|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  **HƯỚNG DẪN SỬ DỤNG MÃ NGUỒN**  **C++ VÀ VERILOG ĐỂ MÔ PHỎNG VÀ KIỂM TRA NEURON GRID**  Người thực hiện: Nguyễn Lê Trung 20186076  Phạm Huy Hoàng 20182544  Vũ Hoàng Long 20182926  Giảng viên hướng dẫn: PGS. TS. Nguyễn Đức Minh  Hà Nội, 02-2022 |

**LỜI NÓI ĐẦU**

Tài liệu này được sinh viên Viện Điện tử - Viễn thông, trường Đại học Bách Khoa Hà Nội soạn thảo và ban hành nhằm mục đích hướng dẫn sử dụng mã nguồn C++ và verilog để mô phỏng neuron grid.

Chúng tôi xin chân thành cảm ơn PGS. TS. Nguyễn Đức Mình đã hướng dẫn chúng tôi hoàn thành báo cáo này.

**LỜI CAM ĐOAN**

Chúng tôi là Nguyễn Lê Trung, Phạm Huy Hoàng và Vũ Hoàng Long sinh viên lớp CTTT Điện tử - Viễn thông, khóa 63. Người hướng dẫn là PGS. TS. Nguyễn Đức Minh. Chúng tôi xin cam đoan toàn bộ nội dung được trình bày trong *Hướng dẫn sử dụng mã nguồn C++ và verilog để mô phỏng và kiểm tra Neuron Grid* là kết quả quá trình tìm hiểu và nghiên cứu của chúng tôi. Các dữ liệu được nêu trong bản hướng dẫn là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Chúng tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ i](#_Toc96462422)

[CHƯƠNG 1. TỔNG QUAN CÁCH SỬ DỤNG 1](#_Toc96462423)

[CHƯƠNG 2. CÁCH SỬ DỤNG CODE C++ 2](#_Toc96462424)

[2.1 Tạo file text chứa dữ liệu 2](#_Toc96462425)

[2.1.1 Tạo file chứa axon spikes 1 tick và nhiều tick 2](#_Toc96462426)

[2.1.2 Tạo file chứa parameter trong CSRAM 3](#_Toc96462427)

[2.2 Chạy mô phỏng 4](#_Toc96462428)

[2.2.1 Chạy mô phỏng với 1 tick 4](#_Toc96462429)

[2.2.2 Chạy mô phỏng với nhiều tick 5](#_Toc96462430)

[CHƯƠNG 3. CÁCH SỬ DỤNG VERILOG 6](#_Toc96462431)

# DANH MỤC HÌNH VẼ

[*Hình 1 Sơ đồ tổng quan hoạt động* 1](#_Toc96462432)

[*Hình 2 Địa chỉ chứa file axon spikes 1 tick và nhiều tick* 2](#_Toc96462433)

[*Hình 3 Hai hàm tạo axon spikes 1 tick và nhiều tick được gọi* 2](#_Toc96462434)

[*Hình 4* *Địa chỉ chứa file parameter của CSRAM* 3](#_Toc96462435)

[*Hình 5 Hàm generateCSRAM()* 3](#_Toc96462436)

[*Hình 6 Hàm generateCSRAM() được gọi trong hàm main()* 4](#_Toc96462437)

[*Hình 7 Địa chỉ file chứa Neuron Instructions và file đầu ra* 4](#_Toc96462438)

[*Hình 8 Hàm NeuronGrid() được gọi trong hàm main()* 5](#_Toc96462439)

[*Hình 9 Hàm ManyTicks() được gọi trong hàm main()* 5](#_Toc96462440)

[*Hình 10 Số lượng tick và địa chỉ các file cần thiết trong tb\_neuron\_grid* 6](#_Toc96462441)

[*Hình 11 Transcript thông báo test không gặp lỗi* 6](#_Toc96462442)

[*Hình 12* *Transcript thông báo test gặp lỗi* 6](#_Toc96462443)

# TỔNG QUAN CÁCH SỬ DỤNG

Chức năng của file *neuron\_grid\_codeC* là tạo ra các file text chứa dữ liệu và dùng chính những file dữ liệu đó hoặc file từ bên ngoài để chạy mô phỏng. Đầu ra của code C++ là:

* Các file text chứa dữ liệu.
* File text chứa kết quả mô phỏng C++.

Chức năng của các file Verilog là sử dụng các dữ liệu từ file text để mô phỏng hoạt động của *Neuron Grid.* Đầu ra của code Verilog là file text chứa kết quả mô phỏng Verilog.

Vì vậy, ta có thể sử dụng hai file code này để kiểm tra hoạt động và kết quả đầu ra như sau:

Diagram

Description automatically generated

*Hình 1 Sơ đồ tổng quan hoạt động*

Theo như *Hình 1,* cả file Code C++và Verilog đều có thể lấy đầu vào từ file text chứa dữ liệu bên ngoài, tuy nhiên file Code C++ có thể tự tạo ra các file text chứa dữ liệu để làm đầu vào cho Code Verilog.

# CÁCH SỬ DỤNG CODE C++

## Tạo file text chứa dữ liệu

Code C++ có khả năng tạo ra 3 file text chứa dữ liệu:

* File chứa axon spikes với 1 tick
* File chứa axon spikes với nhiều tick
* File chứa parameter trong CSRAM

### Tạo file chứa axon spikes 1 tick và nhiều tick

Địa chỉ chứa dữ liệu của file chứa axon spikes 1 tick và nhiều tick có thể được thay đổi tại 2 dòng define:



*Hình 2 Địa chỉ chứa file axon spikes 1 tick và nhiều tick*

Hai hàm generateAxonSpikes(N\_a) và generateManyAxonSpikes (N\_a, num\_tick) cần được gọi ở hàm main() trước khi Run (với N\_a là số lượng axon và num\_tick là số lượng tick):

Text

Description automatically generated

*Hình 3 Hai hàm tạo axon spikes 1 tick và nhiều tick được gọi*

*Hình 3* thể hiện hàm generateManyAxonSpikes() tạo file text với 16 dòng, mỗi dòng thể hiện 1 tick và chứa dữ liệu axon spikes của 256 axon. Tương tự, hàm generateAxonSpikes() tạo file text với 1 dòng ứng với 1 tick và chứa dữ liệu axon spikes của 256 axon.

Các axon spikes này đều được chọn ngẫu nhiên giữa 0 và 1.

### Tạo file chứa parameter trong CSRAM

Địa chỉ chứa dữ liệu của file chứa parameter trong CSRAM có thể được thay đổi tại dòng define:



*Hình 4* *Địa chỉ chứa file parameter của CSRAM*

Trong hàm khởi tạo generateCSRAM(), file text được tạo ra có số dòng bằng số lượng neuron (mặc định là 256), mỗi dòng là các parameter của mỗi neuron được sinh ngẫu nhiên với độ rộng tuân theo kiến trúc RANC (368 bit). Khoảng ngẫu nhiên có thể được thay đổi trong hàm random(a, b) (với a là giá trị nhỏ nhất và b là giá trị lớn nhất trong khoảng dưới dạng thập phân).

Graphical user interface, text

Description automatically generated

*Hình 5 Hàm generateCSRAM()*

LƯU Ý: Giá trị các kết nối synapse chỉ được sinh ngẫu nhiên cho 64 Neurons đầu tiên và 11 axon đầu tiên đối với mỗi Neurons đó, các Neurons và axon còn lại đặt mặc định bằng 0. Bởi vì trong kiến trúc RANC, thông tin điện áp và spike đầu ra của mỗi Neuron chỉ được xét sau khi tất cả 256 Axons đối với Neuron đó được tính toán. Tuy nhiên độ rộng mặc định của *current\_potential* chỉ là 9 bit có dấu (tính theo số bù 2). Cho nên nếu có quá nhiều kết nối synapse hoặc weight quá lớn , khi điện áp được tích lũy nhiều lần thì nó có thể vượt qua giá trị 8 bit dương và tràn lên 9 bit (bit MSB là bit dấu) dẫn đến sai số. (Ví dụ: 25610 = 100000000­2 trong kiểu số unsigned = -25610 trong kiểu signed – số bù 2).

Trong thực tế, tất cả Neuron Parameters được sinh ra từ model đã được train nên sai số này khó có thể xảy ra.

Hàm generateCSRAM() phải được gọi ở hàm main() trước khi Run để tạo file text chứa parameter:

Text

Description automatically generated

*Hình 6 Hàm generateCSRAM() được gọi trong hàm main()*

## Chạy mô phỏng

Trước khi mô phỏng, cần đảm bảo những giá trị sau đây được tùy chỉnh phù hợp:



*Hình 7 Địa chỉ file chứa Neuron Instructions và file đầu ra*

Những giá trị này là địa chỉ của các file cần thiết cho việc chạy mô phỏng:

* File đầu ra
* File chứa Neuron Instruction: chứa tất cả những thông tin loại axons và có thể download tại: [RANC/hardware/src/simulations/memory\_files/vmm at master · UA-RCL/RANC · GitHub](https://github.com/UA-RCL/RANC/tree/master/hardware/src/simulations/memory_files/vmm) hoặc tại đây:



### Chạy mô phỏng với 1 tick

Trước khi Run code, hàm NeuronGrid() phải được gọi trong hàm main().

Text

Description automatically generated

*Hình 8 Hàm NeuronGrid() được gọi trong hàm main()*

Sau khi Run code, file text output sẽ được đặt trong thư mục có tên và địa chỉ đã được đặt trong dòng lệnh define ở *Hình 7.* File text output mặc định gồm 1 dòng 256 bit chứa thông tin spike của từng neuron. Cửa sổ cmd sẽ hiện những neuron có spike.

### Chạy mô phỏng với nhiều tick

Sử dụng hàm *ManyTick(num\_tick)* để chạy mô phỏng:

Text

Description automatically generated

*Hình 9 Hàm ManyTicks() được gọi trong hàm main()*

Sau khi Run code, file text output sẽ được đặt trong thư mục có tên và địa chỉ đã được đặt trong dòng lệnh define ở *Hình 7.* File text output mặc định gồm 16 dòng tương đương với 16 tick, mỗi dòng 256 bit chứa thông tin spike của từng neuron. Cửa sổ cmd sau khi chạy hiện số lượng spike đầu ra.

# CÁCH SỬ DỤNG VERILOG

Để chạy mô phỏng code Verilog, trong file tb\_neuron\_grid, số lượng tick cần test cần được cập nhật và địa chỉ của 6 file sau cần được tạo: file chứa axon spikes với 1 tick, file chứa axon spikes với nhiều tick, file chứa parameter CSRAM, file chứa neuron instruction, file chứa kết quả mô phỏng code C++ và file chứa kết quả mô phỏng Verilog.

Text

Description automatically generated

*Hình 10 Số lượng tick và địa chỉ các file cần thiết trong tb\_neuron\_grid*

Sau khi mô phỏng, kết quả file output bằng Verilog sẽ được so sánh với file output code C++. Nếu giống nhau, Transcript sẽ hiện lên thông báo “Test pass without error” :

Text

Description automatically generated

*Hình 11 Transcript thông báo test không gặp lỗi*

Nếu không giống nhau, Transcript sẽ hiện lên thông báo những neuon lỗi ở từng tick:

Table

Description automatically generated with medium confidence

*Hình 12* *Transcript thông báo test gặp lỗi*