SEQUENTIAL LOGIC DESIGN

Lesson 13

Digital Logic

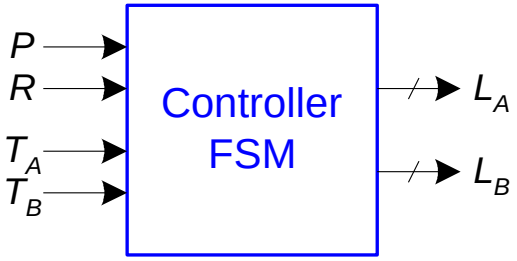
Junying Chen

Factoring State Machines

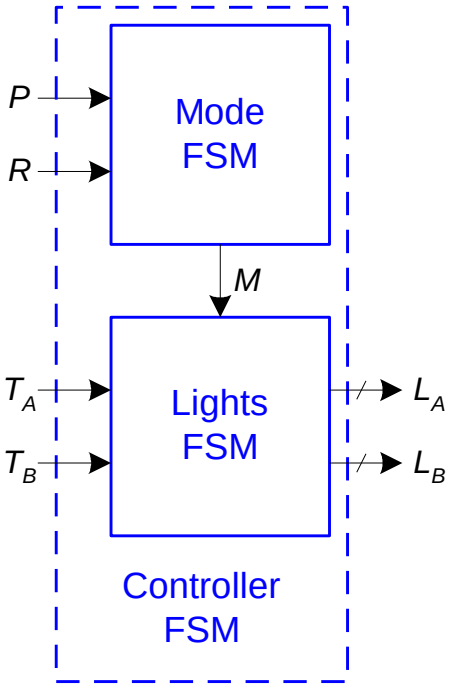
- Break complex FSMs into smaller interacting FSMs
- Example: Modify traffic light controller to have Parade Mode.
  - Two more inputs: *P*, *R*
  - When ***P* = 1**, enter Parade Mode & Bravado Blvd light stays green
  - When ***R* = 1**, leave Parade Mode

Parade FSM

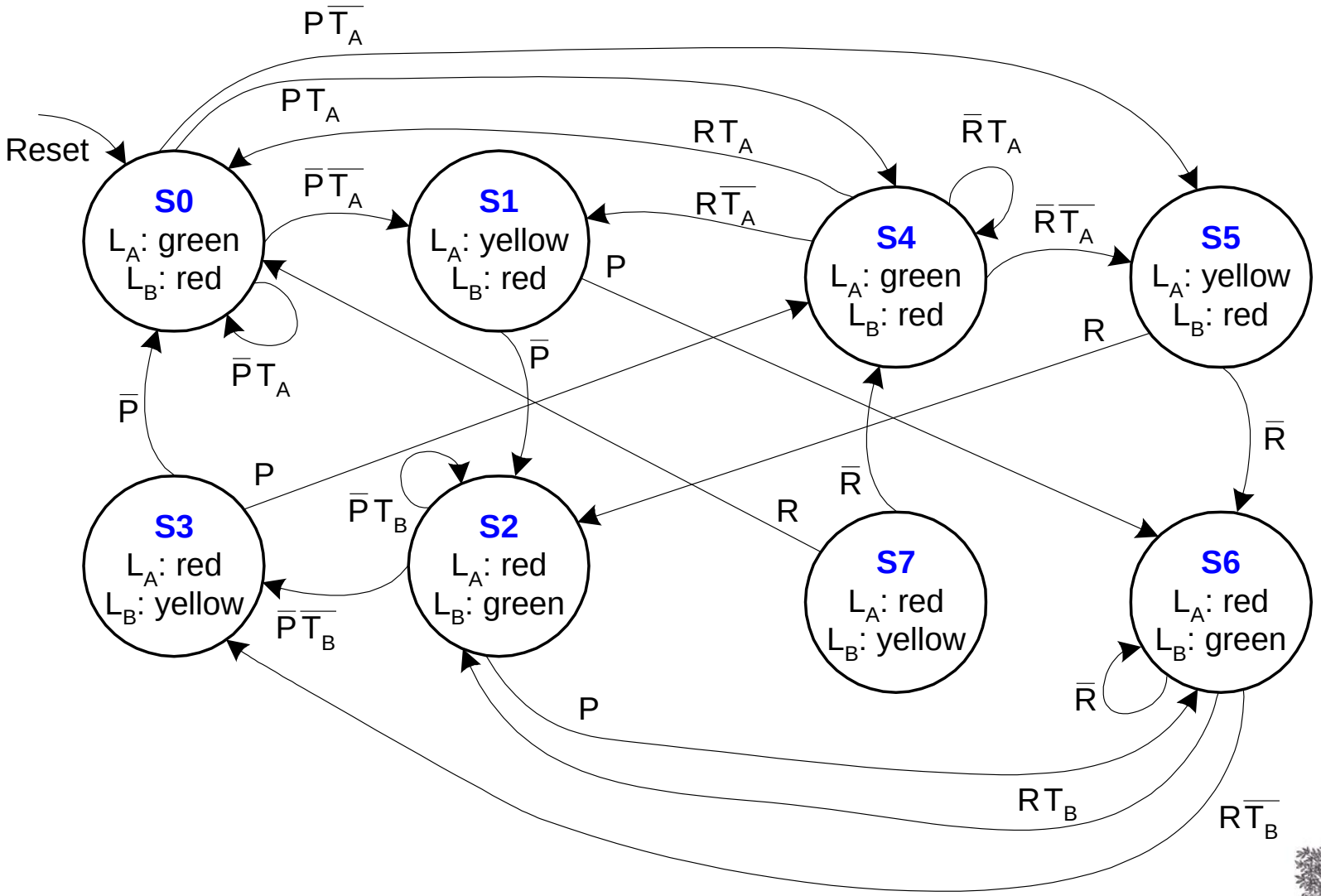
Unfactored FSM



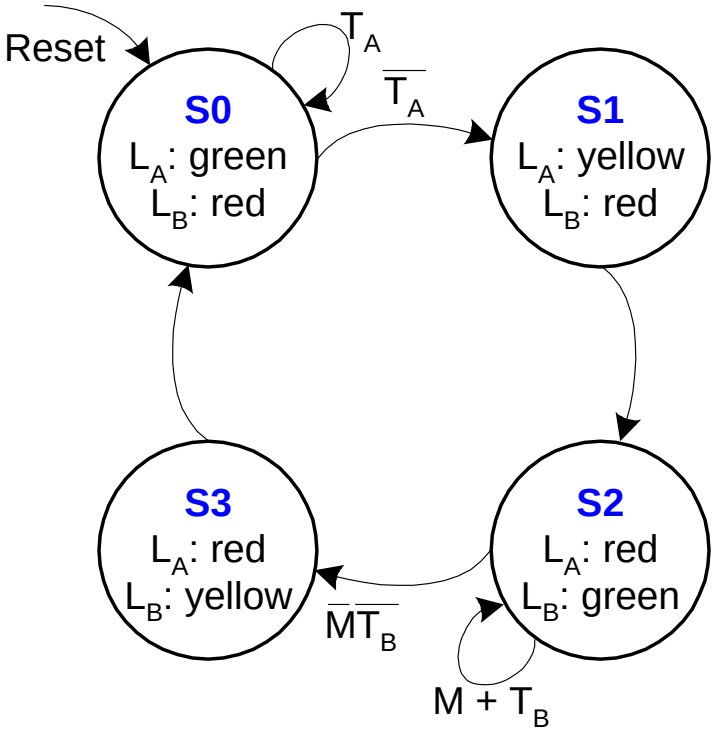
Factored FSM



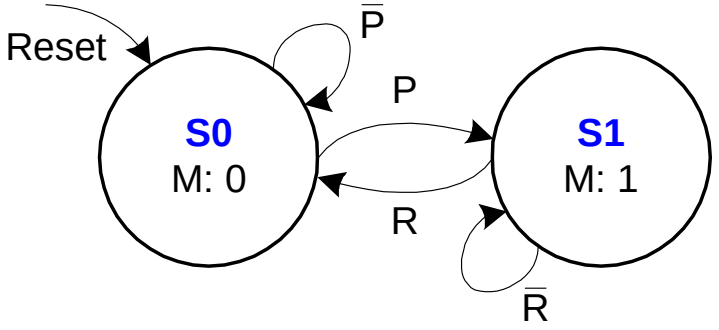
Unfactored FSM



Factored FSM



Lights FSM



Mode FSM

FSM Design Procedure

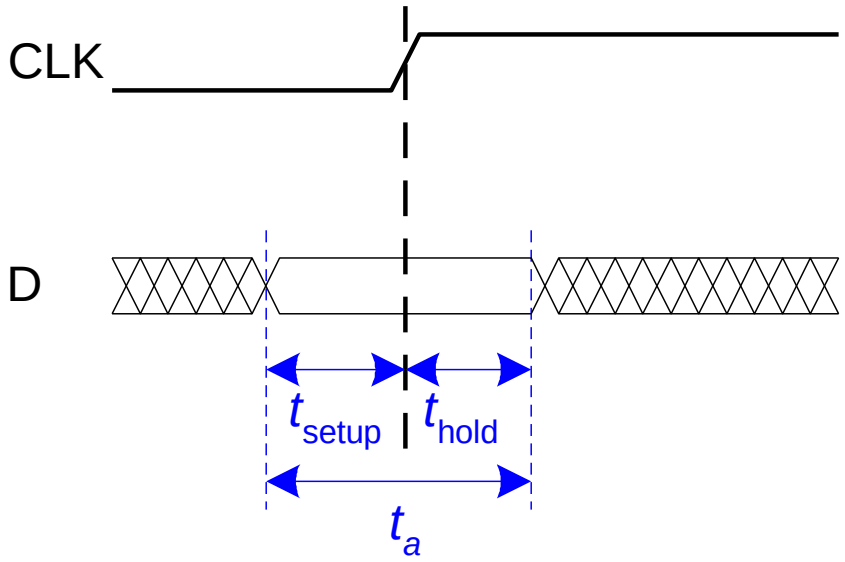
1. Identify inputs and outputs
2. Sketch state transition diagram
3. Write state transition table
4. Select state encodings
5. For Moore machine:
   1. Rewrite state transition table with state encodings
   2. Write output table
6. For a Mealy machine:
   1. Rewrite combined state transition and output table with state encodings
7. Write Boolean equations for next state and output logic
8. Sketch the circuit schematic

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

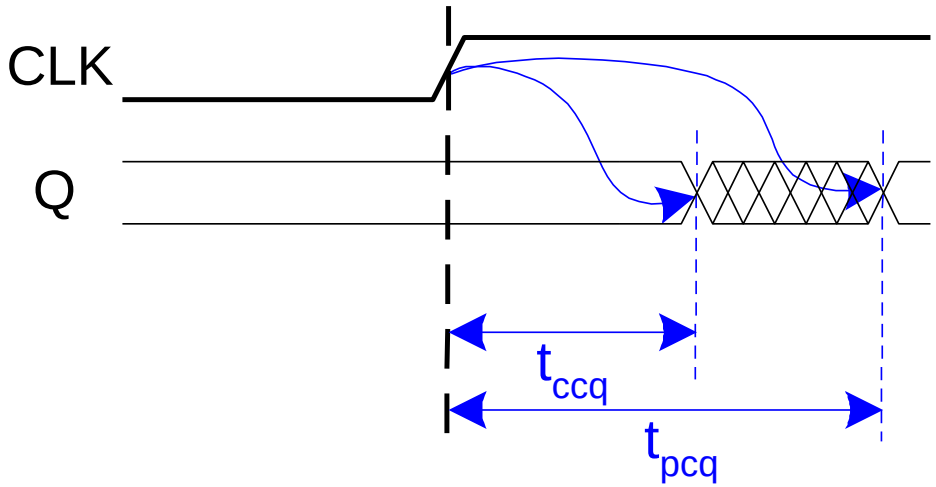
Input Timing Constraints

- **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 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)

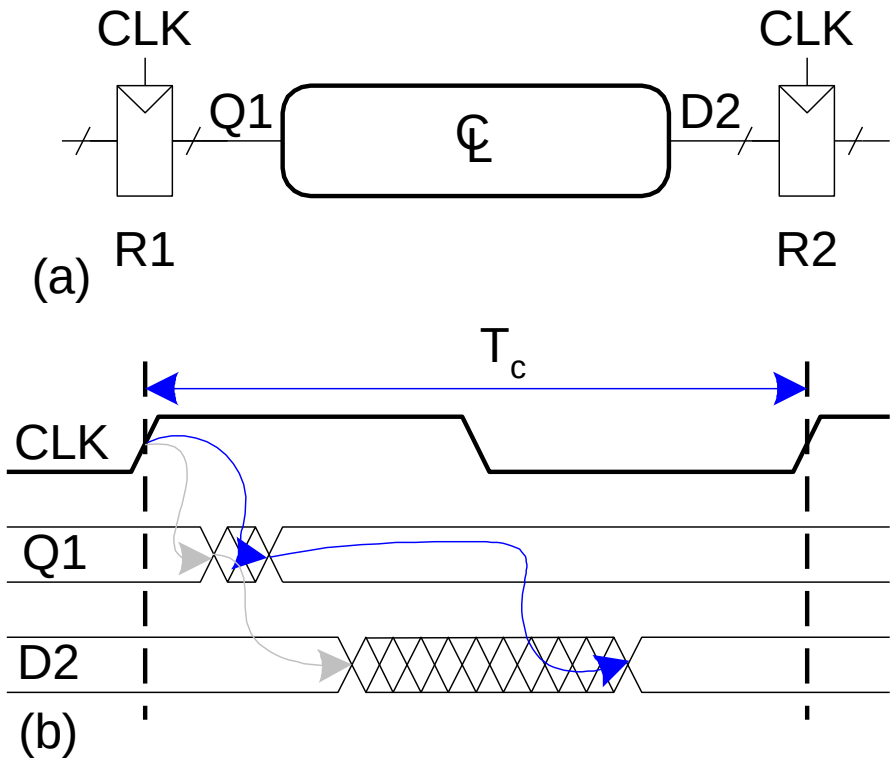


Dynamic Discipline

- 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

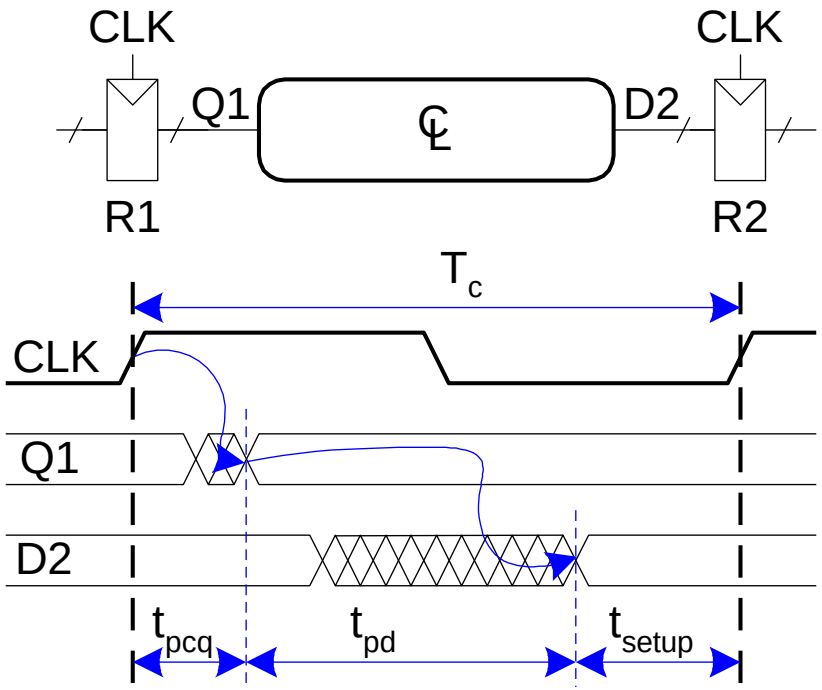
Dynamic Discipline

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



Setup Time Constraint

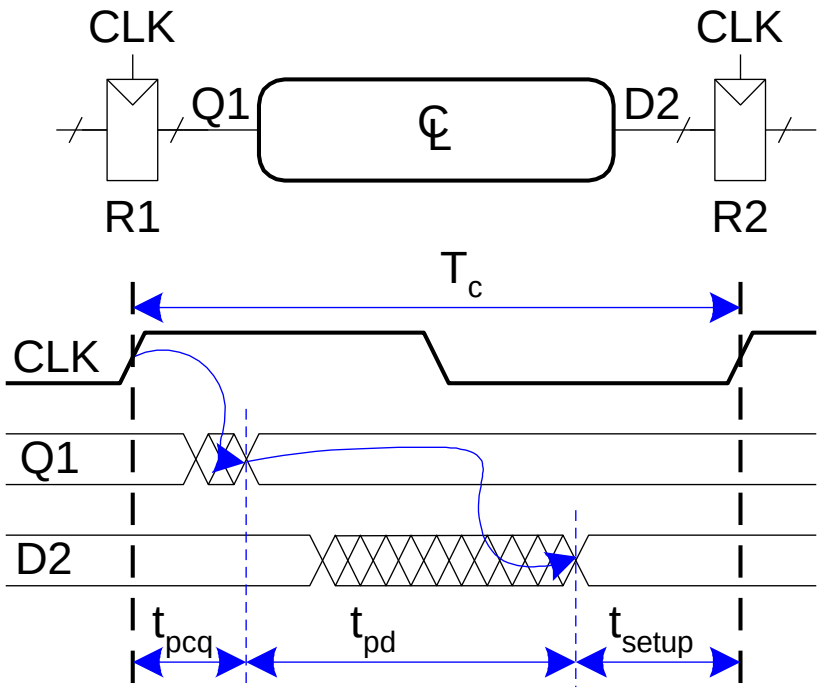
- Depends on the **maximum** delay from register R1 through combinational logic to R2
- The input to register R2 must be stable at least t_{setup} before clock edge



$$T_c \geq$$

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

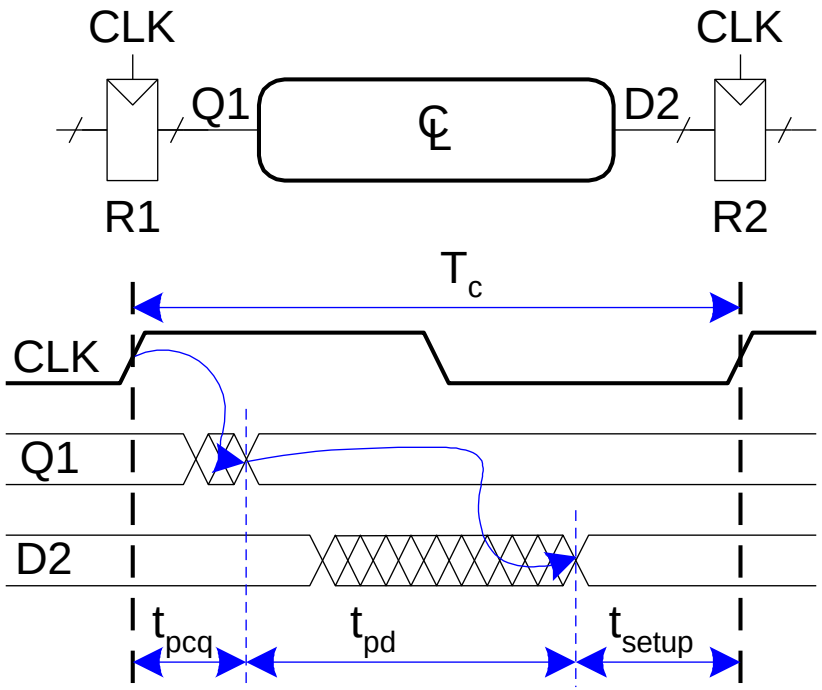


$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

$$t_{pd} \leq$$

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



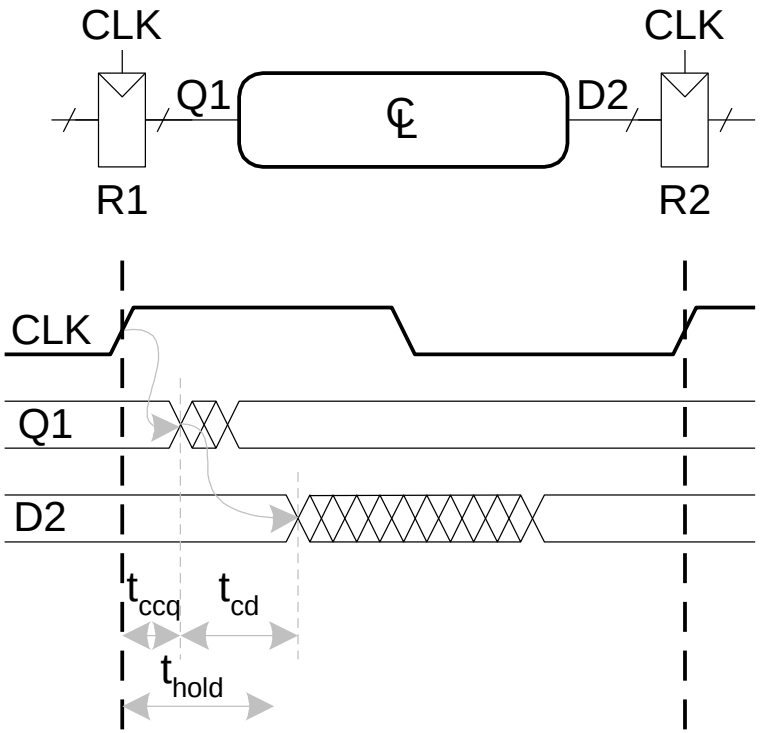
$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

$$t_{pd} \leq T_c - (t_{pcq} + t_{setup})$$

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

Hold Time Constraint

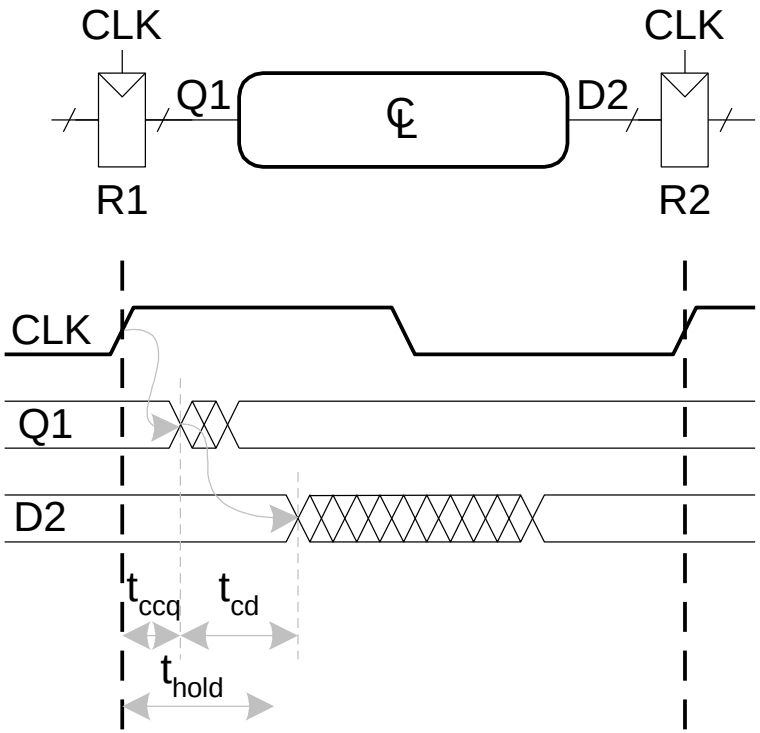
- Depends on the **minimum** delay from register R1 through the combinational logic to R2
- The input to register R2 must be stable for at least t_{hold} after the clock edge



t_{hold} <

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

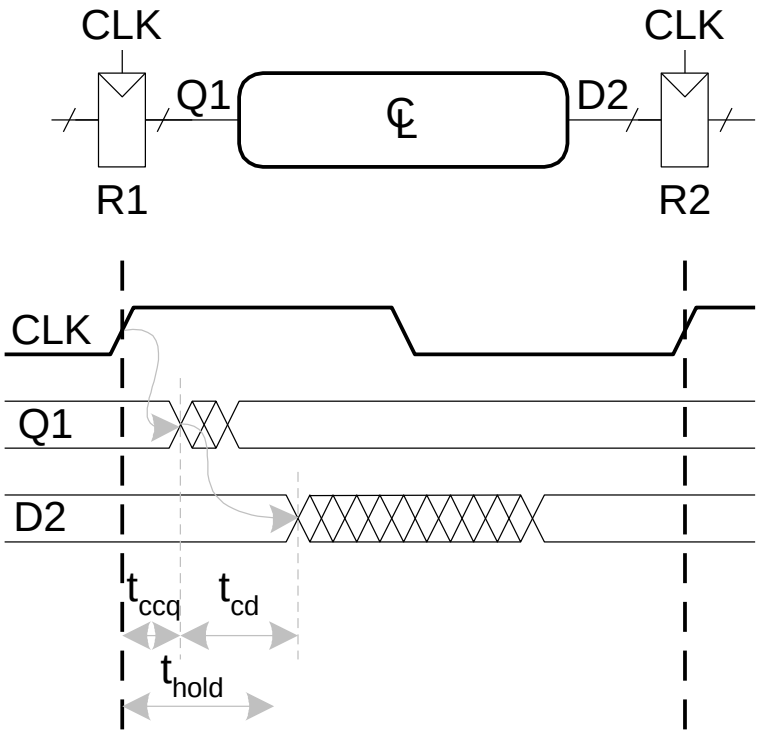


$$t_{hold} < t_{ccq} + t_{cd}$$

$$t_{cd} >$$

Hold Time Constraint

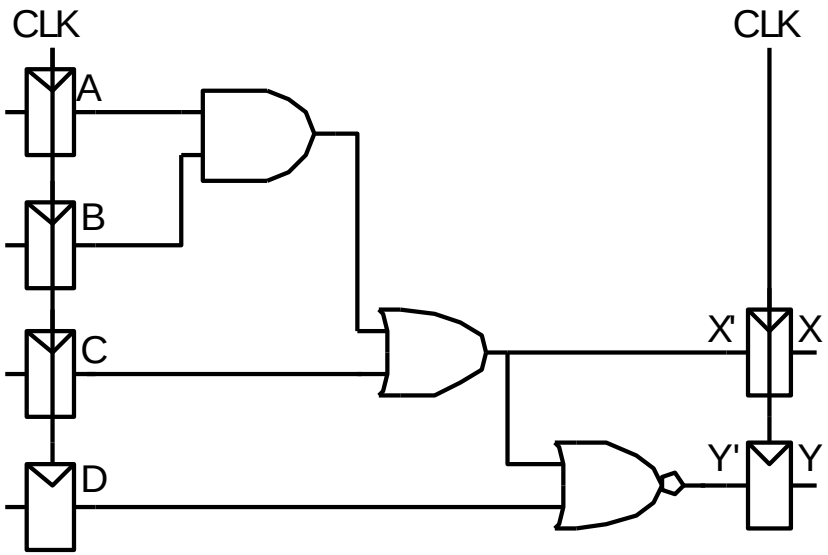
- Depends on the **minimum** delay from register R1 through the combinational logic to R2
- The input to register R2 must be stable for at least t_{hold} after the clock edge



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

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

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:

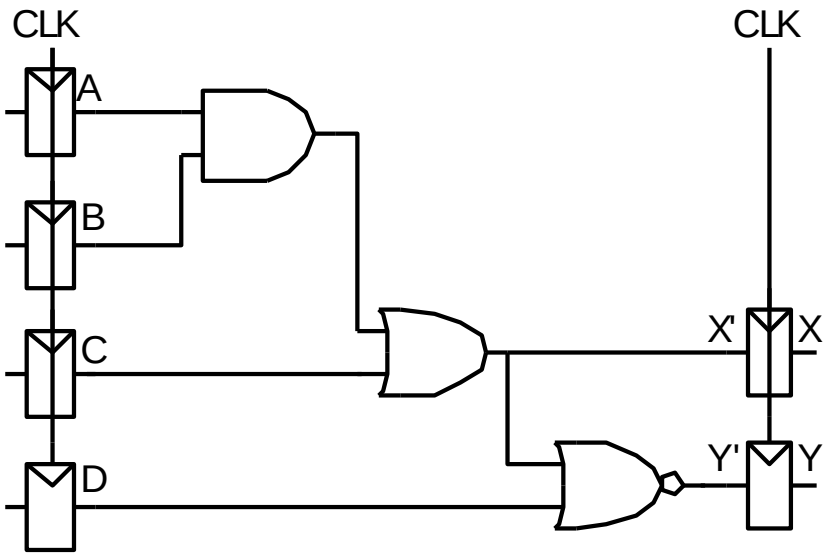
$T_c \geq$

f_c =

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

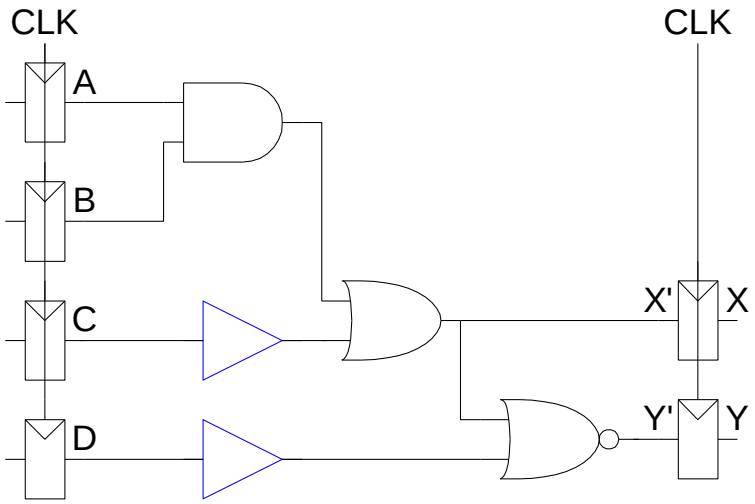
Hold time constraint:

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

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

Timing Analysis

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

per gate:

t_{pd} = 35 ps

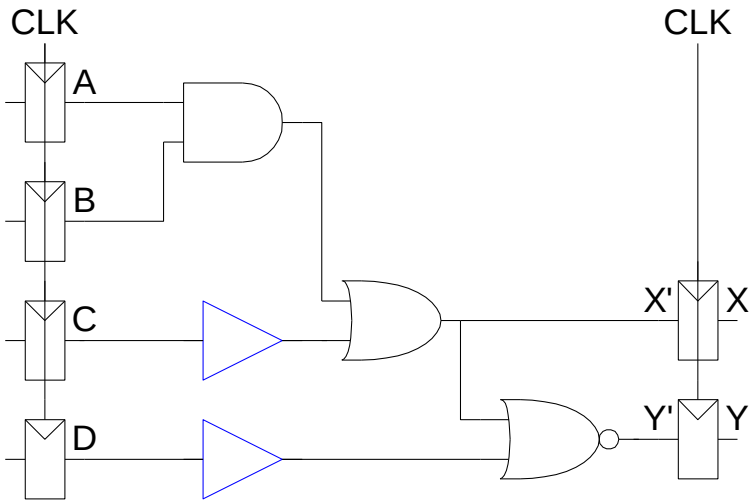
t_{cd} = 25 ps

Hold time constraint:

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

Timing Analysis

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

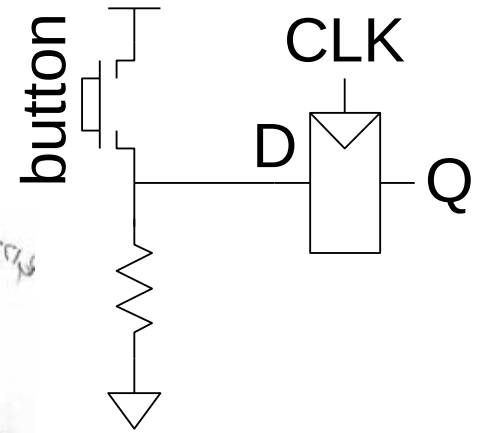
Hold time constraint:

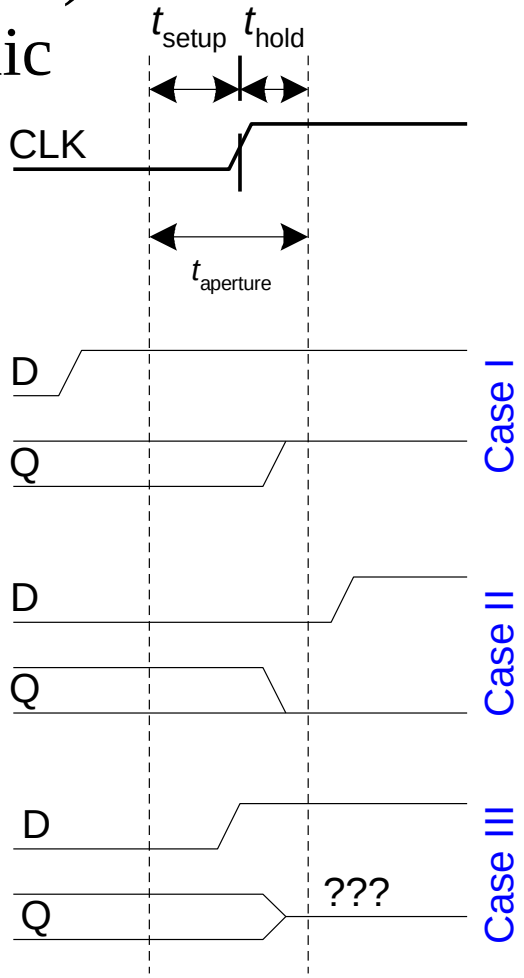
$t_{ccq} + t_{cd} > t_{hold}$?

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

Violating Dynamic Discipline

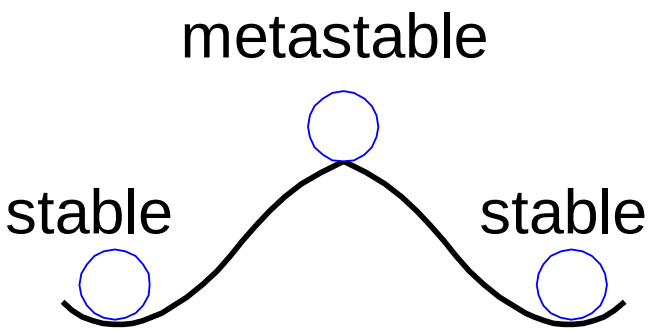
- Asynchronous (for example, user) inputs might violate the dynamic discipline





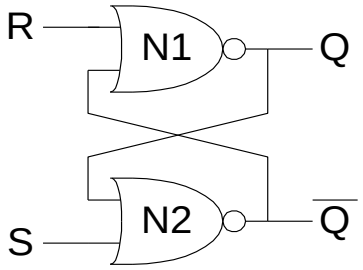
Metastability

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



Flip-Flop Internals

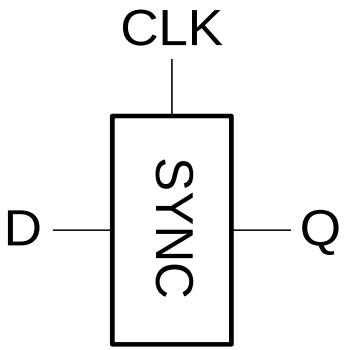
- Flip-flop has **feedback**: if *Q* is somewhere between 1 and 0, cross-coupled gates drive output to either rail (1 or 0)



- **Metastable signal:** if output hasn't resolved to 1 or 0
- If flip-flop input changes at random time (flip-flop samples metastable input), if you wait long enough time, the **metastable** output will resolve to 1 or 0 with **high probability**

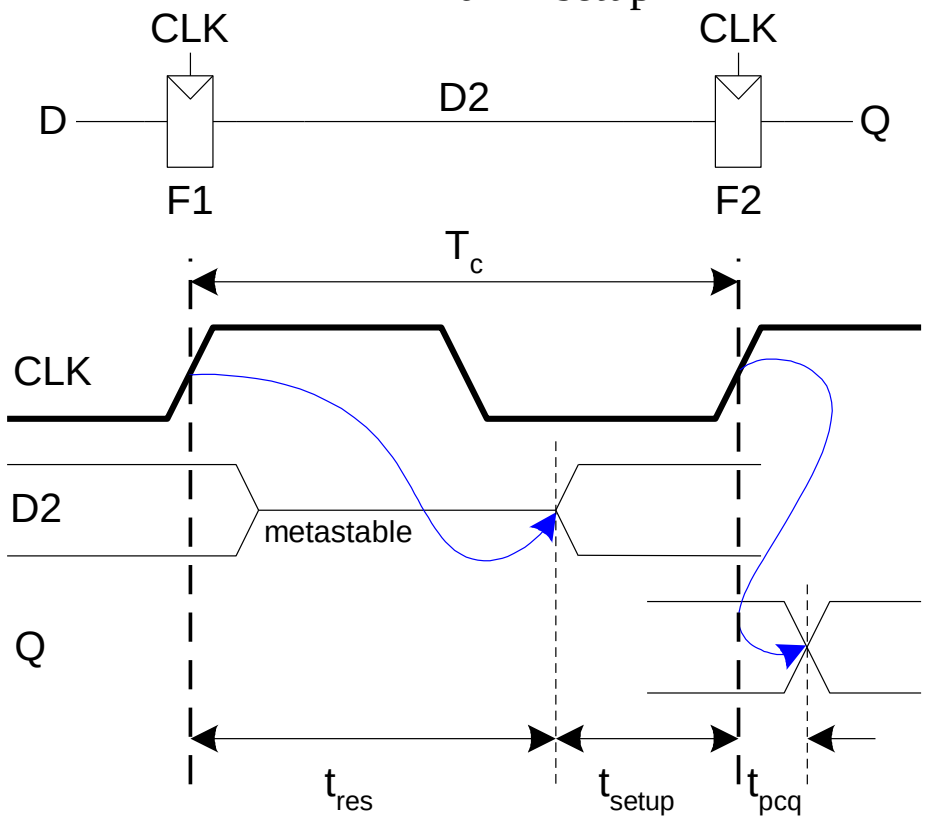
Synchronizers

- **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



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?

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?

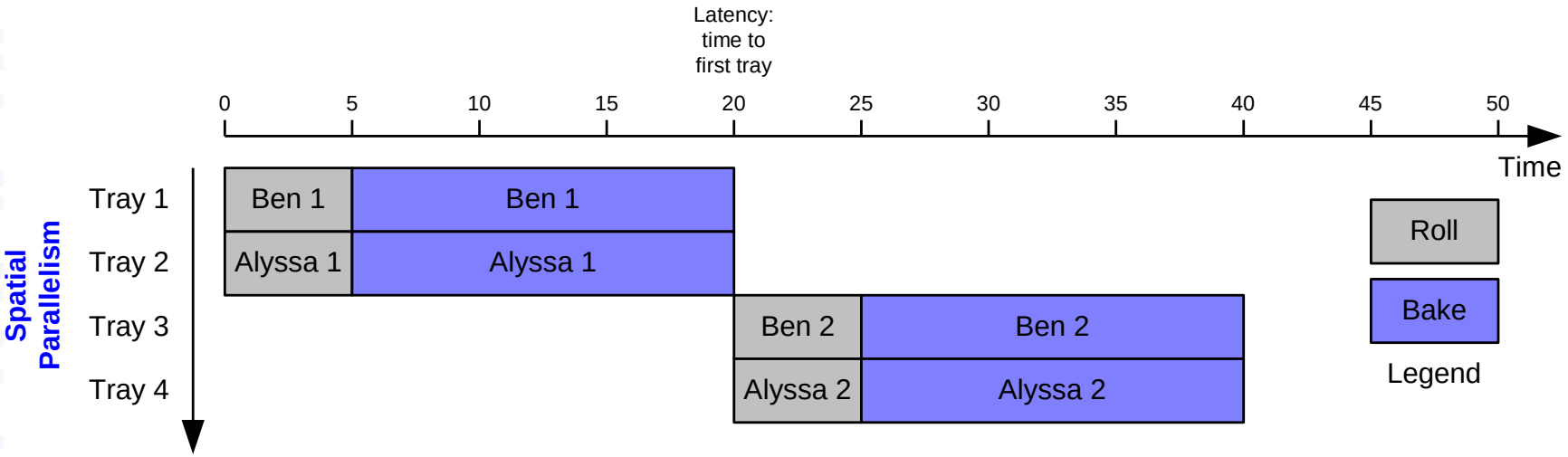
Latency = 5 + 15 = 20 minutes = **1/3 hour**

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:**
    - two stages: rolling and baking
    - He uses two trays
    - While first batch is baking, he rolls the second batch, etc.

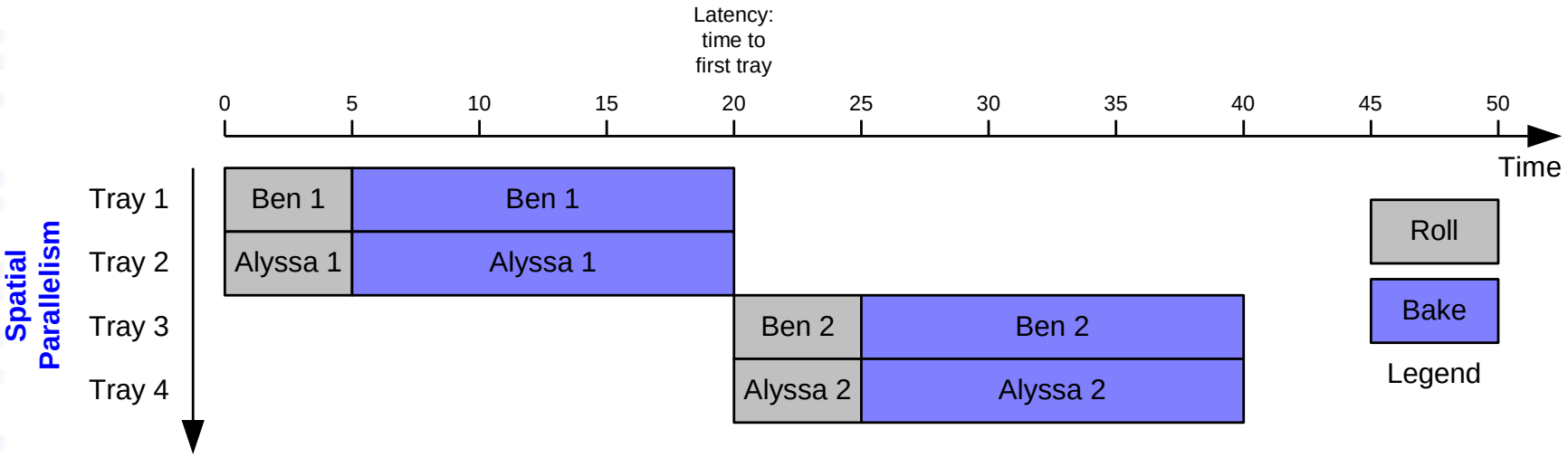
Spatial Parallelism



Latency = ?

Throughput = ?

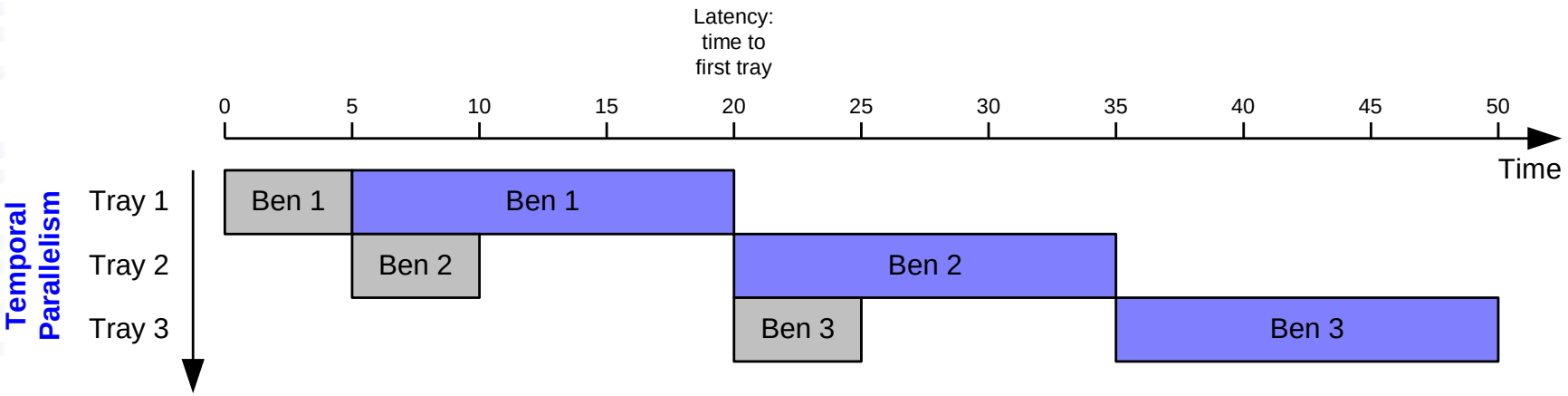
Spatial Parallelism



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

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

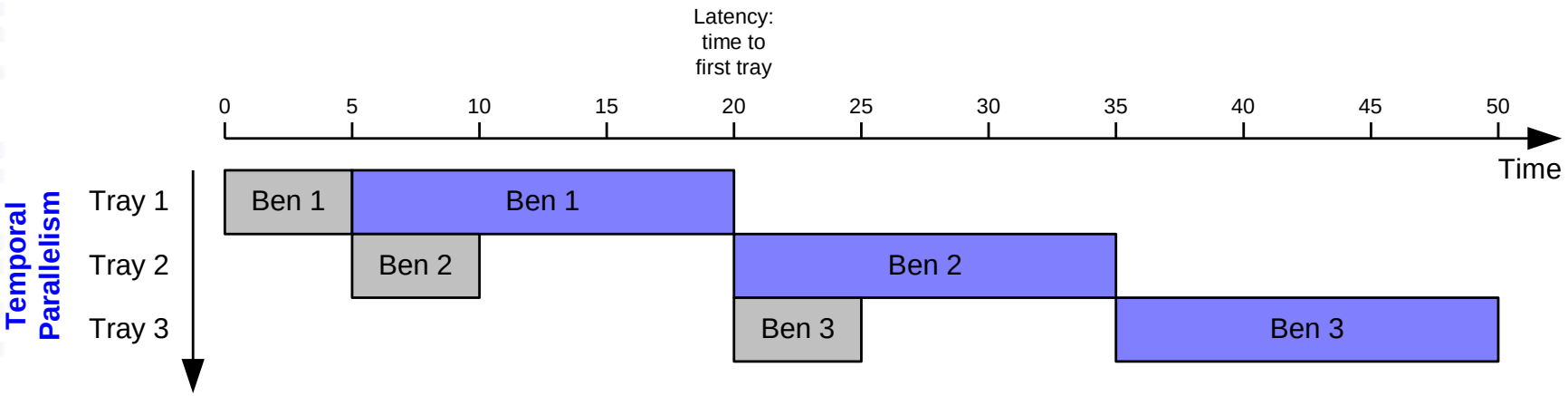
Temporal Parallelism



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**