Concepts

In sequential logic, the outputs depend not only on the inputs, but also on the preceding input values... it has memory.

Memory can be implemented in 2 ways:

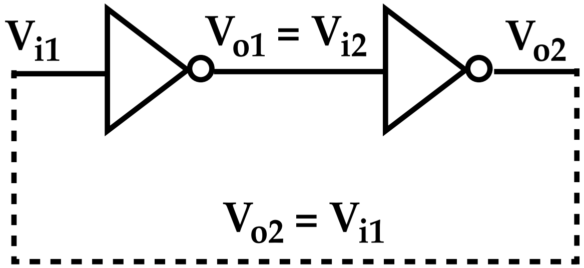
- Positive feedback or **regeneration** (static):
  One or more output signals are connected back to the inputs via storage elements.
  These circuits are called *multivibrator*.
  Bistable elements such as flip-flops are most common but *monostable* and *astable* circuits are also used.
- Charge storage (dynamic):
  As we know, a periodic refresh is necessary here.

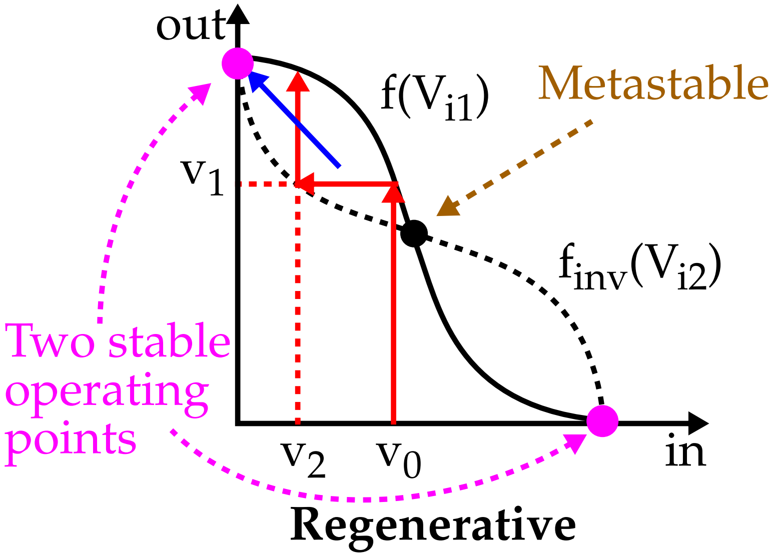
The **bistable** element can be either static or dynamic and is an essential library element called a register.

An **astable** multivibrator acts as an oscillator (clock generator) while a **monostable** multivibrator can be used as a pulse generator.

Static Sequential Circuits

We've already discussed the regenerative property.





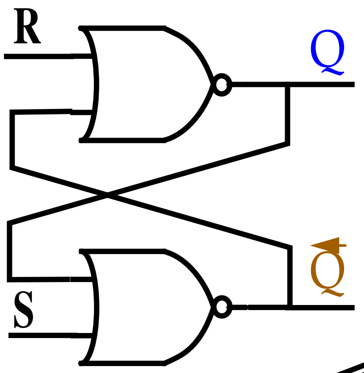
If the gain of the inverter in the transient region is *greater than 1*, there are only two stable operating points.

Storing a new value usually involves applying a *trigger pulse* for a duration equal to the propagation delay through the two inverters.

The trigger pulse takes either V_{i1} or V_{i2} temporarily out of the region where the gain, G, is *less than 1* to the unstable region where $G > 1$.

Flip-flop Classification

NOR version



Positive logic

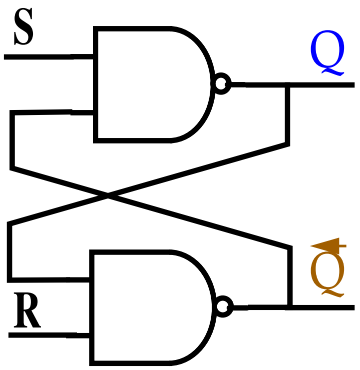
S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	0	0

Set-Reset Flip-flop

The length of the *trigger pulse* applied to S or R has to **larger** than the loop delay of the cross-coupled pair.

Note that this mode is forbidden since the constraint Q and $\overline{Q}$ are not complementary. Also, the return to 00/11 leaves the FF in an unpredictable state.

NAND version

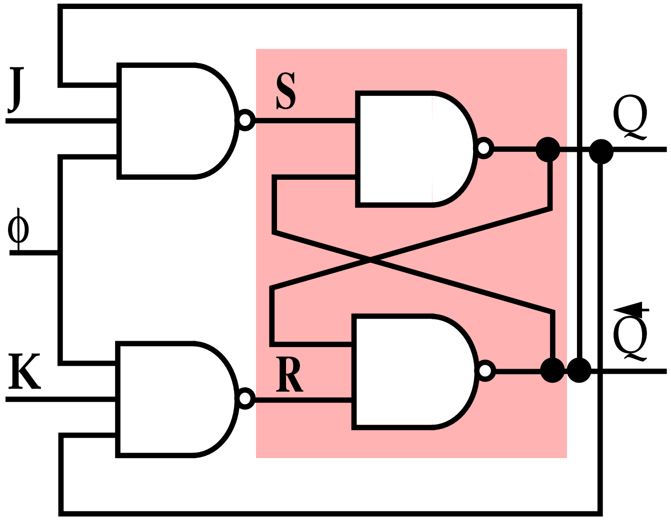


Negative logic

S	R	Q	$\overline{Q}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	Q	$\overline{Q}$

Flip-flop Classification

The ambiguity of having a *non-allowed mode* caused by trigger pulses going active simultaneously can be avoided by adding two feedback lines:



J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

Note the characteristic table is similar to SR FF except for the forbidden mode

Note if both *J* and *K* are high, and clock pulses, the output is complemented.

However, doing so enables the other input and the FF *oscillates*.

This places some stringent constraints on the **clock pulse width** (e.g. < than the propagation delay through the FF).

Flip-flop Classification

Synchronous circuit:

Changes in the output logic states of all FFs are synchronized with the clock signal, ϕ.

Note that:

- T FF (*toggle FF*) is a special case of the JK with J and K tied together.
- D FF (*delay FF*) is a special case with J and K connected with complementary values of the D input.

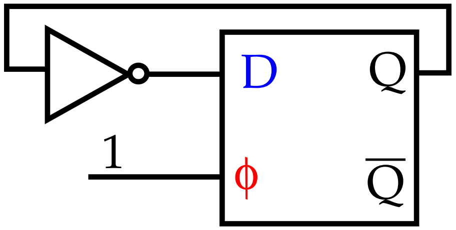
It generates a delayed version of the input synchronized with the clock.

These FFs are also called **latches**.

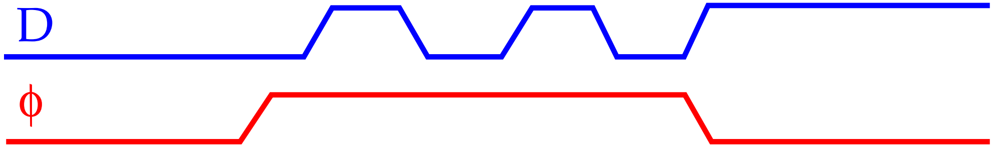
A FF is a latch if the gate is transparent while the clock is high (low).

Any changes in the input appear in the output after a nominal delay.

The transparent nature can cause **race** problems:

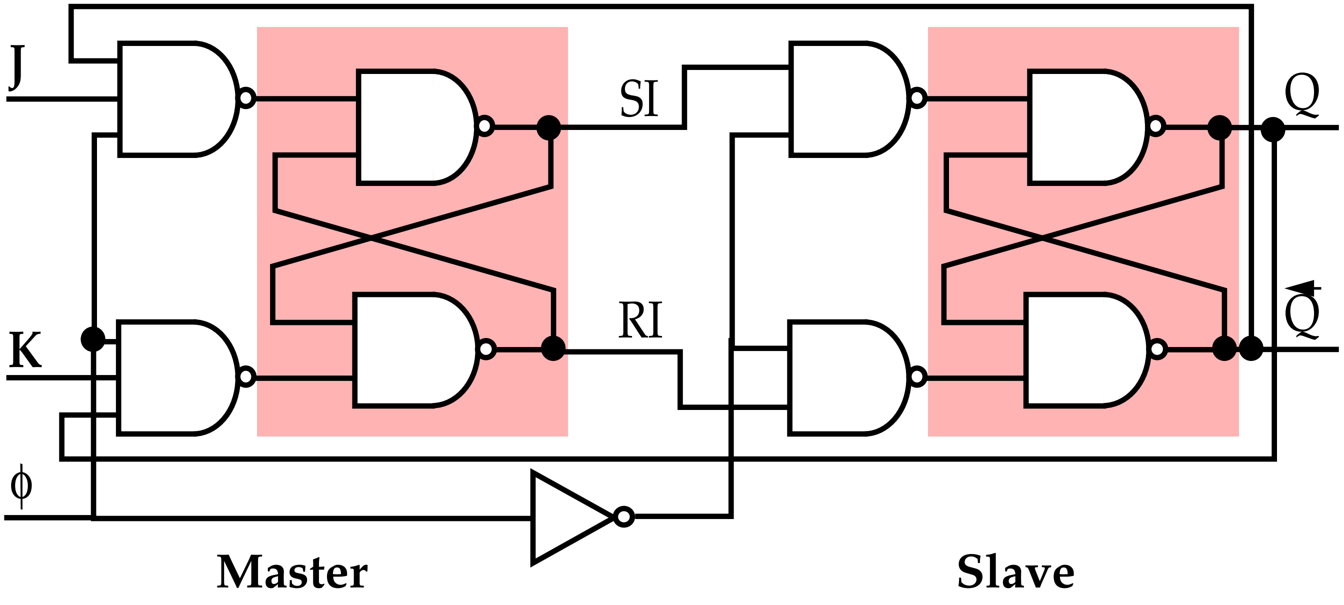


This circuit oscillates as long as ϕ remains high.



Master-Slave FFs

One way to avoid the race is to use the master-slave approach.



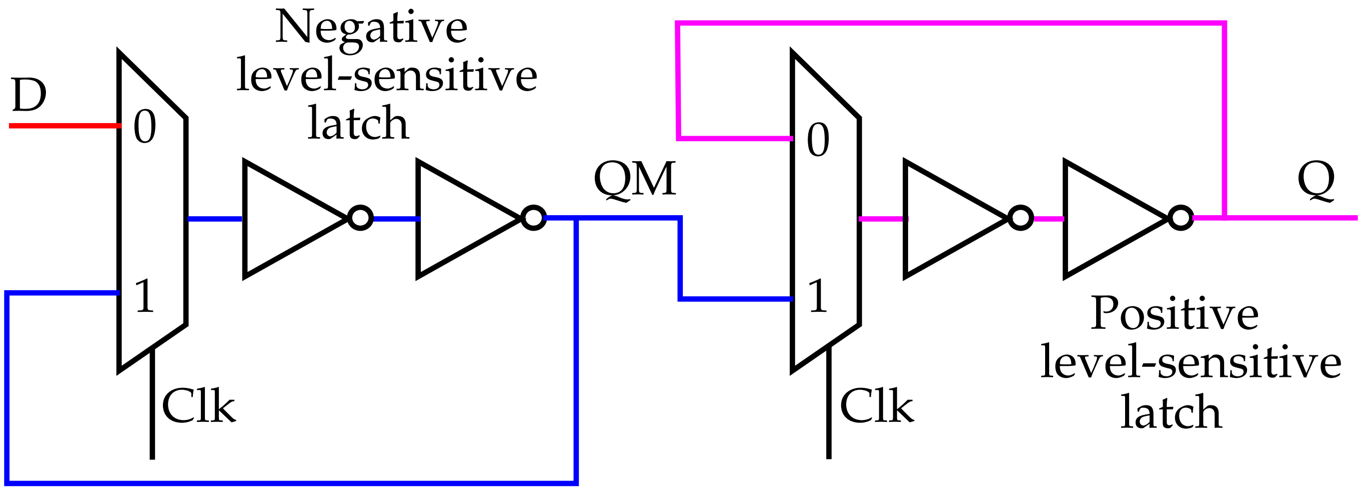
The master on the left is active (*J* and *K* are enabled) when ϕ is high.

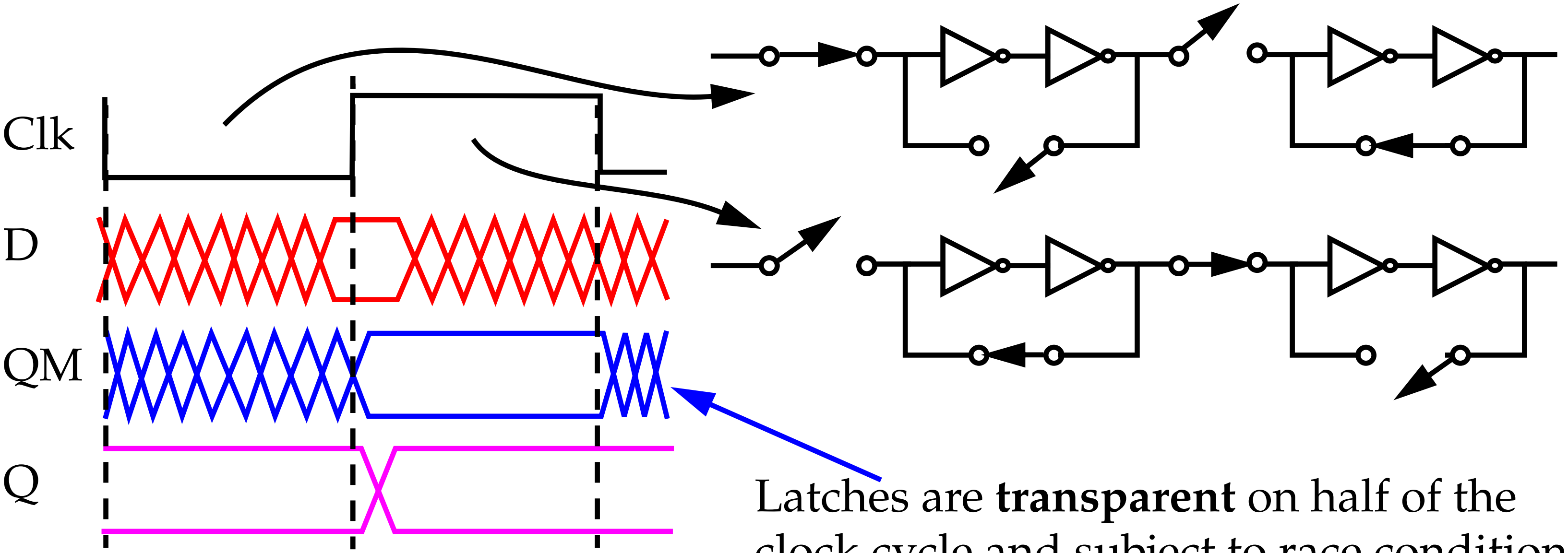
The slave on the right is in hold mode, preventing changes on *SI* and *RI* from propagating to the output, *Q*.

When ϕ goes low, the state of the master is frozen and the NAND gates in the slave are enabled.

There is no constraint on the **maximum** width of ϕ for proper operation.

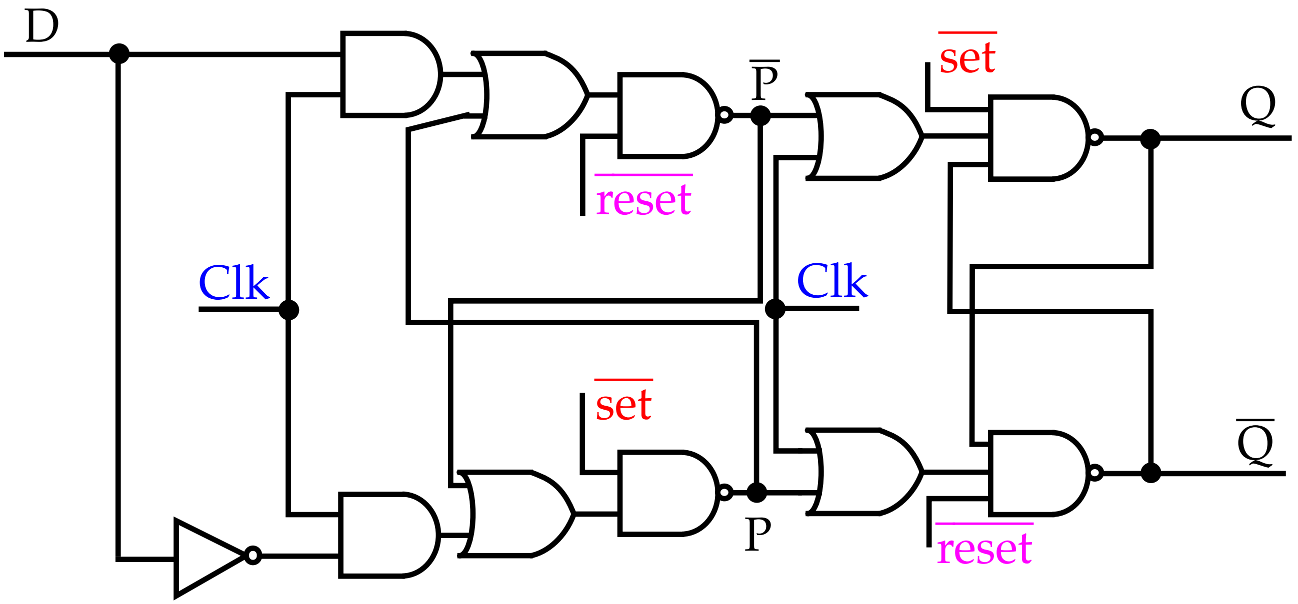
Master-Slave FFs



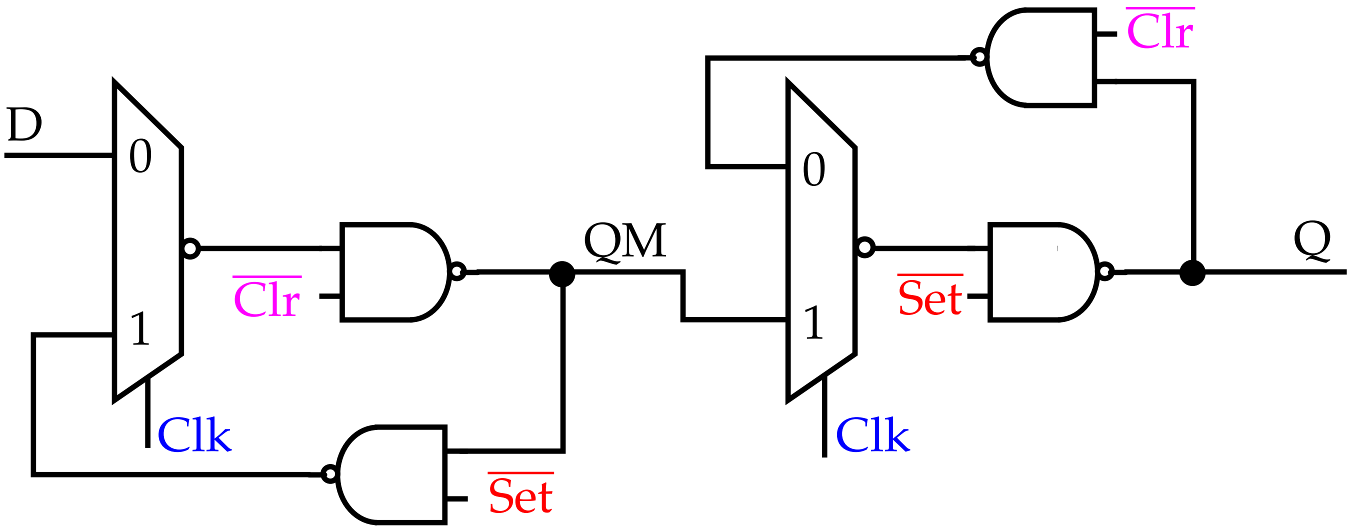


Latches are **transparent** on half of the clock cycle and subject to race conditions.

Master-Slave Set/Clear Asynchronous FFs

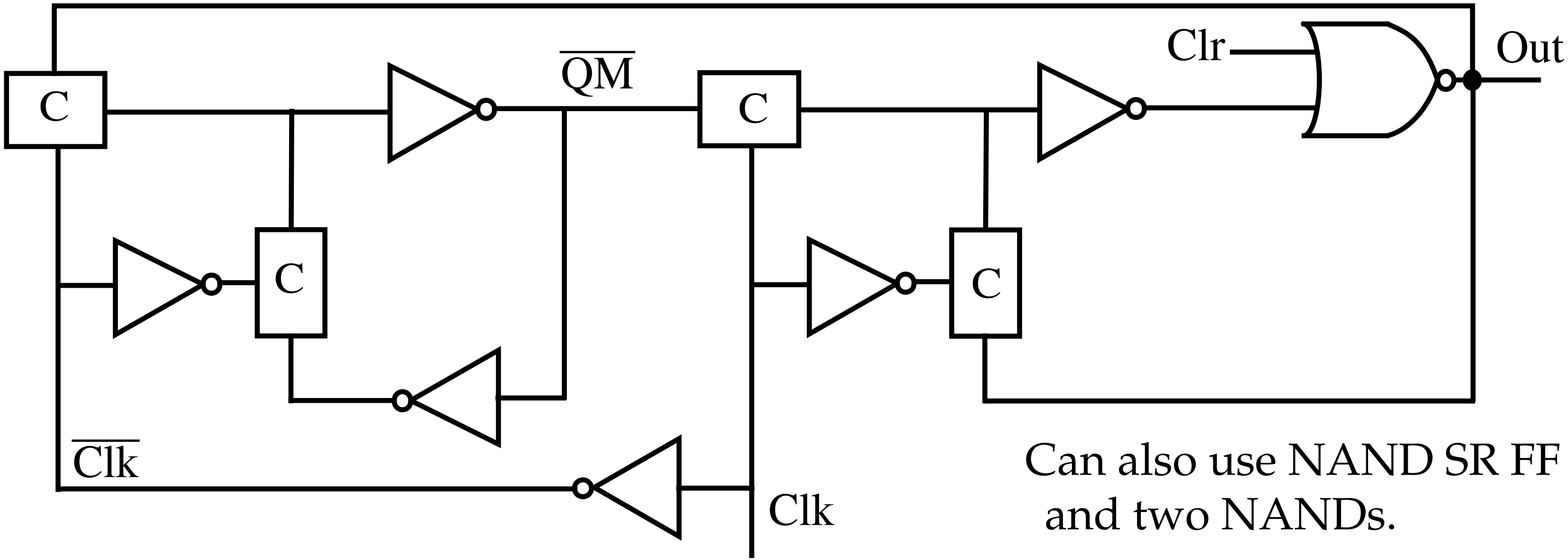


Or



Toggle Flip-Flop with Asynchronous Clear:

T FF



Can also use NAND SR FF and two NANDs.

Divides Clk by 2.
Used in counters.

Edge-triggered FFs

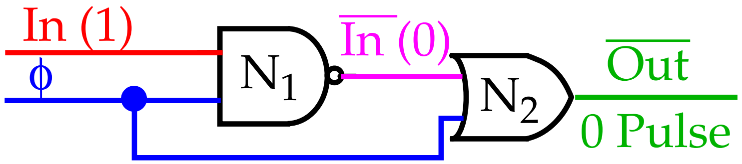
Problem with master-slave approach:

The circuit is sensitive to changes in the input signals as long as ϕ is high.

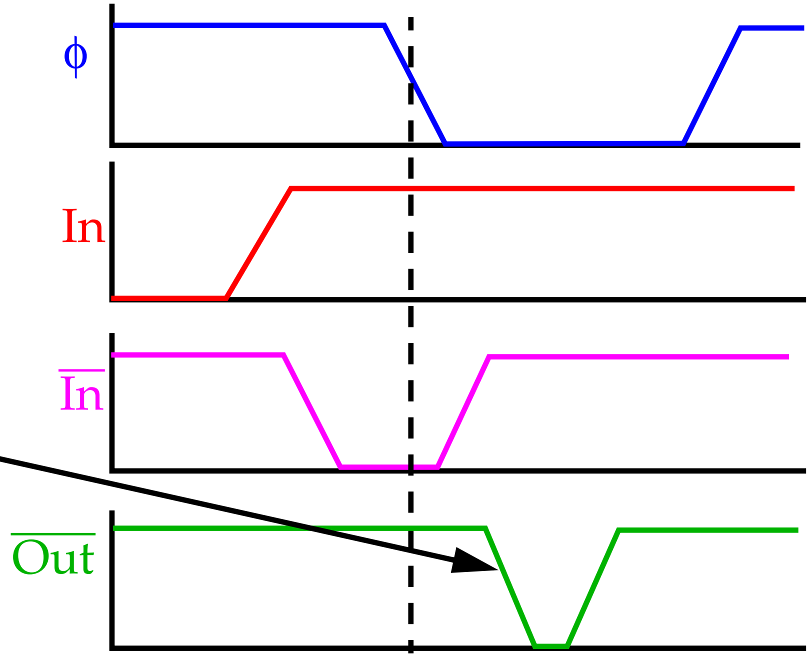
In the case of the JK FF, the inputs MUST stay constant with ϕ high.

If FF is reset, it is sensitive to the level of *J*, e.g., 1 glitches.

The fix is to allow the state of the FF to change only at the rising (falling) edge of the clock.

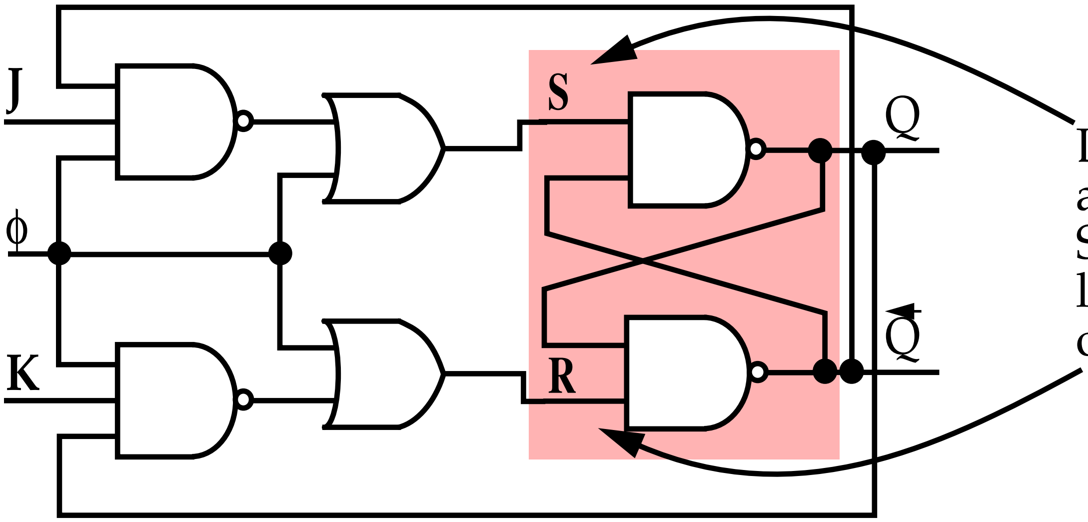


Results in a short low-going pulse at the output of N_2 with length approximately equal to the propagation delay through N_1.



Edge-triggered FFs

The modification applied to the *JK* FF is shown below.



Low-going pulses
are generated on
S and R with the
low going edge
of the clock.

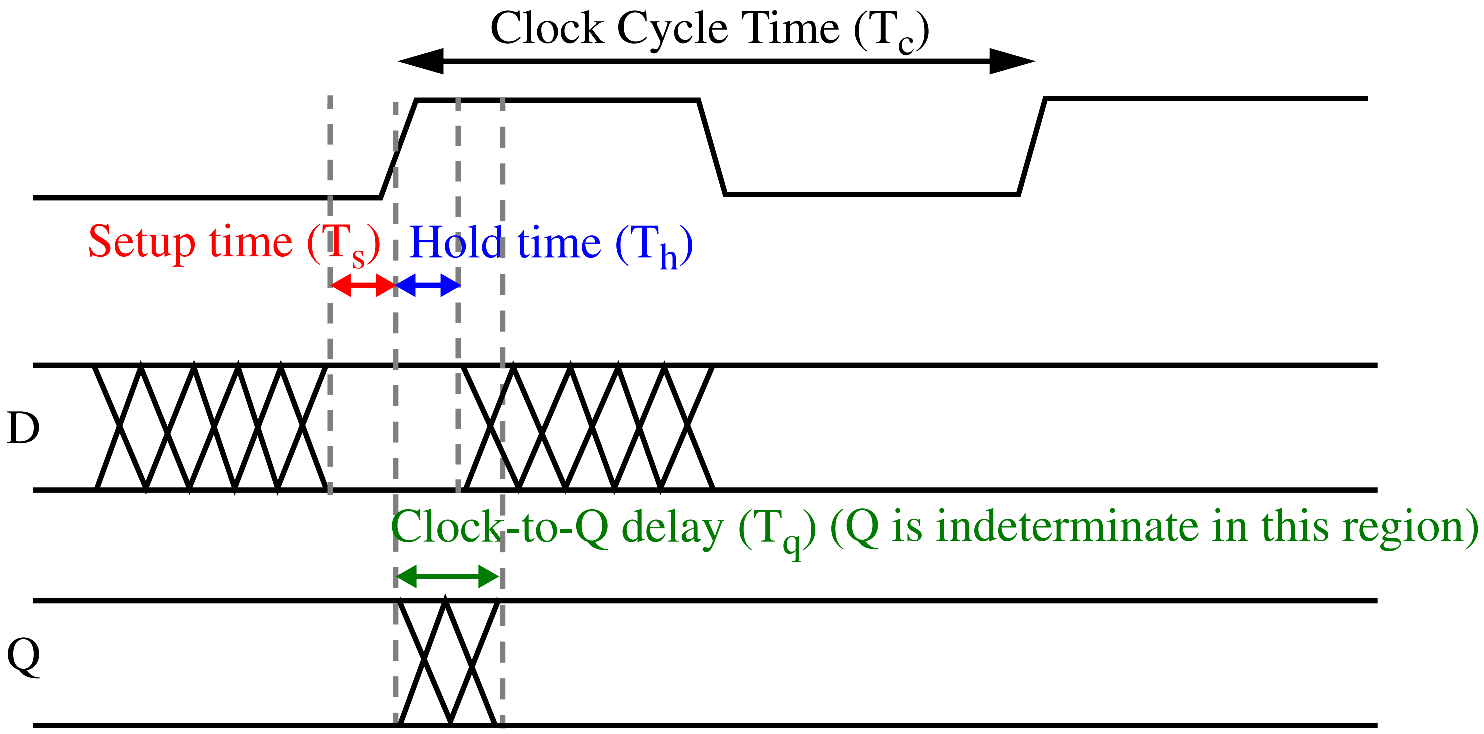
Note that the inputs must be stable for some time before the clock goes low.

This is also true for the master-slave D FF, but the constraints are different.

Let’s first define some terms.

Flip-Flop Timing Definitions

Timing diagram showing the terms defining the proper operation of a FF.



T_c: Clock Cycle Time.

T_s: The amount of time *before* the clock edge that the D input has to be stable.

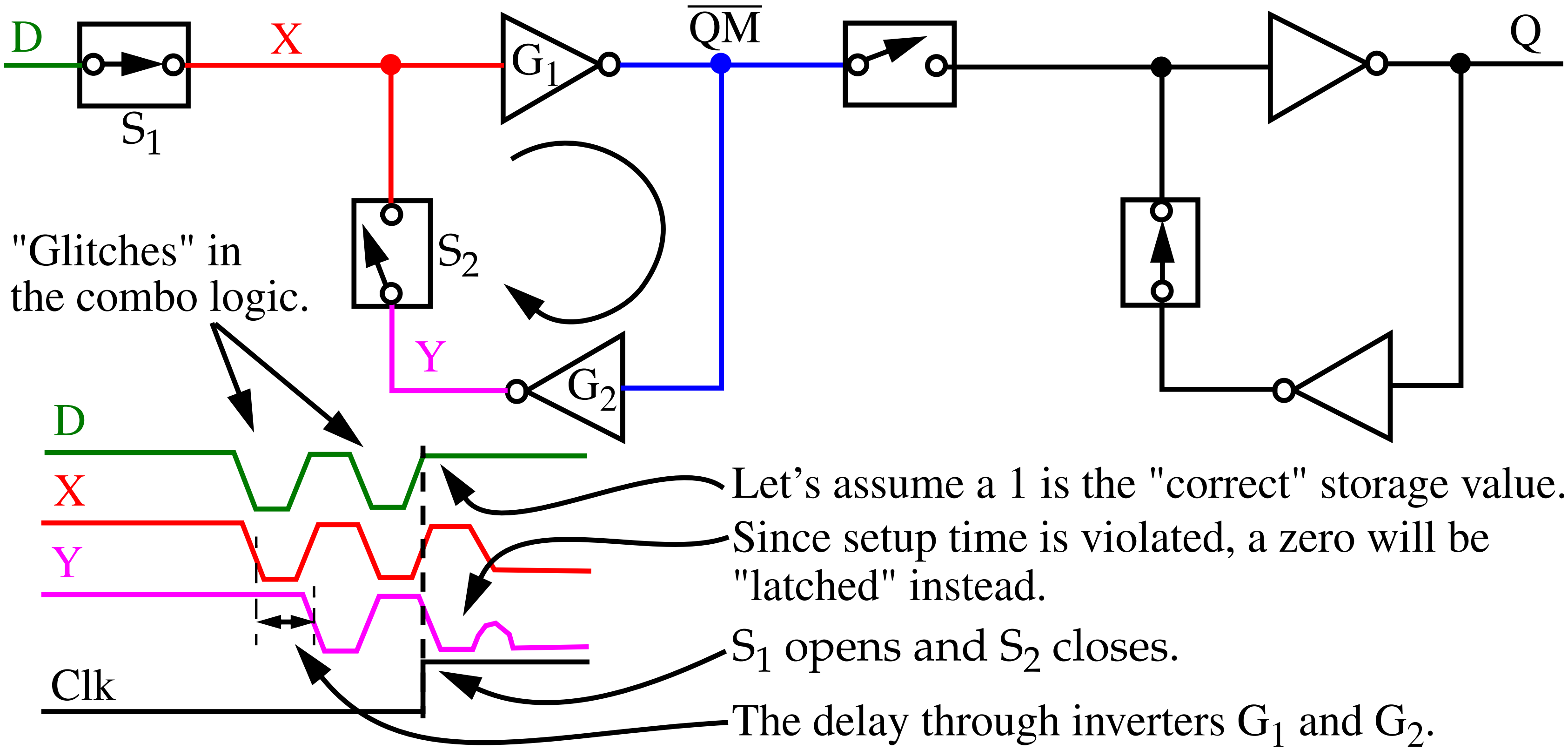
T_h: Data has to be held for this period while clock travels to point of storage.

T_q: Clock-to-Q delay: Delay from the positive clock input to new value of Q.

Setup/Hold Time Violations

Depending on the design, one or both of T_s and T_h may have to be non-zero.

For example, the master-slave D FF is likely to require a longer setup time than the edge-triggered D FF.



"Glitches" in the combo logic.

Let's assume a 1 is the "correct" storage value.

Since setup time is violated, a zero will be "latched" instead.

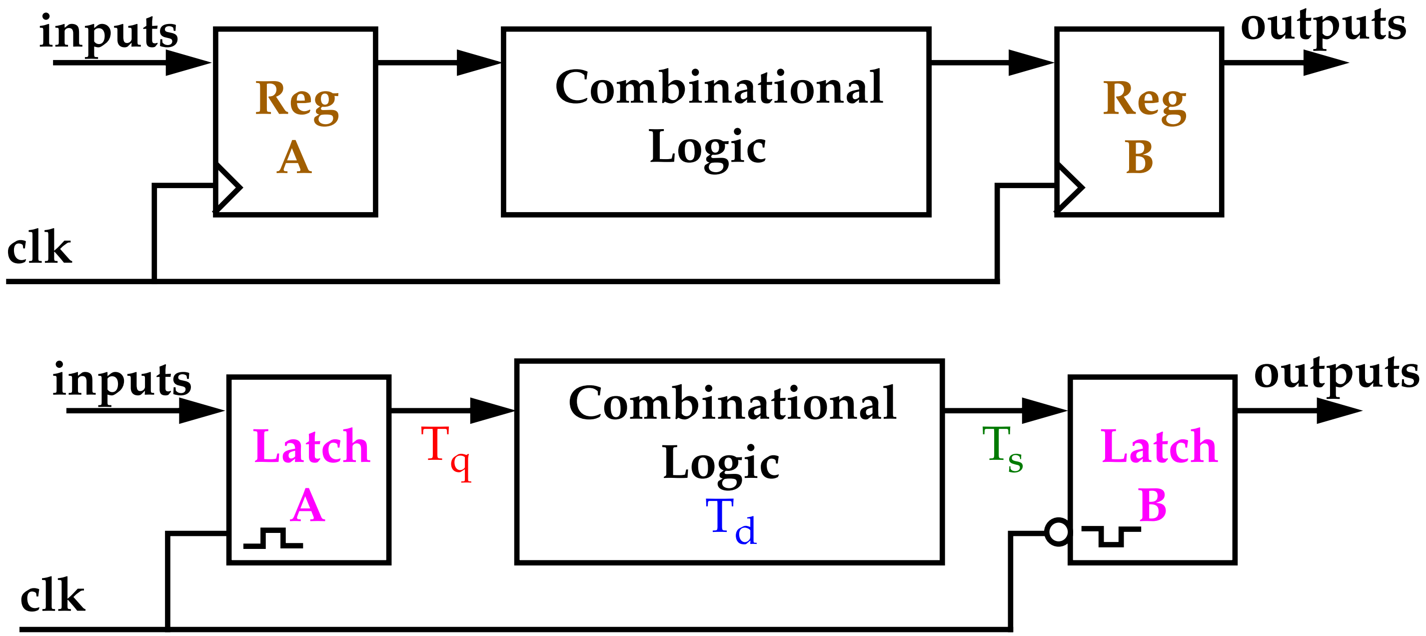
S_1 opens and S_2 closes.

The delay through inverters G_1 and G_2.

Edge triggered FF prevents the "master" from following the *D* input so the FF's internal delay does not affect setup time.

System Timing

Two possible strategies to implement clocked systems:



Latches are a more economical implementation strategy but are transparent on half of the clock cycle, i.e., cannot be used in feedback systems.

Also, the following constraint must be met for latches:

$T_d < T_c/2 - T_q - T_s$

where T_d is the worst case propagation delay, T_c is the clock cycle time, T_q is the Clock-to-Q time of latch A and T_s is the setup time for latch B.

Clock Race Conditions

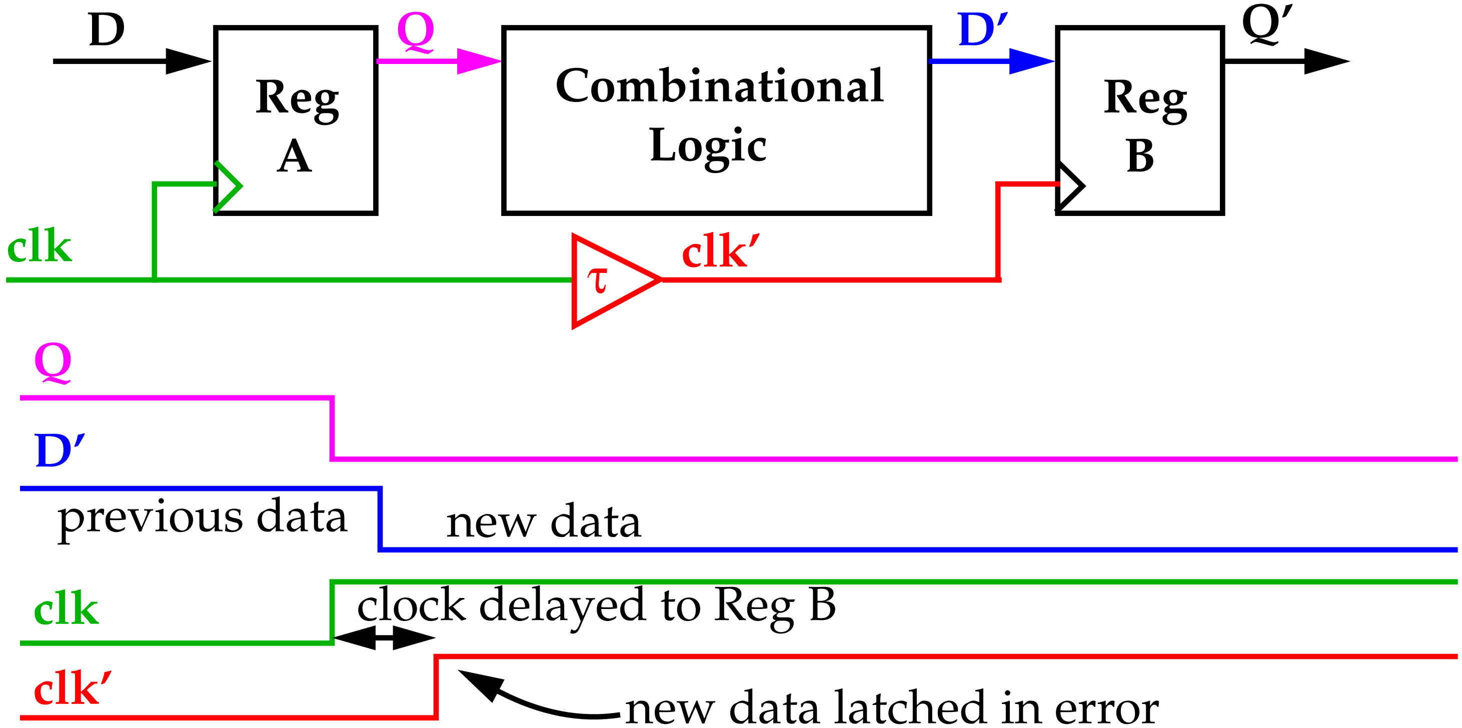
For edge-triggered FFs, the following time constraint must be met:

$$T_q + T_d + T_s < T_c$$

Clock races caused by:

- Delays in the clock line to Reg B.

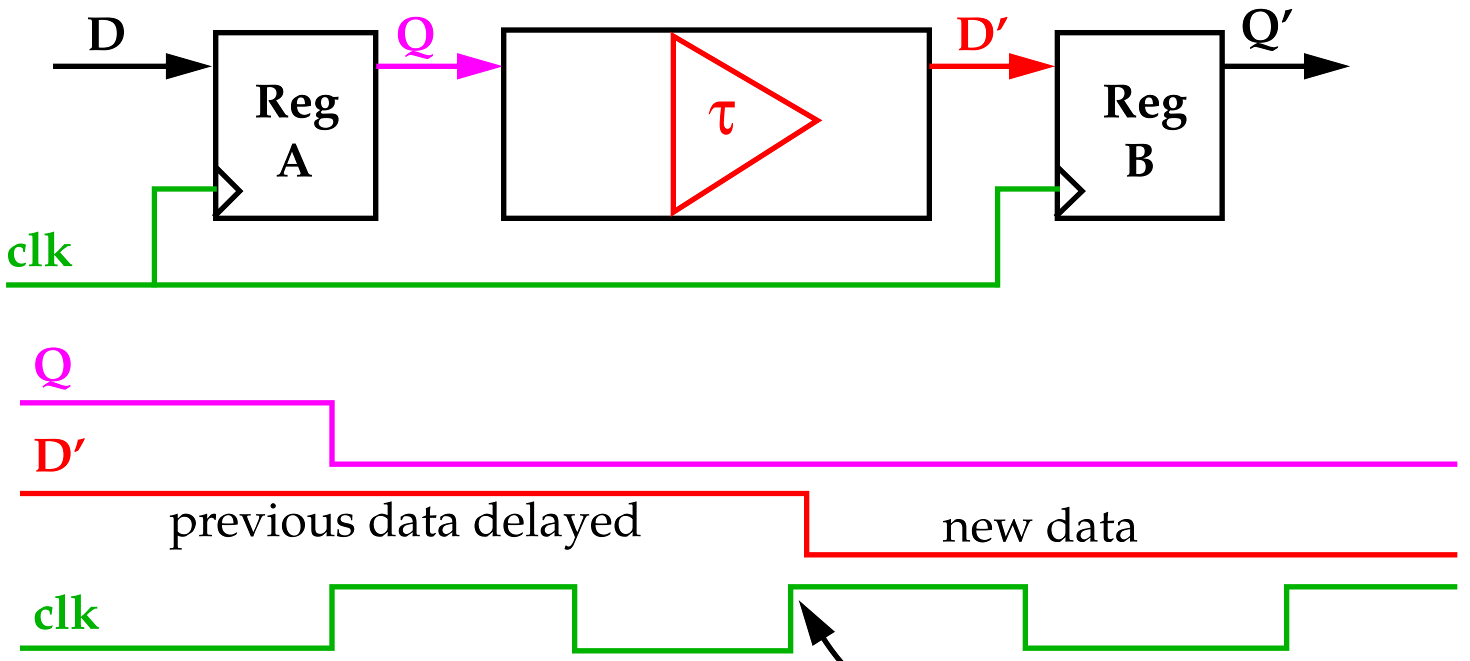
New data stored instead of previous data:



Clock Race Conditions

- Delays in the combinational logic that are larger than the clock cycle time.

  Data arrives late at Reg B, old data retained instead of latching new data.

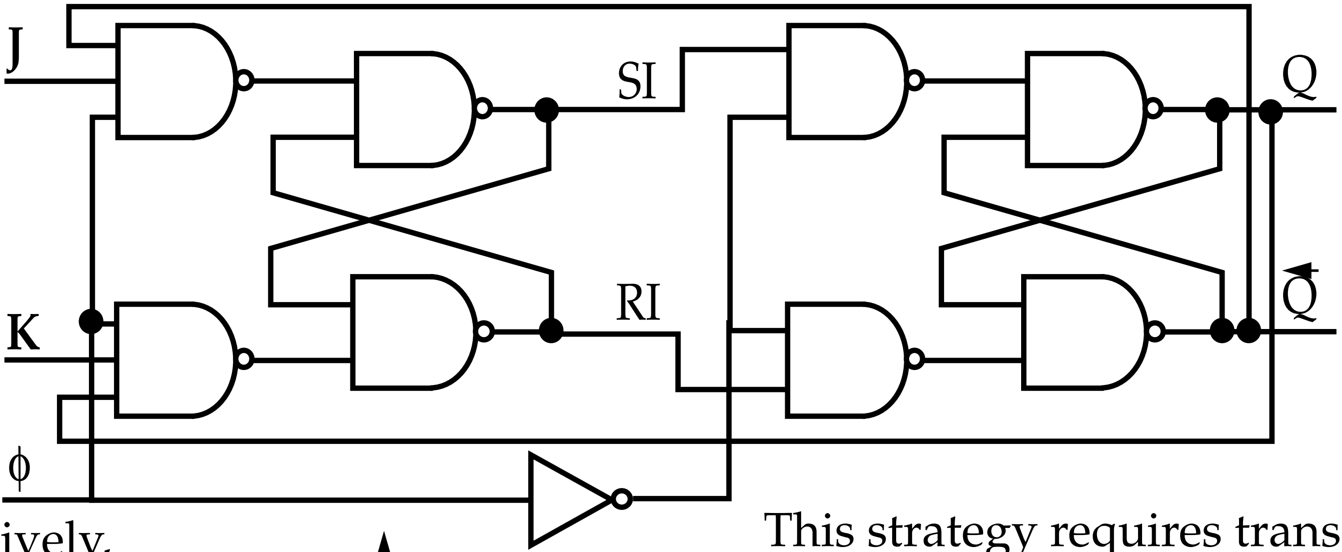


Old data latched in error.

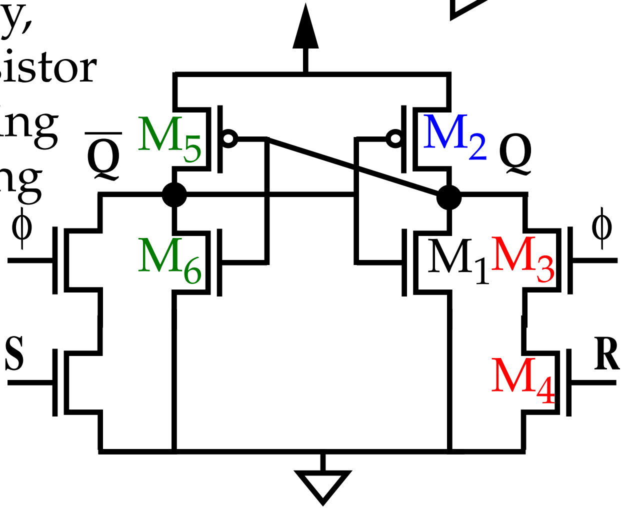
As you can see, designers have to walk a temporal ’tight-rope’, e.g., they have to minimize clock skew while considering worst and best case delays through combinational logic.

CMOS Static Flip-Flops

Full complementary version of the master-slave FF requires 38 transistors!



Alternatively, an 18 transistor version using this building block:



This strategy requires transistor sizes to be taken into account.

Assume Q is high and R pulsed. M_3 and M_4 must "overpower" M_2 and reduce Q to < threshold of M_5 and M_6.

A variation of this, which combines ϕ and R/S, is the 6 trans. SRAM.