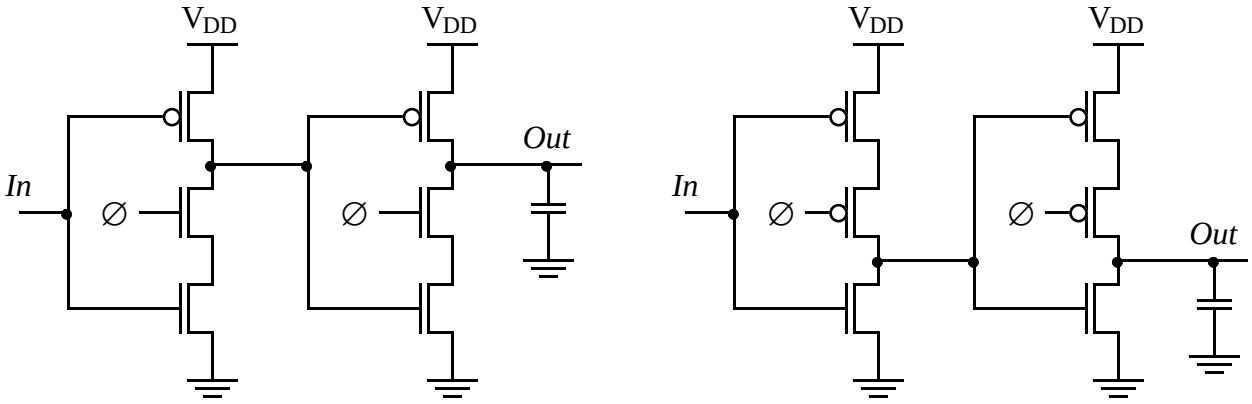
What we <u>really</u> want is **True Single-Phase Clocked (TSPC) Logic** in which we only have <u>one</u> <u>clock</u>, and <u>do</u> <u>not</u> need an inverted clock.

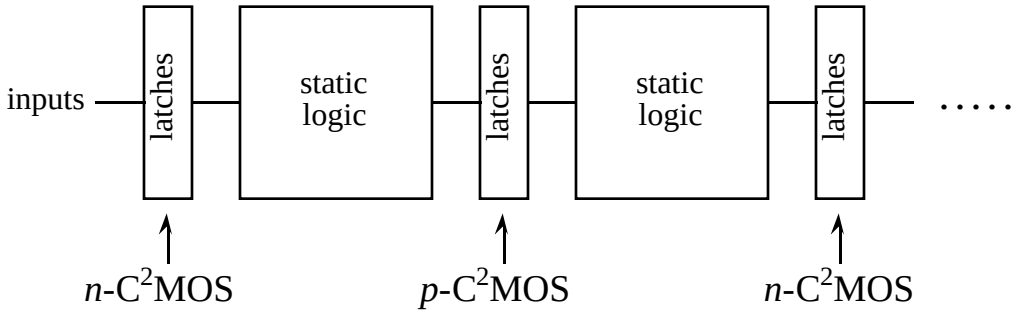
Redesign C^{2}MOS latch and create the **doubled *n*-C^{2}MOS latch** and the **doubled *p*-C^{2}MOS latch**



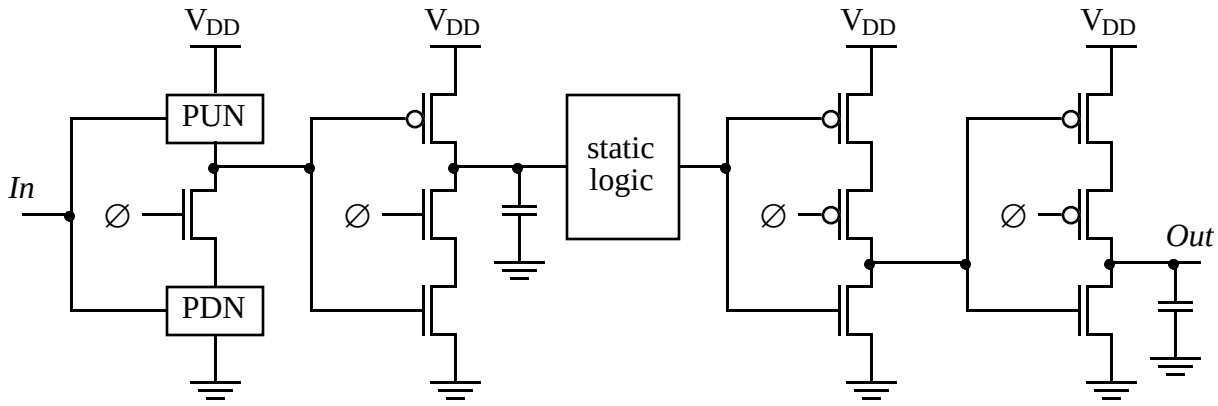
doubled *n*-C^{2}MOS latch doubled *p*-C^{2}MOS latch

Doubling of the latch ensures that signal cannot propagate from input to output when latch is in <u>hold</u> mode (*n*-C^{2}MOS latch ∅ = 0, *p*-C^{2}MOS latch ∅ = 1)

How do I make a pipelined system using these latches?



Can also include logic into the TSPC latches!



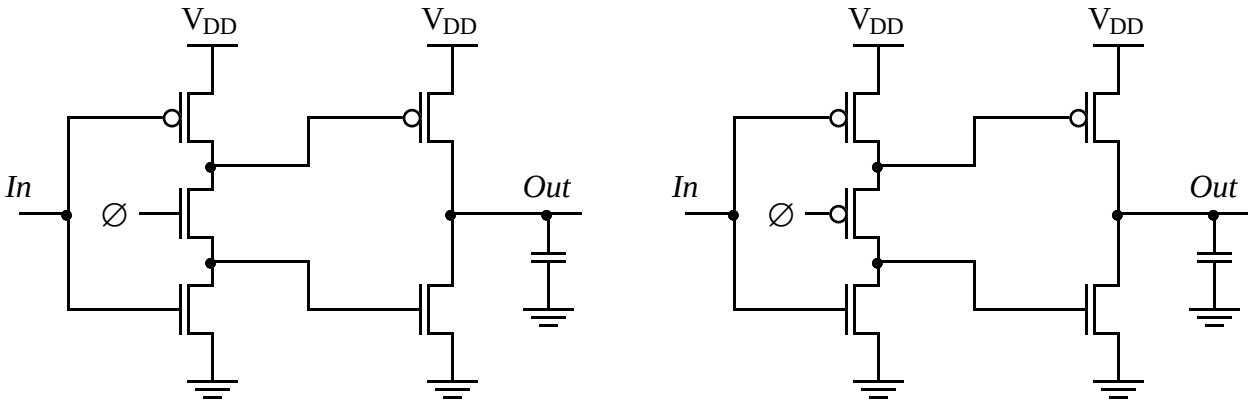
Inserting logic into the latch Inserting logic between latches

PUN - pullup network, PDN - pulldown network, logic directly implemented in the latch.

No requirements on inversion of static logic between latches.

This design method was the style of choice for the (original) DEC Alpha microprocessor, which ran at 200MHz in 0.75μm CMOS technology.

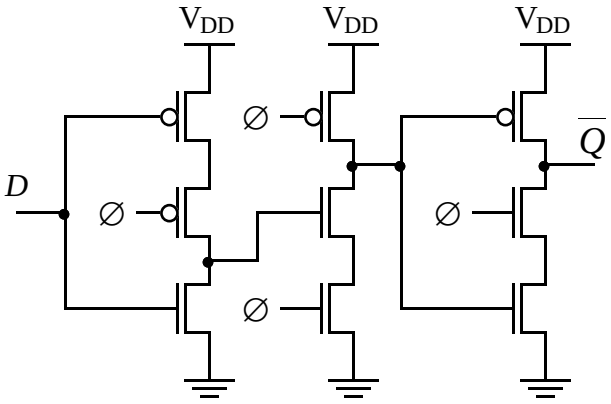
Other improvements - **Simplified TSPC latch** (split output latch)



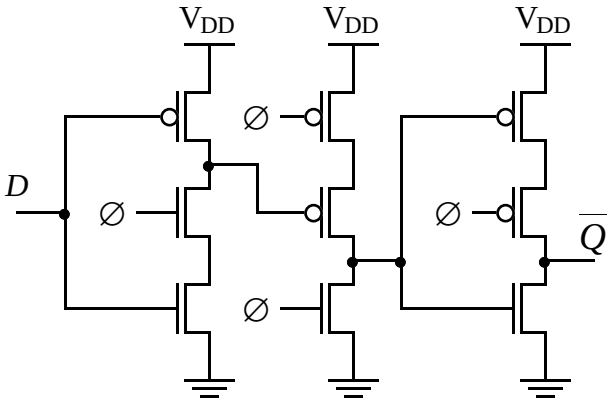
∅-latch $\overline{\text{∅}}$-section

Reduces transistor count by two, cuts clock load in half at the cost of reduced voltage swings on internal nodes (lowers the noise margin)

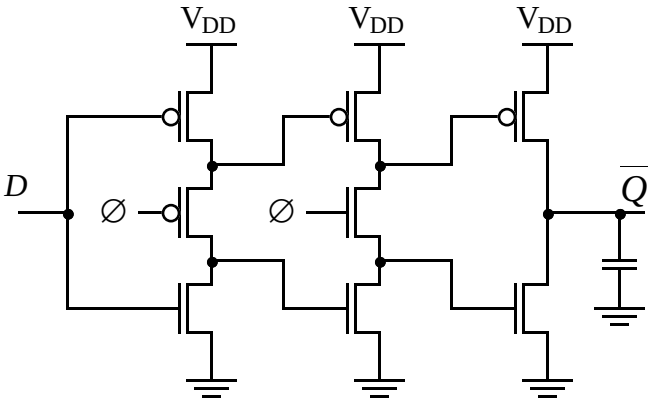
Creating D-flip flops out of TSPC latches



Positive edge-triggered D flip-flop



Negative edge-triggered D flip-flop



Positive edge-triggered D flip-flop using split-output latches

Dynamic latches (and dynamic logic, to be discussed later) is currently the principle design style for high speed digital circuits

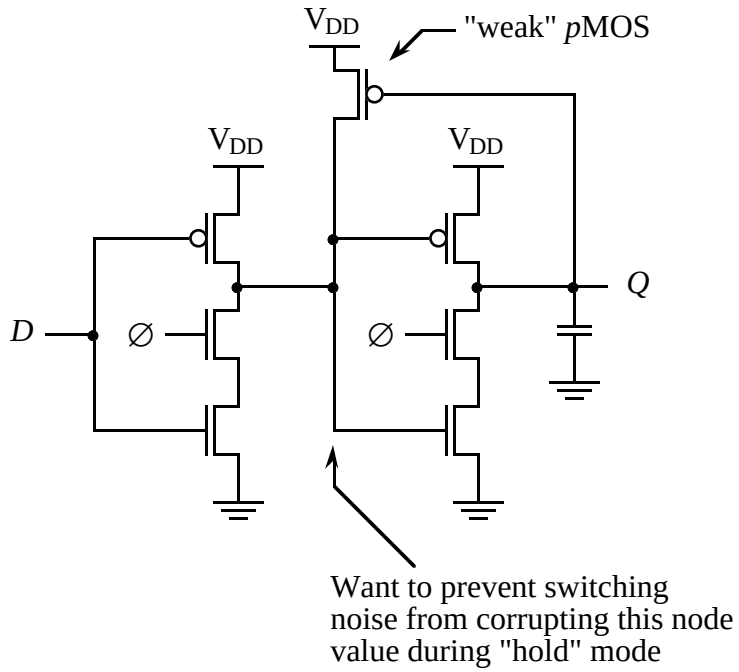
Question: In CMOS, with edge-triggered storage elements, would we want negative edge-triggered or positive edge-triggered?

Typically, want the clock edges as sharp as possible. Because *n*MOS devices provide stronger pulldown, usually use negative edge-triggered devices in CMOS.

The C^2MOS latch, doubled *n*-C^2MOS, and doubled *p*-C^2MOS latches all require sharp clock edges for correct operation.

In the DEC Alpha case, clock rise and fall times below 0.8ns caused no failures while above 1.0ns caused failures. A value of 0.5ns was set as the target for clock rise/fall.

To reduce noise susceptibility, can also add a feedback transistor:



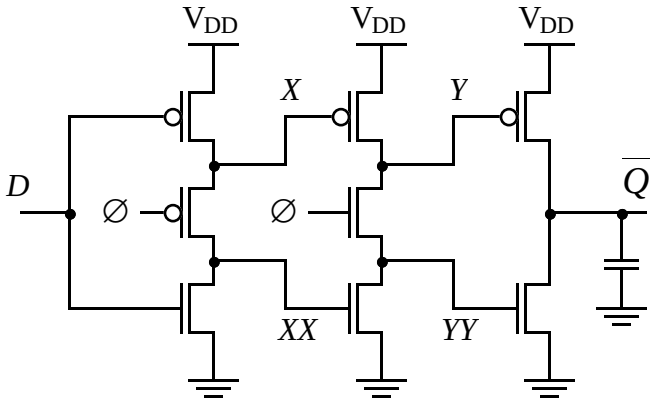
Feedback transistor sharpens output edges, reduces susceptibility to noise

When *Q* goes $1 \rightarrow 0$, feedback sharpens (speeds up) transistion

When $Q = 0$, helps hold 0 value by keeping *p*-device on (internal node at V_{DD})

When *Q* goes $0 \rightarrow 1$, feedback transistor will act against this transistion but will be overdriven by stronger pulldown in front stage

How to add reset (asynchronous), low true



Positive edge-triggered TSPC Split Ouput D-flip flop
(revisited for this example)

When clock = 0, $X = XX = \overline{D}$, $YY = 0$ or YY_{old}, $Y = 1$ or Y_{old}.

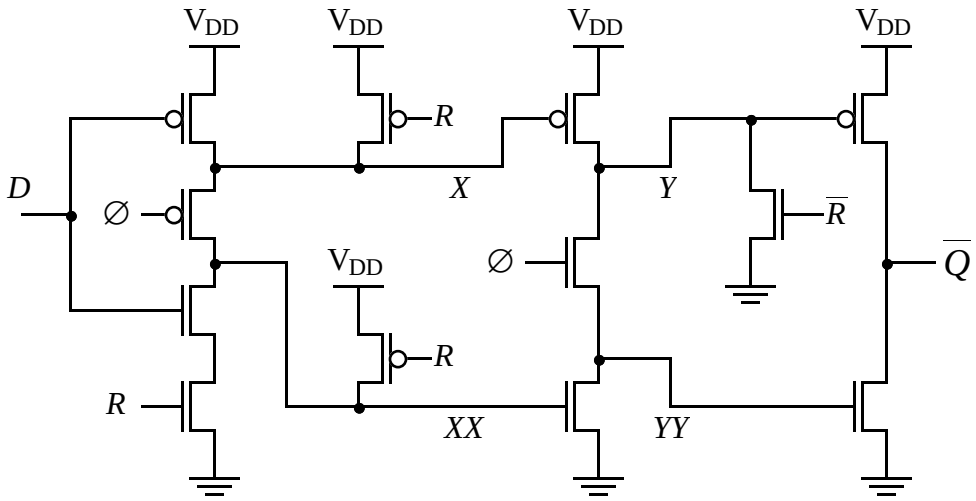
For **reset** we want $YY = YY_{\text{old}} = 0, \rightarrow \overline{Q} = 1$

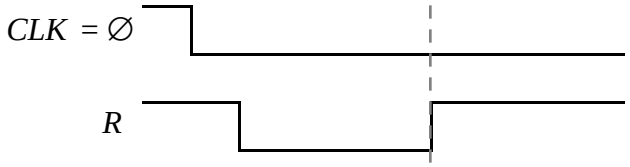
clock	D	$X = XX$	Y	YY	$\overline{Q}$
0	0	1	$Y_{\text{old}} = 0$	0	1
			$= 1$	0	$\overline{Q}_{\text{old}}$
0	1	0	$Y = 1$	$YY_{\text{old}} = 0$	$\overline{Q}_{\text{old}}$
				$YY_{\text{old}} = 1$	0

With clock = 0, to reset correctly must affect both Y or YY.

Similarly, with clock = 1, a reset operation must affect both X or XX.

Split Output Reset





$Y = 0$
$YY = 0$
$\overline{Q} = 1$
$X = 1$
$XX = 1$

$XX = X = 1$ or 0
$Y = 1$ or $Y_{old} = 0$
$YY = 0$ or $YY_{old} = 0$
$\overline{Q} = 1$ or $\overline{Q}_{old} = 1$

$XX = 1$
$X = 1$
$Y = YY = 0$
$\overline{Q} = 1$

$X = 1$ or $X_{old} = 1$
$XX = 1$ or 0
$Y = YY = 0$ or $Y_{old} = YY_{old} = 0$
$\overline{Q} = 1$

Final transistor count = 12. Recall that a static *D*-flip flop can require 33 transistors!

For **practice**, <u>everybody</u> is to add asynchronous , low true reset to the other two TPSC *D*-flip flops (you are doing one of these in lab anyway).