ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

LƯƠNG VĂN ĐẠI

KHÓA LUẬN TỐT NGHIỆP

ĐÁNH GIÁ HIỆU QUẢ THIẾT KẾ CNN TRÊN HỆ THỐNG SOC QUA PHƯƠNG PHÁP TỰ THIẾT KẾ VÀ HLS

EVALUATING EFFICIENCY OF CNN DESIGN ON SOC SYSTEM THROUGH SELF-DESIGN AND HLS METHODS

CỬ NHÂN KỸ THUẬT MÁY TÍNH

TP. HỒ CHÍ MINH, 2025

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

LƯƠNG VĂN ĐẠI – 21521913

KHÓA LUẬN TỐT NGHIỆP

ĐÁNH GIÁ HIỆU QUẢ THIẾT KẾ CNN TRÊN HỆ THỐNG SOC QUA PHƯƠNG PHÁP TỰ THIẾT KẾ VÀ HLS

EVALUATING EFFICIENCY OF CNN DESIGN ON SOC SYSTEM THROUGH SELF-DESIGN AND HLS METHODS

CỬ NHÂN KỸ THUẬT MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN
TS. TRẦN THỊ ĐIỂM

TP. HỒ CHÍ MINH, 2025

THÔNG TIN HỘI ĐỒNG CHẤM KHÓA LUẬN TỐT NGHIỆP

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số ……
ngày …….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

LỜI CẢM ƠN

Hành trình nghiên cứu và hoàn thành luận văn tốt nghiệp không chỉ là chuỗi ngày miệt mài với sách vở, số liệu và trang viết, mà còn là những khoảnh khắc được tiếp thêm năng lượng từ những tấm lòng tận tâm, những lời khích lệ chân thành. Trong không khí hân hoan của ngày hoàn thành chặng đường học tập, em muốn gửi lời tri ân sâu sắc đến tất cả những ai đã chung sức, đồng hành cùng em.

Trước hết, em xin dành trọn vẹn sự biết ơn đến Tiến sĩ Trần Thị Điểm, người hướng dẫn đã dìu dắt em bằng nhiều buổi trao đổi thấu đáo, những góp ý tỉ mỉ và không ngừng khích lệ. Chính sự tận tâm trong giảng dạy và niềm đam mê nghiên cứu của cô đã thắp lên ngọn lửa đam mê trong em, giúp em tự tin vượt qua thử thách và hoàn thiện đề tài này.

Em xin gửi lời cảm ơn chân thành đến Khoa Kỹ thuật Máy tính và Trường Đại học Công Nghệ Thông Tin – ĐHQG TP.HCM, nơi đã tạo dựng môi trường học tập năng động, trang thiết bị hiện đại và chương trình đào tạo chuyên sâu. Tất cả những điều kiện ấy đã góp phần quan trọng để em mở rộng kiến thức, rèn luyện kỹ năng và phát huy khả năng nghiên cứu.

Em cũng xin tri ân các thầy cô trong Hội đồng chấm luận văn đã dành thời gian đọc kỹ, phân tích và đưa ra những nhận xét khách quan, xây dựng. Mỗi ý kiến quý báu của quý thầy cô là kim chỉ nam giúp em nhìn nhận lại công trình, hoàn thiện hơn và có động lực tiếp tục nghiên cứu trong tương lai.

Cuối cùng, em không thể không nói lời cảm ơn đến gia đình và bạn bè – những người luôn ở bên, chia sẻ niềm vui lẫn khó khăn. Sự ủng hộ về tinh thần lẫn vật chất của mọi người là động lực vô giá, giúp em vững tin bước tiếp trên con đường học tập và nghiên cứu.

Dù đã cố gắng hết mình, luận văn vẫn còn những hạn chế nhất định. Em rất mong nhận được thêm ý kiến đóng góp để hoàn thiện hơn ở những công trình kế tiếp.

Trân trọng cảm ơn!

TP. Hồ Chí Minh, ngày 13, tháng 6, năm 2025

Lương Văn Đại

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

KLTN	Khóa luận tốt nghiệp
GVHD	Giảng viên hướng dẫn
CNN	Convolutional Neural Network
SoC	System on Chip
ASIC	Application-specific integrated circuit
FPGA	Field-programmable gate array
HLS	High-Level Synthesis
PE	Processing Element
LDM	Local Data Memory
LSU	Load/Store Unit
ALU	Arithmetic Logic Unit
MAC	Multiply-Accumulate

TÓM TẮT KHÓA LUẬN

Mạng nơ-ron tích chập (CNN) là một trong những kiến trúc học sâu nổi bật, được thiết kế để xử lý dữ liệu có cấu trúc lưới như ảnh hoặc chuỗi tín hiệu. Nhờ khả năng học đặc trưng tự động thông qua các lớp tích chập và trích xuất đặc trưng đa cấp, CNN đã đạt được những bước tiến vượt bậc trong nhiều bài toán thị giác máy tính, nhận dạng mẫu, và phân loại tín hiệu. Sự phát triển mạnh mẽ của phần cứng và nền tảng phần mềm đã góp phần đưa CNN từ nghiên cứu lý thuyết vào thực tiễn trong nhiều lĩnh vực công nghiệp và dân dụng..

Bên cạnh các ứng dụng về phân loại và xác định vật thể, CNN đang được tích hợp vào các thiết bị y tế thông minh như máy theo dõi tim đeo tay, thiết bị chăm sóc sức khỏe tại nhà, và các hệ thống chẩn đoán sớm. Trong đó, tín hiệu ECG – một chỉ số quan trọng phản ánh hoạt động điện sinh lý của tim – được xem là ứng dụng điển hình cho việc áp dụng CNN 1D trong phát hiện các rối loạn nhịp tim, nhằm cải thiện độ hiệu quả trong chẩn đoán bệnh sớm và từ đó có những trị liệu hiệu quả ở những giai đoạn đầu. Các bệnh về tim mạch đang được dần trở nên trẻ hóa do đó, việc có thể xác định các rối loạn về tim mạch từ sớm sẽ mang lại hiệu quả chữa trị cao. Tuy nhiên việc triển khai CNN trên các thiết bị y tế đang gặp nhiều thách thức, do yêu cầu về kích thước nhỏ, tiêu thụ năng lượng thấp, và thời gian phản hồi nhanh. Để giải quyết vấn đề này, các hệ thống tích hợp trên chip (System-on-Chip – SoC), đặc biệt là các SoC có tích hợp FPGA, là giải pháp hứa hẹn giúp cân bằng giữa hiệu năng và chi phí phần cứng.

Trong khuôn khổ luận văn này, hai phương pháp tiếp cận chính được lựa chọn để triển khai CNN trên SoC là phương pháp tổng hợp cấp cao (HLS) trong đó mô hình CNN được thiết kế ở cấp độ ngôn ngữ C/C+ sẽ được tự động chuyển đổi sang phần cứng, giúp giảm thời gian phát triển, tăng khả năng linh hoạt và phương pháp sử dụng Verilog để tự thiết kế phần cứng cho phép tối ưu hóa luồng dữ liệu, băng thông bộ nhớ và hiệu suất tính toán.

Các phương pháp được đề xuất trong bài đã hoạt động đúng như chức năng của nó giúp cải thiện về cả phần cứng và sai số của mô hình.

- Chương 1 Mở đầu
- Chương 2 Tổng quan
- Chương 3 Cơ sở lý thuyết
- Chương 4 Kiến trúc mạng CNN
- Chương 5 Quá trình thực hiện

Chương 1. **MỞ ĐẦU**

1.1. Lí do chọn đề tài

Trong thời đại hiện nay, công nghệ máy học (machine learning) đang phát triển mạnh mẽ và ngày càng khẳng định vai trò quan trọng trong nhiều lĩnh vực, từ phân tích các dữ liệu y tế [1-5], truy xuất thông tin [6], cho đến các hệ thống khuyến nghị [7-9]. Tuy nhiên, một hạn chế lớn của nhiều kỹ thuật học máy truyền thống là phụ thuộc vào quá trình trích xuất đặc trưng thủ công, tốn nhiều thời gian và công sức. Sự ra đời của học sâu (Deep Learning) đã tạo ra bước ngoặt mới nhờ khả năng học trực tiếp đặc trưng từ dữ liệu thông qua các tầng biểu diễn đa cấp. Kiến trúc này cho phép mô hình học được những đặc trưng phức tạp mà không cần sự can thiệp của con người, từ đó cải thiện đáng kể hiệu suất và độ chính xác trong nhiều bài toán [10-11].

Theo các nghiên cứu so sánh về độ hiệu quả giữa các kiến trúc mạng học sâu [12-14], mạng nơ-ron tích chập (CNN) được đánh giá là một trong những kiến trúc học sâu hiệu quả trong việc trích xuất đặc trưng không gian cục bộ, đặc biệt phù hợp với các bài toán nhận dạng mẫu trong tín hiệu sinh lý như ECG. So với các mô hình tuần tự như RNN hoặc LSTM, CNN cho phép huấn luyện nhanh hơn, tận dụng tính song song của phần cứng và đạt độ chính xác cao trong các tác vụ phân loại khi đặc trưng đầu vào mang tính cấu trúc rõ ràng. Trong khi đó, các mô hình như RNN/LSTM lại phù hợp hơn với việc học quan hệ thời gian dài hạn, nhưng thường gặp khó khăn về hiệu năng tính toán và hội tụ.

Tuy nhiên, một thách thức lớn khi triển khai các mô hình CNN trong thực tế là yêu cầu phần cứng lớn, tiêu thụ nhiều tài nguyên tính toán và năng lượng. Trong bối cảnh các thiết bị y tế đang ngày càng hướng tới tính di động, chi phí thấp và tiết kiệm năng lượng, việc triển khai CNN trên nền tảng nhúng, đặc biệt là SoC tích hợp cả CPU và FPGA, đang trở thành hướng tiếp cận tiềm năng [15-16]. Xuất phát từ thực tế đó, đề tài đã tham khảo [17-19, 23-29] và lựa chọn hai phương pháp triển khai mô hình CNN 1D cho bài toán phân loại tín hiệu ECG trên SoC là phương pháp tự thiết kế phần cứng và phương pháp sử dụng tổng hợp bậc cao (HLS). Việc đánh giá và so sánh hai

phương pháp về mặt độ chính xác, độ trễ, tiêu thụ tài nguyên và khả năng mở rộng sẽ mang lại cái nhìn tổng quan, góp phần xác định phương pháp tối ưu để triển khai CNN trong các thiết bị nhúng y tế hiện đại.

Theo thống kê từ Tổ chức Y tế Thế giới (WHO), phiếu thông tin tóm tắt [20] năm 2021, bệnh tim mạch là nguyên nhân gây tử vong hàng đầu hiện nay theo, với xu hướng ngày càng trẻ hóa với người dưới 25 tuổi [21]. Tín hiệu điện tâm đồ (ECG) là một công cụ chẩn đoán không xâm lấn, dễ thu thập và mang nhiều thông tin có giá trị trong đánh giá chức năng tim, trong đó, phức hợp QRS là đặc trưng then chốt, phản ánh hoạt động khử cực của tâm thất, từ đó hỗ trợ phát hiện sớm các bất thường như rung thất, nhồi máu cơ tim, hay rối loạn nhịp tim [22]. Việc áp dụng CNN 1D để tự động phân loại tín hiệu ECG sẽ giúp nâng cao độ chính xác và giảm gánh nặng cho các hệ thống y tế.

1.2. Mục đích

Theo các nghiên cứu đã được thực hiện trước đó, trong lĩnh vực xử lý ảnh, nghiên cứu [23] đã tiến hành so sánh giữa Vivado HLS và thiết kế RTL thủ công, cho thấy RTL có ưu thế rõ rệt về tần số hoạt động và mức sử dụng tài nguyên, trong khi HLS lại mang lại lợi ích lớn về mặt thời gian và độ phức tạp phát triển. Tương tự, nghiên cứu [24] khi triển khai thuật toán Sobel trên SoC cũng chỉ ra rằng thiết kế bằng RTL đạt mức tối ưu cao về diện tích và công suất, còn HLS nổi bật ở khả năng tăng tốc độ phát triển.

Tương tự, nghiên cứu [25] đánh giá thiết kế bộ lọc 9×9 bằng hai phương pháp này và nhận thấy rằng tuy HLS cho hiệu quả phát triển tốt hơn, nhưng kết quả tổng hợp ban đầu không đạt yêu cầu về timing và yêu cầu chỉnh sửa mã thủ công để tối ưu.

Trong môi trường thực tế, nghiên cứu [26] tại CERN đã triển khai khối switch matrix bằng cả RTL và HLS trên FPGA cho hệ thống đọc dữ liệu LAr, từ đó cho thấy HLS có thể tiếp cận được mức hiệu năng gần với RTL nếu được cấu hình cẩn thận, tuy nhiên vẫn tồn tại độ chênh lệch về độ trễ và logic sử dụng.

Ngoài ra, trong lĩnh vực xử lý ma trận–vector, nghiên cứu [27] cho thấy phiên bản RTL sử dụng ít hơn 50% tài nguyên so với bản HLS, tuy nhiên mức độ linh hoạt và khả năng bảo trì dài hạn của HLS vẫn là ưu điểm lớn.

Một khảo sát tổng quan từ [28] cho thấy HLS đã phát triển đáng kể và có thể đạt hiệu năng gần với RTL trong nhiều ứng dụng, với điều kiện có kinh nghiệm và cấu hình hợp lý.

Gần đây, nghiên cứu [29] khẳng định rằng HLS ngày càng được ứng dụng rộng rãi nhờ mức độ trừu tượng cao, hỗ trợ tái sử dụng và bảo trì tốt hơn, tuy nhiên vẫn tồn tại khoảng cách nhỏ về hiệu quả phần cứng so với RTL.

Từ các nghiên cứu nêu trên, có thể thấy rằng việc lựa chọn giữa phương pháp HLS và thiết kế RTL phụ thuộc vào bài toán cụ thể, yêu cầu mục tiêu tối ưu hóa riêng. Tuy nhiên, phần lớn các nghiên cứu tập trung vào các thuật toán xử lý ảnh hoặc xử lý tín hiệu đơn lẻ.

Vì vậy, mục đích của khóa luận này là đánh giá hiệu quả thiết kế mạng nơ-ron tích chập (CNN) trên nền tảng SoC, bằng cách triển khai hai phương pháp: sử dụng High-Level Synthesis (HLS) và tự thiết kế RTL truyền thống, từ đó so sánh chi tiết về các khía cạnh hiệu năng, độ trễ, tài nguyên phần cứng sử dụng, và năng suất thiết kế.

1.3. Đối tượng và phạm vi nghiên cứu

Đề tài tập trung vào mô hình mạng nơ-ron tích chập một chiều (1-D CNN) nhằm giải quyết bài toán phân loại tín hiệu điện tâm đồ (ECG), hướng đến ứng dụng trong các hệ thống y sinh nhúng. Mô hình được huấn luyện và đánh giá trên tập dữ liệu MIT-BIH, một tập dữ liệu phổ biến trong nghiên cứu phát hiện rối loạn nhịp tim. Hai phương pháp hiện thực phần cứng được áp dụng trong đề tài bao gồm thiết kế thủ công và thiết kế sử dụng công cụ tổng hợp cấp cao Vitis HLS. Việc áp dụng song song hai phương pháp giúp so sánh và đánh giá sự khác biệt về hiệu quả phần cứng như diện tích sử dụng, độ trễ, tần số hoạt động và mức tiêu thụ năng lượng. Cả hai phương pháp đều được triển khai trên nền tảng phần cứng là board Xilinx Kria

KV260. Công cụ Vivado được sử dụng để triển khai và phân tích thiết kế trên FPGA, giúp đánh giá hiệu quả thực thi trong môi trường thực tế. Trong khi đó, Vitis HLS hỗ trợ việc tạo ra thiết kế phần cứng từ mã C/C++, từ đó rút ngắn quá trình phát triển phần cứng. Thông qua việc triển khai mô hình CNN 1D trên nền tảng KV260 theo hai hướng tiếp cận khác nhau, đề tài hướng đến việc đánh giá toàn diện khả năng hiện thực hóa các mô hình học sâu trên SoC, từ đó đưa ra nhận định phù hợp cho việc tích hợp vào các thiết bị y tế có yêu cầu khắt khe về tài nguyên và năng lượng.

Chương 2. **TỔNG QUAN**

Chương 2 sẽ mô tả các hướng nghiên cứu trên thế giới cũng như đánh giá về các nghiên cứu đã được thực hiện ngày nay. Những điều này sẽ giúp có những cái nhìn tổng quan hơn về đề tài cũng như những cải tiến đã giúp cho việc tối ưu về diện tích và năng lượng.

2.1. Các hướng nghiên cứu hiện nay và đánh giá

Hiện nay, trên thế giới đã có các nghiên cứu về việc phân loại tín hiệu ECG tập trung vào việc nâng cao độ chính xác chẩn đoán, phát hiện sớm rối loạn nhịp tim và triển khai hiệu quả trên thiết bị nhúng. Các phương pháp học máy truyền thống như SVM [30], k-NN [31] hay Random Forest [32] thường kết hợp với kỹ thuật trích xuất đặc trưng như QRS complex, khoảng RR, sóng P và T. Tuy nhiên, chúng phụ thuộc vào tiền xử lý thủ công nên khả năng tổng quát hóa còn hạn chế.

Mạng nơ-ron tích chập (CNN) đang trở thành giải pháp nổi bật nhờ khả năng tự học đặc trưng trực tiếp từ tín hiệu đầu vào mà không cần bước trích xuất đặc trưng thủ công. CNN học được các mẫu thời gian trong tín hiệu ECG và thường cho kết quả phân loại chính xác, ổn định hơn. Ngoài ra, CNN có thể được tối ưu hóa để triển khai trên FPGA hoặc SoC, đáp ứng tốt yêu cầu về hiệu năng, tiêu thụ năng lượng thấp và thời gian thực, phù hợp cho thiết bị đeo tay và hệ thống giám sát sức khỏe từ xa [1-5]. Trong đó, kiến trúc mạng được sử dụng phổ biến cho việc phân loại là 1D-CNN (One-Dimensional Convolutional Neural Network), đây là kiến trúc CNN nhỏ và tiết kiệm tài nguyên hơn so với 2D-CNN và được sử dụng hiệu quả cho các bài toán phân loại tín hiệu hoặc chuỗi các dữ liệu liên tục, đặc biệt cho việc phát hiện và phân loại tín hiệu điện tâm đồ (ECG) nhằm ứng dụng vào phát triển các thiết bị y tế di động [1-5]. Tuy nhiên, thiết kế của các thiết bị y tế di động phải đối mặt với thách thức về tài nguyên phần cứng hạn chế và yêu cầu độ chính xác cao. Vì vậy, nhiều nghiên cứu đã được đưa ra với mục tiêu nhằm tối ưu hiệu quả tài nguyên phần cứng và hiện thực trên FPGA.

Bài báo [1] đề xuất kiến trúc phần cứng cho mô hình 1D U-Net với 22 lớp tích chập, tập trung vào phân loại ECG theo mức điểm ảnh. Kiến trúc sử dụng cấu trúc pipeline Winograd hai giai đoạn nhằm tối ưu hóa hiệu quả tính toán và sử dụng bộ nhớ. Kết quả được thực hiện trên FPGA đạt độ chính xác 95,55% cho phân loại ECG.

Bài báo [2] đề xuất một kiến trúc phần cứng dựa trên mô hình 1D CNN với việc sử dụng lớp Global Average Pooling (GAP). Kiến trúc này sử dụng các khối Processing Unit (PU) cùng với thiết kế fully pipeline để tăng hiệu quả tính toán. Kết quả được thực hiện trên FPGA, đạt độ chính xác lên tới 99,1%.

Bài báo [3] đề xuất kiến trúc phần cứng dựa trên 1D CNN với mục tiêu hỗ trợ phân loại ECG đa dạng trong thiết bị y tế di động. Kiến trúc đề xuất khối tính toán sử dụng mảng systolic có khả năng tùy chỉnh với các đơn vị xử lý có cấu hình và thanh ghi lệnh có thể điều chỉnh, nhằm giảm tiêu thụ điện năng và tăng khả năng tái sử dụng dữ liệu. Kết quả hiện thực trên FPGA với độ chính xác cao (98,9% và 99,3%)

Bài báo [4] đề xuất một bộ gia tốc phần cứng tối ưu hóa cho mô hình 1D CNN trong việc phân loại tín hiệu ECG, EEG và EMG trên thiết bị y tế đeo được. Kiến trúc sử dụng pipeline với thanh ghi trung gian để tăng tốc độ truyền dữ liệu và thay thế phép nhân bằng dịch bit nhằm giảm tiêu thụ tài nguyên phần cứng. Hệ thống được triển khai trên FPGA Xilinx Zynq xc7z045, đạt hiệu suất 1.145 TFLOP/s, hiệu suất năng lượng 161 GOP/s/W và hiệu suất tài nguyên 28 GOP/s/KLUT, cải thiện 1.67 lần so với các mô hình trước đó.

Bài báo [5] đề xuất kiến trúc phần cứng cho mô hình Mini InceptionNet, một dạng 1D CNN với các khối Inception song song nhằm tối ưu số lượng tham số và độ chính xác phân loại ECG. Kiến trúc phần cứng sử dụng PEA (Processing Element Array) linh hoạt kết hợp với Sharing Buffer Allocator (SBA) cho phép hỗ trợ đa dạng kích thước kernel, stride và số kênh. Mỗi PE được trang bị bốn bộ nhớ nội (LDM) giúp lưu trữ dữ liệu trung gian như residual và nhánh song song, giảm thiểu truy xuất bộ nhớ ngoài. Toàn bộ hệ thống được thiết kế theo kiến trúc fully pipeline ba giai đoạn (load → allocate → compute/store) nhằm giảm độ trễ và tăng hiệu suất xử lý. MINA

được triển khai trên FPGA Xilinx ZCU102 với 40 PEs, đạt độ chính xác 99,37% trên tập dữ liệu MIT-BIH và cải thiện 1.53 lần về chỉ số area-delay product (ADP) so với các mô hình trước. Ngoài ra, kiến trúc còn hỗ trợ pruning trọng số, giúp giảm thêm thời gian suy luận và tài nguyên sử dụng.

Trong đó kiến trúc 1D-CNN từ bài báo [2] khá nổi bật vì đơn giản, có cấu trúc rõ ràng kèm theo độ chính xác rất cao, đây là một mô hình lý tưởng để thực hiện việc so sánh giữa hai phương pháp. Mặt khác kiến trúc phần cứng của bài báo này sử dụng một mảng PE để thực hiện Convolutional theo phương pháp Systolic Array. Kiến trúc này sẽ có tốc độ khá tốt ở những lớp Convolutional đầu khi mà kích thước dữ liệu mỗi channel còn lớn. Sau khi qua một số lượng lớp Convolutional, kích thước dữ liệu mỗi channel sẽ nhỏ đi làm cho một số lượng PE sẽ không được sử dụng trong khi lượng dữ liệu cần xử lý còn khá nhiều kèm theo việc chỉ sử dụng hai bộ nhớ để lưu dữ liệu đầu vào và trọng số làm cho thiết kế này khá chậm trong việc lấy dữ liệu cho các PE xử lý và không tối ưu được nhiều băng thông. Ngoài ra thiết kế này sử dụng kernel size cố định làm cho thiết kế khó tái sử dụng hoặc phải thiết kế lại nếu kiến trúc mạng CNN ban đầu có thay đổi. Ngược lại với đó kiến trúc phần cứng ở bài báo [5] tỏ ra rất hiệu quả khi sử dụng một mảng PE với các khối ALU và bộ đệm riêng biệt. Các PE sẽ được phân bổ dữ liệu một cách đều đặn bằng một khối Shared Global Allocator theo thủ thuật Round Robin giúp các PE sẽ được nhận và xử lý một số lượng dữ liệu bằng nhau giúp tăng tốc độ xử lý và tránh tắc nghẽn băng thông.

Bằng cách dùng mô hình CNN ở bài báo [2] và kiến trúc [5] làm mô hình thiết kế thủ công, luận văn sẽ có cơ sở rõ ràng để so sánh ưu–nhược điểm hai phương pháp trên cùng một bài toán phân loại ECG: từ đó rút ra kết luận về mức độ "hiệu quả" (efficiency) của HLS so với tự thiết kế.

2.2. Những nội dung đề tài tập trung

Từ việc phân tích và đánh giá các công trình trước, đề tài sẽ tập trung vào việc triển khai kiến trúc CNN đơn giản theo bài báo [2] bằng phương pháp High-Level

Synthesis (HLS). Thông qua HLS, các vòng lặp convolution và Global Average Pooling sẽ được tự động pipeline và unroll, giúp rút ngắn thời gian phát triển phần cứng, đồng thời dễ dàng tinh chỉnh tham số. Kết quả thu được sẽ được đánh giá chi tiết về độ trễ (latency), thông lượng (throughput), mức độ sử dụng tài nguyên (LUT, FF, DSP) và tiêu thụ năng lượng, từ đó làm rõ ưu điểm và hạn chế của HLS khi áp dụng cho mô hình CNN trên SoC.

Sau đó, luận văn sẽ tiến hành tự thiết kế RTL thủ công dựa trên kiến trúc tham khảo từ bài báo [5] với Processing Element Array (PEA) linh hoạt, Sharing Buffer Allocator và bốn Local Data Memories cho mỗi PE. Thiết kế này sẽ tập trung tối ưu hóa băng thông nội bộ, chồng lấn fully-pipeline giữa các giai đoạn load, allocate và compute/store, giúp giảm số chu kỳ xử lý và thu hẹp area-delay product. Kết quả thực nghiệm tự thiết kế sẽ chỉ ra hiệu quả của việc tự viết mã RTL so với kết quả HLS.

Cuối cùng, luận văn sẽ thảo luận khả năng mở rộng và ứng dụng thực tế của hai phương pháp thiết kế trên SoC cho các thiết bị giám sát sức khỏe cầm tay. Các khuyến nghị sẽ xoay quanh yếu tố thời gian đưa sản phẩm ra thị trường, chi phí phát triển và mức độ linh hoạt khi tích hợp thêm các chức năng mới, từ đó giúp xác định phương pháp thiết kế phù hợp nhất cho từng yêu cầu cụ thể.

Chương 3. CƠ SỞ LÝ THUYẾT

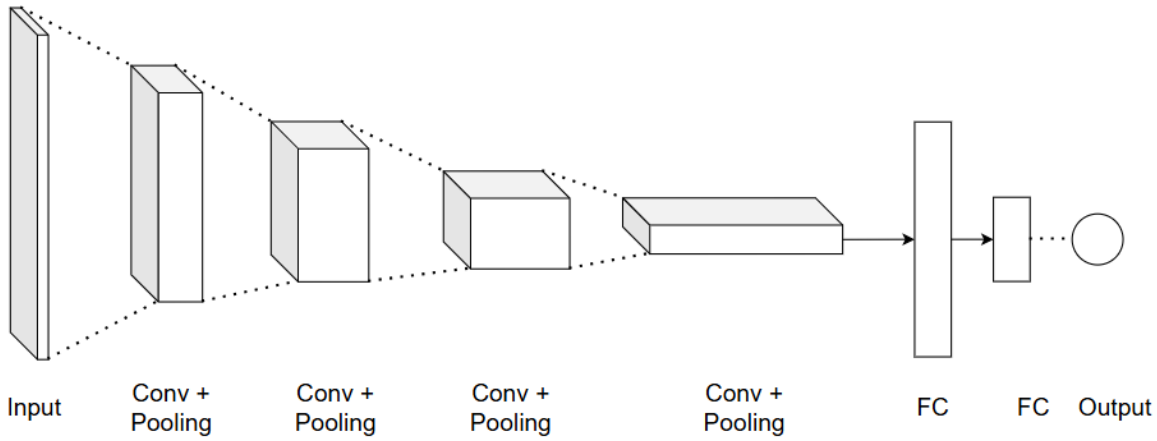
3.1. Mạng nơ ron tích chập là gì

Mạng nơ-ron tích chập (CNN) là một kiến trúc học sâu nổi bật, được thiết kế để xử lý dữ liệu có cấu trúc dạng lưới như hình ảnh, lấy cảm hứng từ cơ chế hoạt động của vỏ não thị giác ở sinh vật. CNN hoạt động dựa trên nguyên tắc trích xuất đặc trưng tự động thông qua nhiều tầng, từ những đặc trưng đơn giản đến phức tạp. Cấu trúc cơ bản của một mạng CNN bao gồm ba loại lớp chính: lớp tích chập (convolutional), lớp gộp (pooling) và lớp kết nối đầy đủ (fully connected). Trong đó, hai lớp đầu tiên đảm nhiệm vai trò trích xuất đặc trưng từ dữ liệu đầu vào, còn lớp cuối cùng có nhiệm vụ tổng hợp và suy diễn các đặc trưng đó để đưa ra kết quả đầu ra, chẳng hạn như phân loại. Trong các lớp tích chập, việc tính toán được thực hiện bởi các kernel – các bộ lọc nhỏ di chuyển trên dữ liệu đầu vào (ví dụ như ảnh), thực hiện phép nhân chập để phát hiện các mẫu đặc trưng. Mỗi pixel trong ảnh được biểu diễn bằng một số trong ma trận hai chiều, và kernel sẽ quét qua toàn bộ ảnh, tìm ra các đặc điểm có tính lặp lại, bất kể vị trí xuất hiện. Cơ chế này giúp CNN đặc biệt hiệu quả trong các tác vụ xử lý hình ảnh. Sau mỗi lớp, đầu ra lại trở thành đầu vào cho lớp tiếp theo, tạo thành một chuỗi các đặc trưng ngày càng sâu và có ý nghĩa hơn. Việc huấn luyện mạng CNN được thực hiện bằng cách điều chỉnh các tham số của kernel thông qua thuật toán lan truyền ngược (backpropagation), giúp giảm sai số giữa đầu ra mô hình và nhãn đúng của dữ liệu huấn luyện.

Ngoài các ứng dụng phổ biến trong thị giác máy tính, CNN còn được ứng dụng rộng rãi trong lĩnh vực y sinh. Cụ thể, trong khóa luận tốt nghiệp này, CNN được sử dụng để phân loại tín hiệu ECG – một bước quan trọng trong chẩn đoán các vấn đề tim mạch. Việc ứng dụng CNN không chỉ nâng cao độ chính xác mà còn giúp đẩy nhanh tốc độ phân tích, hỗ trợ bác sĩ trong việc phát hiện sớm bệnh lý và xây dựng phác đồ điều trị hiệu quả hơn.

3.2. Cấu trúc mạng nơ-ron tích chập

Hình 3.1 minh họa cấu trúc của một mạng nơ-ron tích chập, bao gồm các lớp cơ bản như lớp tích chập (convolution), lớp gộp (pooling) và lớp kết nối đầy đủ (fully-connected). Trong một kiến trúc CNN tiêu chuẩn, các lớp convolution và pooling thường được sắp xếp xen kẽ và lặp lại nhiều lần nhằm trích xuất các đặc trưng từ dữ liệu đầu vào. Sau các lớp này, các lớp fully-connected sẽ đảm nhiệm vai trò tổng hợp và phân loại. Toàn bộ quá trình xử lý dữ liệu từ đầu vào qua từng lớp mạng đến khi tạo ra đầu ra cuối cùng được gọi là quá trình lan truyền tiến (forward propagation).



Hình 3.1. Cấu trúc của mạng nơ-ron tích chập

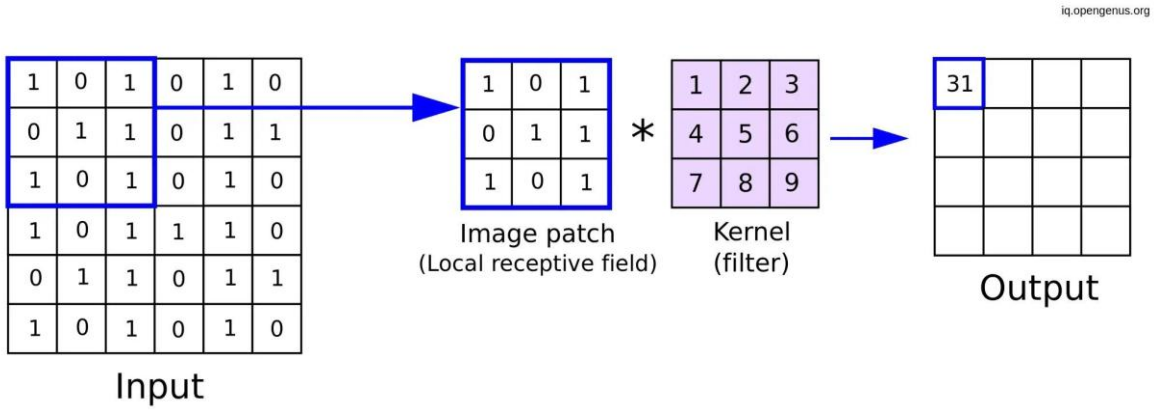
3.2.1. Lớp tích chập

Lớp tích chập là một thành phần cốt lõi trong kiến trúc mạng CNN, đóng vai trò chính trong việc trích xuất đặc trưng từ dữ liệu đầu vào. Quá trình này được thực hiện thông qua sự kết hợp giữa các phép biến đổi tuyến tính (như phép nhân chập với kernel) và phi tuyến tính (thông qua các hàm kích hoạt như ReLU). Kết quả đầu ra của lớp này là các đặc trưng quan trọng đã được rút trích, làm cơ sở cho các lớp tiếp theo xử lý và phân loại.

3.2.2. Tích chập

Tích chập là một phép biến đổi tuyến tính quan trọng trong quá trình trích xuất đặc trưng của mạng CNN. Trong phép toán này, một mảng giá trị gọi là kernel sẽ được

áp dụng lên tensor đầu vào – tức là mảng các giá trị đại diện cho dữ liệu, chẳng hạn như hình ảnh. Tại mỗi vị trí, các giá trị của kernel sẽ nhân với các giá trị tương ứng trong vùng ảnh, sau đó cộng lại để tạo thành một giá trị duy nhất. Giá trị này sẽ là phần tử trong đầu ra – được gọi là feature map (bản đồ đặc trưng). Hình 3.2 minh họa trực quan cách thức thực hiện phép tích chập này. Quá trình này sẽ được lặp lại nhiều lần với các kernel khác nhau để sinh ra nhiều feature map, mỗi bản đồ đặc trưng thể hiện một kiểu đặc trưng riêng biệt của dữ liệu đầu vào. Chính nhờ khả năng trích xuất đa dạng các đặc trưng mà kernel được xem như những "bộ lọc đặc trưng". Trong quá trình thiết kế mạng CNN, có hai siêu tham số (hyperparameters) quan trọng xác định hoạt động của lớp tích chập: kích thước kernel và số lượng kernel. Kích thước kernel phổ biến thường là 3×3, nhưng cũng có thể là 5×5 hoặc 7×7 tùy theo yêu cầu của bài toán và độ phức tạp cần thiết. Số lượng kernel càng nhiều thì càng trích xuất được nhiều đặc trưng, từ đó cải thiện độ chính xác của mô hình. Tuy nhiên, điều này cũng kéo theo sự gia tăng về số lượng tham số, độ phức tạp của mạng và thời gian huấn luyện. Vì vậy, việc lựa chọn kích thước và số lượng kernel cần được cân nhắc kỹ lưỡng để đạt được sự cân bằng giữa độ chính xác và hiệu năng xử lý.



Hình 3.2. Phép tích chập

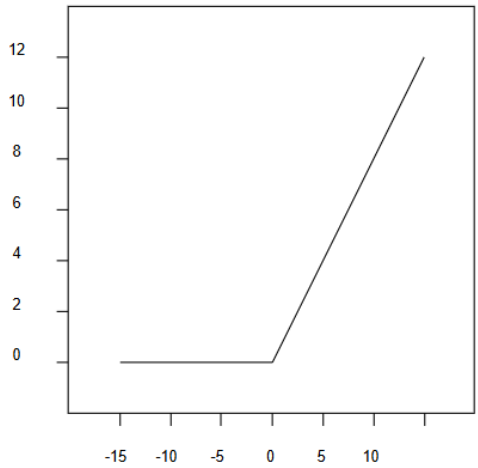
Quá trình tích chập được mô tả ở phần trước có một hạn chế là không thể bao phủ hoàn toàn các phần tử nằm ở rìa của tensor đầu vào. Kết quả là kích thước của tensor đầu ra sau mỗi lớp tích chập sẽ nhỏ hơn so với đầu vào do mất đi các giá trị biên. Để khắc phục điều này, kỹ thuật padding đã được đưa ra – thường là zero padding, tức

là thêm các giá trị 0 vào xung quanh viền của tensor đầu vào. Việc padding đặc biệt quan trọng khi sử dụng các kernel có kích thước lớn, bởi nếu không thêm giá trị đệm ở viền, phần lớn các vùng biên sẽ không được tính đến hoặc giá trị đầu ra sẽ bị lệch khỏi tâm của kernel. Padding giúp đảm bảo rằng mỗi vị trí trong tensor đầu vào, bao gồm cả các điểm rìa, đều có cơ hội được học đặc trưng thông qua quá trình tích chập.

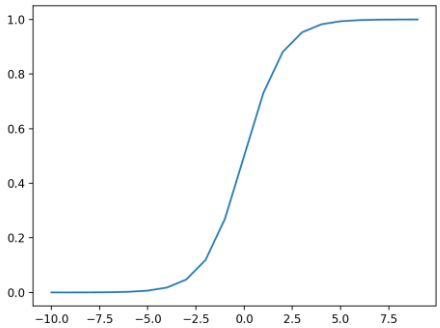
Một lợi ích quan trọng khác của padding là nó giúp giữ nguyên kích thước đầu vào và đầu ra của tensor sau mỗi lớp tích chập. Điều này rất hữu ích trong việc thiết kế các mô hình CNN hiện đại, nơi mà chiều dài hoặc chiều rộng dữ liệu cần được bảo toàn để phù hợp với kiến trúc sâu nhiều lớp. Vì lý do đó, zero padding hiện nay là một kỹ thuật phổ biến và gần như mặc định trong các mạng CNN để duy trì tính toàn vẹn của dữ liệu qua các tầng mạng.

3.2.3. Hàm kích hoạt

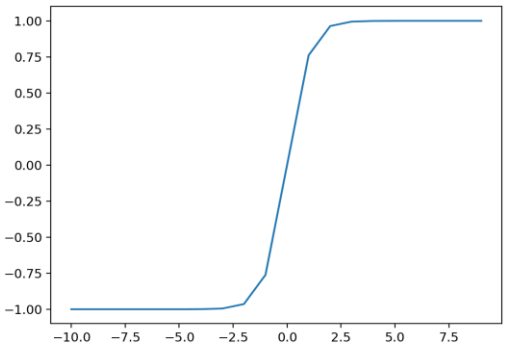
Sau khi thực hiện các phép tính tuyến tính như tích chập, đầu ra sẽ được đưa qua các hàm kích hoạt phi tuyến tính nhằm tăng khả năng học các đặc trưng phức tạp của mô hình. Trước đây, hai hàm phi tuyến phổ biến được sử dụng rộng rãi là sigmoid và tanh (hyperbolic tangent) – lần lượt được minh họa ở Hình 3.4 và Hình 3.5. Các hàm này có thể được xem như những mô hình toán học mô phỏng hành vi kích hoạt của các nơ-ron sinh học. Tuy nhiên, trong các mô hình học sâu hiện đại, hàm ReLU (Rectified Linear Unit) – được mô tả ở Hình 3.3 – đã trở thành lựa chọn mặc định trong hầu hết các kiến trúc mạng. ReLU đơn giản nhưng hiệu quả, với đặc điểm chỉ giữ lại các giá trị dương và triệt tiêu các giá trị âm. Nhờ tính chất này, ReLU không chỉ giúp quá trình huấn luyện diễn ra nhanh hơn mà còn giảm đáng kể độ phức tạp tính toán so với sigmoid hoặc tanh, đồng thời hạn chế hiện tượng gradient biến mất trong các mạng sâu. Chính vì vậy, ReLU đã trở thành hàm kích hoạt tiêu chuẩn trong đa số các mạng CNN hiện nay.



Hình 3.3 Hàm ReLU



Hình 3.4. Hàm sigmoid



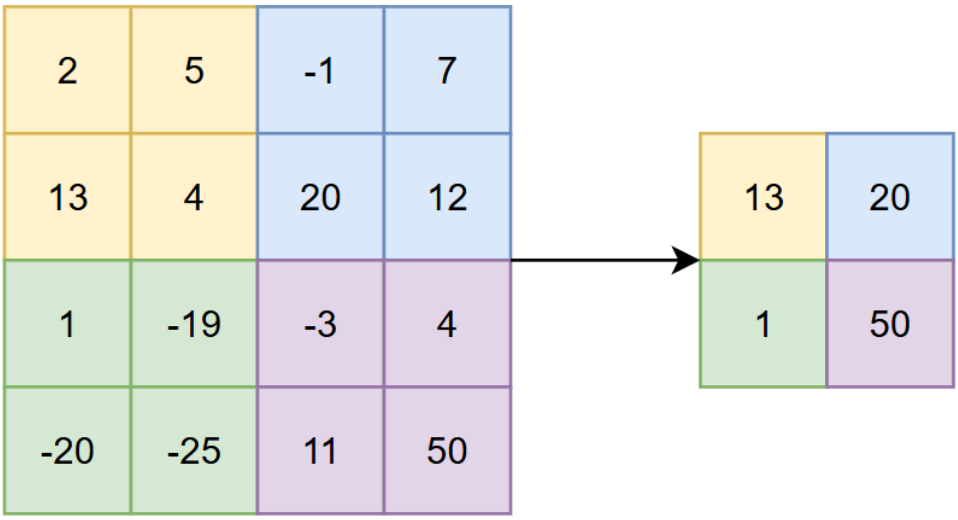
Hình 3.5. Hàm Tanh

3.2.4. Lớp gộp - Pooling Layer

Lớp pooling là một phép biến đổi có tính chất downsampling, được sử dụng phổ biến trong các mạng CNN nhằm giảm kích thước không gian của các feature map đầu vào. Mục tiêu chính của lớp này là làm giảm độ nhạy với các chuyển dịch nhỏ và biến dạng cục bộ trong dữ liệu, qua đó giúp giữ được tính bất biến của các đặc trưng quan trọng. Ngoài ra, lớp pooling còn góp phần giảm độ phức tạp tính toán và số lượng tham số của mô hình, giúp cải thiện hiệu suất và giảm thiểu tình trạng quá khớp. Tương tự như lớp tích chập, lớp pooling cũng có các siêu tham số quan trọng như kích thước kernel và stride. Kích thước của cửa sổ pooling (kernel window) sẽ quyết định độ nén của đầu ra, trong khi stride điều khiển bước nhảy của cửa sổ, ảnh hưởng trực tiếp đến tốc độ giảm kích thước và hiệu quả tính toán. Hiện nay, có hai phương pháp pooling phổ biến là max pooling – chọn giá trị lớn nhất trong mỗi vùng nhỏ – và average pooling – tính trung bình các giá trị trong vùng. Trong đó, max pooling thường được sử dụng nhiều hơn trong các mô hình CNN hiện đại do khả năng giữ lại đặc trưng nổi bật tốt hơn.

3.2.4.1. Max Pooling

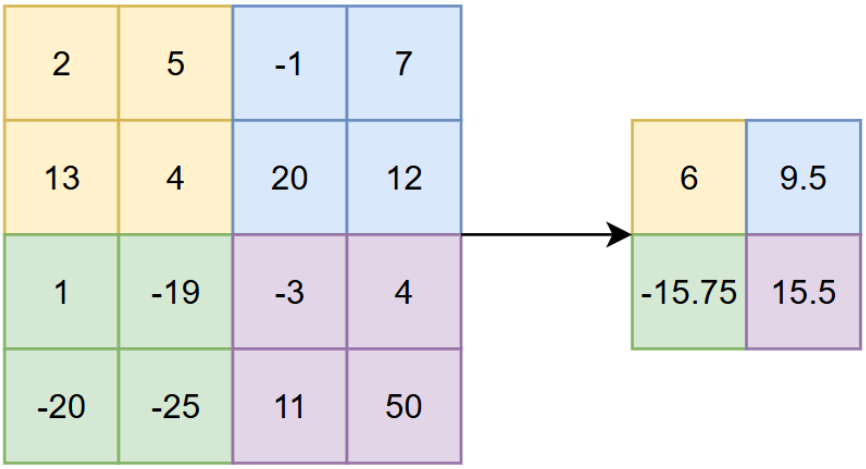
Max pooling là phương pháp pooling phổ biến nhất trong các mô hình học sâu hiện nay. Phép toán này chọn giá trị lớn nhất trong mỗi vùng nhỏ của feature map, giúp giữ lại đặc trưng nổi bật và loại bỏ nhiễu, như minh họa ở Hình 3.6. Ưu điểm chính của max pooling là khả năng chống nhiễu và giữ được tính ổn định khi dữ liệu đầu vào có biến đổi nhỏ. Đồng thời, nó giúp giảm số phép tính, rút ngắn thời gian huấn luyện so với average pooling. Nhờ cấu trúc tính toán đơn giản – chỉ cần tìm giá trị lớn nhất – max pooling cũng rất phù hợp cho hiện thực phần cứng, giúp tiết kiệm tài nguyên và tăng hiệu quả hệ thống.



Hình 3.6. MaxPooling Operation

3.2.4.2. Avarage Pooling

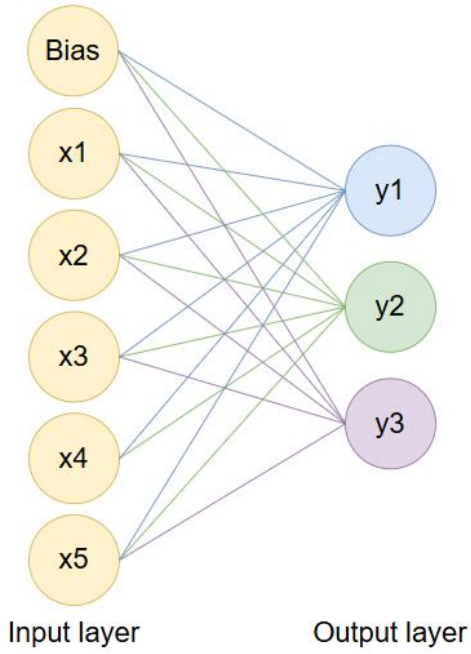
Bên cạnh max pooling, một phương pháp khác cũng thường được sử dụng là average pooling, được minh họa trong Hình 3.7. Average pooling thực hiện bằng cách tính trung bình các giá trị trong mỗi vùng nhỏ của feature map, rồi gán giá trị trung bình đó vào output feature map. Khác với max pooling – vốn tập trung vào các đặc trưng nổi bật – average pooling giữ lại toàn bộ thông tin trong vùng lọc một cách cân bằng hơn. Phương pháp này thường được sử dụng ở giai đoạn gần cuối mô hình, đặc biệt trước các lớp kết nối toàn bộ. Tuy nhiên, nhược điểm của average pooling là làm mờ các đặc trưng cạnh, dẫn đến giảm khả năng phát hiện các đặc điểm sắc nét, đặc biệt trong các tác vụ yêu cầu nhận diện biên rõ ràng.



Hình 3.7. Avarage Pooling Operation

3.2.5. Lớp kết nối toàn bộ - Fully-Connected Layer

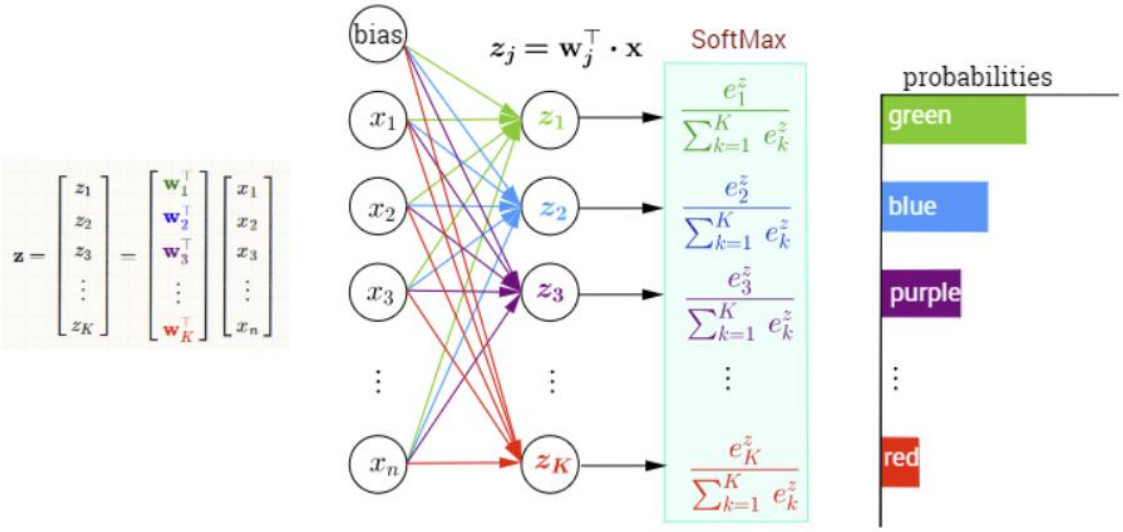
Các feature map đầu ra từ các lớp tích chập và lớp pooling thường được chuyển đổi từ dạng ma trận 3 chiều (3-D) sang dạng vector 1 chiều (1-D) thông qua quá trình làm phẳng (flatten). Vector này sau đó sẽ được đưa vào một hoặc nhiều lớp kết nối đầy đủ (Dense layer), trong đó mỗi giá trị đầu vào được liên kết với tất cả các giá trị đầu ra qua các trọng số (weight). Khi các đặc trưng được trích xuất từ các lớp tích chập và giảm kích thước nhờ các lớp pooling, chúng sẽ được truyền qua một chuỗi các lớp tích chập cuối cùng để tạo ra các giá trị đầu ra của mạng, chẳng hạn như xác suất cho từng lớp trong bài toán phân loại. Lớp fully-connected cuối cùng có số lượng nút bằng với số lớp cần phân loại, và thường được theo sau bởi hàm kích hoạt như ReLU, tương tự như các lớp tích chập trước đó. Hình 3.8 minh họa trực quan cách hoạt động của lớp fully-connected, trong đó số lượng giá trị đầu ra tương ứng với số nhãn cần phân loại. Lớp fully-connected cuối cùng kết hợp với hàm softmax được gọi là lớp phân loại (classification layer), sẽ được trình bày chi tiết hơn trong mục 3.2.6.



Hình 3.8. Fully-Connected Layer

3.2.6. Lớp phân loại – Classification Layer

Đây là lớp quan trọng dùng để xác định nhãn và tạo ra dự đoán cuối cùng của mô hình. Dự đoán được đưa ra dựa trên xác suất của từng node trong vector đầu ra của lớp cuối; node có xác suất cao nhất sẽ được chọn làm kết quả dự đoán. Khác với các lớp kết nối toàn bộ thường sử dụng hàm kích hoạt như ReLU hoặc tanh, lớp cuối cùng của mô hình sẽ dùng hàm softmax để chuyển đổi các giá trị trong vector thành một tập hợp các xác suất, mỗi xác suất tương ứng với một nhãn. Xác suất càng lớn cho thấy mức độ tin cậy cao hơn đối với nhãn đó.

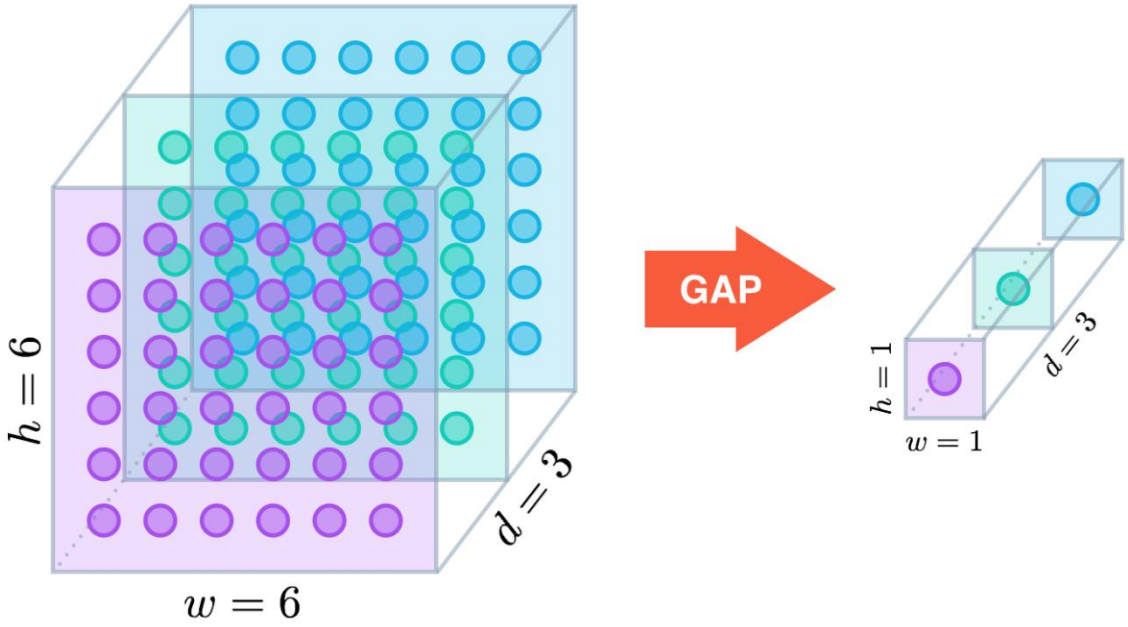


Hình 3.9. Classification Layer

Hình 3.9 minh họa trực quan cách hoạt động của lớp Classification, thể hiện quá trình tính toán các giá trị xác suất cho từng nhãn như đã trình bày trong đoạn trên. Bên cạnh đó, hình cũng trình bày công thức của hàm softmax, được sử dụng để chuyển đổi các giá trị đầu ra thành xác suất. Hàm softmax đóng vai trò rất quan trọng trong việc giúp mô hình máy học đưa ra dự đoán chính xác.

3.2.7. Lớp Global Average Pooling

Lớp Global Average Pooling (GAP) là một kỹ thuật thường được sử dụng trong các mạng nơ-ron tích chập (CNN) nhằm giảm chiều dữ liệu đầu ra của các tầng convolution mà không cần sử dụng các lớp fully connected cồng kềnh. Thay vì duy trì toàn bộ không gian đặc trưng (feature map), GAP thực hiện tính giá trị trung bình của toàn bộ phần tử trong từng kênh (channel) đầu vào, nhờ đó chuyển mỗi feature map 2D (hoặc 1D trong bài toán chuỗi) thành một giá trị duy nhất như được mô tả ở hình 3.10. Cách tiếp cận này không chỉ giúp giảm số lượng tham số và chi phí tính toán mà còn giảm nguy cơ overfitting. Ngoài ra, GAP giữ lại bản chất tổng quát của đặc trưng đầu vào và thường được sử dụng trước lớp softmax trong các mô hình phân loại để biểu diễn mức độ kích hoạt tổng thể của từng kênh, đóng vai trò như một đặc trưng toàn cục.

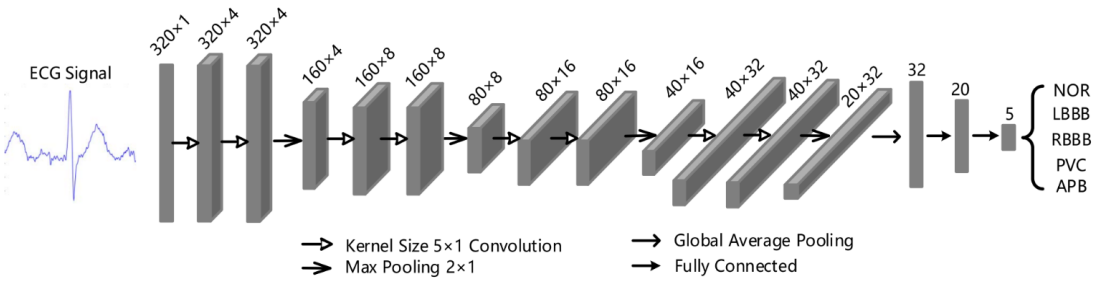


Hình 3.10 Lớp Global Average Pooling

3.3. Kiến trúc mạng 1-D CNN

Mạng 1D CNN được sử dụng để phân loại tín hiệu ECG vì tín hiệu này là dữ liệu dạng chuỗi một chiều theo thời gian, rất phù hợp để các bộ lọc của 1D CNN trượt qua và phát hiện các đặc trưng quan trọng. 1D CNN giúp tự động trích xuất các mẫu đặc trưng cục bộ trong ECG mà không cần phải xử lý thủ công. Đồng thời, phương pháp này giữ được cấu trúc tuần tự của tín hiệu, giúp mô hình hiểu rõ ngữ cảnh thời gian. Ngoài ra, 1D CNN còn có ưu điểm về hiệu quả tính toán cao và khả năng giảm nhiễu nhờ các lớp pooling, góp phần nâng cao độ chính xác khi phân loại các loại tín hiệu ECG khác nhau.

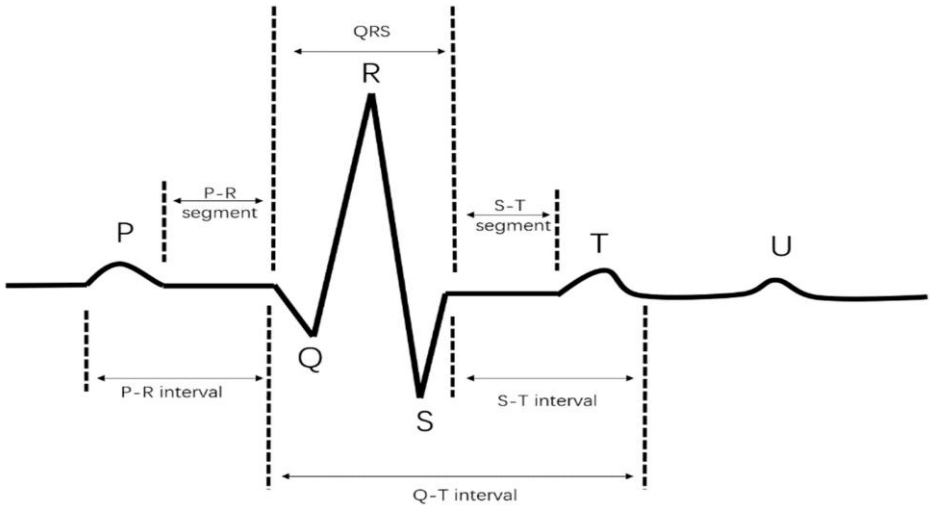
Hình 3.11 mô tả cấu trúc mạng của kiến trúc 1-D CNN. Trong đó, kích thước dữ liệu đầu vào là $320 * 1$, có padding



Hình 3.11. Cấu trúc mô hình 1-D CNN [2]

3.4. Khái quát ECG

Điện tâm đồ (ECG) là một trong những loại tín hiệu sinh lý được ghi nhận phổ biến nhất và đóng vai trò then chốt trong các phân tích chẩn đoán liên quan đến hoạt động của tim. Tín hiệu ECG phản ánh hoạt động điện học của tim và được ghi lại theo từng nhịp đập. Mỗi nhịp tim tạo ra một chuỗi tín hiệu có hình dạng đặc trưng và đều đặn (như minh họa trong Hình 3.12), giúp bác sĩ đánh giá tình trạng tim mạch của bệnh nhân một cách chính xác.



Hình 3.12. Tín hiệu sóng ECG

- Sóng P: là sóng đầu tiên trong mỗi chu kỳ ECG và đại diện cho quá trình khử cực của tâm nhĩ, tức là quá trình tâm nhĩ co bóp để đẩy máu xuống tâm thất. Sóng này thường có biên độ nhỏ, hình tròn và xuất hiện đều đặn nếu nhịp tim bình thường.

Bất thường về sóng P có thể phản ánh các rối loạn như rung nhĩ, ngoại tâm thu nhĩ hoặc phì đại nhĩ.

- Phức bộ QRS: biểu thị quá trình khử cực của tâm thất – giai đoạn mà tâm thất co bóp để bơm máu đi khắp cơ thể. Đây là phần nổi bật nhất trong tín hiệu ECG do biên độ lớn và thời gian ngắn. Phức bộ gồm ba thành phần: sóng Q (sóng âm nhỏ, đôi khi không xuất hiện), sóng R (sóng dương lớn nhất) và sóng S (sóng âm theo sau sóng R). Phức bộ QRS rộng hoặc có hình dạng bất thường thường liên quan đến các bệnh lý như block nhánh, phì đại thất hoặc nhồi máu cơ tim.

- Sóng T: thể hiện quá trình tái cực của tâm thất, tức là giai đoạn tâm thất phục hồi về trạng thái nghỉ sau khi co bóp. Sóng T thường có dạng dương, mềm và xuất hiện sau phức bộ QRS. Hình dạng và hướng của sóng T rất quan trọng trong chẩn đoán các rối loạn như thiếu máu cơ tim, rối loạn điện giải (như tăng hoặc hạ kali máu), hoặc hội chứng QT kéo dài.

- Sóng U: là một thành phần không phải lúc nào cũng xuất hiện trên ECG và thường có biên độ nhỏ hơn sóng T. Sóng U xuất hiện sau sóng T và được cho là liên quan đến sự tái cực chậm của các cấu trúc như hệ thống Purkinje hoặc một số phần của cơ thất. Sóng U rõ ràng hoặc bất thường đôi khi được ghi nhận trong các trường hợp rối loạn điện giải, nhịp chậm hoặc tác dụng phụ của thuốc. Tuy nhiên, cơ chế hình thành sóng U vẫn chưa được hiểu đầy đủ.

3.5. Giới thiệu về tập dữ liệu MIT-BIH

MIT-BIH Arrhythmia Database là một trong những tập dữ liệu ECG nổi tiếng và được sử dụng rộng rãi nhất trong các nghiên cứu về phân tích nhịp tim và phát hiện rối loạn nhịp. Tập dữ liệu này được phát triển vào cuối những năm 1970 bởi Bệnh viện Beth Israel và Viện Công nghệ Massachusetts (MIT), và hiện được lưu trữ công

khai trên nền tảng PhysioNet – một kho dữ liệu sinh lý học mở dành cho cộng đồng nghiên cứu.

Tập dữ liệu bao gồm tổng cộng 48 bản ghi ECG, mỗi bản ghi có độ dài khoảng 30 phút. Dữ liệu được ghi với tần số lấy mẫu 360 Hz, tức là 360 mẫu mỗi giây, đảm bảo độ chính xác cao trong việc phân tích sóng ECG. Mỗi bản ghi chứa 2 kênh tín hiệu ECG, thường là các chuyển đạo DII và V5, cho phép phân tích tín hiệu ở nhiều vị trí trên cơ thể. MIT-BIH Arrhythmia bao gồm tín hiệu ECG từ 47 bệnh nhân khác nhau, trong đó có cả người khỏe mạnh lẫn người mắc các vấn đề về rối loạn nhịp tim.

Mỗi nhịp tim trong tập dữ liệu đều đã được chú thích thủ công bởi các chuyên gia y tế với nhãn tương ứng như N (nhịp bình thường), L (block nhánh trái), R (block nhánh phải), V (ngoại tâm thu thất), A (ngoại tâm thu nhĩ), v.v. Nhờ có tính đa dạng và độ tin cậy cao, MIT-BIH Arrhythmia Database được xem là tập dữ liệu tiêu chuẩn để đánh giá hiệu suất của các mô hình học máy, mạng nơ-ron, và các thuật toán phân tích tín hiệu ECG trong các nghiên cứu y sinh.

Chương 4. **KIẾN TRÚC MẠNG CNN**

4.1. Chuẩn bị

4.1.1. Chuẩn bị dữ liệu huấn luyện

Trong đồ án này, dữ liệu ECG được lấy từ kênh MLII (Modified Lead II), vốn là chuyển đạo phổ biến và cho tín hiệu có đỉnh R rõ ràng, biên độ cao, rất thuận lợi cho việc phát hiện nhịp tim và trích xuất đặc trưng. Tuy nhiên, riêng hai bản ghi số 102 và 104 không có tín hiệu từ kênh MLII do lỗi kỹ thuật hoặc không ghi nhận được tín hiệu, nên bắt buộc sử dụng kênh V2 thay thế.

Việc sử dụng hai kênh khác nhau dẫn đến sự khác biệt trong hình dạng tín hiệu, do mỗi chuyển đạo phản ánh điện thế theo một hướng không gian khác nhau trên cơ thể. Điều này có thể gây ra sự mất cân bằng về phân bố đặc trưng trong tập huấn luyện, đặc biệt khi số lượng bản ghi từ kênh V2 quá ít so với các bản ghi MLII. Nếu không xử lý phù hợp, mô hình có thể học sai lệch theo hướng ưu tiên MLII, ảnh hưởng đến khả năng tổng quát hóa đối với tín hiệu từ các kênh khác.

Sóng được phân loại vào 5 nhãn: N(Normal), A(APC), V(PVC), L(LBBB), R(RBBB):

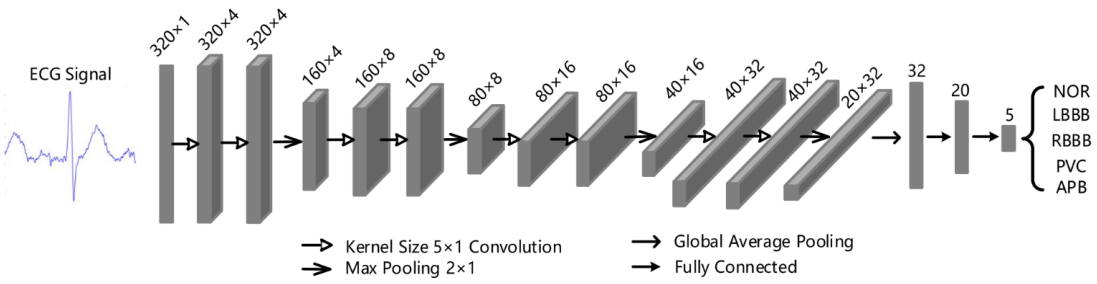
- N (Normal): Nhịp bình thường – đại diện cho hoạt động điện học tim ổn định, không có dấu hiệu bất thường.
- A (Atrial Premature Contraction – APC): Ngoại tâm thu nhĩ – nhịp đập xuất hiện sớm từ tâm nhĩ, thường gây cảm giác tim đập hụt hoặc lệch nhịp.
- V (Premature Ventricular Contraction – PVC): Ngoại tâm thu thất – nhịp tim xuất hiện sớm từ tâm thất, là một loại rối loạn nhịp phổ biến, có thể nguy hiểm nếu xảy ra nhiều.
- L (Left Bundle Branch Block – LBBB): Block nhánh trái – tình trạng dẫn truyền điện tim qua nhánh trái bị chậm hoặc gián đoạn, ảnh hưởng đến chu kỳ co bóp của tâm thất trái.
- R (Right Bundle Branch Block – RBBB): Block nhánh phải – tương tự LBBB nhưng xảy ra ở nhánh phải, ảnh hưởng đến hoạt động tâm thất phải.

Tách chú thích và thời gian của từng nhãn vào một tập dataset sau đó chia ra thành các tập riêng để huấn luyện, kiểm tra và validation với tỉ lệ: tập huấn luyện 70%, tập kiểm tra 15% và tập validation 15%.

Class	Train	Validate	Test	Total
N	52535	11258	11257	75050
L	5653	1211	1211	8075
R	5081	1089	1089	7259
V	4991	1069	1070	7130
A	1782	382	382	2546
Total	70042	15009	15009	100060

Bảng 4.1. Số lượng dữ liệu trong các tập train, test, validate

4.1.2. Cấu trúc mạng CNN



Hình 4.1. Kiến trúc mạng CNN [2]

Layer	Size	Kernel	Kernel size	Stride	Padding	Activation	Parameter
Input	320	1	-	-	-	-	-
Conv1d	320	4	5	1	Same	Relu	24
Conv1d	320	4	5	1	Same	Relu	84
Maxpool	160	4	2	1	-	-	-
Conv1d	160	8	5	1	Same	Relu	168
Conv1d	160	8	5	1	Same	Relu	328
Maxpool	80	8	2	1	-	-	-
Conv1d	80	16	5	1	Same	Relu	656
Conv1d	80	16	5	1	Same	Relu	1296
Maxpool	40	16	2	1	-	-	-
Conv1d	40	32	5	1	Same	Relu	2592
Conv1d	40	32	5	1	Same	Relu	5152
Maxpool	20	32	2	1	-	-	-
GAP	32	-	-	-	-	-	-
Fully connected	32	20	-	-	-	Relu	660
Fully connected	20	5	-	-	-	-	105
Total parameter	-	-	-	-	-	-	**11065**

Hình 4.2. Tổng quát kiến trúc

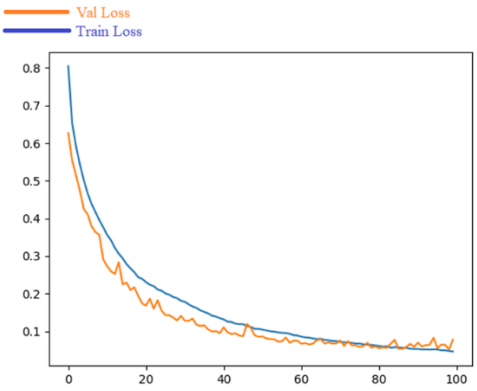
Mạng 1-D CNN này được lấy cảm hứng từ mạng VGG16 bao gồm 8 lớp Convolutional, sau mỗi 2 lớp Convolutional sẽ có một lớp MaxPooling khiến cho mạng đủ sâu nhưng không quá phức tạp khi được hiện thực lên phần cứng. Việc sử dụng hai lớp convolution liên tiếp cho phép mạng học được các đặc trưng phức tạp hơn từ dữ liệu đầu vào trước khi thực hiện giảm kích thước qua pooling, nhờ khả năng kết hợp nhiều bộ lọc cục bộ ở các cấp độ khác nhau. Lớp max pooling sau đó giúp giảm chiều dài tín hiệu, làm giảm số lượng tính toán và tham số ở các lớp sau, đồng thời vẫn giữ lại thông tin quan trọng. Cấu trúc này cũng gia tăng số lượng hàm kích hoạt phi tuyến (ReLU), giúp mô hình có khả năng biểu diễn các quan hệ phức tạp hơn trong dữ liệu. Ngoài ra, việc sử dụng nhiều lớp convolution với kernel nhỏ còn giúp giảm thiểu số lượng tham số so với việc dùng một kernel lớn, đồng thời tăng khả năng khái quát hóa của mô hình. Do đó, việc lựa chọn cấu trúc 2 lớp convolution kết hợp 1 lớp pooling là sự cân bằng hợp lý giữa hiệu quả trích xuất đặc trưng và tiết kiệm tài nguyên, đặc biệt phù hợp với yêu cầu của hệ thống nhúng trong thiết bị đeo. Ngoài các lớp như mạng CNN thông thường, trong kiến trúc này sử dụng lớp Global Average Pooling (GAP) thay vì flatten trước các lớp fully connected nhằm giảm thiểu số lượng tham số và độ phức tạp của mô hình, điều này đặc biệt cần thiết đối với hệ thống nhúng có tài nguyên phần cứng hạn chế. GAP cho phép thu gọn mỗi bản đồ đặc trưng (feature map) thành một giá trị duy nhất bằng cách lấy trung bình, từ đó làm giảm đáng kể số lượng đầu vào cho lớp FC.

4.2. Kết quả huấn luyện mô hình

Trong quá trình huấn luyện, trọng số của mạng CNN 1-D được khởi tạo bằng thuật toán He_normal. Thuật toán này được thiết kế để cải thiện việc khởi tạo trọng số cho các mạng nơ-ron sâu, đặc biệt là những mạng sử dụng hàm kích hoạt ReLU. Sử dụng thuật toán tối ưu hóa Adam trong quá trình lan truyền ngược (back-propagation) để tăng tốc quá trình huấn luyện.

Các chỉ số được sử dụng để đánh giá hiệu suất của mô hình 1-D CNN đề xuất là độ chính xác (ACC), độ nhạy (SEN), độ đặc hiệu (SPEC) và giá trị dự đoán tích cực (PPV), được tính toán dựa trên ma trận nhầm lẫn chuẩn hóa (Confusion Matrix) sử dụng các giá trị true positive (TP), true negative (TN), false positive (FP) và false negative (FN). Công thức của các chỉ số là như sau:

- ACC = $\frac{TP+TN}{TP+TN+FP+FN}$
- SEN = $\frac{TP}{TP+FN}$
- SPEC = $\frac{TN}{FP+TN}$
- PPV = $\frac{TP}{TP+FP}$



Hình 4.3. Train Loss và Val Loss sau huấn luyện

	[2]	Khóa luận này
Tập dữ liệu	MIT-BIH	MIT-BIH
ACC	99.10	99.02
SEN	99.13	98.96
SPEC	98.59	98.48
PPV	99.10	98.91

Bảng 4.2. Kết quả huấn luyện mô hình

Chương 5. QUÁ TRÌNH THỰC HIỆN

5.1. Phương pháp HLS

5.1.1. Chuyển mô hình sang ngôn ngữ C

Chuyển đổi từng lớp của mô hình sang ngôn ngữ C trước khi đưa vào Vitis HLS để tạo ra phần cứng. Dữ liệu được sử dụng sẽ là floating-point 32bit và fixed-point 16bit, sau đó chạy synthesis và đánh giá độ hiệu quả.

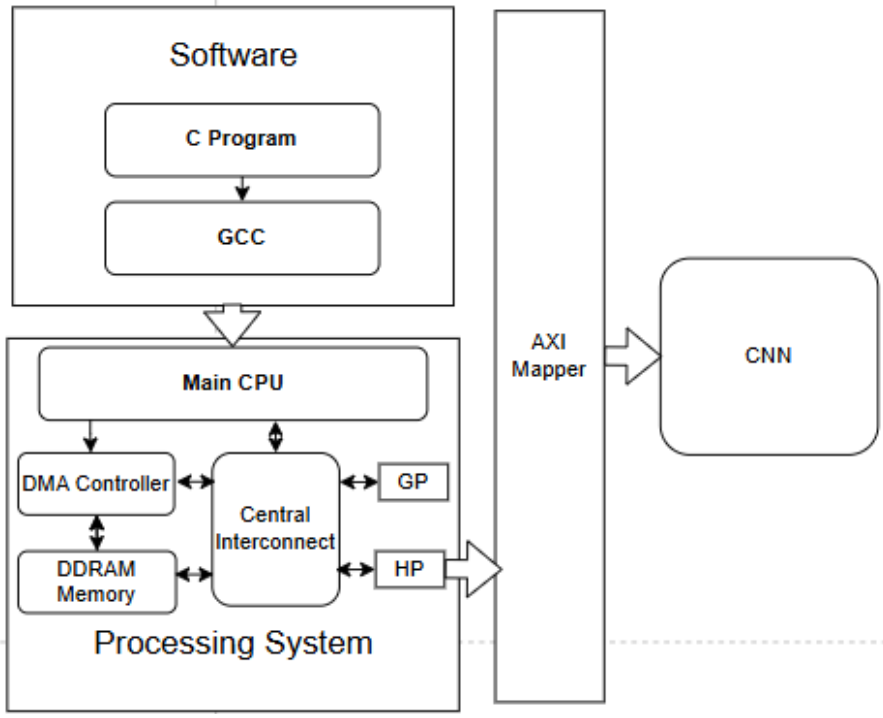
	Floating-point	Fixed-point
Độ dài	32bit	16bit
ACC	98.14	95.58

Bảng 5.1 So sánh kết quả giữa fixed-point và floating-point sau synthesis

Bảng 5.1 so sánh hiệu quả giữa việc sử dụng floating-point 32bit và fixed-point 16bit.

5.1.2. Thiết kế SoC

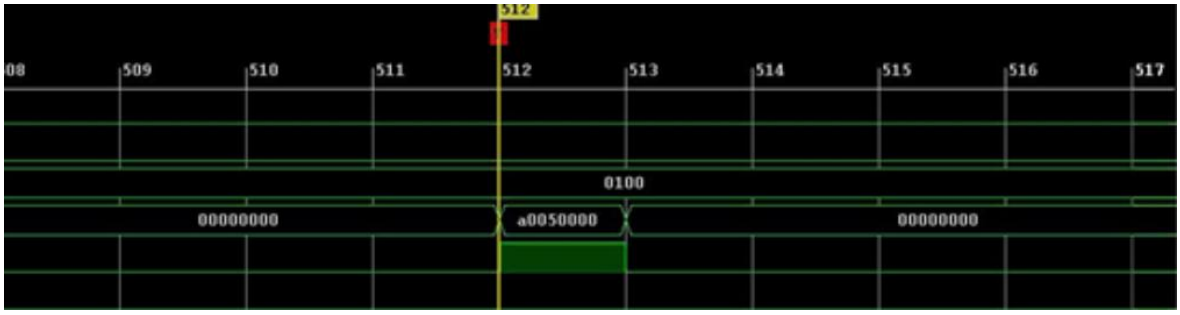
Sau khi có được RTL từ Vitis HLS, tiến hành đóng gói IP và tạo vỏ bọc AXI cho IP(ở đồ án này sử dụng AXI4). Tạo file bitstream, tiến hành boot petalinux lên SD Card sau đó chạy phần mềm thông qua petalinux, sử dụng DMA để truyền dữ liệu vào mô hình thông qua các file .txt đã tạo.



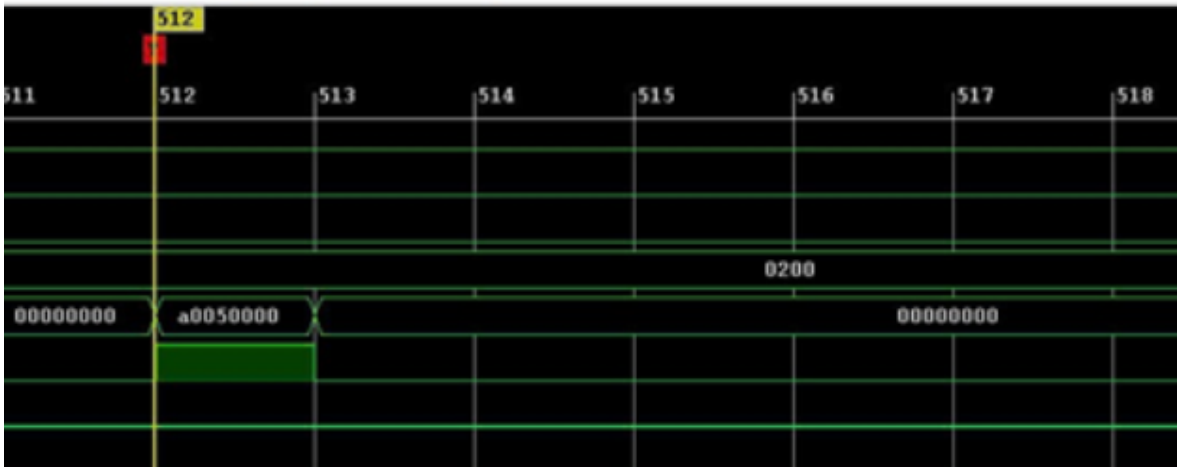
Hình 5.1. Tổng quan kiến trúc SoC với khối CNN được tạo ra từ HLS

5.1.3. Kết quả

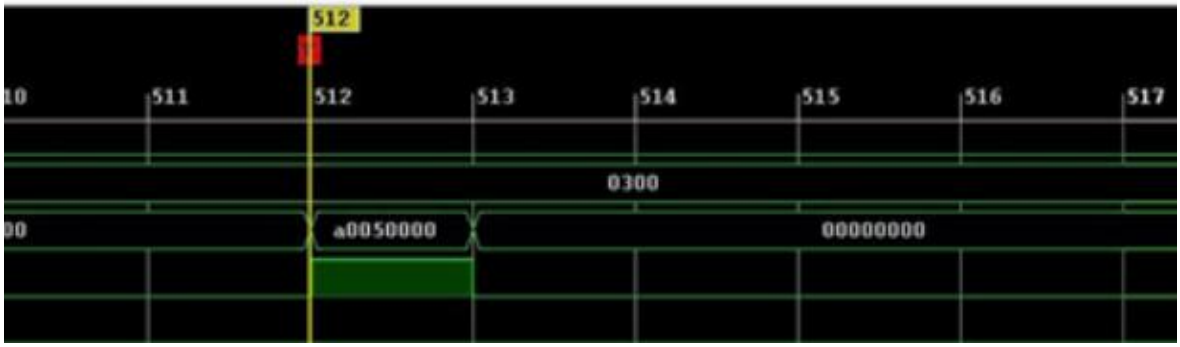
5.1.3.1. Dạng sóng



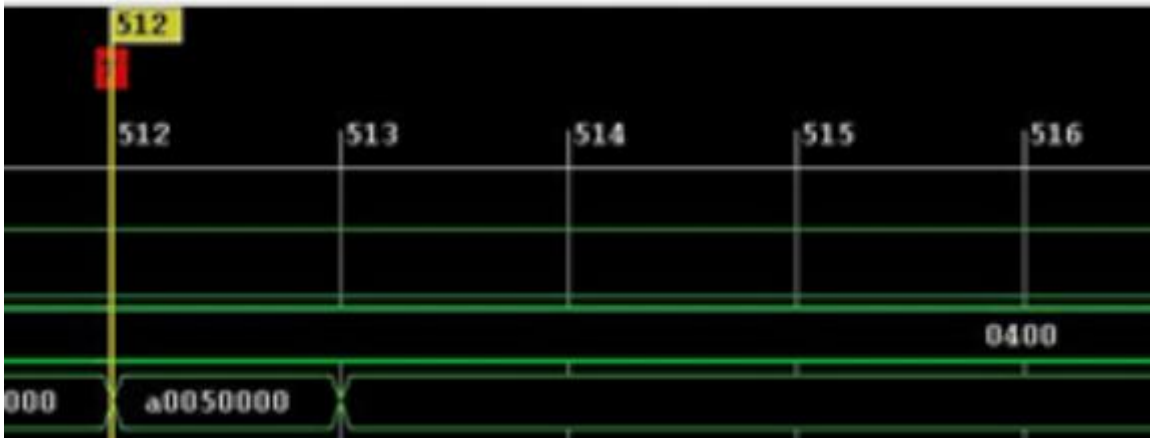
Hình 5.2. Kết quả dự đoán nhãn L



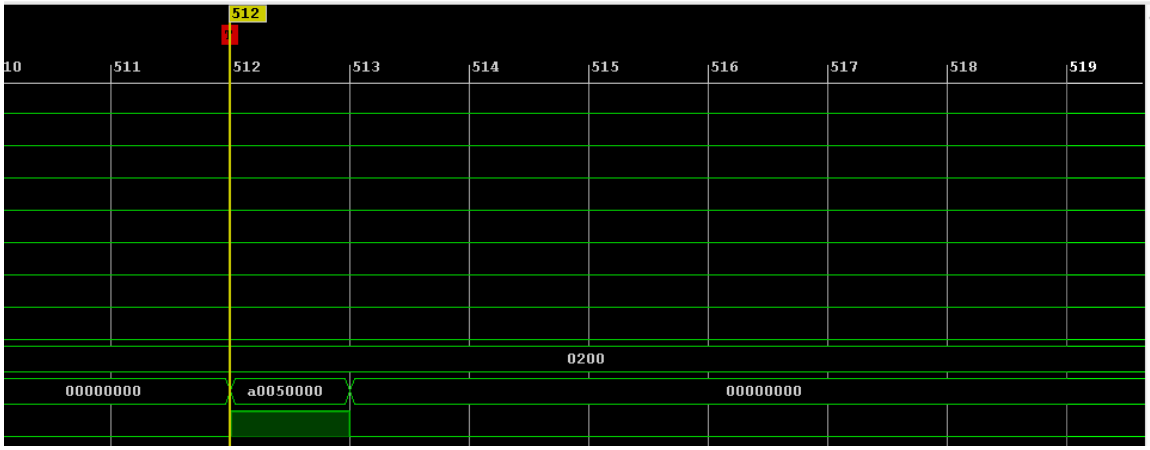
Hình 5.3. Kết quả dự đoán nhãn R



Hình 5.4. Kết quả dự đoán nhãn V



Hình 5.5. Kết quả dự đoán nhãn A



Hình 5.6. Kết quả dự đoán nhãn sai nhãn A

Hình 5.2 đến 5.6 biểu thị kết quả chạy kết quả dạng sóng trên Vivado

5.1.3.2. Đánh giá tài nguyên phần cứng

	[1]	[2]	[3]	[4]	[5]	HLS
CNN Model	1D U-Net	1D-CNN	1D-CNN	1D-CNN	1D-CNN	1D-CNN
FBGA	Zynq XC7Z045	Zynq XC7Z045	Zynq XC720	Zynq XC7Z045	Zynq ZCU102	Xilinx Kria KV260
Clock(MHz)	200	200	200	442.948	200	100
CNN Size(GOP)	0.012	$1.028.10^{-3}$	$0.0578*10^{-3}$	-		$1.59*10^{-3}$
Parameters	-	11065	-	3878178	6457	11065
Precision	16bit Fixed-point	16bit Fixed-point	16bit Fixed-point	16,32bit Fixed-point	16bit Fixed-point	16bit Fixed-point
DSP Utilizition	64	80	48	9	40	189
kLUT	3.8	1.538	3.2	1.067	24,6	14.165
BRAM Utilization	-	12	24.5	50	4.5	53.5
Resource Utilizition(eLUT)	21,720	33,538	61,200	43,587	39,400	109,885

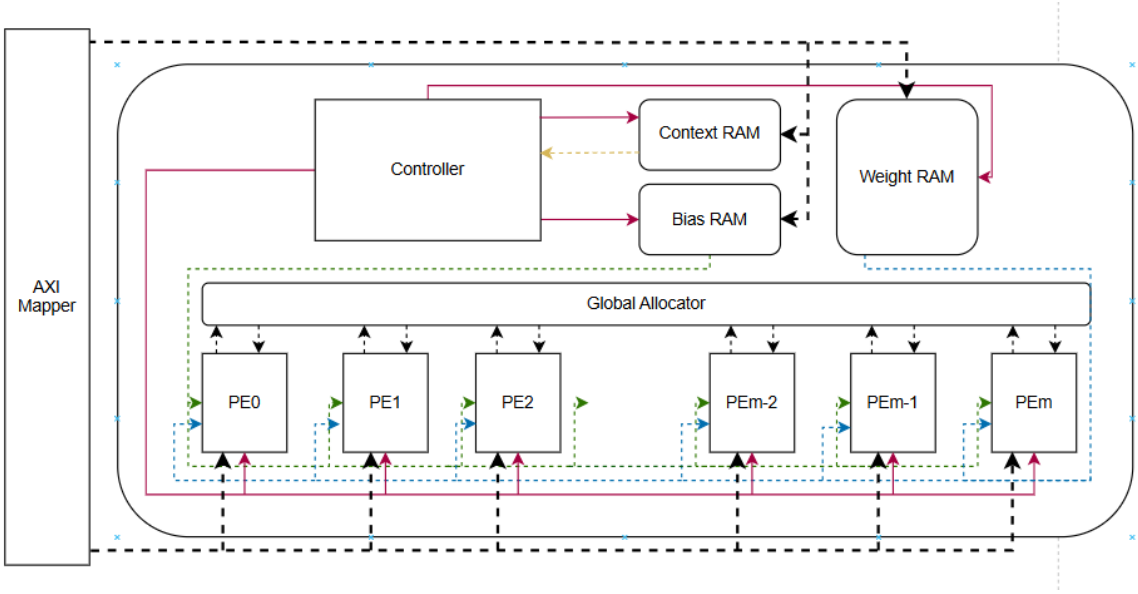
Throughput(GOP/s)	31.44	25.7	16.28	-	9.85	3.106
Hardware resource effiency (GOP/s/MeLUT)	991.173	766.295	266.013	685.526	250	18.272

Bảng 5.2. Bảng báo cáo tài nguyên phần cứng

Bảng 5.2 thể hiện so sánh báo cáo tài nguyên phần cứng của khóa luận này với các bài báo tham khảo trước đó với eLUT = kLUT + BRAM * 800 + DSP * 280 được tính theo [5].

5.2. Phương pháp tự thiết kế

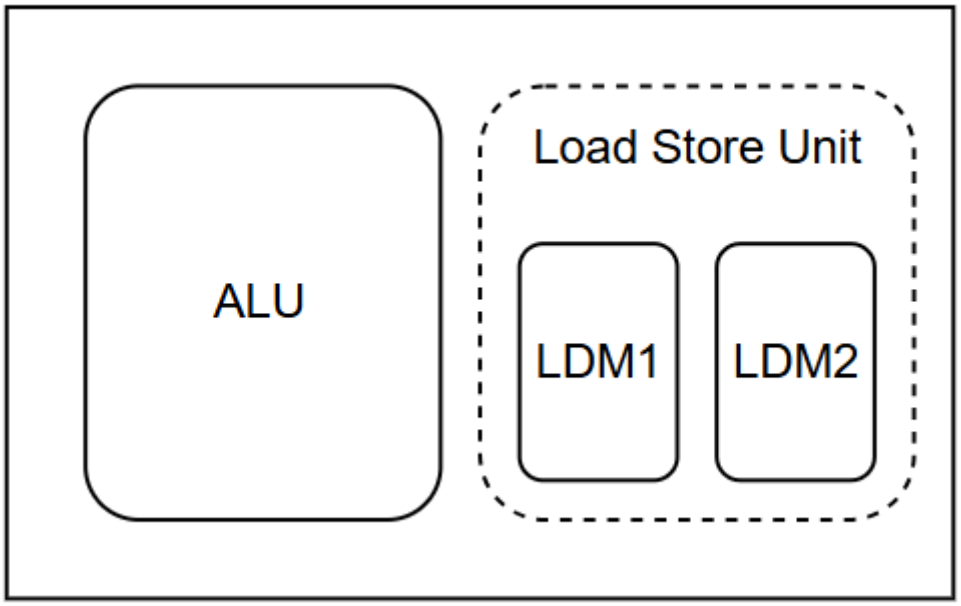
5.2.1. Kiến trúc phần cứng



Hình 5.7. Tổng quan về kiến trúc CNN tự thiết kế

Mô hình mô tả trong hình 5.7 được xây dựng theo phương pháp multi Processing Element(PE). Qua đó mỗi PE sẽ xử lý một phần dữ liệu. Dữ liệu sẽ được điều phối xoay vòng qua các PE thông qua một Global Allocator theo kĩ thuật round-robin. Mỗi PE sẽ bao gồm 1 khối ALU có thể thực hiện được 3 phép toán Multi Accumulate(MAC), phép Add và phép MAX. Mỗi PE cũng được bao gồm 1 khối Load/Store Unit gồm 4 Dual-Port RAM nhằm lưu các dữ liệu trọng số, tín hiệu và

các kết quả tính toán thay vì truy cập BRAM ngoài nhằm tăng băng thông và giảm độ trễ.



Hình 5.8. Tổng quan kiến trúc một PE

Hình 5.8 thể hiện tổng quan một PE gồm một khối ALU để thực hiện viêc tính toán, hai khối LDM1 và LDM2 sẽ luân phiên lưu giữ liệu qua các lớp và khối Load/Store Unit sẽ chịu trách nhiệm đều phối duẽ liệu từ ALU vào và ra LDM.

5.2.2. Thuật toán

5.2.2.1. Chia dữ liệu đầu vào

1: **Input:** $X[K\times Y'], W[N\times K\times J], b[N]$
2: **Output:** $Z[N\times Y]$
3: **Where:** *K, Y' and M are the input channel, input size and the number of PE;*
4: **for** $k = 0$ *to* $K-1$ **do**
5: **for** $y' = 0$ *to* $Y'-1$ **do**
6: $m \leftarrow (k \times Y' + y')\ \%\ M$; ▷*m is the* m^{th} *PE*
7: $d \leftarrow (k \times Y' + y')\ /\ M$; ▷*d is the LDM address in PEs*
8: $PE_m[d] \leftarrow X[m]$; ▷*Pixel inputs areaccessible in the PEs.*

Hình 5.9. Thuật toán chia dữ liệu đầu vào cho các PE

Hình 5.9 mô tả thuật toán phân chia dữ liệu cho các PE theo thủ thuật Round-Robin.

Theo thủ thuật Round-Robin, dữ liệu đầu vào sẽ được chia đều cho các PE, mỗi PE sẽ nhận và xử lý một lượng dữ liệu khác nhau

5.2.2.2. Convolutional

Input: $X[K\times Y'], W[N\times K\times J], b[N]$
Output: $Z[N\times Y]$
Where: *K, Y' and M are the input channel, input size and the number of PE;*
for $n = 0$ *to* $N-1$ **do**
  for $y = 0$ *to* $Y-1$ *with step size* M **do**
    Each m^{th} *PE operates inparallel:*
    $s_m \leftarrow 0$
    for $k = 0$ *to* $K-1$ **do**
      for $j = 0$ *to* $J-1$ **do**
        $d \leftarrow (k \times Y' + y_m \times s + j) / M;$
        $m_i \leftarrow n \times K \times J + k \times K + j;$
        $s_m \leftarrow s_m + W[w_i] \times PE(m + j) \% M[d];$
      $Z[n \times Y + y_m] \leftarrow s_m + b[n];$
      $z_m \leftarrow Z[n \times Y + y_m];$
      $m \leftarrow (n \times Y + y) \% M;$ ▷m *is the* m^{th} *PE*
      $d \leftarrow (n \times Y + y) / M;$ ▷d *is the LDM address in PEs*
      $PE_m[d] \leftarrow z_m;$ ▷*Store pixel outputs into PE's LDM.*

Hình 5.10. Thuật toán cho lớp Convolutional trên mỗi PE

Thuật toán cho các lớp Convolutional được mô tả trong hình 5.10.

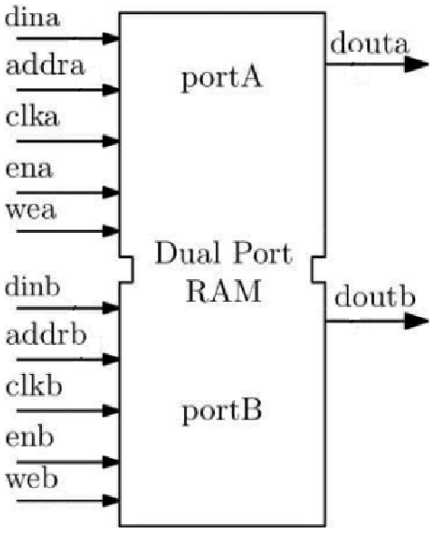
Trong kiến trúc phần cứng xử lý song song, các khối Weight RAM, Bias RAM và Context RAM đóng vai trò là bộ nhớ chuyên dụng, đảm nhận việc lưu trữ các tham số học được và cấu hình điều khiển cần thiết cho từng lớp trong mạng nơ-ron. Cả ba khối đều được thiết kế dưới dạng dual-port RAM, cho phép hai luồng truy cập đồng thời—một dành cho khối điều khiển (Controller) và một cho các khối xử lý như PE hoặc giao tiếp qua AXI.

Weight RAM là nơi lưu trữ toàn bộ trọng số (weight) của các lớp trong mô hình. Tổng số trọng số cần lưu là 10.950, vì vậy bộ nhớ này được cấp phát với 14 bit địa

chỉ, tương ứng với không gian địa chỉ lên tới 16.384 vị trí. Dù số địa chỉ này dư khá nhiều, nhưng việc sử dụng 14 bit giúp đơn giản hóa thiết kế mạch, dễ mở rộng về sau và tận dụng hiệu quả các khối BRAM có sẵn trong FPGA hoặc ASIC.

Bias RAM, ngược lại, có quy mô nhỏ hơn đáng kể với 145 giá trị bias cần lưu. Bộ nhớ này được cấp phát với 8 bit địa chỉ, tương ứng với không gian 256 giá trị.

Context RAM lưu trữ các từ context—mỗi từ là một chuỗi 26 bit chứa thông tin cấu hình cho một lớp cụ thể như số channel, kernel size, input channel, input, source LDM, destination LDM, CFG_ALU, starting address. Tổng cộng có 38 context cần lưu, và RAM này được cấp phát với 6 bit địa chỉ, cho phép không gian lưu trữ lên đến 64 context. Đây là lựa chọn tối ưu: vừa đủ để đáp ứng nhu cầu hiện tại, vừa đảm bảo khả năng mở rộng mô hình về sau mà không cần sửa đổi lớn trong cấu trúc bộ nhớ.



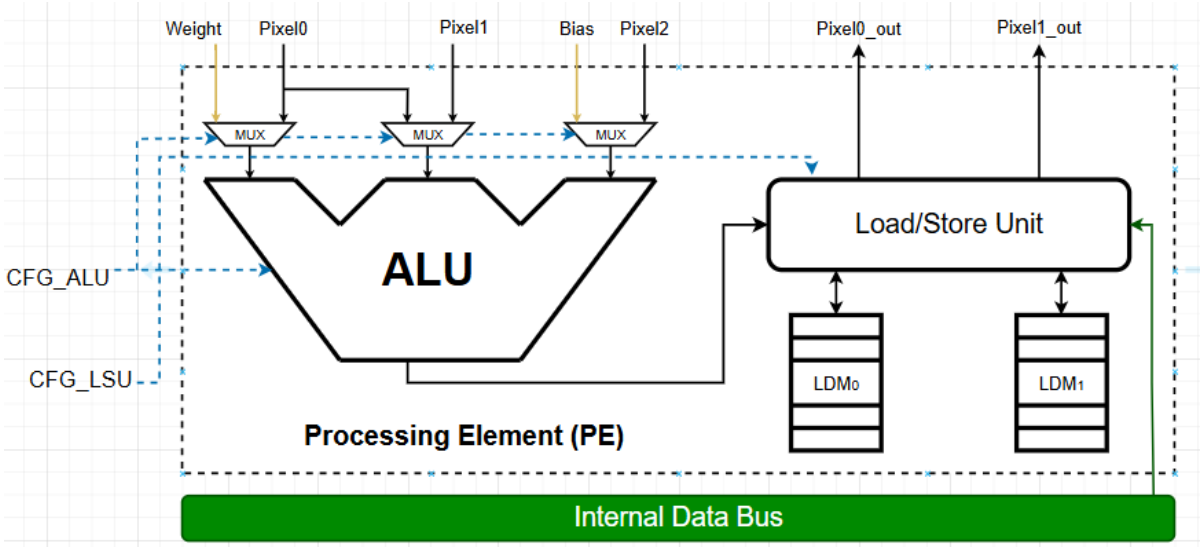
Hình 5.11 Tổng quan một RAM hai cổng (dual-port RAM)

Hình 5.11 mô tả một Dual-port RAM có thể thực hiện đọc ghi đồng thời trên 2 cổng A và B.

5.2.3. Khối Processing Element (PE)

Khối Processing Element(PE) bao gồm 2 module chính là ALU và LSU thực hiện điều phối dữ liệu vào 2 Local Data Memory nhận dữ liệu từ nhiều nguồn: bộ nhớ

weight, bộ nhớ bias, Global Buffer (pixel), và Controller. Mỗi PE sẽ thực hiện tính toán nhờ ALU kết quả sẽ được gửi đến LDM đích thông qua LSU, bên cạnh đó LSU cũng lấy dữ liệu từ LDM nguồn va gửi ra Pixel out của PE.



Hình 5.12 Tổng quan kiến trúc PE

Tổng quan các bước xử lý của một PE được mô tả ở hình 5.12:

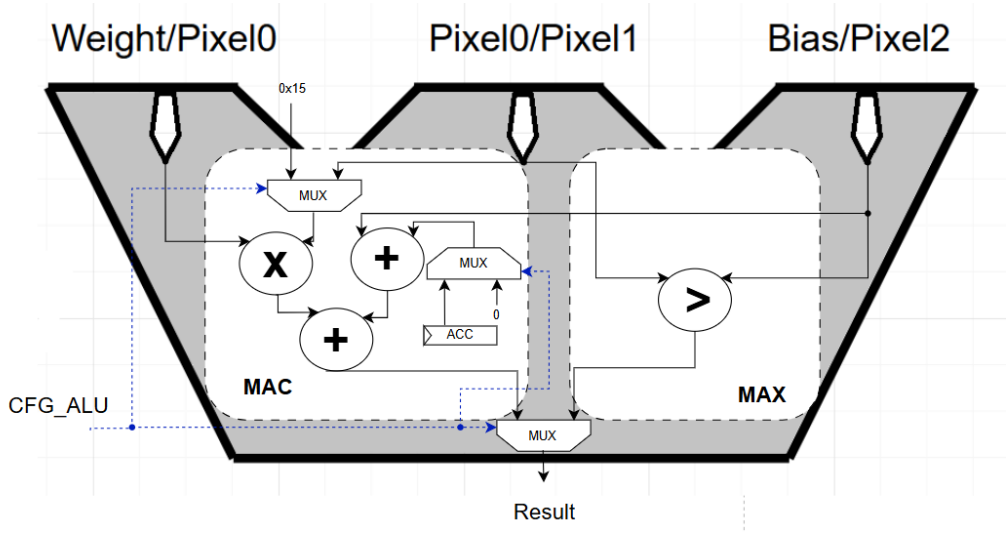
1. Nhận dữ liệu input đầu vào từ AXI hoặc các dữ liệu weight, bias từ weight và bias ram cùng với các tín hiệu điều khiển từ Controller.
2. Weight, bias và dữ liệu pixel được đưa vào ALU theo Opcode từ Controller.
3. ALU thực hiện tính toán sau đó chuyển đến LDM đích thông qua LSU và địa chỉ được Controller tính toán.
4. PE sẽ nhận được 2 pixel liên tục và đưa ra SBA thực hiện phân chia dữ liệu.

5.2.3.1. ALU

ALU nhận các tín hiệu Pixel và tín hiệu Opcode điều khiển từ PE có thể thực hiện 2 phép toán là phép MAC và MAX

Hình 5.13 mô tả kiến trúc của một ALU (Arithmetic Logic Unit), trong kiến trúc này ALU được thiết kế linh hoạt để thực hiện các phép toán khác nhau tùy theo giá trị của tín hiệu điều khiển CFG_ALU (hay còn gọi là opcode). Các thành phần dữ liệu chính

đầu vào bao gồm: Weight, Pixel0, và Bias. ALU gồm hai khối chức năng chính: khối MAC và khối MAX, kết quả được lựa chọn qua một bộ MUX đầu ra.



Hình 5.13 Kiến trúc đơn giản của ALU

Opcode	Phép tính	Input1	Input2	Input3
00	MAC	Weight	Pixel0	Bias
01	MAX	Pixel0(useless)	Pixel1	Pixel2
10	MAC	Weight	Pixel0	Bias
11	MAX	Pixel0(useless)	Pixel1	Pixel2

Bảng 5.3 Bảng phân công input đầu vào của khối ALU

Bảng 5.3 thể hiện input đầu vào của khối ALU cho trưng Opcode với bit đầu tiên của opcode thể hiện tín hiệu cho phép thực hiện hàm kích hoạt ReLU.

1. MAC

MAC(Multiply-accumulate) là phép toán trung tâm trong thiết kế trên nơi mà các phép nhân và cộng liên tiếp được thực hiện để tính toán trọng số và đầu vào. Trong module MAC, phép toán MAC (Multiply-Accumulate) được thực hiện bằng kỹ thuật shift-and-add, thay cho phép nhân truyền thống, nhằm tối ưu hóa tài nguyên phần cứng. Cụ thể, đầu vào S0_in (trọng số) và S1_in (dữ liệu đầu vào) là các số fixed-point 16-bit. Nếu cả hai cùng dấu hoặc cùng âm, chúng sẽ được lấy trị tuyệt đối để thực hiện phép nhân. Phép nhân này được thực hiện bằng cách dịch trái (<<) S0_in

theo từng bit 1 trong S1_in, sau đó cộng dồn tất cả các kết quả để tạo ra tích cuối cùng.

Quá trình shift-and-add này được chia thành hai giai đoạn pipeline (2-stage pipeline) nhằm nâng cao hiệu suất và cho phép xử lý song song các phép nhân ở các chu kỳ khác nhau:

- Tầng 1: Tính toán các partial sum từ bit 0 đến bit 9 của S1_in. Tất cả các dịch trái ứng với các bit này được cộng lại thành tổng sum_stage1_wr. Kết quả của giai đoạn này được lưu tạm vào thanh ghi sum_stage1_rg vào cạnh lên xung clock.
- Tầng 2: Tính toán các partial sum từ bit 10 đến bit 15 của S1_in (đã được lưu vào factor_rg) và cộng chúng với kết quả từ stage 1 (sum_stage1_rg) để tạo thành sum_stage2_wr. Sau đó, tổng này được làm tròn, chia tỉ lệ (scale down) để giữ đúng định dạng fixed-point, và cuối cùng sẽ được đổi dấu nếu hai số đầu vào khác dấu.

Sau bước nhân, kết quả được cộng với giá trị tích lũy từ vòng trước (accumulation_rg) và bias (S2_in) để tạo thành đầu ra cuối cùng. Nếu ReLU_en_in được bật, kết quả sẽ đi qua hàm ReLU: nếu âm thì bị ép về 0. Kết quả này được xuất ra tại MAC_out, kèm theo tín hiệu MAC_valid_out.

Nhờ việc tổ chức pipeline theo 2 tầng như trên, hệ thống có thể tiếp nhận dữ liệu đầu vào mới ở mỗi chu kỳ xung nhịp, trong khi các tầng sau vẫn đang xử lý dữ liệu cũ. Điều này không chỉ tăng throughput (băng thông xử lý) của hệ thống, mà còn giảm critical path, từ đó cải thiện đáng kể hiệu suất và khả năng mở rộng của hệ thống.

- Convolutional

Khi CFG_ALU = 01, ALU sẽ hoạt động ở chế độ nhân tích lũy (MAC). Dữ liệu đầu vào gồm: Weight , Pixel0, và Bias. ALU thực hiện phép nhân giữa Weight và Pixel, sau đó cộng với Bias và giá trị tích lũy hiện tại trong thanh ghi ACC. Nếu có tín hiệu kích hoạt ReLU, sau khi tính toán xong, kết quả sẽ được đưa qua hàm ReLU (gán 0 nếu giá trị âm). Kết quả được lưu vào ACC để sử dụng cho lần tính tiếp theo hoặc xuất ra nếu hoàn tất. Sau khi kết thúc một lớp Bias sẽ được truyền vào lúc này kết quả cuối cùng sẽ được cộng với Bias và sẽ được đưa tới bộ MUX đầu ra.

- Global Average Pooling

Trong chế độ này, ALU thực hiện việc tính trung bình trên toàn bộ một channel (channel lúc này có kích thước 12x1). Thay vì cộng toàn bộ 12 giá trị rồi chia cho 12, ALU sử dụng phép nhân cộng dồn: Mỗi phần tử Pixel0 được nhân với hệ số cố định 0x15 (tương đương 1/12 trong định dạng fixed-point 16-bit (1 dấu, 7 nguyên, 8 thập phân)). Kết quả được cộng dồn vào ACC. Sau khi duyệt hết 12 phần tử của channel, giá trị Bias (khi này sẽ bằng 0) sẽ được cộng vào để đảm bảo Result valid. Kết quả trung bình được xuất ra từ bộ MUX cuối.

- Dense

Lớp Dense 32→20 trong hệ thống được triển khai ngay sau lớp Global Average Pooling, khi toàn bộ đầu ra từ các PE đã được làm phẳng thành một vector 1 chiều gồm 32 phần tử. Khác với các lớp trước đây được phân phối xử lý song song qua nhiều PE, toàn bộ dữ liệu đầu vào của lớp Dense được gom về lưu trữ tại PE0, nơi sẽ thực hiện toàn bộ phép tính trọng số cho lớp này.

Để thực hiện phép biến đổi từ 32 đầu vào sang 20 đầu ra, lớp Dense được ánh xạ dưới dạng 20 phép tích vô hướng giữa vector đầu vào 32 phần tử và 20 hàng trọng số tương ứng. Mỗi hàng trọng số gồm 32 giá trị, được lưu trong WRAM, và được nạp dần vào PE0 để thực hiện phép nhân–cộng với đầu vào. Mỗi lần thực hiện phép tính, kết quả đầu ra sẽ được ghi vào bộ nhớ cục bộ (LDM) theo địa chỉ đích được xác định bởi tín hiệu starting_address, và các địa chỉ này sẽ tiến dần từ 0 đến 20, tương ứng với 20 node đầu ra của lớp Dense.

Cách triển khai này cho phép lớp Dense hoạt động chính xác theo mô hình fully-connected truyền thống, với đầu vào là vector chiều 32 và đầu ra là vector chiều 20, trong khi vẫn đảm bảo toàn bộ phép toán được thực hiện gọn gàng bên trong PE0 nhờ việc gom toàn bộ dữ liệu vào một điểm xử lý duy nhất.

Lớp Dense 20->5 tiếp theo cũng tương tự, sau lớp này tín hiệu complete sẽ bật lên 1, lúc đó output của toàn hệ thống sẽ là giá trị lớn nhất của 5 Pixel trong destination LDM của lớp Dense cuối cùng.

2. MAX

Khi CFG_ALU = 11, ALU sẽ thực hiện phép so sánh cực đại (MAX). Trong chế độ này, ALU lấy hai đầu vào là Pixel1 và Pixel2 để thực hiện phép so sánh. Kết quả là giá trị lớn hơn giữa hai giá trị đầu vào sẽ được chọn và đưa ra qua bộ MUX đầu ra. Chế độ này được sử dụng trong hàm Max-Pooling

5.2.3.2. Load/Store Unit(LSU)

Trong các thiết kế truyền thống, PE (Processing Element) thường chỉ thực hiện các phép toán MAC cơ bản cho các lớp tích chập và phụ thuộc vào bộ nhớ ngoài để lưu trữ dữ liệu trung gian. Điều này không chỉ làm tăng yêu cầu băng thông mà còn giảm hiệu quả xử lý trong các mạng nơ-ron thế hệ mới. Để khắc phục các hạn chế này, thiết kế PE được đề xuất tích hợp bộ nhớ cục bộ trực tiếp bên trong PE, giúp giảm tắc nghẽn băng thông và hỗ trợ hiệu quả các kỹ thuật nâng cao như kết nối dư và tính toán đa lớp.

Hình 5.12 minh họa kiến trúc PE được thiết kế nhằm tăng hiệu năng và tính linh hoạt cho các phép tính mạng nơ-ron hiện đại. PE tích hợp một ALU (Arithmetic Logic Unit), một LSU (Load/Store Unit) và hai bộ nhớ dữ liệu cục bộ (LDM). Không giống như các thiết kế sử dụng nhiều LDM để xử lý song song các nhánh tích chập, kiến trúc này sử dụng hai LDM luân phiên làm nguồn (source) và đích (destination) trong quá trình tính toán. Sau mỗi lớp, vai trò của hai LDM sẽ được hoán đổi để phục vụ lớp tiếp theo, đảm bảo dữ liệu luôn được lưu trữ và sử dụng hiệu quả mà không cần truy cập bộ nhớ ngoài. Trê

Với kích thước mạng có thể lên tới 1280 pixel mỗi lớp, hệ thống sử dụng 40 PE, mỗi PE chịu trách nhiệm xử lý một phần dữ liệu tương ứng với 32 pixel. Do đó, mỗi LDM trong PE được thiết kế dưới dạng dual-port RAM gồm 32 ô nhớ, mỗi ô có độ rộng 16 bit, cho phép lưu trữ toàn bộ phần dữ liệu của một PE trong mỗi lớp. AXI và Controller sẽ thực hiện trên port-A, AXI và ALU sẽ thực hiện trên port-B trong đó AXI sẽ được ưu tiên hơn ở cả 2 port. Cấu trúc này đảm bảo dữ liệu được giữ cục bộ

trong suốt quá trình truyền qua nhiều lớp, đồng thời giảm tối đa độ trễ và xung đột truy cập bộ nhớ.

LSU (Load/Store Unit) chịu trách nhiệm điều phối luồng dữ liệu giữa các khối trong PE. LSU nhận tín hiệu điều khiển từ PE, từ đó định tuyến chính xác dữ liệu đầu ra của ALU tới đúng LDM đích (destination LDM) để ghi kết quả tính toán, đồng thời lấy dữ liệu từ LDM nguồn (source LDM) để cung cấp cho ALU thực hiện các phép toán tiếp theo. LSU sử dụng một cơ chế định tuyến có thể cấu hình nhằm tối ưu hóa mẫu truy cập bộ nhớ, đảm bảo rằng các truy cập đọc/ghi không bị xung đột và đạt được hiệu suất cao.

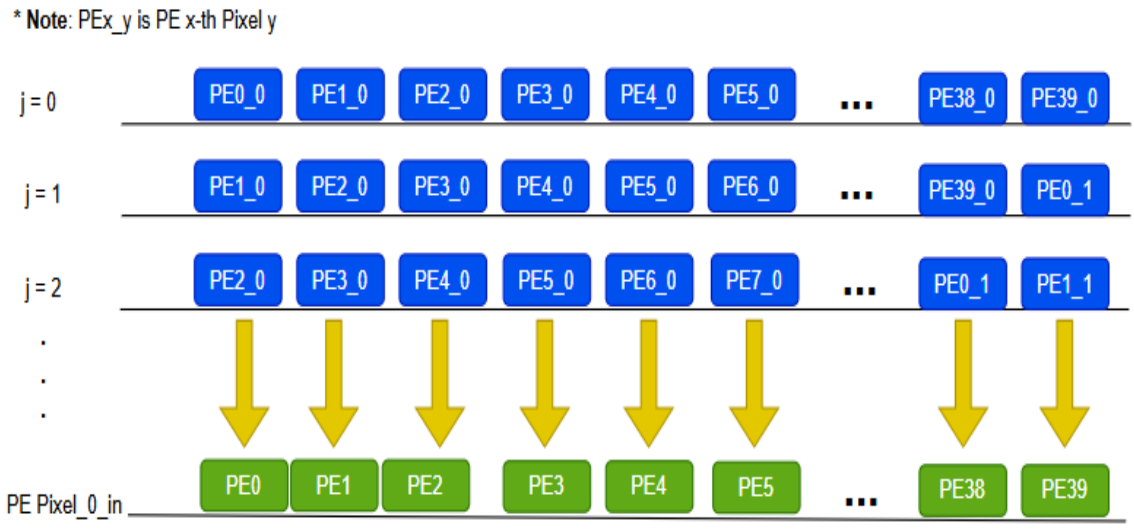
5.2.4. Global Allocator

Khối Global Allocator đóng vai trò là thành phần trung tâm trong hệ thống phân phối dữ liệu cho các PE, đảm bảo quá trình xử lý diễn ra song song và đồng bộ. Mỗi PE sẽ gửi về hai pixel – là hai điểm dữ liệu đầu vào tiếp theo mà PE đó cần xử lý – đến Global Allocator. Nhiệm vụ của Global Allocator là tổ chức phân phối lại các pixel này đến toàn bộ hệ thống PE một cách hợp lý và hiệu quả.

Để đạt được sự cân bằng và tận dụng tối đa tài nguyên xử lý, Global Allocator sử dụng thuật toán phân phối xoay vòng (Round-Robin) dành cho phép MAC. Cơ chế này đảm bảo rằng các PE sẽ lần lượt nhận dữ liệu mới theo thứ tự đều đặn, tránh tình trạng một PE bị đói dữ liệu trong khi PE khác bị quá tải. Cụ thể, pixel đầu tiên sẽ được phân phối cho các PE lân cận hoặc sử dụng ngay trong PE kế tiếp, trong khi pixel thứ hai sẽ được phân phối cho các PE đang xử lý ở phần sau của mảng. Ví dụ, pixel đầu tiên của PE0 có thể được sử dụng cho phép MAC tại PE1, trong khi pixel thứ hai của PE0 – là dữ liệu tiếp theo mà PE0 cần xử lý – sẽ được định tuyến tới các PE ở cuối như PE38 hoặc PE39 để tiếp tục xử lý các vị trí tiếp theo trong mảng đầu vào. Phương thức hoạt động này được mô tả trong hình 5.14.

Việc xoay vòng được hiện thực hóa bằng cách sử dụng 40 MUX loại 40-1, trong đó mỗi MUX điều khiển một đầu vào cho một PE. Các MUX này sẽ chọn một trong 40 nguồn pixel đầu vào dựa trên chỉ số j (tương ứng với kernel size đang được đếm bởi

controller). Điều này cho phép định tuyến dữ liệu một cách chính xác, tuần tự và linh hoạt, đảm bảo dữ liệu được cung cấp đúng thời điểm cho mỗi PE theo vị trí đang tính toán trong kernel, từ đó duy trì hiệu suất xử lý cao và đồng bộ giữa các thành phần trong toàn hệ thống.

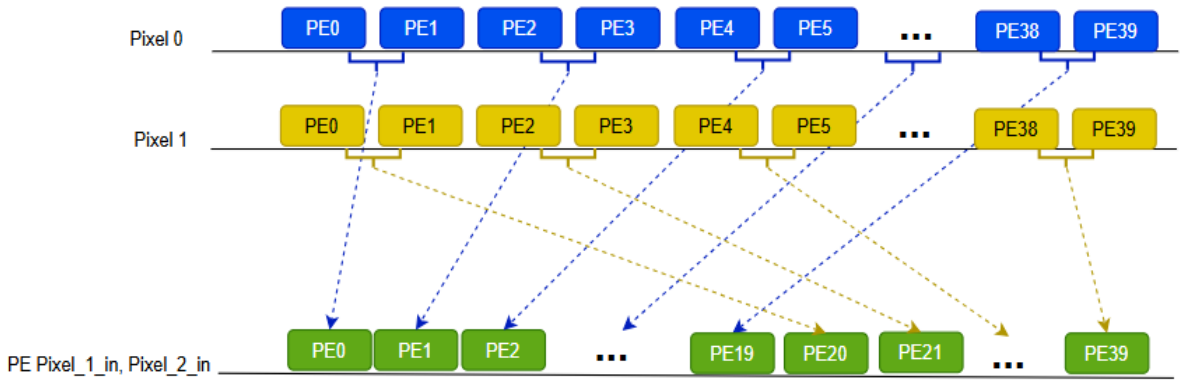


Hình 5.14 Cách phân phối dữ liệu của allocator cho phép MAC

Hình 5.14 thể hiện phương pháp phân phối dữ liệu từ allocator đến các PE

Nhờ vào cấu trúc phân phối này, Global Allocator giúp đảm bảo mọi PE đều nhận đủ dữ liệu kịp thời, cho phép toàn bộ hệ thống hoạt động song song, liên tục, và đồng bộ, từ đó tối ưu hóa hiệu suất xử lý tổng thể của mô hình mạng nơ-ron trên phần cứng.

Với phép MAX, khối Global Allocator đóng vai trò điều phối và chuẩn bị dữ liệu đầu vào cho các PE xử lý một cách song song và hiệu quả. Cụ thể theo hình 5.15 mô tả, từ mỗi PE đầu vào, hai giá trị pixel sẽ được gửi đến Global Allocator, đây là các pixel mà PE đó cần xử lý trong bước tiếp theo. Global Allocator sẽ thu thập các giá trị pixel này từ toàn bộ hệ PE và ghép chúng thành từng cặp, sau đó phân phối đến các PE thực hiện phép toán Max.



Hình 5.15 Phương thức ghép cặp dữ liệu cho phép MAX của allocator

Hình 5.15 thể hiện phương pháp ghép cặp dữ liệu được áp dụng lên kiến trúc phần cứng tự thiết kế.

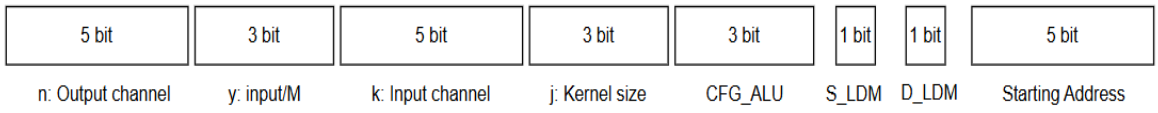
Với kiến trúc sử dụng 40 PE, hệ thống có thể thực hiện 40 phép toán Max trên 40 cặp pixel mỗi chu kỳ, tức là xử lý tổng cộng 80 giá trị pixel trong cùng một thời điểm. Thiết kế này cho phép khai thác tối đa khả năng xử lý song song, đảm bảo dữ liệu được cung cấp kịp thời và đồng bộ giữa các PE, từ đó nâng cao hiệu suất trong lớp MaxPooling. Việc tổ chức phân phối theo cơ chế ghép cặp tự động thay vì yêu cầu tính liên tiếp của pixel giúp hệ thống hoạt động linh hoạt, chính xác và đạt hiệu quả xử lý cao hơn.

Đầu ra của Global Allocator bao gồm ba giá trị pixel: pixel_0 là dữ liệu được phân phối theo cơ chế xoay vòng để phục vụ cho các phép toán MAC trong các tầng tích chập, còn pixel_1 và pixel_2 là hai giá trị được gom nhóm theo cặp, chuẩn bị cho các phép toán Max ở các tầng kế tiếp như MaxPooling. Cách tổ chức này cho phép cùng lúc hỗ trợ nhiều kiểu xử lý khác nhau trong pipeline mạng nơ-ron, đồng thời tối ưu hóa việc sử dụng tài nguyên phần cứng.

5.2.5. Controller

Trong kiến trúc này, khối Controller đóng vai trò là bộ điều khiển trung tâm, chịu trách nhiệm điều phối luồng dữ liệu và phát sinh tín hiệu điều khiển để tổ chức việc thực thi một lớp trong mạng nơ-ron tích chập. Đầu vào chính của Controller là một từ context dài 26 bit như được thể hiện trong hình 5.16, được lưu trữ trong bộ nhớ

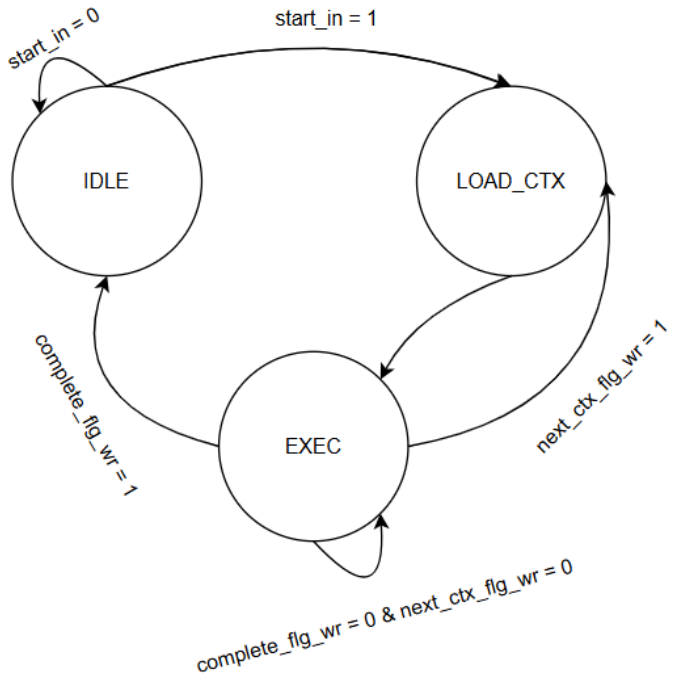
CRAM, đóng gói toàn bộ thông tin cấu hình cần thiết cho lớp hiện tại. Cụ thể, 5 bit đầu tiên biểu diễn số lượng kênh đầu ra (n), tiếp theo là 5 bit cho số lượng kênh đầu vào (k), 3 bit cho kích thước đoạn dữ liệu (chunk) mà mỗi PE sẽ xử lý (y, với M = 40 là số PE), 3 bit cho kích thước kernel (j), 3 bit tiếp theo là cấu hình phép toán (CFG_ALU), trong đó 1 bit điều khiển bật/tắt hàm ReLU và 2 bit chọn loại phép toán (Opcode). Hai nhóm 2 bit tiếp theo lần lượt chỉ định vùng bộ nhớ cục bộ nguồn (source_LDM) và vùng đích (destination_LDM), và 5 bit cuối là địa chỉ ghi bắt đầu trong bộ nhớ đích. Hình 5.16 dưới đây thể hiện cấu trúc của một context:



Hình 5.16 Cấu trúc của một Context

Quá trình điều khiển của Controller được triển khai theo một Finite State Machine (FSM) được mô tả trong hình 5.17 gồm ba trạng thái chính: IDLE, LOAD_CTX, và EXEC. Ở trạng thái IDLE, Controller ở chế độ chờ và sẽ chuyển sang LOAD_CTX khi nhận được tín hiệu start_in. Tại LOAD_CTX, từ context được đọc từ CRAM và giải mã thành các tham số điều khiển nội bộ. Sau đó, FSM chuyển sang trạng thái EXEC, nơi diễn ra toàn bộ quá trình cấp phát dữ liệu, phát tín hiệu điều khiển và phối hợp xử lý song song giữa các khối PE, RAM và LDM.

Bên trong trạng thái EXEC, Controller sử dụng một hệ thống vòng lặp đếm lồng nhau, bao gồm: j_count để duyệt theo từng vị trí của kernel, k_count để quét từng input channel, y_count để phân chia từng đoạn dữ liệu đầu vào cho các PE, và n_count để chuyển sang từng output channel tương ứng. Cách tổ chức lồng nhau của các bộ đếm này giúp bao phủ toàn bộ không gian tính toán theo chiều sâu của một lớp trong mạng nơ-ron, đảm bảo rằng mọi phép toán đều được thực hiện đúng thứ tự và đúng thời điểm.



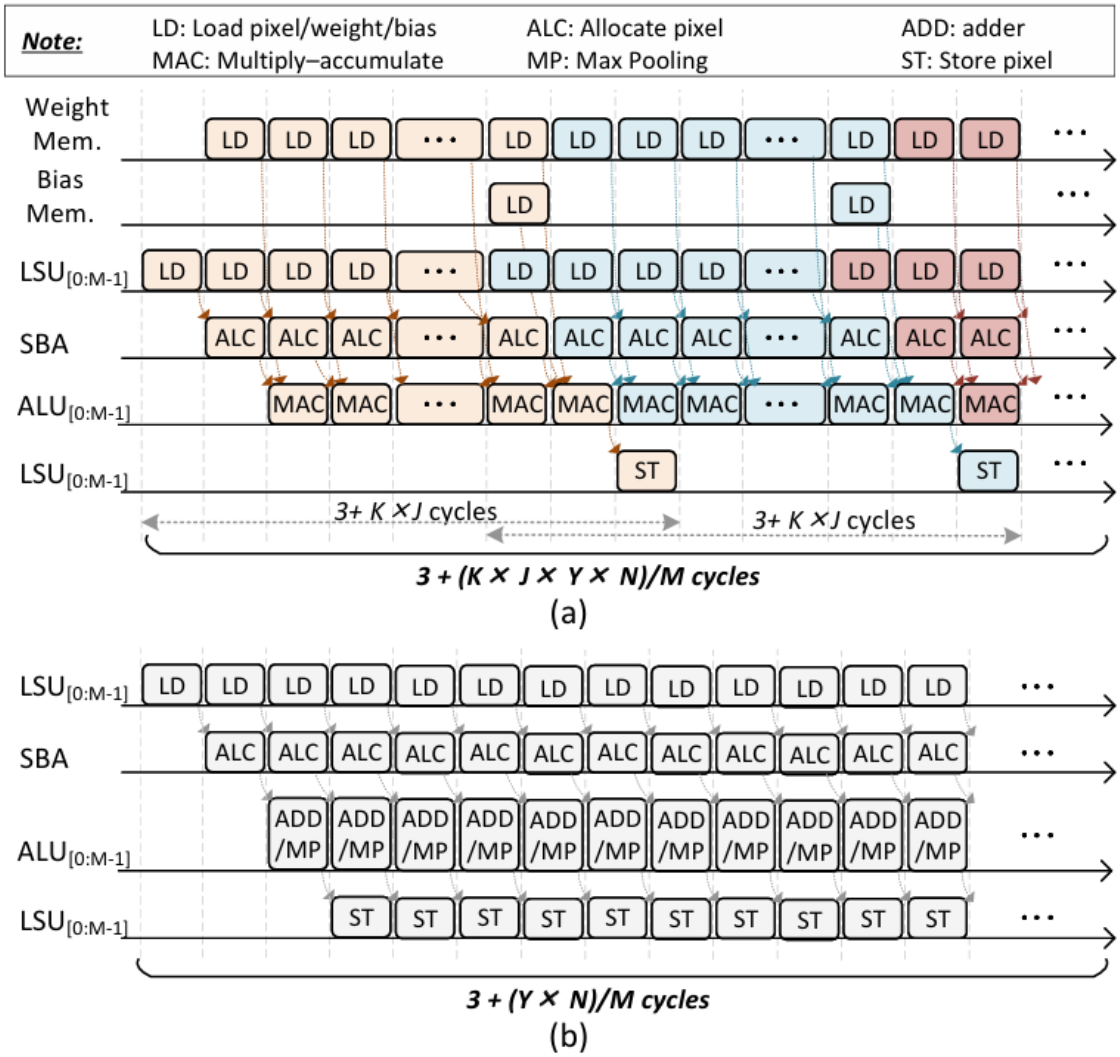
Hình 5.17 Sơ đồ trạng thái của FSM

Đặc biệt, Controller không chỉ hoạt động theo kiểu tuần tự, mà còn được thiết kế như một pipeline tín hiệu điều khiển, trong đó các tín hiệu như địa chỉ đọc/ghi, enable, write, opcode, stride và MUX_Selection được phát ra liên tục theo từng chu kỳ đồng hồ. Các thanh ghi trung gian như next_ctx_flg_1 → next_ctx_flg_4, Overarray → Overarray_2, và các bộ đếm trạng thái được sử dụng như tầng delay để đồng bộ tín hiệu với luồng dữ liệu từ các khối RAM và PE. Nhờ đó, các tín hiệu như En_out, Opcode_out, CTRL_LDM_addra_out, hay MUX_Selection_out luôn được phát ra đúng thời điểm dữ liệu sẵn sàng, tránh tình trạng PE bị "đói" dữ liệu hoặc xảy ra xung đột truy cập.

Bằng cách kết hợp một FSM ba trạng thái rõ ràng, hệ thống đếm lồng nhau chặt chẽ và các tầng pipeline delay chính xác, Controller đã xây dựng được một kiến trúc điều khiển đồng bộ, liên tục và hiệu quả. Nhờ thiết kế này, toàn bộ hệ thống gồm 40 PE có thể hoạt động song song với hiệu suất cao, mà không xảy ra hiện tượng chờ đợi hay nghẽn luồng dữ liệu. Đây là minh chứng điển hình cho việc áp dụng pipeline điều khiển trong hệ thống xử lý song song thời gian thực, góp phần nâng cao thông lượng và độ tin cậy của toàn bộ kiến trúc.

5.2.6. Luồng hoạt động theo Pipeline

Trong thiết kế này, độ trễ phát sinh từ nhiều giai đoạn khác nhau, bao gồm độ trễ đọc/ghi dữ liệu từ bộ nhớ, độ trễ định tuyến trong Global Allocator, độ trễ tính toán trong ALU, và độ trễ ghi kết quả trở lại. Các độ trễ này có thể tích lũy và làm tăng đáng kể tổng thời gian xử lý. Để khắc phục vấn đề này, kiến trúc pipeline hoàn chỉnh đã được triển khai nhằm chồng chéo các giai đoạn tải dữ liệu, phân bổ, tính toán và lưu trữ, đảm bảo rằng thời gian xử lý tổng thể chủ yếu phụ thuộc vào số chu kỳ tính toán trong ALU.



Hình 5.18 Biểu đồ thời gian cho hoạt động đường ống đầy đủ trong (a) lớp tích chập và (b) lớp gộp tối đa/cộng dồn được trích dẫn từ [5].

Như thể hiện trong Hình 5.18(a), pipeline của lớp tích chập trong kiến trúc bao gồm ba giai đoạn chính: tải dữ liệu, phân bổ dữ liệu và thực hiện phép MAC kết hợp lưu trữ kết quả. Ở giai đoạn đầu tiên, LSU (Load Store Unit) nạp các giá trị pixel đầu vào, trong khi bộ nhớ trọng số sẽ nạp các giá trị trọng số và phân phối chúng đến M PE (Processing Element). Quá trình này chỉ mất đúng 1 chu kỳ nhờ chính sách truy xuất dữ liệu cố định của BRAM hoặc SRAM. Tiếp theo, allocator sẽ định tuyến các giá trị pixel đã nạp đến các ALU bên trong PE, và giai đoạn phân bổ này cũng chỉ yêu cầu 1 chu kỳ. Cuối cùng, các ALU bên trong PE sẽ thực hiện phép toán MAC (Multiply-Accumulate) để tính kết quả tích chập cho N kênh đầu ra tại Y vị trí đầu ra.

Khi các vòng lặp j và k lần lượt đạt đến giá trị J và K, bộ nhớ bias sẽ nạp các giá trị bias đến PE đồng thời với trọng số, cho phép các ALU thực hiện đồng thời phép MAC và phép cộng bias trong cùng một chu kỳ. Các kết quả tính toán sẽ được lưu vào bộ nhớ cục bộ LDM. Phép MAC yêu cầu tổng cộng (K × J × Y × N)/M chu kỳ, trong đó K là số kênh đầu vào, J là kích thước kernel, và M là số PE xử lý song song. Tính thêm 2 chu kỳ cho quá trình tải dữ liệu và định tuyến Global Allocator, cùng 1 chu kỳ để lưu kết quả vào LDM, tổng số chu kỳ xử lý cho một lớp convolution trong pipeline được tính theo công thức:

$$\#Cycle_{convolutional} = 3 + \frac{N * Y * K * J}{M}$$

Trong khi đó, đối với Hình 5.18(b), các giai đoạn trong pipeline của lớp cộng hoặc lớp max pooling có cấu trúc tương tự như pipeline của lớp tích chập. Tổng số chu kỳ xử lý (#CycleAdd/Pool) đối với các lớp này được tính theo công thức:

$$\#Cycle_{max} = 3 + \frac{N * Y}{M}$$

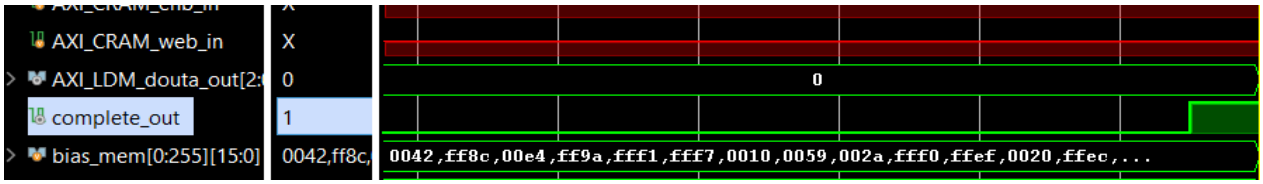
Nhưng ở thiết kế này, với phương pháp gom nhóm dữ liệu cho phép MAX, mỗi chu kì, số lượng điểm đầu vào được xử lý lên đến 2*M, vậy nên số lượng chu kì xử lý lớp Max Pooling được giảm xuống gần như một nửa:

$$\#Cycle_{max} = 3 + \frac{N * Y}{2 * M}$$

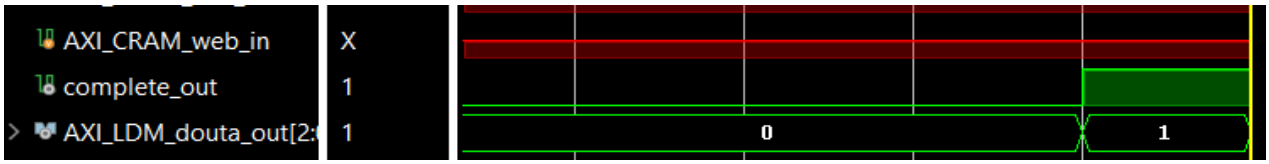
Tổng thể, quy trình pipeline cho phép kiến trúc đạt được số chu kỳ xử lý gần như tương ứng với mức độ song song được xác định bởi số lượng phần tử xử lý (PE) MMM trong tất cả các lớp của mô hình mạng nơ-ron tích chập một chiều (1D-CNN).

5.2.7. Kết quả

5.2.7.1. Kết quả chạy dạng sóng

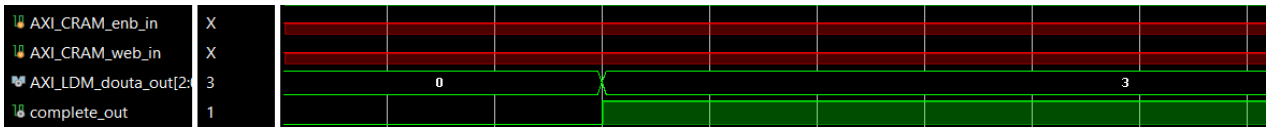


Hình 5.19. Kết quả dự đoán nhãn N



Hình 5.20. Kết quả dự đoán nhãn L

Hình 5.21. Kết quả dự đoán nhãn R



Hình 5.22. Kết quả dự đoán nhãn V

Hình 5.23. Kết quả dự đoán nhãn A

Hình 5.19 đến 5.23 thể hiện kết quả mô phỏng dạng sóng để xác minh tính đúng của kết quả trên 5 nhãn N:0, L:1, R:2, V:3, A:4

5.2.7.2. Đánh giá tài nguyên phần cứng

	[1]	[2]	[3]	[4]	[5]	Tự thiết kế
CNN Model	1D U-Net	1D-CNN	1D-CNN	1D-CNN	1D-CNN	1D-CNN
FBGA	Zynq XC7Z045	Zynq XC7Z045	Zynq XC720	Zynq XC7Z045	Zynq ZCU102	Xilinx Kria KV260
Clock(MHz)	200	200	200	442.948	200	200
CNN Size(GOP)	0.012	$1.028*10^{-3}$	$0.0578*10^{-3}$	-	$0.32768*10^{-3}$	$1.59*10^{-3}$
Parameters	-	11065	-	3878178	6457	11065
Precision	16bit Fixed-point	16bit Fixed-point	16bit Fixed-point	16,32bit Fixed-point	16bit Fixed-point	16bit Fixed-point
DSP Utilizition	64	80	48	9	40	0
kLUT	3.8	1.538	3.2	1.067	24,6	37.941
BRAM Utilization	-	12	24.5	50	4.5	0
Resource Utilization (eLUT)	21,720	33,538	61,200	43,587	39,400	45,007
Throughput (GOP/s)	33.44	25.7	16.28	-	9.85	9.961
Hardware resource efficiency (GOP/s/MeLUT)	991.173	766.295	266.013	685.526	250	221.321

Bảng 5.4 Báo cáo tài nguyên phần cứng

Bảng 5.4 thể hiện đánh giá tài nguyên của phần cứng tự thiết kế và so sánh với các bài báo tham khảo.

5.3. Đánh giá kết quả

Phương pháp	Clock (MHz)	Resource Utilizition (eLUT)	Throughput (GOP/s)	Hardware resource effiency (GOP/s/MeLUT)	Time (cycle)	Power (W)	Temperature (°C)
HLS	100	109,885	3.106	18.272	50,200	2.953	31.9
Tự thiết kế	200	45,007	9.961	221.321	15,065	3.002	32

Bảng 5.3.1 Bảng đánh giá 2 phương pháp

Bảng 5.3.1 so sánh về hiệu quả giữa hai phương pháp HLS và tự thiết kế, kết quả như sau:

- Về tốc độ, phần cứng tự thiết kế cho kết quả nhanh gấp 3,3 lần phương pháp HLS.
- Về độ hiệu quả tài nguyên phần cứng, mô hình tự thiết kế tốn ít tài nguyên phần cứng hơn HLS (ít hơn khoảng 3,1 lần) và cho độ hiệu quả tài nguyên phần cứng tự thiết kế cao hơn HLS khoảng 12,1 lần.
- Về năng lượng, hai thiết kế sử dụng gần như tương đương nhau về mặt năng lượng và nhiệt độ.

5.4. Hướng phát triển trong tương lai

Trong tương lai, hướng phát triển của đề tài sẽ tập trung vào hai khía cạnh chính nhằm nâng cao hiệu quả tổng thể của hệ thống SoC. Thứ nhất, cần nghiên cứu và lựa chọn các mô hình mạng nơ-ron tích chập (CNN) có cấu trúc nhỏ gọn hơn nhưng vẫn đảm bảo độ chính xác, chẳng hạn như InceptionNet. Những mô hình này được thiết kế tối ưu cho môi trường nhúng với giới hạn về tài nguyên phần cứng, giúp giảm thiểu lượng tính toán, bộ nhớ sử dụng và tiêu thụ năng lượng, từ đó phù hợp hơn với các kiến trúc SoC hướng đến hiệu năng–năng lượng. Thứ hai, về mặt phần cứng, có thể tiếp tục tối ưu hóa kiến trúc xử lý bằng cách thiết kế các khối chức năng chuyên biệt cho từng loại tầng trong mạng CNN, như tầng tích chập, tầng gom mẫu và tầng kết nối đầy đủ. Việc tùy biến kiến trúc phần cứng theo đặc điểm của từng loại tầng sẽ giúp tận dụng tối đa tài nguyên logic và băng thông bộ nhớ, đồng thời cải thiện độ song song và hiệu suất xử lý. Hai hướng phát triển này kết hợp với nhau sẽ mở ra khả

năng xây dựng các hệ thống SoC hiệu quả hơn, phù hợp với các ứng dụng học sâu trong thời gian thực trên các thiết bị nhúng.

TÀI LIỆU THAM KHẢO

[1] CHENG, Xuan, et al. Efficient hardware design of a deep U-net model for pixel-level ECG classification in healthcare device. Microelectronics journal, 2022, 126: 105492.

[2] LU, Jiahao, et al. Efficient hardware architecture of convolutional neural network for ECG classification in wearable healthcare device. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68.7: 2976-2985.

[3] ZHANG, Chen, et al. A configurable hardware-efficient ECG classification inference engine based on CNN for mobile healthcare applications. Microelectronics Journal, 2023, 141: 105969

[4] A. K. Jameil and H. Al-Raweshidy, "Efficient CNN Architecture on FPGA Using High Level Module for Healthcare Devices," in IEEE Access, vol.10, pp. 60486-60495, 2022, doi: 10.1109/ACCESS.2022.3180829.

[5] H. Luan Pham, T. D. Tran, V. Trung Duong Le and Y. Nakashima, "MINA: A Hardware-Efficient and Flexible Mini-InceptionNet Accelerator for ECG Classification in Wearable Devices," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 72, no. 6, pp. 2740-2753, June 2025, doi: 10.1109/TCSI.2025.3553837.

[6] Yashar Deldjoo, Zhankui He, Julian McAuley, Anton Korikov, Scott Sanner, Arnau Ramisa, René Vidal, Maheswaran Sathiamoorthy, Atoosa Kasirzadeh, and Silvia Milano. 2024. A Review of Modern Recommender Systems Using Generative Models (Gen-RecSys). In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24). Association for Computing Machinery, New York, NY, USA, 6448–6458. https://doi.org/10.1145/3637528.3671474

[7] Gheewala, S., Xu, S. & Yeom, S. In-depth survey: deep learning in recommender systems—exploring prediction and ranking models, datasets, feature analysis, and emerging trends. Neural Comput & Applic 37, 10875–10947 (2025). https://doi.org/10.1007/s00521-024-10866-z

[8] Li, Yang & Liu, Kangbo & Satapathy, Ranjan & Wang, Suhang & Cambria, Erik. (2024). Recent Developments in Recommender Systems: A Survey [Review Article]. IEEE Computational Intelligence Magazine. 19. 78-95. 10.1109/MCI.2024.3363984.

[9] Kalideen, M., & Yağlı, C. (2025). Machine Learning-based Recommendation Systems: Issues, Challenges, and Solutions. Journal of Information and Communication Technology, 02(Special Issue), 06-12. https://www.researchgate.net/publication/388959637_Machine_Learning-based_Recomm endation_Systems_Issues_Challenges_and_Solutions

[10] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539

[11] Jürgen Schmidhuber, Deep learning in neural networks: An overview, Neural Networks, Volume 61, 2015, Pages 85-117, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2014.09.003.

[12] Shiri, Farhad Mortezapour, et al. "A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU." arXiv preprint arXiv:2305.17473 (2023).https://doi.org/10.48550/arXiv.2305.17473

[13] L. Zhao and S. Ji, "CNN, RNN, or ViT? An Evaluation of Different Deep Learning Architectures for Spatio-Temporal Representation of Sentinel Time Series," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 16, pp. 44-56, 2023, doi: 10.1109/JSTARS.2022.3219816

[14] Bowen Cui, Minyi Liu, Shanqiang Li, Zhifan Jin, Yu Zeng, Xiaoying Lin, Deep learning methods for atmospheric PM2.5 prediction: A comparative study of

transformer and CNN-LSTM-attention, Atmospheric Pollution Research, Volume 14, Issue 9, 2023, 101833, ISSN 1309-1042, https://doi.org/10.1016/j.apr.2023.101833.

[15] Zhong, Guanwen, et al. "Synergy: An hw/sw framework for high throughput cnns on embedded heterogeneous soc." ACM Transactions on Embedded Computing Systems (TECS) 18.2 (2019): 1-23.

[16] Carballo-Hernández, Walther, Maxime Pelcat, and François Berry. "Why is FPGA-GPU heterogeneity the best option for embedded deep neural networks?." arXiv preprint arXiv:2102.01343 (2021).https://doi.org/10.48550/arXiv.2102.01343

[17] S. Lahti, P. Sjövall, J. Vanne and T. D. Hämäläinen, "Are We There Yet? A Study on the State of High-Level Synthesis," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 5, pp. 898-911, May 2019, doi: 10.1109/TCAD.2018.2834439.

[18] Hussain, Muhammad Awais, Rabiah Badar, and Syed Waqar Nabi. "Comparison of Hand-Written RTL code against High-Level Synthesis for Blowfish and Tiny Encrpytion Algorithm (TEA)." (2017).

[19] A. Kamkin, M. Chupilko, M. Lebedev, S. Smolov and G. Gaydadjiev, "High-Level Synthesis versus Hardware Construction," 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 2023, pp. 1-6, doi: 10.23919/DATE56975.2023.10136904.

[20] WHO (2021), Cardiovascular diseases (CVDs), WHO, 11 June 2021.

[21] WHO (2025), Rheumatic heart disease, WHO, Jan 2025.

[22] Sattar, Y., & Chhabra, L. (2023, June 5). Electrocardiogram. In StatPearls. Treasure Island (FL): StatPearls Publishing. PMID: 31747210.

[23] Gurel, Muhsin. A comparative study between rtl and hls for image processing applications with fpgas. University of California, San Diego, 2016.

[24] Millón, Roberto, Emmanuel Frati, and Enzo Rucci. "A comparative study between HLS and HDL on SoC for image processing applications." arXiv preprint arXiv:2012.08320 (2020).

[25] Zwagerman, Michael D. "High level synthesis, a use case comparison with hardware description language." (2015).

[26] Trajkovski, Filip, and Gabriel Alberto Barrios De Leon. Comparison of HLS and RTL Implementation of a Switch Matrix. No. CERN-STUDENTS-Note-2024-104. 2024.

[27] Alam, Syed Asad, et al. "On the RTL implementation of FINN matrix vector compute unit." arXiv preprint arXiv:2201.11409 (2022).

[28] Meeus, Wim, and Dirk Stroobandt. "Today's High Level Synthesis tools: a comparison." STW. ICT Conference (PRORISC-2010). 2010.

[29] Sjövall, Panu, et al. "High-level synthesis implementation of an embedded real-time HEVC intra encoder on FPGA for media applications." ACM Transactions on Design Automation of Electronic Systems (TODAES) 27.4 (2022): 1-34.

[30] Sraitih M, Jabrane Y, Hajjam El Hassani A. A Robustness Evaluation of Machine Learning Algorithms for ECG Myocardial Infarction Detection. J Clin Med. 2022 Aug 23;11(17):4935. doi: 10.3390/jcm11174935. PMID: 36078865; PMCID: PMC9456488.

[31] Vural, M.S., Heryan, K., Sieciński, S., Biłko, P., Grzegorzek, M. (2025). Classification of the Heartbeats in Electrocardiograms with K-Nearest Neighbors Algorithm, Random Forests, and Support Vector Machines - A Pilot Study. In: Gzik, M., Paszenda, Z., Piętka, E., Milewski, K., Jurkojć, J. (eds) Innovations in Biomedical Engineering 2024. MaST 2024. Lecture Notes in Networks and Systems, vol 1202. Springer, Cham. https://doi.org/10.1007/978-3-031-82143-1_20

[32] Sraitih, M.; Jabrane, Y.; Hajjam El Hassani, A. A Robustness Evaluation of Machine Learning Algorithms for ECG Myocardial Infarction Detection. J. Clin. Med. 2022, 11, 4935. https://doi.org/10.3390/jcm11174935.