

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



Nguyễn Minh Hiếu

**NGHIÊN CỨU THIẾT KẾ LÕI IP TRÍCH XUẤT ĐẶC
TRUNG SỬ DỤNG THUẬT TOÁN MRELBP
(MEDIAN ROBUST EXTENDED LOCAL BINARY
PATTERN)**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Kỹ thuật máy tính**

HÀ NỘI - 2025

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Minh Hiếu

NGHIÊN CỨU THIẾT KẾ LÕI IP TRÍCH XUẤT ĐẶC
TRUNG SỬ DỤNG THUẬT TOÁN MRELBP
(MEDIAN ROBUST EXTENDED LOCAL BINARY
PATTERN)

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Kỹ thuật máy tính

Cán bộ hướng dẫn: TS. Nguyễn Kiêm Hùng

(Ký tên)

HÀ NỘI - 2025

LỜI CAM ĐOAN

Tôi xin cam đoan đồ án tốt nghiệp **Nghiên cứu thiết kế lõi IP trích xuất đặc trưng sử dụng thuật toán MRELBP (Median Robust Extended Local Binary Pattern)** là công trình nghiên cứu thực sự của tôi, được thực hiện dựa trên cơ sở lý thuyết, kiến thức chuyên ngành dưới sự hướng dẫn khoa học của TS. Nguyễn Kiêm Hùng.

Tôi xin cam đoan những công việc trong đồ án thực hiện chưa từng được các tác giả khác đề xuất. Với sự hiểu biết của mình, tôi chắc chắn các số liệu, kết quả trong đồ án là trung thực và chưa được công bố ở đâu và trong bất cứ công trình nào trừ công trình của tác giả và tài liệu tham khảo.

Nếu có gì sai trái, tôi xin hoàn toàn chịu trách nhiệm.

Hà Nội, ngày ... tháng ... năm 2025

Sinh viên

Nguyễn Minh Hiếu

LỜI CẢM ƠN

Đầu tiên, em xin cảm ơn đến khoa Điện Tử Viễn Thông, trường Đại học Công Nghệ
Đại học Quốc Gia Hà Nội, các thầy cô đã tận tình chỉ dạy và trang bị cho em những
kiến thức cần thiết trong suốt thời gian trên ghế giảng đường, làm nền tảng cho em có
thể hoàn thành được bài đồ án này.

Em xin gửi lời cảm ơn đến TS. Nguyễn Kiêm Hùng, thầy đã định hướng, hỗ trợ cho
em từ trang thiết bị cho tới các kiến thức bổ ích, giải đáp các thắc mắc cũng như đề xuất
những giải pháp phù hợp để công trình được hoàn thành. Thầy đã đồng hành cùng em
một chặng đường rất dài trong suốt quãng thời gian qua. Em xin được gửi tới thầy những
lời chúc tốt đẹp nhất.

Hà Nội, ngày ... tháng ... năm 2025

Sinh viên

Nguyễn Minh Hiếu

TÓM TẮT

Tóm tắt: Đồ án thực hiện thiết kế ở mức RTL một bộ trích xuất đặc trưng dựa trên thuật toán MRELBP (Median Robust Extended Local Binary Pattern). Phương pháp MRELBP được lựa chọn do khắc phục được hạn chế của LBP truyền thống trong việc nhận diện cấu trúc vĩ mô và tính nhạy với nhiễu. Việc thiết kế phần cứng chuyên biệt giúp tăng tốc xử lý trích xuất đặc trưng, đáp ứng với các yêu cầu cho hệ thống thời gian thực. Hệ thống SoC được triển khai trên nền tảng phần cứng ZCU106 Ultrascale và đánh giá độ chính xác bằng tập dữ liệu Outex-TC với ảnh kích thước 128x128. Kết quả thực thi cho thấy thời gian trích xuất đặc trưng nhanh gấp khoảng 1700 lần phiên bản xử lý bằng phần mềm với ngôn ngữ C++, độ chính xác với các tinh chỉnh với phiên bản phần cứng không thay đổi nhiều so với các phiên bản phần mềm.

Từ khóa: LBP, MRELBP, ZCU106, SoC.

MỤC LỤC

Lời cam đoan	
Lời cảm ơn	
Tóm tắt	
Mục lục	i
Danh mục từ viết tắt	iii
Danh mục hình vẽ	iv
Danh mục bảng biểu	vii
Mở đầu	1
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT	5
1.1. Đặt vấn đề	5
1.2. Trích xuất đặc trưng	5
1.2.1. Khái niệm trích xuất đặc trưng	5
1.2.2. Kết cấu thị giác	6
1.2.3. Các phương pháp trích xuất đặc trưng	7
1.2.4. Local Binary Pattern	8
1.2.5. Các biến thể của LBP	10
1.2.6. Median Robust Extended Local Binary Pattern	11
1.3. Kiến trúc xử lý và giao tiếp dữ liệu	13
1.3.1. Pipelining	13
1.3.2. Giao diện AXI4-Stream	14
1.4. Phần mềm thiết kế và mô phỏng Vivado	15
CHƯƠNG 2. ĐẶC TẢ KỸ THUẬT	17
2.1. Yêu cầu hệ thống	17
2.2. Xác định thiết kế	17
2.2.1. Tối ưu thuật toán phù hợp với phần cứng	17
2.2.2. Sơ đồ triển khai thuật toán	18
2.3. Đặc tả kỹ thuật	20
2.3.1. Đặc tả mô-đun MRELBP	20
2.3.2. Đặc tả mô-đun MedianProcessing	21

2.3.3. ĐẶC TẢ MÔ-ĐUN CI	26
2.3.4. ĐẶC TẢ MÔ-ĐUN NIRD	29
2.3.5. ĐẶC TẢ MÔ-ĐUN JointHistogram	35
CHƯƠNG 3. THIẾT KẾ RTL	37
3.1. MÔ-ĐUN MedianProcessing	37
3.1.1. MÔ-ĐUN LineBuffer	37
3.1.2. MÔ-ĐUN ZeroPadding	38
3.1.3. MÔ-ĐUN MedianCalculation	45
3.2. MÔ-ĐUN CI	50
3.2.1. CƠ SỞ XÂY DỰNG MÔ-ĐUN PatchSum	51
3.2.2. XÂY DỰNG MÔ-ĐUN PatchSum	51
3.3. MÔ-ĐUN NIRD	54
3.3.1. MÔ-ĐUN Interpolation	54
3.3.2. MÔ-ĐUN RIU2	55
3.4. MÔ-ĐUN JointHistogram	55
3.5. MÔ-ĐUN MRELBP	56
3.6. TÍNH TOÁN THỜI GIAN HOẠT ĐỘNG CỦA TOÀN BỘ MÔ-ĐUN	58
CHƯƠNG 4. MÔ PHỎNG VÀ KIỂM THỬ	60
4.1. CƠ SỞ LÝ THUYẾT KIỂM THỬ	60
4.1.1. LÝ THUYẾT VỀ ĐỘ BAO PHỦ	60
4.1.2. CÁC PHƯƠNG PHÁP KIỂM THỬ	60
4.2. PHƯƠNG PHÁP XÂY DỰNG KIỂM THỬ VÀ KẾT QUẢ	61
CHƯƠNG 5. THỰC THI VÀ ĐÁNH GIÁ	65
5.1. THÔNG TIN VỀ BO MẠCH ZYNQ ULTRASCALE+MPSoC ZCU106 EVALUATION KIT .	65
5.2. XÂY DỰNG HỆ THỐNG SoC	65
5.3. KẾT QUẢ VÀ ĐÁNH GIÁ	68
KẾT LUẬN	74
TÀI LIỆU THAM KHẢO	75

DANH MỤC TỪ VIẾT TẮT

STT	Ký hiệu viết tắt	Tên chữ đầy đủ
1	RTL	Register Transfer Level
2	IP	Intellectual Property
3	FGPA	Field Programmable Gate Array
4	SoC	System on Chip
5	LUT	Look Up Table
6	HDL	Hardware Description Language
7	PS	Processing System
8	PL	Programmable Logic
9	LBP	Local Binary Pattern
10	MRELBP	Median Robust Extended LBP
11	SVM	Support Vector Machine
12	KNN	K-Nearest Neighbor
13	ELBP	Extended Local Binary Pattern
14	CNN	Convolutional Neural Network
15	ILA	Integrated Logic Analyzer
16	MRF	Markov Random Field
17	ILA	Integrated Logic Analyzer

DANH MỤC HÌNH VẼ

Hình 1.1	Hình ảnh mẫu về kết cấu	6
Hình 1.2	Hoạt động của LBP với bán kính r và p điểm ảnh xung quanh [1] . .	8
Hình 1.3	LBP với các mẫu bán kính và điểm ảnh lân cận[14]	9
Hình 1.4	36 giá trị đặc trưng ứng với 8 giá trị điểm ảnh lân cận [14]	10
Hình 1.5	Số lượng các mẫu theo U với $p = 8$ [1]	11
Hình 1.6	Sự khác nhau giữa trích xuất đặc trưng của ELBP và RELBP [1] . .	11
Hình 1.7	Mạch trước khi pipelining	13
Hình 1.8	Mạch sau khi pipelining	14
Hình 1.9	Truyền dữ liệu ở kênh AXI4-Stream	14
Hình 1.10	Chu trình thiết kế và tổng hợp của Vivado	15
Hình 2.1	Phiên bản thiết kế phần cứng của biểu diễn điểm ảnh trung tâm - CI	19
Hình 2.2	Quá trình tạo ra đặc trưng của thuật toán RELBP [16]	19
Hình 2.3	Sơ đồ kiến trúc tổng quát của MRELBP IP	19
Hình 2.4	Sơ đồ khối của mô-đun MRELBP	21
Hình 2.5	Dạng sóng của mô-đun MRELBP	21
Hình 2.6	Dạng sóng của mô-đun MedianProcessing	22
Hình 2.7	Sơ đồ khối của mô-đun Buffer6Rows	23
Hình 2.8	Dạng sóng của mô-đun LineBuffer	24
Hình 2.9	Dạng sóng của mô-đun Buffer6Rows	24
Hình 2.10	Dạng sóng của mô-đun ZeroPadding	25
Hình 2.11	Dạng sóng của mô-đun MedianCalculation	26
Hình 2.12	Dạng sóng của mô-đun CI	27
Hình 2.13	Sơ đồ khối của mô-đun CI	28
Hình 2.14	Dạng sóng của mô-đun MRELBP_CI	29
Hình 2.15	Sơ đồ khối của mô-đun NIRD	31
Hình 2.16	Dạng sóng của mô-đun NIRD	31
Hình 2.17	Dạng sóng của mô-đun NI	33
Hình 2.18	Dạng sóng của mô-đun RD	34
Hình 2.19	Dạng sóng của mô-đun RIU2	35

Hình 2.20 Biểu đồ sóng của mô-đun JointHistogram	35
Hình 3.1 Định nghĩa các ký hiệu được sử dụng khi mô tả kiến trúc RTL	37
Hình 3.2 Định nghĩa các tín hiệu ra vào một mô-đun	37
Hình 3.3 Mô tả RTL của mô-đun LineBuffer	38
Hình 3.4 Sơ đồ chuyển trạng thái của mô-đun LineBuffer	39
Hình 3.5 Sơ đồ chuyển trạng thái của mô-đun ZeroPadding	40
Hình 3.6 Mô tả RTL (1) của mô-đun ZeroPadding ứng với cửa sổ 3×3	40
Hình 3.7 Mô tả RTL (2) của mô-đun ZeroPadding ứng với cửa sổ 3×3	41
Hình 3.8 Mô tả RTL của mô-đun ZeroPadding ứng với cửa sổ 5×5	41
Hình 3.9 Mô tả RTL của mô-đun ZeroPadding ứng với cửa sổ 7×7	43
Hình 3.10 Mô tả về mạng sắp xếp với 3 phần tử [18]	46
Hình 3.11 Mô tả cấu trúc bộ so sánh và hoán đổi	46
Hình 3.12 Thực hiện và ví dụ của tìm trung vị của cửa sổ 3×3	47
Hình 3.13 Thực hiện và ví dụ của tìm trung vị của cửa sổ 3×3	47
Hình 3.14 Thực hiện và ví dụ của tìm trung vị của cửa sổ 5×5 [19]	48
Hình 3.15 Xây dựng bộ sắp xếp 5 phần tử dựa trên bộ sắp xếp 3	48
Hình 3.16 Kiến trúc của bộ sắp xếp tăng dần 5 phần tử	49
Hình 3.17 Thực hiện và ví dụ của tìm trung vị của cửa sổ 7×7	50
Hình 3.18 Xây dựng bộ sắp xếp 7 phần tử dựa trên bộ sắp xếp 5 và bộ sắp xếp 3	50
Hình 3.19 Ví dụ về nguyên lý cửa kỹ thuật cửa sổ trượt	51
Hình 3.20 Kiến trúc của mô-đun sum_cum	51
Hình 3.21 Sơ đồ chuyển trạng thái của mô-đun PatchSum	52
Hình 3.22 Kiến trúc RTL (1) của mô-đun PatchSum	53
Hình 3.23 Kiến trúc RTL (2) của mô-đun PatchSum với $r = 2$	53
Hình 3.24 Nội suy song tuyến tính	54
Hình 3.25 Kiến trúc RTL của mô-đun Interpolation	55
Hình 3.26 Kiến trúc RTL của mô-đun RIU2	56
Hình 3.27 Kiến trúc RTL của mô-đun JointHistogram	57
Hình 3.28 Sơ đồ chuyển trạng thái của mô-đun JointHistogram	57
Hình 3.29 Sơ đồ chuyển trạng thái của mô-đun MRELBP	58
Hình 3.30 Mô tả các tín hiệu yêu cầu đọc dữ liệu (1) của mô-đun MRELBP . .	58
Hình 3.31 Mô tả các tín hiệu yêu cầu đọc dữ liệu (2) của mô-đun MRELBP .	59
Hình 3.32 Kết quả mô phỏng với kích thước ảnh 128×128	59

Hình 4.1	Biểu đồ phân tích thời gian và độ bao phủ của các bộ kiểm thử	61
Hình 4.2	Các thành phần cơ bản của một bộ kiểm tra	62
Hình 4.3	Luồng hoạt động của bộ kiểm thử	63
Hình 4.4	Bộ kiểm tra phân tầng	64
 Hình 5.1	Bo mạch ZCU106 UltraScale+MPSoC	65
Hình 5.2	Giao diện của IP MRELBP	66
Hình 5.3	Sơ đồ khái niệm SoC	66
Hình 5.4	Thiết kế khái niệm SoC	67
Hình 5.5	Lưu đồ hoạt động của chương trình phần mềm	67
Hình 5.6	Báo cáo kết quả tổng hợp của phần mềm Vivado	68
Hình 5.7	Thông số định thời của thiết kế	69
Hình 5.8	Công suất tiêu thụ năng lượng của thiết kế	70
Hình 5.9	ILA với điểm kích hoạt là <i>o_intr</i>	72
Hình 5.10	ILA với dữ liệu điểm ảnh đầu vào	72

DANH MỤC BẢNG BIỂU

Bảng 1.1 Các phương pháp trích xuất đặc trưng	7
Bảng 1.2 Một vài biến thể của LBP	12
Bảng 2.1 Các yêu cầu phi chức năng cho thiết kế	18
Bảng 2.2 Danh sách các tham số của mô-đun MRELBP	20
Bảng 2.3 Danh sách các tín hiệu của giao diện mô-đun MRELBP	20
Bảng 2.4 Tham số của mô-đun MedianProcessing	22
Bảng 2.5 Danh sách các tín hiệu của mô-đun MedianProcessing	22
Bảng 2.6 Danh sách các tham số của mô-đун LineBuffer	24
Bảng 2.7 Danh sách các tín hiệu của giao diện mô-đun LineBuffer	24
Bảng 2.8 Danh sách các tham số của mô-đun ZeroPadding	25
Bảng 2.9 Danh sách các tín hiệu của giao diện mô-đun ZeroPadding	25
Bảng 2.10 Danh sách các tham số của mô-đun MedianCalculation	26
Bảng 2.11 Danh sách các tín hiệu của giao diện mô-đun MedianCalculation	26
Bảng 2.12 Tham số của mô-đun CI	27
Bảng 2.13 Danh sách các tín hiệu của mô-đun CI	27
Bảng 2.14 Tham số của mô-đun MRELBP_CI	29
Bảng 2.15 Danh sách các tín hiệu của mô-đun MRELBP_CI	29
Bảng 2.16 Tham số của mô-đun NIRD	30
Bảng 2.17 Danh sách các tín hiệu của mô-đun NIRD	30
Bảng 2.18 Tham số của mô-đun Interpolation	31
Bảng 2.19 Danh sách các tín hiệu của mô-đun Interpolation	32
Bảng 2.20 Tham số của mô-đun NI	33
Bảng 2.21 Danh sách các tín hiệu của mô-đun NI	33
Bảng 2.22 Danh sách các tín hiệu của mô-đun RD	34
Bảng 2.23 Danh sách các tín hiệu của mô-đun RIU2	35
Bảng 2.24 Danh sách các tín hiệu của mô-đun JointHistogram	36
Bảng 3.1 Bảng điều kiện cho dữ liệu đầu ra ứng với cửa sổ 5x5	42
Bảng 3.2 Số chu kỳ thực hiện của các mô-đun ZeroPadding	43

Bảng 3.3	Bảng điều kiện cho dữ liệu đầu ra ứng với cửa sổ 7×7	43
Bảng 3.4	Số chu kỳ thực hiện của các mô-đun mô-đun sắp xếp và mô-đun MedianCalculation	49
Bảng 3.5	Bảng điều kiện cho mô tả hình 3.22	52
Bảng 3.6	Số chu kỳ thực hiện của các mô-đun ZeroPadding	58
Bảng 5.1	Tài nguyên sử dụng	69
Bảng 5.2	Độ chính xác trên các tập dữ liệu với SVM	70
Bảng 5.3	Độ chính xác trên các tập dữ liệu với Logistic Regression	71
Bảng 5.4	Độ chính xác trên các tập dữ liệu với Naive Bayes	71
Bảng 5.5	Độ chính xác trên các tập dữ liệu với KNN	72

MỞ ĐẦU

Lý do chọn đề tài

Với sự phát triển nhanh chóng của dữ liệu, phần cứng, các nhu cầu về ứng dụng của trí tuệ nhân tạo trong đời sống đang trở nên cấp thiết. Các phương pháp xử lý ảnh và thị giác máy hiện đại đã đóng một vai trò quan trọng trong các bài toán về phân loại hình ảnh, nhận diện mẫu, ... Kết cấu là khía cạnh cơ bản nhất của một bức tranh hay hình ảnh, góp phần tạo nên sự nhận dạng của bức tranh đó. Trong các bài toán về thị giác máy, thông tin kết cấu đóng vai trò quan trọng trong việc phân biệt các đối tượng có hình dạng tương tự nhưng thuộc các lớp khác nhau. Số lượng lớn các hình ảnh của vệ tinh, lâm nghiệp,... có thể được xác định dựa vào kết cấu của chúng. Chẳng hạn, trong phân loại hình ảnh vệ tinh, kết cấu giúp phân biệt giữa rừng, đất nông nghiệp.

Mặc dù các phương pháp học sâu đã ngày càng trở nên phổ biến hơn trong nhiều ứng dụng trong suốt thập kỷ qua, việc sử dụng các mô tả (descriptors) vẫn quan trọng trong nhiều lĩnh vực, những nơi mà không có đủ dữ liệu và tài nguyên tính toán để huấn luyện các mô hình phức tạp như CNN. MRELBP là một biến thể nâng cao của LBP (Local Binary Pattern), được thiết kế để khắc phục nhược điểm của LBP gốc như nhạy cảm với nhiễu và giới hạn trong việc phát hiện các cấu trúc có quy mô lớn. MRELBP sử dụng giá trị trung vị thay vì giá trị trung tâm để mã hóa thông tin kết cấu, giúp tăng tính ổn định khi gặp nhiễu và giữ lại thông tin tốt hơn ở các vùng có biến thiên nhỏ. Điểm mạnh của MRELBP nằm ở khả năng tính toán đơn giản. MRELBP có thể kết hợp với các mô hình huấn luyện cổ điển như SVM, KNN với tập dữ liệu không nhiều nhưng cho những kết quả tốt. Với CNN, đặc trưng được trích xuất thông qua các tầng tích chập và phi tuyến. Sau đó, đặc trưng được cung cấp vào những lớp huấn luyện phức tạp. Nhờ vào khả năng học biểu diễn mạnh mẽ, CNN vượt trội trong các bài toán phân loại hình ảnh phức tạp như nhận dạng khuôn mặt, nhận dạng đối tượng. Tuy nhiên, CNN cần một lượng lớn dữ liệu để huấn luyện, tài nguyên tính toán mạnh mẽ và tiêu tốn nhiều tài nguyên phần cứng vì cần rất nhiều các lớp tích chập để đưa ra được các đặc trưng nhiều ý nghĩa. Đã có những chứng minh cho thấy rằng việc sử dụng các phương pháp mô tả truyền thống như LBP đạt được những kết quả cao hơn CNNs trong nhiều tình huống

[2]. Do đó, LBP vẫn chứng minh được tính hiệu quả của mình. Tuy nhiên, LBP là một phương pháp nhạy cảm với các yếu tố về nhiễu và không có khả năng nắm bắt các cấu trúc vĩ mô. Có rất nhiều biến thể của LBP đã được đề xuất như Local Tenary Pattern (LTP), Extended Local Binary Pattern (ELBP),... Trong số đó, MRELBP là một trong những biến thể có được kết quả cao nhất với kết quả đạt lần lượt 99.82%, 99.38% và 99.77% trong 3 tập dữ liệu của bộ dữ liệu Outex-TC [1].

Việc ứng dụng các giải pháp trí tuệ nhân tạo vào đời sống là điều cần thiết, tuy nhiên, các thách thức về thời gian thực vẫn luôn là thách thức lớn. Các hệ thống nhúng nhỏ gọn dùng để thực hiện một chức năng chuyên biệt đang ngày càng chiếm vị trí quan trọng vì chúng có thể đạt được hiệu năng cao hơn rất nhiều. Do đó, mục tiêu của đồ án này là nghiên cứu và thực hiện phần cứng tăng tốc xử lý cho bộ trích xuất đặc trưng sử dụng MRELBP nhằm giảm thời gian tính toán. Để triển khai và kiểm thử trong thực tế, sinh viên sẽ xây dựng một hệ thống System on Chip (SoC) trên bo mạch ZCU106 với IP đã được thiết kế để đảm bảo tính chính xác, khả năng mở rộng và các yếu tố về tăng tốc thời gian.

Phương pháp nghiên cứu

Trong đồ án, để đạt được mục đích nghiên cứu, sinh viên đã tìm hiểu các tài liệu, bài báo, tạp chí quốc tế,... có uy tín, thực hiện việc tính toán mô hình dữ liệu, phân tích số học để đưa ra các hướng giải quyết hợp lý, và sau đó kiểm nghiệm lại kết quả bằng hình thức mô phỏng bằng simulation trên phần mềm Vivado, triển khai tích hợp IP với hệ thống SoC và kiểm nghiệm lại với Integrated Logic Analyzer (ILA), so sánh kết quả đạt được với phần mềm đã có. Cụ thể các phương pháp nghiên cứu sau đã được sử dụng trong đồ án:

- Sử dụng kỹ thuật Line Buffer để giảm truy cập bộ nhớ, đồng thời xây dựng nên các cửa sổ phù hợp cho việc tính toán.
- Sử dụng kiến trúc Systolic cho bộ tính toán trung vị.
- Nghiên cứu phương pháp cửa sổ dịch cho các bộ tính toán tổng ma trận.
- Tính toán, thiết kế bộ nội suy song tuyến tính cho 24-bit dấu phẩy tĩnh.
- Mô phỏng và kiểm thử hệ thống trên phần mềm Vivado để đánh giá tính đúng đắn của thiết kế.

- Sử dụng công cụ ILA (Integrated Logic Analyzer) để phân tích tín hiệu nội bộ và xác thực hoạt động của hệ thống trên nền tảng FPGA.

Nội dung nghiên cứu

- Tìm hiểu về thuật toán MRELBP, đánh giá, so sánh với các biến thể khác của LBP và các thuật toán khác cho bài toán trích xuất đặc trưng.
- Xây dựng kiến trúc tổng thể và chi tiết cho toàn bộ hệ thống.
- Triển khai HDL, mô phỏng và kiểm thử.
- Tìm hiểu bo mạch ZCU106 và triển khai hệ thống SoC.
- Kiểm nghiệm lại toàn bộ hệ thống, so sánh và đánh giá kết quả đạt được.

Đóng góp của đề tài

Với sự hiểu biết của sinh viên, những kết quả nghiên cứu trong đồ án đã đạt được mục đích nghiên cứu đề ra. Những kết quả này bao gồm:

- Xây dựng được kiến trúc cho bộ tăng tốc phần cứng.
- Triển khai HDL, mô phỏng và kiểm tra.
- Kiểm nghiệm kết quả phần cứng và phần mềm với tập dữ liệu Outex-TC.
- Xây dựng hệ thống SoC sử dụng IP đã thiết kế.
- Đánh giá, phân tích kết quả thu được với kết quả mô phỏng và thực tế.

Bố cục của đồ án

Nội dung chính của đồ án được trình bày như sau:

- **Mở đầu:** Trình bày mục đích, phương pháp nghiên cứu, nội dung, đóng góp và bố cục của đồ án.
- **Chương 1: Cơ sở lý thuyết** - Các khái niệm cơ bản về các phương pháp trích xuất đặc trưng truyền thống, LBP, MRELBP, kỹ thuật xử lý, chuẩn giao tiếp và phần mềm sử dụng.

- **Chương 2: Đặc tả kỹ thuật** - Đưa ra các mô tả và yêu cầu kỹ thuật cho từng mô-đun trong thiết kế IP.
- **Chương 3: Thiết kế RTL** - Trình bày chi tiết ở mức RTL cho các mô-đun của thiết kế, thuật toán sử dụng.
- **Chương 4: Kiểm thử** - Trình bày phương pháp kiểm thử, mô phỏng và kiểm tra quá trình, kết quả hoạt động.
- **Chương 5: Thực thi và đánh giá** - Thực hiện thực thi, đánh giá về bộ tăng tốc trên bo mạch ZCU106, các hạn chế và đề xuất phương hướng.
- **Kết luận:** Tổng kết về công việc đã thực hiện và kết quả đạt được.

CHƯƠNG 1

CƠ SỞ LÝ THUYẾT

1.1. Đặt vấn đề

Sự phát triển mạnh mẽ của AI trong nhiều năm trở lại đây đã thúc đẩy mạnh mẽ tới sự phát triển của nhiều ngành nghề. Các ứng dụng và giải pháp AI xuất hiện ngày càng phổ biến trong nhiều lĩnh vực bao gồm tài chính, chăm sóc sức khỏe, bảo mật, ... Năm 2022, thị trường AI toàn cầu được định giá khoảng 454.12 tỷ USD và dự kiến đạt 2575.16 tỷ USD vào năm 2032 [3]. Theo khảo sát của McKinsey, tỷ lệ doanh nghiệp áp dụng AI ít nhất trong một chức năng đã tăng từ 20% năm 2017 lên 50% năm 2022 [4].

Với sự phát triển mạnh mẽ đó, các nhu cầu về việc tăng tốc AI bằng các phần cứng chuyên biệt cũng được tăng theo. Thị trường tăng tốc AI toàn cầu đạt khoảng 19.89 tỷ USD và dự kiến tăng trưởng với tốc độ CAGR 29.4% từ 2024 đến 2030 [5]. Field-Programmable Gate Arrays (FPGAs) là một trong những giải pháp linh hoạt, hiệu quả cho việc triển khai các thiết kế phần cứng tùy chỉnh cho từng ứng dụng AI. Với khả năng tính toán song song mạnh mẽ, ta có thể đạt được hiệu suất cao và độ trễ thấp kể cả đối với các thuật toán xử lý phức tạp. Bên cạnh đó, việc tối ưu hóa năng lượng tiêu thụ và tối ưu hóa I/O có thể đem lại những giá trị có lợi hơn cho nhiều giải pháp. Các nghiên cứu của Apriorit cũng chỉ ra rằng, FPGAs có thể cải thiện hiệu suất tổng thể của các ứng dụng AI [6].

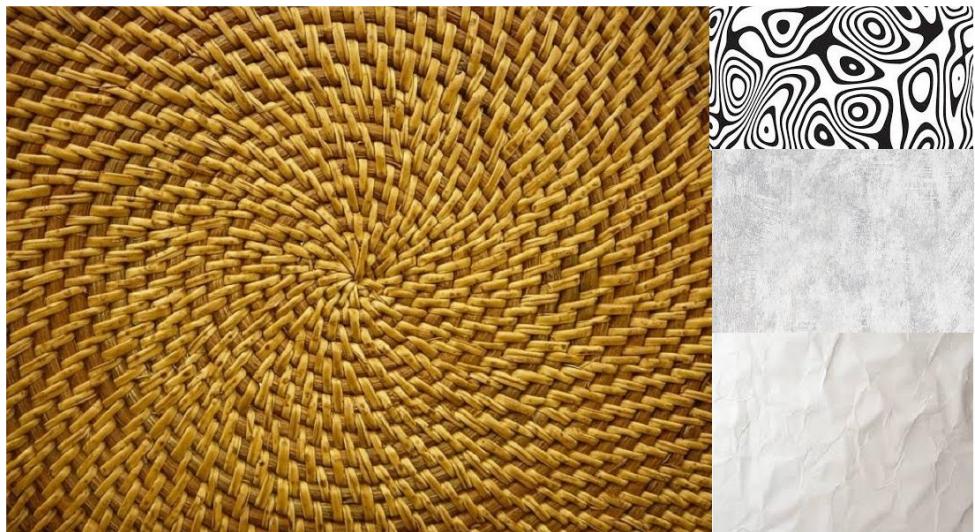
1.2. Trích xuất đặc trưng

1.2.1. Khái niệm trích xuất đặc trưng

Trích xuất đặc trưng là quá trình chuyển đổi các dữ liệu thô thành các dữ liệu số mà có thể xử lý bởi các mô hình mà vẫn đảm bảo được thông tin từ dữ liệu gốc. Việc trích xuất đặc trưng sẽ đem lại kết quả tốt hơn đối với nhiều thuật toán, giảm chiều và độ phức tạp của dữ liệu. Việc trích xuất đặc trưng có thể bao gồm việc lấy ra các dữ liệu đã có trong dữ liệu gốc hoặc tạo ra các đặc trưng mới bằng các kỹ thuật đặc trưng. Các đặc trưng này có thể coi là đặc tính của một đối tượng dữ liệu. Từ những đặc tính đã được trích xuất, các mô hình có thể đạt được các kết quả tốt hơn trong khi lượng thông tin cần xử lý là ít hơn, đảm bảo được hiệu suất xử lý.

Trong xử lý ảnh, trích xuất đặc trưng mô tả các thông tin mang tính hình dạng có liên quan trong một mẫu để nhiệm vụ phân loại mẫu có thể thực hiện dễ dàng [7]. Có thể coi trích xuất đặc trưng là một dạng đặc biệt của giảm chiều dữ liệu. Mục tiêu chính của trích xuất đặc trưng là đạt được các thông tin liên quan của đối tượng từ dữ liệu gốc và biểu diễn chúng ở một chiều dữ liệu nhỏ hơn. Khi mà đầu vào của một thuật toán là quá lớn để xử lý, nó cần được loại bỏ dư thừa. Do đó, dữ liệu cần phải được giảm thiểu bằng cách biểu diễn dưới tập các đặc trưng có ý nghĩa.

1.2.2. Kết cấu thị giác



Hình 1.1. Hình ảnh mẫu về kết cấu

Texture hay kết cấu bề mặt là một khái niệm cơ bản trong lĩnh vực hình ảnh và thị giác máy tính đề cập đến các đặc điểm cơ bản của hầu hết mọi bề mặt tự nhiên có mặt ở khắp mọi nơi trong hình ảnh tự nhiên. Những đặc điểm này bao gồm kích thước, hình dáng, mật độ, sự sắp xếp và tỷ lệ của các thành phần cơ bản tạo nên thành phần đó. Trong thị giác máy tính, kết cấu đóng vai trò là một trong những đặc điểm quan trọng nhất để nhận diện và phân biệt các đối tượng hoặc các vùng quan tâm trong ảnh. Phân loại kết cấu là bài toán cơ bản được ứng dụng trong nhiều lĩnh vực như chuẩn đoán y khoa [8], dầu và gas [9], nông nghiệp [10], phân tích hình ảnh y sinh, hình ảnh vệ tinh, ... Mục tiêu chính của phân loại kết cấu là xây dựng một mô hình có khả năng mô tả nội dung kết cấu đã biết dựa trên dữ liệu huấn luyện. Tuy nhiên, để phân biệt được sự khác biệt giữa các kết cấu là vấn đề khá khó khăn vì các kết cấu thực tế thường có nhiều độ phân giải khác nhau, độ quay tùy ý và có thể được chiếu sáng bởi nhiều điện kiện chiếu sáng khác nhau. Điều này yêu cầu các phương pháp cần có khả năng phân tích được tính

bất biến của đối tượng.

Một đặc trưng kết cấu tốt được mong đợi bởi đáp ứng 2 yếu tố: Độ phức tạp tính toán thấp cho phép phù hợp với các tác vụ phân loại thời gian thực và nắm bắt được thông tin kết cấu đại diện của một lớp kết cấu, sao cho các lớp kết cấu khác nhau có thể được phân biệt mặc dù có sự hiện diện của nhiều dạng hình ảnh khác nhau (bao gồm độ sáng, độ xoay, tỷ lệ, điểm nhìn, nhiễu, ...).

1.2.3. Các phương pháp trích xuất đặc trưng

Để phân loại kết cấu một cách hiệu quả, cần phải có các phương pháp mô tả và trích xuất đặc trưng của từng loại kết cấu. Có thể chia chúng thành 3 loại bao gồm: phương pháp thống kê, phương pháp mô hình và phương pháp bộ lọc [11].

Bảng 1.1. Các phương pháp trích xuất đặc trưng

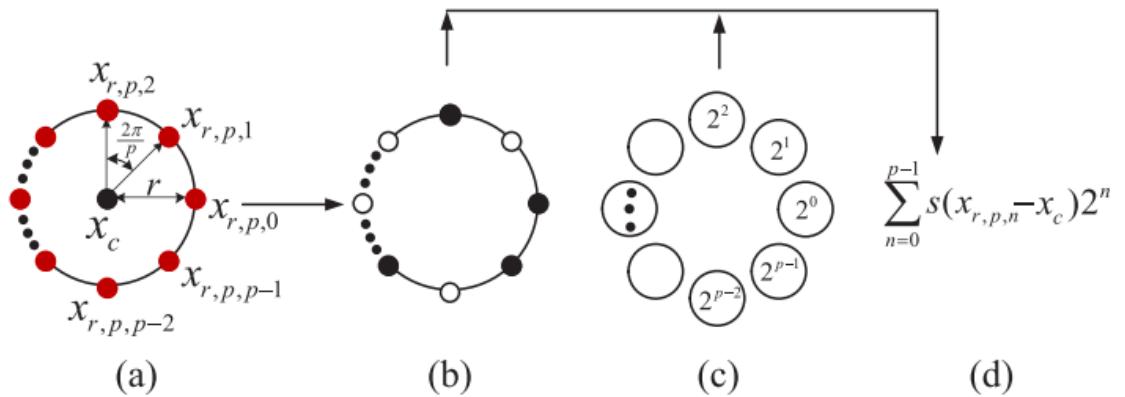
Phương pháp	Đặc điểm	Phân loại	Thuật toán
Thống kê	Phân tích sự phân bố không gian của các giá trị độ xám trong ảnh bằng cách tính toán các đặc trưng cục bộ, cung cấp các thông tin về độ sáng, độ tương phản, ...	Bậc nhất, bậc 2 và bậc cao	GLCM, LBP, ...
Mô hình	Xây dựng các mô hình toán học để mô tả kết cấu trong ảnh, sau đó trích xuất các tham số của mô hình này thông qua phân tích kết cấu. Bằng cách giả định rằng kết cấu được tạo ra bởi một quá trình ngẫu nhiên hoặc tuân theo một quá trình toán học cụ thể, các tham số của mô hình có thể được sử dụng làm đặc trưng để phân loại	Ngẫu nhiên, fractal, tự hồi quy	MRF, ...
Bộ lọc	Sử dụng các bộ lọc để biến đổi ảnh gốc, sau đó nâng lượng của các phản hồi bộ lọc được tính toán tạo thành các đặc trưng kết cấu	Miền không gian, miền tần số, miền không gian - tần số	Gabor filters, Wavelet transforms, ...

1.2.4. Local Binary Pattern

Một vài cách tiếp cận về tính bất biến của độ quay của kết cấu có thể kể đến ảnh cực đồ [13]. Một vài cách khác được đề xuất bằng cách chỉnh sửa các phương pháp đã có như mô hình MRF (Markov Random Field), bộ lọc Gabor hoặc LBP. Trong đó, LBP là một trong những phương pháp thường được sử dụng để mô tả các đặc trưng kết cấu bởi độ phức tạp tính toán thấp, dễ triển khai và bất biến với sự thay đổi ánh sáng đơn điệu [1].

Local Binary Pattern (LBP) là một phương pháp thuộc nhóm phương pháp thống kê, được sử dụng để mô tả các mô tả kết cấu (texture descriptors). LBP được giới thiệu lần đầu tiên vào năm 1994 [12], có khả năng mô tả các cấu trúc không gian của một vùng bên trong hình ảnh bằng cách mã hóa sự khác biệt giữa giá trị điểm ảnh ở vị trí trung tâm vùng với các điểm ảnh xung quanh nó, giá trị thập phân của mẫu nhị phân sau khi so sánh được sử dụng làm nhãn cho giá trị điểm ảnh ở trung tâm đó. Hình 1.2 mô tả hoạt động của LBP với điểm ảnh trung tâm là x_c , kết quả được tính toán bằng cách so sánh giá trị với p các giá trị hàng xóm $\{x_{r,p,n}\}_{n=0}^{p-1}$ được phân bố đồng đều trên một đường tròn bán kính r , kết quả được tính theo công thức sau:

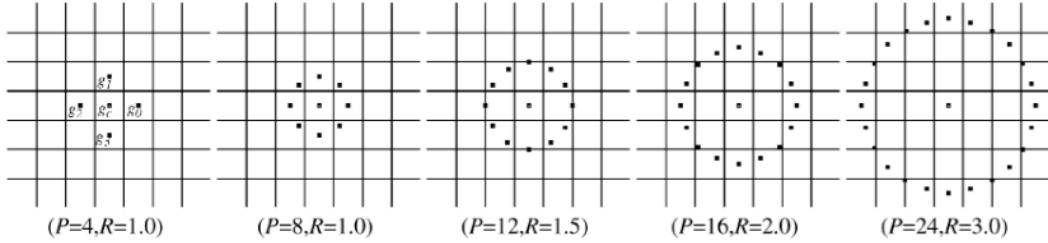
$$LBP_{r,p}(x_c) = \sum_{n=0}^{p-1} s(x_{r,p,n} - x_c) 2^n \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (1.1)$$



Hình 1.2. Hoạt động của LBP với bán kính r và p điểm ảnh xung quanh [1]

Với $s()$ là hàm dấu (sign function). Nếu tọa độ của x_c là $(0, 0)$ thì tọa độ của các điểm xung quanh $x_{p,r,n}$ là $(-rsin(2\pi n/p), rcos(2\pi n/p))$. Giá trị của $x_{p,r,n}$ có thể không

là giá trị ở vị trí trung tâm mà có thể sẽ được ước tính bằng phương pháp nội suy. Tuy



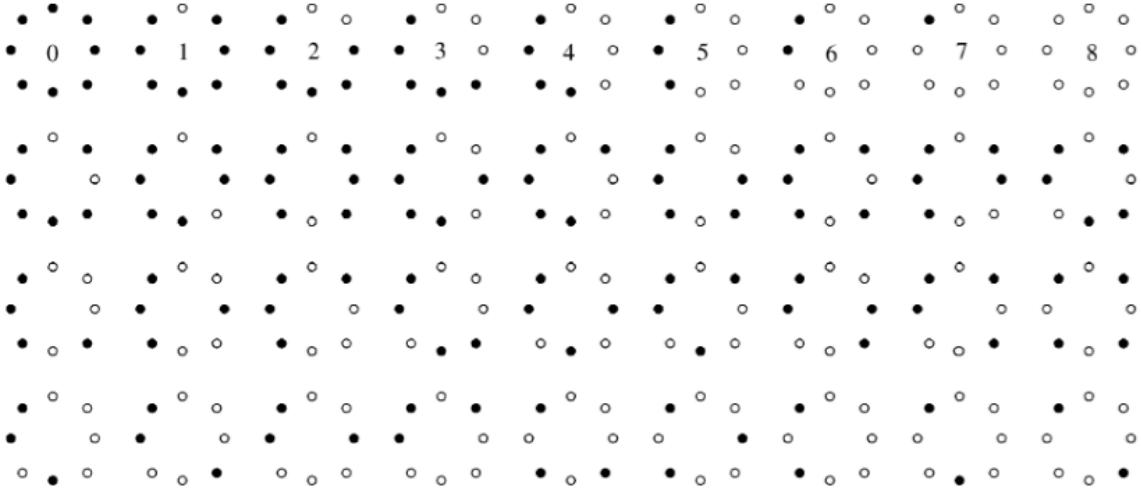
Hình 1.3. LBP với các mẫu bán kính và điểm ảnh lân cận[14]

nhiên, LBP vẫn bị ảnh hưởng nhiều bởi yếu tố độ xoay. Để giảm thiểu sự ảnh hưởng này, một biến thể LBP khác được đưa ra để đạt được sự bất biến của độ quay. Với p giá trị điểm ảnh lân cận, dựa vào toán tử LBP có thể có 2^p giá trị đầu ra khác nhau. Khi ảnh được xoay, giá trị nhị phân có được ở vị trí thứ x sẽ di chuyển xung quanh giá trị trung tâm. Do đó, để giảm sự ảnh hưởng của sự quay, Ojala [14] đã định nghĩa ra một định nghĩa như sau:

$$LBP_{r,p}^{ri} = \min\{ROR(LBP_{r,p}i) | i = 0, 1, 2, \dots, P - 1\} \quad (1.2)$$

Với $ROR(x, i)$ thực hiện việc dịch bit sang phải theo hình tròn. Với p điểm ảnh xung quanh, thì ta sẽ cần dịch đi i lần. Có thể hiểu đơn giản là ta sẽ đi tìm giá trị nhỏ nhất của một dãy bit bằng cách dịch chúng. Ví dụ với 8 điểm ảnh xung quanh, với dãy bit là 10000000_2 , ta có thể có rất nhiều giá trị như 01000000_2 ứng với 64_{10} hay 00000001_2 ứng với 1_{10} . Nhưng khi thực hiện toàn bộ phép dịch và lấy giá trị nhỏ nhất, ta có thể đạt được giá trị là 1_{10} , và đây là giá trị duy nhất. Các điểm ảnh khác nếu có thể đạt được giá trị như 00010000_2 , sau đó cũng được chuyển thành giá trị 1_{10} ở hệ thập phân. Hình 1.4 mô tả số giá trị đặc trưng có thể có với $p = 8$. Mẫu số 0 giúp nhận biết điểm sáng, số 8 giúp nhận biết điểm tối, mẫu số 4 giúp nhận biết biên.

Để cải thiện thêm về tính bất biến của độ xoay, một khái niệm được đưa ra là **đồng nhất (uniform)**. Một mẫu đồng nhất sẽ chứa ít sự thay đổi về mặt không gian. Chúng hoạt động như các khuôn mẫu cho các cấu trúc vi mô như điểm sáng (0), vùng tối (8) hoặc các cạnh có độ cong âm hay dương (1-7). Để định nghĩa một mẫu là đồng nhất, Ojala giới thiệu một phương pháp gọi là U("pattern"), chúng chỉ đo lường sự thay đổi về mặt không gian (tức là sự thay đổi các bit từ 0 thành 1 hay từ 1 thành 0). Ví dụ, mẫu 00000000_2 và mẫu 11111111_2 có giá trị U bằng 0, trong khi đó 7 giá trị khác ở dòng đầu



Hình 1.4. 36 giá trị đặc trưng ứng với 8 giá trị điểm ảnh lân cận [14]

ở hình 1.4 có giá trị U bằng 2 vì có chính xác 2 lần thay đổi bit từ 0 thành 1 và từ 1 thành 0. Trong khi đó, 27 mẫu còn lại có giá trị U ít nhất là 4. Từ đó, ta có định nghĩa về LBP đồng nhất như sau (đặt ngưỡng giá trị của U tối đa là T):

$$LBP_{r,p}^{riuT} = \begin{cases} \sum_{i=0}^{p-1} s(g_i - g_c) & \text{nếu } U(LBP) \leq T \\ p+1 & \text{còn lại} \end{cases} \quad (1.3)$$

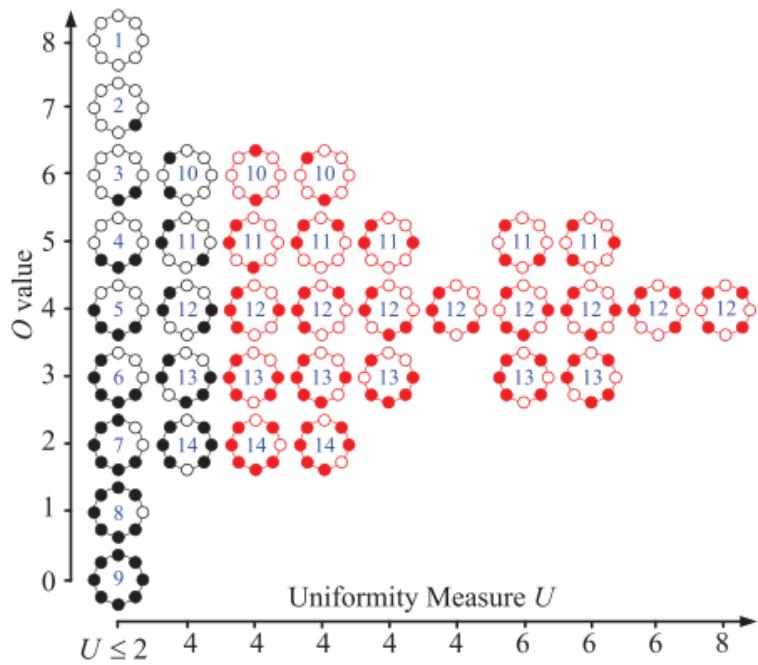
Với

$$U(LBP_{r,p}) = |s(g_{p-1} - g_c) - s(g_0 - g_c)| + \sum_{i=1}^{p-1} |s(g_i - g_c) - s(g_{i-1} - g_c)| \quad (1.4)$$

Từ **riuT** thể hiện việc sử dụng mẫu đồng nhất với ngưỡng T và áp dụng việc dịch phải theo hình tròn. Nếu chỉ áp dụng mẫu đồng nhất T, ta có thể sử dụng từ **uT**. Ví dụ với T = 2, thì ta sẽ có phiên bản $LBP_{r,p}^{u2}$. Với p = 8, ta sẽ có tập hợp các mẫu ứng với giá trị của U, được mô tả trong hình 1.5.

1.2.5. Các biến thể của LBP

Có rất nhiều các biến thể của LBP đã được nghiên cứu nhằm mục đích tăng cường sự mạnh mẽ và khả năng phân biệt. Từ khảo sát từ Di Huang [15], bảng 1.2 mô tả một vài biến thể của LBP.



Hình 1.5. Số lượng các mẫu theo U với p = 8 [1]

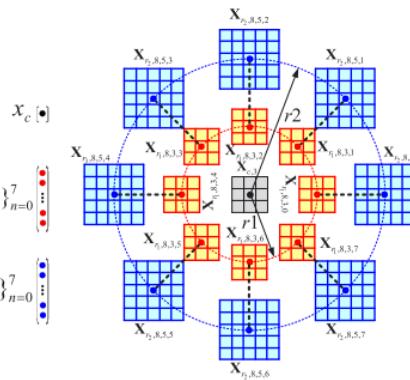
1.2.6. Median Robust Extended Local Binary Pattern

ELBP

$$(b1) \text{ ELBP_CI}(x_c) \\ = s(x_c - \beta) \\ \beta = \frac{1}{N} \sum_{c=0}^N x_c$$

$$(b2) \text{ ELBP_NI}_{r_2,8}(x_c) \\ = \sum_{n=0}^7 s(x_{r_2,8,n} - \beta_{r_2,8}) 2^n \\ \beta_{r_2,8} = \frac{1}{8} \sum_{n=0}^7 x_{r_2,8,n}$$

$$(b3) \text{ ELBP_RD}_{r_2,r_1,8}(x_c) \\ = \sum_{n=0}^7 s(x_{r_2,8,n} - x_{r_1,8,n}) 2^n$$



RELBP

$$(c1) \text{ RELBP_CI}(x_c) \\ = s(\phi(\mathbf{X}_{c,3}) - \mu_3) \\ \mu_3 = \frac{1}{N} \sum_{c=0}^N \phi(\mathbf{X}_{c,3})$$

$$(c2) \text{ RELBP_NI}_{r_2,8,5}(x_c) \\ = \sum_{n=0}^7 s(\phi(\mathbf{X}_{r_2,8,5,n}) - \mu_{r_2,8,5}) 2^n \\ \mu_{r_2,8,5} = \frac{1}{8} \sum_{n=0}^7 \phi(\mathbf{X}_{r_2,8,5,n})$$

$$(c3) \text{ RELBP_RD}_{r_2,r_1,8,5,3}(x_c) \\ = \sum_{n=0}^7 s(\phi(\mathbf{X}_{r_2,8,5,n}) - \phi(\mathbf{X}_{r_1,8,5,n})) 2^n$$

Hình 1.6. Sự khác nhau giữa trích xuất đặc trưng của ELBP và RELBP [1]

Với ELBP, một trong những khuyết điểm của nó là bị ảnh hưởng nhiều bởi nhiễu, do đó cần phải thay thế cường độ pixel riêng lẻ tại một điểm bằng một số biểu diễn trên một vùng. Các phương pháp đáng chú ý theo hướng này bao gồm BRIEF, BRISK và FREAK, trong đó trong mọi trường hợp, một vector mô tả nhị phân được tính bằng cách so sánh cường độ của một số cặp pixel sau khi áp dụng làm mịn Gaussian để làm giảm độ nhiễu. Li Liu [1] xem xét tác động của việc thay thế các giá trị độ xám riêng lẻ tại các điểm lấy mẫu bằng cách dùng các bộ lọc đơn giản từ ảnh gốc và các vùng trong ảnh gốc.

Bảng 1.2. Một vài biến thể của LBP

Tên biến thể	Đặc điểm	Mục tiêu
Extended LBP	Sử dụng thêm một số đơn vị nhị phân bổ sung	Nâng cao khả năng phân biệt
Completed LBP	Xem xét cả dấu và độ lớn của các mẫu cục bộ	Nâng cao khả năng phân biệt
Soft LBP	Kết hợp các thành viên trong việc biểu diễn các mẫu cục bộ	Tăng cường khả năng chống nhiễu
Local Tenary Binary	Sử dụng ngưỡng để tạo sự khác biệt	Tăng cường khả năng chống nhiễu

Đặc trưng của ELBP được thay đổi từ cường độ pixel tại các điểm riêng lẻ được thay thế bằng phản hồi từ các bộ lọc $\phi()$. Hình 1.6 thể hiện sự khác nhau giữa 2 phiên bản. Từ đó, định nghĩa ra Robust Extended Local Binary Pattern (RELBP) với các mô tả như sau:

1. Biểu diễn điểm ảnh trung tâm

$$RELBP_CI(x_c) = s(\phi(\mathbf{X}_{c,w}) - \mu_w) \quad (1.5)$$

$\left\{ \begin{array}{l} \phi(\mathbf{X}_{c,w}) \text{ là kết quả sau khi áp dụng bộ lọc lên điểm ảnh } X, \\ \text{Vùng cục bộ có kích thước } w \times w \text{ ở xung quanh vị trí điểm ảnh trung tâm}, \\ \mu_w \text{ kí hiệu cho giá trị trung bình của } \phi(X_{c,w}) \text{ trong toàn bộ bức ảnh.} \end{array} \right.$

2. Biểu diễn các giá trị lân cận

$$\begin{aligned} RELBP_NI_{r,p}(x_c) &= \sum_{n=0}^{p-1} s(\phi(\mathbf{X}_{r,p,w_r,n}) - \mu_{r,p,w_r}) 2^n \\ \mu_{r,p,w_r} &= \frac{1}{p} \sum_{n=0}^{p-1} \phi(\mathbf{X}_{r,p,w_r,n}) \end{aligned} \quad (1.6)$$

3. Biểu diễn chênh lệch hướng tâm

$$RELBP_RD_{r,r-1,p,w_r,w_{r-1}}(x_c) = \sum_{n=0}^{p-1} s(\phi(\mathbf{X}_{r,p,w_r,n}) - \phi(\mathbf{X}_{r-1,p,w_{r-1},n})) 2^n \quad (1.7)$$

Các công thức 1.5, 1.6, 1.7 là cách thực hiện tính toán đặc trưng với các thông tin

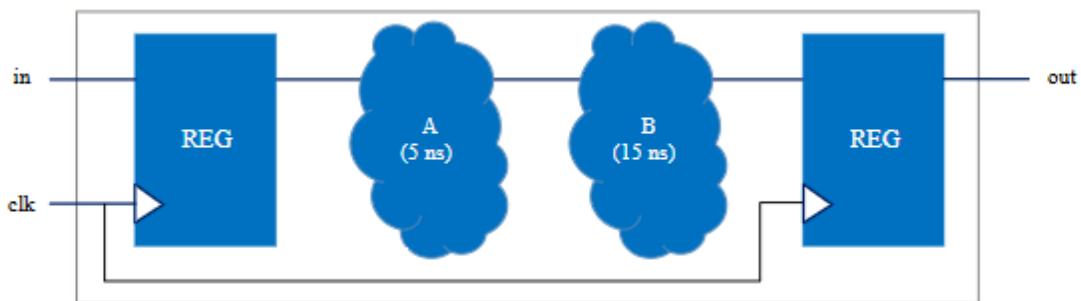
có trong ảnh. Median Robust Extended Local Binary Pattern là phiên bản RELBP với bộ lọc được sử dụng là median. Phương pháp này có ưu điểm so với LBP truyền thống là mạnh hơn trước nhiều và sự thay đổi của ánh sáng. Nó nắm bắt thông tin kết cấu và thông tin không gian tốt hơn thông qua bộ lọc trung vị và mở rộng mẫu bên trong ảnh. Tuy vậy, MRELBP vẫn tồn tại một số nhược điểm như bị giới hạn ở các mẫu cục bộ. Tuy đã được mở rộng nhưng các kỹ thuật LBP vẫn tập trung vào các chi tiết cục bộ mà có thể bỏ sót các thay đổi ở kết cấu quy mô lớn. Kích thước bán kính cố định cũng là một nhược điểm, làm nó thiếu đi tính linh hoạt ở nhiều tỷ lệ khác nhau.

1.3. Kiến trúc xử lý và giao tiếp dữ liệu

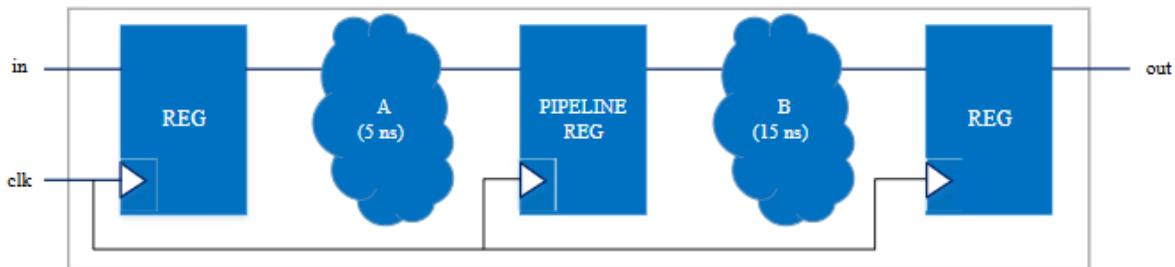
1.3.1. Pipelining

Kỹ thuật đường ống (Pipelining) là một kỹ thuật được sử dụng trong các mạch số đồng bộ để tăng tần số hoạt động tối đa. Kỹ thuật này liên quan đến việc chèn thêm các thanh ghi ở các đường dẫn quan trọng, làm giảm số lượng logic giữa mỗi thanh ghi. Ít logic hơn sẽ tiêu tốn ít thời gian để thực hiện hơn, từ đó tăng tần số hoạt động tối đa.

Đường dẫn quan trọng (critical path) trong mạch là đường dẫn giữa 2 thanh ghi liên tiếp với độ trễ cao nhất. Điều đó cũng tương đương với việc thời gian để có giá trị đầu ra giữa 2 thanh ghi này là lâu nhất. Hình 1.7 mô tả mạch trước khi được pipelining. Trong đó, thời gian thực hiện 2 mạch tổ hợp giữa 2 thanh ghi là thời gian của mạch A cộng với mạch B. Do đó, thời gian tính toán tổ hợp tổng 20ms. Ở hình 1.8 đã thực hiện pipelining bằng cách chèn vào giữa A và B một thanh ghi, do đó thời gian tối đa trong tính toán tổ hợp chỉ là 15ms. Do đó, kỹ thuật pipelining sẽ giúp tăng lên tần số hoạt động tối đa của mạch.



Hình 1.7. Mạch trước khi pipelining

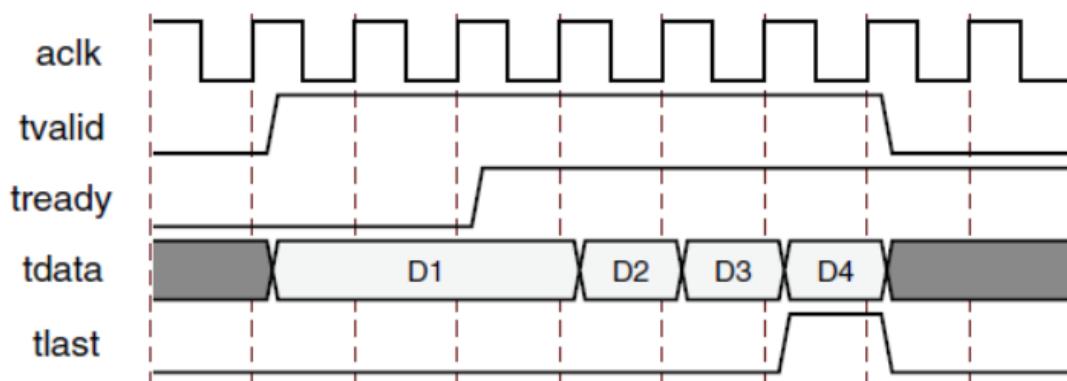


Hình 1.8. Mạch sau khi pipelining

1.3.2. Giao diện AXI4-Stream

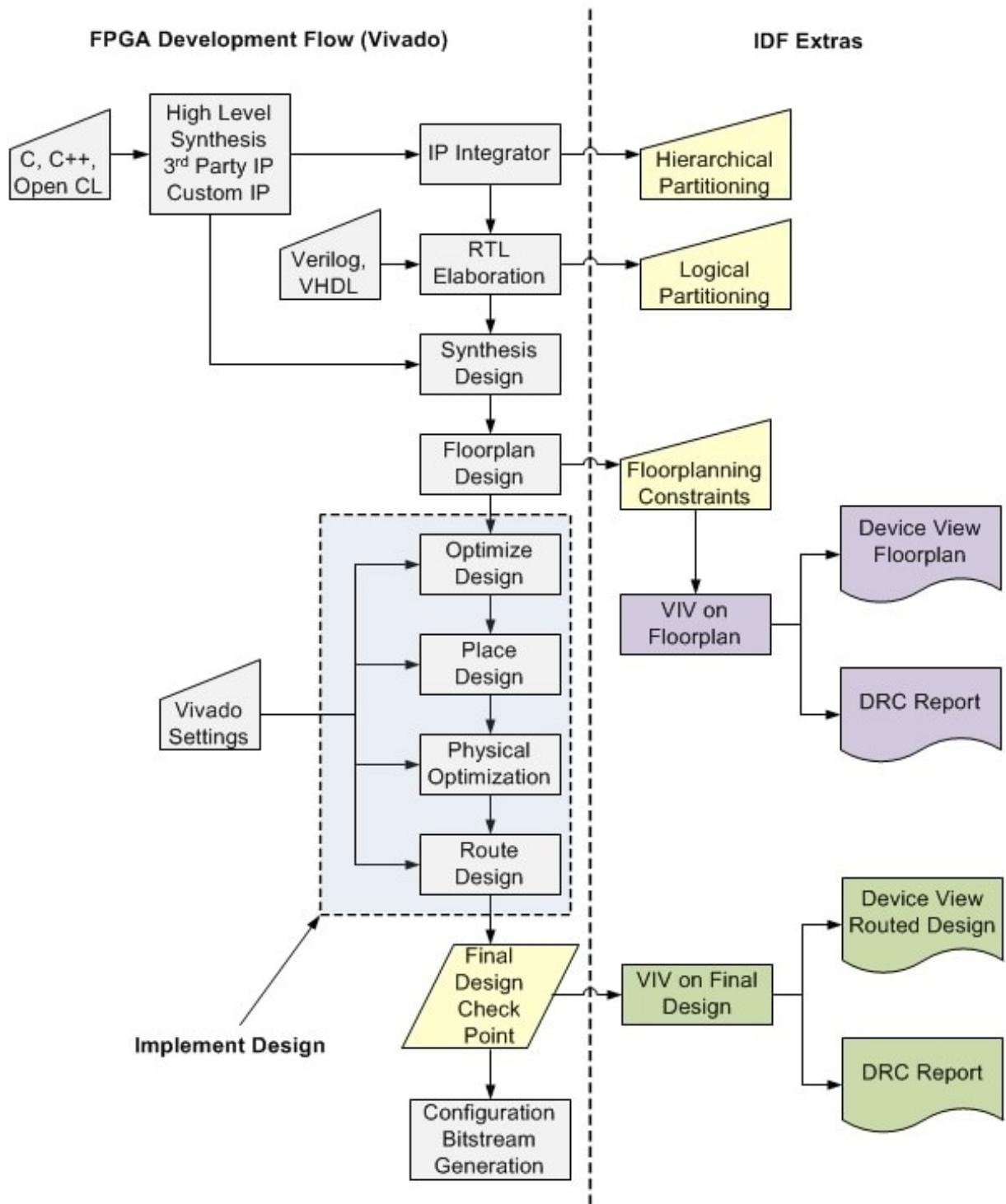
Trong một hệ thống System on Chip (SoC), BUS là thành phần chính kết nối các thiết bị chính và thiết bị phụ. Nó giúp các thiết bị chính có thể truy xuất (đọc/ghi) các slave bằng cách chuyển thông tin điều khiển từ thiết bị chính đến thiết bị phụ, đồng thời chuyển dữ liệu và thông tin phản hồi từ thiết bị phụ đến thiết bị chính. AXI (Advanced eXtensible Interface) là một trong các giao thức BUS trong họ AMBA (Advanced Microcontroller Bus Architecture), được phát triển bởi hãng ARM.

AXI4-Stream là một liên kết điểm tới điểm, bên gửi là thiết bị chính còn bên nhận là thiết bị phụ. Để giao tiếp được giữa hai bên, cần một quá trình gọi là bắt tay. Hình 1.9 mô tả quá trình truyền dữ liệu đối với kênh AXI4-Stream. Tín hiệu **tvalid** được phát bởi bên truyền và **tready** được phát bởi bên nhận. Tín hiệu **tvalid** chỉ ra rằng dữ liệu **tdata** là hợp lệ. Tín hiệu **tready** chỉ ra rằng bên nhận sẵn sàng để nhận dữ liệu. Khi mà cả **tvalid** và **tready** đều ở mức cao thì quá trình truyền diễn ra.



Hình 1.9. Truyền dữ liệu ở kênh AXI4-Stream

1.4. Phần mềm thiết kế và mô phỏng Vivado



Hình 1.10. Chu trình thiết kế và tổng hợp của Vivado

Vivado Design Suite là một tổ hợp các phần mềm của hãng Xilinx để tổng hợp và phân tích các thiết kế từ ngôn ngữ mô tả phần cứng. Trình mô phỏng Vivado là một thành phần của Vivado Design Suite, dùng để mô phỏng với các ngôn ngữ mô tả phần

cứng được biên dịch, trong đó hỗ trợ tập lệnh TCL, IP đã thiết kế và xác minh nâng cao. Hình 1.10 mô tả chu trình phát triển FPGA sử dụng phần mềm Xilinx Vivado. Đối với các phương pháp tổng hợp mức cao, mô-đun sẽ được thiết kế sử dụng những ngôn ngữ bậc cao như C, C++... Thông qua các trình tổng hợp mức cao để có được những thiết kế có thể tổng hợp. Đối với phương pháp thiết kế ở mức RTL (Register Transfer Level), Vivado hỗ trợ các trình biên dịch có thể biên dịch với các ngôn ngữ mô tả phần cứng bao gồm SystemVerilog, Verilog và VHDL. Sau khi có thiết kế, có thể tiến hành tổng hợp tạo thành các mạng lưới cổng logic (netlist) cụ thể cho thiết bị FPGA. Sau đó đến các quá trình thực thi (implementation) biến các thiết kế logic thành các bộ trí cụ thể trên FPGA. Sau khi đã đảm bảo các yếu tố về thời gian, DRC (Design Rule Check) có thể tiến hành tạo ra các bitstream để cấu hình cho FPGA.

CHƯƠNG 2

ĐẶC TẢ KỸ THUẬT

Chương này sẽ cung cấp các thông tin về các yêu cầu chức năng và phi chức năng của IP cần thiết kế, từ mô-đun lớn nhất đến các mô-đun nhỏ hơn. Các thông tin về đặc tả kỹ thuật sẽ bao gồm thông tin về chức năng, danh sách tham số và tín hiệu của mô-đun, kiến trúc ở mức hành vi và mô tả dạng sóng của mô-đun.

2.1. Yêu cầu hệ thống

Bảng 2.1 đây mô tả các yêu cầu hệ thống cho IP (bao gồm yêu cầu chức năng và yêu cầu phi chức năng):

2.2. Xác định thiết kế

2.2.1. *Tối ưu thuật toán phù hợp với phần cứng*

Đồ án sẽ xây dựng IP trích xuất đặc trưng sử dụng thuật toán Median Robust Extended Local Bianry Pattern. Đây là một thuật toán hoạt động tương đối phức tạp với 3 loại đặc trưng được tính toán theo các công thức 1.5, 1.6, 1.7. Tuy nhiên, đối với công thức 1.5, việc phải lưu lại toàn bộ bức ảnh để tính trung bình sẽ gây ảnh hưởng lớn về mặt thời gian và lưu trữ, do đó, để cải tiến, ta sẽ có mô tả như sau:

$$RELBP_CI(x_c) = s(\phi(\mathbf{X}_{c,w}) - \mu_w) \quad (2.1)$$

$\phi(\mathbf{X}_{c,w})$ là kết quả sau khi áp dụng bộ lọc lên điểm ảnh X ,
Vùng cục bộ có kích thước $w \times w$ ở xung quanh vị trí điểm ảnh trung tâm,
 μ_w kí hiệu cho giá trị trung bình của $\phi(X_{c,w})$ trong một vùng cục bộ kích thước $w \times w$.

Đối với đặc trưng NI, sinh viên sẽ tính trung bình của một cửa sổ ảnh thay thế cho trung bình của p điểm ảnh trong một cửa sổ. Hình 2.1 mô tả phiên bản phù hợp hơn của biểu diễn điểm ảnh trung tâm (CI) với việc thay đổi từ tính toán trung bình của toàn bộ bức ảnh thành tính toán trung bình của một vùng nội bộ trong bức ảnh. Điều này sẽ làm tăng tính xử lý thời gian thực của thiết kế lên vì không cần phải lưu lại toàn bộ bức ảnh. Kết quả nghiên cứu của Wang và Zhang [16] đã cho thấy sự thay đổi về tính toán không gây ra sai số quá lớn về độ chính xác, giữ nguyên với Outex_TC10 với, giảm 0.052% với

Bảng 2.1. Các yêu cầu phi chức năng cho thiết kế

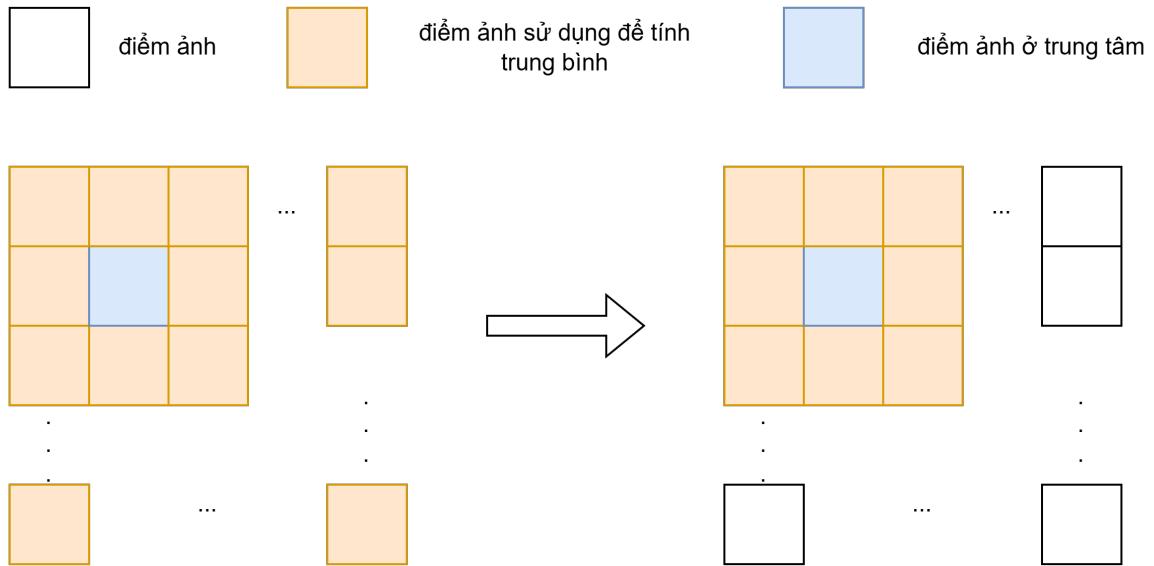
Yêu cầu	Mô tả
Chức năng	Thực hiện trích xuất đặc trưng sử dụng thuật toán MRELBP
Thông số kỹ thuật	Các thông số thuật toán: $r = 2, 4, 6$; $p = 8$ Độ rộng dữ liệu: Đầu vào 8-bit, đầu ra 32-bit Kích thước ảnh đầu vào: có thể tùy chỉnh theo tham số cấu hình Số lượng dữ liệu đầu ra: 600 dữ liệu Độ chính xác: dấu phẩy tĩnh (16-bit thập phân, 8-bit phần nguyên)
Hiệu suất (Performance)	Tần số hoạt động tối thiểu 100 MHz
Tính đồng bộ	Tất cả các tín hiệu đồng bộ theo 1 clock chính
Reset	Tất cả các mô-đun tuân theo reset đồng bộ
Tài nguyên sử dụng	Tối đa 30000 LUTs, 30000 Registers, 200 DSPs, 100 BRAM
Khả năng kiểm thử	Hỗ trợ kiểm thử test-bench ngẫu nhiên và test-bench trực tiếp
Chuẩn giap tiếp	Có thể tương thích với chuẩn AXI4-Stream
Tương thích công cụ	Có thể tổng hợp trên phần mềm như Vivado
Khả năng tích hợp	Kiểm nghiệm trên hệ thống SoC
Độ bao phủ	Tối thiểu 95% về mặt chức năng
Kiểm thử tự động	Xây dựng hệ thống giúp kiểm thử về mặt chức năng một cách tự động

tập Outex_TC12_000 và giảm 0.104% với tập Outex_TC12_002.

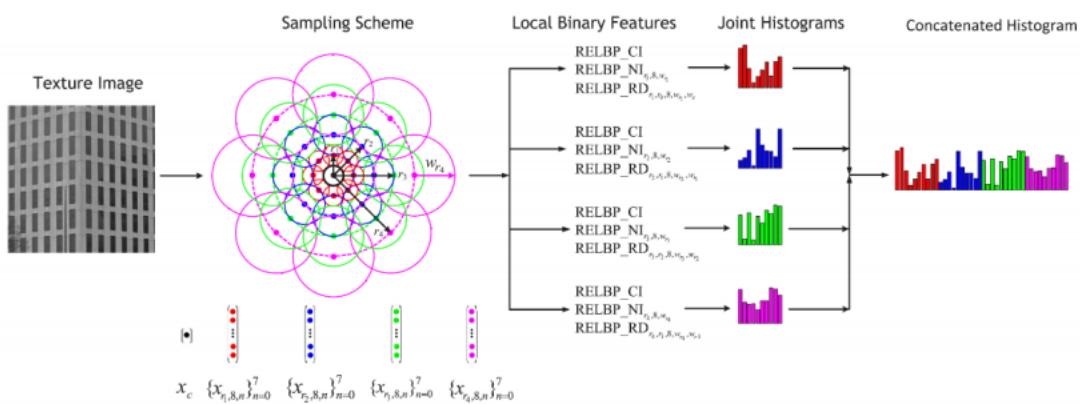
2.2.2. Sơ đồ triển khai thuật toán

Quá trình tạo ra biểu đồ histogram được mô tả theo hình 2.2. Với đầu vào là ảnh kết cấu, sau quá trình lấy mẫu theo các công thức 2.1, 1.6, 1.7 ta được các mô tả thô của ảnh. Sau đó, thực hiện một giai đoạn là "Joint Histogram", tại giai đoạn này, thực tế ta sẽ đếm xem là mô tả đầy xuất hiện bao nhiêu lần, do đó kết quả đạt được là một biểu đồ histogram. Sau đó ta sẽ nối lần lượt các biểu đồ này lại để được đặc trưng cuối cùng.

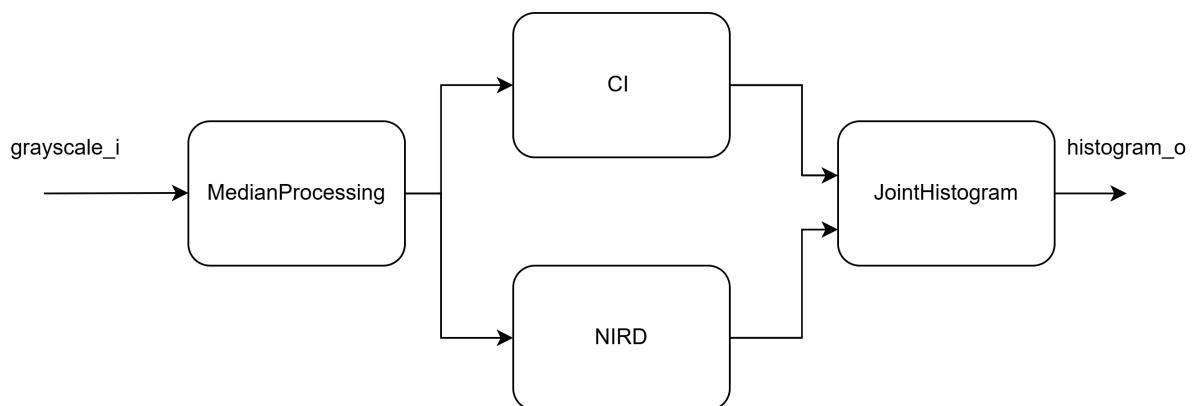
Với những mô tả trên, sơ đồ thuật toán được đưa ra như hình 2.3. Một cách cụ thể hơn, vì MRELBP là phiên bản của RELBP với $\phi()$ là bộ tính trung vị, nên khi thiết kế



Hình 2.1. Phiên bản thiết kế phần cứng của biểu diễn điểm ảnh trung tâm - CI



Hình 2.2. Quá trình tạo ra đặc trưng của thuật toán RELBP [16]



Hình 2.3. Sơ đồ kiến trúc tổng quát của MRELBPIP

phân cứng, ta sẽ đưa ra toàn bộ các phiên bản tính trung vị của ảnh đầu vào. Sau đó, sẽ có 2 bộ là CI và NIRD để tính các giá trị đặc trưng theo mô tả công thức 2.1, 1.6, 1.7. Vì NIRD có cách tính và yêu cầu đầu vào tương đối giống nhau, do đó, 2 mô tả này có thể xây dựng trong cùng 1 mô-đun để tiết kiệm tài nguyên. Sau đó 3 mô tả này sẽ thực hiện qua bước "JointHistogram" để đạt được đặc trưng đầu ra.

2.3. Đặc tả kỹ thuật

2.3.1. Đặc tả mô-đun MRELBP

Bảng 2.2, 2.3 mô tả các tham số và tín hiệu đầu vào, đầu ra của mô-đun MRELBP.

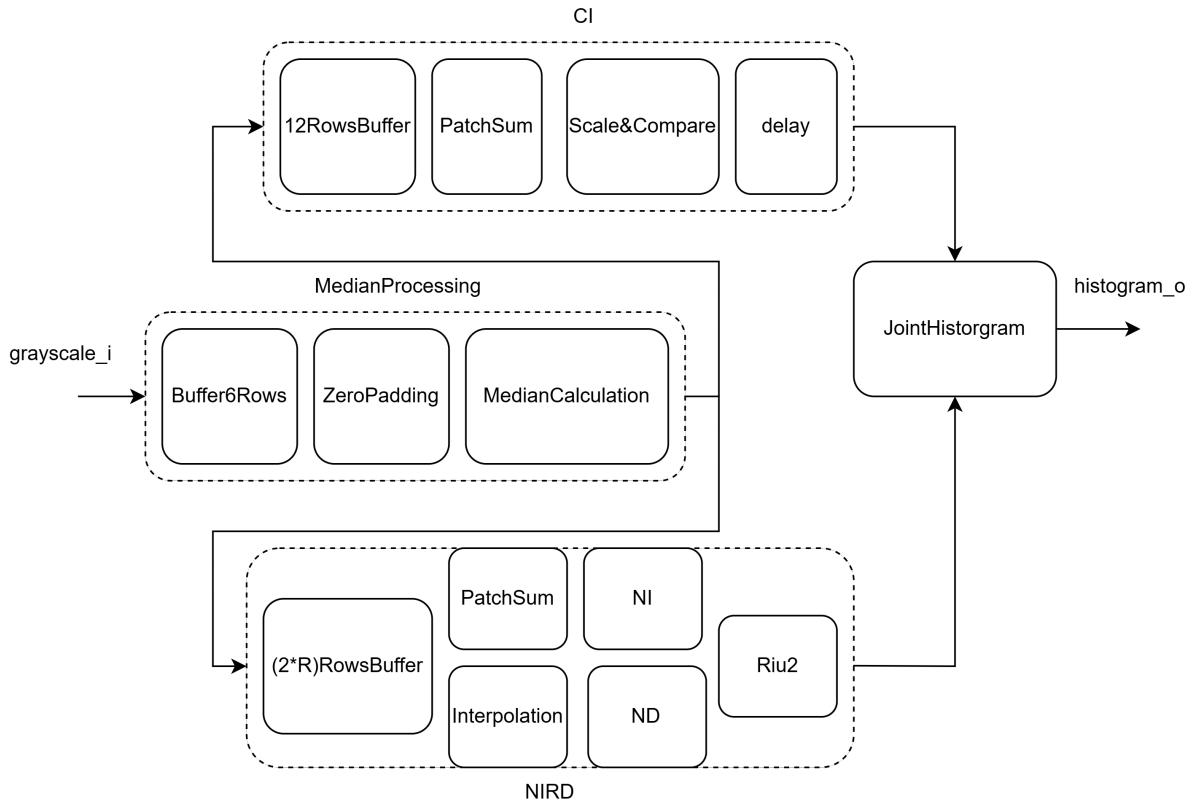
Bảng 2.2. Danh sách các tham số của mô-đun MRELBP

Tham số	Giá trị mặc định	Mô tả
ROWS	128	Kích thước chiều cao của ảnh
COLS	128	Kích thước chiều rộng của ảnh

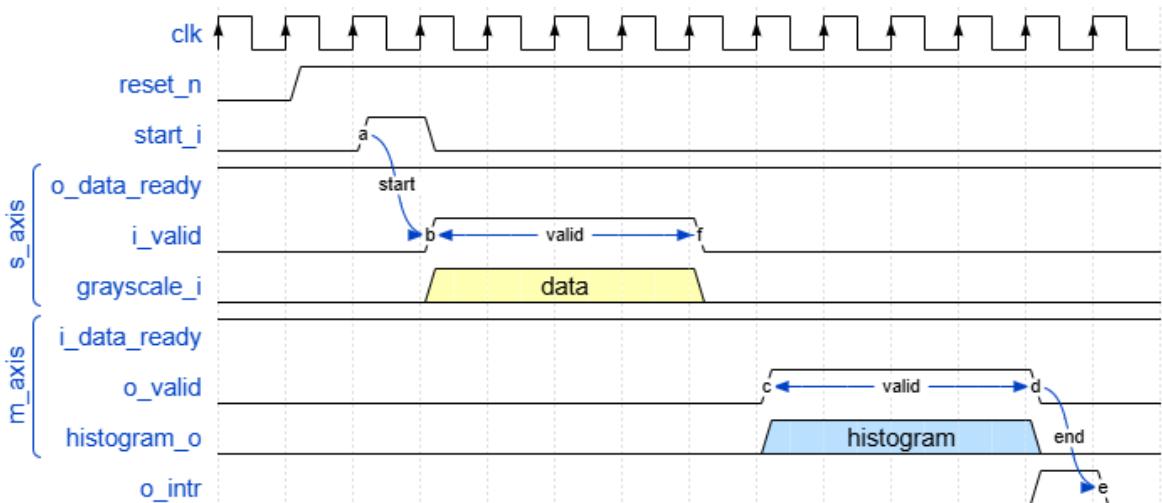
Bảng 2.3. Danh sách các tín hiệu của giao diện mô-đun MRELBP

Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Reset đồng bộ, kích hoạt mức thấp
start_i	1	Vào	Tín hiệu bắt đầu (liên quan đến đầu ra)
grayscale_i	8	Vào	Dữ liệu điểm ảnh
i_valid	1	Vào	Tín hiệu báo hiệu dữ liệu đầu vào là hợp lệ
o_data_ready	1	Ra	Tín hiệu thông báo sẵn sàng cho đầu ra
histogram_o	32	Ra	Đặc trưng histogram
o_valid	1	Ra	Tín hiệu báo hiệu dữ liệu đầu ra là hợp lệ
i_data_ready	1	Vào	Tín hiệu thông báo sẵn sàng nhận dữ liệu
o_intr	1	Ra	Tín hiệu báo hiệu đã xử lý xong

Hình 2.4 mô tả sơ đồ khối của mô-đun MRELBP một cách chi tiết hơn. mô-đun **MedianProcessing** được cấu thành từ mô-đun Buffer6Rows, mô-đun ZeroPadding và mô-đun MedianCalculation. Dữ liệu đầu ra là giá trị trung vị của ảnh, sau đó sẽ đi đến các mô-đun sau là **CI** và **NIRD**. Mô tả chi tiết hơn về đường đi của dữ liệu sẽ được giới thiệu tại các phần sau.



Hình 2.4. Sơ đồ khối của mô-đun MRELBP



Hình 2.5. Dạng sóng của mô-đun MRELBP

2.3.2. Đặc tả mô-đun MedianProcessing

Theo mô tả từ hình 2.4 thì mô-đun MedianProcessing gồm 3 thành phần chính hay 3 mô-đun bao gồm **Buffer6Rows**, **ZeroPadding**, **MedianCalculation**. **Lưu ý:** Do mô-đun MedianProcessing sẽ tính giá trị trung vị ứng với 3 cửa sổ theo bán kính r , nên tại

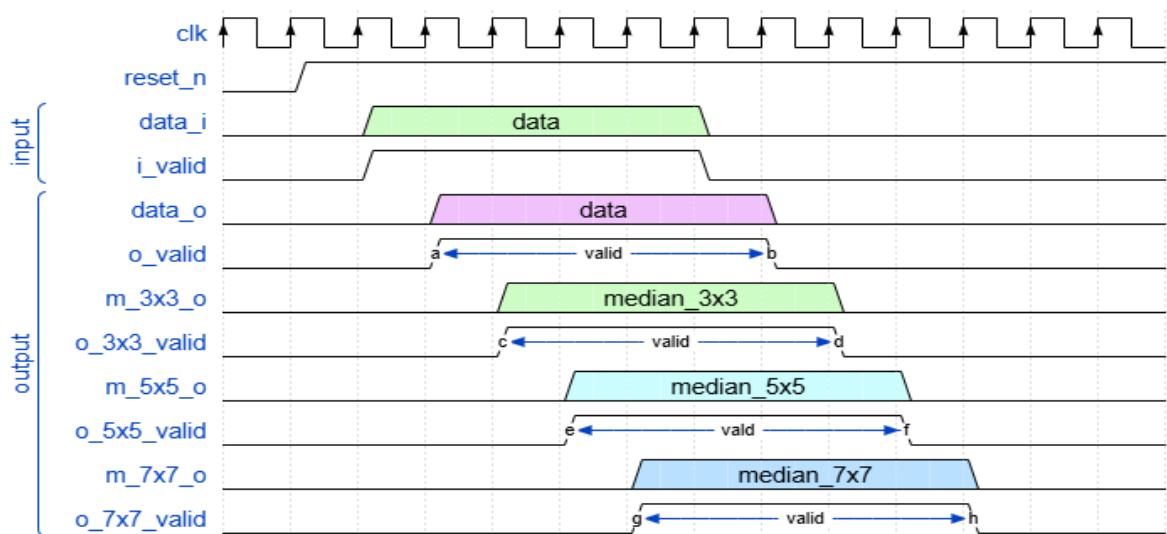
phân này sẽ mô tả một cách tổng quát từng module mà không chi tiết cụ thể theo từng giá trị r.

Bảng 2.4. Tham số của module MedianProcessing

Tham số	Giá trị mặc định	Mô tả
COLS	128	Kích thước độ rộng của ảnh
ROWS	128	Kích thước chiều cao của ảnh

Bảng 2.5. Danh sách các tín hiệu của module MedianProcessing

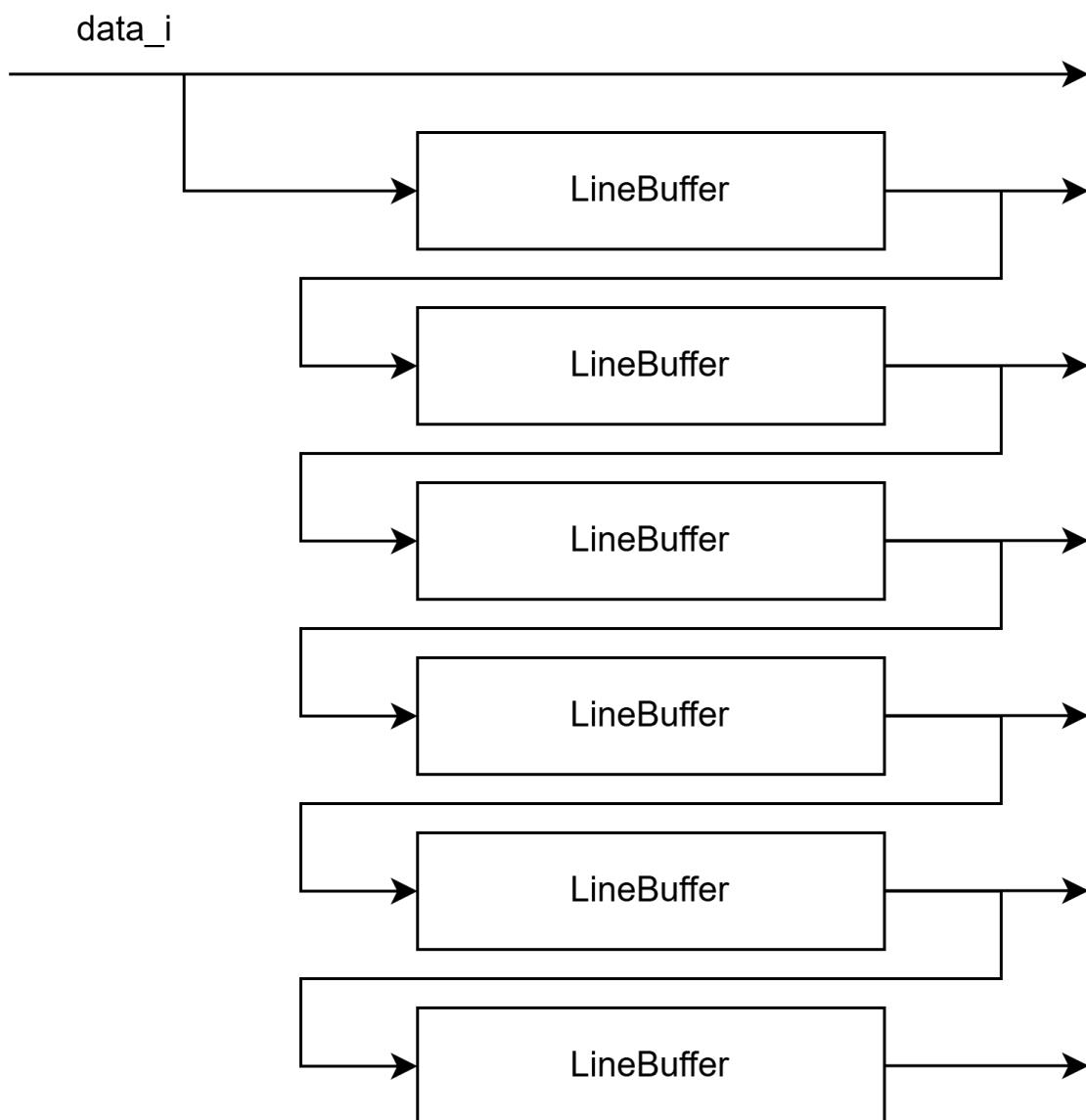
Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Reset đồng bộ, kích hoạt mức thấp
data_i	8	Vào	Dữ liệu điểm ảnh đầu vào
i_valid	1	Vào	Tín hiệu thông báo giá trị đầu vào là hợp lệ
data_o	8	Ra	Dữ liệu điểm ảnh đầu ra gốc
o_valid	1	Ra	Tín hiệu thông báo dữ liệu đầu ra gốc là hợp lệ
m_3x3_o	8	Ra	Giá trị trung vị của cửa sổ 3x3
o_3x3_valid	1	Ra	Tín hiệu thông báo dữ liệu trung vị 3x3 hợp lệ
m_5x5_o	8	Ra	Giá trị trung vị của cửa sổ 5x5
o_5x5_valid	1	Ra	Tín hiệu thông báo dữ liệu trung vị 5x5 hợp lệ
m_7x7_o	8	Ra	Giá trị trung vị của cửa sổ 7x7
o_7x7_valid	1	Ra	Tín hiệu thông báo dữ liệu trung vị 7x7 hợp lệ



Hình 2.6. Dạng sóng của module MedianProcessing

2.3.2.1. Đặc tả mô-đun Buffer6Rows

mô-đun Buffer6Rows là tập hợp của **6** mô-đun LineBuffer tạo thành, cụ thể được mô tả trong hình 2.7. Do đó, để mô tả đặc tả cho mô-đun Buffer6Rows, ta sẽ đi từ mô-đun LineBuffer. mô-đun LineBuffer là mô-đun cơ sở để xây dựng lên các mô-đun lớn hơn như Buffer6Rows. Chức năng của mô-đun này là đệm dữ liệu theo từng hàng để thuận tiện cho việc định dạng dữ liệu cho các mô-đun đứng sau.



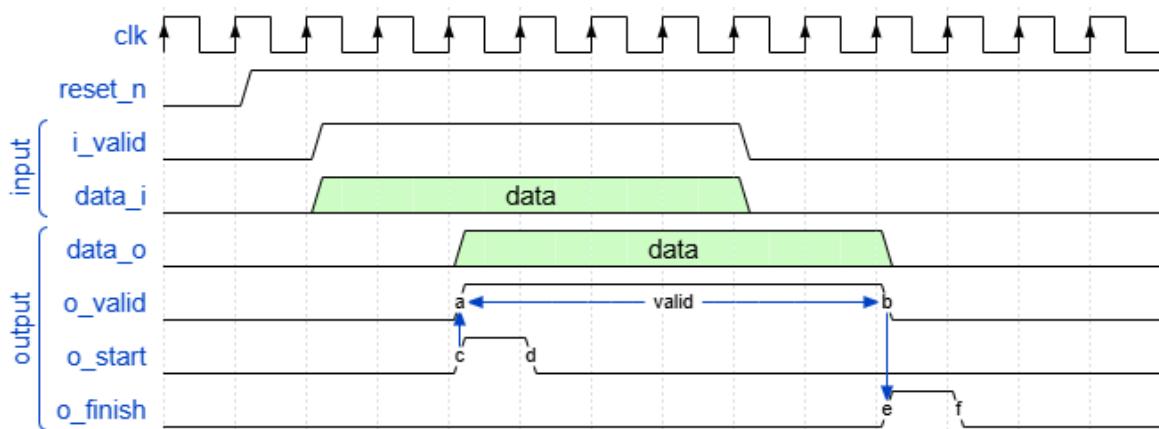
Hình 2.7. Sơ đồ khái niệm của mô-đun Buffer6Rows

Bảng 2.6. Danh sách các tham số của mô-đun LineBuffer

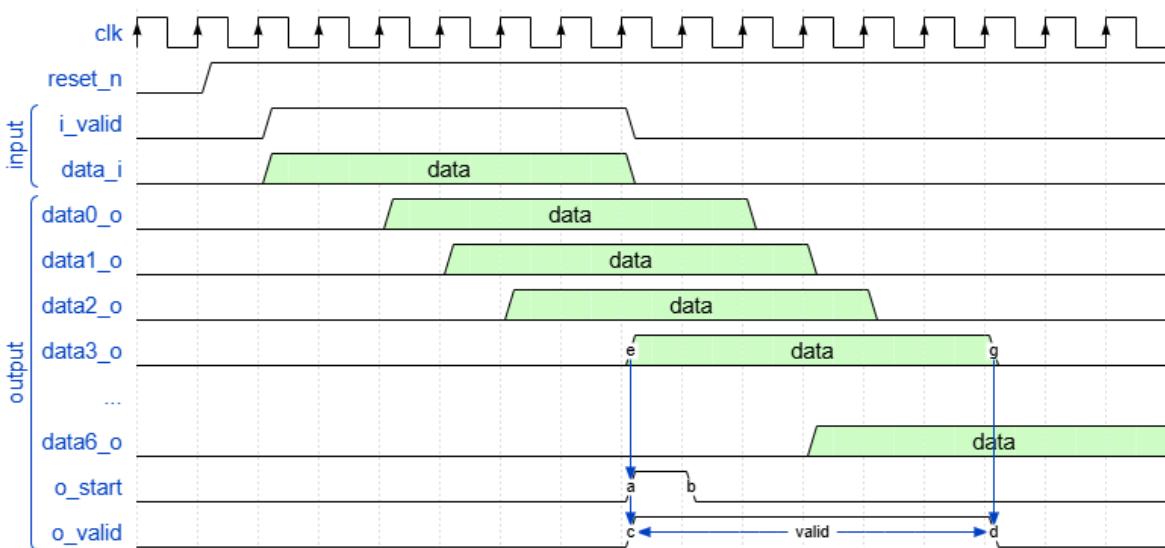
Tham số	Giá trị mặc định	Mô tả
DEPTH	128	Kích thước độ rộng của ảnh

Bảng 2.7. Danh sách các tín hiệu của giao diện mô-đun LineBuffer

Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Reset đồng bộ, kích hoạt mức thấp
i_valid	1	Vào	Tín hiệu thông báo dữ liệu đầu vào là hợp lệ
data_i	8	Vào	Dữ liệu đầu vào
data_o	8	Ra	Dữ liệu đầu ra
o_start	1	Ra	Tín hiệu thông báo bắt đầu có dữ liệu đầu ra
o_valid	1	Ra	Tín hiệu thông báo dữ liệu đầu là hợp lệ
o_finish	1	Ra	Tín hiệu thông báo kết thúc đệm dữ liệu



Hình 2.8. Dạng sóng của mô-đun LineBuffer



Hình 2.9. Dạng sóng của mô-đun Buffer6Rows

2.3.2.2. Đặc tả mô-đun ZeroPadding

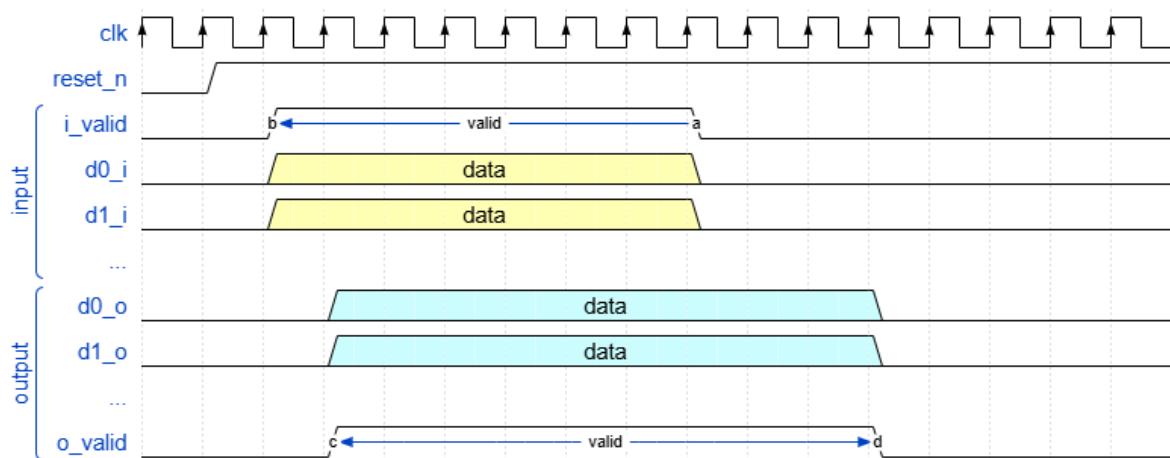
Đệm 0 là một kỹ thuật được sử dụng để thêm hàng và cột vào rìa bức ảnh, mục đích để khi xử lý với một số kỹ thuật như nhân chập, kích thước đầu ra sẽ không bị thay đổi so với đầu vào (*danh sách tín hiệu sẽ có mô tả chung về số lượng đầu vào&ra vì các tín hiệu của mô-đun sẽ thuộc vào kích thước cửa sổ, xem thêm ở phần RTL của ZeroPadding*).

Bảng 2.8. Danh sách các tham số của mô-đun ZeroPadding

Tham số	Giá trị mặc định	Mô tả
COLS	128	Kích thước độ rộng của ảnh
ROWS	128	Kích thước độ cao của ảnh

Bảng 2.9. Danh sách các tín hiệu của giao diện mô-đun ZeroPadding

Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Reset đồng bộ, kích hoạt mức thấp
i_valid	1	Vào	Tín hiệu thông báo dữ liệu đầu vào là hợp lệ
d0_i	8	Vào	Dữ liệu đầu vào
d1_i	8	Vào	Dữ liệu đầu vào
Tùy thuộc vào kích thước cửa sổ, 3x3 thì sẽ có 3 đầu vào, 5x5 có 5 đầu vào, 7x7 có 7 đầu vào			
d0_o	8	Ra	Dữ liệu đầu ra
d1_o	8	Ra	Dữ liệu đầu ra
Tùy thuộc vào kích thước cửa sổ, 3x3 thì sẽ có 9 đầu ra, 5x5 có 25 đầu ra, 7x7 có 49 đầu ra			
o_valid	1	Ra	Tín hiệu thông báo dữ liệu đầu là hợp lệ



Hình 2.10. Dạng sóng của mô-đun ZeroPadding

2.3.2.3. Đặc tả mô-đun MedianCalculation

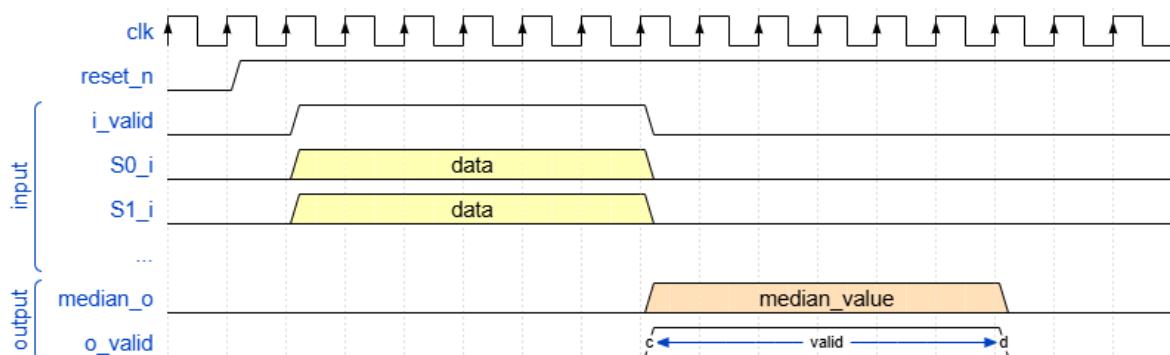
Vì kiến trúc của mô-đun MedianCalculation sẽ phụ thuộc vào từng loại cửa sổ, tức là cửa sổ 3×3 sẽ có cách tính khác cửa sổ 5×5 , nên để thuận lợi cho việc mô tả, sinh viên sẽ mô tả theo 1 giao diện chung mà số lượng các cổng sẽ khác đối với từng loại cửa sổ. Cụ thể hơn sẽ được mô tả tại thiết kế RTL cho mô-đun MedianCalculation

Bảng 2.10. Danh sách các tham số của mô-đun MedianCalculation

Tham số	Giá trị mặc định	Mô tả
COLS	128	Kích thước độ rộng của ảnh
ROWS	128	Kích thước độ cao của ảnh

Bảng 2.11. Danh sách các tín hiệu của giao diện mô-đun MedianCalculation

Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Reset đồng bộ, kích hoạt mức thấp
i_valid	1	Vào	Tín hiệu thông báo dữ liệu đầu vào là hợp lệ
S0_i	8	Vào	Dữ liệu đầu vào
S1_i	8	Vào	Dữ liệu đầu vào
Tùy thuộc vào kích thước cửa sổ, 3×3 thì sẽ có 9 đầu vào, 5×5 có 25 đầu vào, 7×7 có 49 đầu vào			
median_o	8	Ra	Dữ liệu đầu ra
o_valid	1	Ra	Tín hiệu thông báo dữ liệu đầu là hợp lệ



Hình 2.11. Dạng sóng của mô-đun MedianCalculation

2.3.3. Đặc tả mô-đun CI

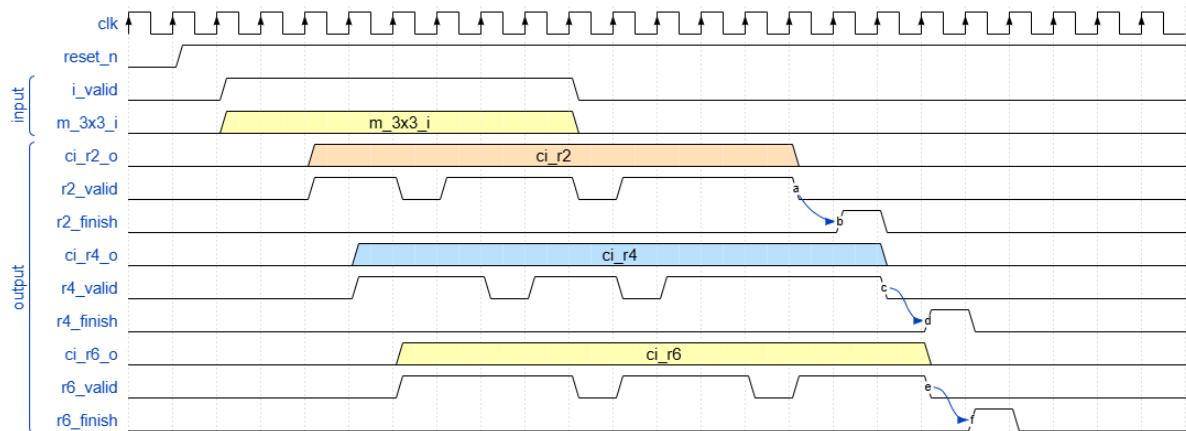
Bảng 2.12, 2.13 mô tả các tham số và tín hiệu đầu vào, đầu ra của mô-đun CI.

Bảng 2.12. Tham số của mô-đun CI

Tham số	Giá trị mặc định	Mô tả
COLS	128	Kích thước độ rộng của ảnh
ROWS	128	Kích thước chiều cao của ảnh

Bảng 2.13. Danh sách các tín hiệu của mô-đun CI

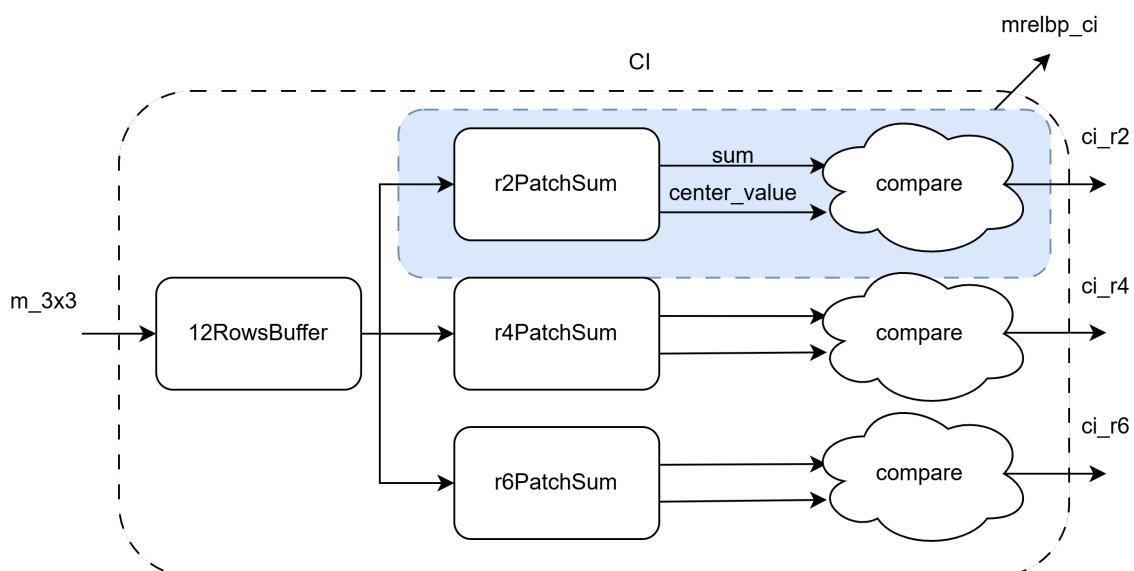
Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Reset đồng bộ, kích hoạt mức thấp
m_3x3_i	8	Vào	Dữ liệu trung vị của cửa sổ 3x3
ci_r2_o	1	Ra	Mô tả ci đầu ra ứng với $r = 2$
r2_valid	1	Ra	Tín hiệu thông báo ci ứng với $r = 2$ là hợp lệ
r2_finish	1	Ra	Tín hiệu thông báo đã kết thúc quá trình tính ci ứng với $r = 2$
ci_r4_o	1	Ra	Mô tả ci đầu ra ứng với $r = 4$
r4_valid	1	Ra	Tín hiệu thông báo ci ứng với $r = 4$ là hợp lệ
r4_finish	1	Ra	Tín hiệu thông báo đã kết thúc quá trình tính ci ứng với $r = 4$
ci_r6_o	1	Ra	Mô tả ci đầu ra ứng với $r = 6$
r6_valid	1	Ra	Tín hiệu thông báo ci ứng với $r = 6$ là hợp lệ
r6_finish	1	Ra	Tín hiệu thông báo đã kết thúc quá trình tính ci ứng với $r = 6$



Hình 2.12. Dạng sóng của mô-đun CI

Hình 2.13 mô tả kiến trúc tổng quát của mô-đun CI. Sau khi giá trị trung vị tính từ cửa sổ 3x3 được tính xong sẽ đưa vào mô-đun này. Trong đây sẽ cần một bộ 12Rows-Buffer để đệm dữ liệu nhằm tạo cửa sổ cho khôi sau sử dụng với kích thước là $(2*r+1)$,

2^*r+1), vì $r = 6$ nên cửa sổ lớn nhất là 13×13 , do đó nên cần đệm 12 hàng. Dữ liệu sau khi đệm sẽ đưa vào một khối gọi là "PatchSum", khối này có nhiệm vụ ghép dữ liệu thành một cửa sổ, sử dụng kỹ thuật cửa sổ trượt để tính tổng của cửa sổ và đầu ra sẽ bao gồm tổng và giá trị ở giữa của cửa sổ. Sau đó khối compare là mạch logic tổ hợp có thể so sánh và đưa ra giá trị CI ứng với cửa sổ đang được tính. Hình 2.4 có mô tả thêm khối "**delay**", thì khối này thực tế để làm trễ dữ liệu CI đầu ra để đồng bộ với NIRD, thực tế nó là các thanh ghi dịch, do đó sẽ không trình bày phần delay này. mô-đun 12Rows-Buffer có nguyên tắc hoạt động tương tự với mô-đun Buffer6Rows. Phần gộp vào giữa PatchSum và khối logic tổ hợp gọi là *MRELBP_CI*.



Hình 2.13. Sơ đồ khái niệm của mô-đun CI

2.3.3.1. Đặc tả mô-đun *MRELBP_CI*

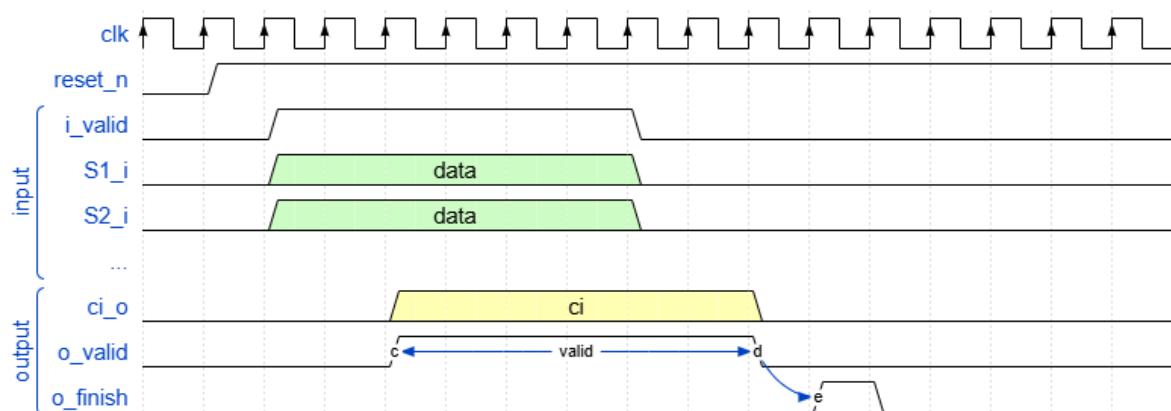
Bảng 2.14, 2.15 mô tả về giao diện kết nối của mô-đun *MRELBP_CI*. Với mô-đun này, sẽ được thiết kế theo nhiều giá trị r , ứng với đó thì sẽ có số lượng đầu vào khác nhau. Đầu ra gồm tín hiệu ci_o , tín hiệu này chỉ có 1-bit ứng với 2 giá trị có thể là 0 hoặc 1. Theo hình 2.13, bên trong mô-đun này sẽ có mô-đun *PatchSum* để tính toán tổng của các điểm ảnh trong cửa sổ và sau đó là một khối logic tổ hợp để so sánh giá trị của **sum** và **center_value** tương ứng với tổng của cửa sổ và điểm ảnh nằm giữa của cửa sổ. Chi tiết hơn sẽ được mô tả tại thiết kế RTL của mô-đun *MRELBP_CI*.

Bảng 2.14. Tham số của mô-đun MRELBP_CI

Tham số	Giá trị mặc định	Mô tả
COLS	128	Kích thước độ rộng của ảnh
ROWS	128	Kích thước chiều cao của ảnh

Bảng 2.15. Danh sách các tín hiệu của mô-đun MRELBP_CI

Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Reset đồng bộ, kích hoạt mức thấp
i_valid	1	Vào	Tín hiệu thông báo dữ liệu đầu vào là hợp lệ
S1_i	8	Vào	Dữ liệu đầu vào 1
S2_i	8	Vào	Dữ liệu đầu vào 2
S3_i	8	Vào	Dữ liệu đầu vào 3
S4_i	8	Vào	Dữ liệu đầu vào 4
S5_i	8	Vào	Dữ liệu đầu vào 5
Tùy thuộc vào giá trị bán kính, r = 2 thì sẽ có 5 đầu vào, r = 4 có 9 đầu vào và r = 6 có 13 đầu vào			
ci_o	1	Ra	Mô tả CI đầu ra
o_valid	1	Ra	Tín hiệu thông báo dữ liệu đầu ra là hợp lệ
o_finish	1	Ra	Tín hiệu thông báo kết thúc quá trình tính CI



Hình 2.14. Dạng sóng của mô-đun MRELBP_CI

2.3.4. Đặc tả mô-đun NIRD

Hình 2.15 mô tả sơ đồ khối của mô-đun NIRD. Tại mô-đun này ta cần tính hai mô tả là NI và RD theo công thức 1.6 và 1.7. Vì khi tính RD cần tham chiếu đến hai cửa sổ kích thước khác nhau và dữ liệu của điểm ảnh cũng là đầu ra của bộ lọc trung vị cửa sổ khác nhau, do đó đầu vào sẽ có 2 dữ liệu ứng bán kính r và r_1 với $r_1 = r - 1$. Dữ liệu cũng

sẽ cần được đếm qua hai bộ đếm có số lượng mô-đun LineBuffer bằng 2^*r với r có thể là 2, 4, 6. Sau đó sẽ đi đến mô-đun WindowBuffer với mục đích để hình thành các cửa sổ dữ liệu để thuận lợi cho việc tính toán của các mô-đun sau. Khi tính mô tả NI sẽ là so sánh 8 giá trị trong đường tròn bán kính r so với trung bình tổng bán kính của cửa sổ, do đó sẽ có đầu vào là tổng điểm ảnh của cửa sổ và 8 điểm ảnh trong đường tròn bán kính r . 8 điểm ảnh này sẽ có 4 điểm cần phải suy ra bằng phương pháp nội suy. Với mô tả RD sẽ là so sánh của 8 điểm ảnh đường tròn bán kính r và 8 điểm ảnh trong đường tròn bán kính r_1 . Dữ liệu sau khi đạt được sẽ cần thực hiện một thao tác gọi là RIU2-mapping hay ảnh xạ RIU2. Thực tế, mô-đun này sẽ làm nhiệm vụ chuyển 8-bit có được sau mô-đun NI hoặc RD thành một dạng biểu diễn khác ít bit hơn và mang nhiều ý nghĩa về mặt đặc trưng. Lý thuyết RIU2 đã được trình bày ở tiểu mục 1.2.4.

Bảng 2.16. Tham số của mô-đun NIRD

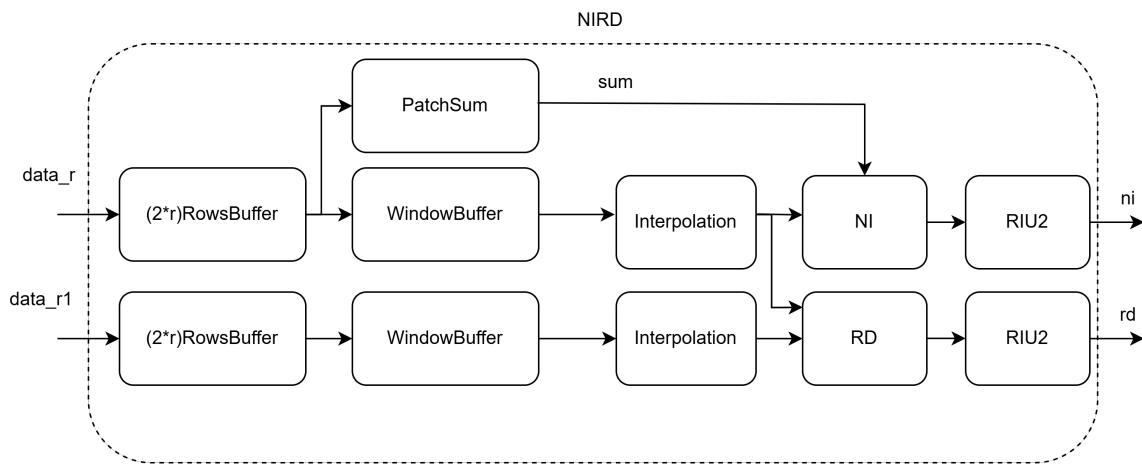
Tham số	Giá trị mặc định	Mô tả
COLS	128	Kích thước độ rộng của ảnh
ROWS	128	Kích thước chiều cao của ảnh

Bảng 2.17. Danh sách các tín hiệu của mô-đun NIRD

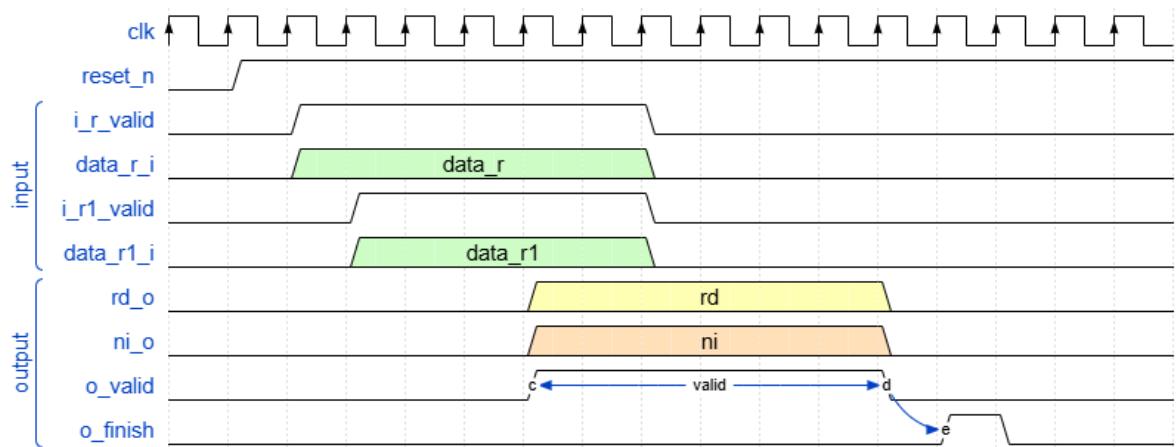
Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Reset đồng bộ, kích hoạt mức thấp
data_r_i	8	Vào	Dữ liệu vào ứng với bán kính r
i_r_valid	1	Vào	Tín hiệu thông báo dữ liệu đầu vào bán kính r hợp lệ
data_r1_i	8	Vào	Dữ liệu vào ứng với bán kính r_1
i_r1_valid	1	Vào	Tín hiệu thông báo dữ liệu vào bán kính r_1 hợp lệ
ni_o	4	Ra	Giá trị mô tả NI đầu ra
rd_o	4	Ra	Giá trị mô tả RD đầu ra
o_valid	1	Ra	Tín hiệu thông báo đầu ra là hợp lệ
o_finish	1	Ra	Tín hiệu thông báo kết thúc đầu ra

2.3.4.1. Đặc tả mô-đun Interpolation

mô-đun này sẽ thực hiện chức năng tính toán giá trị chưa biết ở một vài vị trí bán kính bằng phương pháp nội suy. Vì phương pháp này sẽ xuất hiện các giá trị thập phân,



Hình 2.15. Sơ đồ khối của mô-đun NIRD



Hình 2.16. Dạng sóng của mô-đun NIRD

để tiện cho việc biểu diễn, sinh viên sử dụng dấu phẩy tĩnh với 8-bit phần nguyên và 16-bit phần thập phân, tổng cộng là 24-bit cho dữ liệu đầu ra. Các dữ liệu ở các góc đặc biệt như 0, 90, 180, 270 độ sẽ không cần thực hiện nội suy, do đó, trong mô-đun cũng sẽ thực thi chức năng đệm dữ liệu để đảm bảo đầu ra có đủ 8 giá trị trên đường tròn bán kính r ở tại một thời điểm. Vì phương pháp nội sử dụng là song tuyến tính, do đó cần 4 điểm xung quanh vị trí cần tính, nên ứng với các góc 45, 135, 225 và 315 thì sẽ cần đầy đủ 4 giá trị đầu vào để tính toán.

Bảng 2.18. Tham số của mô-đun Interpolation

Tham số	Giá trị mặc định	Mô tả
R	2	Kích thước bán kính
ANGLE	45	Giá trị góc

Bảng 2.19. Danh sách các tín hiệu của mô-đun Interpolation

Tên tín hiệu	Độ rộng	Vào ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Tín hiệu reset đồng bộ, kích hoạt mức thấp
i_valid	1	Vào	Thông báo rằng dữ liệu đầu vào là hợp lệ
i_finish	1	Vào	Thông báo không còn dữ liệu đầu vào
S_0_i	8	Vào	Dữ liệu đầu vào tại góc 0 độ
S_90_i	8	Vào	Dữ liệu đầu vào tại góc 90 độ
S_180_i	8	Vào	Dữ liệu đầu vào tại góc 180 độ
S_270_i	8	Vào	Dữ liệu đầu vào tại góc 270 độ
S_45_i_1	8	Vào	Dữ liệu đầu vào thứ 1 tại hướng 45 độ
S_45_i_2	8	Vào	Dữ liệu đầu vào thứ 2 tại hướng 45 độ
S_45_i_3	8	Vào	Dữ liệu đầu vào thứ 3 tại hướng 45 độ
S_45_i_4	8	Vào	Dữ liệu đầu vào thứ 4 tại hướng 45 độ
S_135_i_1	8	Vào	Dữ liệu đầu vào thứ 1 tại hướng 135 độ
S_135_i_2	8	Vào	Dữ liệu đầu vào thứ 2 tại hướng 135 độ
S_135_i_3	8	Vào	Dữ liệu đầu vào thứ 3 tại hướng 135 độ
S_135_i_4	8	Vào	Dữ liệu đầu vào thứ 4 tại hướng 135 độ
S_225_i_1	8	Vào	Dữ liệu đầu vào thứ 1 tại hướng 225 độ
S_225_i_2	8	Vào	Dữ liệu đầu vào thứ 2 tại hướng 225 độ
S_225_i_3	8	Vào	Dữ liệu đầu vào thứ 3 tại hướng 225 độ
S_225_i_4	8	Vào	Dữ liệu đầu vào thứ 4 tại hướng 225 độ
S_315_i_1	8	Vào	Dữ liệu đầu vào thứ 1 tại hướng 315 độ
S_315_i_2	8	Vào	Dữ liệu đầu vào thứ 2 tại hướng 315 độ
S_315_i_3	8	Vào	Dữ liệu đầu vào thứ 3 tại hướng 315 độ
S_315_i_4	8	Vào	Dữ liệu đầu vào thứ 4 tại hướng 315 độ
S1_o → S8_o	24	Ra	Dữ liệu nội suy đầu ra hướng từ 0 → 315 độ
o_valid	1	Ra	Thông báo đầu ra là hợp lệ
o_finish	1	Ra	Tín hiệu cho biết đã hết dữ liệu đầu ra

2.3.4.2. Đặc tả mô-đun NI

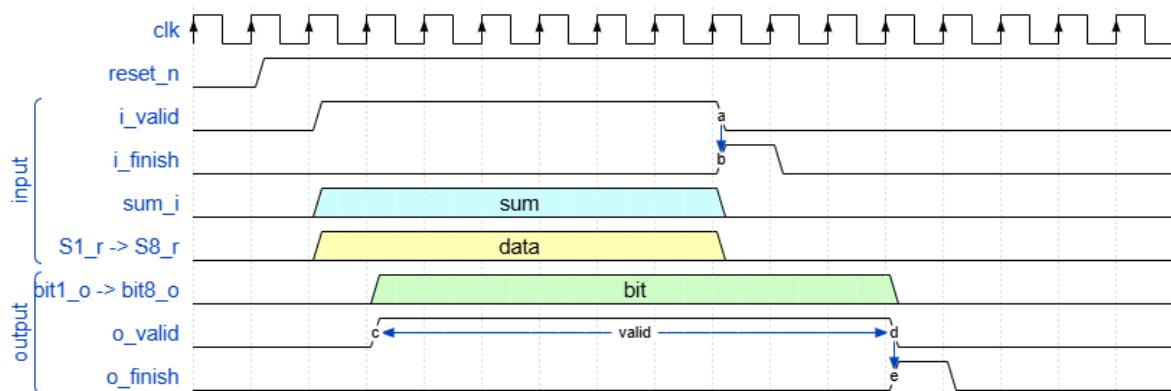
mô-đun NI sẽ tính toán giá trị của 8 điểm trong đường tròn bán kính r, theo mô tả tại bảng 2.21 là từ S1_r -> S8_r với giá trị sum_i. Theo công thức 1.6 thì cần tính trung bình tổng của một cửa sổ, tuy nhiên thực hiện bộ chia sẽ tốn nhiều tài nguyên và phức tạp hơn bộ nhân, do đó mô-đun sẽ triển khai nhân các giá trị đầu vào với tham số **GAIN**.

Bảng 2.20. Tham số của mô-đun NI

Tham số	Giá trị mặc định	Mô tả
WIDTH	10	Độ rộng bit của dữ liệu đầu vào sum
GAIN	25	Hệ số nhân đối với sum

Bảng 2.21. Danh sách các tín hiệu của mô-đun NI

Tên tín hiệu	Độ rộng	Vào/Ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Tín hiệu reset đồng bộ, mức kích hoạt thấp
i_valid	1	Vào	Tín hiệu báo dữ liệu đầu vào là hợp lệ
i_finish	1	Vào	Tín hiệu báo đã hết dữ liệu đầu vào
S1_r -> S8_r	24	Vào	Các giá trị đầu vào từ 8 hướng tại bán kính r
sum_i	WIDTH	Vào	Tổng giá trị từ mô-đun trước
o_valid	1	Ra	Tín hiệu báo dữ liệu đầu ra là hợp lệ
o_finish	1	Ra	Tín hiệu báo đã hết dữ liệu đầu ra
bit1_o -> bit8_o	1	Ra	8 giá trị đầu ra nhị phân ứng với 8 hướng



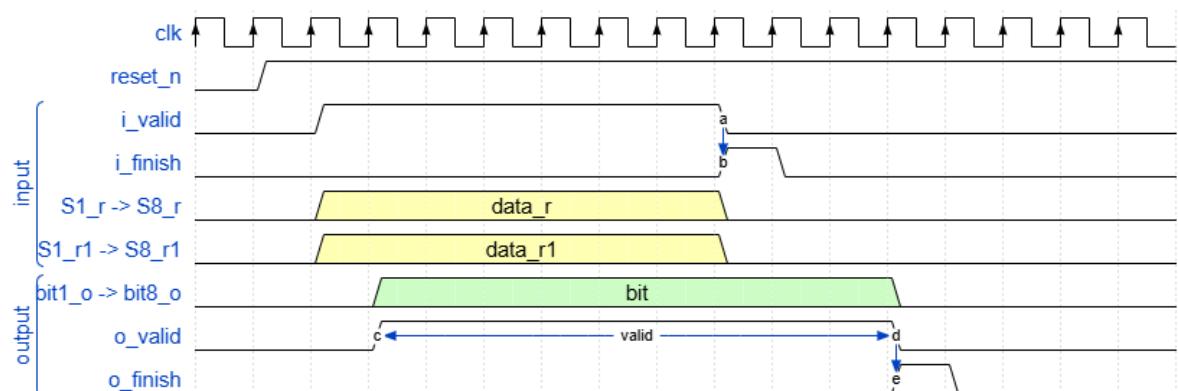
Hình 2.17. Dạng sóng của mô-đun NI

2.3.4.3. Đặc tả mô-đun RD

mô-đun RD sẽ tính giá trị mô tả bằng cách so sánh lần lượt các giá trị nằm trên 2 đường tròn bán kính r và r1. Để thuận tiện cho việc đồng bộ dữ liệu, đầu vào sẽ cần có đủ 8 dữ liệu bán kính r và 8 dữ liệu bán kính r1. Dữ liệu đầu ra sẽ lần lượt là 8 bit lần lượt với từng vị trí trên đường tròn. Ví dụ tại vị trí 0 độ, sẽ so sánh giá trị S1_r và S1_r1, tương tự với các vị trí khác.

Bảng 2.22. Danh sách các tín hiệu của mô-đun RD

Tên tín hiệu	Độ rộng	Vào/Ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Tín hiệu reset đồng bộ, mức kích hoạt thấp
i_valid	1	Vào	Tín hiệu báo dữ liệu đầu vào là hợp lệ
i_finish	1	Vào	Tín hiệu báo đã hết dữ liệu đầu vào
S1_r → S8_r	24	Vào	Các giá trị đầu vào từ 8 hướng tại bán kính r
S1_r1 → S8_r1	24	Vào	Các giá trị đầu vào từ 8 hướng tại bán kính r1
o_valid	1	Ra	Tín hiệu báo dữ liệu đầu ra là hợp lệ
o_finish	1	Ra	Tín hiệu báo đã hết dữ liệu đầu ra
bit1_o → bit8_o	1	Ra	8 giá trị đầu ra nhị phân ứng với 8 hướng



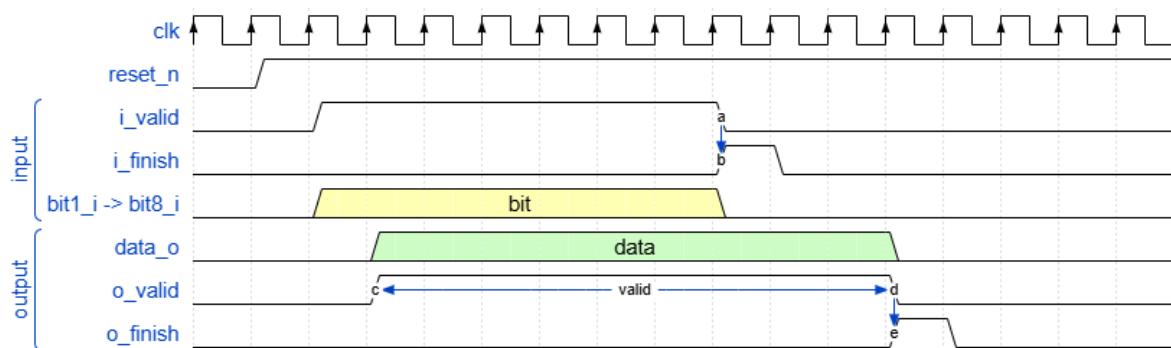
Hình 2.18. Dạng sóng của mô-đun RD

2.3.4.4. Đặc tả mô-đun RIU2

mô-đun RIU2 sẽ tính toán ra các đặc trưng dựa vào 8-bit đầu vào được tính hoặc từ mô-đun NI hoặc từ mô-đun RD theo công thức 1.3.

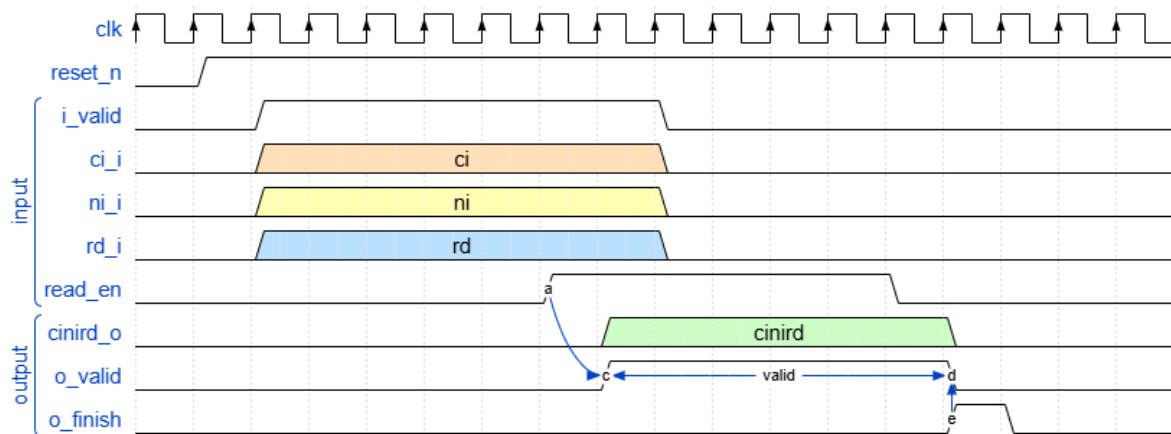
Bảng 2.23. Danh sách các tín hiệu của mô-đun RIU2

Tên tín hiệu	Độ rộng	Vào/Ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Tín hiệu reset đồng bộ, mức kích hoạt thấp
i_valid	1	Vào	Tín hiệu báo dữ liệu đầu vào là hợp lệ
i_finish	1	Vào	Tín hiệu báo đã hết dữ liệu đầu vào
bit1_i -> bit8_i	1	Vào	8 giá trị nhị phân đầu vào
o_valid	1	Ra	Tín hiệu báo dữ liệu đầu ra là hợp lệ
o_finish	1	Ra	Tín hiệu báo đã hết dữ liệu đầu ra
data_o	4	Ra	Giá trị riu2



Hình 2.19. Dạng sóng của mô-đun RIU2

2.3.5. Đặc tả mô-đun JointHistogram



Hình 2.20. Biểu đồ sóng của mô-đun JointHistogram

mô-đun JointHistogram sẽ có mục đích ghép 3 loại dữ liệu CI, NI và RD theo một công thức toán học nào đó để đạt được đặc trưng đầu ra. Công thức 2.2 mô tả cách một đặc trưng được tính toán, mỗi lần đặc trưng này xuất hiện thì giá trị tại bộ nhớ nơi lưu trữ giá trị đây được tăng lên 1 đơn vị. Vì ci chỉ nhận được giá trị từ 0 -> 1, các giá trị ni và rd sẽ chỉ nhận giá trị từ 0 -> 9 nên thực tế đặc trưng đầu ra có chiều dữ liệu là 200 ứng với mỗi r.

$$f = ci * 100 + ni * 10 + rd \quad (2.2)$$

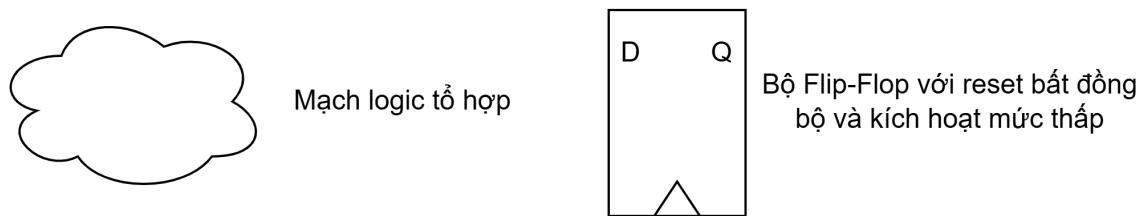
Bảng 2.24. Danh sách các tín hiệu của mô-đun JointHistogram

Tên tín hiệu	Độ rộng	Vào/Ra	Mô tả
clk	1	Vào	Tín hiệu clock
rst_n	1	Vào	Tín hiệu reset đồng bộ, mức kích hoạt thấp
i_valid	1	Vào	Tín hiệu báo dữ liệu đầu vào là hợp lệ
ci_i	1	Vào	Mô tả ci đầu vào
ni_i	4	Vào	Mô tả ni đầu vào
rd_i	4	Vào	Mô tả rd đầu vào
read_en	1	Vào	Tín hiệu cho phép đọc từ bộ nhớ
cinird_o	16	Ra	Đặc trưng đầu ra
o_valid	1	Ra	Thông báo dữ liệu đầu ra là hợp lệ
o_finish	1	Ra	Tín hiệu thông báo đã hết dữ liệu đầu ra

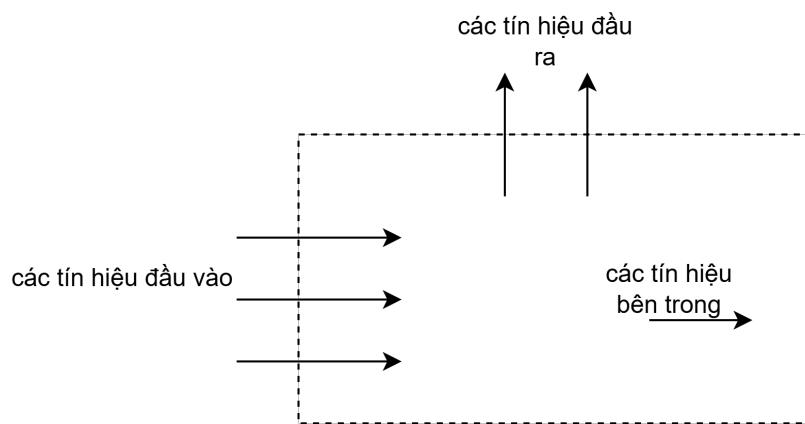
CHƯƠNG 3

THIẾT KẾ RTL

Nội dung chương này mô tả về các thiết kế của các mô-đun ở mức Register-Transfer Level (RTL). Mỗi phần mô tả một mô-đun sẽ bao gồm nội dung về kiến trúc RTL và mô tả sơ đồ chuyển trạng thái. Bên cạnh đó, sẽ giải thích nguyên nhân sử dụng kiến trúc đó, ước tính số chu kỳ thực hiện của từng mô-đun. Bộ cục của chương sẽ bắt đầu từ các thành phần mô-đun nhỏ đến các mô-đun lớn hơn.



Hình 3.1. Định nghĩa các ký hiệu được sử dụng khi mô tả kiến trúc RTL



Hình 3.2. Định nghĩa các tín hiệu ra vào một mô-đun

3.1. mô-đun MedianProcessing

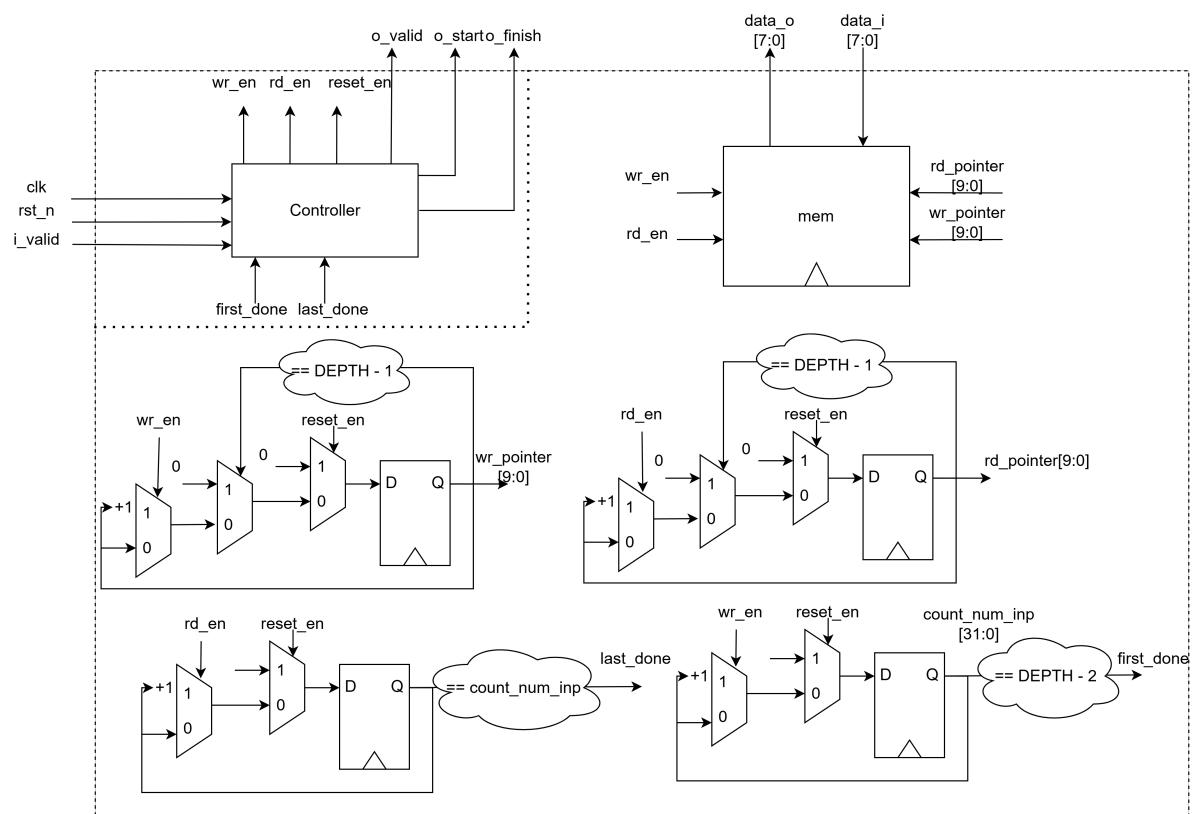
3.1.1. mô-đun LineBuffer

Nguyên lý hoạt động của mô-đun LineBuffer là khi có dữ liệu đầu vào, dữ liệu sẽ được ghi vào một vùng nhớ cụ thể, sau khi đạt đến một thời gian hoặc điều kiện đặt ra, dữ liệu từ vùng nhớ đã được ghi sẽ được đọc ra và thứ tự đầu ra sẽ theo nguyên lý FIFO

(First In First Out, dữ liệu ghi trước sẽ được đọc ra trước). Hình 3.3 mô tả kiến trúc RTL của mô-đun LineBuffer và hình 3.4 mô tả sơ đồ chuyển trạng thái của bộ *controller* cho mô-đun này. Có thể mô tả ngắn gọn cách thức hoạt động của mô-đun theo mô tả sau: Dữ liệu sẽ được đếm vào bộ nhớ, sau khi dữ liệu ở hàng đầu tiên được đếm (tức là lần đầu tiên bộ nhớ đầy) thì lúc này sẽ có tín hiệu cho phép dữ liệu ra, và quá trình này sẽ kết thúc khi toàn bộ dữ liệu đầu vào được đếm đến đầu ra.

Số lượng chu kỳ từ lúc có dữ liệu vào đến khi có dữ liệu đầu ra: DEPTH

*Số lượng chu kỳ từ lúc có dữ liệu vào đến khi có dữ liệu đầu ra của mô-đun Buffer6Rows: 3 * DEPTH*

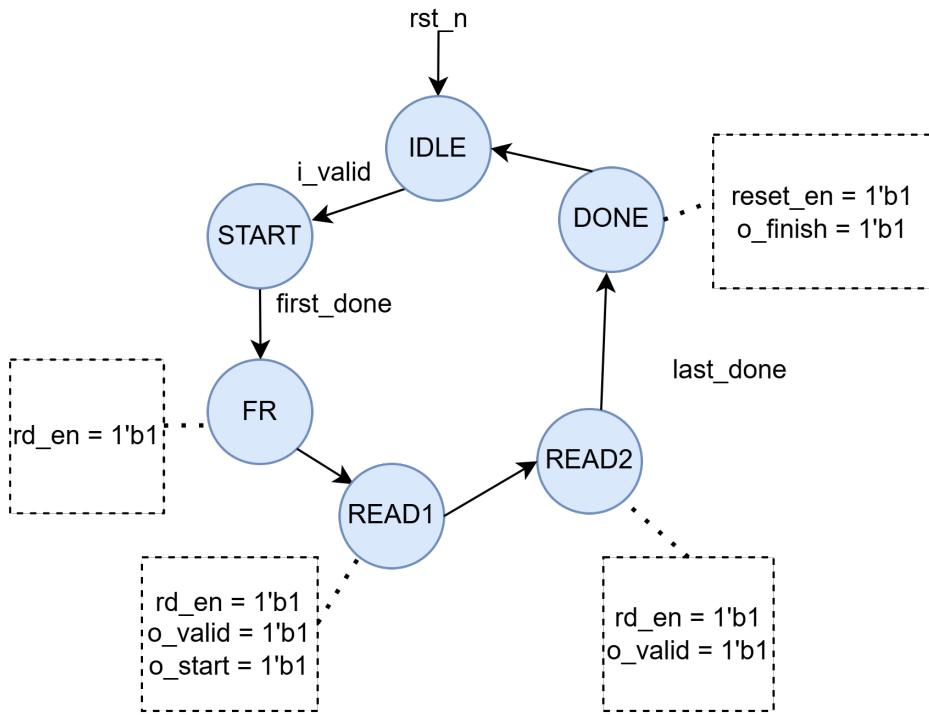


Hình 3.3. Mô tả RTL của mô-đun LineBuffer

3.1.2. mô-đun ZeroPadding

Mô-đun **ZeroPadding** sẽ được thiết kế riêng biệt cho ba loại cửa sổ là 3×3 , 5×5 và 7×7 . Tuy nhiên, chúng đều sử dụng chung một bộ điều khiển, hoạt động theo mô tả ở hình 3.5. Bộ điều khiển này bao gồm bốn trạng thái: **IDLE**, **START**, **DATA** và **DONE**.

Trạng thái **START** có chức năng chờ cho đến khi dữ liệu được đếm trong mô-đun ZeroPadding đủ để đáp ứng yêu cầu đầu ra của cửa sổ. Ví dụ, với cửa sổ 3×3 , khi xử lý ở



Hình 3.4. Sơ đồ chuyển trạng thái của mô-đun LineBuffer

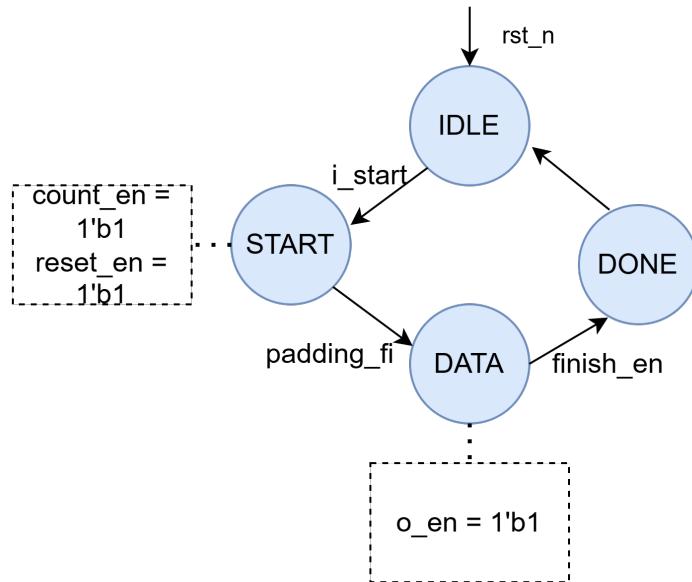
rìa ảnh sẽ cần đếm thêm các giá trị 0. Khi đó, trong mô-đun cần có sẵn ít nhất hai điểm ảnh, để sau khi đếm thêm một điểm ảnh 0 sẽ đủ ba điểm ảnh cho một hàng hoặc một cột của cửa sổ.

Trạng thái **DATA** có chức năng cho phép xuất dữ liệu đầu ra. Khi tín hiệu **o_en** ở mức logic 1, các thanh ghi chốt (flip-flop) chứa dữ liệu đầu ra mới được phép truyền dữ liệu ra ngoài. Ví dụ, trong hình 3.7, để dữ liệu đầu ra **d0_o** được phép phát ra ngoài, điều kiện là $o_en \& (i_row_lt_1 | i_col_lt_1)$. Điều kiện này cũng được áp dụng tương tự cho tất cả các phiên bản khác của mô-đun ZeroPadding.

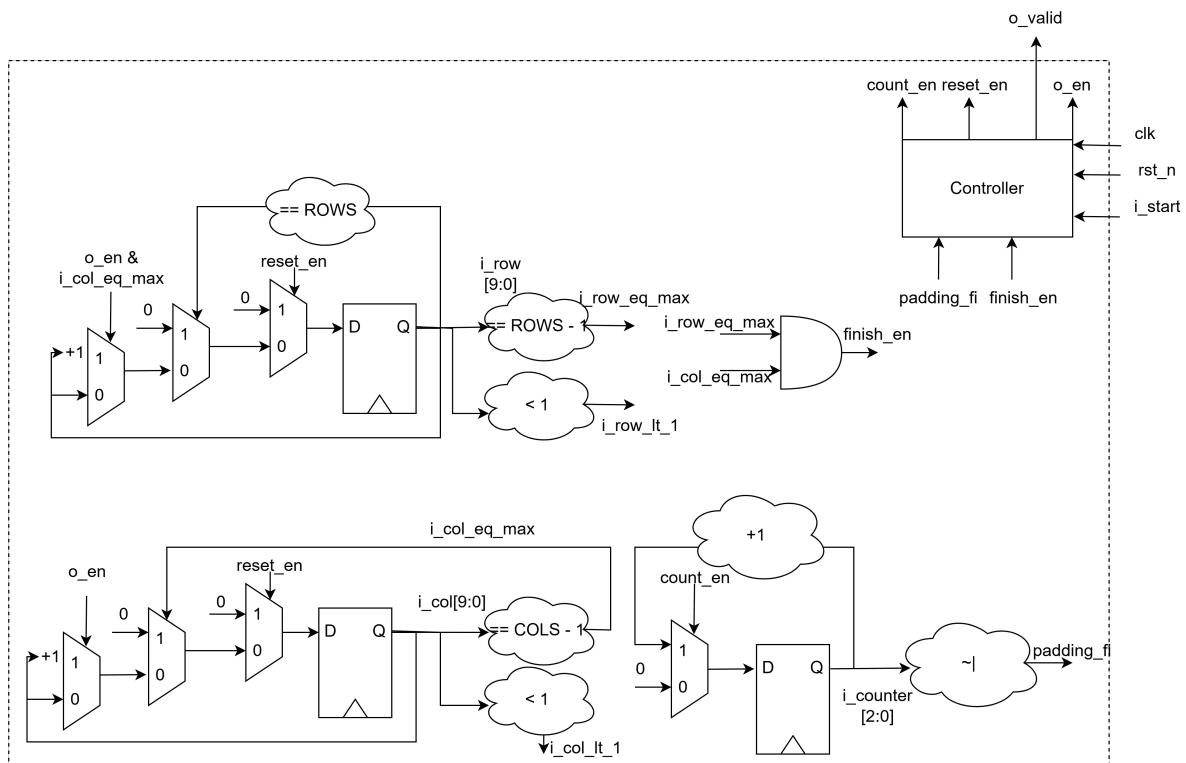
3.1.2.1. Cửa sổ 3x3

Hình 3.6 mô tả kiến trúc RTL đối với bộ ZeroPadding của cửa sổ 3x3, dữ liệu đầu vào của từng hàng sẽ được đếm qua 3 thanh ghi để tạo ra 3 giá trị ứng với mỗi hàng của cửa sổ đầu ra. Dữ liệu đầu ra sẽ dựa vào các điều kiện như ở hình 3.7, chính là các tín hiệu điều khiển bộ mạch ghép kênh (mux) để lựa chọn đầu ra là dữ liệu nào.

Bởi vì đối với các cửa sổ 5x5 và 7x7, số lượng các đầu vào và điều kiện cho đầu ra là rất nhiều, khó có thể mô tả bằng hình vẽ, do đó sinh viên sẽ đưa ra bảng giá trị tham chiếu và điều kiện cho từng giá trị đầu ra.



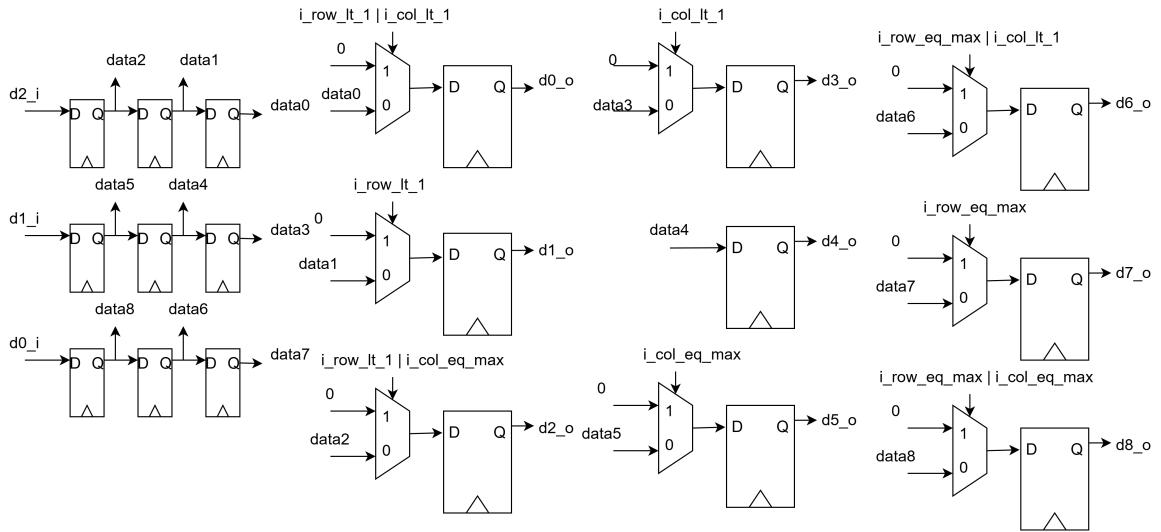
Hình 3.5. Sơ đồ chuyển trạng thái của mô-đun ZeroPadding



Hình 3.6. Mô tả RTL (1) của mô-đun ZeroPadding ứng với cửa sổ 3×3

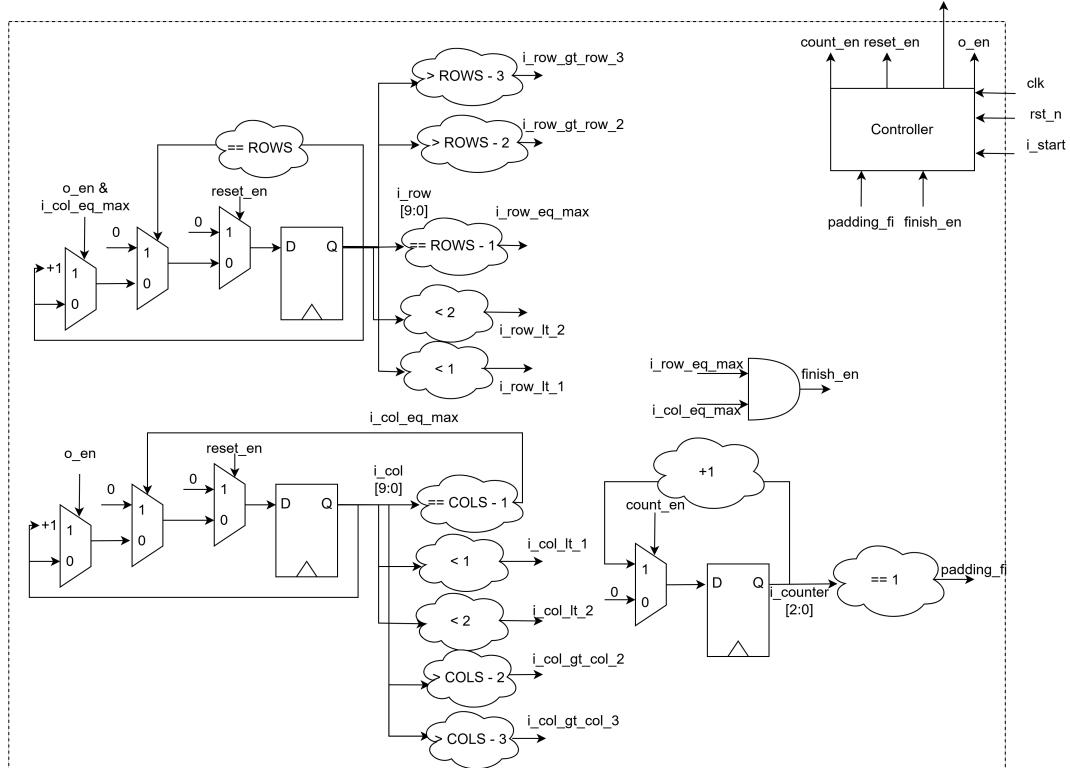
3.1.2.2. Cửa sổ 5×5

Hình 3.8 mô tả kiến trúc ở mức RTL cho mô-đun ZeroPadding ứng với cửa sổ đầu vào 5×5 . Về cơ bản, nguyên lý hoạt động của mô-đun này tương tự như đối với cửa sổ



Hình 3.7. Mô tả RTL (2) của mô-đun ZeroPadding ứng với cửa sổ 3×3

3×3 , nhưng có thêm một vài điều kiện thêm cho đầu ra như $i_row_gt_row_3$, ... Những điều kiện cho đầu ra đã được mô tả chi tiết trong bảng 3.1. Cột điều kiện tương ứng với tín hiệu chọn cho bộ mạch ghép kênh, khi điều kiện đúng thì đầu ra sẽ là 0, khi điều kiện sai, dữ liệu đầu ra sẽ ứng với giá trị của giá trị tham chiếu đến.



Hình 3.8. Mô tả RTL của mô-đun ZeroPadding ứng với cửa sổ 5×5

Bảng 3.1. Bảng điều kiện cho dữ liệu đầu ra ứng với cửa sổ 5x5

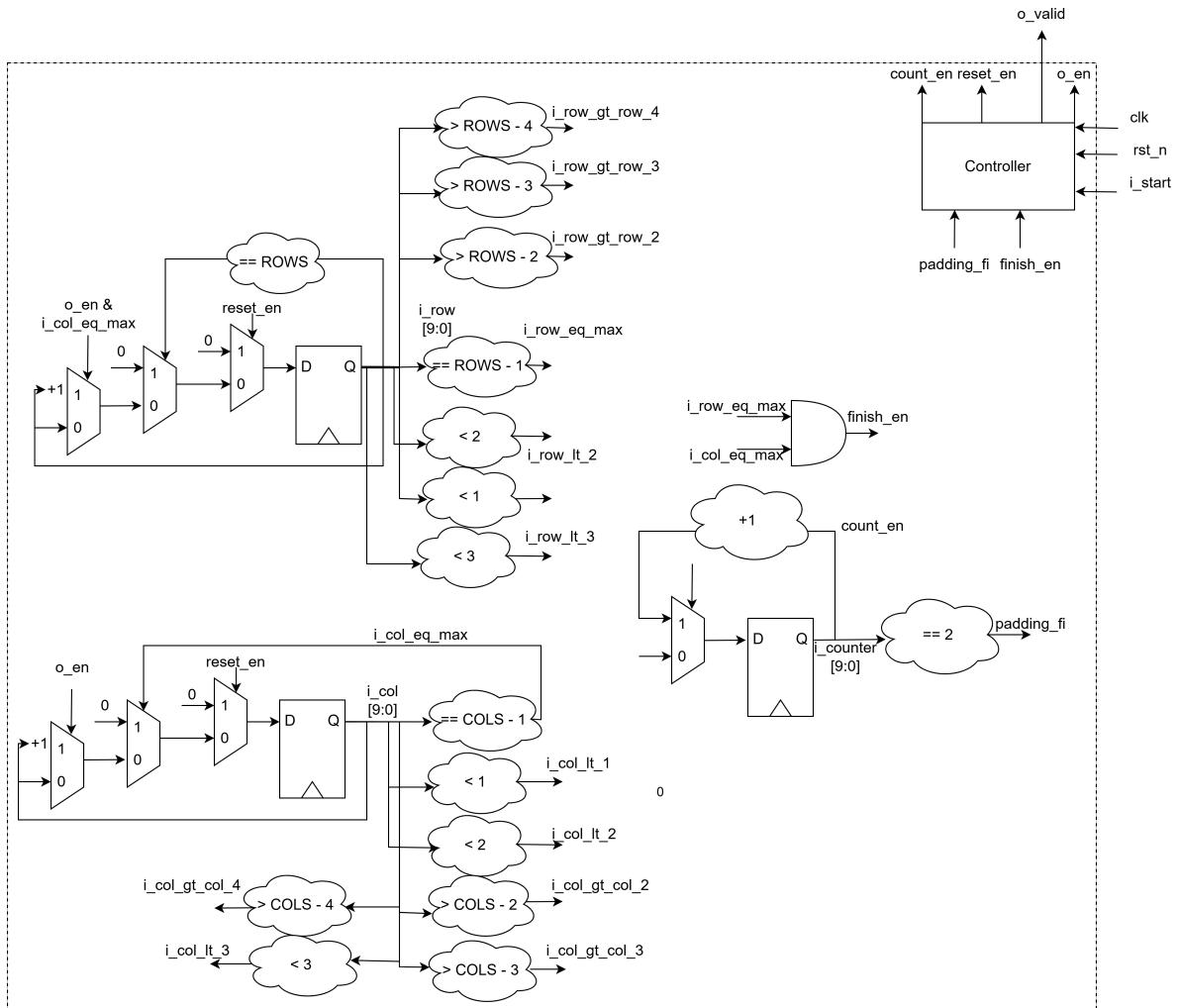
Tên đầu ra	Điều kiện	Giá trị tham chiếu
d0_o	i_row_lt_2 i_col_lt_2	data0
d1_o	i_row_lt_2 i_col_lt_1	data1
d2_o	i_row_lt_2	data2
d3_o	i_row_lt_2 i_col_gt_col_2	data3
d4_o	i_row_lt_2 i_col_gt_col_3	data4
d5_o	i_row_lt_1 i_col_lt_2	data5
d6_o	i_row_lt_1 i_col_lt_1	data6
d7_o	i_row_lt_1	data7
d8_o	i_row_lt_1 i_col_gt_col_2	data8
d9_o	i_row_lt_1 i_col_gt_col_3	data9
d10_o	i_col_lt_2	data10
d11_o	i_col_lt_1	data11
d12_o	– (không có điều kiện)	data12
d13_o	i_col_gt_col_2	data13
d14_o	i_col_gt_col_2	data14
d15_o	i_row_gt_row_2 i_col_lt_2	data15
d16_o	i_row_gt_row_2 i_col_lt_1	data16
d17_o	i_row_gt_row_2	data17
d18_o	i_row_gt_row_2 i_col_gt_col_2	data18
d19_o	i_row_gt_row_2 i_col_gt_col_3	data19
d20_o	i_row_gt_row_3 i_col_lt_2	data20
d21_o	i_row_gt_row_3 i_col_lt_1	data21
d22_o	i_row_gt_row_3	data22
d23_o	i_row_gt_row_3 i_col_gt_col_2	data23
d24_o	i_row_gt_row_3 i_col_gt_col_3	data24

3.1.2.3. Cửa sổ 7x7

Bảng 3.6 mô tả số chu kỳ cần thiết từ lúc có dữ liệu vào đến khi có dữ liệu ra của từng mô-đun ZeroPadding ứng với từng loại cửa sổ. Thời gian tính toán tính từ thời điểm mà tín hiệu *i_start* chuyển trạng thái đến khi mà tín hiệu đầu ra *o_valid* chuyển trạng thái.

Bảng 3.2. Số chu kỳ thực hiện của các mô-đun ZeroPadding

Tên mô-đun	Số chu kỳ
ZeroPadding3x3	3 chu kỳ
ZeroPadding5x5	4 chu kỳ
ZeroPadding7x7	5 chu kỳ



Hình 3.9. Mô tả RTL của mô-đun ZeroPadding ứng với cửa sổ 7x7

Bảng 3.3. Bảng điều kiện cho dữ liệu đầu ra ứng với cửa sổ 7x7

Tên đầu ra	Điều kiện	Giá trị tham chiếu
d0_o	i_row_lt_3 i_col_lt_3	data0
d1_o	i_row_lt_3 i_col_lt_2	data1

Tên đầu ra	Điều kiện	Giá trị tham chiếu
d2_o	i_row_lt_3 i_col_lt_1	data2
d3_o	i_row_lt_3	data3
d4_o	i_row_lt_3 i_col_gt_col_2	data4
d5_o	i_row_lt_3 i_col_gt_col_3	data5
d6_o	i_row_lt_3 i_col_gt_col_4	data6
d7_o	i_row_lt_2 i_col_lt_3	data7
d8_o	i_row_lt_2 i_col_lt_2	data8
d9_o	i_row_lt_2 i_col_lt_1	data9
d10_o	i_row_lt_2	data10
d11_o	i_row_lt_2 i_col_gt_col_2	data11
d12_o	i_row_lt_2 i_col_gt_col_3	data12
d13_o	i_row_lt_2 i_col_gt_col_4	data13
d14_o	i_row_lt_1 i_col_lt_3	data14
d15_o	i_row_lt_1 i_col_lt_2	data15
d16_o	i_row_lt_1 i_col_lt_1	data16
d17_o	i_row_lt_1	data17
d18_o	i_row_lt_1 i_col_gt_col_2	data18
d19_o	i_row_lt_1 i_col_gt_col_3	data19
d20_o	i_row_lt_1 i_col_gt_col_4	data20
d21_o	i_col_lt_3	data21
d22_o	i_col_lt_2	data22
d23_o	i_col_lt_1	data23
d24_o	– (không có điều kiện)	data24
d25_o	i_col_gt_col_2	data25
d26_o	i_col_gt_col_3	data26
d27_o	i_col_gt_col_4	data27
d28_o	i_row_gt_row_2 i_col_lt_3	data28
d29_o	i_row_gt_row_2 i_col_lt_2	data29
d30_o	i_row_gt_row_2 i_col_lt_1	data30
d31_o	i_row_gt_row_2	data31
d32_o	i_row_gt_row_2 i_col_gt_col_2	data32
d33_o	i_row_gt_row_2 i_col_gt_col_3	data33

Tên đầu ra	Điều kiện	Giá trị tham chiếu
d34_o	i_row_gt_row_2 i_col_gt_col_4	data34
d35_o	i_row_gt_row_3 i_col_lt_3	data35
d36_o	i_row_gt_row_3 i_col_lt_2	data36
d37_o	i_row_gt_row_3 i_col_lt_1	data37
d38_o	i_row_gt_row_3	data38
d39_o	i_row_gt_row_3 i_col_gt_col_2	data39
d40_o	i_row_gt_row_3 i_col_gt_col_3	data40
d41_o	i_row_gt_row_3 i_col_gt_col_4	data41
d42_o	i_row_gt_row_4 i_col_lt_3	data42
d43_o	i_row_gt_row_4 i_col_lt_2	data43
d44_o	i_row_gt_row_4 i_col_lt_1	data44
d45_o	i_row_gt_row_4	data45
d46_o	i_row_gt_row_4 i_col_gt_col_2	data46
d47_o	i_row_gt_row_4 i_col_gt_col_3	data47
d48_o	i_row_gt_row_4 i_col_gt_col_4	data48

3.1.3. mô-đun MedianCalculation

mô-đun MedianCalculation sẽ tính toán giá trị trung vị đầu ra dựa trên các điểm ảnh đầu vào. Vì sẽ tính toán cho riêng biệt 3 cửa sổ là 3x3, 5x5 và 7x7, nên để tối ưu, sinh viên sẽ thiết kế riêng 3 bộ tính trung vị ứng với từng cửa sổ.

3.1.3.1. Thuật toán tính toán trung vị

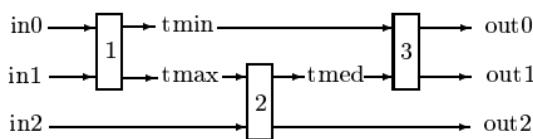
Đầu tiên, một mạng sắp xếp(sorting network) được định nghĩa là một chuỗi các hoạt động so sánh và hoán đổi các phần tử. Mặc dù một mạng sắp xếp với một số lượng phân tử cố định cần yêu cầu nhiều phép so sánh hơn so với các phương pháp so sánh như sắp xếp nhanh, ... Tuy nhiên, nó có một lợi thế là không phải phụ thuộc vào kết quả của những phép so sánh trước đó, do đó không cần sự điều khiển. Từ đó, nó sẽ phù hợp với các bài toán tính toán song song.

Hình 3.10 mô tả một mạng sắp xếp cho 3 phần tử với out0, out1 và out2 lần lượt là các giá trị nhỏ nhất, trung vị và lớn nhất trong 3 phần tử đó. Các ô 1, 2 và 3 là các bộ so sánh và hoán đổi có kiến trúc được mô tả ở hình 3.11. **Chú ý:** *Ở trong các hình ảnh bên dưới, mạng sắp xếp sẽ được gọi là Sorting_network và bộ so sánh và hoán đổi được gọi*

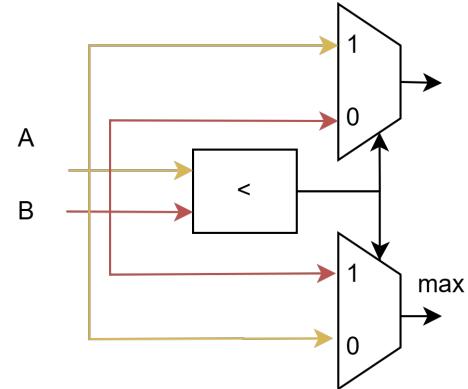
là **Node**. Thuật toán 3.1 mô tả cách tính toán giá trị trung vị ứng với một mảng có kích thước $N \times N$, bao gồm 3 bước chính là sắp xếp từng hàng, sắp xếp từng cột, và sau đó sắp xếp các đường chéo với điều kiện xác định. Thuật toán này rất phù hợp với các bộ có kích thước nhỏ, và thành phần phù hợp để triển khai là các mạng sắp xếp.

Thuật toán 3.1 Tìm trung vị của một mảng $N \times N$ với N là số lẻ [18]

- 1: **Đầu vào:** Một mảng A có kích thước $N \times N$
 - 2: **Đầu ra:** Giá trị trung vị của mảng
 - 3: $M = \frac{N-1}{2}$
 - 4: Sắp xếp các hàng của mảng theo thứ tự tăng dần
 - 5: Sắp xếp các cột của mảng theo thứ tự tăng dần
 - 6: Sắp xếp các đường chéo có độ dốc k
 - 7: **for** $k = 1$ to M **do**
 - 8: **for** $s = k \cdot (M + 1)$ to $k \cdot (M - 1) + (N - 1)$ **do**
 - 9: Dòng được sắp xếp được xác định bởi phương trình $k \cdot r + c = s$
 - 10: Với mọi r : $A[r - 1, s - k \cdot (r - 1)] \leq A[r, s - k \cdot r]$
 - 11: **end for**
 - 12: **end for**
 - 13: **Trả về:** $A[M, M]$
-



Hình 3.10. Mô tả về mạng sắp xếp với 3 phần tử [18]

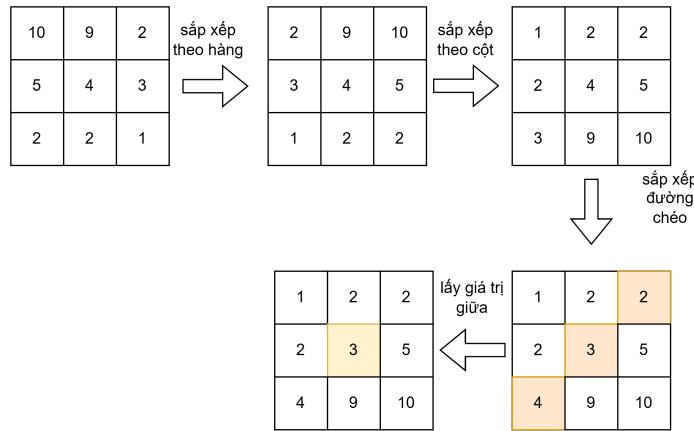


Hình 3.11. Mô tả cấu trúc bộ so sánh và hoán đổi

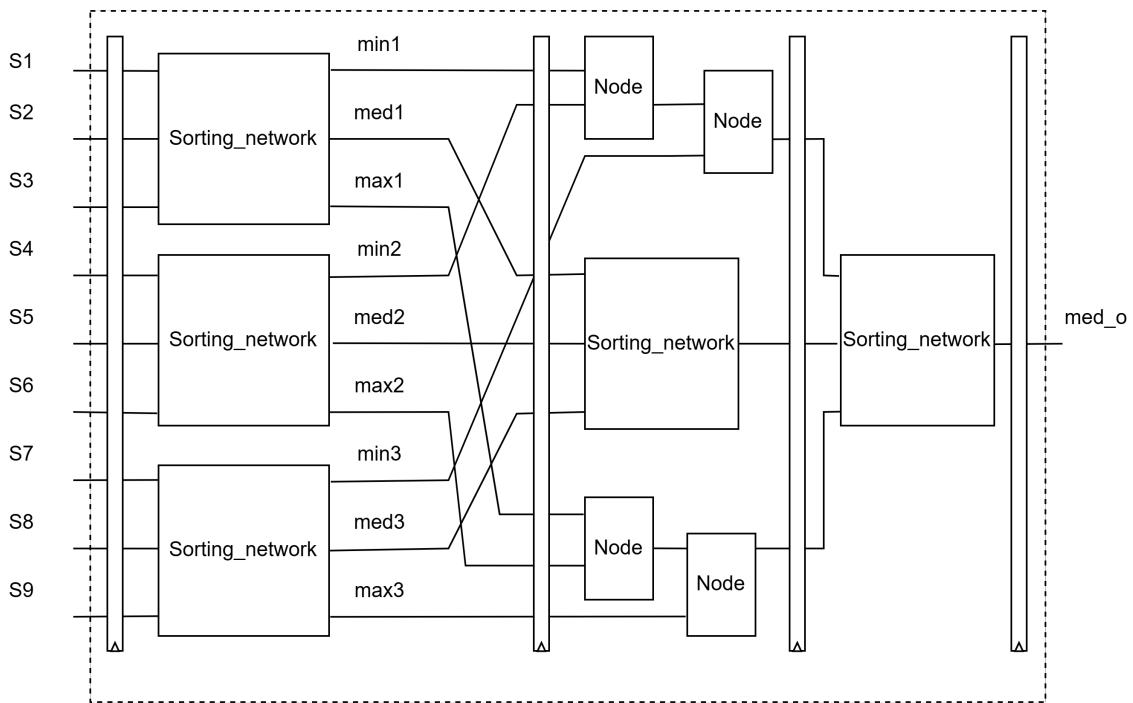
3.1.3.2. Cửa sổ 3×3

Hình 3.12 mô tả nguyên lý và ví dụ về cách tìm giá trị trung vị của một ma trận kích thước 3×3 . Với chuỗi đầu vào là 10, 9, 2, 5, 4, 3, 2, 2, 1. Giá trị trung vị ta mong đợi đạt được là 3. Các bước thực hiện bao gồm việc sắp xếp các hàng và các cột tăng dần. Sau khi đã xong hai bước trên, lúc này đường chéo gồm các phần tử 2, 4, 3. Lúc này, sẽ thực hiện sắp xếp theo thứ tự tăng dần, kết quả đạt được là 2, 3, 4. Vậy giá trị trung vị đạt được là 3, đã giống với giá trị mong đợi đạt được. Hình 3.13 mô tả kiến trúc RTL của

mô-đun MedianCalculation với cửa sổ 3×3 . Sinh viên đã thay thế các mảng sắp xếp khi đến bước sắp xếp cột để giảm tài nguyên phần cứng sử dụng (thay bộ Sorting_network bằng chỉ 2 Node, vì không cần đủ 3 giá trị cho bước tiếp theo), hai là đã chèn thêm các thanh ghi giữa các bước để giảm trễ lan truyền, giúp tăng tần số hoạt động của mạch.



Hình 3.12. Thực hiện và ví dụ của tìm trung vị của cửa sổ 3×3

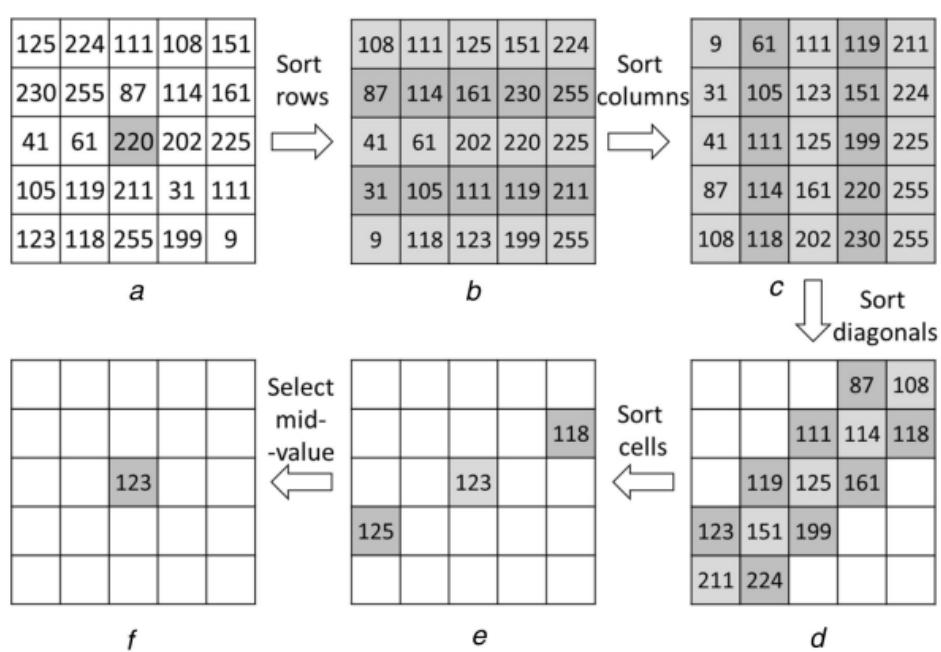


Hình 3.13. Thực hiện và ví dụ của tìm trung vị của cửa sổ 3×3

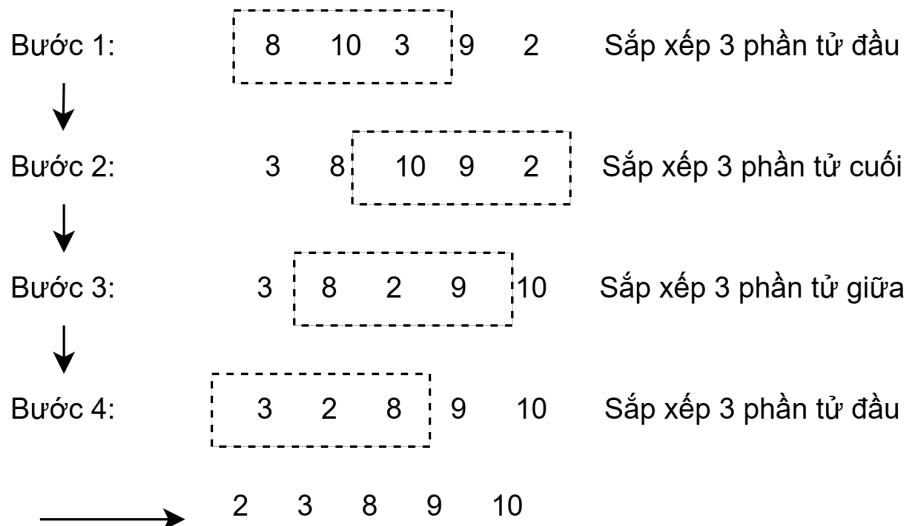
3.1.3.3. Cửa sổ 5×5

Hình 3.14 mô tả nguyên lý và ví dụ về cách tìm giá trị trung vị đối với cửa sổ 5×5 . Để tối ưu cho tốc độ sắp xếp và sự không phụ thuộc, sinh viên sẽ thiết kế một sắp xếp 5

phần tử theo thứ tự tăng dần theo các bước được mô tả trong hình 3.16.



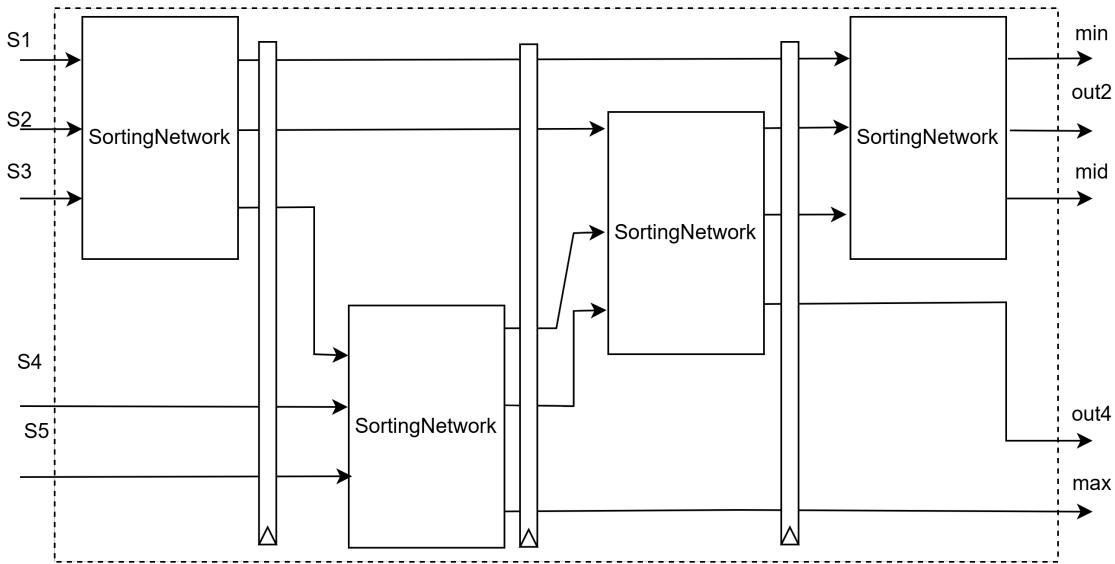
Hình 3.14. Thực hiện và ví dụ của tìm trung vị của cửa sổ 5x5 [19]



Hình 3.15. Xây dựng bộ sắp xếp 5 phần tử dựa trên bộ sắp xếp 3

3.1.3.4. Cửa sổ 7x7

Hình 3.17 mô tả cách tìm ra giá trị trung vị đối với một ma trận kích thước 7x7. Điểm khác biệt đối với cửa sổ này là sau khi đã sắp xếp hàng và sắp xếp cột xong, ta sẽ tìm ra 25 phần tử thỏa mãn điều kiện, sau đó dựa vào bộ tìm trung vị đối với ma trận



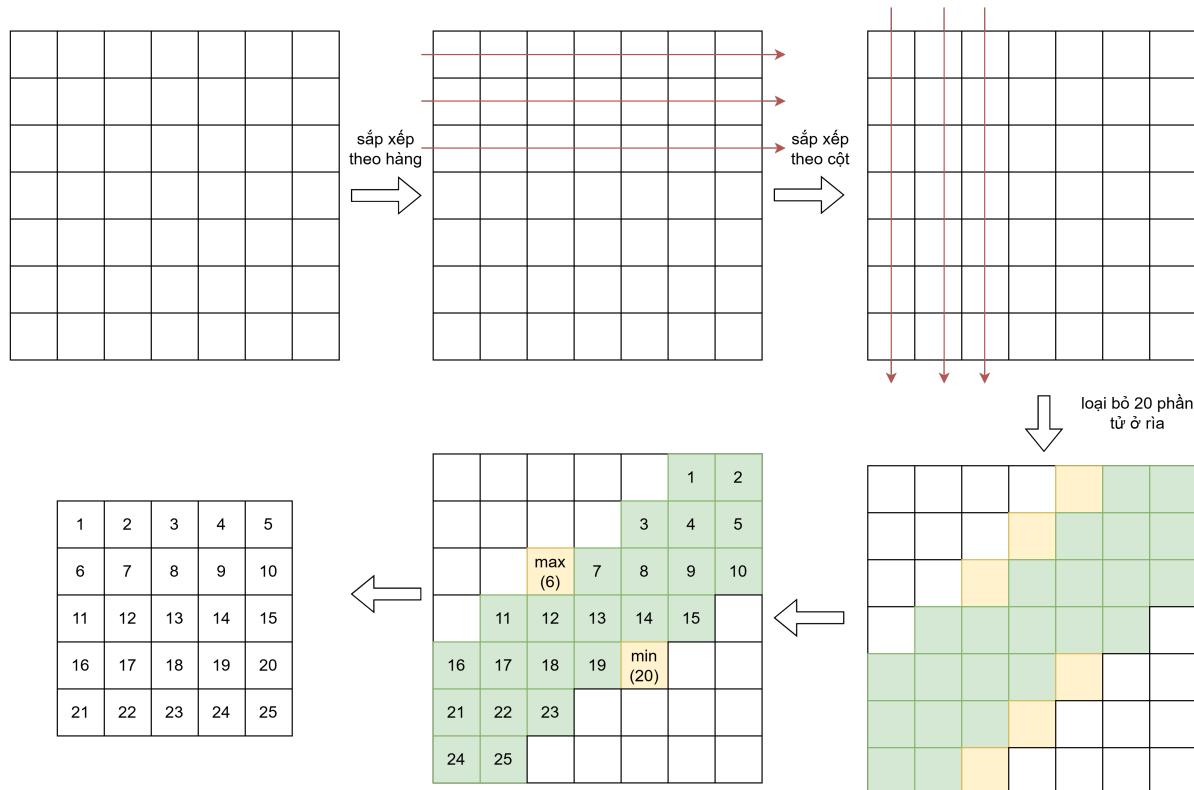
Hình 3.16. Kiến trúc của bộ sắp xếp tăng dần 5 phần tử

5x5 để tìm trung vị chứ không thực hiện trực tiếp việc sắp xếp đường chéo theo mô tả tại thuật toán 3.1. Sau khi sắp xếp theo hàng và cột xong, giá trị tại các ô xanh trong hình 3.17 đã thỏa mãn điều kiện, tuy nhiên ở mỗi góc sẽ tồn tại 3 giá trị như mô tả tại ô màu vàng. Đối với phía trên, ta cần tìm giá trị lớn nhất của nó và đối với phía dưới, ta cần tìm giá trị nhỏ nhất. Từ đó, ta sẽ có đủ 25 phần tử và sử dụng nó có thể tìm kiếm được giá trị trung vị.

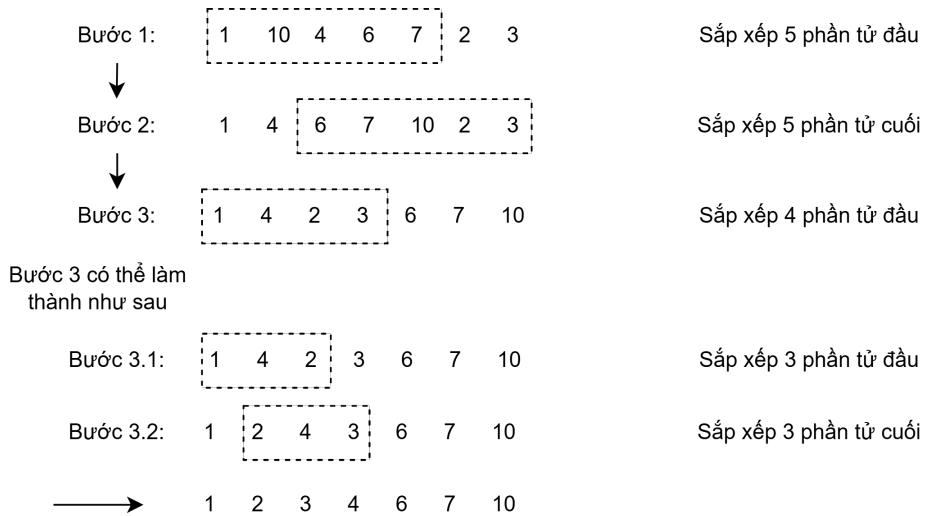
Để thực hiện sắp xếp tăng dần cho một cửa sổ 7x7, ta cần bộ sắp xếp tăng dần cho 7 phần tử, nó có thể được xây dựng trên các bộ sắp xếp 5 và bộ sắp xếp 3.

Bảng 3.4. Số chu kỳ thực hiện của các mô-đun mô-đun sắp xếp và mô-đun MedianCalculation

Tên mô-đun	Số chu kỳ
Bộ sắp xếp 3 phần tử	1 chu kỳ
MedianCalculation3x3	4 chu kỳ
Bộ sắp xếp 5 phần tử	3 chu kỳ
MedianCalculation5x5	14 chu kỳ
Bộ sắp xếp 7 phần tử	9 chu kỳ
MedianCalculation7x7	35 chu kỳ



Hình 3.17. Thực hiện và ví dụ của tìm trung vị của cửa sổ 7×7



Hình 3.18. Xây dựng bộ sắp xếp 7 phần tử dựa trên bộ sắp xếp 5 và bộ sắp xếp 3

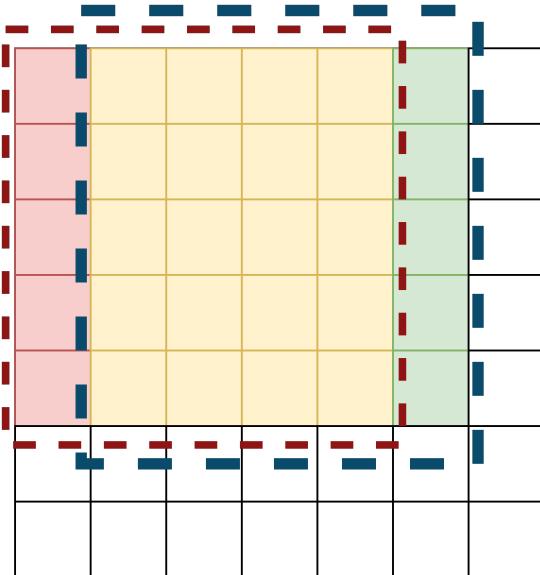
3.2. mô-đun CI

mô-đun CI thực chất là tập hợp của các 3 mô-đun MRELBP_CI, trong mỗi mô-đun đó là khối PatchSum nối với một mạch logic tổ hợp để nhằm mục đích so sánh và đưa ra giá trị so sánh đối với các phần tử trung tâm của một cửa sổ. Về cơ bản 3 mô-đun

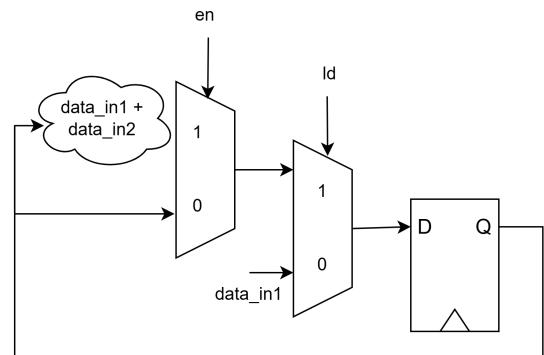
PatchSum đều xây dựng trên nguyên tắc chung sẽ được trình bày tại phần ngay sau.

3.2.1. Cơ sở xây dựng mô-đun PatchSum

Một cách đơn giản nhất để tính tổng của các cửa sổ là cộng tất cả các giá trị trong cửa sổ lại với nhau, nếu với cửa sổ kích thước 5×5 , ta sẽ cần cộng khoảng 25 giá trị trong 1 chu kỳ (nếu không sử dụng phương pháp đường ống), hoặc thực hiện một bộ cộng tuần tự, tuy nhiên nó sẽ giảm khả năng xử lý thời gian thực. Để giải quyết vấn đề này, sinh viên sẽ thực hiện một kỹ thuật triển khai giống với cửa sổ trượt. Hình 3.19 mô tả một ví dụ về kỹ thuật cửa sổ trượt. Cửa sổ mà có viền màu đỏ là cửa sổ ban đầu. Cửa sổ mà có viền màu xanh là cửa sổ ngay sau cửa sổ màu đỏ. Ta thấy, tổng của 2 cửa sổ này đều có một đặc điểm là có các giá trị ở các ô màu vàng. Giá trị tổng của cửa sổ sau sẽ bằng giá trị của cửa sổ trước đó trừ đi giá trị ở các ô màu đỏ và cộng với các giá trị ở các ô màu xanh. Bằng nguyên lý đó, ta có thể giảm bớt số lượng phép cộng cần thực hiện trong 1 chu kỳ.



Hình 3.19. Ví dụ về nguyên lý cửa kỹ thuật cửa sổ trượt

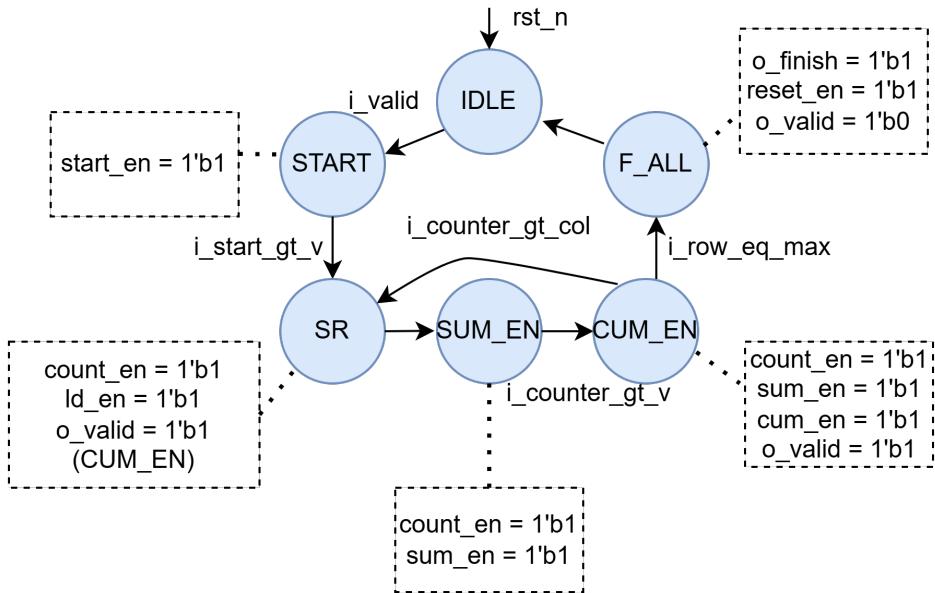


Hình 3.20. Kiến trúc của mô-đun sum_cum

3.2.2. Xây dựng mô-đun PatchSum

3.2.2.1. Sơ đồ chuyển trạng thái

Vì sẽ xây dựng 3 mô-đun PatchSum ứng với 3 giá trị bán kính r khác nhau, tuy nhiên cả 3 mô-đun này đều có cùng 1 nguyên lý hoạt động, do đó sinh viên sẽ xây dựng chung 1 bộ sơ đồ chuyển trạng thái, được mô tả tại hình 3.21.



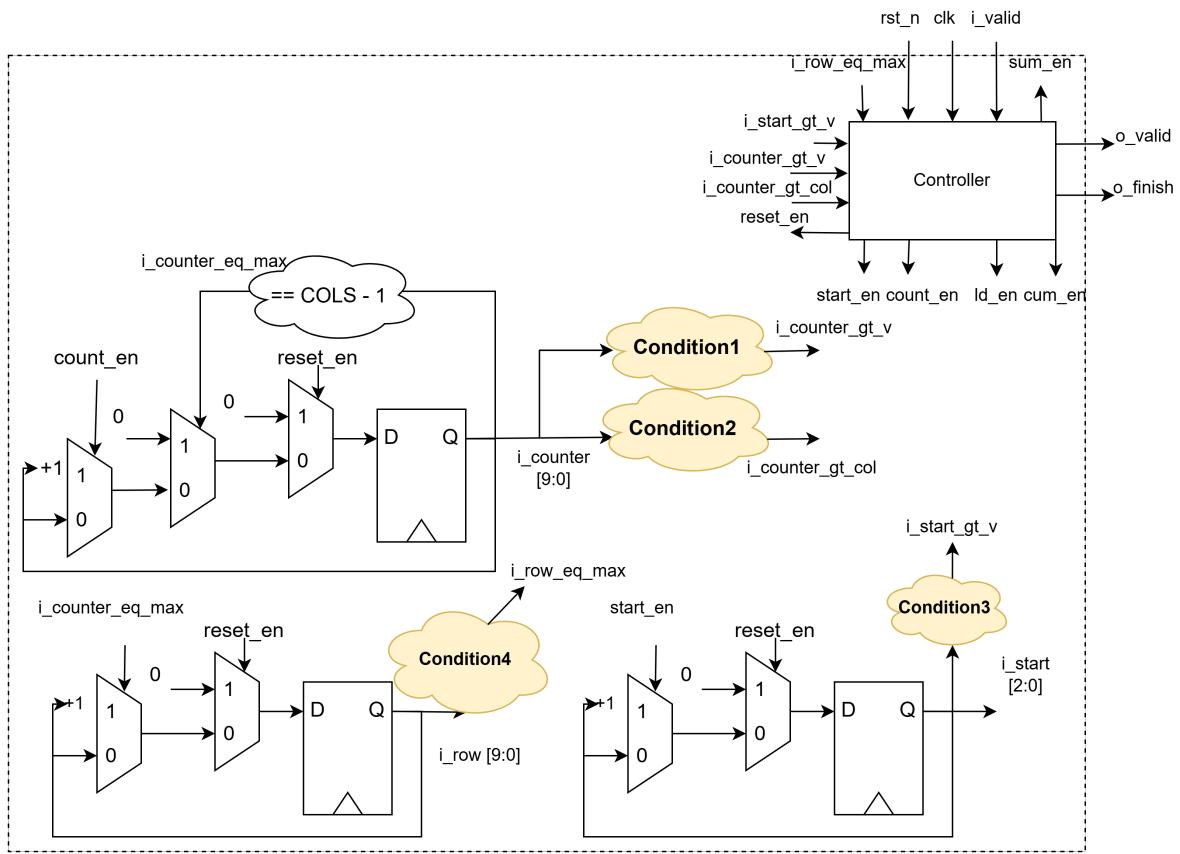
Hình 3.21. Sơ đồ chuyển trạng thái của mô-đun PatchSum

3.2.2.2. Kiến trúc RTL

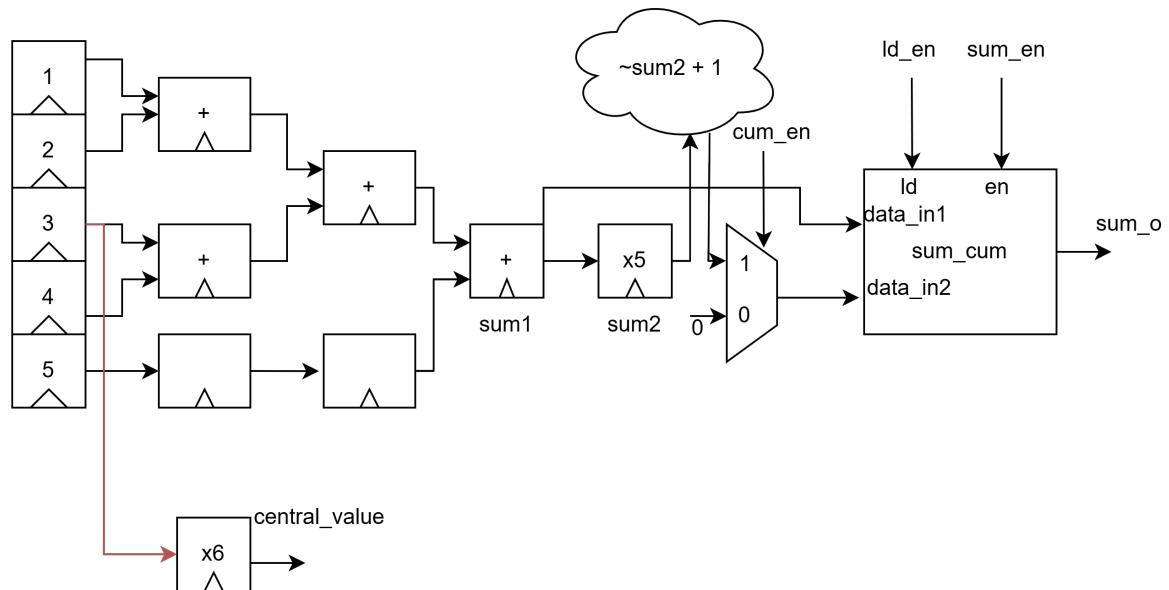
Hình 3.22 mô tả kiến trúc RTL chung của cả 3 bộ PatchSum ứng với các bán kính $r = 2, 4, 6$. Đây là các bộ giúp kiểm tra các điều kiện để máy trạng thái chuyển trạng thái, cả 3 mô-đun ứng với 3 loại r khác nhau đều sử dụng các bộ này, tuy nhiên thì mỗi bán kính sẽ ứng với các điều kiện khác nhau và các điều kiện Condition sẽ được mô tả tại bảng 3.5. Một mô PatchSum ứng với một giá trị r cụ thể sẽ có mô tả RTL ở cả 2 hình 3.22 và 3.23. Tuy nhiên hình 3.23 là mô tả RTL đầu ra cho mô-đun PatchSum với $r = 2$, với $r = 4$ và $r = 6$, kiến trúc sẽ có sự khác biệt về số lượng thanh ghi, số chu kỳ thực hiện, nhưng về nguyên lý hoạt động là tương tự nhau. Hình 3.20 mô tả cấu trúc của mô-đun sum_cum được sử dụng ở cuối kiến trúc RTL (2) của mô-đun PatchSum.

Bảng 3.5. Bảng điều kiện cho mô tả hình 3.22

Loại điều kiện \ Bán kính	$r = 2$	$r = 4$	$r = 6$
Condition1	> 3	> 7	> 11
Condition2	COLS - 2	COLS - 2	COLS - 2
Condition3	> 1	> 2	> 2
Condition4	ROWS - 4	ROWS - 8	ROWS - 12



Hình 3.22. Kiến trúc RTL (1) của mô-đun PatchSum



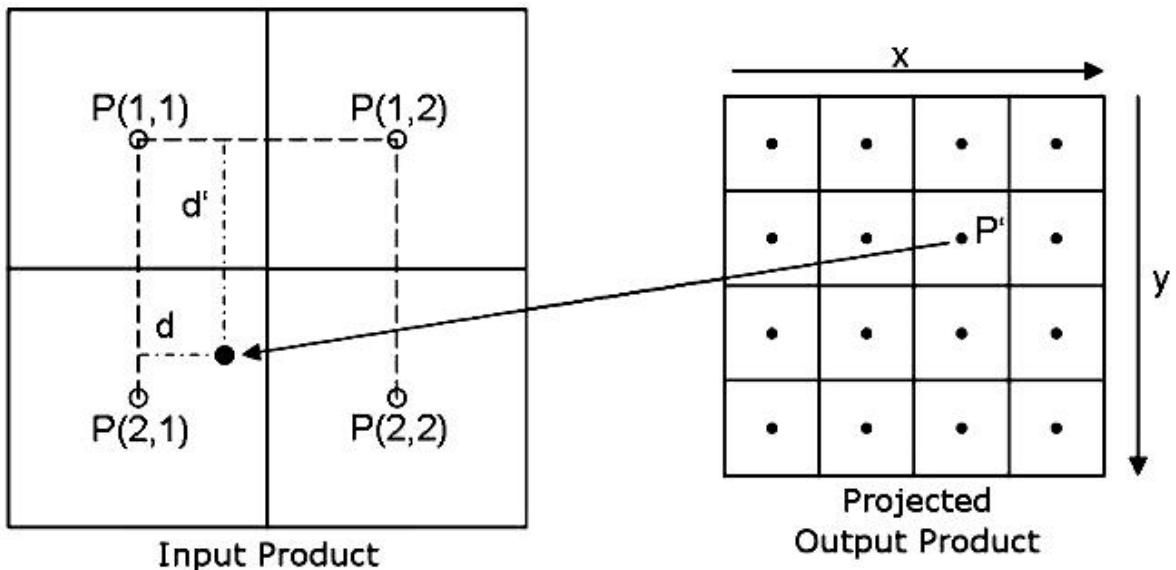
Hình 3.23. Kiến trúc RTL (2) của mô-đun PatchSum với $r = 2$

3.3. mô-đun NIRD

mô-đun NIRD có sơ đồ khối được mô tả theo hình 2.15 bao gồm khá nhiều mô-đun con, tuy nhiên các thành phần đó đã được mô tả một phần ở các nội dung trước, do đó trong phần này, sinh viên sẽ trình bày về kiến trúc RTL của các mô-đun con bao gồm mô-đun **Interpolation**, mô-đun **RIU2**.

3.3.1. mô-đun Interpolation

Để nội suy ra các giá trị cần thiết trong ảnh, thiết kế sẽ sử dụng phương pháp nội suy song tuyến tính. Phương pháp này sẽ tính toán giá trị một điểm dựa trên 4 điểm bên ngoài.



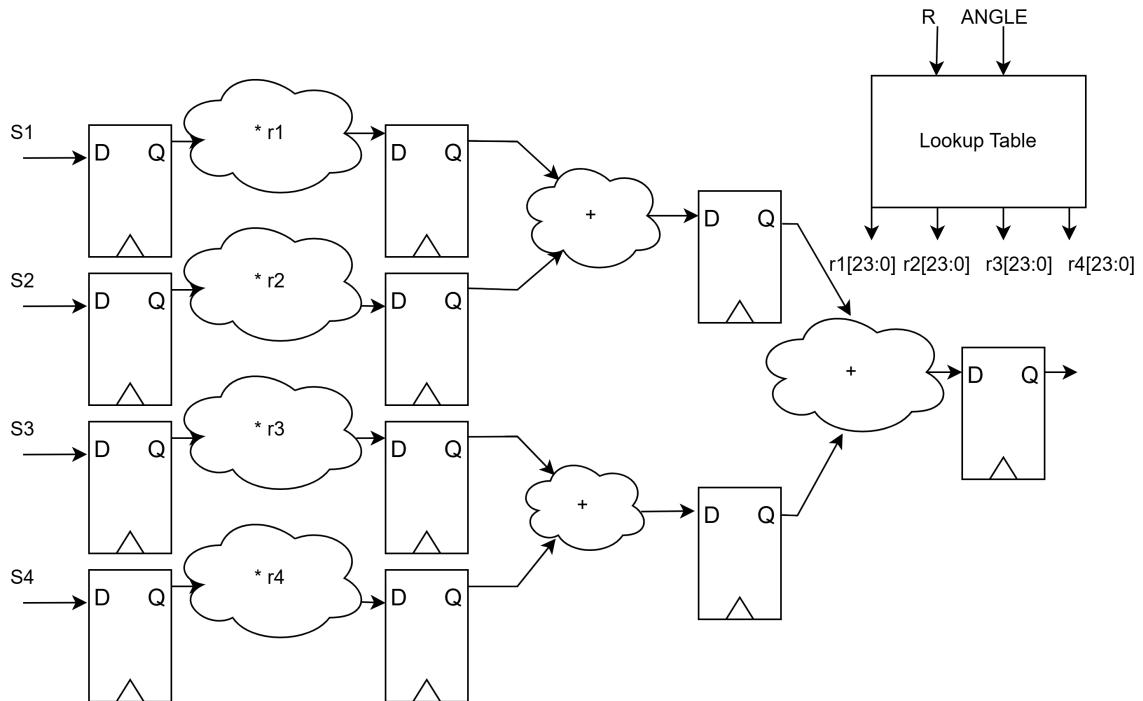
Hình 3.24. Nội suy song tuyến tính

Giá trị của điểm cần tính sẽ theo công thức:

$$f(x, y) = P(1, 1) \cdot \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} + P(2, 1) \cdot \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} \\ + P(1, 2) \cdot \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} + P(2, 2) \cdot \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} \quad (3.1)$$

mô-đun Interpolation sẽ có đầu vào là 4 điểm ảnh và đầu ra là giá trị nội suy từ 4 điểm ảnh đó. Ở đây, sinh viên sử dụng một **Lookup Table** với các giá trị được tính toán trước đó từ mã python, với mỗi bán kính và góc khác nhau, sẽ có thể lấy ra được tương ứng 4 giá trị r1, r2, r3, r4 ứng với 4 giá trị nhân với 4 đầu vào. Sinh viên sử dụng số thập

phân với dấu phẩy tinh 24 bit với 8 bit cao nhất là giá trị phần nguyên, còn lại là giá trị thập phân, như vậy giá trị nội suy lối ra tương ứng cũng sẽ là 24 bit.



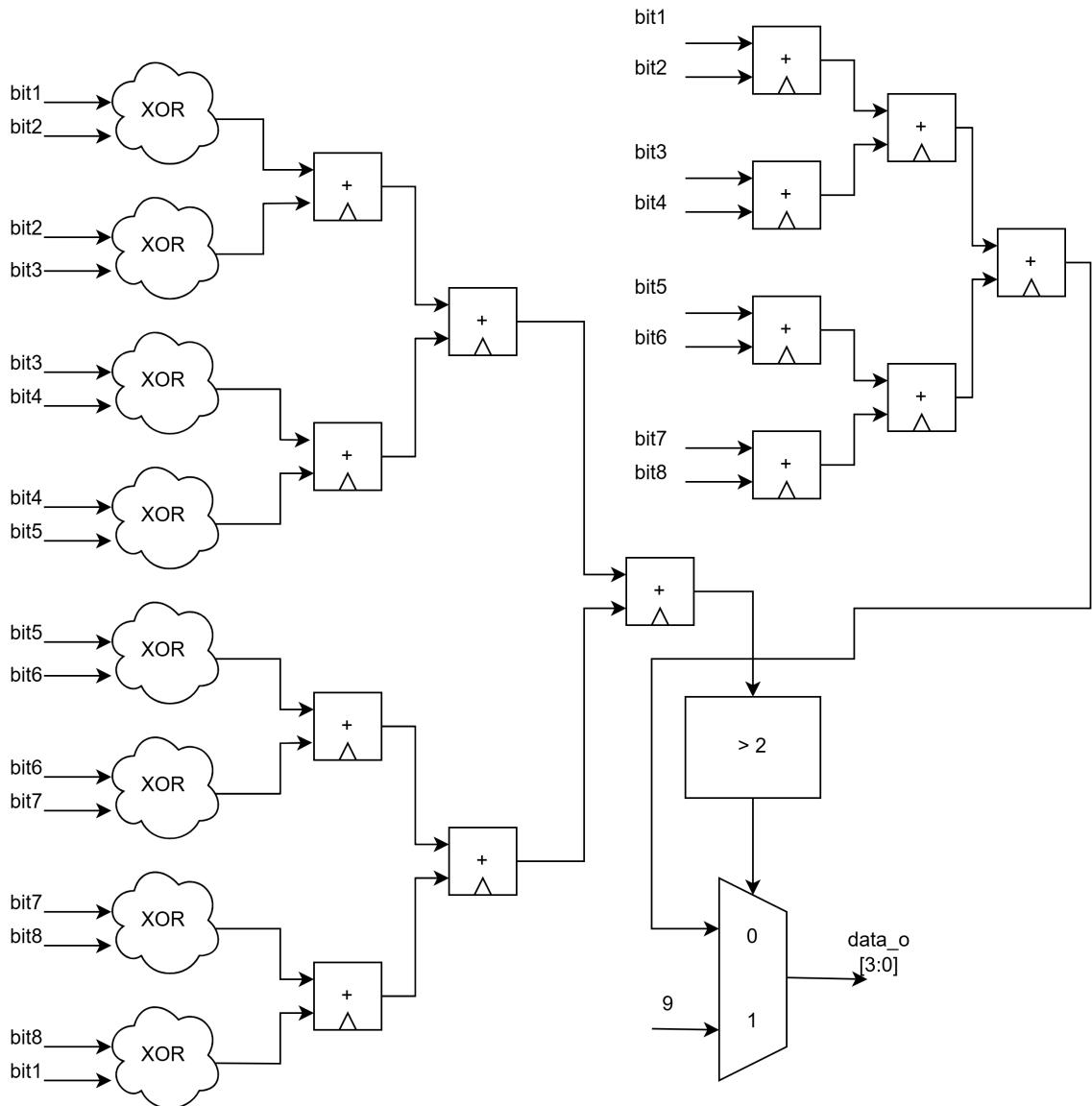
Hình 3.25. Kiến trúc RTL của mô-đun Interpolation

3.3.2. mô-đun RIU2

Mô tả với tên là RIU2 bao gồm 2 phần là RI và U2. RI là tìm giá trị nhỏ nhất khi xoay theo vòng tròn của dữ liệu trong khi đó U2 là ngưỡng của sự chuyển bit trong dữ liệu tối đa là 2. Tuy nhiên thì ta thực tế không cần quan tâm đến RI và giá trị lớn nhất hay nhỏ nhất cũng sẽ không ảnh hưởng tới số lượng chuyển bit trong dữ liệu, hay số lần chuyển bit trong dữ liệu là không đổi dù có thực hiện xoay vòng tròn. Do đó, mô-đun này sẽ tìm ra số lượng chuyển bit trong dữ liệu và quyết định đầu ra là gì. Ta biết, hai bit nếu khác nhau thì khi thực hiện phép **XOR**, giá trị đầu ra sẽ ra 1. Từ đó, ta sẽ xây dựng được kiến trúc mô tả trong hình 3.26.

3.4. mô-đun JointHistogram

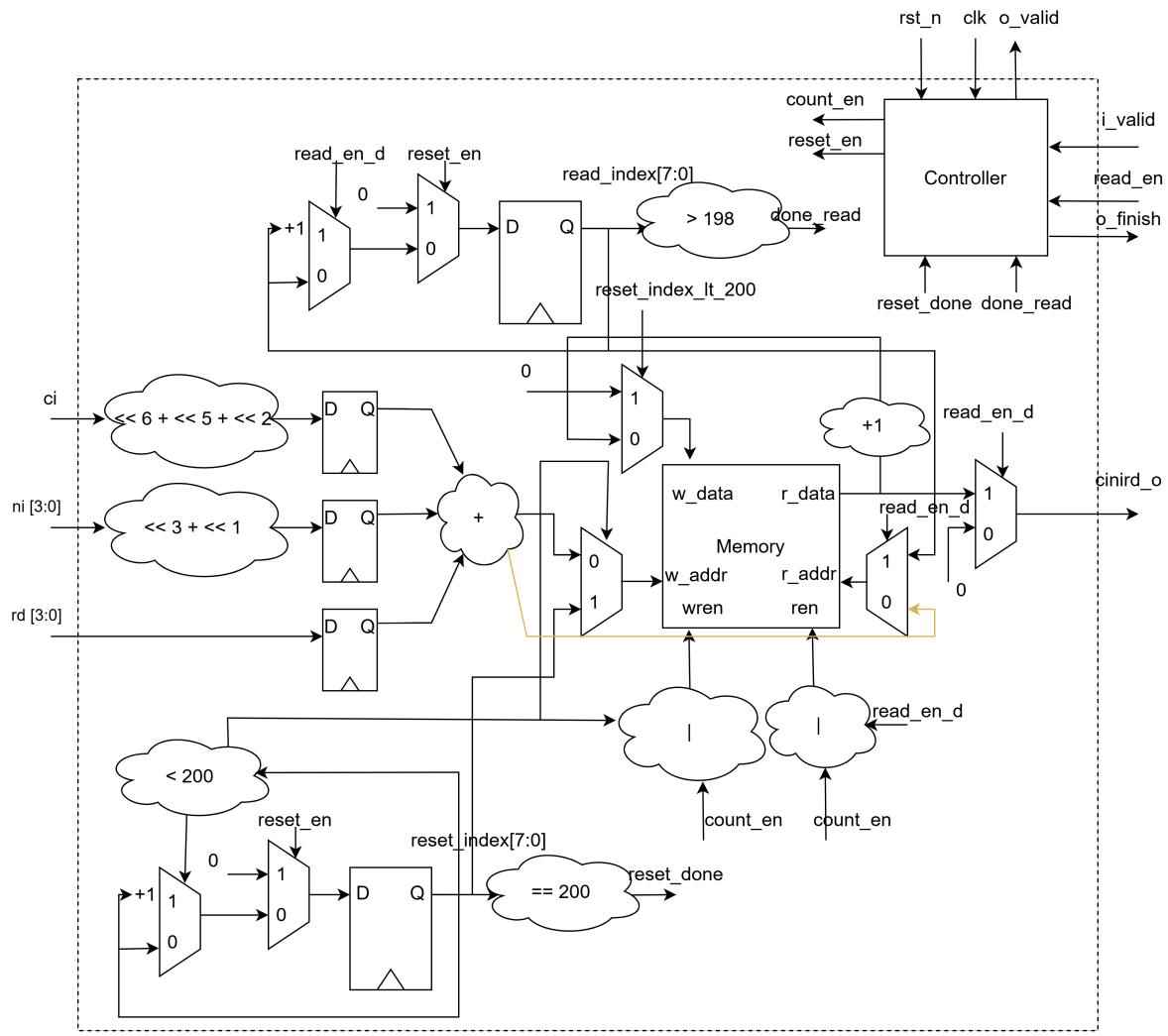
mô-đun JointHistogram được triển khai theo kiến trúc và bộ điều khiển được mô tả lần lượt lại hình 3.27, 3.28.



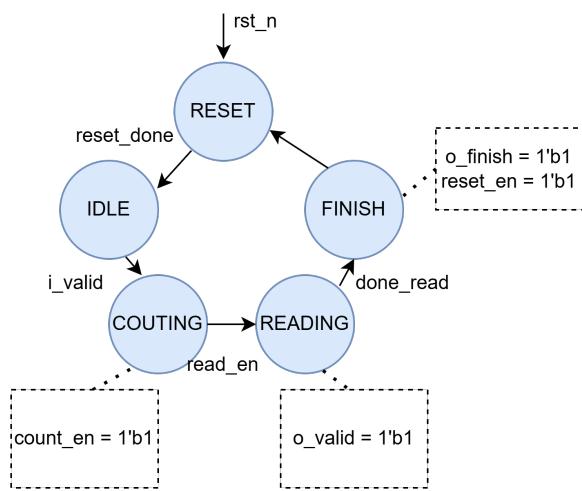
Hình 3.26. Kiến trúc RTL của mô-đun RIU2

3.5. mô-đun MRELBP

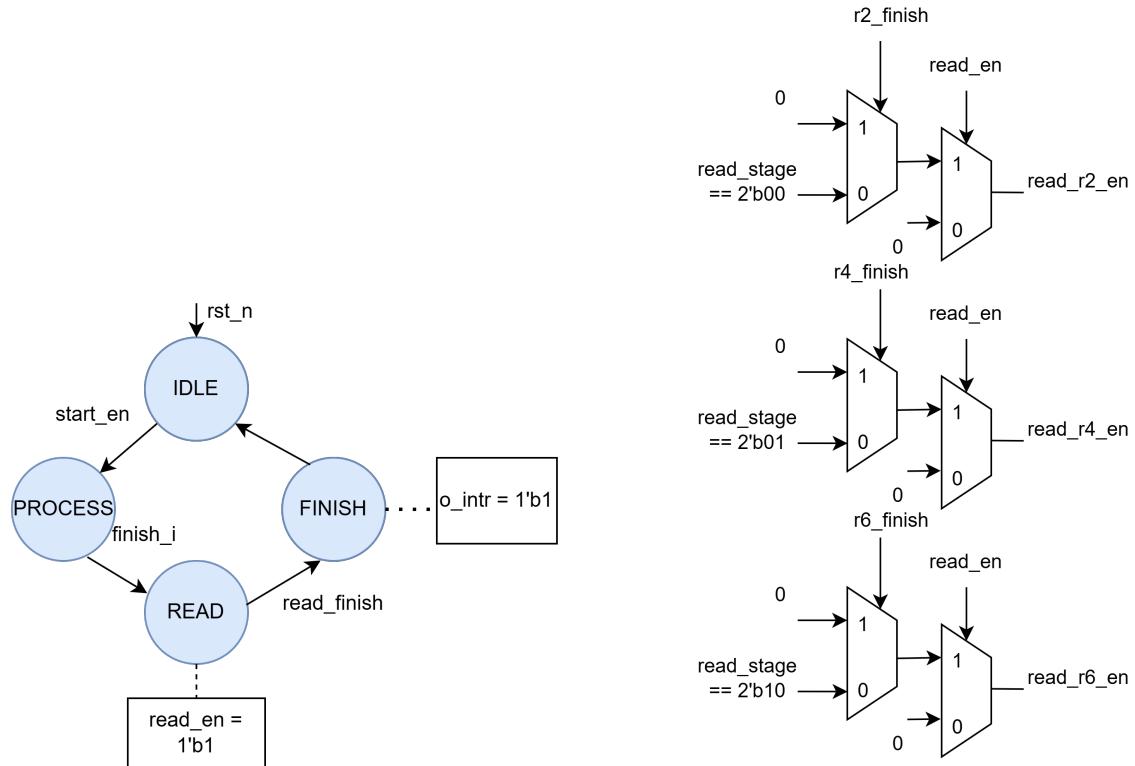
mô-đun MRELBP hay còn gọi là top mô-đun, là mô-đun chứa các thành phần đã nêu ở các phần trên, đồng thời còn có thêm các tín hiệu điều khiển quá trình đọc ghi. Vì mô-đun JointHistogram sẽ lưu giá trị đầu ra trong một bộ nhớ, và sẽ được đọc ra theo một yêu cầu nhất định, vì vậy, mô-đun MRELBP ngoài là một lớp chứa các thành phần trên ra còn có thêm các tín hiệu điều khiển cho mô-đun JointHistogram đọc các giá trị từ trong bộ nhớ của nó ra ngoài. Nguyên nhân là vì thực tế chỉ có 1 đầu ra mà sẽ có 3 đặc trưng ứng với 3 bán kính khác nhau, nên cần có một bộ điều phối đầu ra của cả 3 sao cho chúng là tuần tự.



Hình 3.27. Kiến trúc RTL của mô-đun JointHistogram



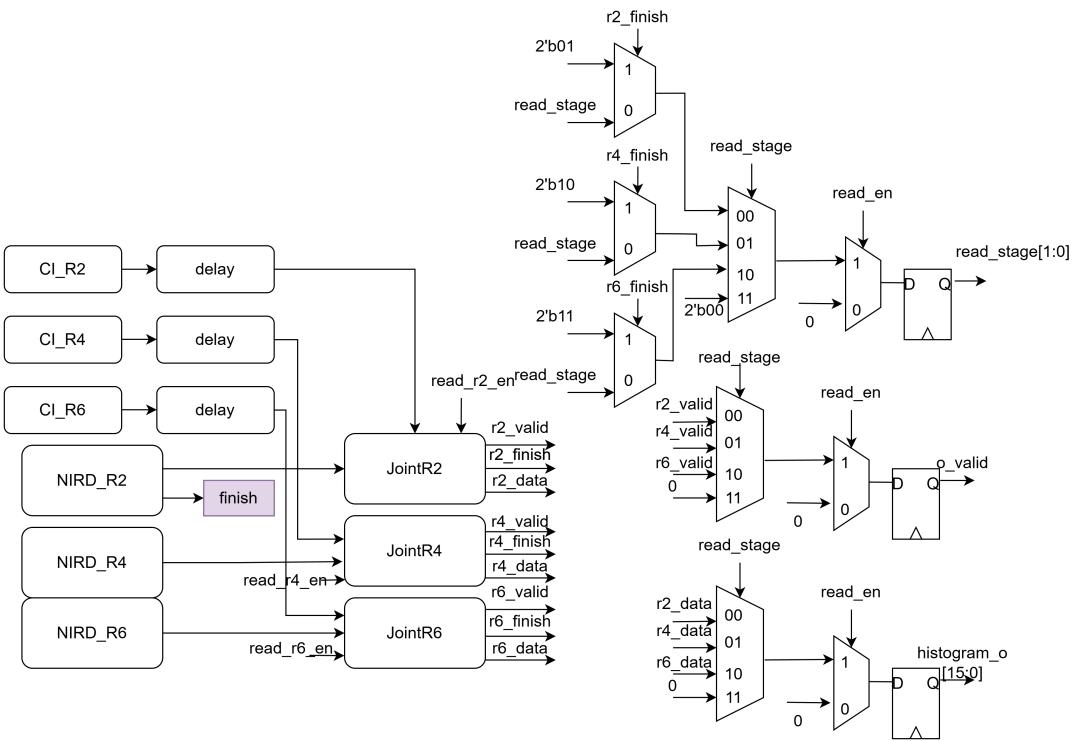
Hình 3.28. Sơ đồ chuyển trạng thái của mô-đun JointHistogram



3.6. Tính toán thời gian hoạt động của toàn bộ mô-đun

Bảng 3.6. Số chu kỳ thực hiện của các mô-đun ZeroPadding

Tên mô-đun	Số chu kỳ thực hiện
Buffer6Rows	$3 * \text{COLS}$ chu kỳ
ZeroPadding3x3	3 chu kỳ
ZeroPadding5x5	4 chu kỳ
ZeroPadding7x7	5 chu kỳ
MedianCalculation3x3	4 chu kỳ
MedianCalculation5x5	14 chu kỳ
MedianCalculation7x7	35 chu kỳ
MedianProcessing	$3 * \text{COLS} + 40$ chu kỳ
Tổng với kích thước ảnh $H * W$	$(H + 3) * W + 635$ chu kỳ



Hình 3.31. Mô tả các tín hiệu yêu cầu đọc dữ liệu (2) của mô-đun MRELBP

Minh chứng mô phỏng với các kích thước ảnh như sau:

- Ảnh đầu vào kích thước 128 * 128: 17403 chu kỳ
- Ảnh đầu vào kích thước 256 * 256: 66939 chu kỳ



Hình 3.32. Kết quả mô phỏng với kích thước ảnh 128x128

CHƯƠNG 4

MÔ PHỎNG VÀ KIỂM THỬ

Để đánh giá độ chính xác của thiết kế, một bước quan trọng trong quá trình thiết kế là cần có các bài mô phỏng và các trường hợp kiểm thử. Chương 4 sẽ mô tả chi tiết về một số phương pháp kiểm thử được sử dụng, các điều kiện để đánh giá độ chính xác của kiểm thử, mô hình giúp tự động hóa quá trình kiểm thử.

4.1. Cơ sở lý thuyết kiểm thử

4.1.1. Lý thuyết về độ bao phủ

Độ bao phủ (coverage) là phương pháp thống kê trong quá trình kiểm tra thiết kế để xác định được chất lượng của quá trình kiểm tra. Độ bao phủ là một yếu tố quan trọng để khẳng định được thiết kế có được kiểm tra theo các yêu cầu đưa ra hay chưa. Từ đó, có thể xem xét tạo ra các bài kiểm tra để đảm bảo bao quát được các trường hợp.

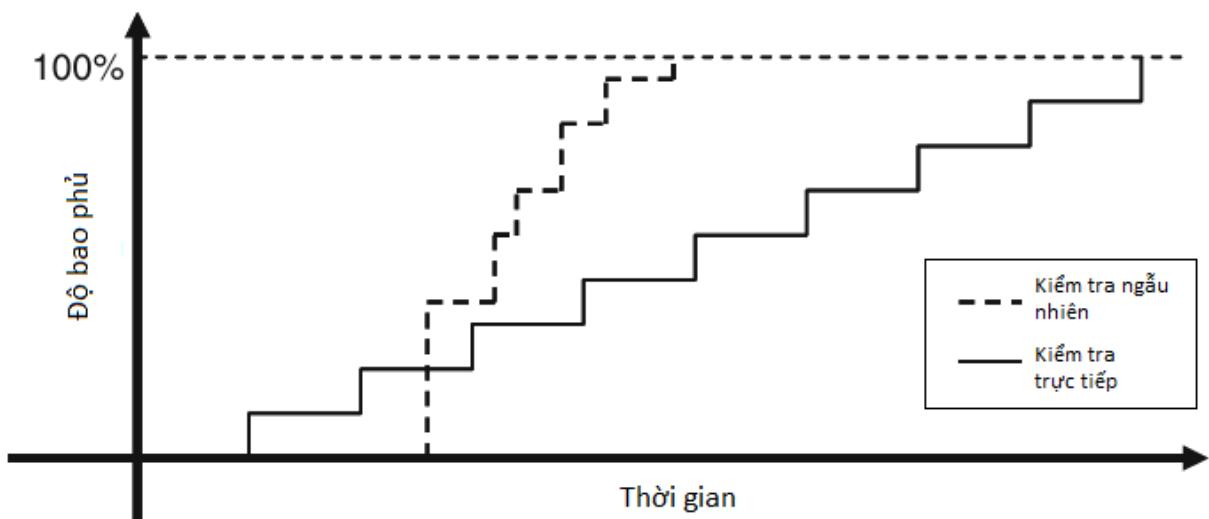
Có hai yếu tố bao phủ thường được đánh giá là độ bao phủ về mã nguồn (code coverage) và độ bao phủ về mặt chức năng (functional coverage). Trong đó, độ bao phủ về mã nguồn sẽ đánh giá xem là các điều kiện ví dụ các điều kiện rẽ nhánh có được thực hiện ít nhất 1 lần hay không, do đó nó không có ý nghĩa về mặt đánh giá độ chính xác của thiết kế mà chỉ đảm bảo rằng bộ kiểm thử sinh ra đã bao quát được hết mã nguồn. Độ bao phủ về mặt chức năng sẽ đánh giá độ bao phủ của thiết kế dựa trên tính năng đã được kiểm tra. Do đó, việc định nghĩa ra các tính năng nào cần được đánh giá sẽ quyết định chất lượng của độ bao phủ về mặt chức năng. Độ bao phủ về mặt mã nguồn thường được thu thập từ các phần mềm sử dụng để thiết kế ví dụ Vivado, QuetaSim,... Trong đó, độ bao phủ về mặt chức năng phải do kỹ sư tự định nghĩa, kỹ sư cần tự định nghĩa điểm cần bao phủ. Ví dụ về kiểm tra về mặt chức năng, giả sử kỹ sư cần kiểm tra một biến có thay đổi theo như mong muốn hay không. Nếu đúng, biến đấy phải thay đổi giá trị từ 3 -> 5 -> 7 chẳng hạn. Độ bao phủ về mặt mã nguồn chỉ biết là biến đấy đã thành 3, 5 hoặc 7 mà không thể đánh giá được về mặt thứ tự xuất hiện.

4.1.2. Các phương pháp kiểm thử

Trong thiết kế nói chung, sau khi đã thiết kế được các mạch từ mã nguồn hoặc thông qua các công cụ, kỹ sư cần thiết kế các bài kiểm thử để đánh giá tính đúng đắn của

thiết kế. Thông thường, đối với thiết kế số, kỹ sư sẽ thiết kế ra các phương pháp kiểm thử trực tiếp (directed testing). Khi sử dụng phương pháp này, kỹ sư sẽ nhìn vào các yêu cầu kỹ thuật, viết ra các trường hợp có thể xảy ra và kiểm tra nó với một kết quả đã biết trước. Để kiểm tra, kỹ sư có thể nhìn dạng sóng mô phỏng, sử dụng các điều kiện để kiểm tra. Đây là một phương pháp nhanh, phù hợp với kỹ sư cần kiểm tra ngay lập tức thiết kế là đúng hay sai, nhưng sẽ khó để đảm bảo hoàn toàn về mặt thiết kế. Giả sử, nếu thiết kế có 10, 100 tín hiệu đầu vào, vậy nếu viết tay và tính toán các trường hợp kiểm thử sẽ rất tốn thời gian và có thể dễ mắc đến các sai lầm.

Vậy để có thể kiểm tra được toàn bộ thiết kế một cách đầy đủ, các kỹ sư sẽ thiết kế ra các kích thích sinh ngẫu nhiên với ràng buộc (constrained-random stimulus). Các kích thích này sẽ sinh ra các tín hiệu ngẫu nhiên giá trị, tuy nhiên phải tuân theo các ràng buộc cho trước. Ví dụ, kỹ sư có thể yêu cầu khi sinh ra dữ liệu điểm ảnh, thì tín hiệu hợp lệ phải ở mức 1. Từ đó, kỹ sư có thể kiểm tra được nhiều trường hợp hơn với những dữ liệu ngẫu nhiên đó. Hình 4.1 mô tả biểu đồ phân tích thời gian và độ bao phủ của các bộ kiểm thử. Có thể thấy, với cách tạo ra các kích thích trực tiếp, thời gian để đạt độ bao phủ cần thiết là rất lớn, trong khi đó với các kích thích được sinh ngẫu nhiên với các ràng buộc, thời gian này đã giảm đi một cách đáng kể.

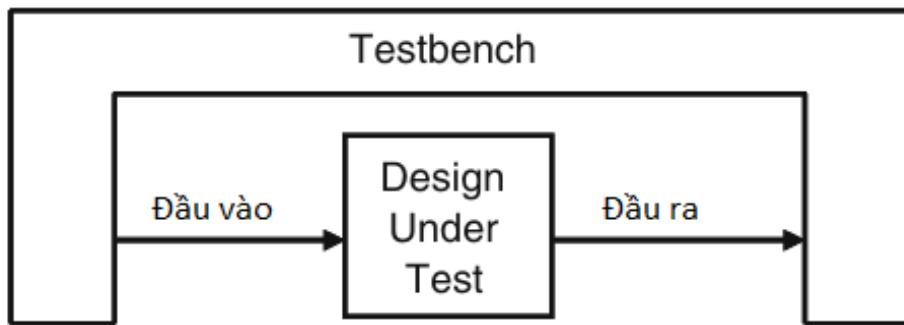


Hình 4.1. Biểu đồ phân tích thời gian và độ bao phủ của các bộ kiểm thử

4.2. Phương pháp xây dựng kiểm thử và kết quả

Hình 4.2 mô tả các thành phần cơ bản của một bộ kiểm tra, bao gồm testbench sẽ sinh ra các kích thích đầu vào và sau đó thu thập các dữ liệu đầu ra. Trong khi thiết kế, kỹ sư sẽ cần kiểm tra ngay lập tức chức năng của một hoặc nhiều mô-đun để đảm bảo

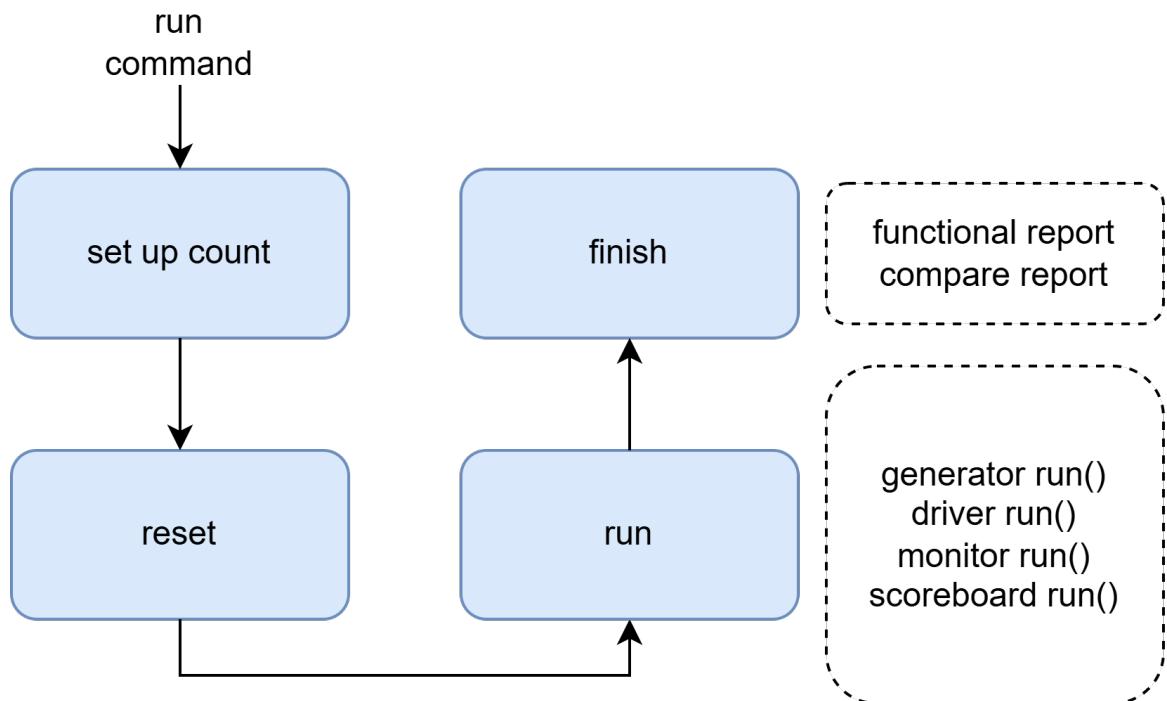
tính chính xác một cách chưa đầy đủ, tức là kiểm tra mô-đun đó đã hoạt động đúng tính năng mong muốn hay chưa, tuy nhiên có thể có những trường hợp ngoại lệ mà kỹ sư có thể không nghĩ đến. Do đó, thường thì kỹ sư sẽ kiểm tra bằng cách tạo ra các bộ kiểm thử với kích thích sinh trực tiếp để tiết kiệm về mặt thời gian. Do đó, đối với các mô-đun nhỏ, sinh viên sẽ tự đưa ra các bộ kích thích trực tiếp và sau đó kiểm tra kết quả với đầu ra đã có trước.



Hình 4.2. Các thành phần cơ bản của một bộ kiểm tra

Tuy nhiên, để kiểm tra tự động với những đầu vào ngẫu nhiên đồng thời đánh giá về độ bao phủ của bộ kiểm tra, sinh viên đã định nghĩa ra một bộ kiểm thử có kiến trúc được mô tả tại hình 4.4. *Lưu ý: Ngôn ngữ sử dụng để xây dựng bộ kiểm tra phân tầng là SystemVerilog.* Về cơ bản, bộ kiểm tra này sẽ gồm các lớp thực hiện các chức năng riêng biệt theo một kịch bản đã định nghĩa trước. Đầu tiên là lớp Transaction, lớp này sẽ định nghĩa ra các giao dịch, các ràng buộc để tạo dữ liệu đầu và đồng thời đưa ra các phương thức để sao chép. Lớp Generator sẽ có nhiệm vụ thực hiện việc sinh ra các dữ liệu đầu vào và sau khi sinh xong hết, nó sẽ có một lời gọi hệ thống để chạy 1 bộ tính toán ra dữ liệu mong muốn ở bên ngoài bộ kiểm tra. Dữ liệu được sinh ra sẽ được đẩy xuống lớp Driver qua một cách vận chuyển gọi là mailbox. Đây là các để chuyển dữ liệu qua 2 lớp của ngôn ngữ SystemVerilog. Lớp Driver sẽ thực hiện hai phương thức chính là **reset** và **run**. Trong đó reset giúp khởi động lại toàn bộ hệ thống, còn run sẽ thực hiện đưa dữ liệu đó vào DUT để thực hiện tính toán. Lớp Monitor sẽ liên tục giám sát DUT, nếu đã có dữ liệu đầu ra, lớp này sẽ bắt lại nó, sau đó đưa vào lớp Scoreboard. Lớp Scoreboard sẽ đọc dữ liệu mong muốn và dữ liệu thực tế, so sánh và đưa ra đánh giá. Trong lúc tất cả các lớp khác đang hoạt động, lớp đánh giá độ bao phủ về mặt chức năng cũng đồng thời

đánh giá các trường hợp. Sau khi đã kiểm tra hết số lượng kiểm thử đã định nghĩa, một báo cáo về độ bao phủ về mặt chức năng sẽ được đưa ra. Các event là một kiểu dữ liệu của SystemVerilog dùng để chờ hoặc kích hoạt nhằm đảm bảo kịch bản sẽ hoạt động chính xác theo trình tự.

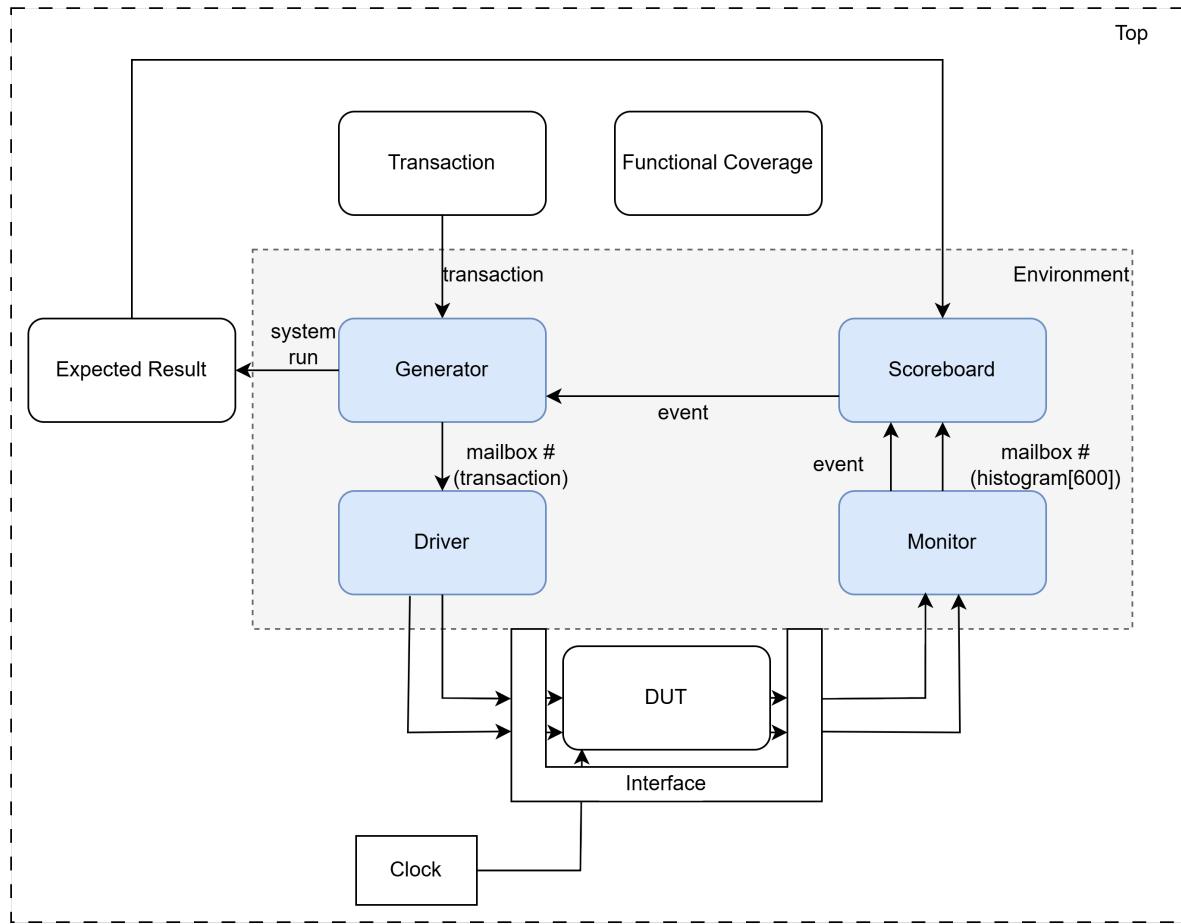


Hình 4.3. Luồng hoạt động của bộ kiểm thử

Hình 4.3 mô tả luồng hoạt động của bộ kiểm tra phân tầng. Bộ kiểm tra có thể tùy ý thay đổi số lượng test-bench kiểm tra dựa vào biến **count**. Sau khi đặt xong giá trị đếm, có thể bắt đầu kiểm tra hoạt động với hoạt động cài lại, sau đó các lớp con sẽ tiến hành chạy, sau khi đã hoàn thành đủ số lượng test-bench, tiến hành đưa ra 2 báo cáo bao gồm độ bao phủ chức năng và so sánh tỉ lệ chính xác với phiên bản phần mềm.

Để đánh giá về độ bao phủ về mặt chức năng, kỹ sư cần đưa ra các điểm cần bao phủ. Đối với ngôn ngữ SystemVerilog, ngôn ngữ hỗ trợ các tính năng sau giúp đánh giá độ bao phủ:

- Coverpoint - điểm bao phủ: là điểm mà sẽ thực hiện đánh giá bao phủ, thực chất điểm ở đây là các tín hiệu hoặc các biến
- Covergroup - nhóm bao phủ: là tập hợp các điểm bao phủ sẽ được lấy mẫu.



Hình 4.4. Bộ kiểm tra phân tầng

- Bins định nghĩa ra các trường hợp mà coverpoint sẽ cần xảy ra, nếu coverpoint xuất hiện giá trị được định nghĩa trong bins thì coi như là "hit" nếu không xuất hiện bất cứ lần nào thì là "miss".
- Cross: có thể gộp 2 hoặc nhiều coverpoint theo các tổ hợp có thể xảy ra với cross, ví dụ nếu định nghĩa coverpoint tín hiệu `i_valid`, thì có thể cross `i_valid` và dữ liệu để đánh giá xem trường hợp khi giá trị đầu vào là hợp lệ thì dữ liệu có hợp lệ hay không.
- `Covergroup.get_coverage()`: sẽ đưa ra tỉ lệ hit trên tổng số các trường hợp có thể xảy ra của các coverpoint được định nghĩa trong covergroup đó.

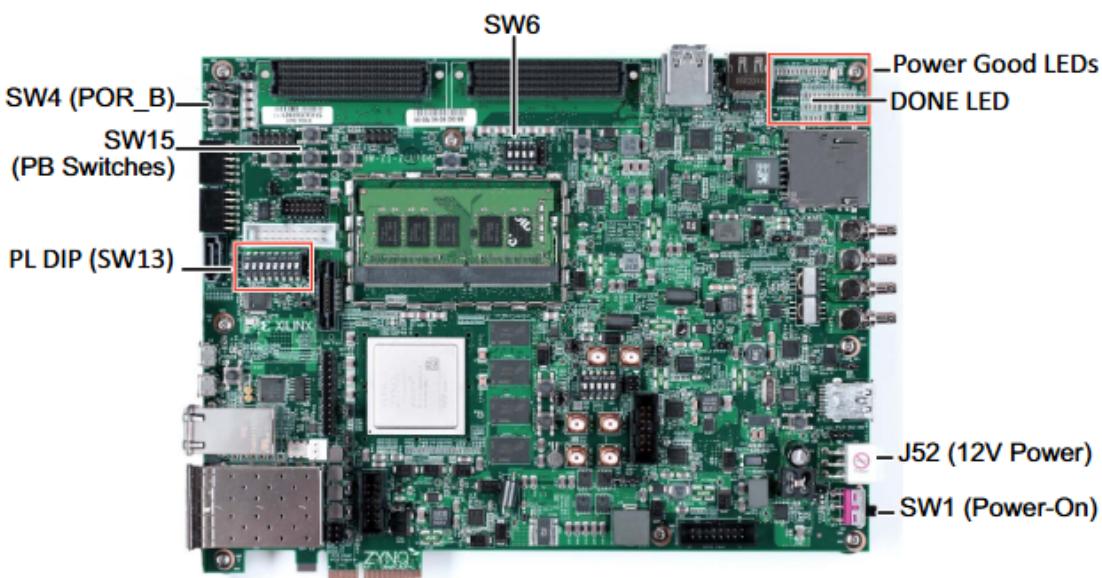
CHƯƠNG 5

THỰC THI VÀ ĐÁNH GIÁ

Chương 5 sẽ mô tả về quá trình thực thi và kiểm thử thực tế với một hệ thống System on Chip để đánh giá về hiệu năng, đồng thời chứng minh IP đã thiết kế đảm bảo khả năng hoạt động với các thành phần khác.

5.1. Thông tin về bo mạch Zynq UltraScale+MPSoC ZCU106 Evaluation Kit

ZCU106 Evaluation Kit là bo mạch được phát triển để phục vụ mục đích thiết kế và tạo mẫu các hệ thống nhúng trên nền tảng của nhà sản xuất Xilinx. Nó là tổ hợp của hai phần gồm hệ thống xử lý (PS) và FPGA (PL). Với PL là hệ logic khả trinh cho phép người dùng lập trình phần cứng để thực hiện các chức năng chuyên biệt với công nghệ 16nm FinFET+. PS là ARM flagship Cortex-A53 64-bit quad-core processor và Cortex-R5F dual-core real-time processor. Ngoài ra trên bo mạch còn tích hợp rất nhiều các ngoại vi như HDMI, Ethernet, ...

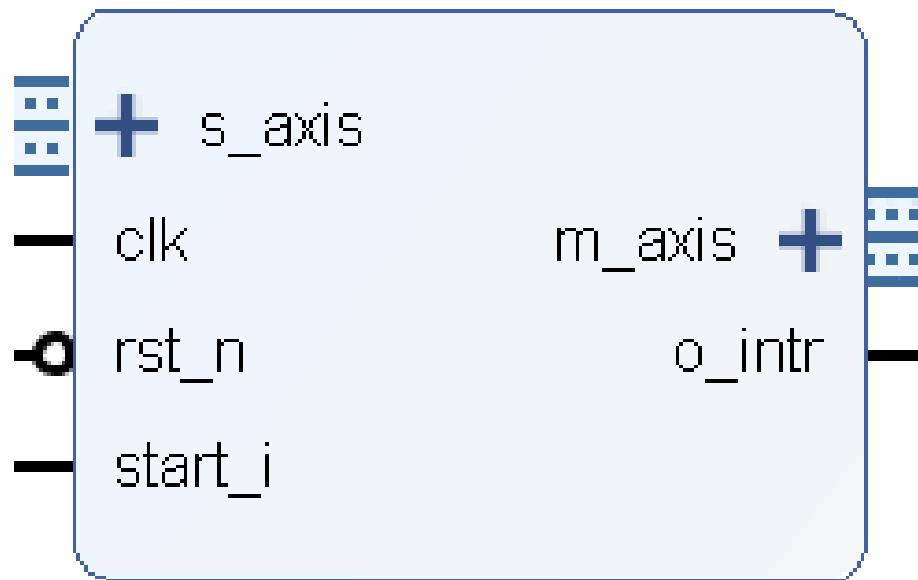


Hình 5.1. Bo mạch ZCU106 UltraScale+MPSoC

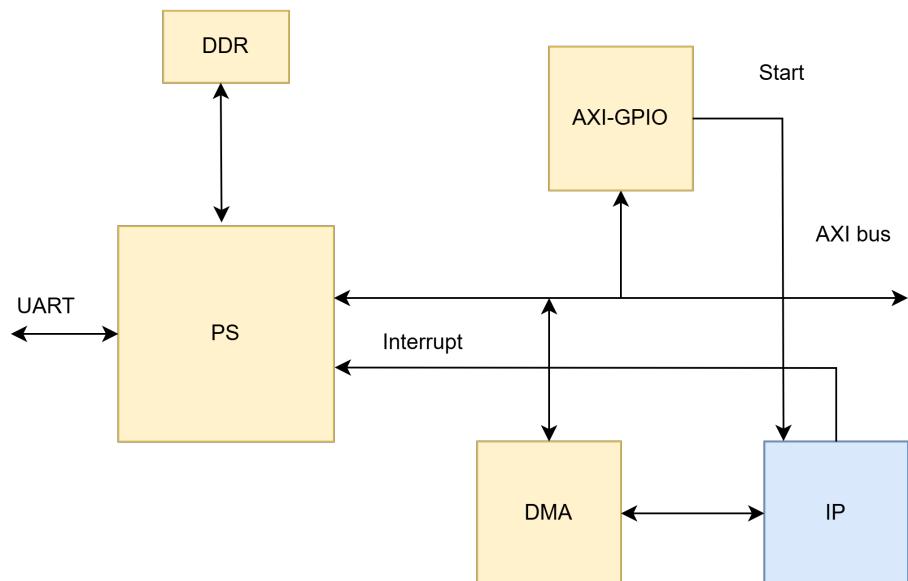
5.2. Xây dựng hệ thống SoC

Sau khi đã có những thiết kế RTL, cần đóng gói thiết kế thành IP. IP sẽ sử dụng các giao tiếp theo chuẩn giao tiếp AXI4-Stream. Giao diện của IP được mô tả tại hình 5.2

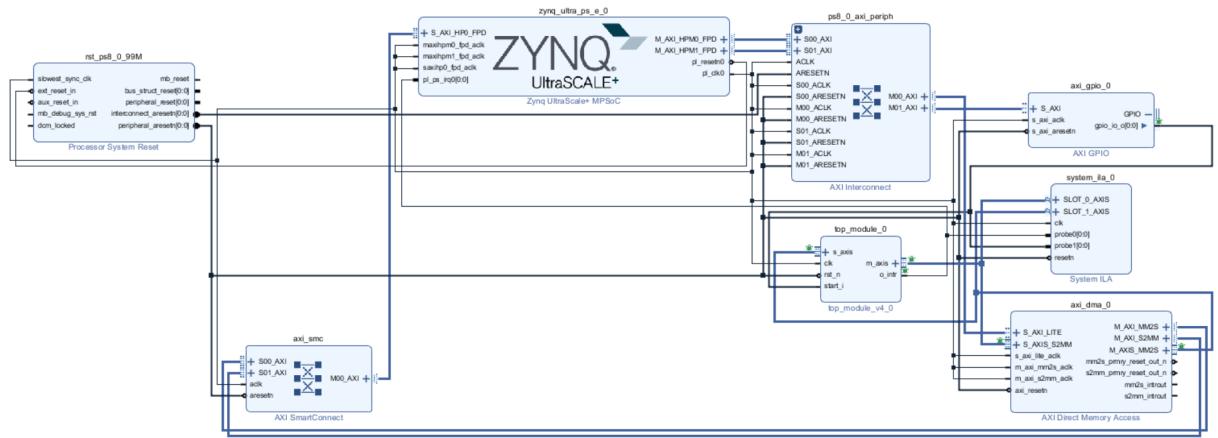
Hình 5.3 mô tả về sơ đồ khối của hệ thống SoC sẽ được xây dựng. Ảnh có thể được lưu trữ trong DDR, sau đó DMA sẽ lấy dữ liệu ảnh trong DDR đó, chuyển vào khối IP. Vì có tín hiệu **start** để yêu cầu việc bắt đầu quá trình nên sẽ có một khối GPIO được điều khiển bởi PS để thực hiện bật, tắt chân start. Từ IP, một tín hiệu ngắn cũng sẽ được nối trở về PS để thực hiện thông báo ngắn khi cần.



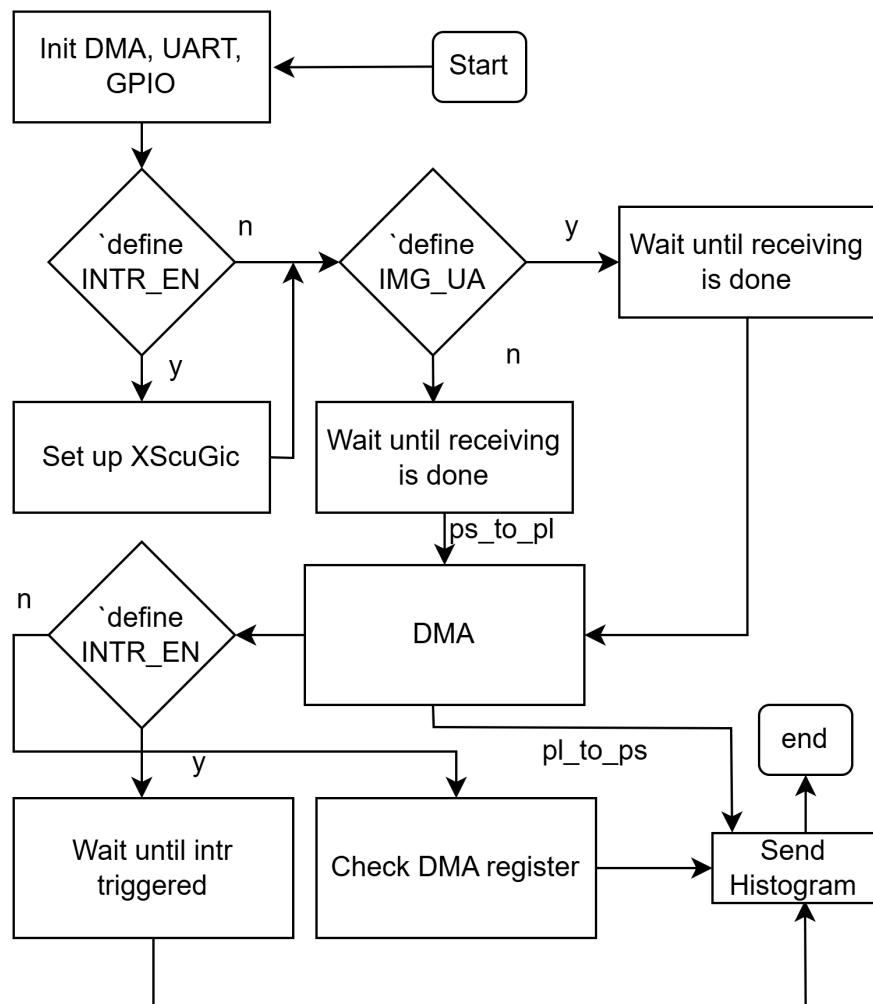
Hình 5.2. Giao diện của IP MRELBP



Hình 5.3. Sơ đồ khối hệ thống SoC



Hình 5.4. Thiết kế khối hệ thống SoC



Hình 5.5. Lưu đồ hoạt động của chương trình phần mềm

Hình 5.5 mô tả lưu đồ hoạt động của chương trình phần mềm cho hệ thống SoC đã thiết kế. Khi chương trình bắt đầu thì sẽ cần khởi tạo các khôi cần sử dụng bao gồm DMA, UART và GPIO. Nếu định nghĩa một cờ là INTR_EN thì sẽ cần thiết lập về mức ưu tiên ngắn và thời điểm kích hoạt ngắn. Nếu định nghĩa cờ IMG_UA thì sẽ kích hoạt UART nhận dữ liệu từ bên ngoài, nếu không sẽ lấy được ảnh đã lưu trữ cục bộ, sau đó thông qua bộ DMA đưa dữ liệu vào IP đã thiết kế. Nếu định nghĩa cờ ngắn, quá trình đợi tính toán hoàn tất sẽ kết thúc khi mà ngắn được kích hoạt, nếu không sẽ cần kiểm tra các thanh ghi của DMA để xem DMA đã hoàn tất quá trình hoạt động hay chưa. Sau khi kết thúc có thể tiến hành gửi các giá trị đặc trưng qua kết nối UART để quan sát hoặc sử dụng cho mục đích khác.

5.3. Kết quả và đánh giá

Kết quả về tổng hợp và thực thi sẽ được thực hiện trên phần mềm Vivado. Kết quả tổng hợp bao gồm về tài nguyên sử dụng, tần số tối đa, năng lượng tiêu thụ. Để đánh giá về độ chính xác của mô hình sẽ thực hiện với tập dữ liệu Outex, so sánh thời gian thực tế với thời gian chạy của các mô hình phần mềm.

Tài nguyên sử dụng được mô tả trong bảng 5.1

Site Type	Used	Fixed	Available	Util%
CLB LUTs*	20463	0	230400	8.88
LUT as Logic	19076	0	230400	8.28
LUT as Memory	1387	0	101760	1.36
LUT as Distributed RAM	240	0		
LUT as Shift Register	1147	0		
CLB Registers	22559	0	460800	4.90
Register as Flip Flop	22559	0	460800	4.90
Register as Latch	0	0	460800	0.00
CARRY8	1714	0	28800	5.95
F7 Muxes	0	0	115200	0.00
F8 Muxes	0	0	57600	0.00
F9 Muxes	0	0	28800	0.00

Hình 5.6. Báo cáo kết quả tổng hợp của phần mềm Vivado

Bảng 5.1. Tài nguyên sử dụng

Tài nguyên	Sử dụng	Có sẵn	Phần trăm	Wang [16]	HLS
LUT	20463	230400	8.88%	49033	30378
FF	22559	460800	4.9%	19969	13713
BRAMs	31.5	312	10.1%	41.5	88

Thông số định thời của hệ thống được thể hiện trên hình 5.7. Với các thông số định thời ta sẽ có một số tiêu chí đánh giá như sau:

- WNS: Độ trễ trong trường hợp xấu nhất của một xung nhịp, $WNS = \min(T - T_n)$ với T là một chu kỳ của xung nhịp, T_n là thời gian truyền. Để mạch hoạt động chính xác thì phải đảm bảo giá trị WNS lớn hơn 0.
- Tần số F_{max} : Tần số có thể chạy trên phần cứng trong một triển khai nhất định.

Hệ thống được xây dựng với ràng buộc đầu là 100MHz tương đương với chu kỳ xung nhịp là 10ns, từ đó ta có thể tính tần số tối đa như sau:

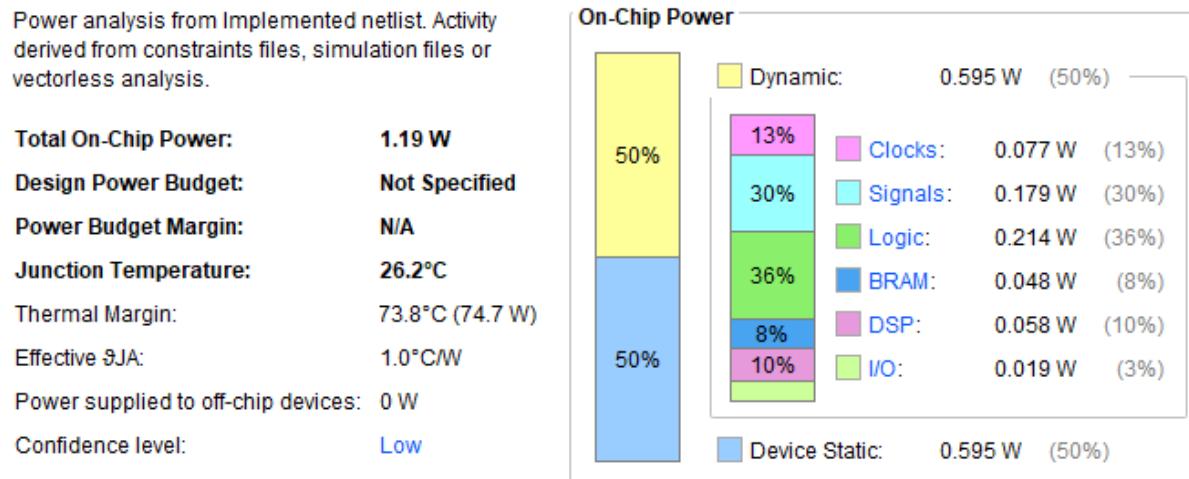
$$F_{max} = \frac{1}{T - WNS} = \frac{1}{(10 - 5.050) * 10^{-9}} = 202(MHz)$$

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.050 ns	Worst Hold Slack (WHS): 0.010 ns	Worst Pulse Width Slack (WPWS): 4.458 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 47824	Total Number of Endpoints: 47824	Total Number of Endpoints: 23967

Hình 5.7. Thông số định thời của thiết kế

Công suất tiêu thụ của hệ thống được thể hiện trong hình 5.8. Công suất tiêu thụ của mạch khi hoạt động là 0.595 Watt. Đây là mức tiêu thụ năng lượng không quá lớn để tích hợp với các hệ thống khác vì hoạt động của IP rất phức tạp và cần tối ưu về mặt đáp ứng thời gian thực.

Để kiểm tra độ chính xác của thiết kế, ta sẽ đánh giá đặc trưng đã trích xuất với tập dữ liệu Outex[20] và các mô hình cổ điển. Dữ liệu từ tập dữ liệu Outex được xây dựng nhằm phát triển một bộ khung linh hoạt và cơ sở dữ liệu hình ảnh phục vụ cho việc đánh giá thực nghiệm các thuật toán phân tích kết cấu. Dữ liệu này bao gồm một tập



Hình 5.8. Công suất tiêu thụ năng lượng của thiết kế

hợp lớn các kết cấu bề mặt được thu thập dưới nhiều điều kiện khác nhau, giúp xây dựng nhiều bài toán phân tích kết cấu đa dạng. 4 mô hình học máy cổ điển được sử dụng là SVM (Support Vector Machine), Logistic Regression, Naive Bayes và KNN (K-Nearest Neighbor) với K bằng 3.

Bảng 5.2. Độ chính xác trên các tập dữ liệu với SVM

Tập dữ liệu	MRELBP(1)	MRELBP(2)	MRELBP(3)	MRELBP(4)	MRELBP(5)
TC_00010_r	100	98.75	99.79	100	100
TC_00010_c	100	98.75	99.79	100	100
TC_00011_r	94.58	92.5	93.5	94.58	92.29
TC_00012_i	99.58	97.58	98.54	99.58	97.92
TC_00020	99.17	97.5	99.17	99.17	99.58
TC_00023	97.92	98.54	98.12	97.92	98.33
TC_00024	96.46	92.71	96.25	96.46	96.88
TC_00030	99.17	97.5	99.17	99.17	99.38
TC_00033	97.08	96.46	97.08	97.08	96.25
TC_00034	96.67	92.71	96.25	96.67	96.88

Bảng 5.2, 5.3, 5.4, 5.5 là kết quả đánh giá dựa trên tập huấn luyện Outex. Với MRELBP(1) là phiên bản với các giá trị bán kính là 2, 4, 6, thực hiện đúng các thao tác của công thức 1.5, 1.6, 1.7. MRELBP(2) với công thức CI theo 2.1. MRELBP(4) là phiên bản với các giá trị tính nội suy tham chiếu được cố định theo 24-bit trong đó 16-bit biểu diễn phần thập phân. Phiên bản MRELBP(3) là phiên bản với các giá trị tính nội suy cố định và CI tối ưu cho phần cứng. Phiên bản MRELBP(5) là phiên bản với các giá

Bảng 5.3. Độ chính xác trên các tập dữ liệu với Logistic Regression

Tập dữ liệu	MRELBP(1)	MRELBP(2)	MRELBP(3)	MRELBP(4)	MRELBP(5)
TC_00010_r	99.58	99.92	100	99.58	100
TC_00010_c	99.58	97.5	100	99.58	100
TC_00011_r	88.96	91.46	93.75	88.96	93.13
TC_00012_i	95.21	92.71	96.04	95.21	96.67
TC_00020	96.46	94.79	98.75	96.46	98.33
TC_00023	96.88	97.29	98.33	96.88	97.92
TC_00024	89.8	92.5	93.96	89.8	96.04
TC_00030	96.67	95.0	98.96	96.67	98.33
TC_00033	96.67	96.25	98.33	96.67	97.29
TC_00034	86.88	91.88	95.42	86.88	95.63

trị bán kính là 2, 4, 6, 8 với các giá trị tính nội suy cố định và CI tối ưu cho phần cứng. Kết quả đánh giá cho thấy, so với phiên bản cùng số lượng bán kính thì kết quả đạt được không có nhiều sự khác biệt, tương tự với phiên bản có thêm giá trị bán kính bằng 8 cũng cho thấy sự chênh lệch là rất ít.

Bảng 5.4. Độ chính xác trên các tập dữ liệu với Naive Bayes

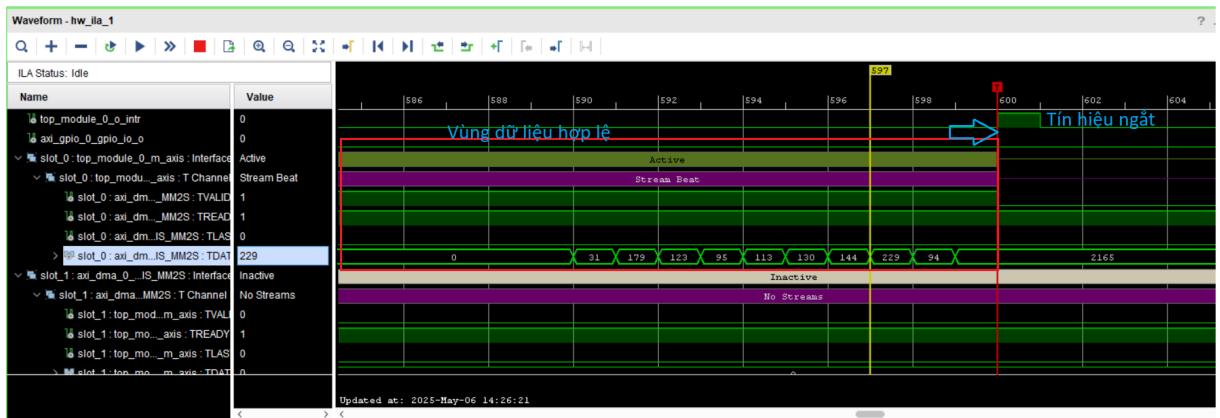
Tập dữ liệu	MRELBP(1)	MRELBP(2)	MRELBP(3)	MRELBP(4)	MRELBP(5)
TC_00010_r	98.33	97.67	98.75	98.33	98.75
TC_00010_c	98.33	97.67	98.75	98.33	98.75
TC_00011_r	95	93.54	95.63	95	97.08
TC_00012_i	98.75	94.17	98.96	98.3	99.17
TC_00020	95.42	93.23	96.67	95.2	96.04
TC_00023	97.29	96.13	97.5	97.1	96.45
TC_00024	96.04	93.5	96.04	95.82	96.04
TC_00030	95.42	95.0	96.875	94.67	96.04
TC_00033	97.29	95.7	97.5	96.87	96.25
TC_00034	96.04	94.5	96.04	94.01	96.04

Thời gian đo lường được với hệ thống SoC với kích thước ảnh 128x128 và tần số hoạt động 100MHz là 250 μ s. Thời gian đo lường với phần mềm xây dựng với ngôn ngữ c++ với các tập dữ liệu như trên lần lượt là 0.4282s, 0.4326s, 0.4173s, 0.4402s, 0.4386s, 0.4389s, 0.4313s, 0.4313s, 0.4158s, 0.4177s. Do đó, tỉ lệ tăng tốc của bộ tăng tốc phần

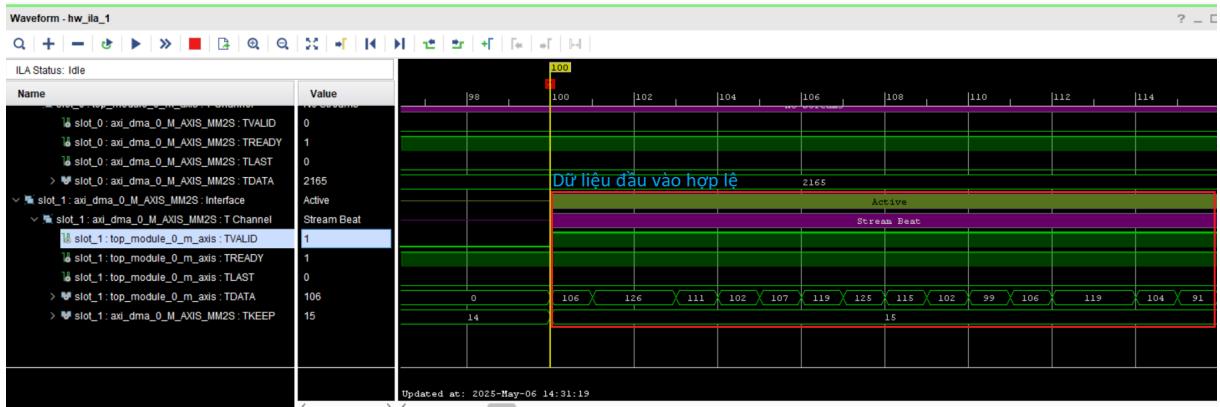
Bảng 5.5. Độ chính xác trên các tập dữ liệu với KNN

Tập dữ liệu	MRELBP(1)	MRELBP(2)	MRELBP(3)	MRELBP(4)	MRELBP(5)
TC_00010_r	100	99.58	99.79	100	100
TC_00010_c	100	99.58	99.79	100	100
TC_00011_r	96.05	96.25	97.08	96.04	98.8
TC_00012_i	100	99.17	100	100	99.79
TC_00020	99.79	99.58	99.79	99.79	100
TC_00023	99.38	97.5	99.38	99.38	98.54
TC_00024	97.71	97.08	97.92	97.71	98.33
TC_00030	99.79	99.58	99.79	99.79	100
TC_00033	99.17	97.92	99.17	99.17	99.17
TC_00034	97.71	97.1	97.92	97.71	98.33

cứng là $\frac{0.429}{250 \cdot 10^{-6}} = 1716$ lần.



Hình 5.9. ILA với điểm kích hoạt là *o_intr*



Hình 5.10. ILA với dữ liệu điểm ảnh đầu vào

Để giám sát hoạt động của hệ thống sau khi triển khai, một IP được sử dụng trong thiết kế là Integrated Logic Analyzer (ILA). Kết quả thu được tương ứng với các hình 5.9, 5.10. Hình 5.9 cho thấy sau ngay sau khi nhận xong toàn bộ dữ liệu, tín hiệu ngắn được kích hoạt lên mức cao trong 1 chu kỳ, đảm bảo vi xử lý có thể xử lý được dữ liệu ngay khi hoàn tất quá trình tính toán.

KẾT LUẬN

Dựa trên những kết quả đã đạt được, đồ án đã hoàn thành các mục tiêu đã đề ra. Cụ thể, đồ án đã cung cấp đầy đủ thông tin về kỹ thuật trích xuất đặc trưng sử dụng MRELBP, thiết kế phần cứng tăng tốc ở mức RTL và triển khai hệ thống SoC nhằm kiểm chứng hoạt động của IP. Các nhiệm vụ chính đã được thực hiện bao gồm:

- Phân tích và triển khai thuật toán MRELBP dựa trên cơ sở lý thuyết.
- Phân tích yêu cầu và đưa ra thiết kế phần cứng ở mức RTL phù hợp.
- Xây dựng được hệ thống kiểm thử thiết kế một cách tự động, đánh giá được độ bao phủ về mặt chức năng.
- Xây dựng hệ thống SoC nhằm kiểm chứng khả năng tương thích và hoạt động ổn định của IP.
- Thực nghiệm và đánh giá hiệu quả của hệ thống thông qua các kết quả thu được.

Tuy nhiên, đồ án vẫn còn tồn tại một số hạn chế nhất định, cụ thể:

- Công suất tiêu thụ của hệ thống vẫn còn tương đối cao, chưa phù hợp cho các ứng dụng yêu cầu tiết kiệm năng lượng.
- Một số khối trong thiết kế chưa được áp dụng kỹ thuật pipelining một cách tối ưu, ảnh hưởng đến hiệu suất xử lý tổng thể.

Những điểm hạn chế nêu trên là cơ sở để định hướng cho việc cải tiến, tối ưu thiết kế trong các nghiên cứu và phát triển tiếp theo trong tương lai.

TÀI LIỆU THAM KHẢO

Tiếng Anh

- [1] Li Liu, Songyang Lao, Paul W. Fieguth, Yulan Guo, Xiaogang Wang, Matti Pietikäinen, "Median Robust Extended Local Binary Pattern for Texture Classification", *IEEE Transactions on Image Processing*, 2016.
- [2] Li Liu, Songyang Lao, Paul W. Fieguth, Yulan Guo, Xiaogang Wang, Matti Pietikäinen, "Local binary features for texture classification: Taxonomy and experimental study", *Pattern Recognition Volumne 62*, 2017.
- [3] enjamin Stratton , "AI Industry Growth Statistics: Exploring Key Metrics Driving Growth of AI (updated for 2025)", 2025.
[Online]. Available: <https://bluetree.digital/ai-industry-growth-metrics>
- [4] "The state of AI in 2022—and a half decade in review", 2022.
[Online]. Available: <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2022-and-a-half-decade-in-review>
- [5] "AI Accelerator Market Size, Share & Trends Analysis Report By AI Accelerator Types (GPUs, TPUs), By Technology Integration, By End-use (IT & Telecom, Automotive), By Region, And Segment Forecasts, 2024 - 2030", 2023.
[Online]. Available: <https://www.grandviewresearch.com/industry-analysis/ai-accelerator-market-report>
- [6] "FPGAs for Artificial Intelligence: Possibilities, Pros, and Cons", 2018.
[Online]. Available: <https://www.apriorit.com/dev-blog/586-fpgas-for-ai>
- [7] Gaurav Kumar, Pradeep Kumar Bhatia , "A Detailed Review of Feature Extraction in Image Processing Systems", *2014 Fourth International Conference on Advanced Computing & Communication Tecnologies*, 2014.
- [8] April Khademi; Sridhar Krishnan , "Medical image texture analysis: A case study with small bowel, retinal and mammogram images", *2008 Canadian Conference on Electrical and Computer Engineering*, 2008.
- [9] Rodrigo da Silva Ferreira; Andrea Britto Mattos; Emilio Vital Brazil; Renato Cerqueira; Marco Ferraz; Sérgio Cersosimo , "Multi-scale Evaluation of Texture Features for Salt Dome Detection", *2016 IEEE International Symposium on Multi-media (ISM)*, 2016.
- [10] Monika Jhuria; Ashwani Kumar; Rushikesh Borse , "Image processing for smart farming: Detection of disease and fruit grading", *2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)*, 2013.

- [11] Mohammad Reza Keyvanpour, Shokofeh Vahidiansadegh, Zahra Mirzakhani , "An analytical review of texture feature extraction approaches", *International Journal of Computer Applications in Technology* , 2021.
- [12] T. Ojala, M. Pietikäinen, and D. Harwood , "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions", *Proceedings of the 12th IAPR International Conference on Pattern Recognition* , 1994.
- [13] Larry S. Davis , "Polarograms: A new tool for image texture analysis", *Pattern Recognition Volume 13, Issue 3, pages 219-223*, 1981.
- [14] T. Ojala, M. Pietikainen, T. Maenpaa , "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", *IEEE Transactions on Pattern Analysis and Machine Intelligence Volume 24, Issue 7* , 2002.
- [15] Di Huang, Caifeng Shan, Mohsen Ardabilian, Yunhong Wang, Liming Chen , "Local Binary Patterns and Its Application to Facial Image Analysis: A Survey", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) Volume 41, Issue 6* , 2011.
- [16] Yanjun Zhang, Mengnan Wang , "Real-Time Texture Extraction Based on the Improved Median Robust Extended Local Binary Pattern", *ICCP 2020: 2020 9th International Conference on Computing and Pattern Recognition* , 2020.
- [17] Rasheed, A.H., "FPGA-based optimized systolic design for median filtering algorithms", *International Journal of Applied Engineering Research*, 2017.
- [18] P. Kolte, R. Smith, W. Su, "A fast median filter using AltiVec", *Proceedings 1999 IEEE International Conference on Computer Design: VLSI in Computers and Processors (Cat. No.99CB37040)*, 1999.
- [19] Vineet Kumar1, Abhijit Asati1, Anu Gupta1, "Low-latency median filter core for hardware implementation of 5×5 median filtering", 2017, *IET Image Processing*, 2017.
- [20] T. Ojala; T. Maenpaa; M. Pietikainen; J. Viertola; J. Kyllonen; S. Huovinen, "Outex - new framework for empirical evaluation of texture analysis algorithms", 2002 *International Conference on Pattern Recognition*, 2002