

CHAP 4: Synchronous Design

4.1. Timing Issue of Logic Data.

Timing Issue of Logic Data (Vấn đề thời gian của dữ liệu logic)

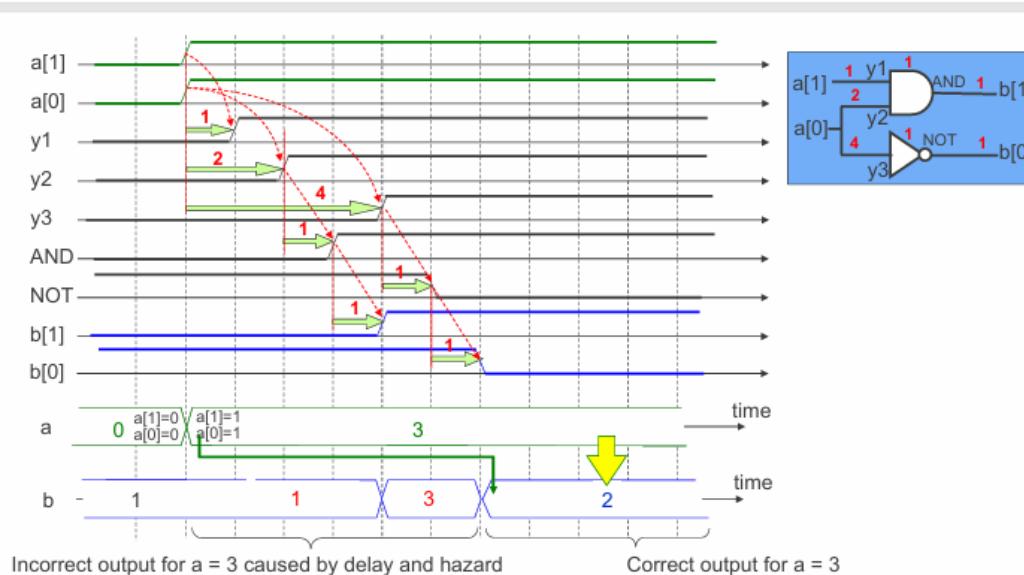
Phân tích xem điều gì xảy ra ở tín hiệu đầu ra b khi tín hiệu đầu vào a thay đổi, trong khi các kết nối dây, cổng AND, và cổng NOT đều có độ trễ (delay).

Kết quả đầu ra b như sau:

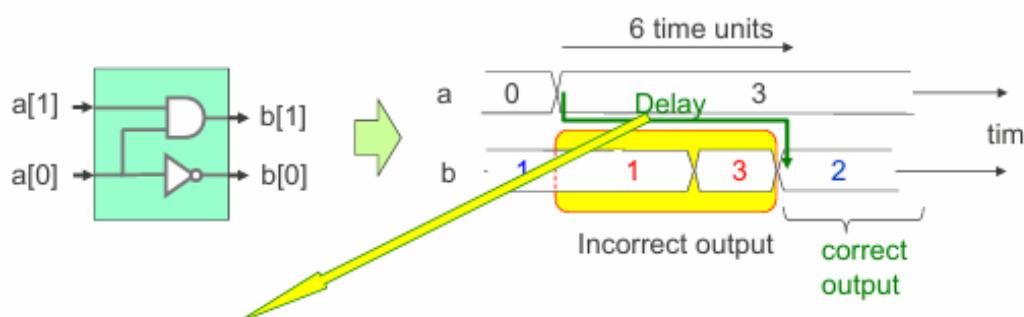
Thời gian	b[1]	b[0]
t = 0	0	1
t = 3	1	1
t = 5	1	0

Kết luận & Kiến thức trọng tâm:

- **Delay không đồng đều** trong hệ thống dẫn đến **hiện tượng glitch/tạm thời sai lệch** nếu không đồng bộ tốt.
- Trong thiết kế **synchronous**, cần cẩn thận với **clock skew, setup/hold time, và delay mismatches** như ví dụ này.
- Cần cân nhắc dùng các mạch **register, pipeline, hoặc clocked logic** để đảm bảo tính chính xác khi thiết kế logic đồng bộ.



The result shows that output **b** does not have a correct value corresponding to the input **a** until 6 time units, the largest delay in the logic block, passes.



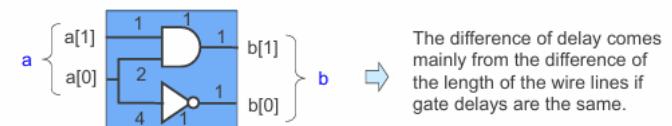
However, delay is not an only problem we will have for timing issues.

We will have much **serious problem**, if **input changes in a short time**.

Let's see what happens, if input changes as follows.

Timing Issue of Logic Data

Let's study what will come out to output signal **b** when input signal **a** changes while wire connections, AND, and NOT gate have delay.



The difference of delay comes mainly from the difference of the length of the wire lines if gate delays are the same.

Numbers in the block diagram are **delay** of each element, wire-delay and gate-delay.

Suppose **a** changes from **0** to **3**, then **b** shall change from **1** to what ??



Draw the time chart for the given conditions above.

Biểu đồ b: Kết quả sai tạm thời (Incorrect Output):

- Tại thời điểm $a = 3$, b **đáng lẽ phải là $b = 2$** (tức là $b[1]=1, b[0]=0$).
- Tuy nhiên, do delay không đồng đều và xuất hiện **hazard** (sự thay đổi chưa đồng bộ), b tạm thời nhảy sang 1 $\rightarrow 3 \rightarrow$ **sai ngắn hạn**.

Vấn đề xảy ra:

- **Tín hiệu chỉ giữ giá trị 3 trong 3 units time**, trong khi mạch cần **6 đơn vị** mới hoàn tất delay logic.
- Do đó:
 - $b[0] = \text{NOT}(a[0])$ chỉ thấy một phần thay đổi, có thể chia thành 0 rồi về lại 1.
 - $b[1] = a[1] \text{ AND } a[0]$ có thể không kịp cập nhật \rightarrow vẫn là 0.

Kết quả:

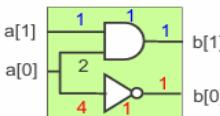
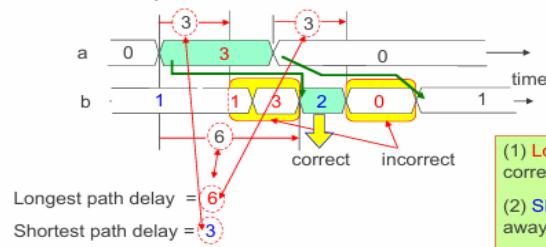
- **b** có thể hoàn toàn **không phản ánh đúng giá trị $a = 3$** trong đầu ra.
- Có thể có **glitch tạm thời**, hoặc không có thay đổi gì cả tại $b[1]$.

“Nếu tín hiệu đầu vào thay đổi quá nhanh, đầu ra sẽ không bao giờ cho kết quả chính xác.”



Input is 3 for just 3 time units

The result on the previous slide can be summarized as follows.



- (1) **Longest path delay:** The time needed for the correct output to appear at the output.
 (2) **Shortest path delay:** The correct output will go away after this time passes if the input changes.
 Therefore,
 (3) **Longest path delay - Shortest path delay:**
 (a) The input must be kept unchanged during this mount of the time period, otherwise the correct output will never appear at the output.
 (b) The time while incorrect output appears at the output because of the hazard.

Tóm tắt kiến thức về Delay và Hazard

1. Độ trễ đường dài nhất (Longest path delay)

Thời gian tối đa cần để đầu ra chính xác xuất hiện sau khi đầu vào thay đổi.

2. Độ trễ đường ngắn nhất (Shortest path delay)

Thời gian tối thiểu sau khi đầu vào thay đổi, đầu ra bắt đầu mất giá trị chính xác.

3. Hiệu số Delay (Hazard Window)

Khoảng thời gian dễ xảy ra lỗi nếu input không giữ ổn định

Ý nghĩa:

- (a) Input phải được giữ ổn định trong khoảng này để đảm bảo đầu ra chính xác xuất hiện.
- (b) Nếu không, hazard sẽ xảy ra → đầu ra có thể sai tạm thời hoặc không ổn định.

Lưu ý quan trọng:

Nếu input **thay đổi liên tục** trong khoảng T_{hazard} , thì output **không bao giờ chính xác** → hệ thống dễ gặp **glitch / lỗi logic**.

Tổng kết kiến thức về Timing Issue (đến hiện tại):

Vấn đề	Nguyên nhân	Hậu quả
Delay chênh lệch	Cổng logic và đường dây có trễ khác nhau	Đầu ra cập nhật sai thời điểm
Glitch	Tín hiệu thay đổi trong khi tín hiệu khác chưa đến	Đầu ra nhấp nháy tạm thời
Input không giữ đủ lâu	Input thay đổi trước khi logic hoàn thành xử lý	Đầu ra sai hoặc không đổi
Race/Hazard	Các đường tín hiệu "đua" nhau đến đích	Logic sai hoặc bất định

Giải pháp: Đồng bộ bằng Flip-Flop (FF)

Vấn đề:

Đầu ra chỉ đúng nếu đầu vào được giữ ổn định đủ lâu (ít nhất bằng thời gian delay dài nhất - delay ngắn nhất).

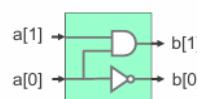
Nếu không, sẽ xảy ra **glitch** hoặc **output sai tạm thời** (hazard).

Giải pháp:

Chèn các FF ở đầu vào của logic block để:

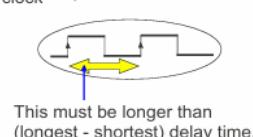
- Giữ giá trị ổn định → đầu vào không thay đổi liên tục trong thời gian nguy hiểm.
- Đồng bộ hóa tín hiệu với clock → dễ kiểm soát, predict hơn.

Đây là **kỹ thuật đồng bộ hóa** chuẩn trong thiết kế mạch số.

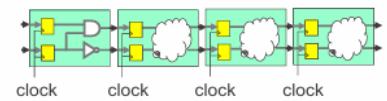


For the logic block on the left, it is mandatory to **keep the input unchanged** for (longest - shortest) delay time to get the correct output.

Placing FFs at the input of the logic block is the most simple answer for this issue.



Now, study about the cases to build a larger scale system by combining many blocks as shown below.



Áp dụng vào hệ thống lớn:

Khi xây dựng hệ thống nhiều block liên tiếp:

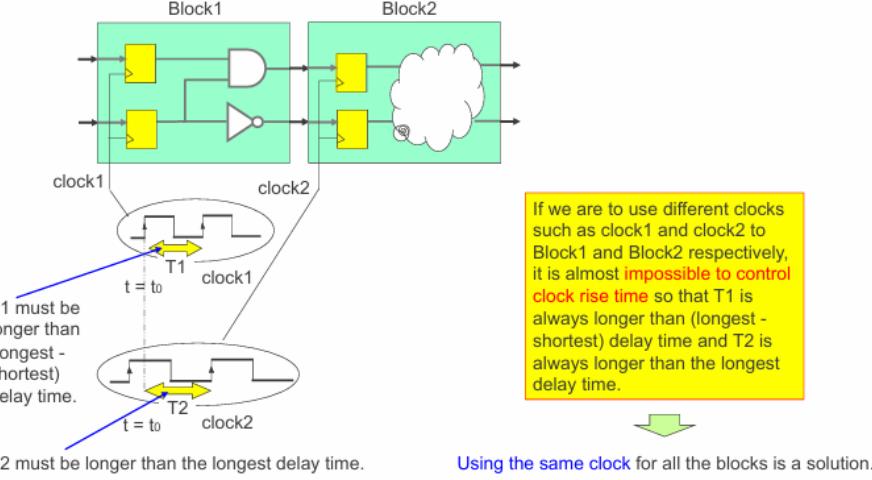
- Clock dùng chung để đồng bộ toàn hệ thống.
- Mỗi block có **FF đầu vào** để "chốt dữ liệu" trước khi xử lý.
- Khoảng thời gian giữa các clock (T_{clock}) **phải lớn hơn** delay dài nhất - delay ngắn nhất trong mỗi block.

Lưu ý: $T_{clock} > (T_{long} - T_{short})$

Nếu không, hazard sẽ lan truyền từ block này sang block khác → cực kỳ khó debug và gây lỗi logic lớn.

TRONG HỆ THỐNG LỚN:

- Khi bạn kết nối nhiều khối logic \rightarrow FF \rightarrow logic \rightarrow FF...
- Tốt nhất: **sử dụng cùng một xung clock cho toàn bộ hệ thống.**



Nếu dùng nhiều clock:

- Clock lệch nhau (clock skew).
- Không đồng bộ hóa tín hiệu.
- Khó đảm bảo đúng T1, T2 (trong slide cuối).
- Gây **race condition, metastability** (lỗi logic rất khó debug).

4.2. Synchronous Design and Static Timing Analysis (STA)

Khái niệm Synchronous Design

Synchronous design (thiết kế đồng bộ) là kỹ thuật sử dụng chung một tín hiệu xung nhịp (clock) cho toàn bộ các khối logic trong mạch. Điều này giúp đồng bộ hóa các quá trình xử lý, đảm bảo dữ liệu được ghi/đọc chính xác tại các thời điểm xác định.

Trong sơ đồ:

- Các khối P1, P2, P3 là các khối logic (có thể là module xử lý, bộ cộng, các khối logic phức tạp, v.v.).
- Tín hiệu được truyền qua các flip-flop (FF – các ô nhớ) nằm giữa các khối logic.
- Tất cả các flip-flop đều được điều khiển bởi cùng một tín hiệu clock.

Nguyên lý hoạt động

1. Clock Cycle (Chu kỳ xung nhịp):

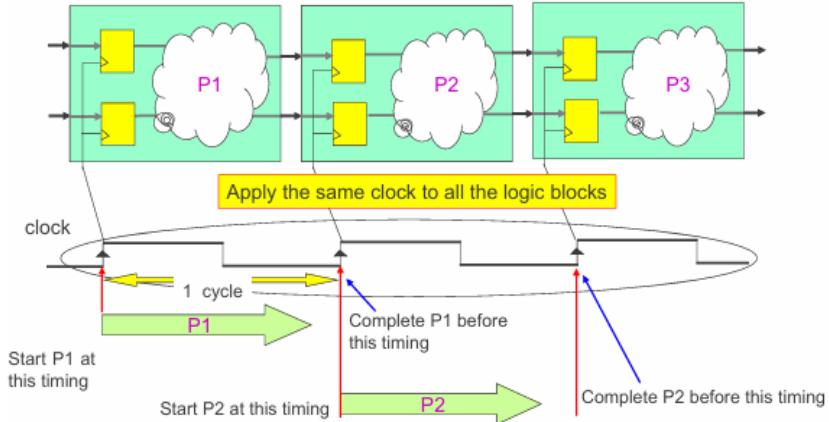
- Một chu kỳ clock là khoảng thời gian giữa hai cạnh lên liên tiếp của tín hiệu clock.
- Mỗi chu kỳ này là thời gian để dữ liệu được xử lý xong tại một khối logic.

2. Đồng bộ hóa xử lý:

- Ví dụ: tại cạnh lên của clock, P1 bắt đầu xử lý.
 - Trong 1 chu kỳ, P1 phải hoàn thành xử lý để dữ liệu được gửi đến flip-flop tiếp theo.
 - Ở cạnh lên tiếp theo, dữ liệu sẽ được chuyển sang P2 để xử lý tiếp.
- 3. Yêu cầu về thời gian (Timing Requirements):**
- Đầu ra từ một khối logic phải sẵn sàng trước khi clock kích hoạt flip-flop tiếp theo.
 - Nếu không, sẽ có lỗi timing (vi phạm setup hoặc hold).

Synchronous Design and STA

Synchronous design uses one clock for all the logic blocks and capture the signals at clock rise time to make them unchanged during one cycle for the processing.



Ví dụ trong hình:

- P1 bắt đầu tại cạnh lên đầu tiên \rightarrow phải hoàn thành trong 1 chu kỳ clock.
- Nếu không, khi P2 bắt đầu (ở cạnh lên kế tiếp), nó sẽ không có dữ liệu đầu vào đúng.
- Vì vậy, **delay(P1)** phải “nhỏ hơn” **clock period** – một điều kiện bắt buộc mà STA kiểm tra.

Static Timing Analysis (STA) là gì?

STA là phương pháp phân tích thời gian mà không cần mô phỏng (simulation), bằng cách kiểm tra các đường truyền tín hiệu trong thiết kế để đảm bảo tín hiệu đến đúng thời điểm.

STA thực hiện:

- Tính **delay** của các đường truyền giữa các flip-flop.
- So sánh delay với **chu kỳ clock** để đảm bảo tín hiệu đến kịp.
- Phát hiện vi phạm:
 - Setup violation:** tín hiệu đến quá trễ, chưa kịp ghi vào flip-flop.
 - Hold violation:** tín hiệu thay đổi quá sớm, gây mất ổn định.

Tại sao STA quan trọng?

- Tối ưu tần số hoạt động (clock frequency).
- Phát hiện sớm lỗi thời gian trước khi mạch được chế tạo (fabricated).
- Giảm thời gian phát triển và tăng độ tin cậy của chip.

Các cách sử dụng clock trong thiết kế đồng bộ

1. Single Phase Clock – One Cycle Transfer (Khuyến nghị sử dụng)

Đặc điểm:

Chỉ sử dụng **cạnh lên (rising edge)** của một tín hiệu clock duy nhất.

Dữ liệu được chuyển từ **FF1 đến FF2** trong **1 chu kỳ clock**.

Ưu điểm:

Đơn giản, dễ phân tích timing, không có xung đột pha (clock skew).

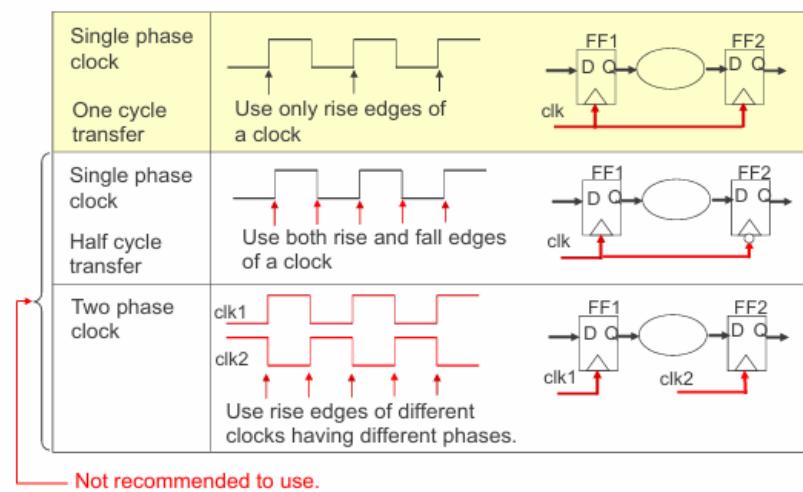
Dễ dàng áp dụng STA để kiểm tra delay của đường truyền giữa các flip-flop.

Timing dễ phân tích:

Setup time: Dữ liệu từ FF1 đến FF2 phải đến trước cạnh lên tiếp theo của clock.

Hold time: Dữ liệu phải giữ ổn định một thời gian ngắn sau cạnh clock.

There are several ways to use clocks in synchronous design as shown below. However, using one clock at its rise time is recommended.



2. Single Phase Clock – Half Cycle Transfer

Đặc điểm:

Sử dụng cả **cạnh lên và cạnh xuống** của **cùng một clock**.

Cho phép dữ liệu được truyền trong **nửa chu kỳ**.

Nguy cơ:

Khó kiểm soát timing hơn do:

Delay logic cần **nhỏ hơn một nửa chu kỳ clock** → Khó thiết kế nếu tần số cao.

Dễ xảy ra lỗi timing khi có **clock jitter** hoặc **clock skew**.

Khi nào dùng: Có thể dùng để **tăng throughput**, nhưng phải **thiết kế cẩn thận** và dùng STA kỹ lưỡng.

3. Two Phase Clock (Không khuyến nghị sử dụng)

Đặc điểm:

- Dùng **2 tín hiệu clock khác nhau (clk1, clk2)** có pha lệch nhau.
- FF1 và FF2 được điều khiển bởi **các cạnh lên của clk1 và clk2** tương ứng.

Lý do không khuyến nghị:

- Rất **khó kiểm soát độ lệch pha (phase skew)** giữa clk1 và clk2.
- STA **khó phân tích chính xác**, dễ xảy ra lỗi không dự đoán được.
- Khó tích hợp với công cụ EDA hiện đại.

Timing Constraints trong thiết kế đồng bộ

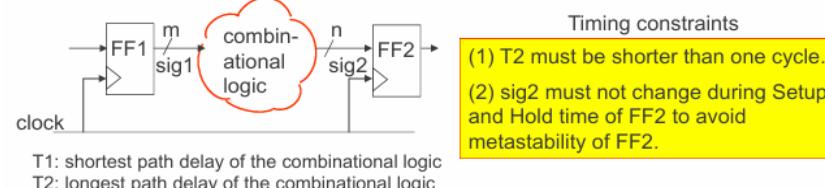
Mô hình hệ thống

- FF1: Flip-Flop đầu vào, chứa tín hiệu sig1
- FF2: Flip-Flop đầu ra, chứa tín hiệu sig2
- Giữa FF1 và FF2 là logic tổ hợp (combinational logic).
- Clock điều khiển cả FF1 và FF2.

Đường thời gian:

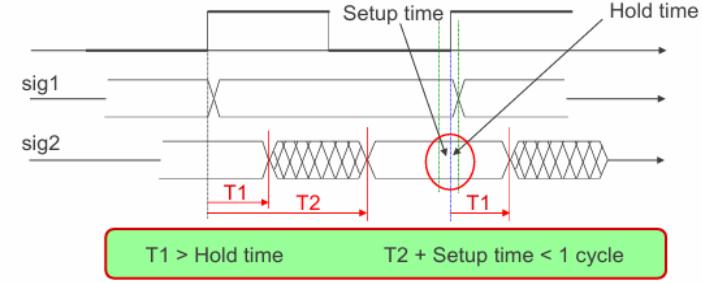
- sig1: Tín hiệu đầu ra từ FF1 (dữ liệu ổn định tại cạnh clock)
- sig2: Tín hiệu đầu vào FF2 sau khi đi qua logic tổ hợp
- Các điểm quan trọng:
 - T1: Thời gian delay ngắn nhất → phải > Hold time
 - T2: Thời gian delay dài nhất → phải + Setup time < 1 cycle

Timing constraints in synchronous design can be summarized as below.



T1: shortest path delay of the combinational logic

T2: longest path delay of the combinational logic



Nếu không thỏa mãn:

Setup Violation FF2 có thể không nhận được dữ liệu đúng, hoặc rơi vào trạng thái không xác định (metastability)

Hold Violation Dữ liệu thay đổi khi FF2 đang lấy mẫu, gây sai dữ liệu

Yêu cầu quan trọng: Dữ liệu phải hoàn tất xử lý trong một chu kỳ clock.

Các khái niệm chính:

T1: Shortest Path Delay

- Là độ trễ nhỏ nhất từ FF1 đến FF2.
- Nếu quá ngắn, tín hiệu đến quá sớm → dễ vi phạm hold time.

T2: Longest Path Delay

- Là độ trễ lớn nhất từ FF1 đến FF2.
- Nếu quá dài, tín hiệu đến sát cạnh clock tiếp theo → dễ vi phạm setup time.

Ràng buộc thời gian cần thỏa mãn:

(1) T2 + Setup Time < 1 cycle

- Giúp dữ liệu đến trước cạnh clock kế tiếp.
- Đảm bảo FF2 nhận dữ liệu kịp lúc trước khi lấy mẫu.
- Nếu không → vi phạm setup → gây metastability (tình trạng không xác định).

(2) T1 > Hold Time

- Dữ liệu phải giữ ổn định một khoảng thời gian ngắn sau clock edge.
- Nếu dữ liệu tới quá sớm, sẽ thay đổi khi FF2 đang lấy mẫu.
- Nếu không thỏa → vi phạm hold → FF2 lấy dữ liệu sai.

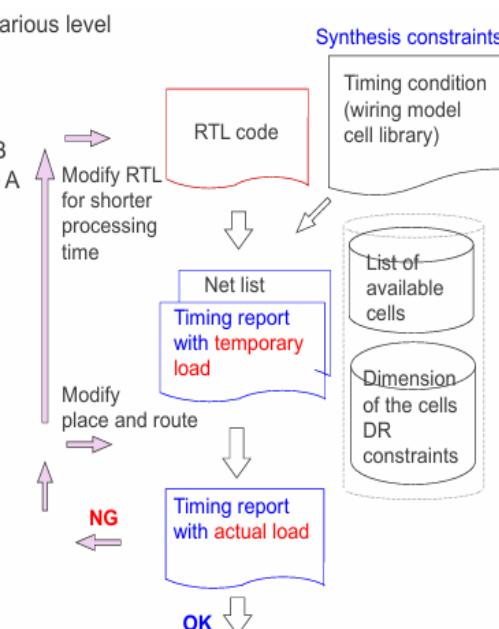
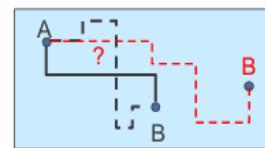
Liên hệ STA:

Trong Static Timing Analysis, công cụ sẽ:

- Phân tích tất cả các con đường truyền dữ liệu (timing paths) từ FF đến FF.
- Kiểm tra delay so với các ràng buộc setup và hold.
- Báo lỗi nếu có vi phạm timing.

STA is carried out several times by using various level of accuracy of delay data.

After layout (place and route), we must check the timing because delay from A to B is very much different depending on where A and B are placed and connected by which route.



Final check is done based on the actual load, which takes account electronic characteristics of wire lines.

Thực hiện STA qua các giai đoạn

STA được chạy nhiều lần trong quá trình thiết kế:

1. Trước layout:

- Dùng delay ước lượng (temporary load).
- Kiểm tra sơ bộ logic và timing.

2. Sau layout (Place & Route):

- Dùng delay thực tế (actual load) từ chiều dài dây, trở kháng,...
- Phân tích chi tiết → Nếu không đạt timing, cần: Sửa layout (thay đường đi) hoặc tối ưu RTL.

Lưu ý: Dây dài và uốn khúc (route khác nhau) có thể gây **chênh lệch lớn về delay**.

Phân tích sơ đồ STA với nhiều mức độ chính xác của dữ liệu trễ

Mục tiêu sơ đồ: Cho thấy quá trình STA **không chỉ thực hiện một lần duy nhất**, mà lặp lại nhiều lần với **mức độ chính xác tăng dần** khi thiết kế hoàn thiện dần.

1. Bắt đầu từ RTL code: RTL (Register Transfer Level): là thiết kế ban đầu của bạn (Viết bằng Verilog/VHDL).

Sử dụng các **ràng buộc tổng hợp (Synthesis Constraints)**:

- Thư viện cell:** danh sách cell có thể dùng.
- Kích thước cell.**
- DRC constraints** (giới hạn về khoảng cách, kích thước dây, điện áp, v.v.).

Sử dụng những ràng buộc này để tạo **Netlist**.

2. Timing Report với Temporary Load: Sau khi có netlist,

tín hành STA **tạm thời** với:

- Giá trị ước lượng về tải (temporary load)** – chưa có layout thật.
- Routing vẫn chưa thực hiện**, nên delay chỉ là tương đối.

Nếu **vi phạm timing**, có thể:

- Quay lại sửa RTL để làm mạch nhanh hơn.
- Hoặc đổi lại cấu trúc logic (ví dụ: thêm pipeline).

3. Place & Route Khi layout hoàn thành:

- Cell **được đặt** ở vị trí cụ thể.
- Routing** (nối dây) **được thực hiện**.

Giờ đây, delay **thực sự phụ thuộc** vào:

- Chiều dài dây.
- Tải điện dung.
- Vị trí các khối logic.

4. Timing Report với Actual Load

Chạy STA lại với:

- Dữ liệu trễ thực tế (actual delay)**.
- Dựa trên layout thật.

Nếu đạt (OK): thiết kế đã sẵn sàng.

Nếu không đạt (NG): cần quay lại chỉnh layout hoặc RTL.

Điểm mấu chốt

- Delay từ **A đến B** có thể rất khác nhau nếu:
 - Đặt ở vị trí khác.
 - Nối bằng dây ngắn hoặc dài hơn.
- Do đó, **phải chạy STA sau layout** để đảm bảo mạch hoạt động đúng.

Dividing a Process and Multi-Cycle Path

Một vấn đề phổ biến trong thiết kế đồng bộ: "Nếu một phép xử lý (process) không thể hoàn tất trong một chu kỳ clock, thì phải làm gì?". Đây là tình huống rất thường gặp trong thiết kế thực tế, và tài liệu đề xuất 2 giải pháp (countermeasures) như sau:

Giải pháp 1: Chia nhỏ xử lý – Pipelining

Ý tưởng: Chia phép xử lý phức tạp (W) thành **nhiều**

phần nhỏ hơn, mỗi phần chỉ cần **1 chu kỳ clock** để hoàn thành.

Cách làm:

- Giả sử bạn có xử lý W ($X, Y, Z \rightarrow Q$) quá phức tạp.
- Chia W thành 3 phần: W1, W2, W3
- Mỗi phần chỉ tính toán một phần nhỏ và hoàn thành trong 1 chu kỳ.

Lợi ích:

- Dễ thiết kế đồng bộ hơn.
- Dễ kiểm tra timing bằng **Static Timing Analysis (STA)**.
- Tạo điều kiện cho **pipeline processing** → tăng throughput.

Ghi chú từ sơ đồ:

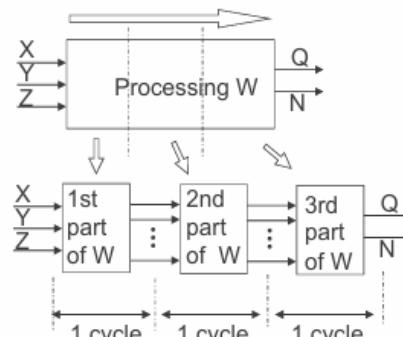
"You may use multi-cycle path specification of DA tool instead of dividing the process."

Có nghĩa là thay vì chia nhỏ logic thủ công, bạn có thể **Thông báo cho tool STA** rằng path này được phép dùng nhiều chu kỳ.

Next, a question may be raised that "What shall we do if an assumption that every process can be done in one clock cycle does not hold?"

⇒ A countermeasure (1)

Divide the process into several blocks so that each block can be completed in one clock cycle.



You may use multi cycle path specification of DA tool instead of dividing the process.

Giải pháp 2: Sử dụng tín hiệu "Valid" – Multi-Cycle with Handshake

Ý tưởng: Không chia nhỏ quá trình xử lý, nhưng thêm một tín hiệu "**valid**" hoặc "**ready**" để báo khi nào dữ liệu đã sẵn sàng sau nhiều chu kỳ.

Cách làm:

- Block xử lý W có thể mất hơn 1 chu kỳ để tính xong.
- Khi tính xong, nó bật tín hiệu **Valid**.
- Block tiếp theo (U) chỉ xử lý khi Valid = 1.

Lợi ích:

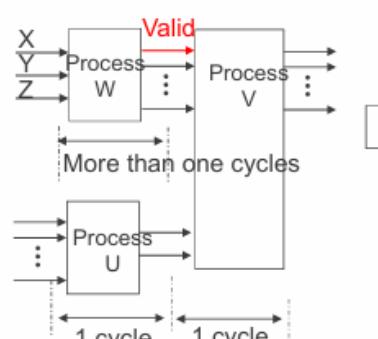
- Không cần chia nhỏ logic W.
- Giữ nguyên thiết kế phức tạp nhưng vẫn đảm bảo đúng dữ liệu.

Nhược điểm:

- Đây là **partial asynchronous logic** (tức là không hoàn toàn đồng bộ).
- Làm phức tạp kiểm tra timing hơn → STA sẽ cần xử lý path này khác đi.

⇒ Another countermeasure (2)

For data where processing cannot be ended within 1 clock cycle, introduce some signal such as valid or ready to notify the block waiting for the data that the data is ready.



This means introducing asynchronous logic partially.

So sánh nhanh:

Tiêu chí

Chia nhỏ xử lý (1)

Tín hiệu Valid (2)

Cách hoạt động Chia logic thành nhiều bước Thêm handshake bằng Valid

Tính chất 100% đồng bộ Có logic bất đồng bộ

Hỗ trợ bởi STA Dễ kiểm tra Cần báo multi-cycle / false path

Dễ pipeline Rất tốt Không tốt bằng

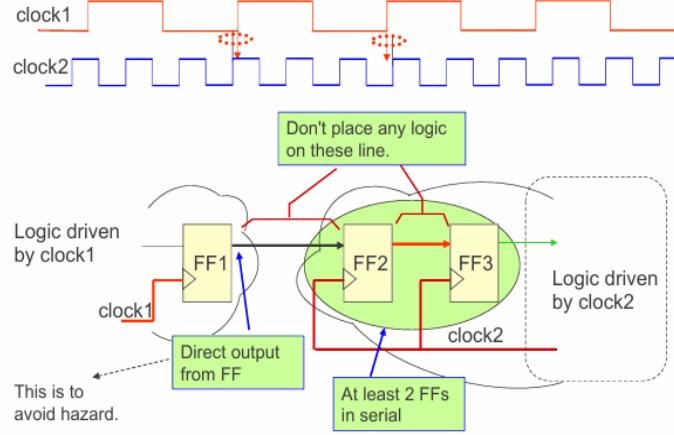
Truyền dữ liệu không đồng bộ (asynchronous data transfer)

Hệ thống đồng bộ rất tiện để kiểm soát timing, nhưng không thể áp dụng cho mọi trường hợp. Hai trường hợp phổ biến cần truyền dữ liệu không đồng bộ:

- Kết nối hai hệ thống chạy trên hai xung clock khác nhau.
- Nhận dữ liệu từ nguồn bên ngoài chip (như tín hiệu interrupt).

Lưu ý: Khi hai hệ thống dùng clock khác nhau, thì việc truyền dữ liệu phải đảm bảo tránh **metastability**.

Placing FFs in between clock1 system and clock 2 system as shown below is one of the known solution for the problem.



Giải pháp với 2 FF để tránh metastability

- Đặt ít nhất 2 flip-flop (FF) nối tiếp nhau khi truyền dữ liệu từ hệ thống clock1 sang clock2.
- Không thêm logic giữa hai FF này để tránh làm tăng độ trễ → giảm khả năng khắc phục metastability.
- FF1 lấy dữ liệu từ clock1 → FF2 và FF3 đồng bộ lại ở clock2.

Mô phỏng hoạt động – Xuất hiện metastability

Mô tả:

- Clock1 và clock2 có tần số khác nhau → không đồng bộ.
- Dữ liệu q1 thay đổi đúng lúc clock2 đang lấy mẫu → có thể gây ra metastability ở q2 (FF2).
- Dùng FF3** sẽ giúp làm sạch tín hiệu q2 trước khi đến hệ thống clock2.

Các ký hiệu:

- O: hợp lệ (thỏa mãn setup và hold)
- X: vi phạm (dễ gây metastability)

Chi tiết xử lý metastability

Giải thích:

- Dữ liệu q1 từ clock1 thay đổi từ 1 → 0 đúng lúc clock2 lên cạnh.
- Điều này làm q2 có thể ở trạng thái metastability.
- Tuy nhiên, **q2 vẫn chuyển về trạng thái đúng sau một thời gian ngắn.**
- FF3 (q3)** giúp loại bỏ ảnh hưởng của metastability trước khi sử dụng dữ liệu.

Kết luận & Ghi nhớ

- Khi truyền dữ liệu giữa hai clock không đồng bộ:
 - Không dùng trực tiếp FF giữa hai clock.
 - Luôn dùng **hai FF nối tiếp** trong miền clock nhận để tránh metastability.
 - Tránh thêm logic giữa hai FF này.
- Metastability không thể loại bỏ hoàn toàn, nhưng có thể **giảm xác suất xảy ra và ảnh hưởng** bằng các kỹ thuật này.

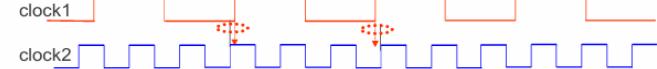
Asynchronous Data Transfer

Synchronous design makes timing issue very simple. However, it cannot be applied to every system. We have to introduce asynchronous design in some cases. Typical cases are:

- Connecting two system running on independent clock
- Getting asynchronous data, such as interrupt, from outside the chip

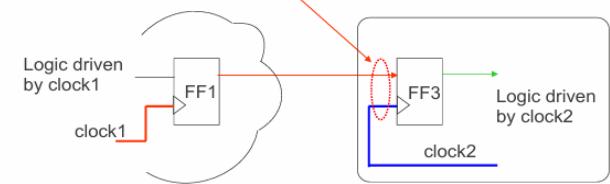
Connecting two system running on independent clock

⇒ Data transfer among the logic with different clock systems

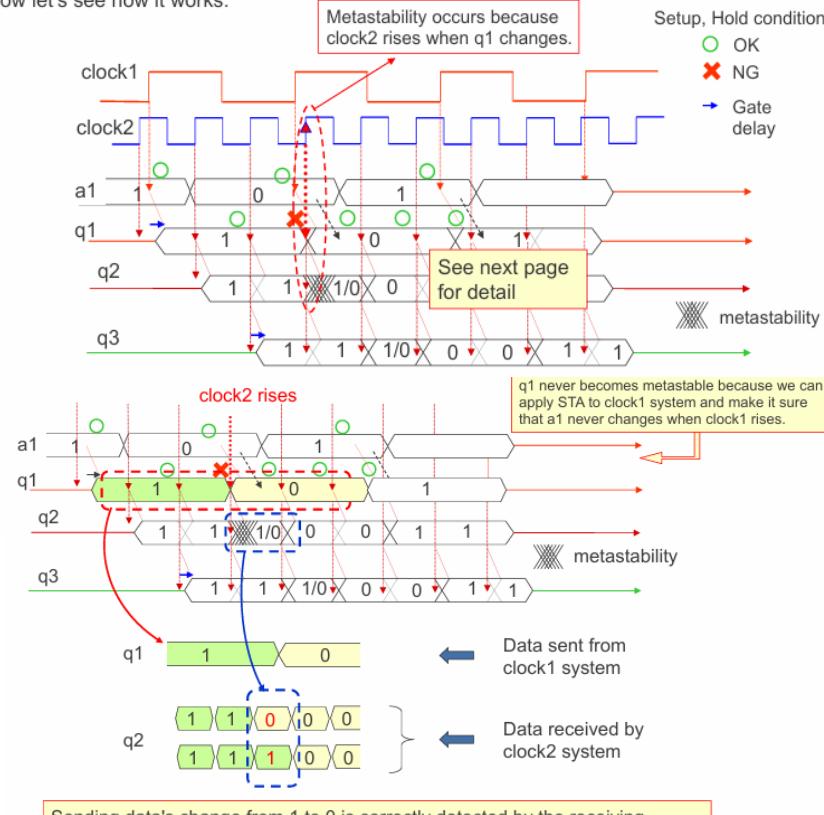


Because clocks 1 and 2 are running without taking any synchronization, their rise edges may collide.

Therefore, FF3 getting data from clock 1 system may fall in metastable state.



Now let's see how it works.



Sending data's change from 1 to 0 is correctly detected by the receiving system because q2 changes from 1 to 0 correctly even if metastability occurs.

4.3. Synchronous Design Summary

- Tất cả các hoạt động logic trong mạch đều được đồng bộ hóa theo một tín hiệu clock duy nhất.
- Tín hiệu clock làm mốc để xác định thời điểm cập nhật dữ liệu trong tất cả các Flip-Flop (FF).
- Thiết kế RTL (Register Transfer Level) có thể tổng hợp đầy đủ.
- Dựa trên các FF và một clock duy nhất → dễ dàng phân tích thời gian (timing analysis).

Implementation

Giải thích sơ đồ mạch:

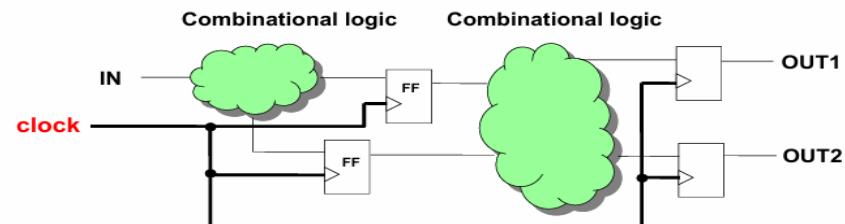
IN → Combinational Logic → FF →

Combinational Logic → OUT

- IN:** Tín hiệu đầu vào.
- Combinational Logic:** Xử lý tín hiệu không đồng hồ, chỉ dựa vào logic tổ hợp.
- FF (Flip-Flop):** Cập nhật dữ liệu theo **cạnh xung clock**.
- OUT1, OUT2:** Tín hiệu đầu ra đã được đồng bộ theo clock.

All operations of one circuit block of interest are uniquely defined in synchronous design with base clock signal.

RTL design ...fully synthesizable design, configured with single clock signal and registers (FF)



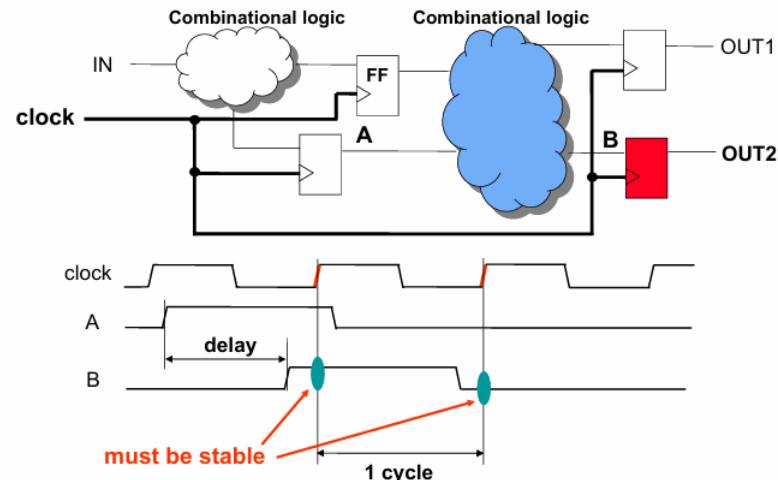
Clock đồng bộ toàn bộ mạch

- Tất cả FF trong sơ đồ đều chia sẻ cùng **một tín hiệu clock** → đảm bảo sự đồng bộ tuyệt đối.
- Điều này giúp:
 - Đảm bảo **tín hiệu ra không bị nhiễu thời gian (timing hazard)**.
 - Dễ dàng thực hiện STA vì mọi đường truyền đều dựa trên chu kỳ clock.

Ưu điểm của thiết kế đồng bộ

- ✓ Dễ phân tích thời gian (timing).
- ✓ Tránh được metastability (nếu không có asynchronous input).
- ✓ Hoạt động ổn định, đáng tin cậy trong công nghiệp.
- ✓ Tổng hợp và triển khai dễ dàng qua công cụ EDA.

One Cycle Operation



Hoạt động trong một chu kỳ clock (One Cycle Operation)

Yêu cầu quan trọng nhất: B must be stable for 1 cycle (giữa hai cạnh clock). Đây là nguyên tắc cốt lõi của phân tích timing: Đầu ra của logic tổ hợp phải ổn định tại đầu vào FF tiếp theo, ít nhất là:

- trước thời điểm clock lên (setup time)
- và giữ nguyên sau clock (hold time)

Liên hệ với Static Timing Analysis (STA)

Trong STA, ta tính toán để đảm bảo rằng:

- Delay của logic tổ hợp từ A đến B < thời gian chu kỳ clock - setup time**
- Đồng thời, không xảy ra **vi phạm hold time** (delay không được quá ngắn)

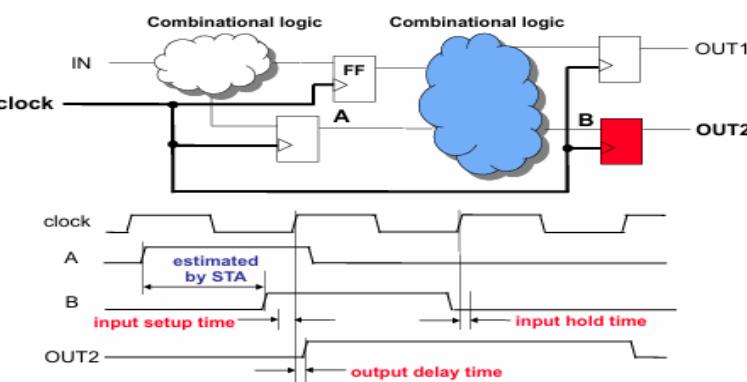
Thiết kế đồng bộ (Synchronous Design)

Cấu trúc cơ bản: Dữ liệu đi qua **logic tổ hợp** → lưu vào **flip-flop (FF)** → truyền sang **logic tổ hợp kế tiếp** → lưu tiếp vào FF → ...

Thành phần chính:

- Clock (xung nhịp):** điều khiển thời điểm FF chốt dữ liệu.
- Combinational Logic (Logic tổ hợp):** xử lý dữ liệu giữa các FF.
- Flip-Flop (FF):** phần tử nhớ, chốt dữ liệu theo cạnh clock.

Synchronous Design



Thời gian liên quan:

- **Setup Time (Tsu):** thời gian dữ liệu phải ổn định trước cạnh lên của clock.
- **Hold Time (Tho):** thời gian dữ liệu phải giữ nguyên sau cạnh clock.
- **Output Delay Time:** thời gian từ cạnh clock đến khi dữ liệu đầu ra ổn định.

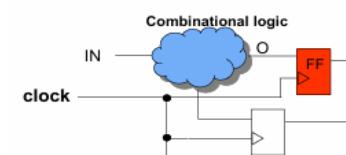
Yêu cầu thời gian (Timing Requirements)

Các thông số chính:

Thông số Ý nghĩa

Tcyc	Chu kỳ clock
Tsu	Setup time (thời gian thiết lập)
Tho	Hold time (thời gian giữ)
Tpd	Propagation delay - độ trễ qua logic tổ hợp

Timing Requirements



Requirements

Tcyc - Tsu > Tpd > Tho

: timing margin

Timing specifications

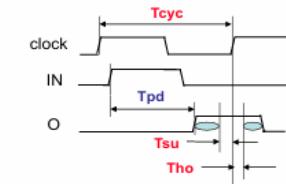
Tcyc: clock cycle time

Tsu: input setup time

Tho: input hold time

Estimated by STA

Tpd: propagation delay time



Điều kiện để mạch hoạt động đúng: Tcyc - Tsu > Tpd > Tho

- **Tcyc - Tsu > Tpd:** logic tổ hợp cần hoàn tất xử lý và dữ liệu phải sẵn sàng tại FF trước thời gian setup.
- **Tpd > Tho:** dữ liệu phải đủ trễ để không vi phạm thời gian giữ.

Lưu ý:

- **Tpd** được xác định bằng kỹ thuật **Static Timing Analysis (STA)**.
- Thiết kế phải đảm bảo **không vi phạm setup & hold time**.

Tóm lại:

Một thiết kế đồng bộ đúng đắn cần đảm bảo rằng:

- Dữ liệu được xử lý kịp thời qua logic tổ hợp.
- Được giữ ổn định đúng lúc trước và sau cạnh xung clock.
- Chu kỳ clock đủ dài để bao trọn các trễ và yêu cầu thời gian.