Báo Cáo Kết Thúc Môn Học

Các vấn đề hiện của công nghệ điện, điện tử và viễn thông

Học viên: Đinh Văn Nam
GV1: Trần Đức Tân
GV2: Đinh Triều Dương

Mã học viên (ID): 17028023

Contents

Tóm Tắt

Ngày nay, với sự phát triển nhanh chóng của các hệ thống vi mạch tích hợp cỡ vô cùng lớn khiến cho độ phức tạp trong các mô hình truyền thông gặp thách thức không hề nhỏ. Do bởi sự tích hợp ngày càng nhiều các vi mạch trên cùng một đế chip. Do đó, việc nghiên cứu và phát triển các mô hình truyền thông trên chip đang là một xu hướng được các nhà khoa học trong lĩnh vực thiết kế vi mạch số tích hợp quan tâm.

Với các bài nghiên cứu cơ bản về các thách thức cho mô hình mạng trên chip [1] cũng như việc truyền thông trên chip bằng môi trường không dây với các thách thức không hề nhỏ [2]. Ngoài ra, với việc tăng độ tin cậy của hệ thống mạng trên chip cũng được quan tâm với nhiều vấn đề quan trọng cần giải quyết như tránh Crosstalk trong mạng dựa trên hệ thống số Fibonaci qua TSV [3]. Hay việc kết hợp mã bảo mật với độ tin cậy của mạng cũng được Wehbe và các đồng nghiệp trình bày một cách chi tiết [4]. Thậm chí những năm gần đây, thiết kế mạng trên chip có khả năng chống lỗi với mô hình mạng 3D cũng được nhiều người quan tâm như bài nghiên cứu của Das và các đồng nghiệp [5]

Cùng với xu hướng phát triển công nghệ và hướng nghiên cứu cập nhật như trên, nghiên cứu sinh đã mạnh dạn đề xuất nghiên cứu về đề tài *"Nghiên cứu về kiến trúc mạng trên chip có khả năng chịu lỗi"*, qua việc đề xuất kiến trúc, thuật toán có khả năng chịu lỗi áp dụng cho mô hình mạng trên chip, nghiên cứu sinh tiếp cận hướng phát triển đề tài bằng việc tìm hiểu và nghiên cứu về một hệ thống mạng trên chip mã nguồn mở và có khả năng cấu hình hệ thống tùy biến để áp dụng các giải thuật định tuyến một cách linh hoạt và có tính tin cậy cao. Kết hợp kinh nghiệm bản thân đã từng làm về chủ đề ánh xạ ứng dụng cho mạng trên chip của tác giả trong bài báo [6], hiện nay, kết hợp môn học ***"Các vấn đề hiện đại của công nghệ điện, điện tử và viễn thông"***, nghiên cứu sinh quyết định thực hiện đề tài này nhằm bước đầu khảo sát về cách thức thực hiện các mô phỏng, kiểm chứng kết quả nghiên cứu, phục vụ cho việc nghiên cứu phát triển đề tài về sau.

Cấu trúc của bài báo cáo gồm các phần chính như mục tiêu của đề tài như liệt kê trong phần 1. Các cơ sở thực thi của đề tài được đề xuất trong phần 2 với việc đề cập về các công cụ mã nguồn mở hiện đại như gem5 trong phần 2.1 và mô hình mạng Garnet2.0 trong phần 2.2; Các kết quả đạt được của đề tài và các kết luận lần lượt được thực hiện trong các phần 3.1 và 3.2 thông qua kịch bản mô phỏng được mô tả trong đầu phần 3.

Các từ khóa: *NoC, Fault-tolerant NoC, Crosstalk NoC 3D, Fault-tolerant architectures, Gem5, Garnet2.0*

1 Mục tiêu và phạm vi đề tài

Nội dung chính của đề tài hướng đến các nội dung chính sau:

1. Cách cài đặt Gem5 để xây dựng một kiến trúc máy tính lõi chip như ARM, X86, etc...

2. Khảo sát mạng Garnet2.0 sử dụng nền tảng Gem5 để có một cách tiếp cận hướng nghiên cứu về phát hiện và chống lỗi cho hệ thống mạng trên chip.

3. Hiểu cơ bản về mạng trên chip Garnet2.0 để áp dụng cho việc nghiên cứu phục vụ cho khóa luận tốt nghiệp.

4. Mục tiêu cuối cùng của đề tài là khảo sát điểm bão hòa của mạng trên chip (tiêu chí khảo sát dựa trên đầu ra là độ trễ truyền thông gói tin trung bình của mạng).

2 cơ sở thực hiện mô phỏng

2.1 Nền tảng Gem5

Gem5 có thể coi như là một bộ công cụ mã nguồn mở, là nền tảng giúp cho các nhà nghiên cứu làm trong lĩnh vực thiết kế kiến trúc các hệ thống máy tính. Gem5 bao gồm nhiều đặc tính như là các mô hình đơn vị xử lý trung tâm có khả năng đa tác vụ (Multiple interchangeable CPU models), hay đơn vị xử lý đồ họa, điều khiển điều hướng sự kiện bộ nhớ... và đặc biệt nó có khả năng mô hình hóa và mô phỏng một hệ thống đầy đủ (full system capability). Ngoài ra, Gem5 còn hỗ trợ việc mô phỏng kết hợp với SystemC.

Các đặc tính có thể được tìm hiểu một cách đầy đủ và chi tiết hơn trong đường liên kết các đặc điểm chính của Gem5

Ngoài ra, Gem5 còn được biết đến như là một hệ thống máy tính điều khiển sự kiện hướng mô-đun. Điều này có nghĩa là:

1. Các thành phần của Gem5 có thể được sắp xếp lại, thay đổi các thông số, cũng như việc cấu hình lại một cách phù hợp theo mục đích của người dùng một cách dễ dàng.

2. Gem5 thực hiện mô phỏng chuỗi các sự kiện rời rạc một cách tuần tự theo thời gian.

3. Gem5 có thể sử dụng để mô phỏng một hay nhiều hệ thống máy tính theo nhiều kịch bản (cách) khác nhau.

4. Gem5 không chỉ đơn thuần là một bộ công cụ mô phỏng, nó như là một nền tảng mô phỏng cho phép chúng ta có thể dùng nhiều các thành phần kiến trúc tự thiết kế để có thể xây dựng và mô phỏng một thiết kế riêng biệt.

Gem5 là nền tảng mã nguồn mở được viết chủ yếu dựa trên ngôn ngữ C++ và python. Hầu hết các thành phần đều được cấp phép bởi BSD licenses. Nó có thể mô phỏng một hệ thống hoàn chỉnh với đầy đủ các thiết bị, ngoại vi, hệ điều hành trong chế độ FS (full system mode). Hoặc nó cũng có thể được sử dụng chế độ mô phỏng hệ thống (**SE** = ***S****yscal* ***E****mulation mode*).

Các bước thực hiện cài đặt Gem5 trên hệ điều hành Ubuntu:

1. Cài đặt git với câu lệnh: sudo apt-get install git
2. Cài đặt gcc 4.8+: sudo apt-get install build-essential
3. Gem5 dùng Scons để xây dựng môi trường, để cài đặt Scons ta có thể tham khảo lệnh sau: sudo apt-get install scons
4. Cài đặt Python 2.7+ với lệnh tham khảo: sudo apt-get install python-dev.
5. Cài đặt SWIG 2.0.4+ với lệnh tham khảo: sudo apt-get install swig
6. Cài đặt pr 2.1+ với lệnh tham khảo: sudo apt-get install libprotobuf-dev python-protobuf protobuf-compiler libgoogle-perftools-dev
7. Cập nhật code mới nhất của gem5 với lệnh:
   git clone https://gem5.googlesource.com/public/gem5
   (chú ý: có thể tải bằng cách clone trực tiếp từ github.com với link tham khảo với từ khóa gem5+github từ google.com và truy cập gem5 link)

Việc cài đặt và sử dụng Gem5 có thể được tham khảo chi tiết tại hướng dẫn theo đường link tham khảo hỗ trợ cài đặt Gem5

2.2 Garnet2.0

Garnet2.0 là một mô hình mạng trên chip được thiết kế và nhúng vào trong Gem5. Hiện tại, Garnet vẫn đang được tiếp tục phát triển thêm nhiều các đặt trưng để có thể tích hợp vào trong gem5. Phiên bản Garnet lần đầu tiên được công bố và nhúng vào trong gem5 là từ năm 2009.

Garnet2.0 hỗ trợ thực thi vi kiến trúc (micro-architecture) của một bộ định tuyến trong mạng trên chip với chu kỳ chính xác. Nó như là đòn bẩy phát triển Topology và Routing trong mạng được hỗ trợ bởi mô hình hệ thống bộ nhớ Ruby trong Gem5. Bộ định tuyến mặc định một kỹ thuật đường ống một chu kỳ. Kỹ thuật này giúp cho việc thêm vào bất kỳ bộ định tuyến nào với độ trễ tùy ý bằng cách chỉ ra với topology nào.

Hệ thống mã nguồn mở với Garnet2.0 có thể hỗ trợ việc mô hình hóa một mạng trên chip bằng cách tham khảo các tệp liên quan trong cây thư mục có liên quan như liệt kê dưới đây:

- src/mem/ruby/network/Network.py
- src/mem/ruby/network/garnet2.0/GarnetNetwork.py
- src/mem/ruby/network/Topology.cc

Một số cấu hình cơ bản trong Garnet2.0:

number_of_virtual_networks: Đây là thông số xác định số lượng mạng ảo tối đa. Số lượng mạng thực được xác định bởi protocol mạng. **control_msg_size**: Kích thước của gói tin điều khiển tính bằng byte. Giá trị mặc định của thông số này là 8. thông số *m_data_msg_size* trong tập tin

Network.cc là thiết lập để kích thước khối tính bằng byte cộng với giá trị kích thước điều khiển gói tin *control_ msg_ size*.

Một số thông số đặc tả trong mạng garnet2.0 trong tệp tin *GarnetNetwork.py*:

- **ni_flit_size**: flit tính bằng đơn vị bytes. Các flit là đại lượng thông tin được truyền từ một bộ định tuyến này qua một bộ định tuyến khác. Giá trị mặc định là 16 ⇒ 128 bits. Garnet yêu cầu *ni_flit_ size* giống với thông số động rộng băng thông trong mạng (bandwidth_factor) khi mà nó không được mô hình hóa các biến băng thông bên trong mạng. Thông số này cũng có thể được truyền từ lệnh với từ khóa - - *link-width-bits*.

- **vcs_per_vnet**: Số lượng kênh ảo trên mạng ảo. Giá trị mặc định của tham số này là 4. Giá trị này cũng có thể được thiết lập thông qua dòng lệnh: - - *vcs-per-vnet*.

- **buffers_per_data_vc**: số lượng bộ đệm flit trên một kênh ảo trong một phân lớp dữ liệu được truyền đi. Bởi vì các gói tin thì được chia thành 5 flits, và một kênh ảo chỉ có thể giữ một gói tin tại một thời điểm, giá trị này có giá trị mặc định là bằng 1.

- **routing_algorithm**: 0: là giá trị mặc định, 1: giải thuật định tuyến XY, 2: là tùy biến theo người dùng thiết lập

3 Kịch bản mô phỏng

Kịch bản mô phỏng với mục đích làm rõ mục tiêu muốn thực thi của đề tài. Với mục tiêu muốn khảo sát điểm bão hòa của mạng hướng đến thông số trễ truyền thông trung bình gói tin trong hệ thống mạng trên chip. Việc khảo sát đến thông số trễ trung bình truyền thông gói tin trong mạng có thể đánh giá được hiệu năng của hệ thống, bởi vì một trong các tham số quan trọng để đánh giá hiệu năng của hệ thống chính là độ trễ truyền thông của các gói tin được đi từ IP (Intellectual Property) nguồn đến IP đích.

Mỗi một kịch bản mô phỏng được thực hiện bằng cách thay đổi kết hợp giữa tốc độ tải tin (tải gói tin) vào trong mạng và thay đổi các benchmark khác nhau.

Tốc độ tải gói tin vào mạng (**IR** = **I**njection **R**ate) được thay đổi với các giá trị từ 0.01 cho đến giá trị 1.0. Giá trị này biểu diễn số gói tin trên một node trong một chu kỳ xung nhịp của mạng (package/node/cycle). Giá trị này có độ chính xác đến 1 phần nghìn (tức là sau dấu phẩy 3 chữ số) và được cấu hình bởi lệnh - - *precision* trong tệp tin *garnet_ synth_ traffic.py*

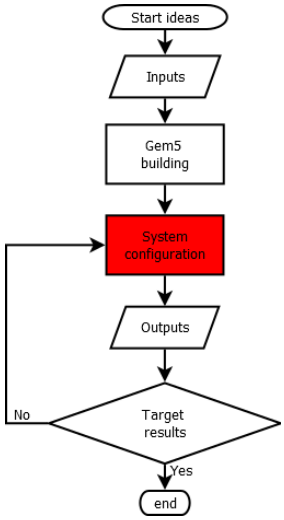
Cụ thể việc thay đổi các giá trị tốc độ tải tin được thực hiện tương ứng với các giá trị thông số IR truyền vào mạng là như bảng sau:

0.01	0.05	0.1	0.2	0.3	0.32	0.34	0.35	0.4	0.45
0.5	0.55	0.6	0.65	0.70	0.75	0.8	0.85	0.9	0.95

Các benchmark được sử dụng để cấu hình đánh giá hệ thống gồm các loại được liệt kê như bảng dưới đây:

Uniform_ramdom	Tornado	Complement	Reverse
Rotation	Neighbor	Shuffle	Transpose

Việc thực thi hệ thống và triển khai mô hình được minh họa theo sơ đồ khối thực thi dưới đây:



Hình 1: Phương pháp thực thi mô phỏng của đề tài

3.1 Kết quả mô phỏng thực nghiệm

Việc thực hiện mô phỏng thực nghiệm được dựa trên cơ sở các phân tích đã được nêu ra trong hình 1 trang 6

Sau khi thực hiện việc mô phỏng và kết quả đạt được với các điểm xác định bão hòa như hình trong bảng thống kê hình 2 trang 7

Nhận xét kết quả mô phỏng: Với kết quả thu được trong hình 2, một điểm chung ta có thể thấy rõ đối với hầu hết tất cả các benchmark đó chính là điểm bão hòa của mạng tại điểm IR xấp xỉ bằng 30% tương đương với giá trị truyền vào mạng khi cấu hình là 0.3. Tuy nhiên, vẫn còn một số điểm bất thường cần khảo sát thêm với các kết quả đầu ra bởi thông số độ trễ truyền thông gói tin trung bình trong mạng là chưa xác định rõ được ví dụ như tại các điểm IR = {0.8, 0.85} của benchmark "*shuffle*" hay IR = 0.9 của benchmark "*reverse*" và được biểu diễn bằng ký hiệu "x". Một điểm có thể phát triển hướng đề tài là việc giải thích các điểm gây ra hiện tượng kết quả bất thường như trên có thể được đi sâu xem xét đến việc cấu hình mạng cũng như đặc trưng riêng của từng benchmark đối với hệ thống mạng trên chip garnet2.0 đang xét đến trong hệ thống này.

stt	pir	uniform_random	tornado	complement	reverse	rotation	neighbor	shuffle	transpose
1	0.01	11.652536	9.581563	14.677718	11.568971	10.595888	9.581563	10.589482	11.572903
2	0.05	12.147911	9.738065	15.34043	12.167029	10.887875	9.738065	10.898337	12.081328
3	0.1	13.062851	9.97517	17.074337	13.790059	11.44403	9.97517	11.484684	13.424858
4	0.2	17.590496	10.757214	2644.358764	4208.876992	21.513635	10.757214	22.439482	3233.023231
5	0.3	1180.584745	12.788558	12135.681305	7408.833077	4224.514354	12.788558	3287.98703	6042.497596
6	0.32	2933.36754	13.642295	12770.178847	7269.379548	4559.855501	13.642295	3055.187581	6033.67835
7	0.34	4836.391358	14.881653	13466.321904	7081.115891	4913.888672	14.881653	3467.846164	5931.815727
8	0.35	5721.168206	15.796429	13786.345764	7113.159266	5146.231448	15.796429	3185.780304	5922.511436
9	0.4	9724.610312	28.444784	14206.039349	6834.518698	5159.014599	28.444784	4115.202248	6069.660787
10	0.45	12430.423395	2247.287741	14405.715216	8815.495209	6385.116608	2247.287741	11900.603274	7635.638824
11	0.5	13954.873314	6549.641906	16791.949694	14532.894733	9690.275702	6549.641906	13651.641361	11296.409535
12	0.55	14610.471286	9706.033464	19387.090275	16554.44609	11965.736669	9706.033464	14614.788157	13416.826389
13	0.6	14722.159676	11941.347926	21087.234591	18274.216166	13675.581903	11941.347926	16083.716717	15505.279098
14	0.65	14451.482382	13463.289523	22515.966221	21149.847293	14675.486383	13463.289523	16371.994648	16988.770157
15	0.7	14932.795972	14461.573254	23917.596759	21745.25139	15287.582745	14461.573254	16737.678933	17615.180765
16	0.75	15968.134858	14996.372911	25176.282532	21578.739169	16267.290438	14996.372911	16794.842641	18672.489933
17	0.8	17241.494192	15158.616001	28149.108787	21685.760708	16214.087772	15158.616001	x	18788.123869
18	0.85	18668.451182	15043.788086	29262.021533	21899.794758	17025.265589	15043.788086	x	19128.67982
19	0.9	20430.054003	14680.069899	30091.551035	x	17262.291166	14680.069899	x	19128.307677
20	0.95	21932.435255	14116.723282	31477.214826	x		14116.723282	x	19778.672904
21	1	23303.692783	13505.478339	x	x		13505.478339	x	

Hình 2: Kết quả mô phỏng khảo sát điểm làm việc bão hòa với thông số độ trễ trung bình gói tin trong mạng

3.2 Kết luận

Như vậy với việc thực thi cài đặt bộ mô phỏng và khảo sát điểm bão hòa của thông số độ trễ truyền trung bình trong mạng trên chip tại các điểm tải tin xác định đã đạt được một số kết quả như sau:

- Biết cách vận hành công cụ Gem5 để xây dựng các lõi vi xử lý tiên tiến như RISCV, X86, ARM, SPARC...
- Biết cách cấu hình và mô phỏng một hệ thống mạng trên chip (Garnet2.0) trên Gem5, là tiền đề, cơ sở để phát triển hướng nghiên cứu về sau.
- Khảo sát được điểm bão hòa của mạng trên chip với đầu ra khảo sát là độ trễ truyền thông tải tin trung bình trong mạng.

Tuy nhiên, còn tồn tại một số nhược điểm dưới đây:

- Các kịch bản khảo sát chưa đủ lớn, chưa bao quát được tất cả các trường hợp sinh lỗi hệ thống.
- Một số điểm bất thường chưa tính toán được trễ truyền thông trung bình gói tin mà chưa tìm ra được nguyên nhân gây lỗi.
- Chưa cấu hình chạy với chế độ FS để có thể minh chứng được các kết quả mô phỏng có tính thuyết phục hơn.

Một số hướng nghiên cứu và phát triển của đề tài có thể xem xét đến đó là làm sao để cấu hình hệ thống chạy mô phỏng với nhiều benchmark hơn cũng như có thể chạy với chế độ FS; Bên cạnh đó cũng xây dựng được các mô hình kiến trúc NoC có khả năng chống lỗi, các thuật toán dò lỗi, sửa lỗi có thể áp dụng vào trong hệ thống để các kết quả có đóng góp ý nghĩa hơn về mặt khoa học và ứng dụng thực tiễn sẽ thiết thực hơn.

References

[1] M. S. Gaur, V. Laxmi, M. Zwolinski, M. Kumar, N. Gupta, and Ashish, "Network-on-chip: Current issues and challenges," in *2015 19th International Symposium on VLSI Design and Test*, pp. 1–3, June 2015.

[2] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Wireless noc as interconnection backbone for multicore chips: Promises and challenges," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, pp. 228–239, June 2012.

[3] X. Cui, X. Cui, Y. Ni, M. Miao, and J. Yufeng, "An enhancement of crosstalk avoidance code based on fibonacci numeral system for through silicon vias," vol. 25, no. 5, pp. 1601–1610.

[4] T. Wehbe and X. Wang, "Secure and dependable NoC-connected systems on an FPGA chip," vol. 65, no. 4, pp. 1852–1863.

[5] S. Das, J. R. Doppa, P. P. Pande, and K. Chakrabarty, "Robust TSV-based 3d NoC design to counteract electromigration and crosstalk noise," in *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '17, pp. 1366–1371, European Design and Automation Association.

[6] V. N. Dinh, K. H. Nguyen, M. T. Pham, and X. T. Tran, "An IDPSO algorithm-based application mapping method for network-on-chips," in *The 7th International Conference on Integrated Circuits, Design, and Verification (ICDV)*, pp. 104–110, 2017.