CHAPTER 6

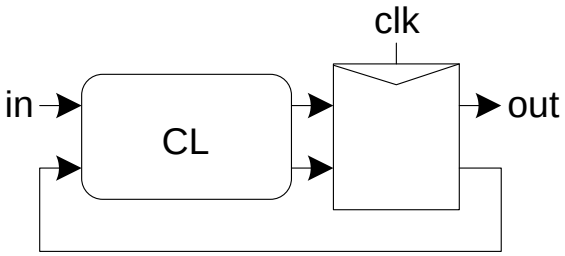
Sequential Circuit Design

Outline

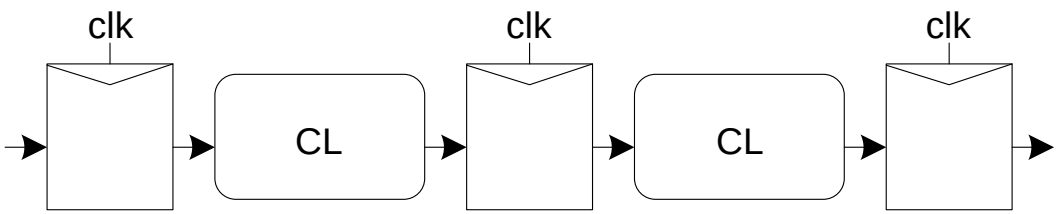
1. **Sequencing**
2. Sequencing Element Design
3. Max and Min-Delay
4. Time Borrowing
5. Clock Skew
6. Two-Phase Clocking

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
- In most circuits, dispersion is high
  - Delay fast tokens so they don't catch slow ones.

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

Outline

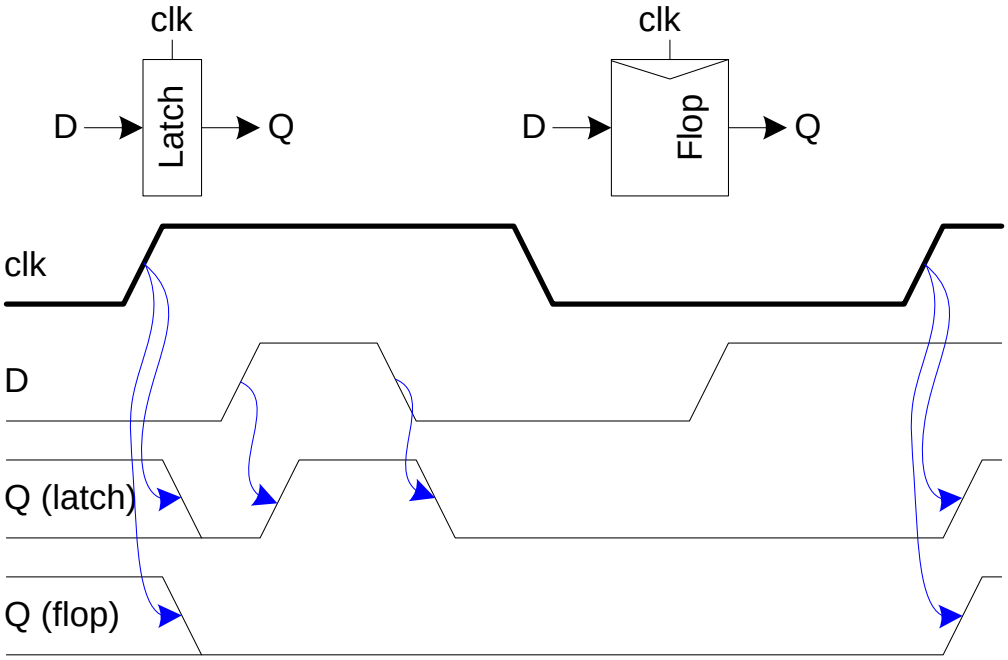
1. Sequencing
2. **Sequencing Element Design**
3. Max and Min-Delay
4. Time Borrowing
5. Clock Skew
6. Two-Phase Clocking

Static vs Dynamic Storage

- Static storage
  - preserve state as long as the power is on
  - have positive feedback (regeneration) with an internal connection between the output and the input
  - useful when updates are infrequent (clock gating)
- Dynamic storage
  - store state on parasitic capacitors
  - only hold state for short periods of time (milliseconds)
  - require periodic refresh
  - usually simpler, so higher speed and lower power

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

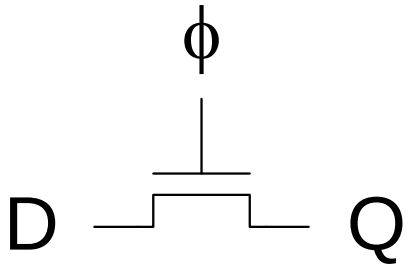


Latches vs Flipflops

- Latches
  - level sensitive circuit that passes inputs to Q when the clock is high (or low) - transparent mode
  - input sampled on the falling edge of the clock is held stable when clock is low (or high) - hold mode
- Flipflops (edge-triggered)
  - edge sensitive circuits that sample the inputs on a clock transition
    - positive edge-triggered: $0 \rightarrow 1$
    - negative edge-triggered: $1 \rightarrow 0$
  - built using latches (e.g., master-slave flipflops)

Latch Design

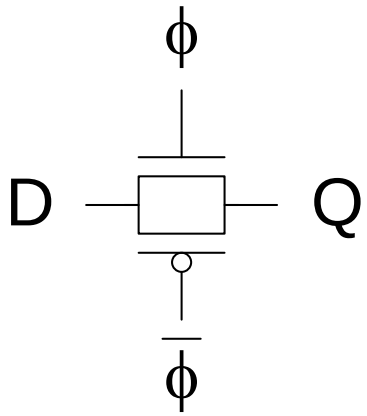
- Pass Transistor Latch
- Pros
  - + Tiny
  - + Low clock load
- Cons
  - – V_t drop (not rail-to-rail)
  - – Nonrestoring
  - – Backdriving
  - – Output noise sensitivity
  - – Dynamic (floating when opaque)
  - – Diffusion input



Used in 1970's

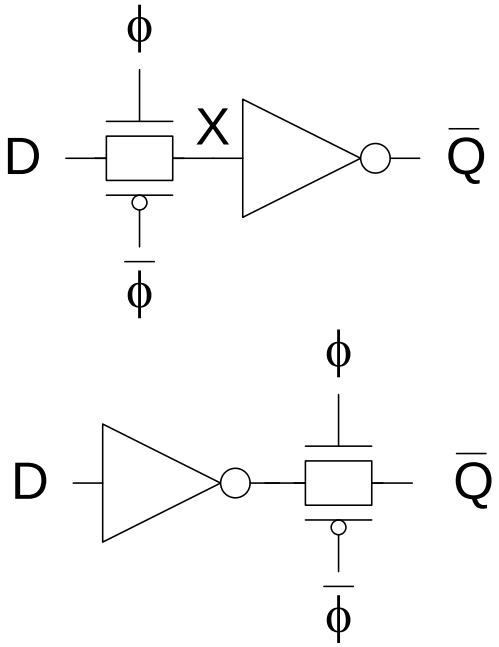
Latch Design

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



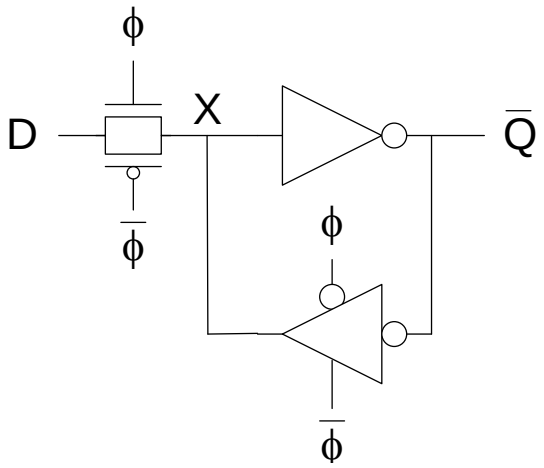
Latch Design

- Inverting buffer
  - + Restoring
  - + No backdriving
  - + Fixes either
    - Output noise sensitivity
    - Or diffusion input
  - – Inverted output

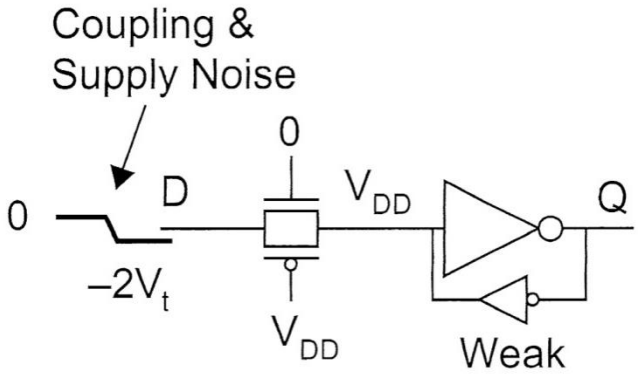


Latch Design

- Tristate feedback
  - + Static
  - – Backdriving risk
  - – Diffusion input



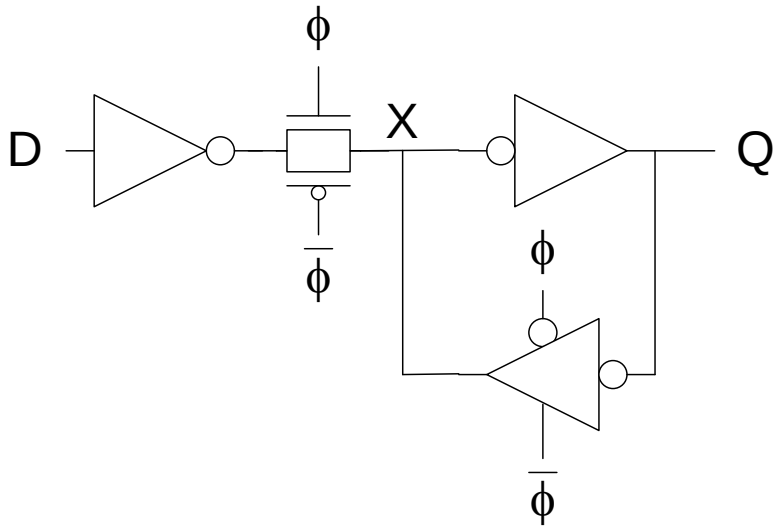
- Noise on diffusion input
  - – Noise couple will turn ON the "OFF" transmission gate and destroy the output



- Static latches are now essential

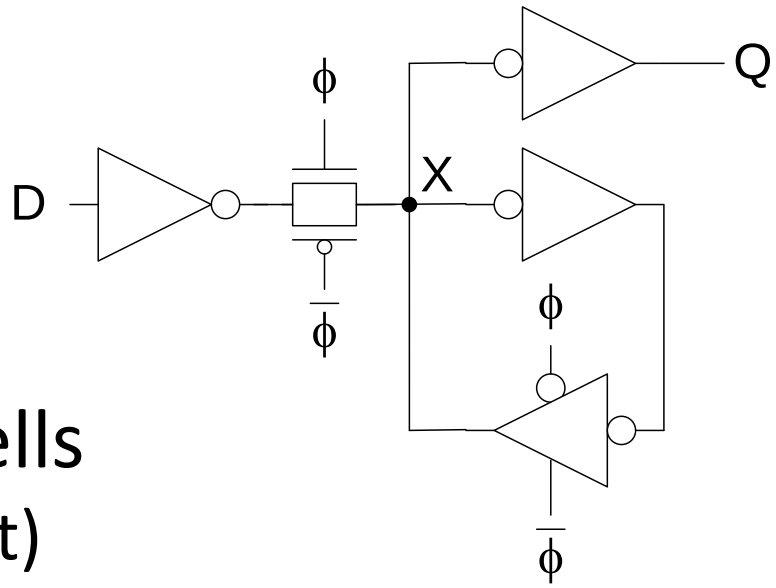
Latch Design

- Buffered input
  - + Fixes diffusion input
  - + Noninverting
  - – Ouput noise backdriving



Latch Design

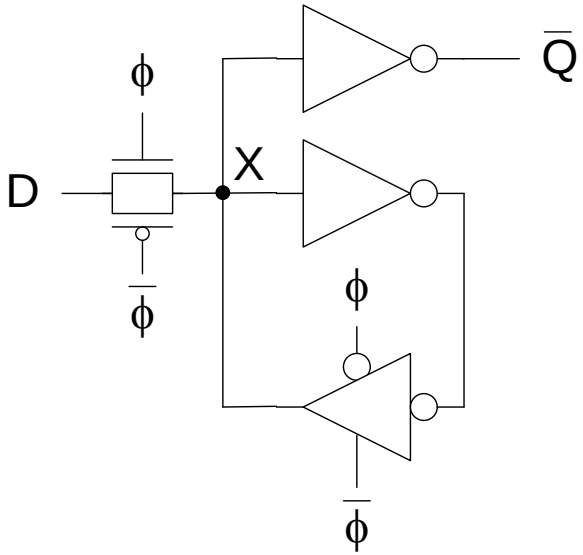
- Buffered output
  + No backdriving



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

Latch Design

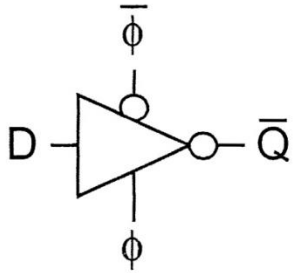
- Datapath latch
  - + Smaller, faster
  - - Unbuffered input
  - - Need careful input noise control

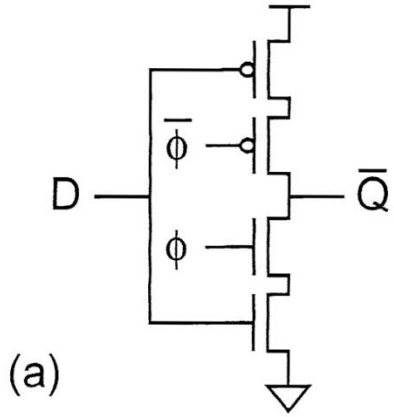


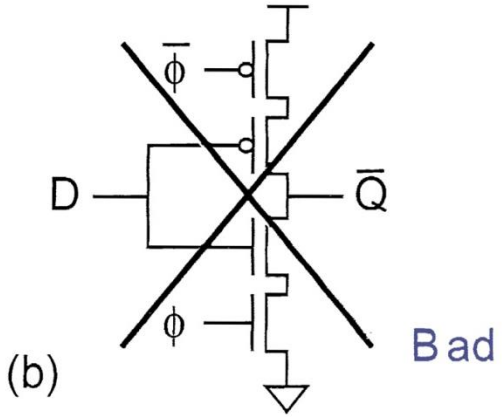
Clocked CMOS : C²MOS

- C²MOS latch
  - \+ Smaller
  - \- Slower

D toggle will cause charge sharing noise while opaque.

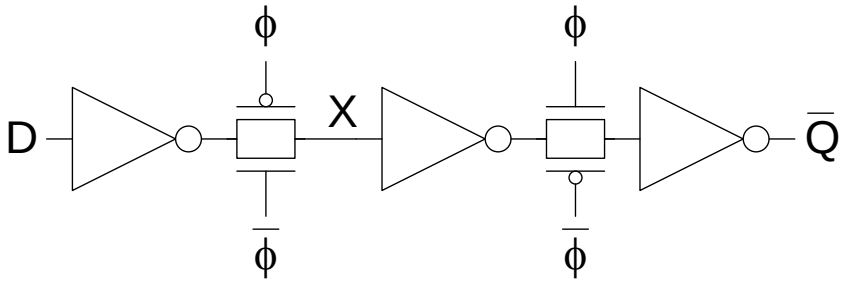


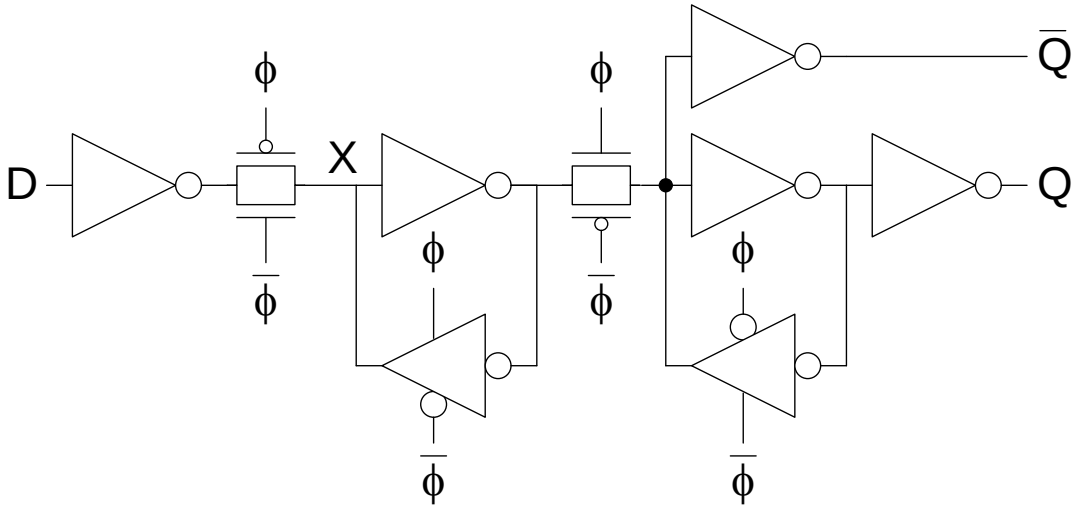




Flip-Flop Design

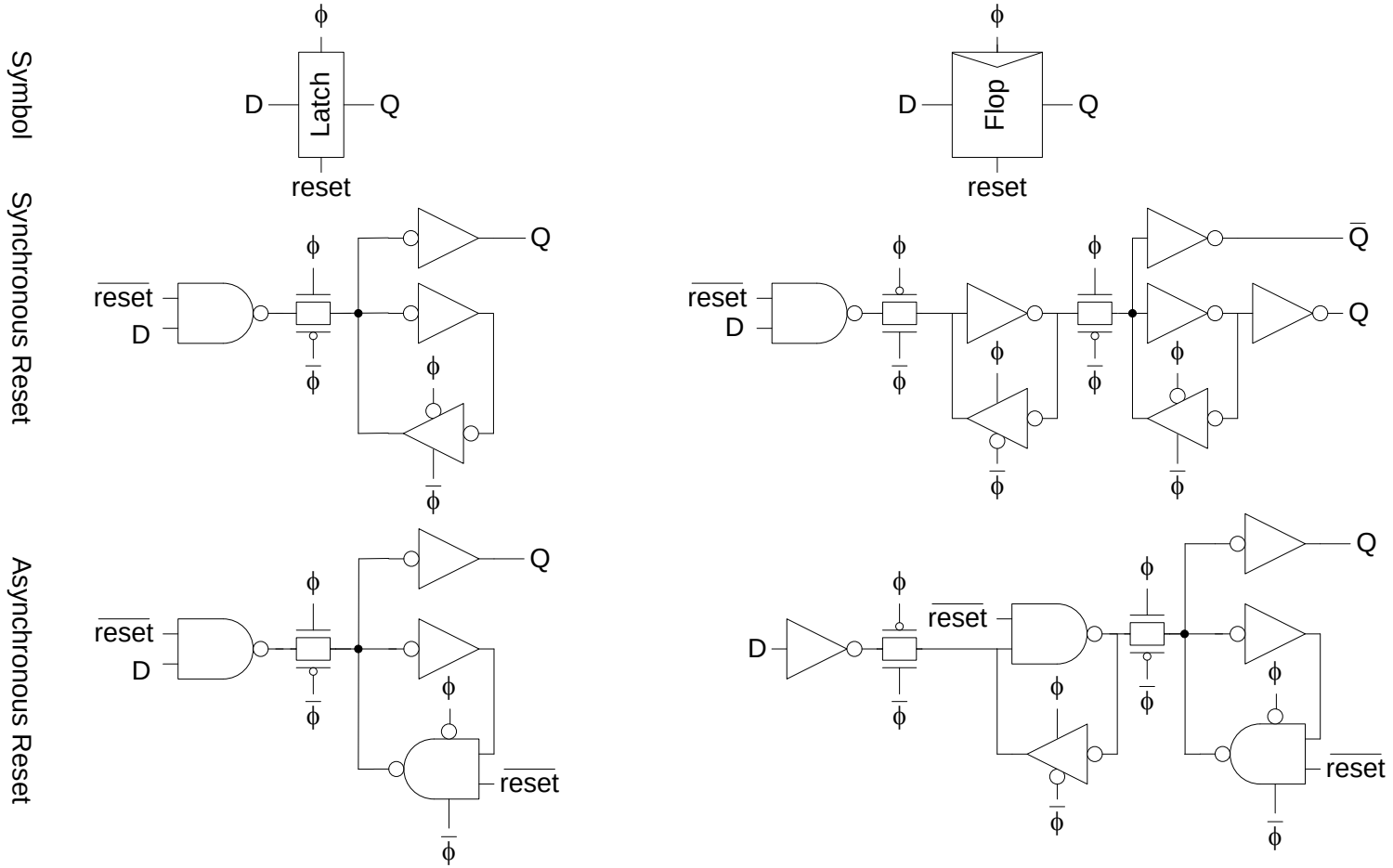
- Flip-flop is built as pair of back-to-back latches





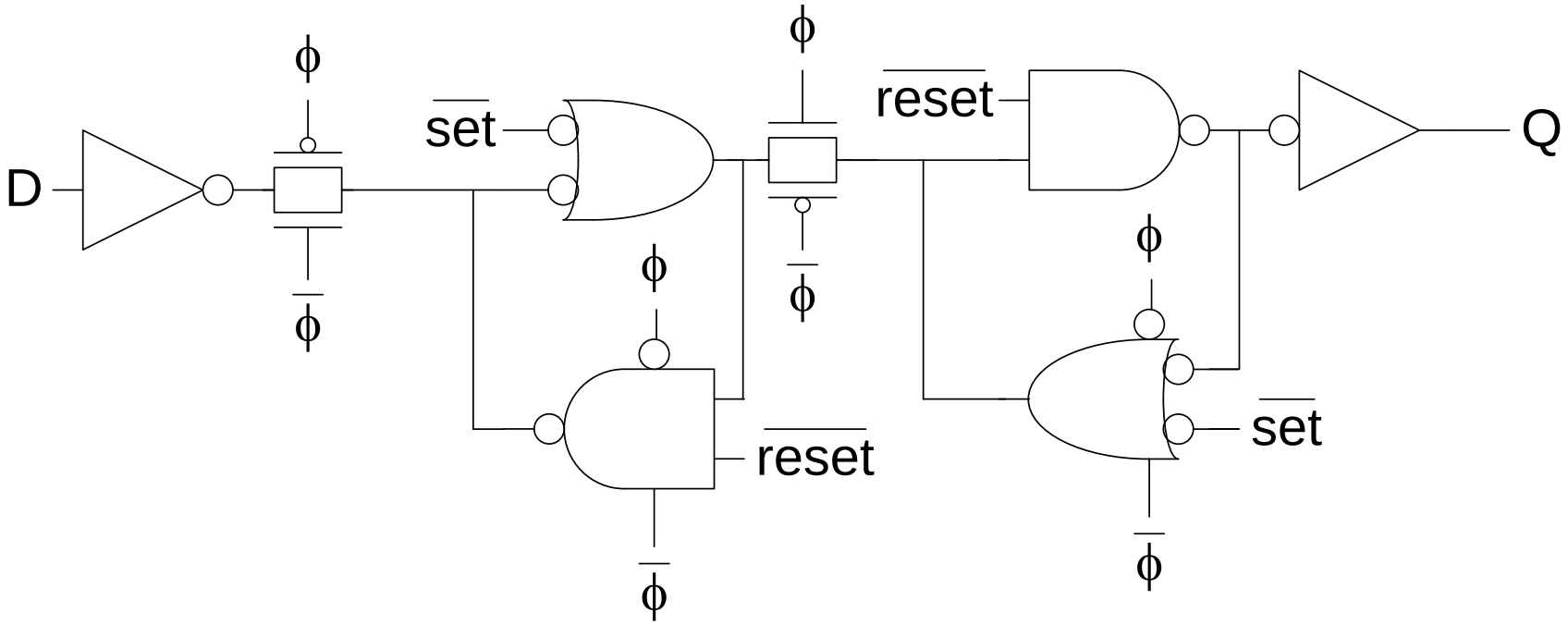
Reset

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



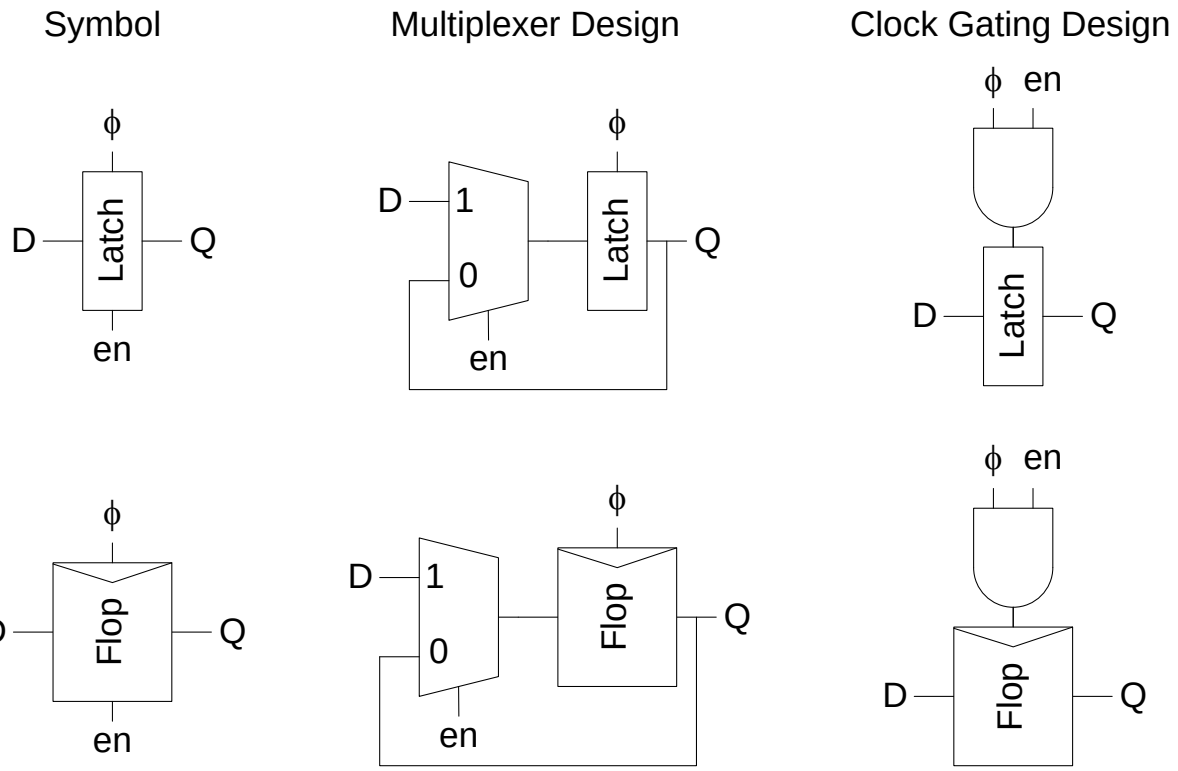
Set / Reset

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



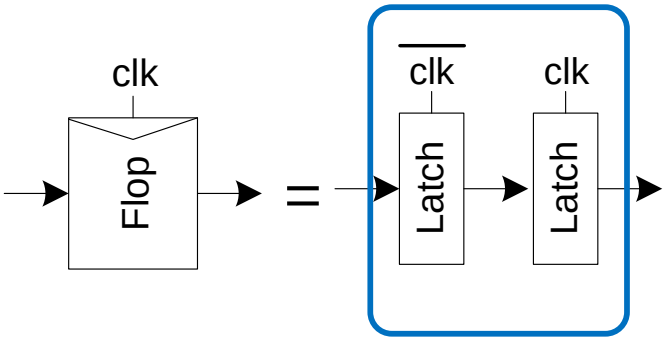
Enable

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



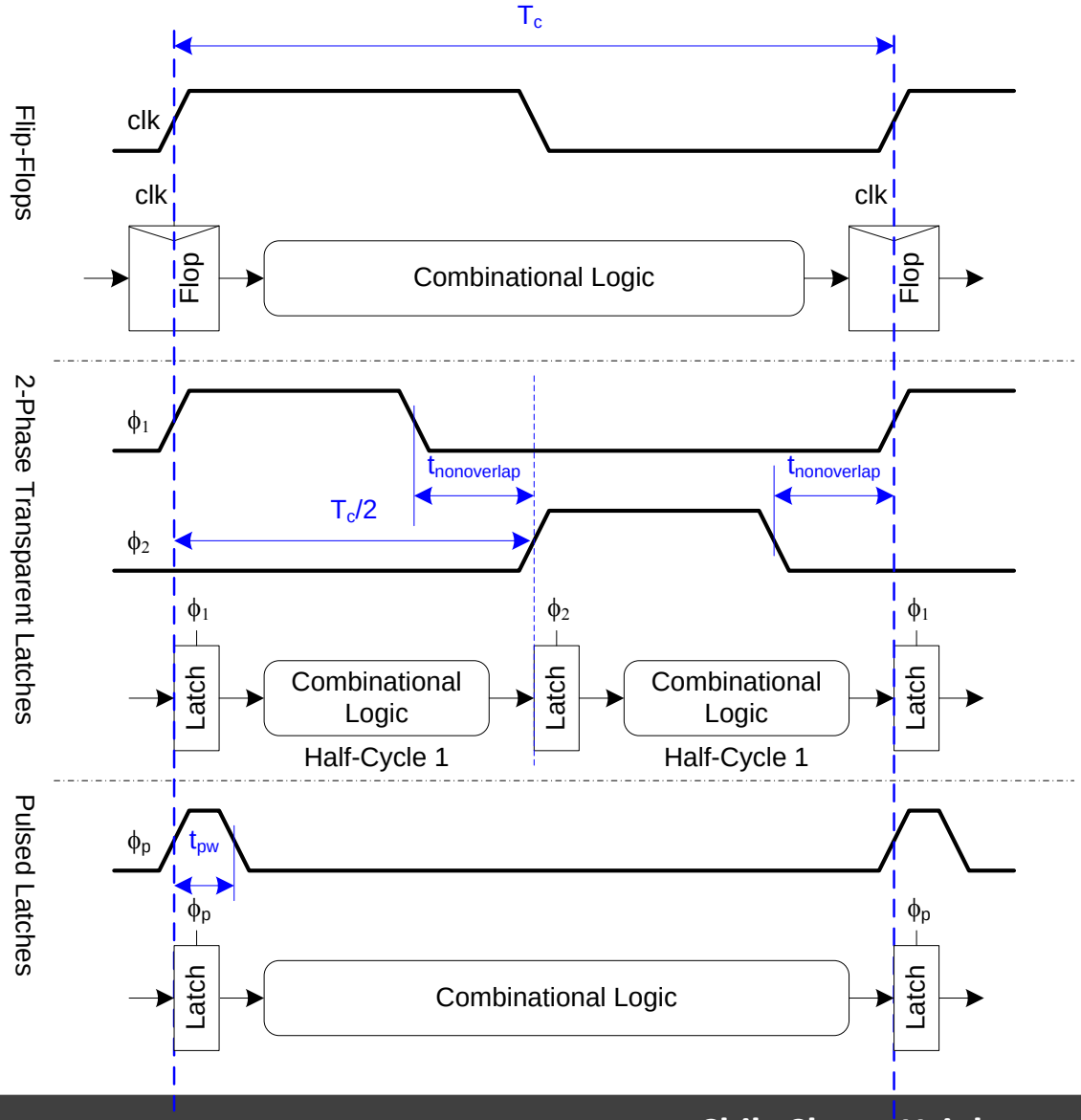
Sequencing Methods

- Flip-flops



- 2-Phase Latches

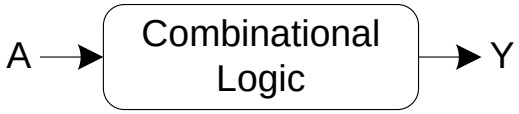
- Pulsed Latches

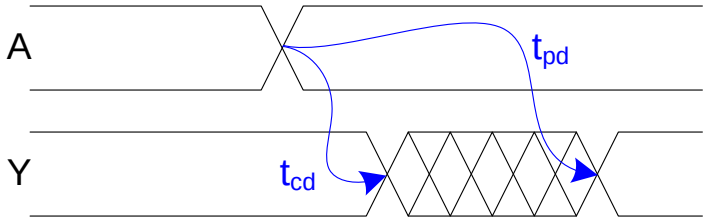


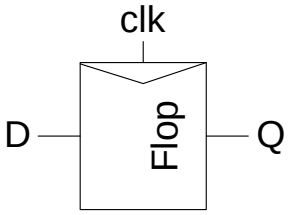
Timing Diagrams

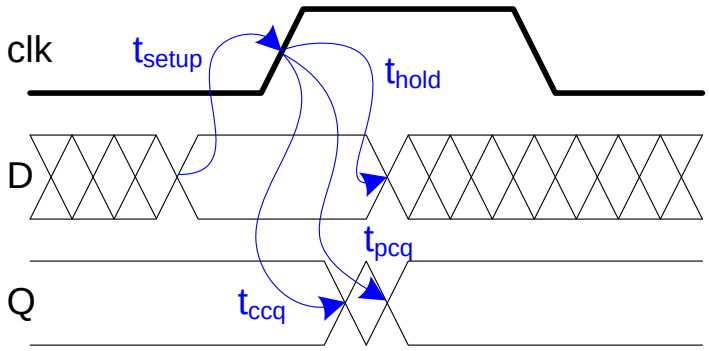
Contamination and Propagation Delays

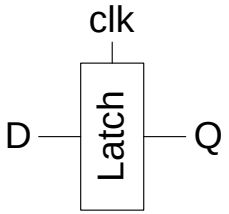
t_{pd}	Logic Prop. Delay
t_{cd}	Logic Cont. Delay
t_{pcq}	Latch/Flop Clk-Q Prop Delay
t_{ccq}	Latch/Flop Clk-Q Cont. Delay
t_{pdq}	Latch D-Q Prop Delay
t_{cdq}	Latch D-Q Cont. Delay
t_{setup}	Latch/Flop Setup Time
t_{hold}	Latch/Flop Hold Time

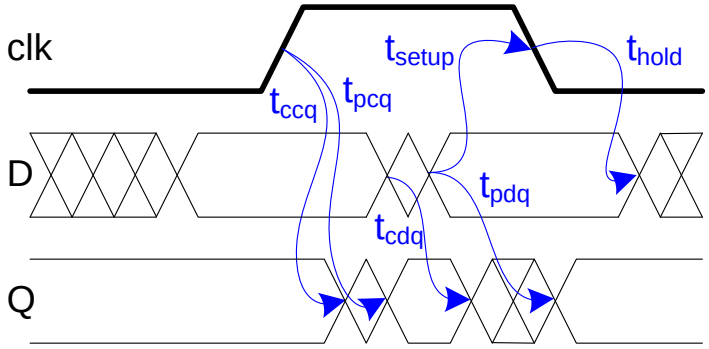








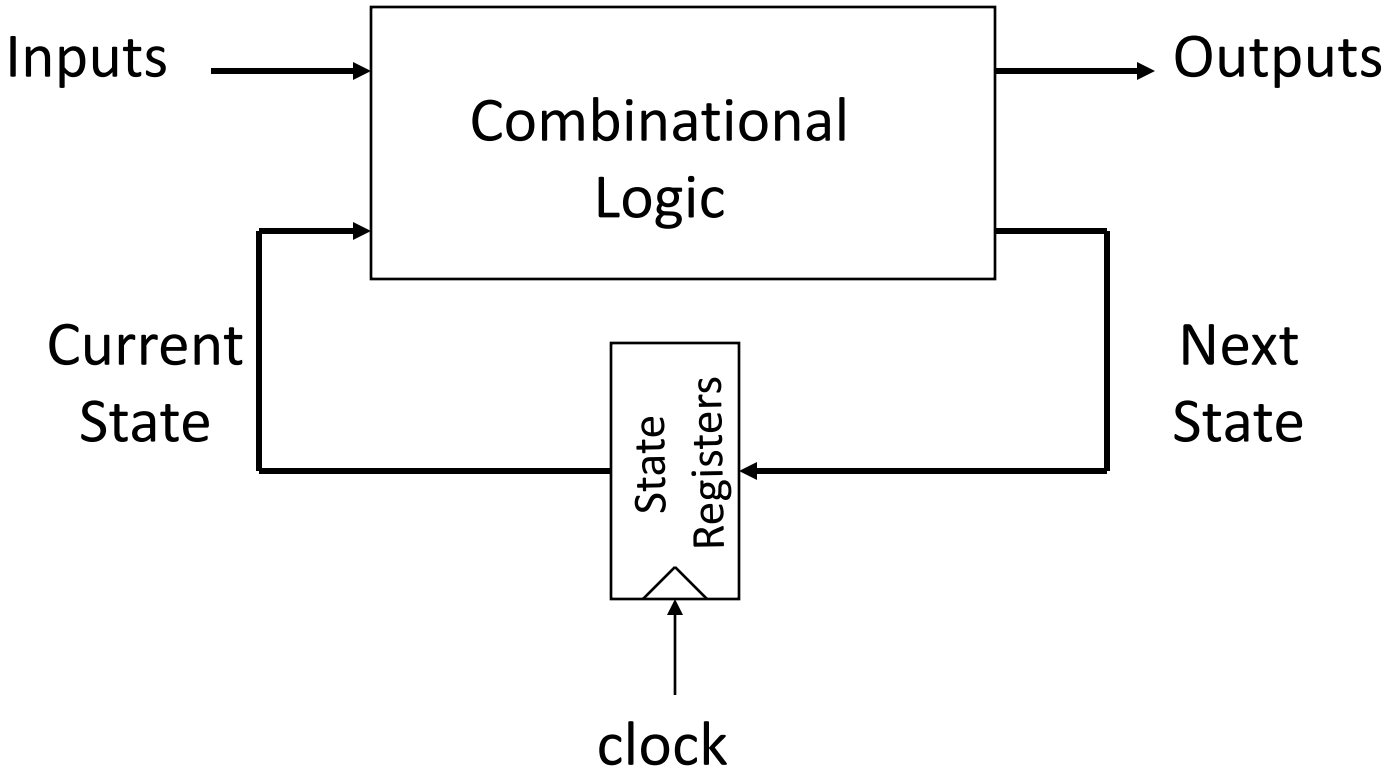




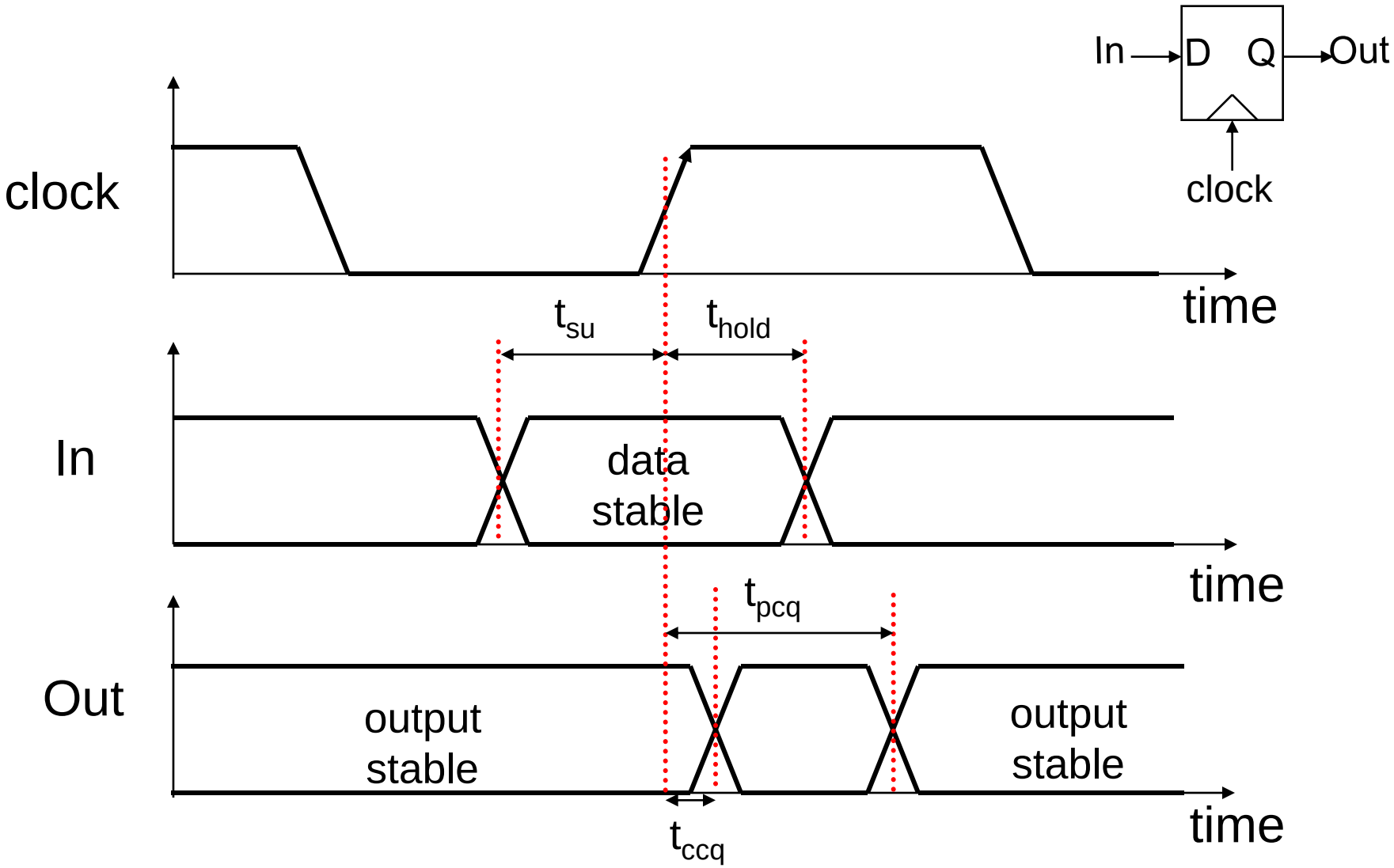
Outline

1. Sequencing
2. Sequencing Element Design
3. **Max and Min-Delay**
4. Time Borrowing
5. Clock Skew
6. Two-Phase Clocking

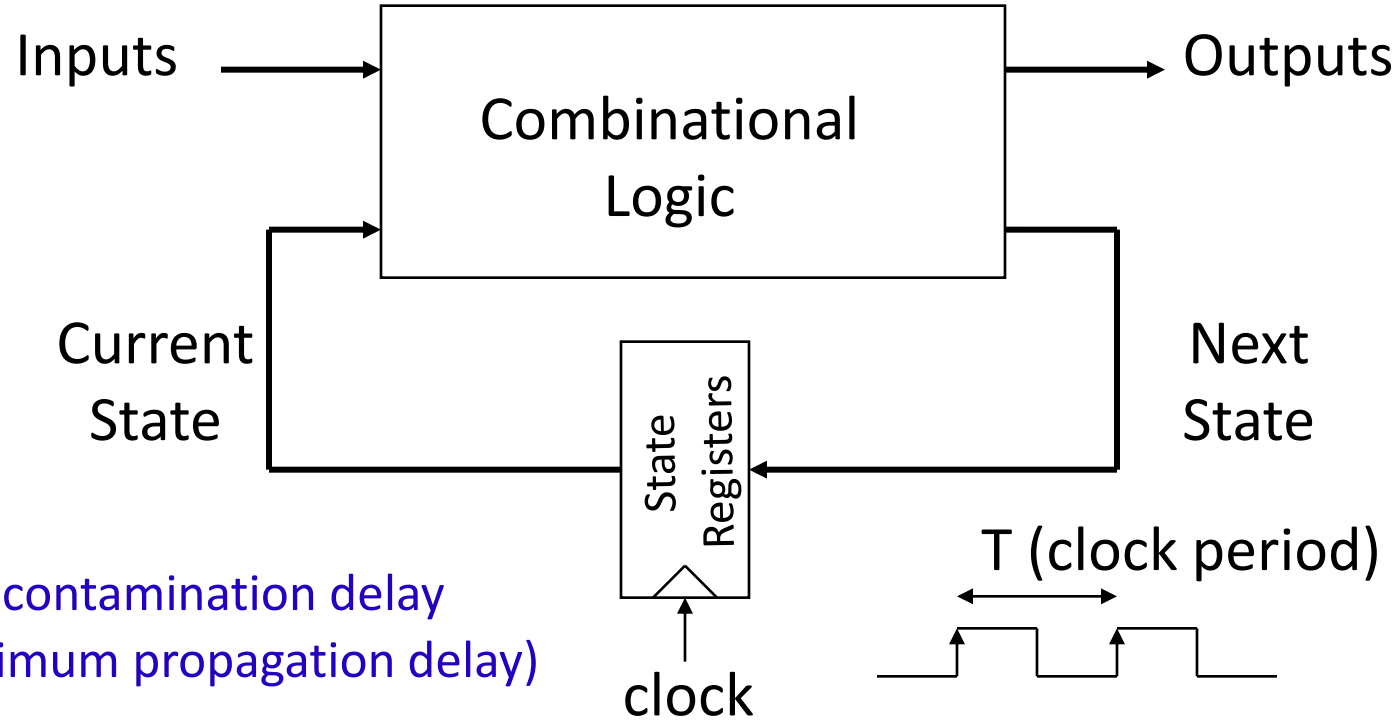
Sequential Logic



Timing Metrics



System Timing Constraints



t_{cd} : contamination delay
(minimum propagation delay)

$t_{ccq} + t_{cdlogic} \geq t_{hold}$

$T \geq t_{pcq} + t_{plogic} + t_{su}$

t_{ccq} : minimum delay of register

$t_{cdlogic}$: minimum delay of logic

t_{pcq} : maximum delay of register

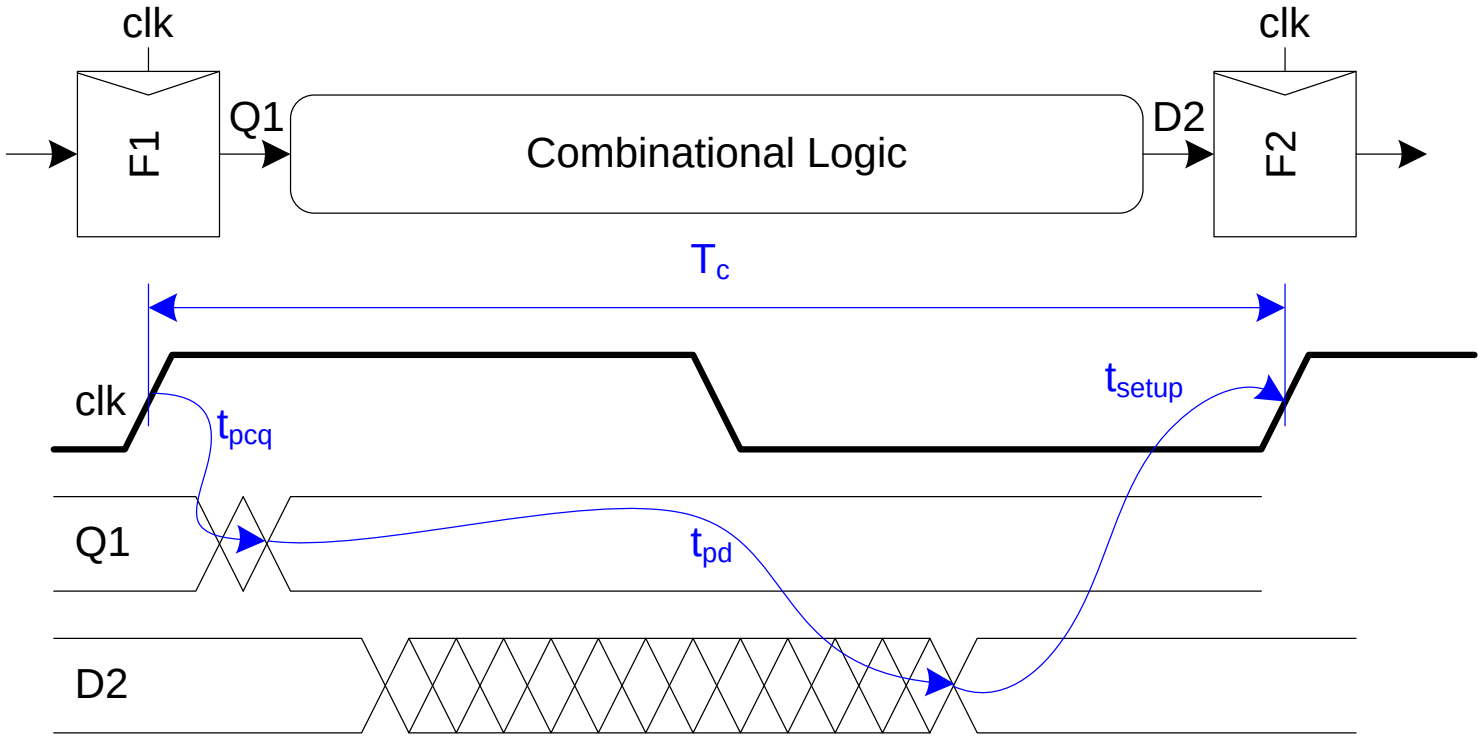
t_{plogic} : maximum delay of logic

Max/Min Delay Constraints

- Max-Delay Constraints
  - Combinational logic delay is too great, the receiving element will miss its setup time and sample the wrong value : ***setup time failure*** (***max-delay failure***)
  - It can be solved by redesign faster logic or by increasing the clock period.
- Min-Delay Constraints
  - If the hold time is large and the contamination delay is small, data can incorrectly propagate through on one clock edge and corrupt the state : ***race condition***, ***hold time failure***, or ***min-delay failure***.
  - Redesign slower logic.

Max-Delay: Flip-Flops

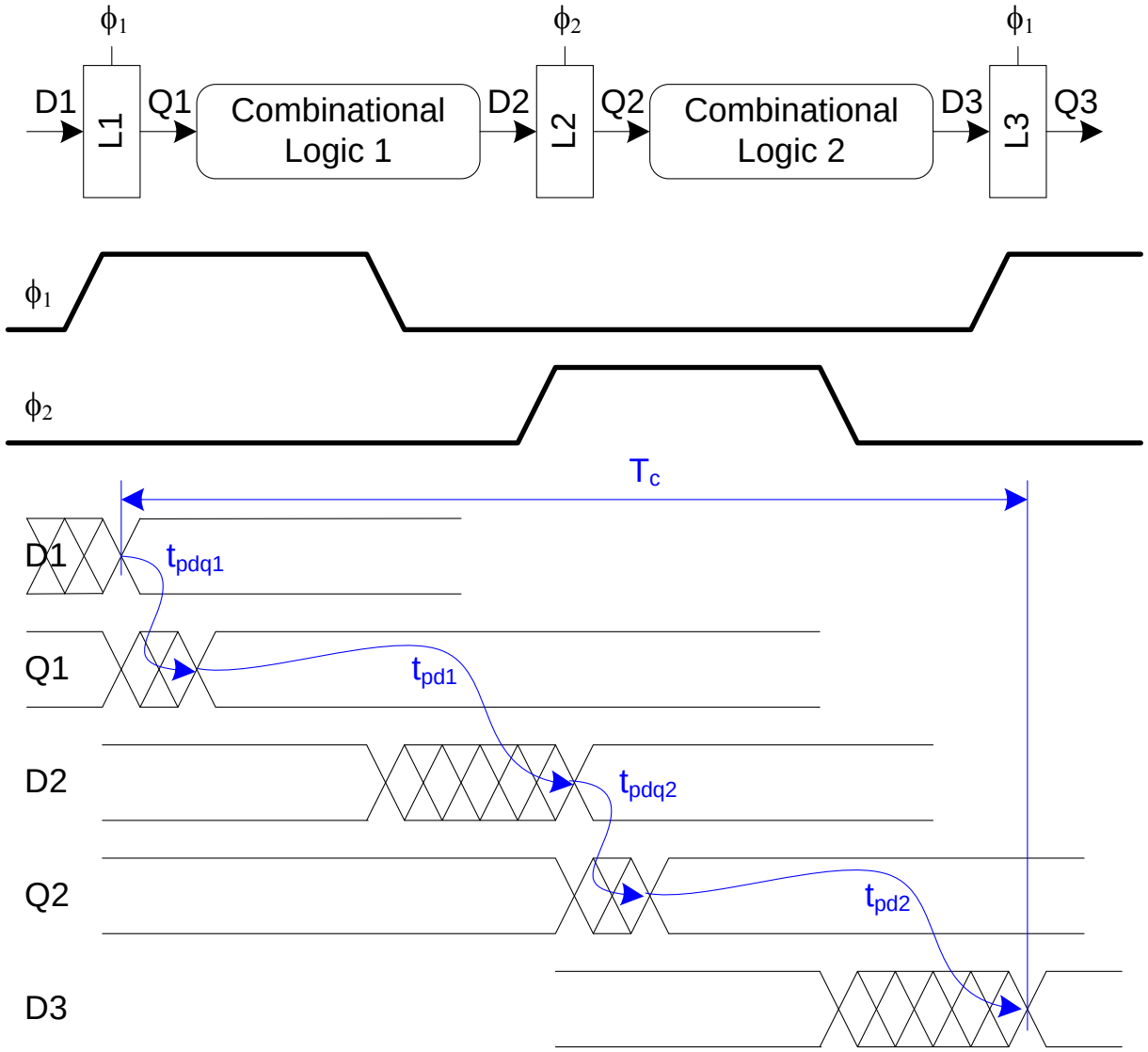
$$t_{pd} \leq T_c - \underbrace{\left(t_{\text{setup}} + t_{pcq}\right)}_{\text{sequencing overhead}}$$



Max Delay: 2-Phase Latches

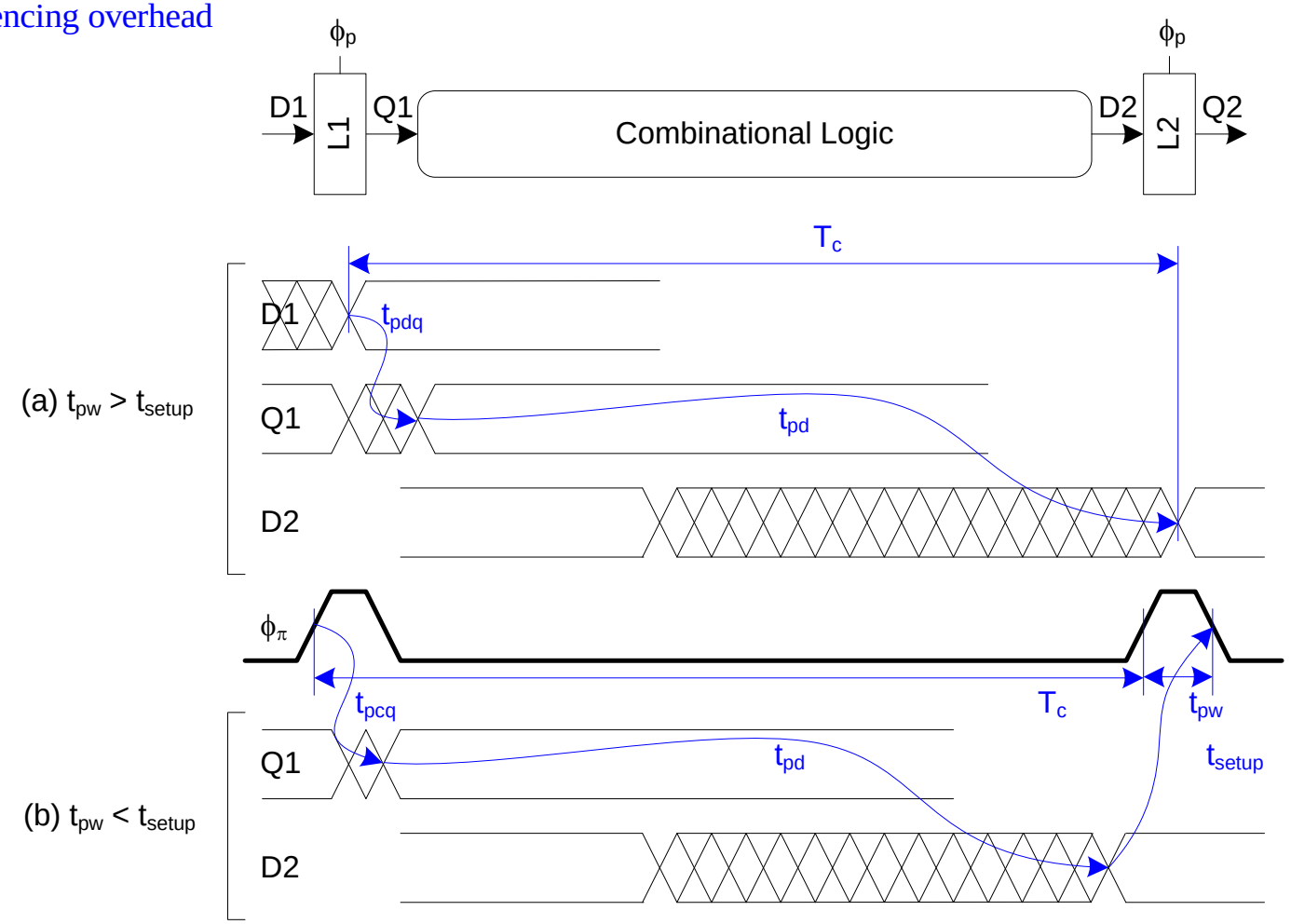
$$t_{pd} = t_{pd1} + t_{pd2}$$

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



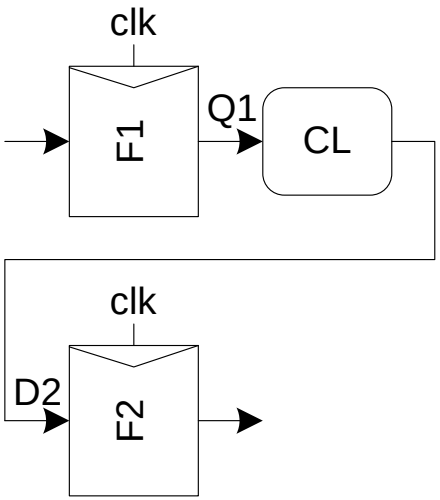
Max Delay: Pulsed Latches

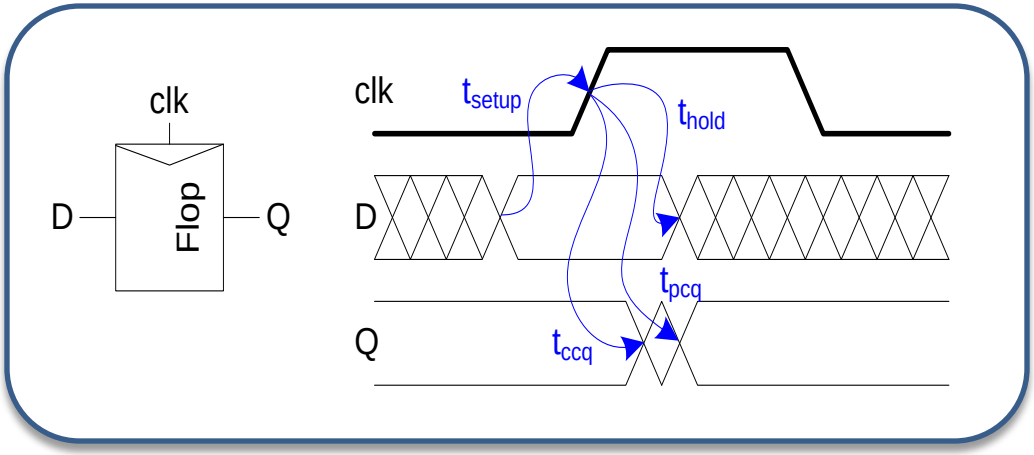
$$t_{pd} \le T_c - \underbrace{\max\left(t_{pdq}, t_{pcq} + t_{\text{setup}} - t_{pw}\right)}_{\text{sequencing overhead}}$$

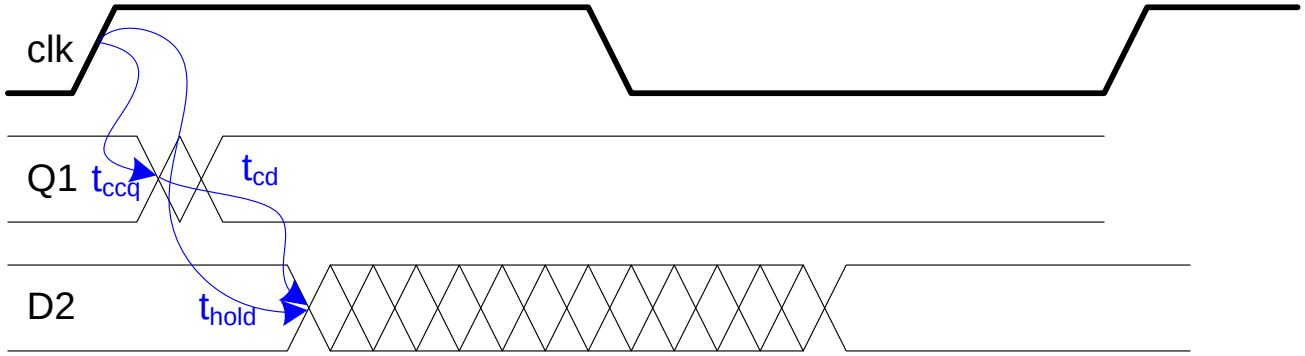


Min-Delay: Flip-Flops

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

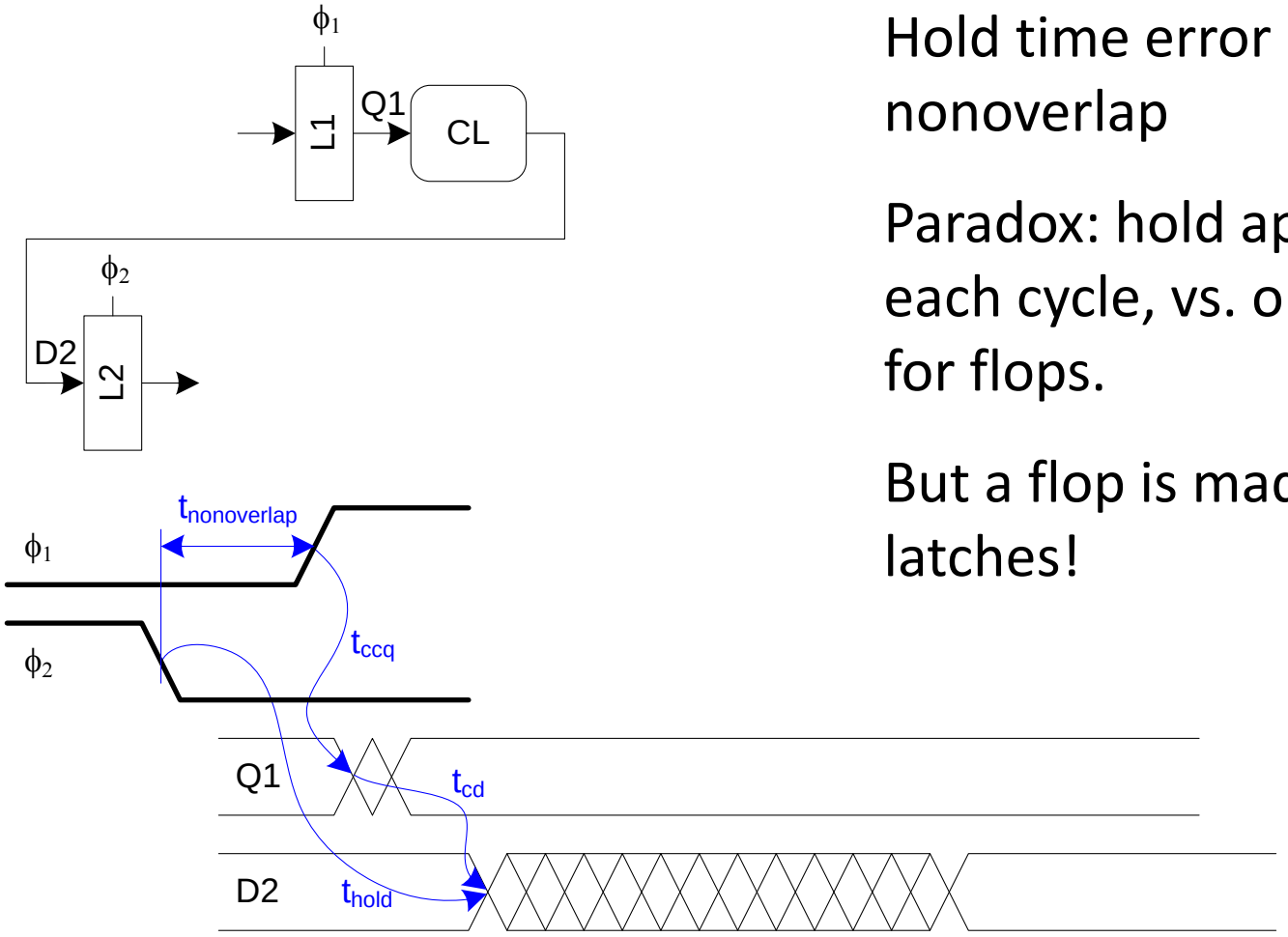






Min-Delay: 2-Phase Latches

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



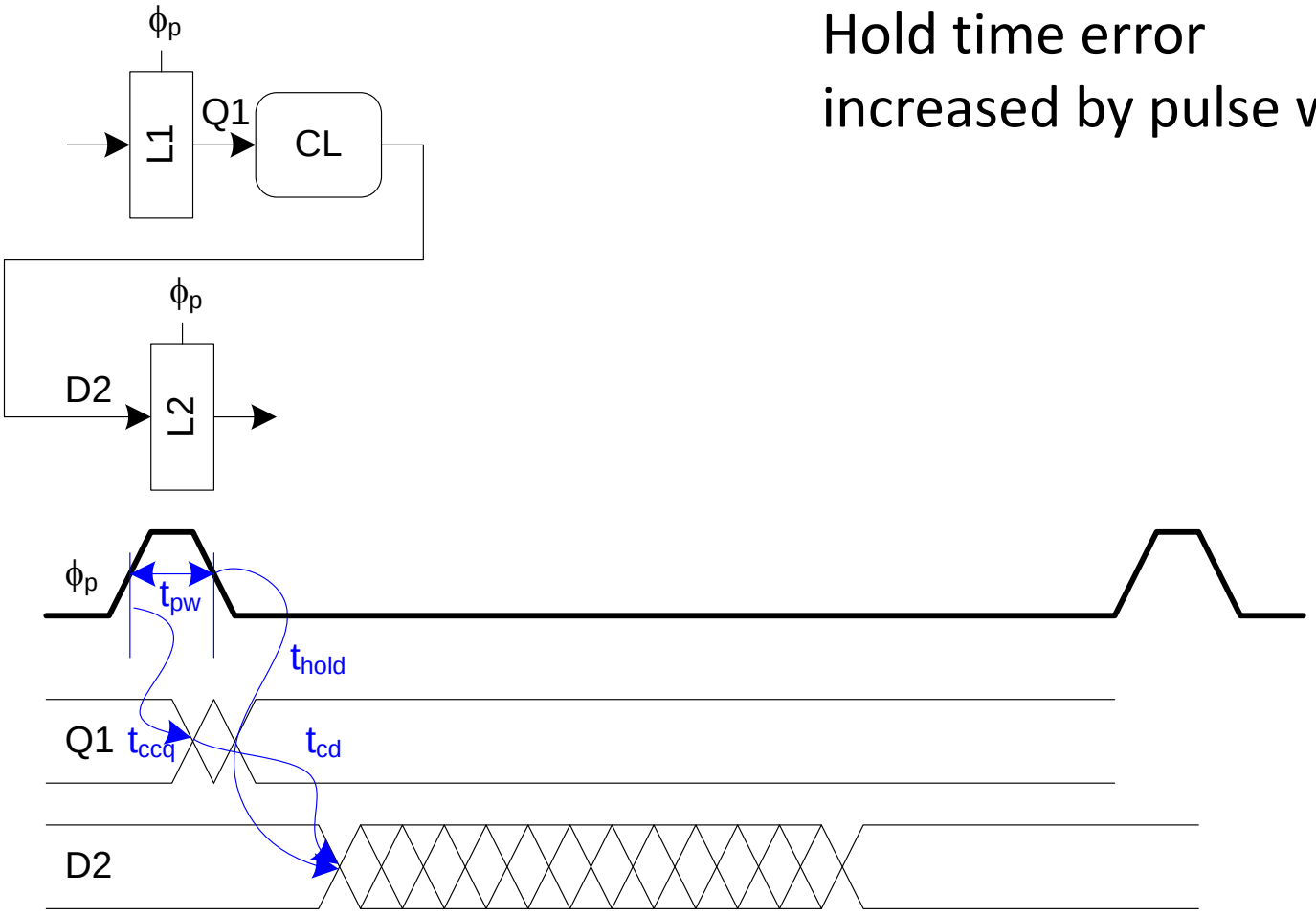
Hold time error reduced by nonoverlap

Paradox: hold applies twice each cycle, vs. only once for flops.

But a flop is made of two latches!

Min-Delay: Pulsed Latches

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



Hold time error increased by pulse width

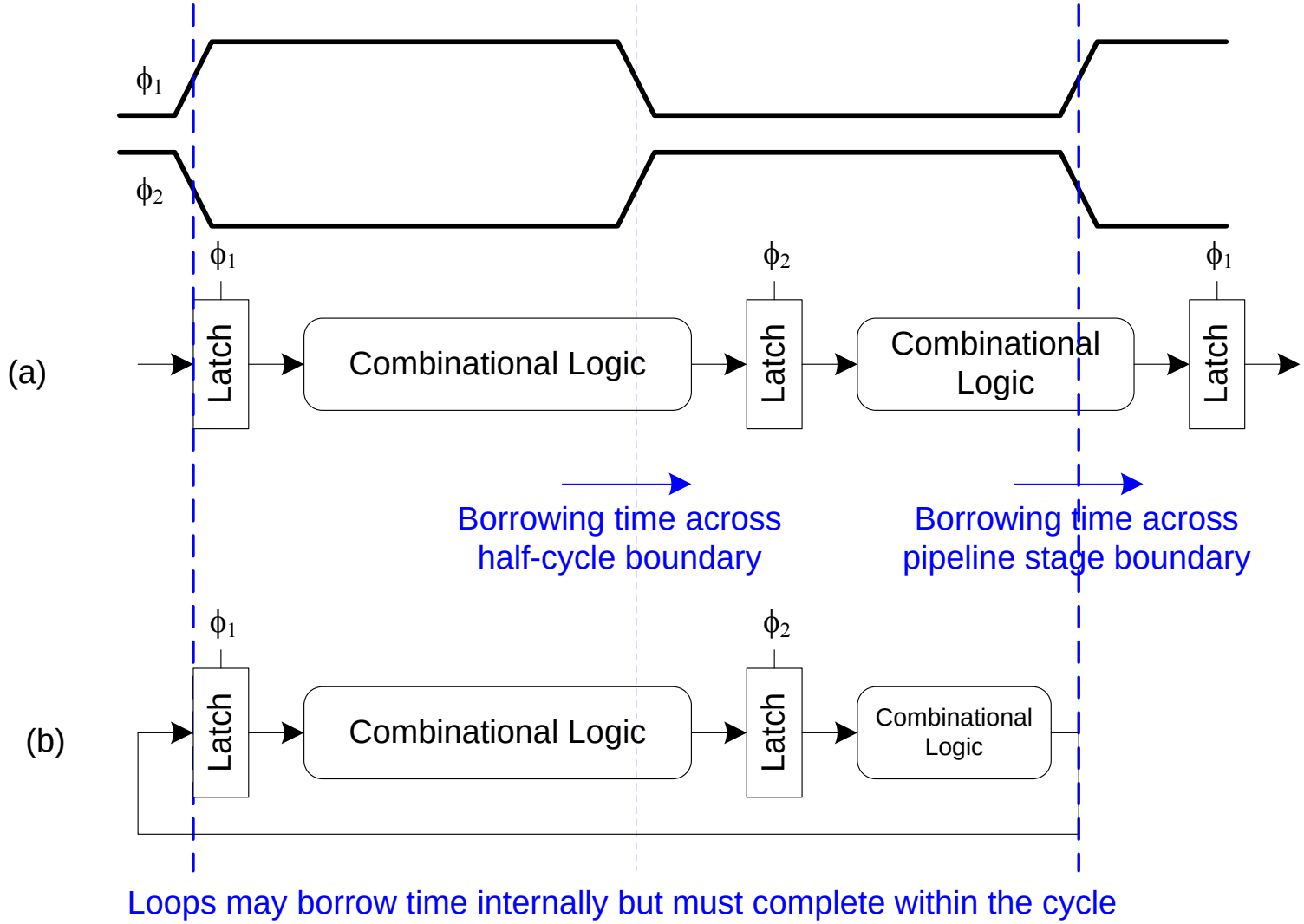
Outline

1. Sequencing
2. Sequencing Element Design
3. Max and Min-Delay
4. **Time Borrowing**
5. Clock Skew
6. Two-Phase Clocking

Time Borrowing

- In a flop-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
  - Flops 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



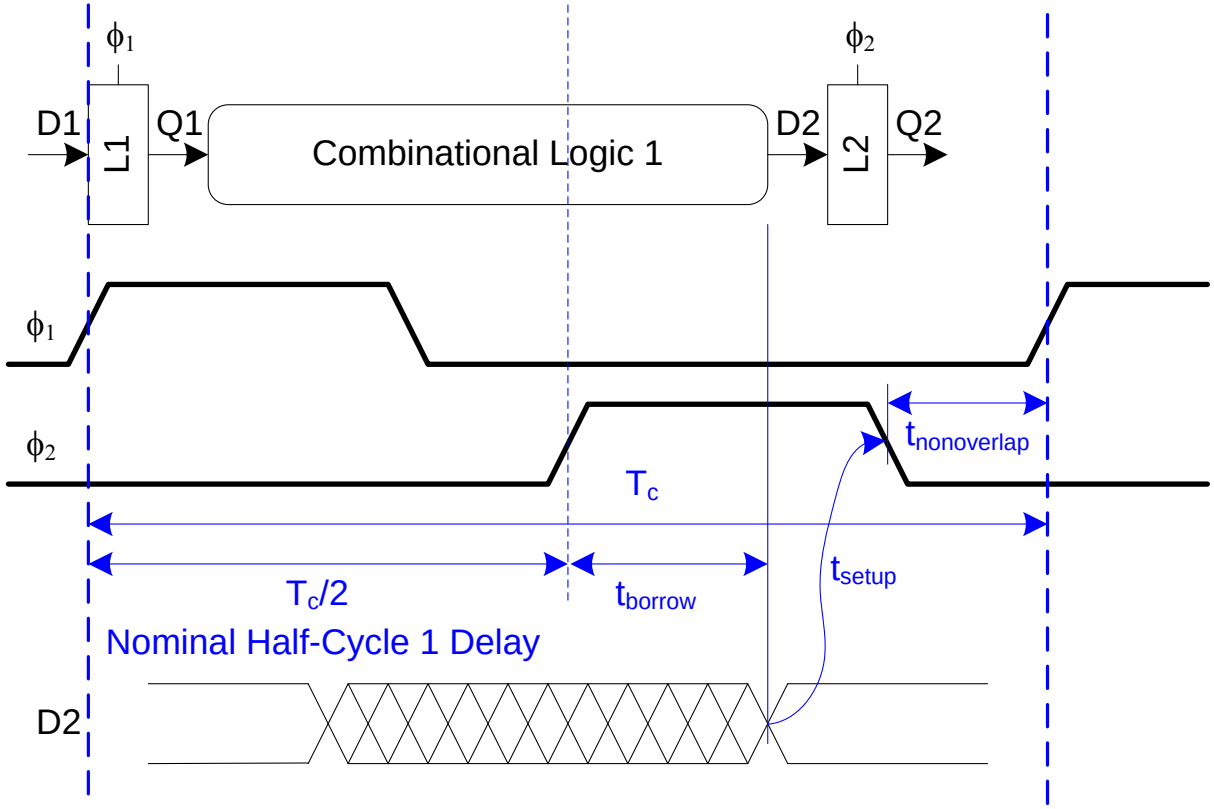
How Much Borrowing?

2-Phase Latches

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

Pulsed Latches

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



Outline

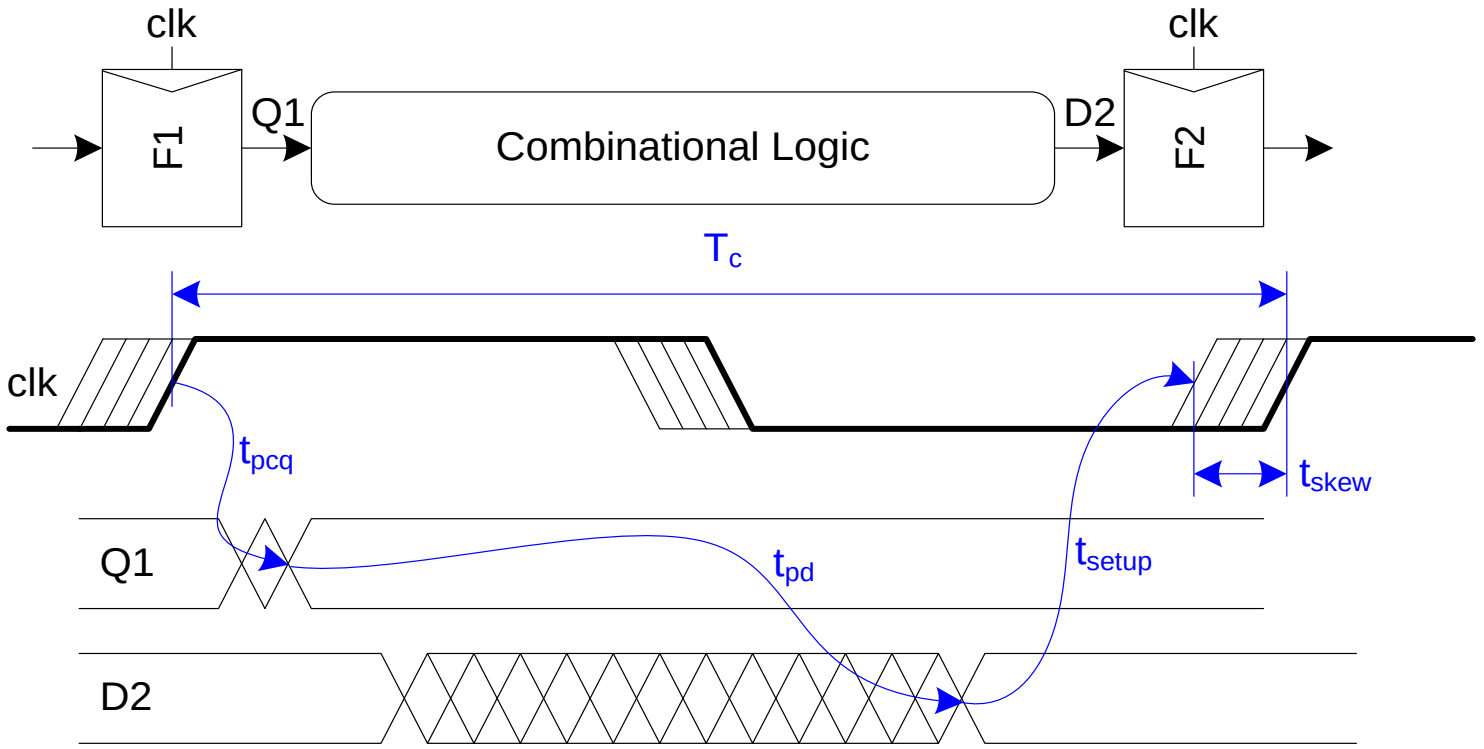
1. Sequencing
2. Sequencing Element Design
3. Max and Min-Delay
4. Time Borrowing
5. **Clock Skew**
6. Two-Phase Clocking

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

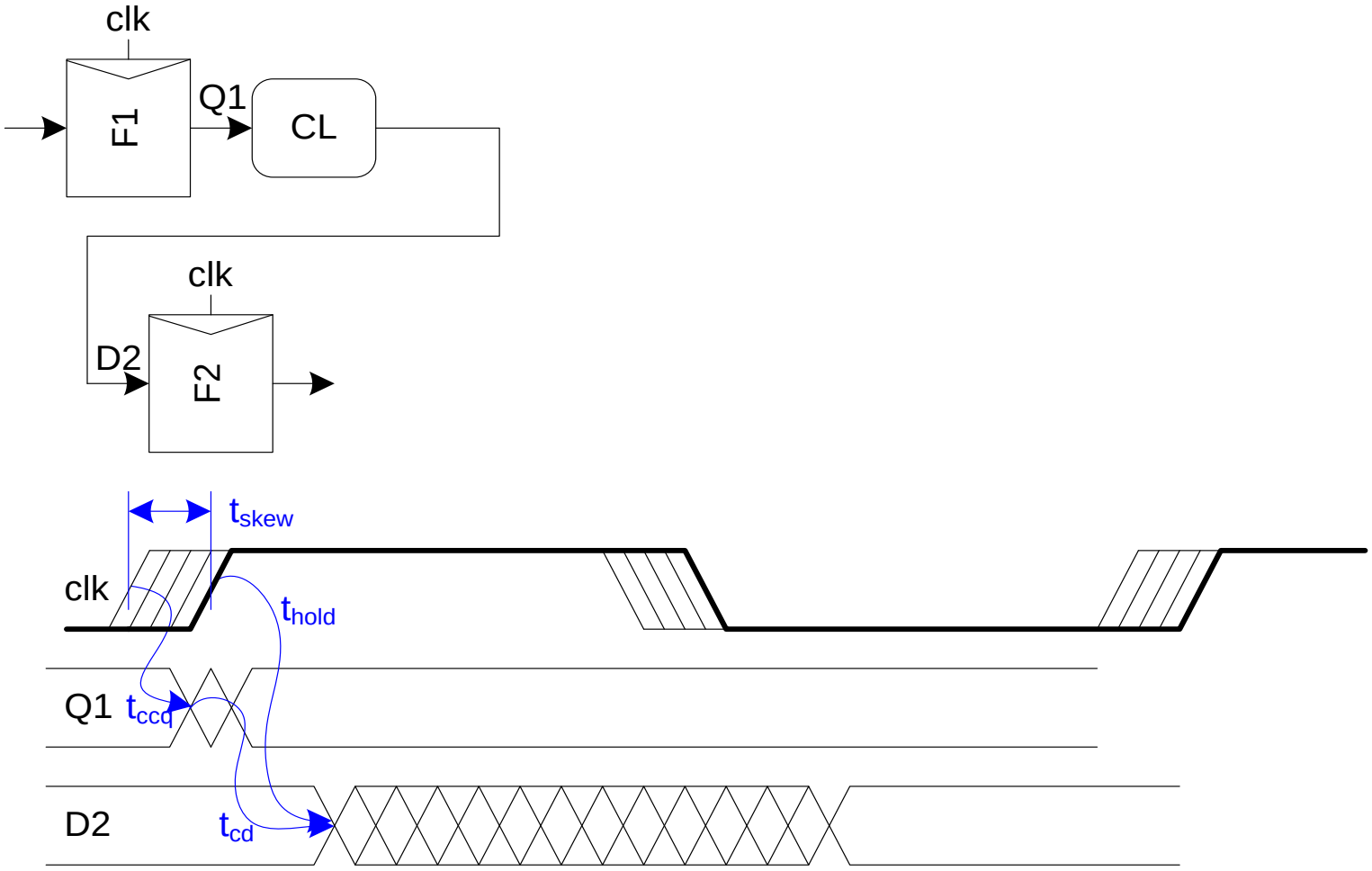
Skew: Flip-Flops: Max Delay

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



Skew: Flip-Flops: Min Delay

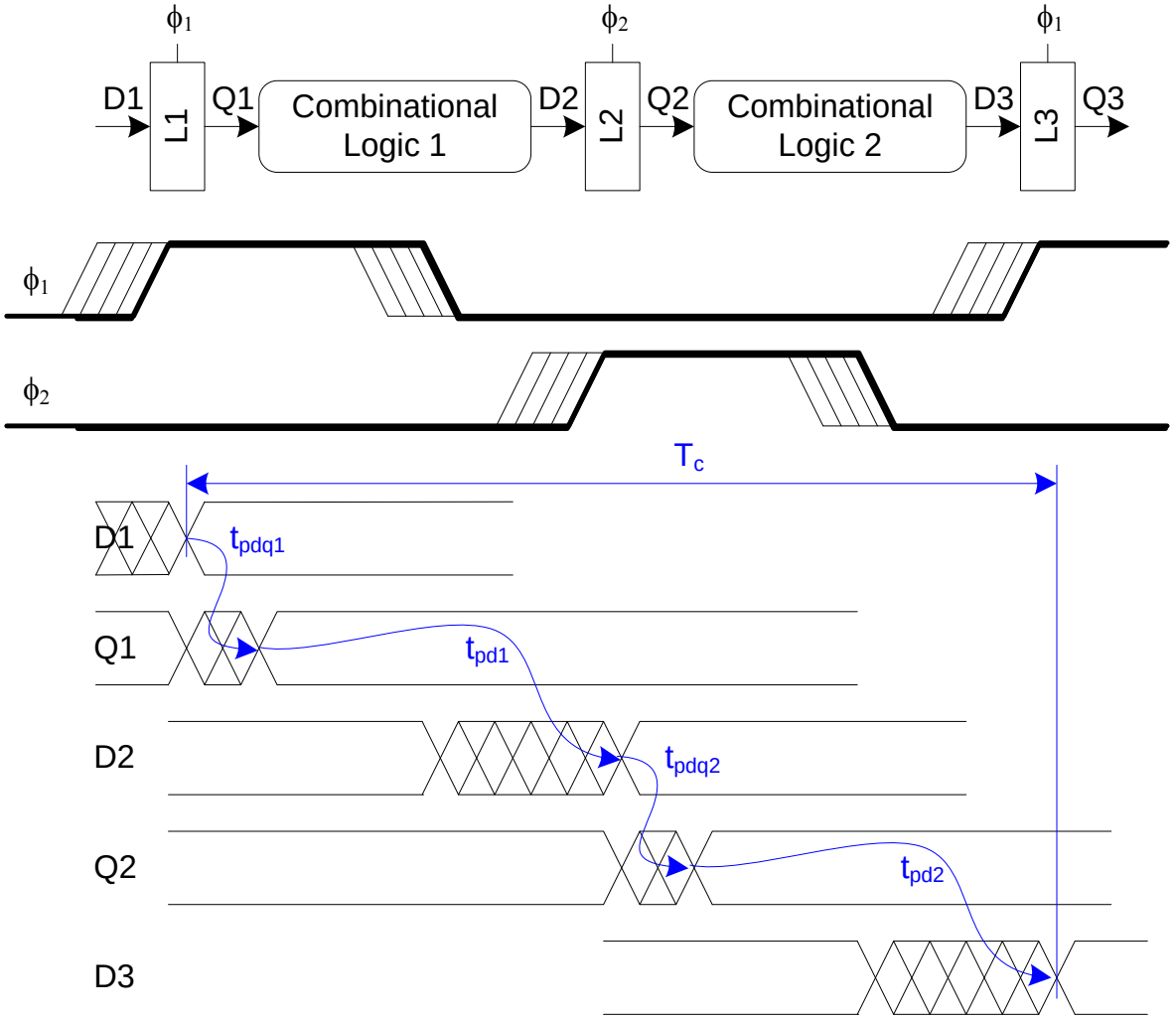
$$t_{cd} \geq t_{\text{hold}} - t_{ccq} + t_{\text{skew}}$$



Skew: 2-Phase Latches: Max Delay

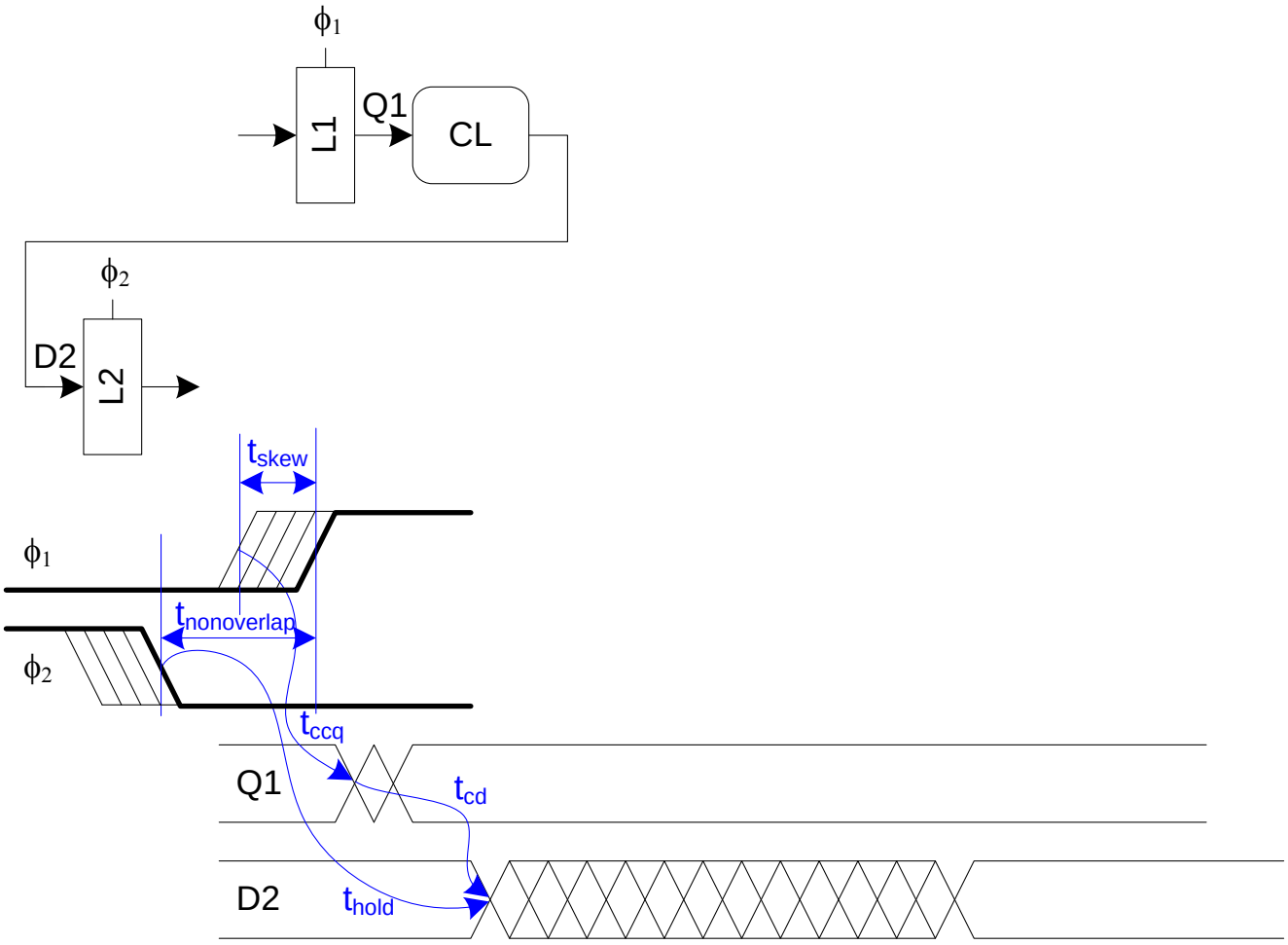
$$t_{pd} \leq T_c - \underbrace{(2t_{pdq})}_{\text{sequencing overhead}}$$

No change ➔ Latch-based systems are ***skew-tolerant***



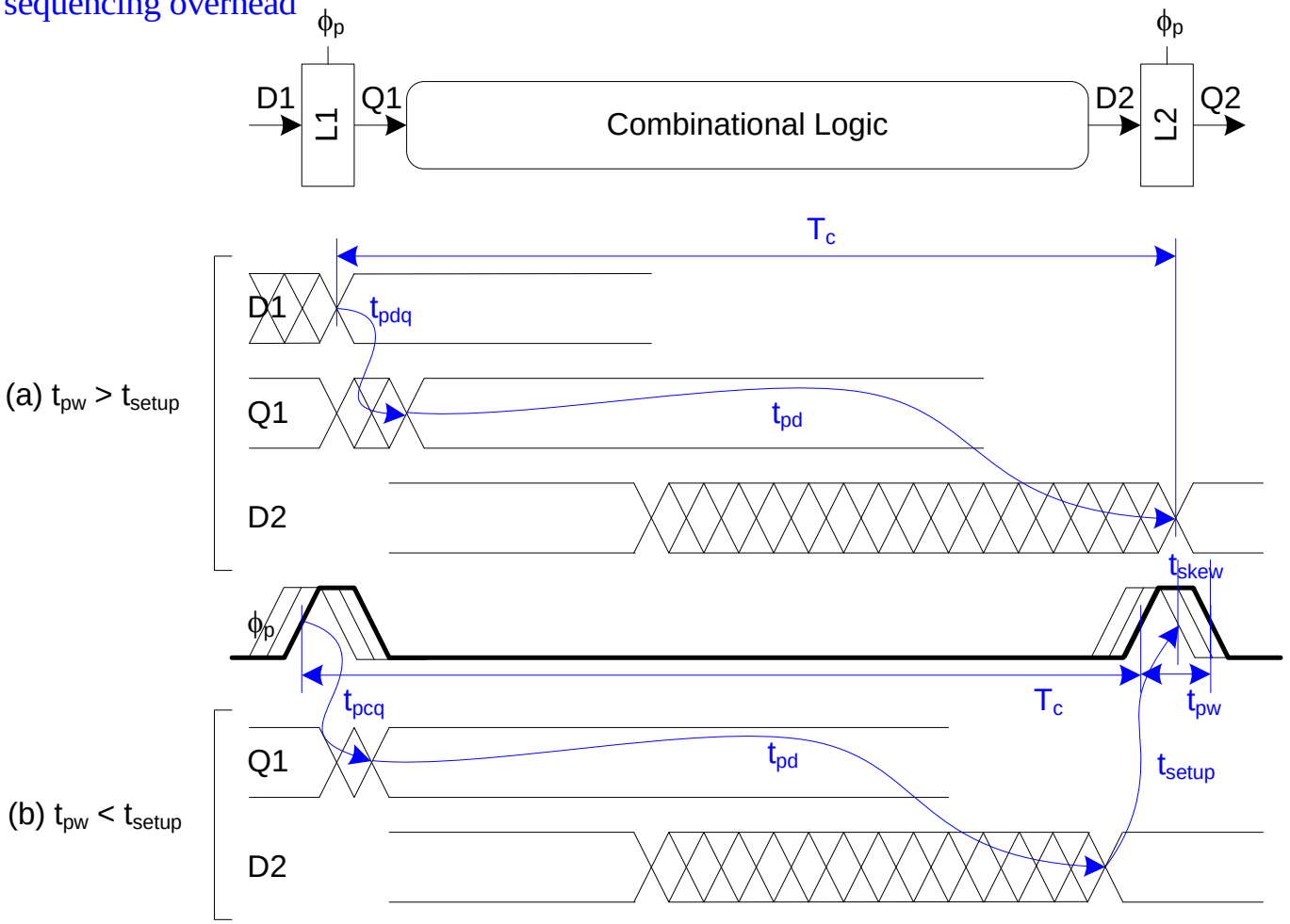
Skew: 2-Phase Latches: Min Delay

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



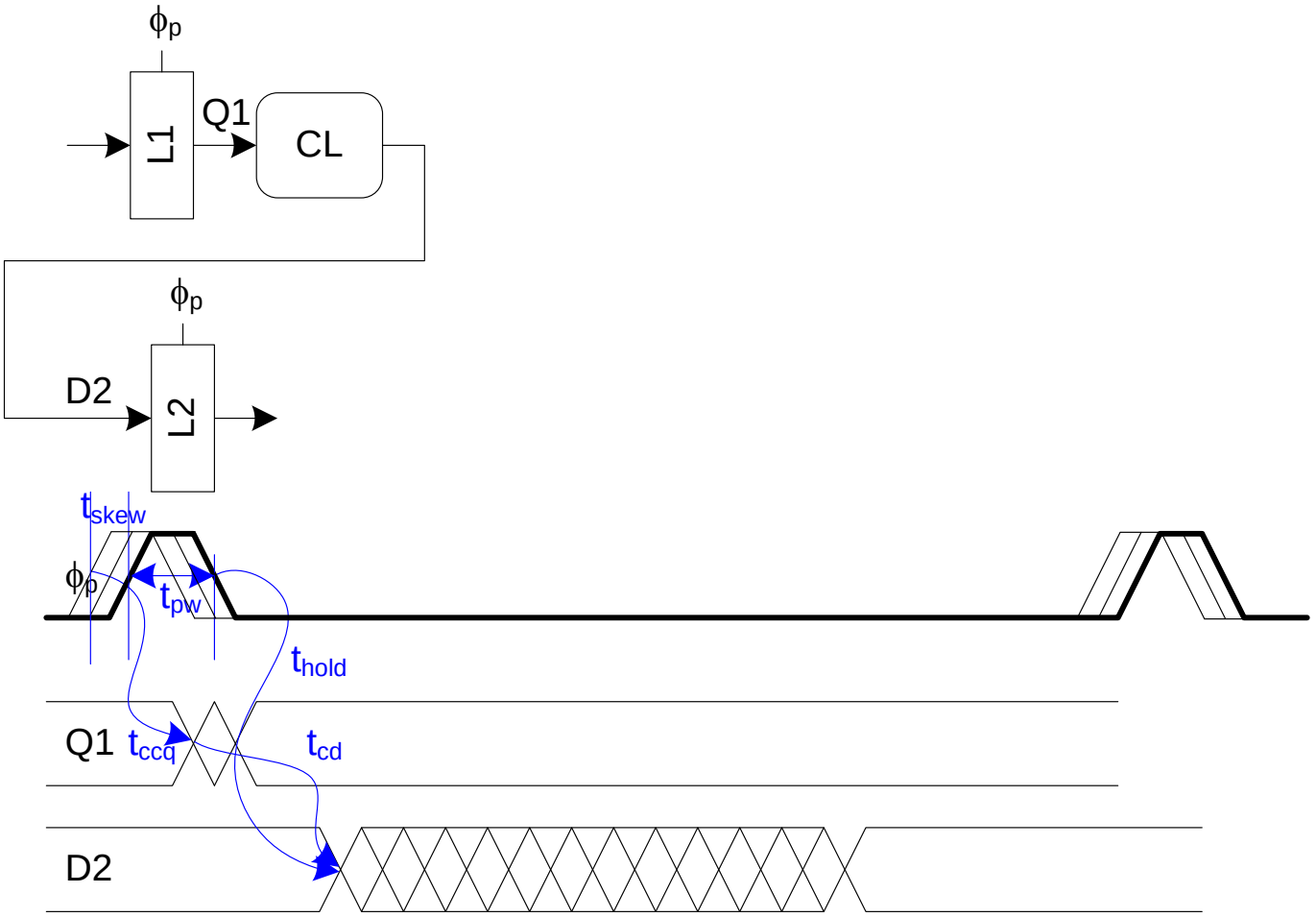
Skew: Pulsed Latches: Max Delay

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



Skew: Pulsed Latches: Min Delay

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



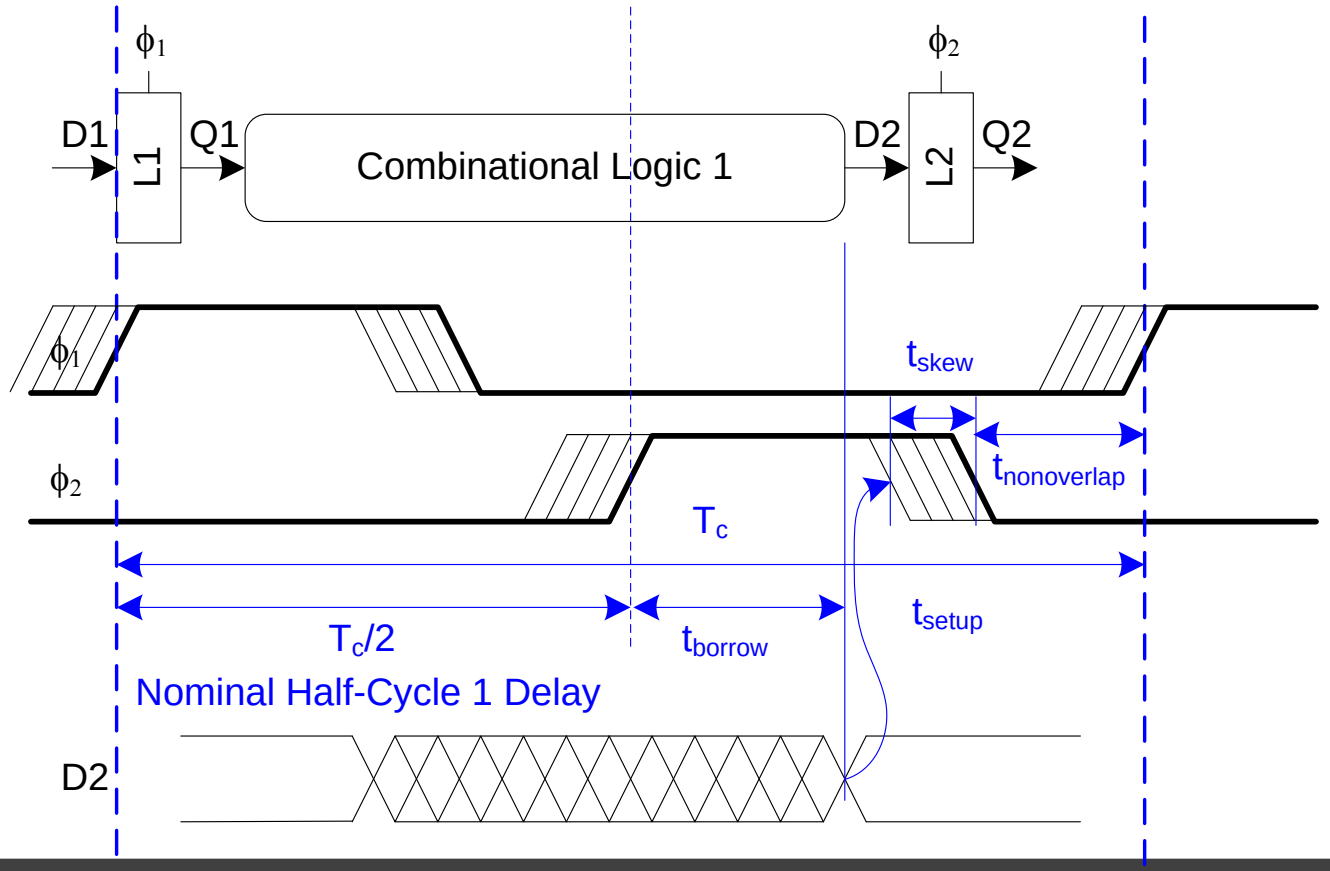
Skew: How Much Borrowing ?

2-Phase Latches

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

Pulsed Latches

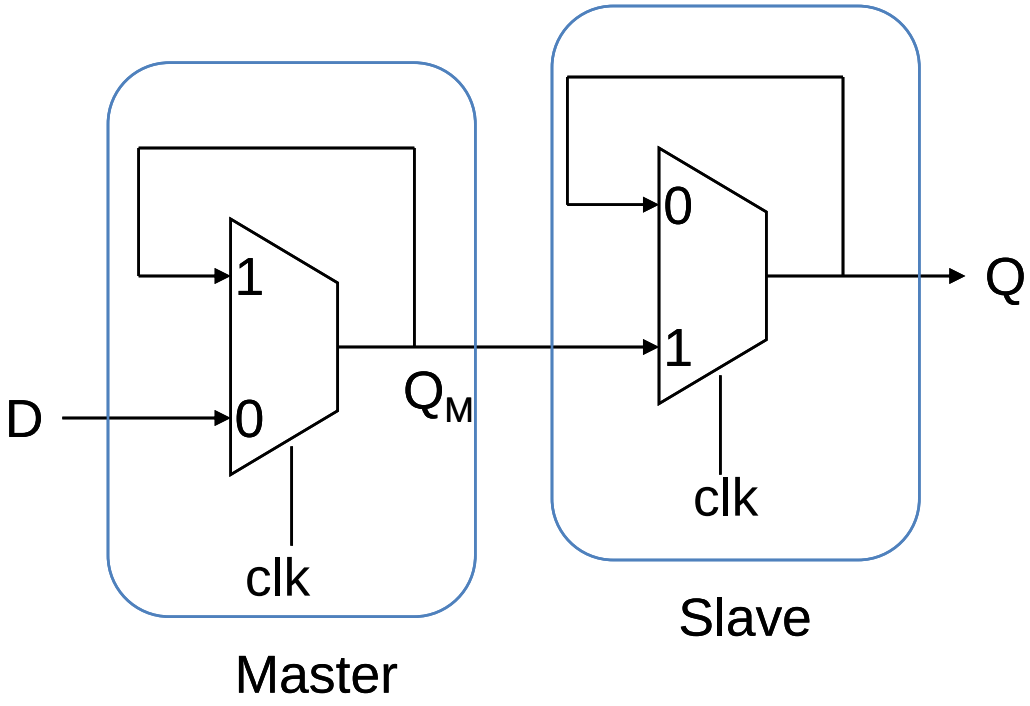
$$t_{\text{borrow}} \le t_{pw} - \left(t_{\text{setup}} + t_{\text{skew}} \right)$$

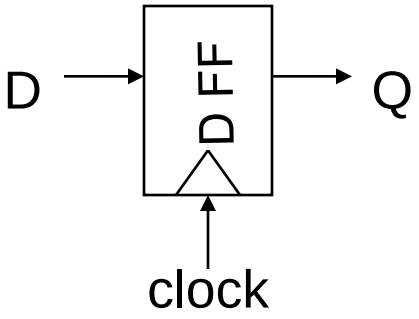


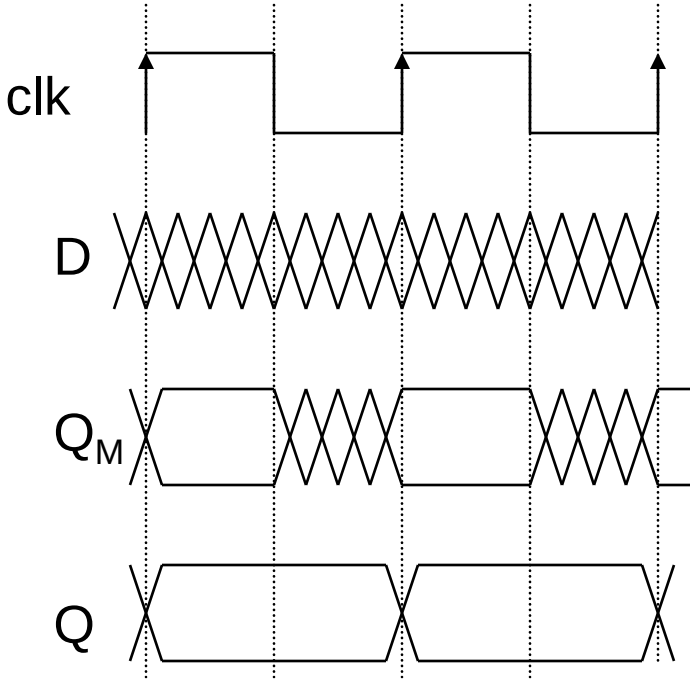
Outline

1. Sequencing
2. Sequencing Element Design
3. Max and Min-Delay
4. Time Borrowing
5. Clock Skew
6. **Two-Phase Clocking**

Master Slave Based ET Flipflop



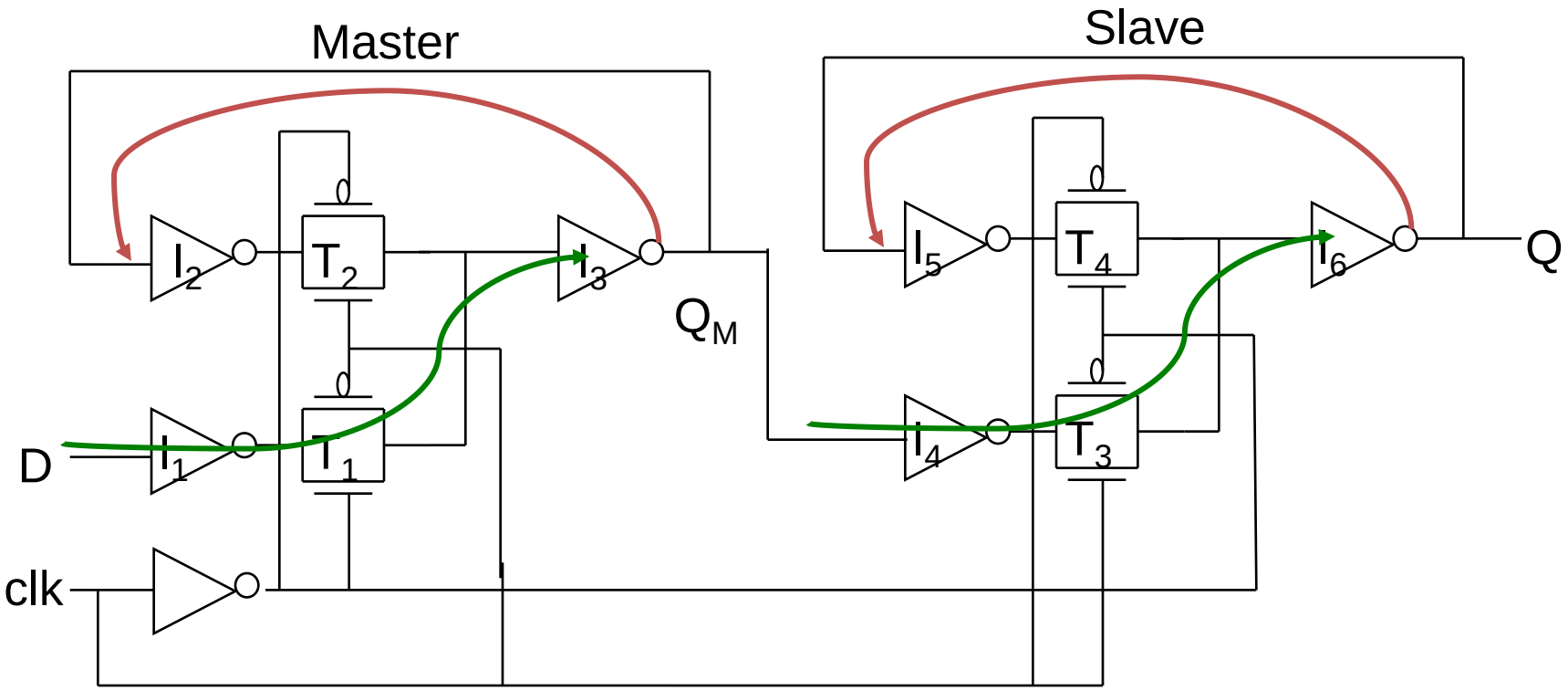




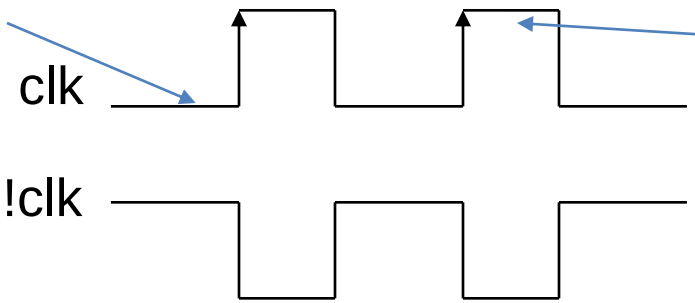
clk = 0 transparent hold

clk = 0→1 hold transparent

MS ET Implementation



master transparent
slave hold



master hold
slave transparent

MS ET Timing Properties

- Assume propagation delays are t_{pd_inv} and t_{pd_tx}, that the contamination delay is 0, and that the inverter delay to derive !clk is 0
- Set-up time - time before rising edge of clk that D must be valid

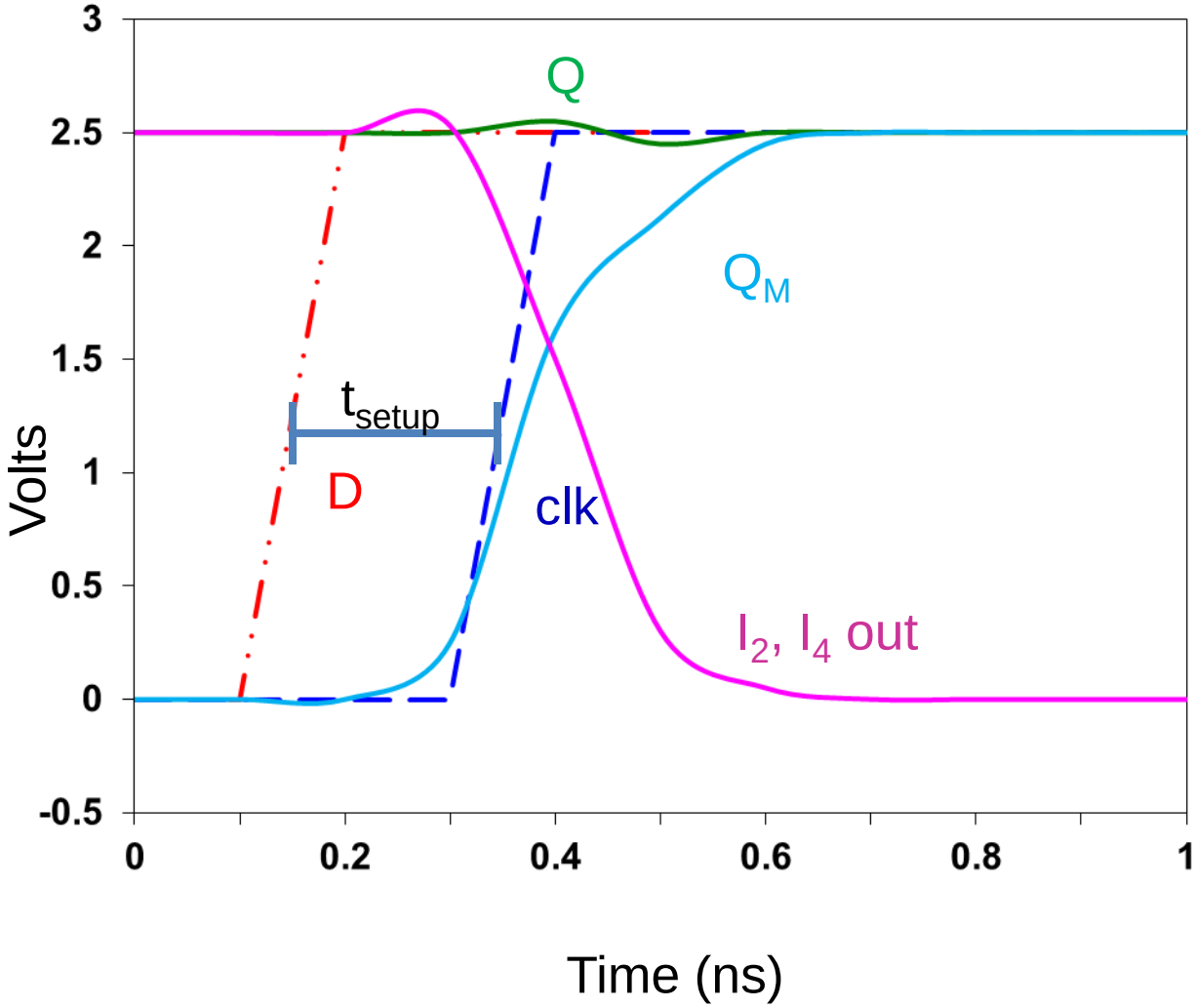
$$t_{su} = 3 * t_{pd\_inv} + t_{pd\_tx}$$

- Propagation delay - time for data to reach Q

$$t_{c\text{-}q} = t_{pd\_inv} + t_{pd\_tx}$$

- Hold time - time D must be stable after rising edge of clk - t_{hold} = zero

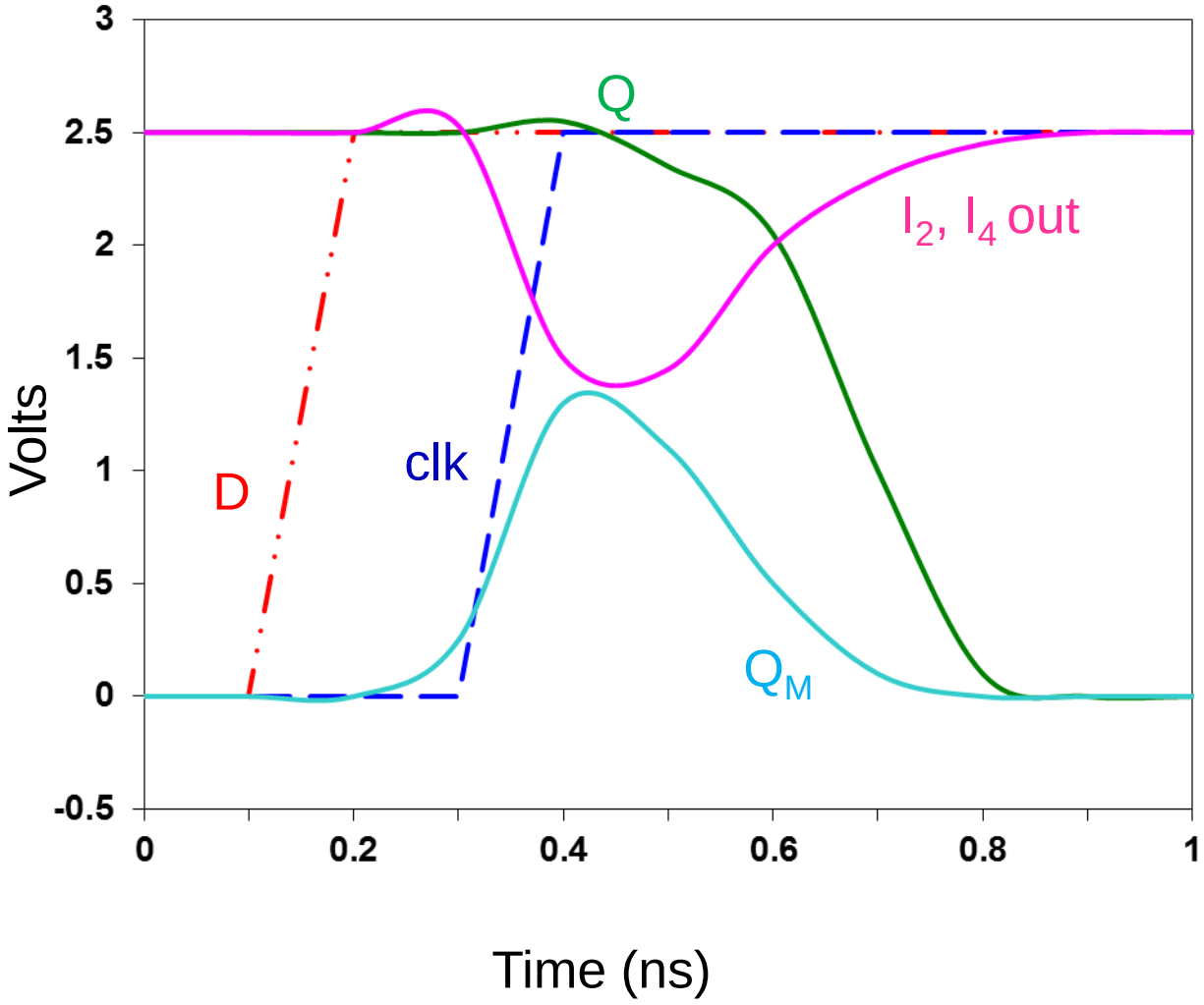
Set-up Time Simulation



t_{setup} = 0.21 ns

works correctly

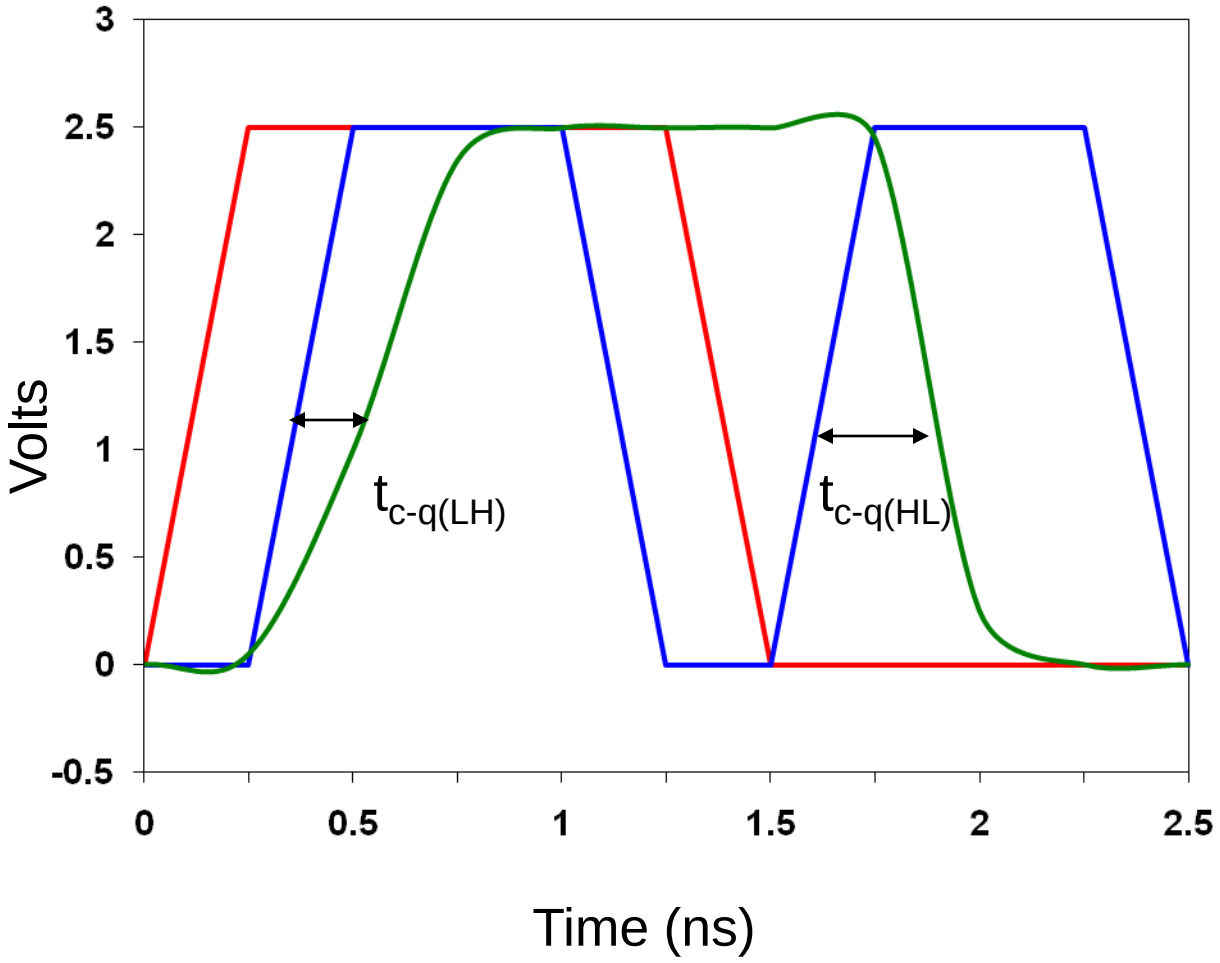
Set-up Time Simulation



t_{setup} = 0.20 ns

fails

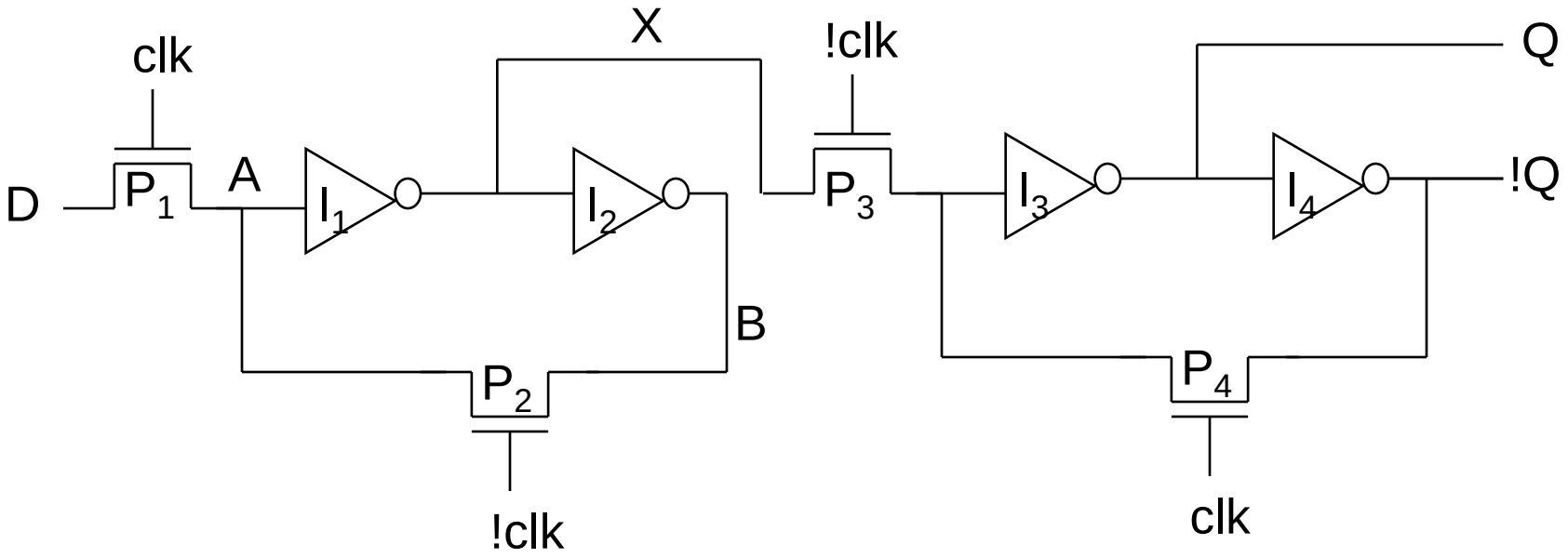
Propagation Delay Simulation



$t_{c\text{-}q(LH)}$ = 160 psec

$t_{c\text{-}q(HL)}$ = 180 psec

Example of Clock Skew Problems



Race condition – direct path from D to Q during the short time when both clk and !clk are high (1-1 overlap)

Undefined state – both B and D are driving A when clk and !clk are both high

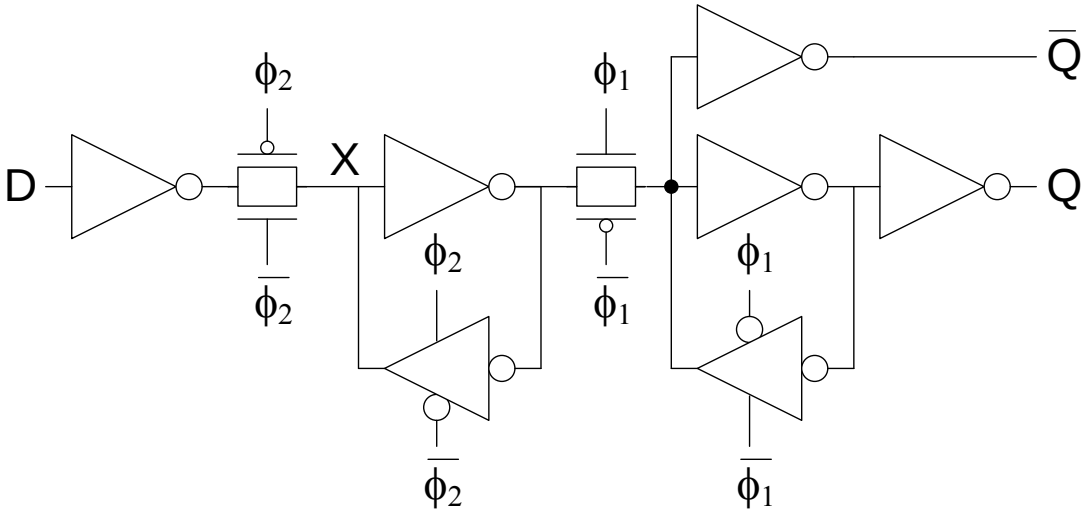
Dynamic storage – when clk and !clk are both low (0-0 overlap)

Two-Phase Clocking

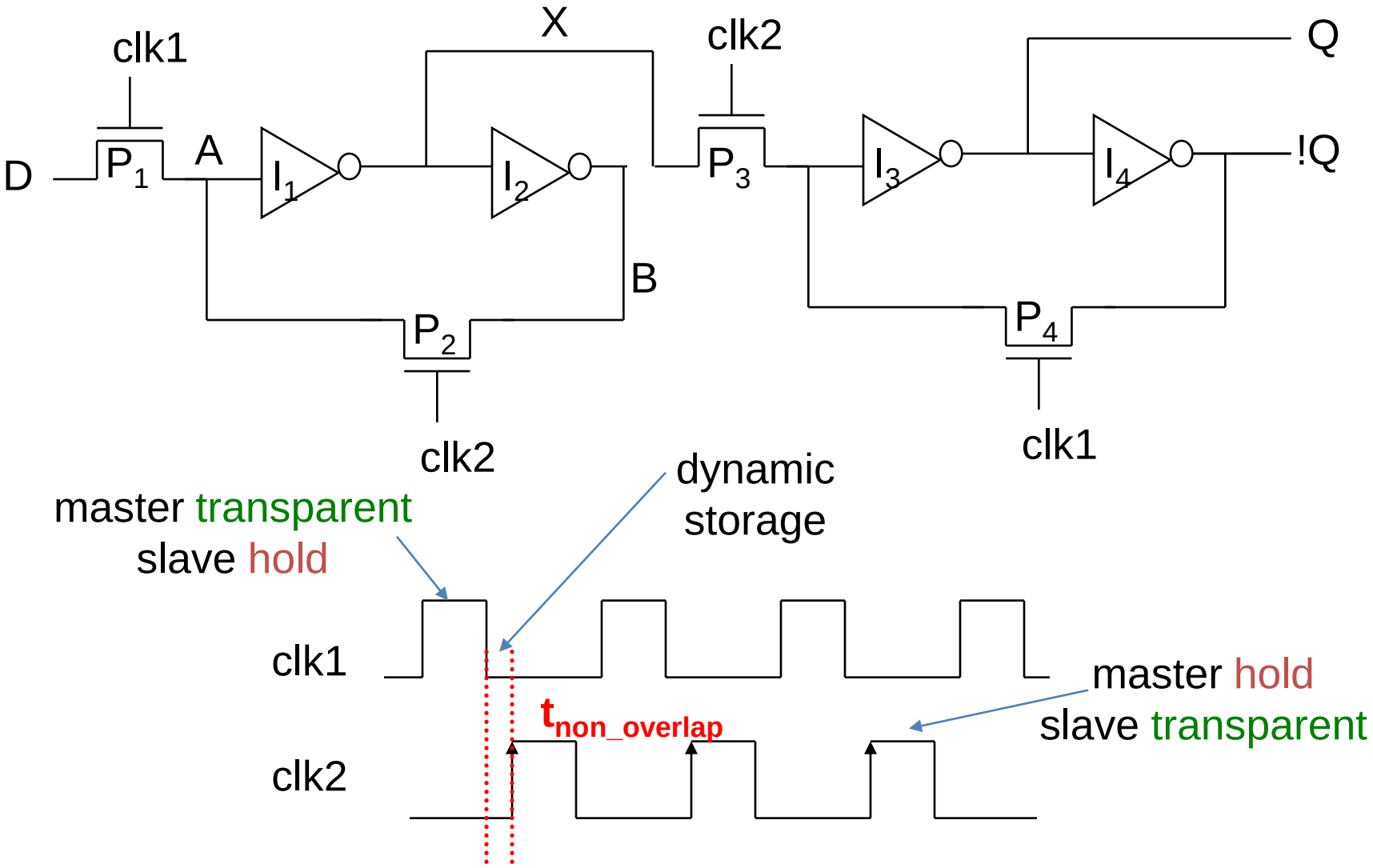
- If setup times are violated, reduce clock speed
- If hold times are violated, chip fails at any speed
- In this class, working chips are most important
  - No tools to analyze clock skew
- An easy way to guarantee hold times is to use 2-phase latches with big nonoverlap times
- Call these clocks ϕ_1,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

Safe Flip-Flop

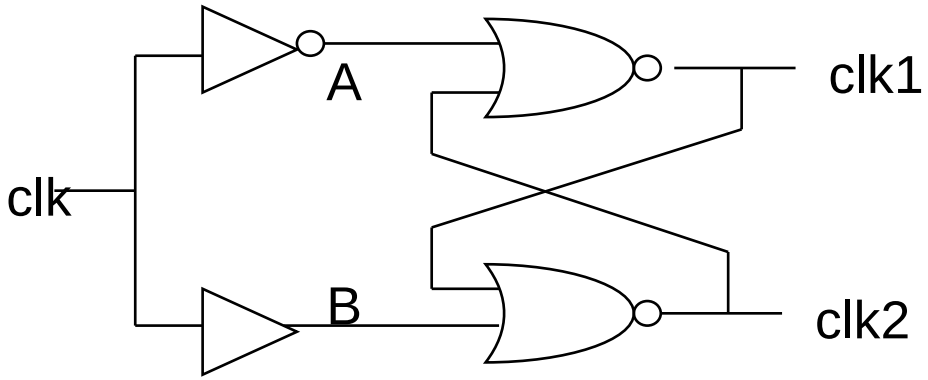
- In class, use flip-flop with nonoverlapping clocks
  - Very slow – nonoverlap adds to setup time
  - But no hold times
- In industry, use a better timing analyzer
  - Add buffers to slow signals if hold time is at risk

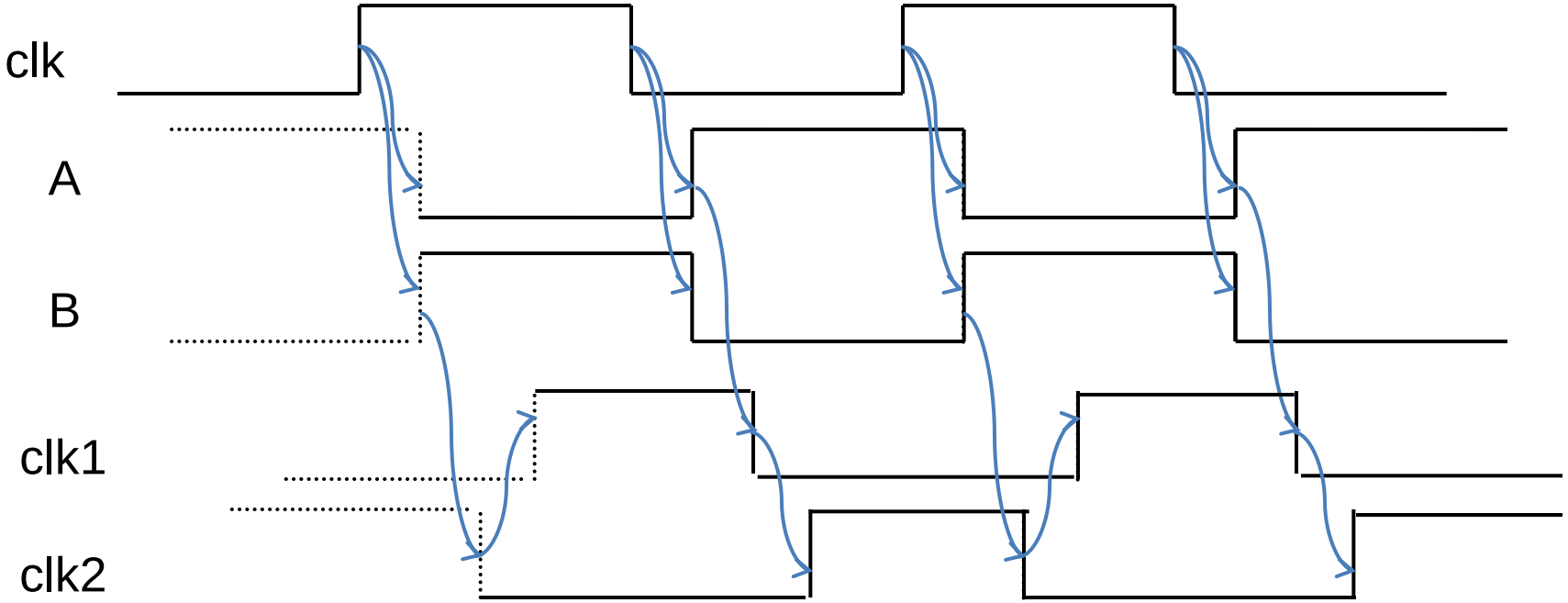


Pseudostatic Two-Phase ET FF



Two Phase Clock Generator





Summary

- Flip-Flops:
  - Very easy to use, greatest sequencing overhead
- 2-Phase Transparent Latches:
  - Lots of skew tolerance and time borrowing, great design effort to partition logic to half-cycles.
- Pulsed Latches:
  - Fast, some skew tol & borrow, hold time risk

Table 7.4 Comparison of sequencing elements

	Sequencing overhead ($T_c - t_{pd}$)	Minimum logic delay t_{cd}	Time borrowing t_{borrow}
Flip-Flops	$t_{pcq} + t_{setup} + t_{skew}$	$t_{hold} - t_{ccq} + t_{skew}$	0
Two-Phase Transparent Latches	$2t_{pdq}$	$t_{hold} - t_{ccq} - t_{nonoverlap} + t_{skew}$ in each half-cycle	$\frac{T_c}{2} - (t_{setup} + t_{nonoverlap} + t_{skew})$
Pulsed Latches	$\max(t_{pdq}, t_{pcq} + t_{setup} - t_{pw} + t_{skew})$	$t_{hold} - t_{ccq} + t_{pw} + t_{skew}$	

Static Sequencing Element Methodology

- Choice of elements
  - Flip-flops have fairly high sequencing overheads but are popular because they are so simple.
  - Transparent latches have lower sequencing overheads and are attractive because of time borrowing (nearly half cycle)
- Low power sequential design
  - Keep device sizes small inside the core latch and minimize the number of clocked transistors
  - Clock gating

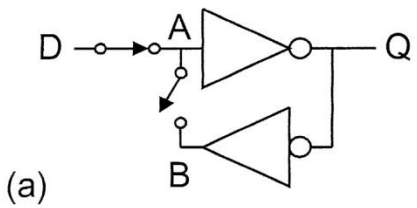
Outline

1. Sequencing
2. Sequencing Element Design
3. Max and Min-Delay
4. Time Borrowing
5. Clock Skew
6. Two-Phase Clocking
7. **Synchronizers**

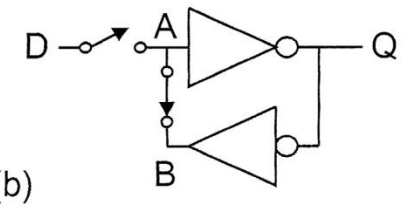
Synchronizers

- Data input should change outside the **aperture** between the setup time and hold time.
- What if cannot?
- Synchronizer
  - Accepts an input that can change at arbitrary times and produces an output aligned to the synchronizer's clock

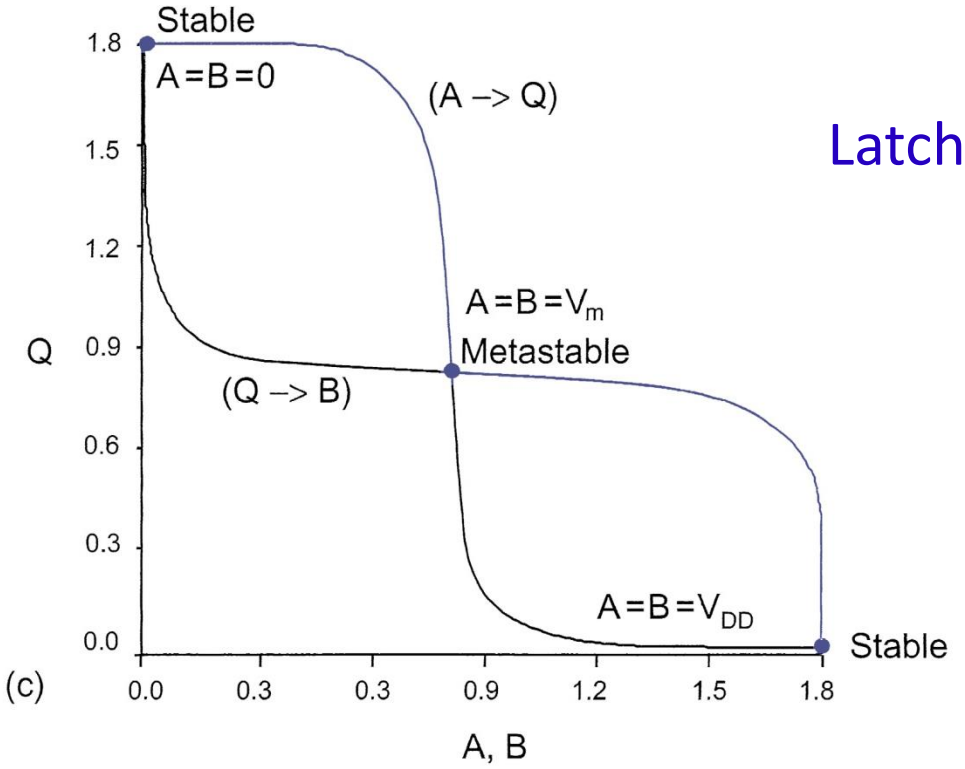
Metastability



(a)

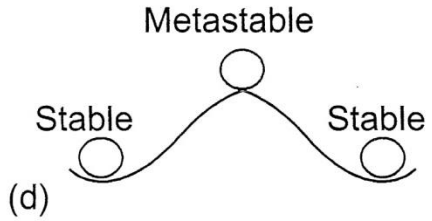


(b)



(c)

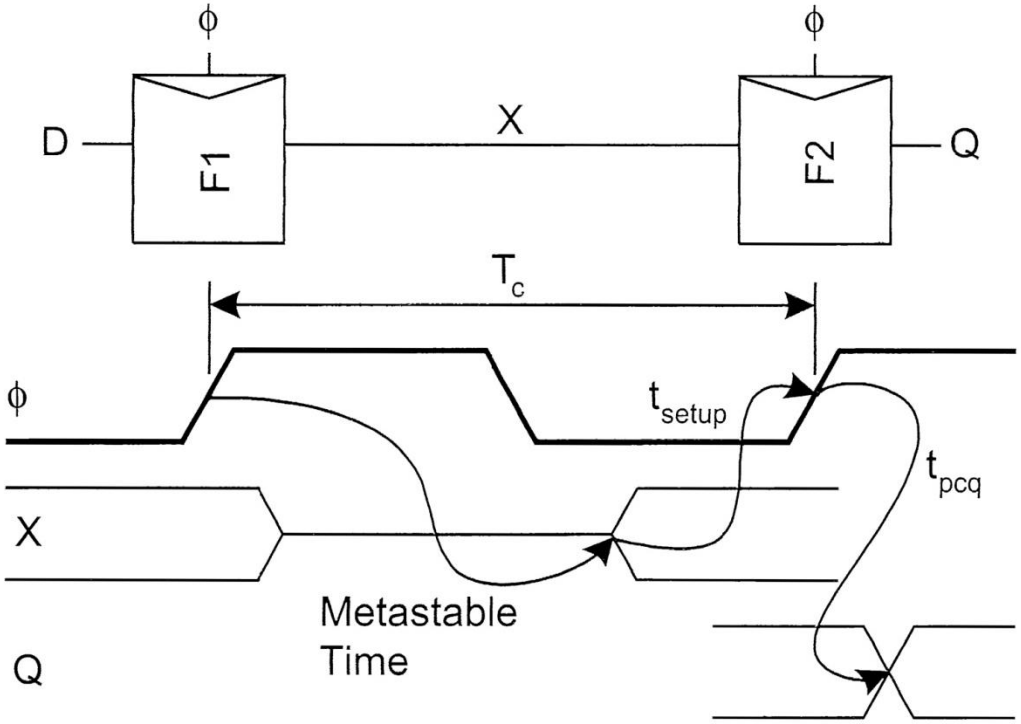
Latch is a bistable device



(d)

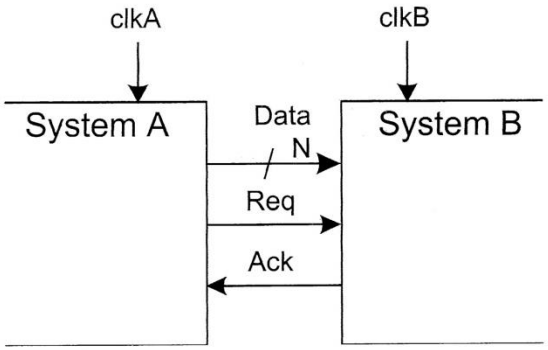
A Simple Synchronizer

- Synchronizer
  - Data should be stable during the aperture = t_{setup} + t_{hold} around clock rising edge.
  - One clock cycle latency

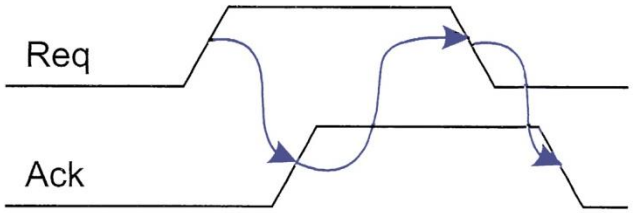


Communication of Async. Clock Domains

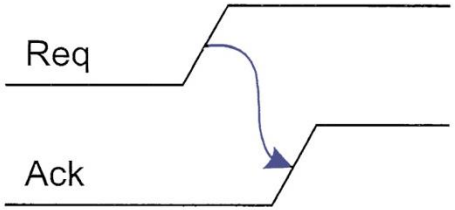
- Communication between asynchronous systems
  - Not share a common clock



- 4-phase and 2-phase handshake protocols
  - Request and acknowledge



(a) Four-phase Level sensitive



(b) Two-phase Edge trigger

Arbiters

- The arbiter is used to determine which of two inputs arrived first
  - If the time spacing exceeds aperture, first input will be acknowledged
  - If time spacing is too small, the choice is arbitrary