ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Lõi IP so trùng chuỗi DNA sử dụng giải thuật Smith Waterman phục vụ y tế thông minh

GVHD: PGS.TS Phạm Quốc Cường

Nhóm SV thực hiện: Nguyễn Minh Hưng – 2211367

Lâm Nữ Uyển Nhi – 2212429

Vũ Đức Lâm – 2211824

Hồ Đăng Khoa – 2211588

Thành phố Hồ Chí Minh, tháng 2 năm 2025

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Giới thiệu đề tài

Trong kỷ nguyên đô thị thông minh, y tế dự phòng và cá nhân hóa đóng vai trò then chốt trong việc nâng cao chất lượng cuộc sống. Việc giải mã và phân tích ADN nhanh chóng là nền tảng cho các ứng dụng y sinh tiên tiến, từ chẩn đoán sớm bệnh tật đến phát triển liệu pháp gen phù hợp cho từng cá nhân. Tuy nhiên, quá trình căn chỉnh chuỗi ADN, đặc biệt là với thuật toán Smith-Waterman, đòi hỏi nguồn lực tính toán lớn, gây cản trở việc triển khai rộng rãi trong môi trường đô thị.

Đề tài này tập trung vào thiết kế vi mạch FPGA hiệu năng cao, tối ưu hóa thuật toán Smith-Waterman, nhằm tăng tốc quá trình căn chỉnh ADN. Bằng cách tận dụng khả năng xử lý song song và tính linh hoạt của FPGA, chúng tôi hướng đến việc tạo ra một giải pháp vi mạch mạnh mẽ, đáp ứng nhu cầu phân tích gen nhanh chóng và hiệu quả trong các ứng dụng y tế thông minh đô thị. Giải pháp này không chỉ góp phần nâng cao hiệu quả y tế dự phòng, mà còn mở ra tiềm năng ứng dụng rộng rãi trong các lĩnh vực khác của đô thị thông minh, như quản lý môi trường và nông nghiệp đô thị.

2 Kiến thức nền tảng

2.1 Giải thuật Smith Waterman và quy tắc song song trên ma trận

2.1.1 Giải thuật Smith Waterman

Thuật toán Smith-Waterman là một phương pháp lập trình động (dynamic programming) được sử dụng để thực hiện căn chỉnh trình tự (sequence alignment) trong cơ sở dữ liệu di truyền, nhằm tìm ra sự tương đồng tối ưu giữa hai chuỗi DNA, RNA hoặc protein. Được phát triển bởi Temple F. Smith và Michael S. Waterman vào năm 1981, đây là một thuật toán "tối ưu" (optimal), có nghĩa là nó đảm bảo tìm ra tất cả các cặp khớp và gần khớp (homologues) giữa hai chuỗi, với độ nhạy cao hơn so với các phương pháp heuristic như BLAST hay FASTA. Tuy nhiên, nhược điểm lớn của nó là độ phức tạp tính toán cao, khiến nó tốn thời gian và tài nguyên bộ nhớ, đặc biệt khi xử lý cơ sở dữ liệu lớn hoặc số lượng lớn các chuỗi ngắn.

Thuật toán này đặc biệt hữu ích trong sinh học phân tử để:

- Xác định protein được mã hóa bởi một trình tự DNA.
- Nghiên cứu xu hướng tiến hóa giữa các loài qua sự tương đồng trình tự.
- Tìm mối liên hệ giữa bệnh tật và di truyền bằng cách so sánh trình tự DNA của người bệnh và người khỏe mạnh.

2.1.2 Nguyên lý hoạt động

Thuật toán Smith-Waterman sử dụng một ma trận hai chiều (2D table) để tính toán điểm số căn chỉnh tối ưu giữa hai chuỗi DNA hoặc protein. Hai chuỗi được so sánh thường được ký hiệu là:

- **S**: Chuỗi thứ nhất (ví dụ: "ACG").
- **T**: Chuỗi thứ hai (ví dụ: "ATC").

Mục tiêu là tìm ra "khoảng cách chỉnh sửa" (edit distance) tối thiểu giữa hai chuỗi hoặc điểm số căn chỉnh tối ưu, dựa trên việc cho phép các lỗi khớp (mismatch), chèn (insertion), và xóa (deletion).

2.1.3 Công thức Điền Ma Trận

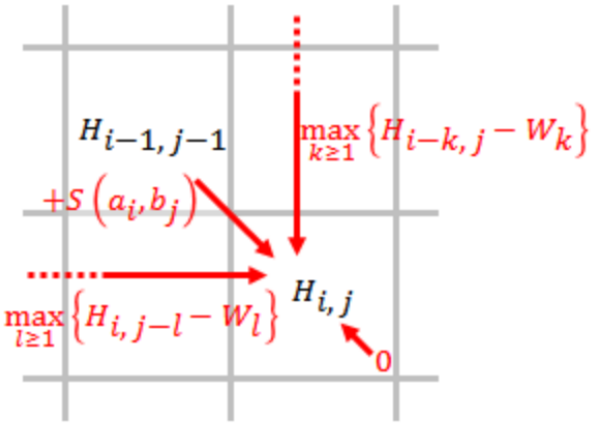
Cho hai chuỗi ký tự $A = a_1a_2...a_m$ và $B = b_1b_2...b_n$, ta định nghĩa một ma trận điểm số H với kích thước $(m+1) \times (n+1)$, trong đó mỗi phần tử $H(i,j)$ được tính theo công thức:

$$H(i,j) = \max \begin{cases} 0, \\ H(i-1,j-1) + S(a_i, b_j), \\ H(i-1,j) - d, \\ H(i,j-1) - d \end{cases}$$

Trong đó:

- $S(S_i, T_j)$ là điểm số thay thế (substitution score),
- d là điểm phạt cho khoảng trống (gap penalty).
- Nếu điểm số nhỏ hơn 0, ta đặt $H[i, j] = 0$ để chỉ giữ lại các vùng căn chỉnh có ý nghĩa.

Sau khi hoàn tất ma trận, ta tìm **giá trị lớn nhất** trong bảng để xác định vùng căn chỉnh tối ưu.



Hình 1: Hình ảnh minh họa thuật toán

Giả sử ta so sánh hai chuỗi:

$$S = \text{GTTACG}$$

$$T = \text{TGTTAC}$$

Với điểm số **match = 1**, **mismatch = -1**, **gap penalty = -2**, ta xây dựng ma trận như sau:

Bước 1: Khởi tạo ma trận

	–	G_1	T_2	T_3	A_4	C_5	G_6
–	0	0	0	0	0	0	0
T_1	0						
G_2	0						
T_3	0						
T_4	0						
A_5	0						
C_6	0						

Bước 2: Điền giá trị vào ma trận

	–	G_1	T_2	T_3	A_4	C_5	G_6
–	0	0	0	0	0	0	0
T_1	0	0	1	1	0	0	0
G_2	0	1	0	0	0	0	1
T_3	0	0	2	1	0	0	0
T_4	0	0	1	3	1	0	0
A_5	0	0	0	1	4	2	0
C_6	0	0	0	0	2	5	3

2.1.4 Truy vết (Backtracking)

Sau khi hoàn thành ma trận H, ta tìm giá trị lớn nhất trong ma trận và truy ngược lại theo các bước đi hợp lệ để tìm đoạn căn chỉnh tốt nhất.

	–	G_1	T_2	T_3	A_4	C_5	G_6
–	0	0	0	0	0	0	0
T_1	0	0	1	1	0	0	0
G_2	0	1	0	0	0	0	1
T_3	0	0	2	1	0	0	0
T_4	0	0	1	3	1	0	0
A_5	0	0	0	1	4	2	0
C_6	0	0	0	0	2	5	3

Results:

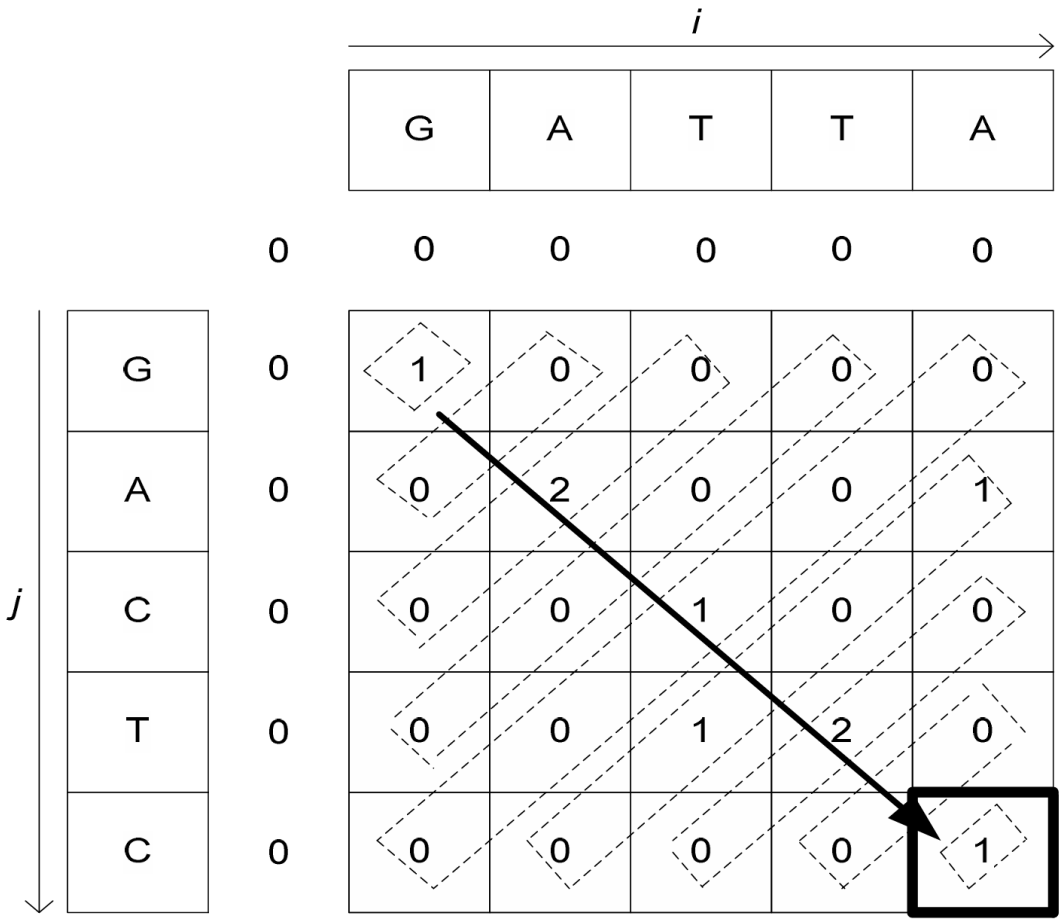
GTTAC

*****

GTTAC

2.1.5 Quy Tắc Song Song Trên Ma Trận

Để tăng tốc độ tính toán, thuật toán Smith-Waterman có thể được thực hiện trên kiến trúc song song, thường sử dụng quy tắc song song sau:



Hình 2: Song Song trên ma trận

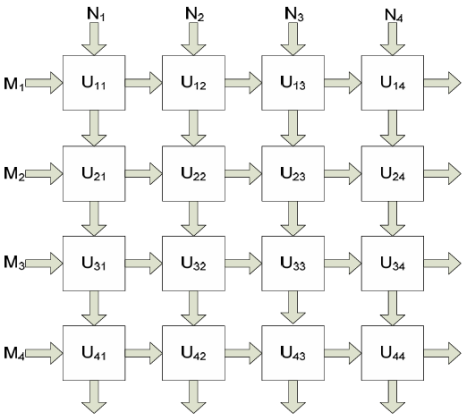
Ma trận H có thể được tính theo đường chéo thay vì hàng và cột. Do mỗi ô $H(i, j)$ chỉ phụ thuộc vào các ô $H(i-1, j-1)$, $H(i-1, j)$ và $H(i, j-1)$, ta có thể tính toán các phần tử theo đường chéo một cách song song.

2.2 Kiến trúc mảng Systolic

2.2.1 Khái niệm

Kiến trúc **Systolic Array** là một mô hình tính toán song song được thiết kế để tăng tốc các thuật toán có tính chất lặp lại, đặc biệt là trong xử lý tín hiệu số, đồ họa máy tính và sinh học tính toán. Hệ thống này gồm một tập hợp các bộ xử lý (Processing Elements - PE) được sắp xếp theo dạng lưới hoặc chuỗi, trong đó dữ liệu được đẩy qua mạng lưới theo nhịp đồng hồ mà không cần truy xuất bộ nhớ ngoài liên tục.

Hình 4 là một ví dụ đơn giản về kiến trúc mảng systolic. Trong cấu hình này, có hai đầu vào là hai vectơ và các ô xử lý có giá trị $H_{i,j}$, thường là kết quả của một thuật toán được định nghĩa trong các ô.



Hình 3: Ví dụ đơn giản về kiến trúc mảng systolic

2.2.2 Cấu trúc tổng quát

Đặc điểm quan trọng của Systolic Array:

- **Dữ liệu chảy liên tục:** Dữ liệu được truyền từ một PE sang PE khác mà không cần truy cập bộ nhớ chính.
- **Tính song song cao:** Các PE hoạt động đồng thời để tối ưu hóa tốc độ xử lý.
- **Thiết kế phần cứng chuyên dụng:** Thường được triển khai trên FPGA, ASIC hoặc các vi xử lý có hỗ trợ SIMD.

2.2.3 Systolic Array trong thuật toán Smith-Waterman

Trong bài toán **căn chỉnh chuỗi ADN**, Systolic Array được sử dụng để tính toán ma trận điểm số với quy tắc song song:

- Các phần tử của ma trận được tính toán theo từng đường chéo (anti-diagonal), thay vì từng hàng hoặc từng cột.
- Các PE trong hệ thống thực hiện phép so sánh, tính điểm và truyền giá trị một cách đồng bộ.
- Việc triển khai trên FPGA giúp tăng tốc thuật toán lên hàng trăm lần so với phần mềm chạy trên CPU truyền thống.

2.2.4 Ưu và Nhược điểm

Ưu điểm:

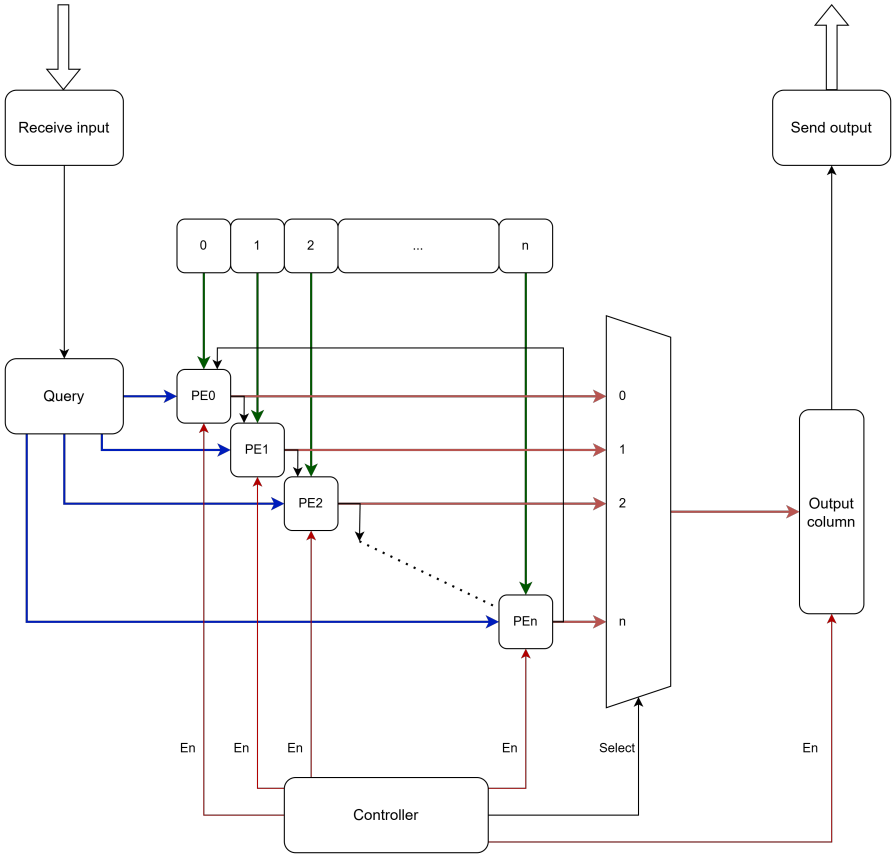
- Hiệu suất cao nhờ khả năng tính toán song song.
- Giảm độ trễ bộ nhớ vì dữ liệu được truyền trực tiếp giữa các phần tử.
- Tối ưu tài nguyên phần cứng, đặc biệt là trên FPGA.

Nhược điểm:

- Thiết kế phần cứng phức tạp, khó lập trình hơn CPU/GPU thông thường.
- Không linh hoạt với những bài toán có cấu trúc dữ liệu thay đổi liên tục.

3 Kiến trúc hệ thống

Hệ thống được thiết kế để xử lý dữ liệu sinh học trên nền tảng FPGA, bao gồm nhiều module hoạt động phối hợp để thực hiện truy vấn và tính toán trên dữ liệu nucleotide. Hệ thống sử dụng kiến trúc **Systolic Array** cùng thuật toán **Smith Waterman** bao gồm các thành phần chính như module nhận dữ liệu đầu vào, module lưu trữ database, module truy vấn, các phần tử xử lý (PE), bộ điều khiển trung tâm, module xuất kết quả và module gửi kết quả lên server. Mỗi module đảm nhiệm một chức năng cụ thể, từ nhập dữ liệu, lưu trữ, truy vấn, đến xử lý và xuất kết quả. Hệ thống đảm bảo khả năng truy cập dữ liệu nhanh chóng, tối ưu hiệu suất tính toán, phù hợp với các ứng dụng sinh học yêu cầu xử lý dữ liệu lớn.



Hình 4: Sơ đồ khối hệ thống

1. **Module nhận dữ liệu đầu vào**
   - **Input:** Dữ liệu query và database (phần tử) gửi từ network.
   - **Output:** Dữ liệu input đã được xử lý thành từng nucleotide (dạng mã hóa nhị phân).
2. **Module Database**
   - Là memory chứa n vùng nhớ, mỗi vùng nhớ gồm 2 bit (đối với nucleotide mã hóa cần 2 bit).
   - Mỗi vùng nhớ được cấp riêng cho một PE nhất định.

3. **Module Query**
   - Tương tự database, tuy nhiên chứa m vùng nhớ truy cập.
   - Toàn bộ vùng nhớ của module PE nào cũng có thể truy cập.
4. **Module PE**
   - Mỗi PE đều có bộ nhớ riêng chứa m vùng nhớ (kích thước module query).
   - PE sẽ đọc và giải dữ liệu database ở vùng nhớ, truy xuất giá trị query (trạng thái của PE 0 sẽ lấy từ database địa chỉ là 0).
   - Giá trị mà PE tính toán sẽ lưu vào bộ nhớ của nó.
5. **Module Controller**
   - Cho phép PE ghi giá trị tính toán vào bộ nhớ của PE nếu $En = 1$.
   - Lựa chọn bộ nhớ của PE nào sẽ là giá trị output.
6. **Module output column**
   - Vùng nhớ có kích thước bằng m (tương ứng với kích thước module query)
7. **Module gửi kết quả**
   - Kết quả đã được qua xử lý được trả về máy của client thông qua mạng.
   - Dữ liệu sau khi được xử lý hoàn tất sẽ được backtracking tìm chuỗi DNA con có sự tương đồng cao nhất so với chuỗi DNA được gửi vào query.

Hệ thống được thiết kế dựa trên thuật toán Smith-Waterman kết hợp với kiến trúc *Systolic Array* nhằm tối ưu hóa quá trình xử lý và so khớp chuỗi sinh học. Với kiến trúc này, việc tính toán được thực hiện theo phương thức song song, giúp tăng tốc đáng kể quá trình xử lý ma trận điểm số so khớp. Để hiện thực hóa điều này, nhóm tác giả sử dụng kiến trúc *Cycle PE*, trong đó mỗi *Processing Element* (PE) đóng vai trò như một phần tử xử lý tuần tự nhưng hoạt động theo chu kỳ và phụ thuộc vào dữ liệu của PE trước đó.

Cụ thể, tại thời điểm ban đầu, PE_0 sẽ bắt đầu thực hiện tính toán dựa trên dữ liệu đầu vào được cung cấp từ bộ nhớ *query* và *database*. Ở chu kỳ tiếp theo PE_0 và PE_1 sẽ được tính toán cùng lúc trên cùng 1 đường chéo. Giá trị đầu vào của PE_1 lúc này không chỉ đến từ dữ liệu ban đầu mà còn được cập nhật từ kết quả tính toán của PE_0 tại chu kỳ trước đó. Điều này tạo nên một quá trình truyền dữ liệu và tính toán liên tục theo dạng dòng chảy dữ liệu, giúp các PE có thể tận dụng thông tin từ các phần tử lân cận để đảm bảo tính chính xác của phép so khớp chuỗi.

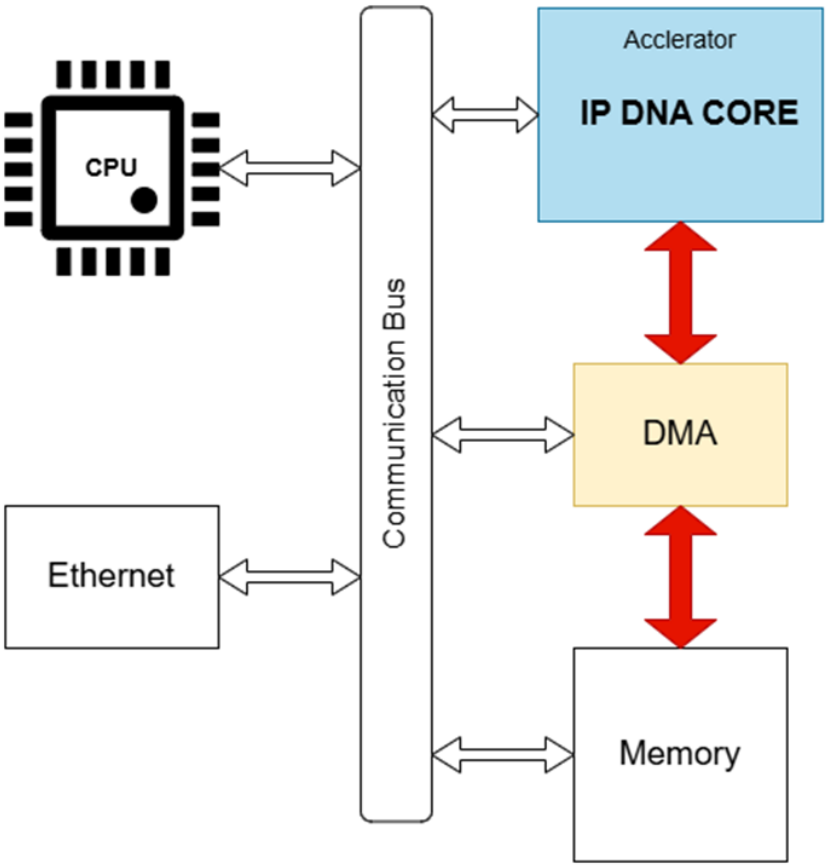
Sau khi PE_0 hoàn tất tính toán tại một chu kỳ nhất định, *module output column* sẽ đảm nhiệm việc lưu trữ giá trị kết quả do PE_0 tạo ra. Đồng thời, PE_0 sẽ tiếp tục lấy giá trị mới từ *database* (tại vị trí $n + 1$) để tiếp tục quá trình tính toán tại xung *clock* kế tiếp. Quá trình này tiếp diễn theo chu kỳ lặp, trong đó mỗi PE sẽ liên tục lấy dữ liệu mới, tính toán và chuyển giao kết quả sang các PE khác để đảm bảo dòng dữ liệu được duy trì mà không gây gián đoạn.

Một điểm quan trọng của kiến trúc này là sự dịch chuyển dữ liệu dọc theo các PE theo từng xung *clock*. Cụ thể, tại xung *clock* thứ $x = x + kn$ sau, với $x_0 = n$ và $k \in \mathbb{N}$, PE_0 sẽ tiếp tục nhận dữ liệu mới tại vị trí $kn + 1$ và thực thi tính toán. Điều này có nghĩa là khi toàn bộ hệ thống đạt đến trạng thái ổn định, mỗi PE sẽ đóng vai trò như một điểm tính toán trong quá trình quét và so khớp ma trận điểm số giữa *query* và *database*. Nhờ vào cách tổ chức này, toàn bộ hệ thống có thể khai thác tối đa khả năng xử lý song song của FPGA, giúp rút ngắn đáng kể thời gian tính toán so với các kiến trúc tuần tự truyền thống.

4 Đề xuất hệ thống

Co-processor là một bộ xử lý phụ trợ, hoạt động song song với CPU chính nhằm đảm nhiệm các tác vụ chuyên biệt, chẳng hạn như xử lý đồ họa (GPU) hoặc tính toán số học (FPU). Co-Processor DNA được thiết kế dựa trên các **lý thuyết đã trình bày** ở trên để xử lý các tác vụ liên quan đến so trùng chuỗi DNA, giúp tăng tốc quá trình phân tích dữ liệu và nâng cao độ chính xác.

Hệ thống Co-Processor DNA bao gồm các thành phần chính: **CPU**, **Accelerator (IP DNA Core)**, **DMA (Direct Memory Access)**, **Memory**, **Ethernet** và **Communication Bus**. Sơ đồ khối dưới đây mô tả mối quan hệ giữa các thành phần và quá trình xử lý dữ liệu trong hệ thống:



Hình 5: Co-Processor DNA System

4.1 Các thành phần chính của hệ thống

- **CPU**:
  - Đóng vai trò trung tâm, điều khiển và giám sát toàn bộ hệ thống.
  - Gửi lệnh cấu hình và điều khiển các thành phần thông qua Communication Bus.
  - Đảm bảo tính đồng bộ và tối ưu hoá quá trình xử lý.
- **IP DNA Core (Accelerator)**:

  - Là bộ gia tốc chuyên dụng, chịu trách nhiệm thực hiện các tác vụ tìm kiếm mẫu trong chuỗi DNA.
  - Giảm tải công việc cho CPU bằng cách xử lý các phép tính phức tạp một cách độc lập.
  - Tăng tốc độ xử lý và cải thiện hiệu suất của toàn bộ hệ thống.

- **DMA (Direct Memory Access)**:
  - Quản lý việc truyền dữ liệu trực tiếp giữa Memory và IP DNA Core.
  - Giảm thiểu sự phụ thuộc vào CPU trong quá trình truyền dữ liệu, từ đó ngăn ngừa tình trạng tắc nghẽn.
  - Đảm bảo tốc độ truyền dữ liệu cao và hiệu quả trong các ứng dụng xử lý dữ liệu lớn.

- **Memory**:
  - Lưu trữ dữ liệu DNA và kết quả xử lý trung gian.
  - Hỗ trợ truy xuất nhanh và đáng tin cậy, đảm bảo các thông tin cần thiết luôn sẵn sàng cho quá trình tính toán.

- **Ethernet**:
  - Cung cấp giao diện truyền và nhận dữ liệu với tốc độ cao.
  - Cho phép kết quả phân tích DNA được gửi đến các máy chủ hoặc hệ thống khác để xử lý thêm.
  - Hỗ trợ tích hợp với các hệ thống mạng phân tán, từ đó mở rộng khả năng kết nối và chia sẻ dữ liệu.

- **Communication Bus**:
  - Kết nối tất cả các thành phần của hệ thống, cho phép truyền dữ liệu và lệnh cấu hình một cách nhanh chóng.
  - Áp dụng các giao thức bus tốc độ cao như AMBA AXI hoặc Avalon (Intel FPGA) nhằm đảm bảo hiệu suất tối ưu.
  - Hỗ trợ việc giám sát và điều khiển toàn bộ hệ thống từ CPU, giúp hệ thống hoạt động ổn định và hiệu quả.

Dữ liệu DNA được lưu trữ ban đầu trong Memory. Qua cơ chế DMA, dữ liệu này được truyền trực tiếp đến IP DNA Core thông qua đường High-Speed Data Path (được hiển thị bằng màu đỏ trong sơ đồ). Quá trình này giúp giảm thiểu độ trễ và tăng tốc độ xử lý, bởi vì CPU không cần phải can thiệp vào từng khâu truyền dữ liệu. Sau khi được xử lý, kết quả phân tích có thể được lưu trữ lại trong Memory để phục vụ cho các bước xử lý tiếp theo. Nhằm phục vụ hệ phân tán, dữ liệu có thể được gửi ra ngoài qua Ethernet để trao đổi với các hệ thống khác như máy chủ hoặc trung tâm xử lý dữ liệu. Communication Bus đảm bảo rằng CPU luôn có thể giám sát và cấu hình lại toàn bộ quá trình, giúp duy trì hiệu suất và độ ổn định của hệ thống.

4.2 Tính mở rộng và Ứng dụng

Thiết kế của Co-Processor DNA không chỉ đáp ứng các yêu cầu xử lý dữ liệu hiện tại mà còn có tính mở rộng cao, cho phép tích hợp với nhiều hệ thống và ứng dụng khác nhau:

- **Ứng dụng trong y học**:
  - Hệ thống có thể được tích hợp vào các thiết bị y tế hiện đại, hỗ trợ chẩn đoán bệnh và phân tích gen nhanh chóng.
  - Nâng cao khả năng xử lý dữ liệu trong các quá trình chẩn đoán và điều trị bệnh dựa trên phân tích DNA.
- **Nghiên cứu di truyền học**:
  - Giúp các nhà khoa học phân tích dữ liệu di truyền một cách hiệu quả, rút ngắn thời gian nghiên cứu.
  - Tăng cường độ chính xác trong việc xác định các mẫu DNA, từ đó hỗ trợ các nghiên cứu về di truyền, tiến hóa và đa dạng sinh học.
- **Hệ thống phân tán (Clusters)**:
  - Nhờ vào giao diện Ethernet và API chuẩn, Co-Processor DNA có thể dễ dàng được triển khai trong các cụm máy chủ xử lý dữ liệu quy mô lớn.
  - Hỗ trợ xử lý dữ liệu song song ở quy mô rộng, từ đó tăng hiệu suất xử lý tổng thể của hệ thống.
- **Các ứng dụng công nghệ sinh học khác**:
  - Khả năng tích hợp với các hệ thống khác mở ra nhiều cơ hội ứng dụng trong công nghiệp thực phẩm, nghiên cứu môi trường và các lĩnh vực công nghệ sinh học tiên tiến.

Hệ thống Co-Processor DNA là một giải pháp tiên tiến, kết hợp sức mạnh của các bộ gia tốc chuyên dụng và khả năng truyền dữ liệu tốc độ cao nhằm giải quyết các bài toán xử lý DNA phức tạp. Với tính mở rộng và khả năng tích hợp linh hoạt, hệ thống không chỉ tối ưu hoá quá trình phân tích dữ liệu mà còn mở ra nhiều hướng ứng dụng đa dạng trong y học, nghiên cứu di truyền học và các lĩnh vực công nghệ sinh học hiện đại.