|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  **ĐỒ ÁN THIẾT KẾ III**  **ĐỀ TÀI:**  **TRIỂN KHAI SPIKING NEURAL NETWORKS NHẬN DIỆN CHỮ SỐ VIẾT TAY TRÊN FPGA**   |  |  | | --- | --- | | **Sinh viên thực hiện:** | Trương Tuấn Vũ – 20164724 | | **Giảng viên hướng dẫn:** | PGS.TS Nguyễn Đức Minh |   Hà Nội, 5-2020 |

**MỤC LỤC**

[DANH MỤC HÌNH VẼ i](#_Toc58948158)

[DANH MỤC BẢNG BIỂU ii](#_Toc58948159)

[TÓM TẮT ĐỒ ÁN iii](#_Toc58948160)

[CHƯƠNG 1. GIỚI THIỆU CHUNG 4](#_Toc58948161)

[1.1 Tổng quan Neuronscience Computation 4](#_Toc58948162)

[1.2 Giới thiệu Spiking Neural Networks 4](#_Toc58948163)

[1.3 Thư viện ảnh MNIST 4](#_Toc58948164)

[1.4 Xilinx Zynq ZC-702 System on Chip 4](#_Toc58948165)

[CHƯƠNG 2. ĐẶC TẢ THÔNG SỐ KỸ THUẬT 5](#_Toc58948166)

[2.1 Mục tiêu thiết kế 5](#_Toc58948167)

[2.1.1 Hướng tối ưu cho thiết kế 5](#_Toc58948168)

[2.1.2 Yêu cầu chức năng 5](#_Toc58948169)

[2.2 Kiến trúc TrueNorth cho mạng SNN 5](#_Toc58948170)

[2.2.1 Mô hình tương quan phần cứng - tế bào thần kinh sinh học 5](#_Toc58948171)

[2.2.2 Mô hình mạng lưới tế bào chia thành nhiều Core 6](#_Toc58948172)

[2.2.3 Kiến trúc của một Core (Cell) trong mạng 8](#_Toc58948173)

[2.2.4 Cấu trúc của mỗi gói ứng với một kích thích lan truyền trong mạng 9](#_Toc58948174)

[2.3 Kiến trúc Top module – SNN\_NetworkGrid 10](#_Toc58948175)

[2.3.1 Sơ đồ khối Top module 10](#_Toc58948176)

[2.3.2 Mô tả chức năng khối Top module 11](#_Toc58948177)

[2.4 Kiến trúc và hoạt động của các khối con trong một Core 12](#_Toc58948178)

[2.4.1 Neuron Block 12](#_Toc58948179)

[2.4.2 CSRAM 13](#_Toc58948180)

[2.4.3 Router 14](#_Toc58948181)

[2.4.4 Scheduler 17](#_Toc58948182)

[2.4.5 Token Controller 19](#_Toc58948183)

[CHƯƠNG 3. ĐỀ XUẤT KIẾN TRÚC MẠNG SNN FULLY PARALLEL TỐI ƯU TỐC ĐỘ XỬ LÝ 21](#_Toc58948184)

[3.1 Tổng quan ý tưởng cho mạng SNN Fully Parallel 21](#_Toc58948185)

[3.2 Các thay đổi về kiến trúc mỗi Core so với kiến trúc thông thường 21](#_Toc58948186)

[3.2.1 Thay đổi cấu tạo của một Core 22](#_Toc58948187)

[3.2.2 Neurons Grid thay thế cho CSRAM & Neuron Block 23](#_Toc58948188)

[3.2.3 Thiết kế lại máy trạng thái Token Controller 25](#_Toc58948189)

[3.2.4 Tăng kích thước bộ đệm trong Router 27](#_Toc58948190)

[3.2.5 Đề xuất tối giản Scheduler 27](#_Toc58948191)

[3.3 So sánh và nhận xét kết quả 28](#_Toc58948192)

[3.3.1 Chức năng của mạng theo kiến trúc Fully Parallel 28](#_Toc58948193)

[3.3.2 Phương pháp so sánh các thiết kế 28](#_Toc58948194)

[3.3.3 Các kết quả thu được sau khi tổng hợp và khảo sát 29](#_Toc58948195)

[3.3.4 Nhận xét kết quả 29](#_Toc58948196)

[CHƯƠNG 4. MÔ PHỎNG VÀ KIỂM THỬ KIẾN TRÚC SNN THỰC HIỆN NHẬN DIỆN CHỮ SỐ VIẾT TAY 31](#_Toc58948197)

[4.1 Phương thức huấn luyện mạng SNN 31](#_Toc58948198)

[4.2 Mô phỏng và kiểm thử sử dụng phần mềm Modelsim 31](#_Toc58948199)

[4.2.1 Phương thức sinh kích thích đầu vào cho mạng 31](#_Toc58948200)

[4.2.2 Thực hiện mạng SNN gồm 5 Core đọc ảnh MNIST 31](#_Toc58948201)

[4.2.3 Phương thức giải mã đầu ra từ các kích thích thu được 31](#_Toc58948202)

[4.3 Nhận xét kết quả mô phỏng kiểm thử 31](#_Toc58948203)

[CHƯƠNG 5. TRIỂN KHAI KIẾN TRÚC SNN TRÊN BOARD XILINX ZYNQ ZC-702 32](#_Toc58948204)

[5.1 Các bước tiến hành 32](#_Toc58948205)

[5.1.1 Yêu cầu hệ thống 32](#_Toc58948206)

[5.1.2 Quy trình tổng quan triển khai hệ thống 32](#_Toc58948207)

[5.1.3 Thiết kế sơ đồ khối hệ thống 32](#_Toc58948208)

[5.1.4 Tổng hợp trên FPGA 32](#_Toc58948209)

[5.1.5 Kết nối các khối và tiến hành kiểm thử 32](#_Toc58948210)

[5.2 Nhận xét kết quả triển khai FPGA 32](#_Toc58948211)

[KẾT LUẬN 33](#_Toc58948212)

[Kết luận chung 33](#_Toc58948213)

[Hướng phát triển 33](#_Toc58948214)

# DANH MỤC HÌNH VẼ

[Hình 2.1 Mô hình tương quan các thành phần trong mạng thần kinh 6](#_Toc58948215)

[Hình 2.2 Kiến trúc TrueNorth cho mạng SNN 7](#_Toc58948216)

[Hình 2.3 Mạng SNN hình thành từ các Core 8](#_Toc58948217)

[Hình 2.4 Kiến trúc của một Core trong mạng 9](#_Toc58948218)

[Hình 2.5 Cấu trúc một gói kích thích 9](#_Toc58948219)

[Hình 2.6 Sơ đồ khối SNN\_NetworkGrid 10](#_Toc58948220)

[Hình 2.7 Biểu đồ thời gian khối Top module 11](#_Toc58948221)

[Hình 2.8 Kiến trúc giản lược Neuron Block 13](#_Toc58948222)

[Hình 2.9 Kiến trúc Router 15](#_Toc58948223)

[Hình 2.10 Gói kích thích được lan truyền thông qua các Router 16](#_Toc58948224)

[Hình 2.11 Kiến trúc Scheduler 17](#_Toc58948225)

[Hình 2.12 Mô tả hoạt động của Scheduler 17](#_Toc58948226)

[Hình 2.13 Biểu đồ thời gian của Scheduler 18](#_Toc58948227)

[Hình 2.14 Máy trạng thái Token Controller 19](#_Toc58948228)

[Hình 2.15 Mô tả hoạt động Token Controller 20](#_Toc58948229)

[Hình 2.16 Biểu đồ thời gian Token Controller 20](#_Toc58948230)

[Hình 3.1 Hoạt động song song của 256 Neurons trong một Core 22](#_Toc58948231)

[Hình 3.2 Cấu tạo một Core trong kiến trúc Fully Parallel 22](#_Toc58948232)

[Hình 3.3 Kiến trúc tổng quan Neurons Grid 23](#_Toc58948233)

[Hình 3.4 Kiến trúc chi tiết khối NG\_Spike 24](#_Toc58948234)

[Hình 3.5 Biểu đồ thời gian khối NG\_Spike 24](#_Toc58948235)

[Hình 3.6 Biểu đồ thời gian Token Controller 25](#_Toc58948236)

[Hình 3.7 Sơ đồ dịch chuyển trạng thái của Token Controller 26](#_Toc58948237)

# DANH MỤC BẢNG BIỂU

[Bảng 2.1 Bảng mô tả tín hiệu khối SNN\_NetworkGrid 11](#_Toc58948238)

[Bảng 2.2 Bảng mô tả các trường trong CSRAM 14](#_Toc58948239)

[Bảng 3.1 So sánh thông số các mạng SNN 29](#_Toc58948240)

# TÓM TẮT ĐỒ ÁN

Nêu thực trạng công nghệ trên thế giới đang hướng tới phát triển những sản phẩm thông minh, ứng dụng AI,… Tuy nhiên các thuật toán AI tốn quá nhiều tài nguyên và năng lượng, khó ứng dụng thực tế,… à Đề xuất triển khai mạng SNN tối ưu hóa về các mặt này.

Giới thiệu qua về mạng SNN và quy trình triển khai trên FPGA.

Gửi lời cảm ơn giảng viên hướng dẫn.

# GIỚI THIỆU CHUNG

## Tổng quan Neuronscience Computation

## Giới thiệu Spiking Neural Networks

## Thư viện ảnh MNIST

## Xilinx Zynq ZC-702 System on Chip

# ĐẶC TẢ THÔNG SỐ KỸ THUẬT

## Mục tiêu thiết kế

### Hướng tối ưu cho thiết kế

Nhằm khắc phục những hạn chế của các thuật toán AI khi phải thực hiện hàng loạt các phép tính toán phức tạp, tốn nhiều tài nguyên, năng lượng,… Kiến trúc SNN triển khai trực tiếp trên phần cứng số, mô phỏng cấu trúc và hoạt động của các tế bào thần kinh Neurons trong não người, hướng tới một IC có khả năng xử lý các tác vụ tương ứng với các thuật toán DeepLearning, tuy nhiên lại có sự đột phá về năng lượng tiêu thụ, tốc độ xử lý,…

### Yêu cầu chức năng

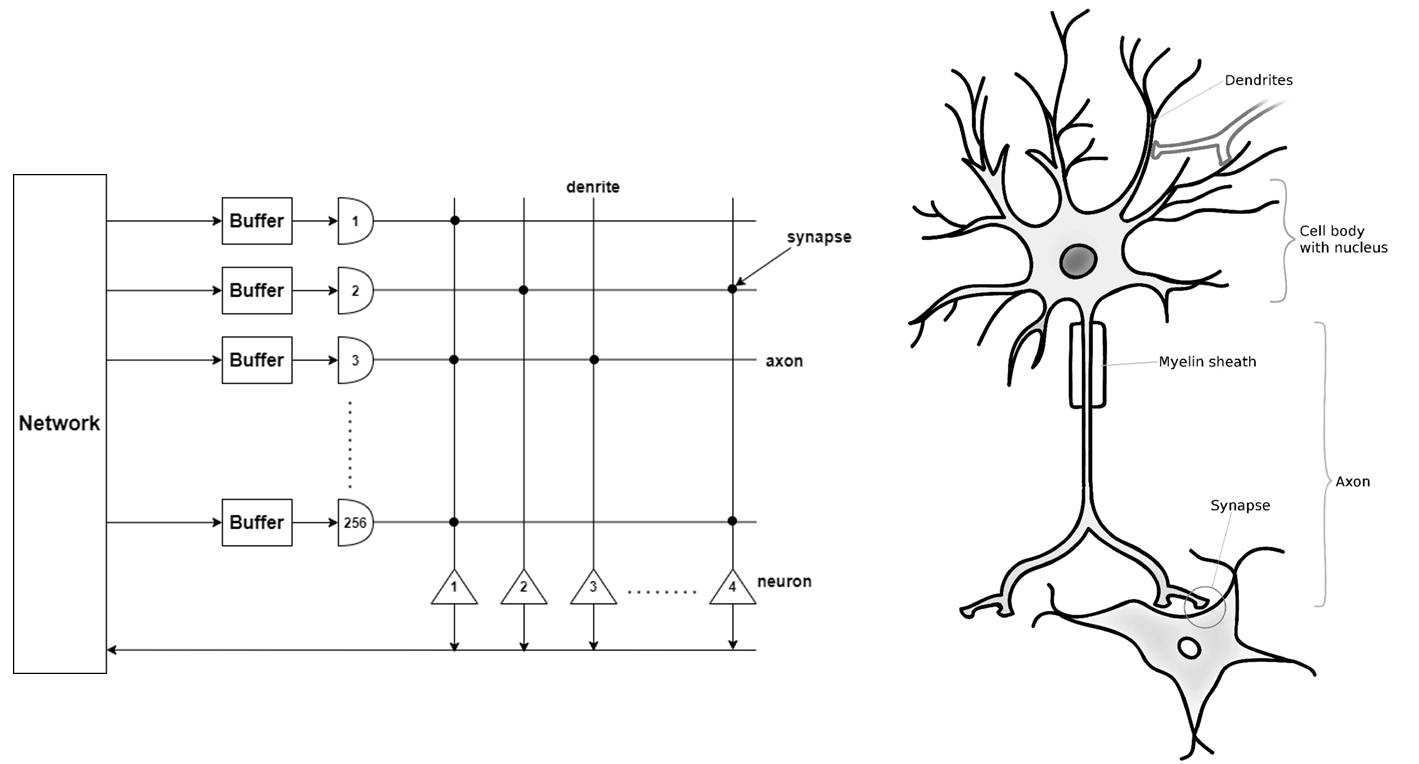
* Tần số hoạt động: 100 MHz.
* Có thể thực hiện chức năng đọc và nhận diện chữ số viết tay.
* Có khả năng mở rộng cho nhiều chức năng khác tùy vào bộ tham số được huấn luyện từ trước.
* Có thể tổng hợp mạch trên FGPA và công nghệ ASIC.

## Kiến trúc TrueNorth cho mạng SNN

### Mô hình tương quan phần cứng - tế bào thần kinh sinh học

Mạng SNN cố gắng mô phỏng lại chính xác nhất có thể về cấu tạo và hoạt động của não người, bao gồm gần như đầy đủ các thành phần cơ bản của bộ não, biểu diễn dưới dạng các khối phần cứng số.

Khả năng lưu trữ dữ liệu, hay trí nhớ của bộ não, được hình thành từ việc kết nối các tế bào thần kinh Neurons, dây thần kinh Axons và đặt các trọng số (weights) dựa trên bộ tham số huấn luyện, thay vì có các khối bộ nhớ như RAM, ROM thông thường đối với kiến trúc Von Neumann.



Hình 2.1 Mô hình tương quan các thành phần trong mạng thần kinh

Các thành phần cơ bản của một mô hình mạng lưới thần kinh sinh học:

* Neuron: các tế bào thần kinh tích lũy giá trị màng (Membrain Potential), khi giá trị này vượt một ngưỡng xác định thì Neuron sẽ phóng kích thích.
* Dendrite: các đường rẽ nhánh từ một Neuron.
* Axon: các đường dây thần kinh kết nối các Neurons với nhau.
* Synapse: điểm kết nối giữa một Axon và một Denrite.

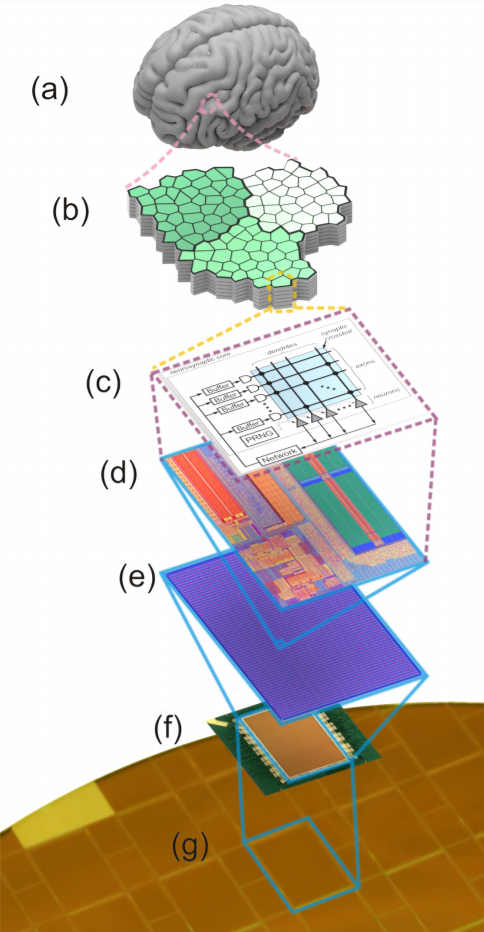
Nhận xét chung:

* Một Neuron có thể nhận được các kích thích từ nhiều Axons thông qua các Denrites của nó.
* Một Axons có thể kết nối tới nhiều Neuron để lan truyền kích thích.

(Kết nối Synapse giữa Axon và Denrite được thể hiện bằng chấm đậm)

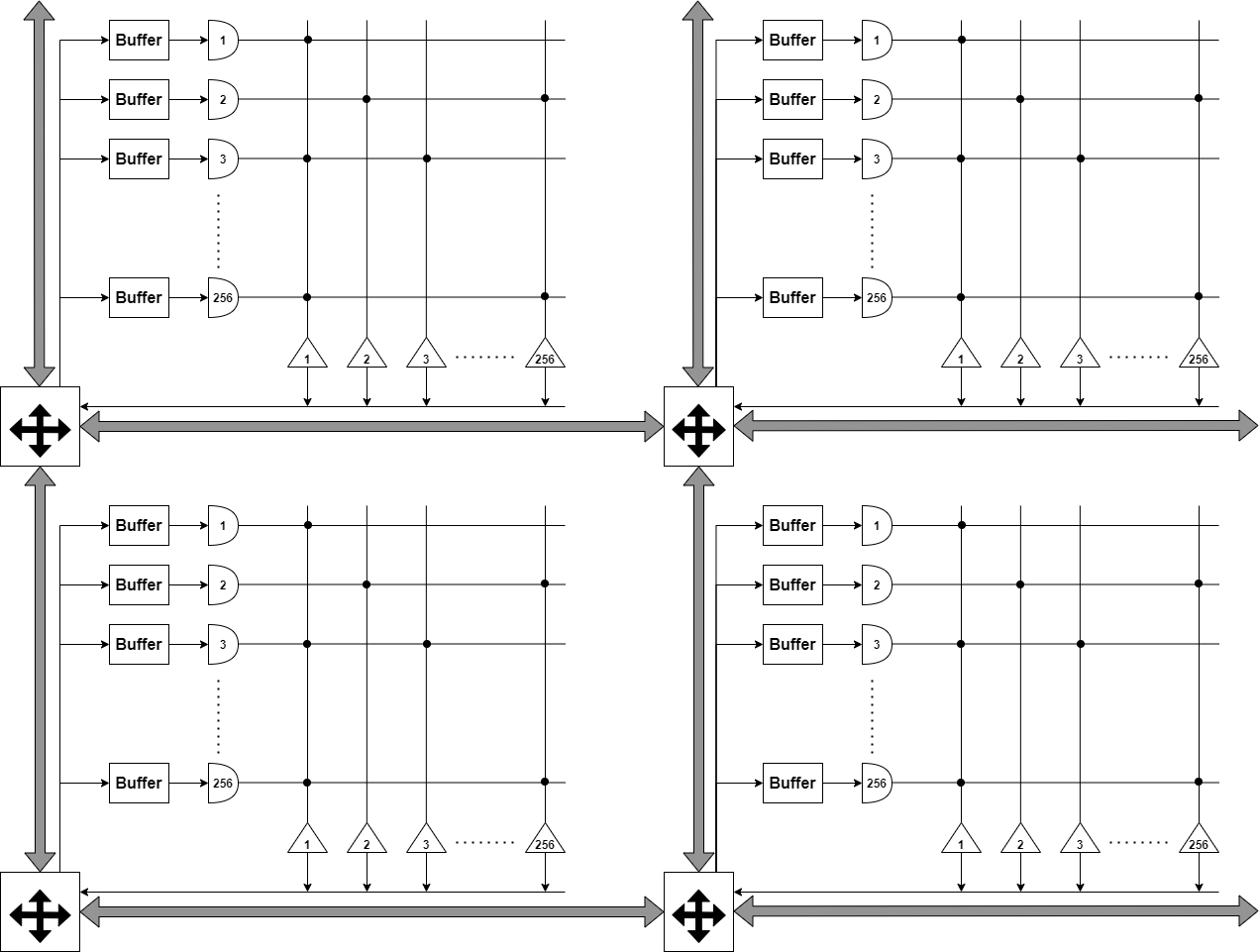
### Mô hình mạng lưới tế bào chia thành nhiều Core

Mỗi Core mô phỏng lại một phần sinh học của bộ não (cell), bao gồm 256 Neurons, nhận kích thích (spikes) từ mạng, tích lũy giá trị Potential dựa trên các kích thích này và có thể phóng ra các kích thích tương ứng nếu giá trị Potential vượt qua một ngưỡng nhất định (threshold). Sau khi phóng ra một kích thích, giá trị Potential của Neuron sẽ được thiết lập lại về giá trị khởi tạo.



Hình 2.2 Kiến trúc TrueNorth cho mạng SNN

Khi hết hợp nhiều Core thành một mạng lưới lớn, mạng SNN được hình thành và thực hiện được các chức năng ứng với các bộ tham số kết nối đã được huấn luyện từ trước. Khi số lượng các Core tăng lên, mạng càng thực hiện được các chức năng phức tạp hơn, với độ chính xác cao hơn, mang tính ứng dụng thực tiễn.



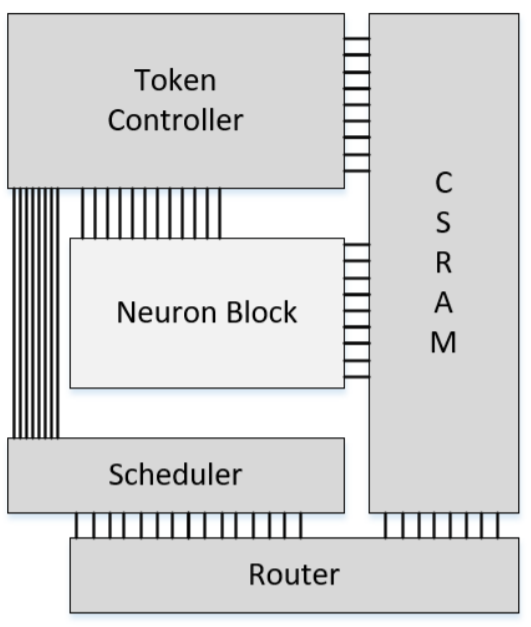
Hình 2.3 Mạng SNN hình thành từ các Core

Kích thích phóng ra từ Neurons trong mỗi Core sẽ được đưa tới các Neurons khác trong mạng bằng khối Router trong Core đó, thông qua các dây thần kinh Axons. Mỗi Neuron đều chỉ phóng kích thích tới một đích đến duy nhất, xác định bởi các trường trong một gói tin được đề cập tại mục 2.2.4.

### Kiến trúc của một Core (Cell) trong mạng

Để thực hiện được cấu trúc các Core như mô tả trong phần 2.2.1, kiến trúc TrueNorth được đề xuất với mỗi Core gồm 5 khối chính bao gồm Neuron Block, CSRAM, Scheduler, Token Controller và Router.

Mạng SNN khi này có thể mở rộng không giới hạn dựa trên các Core khi kết nối với nhau, số lượng Core càng lớn thì khả năng ứng dụng của mạng càng cao, thực hiện được nhiều chức năng phức tạp hơn, tuy nhiên diện tích mạch và năng lượng tiêu thụ sẽ lớn hơn.



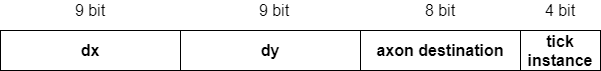
Hình 2.4 Kiến trúc của một Core trong mạng

Khi Router nhận được gói tin có đích đến là Core hiện tại, nó sẽ đưa các gói kích thích tới Scheduler để lưu trữ. Tại tick tiếp theo, các gói này quyết định việc những Axons nào trong 256 Axons của Core sẽ phóng ra kích thích. Sau đó Token Controller là một máy trạng thái sẽ điều khiển hoạt động của CSRAM và Neuron Block một cách phù hợp để tính toán đầu ra cho 256 Neurons, đưa các kích thích tới mạng qua Router.

Lưu ý: ngoài chu kỳ đồng hồ clk, kiến trúc mạng sử dụng thêm một đơn vị tính toán thời gian là “tick”. Mỗi tick sẽ được kích hoạt sau một số chu kỳ clk nhất định, bắt đầu cho toàn bộ hoạt động tính toán của một Core. Nói cách khác, khoảng thời gian giữa các tick là thời gian mà mạng cần để xử lý hoàn tất cho một ảnh. Việc lựa chọn giá trị tick hợp lý cũng quyết định rất nhiều tới tốc độ xử lý của mạng. Nếu thời gian tick quá nhỏ, ảnh chưa kịp xử lý xong đã xuất hiện tick mới, gây ra lỗi cho mạng. Ngược lại nếu thời gian tick quá lớn, mạng xử lý xong ảnh sẽ phải chờ tick mới xuất hiện để tiếp tục hoạt động, gây lãng phí hiệu năng của mạng.

### Cấu trúc của mỗi gói ứng với một kích thích lan truyền trong mạng

Mỗi gói kích thích lan truyền trong mạng được biểu diễn bởi một bus dữ liệu cócd mnt d kích thước 30 bit, gồm các trường dx, dy, axon destination và tick instance.



Hình 2.5 Cấu trúc một gói kích thích

Trong đó, ý nghĩa các trường được thể hiện:

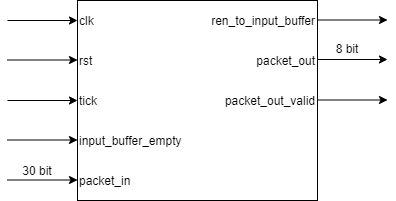
* dx, dy: hai trường ứng với chiều ngang, dọc, xác định vị trí của Core đích mà gói tin hướng đến.
* Axon destination: sau khi đã xác định được Core đích, trường axon destination chỉ ra Axon nào trong 256 Axons của Core đó sẽ nhận được kích thích để đưa tới các Neurons trong Core.
* Tick instance: sau khi gói được đưa từ Router tới Scheduler để lưu trữ, trường tick instance xác định thời điểm mà kích thích được đưa ra từ Scheduler tới các Axons trong Core. Giá trị tick instance càng lớn thì kích thích càng lâu được đưa ra hơn, và khoảng thời gian này là số nguyên lần của thời gian “tick”.

Lưu ý: hai khái niệm “tick” và “tick instance” là hoàn toàn khác nhau. Trong đó “tick” là một tín hiệu được kích hoạt định kỳ sau mỗi khoảng thời gian nhất định, được dùng để xác định thời gian xử lý xong một ảnh. Còn “tick instance” mang ý nghĩa là một trường trong gói kích thích lan truyền trong mạng.

## Kiến trúc Top module – SNN\_NetworkGrid

### Sơ đồ khối Top module

Top module là khối SNN\_NetworkGrid, đã bao gồm một mạng lưới gồm nhiều Core bên trong kết nối với nhau. Trong đó Core đầu vào nhận các gói, sử dụng Router để đưa các gói này tới những Core đích đến thích hợp dựa vào thông tin các trường trong gói. Sau đó các gói kích thích được lan truyền trong mạng và tạo ra thêm các kích thích mới sinh ra bởi các Neurons, kết thúc quá trình này, các gói được gửi tới Core đầu ra và trở thành Output chung cho toàn mạng.



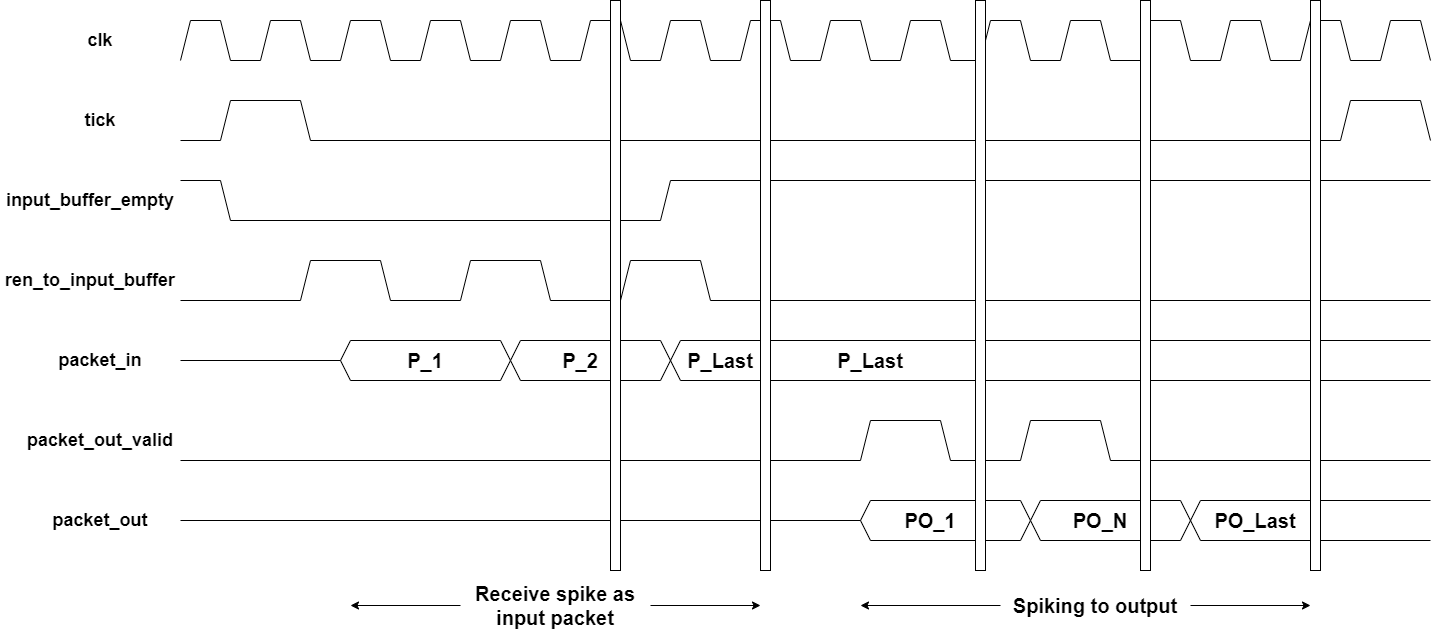
Hình 2.6 Sơ đồ khối SNN\_NetworkGrid

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Type** | **Description** |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **rst** | 1 | Input | Tín hiệu reset tích cực mức cao |
| **tick** | 1 | Input | Tín hiệu kích hoạt quá trình đọc và xử lý ảnh |
| **input\_buffer\_empty** | 1 | Input | Tín hiệu báo hiệu bộ đệm đầu vào đang rỗng hay đã sẵn sàng gửi các gói vào mạng |
| **packet\_in** | 30 | Input | Các gói kích thích đầu vào của mạng |
| **ren\_to\_input\_buffer** | 1 | Output | Tín hiệu read enable từ mạng gửi tới bộ đệm đầu vào để nhận gói kích thích |
| **packet\_out** | 8 | Output | Các gói đầu ra sau khi mạng hoàn tất quá trình xử lý trong một tick |
| **packet\_out\_valid** | 1 | Output | Tín hiệu báo hiệu có gói kích thích đầu ra |

Bảng 2.1 Bảng mô tả tín hiệu khối SNN\_NetworkGrid

### Mô tả chức năng khối Top module

Top module mô tả mức cao nhất cho mạng, trực tiếp nhận các gói kích thích đầu vào, xử lý và đưa ra những gói đầu ra tương ứng để thực hiện một chức năng nhất định.



Hình 2.7 Biểu đồ thời gian khối Top module

Khi nhận được tín hiệu tick, khối xử lý đầu vào sẽ kiểm tra trạng thái của bộ đệm đầu vào. Nếu bộ đệm đầu vào có dữ liệu (không rỗng), mạng đưa ra liên tiếp các tín hiệu read enable (ren\_to\_input\_buffer) để kích hoạt việc đọc đầu vào. Khi này các đầu vào được đưa vào mạng dưới dạng packet thông qua đường bus packet\_in có độ rộng 30 bit. Được biểu diễn trên Hình 2.7 trong khoảng “Receive spike as input packet”.

Sau khi nhận đủ gói, các gói kích thích này lan truyền trong mạng và tạo ra một mạng lưới kích thích qua lại giữa các Neurons. Kết quả cuối cùng của quá trình này là các gói packet\_out ở đầu ra, đi kèm với tín hiệu packet\_out\_valid báo hiệu đầu ra có dữ liệu. Được biểu diễn trên Hình 2.7 trong khoảng “Spiking to output”.

Lưu ý: tín hiệu tick chỉ lên trong chính xác một chu kỳ đồng hồ. Sau khi xử lý xong một ảnh và đưa ra các đầu ra thích hợp, mạng dừng hoạt động và tiếp tục chờ tín hiệu tick tiếp theo.

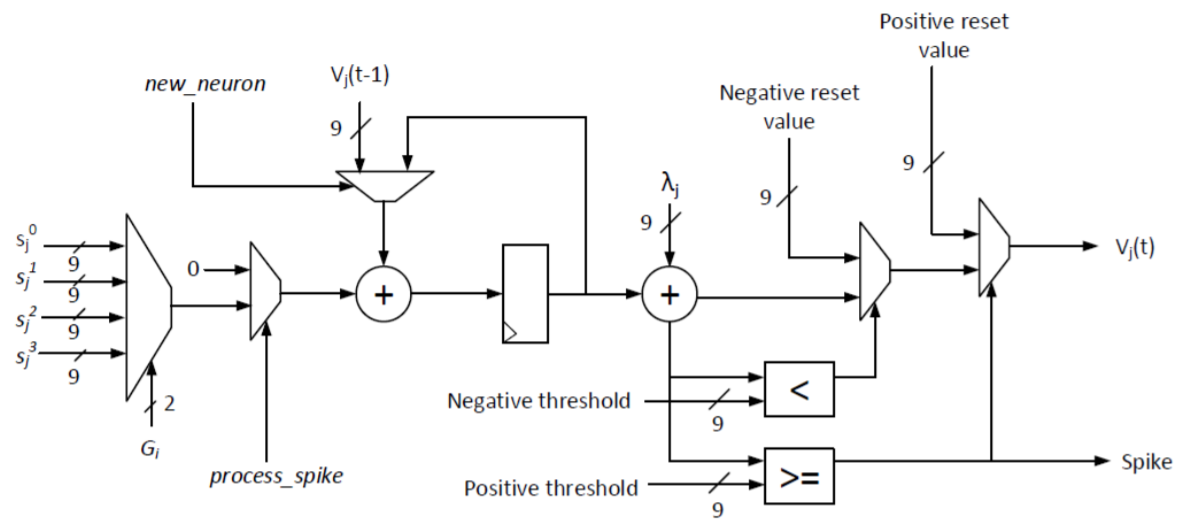
## Kiến trúc và hoạt động của các khối con trong một Core

### Neuron Block

Neuron Block là khối tính toán chính trong kiến trúc TrueNorth. Các bước tính toán xoay quanh một giá trị gọi là Potential (Membrain Potential), là giá trị tích lũy của Neuron, và giá trị này hoàn toàn độc lập giữa các Neuron khác nhau.

Bản thân khối Neuron Block trong thiết kế chỉ là một “lớp vỏ” gồm các khối cộng trừ, so sánh. Để Neuron Block thực sự trở thành một Neuron đúng nghĩa, nó cần phải được nạp vào một bộ tham số gồm 368 bits mô tả các thuộc tính của Neuron. Bộ tham số này quyết định việc Neuron được kết nối tới những Axons nào trong số 256 Axons trong Core, trọng số (weight) của các kết nối này là bao nhiêu, và khi Neuron phóng ra kích thích thì gói kích thích này sẽ được chuyển tới đích đến là Core nào… Bộ tham số của các Neurons trong Core đều được lưu trữ trong khối CSRAM và ý nghĩa các trường trong bộ tham số sẽ được đề cập chi tiết hơn tại mục 2.4.2.

Như vậy, tại một thời điểm xác định sẽ chỉ thật sự tồn tại một Neuron đang hoạt động thay vì 256 Neurons trong 1 Core. Neuron Block sẽ lần lượt được nạp những bộ tham số của các Neuron từ 0 đến 255, sau khi tính toán xong cho một Neuron, CSRAM lại tiếp tục nạp Neuron tiếp theo vào Neuron Block để tính toán. Quá trình “nạp nhân” vào “lớp vỏ” này được điều khiển bởi Token Controller trong mục 2.4.5.



Hình 2.8 Kiến trúc giản lược Neuron Block

Khi một kích thích được đưa vào Neuron, một trong số 4 trọng số được chọn dựa trên bộ tham số, giá trị này được cộng tích lũy với giá trị Potential ở trạng thái hiện tại của Neuron, tạo nên giá trị Potential ở thời điểm tiếp theo cho Neuron. Giá trị Potential khi này được so sánh với một giá trị ngưỡng (gồm ngưỡng âm và ngưỡng dương). Nếu giá trị Potential vượt quá ngưỡng dương, Neuron sẽ phóng ra một kích thích và thiết lập lại giá trị Potential về giá trị khởi tạo dương (Positive reset value). Ngược lại, nếu vượt ngưỡng âm, Neuron được thiết lập về giá trị khởi tạo âm (Negative reset value) mà không phóng ra kích thích. Nếu giá trị Potential không vượt qua ngưỡng âm hoặc ngưỡng dương, kích thích cũng sẽ không được phóng ra mạng, giá trị Potential của Neuron được bảo toàn cho lần tính toán tiếp theo. Các giá trị về ngưỡng hay giá trị reset đều được lưu trong CSRAM.

### CSRAM

CSRAM là khối lưu trữ chính của một Core, là một bộ nhớ gồm 256 hàng, trong đó mỗi hàng là 1 bộ 368 bits tham số ứng với một Neuron. Bộ tham số này thu được sau quá trình huấn luyện cho mạng. Chi tiết các trường trong 368 bits tham số được thể hiện trong bảng sau:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Width** | **Description** |
| **Synapse Connections** | 256 bits | Thể hiện kết nối của Neuron đang xét tới 256 Axons trong Core, bit 1 ứng với có kết nối. |
| **Current Potential** | 9 bits | Giá trị Potential tích lũy của Neuron. |
| **Reset Potential** | 9 bits | Giá trị Potential của Neuron sẽ được reset về giá trị này khi Neuron phóng ra kích thích. |
| **Weights 0 to 3** | 9 bits each | Giá trị trọng số của kết nối synapse đặc trưng cho từng Neuron, |
| **Leak Value** | 9 bits | Giá trị Potential rò rỉ khi tính toán. |
| **Positive Threshold** | 9 bits | Ngưỡng Potential dương. |
| **Negative Threshold** | 9 bits | Ngưỡng Potential âm. |
| **Reset mode** | 1 bit | Lựa chọn 2 Mode reset khác nhau của Neuron. |
| **Spike Destination** | 26 bits | Gồm các trường dx, dy rộng 9 bit và Axons destination 8 bit, giống với cấu trúc gói tin |
| **Tick delivery** | 4 bits | Giống với tick instance trong gói tin |

Bảng 2.2 Bảng mô tả các trường trong CSRAM

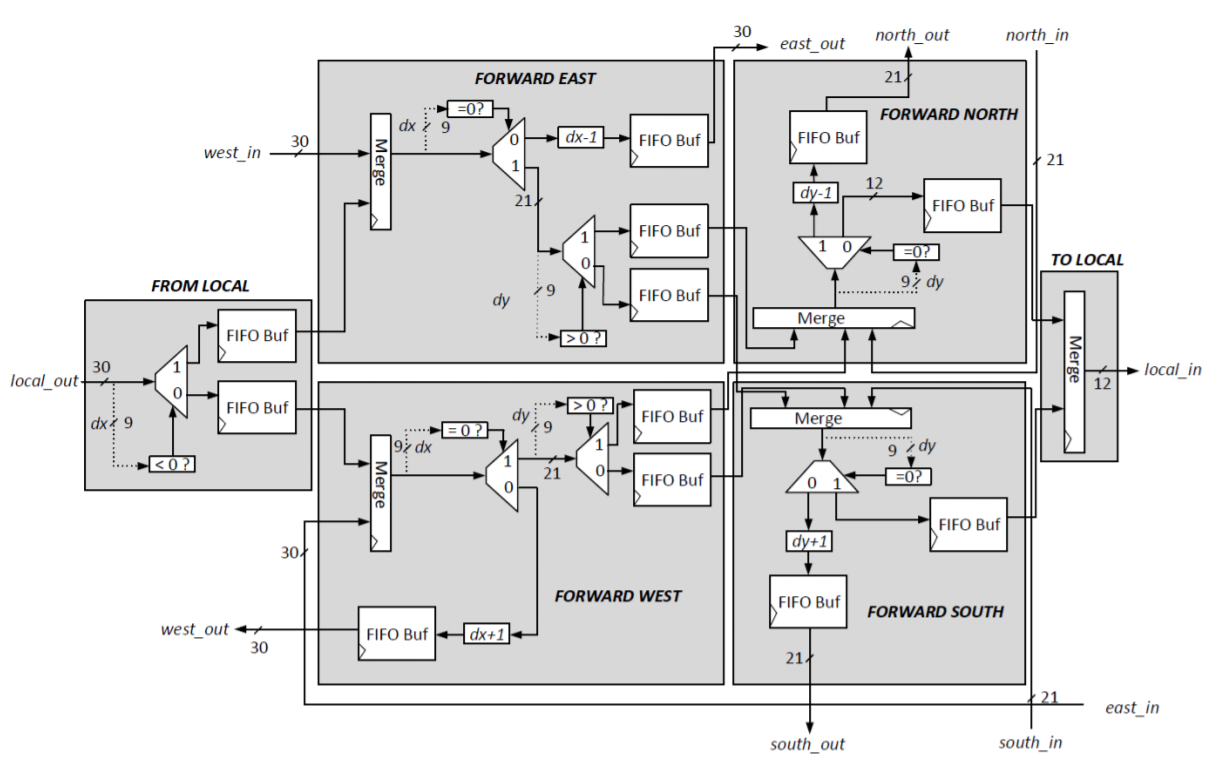
Như vậy, CSRAM lưu trữ toàn bộ giá trị tham số cho 256 Neuron trong mạng. Các tham số này quyết định tới hoạt động tính toán của Neuron, cũng như đích đến của các kích thích sinh ra bởi Neuron.

Ngoài ra, 9 bits Current Potential cũng liên tục được cập nhật lại trong CSRAM ứng với sự thay đổi về giá trị Potential của Neuron sau khi hoàn tất một quá trình tính toán.

### Router

Router phụ trách nhiệm vụ phân phối các gói kích thích lan truyền giữa các Core trong mạng. Khi một gói kích thích được đưa vào Router, nó sẽ bóc tách các trường trong gói tin để tính toán và quyết định việc chuyển tiếp gói tin này sang Core nào trong 4 Core lân cận theo các hướng Đông, Tây, Nam, Bắc. Nếu gói kích thích có địa chỉ đích chính là Core đang xét, gói sẽ được Router đưa vào trong Scheduler của Core thay vì đưa sang các Core xung quanh.

Router gồm 6 khối chính là From Local, To Local, Forward East, Forward West, Forward North và Forward South.

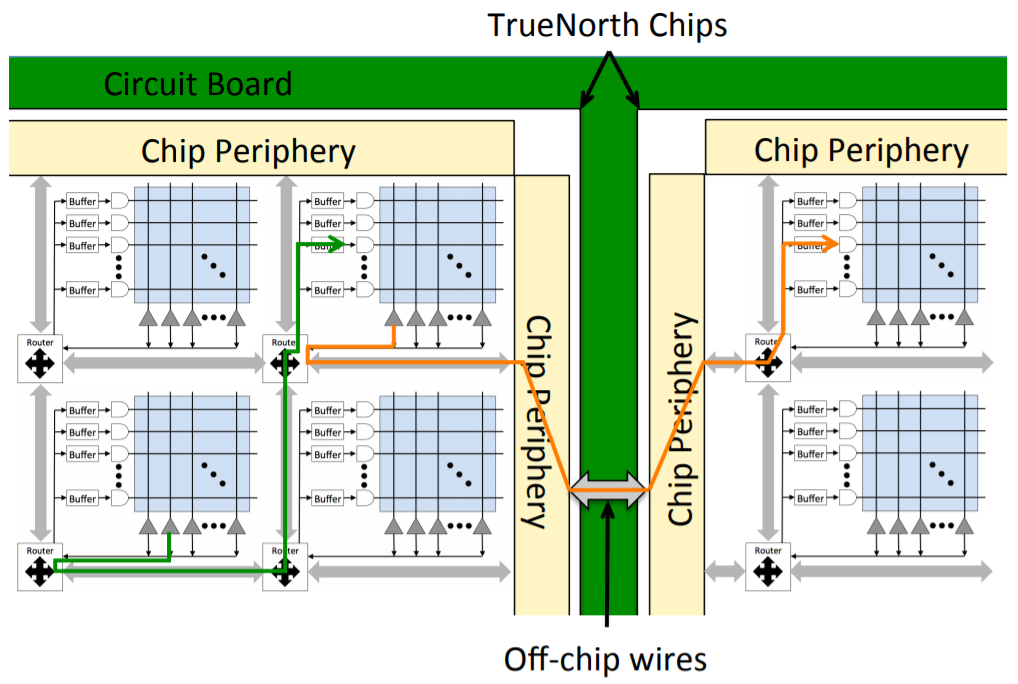


Hình 2.9 Kiến trúc Router

Một gói kích thích có thể đi tới Router từ các Core lân cận ở 4 hướng Đông, Tây, Nam, Bắc ứng với các đường east\_in, west\_in, south\_in và north\_in. Ngoài ra, cũng có thể gói tin tới Router được sinh ra bởi chính các Neurons trong Core, khi mà các Neurons này thực hiện tính toán và đưa ra các kích thích tới Router thông qua khối From Local.

Một gói kích thích đưa ra từ Router có thể được chuyển sang các Core lân cận theo 4 hướng Đông, Tây, Nam, Bắc thông qua các khối Forward East, Forward West, Forward South và Forward North. Nếu gói tin có đích đến là Core đang xét, kích thích sẽ được đưa vào Core thông qua khối To Local. Khi này thì gói kích thích chỉ còn giữ lại 12 bit gồm tick instance và axon destination.

Các đường màu xanh và cam trong Hình 2.10 mô tả các ví dụ về việc gói kích thích được phóng ra từ một Neuron trong Core, sau đó được gửi tới một đích đến xác định thông qua các Router trên đường di chuyển. Các gói sẽ được di chuyển lần lượt theo chiều ngang, dọc và tới được Core đích, cũng như Axon đích tại Core này.



Hình 2.10 Gói kích thích được lan truyền thông qua các Router

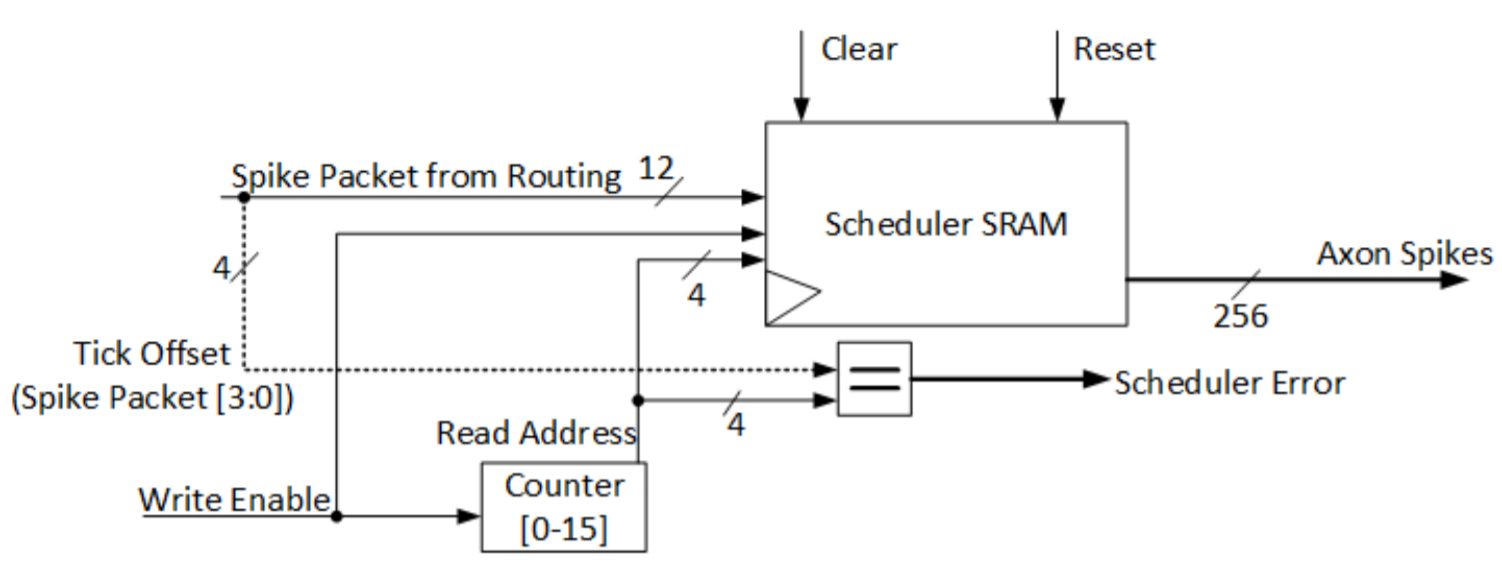
Một gói kích thích có cấu trúc như trong Hình 2.5 được lan truyền trong mạng lưới gồm nhiều Core như sau:

* Đầu tiên, Router trích xuất trường dx trong gói, đưa gói kích thích di chuyển theo chiều ngang:
* Nếu dx > 0: gói kích thích được truyền sang Core hướng Đông (sang phải), thay đổi trường dx trong gói: dx = dx – 1.
* Nếu dx < 0: gói kích thích được truyền sang Core hướng Tây (sang trái), thay đổi trường dx trong gói: dx = dx + 1.
* Nếu dx = 0: loại bỏ trường dx trong gói, bắt đầu truyền dọc.
* Sau khi giá trị dx = 0, gói kích thích loại bỏ đi trường dx, khi này gói chỉ còn 21 bits và tiếp tục được di chuyển theo chiều dọc dựa theo trường dy:
* Nếu dy > 0: gói kích thích được truyền sang Core hướng Bắc (lên trên), thay đổi trường dy trong gói: dy = dy – 1.
* Nếu dy < 0: gói kích thích được truyền sang Core hướng Nam (xuống dưới), thay đổi trường dy trong gói: dy = dy + 1.
* Nếu dy = 0: gói kích thích đã tới được Core đích. Tiếp tục loại bỏ trường dy trong gói và đưa gói vào Scheduler trong Core đang xét.

### Scheduler

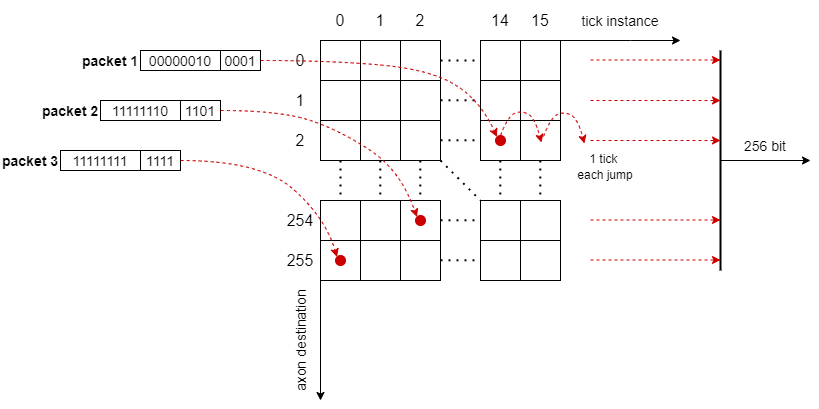
Khi các gói kích thích truyền thành công tới Core đích, các gói này được Router trong Core này đưa vào lưu trữ tại Scheduler. Khi này gói tin chỉ còn 12 bits gồm hai trường tick instance và axon destination, do các trường xác định Core là dx, dy đã bị loại bỏ.

Tại tick tiếp theo, Scheduler thực hiện nhiệm vụ phân phát các gói kích thích đã được nhận từ Router trong các tick trước đó tới 256 Axons trong Core. Như vậy Scheduler cần một khối “Scheduler SRAM” để lưu trữ, cùng với đó là một khối sinh địa chỉ đọc, ứng với bộ đếm Counter, thể hiện như trong Hình 2.11.



Hình 2.11 Kiến trúc Scheduler

Trong đó khối Scheduler SRAM có cấu trúc gồm 256 hàng và 16 cột, ứng với hai trường là axon destination 8 bits và tick instance 4 bits.



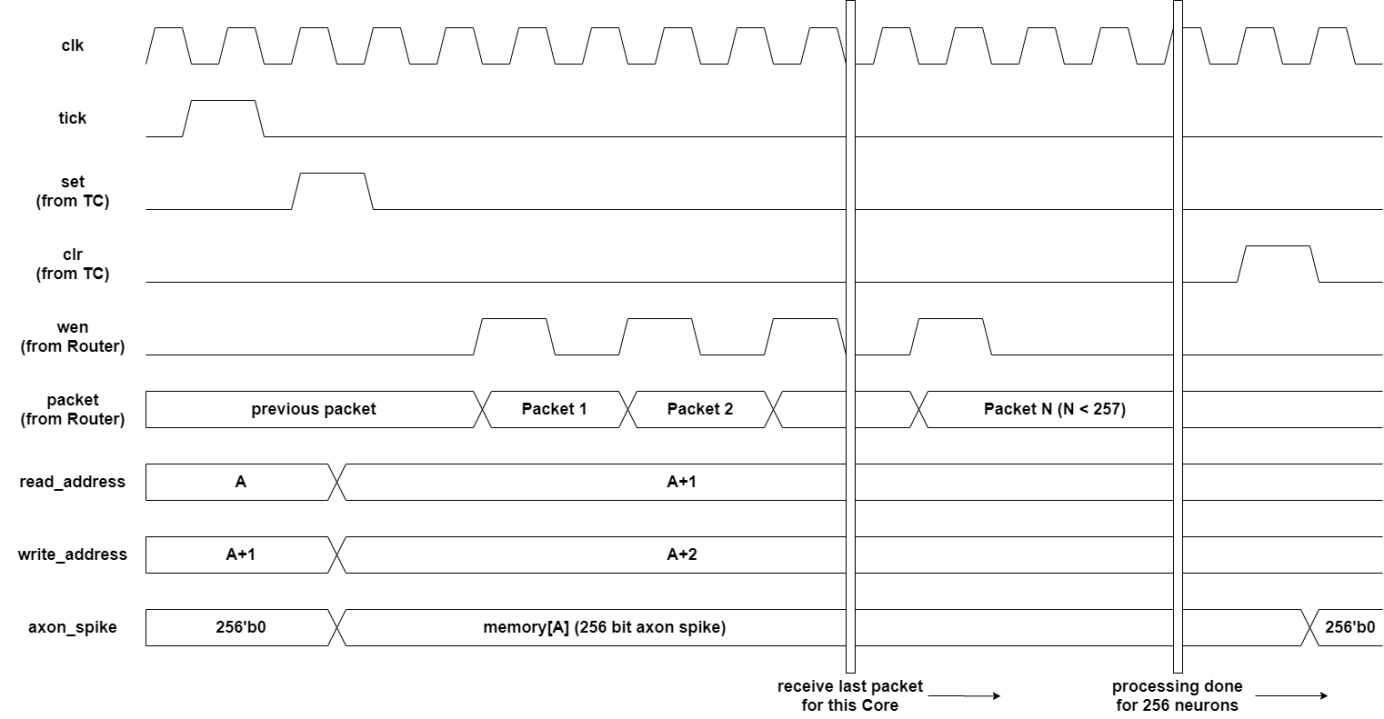
Hình 2.12 Mô tả hoạt động của Scheduler

Hoạt động của Scheduler được mô tả một cách đơn giản hơn như sau (ứng với hình Hình 2.12):

* Tại mỗi tick, các gói được gửi tới Scheduler và điền vào các ô nhớ dưới dạng các chấm đỏ như trong hình. Việc điền vào hàng nào được xác định bởi 8 bits axon destination, ứng với việc kích thích đó sẽ được phóng ra từ Axon nào. Tương tự, việc điền vào cột nào được xác định bởi 4 bits tick instance, xác định thời điểm mà Axon trong core sẽ phóng ra kích thích.
* Sau mỗi tick, dữ liệu trong các cột đều được dịch phải sang cột lân cận, các gói trong cột thứ cuối cùng (cột 15) được đưa ra thành các kích thích của 256 Axons trong tick hiện tại, ứng với 256 bits Axon Spikes trong Hình 2.11.

Như vậy, có thể coi Scheduler thực hiện 2 chức năng chính như sau:

* Với 4 bits tick instance: Scheduler có chức năng như một hàng đợi/tập thanh ghi dịch. Các gói kích thích sẽ được giữ tại Scheduler tối đa trong khoảng thời gian 16 ticks, trước khi chuyển thành các kích thích phóng tới các Neuron.
* Với 8 bits axon destination: Scheduler có chức năng như một bộ giải mã. Đầu vào là các gói với 8 bit axon destination dưới dạng mã nhị phân, đầu ra là 256 bits Axon Spikes (các vị trí có bit 1 trong 256 bits Axon Spikes thể hiện là có kích thích và ngược lại với bit 0).



Hình 2.13 Biểu đồ thời gian của Scheduler

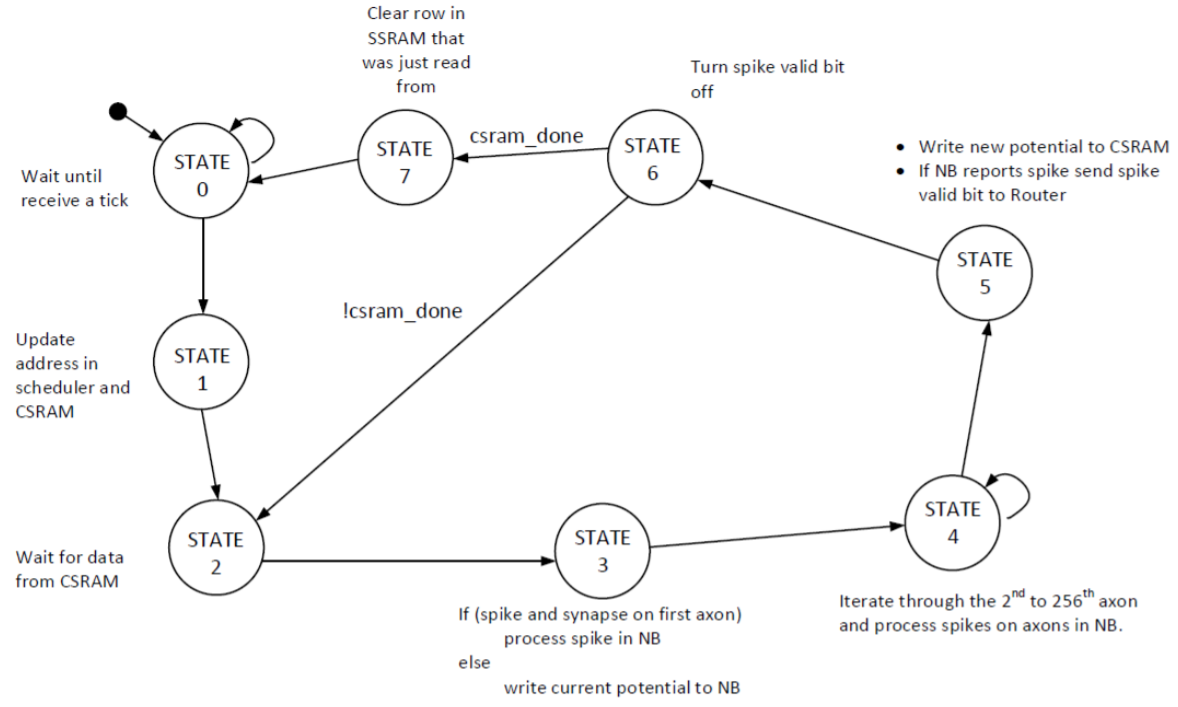
### Token Controller

Token Controller là một máy trạng thái với tổng cộng 8 trạng thái, điều khiển quá trình “nạp nhân” lần lượt cho 256 Neurons từ CSRAM vào Neuron Block.

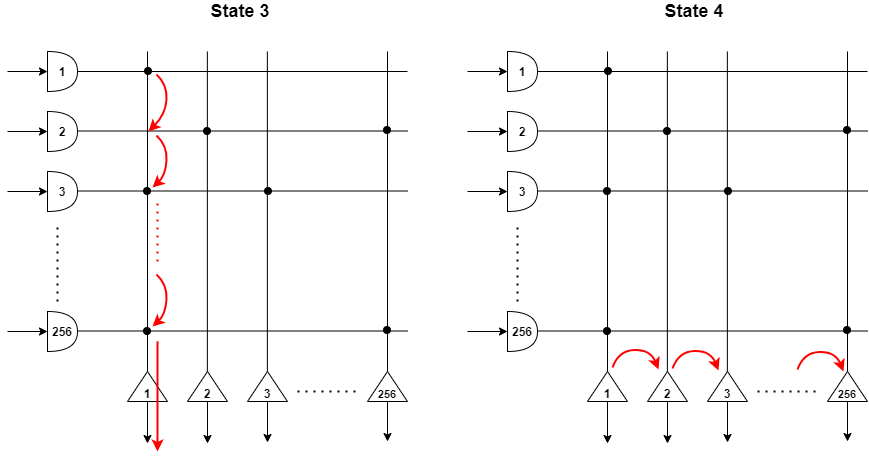
Dựa vào 256 bits Axon Spikes nhận được từ Scheduler, Token Controller duyệt từng bit (từng Axon) để lấy ra các spike tại các vị trí có bit 1 và kiểm tra kết nối synapse. Nếu thỏa mãn đồng thời hai điều kiện là Axon có spike, và có tồn tại liên kết synapse giữa Axon và Neuron đang xét, thì kích thích được đưa từ Axon vào Neuron để tính toán, tích lũy giá trị Potential. Sau khi duyệt xong 256 Axons cho một Neuron, Token Controller chuyển sang Neuron tiếp theo và lặp lại quá trình duyệt này cho tới Neuron cuối cùng.

Sau khi tính toán xong 256 Axons cho một Neuron, nếu Neuron này đạt tới giá trị ngưỡng và phóng ra kích thích, Token Controller cũng nhận kích thích này và đưa vào Router dưới dạng một gói tin gồm 30 bits, đồng thời cập nhật lại giá trị Current Potential của Neuron đang xét vào trong CSRAM.

Cuối cùng, sau khi hoàn thành tính toán cho toàn bộ 256 Neurons trong Core, Token Controller gửi ra tín hiệu clear\_scheduler giúp xóa toàn bộ dữ liệu trong cột cuối cùng của khối Scheduler.

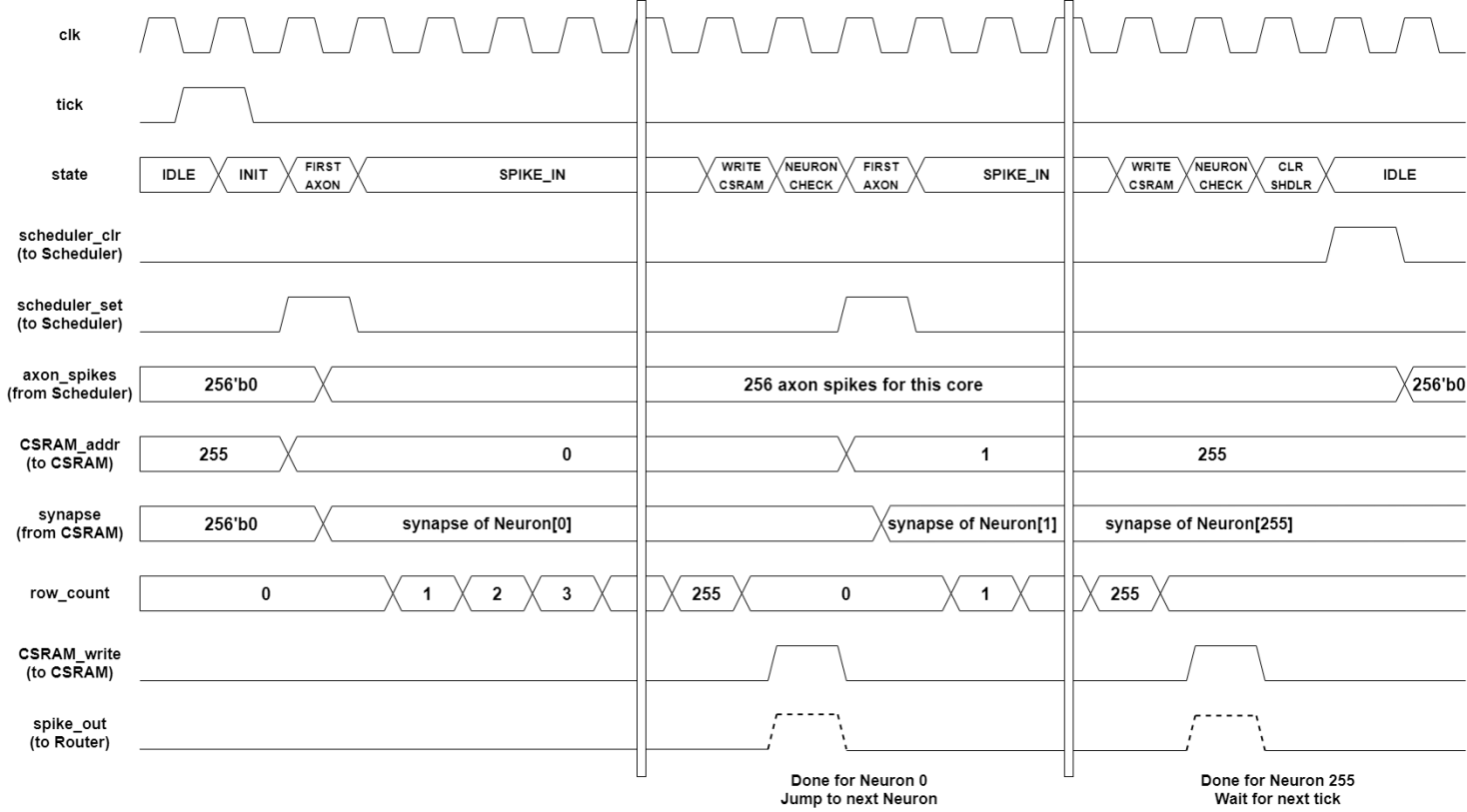


Hình 2.14 Máy trạng thái Token Controller



Hình 2.15 Mô tả hoạt động Token Controller

Hình 2.15 mô tả quá trình tính toán của một Neuron khi duyệt qua 256 Axons theo chiều dọc, và quá trình này được lặp lại cho toàn bộ 256 Neurons trong Core.



Hình 2.16 Biểu đồ thời gian Token Controller

Như vậy, có thể ước lượng một cách gần đúng thời gian dành riêng cho quá trình tính toán này xấp xỉ 256 \* 256 = 65536 chu kỳ clk (duyệt 256 Neurons, mỗi Neuron duyệt 256 Axons). Đây là một con số rất lớn, gây ra do việc các Neurons đang được tính toán một cách tuần tự, lần lượt. Từ đây hình thành ý tưởng về việc hoạt động song song cho 256 Neurons trong Core, tạo ra một mạng SNN với kiến trúc Fully Parallel, tối ưu tốc độ xử lý một cách đột phá, được đề xuất tại CHƯƠNG 3.

# ĐỀ XUẤT KIẾN TRÚC MẠNG SNN FULLY PARALLEL TỐI ƯU TỐC ĐỘ XỬ LÝ

## Tổng quan ý tưởng cho mạng SNN Fully Parallel

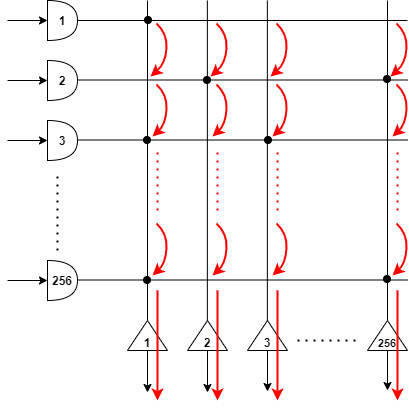
Đối với kiến trúc TrueNorth, các Neurons được tính toán một cách hoàn toàn độc lập, không ảnh hưởng đến kết quả lẫn nhau. Tuy nhiên quá trình tính toán này lại được thực hiện một cách tuần tự, lần lượt giữa các Neurons. Nghĩa là trong một thời điểm chỉ có duy nhất một Neuron thực sự xuất hiện, phải chờ đến khi Neuron này hoàn tất quá trình tính toán thì một Neuron tiếp theo mới được nạp vào Neuron Block để tiếp tục tính toán.

Quá trình hoạt động của mỗi Neuron lại có sự tương đồng khi cùng phải duyệt qua 256 Axons để nhận kích thích từ các Axons này (nếu có). Xuất phát từ sự tương đồng này, kiến trúc SNN Fully Parallel được đề xuất, hướng tới việc thực hiện tính toán cho đồng thời cả 256 Neurons. Như vậy việc duyệt qua 256 Axons sẽ được thực hiện song song giữa các Neurons trong một Core. Quá trình tích lũy Potential cho Neurons sẽ chỉ mất 256 thay vì 256\*256 chu kỳ đồng hồ clk, tối ưu đột phá về mặt tốc độ so với kiến trúc ban đầu.

Trong một mạng SNN lớn gồm nhiều Core, bản thân các Core đã được hoạt động song song với nhau. Vậy khi các phần tử tính toán bên trong Core là Neurons cũng hoạt động song song, ta sẽ thu được một mạng lưới lớn gồm vô số Neurons hoạt động đồng thời, mô tả chính xác hơn về cấu tạo bộ não con người.

## Các thay đổi về kiến trúc mỗi Core so với kiến trúc thông thường

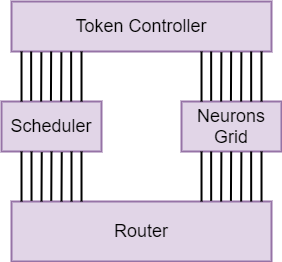
Để thực hiện được kiến trúc Fully Parallel, ý tưởng về hoạt động “nạp nhân” CSRAM vào “lớp vỏ” Neuron Block sẽ bị loại bỏ. Thay vào đó, module Neurons Grid được hình thành để thay thế, với toàn bộ 256 Neurons cùng xuất hiện đồng thời với đầy đủ các tham số cần thiết. Như vậy về kiến trúc chung, mỗi Core sẽ thay thế hai khối CSRAM và Neuron Block thành một khối Neurons Grid.



Hình 3.1 Hoạt động song song của 256 Neurons trong một Core

Ngoài ra, máy trạng thái Token Controller cũng cần được thay đổi để điều khiển quá trình tính toán cho Neurons Grid một cách thích hợp. Cùng với đó là một số sự thay đổi khác về kiến trúc sẽ được đề cập kỹ hơn ở các mục dưới.

### Thay đổi cấu tạo của một Core



Hình 3.2 Cấu tạo một Core trong kiến trúc Fully Parallel

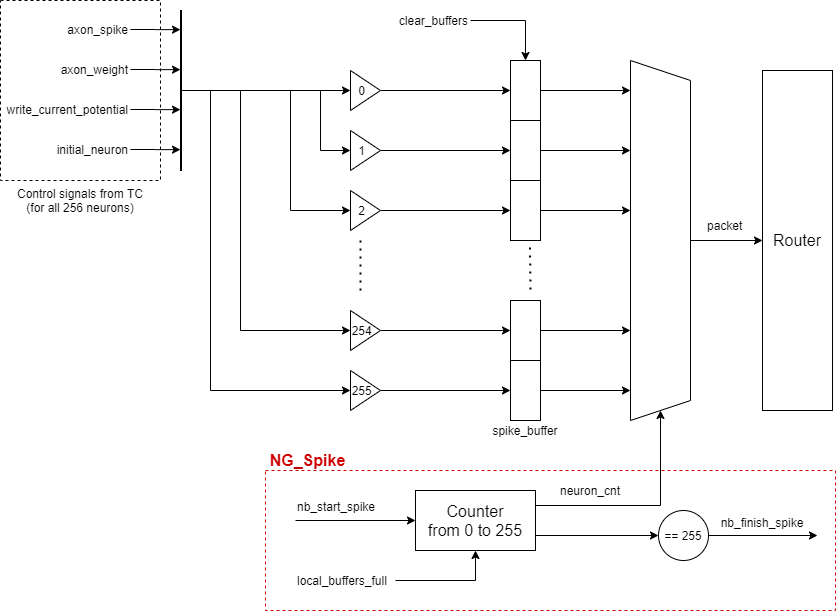
Hình 3.2 mô tả kiến trúc của một Core đối với kiến trúc SNN Fully Parallel. Khi này Core chỉ còn 4 khối chính là Router, Scheduler, Token Controller và Neurons Grid. So với kiến trúc cũ, hai khối CSRAM và Neuron Block đã bị loại bỏ và thay thế bởi module Neurons Grid.

### Neurons Grid thay thế cho CSRAM & Neuron Block

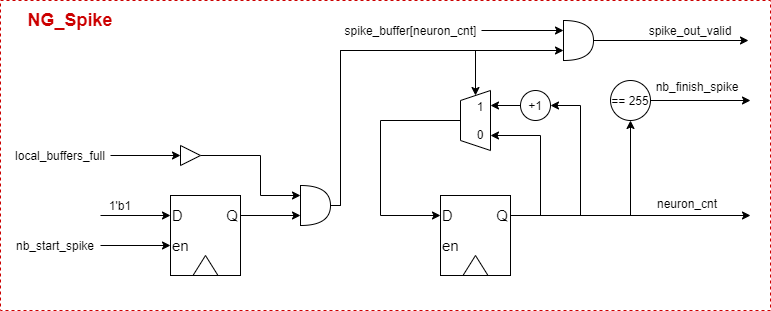
Neurons Grid chứa tổng cộng 256 mẫu tham chiếu của Neuron Block, ứng với 256 Neurons trên mạch. Các Neurons này đã được nạp sẵn bộ tham số thu được trong quá trình huấn luyện mạng. Như vậy có thể hiểu, Neurons Grid thu được khi sát nhập CSRAM vào 256 Neuron Block, khiến cho 256 Neurons đồng thời hoạt động, tích lũy Potential và cùng (có thể) phóng ra kích thích.

Tuy nhiên, việc hoạt động song song dẫn đến một hệ quả quan trọng cần phải khắc phục. Đối với kiến trúc ban đầu, mỗi Neuron cần thực hiện tính toán với 256 Axons, tức là mất 256 chu kỳ clk để có thể đưa ra một gói kích thích tới Router, sau đó mới đến Neuron tiếp theo hoạt động. Như vậy, khoảng cách gần nhất giữa việc hai gói tin được gửi tới Router là 256 chu kỳ. Nhưng với kiến trúc song song, sau khi duyệt tới Axon cuối cùng trong 256 Axons, cả 256 Neurons đều có thể đồng loạt phóng ra kích thích tại cùng một thời điểm, đặt ra một thách thức lớn trong việc phân phối hàng loạt gói tin này (tối đa 256 gói) tới Router một cách thích hợp.

Vì vậy, trong kiến trúc Neurons Grid sẽ bao gồm một khối con là NG\_Spike, và hoạt động của Neurons Grid được mô tả chi tiết như sau (ứng với Hình 3.3):

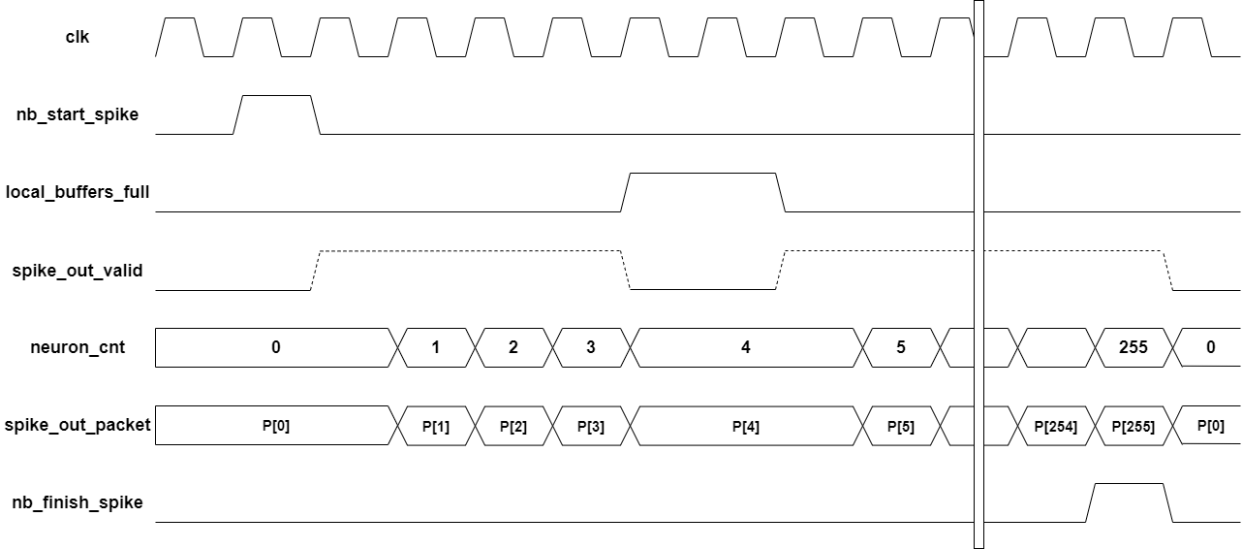


Hình 3.3 Kiến trúc tổng quan Neurons Grid



Hình 3.4 Kiến trúc chi tiết khối NG\_Spike

* 256 Neuron Block thuộc Neurons Grid nhận được các tín hiệu điều khiển tới từ Token Controller trong 256 chu kỳ liên tục, tính toán và tích lũy Potential theo 256 Axons.
* Khi duyệt tới Axons cuối cùng, những Neurons có giá trị Potential vượt ngưỡng dương sẽ đồng loạt phóng kích thích tới một bộ đệm là spike\_buffer. Như vậy bộ đệm này sẽ lưu 256 bit ứng với 256 Neurons, tại vị trí bit thứ n xuất hiện bit 1 có nghĩa là Neurons thứ n có phóng ra kích thích, và ngược lại với bit 0.
* Sau khi thu thập đủ spike\_buffer, khối NG\_Spike đợi tín hiệu kích hoạt tới từ Token Controller. Khi nhận được tín hiệu kích hoạt này, NG\_Spike sẽ lần lượt đưa các gói kích thích được lưu trong spike\_buffer tới Router trong vòng 256 chu kỳ clk (nếu bộ đệm tại Router không bị đầy).
* Sau khi hoàn tất việc đưa các gói từ spike\_buffer tới Router, NG\_Spike đưa ra tín hiệu finish tới Token Controller báo hiệu hoàn thành quá trình phóng kích thích vào mạng.

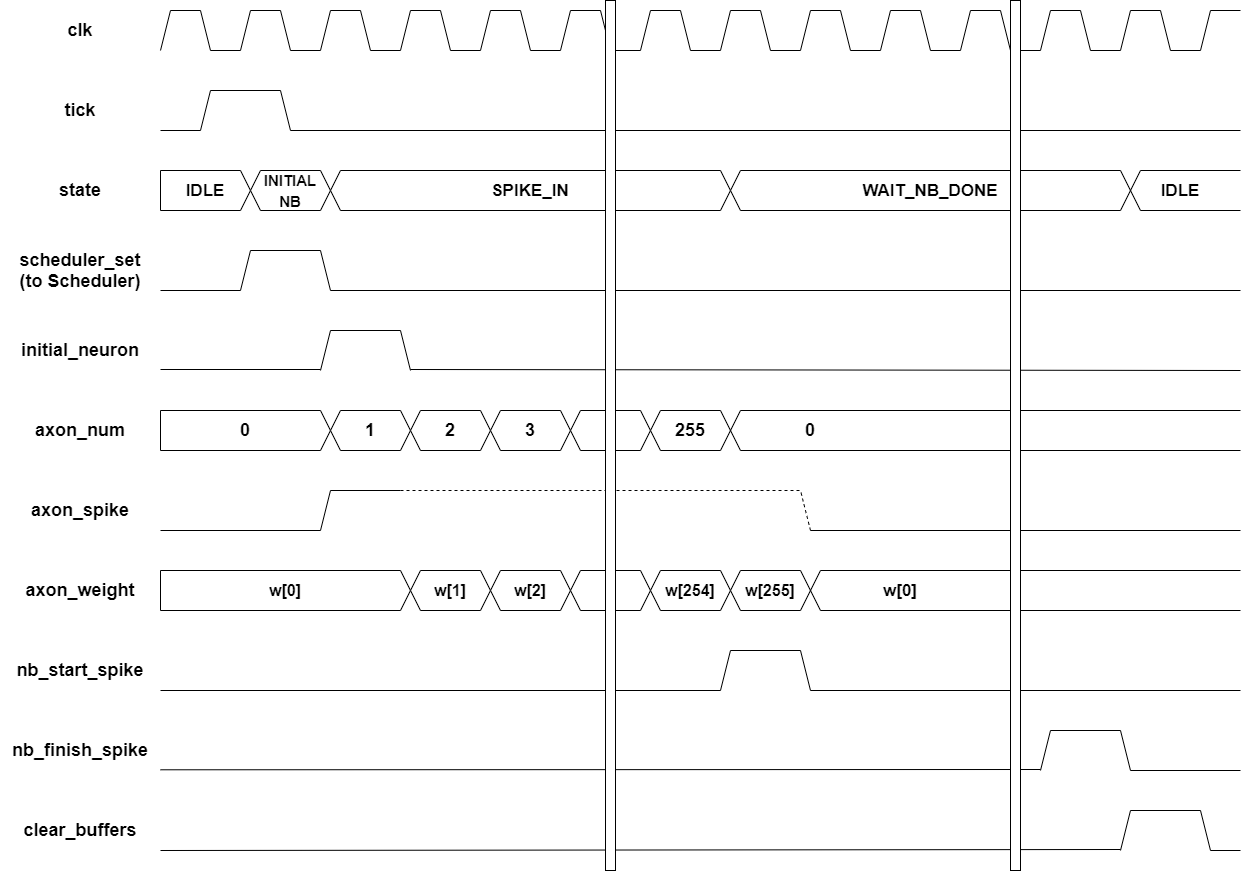


Hình 3.5 Biểu đồ thời gian khối NG\_Spike

### Thiết kế lại máy trạng thái Token Controller

Đối với kiến trúc mạng SNN Fully Parallel, các hoạt động giao tiếp, điều khiển giữa các khối trong mạng đã được thay đổi gần như hoàn toàn. Vì vậy khối Token Controller cũng được thiết kế lại để điều khiển hoạt động song song cho 256 Neurons, và điều khiển quá trình đưa các gói từ Neurons Grid vào mạng.

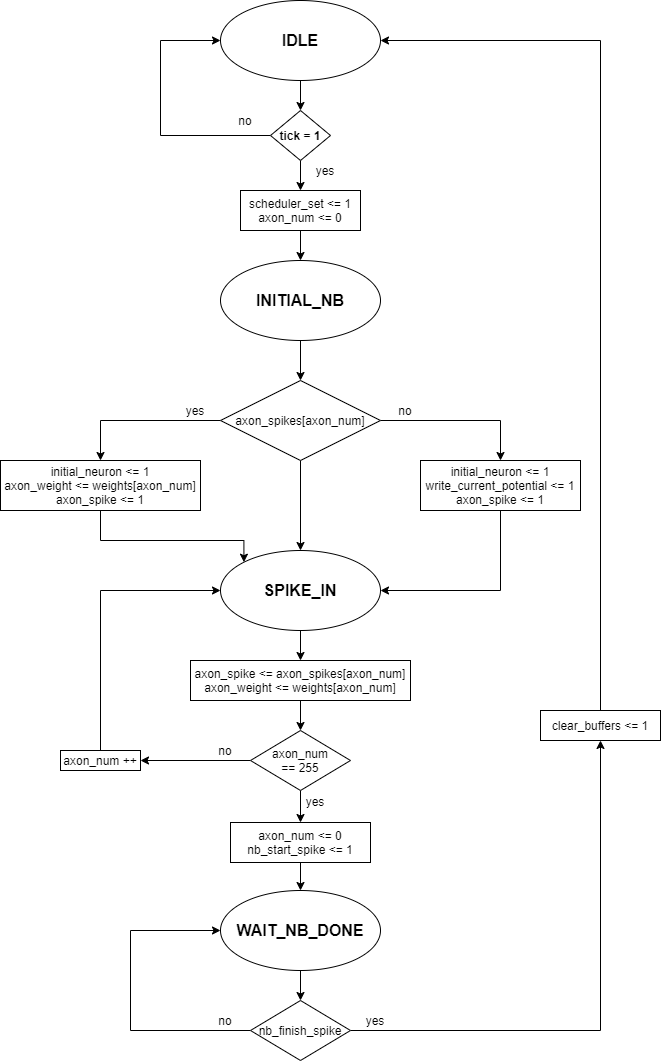
Sau khi được thiết kế lại một cách phù hợp và loại bỏ đi một số trạng thái dư thừa không thực sự cần thiết trong kiến trúc ban đầu, máy trạng thái Token Controller khi này còn tổng cộng 4 thay vì 8 trạng thái, thiết kế theo kiểu Mealy.



Hình 3.6 Biểu đồ thời gian Token Controller

Khi nhận được một tick, Token Controller đưa ra tín hiệu scheduler\_set để kích hoạt khối Scheduler, sau đó lần lượt đưa ra các tín hiệu điều khiển tới Neurons Grid để bắt đầu quá trình duyệt qua 256 Axons. Tại vị trí mà Axon thứ n có phóng kích thích, tín hiệu axon\_spike được đưa lên 1, đồng thời trọng số của Axon thứ n đó cũng được đưa tới Neurons Grid để tính toán cho các Neurons.

Sau khi duyệt xong tới Axon cuối cùng, Token Controller đợi tín hiệu finish từ Neurons Grid để biết khối này đã phóng hết tất cả kích thích vào mạng, từ đó chuyển về trạng thái IDLE ban đầu, đưa ra tín hiệu giúp xóa các bộ đệm nếu cần thiết.



Hình 3.7 Sơ đồ dịch chuyển trạng thái của Token Controller

### Tăng kích thước bộ đệm trong Router

Khi các Neurons hoạt động song song, đồng nghĩa với việc các gói kích thích được sinh ra từ các Neurons và phóng vào mạng với tần suất dày đặc hơn. Đặc biệt trong trường hợp một Core đồng thời nhận được gói từ Neurons của nhiều Core lân cận khác, việc tràn bộ đệm trong Router tất yếu sẽ xảy ra, khiến cho mạng trở nên tắc nghẽn, gây ra mất gói, khiến mạng không thực hiện được đúng chức năng ban đầu.

Để khắc phục tình trạng tắc nghẽn này, các bộ đệm trong Router cần thiết phải tăng kích cỡ để có thể lưu trữ được nhiều gói tin hơn. Các gói tin sẽ chủ yếu được hình thành và đưa tới Router trong quá trình 256 chu kỳ hoạt động của khối NG\_Spike tại mục 3.2.2, từ đó điền đầy vào các bộ đệm trong Router. Sau quá trình này, NG\_Spike sẽ dừng đưa gói ra trong một khoảng thời gian nhất định, cho tới khi nhận được tick tiếp theo. Khoảng thời gian này đủ để Router lan truyền các gói tới các đích đến phù hợp, giải phóng cho các bộ đệm.

Vậy mấu chốt của vấn đề sẽ được giải quyết nếu các bộ đệm trong Router đủ lớn để tải được toàn bộ các kích thích đưa ra từ NG\_Spike trong một thời gian ngắn (256 chu kỳ). Khi tiến hành khảo sát thực tế, kích thước bộ đệm trong các Router tăng lên từ 4 thành 128 hàng sẽ đảm bảo được hoạt động của mạng ổn định, không bị tắc nghẽn (với mỗi hàng có kích thước là 30 bits, bằng với kích thước của một gói kích thích).

Tuy nhiên việc tăng kích thước bộ đệm trong các Router sẽ khiến cho kích thước của mỗi Core bị tăng lên, và làm tăng diện tích mạch đối với toàn bộ mạng. Các thông số đo đạc về sự chênh lệch diện tích này sẽ được đề cập chi tiết tại mục 3.3.

### Đề xuất tối giản Scheduler

Trong thực tế, một kích thích được truyền từ môi trường bên ngoài vào trong mạng sẽ liên tục được lan truyền để tới được các Neurons đích. Việc một gói kích thích được giữ lại tại Scheduler của một Core trong nhiều ticks dựa trên trường tick instance không thực sự mang nhiều ý nghĩa. Và các hoạt động của mạng liên quan tới chức năng như đọc và xử lý ảnh trên thực tế cũng chưa sử dụng tới trường này (4 bit tick instance được thiết lập giá trị 0 trong tất cả các gói đưa vào mạng).

Như vậy, đối với kiến trúc Scheduler như trong mục 2.4.4, thực tế chỉ có cột cuối cùng trong tổng cộng 16 cột của Scheduler SRAM được sử dụng. Với những cột còn lại, do trường tick instance có giá trị là 0 nên sẽ không được điền vào bởi các gói kích thích. Vì vậy, kiến trúc SNN Fully Parallel trong tài liệu đề xuất việc tối giản khối Scheduler với việc chỉ giữ lại cột một cột thanh ghi gồm 256 bit. Các gói gửi tới Scheduler trong một tick được điền vào cột duy nhất này, và ngay tại tick tiếp theo dữ liệu trong cột sẽ được đưa ra Router, từ đó giảm diện tích mạch của toàn mạng, cải thiện một phần đánh đổi về diện tích mạch khi tăng bộ đệm Router tại 3.2.4. Những thông số về mặt diện tích, năng lượng cũng được tổng hợp, thống kê và so sánh một các chi tiết tại mục 3.3.

## Tính toán tốc độ xử lý dựa trên lý thuyết

* Kiến trúc Fully Parallel: tại chu kỳ đầu, máy trạng thái chuyển từ IDLE sang INITIAL\_NB, sau đó liên tục duyệt cho 256 Axons tại mỗi chu kỳ. Sau khi duyệt xong cho các Axons, khối Neurons Grid bắt đầu đưa ra các kích thích dựa trên 256 Neurons trong Core. Để tránh trường hợp bị tràn bộ đệm, Neurons Grid được thiết kế duyệt mỗi Neuron trong 2 chu kỳ với mục đích giảm tần suất xuất hiện kích thích. Sau đó mạng hoàn thành tính toán cho một tick và dừng hoạt động, chờ tick tiếp theo. Như vậy kiến trúc Fully Parallel sẽ mất tổng cộng (1 + 256 + 256\*2) = 769 chu kỳ để hoàn thành tính toán cho một ảnh.
* Kiến trúc ban đầu (sử dụng thiết kế RANC): sau hai chu kỳ chuyển trạng thái, Token Controller bắt đầu tính toán cho từng Neuron trong tổng cộng 256 Neurons của một Core. Với mỗi Neuron, máy trạng thái duyệt qua 256 Axons trong mỗi chu kỳ, sau đó mất thêm 2 chu kỳ chuyển trạng thái để xóa bộ đệm Scheduler và thiết lập về trạng thái ban đầu để tiếp tục tính toán cho Neuron kế tiếp. Như vậy kiến trúc thông thường dựa theo thiết kế RANC sẽ mất tổng cộng (2 + (256+2)\*256) = 66050 chu kỳ để hoàn thành tính toán cho một ảnh.

Như vậy, trên lý thuyết, nếu thực hiện trên cùng một tần số thì tốc độ xử lý của kiến trúc Fully Parallel có thể tăng gấp khoảng 86 lần so với kiến trúc TrueNorth thông thường.

## So sánh và nhận xét kết quả

### Chức năng của mạng theo kiến trúc Fully Parallel

Sau khi tiến hành thiết kế mạng SNN Fully Parallel với tất cả những thay đổi về kiến trúc được đề cập tại các mục trên. Mạng SNN đã được mô phỏng và tổng hợp lại một cách toàn diện, đảm bảo các chức năng, kết quả tính toán vẫn được giữ nguyên so với kiến trúc ban đầu.

Việc tiến hành kiểm thử và so sánh chức năng được thực hiện trên tập MNIST gồm 10.000 ảnh chữ số viết tay với cùng một bộ tham số huấn luyện, sử dụng phần mềm Modelsim, ngôn ngữ Verilog, được mô tả kỹ hơn tại CHƯƠNG 4.

### Phương pháp so sánh các thiết kế

Các kiến trúc trước và sau khi thực hiện những thay đổi khác nhau trong mạng đều được tổng hợp bằng phần mềm chuyên dụng theo công nghệ ASIC, với mục đích so sánh các thông số về tốc độ xử lý, diện tích mạch, năng lượng tiêu hao:

* Phần mềm tổng hợp: Design Compiler – Synopsys.
* Tần số tổng hợp: 100 MHz - đây cũng là tần số tối đa của cả hai thiết kế để không vi phạm phải các ràng buộc về mặt timing khi tổng hợp.
* Công nghệ: sử dụng công nghệ 40nm TSMC.
* Thư viện: Dolphin Technology Standard Cells.
* Số lượng Core trong mạng: 5 Core.

Kiến trúc mạng SNN Fully Parallel trong bài sẽ được so sánh với **RANC**, một thiết kế mã nguồn mở được xây dựng dựa trên kiến trúc True North.

### Các kết quả thu được sau khi tổng hợp và khảo sát

Hai thiết kế RANC và Fully Parallel được tổng hợp trên cùng tần số 100 MHz với cùng một thư viện công nghệ. Mọi thông số về điều kiện hoạt động, các ràng buộc, lựa chọn tối ưu của phần mềm khi tổng hợp cũng đều được thiết lập giống nhau để thu được kết quả so sánh trực quan nhất.

Ngoài ra một các sự thay đổi về kích thước bộ đệm trong Router, đề xuất tối giản cho Scheduler cũng được thử nghiệm riêng trong từng trường hợp, từ đó giúp quan sát được tác động của những sự thay đổi này lên các thông số tổng hợp mạch.

Tất cả các sự thay đổi được liệt kê so sánh đều phải đảm bảo mạng SNN vẫn hoạt động đúng với chức năng ban đầu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Synthesis Network with 5 Core and 1 output Router** | **Speed (image/s)** | **Area (µm²)** | **Power (mW)** | **Cells** |
| Fully Parallel (our implemented) | 130039 | 4241755 | 184.9 | 1674856 |
| RANC | 1514 | 3313342 | 97.9 | 1231540 |
| RANC with simplier Scheduler | 1514 | 3178051 | 93.6 | 1189308 |
| RANC with Buffer depth 128 | 1514 | 4720340 | 144.3 | 1770562 |

Bảng 3.1 So sánh thông số các mạng SNN với 5 Core

Trong đó thông số về tốc độ xử lý (ảnh/s) thu được từ quá trình mô phỏng chức năng của mạng. Với thiết kế RANC, khoảng thời gian tính từ khi nhận được 1 tick cho tới khi hoàn thành quá trình xử lý 1 ảnh (trạng thái cuối của Token Controller) mất tổng cộng 66050 chu kỳ clk. Khoảng thời gian này đối với thiết kế Fully Parallel trong bài là 769 chu kỳ clk. Kết hợp với thông số về tần số hoạt động 100 MHz, có thể dễ dàng tính được tốc độ xử lý ảnh của các thiết kế này.

### Nhận xét kết quả

Với kiến trúc mạng SNN Fully Parallel được đề xuất trong bài, tốc độ xử lý dữ liệu được tăng cường một cách đột phá, gấp khoảng 86 lần so với kiến trúc True North thông thường (13021 ảnh/giây so với 151 ảnh/giây).

Tuy nhiên, đánh đổi với tốc độ xử lý, diện tích mạch sẽ tăng xấp xỉ 28% (từ 3313342µm² lên 4241755 µm²) do hệ quả khi phải tăng kích thước bộ đệm trong các Router, cũng như việc xuất hiện động thời 256 Neuron Block thay vì chỉ 1 Neuron Block duy nhất. Thông số này cũng được cải thiện một phần nhờ vào việc tối giản thiết kế của Scheduler, cũng như thiết kế lại máy trạng thái Token Controller và một số thay đổi khác.

Về năng lượng tiêu thụ, do có số lượng cells lớn, kiến trúc SNN fully parallel tăng gần gấp đôi về năng lượng tiêu thụ so với RANC, 184.9 mW so với 97,9 mW. Tuy nhiên trong cùng một khoảng thời gian, mạng theo kiến trúc song song lại xử lý được nhiều ảnh hơn (gấp 86 lần). Vì vậy có thể coi kiến trúc mới đã giúp tiết kiệm khoảng 46 lần về năng lượng tiêu thụ (tính theo J/sample).

Ngoài ra các sự thay đổi trong kiến trúc cũng được thử nghiệm riêng lẻ trên thiết kế RANC ban đầu để quan sát những thông số một cách chính xác:

* Tăng kích thước bộ đệm của các Router: diện tích mạch tăng 42% so với ban đầu. Năng lượng tiêu thụ khi tổng hợp cũng tăng 47% về mặt số liệu, tuy nhiên con số này sẽ nhỏ hơn trên thực tế do đặc trưng của phần mềm tổng hợp.
* Tối giản Scheduler: tất cả các thông số về diện tích mạch, năng lượng tiêu thụ, số lượng cells đều được cải thiện khoảng 3 - 5 %.

Tổng kết lại, kiến trúc SNN Fully Parallel đã thật sự có sự đột phá trong tốc độ xử lý khi tăng gấp khoảng 86lần so với thiết kế RANC thông thường. Các đánh đổi về mặt diện tích mạch cũng như năng lượng là không quá lớn và hoàn toàn có thể chấp nhận được. Thậm chí nếu xét tiêu chí năng lượng dựa trên năng lượng tiêu thụ của mạch với mỗi mẫu ảnh xử lý được (tức J/sample), thì mạng SNN Fully Parallel lại thể hiện sự vượt trội về mặt tiết kiệm năng lượng.

# MÔ PHỎNG VÀ KIỂM THỬ KIẾN TRÚC SNN THỰC HIỆN NHẬN DIỆN CHỮ SỐ VIẾT TAY

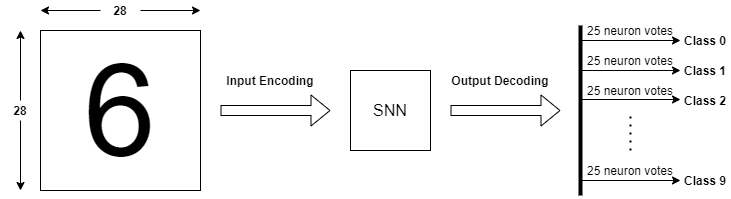
## Phương thức huấn luyện mạng SNN

Trình bày về Backpropagation

## Mô phỏng và kiểm thử sử dụng phần mềm Modelsim

### Phương thức sinh kích thích đầu vào cho mạng

Đầu vào của mạng được lấy từ 10.000 ảnh trong tập test của MNIST dataset. Kích thước mỗi ảnh là 28x28, thể hiện một số viết tay bất kì từ 0 đến 9 (10 class).

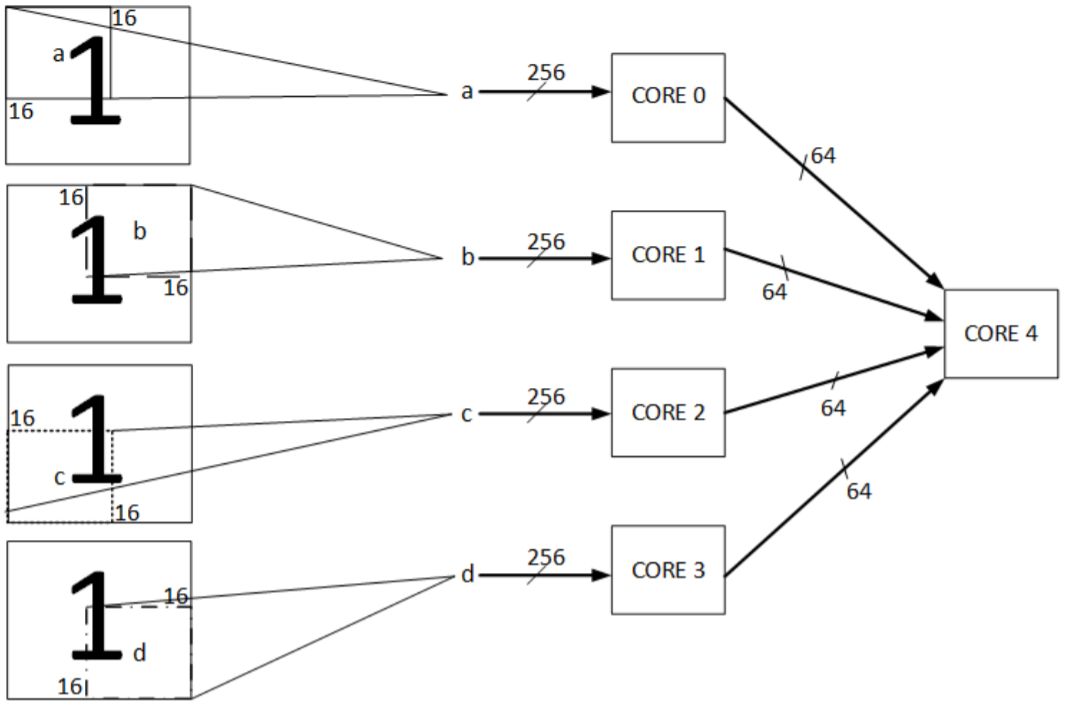


Hình 4.1 Kiểm thử chức năng mạng SNN

Các ảnh chữ số viết tay được mã hóa trở thành các spike như sau:

* Các pixels có giá trị lớn hơn hoặc bằng 126 được chuẩn hóa về giá trị 1.
* Các pixels có giá trị nhỏ hơn 126 được chuẩn hóa về giá trị 0.
* Những vị trí có giá trị 0 được coi là không có kích thích và bị bỏ qua.
* Những vị trí có giá trị 1 được mã hóa thành các gói kích thích có độ rộng 30 bits, gửi vào mạng SNN ngay khi tín hiệu tick được kích hoạt.
* Nội dung của các gói kích thích phụ thuộc vào số lượng Core đầu vào và việc chia ảnh theo các Core đầu vào này.

### Thực hiện mạng SNN gồm 5 Core đọc ảnh MNIST



Hình 4.2 Mô hình 5 Core đọc ảnh MNIST

Để mã hóa ảnh MNIST 28x28, trong bài sử dụng mô hình mạng SNN gồm 2 tầng, trong đó tầng đầu gồm 4 Core, tầng thứ hai gồm 1 Core:

* 4 Core đầu vào: mỗi Core gồm 256 Axons, mỗi Axons sẽ nhận kích thích ứng với một pixel trên ảnh. Như vậy một Core chỉ có thể nhận được tối đa 1 ảnh 16x16 pixels. Vì vậy để có thể đọc toàn bộ 1 ảnh 28x28, ta cần sử dụng 4 Core quét qua toàn bộ ảnh như trên Hình 4.2.
* Ví dụ: một ảnh được coi là mảng 2 chiều image[27:0][27:0]. Giả sử tại pixel image[0][3] (hàng 0/cột 3) có giá trị 1, gói kích thích được hình thành như sau:
* Xác định Core: dễ thấy pixel này thuộc Core 0 à giá trị 2 trường dx và dy đều là 9’b000000000.
* Xác định Axon: trong Core 0, pixel này ở vị trí có index là 3 à giá trị trường axon destination là 8’b00000011
* Tick instance: luôn gán giá trị 4’b0000.

Vậy ta thu được gói kích thích có độ rộng 30 bits (tham khảo 2.2.4):

30’b|000000000|000000000|00000011|0000|.

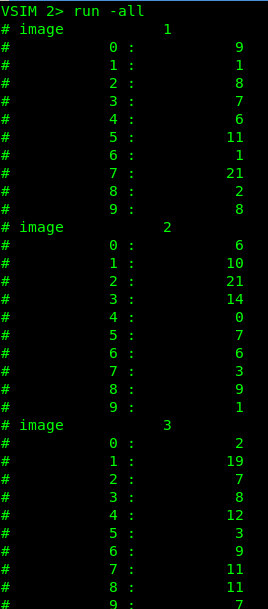
* Mỗi Core đầu vào nhận kích thích từ 256 Axons, sau đó tính toán và đưa kích thích ra từ 64 Neurons của mình. Ở đây mỗi Core tại tầng 1 chỉ sử dụng tới 64 Neurons để phóng kích thích thay vì dùng toàn bộ 256 Neurons. Lý do vì tầng 2 chỉ gồm 1 Core với 256 Axons đầu vào, nghĩa là tầng 2 có thể nhận tối đa 256 kích thích từ tầng 1 (64 Neurons x 4 Cores = 256).

(1 pixel có thể gửi gói tới nhiều hơn 1 Core đầu vào, nếu các Core này quét qua phần ảnh trùng lấn với nhau)

### Phương thức giải mã đầu ra từ các kích thích thu được

Tại Core đầu ra, ta cũng chỉ sử dụng 250 Neurons phóng kích thích ra ngoài mạng thay vì 256 Neurons, để phù hợp với số lượng class (10 class ứng với các số từ 0 đến 9, và 250 chia hết cho 10). Trong đó 25 Neurons đầu tiên sẽ vote cho số 0, 25 Neurons tiếp theo vote cho số 1,… tương tự cho các số (class) còn lại. Mỗi kích thích được phóng ra từ 1 Neuron đầu ra sẽ được tính là 1 vote.

Như vậy, sau khi hoàn thành xử lý cho một ảnh, đầu ra sẽ thu được các lượt votes ứng với các class, dựa trên số Neurons phóng kích thích. Mỗi class sẽ có tối đa 25 lượt votes (đạt giá trị này khi cả 25 Neurons ứng với class này đều phóng kích thích). Class nào có số lượt votes lớn nhất sẽ được chọn là class dự đoán của mạng tại tick đó.



Hình 4.3 Kết quả thu được sau khi giải mã đầu ra

Hình 4.3 mô tả kết quả của mạng SNN dưới dạng các votes dành cho mỗi class từ 0 đến 9. Dễ thấy 3 ảnh được dự đoán lần lượt là các số 7, 2, 1, và đây cũng chính là 3 ảnh đầu tiên trong tập MNIST.

## Nhận xét kết quả mô phỏng kiểm thử

Kết quả chạy mô phỏng cho 10.000 ảnh chữ số viết tay trong dataset MNIST được ghi ra file text. Với những ảnh xuất hiện nhiều hơn một class có cùng số lượt votes cao nhất, số nhỏ nhất sẽ được chọn làm kết quả dự đoán.

Sử dụng file output thu được để so sánh với nhãn của 10.000 ảnh này, kết quả thu được 9404 ảnh dự đoán đúng, tức độ chính xác 94.04% (đối với cả thiết kế RANC và Fully Parallel).

Như vậy, có thể kết luận:

* Mạng SNN với bộ tham số huấn luyện đã thực hiện chính xác chức năng nhận diện chữ số viết tay.
* Kiến trúc mới được đề xuất trong bài có chức năng hoàn toàn tương tự so với thiết kế RANC ban đầu.
* Chỉ với cấu trúc đơn giản gồm 5 Core chia làm 2 tầng, mạng đã có độ chính xác rất cao lên tới 94.04%. Từ đó thể hiện tiềm năng rất lớn của các kiến trúc mạng SNN khi tích hợp số lượng Core lớn hơn.

# TRIỂN KHAI KIẾN TRÚC SNN TRÊN BOARD XILINX ZYNQ ZC-702

## Các bước tiến hành

### Yêu cầu hệ thống

### Quy trình tổng quan triển khai hệ thống

### Thiết kế sơ đồ khối hệ thống

### Tổng hợp trên FPGA

### Kết nối các khối và tiến hành kiểm thử

## Nhận xét kết quả triển khai FPGA

# KẾT LUẬN

## Kết luận chung

## Hướng phát triển