

# Sequential circuit design with metastability

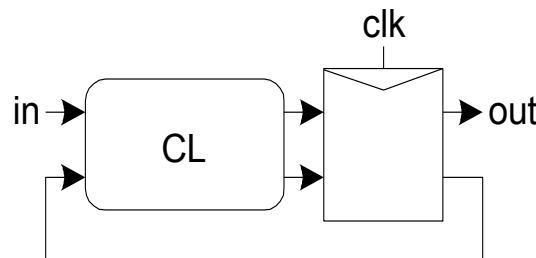
Lecture 11

October 16, 2018

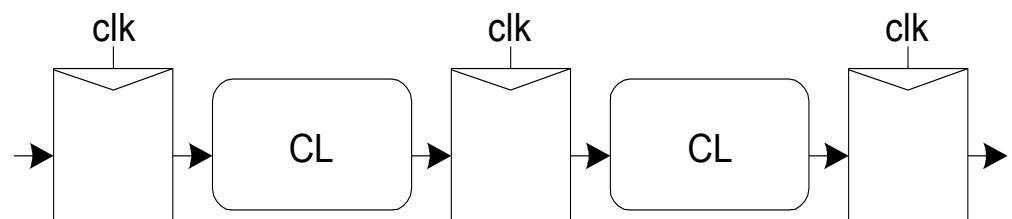
Preliminary version

# Sequential Logic

- *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*
  - Examples: Finite state machine (FSM), pipeline



Finite State Machine



Pipeline

# Sequencing

- If tokens move through pipeline at constant speed, no sequencing elements are 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
  - Solution: delay fast tokens so they don't catch up on slow ones.

# Sequencing overhead

- Solution: Use flip-flops to delay fast tokens so they move through exactly one stage each cycle.
- Drawback: Adds some delay also to the slow tokens
- Thus, circuit is slower than just logic propagation delay
  - Added delay is called *sequencing overhead*
- Some people call this clocking overhead
  - But it applies to asynchronous circuits too
  - Inevitable side effect of maintaining sequence

# This lecture

- Compute *sequencing overhead* with flip-flops
  - Compute maximum clock frequency
  - Or constraints on logic delay with given clock frequency
  - Also with clock skew
- Understand causes of delays in flip-flops and their design tradeoffs
  - But not design them
- Metastability

# Return to adder example

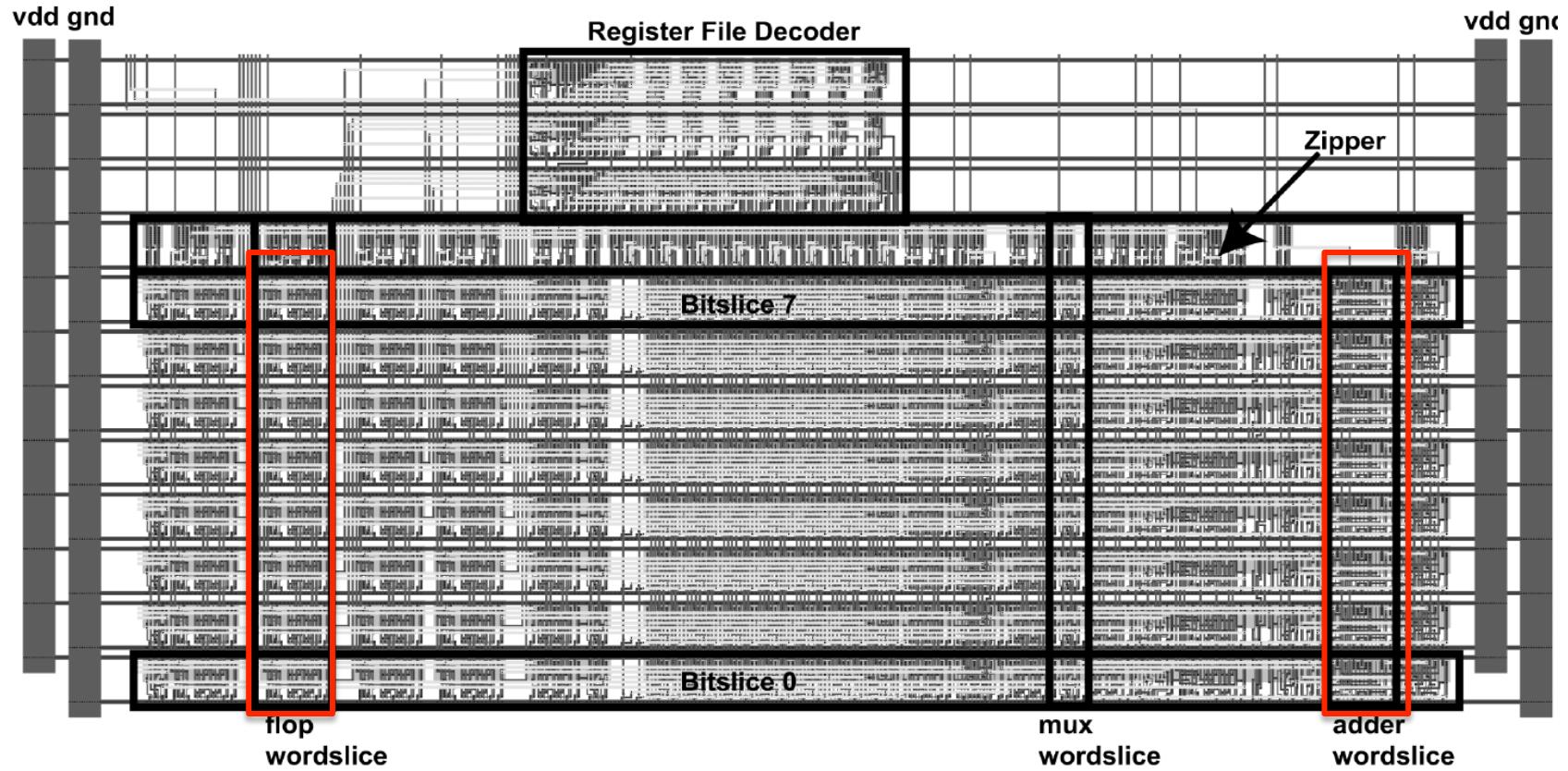
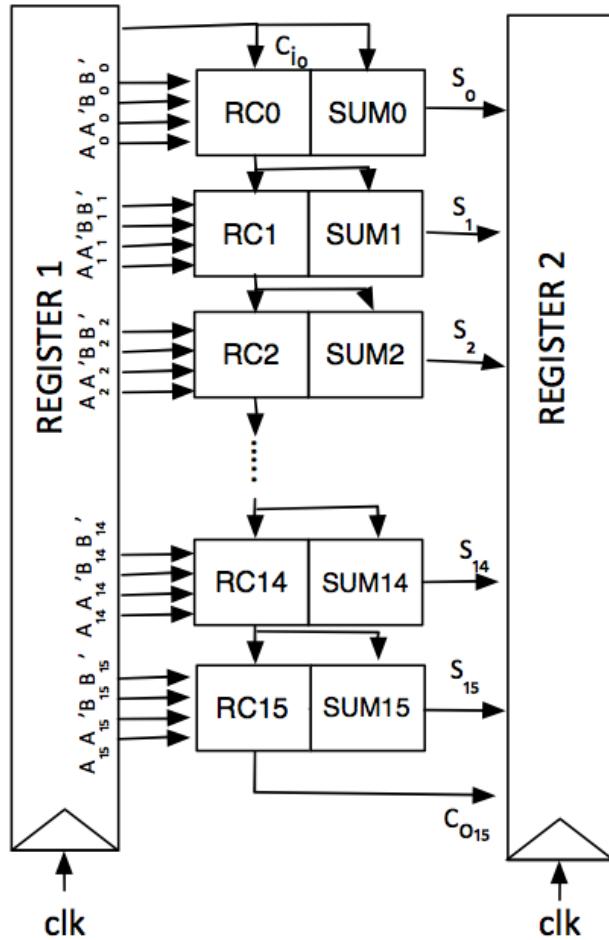
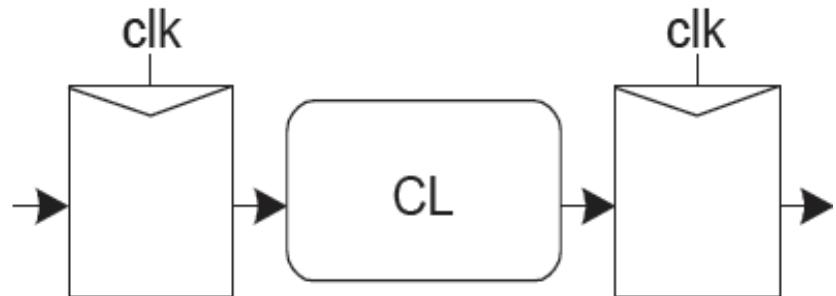


Figure 1.67 from W&H

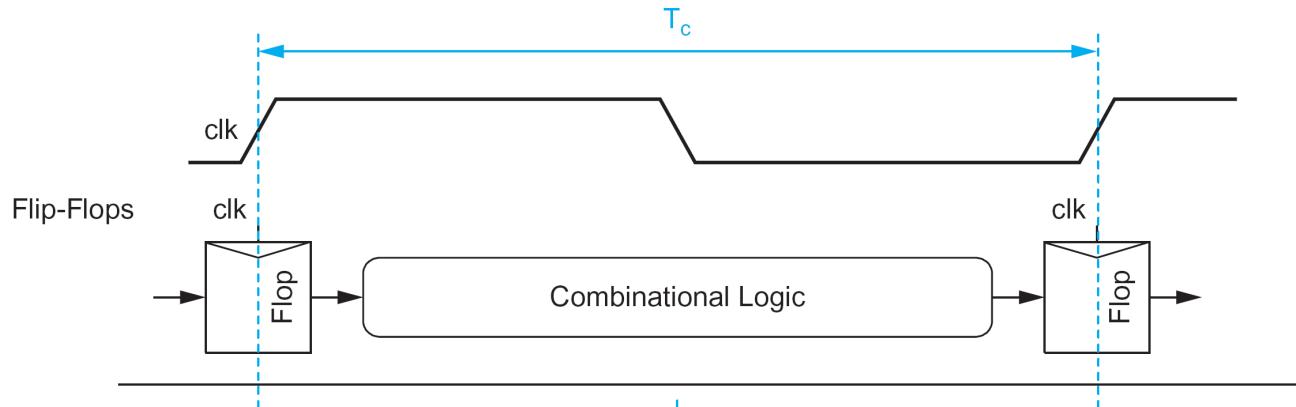
# Ripple-carry adder (from postlab 2)



A way of drawing  
the same thing with less detail for  
the combinational logic (CL)



# Sequencing with flip-flops



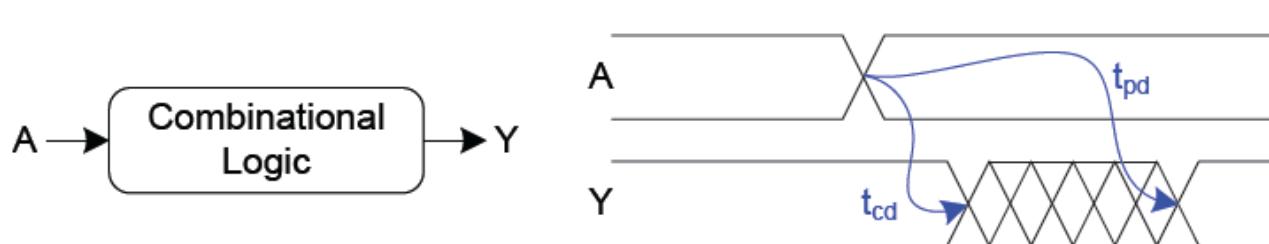
Flip-flop: Edge-triggered (positive edge) holds data until next edge

Goals today:

- Given a certain CL determine the minimum possible  $T_c$  with and without clock skew.
- Given certain  $f_c$ , clock skew and flip-flops determine timing requirements on CL.

# Characterizing combinational logic

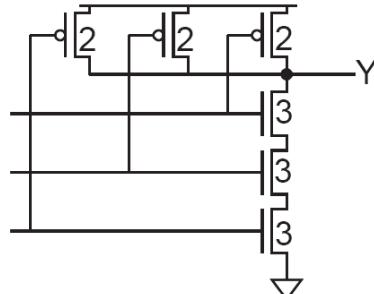
- Propagation delay,  $t_{pd}$ :
  - **Maximum** time until output **finally** reaches  $V_{DD}/2$ .
  - Output is guaranteed not to change **after**  $t_{pd}$ .
- Contamination delay,  $t_{cd}$ :
  - **Minimum** time until output **initially** reaches  $V_{DD}/2$ .
  - Output is guaranteed not to change **before**  $t_{cd}$ .



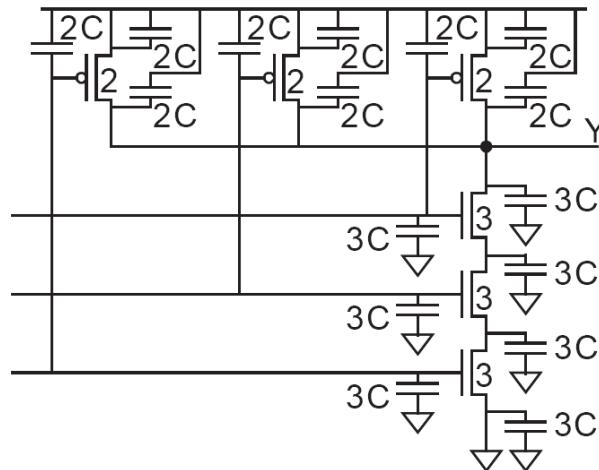
# Contamination delay

## When is delay shorter than $t_{pd}$ ?

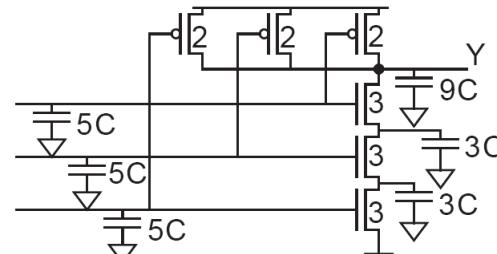
3-input NAND gate, Example 4.2 from W&H, Figure 4.7



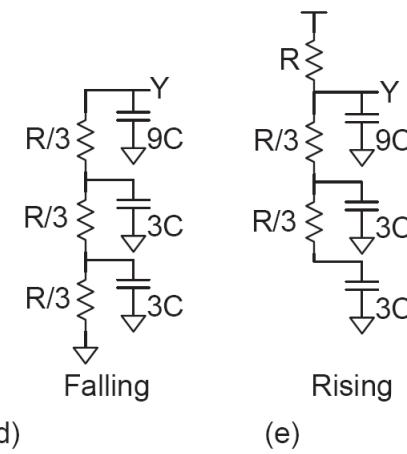
(a)



(b)



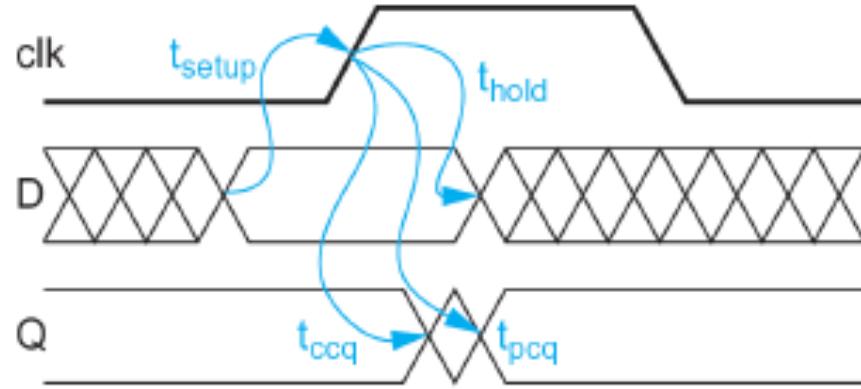
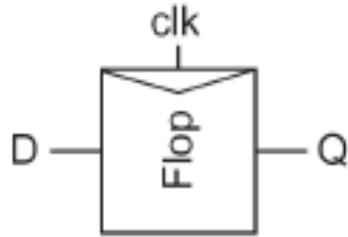
(c)



(d)

(e)

# Flip-flop operation and delays

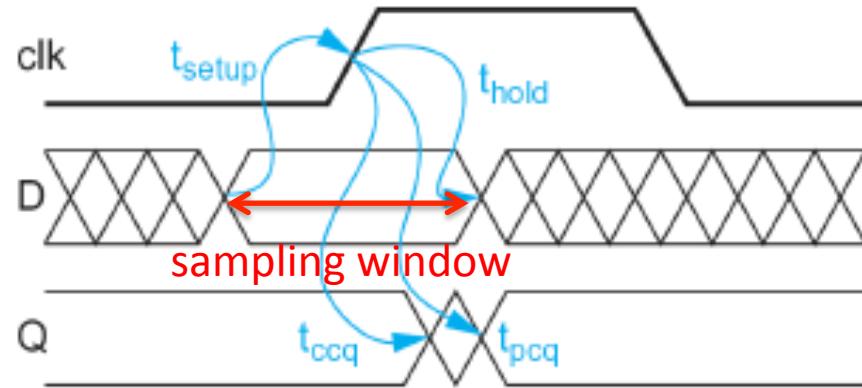
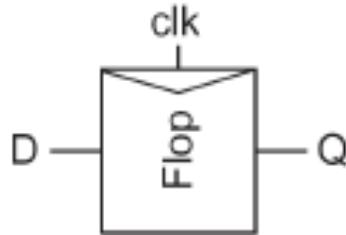


Edge-triggered (positive edge) holds data until next positive clock edge.  
Input is called D output is called Q.

Input D has to be ready before clock edge happens, so that correct data is read.  
Input D has to be stable long enough after clock edge happened to be correctly locked by flip-flop.

From clock edge happening it takes some time until output Q is available, there is a maximum and a minimum delay.

# Flip-flop operation and delays

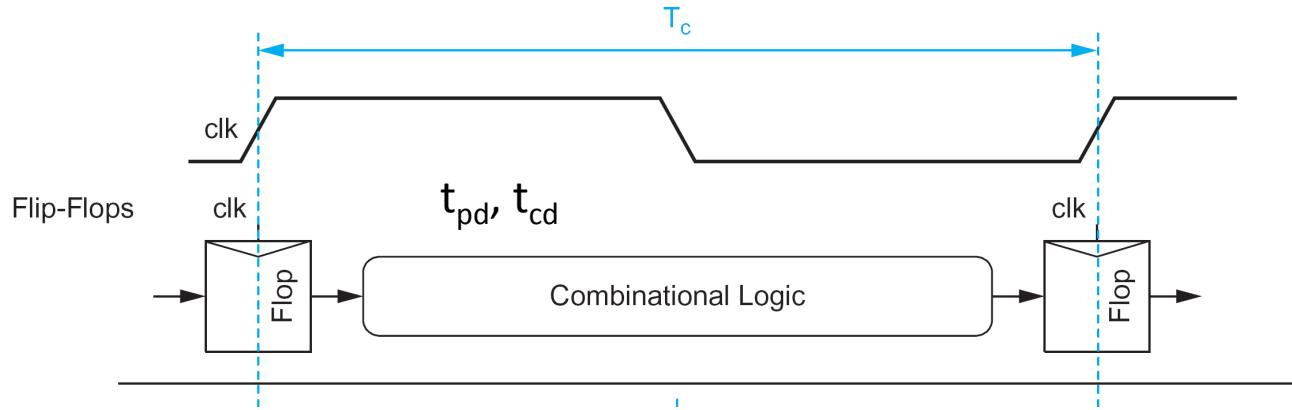


Abbreviation	Flip-flop delay
$t_{pcq}$	Clk-to-Q propagation delay
$t_{ccq}$	Clk-to-Q contamination delay
$t_{setup}$	Setup time
$t_{hold}$	Hold time

**Setup time** is always positive.

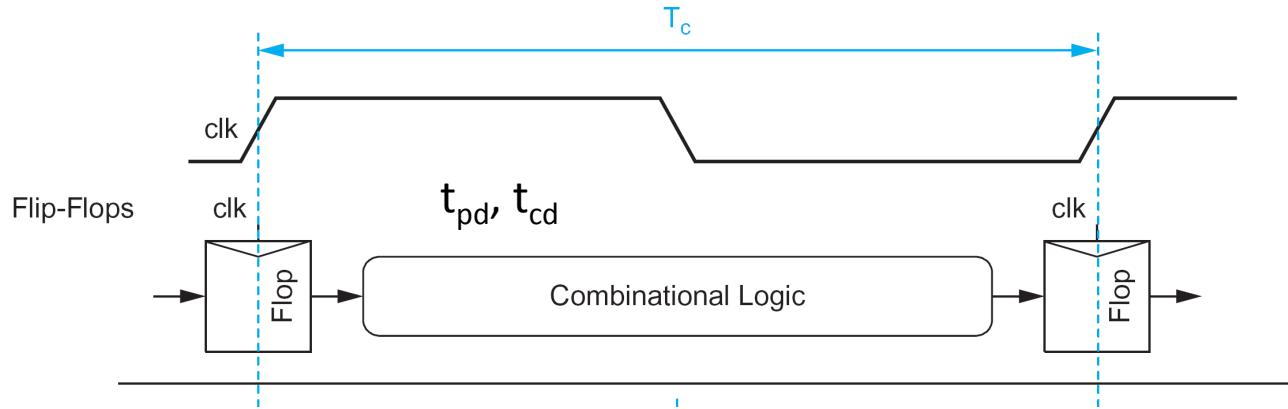
**Hold time** can be positive, zero or negative.

# What is minimum $T_c$ ?



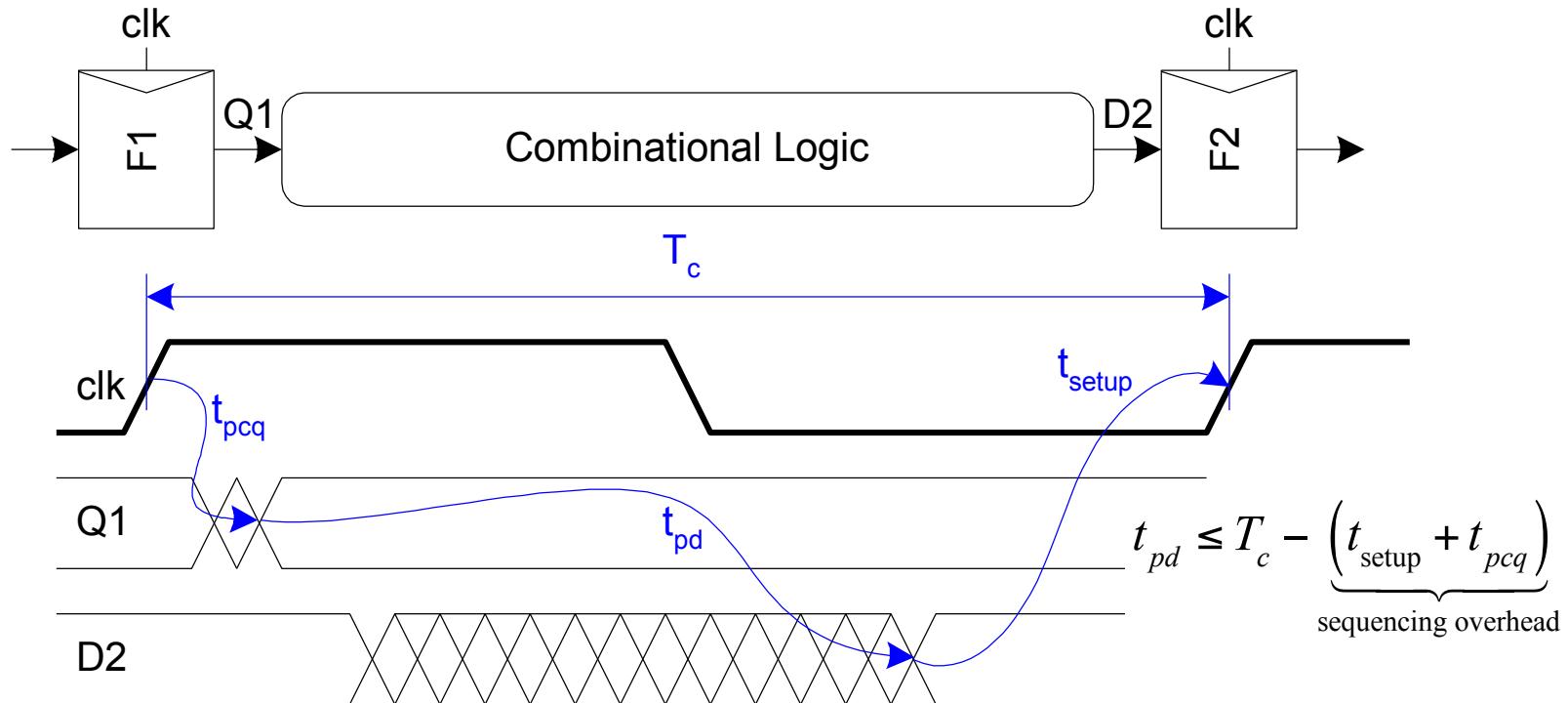
Abbreviation	Flip-flop delay
$t_{pcq}$	Clk-to-Q propagation delay
$t_{ccq}$	Clk-to-Q contamination delay
$t_{\text{setup}}$	Setup time
$t_{\text{hold}}$	Hold time

# What is minimum $T_c$ ?

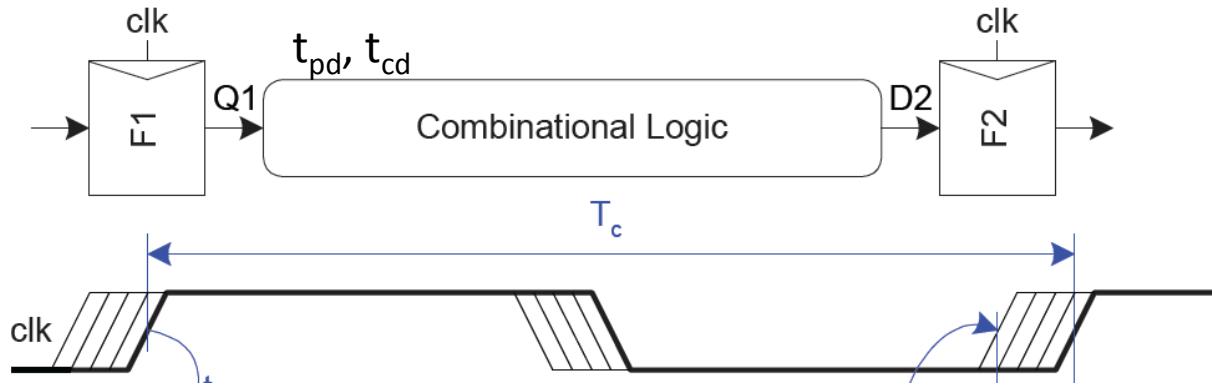


Make sure result from CL is always there when next clock edge happens:  
Otherwise there is a **setup violation**.

# Maximum propagation delay



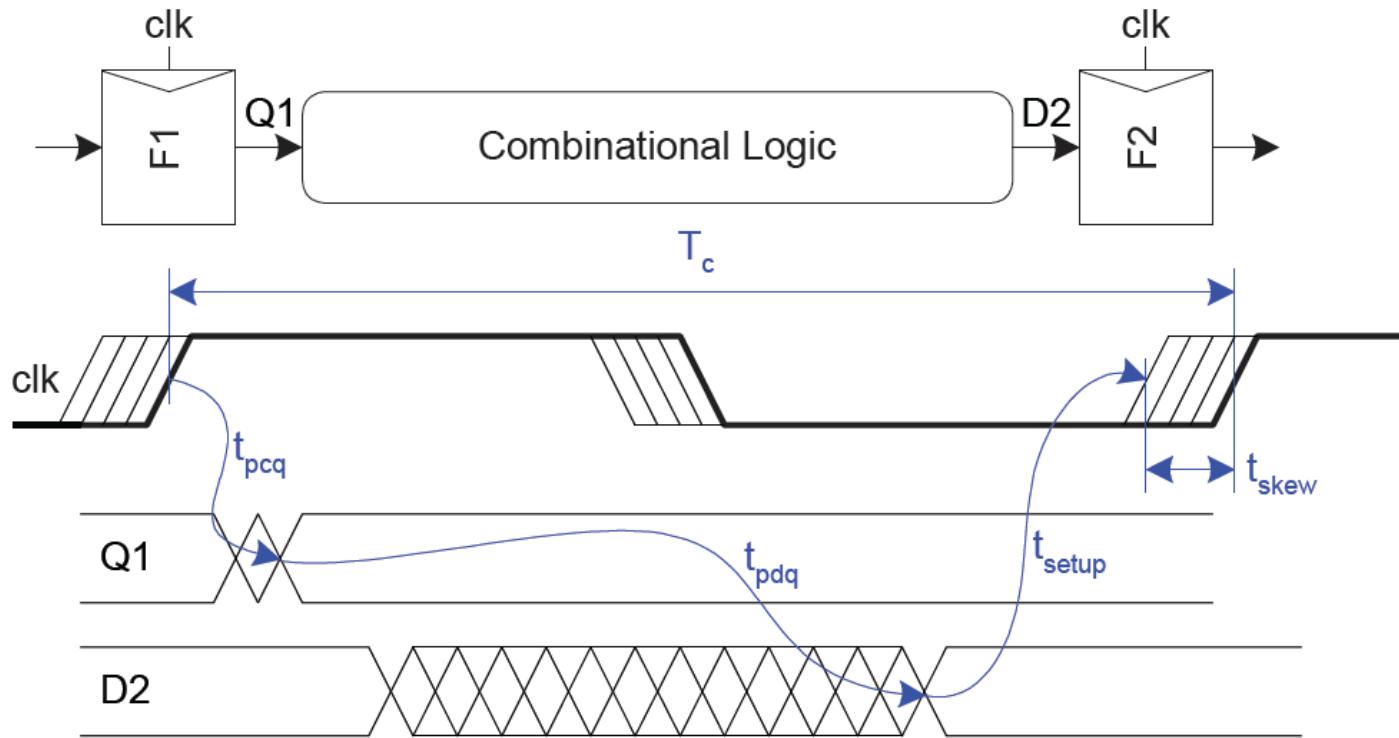
# What is minimum $T_c$ with clock skew?



Clock skew = difference in delay (arrival time) for clock signals.  
We do not know which way the difference will go.

Make sure result from CL is always there anyways when next clock edge happens

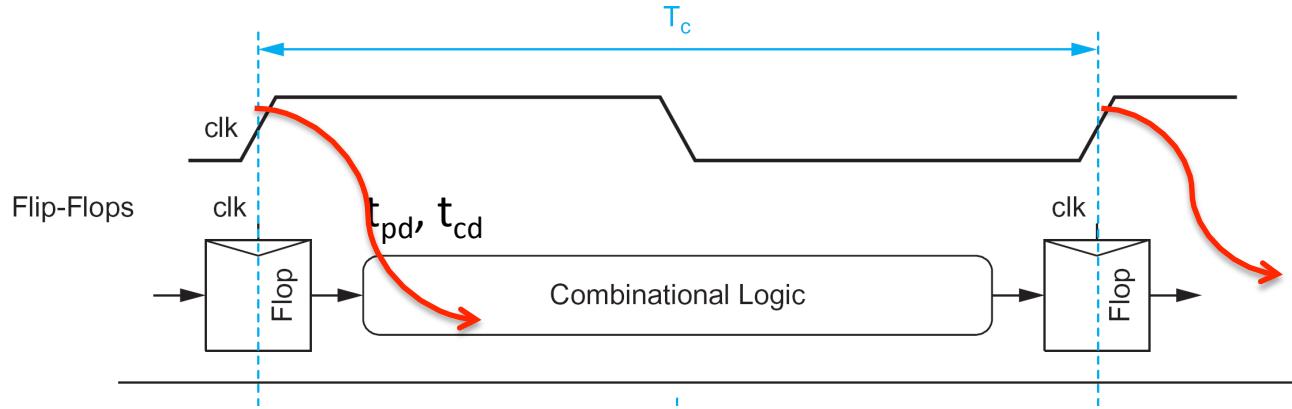
# Maximum propagation delay with clock skew



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

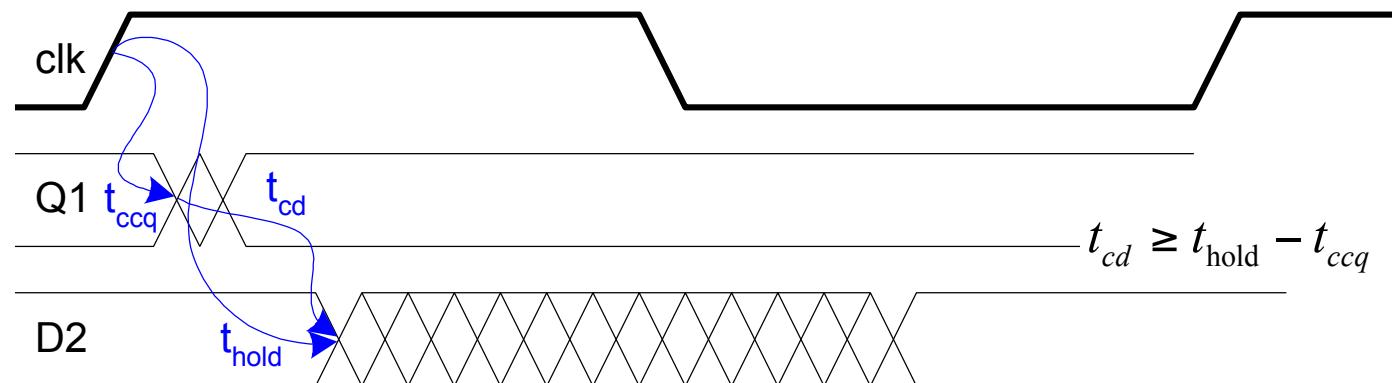
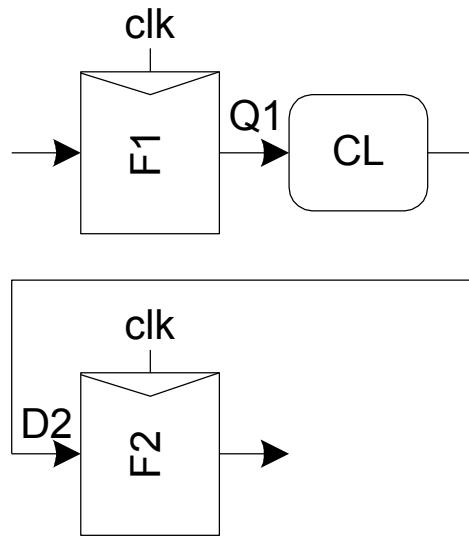
Decreases maximum propagation delay

# Another potential problem

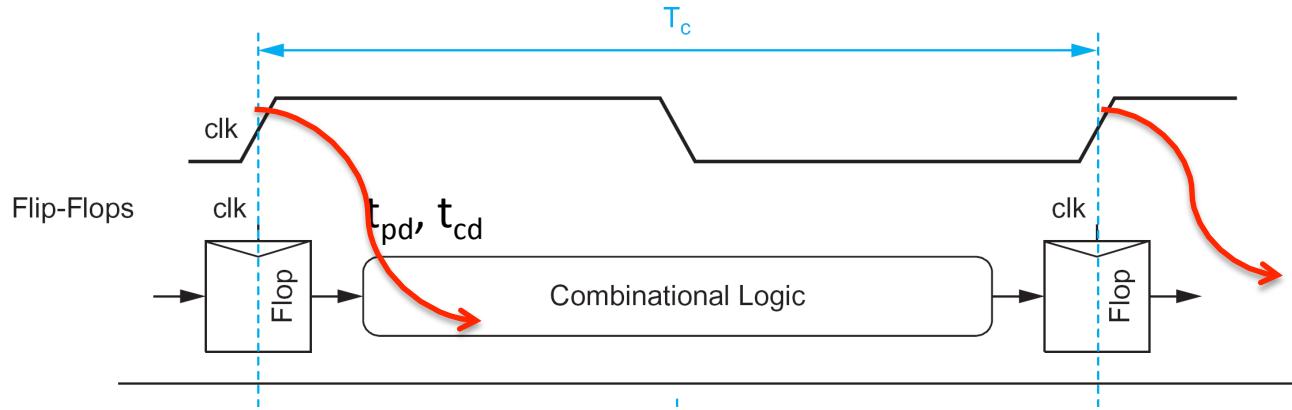


What if result from CL changes **before** the previous result has been locked in?  
Then tokens are merged and data is lost.

# Minimum Propagation Delay



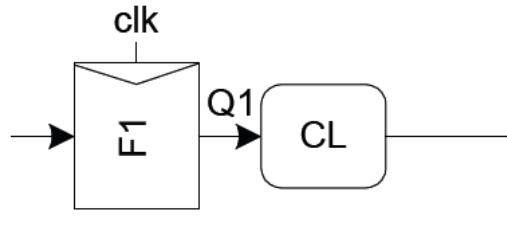
# Hold violation with clock skew?



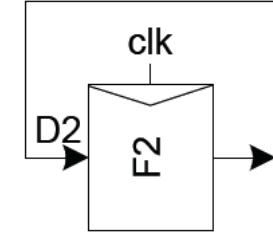
What if result from CL changes **before** the previous result has been read and we have clock skew?

Then tokens are merged and data is lost.

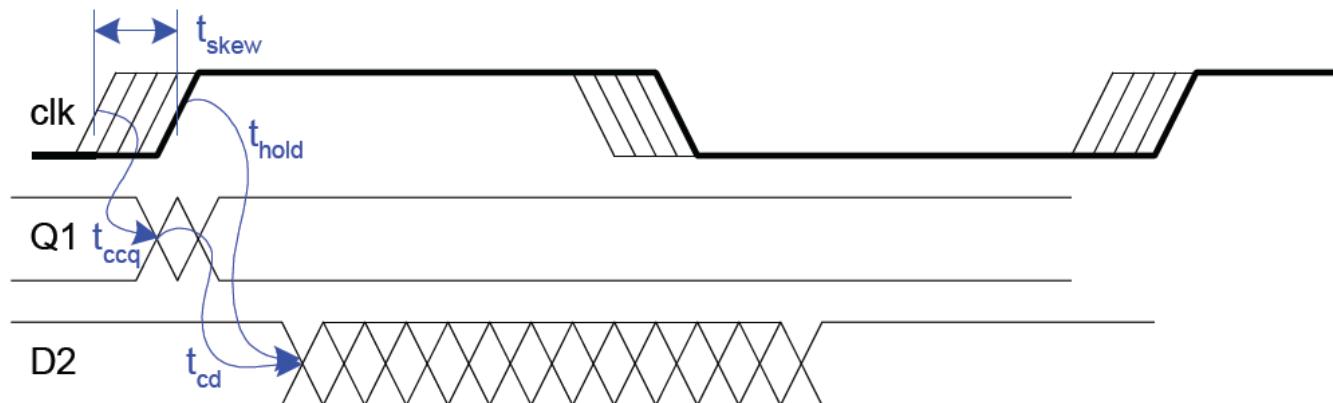
# Minimum propagation delay with clock skew



Increases minimum contamination delay



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



# Are setup or hold violations worse?

# System balancing

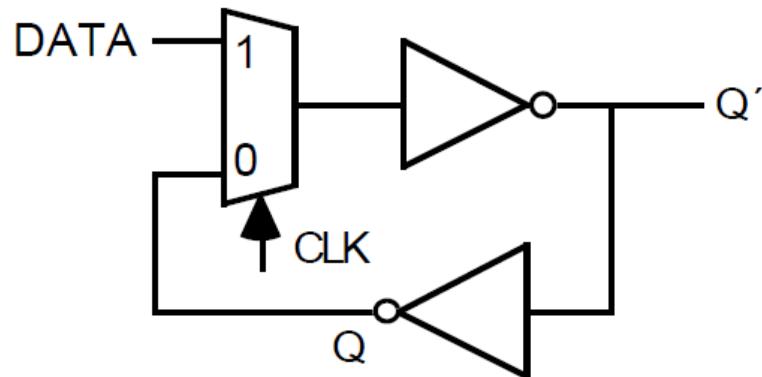
- A well-designed system is **balanced**
- No need to overdesign one part if another part limits system performance.
- Tradeoff between:
  - CL: delays, area, power
  - Flip-flop design: delays, area, power, metastability
  - Clock generation and distribution: ~~delay~~ clock skew, area, power

# How design a flip-flop?

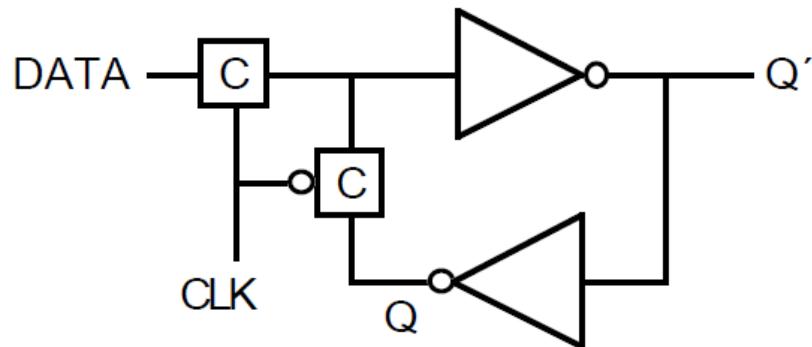
- Important requirements:
  - Data is always maintained and restored
  - Short delays
  - Low capacitive load on clocks signals
  - Small
  - Enable
  - Reset/set (synchronous or asynchronous)

# D-Latch Example

A MUX on the input allows us to either load new data or keep old data

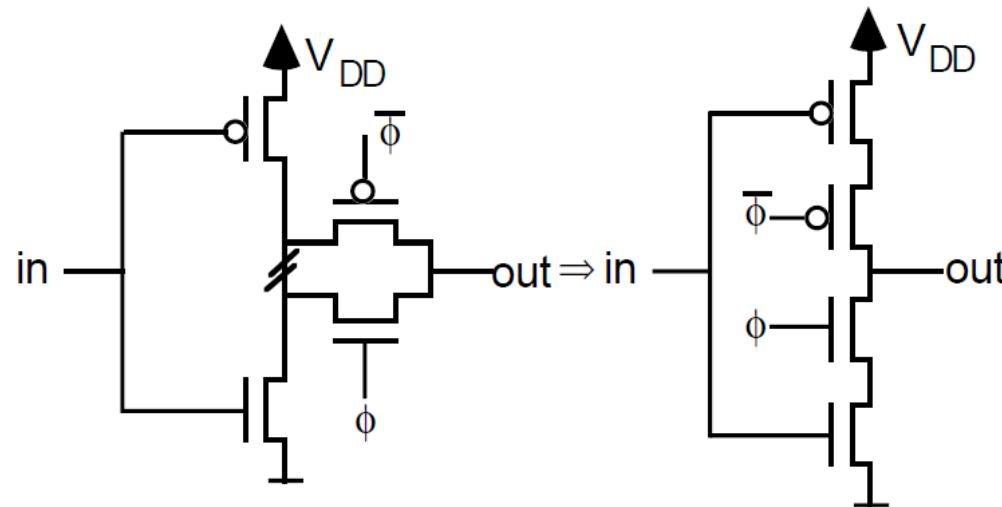
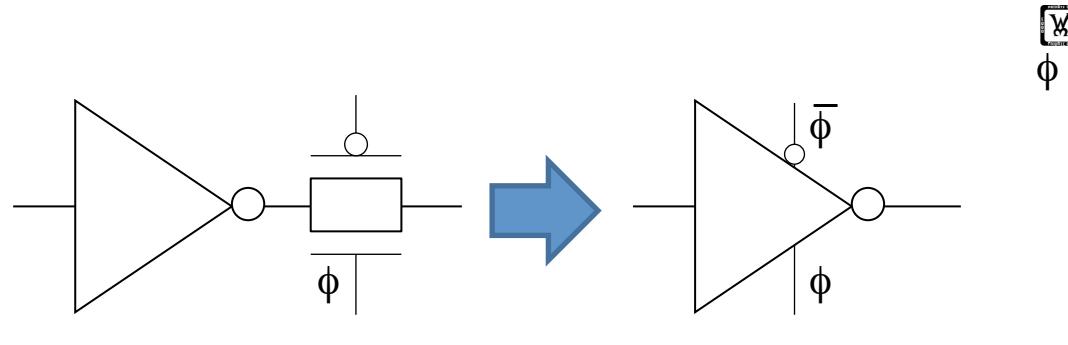


Two C-switches make a simple MUX



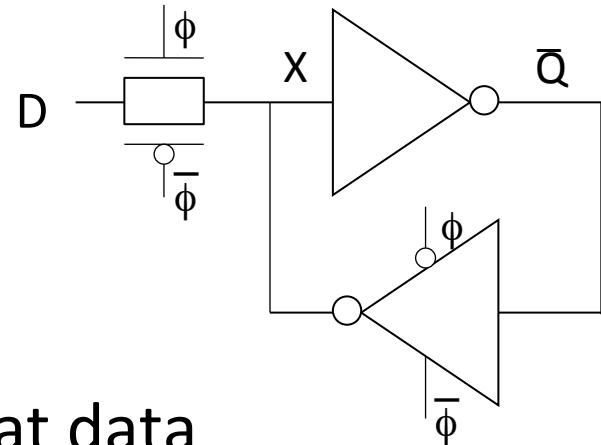
# Using tri-state inverters

However, an inverter and a C-switch can be replaced by tri-state inverter



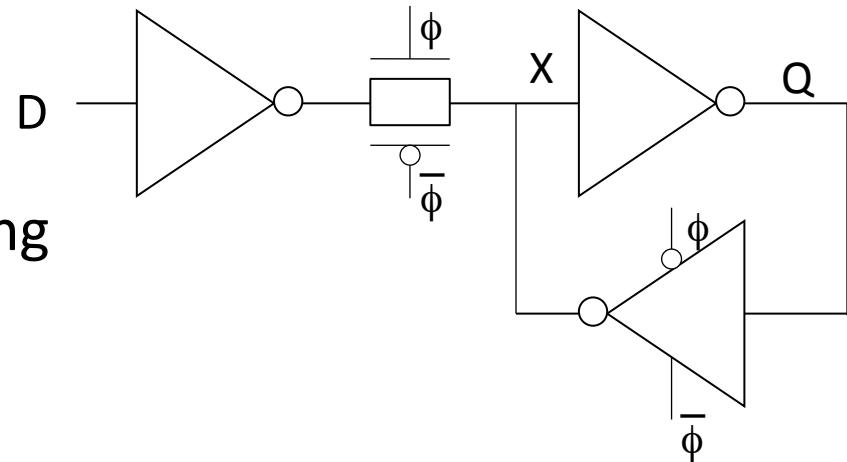
# D-Latch Design

- Tristate feedback
  - + Static
  - Backdriving risk
- Static latches are essential so that data does not disappear because of leakage



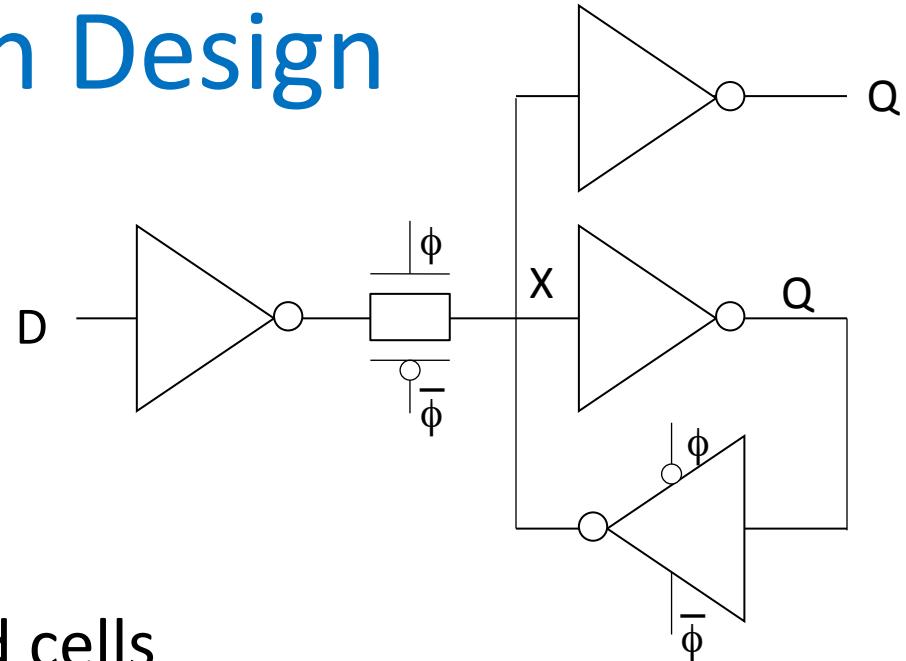
# D-Latch Design

- Buffered input
  - + Fixes diffusion input
  - + Makes latch noninverting



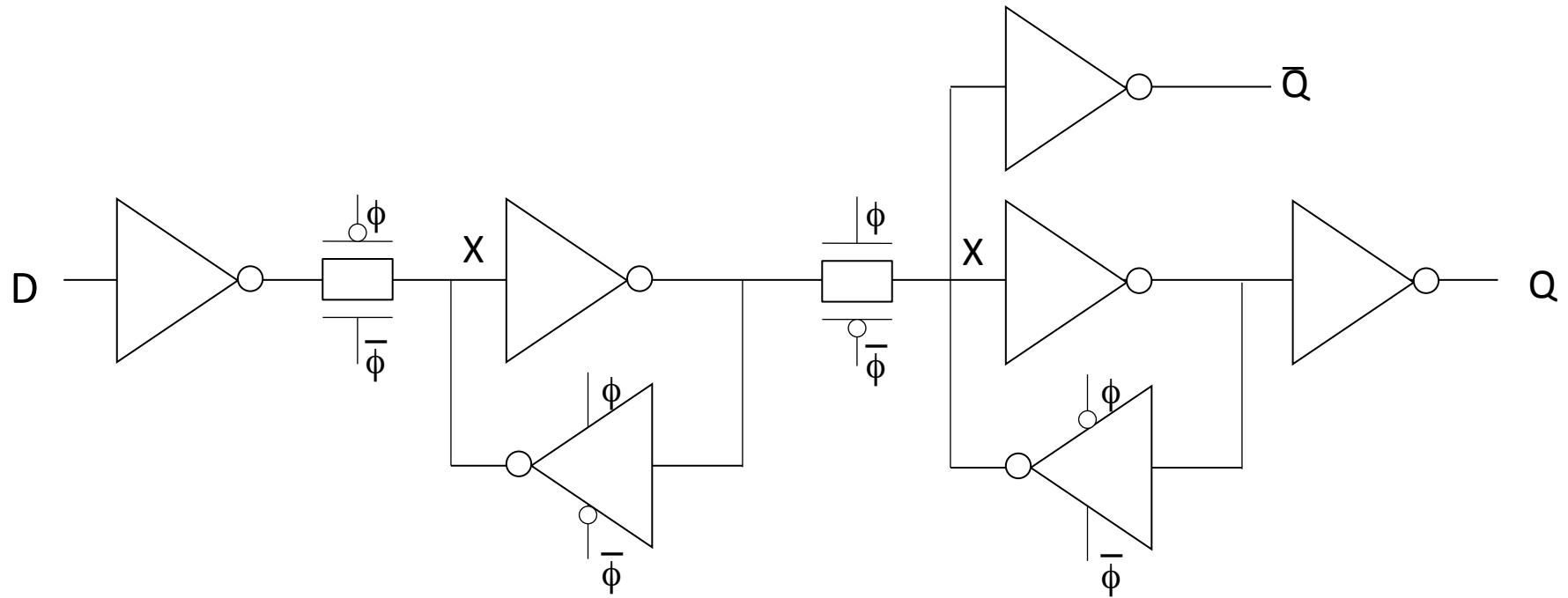
# D-Latch Design

- 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



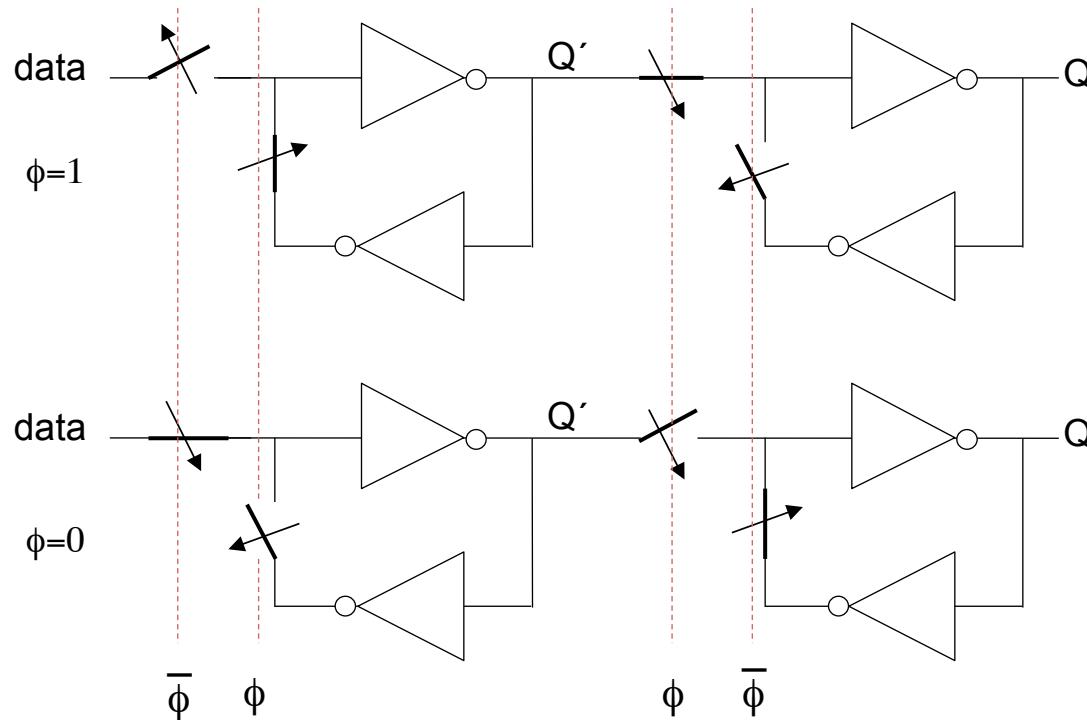
# D-type Flip-Flop Design

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



# D-type Flip-Flop Design

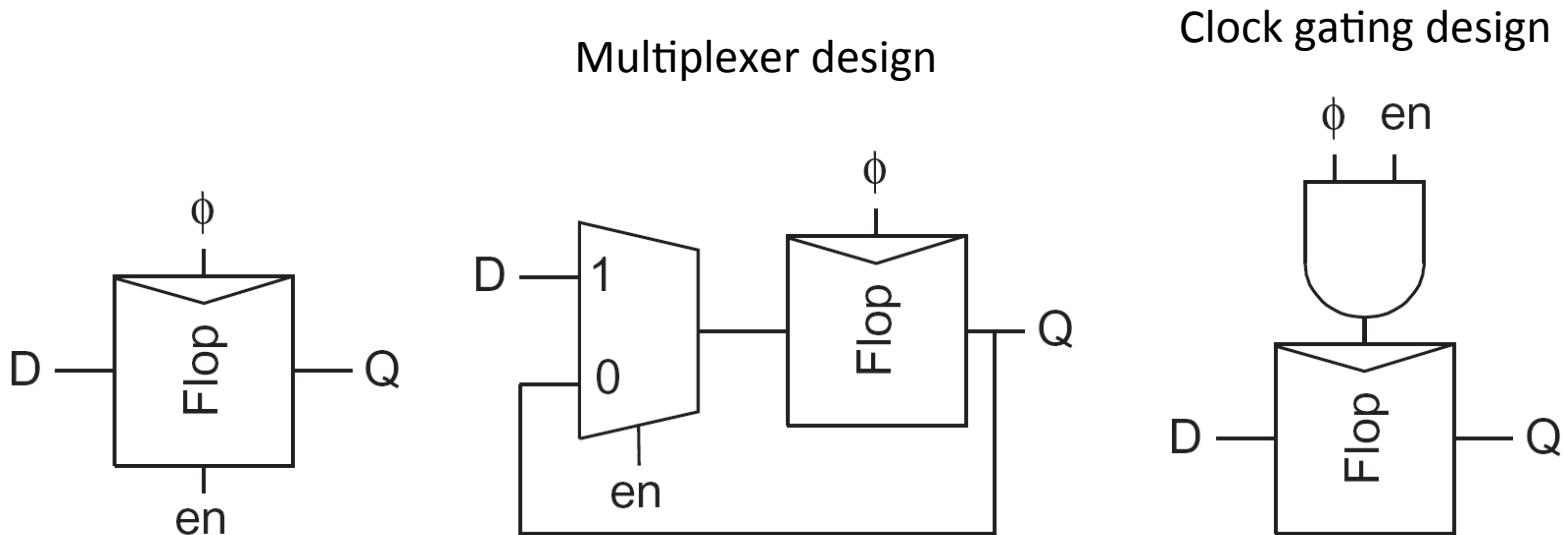
- Data is transferred from master to slave when  $CLK=1$



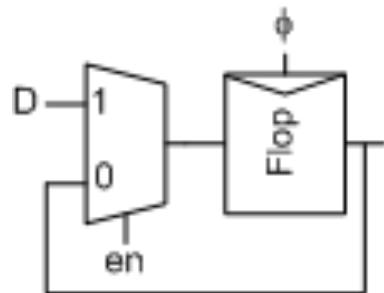
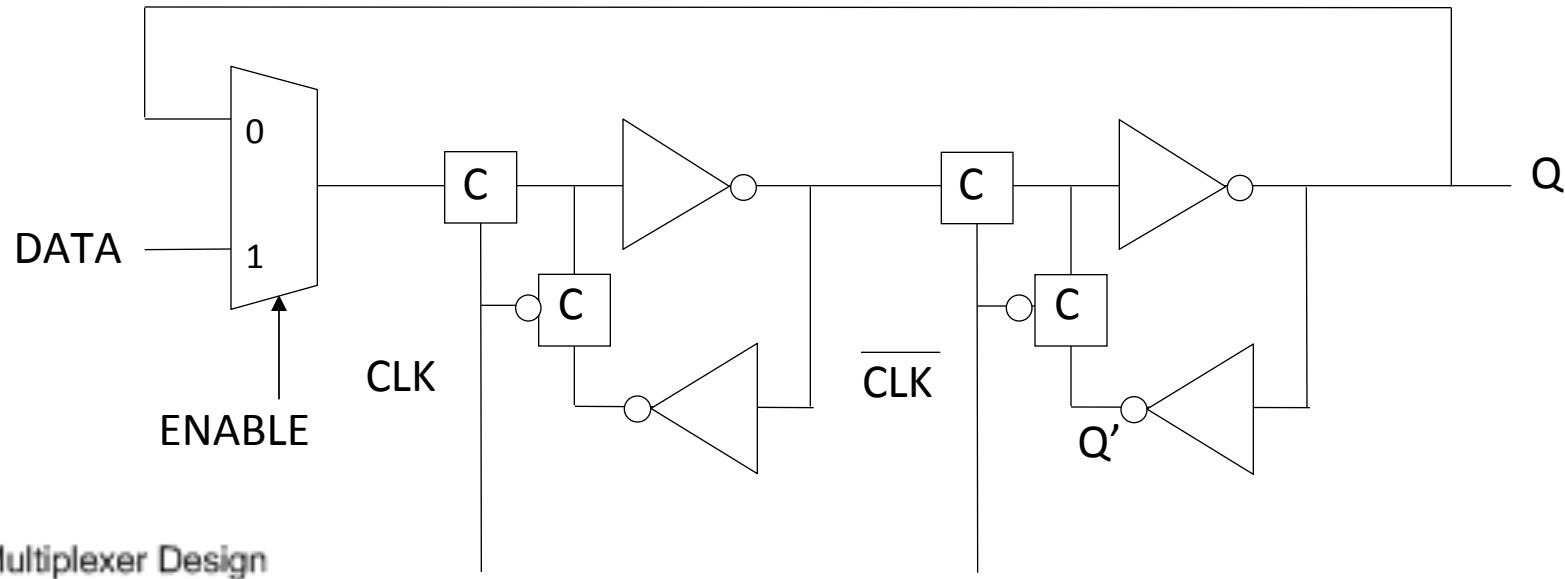
- New data is received by master when  $CLK=0$ , while slave stores previous data

# Enable

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



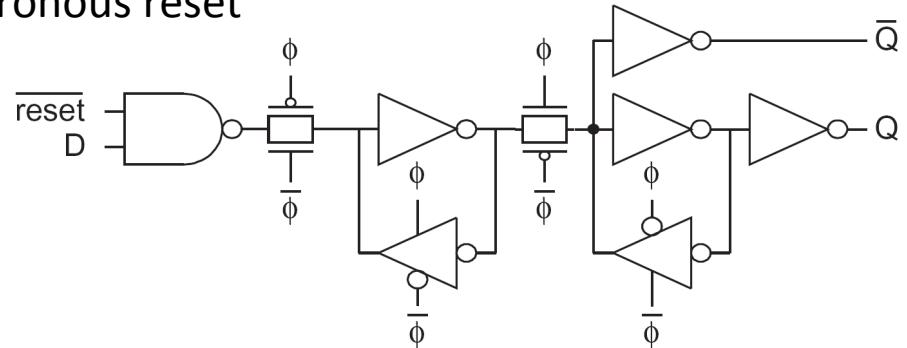
# Enable circuit implementation



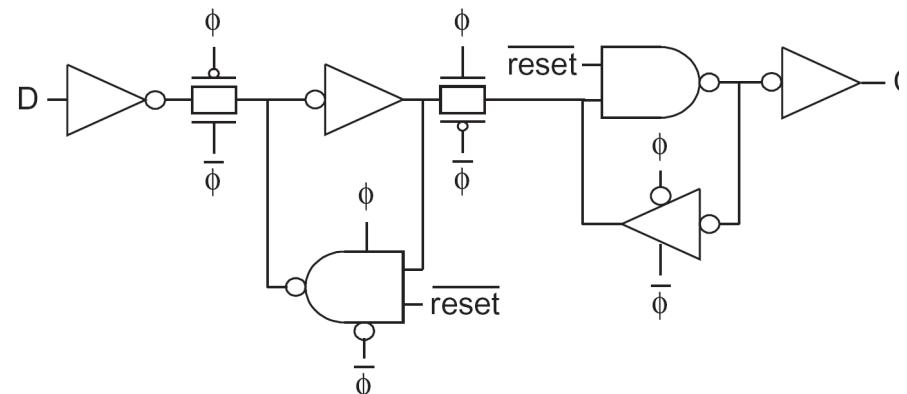
# Reset

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

## Synchronous reset

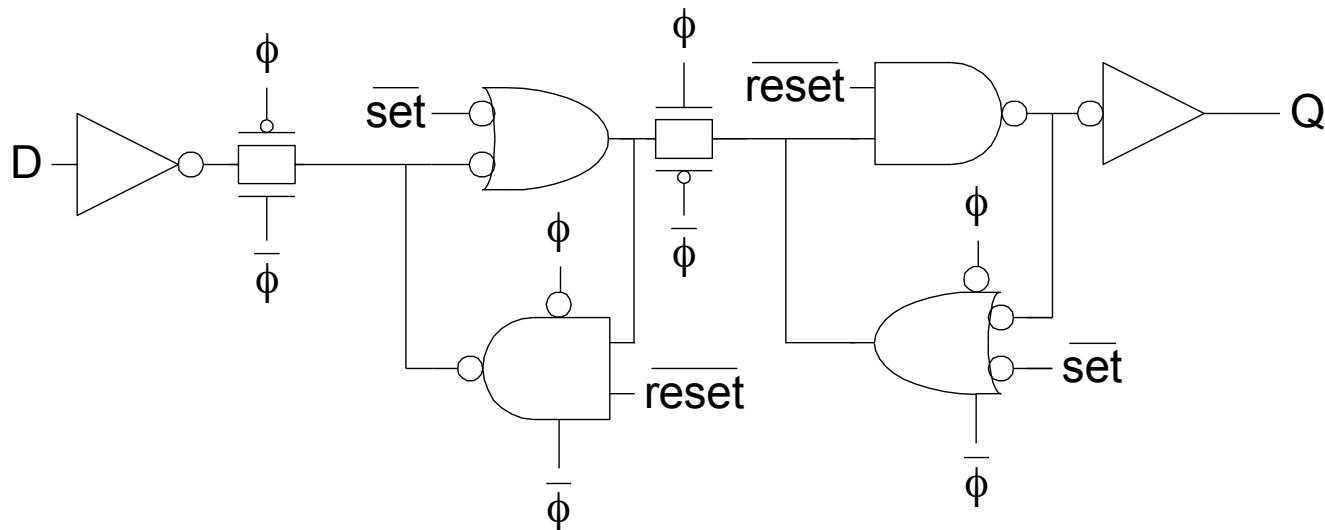


## Asynchronous reset



# Set/Reset

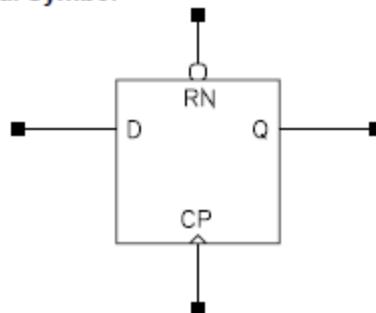
- Set forces output high when enabled
- Flip-flop with asynchronous set and reset



# HS65\_LS\_DFPRQ

## HS65\_LS\_DFPRQ

### Logical Symbol



### Cell Description

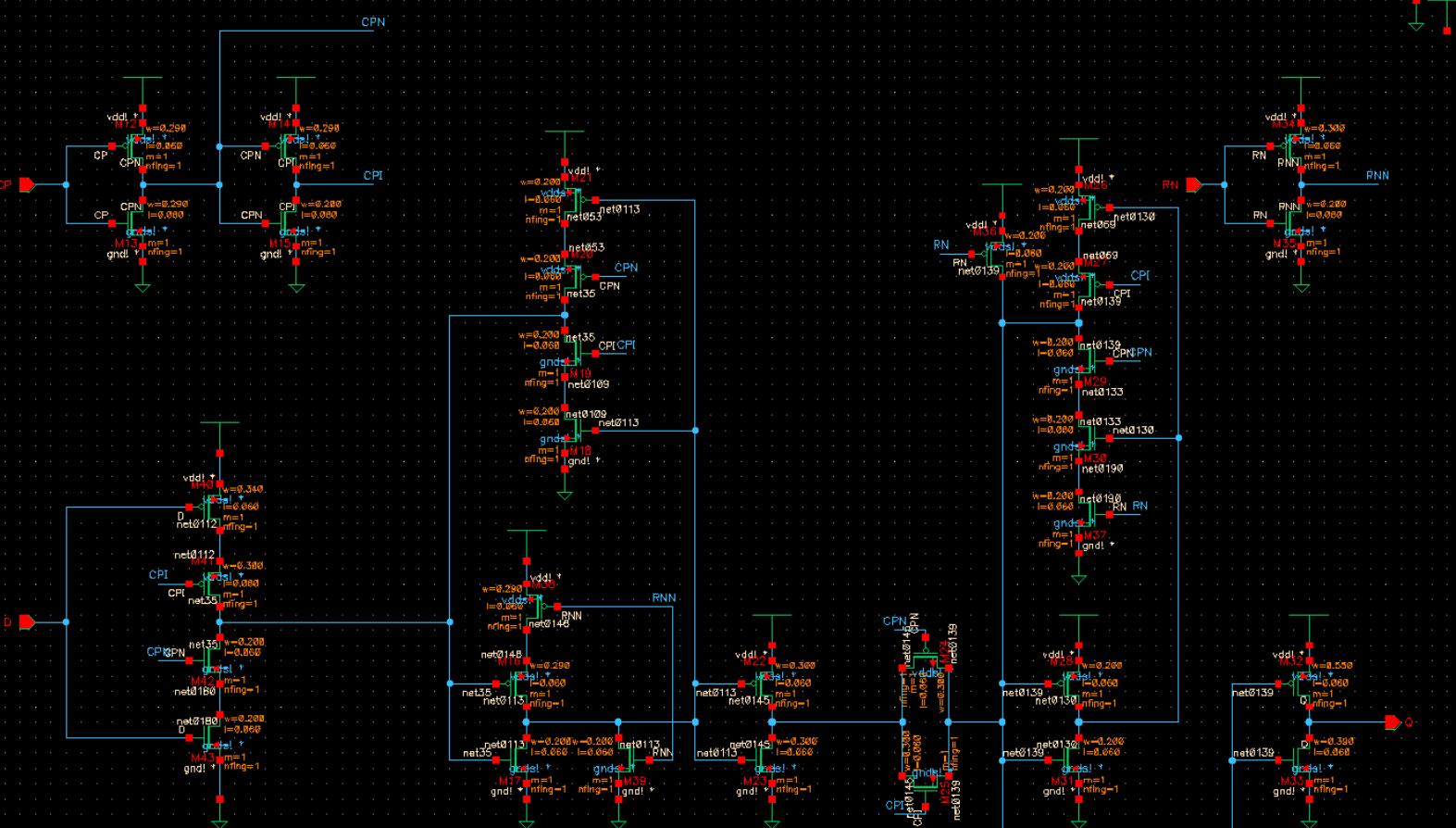
D-Type Flipflop; 1 Phase Positive Edge Triggered;  
Active Low Reset; Q Output Only  
The cell has "dont\_use" attribute set in the Synopsys STF.

### Functions

D	CP	RN	IQ	IQ
-	-	0	-	0
D	/	1	-	D
-	-	1	IQ	IQ

### Cell Size

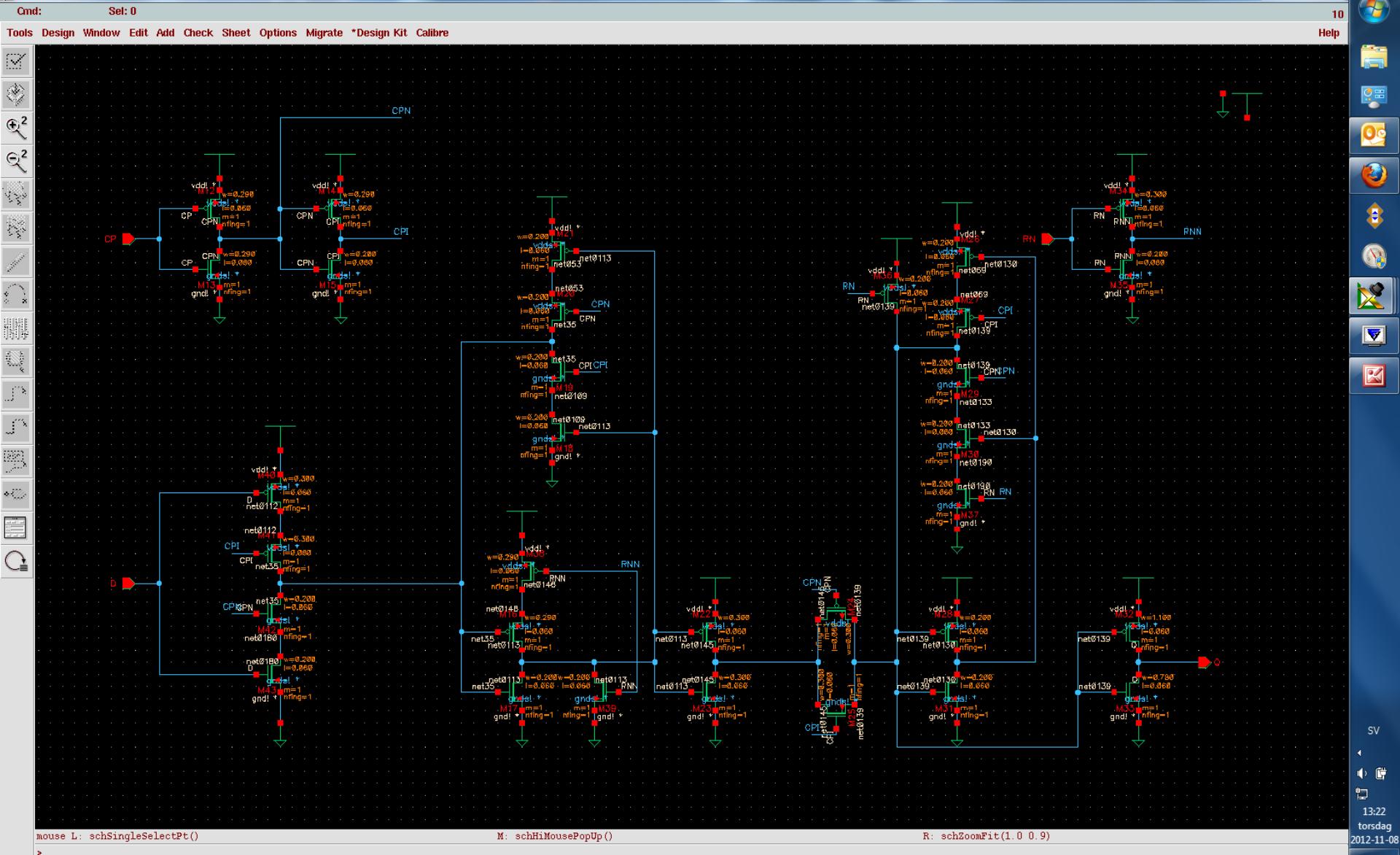
Drive Strength	Height (um)	Width (um)	Area (um <sup>2</sup> )
X4	2.6	4.00	10.4000
X9	2.6	4.00	10.4000
X18	2.6	4.20	10.9200
X27	2.6	4.60	11.9600
X35	2.6	4.80	12.4800



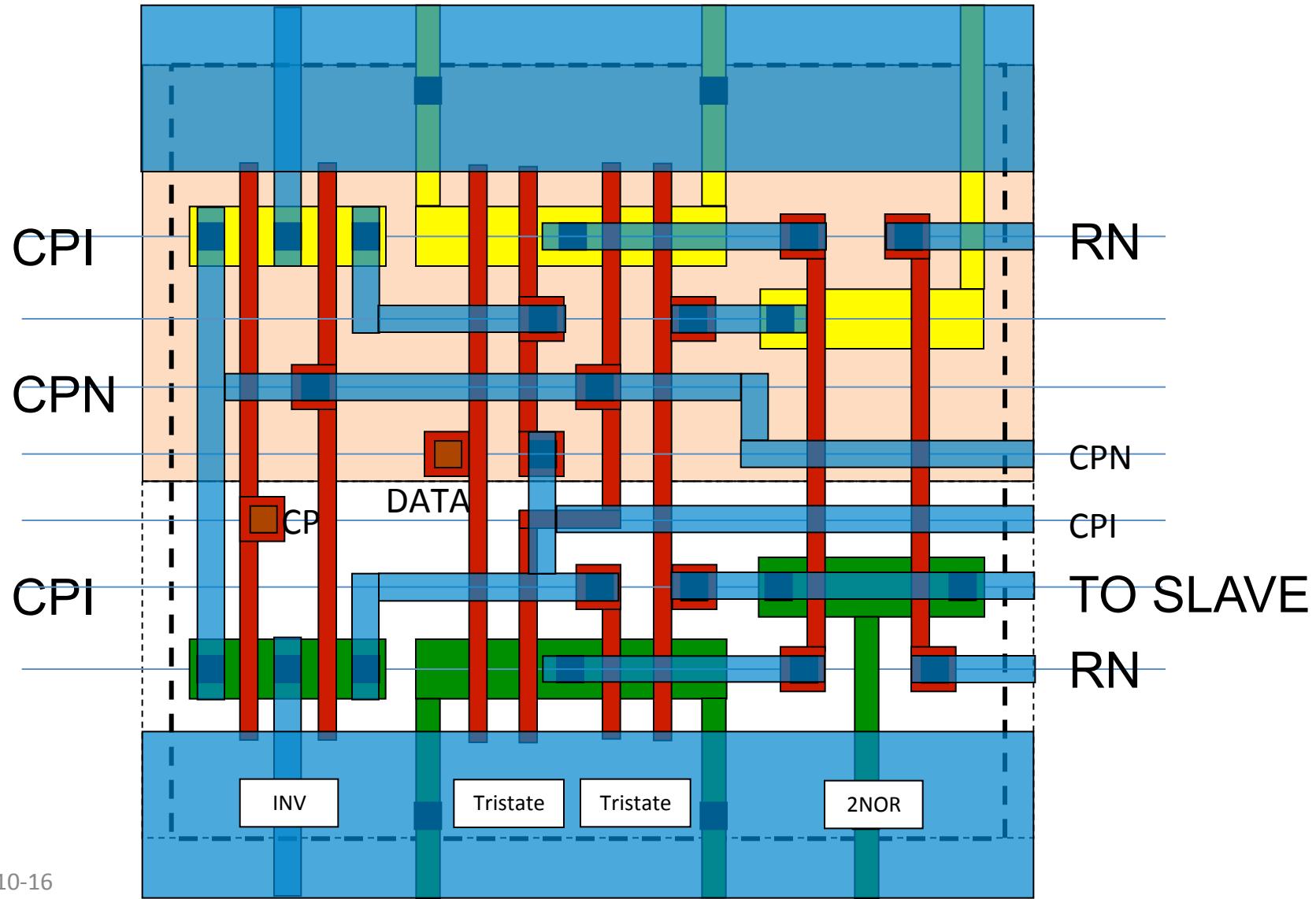
mouse L: schSingleSelectPt()

M: schHiMousePopUp()

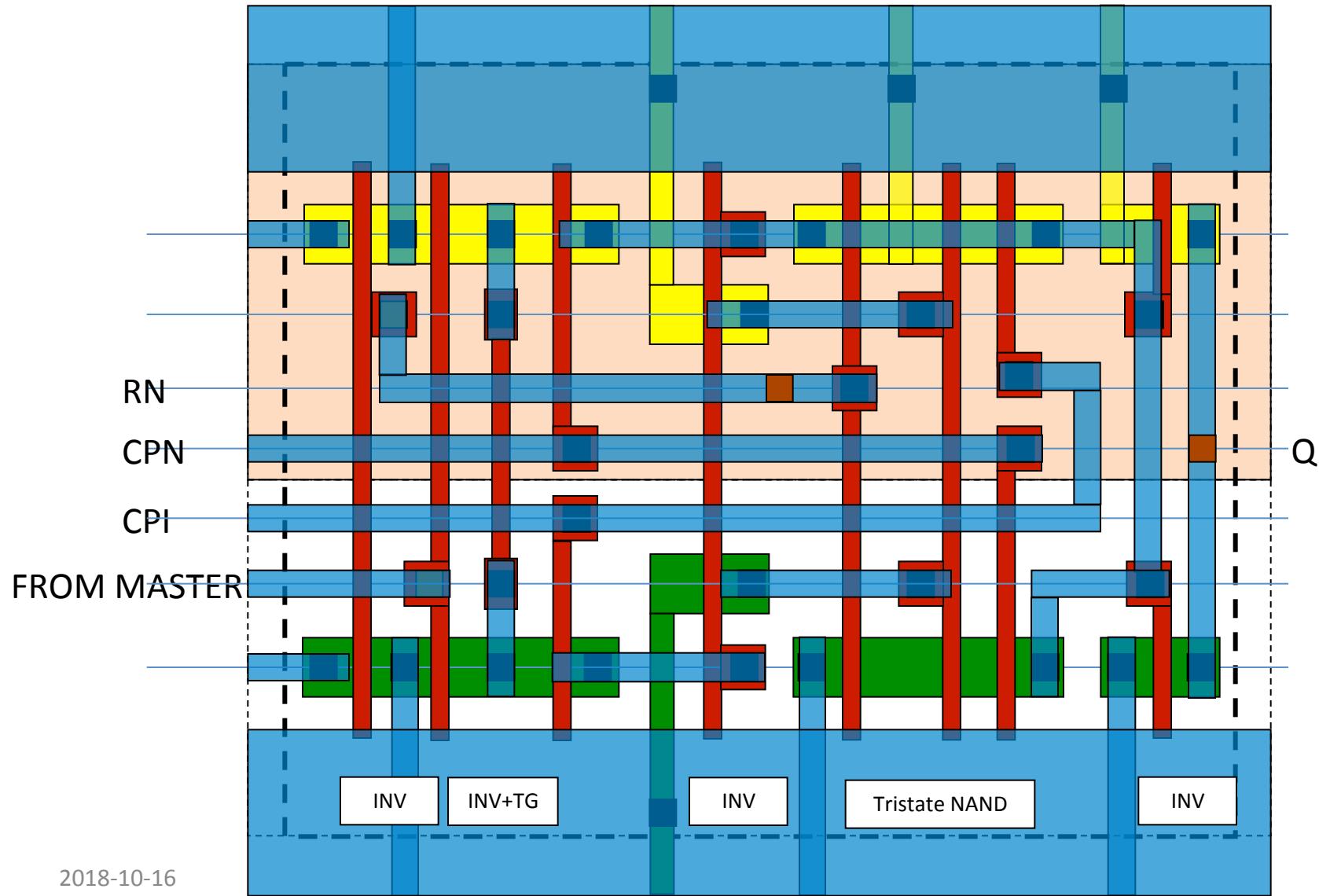
R: schZoomFit(1.0 0.9)



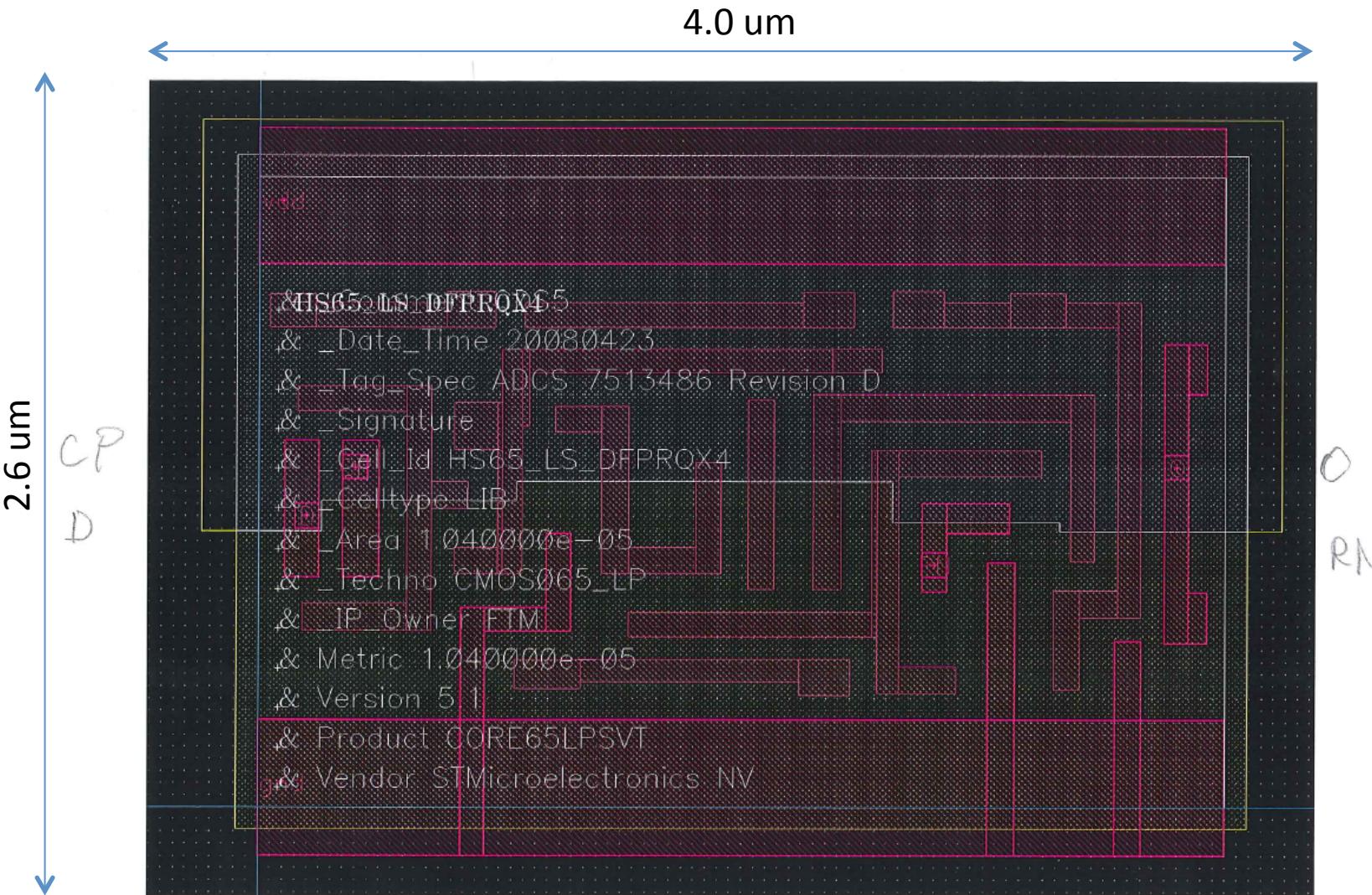
# Master latch 2.6 um x 2.8 um



# Slave latch 2.6um × 3.2um



# DFF layout in cell library



# Summary

- We took a step from combinatorial logic to sequential logic:
  - CL delays
  - FF delays
- Setup and hold violations:
  - Constraints on CL delays from  $f_c$ , flip-flop delays and clock skew
- System tradeoffs
- Edge-triggered flip-flop from two back-to-back latches: one master, one slave
  - Enable
  - Set/Reset
  - Scan chains
- Had a look at a D-type FF in STMicroelectronics cell library
- Compared to a master/slave design using our design template.
  - Results: 6.0  $\mu\text{m}$  wide compared with ST 4.0  $\mu\text{m}$
- Next up: metastability