Sequential Circuits: Timing

Design of Digital Circuits 2014
Srdjan Capkun
Frank K. Gürkaynak

http://www.syssec.ethz.ch/education/Digitaltechnik_14

What will we learn?

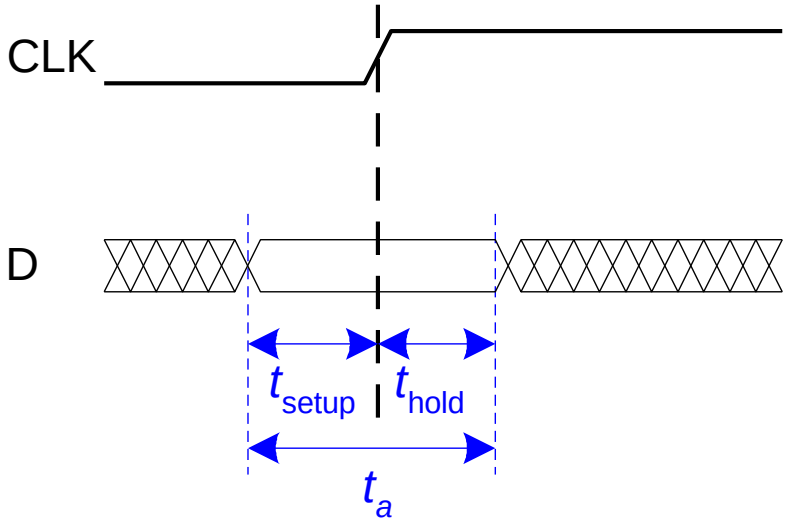
- **Timing of Sequential Logic**
- **Parallelism**

Timing

- **Flip-flop samples D at clock edge**
- **D must be stable when it is sampled**
- **Similar to a photograph, D must be stable around the clock edge**
- **If D is changing when it is sampled, *metastability* can occur**
  - Recall that a flip-flop copies the input D to the output Q on the rising edge of the clock. This process is called sampling D on the clock edge. If D is stable at either 0 or 1 when the clock rises, this behavior is clearly defined. But what happens if D is changing at the same time the clock rises?

Input Timing Constraints

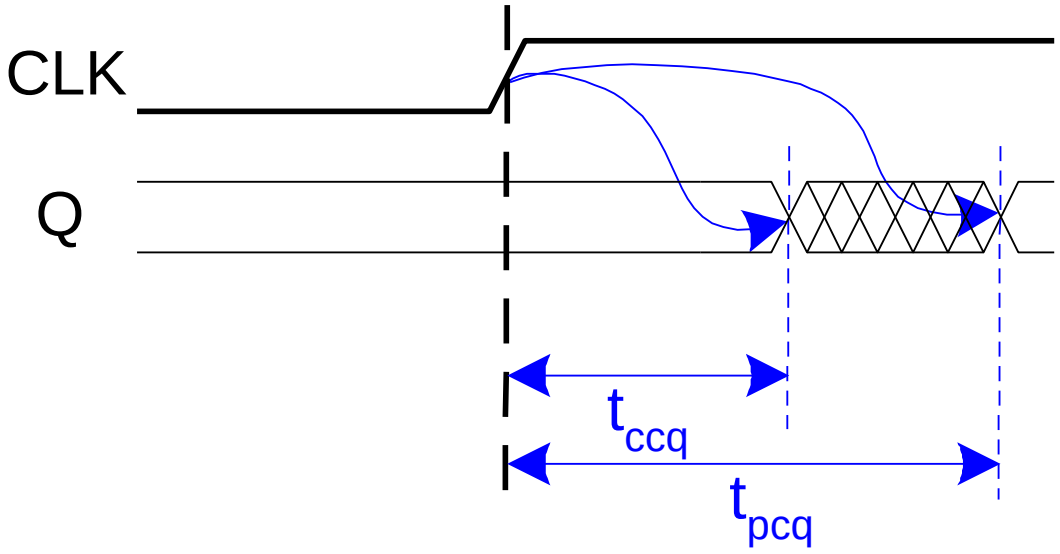
- ***Setup time***: t_{setup} = time before the clock edge that data must be stable (i.e. not changing)
- ***Hold time***: t_{hold} = time after the clock edge that data must be stable
- ***Aperture time***: t_a = time around clock edge that data must be stable ($t_a = t_{setup} + t_{hold}$)



Output Timing Constraints

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

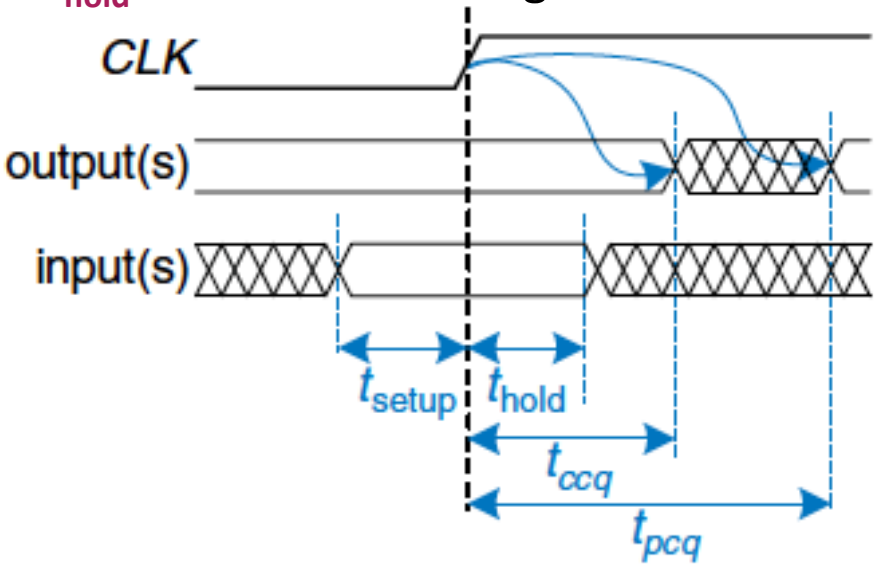
- ***Contamination delay***: t_{ccq} = time after clock edge that Q might be unstable (i.e., start changing)



Dynamic Discipline

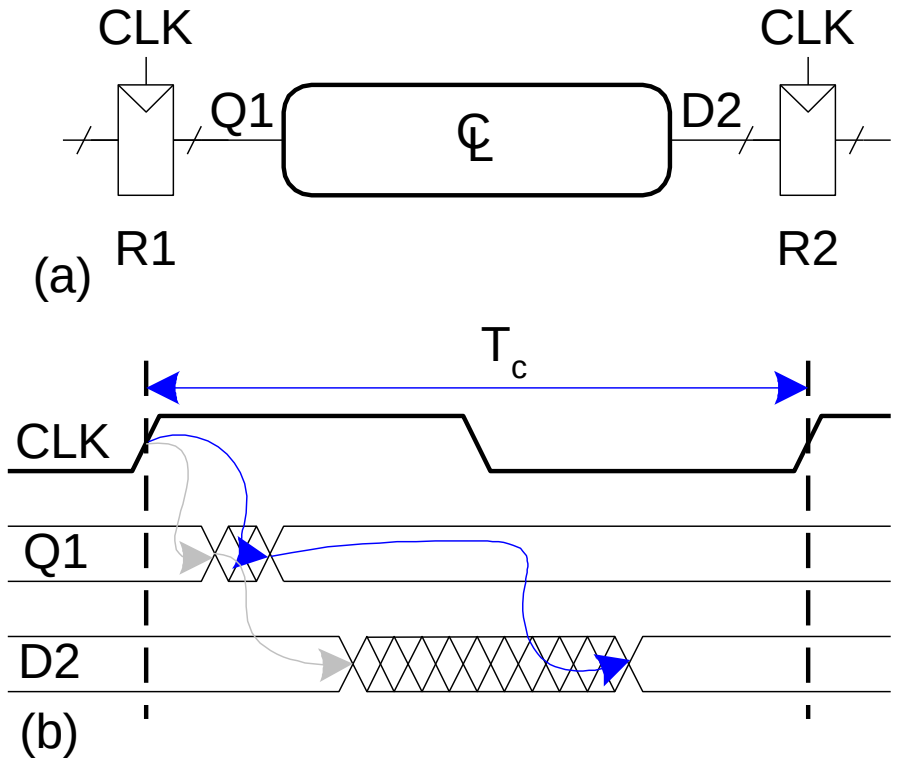
- **The input to a synchronous sequential circuit must be stable during the aperture (setup and hold) time around the clock edge.**

- **Specifically, the input 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**



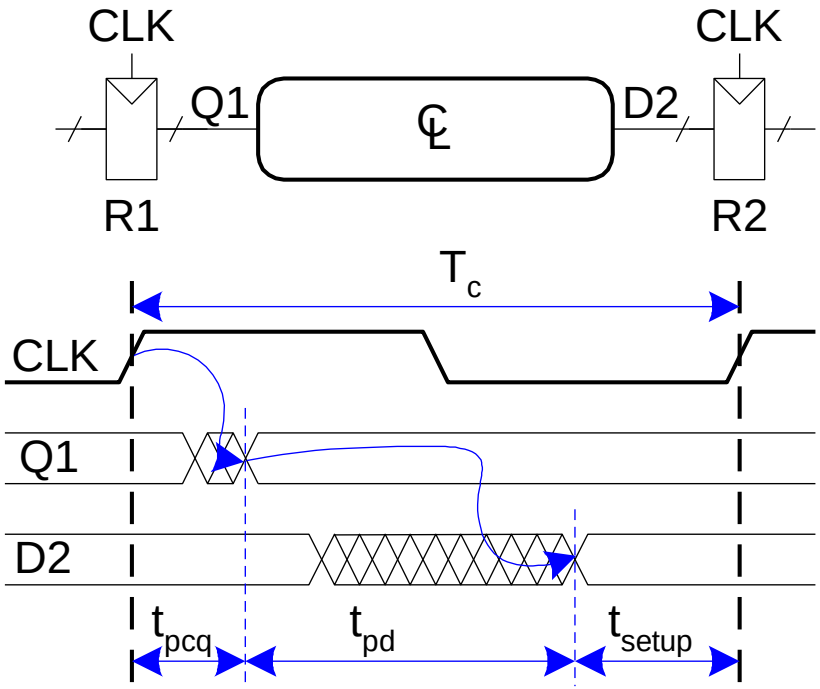
Setup Time Constraint

- **The clock period or cycle time, T_c, is the time between rising edges of a repetitive clock signal. Its reciprocal, $f_c=1/T_c$, is the clock frequency.**
- **All else being the same, increasing the clock frequency increases the work that a digital system can accomplish per unit time.**
- **Frequency is measured in units of Hertz (Hz), or cycles per second:**
  - 1 megahertz (MHz) 10^6 Hz
  - 1 gigahertz (GHz) 10^9 Hz.

Setup Time Constraint

- **The setup time constraint depends on the maximum delay from register R1 through the combinational logic.**

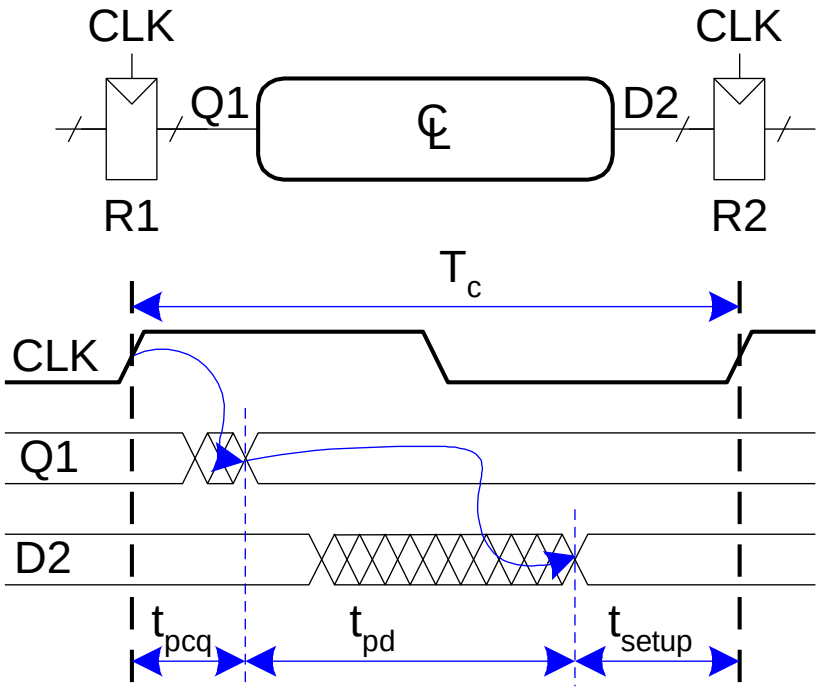
- **The input to register R2 must be stable at least t_{setup} before the clock edge.**



$T_c >=$

Setup Time Constraint

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



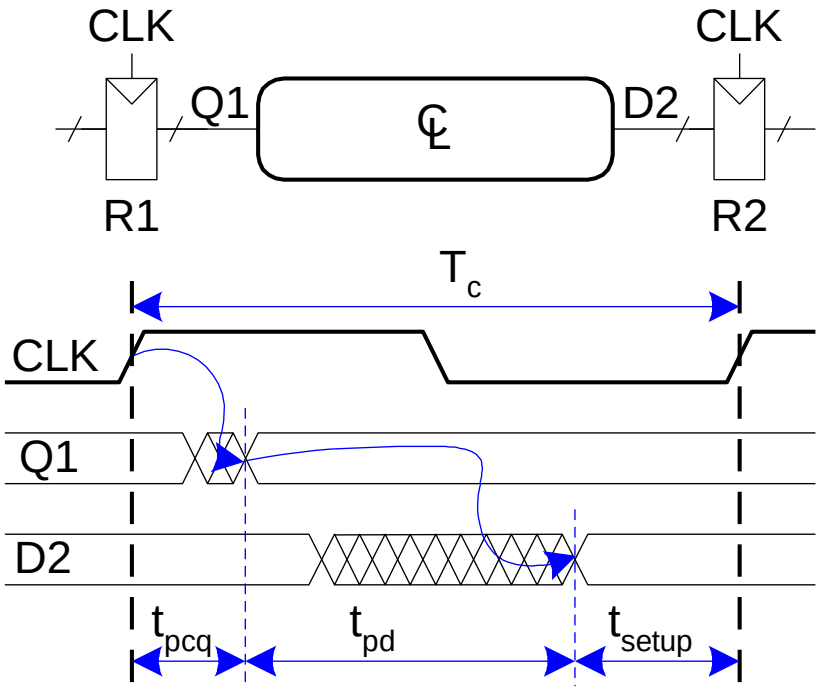
$T_c >= t_{pcq} + t_{pd} + t_{setup}$

$t_{pd} <=$

Setup Time Constraint

- **The setup time constraint depends on the maximum delay from register R1 through the combinational logic.**

- **The input to register R2 must be stable at least t_{setup} before the clock edge.**



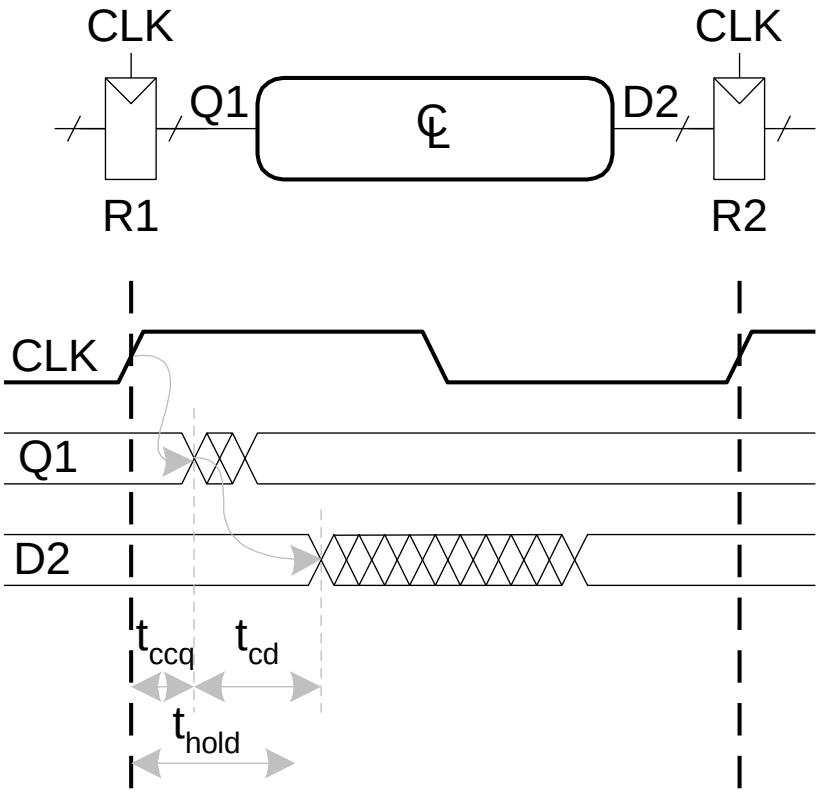
$$T_c >= t_{pcq} + t_{pd} + t_{setup}$$

$$t_{pd} <= T_c - (t_{pcq} + t_{setup})$$

Hold Time Constraint

- **The hold time constraint depends on the minimum delay from register R1 through the combinational logic.**

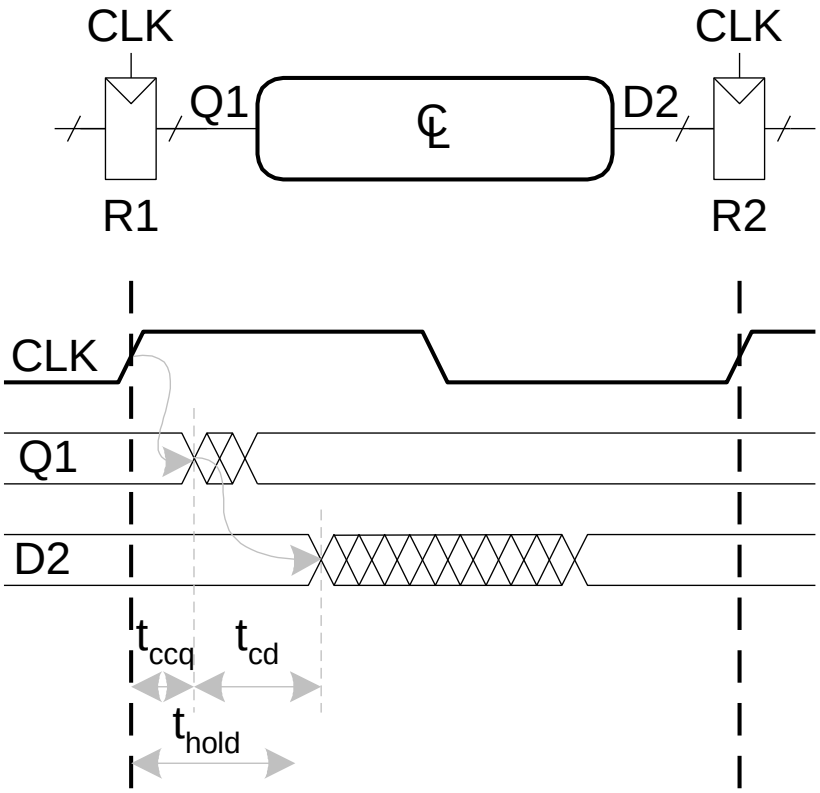
- **The input to register R2 must be stable for at least t_{hold} after the clock edge.**



t_{hold} <

Hold Time Constraint

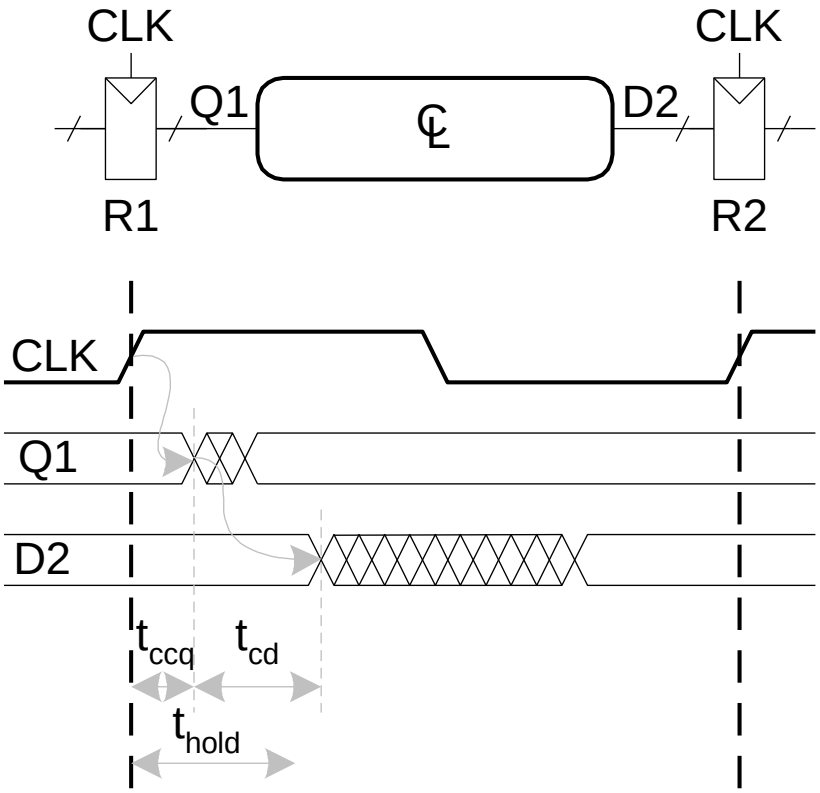
- **The hold time constraint depends on the minimum delay from register R1 through the combinational logic.**
- **The input to register R2 must be stable for at least t_{hold} after the clock edge.**



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

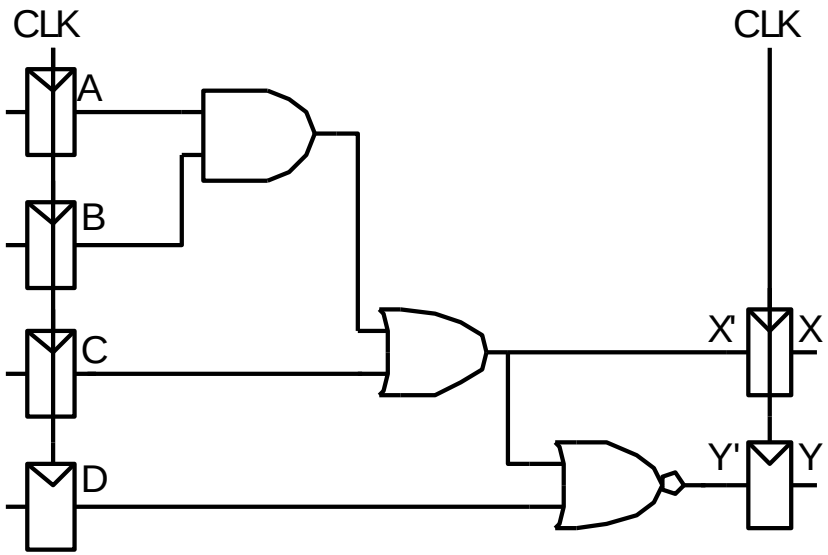
Hold Time Constraint

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



$$t_{hold} < t_{ccq} + t_{cd}$$
$$t_{cd} > t_{hold} - t_{ccq}$$

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} =

t_{cd} =

Setup time constraint:

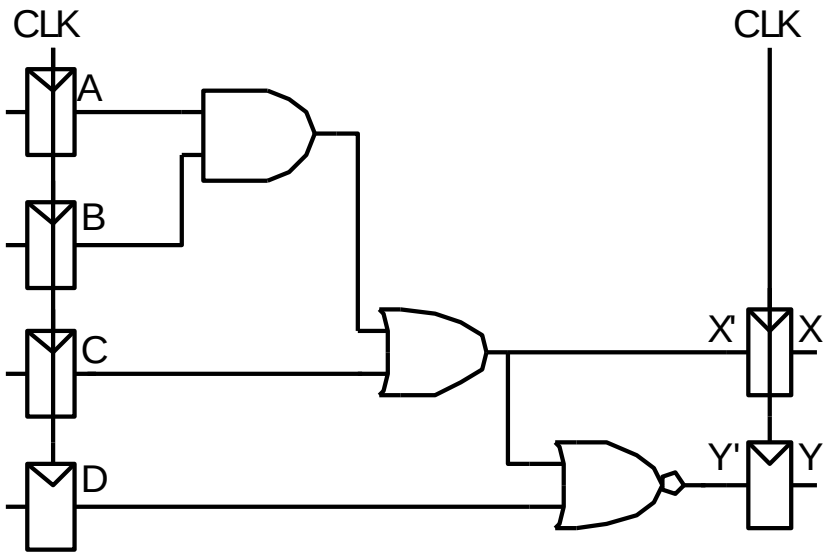
$T_c \geq$

$f_c = 1/T_c =$

Hold time constraint:

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

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:

$T_c \geq (50 + 105 + 60)$ ps = 215 ps

$f_c = 1/T_c$ = 4.65 GHz

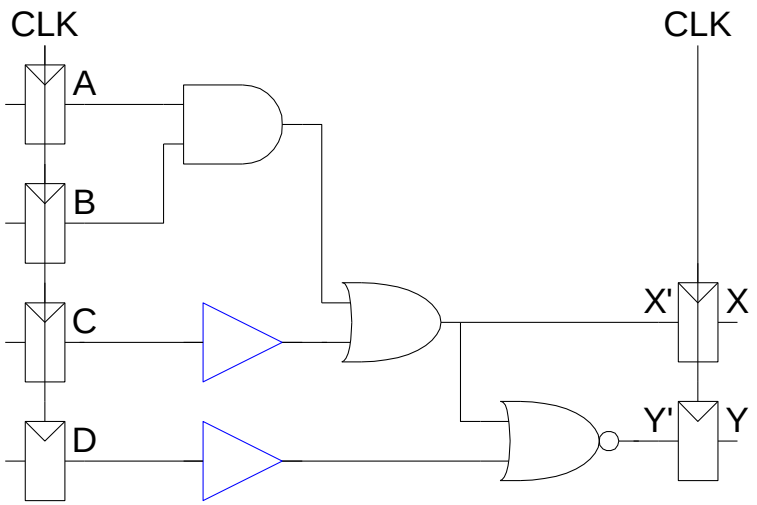
Hold time constraint:

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

(30 + 25) ps > 70 ps ? No!

Fixing Hold Time Violation

Add buffers to the short paths:



t_{pd} =

t_{cd} =

Setup time constraint:

$T_c \geq$

f_c =

Timing Characteristics

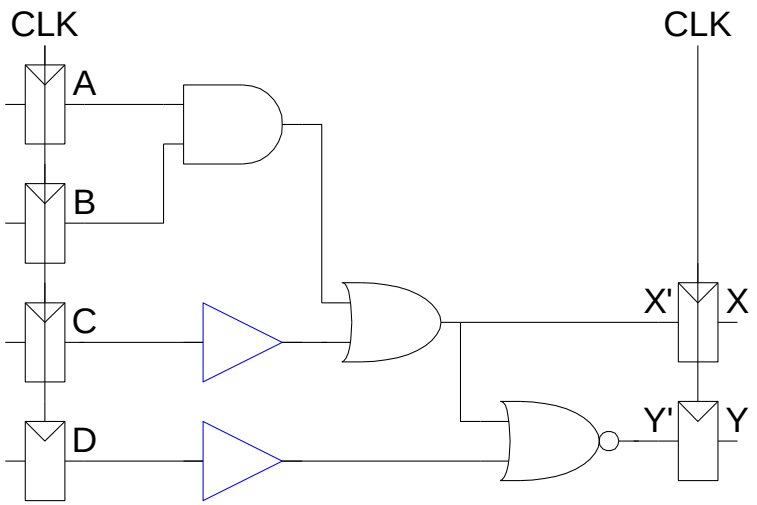
t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

Hold time constraint:

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

Fixing Hold Time Violation

Add buffers to the short paths:



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

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

Setup time constraint:

$T_c \geq$ (50 + 105 + 60) ps = 215 ps

$f_c = 1/T_c$ = 4.65 GHz

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

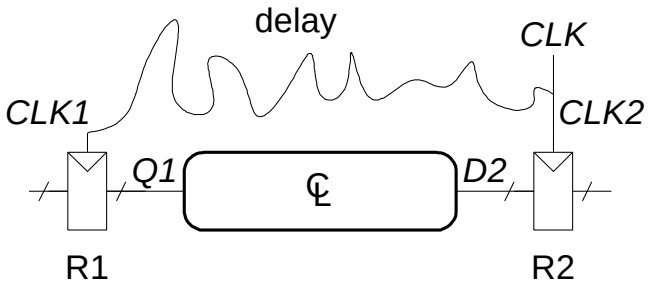
Hold time constraint:

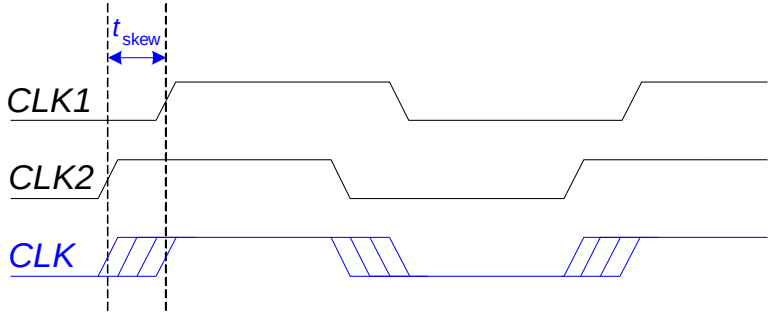
$t_{ccq} + t_{cd} > t_{hold}$?

(30 + 50) ps > 70 ps ? Yes!

Clock Skew

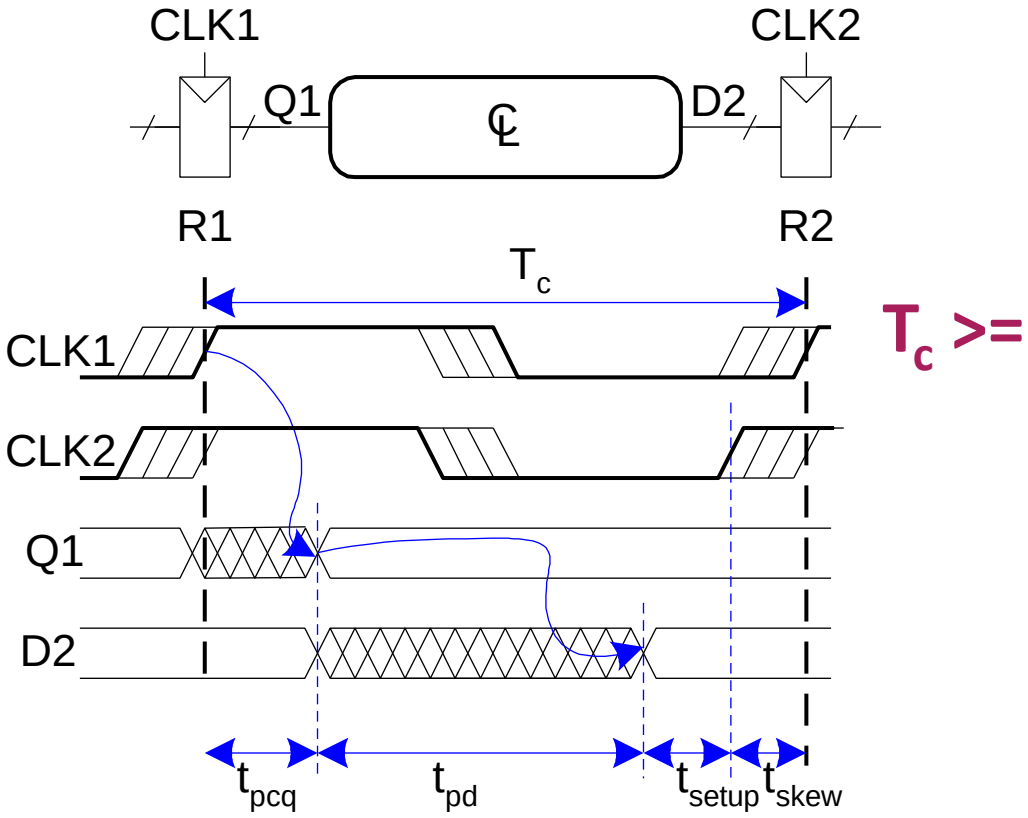
- The clock doesn't arrive at all registers at the same time
- Skew is the difference between two clock edges
- Examine the worst case to guarantee that the dynamic discipline is not violated for any register – many registers in a system!





Setup Time Constraint with Clock Skew

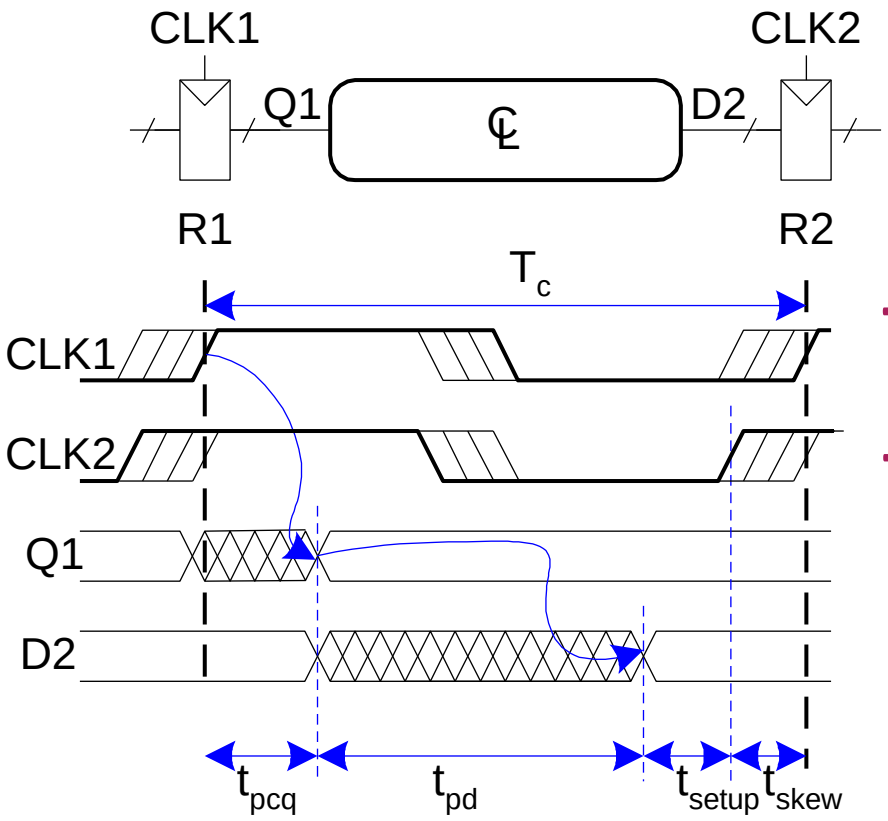
- In the worst case, the CLK2 is earlier than CLK1



$T_c >=$

Setup Time Constraint with Clock Skew

- In the worst case, the CLK2 is earlier than CLK1

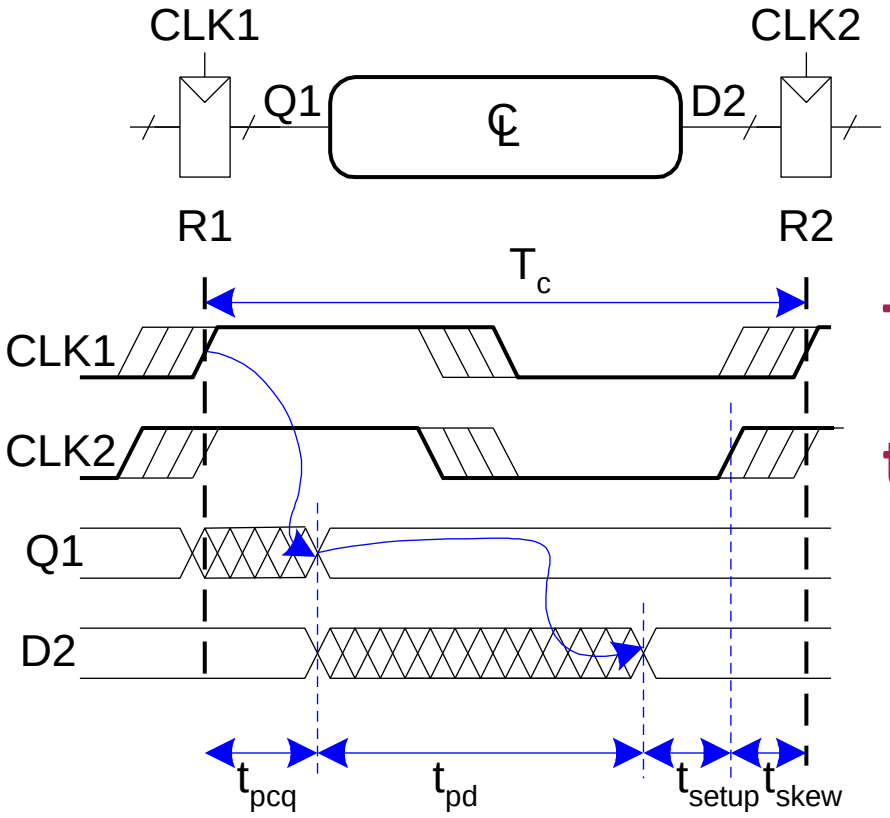


$T_c >= t_{pcq} + t_{pd} + t_{setup} + t_{skew}$

$t_{pd} <=$

Setup Time Constraint with Clock Skew

- In the worst case, the CLK2 is earlier than CLK1

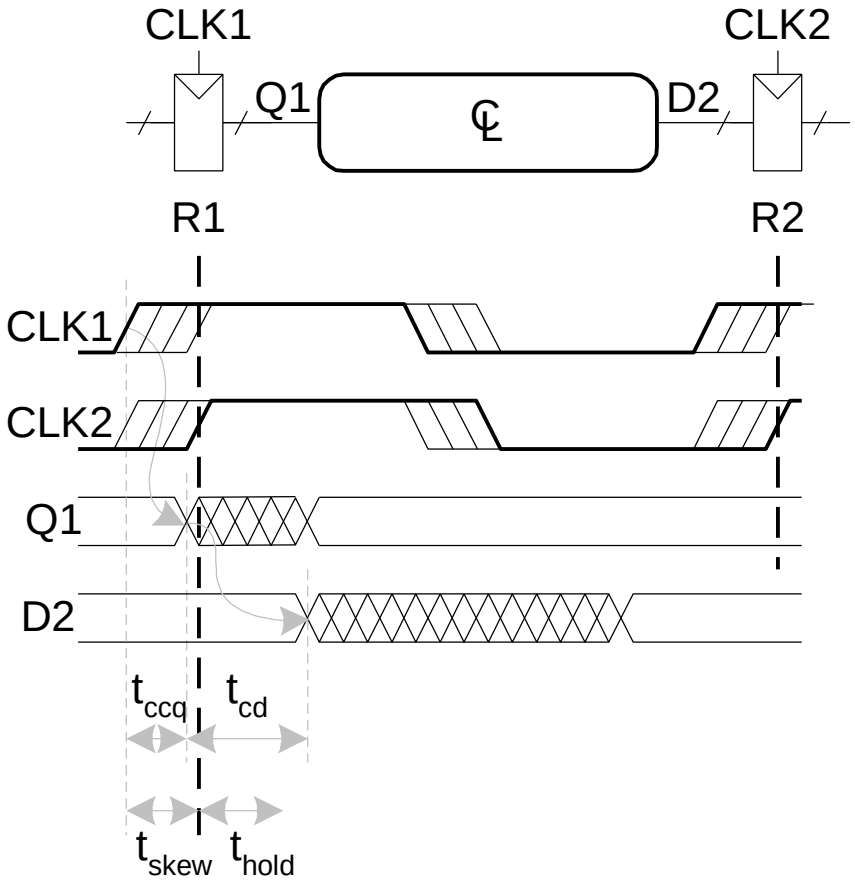


$$T_c >= t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$t_{pd} <= T_c - (t_{pcq} + t_{setup} + t_{skew})$$

Hold Time Constraint with Clock Skew

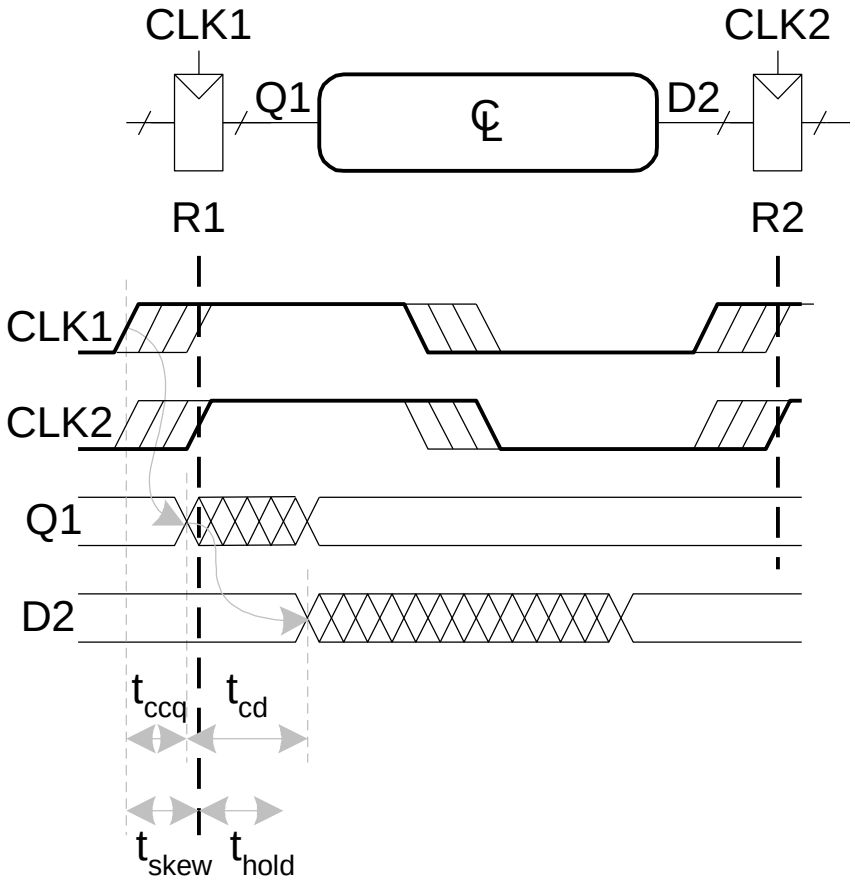
- In the worst case, CLK2 is later than CLK1



$t_{hold} + t_{skew} =<$

Hold Time Constraint with Clock Skew

- In the worst case, CLK2 is later than CLK1

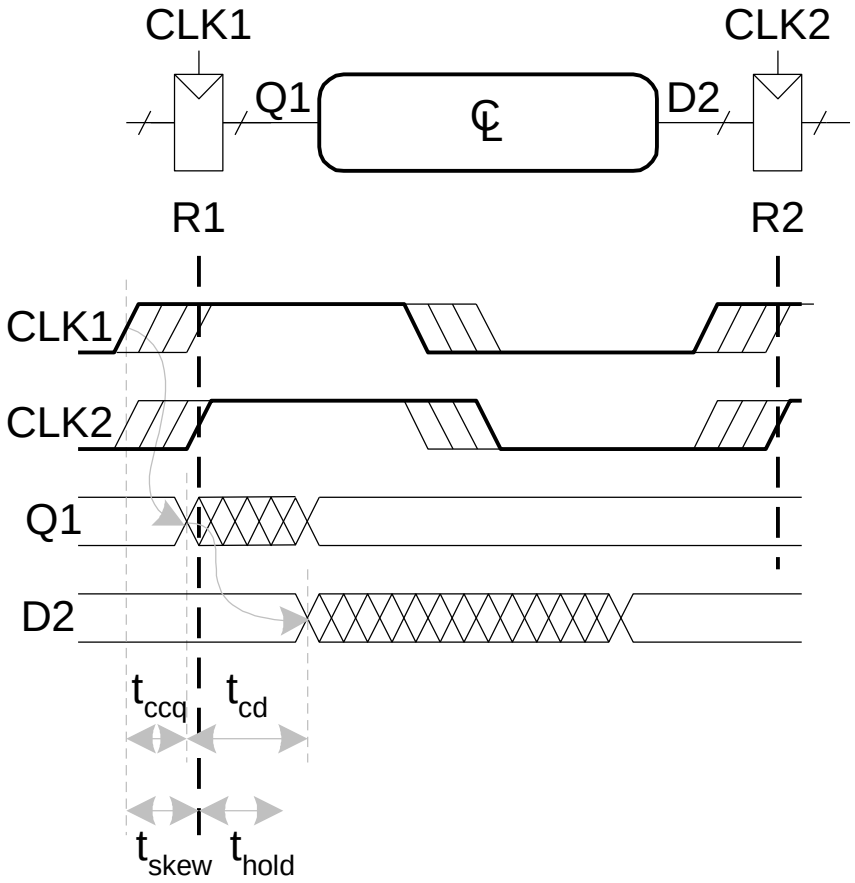


$$t_{hold} + t_{skew} =< t_{ccq} + t_{cd}$$

$$t_{cd} >=$$

Hold Time Constraint with Clock Skew

- In the worst case, CLK2 is later than CLK1

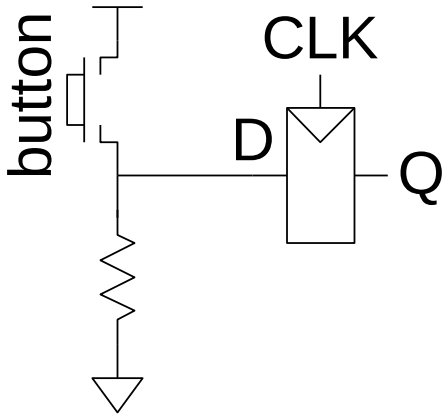


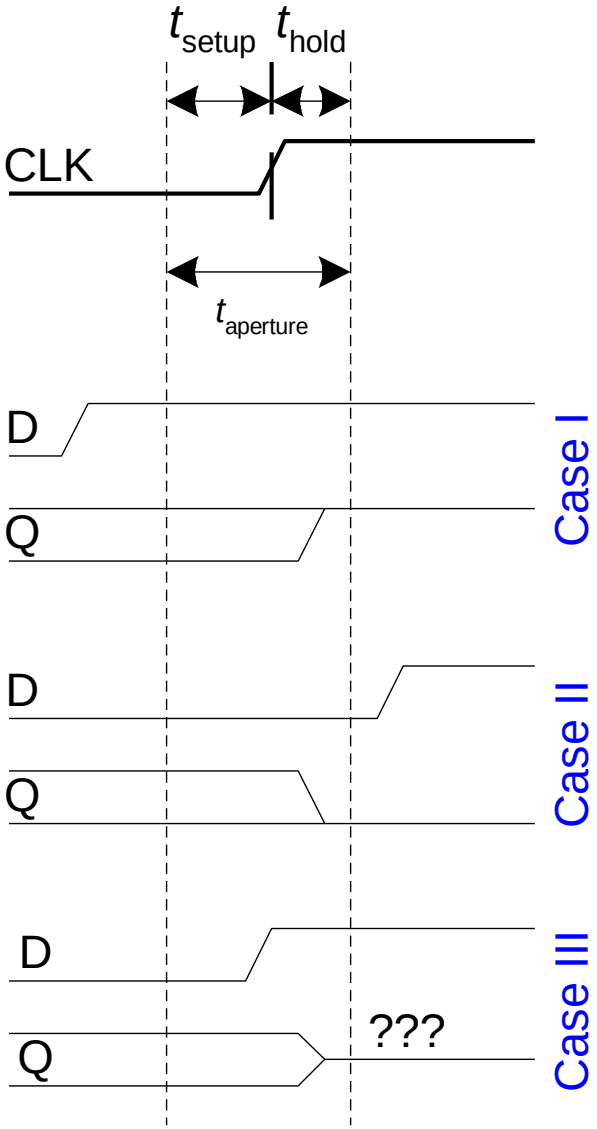
$$t_{hold} + t_{skew} =< t_{ccq} + t_{cd}$$

$$t_{cd} >= t_{hold} + t_{skew} - t_{ccq}$$

Violating the Dynamic Discipline

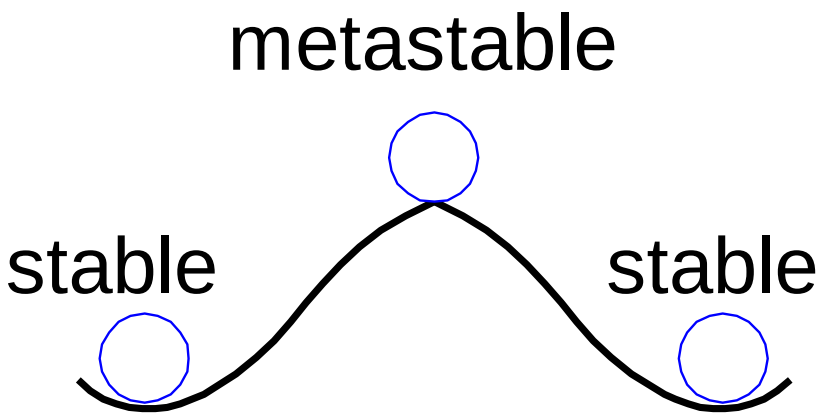
- **Asynchronous (for example, user) inputs might violate the dynamic discipline**





Metastability

- **Any bi-stable device has two stable states and a metastable state between them**
- **A flip-flop has two stable states (1 and 0) and one metastable state**
- **If a flip-flop lands in the metastable state, it could stay there for an undetermined amount of time**



Metastability

- T_0/T_c describes the probability that the input changes at a bad time, i.e., during the aperture time

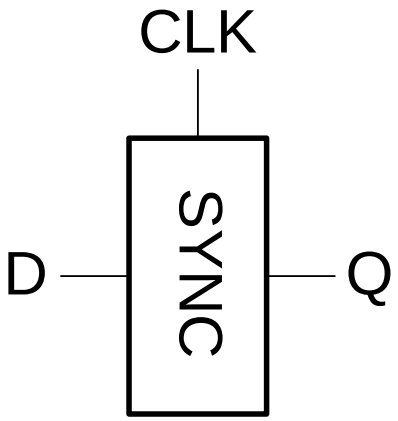
$$P(t_{res} > t) = (T_0/T_c)\, e^{-t/\tau}$$

- τ is a time constant indicating how fast the flip-flop moves away from the metastable state; it is related to the delay through the cross-coupled gates in the flip-flop

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

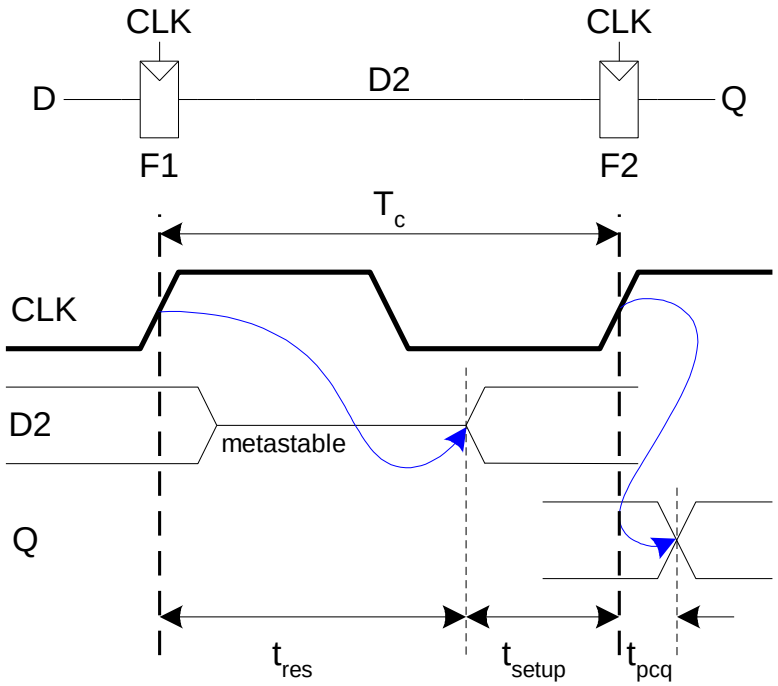
Synchronizers

- Asynchronous inputs (D) are inevitable (user interfaces, systems with different clocks interacting, etc.).
- The goal of a synchronizer is to make the probability of failure (the output Q still being metastable) low.
- A synchronizer cannot make the probability of failure 0.



Synchronizer Internals

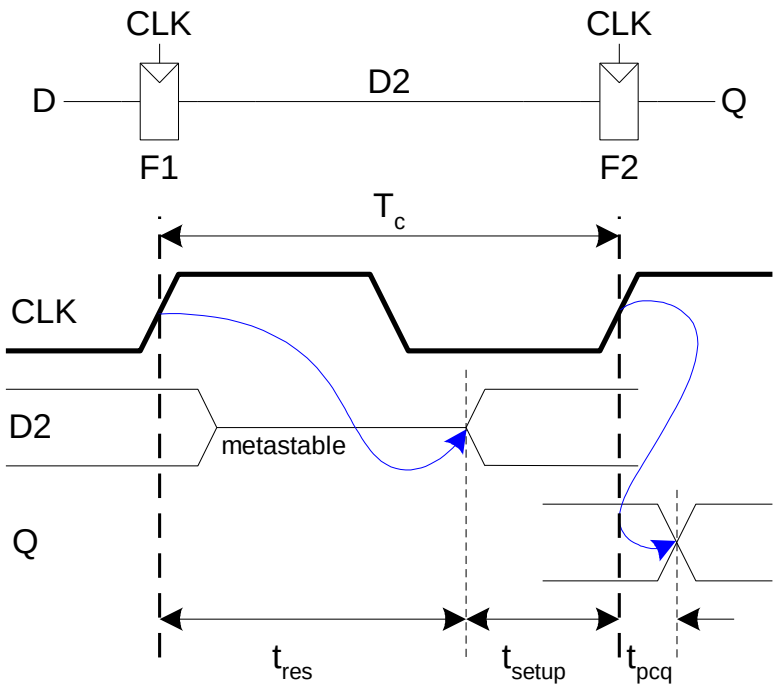
- A synchronizer can be built with two back-to-back flip-flops.
- Suppose the input D is changing when it is sampled by F1.
- Internal signal D2 has (T_c - t_{setup}) time to resolve a 1 or 0.



Synchronizer Probability of Failure

- **For each sample, the probability of failure of this synchronizer is:**

$$P(\text{failure}) = \frac{T_0}{T_c} e^{-\frac{T_c - t_{\text{setup}}}{\tau}}$$



Synchronizer Mean Time Before Failure

- If the asynchronous input changes once per second, the probability of failure per second of the synchronizer is simply P(failure).

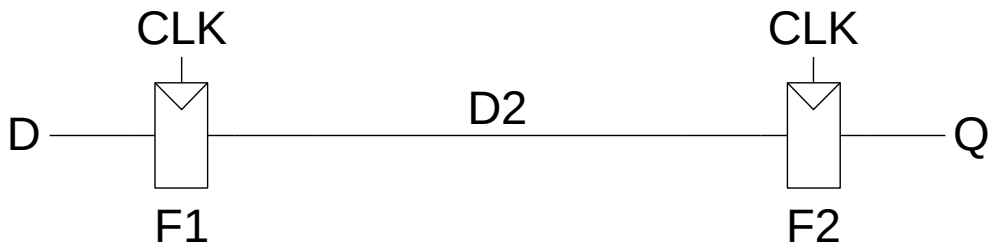
- In general, if the input changes N times per second, the probability of failure per second of the synchronizer is:

$$P(\text{failure})/\text{sec} = N \frac{T_0}{T_c} e^{-\frac{T_c - t_{\text{setup}}}{\tau}}$$

- Thus, the synchronizer fails, on average,1/[P(failure)/second] This is called the mean time between failures, MTBF:

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

Example Synchronizer



Suppose: T_c = 1/500 MHz = 2 ns τ = 200 ps

T_0 = 150 ps t_{setup} = 100 ps

N = 10 events per second

What is the probability of failure? MTBF?

P(failure) =

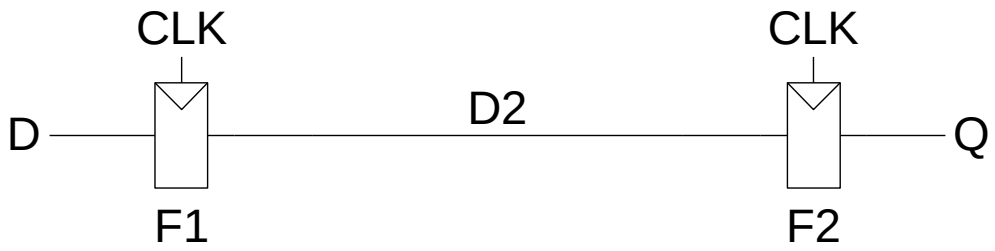
=

P(failure)/second =

=

MTBF =

Example Synchronizer



Suppose: T_c = 1/500 MHz = 2 ns τ = 200 ps

T_0 = 150 ps t_{setup} = 100 ps

N = 10 events per second

What is the probability of failure? MTBF?

P(failure) = (150 ps/2 ns) $e^{-(1.9\ ns)/200\ ps}$

= 5.6×10^{-6}

P(failure)/second = $10 \times (5.6 \times 10^{-6})$

= 5.6×10^{-5} / second

MTBF = 1/[P(failure)/second] ≈ 5 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

- **Some definitions:**
  - ***Token***: A group of inputs processed to produce a group of outputs
  - ***Latency***: Time for one token to pass from start to end
  - ***Throughput***: The number of tokens that can be produced per unit time
- **Parallelism increases throughput.**

Parallelism Example

- **Example:**

  Ben Bitdiddle is baking cookies to celebrate the installation of his traffic light controller. It takes 5 minutes to roll the cookies and 15 minutes to bake them. After finishing one batch he immediately starts the next batch. What is the latency and throughput if Ben doesn't use parallelism?

Latency =

Throughput =

Parallelism Example

- **Example:**

  Ben Bitdiddle is baking cookies to celebrate the installation of his traffic light controller. It takes 5 minutes to roll the cookies and 15 minutes to bake them. After finishing one batch he immediately starts the next batch. What is the latency and throughput if Ben doesn't use parallelism?

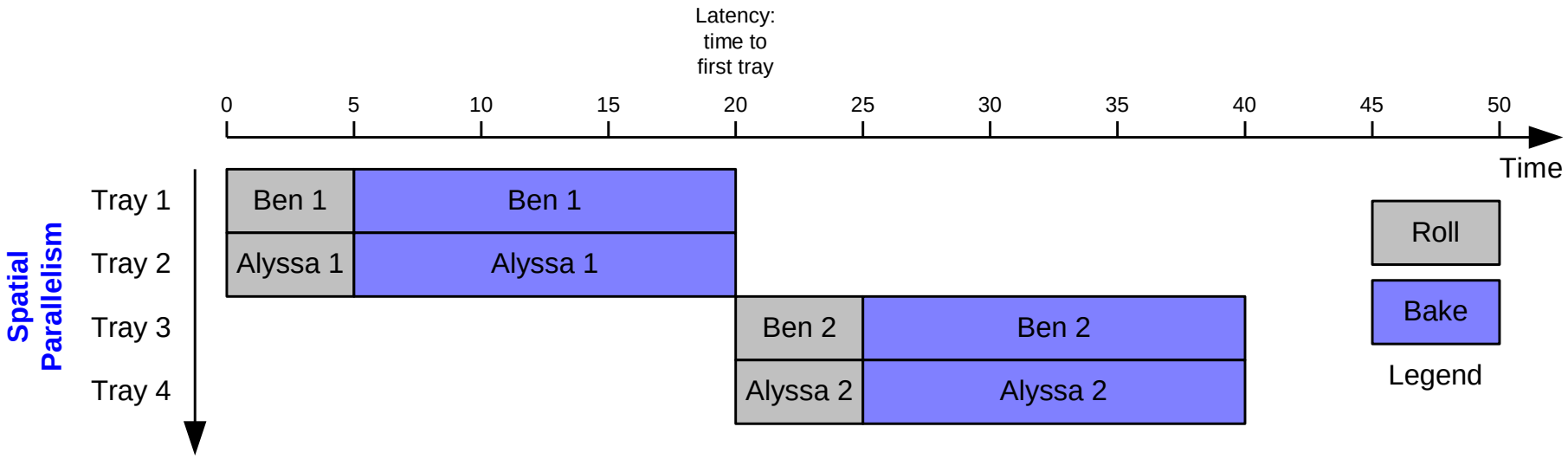
Latency **= 5 + 15 = 20 minutes** **= 1/3 hour**

Throughput **= 1 tray/ 1/3 hour** **= 3 trays/hour**

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**: Ben breaks the task into two stages: roll and baking. He uses two trays. While the first batch is baking he rolls the second batch, and so on.

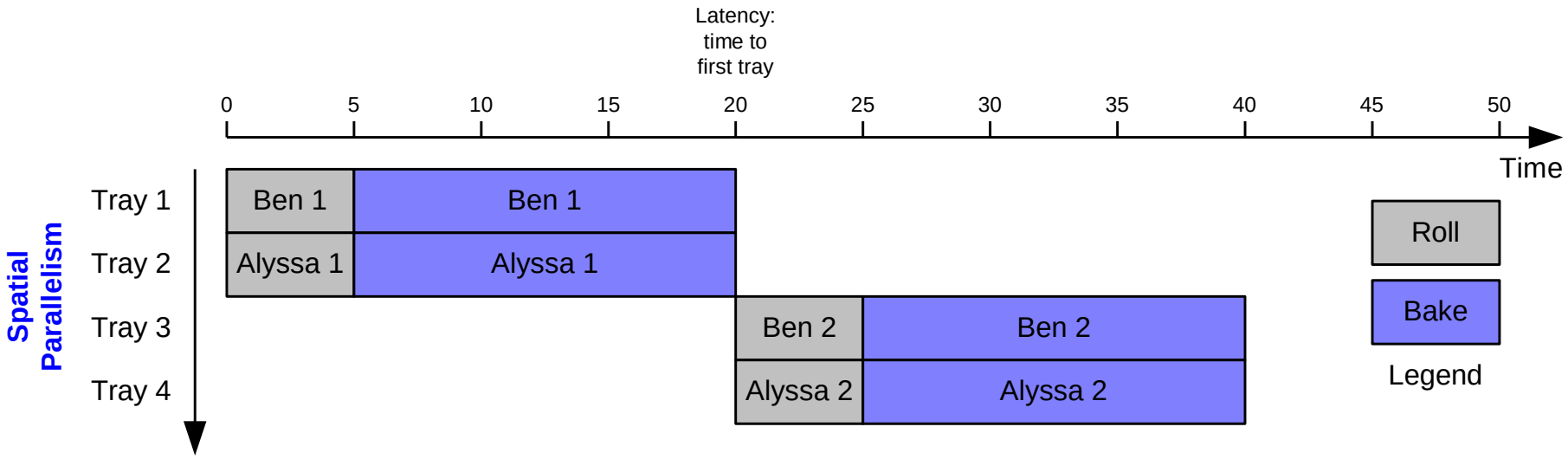
Spatial Parallelism



Latency =

Throughput=

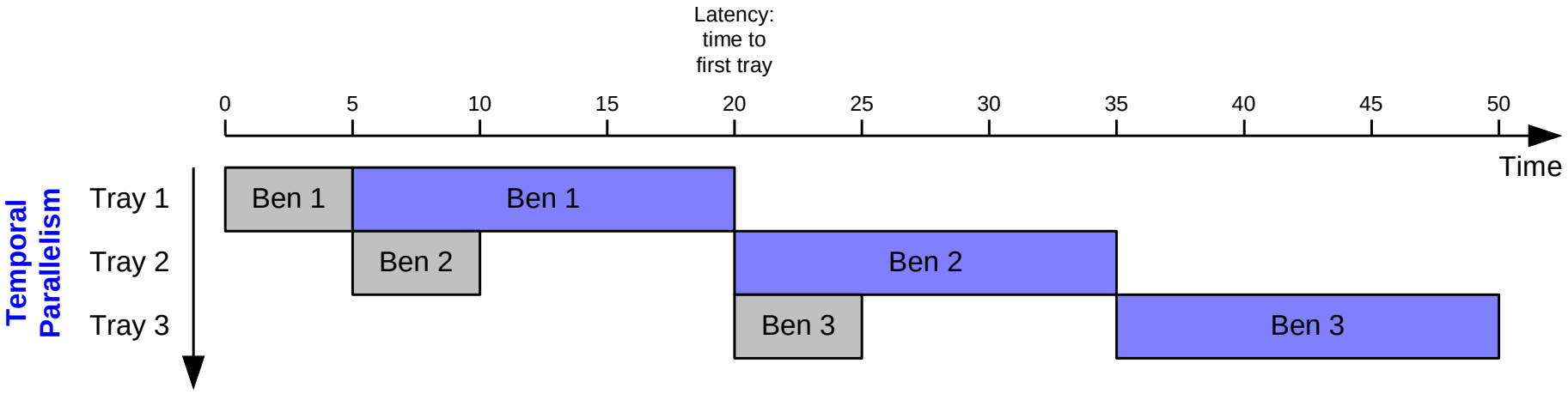
Spatial Parallelism



Latency = 5 + 15 = 20 minutes = 1/3 hour

Throughput= 2 trays/ 1/3 hour = 6 trays/hour

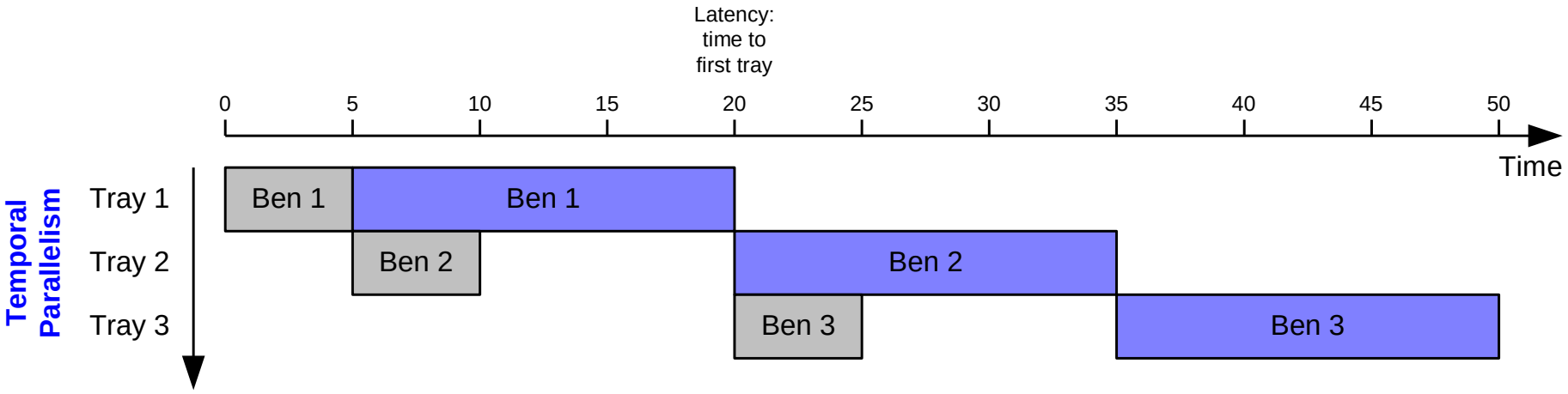
Temporal Parallelism



Latency =

Throughput=

Temporal Parallelism



Latency = 5 + 15 = 20 minutes = 1/3 hour

Throughput= 1 trays/ 1/4 hour = 4 trays/hour

Using both techniques, the throughput would be 8 trays/hour

What did we learn ?

- **Timing constraints for sequential circuits**
  - ***Setup time***, time needed for the inputs to be present before clock
  - ***Hold time***, time needed after the clock where inputs should not change
  - ***Aperture time***, time around clock event where inputs should stay stable
- **Problems related to clock skew**
- **Metastability and Synchronization**
- **Parallelism**
  - Spatial parallelism, more than one unit to the same job
  - Temporal parallelism, breaking job into multiple parts