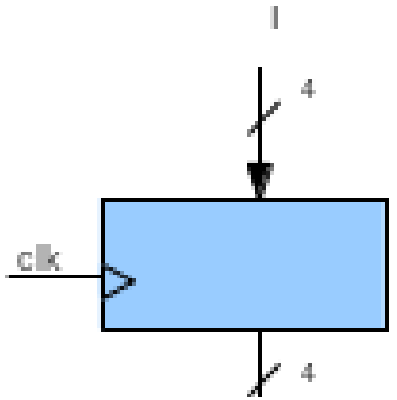


Decimal

Register

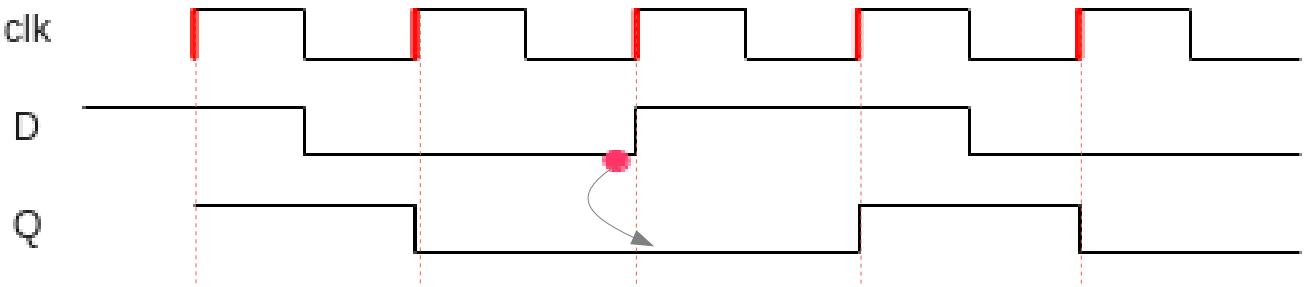


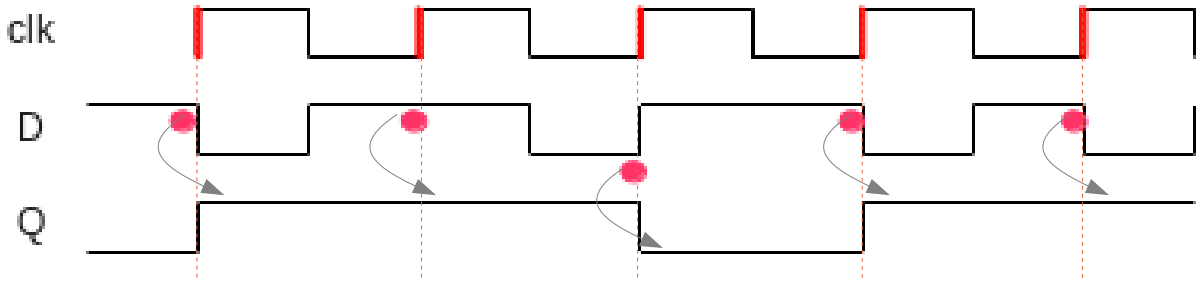


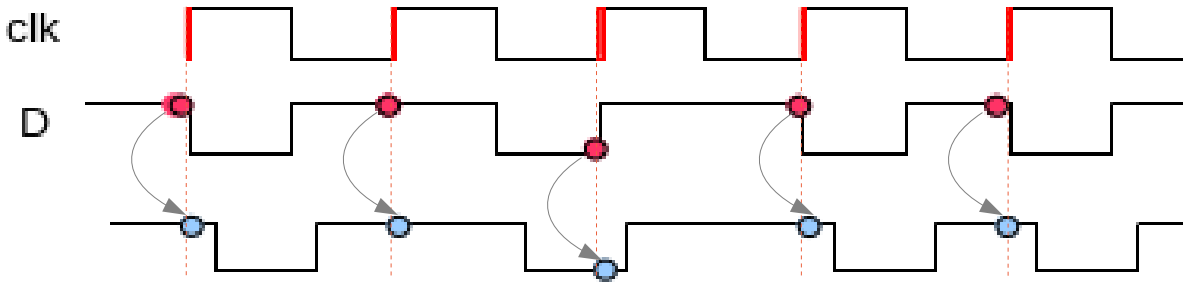


Decimal

FF Timing





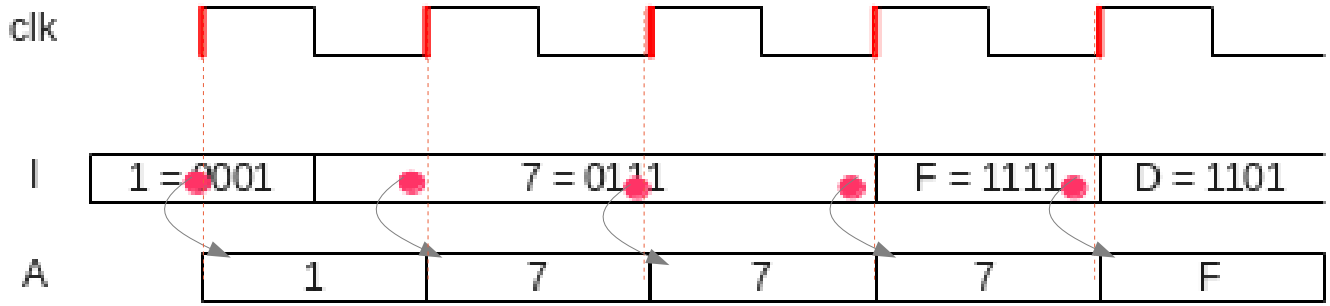


input signal with a delay ignored (ideal case)

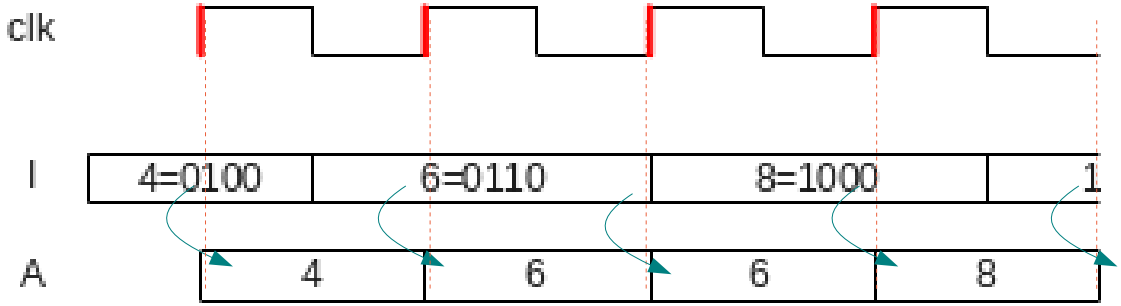
input signal with a delay explicitly shown

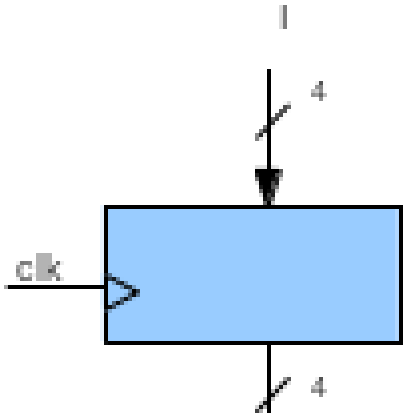
Register Timing

input signal with a delay explicitly shown

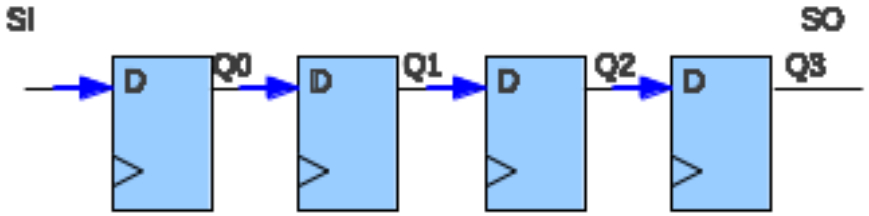


input signal with a delay explicitly shown

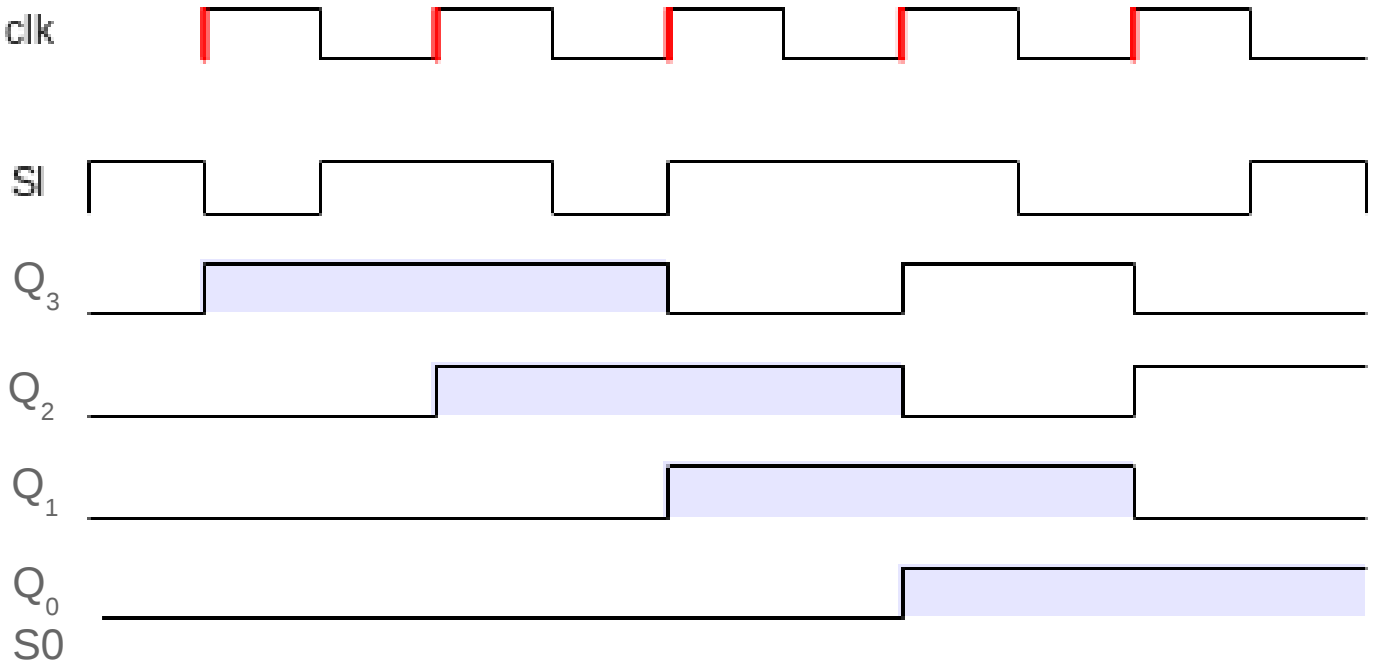




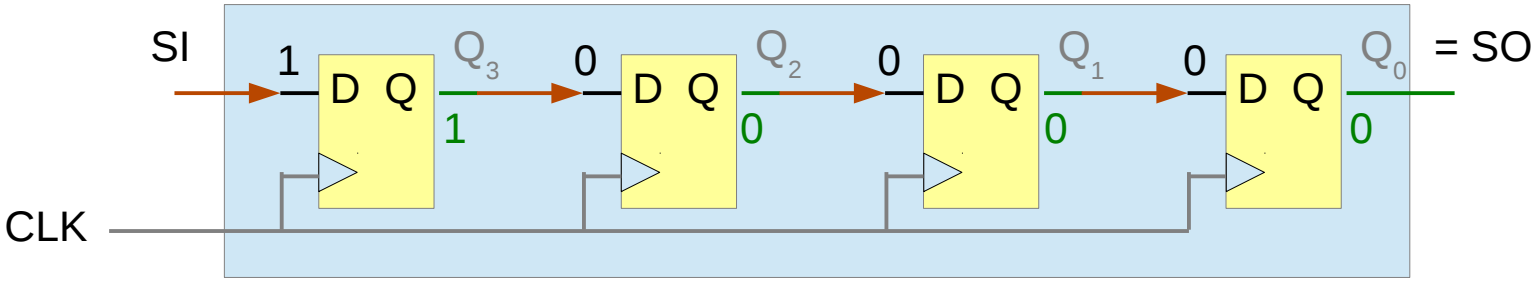
Shift Register Timing



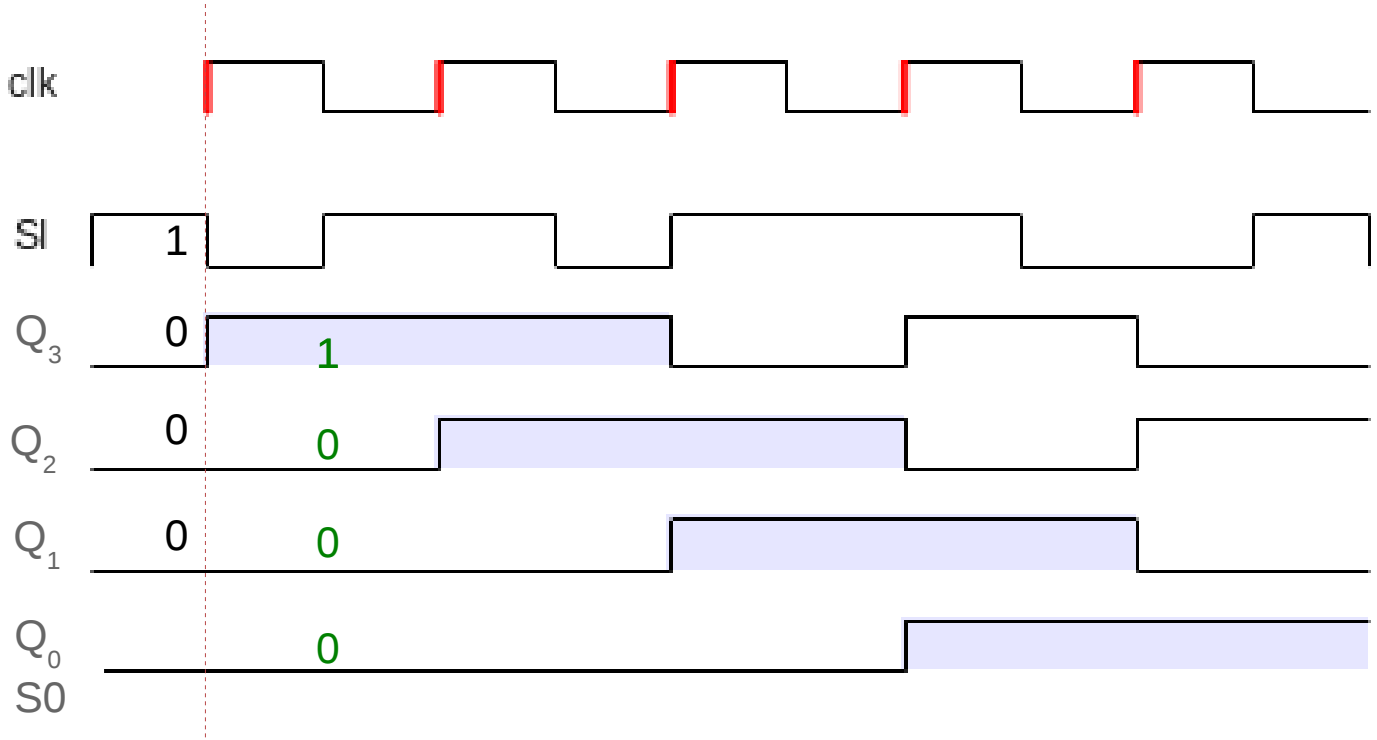
Connected in serial,
but parallel assignments



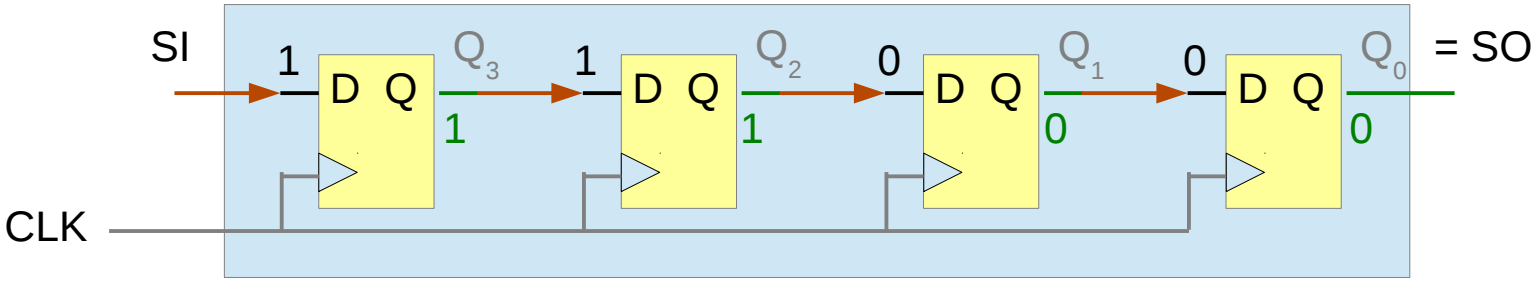
Shift Register Timing – Cycle 1



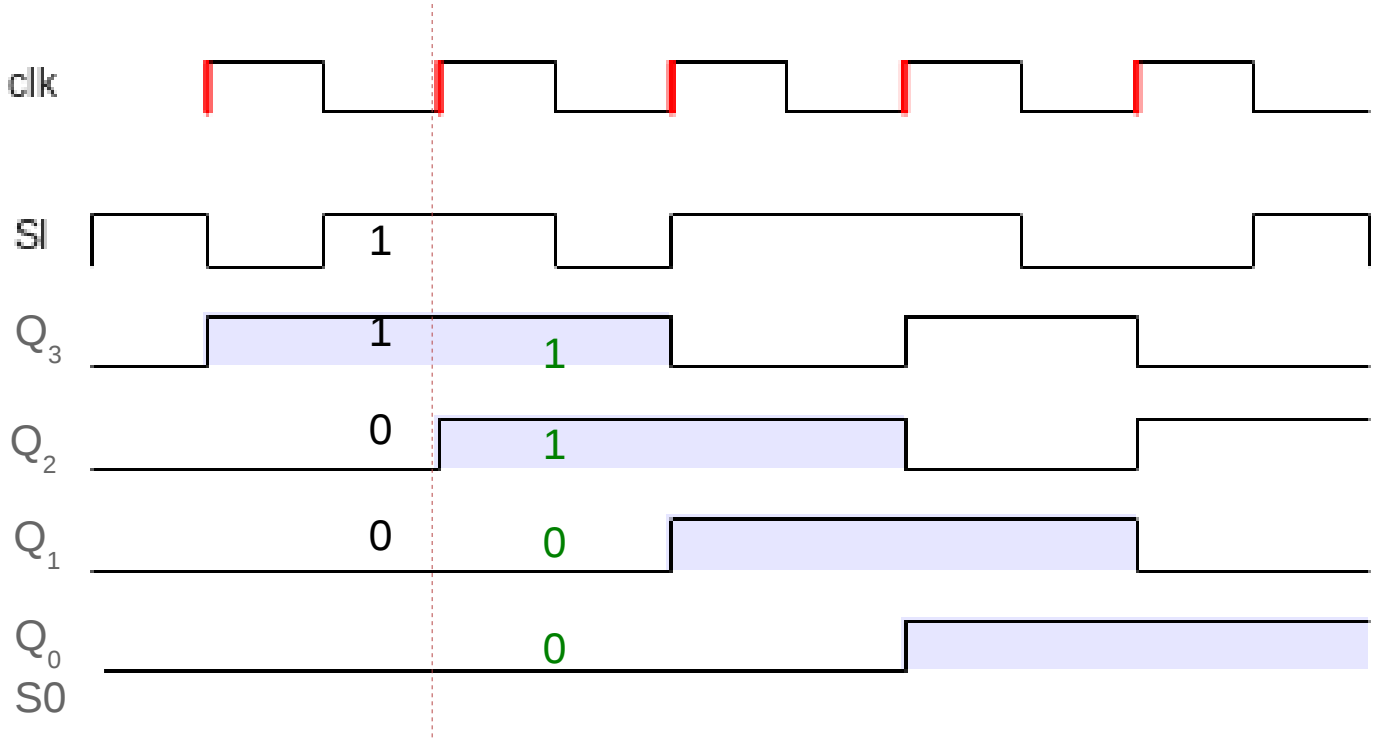
Connected in serial,
but parallel assignments



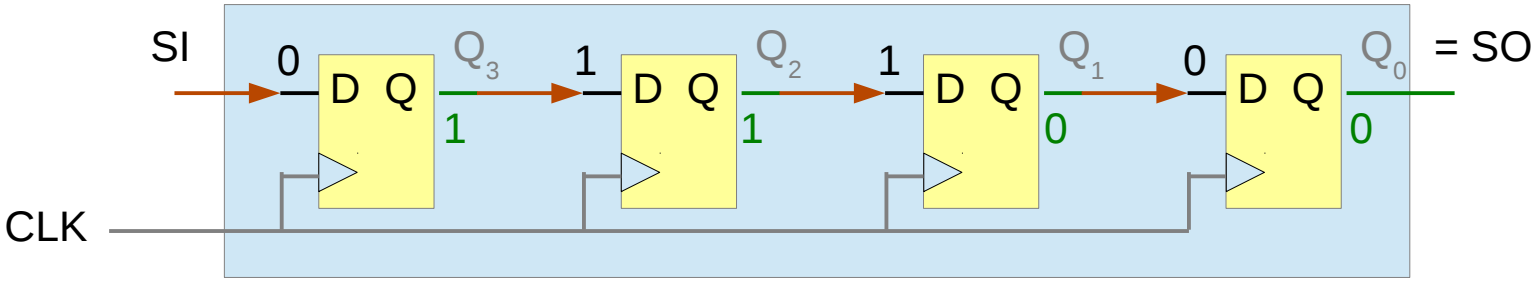
Shift Register Timing – Cycle 2



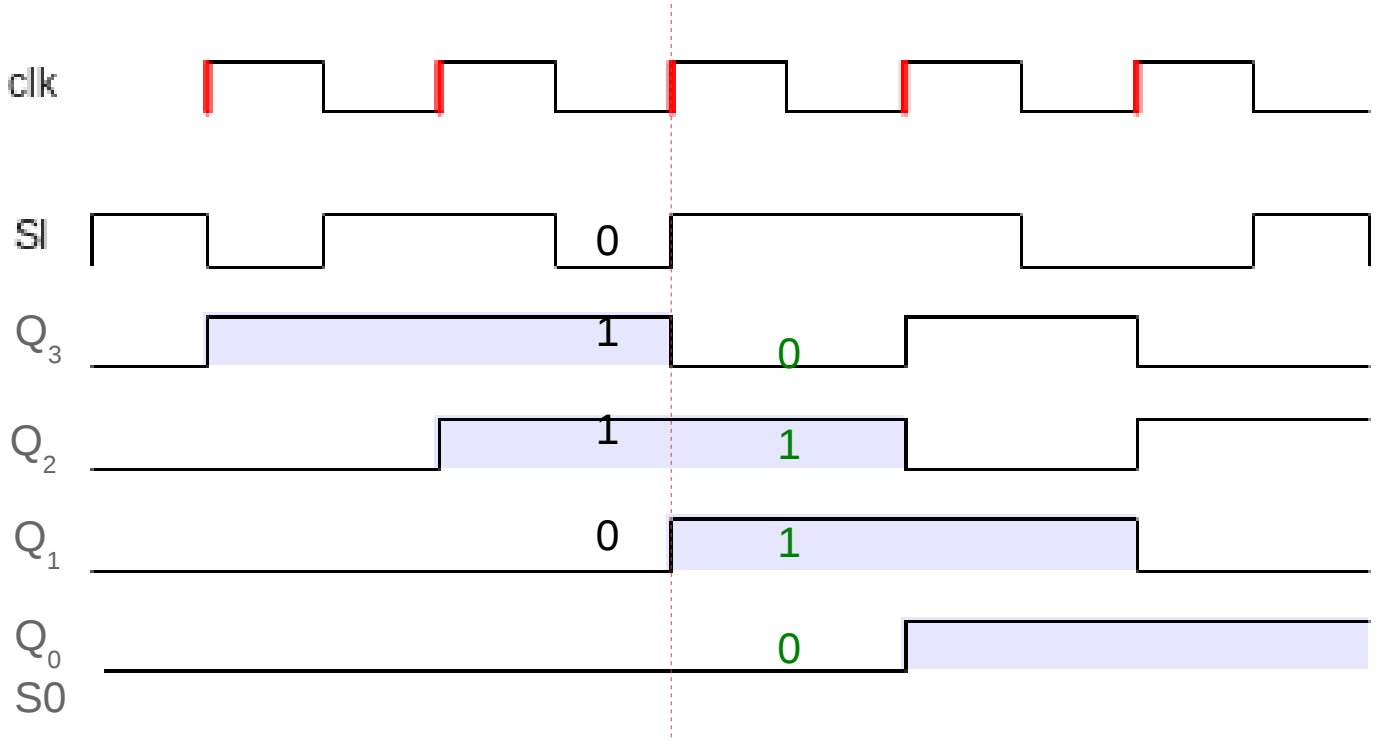
Connected in serial,
but parallel assignments



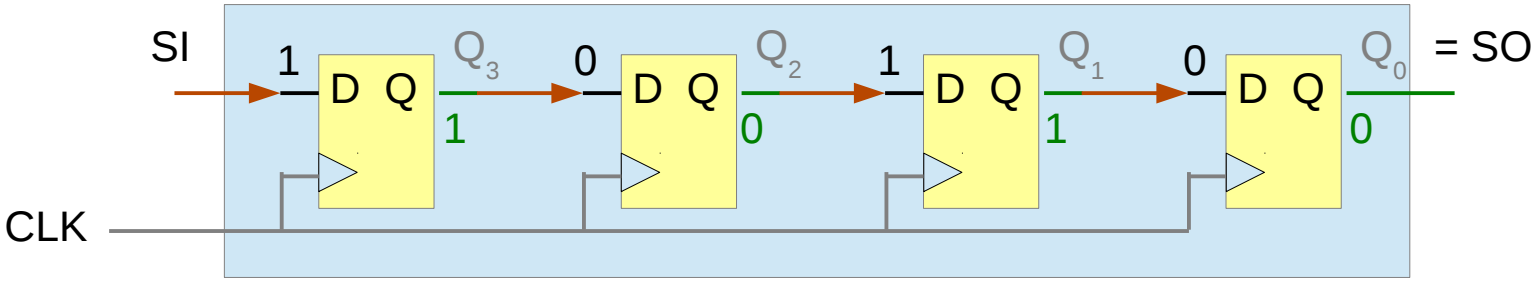
Shift Register Timing – Cycle 3



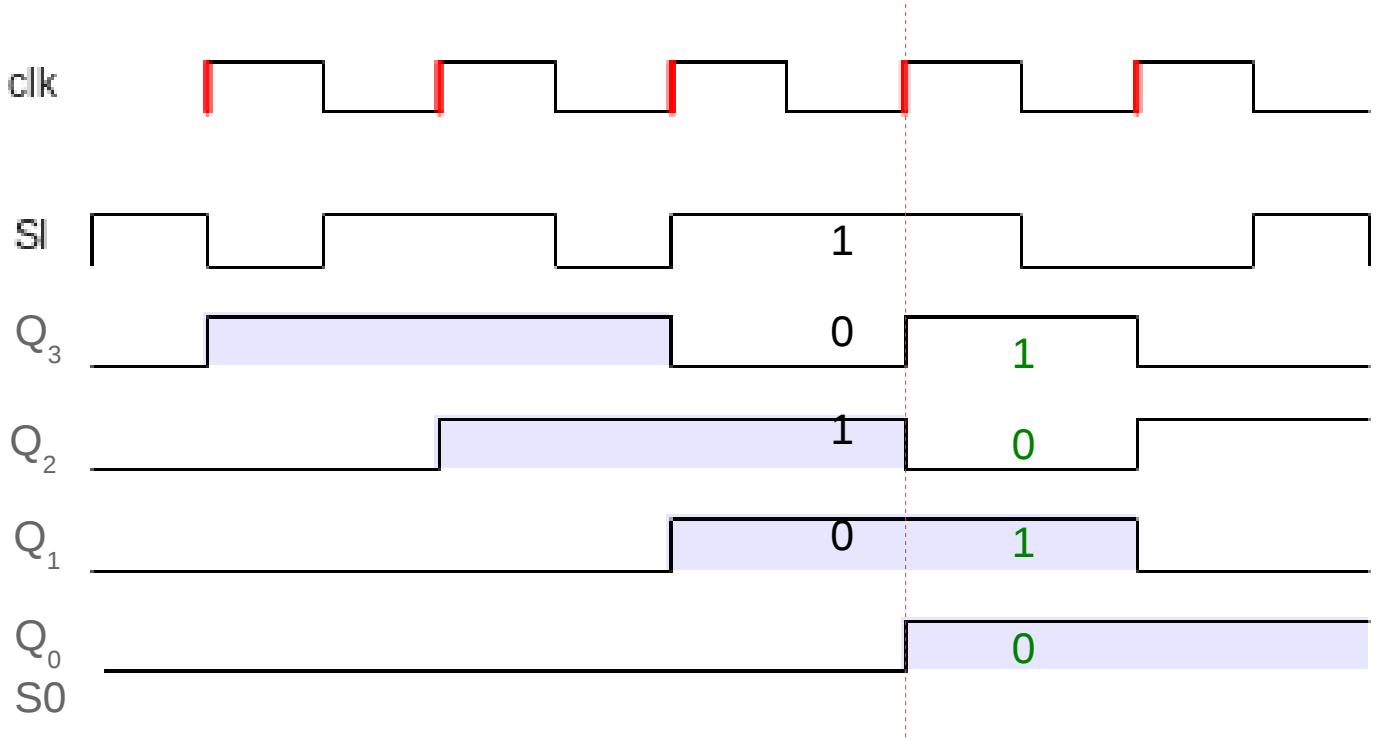
Connected in serial,
but parallel assignments



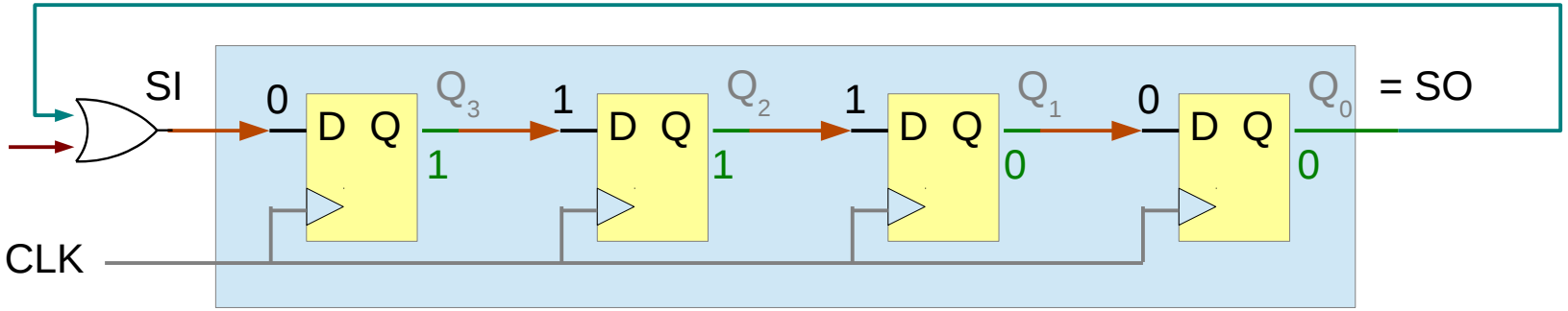
Shift Register Timing – Cycle 4



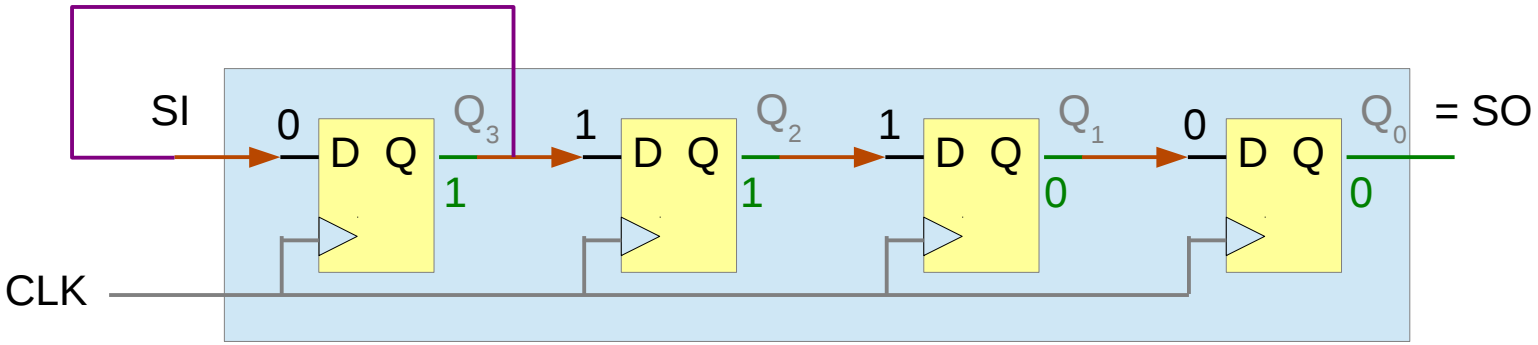
Connected in serial,
but parallel assignments



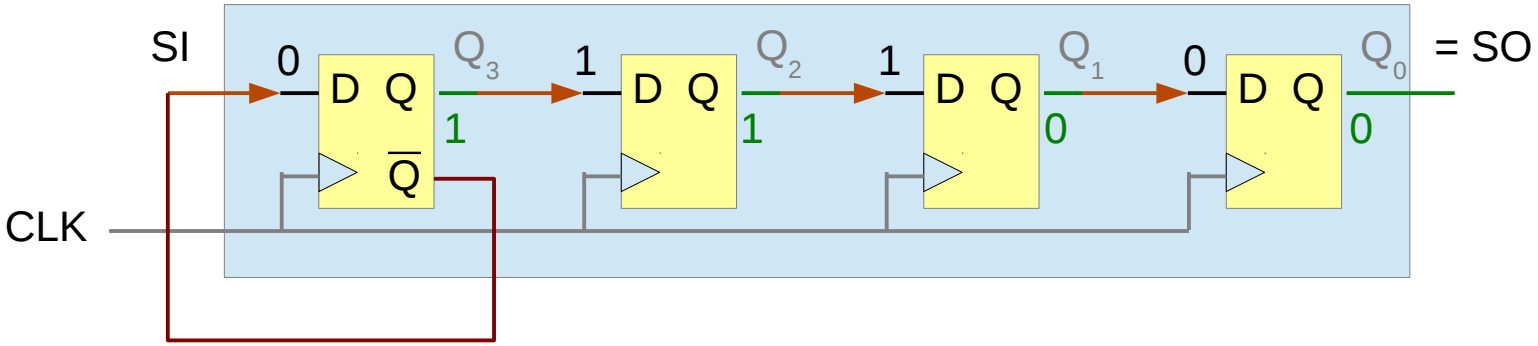
Rotate



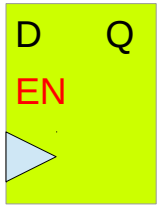
Divide By 2 with a Sign Extension

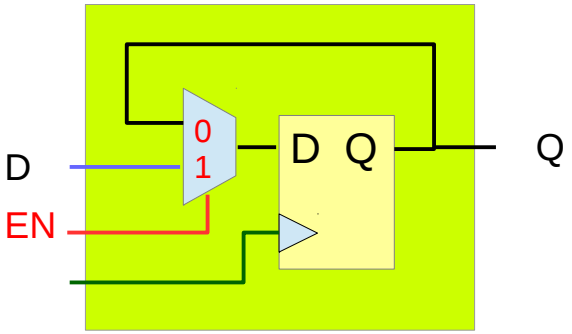


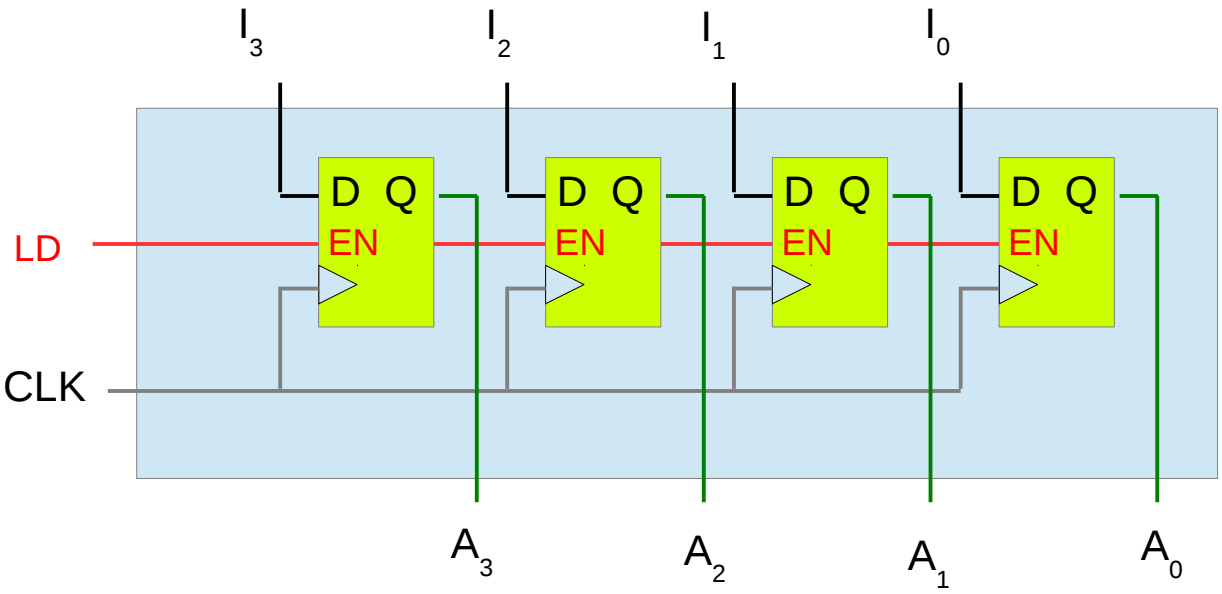
Toggling Input

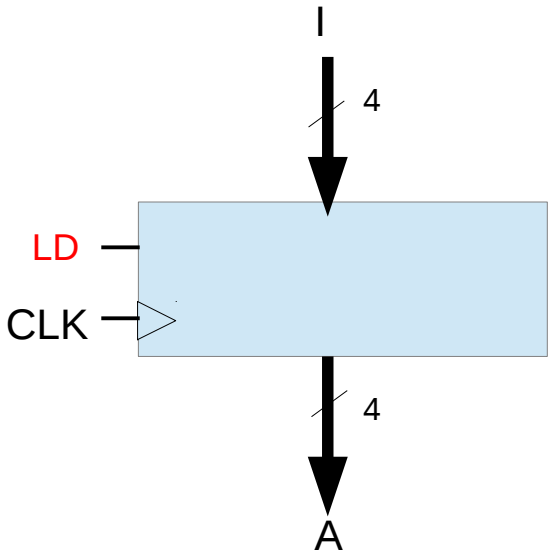


Register with Parallel Load

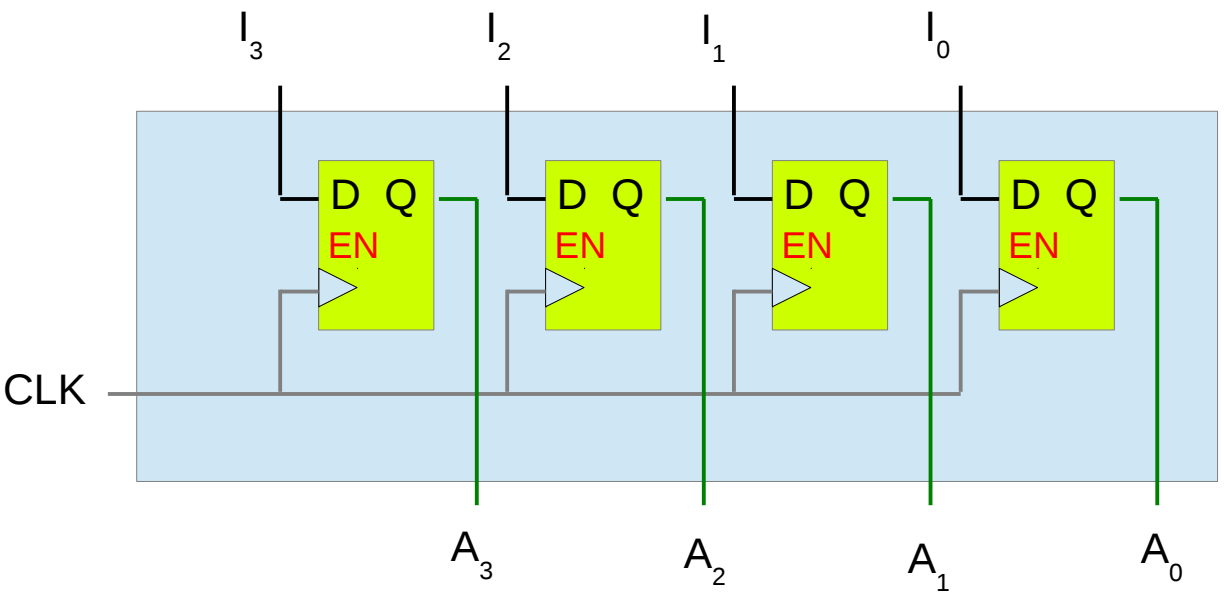


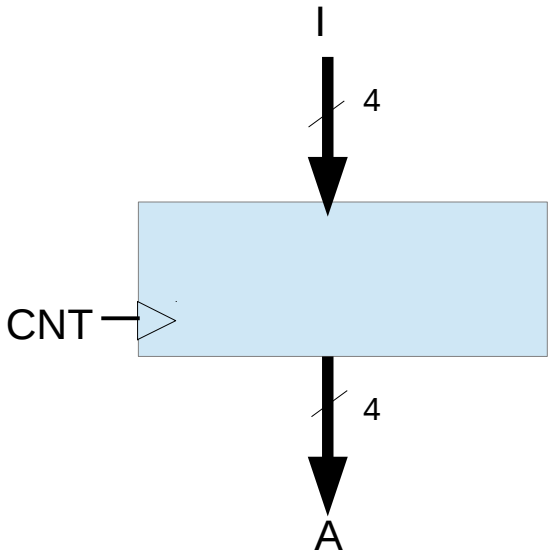




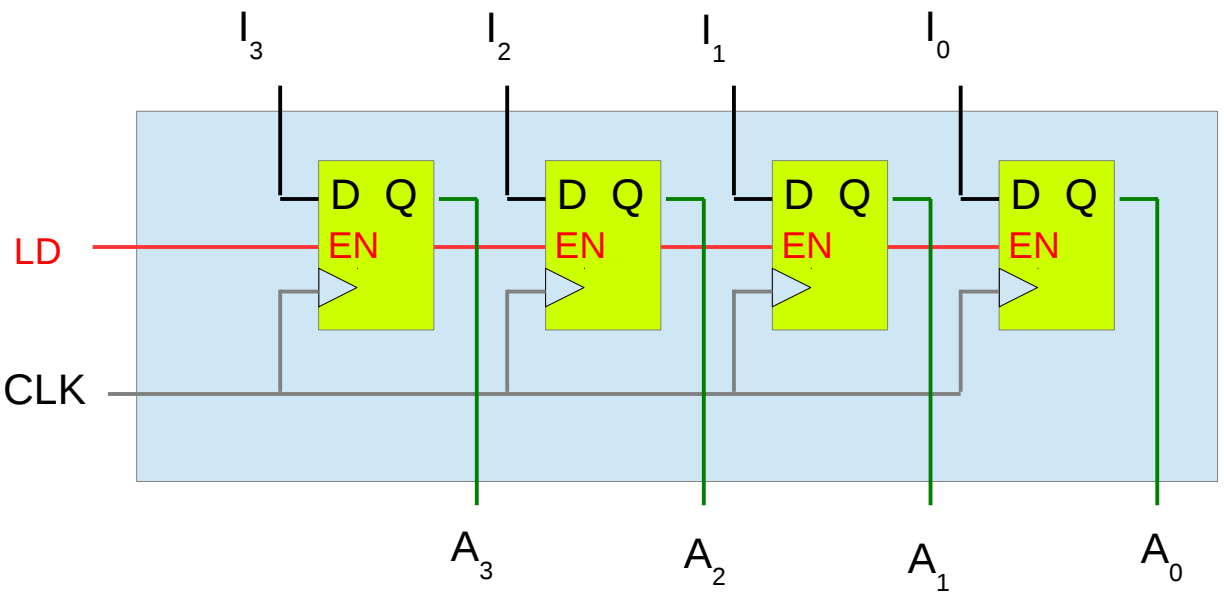


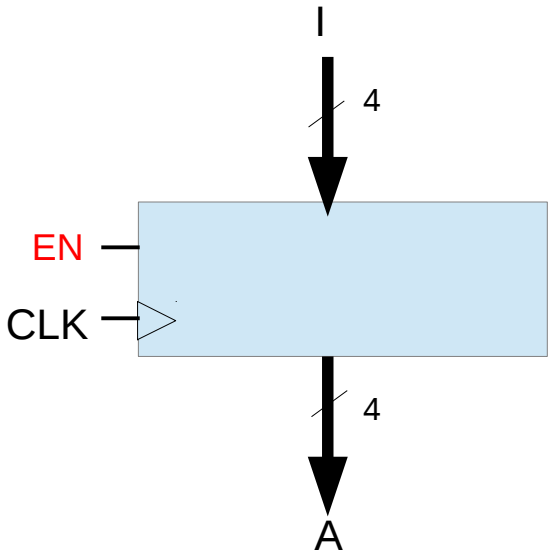
Ripple Counter





Synchronous Binary Counter





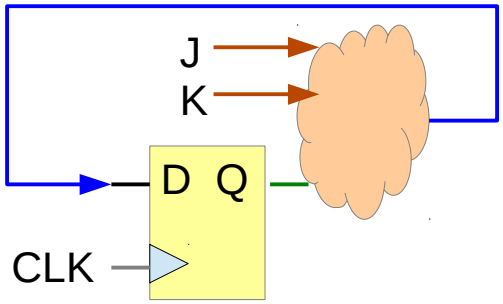
Counters

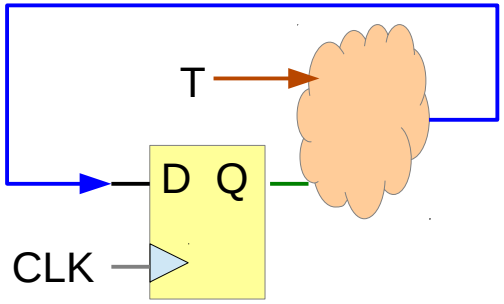
Toggling Input

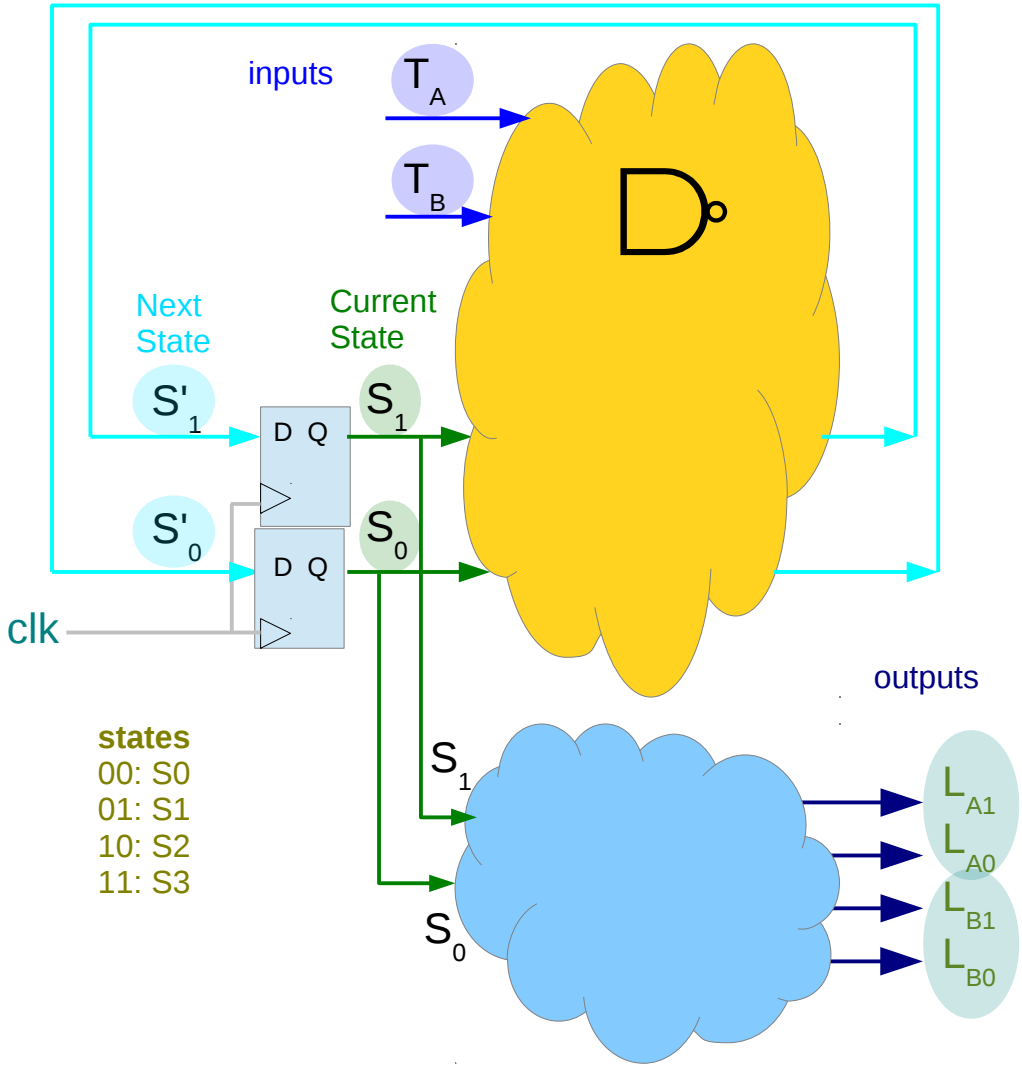
Shift Register
Feedback Flip Flop
- JK Flip Flip
- T Flip Flop
- Toggling D Flip Flop

Pipeline Stage Register
Feedback Register
- FSM State Register
- Counter Register
-

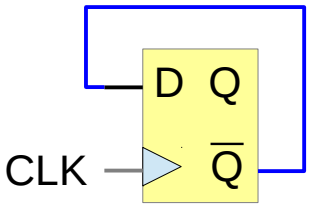
JKFF & TFF v.s. FSM

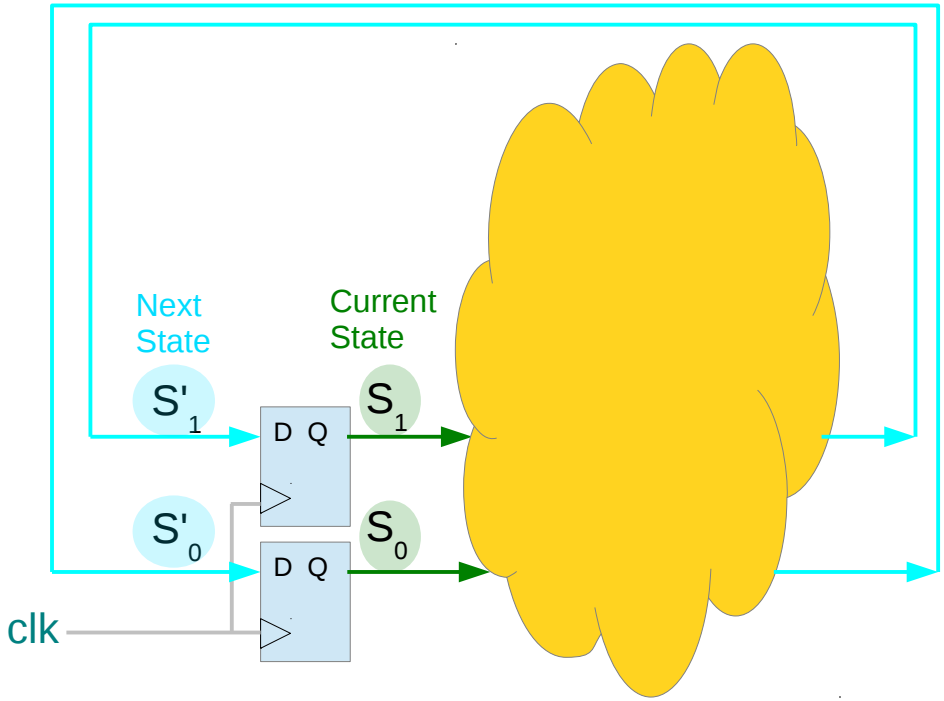






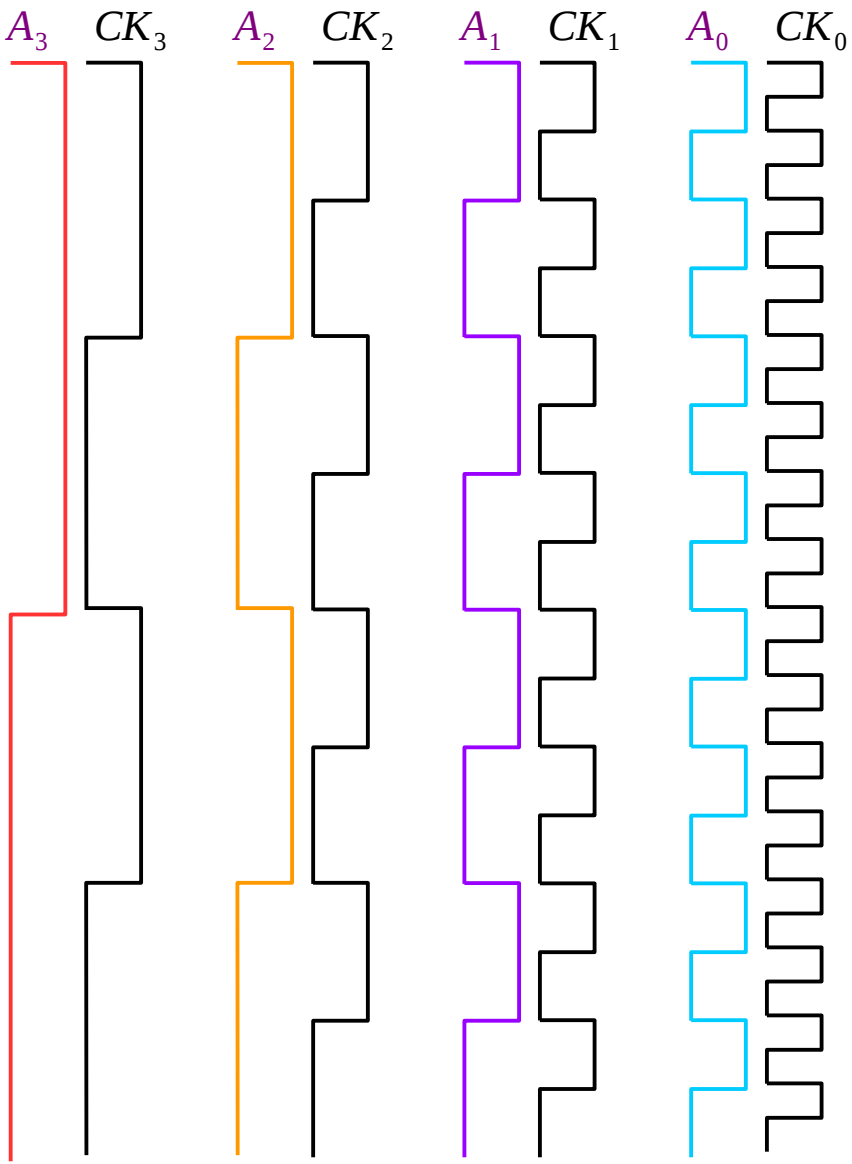
Toggling DFF v.s. Counter

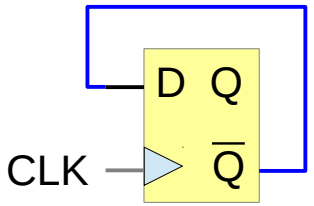




Toggle Count Down

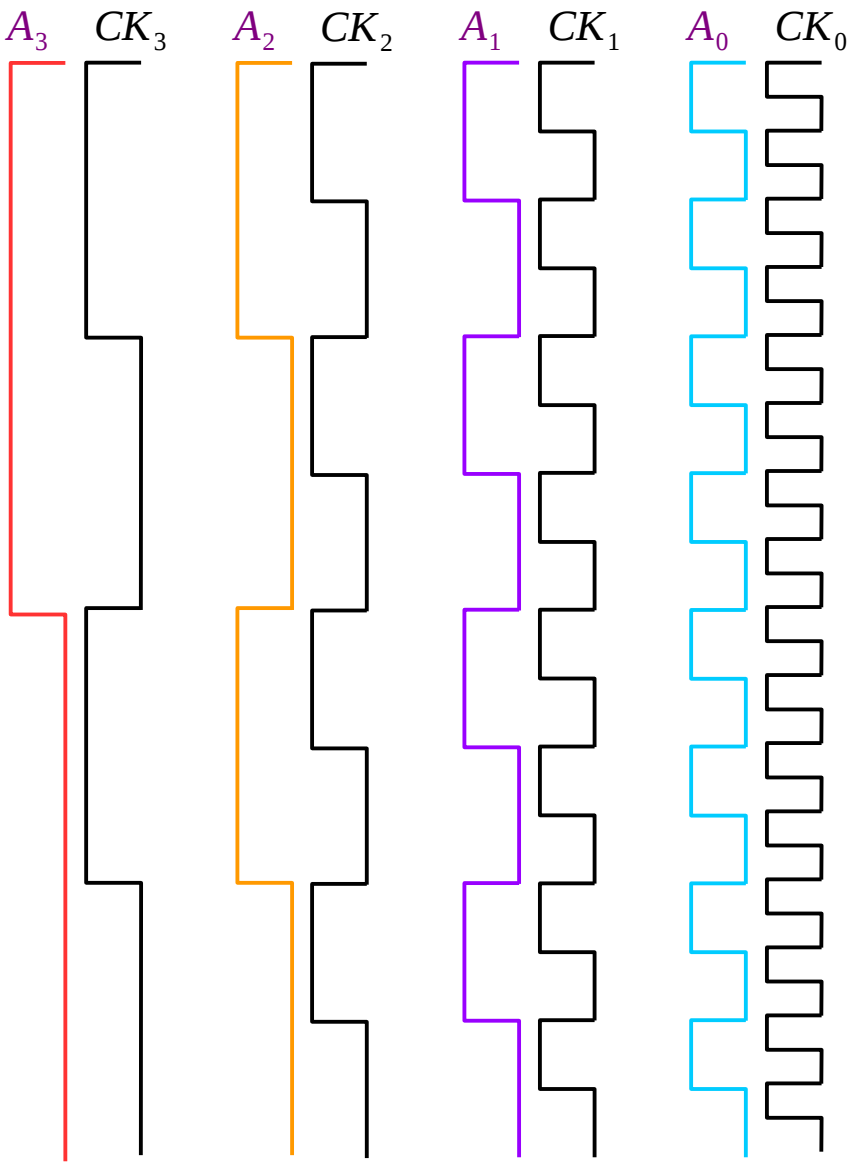
2^3	2^2	2^1	2^0
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0

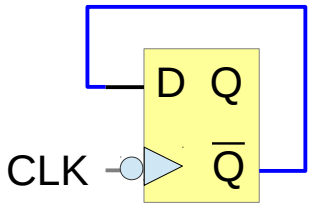




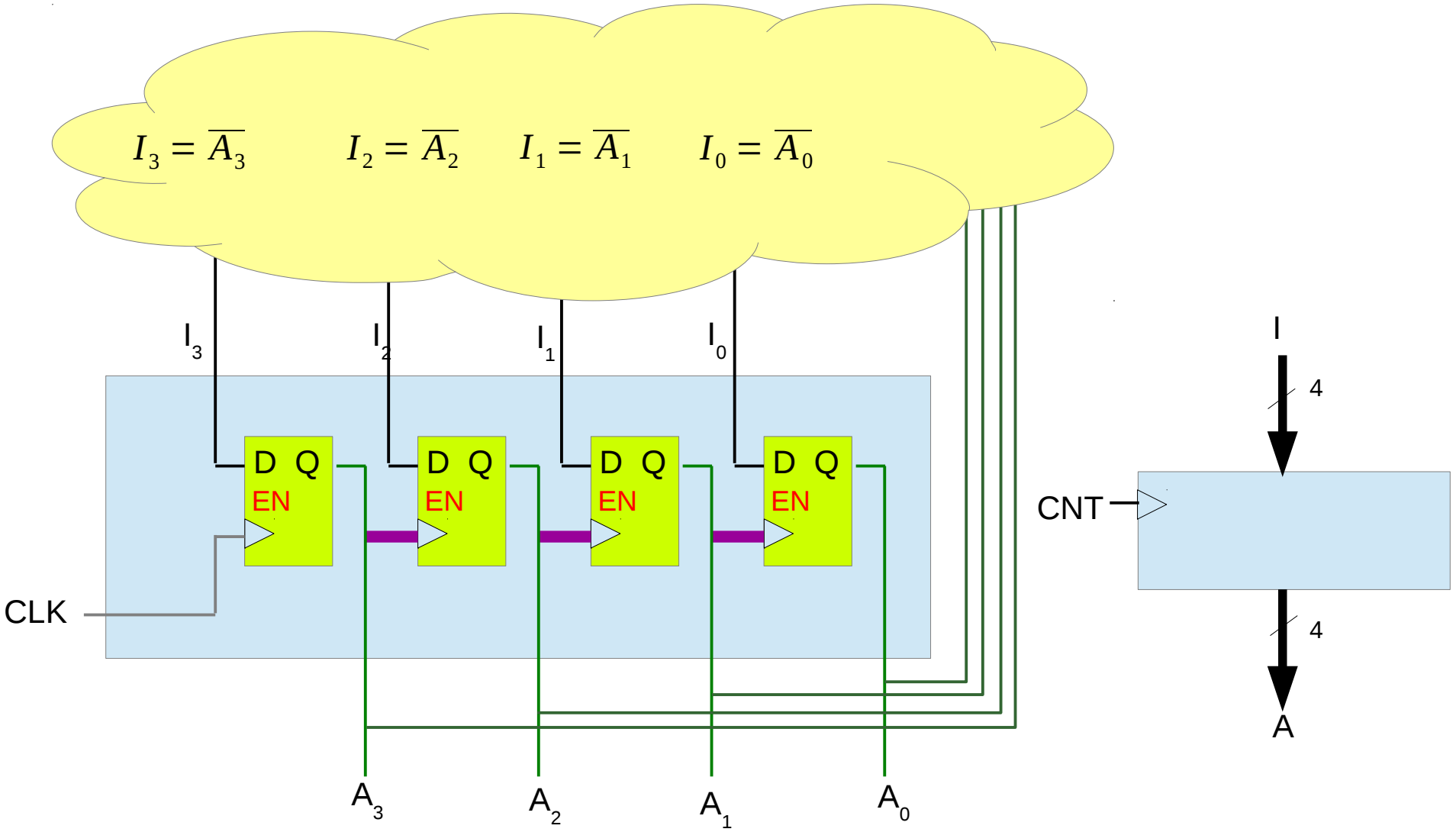
Toggle Count Up

2^3	2^2	2^1	2^0
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

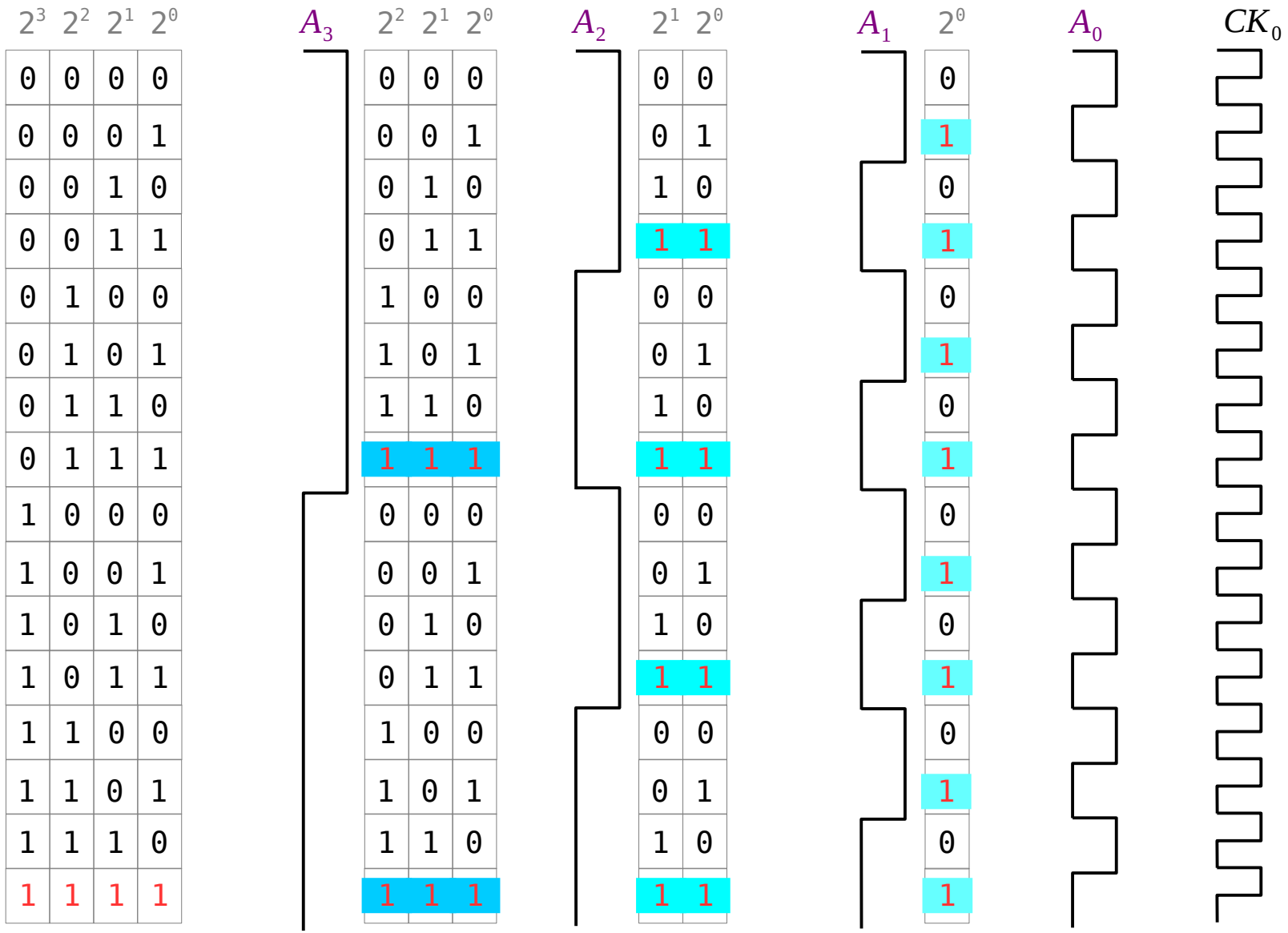




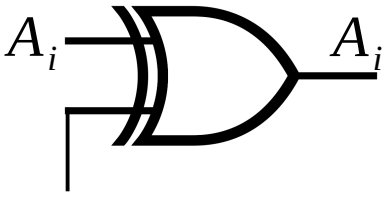
Ripple Counter – multiple clocks



Toggle Conditions



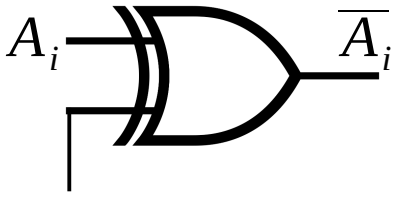
Toggle Conditions



$x \oplus 0 = x$

0

C_i



$x \oplus 1 = \bar{x}$

1

C_i

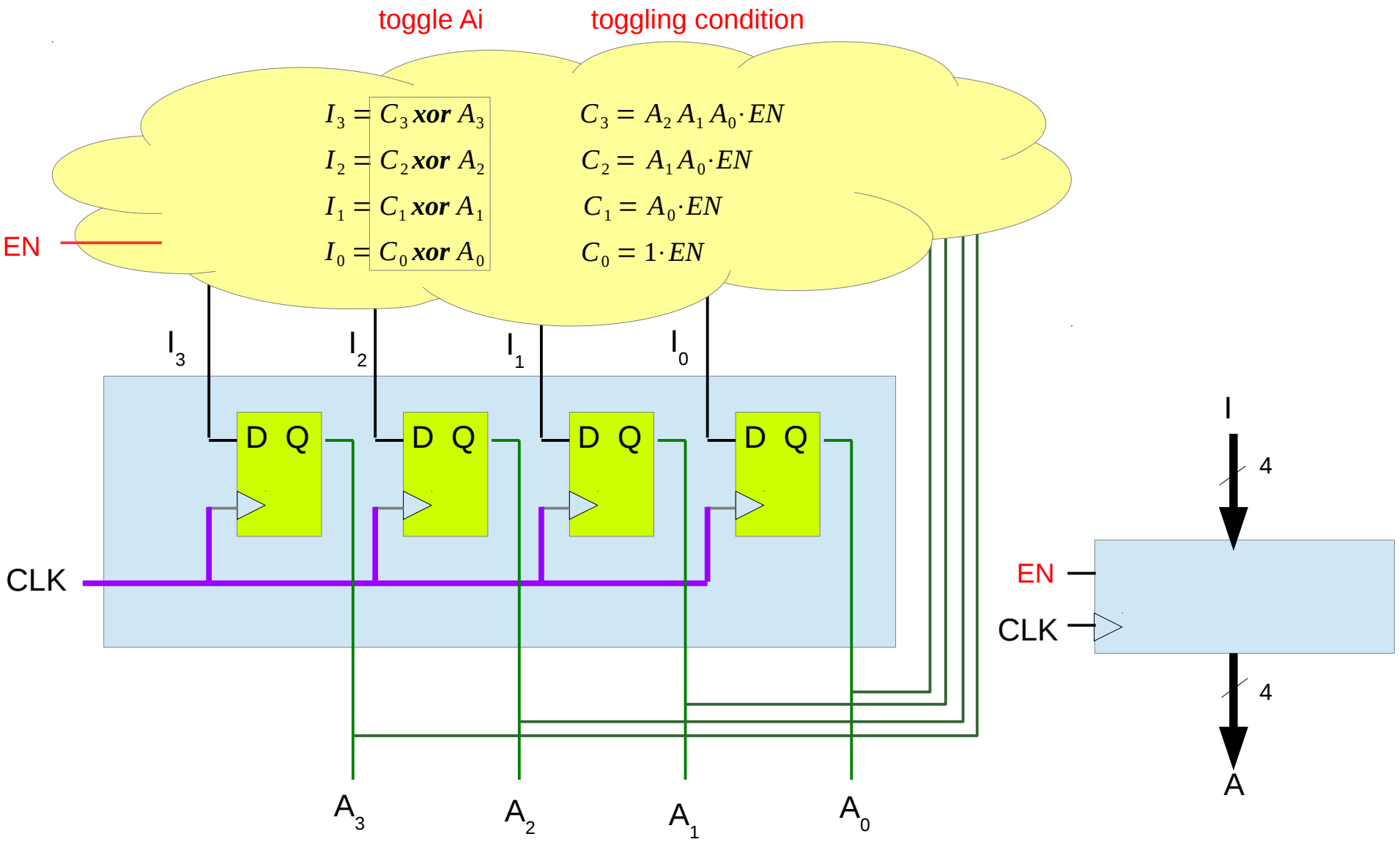
$$I_3 = C_3 \textbf{ xor } A_3 \qquad C_3 = A_2 A_1 A_0 \cdot EN$$

$$I_2 = C_2 \textbf{ xor } A_2 \qquad C_2 = A_1 A_0 \cdot EN$$

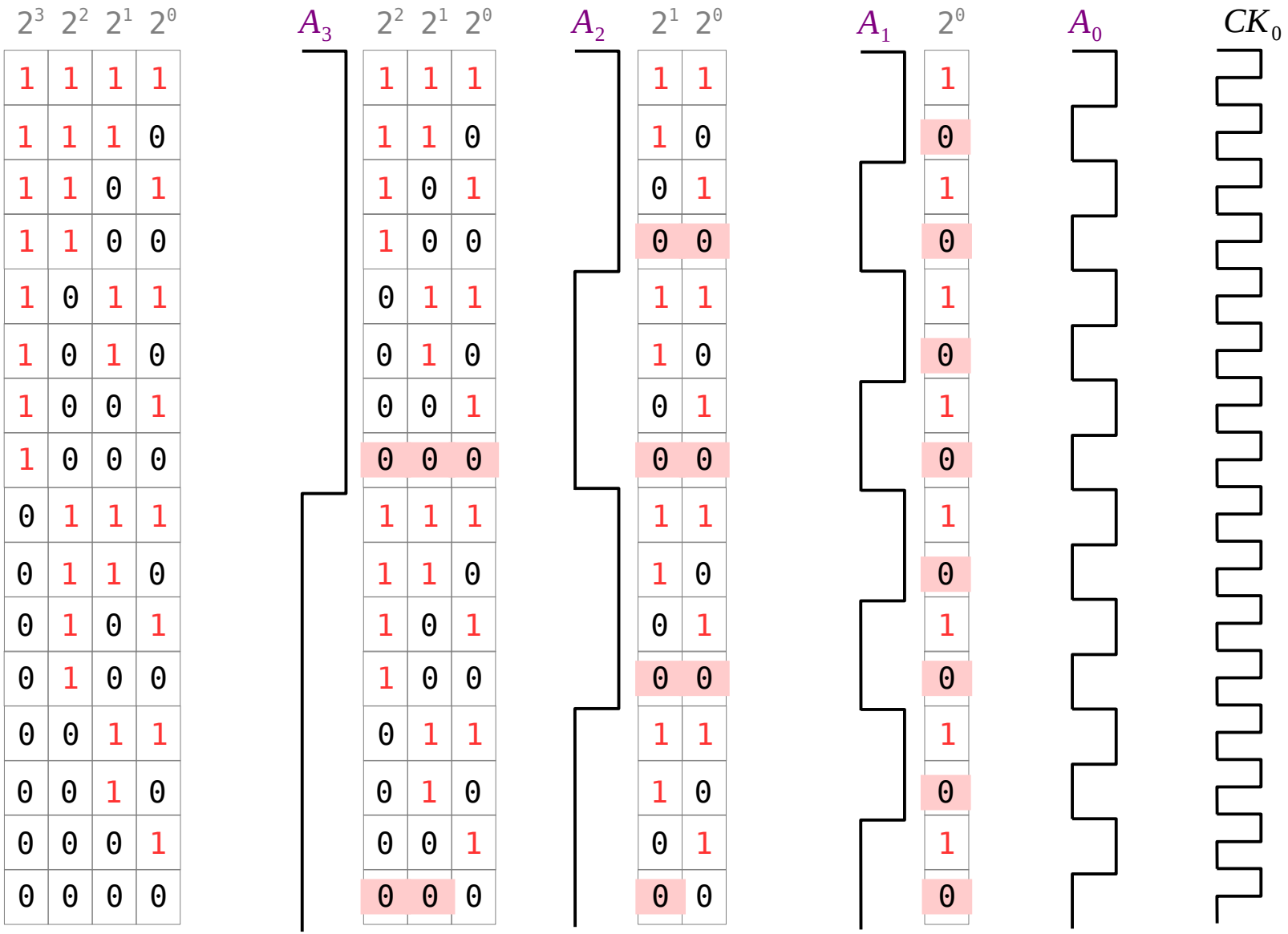
$$I_1 = C_1 \textbf{ xor } A_1 \qquad C_1 = A_0 \cdot EN$$

$$I_0 = C_0 \textbf{ xor } A_0 \qquad C_0 = 1 \cdot EN$$

Synchronous Binary Counter – a single clock



Toggle Conditions



Synchronous UpDown Counter – a single clock

toggle Ai | up counting condition | down counting condition

$$I_3 = C_3 \textbf{ xor } A_3 \qquad C_3 = A_2 A_1 A_0 \cdot EN \qquad C_3 = \overline{A_2}\,\overline{A_1}\,\overline{A_0} \cdot EN$$
$$I_2 = C_2 \textbf{ xor } A_2 \qquad C_2 = A_1 A_0 \cdot EN \qquad C_2 = \overline{A_1}\,\overline{A_0} \cdot EN$$
$$I_1 = C_1 \textbf{ xor } A_1 \qquad C_1 = A_0 \cdot EN \qquad C_1 = \overline{A_0} \cdot EN$$
$$I_0 = C_0 \textbf{ xor } A_0 \qquad C_0 = 1 \cdot EN \qquad C_0 = 1 \cdot EN$$

toggle Ai | toggling condition

$$I_3 = G_3 \textbf{ xor } A_3 \qquad G_3 = \overline{S}\,A_2 A_1 A_0 + S\,\overline{A_2}\,\overline{A_1}\,\overline{A_0} \cdot EN$$
$$I_2 = G_2 \textbf{ xor } A_2 \qquad G_2 = \overline{S}\,A_1 A_0 + S\,\overline{A_1}\,\overline{A_0} \cdot EN$$
$$I_1 = G_1 \textbf{ xor } A_1 \qquad G_1 = \overline{S}\,A_0 + S\,\overline{A_0} \cdot EN$$
$$I_0 = G_0 \textbf{ xor } A_0 \qquad G_0 = 1 \cdot EN$$

S=0 Up Counting

S=1 Down Counting

References

[1] http://en.wikipedia.org/
[2] M. M. Mano, C. R. Kime, “Logic and Computer Design Fundamentals”, 4th ed.
[3] J. Stephenson, Understanding Metastability in FPGAs. Altera Corporation white paper. July 2009.