BEE 271 Digital circuits and systems

Spring 2017

Lecture 10: Sequential circuits

Nicole Hamilton

https://faculty.washington.edu/kd1uj

Topics

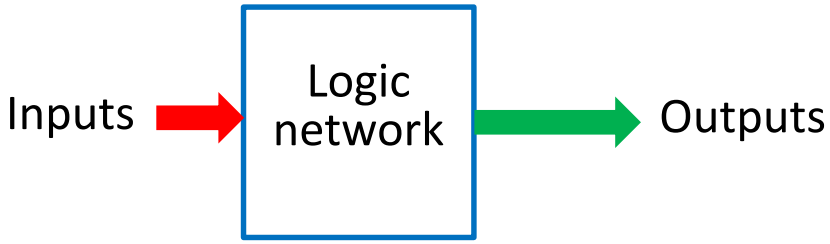
1. Introduction to sequential circuits

Chapter 5

Flip-Flops, Registers, and Counters

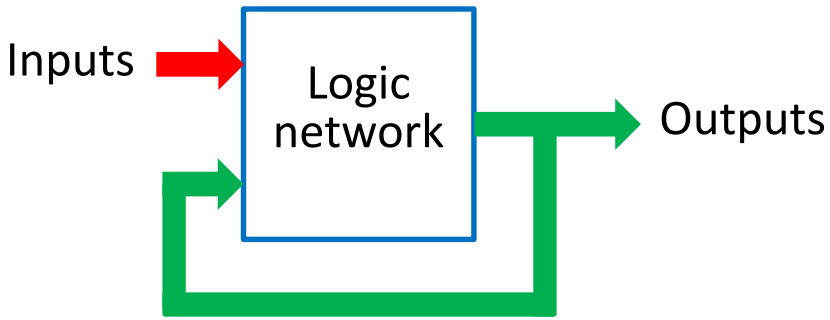
Combinatorial vs. Sequential Logic

Combinatorial or combinational logic



1. No memory elements.
2. No feedback from the outputs to the inputs.
3. Outputs depend only on the inputs.

Sequential logic



1. Contains memory that can remember a present state.
2. Outputs feed back to the inputs.
3. Outputs depend on both inputs and the present state.

Two problems

1. Would like to save state, e.g., count things.
2. Need to wait until signals settle

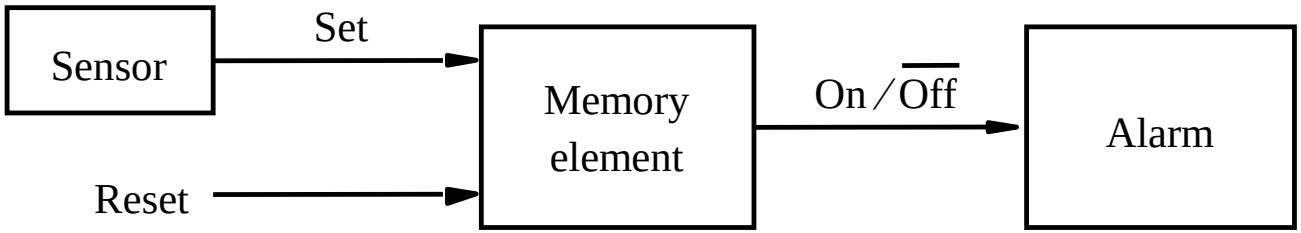
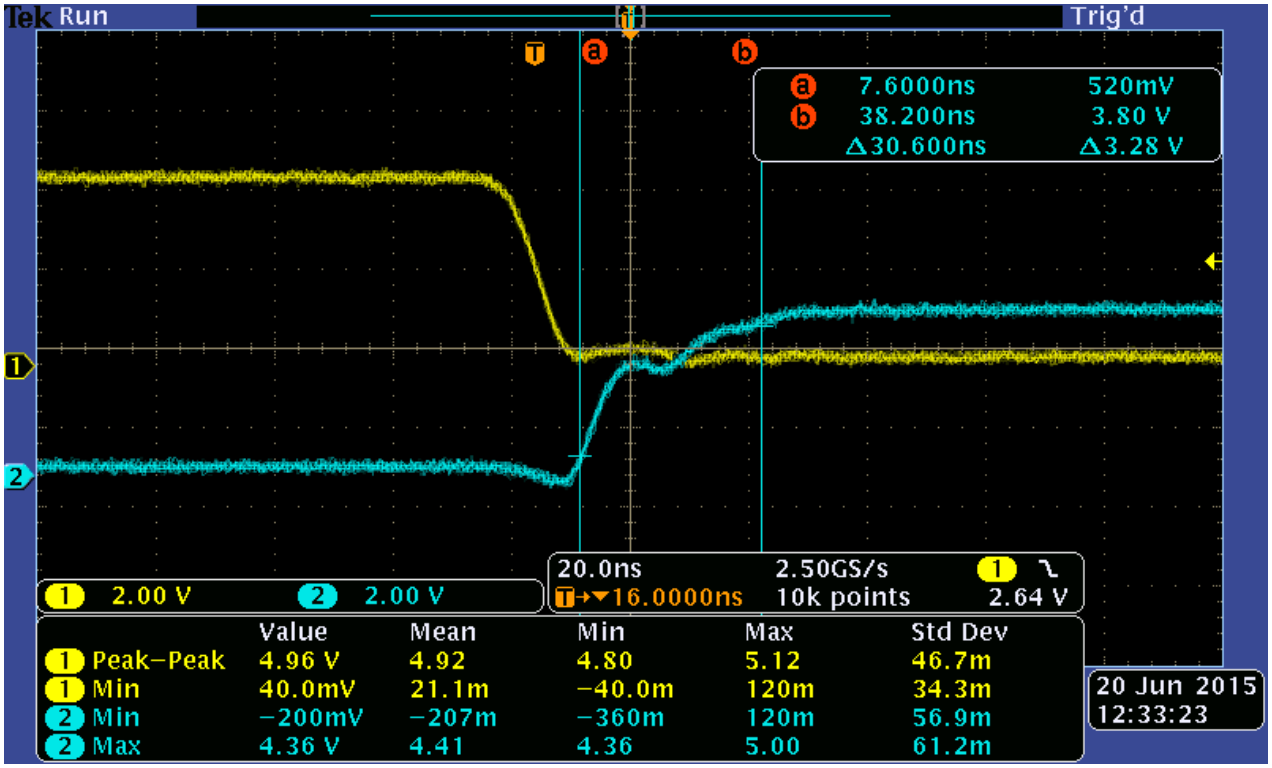


Figure 5.1. Control of an alarm system.

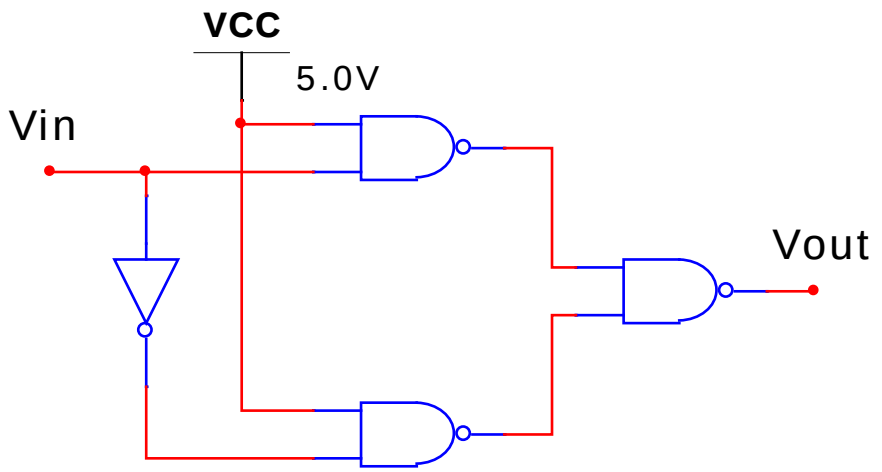
Real gates have propagation and rise and fall times.

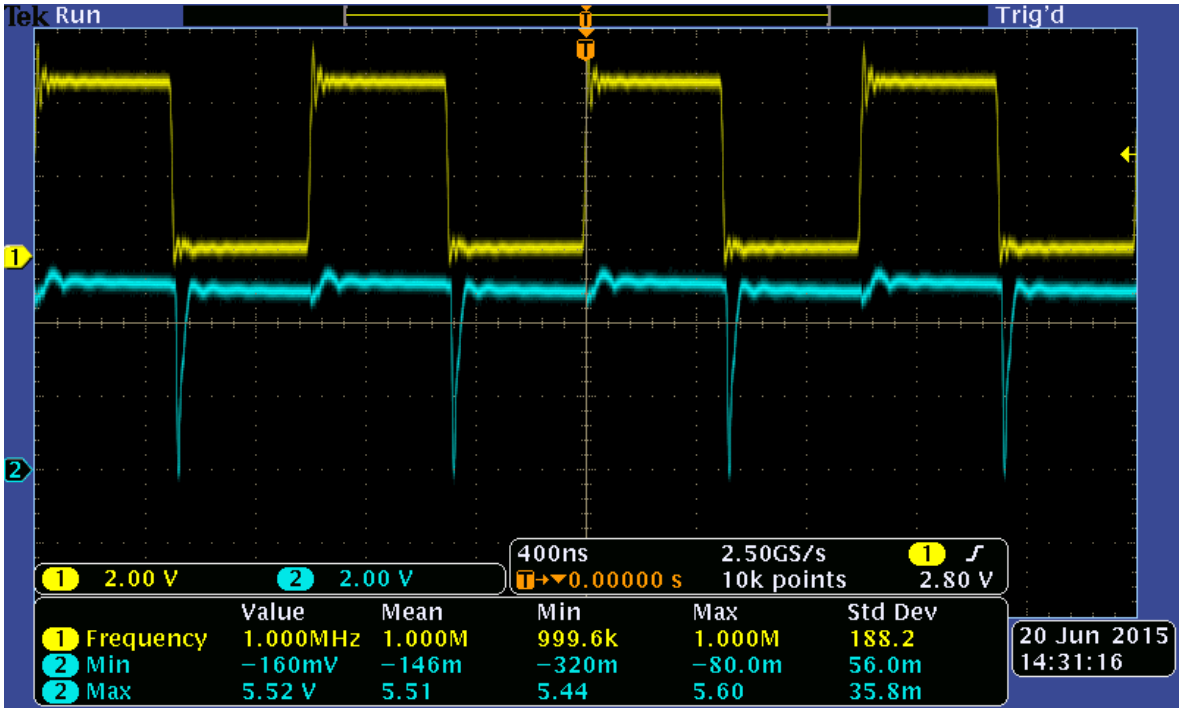
Causes hazards where outputs change when they shouldn't.

Must wait until signals have settled.



A circuit with a hazard in lab 1.

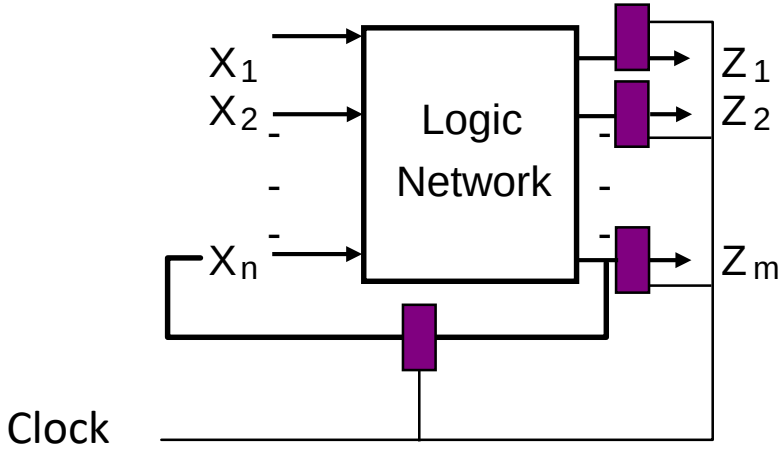


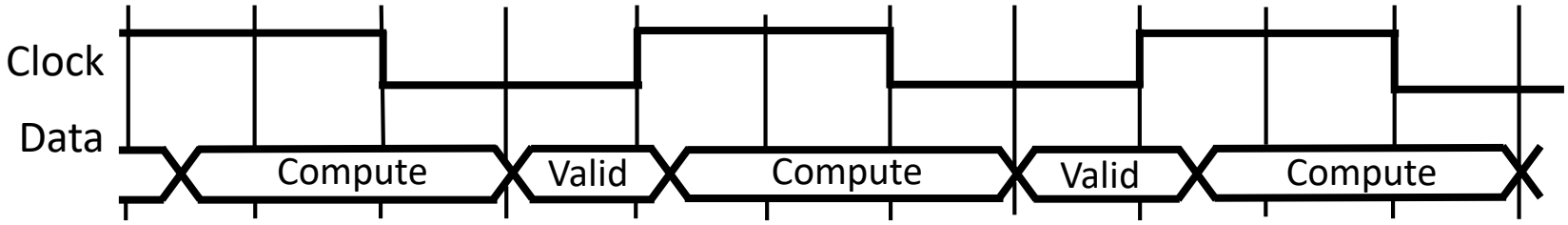


Safe Sequential Circuits

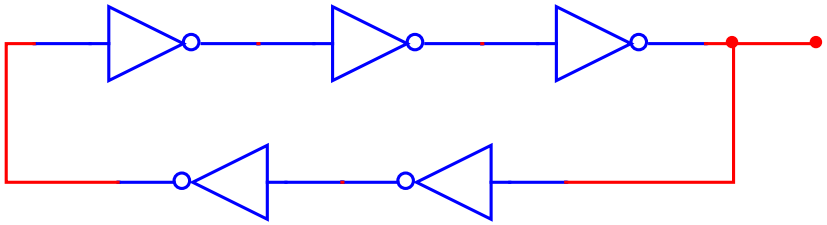
Clocked elements on feedback, perhaps outputs

- Clock signal synchronizes operation
- Clocked elements hide glitches/hazards

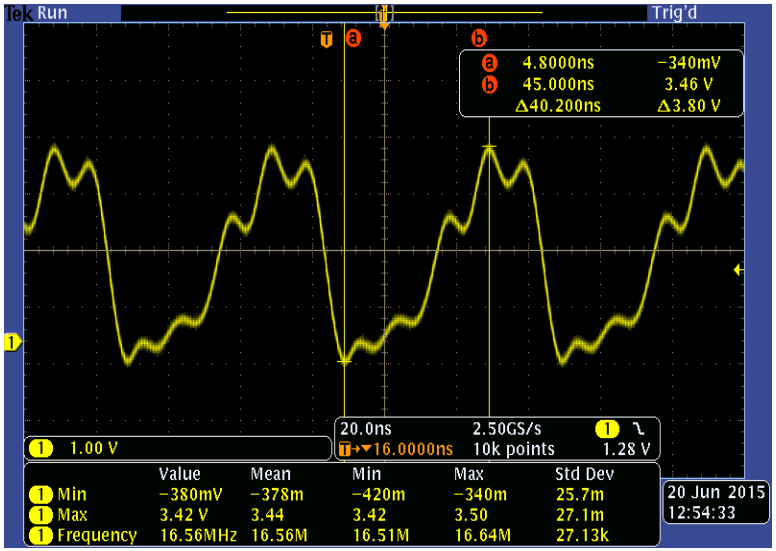


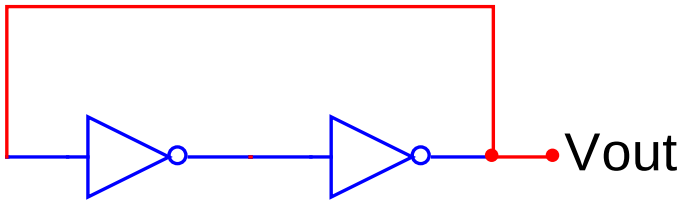


Feedback



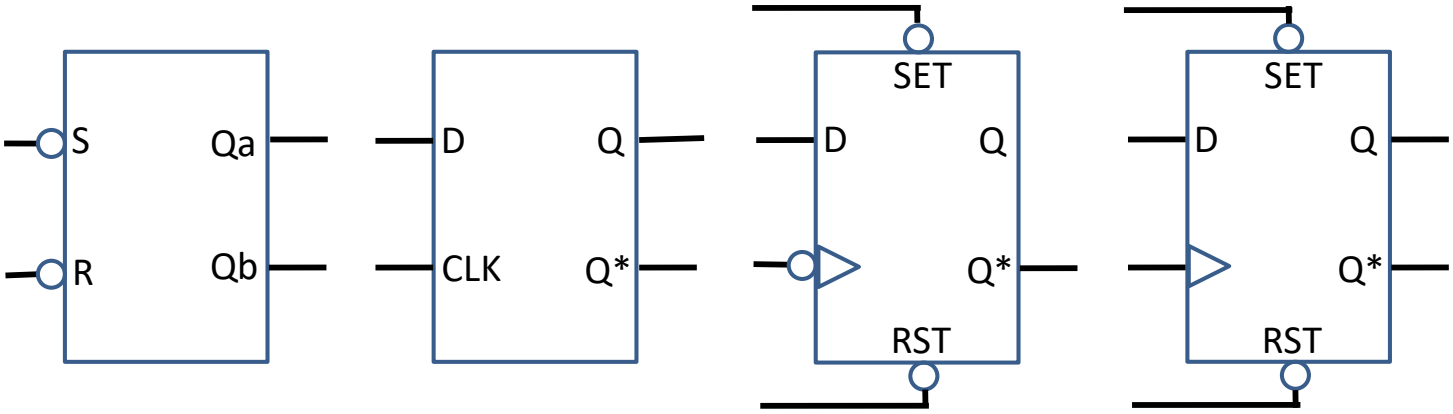
Ring oscillator with an odd number of inverters





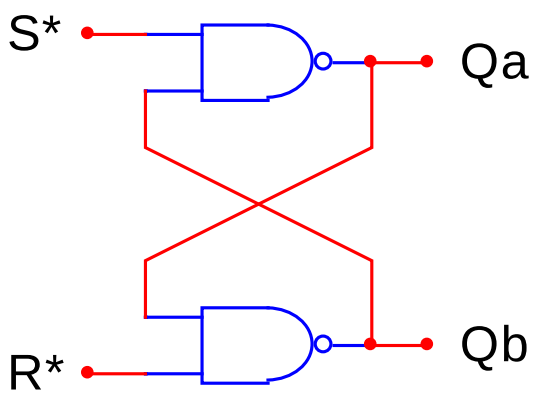
Bistable latch with an even number of inverters

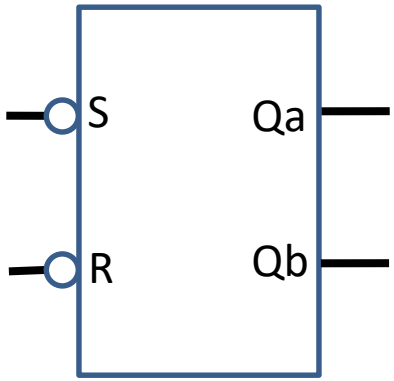
Vout can be either 1 or 0 but whatever it is will be stable.



Latch Gated Latch Master-slave Edge-triggered

Set/reset latches





S'R' latch

S*	R*	Qa	Qb
0	0	1	1
0	1	1	0
1	1	no change	no change
1	0	0	1

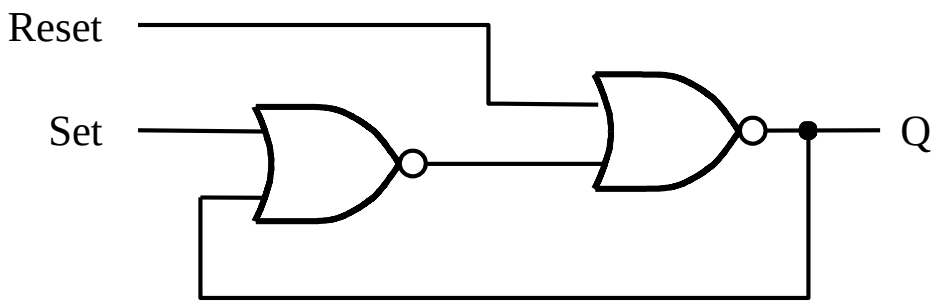


Figure 5.3. A memory element with NOR gates.

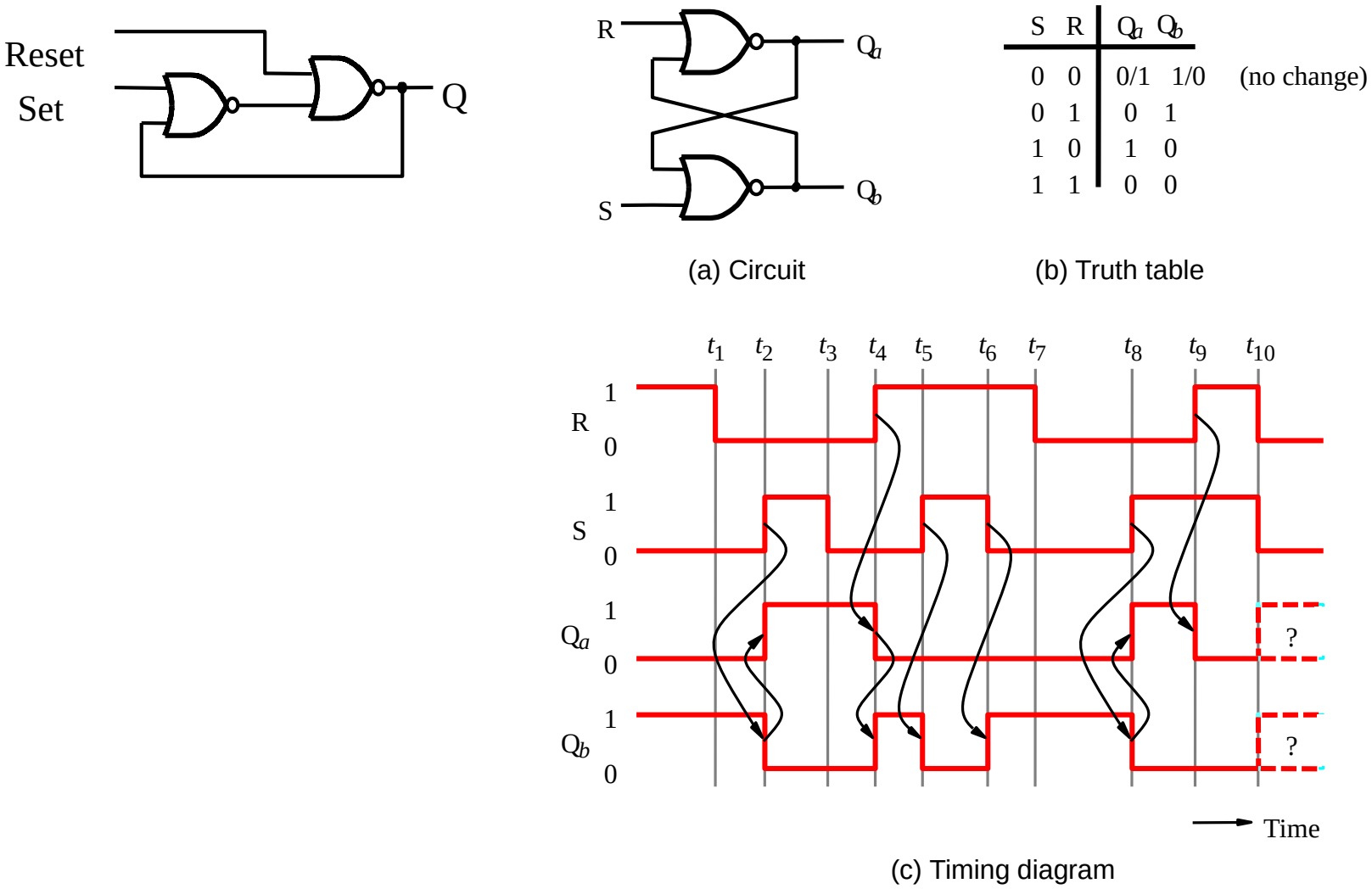
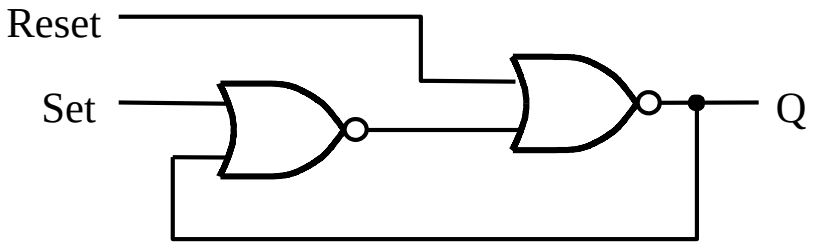
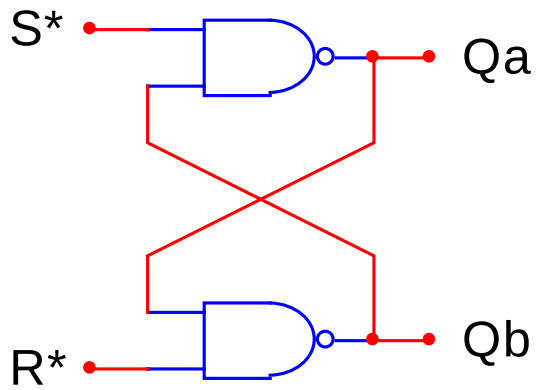


Figure 5.4. A basic latch built with NOR gates.

```
module SetResetLatch( input set, reset,
      output Q, Qn );

   nor ( Qn, set,   Q  );
   nor ( Q,  reset, Qn );

endmodule
```




```
module SetnResetnLatch( input setn, resetn,
      output Q, Qn );

   nand ( Q,  setn,   Qn );
   nand ( Qn, resetn, Q  );

endmodule
```



```
module SRLatch( input s, r, output reg Q );

   always @( * )
      casex ( { s, r } )
         'b1x: Q = 1;
         'b01: Q = 0;
      endcase

endmodule
```

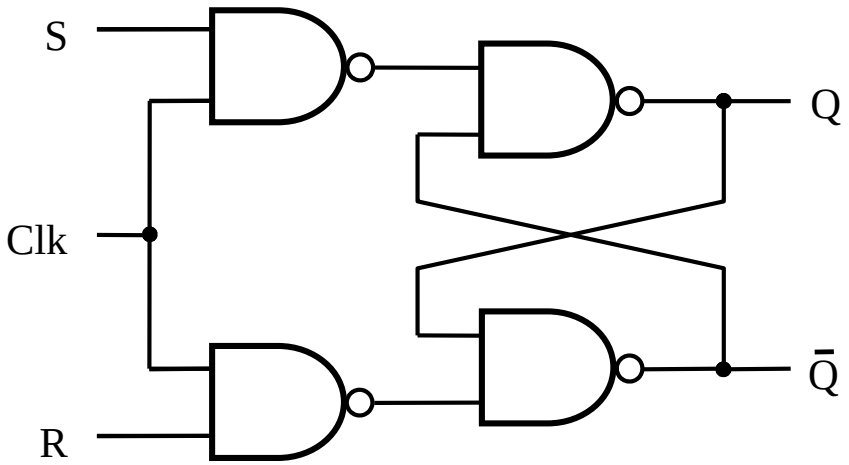
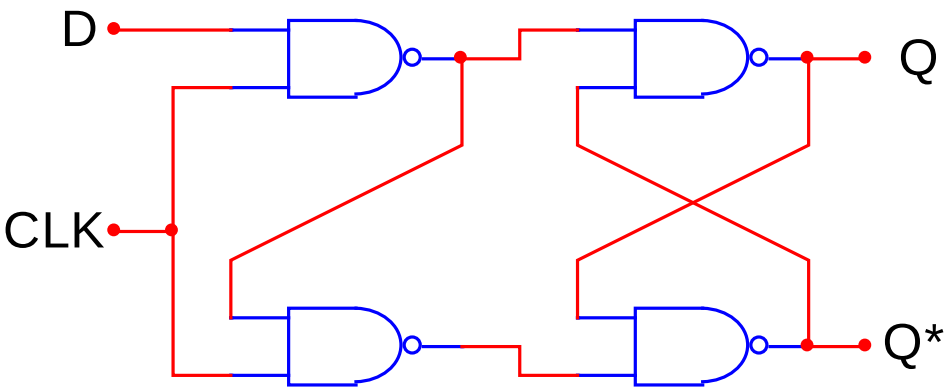
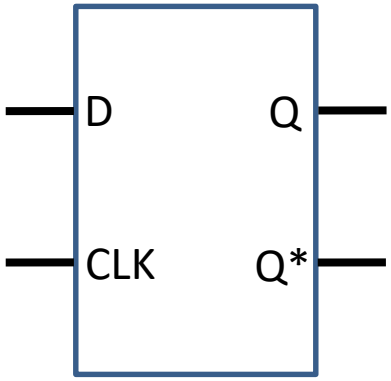


Figure 5.6. Gated SR latch with NAND gates.

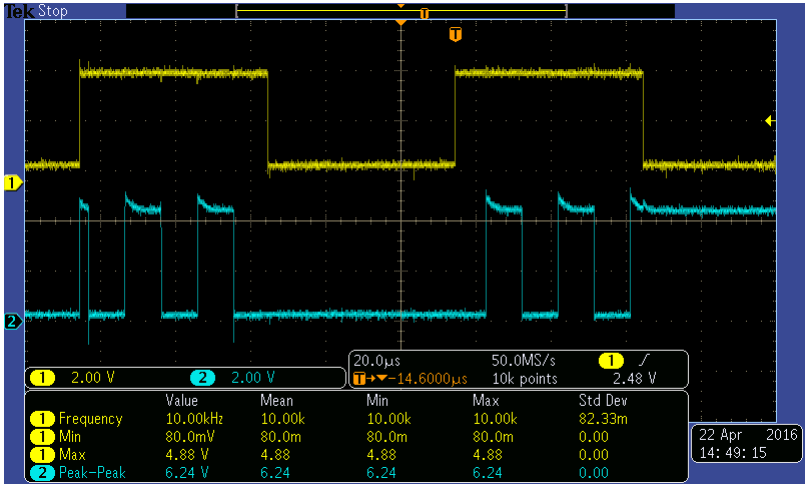
Gated latches





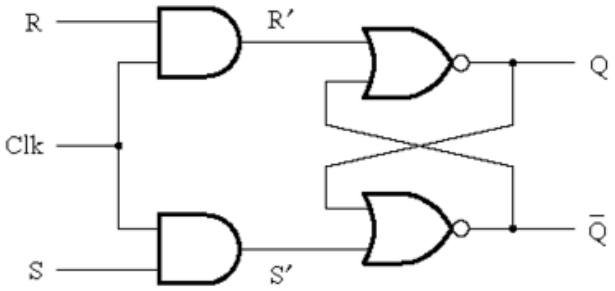
Gated D latch

Clock used to sample the input when the clock is high and hold it when the clock is low.



clock = 10 KHz / D = 52 KHz

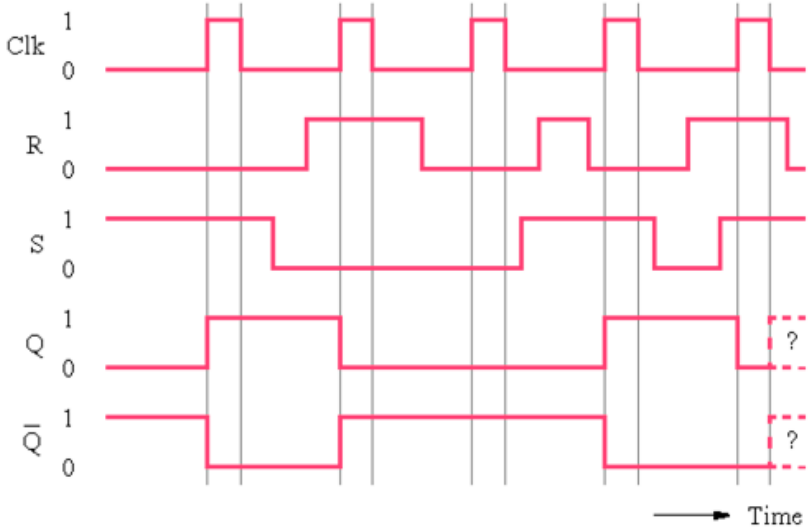
```
module DLatch1( input clock, D, output reg Q );
   always @( * )
      if ( clock )
         Q = D;     // Implied memory
endmodule


module DLatch2( input clock, D, output reg Q );
   always @( * )
      Q = clock ? D : Q;
endmodule
```



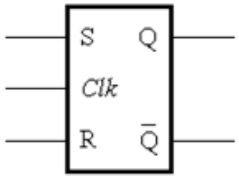
(a) Circuit

Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x

(b) Characteristic table

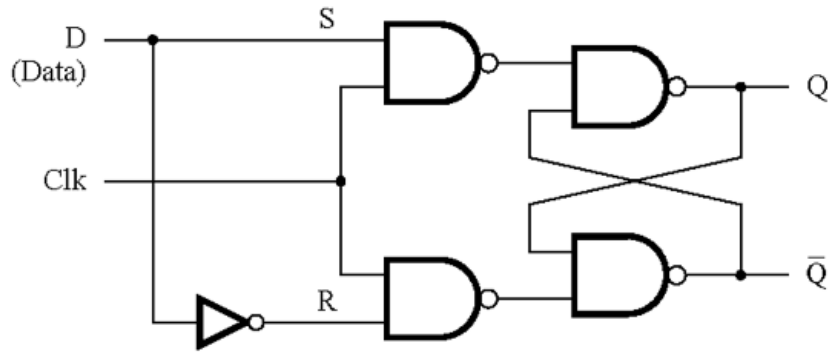


(c) Timing diagram



(d) Graphical symbol

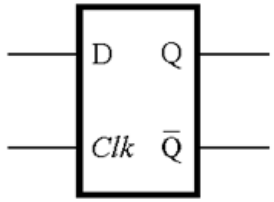
Figure 5.5.
Gated SR latch.



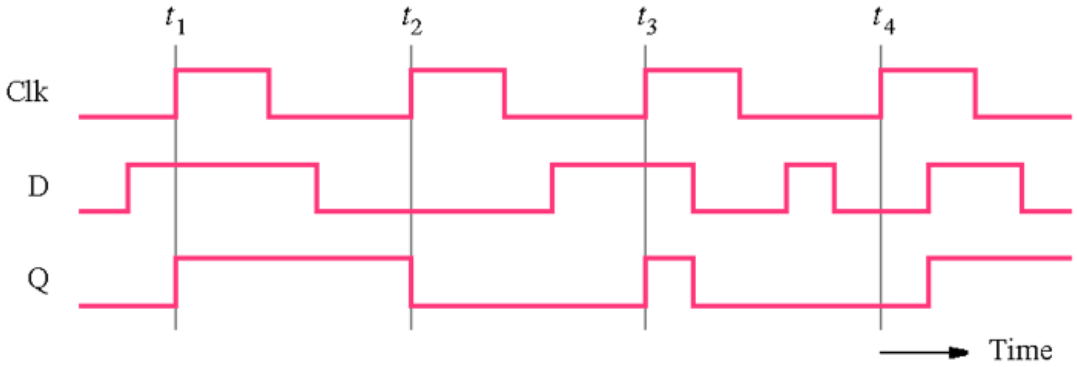
(a) Circuit

Clk	D	Q(t + 1)
0	x	Q(t)
1	0	0
1	1	1

(b) Characteristic table



(c) Graphical symbol



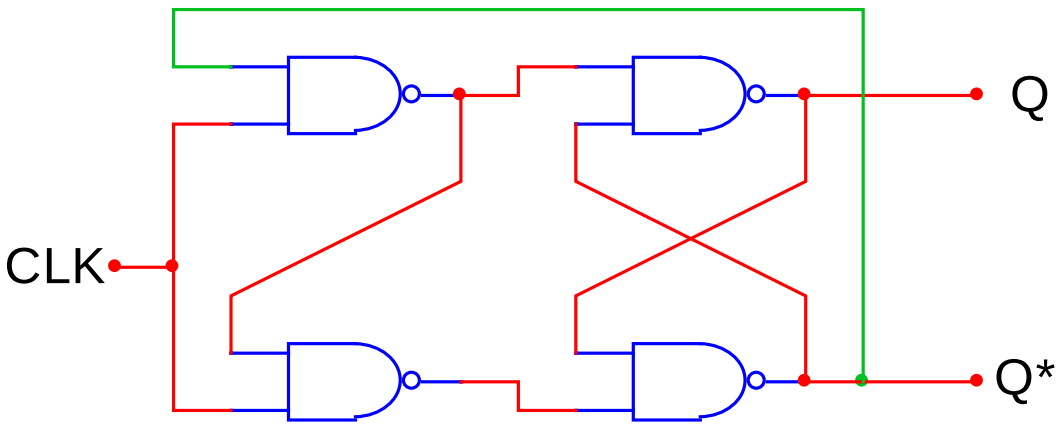
(d) Timing diagram

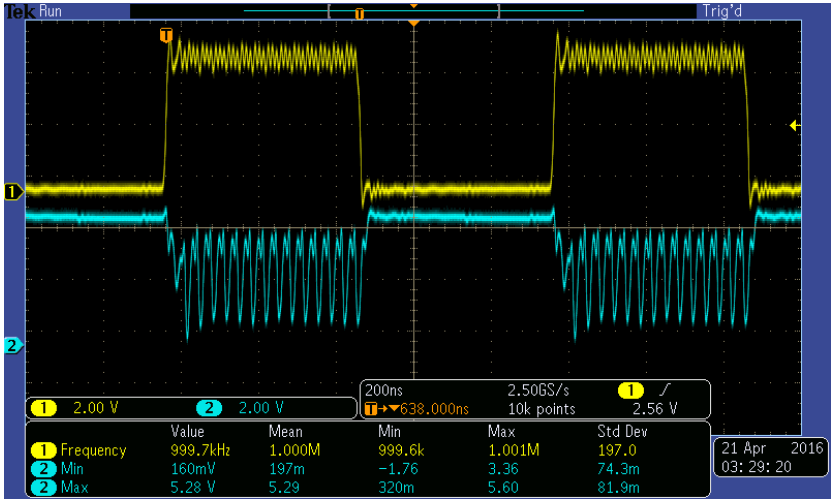
Figure 5.7. Gated D latch.

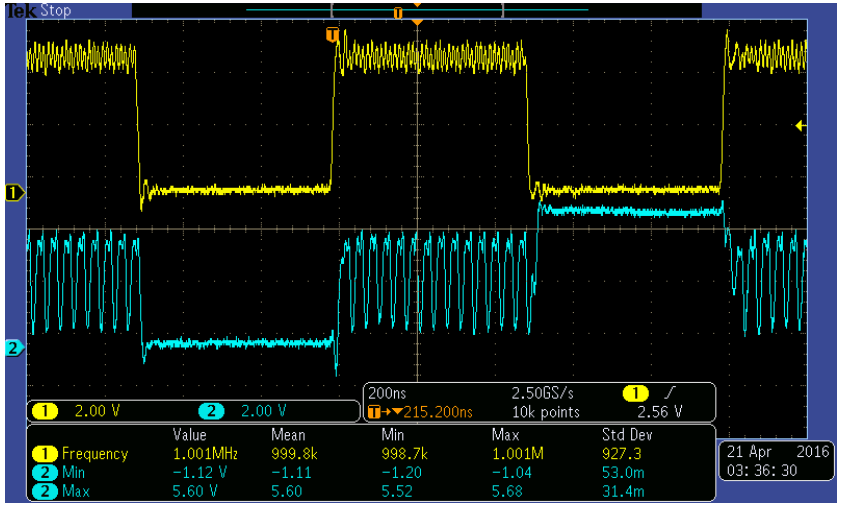
Flip Flops

- Problem: latches are sensitive to any changes that occur with the input while the clock or control signal is high
  - Glitches/Hazards
  - Unsynchronized changes
- Solution: use flip-flops, devices that react only on the clock edge

But we cannot build reliable counters or anything else requiring feedback with latches.







What you get depends on small differences in the length of the green wire.

Solution is to isolate inputs from outputs

Three popular alternatives:

1. Two-phase clocking
2. Master-slave
3. Edge-triggered

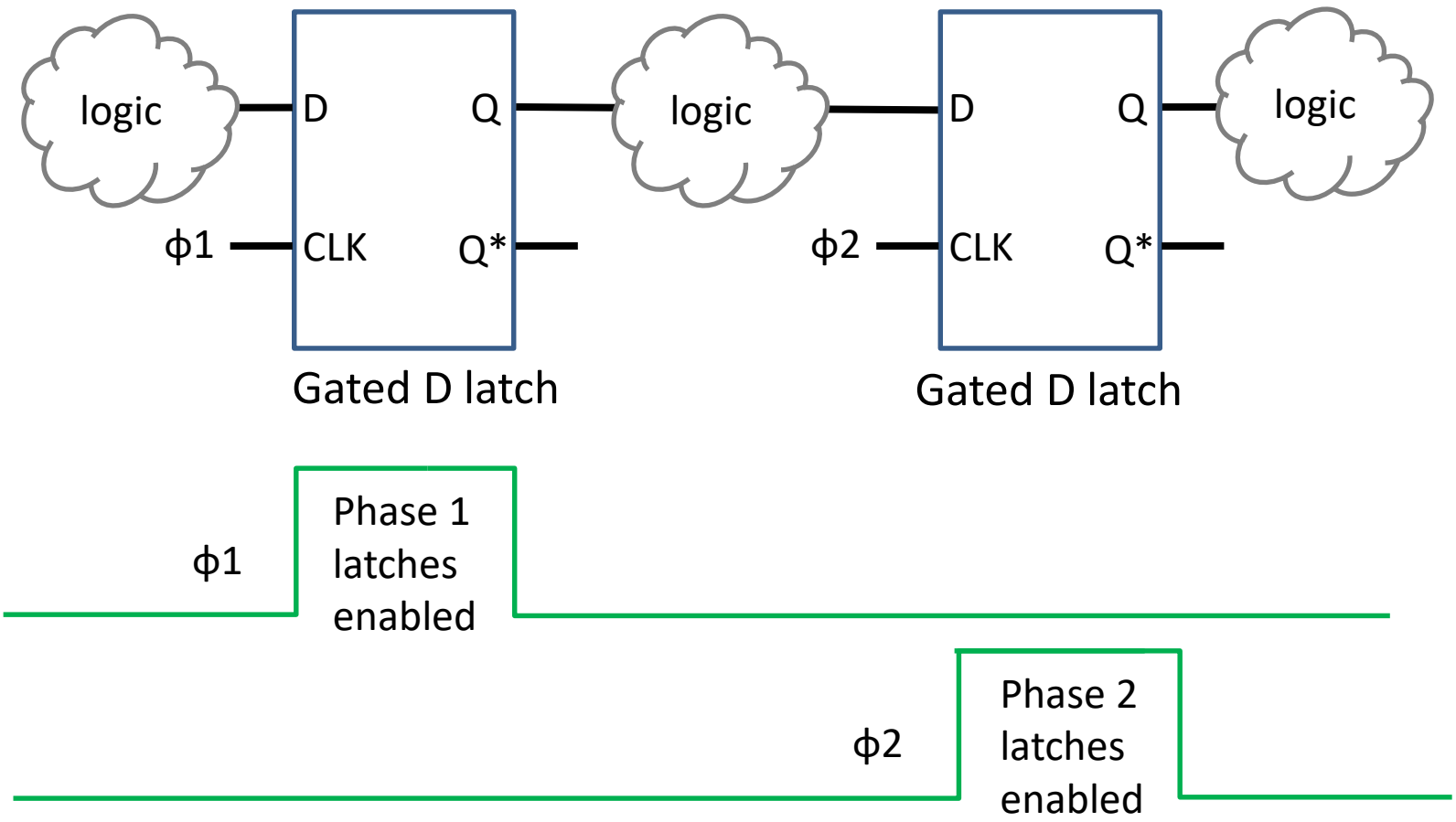
Terminology

Basic latch: A feedback connection of two NORs or two NANDs that can store 1 bit. Set using the S input and reset using the R input.

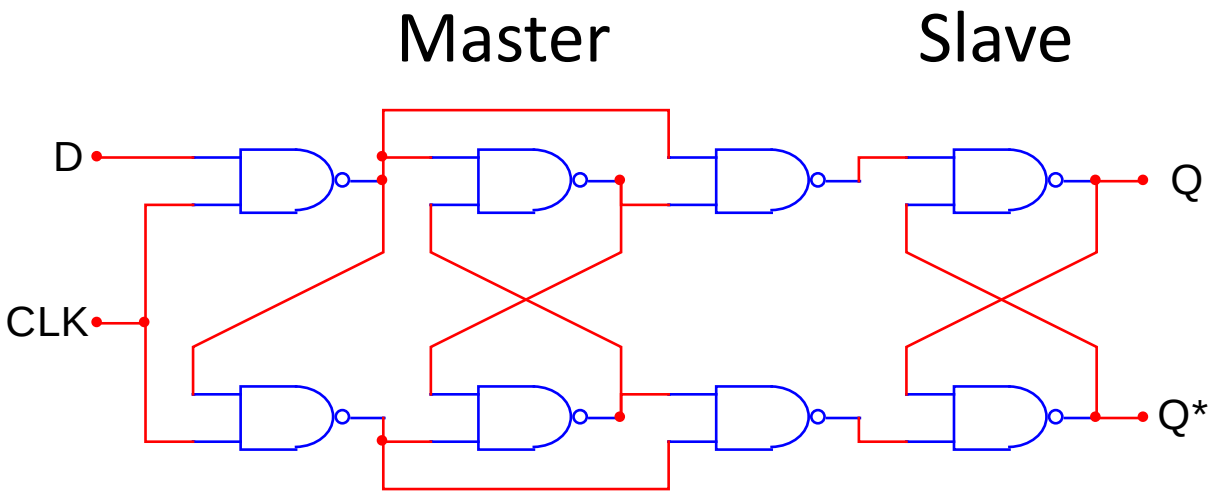
Gated latch: A basic latch that includes input gating and a control input signal. Retains its value when the control signal is 0, changes state when the control signal is 1.

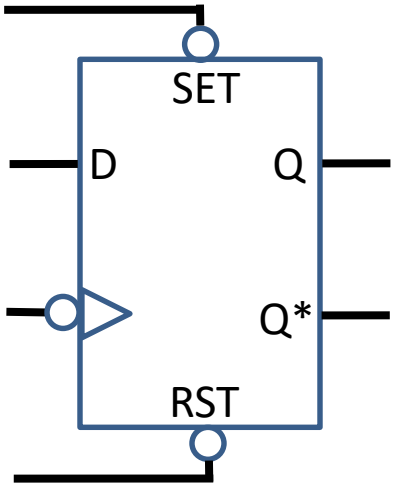
Flip-flop: A storage element whose output changes only on the edge of a controlling clock. Can be either positive or negative edge-triggered.

Two-phase clock

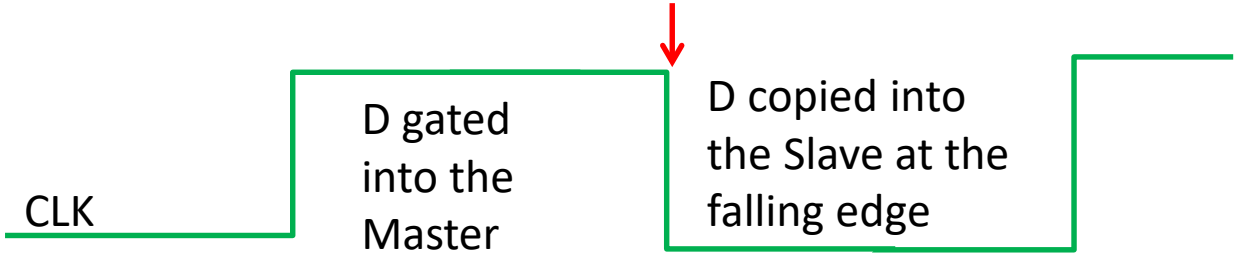


Non-overlapping clocks for phases 1 and 2. Advantages are fewer gates per bit and that you can put logic between both phases so long phase 1 latches only use phase 2 inputs and vice versa.



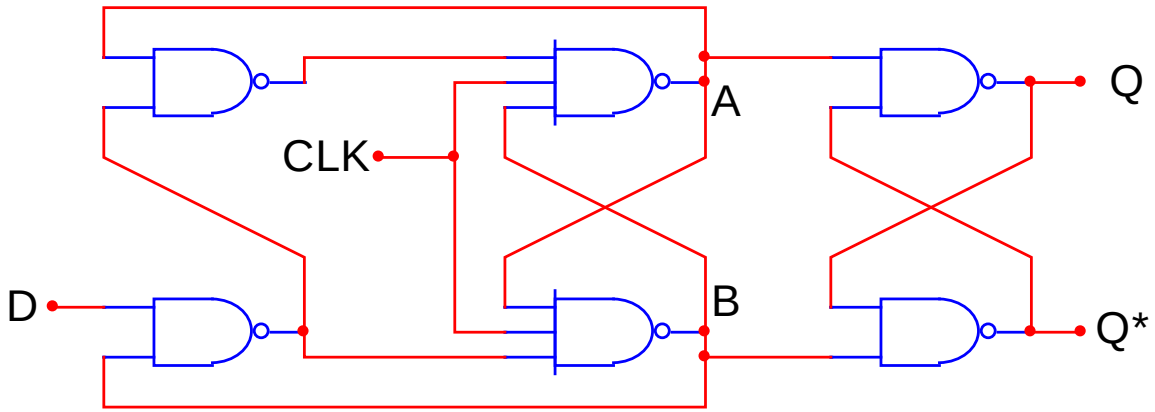


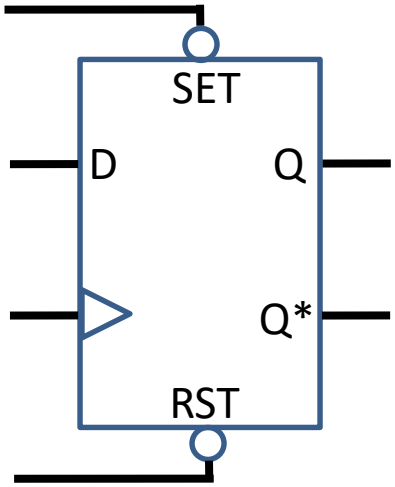
Master-slave
D flip-flop



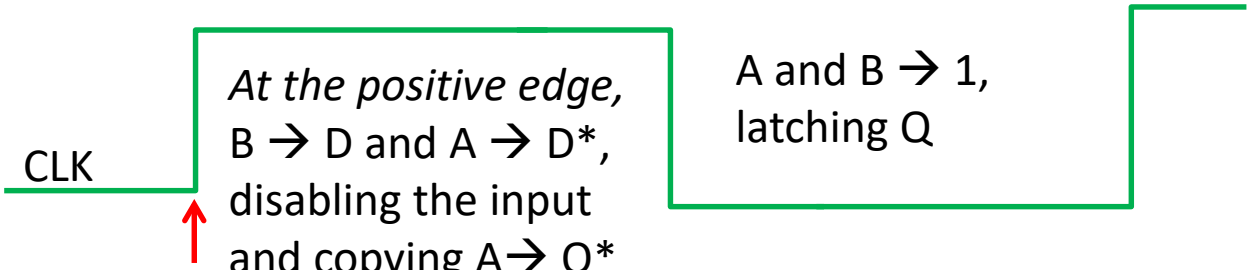
Because of the signal path is longer to the slave, we can guarantee the input to the master will be disabled before the input to the slave is enabled.

Edge-triggered





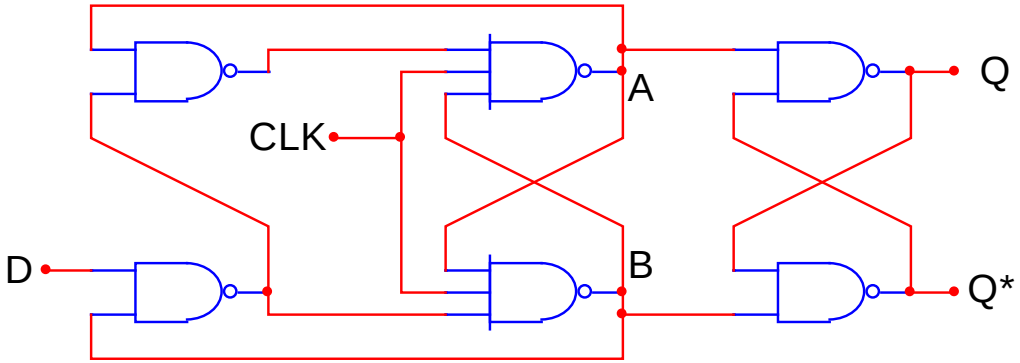
Edge-triggered
D flip-flop

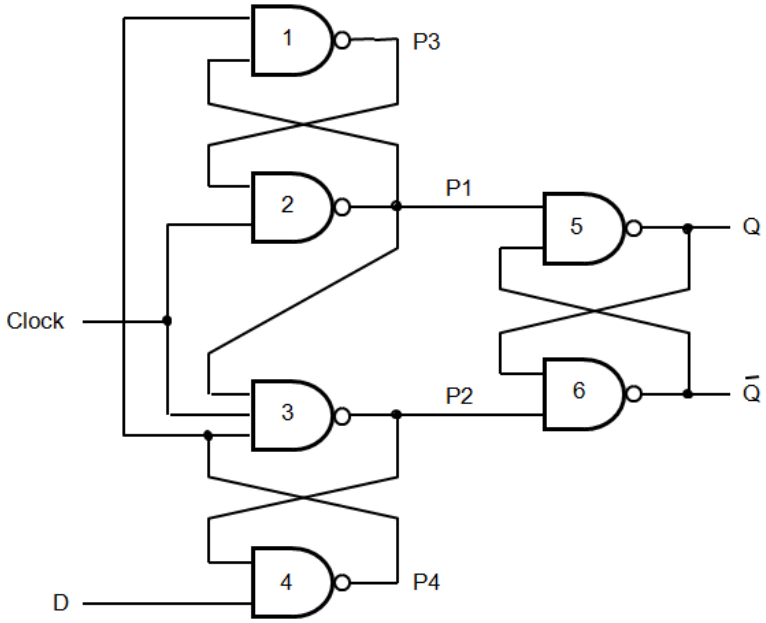


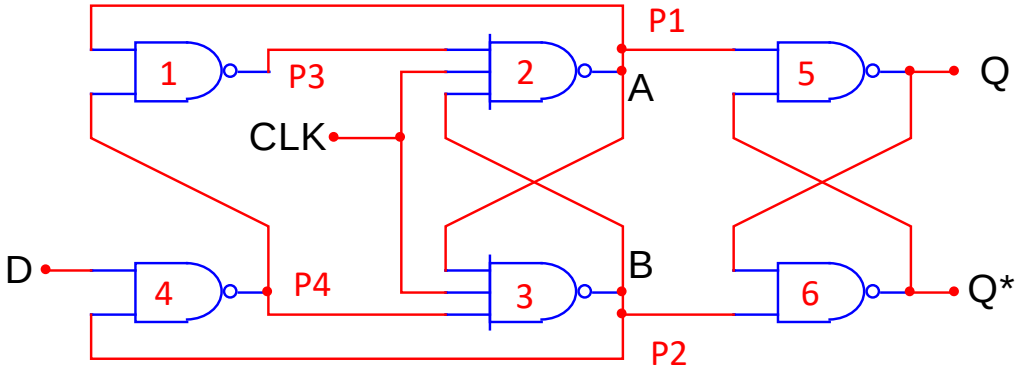
At the positive edge,
B → D and A → D*,
disabling the input
and copying A→ Q*
and B → Q

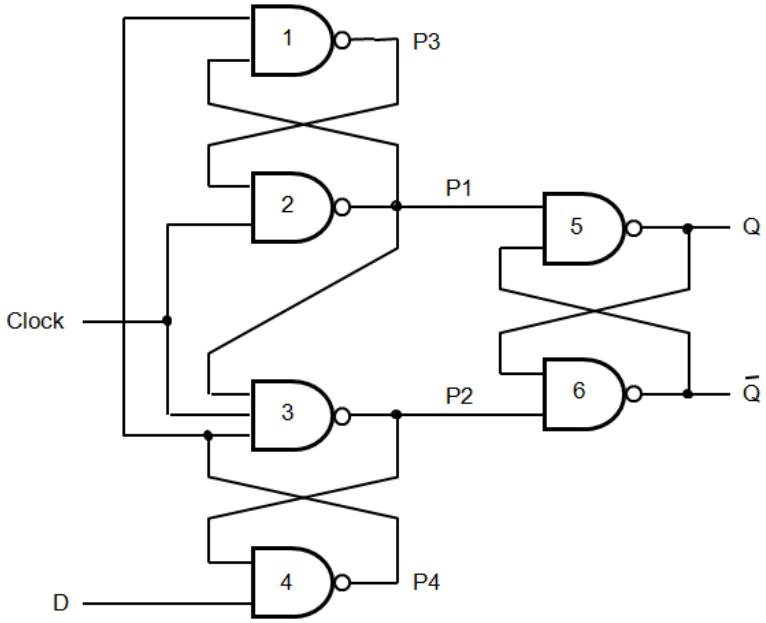
A and B → 1,
latching Q

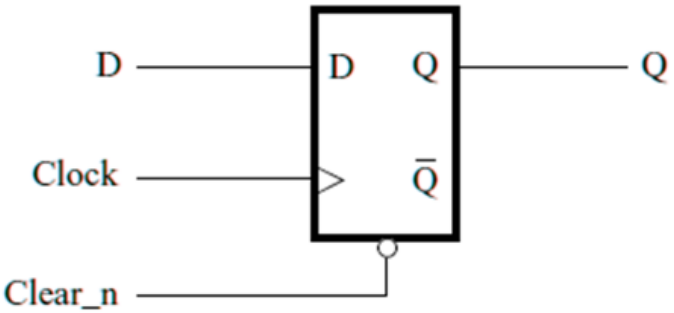
Requires a hold time: D is not allowed to change until the rising edge of the clock has propagated through A or B back to the input NANDs, latching whichever side was a zero.


Q
A
CLK
D
B
Q*

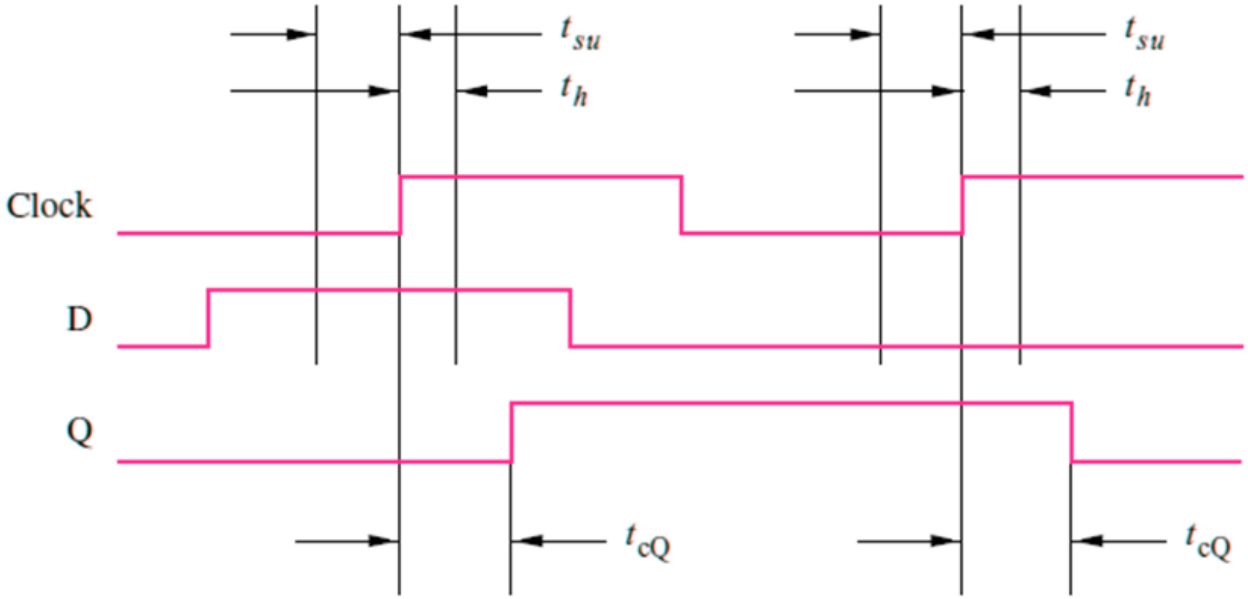

1
P3
2
P1
5
Q
Clock
3
P2
6
Q̄
4
P4
D


P1
1
P3
2
A
5
Q
CLK
D
4
P4
3
B
6
Q*
P2


1
P3
2
P1
5
Q
Clock
3
P2
6
Q̄
4
P4
D

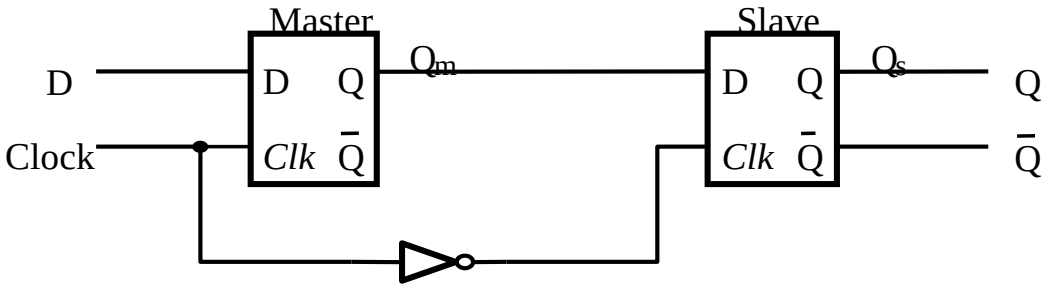


(a) D flip-flop with asynchronous clear

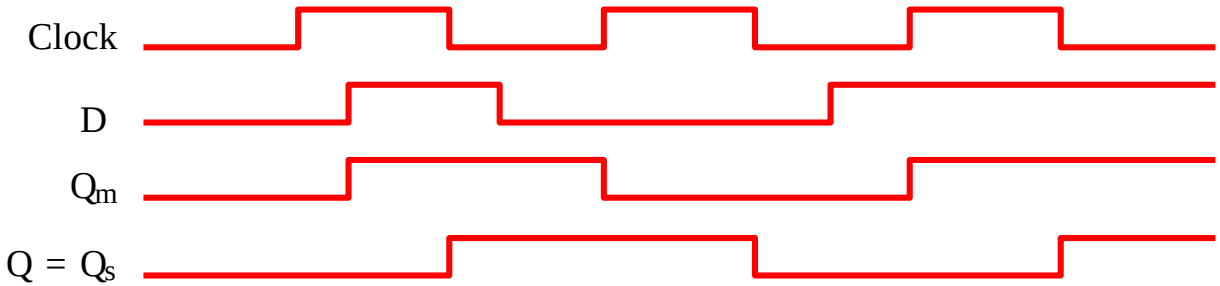


(b) Timing diagram

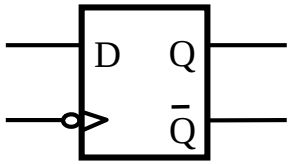
$t_{su} + t_h$ is called the "aperture" or "window" when the input must be good.



(a) Circuit

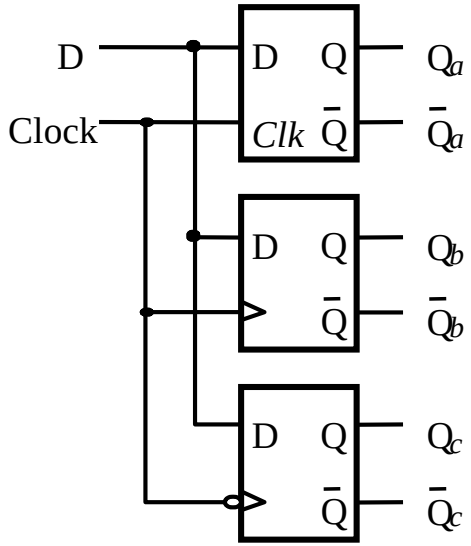


(b) Timing diagram

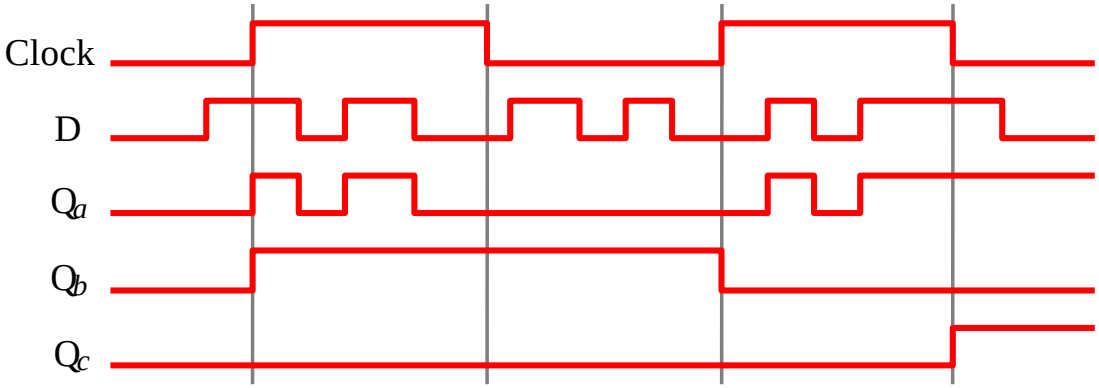


(c) Graphical symbol

Figure 5.9. Master-slave D flip-flop.



(a) Circuit



(b) Timing diagram

Figure 5.10. Comparison of level-sensitive and edge-triggered D storage elements.

```
module DMasterSlave1( input clock, D, output reg Q );

   wire Qm;
   DLatch1 master ( clock,  D,  Qm );
   DLatch1 slave  ( ~clock, Qm, Q  );

endmodule

module DMasterSlave2( input clock, D, output reg Q );

   reg Qm;
   always @( * )
      if ( clock )
         Qm = D;
      else
         Q = Qm;

endmodule
```

```
module DMasterSlave3( input clock, D, output reg Q );

    reg Qm;
    always @( negedge clock )
       Q <= D;

endmodule
```

```
module DMasterSlave3( input clock, D, output reg Q );

    reg Qm;
    always @( negedge clock )
       Q <= D;

endmodule

module DEdgeTriggered( input clock, D, output reg Q );

    always @( posedge clock )
       Q <= D;


endmodule
```

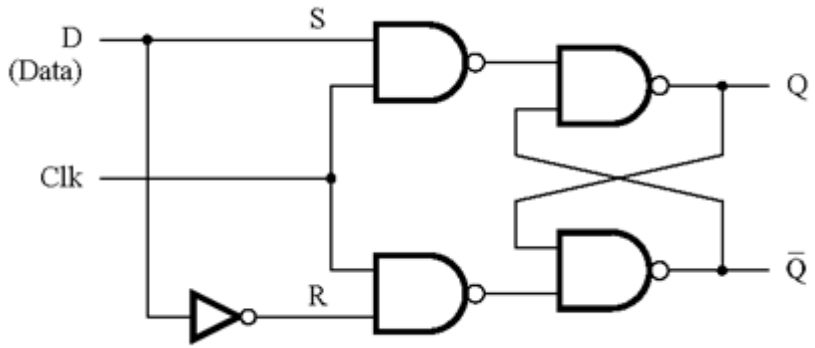
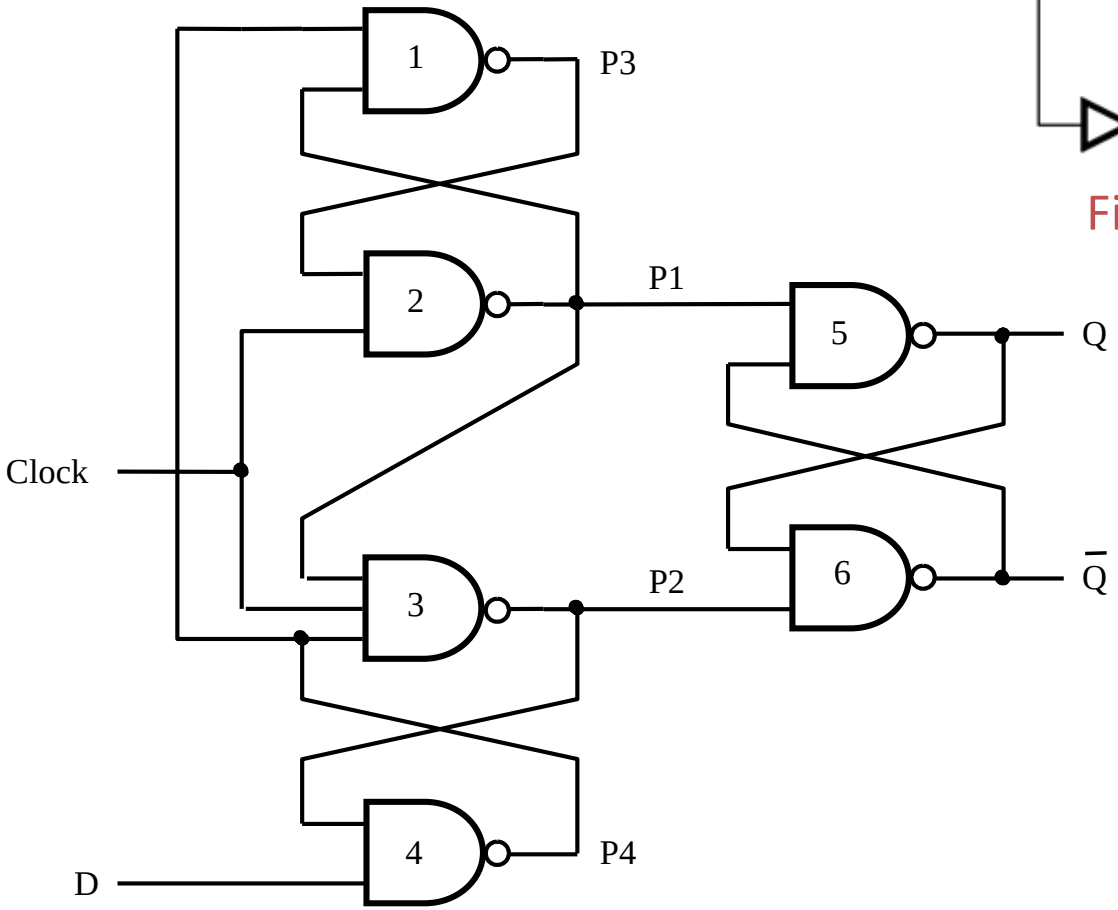
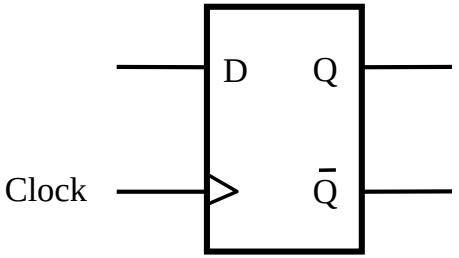


Figure 5.7. Gated D latch.

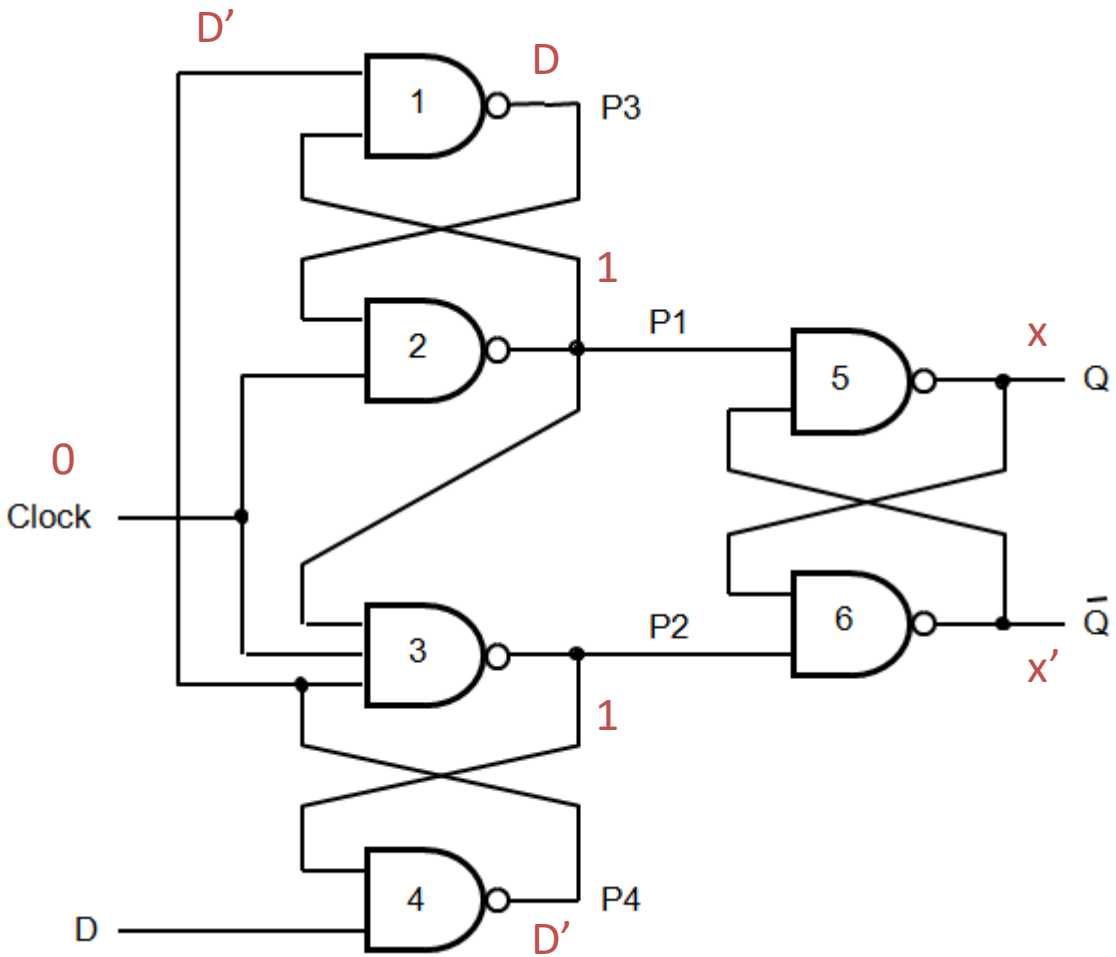




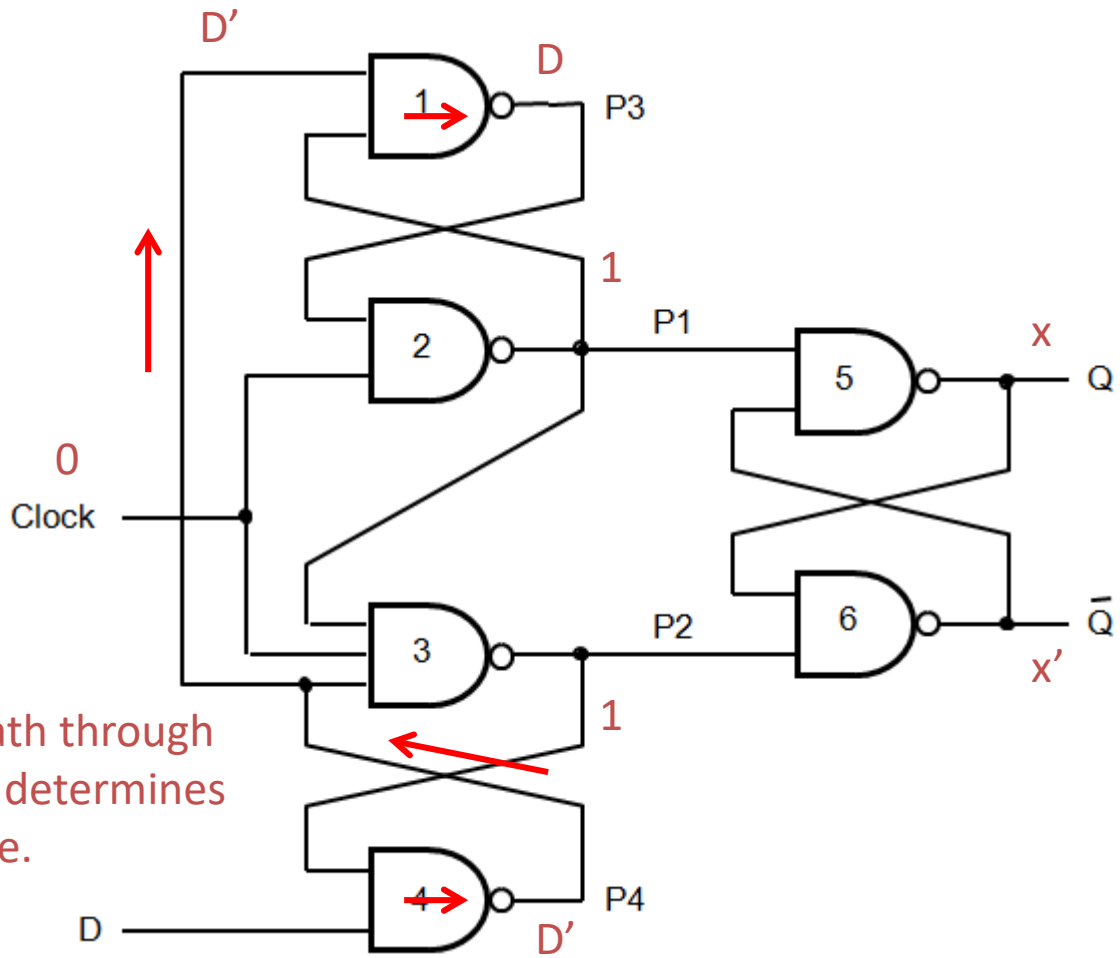
(a) Circuit

(b) Graphical symbol

Figure 5.11. A positive-edge-triggered D flip-flop.

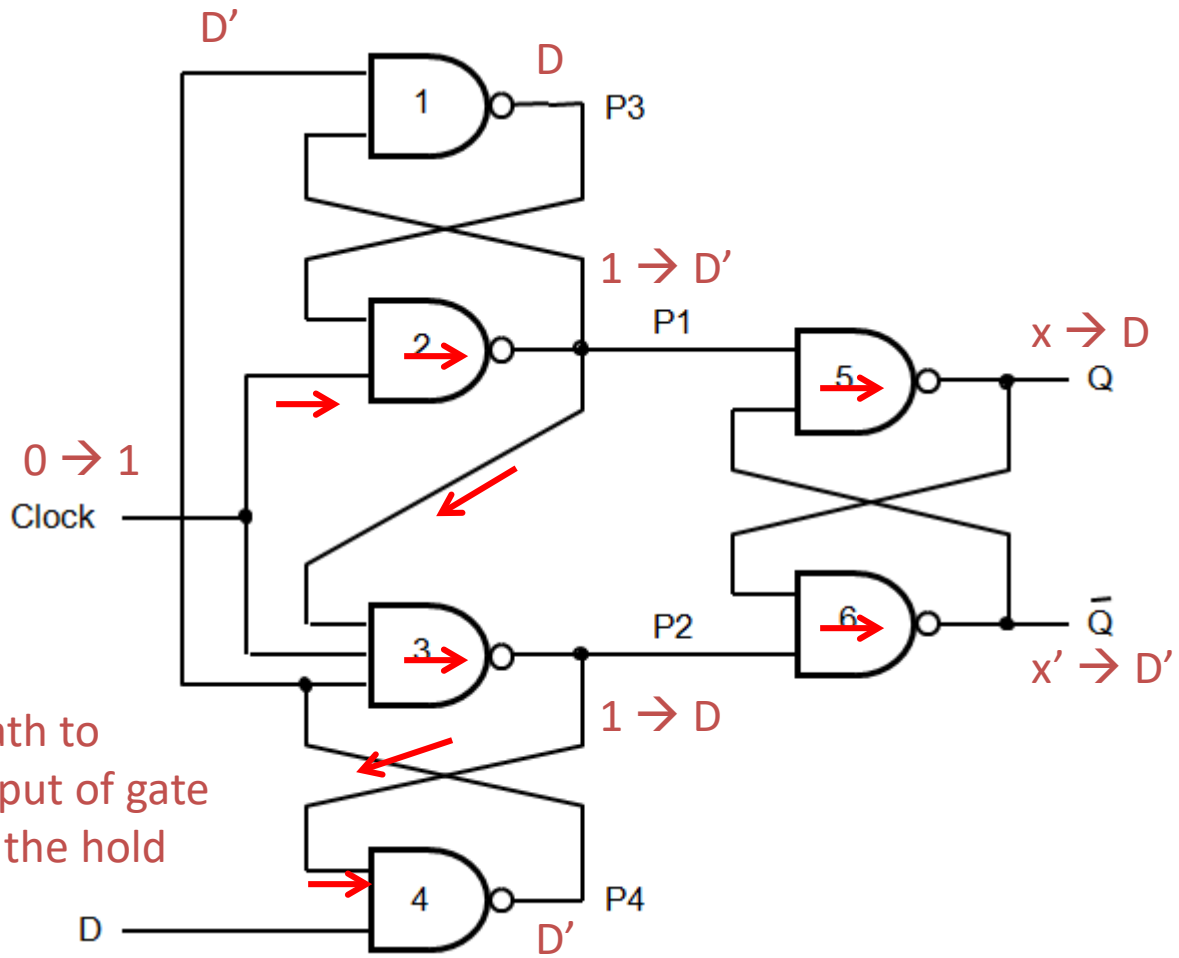


Steady-state clock = 0.



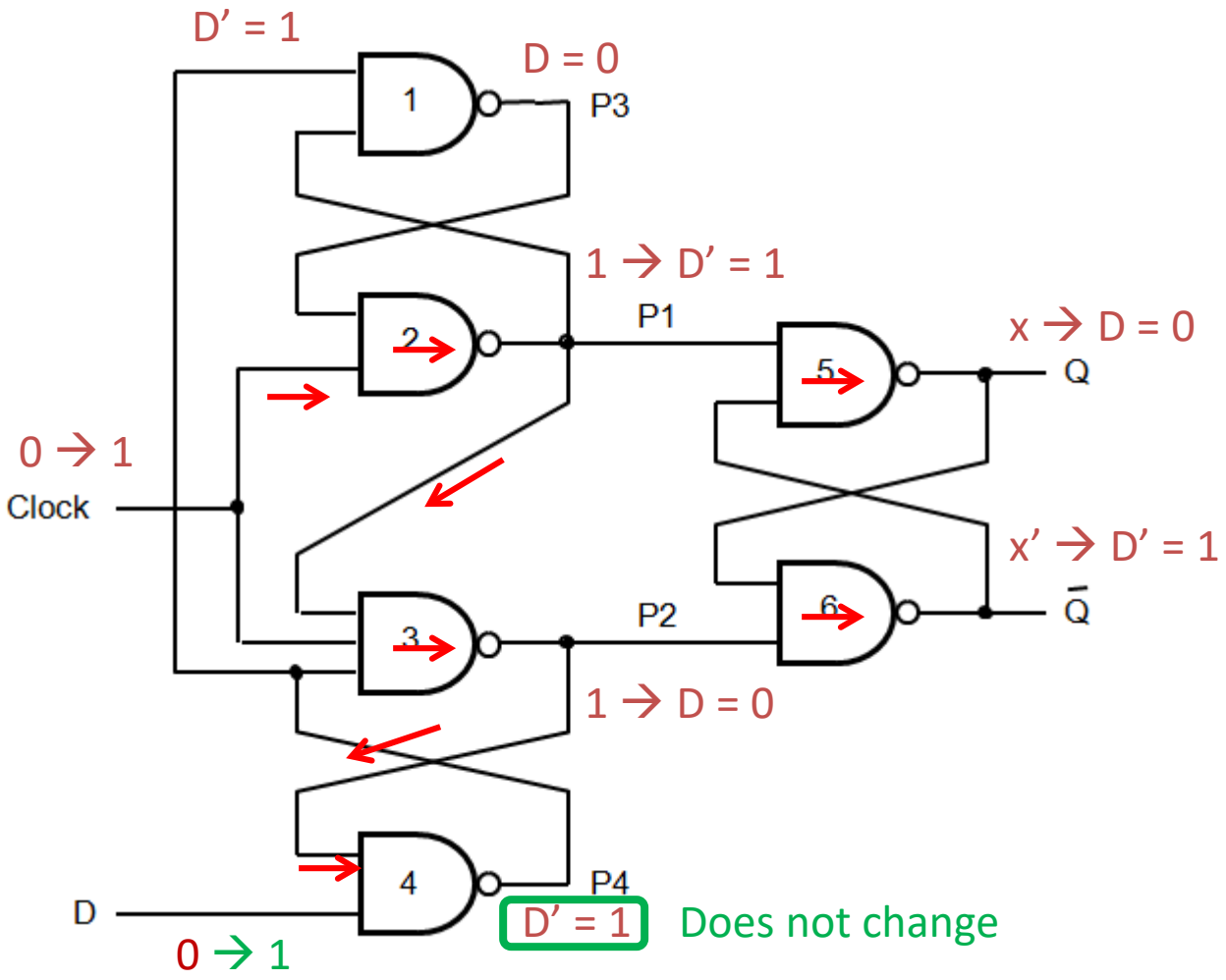
The critical path through gates 4 and 1 determines the setup time.

Steady-state clock = 0.

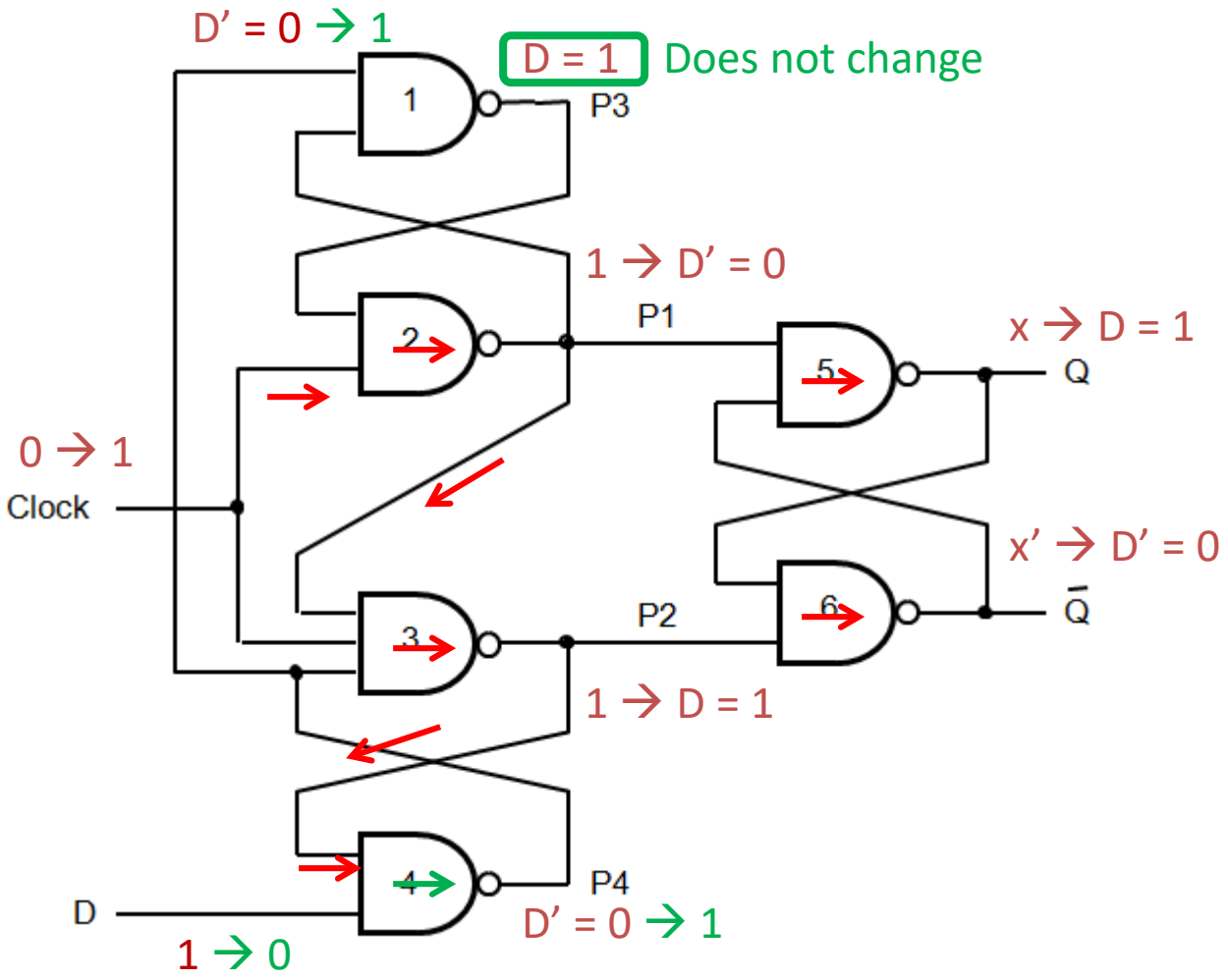


The critical path to disable the input of gate 4 determines the hold time.

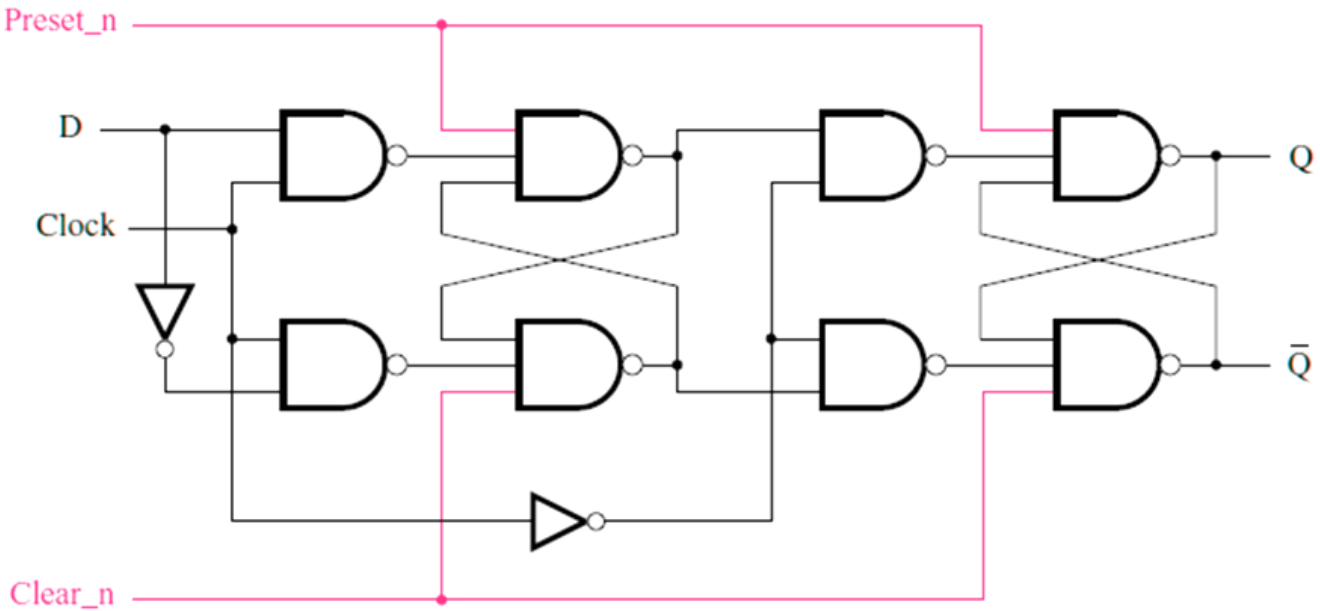
Clock transition 0 → 1



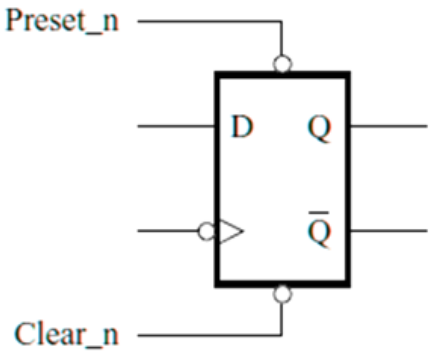
D transitions 0 → 1 after the hold time



D transitions 1 → 0 after the hold time

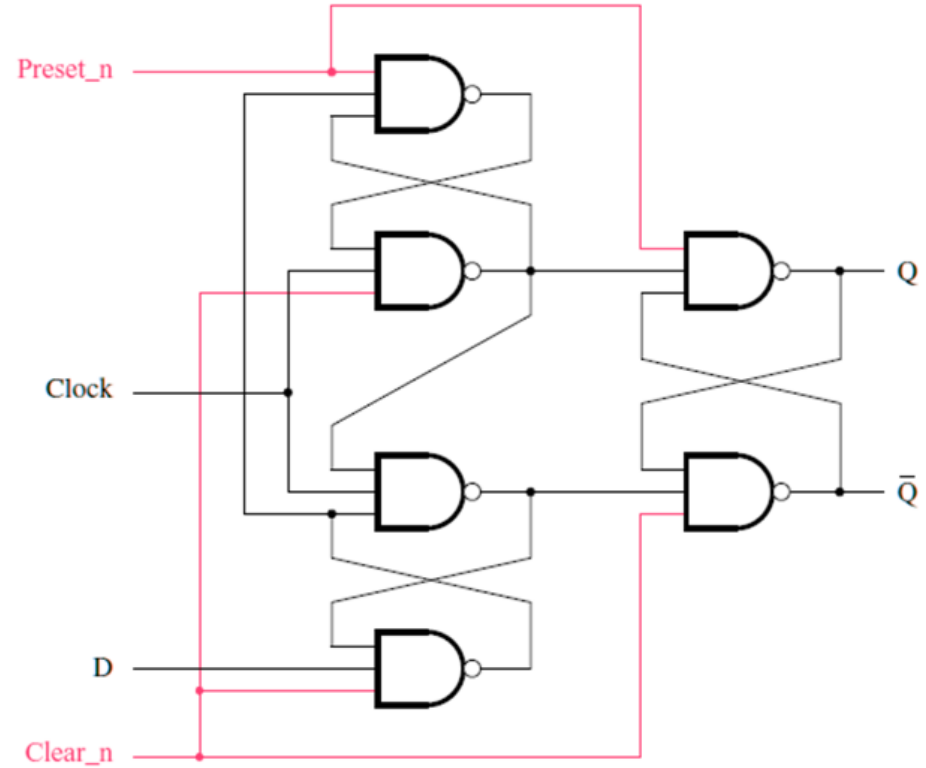


(a) Circuit

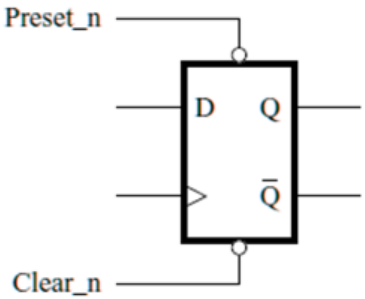


(b) Graphical symbol

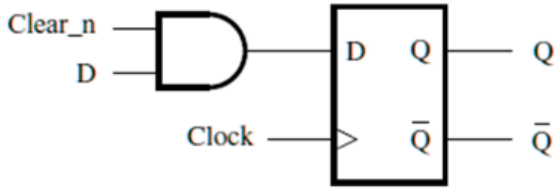
Figure 5.12. Master-slave D flip-flop with *Clear* and *Preset*.



(a) Circuit

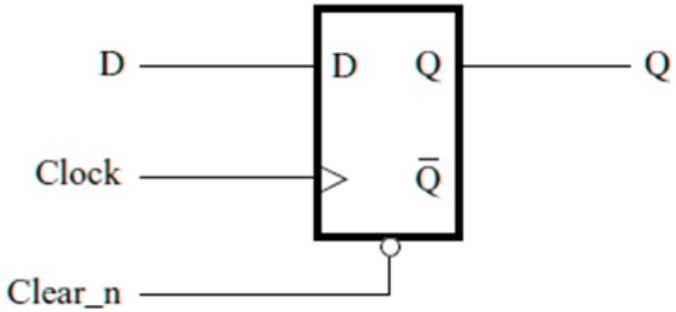


(b) Graphical symbol

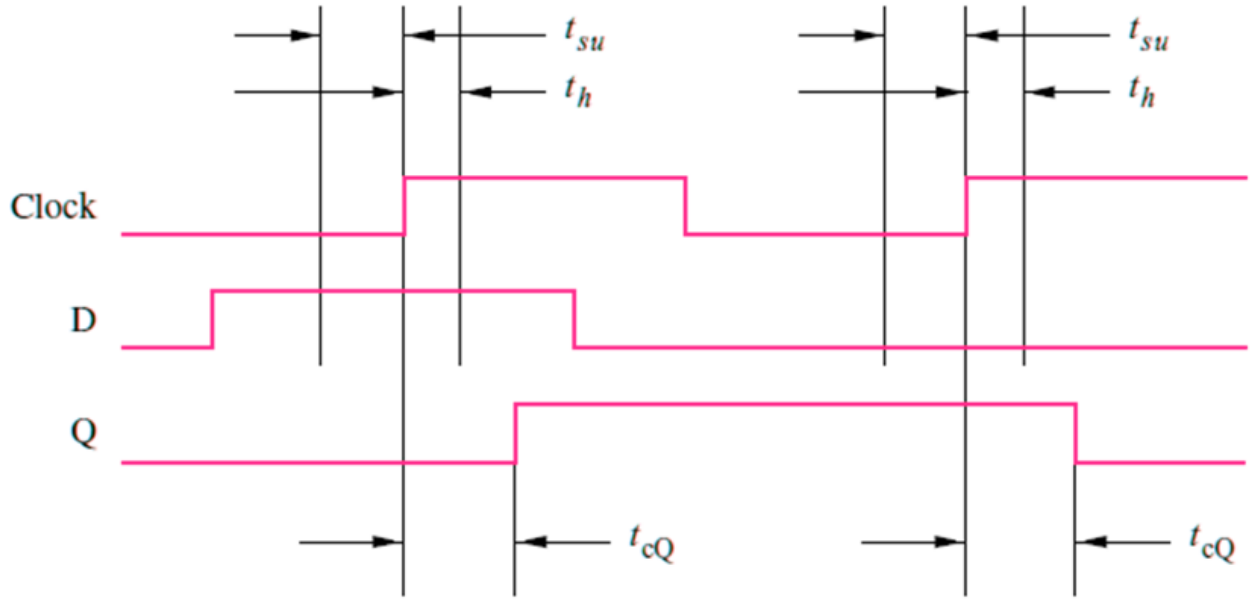


(c) Adding a synchronous clear

Figure 5.13. Positive-edge-triggered D flip-flop with *Clear* and *Preset*.

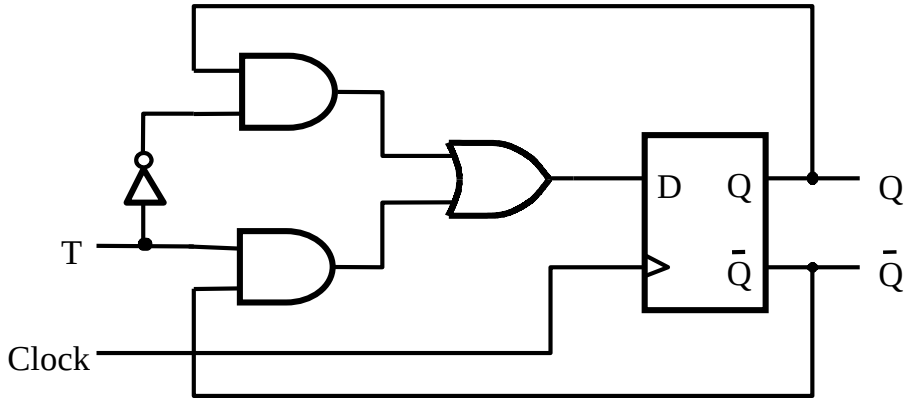


(a) D flip-flop with asynchronous clear



(b) Timing diagram

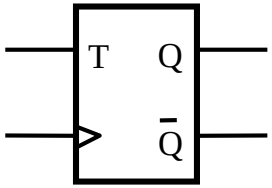
Figure 5.14. Timing for a flip-flop.



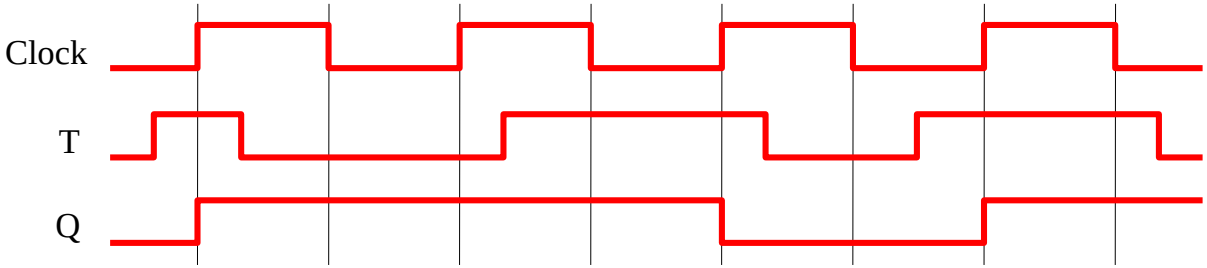
(a) Circuit

T	Q(t + 1)
0	Q(t)
1	$\overline{Q}(t)$

(b) Truth table

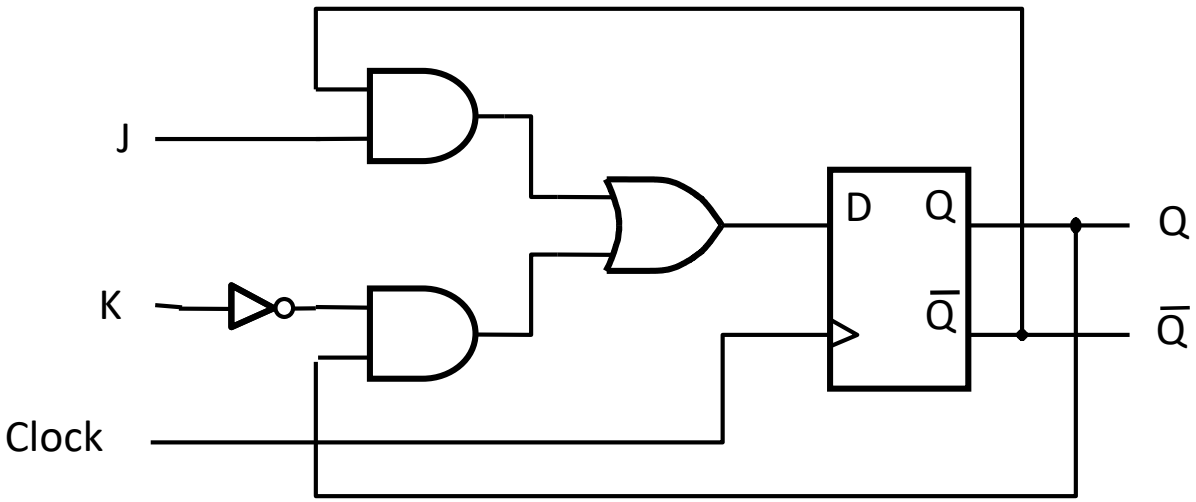


(c) Graphical symbol



(d) Timing diagram

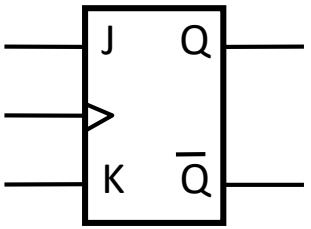
Figure 5.15. T flip-flop.



(a) Circuit

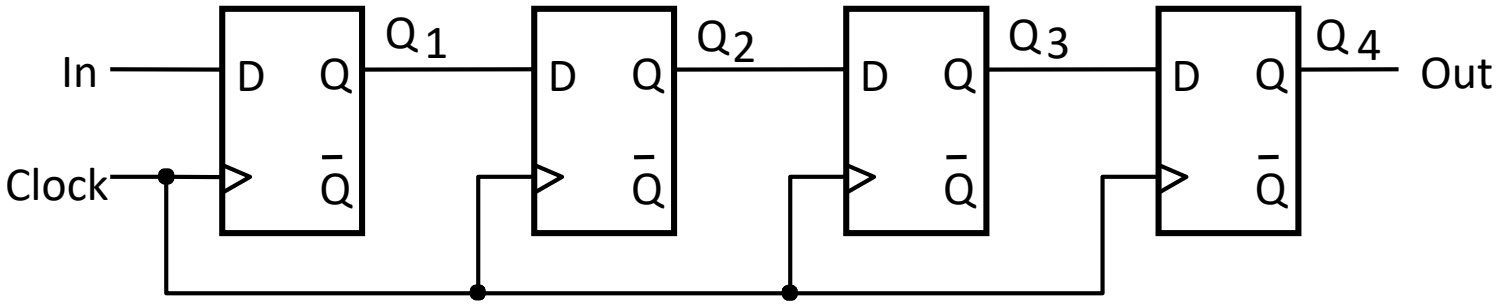
J	K	Q (t+1)
0	0	Q (t)
0	1	0
1	0	1
1	1	$\overline{Q}$ (t)

(b) Truth table



(c) Graphical symbol

Figure 5.16. JK flip-flop.



(a) Circuit

	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

(b) A sample sequence

Figure 5.17. A simple shift register.