Rutgers University
School of Engineering

Fall 2022

332:231 – Digital Logic Design

Sophocles J. Orfanidis
ECE Department
orfanidi@rutgers.edu

Unit 6 – Sequential circuits, latches, flip-flops

Course Topics

1. Introduction to DLD, Verilog HDL, MATLAB/Simulink
2. Number systems
3. Analysis and synthesis of combinational circuits
4. Decoders/encoders, multiplexers/demultiplexers
5. Arithmetic systems, comparators, adders, multipliers
6. Sequential circuits, latches, flip-flops (Wakerly, Ch. 9 & 10)
7. Registers, shift registers, counters, LFSRs (Wakerly, Ch. 11)
8. Finite state machines, analysis and synthesis (Wakerly, Ch. 9)

Text: J. F. Wakerly, *Digital Design Principles and Practices*, 5/e, Pearson, 2018
additional references on Canvas Files > References

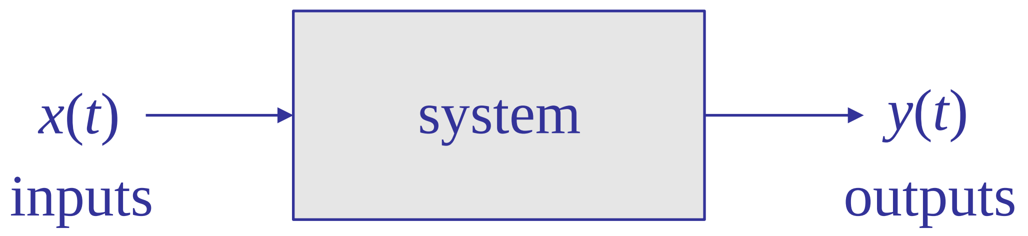
Sequential circuits (Wakerly, Ch. 9 & 10)

Topics discussed are:

- Finite state machines – Mealy/Moore
- State-holding elements – bistable elements
- SR latches / S′R′ latches
- D latches
- D flip-flops
- T flip-flops
- JK flip-flops
- Converting between flip-flop types

Unit-6 Contents:

sequential circuits and finite state-machines are special cases of dynamic systems



In general, a system can be defined by specifying the I/O computational rule that determines the output signal $y(t)$ from the input signal $x(t)$.

The time variable t can be continuous or discrete. The system can be linear or non-linear, time-invariant or time-varying, and can be described by differential or difference equations.

State Machines

State-space realizations, also known as state-space models, have a very large number of applications in many diverse fields, such as,

- digital logic design
- differential equations for physical systems
- system theory
- electric circuits
- linear systems
- digital signal processing
- control systems
- communication systems
- biomedical signal processing
- geophysical signal processing
- aerospace engineering
- military systems
- statistics and time series analysis
- predictive analytics
- econometrics and financial engineering

State Machines

State-space realizations are very powerful representations of systems (linear or nonlinear, time-invariant or not).

The system is described by a set of internal states at each time instant t, denoted for example by $Q(t)$, and these states are used to compute the current output $y(t)$ in terms of the current input $x(t)$, and then update the states to their next values, $Q(t+1)$, so that they can be used at time $t+1$ (or, more generally at, $t+\Delta t$).

In other words, the system's time evolution is described iteratively by a computational algorithm of the form,

```
for each time instant $t$, do:
    $y(t) = G\big(x(t), Q(t)\big)$          (compute output)
    $Q(t+1) = F\big(x(t), Q(t)\big)$        (update state)
```

[to get started, one needs to know the initial state, $Q(0)$, at $t=0$]

State Machines

For example, in going from time *t* to time *t*+2, one carries out the steps:

at time *t*,

$$y(t) = G\big(x(t), Q(t)\big)$$

$$Q(t+1) = F\big(x(t), Q(t)\big)$$

at time *t*+1,

$$y(t+1) = G\big(x(t+1), Q(t+1)\big)$$

$$Q(t+2) = F\big(x(t+1), Q(t+1)\big)$$

at time *t*+2,

$$y(t+2) = G\big(x(t+2), Q(t+2)\big)$$

$$Q(t+3) = F\big(x+2), Q(t+2)\big)$$

etc.

F( ) and *G*( ) depend on application

in DLD, *F* is referred to as
next-state logic, or excitation logic

and, *G* is referred to as output logic

from here on, we'll use the simplified notation,

$$x_t\,,\ y_t\,,\ Q_t$$

for

$$x(t),\ y(t),\ Q(t)$$

State Machines

It should be emphasized that the updated state Q_{t+1} is being computed at time t, and becomes available at time t, replacing Q_t , but it is saved until it is used later at time t+1.

The computations can be cast as a repetitive algorithm, in which the present state is overwritten by the next state.

initialize state Q (typically at t=0), then,

```
at each time $t$, do,
    $y_t = G(x_t, Q)$
    $Q = F(x_t, Q)$
```

or,

```
at each time $t$, do,
    $y_t = G(x_t, Q)$
    $Q_{\text{next}} = F(x_t, Q)$
    $Q = Q_{\text{next}}$
```

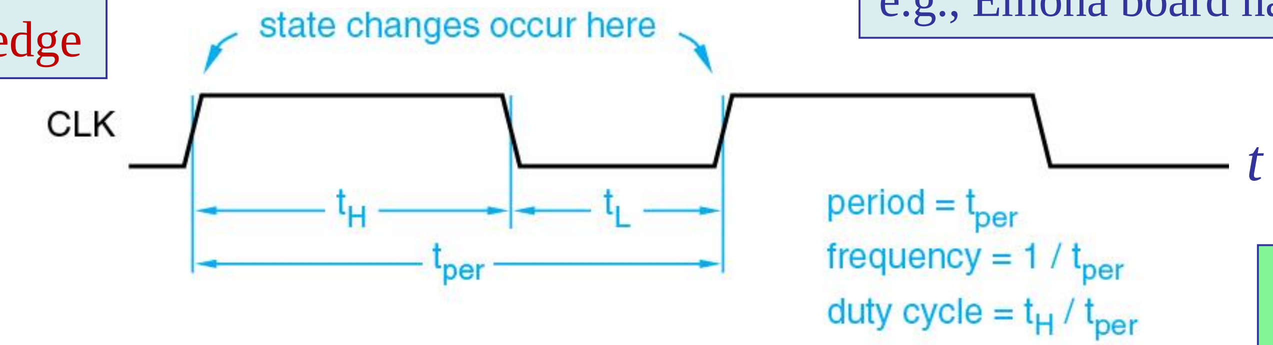
State Machines

DLD notation: we assume that time is discretized in units of 1, which means one clock period, so that $t+1$ means one clock period ahead of t.

In DLD (essentially all) sequential circuits are synchronously driven by a clock, and the state changes occur during the rising edge of the clock period (or, alternatively - but less commonly - during the falling edge.)

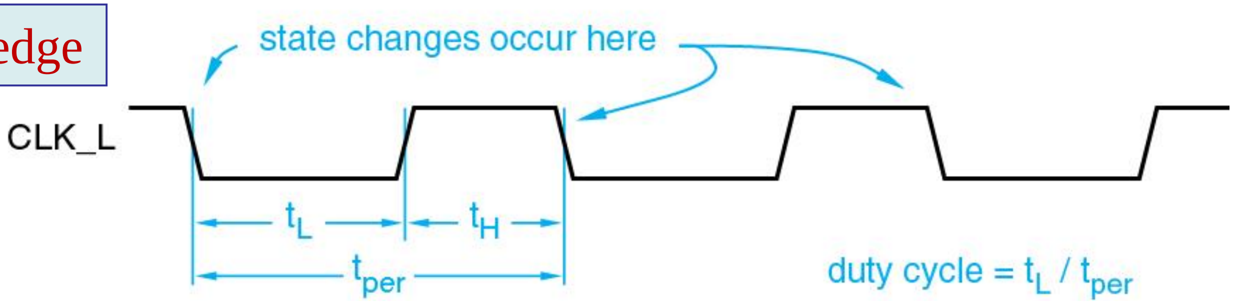
clock period depends on application, e.g., Emona board has 10 μsec period

rising edge



clock duty cycles are usually, 50%

falling edge



State Machines

State machines in DLD fall into two general families:

Moore type: output equation depends only on Q, i.e., $y = G(Q)$

Mealy type: output equation depends on both x and Q, i.e., $y = G(x, Q)$

for each time instant t, do:

$$y_t = G(x_t, Q_t)$$
$$Q_{t+1} = F(x_t, Q_t)$$

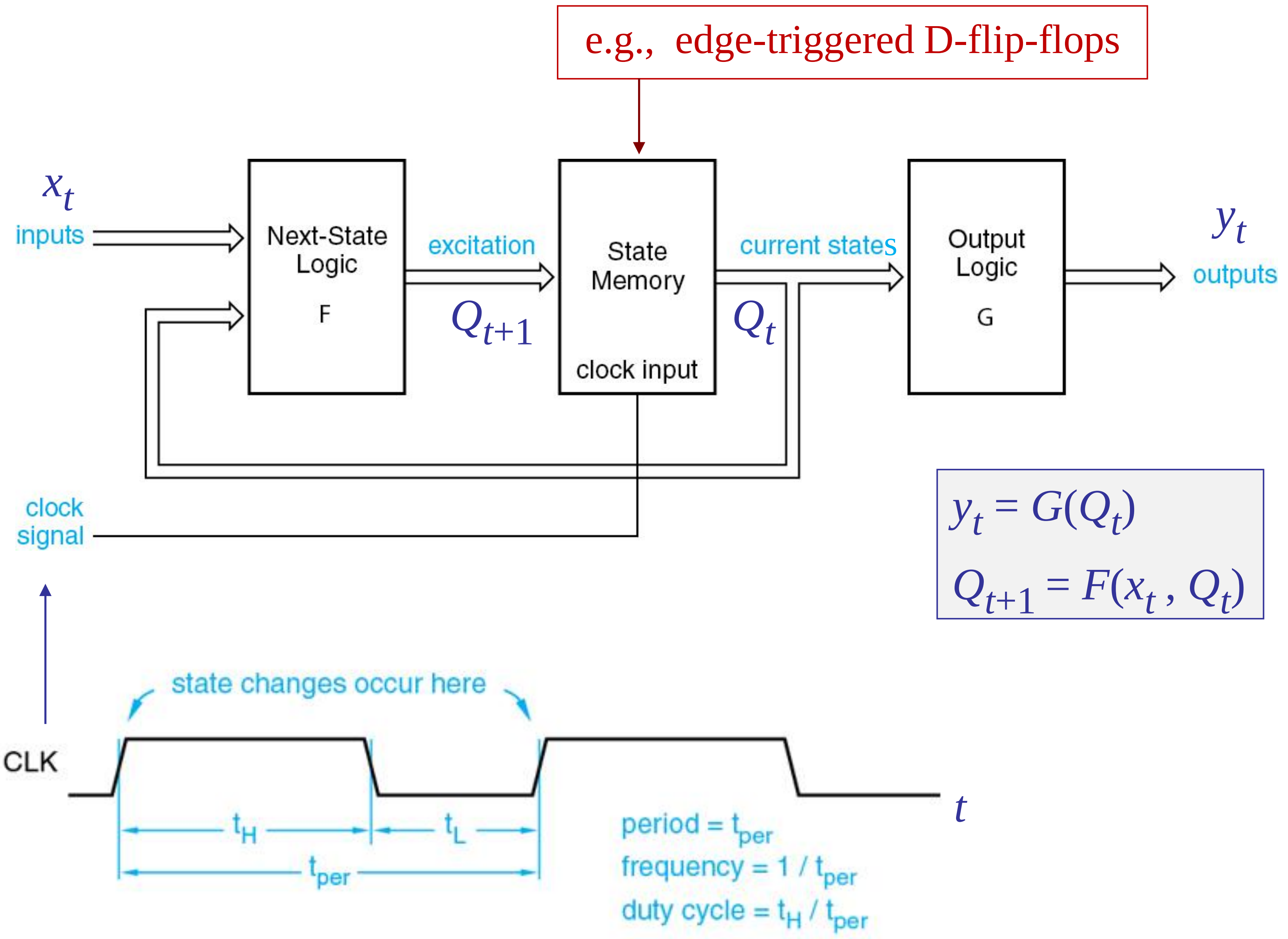
$\xrightarrow{\text{simplified notation}}$

$$y = G(x, Q)$$
$$Q_{\text{next}} = F(x, Q)$$

changes occur at clock rising edges

in general, one can have multiple inputs, multiple outputs (MIMO systems), and multiple states

State Machines – Moore

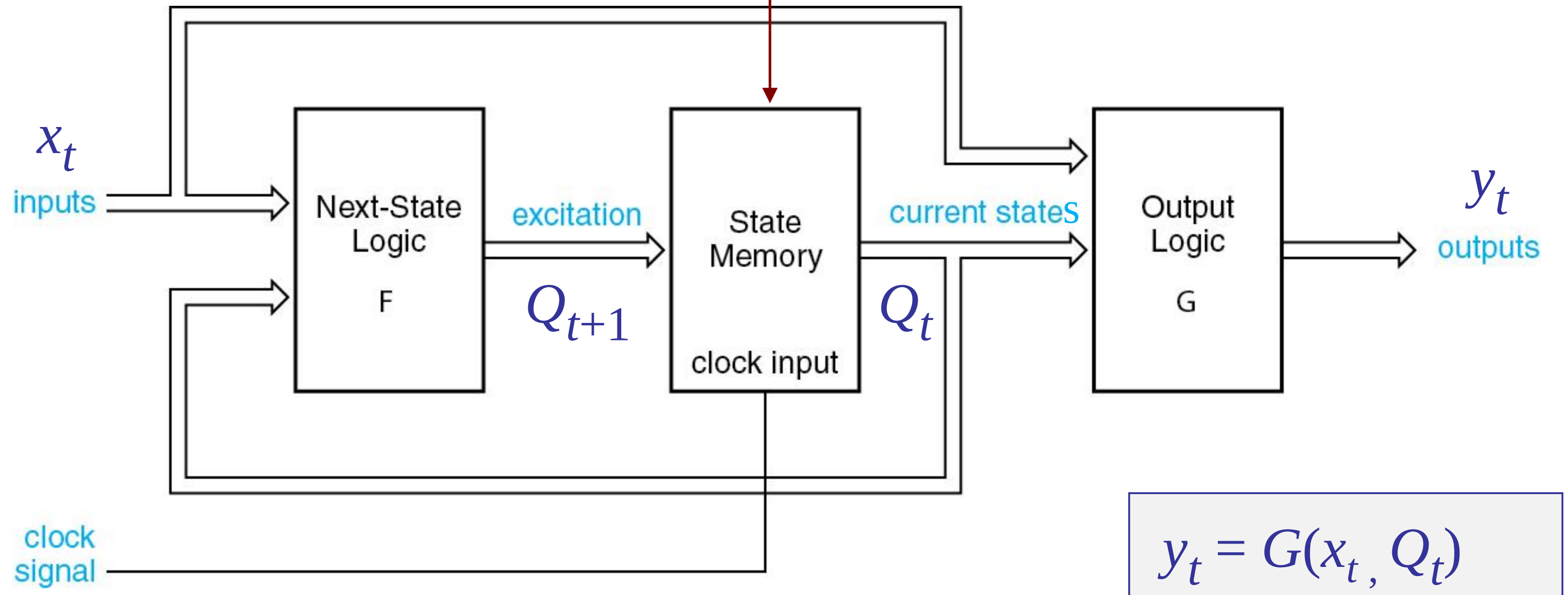


$$y_t = G(Q_t)$$

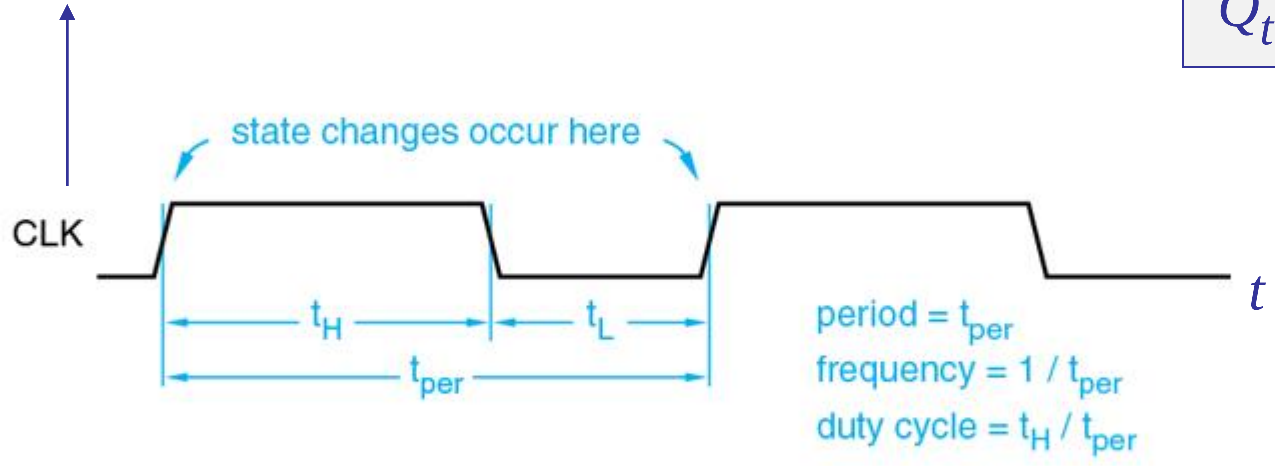
$$Q_{t+1} = F(x_t\,,\, Q_t)$$

State Machines – Mealy

e.g., edge-triggered D-flip-flops



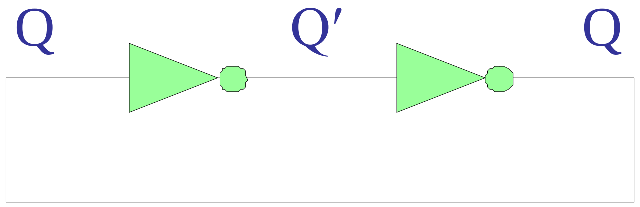
$$y_t = G(x_t, Q_t)$$
$$Q_{t+1} = F(x_t, Q_t)$$



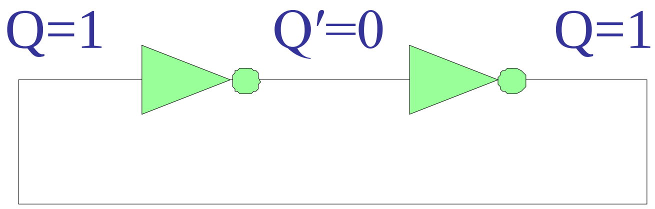
period = t_{per}
frequency = $1 / t_{per}$
duty cycle = t_H / t_{per}

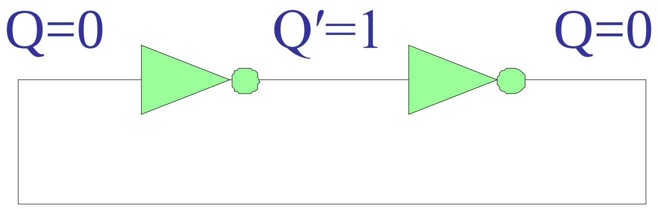
state-holding elements
bistable elements
latches

what are state-holding elements?
how to load them?

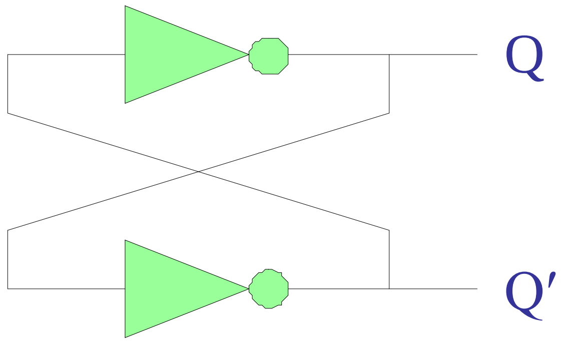


two stable states
Q=1 and Q=0

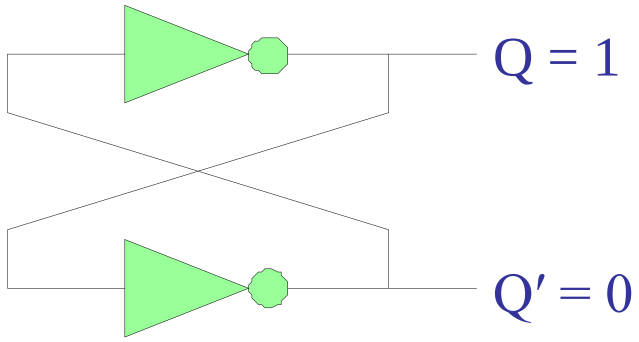


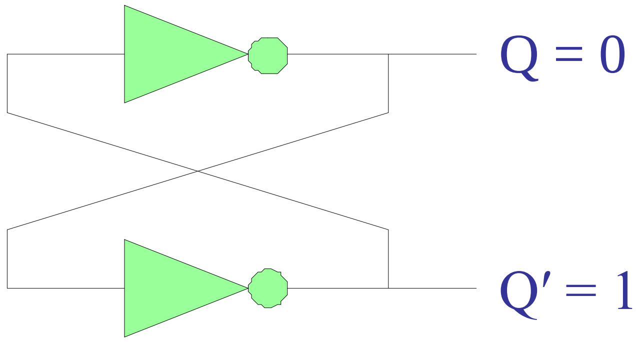


state-holding elements
bistable elements
latches



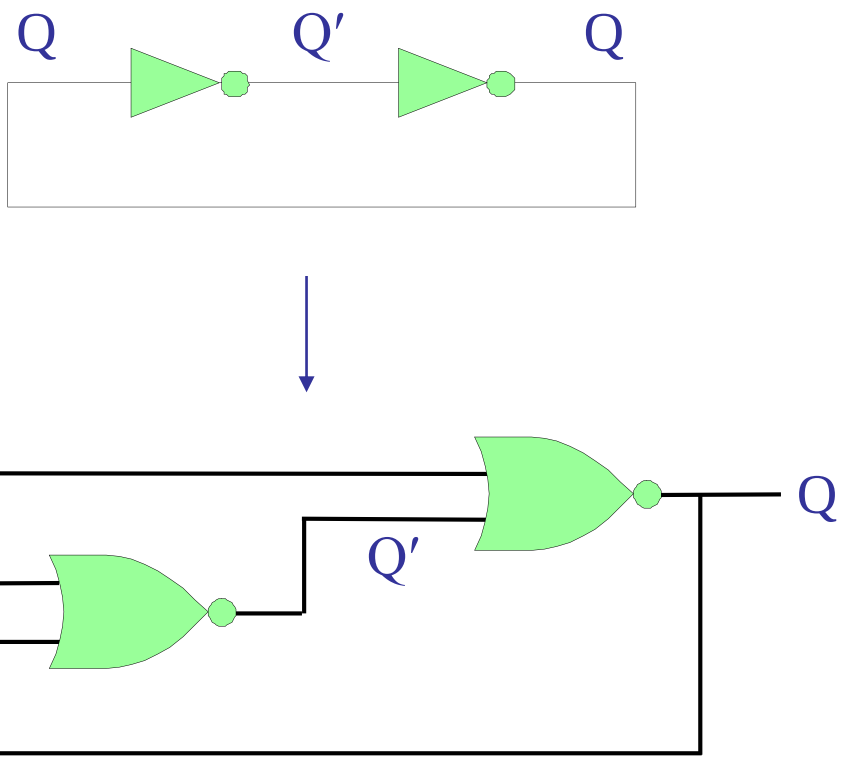
two stable states - but how to load them?



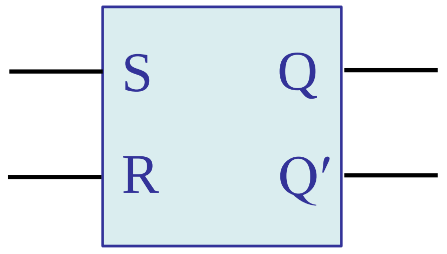


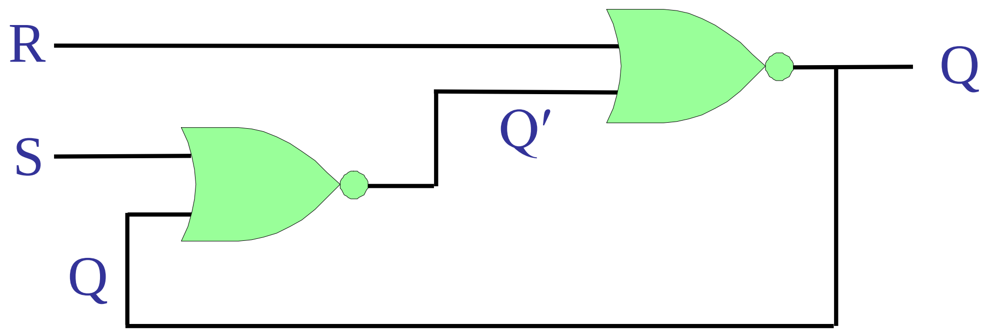
state-holding elements
bistable elements
latches

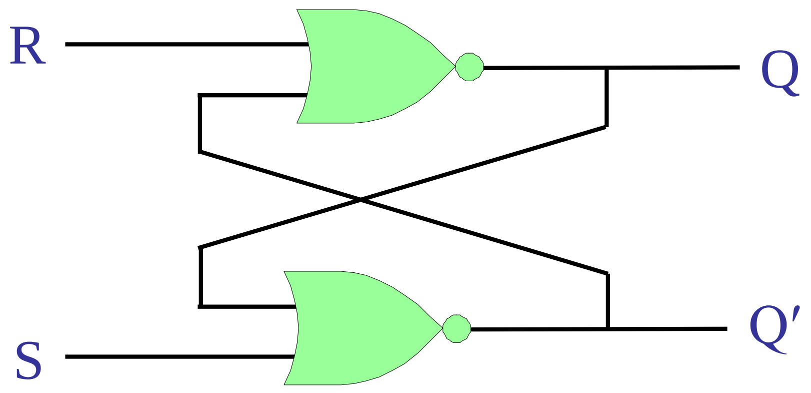
the state-loading problem is solved with SR latches, which provide external inputs to the bistable elements.



state-holding elements
bistable elements
latches



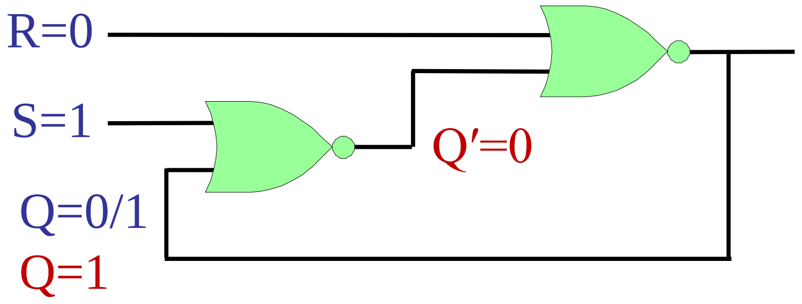




S R-latch

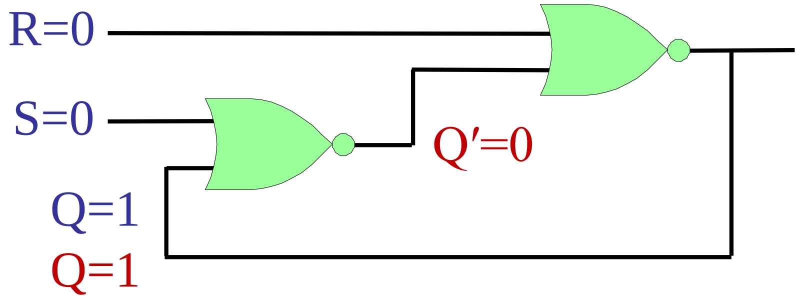
NOR version

normal operation



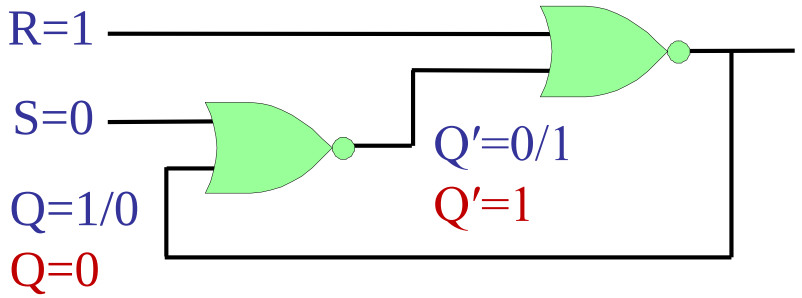
Q=0/1 before
Q=1 next

set, Q=1 state



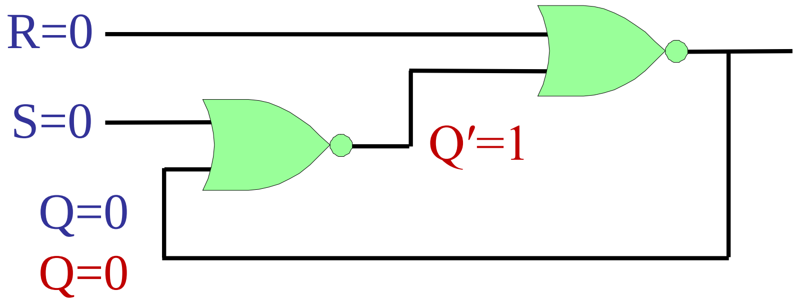
Q=1 before
Q=1 next

hold, Q=1 state



Q=1/0 before
Q=0 next

reset, Q=0 state

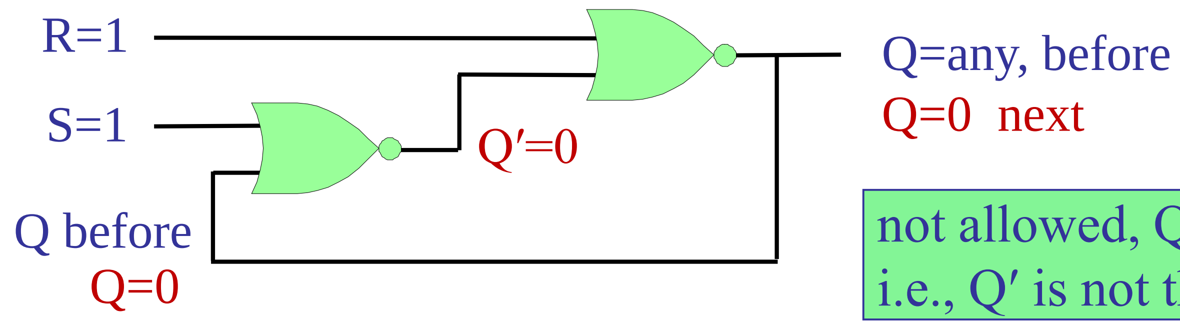


Q=0 before
Q=0 next

hold, Q=0 state

	S	R	Q	Q_{next}
hold	0	0	Q	Q
reset	0	1	Q	0
set	1	0	Q	1
	1	1	Q	?

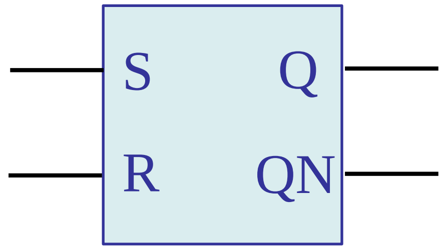
X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



abnormal operation

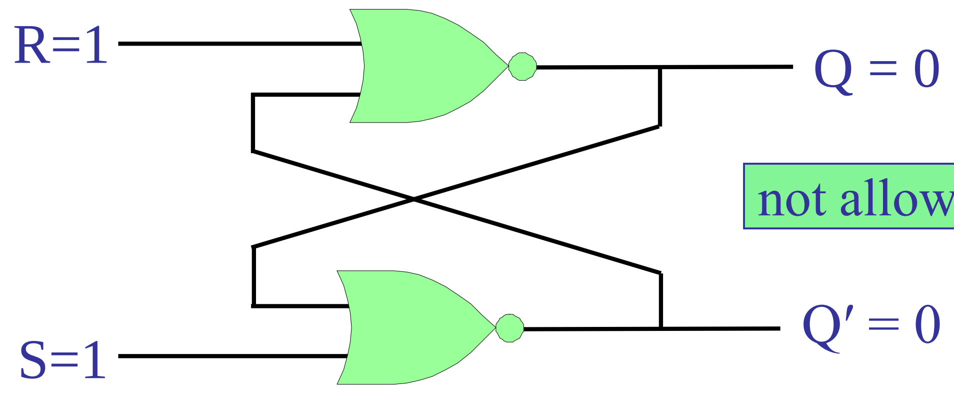
not allowed, Q=0 and Q′ = 0,
i.e., Q′ is not the inverse of Q

	S R Q	Q_{next}
hold	0 0 Q	Q
reset	0 1 Q	0
set	1 0 Q	1
	1 1 Q	?



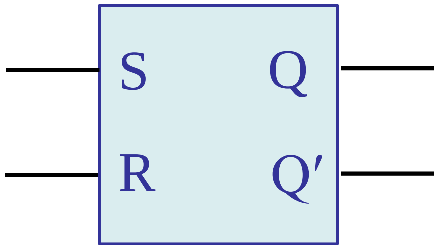
R=S=1 are not allowed because if R,S are de-asserted at exactly the same time, that is, R=S=0, then, the latch will enter into a metastable, race condition, with the Q, Q′ outputs oscillating between the values 0,0 and 1,1, as explained below.

$$0\,0 \to 1\,1 \to 0\,0 \to 1\,1 \ldots$$

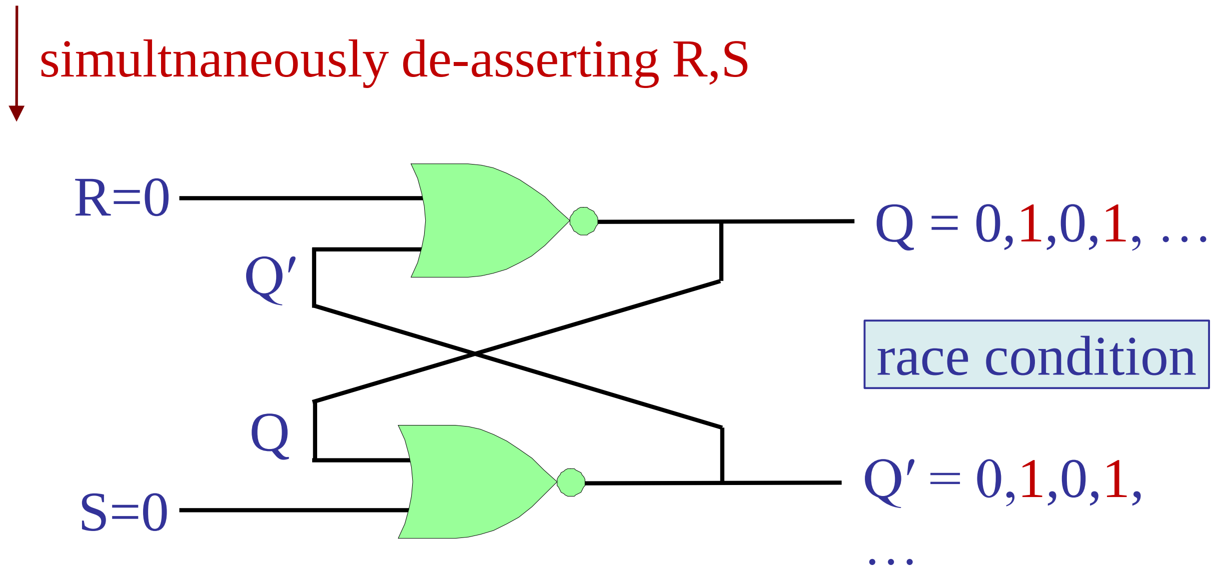


not allowed, Q=0 and Q′ = 0

abnormal operation



simultnaneously de-asserting R,S

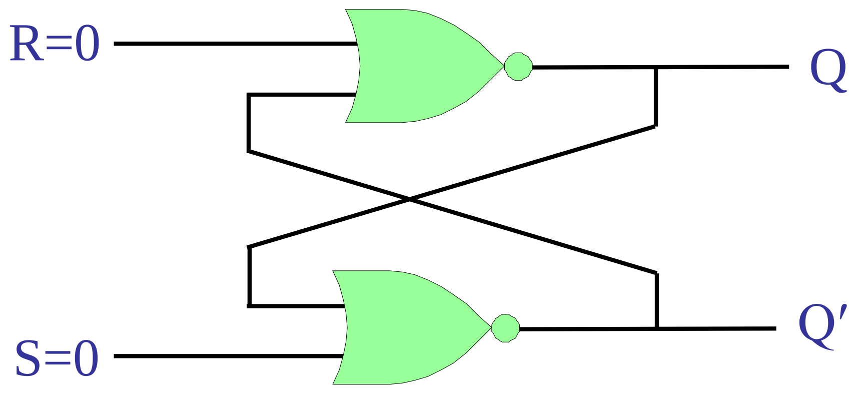


race condition

R=S=1 are not allowed because if R,S are de-asserted at exactly the same time, that is, R=S=0, then, the latch will enter into a metastable, race condition, with the Q, Q′ outputs oscillating between the values 0,0 and 1,1, as seen above.

$0\,0 \rightarrow 1\,1 \rightarrow 0\,0 \rightarrow 1\,1 \ldots$

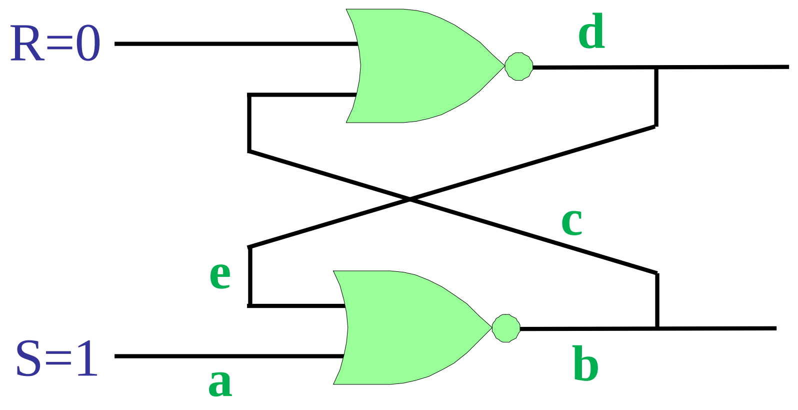
normal operation



inactive stable state
hold Q and Q′

sequence of events

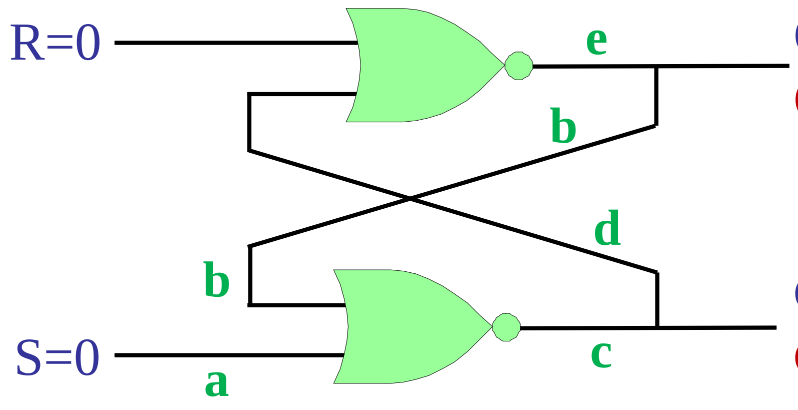
$a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$



Q before
Q=1 next

set, Q=1 state

Q′ before
Q′ =0 next

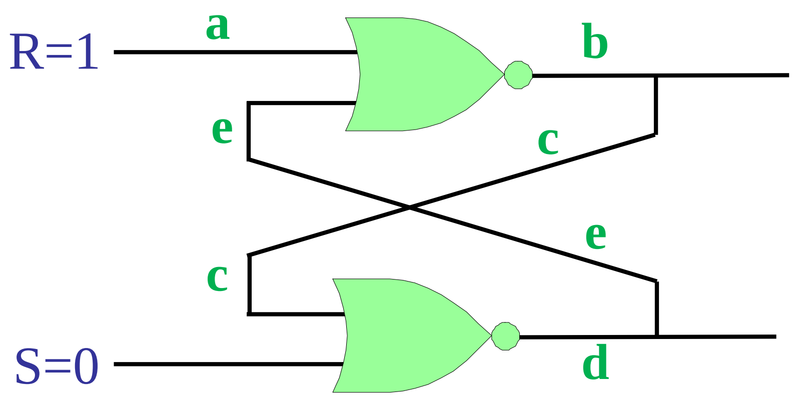


Q=1 before
Q=1 next

hold, Q=1 state

Q′ =0 before
Q′ =0 next

normal operation



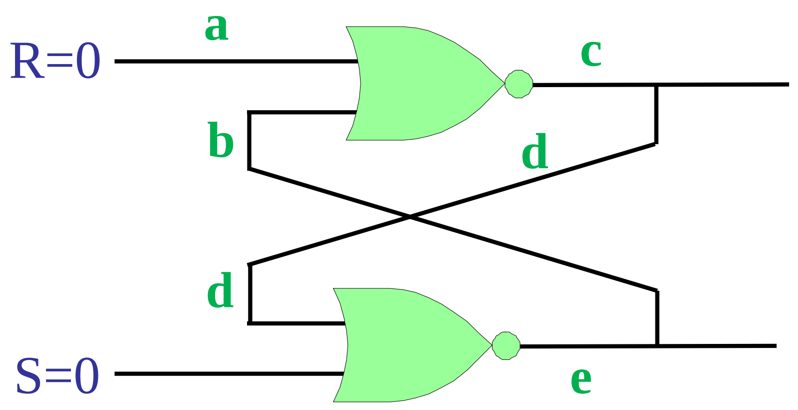
Q=1 before
Q=0 next

reset, Q=0 state

Q′ =0 before
Q′ =1 next

sequence of events

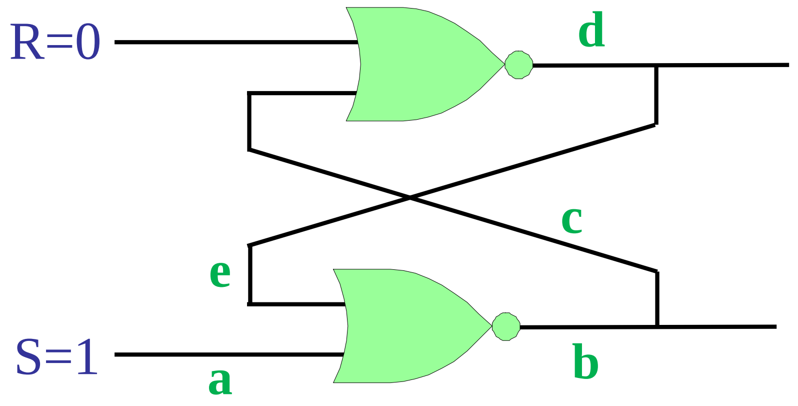
a → b → c → d → e



Q=0 before
Q=0 next

hold, Q=0 state

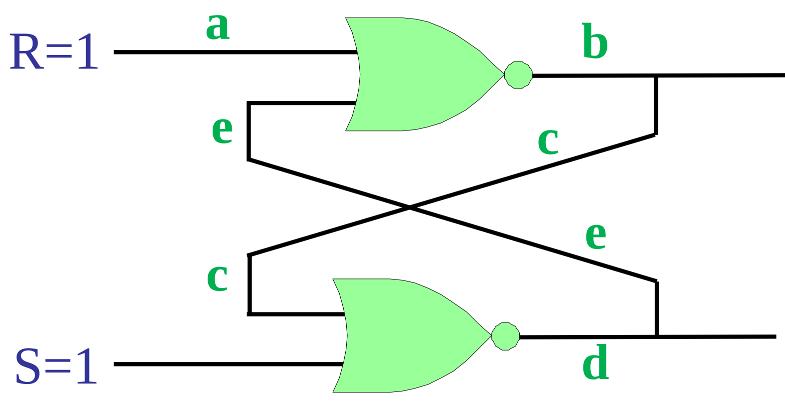
Q′ =1 before
Q′ =1 next



Q=0 before
Q=1 next

set, Q=1 state

Q′=1 before
Q′ =0 next



Q=1 before
Q=0 next

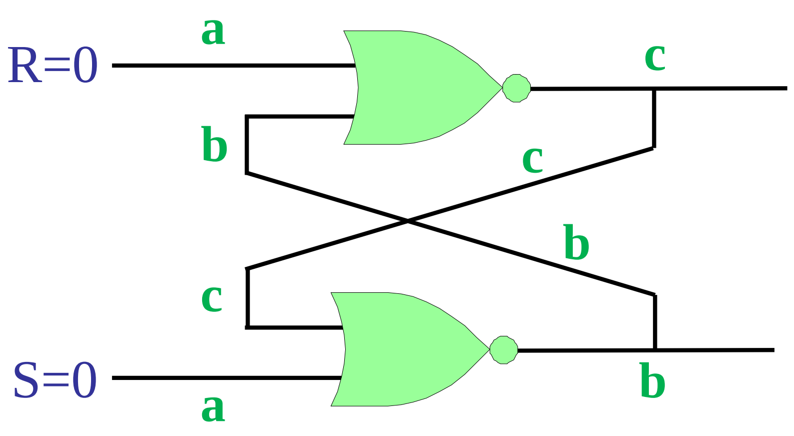
not allowed, Q=0 and Q′ = 0

abnormal operation

Q′ =0 before
Q′ =0 next

sequence of events
a → b → c → d → e

simultnaneously de-asserting R,S



Q=0 before
Q=1 next
Q=0 next, etc

sequence of events
a → b → c

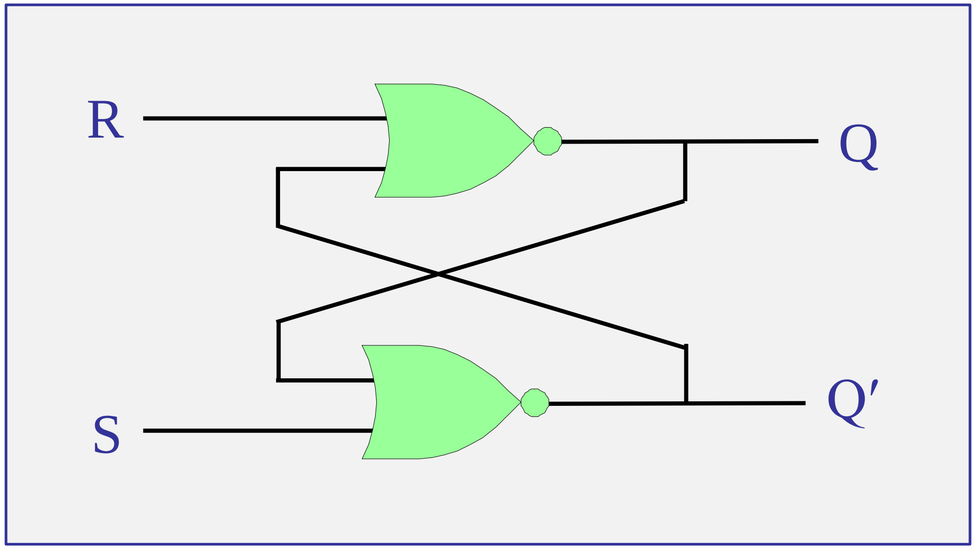
Q′ =0 before
Q′ =1 next
Q′ =0 next, etc

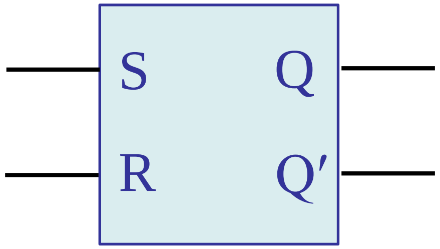
latch enters into a metastable, race condition, with the Q, Q′ outputs oscillating between the values 0,0 and 1,1

0 0 → 1 1 → 0 0 → 1 1 . . .

SR latch

NOR implementation





	S	R	Q	Q_{next}	Q'_{next}
hold	0	0	Q	Q	Q′
reset	0	1	Q	0	1
set	1	0	Q	1	0
not allowed	1	1	Q	0	0

characteristic table

characteristic equation

$$Q_{next} = R' S + R' Q$$

but with these as “don’t cares”

$$Q_{next} = S + R' Q$$

SR latch

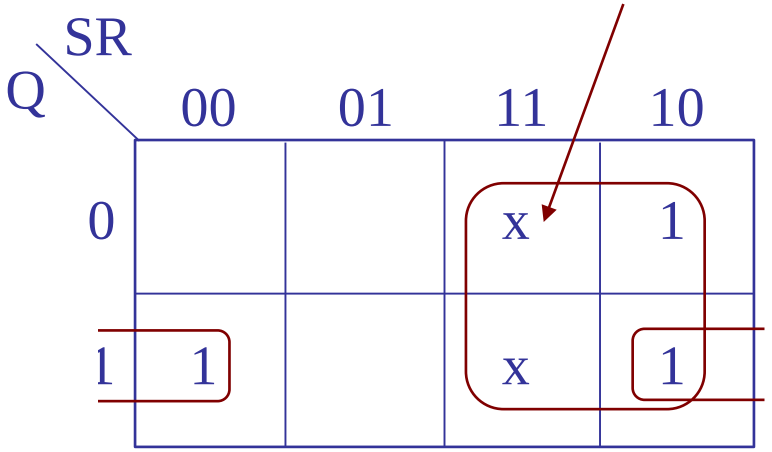
characteristic table

	S	R	Q	Q_{next}	Q'_{next}
hold	0	0	Q	Q	Q′
reset	0	1	Q	0	1
set	1	0	Q	1	0
not allowed	1	1	Q	0	0

characteristic equation

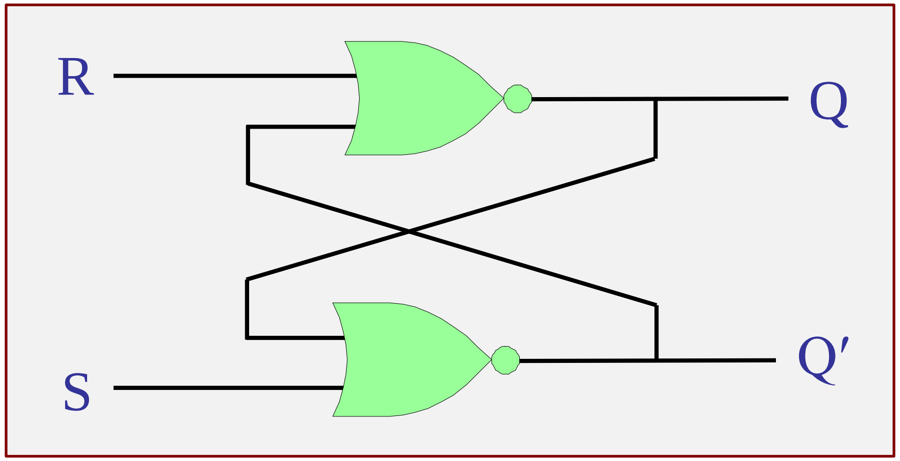
$$\boxed{Q_{next} = R'\,S + R'\,Q}$$

if treated as "don't cares"

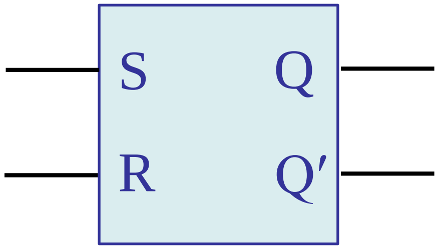


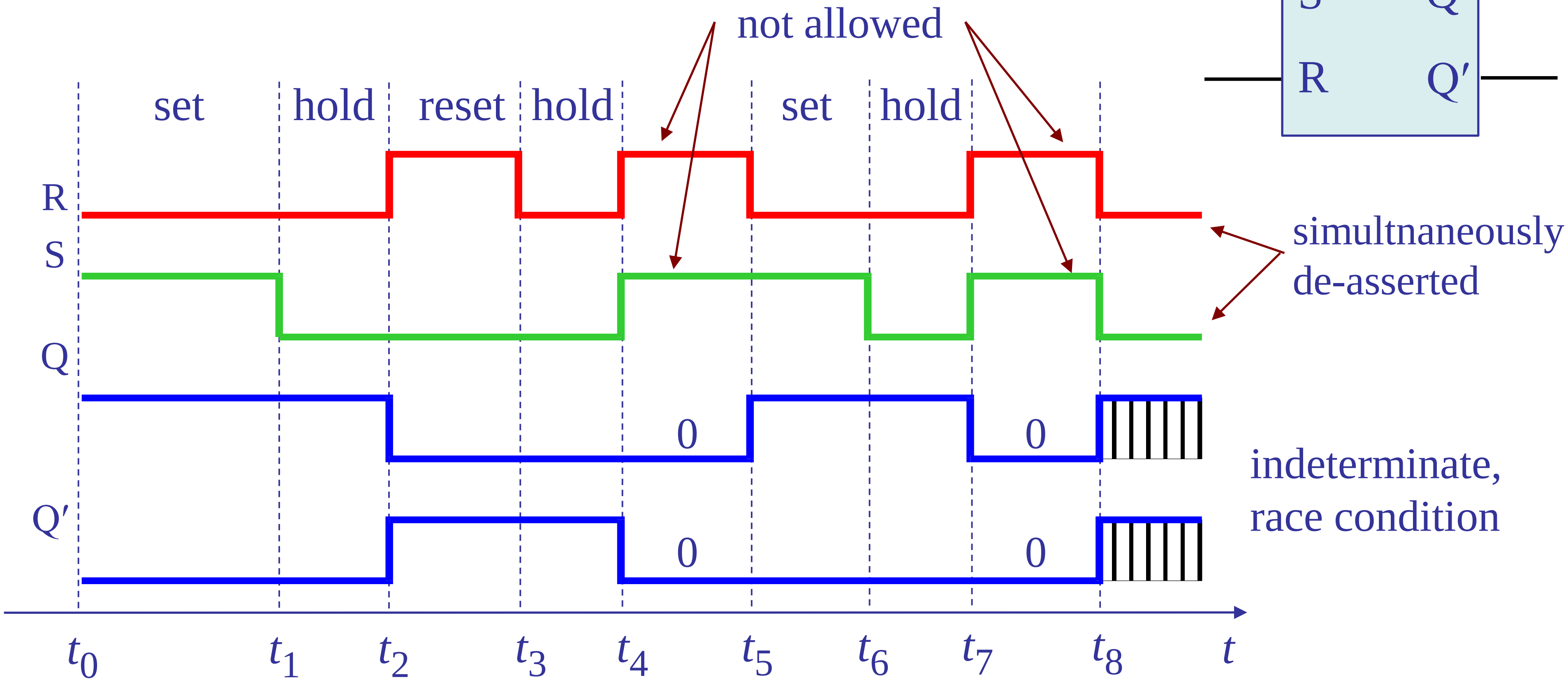
$$\Longrightarrow \boxed{Q_{next} = S + R'\,Q}$$

SR latch



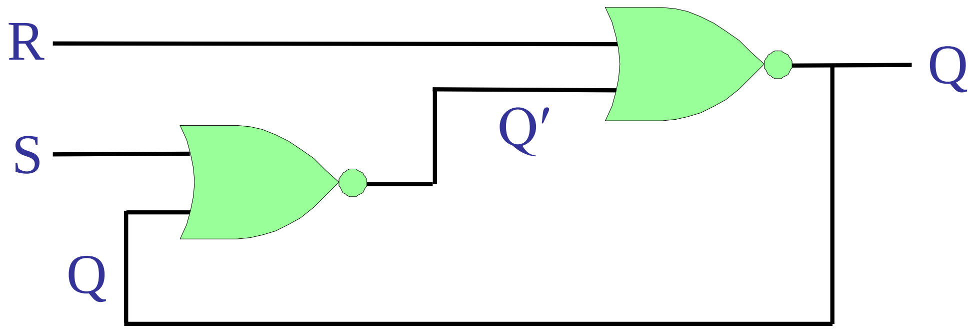
S	R	Q	Q_{next}	Q'_{next}
0	0	Q	Q	Q′
0	1	Q	0	1
1	0	Q	1	0
1	1	Q	0	0





SR latches – NOR and NAND realizations

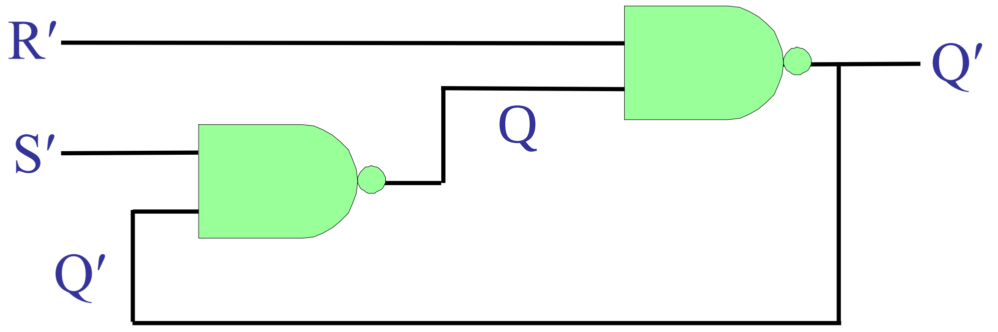
NOR



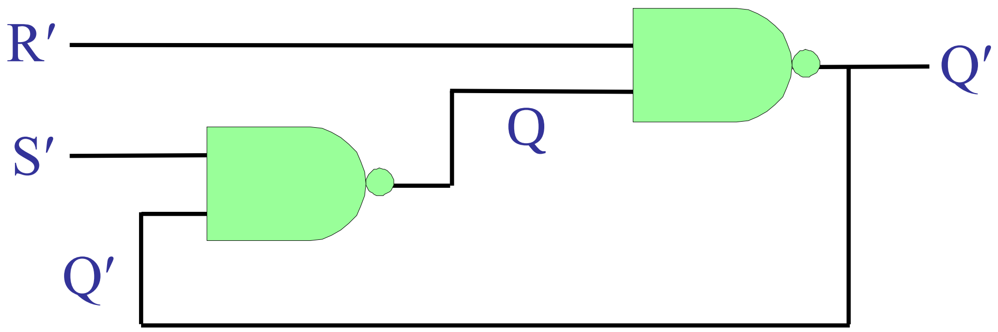
using De Morgan duality theorem:

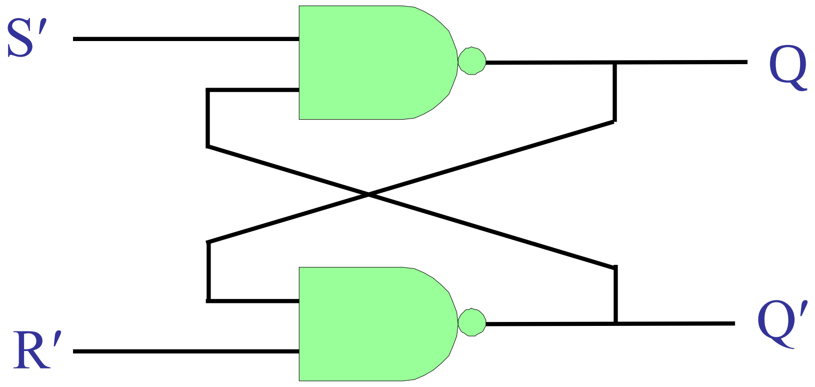
$F(X,Y,Z,\ldots)' = F_{dual}(X',Y',Z',\ldots)$

NAND



state-holding elements
bistable elements
latches



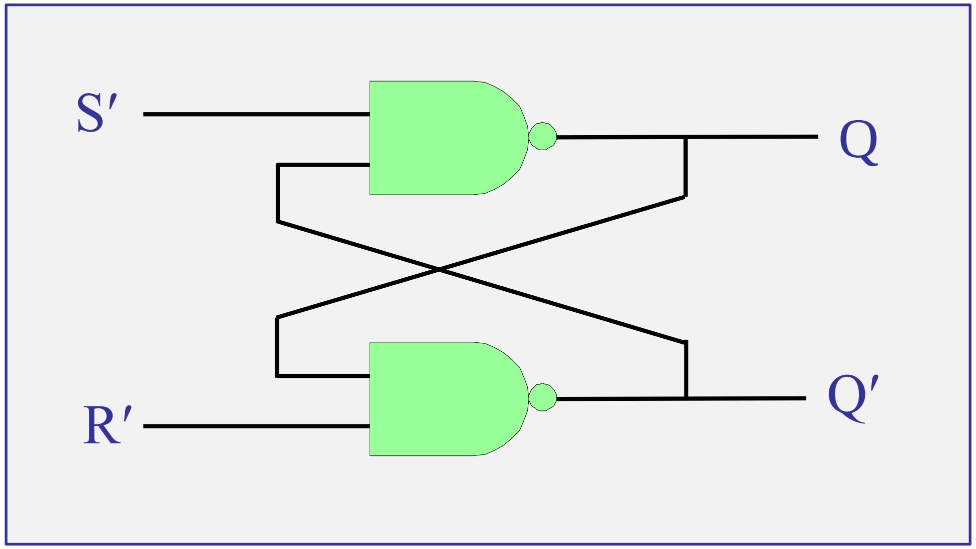


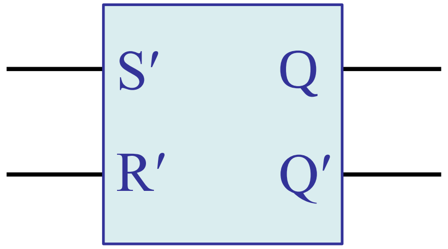
$\overline{S}\,\overline{R}$ latch
or
S′R′ latch

$\overline{S}\,\overline{R}$ latch

NAND implementation

active low $\longrightarrow$





	S′	R′	Q	Q_{next}	Q'_{next}
not allowed	0	0	Q	1	1
active low $\longrightarrow$ set	0	1	Q	1	0
active low $\longrightarrow$ reset	1	0	Q	0	1
hold	1	1	Q	Q	Q′

characteristic table

characteristic equation

$$Q_{next} = S + R'\,Q$$

$$Q_{next} = (\,S'\,(R'\,Q)'\,)'$$

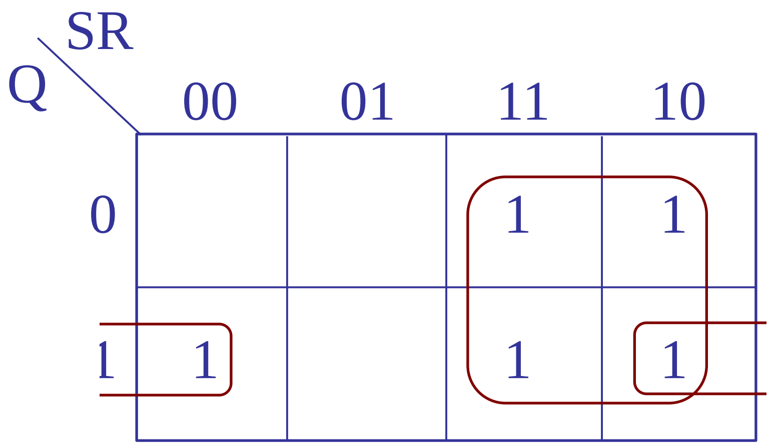
$\overline{S}\,\overline{R}$ latch

characteristic table

	S'	R'	Q	Q_{next}	Q'_{next}
not allowed	0	0	Q	1	1
set	0	1	Q	1	0
reset	1	0	Q	0	1
hold	1	1	Q	Q	Q′

characteristic equation

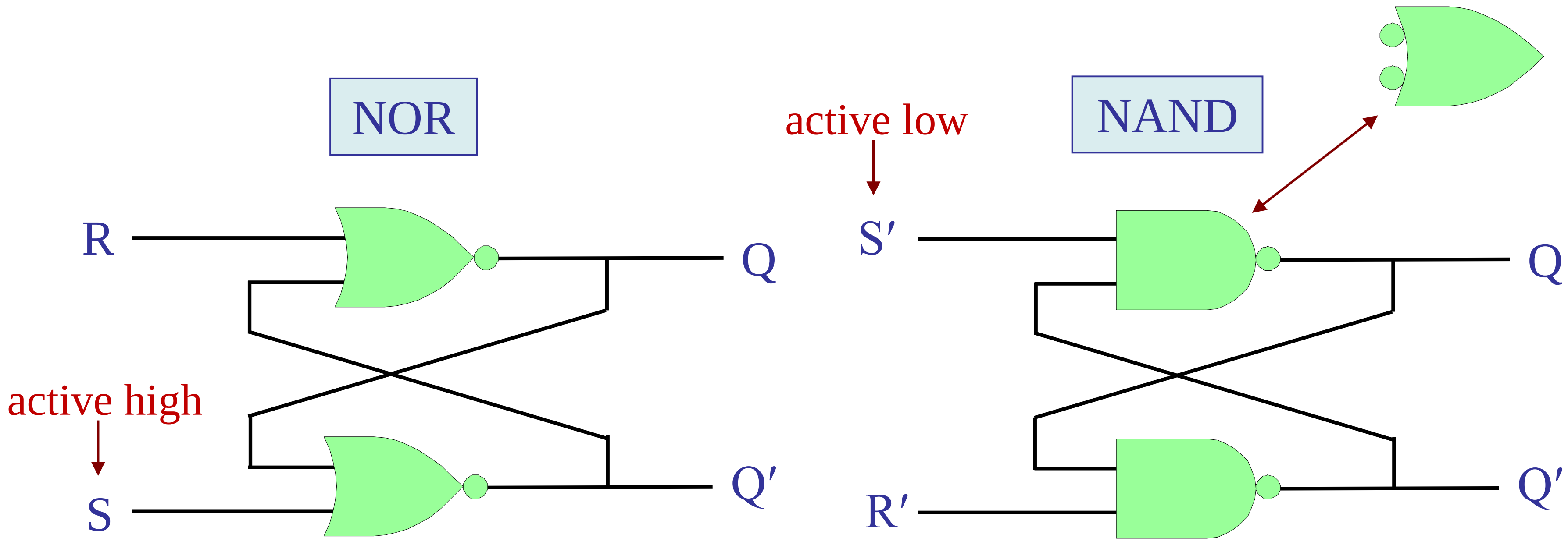
$$\boxed{Q_{next} = S + R'\,Q}$$



$\Longrightarrow$ $\boxed{Q_{next} = S + R'\,Q}$

same K-map as in the NOR case

SR latches - Summary

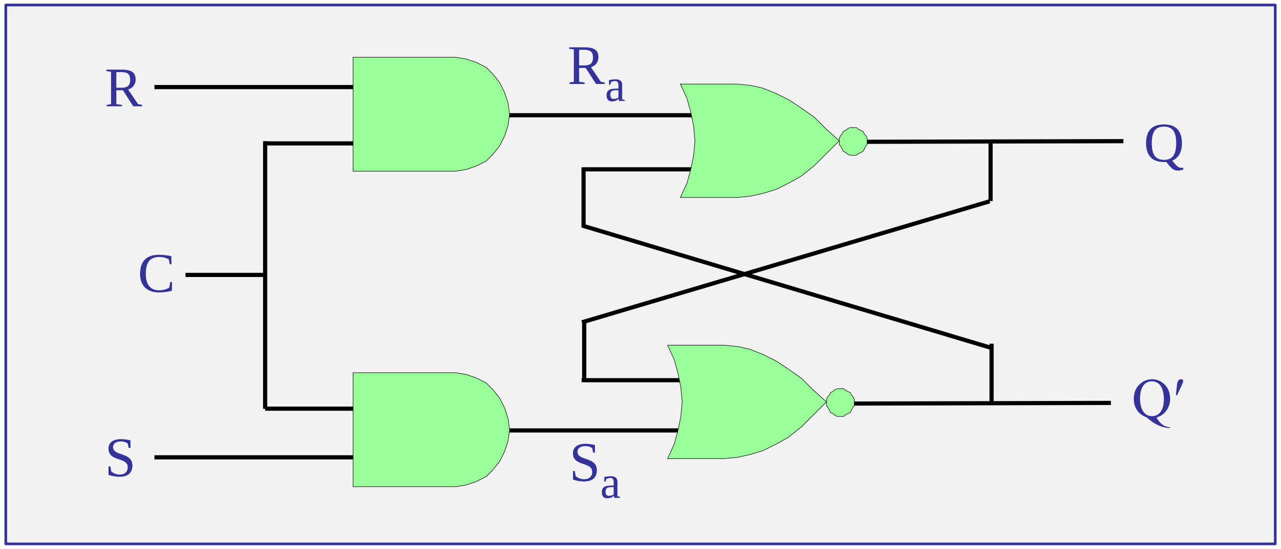


	S	R	Q	Q_{next}	Q'_{next}
hold	0	0	Q	Q	Q′
reset	0	1	Q	0	1
set	1	0	Q	1	0
xx	1	1	Q	0	0

S′	R′	Q	Q_{next}	Q'_{next}	
0	0	Q	1	1	xx
0	1	Q	1	0	set
1	0	Q	0	1	reset
1	1	Q	Q	Q′	hold

characteristic tables – in both cases, set means Q=1, reset, Q=0

SR latch – with enable/control/clock signal – NOR version



	S R C	Q_{next}	Q'_{next}
hold	x x 0	Q	Q′
hold	0 0 1	Q	Q′
reset	0 1 1	0	1
set	1 0 1	1	0
not allowed	1 1 1	0	0

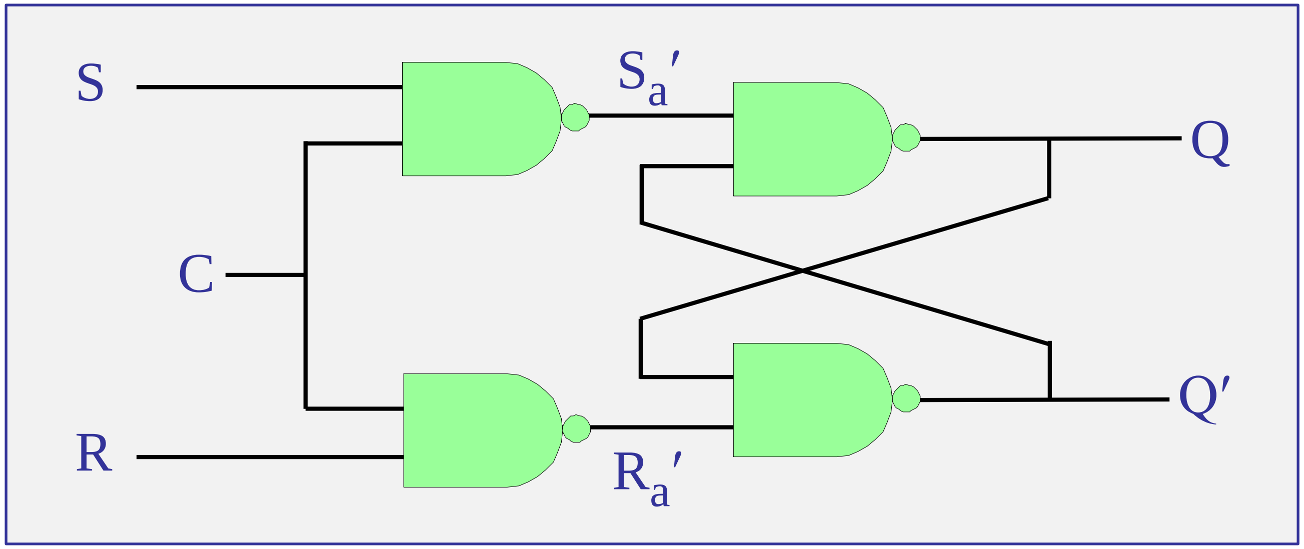
characteristic table

characteristic equation

$$Q_{next} = C\,R'\,S + C'\,Q + R'\,Q$$

$$R_a = C\,R$$
$$S_a = C\,S$$
$$Q_{next} = R_a'\,S_a + R_a'\,Q$$

SR latch – with enable/control/clock signal – NAND version



	S	R	C	Q_{next}	Q'_{next}
hold	x	x	0	Q	Q′
hold	0	0	1	Q	Q′
reset	0	1	1	0	1
set	1	0	1	1	0
not allowed	1	1	1	1	1

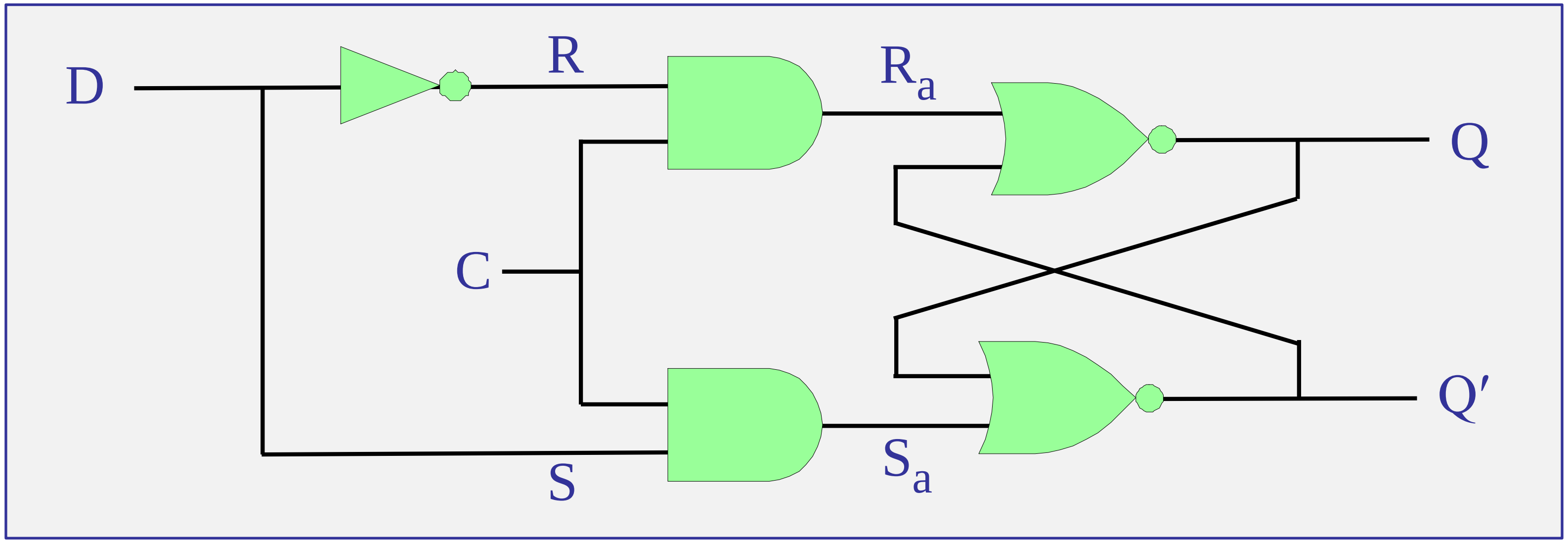
characteristic table

characteristic equation

$$Q_{next} = C\,S + (C' + R')Q$$

D latch

NOR implementation



characteristic table

	D	C	Q	Q_{next}
hold Q	x	0	Q	Q
follow D	0	1	Q	0
follow D	1	1	Q	1

characteristic equation

$$Q_{next} = C\,D + C'\,Q$$

$$R_a = C\,D'$$

$$S_a = C\,D$$

$$Q_{next} = R_a'\,S_a + R_a'\,Q$$

if C=1,
$Q_{next} = D$

if C=0,
$Q_{next} = Q$

i.e., Q follows D while C=1, otherwise Q remains unchanged

D latch

$R_a = C\,D'$

$S_a = C\,D$

$Q_{next} = R_a'\,S_a + R_a'\,Q$

$= (C\,D')'\,(CD) + (C\,D')'\,Q$

$= (C' + D)\,(CD) + (C' + D)\,Q$

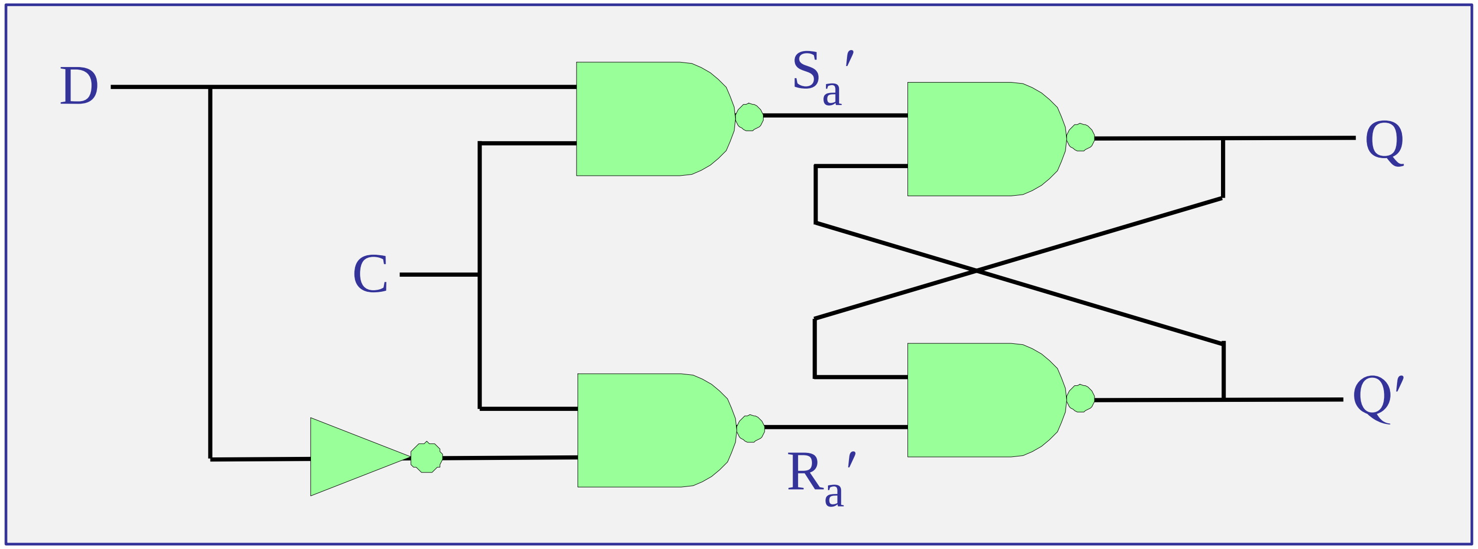
$= C'\,CD + D\,C\,D + C'\,Q + D\,Q$

$= C\,D + C'\,Q + D\,Q$

$= C\,D + C'\,Q$ (by consensus theorem)

D latch

NAND implementation



characteristic table

	D	C	Q	Q_{next}
hold Q	x	0	Q	Q
follow D	0	1	Q	0
follow D	1	1	Q	1

while C=1, Q follows D
while C=0, Q is unchanged

characteristic equation

$$Q_{next} = C\,D + C'\,Q$$

$$\uparrow$$

$R_a = C\,D'$

$S_a = C\,D$

$Q_{next} = S_a + R_a'\,Q$

$\uparrow$

see p.29

if C=1,
$Q_{next} = D$

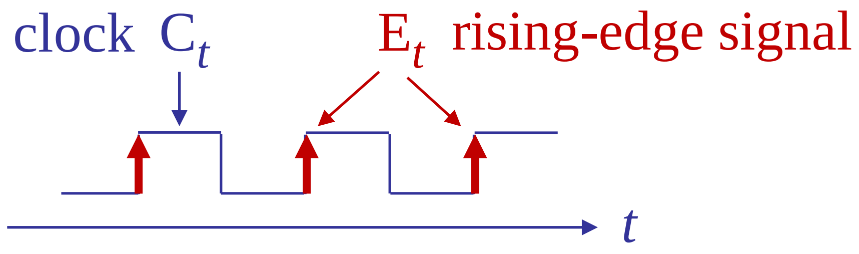
if C=0,
$Q_{next} = Q$

turning a D latch into a D flip-flop $\longrightarrow$

latches
vs.
flip-flops

D-latches are level-sensitive storage elements

D-flip-flops are edge-triggered storage elements



	D	C	Q_{next}
hold Q	x	0	Q
follow D	0	1	0
follow D	1	1	1

D-latch
characteristic equation
level-sensitive

$$\boxed{Q_{next} = C\,D + C'\,Q}$$

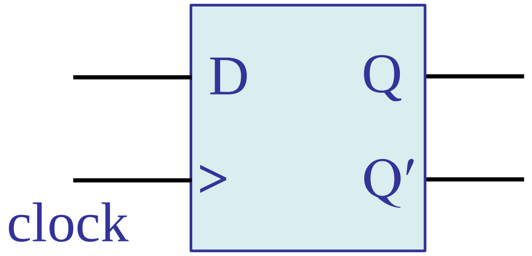
	D	E	Q_{next}
hold Q	x	$\not\uparrow$	Q
follow D	0	$\uparrow$	0
follow D	1	$\uparrow$	1

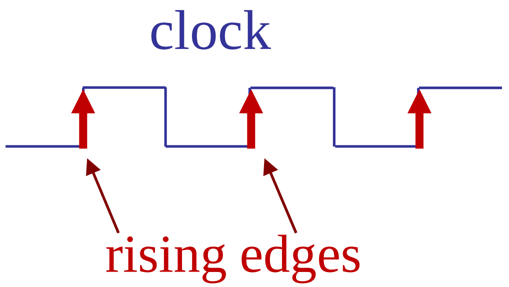
D-flip-flop
characteristic equation
edge-triggered

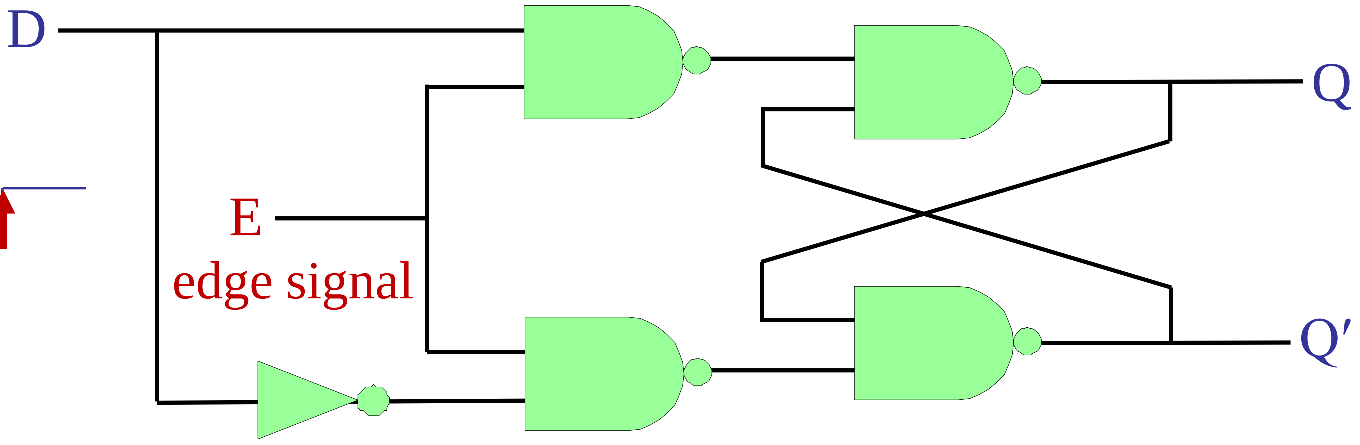
$$\boxed{Q_{next} = E\,D + E'\,Q}$$

D flip-flop

positive-edge-triggered D flip-flop







characteristic table

	D_t	E_t	Q_{t+1}
hold Q	x	$\not\uparrow$	Q_t
follow D	0	$\uparrow$	0
follow D	1	$\uparrow$	1

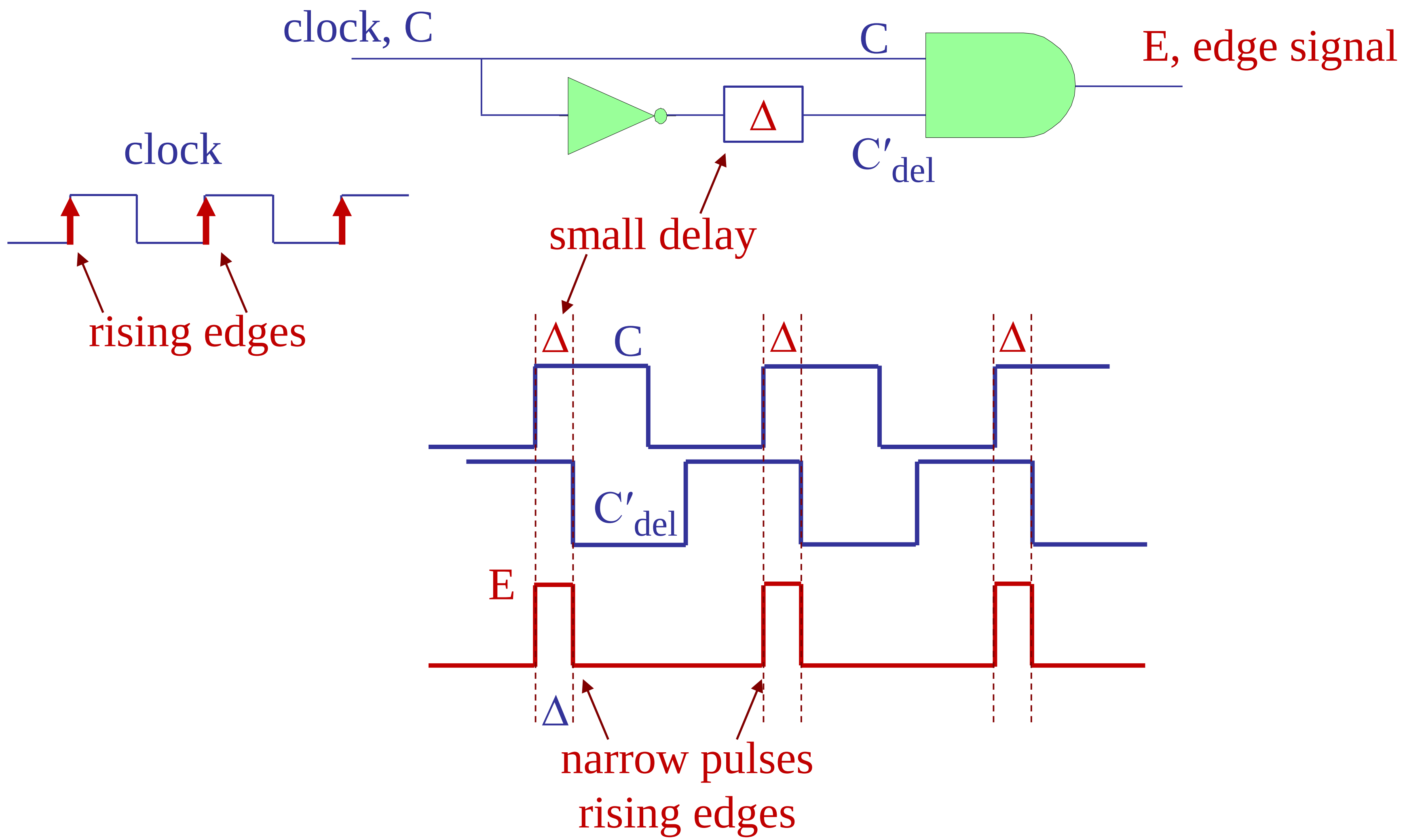
characteristic equation

$$Q_{t+1} = E_t\, D_t + E_t'\, Q_t = \begin{cases} D_t\,, & \text{if } t \text{ at edge} \\ Q_t\,, & \text{if } t \text{ not at edge} \end{cases}$$

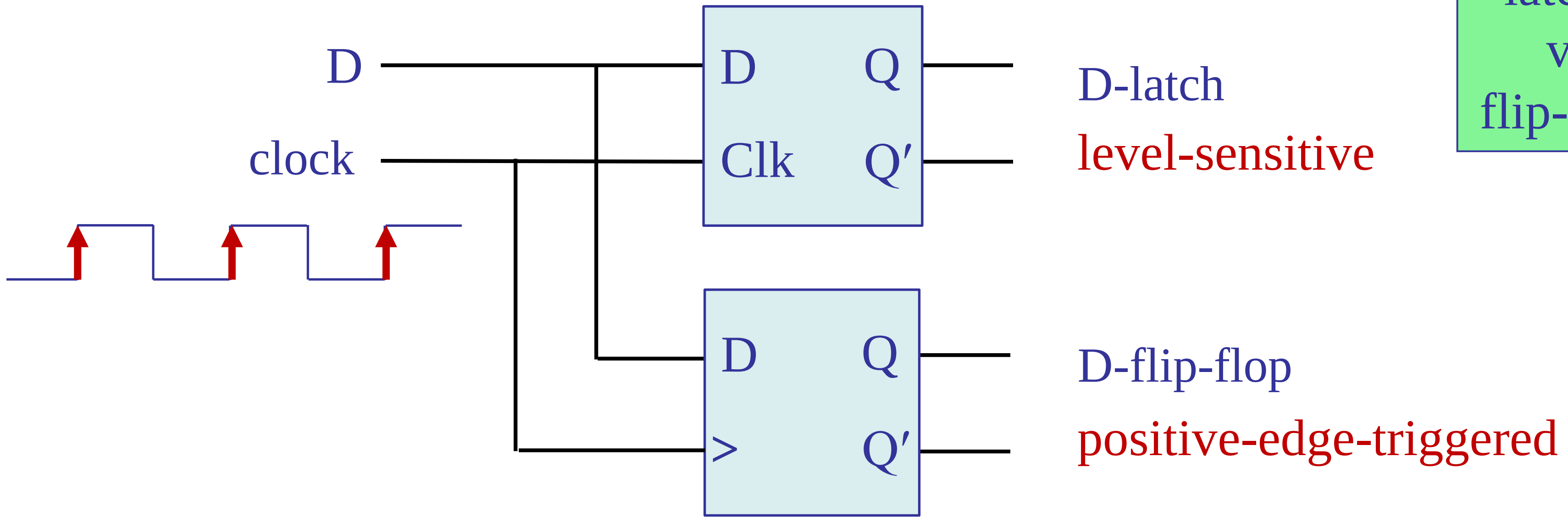
i.e., D flip-flop copies *D* to *Q* on the rising edge of the clock, and remembers *Q* at all other times

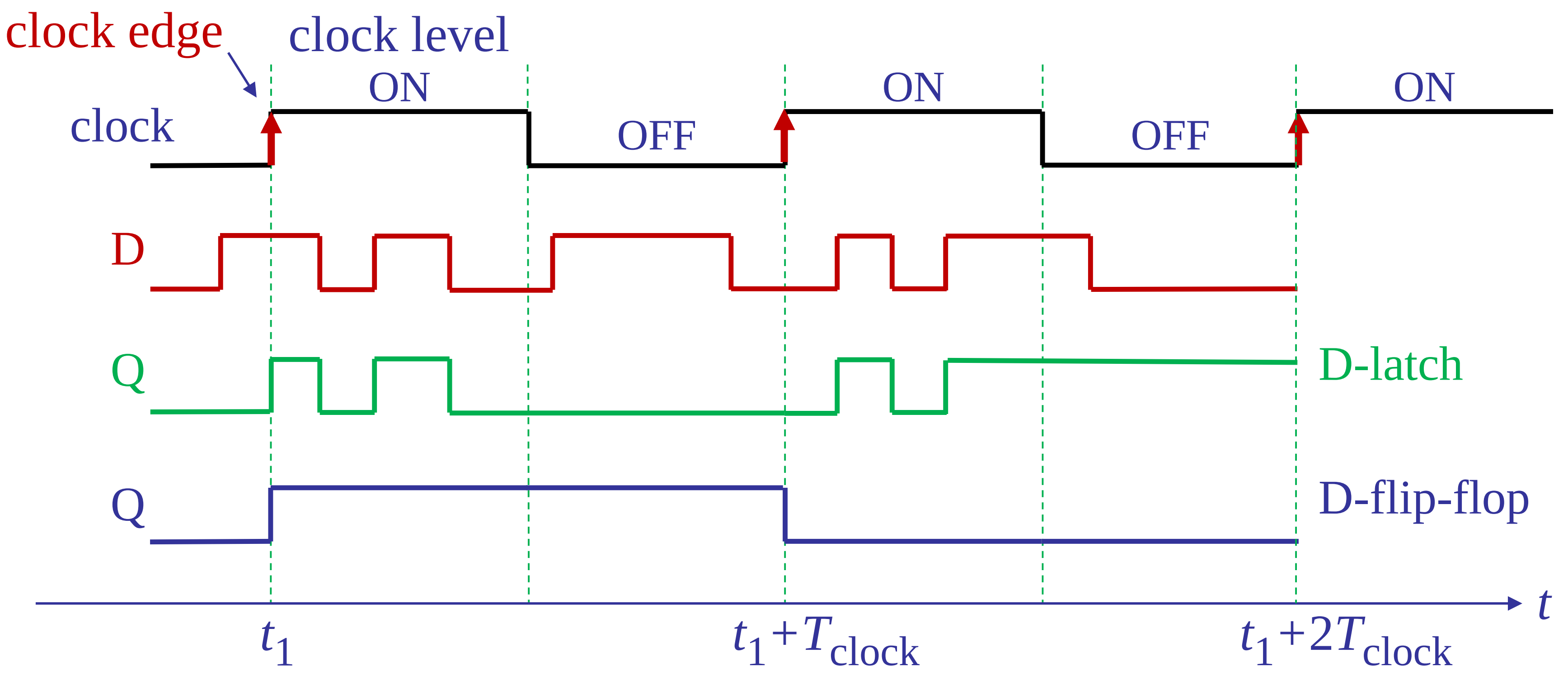
D flip-flop

edge-detector – generating an edge signal from the clock



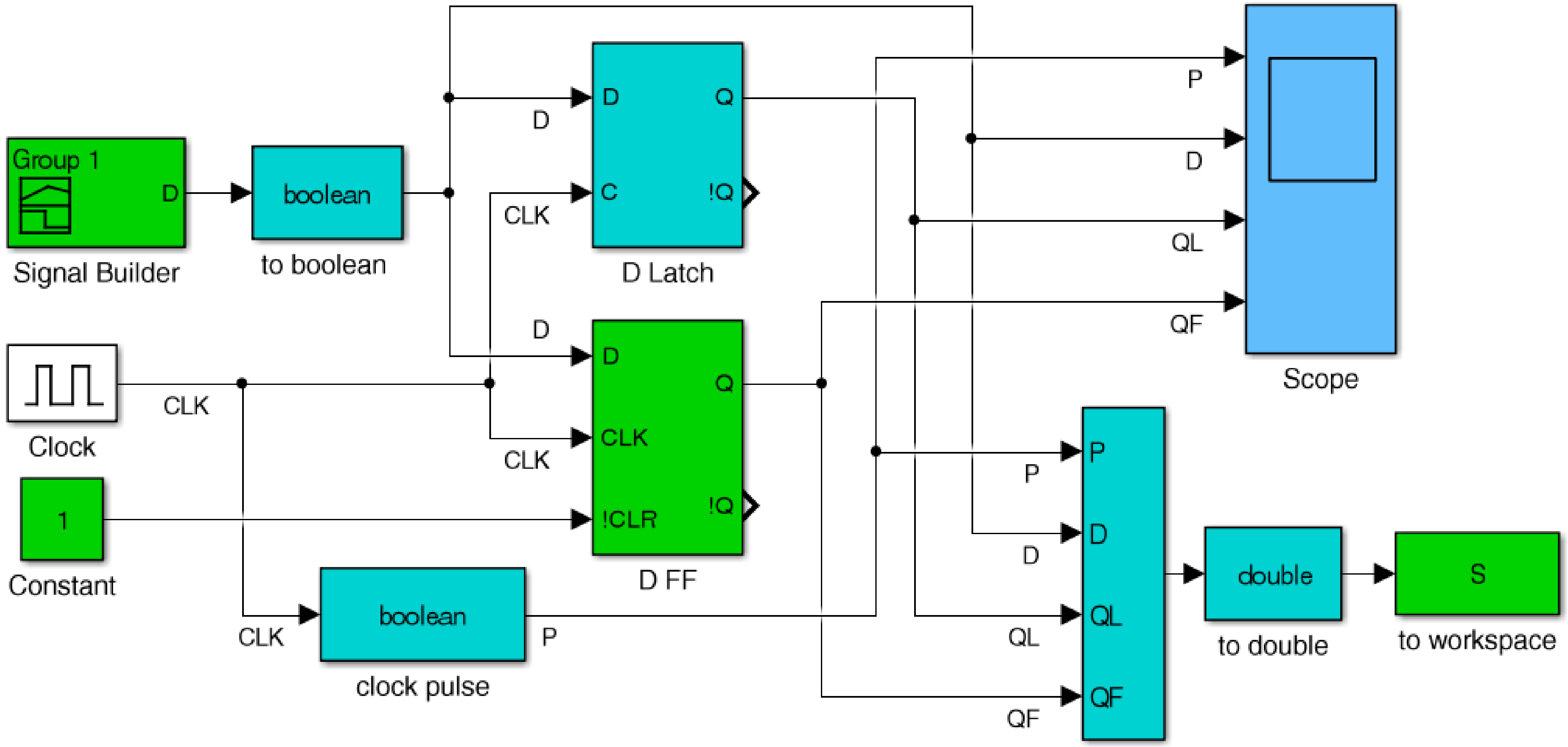
latches vs. flip-flops





D flip-flop

positive-edge-triggered D flip-flop vs. D latch

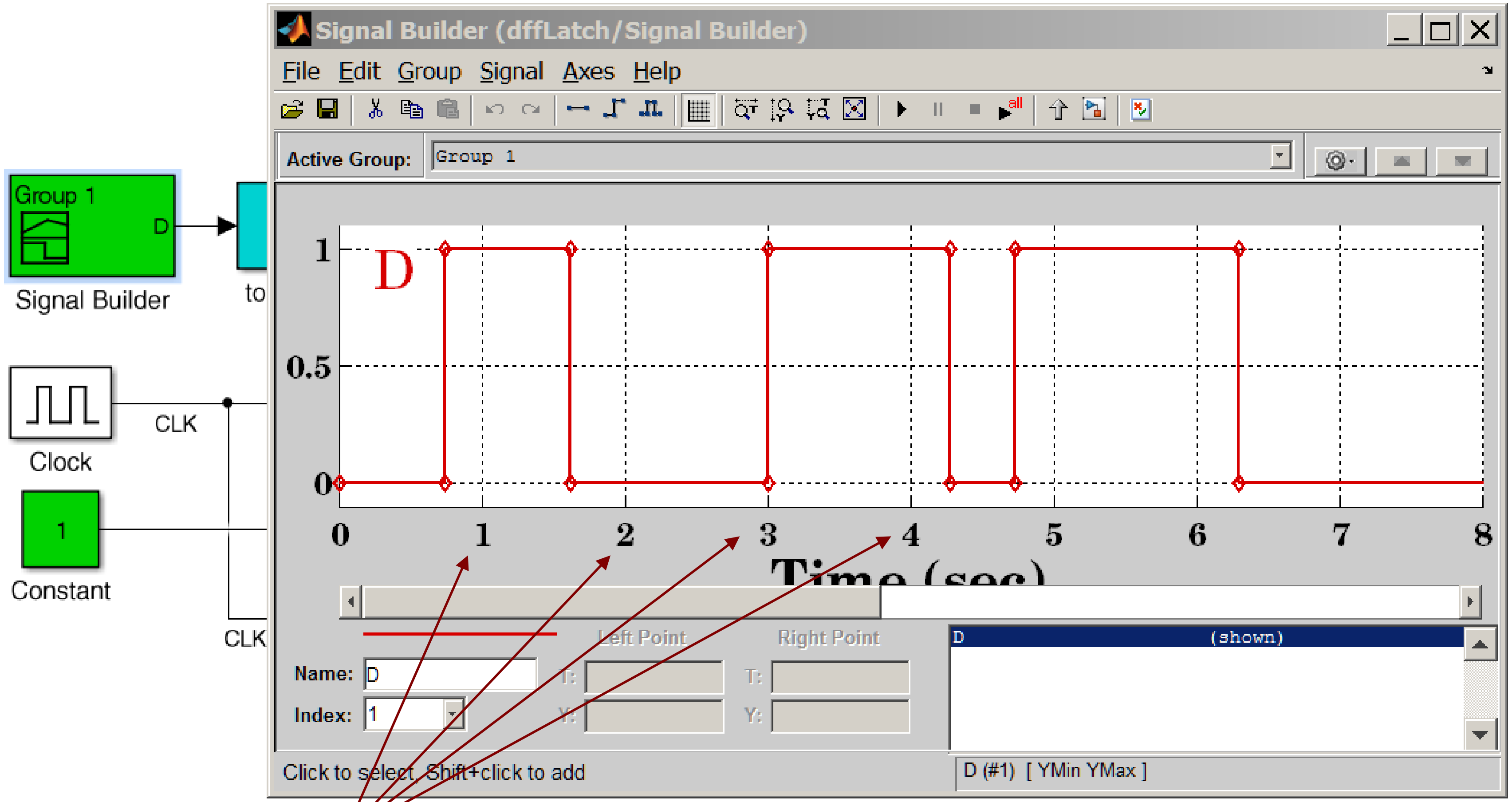


DffLs.slx file on Canvas

flip-flops and clock are found in Simulink library under Simulink extras/flip flops

D flip-flop

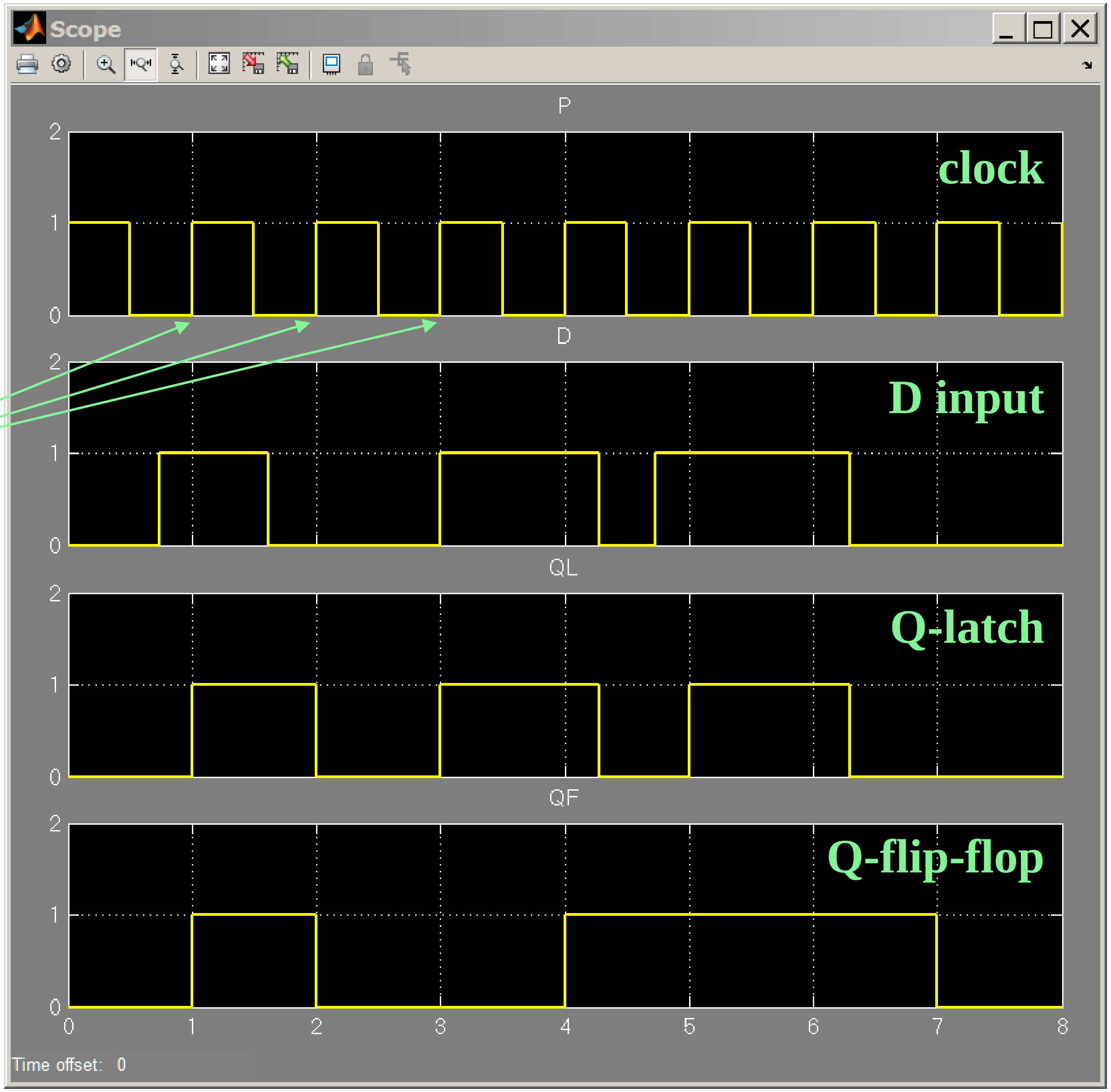
positive-edge-triggered D flip-flop vs. D latch



rising clock edges

D flip-flop

positive-edge-triggered D flip-flop vs. D latch

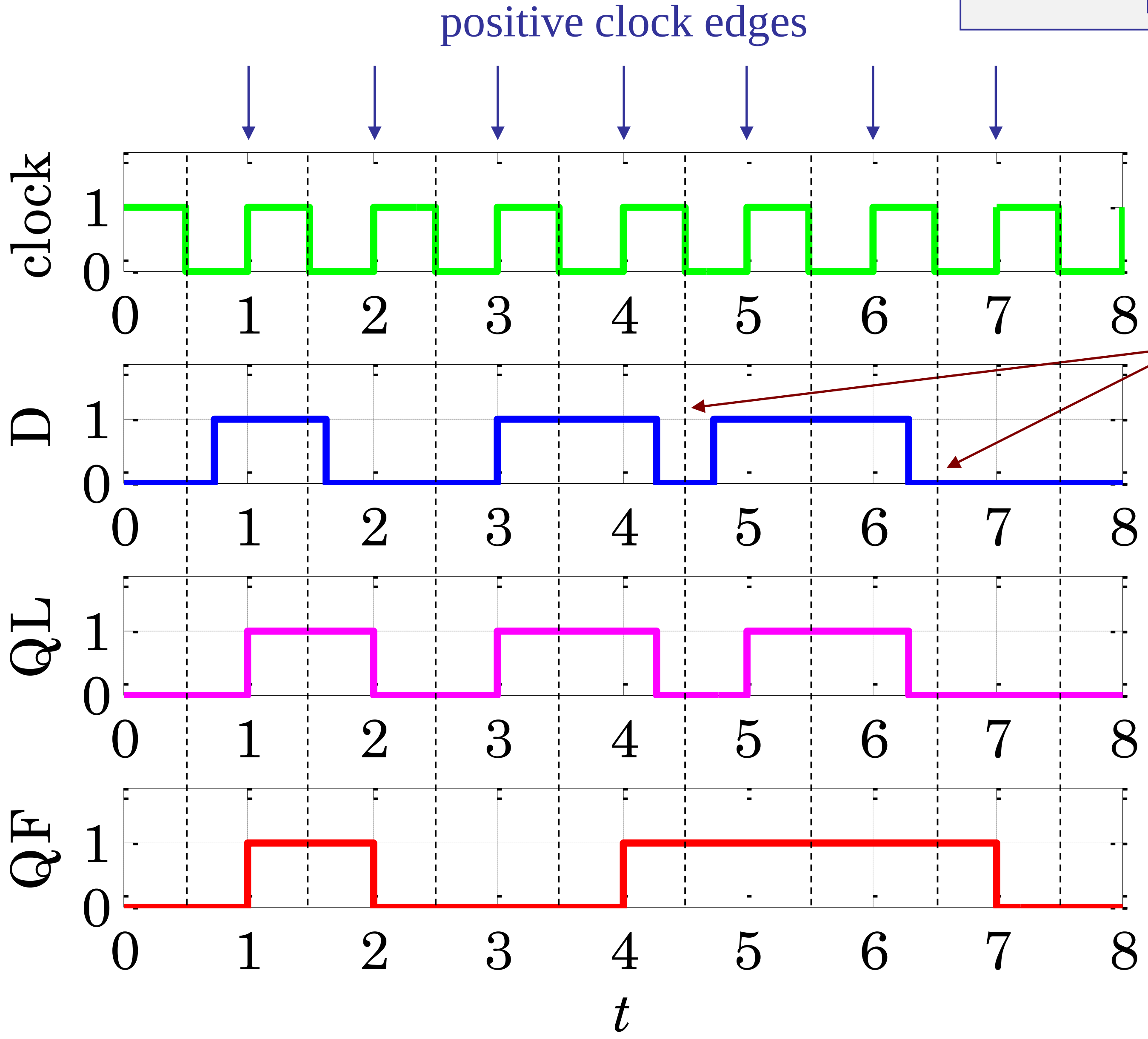


clock edges

D flip-flop

$$Q_{t+1} = \begin{cases} D_t\,, & \text{if } t \text{ at edge} \\ Q_t\,, & \text{if } t \text{ not at edge} \end{cases}$$

MATLAB code ⟶



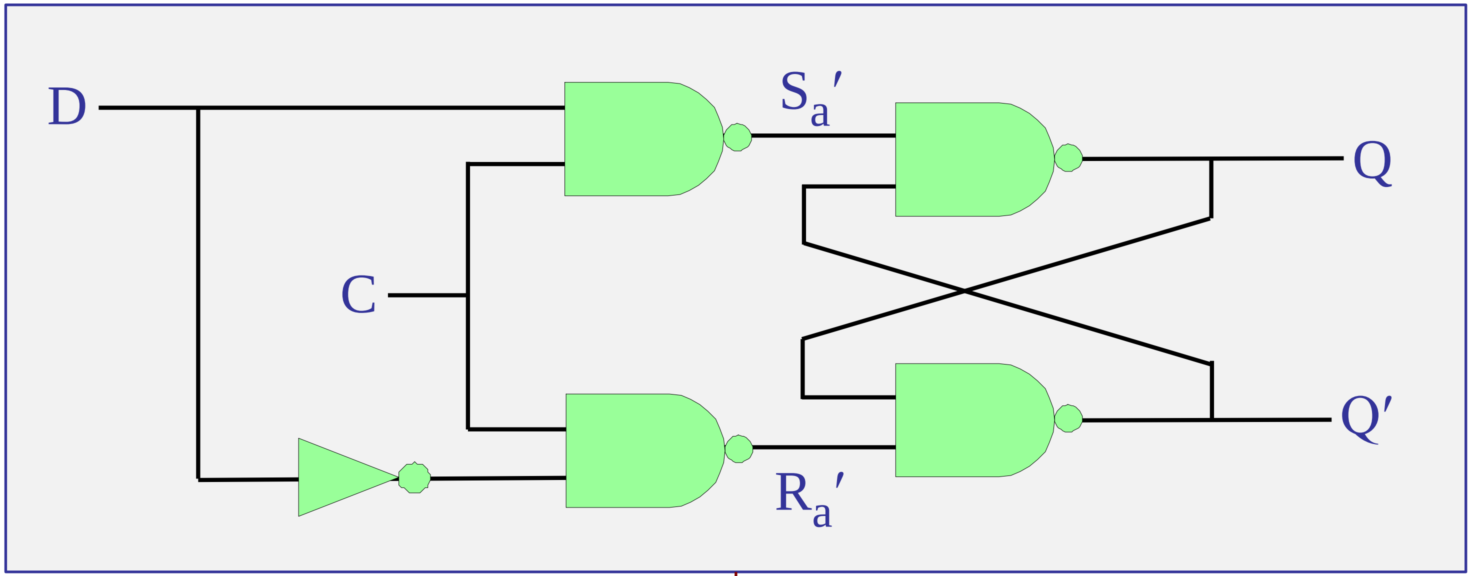
for the flip-flop, input variations are ignored between clock rising edges

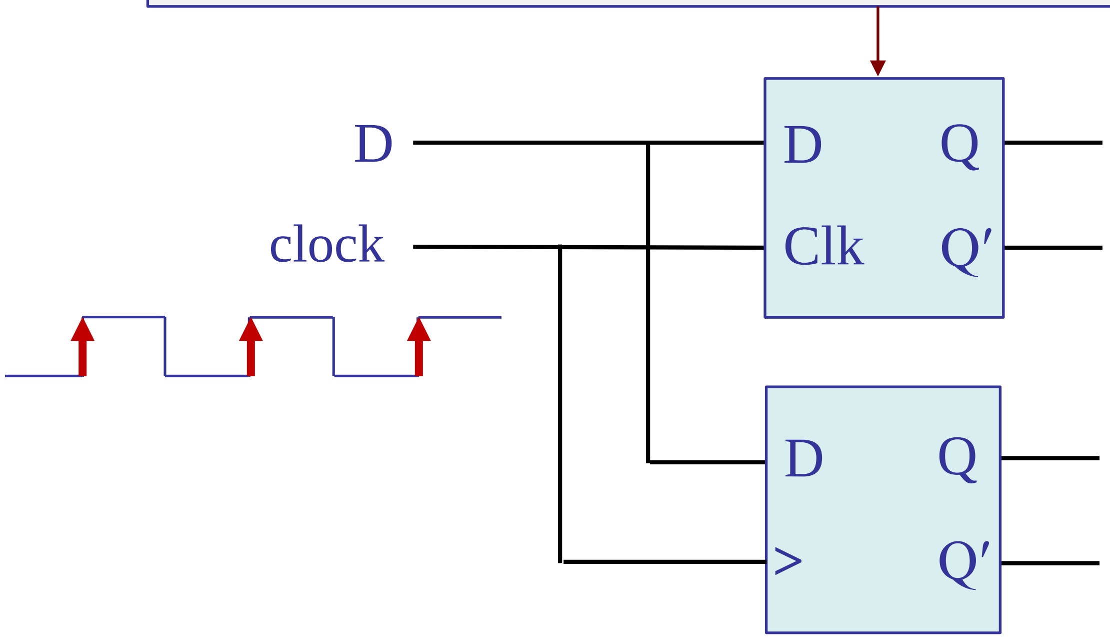
for the latch, input variations are applied only when clock is ON, and QL output is preserved while clock is OFF

D flip-flop

```
%% DffLm.m - D flip-flop vs. D latch - on Canvas
%  run DffLs.slx first to generate structure S

t = S.time;             % time
P = S.data(:,1);        % clock pulse
D = S.data(:,2);        % D input
QL = S.data(:,3);       % latch output
QF = S.data(:,4);       % flip-flop output

set(0,'DefaultAxesFontSize',14);

figure;
subplot(4,1,1); stairs(t,P,'g-','linewidth',2);
       xaxis(0,8,0:8); yaxis(0,1.9,0:1); ylabel('clock')
subplot(4,1,2); stairs(t,D,'b-','linewidth',2);
       xaxis(0,8,0:8); yaxis(0,1.9,0:1); ylabel('D'); grid
subplot(4,1,3); stairs(t,QL,'m-','linewidth',2);
       xaxis(0,8,0:8); yaxis(0,1.9,0:1); ylabel('QL'); grid
subplot(4,1,4); stairs(t,QF,'r-','linewidth',2);
       xaxis(0,8,0:8); yaxis(0,1.9,0:1); ylabel('QF'); grid
xlabel('{\itt}')
```

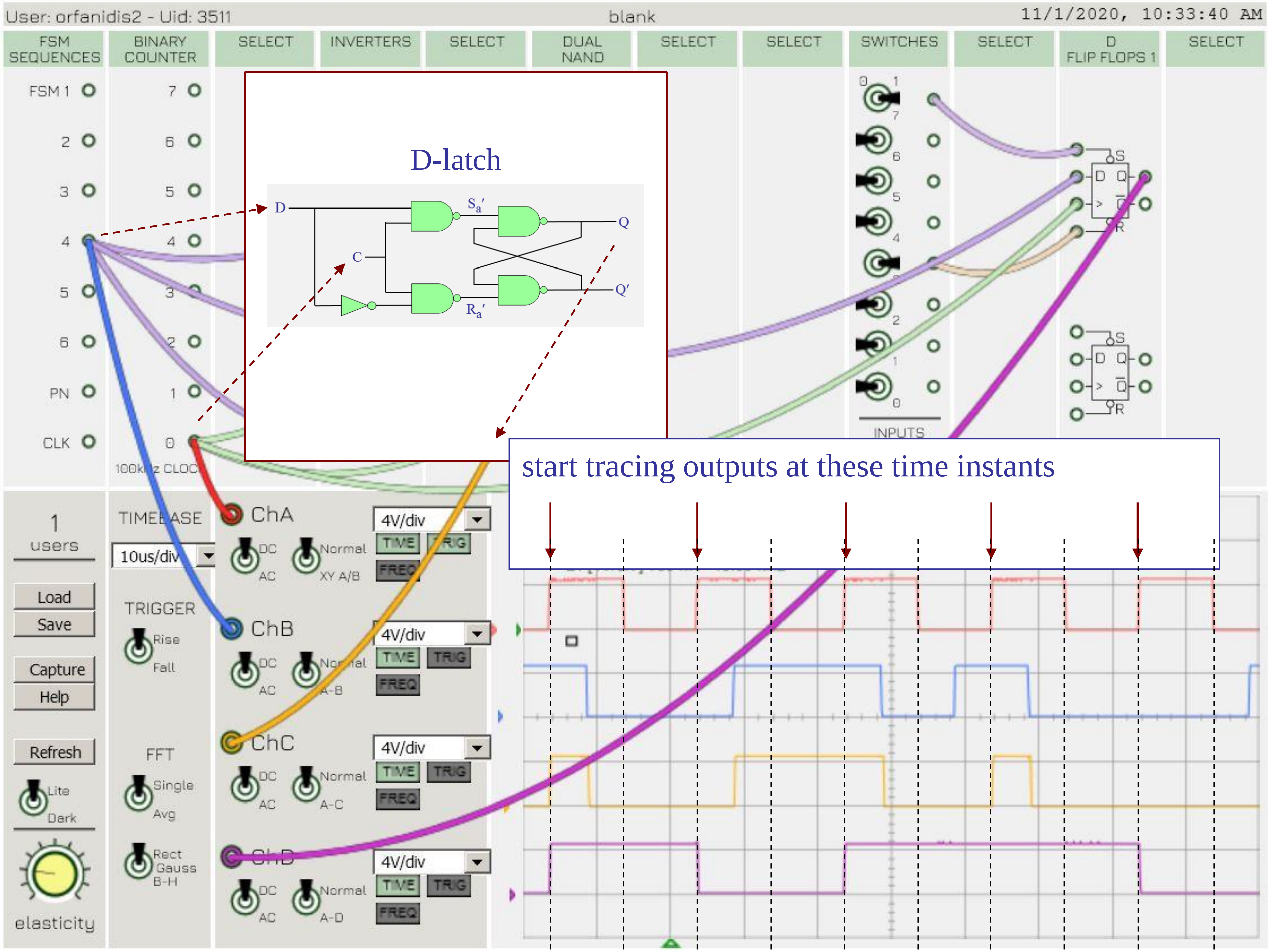
Emona lab 5 experiment – comparing D-latches with D-flip-flops





D-latch
level-sensitive

D-flip-flop
positive-edge-triggered



User: orfanidis2 - Uid: 3511

blank

11/1/2020, 10:33:40 AM

1 users

TIMEBASE 10us/div

ChA 4V/div

ChB 4V/div

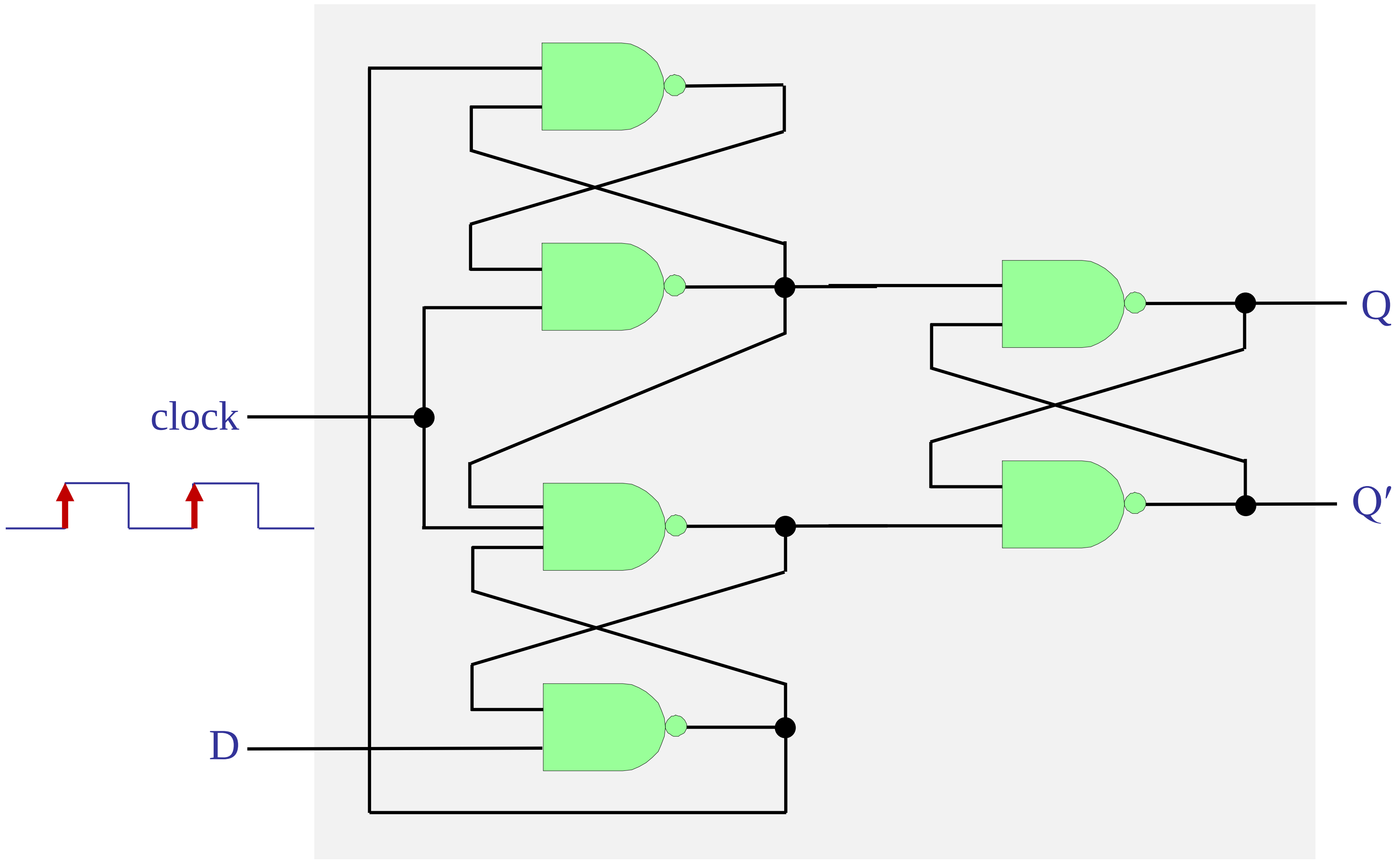
ChC 4V/div

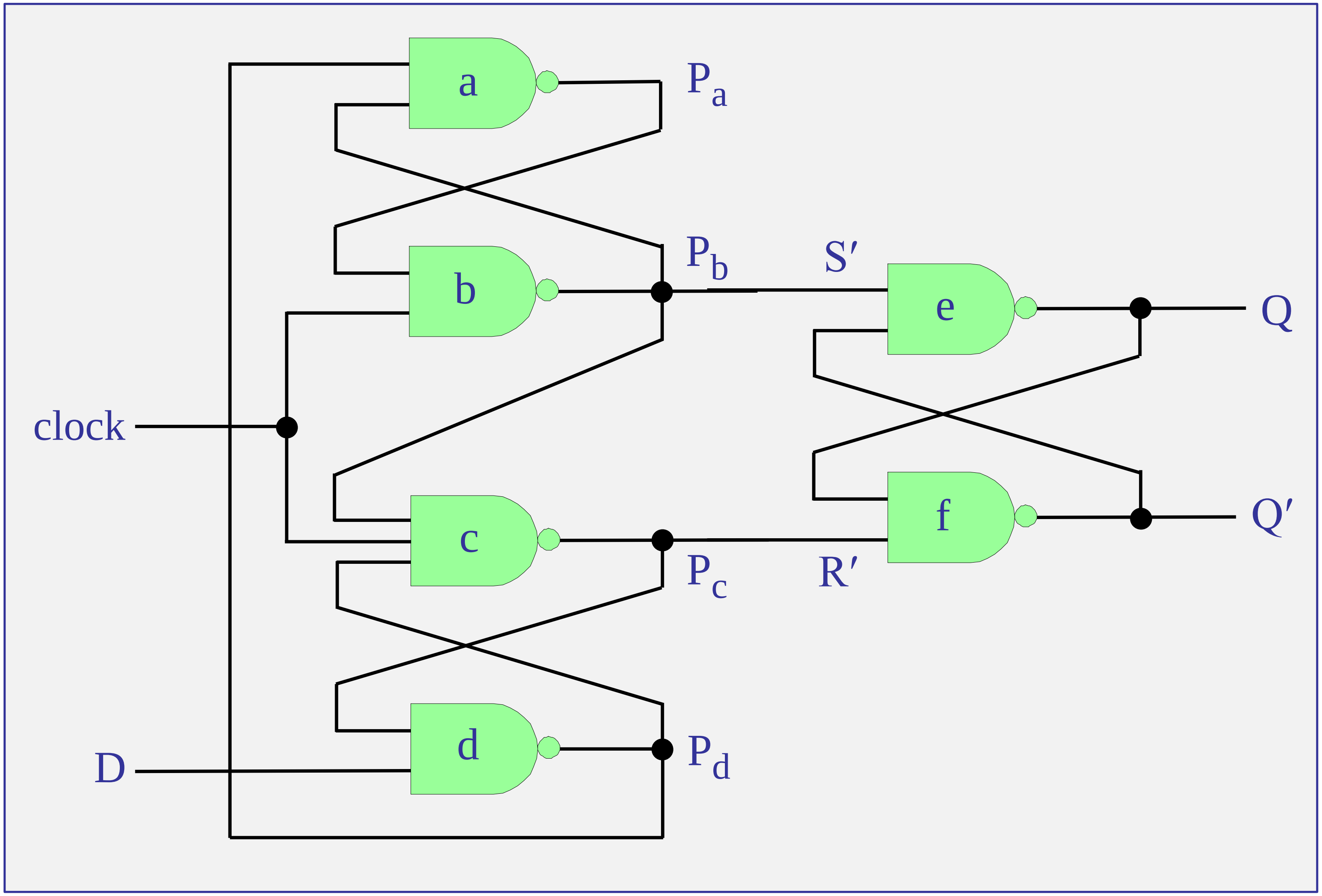
ChD 4V/div

start tracing outputs at these time instants

D flip-flop

positive-edge-triggered D flip-flop





See next page for an explanation of its operation with the help of the truth-table of an S′R′ latch. It will be explored further in the DLD lab (lab5).

When *clock* = 0, the outputs of gates b & c are $P_b = P_c = 1$, which maintains the output latch (gates e & f) in its present state. In addition, $P_d = D'$ and $P_a = D$.

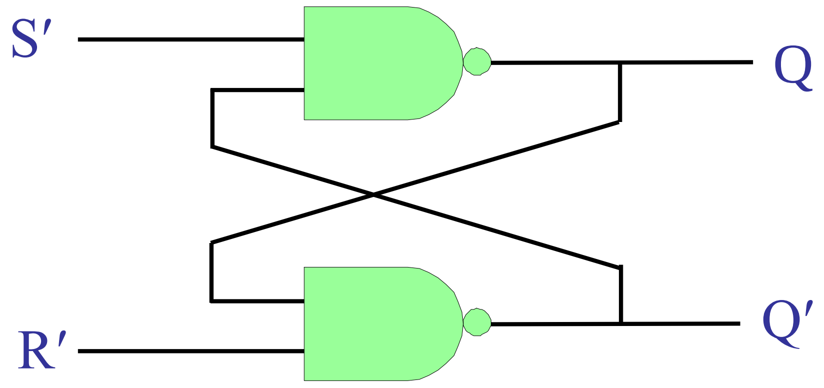
When the clock changes to *clock* = 1, then, the values of P_a and P_d are transmitted through gates b & c to cause $P_b = D'$ and $P_c = D$, thus, resulting in, $Q = D$ and $Q' = D'$.

After *clock* changes to 1, any further changes in D should not affect the output latch, as long as, *clock* = 1. There are two possibilities:

(a) if $D = 0$ at the positive edge of the clock, then, $P_c = 0$, keeping the output $P_d = 1$, as long as, *clock* = 1, regardless of the value of the D input, and maintaining $Q = 0 = D_{edge}$.

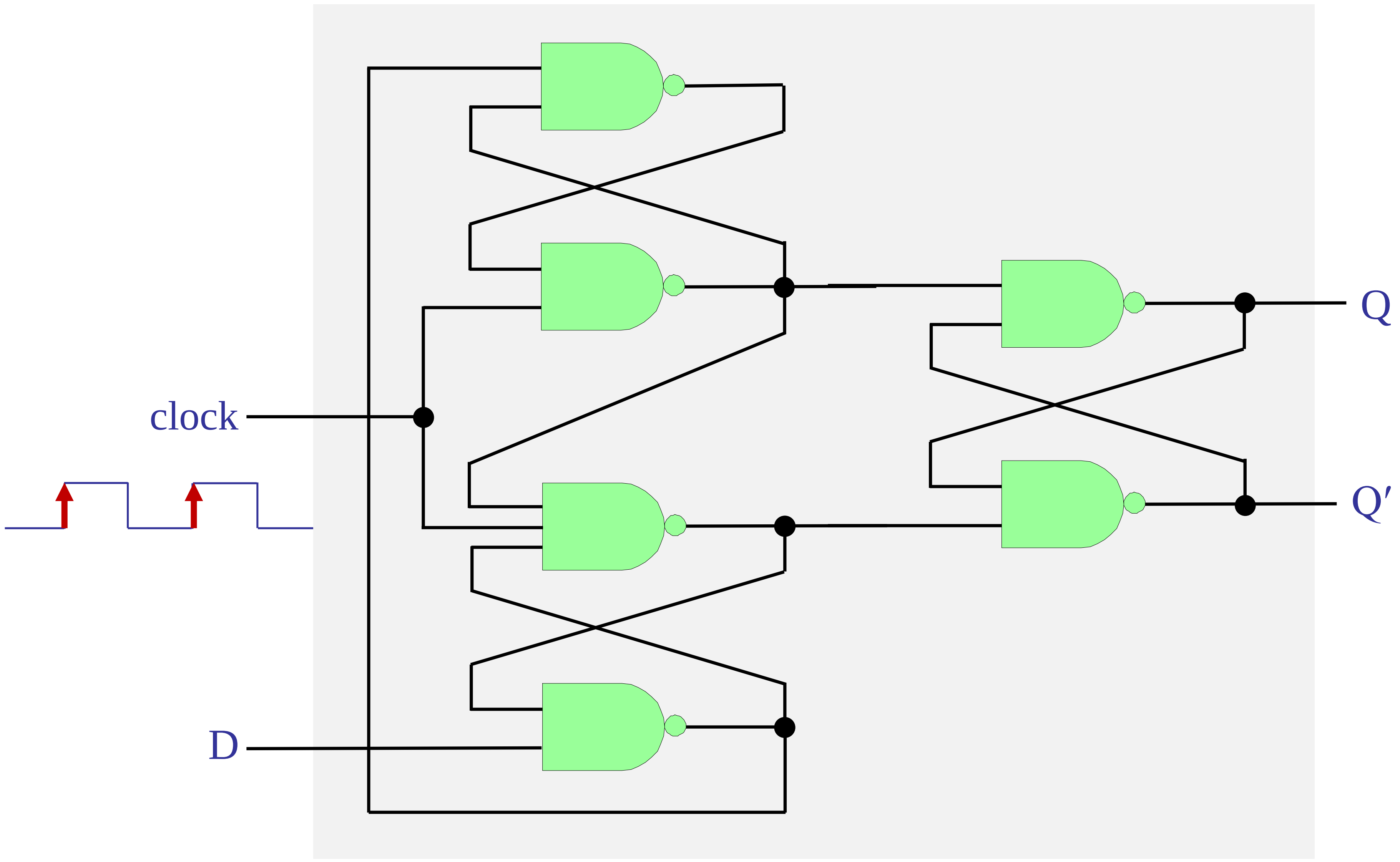
(b) if $D = 1$ at the positive edge of the clock, then, $P_b = 0$, forcing the outputs, $P_a = 1$, $P_c = 1$, regardless of the D input, and maintaining the output equal to $Q = 1 = D_{edge}$.

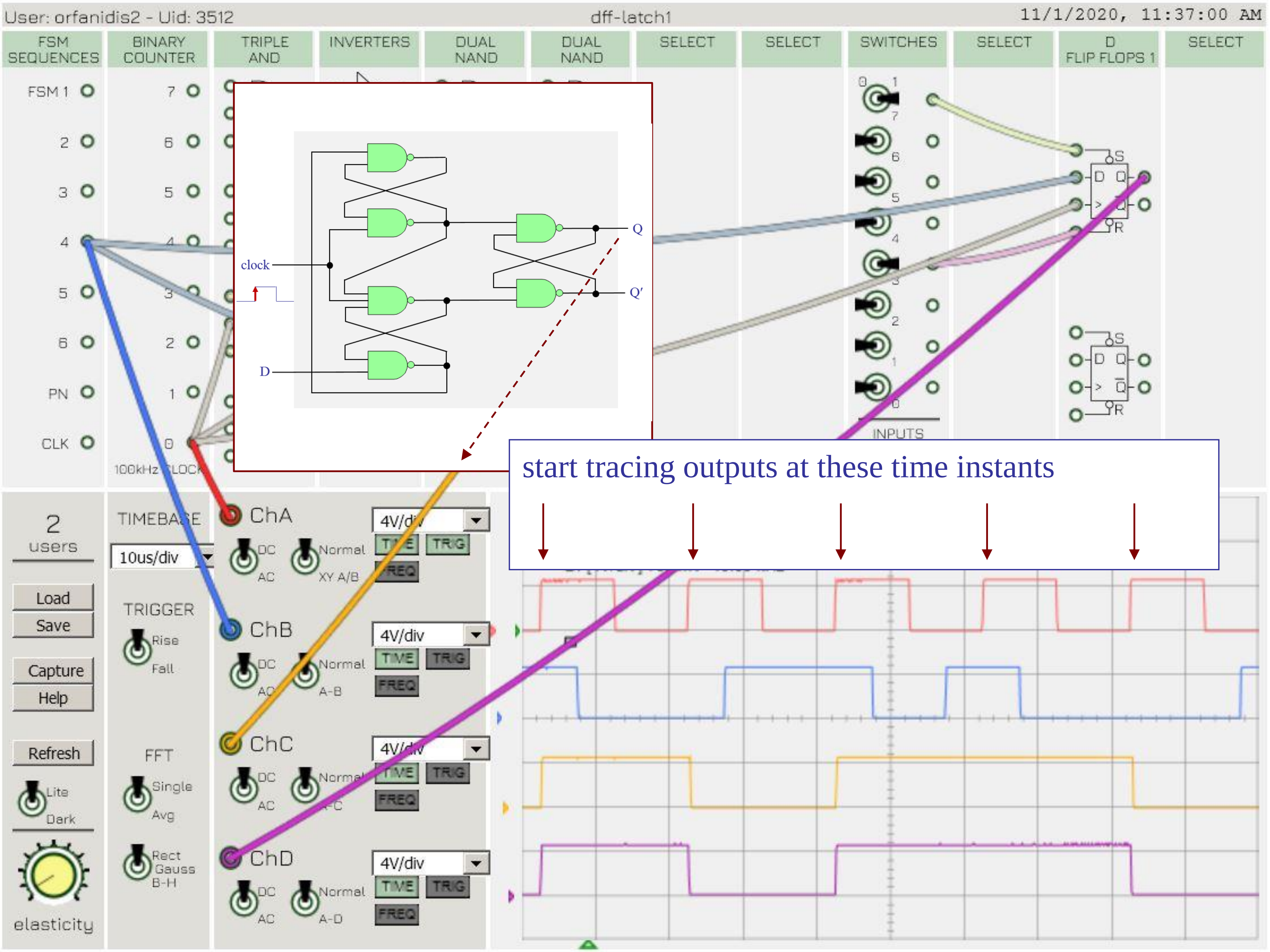
Therefore, the flip-flop ignores changes in the D input, while *clock* = 1. Hence, the circuit behaves as a positive-edge-triggered flip-flop.



S′	R′	Q	Q_{next}	Q'_{next}
0	0	Q	1	1
0	1	Q	1	0
1	0	Q	0	1
1	1	Q	Q	Q′

Emona lab 5 experiment – verifying the six-NAND implementation

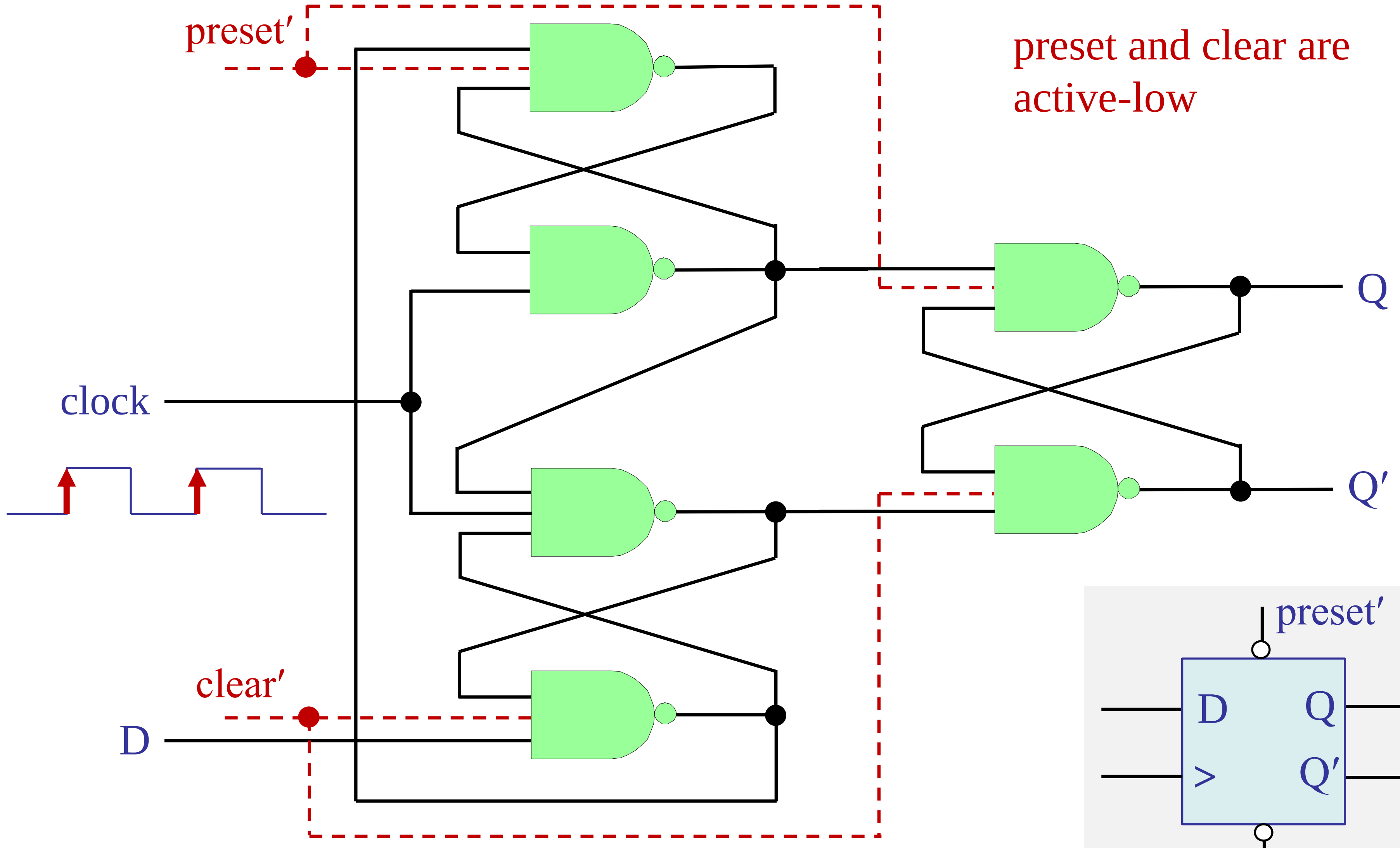




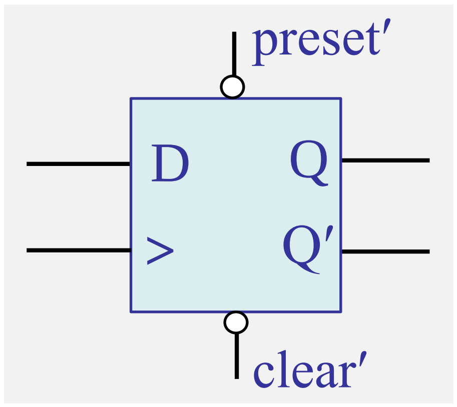
start tracing outputs at these time instants

D flip-flop

positive-edge-triggered D flip-flop – with preset/clear



preset and clear are active-low



D flip-flop

D flip-flop ICs

54LS74

74LS74

from Motorola, TI, Fairchild

MOTOROLA

DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOP

The SN54/74LS74A dual edge-triggered flip-flop utilizes Schottky TTL circuitry to produce high speed D-type flip-flops. Each flip-flop has individual clear and set inputs, and also complementary Q and $\overline{Q}$ outputs.

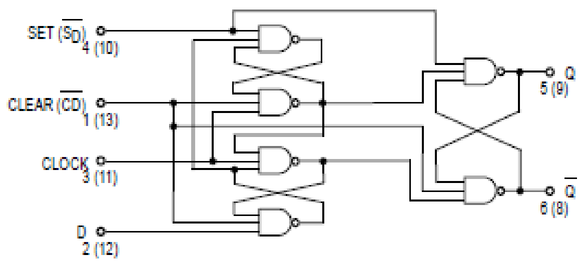
Information at input D is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the HIGH or the LOW level, the D input signal has no effect.

SN54/74LS74A

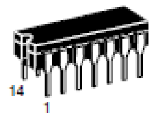
DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOP

LOW POWER SCHOTTKY

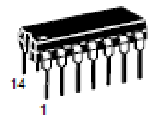
LOGIC DIAGRAM (Each Flip-Flop)



SET ($\overline{S_D}$) 4 (10)
CLEAR ($\overline{C_D}$) 1 (13)
CLOCK 3 (11)
D 2 (12)
Q 5 (9)
$\overline{Q}$ 6 (8)



J SUFFIX
CERAMIC
CASE 632-08



N SUFFIX
PLASTIC
CASE 646-06

D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

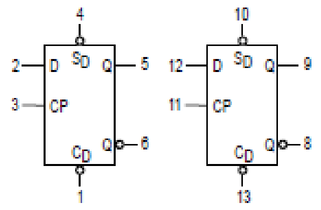
MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	$\overline{S}_D$	$\overline{S}_D$	D	Q	$\overline{Q}$
Set	L	H	X	H	L
Reset (Clear)	H	L	X	L	H
*Undetermined	L	L	X	H	H
Load "1" (Set)	H	H	h	H	L
Load "0" (Reset)	H	H	l	L	H

* Both outputs will be HIGH while both $\overline{S}_D$ and $\overline{C}_D$ are LOW, but the output states are unpredictable if $\overline{S}_D$ and $\overline{C}_D$ go HIGH simultaneously. If the levels at the set and clear are near V_{IL} maximum then we cannot guarantee to meet the minimum level for V_{OH}.

H, h = HIGH Voltage Level
L, l = LOW Voltage Level
X = Don't Care
l, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the HIGH to LOW clock transition.

LOGIC SYMBOL



V_{CC} = PIN 14
GND = PIN 7

D flip-flop

D flip-flop ICs

54LS74

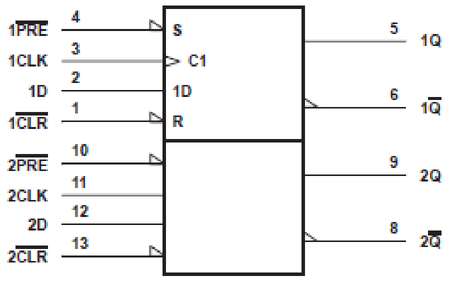
74LS74

from Motorola, TI, Fairchild

SN54ALS74A, SN54AS74A, SN74ALS74A, SN74AS74A
DUAL POSITIVE-EDGE-TRIGGERED D-TYPE FLIP-FLOPS
WITH CLEAR AND PRESET

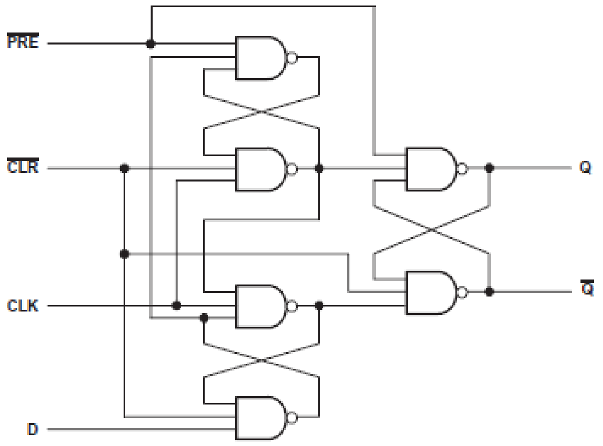
SDAS143C – APRIL 1982 – REVISED AUGUST 1995

logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.
Pin numbers shown are for the D, J, and N packages.

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)‡

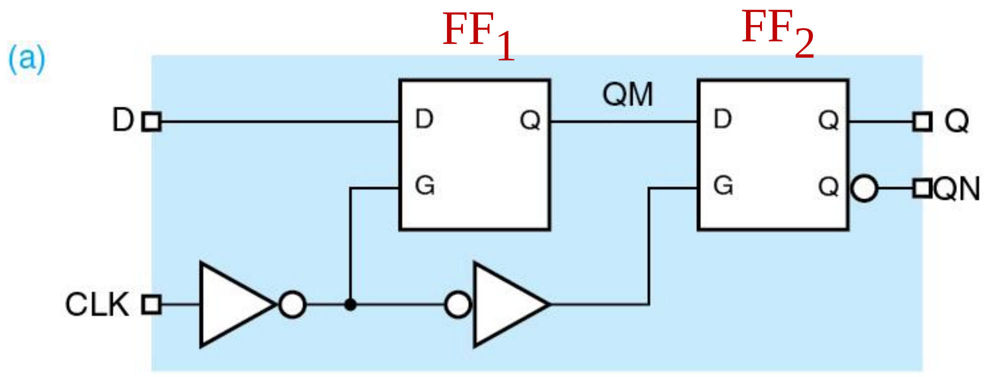
Supply voltage, V_{CC}		7 V
Input voltage, V_I		7 V
Operating free-air temperature range, T_A:	SN54ALS74A	−55°C to 125°C
	SN74ALS74A	0°C to 70°C
Storage temperature range		−65°C to 150°C

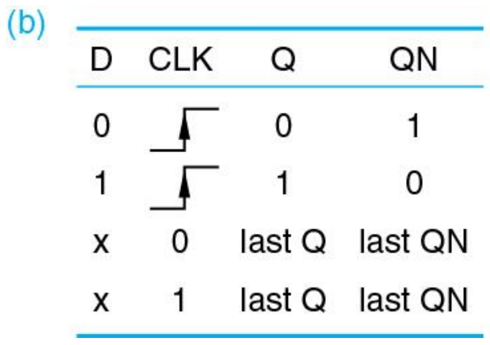
‡ Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

TEXAS INSTRUMENTS

D flip-flop

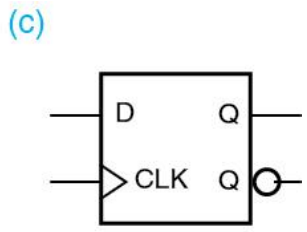
positive-edge-triggered D flip-flop
can also be constructed by cascading two D-latches,
but driven by opposite clocks, see Wakerly, Sect. 10.2.4





(b)

D	CLK	Q	QN
0	↑	0	1
1	↑	1	0
x	0	last Q	last QN
x	1	last Q	last QN



when CLK = 0, FF$_1$ is open and follows its input, D

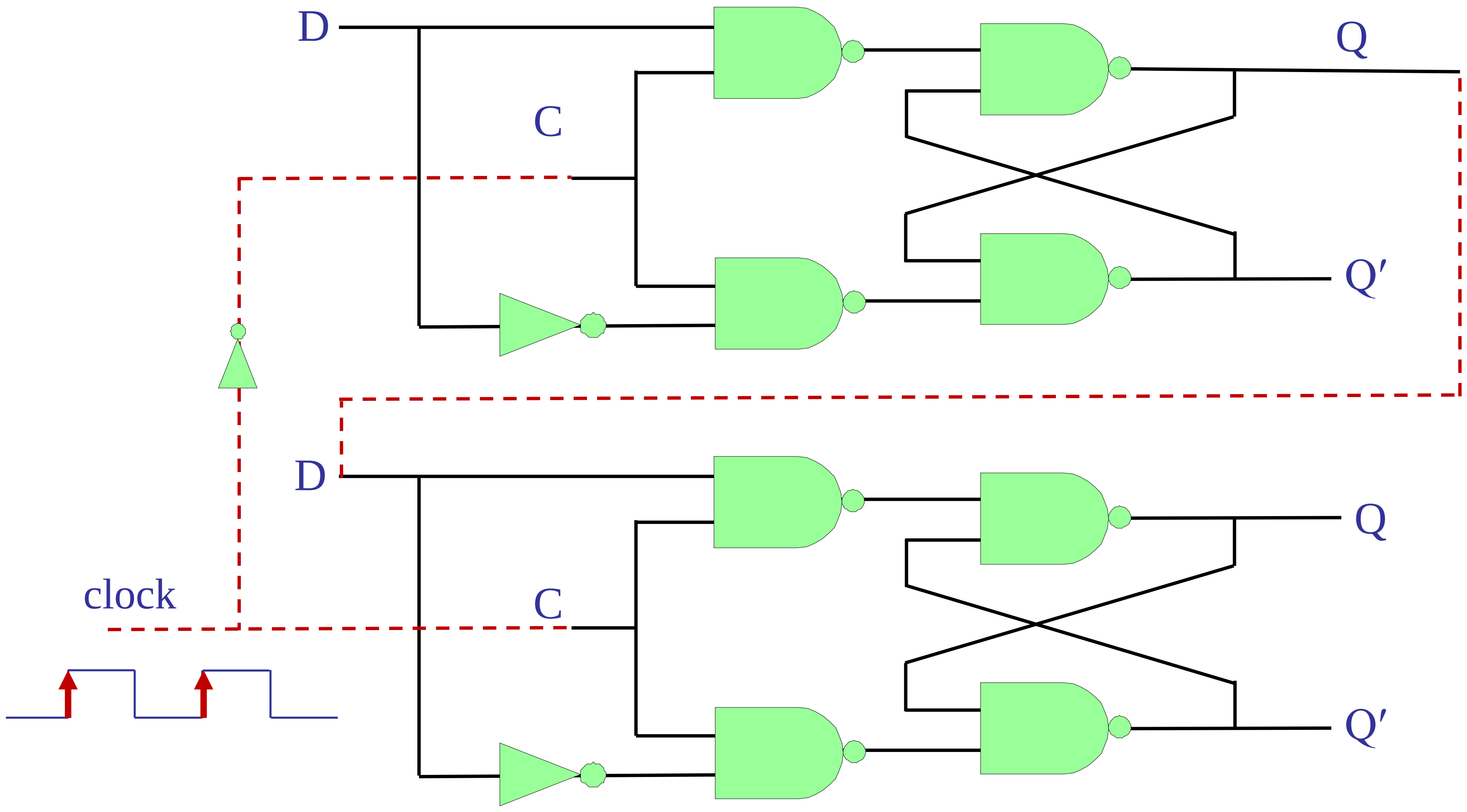
when CLK = 1, FF$_1$ is closed and its current output QM is transferred to FF$_2$'s output Q, and QM is prevented from changing until CLK=0 again,

FF$_2$ remains open while CLK = 1, but changes only at the rising edge of that interval because FF$_1$ is closed and not changing during the rest of the interval,

so effectively, D is transferred to Q only at the rising edges (0 to 1) of the clock period and Q maintains its state during the rest of the clock period

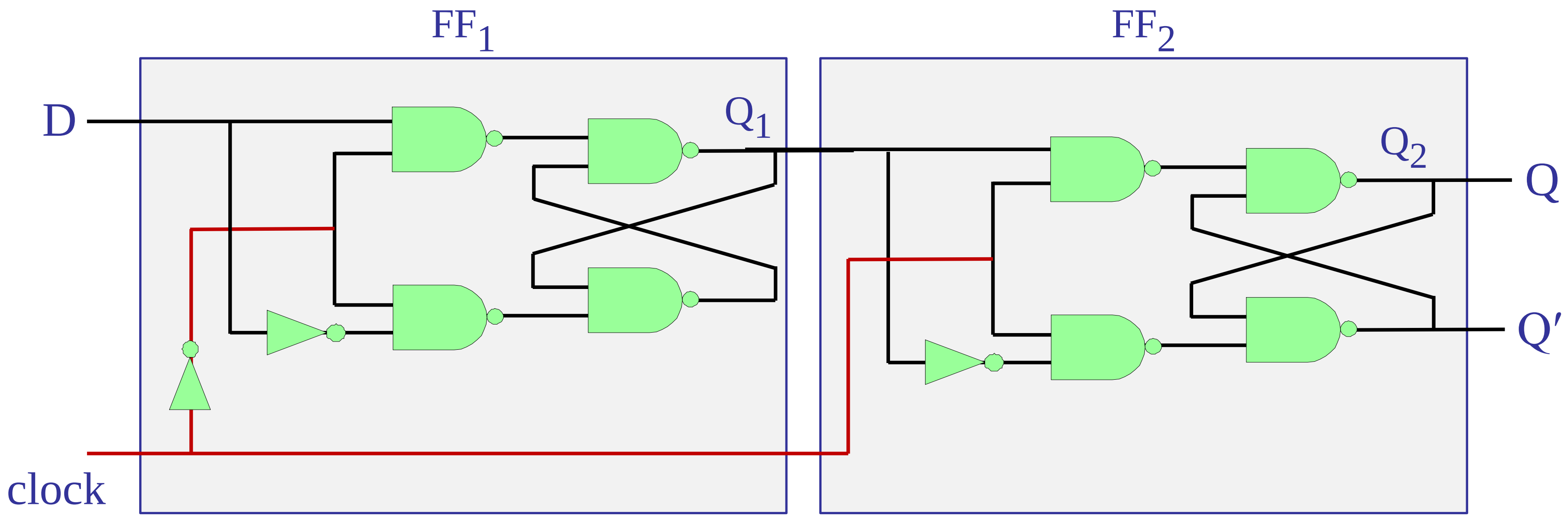
D flip-flop

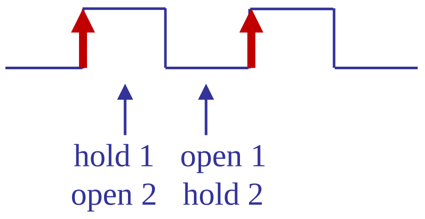
cascading two D-latches together, and tying their control signals to opposite clocks



D flip-flop

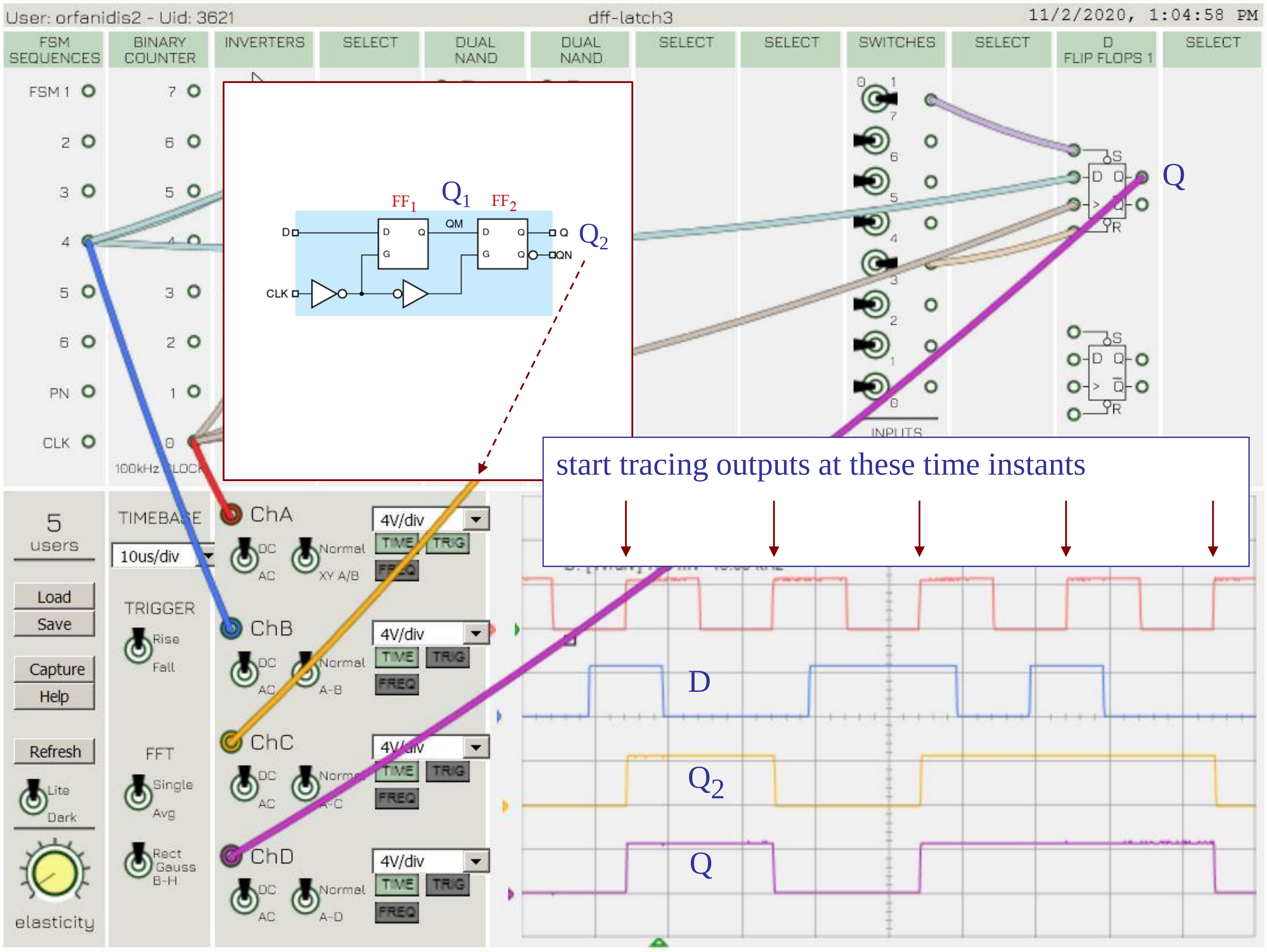
cascaded D latches



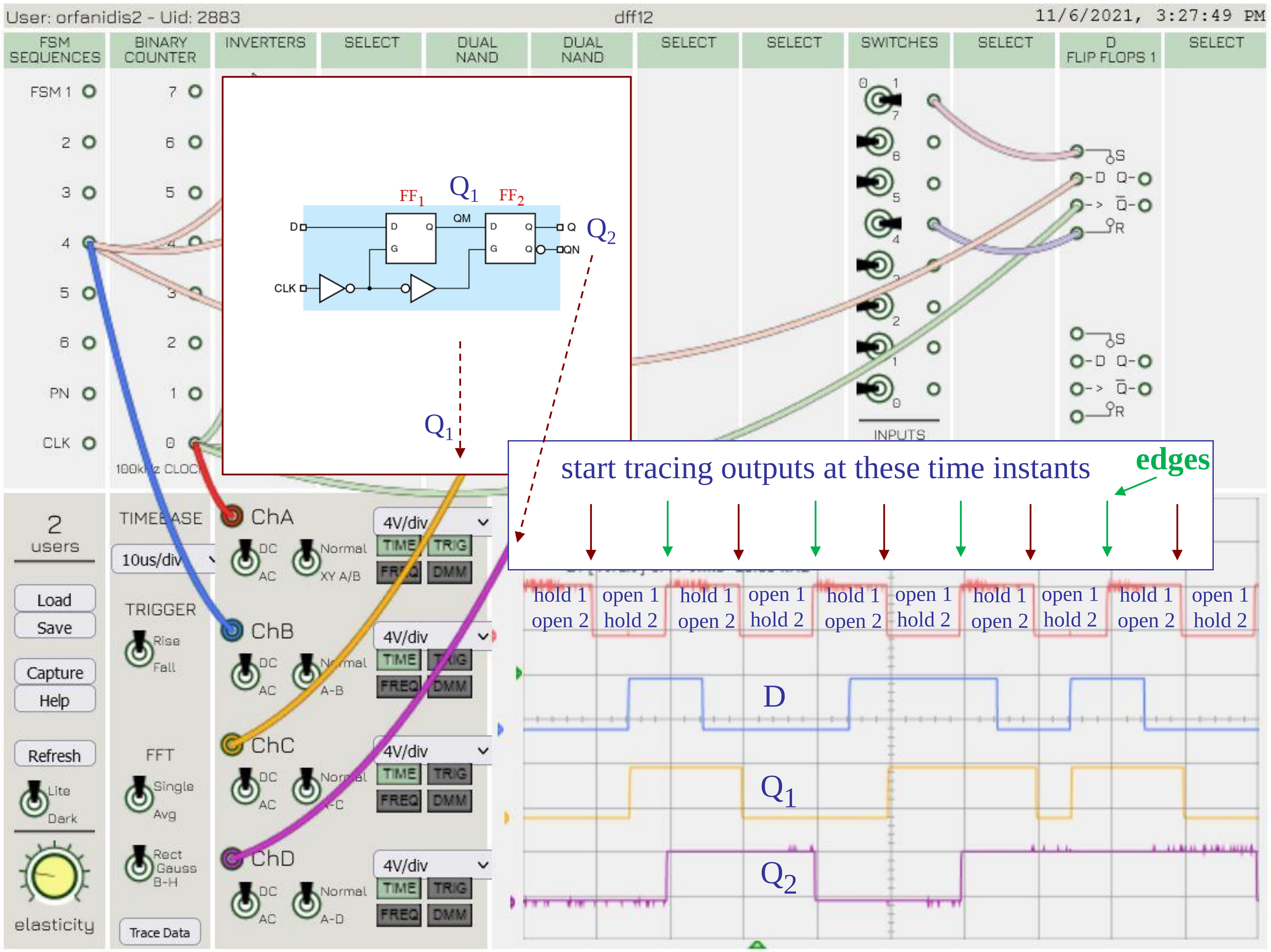


see Wakerly, Fig.10-13 for a timing diagram

This implementation will be explored in lab5, but note however, that the former implementation that uses three SR-latches (p. 53) is slightly more efficient, since it requires six NAND gates instead of eight, and is used in commercially available D-flip-flop ICs (see p. 54-55).



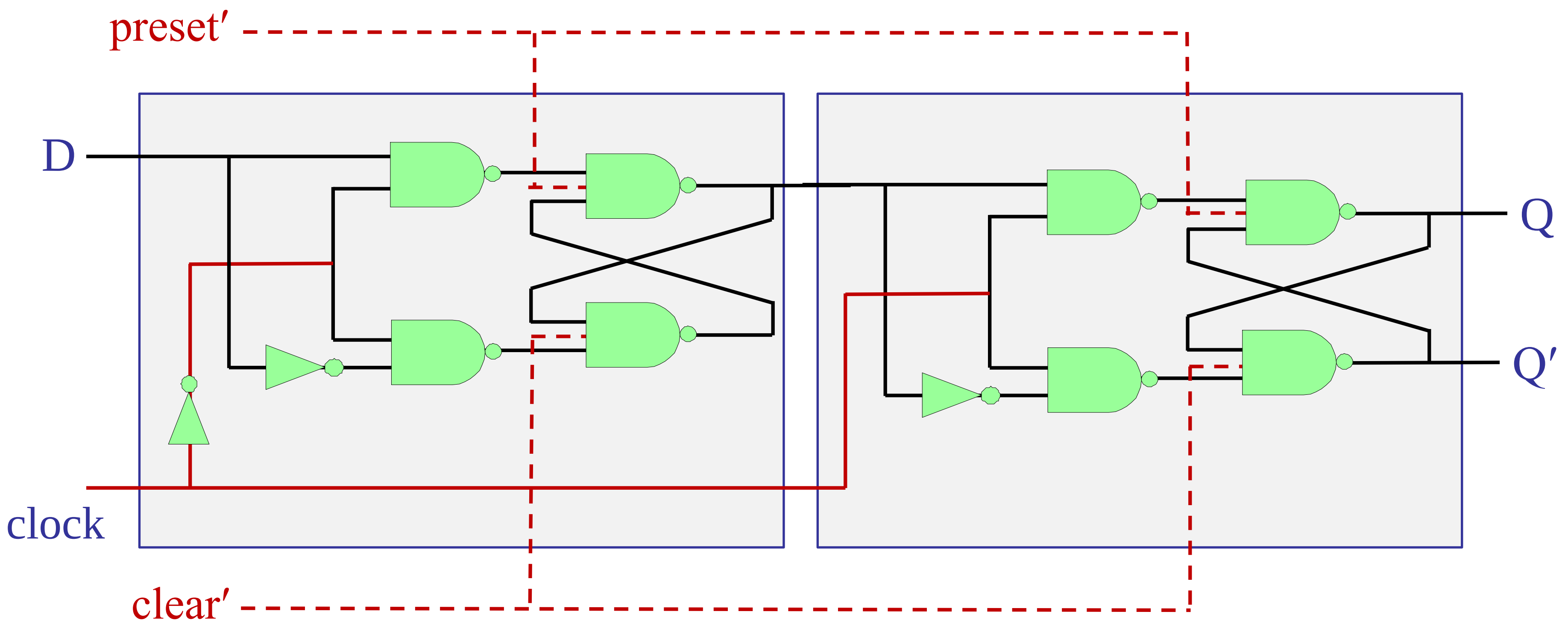
start tracing outputs at these time instants



start tracing outputs at these time instants

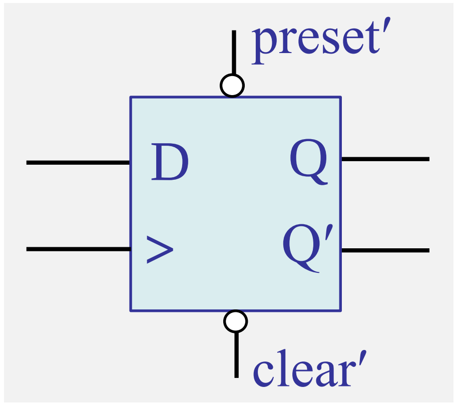
D flip-flop

adding preset/clear inputs



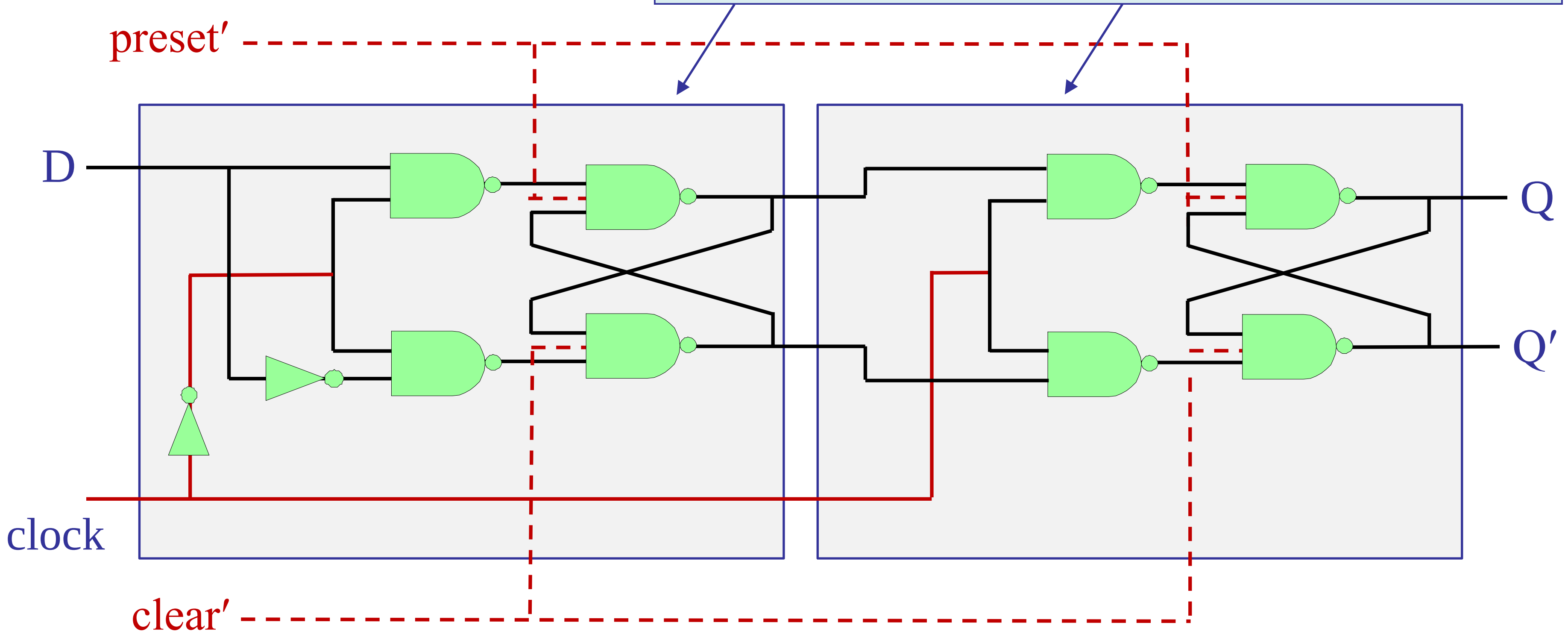
preset and clear are active-low

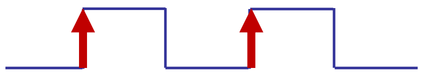
preset′ = 0, sets Q = 1
clear′ = 0, sets Q = 0
preset′ = 1, has no effect
clear′ = 1, has no effect



D flip-flop

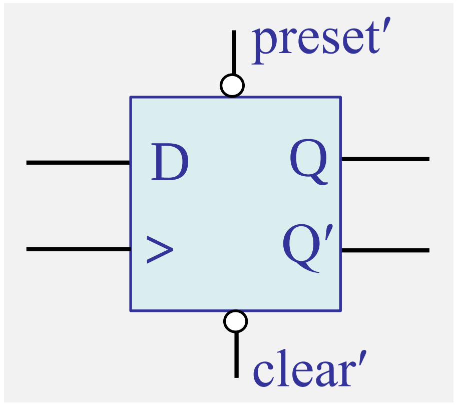
cascaded D latches: slight variation that uses a D latch followed by an SR latch



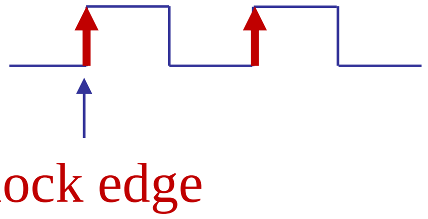


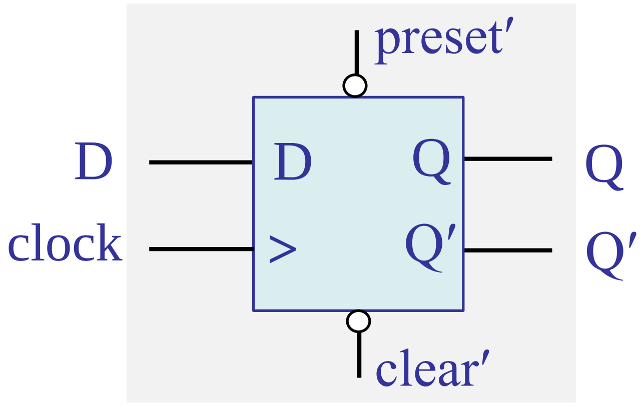
preset and clear are active-low

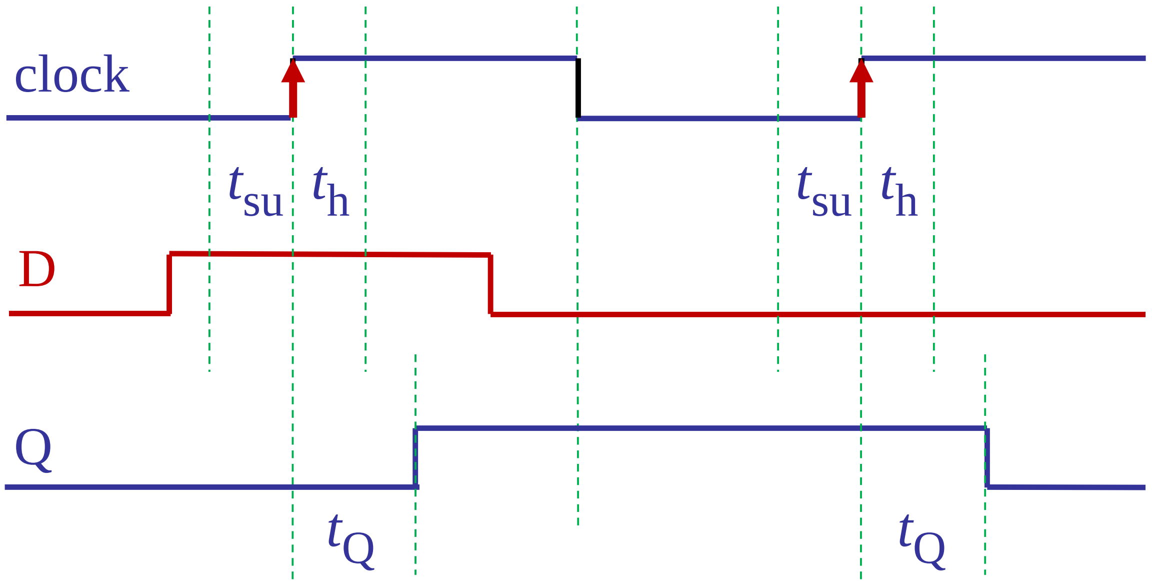
preset′ = 0, sets Q = 1
clear′ = 0, sets Q = 0
preset′ = 1, has no effect
clear′ = 1, has no effect



D flip-flop - timing parameters







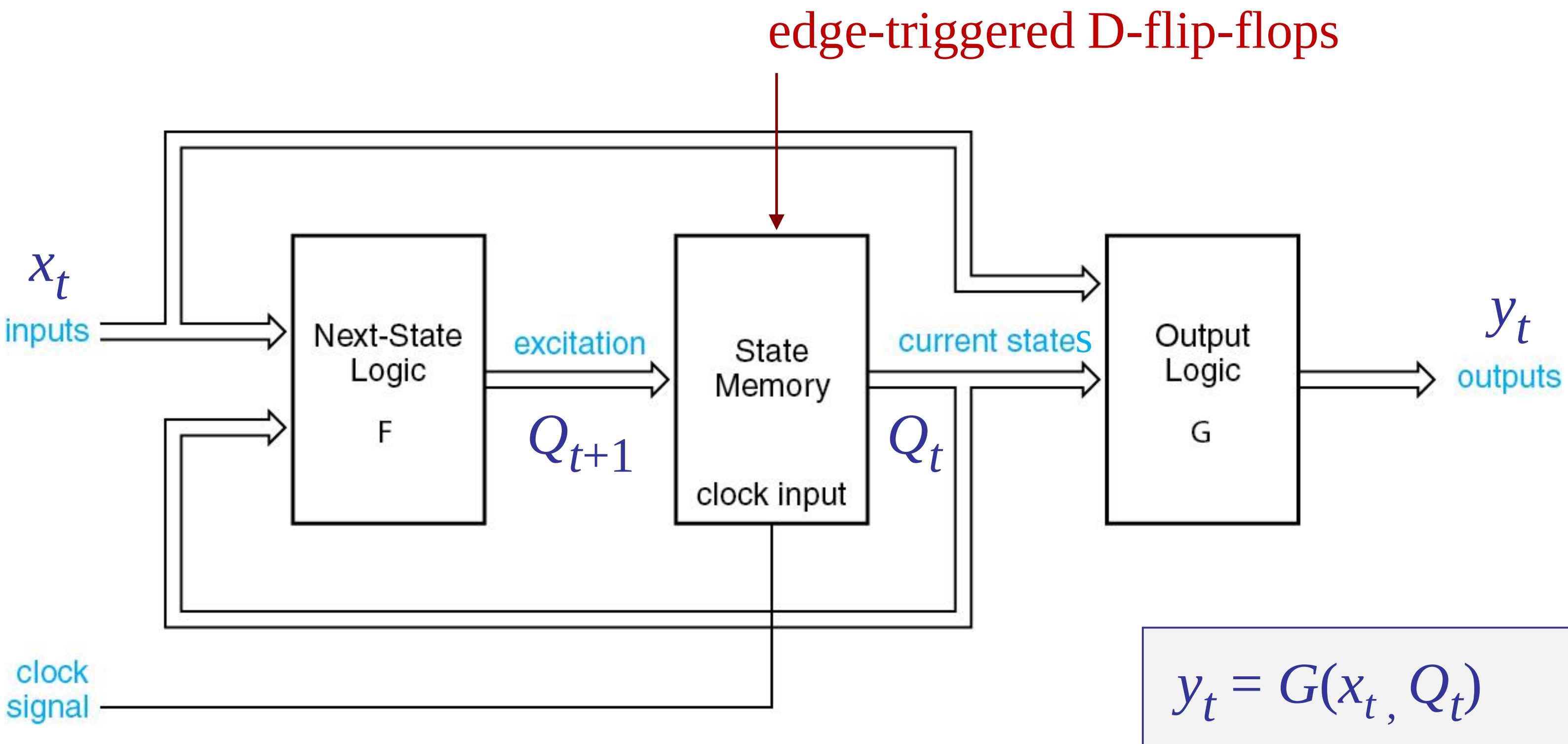
t_{su} = setup time

t_h = hold time

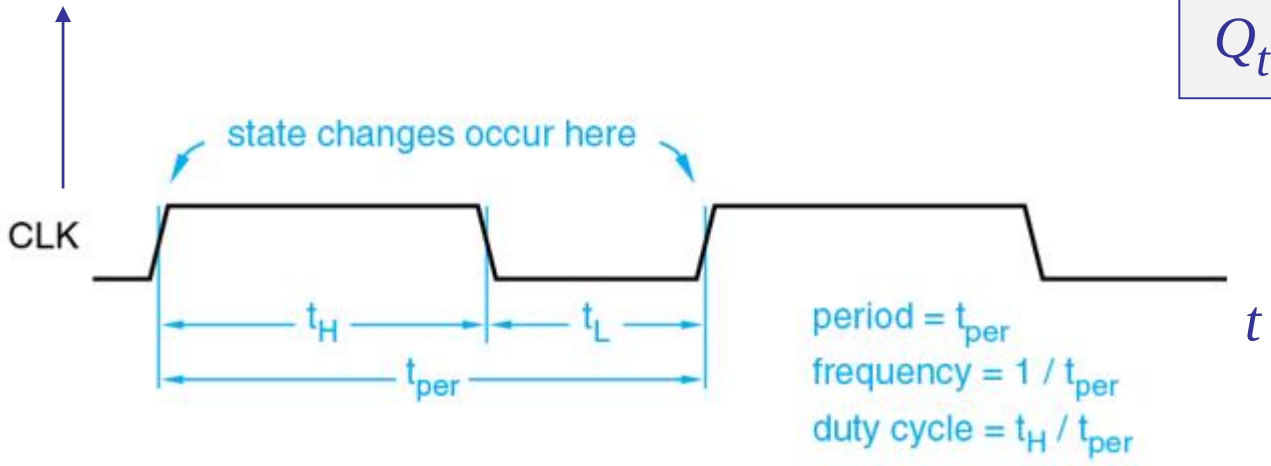
t_Q = clock-to-Q delay

all typically, 1-20 ns

D flip-flops and State Machines



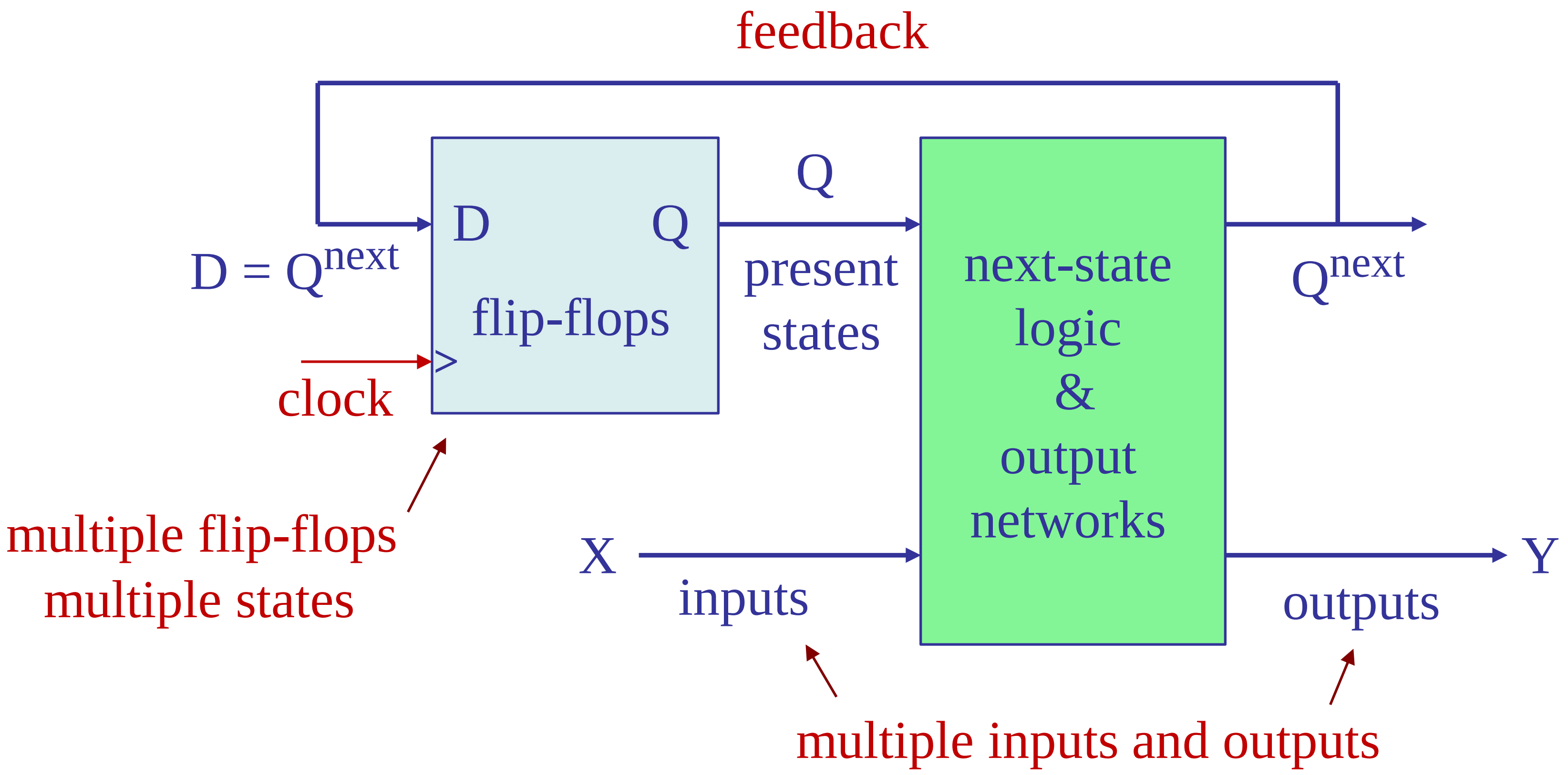
$$y_t = G(x_t\,,\ Q_t)$$
$$Q_{t+1} = F(x_t\,,\ Q_t)$$



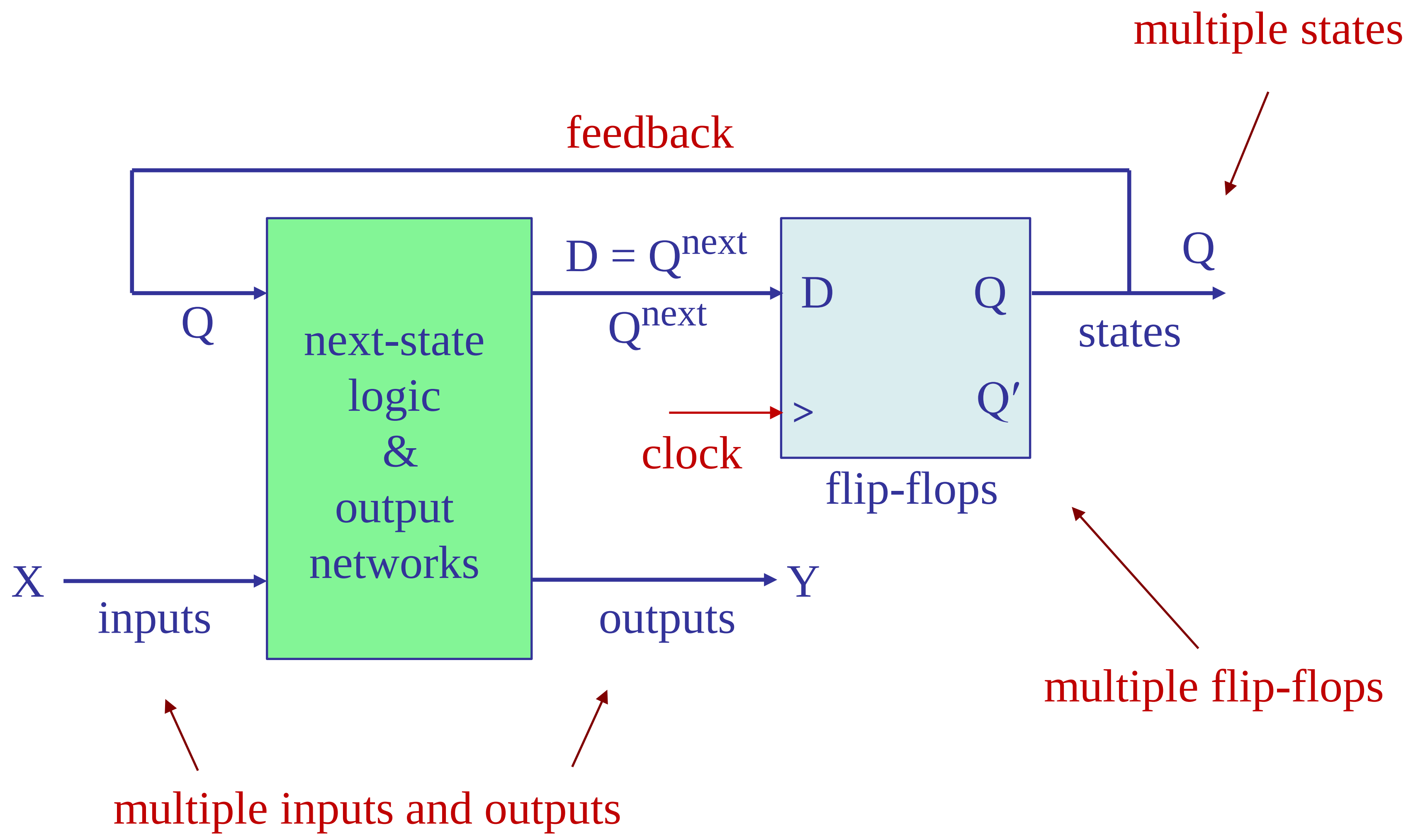
period = t_{per}
frequency = 1 / t_{per}
duty cycle = t_H / t_{per}

D flip-flops and State Machines

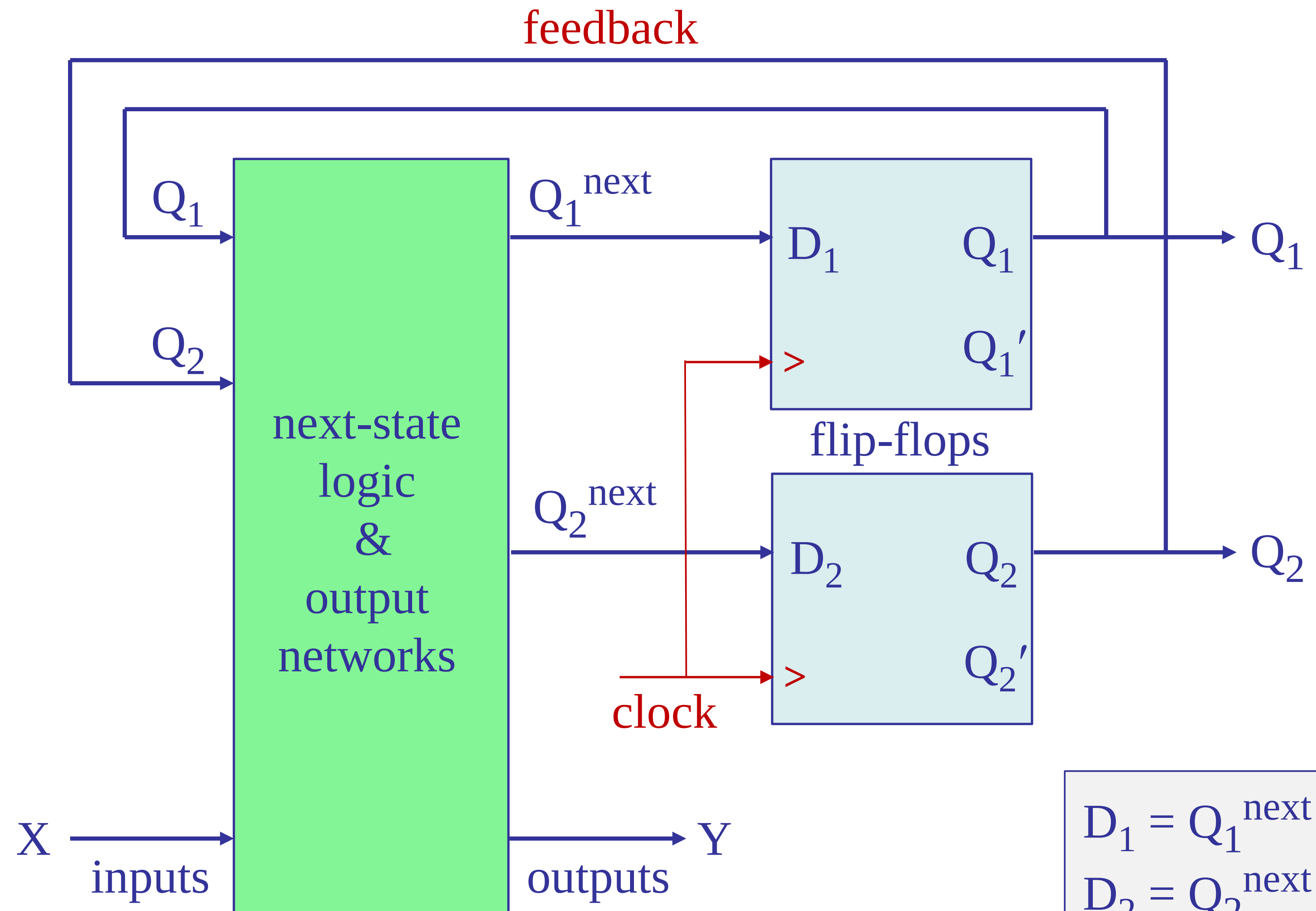
D flip-flops are widely used for the implementation of finite-state machines. Their advantage is that the next states, Q^{next}, are the excitation inputs to the flip-flops, i.e., $D = Q^{next}$. See next page for an alternative drawing.


feedback
D = Q^next
clock
D
Q
flip-flops
Q
present states
next-state logic & output networks
Q^next
X
inputs
Y
outputs
multiple flip-flops multiple states
multiple inputs and outputs

D flip-flops and State Machines



D flip-flops and State Machines – example with two states



$D_1 = Q_1^{next}$

$D_2 = Q_2^{next}$

other flip-flop types

SR flip-flops

T flip-flops

JK flip-flops

conversions between types ← explored in recitations

characteristic tables
characteristic equations
excitation tables
excitation equations

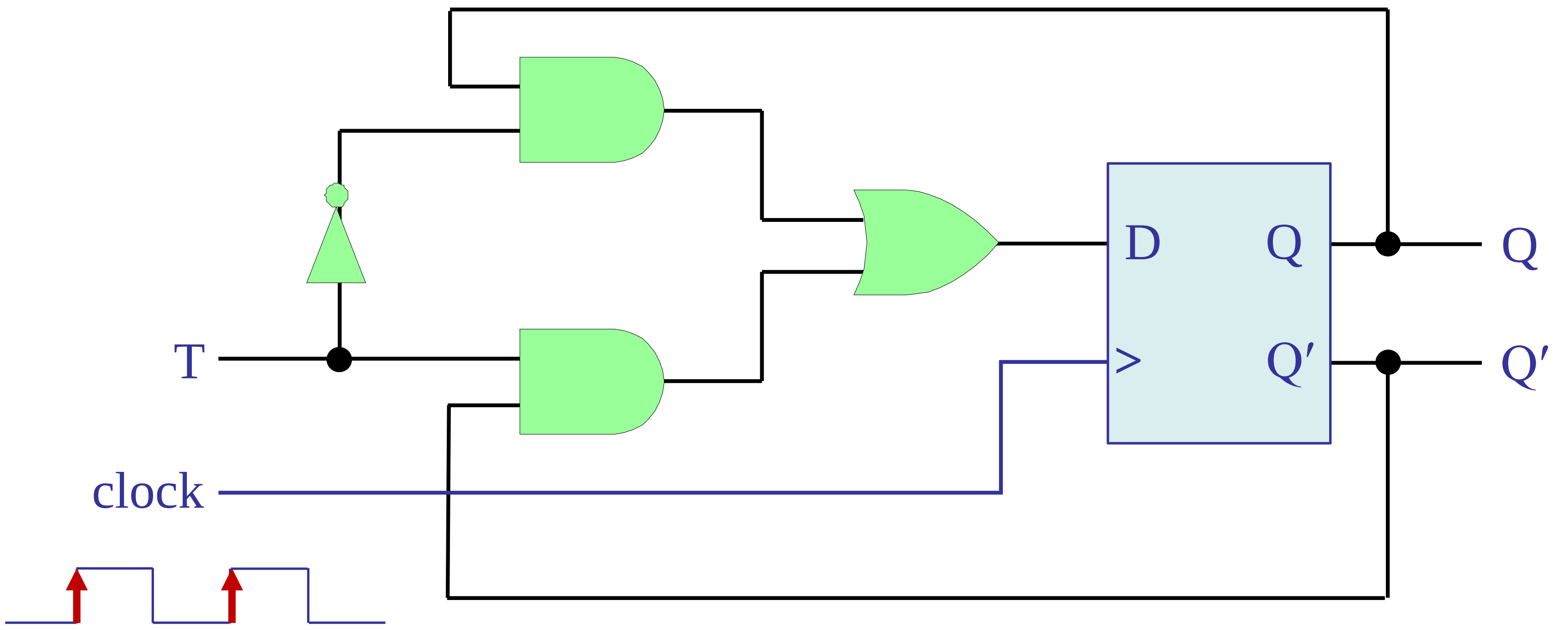
D to JK
JK to D

D to T
T to D

JK to T
T to JK

SR to JK
JK to SR

T flip-flop – constructed from a D flip-flop



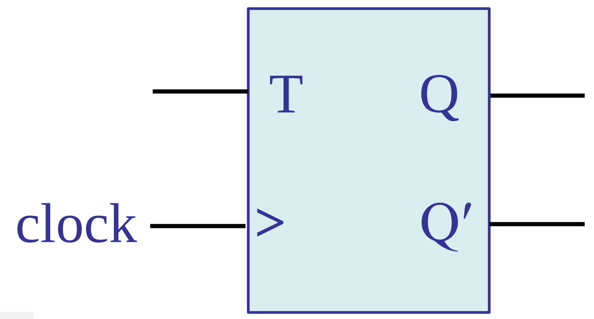
	T	E	Q_{next}
hold	0	↑	Q
toggle	1	↑	Q′

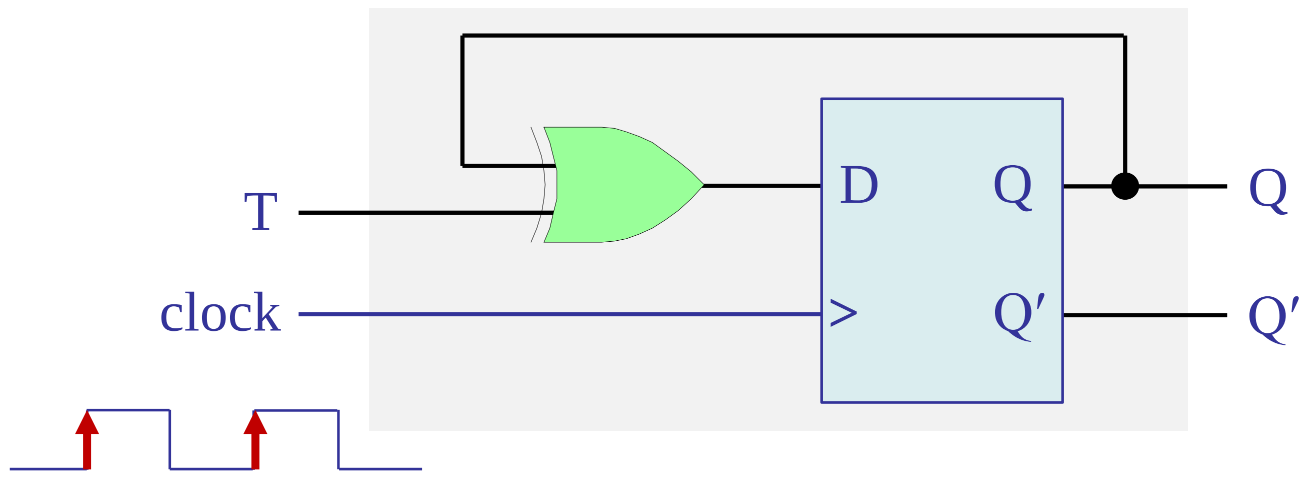
T flip-flop
characteristic equation

XOR

$$Q_{next} = D = T'\,Q + T\,Q' = T \oplus Q$$

T flip-flop – constructed from a D flip-flop





$$Q_{next} = D = \underline{T \oplus Q}$$

$$0 \oplus X = X$$
$$1 \oplus X = X'$$

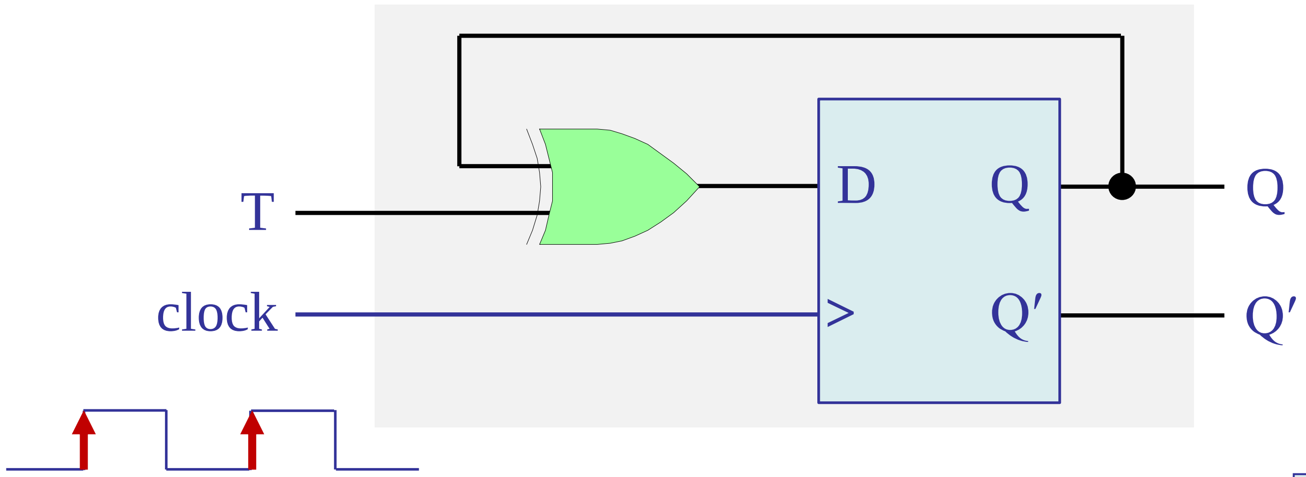
characteristic table

	T	E	Q_{next}
hold	0	↑	Q
toggle	1	↑	Q′

T flip-flop
characteristic equation

$$Q_{next} = D = T'Q + TQ' = T \oplus Q$$

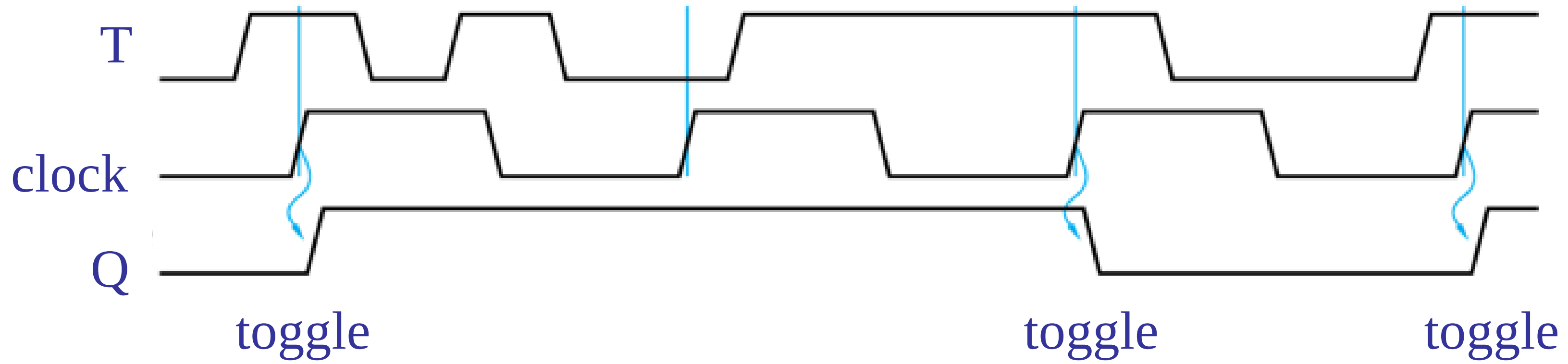
T flip-flop – constructed from a D flip-flop



$Q_{next} = D = \underline{T \oplus Q}$

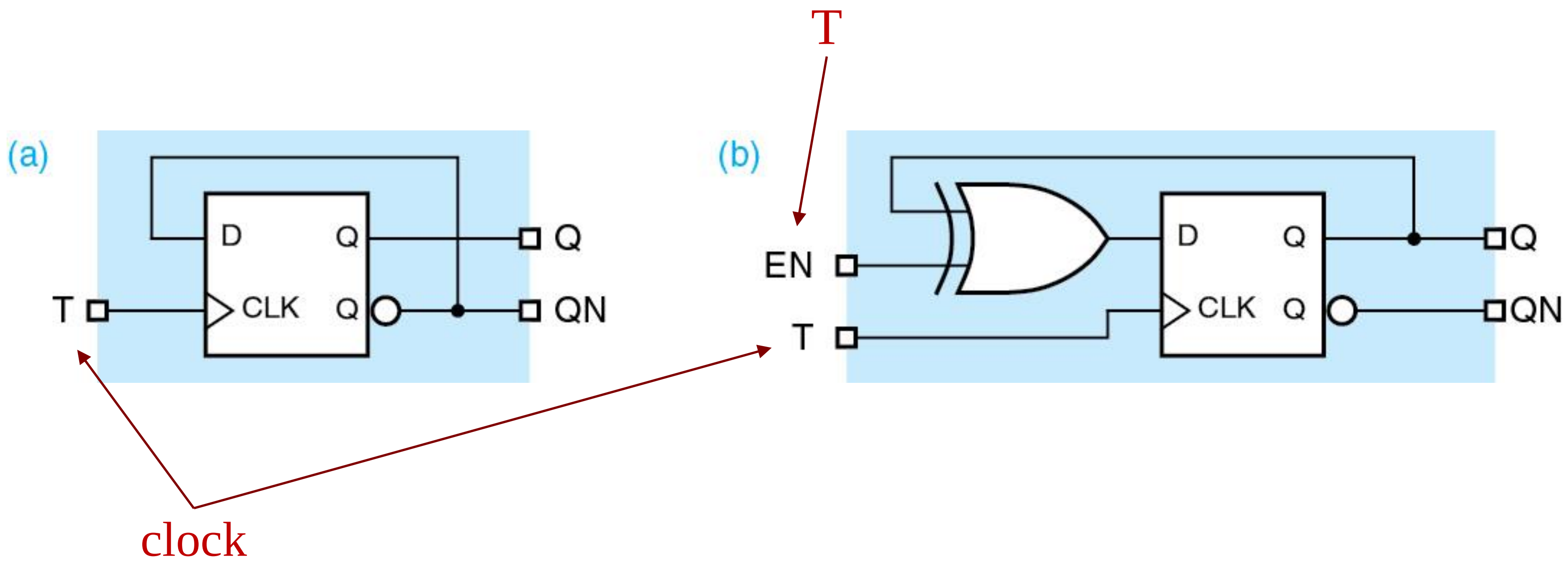
$0 \oplus X = X$

$1 \oplus X = X'$

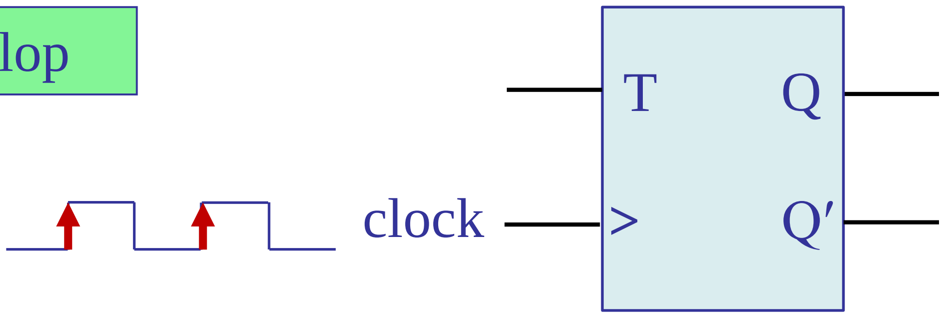


T flip-flop – constructed from a D flip-flop

Wakerly version



T flip-flop



characteristic table

	T	Q_{next}
hold	0	Q
toggle	1	Q′

excitation table

Q	Q_{next}	T
0	0	0
0	1	1
1	0	1
1	1	0

characteristic equation

$$Q_{next} = T' Q + T Q' = T \oplus Q$$

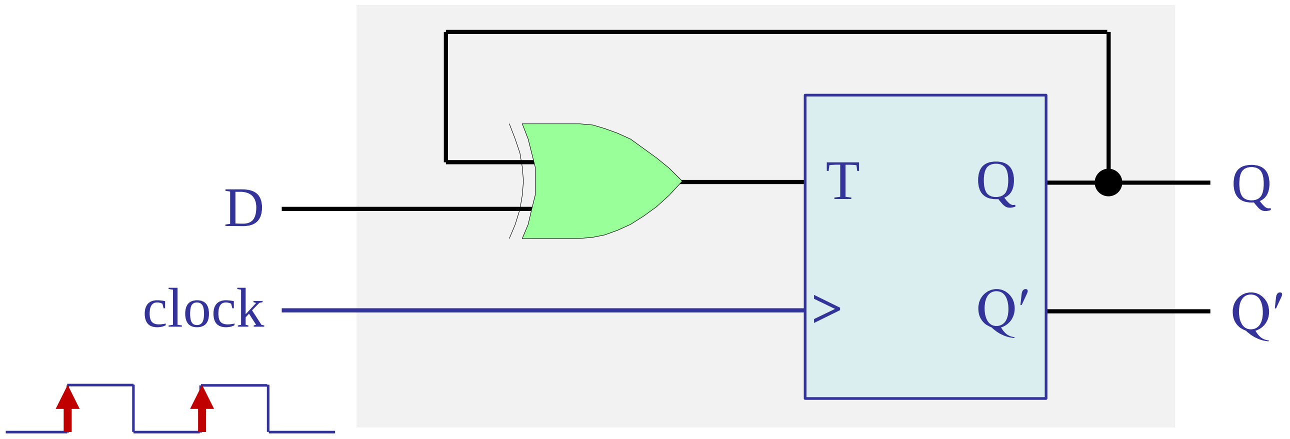
excitation equation

$$T = Q_{next} \oplus Q$$

excitation tables are useful because we usually know Q and what Q_{next} should be, and we need to determine the proper inputs to the flip-flops

T flip-flop

converting a T flip-flop to a D flip-flop

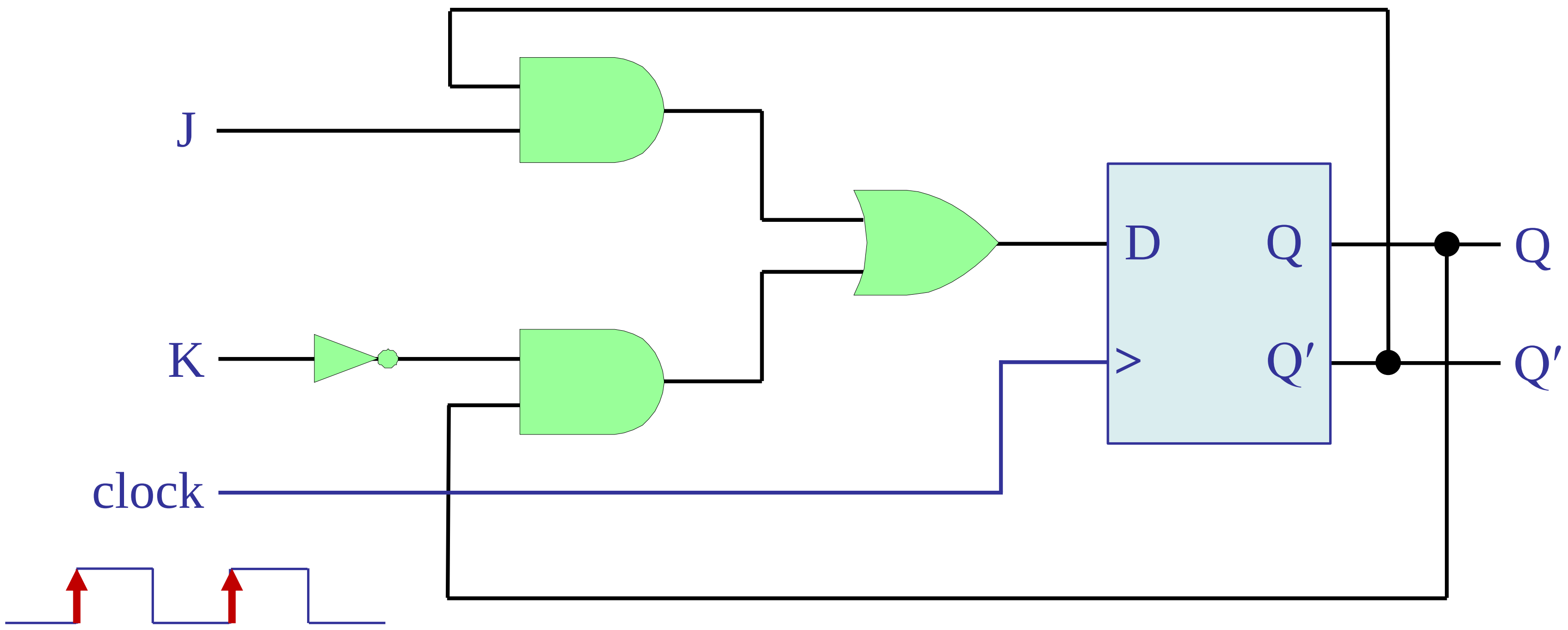


$T = D \oplus Q$

$Q_{next} = T \oplus Q = (D \oplus Q) \oplus Q = D$

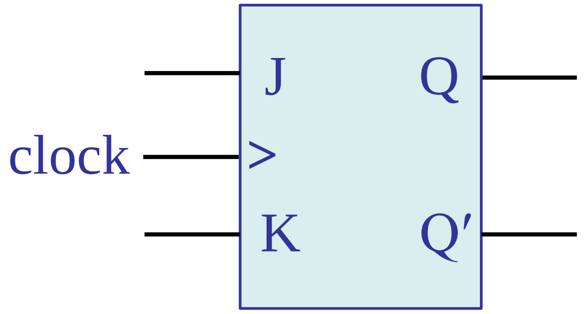
JK flip-flop

positive-edge-triggered JK flip-flop,
acts like an SR flip-flop, but toggling when S=R=1



JK flip-flop
characteristic equation

$$Q_{next} = D = J\,Q' + K'\,Q$$



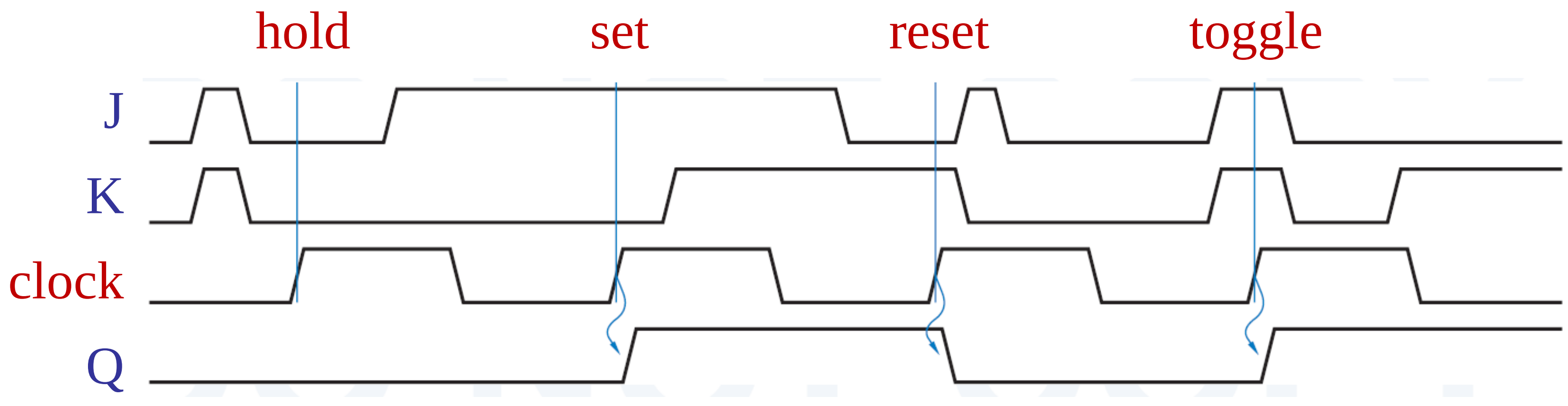
JK flip-flop

clock edge signal

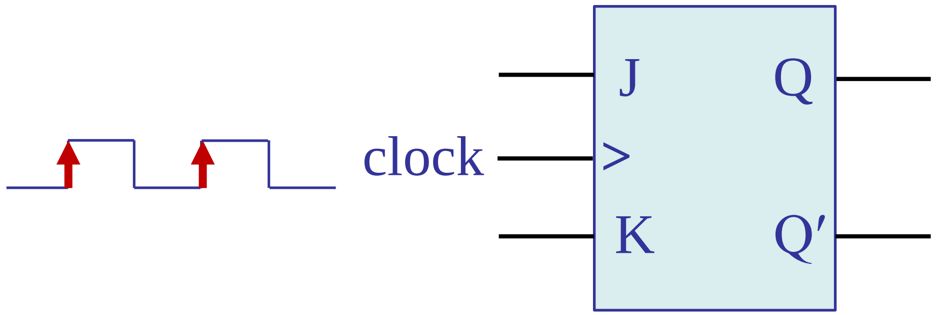
	J	K	E	Q_{next}
hold	x	x	$\not\uparrow$	Q
hold	0	0	$\uparrow$	Q
reset	0	1	$\uparrow$	0
set	1	0	$\uparrow$	1
toggle	1	1	$\uparrow$	Q′

JK flip-flop
characteristic equation

$$\boxed{Q_{next} = J\,Q' + K'\,Q}$$



JK flip-flop



characteristic table

J	K	Q_{next}
0	0	Q
0	1	0
1	0	1
1	1	Q′

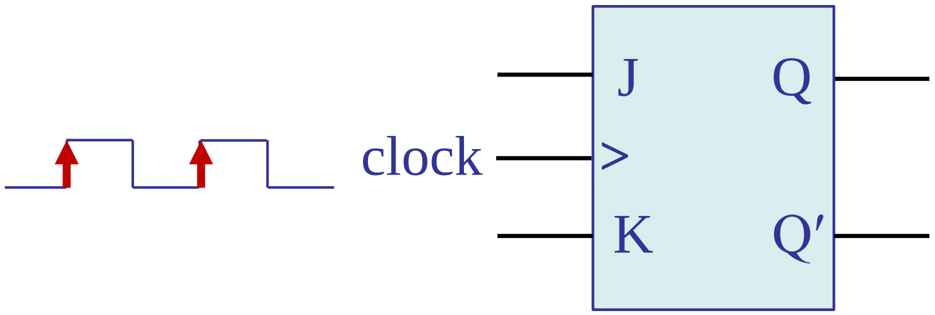
excitation table

Q	Q_{next}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

characteristic equation

$$Q_{next} = J\,Q' + K'\,Q$$

JK flip-flop



characteristic table

J	K	Q_{next}
0	0	Q
0	1	0
1	0	1
1	1	Q′

characteristic equation

$$Q_{next} = J\,Q' + K'\,Q$$

characteristic table

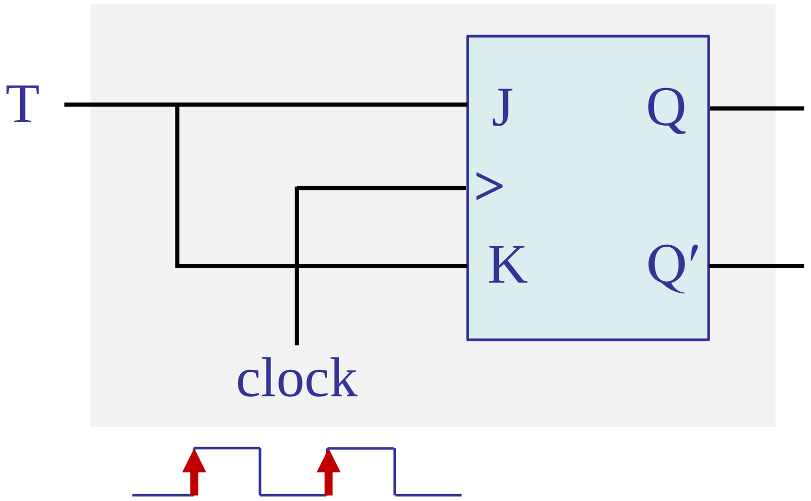
J	K	Q	Q_{next}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

excitation table

Q	Q_{next}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

JK flip-flop

converting a JK flip-flop to a T flip-flop



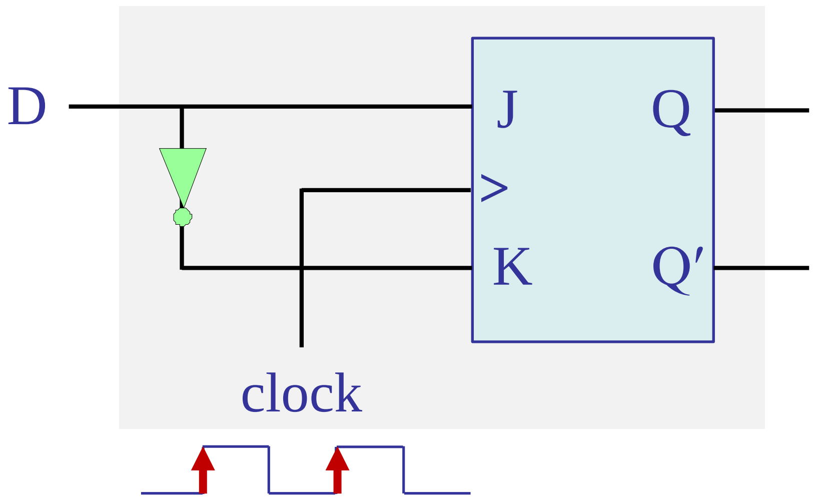
$J = K = T$

$Q_{next} = J\,Q' + K'\,Q$

$Q_{next} = T\,Q' + T'\,Q = T \oplus Q$

JK flip-flop

converting a JK flip-flop to a D flip-flop



$J = D$

$K = D'$

$Q_{next} = J\,Q' + K'\,Q = D\,Q' + D\,Q \;= D$

$Q_{next} = D$

$\longleftarrow$

excitation table

Q	Q_{next}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

flip-flop conversions - summary & web links

D to JK: $D = JQ' + K'Q$

JK to D: $J = D, \; K = D'$

D to T: $D = T \oplus Q$

T to D: $T = D \oplus Q$

JK to T: $J = K = T$

T to JK: $T = JQ' + KQ$

SR to JK: $S = JQ', \; R = KQ$

JK to SR: $J = S, \; K = R$

to be explored further in recitations

flip-flop conversions - part 1

flip-flop conversions - part 2

flip-flop conversions - part 3

flip-flop conversions - part 4

excitation tables

excitation tables are useful because we usually know what Q and Q_{next} should be, and wish to determine the proper inputs to the flip-flops

D flip-flop

Q	Q_{next}	D
0	0	0
0	1	1
1	0	1
1	1	0

T flip-flop

Q	Q_{next}	T
0	0	0
0	1	1
1	0	1
1	1	0

$Q_{next} = D$ ↑ characteristic equations ↓ $Q_{next} = S + R'Q$

$Q_{next} = T \oplus Q$ ↑ characteristic equations ↓ $Q_{next} = JQ' + K'Q$

SR flip-flop

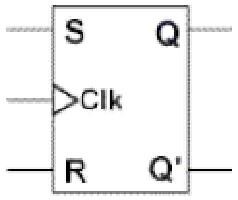
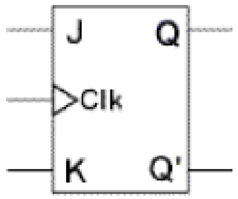
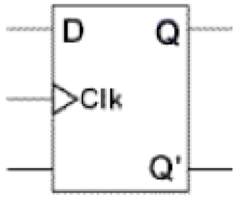
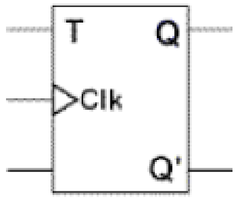
Q	Q_{next}	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

JK flip-flop

Q	Q_{next}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

flip-flops - summary

Reference: A .F. Kana, DLD lectures (on Canvas)

TYPE	FLIP-FLOP SYMBOL	CHARACTERISTIC TABLE			CHARACTERISTIC EQUATION	EXCITATION TABLE			
SR		S	R	Q(next)	$Q(next) = S + R'Q$	Q	Q(next)	S	R
		0	0	Q	$SR = 0$	0	0	0	X
		0	1	0		0	1	1	0
		1	0	1		1	0	0	1
		1	1	?		1	1	X	0
JK		J	K	Q(next)	$Q(next) = JQ' + K'Q$	Q	Q(next)	J	K
		0	0	Q		0	0	0	X
		0	1	0		0	1	1	X
		1	0	1		1	0	X	1
		1	1	Q′		1	1	X	0
D		D	Q(next)		$Q(next) = D$	Q	Q(next)	D	
		0	0			0	0	0	
		1	1			0	1	1	
						1	0	0	
						1	1	1	
T		T	Q(next)		$Q(next) = TQ' + T'Q$	Q	Q(next)	T	
		0	Q			0	0	0	
		1	Q′			0	1	1	
						1	0	1	
						1	1	0	