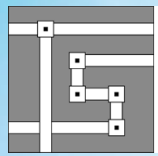


TRILOBYTE SYSTEMS

snug
Synopsys Users Group

Synchronization and Metastability

Steve Golson
Trilobyte Systems

March 24, 2014
SNUG Silicon Valley

Outline

Preface

Introduction and review

The rest of the story

Recommendations

Common fallacies

Methodology

Conclusions

Preface

or, wisdom from the experts

What do these have in common?

- LINC
- DEC PDP-11/45
- DEC PDP-15
- Space Shuttle PASS
- Zilog Z-80 SIO
- Honeywell 516 ARPANET IMP
- AMD Am29000
- AMD 9513
- AMD 9519
- Intel 8048
- Intel 8202
- Intel 8085
- Synopsys DW04_sync

What do these have in common?

- LINC
- DEC PDP-11/45
- DEC PDP-15
- Space Shuttle PASS
- Zilog Z-80 SIO
- Honeywell 516 ARPANET IMP
- AMD Am29000
- AMD 9513
- AMD 9519
- Intel 8048
- Intel 8202
- Intel 8085
- Synopsys DW04_sync

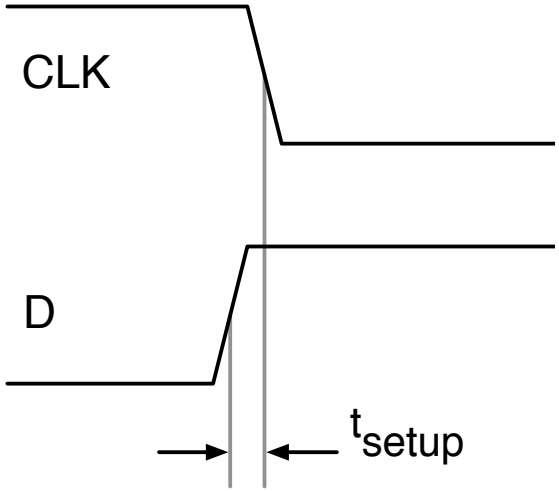
All of these had synchronizer failures!

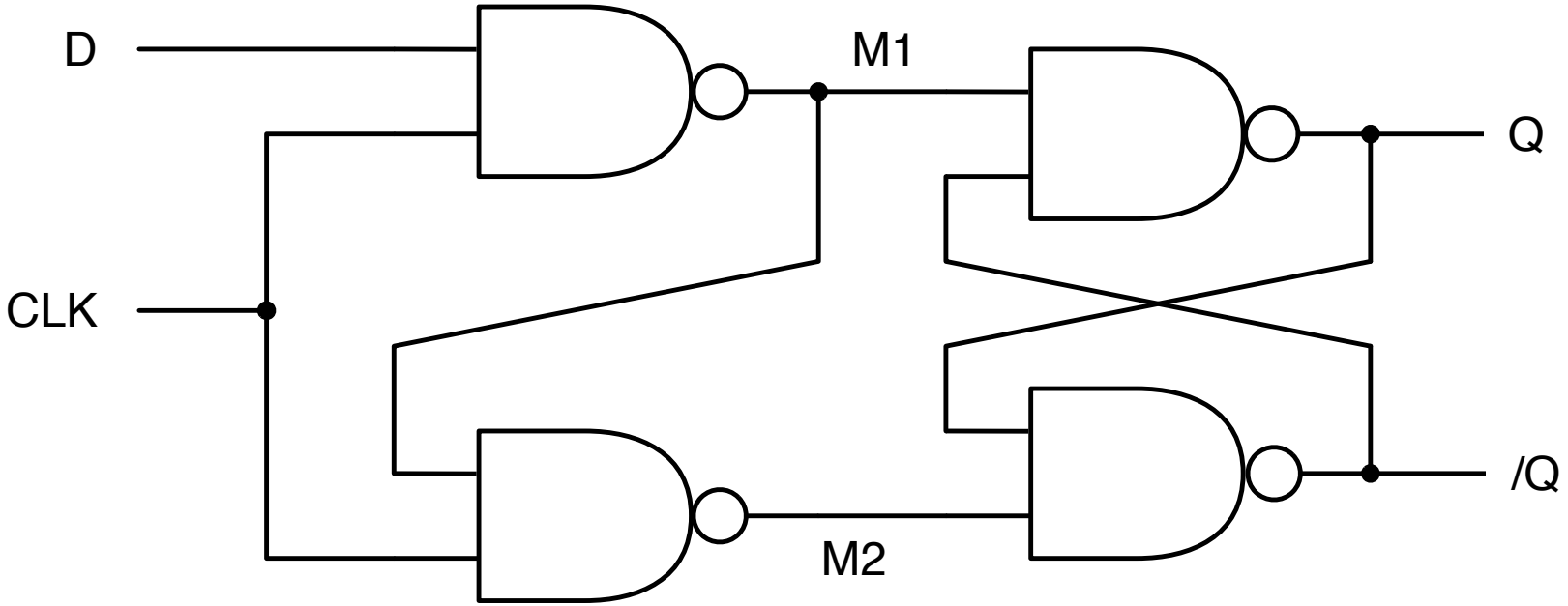
Introduction and review

or, what you learned as an undergrad EE student

Simple D latch

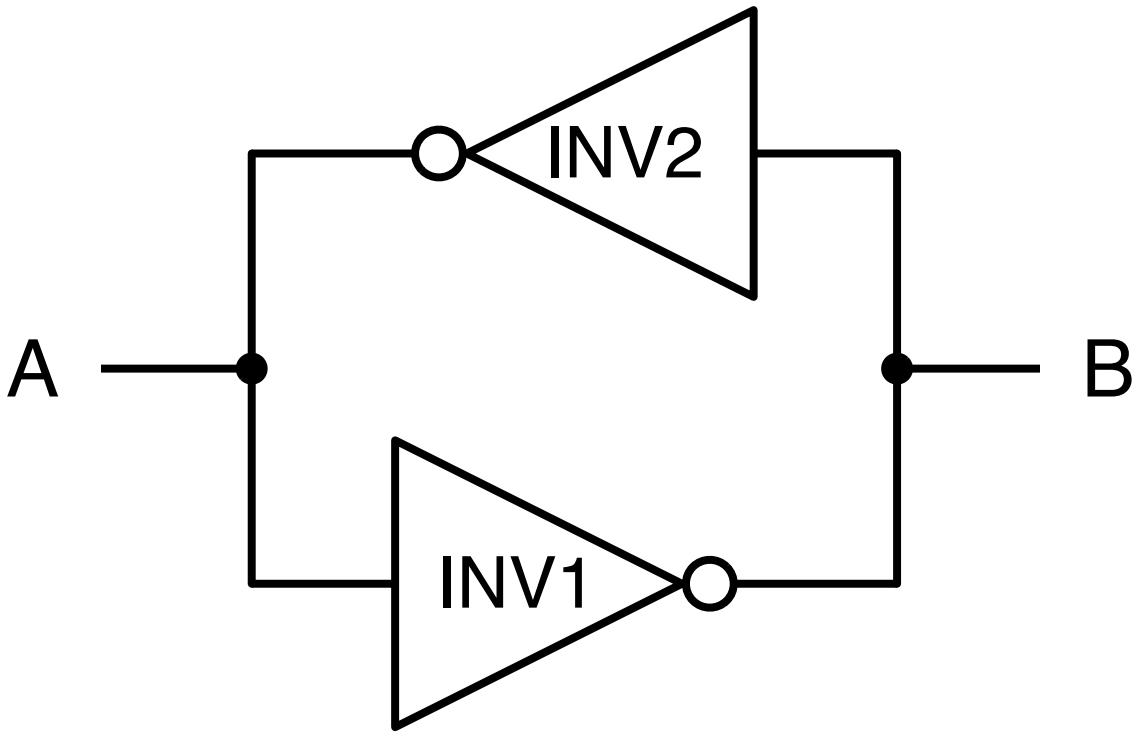
Bistable circuit



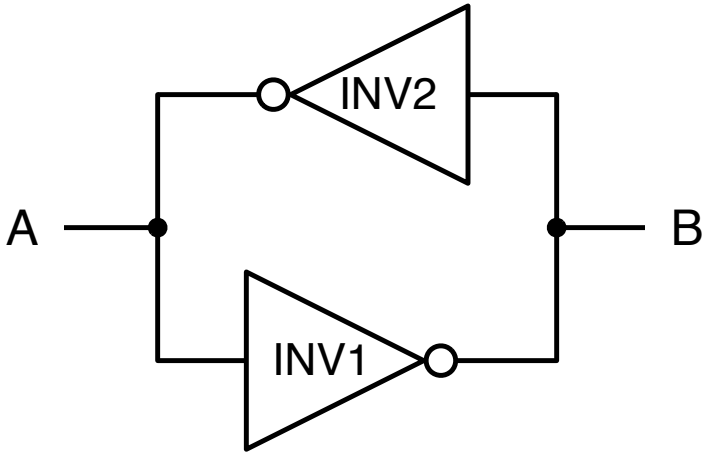


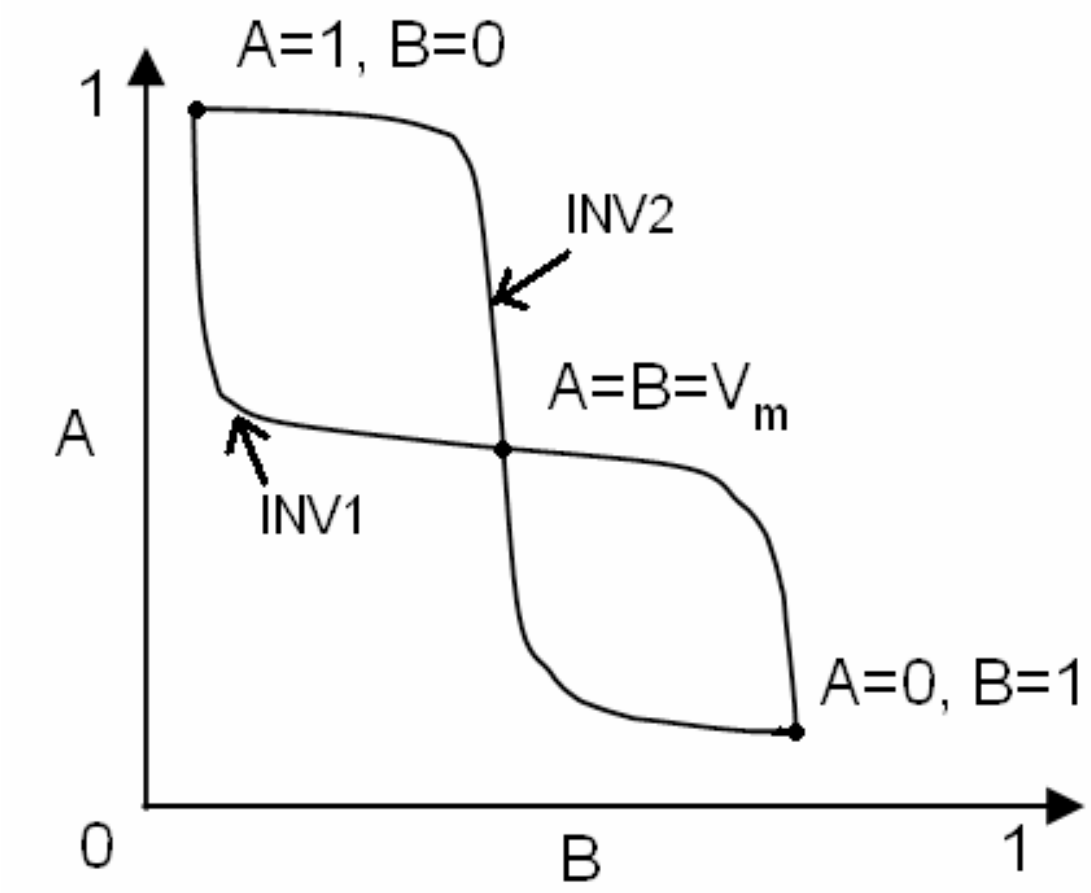
Cross-coupled inverters

Simple D latch after CLK goes low — Bistable circuit



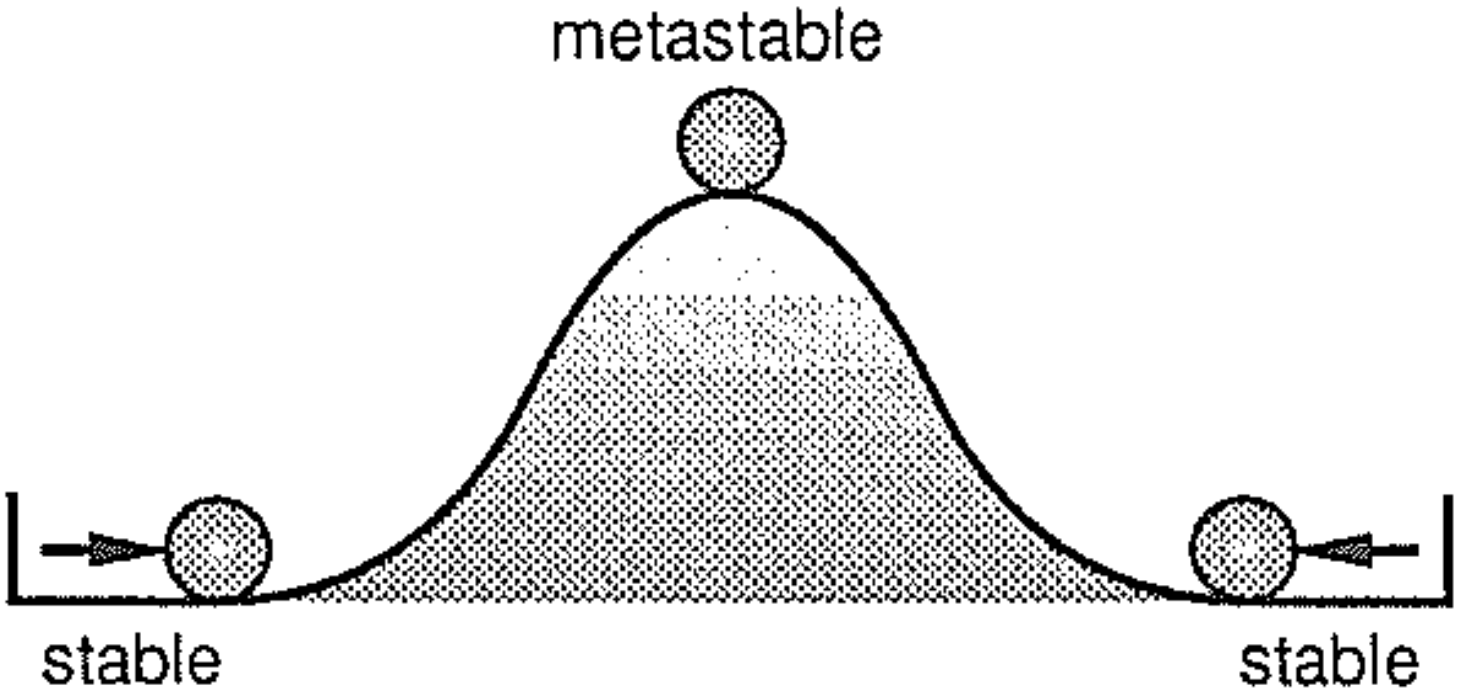
Two stable solutions
One *metastable* solution





[from Zhou 2008 fig. 2.6]

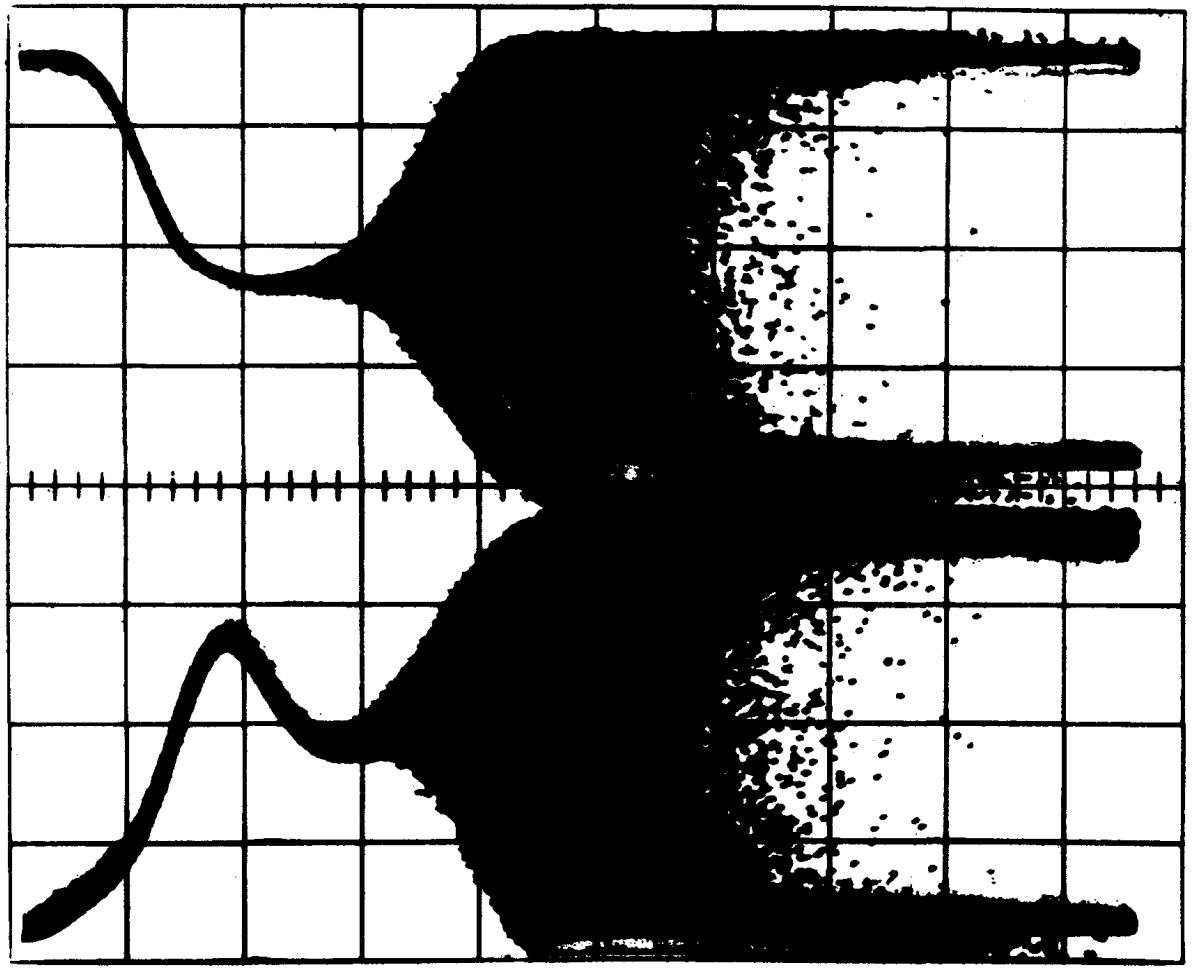
Two stable solutions
One *metastable* solution



[from Wakerly 1987 fig. 5]

Metastability in a D flip-flop

Sampling oscilloscope trace showing Q and /Q



[from Chaney and Molnar 1973 fig. 1]

Probability

$P(\text{metastable}) = P(\text{enter metastability}) \times P(\text{still in state after } t_R)$

$$P_E = f_c T_0 \qquad P_S = e^{-t_R/\tau}$$

$$P_F = P_E P_S = f_c T_0 e^{-t_R/\tau}$$

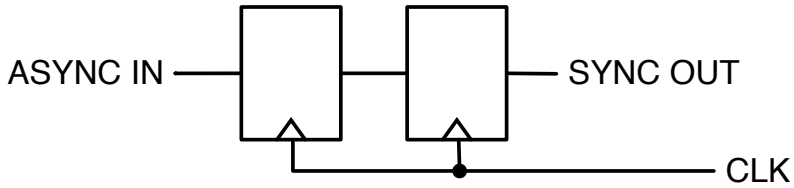
$$\lambda = f_d P_F = f_d f_c T_0 e^{-t_R/\tau}$$

Mean time between failures

$$MTBF = \frac{e^{t_R/\tau}}{f_d f_c T_0}$$

Guidelines from your professor

- Use two flops to build a *synchronizer*



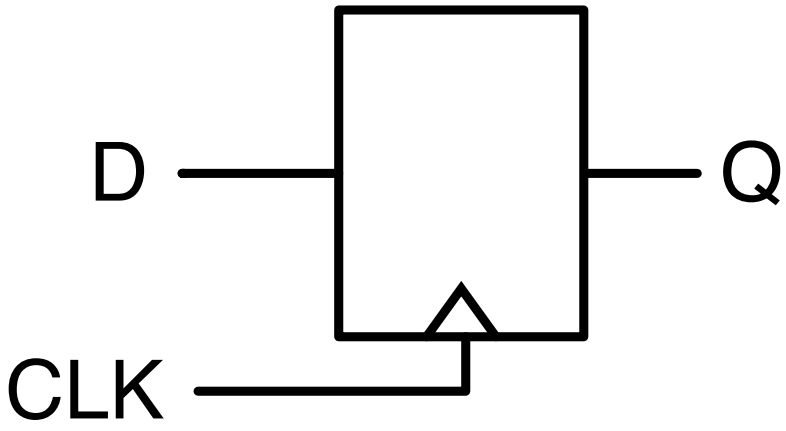
- No logic in between the flops
- ASYNC IN signal must be driven from a flop (clean edge)
- Synchronize a signal at only one point
- Do not synchronize multi-bit buses
  - instead, synchronize a single-bit control signal, that indicates when the bus is stable

The rest of the story

or, things you didn't learn as an undergrad EE student

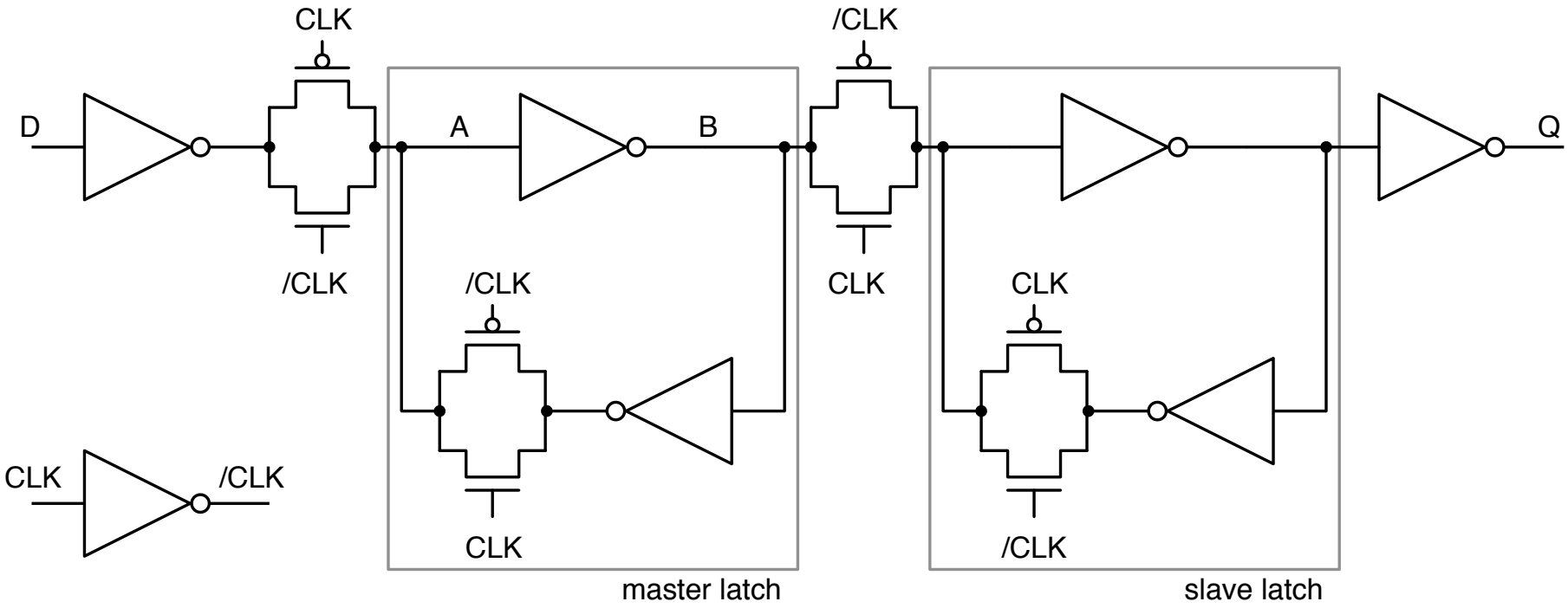
How many bistable devices?

Edge-triggered D flip-flop



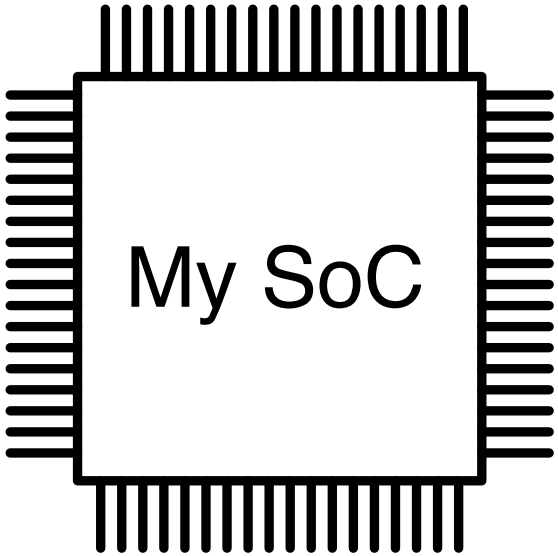
Two latches

D flip-flop internal structure



- τ for flop depends on τ for each latch, clock duty cycle, propagation delay between latches, process variation…

How many synchronizers?



MTBF for *N* synchronizers

$$MTBF(N) = \frac{1}{\sum_{n=1}^{N} \frac{1}{MTBF_n}}$$

$$MTBF(N) = \frac{e^{t_R/\tau}}{N\, f_d\, f_c\, T_0}$$

How many synchronizers?

[from ExtremeTech.com 2011]

Reliability and failure

Things you need to do

1. Specify reliability goals for your product
   - what % failures are acceptable over product lifetime?
2. Find metastability parameters τ , T_0 for *all* synchronizers
3. Write an equation to calculate probability of failure
   - sales volume
   - lifetime of system
   - τ and T_0
   - clock frequencies
   - data rates
   - resolution times

Example reliability goal

- Given:
  - we plan to sell 100,000 of this chip
  - each chip contains 1,000 synchronizers
  - each chip is expected to operate for 5 years
- Then your reliability goal might be:

With probability 85%,
not a single synchronizer out of the entire population of 10^8
will have a metastability failure during its operating lifetime

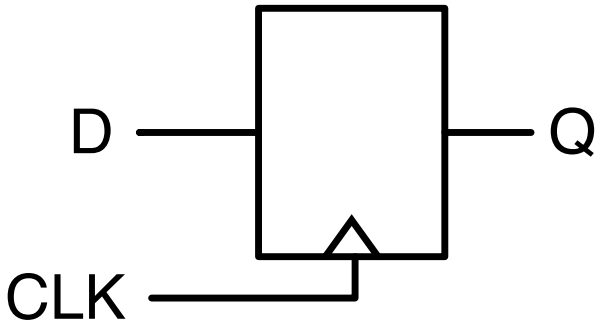
Where to get τ and T_0?

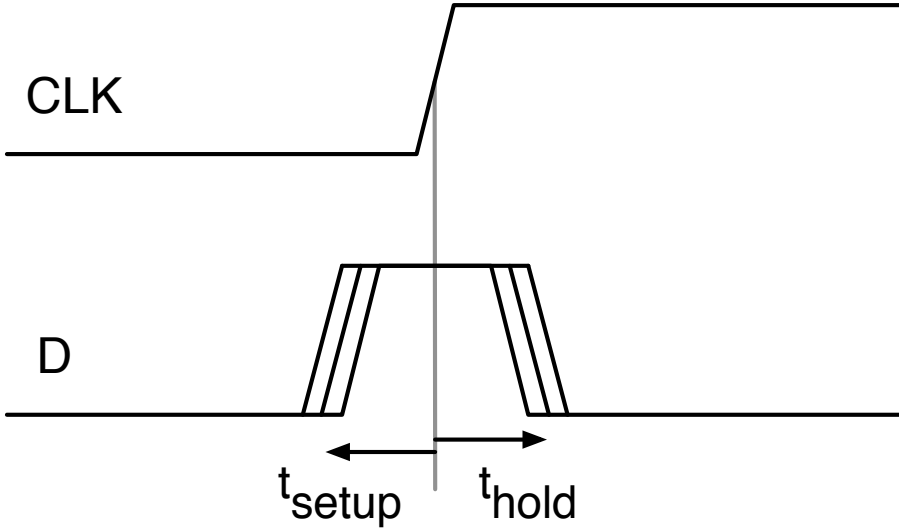
You need metastability parameters for every flip-flop

- Provided by library / fab vendors
- Measurement from actual silicon
  - tricky high-speed lab techniques
- Simulation
  - do you have SPICE?
  - do you have process models?
  - do you have netlists for your library cells?

Simulation setup

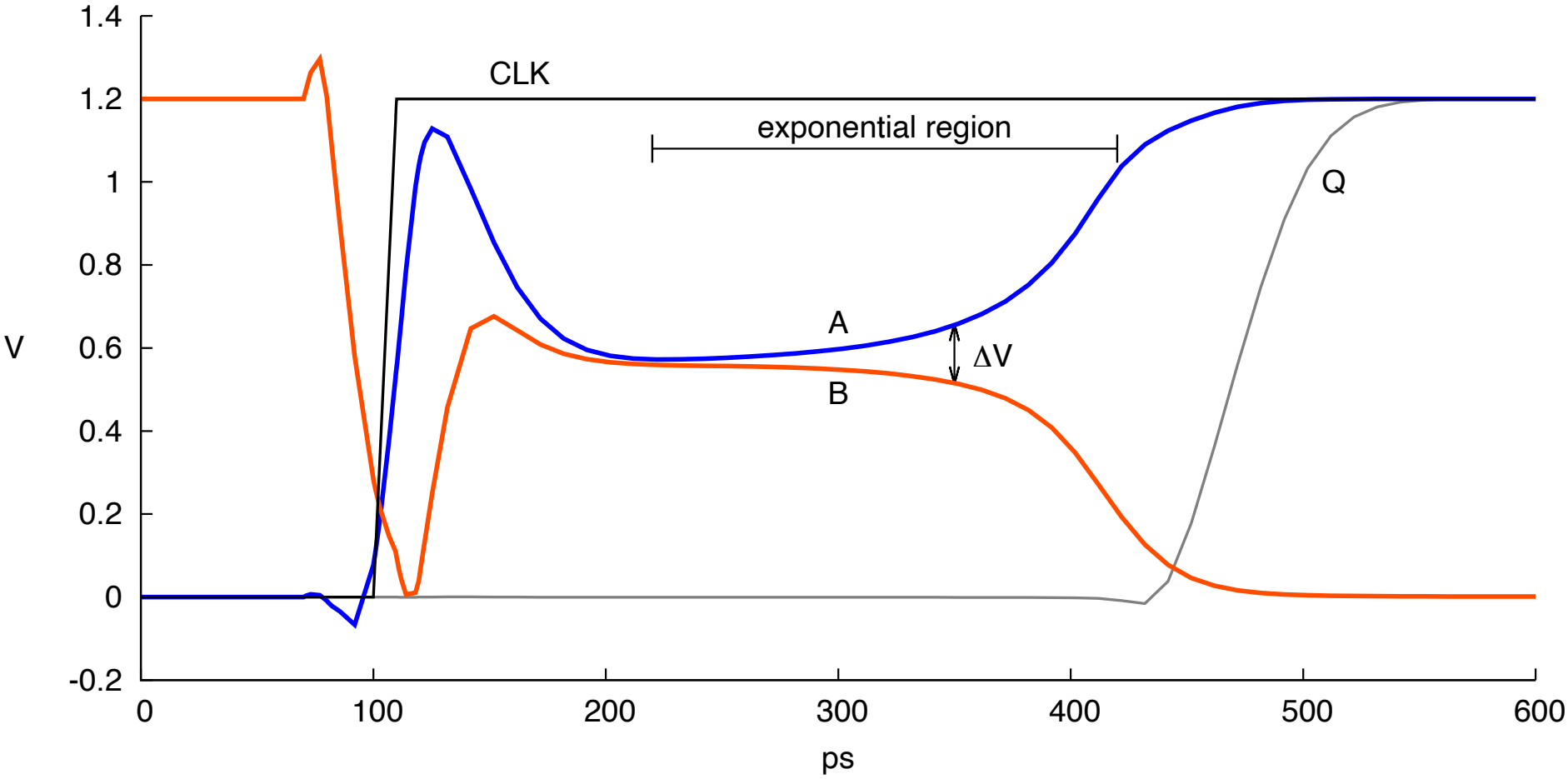
Simple D flip-flop





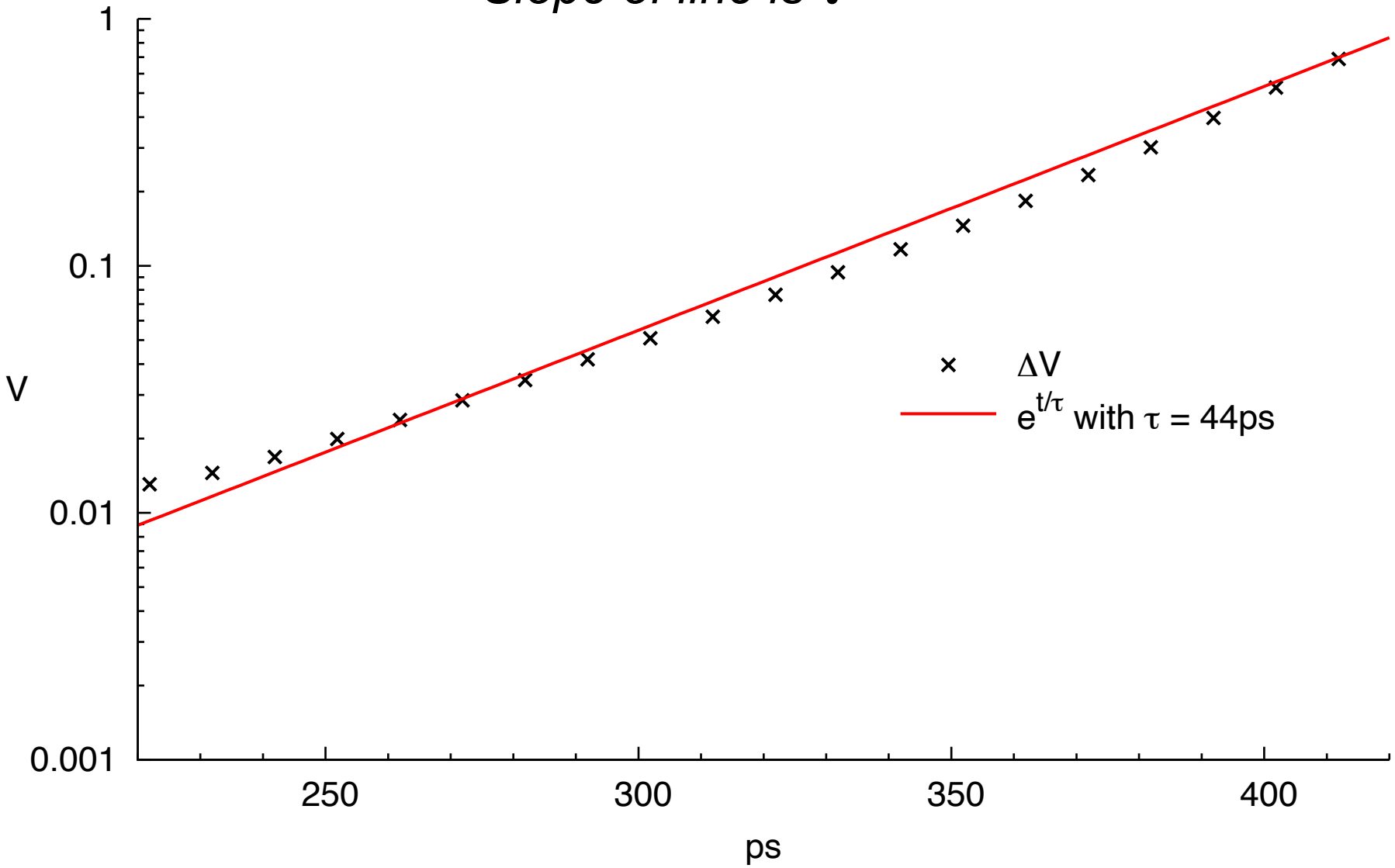
SPICE results

D flip-flop master latch, very small t_{hold} time



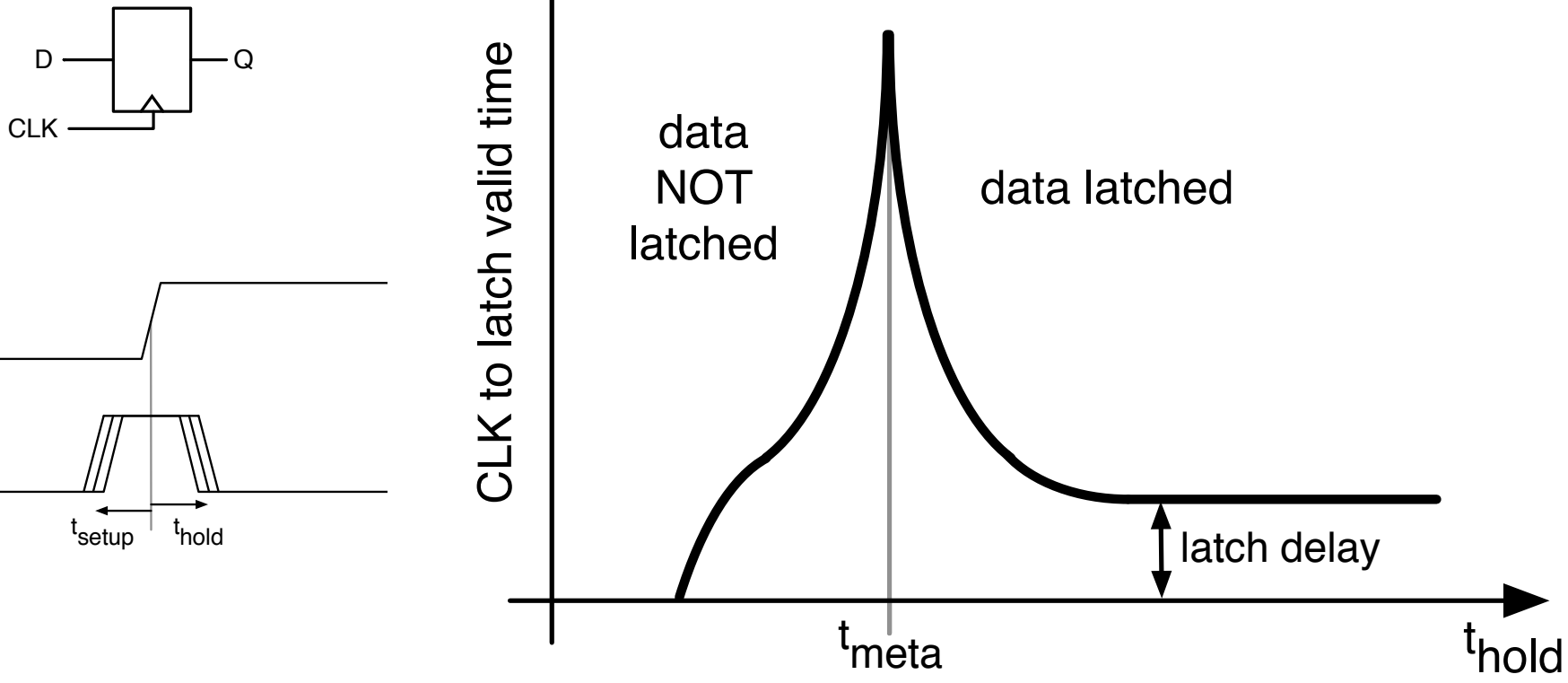
ΔV in exponential region

Slope of line is τ



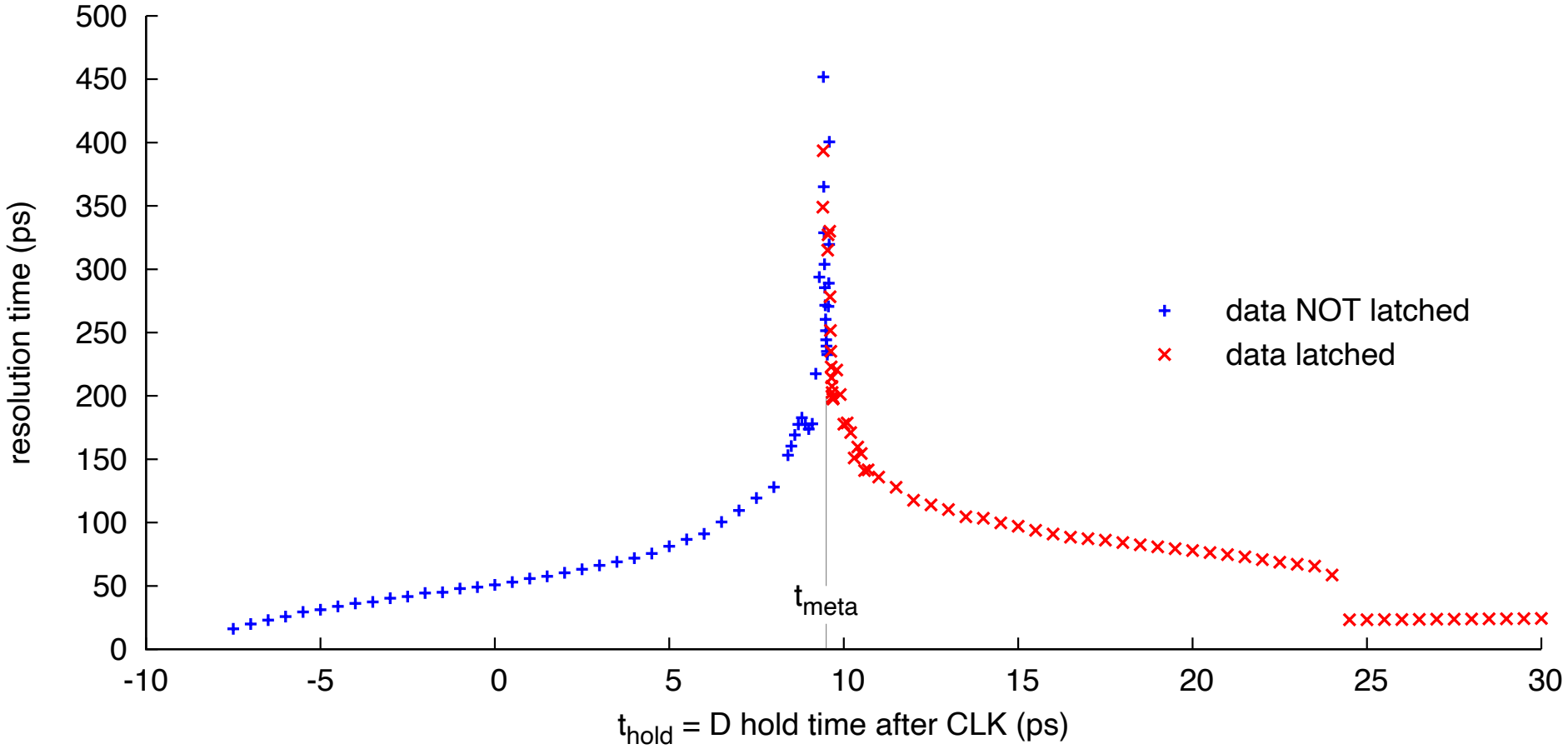
Another simulation technique

Multiple runs: fixed setup time, vary the hold time

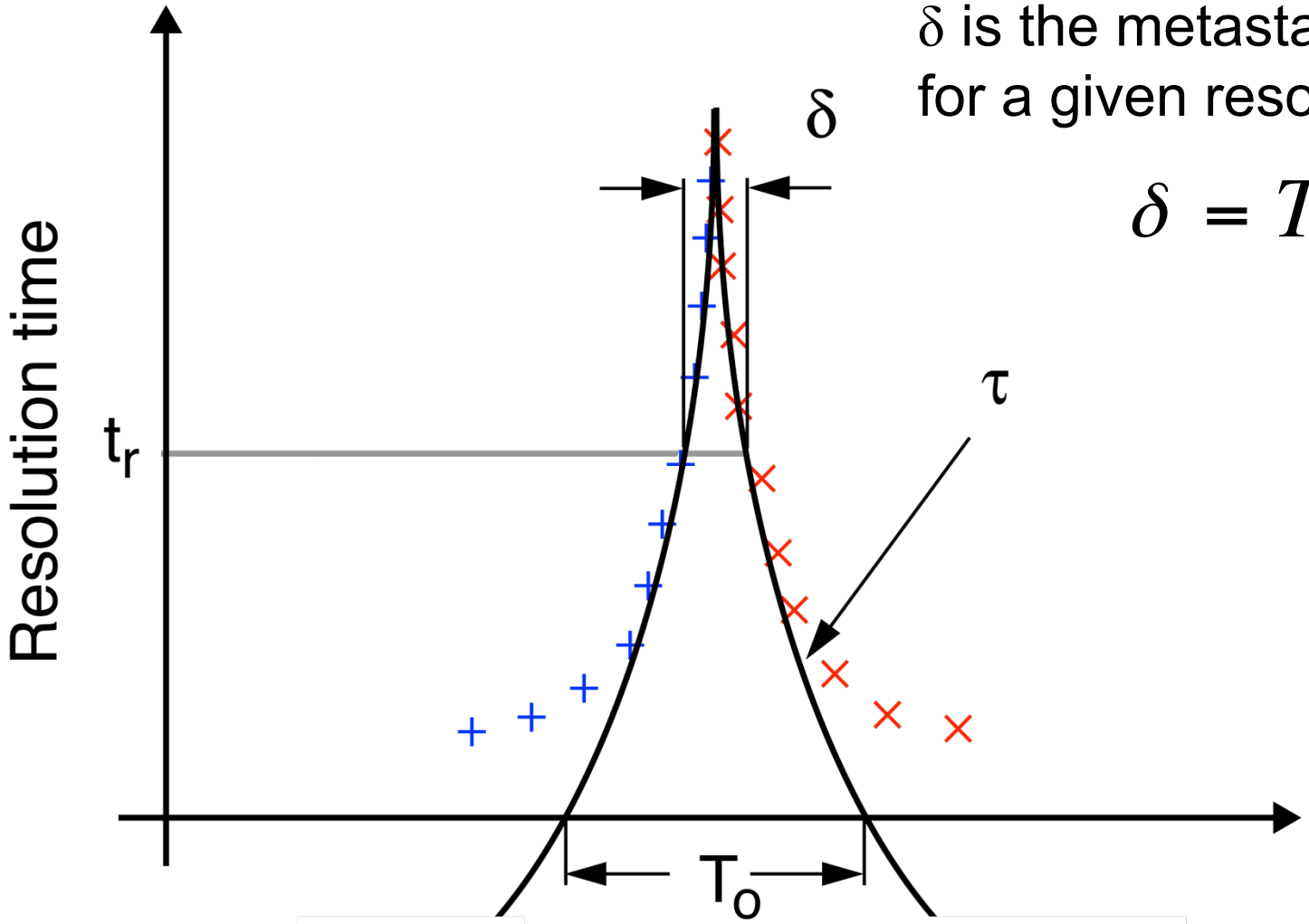


SPICE results

Fixed setup time, vary the hold time



Fit model to SPICE results



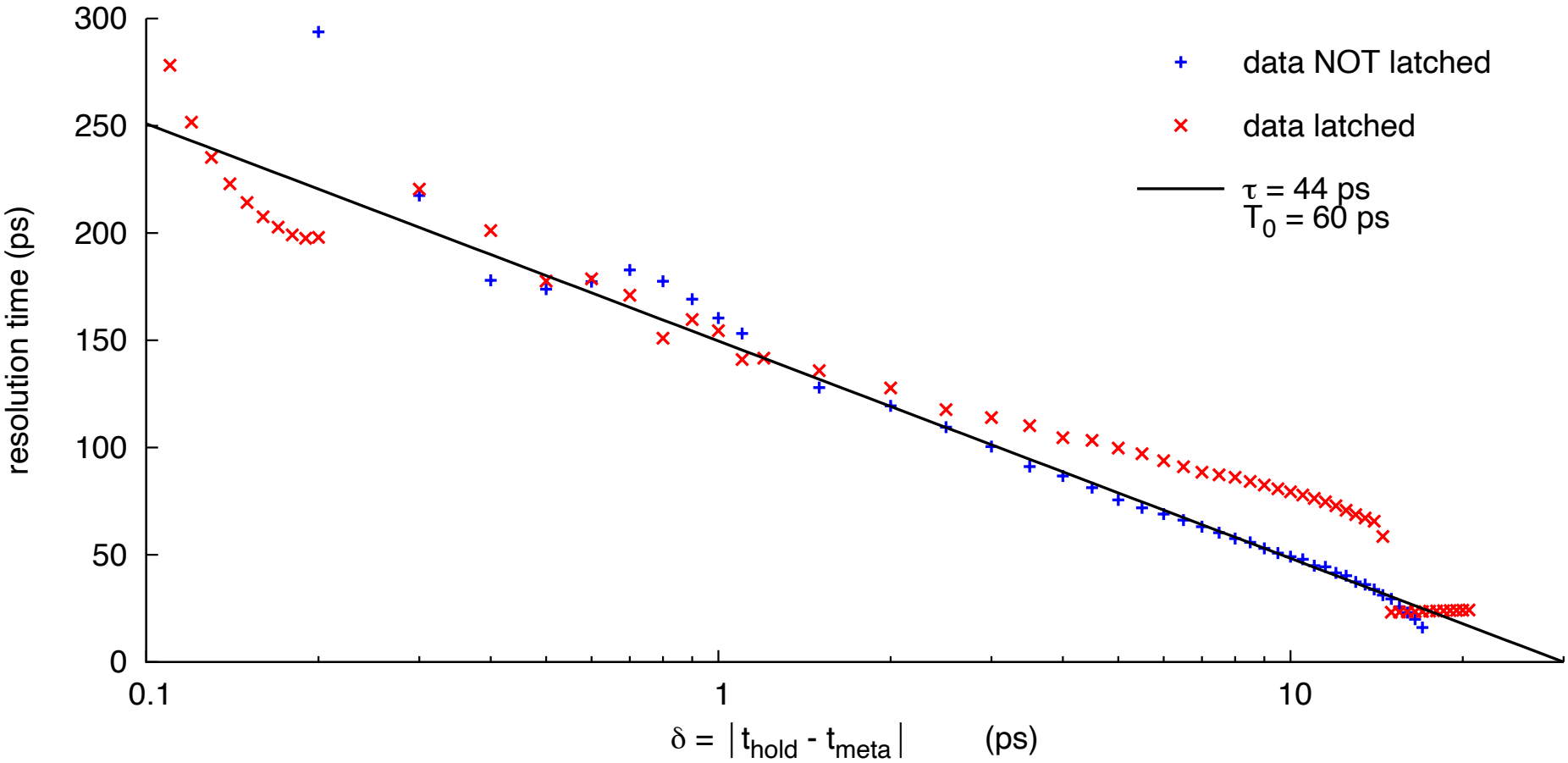
δ is the metastability window for a given resolution time t_R

$$\delta = T_0 e^{-t_R/\tau}$$

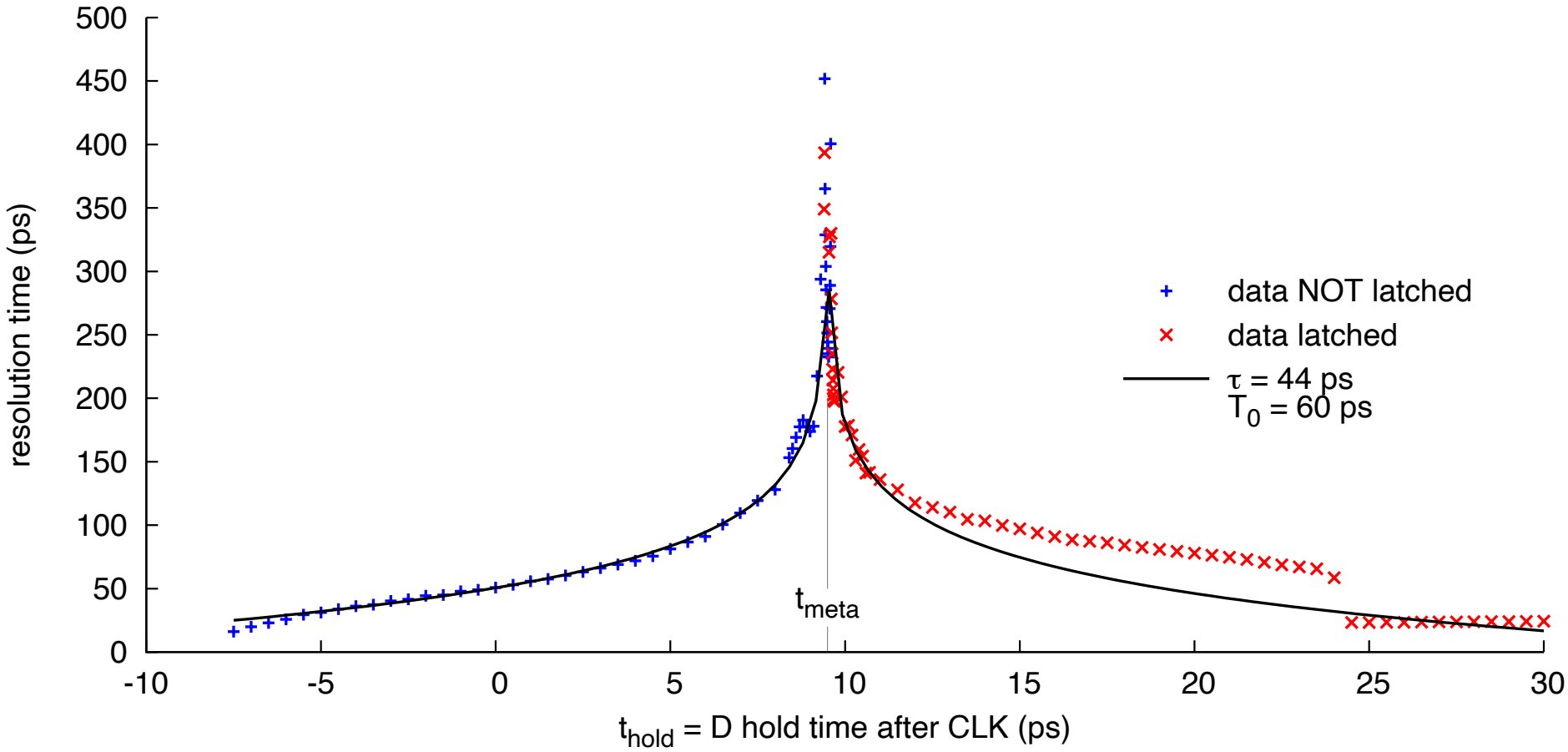
[from Portmann 1995 fig. 2.9b]

Extract model from SPICE results

Slope is τ, intercept is $T_0/2$



SPICE results and model



Recommendations

or, how to improve your reliability

How to improve your reliability

$$MTBF = \frac{e^{t_R/\tau}}{f_d f_c T_0}$$

$$MTBF(N) = \frac{1}{\sum_{n=1}^{N} \frac{1}{MTBF_n}}$$

t_R = resolution time (s)

τ = time constant (s)

T_0 = metastability window (s)

f_d = data arrival rate (Hz)

f_c = clock sampling rate (Hz)

N = number of synchronizers in your system

Common fallacies

or, misconceptions your co-workers may have

Common fallacies

Mean time between failures

"Our required MTBF is ten years, because nobody owns a computer for longer than that."

Mean time between failures

Synopsys Users Group

After MTBF years,
how many of your systems have failed?

Mean time between failures

Synopsys Users Group

After MTBF years,
how many of your systems have failed?

63%

What is MTBF of SNUG attendees?

- Model our population as 2,000 males age 35 years
- Probability of failure (death) over the next year is 0.001612
- So we can expect

$$2{,}000 \times 0.001612 = 3.22 \text{ failures in one year}$$

MTBF(SNUG) = (2,000 units x 1 year operation)/3.22 failures

What is MTBF of SNUG attendees?

- Model our population as 2,000 males age 35 years
- Probability of failure (death) over the next year is 0.001612
- So we can expect

$$2{,}000 \times 0.001612 = 3.22 \text{ failures in one year}$$

MTBF(SNUG) = (2,000 units x 1 year operation)/3.22 failures

621 years

Mean time between failures?

- MTBF is *not* the lifetime of your part!
- MTBF is *not* service time!
- MTBF is *not* operating time!
- MTBF is *not* time to first failure!

MTBF is the inverse of failure rate λ

MTBF is inverse of failure rate

If we say

$$MTBF = 10 \text{ years}$$

then our failure rate is

$$\lambda = \frac{1}{MTBF} = \frac{1}{10 \text{ years}} = 0.10 \frac{1}{\text{year}}$$

We expect 10% to fail every year

Common fallacies

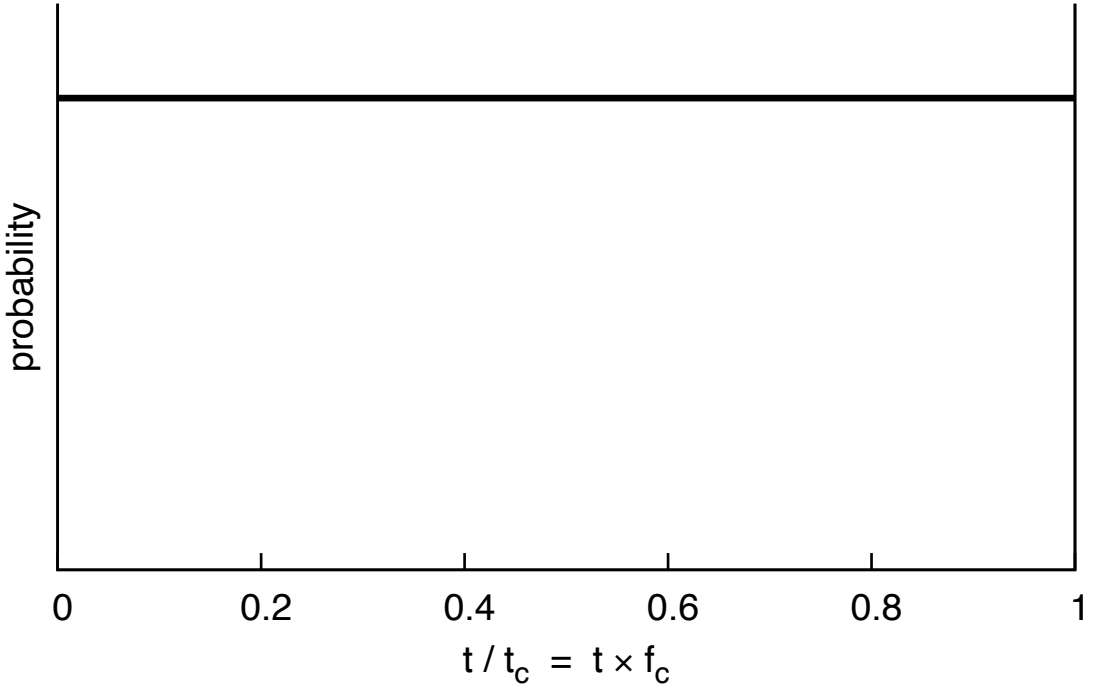
Coherent clocks

Synopsys Users Group

"Of course these two clocks are asynchronous."

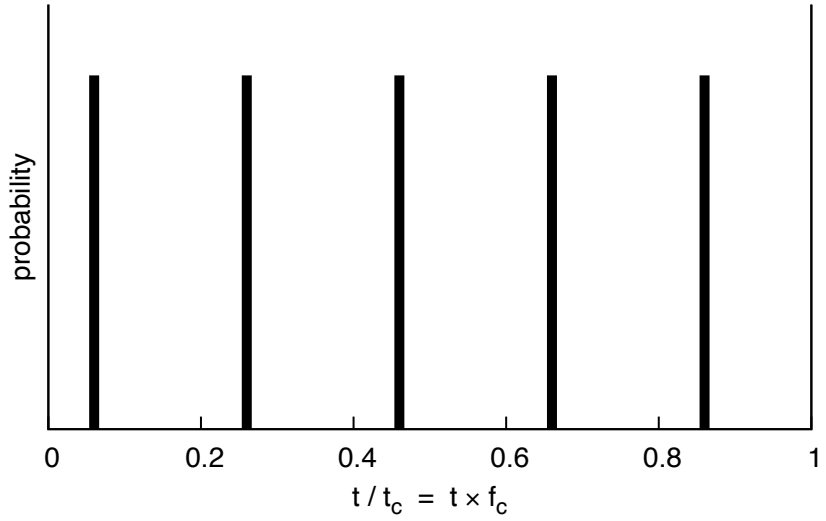
Asynchronous clocks

- Fundamental assumption of our MTBF calculation
  - data input and sampling clock are completely asynchronous
  - data can arrive *at any time* during the sampling clock period



Coherent clocks

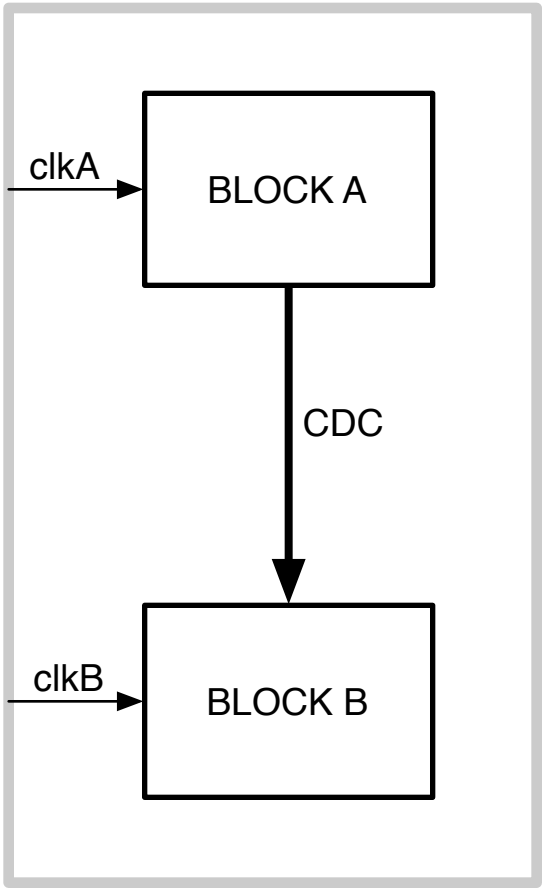
- If the clocks are *coherent* then
  - clocks are ultimately sourced from the same oscillator
  - data will arrive *at predictable locations* in the clock period



  - these locations might coincide with the metastability window…

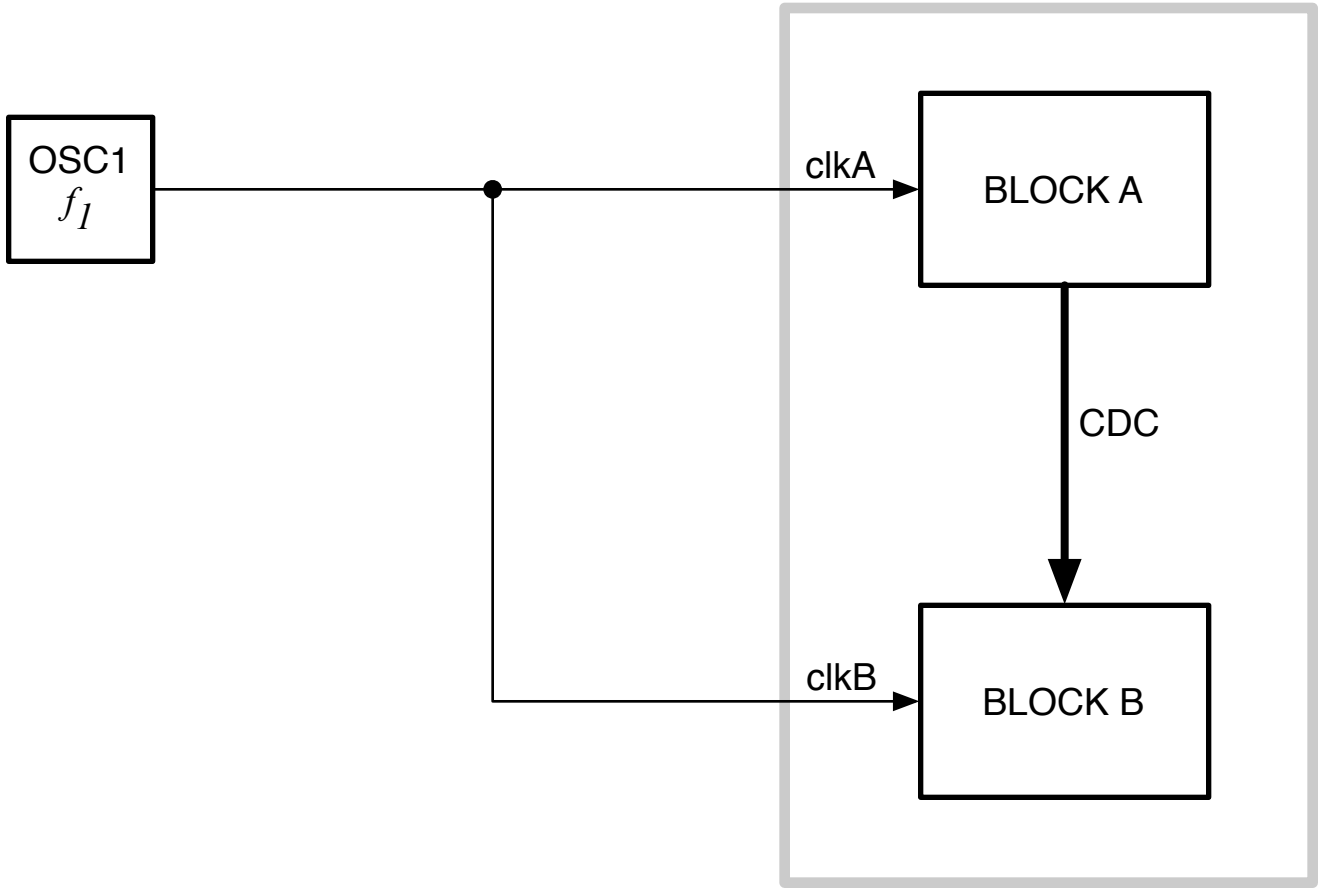
- MTBF can worsen by *several orders of magnitude*

Asynchronous clocks?



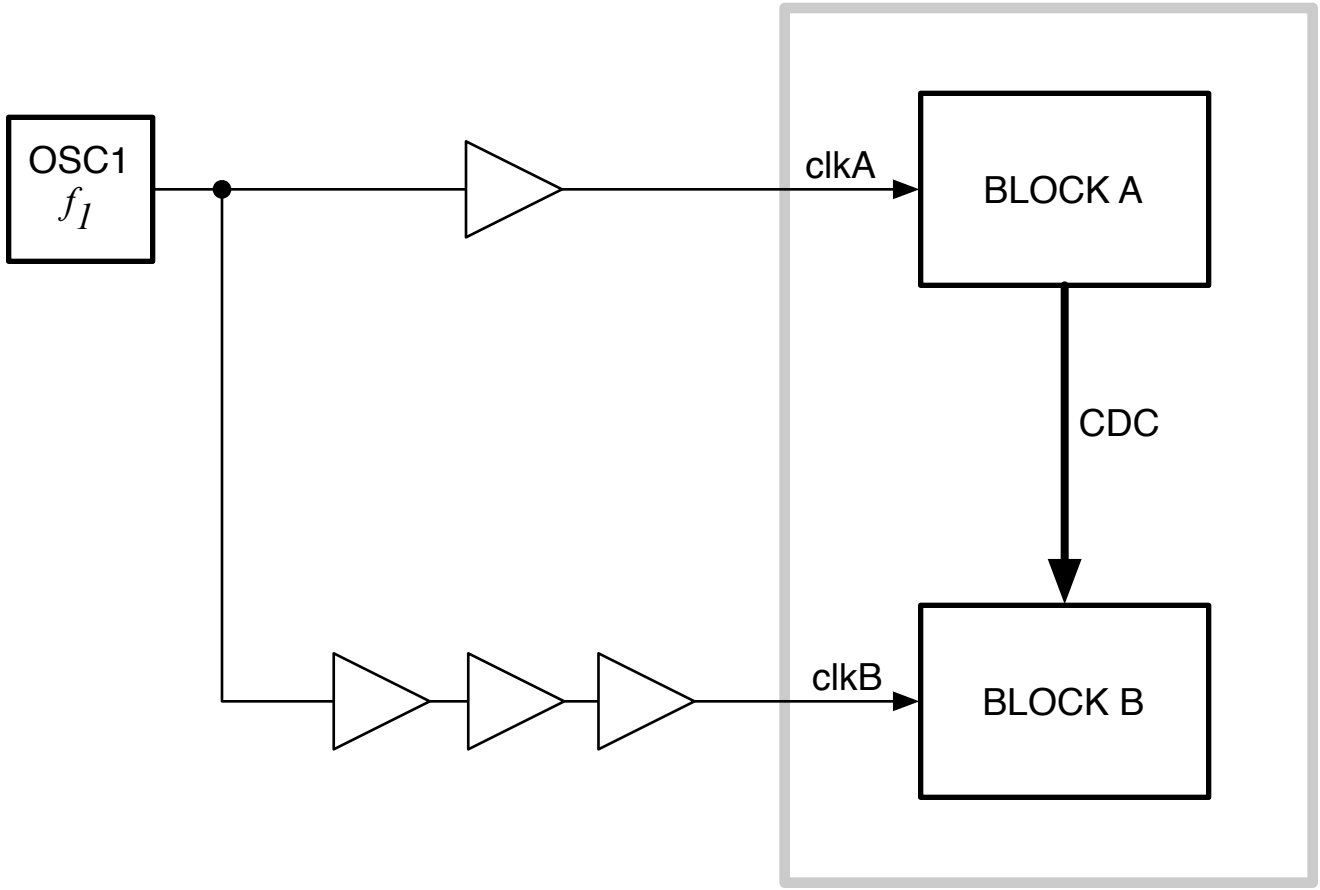
Synchronous clocks

Same source, same frequency, same phase



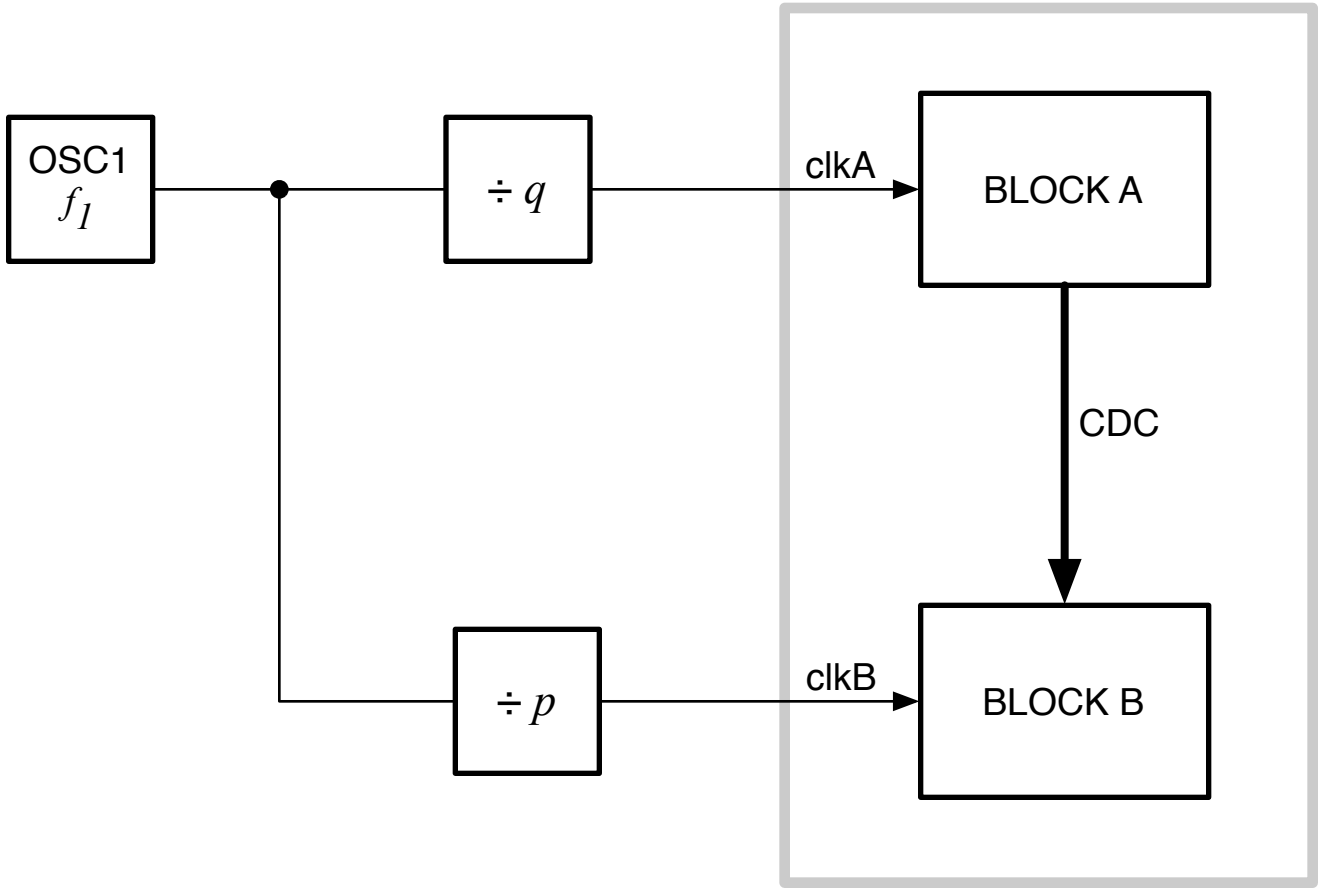
Mesochronous clocks

Same source, same frequency, different phase



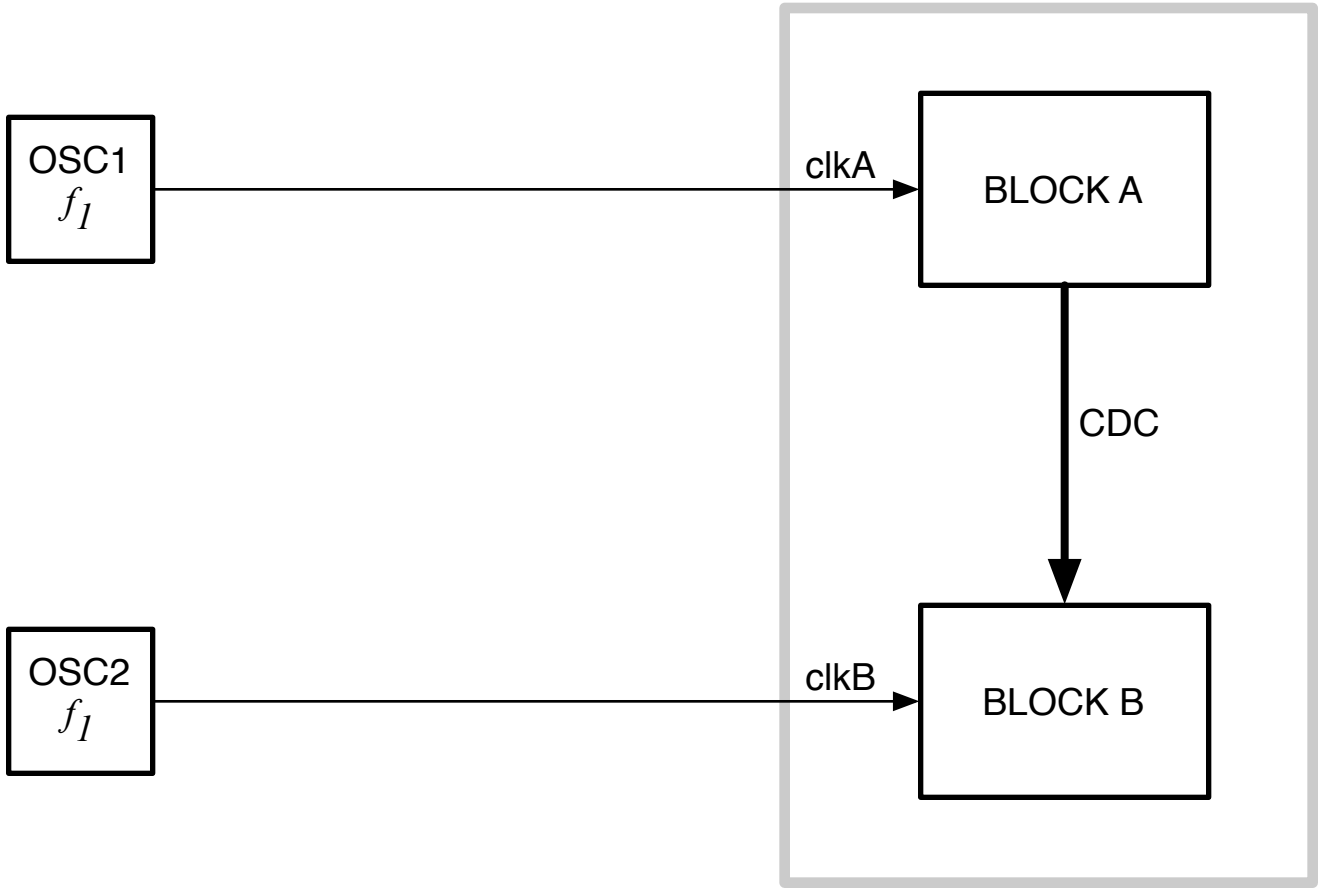
Rational clocks

Same source, rational frequency



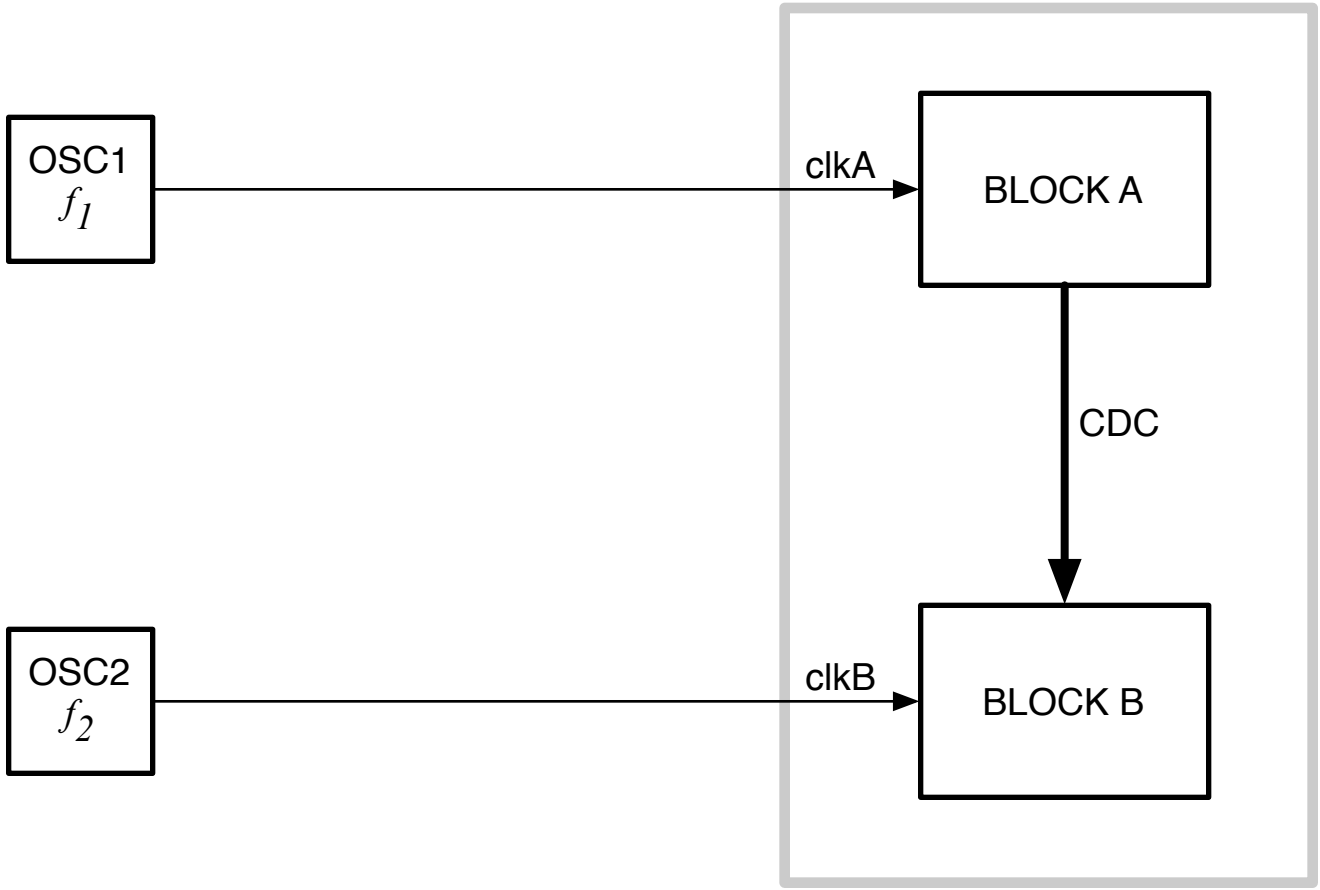
Plesiochronous clocks

Different source, same frequency



Asynchronous clocks

Different source, different frequency



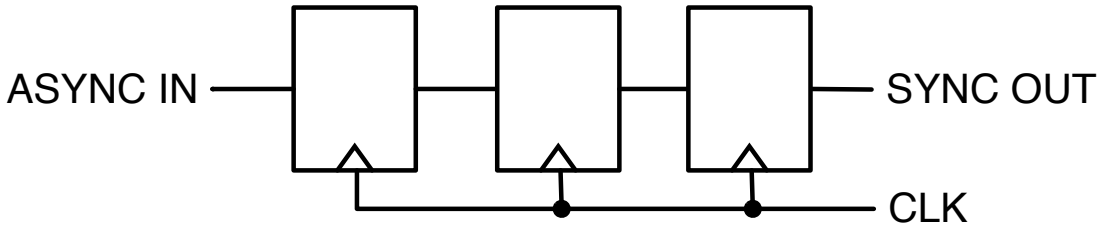
Common fallacies

Multistage synchronizer

Synopsys Users Group

"A two-flop synchronizer isn't enough.
So we use three.
Maybe four.
Five for sure."

Multistage synchronizer



- Correct analysis of MTBF can be surprisingly complicated

$$MTBF = \frac{e^{t_{R_1}/\tau_1} \cdot e^{t_{R_2}/\tau_2}}{f_d f_c T_{0_1}} \qquad MTBF = \frac{e^{2t_R/\tau}}{f_d f_c T_0}$$

- Each stage adds additional propagation delay
- Complex place-and-route constraints
- Reliability reduced by back edge effects
- Instead try a two-flop synchronizer with slower clock…

Example MTBF calculation

Two flip-flop synchronizer using our simple D flip-flop

τ = 44 ps (from our SPICE simulation)

T_0 = 350 ps (from our SPICE simulation)

f_c = 600 MHz (nice fast design)

f_d = 125 MHz

t_R = $1/f_c$ – setup and propagation delay
= 1667 ps – 400 ps
= 1267 ps

Example MTBF calculation

Two flip-flop synchronizer using our simple D flip-flop

$$MTBF = \frac{e^{t_R/\tau}}{f_d f_c T_0}$$

$$= \frac{e^{\frac{1267\,\text{ps}}{44\,\text{ps}}}}{(125 \cdot 10^6\,\text{Hz}) \times (600 \cdot 10^6\,\text{Hz}) \times (350 \cdot 10^{-12}\,\text{sec})}$$

$$= 122 \cdot 10^3\,\text{sec}$$

$$\approx 34\,\text{hours}$$

Example MTBF calculation

Two flip-flop synchronizer using our simple D flip-flop

- Increase resolution time by using divided-by-two clock
- t_R more than doubles!

t_R = 2 × $(1/f_c)$ – setup and propagation delay
= 2 × (1667 ps) – 400 ps
= 2934 ps

Example MTBF calculation

Two flip-flop synchronizer using our simple D flip-flop

$$MTBF = \frac{e^{\frac{2934\,\text{ps}}{44\,\text{ps}}}}{(125 \cdot 10^6\,\text{Hz}) \times (300 \cdot 10^6\,\text{Hz}) \times (350 \cdot 10^{-12}\,\text{sec})}$$

$$= 6.9 \cdot 10^{21}\,\text{sec}$$

$$\approx 220\,\text{trillion years}$$

$$\approx 16{,}000 \times \text{age of the universe}$$

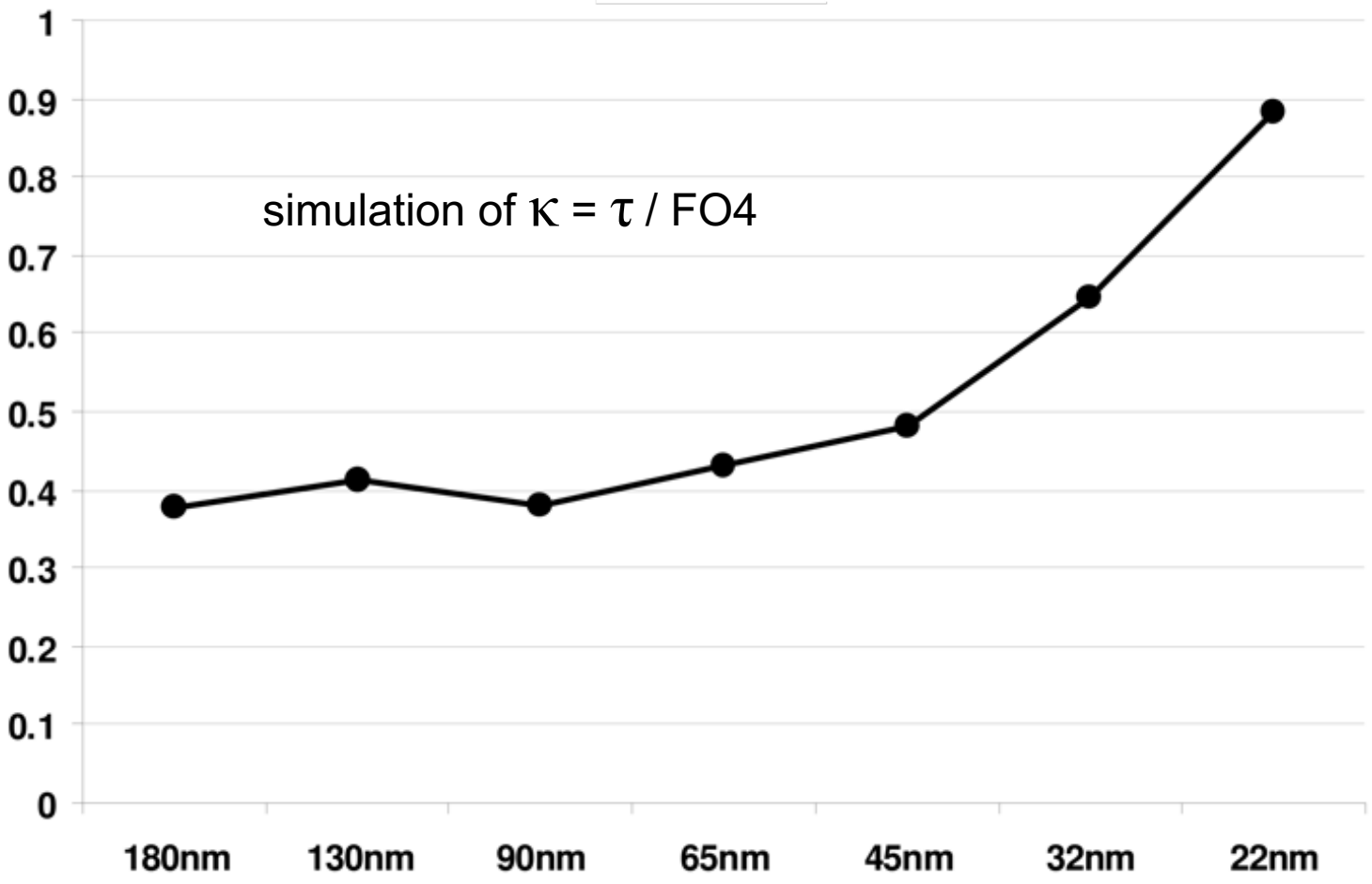
Common fallacies

Moore's Law

"I'm sure at 22nm
all these problems will go away."

τ degradation effect

Worse synchronizer performance at smaller nodes



[Beer et al., ASYNC2010, fig. 19]

Common fallacies

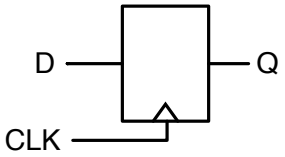
Cell characterization

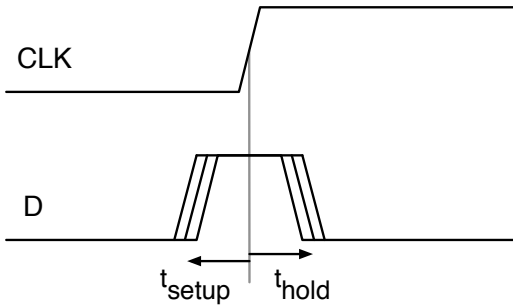
Synopsys Users Group

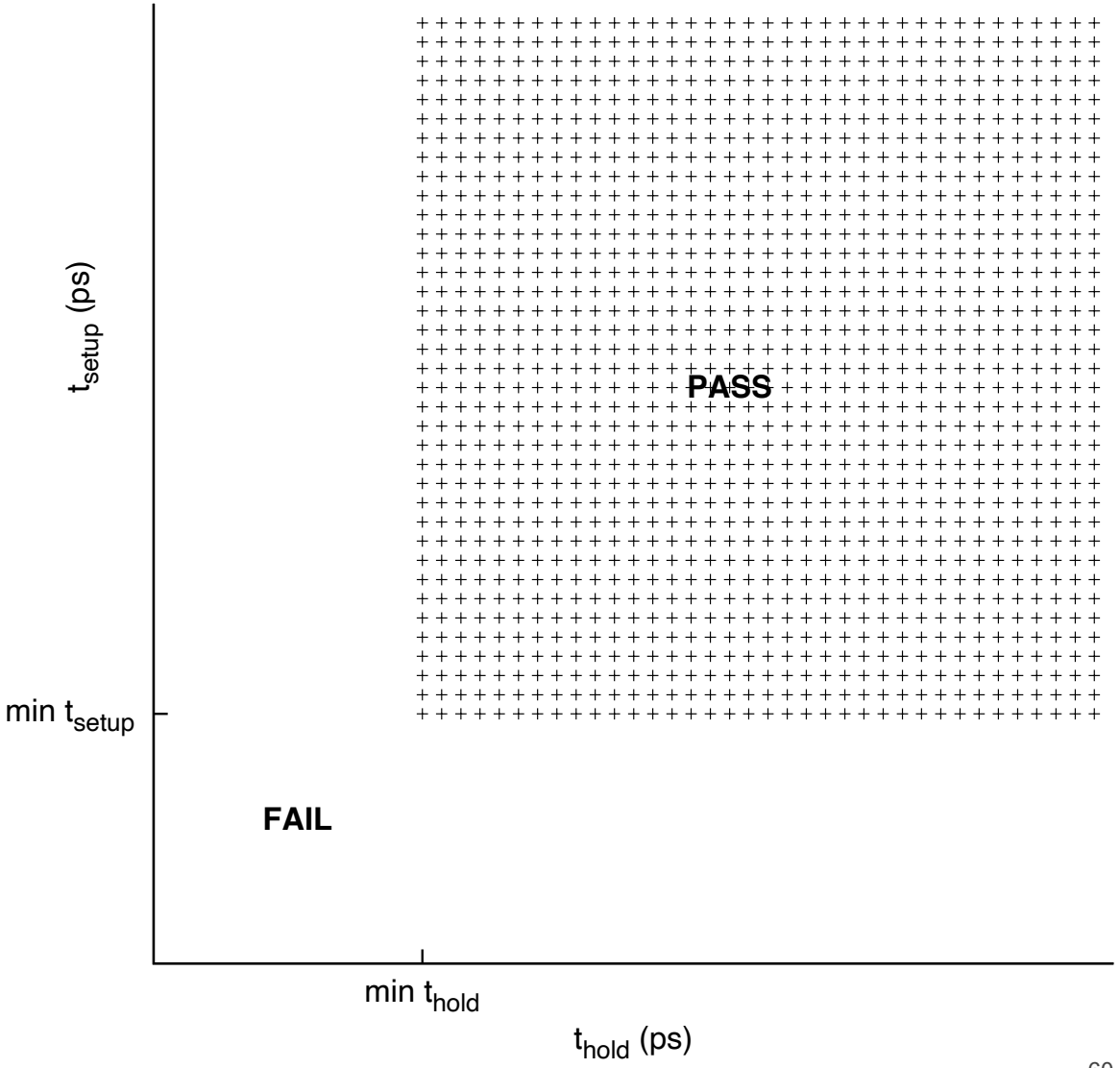
"I trust the characterization of our library.
Right down to the picosecond."

Expected schmoo plot

Simple D flip-flop

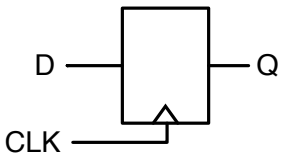


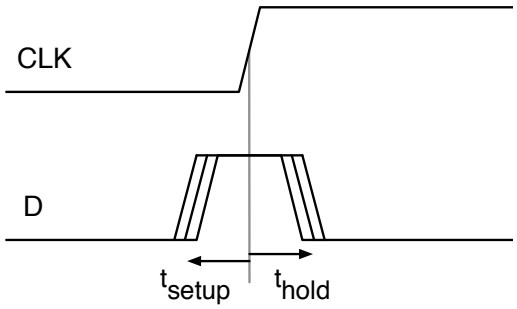


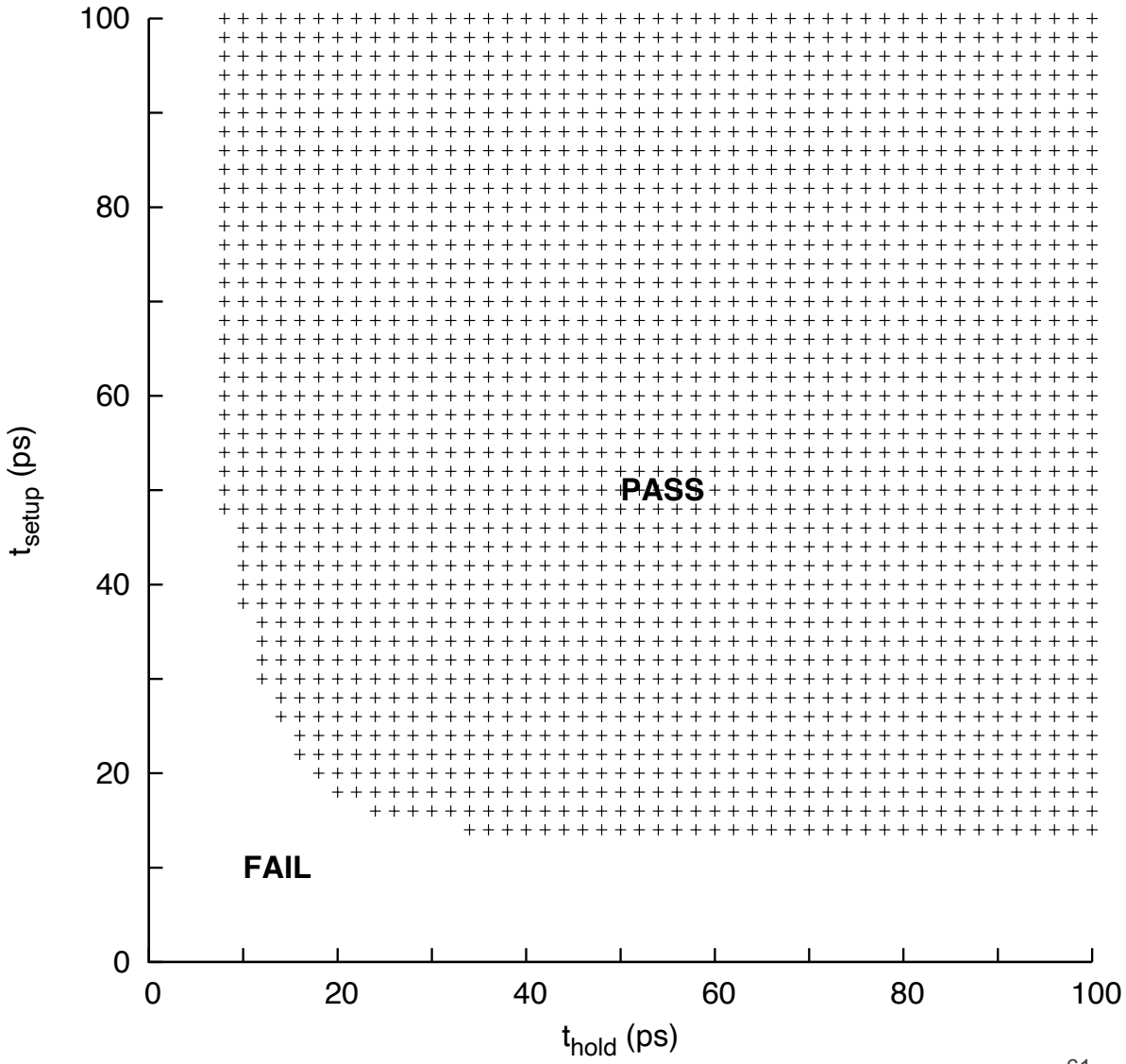


SPICE schmoo plot

Simple D flip-flop

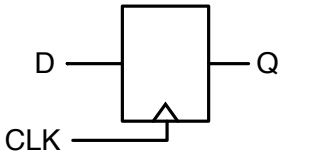


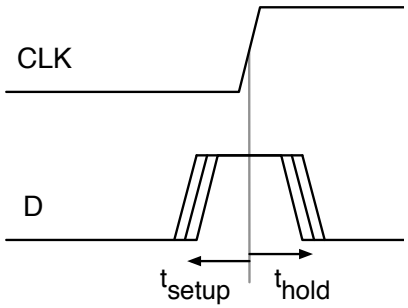


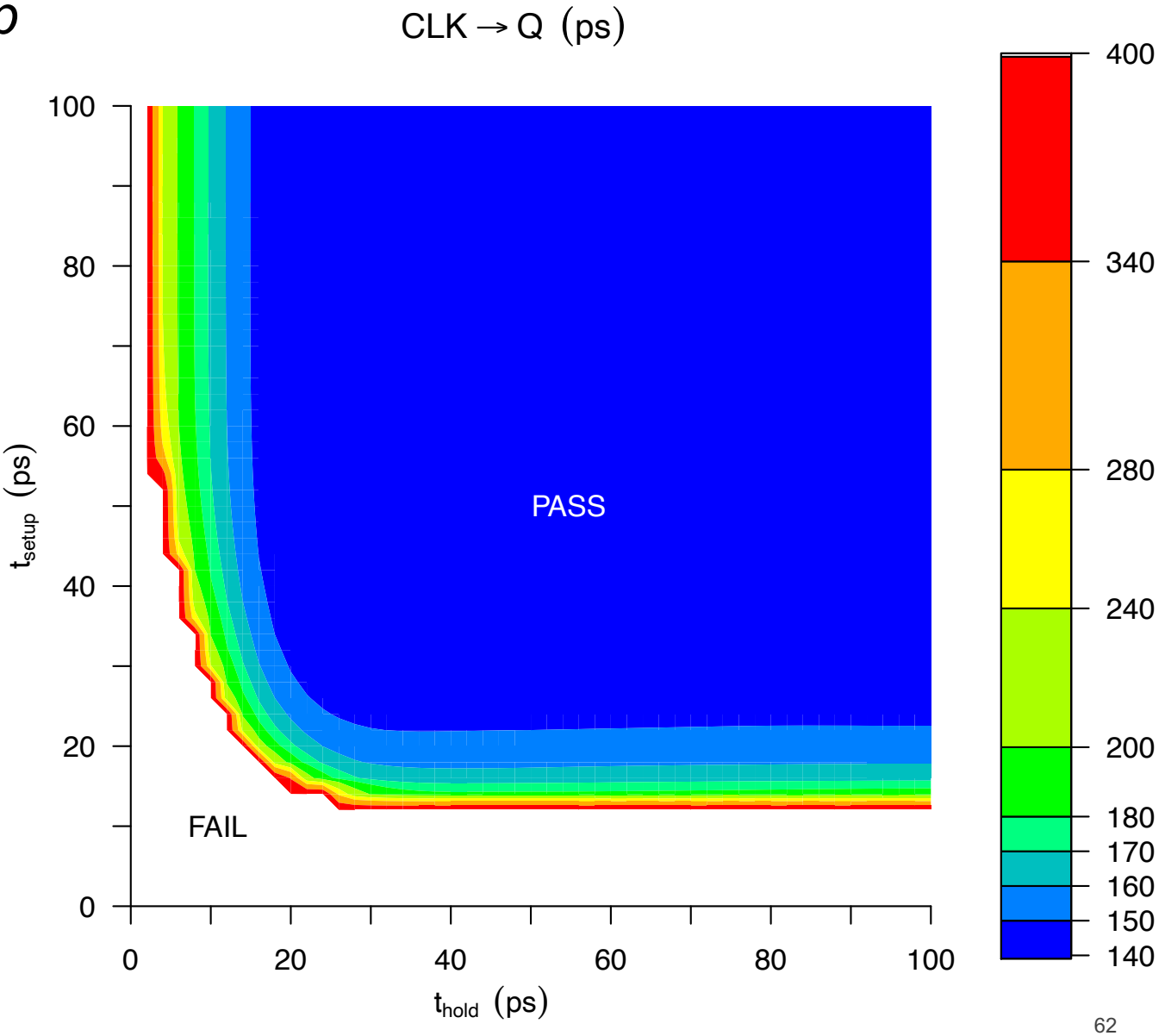


SPICE contoured schmoo plot

Simple D flip-flop







Common fallacies

Gray code

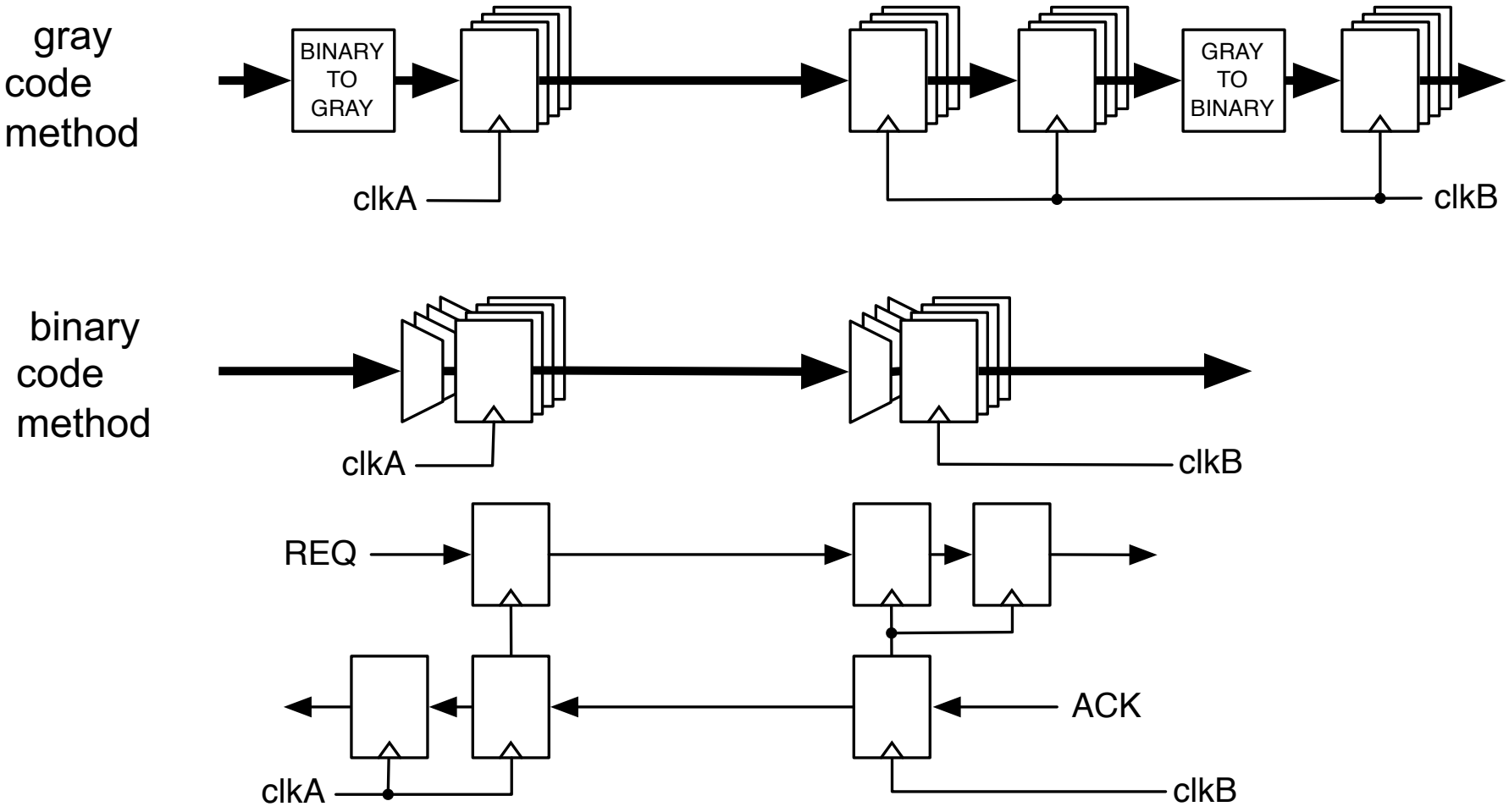
Synopsys Users Group

"We use gray code for our buses.
No problem."

Gray code

- Gray code is sometimes used for FIFO pointers
- Multibit bus has individual synchronizers on each bit
- Problems
  - only works for value changes of ±1
  - only works for 2^n values
  - requires more synchronizers
  - requires more logic
- Design reuse may disrupt your careful assumptions…

Gray code requires more logic



Common fallacies

I just don't believe it

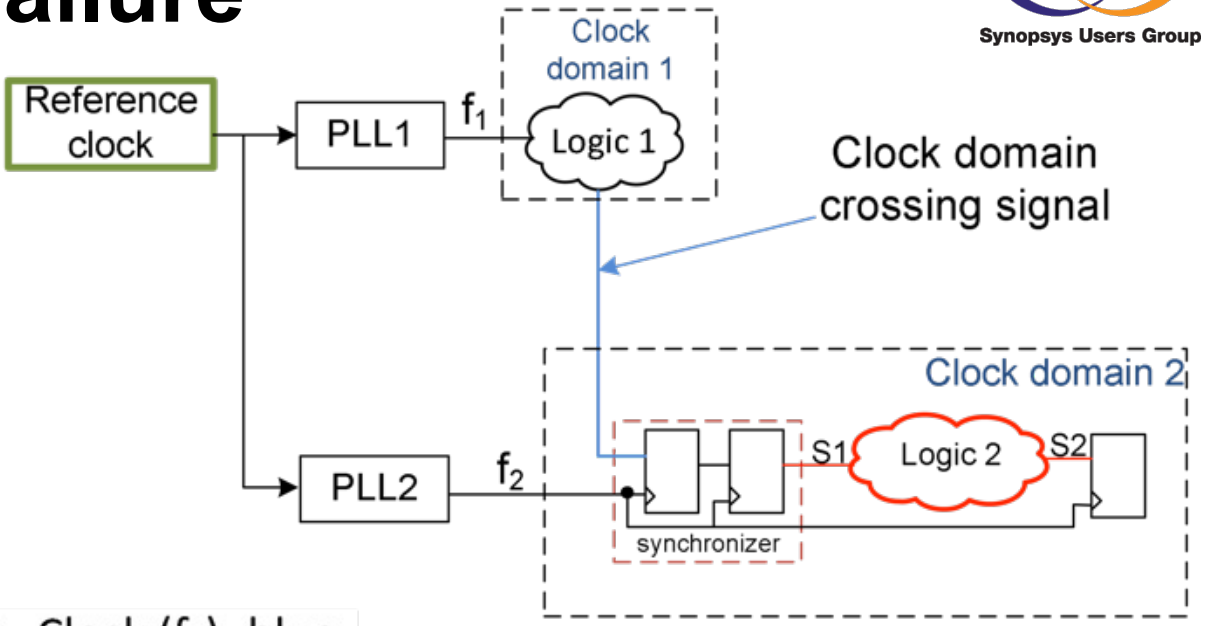
"I've never seen a synchronizer failure.
I don't believe such a thing exists."

It's not about *belief*
It's about *understanding*

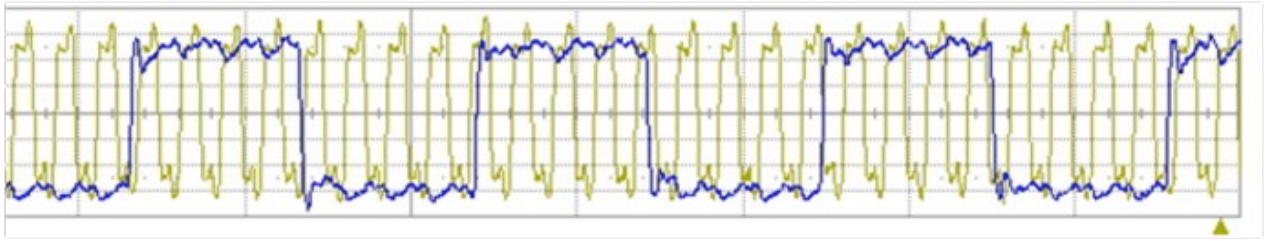
- What does a synchronizer failure look like?
- How often does it occur?
- How critical is the signal?

Metastability failure

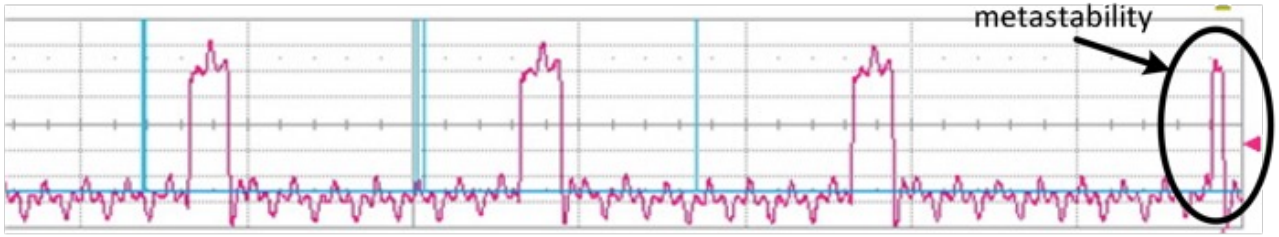
40nm SoC



Clock (f_2) – yellow , Clock (f_1) -blue



Synchronizer output (s1)



[from Beer et al. ASYNC 2013]

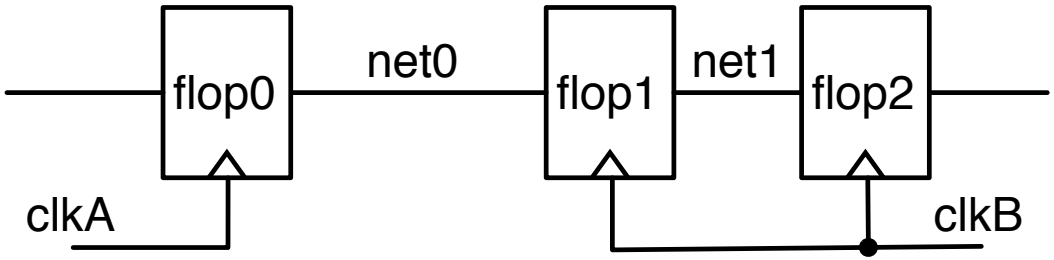
Common fallacies

- MTBF
- Related clocks
- Gray code
- Flop selection
- Basic 2-flop synchronizer
- Multistage synchronizer
- Custom synchronizer cells
- Design flow
- Design reuse
- Cell characterization
- Moore’s Law
- I just don’t believe it

Methodology for constraining synchronizers

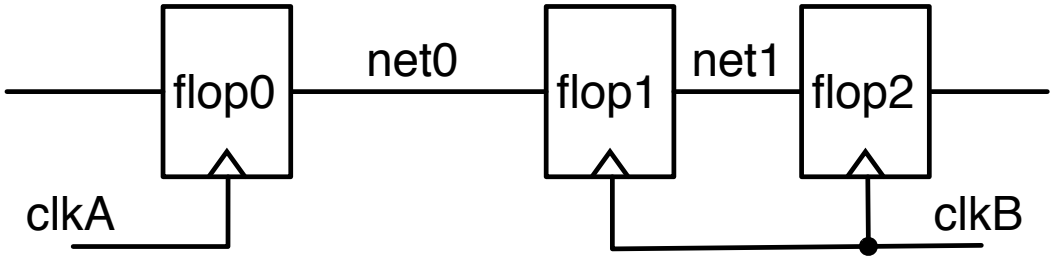
or, it's a Synopsys conference so I'd better talk about tools

Find all the synchronizers



- In your RTL, put a dummy cell on `net0`
- Have a special name pattern for sync flops like `flop1`
- Instantiate a special SYNC cell from your library
- Use a CDC tool to find all the synchronizers
- Use a custom PrimeTime script to locate CDC nets
- All of the above

Automatically generate a custom SDC file



- Constrain delay between `flop0` and `flop1`

```
create_clock -name sync_launch_clock  [get_pins flop0/CK]
create_clock -name sync_capture_clock [get_pins flop1/CK]
set_false_path -to   [get_clock sync_launch_clock]
set_false_path -from [get_clock sync_capture_clock]
```

- Fast edges on nets

```
set_max_transition [get_net {net0 net1}]
set_max_capacitance [get_net {net0 net1}]
```

Conclusions

or, your journey is just beginning

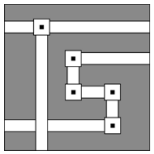
Expertise in metastability

or, things you need to know about

- front-end flow
- digital design
- analog design
- cell library design
- library characterization
- reliability engineering
- high-speed lab techniques
- risk evaluation
- design for test
- back-end implementation
- fault simulation
- data presentation
- statistics
- probability
- combinatorics
- marketing
- sales forecasting
- human psychology

You cannot prevent metastability but you can *manage it*

- You are empowered!
- Avoid the Fallacies
- Follow the Recommendations



TRILOBYTE SYSTEMS

snug
Synopsys Users Group

Thank You

Methodology for constraining managers

or,

it may look like a technical problem,

but really it might be a people problem

How to convince your company to change its methodology

This can be challenging!

- Making a change can imply
  - that your current methodology is broken
  - that your current chips could fail
- This is unpleasant to contemplate!
- What to do?

How to convince your company to change its methodology

This can be challenging!

- Making a change can imply
  - that your current methodology is broken
  - that your current chips could fail
- This is unpleasant to contemplate!
- What to do?

Focus on your new designs, rather than your old ones

Focus on your new designs

Synopsys Users Group

Our new chips are more complex,
with more clocks,
and many more clock domain crossings.

Metastability seems to be getting worse below 65nm.
τ may no longer scale with process.

Running at higher clock rates
makes this problem deserving of more careful study.

New process features (e.g., FinFETs)
make it prudent to review our methodology.