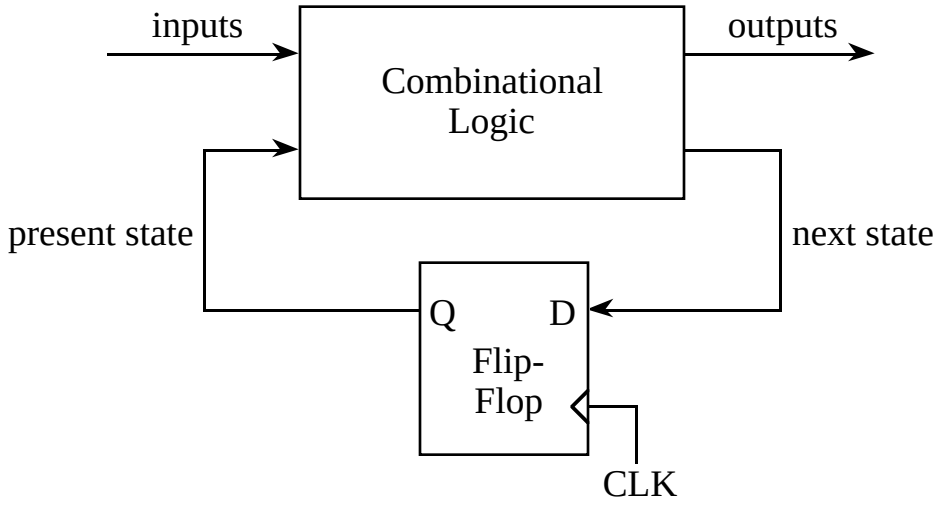
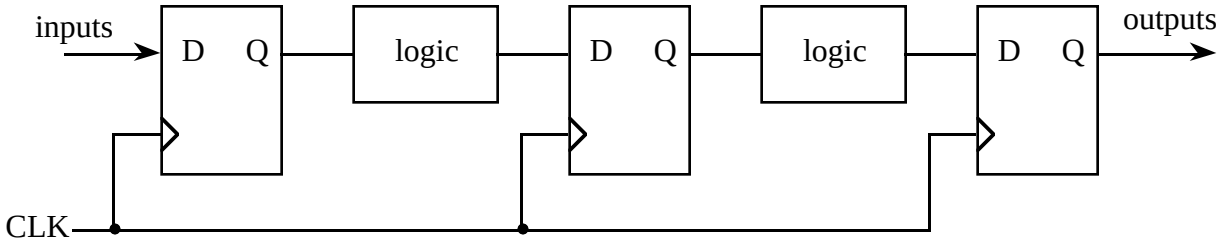
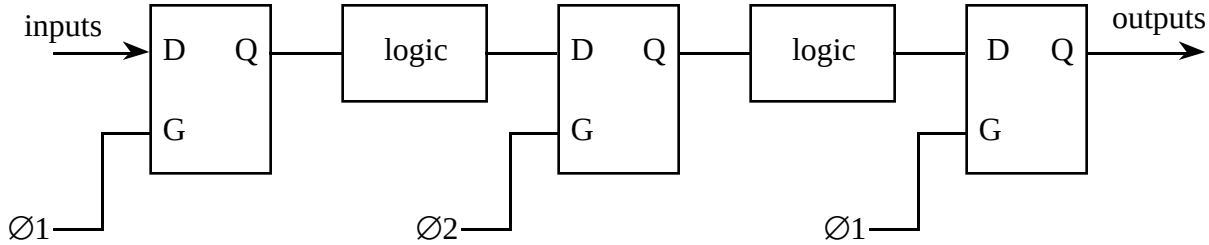
Clocked Systems



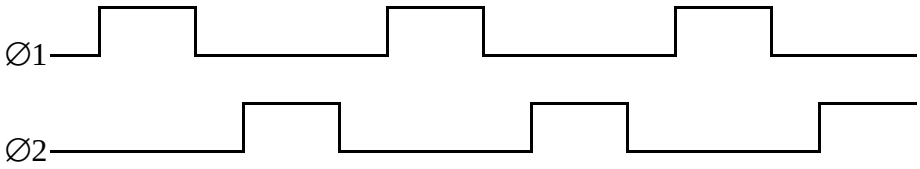
Single Phase System, edge triggered devices



Pipelined system, single phase clock, edge-triggered



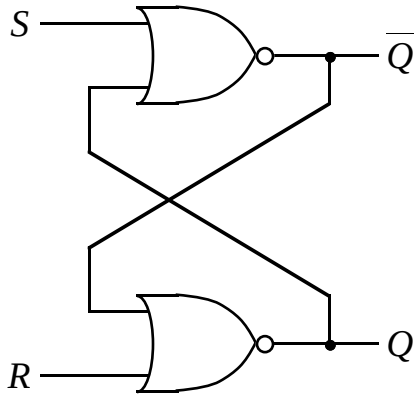
Two phase clocks, level-sensitive devices (latches)

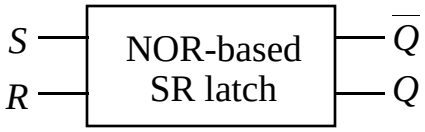


Non-overlapping clock phases

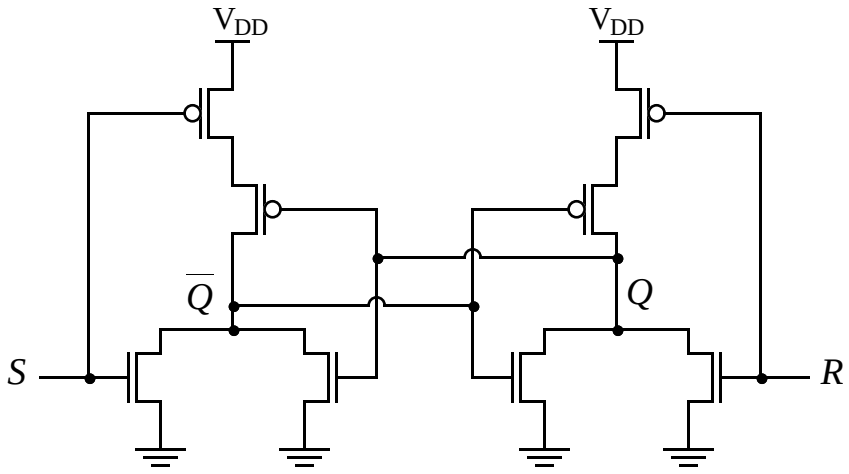
Latches (level sensitive)

S-R latch (NOR-based, non-clocked):

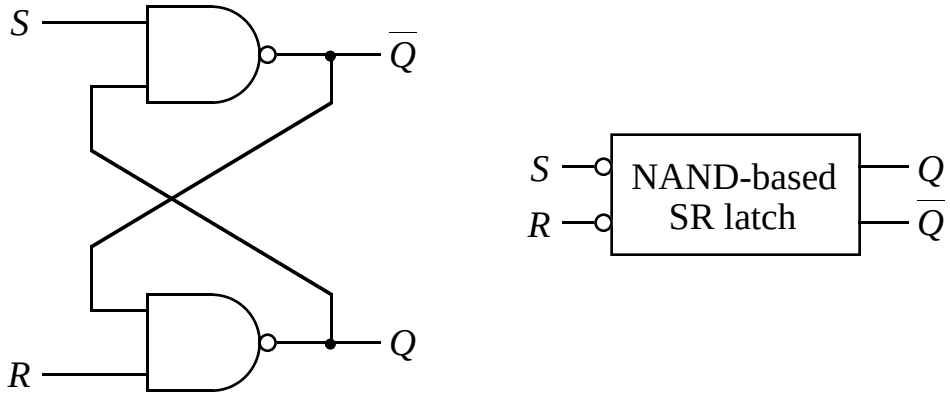




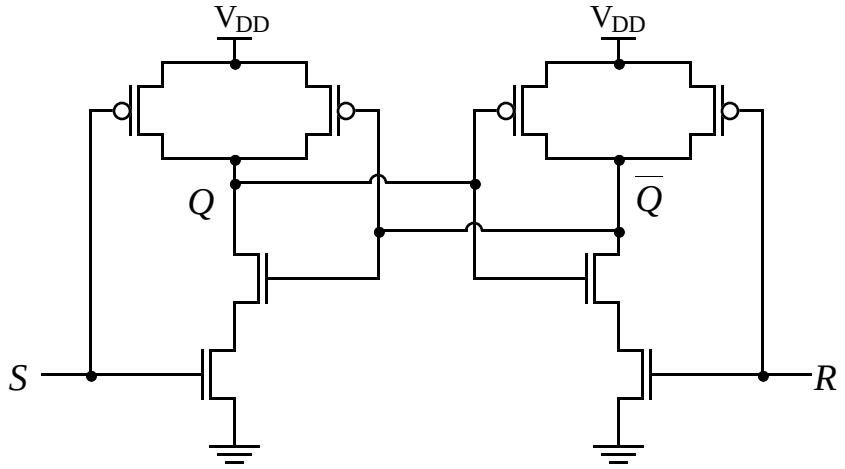
S	R	Q_{n+1}	$\overline{Q}_{n+1}$	Operation
0	0	Q_n	$\overline{Q}_n$	*hold*
1	0	1	0	*set*
0	1	0	1	*reset*
1	1	0	0	*not allowed*



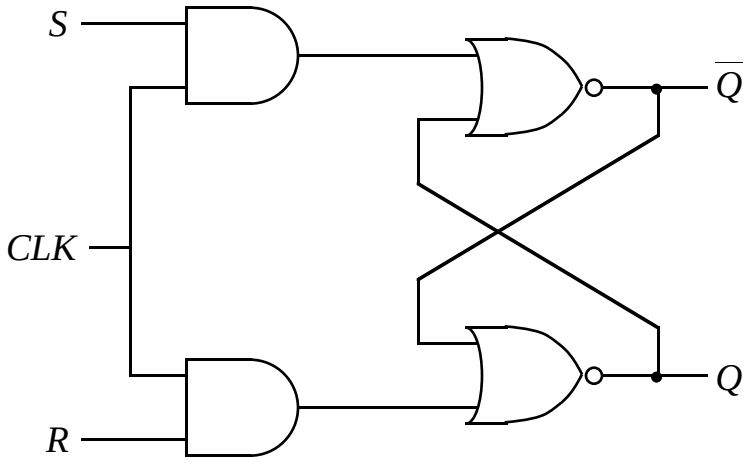
S-R latch (NAND-based, non-clocked):

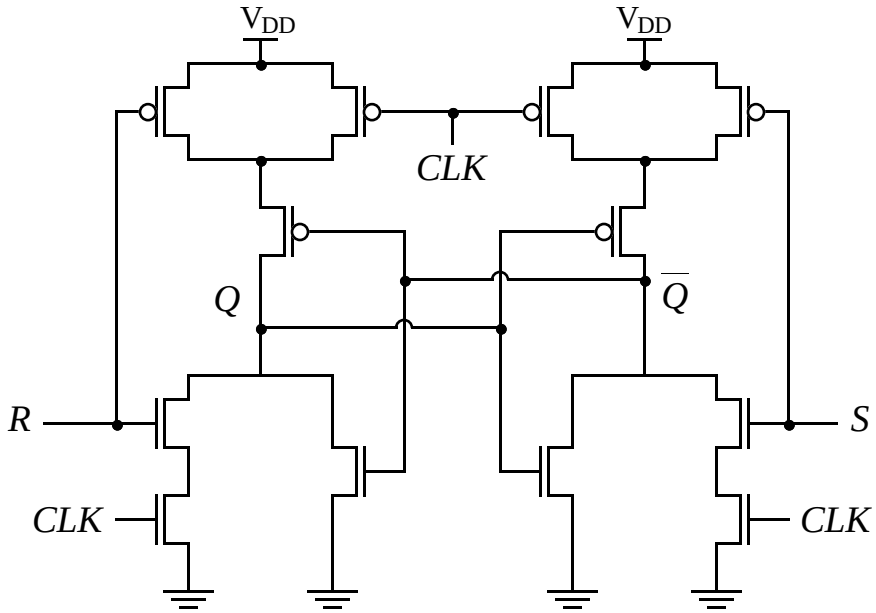


S	R	Q_{n+1}	$\overline{Q}_{n+1}$	Operation
0	0	1	1	*not allowed*
0	1	1	0	*set*
1	0	0	1	*reset*
1	1	Q_n	$\overline{Q}_n$	*hold*



Clocked SR latch





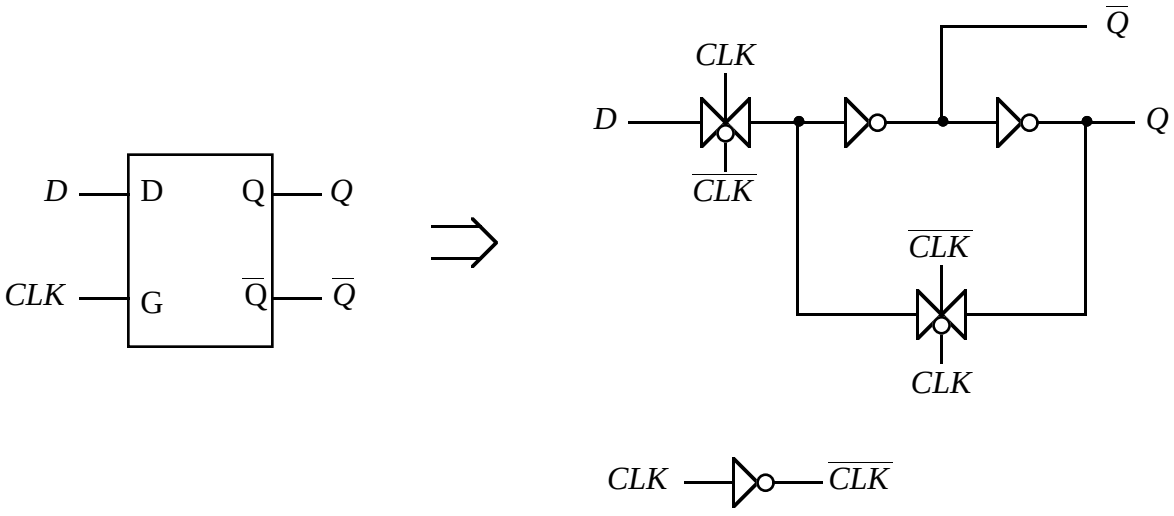
S, *R* high true; *CLK* high true

Note that the transistor implementation is <u>not</u> a direct gate level implementation of the design. Used complex gate design:

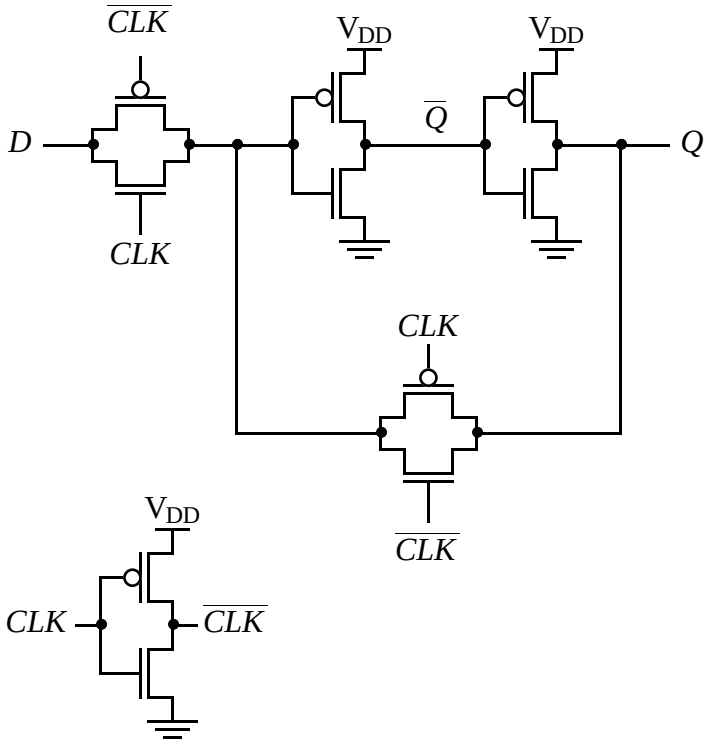
$$\overline{Q} = \overline{S \cdot CLK + Q} \qquad\qquad Q = \overline{R \cdot CLK + \overline{Q}}$$

S, *R* only affects state when *CLK* is high.

Clocked D-latch



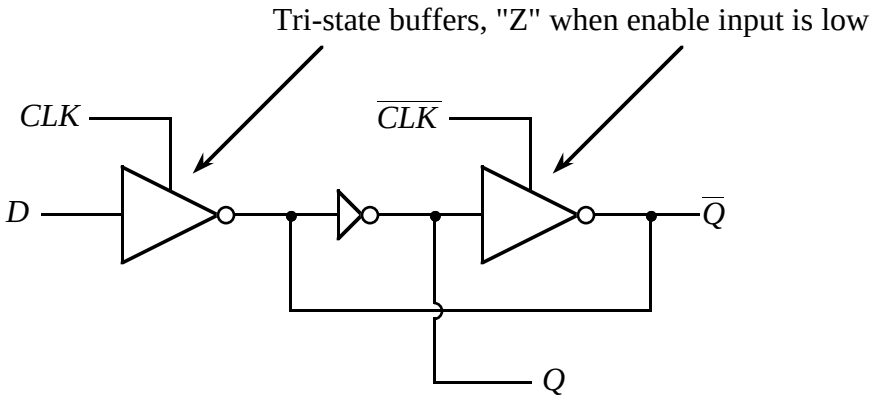
Transistor-level implementation:



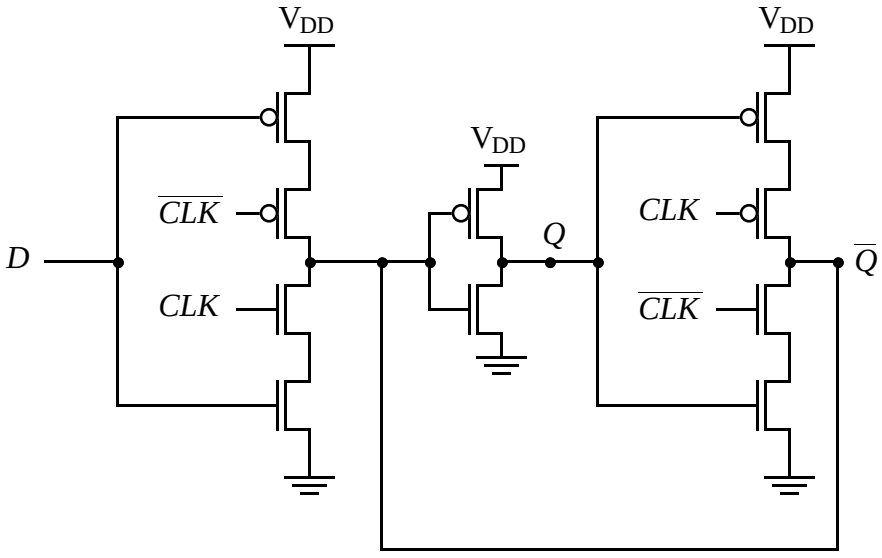
Transistor count (including clock inversion) = 10

Do local clock inversion to avoid loading of clock input

Another implementation of the clocked D-latch



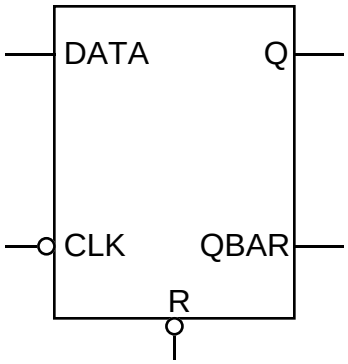
Transistor-level implementation:



D-latch with low-true clock, low-true reset

CLOCKED LATCH with RESET cell name: **CLAT**

Logic Symbol



Size

$70\lambda \times 250\lambda$

1.2μm: 42μm × 150μm

2.0μm: 56μm × 200μm

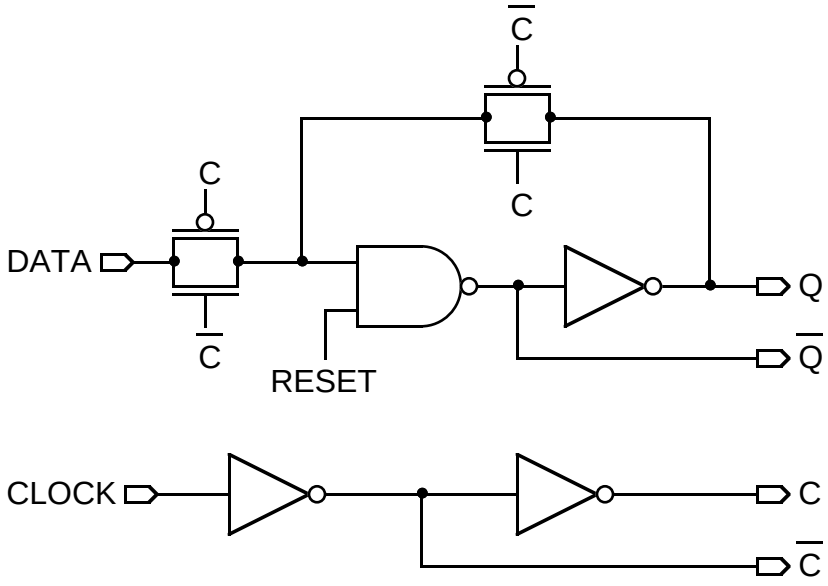
Functional Table

CLK	DATA	RST	Q	QBAR
0	*	1	DATA	$\overline{\text{DATA}}$
*	*	0	0	1
1	*	1	Q_{N-1}	$QBAR_{N-1}$
0	0	*	0	1

Input Capacitance (pF)

Signal	1.2μm
CLOCK	0.12
DATA	0.12
RESET	0.16

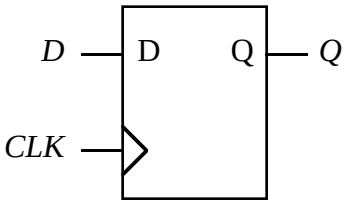
Functional Diagram



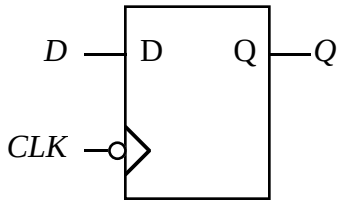
From the CMOSN library distributed by MOSIS; original library done by the National Security Agency (NSA).

D-Flip Flop

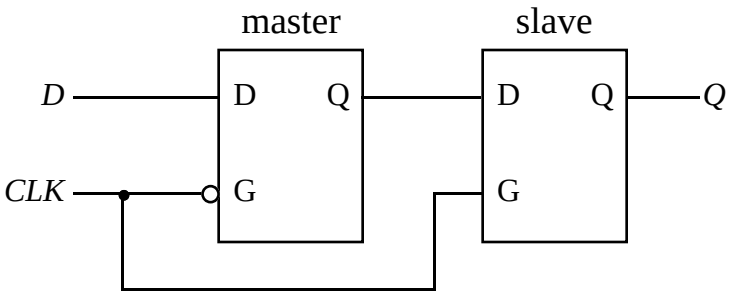
A D-flip flop is two latches in master-slave arrangement



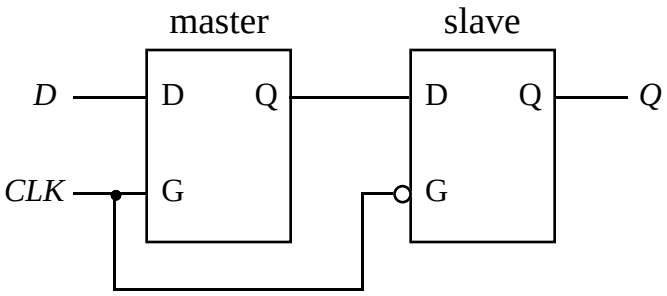
rising-edge triggered



falling-edge triggered



rising-edge triggered

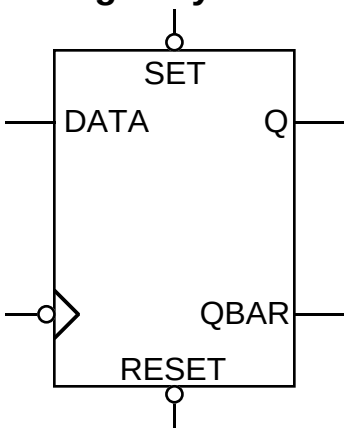


falling-edge triggered

Falling-Edge Triggered D-Flip Flop w/ asynchronous low-true set & reset

D-FLIP FLOP w/ ASY SET, RESET cell name: **DFFSR**

Logic Symbol



Size

$90\lambda \times 250\lambda$

1.2μm: 54μm × 150μm

2.0μm: 72μm × 200μm

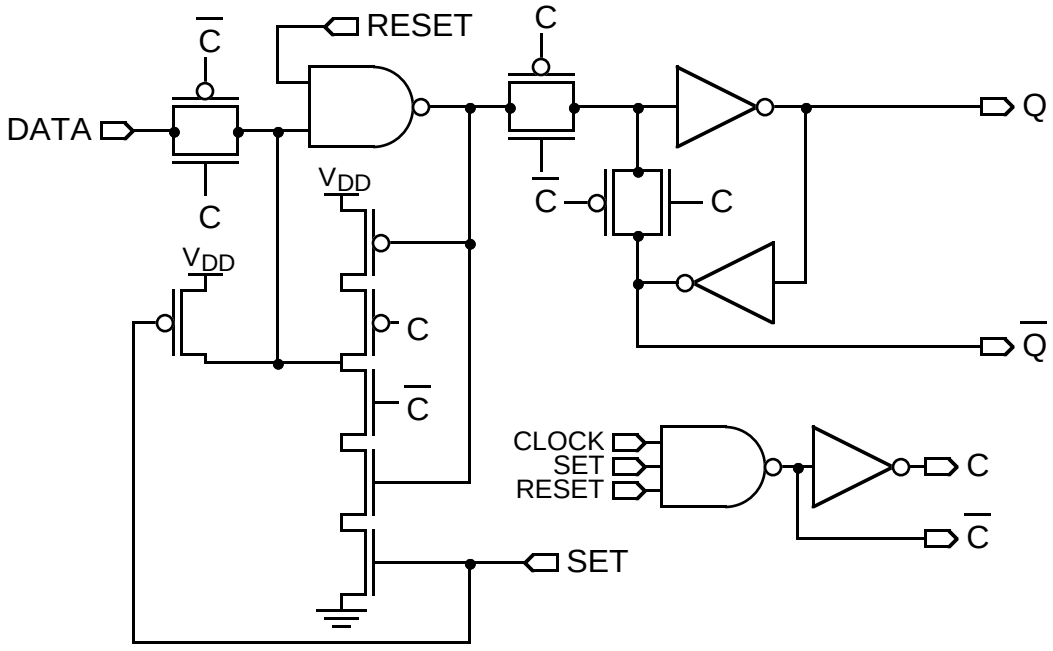
Functional Table

CLK	SET	RST	Q	QBAR
↗	1	1	Q_{N-1}	$QBAR_{N-1}$
↘	1	1	DATA	$\overline{DATA}$
*	*	0	0	1
*	0	1	1	0

Input Capacitance (pF)

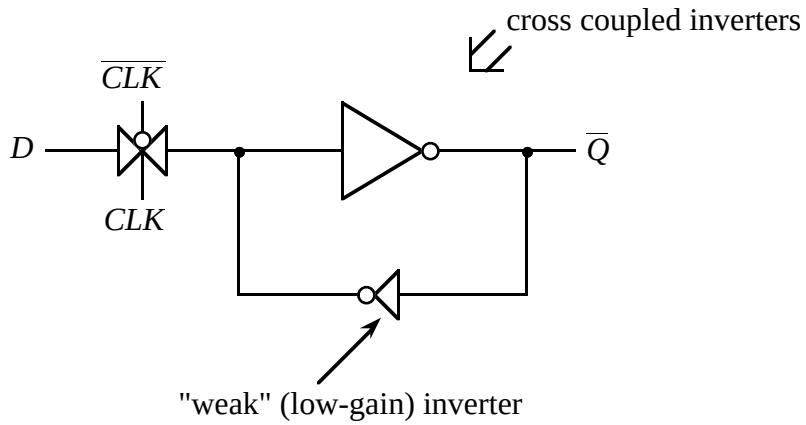
Signal	1.2μm
CLOCK	0.18
DATA	0.13
SET	0.27
RESET	0.31

Functional Diagram



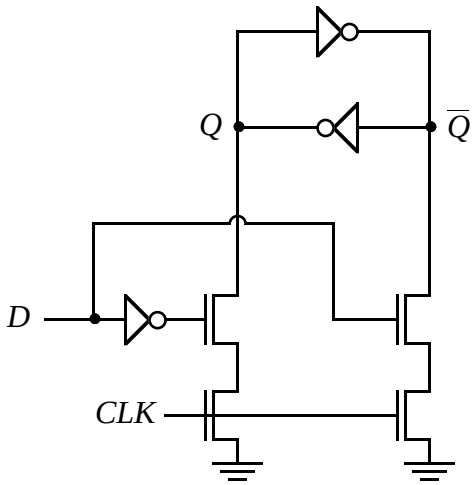
Note that SET, RESET disable the internal clock signals C, $\overline{C}$

Previous examples were "traditional" static designs. Always looking for latch/flip-flop structures which reduce the numbers of transistors.



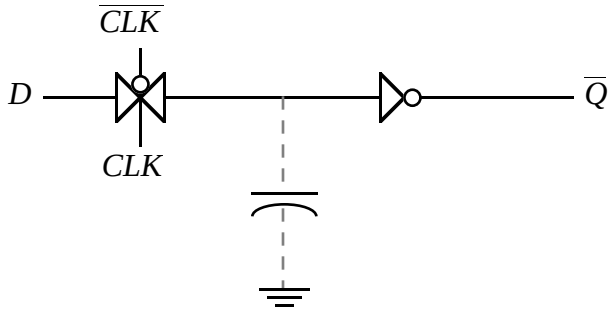
When CLK is high, D input must be able to overdrive the feedback inverter output. Use low-gain devices in feedback inverter.

Another example:



This implementation requires 10 transistors but it might actually take up less area than the traditional clocked D-latch design because it avoids pass transistors.

To eliminate more transistors, can eliminate feedback elements and create "**dynamic**" **registers**



In this case, the gate capacitance of the inverter becomes the state holding element.

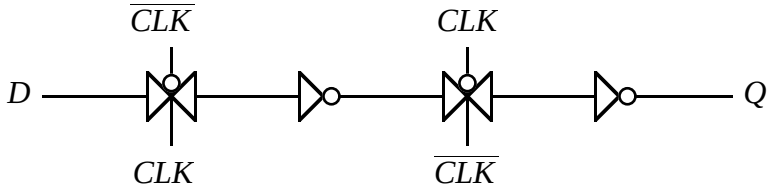
"Dynamic" because the charge on the gate capacitance will eventually leak off.

How long for leakage? For example, assuming leakage current of 1nA and storage capacitance of 20fF, the total time for 5V (i.e., a 5V logic level) to "leak" off is

$$C \times \frac{\Delta V}{\Delta i} = \frac{20 \times 10^{-15} \times 5}{1 \times 10^{-9}} = 100\mu s$$

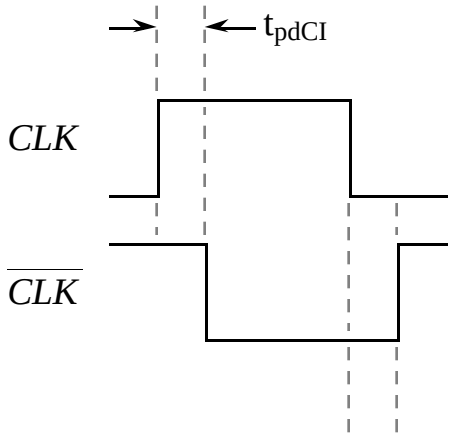
Hence, after approximately 100μs, the 20fF capacitor would be completely discharged to 0V.

A dynamic D-flip flop (falling-edge triggered)

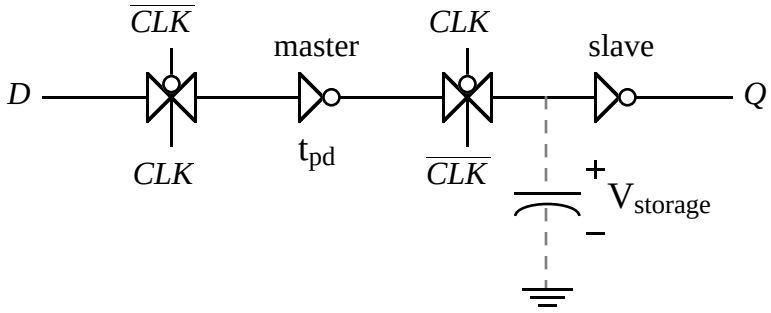


Really need *CLK* and $\overline{CLK}$ to be non-overlapping. If *CLK* and $\overline{CLK}$ overlapped, then a race condition could occur because there would be direct path from *D* to *Q*, particularly if the overlap period was large.

The propagation delay, t_{pdCI}, of the inverter which does the clock inversion = the overtime ($t_{overlap}$) in which a race condition could occur.



With high frequency clocks, this overlap period can be a problem

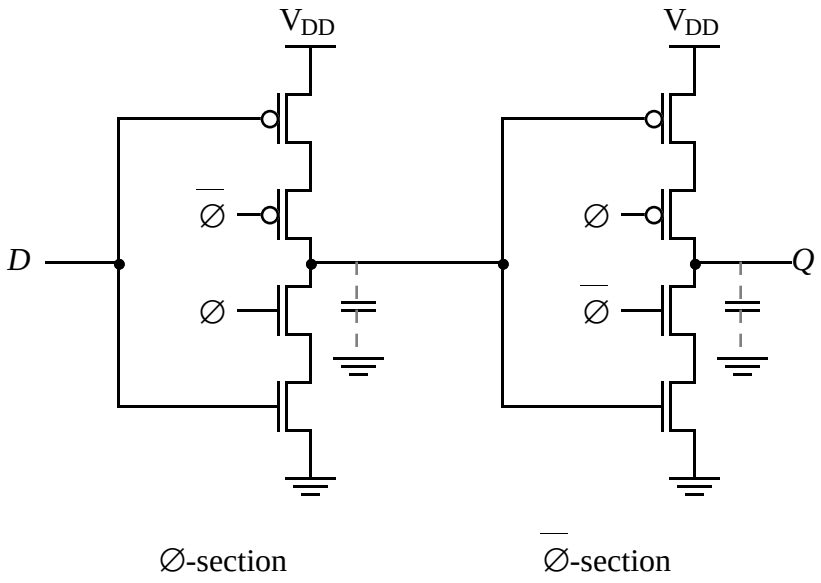


If $t_{pd} < t_{overlap}$, then when $CLK = 0 \rightarrow 1$ then $\overline{CLK} = 1$ to 0 **after** $t_{overlap}$ and the $V_{storage}$ value will get set equal to *D*!!!

We only want $V_{storage}$ to be set equal to *D* value when $CLK = 1 \rightarrow 0$ (falling edge)

Race condition exists during 1 — 1 overlap condition, *D* feeds thru to *Q*

A better dynamic latch - **C^{2}MOS dynamic register**



Insensitive to overlap (proved later)

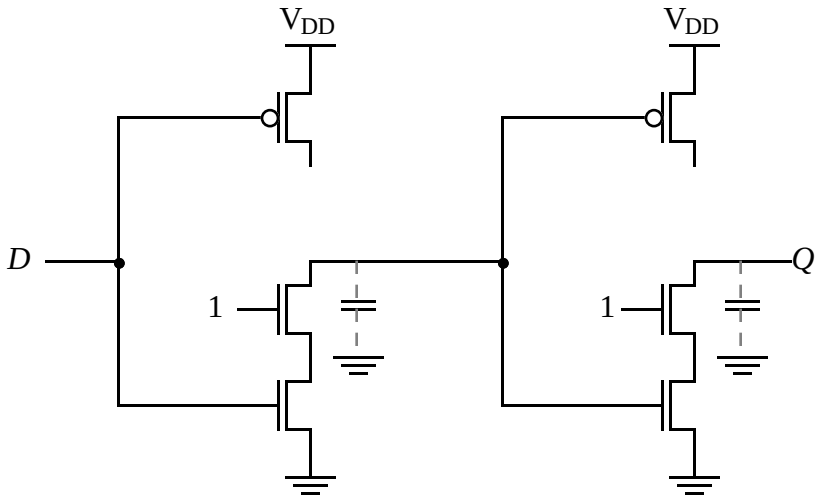
Basic operation:

1) $\varnothing = 1$, ($\overline{\varnothing} = 0$)
   $\varnothing$-section in evaluation mode, $\overline{\varnothing}$-section in hold mode

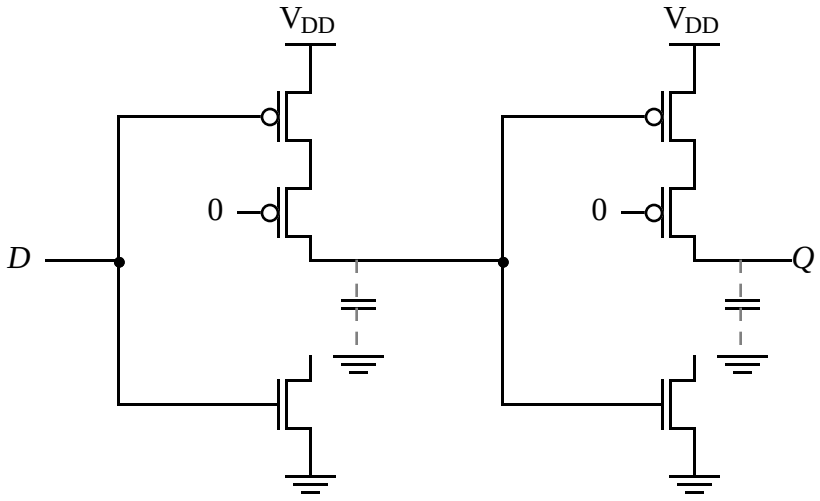
2) $\varnothing = 0$, ($\overline{\varnothing} = 1$)
   roles now reversed, $\varnothing$-section in hold mode, $\overline{\varnothing}$-section in evaluation mode

Why is the C^{2}MOS dynamic register insensitive to overlap?

During overlap, want to make sure that there is no possibility of a race condition in which *D* feeds directly thru to *Q*



(1 - 1) overlap

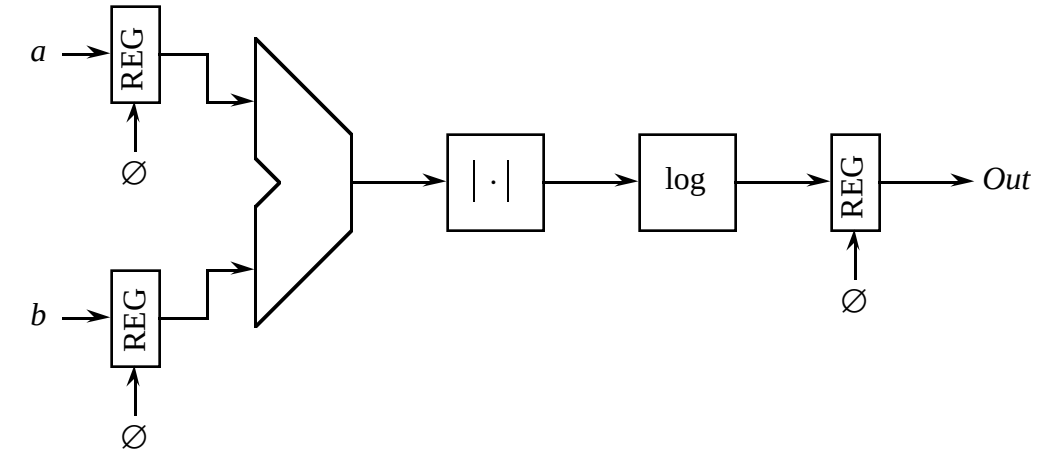


(0 - 0) overlap

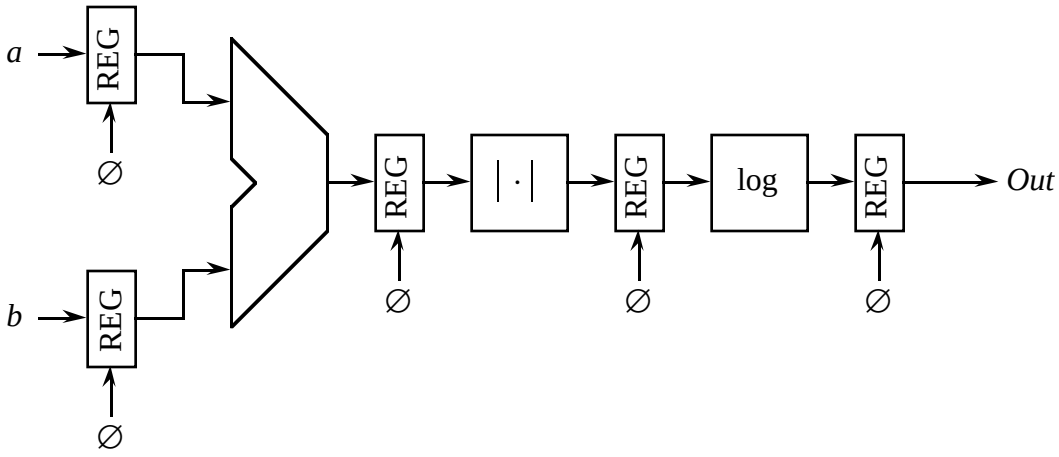
No feed thru path exists for either 1 - 1 case or 0 - 0 case.

Want to use **dynamic latches** to form **fast pipelined circuits**

Consider the datapath for computing *log(|a + b|)*:



Nonpipelined version

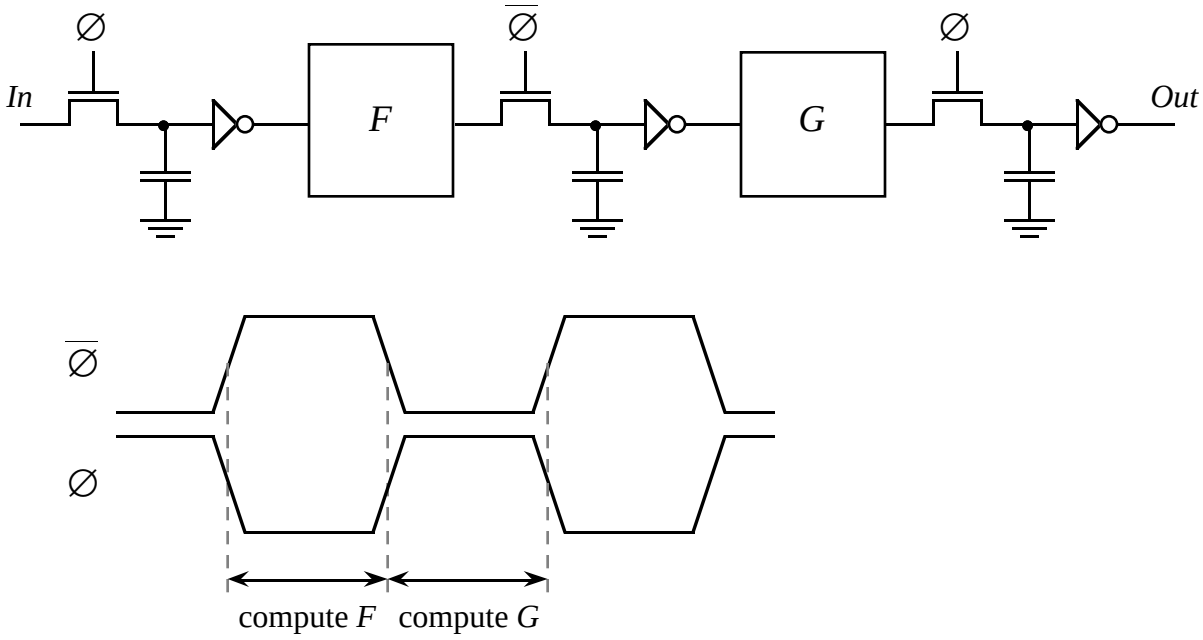


Pipelined version

Clock period $T_{min} = t_{clk\text{-}out\ (register)} + t_{pd\ logic\ block} + t_{setup\ register}$

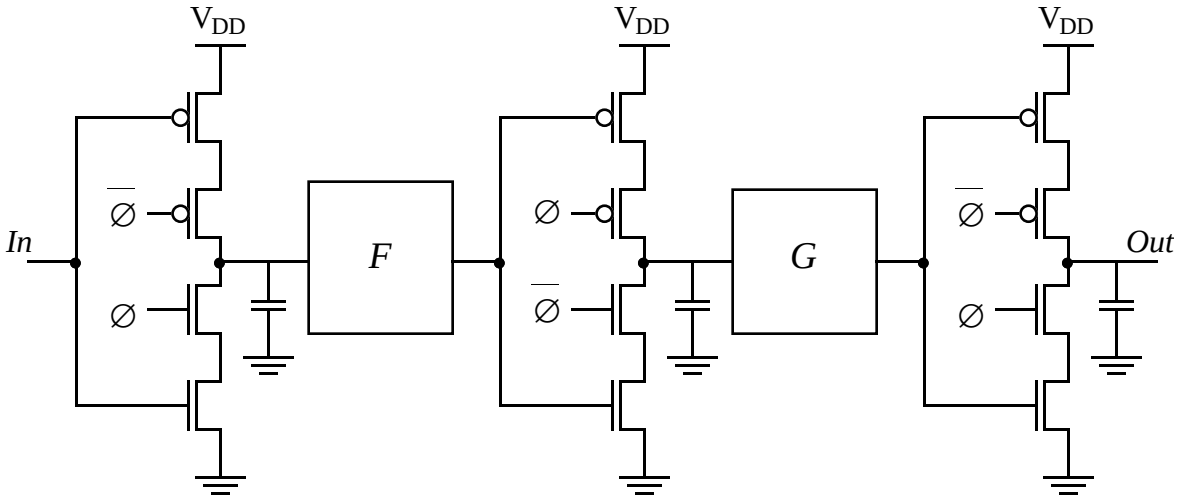
Minimize $t_{clk\text{-}out}$, t_{setup}

Pipelined System with Dynamic Latches



Suffers from clock overlap problem

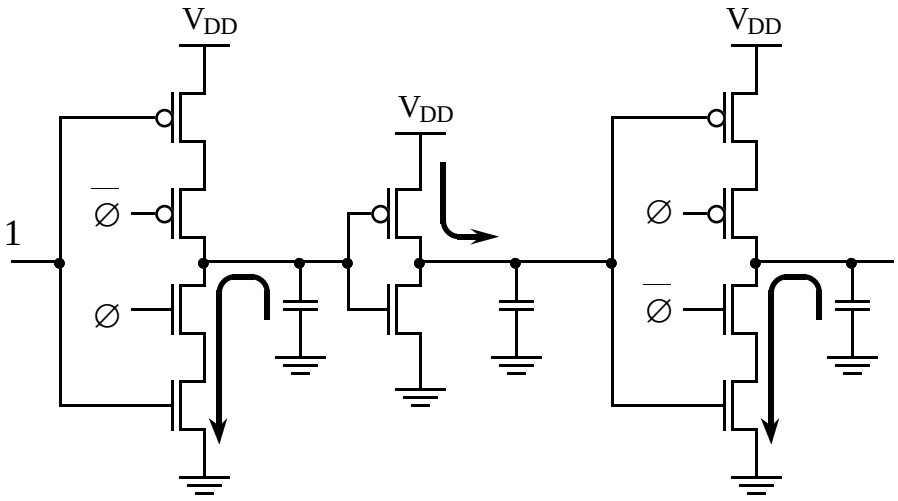
Try C^{2}MOS latches



A C^{2}MOS-based pipelined circuit is race-free as long as all the logic functions *F* (implemented with static logic) between the latches are non-inverting! Why?

Use CAD tools to ensure this.

Here's a potential race condition during (1-1) overlap in C^2MOS-based design:



The above circuit would require sharp clock edges for correct operation.