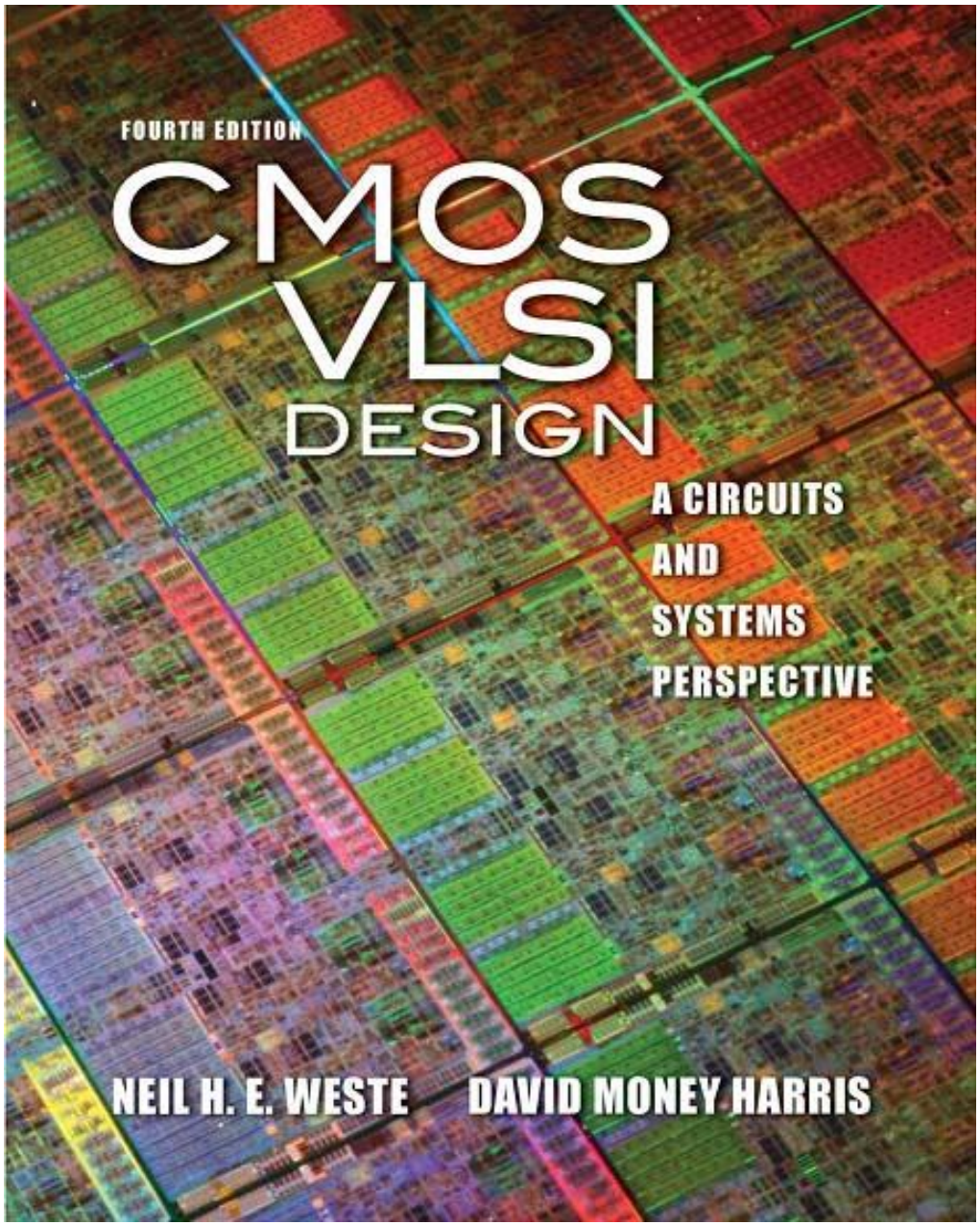


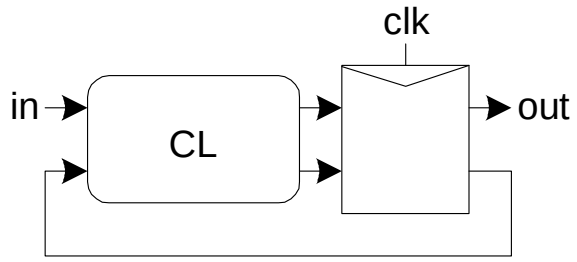
Sequential Circuit Design

Outline

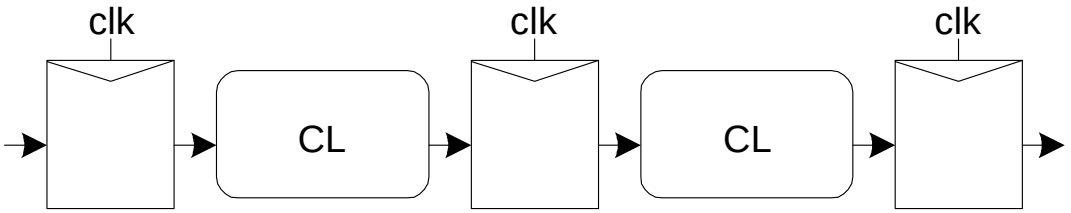
- Sequencing
- Sequencing Element Design
- Max and Min-Delay
- Clock Skew
- Time Borrowing
- Two-Phase Clocking

Sequencing

- *Combinational logic*
  - Output depends on current inputs
- *Sequential logic*
  - Output depends on current and previous inputs
  - Requires separating previous, current, future
  - Called *state* or *tokens*
  - Ex: FSM, pipeline



Finite State Machine



Pipeline

Sequencing (Cont.)

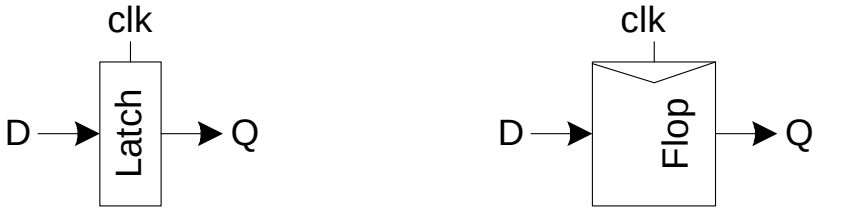
- If tokens moved through pipeline at constant speed, no sequencing elements would be necessary
- Ex: fiber-optic cable
  - Light pulses (tokens) are sent down cable
  - Next pulse sent before first reaches end of cable
  - No need for hardware to separate pulses
  - But *dispersion* sets min time between pulses
- This is called *wave pipelining* in circuits
- It doesn't work in most electrical circuits:
  - Various paths have varying delays

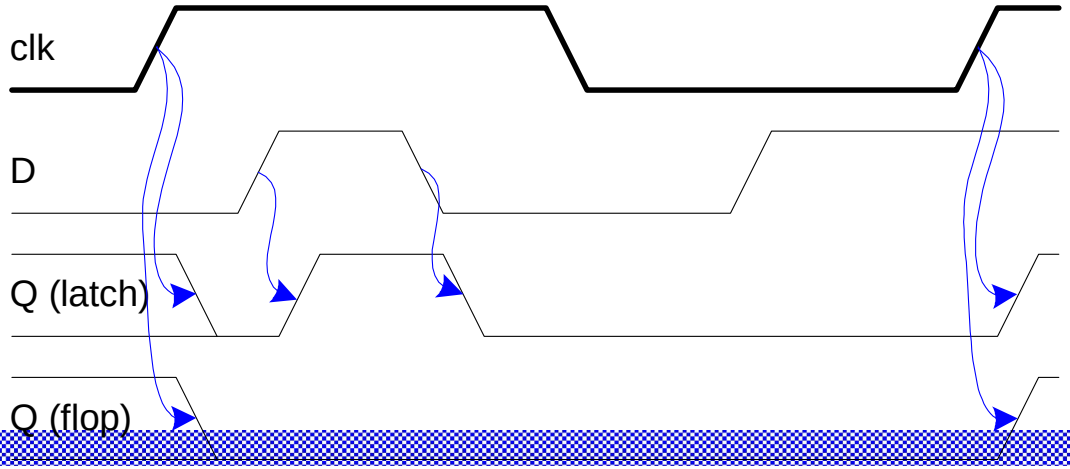
Sequencing Overhead

- Use flip-flops to delay fast tokens so they move through exactly one stage in each cycle
- Inevitably adds some delay to the critical tokens
- Makes circuit slower than just the logic delay
  - Called sequencing overhead
- Some people call this clocking overhead
  - But it applies to asynchronous circuits too
  - Inevitable side effect of maintaining sequence

Sequencing Elements

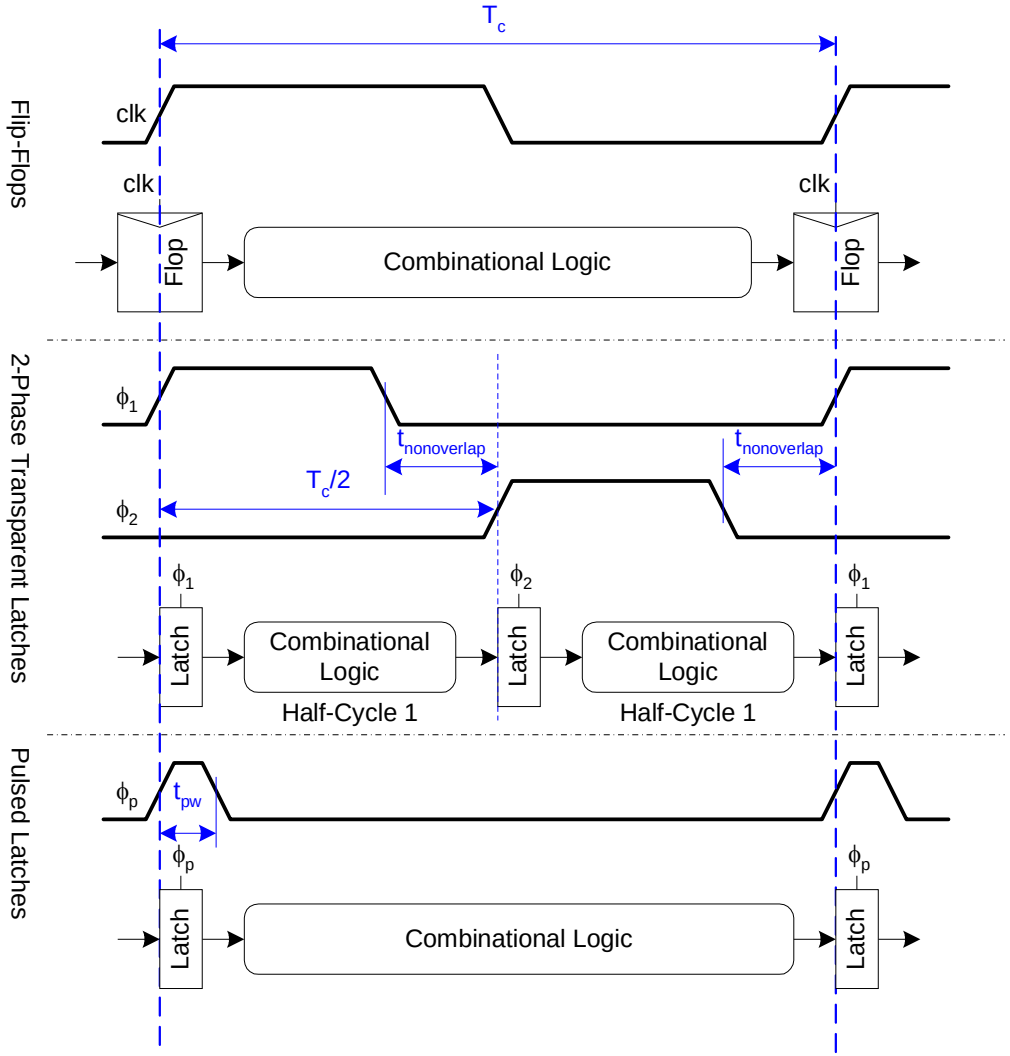
- **Latch**: Level sensitive
  - a.k.a. transparent latch, D latch
- **Flip-flop**: edge triggered
  - a.k.a. master-slave flip-flop, D flip-flop, D register





Sequencing Methods

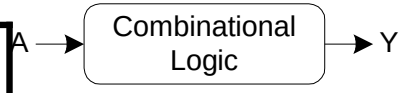
- Flip-flops
- 2-Phase Latches
- Pulsed Latches

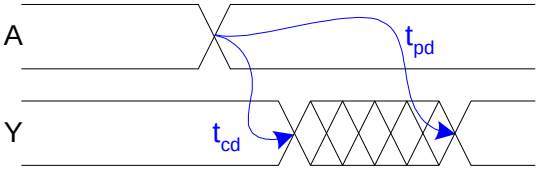


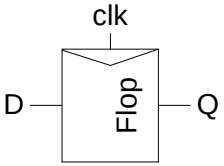
Timing Diagrams

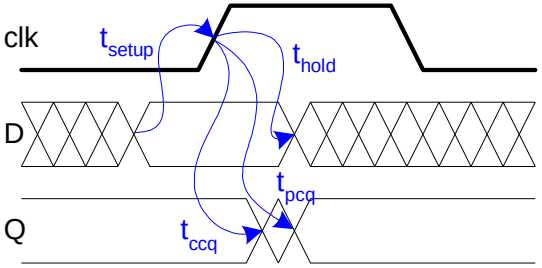
Contamination and Propagation Delays

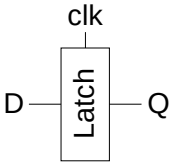
t_{pd}	Logic Prop. Delay
t_{cd}	Logic Cont. Delay
t_{pcq}	Latch/Flop Clk->Q Prop. Delay
t_{ccq}	Latch/Flop Clk->Q Cont. Delay
t_{pdq}	Latch D->Q Prop. Delay
t_{cdq}	Latch D->Q Cont. Delay
t_{setup}	Latch/Flop Setup Time
t_{hold}	Latch/Flop Hold Time

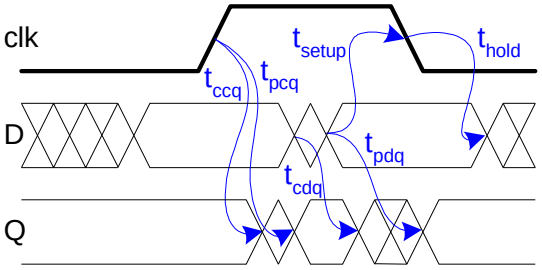






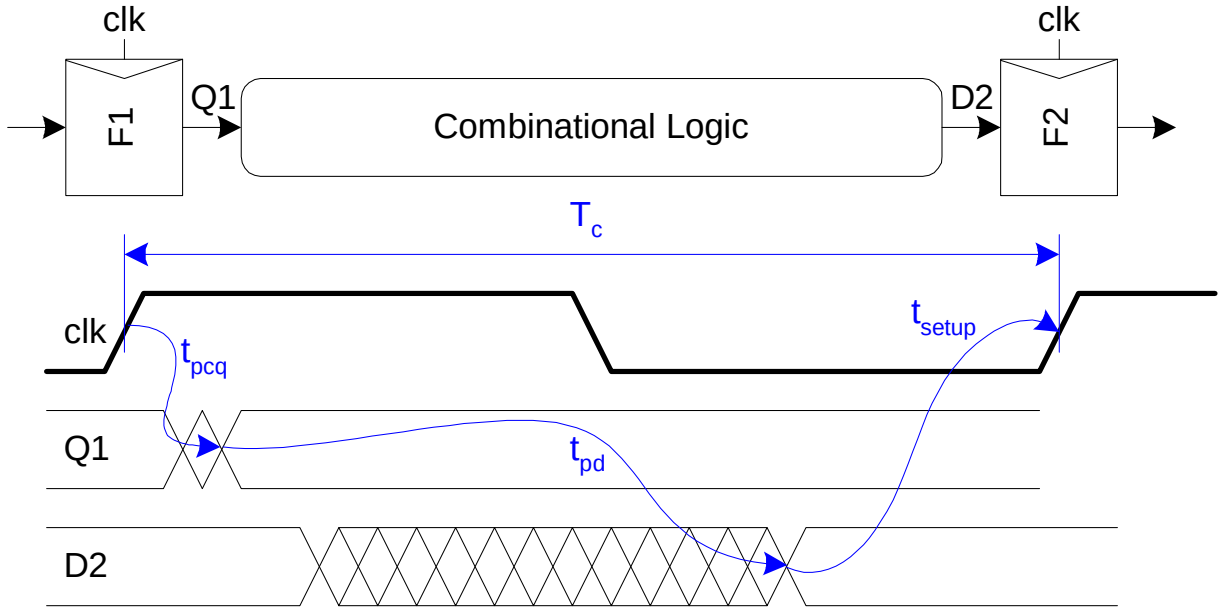






Max-Delay Violation: Flip-Flops

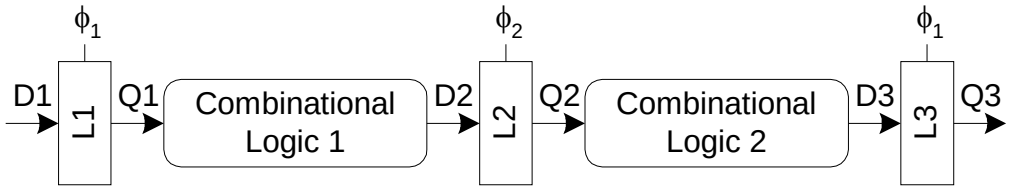
$$t_{pd} \leq T_c - \underbrace{\left(t_{\text{setup}} + t_{pcq}\right)}_{\text{sequencing overhead}}$$



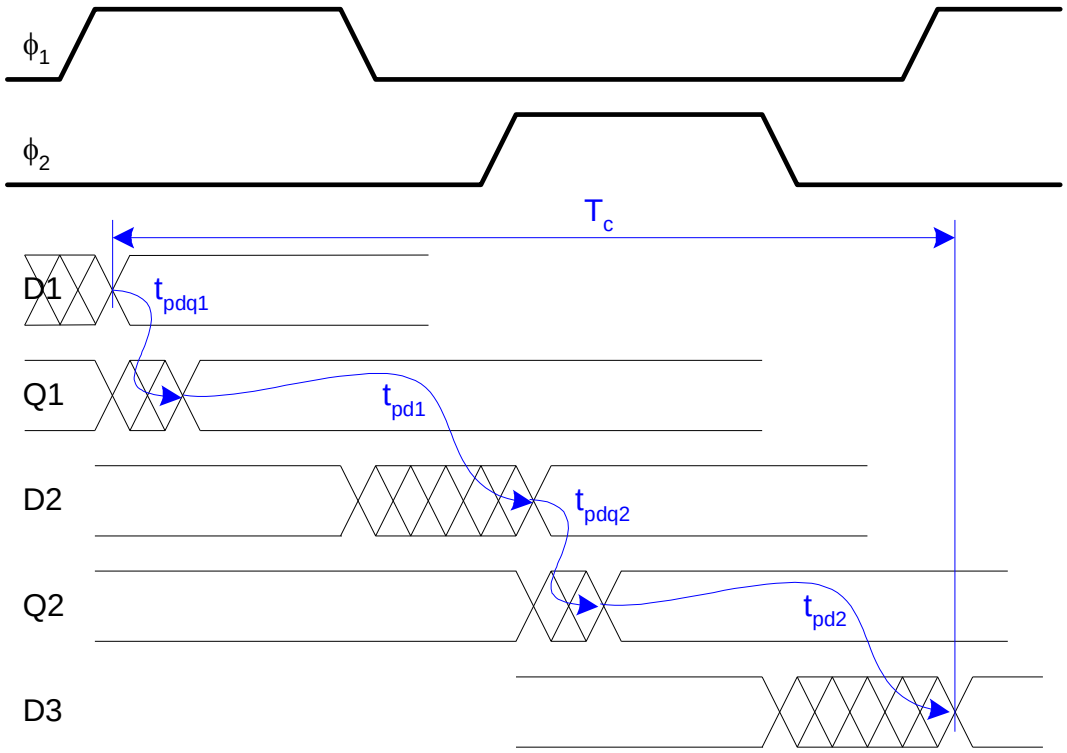
- Ideal case: the sequencing overhead is zero
- Reality: the overhead is not zero and reduces either
  - the operating frequency ($1/T_c$)
  - or the maximum tolerable combinational delay (T_{pd})

Max-Delay Violation: 2-Phase Latches

$$t_{pd} = t_{pd1} + t_{pd2} \leq T_c - \underbrace{(2t_{pdq})}_{\text{sequencing overhead}}$$



- Again, in reality the overhead is not zero and reduces either $1/T_c$ or T_{pd}

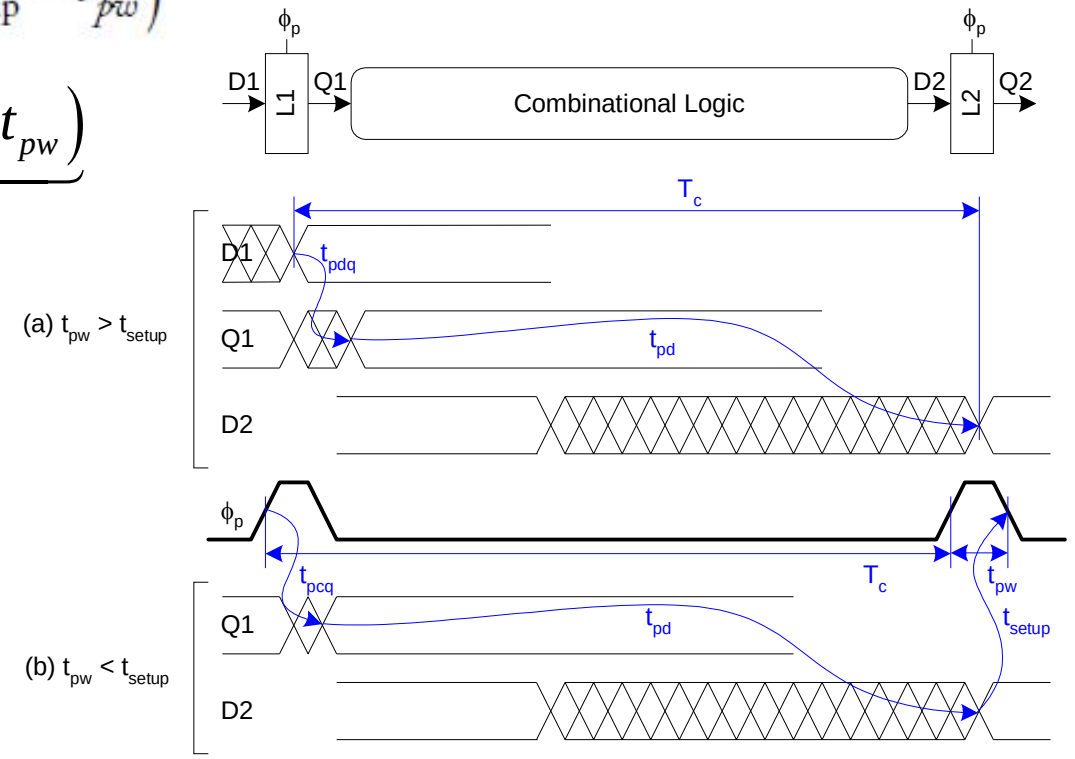


Max-Delay Violation: Pulsed Latches

$$T_c \geq \max\left(t_{pdq} + t_{pd}, t_{pcq} + t_{pd} + t_{\text{setup}} - t_{pw} \right)$$

$$t_{pd} \leq T_c - \underbrace{\max\left(t_{pdq}, t_{pcq} + t_{\text{setup}} - t_{pw}\right)}_{\text{sequencing overhead}}$$

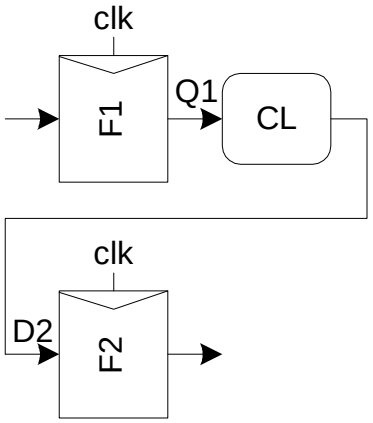
- Again, in reality the overhead is not zero and reduces either $1/T_c$ or T_{pd}

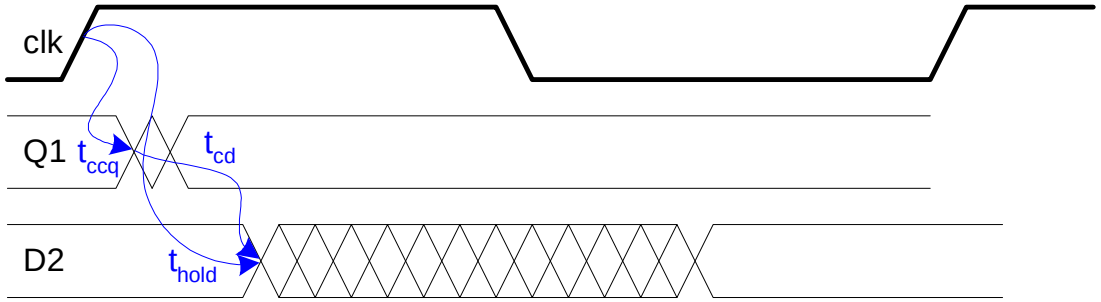


Min-Delay Violation: Flip-Flops

$$t_{cd} \geq t_{\text{hold}} - t_{ccq}$$

- Can back-to-back flip-flops be used (like in scan chains)?
- If the contamination delays through the flip-flops exceed the hold time, yes.
- If not, add delay between the flip-flops:
  - using buffers
  - using slow flip-flops with large contamination delay

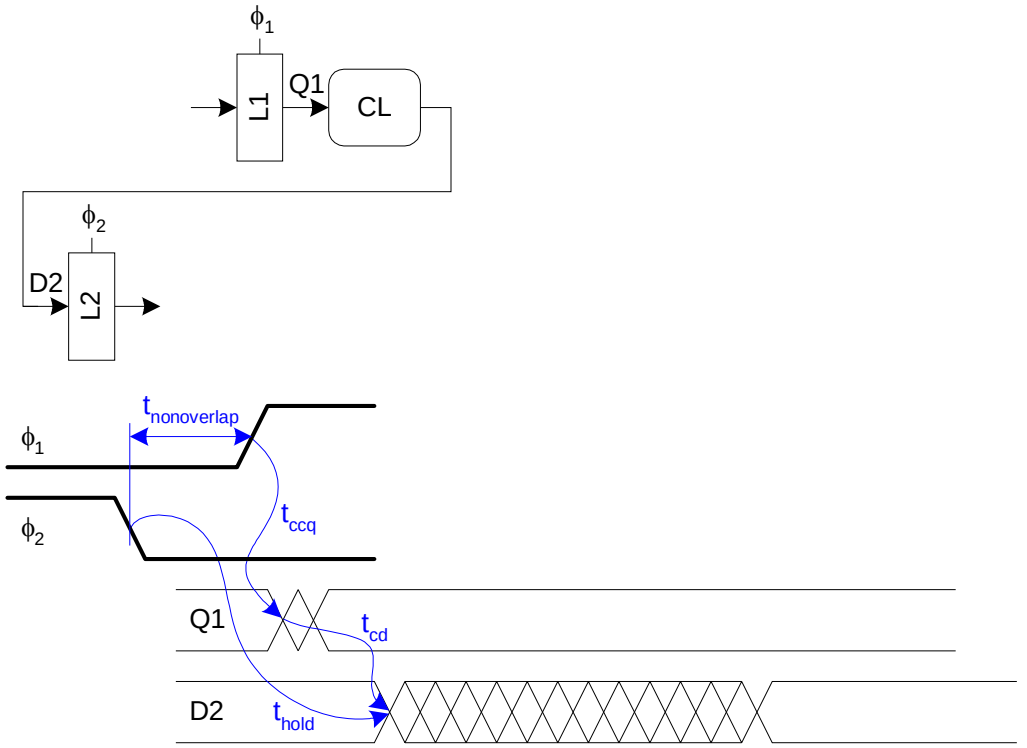




Min-Delay Violation: 2-Phase Latches

$$t_{cd1}, t_{cd2} \geq t_{\text{hold}} - t_{ccq} - t_{\text{nonoverlap}}$$

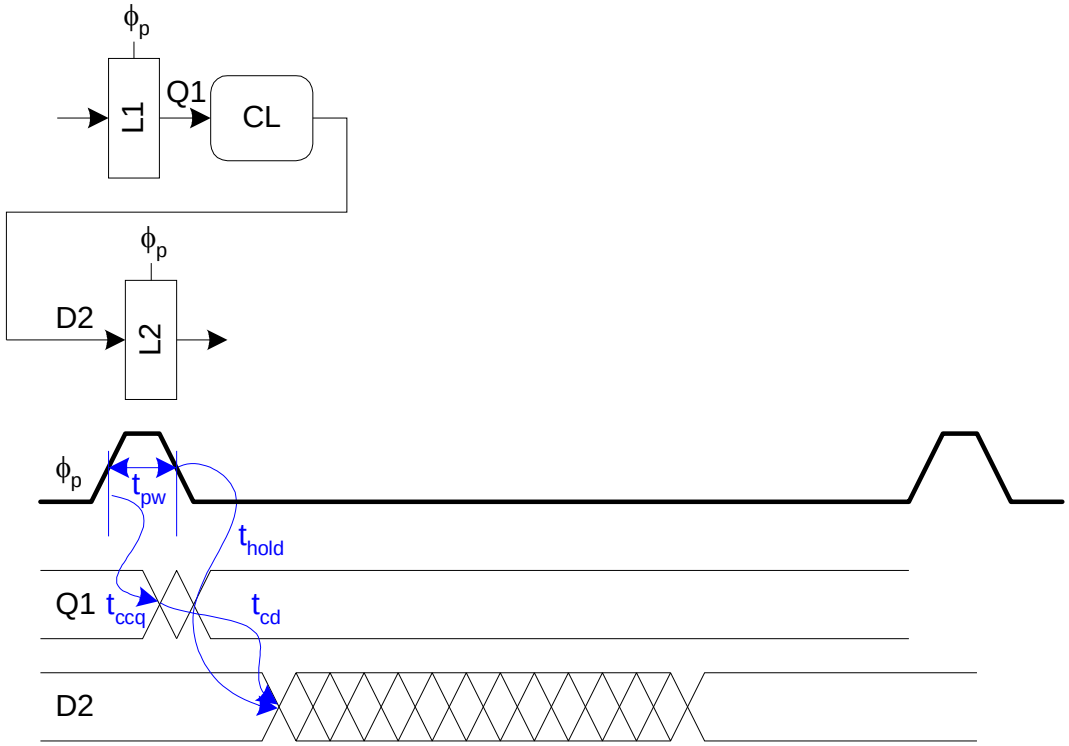
- Hold time is reduced by nonoverlap time.
- Hold applies twice each cycle, vs. only once for flipflops.
- But a flipflop is made of two latches!



Min-Delay Violation: Pulsed Latches

$$t_{cd} \geq t_{\text{hold}} - t_{ccq} + t_{pw}$$

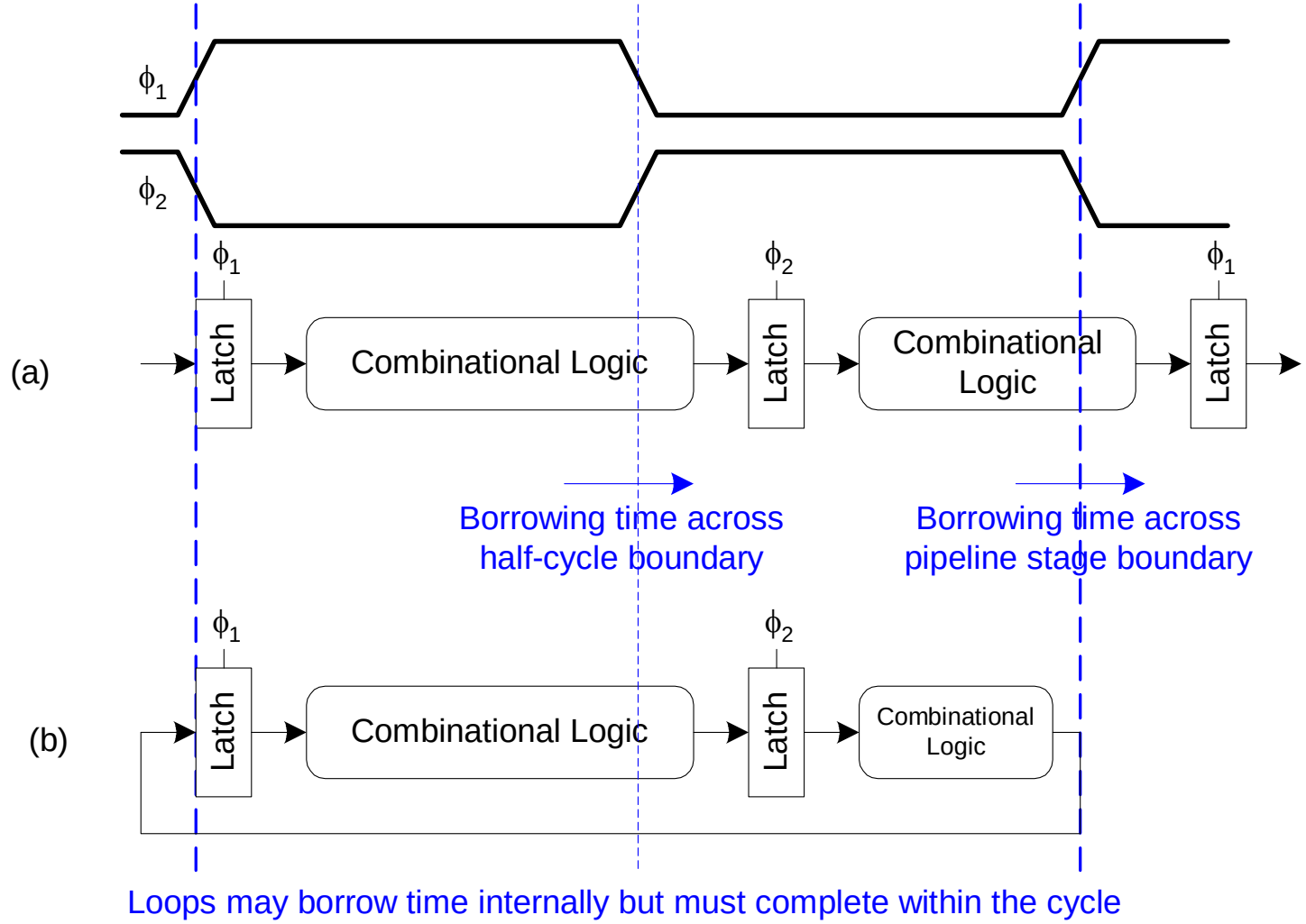
Hold time increased by pulse width



Time Borrowing

- In a flipflop-based system:
  - Data launches on one rising edge
  - Must setup before next rising edge
  - If it arrives late, system fails
  - If it arrives early, time is wasted
  - Flipflops have hard edges
- In a latch-based system
  - Data can pass through latch while transparent
  - Long cycle of logic can borrow time into next
  - As long as each loop completes in one cycle

Time Borrowing Example



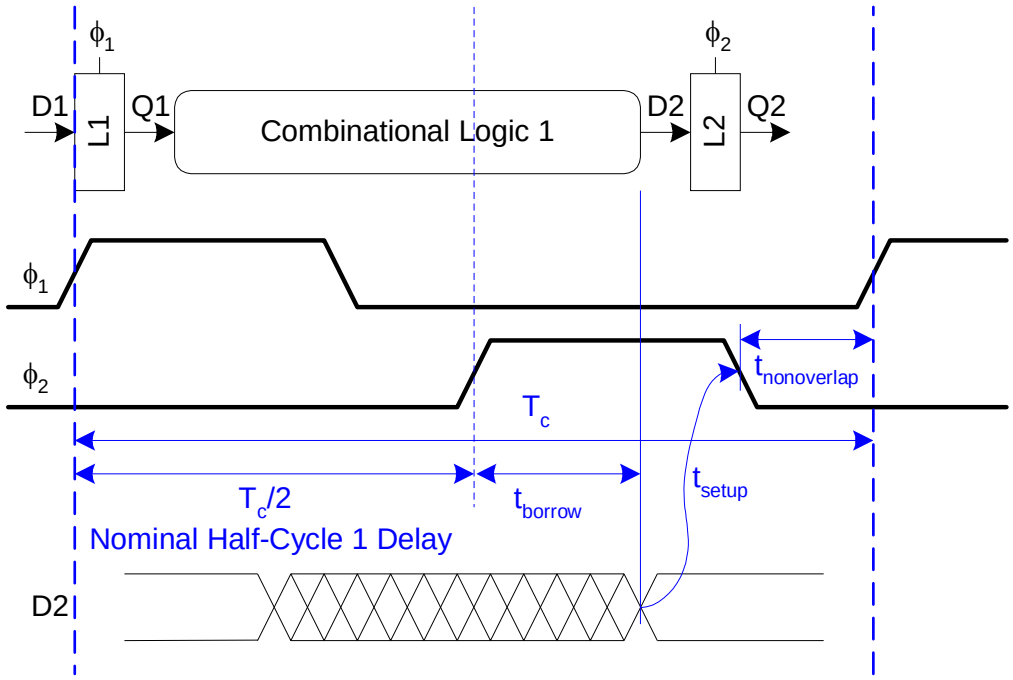
How Much Borrowing?

2-Phase Latches

$$t_{\text{borrow}} \leq \frac{T_c}{2} - \left(t_{\text{setup}} + t_{\text{nonoverlap}} \right)$$

Pulsed Latches

$$t_{\text{borrow}} \leq t_{pw} - t_{\text{setup}}$$



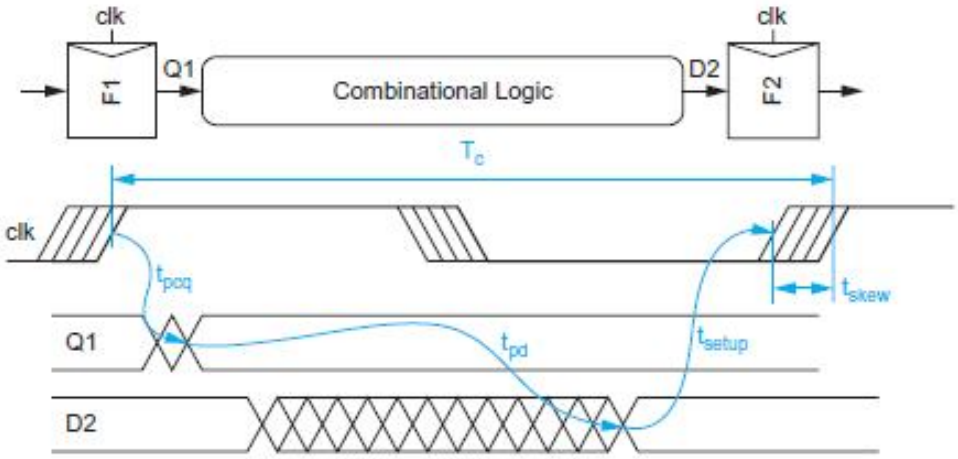
Clock Skew

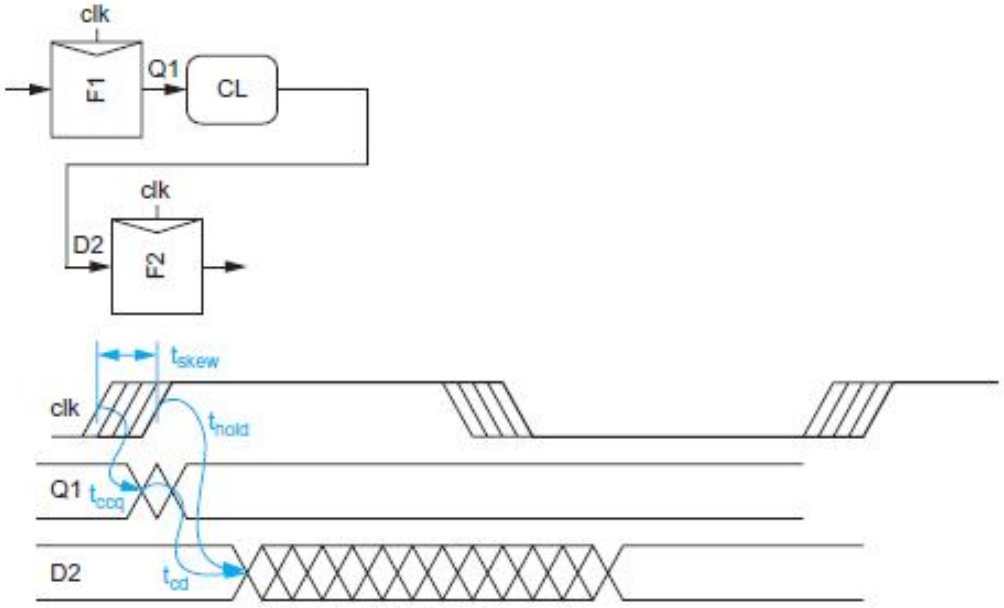
- We have assumed zero clock skew
- Clocks really have uncertainty in arrival time
  - Decreases maximum propagation delay
  - Increases minimum contamination delay
  - Decreases time borrowing

Skew: Flip-Flops

$$t_{pd} \leq T_c - \underbrace{\left(t_{pcq} + t_{\text{setup}} + t_{\text{skew}}\right)}_{\text{sequencing overhead}}$$

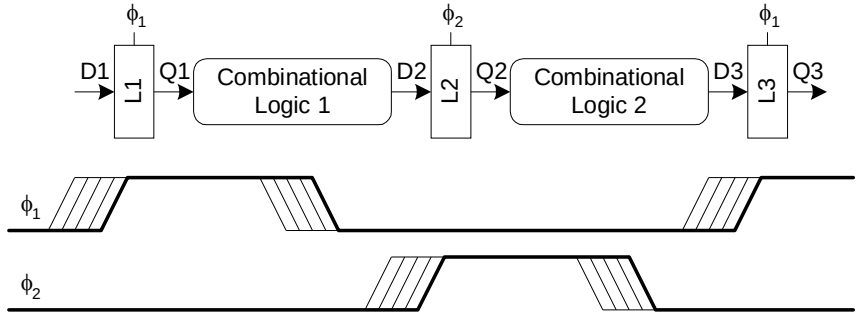
$$t_{cd} \geq t_{\text{hold}} - t_{ccq} + t_{\text{skew}}$$





Skew: 2-phase Latches

- No sensitivity to the edge for max-delay => max-delay is skew-tolerant
- Sensitive to edges for min-delay => min-delay is increased by the skew
- Borrowed time is reduced by the skew



$$t_{pd} \leq T_c - \underbrace{\left(2t_{pdq}\right)}_{\text{sequencing overhead}}$$

$$t_{cd1}, t_{cd2} \geq t_{\text{hold}} - t_{ccq} - t_{\text{nonoverlap}} + t_{\text{skew}}$$

$$t_{\text{borrow}} \leq \frac{T_c}{2} - \left(t_{\text{setup}} + t_{\text{nonoverlap}} + t_{\text{skew}}\right)$$

Skew: Pulsed Latches

- Sensitivity to skew (i.e., edges) depends on pulse width:
  - Wide pulse => similar to 2-phase case => no sensitivity to the edge => skew-tolerant for max-delays
  - Narrow pulse => similar to flip-flops => max-delay is reduced by the skew
- Regardless of the pulse width, min-delay and borrowed time are affected by the skew

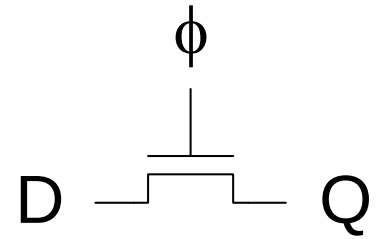
$$t_{pd} \leq T_c - \underbrace{\max\left(t_{pdq}, t_{pcq} + t_{\text{setup}} - t_{pw} + t_{\text{skew}}\right)}_{\text{sequencing overhead}}$$

$$t_{cd} \geq t_{\text{hold}} + t_{pw} - t_{ccq} + t_{\text{skew}}$$

$$t_{\text{borrow}} \leq t_{pw} - \left(t_{\text{setup}} + t_{\text{skew}}\right)$$

Latch Design (simplest)

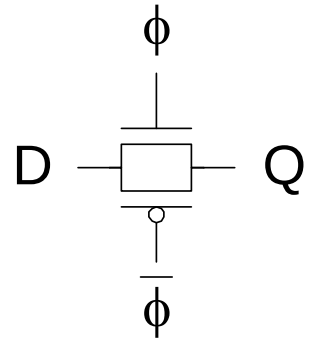
- Pass Transistor Latch
- Pros
  - + Tiny
  - + Low load on clock input
- Cons
  - – V_t drop
  - – Nonrestoring
  - – Output is dynamic (it is float when opaque which increases noise sensitivity)
  - – Diffusion input (making hard to model/analyze)
  - – Noise from output can propagate to the input



Used in 1970's

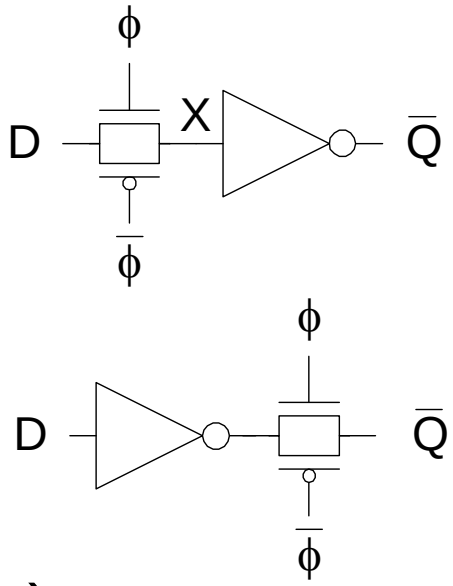
Latch Design (slightly improved)

- Transmission gate
  - \+ No V_t drop
  - \- Requires inverted clock



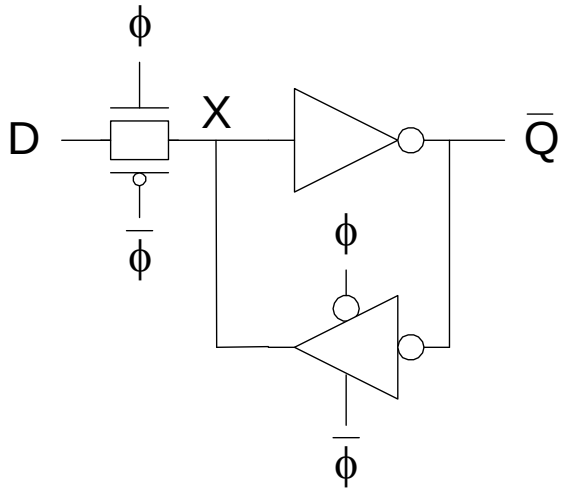
Latch Design (slightly improved)

- Inverting buffer
  - + Restoring
  - + No backdriving (Noise from output doesn't propagate to the input)
  - + Fixes either
    - Output noise sensitivity (top one)
    - Or diffusion input (bottom one)
  - – Inverted output



Latch Design (slightly improved)

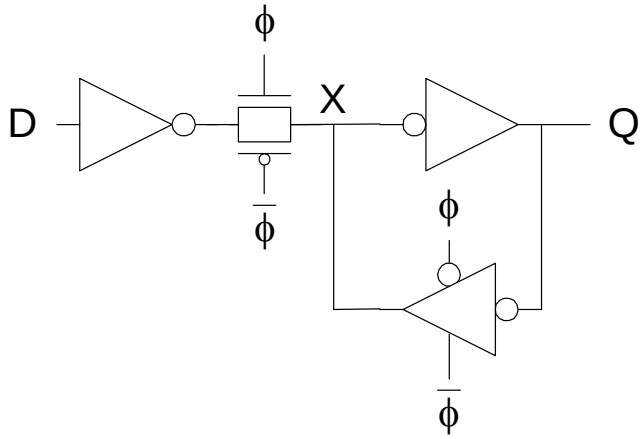
- Tristate inverter feedback
  - \+ Static
  - – Backdriving risk



- Subthreshold leakage high in modern processes => being static is a requirement

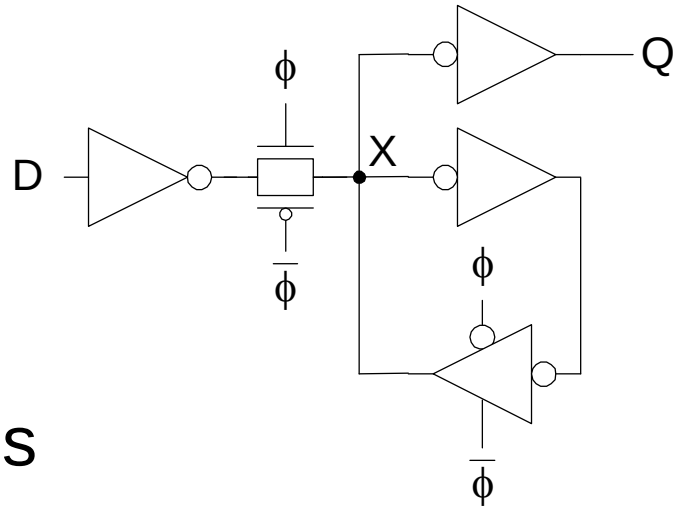
Latch Design (slightly improved)

- Buffered input
  - + Fixes diffusion input
  - + Noninverting



Latch Design (most common)

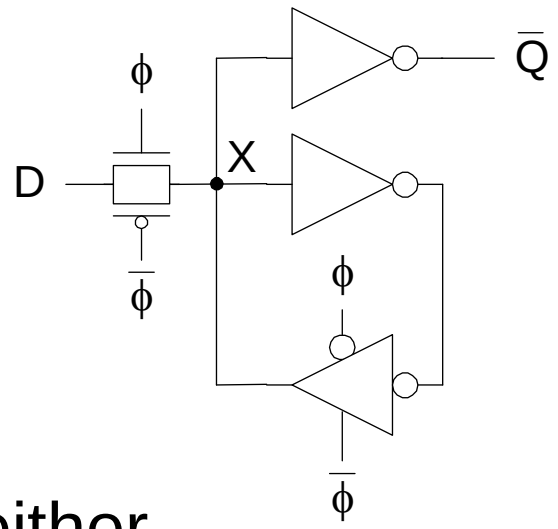
- Buffered output
  - + No backdriving



- Widely used in standard cells
  - + Very robust (most important)
  - - Rather large
  - - Rather slow (1.5 – 2 FO4 delays)
  - - High clock loading

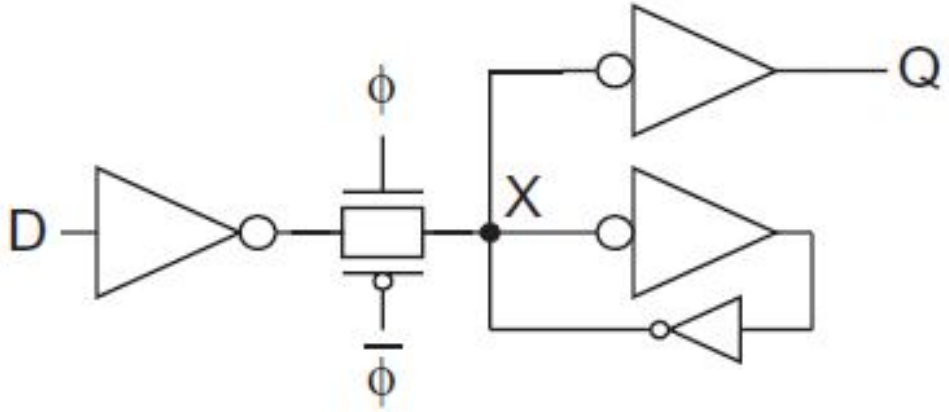
Latch Design (a step back)

- Datapath latch
  - + Smaller
  - + Faster
  - - Unbuffered input
  - - Inverting
- Used only when input noise is either
  - low
  - or predictable
- Mainly used by Intel



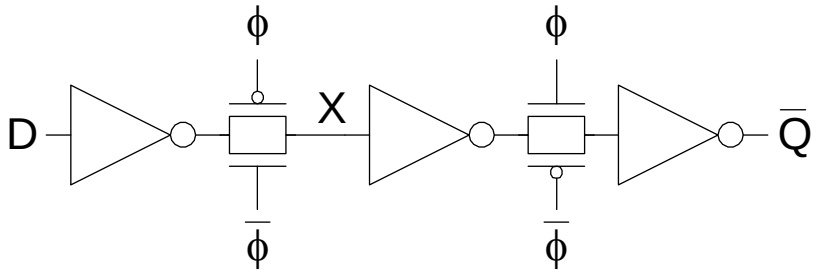
Jamb Latch Design

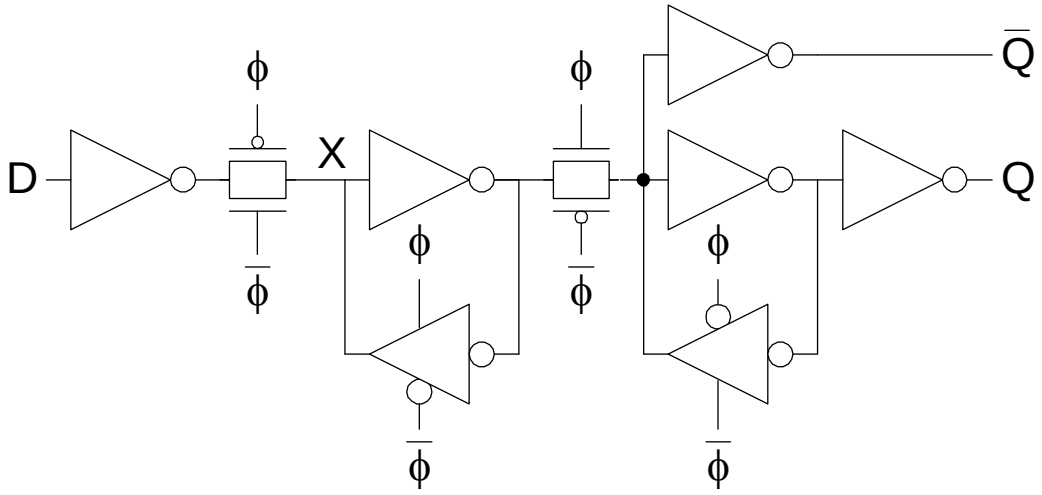
- Switch the tristate with a small inverter
  - + Saves 2 transistors
  - + Lowers clock load
  - + No backdriving
  - + No diffusion input
  - + Noninverting
  - - Careful ratio design is needed
  - - Slower than the previous one



Flip-Flop Design

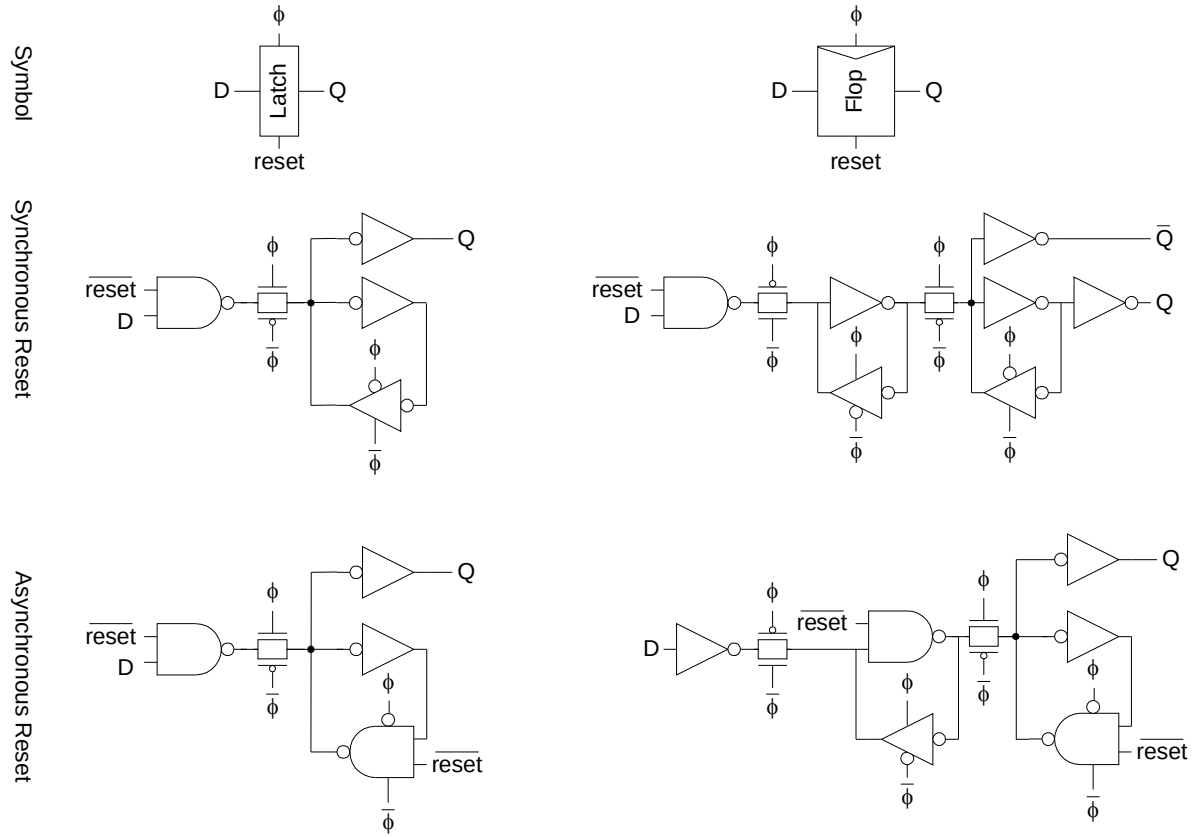
- Flip-flop is built as a pair of back-to-back latches





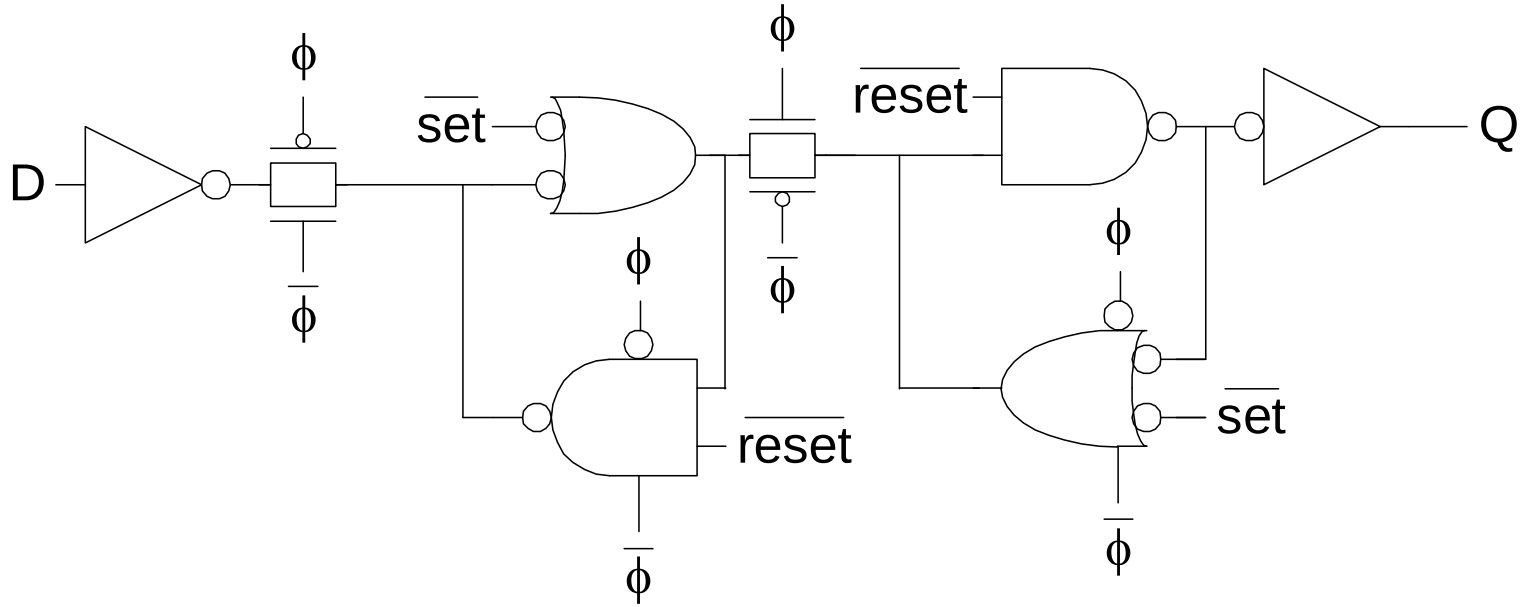
Reset

- Force output low when reset asserted
- Synchronous vs. asynchronous



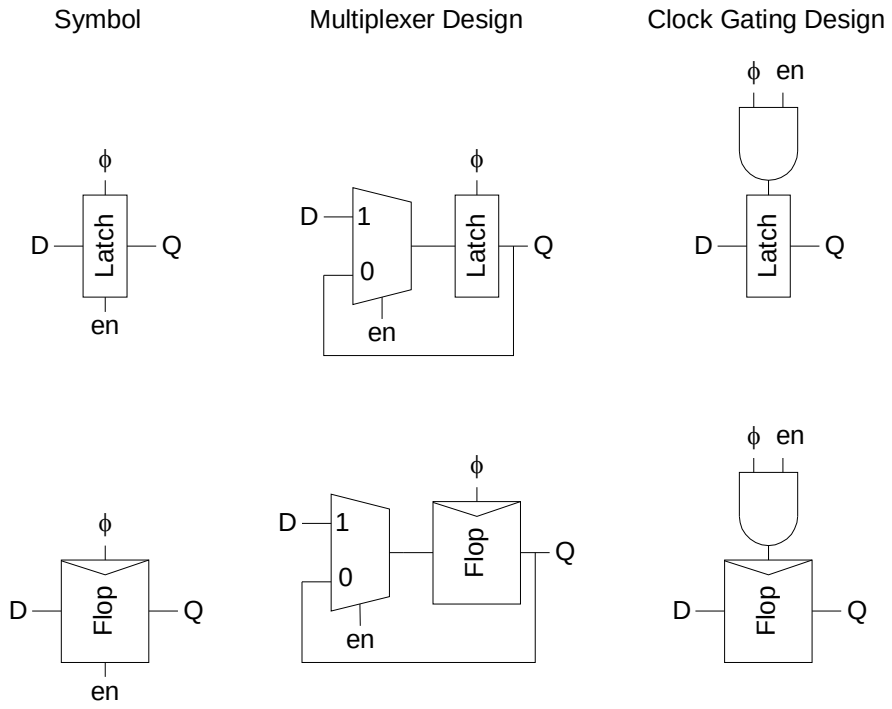
Set / Reset

- Set forces output high when enabled
- Flip-flop with asynchronous set and reset



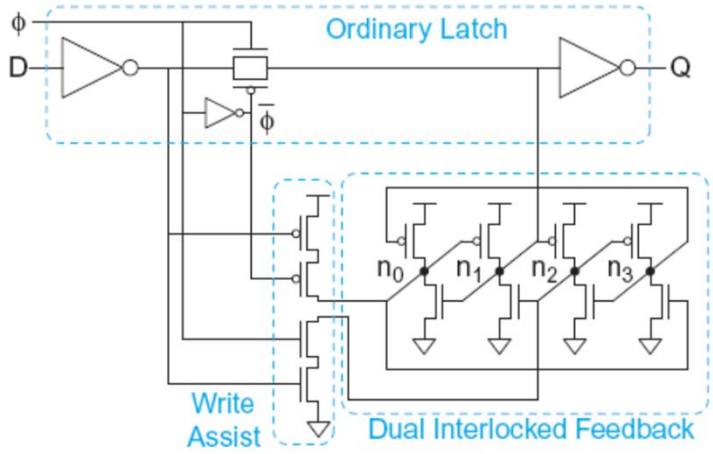
Enable

- Enable: ignore clock when en = 0
  1. Mux: increases latch D-Q delay, area
  2. Clock Gating:
     - \+ Doesn't add delay
     - \+ Small area overhead
     - \+ May reduce power
     - − Increases clock skew
     - − May cause clock glitch



Radiation Hardening

- Radiation hardening reduces soft errors
  - Increase node capacitance to minimize impact of collected charge
  - Or use redundancy
  - Example:
  - *dual-interlocked cell (Dice)*



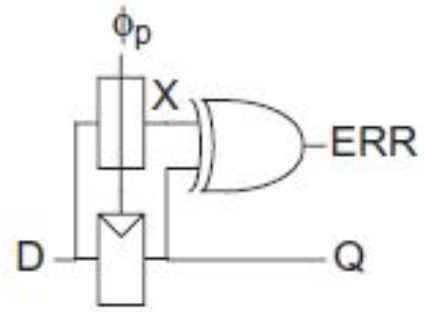
- Error-correcting codes
  - Correct for soft errors that do occur

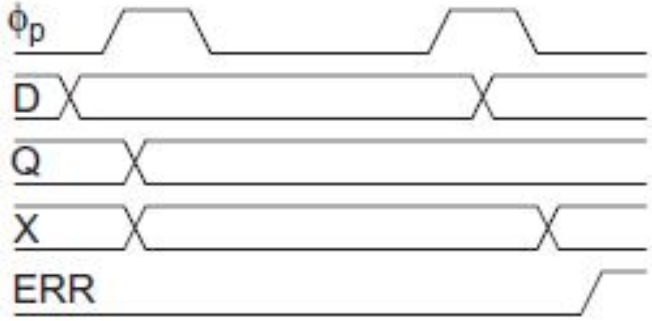
Adaptive Sequencing

- Designers have to include timing margin because of:
  - Voltage variation
  - Temperature variation
  - Process variation
  - Data dependency
  - Tool inaccuracies
- Alternative: run faster and check for near failures
  - Idea introduced as “Razor”
    - Increase frequency until at the verge of error
    - Can reduce cycle time by ~30%

Razor Circuit Idea

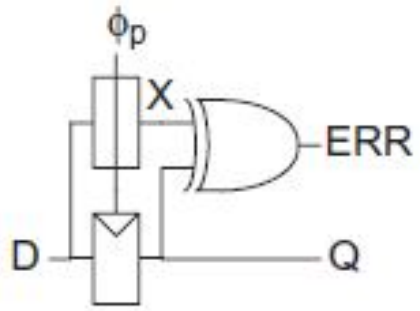
- The main path through the sequential element is unchanged
- A secondary checking path samples the input slightly later.
- If the two results agree, the circuit is operating correctly.
- If they differ, the data missed its setup time at the main path but made it for the later sampler, so the frequency is slightly too high or the voltage is slightly too low.
- The error is reported to repeat operations from the last known good state.



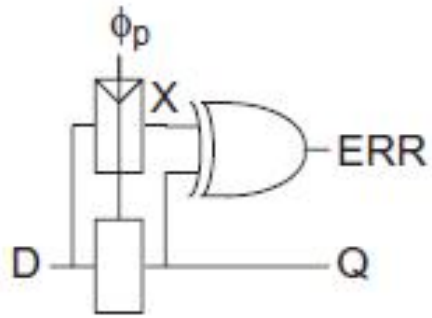


DSTB Idea

- Razor circuit drawback: the flip-flop may become metastable if D changes between setup and hold times
  - If Q resolves to the same value as the latch, no error will be flagged, but the propagation through the flip-flop can increase by an unbounded amount of time.



- Solution: Double Sampling with Time Borrowing
- Moves metastability out of the data path and onto the error path
- If data arrives slightly late, the pulsed latch will still capture it correctly.
  - The flip-flop may miss it (asserts ERR), or becomes metastable
  - Metastability can resolve before ERR is sampled (due to error path slack)

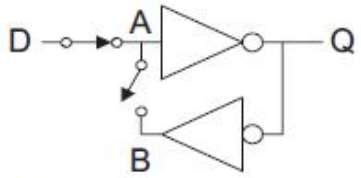


Synchronizer

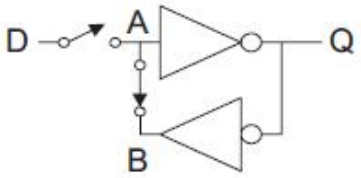
- Data changes during the aperture (between the setup and hold times) =>
  - Output may become unpredictable
  - Time to reach stable output may be unbounded (metastability)
- Need a guarantee that data is stable out of aperture
  - Impossible when working with external inputs
- Hence the need for synchronizers:
  - A circuit that accepts an input that can change at arbitrary times and produces an output aligned to the synchronizer's clock.

Metastability

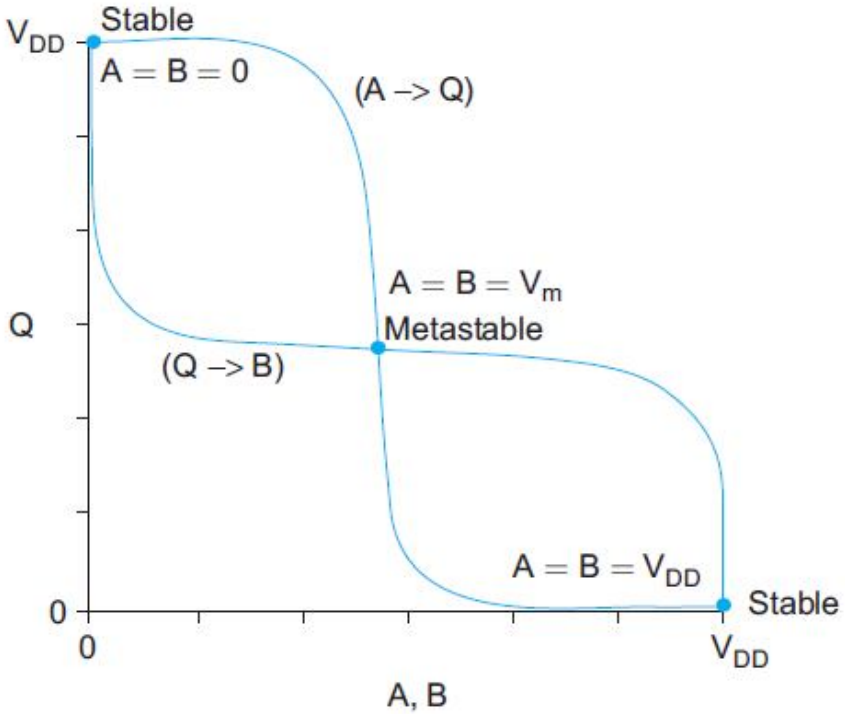
- Consider a simple latch: two inverters + two switches
- Two stable states, one metastable state

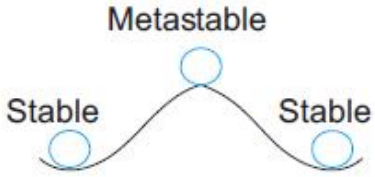


(a)



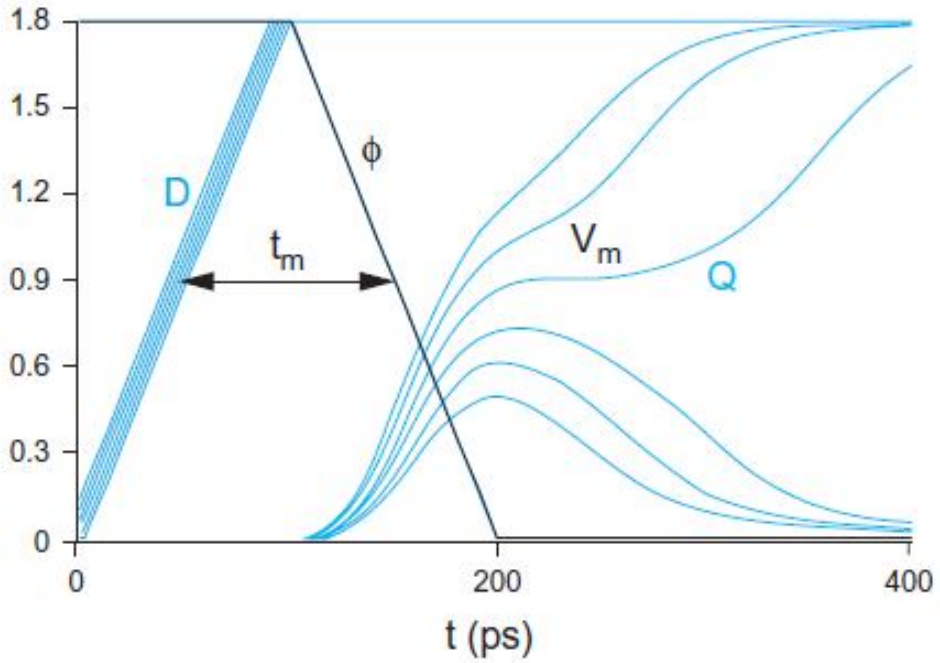
(b)

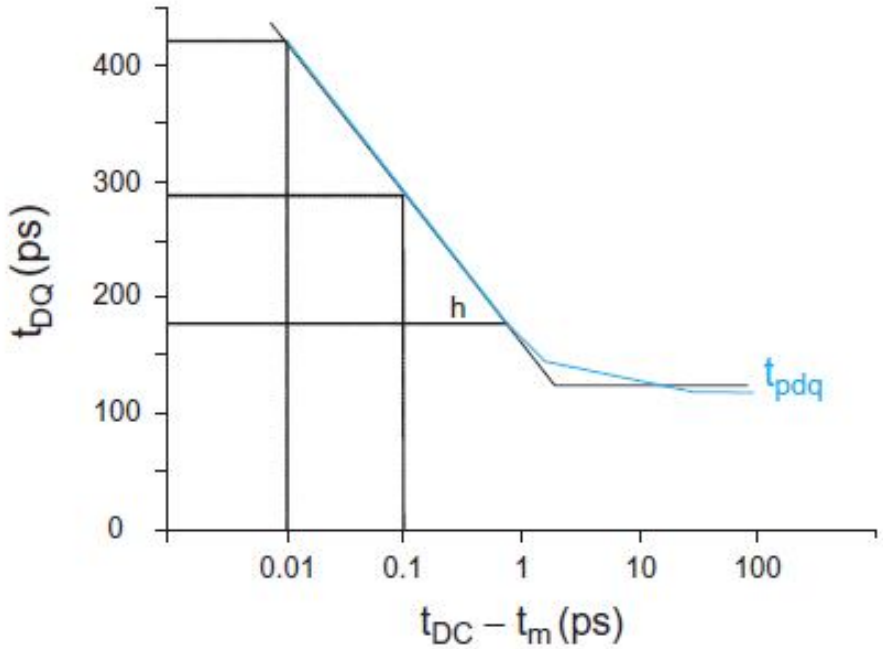




Metastable Delay (t_m)

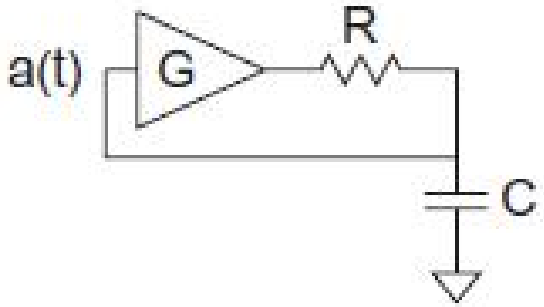
- If D changes before t_{su}, normal output (2 left sides)
- If D changes after t_{hold}, normal output (not seen)
- Define: t_m within aperture => V_m output (metastable)
- Right plot: time to recover from metastability





Metastability Modeling

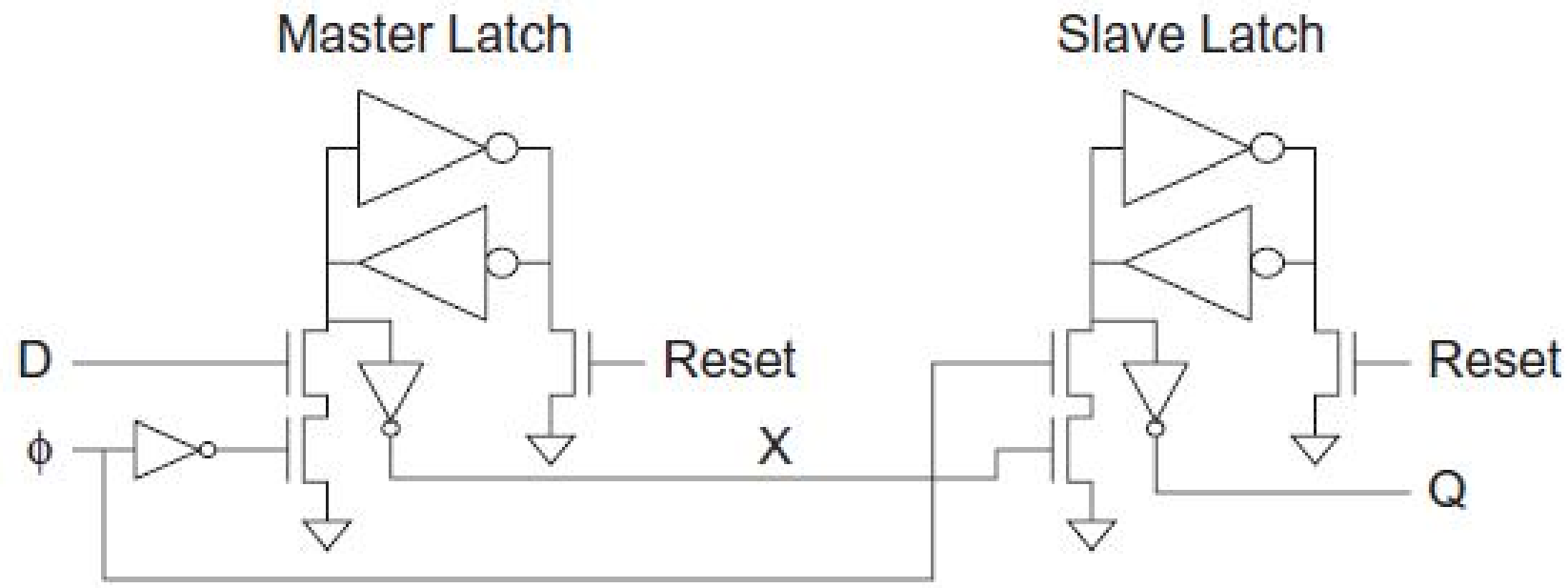
- Near metastable voltage (Vm) cascaded inverters behave like a linear amplifier with gain G
- Inverters' delays can be modeled with an output resistance R and load capacitance C.
- A latch should have a high gain and a small RC delay to resolve from metastability quickly



$$a(t) = a(0)e^{\frac{t}{\tau_s}};\ \tau_s = \frac{RC}{G-1}$$

Fast Synchronizer FF

- Conventional latches have data and clock transistors in series => increased delay
- This FF has a high G and a small RC => it's fast



Simple Synchronizer