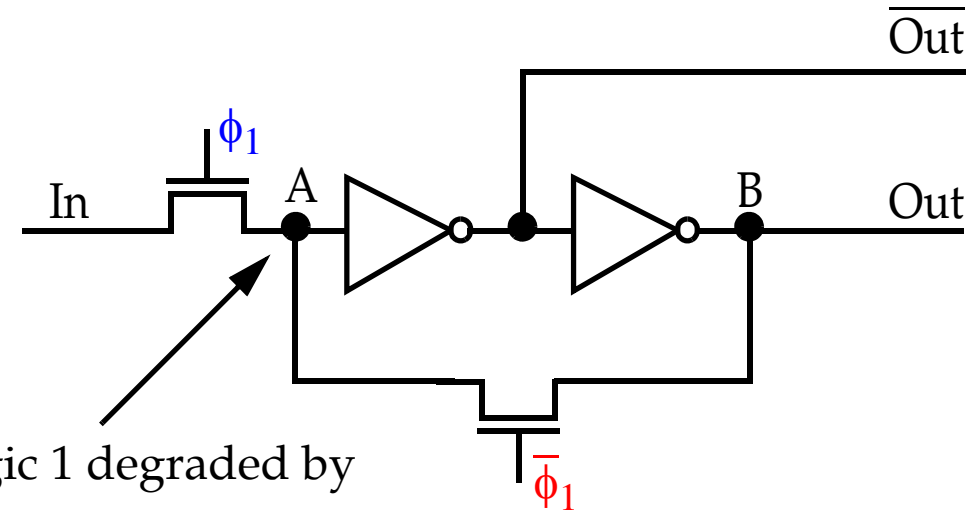
Smaller Static Flip-Flops

Positive feedback is not the only means to implement a memory function.

A **capacitor** can act as a memory element as well.

In this case, a *periodic refresh* is required (in the millisecond range) due to leakage (hence the word **dynamic**).

Consider the following "cheaper" (1/2 transmission gate) *positive level-sensitive* static latch as a step toward deriving a dynamic FF:



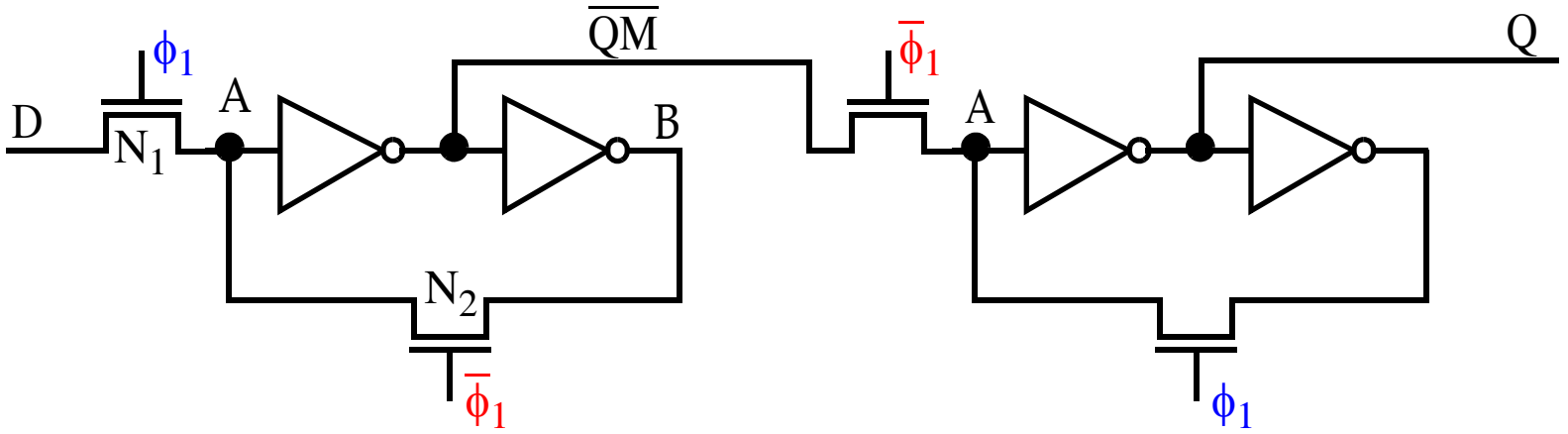
Logic 1 degraded by V_t.

Static as long as ϕ_1 is kept low.

When ϕ_1 is high, *In* is sampled and stored on internal capacitors.

Smaller Static Flip-Flops

A master-slave FF is created by cascading two of these latches and reversing the clocks.

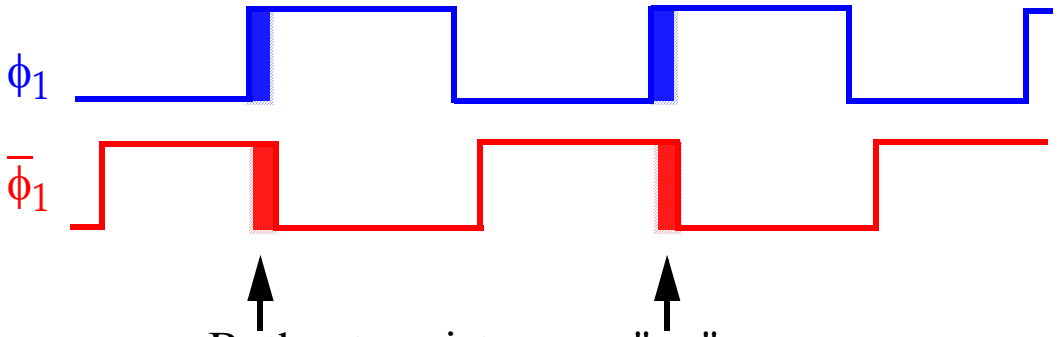


The problem with this latch is that ϕ_1 and $\overline{\phi}_1$ might overlap, which may cause two types of failures:

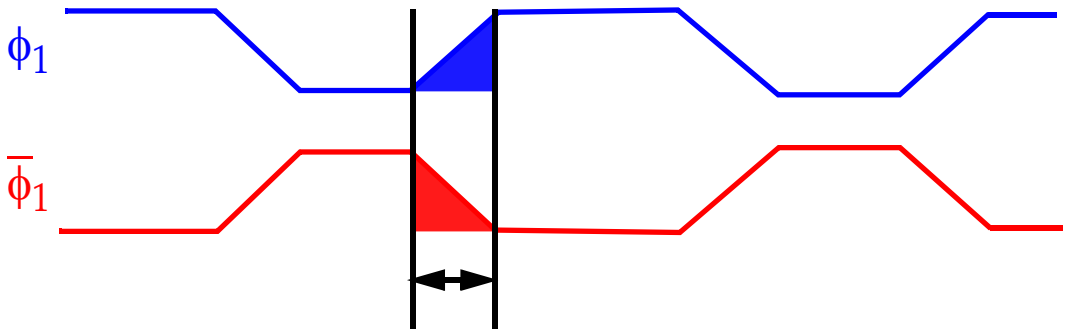
- Node A can become undefined as it is driven by both *D* and *B* when ϕ_1 and $\overline{\phi}_1$ are both high.
- *D* can propagate through both the master and slave if both ϕ_1 and $\overline{\phi}_1$ are high simultaneously for a long enough period (race condition).

Single Phase Clock Skew/Slew

Clock skew causes conflicts and transparency.



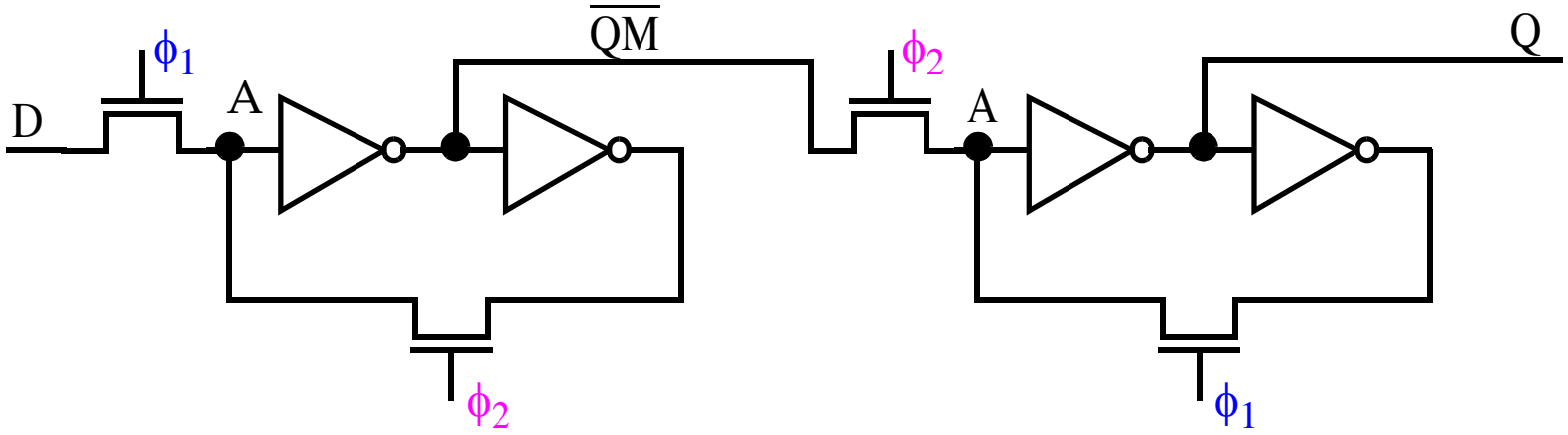
Clock slew (slow rise and fall times) can also cause transparency:



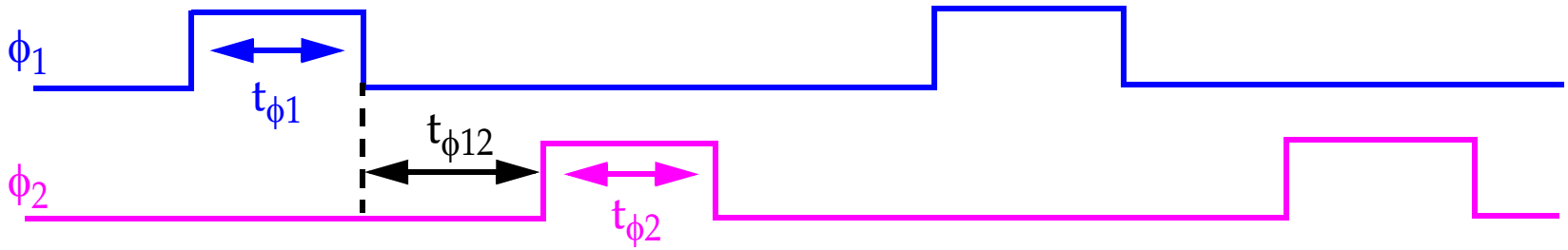
Clock skew is a dominant problem in current high performance designs.

Pseudo-Static Two-Phase Flip-Flops

The fix is to use two non-overlapping clocks ϕ_1 and ϕ_2:



A large $t_{\phi-12}$ allows proper operation even in the presence of clock skew.

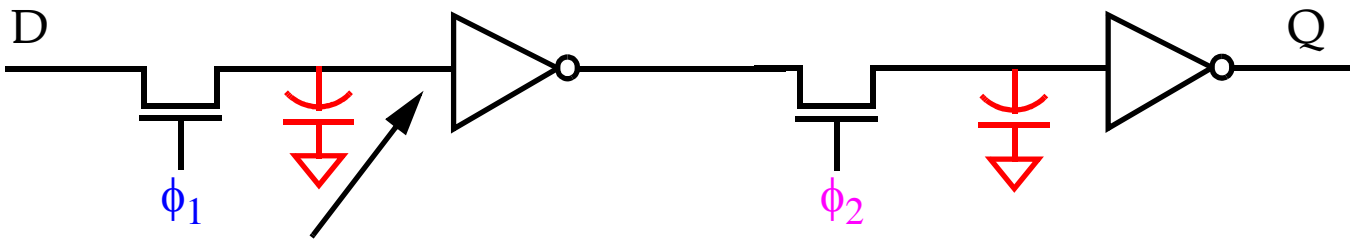


Note that node *A* floats (dynamic) during the time period $t_{\phi-12}$ but is driven during $t_{\phi-1}$ and $t_{\phi-2}$ (static).

Hence, the name *pseudostatic*.

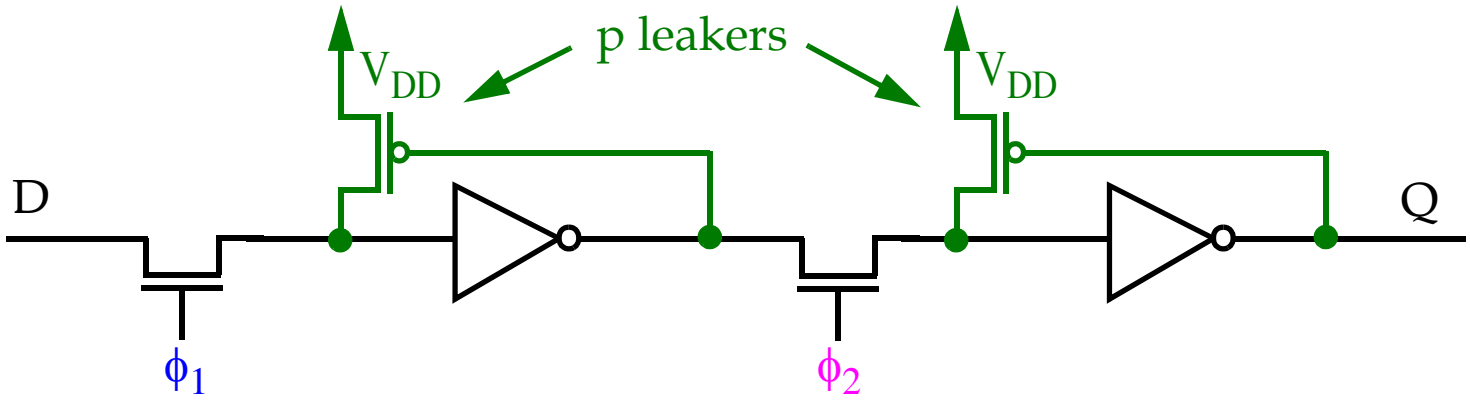
CMOS Dynamic Two-Phase Flip-Flops

This version is simplier (6 trans) and is often used in pipelined datapaths for microprocessors and signal processors.



Degraded '1' values may increase static current if below V_{tp}.

Disadv: **2 non-overlapping clocks** required (4 if transmission gates are used).



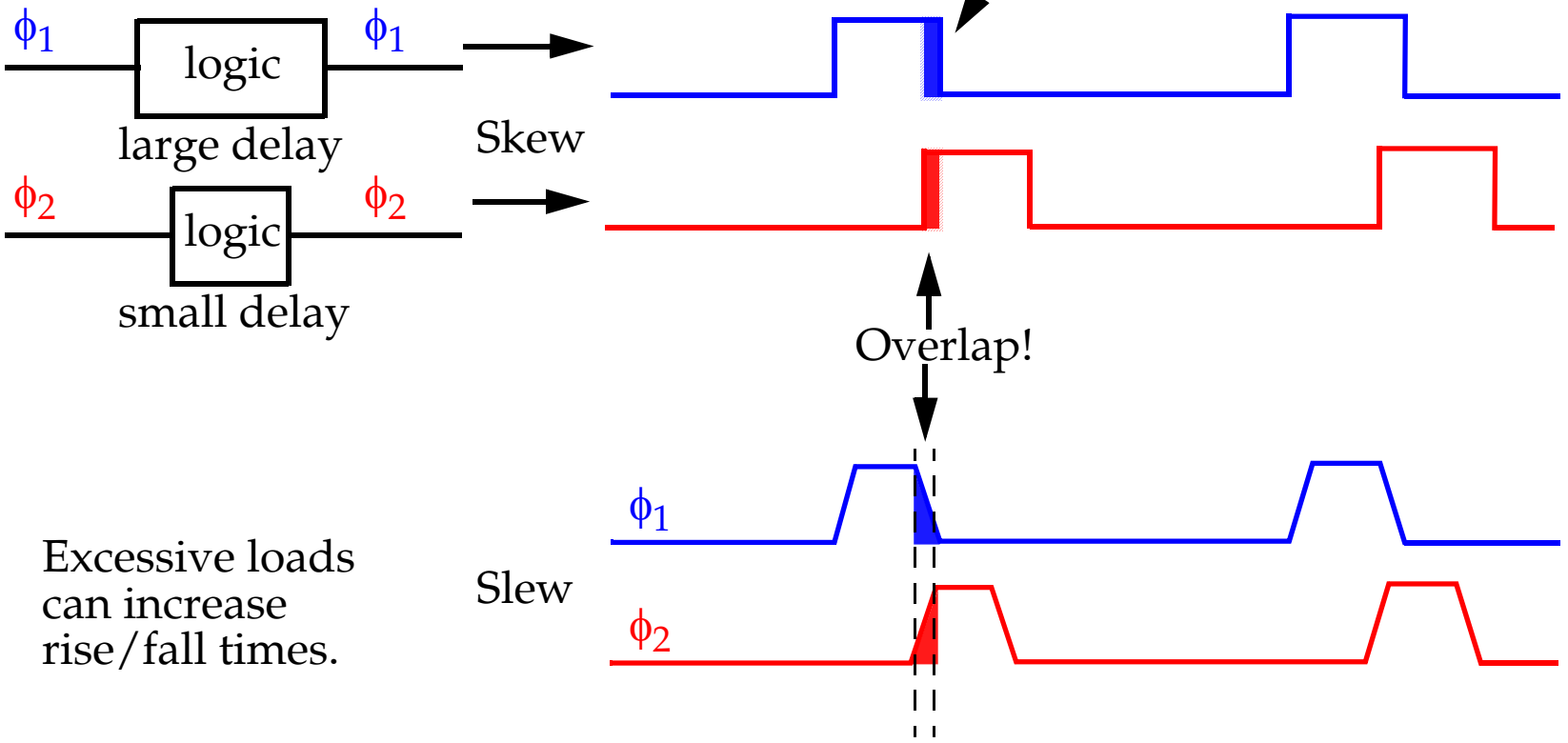
p leakers provide fully restored logic levels.

These implementations MUST be simulated at all process corners (under worst-case conditions).

Two-Phase Clocking

Clock skew/slew:

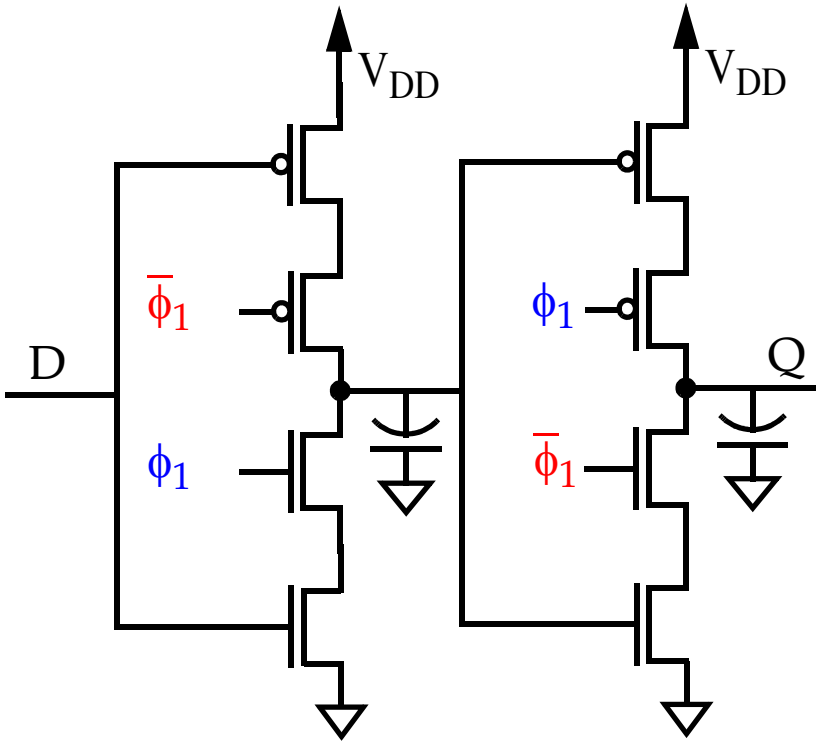
Nonoverlapping clocks:

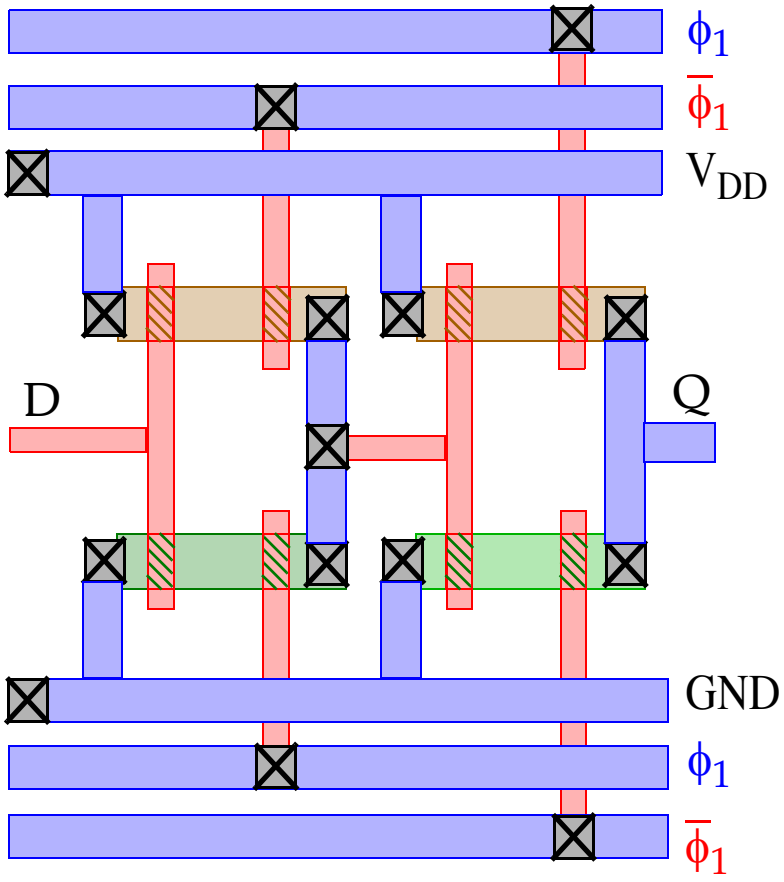


Excessive loads can increase rise/fall times.

C^2MOS Register

C^2MOS: A clever method which is *insensitive* to clock skew:



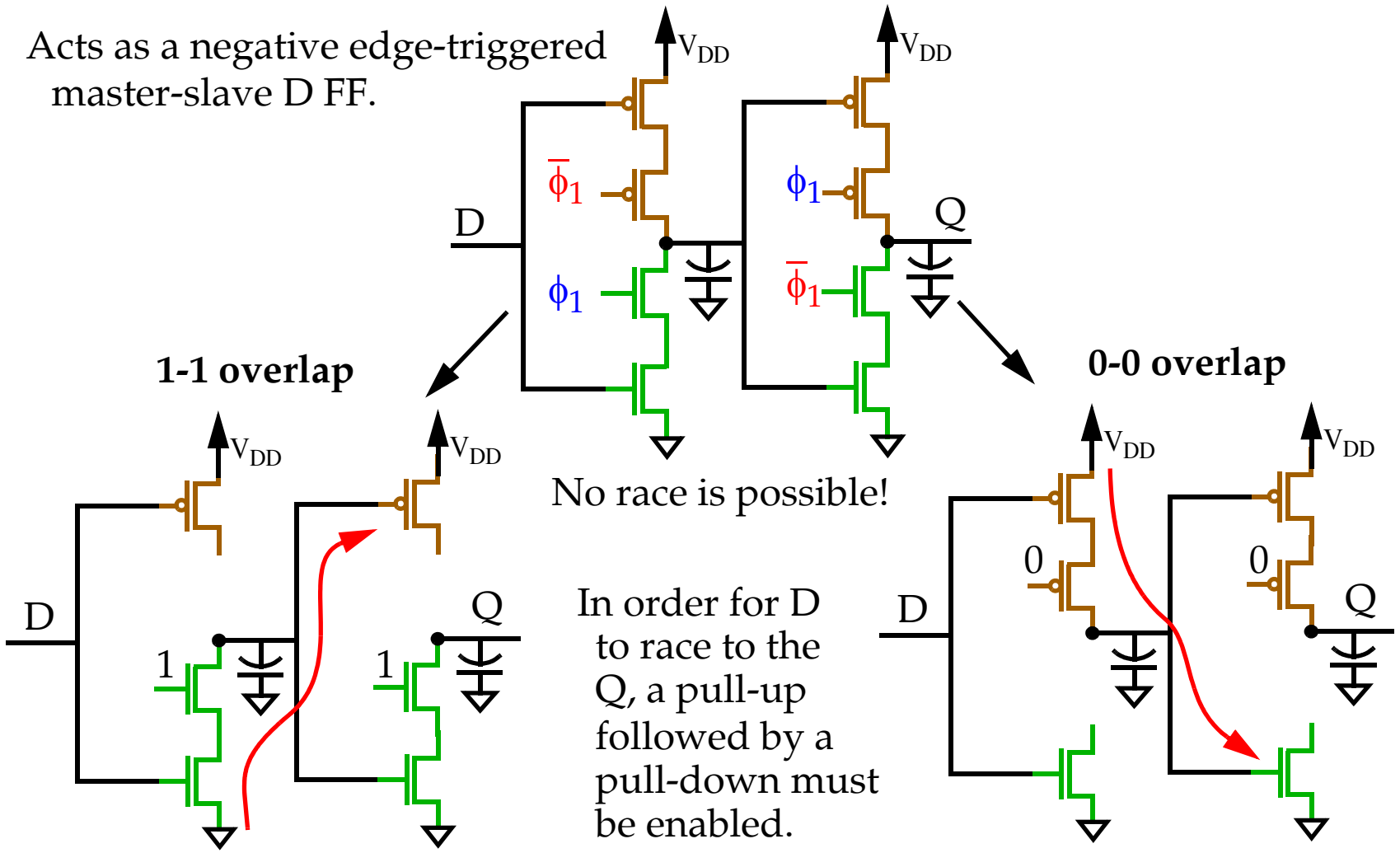


Note: Dual phase version is identical except ϕ_2 and $\overline{\phi}_2$ are used to drive the n/p-trans in the right inverter.

C^2MOS Register

C^2MOS is *insensitive* to overlap as long as the rise and fall times of the clk edges (clock slew) are sufficiently small:

Acts as a negative edge-triggered master-slave D FF.



In order for D to race to the Q, a pull-up followed by a pull-down must be enabled.

C^2MOS Register

Races are just not possible since the overlaps activate either the pull-up or the pull-down networks but never both simultaneously.

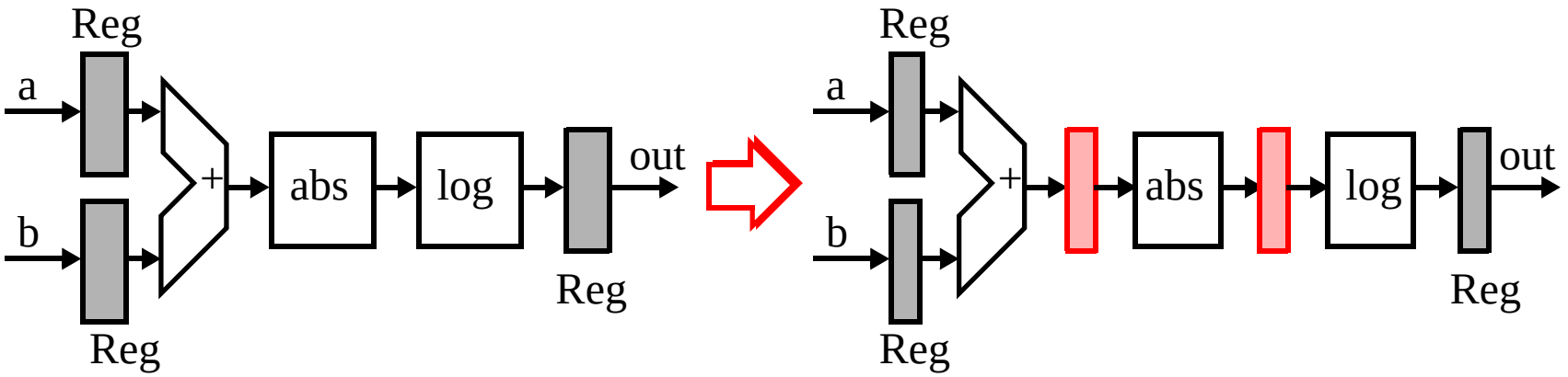
The inverters force 0-1 and 1-0 propagation modes only.

However, if the rise and fall times of the clock are slow, there exists a time slot in which *both* n- and p-transistors are conducting simultaneously.

Correct operation requires the clock rise/fall times be smaller than about **5 times** the propagation delay through the FF.

This is not hard to meet in practical designs, making C^2MOS especially attractive in high speed designs where avoiding clock overlap is hard.

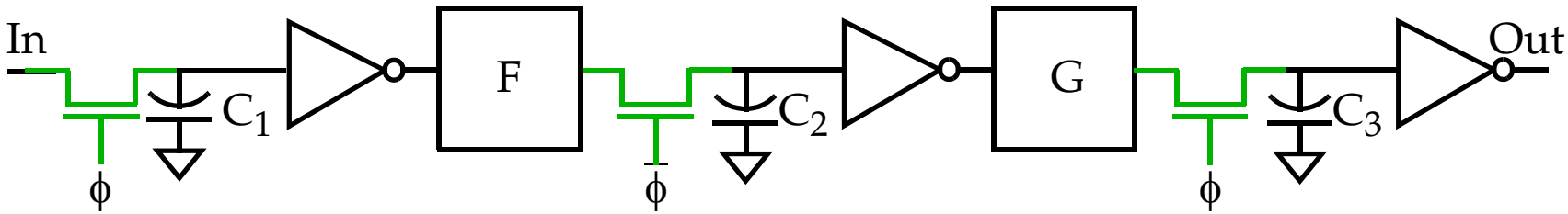
Pipelining



The minimum allowed clock for the pipelined system is:

$$T_{min} = t_q + \max(t_{d,add}, t_{d,abs}, t_{d,log}) + t_s$$

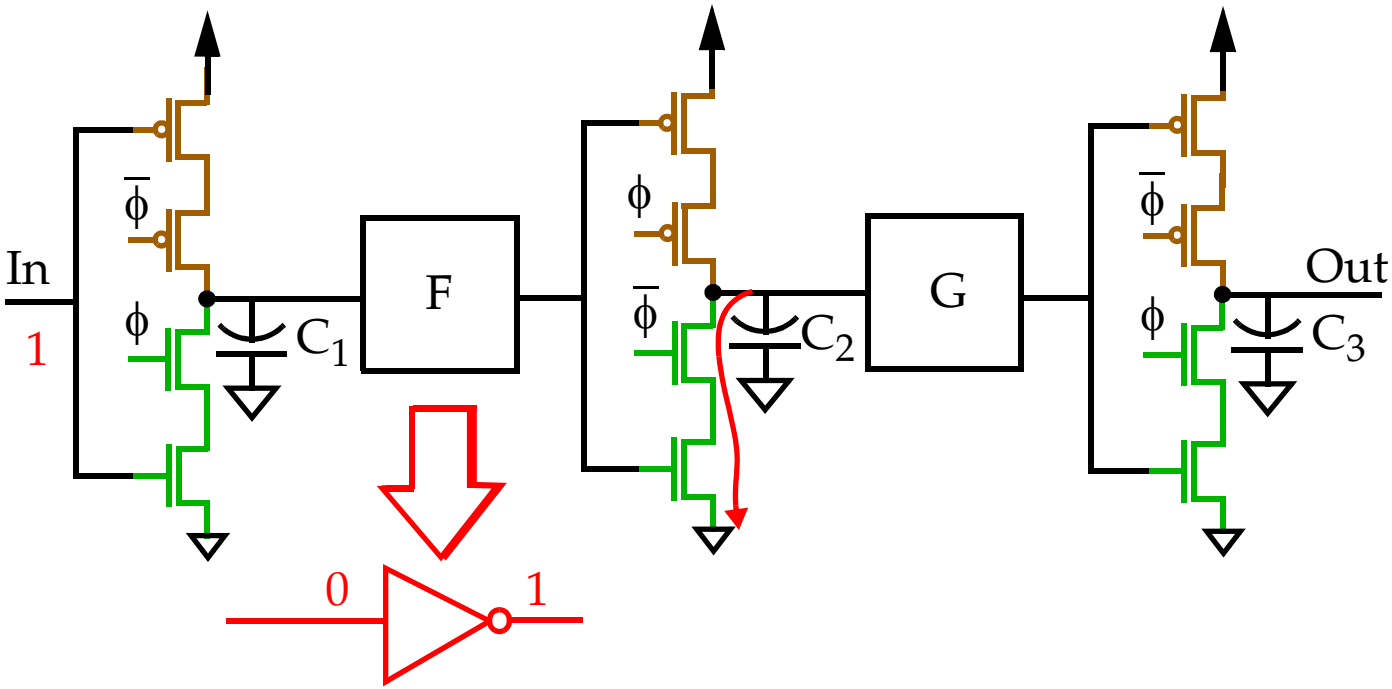
Implementation using pass-transistor based D latches



As indicated, races can occur when ϕ and $\overline{\phi}$ overlap.

Pipelining

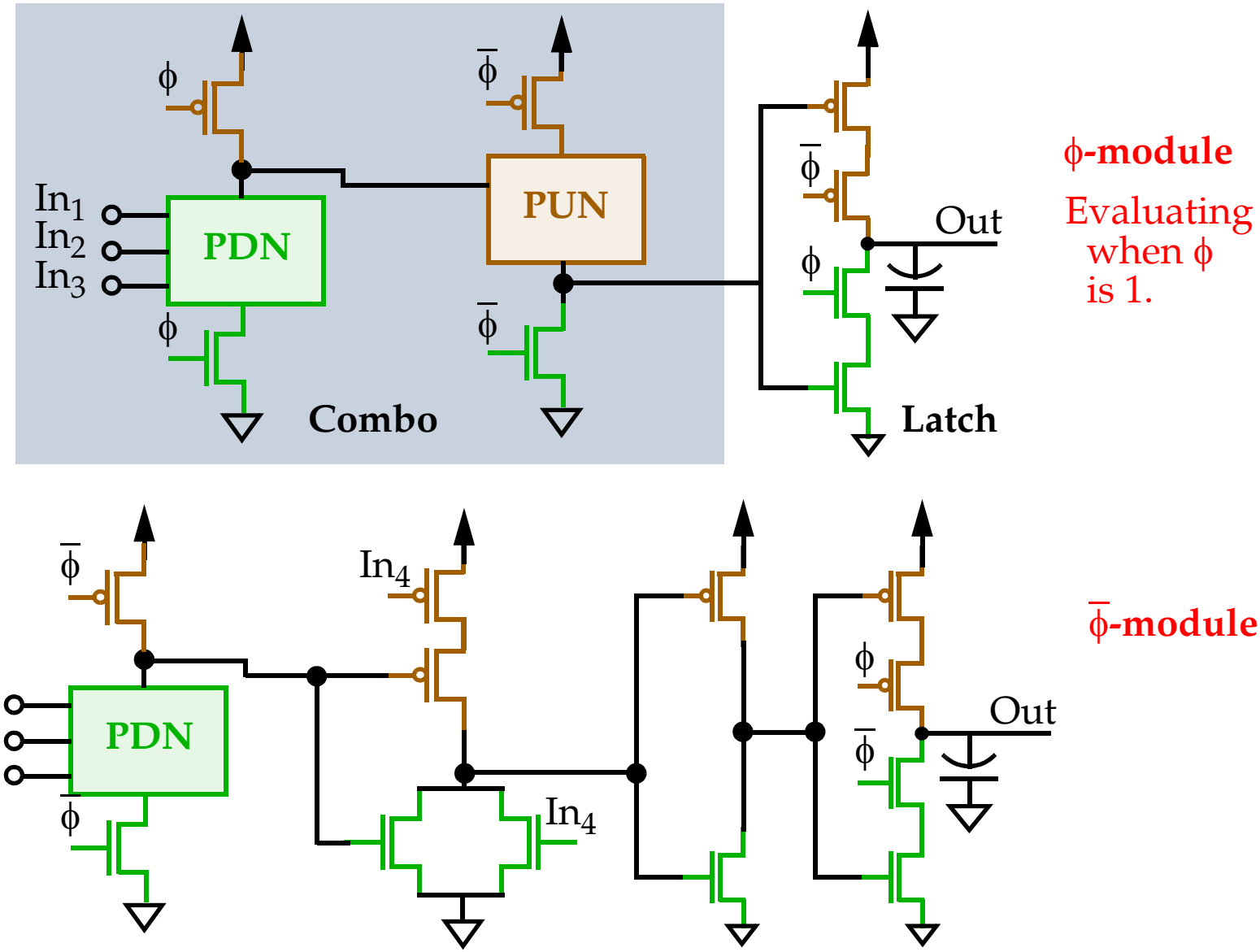
C^2MOS latches can be used instead, but ONLY if the logic functions, *F*, implemented between the latches are **non-inverting**.



If *F* is inverting, and ϕ and $\overline{\phi}$ overlap (1-1), then C_2 is discharged as shown above.

NORA-CMOS (NO-RAce) targets the implementation of fast pipelined datapaths by combining C^2MOS with *np*-CMOS dynamic function blocks.

Pipelining with NORA-CMOS



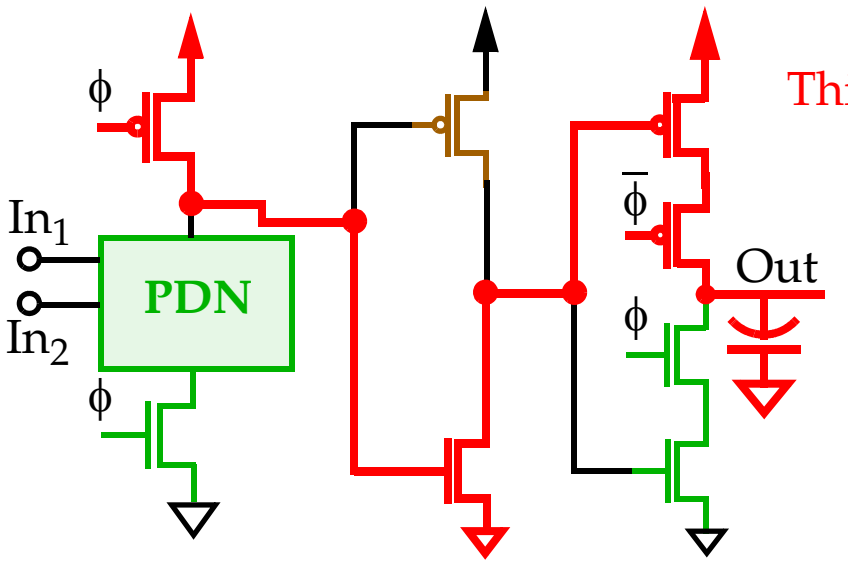
Pipelining with NORA-CMOS

The NORA datapath consists of a chain of alternating ϕ and $\overline{\phi}$ modules.

While one class of modules is *precharging* with its output latch in **hold mode**, the other class is *evaluating*.

Note that dynamic and static logic can be mixed freely.

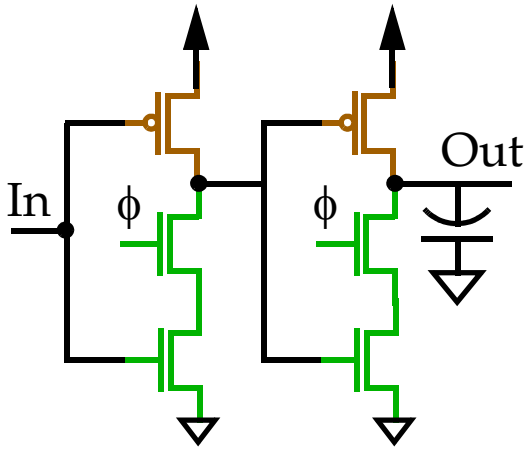
0-0 clock overlap



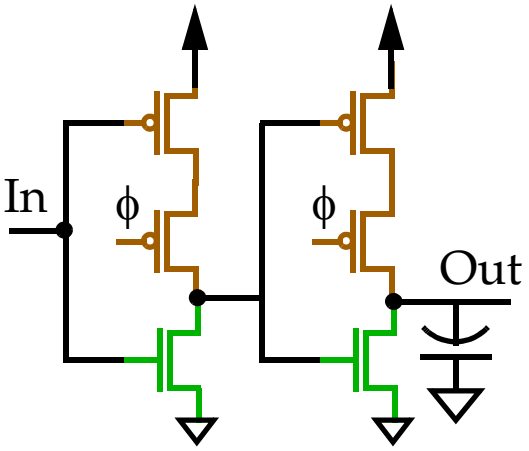
Rule: # of static inversions between C^2MOS latches should be **even**. When *dynamic* gates are present, the # of static inverters between a latch and dynamic gate and between the last dynamic gate and latch should be **even**.

True Single-Phase Clocked Logic (TSPCL)

The NORA design style can be simplified so that a **single clock** is sufficient.



Doubled n-C^{2}MOS latch



Doubled p-C^{2}MOS latch

For the doubled *n*-C^{2}MOS latch, when ϕ = 1, the latch is in the **transparent** *evaluate* mode and corresponds to 2 cascaded inverters (non-inverting).

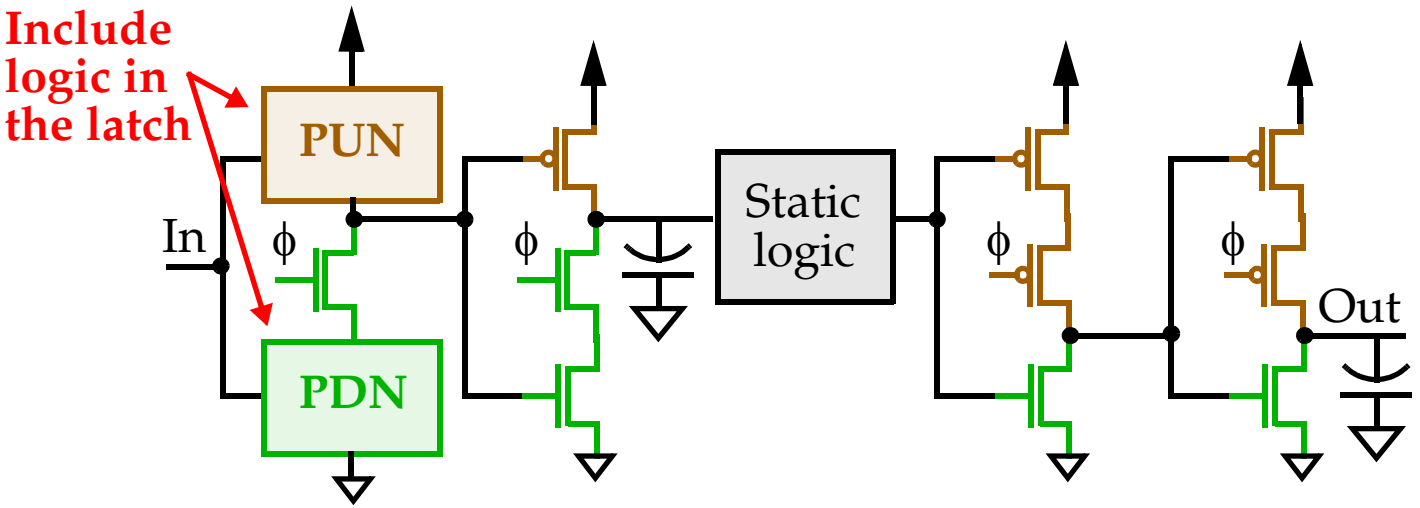
When ϕ = 0, both inverters are disabled (*hold* mode) -- only the pull-up network is still active.

The dual stage approach **completely eliminates races**.

This style combines the advantages of C^{2}MOS and eliminates all constraints.

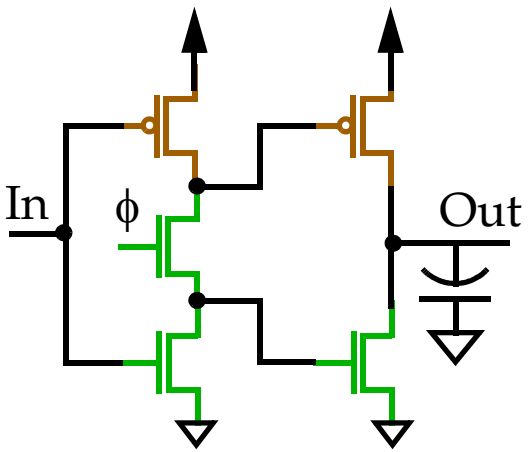
True Single-Phase Clocked Logic (TSPCL)

The one disadvantage is that **6** transistors (vs. **4**) are needed per latch.

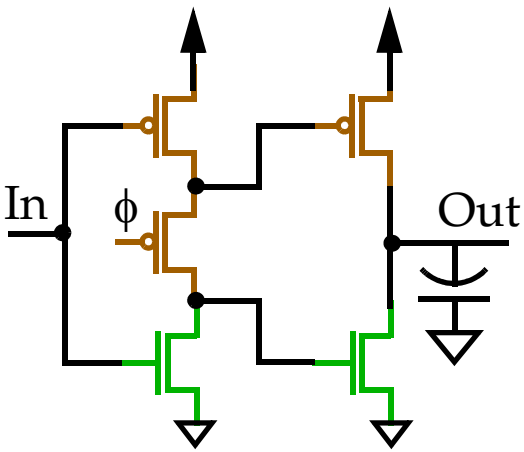


A further simplification is to control *only* the first inverter with the clock.

ϕ-latch



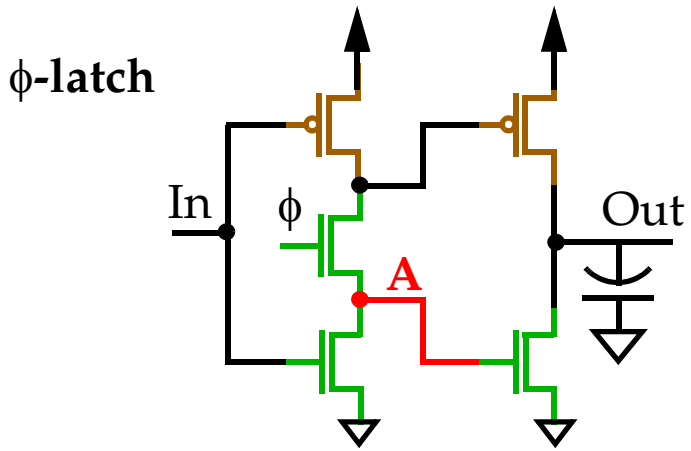
$\overline{\phi}$-latch



True Single-Phase Clock Logic (TSPC)

This reduces the number of transistors and the clock load is **reduced in half**.

Problem: not all node voltages experience the full logic swing.



Node *A* (for $V_{in} = 0V$) maximally reaches $V_{DD} - V_{Tn}$.

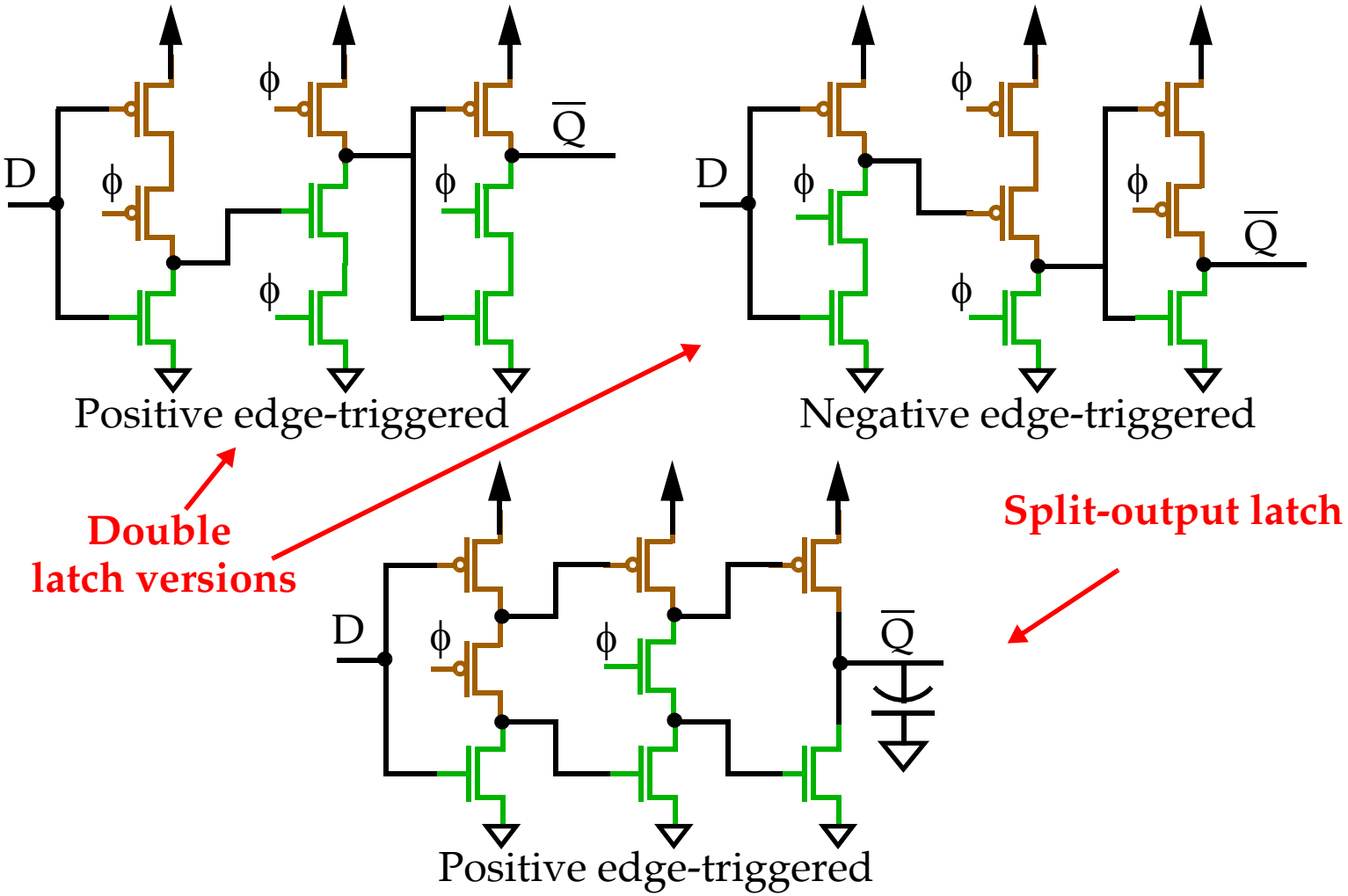
This results in a reduced drive for the output NMOS transistor and a loss in performance.

This design methodology is called **True Single-Phase Clock Logic** (TSPC).

It allows for the implementation of dynamic sequential circuits with a single clock.

True Single-Phase Clock Logic (TSPC)

Edge-triggered FFs:



Split-output version reduces clock load in half, while performing well.

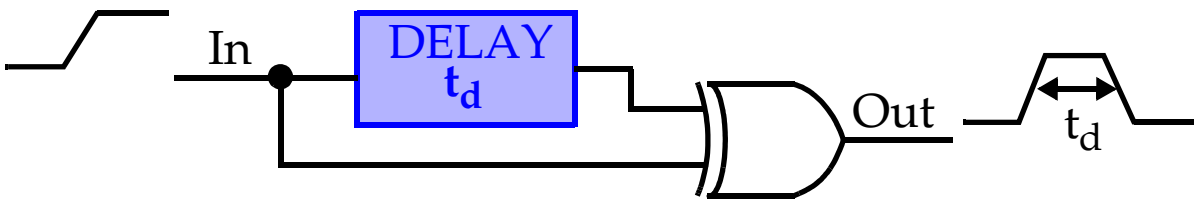
Monostable Sequential Circuits

A circuit that generates a pulse of a *predetermined width* every time the circuit is triggered by a pulse or transition event (**one-shot**).

The circuit has **only one stable state** -- the quiescent state.

The trigger causes the circuit to go temporarily into a *quasi-stable* state.
It returns to its quiescent state after a time period determined by the circuit parameters.

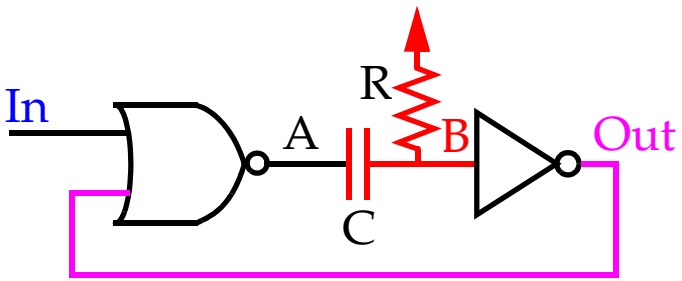
Useful for *address transition detection* (ATD) to generate timing in static memories for subsequent operations.

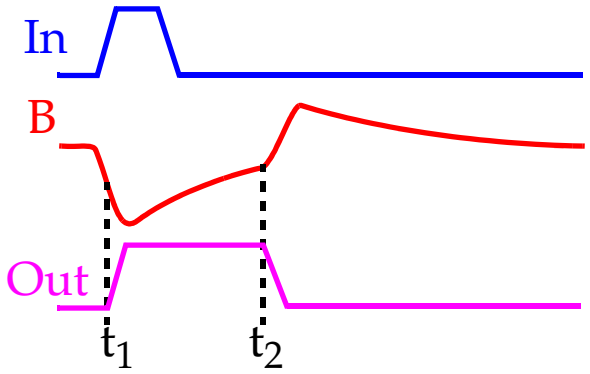


We've seen this version in edge-triggered FFs.

Monostable Sequential Circuits

A second class uses feedback combined with an *RC timing network* to generate a pulse of fixed width.





Initially, *In* and *Out* are low and therefore *A* is high. *B* is high via resistor R.

Pulsing *In* high causes *A* to go low, pulling node *B* with it.

Node *B* gets pulled high again with time constant RC.

Out goes low when *B* reaches V_M, which causes *A* to go high again (note *In* has already gone low again).

The width (t_2 - t_1) is determined by the time-constant RC and V_M.

Unfortunately, V_M is relatively sensitive to process variations.

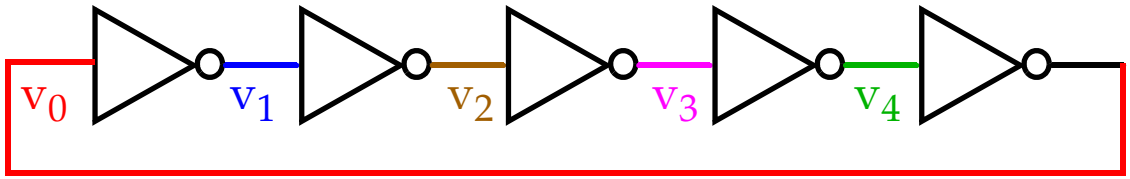
Astable Sequential Circuits

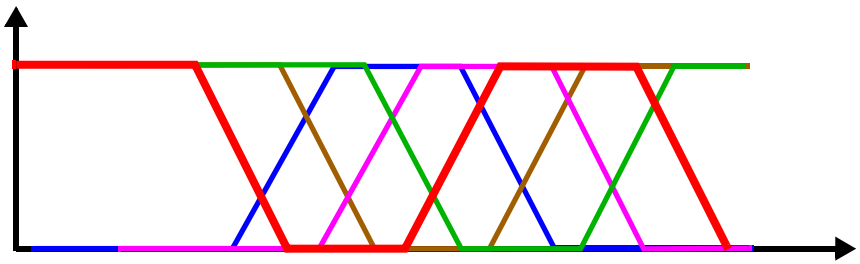
A circuit with **no stable states**.

The output oscillates back and forth between two *quasi-stable* states with a period determined by circuit parameters.

The main application of such a circuit is **on-chip clock generation**.

We already looked at the *ring oscillator* as an example.





Ring oscillator

The period T of the oscillation is:

$$T = 2 \times t_p \times N$$ where N is the # of inverters in the chain.

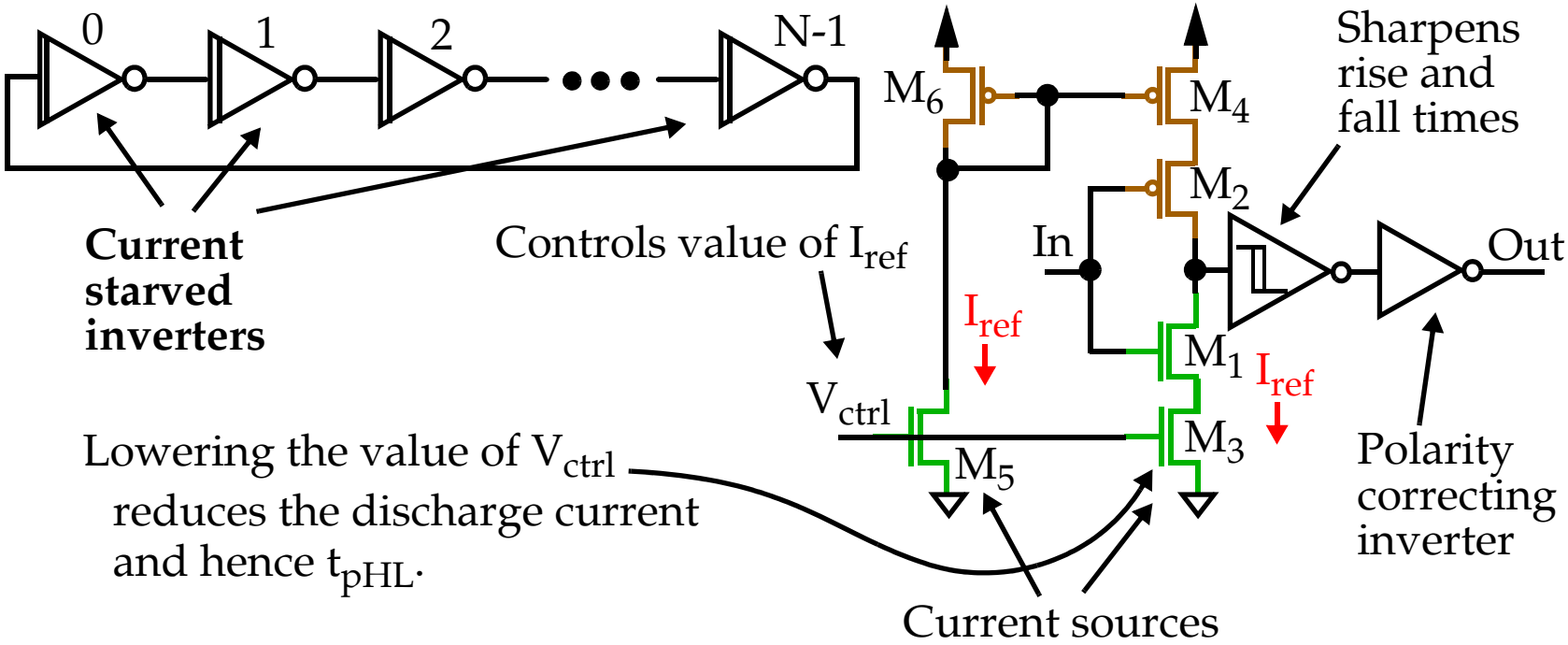
where t_p is the propagation delay of the composing gates.

Astable Sequential Circuits

By tapping the ring oscillator at different stages, a wide range of clock signals with different duty-cycles and phases can be derived.

It is often desirable to tune the frequency of oscillation.

An example is a **Voltage-controlled Oscillator** (VCO), whose frequency is proportional to the value of a control voltage.



Astable Sequential Circuits

Charging current is controlled via M_5.

I_{ref} is translated into a charging current through the **current mirror** M_6 - M_4.

Here, M_6 acts as a **diode** and sets a bias voltage V_{GS6}, that is controlled by I_{ref}.

With $V_{GS4} = V_{GS6}$ and both devices operating in *saturation*, $I_{DS4} = I_{DS6} = I_{ref}$.

Since both M_3 and M_5 operate in *saturation*, a quadratic relation exists between V_{ctrl} and I_{ref} (and t_p).

This allows the frequency of the VCO to be controlled over a large range.

A Schmitt trigger is used to **sharpen** the weakened rise and fall times of the *current-stared inverter*.

Note that transistors M_5 and M_6 can be shared over all inverters in the chain.