Tiêu chí	RISC	CISC
Lịch sử phát triển	• Ra đời vào những năm 1980 để cải thiện hiệu suất và tiết kiệm năng lượng. • Tập trung vào đơn giản hóa phần cứng và tăng tốc độ thực thi lệnh.	• CISC (**COMPLEX INSTRUCTION SET COMPUTER**) là hai kiến trúc vi xử lý phổ biến trong lĩnh vực điện toán. • Ra đời từ những năm 1970-1980. • Được thiết kế để giảm bớt gánh nặng lập trình bằng cách cung cấp các lệnh phức tạp thực hiện nhiều tác vụ trong một lệnh. • Phù hợp với thời kỳ bộ nhớ và tài nguyên phần cứng còn hạn chế.
Nguyên lý hoạt động	• Tập lệnh tối giản, chỉ bao gồm các lệnh cơ bản như load, store, add, subtract, v.v. • Mỗi lệnh thường hoàn thành trong **một chu kỳ xung nhịp**. • Sử dụng pipeline hiệu quả hơn nhờ các lệnh đồng nhất về độ dài và độ phức tạp.	• Tập lệnh đa dạng và phức tạp, có thể thực hiện các phép toán cấp cao (như nhân, chia, xử lý chuỗi) trong một lệnh duy nhất. • Hỗ trợ chế độ định địa chỉ phức tạp (complex addressing modes), giúp xử lý dữ liệu linh hoạt từ bộ nhớ. • Một lệnh cisc có thể cần nhiều chu kỳ xung nhịp để thực thi do độ phức tạp.
Định nghĩa	Tập lệnh tối giản, đơn giản hóa xử lý.	Tập lệnh phức tạp, nhiều chức năng.
Tập lệnh	Số lượng lệnh ít, thường chỉ vài chục lệnh.	Số lượng lệnh nhiều, phức tạp.
Độ phức tạp của lệnh	Mỗi lệnh chỉ thực hiện một tác vụ đơn giản.	Một lệnh có thể thực hiện nhiều tác vụ.
Thời gian thực thi	Thời gian thực thi lệnh ngắn (thường 1 chu kỳ).	Thời gian thực thi lệnh dài (nhiều chu kỳ).
Phần cứng	Đơn giản hơn, dễ tối ưu hóa tốc độ.	Phức tạp hơn, yêu cầu nhiều logic xử lý.
Bộ nhớ và mã lệnh	Mã lệnh ngắn gọn, tiết kiệm bộ nhớ hơn.	Mã lệnh dài hơn, tốn nhiều bộ nhớ.
Pipeline	Tối ưu hóa pipeline tốt hơn nhờ lệnh đơn giản.	Hạn chế do độ phức tạp của lệnh.
Hiệu quả năng lượng	Hiệu quả năng lượng cao hơn.	Thường tiêu tốn nhiều năng lượng hơn.
Ví dụ kiến trúc	ARM, MIPS, PowerPC, PIC	Intel x86, AMD.

CHAPTER 1: ADVANCED SEQUENTIAL LOGIC

- **Latches** are level-sensitive since they respond to input changes *during clock width* → It is difficult to work.(**Latches** nhạy theo mức (*level-sensitive*), vì chúng phản hồi với các thay đổi đầu vào trong khoảng thời gian tín hiệu đồng hồ hoạt động (*clock width*) → Điều này khiến chúng khó sử dụng.)
- **Flip-Flops** respond to input changes only *during the change in clock signal edge*. (Các **flip-flop (FF)** chỉ phản hồi với các thay đổi đầu vào trong lúc biên của tín hiệu đồng hồ thay đổi (*clock edge*).)
- **FFs** are easy to work with though more expensive than latches. (FFs dễ sử dụng hơn, mặc dù chi phí cao hơn so với Latches)
- Two styles of flip-flops are available.
  1. **master-slave (MS)**
  2. **edge-triggered (ET)**

FF name	Characteristic Table	Characteristic equation	Excitation Table
SR	S R Qnext 0 0 Q 0 1 0 1 0 1 1 1 NA	Q(next) = S + R'Q SR = 0	Q Qnext S R 0 0 0 X 0 1 1 0 1 0 0 1 1 1 X 0
JK	J K Qnext 0 0 Q 0 1 0 1 0 1 1 1 Q'	Q(next) = JQ' + K'Q	Q Qnext J K 0 0 0 X 0 1 1 X 1 0 X 1 1 1 X 0
D	D Q(next) 0 0 1 1	Q(next) = D	Q Q(next) D 0 0 0 0 1 1 1 0 0 1 1 1
T	T Q(next) 0 Q 1 Q'	Q(next) = TQ' + T'Q	Q Q(next) T 0 0 0 0 1 1 1 0 1 1 1 0

Analysis procedure for sequential circuits (Quy trình phân tích mạch tuần tự)
Derive excitation equations (Xác định phương trình kích thích)
Derive next-state and output equations (Xác định phương trình trạng thái kế tiếp và phương trình đầu ra)
Generate next-state and output tables (Tạo bảng trạng thái kế tiếp và bảng đầu ra)
Generate state diagram (Tạo sơ đồ trạng thái)
Develop timing diagram (Phát triển sơ đồ thời gian)
Simulate logic schematic (Mô phỏng sơ đồ logic)

Finite State Machine Model
- ***f : S x I → S***
- **h: S x I → O (Mealy-type)**
  S → O (Moore-type)

Synthesis procedure for sequential logic (Quy trình tổng hợp logic tuần tự)
Design description or timing diagram
Develop state diagram (Phát triển sơ đồ trạng thái)
Generate next-state and output tables (Tạo bảng trạng thái kế tiếp và đầu ra)
Minimize states (Tối thiểu hóa các trạng thái)
Encode inputs, states, outputs (Mã hóa các đầu vào, trạng thái, và đầu ra)
Derive next-state and output equations (Xác định phương trình trạng thái kế tiếp và phương trình đầu ra)
Choose memory elements (Chọn phần tử nhớ)
Derive excitation equations (Xác định phương trình kích thích)
Optimize logic implementation (Tối ưu hóa việc triển khai logic)
Derive logic schematic and timing diagrams (Tạo sơ đồ logic và sơ đồ thời gian)
Simulate logic schematic (Mô phỏng sơ đồ logic)
Verify functionality and timing (Xác minh chức năng và thời gian)

STATE MINIMIZATION (tối ưu hóa trạng thái)

Tối ưu hóa trạng thái giảm số lượng trạng thái, do đó cũng giảm số lượng flip-flop cần thiết để triển khai mạch.
Tối ưu hóa trạng thái dựa trên khái niệm **tương đương hành vi**, trong đó hai trạng thái được coi là tương đương nếu chúng tạo ra cùng một chuỗi trạng thái tiếp theo (*next-state*) và các ký hiệu đầu ra (*output symbols*) cho mọi chuỗi ký hiệu đầu vào (*input symbols*).
Cụ thể hơn, hai trạng thái sjs_jsj và sks_ksk trong một máy trạng thái hữu hạn (FSM) được coi là tương đương nếu thỏa mãn hai điều kiện sau:

- **Điều kiện 1**: Cả hai trạng thái sjs_jsj và sks_ksk tạo ra cùng một ký hiệu đầu ra cho mọi ký hiệu đầu vào iii: nghĩa là h(sj,i)=h(sk,i)h(s_j, i) = h(s_k, i)h(sj ,i)=h(sk,i).
- **Điều kiện 2**: Cả hai trạng thái có trạng thái tiếp theo tương đương cho mọi ký hiệu đầu vào iii: nghĩa là f(sj,i)=f(sk,i)f(s_j, i) = f(s_k, i)f(sj,i)=f(sk,i).

Quy trình tối ưu hóa:
1. Phân chia các trạng thái thành các lớp tương đương.
2. Xây dựng FSM mới với mỗi trạng thái đại diện cho một lớp tương đương

State reduction for modulo-3 counter
State reduction with implication table
State encoding

TRONG STATE ENCODING CÓ:

Three different heuristics (3 phương thức khác nhau)
Thay đổi số bit tối thiểu (minimum-bit-change): Chiến lược thay đổi số bit tối thiểu gán mã cho các trạng thái sao cho tổng số lần thay đổi bit cho tất cả các chuyển đổi trạng thái được giảm thiểu.
Ưu tiên trạng thái liền kề (prioritized adjacency): Chiến lược ưu tiên trạng thái liền kề gán mã hóa liền kề cho tất cả các trạng thái có cùng nguồn, cùng đích hoặc cùng đầu ra.
- Ưu tiên cao nhất được dành cho các trạng thái có cùng trạng thái tiếp theo (next state) với một giá trị đầu vào nhất định → mã hóa của trạng thái tiếp theo giống nhau sẽ xuất hiện ở các ô liền kề trên bản đồ Karnaugh.
- Ưu tiên thứ hai được dành cho các trạng thái tiếp theo của cùng một trạng thái → các trạng thái tiếp theo cũng có thể xuất hiện liền kề trên bản đồ Karnaugh.
- Ưu tiên thứ ba được dành cho các trạng thái có cùng giá trị đầu ra với cùng các giá trị đầu vào → chúng có thể liền kề trên bản đồ đầu ra

Mã hóa One-Hot (one-hot encoding):
- Mã hóa Hot-one là một phương pháp mã hóa dư thừa, trong đó mỗi trạng thái được gán cho một flip-flop có giá trị 1, trong khi các flip-flop khác có giá trị 0.
- Mã hóa Hot-one quá tốn kém đối với các FSM có số lượng trạng thái lớn. Phương pháp này chỉ được sử dụng trong các FSM nhỏ

CHAPTER 2 (PART 1): STORAGE COMPONENT

Các thành phần lưu trữ lưu trữ dữ liệu và thực hiện một số thao tác đơn giản.

CHAPTER 2 (PART 2): STORAGE COMPONENT

Các thành phần tổ hợp (combinatorial components) và lưu trữ được sử dụng để xây dựng:

Các thành phần lưu trữ bao gồm:

Thanh ghi (registers)

- Các thanh ghi là phần mở rộng theo từng bit của các flip-flop.
- Thanh ghi lưu trữ một từ dữ liệu.

Registers with asynchronous set and reset

- Thiết lập và đặt lại không đồng bộ độc lập với tín hiệu đồng hồ.
- Các đầu vào không đồng bộ được sử dụng để khởi tạo thanh ghi.

Register with parallel load

- Thanh ghi tải song song có thể giữ dữ liệu vô thời hạn.
- Nó cũng có thể tải dữ liệu mới khi tín hiệu tải là 1.

Bộ đếm (counters) bao gồm:

- 4-bit binary counter
- 4-bit up/down binary counter
- 4-bit up/down binary counter with parallel load
- BCD counters
- Asynchronous counters (bộ đếm bất đồng bộ):
  - Mỗi FF trong bộ đếm đồng bộ thay đổi đầu ra của nó cùng một lúc.
  - Đầu ra của FF trong bộ đếm không đồng bộ thay đổi giá trị tại các thời điểm khác nhau.
  - Ưu điểm của bộ đếm không đồng bộ là đơn giản và chi phí thấp (ít cổng hơn).
  - Điểm yếu của bộ đếm không đồng bộ là độ trễ dài hơn so với bộ đếm đồng bộ.

Tệp thanh ghi (register files):

- Được sử dụng làm nơi lưu trữ tạm thời nhanh chóng

Register-file with 1 write port and 2 read ports

- Được sử dụng để đọc hai toán hạng và ghi một kết quả trong mỗi chu kỳ đồng hồ

Các đường dữ liệu (datapaths)

Bộ điều khiển (controllers)

Đây là các hệ thống chính của các bộ xử lý hiện đại và các vi mạch khác.

DESIGN MODEL

CONTROL UNIT	DATAPATH
CONTROL INPUTS	DATAPATH INPUTS
CONTROL OUTPUTS (DONE signal)	DATAPATH OUTPUTS
NEXT-STATE LOGIC	
STATE REGISTER (CURRENT STATE)	
OUTPUT LOGIC (CONTROL signals)	STATUS SIGNAL (Tín hiệu trạng thái)

Static RAM (SRAM hay RAM tĩnh) là một loại bộ nhớ sử dụng công nghệ bán dẫn. Từ "tĩnh" nghĩa là bộ nhớ vẫn lưu dữ liệu nếu có điện, không như RAM động cần được nạp lại thường xuyên. Không nên nhầm RAM tĩnh với bộ nhớ chỉ đọc và bộ nhớ flash vì RAM tĩnh chỉ lưu được dữ liệu khi có điện.

Dynamic RAM (DRAM hay RAM động) là một loại bộ nhớ truy cập ngẫu nhiên lưu mỗi bit dữ liệu trong một tụ điện riêng biệt trên một mạch tích hợp. Vì các tụ điện bị rò điện tích nên thông tin sẽ bị mất dần trừ khi dữ liệu được nạp lại đều đặn. Đây là điểm khác biệt so với RAM tĩnh. Ưu điểm của DRAM là có cấu trúc đơn giản: chỉ cần một transistor và một tụ điện cho mỗi bit trong khi cần sáu transistor đối với SRAM. Điều này cho phép DRAM lưu trữ với mật độ cao. Vì DRAM mất dữ liệu khi không có điện nên nó thuộc loại thiết bị nhớ tạm thời.

Ngăn xếp (stacks) và Hàng đợi (queues)

- Mục đích: Stack (LIFO) hay Queue (FIFO) được sử dụng khi có nhiều dữ liệu cần CPU xử lý cùng 1 lúc, nhưng CPU chỉ xử lý được trên 1 data tại 1 thời điểm
- → Để tránh việc mất data đến sau và xử lý theo trình tự, cần có 1 nơi chứa dữ liệu tạm, sau đó CPU sẽ đọc ra và xử lý
- Cách hoạt động của stack: hoạt động giống LIFO (Last In, First Out), dữ liệu ghi vào sau cùng sẽ được đọc ra đầu tiên

---

CHAPTER 3 (P1): Register Transfer Specification & Design

Thiết kế Chuyển Dữ Liệu Thanh Ghi (Register Transfer - RT)

- **Mô hình FSMD (Finite State Machine with Data Path)**: Mô hình thiết kế hệ thống kỹ thuật số kết hợp giữa máy trạng thái hữu hạn (FSM) và đường dẫn dữ liệu.
- **Đặc tả RT (Register Transfer Specification):** Mô tả hành vi của hệ thống bằng cách chỉ định các thao tác chuyển dữ liệu giữa các thanh ghi và các phép toán thực hiện trên dữ liệu.
- **Bảng hành động tĩnh (Static-action tables):** Một phương pháp biểu diễn đặc tả RT bằng bảng. Mỗi hàng trong bảng đại diện cho một trạng thái của FSM. Các cột trong bảng mô tả các thao tác chuyển dữ liệu và phép toán được thực hiện trong mỗi trạng thái.
- **Biểu đồ ASM (Algorithmic State Machine charts): Một phương pháp biểu diễn trực quan đặc tả** RT bằng biểu đồ. Biểu đồ ASM kết hợp giữa biểu đồ trạng thái và các khối mô tả các thao tác chuyển dữ liệu và phép toán.

Quy trình tổng hợp từ đặc tả thời gian thực (RT)

ASM RULES

Quy tắc 1: Biểu đồ phải xác định một trạng thái tiếp theo duy nhất cho mỗi trạng thái và mỗi tập hợp điều kiện.

Quy tắc 2: Mọi đường dẫn được xác định bởi mạng lưới các hộp quyết định phải dẫn đến một trạng thái

---

CHAPTER 3 (P2): Register Transfer Specification & Design

Tối ưu hóa thiết kế thông qua:

Register sharing (Variable merging)

Tập hợp các biến có thời gian tồn tại không chồng lấn vào một thanh ghi

- Các biến không được sử dụng (ghi hoặc đọc) trong cùng một trạng thái được gọi là các biến có thời gian tồn tại không chồng lấn.
- Thời gian tồn tại của một biến: Tập hợp các trạng thái mà biến đó tồn tại. Nó bắt đầu từ trạng thái đầu tiên mà biến được ghi (trạng thái ghi) cho đến trạng thái cuối cùng mà biến được sử dụng (trạng thái đọc), bao gồm tất cả các trạng thái giữa trạng thái đầu tiên và trạng thái cuối cùng.
- Tập hợp các biến làm giảm số lượng thanh ghi cần thiết trong thiết kế →Giảm diện tích và chi phí.
- **Hai thuật toán:**
  - **left-edge**: Thuật toán đơn giản, chọn một biến bất kỳ làm đại diện cho nhóm và thêm các biến có thời gian tồn tại không chồng lấn với biến đại diện vào nhóm đó.
  - **graph-partitioning**: Thuật toán phức tạp hơn, xây dựng một đồ thị biểu diễn mối quan hệ giữa các biến và sử dụng các thuật toán phân hoạch đồ thị để tìm các nhóm biến tối ưu.

Functional-unit sharing (operator merging)

Tập hợp các hoạt động không đồng thời

- Mỗi nhóm chia sẻ một đơn vị chức năng.
- Chia sẻ giảm số lượng đơn vị chức năng.
- Tập hợp ưu tiên bằng cách giảm kết nối.
- Thuật toán phân cụm được sử dụng để tập hợp.

Bus sharing (connection merging)

Tập hợp các kết nối không được sử dụng đồng thời

- Mỗi nhóm tạo thành một bus.
- Kết hợp kết nối giảm số lượng dây dẫn.
- Thuật toán phân cụm được minh họa.

Register merging

Tập hợp các thanh ghi có truy cập không chồng lấn

- Mỗi nhóm được gán cho một tập tin thanh ghi.
- Tập hợp thanh ghi giảm số lượng cổng và do đó giảm số lượng bus.
- Minh họa với thuật toán phân cụm.

CHAPTER 3 (P3): Register Transfer Specification & Design

Chaining and multicycling (Liên kết và đa chu kỳ)

- Chaining cho phép thực hiện tuần tự hai hoặc nhiều thao tác trong mỗi trạng thái.
- Chaining giảm số lượng trạng thái và tăng hiệu suất.
- Multicycling cho phép một thao tác được thực hiện trong hai hoặc nhiều chu kỳ đồng hồ.
- Multicycling giảm kích thước của các đơn vị chức năng
- Chaining và multicycling được sử dụng trên các đường dẫn không quan trọng (noncritical paths) để cải thiện việc sử dụng tài nguyên và hiệu suất.

Các đường dẫn không quan trọng là những đường dẫn trong hệ thống không ảnh hưởng đến thời gian thực thi tổng thể của hệ thống. Sử dụng chaining và multicycling trên các đường dẫn này giúp tối ưu hóa việc sử dụng tài nguyên và cải thiện hiệu suất tổng thể của hệ thống.

Pipilining

Kỹ thuật đường ống (Pipelining) cải thiện hiệu suất với chi phí bổ sung rất nhỏ.

Pipelining chia nhỏ tài nguyên thành các giai đoạn và sử dụng đồng thời tất cả các giai đoạn cho các dữ liệu khác nhau (nguyên lý dây chuyền lắp ráp).

Nguyên lý của Pipelining hoạt động trên nhiều cấp độ:

- **Units pipelining:** Chia nhỏ các đơn vị chức năng (như bộ cộng, bộ nhân) thành nhiều giai đoạn con để tăng tốc độ thực hiện phép toán.
- **Control pipelining:** Chia nhỏ quá trình điều khiển của hệ thống thành các giai đoạn con, cho phép xử lý nhiều lệnh cùng một lúc**.**
- **Datapath pipelining:** Chia nhỏ đường dẫn dữ liệu thành các giai đoạn con, cho phép truyền dữ liệu đồng thời giữa các thành phần khác nhau của hệ thống.

Scheduling

Mô tả RT như biểu đồ ASM chỉ định các thao tác dữ liệu trong mỗi trạng thái.

Sơ đồ khối hoặc ngôn ngữ lập trình không có khái niệm trạng thái, mà chỉ chỉ định thứ tự thực hiện các thao tác.

Lập lịch (Scheduling) chuyển đổi sơ đồ khối hoặc chương trình với mô tả RT.

Hai loại lập lịch:

- Lập lịch bị ràng buộc bởi tài nguyên (tài nguyên cho trước, tối thiểu hóa thời gian)
- Lập lịch bị ràng buộc bởi thời gian (thời gian cho trước, tối thiểu hóa tài nguyên)