(12) **BẢN MÔ TẢ GIẢI PHÁP HỮU ÍCH THUỘC BẰNG ĐỘC QUYỀN GIẢI PHÁP HỮU ÍCH**

(19) **Cộng hòa xã hội chủ nghĩa Việt Nam (VN)**
CỤC SỞ HỮU TRÍ TUỆ

(11) **2-0003247**

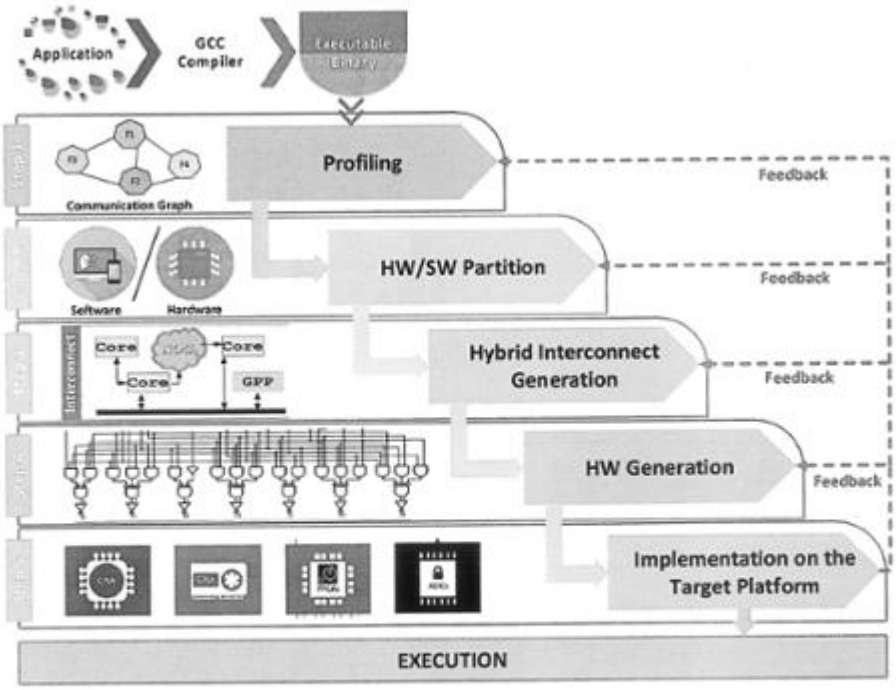
(51) **G06F 9/50**
2020.01

(13) **Y**

(21) 2-2019-00437
(22) 09/10/2019
(45) 25/08/2023 425
(43) 26/04/2021 397
(73) Trường Đại học Bách khoa - Đại học Quốc gia Thành phố Hồ Chí Minh (VN)
268 Lý Thường Kiệt, phường 14, quận 10, thành phố Hồ Chí Minh
(72) Phạm Quốc Cường (VN).

(54) PHƯƠNG PHÁP HỖ TRỢ THIẾT KẾ TĂNG TỐC PHẦN CỨNG VỚI KIẾN TRÚC KẾT NỐI LAI

(57) Công trình hiện thực một phương pháp hỗ trợ việc thiết kế hệ thống tăng tốc phần cứng với kiến trúc kết nối lai bằng cách tạo ra khung sườn hỗ trợ thiết kế. Khung sườn cung cấp những hỗ trợ cần thiết cho việc đơn giản hóa việc thiết kế, phân tích, đánh giá và lựa chọn kiến trúc kết nối phù hợp nhất với ứng dụng nhằm đạt được hiểu quả cao nhất trong hiệu suất xử lý và ít tốn tài nguyên phần cứng. Những hỗ trợ đã được đề cập bao gồm mô-đun phân tích đặc tính giao tiếp dữ liệu, mô-đun phân hoạch phần cứng phần mềm, mô-đun sinh kiến trúc kết nối lai, công cụ sinh mã đặc tả phần cứng và công cụ tổng hợp hệ thống. Phương pháp hướng tới việc tạo lập một kiến trúc kết nối lai linh động phù hợp nhất với đặc tính giao tiếp dữ liệu giữa các thành phần trong một ứng dụng cụ thể. Kiến trúc kết nối lai sẽ khác nhau cho từng ứng dụng cả về thành phần lẫn tài nguyên cần thiết cho các thành phần.



Chú thích:

- Application: Ứng dụng
- Executable Binary: Mã nhị phân thực thi
- Profiling: Lập hồ sơ
- Hybrid Interconnect Generation: Sinh kết nối lai
- Implementation on Target Platform: Triển khai trên nền tảng phần cứng
- GCC Compiler: Trình biên dịch GCC
- Communication graph: đồ thị giao tiếp
- HW/SW Partition: phân hoạch phần cứng/phần mềm
- HW Generation: Sinh phần cứng

Lĩnh vực kỹ thuật được đề cập

Công trình hướng đến việc hiện thực một phương pháp hỗ trợ thiết kế tăng tốc phần cứng với kiến trúc kết nối lai, được sử dụng trong lĩnh vực khoa học tính toán, đặc biệt là tính toán với hiệu năng cao với chi phí phần cứng tiết kiệm và năng lượng tiêu thụ thấp. Phương pháp đơn giản hóa quá trình tạo ra các hệ thống tăng tốc phần cứng, có thể được sử dụng bởi đại đa số các đối tượng người dùng trên phần cứng tái cấu hình với độ linh hoạt cao với chi phí sản xuất thấp. Ngoài ra, kiến trúc kết nối lai giúp cho hiệu suất tính toán của các hệ thống tăng tốc phần cứng cao hơn.

Tình trạng kỹ thuật của công trình

Nhu cầu tính toán ngày càng tăng lên, đặc biệt là khi chúng ta đang bước vào kỷ nguyên dữ liệu lớn. Tuy nhiên, tốc độ tăng tần số hoạt động của CPU đã chậm lại trong vài năm qua. Do đó, các lựa chọn thay thế nên được nghiên cứu để đáp ứng yêu cầu này. Mặt khác, ngày càng nhiều bóng bán dẫn có khả năng tích hợp trên một chip đơn. Cho đến thời điểm hiện tại, có thể tích hợp hơn 50 tỷ bóng bán dẫn trên một chip đơn. Tuy nhiên, nhiều thách thức cần được giải quyết khi một số lượng lớn các bóng bán dẫn như vậy được tích hợp trên một con chip như phát xạ nhiệt, tiêu thụ điện năng và tắc nghẽn trong quá trình truy cập bộ nhớ. Do đó, các kiến trúc đa lõi đồng nhất và không đồng nhất đã được giới thiệu để tiếp tục cải thiện hiệu năng hệ thống và cho phép sử dụng số lượng lớn các bóng bán dẫn như vậy. So với các hệ thống đa lõi đồng nhất, các hệ thống không đồng nhất cung cấp nhiều sức mạnh tính toán hơn trong khi tiêu thụ năng lượng hiệu quả hơn do sử dụng các lõi chuyên dụng cho các chức năng cụ thể.

Là một trong những xu hướng chính trong đa lõi không đồng nhất, hệ thống tăng tốc phần cứng được xem như hướng tiếp cận chính để tiếp tục cải thiện hiệu suất trong tương lai. Trong các hệ thống như vậy, thường có một bộ xử lý đa dụng (General Purpose Processor – GPP) hoạt động như một bộ xử lý chủ và một hoặc nhiều bộ tăng tốc phần cứng (còn gọi là nhân) hoạt động như bộ đồng xử lý để tăng tốc xử lý cho các chức năng tính toán chuyên sâu của ứng dụng chạy trên bộ xử lý chủ. Do sự kết hợp của cả GPP và nhân đặc trưng cho từng ứng dụng cụ thể, hệ thống tăng tốc phần cứng tiết kiệm năng lượng hơn và có hiệu suất cao hơn GPP trong khi vẫn cung cấp mức độ linh hoạt đáng kể.

Rõ ràng, các nhân tăng tốc và hành vi giao tiếp giữa chúng là khác nhau từ ứng dụng này sang ứng dụng khác. Một ứng dụng cụ thể cần có một kiến trúc kết nối cụ thể chuyên biệt cho các đặc tính giao tiếp dữ liệu của nó. Kết nối cụ thể phải có hiệu suất tối ưu trong khi vẫn giữ mức sử dụng tài nguyên phần cứng tối thiểu. Do đó, chúng tôi đề xuất một phương pháp thiết kế tự động lấy các đặc tính giao tiếp dữ liệu của một ứng dụng nhờ vào phương pháp thống kê để hiện thực một hệ thống tăng tốc phần cứng cho ứng dụng có kết nối lai. Kết nối lai được cung

cấp phù hợp nhất cho việc hỗ trợ các mẫu giao tiếp bên trong ứng dụng trong khi sử dụng tài nguyên phần cứng càng thấp càng tốt.

Đã biết thông tin về những hệ thống tăng tốc phần cứng, tuy nhiên các hệ thống này đều chỉ sử dụng một loại kết nối thay vì sử dụng hệ thống kết nối lai gồm nhiều loại kết nối khác nhau.

Hệ thống tăng tốc phần cứng Molen (công bố trong "The molen polymorphic processor"), Bộ xử lý Warp (công bố trong "Design and implementation of a microblaze-based warp processor"), Hệ thống IMORC (công bố trong "IMORC: An infrastructure and architecture template for implementing high-performance reconfigurable FPGA accelerators"), Hệ thống PowerEN của IBM, Hệ thống FUSE (công bố trong "Fuse: Front-end user framework for o/s abstraction of hardware accelerators"), Hệ thống được đề xuất bởi C. Pilato (công bố trong "On the automatic integration of hardware accelerators into fpga-based embedded systems") và Hệ thống tăng tốc phần cứng được đề xuất bởi A. Canis (công bố trong "Legup: An open-source high-level synthesis tool for fpga-based processor/accelerator systems") chỉ sử dụng loại kết nối bus cho kết nối giữa bộ xử lý chủ và các nhân phần cứng.

Hệ thống MORPHEUS (được công bố trong "The morpheus heterogeneous dynamically reconfigurable platform"), Hệ thống tăng tốc phần cứng được đề xuất bởi E. Chung (được công bố trong "Coram: An in-fabric memory architecture for fpga- based computing") và Hệ thống P2012 (được công bố trong "P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator") chỉ sử dụng mạng kết nối trên chip (Network-on-Chip) để kết nối bộ xử lý chủ và các nhân phần cứng.

Hệ thống Convey, Hệ thống IBM Power 8, Hệ thống Microsoft Catapult và Hệ thống tăng tốc phần cứng của Intel (công bố trong "High-performance, energy-efficient platforms using in-socket fpga accelerators") sử dụng bộ nhớ dùng chung để trao đổi dữ liệu giữa bộ xử lý chủ và nhân hoặc giữa các nhân.

Hệ thống được đề xuất bởi B. Betkaoui (công bố trong "A framework for fpga acceleration of large graph problems: Graphlet counting case study") và Hệ thống được đề xuất bởi J. Cong (công bố trong "Optimization of interconnects between accelerators and shared memories in dark silicon") sử dụng chuyển mạch (crossbar) để kết nối giữa bộ xử lý chủ và các nhân phần cứng.

Bản chất kỹ thuật của công trình

Công trình được thực hiện nhằm giải quyết vấn đề kỹ thuật nêu trên. Mục đích chính của công trình là đề xuất một phương pháp hỗ trợ cho việc thiết kế hệ thống tăng tốc phần cứng với kiến trúc kết nối lai. Phương pháp được xây dựng thành một khung sườn để hỗ trợ người sử dụng. Khung sườn cung cấp một giao diện người dùng trực quan, thực hiện nhận dữ liệu đầu vào là tập tin mã nguồn bằng ngôn ngữ C. Khung sườn cung cấp khả năng đưa ra các thống kê về thời gian hoạt động cũng như đặc tính giao tiếp dữ liệu của các thành phần trong ứng dụng được đặc tả bằng ngôn ngữ C. Khung sườn cũng được ra các kết quả kết quả thiết kế kiến trúc kết nối lai

dựa trên đặc tính giao tiếp dữ liệu. Cuối cùng, khung sườn đưa ra tập tin cấu hình hệ thống phần cứng nhằm xây dựng hệ thống tăng tốc phần cứng trên các nền tảng tính toán.

Để thực hiện mục đích này, nền tảng FPGA và các công cụ hỗ trợ tổng hợp FPGA được sử dụng như một phần của hệ thống được thiết kế. Cụ thể là việc sử dụng công cụ tổng hợp Quartus được cung cấp bởi hãng Intel Altera cho các chip FPGA tương ứng của hãng, công cụ tổng hợp ISE, Vivado của Xilinx cho các chip tương ứng của Xilinx. Ngoài ra, phần mềm thiết kế hoạt động trên hệ điều hành Linux hỗ trợ đầy đủ các thư viện được sử dụng trong chương trình.

Khung sườn nhận dữ liệu đầu vào là tập tin mã nguồn phát triển bằng ngôn ngữ cấp cao C. Các công cụ QUAD và gprof được sử dụng để tiến hành phân tích ứng dụng để rút trích được các đặc tính giao tiếp dữ liệu và thời gian thực thi của các phần trong mã nguồn đầu vào.

Dựa trên các kết quả phân tích này, khung sườn cung cấp một mô-đun phân hoạch phần cứng phần mềm để phân tách các phần sẽ được hiện thực trên phần cứng và các phần tiếp tục được thực thi bằng phần mềm trên máy chủ. Mô-đun nhận đầu vào là kết quả phân tích do công cụ QUAD và gprof sinh ra để đưa ra được kết quả phân hoạch.

Khung sườn được tích hợp mô-đun sinh kiến trúc kết nối lai để tạo ra các kết nối phù hợp nhất cho các phần của ứng dụng sẽ hiện thực trên phần cứng. Đầu vào để thực hiện mô-đun này là đặc tính giao tiếp dữ liệu đã được rút trích ở bước trước. Kết quả đầu ra của mô-đun là một kiến trúc kết nối phù hợp với các phần của ứng dụng sẽ được xây dựng trên phần cứng.

Bên cạnh đó, khung sườn được tích hợp các công cụ sinh mã đặc tả phần cứng để chuyển đổi các phần sẽ hiện thực trên phần cứng từ ngôn ngữ C sang mã đặc tả phần cứng. Đầu vào của các công cụ sinh mã này là mã nguồn bằng ngôn ngữ C cho phần ứng dụng sẽ được hiện thực trên phần cứng. Đầu ra của bước này sẽ là mã đặc tả phần cứng tương ứng khả hiện trên các thiết bị phần cứng.

Cuối cùng, dưới sự hỗ trợ của các công cụ tổng hợp Quartus cho thiết bị của Intel Altera và ISE, Vivado cho thiết bị của Xilinx, toàn bộ đặc tả phần cứng cho các phần ứng dụng được hiện thực trên phần cứng và kiến trúc kết nối lai sẽ được tổng hợp để tạo ra tập tin cấu hình thiết bị phần cứng tương ứng.

Mô tả vắn tắt các hình vẽ

Hình 1 mô tả phương pháp hỗ trợ thiết kế mà chúng tôi đề xuất để hiện thực một ứng dụng cụ thể trên các hệ thống tăng tốc phần cứng với kết nối lai.

Hình 2 mô tả kiến trúc tổng quan hệ thống tăng tốc phần cứng dùng kết nối lai.

Hình 3 mô tả mô hình tính toán song song nhiều nhân cho một hàm.

Hình 4 mô tả mô hình tính toán xử lý ống (pipeline) nhiều nhân cho nhiều hàm.

Hình 5 mô tả thuật toán xây dựng tự động kết nối lai cho hệ thống tăng tốc phần cứng.

Mô tả chi tiết giải pháp hữu ích

Hình 1 mô tả phương pháp hỗ trợ thiết kế mà chúng tôi đề xuất để hiện thực một ứng dụng cụ thể trên các hệ thống tăng tốc phần cứng với kết nối lai. Các bước tiến hành và cách thức thực hiện từng bước được mô tả như sau:

1. Lập hồ sơ: Ứng dụng được lập hồ sơ bằng công cụ (phương tiện) *QUAD* để trích xuất các đặc tính giao tiếp dữ liệu của ứng dụng và bằng công cụ *gprof* để xác định các hàm tính toán cần được tăng tốc. Kết quả của bước này được sử dụng trong bước "Phân hoạch phần cứng/phần mềm" tiếp theo cũng như bước "Kết nối lai".

2. Phân hoạch phần cứng/phần mềm: Mục tiêu chính của việc Phân hoạch phần cứng/phần mềm trong các hệ thống phần cứng tăng tốc không đồng nhất là cải thiện hiệu năng hệ thống, từ đó, đạt được mức độ tăng tốc cao hơn. Để đạt được mục tiêu này, các phần của chương trình có thời gian thực thi lớn (các hàm tính toán chuyên sâu) thường được ánh xạ lên phần cứng, trong khi các phần cần ít thời gian tính toán được thực hiện trên máy chủ. Bước này nhận đầu vào là kết quả phân tích của bước "Lập hồ sơ" để chọn lựa các phần của ứng dụng sẽ được tăng tốc dựa trên thời gian thực thi của nó. Kết quả của bước này là một bảng phân hoạch phần được thực thi trên phần cứng và phần được thực thi trên máy chủ. Kết quả của bước này sẽ là đầu vào cho hai bước "Kết nốt lai" và "Sinh phần cứng" tiếp theo.

3. Kết nối lai: Sử dụng các hồ sơ chi tiết của các đặc tính giao tiếp dữ liệu sinh ra trong bước "Lập hồ sơ", một nhân phần cứng tăng tốc sẽ biết chính xác nhân nào sẽ sử dụng các kết quả đầu ra của nó. Do đó, nhân có thể phân phối đầu ra của nó trực tiếp đến các nhân sử dụng khi kết quả đã sẵn sàng (thay vì chuyển đầu ra của nó trở lại máy chủ như trong các mô hình thực thi thông thường). Để hỗ trợ mô hình này khi triển khai lên một hệ thống phần cứng tăng tốc hiện có, bên cạnh cơ sở hạ tầng truyền thông có sẵn, một kết nối lai giữa các nhân sẽ được triển khai. Kết quả của bước này là một đặc tả các kết nối sẽ được sử dụng để xây dựng hệ thống. Kết quả của bước này cũng sẽ được sử dụng ở bước "Sinh phần cứng" tiếp theo.

4. Sinh ra phần cứng: Như đã nói ở trên, chúng tôi sử dụng các nền tảng điện toán có thể tái cấu hình làm nền tảng chủ yếu. Các công cụ tổng hợp cấp cao như *Xilinx Vivado*, *DWARV*, v.v., được sử dụng để tạo các đặc tả phần cứng có thể tổng hợp cho các nhân từ các đặc tả bằng ngôn ngữ cấp cao tương ứng. Kết quả của bước này là một tập tin dùng để cấu hình các hệ thống phần cứng tương thích và được sử dụng trong bước cuối cùng "Triển khai trên nền tảng phần cứng".

5. Triển khai trên nền tảng phần cứng: Ở bước cuối cùng này, tất cả các thành phần trong thiết kế được tổng hợp và triển khai trên nền tảng tính toán tái cấu hình bằng các công cụ của một nhà cung cấp cụ thể như Xilinx khi triển khai trên nền tảng Xilinx hoặc công cụ Convey PDK khi sử dụng máy Convey. Phần mềm được thực thi trên máy chủ trong khi phần cứng tăng tốc được triển khai trên thiết bị tính toán tái cấu hình. Cơ sở hạ tầng truyền thông của hệ thống đảm bảo việc giao tiếp dữ liệu giữa các bộ tăng tốc phần cứng và phần mềm, trong khi kết nối lai của chúng tôi sẽ thực hiện việc liên kết các nhân.

Hình 2 mô tả kiến trúc tổng quan hệ thống tăng tốc phần cứng với kết nối lai. Trong kiến trúc này, bộ xử lý chủ có thể là CPU hiệu suất cao đa dụng (ví dụ: CPU Intel x86) hoặc bộ xử lý nhúng (ví dụ: Xilinx PowerPC) hoặc bộ xử lý mềm (ví dụ: MicroBlaze hoặc Nios). Các nhân của hệ thống tăng tốc phần cứng được triển khai dưới các nền tảng phần cứng như FPGA, bộ xử lý tín hiệu số (DSP), GPU (Bộ xử lý đồ họa), v.v. Trong khi bộ xử lý chủ sử dụng bộ nhớ chính để lưu trữ dữ liệu của ứng dụng, các nhân có bộ nhớ cục bộ của chúng để lưu trữ dữ liệu cục bộ (bộ đệm dữ liệu) để cải thiện khả năng song song giữa các nhân. Trong kiến trúc này, chia sẽ bộ nhớ cục bộ và mạng kết nối trên chip được dùng để kết nối các nhân tính toán tạo thành kiến trúc kết nối lai.

Hình 3 mô tả mô hình tính toán nhiều nhân cho một hàm (song song về dữ liệu). Song song về dữ liệu là một kịch bản thực thi trong đó dữ liệu được chia nhỏ thành các phân đoạn và các nhân xử lý song song đảm nhận các phân đoạn đó. Nói cách khác, một hàm tính toán chuyên sâu có thể được tăng tốc bởi một số lượng các nhân đồng thời. Mỗi nhân xử lý từng phân đoạn dữ liệu. Giả sử rằng một hàm tính toán chuyên sâu có n nhân tăng tốc, dữ liệu đầu vào cho hàm được phân chia thành n phân đoạn. Việc giảm thời gian xử lý của hàm này so với một nhân xử lý duy nhất là $\Delta_{dp} = \frac{\tau_i(n-1)}{n} - O$, trong đó τ_i là thời gian để xử lý toàn bộ dữ liệu chỉ với một nhân và O là chi phí phát sinh khi xử lý song song ở mức dữ liệu. Chi phí này phụ thuộc vào thuật toán được áp dụng và sinh ra do các dữ liệu bổ sung cần được xử lý để đạt được kết quả chính xác cho từng phân đoạn.

Hình 4 mô tả mô hình tính toán xử lý ống (pipeline) nhiều nhân cho một hàm (song song về lệnh). Song song ở mức lệnh là một kịch bản thực thi trong đó các nhân tăng tốc của hàm tạo thành một đường ống để xử lý một luồng chứa các phân đoạn dữ liệu. Mỗi nhân phụ trách của mỗi chức năng được tăng tốc có vai trò như một giai đoạn của đường ống. Các phân đoạn dữ liệu được truyền qua từng giai đoạn đó. Khác với kịch bản xử lý tuần tự, trong xử lý song song ở mức lệnh, các nhân tham gia làm việc cùng một lúc. Độ sâu của đường ống là số lượng phân đoạn dữ liệu được xử lý đồng thời. Giả sử rằng có n giai đoạn xử lý, tổng thời gian thực thi trong kịch bản tuần tự khi kết nối lai được áp dụng sẽ đảm nhiệm việc truyền dữ liệu giữa các nhân được định nghĩa theo phương trình:

$$T_{serial} = \sum_{i=0}^{n-1} \tau_i + \sum_{i=0}^{n-1} (D_{i(in)}^{H} + D_{i(out)}^{H})\theta \qquad (1)$$

Trong đó $D_{i(in)}^{H}$ và $D_{i(out)}^{H}$ là lượng dữ liệu đầu vào của nhân được tạo bởi máy chủ và dữ liệu đầu ra của nhân được sử dụng bởi máy chủ.

Khi song song ở mức lệnh được khai thác với độ sâu m (giả sử rằng $m \geq n$), tổng thời gian thực hiện được xấp xỉ bởi phương trình:

$$
\begin{aligned}
T_{pipeline} = & \left(\frac{\tau_0}{m} + O_0\right) + \max_{0 \le i \le 1}\left(\frac{\tau_i}{m} + O_i\right) + \ldots \\
& + \max_{0 \le i \le n-1}\left(\frac{\tau_i}{m} + O_i\right) \times (m - n + 1) \\
& + \max_{1 \le i \le n-1}\left(\frac{\tau_i}{m} + O_i\right) + \ldots + \max_{n-2 \le i \le n-1}\left(\frac{\tau_i}{m} + O_i\right) \\
& + \left(\frac{\tau_{n-1}}{m} + O_{n-1}\right) + \sum_{i=0}^{n-1}\left(D^H_{i(in)} + D^H_{i(out)}\right)\theta
\end{aligned}
\tag{2}
$$

trong đó O_i là chi phí phát sinh. Sự song song ở mức lệnh có lợi khi $T_{pipeline} < T_{serial}$.

Hình 5 mô tả thuật toán được đề xuất sử dụng đặc tính giao tiếp dữ liệu của ứng dung để triển khai ứng dụng đó trên nền tảng tăng tốc phần cứng. Kết quả của thuật toán là một hệ thống tăng tốc phần cứng với một kết nối lai được tối ưu hóa về thời gian giao tiếp trong khi vẫn giữ được mức sử dụng tài nguyên phần cứng tối thiểu để thiết lập các kết nối. Kết nối lai bao gồm các thành phần kết nối được đề cập ở trên. Trước tiên, thuật toán chọn các chức năng tính toán chuyên sâu nhất và phù hợp để tăng tốc trên phần cứng (tức là, các chức năng có thể được tổng hợp trên phần cứng) (dòng 1). Các hàm tính toán chuyên sâu được xem xét áp dụng khả năng song song về dữ liệu nếu được chấp nhận (dòng 2-8). Sau đó, thuật toán sử dụng công cụ lập hồ sơ QUAD để tạo hồ sơ định lượng mức độ truyền thông dữ liệu của ứng dụng (dòng 9). Dựa trên hồ sơ này, một kết nối lai hiệu quả được chỉ định. Trong thuật toán này, bộ nhớ cục bộ dùng chung (dòng 10-15) được xem xét trước. Bước tiếp theo là ánh xạ tất cả các nhân còn lại không được kết nối bằng giải pháp bộ nhớ cục bộ được chia sẻ vào mạng kết nối trên chip. Cuối cùng, cơ chế song song ở mức lệnh được xem xét để tiếp tục giảm thời gian thực thi nếu được chấp nhận (dòng 17).

YÊU CẦU BẢO HỘ

1. Phương pháp thiết kế tự động lấy các đặc tính giao tiếp dữ liệu của một ứng dụng để hiện thực một hệ thống tăng tốc phần cứng cho ứng dụng kết nối lai bằng cách tạo ra một khung sườn hỗ trợ thiết kế tăng tốc phần cứng, phương pháp bao gồm các bước:

lặp hồ sơ: sử dụng công cụ *QUAD* để trích xuất các đặc tính giao tiếp dữ liệu của ứng dụng và công cụ *gprof* để xác định thời gian thực thi của các phần khác nhau trong ứng dụng;

phân hoạch phần cứng/phần mềm: các phần của ứng dụng có thời gian thực thi lớn đã được xác ở bước trước được lựa chọn ánh xạ lên phần cứng, trong khi các phần cần ít thời gian tính toán được lựa chọn thực hiện bằng phần mềm;

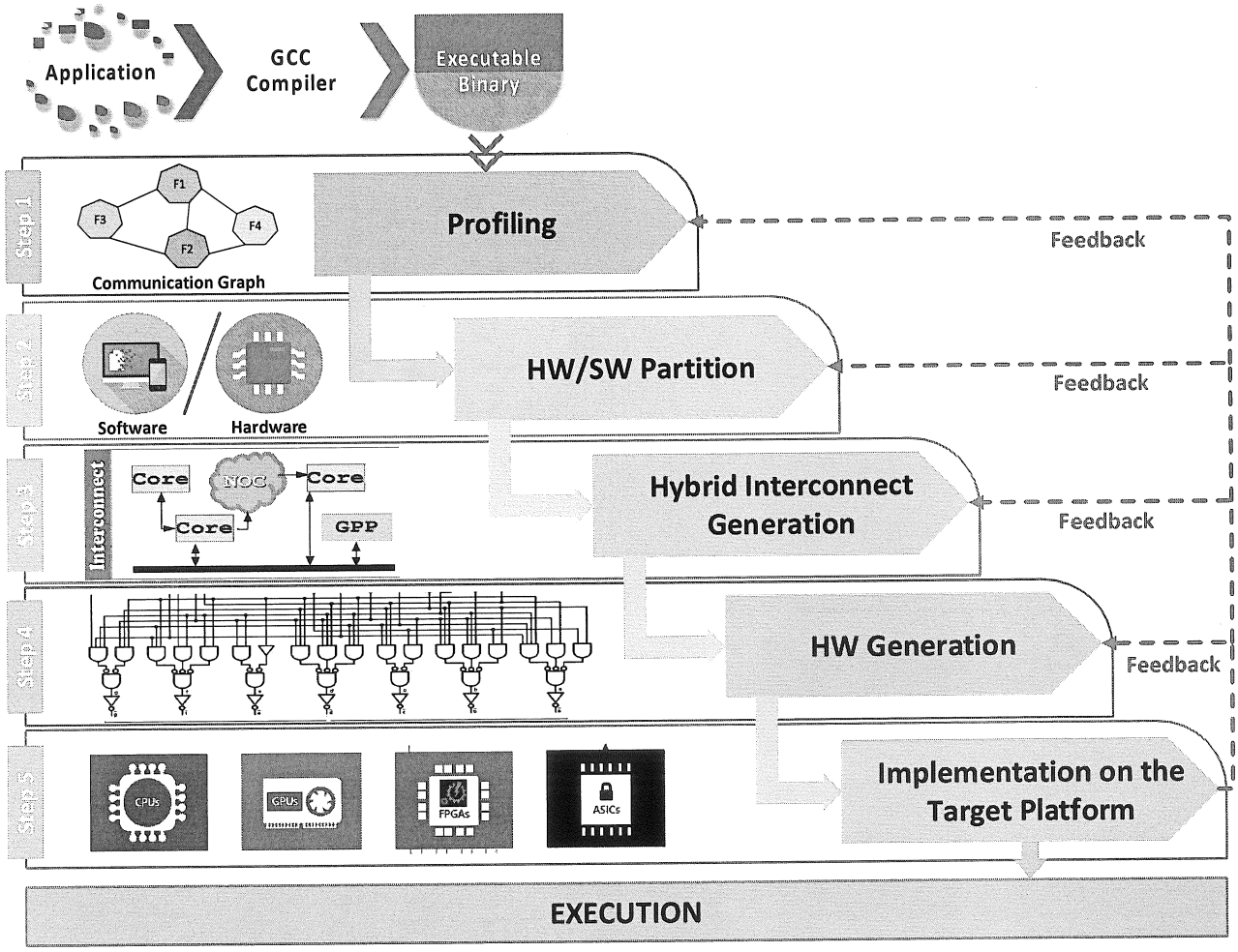
kết nối lai: sử dụng các hồ sơ chi tiết của các đặc tính giao tiếp dữ liệu để kết nối các lõi phần cứng tăng bằng một kiến trúc kết nối phù hợp có thể gồm mạng trên chip, đường dữ liệu chung và chia sẻ bộ nhớ;

sinh ra phần cứng: các công cụ tổng hợp cấp cao *Xilinx Vivado HLS* hay *DWARV*, được sử dụng để tạo các đặc tả phần cứng có thể tổng hợp cho các lõi phần cứng từ các đặc tả bằng ngôn ngữ cấp cao tương ứng;

triển khai trên nền tảng phần cứng: tất cả các thành phần trong thiết kế được tổng hợp và triển khai trên nền tảng tính toán tái cầu hình bằng các công cụ hãng sản xuất nền tảng phần cứng dùng để hiện thực ứng dụng.

2. Phương pháp theo điểm 1, trong đó tích hợp giải thuật sinh tự động kiến trúc kết nối lai tuỳ theo đặc tính giao tích dữ liệu của các phần trong ứng dụng.

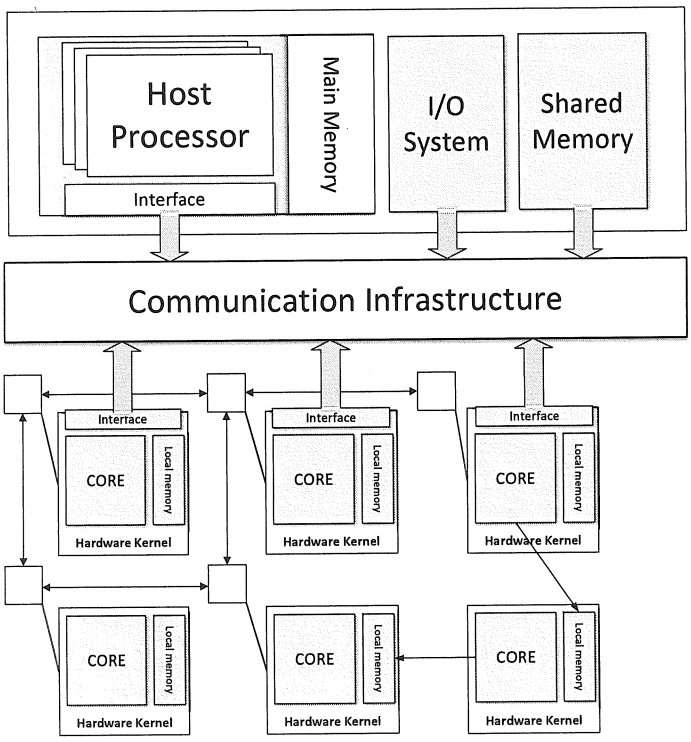
3. Phương pháp theo điểm 1, trong đó tích hợp kiến trúc kết nối lai bao gồm bộ nhớ cục bộ chia sẻ và mạng kết nối trên chip có cấu hình phù hợp với đặc tính giao tiếp dữ liệu của ứng dụng.



Chú thích:

- Application: Ứng dụng
- Executable Binary: Mã nhị phân thực thi
- Profiling: Lập hồ sơ
- Hybrid Interconnect Generation: Sinh kết nối lai
- GCC Compiler: Trình biên dịch GCC
- Communication graph: đồ thị giao tiếp
- HW/SW Partition: phân hoạch phần cứng/phần mềm
- HW Generation: Sinh phần cứng
- Implementation on Target Platform: Triển khai trên nền tảng phần cứng

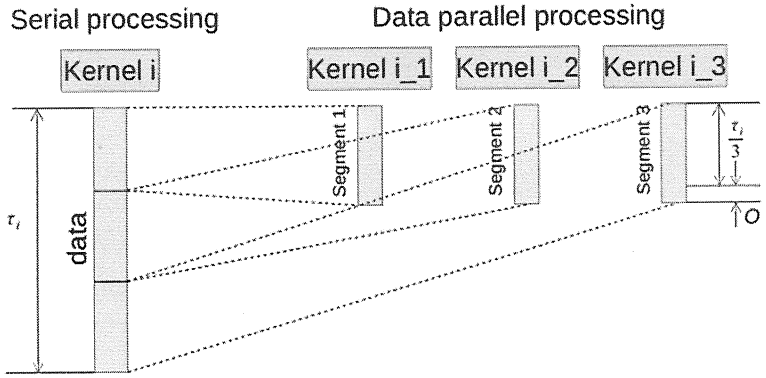
Hình 1



Chú thích:

- Host processor: Máy chủ
- Main memory: Bộ nhớ chính
- I/O system: Hệ thống vào ra
- Shared memory: Bộ nhớ chia sẽ
- Communication Infrastructure: Hệ thống giao tiếp
- Interface: Giao tiếp
- Hardware kernel: Lõi phần cứng
- Local memory: Bộ nhớ cục bộ

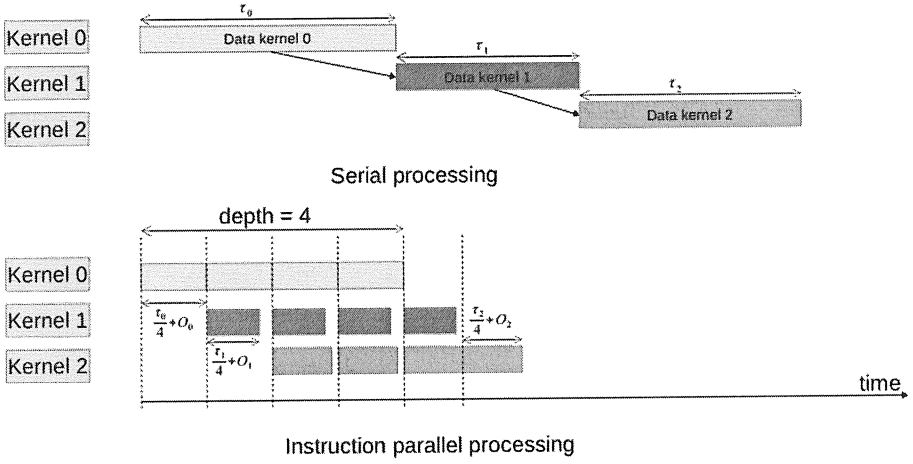
Hình 2



Chú thích:

- Serial processing: xử lý tuần tự
- Data parallel processing: xử lý song song mức dữ liệu
- Segment: phần ứng dụng
- Data: dữ liệu
- Kernel: lõi tính toán

Hình 3



Chú thích:

- Serial processing: xử lý tuần tự
- Instruction parallel processing: xử lý song song mức lệnh
- Segment: phần ứng dụng
- Data: dữ liệu
- Kernel: lõi tính toán
- Data kernel: dữ liệu cho lõi tính toán
- Depth: độ sâu

Hình 4

ALGORITHM 1: Custom interconnect design with NoC

Input: Application source code
Output: The most optimized interconnect
$L_{hw} \leftarrow$ List of the most computationally intensive functions suitable to implement on hardware; 1
repeat 2
  for *each HW in* L_{hw} **do** 3
    if *HW satisfies the data parallelism solution* ($\Delta_{dp} > 0$) *& resource is available* **then** 4
      Replicate HW in L_{hw} 5
    end 6
  end 7
until L_{hw} *does not change*; 8
$G \leftarrow$ Quantitative data communication profiling for functions in L_{hw}; 9
for *each communication* $[HW_i \rightarrow HW_j : D_{ij}]$ *in* G **do** 10
  if $D^K_{j(in)} = D_{ij}$ **then** 11
    Apply the shared local memory solution for HW_i and HW_j; 12
    Remove HW_i from L_{hw}; 13
  end 14
end 15
Map all HW in L_{hw} to the NoC using adaptive mapping algorithm; 16
Apply the instruction parallelism solution if $T_{pipeline} < T_{serial}$; 17

Hình 5