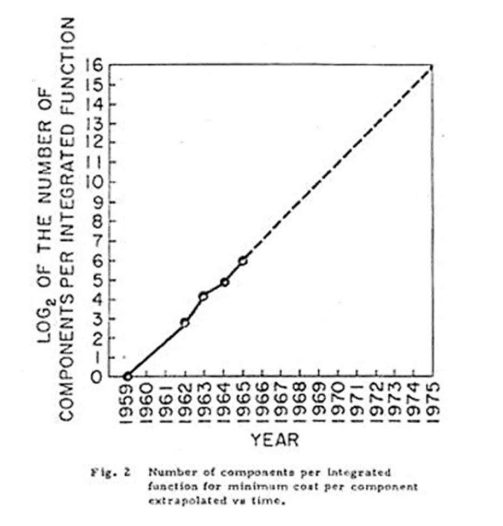
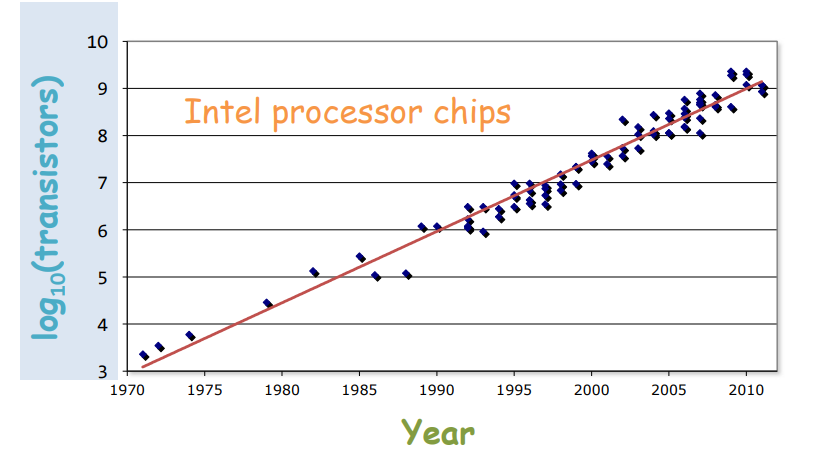
**Kiến Trúc bộ tăng tốc**

1. **Moore’s Law**

* **Định nghĩa**: Hiệu suất CPU và chip sẽ tăng gấp đôi mỗi hai năm (hoặc 12-18 tháng).
* **Các yếu tố liên quan**: Bao gồm sự cải thiện về tốc độ, kích thước transistor, và số lượng thiết bị trên chip, đồng thời đạt được hiệu quả kinh tế tối ưu.



Đồ thị biểu diễn: LOG₂ của số lượng thành phần trên mỗi mạch tích hợp tính theo năm.



Biểu đồ:Số lượng transitor trên chip theo năm.

A graph showing the growth of a number of individuals

Description automatically generated with medium confidence

Biểu đồ: Hiệu suất chip tính theo năm.

* **Năng lượng tiêu thụ (Joules):** E=α×C×

→ Năng lượng tiêu thụ phụ thuộc vào hệ số chuyển mạch (α), điện dung (C), và bình phương điện áp (V).

* **Công suất tiêu thụ (Watts - Joules/giây):** P=α×C×x f

→ Công suất tiêu thụ tăng khi tần số (f) tăng, vì tín hiệu chuyển đổi nhanh hơn.

**Các thành phần liên quan:**

* Hệ số hoạt động (α): Dao động từ 0 đến 1, thể hiện mức độ chuyển mạch của tín hiệu.
* Điện dung tải (C): Lượng điện tích cần thiết để sạc/xả mỗi lần chuyển đổi trạng thái.
* Điện áp nguồn (V): Yếu tố quan trọng nhất, vì năng lượng tiêu thụ tăng theo
* Tần số hoạt động (f): Tăng f giúp tăng tốc xử lý nhưng cũng làm tăng công suất tiêu thụ.

A diagram of a glass of water

Description automatically generated

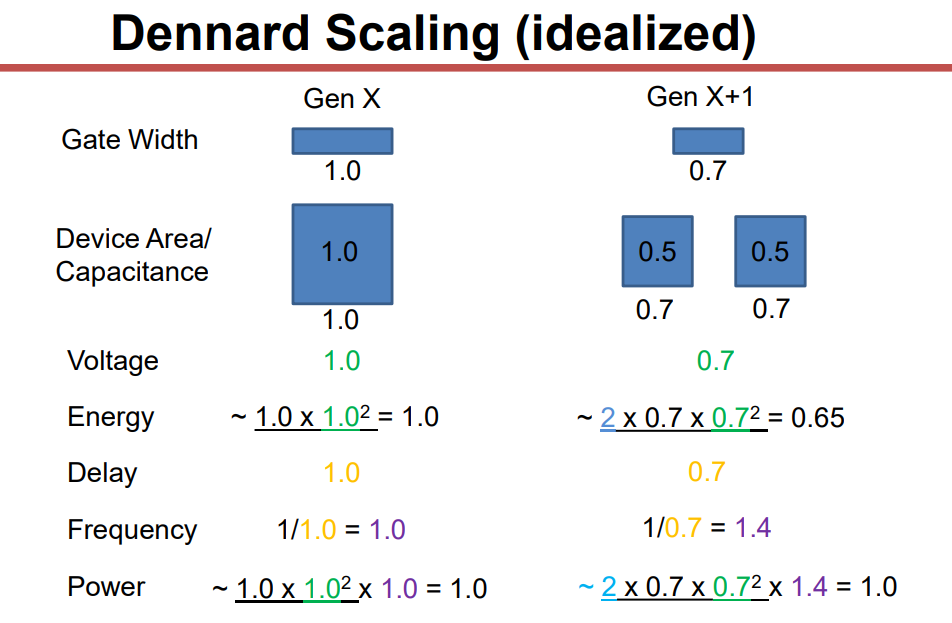
* **Diện tích ngang của xô** → **Điện dung (C)**: Quyết định khả năng chứa nước, tương tự như điện dung quyết định khả năng lưu trữ điện tích.
* **Độ sâu của nước** → **Hiệu điện thế (V)**: Nước càng sâu, năng lượng càng cao, giống như điện áp cao thì năng lượng điện tích trữ càng lớn.
* **Tổng lượng nước** → **Điện tích (Q)**: Nước tích trữ trong xô tương ứng với lượng điện tích trong tụ điện.
* Cho thấy tại sao tăng điện áp có ảnh hưởng lớn hơn tăng điện dung đến năng lượng tích trữ

A diagram of a water drain

Description automatically generated

* Độ mở của van (GATE voltage)
* Áp suất nước (điện áp nguồn)
* Kích thước ống dẫn (điện trở kênh)
* Kích thước lỗ thoát (điện trở DRAIN)

1. **Dennard Scaling**



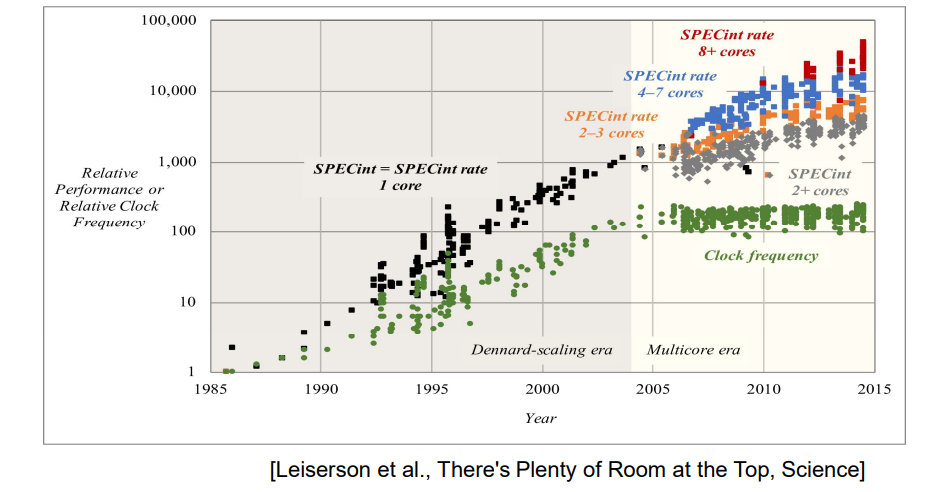
* **Khái niệm**: Định luật Dennard (Dennard Scaling) được đưa ra vào năm 1974, phát biểu rằng khi kích thước bóng bán dẫn giảm (thu nhỏ tiến trình), điện áp và dòng điện cũng giảm theo tỷ lệ tương ứng, giúp giữ mức tiêu thụ điện năng không đổi trên một đơn vị diện tích.

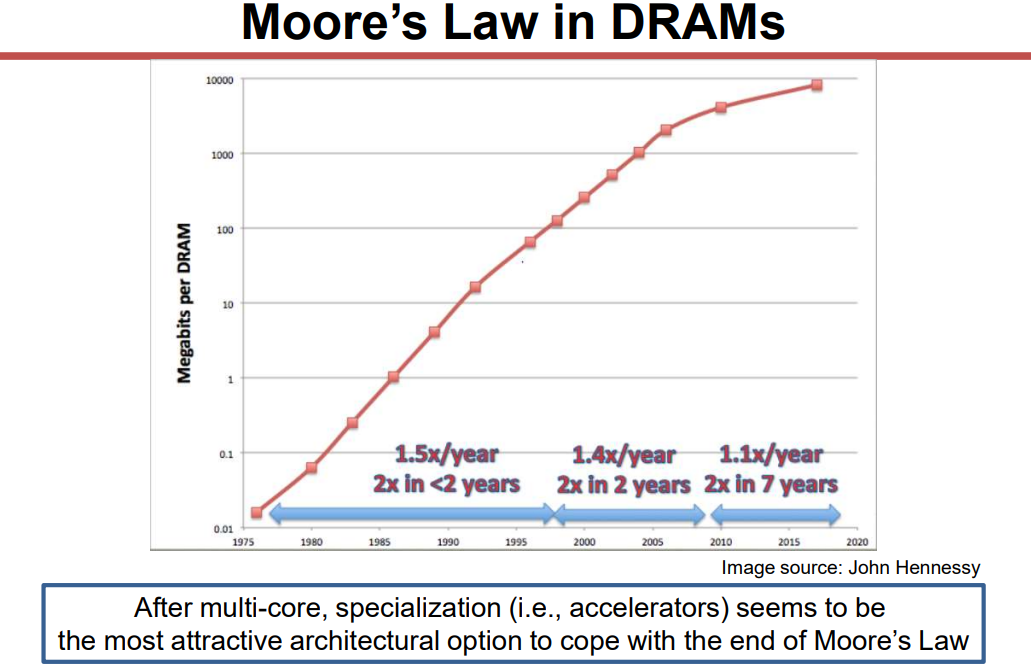
Điều này cho phép:

* Khi công nghệ vi mạch thu nhỏ (ví dụ từ 90nm xuống 45nm), có thể tăng tốc độ xử lý mà **không làm tăng mức tiêu thụ điện năng hay nhiệt độ**.
* Các CPU có thể tăng tần số xung nhịp liên tục trong nhiều thập kỷ mà không gặp vấn đề về nhiệt.

Đến khoảng năm 2005, định luật này không còn đúng nữa do các lý do sau:

* **Điện áp không thể tiếp tục giảm vô hạn** mà vẫn đảm bảo hoạt động ổn định của bóng bán dẫn.
* **Rò rỉ dòng điện (leakage current) gia tăng** khi kích thước bóng bán dẫn giảm, dẫn đến tiêu thụ năng lượng tăng ngay cả khi không hoạt động.
* **Hiện tượng "Power Wall" xuất hiện**: Khi tiêu thụ điện năng và sinh nhiệt quá cao, việc tăng tần số xung nhịp không còn khả thi nữa.
* Sau đó, ngành công nghiệp chuyển sang **đa lõi và xử lý song song** để tiếp tục tăng hiệu suất.





1. **Thiết Kế Tăng Tốc**

Bộ gia tốc (accelerator) là các phần cứng chuyên dụng giúp tăng tốc các tác vụ tính toán nặng, chẳng hạn như **GPU, TPU, FPGA, hay ASIC.** Khi thiết kế kiến trúc của bộ gia tốc, có 4 yếu tố quan trọng cần xem xét:

1. **Trạng thái (State):**

* Nếu bộ gia tốc có **trạng thái cục bộ**, nó có thể **tự xử lý dữ liệu** mà không cần truy cập bộ nhớ ngoài liên tục, giúp giảm độ trễ.
* Nếu bộ gia tốc sử dụng **trạng thái toàn cục**, nó chia sẻ bộ nhớ chính với CPU, giúp dễ dàng giao tiếp nhưng có thể làm chậm hiệu suất do tắc nghẽn bộ nhớ.

1. **Thao tác dữ liệu (Data Operations):**

* Một số bộ gia tốc có các **lệnh và toán tử dữ liệu đặc biệt** không có trong CPU, chẳng hạn như phép nhân ma trận trong TPU hoặc các lệnh SIMD (Single Instruction, Multiple Data) trong GPU.

1. **Truy cập bộ nhớ (Memory Access):**

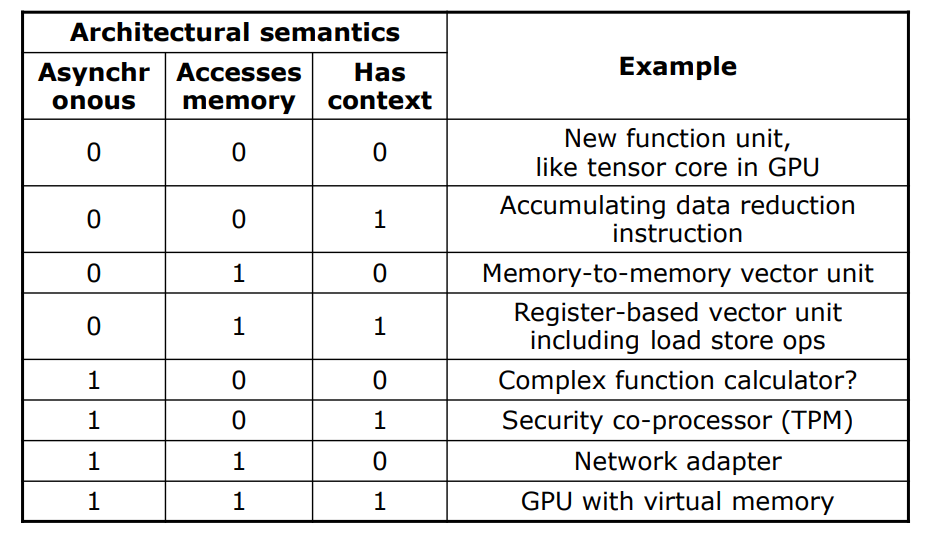
* Nếu bộ gia tốc có thể truy cập bộ nhớ toàn cục, cần có cơ chế đồng bộ hóa để tránh tranh chấp dữ liệu giữa CPU và accelerator.
* Một số thiết kế sử dụng **bộ đệm (cache)** hoặc **bộ nhớ cục bộ** để giảm tải truy cập bộ nhớ chính.

1. **Luồng điều khiển (Control Flow):**

* Bộ gia tốc có thể **hoạt động độc lập** hoặc phải **tuân theo điều khiển từ CPU.**
* Ví dụ:
  + GPU có thể xử lý nhiều tác vụ đồng thời với CPU.
  + Một số bộ gia tốc chuyên dụng (như bộ gia tốc mã hóa video) có thể cần CPU điều khiển trực tiếp.

1. **Phân loại Hoạt Động**

* **Đồng bộ**: Bộ gia tốc tích hợp trong CPU, điều khiển bởi lệnh tiêu chuẩn.
* **Không đồng bộ**: Bộ gia tốc chạy độc lập, CPU chỉ khởi động rồi tiếp tục công việc khác.
* Lựa chọn giữa hai kiểu tùy thuộc vào độ trễ, kích thước mạch, dữ liệu đầu vào và mức độ hoạt động song song.
* **Các Lựa Chọn Kiến Trúc**

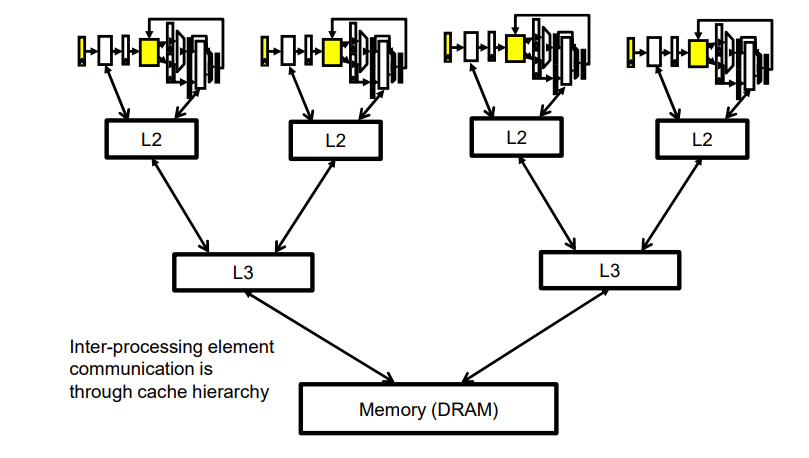


* Các lựa chọn kiến trúc khác nhau dựa trên việc có hay không truy cập bộ nhớ và có trạng thái hay không. A screenshot of a computer

  Description automatically generated

1. **Trình tự hoạt động (Operation Sequencing)**

* **Kiến trúc temporal** (điều khiển theo thời gian):
  + Sử dụng luồng lệnh động và bộ nhớ chia sẻ (ví dụ: CPU, GPU).
  + Phụ thuộc vào bộ nhớ phân cấp, tối ưu cho các tác vụ đa dạng.
* **Kiến trúc spatial** (điều khiển theo không gian):
  + Gồm các phần tử xử lý (PE) nhỏ giao tiếp trực tiếp, không cần bộ nhớ chia sẻ. (ví dụ: FPGA, ASIC).
  + Tối ưu cho thông lượng cao và giảm truy cập bộ nhớ



A diagram of a complex structure

Description automatically generated

1. **Kiến Trúc Spatial cho Deep Learning**

**Thành phần chính**:

* + **Processing Element (PE)**: Đơn vị tính toán cơ bản với thanh ghi cục bộ (0.5–1.0 kB).
  + **Global Buffer**: Bộ nhớ đệm tập trung (100–500 kB) để tái sử dụng dữ liệu.
  + **Mạng liên kết trực tiếp**: Giảm độ trễ truyền dữ liệu giữa các PE.
* **Lợi ích**: Giảm 50–90% số lần truy cập bộ nhớ so với kiến trúc temporal

Spatial architecture tối ưu hóa việc giảm di chuyển dữ liệu, phù hợp cho các tác vụ học sâu cần thông lượng cao.

Temporal architecture phụ thuộc vào bộ nhớ chia sẻ, phù hợp cho tác vụ đa dạng nhưng kém hiệu quả về năng lượng.

**Các khối logic có thể cấu hình (Configurable Logic Blocks - CLB)**

* CLB được sử dụng để triển khai logic tuần tự (sequential) và logic tổ hợp (combinational).

**CLB (Configurable Logic Blocks)** là thành phần cốt lõi trong FPGA, dùng để triển khai **logic tổ hợp** và **logic tuần tự**.

**Cấu trúc CLB** gồm nhiều **Basic Logic Elements (BLE)**, trong đó mỗi BLE có:

* **LUT (Look-Up Table):** Lưu bảng giá trị để thực hiện hàm logic.
* **Registers:** Lưu trữ dữ liệu cho logic tuần tự.
* **Multiplexer:** Chọn đầu ra mong muốn.

**Ứng dụng:** CLB giúp FPGA linh hoạt trong thiết kế phần cứng, ứng dụng rộng rãi trong xử lý tín hiệu số, hệ thống nhúng và các giải pháp tùy chỉnh.

A diagram of a clock

Description automatically generated

A graph with red arrows and black text

Description automatically generated

1. **Số lượng BLE tăng lên:**

* LUT lớn hơn có thể thực hiện nhiều hàm logic hơn, nhưng đồng thời cũng cần thêm BLE để tận dụng hết khả năng của LUT.
* Vì vậy, khi LUT kích thước lớn hơn, số lượng BLE trong FPGA cũng tăng lên để đảm bảo hiệu quả sử dụng tài nguyên.

2. **Diện tích trên mỗi BLE (Area/BLE) giảm:**

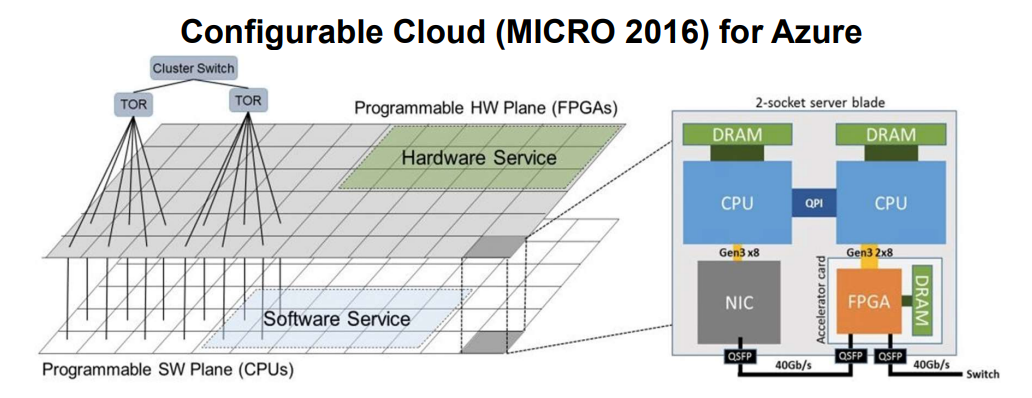
* LUT lớn hơn giúp giảm số lượng kết nối liên khối (interconnect), do đó **diện tích trung bình trên mỗi BLE nhỏ hơn**.
* Khi LUT có thể xử lý nhiều logic hơn, số lượng kết nối giữa BLE giảm đi, làm cho tổng diện tích trung bình mà mỗi BLE chiếm dụng nhỏ hơn.
* **Ưu điểm:**  
  - LUT lớn giúp tối ưu hóa việc sử dụng tài nguyên, giảm overhead từ kết nối giữa các BLE.  
  - Diện tích trung bình trên mỗi BLE giảm giúp tiết kiệm tài nguyên FPGA.
* **Nhược điểm:**  
  - Khi LUT lớn hơn, **số lượng BLE cũng tăng**, có thể làm tiêu tốn nhiều tài nguyên tổng thể.  
  - LUT lớn có thể làm giảm khả năng tái sử dụng linh hoạt của FPGA trong một số ứng dụng.

A graph with red arrows and a red arrow pointing to the same level

Description automatically generated

* **LUT lớn hơn có thể thực hiện nhiều logic hơn trong một bước** → Giảm số lượng BLE cần thiết để hoàn thành một phép toán logic.
* **Giảm số mức logic (logic levels)** → Vì mỗi LUT có thể tính toán nhiều hơn, nên số BLE trong đường tới hạn giảm.
* LUT lớn hơn có nhiều đầu vào hơn → **Mạch bên trong phức tạp hơn** → **Thời gian xử lý lâu hơn**.
* LUT cần nhiều tầng MUX bên trong để chọn đầu ra từ nhiều đầu vào hơn → **Độ trễ tăng do phải đi qua nhiều tầng logic hơn**.

1. Microsoft Project Catapult:



* Là một dự án của Microsoft tích hợp FPGA vào hệ thống điện toán đám mây
* Được sử dụng để tăng tốc và giảm độ trễ cho:
  + Công cụ tìm kiếm Bing
  + Mạng được định nghĩa bằng phần mềm
  + Mã hóa và giải mã

1. Khối không đồng nhất (Heterogeneous Blocks) trong FPGA:

* Thêm các khối logic chuyên biệt vào FPGA:
  + Specific Purpose Logic: Thêm logic chuyên biệt vào FPGA
  + Soft fabric: LUT, flops, phép cộng, trừ, logic carry
  + Memory block (BRAM): Có thể cấu hình kích thước word và địa chỉ
  + Multipliers/MACs -> DSP
  + CPUs và processing elements

Việc sử dụng các khối không đồng nhất này giúp FPGA:

* Tối ưu hiệu năng cho các ứng dụng cụ thể
* Tiết kiệm diện tích và năng lượng
* Tăng tính linh hoạt trong thiết kế
* Cải thiện hiệu suất tổng thể của hệ thống

A diagram of different colors of memory blocks

Description automatically generated with medium confidence

1. Accelerator Taxonomy:

* Phân loại các kiến trúc accelerator thành:
  + Temporally Programmed (CPU, GPU)
  + Spatially Programmed:
    - Fine Grained (FPGA)
    - Coarse Grained (Triggered Instructions như RAW, AsAP, PicoChip...)

**1. Systolic Array là gì?**

* Là một kiến trúc tính toán song song, trong đó dữ liệu được truyền tuần tự qua một mạng lưới các phần tử xử lý (PE).
* Mỗi PE thực hiện một phép toán cụ thể trước khi truyền kết quả cho PE tiếp theo.

**2. Các đặc điểm:**

* **Mỗi PE được cố định cho một phép toán:**
  + Không thể thay đổi chức năng của PE trong quá trình hoạt động.
  + Điều này giúp tối ưu hiệu suất nhưng giảm tính linh hoạt.
* **Pipelined operation - Không có tắc nghẽn (backpressure):**
  + Dữ liệu được truyền qua các PE theo chu kỳ cố định.
  + Không có độ trễ do tắc nghẽn như trong các hệ thống không đồng bộ.
* **Độ đồng thời cao (High-concurrency):**
  + Nhiều PE có thể xử lý dữ liệu cùng lúc → Tăng tốc tính toán.
  + Thường được sử dụng trong các ứng dụng như xử lý tín hiệu số (DSP) hoặc AI (deep learning).
* **Thiết kế có tính quy tắc (Regular design):**
  + Cấu trúc mảng đơn giản, dễ mở rộng và tối ưu hóa.
  + Dễ dàng triển khai trong FPGA hoặc ASIC.
* **Chỉ hỗ trợ song song dạng quy tắc (Regular parallelism only):**
  + Hạn chế khi xử lý các bài toán không có cấu trúc dữ liệu cố định.
  + Không phù hợp với các thuật toán có logic điều khiển phức tạp hoặc luồng dữ liệu không đồng nhất.

A diagram of a computer processor

Description automatically generated

1. **Fixed Operation - Google TPU:**

* Google TPU sử dụng mảng systolic để thực hiện phép nhân ma trận 8-bit 256x256 và tích lũy kết quả.

A diagram of a computer system

Description automatically generated

1. **Single Configured Operation - Dyser**

* Kiến trúc Dyser hỗ trợ các đường dữ liệu được chuyên biệt hóa động để tối ưu hiệu suất và tiết kiệm năng lượng.

A diagram of a machine

Description automatically generated

1. **PC-based Control – Wave Computing**

* Kiến trúc dựa trên máy tính PC, sử dụng mô hình kiểm soát dựa trên phần mềm.

A diagram of a ram

Description automatically generated

1. **Guarded Actions**

* Chương trình bao gồm các quy tắc chỉ thực thi khi điều kiện cho phép.
* Cung cấp tính song song thông qua:
  + Song song trong từng quy tắc (Intra-Rule)
  + Song song giữa các quy tắc (Inter-Rule)
  + Song song giữa bộ lập lịch và thực thi quy tắc.

A screenshot of a computer program

Description automatically generated

1. **Triggered Instructions**

* Hạn chế hành động có điều kiện xuống mức phần cứng (ISA core).
* Không có bộ đếm chương trình hoặc lệnh rẽ nhánh.

A screenshot of a computer program

Description automatically generated

1. **Triggered Instruction Scheduler**

* Sử dụng logic tổ hợp để đánh giá và chọn các lệnh được kích hoạt.
* Có thể chọn nhiều lệnh không xung đột để thực thi song song.

A diagram of a diagram

Description automatically generated