

# CHAP5: DESIGN FOR TESTABILITY (DFT)

DFT là gì? DFT (Design for Testability) là một phương pháp thiết kế mạch số (đặc biệt trong LSI – Large Scale Integration) nhằm mục đích giúp việc kiểm tra (test) và phát hiện lỗi trên chip dễ dàng hơn sau khi chip được sản xuất.

## DFT dùng để làm gì?

- Phát hiện lỗi sản xuất:
  - Lỗi stuck-at (ngõ ra luôn bằng 0 hoặc 1),
  - Lỗi short (nối tắt),
  - Lỗi open (hở mạch)...
- Tăng khả năng test mạch logic:  
Cho phép kiểm tra các khối logic tuần tự phức tạp mà nếu không có DFT sẽ rất khó tiếp cận.
- Tự động hóa kiểm thử (ATPG):  
Dễ dàng tạo các mẫu kiểm tra tự động (Automatic Test Pattern Generation – ATPG).
- Rút ngắn thời gian kiểm tra:  
Thay vì phải kiểm tra toàn bộ logic phức tạp, DFT giúp kiểm tra nhanh gọn qua các điểm test được thiết kế sẵn.
- Giảm chi phí và nâng cao độ tin cậy:  
Bằng cách phát hiện lỗi sớm, chi phí sửa chữa thấp hơn và sản phẩm tin cậy hơn.

## DFT bao gồm những gì?

- Scan Design (Scan Chain):  
Chèn các scan flip-flop để tạo chuỗi kiểm tra (scan chains).
- Built-In Self Test (BIST):  
Tự kiểm tra chính nó, dùng trong bộ nhớ (Memory BIST), logic (Logic BIST)...
- JTAG / Boundary Scan:  
 Chuẩn IEEE 1149.1, dùng để kiểm tra chân I/O và nội dung trong chip.

## Tại sao DFT quan trọng trong thiết kế LSI?

- Chip càng lớn, số lượng cổng logic càng nhiều ⇒ xác suất lỗi cao hơn.
- Nếu không có DFT, việc kiểm tra sẽ mất thời gian, tốn chi phí và khó phát hiện lỗi sâu bên trong.
- DFT giúp đảm bảo chất lượng sản phẩm, giảm chi phí kiểm tra và bảo trì.

## 5.1 Manufacturing Defects

**Manufacturing defects (lỗi sản xuất)** là những lỗi vật lý xảy ra trong quá trình chế tạo chip (fabrication). Các lỗi này có thể làm cho mạch không hoạt động đúng, thậm chí bị hỏng hoàn toàn.

Trước đây, biến dạng nhỏ trong dây dẫn (như bị uốn cong nhẹ) **không gây ra vấn đề nghiêm trọng**. Nhưng với công nghệ ngày nay (Deep Submicron – tức transistor rất nhỏ), **cùng một mức biến dạng như trước có thể gây lỗi**, ví dụ:

- Hở mạch (open)**
- Điện trở lớn bất thường (large resistance)**

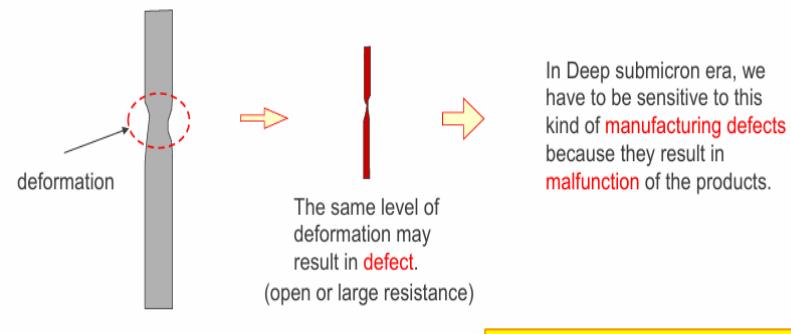
### Vì sao phải quan tâm?

- Trong công nghệ tiên tiến hiện nay, chip có kích thước cực nhỏ, **dễ bị lỗi hơn**.
- Những lỗi nhỏ **rất khó phát hiện bằng mắt thường**, nhưng lại ảnh hưởng lớn đến hoạt động của chip.

### Giải pháp

- Trước khi xuất xưởng sản phẩm, **phải kiểm tra kỹ** để đảm bảo **không có lỗi sản xuất**.
- Dùng các kỹ thuật **DFT (Design for Testability)** để dễ dàng **phát hiện và kiểm tra lỗi** sau khi chế tạo xong.
- “Công nghệ càng nhỏ – lỗi càng nguy hiểm. Kiểm tra kỹ trước khi giao hàng!”

Manufacturing defects are physical defects happening during the manufacturing time.

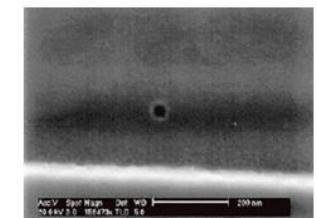


### Physical Defects – Lỗi vật lý trong sản xuất chip

Lỗi vật lý là những **bất thường về mặt cấu trúc** trong các thành phần vật lý của chip (transistor, dây dẫn, via...), thường do quy trình chế tạo chưa hoàn hảo.

#### 1. Lỗi ở Transistor MOS

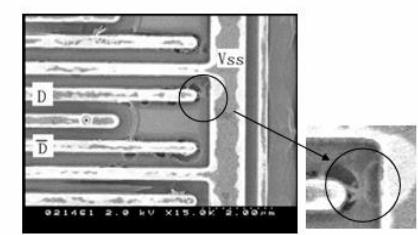
- Lỗi ở lớp gate oxide:** Lớp cách điện giữa cổng (gate) và kênh bị lỗi → transistor hoạt động sai.
- Thiếu transistor:** Do quá trình chế tạo không hình thành được đầy đủ.
- Lỗi về kích thước dây:** Rộng, hẹp hoặc dày không đúng chuẩn → thay đổi dòng điện đi qua.
- Bụi bẩn:** Các hạt lơ rõi vào gây nhiễm bẩn → ảnh hưởng điện trở và dòng.



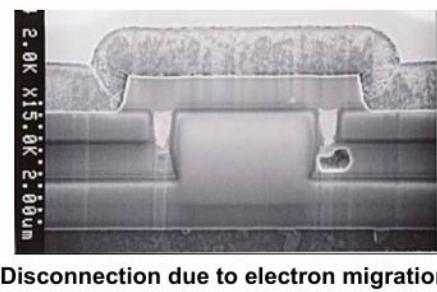
Gate oxide pinhole

#### 2. Lỗi ngắn mạch (shortage in wiring)

- Dư chất sau khi ăn mòn (etching residuum):** Làm cho 2 dây gần nhau bị nối chạm.
- Cầu nối do bụi:** Hạt bụi nối 2 đường dây → tạo ra **ngắn mạch**.
- Dây quá rộng/dày** bất thường → chạm nhau hoặc gây rối tín hiệu.



Shortage due to foreign particle



Disconnection due to electron migration

### Mục đích của kiểm tra (Purpose of Test)

**Mục tiêu chính:** Loại bỏ các lỗi sản xuất (manufacturing defects) Được ví như: "Lá chắn cuối cùng bảo vệ chất lượng của chip LSI trước khi giao đến khách hàng."

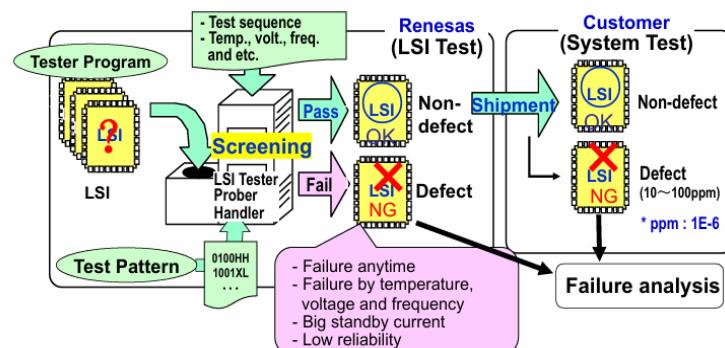
### Purpose of Test

#### Quy trình kiểm tra chip LSI gồm các bước:

##### 1. Test tại công ty sản xuất (ví dụ: Renesas)

- Tester Program:** Hệ thống chạy các chương trình kiểm tra.
- Test Pattern:** Dữ liệu mẫu được đưa vào để kiểm tra phản ứng của chip.
- Test sequence:** Kiểm tra dưới nhiều điều kiện: Nhiệt độ, Điện áp, Tần số
- Kết quả:**
  - Pass → Chip không lỗi → Cho phép xuất hàng.
  - Fail → Chip lỗi → Không giao hàng, cần phân tích lỗi (failure analysis).

- Screening of manufacturing defects
- i.e. "last defense for LSI quality assurance"



##### 2. Test tại phía khách hàng (System Test)

- Sau khi nhận hàng, khách hàng tiếp tục test toàn hệ thống.
- Có thể phát hiện một số lỗi nhỏ (Defect 10~100 x 10<sup>-6</sup> sản phẩm).

### Chi phí kiểm tra (Test Cost) trong thiết kế LSI

Chi phí kiểm tra ngày càng tăng qua các năm (Mặc dù chi phí sản xuất trên mỗi transistor ngày càng giảm.)

#### Nguyên nhân:

- Mạch ngày càng phức tạp.
- Cần test kỹ hơn → tốn thời gian và công cụ hơn.

**Giải pháp:** Cải tiến thiết kế test từ đầu (Test Design Improve) giúp giảm chi phí.

Test cost is increasing year by year

Purpose of test is as follows

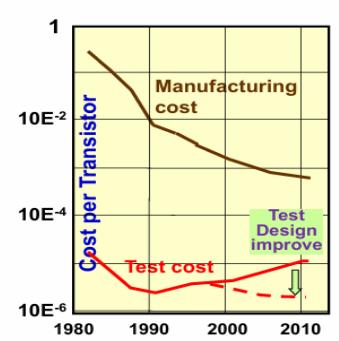
- Maximum quality assurance of LSI
- Adequate test cost

Test Cost (tester-related cost)

- Equipment (tester, prober, handler)
- Operation (labor cost)
  - increase of ratio to manufacturing cost

Test Cost (cost other than tester)

- Test design cost (labor cost)
- Test circuit cost (chip area)
- Failure analysis cost (labor test)



(Source: ITRS2002 Public Conference)

#### Mục tiêu của kiểm tra:

- Đảm bảo **chất lượng tối đa** cho chip LSI.
- Đồng thời giữ được **chi phí kiểm tra hợp lý**.

#### Các loại chi phí kiểm tra bao gồm:

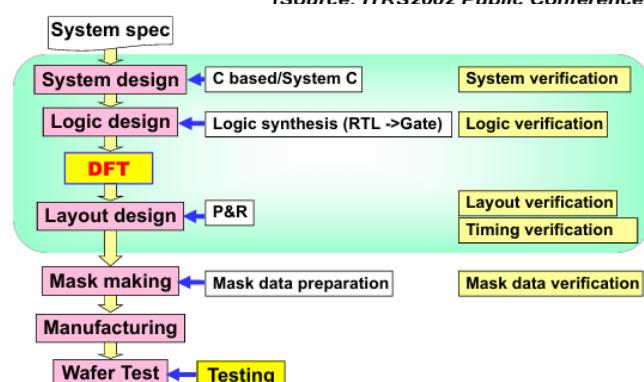
##### 1. Chi phí liên quan đến thiết bị kiểm tra (Tester-related cost):

- Thiết bị: máy kiểm tra (tester), đầu đo (prober), bộ xử lý (handler).
- Nhân công vận hành thiết bị.

**Chi phí này ngày càng chiếm tỷ lệ lớn trong tổng chi phí sản xuất.**

##### 2. Chi phí khác (Cost other than tester):

- Thiết kế mạch kiểm tra (test design): tốn công sức kỹ sư.
- Mạch kiểm tra tích hợp trên chip:** chiếm diện tích → tăng chi phí.
- Phân tích lỗi sau khi test:** tốn thời gian và nhân lực.



#### DFT trong LSI Design Flow

- Thiết kế các cấu trúc hỗ trợ kiểm tra ngay từ đầu.
- Hỗ trợ sinh test pattern, kiểm tra hiệu quả sau sản xuất.

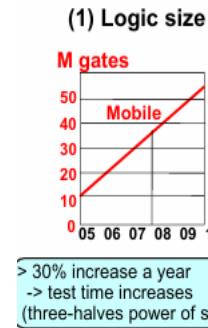
#### Tác động của Xu hướng Thiết kế SoC đến Test (Impact of SoC Design Trend)

**Tình hình thực tế:** Hiện nay, các thiết kế SoC (System-on-Chip) ngày càng:

- Có số lượng cổng logic tăng mạnh.
- Có bộ nhớ tích hợp lớn hơn.
- Tốc độ hoạt động (tần số xung nhịp) tăng cao.

## (1) Logic size tăng – (M gates)

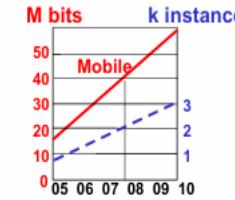
- Mỗi năm tăng >30% về số lượng cổng logic.
- Thời gian test tăng theo **lũy thừa ba phần hai** của kích thước (Three-halves power).
- Giải pháp DFT:**
  - LBIST:** Logic Built-In Self-Test.
  - Compression Test:** Giảm dữ liệu test nhưng vẫn giữ chất lượng kiểm tra.



> 30% increase a year  
→ test time increases  
(three-halves power of size)

LBIST or Compression Test

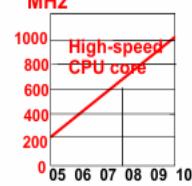
## (2) Memory



> 30% increase a year  
→ test time increases  
(proportion to size)

Concurrent measurement using MBIST

## (3) Frequency



> 30% increase a year  
→ High-speed tester

Enable to lower external clock freq. using PLL  
(around 30 MHz)

DFT reduces the increased test time and needs no tester of high specification

## (2) Memory size tăng – (M bits & k instances)

- Bộ nhớ tích hợp trong SoC tăng đều theo năm.
- Thời gian test tăng theo **tỷ lệ thuận với kích thước bộ nhớ**.
- Giải pháp DFT:**
  - MBIST** (Memory BIST): Tự kiểm tra bộ nhớ.
  - Hỗ trợ kiểm tra **nhiều khối bộ nhớ song song** (Concurrent measurement).

## (3) Frequency tăng – (MHz)

- CPU core tốc độ cao (High-speed CPU core) tăng lên tới hàng GHz.
- Cần **thiết bị test rất nhanh** để bắt kịp.
- Giải pháp DFT:**
  - Dùng **PLL** để test bằng **xung nhịp bên ngoài thấp** (~30 MHz).
  - Không cần tester tốc độ cực cao → **giảm chi phí tester**.

**DFT giúp khắc phục những khó khăn do xu hướng tăng trưởng mạnh của SoC:**

- Cổng logic nhiều → Test lâu → cần LBIST / Nén test
- Bộ nhớ lớn → Test lâu → cần MBIST, kiểm tra song song
- Xung nhịp cao → Cần tester mạnh → Dùng PLL, giảm yêu cầu tester

## Test Methodology (Phương pháp kiểm tra)

Tiêu chí	Functional Test	Structural Test
Khái niệm (Concept)	- Kiểm tra chức năng thực tế của chip. - Giống như điều kiện hoạt động ngoài thực tế.	- Tập trung vào phát hiện lỗi sản xuất (manufacturing). - Dùng mô hình lỗi (fault modeling) để kiểm tra toàn diện.
Quyết định đạt/trượt	- Dựa trên việc chip có hoạt động đúng chức năng hay không.	- Dựa vào việc có lỗi sản xuất hay không ( thông qua mô hình lỗi ).
Chỉ số bao phủ lỗi (Coverage Measure)	- Stuck-at fault (lỗi cố định tại 0 hoặc 1). - Kiểm tra đường tới hạn (critical path) thủ công.	- Dùng công cụ để đánh giá bao phủ lỗi một cách định lượng theo từng mô hình lỗi.
Khả năng đánh giá bao phủ	- Khó đánh giá định lượng vì không có mô hình lỗi cụ thể.	- Dễ đánh giá định lượng, có thể tính % bao phủ lỗi.
Chất lượng kiểm tra	- Không có cơ sở khoa học để đánh giá tiến bộ.	- Có thể đánh giá bằng định lượng khoa học, rất quan trọng để chọn mô hình lỗi phù hợp.

### Tóm lược:

- Functional Test:**
  - Ưu điểm:** Mô phỏng điều kiện sử dụng thật.
  - Nhược điểm:** Khó đánh giá độ bao phủ, không đảm bảo phát hiện hết lỗi sản xuất.
- Structural Test:**
  - Ưu điểm:** Dễ đánh giá độ bao phủ lỗi, rất thích hợp cho kiểm tra sản xuất (DFT).
  - Dựa** trên các mô hình lỗi như stuck-at, transition, bridging

### Vì sao DFT sử dụng Structural Test?

- Vì DFT hướng tới tối ưu hóa việc kiểm tra lỗi sản xuất: nhanh, hiệu quả, có thể định lượng chất lượng test.
- Functional Test khó dùng trong môi trường sản xuất hàng loạt, do tốn công sức và độ bao phủ không đảm bảo.
- DFT ưu tiên Structural Test** vì dễ tự động hóa, bao phủ tốt lỗi sản xuất. Nhưng Functional Test vẫn:
- Hữu ích ở giai đoạn cuối** để kiểm tra toàn diện.
  - Hỗ trợ kiểm tra logic phức tạp**, luồng dữ liệu, tương tác IP.
  - Bổ sung cho Structural Test**, đảm bảo hệ thống hoạt động như mong muốn.

## Khái niệm về Fault Coverage (Độ phủ lỗi)

- Fault Coverage** là tỉ lệ phần trăm các lỗi trong mạch có thể bị phát hiện bằng các mẫu kiểm tra (test pattern) đã sinh ra.
- Đây là một **chỉ số quan trọng** trong kiểm thử mạch tích hợp (LSI) để đánh giá hiệu quả của quá trình test.
- Công thức tính:** Fault Coverage = (Số Lỗi Phát Hiện Được / Tổng số lỗi tồn tại trong mạch) x100

- Fault Coverage còn được gọi là **Failure Detection Rate** (tỉ lệ phát hiện lỗi).
- Việc ước lượng Fault Coverage thường được thực hiện dựa vào các **Fault Models** (mô hình lỗi), như: Stuck-at fault, Bridging fault, Delay fault, Transition fault, v.v.

### Tại sao Fault Coverage quan trọng?

- Giúp đánh giá độ tin cậy của sản phẩm.
- Fault Coverage càng cao, nguy cơ sản phẩm lỗi khi xuất xưởng càng thấp.
- Là mục tiêu quan trọng trong thiết kế DFT (Design for Testability).

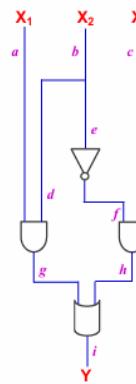
### Stack-at-Fault Model

- Đây là một mô hình **giả định lỗi phổ biến nhất** trong kiểm thử mạch số.
- Giả định** rằng một tín hiệu (đường dây, cổng logic) bị **kẹt (stuck)** tại giá trị logic **0 (SA0)** hoặc **1 (SA1)** bất kể đầu vào là gì.

Modeling Faults Example: Stack-at-Fault Model

#### Giải thích sơ đồ bên trái (Mạch logic):

- Mạch bao gồm: 2 cổng AND, 1 cổng NOT, 1 cổng OR
- Các đường dây được đánh dấu từ **a → i**
- Đầu vào: X1, X2, X3
- Đầu ra: Y

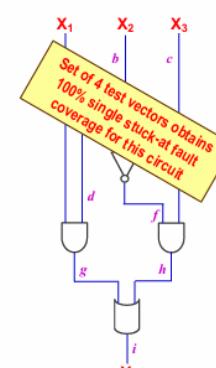


X <sub>1</sub> X <sub>2</sub> X <sub>3</sub>	000	001	010	011	100	101	110	111
Y	0	1	0	0	0	1	1	1
<b>a</b>	0	0	0	0			0	0
	SA0	1	1	1	1	1	1	1
<b>b</b>	0	0	1	1	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>c</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>d</b>	0	0	1	1	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>e</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>f</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>g</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>h</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>i</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1

#### Bảng kiểm thử bên phải:

- Cột ngang: Các tổ hợp đầu vào (X1, X2, X3) từ 000 đến 111
- Cột dọc: Các vị trí lỗi tại từng dây (a, b, c, ..., i) với hai trường hợp:
  - SA0:** kẹt ở 0
  - SA1:** kẹt ở 1
- Bảng hiển thị **kết quả đầu ra Y** với mỗi lỗi giả định

Modeling Faults Example: Stack-at-Fault Model



X <sub>1</sub> X <sub>2</sub> X <sub>3</sub>	000	001	010	011	100	101	110	111
Y	0	1	0	0	0	1	1	1
<b>a</b>	0	0	0	0			0	0
	SA0	1	1	1	1	1	1	1
<b>b</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>c</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>d</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>e</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>f</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>g</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>h</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1
<b>i</b>	0	0	0	0	0	0	0	0
	SA0	1	1	1	1	1	1	1

#### Mục đích của mô hình Stack-at-Fault:

- Giúp đánh giá **khả năng phát hiện lỗi** (fault coverage)
- Là cơ sở cho công cụ **tự động tạo mẫu kiểm thử (ATPG)**

#### Ghi nhớ:

- SA0 (Stuck-at-0):** Tín hiệu luôn bằng 0
- SA1 (Stuck-at-1):** Tín hiệu luôn bằng 1
- Cần dùng **tập mẫu đầu vào thích hợp** để phát hiện từng lỗi

## 5.2. DFT and Scan Method.

### What is DFT? (Design for Testability)

Định nghĩa: DFT là kỹ thuật thiết kế dùng để chèn thêm mạch kiểm thử vào phần cứng LSI (Large Scale Integration).

Mục đích là giúp dễ dàng áp dụng các bài kiểm thử lên LSI sau khi chế tạo.

#### DFT gồm các bước chính:

- Tạo mạch kiểm thử** (Creating test circuits)
- Chèn mạch kiểm thử vào LSI** (Inserting test circuits into hardware)
- Chuẩn bị dữ liệu kiểm thử** để phục vụ phân tích lỗi vật lý (Preparing test data for physical failure analysis)

**Mục đích của DFT:** Đảm bảo chất lượng sản phẩm LSI, Cải thiện quy trình sản xuất, tối ưu chi phí kiểm thử

## Các mục tiêu chính của DFT:

- Giảm: Area overhead (diện tích tăng thêm) Test time (thời gian kiểm thử) Test cost (chi phí kiểm thử)
- Tăng: Test/fault coverage (phạm vi phát hiện lỗi)

## Required Quality and Test Level

### 1. Required Quality (Chất lượng sản phẩm yêu cầu)

- Zero defect → Mục tiêu là không có lỗi nào trong sản phẩm xuất xưởng.
- Guarantee of operation under warranty condition:  
→ Đảm bảo sản phẩm hoạt động ổn định trong điều kiện bảo hành, như: nhiệt độ, điện áp, tần số...

### 2. Test Level (Cấp độ kiểm thử) Các yếu tố cần thiết để đảm bảo sản phẩm đạt yêu cầu chất lượng:

- Test items** → Các hạng mục cần kiểm tra (chức năng, timing, tiêu thụ điện, v.v.)
- Test conditions** → Điều kiện kiểm thử mô phỏng môi trường thực tế: Temperature, Voltage, Frequency/Timing, v.v.
- Test model** (mô hình kiểm thử):
  - Đại diện cho mức độ đầy đủ của bài kiểm thử.
  - Ví dụ: sử dụng stuck-at fault model, transition delay model, v.v.
- Target fault coverage**
  - Mức độ bao phủ lỗi cần đạt được (thường > 95%)
  - Được đánh giá định lượng để đảm bảo bài test đủ mạnh.

## Target Quality of Renesas Products

**Mục tiêu:** Chuẩn hóa mức chất lượng được người tiêu dùng yêu cầu, ứng dụng theo từng nhóm sản phẩm.

### Ghi chú:

- Mỗi cấp độ chất lượng đều tương ứng với **mức độ yêu cầu kiểm thử khác nhau**, bao gồm cả **test model**, **fault coverage**, và **test time**.
- DFT đóng vai trò **cực kỳ quan trọng trong các sản phẩm Q1A/Q1B**, nơi yêu cầu thời gian sử dụng dài và độ tin cậy gần như tuyệt đối.

## Target Quality of Renesas Products

Standardization of common quality level requested by Consumer

Quality Classification		Guaranteed Lifetime	Main products
High Reliability	Q1A	20 years	Automobile parts (Engine controller, etc.) General traffic equipment
	Q1B	10 years	Car electronics (accessories: genuine goods, etc.)
Industry	Q2	10 years	Car electronics, FA equipment, etc.
Consumer	Q3	10 years	PC, Consumer electronics, Mobile equipment
Custom	QX	Determine individually for each product	Game instruments, Ultra high reliable equipment

## Structural Test – Kiểm tra cấu trúc mạch LSI

**Mục tiêu của Structural Test:** Để đảm bảo khối C1 của mạch LSI được sản xuất không lỗi, chúng ta cần kiểm tra:

- Các dây nối: W1, W2, W3, W4, W5
- Các cổng logic: G1 (AND), G2 (OR)

### Vấn đề với Functional Test

- Trong thực tế, **các nhà thiết kế logic thường không biết cổng nào được tạo ở đâu**, và cũng không biết các dây nối cụ thể như thế nào sau khi chế tạo.
- Điều này làm cho việc kiểm tra qua **functional test (kiểm tra hoạt động logic tổng thể)** trở nên không hiệu quả vì:  
→ Không thể **đặt giá trị đầu vào cụ thể** để kiểm tra riêng từng dây hoặc cổng logic sau khi chế tạo.

Do đó: **Functional test không đủ để phát hiện lỗi sản xuất.**

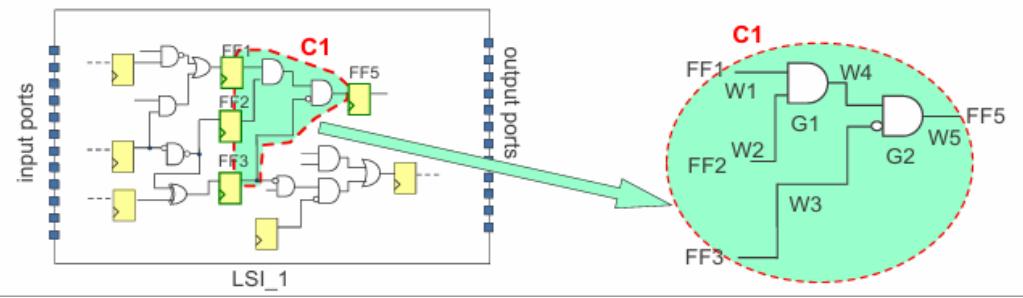
### Lợi ích của Structural Test

- Khi đã có **Gate Netlist** (mô tả cấu trúc cổng logic bên trong IC), ta có thể xác định:
  - Với các đầu vào cụ thể từ FF1, FF2, FF3 thì đầu ra tại W5 **phải có giá trị gì**.
  - Từ đó, nếu giá trị thực tế tại W5 **không khớp** với giá trị mong đợi, ta có thể suy ra **mạch có lỗi sản xuất**.

## Ví dụ về Structural Test

### Mạch C1 bao gồm:

- FF1, FF2, FF3: các thanh ghi (flip-flop) đầu vào
- W1 đến W5: dây kết nối tín hiệu
- G1: cổng AND
- G2: cổng OR



**Cần cần dùng các mẫu A, B, C, D, F, H** là đủ để kiểm tra toàn bộ logic vì Pattern E và G bị **bao phủ** bởi các mẫu khác (C, F).

### Kết luận:

- Structural test sử dụng mô hình mạch thực tế** để kiểm tra xem mỗi phần tử (cổng logic, dây nối) có hoạt động đúng sau khi sản xuất không.
- Giúp **phát hiện lỗi sản xuất phần cứng**, điều mà functional test không làm được hiệu quả.
- Dựa trên bảng mẫu input/output**, ta có thể xác định chính xác các lỗi xảy ra nếu mạch có vấn đề.

(CÓ THỂ CÓ CÂU HỎI BÀI TẬP VỀ STRUCTURAL TEST)

### Bảng giá trị đầu vào và đầu ra mong đợi

Pattern	FF1	FF2	FF3	W5 Output
A	1	1	0	0
B	1	0	1	1
C	1	1	1	1
D	1	0	0	0
E	1	1	0	0 (đã có ở pattern A)
F	0	1	0	0
G	0	1	0	0 (giống F)
H	0	0	0	0

### Scan Method – Phương pháp kiểm tra bằng Scan

#### Scan là gì?

- Là **phương pháp sử dụng các Flip-Flop (FF)** hoặc latch (chốt) để **thiết lập giá trị đầu vào** cho các đường W1, W2, W3.
- Sau đó, **quan sát và ghi nhớ giá trị đầu ra** tại W5 thông qua FF5.

**Mục tiêu của Scan Method:** Để kiểm tra khối logic (C1) có hoạt động đúng không (tức là không bị lỗi sản xuất), chúng ta:

- Tính toán được đầu ra** tại W5 nếu biết các đầu vào W1, W2, W3 (dựa vào netlist – danh sách kết nối cổng).
- Quan sát đầu ra** W5, sau đó so sánh với kết quả mong đợi để kiểm tra lỗi.

#### Cấu trúc mạch đang xét:

##### C1 gồm:

- Đầu vào:** FF1 → W1, FF2 → W2, FF3 → W3
- Cổng logic:** G1 (AND), G2 (OR)
- Đầu ra:** W5 → FF5

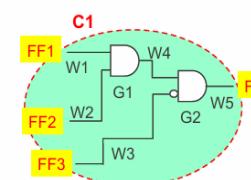
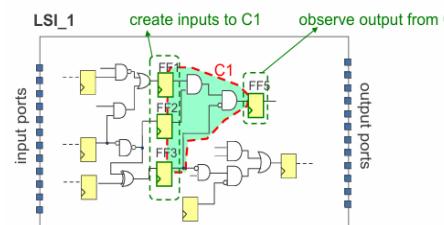
FF1, FF2, FF3 thiết lập các giá trị đầu vào

FF5 lưu trữ (memorize) kết quả đầu ra tại W5

#### Scan Method

For the logic block shown on the right, we can calculate what must come out at W5 output when we apply some input patterns on W1, W2, and W3, based on the net list information.

Scan is a method to use FF or latch to set input patterns on W1, W2, and W3 and observe (memorize) the output on W5.



**MUXscan** is an idea to use FF1, FF2, and FF3, created from the RTL code by a synthesis tool, to set values on W1, W2, and W3 respectively and use FF5 to observe output W5.

#### Tóm lại:

Scan Method: Dùng FF để đặt giá trị vào các đầu vào mạch cần kiểm tra, sau đó dùng FF khác để ghi nhận đầu ra  
Ưu điểm: Dễ dàng thiết lập và quan sát, phát hiện lỗi logic và sản xuất chính xác

Ứng dụng: Kiểm thử cấu trúc LSI, áp dụng trong kiểm thử tự động (ATPG) và DFT

MUXscan: Cách tận dụng FF từ RTL để gắn test point tự động qua tổng hợp

### MUXscan – Scan Flip-Flop trong Kiểm thử DFT

#### MUXscan là gì?

- MUXscan** là một ý tưởng trong kỹ thuật thiết kế kiểm thử tự động (DFT).
- Ý tưởng: sử dụng **FF1, FF2, FF3** (được tổng hợp từ RTL code bằng công cụ tổng hợp như synthesis tool) để:
  - Đặt giá trị đầu vào cho W1, W2, W3**
  - Quan sát kết quả tại W5** bằng cách đọc giá trị lưu tại FF5

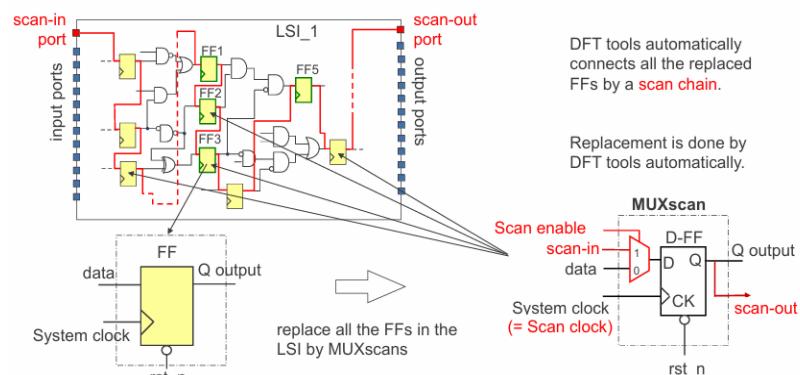
Nói cách khác, **MUXscan biến các FF thành các điểm kiểm tra để:**

- Tạo đầu vào tùy chỉnh**
- Đọc đầu ra dễ dàng**

**Mục tiêu của MUXscan:** Để thực hiện kiểm thử DFT (Design for Testability) hiệu quả, ta cần có khả năng:

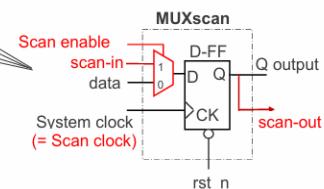
- Đặt các giá trị cụ thể tại các Flip-Flop (FF) trong mạch.**
- Đọc giá trị từ các FF đầu ra** sau khi logic được thực thi.

To apply **MUXscan** we have to replace all FFs in the LSI by "DFT ready FF", specially designed FF suitable for MUXscan.



DFT tools automatically connects all the replaced FFs by a **scan chain**.

Replacement is done by DFT tools automatically.



## Thay thế tất cả FF bằng MUXscan FF:

- Mọi FlipFlop trong LSI** (ví dụ FF1, FF2, FF3, FF5) **được thay thế bằng loại đặc biệt gọi là “DFT ready FlipFlop”** — là **MUXscan Flip-Flop**.
- Quá trình này **tự động được thực hiện bởi các công cụ DFT**, như các phần mềm của Synopsys hoặc Cadence.

## Cấu trúc MUXscan Flip-Flop gồm những gì?

So sánh giữa Flip-Flop thường và MUXscan:

Thành phần	FF thường	MUXscan FF
Đầu vào chính	data	data
Clock	System clock	System clock (dùng làm Scan clock)
Đầu ra	Q	Q
Dành cho DFT	✗	✓ scan-in, scan-out, scan-enable

## MUXscan hoạt động như sau:

- Chế độ hoạt động thông thường:**
  - scan-enable = 0
  - FF hoạt động như bình thường với data từ hệ thống.
- Chế độ scan (test mode):**
  - scan-enable = 1
  - data được chọn từ đường scan-in (thay vì hệ thống).
  - Các FF được nối tiếp thành một scan chain, và dữ liệu quét vào FF1 → FF2 → FF3 → ... → FF5.

## Scan Chain là gì?

Là chuỗi các FF nối nhau qua các cổng scan-in và scan-out giúp truyền dữ liệu kiểm thử:

- Scan-in port: Nhận dữ liệu kiểm thử từ bên ngoài.
- Scan-out port: Xuất kết quả dữ liệu kiểm thử ra bên ngoài.

Nhờ scan chain, ta có thể:

- Gửi giá trị kiểm thử vào tất cả các FF một cách chính xác.
- Đọc tất cả giá trị đầu ra để so sánh với kết quả mong đợi.

## Lợi ích của MUXscan và Scan Chain:

- Cho phép **quan sát và điều khiển hoàn toàn bên trong mạch** LSI, thứ vốn không thể làm được bằng kiểm thử thông thường.
- Tự động hóa kiểm thử** bằng các công cụ ATPG (Automatic Test Pattern Generation).
- Tăng độ phủ lỗi (fault coverage)** mà vẫn **giảm chi phí kiểm thử**.

## Hoạt động của Scan Chain trong công cụ DFT

**Mục tiêu:** Cho phép nạp dữ liệu kiểm thử vào các Flip-Flop (FF) trong một chuỗi, rồi thực thi mạch trong 1 chu kỳ xung nhịp để lấy kết quả đầu ra → từ đó kiểm tra lỗi sản xuất trong mạch.

### (1) Scan-in Operation – Ghi mẫu kiểm thử vào FF

- Scan-in port được dùng để nạp một chuỗi bit kiểm thử, ví dụ: xxx--x101xx--xxxx (x = don't care)
- Mỗi lần có một cạnh lên (rising edge) của xung nhịp scan, một bit dịch vào FF tiếp theo.

**Kết quả:** 1 vào FF1 0 vào FF2 1 vào FF3 ✨ Gọi là thao tác scan-in – đây là giai đoạn "nạp mẫu kiểm thử" vào mạch.

### 2) Capture Operation – Chạy mạch để thu kết quả

- Sau khi các FF đã chứa mẫu cần thiết (ví dụ FF1 = 1, FF2 = 0, FF3 = 1), ta chạy mạch LSI **trong đúng 1 chu kỳ hệ thống**.
- Lúc này, mạch logic **C1** sẽ xử lý các tín hiệu W1, W2, W3 (lấy từ FF1-3) và xuất ra **W5**, lưu vào **FF5**.

**Gọi là thao tác capture** – giúp thu lại giá trị logic đầu ra để kiểm thử.

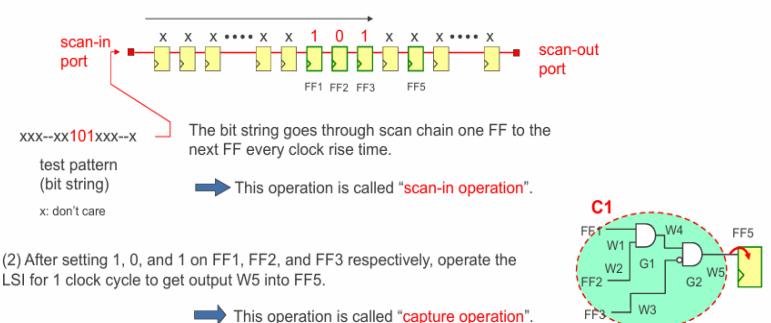
### (3) Scan-out Operation – Đọc dữ liệu đầu ra từ FF5

- Sau khi thực hiện **Capture Operation** (tức là mạch đã tính toán và lưu kết quả vào FF5), ta cần lấy giá trị này ra để so sánh.
- Scan-out Operation** sẽ dịch dữ liệu từ các FF qua **scan-out port**, tương tự như scan-in nhưng theo chiều ngược lại.

Giá trị trong FF5 = 0 → được dịch ra ngoài ở vị trí đã biết (vd: vị trí thứ 8 từ phải). Mỗi lần có **cạnh lên của xung đồng hồ**, một bit từ FF sẽ đi ra ngoài qua **scan chain**.

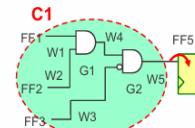
Scan chain created by a DFT tool works as shown below:

(1) If we prepare bit string like xxx--x101xx--xxxx at scan in port and apply clock signal to FFs, then the bit string goes into FFs through the scan chain and as a result 1, 0, and 1 are set to FF1, FF2, and FF3 respectively.

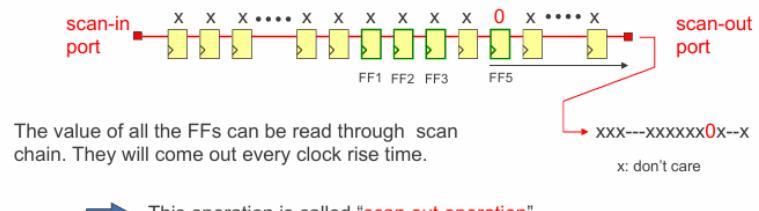


(2) After setting 1, 0, and 1 on FF1, FF2, and FF3 respectively, operate the LSI for 1 clock cycle to get output W5 into FF5.

→ This operation is called "capture operation".



(3) After capture operation, the value of FF5 can be got from the scan out port through scan chain as shown below.



(4) The value of FF5 came through the scan chain is compared to the expected result of C1. If it matches with the expected results for all the test pattern, then C1 is manufactured without defects.

#### (4) So sánh kết quả để kiểm tra lỗi sản xuất

- Giá trị lấy được từ FF5 (qua scan chain) sẽ được **so sánh với giá trị kỳ vọng** của mẫu kiểm thử đang áp dụng.
- Nếu kết quả đầu ra **khớp với kết quả mong đợi** cho tất cả các mẫu kiểm thử → phần tử logic **C1 được xác nhận là không lỗi sản xuất**.

Nếu sai lệch xuất hiện → có lỗi trong mạch (do lỗi kết nối, sai gate, v.v.)

#### Tổng kết toàn bộ quy trình kiểm thử DFT với Scan Chain:

- Scan-in operation: Ghi bit mẫu kiểm thử vào các FF
- Capture operation: Cho mạch chạy trong 1 chu kỳ để tạo kết quả
- Scan-out operation: Đọc giá trị đầu ra từ các FF
- So sánh kết quả: Đối chiếu với dữ liệu kỳ vọng để phát hiện lỗi

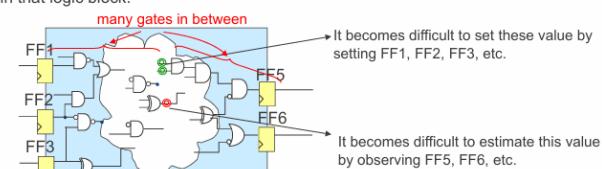
#### Test Point Insertion (TPI) – Chèn điểm kiểm thử trong DFT

Vấn đề khi kiểm thử logic block phức tạp:

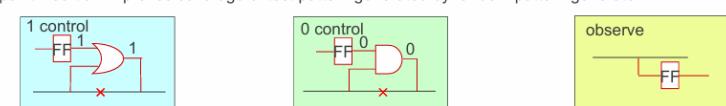
- DFT (Design For Test) tools sử dụng netlist để tạo ra các mẫu dữ liệu kiểm thử (test patterns) kiểm tra các khối logic trong IC.
- Tuy nhiên, nếu giữa các flip-flop (FF) có quá nhiều cổng logic trung gian (many gates in between), thì sẽ:
  - Khó đưa được giá trị cụ thể đến một FF sâu trong mạch.
  - Khó quan sát được đầu ra để xác nhận logic đã hoạt động đúng.

#### Test Point Insertion (TPI)

Because a DFT tool knows the gate net list, it can create data pattern which can test logic blocks in the LSI. However, if the logic scale between FFs is very large, then it is difficult to create patterns to check-out all the defects in that logic block.



Inserting the following logics in proper signal line, setting and observing signals become very easy. This test point insertion improves coverage of test pattern generated by random pattern generator.



Ví dụ trong sơ đồ:

- Rất khó để đặt giá trị đầu vào (từ FF1, FF2, FF3) sao cho giá trị mong muốn đi qua hết các cổng logic trung gian đến được FF5.
- Cũng rất khó để hiểu giá trị đầu ra tại FF6 phản ánh điều gì từ đầu vào, vì mạch quá phức tạp.

#### Giải pháp: Test Point Insertion (TPI)

TPI là kỹ thuật chèn thêm các điểm kiểm thử đặc biệt (test points) vào trong mạch để:

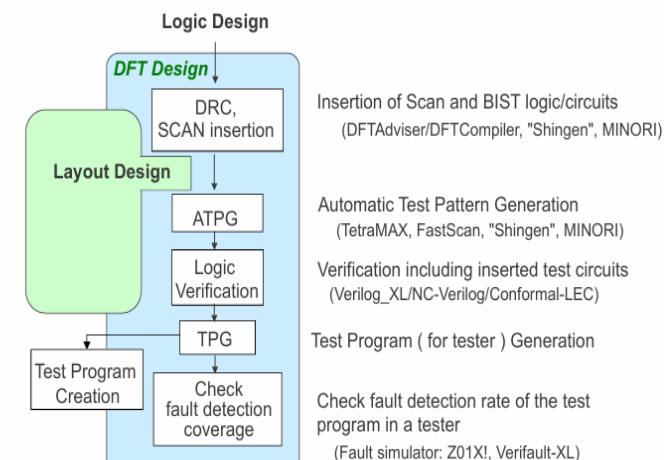
- Tăng khả năng kiểm soát tín hiệu (Controllability)
- Tăng khả năng quan sát tín hiệu (Observability)

Có 2 loại test point:

##### 1. Control Point (Điểm điều khiển):

- Chèn một cổng logic (như AND) có thể ép tín hiệu thành 0 hoặc 1.
- Ví dụ:
  - Chèn AND với đầu vào "0" sẽ ép đầu ra thành 0 (ép tín hiệu)
  - Chèn OR với đầu vào "1" sẽ ép đầu ra thành 1

##### DFT Design Flow



##### 2. Observe Point (Điểm quan sát):

- Chèn thêm một FF hoặc đường tín hiệu để dễ dàng quan sát giá trị tại vị trí giữa mạch.

#### Lợi ích của TPI:

- Dễ dàng tạo mẫu kiểm thử hiệu quả hơn (tăng fault coverage)
- Giảm độ phức tạp trong việc sinh test pattern
- Cải thiện khả năng phát hiện lỗi sâu bên trong mạch

### 5.3. DFT Design Flow.

Mục tiêu: Đảm bảo rằng mạch logic được thiết kế có thể kiểm tra được lỗi sau khi sản xuất (manufacturing), thông qua các bước thiết kế kiểm thử tích hợp vào thiết kế logic. **Các bước chính trong quy trình DFT Design Flow:**

#### 1. DFT Design: DRC & SCAN Insertion:

- Chèn các mạch SCAN (DFT ready FFs, Scan Chains), hoặc mạch BIST (Built-In Self Test) vào mạch logic.
- Công cụ hỗ trợ: DFT Adviser, DFT Compiler, Shingen, MINORI

## 2. ATPG (Automatic Test Pattern Generation)

- Tạo tự động các mẫu kiểm thử dựa trên netlist có SCAN đã được chèn.
- Công cụ hỗ trợ: TetraMAX, FastScan, Shingen, MINORI

## 3. Logic Verification

- Kiểm tra lại mạch logic sau khi thêm các phần tử kiểm thử.
- Đảm bảo mạch vẫn hoạt động đúng và logic không bị sai lệch.
- Công cụ hỗ trợ: Verilog\_XL, NC-Verilog, Conformal-LEC

## 4. TPG (Test Program Generation)

- Tạo chương trình kiểm thử để chạy trên tester (máy kiểm tra vật lý sản phẩm thật)

## 5. Fault Coverage Check

- Đánh giá **độ bao phủ lỗi (fault detection coverage)** của các mẫu kiểm thử đã tạo ra.
- Kiểm tra xem bao nhiêu phần trăm lỗi có thể bị phát hiện.
- Công cụ hỗ trợ: Z01X, Verifault-XL

### SCAN Design Flow

Các bước chính:

#### 1. DRC (Design Rule Check)

- Kiểm tra xem thiết kế có thỏa các ràng buộc để sử dụng phương pháp **MUX scan** hay không.

o Dùng thư viện **DRC Library** để xác minh quy tắc thiết kế.

#### 2. Scan Insertion

- Chèn **scan chain** vào netlist.
- Khi dùng **scan compression**, sẽ chèn thêm các mạch nén và giải nén tín hiệu.

o Đầu vào:

- **Gate-level Netlist**
- **Stitching Library** (thư viện để nối FF thành chuỗi)

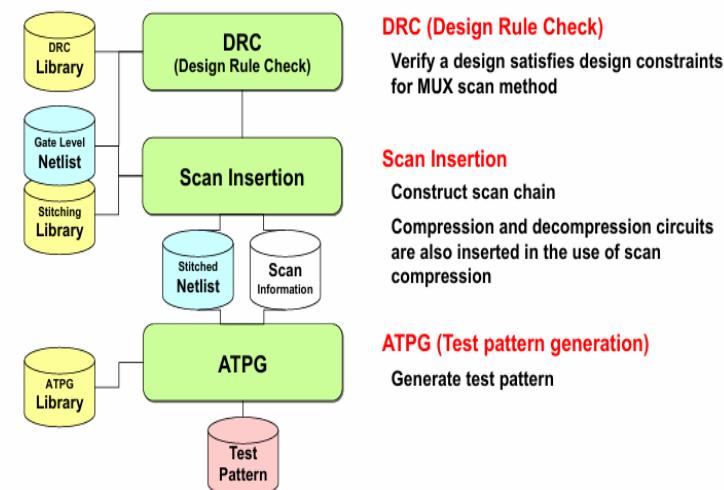
o Đầu ra:

- **Stitched Netlist** (Netlist đã có scan chain)
- **Scan Information** (dùng cho bước ATPG)

#### 3. ATPG (Automatic Test Pattern Generation)

- Sinh ra các **test patterns** để kiểm tra logic bên trong chip.
- Dựa trên netlist đã thêm scan và thông tin scan.
- Đầu ra: **Test Pattern** (nạp vào tester để kiểm tra chip thực tế).

## SCAN Design



#### DRC (Design Rule Check)

Verify a design satisfies design constraints for MUX scan method

#### Scan Insertion

Construct scan chain  
Compression and decompression circuits are also inserted in the use of scan compression

#### ATPG (Test pattern generation)

Generate test pattern

DRC: Đảm bảo thiết kế tương thích với scan

Scan Insertion: Chèn chuỗi scan vào netlist

ATPG: Tạo ra các mẫu kiểm tra logic tự động

### DRC – Design Rule Check trong SCAN Design

Mục tiêu chính:

- Xác minh thiết kế **thỏa mãn các ràng buộc cần thiết cho phương pháp MUX scan**.
- Phát hiện lỗi sớm và **hỗ trợ phân tích/debug hiệu quả**.

Chi tiết về IO Configuration:

- Constraint:** Giá trị cố định cho nguồn clock, test mode,...
- Rule:** Các quy tắc thiết kế phổ thông dùng cho scan (ví dụ: không dùng FF có reset async trong chuỗi scan).
- Result:** Kết quả kiểm tra, bao gồm bản tổng hợp lỗi để debug.

Các thành phần chính trong DRC:

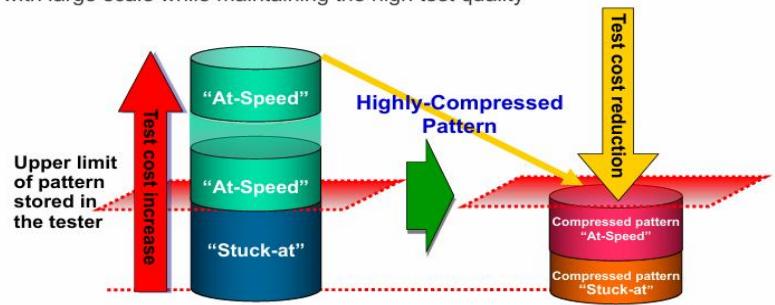
- Netwalker Library:** Thư viện phân tích kết nối các phần tử trong netlist
- Gate Level Netlist:** Mô tả thiết kế ở mức cổng logic
- Constraint:** Ràng buộc về clock, chế độ test, vị trí FF,...
- Rule:** Các quy tắc thiết kế phổ biến cho MUX scan
- Design Rule Checker:** Công cụ kiểm tra toàn bộ ràng buộc và quy tắc
- Result:** Kết quả kiểm tra: pass/fail, lỗi cụ thể, vị trí lỗi

## Tại sao cần dùng Highly-Compressed Pattern? Kiểm tra AC SCAN cần nhiều pattern hơn

- So với kiểm tra stuck-at (kiểm tra lỗi tĩnh), kiểm tra At-Speed (lỗi thời gian) cần từ 3 đến 5 lần nhiều pattern hơn.
- Việc tăng pattern sẽ tăng chi phí kiểm tra (Test cost ↑), do bộ lưu trữ pattern trên tester có giới hạn.

## Necessity for Highly-Compressed Pattern

- Need more pattern generated for AC SCAN test (3 to 5 times)
- Drastically Highly-Compressed Pattern is necessary to cope with large scale while maintaining the high test quality



## Kết luận:

Highly compressed pattern là giải pháp hiệu quả để xử lý kiểm tra các thiết kế lớn mà vẫn giữ chất lượng kiểm tra cao và chi phí thấp.

## Khái niệm cơ bản của Compression SCAN

### Vấn đề: Dữ liệu SCAN rất lớn (Huge Data)

- Khi thực hiện kiểm tra SCAN (đưa dữ liệu test vào các flip-flop), dữ liệu đầu vào và đầu ra có kích thước rất lớn.
- Nếu giữ nguyên, việc lưu trữ và xử lý các pattern này sẽ rất tốn tài nguyên và tăng chi phí kiểm tra.

### Giải pháp: Dùng Decompressor và Compactor

#### 1. Decompressor (Giải nén dữ liệu đầu vào)

- Tại input, chỉ cần **nạp một pattern nhỏ đã được nén** từ phần mềm vào chip.
- **Decompressor bên trong chip** sẽ mở rộng nó thành pattern đầy đủ cho hàng trăm (hàng ngàn) flip-flop trong scan chain.
- → **Giảm băng thông input** cần thiết, tiết kiệm thời gian và bộ nhớ tester.

▼ Ví dụ:

Pattern nạp vào: d0101xx1011x010x1x

Pattern sau khi giải nén: dài gấp nhiều lần, đủ để điều khiển tất cả các FFs

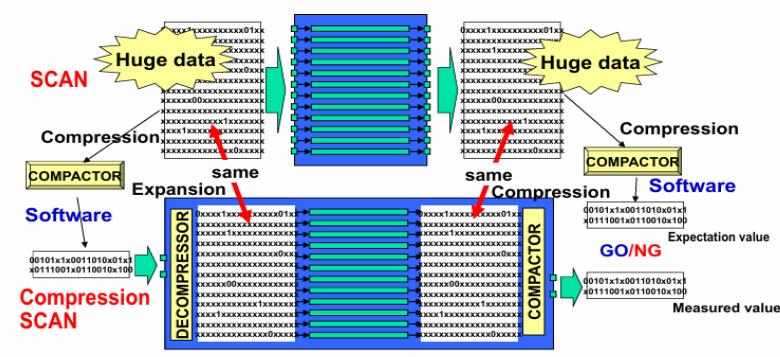
#### 2. Compactor (Nén dữ liệu đầu ra)

- Sau khi thực hiện **capture operation**, các giá trị tại các FFs được thu thập.
- Tuy nhiên, thay vì lấy toàn bộ dữ liệu lớn, ta dùng **compactor để nén chúng lại** thành một giá trị nhỏ hơn.
- Dữ liệu này sẽ được đưa ra ngoài để so sánh với **giá trị mong đợi (Expected Value)**.
- Nếu khớp → **GO**, nếu sai → **NG (No Good)**

## Ý nghĩa của Compression SCAN

- ✓ Giảm pattern size: Chỉ cần nạp và đọc dữ liệu nén
- ✓ Giảm thời gian test: Do giảm số lần giao tiếp input/output
- ✓ Giảm chi phí tester: Tester không cần lưu trữ toàn bộ pattern lớn
- ✓ Giữ nguyên chất lượng kiểm tra: Nhờ mở rộng và nén đúng logic

## Basic Concept of Compression SCAN



Data compression/expansion circuits (DECOMPRESSOR/COMPACTOR) in a LSI for SCAN input/output can reduce pattern data size

**Tóm lại:** Compression SCAN dùng decompressor và compactor để xử lý dữ liệu scan input/output ngay bên trong chip: Giảm đáng kể dữ liệu test, Giảm chi phí và thời gian, Vẫn đảm bảo khả năng phát hiện lỗi

## Built-In Self-Test (BIST)

### 1. Vấn đề với Scan Test truyền thống

#### Cách hoạt động:

- Các **test pattern** (dữ liệu kiểm tra) và **expected values** (giá trị mong đợi) được tạo từ bên ngoài.
- Chúng được tải vào chip thông qua **tester** (thiết bị kiểm tra – thường là ATE: Automatic Test Equipment).
- Sau khi thực hiện scan, kết quả được **xuất ra**, gửi lại về tester để **so sánh**.

#### Nhược điểm:

- Test pattern số lượng lớn** (đặc biệt với thiết kế phức tạp): Tốn thời gian truyền dữ liệu, Chi phí lưu trữ pattern trên tester cao
- Tester rất đắt tiền** (ATE có thể hàng triệu đô la)
- Bị phụ thuộc vào thiết bị kiểm tra** → không phù hợp để test ở giai đoạn hậu sản xuất hoặc ngoài nhà máy.

### 2. Logic BIST là gì? (Chip tự kiểm tra chính nó!)

Logic Built-In Self-Test là phương pháp **nhúng trực tiếp mạch kiểm tra vào trong chip** để:

- Tự tạo test pattern
- Tự đánh giá kết quả
- Chỉ cần **một máy đơn giản** để khởi động và đọc kết quả pass/fail

### 3. Cấu trúc chính của Logic BIST

#### a. Pattern Generator (Bộ tạo mẫu kiểm tra)

- Thường là **LFSR – Linear Feedback Shift Register**
- Tạo ra các **dãy bit giả ngẫu nhiên** → được dùng như tín hiệu kiểm tra
- Ưu điểm: Nhỏ gọn, không cần lưu trữ từ ngoài, Độ phủ lỗi cao nếu dùng đúng cách

### Built-In Self-Test (BIST)

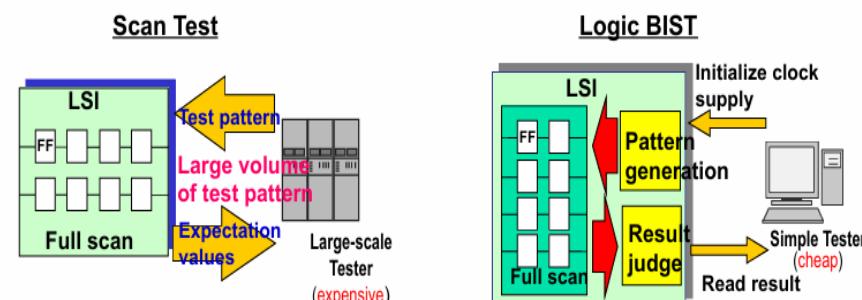
#### b. Full Scan Chain

- Tất cả các flip-flop trong mạch được **liên kết thành chuỗi (scan chain)**
- Dữ liệu từ LFSR được **nạp vào chuỗi này** → kích hoạt mạch logic
- Sau đó, dữ liệu **capture lại giá trị phản hồi**

- Pattern generation circuits and pass/fail judge circuits are inserted into an LSI
- Can test an LSI using pseudo random number patterns
- Can judge pass/fail after a test completed

#### c. Result Judge (Đánh giá kết quả)

- Sử dụng mạch như **MISR – Multiple Input Signature Register**
- MISR thu thập và nén đầu ra scan thành một **chữ ký duy nhất**
- So sánh chữ ký với giá trị kỳ vọng (stored inside chip hoặc tính toán trước)
- Chỉ xuất ra: **Pass / Fail**



### 4. So sánh Scan Test vs Logic BIST

Scan Test	Logic BIST
Cần thiết bị kiểm tra bên ngoài (Large-scale tester – đắt tiền)	Không cần nhiều thiết bị kiểm tra bên ngoài (Simple tester – rẻ tiền)
Test pattern và expectation values được cấp từ bên ngoài	Test pattern và kết quả được tạo và đánh giá nội bộ trong chip
Full scan kết hợp với volume lớn của test pattern	Full scan kết hợp với pattern tự sinh từ mạch logic
Dữ liệu test được tải từ/tới tester qua IO	Dữ liệu test và so sánh xử lý nội bộ – chỉ cần đưa ra kết quả pass/fail

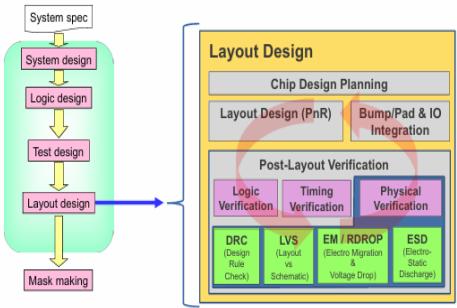
#### Ưu điểm của Logic BIST

- Tự kiểm tra: Không cần thiết bị ngoài – chỉ cần nguồn điện
- Tiết kiệm chi phí: Không cần tester phức tạp, tiết kiệm hàng triệu đô
- Có thể test lại sau khi giao sản phẩm: Phù hợp để kiểm tra định kỳ hoặc kiểm tra tại chỗ
- Dễ tích hợp: Có thể tích hợp trong mọi LSI, SoC hiện đại

# CHAP6: LAYOUT DESIGN

## 6.1. Overview of Layout Design.

LSI Design Flow



### 1. Tổng quan về LSI Design Flow

Quy trình thiết kế LSI bao gồm 6 bước chính:

4. **System spec** – Xác định yêu cầu hệ thống (chức năng, hiệu năng, tiêu thụ điện, kích thước chip, v.v.)
5. **System design** – Thiết kế kiến trúc hệ thống (chia nhỏ thành các khối chức năng)
6. **Logic design** – Thiết kế logic (viết HDL, tạo mô hình logic của mạch)
7. **Test design** – Thiết kế kiểm thử (thêm mạch kiểm tra như scan chain để test chip sau này)
8. **Layout design** – Thiết kế layout (bố trí thực tế các thành phần phần cứng trên silicon)
9. **Mask making** – Tạo mask cho quy trình sản xuất (photolithography)

## 2. Layout Design - Giai đoạn thiết kế layout:

Layout Design là giai đoạn chuyển đổi logic design thành bố trí vật lý trên silicon. Gồm 3 phần chính:

- Chip Design Planning:** Lập kế hoạch cho toàn bộ bố trí chip: kích thước die, phân vùng chức năng, vị trí nguồn, vị trí I/O, hướng clock...
- Layout Design (PnR - Place and Route)**
  - Place (Đặt):** Xác định vị trí các khối logic (standard cells) trên chip.
  - Route (Định tuyến):** Nối các khối logic bằng dây dẫn để truyền tín hiệu.
- Bump/Pad & IO Integration**
  - Tích hợp các cổng đầu vào/ra (pad) và bump (cho flip-chip) để giao tiếp với môi trường bên ngoài chip.

Nếu có lỗi trong verification, quá trình quay lại bước trước (layout chỉnh sửa lại hoặc thậm chí cả logic design nếu cần).

**Kết luận:** Giai đoạn Layout Design là bước cực kỳ quan trọng trong quy trình thiết kế chip, đảm bảo rằng bản thiết kế logic được chuyển thành bố trí vật lý chính xác, hiệu quả và đáng tin cậy để đưa vào sản xuất.

### Đầu vào của Layout Design

#### a. Verilog Gate-level Netlist

- Là kết quả từ giai đoạn thiết kế logic (Logic Design).
- Gồm các cổng logic, module và mạch kiểm thử DFT (Design for Testability).

#### b. Library sử dụng trong thiết kế

- IP libraries:** Các khối chức năng có sẵn như CPU, RAM, USB...
- CELL libraries:** Tập hợp các cell tiêu chuẩn (standard cells).
- Macro libraries:** Các khối đặc biệt như SRAM, PLL...

#### c. Design Specifications

Thông số kỹ thuật đầu vào rất quan trọng:

- Công nghệ chế tạo: 45nm, 28nm...; Kích thước chip, điều kiện hoạt động; Tần số hoạt động: cho CPU, GPU, v.v; Nguồn cấp và tiêu thụ điện năng; Yêu cầu độ tin cậy

### Kết quả đầu ra

- Tất cả thông tin layout, kết quả kiểm tra được lưu vào **Chip Design Database**.
- Nếu đạt yêu cầu → **Chuyển sang quá trình sản xuất (Manufacturing Processes)**.

## 6.2. Cell and Library.

**Cell là gì?** Một Cell là đơn vị chức năng (functional unit) hoặc khối macro được sử dụng để xây dựng các khối logic phức tạp hơn trong thiết kế vi mạch.

### Phân loại Cell:

- Standard Cells (Cell chuẩn):** Các cổng logic cơ bản và phần tử lưu trữ: Inverter, NAND, NOR, Flip-Flop, Latches, Buffers...
- Macro Cells (Khối chức năng lớn hơn):** Các khối mạch có chức năng cụ thể: ADC (Analog to Digital Converter), DAC, USB, PCI, PLL (Clock generator), THS (Thermal Sensor)...

### 3. Post-Layout Verification – Kiểm tra sau thiết kế layout

Mục đích: Đảm bảo layout không vi phạm quy tắc công nghệ và hoạt động đúng như logic ban đầu. Gồm 3 loại kiểm tra:

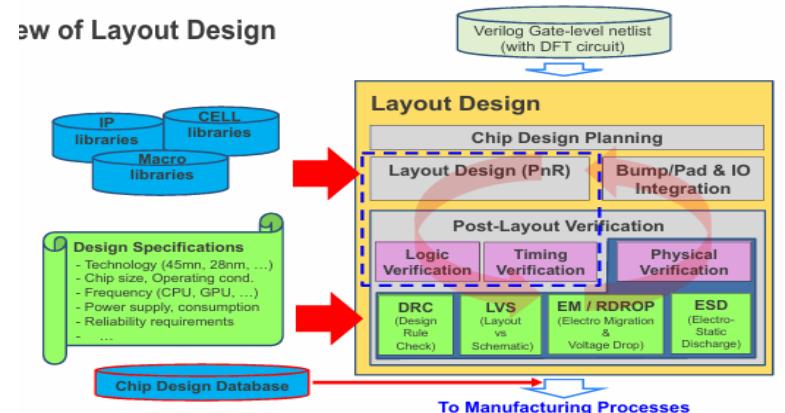
#### a. Logic & Timing Verification

- Logic Verification:** Kiểm tra chức năng logic sau khi place & route.
- Timing Verification:** Đảm bảo tín hiệu đến đúng thời gian (setup/hold time).

#### b. Physical Verification (Kiểm tra các yếu tố vật lý):

- DRC (Design Rule Check):** Kiểm tra xem layout có tuân thủ quy tắc thiết kế của nhà sản xuất (ví dụ: khoảng cách tối thiểu giữa các dây, kích thước vias...)
- LVS (Layout vs Schematic):** So sánh layout với sơ đồ nguyên lý (schematic) để đảm bảo không sai lệch chức năng.
- EM/RDROP:** Kiểm tra vấn đề về dòng điện: **EM (Electromigration)** (Dòng lớn gây hư dây) & **Volt drop** (Giảm điện áp do điện trở đường dây)
- ESD (ElectroStatic Discharge)** Kiểm tra khả năng bảo vệ chip khỏi tĩnh điện gây hỏng mạch.

### Flow of Layout Design





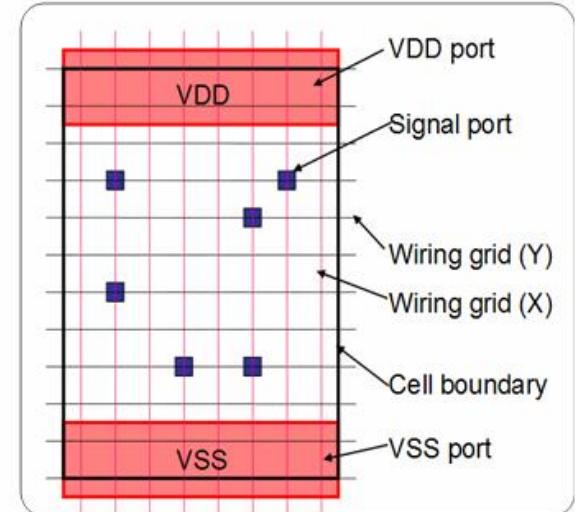
## Cấu trúc Cell chuẩn (Standard Cell) – Phương pháp Cell-based

### ◆ Cấu trúc tổng quát:

- Chiều rộng của cell: Có thể thay đổi (là bội số nguyên của lưới dây X).
- Chiều cao của cell: Cố định theo công nghệ (là bội số nguyên của lưới dây Y).
- Cổng tín hiệu (Signal Port): Nằm trên lưới (on-grid) để dễ định tuyến (route).
- Cổng nguồn (Power Supply Ports): VDD: nằm ở trên cùng, VSS: nằm ở dưới cùng. Vị trí nguồn là cố định với mọi cell trong thư viện.

### Ghi chú thêm:

- Metal wiring (dây kim loại) có thể chạy đè lên cell (over cell), không ảnh hưởng đến chức năng.
- Cấu trúc này giúp quá trình Place & Route (P&R) tự động được dễ dàng và hiệu quả.



## Cell-based Method – Phương pháp dựa trên cell chuẩn

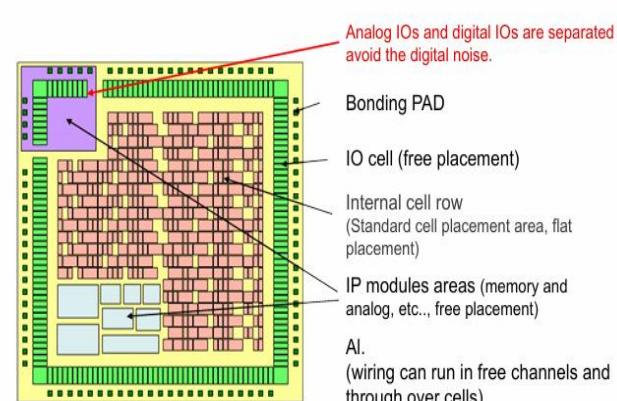
### Tính chất chính:

- Các cell được đặt tự do trong chip.
- Áp dụng tự động Place & Route (P&R) để bố trí và kết nối.
- Sử dụng kim loại đa lớp (multi-layer metal) để định tuyến dây dẫn.

### Các thành phần trong layout:

- Analog IOs và Digital IOs: Được tách biệt để tránh nhiễu số.
- Bonding PAD: Các vùng kết nối ra ngoài chip.
- ◆ IO Cells: Vị trí đặt tự do quanh viền chip.
- Internal Cell Row: Khu vực chứa các standard cells, bố trí phẳng.
- IP Modules Areas: Bao gồm bộ nhớ, mạch analog, v.v. (có thể đặt tự do).
- Ứng dụng AI (wiring area): Dây dẫn có thể đi qua các cell và vùng trống.

Cells can be arbitrarily placed - Automatic P&R is applied - Multi-layer metals are used.



### Ghi chú thêm:

- Cell-based method là nền tảng của **thiết kế VLSI hiện đại**.
- Phù hợp với thiết kế có số lượng lớn cổng logic, tối ưu hóa diện tích chip và hiệu suất định tuyến.

## Hierarchical Cell-based Method

Là phương pháp thiết kế layout chip theo từng khối (hierarchical), nơi mỗi khối logic được thiết kế riêng (PnR riêng), sau đó gắn vào bản thiết kế tổng thể.

### Đặc điểm chính:

- Thiết kế từng phần (khối logic A, B, ...) một cách độc lập
- Tự động P&R (Place & Route – sắp xếp vị trí và đi dây)
- Sử dụng cell tiêu chuẩn (standard cells)
- Có thể tái sử dụng khối đã thiết kế
- Giúp quản lý thiết kế dễ hơn, giảm thời gian và công sức

### Ví dụ cho slide trên

#### ● Hierarchical logic B

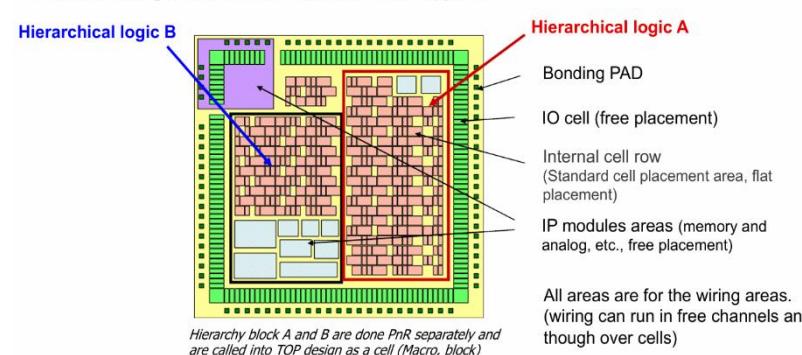
- Đây là **khối logic B**, một phần nhỏ của chip.
- Được thiết kế riêng biệt trước (đặt vị trí, nối dây, vẽ sơ đồ mạch).
- Sau khi xong, mình gắn nó vào bản thiết kế tổng thể.
- Giống như **lắp ráp nguyên một khu phố** rồi chuyển cả khu phố vào bản đồ thành phố lớn!

#### ◆ Dòng chú thích bên dưới:

“Hierarchy block A and B are done PnR separately and are called into TOP design as a cell (Macro, block)”

- “PnR” = **Place and Route**, nghĩa là sắp xếp vị trí và nối dây.
- Khối A và B **được thiết kế (PnR) riêng**, sau đó **đưa vào thiết kế lớn** như một khối hoàn chỉnh (macro).
- Việc này giúp **quản lý thiết kế dễ hơn**, giống như chia bài toán lớn thành nhiều bài toán nhỏ rồi ráp lại.

The standard cell placement area is done layout hierarchically depending on the hierarchical logical structure of the gate level netlist - Automatic P&R is applied.



#### ● Hierarchical logic A

- Tương tự như logic B, đây là **một khối logic khác (A)**.
- Cũng được thiết kế độc lập, sau đó **đặt vào bản thiết kế tổng**.
- Nhờ thiết kế kiểu "chia nhỏ" này, mình có thể làm việc **nhanh hơn, rõ ràng hơn, và dễ sửa lỗi**.

## Internal Cell Row

- Là hàng các cell tiêu chuẩn – như dãy nhà trong thành phố.
- Được sắp xếp phẳng, đều đặn, dễ cho việc nối dây (wiring).
- Đây là nơi đặt các cell logic cơ bản như AND, OR, NOT,...

## IP Modules Areas

- IP = Intellectual Property, là những **khối chức năng đặc biệt** như: Bộ nhớ, Mạch xử lý âm thanh, Giao tiếp
- Các khối này **được đặt tự do** trong chip, không cần theo hàng.

## IO Cell (free placement)

- IO = Input/Output, là **nơi giao tiếp giữa chip và thế giới bên ngoài**.
- Được đặt xung quanh mép ngoài chip.
- Có thể **đặt tự do** (free placement), tùy theo hướng kết nối.

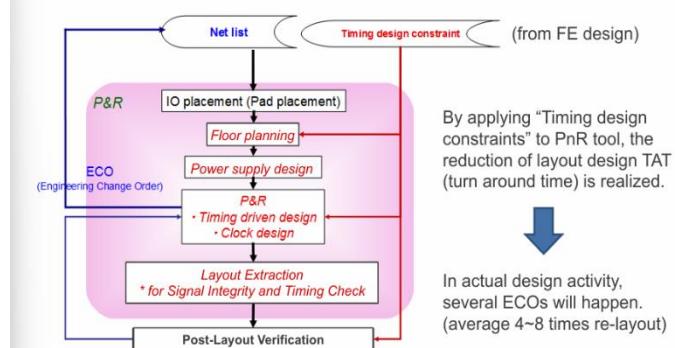
## Bonding PAD

- Là **các điểm tiếp xúc kim loại** để kết nối con chip với phần cứng bên ngoài (như chân chip, bo mạch...). Nằm ở rìa con chip, giống như **cửa ra vào của thành phố** vậy!

## All areas are for wiring

- Nghĩa là toàn bộ các khu vực (kể cả trên nhà, ngoài nhà) đều có thể chạy dây được.
- Chip có thể dùng **nhiều lớp kim loại (multi-layer)** để **nối dây chéo nhau mà không bị đụng nhau**

## (Placement & Routing) General Flow



## 6.4. Chip Design Planning. (NO TEST)

## 6.5. Block and TOP Level Layout Design.

Quy trình PnR tổng quát:

### 1. Input:

- Netlist: Từ thiết kế Front-End, mô tả kết nối giữa các cổng logic.**
- Timing design constraint: Các ràng buộc về thời gian (từ Front-End, ví dụ như tCLK, skew, setup/hold time,...).**

### 2. Các bước chính trong khối màu hồng (PnR block):

#### IO Placement (Pad placement)

- Đặt các chân I/O ở viền die (gắn pad cell).
- Cực kỳ quan trọng với floorplanning và routing sau này.

#### Floor Planning

- Xác định vùng chức năng, vị trí của khối logic, RAM, IO,...
- Phân bổ diện tích hợp lý → ảnh hưởng lớn đến hiệu suất và routing.

#### Power Supply Design

- Thiết kế mạng cấp nguồn (VDD, VSS) → đảm bảo điện áp ổn định.
- Gồm: power ring, power stripe, power grid.

#### PnR (Placement & Routing)

- Timing driven design: Đặt và đi dây các cell sao cho đáp ứng thời gian.
- Clock design: Thiết kế mạng phân phối clock (H-tree hoặc grid), tối ưu skew.

#### Layout Extraction

- Trích xuất các thông tin layout (parasitics – điện trở, điện dung).
- Dùng để check timing và độ toàn vẹn tín hiệu (signal integrity).

#### Post-Layout Verification

- Kiểm tra lại toàn bộ thiết kế sau khi đã có layout:
- DRC: Kiểm tra vi phạm quy tắc thiết kế.
- LVS: So sánh netlist layout vs netlist sơ đồ nguyên lý.
- STA: Static Timing Analysis – phân tích thời gian.

#### ECO (Engineering Change Order):

- Là các vòng chỉnh sửa layout khi có lỗi hoặc yêu cầu thay đổi.
- Trong thực tế, quá trình thiết kế có thể phải ECO 4~8 lần trung bình.

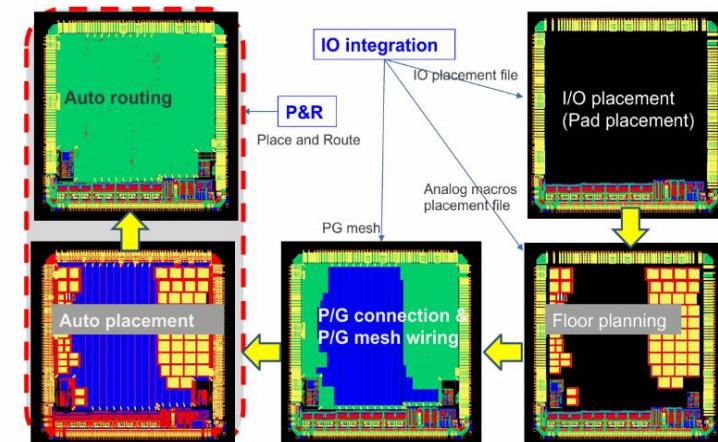
## Ý nghĩa của Timing Constraint

- Giúp công cụ tự động PnR tối ưu theo thời gian.
- Rút ngắn TAT (Turn-Around Time) → thiết kế nhanh và hiệu quả hơn

## Kiến thức cần nhớ

- Input: Netlist, Timing constraint
- Bước đầu: IO placement, Floorplanning
- PnR chính: Placement, Routing, Clock Design
- Extraction: Dùng parasitic để kiểm tra timing
- Verification: DRC, LVS, STA
- ECO: Phải lặp nhiều lần trong thực tế
- Lợi ích: Timing constraints giúp giảm TAT(turn around time)

## PnR General Flow (cont'd)



## 1. IO Placement (Pad placement) (bước đầu tiên bên phải trên)

- Đặt các pad I/O quanh viền die.
- Kết quả sinh ra IO placement file (đầu vào cho bước tiếp theo).
- Tương ứng với "IO placement" trong slide trước.

## 2. Floor Planning (bên phải dưới)

- Phân chia vùng cho các block logic, analog macros, hard IP,...
- Mục tiêu: tối ưu không gian, giảm wire length, giảm delay.
- Sinh ra Analog macros placement file.
- Tương ứng với bước "Floor Planning" của slide trước.

## 3. P/G Mesh Wiring (Power/Ground)

- Tạo lưới cấp nguồn (P/G mesh).
- Đảm bảo toàn bộ chip được cung cấp điện áp ổn định.
- Quan trọng cho reliability và timing.
- Gắn liền với phần Power Supply Design trong PnR.

## 4. Auto Placement

- Cell logic (standard cells) được tự động sắp xếp.
- Mục tiêu: tối ưu mật độ, giảm diện tích và cải thiện timing.

## 5. Auto Routing

- Tự động nối dây giữa các cell.
- Dùng thuật toán để tối ưu đường đi (routing congestion, via, delay...).
- Kết thúc bước này là có layout hoàn chỉnh.

## Floor Planning

**Floor Planning là gì?** Floor Planning là **bước đầu tiên trong thiết kế chip** dùng để phân bổ không gian và xác định vị trí các thành phần của chip trên diện tích silicon. Đây là bước quan trọng để đảm bảo hiệu suất, độ tin cậy và khả năng sản xuất của chip. **Mục tiêu chính của Floor Planning:**

- Quy hoạch không gian:** Xác định kích thước và hình dạng của các block.
- Đảm bảo kết nối hợp lý:** Giảm độ trễ tín hiệu và tối ưu hóa routing.
- Cân đối diện tích:** Đảm bảo các thành phần không quá gần nhau (gây nhiễu) hoặc quá xa nhau (tăng độ trễ và chi phí routing).

**Top Floor Plan là gì?** Top Floor Plan là quá trình lập kế hoạch và phân bổ vị trí cho các thành phần chính trên toàn bộ chip ở cấp độ cao nhất (TOP). **Mục tiêu của Top Floor Plan:**

- Phân bổ không gian:**
  - Định vị các thành phần chính như:
    - Hierarchical Blocks:** Các block con.
    - Hard Macros:** Thành phần cố định (IP cores, memories).
    - I/O Cells:** Các cell kết nối đầu vào/dầu ra.
  - Dự trữ không gian còn lại để sử dụng cho logic cells hoặc các khối khác.
- Tối ưu hóa kết nối:** Đảm bảo các đường tín hiệu giữa các block tối ưu để giảm độ trễ và cải thiện hiệu suất.

## Các yếu tố chính trong Top Floor Plan:

- Kích thước và hình dạng block:** Được xác định dựa trên yêu cầu thiết kế.
- Tối ưu không gian:** Phân bổ diện tích hợp lý giữa các block và khu vực dự trữ.
- Block Interface Models:** Được sử dụng để đảm bảo các block giao tiếp chính xác.

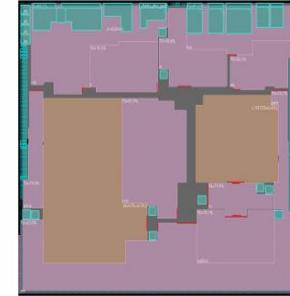
## Quy trình thiết kế trong Hierarchical PnR (Place and Route):

- Chip TOP Floor Planning:** Quyết định kích thước (diện tích) và hình dạng của từng block con (child block).
- Quy trình thiết kế lặp lại cho từng block:** Mỗi block trải qua FP → Place → Route → Timing and Signal Integrity Check.
- Block Interface Model (Mô hình giao diện block):** Được tạo ra cho mỗi block để xác định cách các block giao tiếp; Cập nhật kích thước và hình dạng giữa block và thiết kế TOP.
- Thiết kế TOP sử dụng Block Interface Model:** Chu trình thiết kế TOP cũng áp dụng FP → Place → Route → Timing and Signal Integrity Check.

## Floor Planning

In case of Hierarchical PnR design:

- Chip TOP Floor Planning: decide size (area) and shape for each child block.
- Each block is applied the whole design cycle (FP → place → route → timing and signal integrity check).
- Block interface model is made for each block. Block size and shape is updated to/from TOP design.
- TOP design will use block interface model for its design cycle (FP → Place → Route → Timing and Signal integrity check).



### Top Floor Plan:

- Hierarchical blocks, Hard macros, and I/O cells are placed.
- The remained area (in gray) is reserved for logic cells on TOP level.

## Kiến thức cốt lõi cần nhớ

- Floor Planning:**
  - Là bước đầu tiên trong thiết kế chip, tập trung vào việc phân bổ không gian và xác định vị trí các thành phần.
  - Chu trình cơ bản: Floor Planning → Place → Route → Timing and Signal Integrity Check.
- Top Floor Plan:**
  - Phân bổ không gian và định vị các thành phần chính trên toàn bộ chip.
  - Hỗ trợ kết nối giữa các block, giảm độ trễ tín hiệu và tối ưu hóa hiệu suất.
- Block Interface Model:**
  - Là công cụ kết nối giữa thiết kế từng block và thiết kế TOP.
  - Được sử dụng xuyên suốt trong quá trình thiết kế để cập nhật kích thước và vị trí.

# Power Supply Design

## 1. Mục tiêu của Power Supply Design: Thiết kế power routing nhằm:

- Đảm bảo tuân thủ các quy tắc thiết kế (design rules) như:
  - Giảm thiểu voltage drop.
  - Giảm thiểu electro-migration (hiện tượng dòng điện làm hư hại dây dẫn).
- Cung cấp đủ dòng điện cho từng cell trong chip.
- Tối ưu hóa diện tích chip, tránh lãng phí không gian.

"Power supply design" targets to establish power routing to keep specifications of design rule (for voltage drop, electro-migration, etc..) and to supply sufficient current to each cell, considering to minimize the chip size.

### <Design approach>

#### 1) Manual design

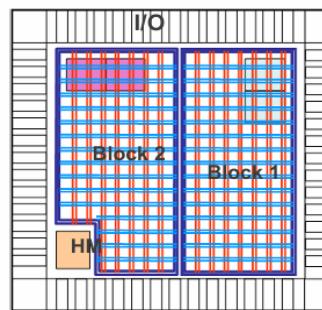
The designer decides **number of wire straps** and their **width**, then uses layout editor and the P&R tool to draw PG mesh **manually**.

#### 2) Semi-automatic design

Use Angel@Ring (Renesas in-house tool) and pre-determined power routing cells.

#### 3) Automatic power design tool

Developed and implemented to EDA tools  
**EDA tools will decide** PG mesh pattern, number of wire straps, metal width, ... and do the PG routing **automatically**



## 2. Phương pháp thiết kế (Design Approach) Có 3 phương pháp chính để thực hiện Power Supply Design:

### 2.1. Manual Design (Thiết kế thủ công):

- Quy trình:
  - Người thiết kế quyết định: Số lượng dây dẫn nguồn (wire straps). Chiều rộng của dây dẫn.
  - Sau đó, sử dụng layout editor và công cụ Place & Route (P&R) để tạo lưới nguồn (PG mesh) một cách thủ công.
- Ưu điểm: Toàn quyền kiểm soát thiết kế.
- Nhược điểm: Tốn thời gian, dễ xảy ra sai sót.

### 2.2. Semi-Automatic Design (Thiết kế bán tự động):

- Quy trình: Sử dụng các công cụ nội bộ (như Angel@Ring của Renesas) và các cell routing nguồn đã được xác định trước.
- Ưu điểm: Tiết kiệm thời gian hơn thiết kế thủ công, Giảm thiểu sai sót.
- Nhược điểm: Cần các công cụ hỗ trợ, không linh hoạt bằng thiết kế thủ công.

### 2.3. Automatic Power Design Tool (Thiết kế tự động):

- Quy trình: Sử dụng các công cụ EDA (Electronic Design Automation) để:
  - Quyết định mẫu PG mesh (PG mesh pattern).
  - Xác định số lượng dây dẫn, chiều rộng dây, kim loại sử dụng,...
  - Thực hiện Power Routing tự động.
- Ưu điểm: Tự động hóa cao, giảm thời gian và công sức, Tối ưu hóa thiết kế dựa trên các thuật toán và công cụ hiện đại.
- Nhược điểm: Phụ thuộc vào công cụ EDA, Cần chi phí cao để triển khai công cụ.

## 3. Hình minh họa thiết kế Power Supply:

- PG Mesh:** Mạng lưới dây dẫn nguồn được phân bố đều trên chip để cung cấp nguồn cho các block.
- Cấu trúc chip:**
  - Các block (Block 1, Block 2) được bao quanh bởi lưới nguồn.
  - I/O Cells:** Được đặt ở rìa chip để kết nối đầu vào/đầu ra.
  - Hard Macros (HM):** Thành phần cố định có vùng nguồn được thiết kế riêng biệt.

## 4. Kiến thức cốt lõi cần nhớ:

- Mục tiêu chính của Power Supply Design:** Đảm bảo cung cấp đủ nguồn điện cho các cell, giảm voltage drop và electro-migration
- Ba phương pháp thiết kế Power Supply:**
  - Thủ công:** Quyết định chi tiết (số lượng, chiều rộng dây) và vẽ lưới nguồn bằng tay.
  - Bán tự động:** Sử dụng công cụ hỗ trợ và các cell định sẵn.
  - Tự động:** Dựa trên công cụ EDA, tối ưu hóa toàn bộ quá trình.
- PG Mesh:** Là mạng lưới dây dẫn nguồn bao phủ chip để cung cấp nguồn điện đồng đều cho các block.

## PnR – Placement

**Placement là gì?** Placement là bước trong quy trình Place and Route (PnR), dùng để xác định vị trí của các thành phần trong chip, bao gồm các block, macro, cổng kết nối (pins/ports), và lưới nguồn (Power/Ground grid). Đây là bước tiếp theo sau Floor Planning.

### TOP Placement (Cấp độ toàn chip):

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li><b>Các thành phần được đặt:</b><ul style="list-style-type: none"><li>I/O cells (các cell kết nối đầu vào/đầu ra).</li><li>Analog macros (các thành phần analog).</li><li>RAM/ROM (các thành phần bộ nhớ).</li></ul></li></ul> | <ul style="list-style-type: none"><li><b>Các quyết định được thực hiện:</b><ul style="list-style-type: none"><li>Hình dạng block: Quyết định kích thước và hình dạng của từng block.</li><li>Lưới nguồn (Power/Ground grid): Thiết kế lưới nguồn để cung cấp năng lượng cho chip.</li><li>Vị trí các pin: Quy hoạch vị trí các pin cho từng block.</li></ul></li></ul> |
|---|--|

# PnR - Placement

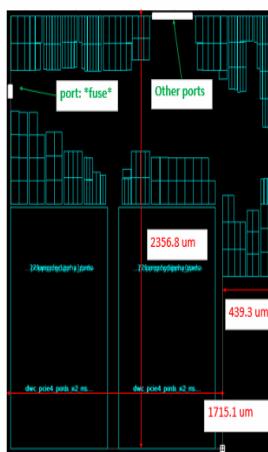
## TOP Placement

- IO, Analog macros, and RAM/ROM are placed
- Block shapes are decided.
- Power/Ground grid is designed.
- Position to place pins for each block is planned.



## Block placement

- RAM/ROM, Analog macros are placed
- Module ports are placed
- Power/Ground grid is implemented.



## Block Placement (Cấp độ block):

- Các thành phần được đặt:
  - RAM/ROM và analog macros.
  - Module ports: Các cổng kết nối giữa các module và block.
- Các yếu tố được triển khai:
  - Lưới nguồn (Power/Ground grid): Được triển khai để đảm bảo dòng điện phân phối đều cho block.

### Hình ảnh 1 (TOP Placement - bên trái):

#### • Minh họa việc:

- Sắp xếp các block lớn trên toàn chip.
- Thiết kế lưới nguồn (Power/Ground Grid).
- Đặt các pin và kết nối đầu vào/đầu ra (I/O).

### Hình ảnh 2 (Block Placement - bên phải):

#### • Minh họa việc:

- Đặt các module nhỏ hơn (RAM/ROM, analog macros) bên trong một block cụ thể.
- Đánh dấu vị trí của các cổng kết nối (ports) và thiết lập các đường dẫn (routing).

## Các bước quan trọng trong Placement

1. Xác định vị trí các thành phần chính: Đặt các block, macros, và RAM/ROM dựa trên yêu cầu hiệu suất và diện tích.
2. Thiết kế lưới nguồn (Power/Ground Grid): Đảm bảo lưới nguồn được thiết kế để cung cấp đủ dòng điện cho toàn bộ chip.
3. Quy hoạch vị trí pin và cổng kết nối: Đặt các pin và module ports để tối ưu hóa kết nối giữa các block và giảm thiểu độ trễ tín hiệu.

## Kiến thức cốt lõi cần nhớ

- TOP Placement:
  - Đặt các thành phần lớn (I/O, RAM/ROM, Analog Macros) ở cấp độ toàn chip.
  - Thiết kế lưới nguồn (Power/Ground grid) và xác định vị trí pin.
- Block Placement:
  - Đặt các thành phần nhỏ hơn (RAM/ROM, Analog Macros) trong từng block.
  - Đặt module ports và triển khai lưới nguồn ở cấp độ block.

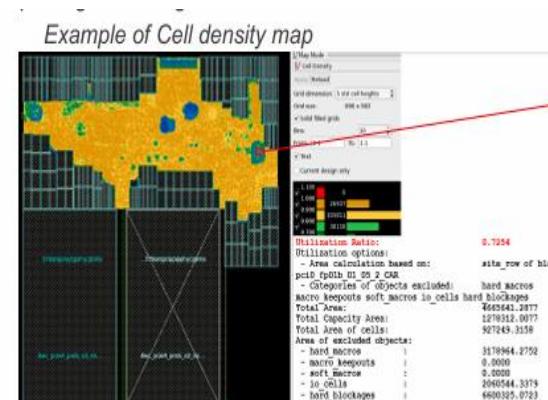
Placement là bước quan trọng để đảm bảo các thành phần trong chip được sắp xếp hợp lý, tối ưu hóa hiệu suất và kết nối tín hiệu.

## PnR - Placement Optimization (Tối ưu hóa vị trí - trong quá trình Place and Route)

Điều kiện tiên quyết (Prerequisite): Cần thiết kế sau khi hoàn thành floor-planning.

Trong một thiết kế điều khiển theo thời gian (Timing-driven design), các bước tối ưu vị trí thực hiện những công việc sau:

1. Tự động đặt các cell logic chuẩn (standard-cells) bằng công cụ EDA (Electronic Design Automation).
2. Xây dựng cây clock ảo (virtual ideal-clock-tree) để tối ưu hóa thời gian.
3. Định tuyến kết nối ảo (virtual routing) giúp đánh giá sơ bộ khả năng kết nối giữa các cell.
4. Tối ưu logic và thay đổi vị trí các cell nhằm:
  - Giảm tắc nghẽn định tuyến,
  - Giảm vi phạm về thời gian (timing violations),
  - Tối ưu hóa điện năng tiêu thụ (power optimization).
5. Thực hiện kiểm tra quy tắc thiết kế (DRC - Design Rule Check) và kiểm tra thời gian (timing check):
  - Tự động phát hiện và xử lý các vi phạm (violation), ví dụ: di chuyển hoặc thay thế các cell.



## Bản đồ mật độ cell (Cell density map):

- Cho thấy sự phân bố mật độ cell trên toàn thiết kế.
- Các vùng màu đỏ thể hiện nơi có mật độ cell cao (dễ gây tắc nghẽn).
- Các công cụ EDA sẽ điều chỉnh để tránh vi phạm và tối ưu layout.

## PnR - Clock Design

Trong quá trình thiết kế PnR, **Clock Design** là một bước cực kỳ quan trọng vì đồng hồ (clock) ảnh hưởng trực tiếp đến **hiệu suất, độ chính xác thời gian, và độ ổn định** của toàn mạch số.

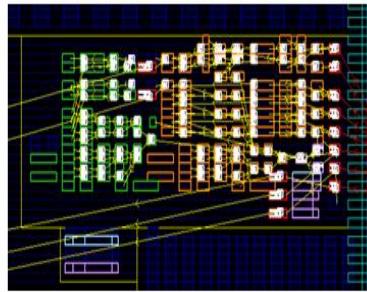
## 1. Phân tích cấu trúc clock (Analyze clock structure)

Mục tiêu: đảm bảo tín hiệu clock đến đúng nơi, đúng thời điểm (đồng bộ toàn hệ thống).

- Bắt đầu từ **Clock Source (CPGM)**.
- Clock được phân phối qua **Clock Distribution Module** gồm các phần tử như:
  - Bộ đệm (buffers),
  - Cổng logic điều khiển,
  - Các điểm phân phối tín hiệu clock tới các flip-flop trong toàn mạch.



Manually place clock cells of clock distribution modules



Insert main clock buffers of the clock tree



## 2. Đặt thủ công các cell clock của module phân phối clock (Manually place clock cells of clock distribution modules)

- Các cell đặc biệt dùng trong clock (ví dụ: clock gates, clock mux, v.v.) cần được **đặt thủ công** để đảm bảo độ chính xác.
- Việc đặt thủ công này giúp kiểm soát tốt vị trí của các phần tử nhạy cảm với thời gian trễ (delay).

## 3. Chèn các bộ đệm chính trong cây clock (Insert main clock buffers of the clock tree)

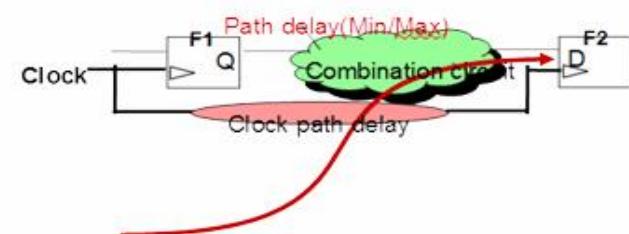
- Tạo **Clock Tree** – một cấu trúc phân phối clock dạng cây để tín hiệu đồng hồ được phân phối đều đến tất cả các điểm trong mạch.
- Clock buffers** được chèn vào tại các điểm thích hợp để đảm bảo:
  - Độ trễ (skew) giữa các nhánh là nhỏ nhất,
  - Độ trễ tổng thể của tín hiệu clock phù hợp với yêu cầu timing.

### PnR - Timing Driven Design

**Timing-driven design** là phương pháp thiết kế layout trong đó công cụ P&R **xác định vị trí và định tuyến các cell logic** dựa trên các **ràng buộc thời gian (timing constraints)** đã được xác định từ giai đoạn thiết kế logic.

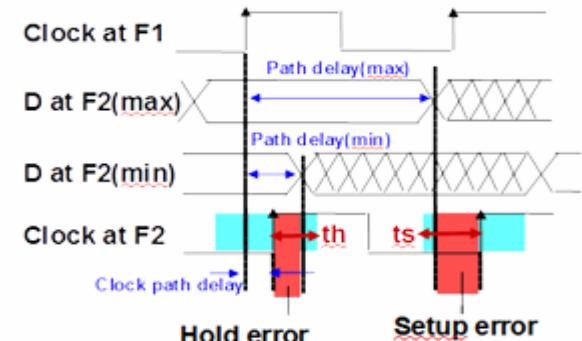
**Timing Constraint** là gì? Là tập hợp các điều kiện thời gian bao gồm:

- Path delay: Độ trễ của đường dữ liệu (data path).
- Setup time: Thời gian yêu cầu để dữ liệu ổn định trước khi xung clock tới cạnh tác động (active edge).
- Hold time: Dữ liệu phải duy trì ổn định sau khi xung clock đến cạnh tác động.



Các thông số này áp dụng giữa:

- Cổng clock và
- Cổng dữ liệu (data port) của các flip-flop trong mạch.



### Ràng buộc cụ thể:

- Setup Time Constraint:**
  - Dữ liệu **phải đến trước** thời điểm cạnh clock tới, với độ trễ nhất định.
  - Data arrival time  $\leq$  Clock arrival time - Setup time (ts)
- Hold Time Constraint:**
  - Dữ liệu **không được thay đổi** quá sớm sau cạnh clock.
  - Data arrival time  $\geq$  Clock arrival time + Hold time (th)

Thiết kế theo **timing-driven** giúp công cụ EDA tự động bố trí logic và định tuyến dây sao cho: **Đảm bảo không vi phạm timing, Tối ưu hóa hiệu suất** hệ thống.

### Cách xử lý khi vi phạm ràng buộc thời gian (timing violation)

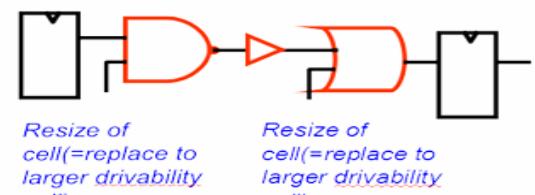
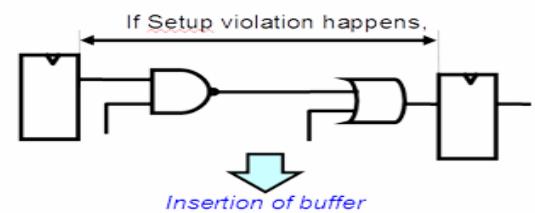
#### 1. Nếu xảy ra vi phạm Setup (Setup violation):

→ Dữ liệu đến quá trễ, không kịp trước cạnh clock.

**Giải pháp:**

- Thêm buffer** vào đường dữ liệu để cải thiện timing.
- Resize cell** (thay cell hiện tại bằng cell có khả năng dẫn mạnh hơn):
  - Thay bằng **cell có khả năng điều khiển lớn hơn** (larger drivability).
  - Giúp tăng tốc độ truyền tín hiệu  $\rightarrow$  dữ liệu đến sớm hơn.

Hình bên trái minh họa việc chèn buffer và thay cell để sửa lỗi Setup.



## 2. Nếu xảy ra vi phạm Hold (Hold violation):

→ Dữ liệu đến quá sớm, không giữ ổn định sau cạnh clock.

### Giải pháp:

- Thêm delay element (tăng trễ nhân tạo) trên đường dữ liệu.

### Resize cell:

- Thay bằng cell có khả năng điều khiển nhỏ hơn (smaller drivability).
- Làm chậm tín hiệu lại để tránh dữ liệu cập nhật quá sớm.

Hình bên phải mô tả kỹ thuật sửa lỗi Hold bằng cách chèn thêm delay, giảm kích thước cell.

### Mục tiêu tổng thể:

- Đảm bảo không vi phạm các ràng buộc thời gian.
- Duy trì hiệu suất và độ ổn định của hệ thống sau khi bố trí và định tuyến.

## Layout Extraction (Phục vụ cho kiểm tra tính toàn vẹn tín hiệu và kiểm tra thời gian - Signal Integrity & Timing Check)

### Khái niệm:

- RC Extraction là quá trình trích xuất giá trị điện dung ký sinh(capacitance) và điện trở dây(resistance) từ layout sau khi hoàn tất P&R.
- Đây là bước quan trọng để đảm bảo rằng mạch hoạt động đúng với các ràng buộc thời gian và không bị lỗi tín hiệu.

### Các loại thành phần được trích xuất:

#### 1. Field Capacity (As, Al):

- Điện dung giữa dây và lớp nền (substrate) hoặc các lớp trên/dưới.
- Bị ảnh hưởng bởi khoảng cách đến lớp nền và hình dạng dây dẫn.

#### 2. Crossing Wiring Capacity (Al<sub>21</sub>, Al<sub>12</sub>):

- Điện dung giữa các lớp dây chéo nhau (ví dụ Metal1 và Metal2).
- Ảnh hưởng bởi khoảng cách và chiều dài phần dây chồng lên nhau.

#### 3. Parallel Wiring Capacity (Ap):

- Điện dung ký sinh giữa các dây song song, thường trên cùng một lớp hoặc lớp liền kề.
- Phụ thuộc vào khoảng cách giữa dây (d) và chiều dài song song.

#### 4. Sheet Resistance (Rs):

- Điện trở của mỗi đơn vị diện tích dây dẫn.
- Phụ thuộc vào vật liệu và chiều rộng/chiều dài dây.

### Lưu ý về ảnh hưởng của cấu trúc layout: Điện dung ký sinh (parasitic capacitance) thay đổi tùy thuộc vào:

- Mật độ dây xung quanh (dense hay coarse),
- Có hay không dây liền kề (adjacent),
- Vị trí dây trên/bên dưới.

### Ví dụ minh họa:

- Bên trái:** Metal1 dày đặc và Metal2 thưa → điện dung thấp hơn.
- Bên phải:** Metal1 và Metal3 dày đặc → điện dung và tương tác cao hơn.

### Mục tiêu của Layout Extraction:

- Tính toán chính xác RC delay cho từng đường dây.
- Hỗ trợ kiểm tra timing (STA) và độ toàn vẹn tín hiệu (signal integrity).
- Từ đó tối ưu lại thiết kế nếu cần (ví dụ: buffer insertion, wire sizing...).

## Timing Verification (Xác minh thời gian hoạt động sau layout)

Timing Verification được thực hiện dựa trên dữ liệu trễ sau layout (post-layout delay data).

Mục tiêu: kiểm tra xem thiết kế có đáp ứng các ràng buộc thời gian (timing constraints) như setup time, hold time, transition delay...

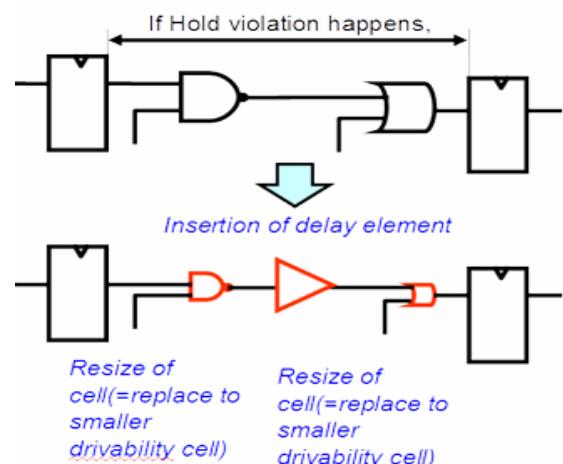
### Quy trình xác minh timing:

#### RC Extraction:

- Trích xuất thông tin điện trở và điện dung từ layout (sử dụng dữ liệu GDSII).
- Gồm: RC info → Wire load data.

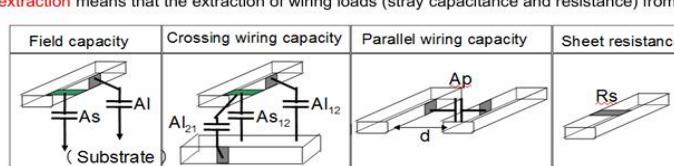
#### Layout Delay Calculation:

- Dựa trên wire load và thông tin thư viện cell (cell delay, wire model).
- Dùng công cụ như PrimeTime, SignalStorm,... để tính post-layout delay.

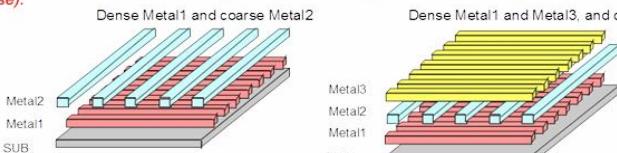


## Layout Extraction (for Signal Integrity and Timing Check)

RC extraction means that the extraction of wiring loads (stray capacitance and resistance) from layout after P&R.



The parasitic capacitance changes depending on peripheral wiring (with/without adjacent wire, top and bottom, coarse or dense).



## Tạo dữ liệu SDF (Standard Delay Format):

- Bao gồm thông tin trễ của toàn mạch, được sinh từ bước tính delay ở trên.

## Tạo Netlist sau layout (Post-layout Gate-level Netlist):

- File netlist Verilog phản ánh chính xác cấu trúc mạch sau P&R.

## Thực hiện Timing Verification:

- Phân tích Static Timing Analysis (STA).
- Nếu cần có thể dùng Dynamic Timing Analysis (DTA) (đối với mạch phức tạp hơn).
- Dùng công cụ như PrimeTime, NC-Verilog,...

## Đưa ra kết quả xác minh:

- Kiểm tra các yếu tố: Setup timing, Hold timing, Transition timing, Các lỗi timing (nếu có)

## Nếu NG (Not Good):

- Quay lại sửa thiết kế (Timing ECO - Engineering Change Order).
- Áp dụng constraint mới → P&R lại.

## Luồng dữ liệu chính:

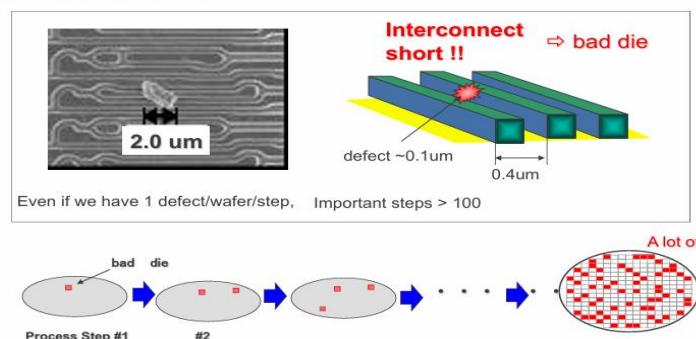
(Layout Data (GDSII) + RC Info + Library Info) → (Layout Delay Calculation) → (SDF File + Netlist) → Timing Verification → Kết quả

## Mục tiêu cuối cùng:

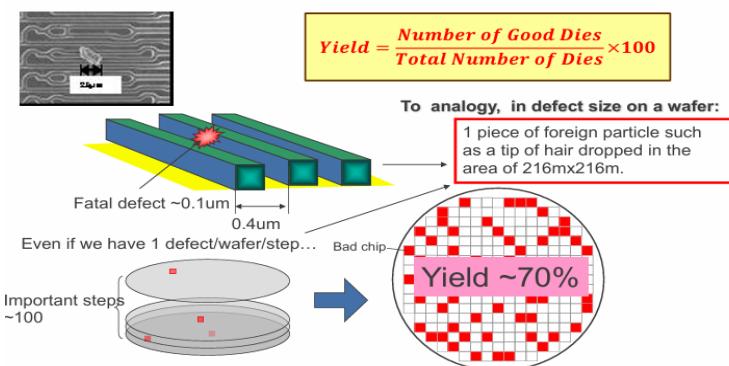
- Thiết kế **đáp ứng tất cả các ràng buộc thời gian**.
- Đảm bảo mạch hoạt động đúng khi triển khai trên chip thực tế (Si – silicon).

## 6.6. Process Defect and Yield

### Patten Defect and Yield



### Patten Defect and Yield (cont'd)



### Yield

Công thức tính yield:

$$Y = \exp [-(S \times D)]$$

Trong đó:

- S: Diện tích chip ( $\text{cm}^2$ )
- D: Mật độ lỗi (defect density, đơn vị: lỗi/ $\text{cm}^2$ )
- Y: Tỉ lệ chip tốt (yield)

### Ví dụ minh họa:

- Cho  $D = 1 \text{ lỗi}/\text{cm}^2$
- Chip có kích thước:  $6.0 \text{ mm} \times 6.0 \text{ mm}$   
→ Diện tích:  $36 \text{ mm}^2 = 0.36 \text{ cm}^2$
- Áp dụng công thức:

$$Y = \exp [-(0.36 \times 1)] = \exp(-0.36) \approx 0.6976 \Rightarrow Y = 69.8\%$$

### Expression – Biểu Thức Hiệu Suất (Tỉ lệ IC tốt)

#### Khái niệm:

- Yield (Y)** là tỉ lệ số chip **tốt** có thể thu được từ một wafer.
- Tỉ lệ này phụ thuộc vào:
  - Diện tích chip (S)** – càng lớn, xác suất gặp lỗi càng cao.
  - Mật độ lỗi (D)** – số lỗi trung bình trên mỗi  $\text{cm}^2$

#### Ý nghĩa:

- Nếu **diện tích chip tăng**, thì yield **giảm** (vì xác suất gặp lỗi tăng).
- Nếu **mật độ lỗi giảm**, thì yield **tăng** (chất lượng wafer tốt hơn).
- Công thức này giúp **ước lượng số lượng chip đạt yêu cầu** sau quá trình sản xuất.

### Contamination Reduction (Giảm thiểu nhiễm bẩn trong sản xuất vi mạch)

**Nguồn gốc nhiễm bẩn:** Con người (Tóc, da chết, mồ hôi, bụi từ cơ thể người), yếu tố khác ...

**Giải pháp:** clean room, clean suit