

# Lecture 15: Sequential Circuit Design

# What is Due

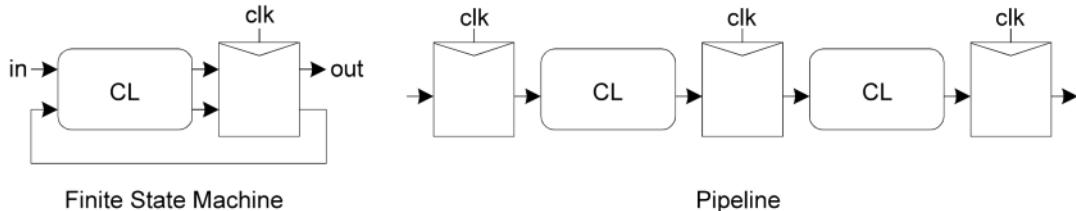
- Reading 10.1 – 10.3
- Project 2 part 3 due on 11/21
- Project 2 part 4 due on 12/5

# 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



16: Sequential Circuits

CMOS VLSI Design 4th Ed.

3

## Sequencing Cont.

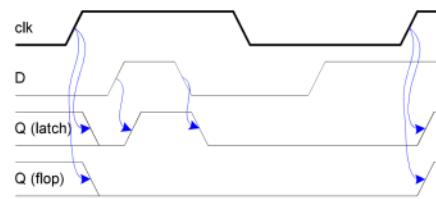
- If tokens moved through pipeline at constant speed, no sequencing elements would be necessary
- Ex: fiber-optic cable
  - Light pulses (tokens) are sent down cable
  - Next pulse sent before first reaches end of cable
  - No need for hardware to separate pulses
  - But *dispersion* sets min time between 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

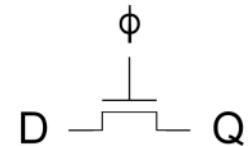
# Sequencing Elements

- **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

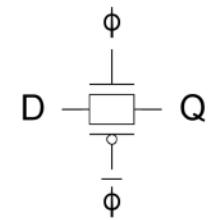
- Pass Transistor Latch
- Pros
  - + Tiny
  - + Low clock load
- Cons
  - $V_t$  drop
  - nonrestoring
  - backdriving
  - output noise sensitivity
  - dynamic
  - diffusion input



Used in 1970's

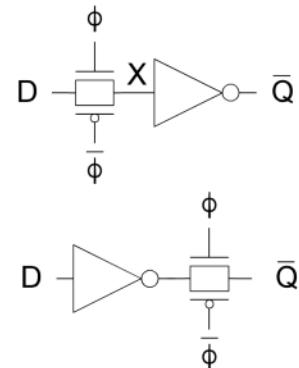
# Latch Design

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



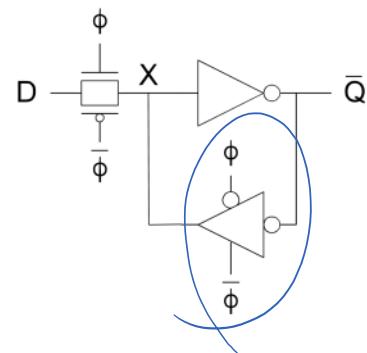
# Latch Design

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



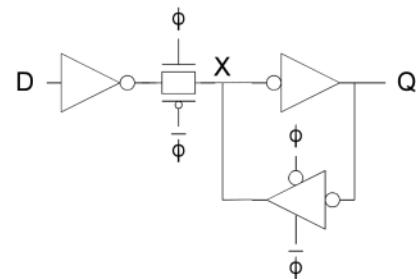
# Latch Design

- Tristate feedback
  - + Static
  - Backdriving risk
- Static latches are now essential because of leakage



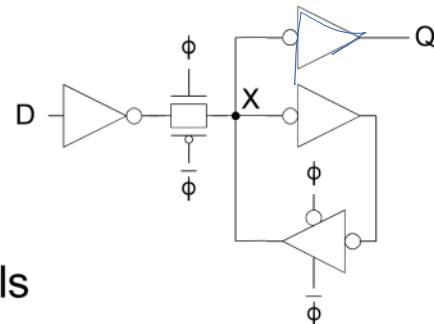
# Latch Design

- Buffered input
  - + Fixes diffusion input
  - + Noninverting



# Latch Design

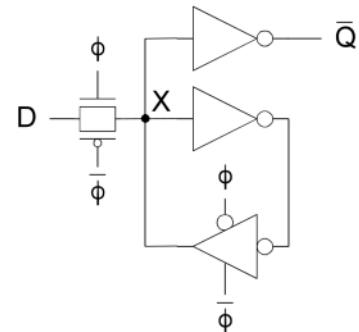
- Buffered output
  - + No backdriving



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

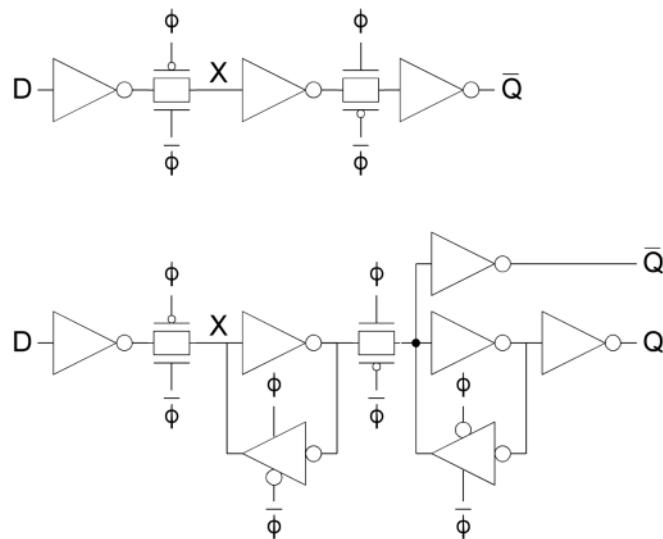
# Latch Design

- Datapath latch
  - + smaller
  - + faster
  - unbuffered input



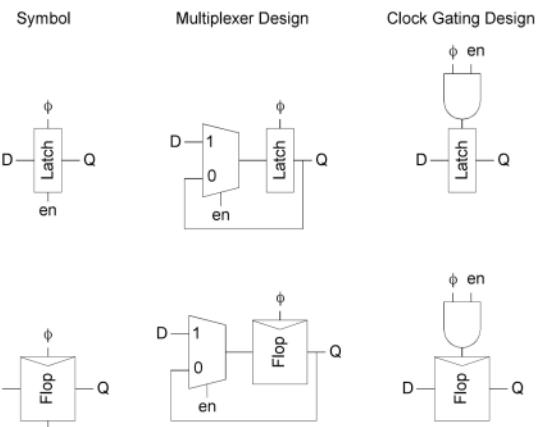
# Flip-Flop Design

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



# Enable

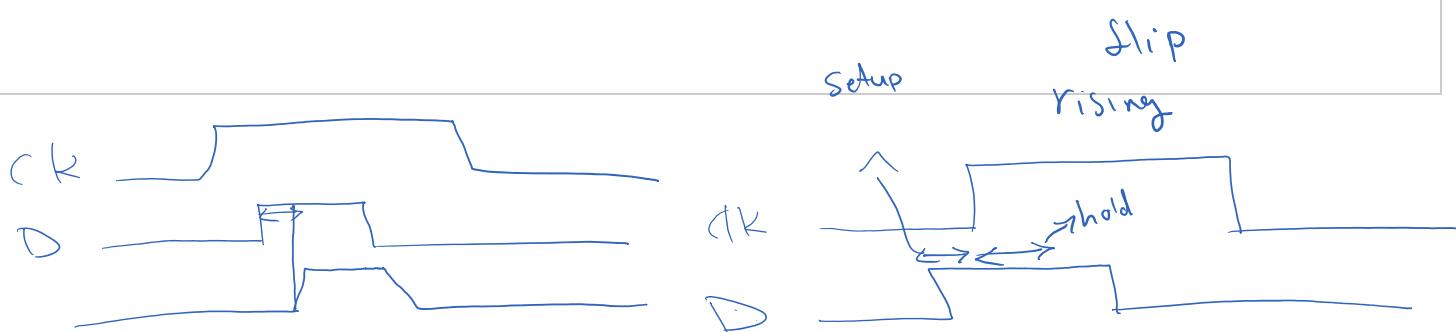
- Enable: ignore clock when en = 0
  - Mux: increase latch D-Q delay
  - Clock Gating: increase en setup time, skew



16: Sequential Circuits

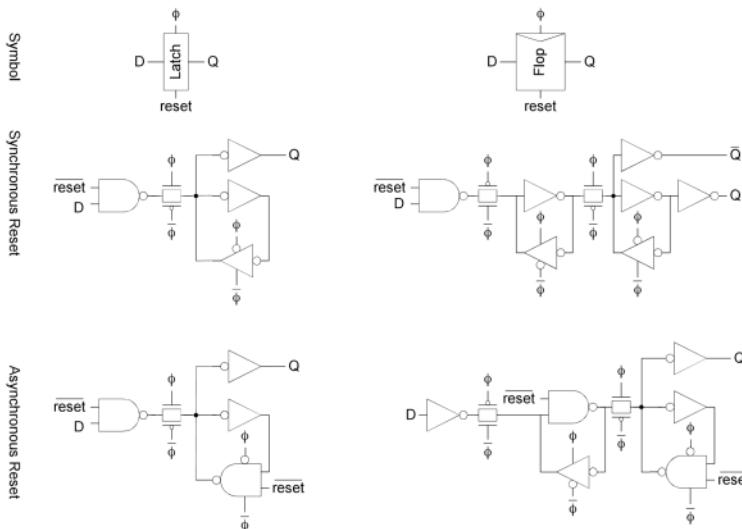
CMOS VLSI Design 4th Ed.

15



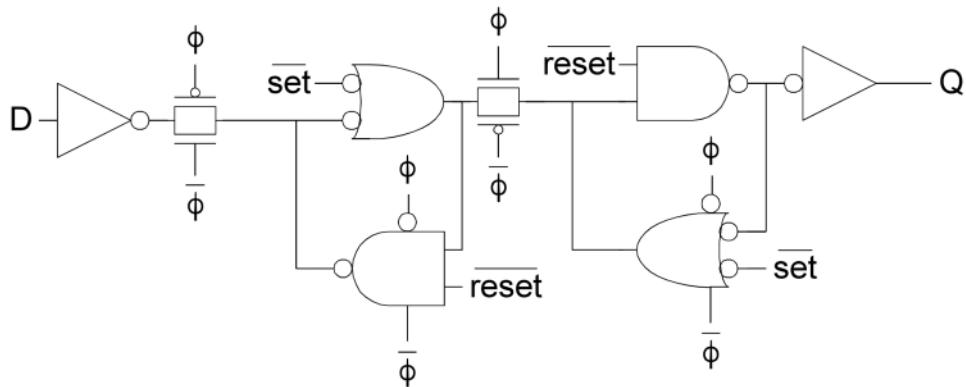
# Reset

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



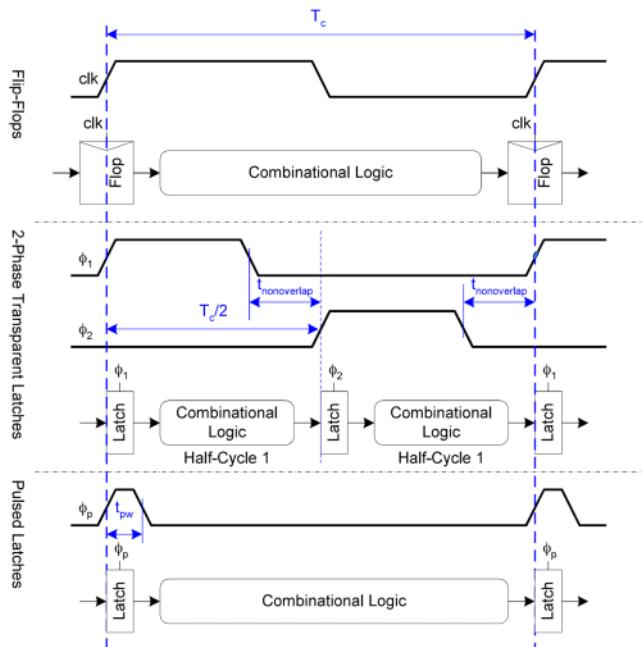
## Set / Reset

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



# Sequencing Methods

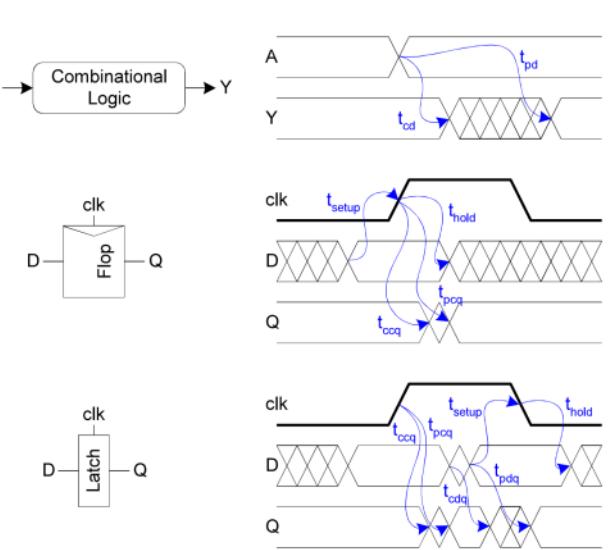
- Flip-flops
- 2-Phase Latches
- Pulsed Latches



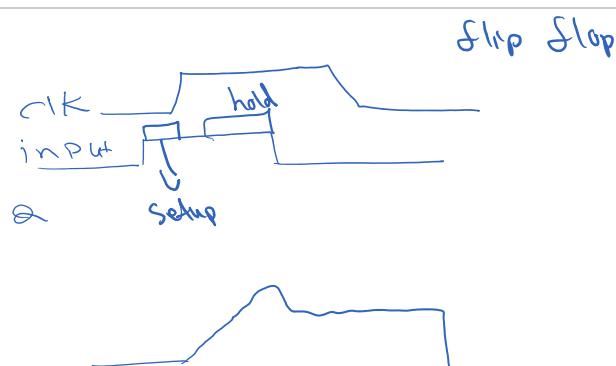
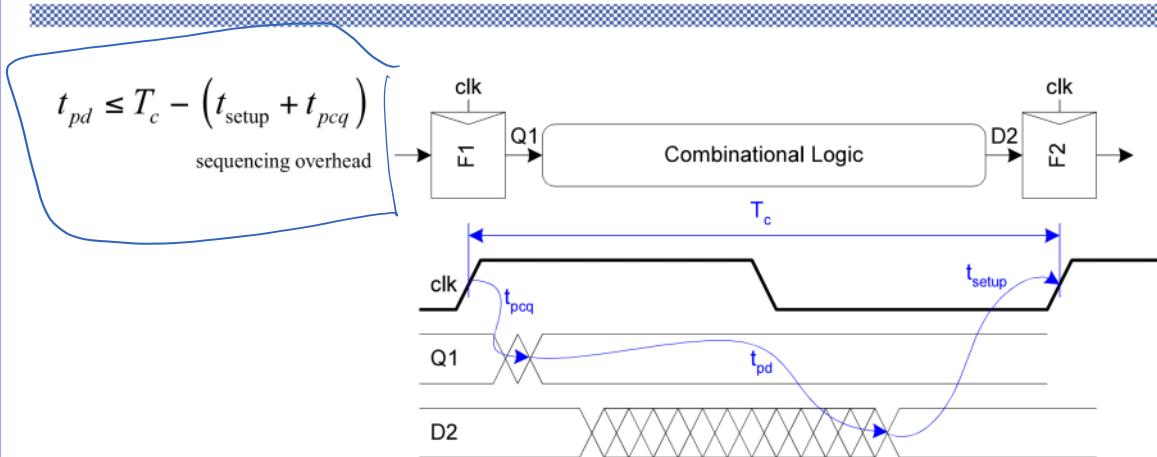
# Timing Diagrams

## 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 <u>Setup Time</u>
$t_{hold}$	Latch/Flop <u>Hold Time</u>



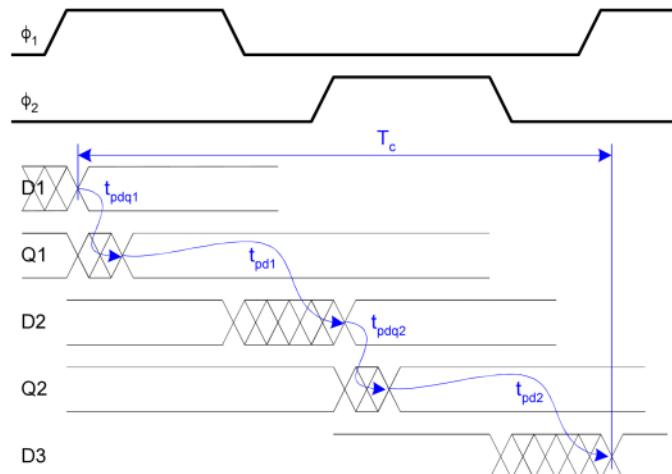
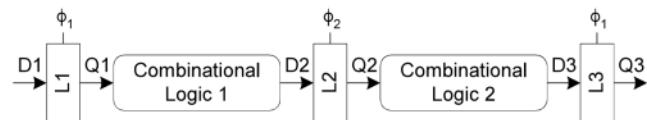
# Max-Delay: Flip-Flops



# Max Delay: 2-Phase Latches

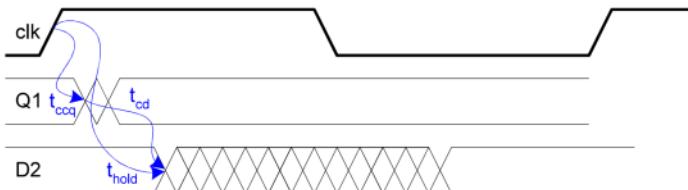
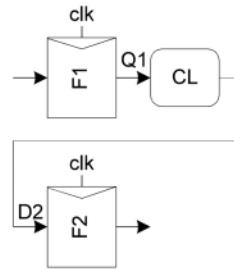
$$t_{pd} = t_{pd1} + t_{pd2} \leq T_c - (2t_{pdq})$$

sequencing overhead



# Min-Delay: Flip-Flops

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



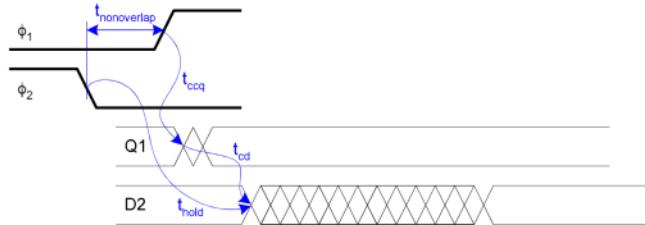
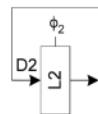
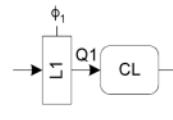
# Min-Delay: 2-Phase Latches

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

Hold time reduced by nonoverlap

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

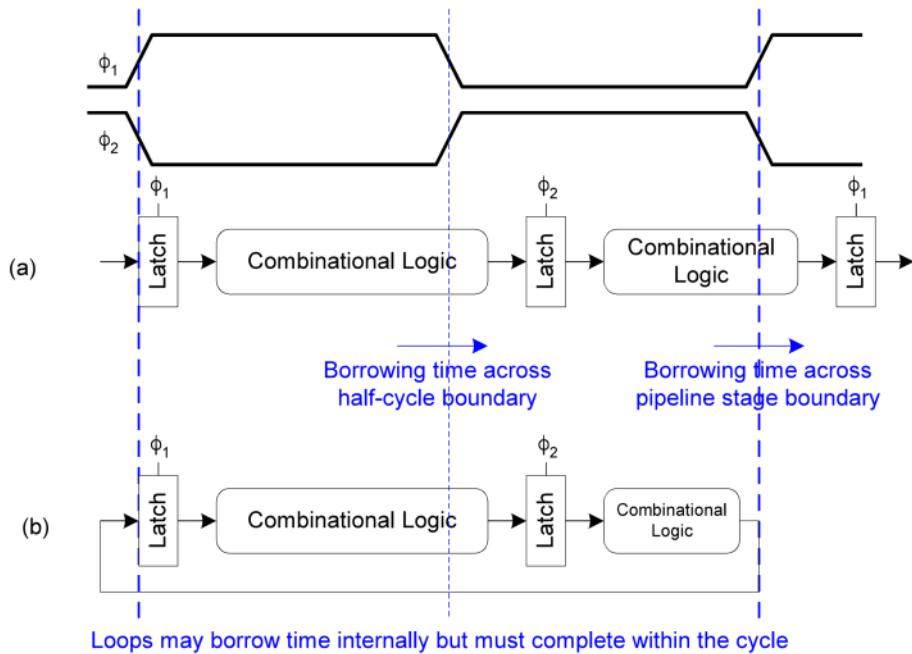
But a flop is made of two latches!



# 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



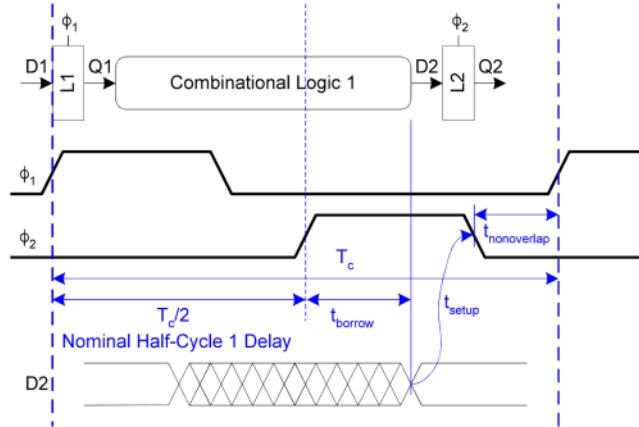
# How Much Borrowing?

## 2-Phase Latches

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

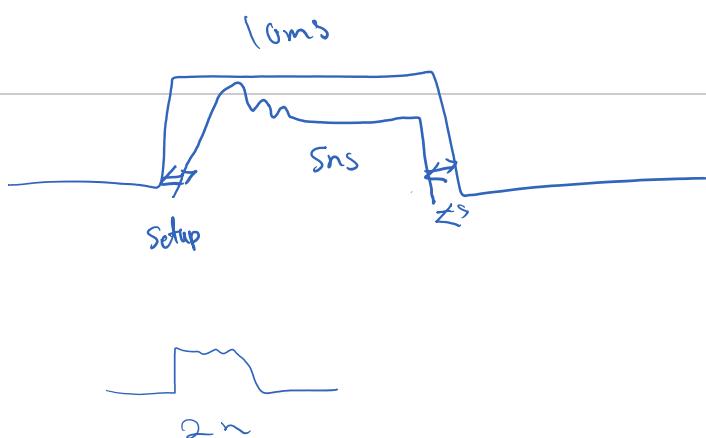
## Pulsed Latches

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



# 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

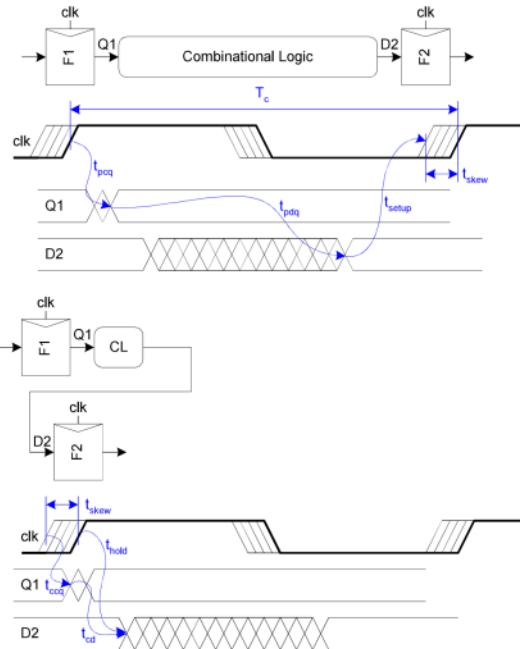


# Skew: Flip-Flops

$$t_{pd} \leq T_c - t_{pcq} + t_{\text{setup}} + t_{\text{skew}} \quad )$$

sequencing overhead

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



# Skew: Latches

## 2-Phase Latches

$$t_{pd} \leq T_c - (2t_{pdq})$$

sequencing overhead

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

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

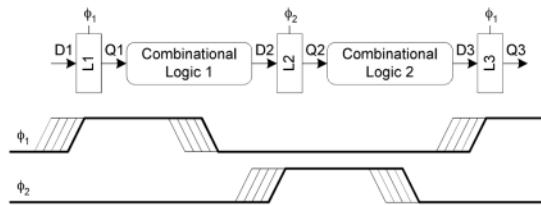
## Pulsed Latches

$$t_{pd} \leq T_c - \max(t_{pdq}, t_{pcq} + t_{setup} - t_{pw} + t_{skew})$$

sequencing overhead

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

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

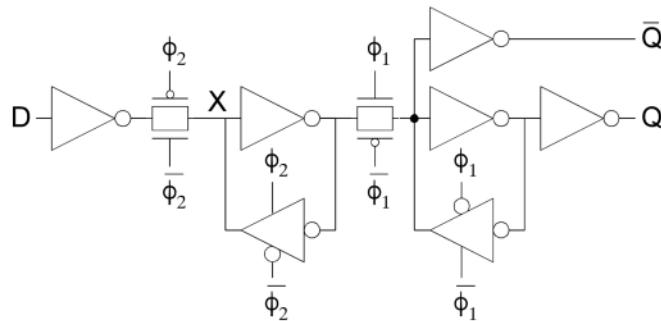


## Two-Phase Clocking

- 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  $\phi_1, \phi_2$  (ph1, ph2)

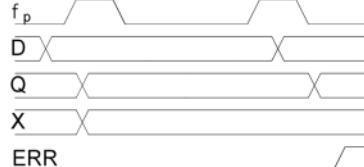
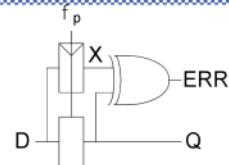
# Safe Flip-Flop

- Past years used flip-flop with nonoverlapping clocks
  - 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



# Adaptive Sequencing

- Designers include timing margin
  - Voltage
  - Temperature
  - 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%

# Summary

- Flip-Flops:
  - Very easy to use, supported by all tools
- 2-Phase Transparent Latches:
  - Lots of skew tolerance and time borrowing
- Pulsed Latches:
  - Fast, some skew tol & borrow, hold time risk

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