ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Hồ Huy Hùng

THIẾT KẾ PHẦN CỨNG MẠNG NƠ-RON NHÂN TẠO SỬ DỤNG KỸ THUẬT TÍNH TOÁN NGẪU NHIÊN, ỨNG DỤNG NHẬN DẠNG CHỮ SỐ VIẾT TAY

KHOÁ LUẬN TỐT NGHIỆP HỆ ĐẠI HỌC CHÍNH QUY

Ngành: Công nghệ kĩ thuật điện tử, truyền thông

HÀ NỘI - 2018

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Hồ Huy Hùng

THIẾT KẾ PHẦN CỨNG MẠNG NƠ-RON NHÂN TẠO SỬ DỤNG KỸ THUẬT TÍNH TOÁN NGẪU NHIÊN, ỨNG DỤNG NHẬN DẠNG CHỮ SỐ VIẾT TAY

KHOÁ LUẬN TỐT NGHIỆP HỆ ĐẠI HỌC CHÍNH QUY

Ngành: Công nghệ kĩ thuật điện tử, truyền thông

Cán bộ hướng dẫn: PGS. TS. Trần Xuân Tú

HÀ NỘI - 2018

LỜI CẢM ƠN

Lời đầu tiên, em xin bày tỏ lòng biết ơn tới thầy giáo, Phó giáo sư, Tiến sĩ Trần Xuân Tú, người đã quan tâm, chỉ bảo tận tình và động viên em rất nhiều. Thầy đã cho em nhiều cơ hội để em rèn luyện và trưởng thành hơn.

Em xin chân thành cảm ơn tới tất cả thành viên của Phòng thí nghiệm trọng điểm Hệ thống tích hợp thông minh (SISLAB), tại đây em đã được giải đáp nhiều thắc mắc và học được nhiều kinh nghiệm quý báu.

Em xin cảm ơn đại gia đình khoa Điện tử, Viễn thông, Trường Đại học Công nghệ, nơi đã cho em một môi trường học tập, tích lũy kinh nghiệm tuyệt vời.

Cuối cùng, em chân thành biết ơn gia đình và bạn bè, những người đã luôn quan tâm, chăm sóc và chia sẻ mọi buồn vui.

Em xin chân thành cảm ơn!

TÓM TẮT

Tóm tắt: Mạng nơ-ron nhân tạo (ANN), một mô hình tính toán lấy cảm hứng từ sinh học, đã và đang chứng tỏ sự hiệu quả trong nhiều ứng dụng thực tế. Thực thi phần cứng mạng nơ-ron nhân tạo có thể giải quyết được nhiều vấn đề cố hữu của phần mềm như tốc độ, chi phí hay công suất tiêu thụ. Tuy nhiên, thực thi phần cứng mạng nơ-ron nhận tạo trên các hệ thống IoT còn gặp nhiều khó khăn với yêu cầu thiết kế hệ thống kích thước lớn tích hợp trong một thiết bị IoT nhỏ gọn. Một kỹ thuật bắt nguồn từ lý thuyết xác suất, kỹ thuật *tính toán ngẫu nhiên* (SC), là một giải pháp triển vọng giúp giảm độ phức tạp và chi phí của các phép toán trong phần cứng. Trong khóa luận nghiên cứu về tổng quan mạng nơ-ron nhân tạo, ứng dụng của nó và trình bày khái quát kỹ thuật tính toán ngẫu nhiên. Khóa luận đề xuất một mô hình mạng nơ-ron nhân tạo chỉ sử dụng một khối nơ-ron duy nhất có khả năng tính toán song song nhiều đầu vào với mục tiêu giảm tối thiểu chi phí phần cứng nhưng vẫn tận dụng khả năng xử lý song song. Sau đó, khóa luận đề xuất kiến trúc khối nơ-ron sử dụng kỹ thuật tính toán ngẫu nhiên, kết quả mô phỏng cho thấy nơ-ron này giảm độ chính xác xấp xỉ so với tính toán nhị phân thông thường trong khi giảm LUTs và có công suất tiêu thụ thấp hơn. Để kiểm tra hiệu năng của mạng, kiến trúc mạng nơ-ron đề xuất được ứng dụng vào hệ thống nhận dạng chữ số viết tay sử dụng thư viện MNIST với kết quả nhận dạng lên đến 92,18%. Kết quả tổng hợp trên FPGA cho thấy mạng nơ-ron có thời gian tính toán là $30{,}93\mu s$, sử dụng 1659 LUTs và 1020 thanh ghi. Với chi phí phần cứng thấp, mạng nơ-ron đề xuất hoàn toàn có thể mở rộng hoặc nhúng vào các thiết bị IoT.

Từ khóa: *Mạng nơ-ron nhân tạo, tính toán ngẫu nhiên, nhận dạng chữ số viết tay.*

LỜI CAM ĐOAN

Tôi xin cam đoan toàn bộ khóa luận tốt nghiệp là kết quả của quá trình nghiên cứu và thực hiện của cá nhân tôi, không sao chép bất cứ nội dung văn bản của tác giả nào. Các tài liệu tham khảo được liệt kê đầy đủ và trích dẫn rõ ràng.

Hà Nội, ngày 03 tháng 05 năm 2018

Sinh viên thực hiện

Hồ Huy Hùng

MỤC LỤC

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG BIỂU

DANH SÁCH KÝ TỰ VIẾT TẮT

Chữ viết tay	Ý nghĩa - tiếng Anh	Ý nghĩa - tiếng Việt (nếu có)
ANN	Artificial Neural Network	Mạng nơ-ron nhân tạo
FPGA	Field-Programmable Gate Array	Mạng logic có thể lập trình tại chỗ
IoT	Internet of Things	Internet vạn vật
MLP	Multi-layer perceptron	Perceptron đa lớp
DNN	Deep Neural Network	Mạng nơ-ron sâu
VHDL	VHSIC Hardware Description Language	Ngôn ngữ mô tả phần cứng VHSIC
VHSIC	Very High Speed Integrated Circuit	Mạch tích hợp tốc độ rất cao
SC	Stochastic Computing	Tính toán ngẫu nhiên
SN	Stochastic Number	Số ngẫu nhiên
SNG	Stochastic Number Generation	Bộ khởi tạo số ngẫu nhiên
SVM	Support Vector Machine	Máy véc-tơ hỗ trợ

MỞ ĐẦU

Mạng nơ-ron nhân tạo (ANN) là một mô hình tính toán dùng trong nhận dạng, được sử dụng cho học máy (machine learning) và là nền tảng của trí tuệ nhân tạo (AI). Mạng nơ-ron nhân tạo được lấy cảm hứng từ sinh học, bắt chước hành vi của mạng nơ-ron sinh học trong não con người. Với khả năng nhận diện các mô hình phức tạp, ANN đã chứng minh sự hiệu quả trong nhiều vấn đề thực tế như nhận dạng ảnh, chữ viết, giọng nói, xử lý ngôn ngữ tự nhiên, xử lý tín hiệu, phân loại dữ liệu…

Hầu hết các ứng dụng ANN hiện nay được viết bằng các ngôn ngữ phần mềm như Python, Matlab, C++... và sử dụng CPU, GPU để thực thi. Tuy nhiên, nhiều ứng dụng đặc biệt như nén video trực tuyến hay nhận dạng đối tượng trên camera cần một mạng nơ-ron kích thước lớn để đào tạo hay nhận dạng, nhưng lại yêu cầu khả năng xử lý thời gian thực và sử dụng hiệu quả năng lượng. Khi đó các kiến trúc ANN chuyên dụng có những lợi thế đáng kể nhờ khả năng xử lý song song.

- **Tốc độ:** Thực thi phần cứng có thể đạt được tốc độ tính toán rất cao nhờ khả năng xử lý song song và không bị hạn chế bởi đường truyền dữ liệu bus, thực thi phần cứng ANN còn ưu thế hơn bởi sự tương đồng với các nơ-ron sinh học xử lý song song. Ví dụ, hệ thống nhận dạng người sử dụng thuật toán HOG [1] thực thi phần cứng trên FPGA có tốc độ khung hình trên giây lên đến 272, phù hợp được các ứng dụng xử lý video tốc độ cao, trong khi thực thi trên CPU chưa đáp ứng yêu cầu.
- **Chi phí:** Thực thi phần cứng ANN giảm nhiều chi phí phần cứng nhờ sử dụng hiệu quả nguồn tài nguyên và không sử dụng CPU để tính toán. Các nền tảng FPGA hiện nay cung cấp hiệu suất tuyệt vời ở mức năng lượng thấp, nhưng cũng có khả năng thay đổi kiến trúc để linh hoạt hơn với yêu cầu của ứng dụng.

Hiện nay có nhiều hệ thống phần cứng mạng nơ-ron được ứng dụng trong thực tế, chúng thường được áp dụng cho học máy, xử lý video và nhiều ứng dụng chuyên biệt khác. Mặc dù vậy, trong kỷ nguyên Internet vạn vật (IoT), thiết kế hệ thống ANN nhỏ gọn tích hợp vào các thiết bị IoT vẫn còn gặp nhiều khó khăn. Thách thức của các thiết bị IoT là chi phí phần cứng thấp, công suất thấp và chu kỳ sống. Trong đó, giảm chi phí phần cứng và hiệu quả công suất là điều thiết yếu để tạo ra các hệ thống ANN IoT nhỏ gọn tích hợp vào cuộc sống.

Một trong những phương án giảm chi phí phần cứng và công suất thấp đang được nghiên cứu hiện nay là kỹ thuật *Tính toán ngẫu nhiên* (SC). Tính toán ngẫu nhiên là một kỹ thuật xử lý dữ liệu dưới hình thức xác xuất số hóa, lần đầu tiên được giới thiệu từ những năm 1960s bởi Gaines [2]–[4]. Kỹ thuật này sử dụng các phép logic cơ bản có độ phức tạp thấp để thay thế cho các phép tính nhị phân. Vì vậy, nó có thể giảm nhiều chi phí phần cứng và tăng tốc độ so với mạch nhị phân thông thường, tuy nhiên điều đó cũng đi liền với độ chính xác bị suy giảm khi tính toán trên miền xác suất ngẫu nhiên. Tuy nhiên, trong những năm gần đây, SC đã trở lại như một sự lựa chọn cho các ứng dụng đòi hỏi phần cứng nhỏ, hiệu quả năng lượng và độ chính xác ở mức chấp nhận được.

Vì vậy, trong khóa luận này thiết kế và thực thi một mô hình mạng nơ-ron nhân tạo sử dụng kỹ thuật tính toán ngẫu nhiên với chi phí phần cứng và công suất thấp, từ đó có tiềm năng tích hợp mạng vào các hệ thống IoT. Những đóng góp chính của công trình này là:

- Khóa luận trình bày các vấn đề cơ bản của mạng nơ-ron nhân tạo và kỹ thuật tính toán ngẫu nhiên.
- Thiết kế một mô hình phần cứng mạng nơ-ron nhân tạo chỉ sử dụng một nơ-ron duy nhất tính toán nhiều đầu vào song song và có khả năng sử dụng kỹ thuật tính toán ngẫu nhiên.
- Thực thi phần cứng thiết kế đề xuất trên công nghệ FPGA, ứng dụng trong nhận dạng chữ số viết tay.

Kiến trúc mô-đun nơ-ron sử dụng kỹ thuật tính toán ngẫu nhiên trên kiến trúc ANN 2-3-2 đã tham gia cuộc thi Thiết kế vi mạch tích hợp cỡ lớn (LSI Design Contest 2018) tại Okinawa (Nhật Bản) và đã giành vị trí thứ hai chung cuộc.

Các phần còn lại của khóa luận được trình bày như sau. Chương 1 giới thiệu khái quát về mạng nơ-ron nhân tạo và kỹ thuật tính toán ngẫu nhiên. Chương 2 trình bày tổng quan, kiến trúc phần cứng của thiết kế đề xuất. Chương 3 trình bày phương pháp đánh giá, kết quả mô phỏng và thực thi trên FPGA. Phần cuối cùng là kết luận.

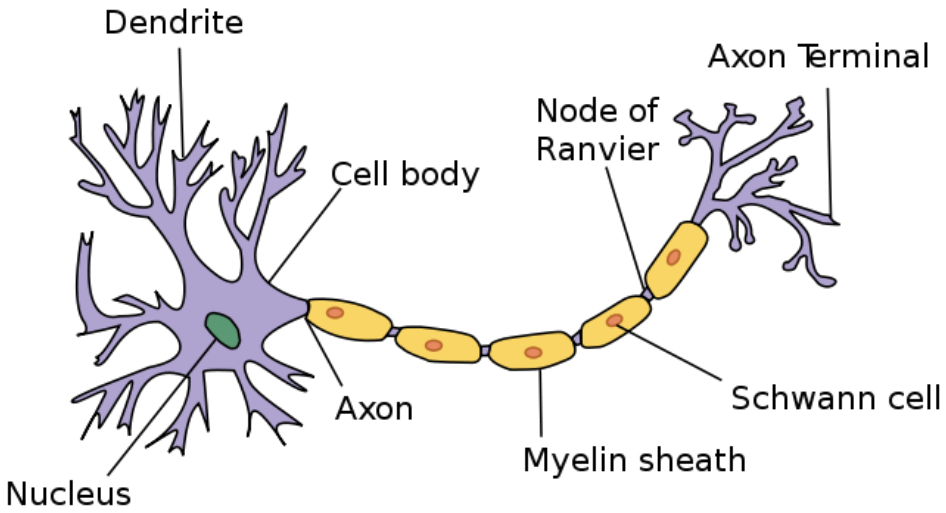
CHƯƠNG 1 KHÁI QUÁT VỀ MẠNG NƠ-RON NHÂN TẠO VÀ KỸ THUẬT TÍNH TOÁN NGẪU NHIÊN

1.1 Giới thiệu về mạng nơ-ron nhân tạo

Lấy cảm hứng từ nơ-ron sinh học của bộ não, một mạng nơ-ron nhân tạo (ANN) được tạo ra bằng cách sử dụng các phép toán mô phỏng hành vi của mạng nơ-ron sinh học. Áp dụng các thuật toán bắt chước này, chúng ta có thể làm cho mạng "học" các khuôn mẫu để giải quyết nhiều vấn đề thực tế. Phần này sẽ giới thiệu khái quát từ nơ-ron sinh học đến mạng nơ-ron nhân tạo.

1.1.1 Nơ-ron sinh học

Một mạng nơ-ron sinh học gồm các nơ-ron sinh học kết nối với nhau. Nơ-ron nhân tạo được thiết kế để mô hình hóa toán học các chức năng của nơ-ron sinh học. Mặc dù có ít nhất 5 loại nơ-ron sinh học khác nhau, nhưng trong phần này chỉ giới thiệu một loại nơ-ron sinh học cơ bản nhất, như Hình 1.1.



Hình 1.1: Cấu trúc một nơ-ron sinh học. *Hình ảnh được lấy từ Wikipedia* [5], *với giấy phép CC BY-SA 3.0.*

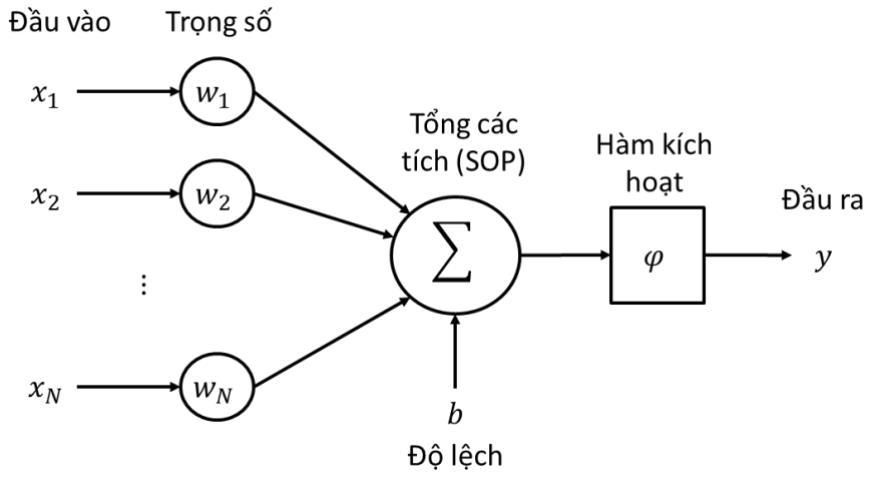
Cấu trúc một nơ-ron sinh học sở hữu một số lượng lớn các nhánh, được gọi là *sợi nhánh* (dendrites), nhận và truyền tín hiệu vào các tế bào. Tại đây các tín hiệu được tích lũy, khi tín hiệu vượt quá một ngưỡng có sẵn, các nơ-ron phát một kích thích điện truyền qua *sợi trục* (axon). Sợi trục liên kết với sợi nhánh của các nơ-ron lân cận thông

qua một vùng tiếp xúc gọi là *xi-náp* (synapses). Các nơ-ron được liên kết với nhau qua các xi-náp này theo một cường độ nhất định (có thể gọi là trọng số).

Quá trình xử lý trong não gồm song song và phân tán: thông tin được lưu trong các kết nối (chủ yếu là trong các sợi trục), phân phối trong mạng nơ-ron và được xử lý với một số lượng lớn các nơ-ron song song. Bộ não có khả năng học tập thích nghi, học hỏi các khuôn mẫu từ lúc sinh ra cho đến khi chết đi. Mạng nơ-ron có khả năng tìm ra các quy tắc mô tả tập dữ liệu huấn luyện và từ những thông tin đã học trước, phản ứng chính xác với các dữ liệu mới.

1.1.2 Nơ-ron nhân tạo

Vào năm 1943, McCulloch và Pitts [6] đã mô hình hóa toán học một nơ-ron như một hàm chuyển đổi từ đầu vào và các trọng số của chúng thành đầu ra kích hoạt hay không kích hoạt. Các nhà khoa học đã chỉ ra rằng mạng lưới các nơ-ron mô hình như vậy có đặc tính tương tự như não người: chúng có thể thực hiện việc nhận dạng khuôn mẫu.



Hình 1.2: Mô hình một nơ-ron nhân tạo.

Một nơ-ron nhân tạo (perceptron) là đơn vị cơ bản để tạo thành mạng nơ-ron nhân tạo. Các yếu tố cơ bản của một nơ-ron là:

(1) một tập hợp các nút đầu vào nhận các tín hiệu đầu vào tương ứng $X = (x_1, x_2, \dots, x_N)^T$;

(2) một tập hợp các kết nối tương tự xi-náp có cường độ được biểu diễn bằng một tập trọng số $W = (w_1, w_2, \dots, w_N)^T$;
(3) độ lệch b;
(4) một hàm kích hoạt φ liên quan đến tổng các đầu vào với các trọng số của chúng cho kết quả đầu ra (kích hoạt hay không kích hoạt) của nơ-ron.

Các thành phần trên được minh họa trong Hình 1.2.

Tổng các đầu vào với các trọng số của chúng được biểu diễn bởi công thức:

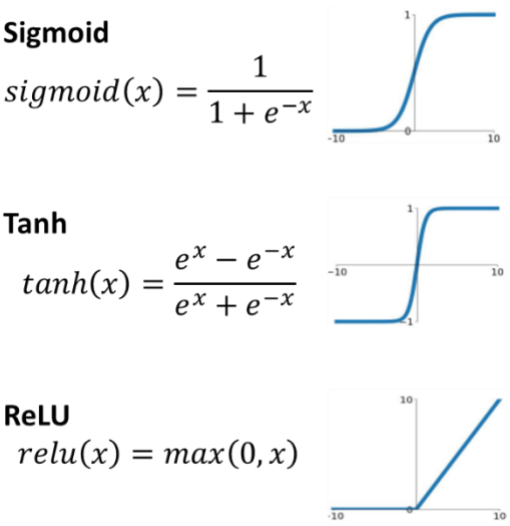
$$u = \sum_{i=1}^{N} x_i \times w_i + b \tag{1.1}$$

Kích hoạt đầu ra, y, sau đó được đưa ra bởi:

$$y = \varphi(u) \tag{1.2}$$

với φ là hàm kích hoạt của nơ-ron.

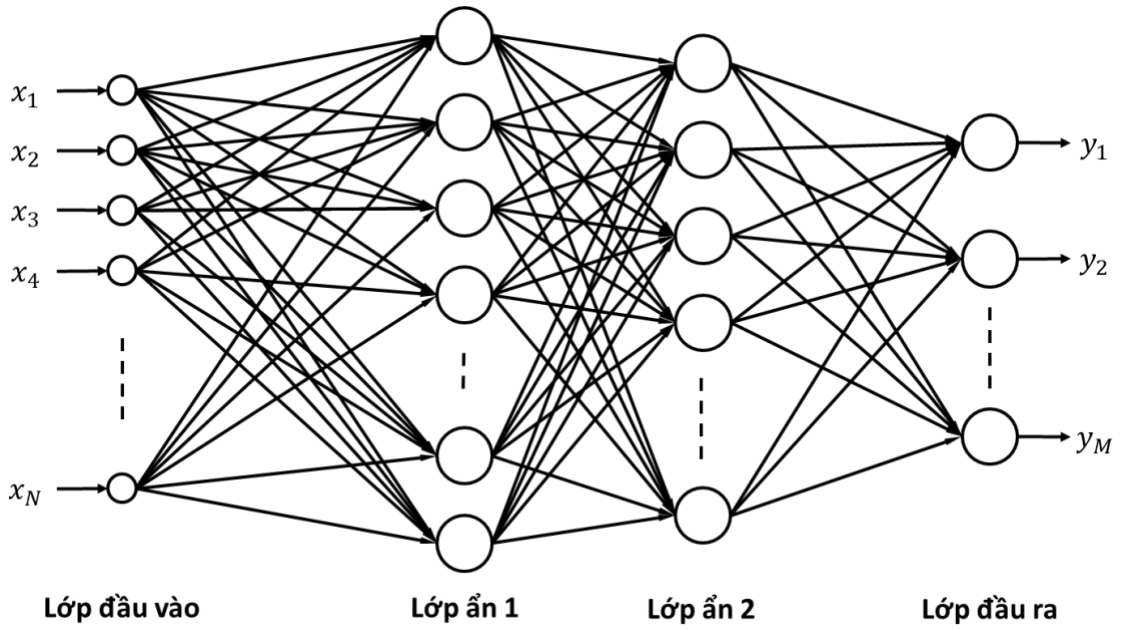
Tổng các đầu vào có trọng số được chuyển đổi đầu ra thông qua một hàm kích hoạt phi tuyến φ. Một số hàm kích hoạt phổ biến được trình bày trong Hình 1.3.



Hình 1.3: Một số hàm kích hoạt phổ biến.

1.1.3 Mạng nơ-ron

Trong khóa luận này tập trung vào một mô hình mạng nơ-ron quan trọng - *Perceptron đa lớp* (MLP), vì nó là cấu trúc ANN được sử dụng nhiều nhất cũng như tiện ích nhất. Perceptron đa lớp là một mô hình mạng nơ-ron nhân tạo truyền thẳng, từ tập hợp các dữ liệu đầu vào truyền qua mạng và nhận được tập hợp các đầu ra phù hợp. Nó là một biến thể của perceptron tuyến tính cổ điển với ba hoặc nhiều lớp nơ-ron và sử dụng hàm kích hoạt phi tuyến. Thông thường, như thể hiện trong Hình 1.4, một MLP tiêu chuẩn bao gồm một lớp đầu vào của các nút, tiếp theo là một hoặc nhiều lớp perceptron được gọi là lớp ẩn và lớp cuối cùng là lớp đầu ra. MLP mạnh hơn so với perceptron vì perceptron chỉ có thể phân tách dữ liệu dạng tuyến tính, còn MLP có thể phân tách dữ liệu dạng không tuyến tính bằng một siêu phẳng [7]. Hiện nay, MLP được sử dụng nhiều nhất là MLP ba lớp (chỉ sử dụng một lớp ẩn) do có kiến trúc đơn giản nhất nhưng vẫn có khả năng nhận diện các khuôn mẫu phức tạp.



Hình 1.4: Kiến trúc mạng nơ-ron MLP.

Đào tạo: Để mạng nơ-ron có thể hoạt động, cần có tập dữ liệu cần nhận dạng và một bộ trọng số tương ứng với các yêu cầu nhận dạng cụ thể. Để có được các trọng số này, cần phải đào tạo mạng nơ-ron: dữ liệu huấn luyện được đưa vào mạng, đầu ra sẽ được đánh giá dựa trên đầu ra kỳ vọng, sau đó trọng số sẽ được điều chỉnh tương ứng cho phù hợp với đầu ra; qua quá trình đào rất nhiều lần, đầu ra sẽ gần với giá trị đầu ra kỳ vọng, khi đó các trọng số có thể sử dụng để nhận dạng. Một trong những phương

pháp đào tạo phổ biến nhất cho perceptron đa lớp là sử dụng thuật toán *Lan truyền ngược* [8]. Quá trình đào tạo mạng có thể thực hiện ngoại tuyến trước, sau đó truyền trọng số vào mạng nơ-ron để thực hiện các ứng dụng nhận dạng mẫu.

1.1.4 Ứng dụng

Mạng nơ-ron nhân tạo đã được sử dụng rộng rãi để mô hình hóa các mối quan hệ phức tạp giữa đầu vào và đầu ra trong nhiều lĩnh vực, ví dụ như chức năng xấp xỉ, phân loại (nhận dạng mẫu và ra quyết định), rô-bốt, nhận dạng và kiểm soát hệ thống, chẩn đoán y tế, khai thác dữ liệu và các ứng dụng tài chính.

Mạng nơ-ron hiện đang là nền tảng cho nhiều ứng dụng trí tuệ nhân tạo hiện đại. Mạng nơ-ron nhân tạo có đột phá đáng kể trong nhận dạng giọng nói [9], được sử dụng trong các sản phẩm của Microsoft. Các mạng nơ-ron sâu (DNN) được ứng dụng trong trong nhiều lĩnh vực tiên tiến từ xe tự lái [10], phát hiện ung thư [11] đến trò chơi cờ vây phức tạp [12].

Năm 2012, trong cuộc thi nhận dạng ảnh trên quy mô lớn (ILSVRC - the ImageNet Large-Scale Visual Recognition Changllege), một nhóm từ đại học Torondo ở Canada đã sử dụng một mạng nơ-ron tên là AlexNet [13] kết hợp thuật toán SuperVision để phân loại 1,2 triệu bức ảnh chất lượng cao thành 1000 lớp khác nhau và đạt được tỷ lệ sai lỗi chỉ 16,2% - vượt xa so với thành tích cao nhất của năm trước đó là 25%. Đó được xem là sự bắt đầu của cuộc cách mạng học máy. Và cho đến nay, kiến trúc mạng nơ-ron ResNet [14] đang có sai số độ chính xác thấp nhất, 3,57%, đánh bại khả năng nhận dạng của con người trong tập dữ liệu ImageNet này.

1.2 Tính toán ngẫu nhiên

Tính toán ngẫu nhiên (SC - Stochastic Computing) là một phương pháp tính toán dựa vào nguyên lý xác suất, SC biểu diễn dữ liệu bởi các chuỗi bit giả ngẫu nhiên (pseudo-random), được gọi là các *số ngẫu nhiên* (SNs). Nó có khả năng thay thế các hàm toán học bằng các mạch logic có kích thước phần cứng nhỏ, công suất thấp và có khả năng chống lỗi. Những đặc điểm này làm cho SC có khả năng áp dụng vào thực tiễn, nơi các ứng dụng có khối lượng tính toán lớn cần xử lý song song, hoặc các thiết bị hoạt động ở môi trường nhiều tạp âm, nơi các tính toán nhị phân thông thường có khả năng gây lỗi. Mạng nơ-ron nhân tạo có tiềm năng áp dụng kỹ thuật tính toán ngẫu

nhiên, do yêu cầu tính toán song song của ANN là rất lớn và các sai số ngẫu nhiên của SC trong một số trường hợp có thể chấp nhận được. Trong phần này sẽ trình bày lý thuyết cơ bản và kiến trúc của các thành phần tính toán ngẫu nhiên.

1.2.1 Giới thiệu

Như đã nhắc ở trên, SC hoạt động trên các chuỗi bit ngẫu nhiên sử dụng các mạch logic thông thường, được gọi là các *mạch ngẫu nhiên*. Nếu X là một chuỗi bit ngẫu nhiên có độ dài K, X được gọi là một *số ngẫu nhiên (SN)* với giá trị biểu diễn là xác suất bit 1 (p_x) trong X. Tức là, nếu X gồm K1 bit 1, p_x được xấp xỉ bởi giá trị p'_x = K1/K. Thông thường $p'_x \approx p_x$, độ chính xác của phép xấp xỉ này tăng tỉ lệ thuận với độ dài K. Ví dụ, với K là 4, 8 và 16, các chuỗi bit 0101, 01011010 và 1010011001010011 đều là các trường hợp biểu diễn của $p_x = 1/2$.

SC có thể biểu diễn xấp xỉ các một số thực x tùy ý nằm trong khoảng (0; 1), được gọi là dạng *đơn cực* với:

$$x = P(X = 1) = p_x \tag{1.3}$$

Để biểu diễn các số có dấu trong khoảng (-1; 1), ta dùng dạng *lưỡng cực* với:

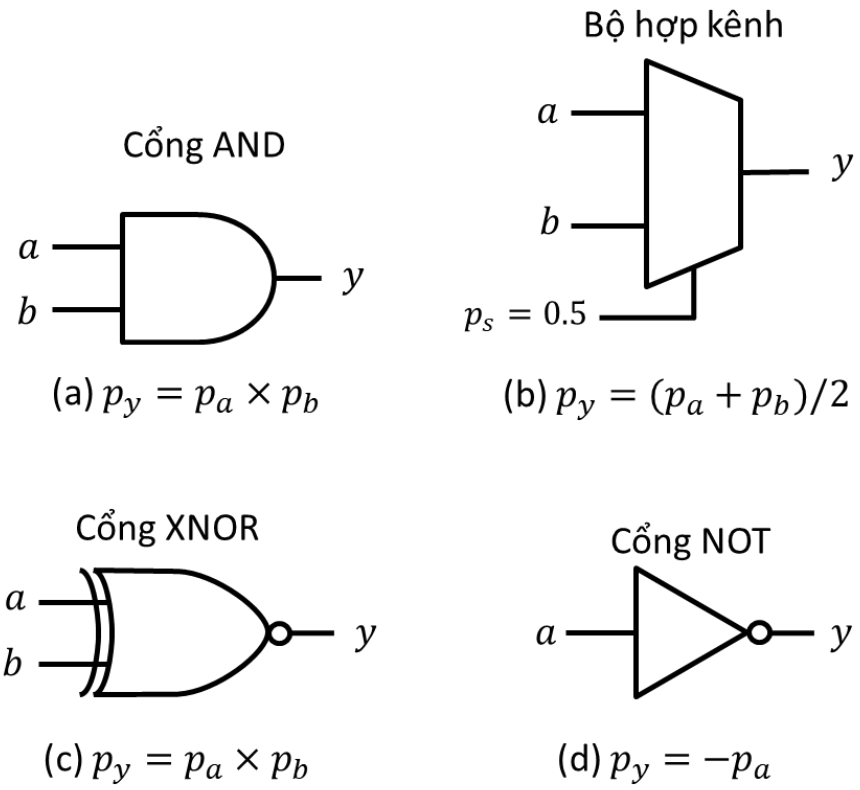
$$x = 2P(X = 1) - 1 = 2p_x - 1 \tag{1.4}$$

Bảng 1.1 tóm tắt các giá trị đơn cực và lưỡng cực của số SN với sự khác nhau của độ dài K của chuỗi bit. Ví dụ, một chuỗi bit 000...001 với duy nhất một bit 1 trong chuỗi, biểu diễn giá trị 1/K dạng đơn cực và 2/K-1 dạng lượng cực. Bảng 1.1 cũng cho thấy SC có nhiều định dạng mã hóa đại diện cho cùng một giá trị, minh họa cho điều này là các chuỗi bit 000...001, 000...010, …, và 100...000 cùng biểu diễn một giá trị duy nhất.

Bảng 1.1: Giá trị biểu diễn của một chuỗi SN K-bit X trong dạng đơn cực và lưỡng cực.

Chuỗi bit X (K bit)	**Số bit 1 trong X**	**Giá trị nguyên biểu diễn**		**Số chuỗi bit biểu diễn cùng giá trị**
		Đơn cực p_x	**Lưỡng cực $2p_x - 1$**	
$000 \dots 000$	0	0	-1	1
$000 \dots 001$ $000 \dots 010$ $\vdots$ $100 \dots 000$	1	$\frac{1}{K}$	$\frac{2}{K} - 1$	K
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$000 \dots 111$ $\vdots$ $111 \dots 000$	$\frac{K}{2}$	$\frac{1}{2}$	0	$\frac{K(K-1) \dots (K/2+1)}{(K/2)(K/2-1) \dots 1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$111 \dots 111$	K	1	1	1

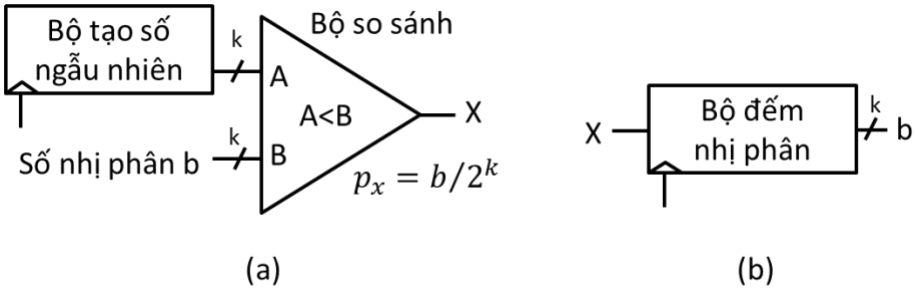
Các phép logic đơn giản áp dụng trong các chuỗi bit SC có thể thực hiện các phép tính số học trên miền xác suất của chúng. Hình 1.5 biểu diễn các bộ tính toán ngẫu nhiên cơ bản dưới dạng đơn cực và lưỡng cực.



Hình 1.5: Các thành phần cơ bản của SC: (a) bộ nhân đơn cực, (b) bộ cộng tỷ lệ đơn cực và lưỡng cực, (c), bộ nhân lưỡng cực, (d) bộ đảo lưỡng cực.

Một cổng AND hai đầu vào thực thi một phép nhân đơn cực $p_x \times p_y$ của hai chuỗi bit A và B trong N chu kỳ đồng hồ. Bộ cộng được thực thi bởi bộ hợp kênh (MUX) với hai đầu vào tính toán và đầu vào lựa chọn có xác suất $p_s = 0{,}5$, kết quả đầu ra là tổng đầu vào tỷ lệ $0{,}5 \times (p_x + p_y)$, sự tỷ lệ nhằm đảm bảo đầu ra luôn nằm trong khoảng xác suất (0; 1). Lưu ý rằng độ chính xác của phép tính phụ thuộc vào sự độc lập (không tương quan) của chuỗi bit đầu vào A và B, các vấn đề này sẽ được thảo luận chi tiết vào phần tiếp theo.

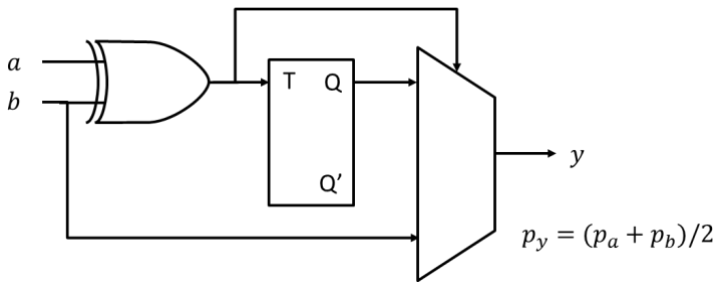
Trong dạng lưỡng cực, một cổng XNOR hai đầu vào thực thi một phép nhân (Hình 1.5c), trong khi bộ MUX tiếp tục được sử dụng như một bộ cộng tỷ lệ. Bộ đảo dấu trong dạng lưỡng cực miền SC có thể thực thi bằng một cổng NOT như Hình 1.5d.



Hình 1.6: Mạch chuyển đổi số: (a) số nhị phân sang SN, (b) SN sang số nhị phân.

Các mạch chuyển đổi số là cần thiết cho bất kì giao tiếp nào giữa mạch nhị phân thông thường và mạch ngẫu nhiên. Hình 1.6a biểu diễn một *bộ tạo số ngẫu nhiên* (SNG) sẽ tạo ra một chuỗi SN có độ dài 2^k-bit biểu diễn giá trị $p_x = b/2^k$ từ số nhị phân k-bit biểu diễn số nguyên b. Một SNG gồm một bộ khởi tạo số ngẫu nhiên (thường sử dụng *Thanh ghi dịch hồi tiếp tuyến tính* - LFSR) và một bộ so sánh dùng để so sánh giá trị của số nguyên b với số ngẫu nhiên và tạo ra bit 1 nếu b lớn hơn số ngẫu nhiên hoặc bit 0 nếu ngược lại. Một bộ đếm nhị phân đơn giản (Hình 1.6b) chuyển một số SN sang số nhị phân thông thường bằng cách đếm số bit 1 trong chuỗi SN.

Bên cạnh đó, bộ cộng SC cũng có thể được thực hiện bằng một bộ cộng tỷ lệ khác với độ chính xác cao hơn và không yêu cầu thêm đầu vào ngẫu nhiên biểu diễn giá trị 0,5 như bộ cộng thông thường ở Hình 1.5b. Bộ cộng này được đề xuất bởi Vincent T.Lee [15] và sử dụng một bộ MUX, một bộ XOR và một mạch lật T (TFF) như Hình 1.7.



Hình 1.7: Bộ cộng tỷ lệ SC độ chính xác cao [15].

Chi phí phần cứng của mạch lật T và các cổng XNOR, MUX không lớn hơn bộ khởi tạo số ngẫu nhiên để tạo ra đầu vào 0,5. Quan trọng hơn, chuỗi bit tạo ra bởi TFF có độ độc lập cao so với các chuỗi bit đầu vào. Điều này có nghĩa là không có sự ràng buộc tương quan giữa hai đầu vào mà chúng ta vẫn có một kết quả phép cộng có độ

chính xác cao (sẽ so sánh chi tiết trong phần sau). Vì vậy trong khóa luận này sẽ sử dụng bộ cộng tỷ lệ này thay cho bộ cộng SC thông thường.

1.2.2 Cơ sở toán học

Một số ngẫu nhiên X là chuỗi bit có giá trị xác suất xuất hiện bit 1 là p_x, khi đó có thể coi X là một biến ngẫu nhiên Bernoulli với tham số p_x [16]. Số ngẫu nhiên X biểu diễn giá trị số thực là p_x ở dạng đơn cực và biểu diễn giá trị số thực $2p_x–1$ ở dạng lưỡng cực.

Xét mạch ngẫu nhiên cổng AND thực hiện phép tính nhân ở dạng đơn cực. Theo lý thuyết xác suất, nếu A và B là hai biến ngẫu nhiên độc lập (không tương quan) thì chuỗi đầu ra Y biểu diễn giá trị là:

$$y = p_a \times p_b = a \times b \tag{1.5}$$

Trong trường hợp hai chuỗi A, B ở dạng lưỡng cực, một cổng XNOR có khả năng thực hiện bộ nhân. Cổng XNOR sẽ nhận giá trị 1 nếu hai đầu vào đồng thời là 0 hoặc đồng thời là 1, khi đó:

$$\begin{aligned} p_y &= p_a \times p_b + p'_a \times p'_b \\ &= 1 - p_a - p_b + 2 \times p_a \times p_b \end{aligned} \tag{1.6}$$

Từ $p_a = \frac{a+1}{2}$, $p_b = \frac{b+1}{2}$, ta có:

$$\begin{aligned} p_y &= 1 - \frac{a+1}{2} - \frac{b+1}{2} + 2\frac{a+1}{2}\frac{b+1}{2} \\ &= \frac{a \times b + 1}{2} \end{aligned} \tag{1.7}$$

Khi đó:

$$y = 2p_y - 1 = a \times b \quad (1.8)$$

Tương tự, đối với bộ cộng tỷ lệ sử dụng bộ hợp kênh như hình Hình 1.5b, đầu ra kỳ vọng khi các đầu vào không tương quan là:

$$y = p_a \times p_s + p_b \times (1 - p_s) \quad (1.9)$$

Nếu đầu vào tín hiệu chọn S là $p_s = 0.5$ thì giá trị kỳ vọng của Y sẽ bằng tổng tỷ lệ của tổng 2 đầu vào

$$y = 0.5 \times p_a + 0.5 \times p_b = \frac{a+b}{2} \quad (1.10)$$

Ngoài cổng XNOR, MUX hay NOT đã biểu diễn trong Hình 1.5, các cổng logic khác cũng có khả năng thực hiện các phép toán khác nhau trên miền SC như Bảng 1.2.

Bảng 1.2: Thực thi các cổng logic trên miền SC [17].

Cổng logic Đầu vào: X1, X2 và X3 (nếu có)	Thực thi trên SC (các đầu vào độc lập)
AND	p_1p_2
AND (X1 đảo)	$p_2(1-p_1)$
NAND	$1-p_1p_2$
OR	$p_1+p_2-p_1p_2$
OR (X1 đảo)	$p_2+(1-p_1)-p_2(1-p_1)$
NOR	$(1-p_1)(1-p_2)$
XOR	$p_1+p_2-2p_1p_2$
XOR (X1 đảo)	$1-p_1-p_2+2p_1p_2$
XNOR	$1-p_1-p_2+2p_1p_2$
MUX (tín hiệu chọn X3)	$p_1-p_1p_3+p_2p_3$
MUX(tín hiệu chọn X3, X1 đảo)	$1-p_1-p_1p_3+p_1p_3+p_2p_3$

1.2.3 Ưu và nhược điểm

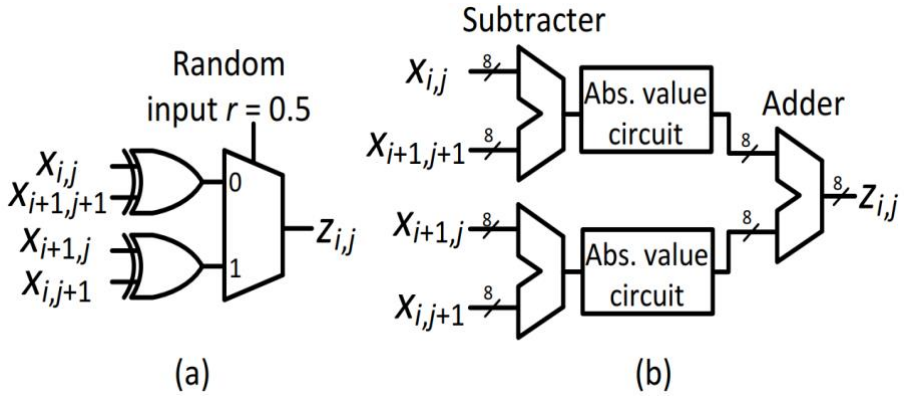
Tính toán ngẫu nhiên có hai ưu điểm chính so với tính toán thông thường:

- **Thay thế các phép tính thông thường bằng các mạch logic đơn giản:** Giả sử có hai số thực được biểu diễn xấp xỉ bởi hai số nhị phân k bit. Sử dụng phương pháp nhân nhị phân thông thường sẽ cần k^2 bộ cộng nhị phân, còn với SC chỉ sử dụng một cổng logic AND (hoặc XNOR) cho hai lối vào và đầu ra sẽ là kết quả mong đợi. Hơn nữa, SC chỉ yêu cầu đầu vào, đầu ra là hai bit (tương ứng với một dây) trong khi phép nhân thông thường yêu cầu mỗi đầu vào, đầu ra là n bit (n dây). Vì vậy, tính toán trong miền SC có chi phí phần cứng nhỏ, nó thích hợp cho các ứng dụng cần tính toán song song lớn.
- **Giảm ảnh hưởng của nhiễu:** SC sử dụng một chuỗi độ dài 2^k để biểu diễn một số nhị phân độ dài k-bit. Vì vậy, nếu ở trong môi trường có nhiễu, khi bị lỗi bit mềm như một số bit bị lật thì giá trị chuỗi biểu diễn sẽ không thay đổi quá nhiều.

Mặc dù vậy, SC có một số nhược điểm thách thức tính hữu dụng của nó:

- **Thời gian tính toán:** Một phép cộng trong tính toán nhị phân thông thường yêu cầu một chu kỳ đồng hồ để hoàn thành còn trong SC cần số chu kỳ bằng độ dài chuỗi (2^k) để hoàn thành tính toán, điều này khiến thời gian tính toán tăng lên. Khả năng sử dụng tần số cao của mạch SC có thể bù đắp phần nào hạn chế này.
- **Tính tương quan:** Quan trọng hơn, SC giả định các chuỗi bit đầu vào không tương quan (độc lập) để có kết quả chính xác, sự tương quan này phụ thuộc vào khả năng ngẫu nhiên của mạch khởi tạo số ngẫu nhiên. Một phương án đơn giản thường được sử dụng là tạo ra các chuỗi giả ngẫu nhiên độc lập thông qua *thanh ghi dịch hồi tiếp tuyến tính* (LFSR), tuy nhiên LFSR không thể tạo ra các đầu vào độc lập hoàn toàn với nhau.

1.2.4 Một số công trình tiêu biểu



Hình 1.8: Bộ phát hiện biên: (a) sử dụng SC và (b) thiết kế thông thường [18].

Trong khoảng vài năm gần đây, SC đã trở lại như một sự lựa chọn hấp dẫn cho các ứng dụng nhỏ gọn và công suất thấp [19] . SC được sử dụng nhiều trong xử lý ảnh với mục tiêu tính toán đơn giản và khả năng chống tạp âm [20], [21]. Hình 1.8 mô tả kiến trúc bộ phát hiện biên, sử dụng tính toán ngẫu nhiên giảm chi phí phần cứng hơn 30 lần so với thiết kế số nhị phân thông thường với thời gian tính toán chậm hơn 3 lần [18]. SC đã được sử dụng để thiết kế bộ lọc FIR [22] với độ phức tạp thấp. SC có khả năng thay thế các phép nhân phức tạp trong phần cứng thành các cổng AND, XNOR đơn giản, điều này dẫn đến khả năng tính toán song song nhiều phép nhân trở nên khả thi.

Đặc biệt, tính toán ngẫu nhiên hưởng nhiều điểm lợi khi áp dụng vào mạng nơ-ron nhân tạo. Các nơ-ron trong mạng có cấu trúc và chức năng giống nhau và mạng nơ-ron là sự kết hợp của số lượng lớn các nơ-ron này. Do đó, thiết kế một cấu trúc nơ-ron điển hình với kích thước nhỏ và công suất thấp là vô cùng cần thiết. Một số thiết kế đã tận dụng được ưu thế của SC trong thiết kế mạng nơ-ron đơn giản và không yêu cầu độ chính xác quá cao [23]–[25].

CHƯƠNG 2 THIẾT KẾ PHẦN CỨNG MẠNG NƠ-RON NHÂN TẠO

Trong chương này sẽ trình bày thiết kế của một mạng nơ-ron nhân tạo sử dụng kỹ thuật tính toán ngẫu nhiên. Bài toán ứng dụng vào nhận dạng chữ số viết tay nhằm mục đích chính là kiểm tra độ khả dụng của thiết kế, còn kiến trúc đề xuất dễ dàng thay đổi tham số cho các bài toán tương tự như nhận dạng khuôn mặt, nhận dạng người.

2.1 Tổng quan

2.1.1 Phương án thiết kế

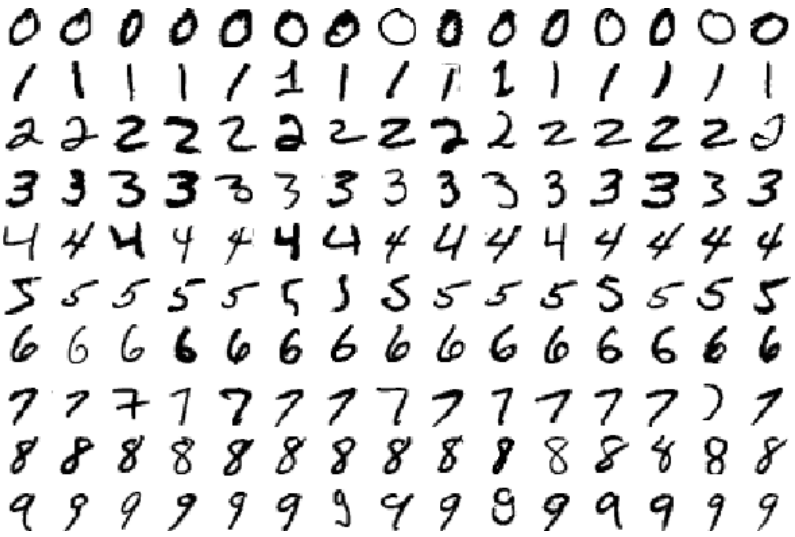
Đào tạo trực tuyến hay ngoại tuyến: Thiết kế phần cứng được giới hạn ở cấu trúc mạng nơ-ron truyền thẳng (feed-forward ANN), còn quá trình đào tạo được thực hiện trên phần mềm. Sau khi đào tạo hoàn thành, các trọng số của mạng sẽ được truyền vào bộ nhớ để tiến hành quá trình nhận dạng. Quá trình này được gọi là đào tạo ngoại tuyến, trái ngược với đào tạo trực tuyến. Sự lựa chọn này có hai mục đích: (i) đào tạo ngoại tuyến là phù hợp để kiểm chứng kiến trúc mạng nơ-ron sử dụng kỹ thuật tính toán ngẫu nhiên là khả thi; (ii) thiết kế mạng nơ-ron ngoại tuyến đơn giản hơn mạng trực tuyến, dễ dàng thiết kế một kiến trúc dạng tham số hóa, có khả năng mở rộng mạng cho các ứng dụng khác nhau.

Khả năng song song: Với mục tiêu với chi phí phần cứng thấp, khóa luận thiết kế mạng nơ-ron nhân tạo chỉ sử dụng một nơ-ron duy nhất, các tính toán sẽ được thực hiện tuần tự trong nơ-ron này. Tuy nhiên, để tận dụng lợi thế tính toán song song của phần cứng, đặc biệt là các phép toán đơn giản trong miền SC, khóa luận thiết kế nơ-ron với M đầu vào song song. Số đầu vào song song M được tham số hóa để có thể thay đổi tùy theo yêu cầu của mạng.

2.1.2 Thư viện nhận dạng chữ số viết tay

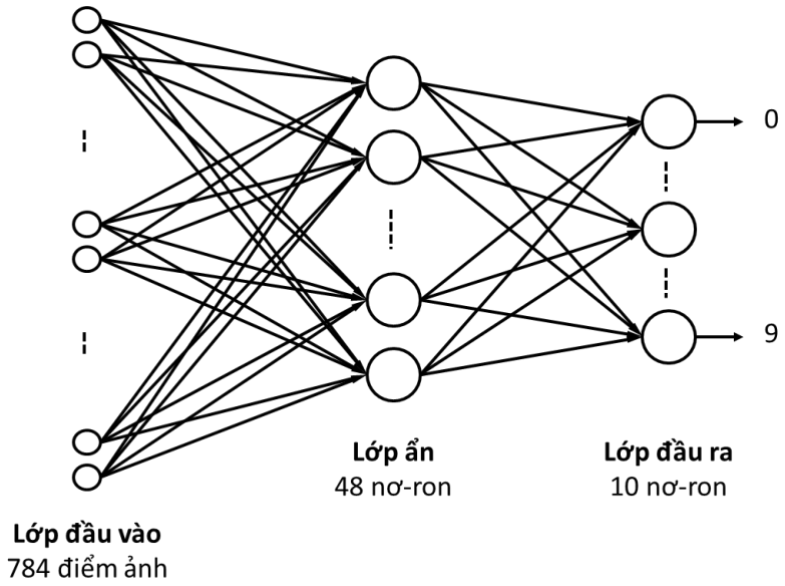
Khóa luận thiết kế một mô hình mạng nơ-ron ứng dụng cho nhận dạng chữ số viết tay sử dụng thư viện MNIST [26]. Cơ sở dữ liệu chữ số viết tay của MNIST bao gồm tập đào tạo 60.000 ảnh và tập nhận dạng 10.000 ảnh, tất cả ảnh đều có kích thước 28×28 điểm ảnh. Các điểm ảnh đã được chuẩn hóa xuống phạm vi (0.0; 1.0) giúp

đơn giản hóa dữ liệu và dễ dàng cho áp dụng kỹ thuật tính toán ngẫu nhiên. Đây là một trong những thư viện thông dụng nhất để kiểm tra hiệu năng của mạng nơ-ron nhân tạo [27], [28].



Hình 2.1: Dữ liệu chữ số viết tay (thư viện MNIST [26]).

Kiến trúc mạng nơ-ron hệ thống nhận dạng chữ số viết tay được thể hiện như Hình 2.2. Đầu vào mạng nơ-ron là 784 điểm ảnh của bức ảnh kích thước 28×28 cần nhận dạng chữ số. Dữ liệu ở lớp đầu vào được truyền qua một lớp ẩn với 48 nơ-ron (có thể thay đổi), sau đó truyền qua 10 nơ-ron lớp đầu ra tương ứng các chữ số từ 0 đến 9 để phân loại.



Hình 2.2: Kiến trúc mạng nơ-ron nhận dạng chữ số viết tay.

2.1.3 Nơ-ron tính toán song song

Một trong những khả năng vượt trội giữa thiết kế phần cứng so với phần mềm là tính toán song song. Tuy nhiên, do giới hạn số điểm ảnh đầu vào thông qua đường bus và tài nguyên của FPGA nên không thể tính toán song song toàn bộ mạng nơ-ron nhân tạo. Thay vào đó, khóa luận đề xuất kiến trúc mạng nơ-ron kết hợp giữa song song và nối tiếp. Mỗi mô-đun nơ-ron có khả năng tính toán song song tối đa M đầu vào, việc thay đổi hằng số M là khả thi sao cho phù hợp với tài nguyên của FPGA và yêu cầu của ứng dụng. Sự tính toán song song này sẽ bù đắp chi phí thời gian do kiến trúc tuần tự chỉ sử dụng một nơ-ron của mạng.

Bảng 2.1: Giả mã thiết kế nơ-ron song song.

Hằng số: N : tổng số đầu vào nơ-ron M : số đầu vào song song **Đầu vào**: $x(i_{1:M})$: dữ liệu ảnh hoặc đầu ra lớp trước $w(i_{1:M})$: trọng số b : độ lệch **Đầu ra**: y : giá trị đầu ra nơ-ron

Calculate:
sum = b;
for i in 1 to N/M
sum = sum + x(1)*w(1) + x(2)*w(2) + … + x(M)*w(M);
Activate:
y = activate(sum);

Giả mã mô-đun nơ-ron song song được thể hiện như Bảng 2.1. Quy trình tính toán được chia làm hai phần chính:

(1) **Calculate:** Giá trị biến *sum* được khởi tạo bằng giá trị của độ lệch *b*. Trong mỗi chu kỳ, tổng của M đầu vào với các trọng số của chúng được tích lũy trong biến

sum. Sau N/M chu kỳ, biến *sum* có giá trị bằng tổng của tất cả tích đầu vào với trọng số cộng với độ lệch.

(2) **Activate:** Sau khi tính xong tất các đầu vào của nơ-ron, hàm kích hoạt *activate* (ReLu hoặc Sigmoid) sẽ thực hiện chức năng để tính toán đầu ra nơ-ron.

2.2 Nơ-ron SC

2.2.1 Giới thiệu

Quá trình tính toán trong mỗi nơ-ron được biểu diễn theo phương trình sau:

$$y = \varphi(\sum_{i=1}^{N} x_i \times w_i + b) \quad (2.1)$$

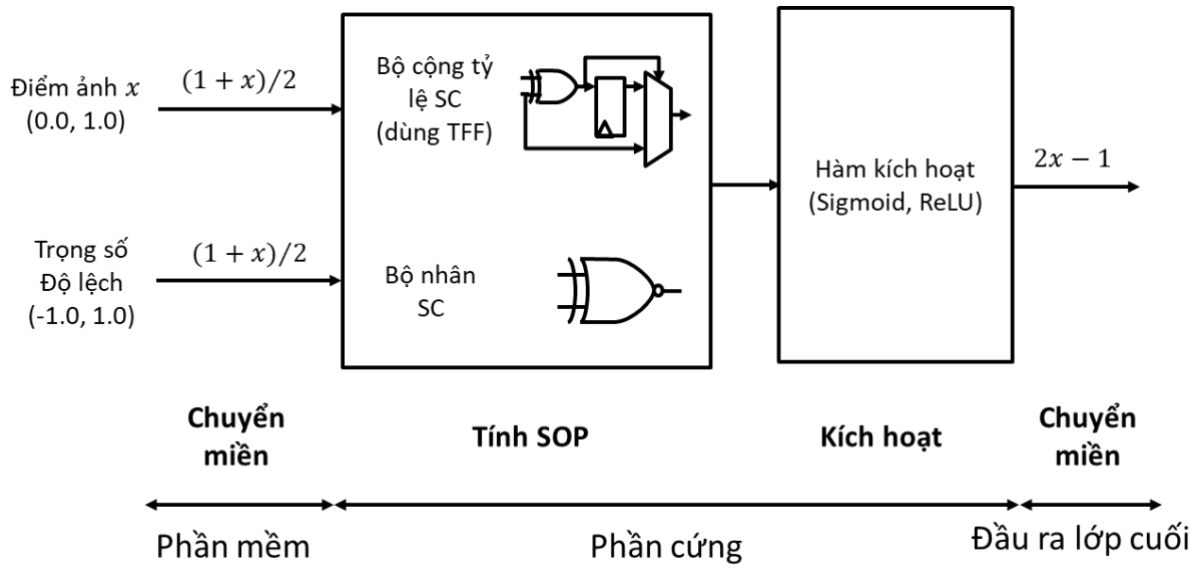
Như đã nhắc ở chương trước, mạng nơ-ron là tập hợp của nhiều nơ-ron cấu trúc giống nhau. Tuy nhiên, các nơ-ron thông thường lại có độ phức tạp cao khi phải tính tổng của các tích (SOP), sử dụng nhiều bộ nhân và bộ cộng. Ngoài ra, hàm kích hoạt cũng góp phần tăng độ trễ tính toán và chi phí phần cứng cho mô-đun nơ-ron này. Đã có các nghiên cứu sử dụng bộ DSP có sẵn trong FPGA để thiết kế mô-đun nơ-ron [29] giúp giảm thời gian tính toán, tuy nhiên chi phí phần cứng và khả năng tính toán song song các nơ-ron vẫn là nút thắt khó giải quyết đối với số nhị phân thông thường.

Sử dụng kỹ thuật tính toán ngẫu nhiên có thể giải quyết được bài toán về chi phí cho các mô-đun nơ-ron này. Sự thay thế hoàn toàn các phép nhân và phép cộng bởi các mạch logic sẽ giảm đáng kể chi phí phần cứng và công suất tiêu thụ. Giao tiếp của nơ-ron SC sẽ có nhiều đầu vào tính toán song song và một đầu ra tương tự như phần 2.1.3 đã nhắc đến, sự tính toán song song này sẽ tận dụng lợi thế của các mạch logic và bù đắp cho nhược điểm về thời gian tính toán của SC.

2.2.2 Tính toán trong SC lưỡng cực

Do các trọng số là các số thực có dấu vì vậy thiết kế đề xuất sử dụng dạng lưỡng cực của SC để tính toán. Mô phỏng phần mềm cho thấy các trọng số và độ lệch là các giá trị số thực có dấu nằm trong khoảng (-1,0; 1,0), vì vậy đảm bảo khả năng biểu diễn của số SN. Còn điểm ảnh đã được chuẩn hóa về khoảng (0,0; 1,0) thay vì (0; 256) như

thông thường. Sau đó, dữ liệu đầu vào (điểm ảnh, trọng số, độ lệch) sẽ được chuyển về dạng lưỡng cực để tính toán trong mạng nơ-ron. Sau khi hoàn thành, dữ liệu kết quả lớp đầu ra sẽ chuyển về dạng số nhị phân thông thường để nhận dạng. Cách chuyển đổi dữ liệu được thể hiện như Hình 2.3.



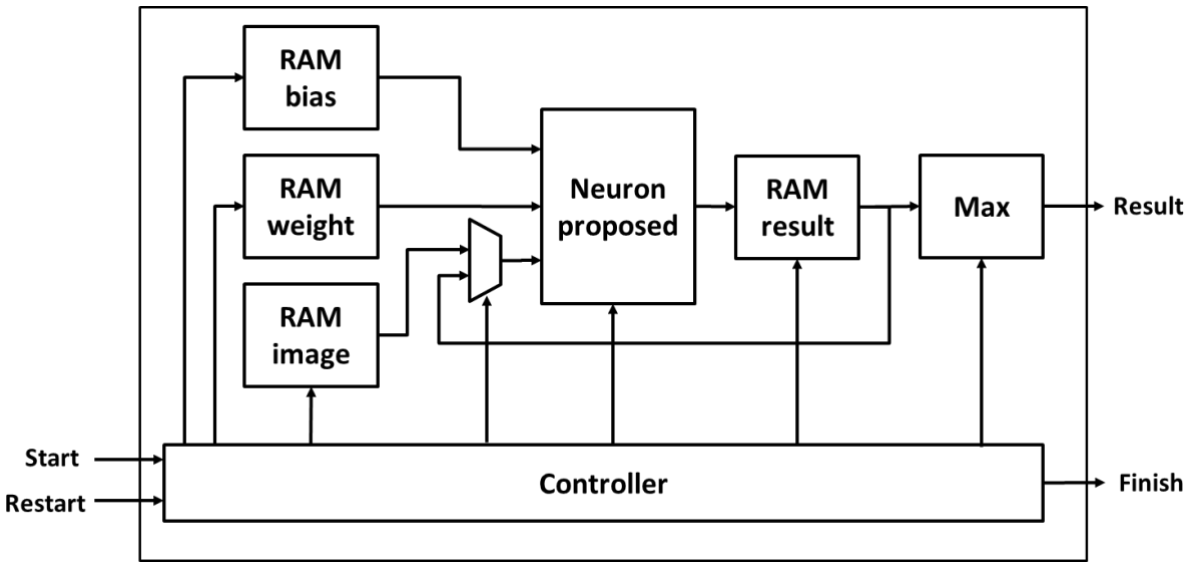
Hình 2.3: Chuyển đổi dữ liệu khi tính toán.

Quy trình chuyển đổi dữ liệu được thể hiện như Hình 2.3. Quá trình chuẩn hóa và chuyển miền dữ liệu được thực hiện trên phần mềm (Matlab), dữ liệu sau đó sẽ được đưa vào mạng nơ-ron để tính toán. Đầu vào kiến trúc phần cứng là dữ liệu đã chuyển sang miền lưỡng cực thông qua các công cụ của Matlab, còn đầu ra phần cứng sẽ chuyển ngược lại từ miền lưỡng cực sang miền nhị phân để có kết quả đầu ra chính xác.

2.3 Kiến trúc phần cứng

Với mục tiêu với chi phí phần cứng thấp, khóa luận thiết kế mô hình mạng nơ-ron nhân tạo chỉ sử dụng một nơ-ron duy nhất, các tính toán sẽ được thực hiện tuần tự trong nơ-ron này thông qua bộ điều khiển.

2.3.1 Kiến trúc chung



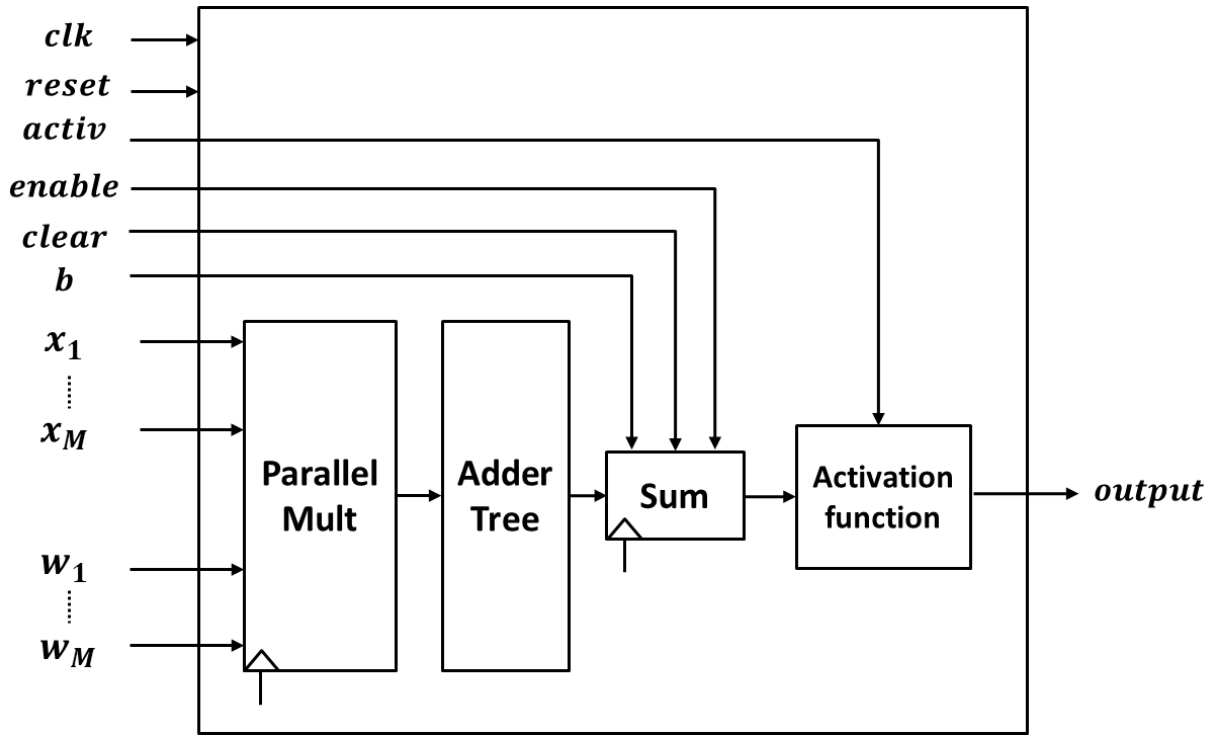
Hình 2.4: Kiến trúc chung mạng nơ-ron.

Sơ đồ khối kiến trúc ANN được thể hiện như Hình 2.4, có thể tóm tắt như sau:

- Toàn bộ dữ liệu được lưu trong 4 RAM*:* **RAM bias** lưu độ lệch*;* **RAM weight** lưu trọng số*;* **RAM image** lưu giá trị các điểm ảnh đã chuẩn hóa*;* **RAM result** lưu kết quả đầu ra nơ-ron. Tất cả dữ liệu được truyền sẵn vào vào RAM trước khi mạng nơ-ron kích hoạt.
- Bộ **Controller** điều khiển toàn bộ quá trình tính toán của mạng nơ-ron. Khi có tín hiệu *Start, Restart* được kích hoạt, nó sẽ điều khiển các quá trình của mạng. Đầu tiên bộ **Controller** điều khiển dữ liệu từ các RAM truyền vào mô-đun **Neuron proposed** và đếm cho đến tất các đầu vào được tính toán, sau đó dữ liệu đầu ra đưa vào **RAM result**. Quy trình lặp lại cho đến khi tất cả nơ-ron của tất cả các lớp được tính toán, khi đó nó sẽ kích hoạt bộ **Max** để tính đầu ra nhận dạng và kích hoạt tín hiệu *Finish* khi hoàn thành.
- Mô-đun **Neuron proposed** thực hiện chức năng nơ-ron xử lý song song M đầu vào, có thể chuyển đổi giữa kiến trúc nhị phân thông thường hoặc kiến trúc tính toán ngẫu nhiên.
- Bộ **Mux** có nhiệm vụ lựa chọn đầu vào mô-đun nơ-ron: điểm ảnh đối với nơ-ron lớp ẩn đầu tiên hoặc kết quả của các lớp trước được đọc từ **RAM result** đối với các lớp còn lại.

- Trong quá trình tính toán lớp đầu ra, bộ **Max** sẽ được kích hoạt để tìm đầu ra có giá trị cao nhất (tương ứng với chữ số có xác suất nhận dạng cao nhất).

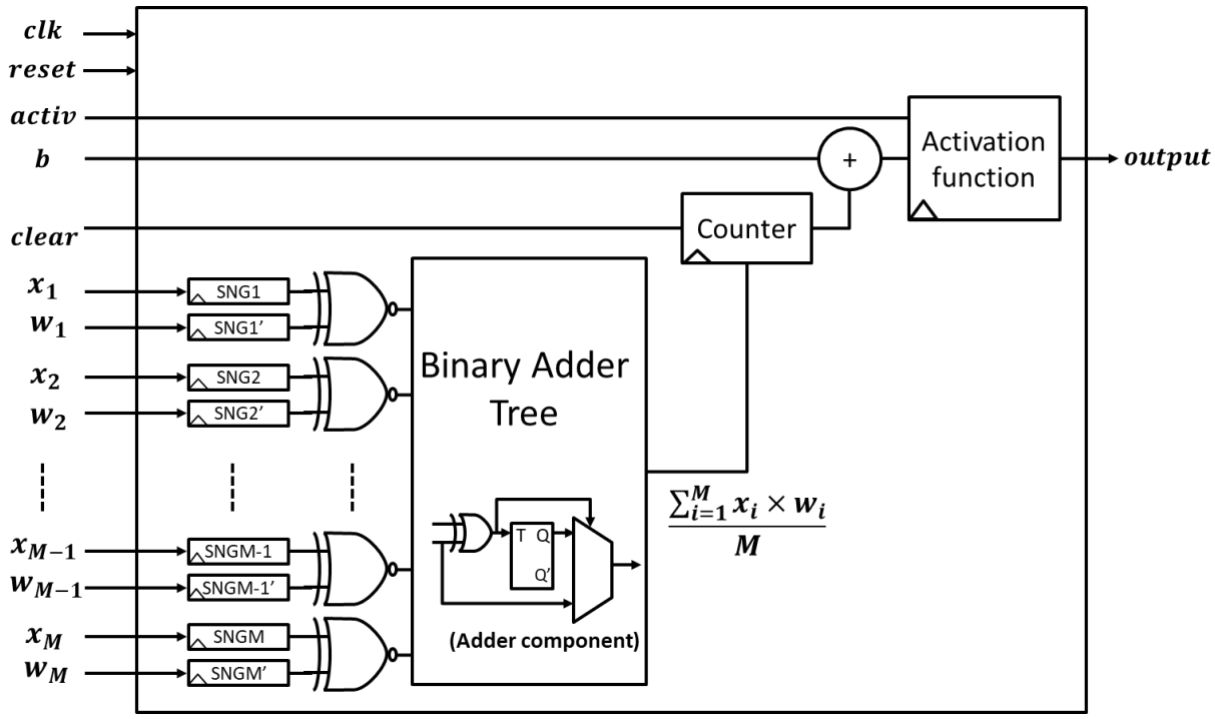
2.3.2 Kiến trúc nơ-ron tính toán song song



Hình 2.5: Kiến trúc nơ-ron song song M đầu vào.

Kiến trúc nơ-ron song song gồm các đường dữ liệu M đầu vào x, M trọng số w, độ lệch b và các tín hiệu điều khiển được truyền từ bộ **Controller** như Hình 2.5. Khi tín hiệu *enable* kích hoạt, nơ-ron sẽ tính toán tổng của các tích (SOP) thông qua M bộ nhân song song **Parallel Mult** và các bộ cộng cây nhị phân **Adder Tree** và lưu giá trị vào thanh ghi **Sum**. Tín hiệu *activ* điều khiển hàm kích hoạt **Activation function** sẽ kích hoạt đầu ra nơ-ron. Sau đó, tín hiệu *clear* xóa tổng cũ để bắt đầu tính nơ-ron ở vị trí tiếp theo trong mạng.

2.3.3 Kiến trúc nơ-ron SC



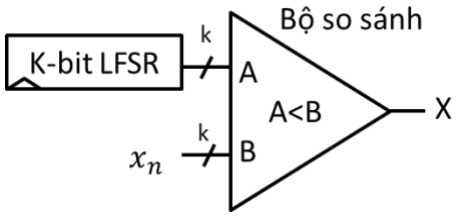
Hình 2.6: Kiến trúc nơ-ron tính toán ngẫu nhiên song song M đầu vào.

Hình 2.6 thể hiện kiến trúc phần cứng mô-đun nơ-ron đề xuất. Chu trình tính toán được thể hiện như sau:

- Tại thời điểm bắt đầu, tín hiệu $clear$ kích hoạt ($clear = 1$), các bộ SNG đặt giá trị ban đầu, bộ đếm (**Counter**) được xóa về 0.
- Sau đó, M đầu vào ($x_1 \rightarrow x_M$) và M trọng số ($w_1 \rightarrow w_M$) truyền qua các bộ SNG để tạo ra các chuỗi SN. Trong 2^k chu kỳ, lần lượt các giá trị của các chuỗi SN đi qua các cổng XNOR thực hiện bộ nhân và qua cây cộng nhị phân (**Binary adder tree***)* với thành phần mỗi bộ cộng gồm một cổng XOR, MUX và TFF như phần 1.2.1 đã giới thiệu. Đầu ra cây cộng nhị phân là chuỗi bit biểu diễn giá trị tổng của các tích (SOP) được đưa vào bộ đếm **Counter** để chuyển sang miền nhị phân lưỡng cực và tích lũy.
- Tổng tại bộ đếm **Counter** được tích lũy cho đến khi tín hiệu kích hoạt (*activ)* được bật, khi đó dữ liệu được cộng với độ lệch *b* và truyền qua bộ kích hoạt **Activation function** (ReLu hoặc Sigmoid) để tính đầu ra nơ-ron.

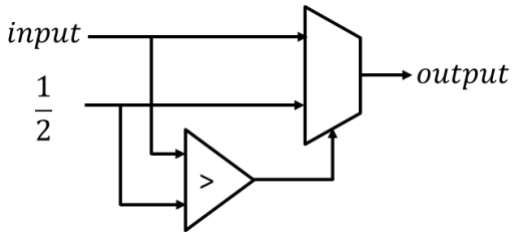
Bộ tạo số ngẫu nhiên (SNG): Mỗi bộ tạo số ngẫu nhiên (SNG) có gồm một thanh ghi dịch hồi tiếp tuyến tính (LFSR) có độ dài bằng độ dài số nhị phân đầu vào và một bộ

so sánh (Hình 2.7). Bộ so sánth xuất bit 1 nếu số ngẫu nhiên tạo bởi LFSR nhỏ hơn số nhị phân đầu vào và bit 0 nếu ngược lại. Sau 2^K chu kỳ, chuỗi đầu ra bộ so sánh có xác suất biểu diễn số nhị phân đầu vào trên miền SC.



Hình 2.7: Kiến trúc bộ tạo số ngẫu nhiên (SNG).

Bộ kích hoạt: Thiết kế đề xuất sử dụng SC dạng lưỡng cực nên đầu ra của bộ đếm là số nhị phân lưỡng cực biểu diễn giá trị lưỡng cực $2p_{SOP} - 1$ (như phần 1.2.1 đã đề cập). Vì vậy, hàm kích hoạt ReLU được thể hiện như Hình 2.8, bao gồm 1 bộ so sánh và 1 bộ MUX. Chú ý rằng số 1/2 dùng để so sánh chính là số 0 sau khi biểu diễn ở dạng lưỡng cực.

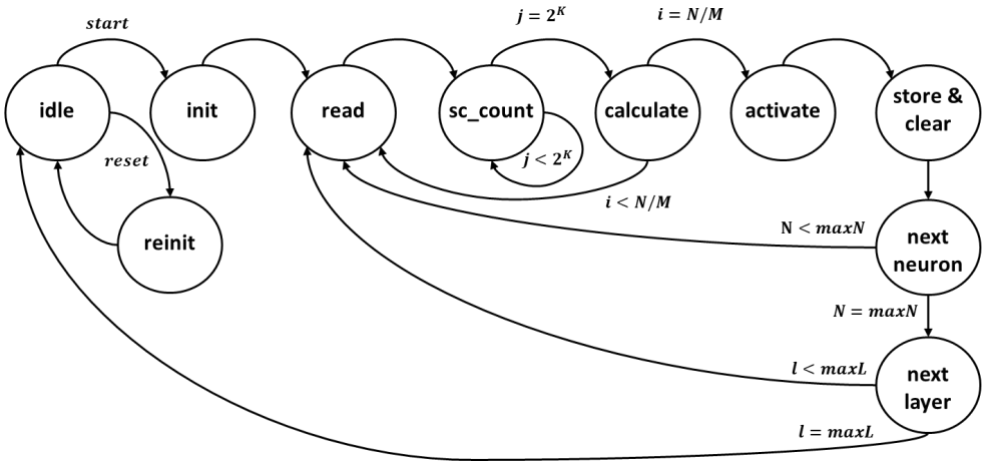


Hình 2.8: Hàm ReLU cho số nhị phân dạng lưỡng cực.

Đối với hàm Sigmoid, do tính toán phức tạp nên không có mạch thực thi chính xác phương trình hàm. Vì vậy, một bộ Look-up table (LUT) tương tự [30] có kích thước 256-byte (8-bit địa chỉ) được sử dụng để xấp xỉ hàm kích hoạt này.

2.3.4 Bộ điều khiển

Bộ điều khiển là một máy trạng thái hữu hạn (FSM) Moore với các trạng thái được hiển thị trong Hình 2.9. Trong trường hợp sử dụng nơ-ron nhị phân thông thường, trạng thái sc_count tương ứng với thời gian để tính toán trong miền SC sẽ được bỏ qua.



Hình 2.9: Máy trạng thái hữu hạn Moore điều khiển mạng nơ-ron.

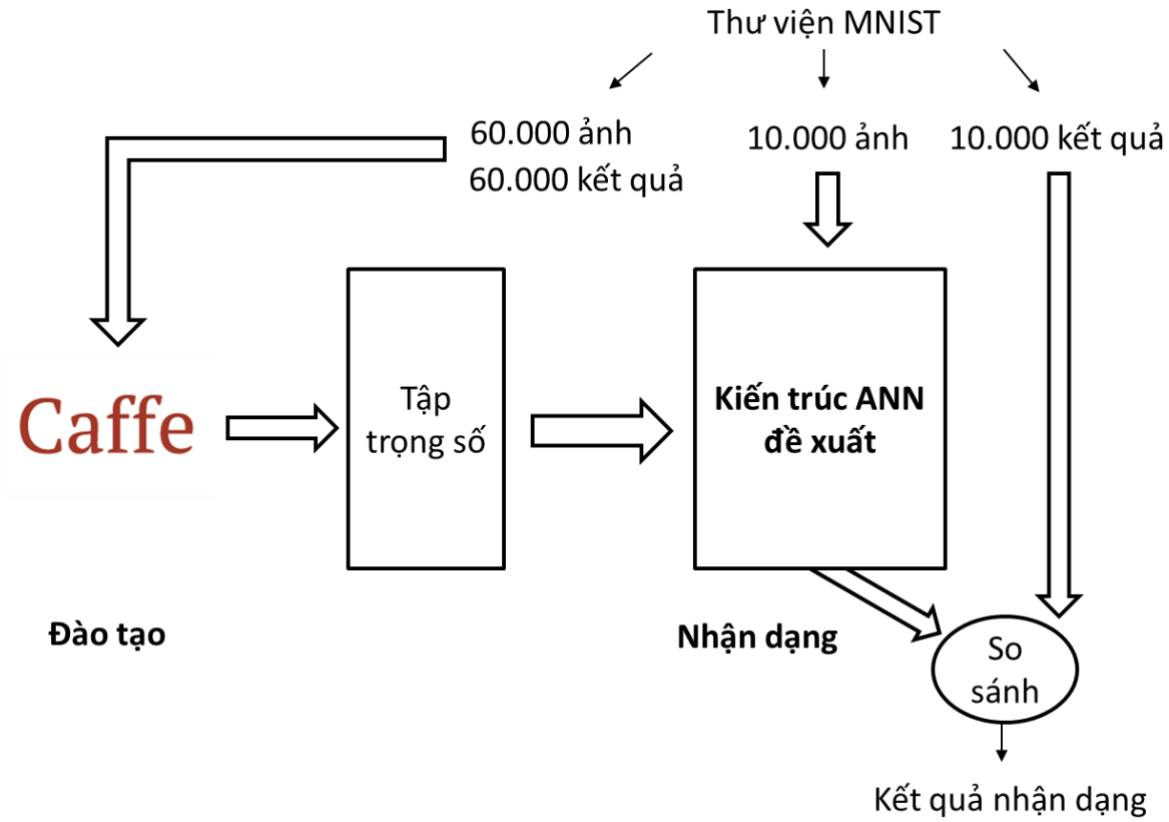
Quy trình hoạt động của máy trạng thái được mô tả như sau: Trạng thái khởi tạo là *idle* và trạng thái này sẽ giữ nguyên cho đến khi có tín hiệu đầu vào (*start, reset).* Khi tín hiệu *start* được kích hoạt, máy sẽ chuyển sang trạng thái *init,* khi đó các RAM sẽ được kích hoạt để chuẩn bị đọc, ghi dữ liệu và chuyển sang trạng thái tiếp theo là *read*. Ở trạng thái *read,* các điểm ảnh, trọng số, độ lệch sẽ được đọc từ RAM. Ở trạng thái tiếp theo, *sc_count,* các bộ nơ-ron SC song song sẽ được kích hoạt và tính toán. Trạng thái sc_*count* sẽ đếm 2^k chu kỳ (k là số bit biểu diễn chuỗi SC) để quá trình khởi tạo tính toán các chuỗi SC hoàn thành thì chuyển sang trạng thái *calculate.* Trạng thái *calculate* sẽ kiểm tra đã đủ tích lũy đủ đầu vào nơ-ron hay chưa, nếu chưa thì quay lại trạng thái *read* để tính toán, còn khi tất quả đầu vào của nơ-ron được tích lũy giá trị thì trạng thái *activate* được kích hoạt. Tại trạng thái *activate,* bộ kích hoạt (ReLU, Sigmoid) sẽ bắt đầu tính đầu ra nơ-ron. Chu kỳ tiếp theo, trạng thái *store & clear* sẽ điều khiển giá trị đầu ra lưu vào *Ram result,* đồng thời điều khiển mô-đun nơ-ron sẽ xóa bộ đếm để chuẩn bị cho giá trị mới. Tại trạng thái *next_neuron,* bộ điều khiển sẽ kiểm tra có phải nơ-ron cuối cùng của lớp hay không, nếu là nơ-ron cuối thì chuyển sang trạng thái *next layer* còn nếu không thì chuyển về về trạng thái *read* để tính toán nơ-ron tiếp theo. Tương tự, rạng thái *next layer* kiểm tra mạng đã hoàn thành tính toán và chuyển sang trạng thái *idle.* Khi tín hiệu *reset* được kích hoạt, FSM sẽ khởi động lại trạng thái và xóa dữ liệu dữ liệu cũ trong các thanh ghi.

Lưu ý rằng ở bộ điều khiển coi M là ước số của N (tức là là N chia hết cho M), trong trường hợp số nơ-ron đầu vào M không chia hết, cần phải chèn thêm các nơ-ron có giá trị 0 vào mạng để hệ thống hoạt động được chính xác nhất. Bộ điều khiển có tham khảo từ mã nguồn mở [31].

CHƯƠNG 3 KẾT QUẢ VÀ ĐÁNH GIÁ

Trong chương trước đã trình bày kiến trúc phần cứng đề xuất, còn trong chương này sẽ trình bày phương pháp đánh giá, kết quả mô phỏng và thực thi trên công nghệ FPGA.

3.1 Phương pháp kiểm chứng



Hình 3.1: Phương pháp kiểm chứng mạng nơ-ron nhân tạo.

Như đã đề cấp ở chương trước, quá trình đào tạo trọng số cho mạng nơ-ron được thực hiện trên phần mềm. Quá trình đào tạo sử dụng mã nguồn mở Caffe [32], được thiết kế bởi phòng thí nghiệm BAIR, đại học Berkeley. Sau khi đào tạo sử dụng Caffe, tập trọng số (bao gồm cả độ lệch) được truyền vào các RAM trong kiến trúc mạng nơ-ron nhân tạo đề xuất. Sau đó, sử dụng 10.000 bộ thử nghiệm của thư viện MNIST để kiểm chứng khả năng nhận dạng của mạng nơ-ron nhân tạo.

Bảng 3.1 so sánh kết quả nhận dạng chữ số viết tay sử dụng 2 hàm kích hoạt phổ biến hiện này là hàm Sigmoid và hàm ReLU với số lượng nơ-ron lớp ẩn khác nhau:

lần lượt là 16, 48, 64 và 800 nơ-ron. Với số nơ-ron lớp ẩn là 800 thì thu được độ chính xác cao nhất, tuy nhiên với số nơ-ron lớp ẩn ít hơn (16, 48, 64) thì độ chính xác không giảm đi nhiều, vì vậy không cần thiết phải tạo nhiều lớp ẩn khiến kiến trúc phức tạp và tăng thời gian tính toán.

Bảng 3.1: Kết quả nhận dạng chữ số viết tay trên mã nguồn mở Caffe [32].

Hàm kích hoạt	Sigmoid				ReLU			
Số nơ-ron lớp ẩn	16	48	64	800	16	48	64	800
Độ nhận dạng (%)	88,04	89,01	89,1	89,48	92,17	92,63	92,91	93,49

Ngoài ra, sử dụng các phương pháp nhận dạng khác như máy vec-tơ hỗ trợ (SVM), mạng nơ-ron sâu (DNN), mạng nơ-ron tích chập (CNN)… có khả năng nhận dạng chữ số viết tay của thư viện MNIST với độ chính xác cao hơn [26]. Tuy nhiên, trong khuôn khổ khóa luận coi nhận dạng chữ số viết tay như một ứng dụng điển hình để chứng minh độ tin cậy và hiệu năng của kiến trúc đề xuất, vì vậy khóa luận không cải thiện độ chính xác thêm.

Thiết kế mạng nơ-ron nhân tạo được viết bằng ngôn ngữ VHDL, một ngôn ngữ mạnh mẽ để mô tả các hệ thống số lớn và phức tạp. Sau đó, quá trình mô phỏng được thực hiện trên công cụ ModelSim của hãng Mentor. Cuối cùng, quá trình tổng hợp được thực hiện trên Kit FPGA Artix-7 của hãng Xilinx. Artix-7 là một FPGA có kích thước và hiệu năng vừa phải, tuy nhiên nó là dòng có mức tiêu thụ nhỏ nhất, phù hợp với mục tiêu giảm chi phí phần cứng của khóa luận này.

3.2 Kết quả mô phỏng

Để đánh giá độ chính xác đầu ra, sai số toàn phương trung bình (MSE) được sử dụng để ước lượng sai số đầu ra của các mô-đun thiết kế so với tính toán số thực dấu phẩy động.

$$MSE = \sum_{i=0}^{N-1} \frac{\left(O_{floating\_point} - O_{proposed}\right)^2}{N} \quad (3\text{-}1)$$

Kiểm chứng kiến trúc SC

Bảng 3.2: So sánh MSE của bộ nhân và bộ cộng SC so với công trình khác.

Mô hình	Vincent Lee [15]	**Kiến trúc thiết kế**	
Kiểu biểu diễn	8-bit đơn cực	8-bit đơn cực	8-bit lưỡng cực
Bộ nhân SC	$2{,}57 \times 10^{-4}$	$7{,}43 \times 10^{-6}$	$1{,}98 \times 10^{-4}$
Bộ cộng SC	$1{,}91 \times 10^{-6}$	$1{,}89 \times 10^{-6}$	$1{,}32 \times 10^{-5}$

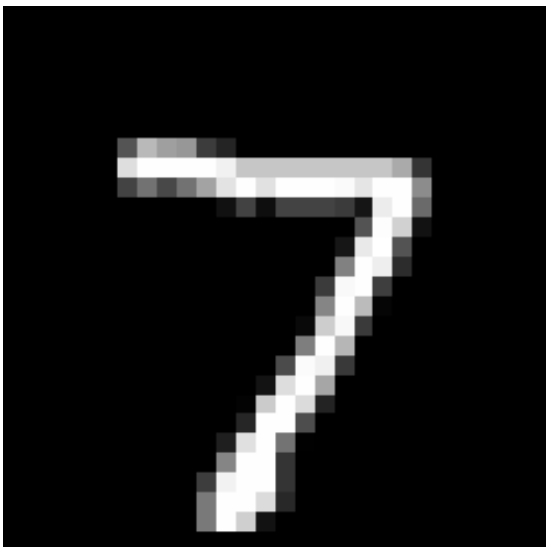
Bảng 3.2 minh họa kết quả mô phỏng kiến trúc bộ nhân vào bộ cộng SC sử dụng 8-bit độ chính xác (chuỗi SC có độ dài 2^8 bit). Bộ nhân SC sử dụng cổng AND cho dạng đơn cực và cổng XNOR cho dạng lưỡng cực, còn bộ cộng tỷ lệ SC sử dụng bộ cộng cải tiến (MUX, XOR, TFF) thay thế cho bộ MUX thông thường, như đã thảo luận ở phần 1.2.1.

Bảng 3.3: So sánh MSE và thời gian tính toán giữa nơ-ron song song nhị phân và nơ-ron song song SC 16 đầu vào sử dụng hàm kích hoạt ReLU.

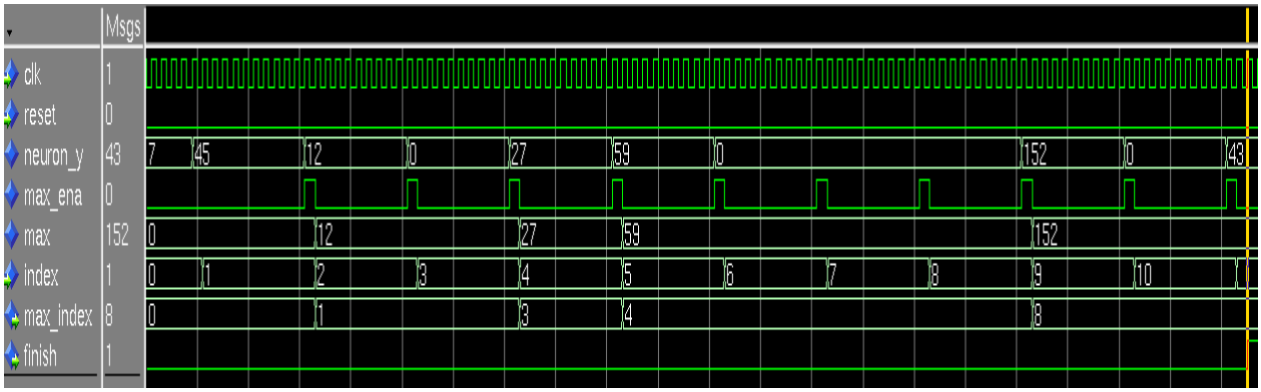
Mô hình	Độ chính xác	Nơ-ron nhị phân 16 đầu vào	Nơ-ron SC 16 đầu vào
MSE	8-bit	$2{,}02 \times 10^{-3}$	$7{,}05 \times 10^{-2}$
	10-bit	$1{,}47 \times 10^{-4}$	$9{,}02 \times 10^{-4}$
Thời gian tính (chu kỳ)	8-bit	2	257
	10-bit	2	1025

Bảng 3.3 so sánh độ chính xác và thời gian tính toán giữa nơ-ron nhị phân thông thường và nơ-ron sử dụng tính toán ngẫu nhiên với 16 đầu vào song song với số bit biểu diễn dữ liệu lần lượt là 8-bit và 10-bit. Từ bảng cho thấy, mô hình nơ-ron tính toán ngẫu nhiên sử dụng nhiều thời gian tính toán hơn so với thông thường. Sai số đầu ra MSE của tính toán ngẫu nhiên không chênh lệch quá nhiều so với tính toán sử dụng số nhị phân.

Hiệu suất nhận dạng chữ số viết tay



Hình 3.2: Chữ số viết tay cần nhận dạng.



Hình 3.3: Kết quả mô phỏng thiết kế trên ModelSim (lớp đầu ra).

Hình 3.2 và Hình 3.3 lần lượt là chữ số viết tay cần nhận dạng và kết quả mô phỏng kiến trúc phần cứng sử dụng số nhị phân 10-bit trên công cụ ModelSim của hãng Mentor tại thời điểm tính toán 10 nơ-ron lớp đầu ra. Tín hiệu *neuron_y* là kết quả đầu ra của nơ-ron còn tín hiệu *max* tìm giá trị lớn nhất từ đầu ra các nơ-ron này. Khi tín hiệu *finish* kết thúc, giá trị *max_index* tương ứng với chỉ số của nơ-ron lớn nhất sẽ được xuất đầu ra. Kết quả cuối cùng của *max_index* là 8, tương ứng với nhận dạng chữ số 7.

Bảng 3.4 là kết quả nhận dạng chữ số viết tay trên 10.000 trường hợp tập nhận dạng của thư viện MNIST. Từ bảng có thể thấy, sử dụng hàm kích hoạt ReLU có kết quả tốt hơn sigmoid. Đặc biệt, với kích thước nơ-ron lớp ẩn là 64 sử dụng hàm ReLU cho kết quả nhận dạng lên đến 92,47%. Sai số nhận dạng giữa kiến trúc phần cứng và

mô phỏng Caffee đến từ việc kiến trúc trúc phần cứng sử dụng số thực dấu phẩy tĩnh 10-bit (fixed-point) thay thế cho số thực dấu phẩy động 32-bit trên phần mềm. Hơn nữa, nếu hàm kích hoạt là Sigmoid thì sai số tăng lên do sử dụng một bộ LUT 256KB để xấp xỉ chức năng của hàm này.

Bảng 3.4: So sánh kết quả nhận dạng trên phần mềm và kiến trúc đề xuất.

Hàm kích hoạt	Sigmoid			ReLU		
Số nơ-ron lớp ẩn	16	48	64	16	48	64
Mô phỏng Caffee C++ (%)	88,04	89,01	89,1	92,17	92,63	92,91
Kiến trúc nhị phân 10-bit (%)	87,60	86,63	85,94	91,56	92,18	**92,47**
Sai lỗi (%)	0,44	2,38	3,16	0,61	0,45	**0,44**

3.3 Thực thi FPGA

Quá trình tổng hợp được thực hiện trên Kit FPGA Artix-7 của hãng Xilinx. Artix-7 là một FPGA có kích thước và hiệu năng vừa phải, tuy nhiên nó là dòng FPGA có mức tiêu thụ nhỏ nhất, phù hợp với mục tiêu giảm chi phí phần cứng của khóa luận này.

Bảng 3.5: Kết quả thực thi của mô-đun nơ-ron sử dụng tính toán nhị phân và tính toán ngẫu nhiên.

Kết quả	Nơ-ron nhị phân 16 đầu vào	Nơ-ron SC 16 đầu vào
Tần số tối đa	250MHz	**286MHz**
LUTs	1268	**416**
Thanh ghi	23	299
IO	277	277
Công suất động	0,045	**0,039**

Bảng 3.5 trình bày kết quả thực thi FPGA giữa 2 mô-đun nơ-ron song song với dữ liệu được biểu diễn dưới dạng số thực dấu phẩy tĩnh 10-bit đã đề xuất. Sử dụng nơ-ron sử dụng tính toán ngẫu nhiên tăng tần số hoạt động tối đa, giảm số LUTs và công suất động trong khi tăng số thanh ghi do sử dụng số lượng lớn *bộ khởi tạo số ngẫu nhiên* (SNG).

Kết quả tổng hợp FGPA Artix-7 được thể hiện trong Bảng 3.6 và Bảng 3.7. Kiến trúc mạng nơ-ron để tổng hợp sử dụng kiến trúc nơ-ron nhị phân, biểu diễn dữ liệu dưới dạng số thực dấu phẩy tĩnh 10-bit, sử dụng 48 nơ-ron lớp ẩn và hàm kích hoạt ReLU. Mạng nơ-ron có tần số tối đa là 158MHz (nhỏ hơn tần số tối đa của mô-đun nơ-ron do chưa tối ưu hoàn toàn bộ điều khiển), thời gian từ lúc bắt đầu cho đến khi quá trình nhận dạng hoàn thành là $30{,}93\mu s$ và tiêu thụ công suất $0{,}202W$ cho toàn bộ thiết kế. Chi phí phần cứng sử dụng là 1659 LUTs và 1020 thanh ghi. Kiến trúc đề xuất sử dụng chi phí phần cứng nhỏ so với tài nguyên có sẵn của FPGA.

Bảng 3.6: Hiệu năng của mạng nơ-ron (số nhị phân 10-bit, 48 lớp ẩn, hàm ReLU).

Kết quả	Kiến trúc đề xuất
Tần số	$\mathbf{158}\boldsymbol{MHz}$
Tổng chu kỳ	4888
Thời gian	$30{,}93\mu s$
Công suất động	$0{,}07W$
Công suất chip	$0{,}202W$

Bảng 3.7: Chi phí phần cứng sử dụng.

Chi phí phần cứng	Kiến trúc đề xuất	Tài nguyên FPGA	Sử dụng
LUTs	1659	133800	1,24%
Thanh ghi	1020	267600	0,38%
Mux	0	100350	0%
Slice	578	33450	1,72%
DSP	0	740	0%
BRAM	0,5	365	0,14%
IO	8	400	2,0%

KẾT LUẬN

Mạng nơ-ron nhân tạo ngày càng có nhiều ứng dụng trong thực tiễn, tuy nhiên, một trong những thách thức khi ứng dụng mạng nơ-ron chính là hiệu năng và chi phí phần cứng. Vì vậy, khóa luận đã nghiên cứu khái quát về mạng nơ-ron nhân tạo và các ứng dụng của nó, sau đó đề xuất sử dụng kỹ thuật tính toán xấp xỉ (stochastic computing), một kỹ thuật tính toán trên miền xác suất để tối ưu cho mạng. Từ đó, một kiến trúc mạng nơ-ron truyền thẳng được thực thi trên công nghệ FPGA và sử dụng ngôn ngữ mô tả phần cứng VHDL để thiết kế. Mạng nơ-ron đề xuất sử dụng tính toán tuần tự, đã được đào tạo ngoại tuyến và truyền sẵn trọng số vào bộ nhớ. Mô hình mạng nơ-ron đề xuất chỉ sử dụng một nơ-ron duy nhất tính toán song song nhiều đầu vào nhằm giảm tối thiểu chi phí phần cứng. Sau đó, khóa luận đề xuất một kiến trúc nơ-ron sử dụng kỹ thuật tính toán ngẫu nhiên 16 đầu vào song song, kết quả mô phỏng cho thấy sai số đầu ra xấp xỉ tính toán nhị phân thông thường nhưng giảm LUTs và công suất tiêu thụ thấp hơn. Cuối cùng, kết quả tổng hợp trên FPGA cho thấy mạng nơ-ron đề xuất có hiệu năng nhận dạng chữ số viết tay lên đến 92,18% khi sử dụng số nhị phân 10-bit độ chính xác, 48 nơ-ron lớp ẩn và hàm kích hoạt ReLU. Kiến trúc nhị phân này có thời gian tính toán là $30{,}93\mu s$, sử dụng 1659 LUTs và 1020 thanh ghi. Với chi phí phần cứng thấp, mạng nơ-ron đề xuất hoàn toàn có thể mở rộng hoặc nhúng vào các thiết bị IoT.

Kiến trúc mô-đun nơ-ron sử dụng kỹ thuật tính toán ngẫu nhiên trên kiến trúc ANN 2-3-2 đã tham gia cuộc thi quốc tế Thiết kế vi mạch tích hợp cỡ lớn (LSI Design Contest 2018) tại Okinawa (Nhật Bản) và đã giành vị trí thứ hai chung cuộc. Trong tương lai, tác giả sẽ tiếp tục hoàn thiện mạng nơ-ron nhận dạng chữ số viết tay sử dụng kỹ thuật tính toán ngẫu nhiên, sau đó thực thi nhận dạng chữ số viết tay để kiểm tra hiệu năng và khả năng ứng dụng của nó.

TÀI LIỆU THAM KHẢO

[1] X. Ma, W. A. Najjar, và A. K. Roy-Chowdhury, "Evaluation and Acceleration of High-Throughput Fixed-Point Object Detection on FPGAs", *IEEE Trans. Circuits Syst. Video Technol.*, vol 25, số p.h 6, tr 1051–1062, tháng 6 2015.

[2] B. R. Gaines, "Stochastic Computing", trong *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, New York, NY, USA, 1967, tr 149–156.

[3] W. J. Poppelbaum, C. Afuso, và J. W. Esch, "Stochastic Computing Elements and Systems", trong *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*, New York, NY, USA, 1967, tr 635–644.

[4] S. T. Ribeiro, "Random-Pulse Machines", *IEEE Trans. Electron. Comput.*, vol EC-16, số p.h 3, tr 261–276, tháng 6 1967.

[5] "File:Neuron.svg - Wikimedia Commons". [Online]. Available at: https://commons.wikimedia.org/w/index.php?curid=1474927. [Truy cập: 20-tháng 4-2018].

[6] W. S. McCulloch và W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bull. Math. Biophys.*, vol 5, số p.h 4, tr 115–133, tháng 12 1943.

[7] "Approximation by superpositions of a sigmoidal function | SpringerLink". [Online]. Available at: https://link.springer.com/article/10.1007/BF02551274. [Truy cập: 17-tháng 4-2018].

[8] D. E. Rumelhart, G. E. Hinton, và R. J. Williams, "Learning representations by back-propagating errors", *Nature*, vol 323, số p.h 6088, tr 533–536, tháng 10 1986.

[9] "Recent Advances in Deep Learning for Speech Research at Microsoft - Microsoft Research". [Online]. Available at: https://www.microsoft.com/en-us/research/publication/recent-advances-in-deep-learning-for-speech-research-at-microsoft/. [Truy cập: 20-tháng 4-2018].

[10] C. Chen, A. Seff, A. Kornhauser, và J. Xiao, "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving", trong *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, Washington, DC, USA, 2015, tr 2722–2730.

[11] A. Esteva *và c.s.*, "Dermatologist-level classification of skin cancer with deep neural networks", *Nature*, vol 542, số p.h 7639, tr 115–118, 02 2017.

[12] "Mastering the game of Go with deep neural networks and tree search | Nature". [Online]. Available at: https://www.nature.com/articles/nature16961. [Truy cập: 20-tháng 4-2018].

[13] A. Krizhevsky, I. Sutskever, và G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", trong *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, USA, 2012, tr 1097–1105.

[14] K. He, X. Zhang, S. Ren, và J. Sun, “Deep Residual Learning for Image Recognition”, *ArXiv151203385 Cs*, tháng 12 2015.

[15] V. T. Lee, A. Alaghi, J. P. Hayes, V. Sathe, và L. Ceze, “Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing”, trong *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, tr 13–18.

[16] T.-H. Chen và J. P. Hayes, “Equivalence among Stochastic Logic Circuits and its Application to Synthesis”, *IEEE Trans. Emerg. Top. Comput.*, tr 1–1, 2016.

[17] M. Parhi, M. D. Riedel, và K. K. Parhi, “Effect of bit-level correlation in stochastic computing”, trong *2015 IEEE International Conference on Digital Signal Processing (DSP)*, 2015, tr 463–467.

[18] A. Alaghi, C. Li, và J. P. Hayes, “Stochastic circuits for real-time image-processing applications”, trong *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, tr 1–6.

[19] “Survey of Stochastic Computing”. [Online]. Available at: https://dl.acm.org/citation.cfm?id=2465794. [Truy cập: 16-tháng 4-2018].

[20] P. Li và D. J. Lilja, “Using stochastic computing to implement digital image processing algorithms”, trong *2011 IEEE 29th International Conference on Computer Design (ICCD)*, 2011, tr 154–161.

[21] P. Li, D. J. Lilja, W. Qian, K. Bazargan, và M. D. Riedel, “Computation on Stochastic Bit Streams Digital Image Processing Case Studies”, *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol 22, số p.h 3, tr 449–462, tháng 3 2014.

[22] B. Yuan và Y. Wang, “High-Accuracy FIR Filter Design Using Stochastic Computing”, trong *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, tr 128–133.

[23] B. D. Brown và H. C. Card, “Stochastic neural computation. I. Computational elements”, *IEEE Trans. Comput.*, vol 50, số p.h 9, tr 891–905, tháng 9 2001.

[24] B. D. Brown và H. C. Card, “Stochastic neural computation. II. Soft competitive learning”, *IEEE Trans. Comput.*, vol 50, số p.h 9, tr 906–920, tháng 9 2001.

[25] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, và W. J. Gross, “VLSI implementation of deep neural networks using integral stochastic computing”, trong *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, tr 216–220.

[26] “MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges”. [Online]. Available at: http://yann.lecun.com/exdb/mnist/. [Truy cập: 14-tháng 4-2018].

[27] Y. Lecun, L. Bottou, Y. Bengio, và P. Haffner, “Gradient-based learning applied to document recognition”, *Proc. IEEE*, vol 86, số p.h 11, tr 2278–2324, tháng 11 1998.

[28] D. Cireşan, U. Meier, và J. Schmidhuber, “Multi-column Deep Neural Networks for Image Classification”, *ArXiv12022745 Cs*, tháng 2 2012.

[29] F. Ortega-Zamorano, J. M. Jerez, D. U. Muñoz, R. M. Luque-Baena, và L. Franco, “Efficient Implementation of the Backpropagation Algorithm in FPGAs

and Microcontrollers", *IEEE Trans. Neural Netw. Learn. Syst.*, vol 27, số p.h 9, tr 1840–1850, tháng 9 2016.

[30] P. K. Meher, "An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks", trong *2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip*, 2010, tr 91–95.

[31] "hkinks / thesis". [Online]. Available at: https://bitbucket.org/hkinks/thesis. [Truy cập: 17-tháng 4-2018].

[32] "Caffe | Deep Learning Framework". [Online]. Available at: http://caffe.berkeleyvision.org/. [Truy cập: 13-tháng 4-2018].