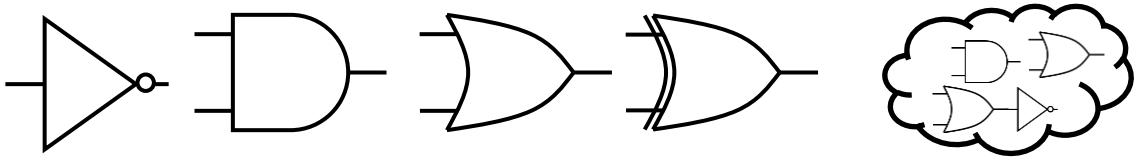
EE-429
Fundamentals of VLSI Design

Basic Sequential Elements

Andreas Burg

Fundamental Components of Digital Circuits

- Combinatorial logic: memoryless
  - Built from basic Boolean logic gates
  - Output is only a function of the current input
  - Combinational logic has no state
- Nets (wires): memoryless
  - Connect components, carrying logic signals
  - One input (driver) and one or multiple outputs



A	Z
0	1
0	0

NOT

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

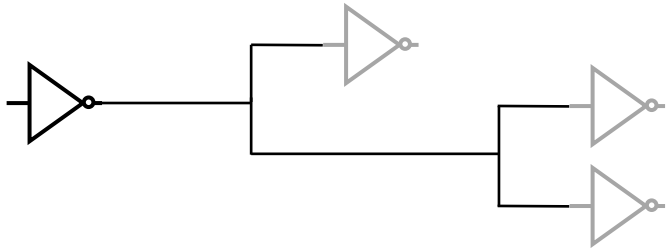
OR

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

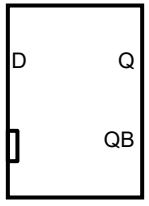
XOR

A	B	Z
.	.	.
.	.	.
.	.	.
.	.	.

Any arbitrary Boolean function

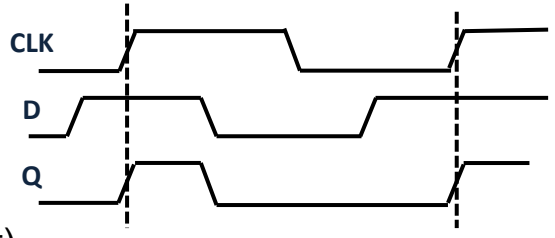


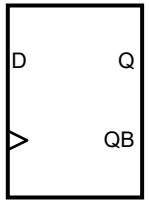
- **Storage (sequential) elements: memory**
  - Output depends on current input *and* on an internal state, defined by previous inputs



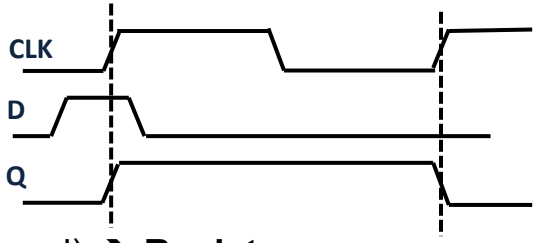
CLK	D	Q	QB
0	-	Q	QB
1	1	1	0
1	0	0	1

Latch
(high transparent)





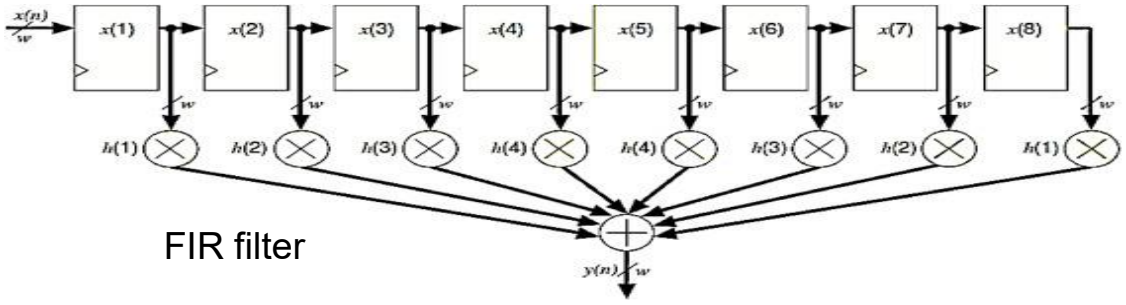
CLK	D	Q	QB
0	-	Q	QB
1	-	Q	QB
↑	0	0	1
↑	1	1	0



Flip-Flop (positive-edge triggered) ➔ **Register**

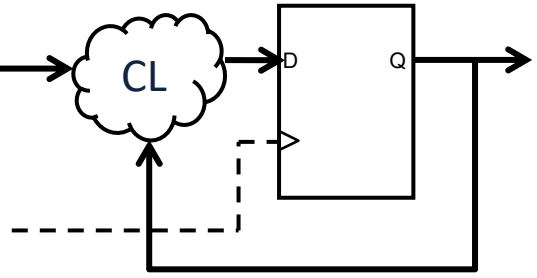
Why do we need Sequential Elements

- **Functional storage:**
  - Collect data that arrives in sequence and is required as a whole

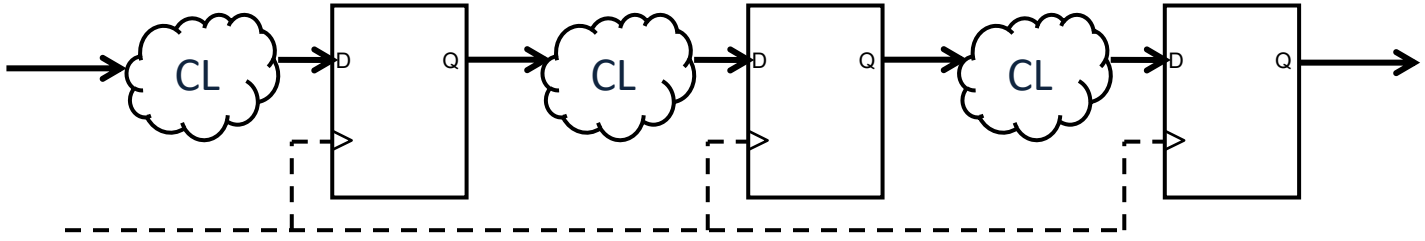


FIR filter

- **Non-functional storage**
  - Re-use of logic for different data over time for resource sharing

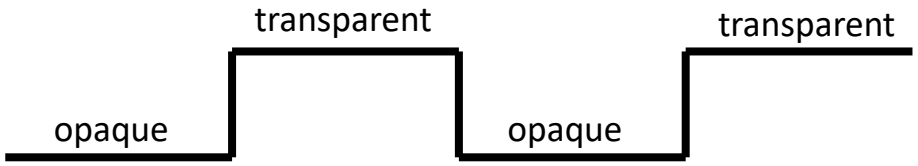


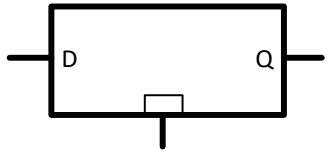
  - Isolation of parts of the logic to operate in parallel through pipelining



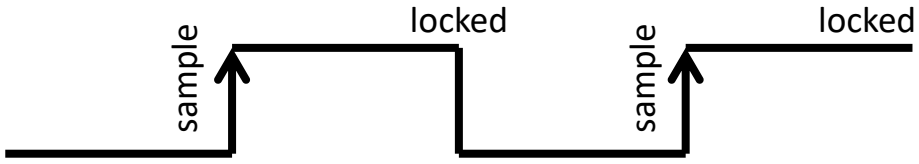
Naming Conventions

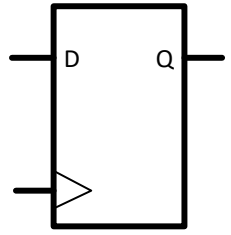
- **Two types of sequential elements:**
  - a **latch** is **level sensitive**





  - a **register** or a **flip-flop** is **edge-triggered**

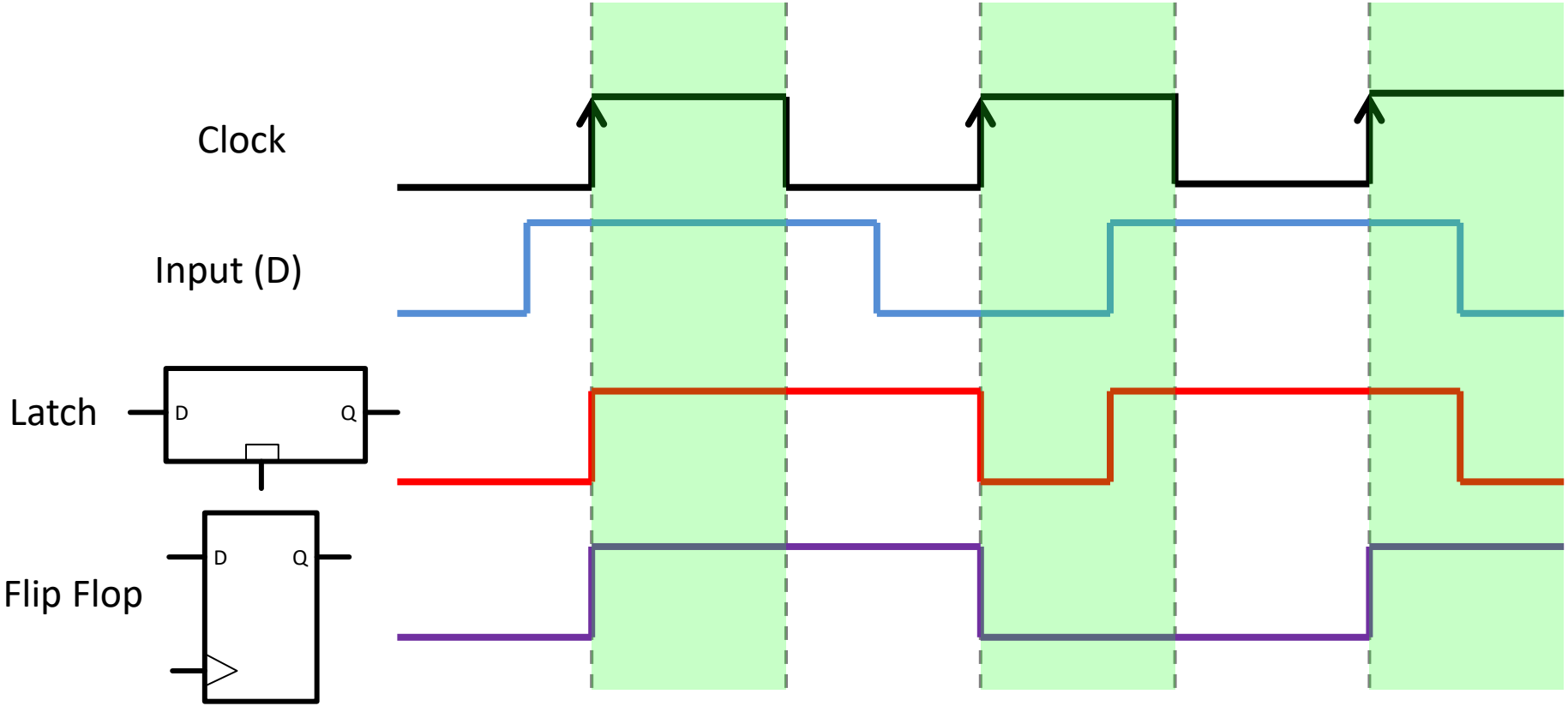




- **There are many different naming conventions**
  - For instance, many books call any bi-stable element flip-flops (such as an SR Latch)
  - However, this **leads to confusion**, so **we will use the convention above** (as used in industry).

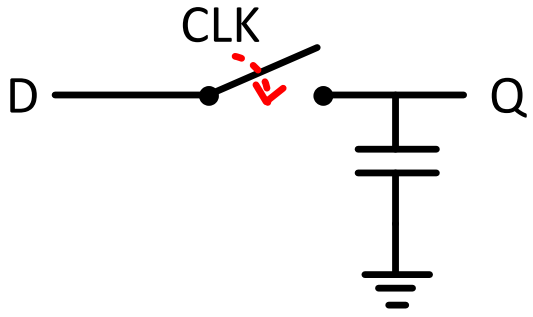
Latch Vs. Register

- A **Latch** is **transparent during one phase** of the clock **and keeps its output** (irrespective of the input) **during the** other clock **phase**.
- However, a **Flip Flop** only **samples the input on the edge.**

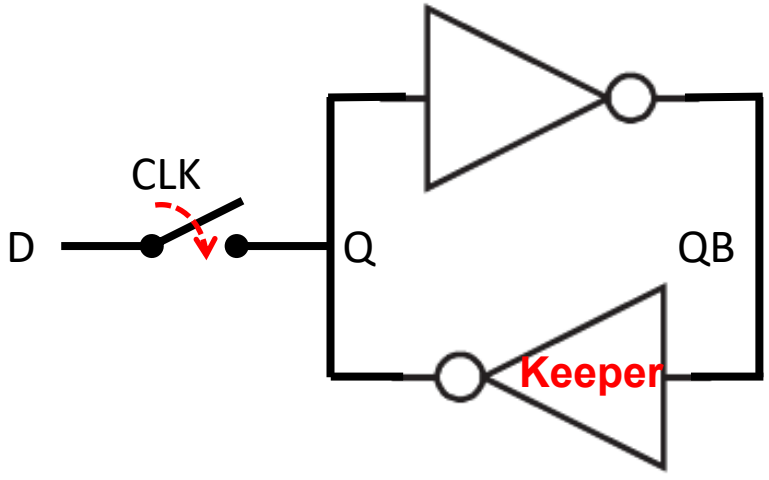


Static Vs. Dynamic Latch

- **Dynamic storage elements store data** (the state) **as charge** on a capacitance
  - No active element involved in keeping the state
  - Since charge leaks away, the stored data degrades over time
- **Static storage elements include a feedback** to constantly restore the state
  - **Active feedback** involved in keeping the state: **KEEPER**



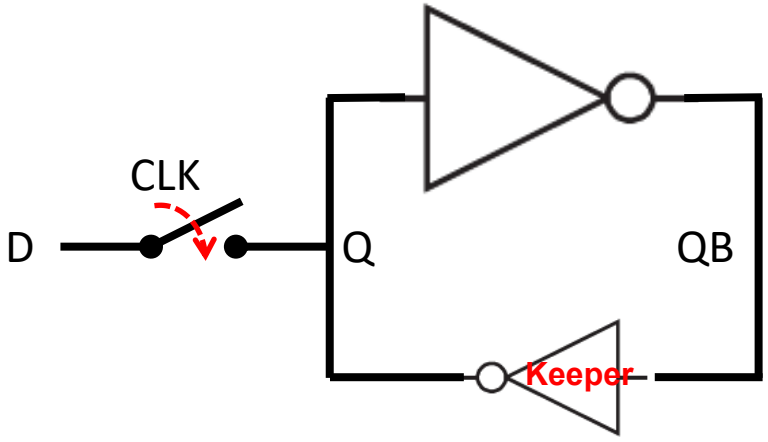
Dynamic Latch



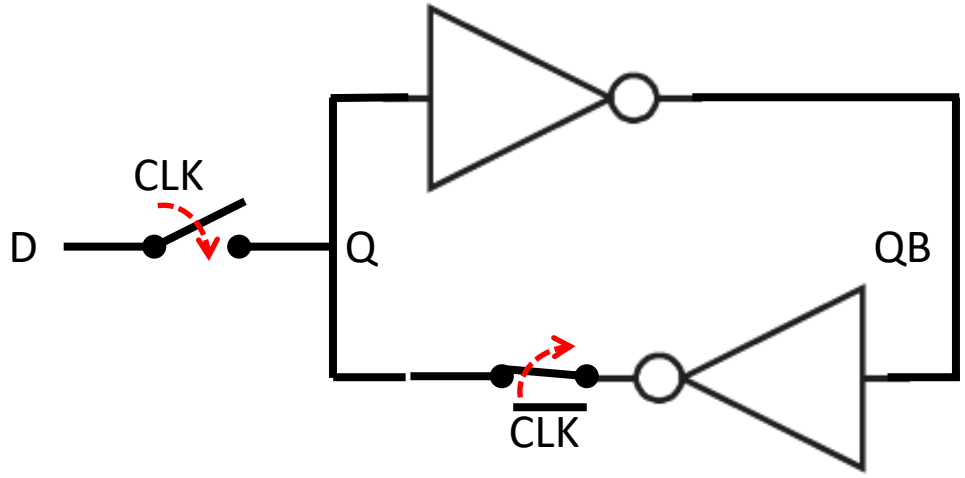
Static Latch

Ratioed vs. Non-Ratioed Latch

- **Writing** to the latch **involves overriding its** internal **state**
  - **Driver must prevail** over the internal keeper
- **Ratioed latch: strong driver overrides a weak keeper**
  - Sensitive to drive strength of the components
- **Non-ratioed logic: deactivate keeper while writing**



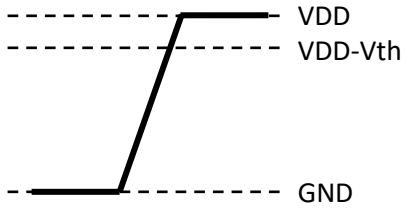
Static
Ratioed
Latch

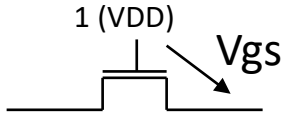


Static
Non-ratioed
Latch

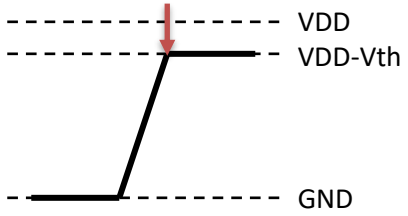
Disconnecting the Feedback Driver

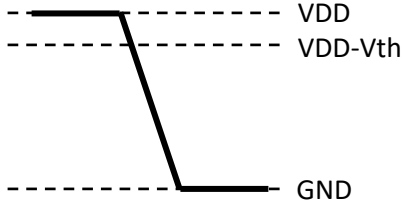
- **Using a series transistor as a switch**
- Which polarity to use: nMOS or pMOS ➔ **both not very well suited…**
  - nMOS can only pass a strong zero
  - pMOS can only pass a strong one

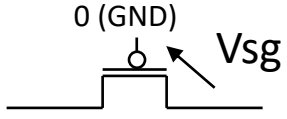




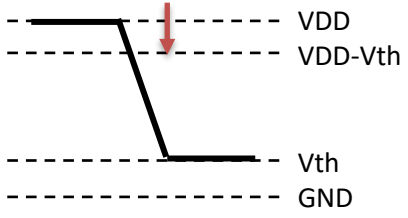
Vgs drops below Vth:
transistor turns off before
output reaches VDD







Vsg drops below Vth:
transistor turns off before
output reaches GND



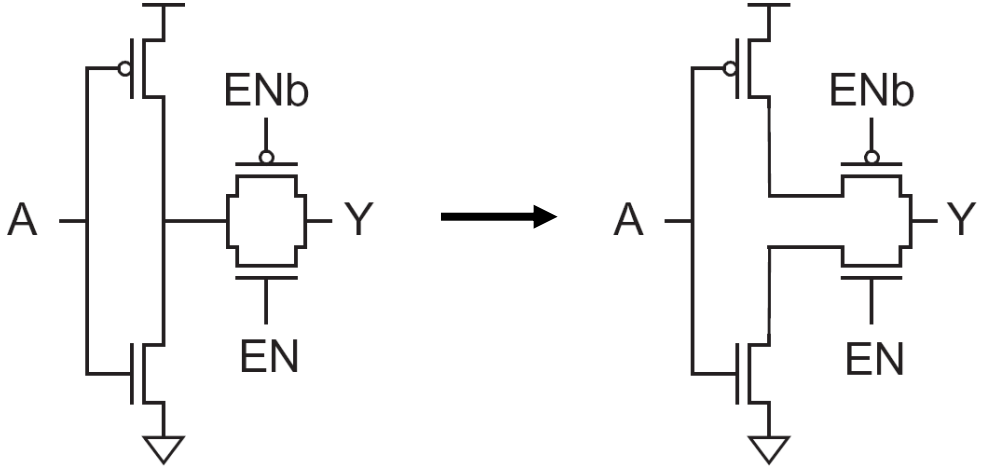
Combine nMOS
and pMOS
to achieve full
swing output

Transmission (T) Gates and Tristate-Inverter

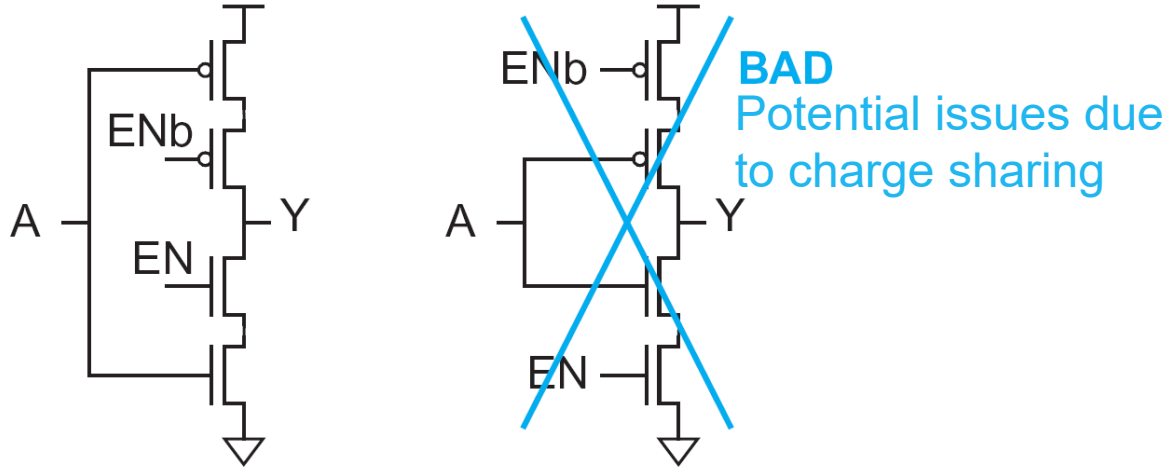
- **Driving both strong ‘0’ and ‘1’ requires a “hybrid” nMOS/pMOS switch**

- **Two almost identical options:**

Transmission Gate



Tristate Inverter



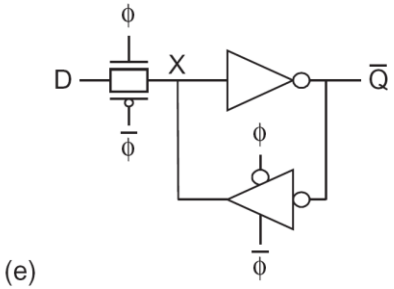
Break nMOS/pMOS connection

- Each part of the switch still used where it is most effective (almost no impact on on-resistance)
- Reduced output load

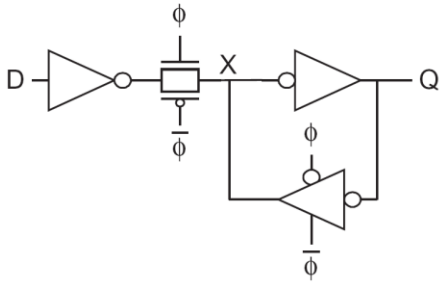
Robust Latches

- As **basic building blocks,** latches **need to be robust** and work in all conditions

- Ensuring a **clean input interface** (capacitive load)
  - Add extra input inverter to T-gate OR
  - Use a tristate input inverter

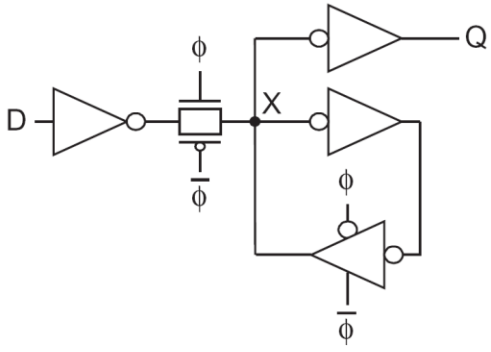


(e)



(f)

- **Protecting** the **storage node (SN)**
  - Any spikes (e.g., from coupling or kickback) on the output (from external circuits) can alter the state
  - Shield output from the storage node with an inverter

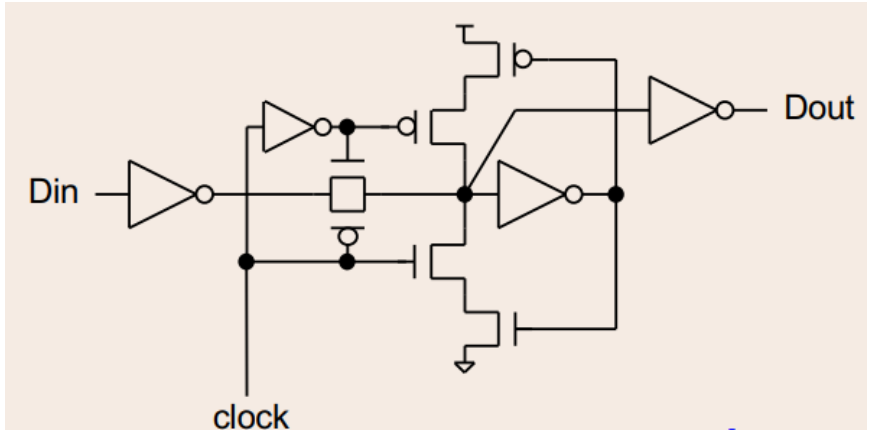


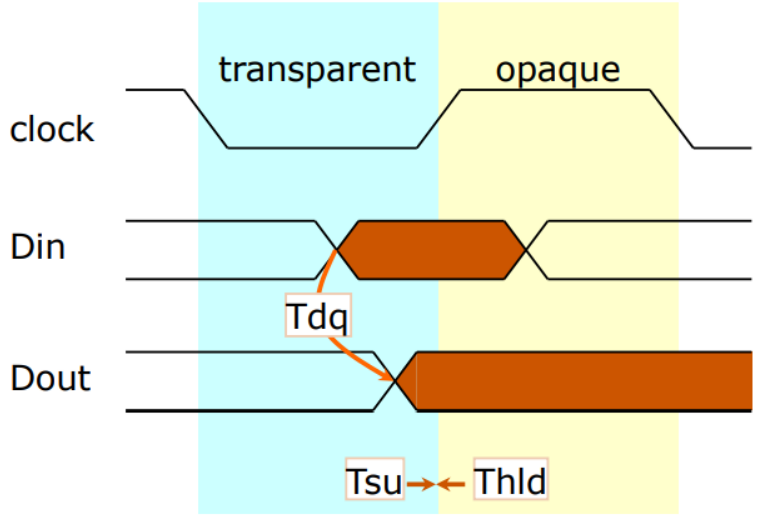
(g)

Latch Polarities

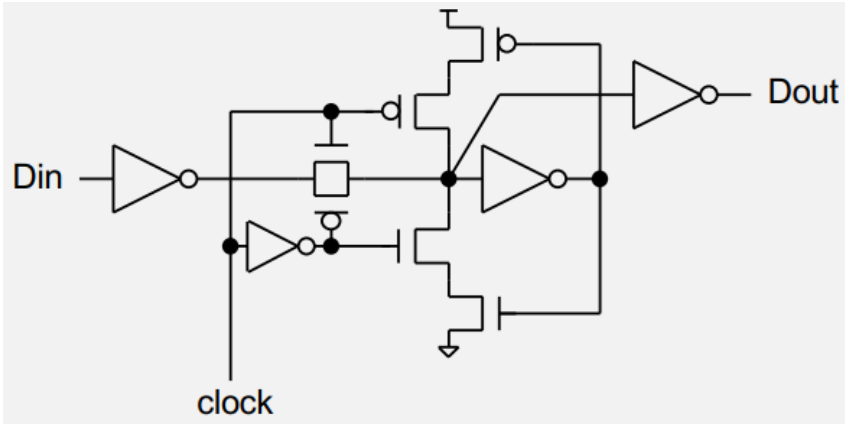
- **Latches** can be built **with different polarities**

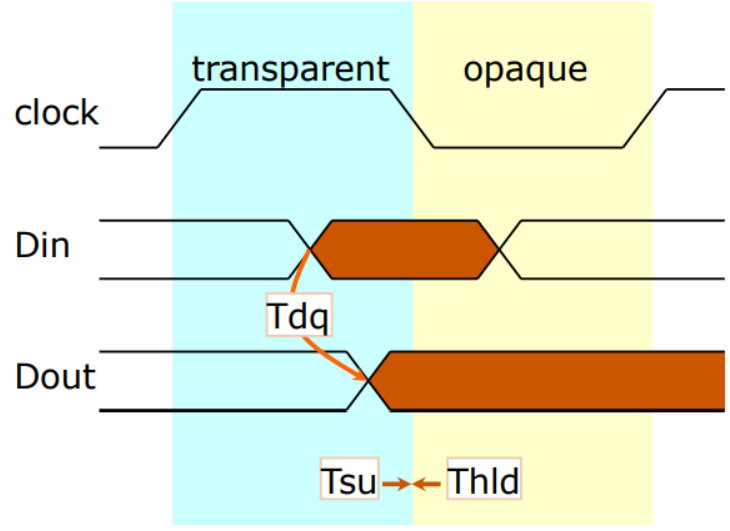
Low-Transparent





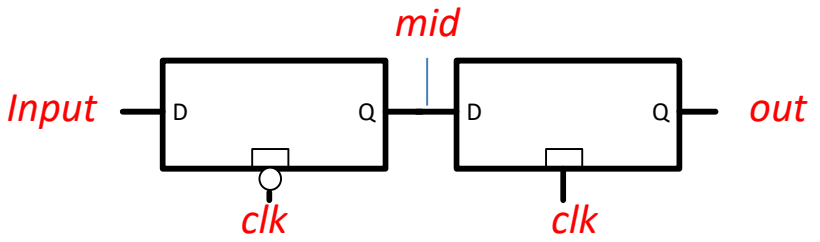
High-Transparent

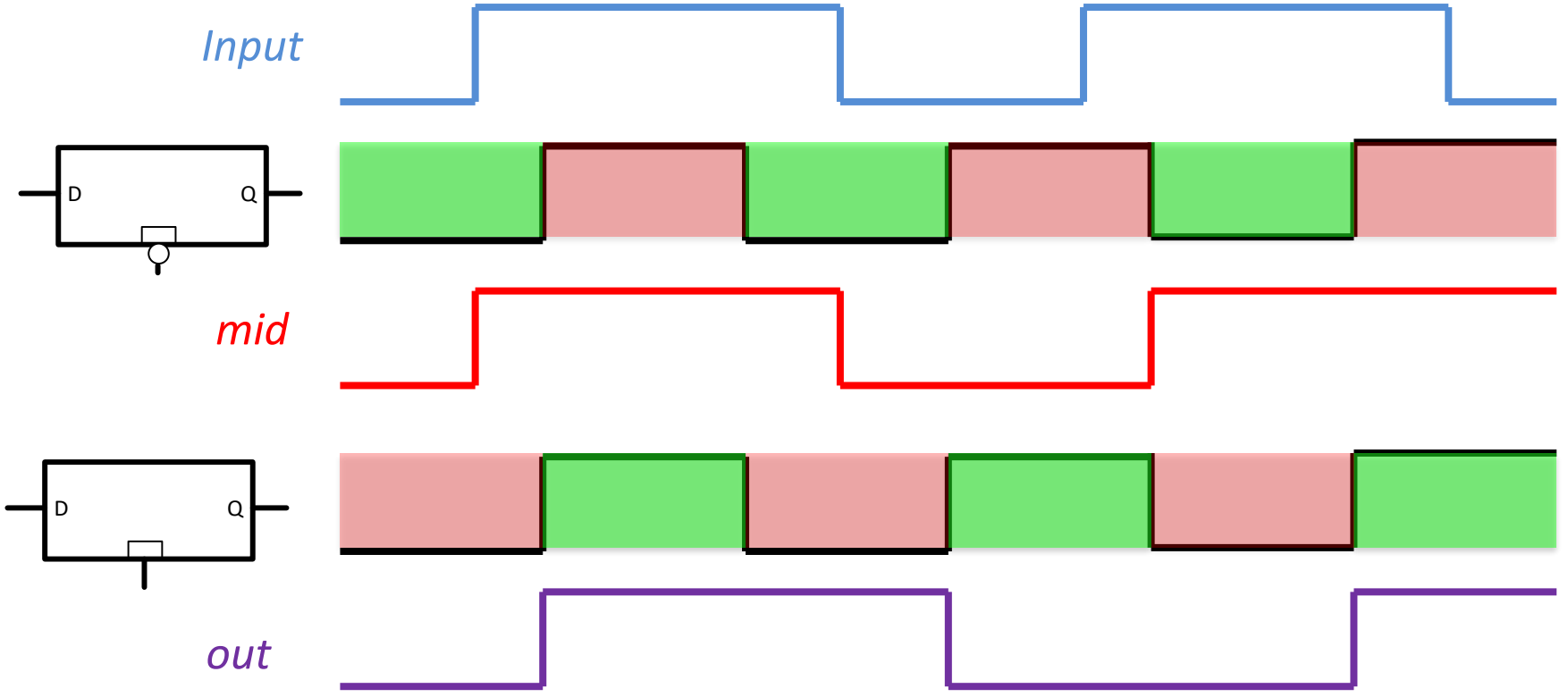




Master-Slave Register (Flip Flop)

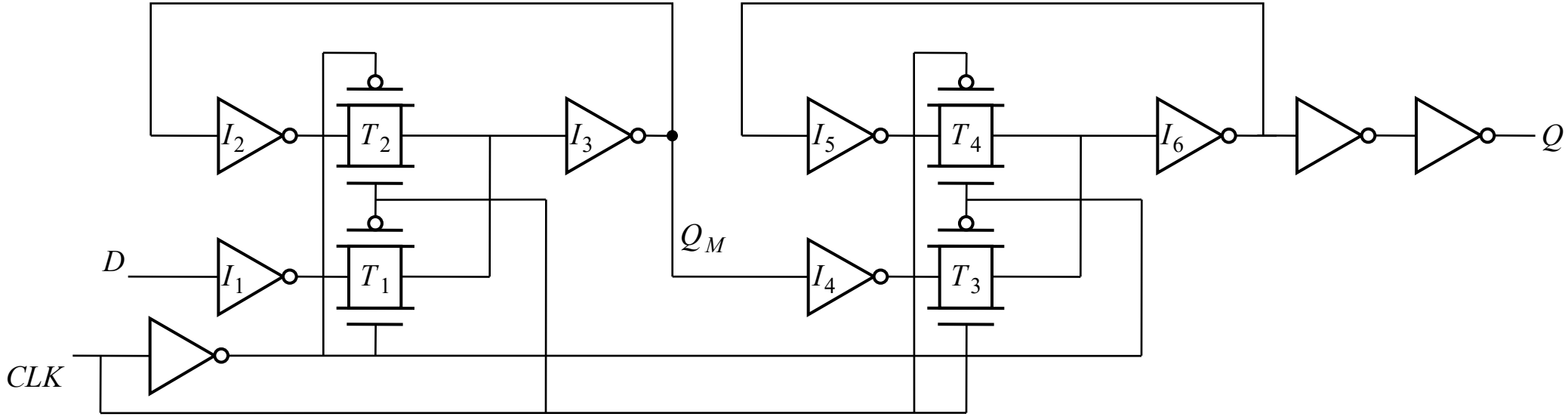
- **An edge triggered flip-flop can be built by combining two opposite-polarity latches**





Master-Slave Register

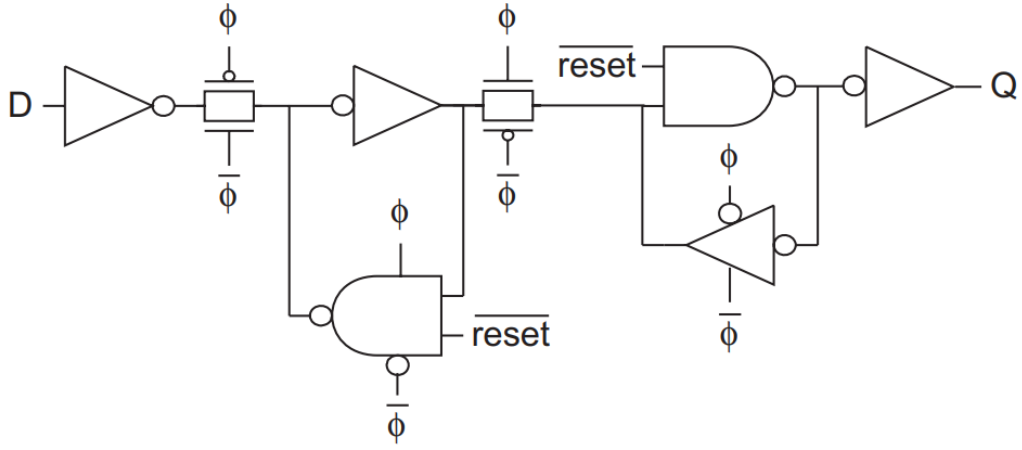
- **Basic ingredients**
  - Two tristate-based pair of latches
  - Output buffers to shield the SN
  - Input buffer to present a capacitive load
  - Output inverters to isolate the storage node



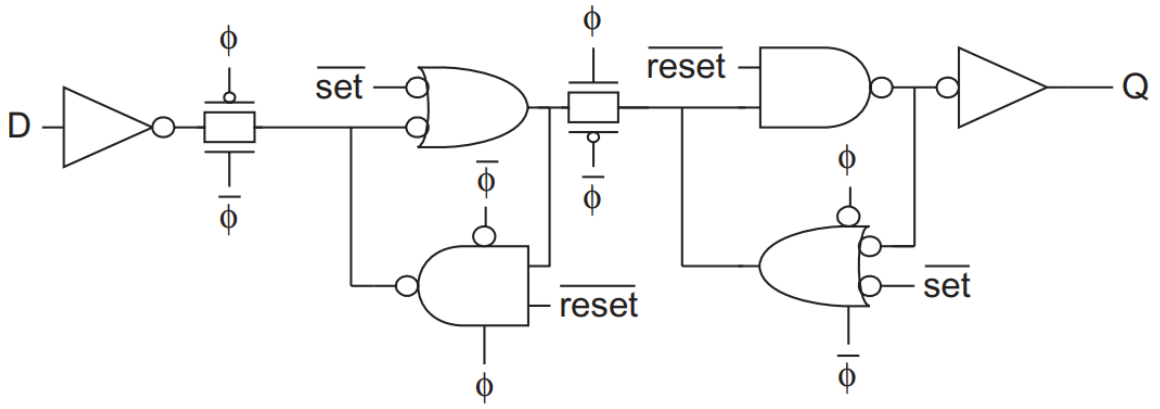
Set / Reset

- To reach a known initial state, **most FFs need to be set** and/**or** **reset**
  - Set/Reset refer to **ASYNCHRONOUS set/reset** (state change independent of clock)
  - Typically either set or reset (both is rare)

- Circuit design examples:
  - **Set/reset** required **on BOTH** master and slave **latches**



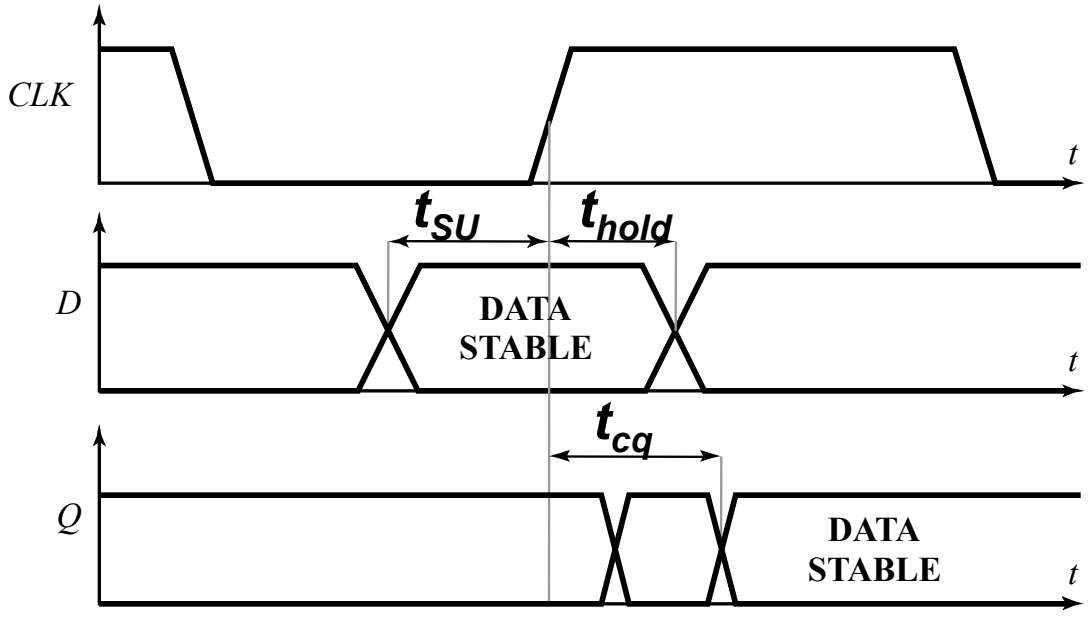
RESET
(low active)

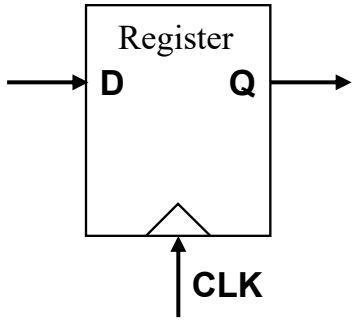


SET and RESET
(low active)

Flip-Flop Delay and Timing Requirements

- As opposed to logic, **Flip-Flops have a delay and timing requirements**
  - Propagation delay t_{cq}: defined from CLK-input to Q-output
  - Setup time t_{SU} : defined from D-input to CLK-input
  - Hold time t_{hold} : defined from CLK-input to D-input





Impact of Timing Requirements

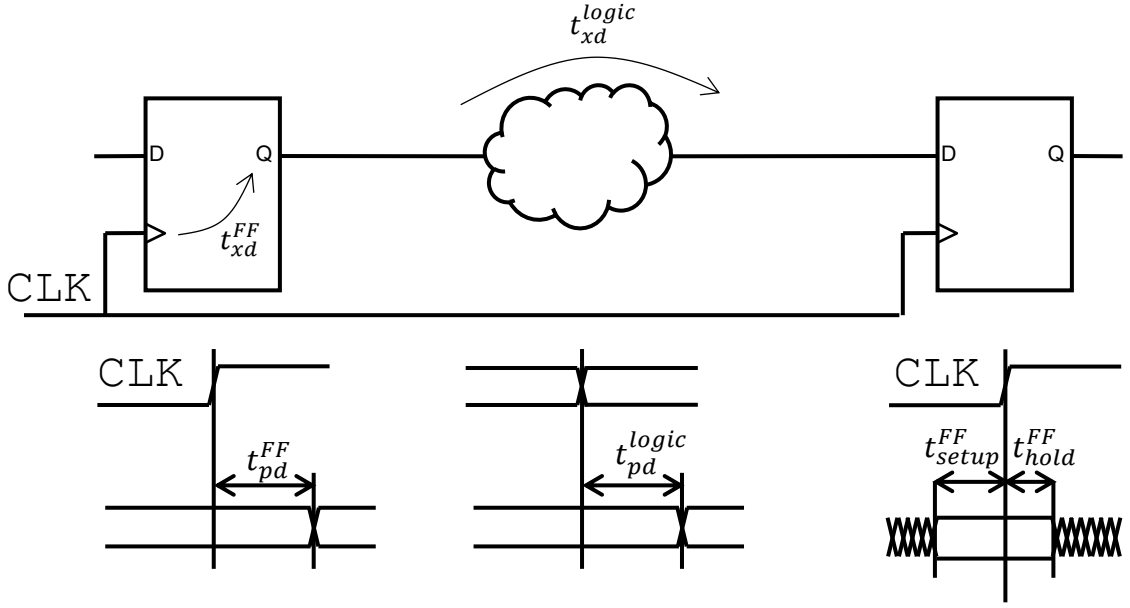
- **Timing requirements for all register-to-register paths**
  - Path **starts from the clock pin** of a register and **ends on** the **data pin** of a register
  - **Setup condition**: latest arrival

$$\underbrace{t_{pd}^{FF} + t_{pd}^{logic}}_{t_{ss}^{max}} < t_{clk} - t_{setup}^{FF}$$

Determines maximum frequency

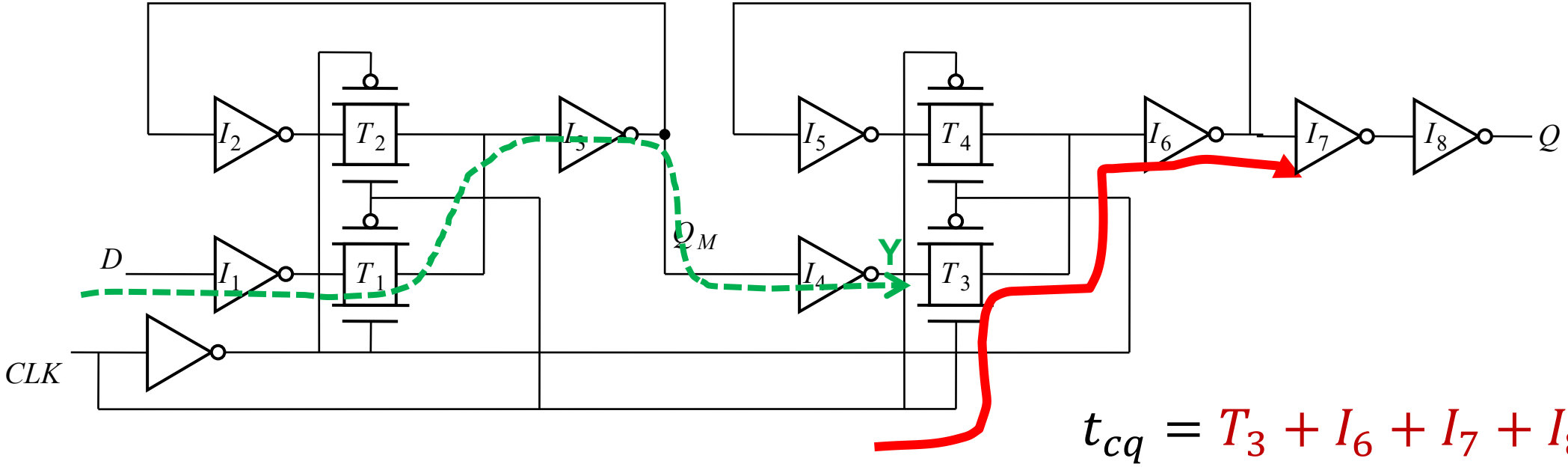
  - Hold condition: earliest arrival

$$\underbrace{t_{cd}^{FF} + t_{cd}^{logic}}_{t_{ss}^{min}} > t_{hold}^{FF}$$



Propagation Delay Calculation

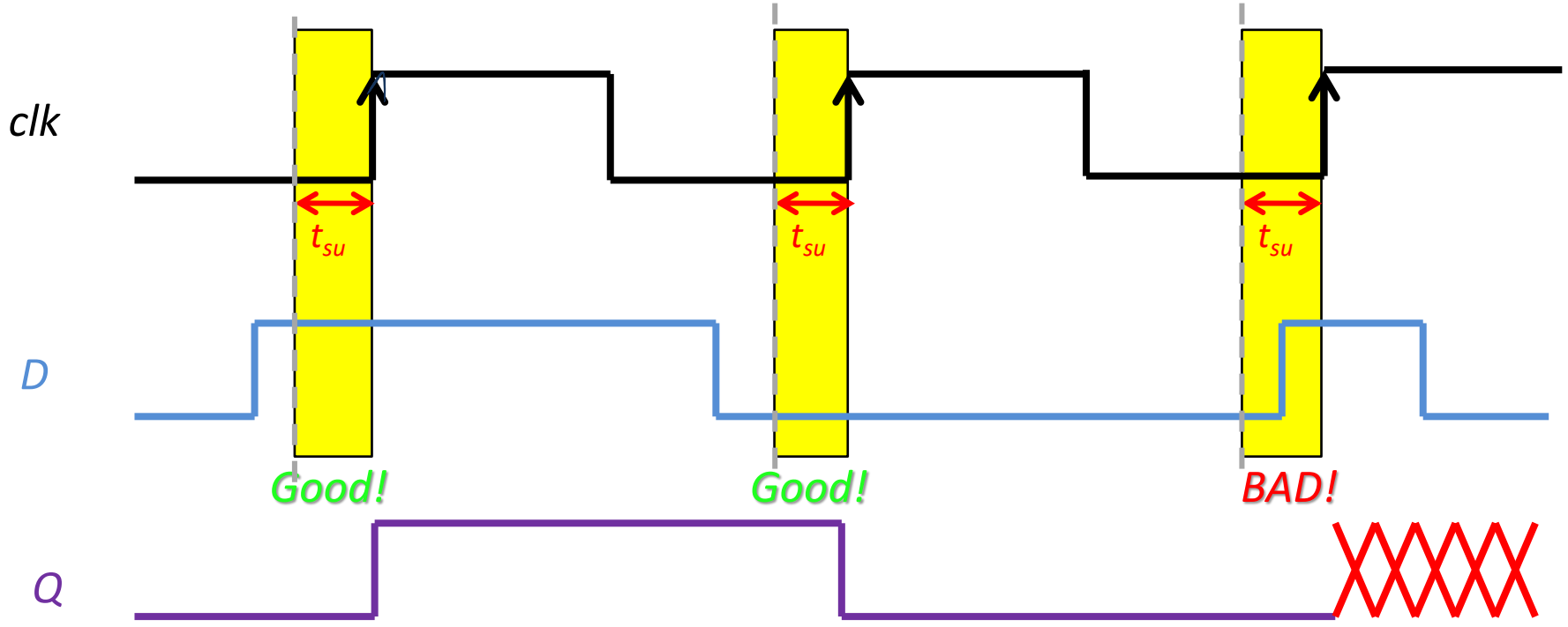
- **Output changes when slave latch turns transparent** after positive clock edge
  - New output already stable at node **Y (slave latch T-gate input)**
- **When clock rises, data still has to propagate through pass gate and inverter**



$$t_{cq} = T_3 + I_6 + I_7 + I_8$$

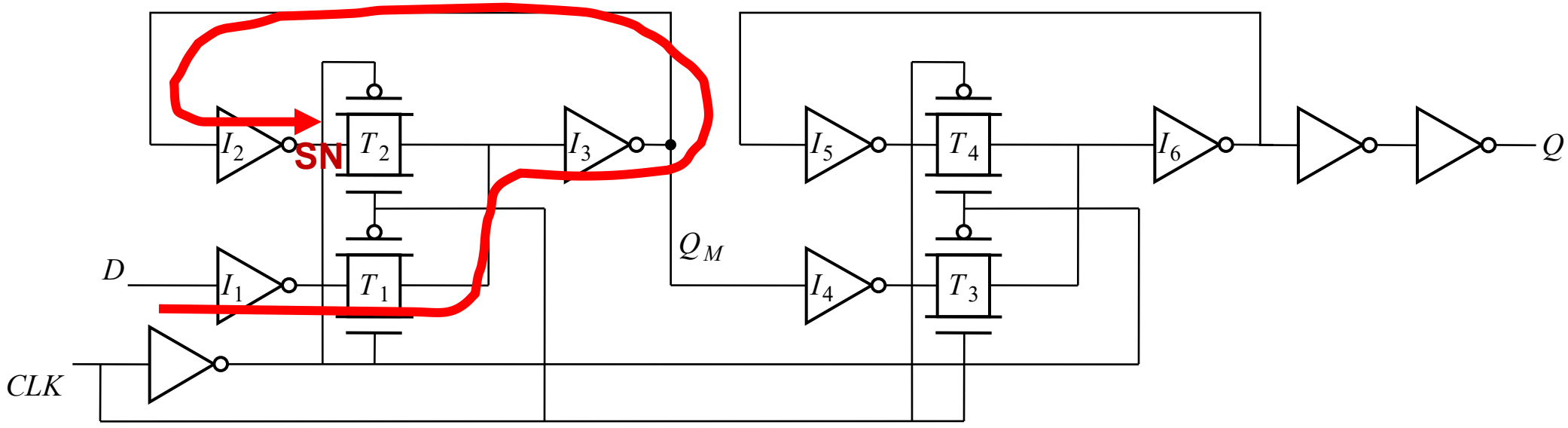
Setup Time Definition

- **Setup time is the time the data has to arrive *before* the clock** to ensure correct sampling



Setup Time Calculation

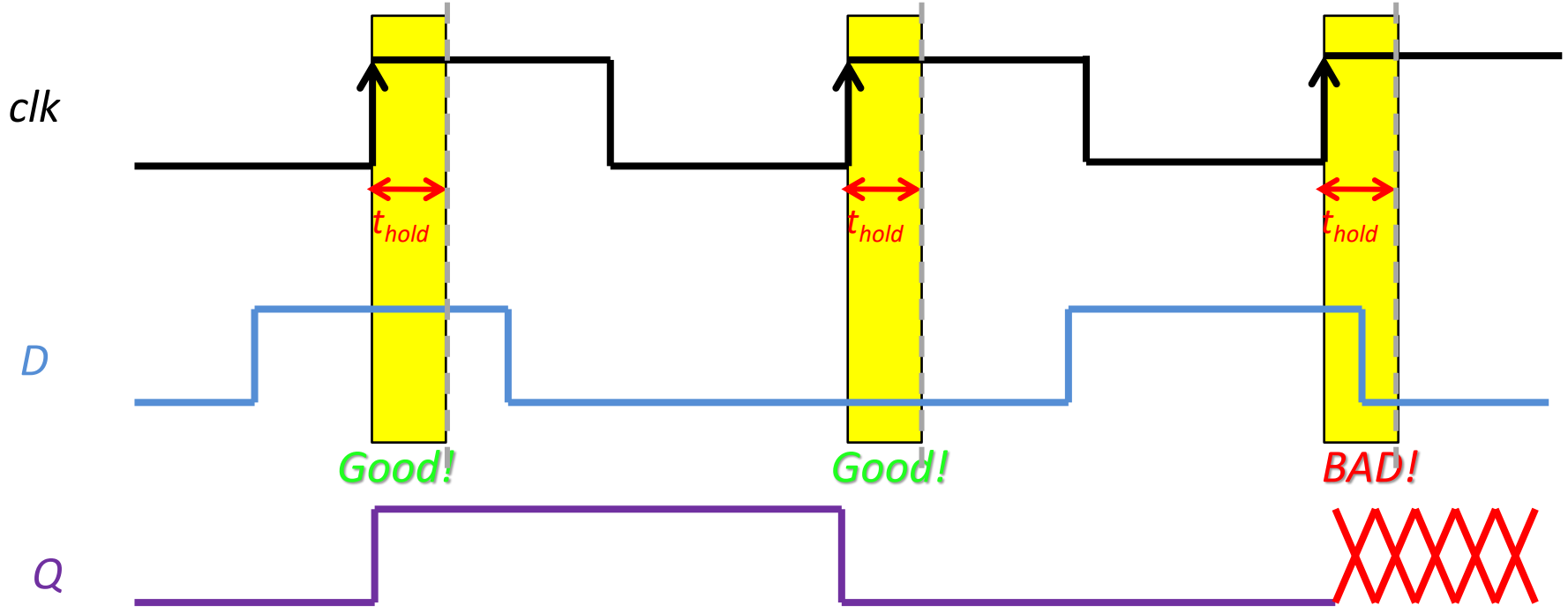
- **Master latch closes on positive clock edge**
  - Feedback loop of the **latch SN** must be stable at the new value before opening
- **Before the clock edge, data should have propagated to SN to avoid restoration of the previous state**



$$t_{su} = I_1 + T_1 + I_3 + I_2 = T + 3I$$

Hold Time Definition

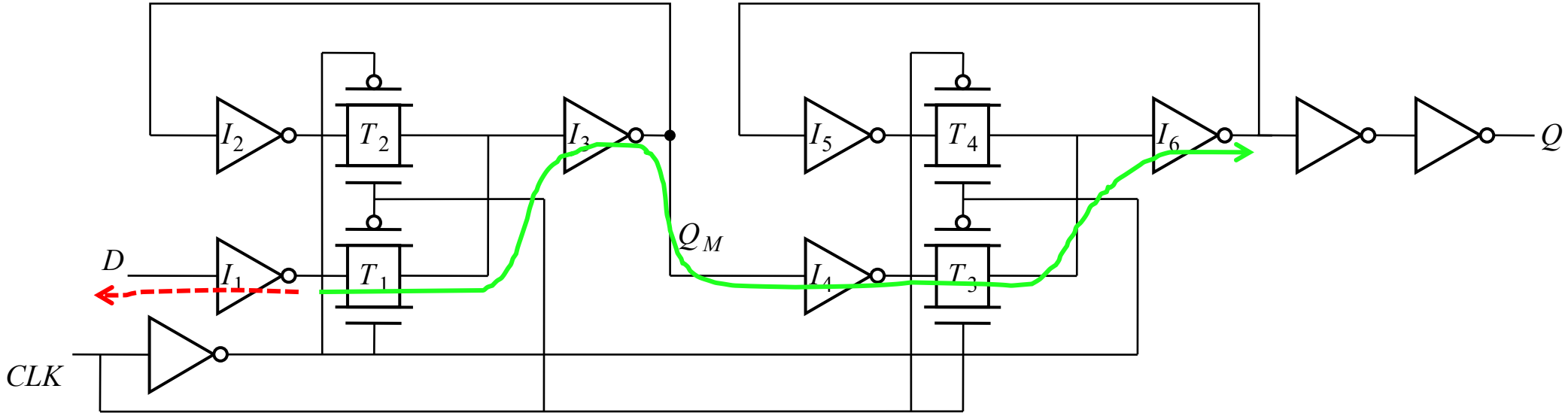
- **Hold time is the time the data has to be stable *after* the clock to ensure correct sampling.**



- **Often (optimally), Hold Time is *negative*!**

Hold Time Calculation

- **Master latch closes on positive clock edge**
  - SN must be stable until driving T-gate closes
  - Additional delays in the data path help to keep SN even after the input changes
- **The hold time can be negative**

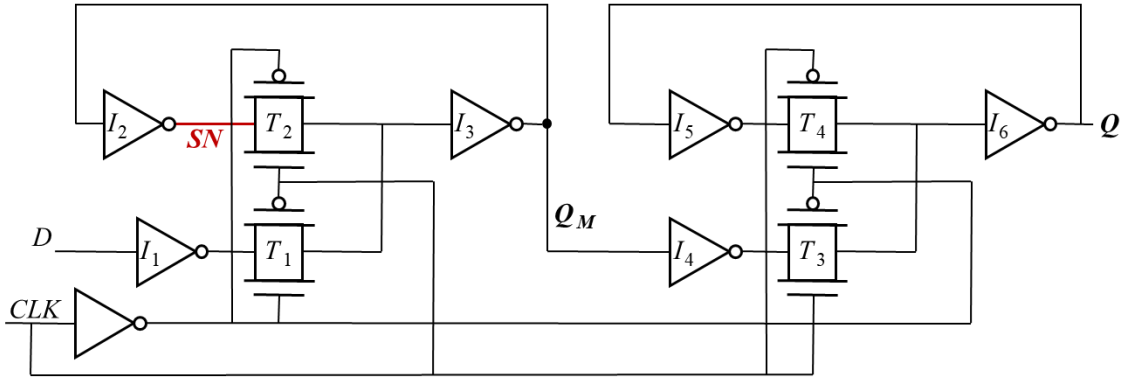


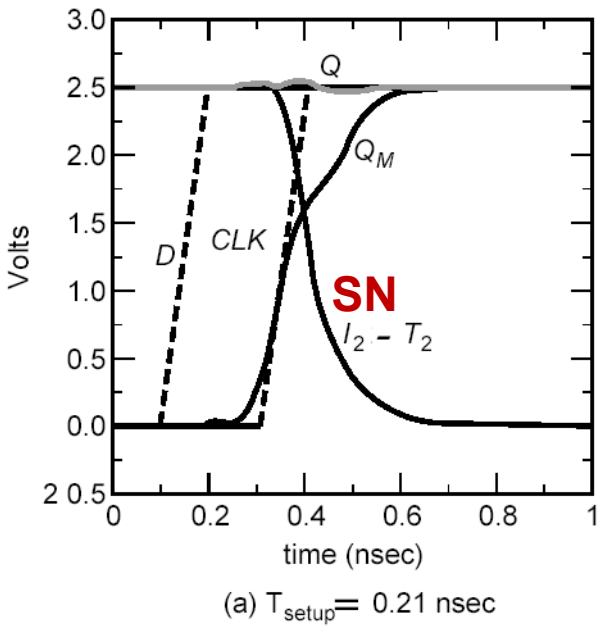
$$t_{hold} = -I_1$$

Impact of Setup Violations & Simulation

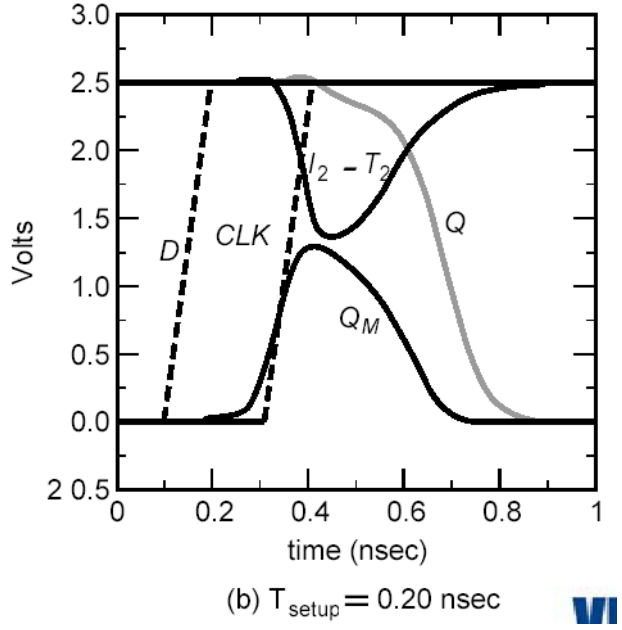
- **Example** for setup violation
  - SN: storage node (inverted)
  - QM: output of the master latch

- We **obtain the setup time** of the register **using SPICE simulations, by progressively skewing the input** with respect **to the clock edge until the circuit fails**





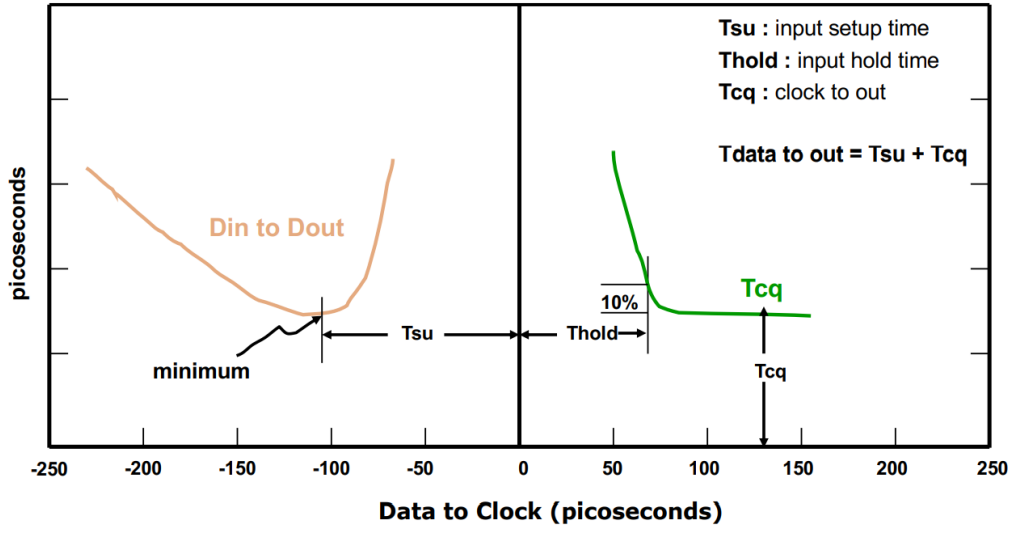
(a) T_{setup}= 0.21 nsec

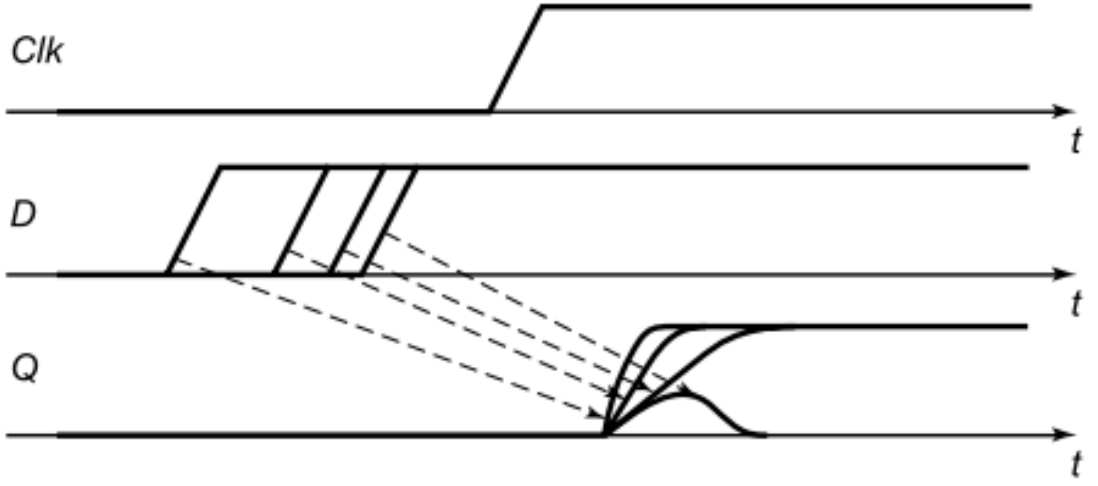


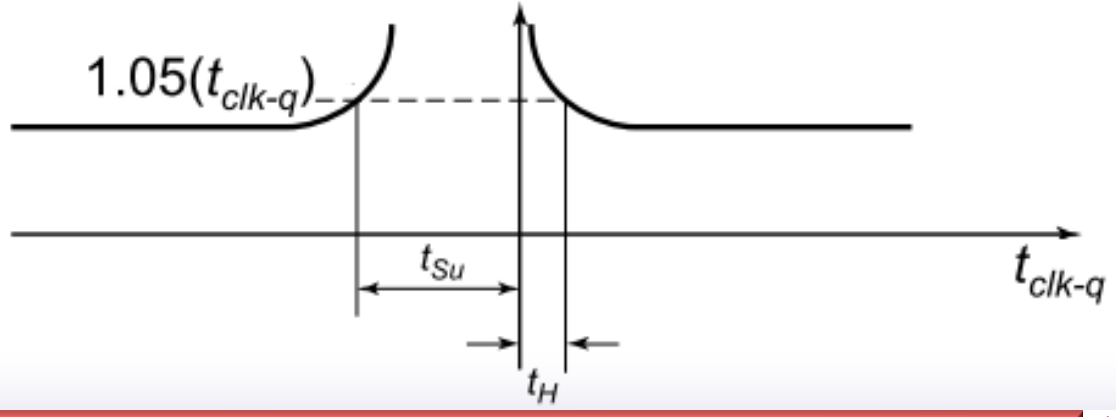
(b) T_{setup} = 0.20 nsec

Impact of Marginal Setup Condition

- **Setup violations do not always cause a complete failure (error)**
  - Marginal triggering leads to increase in the propagation delay before failing
  - Timing models do not represent this case properly
  - Constraints are for accuracy to ensure timing analysis remains valid



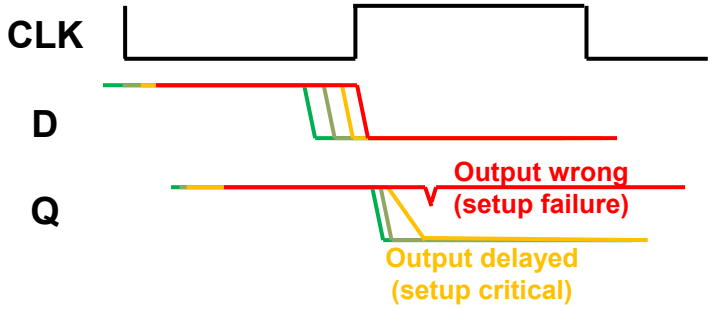




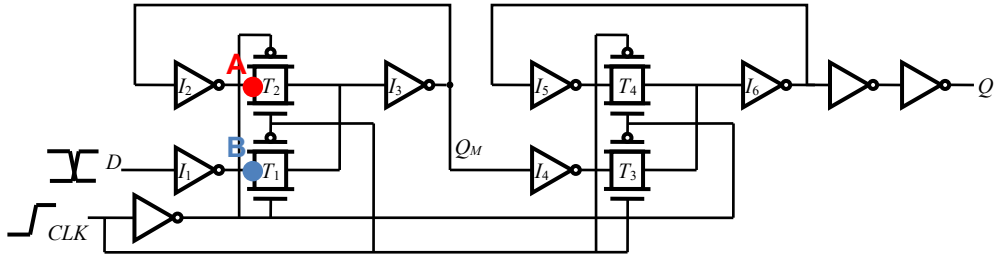
Simulation Based Setup/Hold Characterization

- **Setup/hold can not be measured directly** (requirements rather than delays)
- **Two options to bound these requirements**

1. Brute force simulations (trial and error) with pass fail constraint based on correct output or clk->Q delay constraint

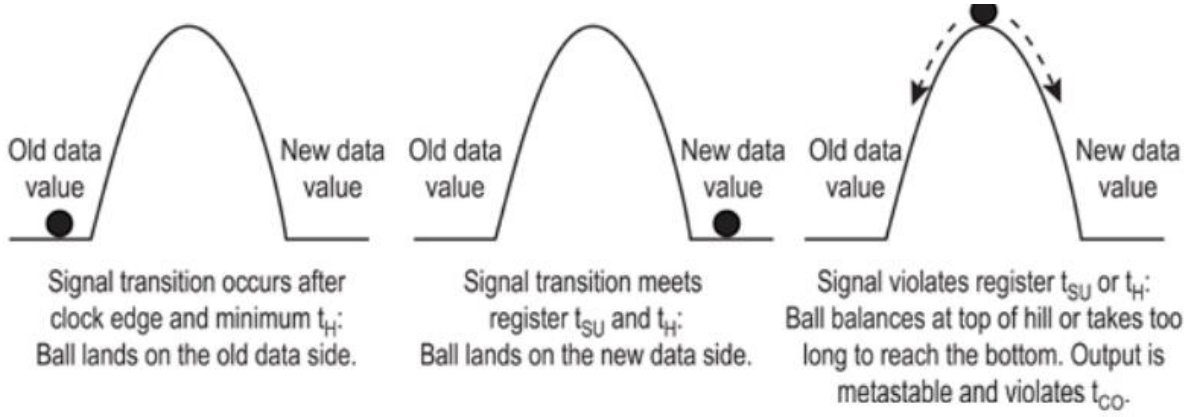


2. Conservative constraint based on delays of data and clock to internal nodes to ensure sufficient margins for proper operation
   - Setup: **A** must be stable (new value) before Tgate opens:
     Tsu=T(D->**A**) – T(CLK->T2)
   - Hold: **B** must be stable (old value) until T-gate closes:
     Tho=T(CLK->T1) – T(D->**B**)

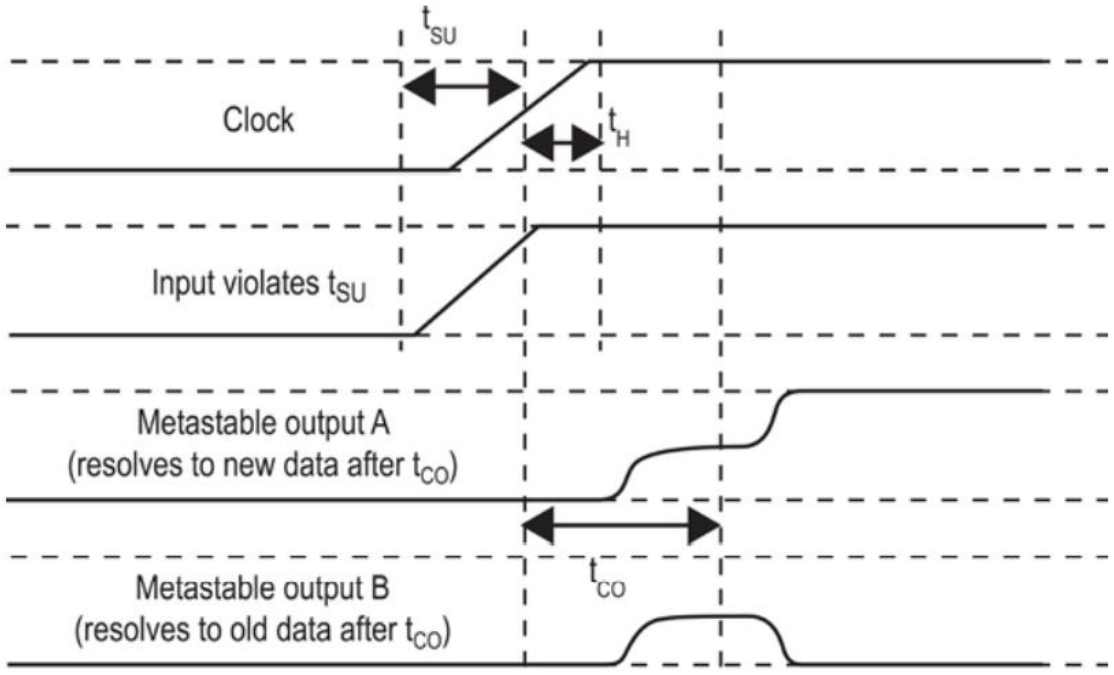


Metastability

- **Metastability:** third, instable state in bi-stable circuits

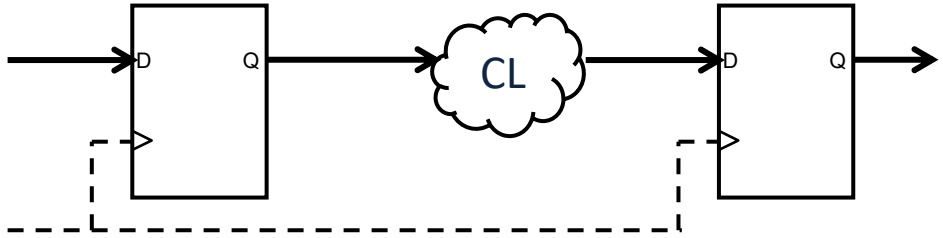


- **Metastability can happen when violating setup/hold times**
  - Metastability is a temporary phenomenon and resolves by itself after some time
  - Time to resolve must be considered a random variable (uncertain)

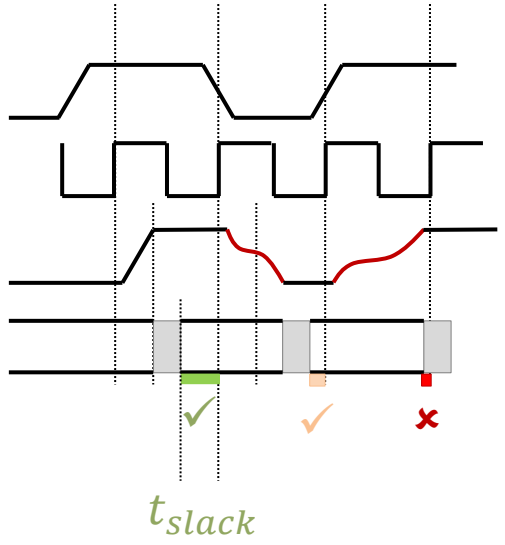


Metastability Failure Mechanism

- **Time available to resolve a metastability depends on the available margin S(slack) with no metastability** t_{slack}



- Any metastability reduces this timing slack



- If **no slack remains** (zero or negative slack), **timing errors occur** in the next flip-flop
  - Circuit no longer operates correctly (FAILURE)

Mean-Time-Between-Failure (MTBF)

- **Since metastability duration varies, not every metastability leads to an error**
- **Mean-Time-Between-Failures (MTBF) provides an estimate of the reliability**
- **The MTBF depends on**
  - Distribution of the metastability duration
  - The available timing slack t_{slack}
  - How often a potentially metastable data input signal toggles f_{DATA}
  - How often the data input signal is samples (i.e., the sampling clock) f_{CLK}
  - Some empirically determined constants that depend on process, operating conditions, circuit, … : T_0, τ_s

Failures per second

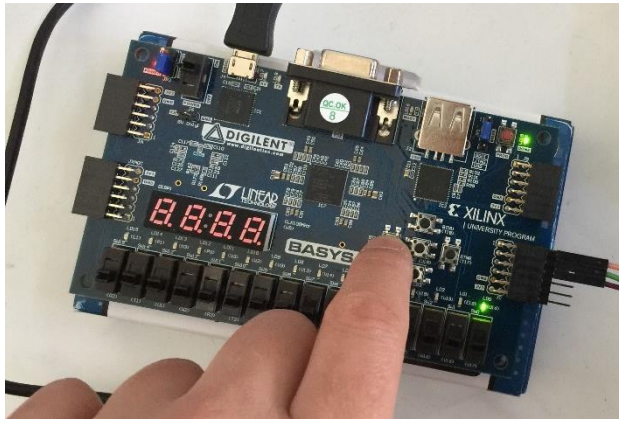
$$P_{failure} = T_0 \cdot f_{DATA} \cdot f_{CLK} \cdot e^{-t_{slack}/\tau_s}$$

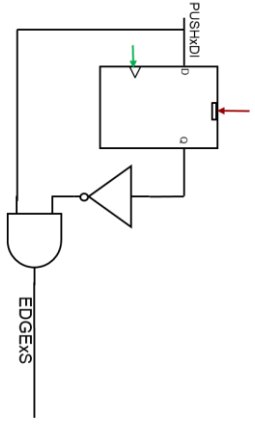
$$MTBF = \frac{e^{t_{slack}/\tau_s}}{T_0 \cdot f_{DATA} \cdot f_{CLK}}$$

Reasons for Metastability Failures

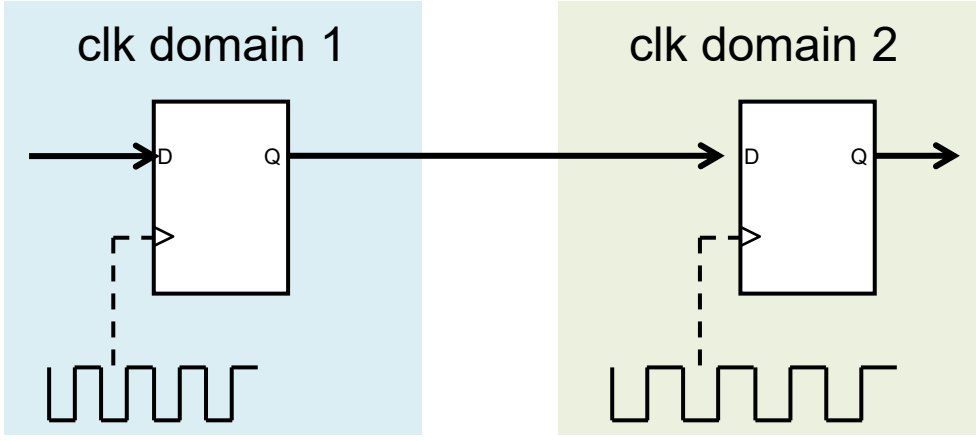
- **A well-built synchronous circuit with the right clock should never face metastability issues** (ensured by static timing analysis)
- **Unfortunately, still sometimes inputs are not synchronized to their clock**
- Examples:

Example 1:
asynchronous primary inputs
(buttons or low-frequency inputs from other chips)





Example 2:
signals between asynchronous clock domains



Measures to Reduce Impact of Metastability

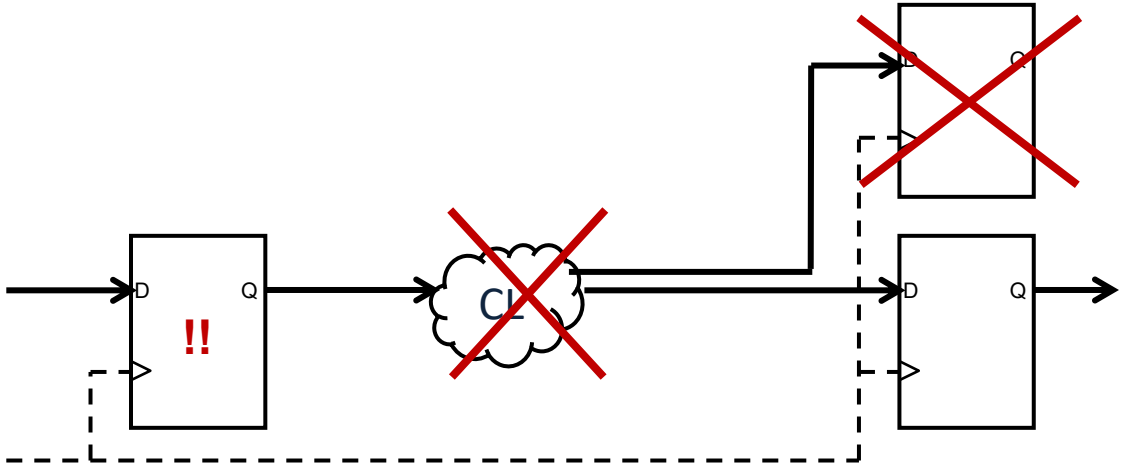
- **If metastability can not be avoided we need to reduce its impact**

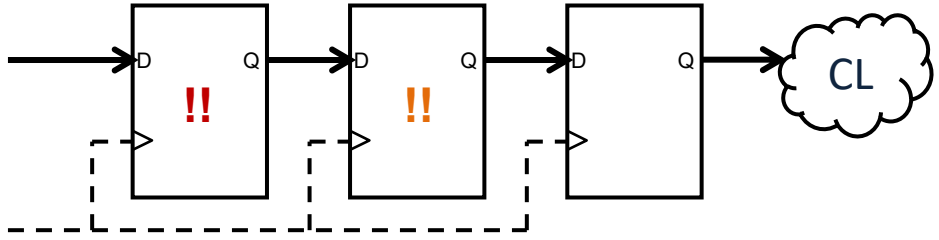
- **Three** main **principles** **to reduce impact of metastability** (reduce risk of failure)

Avoid any logic in potentially metastable signals

NO FANOUT: only impact one subsequent register

MAXIMIZE time to resolve metastability before using the signal





EE-429
Fundamentals of VLSI Design

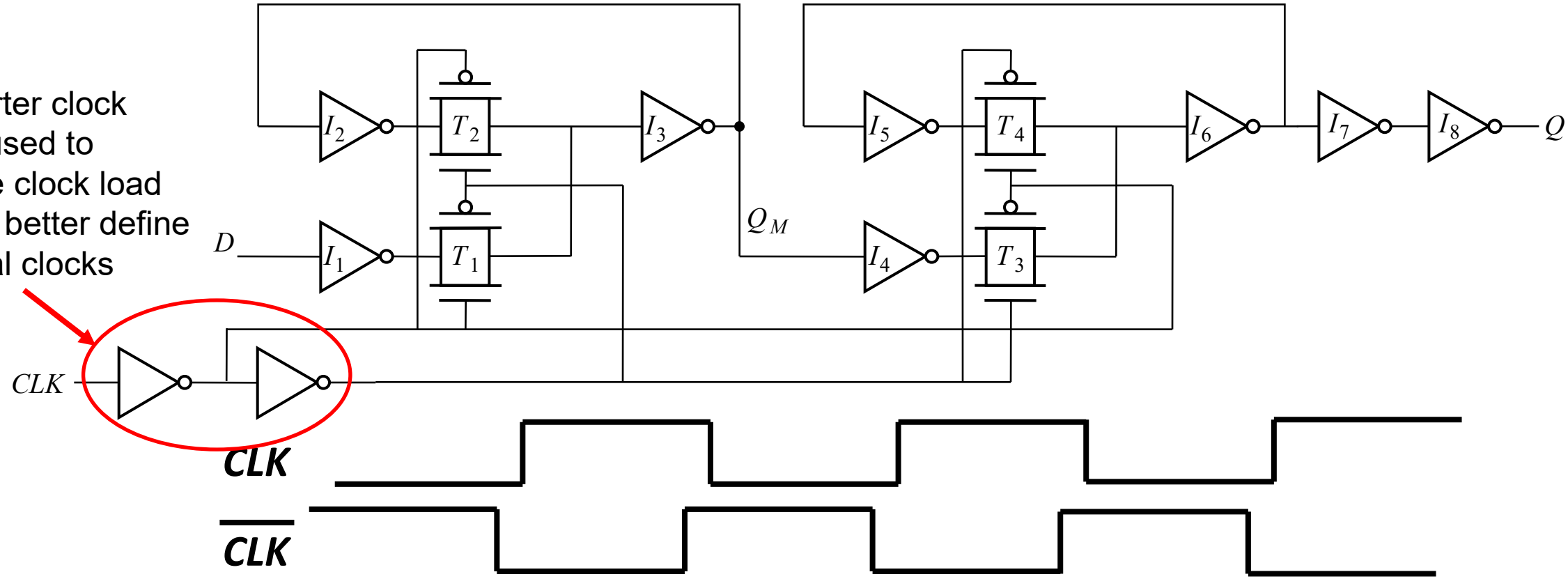
More Sequential Elements

Andreas Burg

The Clock Overlap Problem

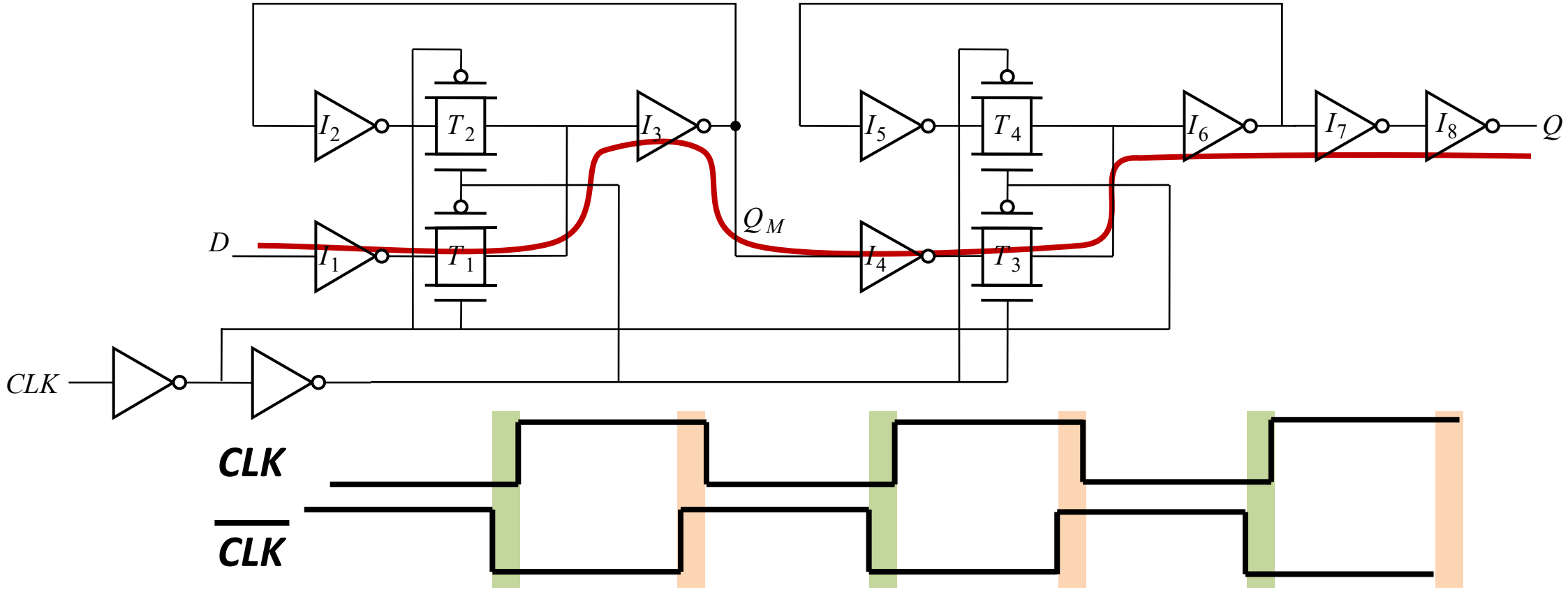
- **Master/Slave FlipFlops use both CLK and $\overline{\mathbf{CLK}}$, which is generate from CLK**
  - Generating $\overline{\mathbf{CLK}}$ from **CLK** usually involves a delay
  - Locally generating $\overline{\mathbf{CLK}}$ may cause this delay when 2 inverters are used

2 inverter clock input used to reduce clock load and to better define internal clocks



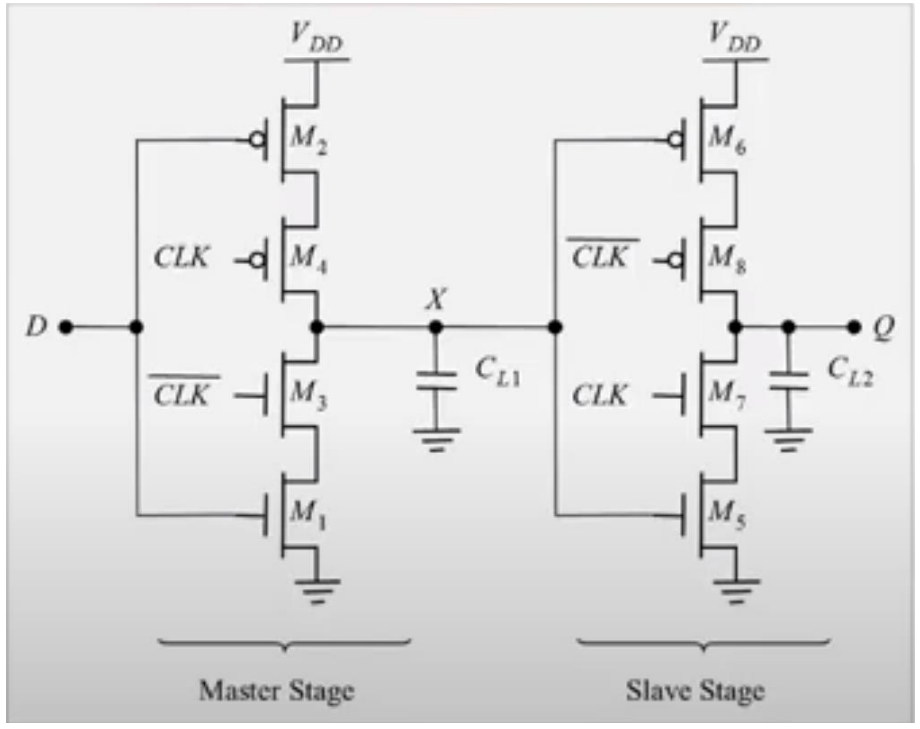
The Clock Overlap Problem

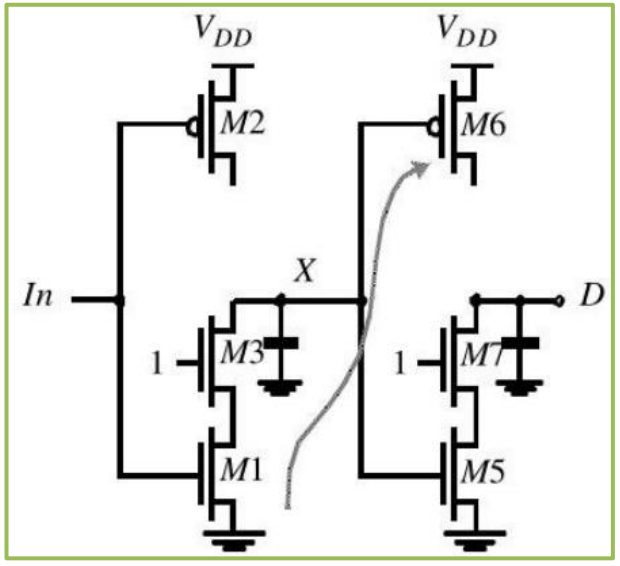
- **Delay of $\overline{\text{CLK}}$ leads to clock overlap on both edges**
- **During the overlaps both Master and Salve latches are briefly open**
  - Signal can pass directly from input to output **causing a race-condition**

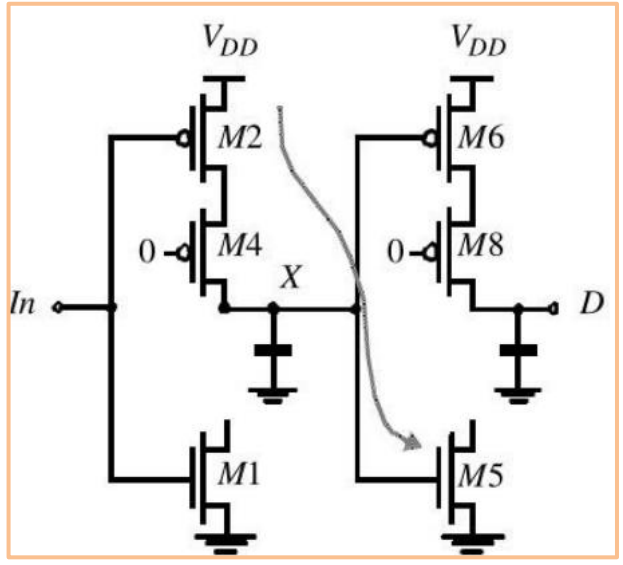


Solution 1: Clocked CMOS (C²MOS)

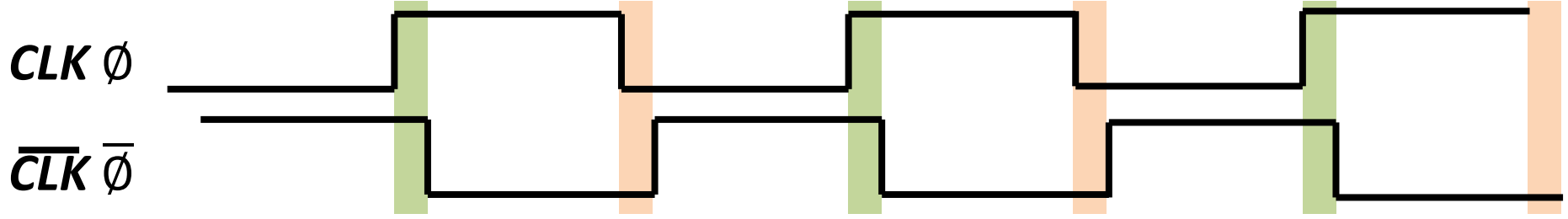
- **Master/Slave-FlipFlop from dynamic latches** avoids sensitivity to clock overlap





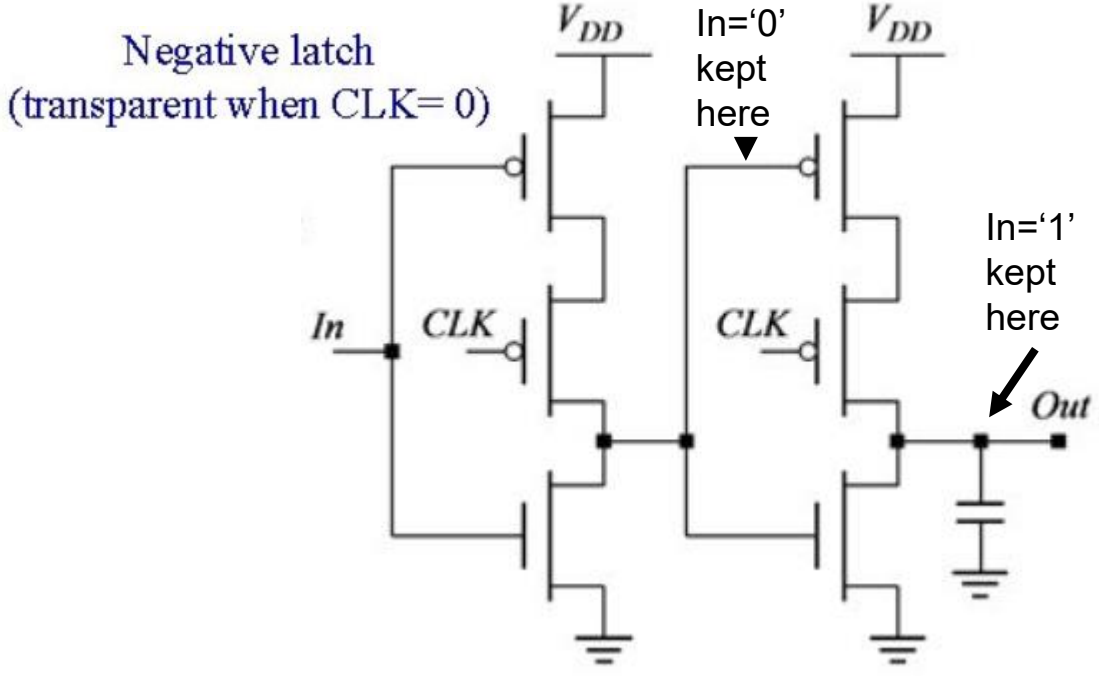


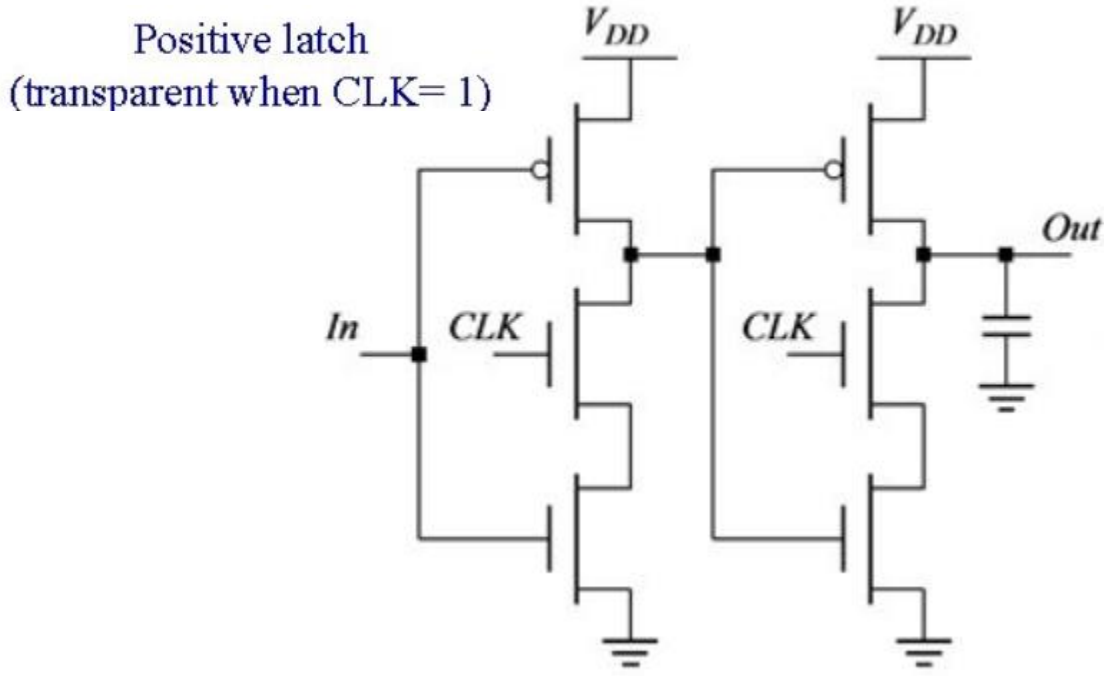
Used a lot by IBM



Solution 2: True Single Phase Clocked (TSPC) Latch

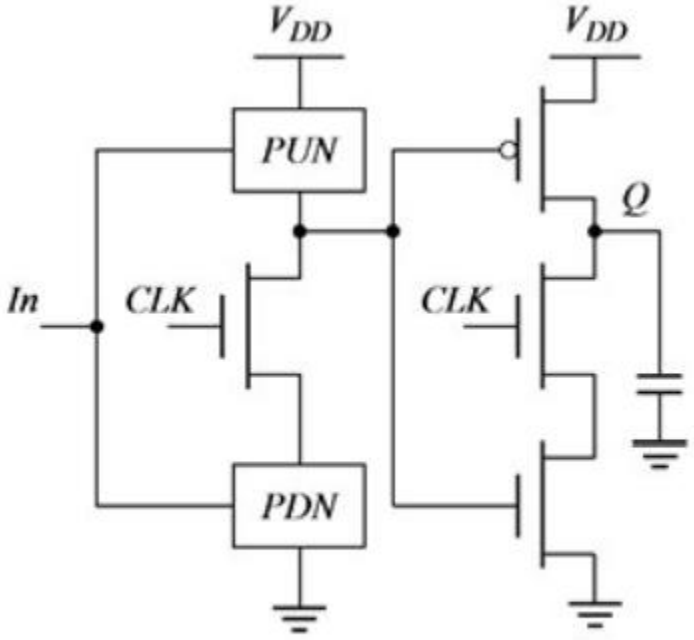
- **C^2MOS still requires two clocks a positive and an inverted one (overhead)**
- **TSPC: different Master/Slave structures for different phases with same clock**
  - Two subsequent inverters that keep either a ZERO (high-transparent) or a ONE (low-transparent)
  - Input levels of '0' and '1' are kept on two different nodes



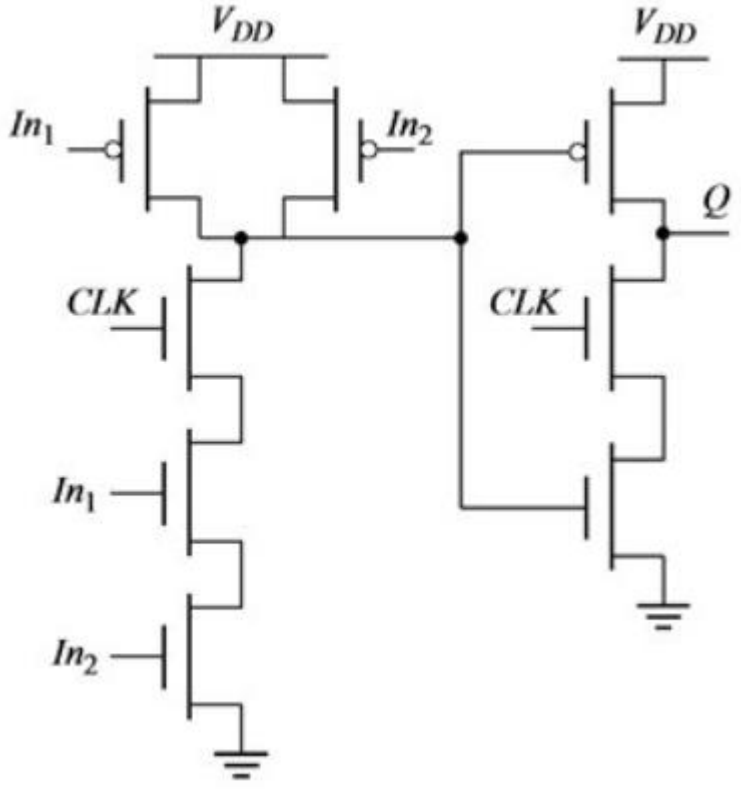


Solution 2: TSPC Latch with Integrated Logic

- **TSPC latches allow to replace the initial inverter with any PMOS/NMOS network to realize an integrated logic function**



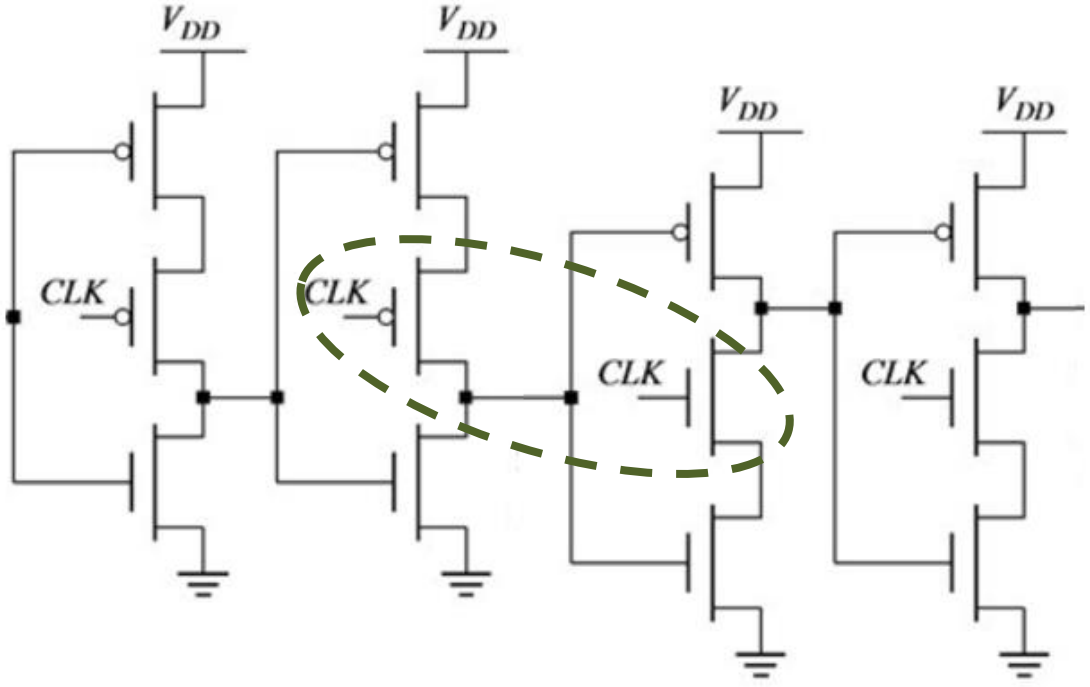
Arbitrary Logic in an TSPC Latch (‘1’ transparent)

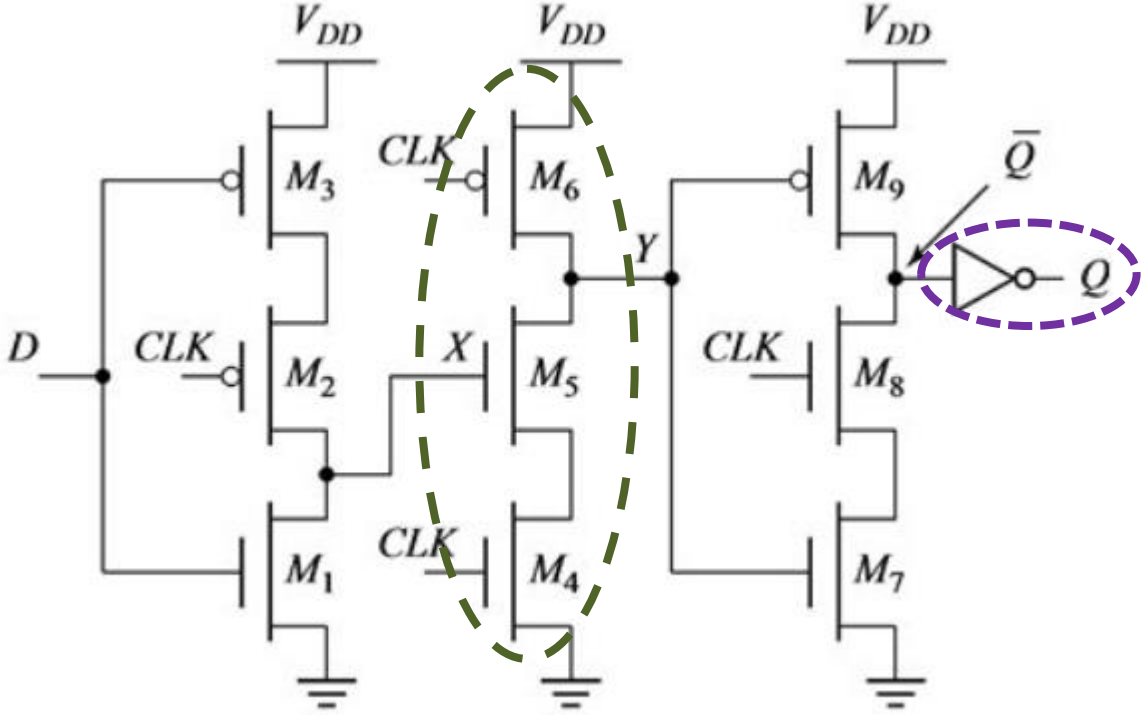


Example: 2-input AND

Solution 2: TSPC FlipFlops

- **TSPC FlipFlops are built by two TSPC latches in Master/Slave configuration**
- **Two center-stages can be combined to remove one stage**
  - Results in an inverted output



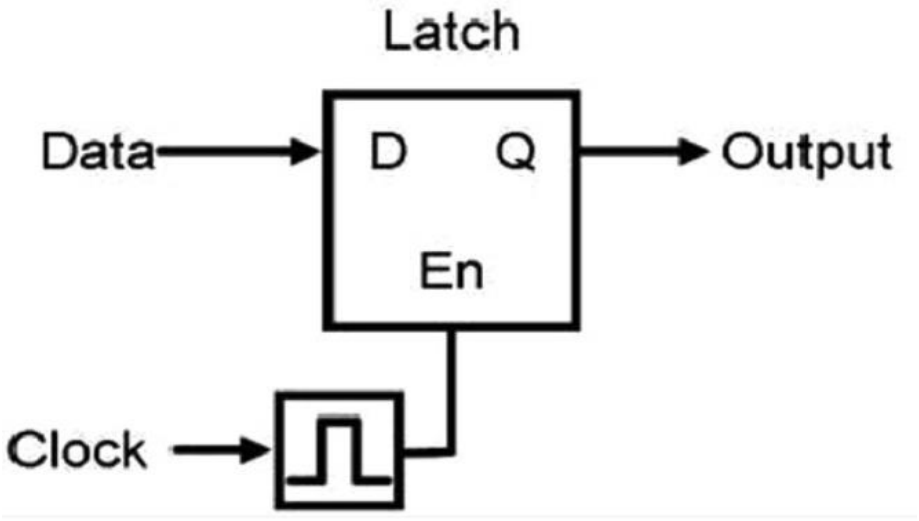


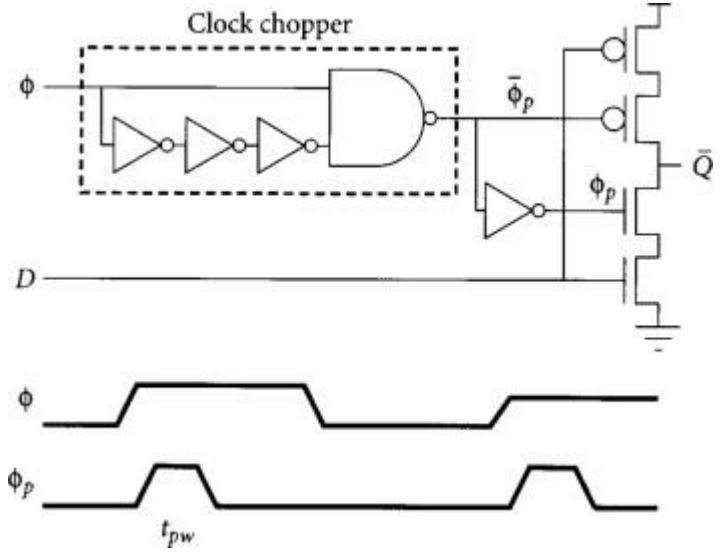
Compact TSPC FF with shielded output

Solution 3: Pulsed (dynamic) Latch as FlipFlop

- **Build a FlipFlop from a latch that is triggered by a short pulse**
  - Pulse generator (dynamic EDGE detector) built from delay elements
- **Many different forms: static, dynamic, single and double-edge triggered**

- **Main issue: pulse duration is tradeoff between robustness and hold time**

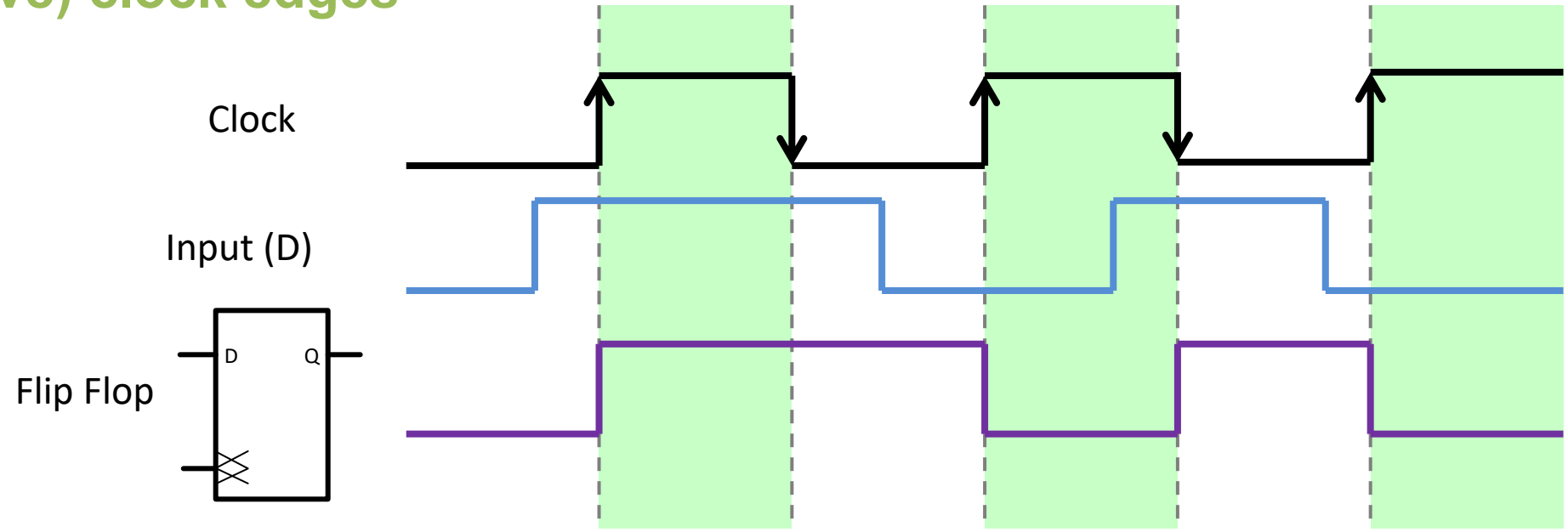




Dual/Double Edge Triggered FlipFlops

- **In single-edge** (positive or negative) **edge-triggered logic, one** (the inactive) **transition of the clock** signal **has no functional role**
  - Every **"operation" cycle requires two clock transitions** => inefficient in terms of power

- **Solution:** **dual-edge triggered FlipFlops store data on both (positive and negative) clock edges**



Dual/Double Edge Triggered FlipFlops (cont.)

- Different realizations of dual-edge triggered FlipFlops are possible

- Most straightforward and most practical **implementation is based on two latches and a MUX**
  - Setup and hold times defined by a latch
  - No clock-overlap problem

  - Clock load is higher than for a simple single-edge triggered FlipFlop (power!!)
  - Need to avoid sneak-paths through the MUX between latches during switching