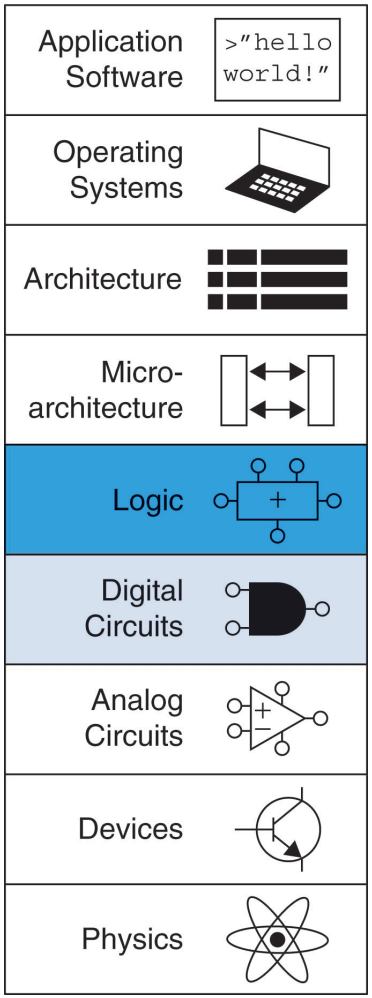
E85 Digital Design & Computer Engineering

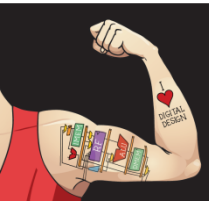
Lecture 5:
Sequential Timing

HARVEY MUDD COLLEGE

Lecture 5

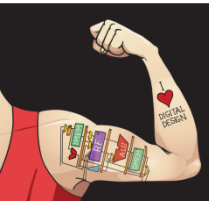
- **Timing of Sequential Logic**
- **Metastability**
- **Parallelism**





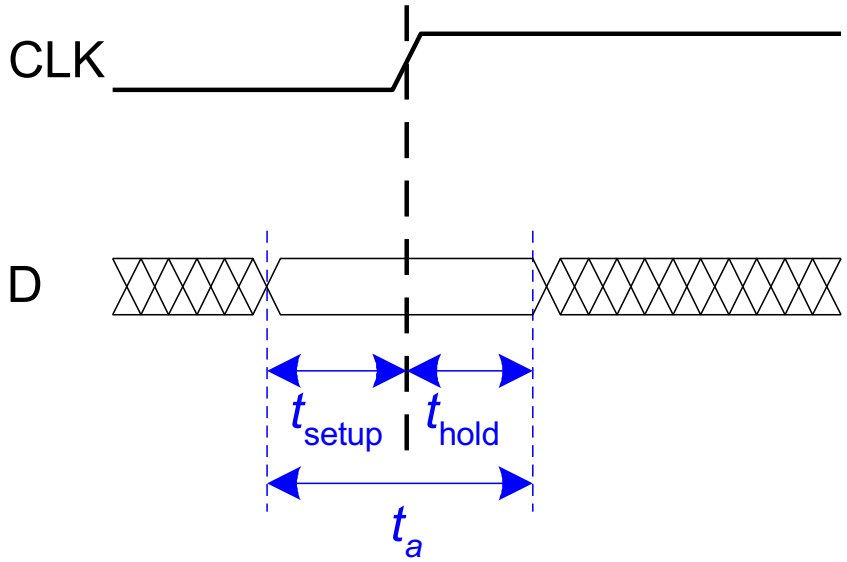
Timing

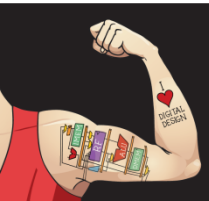
- Flip-flop samples *D* at clock edge
- *D* must be stable when sampled
- Similar to a photograph, *D* must be stable around clock edge
- If not, metastability can occur



Input Timing Constraints

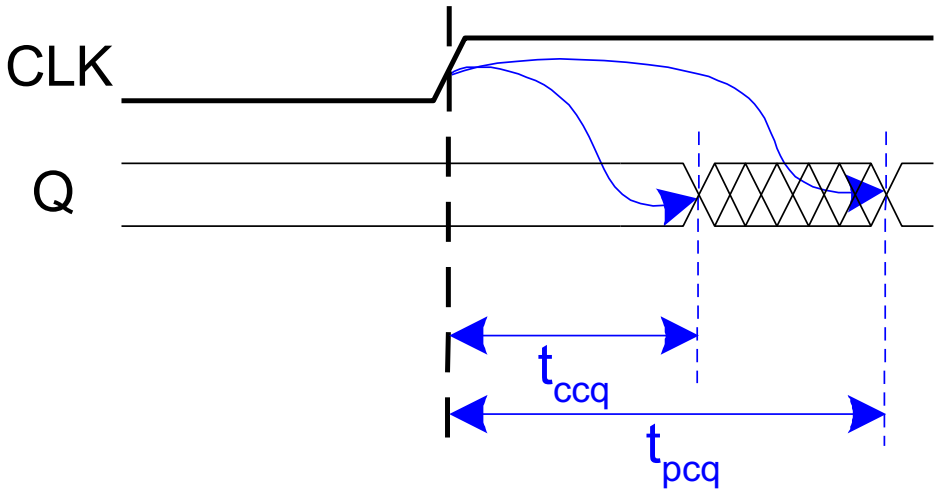
- **Setup time:** t_{setup} = time *before* clock edge data must be stable (i.e. not changing)
- **Hold time:** t_{hold} = time *after* clock edge data must be stable
- **Aperture time:** t_a = time *around* clock edge data must be stable ($t_a = t_{setup} + t_{hold}$)

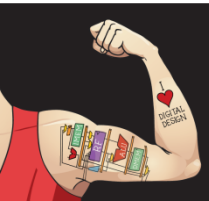




Output Timing Parameters

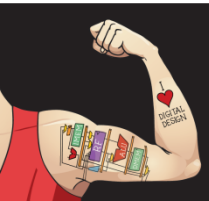
- **Propagation delay:** t_{pcq} = time after clock edge that the output *Q* is guaranteed to be stable (i.e., to stop changing)
- **Contamination delay:** t_{ccq} = time after clock edge that *Q* might be unstable (i.e., start changing)





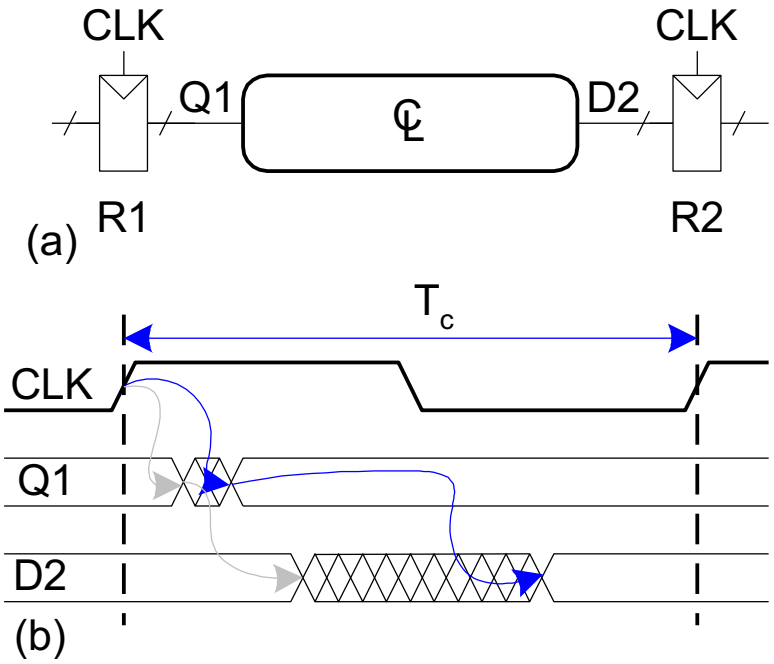
Dynamic Discipline

- Synchronous sequential circuit inputs must be stable during aperture (setup and hold) time around clock edge
- Specifically, inputs must be stable
  - at least t_{setup} before the clock edge
  - at least until t_{hold} after the clock edge



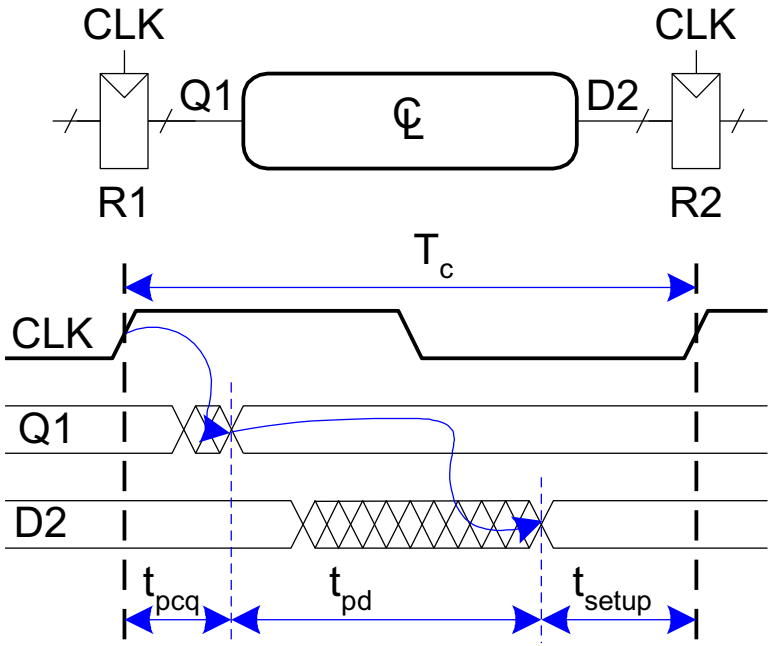
Dynamic Discipline

- The delay between registers has a **minimum** and **maximum** delay, dependent on the delays of the circuit elements



Setup Time Constraint

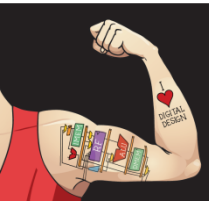
- Depends on the **maximum** delay from register R1 through combinational logic to R2
- The input to register R2 must be stable at least t_{setup} before clock edge



$$T_c \geq$$

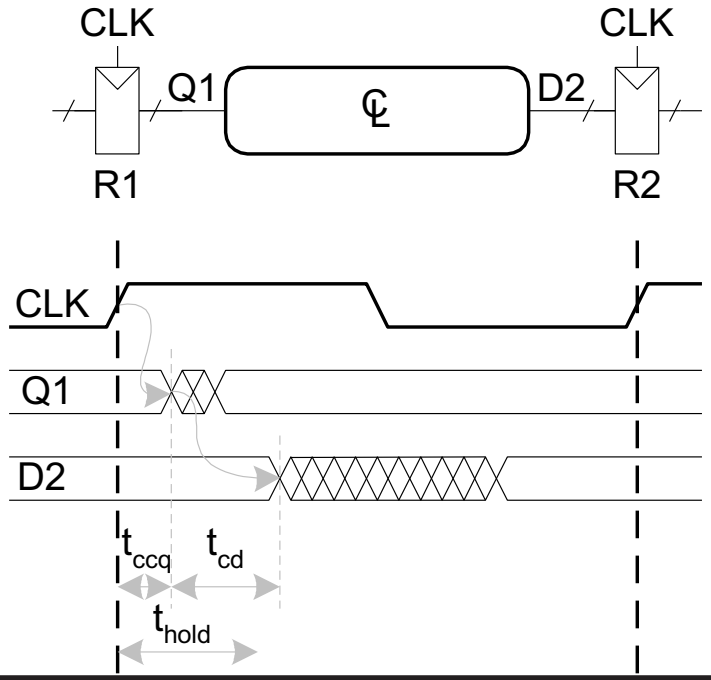
$$t_{pd} \leq$$

$(t_{pcq} + t_{setup})$: *sequencing overhead*



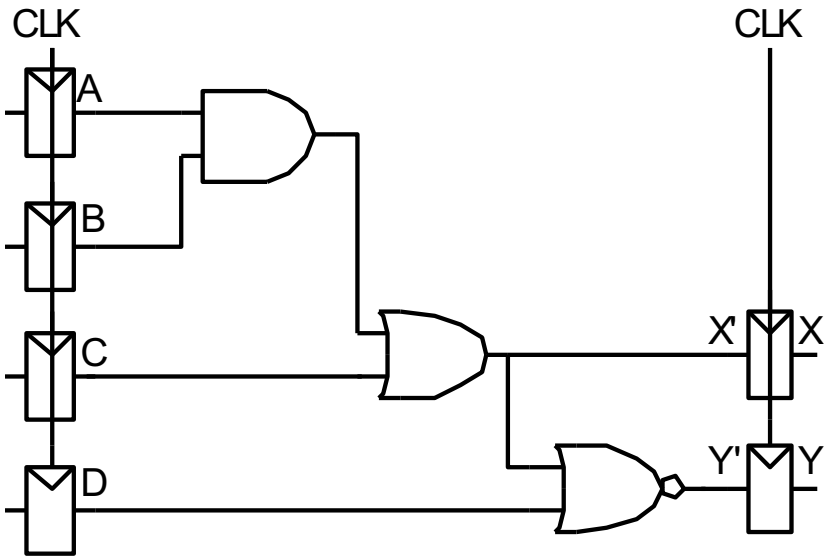
Hold Time Constraint

- Depends on the **minimum** delay from register R1 through the combinational logic to R2
- The input to register R2 must be stable for at least t_{hold} after the clock edge



$$t_{hold} < t_{cd}$$

Timing Analysis



Timing Characteristics

t_{ccq} = 30 ps
t_{pcq} = 50 ps
t_{setup} = 60 ps
t_{hold} = 70 ps

per gate:
t_{pd} = 35 ps
t_{cd} = 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 25 ps

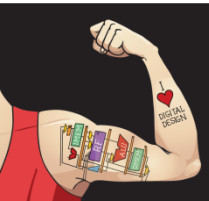
Setup time constraint:

$T_c \geq$

$f_c = 1/T_c =$

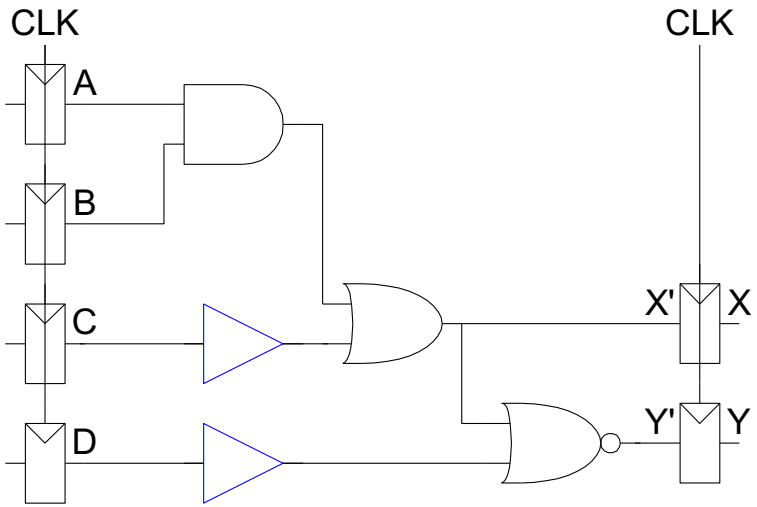
Hold time constraint:

$t_{ccq} + t_{cd} > t_{hold}$?



Timing Analysis

Add buffers to the short paths:



Timing Characteristics

t_{ccq} = 30 ps
t_{pcq} = 50 ps
t_{setup} = 60 ps
t_{hold} = 70 ps

per gate:
t_{pd} = 35 ps
t_{cd} = 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 2 x 25 ps = 50 ps

Setup time constraint:

$T_c \geq$

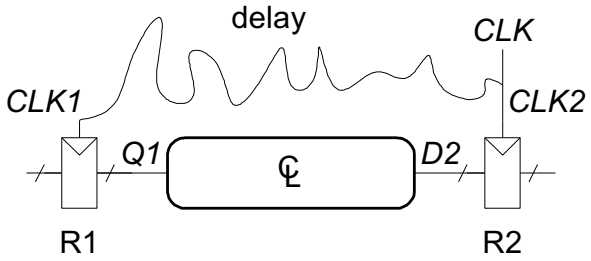
$f_c = 1/T_c =$

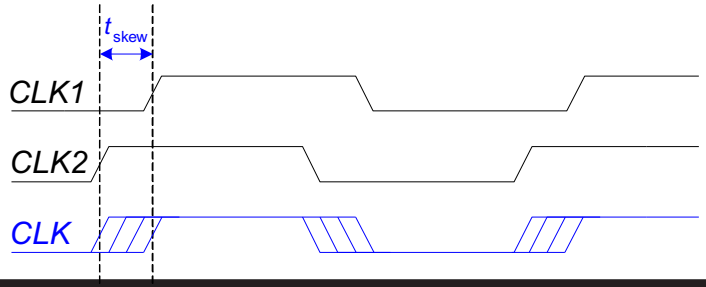
Hold time constraint:

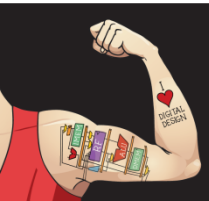
$t_{ccq} + t_{cd} > t_{hold}$?

Clock Skew

- The clock doesn't arrive at all registers at same time
- **Skew:** difference between two clock edges
- Perform **worst case analysis** to guarantee dynamic discipline is not violated for any register – many registers in a system!

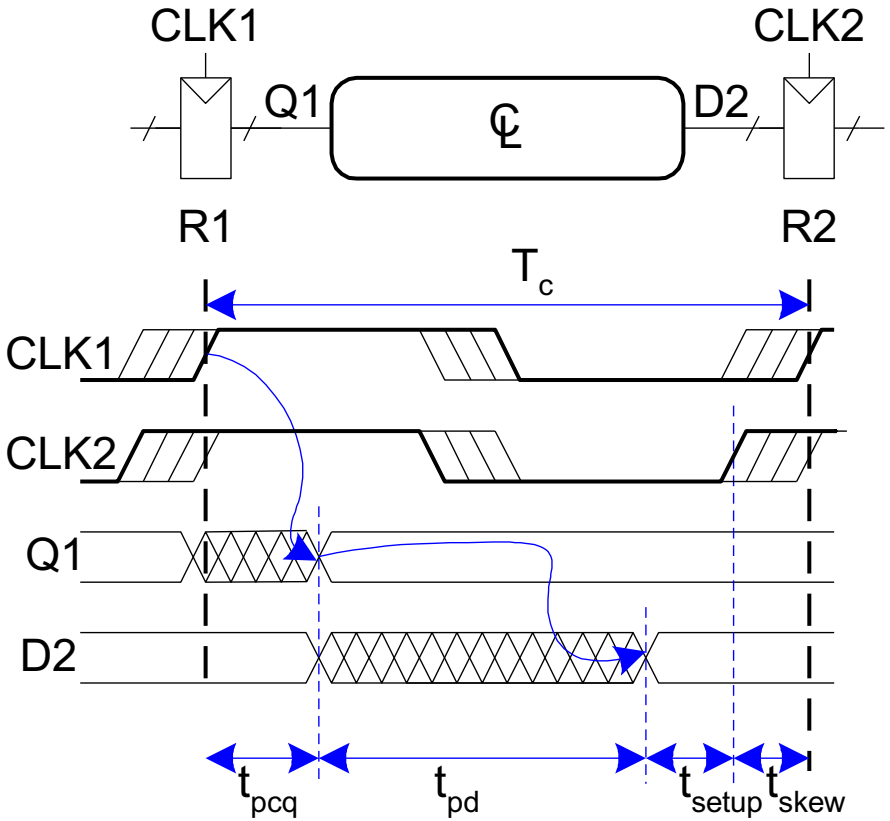






Setup Time Constraint with Skew

- In the worst case, CLK2 is earlier than CLK1

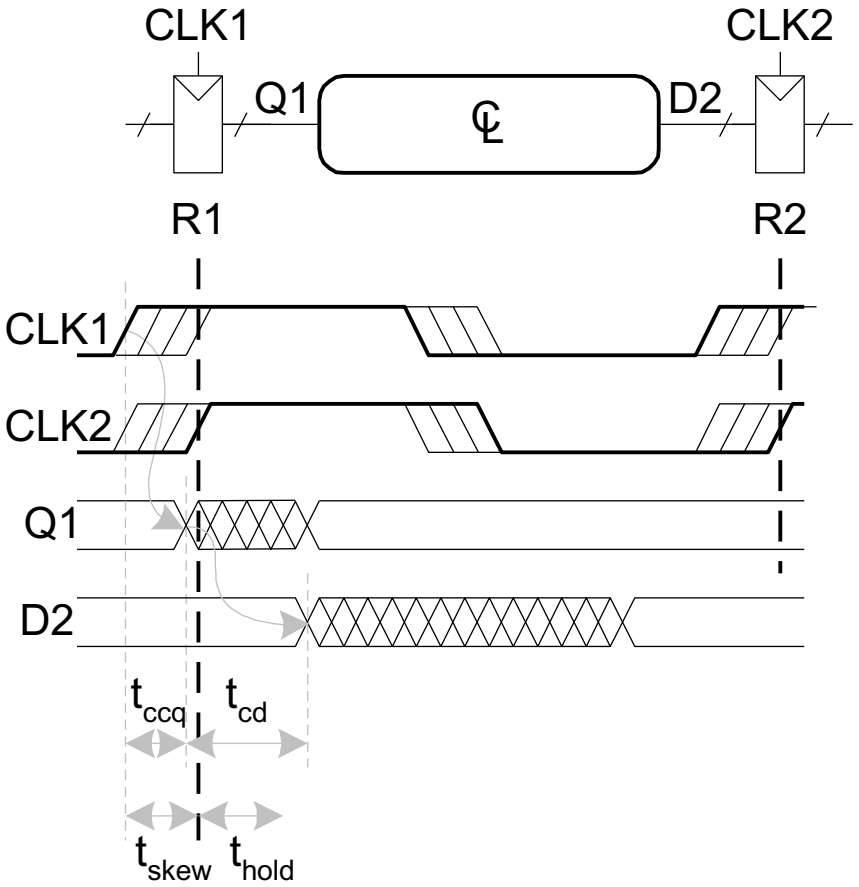


$T_c \geq$

$t_{pd} \leq$

Hold Time Constraint with Skew

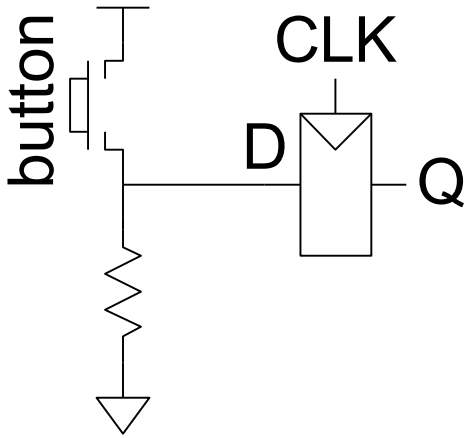
- In the worst case, CLK2 is later than CLK1

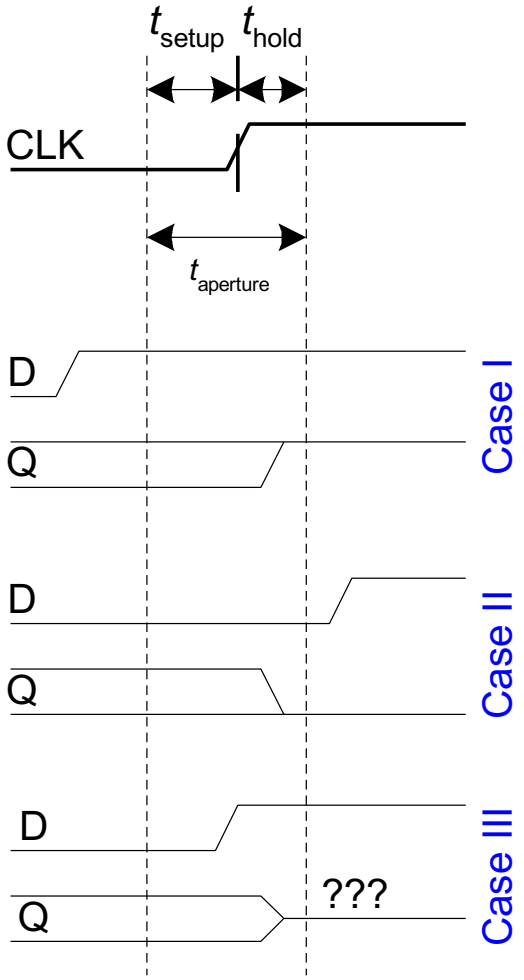


$t_{cd} >$

Violating the Dynamic Discipline

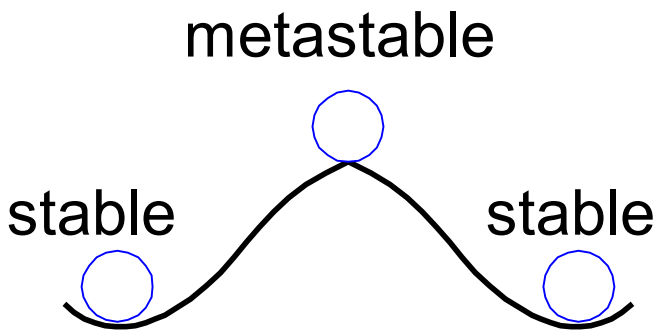
Asynchronous (for example, user) **inputs** might violate the dynamic discipline

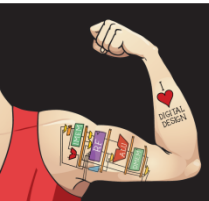




Metastability

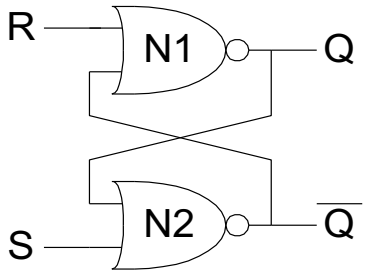
- **Bistable devices:** two stable states, and a metastable state between them
- **Flip-flop:** two stable states (1 and 0) and one metastable state
- If flip-flop lands in metastable state, could stay there for an undetermined amount of time





Flip-Flop Internals

- Flip-flop has **feedback**: if *Q* is somewhere between 1 and 0, cross-coupled gates drive output to either rail (1 or 0)



- **Metastable signal:** if it hasn't resolved to 1 or 0
- If flip-flop input changes at random time, **probability that output *Q* is metastable** after waiting some time, *t*:

$$P(t_{res} > t) = (T_0/T_c)\, e^{-t/\tau}$$

t_{res} : time to resolve to 1 or 0

T_0, τ : properties of the circuit

Metastability

- **Intuitively:**

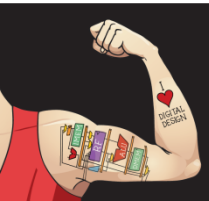
  T_0/T_c: probability input changes at a bad time (during aperture)

  $$P(t_{res} > t) = (T_0/T_c)\, e^{-t/\tau}$$

  τ: time constant for how fast flip-flop moves away from metastability

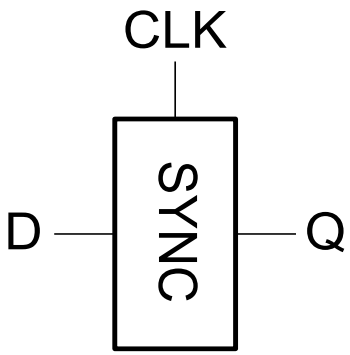
  $$P(t_{res} > t) = (T_0/T_c)\, e^{-t/\tau}$$

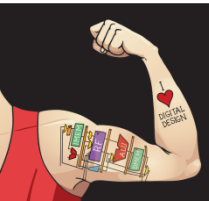
- If flip-flop samples metastable input, if you wait long enough (t), the output will have resolved to 1 or 0 with high probability.



Synchronizers

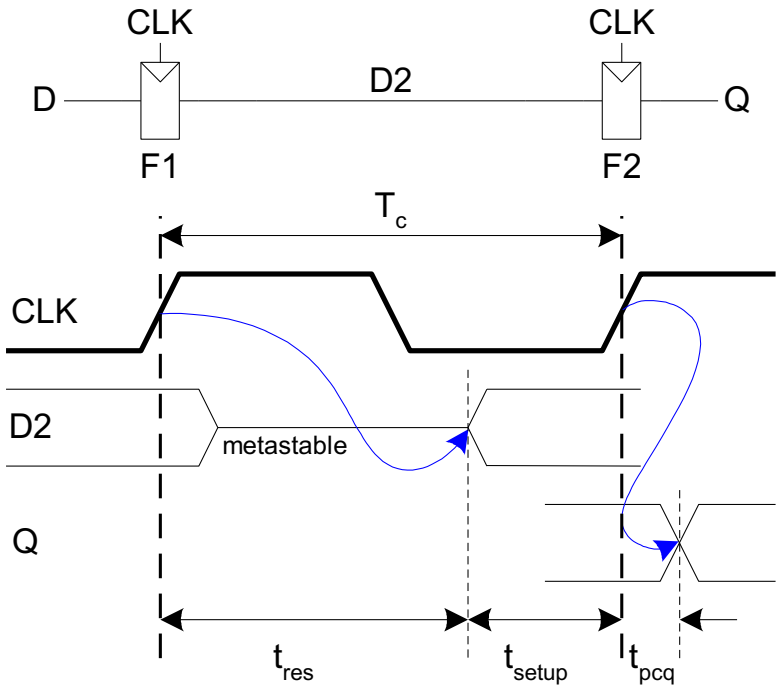
- **Asynchronous inputs are inevitable** (user interfaces, systems with different clocks interacting, etc.)
- **Synchronizer goal:** make the probability of failure (the output *Q* still being metastable) low
- Synchronizer cannot make the probability of failure 0





Synchronizer Internals

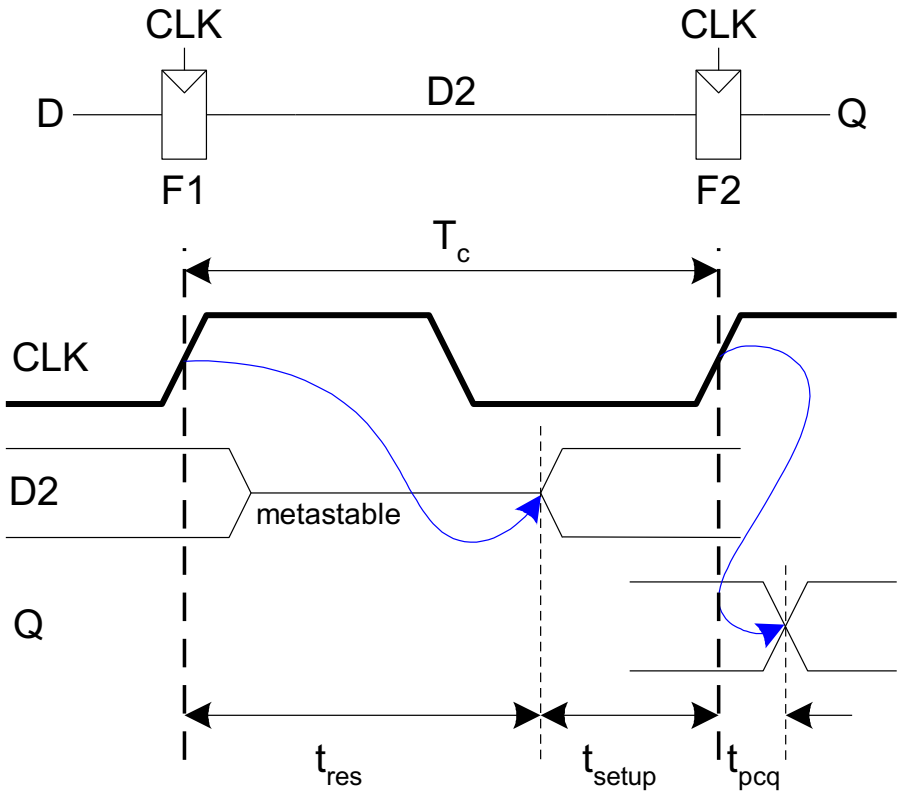
- Synchronizer: built with two back-to-back flip-flops
- Suppose D is transitioning when sampled by F1
- Internal signal D2 has (T_c - t_{setup}) time to resolve to 1 or 0



Synchronizer Probability of Failure

For each sample, probability of failure is:

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



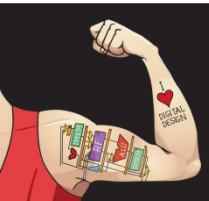
Synchronizer Mean Time Between Failures

- If asynchronous input changes once per second, probability of failure per second is *P*(failure).
- If input changes *N* times per second, probability of failure per second is:

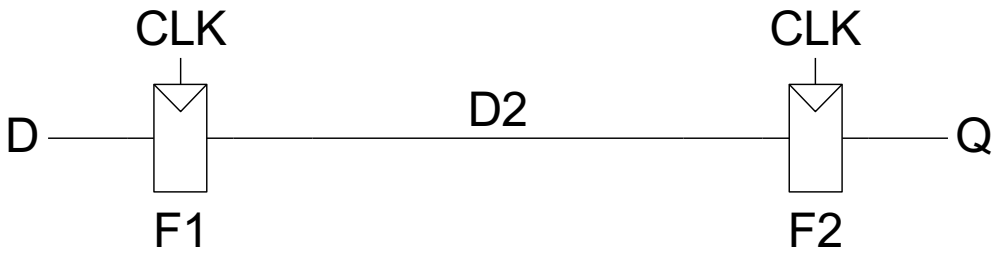
$$P(\text{failure})/\text{second} = (NT_0/T_c)\, e^{-(T_c - t_{setup})/\tau}$$

- Synchronizer fails, on average, 1/[*P*(failure)/second]
- Called ***mean time between failures***, MTBF:

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

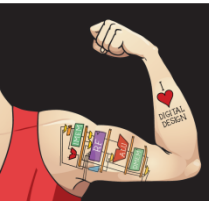


Example Synchronizer



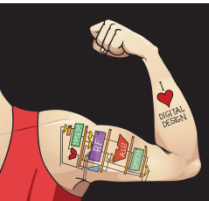
- Suppose: $T_c = 1/500\ \text{MHz} = 2\ \text{ns}$ $\tau = 200\ \text{ps}$
  $T_0 = 150\ \text{ps}$ $t_{setup} = 100\ \text{ps}$
  $N = 10$ events per second
- What is the probability of failure? MTBF?

$$P(\text{failure}) = (150\ \text{ps}/2\ \text{ns})\ e^{-(1.9\ \text{ns})/200\ \text{ps}}$$
$$= \mathbf{5.6 \times 10^{-6}}$$
$$P(\text{failure})/\text{second} = 10 \times (5.6 \times 10^{-6})$$
$$= 5.6 \times 10^{-5}\ /\ \text{second}$$
$$\text{MTBF} = 1/[\text{P(failure)/second}] \approx \mathbf{5\ \text{hours}}$$



Parallelism

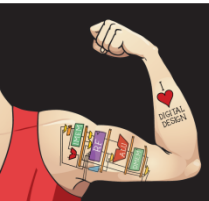
- **Two types of parallelism:**
  - **Spatial parallelism**
    - duplicate hardware performs multiple tasks at once
  - **Temporal parallelism**
    - task is broken into multiple stages
    - also called pipelining
    - for example, an assembly line



Parallelism Definitions

- **Token:** Group of inputs processed to produce group of outputs
- **Latency:** Time for one token to pass from start to end
- **Throughput:** Number of tokens produced per unit time

Parallelism increases throughput

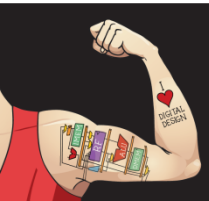


Parallelism Example

- Ben Bitdiddle bakes cookies to celebrate traffic light controller installation
- 5 minutes to roll cookies
- 15 minutes to bake
- What is the latency and throughput without parallelism?

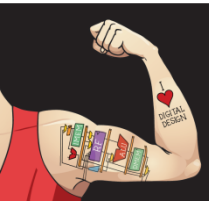
Latency =

Throughput =

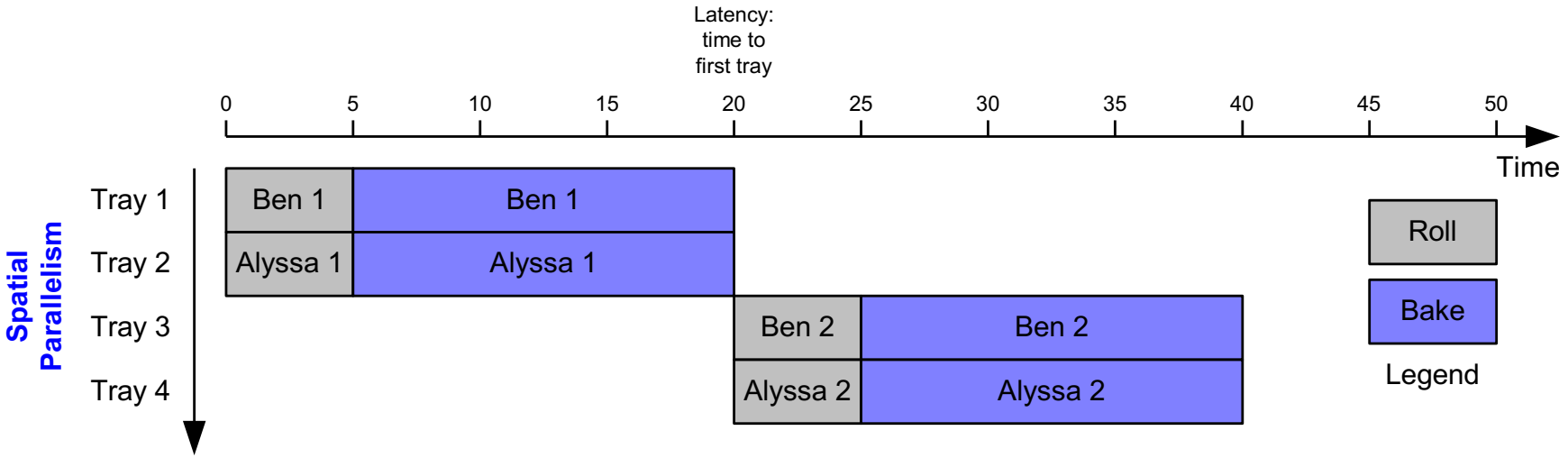


Parallelism Example

- What is the latency and throughput if Ben uses parallelism?
  - **Spatial parallelism:** Ben asks Allysa P. Hacker to help, using her own oven
  - **Temporal parallelism:**
    - two stages: rolling and baking
    - He uses two trays
    - While first batch is baking, he rolls the second batch, etc.

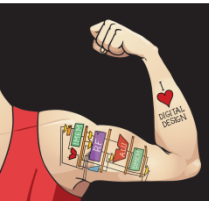


Spatial Parallelism

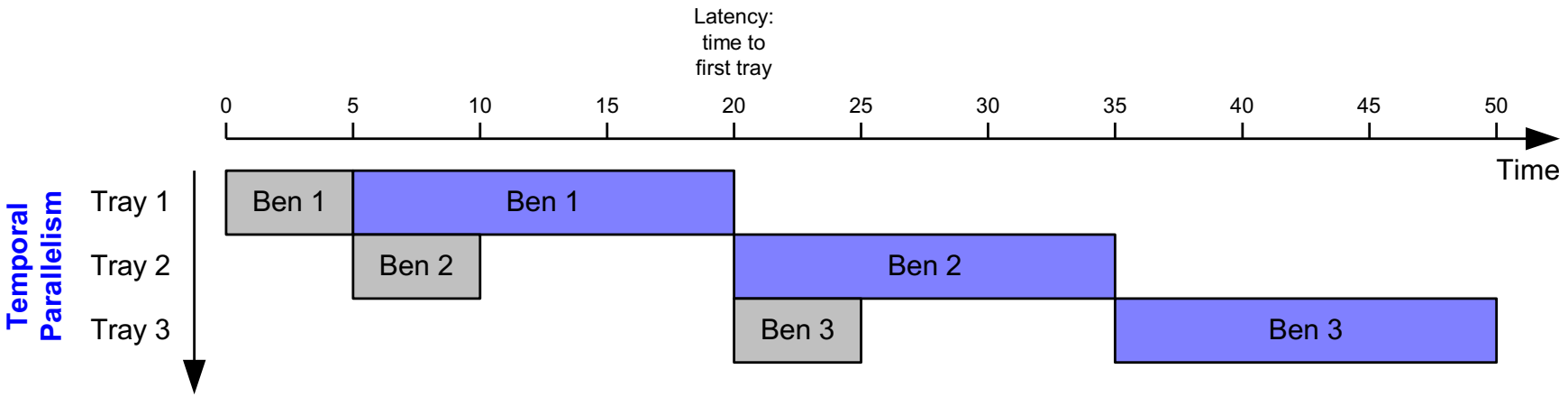


Latency =

Throughput =



Temporal Parallelism



Latency =

Throughput =

Using both techniques, the throughput would be **trays/hour**