## CMOS Logic Structures

Full complementary static CMOS gates may be undesirable because:

- The area overhead.
- Their speed may be too slow.
- The function may not be feasible as a full complementary structure (e.g. PLA).

Smaller faster gates can be implemented at the cost of:

- Increased design time.
- Increased operational complexity.
- Decreased operational margin.

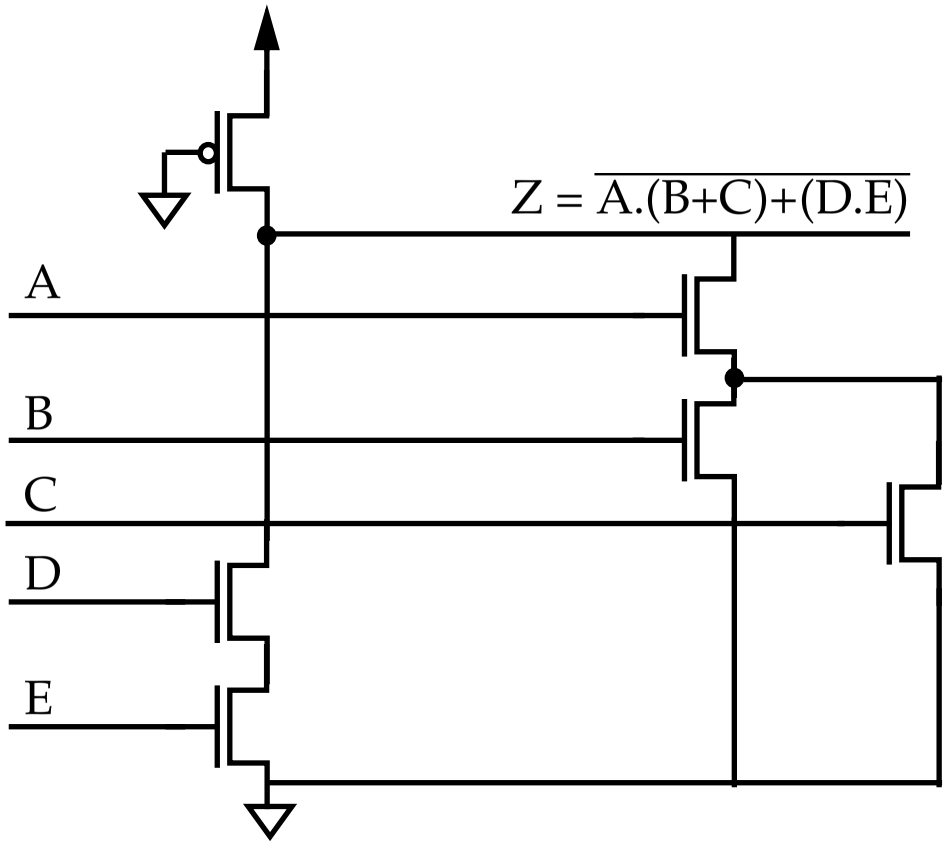
Full complementary gates can be designed as *ratioless* circuits:

- A fixed ratio in size between pull-up and pull-down structures is not required for proper operation.

Unlike those we will consider now.

## CMOS Logic Structures

Pseudo-nMOS logic




Advantages:

Capacitive load on inputs is only one gate unit.
Density advantage over full complementary CMOS.

Main problem:

Static power dissipation

Gain of the pull-up has to be decreased to provide adequate noise margins (when minimum sized transistors are used).

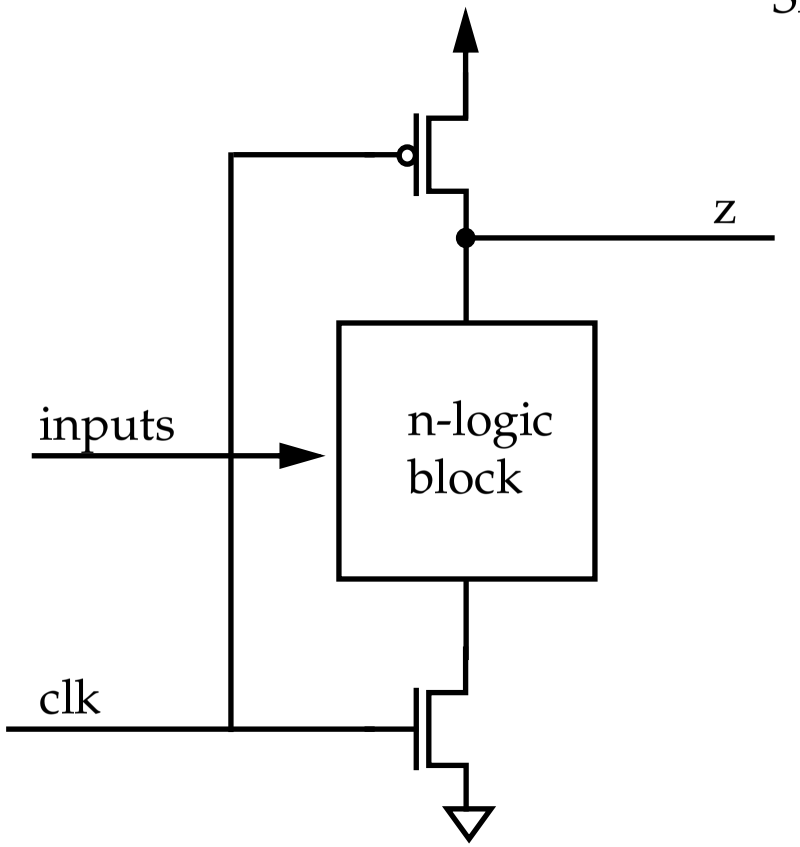
→ Slows rise time.

Gain ratio of n-driver transistors to p-transistor load ($beta_{driver}/beta_{load}$), is important to ensure correct operation.

Accomplished by ratioing the n and p transistor sizes.

## CMOS Logic Structures

### Dynamic CMOS Logic




Simple single phase dynamic CMOS:

Precharge phase: Clk = 0

Evaluate phase: Clk = 1

Input capacitance: Same as that of pseudo-nMOS.

Problem:

Inputs can only change during the precharge phase.

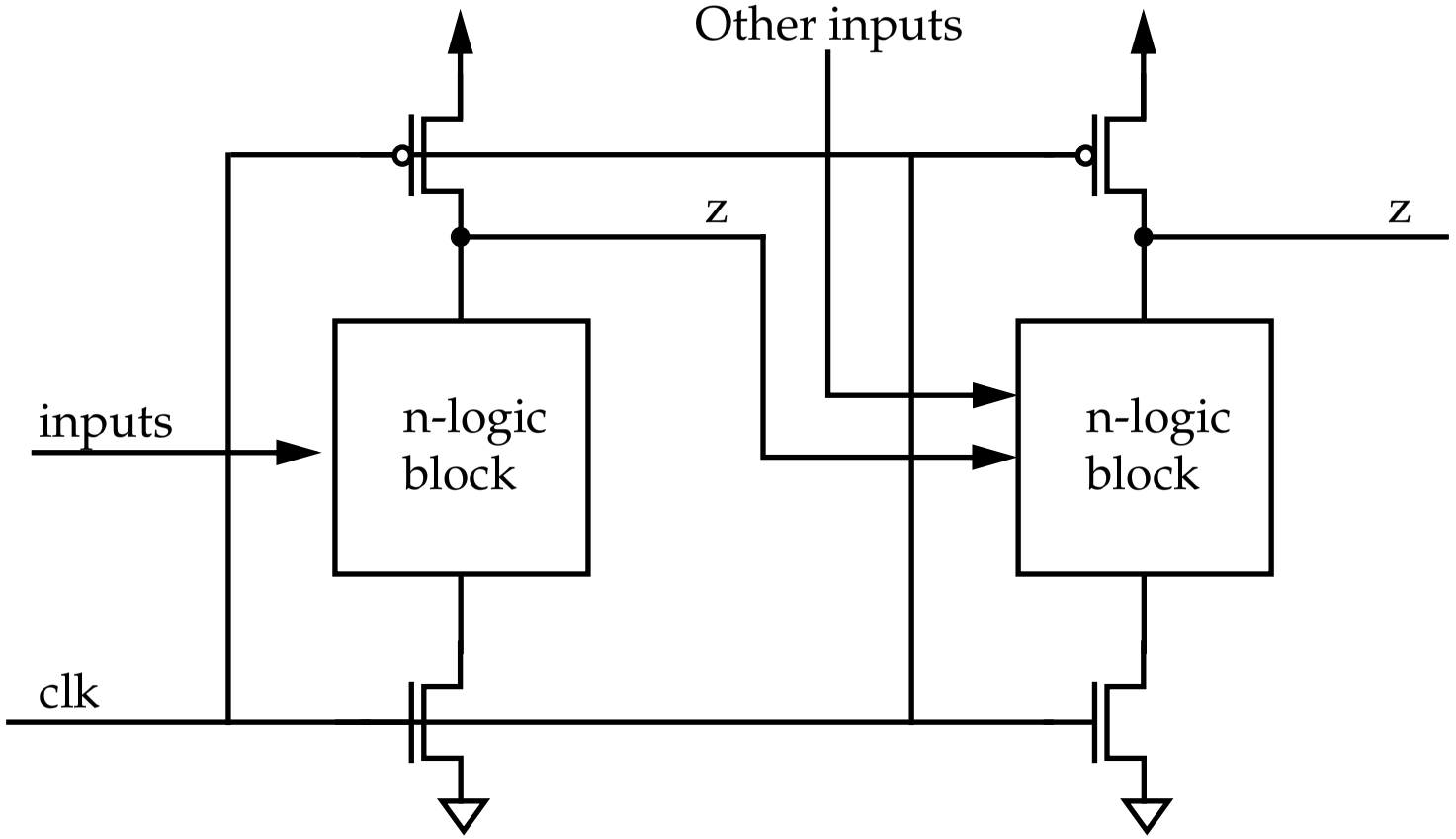
If they change during evaluate, charge redistribution can corrupt output voltage.

Pull-up time improved by virtue of the active switch (p-transistor can be much larger).

Pull-down time increased due to the ground switch.

**CMOS Logic Structures**

Dynamic CMOS Logic

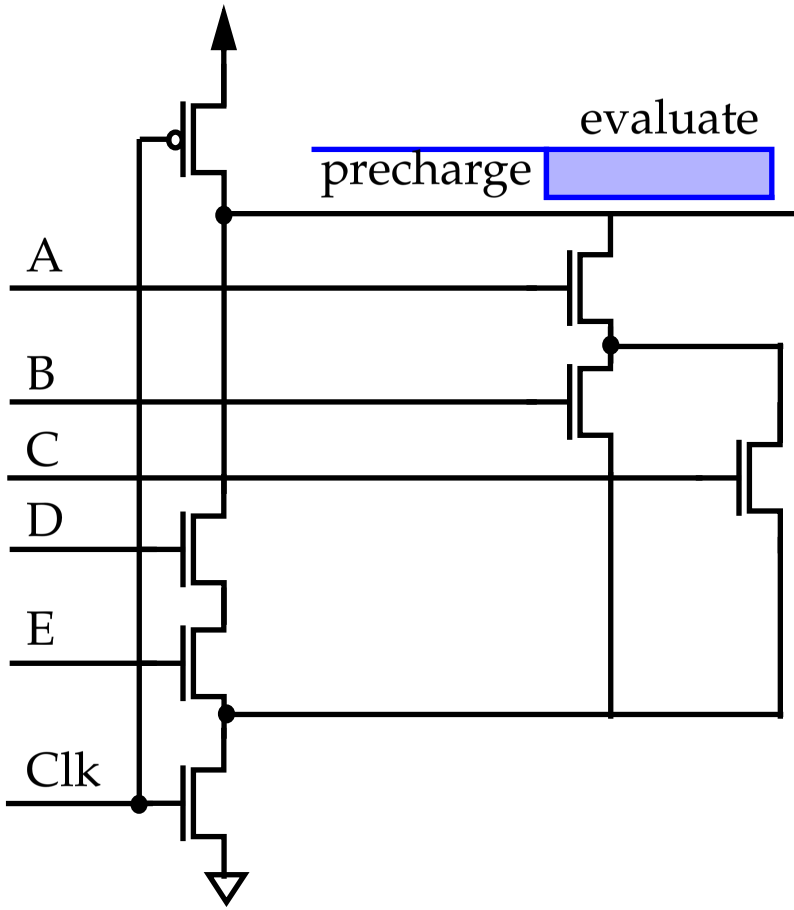



What is wrong with cascading these structures?

(Hint: Consider the delay in the discharge of the left-most n-logic block at the start of the evaluate phase).

## CMOS Logic Structures

CMOS Domino Logic:




Simple mod of Dynamic logic: Added a static CMOS inverter

Next domino stage.

Only non-inverting forms of logic possible.

**Precharge** phase: Output of buffer, z, is zero.

**Evaluate** phase: Output, z, conditionally goes high.

Dynamic circuit as shown:

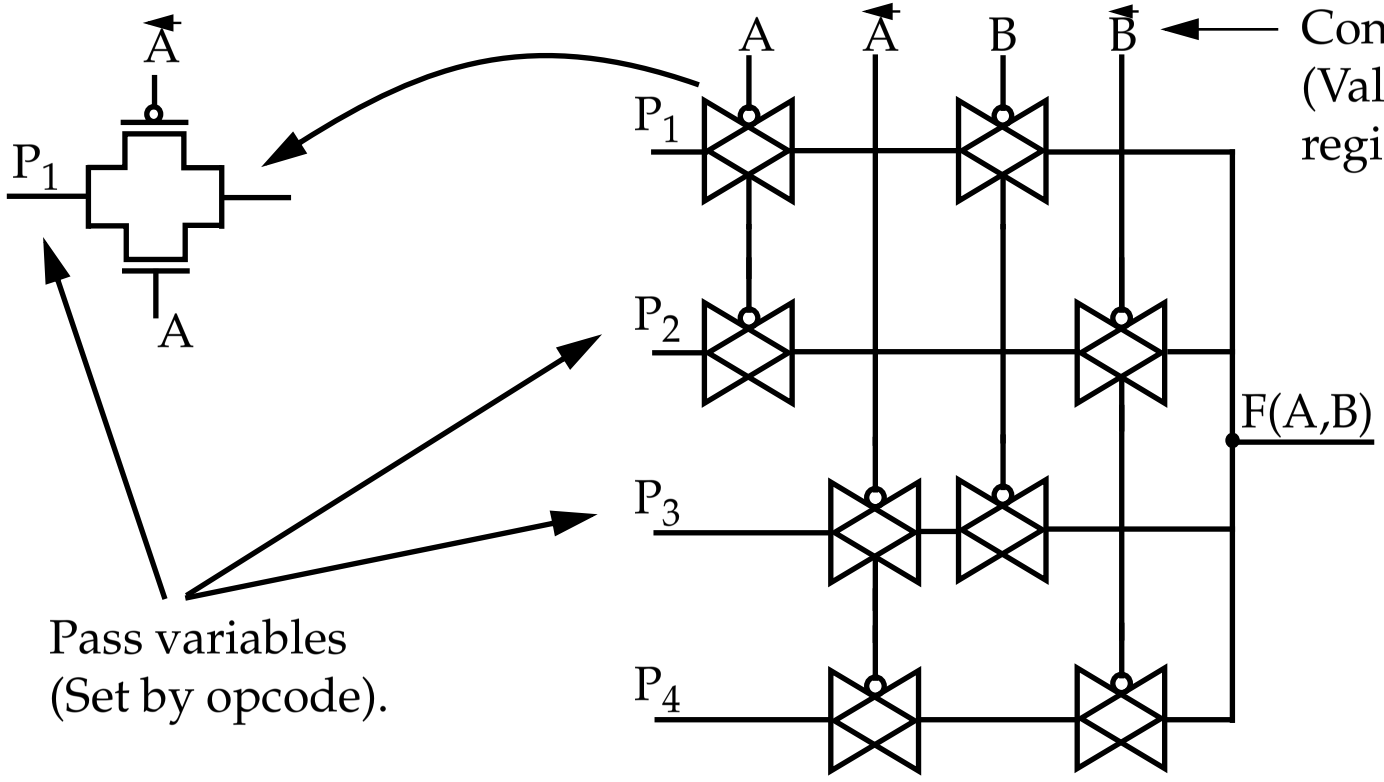
Can be made static or latching with the addition of a (weak) p and feedback.

**These structures can be cascaded.**

In a cascaded set of logic blocks, each stage evaluates and causes the next stage to evaluate (in the same way a line of dominos fall).

## CMOS Logic Structures

Pass-Transistor Logic:




| Operation | $P_4$ | $P_3$ | $P_2$ | $P_1$ |
|---|---|---|---|---|
| AND(A,B) | 0 | 0 | 0 | 1 |
| XOR(A,B) | 0 | 1 | 1 | 0 |
| NOR(A,B) | 1 | 0 | 0 | 0 |

**CMOS Logic Structures**

Other forms of CMOS logic include:

- BiCMOS Logic
- Clocked CMOS Logic ($C^2$MOS).
- NP Domino Logic (Zipper CMOS).
- Cascade Voltage Switch Logic (CVSL).
- Source Follower Pull-up Logic (SFPL).

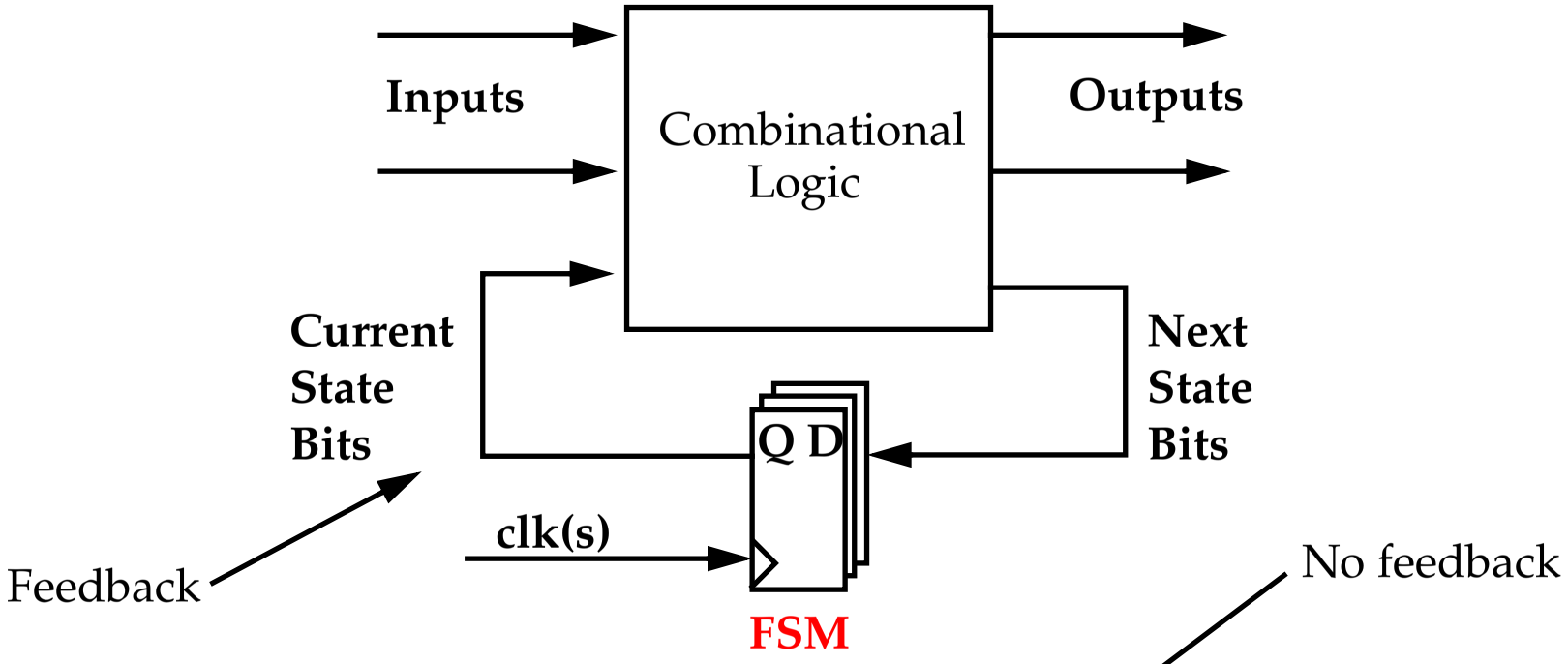
(See Weste and Eshraghian for details.)

Where should one use what gate?

- Complementary: Best option for most cases. Safe, fast, no DC power.
- Pseudo-nMOS: Large fan-in NOR gates, i.e. PLAs, ROMs. DC power.
- Transmission gate: Speed advantage, good for complex boolean functions.
- CMOS domino logic: Low-power, high speed. Requires simulation!

**Clocked Systems**

Majority of VLSI systems are Finite State machines and Pipelined machines:



**Clock Strategy**

One of the most important decisions made at the start of a design is the selection of a clocking strategy.

It effects:

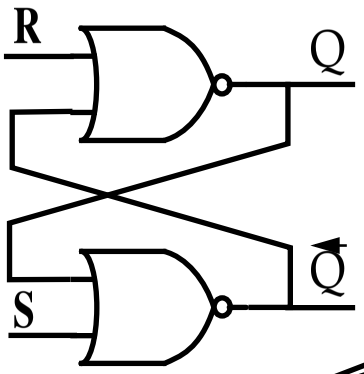
- How many transistors are used per storage element.
- How many clock signals need to be routed throughout the chip.

Topics:

- Latch, Master-Slave Flip-flop and Edge-Triggered Flip-flop designs.
- Setup and Hold time and clock race conditions.
- CMOS Static and Dynamic Flip-flops.
- Single phase clocking, clock skew/slew.
- Two-phase clocking techniques.
- Clock generation techniques.

## Latches and Flip-flops

NOR version




Positive logic

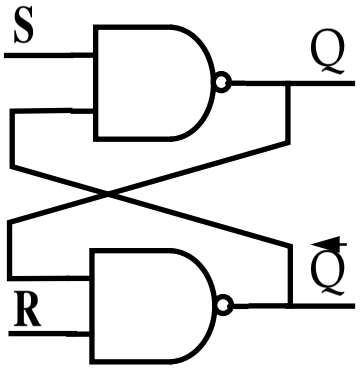
| S | R | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | 0 | Q | $\overline{Q}$ |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

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

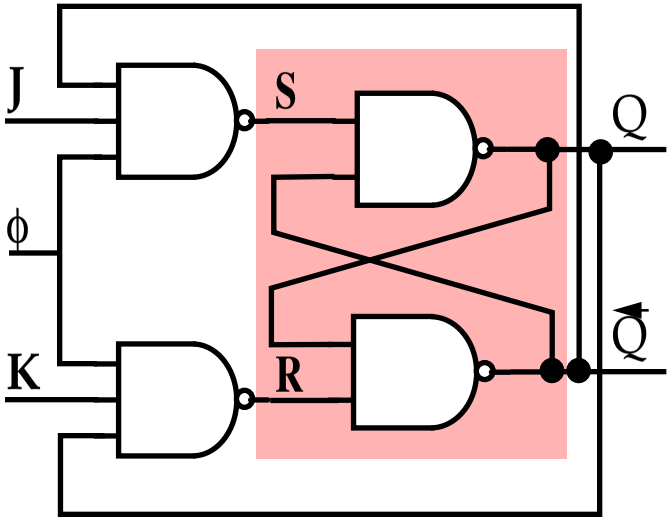



Negative logic

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

**Latches and Flip-flops**

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 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).

Synchronous circuit:

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

**Latches and Flip-flops**

Note that the:

- 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.

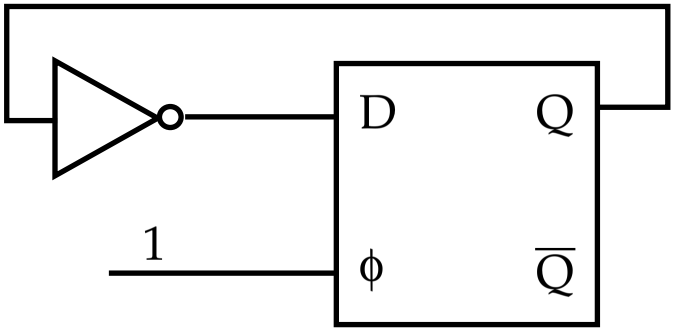
  Here the D FF generates a delayed version of the input signal synchronized with the clock.

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 are reflected in the output after a nominal delay.

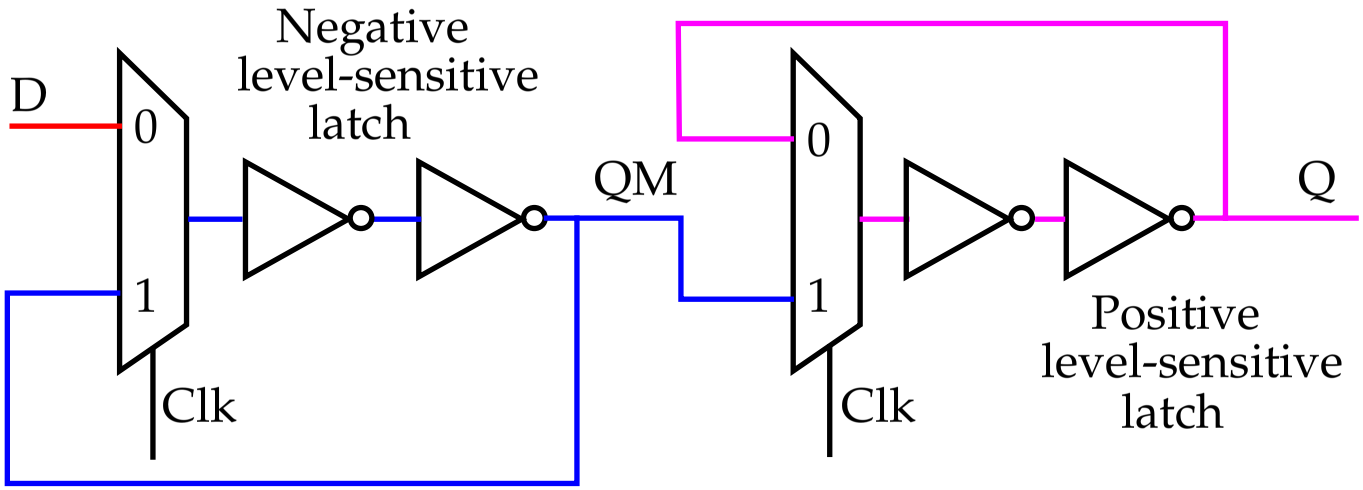
The transparent nature can cause **race** problems:

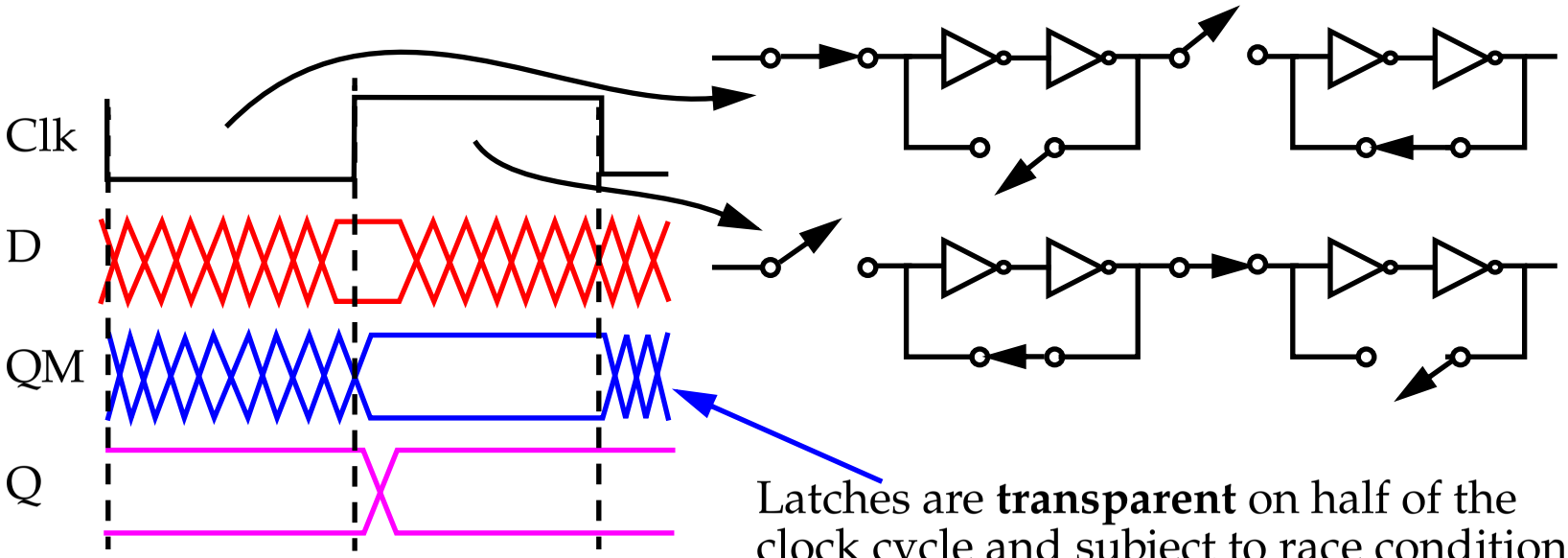



This circuit oscillates as long as phi remains high.

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

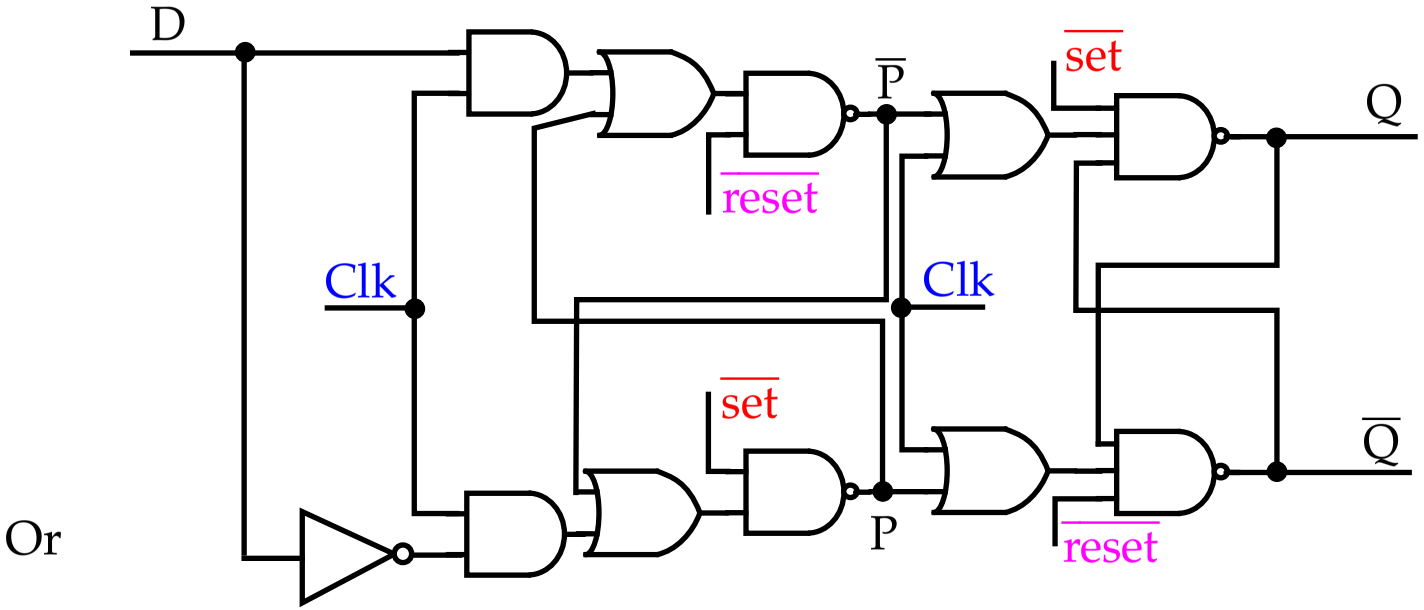
## Master-Slave Flip-flops



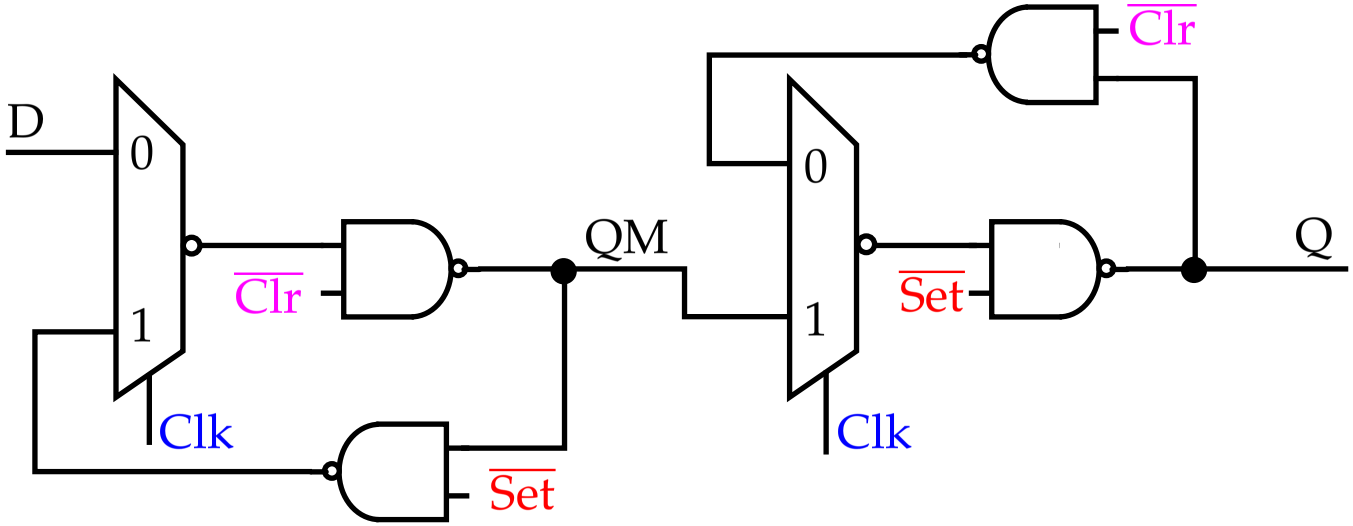




Latches are **transparent** on half of the clock cycle and subject to race conditions.

**Master-Slave Set/Clear Asynchronous FFs**




Or



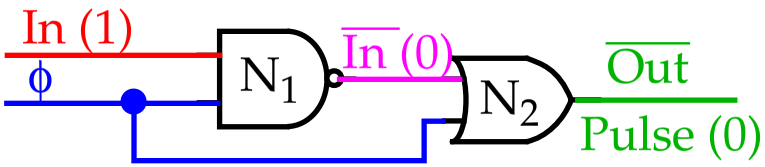
**Edge-triggered FFs**

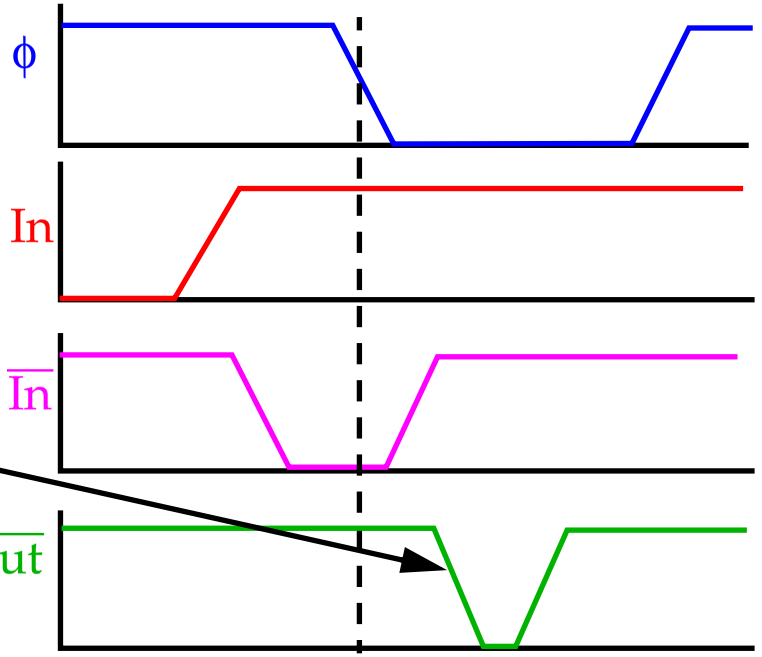
Problem with master-slave approach:

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

If the inputs do not remain constant when the clock is high, the master follows *D*, which, for example, consumes power.

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

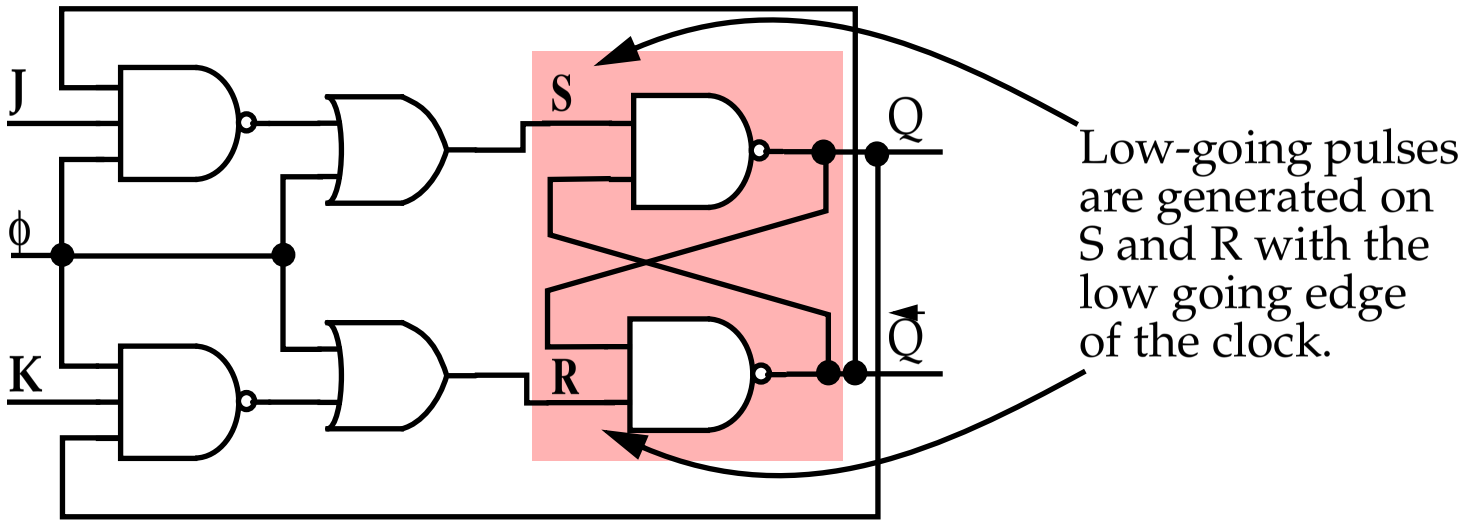



Results in a short low-going pulse at the output of $N_2$ with length approximately equal to the propagation delay through $N_1$.

**Edge-triggered FFs**

The modification applied to the JK FF is shown below.



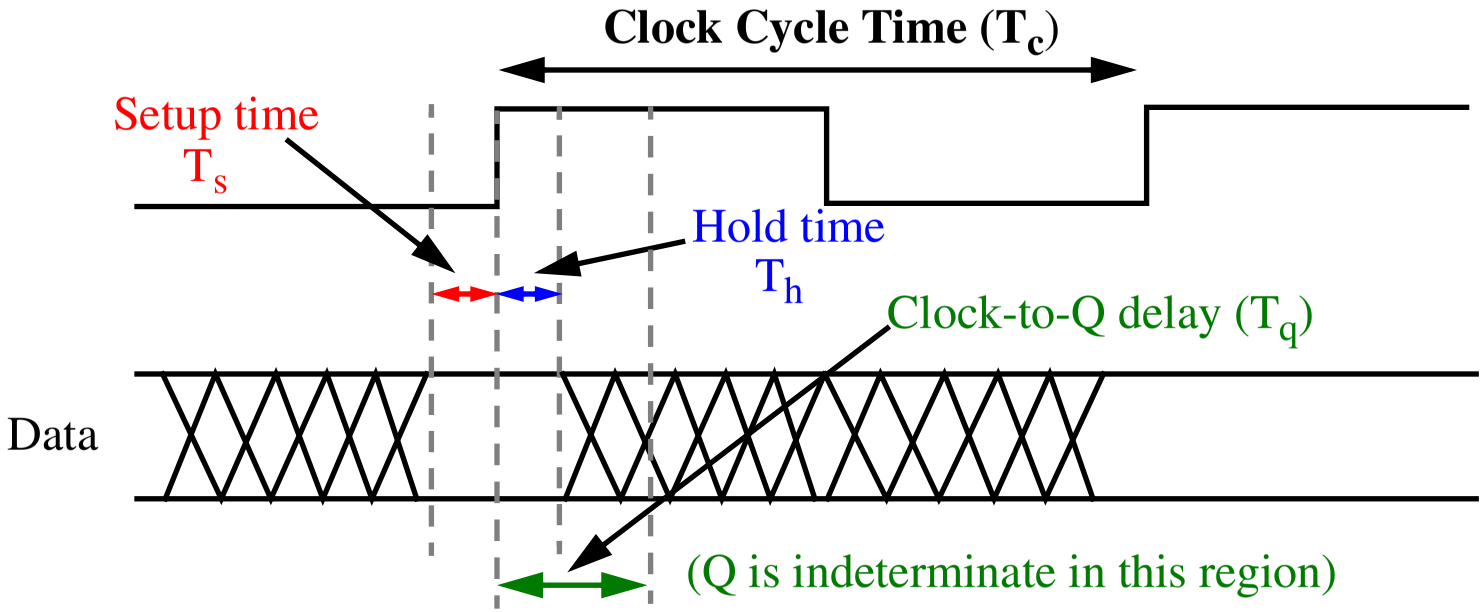

Note that the inputs must be stable for some time before the clock goes low.

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 used to define the proper operation of a Flip-flop.




$T_c$: Clock Cycle Time.

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

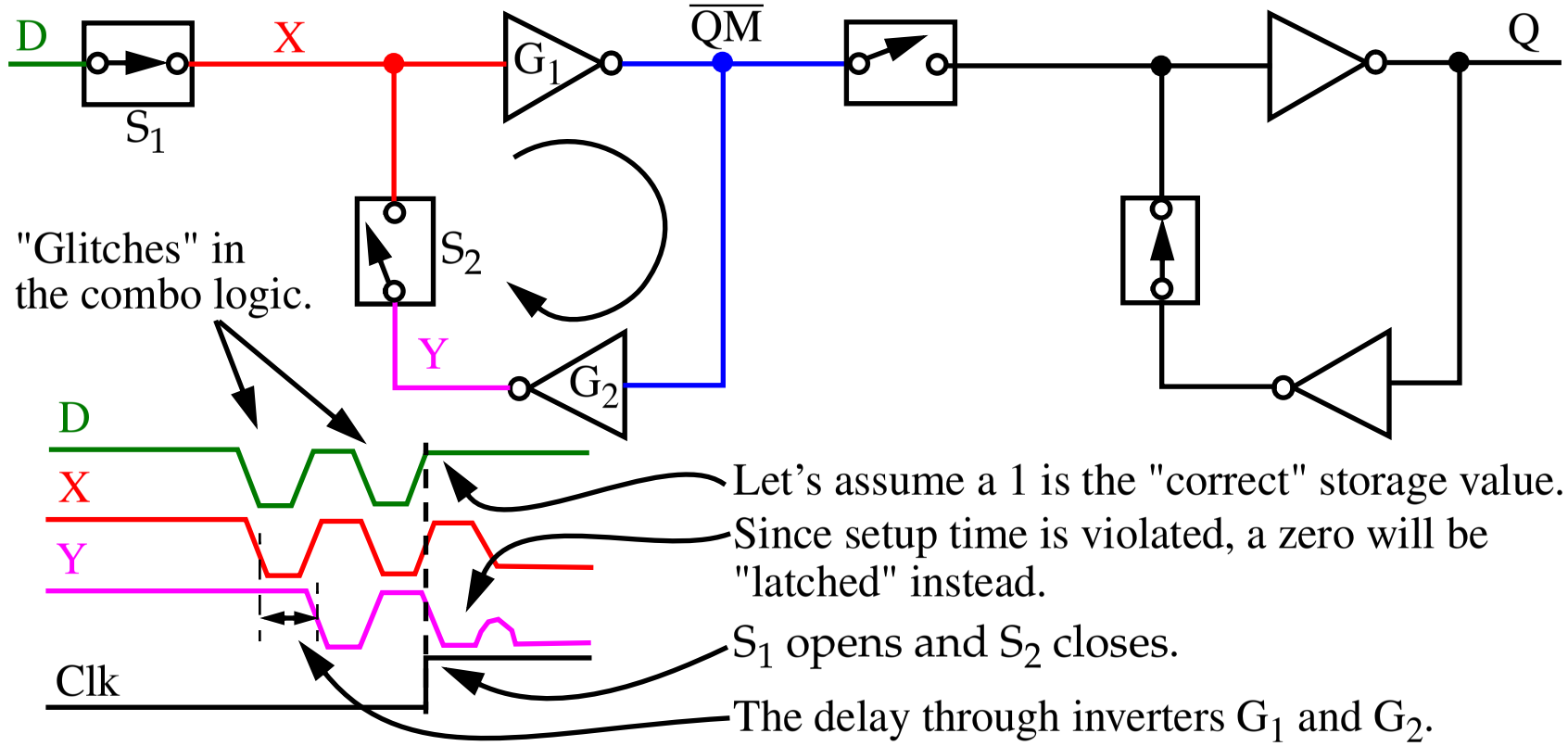
$T_h$: Data has to be held for this period while the clock travels to the point of storage.

$T_q$: Clock-to-Q delay: Delay from the positive clock input to the 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.




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

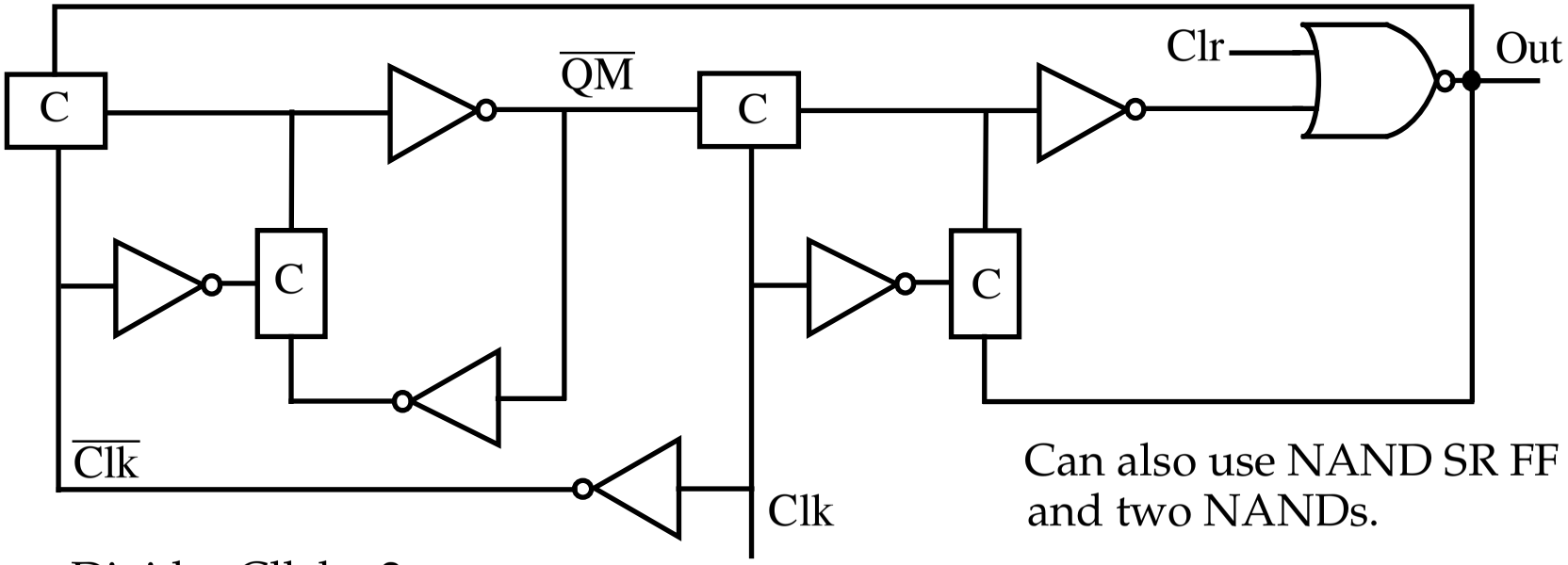
**Setup/Hold Time Violations**

The hold time interval starts with the beginning of the clock transition.

Clock skew and slew and other design details of the FF affect the hold time.

**Toggle Flip-Flop with Asynchronous Clear:**

T FF

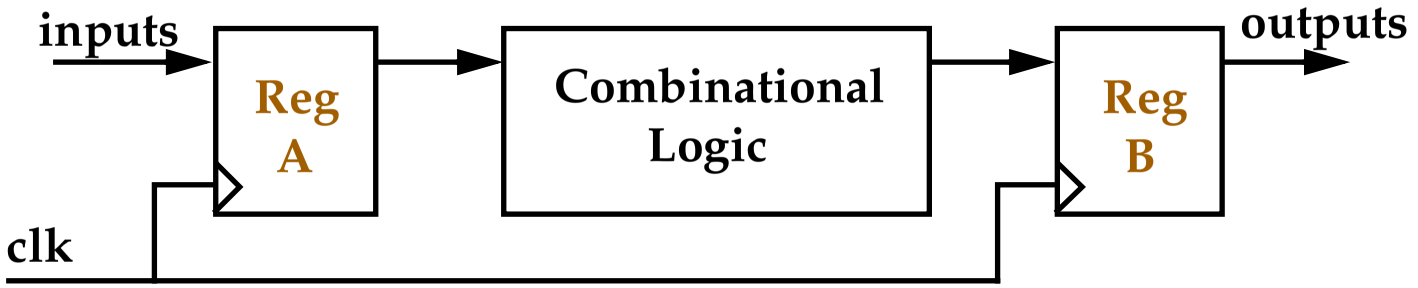


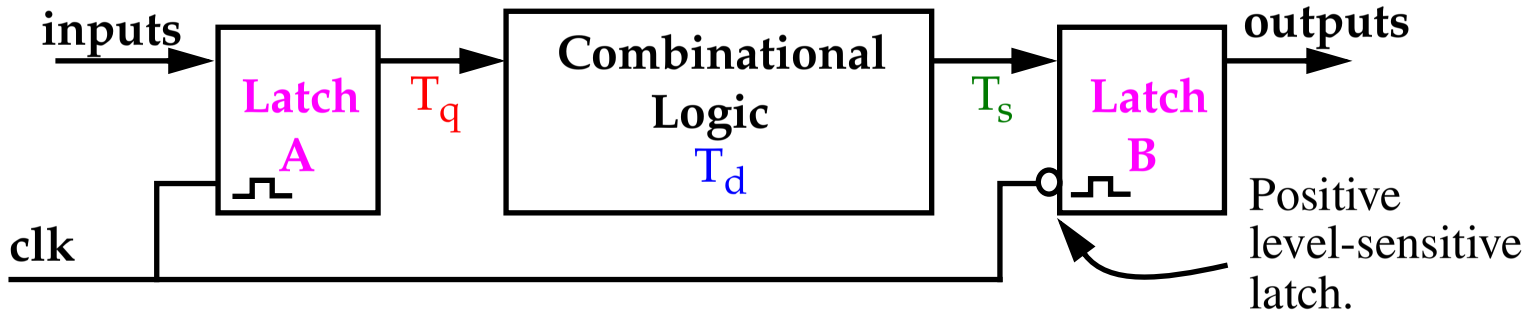

Can also use NAND SR FF and two NANDs.

Divides Clk by 2.
Used in counters.

**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, and cannot be used in feedback systems.

Also, the following constraint must be met for latches:

$T_d < T_c/2 - T_q - T_s$

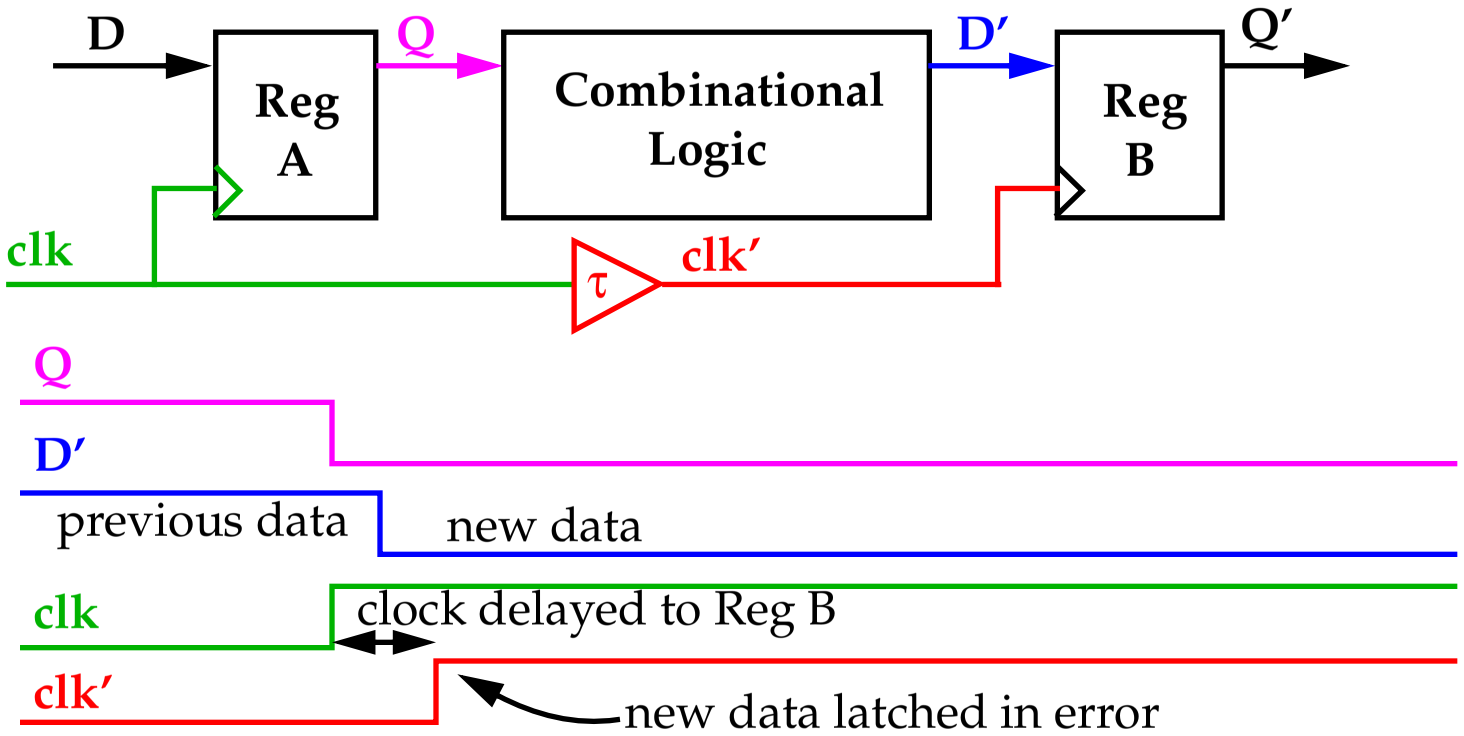
where $T_d$ is the worst case propagation delay, $T_c$ is the clock cycle time, $T_q$ is the Clock-to-Q time of latch A and $T_s$ is the setup time for latch B.

## Clock Race Conditions

Occurs when the data input to the register does not obey the setup and hold-time constraints.

- Delays in the clock line to Reg B (hold-time violation).

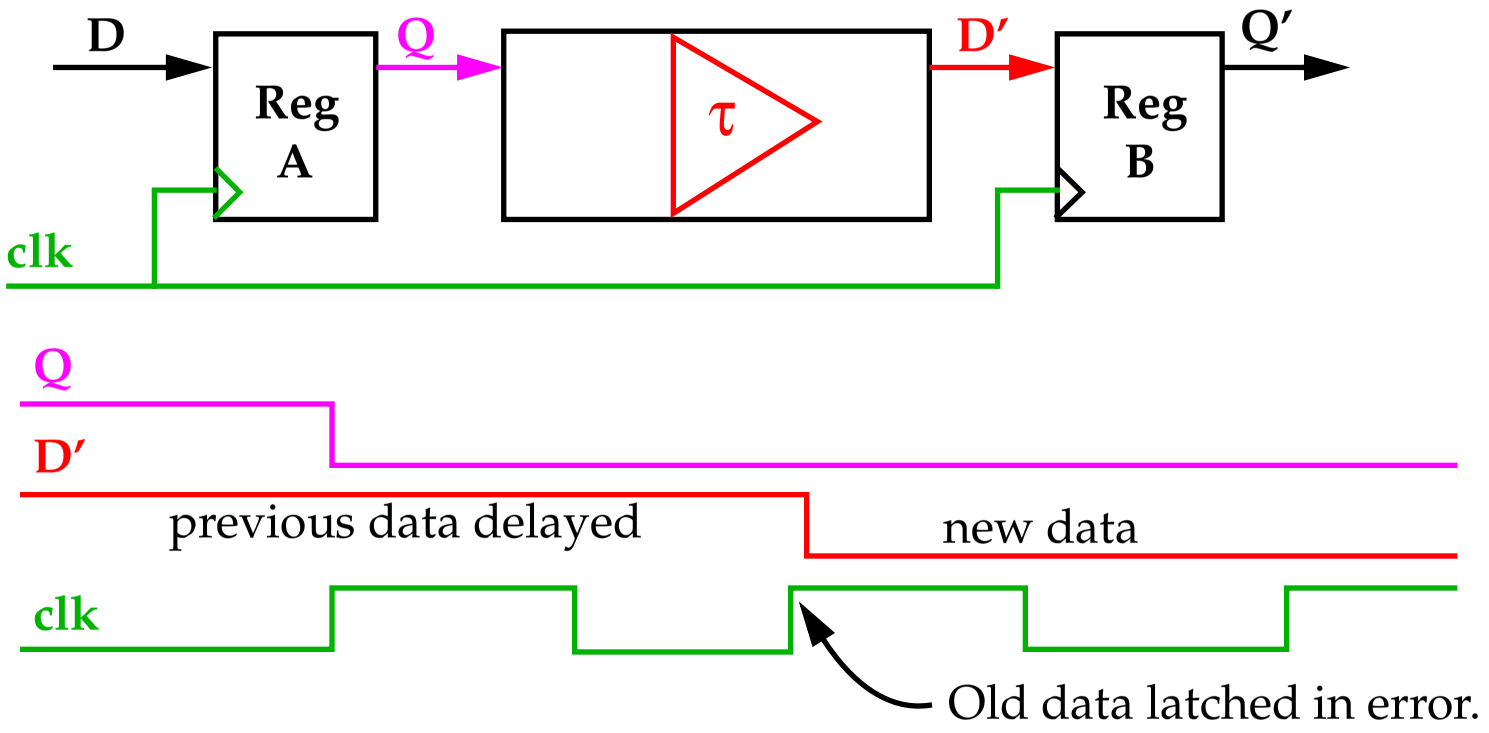
  New data stored instead of previous data:



## Clock Race Conditions

- Delays in the combinational logic that are larger than the clock cycle time (setup violation).

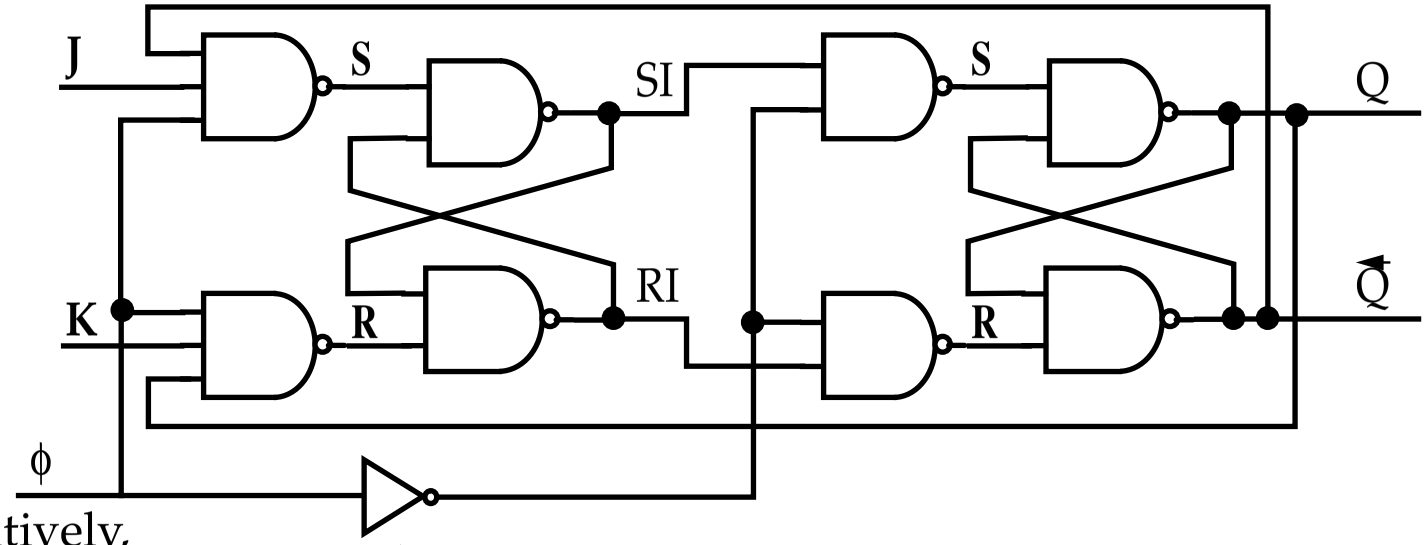
  Data arrives late at Reg B, old data retained instead of latching new data.



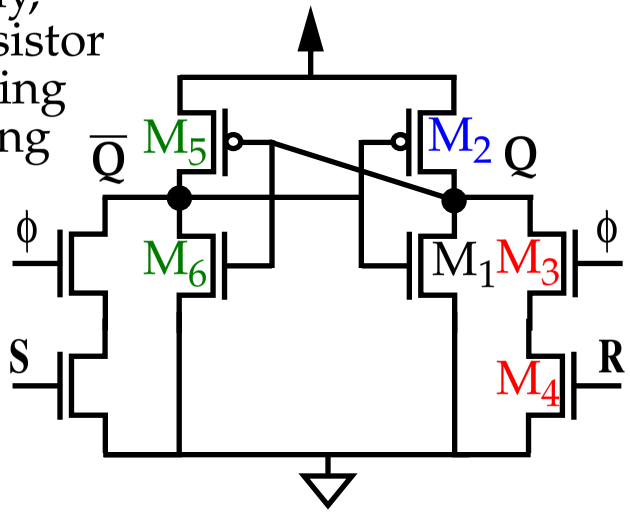

As you can see, designers have to walk a temporal 'tight-rope', e.g., they have to minimize clock skew while considering worst and best case delays through combinational logic.

## 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 phi and R/S, is the 6 trans. SRAM.

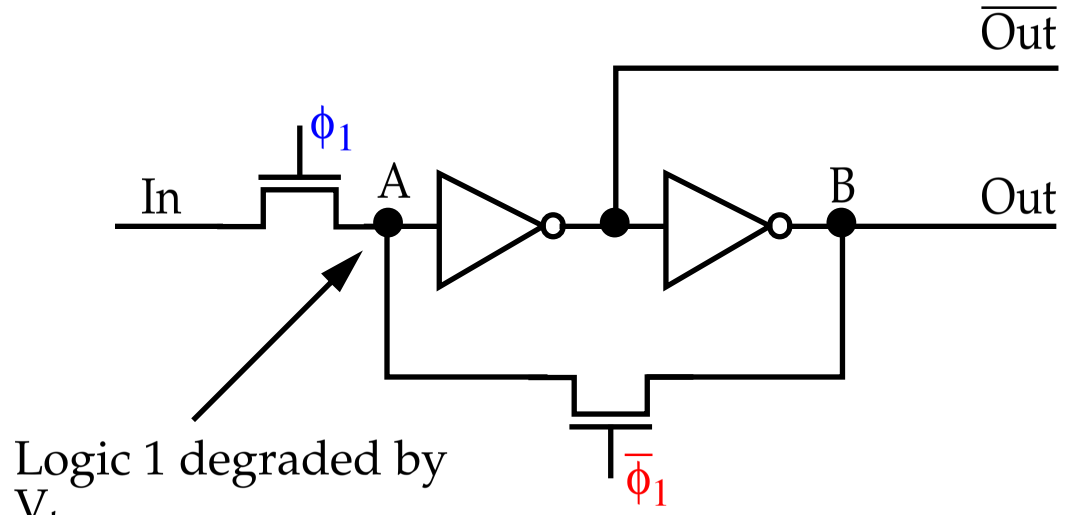
## CMOS Dynamic Flip-Flops

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* latch as a step toward deriving a dynamic FF:

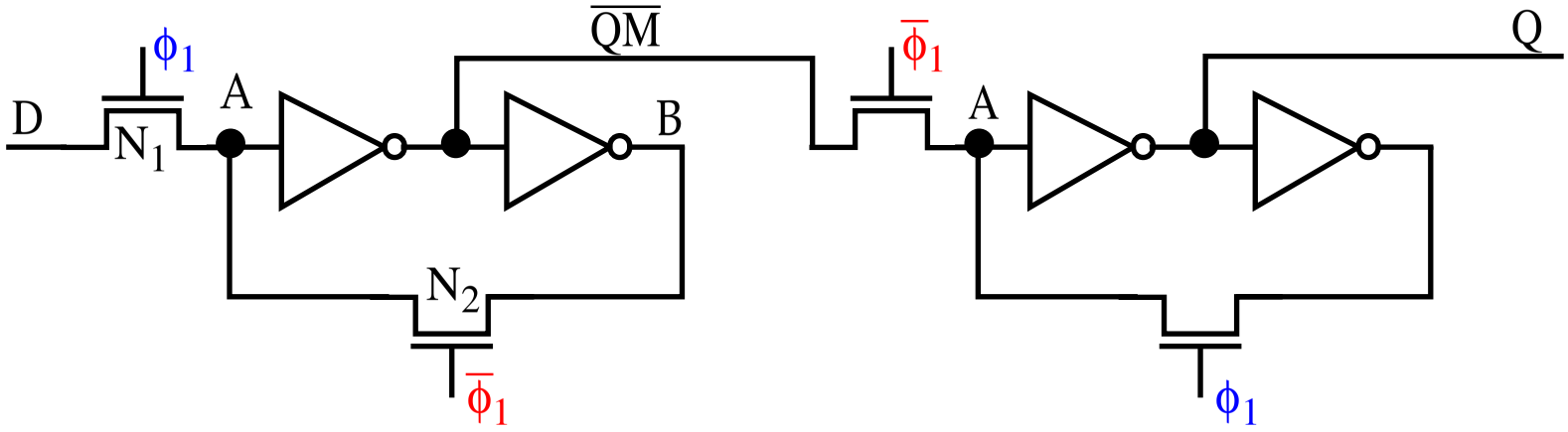



Static as long as $\phi_1$ is kept low.

When $\phi_1$ is high, *In* is sampled and stored on internal capacitors.

## CMOS Dynamic Flip-Flops

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

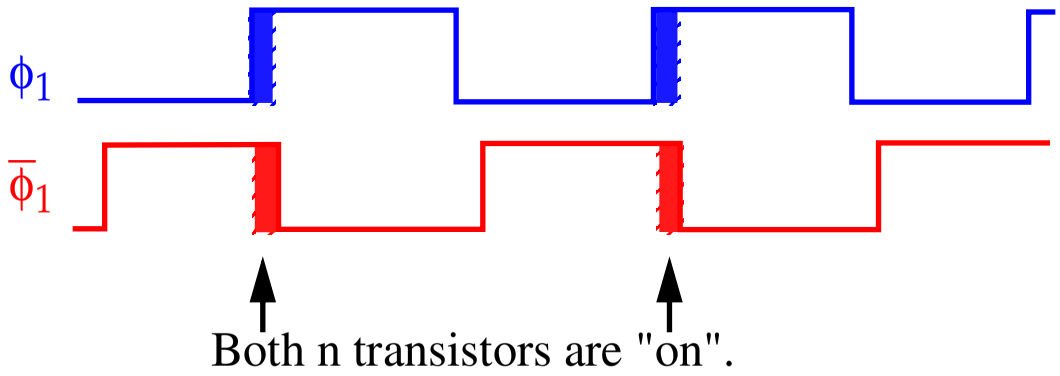



The problem with this latch is that $phi_1$ and $\overline{phi}_1$ might overlap, which may cause two types of failures:

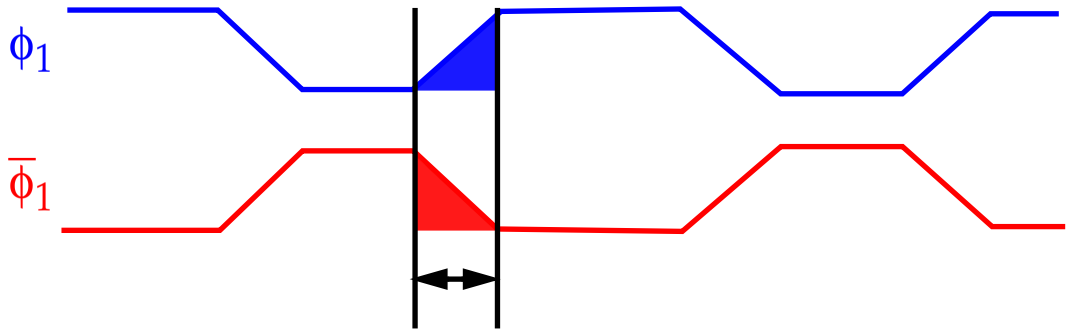
- Node A can become undefined as it is driven by both *D* and *B* when $phi_1$ and $\overline{phi}_1$ are both high.
- *D* can propagate through both the master and slave if both $phi_1$ and $\overline{phi}_1$ are high simultaneously for a long enough period.

## Single Phase Clock Skew/Slew

Clock skew causes conflicts and transparency.



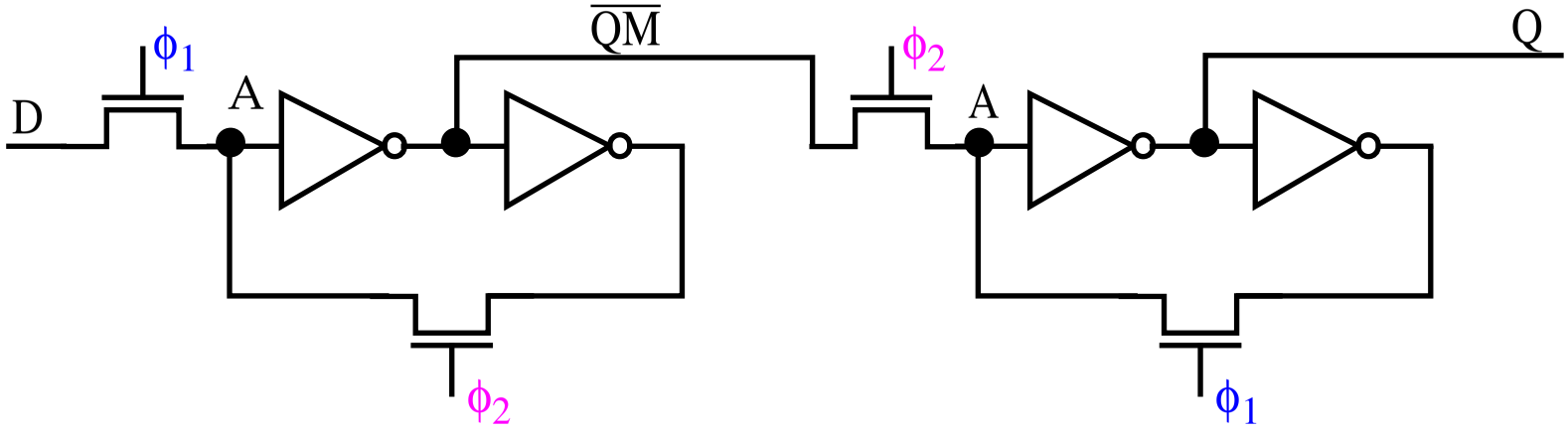

Clock slew (slow rise and fall times) can also cause transparency:



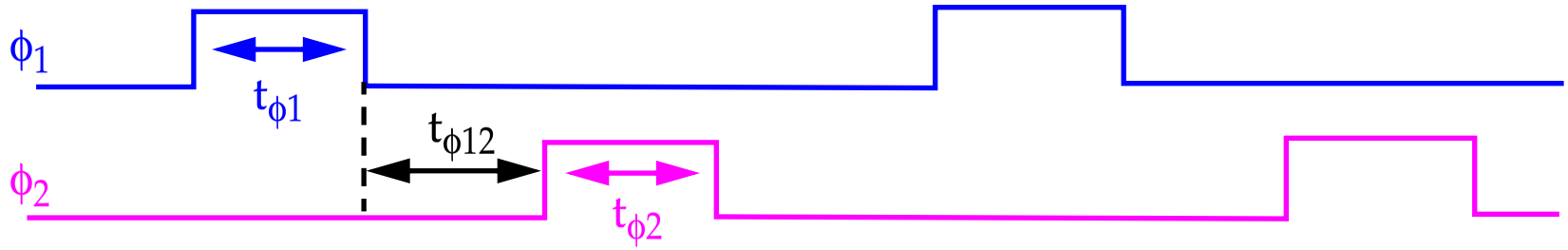

Clock skew is a dominant problem in current high performance designs.

## CMOS Dynamic Two-Phase Flip-Flops

**Pseudostatic FF**: The fix is to use two non-overlapping clocks $phi_1$ and $phi_2$:




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

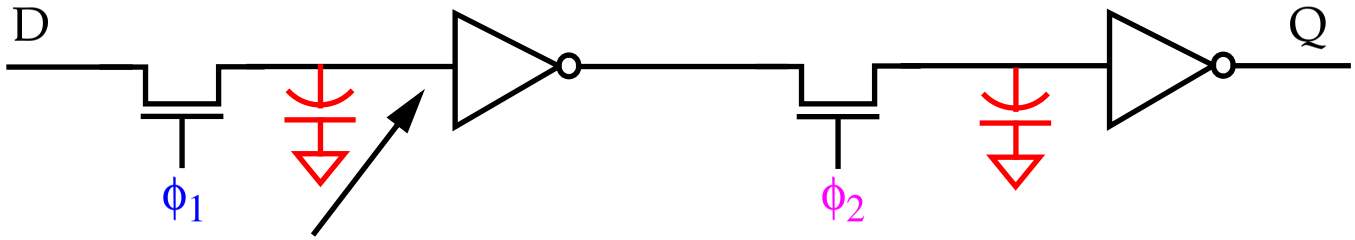



Note that node *A* floats (dynamic) during the time period $t_{phi\text{-}12}$ but is driven during $t_{phi\text{-}1}$ and $t_{phi\text{-}2}$.

Hence, the name *pseudostatic*.

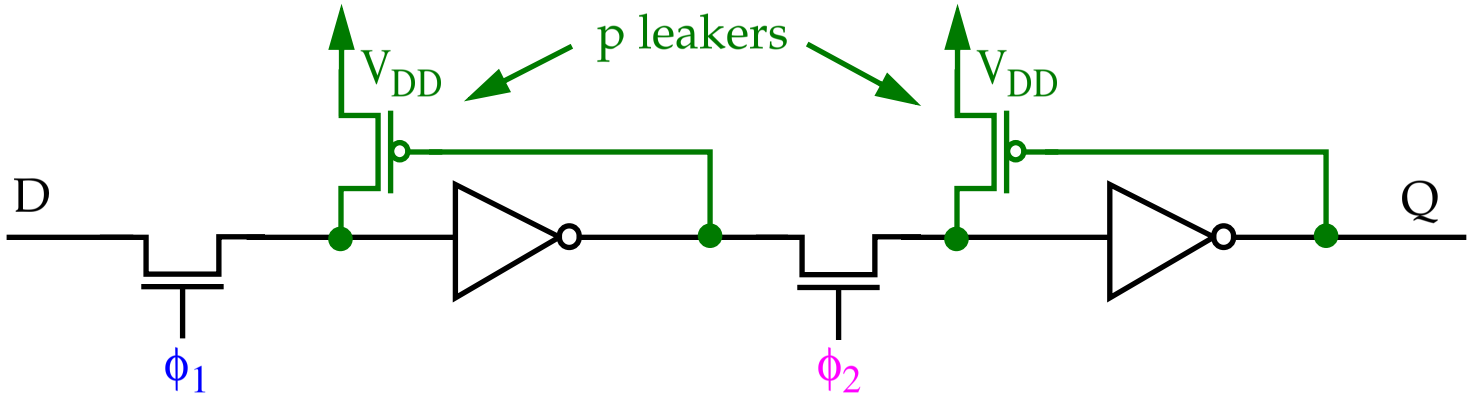
**CMOS Dynamic Two-Phase Flip-Flops**

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).

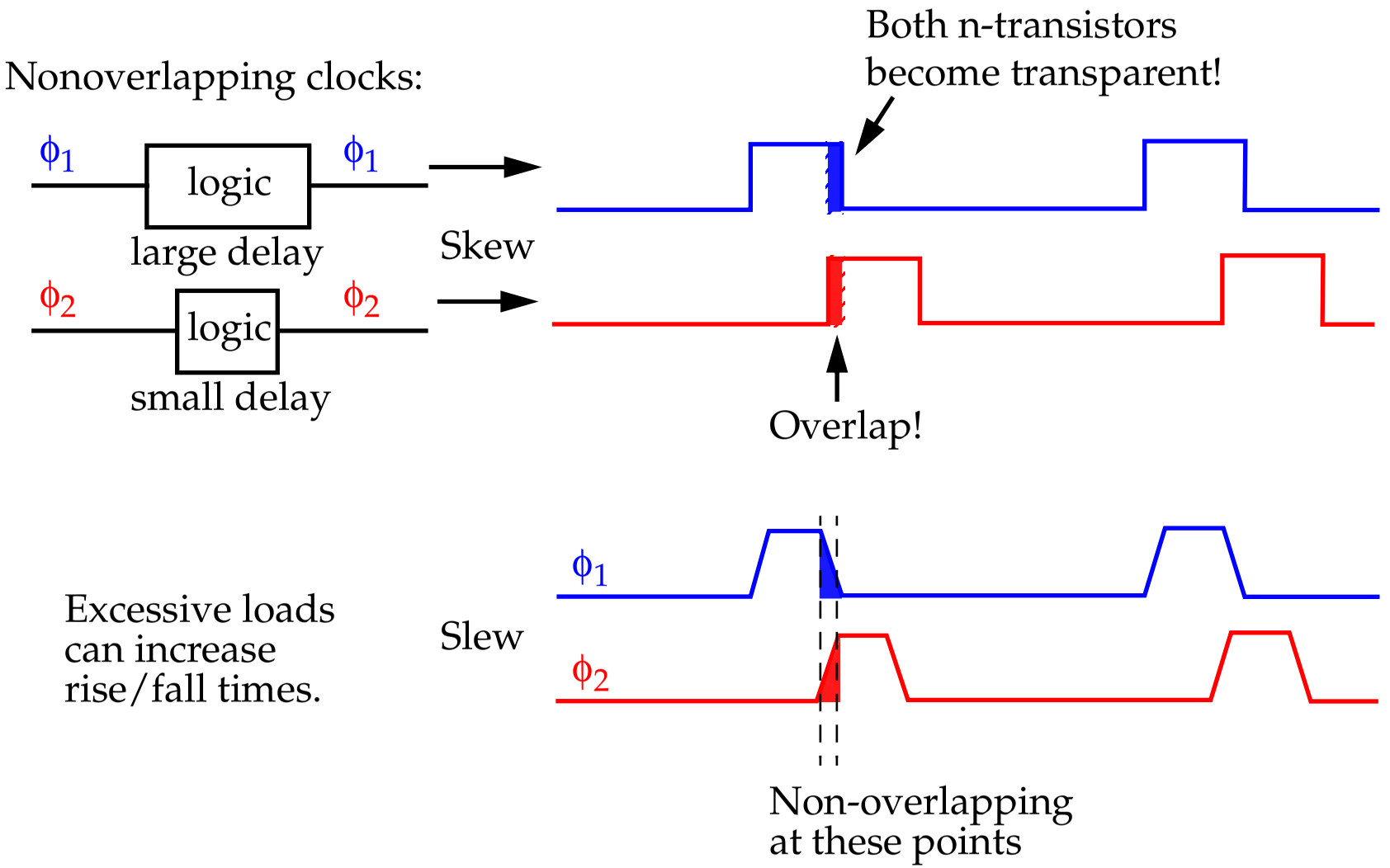



**p leakers** provide fully restored logic levels.

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

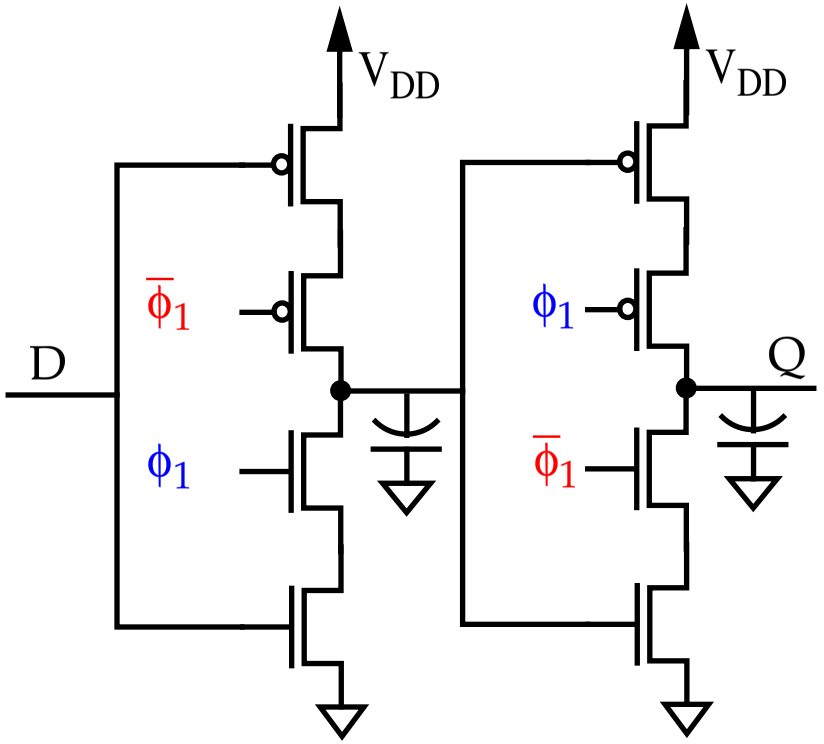
**Two-Phase Clocking**

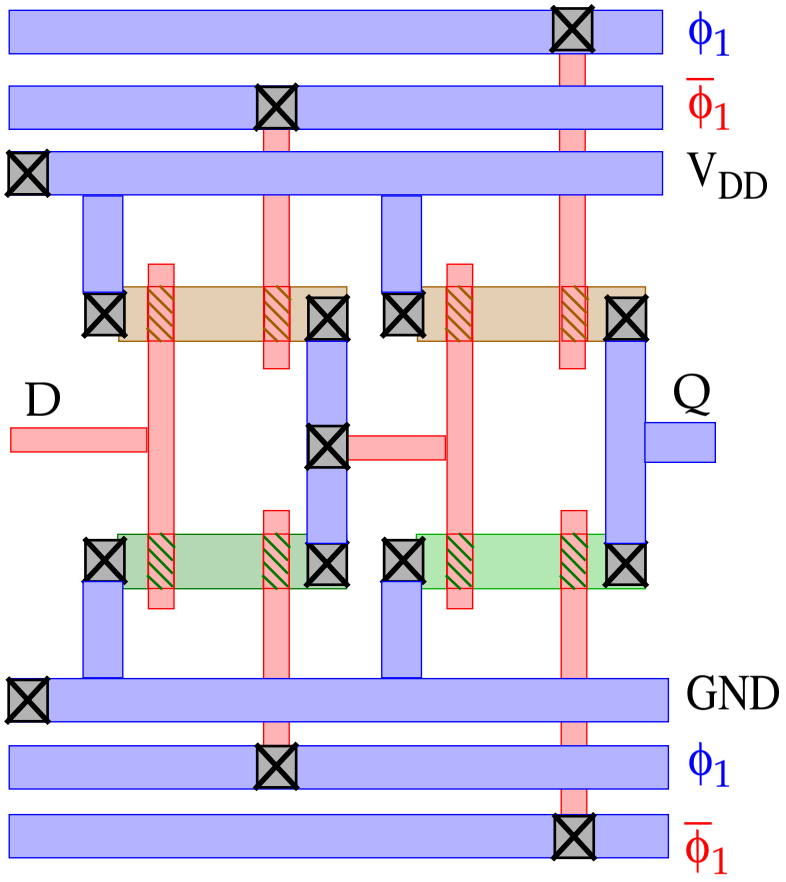
Clock skew/slew:



## CMOS Dynamic Two-Phase Flip-Flops

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

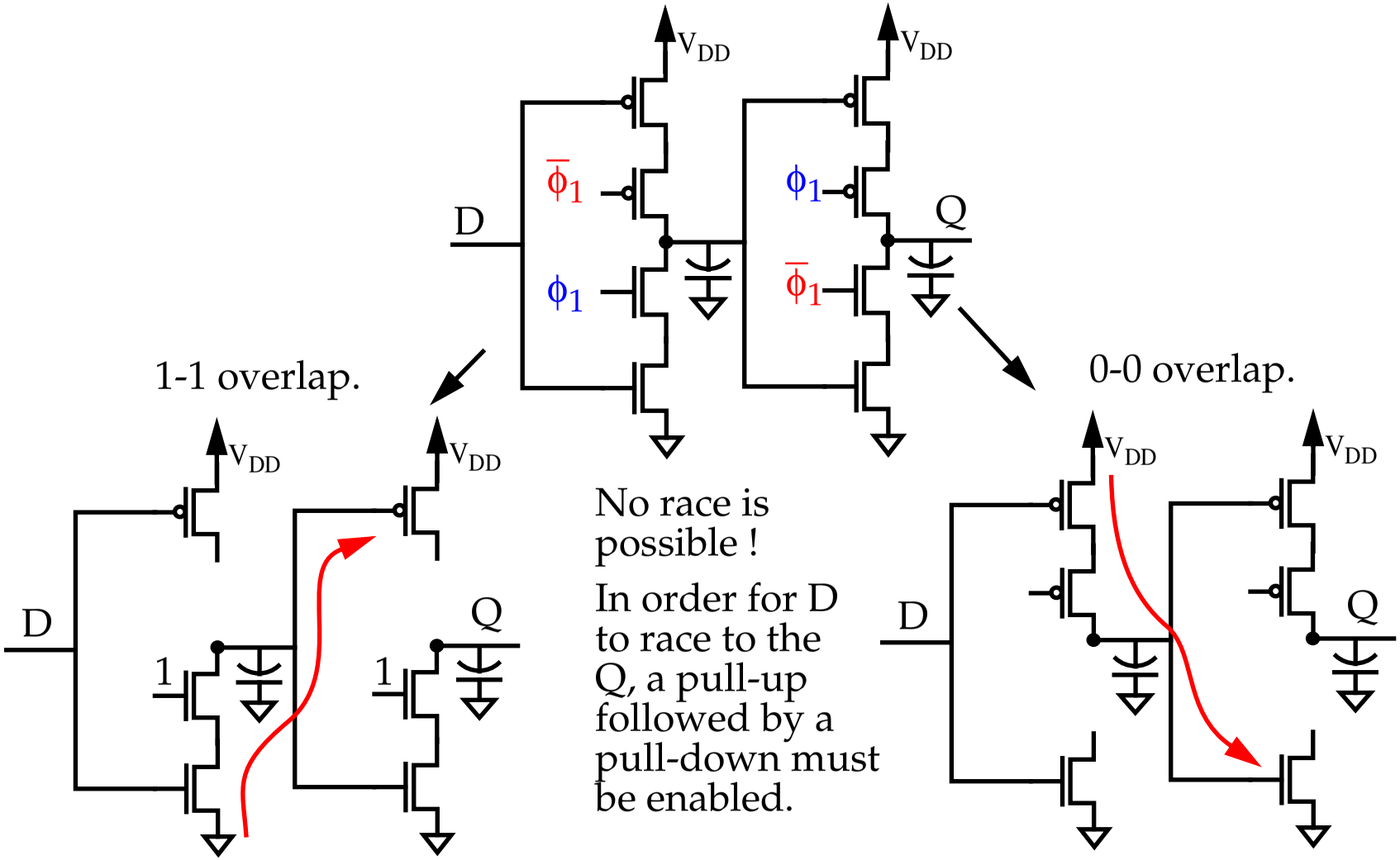



Note: Dual phase version is identical except $\phi_2$ and $\overline{\phi}_2$ are used to drive the n/p-trans in the right inverter.

## CMOS Dynamic Two-Phase Flip-Flops

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



## $C^2MOS$ Flip-Flop

**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.

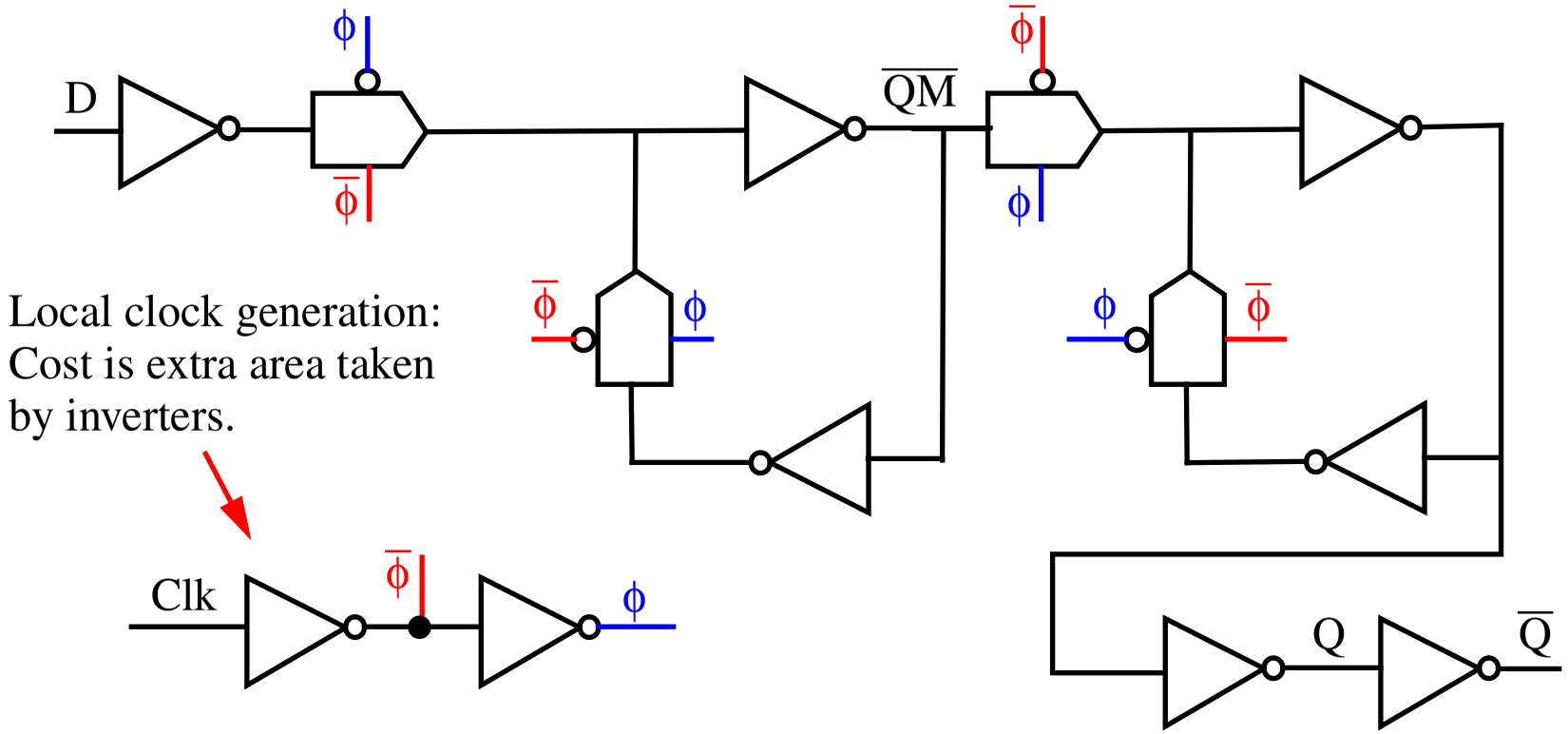
However, if the rise and fall times of the clock are slow, there exists a time slot in which both n- and p-transistors are conducting simultaneously.

Correct operation requires the 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.

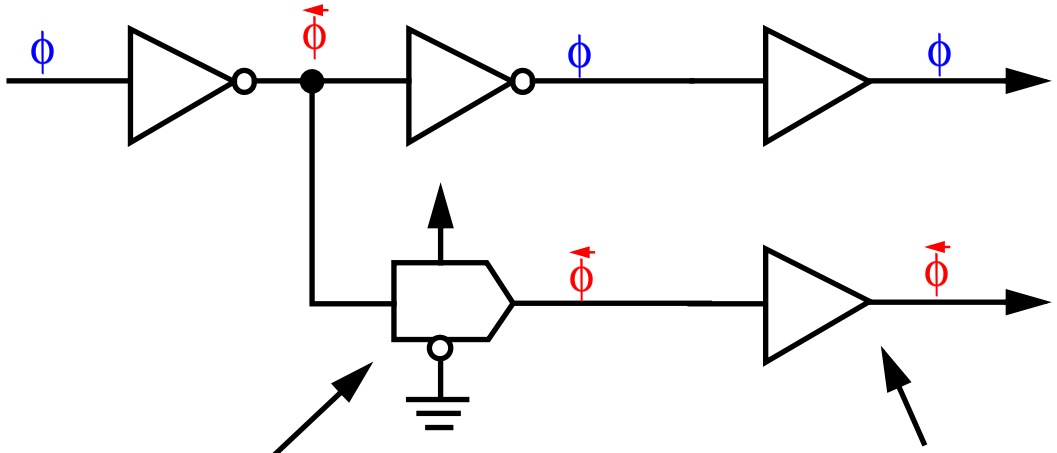
Lots of other possible latch configurations, static and dynamic -- see Weste and Eshraghian.

## Single Phase Local Clock Generation




Clock skew is minimized but area cost is severe.

## Single Phase Global Clock Generation




Pass $\phi$ through a transmission gate as a means of calibrating for $\phi$'s inverter delay.

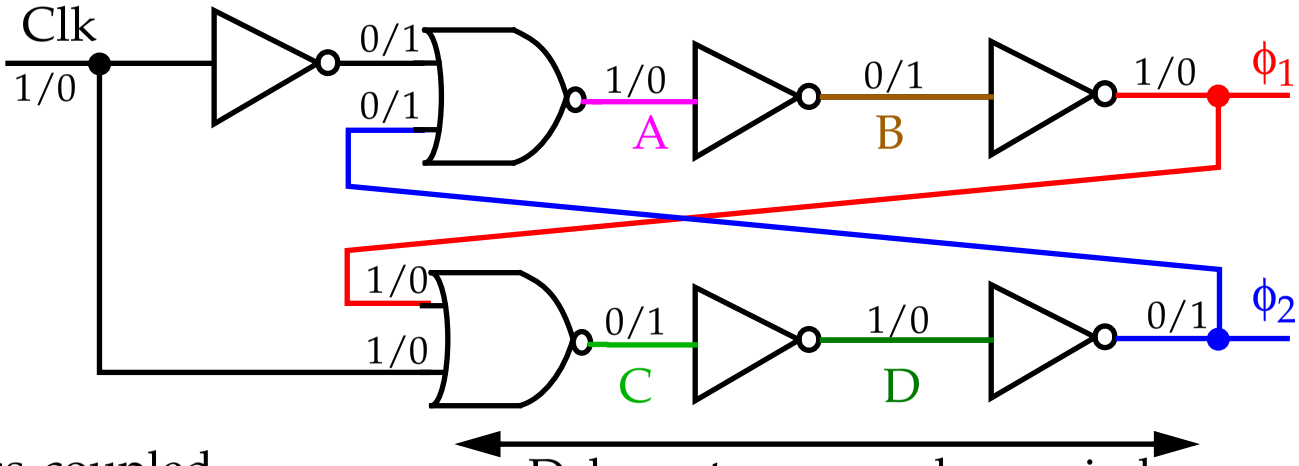
Buffers to drive large loads.

Transistors in the inverter and pass gate should be similar in size.

Keep them small and use buffers to drive the load.

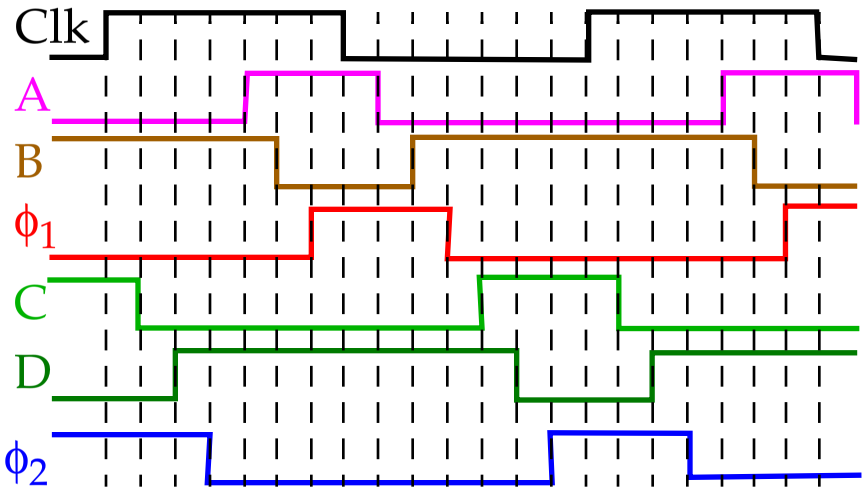
Note: The routing load MUST also be balanced on each of the clk lines.

## Two-phase Global Clock Generation




Cross-coupled RS flip-flop

Delay sets nonoverlap period.
(time period in which both $\phi_1$ and $\phi_2$ are 0).



**Multi-Phase Clocking**

*Four-phase* clocking strategies discussed in Weste and Eshraghian.

Modern designs tend to minimize the number of clock phases used due to problem of generating and distributing multiple clocks.

*Single phase* schemes used for complex, high-speed CMOS circuits.

**Clock Distribution**

Assume all the registers in a large CMOS design result in a capacitive load of 2000 pF. What is the peak current and average dynamic power?

$$V_{DD} = 5V$$

$$C_{register} = 2000pF \text{(20K register bits @0.1pF)}$$

$$T_{clock} = 10ns$$

$$T_{(rise)/(fall)} = 1ns$$

$$I_{peak} = C\frac{dv}{dt} = \frac{2000\times10^{-12}\times 5}{1.0\times10^{-9}} = 10A$$

$$P_d = CV_{DD}^2 f = 2000\times10^{-12}\times 25\times 100\times10^{6} = 5watts$$

Two techniques:

- A single large buffer (cascaded inverters): Use when the module has a large number of diverse modules, i.e. a microprocessor.
- A distributed-clock-tree: Use when design is highly structured and repetitive, i.e. a datapath.