Sequential Design

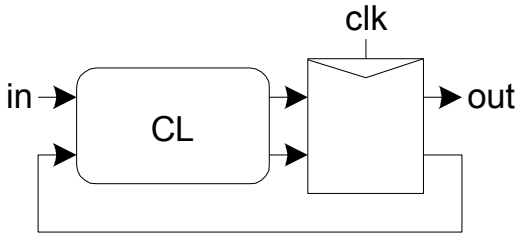
E155

Outline

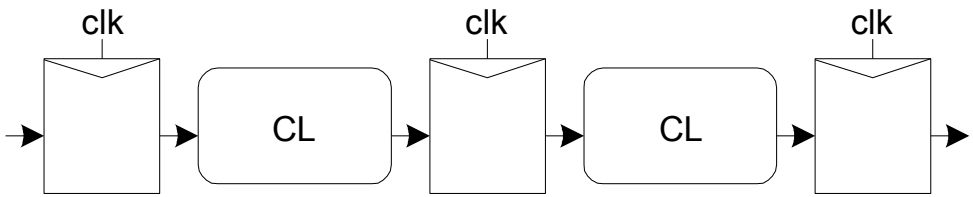
- Sequential design
- Dynamic discipline
- Metastability
- Synchronizer

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



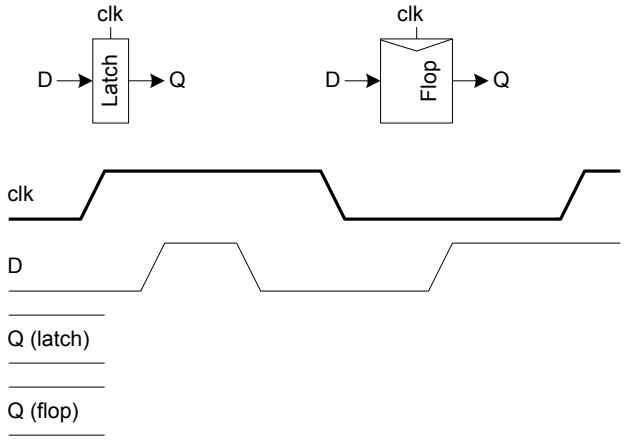
Pipeline

Sequencing Overhead

- Use flip-flops to delay fast tokens so they move through exactly one stage each cycle.
- Inevitably adds some delay to the slow 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

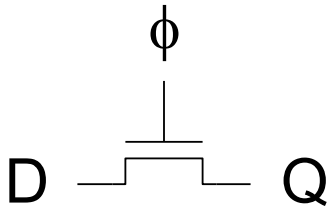
Sequencing Elements

- **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
- Timing Diagrams
  - Transparent
  - Opaque
  - Edge-trigger



Latch Design

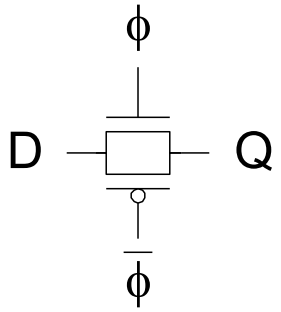
- Pass Transistor Latch
- Pros
  - +
  - +
- Cons
  - –
  - –
  - –
  - –
  - –
  - –



Used in 1970's

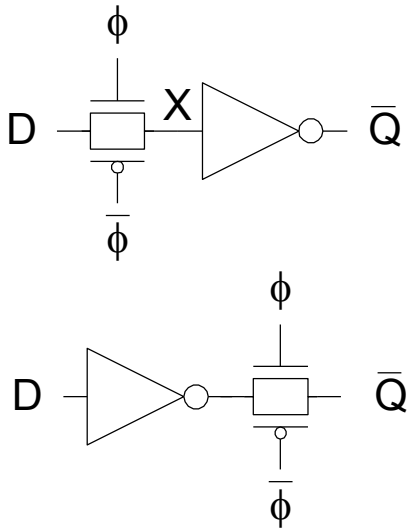
Latch Design

- Transmission gate
  - +
  - -



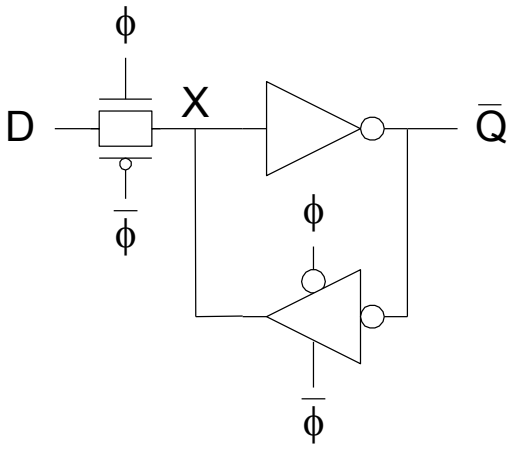
Latch Design

- Inverting buffer
  - +
  - +
  - + Fixes either
    - •
    - •
  - –



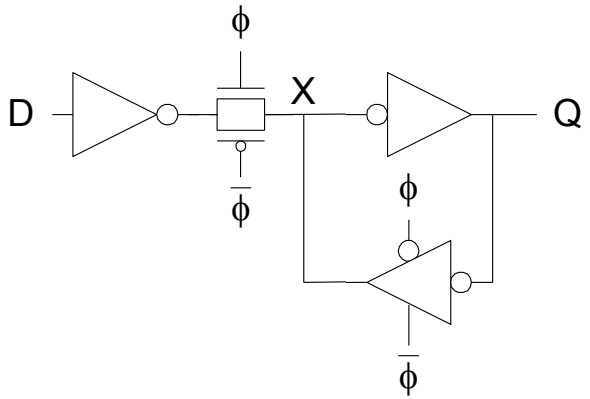
Latch Design

- Tristate feedback
  - \+
  - –
- Static latches are now essential because of leakage



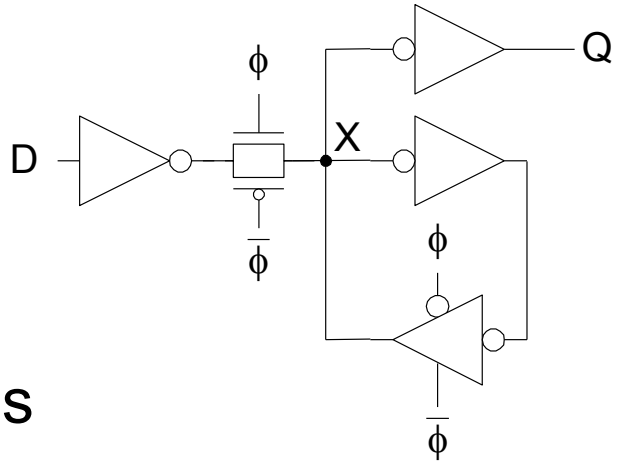
Latch Design

- Buffered input
  - +
  - +



Latch Design

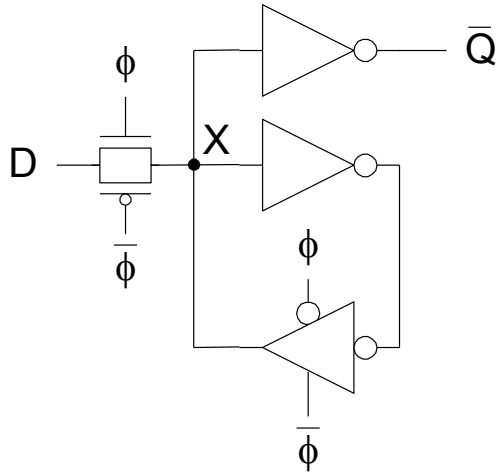
- Buffered output
  - +



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

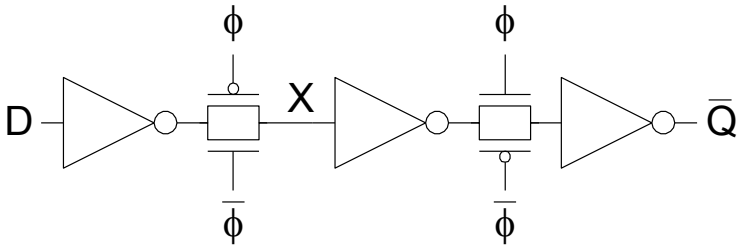
Latch Design

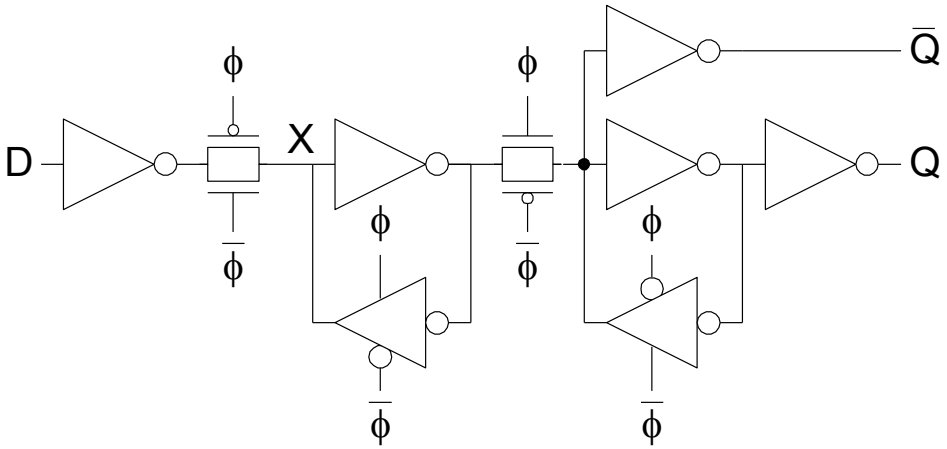
- Datapath latch
  - +
  - +
  - -



Flip-Flop Design

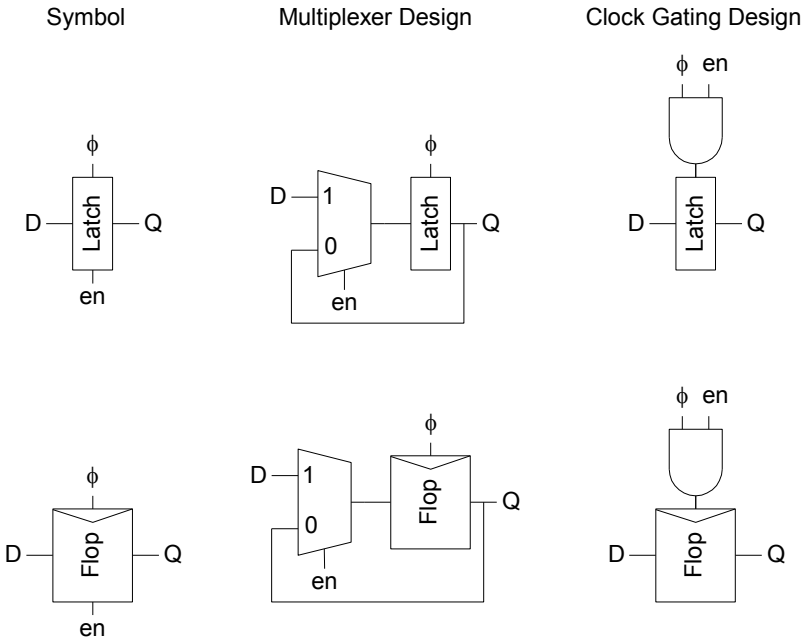
- Flip-flop is built as pair of back-to-back latches





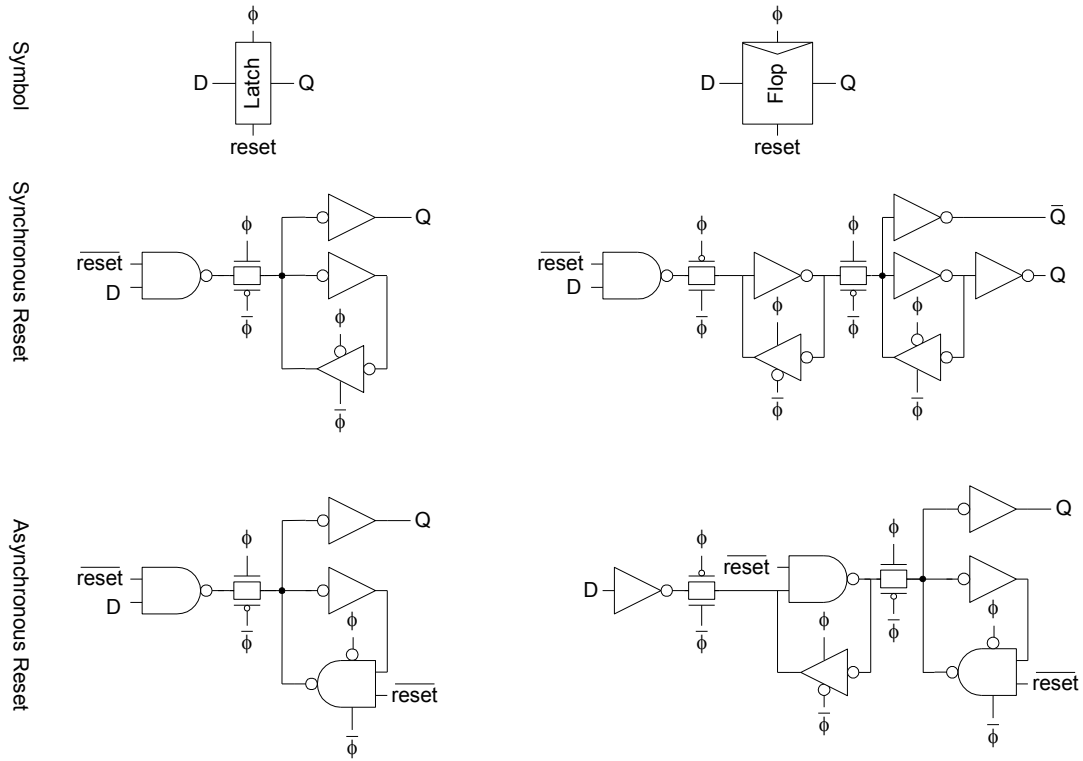
Enable

- Enable: ignore clock when en = 0
  - Mux: increase latch D-Q delay
  - Clock Gating: increase en setup time, skew



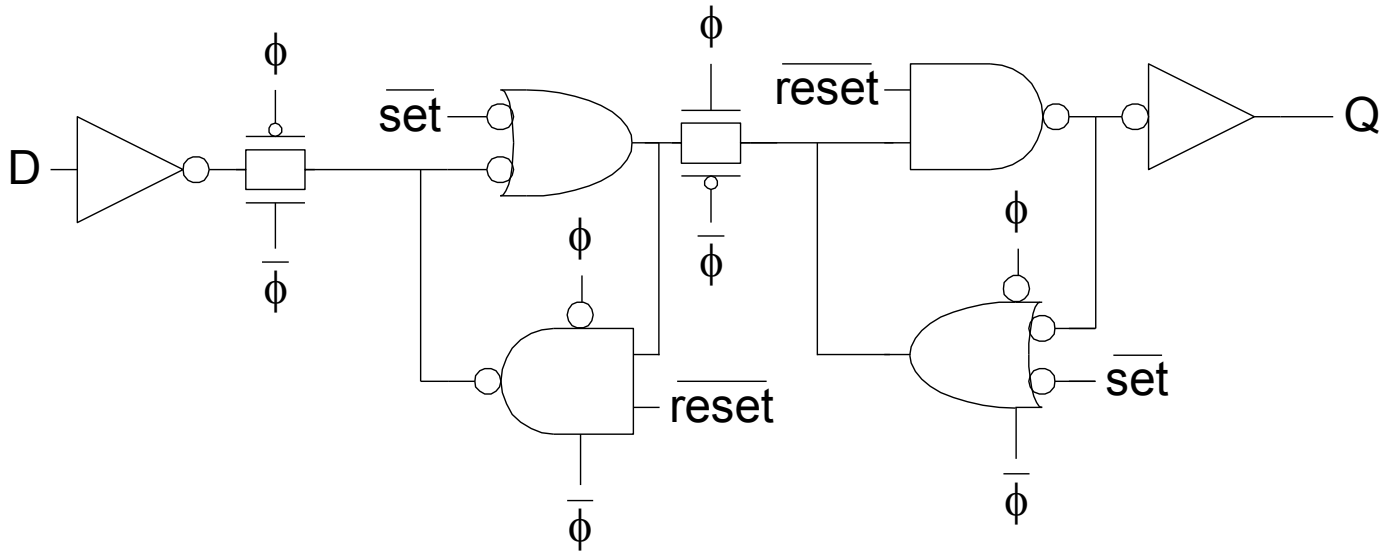
Reset

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



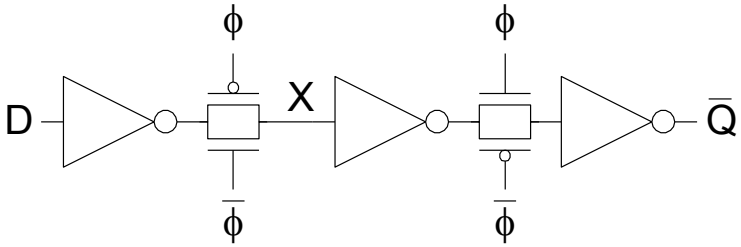
Set / Reset

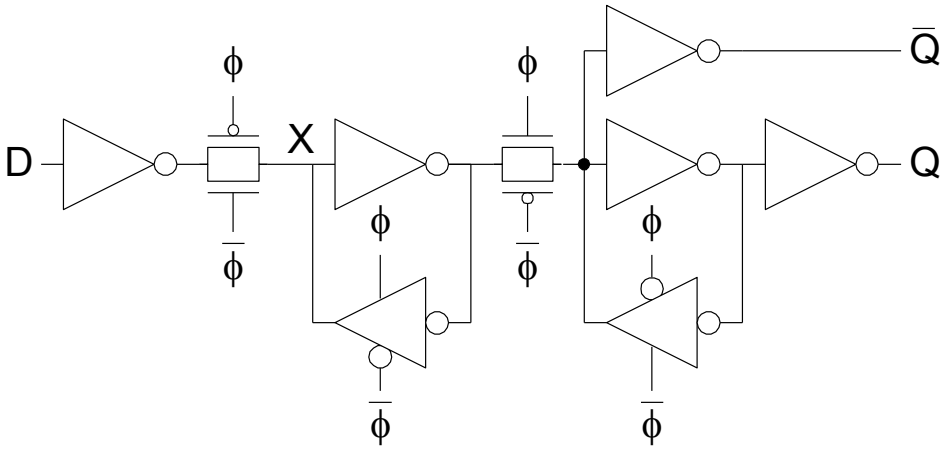
- Set forces output high when enabled
- Flip-flop with asynchronous set and reset



Flip-Flop Design

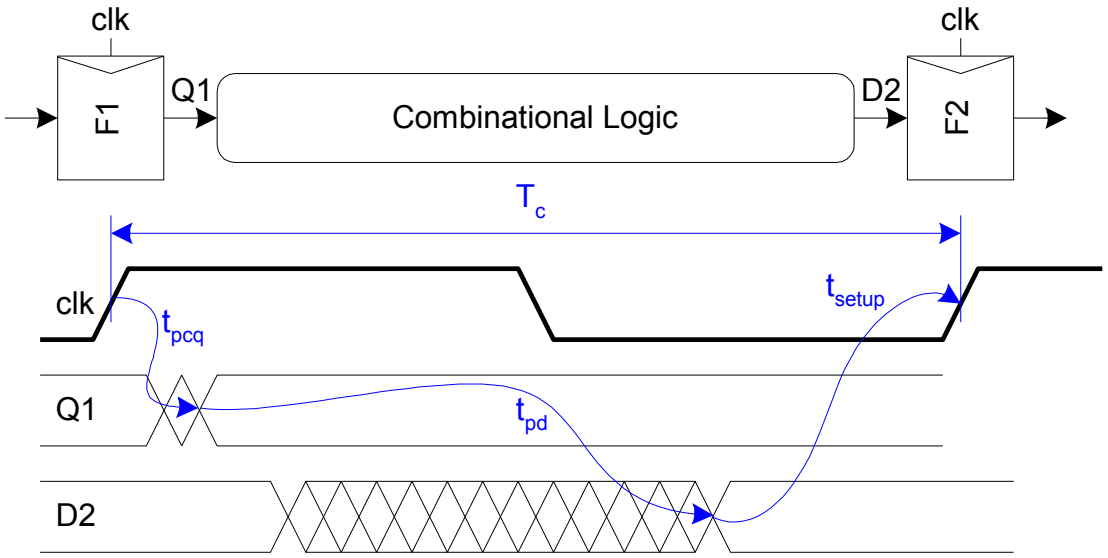
- Flip-flop is built as pair of back-to-back latches





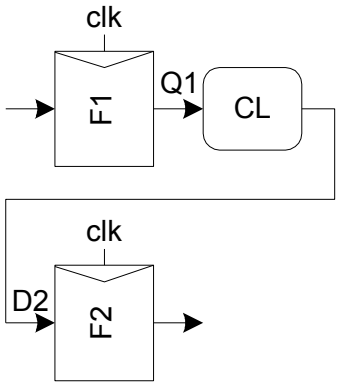
Max-Delay: Flip-Flops

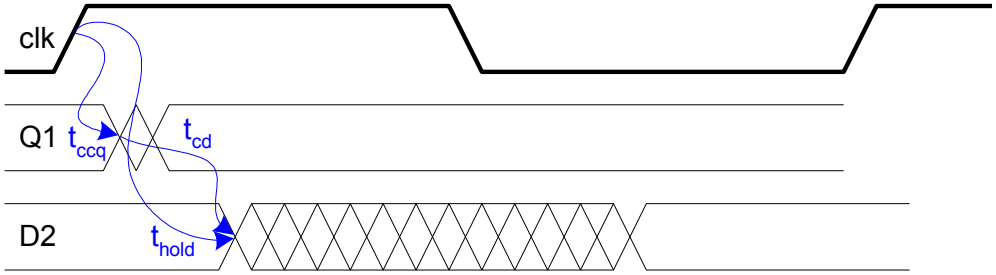
$$t_{pd} \leq T_c - \underbrace{\left(t_{\text{setup}} + t_{pcq}\right)}_{\text{sequencing overhead}}$$



Min-Delay: Flip-Flops

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





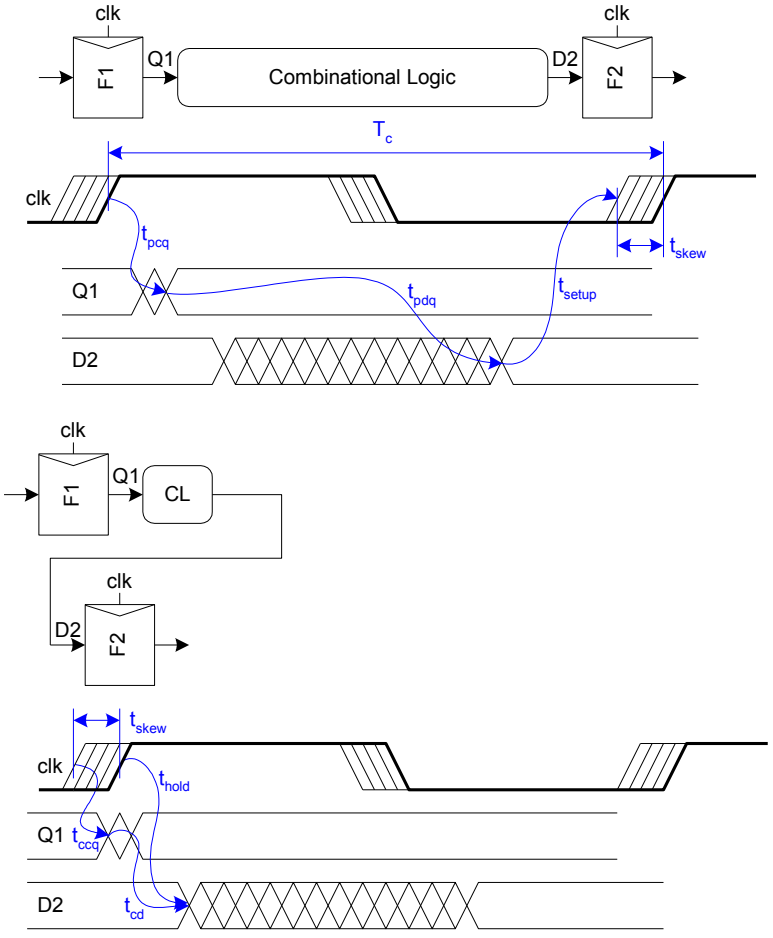
Clock Skew

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

Skew: Flip-Flops

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

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

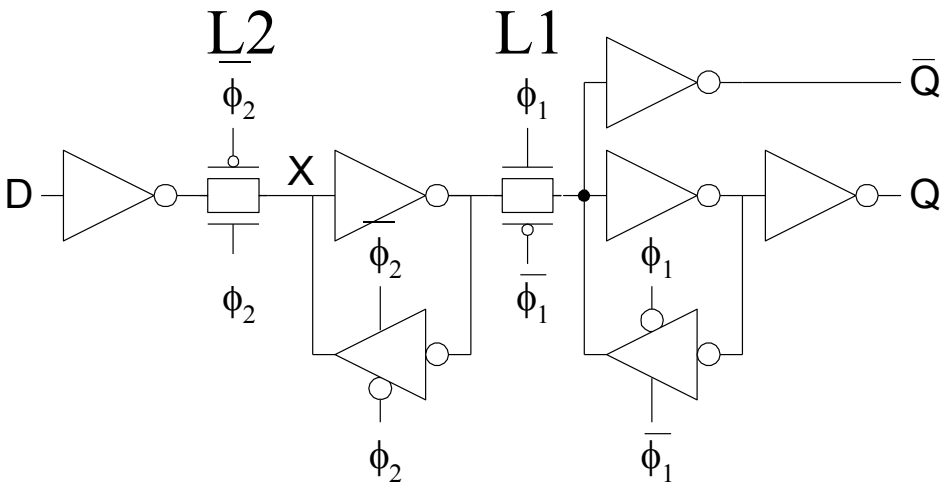


Two-Phase Clocking

- If setup times are violated, reduce clock speed
- If hold times are violated, chip fails at any speed
- In this class, working chips are most important
  - No tools to analyze clock skew
- An easy way to guarantee hold times is to use 2-phase latches with big nonoverlap times
- Call these clocks ϕ_1, ϕ_2 (ph1, ph2)

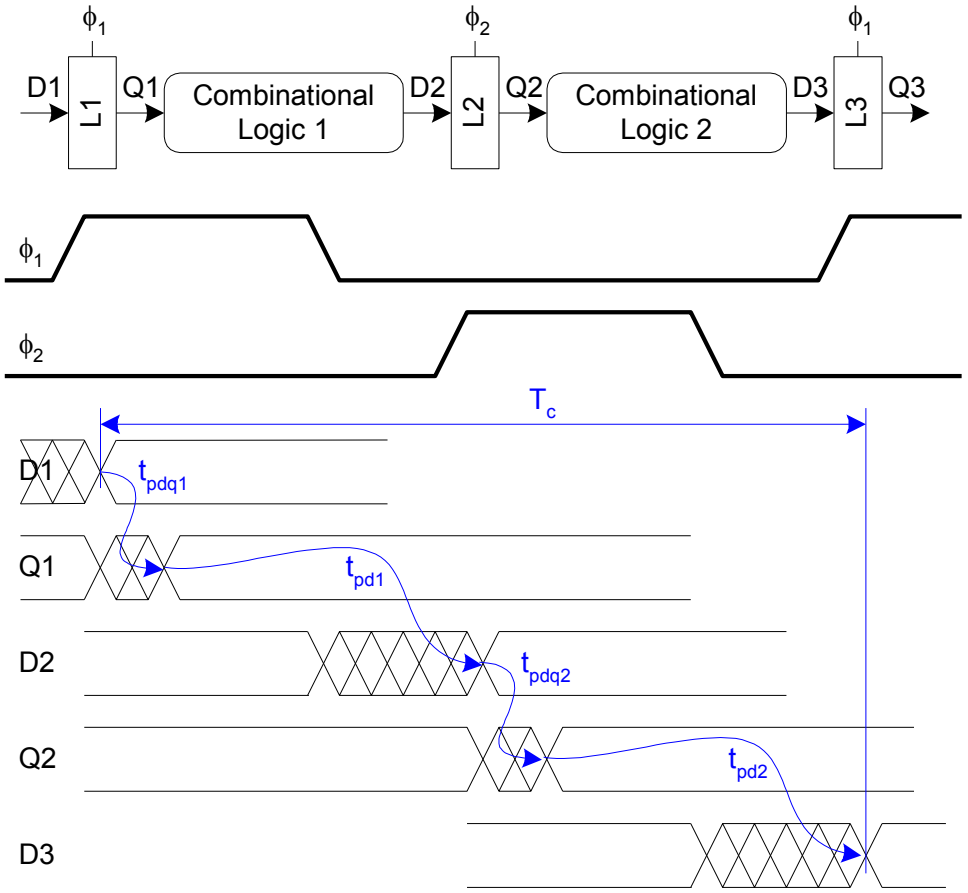
Safe Flip-Flop

- Safe designs use flip-flop with non-overlapping clocks
  - Slow – non-overlap adds to setup time
  - But no hold time problems
- In industry, use a better timing analyzer
  - Add buffers to slow signals if hold time is at risk

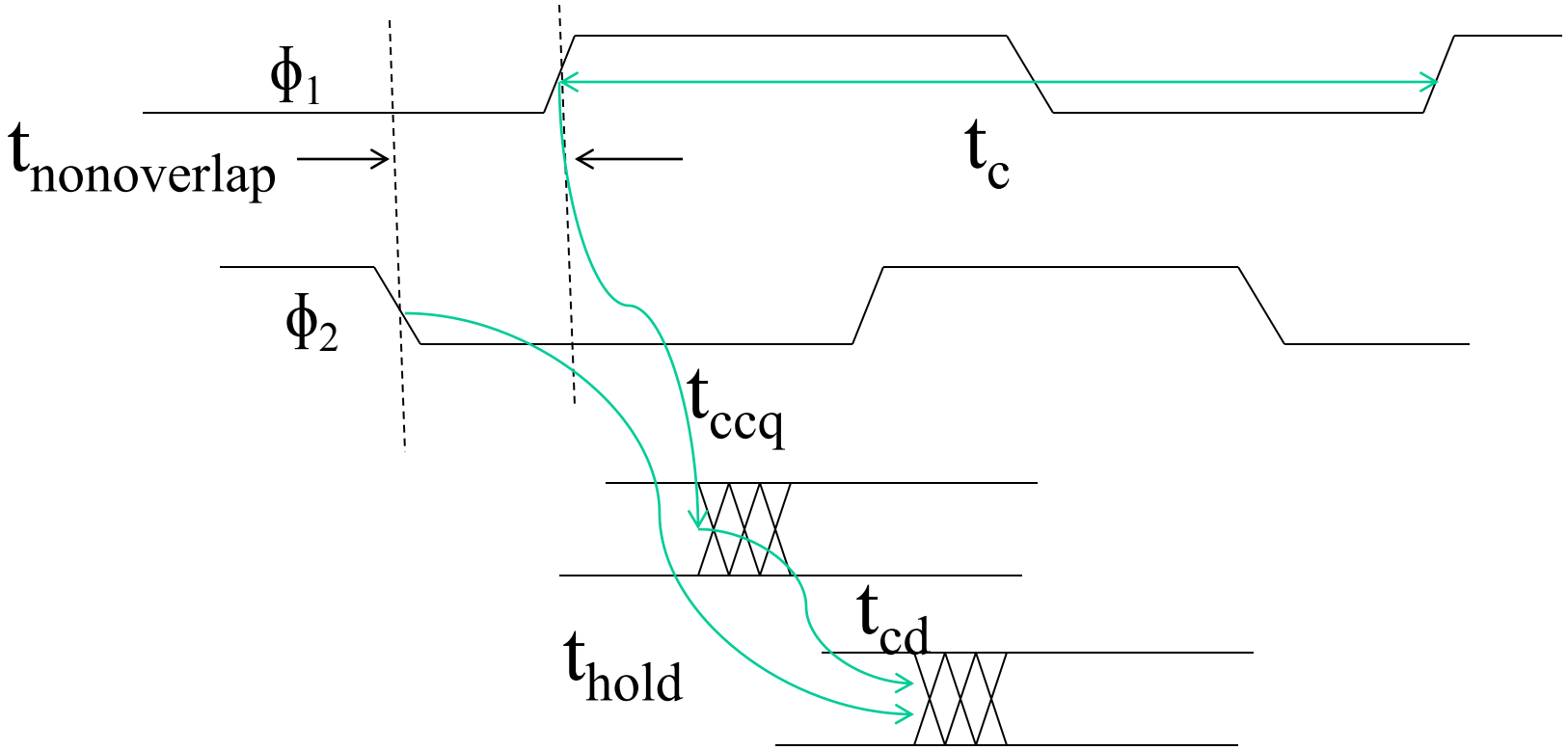


Max Delay: 2-Phase Latches

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



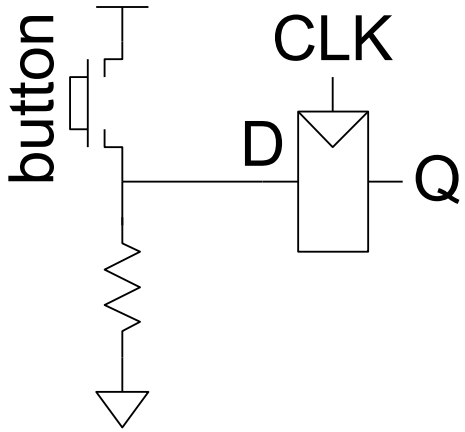
Min Delay 2-Phase Flip-Flop

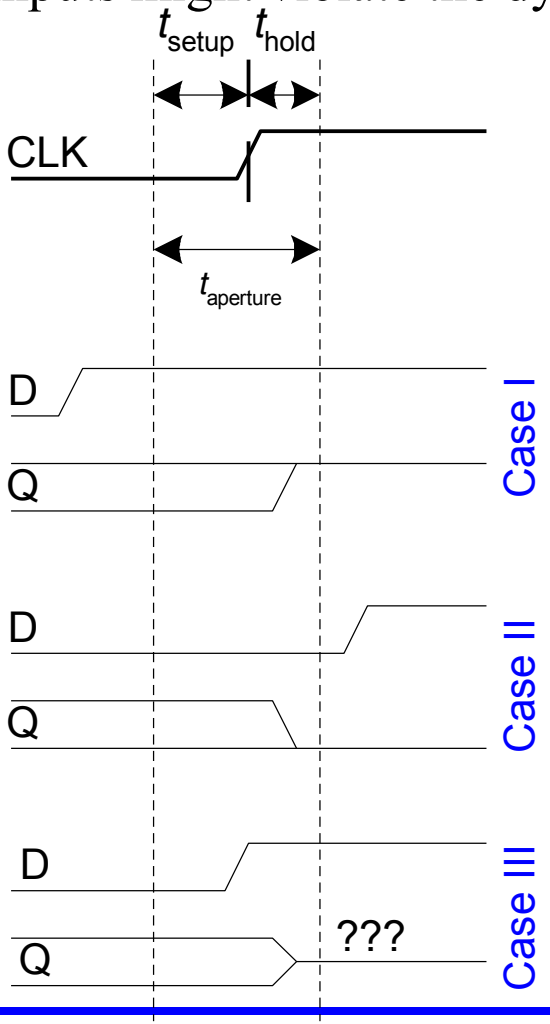


$$t_{cd} > t_{hold} - t_{ccq} - t_{nonoverlap}$$

Violation of Dynamic Discipline

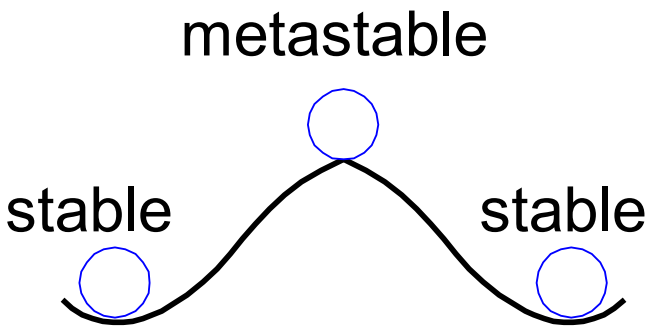
- Asynchronous (for example, user) inputs might violate the dynamic discipline





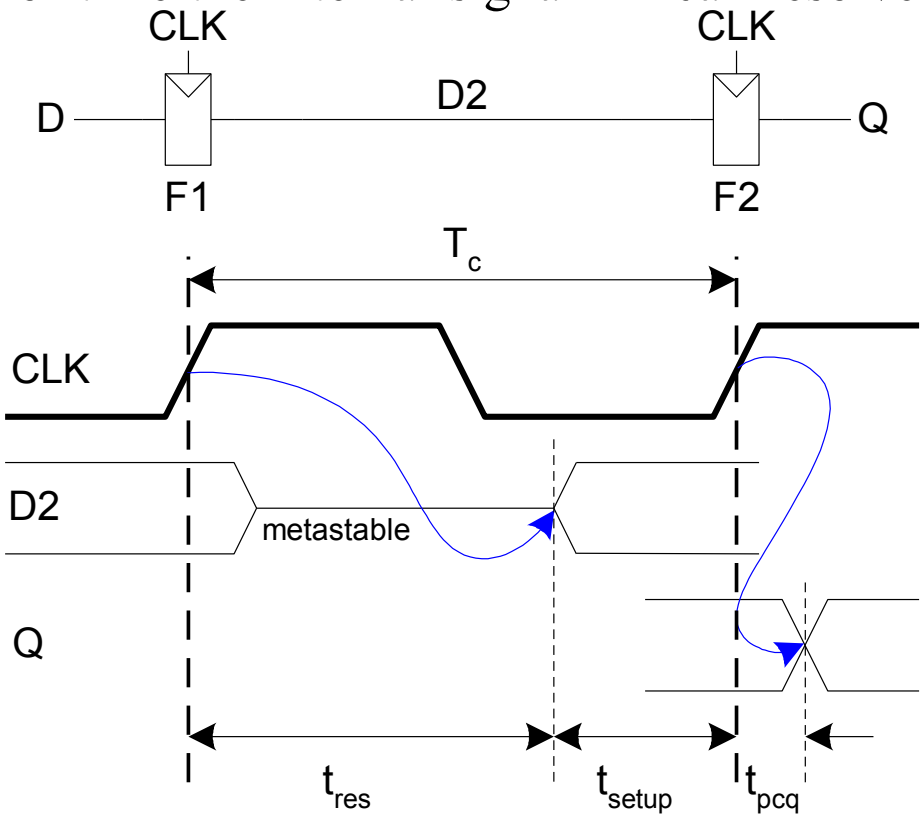
Metastability

- Any bistable device has two stable states and a metastable state between them
- A flip-flop has two stable states (1 and 0) and one metastable state
- If a flip-flop lands in the metastable state, it could stay there for an undetermined amount of time



Synchronizer Internals

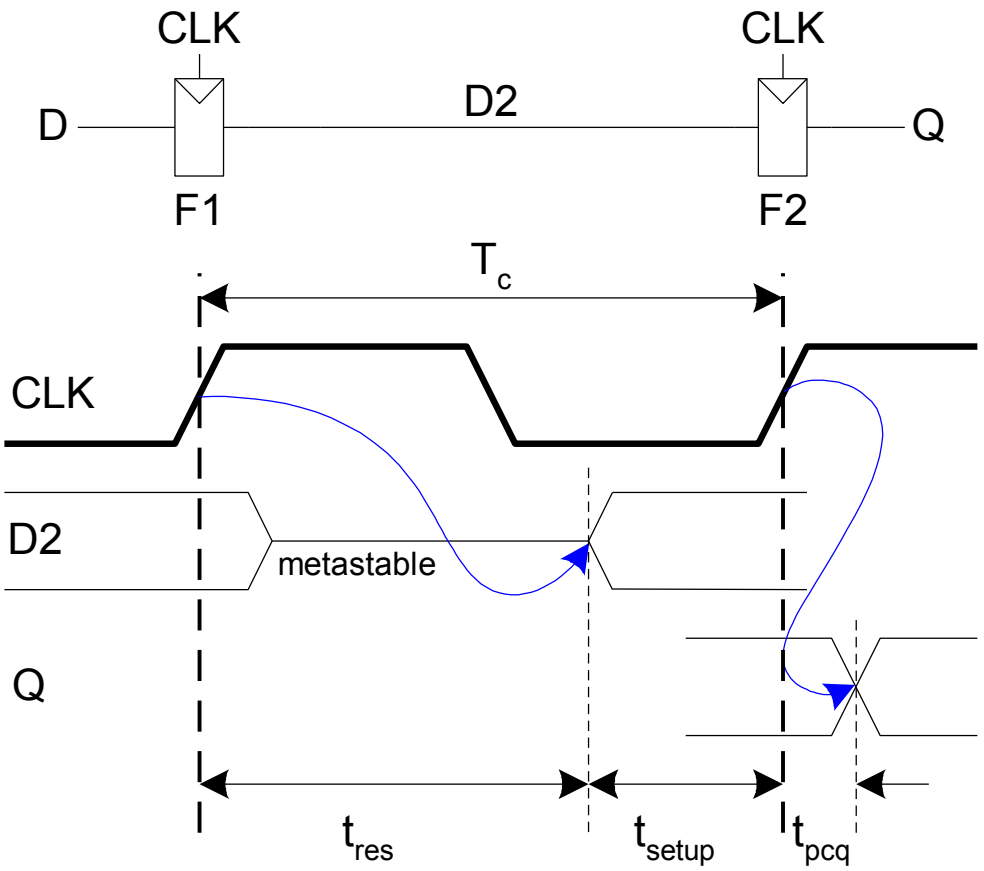
- A synchronizer can be built with two back-to-back flip-flops.
- Suppose the input D is transitioning when it is sampled by flip-flop 1, F1.
- The amount of time the internal signal D2 can resolve to a 1 or 0 is (T_c - t_{setup}).



Synchronizer Probability of Failure

For each sample, the probability of failure of this synchronizer is:

$$P(\text{failure}) = (T_0/T_c)\, e^{-(T_c - t_{setup})/\tau}$$



Synchronizer Mean Time Before Failure

- If the asynchronous input changes once per second, the probability of failure per second of the synchronizer is simply *P*(failure).
- In general, if the input changes *N* times per second, the probability of failure per second of the synchronizer is:

$$P(\text{failure})/\text{second} = (NT_0/T_c)\, e^{-(T_c - t_{setup})/\tau}$$

- Thus, the synchronizer fails, on average, 1/[*P*(failure)/second]
- This is called the *mean time between failures*, MTBF:

$$\text{MTBF} = 1/[P(\text{failure})/\text{second}] = (T_c/NT_0)\, e^{(T_c - t_{setup})/\tau}$$