

# LECTURE 7: Tổng Quan Quy Trình VLSI

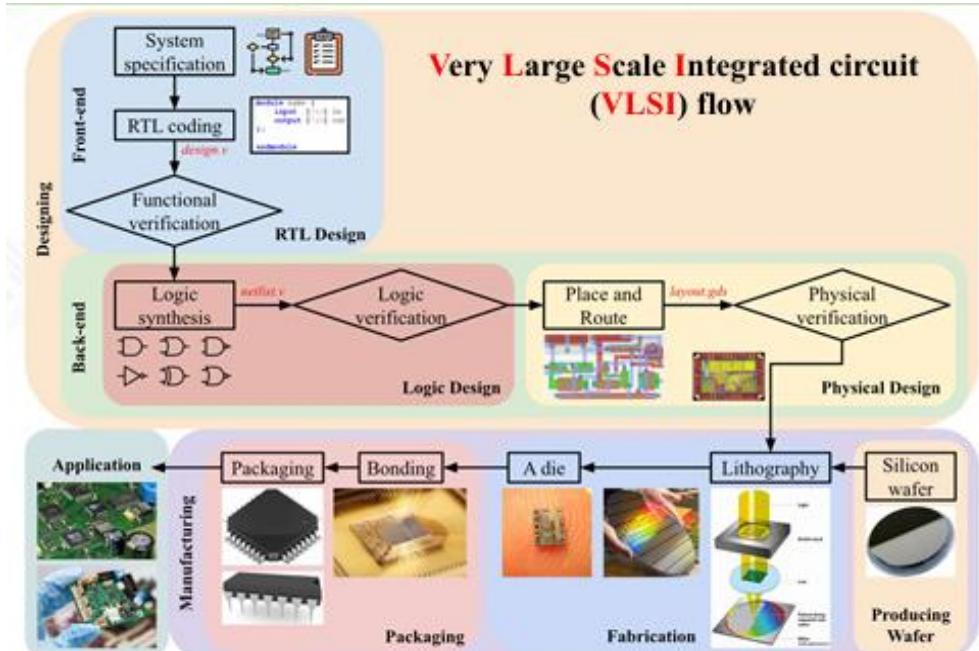
## 7.1 VLSI Manufacturing

### Flow

#### 7.1.1 Overview

Quá trình sản xuất vi mạch gồm:

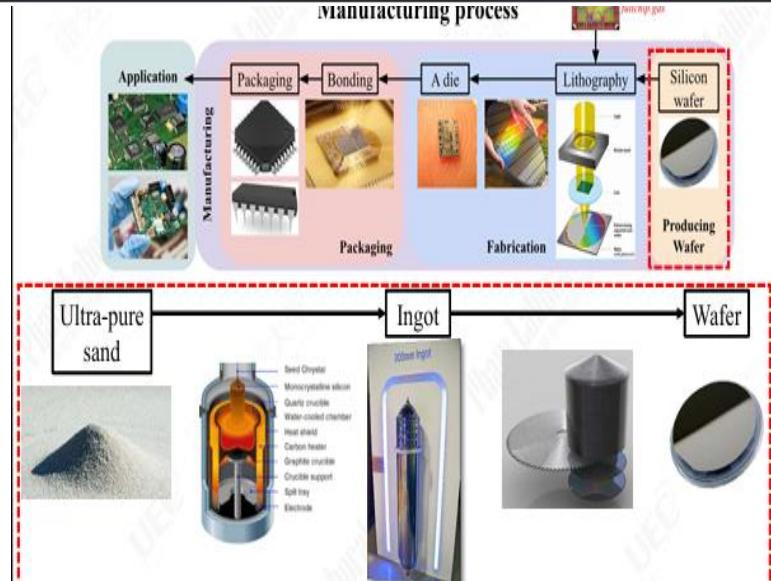
1. Sản xuất silicon wafer
2. Chế tạo
  - + Quang khắc (Lithography)
3. Đóng gói
  - + Bonding
  - + Packaging



#### 7.1.2 Silicon wafer producing

##### 1 Khai thác và tinh chế silicon từ cát thạch anh

- Silicon (Si) là nguyên tố phổ biến thứ hai trên Trái Đất, thường tồn tại dưới dạng cát thạch anh ( $\text{SiO}_2$ ).
- Khai thác cát thạch anh có độ tinh khiết cao, sau đó trải qua phản ứng với than cốc trong lò hồ quang điện ở nhiệt độ khoảng  $1.800 - 2.000^\circ\text{C}$  để tạo ra **silicon thô**.
- **Silicon thô này có độ tinh khiết khoảng 98-99%, nhưng để sử dụng trong vi điện tử, nó cần được tinh chế lên mức 99.9999999% (9N – "nine nines").**



##### 2 Tạo đơn tinh thể silicon bằng phương pháp Czochralski (CZ)

- Phương pháp Czochralski (CZ) – phổ biến nhất: Silicon tinh khiết được nấu chảy trong nồi thạch anh ở nhiệt độ khoảng  $1.400^\circ\text{C}$ .
- Một tinh thể hạt giống (seed crystal) được nhúng vào silicon lỏng và từ từ kéo lên trong khi quay đều. Quá trình này tạo ra một thỏi đơn tinh thể silicon (silicon ingot)

##### 3 Cắt thỏi silicon thành lát wafer mỏng

Sau khi có thỏi silicon, nó được cắt thành các lát mỏng (wafer) bằng lưỡi cưa kim cương hoặc dây cưa. Loại bỏ phần wafer không sử dụng được (defective) Độ dày của wafer thường nằm trong khoảng  $725 - 775 \mu\text{m}$  cho loại 300 mm.

➤ Lưu ý: Bước này rất quan trọng để đảm bảo wafer có độ phẳng cao, giúp các bước chế tạo vi mạch sau này diễn ra chính xác.

##### 4 Mài và đánh bóng wafer để đạt độ phẳng cao:

- Sau khi cắt, bề mặt wafer khá thô ráp, cần trải qua quá trình mài (lapping) và đánh bóng (polishing) để đạt được độ phẳng lý tưởng (chỉ chênh lệch vài nanomet trên toàn bộ bề mặt). Thường sử dụng kỹ thuật CMP (Chemical Mechanical Polishing) kết hợp hóa chất và lực cơ học để làm nhẵn bề mặt.
- Kết quả: Tấm wafer có độ mịn gần như hoàn hảo, sẵn sàng cho quá trình chế tạo vi mạch.

##### 5 Kiểm tra và làm sạch để đảm bảo chất lượng wafer

- Wafer sau khi đánh bóng sẽ được kiểm tra bằng các công cụ đo chính xác như laser interferometer để đảm bảo không có khuyết tật.
- Tiếp theo là quy trình làm sạch bằng hóa chất siêu tinh khiết để loại bỏ bụi bẩn, ion kim loại và tạp chất.
- Kết quả: Tấm wafer sạch và đạt tiêu chuẩn để tiến vào giai đoạn sản xuất chip (IC fabrication).

## 7.1.3 Lithography

### ❶ (1) Thiết kế vi mạch – A Full-chip Layout

Đây là **sơ đồ thiết kế mạch tích hợp (IC layout)**, chứa toàn bộ thông tin về vị trí bóng bán dẫn, đường dây kết nối, và các lớp vật liệu.

Thiết kế này được chuyển thành **photomask** để sử dụng trong quá trình quang khắc.

### ❷ (2) Photomask – Mặt nạ quang học

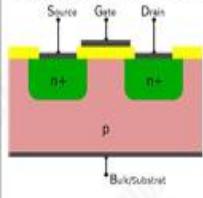
**Photomask** là tấm kính có phủ lớp cản sáng (chrome) chứa hình ảnh của thiết kế vi mạch.

Nó hoạt động như một **bản mẫu**, cho phép ánh sáng truyền qua các vùng mong muốn khi chiếu lên wafer.

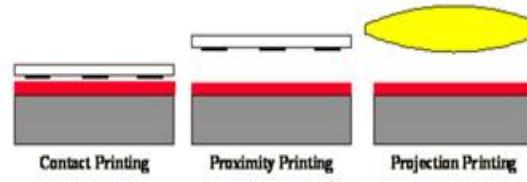
### ❸ (3) Photo-lithography – Quang khắc

Đây là bước cốt lõi trong chế tạo vi mạch, giúp **chuyển thiết kế từ photomask lên wafer**.

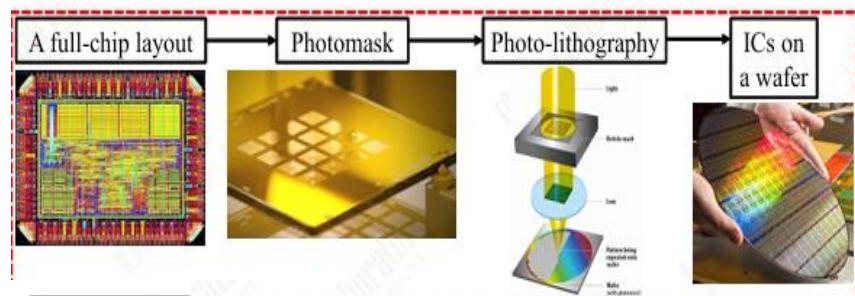
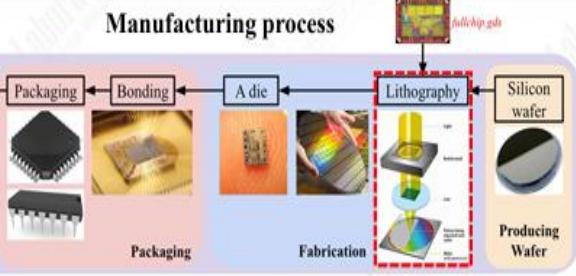
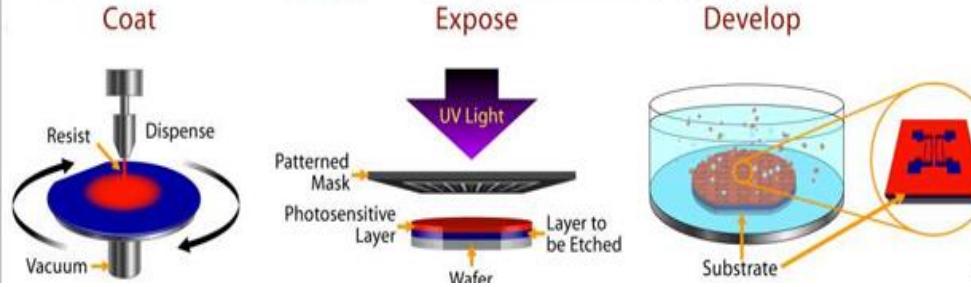
Typical MOSFET



Three types of printing



Three primary steps in photo-lithography



### Ba phương pháp in mẫu trong lithography:

- Contact Printing:** Độ phân giải cao nhưng dễ hỏng mask.
- Proximity Printing:** Giảm rủi ro hỏng mask nhưng độ phân giải thấp hơn contact printing
- Projection Printing:** Công nghệ hiện đại nhất, dùng trong sản xuất chip tiên tiến (EUV, DUV).

#### 1 Contact Printing (In tiếp xúc)

- Nguyên lý:** Photomask đặt trực tiếp lên bề mặt wafer, sau đó ánh sáng được chiếu qua.
- Ưu điểm:** Độ phân giải cao vì không có khoảng cách giữa mask và wafer.
- Nhược điểm:** Mask có thể bị hỏng do tiếp xúc nhiều lần với wafer, không phù hợp cho sản xuất hàng loạt. Phải sản xuất mask có kích thước rất nhỏ, chi phí cao
- Ứng dụng:** Chủ yếu dùng trong nghiên cứu hoặc sản xuất vi mạch có độ phân giải cao nhưng số lượng nhỏ.

#### 2 Proximity Printing (In gần tiếp xúc)

- Nguyên lý:** Photomask được đặt gần wafer nhưng có một khoảng cách nhỏ (khoảng 10-50 μm).
- Ưu điểm:** Giảm nguy cơ làm hỏng mask, kéo dài tuổi thọ mask so với Contact Printing.
- Nhược điểm:** Giảm độ phân giải do ánh sáng bị nhiễu khi đi qua khoảng trống. Phải sản xuất mask có kích thước rất nhỏ, chi phí cao
- Ứng dụng:** Thường dùng cho sản xuất các vi mạch đơn giản, không yêu cầu độ phân giải quá cao.

#### 3 Projection Printing (In chiếu xạ – dùng trong EUV & DUV Lithography hiện đại)

- Nguyên lý:** Sử dụng hệ thống thấu kính để thu nhỏ và chiếu hình ảnh của photomask lên wafer, không có tiếp xúc trực tiếp.
- Ưu điểm:**
  - Độ phân giải rất cao (dưới 10nm, dùng trong chip 5nm, 3nm).
  - Mask không bị hư hỏng do không tiếp xúc với wafer.
  - Sản xuất mask có kích thước lớn hơn, chi phí rẻ hơn, dùng thấu kính hội tụ để có hình ảnh nhỏ hơn
- Nhược điểm:**
  - Hệ thống phức tạp và rất đắt đỏ (máy EUV có giá ~150 triệu USD).
  - Đòi hỏi môi trường chân không và nguồn sáng mạnh (Extreme UV – 13.5 nm).
- Ứng dụng:** Công nghệ chính trong sản xuất chip hiện đại (Intel, TSMC, Samsung)

## 2. Ba bước chính trong Lithography (Photo-lithography)

### 1 Coat – Phủ lớp nhạy sáng (Photoresist Coating)

- Wafer được phủ một lớp photoresist (chất nhạy sáng) bằng kỹ thuật **spin-coating**.
- Photoresist có hai loại:**
  - Positive resist:** Vùng tiếp xúc với ánh sáng sẽ bị hòa tan khi rửa.
  - Negative resist:** Vùng tiếp xúc với ánh sáng sẽ cứng lại và không bị rửa trôi.
- Wafer được **sấy khô (soft bake)** để loại bỏ dung môi và giúp photoresist bám chắc hơn.

### 2 Expose – Chiếu sáng qua photomask (Exposure)

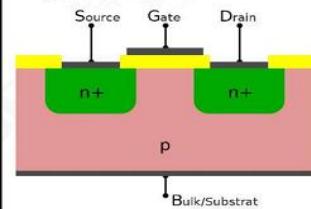
- Photomask** chứa hình ảnh của mạch vi điện tử được đặt lên trên wafer.
- Nguồn sáng (UV hoặc EUV)** chiếu qua photomask và tác động lên lớp photoresist bên dưới.
- Tùy theo loại resist, vùng bị chiếu sáng sẽ bị hòa tan hoặc giữ lại sau bước tiếp theo.

### 3 Develop – Hiện hình (Development)

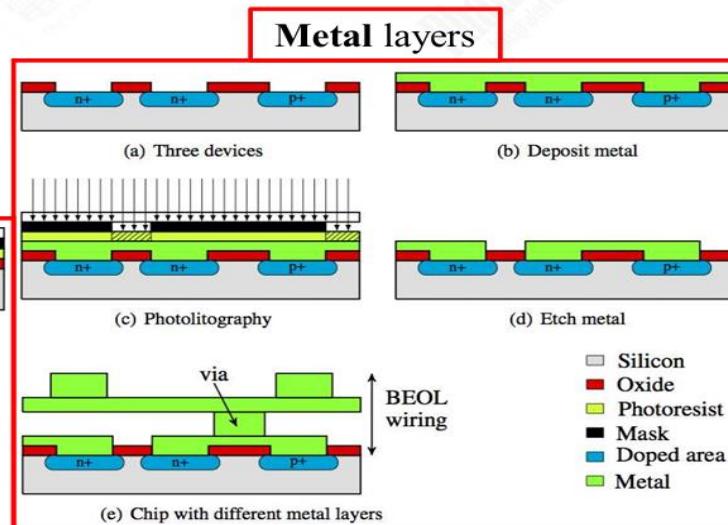
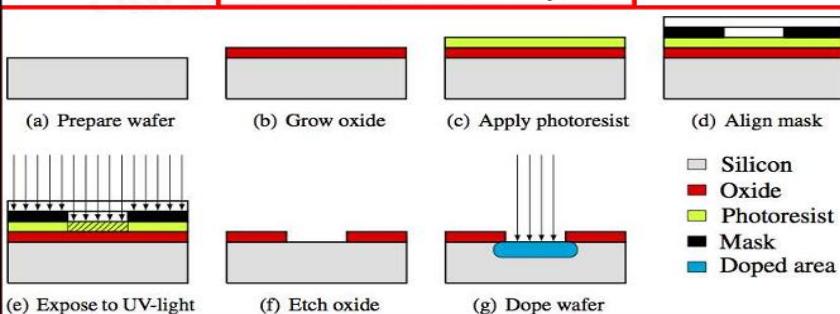
- Wafer được rửa bằng dung dịch hóa học để loại bỏ phần photoresist không mong muốn.
- Kết quả: **Mẫu vi mạch được khắc lên lớp photoresist**, chuẩn bị cho các bước tiếp theo như **etching (ăn mòn)**, **doping (cấy ion)**, và **metallization (mạ kim loại)**.

## 2. VLSI Manufacturing Flow (6/10) (Photo-)Lithography

### Typical MOSFET



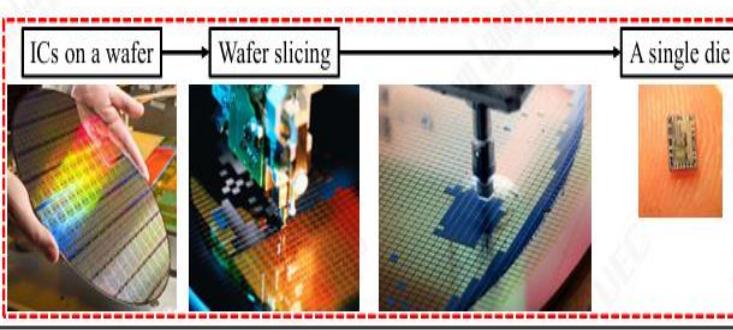
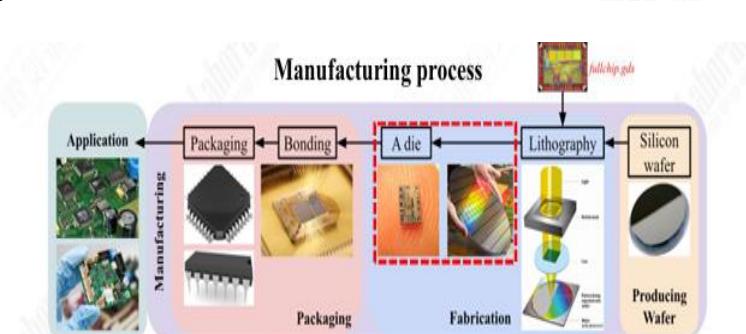
### Active and Gate layers



17

### 1 ICs on a wafer – Các vi mạch trên tấm wafer

- Sau khi hoàn thành **Lithography**, wafer chứa **hàng nghìn vi mạch (ICs - Integrated Circuits)**. Mỗi ô vuông nhỏ trên wafer là một die (**chip riêng lẻ**).
- Trước khi cắt, wafer có thể được kiểm tra chất lượng bằng **wafer probing** (dùng đầu dò điện để kiểm tra tín hiệu).



### 2 Wafer Slicing – Cắt tấm wafer

- Blade Dicing (Cắt bằng lưỡi cưa kim cương):** Dùng **lưỡi cưa kim cương siêu mỏng** để cắt theo đường rãnh trên wafer. Phương pháp phổ biến nhất, phù hợp với nhiều loại chip.
- Laser Dicing (Cắt bằng laser):** Sử dụng tia laser để cắt wafer mà không tiếp xúc trực tiếp. Chính xác cao, ít gây hỏng chip. Chi phí cao hơn blade dicing.
- Stealth Dicing (Cắt ngầm bằng laser bên trong wafer):** Tia laser tạo vết nứt **bên trong wafer**, sau đó wafer được bẻ nhẹ để tách die.

### 3 A Single Die – Một chip riêng lẻ

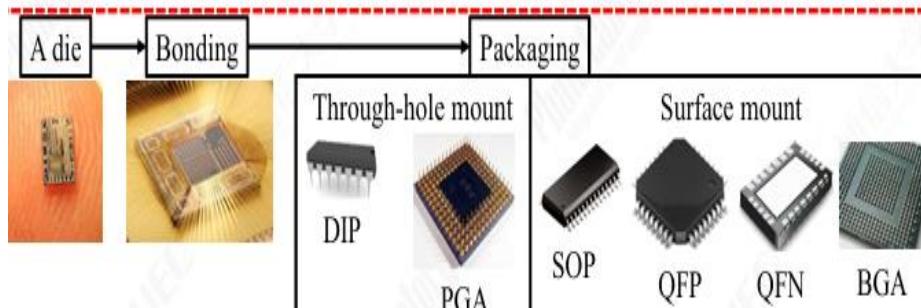
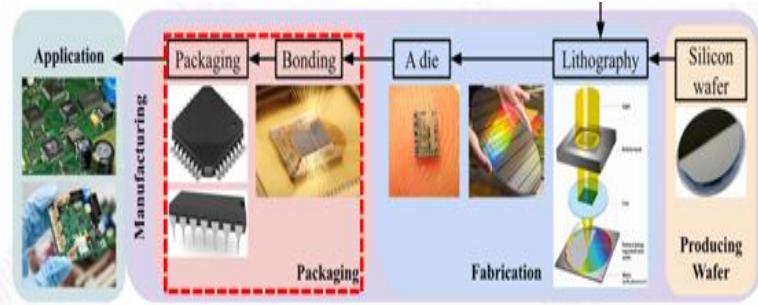
Sau khi cắt, mỗi die (**chip riêng lẻ**) được kiểm tra chất lượng. Các chip đạt chuẩn sẽ được **đóng gói (Packaging)** và hàn kết nối (Bonding). Chip lỗi sẽ bị loại bỏ hoặc dùng cho nghiên cứu.

## 7.1.4 Packaging

### 1. Bonding – Kết nối chip với đế mạch

- Sau khi tách ra từ wafer, **chip (die)** cần được gắn vào một **đế mạch (substrate hoặc lead frame)** để có thể giao tiếp với mạch điện ngoài.
- Có ba phương pháp **bonding** phổ biến:

- ♦ **Wire Bonding**: Dùng dây kim loại (thường là vàng hoặc nhôm) để kết nối chip với chân dẫn. **phương pháp phổ biến nhất** vì đơn giản và tiết kiệm chi phí.
- ♦ **Flip-Chip Bonding**: Lật ngược chip và hàn trực tiếp lên đế (dùng bump hàn).
- ♦ **Tape-Automated Bonding (TAB)**: Sử dụng dải phim dẫn điện để kết nối chip với đế.



### 2. Packaging – Đóng gói vi mạch

Mục tiêu:

- Bảo vệ chip** khỏi hư hỏng cơ học, ẩm, và nhiễu điện tử.
- Tạo giao tiếp điện** giữa chip và bo mạch chủ (PCB).

Có hai kiểu đóng gói chính:

- ♦ **Through-hole mount (Gắn xuyên lỗ – dùng chân cắm vào PCB)**

#### 1. DIP (Dual In-line Package):

- Hai hàng chân cắm xuyên qua bo mạch.
- Dễ hàn, phù hợp với mạch đơn giản.

#### 2. PGA (Pin Grid Array):

- Các chân cắm xếp thành lưới (thường dùng trong CPU cũ như Intel Pentium).
- Dễ thay thế, nhưng không tối ưu cho chip nhỏ gọn.

- ♦ **Surface mount (Gắn bề mặt – không cần chân xuyên lỗ)**

#### 1. SOP (Small Outline Package):

- Kiểu đóng gói nhỏ hơn DIP, có chân nằm hai bên.
- Thường dùng trong vi điều khiển và IC bộ nhớ.

#### 2. QFP (Quad Flat Package):

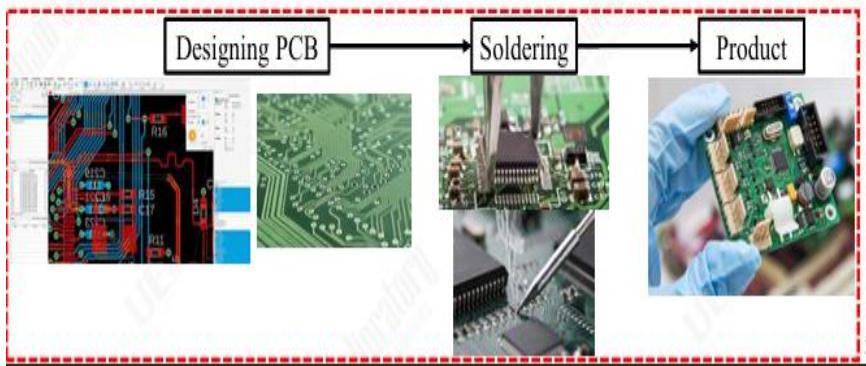
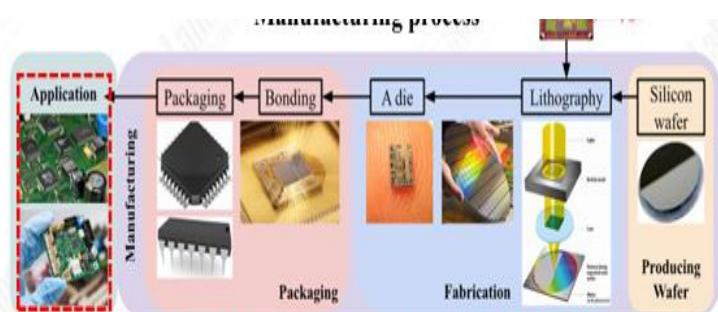
- Chân dẫn mỏng và nhiều hơn SOP, xếp đều bốn cạnh.
- Dùng trong vi xử lý và IC tín hiệu số.

#### 3. QFN (Quad Flat No-lead):

- Không có chân ra ngoài, thay vào đó, chip tiếp xúc trực tiếp với PCB.
- Tản nhiệt tốt, phù hợp với thiết bị di động.

#### 4. BGA (Ball Grid Array):

- Sử dụng các bóng hàn (solder balls) thay vì chân kim loại.
- Đóng gói phổ biến cho CPU, GPU, RAM vì tăng mật độ chân kết nối.

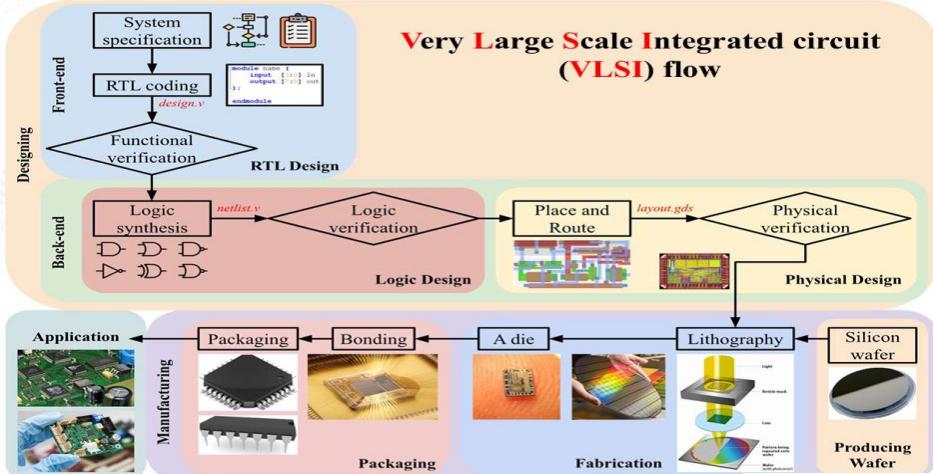


## 7.1.5 Application

Chip sau khi được đóng gói sẽ được sử dụng trong các mạch điện tử, thiết kế PCB,....

# 7.2 VLSI Design Flow

## 3. VLSI Design Flow (1/12) Overview



### 7.2.1 Overview

- Front-end (RTL Design)
- Back-end
- + Logic Design
- + Physical Design

- Quy trình thiết kế VLSI chia thành front-end & back-end.
- Front-end: Làm việc với chức năng. Sản phẩm cuối (đầu ra) là RTL (design.v) file.
- Back-end: Làm việc với thiết kế vật lý. Sản phẩm cuối (đầu ra) layout (layout.gds) file.

## 7.2.2 Logic Synthesis

### Quy trình tổng hợp logic:

1. Đầu vào Verilog/VHDL (Input Verilog) – Thiết kế mô tả hành vi của mạch.
2. Tổng hợp mức generic (Syn: generic) – Chuyển mã HDL thành các cổng logic cơ bản (AND, OR, XOR, FF, etc.).
3. Mapping đến thư viện standard cell (Syn: mapping to standard cells) – Chuyển mạch logic sang các cell có sẵn trong công nghệ sản xuất.

### Standard Cell là gì?

- Standard Cell (StdCell) là các phần tử mạch số được thiết kế sẵn để biểu diễn các hàm logic cơ bản (Boolean function) hoặc hàm lưu trữ (storage function).
- StdCell giúp tự động hóa quá trình thiết kế vi mạch, cho phép công cụ tổng hợp logic mapping (ánh xạ) mạch số từ Verilog/VHDL vào các cell có sẵn.
- Dùng trong việc sản xuất chip, mỗi công ty đều có một thư viện Stdcell nhất định

### Các loại Standard Cell phổ biến:

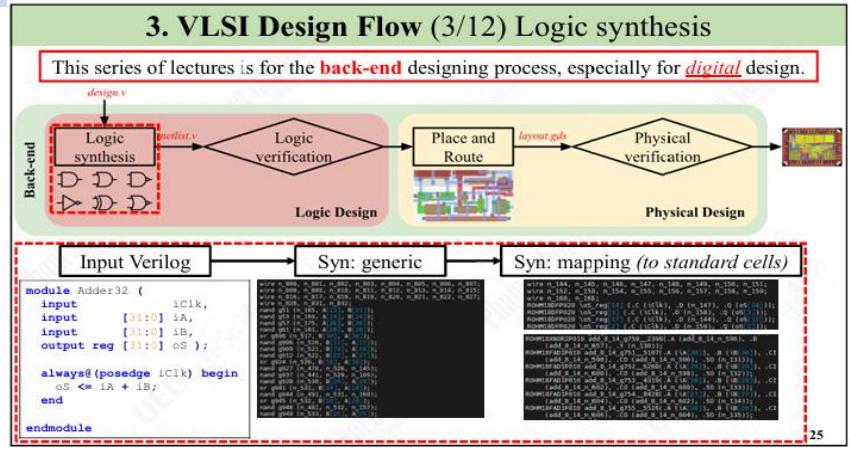
- Cổng logic: AND, OR, NAND, NOR, NOT, XOR, XNOR
- Mạch lưu trữ: Flip-Flop, Latch (dùng để lưu trữ trạng thái trong mạch đồng hồ)
- Mạch số phức tạp: Half-Adder, Full-Adder, Multiplexer (MUX), Decoder, etc

### Cell-based method:

- Phân rã (Break-down) một mạch số lớn thành các hàm Boolean cơ bản và Flip-Flop/Latch.
- Ánh xạ (Mapping) các thành phần đó vào thư viện Standard Cell.
- Tạo layout từ Standard Cell để dễ dàng sản xuất chip.

### Lợi ích của Standard Cell trong thiết kế VLSI

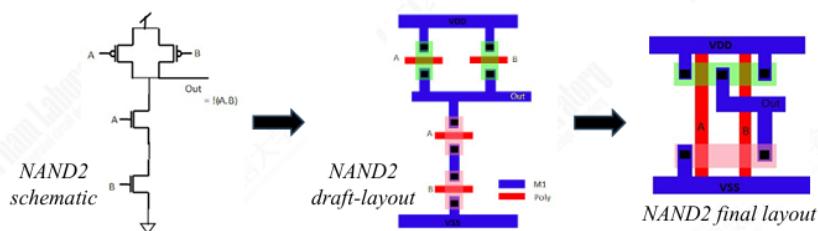
- **Tự động hóa quá trình thiết kế:** Dễ dàng tổng hợp từ Verilog/VHDL.
- **Tiết kiệm thời gian và chi phí:** Không cần thiết kế từng transistor riêng lẻ.
- **Tái sử dụng dễ dàng:** Standard Cell có thể được sử dụng lại trong nhiều thiết kế khác nhau.
- **Tối ưu hóa kích thước & hiệu năng:** Dựa vào công nghệ bán dẫn (CMOS 5nm, 7nm, 14nm, v.v.).



## 3. VLSI Design Flow (4/12) Standard cell

Traditionally, a **Standard Cell (StdCell)** is a direct representation of a Boolean function (i.e., AND, NAND, OR, NOR, NOT, XOR, XNOR) or a storage function (i.e., flip-flop and latch).

Nowadays, complex **StdCells** are also often used, such as half-adder, full-adder, mux, etc.

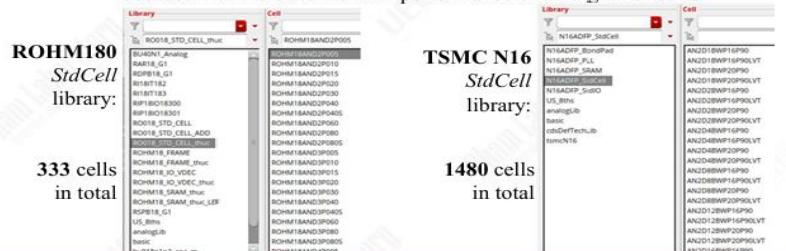


**Cell-based method:** break-down a complex digital circuit into basic Boolean functions and flip-flops/latches, then mapping them to **StdCells**.

## 3. VLSI Design Flow (5/12) Standard cell library

A **Standard Cell Library (StdCell library)** is a group of many StdCells. In modern VLSI design, especially in digital design, a **StdCell library** is always required.

From the beginning, a **StdCell library** should be designed very carefully because it will dictate the final performance of all digital circuits.



Generally speaking, a **StdCell library** with more cells is often better than one with fewer.

## Mục tiêu của Logic Synthesis

- Chuyển mã HDL (Verilog/VHDL) thành mạch số tối ưu hóa để triển khai trên chip.
- Sử dụng công cụ EDA như Synopsys Design Compiler, Cadence Genus, Mentor Graphics.
- Chuyển đổi **mã Verilog** thành **mạch số sử dụng Standard Cells (StdCells)**
- Sử dụng phương pháp **Cell-based design**, giúp ánh xạ mạch số vào thư viện StdCell có sẵn.
- Đây là bước quan trọng trước khi thực hiện **Place & Route (PnR)**.

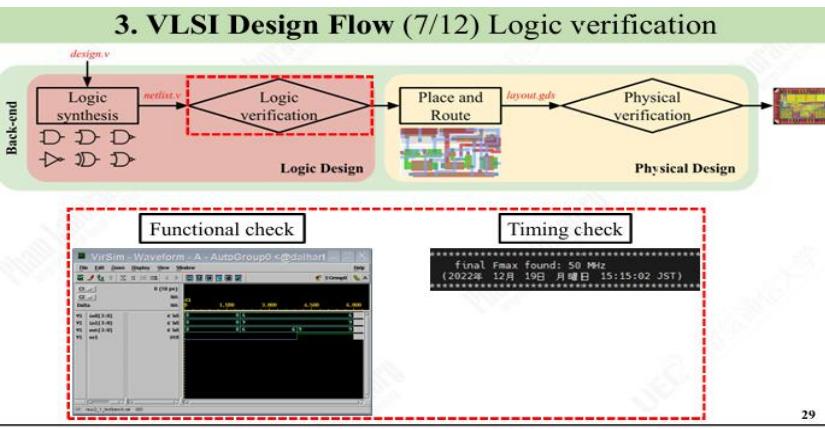


So, by using the cell-based method, the original Verilog file was **synthesized** and **mapped** into **StdCells** in the StdCell library.

That means the next step, **Place-and-Route (PnR)**, doesn't have to deal with *logic* or *function* anymore, it only have to deal with...*placing* the **StdCells** and *routing* them.

## Ý nghĩa của Logic Synthesis

- Giúp tự động hóa thiết kế VLSI, không cần vẽ mạch thủ công.
- Tối ưu hóa kích thước, hiệu suất, và công suất tiêu thụ dựa trên thư viện StdCell.
- Chuẩn bị cho bước tiếp theo – Place & Route (PnR), nơi các StdCells được sắp xếp trên chip và kết nối với nhau.



### 7.2.3 Logic verification

- Logic Verification** là quá trình kiểm tra xem thiết kế mạch số sau khi tổng hợp (Logic Synthesis) có **đúng với yêu cầu** thiết kế ban đầu không.
- Đây là bước **quan trọng** trong giai đoạn **Logic Design**, trước khi tiến hành **Place & Route (PnR)**.
- Nếu có lỗi, cần **sửa lại** thiết kế hoặc **tối ưu** lại **tổng hợp logic** trước khi tiếp tục với PnR.
- Logic Verification gồm **kiểm tra chức năng (Functional Check)** và **kiểm tra thời gian (Timing Check)**.

#### Functional Check (Kiểm tra chức năng)

- Kiểm tra xem **mạch hoạt động đúng theo yêu cầu không**.
- Sử dụng **mô phỏng (simulation)** với testbench để chạy các tín hiệu đầu vào và quan sát kết quả đầu ra.
- Công cụ phổ biến: **ModelSim, VCS, Xcelium, QuestaSim**.
- Hình ảnh trong slide hiển thị **dạng sóng (waveform)** của tín hiệu đầu vào và đầu ra khi kiểm tra.

#### Timing Check (Kiểm tra thời gian)

- Dánh giá **tốc độ hoạt động của mạch**, tìm tần số hoạt động tối đa (**Fmax**).
- Kiểm tra xem có lỗi **Timing Violation** nào không (ví dụ: Setup Time, Hold Time).
- Công cụ phân tích: **PrimeTime, Tempus, STA (Static Timing Analysis)**

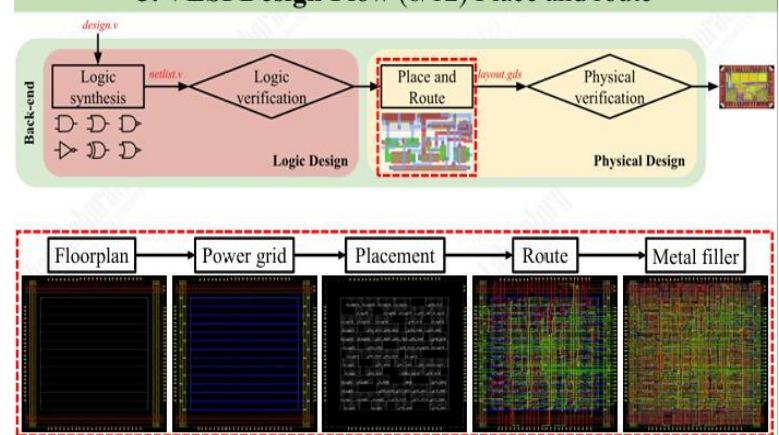
#### Tại sao Logic Verification quan trọng?

- Phát hiện lỗi sớm**: Tránh sai sót trước khi bước sang giai đoạn Place & Route.
- Đảm bảo mạch hoạt động đúng**: Kiểm tra tính chính xác của logic.
- Tối ưu hóa hiệu suất**: Đảm bảo mạch có thể chạy ở tốc độ mong muốn mà không vi phạm thời gian.

### 7.2.4 Place and Route (PnR)

- Place and Route (PnR) là quá trình sắp xếp các Standard Cells (StdCells) lên chip và kết nối chúng bằng các dây dẫn (routing).
- Đây là bước chuyển từ logic design (mô tả chức năng) sang physical design (bố cục vật lý của chip), trước khi xác minh layout và sản xuất chip.
- Đầu vào: Netlist (kết quả của logic synthesis và verification).
- Đầu ra: Layout hoàn chỉnh (file GDSII) để chuẩn bị cho quá trình sản xuất chip.
- Place and Route gồm 5 bước: Floorplan → Power Grid → Placement → Route → Metal Filler.
- Bố cục tốt giúp cải thiện hiệu suất, giảm tiêu thụ năng lượng và tăng độ tin cậy của vi mạch.

### 3. VLSI Design Flow (8/12) Place and route



# Các bước trong Place and Route

## 1 Floorplan 🏠

- Xác định kích thước chip, vị trí các vùng logic, vùng I/O (Input/Output), và các khối quan trọng.

## 2 Power Grid (Mạng phân phối nguồn) ⚡

- Thiết kế hệ thống cấp nguồn VDD, VSS để đảm bảo dòng điện đủ cung cấp cho toàn bộ chip.

## 3 Placement (Sắp xếp các StdCells) □

- Đặt các Standard Cells lên chip sao cho tối ưu diện tích và hiệu suất.
- Tránh hiện tượng congestion (tắc nghẽn kết nối) giữa các cell.

## 4 Route (Kết nối tín hiệu) 🔗

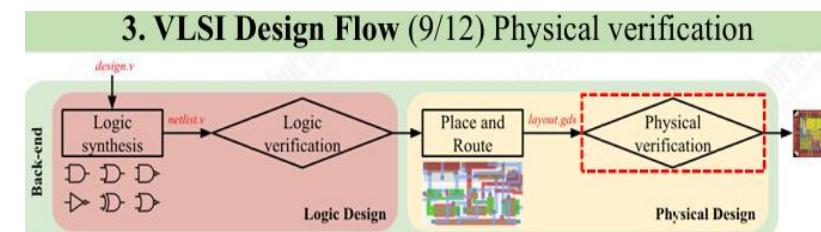
- Tạo dây dẫn kim loại kết nối các cell theo netlist.
- Cố gắng giảm độ trễ (delay) và nhiễu (crosstalk noise) trong kết nối.

## 5 Metal Filler (Bổ sung kim loại) 💫

- Chèn các vùng kim loại để cải thiện tính toàn vẹn tín hiệu (signal integrity) và giúp sản xuất dễ dàng hơn.

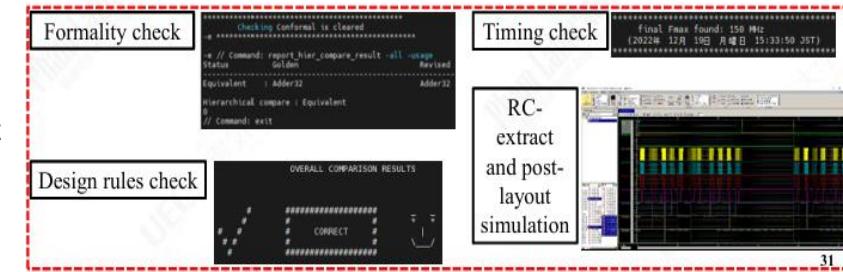
### Tại sao Place and Route quan trọng?

- ✓ Xác định hiệu suất và tiêu thụ điện năng của chip.
- ✓ Tối ưu không gian và kết nối để giảm độ trễ.
- ✓ Chuẩn bị layout GDSII để sản xuất chip thực tế.



## 7.2.5 Physical Verification

- Đây là bước kiểm tra bối cục vật lý (layout) của chip sau khi hoàn thành Place & Route (PnR).
- Đảm bảo rằng layout đáp ứng đầy đủ yêu cầu thiết kế và có thể sản xuất được mà không gặp lỗi.
- Nếu có lỗi, cần chỉnh sửa lại layout hoặc tối ưu lại PnR trước khi đưa vào sản xuất.



### Các bước kiểm tra chính trong Physical Verification

#### 1 Formality Check (Kiểm tra hình thức) ✅

- So sánh thiết kế logic ban đầu (Netlist) với thiết kế sau khi tổng hợp.
- Đảm bảo không có lỗi sai khác giữa thiết kế logic và thiết kế vật lý.
- Công cụ phổ biến: Synopsys Formality, Cadence Conformal.

#### 2 Design Rule Check (DRC – Kiểm tra quy tắc thiết kế) 🔎

- Xác minh layout tuân thủ **các quy tắc công nghệ sản xuất** (ví dụ: khoảng cách giữa các dây, kích thước transistor, độ dày kim loại...).
- Công cụ phổ biến: Calibre DRC (Mentor), PVS (Cadence), IC Validator (Synopsys).

#### 3 Timing Check (Kiểm tra thời gian) ⏳

- Phân tích tốc độ và độ trễ của mạch sau khi layout hoàn tất.
- Xác định tần số tối đa (**Fmax**) mà mạch có thể chạy ổn định.
- Công cụ phổ biến: PrimeTime (Synopsys), Tempus (Cadence).

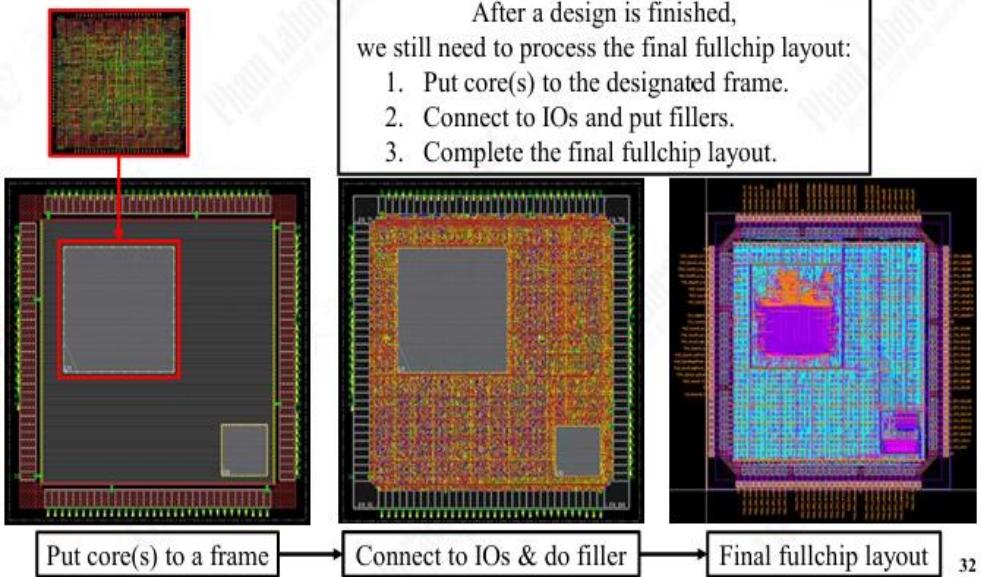
#### 4 RC Extraction & Post-Layout Simulation ⚡

- RC Extraction:** Trích xuất giá trị điện trở (R) và điện dung (C) từ layout để đánh giá ảnh hưởng đến hiệu suất mạch.
- Post-Layout Simulation:** Mô phỏng lại mạch với các thông số RC thực tế để kiểm tra xem có lỗi nào xuất hiện sau khi routing không.
- Công cụ phổ biến: Calibre xRC, StarRC, Quantus QRC.

### Tại sao Physical Verification quan trọng?

- Đảm bảo thiết kế có thể sản xuất mà không bị lỗi.
- Xác minh tính chính xác của mạch sau khi routing.
- Tối ưu hóa hiệu suất và tránh các vấn đề về thời gian, tiêu thụ điện năng.

### 3. VLSI Design Flow (10/12) Fullchip (frame) integration



#### Fullchip Layout là gì?

- Layout là bản vẽ vật lý thể hiện **hình dạng, kích thước, và vị trí các thành phần trên chip** (standard cells, dây nối, vùng nguồn, IO...).
- Fullchip layout** là bối cảnh cuối cùng của **tổng bộ con chip** trước khi xuất ra file **GDSII** để đem đi sản xuất.
- Đảm bảo tối ưu hiệu suất, điện năng, và diện tích
- Chuẩn bị dữ liệu chính xác để **gửi tới nhà máy chế tạo (foundry)**
- Tránh lỗi layout, thiếu kết nối, hoặc vi phạm quy tắc thiết kế (DRC)

### 3. Cấu trúc chính trong layout chip

Thành phần	Mô tả
Core	Phần logic chính của chip
IO Cells	Giao tiếp với thế giới bên ngoài
Pad Ring	Vòng bảo vệ và kết nối điện
Power Grid	Cấp nguồn VDD/VSS cho toàn bộ chip
Metal Layers	Các lớp kim loại để kết nối tín hiệu
Fillers	Cell trống để giữ tính đồng đều và ổn định vật lý

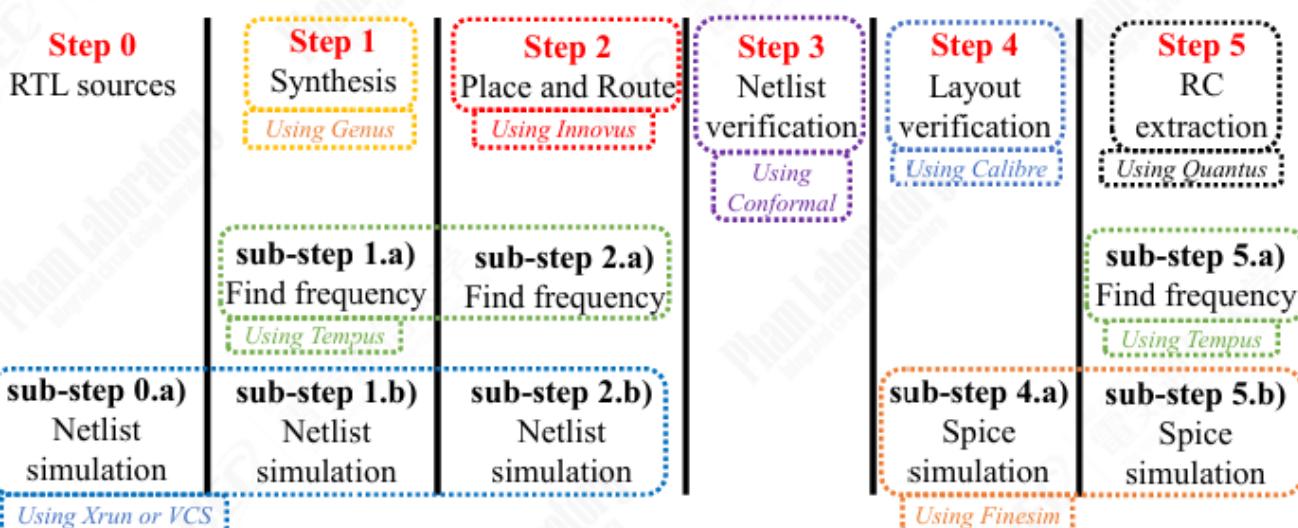
#### Các bước chính trong thiết kế layout (dựa theo slide)

- Bước 1: Put core(s) to a frame**
  - Core là phần mạch xử lý trung tâm (ví dụ: CPU, ALU, hoặc IP block).
  - Core được **đặt vào khung chip (frame)** đã được xác định kích thước và vị trí.
  - Bên ngoài frame sẽ chứa **IO cells** (các khối giao tiếp với ngoại vi) và **pad ring** (vòng tiếp xúc điện).
- Bước 2: Connect to IOs & do filler**
  - Kết nối Core với các IOs (Input/Output cells) như SPI, UART, GPIO, RAM, v.v.
  - Điền các khoảng trống còn lại bằng **filler cells** (các cell rỗng hoặc dummy) để đảm bảo tính liên tục, ổn định về điện và cơ khí.
- Bước 3: Final Fullchip Layout**
  - Tạo layout hoàn chỉnh với đầy đủ: **Vùng Core, Vùng IO, Power grid (mạng cấp nguồn), Metal routing (các lớp dây kết nối), Filler cells**
  - Sẵn sàng cho **Physical Verification** → **Tape-out** để đưa đi sản xuất.

### 3. VLSI Design Flow (11/12) Detail digital flow

The five major steps and their sub steps are visualized as follows.

The tools that used in each step are also listed.

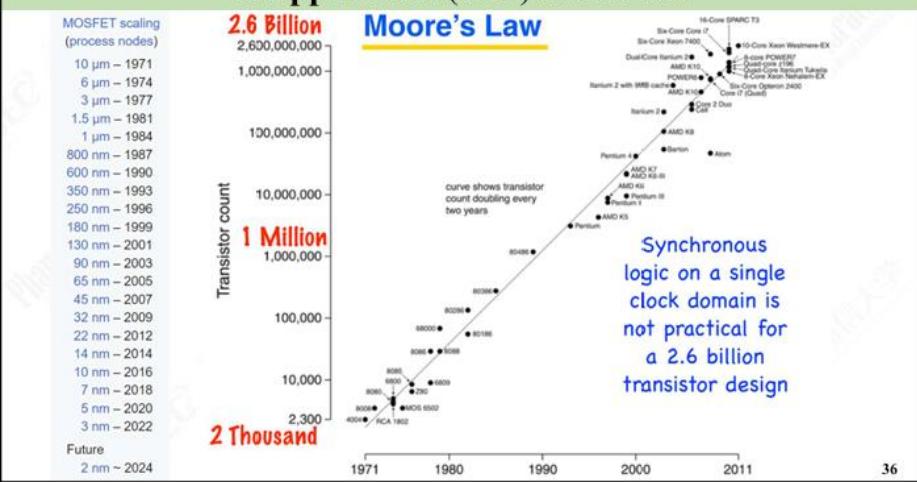


### 3. VLSI Design Flow (12/12) Detail digital flow

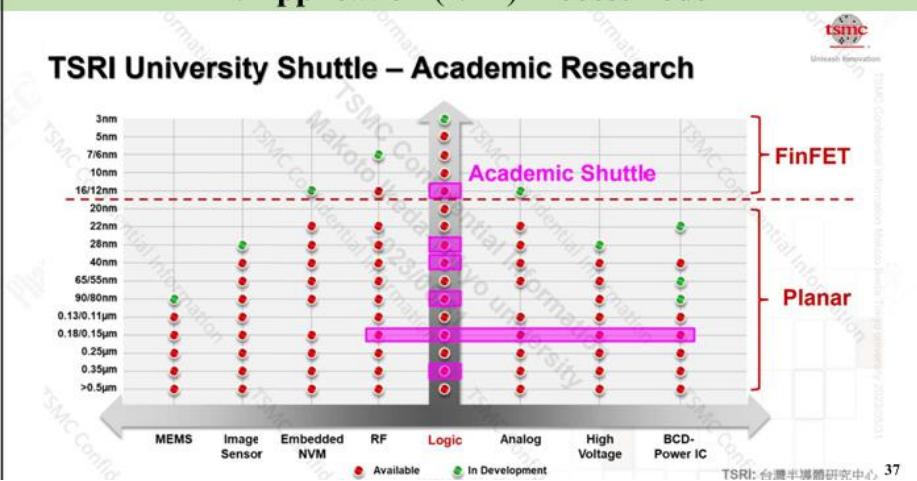
0. **Prepare:** setting up parameters & environment
  - a) *Netlist simulation:* simulate your RTL
1. **Synthesis:** compile your RTL into *synthesis\_net.v* which contains only StdCells (and SRAMs)
  - a) *Find frequency:* find the Fmax of *synthesis\_net.v*
  - b) *Netlist simulation:* simulate the *synthesis\_net.v*
2. **Place and Route:** place the StdCells (& SRAMs) to the floorplan and route the signals. Expected results are the layout, *innovus.gds*, and its netlist equivalent, *innovus.v*
  - a) *Find frequency:* find the Fmax of the *innovus.gds*
  - b) *Netlist simulation:* simulate the *innovus.v*
3. **Netlist verification:** confirm that the *synthesis\_net.v* and *innovus.v* are logically the same
4. **Layout verification:** checks antenna, density, DRC, and grid to make sure that the *innovus.gds* passes foundry requirements. LVS check to make sure that *innovus.gds* is equivalent to *innovus.v*
  - a) *Spice simulation:* convert *innovus.v* to *lvs.src.net*, then run a spice simulation on *lvs.src.net*
5. **RC extraction:** extract parasitic delays of *innovus.gds*. Expected results are *RC.dspf* & *RC.spef*
  - a) *Find frequency:* find the Fmax based on *RC.spef*
  - b) *Spice simulation:* run a spice simulation on *RC.dspf*

## 7.3 Application

### 4. Application (1/12) Moore law



### 4. Application (2/12) Process node



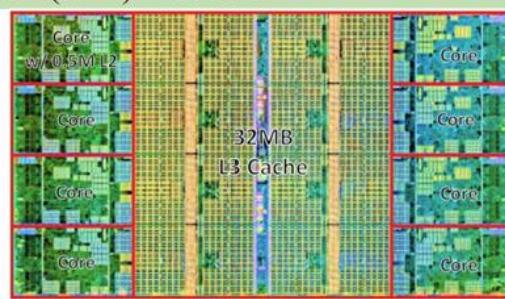
## 4. Application (5/12) Processor

ISSCC 2022 / SESSION 2 / PROCESSORS / 2.7

2.7 Zen3: The AMD 2<sup>nd</sup>-Generation 7nm x86-64 Microprocessor Core

Thomas Burd<sup>1</sup>, Wilson Li<sup>1</sup>, James Pistole<sup>1</sup>, Srividhya Venkataraman<sup>1</sup>, Michael McCabe<sup>1</sup>, Timothy Johnson<sup>1</sup>, James Vint<sup>1</sup>, Thomas Yiu<sup>1</sup>, Mark Wasio<sup>1</sup>, Hon-Hin Wong<sup>1</sup>, Daryl Lieu<sup>1</sup>, Jonathan White<sup>1</sup>, Benjamin Munger<sup>1</sup>, Joshua Lindner<sup>1</sup>, Javin Olson<sup>1</sup>, Steven Bakke<sup>1</sup>, Jesfuh Sniderman<sup>1</sup>, Carson Henrion<sup>1</sup>, Russell Schreiber<sup>1</sup>, Eric Busta<sup>1</sup>, Brett Johnson<sup>1</sup>, Tim Jackson<sup>1</sup>, Aron Miller<sup>1</sup>, Ryan Miller<sup>1</sup>, Matthew Pickett<sup>1</sup>, Aaron Horuchi<sup>1</sup>, Jason Dvorak<sup>1</sup>, Sabesh Balagangadharan<sup>1</sup>, Sajeesh Ammikallian<sup>1</sup>, Pankaj Kumar<sup>1</sup>

<sup>1</sup>AMD, Santa Clara, CA  
<sup>2</sup>AMD, Boxborough, MA  
<sup>3</sup>AMD, Fort Collins, CO  
<sup>4</sup>AMD, Austin, TX  
<sup>5</sup>AMD, Bangalore, India



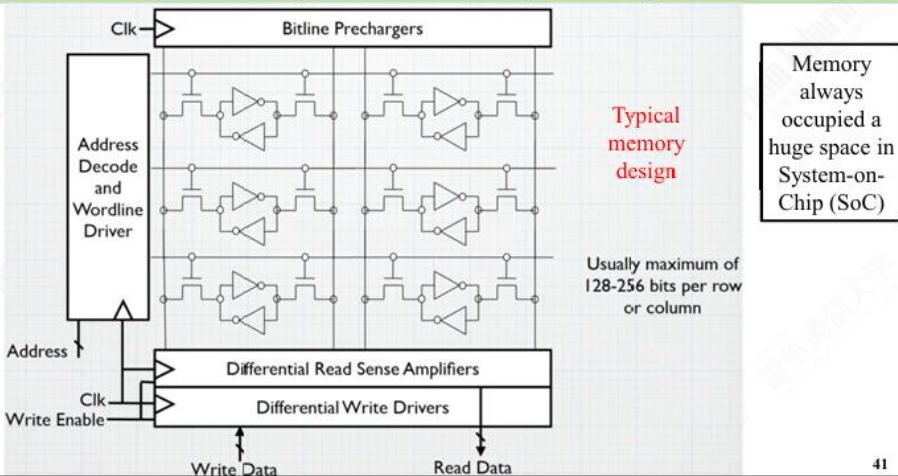
AMD Zen3

(2022)

7nm

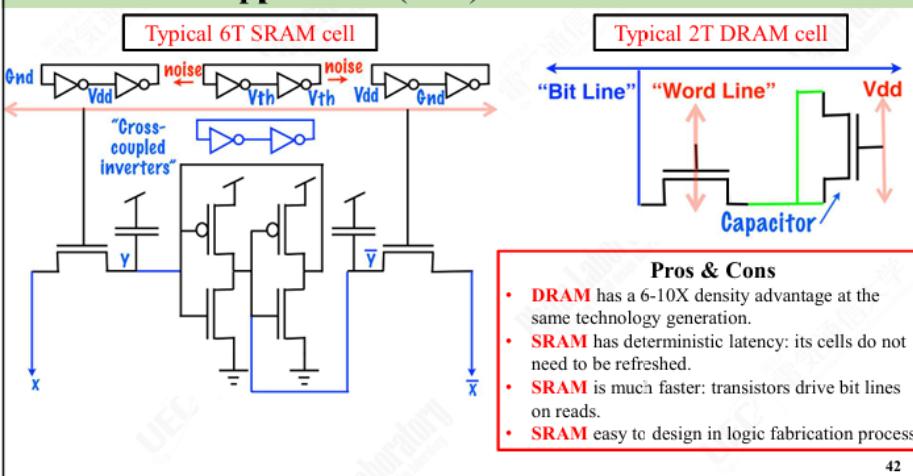
40

## 4. Application (6/12) Memory



41

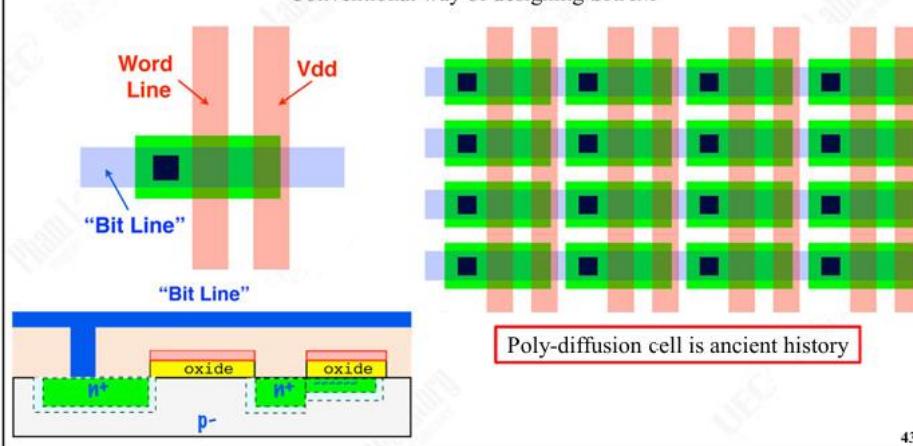
## 4. Application (7/12) SRAM vs. DRAM



42

## 4. Application (8/12) Memory layout

Conventional way of designing SRAM



43

## 4. Application (9/12) Memory layout

Modern design looks something like this

A 31 ns Random Cycle VCAT-Based 4F<sup>2</sup> DRAM  
With Manufacturability and Enhanced Cell Efficiency  
Ki-Whan Song, Jai-Yong Kim, Jai-Min Yoon, Seo Kim, Heejun Kim, Hyun-Woo Chang, Hyungho Kim,  
Kangil Kim, Hwan-Weon Park, Hyun-Chul Kang, Nam-Kyung Tak, Daeha Park, Woo-Sup Kim, Member, IEEE,  
Yong-Tak Lee, Yong-Chul Oh, Gyo-Young Jin, Juhwan Yoo, Donggeun Park, Senior Member, IEEE,  
Kyungsik Oh, Changhyun Kim, Senior Member, IEEE, and Young-Hyun Jun

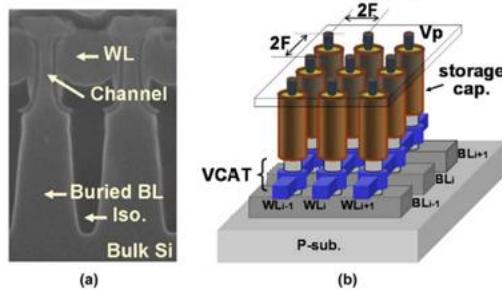
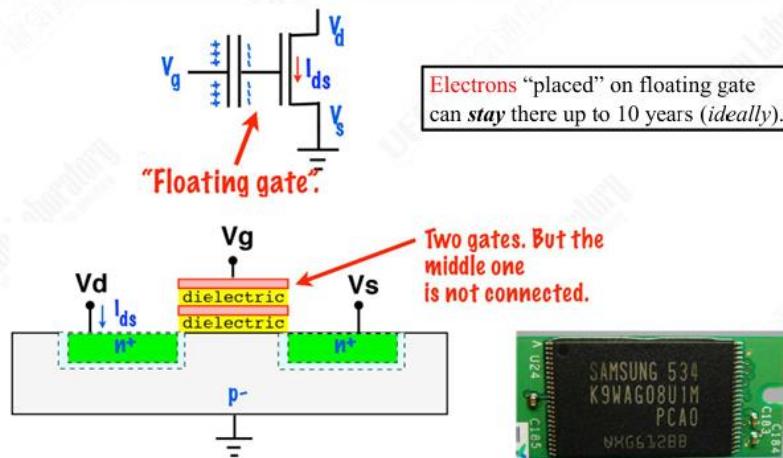


Fig. 1. (a) The cross section of surrounding-gate vertical channel access transistor (VCAT). (b) The schematic diagram of VCAT-based 4F<sup>2</sup> DRAM cell array.

44

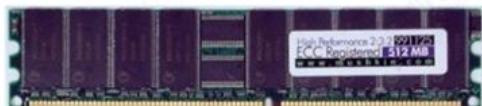
## 4. Application (10/12) Flash (non-volatile memory)



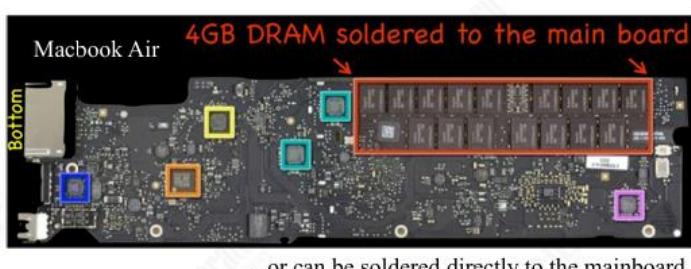
45

## 4. Application (11/12) Memory packaging

Usually, to increase the storage, multiple SRAM/DRAM chips are connected in parallel circuitry.



Many chips form a DIMM RAM,...



46

## 4. Application (12/12) Memory packaging

We can go further though...

Stackable memory dies

SoIC for Low-Temperature, Multi-Layer 3D Memory Integration

M. F. Chen, C. S. Lin, E. B. Liao, W. C. Chou, C. C. Kao, C. C. Hu, C. H. Tsai, C. T. Wang and Douglas Yu  
Integrated Interconnect & Packaging R&D, Taiwan Semiconductor Manufacturing Company, Ltd.  
166, Pei-Wei Ave. II, Hsinchu Science Park, Hsinchu 300, Taiwan, R.O.C.

Phone: 886-3-5616688 Ext 722-3386, Email: mifchen@tsmc.com

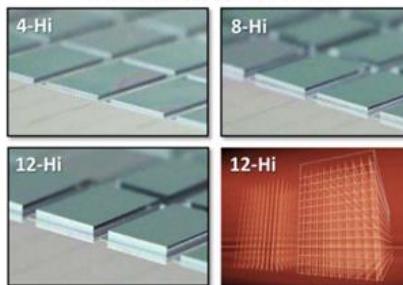


Figure 6. Demonstration of physical images taken from LT-SoIC multi-layer DRAM samples, 4-Hi/ 8-Hi/ 12-Hi, including a 3D-Xray image of 12-Hi DRAM

47



## 7.4 Necessary Libraries

### 7.4.1. Model Files là gì?

- Model files (tệp mô hình) là tập tin chứa thông số mô phỏng của MOSFET, diode hoặc các linh kiện khác.
- Các model này giúp mô phỏng chính xác hoạt động của mạch điện trước khi sản xuất.
- Do tính chất phụ thuộc vào quy trình chế tạo (process-dependent), model files không thể tự tạo mà phải lấy từ foundry (nhà máy sản xuất chip).

### 7.4.2. Tại sao MOSFET Model Files quan trọng?

- ◆ **MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor)** là linh kiện cốt lõi trong **CMOS, bộ khuếch đại, logic số, mạch tương tự (analog), RF, v.v.**
- ◆ Các tham số trong **MOSFET model files** giúp:
  - Xác định **điện áp ngưỡng (Vth)**, dòng điện, độ trễ, tần số hoạt động, v.v.
  - Mô phỏng **hiệu suất thực tế của transistor** trong quá trình thiết kế.
  - Tối ưu **tiêu thụ điện năng, độ nhiễu, độ ổn định, và hiệu suất vi mạch.**

### 7.4.3 Phân loại Model Files theo mục đích

- ◆ Generic Model Files
  - Dùng trong mô phỏng chung
  - Tìm thấy trên internet, nhưng không phản ánh chính xác quy trình chế tạo thực tế
- ◆ Fabrication Model Files
  - Do foundry cung cấp, mô tả thực tế MOSFET sản xuất tại nhà máy
  - Thường bảo mật (confidential) và yêu cầu NDA (Non-Disclosure Agreement)

### 7.4.4 Ngoài MOSFET, còn có những model nào khác?

- Diode Model: Dùng để mô phỏng diode trong các mạch chỉnh lưu, bảo vệ ESD.
- BJT Model (Bipolar Junction Transistor): Dùng trong thiết kế RF, Analog.
- RC & Parasitic Model: Mô phỏng hiệu ứng ký sinh, ảnh hưởng của dây nối.
- Interconnect Model: Dùng trong Place & Route, tối ưu kết nối kim loại.

## BSIM Model Files

- BSIM (Berkeley Short-channel IGFET Model) là phương pháp mô hình hóa MOSFET phổ biến nhất.
- Được phát triển tại UC Berkeley, BSIM mô tả hành vi điện của MOSFET dựa trên các thông số thực tế.
- Đây là tiêu chuẩn công nghiệp và được sử dụng rộng rãi trong cả nghiên cứu lẫn thiết kế vi mạch thương mại.
- BSIM giúp mô phỏng dòng điện, điện áp ngưỡng (Vth), hiệu ứng kênh ngắn, nhiễu, tiêu thụ điện...
- Được dùng trong SPICE simulation (HSPICE, Spectre, Cadence, Synopsys...).
- Giúp tối ưu hiệu suất, tốc độ, và độ ổn định của vi mạch trước khi sản xuất.

### 2. Các phiên bản của BSIM

Phiên bản	Đặc điểm
BSIM3	Tiêu chuẩn đầu tiên được áp dụng rộng rãi trong công nghiệp và học thuật.
BSIM4	Phiên bản mới hơn, áp dụng cho công nghệ từ 130nm đến 20nm.
BSIM cho FinFET (BSIM-CMG, BSIM-IMG)	Dùng cho công nghệ <16nm, khi MOSFET không còn dạng planar mà chuyển sang 3D FinFET.

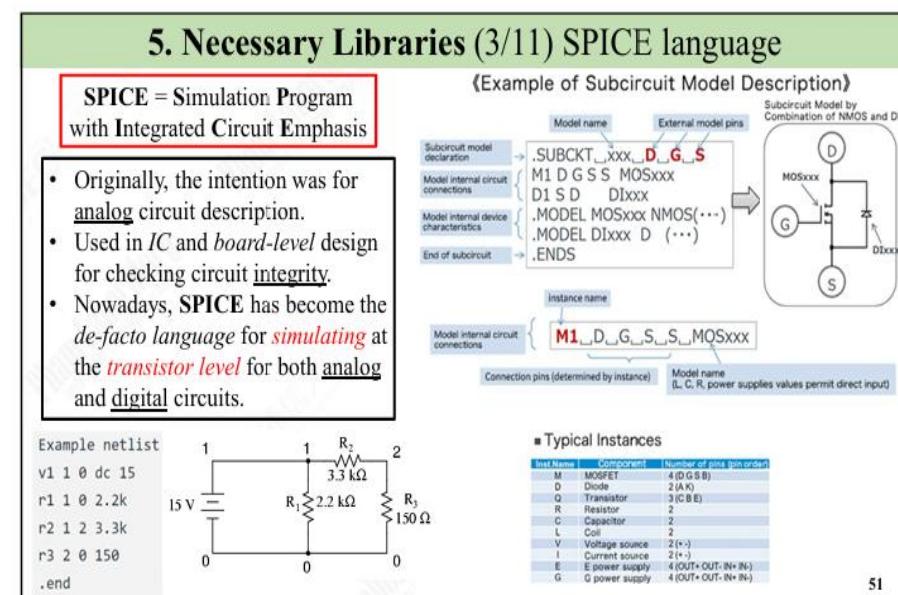
# SPICE

## ✓ 1. SPICE là gì?

- SPICE (Simulation Program with Integrated Circuit Emphasis) là một công cụ mô phỏng mạch điện tử.
- Ban đầu được phát triển để mô tả mạch tương tự (analog circuits) nhưng nay đã trở thành tiêu chuẩn cho cả mạch số (digital circuits).
- Ứng dụng chính:
  - Thiết kế vi mạch IC (Integrated Circuit) và mạch PCB.
  - Kiểm tra tính toàn vẹn của mạch (circuit integrity).
  - Mô phỏng mức transistor cho cả mạch analog & digital.

## ✓ 2. Cấu trúc mô phỏng SPICE

- ◆ Netlist: Danh sách các linh kiện và kết nối của mạch.
- ◆ Mô tả subcircuit (các mô hình transistor, linh kiện tùy chỉnh).
- ◆ Mô phỏng thông số (nguồn, điện trở, dòng điện, v.v.).



## ✓ 3. SPICE trong mô phỏng MOSFET & IC

- ◆ SPICE hỗ trợ mô hình transistor như NMOS, PMOS, BJTs, Diode, Op-Amps,...
- ◆ Mô hình MOSFET được mô tả bằng subcircuit, với các chân D (Drain), G (Gate), S (Source), B (Bulk).
- ◆ Trong thiết kế IC, SPICE giúp kiểm tra độ chính xác của mạch trước khi chế tạo.

## Design Rules (Quy tắc thiết kế)

### 1. Design Rules là gì?

- Design Rule Files là tài liệu quan trọng thứ hai đối với kỹ sư thiết kế mạch (circuit designers).
- Mỗi công nghệ bán dẫn (process node) có một bộ quy tắc thiết kế khác nhau.
- Ngay cả cùng một công nghệ nhưng tại các nhà máy chế tạo khác nhau (foundries) cũng có thể có quy tắc khác nhau.
- Design Rules phụ thuộc vào quy trình công nghệ (process-dependent).

### 2. Design Rules bao gồm những gì?

Design Rules xác định nhiều thông số quan trọng như:

- ◆ Chiều rộng tối thiểu của dây dẫn.
- ◆ Chiều dài và diện tích tối thiểu của linh kiện.
- ◆ Khoảng cách tối thiểu giữa các dây dẫn song song.
- ◆ Các giới hạn vật lý để đảm bảo có thể chế tạo thành công.

### 3. Kiểm tra Design Rule (DRC - Design Rule Check)

Trước khi gửi layout đến nhà máy chế tạo (foundry), phải kiểm tra và vượt qua tất cả các Design Rule Checks (DRCs). Nếu layout không vượt qua DRC, việc chế tạo có thể gặp rủi ro cao (tape-out at your own risk) hoặc không thể sản xuất được.

## 5. Necessary Libraries (4/11) Design rules

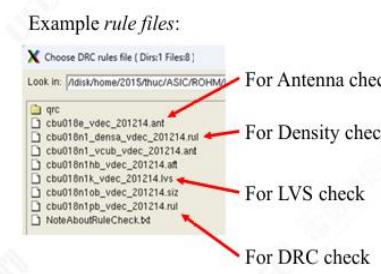
The second most important files for circuit designers are design rule files.

- Each process node will have a different set of design rules. Even the same process at different foundries can have slightly different rules.
- Therefore, design rules are heavily process-dependent. ⇒ So, just like model files, circuit designers cannot create design rules by themselves. Those design rules must be given to them by a foundry.
- Design rules define many things, such as minimum width, maximum length, minimum area, minimum distance between parallel wires, and much more.
- A layout must have passed all the Design Rule Checks (DRCs) before submitting to a foundry.
- A layout that failed DRCs is a “tape-out at your own risk” or even impossible to fabricate.

## 5. Necessary Libraries (5/11) Design rules

Normally, design rules are delivered with:

- One PDF file, called the Design Rule Manual (DRM), for us to read.
- Several rule files for a verification tool (mostly Calibre) to check.



Digital Libraries (Thư viện số)

## 1. Thư viện số là gì?

- ◆ **Digital Libraries** là tập hợp các tệp cần thiết để thiết kế mạch số (**digital circuits**).
    - ◆ Đối với **analog circuits**, chỉ cần **Model files** và **Design Rule files** là đủ.
    - ◆ Tuy nhiên, **digital circuit designers** cần thêm ít nhất **Standard Cell (StdCell) library**.

## 2. Thành phần chính của Digital Libraries

## ✓ Standard Cell Library (StdCell)

- Chứa các công logic cơ bản như AND, OR, NAND, NOR, XOR,...
  - Gồm cả các phần tử lưu trữ (storage elements) như Flip-Flop, Latch.
  - Dùng để tổng hợp và ánh xạ thiết kế từ mã Verilog sang layout.

✓ SRAM Library (tùy vào thiết kế)

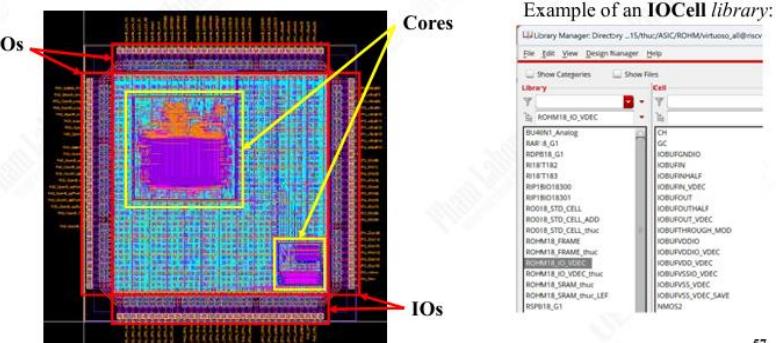
- Nếu thiết kế liên quan đến System-on-Chip (SoC) hoặc cần bộ nhớ, thư viện SRAM cũng rất quan trọng.
  - Cung cấp các khối bộ nhớ có sẵn để tích hợp vào chip.

### 3. Tại sao Digital Libraries quan trọng?

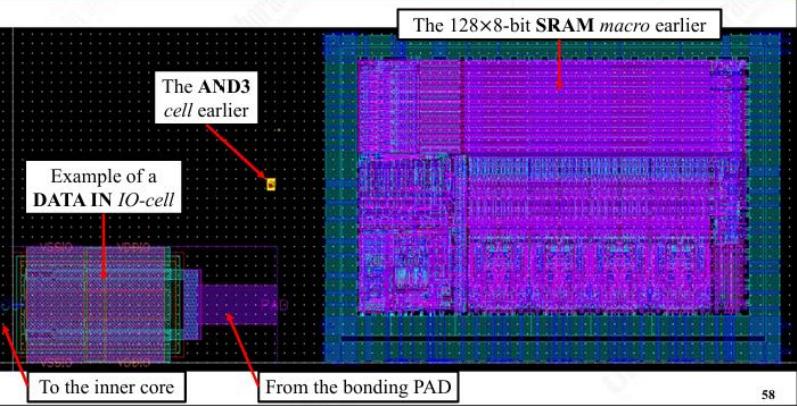
- ◆ Giúp tự động hóa thiết kế mạch số bằng cách ánh xạ các mô tả logic sang các cell có sẵn.
  - ◆ Tăng tốc độ thiết kế và đảm bảo tuân thủ quy tắc công nghệ của nhà sản xuất.
  - ◆ Giúp đơn giản hóa quá trình **Logic Synthesis, Place-and-Route, và Physical Verification**.

## 5. Necessary Libraries (9/11) IOCell library

Finally, to complete a **FullChip integration**, an **IOCell library** is also required.



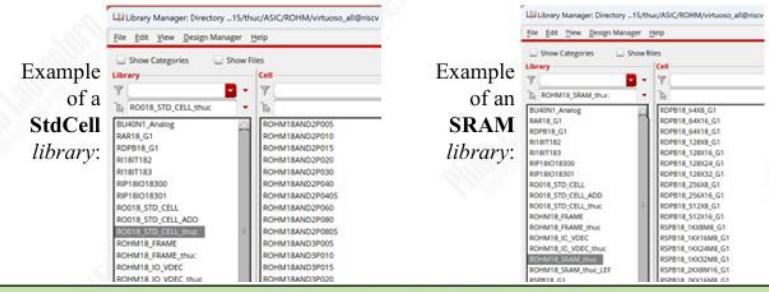
## 5. Necessary Libraries (10/11) IOCCell library



**Model files** and **design rule files** are enough for an *analog circuit designer*.

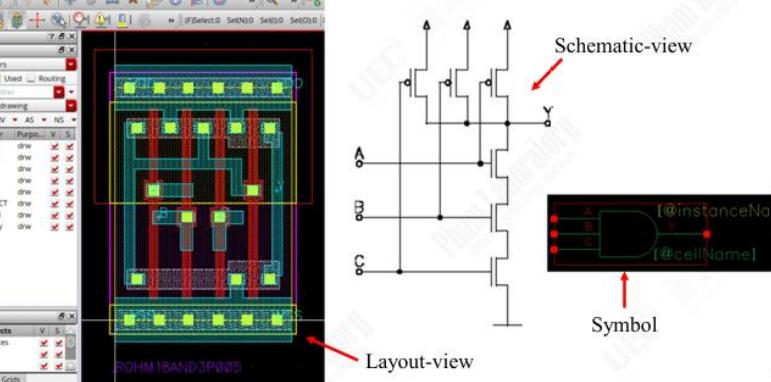
It's not the case for *digital circuit designers*

As mentioned earlier, **digital designs** need at least a **Standard Cell (*StdCell*) library**. Sometimes, an **SRAM library** is also required, especially in System-on-Chip (*SoC*).



## 5. Necessary Libraries (7/11) StdCell library

### Example of an **AND3** cell in a StdCell library.



## 5. Necessary Libraries (8/11) SRAM library

## Example of some **SRAM** macros in a **SRAM** library.

**SRAM** macros are not basic functions. If the **SRAM library** doesn't give you the **SRAM size** you want, you have to improvise by connecting smaller **macros**.



IOCell Library

- ◆ **IOCell Library** là thư viện chứa các cell phục vụ cho việc giao tiếp giữa chip và thế giới bên ngoài.
  - ◆ Các thiết kế mạch số không chỉ có logic bên trong mà còn cần các cổng vào/ra (**IO Cells**) để kết nối với board mạch hoặc các thành phần khác.
  - ◆ IOCell Library thường chứa:
    - **Input cells** (nhận tín hiệu từ bên ngoài vào chip).
    - **Output cells** (truyền tín hiệu từ chip ra ngoài).
    - **Bi-directional cells** (có thể hoạt động cả hai chiều).
    - **Power/Ground pads** (cung cấp nguồn điện cho chip).
    - **Electrostatic Discharge (ESD) protection** (bảo vệ chip khỏi sốc tĩnh điện).

## 5. Necessary Libraries (11/11) Summary

- **Model files** and **design rule files** are made by a *foundry*.  
They are not supposed to be made by *circuit designers*.
- A *foundry* also usually provides **StdCell library**, **SRAM library**, and **IOCell library** to *circuit designers*, but not always.  
That is actually not their job, and *circuit designers* can create their own libraries based on **model files** and **design rules**.
- **Design rules** define many layout rules.  
**Design rule files** are necessary for checking *DRC*, *antenna*, *density*, *LVS*, etc.  
A layout must *pass all the design rules* before submitting to a foundry.
- **Model files** are necessary for **SPICE-level simulation**.  
They need to describe the physical properties of the devices (*MOSFET*, *diode*, etc.) as close to reality as possible to minimize the *simulation gap*.
- **StdCell** and **SRAM libraries** are needed for digital designs, and an **IOCell library** is needed for **FullChip integration**.

59

## 6. EDA Tools (2/4) Industry standard tools

The list of the most common core tools used in the industry.

	Synopsys	Cadence
<i>Synthesis</i>	Design Compiler ( <i>DC</i> )	Genus
<i>PnR</i>	IC Compiler 2 ( <i>ICC2</i> )	Innovus
<i>Netlist verification</i>	Formality	Conformal
<i>Layout verification</i>	Calibre	<b>Mentor Graphic</b>
<i>RC extract</i>	StarRC	Quantus
<i>HDL simulation</i>	VCS	Xrun
<i>Fmax checking</i>	PrimeTime	Tempus
<i>Spice simulation</i>	HSPICE / Finesim	
<i>Layout editor</i>		Virtuoso

62

