triển khai các ứng dụng LÊN nền tảng noc DỰA TRÊN chiến lược chọn vùng

MAPPING DYNAMIC APPLICATIONS ON NOC PLatforms USING region selection strategy

Nguyễn Văn Cường1, Nguyễn Trọng Bằng2, Nguyễn Hữu Tuấn Huy3, Nguyễn Hữu Long4

1Trường Đại học Công nghiệp Tp.HCM; nguyenvancuong@iuh.edu.vn

2Công ty giải pháp mạng Dasan Việt Nam; [trongbang108@gmail.com](mailto:trongbang108@gmail.com)

3Trường Đại học Đông Á; huynht@donga.edu.vn

4Học viện Ngân hàng Hà Nội; longkbg@gmail.com

**Tóm tắt -** Tích hợp nhiều bộ xử lý trên cùng một chip đơn là một xu hướng mới để cung cấp hiệu năng cao hơn cho hệ thống. Ánh xạ tại thời gian chạy cho mạng trên chip (Network on Chip) là thách thức lớn khi xem xét thứ tự của các ứng dụng đến là chưa biết trước. Trong bài báo này, chúng tôi sử dụng một chiến lược chọn vùng gần lồi mới để giải quyết vấn đề ánh xạ cho các ứng dụng lên nền tảng mạng trên chip đồng nhất. Kết quả mô phỏng cho thấy rằng khi sử dụng chiến lược chọn vùng gần lồi để ánh xạ các tác vụ của ứng dụng vào nó giúp cải thiện thông lượng trung bình lên đến 17.8% và cắt giảm 6.5% độ trễ trung bình trong mạng so với giải pháp ánh xạ không sử dụng chọn vùng.

**Từ khóa –** Mạng trên chip; chọn vùng; ánh xạ động; thiết bị nhúng; đa xử lý.

**Abstract -** Integrating multiple processors on a single chip is a new trend to provide better performance for the system. Mapping at runtime for Network on Chip is a big challenge when considering the sequence of the incoming applications is unknown in advance. In this paper, we use one new convex region selection strategy to solve applications mapping problem into homogeneous Network on Chip platforms. The simulation results demonstrate that by using the convex region selection strategy to map tasks of application, this can help network improve throughput up to 17.8% and reduce 6.5% on average latency over network comparing with using non-clustering solution.

**Key words –** Network on chip; region selection; dynamic mapping; embedded systems; multi-core.

# Đặt vấn đề

Với những tính ưu việt của công nghệ bán dẫn hiện nay, số lượng các ứng dụng được tích hợp lên chip ngày càng tăng. Kiến trúc Bus truyền thống không còn phù hợp với các yêu cầu của hướng công nghệ mới này [1-2]. Kiến trúc mạng trên chip (NoC) đã được đề xuất và được xem như là một giải pháp thay thế cho kiến trúc Bus. Kiến trúc NoC cung cấp một cơ sở hạ tầng truyền thông có hiệu năng cao và dễ dàng tích hợp một số lượng lớn các lõi IP (Intellectual Property) lên một hệ thống trên chip (System-on-Chip: SoC) [3-4]. Vì vậy, kiến trúc NoC đã được lựa chọn để thiết kế trong hầu hết các thiết bị nhúng ngày nay. Các thiết bị nhúng ngày càng được tích hợp nhiều ứng dụng, đặc biệt trong miền ứng dụng đa phương tiện. Vì vậy, đặc tính linh hoạt và thích nghi với sự thay đổi của các ứng dụng tại thời gian chạy trong các thiết bị nhúng sử dụng kiến trúc NoC là rất quan trọng. Điều này giúp hệ thống đạt được hiệu năng tốt hơn khi triển khai các ứng dụng lên nó.

Ánh xạ ứng dụng là một trong những vấn đề quan trọng trong nghiên cứu NoC [5]. Ánh xạ liên quan đến vấn đề đặt các tác vụ của một ứng dụng vào các phần tử xử lý của NoC. Một vài cách tiếp cận ánh xạ tại lúc thiết kế hiệu quả đã được đề xuất trong [6-7]. Tuy nhiên, việc tìm kiếm các giải pháp tối ưu cho ánh xạ tại thời gian chạy là khá khó khăn vì thứ tự xuất hiện của các ứng dụng chưa được biết trước [5]. Hầu hết các thuật toán ánh xạ tại thời gian chạy trong [8-10] cố gắng tìm một giải pháp tối ưu cho các ứng dụng đến đang tồn tại trên hệ thống. Một số nghiên cứu khác sử dụng phương pháp cách tiếp cận “tham lam” (greedy) để giải quyết vấn đề này. Tuy nhiên, cách tiếp cận "tham lam" có thể không luôn luôn mang lại giải pháp tối ưu tốt nhất. Giải pháp chọn vùng gần lồi để ánh xạ các tác vụ của ứng dụng vào nó là một cách tiếp cận khác. Ý tưởng chọn vùng lồi được giới thiệu đầu tiên trong [11]. Chiến lược này đầu tiên chọn một vùng gần lồi và sau đó ánh xạ ứng dụng vào vùng đã được chọn. Bằng cách lựa chọn vùng gần lồi, quá trình xử lý tách biệt được giảm thiểu. Vì vậy, các ứng dụng mới có thể dễ dàng đưa thêm vào hệ thống. Bằng cách ánh xạ tác vụ một cách hợp lý hơn, chúng tôi có thể cải thiện hiệu năng cho hệ thống so với cách tiếp cận không sử dụng chọn vùng.

Trong nghiên cứu này, chúng tôi tập trung vào vấn đề ánh xạ các ứng dụng vào nền tảng NoC đồng nhất tại thời gian chạy, sử dụng chiến lược chọn vùng gần lồi để cải thiện hiệu năng cho toàn mạng. Thêm vào đó, chiến lược này cũng cho phép thêm các ứng dụng mới vào hệ thống một cách dễ dàng.

Phần còn lại của bài báo được tổ chức như sau: Trong mục 2, chúng tôi trình bày các định nghĩa và xây dựng vấn đề ánh xạ. Một chiến lược chọn vùng mới và một khuôn khổ cho vấn đề ánh xạ sử dụng chiến lược chọn vùng được giới thiệu trong mục 3. Kết quả mô phỏng và thảo luận sẽ được đưa ra trong mục 4. Cuối cùng, kết luận và công việc trong tương lai được trình bày trong mục 5.

# Các định nghĩa và xây dựng vấn đề

## Mô hình ứng dụng

Các ứng dụng được biểu diễn dưới dạng đồ thị tác vụ như Hình 1. Đồ thị tác vụ của một ứng dụng được mô hình như sau:  trong đó,  là tập các tác vụ có trên ATG,  là số tác vụ của ứng dụng. Mỗi task có các thông số như: Thông số nhận dạng, địa chỉ tác vụ. là tập hợp các cạnh trên ATG. Mỗi cạnh  có các thông số như độ trễ, thông lượng truyền thông, và là hàm trọng lượng đại diện cho tốc độ truyền thông giữa các tác vụ.



***Hình 1.*** *Mô hình hệ thống*

## Mô hình kiến trúc phần cứng

Kiến trúc phần cứng đồng nhất bao gồm các phần tử xử lý giống nhau gọi là PE (Processing Element). Mỗi phần tử xử lý bao gồm một đơn vị xử lý, một bộ nhớ cache, bộ nhớ nội bộ và đơn vị điều khiển. Các phần tử xử lý được nối với nhau thông qua mạng truyền thông NoC. NoC bao gồm 3 thành phần chính: Bộ định tuyến, các liên kết vật lý và bộ giao tiếp mạng. Kiến trúc NoC sử dụng cấu trúc mạng 2D-Mesh, chuyển mạch gói, điều khiển luồng wormhole và giải thuật định tuyến XY (Hình 1). Trong các PE có một PE quản lý toàn cục, ký hiệu là GM, cũng sử dụng một đơn vị xử lý đa mục đích. GM chạy thuật toán ánh xạ tại thời gian chạy, cấp phát các tài nguyên cho các ứng dụng đến cũng như giải phóng tài nguyên phần cứng khi ứng dụng hoàn tất. GM được đặt ở góc dưới bên trái của mạng. Các PE còn lại thực hiện xử lý tác vụ khi GM cấp phát. Nền tảng NoC đồng nhất được định nghĩa như sau: trong đó,  là tập các PE,  là tập các đường liên kết vật lý.

Để tính số đường liên kết từ PE nguồn đến PE đích sau ánh xạ, thông số khoảng cách Manhattan sẽ được sử dụng trong nghiên cứu này. Khoảng cách Manhattan được định nghĩa là số đường liên kết nhỏ nhất mà một gói tin đi qua từ PE nguồn (pxy) đến PE đích (px’y’) và được tính bởi công thức kc((x,y),(x’,y’)) = |x’-x| + |y’-y|. Ví dụ: Nếu gửi một gói tin đi từ nguồn V1 đến đích V3 như trong Hình 1 thì khoảng cách Manhattan được tính là 2.

## Xây dựng vấn đề

Giả sử có M ứng dụng được đưa vào hệ thống đồng thời hoặc tuần tự tại lúc đang chạy và trạng thái của nền tảng NoC.

Tìm *R* vùng cận lồi tương ứng với *M* ứng dụng trên nền tảng NoC và hàm map() cho ứng dụng *Ai* vào vùng *Ri,* ,  với mục tiêu đạt được hiệu năng cao.

Sao cho thỏa mãn điều kiện và .

# Vấn đề ánh xạ tại thời gian chạy

Để triển khai các ứng dụng vào nền tảng NoC tại thời gian chạy chúng tôi thực hiện các bước sau: Đầu tiên tìm các vùng gần lồi cho các ứng dụng đến, sau đó sử dụng dụng thuật toán FF (First Fit) và NN (Nearest Neighbor) trong [9] để ánh xạ các ứng dụng vào các vùng được lựa chọn. Trong đó, thuật toán ánh xạ FF tìm các PE trống theo hàng trên nền tảng để đặt các tác vụ của ứng dụng vào chúng. Nếu không tìm được PE trống ở hàng hiện tại, thuật toán sẽ tiếp tục tìm PE trống ở hàng kế tiếp để đặt tác vụ. Quá trình này được lặp lại cho đến khi tất cả các tác vụ được ánh xạ vào nền tảng.

Thuật toán NN gán tác vụ đầu tiên đến PE khởi tạo đã được lựa chọn. Các tác vụ còn lại được ánh xạ đến các PE trống mà có khoảng cách Manhattan nhỏ nhất đến các PE đã được ánh xạ với tham chiếu đến vị trí khởi tạo. thuât toán NN cố gắng ánh xạ các tác vụ gần nhau trong một khu vực nhỏ.

## Chiến lược chọn vùng gần lồi

Chọn các vùng gần lồi trên nền tảng để ánh xạ các ứng dụng vào chúng là một giải pháp hiệu quả cho vấn đề ánh xạ, nó đã được chứng minh trong các nghiên cứu [8][11].

Tuy nhiên, có một vài hạn chế cần cải tiến, khắc phục trong các nghiên cứu này. Chiến lược chọn vùng trong [11] cố gắng tìm các PE trống trong nền tảng để tạo thành một vùng cho ứng dụng vào với ràng buộc tổng khoảng cách Manhattan là tối thiểu. Chiến lược giả thiết nền tảng đã có một hoặc một vài vùng đã tồn tại. Vùng mới được tạo ra bằng cách tham chiếu đến các vùng đã tồn tại. Do vậy, hiệu quả của vùng mới sẽ phụ thuộc vào các vùng đã tạo ra trước đó. Do vậy, chiến lược này có thể chọn một vùng không gần lồi. Ví dụ, một ứng dụng với 9 tasks được đưa vào nền tảng, nó có thể chọn một vùng 2x5 thay vì chọn một vùng 3x3. Tiếp theo, chúng ta xem xét cách tiếp cận trong [8]. Cách tiếp cận này tạo ra một vùng gần lồi cho ứng dụng vào dựa trên chiến lược chọn lựa một PE trống đầu tiên, sau đó tăng khoảng cách Manhattan để tìm ra các PE trống tiếp theo cho vùng xung quanh PE đầu tiên. Cách tiếp cận này có thể tìm ra một vùng gần lồi tối ưu. Tuy nhiên, nó cũng có thể sinh ra các vùng gần lồi không liền kề hoặc tăng sự phân tán các PE trên nền tảng khi có nhiều ứng dụng vào, gây bất lợi cho các cho các ứng dụng vào sau. Điều này có thể làm giảm hiệu năng cho toàn hệ thống.

Để vượt qua các hạn chế này, chúng tôi đề xuất một chiến lược chọn vùng gần lồi mới nhắm đến các mục tiêu như cấp phát tài nguyên linh hoạt, giảm thiểu khoảng cách truyền thông trung bình trong vùng đã chọn, tạo ra các vùng lồi liền kề tránh phân tán các PE trên nền tảng. Chiến lược được chia thành 3 bước: Bước 1, tính toán cấp phát số PE linh hoạt cho ứng dụng. Bước 2, tìm một vùng gần lồi chứa số PE lớn hơn hoặc bằng số PE ứng dụng yêu cầu dựa trên phương pháp quét theo góc hình học. Bước 3, sử dụng thêm thông số khoảng cách Manhattan tối thiểu để chọn ra một vùng gần lồi tối ưu.

Kỹ thuật chọn vùng dựa trên góc quét hình học kết hợp với khoảng cách Manhattan tối thiểu được giải thích chi tiết như sau: Xác định một PE là tâm hoặc cận tâm trên nền tảng và chọn PE này làm tâm quét. Để có thể quét hết tất cả các PE trên nền tảng, các góc quét thay đổi từ 0->3600 được sử dụng. Hướng quét có thể cùng chiều hoặc ngược chiều kim đồng hồ. Tương ứng với một góc quét xác định chúng ta sẽ tìm ra các PE sẵn có. Bằng cách thực hiện nhiều góc quét khác nhau, liên tục sẽ tìm được số lượng PE sẵn có cấp phát cho ứng dụng vào. Tiếp theo, chúng tôi sử dụng khoảng cách Manhattan tối thiểu để chọn ra một vùng tối ưu cho ứng dụng vào. Vì góc quét liên tục nên yếu tố phân mảnh, khoảng cách của các vùng lân cận được giảm đến mức tối thiểu. Sau mỗi lần quét trạng thái góc được cập nhật để sử dụng cho lần quét tiếp theo khi có ứng dụng mới đưa vào hệ thống.

## Luồng triển khai các ứng dụng vào nền tảng NoC

Chiến lược chọn vùng được đưa ra nhằm mục đích giảm thiểu khoảng cách Manhattan giữa các tác vụ trong ứng dụng và cố gắng tìm, đặt các tác vụ của ứng dụng vùng một vùng có diện tích nhỏ để tăng hiệu năng cho hệ thống. Trong nghiên cứu này, chúng tôi đề xuất một luồng thực hiện ánh xạ động các ứng dụng vào nền tảng NoC tại thời gian chạy có sử dụng chiến lược phân vùng. Luồng thiết kế được chỉ ra như Hình 2.



***Hình 2.*** *Luồng thực hiện ánh xạ sử dụng chiến lược chọn vùng*

Quá trình triển khai các ứng dụng vùng nền tảng được thực hiện theo các bước sau:

Bước 1: Nhập ứng dụng vào cần ánh xạ, kích thước nền tảng NoC.

Bước 2: Kiểm tra tài nguyên đủ để cấp phát cho ứng dụng hay không?

* Nếu đủ tài nguyên chuyển qua Bước 4 thực hiện chọn vùng cho ứng dụng.
* Không đủ tài nguyên, kiểm tra gở ứng dụng để giải phóng tài nguyên?

Bước 3: Nếu không đồng ý gở ứng dụng thì kết thúc chương trình. Nếu đồng ý gở ứng dụng, thực hiện gở ứng dụng để giải phóng tài nguyên và quay lại Bước 2.

Bước 4. Thực hiện chọn vùng và ánh xạ ứng dụng vào nền tảng.

Bước 5: Kiểm thực hiện ánh xạ tiếp không?

* Có, quay lại Bước 1.
* Không, kết thúc chương trình.

# Kết quả mô phỏng và thảo luận

## Thiết lập mô phỏng

Một nền tảng NoC đồng nhất có kích thước 6x6 dạng lưới 2 chiều được sử dụng. Nền tảng bao gồm một PE điều khiển làm nhiệm vụ chạy thuật toán ánh xạ và cấp phát tác vụ, các PE còn lại làm nhiệm vụ xử lý tác vụ. Ứng dụng thực và ứng dụng tổng hợp được lựa chọn để đánh giá. Các ứng dụng thực gồm VOPD (Video object plane decoder) 12 tác vụ, MWD (Multi-Window Display) 12 tác vụ, PIP (Picture In Picture) 8 tác vụ và một số ứng dụng tổng hợp được tạo ra bởi công cụ TGFF[12] có kích thước thay đổi từ 6 đến 12 tác vụ.

Các ứng dụng sẽ được ánh xạ lần lượt vào hệ thống tại thời gian chạy theo luồng thiết kế như trong Hình 2. Các thuật toán FF và NN được sử dụng để triển khai các ứng dụng lên nền tảng có chọn vùng và không chọn vùng. Mỗi kịch bản được chạy 50 lần cho 10 ứng dụng, thứ tự các ứng dụng đưa vào hệ thống một cách ngẫu nhiên. Các thông số hiệu năng như độ trễ, thông lượng được thống kê theo giá trị trung bình để đánh giá.

***Bảng 1.*** *Các thông số mô phỏng*

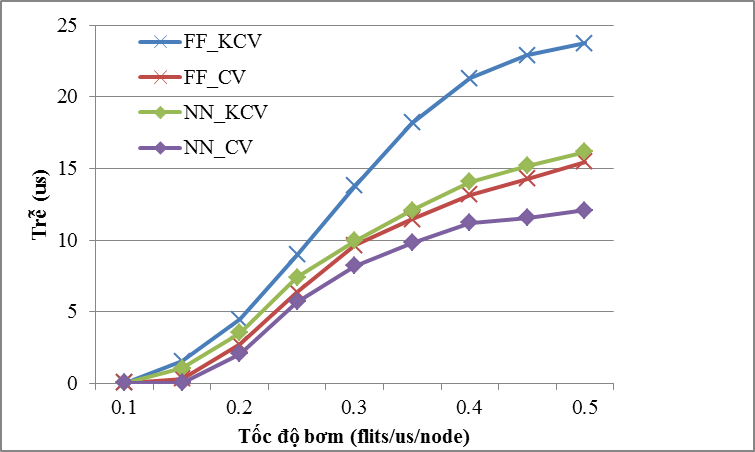
|  |  |
| --- | --- |
| **Thông số** | **Mô tả** |
| Kích thước nền tảng | 6x6 |
| Thuật toán định tuyến | XY |
| Điều khiển luồng | Wormhole |
| Kích thước gói tin (flit) | 8 |
| Tốc độ bơm gói tin (flits/us/node) | 0.1 ÷ 0.5 |
| Thời gian mô phỏng (chu kỳ) | 100000 |
| Thời gian khởi động (chu kỳ) | 10000 |
| Thuật toán ánh xạ | FF, NN và Chọn vùng |

## Công cụ mô phỏng NoCTweak trong [13] được mở rộng và bổ sung chức năng ánh xạ động để thực hiện các kịch bản mô phỏng của chúng tôi. Các thông số mô phỏng được thiết lập như Bảng 1.

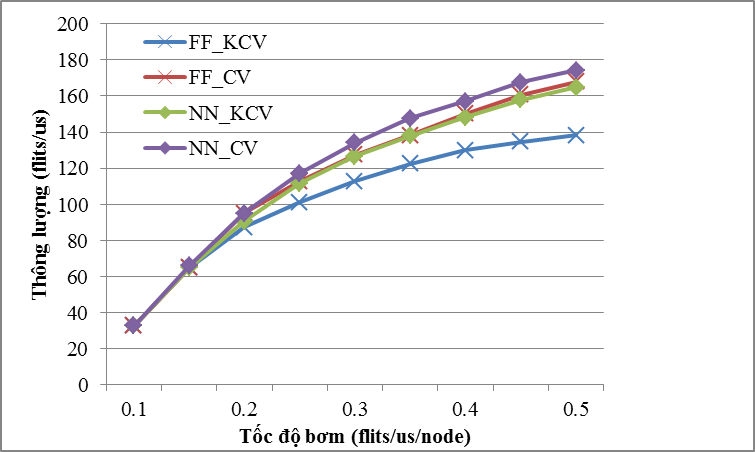
## Đánh giá hiệu năng cho các giải pháp

Chúng tôi sử dụng các thông số như độ trễ và thông lượng để đánh giá sự ảnh hưởng của các giải pháp chọn vùng và không chọn vùng đến hiệu năng của hệ thống.

Hình 3 chỉ ra độ trễ trung bình khi chạy thuật toán FF và NN trên nền tảng có chọn vùng và không chọn vùng. Rõ ràng khi triển khai các thuật toán trên nền tảng có chọn vùng, độ trễ được cắt giảm rõ rệt. Theo đó thuật toán FF và NN cắt giảm trễ trung bình lần lượt là 7.6% và 5.7% so với không sử dụng chọn vùng. Vấn đề này có thể giải thích như sau: Khi nền tảng không chọn vùng, không gian tìm kiếm các PE trống để ánh xạ tác vụ lên chúng lớn. Do vậy, các tác vụ có thể được ánh xạ đến các PE có khoảng cách Manhattan xa dẫn đến độ trễ tăng cao. Ngược lại, khi nền tảng được chọn vùng, các tác vụ được ánh xạ trong cùng một vùng nhỏ nên trễ truyền thông giữa các tác vụ sẽ giảm đáng kể.



***Hình 3.*** *Trễ trung bình khi thực hiện các thuật toán ánh xạ*



***Hình 4.*** *Thông lượng mạng khi thực hiện các thuật toán ánh xạ*

Tiếp theo, chúng tôi đánh giá ảnh hưởng của chiến lược chọn vùng đến thông lượng của hệ thống. Hình 4 cho thấy thông lượng đạt được khi chạy các thuật toán trên nền tảng có chọn vùng cao hơn so với không chọn vùng. Trong trường hợp cụ thể, sử dụng thuật toán NN để ánh xạ lên nền tảng có chọn vùng thông lượng được cải thiện trung bình đến 20.2% so với nền tảng không chọn vùng. Tương tự, thông lượng được cải thiện 15.5% khi sử dụng thuật toán FF.

Từ kết quả mô phỏng chúng ta có thể khẳng định rằng, sử dụng chiến lược chọn vùng để triển khai các ứng dụng lên nền tảng NoC tại thời gian chạy sẽ mang lại hiệu năng cao hơn không chọn vùng.

# Kết luận

Trong bài báo này, một chiến lược chọn vùng gần lồi mới được chúng tôi đề xuất. Việc sử dụng chiến lược chọn vùng để triển khai các ứng dụng lên nền tảng NoC tại thời gian chạy mang lại hiệu quả cao hơn so với không chọn vùng. Điều này rất quan trọng và cần thiết cho một hệ thống quản lý động. Trong tương lai, chúng tôi sẽ mở rộng chiến lược chọn vùng của chúng tôi cho một hệ thống không đồng nhất và đề xuất một thuật toán ánh xạ mới để tăng hiệu năng và tiết kiệm năng lượng cho toàn mạng.

Tài liệu tham khảo

J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 4th Edition, 4th ed. Morgan Kaufmann, 2006.

J. L. J. Liang, S. Swaminathan, and R. Tessier, “ASOC: a scalable, single-chip communications architecture,” in Proceedings 2000 International Conference on Parallel Architectures and Compilation Techniques (Cat. No.PR00622), 2000, pp. 37 – 46.

L. Benini and G. De Micheli, “Network on Chips: A New SoC Paradigm,” IEEE Computer, vol. 35, no. 1, Jan.2002, pp. 70-78.

Beigné, Edith, et al. “An asynchronous NOC architecture providing low latency service and its multi-level design framework,” In: Asynchronous Circuits and Systems, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on. IEEE, 2005. p. 54-63.

R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger and Y. Hoskote, “Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 1, pp. 3-21, 2009.

K. Bhardwaj and R.K. Jena, “Energy and Bandwidth Aware Mapping of IPs onto Regular NoC Architectures Using Multi-Objective Genetic Algorithms,” International Symposium on System-on-Chip, pp. 027-031, 2009.

C. A. M. Marcon and E. I. Moreno, “Comparison of network-on-chip mapping algorithms targeting low energy consumption,” IET Computers Digital Techniques, vol. 2, no. 6, pp. 471-482, 2008.

C. L. Chou and R. Marculescu, “User-aware Dynamic Task Allocation in Networks-on-Chip,” Design, Automation and Test in Europe, pp.1232-1237, 2008.

E. Carvalho, N. Calazans, and F. Moraes, “Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs,” in 18th IEEE/IFIP International Workshop on Rapid System Prototyping, 2007. RSP 2007, 2007, pp. 34–40.

Mandelli, M. et al. "Energy-Aware Dynamic Task Mapping for No Cbased MPSoCs," IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1676-1679, May 2011.

C. L. Chou and R. Marculescu, “Incremental Run-time Application Mapping for Homogeneous NoCs with Multiple Voltage Levels,” IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 161-166, 2007.

R. P. Dick, D. L. Rhodes and W. Wolf, “TGFF: task graphs for free,” Proc. Intl.

NoCTweak a Parameterizable Simulator for Early Exploration of Networks On-Chip, 2013, truy cập lần cuối ngày 02 tháng 09 năm 2016, <http://ncu.dl.sourceforge.net/project/noctweak/v0.9.3/noctweak-v0.9.3.tar.gz.>.

(BBT nhận bài: …/…/2016, phản biện xong: …/…/2016)

**Kính gửi: Ban Biên tập Tạp chí;**

**Các phản biện.**

Nhóm tác giả đã nhận được những góp ý, đề nghị hiệu chỉnh, bổ sung và một số vấn đề cần làm rõ thêm cho bài báo “**Triển khai các ứng dụng lên nền tảng NoC dựa trên chiến lược chọn vùng**”. Nhóm tác giả xin bày tỏ lòng biết ơn đến sự quan tâm giúp đỡ của Ban Biên tập tạp chí; các ý kiến góp ý quý báu của phản biện để nhóm tác giả có thể hiểu rõ hơn nhiều vấn đề trong nghiên cứu của mình.

Nhóm tác giả xin phép được hiệu chỉnh, bổ sung một số nội dung cũng như giải trình một số câu hỏi của phản biện nêu ra, cụ thể dưới đây.

Một lần nữa nhóm tác giả xin chân thành cảm ơn sự quan tâm của Ban Biên tập và các phản biện.

Kính chúc Ban Biên tập và các phản biện sức khỏe và thành đạt trong cuộc sống.

Trân trọng./.

|  |  |  |
| --- | --- | --- |
| **TT** | **Yêu cầu của phản biện** | **Giải trình của tác giả** |
| 1 | Làm rõ hơn khái niệm vùng gần lồi? Làm rõ tính mới của đề xuất trong bài báo với các nghiên cứu đã được công bố trước đây trong [8], [11] | - Vùng gần lồi ở đây được hiểu là vùng được chọn có dạng hình vuông hoặc gần vuông (lồi). Ưu điểm của vùng gần lồi là tổng khoảng cách Manhattan trung bình của nó nhỏ hơn so với tổng khoảng cách Manhattan trung bình trong hình chữ nhật và một vài hình dạng khác. Do vậy, độ trễ sau khi ánh xạ các ứng dụng vào vùng này sẽ thấp hơn so với ánh xạ vào các vùng có hình dạng khác. Ví dụ, một vài dạng vùng gần lồi được minh họa dưới đây.    - Tính mới của đề xuất chiến lược chọn vùng gần lồi của chúng tôi trong bài báo so với các nghiên cứu [8], [11] đã được bổ sung trong mục 3.1 từ dòng số 27-48. |
| 2 | Trình bày thêm định nghĩa về khoảng cách Manhattan? Làm rõ hơn khoảng cách này được ứng dụng trong nghiên cứu của bài báo như thế nào? | - Khoảng cách Manhattan được định nghĩa là số đường liên kết nhỏ nhất mà một gói tin đi từ PE nguồn (pxy) đến PE đích (px’y’) và nó được tính bởi công thức kc((x,y),(x’,y’)) = |x’-x| + |y’-y|. Ví dụ: Nếu gửi một gói tin đi từ nguồn V1 đến đích V3 như trong Hình 1 thì khoảng cách Manhattan là 2. (đã bổ sung vào bài báo tại mục 2.2 từ dòng 21-28)  - Trong nghiên cứu này, khoảng cách Manhattan dùng để tính số liên kết giữa các cặp tác vụ sau khi ánh xạ và tổng số các liên kết trung bình cho một vùng đã được chọn. |
| 3 | Lưu đồ thuật toán Hình 2: nên điều chỉnh lại toàn bộ tiếng Việt | Tác giả đã chỉnh sửa lại bằng tiếng Việt. |
| 4 | Bảng 1. Bổ sung đơn vị của thông số: Tốc độ bơm gói tin | Tác giả đã bổ sung đơn vị của tốc độ bơm gói tin vào Bảng 1. |
| 5 | Giải thích các thuật ngữ: NN, FF và giải thích tóm tắt hoạt động các thuật toán này? | Tác giả đã bổ sung nguyên gốc tiếng anh và giải thích tóm tắt hoạt động của 2 thuật toán FF và NN vào bài báo từ dòng 12-23 trong mục 3. |
| 6 | Giải thích các thuật ngữ VOPD, MWD, PIP trong mục 4.1. | VOPD, MWD và PIP là các ứng dụng đa phương tiện thực. Trong đó, VOPD (Video object plane decoder) là một bộ giả mã VOP, MWD (Multi-Window Display) hiển thị đa cửa sổ, PIP (Picture in Picture) hiển thị ảnh trong ảnh. Tác giả xin phép không Việt hóa các thuật ngữ này trong bài báo. |
| 7 | Lỗi ngữ pháp trong đoạn văn “Để đánh giá sự ảnh hưởng của các giải pháp chọn vùng và không chọn vùng đến hiệu năng của hệ thống. Các thông số như độ trễ và thông lượng được thu thập và phân tích.”, đề nghị sửa chữa. | Tác giả đã chỉnh sửa lỗi ngữ pháp cho đoạn văn này. |
| 8 | Kết quả trình bày trong bài báo cho thấy chiến lược chọn vùng gần lồi cho các hệ thống với độ trễ và thông lượng tốt hơn so với các hệ thống không sử dụng chiến lược chọn vùng; tác giả có thể nói thêm về các khó khăn hay nhược điểm khi sử dụng chiến lược chọn vùng? | Các ưu điểm của chiến lược chọn vùng gần lồi khi áp dụng nó vào việc triển khai các ứng dụng lên nền tản mạng trên chip tại thời gian chạy như giảm trễ, tăng thông lượng. Tuy nhiên, nó cũng có một vài hạn chế như cần phát triển thêm thuật toán chọn vùng dẫn đến thời gian thực hiện tổng thể của quá trình ánh xạ sẽ tăng lên. |
| 9 | Nếu có thể được, bổ sung các tài liệu tham khảo mới hơn gần đây 2011-2016 (không bắt buộc). | Nhóm tác giả xin phép không bổ sung thêm tài liệu tham khảo. |
| Ngoài ra, nhóm tác giả cũng đã chỉnh sửa lại Hình 1 và một số lỗi nhỏ khác được to màu đỏ trong bài báo.  Xin trân trọng cảm ơn./. | | |