Chapter 3 :: Sequential Logic Design (5)

***Digital Design and Computer Architecture*, 2nd Edition**

David Money Harris and Sarah L. Harris

참고도서 : 개정판 논리회로 설계(김종현저) , 홍릉과학출판사.

- 제7장 순차회로의 분석과 설계

Copyright © Elsevier

Chapter 3 :: Topics

- Introduction
- Latches and Flip-Flops
- Synchronous Logic Design
- Finite State Machines
- Shifter

- Timing of Sequential Logic
- Parallelism

Timing (순차논리의 타이밍)

- 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

Input Timing Constraints

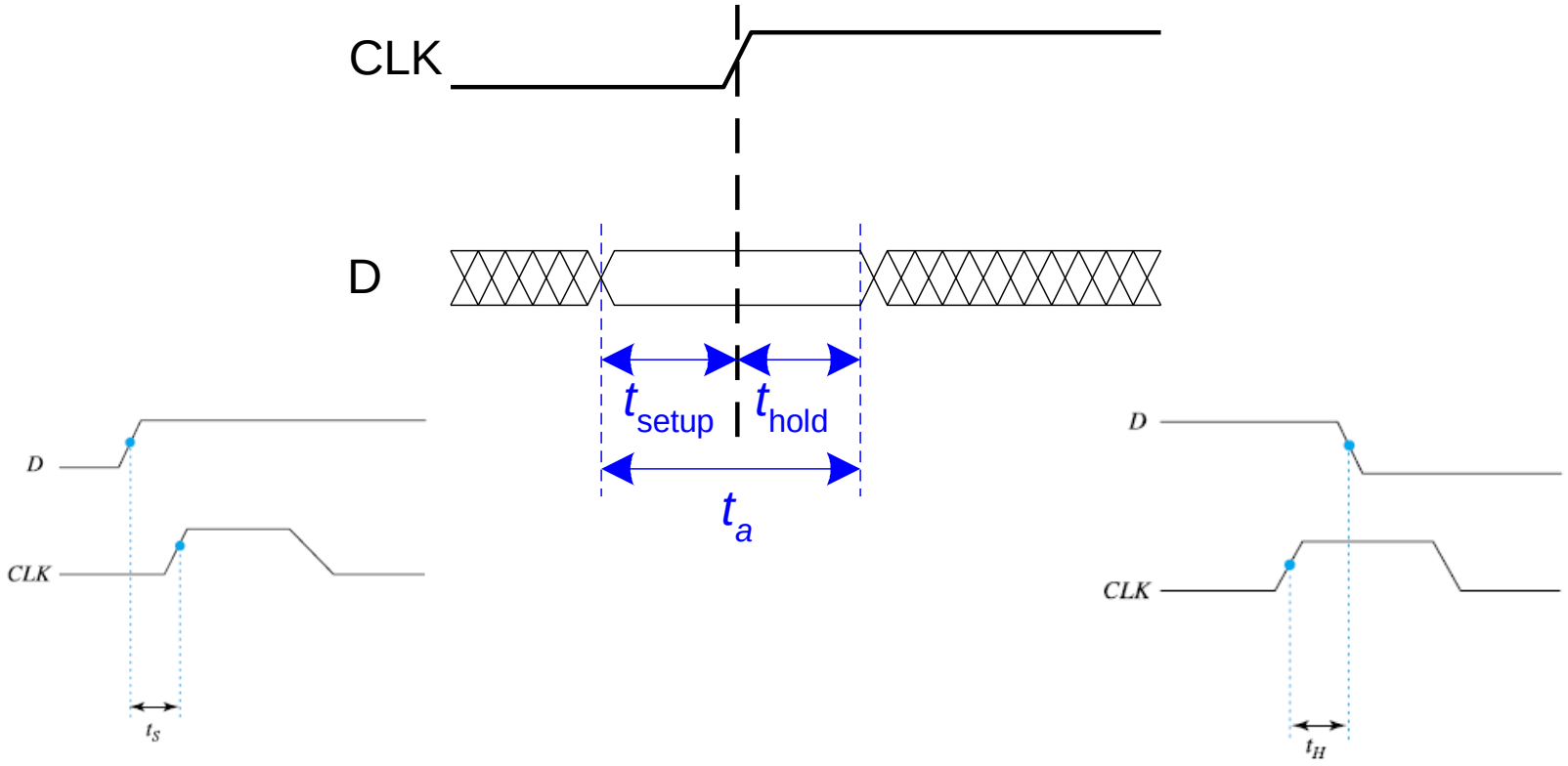
- 전파 지연 시간(**propagation delay time**)
  - 플립-플롭에서 입력 신호가 인가된 순간부터 출력 신호가 발생할 때까지의 시간 간격 (내부 회로 동작 시간에 해당)

- **Setup time**
  - 플립-플롭의 출력 값이 안정된 상태로 결정되도록 하기 위하여, 입력 신호가 미리 세트 되어 있어야 하는 최소 시간
  - t_{setup} = **time *before* the clock edge that data must be stable (i.e. not changing)**

- **Hold time**
  - 플립-플롭의 출력값이 안정된 상태로 결정되도록 하기 위하여, 입력 값이 그 상태를 유지하고 있어야 하는 최소 시간
  - t_{hold} = **time *after* the clock edge that data must be stable**

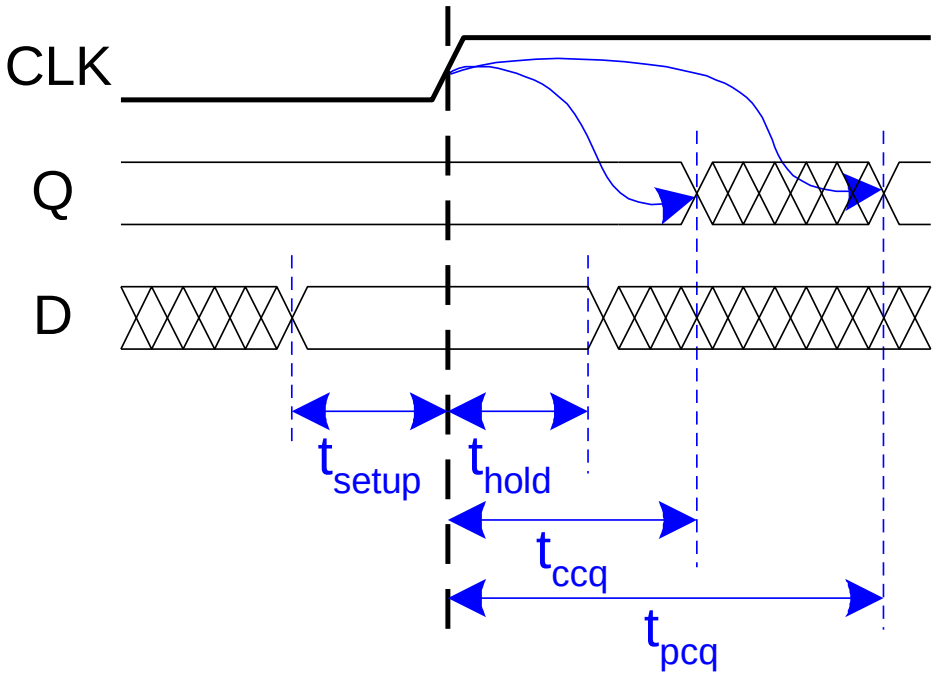
Input Timing Constraints

- Hold time: Aperture time(간극 시간):
- t_a = time around clock edge that data must be stable
- ($t_a = t_{setup} + t_{hold}$)



Output Timing

- 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)



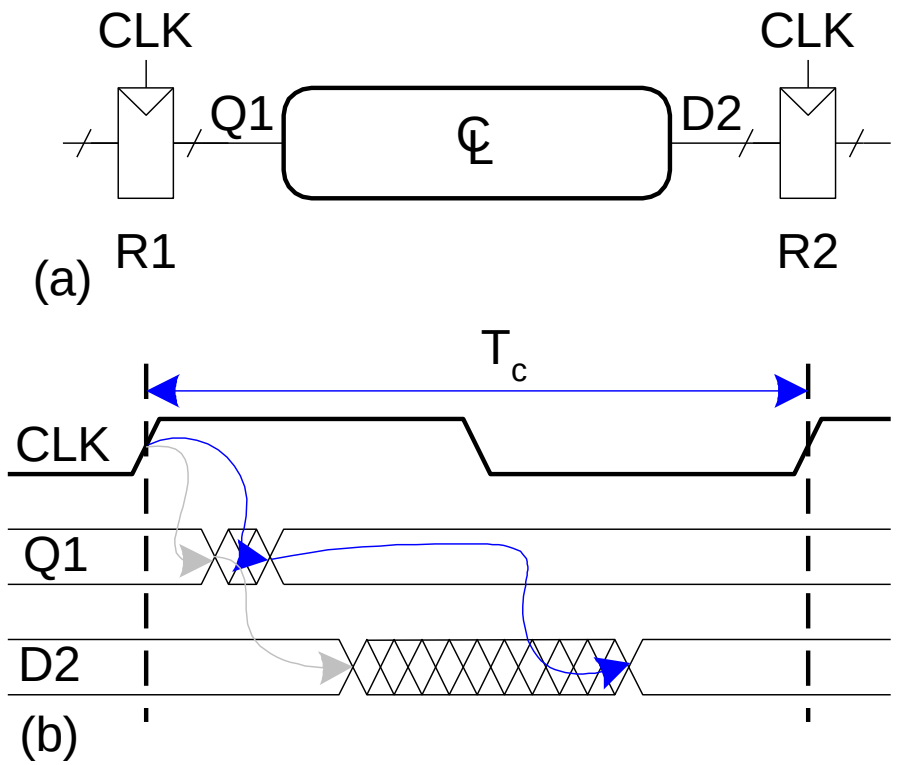
Dynamic Discipline

- The input to a synchronous sequential circuit must be stable during the aperture (setup and hold) time around the clock edge.
  (클럭에지의 Contamination t_{ccq} 동안 안정화되어야 한다)

- Specifically, the input must be stable
  - at least t_{setup} before the clock edge
  - at least until t_{hold} after the clock edge

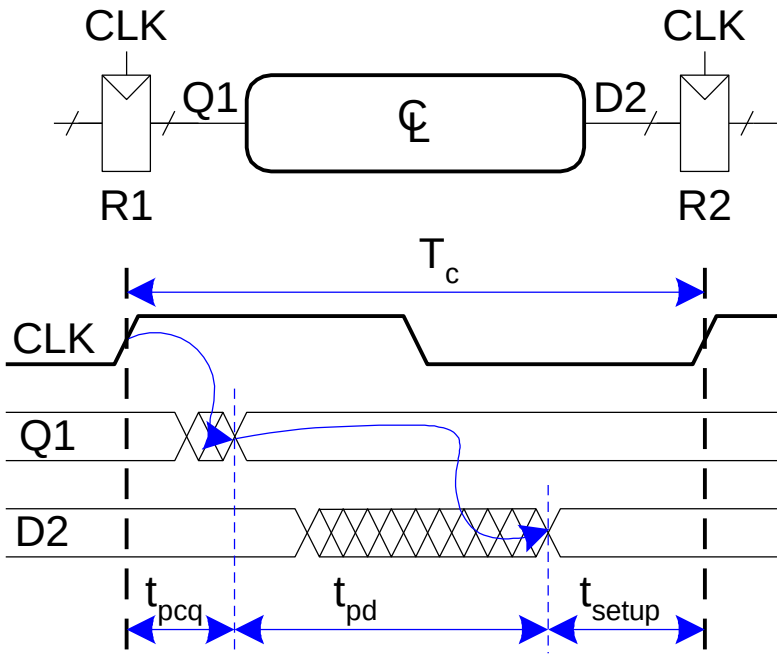
Dynamic Discipline

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



Setup Time Constraint (준비시간제한)

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

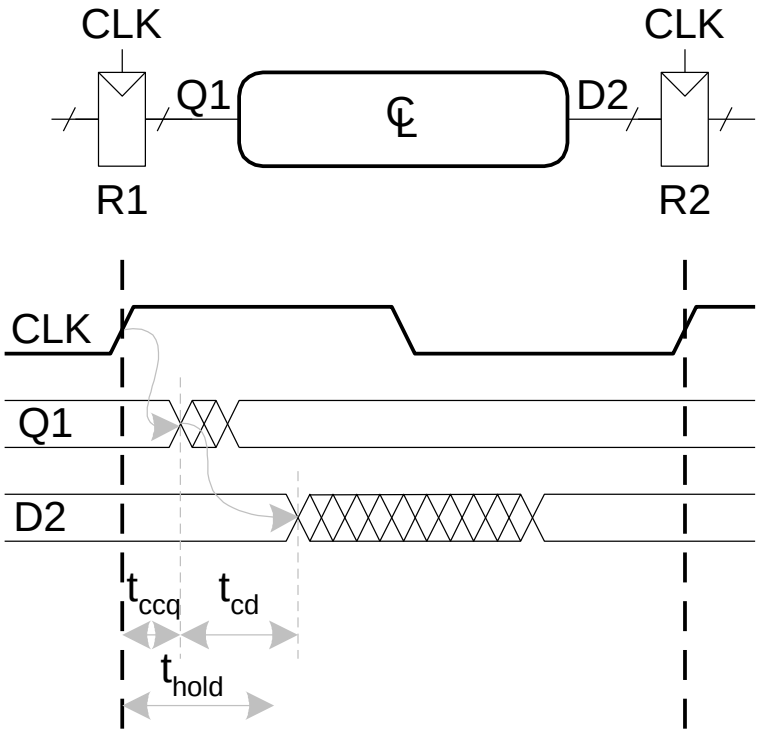


$$T_{c(\text{최소클럭주기})} \geq t_{pcq} + t_{pd} + t_{setup}$$

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

Hold Time Constraint (대기시간제한)

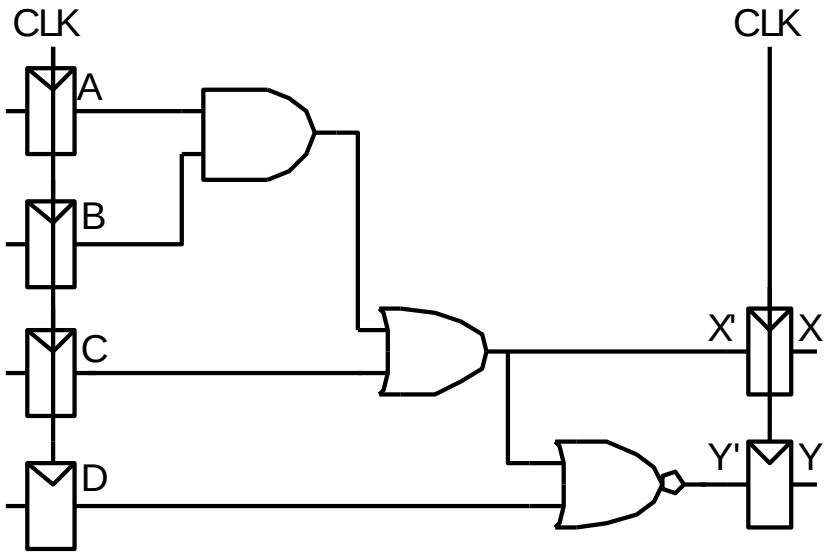
- The hold time constraint depends on the **minimum** delay from register R1 through the combinational logic.
- The input to register R2 must be stable for at least t_{hold} after the clock edge.



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

$$T_{cd(최소혼합지연)} > t_{hold} - t_{ccq}$$

Timing Analysis



Timing Characteristics

t_{ccq} = 30 ps

t_{pcq} = 50 ps

t_{setup} = 60 ps

t_{hold} = 70 ps

per gate

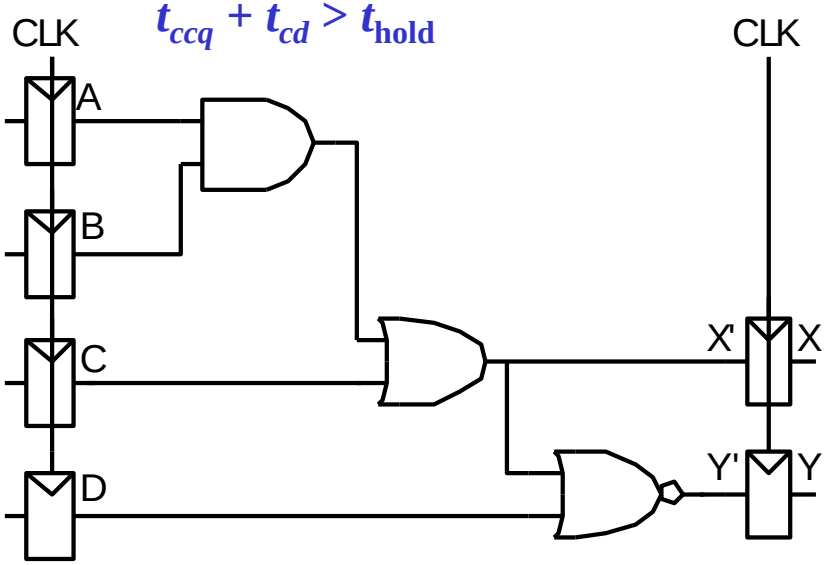
t_{pd} = 35 ps

t_{cd} = 25 ps

Timing Analysis

$T_{c(\text{최소클럭주기})} \geq t_{pcq} + t_{pd} + t_{setup}$

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



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

t_{cd} = 25 ps

Setup time constraint:

$T_c \geq (50 + 105 + 60)$ ps = 175 ps

$f_c = 1/T_c$ = 5.7 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

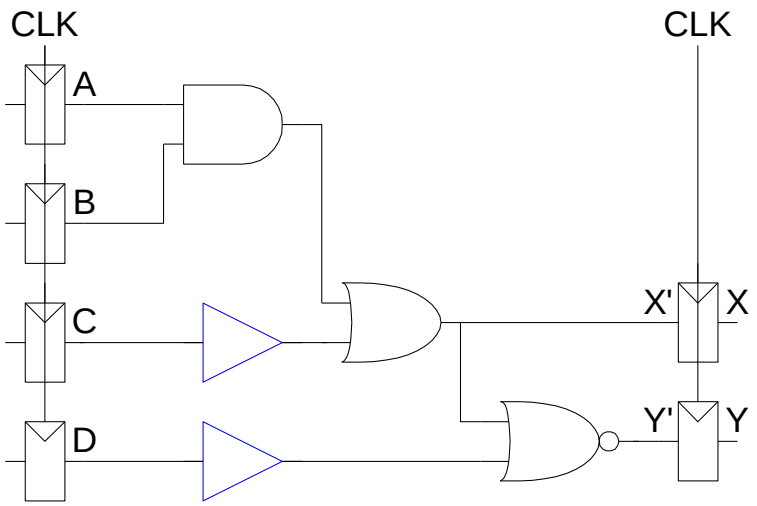
Hold time constraint:

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

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

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 = 175 ps

$f_c = 1/T_c$ = 5.7 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

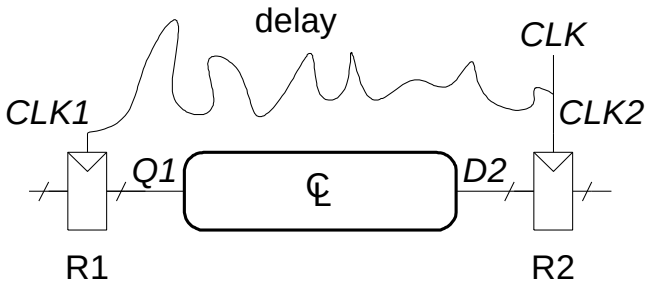
Hold time constraint:

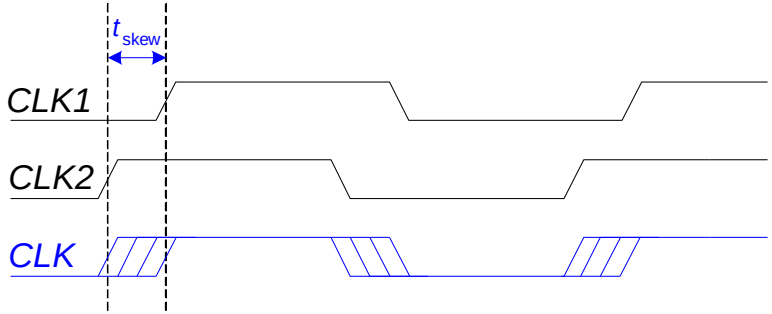
$t_{ccq} + t_{pd} > t_{hold}$?

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

3.5.3 Clock Skew(클록 시차)

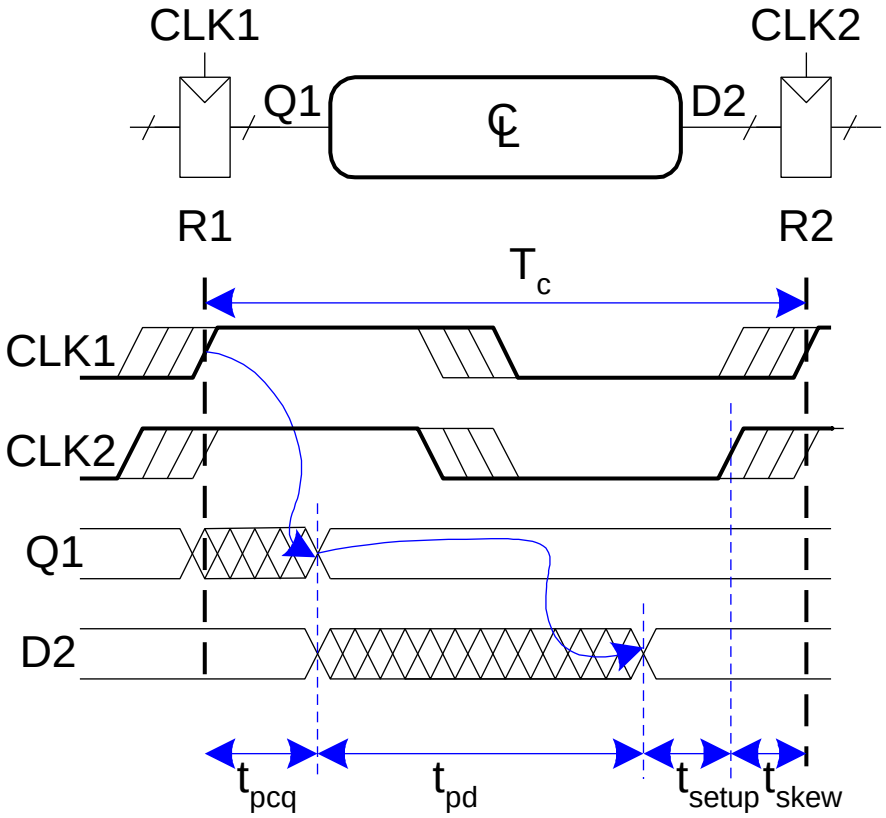
- The clock doesn't arrive at all registers at the same time
  (클록이 모든 레지스터에 같은 시간에 도달하지 않는다)
- This may be caused by delay or other timing noise (딜레이나 노이즈 때문에)
- Skew is the difference between two clock edges
- Because there may be many registers in a system, we examine the worst case for each case to guarantee that the dynamic discipline is not violated for any register





Setup Time Constraint with Clock Skew

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

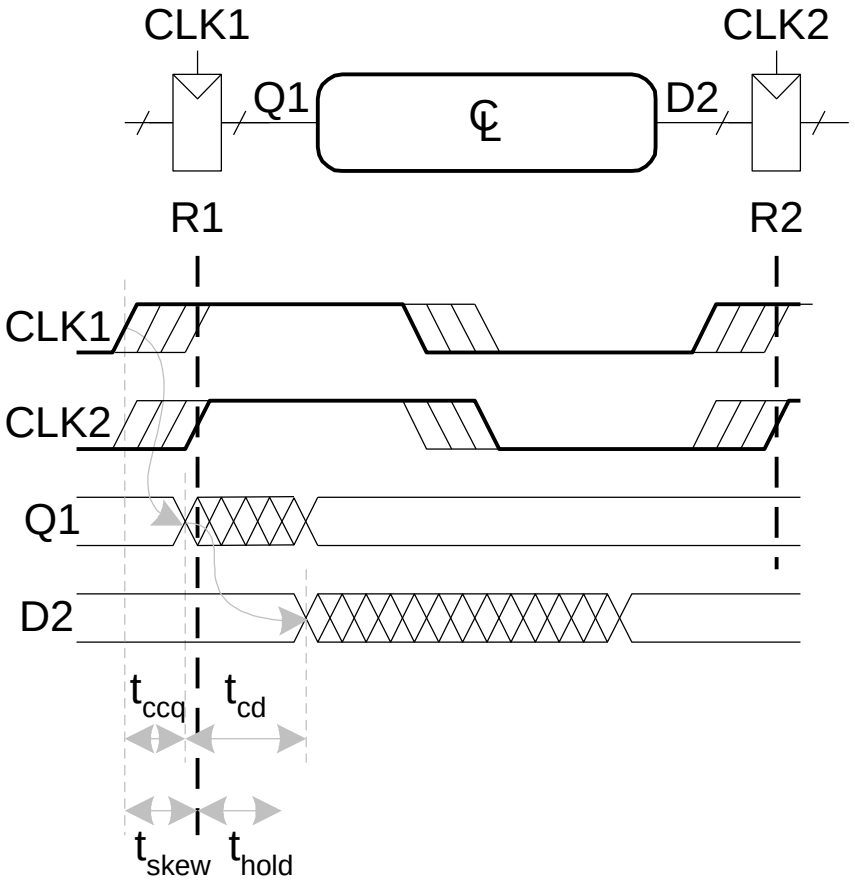


$$T_c - t_{\text{skew}} \geq t_{pcq} + t_{pd} + t_{\text{setup}}$$

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

Hold Time Constraint with Clock Skew

- In the worst case, CLK2 is later than CLK1

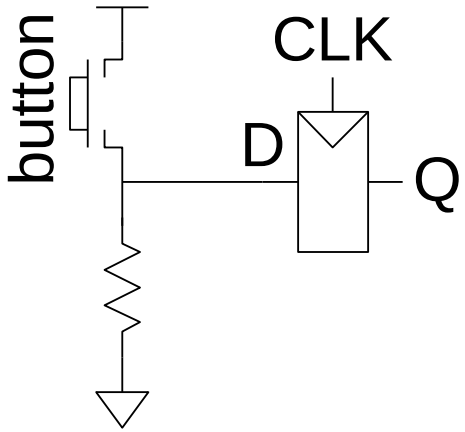


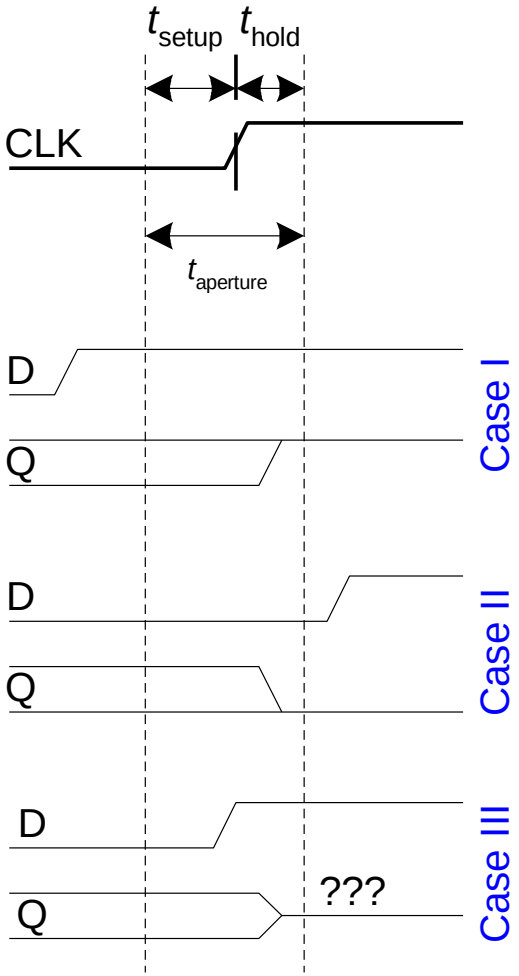
$$t_{ccq} + t_{cd} > t_{\text{hold}} + t_{\text{skew}}$$

$$t_{cd} > t_{\text{hold}} + t_{\text{skew}} - t_{ccq}$$

Violating the Dynamic Discipline

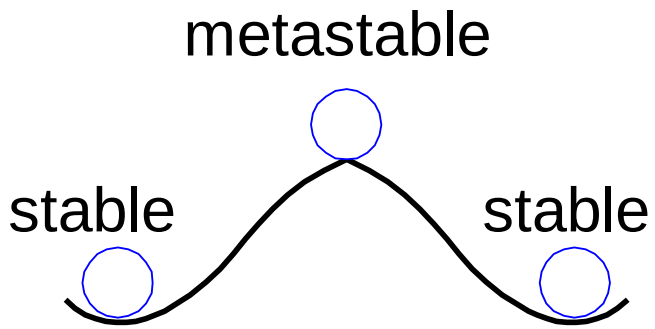
- Asynchronous (for example, user) inputs might violate the dynamic discipline





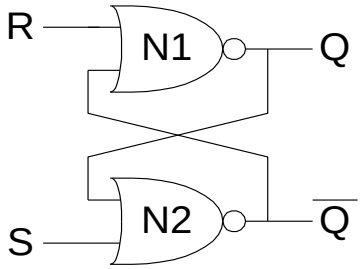
Metastability(준안정 상태)

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



Flip-flop Internals

- Because the flip-flop has feedback, if Q is somewhere between 1 and 0, the cross-coupled nand gates will eventually drive the output to either rail (1 or 0, depending on which one it is closer to).



- A signal is considered metastable if it hasn't resolved to 1 or 0
- The probability that the output Q is still metastable after waiting a time, t, is:

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

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

T_0, τ : properties of the circuit

Metastability

- Intuitively:
  - T_0/T_c 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

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

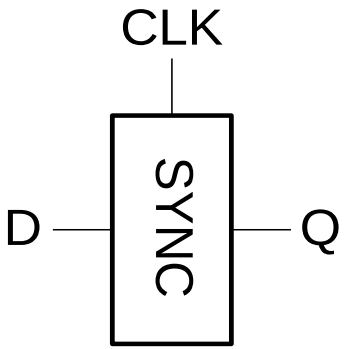
- In short, if a flip-flop samples a metastable input, if you wait long enough (t), the output will have resolved to 1 or 0 with high probability.

3.5.5 Synchronizers(동기화기)

- **Asynchronous inputs (*D*) are inevitable** (user interfaces, systems with different clocks interacting, etc.).
  (비동기 입력은 필연적이다)

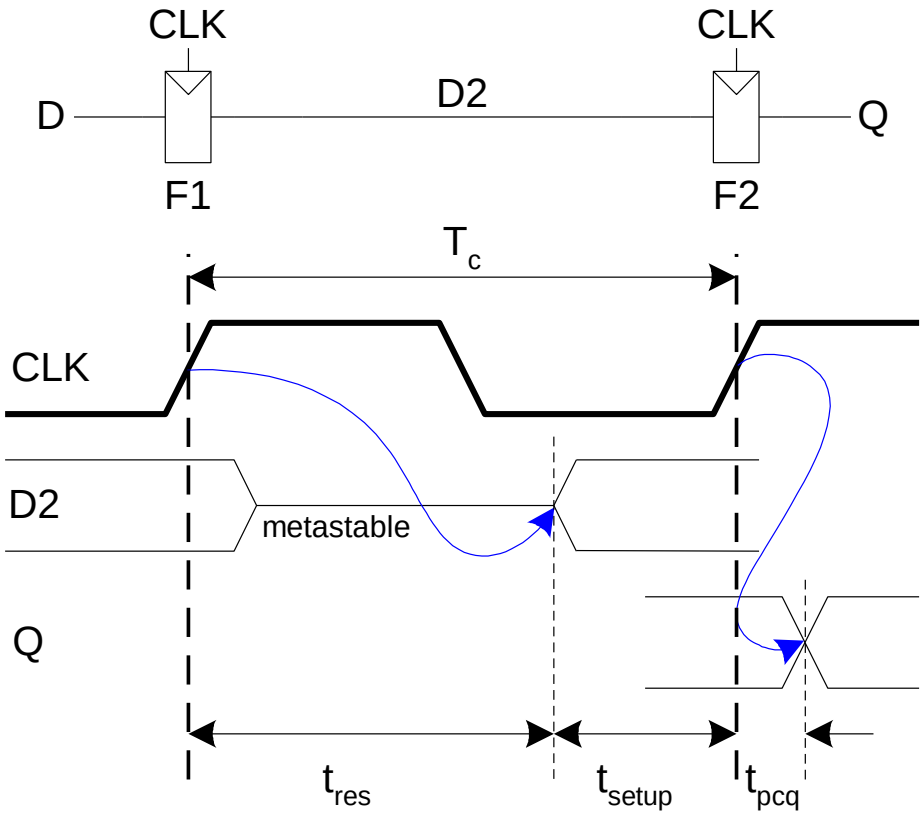
- The **goal of a synchronizer** is to make the probability of failure (the output *Q* still being metastable) low.

- A synchronizer cannot make the probability of failure 0.



3.5.5 Synchronizer Internals(동기화기 내부)

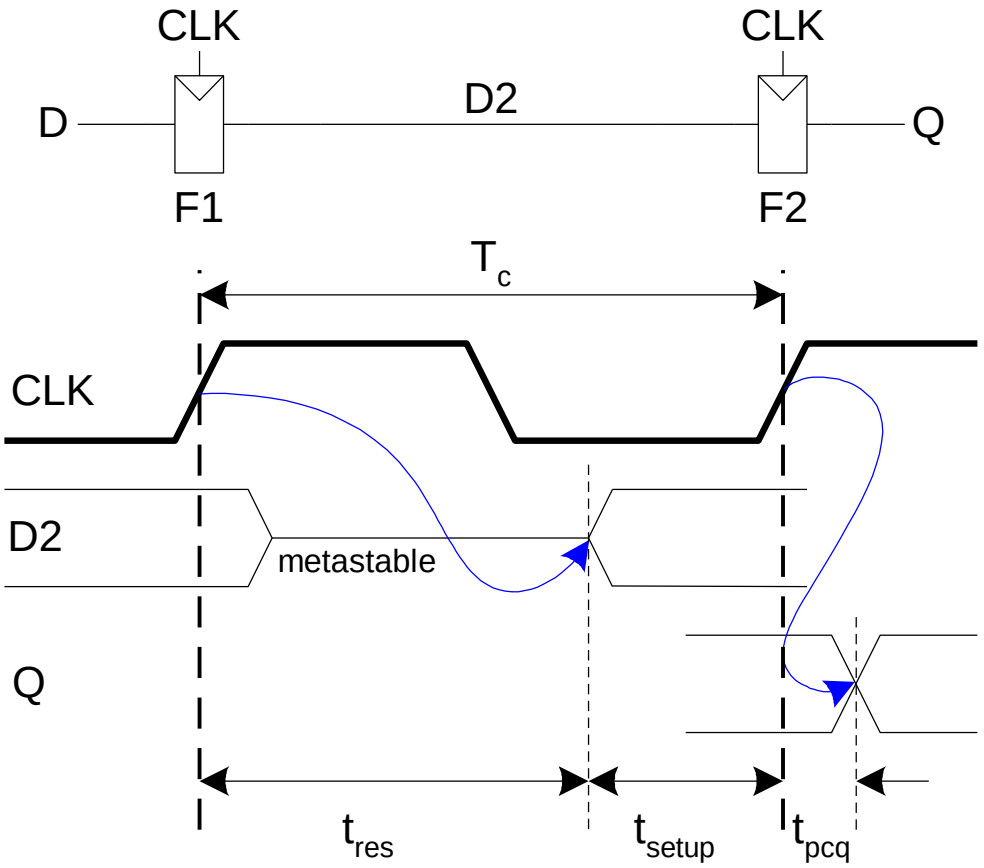
- A **synchronizer** can be built with two back-to-back flip-flops.
- Suppose the input D is transitioning when it is sampled by flip-flop 1, F1.
- The amount of time the internal signal D2 can resolve to a 1 or 0 is (T_c - t_{setup}).



Synchronizer Probability of Failure

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

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



Synchronizer Mean Time Before Failure

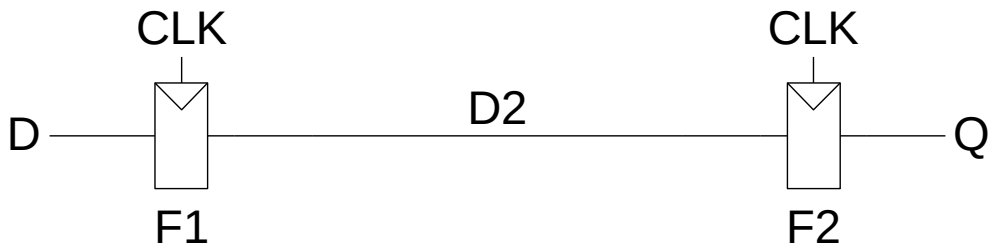
- If the asynchronous input changes once per second, the probability of failure per second of the synchronizer is simply *P*(failure):
- In general, if the input changes *N* times per second, the probability of failure per second of the synchronizer is:

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

- Thus, the synchronizer fails, on average, 1/[*P*(failure)/second]
- This is called the *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 MHzτ = 200 ps
  T_0 = 150 ps t_{setup} = 100 ps
  N = 1 time per second

- What is the probability of failure? MTBF?

$$P(\text{failure}) = (150 \text{ ps}/2 \text{ ns})\, e^{-(1.9 \text{ ns})/200 \text{ ps}}$$
$$= 5.6 \times 10^{-6}$$
$$P(\text{failure})/\text{second} = 1\,(5.6 \times 10^{-6})$$
$$= 5.6 \text{ x } 10^{-6} / \text{ second}$$
$$\text{MTBF} = 1/[\text{P(failure)/second}] \approx 50 \text{ hours}$$

3.6 Parallelism (병렬처리)

- Some definitions:
  - Token: A group of inputs processed to produce a group of outputs
    (출력그룹을 생성하기 위해 처리되는 입력그룹)
  - Latency(잠복시간): Time for one token to pass from start to end
  - Throughput(처리량): The number of tokens that can be produced per unit time
- The purpose of parallelism is to increase throughput.
- Two types of parallelism:
  - **Spatial parallelism** (공간 병렬 처리)
    - duplicate hardware performs multiple tasks at once
  - **Temporal parallelism** (시간 병렬 처리)
    - task is broken into multiple stages
    - also called pipelining
    - for example, an assembly line

Parallelism 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?

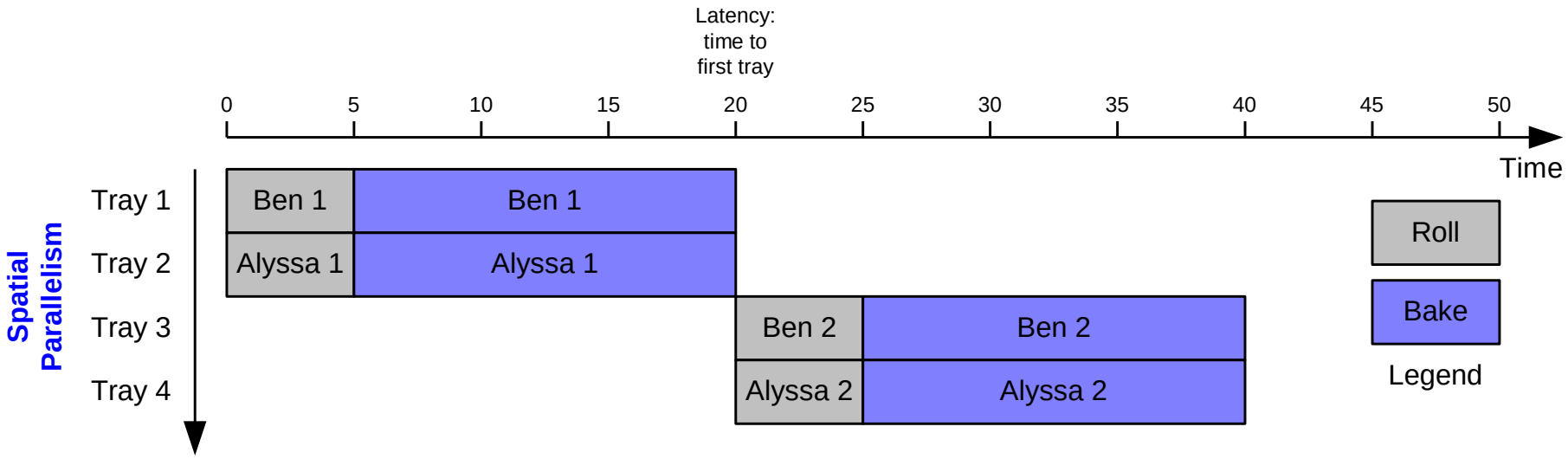
Latency = 5 + 15 = 20 minutes = 1/3 hour

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.

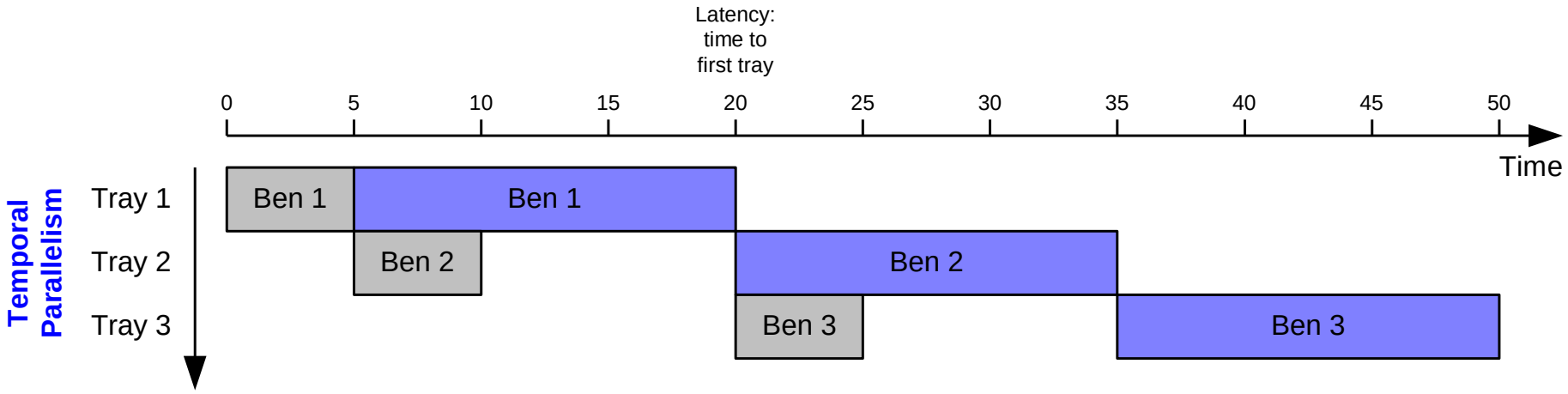
Spatial Parallelism



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

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

Temporal Parallelism

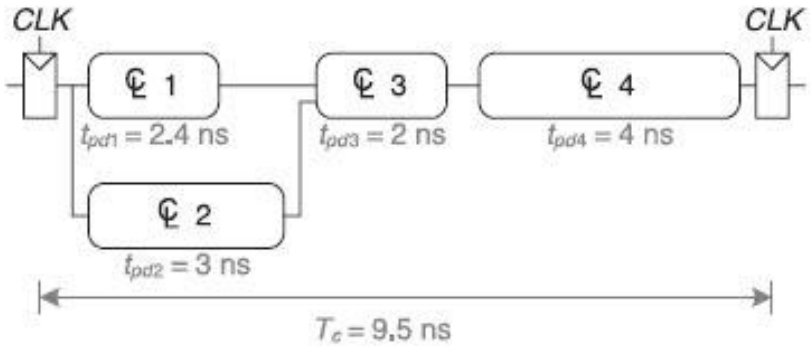


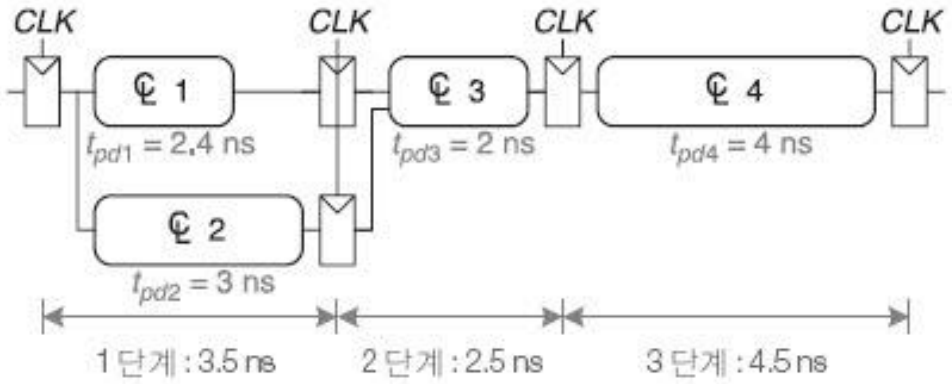
Latency = 5 + 15 = 20 minutes = 1/3 hour

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

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

파이프라인 프로세서


CLK
CLK
Ⓒ 1
Ⓒ 3
Ⓒ 4
Ⓒ 2
t_{pd1} = 2.4 ns
t_{pd3} = 2 ns
t_{pd4} = 4 ns
t_{pd2} = 3 ns
T_c = 9.5 ns


CLK
CLK
CLK
CLK
Ⓒ 1
Ⓒ 3
Ⓒ 4
Ⓒ 2
t_{pd1} = 2.4 ns
t_{pd3} = 2 ns
t_{pd4} = 4 ns
t_{pd2} = 3 ns
1 단계 : 3.5 ns
2 단계 : 2.5 ns
3 단계 : 4.5 ns

Q & A