BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH

LUẬN VĂN THẠC SĨ
PHẠM THỊ TRINH

THỰC HIỆN LỌC NHIỄU ĐỐM CHO ẢNH SIÊU ÂM TRÊN FPGA

NGÀNH: KỸ THUẬT ĐIỆN TỬ - 605270

SKC004178

Tp. Hồ Chí Minh, tháng 04 năm 2014

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH

LUẬN VĂN THẠC SĨ

PHẠM THỊ TRINH

THỰC HIỆN LỌC NHIỄU ĐỐM CHO ẢNH SIÊU ÂM TRÊN FPGA

NGÀNH: KỸ THUẬT ĐIỆN TỬ - 605270

Hướng dẫn khoa học: TS. LÊ CHÍ THÔNG

Tp. Hồ Chí Minh, tháng 4 năm 2014

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của tôi.

Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tp. Hồ Chí Minh, ngày 19 tháng 4 năm 2014

Phạm Thị Trinh

LỜI CẢM ƠN

Sau một thời gian học tập và nghiên cứu tại trường, nay học viên đã hoàn thành luận văn tốt nghiệp cao học của mình. Để có được thành quả này, học viên đã nhận được rất nhiều sự hỗ trợ và giúp đỡ tận tình từ thầy cô, gia đình, cơ quan và bạn bè.

Học viên xin trân trọng bày tỏ lòng biết ơn sâu sắc đến Tiến sĩ Lê Chí Thông, người đã tận tình và trực tiếp hướng dẫn học viên thực hiện luận văn này.

Xin chân thành cảm ơn đến tất cả quý Thầy, Cô Trường Đại Học Sư Phạm Kỹ Thuật Tp. Hồ Chí Minh đã trang bị cho học viên những kiến thức rất bổ ích, quý báu để học viên tiến hành nghiên cứu luận văn này. Đặc biệt, xin chân thành cảm ơn quí Thầy, Cô Trường Trung Cấp Kinh Tế Kỹ Thuật Long An đã tạo điều kiện thuận lợi và hỗ trợ cho học viên rất nhiều trong quá trình học tập cũng như trong thời gian làm luận văn.

Và xin gửi lời cảm ơn đến đồng nghiệp, gia đình, bạn bè đã giúp đỡ cho học viên rất nhiều, đã tạo cho học viên niềm tin và nỗ lực cố gắng để hoàn thành luận văn.

Xin chân thành cảm ơn!

TÓM TẮT LUẬN VĂN

Trong luận văn này tác giả đã tiến hành nghiên cứu các kỹ thuật lọc nhiễu, đặc biệt là nhiễu đốm trên ảnh siêu âm. Tác giả đã thực hiện lọc nhiễu đốm, nhiễu muối tiêu sử dụng Matlab cho các ảnh thông thường và ảnh siêu âm nhằm so sánh các kết quả để cuối cùng chọn ra bộ lọc tốt nhất để lọc ảnh siêu âm bị nhiễu đốm.

Đặc biệt là tác giả đã thực hiện lọc nhiễu đốm cho ảnh siêu âm trên FPGA với hai bộ lọc Median và Modified hybrid median cùng hiển thị trên một màn hình giúp cho người xem dễ dàng phân biệt được ưu, nhược điểm của từng bộ lọc.

Các kết quả thực hiện cụ thể cũng được tác giả trình bày chi tiết trong luận văn này.

ABSTRACT

In this paper the authors have studied the interference filtering techniques, especially the spots on the ultrasound image. The author made a noise spots, salt and pepper noise to the image using Matlab and conventional ultrasound imaging to compare the final results to choose the best filter to filter noise in ultrasound spots.

Especially the authors have made a noise on the ultrasound image spots with two FPGA and Modified hybrid median filter median and displayed on a screen to help viewers easily distinguish the advantages and disadvantages of each filter .

The results of the specific author and are presented in detail in this thesis.

MỤC LỤC

Trang

TRANG TỰA
QUYẾT ĐỊNH GIAO ĐỀ TÀI
LÝ LỊCH KHOA HỌC
LỜI CAM ĐOAN i
LỜI CẢM ƠN ii
TÓM TẮT LUẬN VĂN iii
MỤC LỤC v
DANH SÁCH CHỮ VIẾT TẮT vii
DANH SÁCH CÁC HÌNH viii
DANH SÁCH CÁC BẢNG xi
Chương 1: TỔNG QUAN 1
1.1 Tổng quan chung về lĩnh vực nghiên cứu, các kết quả nghiên cứu trong và ngoài nước đã công bố 1
1.2 Mục đích của đề tài 10
1.3 Nhiệm vụ và giới hạn đề tài 10
1.4 Phương pháp nghiên cứu 10
Chương 2: CƠ SỞ LÝ THUYẾT 12
2.1 Công nghệ FPGA 12
2.1.1. Giới thiệu FPGA 12
2.1.2 Cyclone II 2C35 FPGA 13
2.2. Tổng quan về Verilog 29
2.3 Giới thiệu kit DE2 30
2.4 Khái quát về các loại nhiễu thường gặp trong ảnh siêu âm 32
Chương 3: THỰC HIỆN LỌC NHIỄU ĐỐM SỬ DỤNG MATLAB 34
3.1 Giới thiệu về Matlab và các phương pháp lọc nhiễu sử dụng Matlab 34
3.1.1 Dữ liệu 35
3.1.2 Ứng dụng 35
3.1.3 Hệ thống giao diện 36
3.1.4 Hộp công cụ xử lý ảnh 37
3.1.5 Phương pháp lọc nhiễu 42
3.2 Thiết kế và lọc nhiễu đốm sử dụng Matlab 45
3.2.1 Yêu cầu 45
3.2.2 Lưu đồ giải thuật lọc ảnh 46
3.3 Kết quả và thảo luận 48
Chương 4: THỰC HIỆN LỌC NHIỄU ĐỐM CHO ẢNH SIÊU ÂM TRÊN FPGA 53
4.1 Yêu cầu 53
4.2 Sơ đồ thiết kế 54
4.2.1 Nhiệm vụ khối PixelCount 55
4.2.2 Nhiệm vụ khối Load Data 56
4.2.3 Nhiệm vụ khối Filter 57
4.2.4 Nhiệm vụ khối Data_Collector 59
4.2.5 Nhiệm vụ khối Display_Select 59
4.2.6 Nhiệm vụ khối VGA_Ctr 59
4.2.7 Nhiệm vụ khối Display_Ctr 59

4.3 Giao tiếp với VGA và SRAM trên kit DE2 59
4.3.1 Chức năng và cách thiết kế khối VGA 59
4.3.2 Sơ đồ tổng hợp khối VGA 61
4.3.3 Nhiệm vụ của SRAM 62
4.3.4. Lưu đồ giải thuật đọc và ghi SRAM 65
4.4 Kết quả và thảo luận 66
4.4.1 Truyền ảnh từ PC xuống kit DE2 66
4.4.2 Hiển thị ảnh gốc lên monitor thông qua cổng VGA của kit DE2 68
4.4.3 Thiết kế bộ lọc 68
4.4.4 Lọc ảnh và xuất lên màn hình thông qua cổng VGA của kit DE2 70
Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI 74
5.1 Kết quả 74
5.2 Hạn chế 75
5.3 Hướng phát triển 75
TÀI LIỆU THAM KHẢO 76

DANH SÁCH CHỮ VIẾT TẮT

FPGA	: Field programmable Gate Array
DE	: Developmentand and Education.
PLDs	: Programmable Logic Device
ASICs	: Application-specific Integrated Circuit
LAB	: Logic Array Block
PLL	: Phase-Locked Loop
CLK	: Clock
DPCLK	: Clock dual-purpose
DSP	: Digital signal processing
VGA	: Video Graphics Array
DAC	: Digital-to-analog converter
CRT	: Cathode ray tube
RGB	: Red Green Blue
RAM	: Random Access Memory
SRAM	: Static random-access memory
SDRAM	: Synchronous dynamic random access
PC	: Personal Computer
SNR	: Signal-to-noise ratio
RMSE	: Root-mean-square error
PSNR	: Peak signal-to-noise ratio
QI	: Quality index
SSI	: Similar structure index
EPF	: Ability to preserve edges
MSE	: Mean square error
LUT	: Look up table

DANH SÁCH CÁC HÌNH

Hình 1.1: Ảnh siêu âm tim lấy từ các vị trí khác nhau của tim bệnh nhân khác nhau.. ..... **Error! Bookmark not defined.**

Hình 1.2: Nhiễu đốm trong hình ảnh được lọc bởi SRAD 4

Hình 1.3: Ảnh được lọc với bộ lọc Hybrid đã loại bỏ được cả nhiễu đốm và nhiễu xung. 5

Hình 2.1: Cấu trúc của chip Cyclone II 13

Hình 2.2: Sơ đồ cấu trúc của LE trong Cyclone II 15

Hình 2.3: Cấu trúc LAB trong Cyclone II 17

Hình 2.4: Minh họa cách kết nối trực tiếp 17

Hình 2.5: Mô tả các tín hiệu khối LAB 18

Hình 2.6: Mô tả kết nối nhiều LAB 19

Hình 2.7: Sơ đồ liên kết chuỗi thanh ghi 20

Hình 2.8: Sơ đồ liên kết C4 21

Hình 2.9: Sơ đồ khối điều khiển clock 22

Hình 2.10: Bộ phân kênh xung clock 23

Hình 2.11: Các vùng clock I/O và LAB 23

Hình 2.12: Sơ đồ khối của PLL 24

Hình 2.13 : Mô tả kết nối giữa LAB với M4K 25

Hình 2.14 : Cấu trúc khối nhân 25

Hình 2.15 : Giao tiếp giữa khối nhân với LAB 26

Hình 2.16 : Cấu trúc IOE 26

Hình 2.17 : Sơ đồ kết nối khối IO hàng vào mạng lưới liên kết 27

Hình 2.18 : Sơ đồ kết nối khối IO cột vào mạng lưới liên kết 27

Hình 2.19 : Đường tín hiệu đi qua khối IO 28

Hình 2.20 : Lựa chọn tín hiệu điều khiển cho IOE 28

Hình 2.21 : Cấu hình 2 chiều của IOE 29

Hình 2.22: Kit DE2 30
Hình 2.23: Các bộ phận kết nối kit DE2 31
Hình 2.24: Ảnh nhiễu muối tiêu 33
Hình 2.25: Ảnh nhiễu đốm 33
Hình 3.1: Các kiểu dữ liệu trong Matlab 35
Hình 3.2: Giao diện của Matlab 36
Hình 3.3: Minh họa ảnh nhị phân 37
Hình 3.4: Minh họa ảnh indexed 38
Hình 3.5: Minh họa ảnh mức xám 38
Hình 3.6: Minh họa ảnh RGB 39
Hình 3.7: Tọa độ điểm ảnh 40
Hình 3.8 : Ma trận bộ lọc Modifed hybrid median 43
Hình 3.9: Lưu đồ lọc ảnh sử dụng Matlab 50
Hình 3.10: Lọc ảnh thông thường và ảnh siêu âm nhiễu đốm thật dùng Matlab 51
Hình 3.11: Lọc ảnh thông thường và ảnh siêu âm nhiễu đốm giả dùng Matlab 51
Hình 3.12: Dùng Matlab lọc nhiễu đốm cho ảnh siêu âm 52
Hình 3.13: Dùng matlab lọc nhiễu muối tiêu cho ảnh thông thường (ảnh nhiễu giả và ảnh nhiễu thật) 50
Hình 3.14: Dùng Matlab lọc nhiễu đốm cho ảnh ảnh thông thường (ảnh nhiễu giả và ảnh nhiễu thật) 48
Hình 3.15: Dùng Matlab lọc nhiễu muối tiêu cho ảnh ảnh siêu âm (ảnh nhiễu giả và ảnh nhiễu thật) 48
Hình 3.16: Dùng Matlab lọc nhiễu đốm cho ảnh ảnh siêu âm (ảnh nhiễu giả và ảnh nhiễu thật) 48
Hình 4.1: Sơ đồ khối Image Process 54
Hình 4.2: Sơ đồ khối Control 55
Hình 4.3: Lưu đồ giải thuật khối PixelCount 56
Hình 4.4: Khối Filter 58

Hình 4.5: Khối VGA_control và các port giao tiếp.....61
Hình 4.6: Sơ đồ khối VGA được tổng hợp bởi Quartus II 62
Hình 4.7: Các vùng trong bộ nhớ của SRAM 63
Hình 4.8: Dạng sóng giao tiếp với SRAM.....64
Hình 4.9: Đọc SRAM 65
Hình 4.10: Ghi SRAM 66
Hình 4.11: Dữ liệu ảnh gốc 67
Hình 4.12: Dữ liệu lấy từ SRAM 67
Hình 4.13: Ảnh gốc được xuất ra từ cổng VGA của kit DE2 lên màn hình CRT 68
Hình 4.14: Dạng sóng mô tả bộ lọc Modified hybrid median 70
Hình 4.15: Ảnh gốc (a) và ảnh sau lọc bằng bộ lọc Median (b); các mũi tên chỉ các điểm nhiễu đốm.....71
Hình 4.16: Ảnh gốc (a) và ảnh sau lọc bằng bộ lọc bằng Median (b), bộ lọc Modified hybrid median hình "chữ X" (c) và bộ lọc Modified hybrid median hình "chữ +" (d); các mũi tên chỉ các điểm nhiễu đốm.....73

DANH SÁCH CÁC BẢNG

Bảng 1.1: Thống kế thông số các tiêu chuẩn như PSNR, MSE, RMSE của bộ lọc đề xuất và phương pháp SRAD đối với hình ảnh siêu âm tim. 6
Bảng 1.2: Chất lượng bộ lọc đánh giá thông qua các thông số PSNR, IQI, ENL, MSSIM và thời gian thực hiện yêu cầu 7
Bảng 3.1 : Kết quả so sánh hai bộ lọc Median và Modified hybrid median với $\sigma^2 = 0,2$. 45
Bảng 3.2 : Kết quả so sánh hai bộ lọc Median và Modified hybrid median cho ảnh dùng trong đề tài. 48

Chương 1
TỔNG QUAN

1.1 Tổng quan chung về lĩnh vực nghiên cứu, các kết quả nghiên cứu trong và ngoài nước đã công bố

Trong những năm gần đây, siêu âm đang được sử dụng rộng rãi để chẩn đoán các cơ quan khác nhau trong cơ thể người như: tim, thận, tuyến tiền liệt, gan, buồng trứng, tử cung, tuyến giáp, v.v đặc biệt là siêu âm thai nhi, phương pháp siêu âm đã mang lại những kết quả nhất định trong lĩnh vực y khoa. Tuy nhiên, một trong những hạn chế của nó là sự tương phản thấp, hình ảnh nhiễu cao ảnh hưởng đến kết quả là điều không thể tránh khỏi. Nhiễu thường gặp trong ảnh siêu âm là nhiễu đốm, nhiễu đốm đã che lấp một phần hoặc toàn phần các chi tiết trong một hình ảnh hoặc có thể gây chuẩn đoán nhầm và dẫn đến kết luận sai. Do đó lọc nhiễu đốm là một điều kiện tiên quyết trong siêu âm, với điều kiện là các tính năng cho chẩn đoán tương đối chuẩn xác.

Hiện nay xử lý ảnh là một kỹ thuật đang được ứng dụng rất phổ biến, bằng nhiều kỹ thuật khác nhau. Với sự tiến bộ không ngừng của nền khoa học kỹ thuật trong và ngoài nước, lĩnh vực xử lý ảnh ngoài việc tạo ra một bức ảnh đẹp, sắc nét, đáp ứng thẩm mỹ của con người, mà còn có ý nghĩa trong lĩnh vực y khoa vì xử lý ảnh có thể tạo ra những hình ảnh trung thực, rõ nét, giúp đội ngũ y bác sĩ có thể chẩn đoán chính xác cho bệnh nhân trong siêu âm, giảm sai sót đáng tiếc có thể xảy ra.

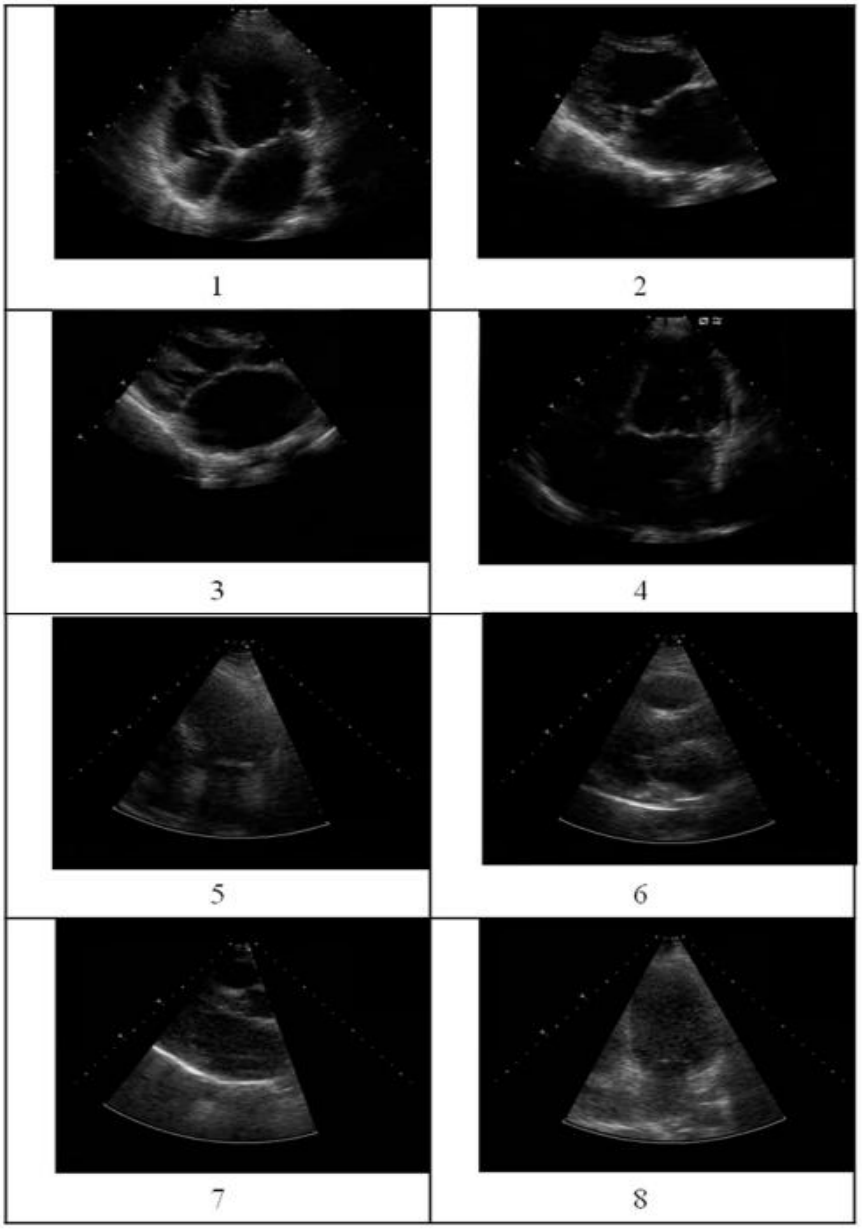
Đề tài về: "Lọc nhiễu đốm trong ảnh siêu âm" đã được nhiều thế hệ, nhiều tác giả trong và ngoài nước nghiên cứu với nhiều mức độ và cho ra các kết quả khác nhau, tuy nhiên cũng còn có nhiều vấn đề cần cải tiến trong các phương pháp lọc nhiễu đốm trên ảnh siêu âm như: giảm nhiễu, khả năng bảo quản cạnh, …, và đó cũng là lý do cho tác giả chọn đề tài " Thực hiện lọc nhiễu

đốm cho ảnh siêu âm trên FPGA", với mong muốn tạo ra mong kết quả tốt cho ảnh siêu âm, có thể so sánh nhiều bộ lọc với nhau ngay trên cùng thời điểm nhằm chọn kết quả tốt nhất, giảm sai sót trong kết quả siêu âm một cách thấp nhất. Từ đó, có thể mở ra một hướng mới cho phương pháp lọc ảnh trong siêu âm.

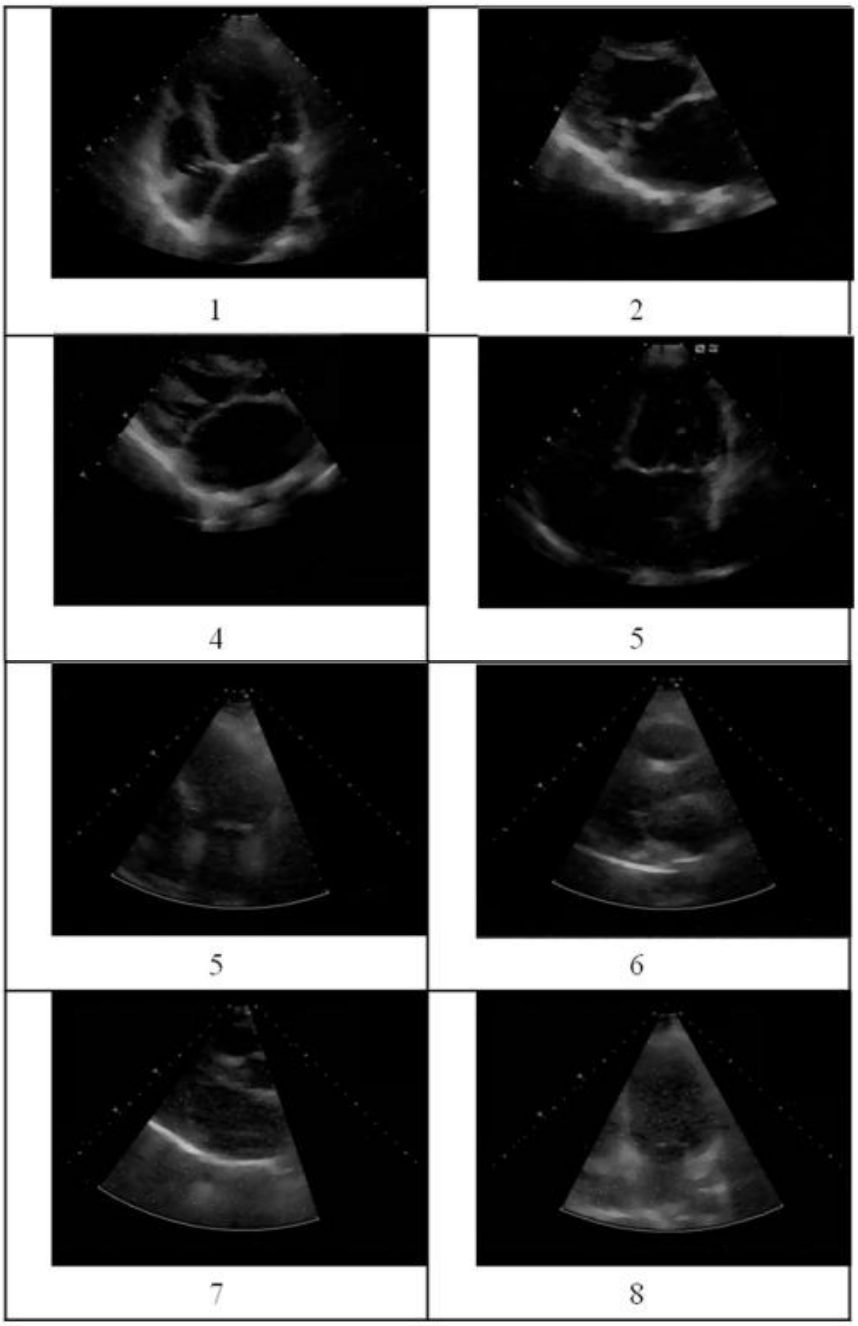
Một số kết quả nghiên cứu trong và ngoài nước đã công bố:

- Một thay đổi bộ lọc Hybrid cho việc loại bỏ nhiễu trong ảnh siêu âm tim – A Modified hybrid Filter for Echocardiographic Image [16]

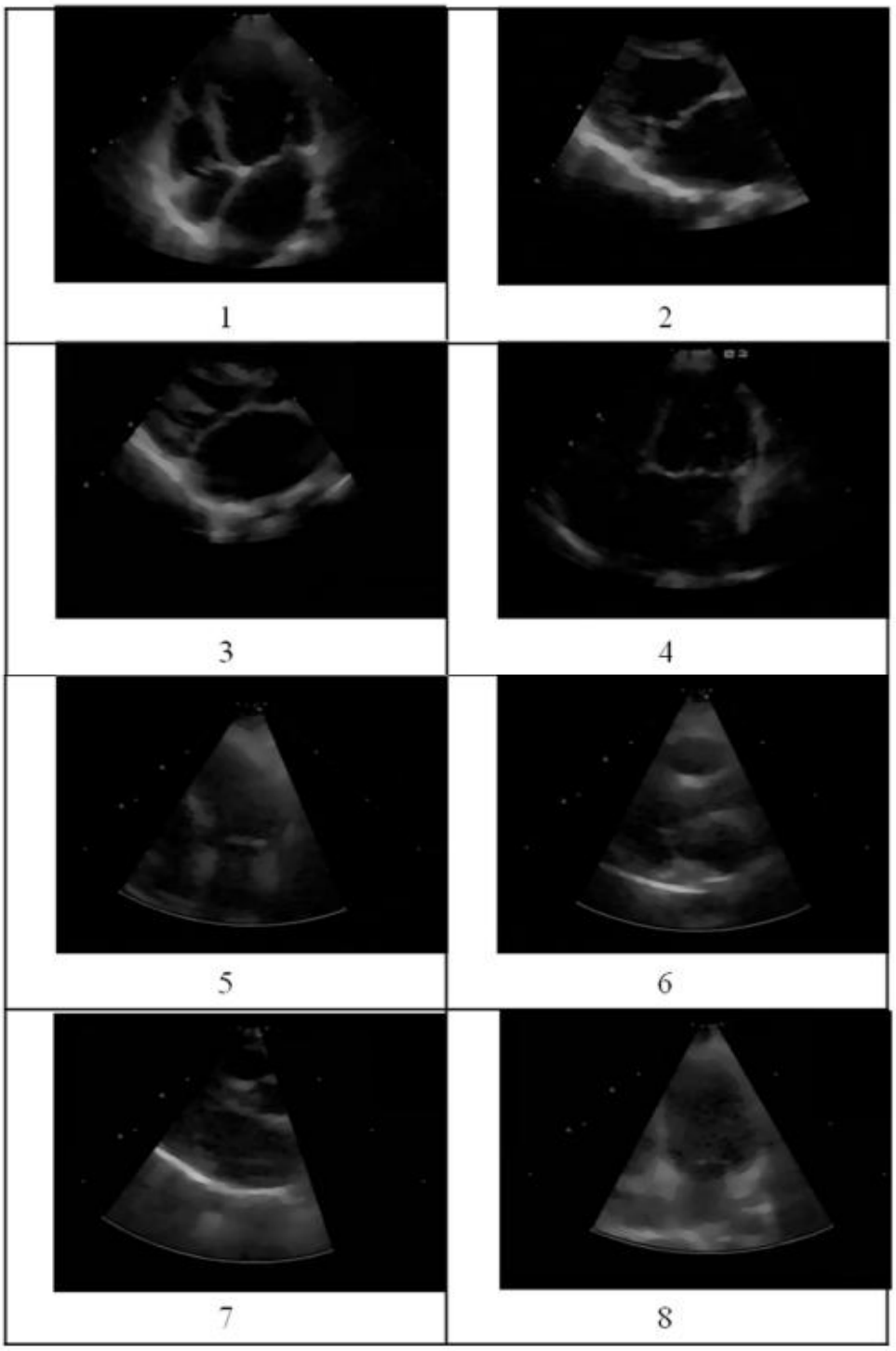
Trong tháng 6 năm 2012, các tác giả giới thiệu bộ lọc Hybrid**.** Kết quả cho thấy hình ảnh bị nhiễu đốm đã giảm đáng kể. Ngoài ra bộ lọc còn loại bỏ nhiễu đốm xung quanh nhằm tăng cường các cạnh trong hình ảnh. Ở đây, các tác giả sử dụng thuật toán lọc nhiễu đốm SRAD kết hợp với bộ lọc trung vị. SRAD không chỉ bảo tồn các cạnh mà còn tăng cường cạnh bằng cách khuếch tán trên cạnh và cho phép tăng cường ở hai bên cạnh. SRAD là thích nghi và không sử dụng ngưỡng khó để thay đổi thực hiện ở các vùng đồng nhất hoặc trong khu vực gần cạnh và tính năng nhỏ. Kỹ thuật khuếch tán mới được dựa trên cùng một mức tối thiểu có nghĩa là bình phương lỗi vuông (MMSE) phương pháp tiếp cận các bộ lọc như Lee (Kuan) và Frost. Trong thực tế, nó đã được chỉ ra rằng các SRAD có thể liên quan trực tiếp đến Lee và Frost cửa sổ dựa trên bộ lọc. Vì vậy, SRAD là phần mở rộng linh hoạt cho lọc nhiễu đốm và tăng cường cạnh theo cách tương tự mà Perona và Malik khuếch tán khác hướng ban đầu là phần mở rộng cạnh linh hoạt của bộ lọc trung vị. Bộ lọc trung vị đơn có thể được sử dụng kết hợp với SRAD để loại bỏ nhiễu tăng vọt (Theo Kalpana Saini, *A Modified hybrid Filter for Echocardiographic Image*. June 2012. Vol.5, No. 2)



Hình 1.1: Cho thấy những hình ảnh siêu âm tim lấy từ các vị trí khác nhau của tim bệnh nhân khác nhau. Ở đây chúng ta có thể thấy rằng có rất nhiều nhiễu đốm xuất hiện trong những hình ảnh và làm khó khăn để trích xuất biên cũng như các chẩn đoán khác [16].



Hình 1.2: Cho thấy những đốm trong hình ảnh được lọc bởi SRAD. Nhưng nhiễu xung, nhiễu muối tiêu lại xuất hiện, cũng tạo ra vấn đề trong chẩn đoán. Vì vậy, cần đề xuất một bộ lọc – đó là bộ lọc Hybrid để loại nhiễu đốm cũng như nhiễu xung [16]



Hình 1.3: Cho thấy hình ảnh được lọc với bộ lọc Hybrid được đề xuất đã loại bỏ được cả hai loại nhiễu đốm và nhiễu xung [16]

Bảng 1.1: Phân tích thống kê của hai bộ lọc này. Thông số thống kê cho thấy PSNR cao, MSE và RMSE có giá trị thấp cho thấy bộ lọc đề xuất tốt hơn. Phân tích trong bảng 1 còn cho thấy rằng các bộ lọc được đề xuất lọc nhiễu đốm tốt hơn đối với hình ảnh siêu âm tim của bệnh nhân [16].

Hình	**Phương pháp SRAD**			**Phương pháp đề xuất**		
	PSNR(dB)	**MSE**	**RMSE**	**PSNR(dB)**	**MSE**	**RMSE**
1	+37.83	10.7070	3.2722	+38.64	8.8847	2.9807
2	+39.94	6.5881	2.5667	+40.70	5.5337	2.3524
3	+39.56	7.2038	2.6840	+40.57	5.7077	2.3891
4	+41.74	4.3533	2.0865	+43.27	3.0653	1.7508
5	+40.11	6.3468	2.5193	+40.90	5.2892	2.2998
6	+38.20	9.8513	3.1387	+39.55	7.2064	2.6845
7	+38.45	9.2970	3.0491	+39.43	7.4220	2.7243
8	+37.16	12.5072	3.5366	+37.82	10.7328	3.2761

Theo [16] có thể thấy rằng các dòng dữ liệu liên quan đến SRAD bộ lọc trên các dòng dữ liệu liên quan đến phương pháp được đề xuất trong trường hợp PSNR. Nhưng trong trường hợp MSE, RMSE các dữ liệu liên quan đến SRAD là dưới các dòng dữ liệu liên quan đến phương pháp đề xuất. Phương pháp được đề xuất với bộ lọc Hybid là tốt hơn thuật toán lọc SRAD vì nhiễu đốm giảm đáng kể, các hình ảnh đáp ứng được các chỉ số PSNR, MSE, RMSE.

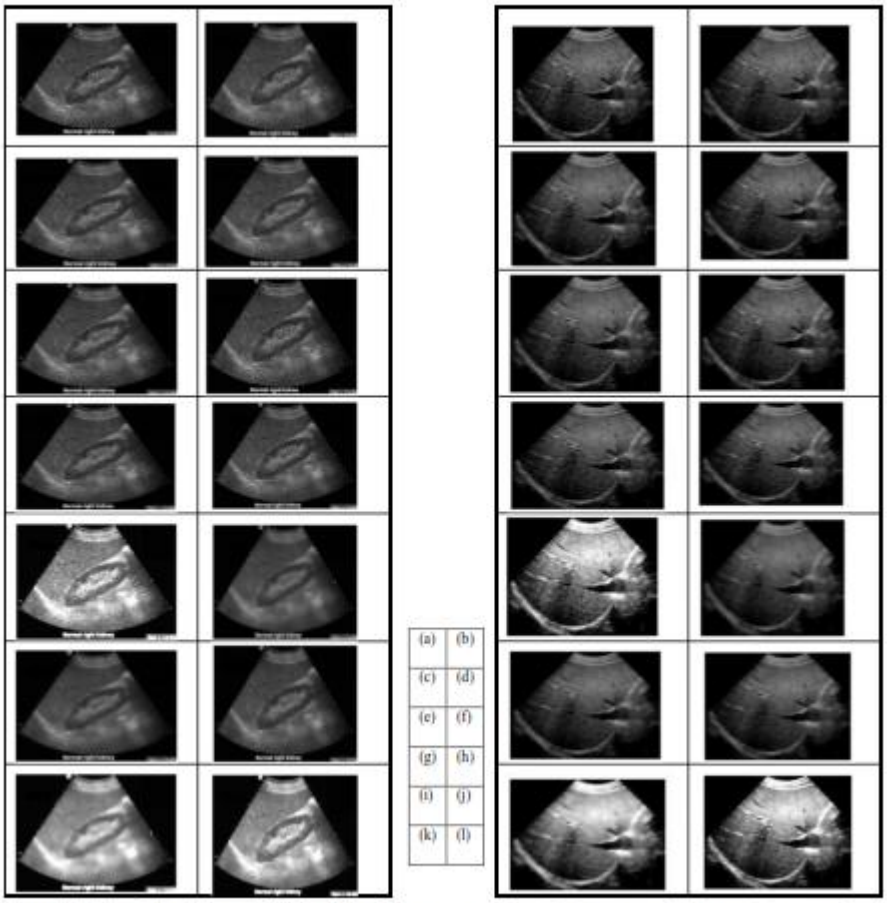
- Một cách tiếp cận mới để giảm nhiễu đốm và tăng cường các hình ảnh siêu âm - A Novel Approach for Speckle Reduction andEnhancement of Ultrasound Images [17]

Tháng 5 năm 2012 bài báo của Shruthi G Department, trình bày một phương pháp lọc Hybrid và bộ lọc đa tầng để giảm nhiễu đốm và cải thiện chất lượng hình ảnh để chẩn đoán tốt hơn. Việc thực hiện các phương pháp tiếp cận của các tác giả được so sánh với các bộ lọc giảm đốm khác trên cơ sở các thông số chất lượng hình ảnh như: Tỉ số tín hiệu trên đỉnh nhiễu (PSNR), hiệu quả hình

ảnh (ENL), chỉ số chất lượng hình ảnh (IQI) và cấu trúc trung bình tương đồng trong bản đồ (MSSIM) (Theo Shruthi G Department of ECE, *A Novel Approach for Speckle Reduction andEnhancement of Ultrasound Images*. May 2012. Vol. 45, No. 20)

Bảng 1.2: Cho thấy chất lượng bộ lọc được đánh giá thông qua các thông số PSNR, IQI, ENL, MSSIM và thời gian thực hiện yêu cầu [17].

Parametter	**PSNR**		**MSSIM**		**ENL**		**IQI**		**Exccution Time in seconds**
	Kidney	**Liver**	**Kidney**	**Liver**	**Kidney**	**Liver**	**Kidney**	**Liver**	**Kidney/Liver**
Mean Filter	28.471	34.671	0.9112	0.8525	0.0697	0.562	0.8614	0.7922	2.866
Lee Filter	35.996	34.662	0.8775	0.8415	0.0398	0.0352	0.836	0.7919	10.437
Forst Filter	27.041	26.018	0.8228	0.8162	0.0376	0.0338	0.8019	0.7703	6.326
Kuan Filter	35.991	34.527	0.8516	0.8249	0.0397	0.0336	0.8256	0.7902	6.572
SAWM	43.741	41.915	0.9850	0.9658	0.0481	0.0481	0.9304	0.9601	17.107
Gaussian smoothing	62.424	59.352	0.6474	0.6025	0.613	0.0641	0.6039	0.6397	0.8405
Bilateral Filter	74.558	79.207	0.9919	1.0000	0.097	0.0966	0.7512	0.7149	0.2364
Median Filter	33.567	35.118	0.9358	0.8419	0.0697	0.065	0.8855	0.8032	1.9905
Adaptive Median Filter	42.440	40.667	0.9767	0.9516	0.0697	0.0652	0.948	0.9379	12.763
SRAD	43.669	33.467	0.8743	0.7328	0.0665	0.0651	0.7271	0.6922	1.1036
Weiner Filter	33.227	33.655	0.7149	0.6950	0.0651	0.0652	0.607	0.5933	1.9267
Hybrid Filter	63.423	61.948	0.8077	0.8141	0.0689	0.0676	0.7407	0.7534	1.9234
Multistage Filter	**79.915**	**92.740**	**0.9945**	**0.9843**	**0.0984**	**0.0789**	**0.9497**	**0.9695**	**17.302**



Theo [17] ta thấy rằng bộ lọc Hybrid lọc mịn hơn (PSNR = 63,423), ít biến dạng (IQI = 0,7407), nhiễu đốm còn ít hơn (ENL = 0,0689) và tương phản tốt hơn (MSSIM = 0,8077) so với kỹ thuật lọc dùng tiêu chuẩn SRAD và Gaussian. Các bộ lọc đa tầng có PSNR (79,915), IQI (0,9497), MSSIM (0,9945) và ENL (0,0984). Kết quả trên chỉ ra rằng bộ lọc đa tầng cũng lọc mịn hơn (giá trị PSNR cao), biến dạng ít hơn (IQI gần để thống nhất), đốm thấp hơn (giá trị ENL là cao) và độ tương phản tốt (giá trị MSSIM là cao) [17]

Từ các cuộc thảo luận ở trên, nó được quan sát thấy rằng Hybrid và đa tầng lọc đáp ứng tất cả các yêu cầu được liệt kê cho một bộ lọc nhiễu đốm. Việc thực hiện với bộ lọc đa tầng là tốt hơn vì nó có giá trị cao hơn của PSNR, IQI, ENL và MSSIM so với bộ lọc Hybrid. Phân tích ta thấy các bộ lọc đa tầng lọc còn ít nhiễu đốm, bảo quản tốt các cạnh và cải thiện độ tương phản hơn so với

bộ lọc Hybrid. Từ bảng 1.2, quan sát thấy rằng thời gian thực hiện của bộ lọc Hybrid là thấp hơn so với bộ lọc đa tầng. Tuy nhiên, đa tầng lọc cho hiệu suất tốt hơn so với bộ lọc Hybrid được nhìn thấy cả hai trong phân tích định lượng và định tính. Phương pháp tiếp cận với bộ lọc đa tầng cho hiệu suất tốt hơn với giá trị cao hơn của PSNR (> 79), IQI (> 0,94), MSSIM (0,9945) và ENL (0,0984) so với bộ lọc Hybrid PSNR (62), IQI (0,81), MSSIM (0,8) và ENL (0,067). Phân tích định tính cho thấy rằng phương pháp lọc đa tầng làm giảm đáng kể nhiễu đốm, bảo tồn các cạnh và cải thiện độ tương phản nhưng có một thời gian thực hiện cao hơn (17,3 giây) so với bộ lọc Hybrid (1,92 giây) [17].

- Bộ lọc biên hiệu quả cho lọc nhiễu đốm trong ảnh siêu âm y tế - Modified Hybrid Median Filter for Effective Speckle Reduction in Ultrasound Images [18]

Trong bài báo này cho thấy khi các tác giả sử dụng các bộ lọc khác nhau trong ảnh siêu âm sẽ cho kết quả khác nhau, ở đây tác giả có so sánh các tiêu chuẩn giữa các bộ lọc khác nhau như: bộ lọc Lee, lọc Kuan, lọc Frost, lọc trung vị, lọc giả trung vị, …. Trong đó bộ lọc Modified Hybrid Median cho kết quả tốt nhất (Theo R.Vanithamani, *Modified Hybrid Median Filter for Effective Speckle Reduction in Ultrasound Images*)

Hầu hết các nghiên cứu trên đều cho thấy mục đích sau cùng của các tác giả trong các bài báo trên đều mong muốn chọn được bộ lọc hiệu quả cho ảnh siêu âm bị nhiễu đốm, các tác giả đã sử dụng rất nhiều bộ lọc khác nhau để lọc nhiễu đốm trên ảnh siêu âm như: bộ lọc Lee, lọc Kuan, lọc Frost, lọc trung vị, lọc giả trung vị. Tuy nhiên các kết quả thu được còn giới hạn, chưa phong phú, chỉ thực hiện trên ảnh siêu âm, thời gian xử lý còn chậm …Trong luận văn này tác giả muốn cải thiện tốt độ xử lý nhanh hơn; thực hiện lọc hai loại nhiễu khác nhau trên hai bộ lọc khác nhau: Median và Modified hybrid median dùng Matlab; đồng thời cũng thực hiện lọc nhiễu đốm cho ảnh siêu âm trên FPGA với hai bộ lọc Median và Modified hybrid median; từ đó so sánh các kết quả thực hiện được nhằm chọn ra bộ lọc tốt nhất khi lọc ảnh siêu âm bị nhiễu đốm, góp

phần giảm sai sót trong quá trình chẩn đoán kết quả của Bác sĩ.

1.2 Mục đích của đề tài

Tác giả dùng bộ lọc Median và Modify Hybrid Median để lọc nhiễu đốm cho ảnh siêu âm, trên cùng một màn hình hiển thị được ảnh gốc và ảnh sau khi lọc bằng từ hai bộ lọc khác nhau, nhằm loại bỏ nhiễu bằng mắt thường có thể nhìn thấy được và tạo tiền đề cho các cuộc khảo sát sau này.

1.3 Nhiệm vụ và giới hạn đề tài

Nhiệm vụ: Tác giả thực hiện lọc nhiễu đốm cho ảnh thông thường và ảnh siêu âm bằng Matlab với hai bộ lọc: Median và Modify Hybrid Median; Tiếp đó thực hiện lọc nhiễu đốm cho ảnh siêu âm trên FPGA.

Giới hạn đề tài:

- Nghiên cứu về nhiễu đốm, các phương pháp lọc nhiễu đốm.
- Tìm hiểu về cơ sở lý thuyết lọc ảnh, về Verilog, FPGA, Kit DE 2
- Thực hiện lọc nhiễu đốm cho ảnh thông thường và ảnh siêu âm dùng Matlab
- Thực hiện lọc ảnh nhiễu đốm trên FPGA.

1.4 Phương pháp nghiên cứu

Các phương pháp nghiên cứu được tác giả sử dụng trong luận văn bao gồm:

- Khảo sát, phân tích, tổng hợp
- Mô phỏng trên máy tính, thực nghiệm trên kit DE 2.
- Đánh giá dựa trên kết quả thu được.

Nội dung còn lại của luận văn:

Chương 2: Trong chương này tác giả nêu về công nghệ FPGA, tổng quan về Verilog, kit DE2, và các loại nhiễu trong ảnh siêu âm cũng như các phương pháp lọc nhiễu đốm thường gặp cho ảnh siêu âm.

Chương 3: Trong chương này tác giả nêu các phương pháp lọc nhiễu đốm, sử dụng phần mềm Matlab để lọc ảnh nhiễu đốm, nhiễu muối tiêu (để so sánh)

sử dụng mạch lọc Median và Modified Hybrid Median; đồng thời chọn ra mạch lọc tốt nhất để lọc nhiễu đốm cho ảnh siêu âm.

Chương 4: Trong chương này tác giả dùng phần Quartus II 12.0sp2 Web Edition (32 – bit) để lập trình ngôn ngữ verilog trên FPGA, giao tiếp SRAM, VGA với kit DE2.

Chương 5: Kết luận và hướng phát triển luận văn

Chương 2
CƠ SỞ LÝ THUYẾT

2.1 Công nghệ FPGA

2.1.1. Giới thiệu FPGA:

FPGAs (Field Programmable Gate Arrays) là những mạch tích hợp bao gồm các khối logic cấu hình lập trình được cùng với mãng liên kết giữa các khối. Phụ thuộc vào cái họ mong muốn thực hiện mà một vài FPGA chỉ lập trình một lần trong khi những FPGA khác có thể lập trình nhiều lần.

Field Programmable: có thể lập trình được ở dạng trường, khác với các thiết bị khác chứa phần cứng bị ràng buộc bởi nhà sản xuất. Ở đây FPGA được định hình trong phòng thí nghiệm hoặc nó hướng tới điều chỉnh chức năng của một thiết bị riêng trong hệ thống điện tử mà có thể triển khai rộng trên thế giới.

FPGA chiếm giữ một vị trí giữa PLDs và ASICs vì chứa các chức năng giống như PLDs và có hàng triệu cổng logic và thực hiện những chức năng vô cùng lớn và phức tạp.

Chi phí để thiết kế một FPGA thấp hơn nhiều so với ASIC (mặc dù các thành phần của ASIC thì rẻ hơn nhiều khi sản xuất lớn). Nhưng việc thay đổi thiết kế trong FPGA lại dễ dàng hơn và thời gian đưa ra thị trường nhanh hơn.

FPGA được làm từ rất nhiều cái nhỏ, điều này giúp các công ty thiết kế phổ biến và dễ dàng tồn tại trên thị trường. FPGA còn tạo điều kiện phát triển theo kiểu “Fred – in – the – shed”, điều này giúp các kỹ sư hoặc các nhóm nhỏ có thể thực thi phần cứng và phần mềm của mình trên một nền FPGA mà ít phải chịu chi phí kỹ thuật hoặc mua sắm những công cụ đắt tiền.

FPGA là công nghệ chứa hàng triệu cổng logic có sẵn, một vài thiết bị chứa lõi xử lý nhúng, cổng I/O tốc độ cao. Ngày nay FPGA được sử dụng trong rất nhiều lĩnh vực: viễn thông, rađa, xử lý tín hiệu số (DSP), xử lý ảnh, …

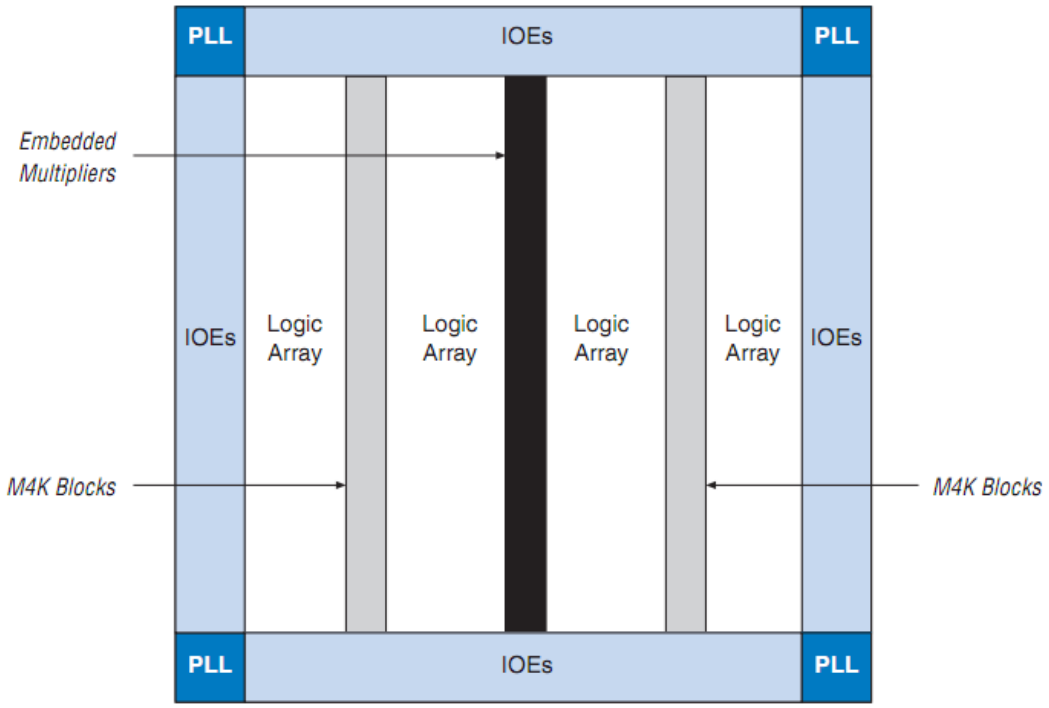
Tóm lại: công nghệ FPGA là một công nghệ tích hợp cho phép người dùng thiết kế mạch ở mức cổng logic hoặc mức thanh ghi, FPGA còn được gọi là thiết bị có khả năng tái cấu hình được ứng dụng rộng rãi cho việc thiết kế các IC chuyên dụng. Công nghệ FPGA cho phép thiết kế IC mà không cần dây chuyền công nghệ chế tạo IC từ những tấm bán dẫn.

FPGA được cấu tạo từ 3 thành phần:

- Khối logic
- Mạng kết nối trong có thể lập trình
- Khối I/O.

2.1.2 Cyclone II 2C35 FPGA [1]:

Dưới đây là cấu trúc của FPGA Cyclone II:



Hình 2.1: Cấu trúc của chip Cyclone II [5]

Cyclone II 2C35 FPGA có cấu trúc 2 chiều theo hàng và cột để thực hiện các chức năng logic

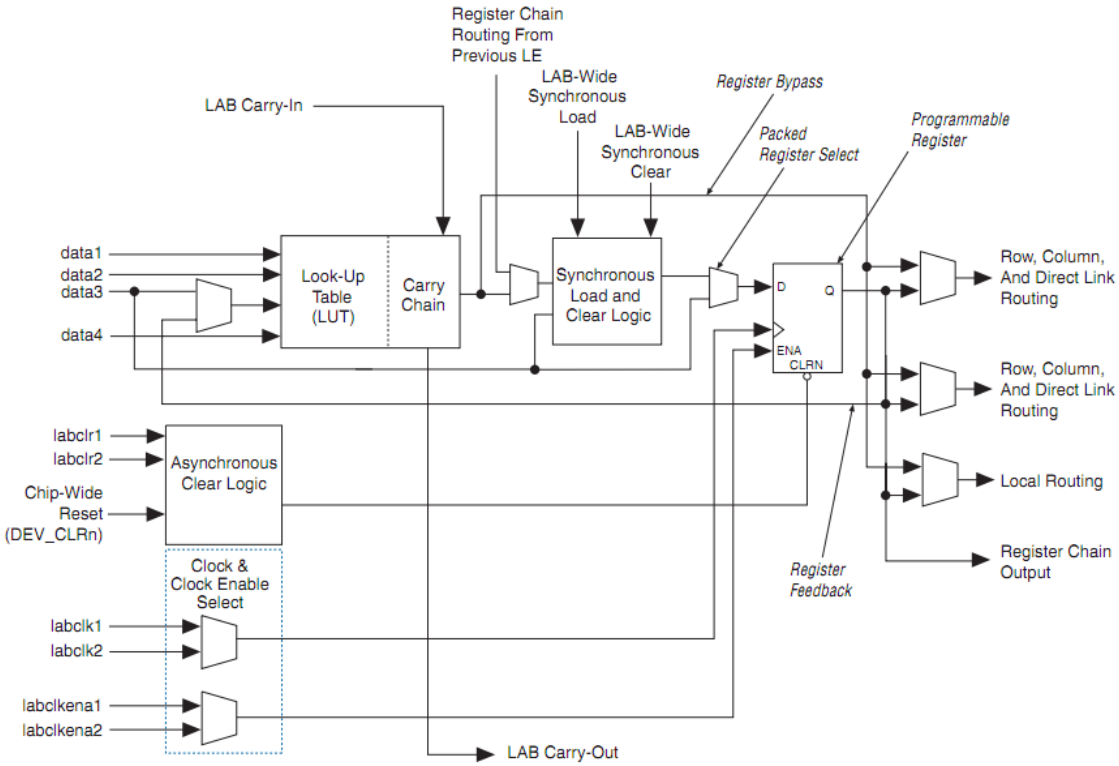
- 33216 Les (logic elements)
- 105 khối M4K RAM (tốc độ clock có thể lên đến 250 Mhz)
- 483,840 RAM bits
- 35 khối nhân nhúng
- 4PLLs
- 475 chân I/O
- FineLine BGA 672 - pin package

2.1.2.1 Các phần tử logic LE:

Các LE hoàn toàn tương thích và cung cấp những tính năng nâng cao cho việc thực hiện một cách hiệu quả các chức năng logic.

Mỗi LE gồm :

- Một bảng tra 4 ngõ vào (LUT) có khả năng thực hiện bất kỳ hàm 4 biến nào.
- Một thanh ghi lệnh lập trình được.
- Một kết nối chuỗi carry (carry chain connection).
- Một kết nối chuỗi thanh ghi (register chain connection).
- Khả năng lái tất cả các loại liên kết: cục bộ, hàng, cột, chuỗi thanh ghi và các kết nối trực tiếp.
- Hỗ trợ việc đóng gói thanh ghi (register packing).
- Hỗ trợ việc hồi tiếp thanh ghi (register feedback).



Hình 2.2: Sơ đồ cấu trúc của LE trong Cyclone II [5]

Mỗi thanh ghi lập trình được của LE có thể cấu hình thành các dạng D,T, JK, SR. Mỗi thanh ghi có các chân tín hiệu ngõ vào như data, clock, clock enable, clear. Với các hàm tổ hợp, ngõ ra của LUT sẽ bỏ qua các thanh ghi và điều khiển trực tiếp các ngõ ra LE.

Mỗi LE có 3 ngõ ra sẽ điều khiển các nguồn cục bộ, hàng và cột. Các thanh ghi cũng như các ngõ ra LUT có thể điều khiển mỗi ngõ một cách độc lập. Hai ngõ ra sẽ điều khiển hàng hay cột và các kết nối trực tiếp và ngõ còn lại sẽ điều khiển các nguồn kết nối cục bộ, cho phép LUT điều khiển một ngõ ra trong khi thanh ghi sẽ điều khiển ngõ ra còn lại. Tính năng này gọi là đóng gói thanh ghi giúp nâng cao khả năng thực hiện công việc bởi ta có thể dung các thanh ghi cũng như các LUT cho các hàm riêng biệt. Tính năng hồi tiếp thanh ghi cho phép ngõ ra thanh ghi hồi tiếp về LUT của cùng LE, giúp nâng cao hiệu quả làm việc của LE.

Ngoài 3 ngõ ra này, giữa các LE trong cùng một LAB có các ngõ ra chuỗi thanh ghi. Ngõ ra này cho phép các thanh ghi trong cùng LAB xếp tầng với

nhau. Nó cho phép một LAB sử dụng các LUT cho các hàm tổ hợp cũng như các thanh ghi cho các chức năng dịch thanh ghi riêng biệt. Tính năng này giúp tăng tốc độ kết nối giữa các LAB mà vẫn tiết kiệm được tài nguyên liên kết.

Các chế độ hoạt động của LE:

- Chế độ bình thường (nomal mode): Chế độ này phù hợp với các ứng dụng tổng quát và các hàm tổ hợp. Trong chế độ này, bốn ngõ vào dữ liệu của LAB là các ngõ vào LUT. Chế độ này cũng hỗ trợ đóng gói thanh ghi cũng như hồi tiếp thanh ghi.
- Chế độ số học: Chế độ này lý tưởng cho việc thực hiện các bộ so sánh, bộ đếm, bộ cộng và bộ cộng tích lũy. Một LE sẽ thực hiện một bộ cộng toàn phần 2 bit và chuỗi carry cơ bản. Các LE trong chế độ này có thể được dùng để điều khiển các ngõ ra LUT có gán thanh ghi hay không có gán thanh ghi. Đóng gói thanh ghi cũng như hồi tiếp thanh ghi cũng sẽ được hỗ trợ trong chế độ này.

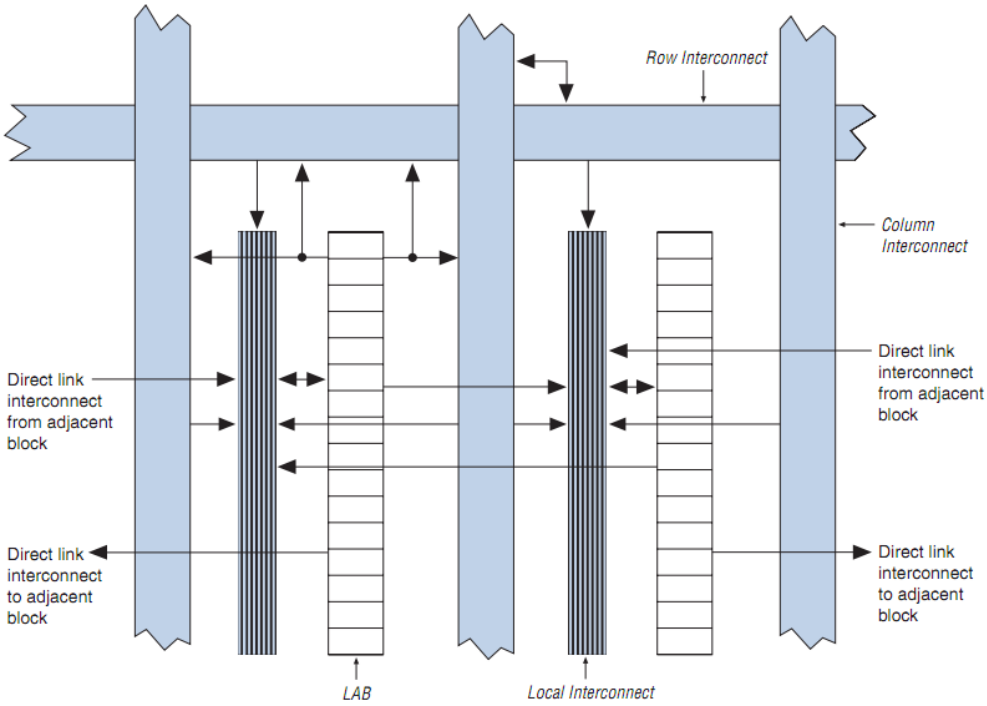
2.1.2.2 Các khối dãy logic (Logic Array Block – LAB)

a. Cấu trúc:

Mỗi LAB bao gồm:

- 16 LE
- Các tín hiệu điều khiển LAB
- Các chuỗi carry LE.
- Các chuỗi thanh ghi: truyền ngõ ra của một thanh ghi trong LE tới thanh ghi của LE kế tiếp trong cùng một LAB

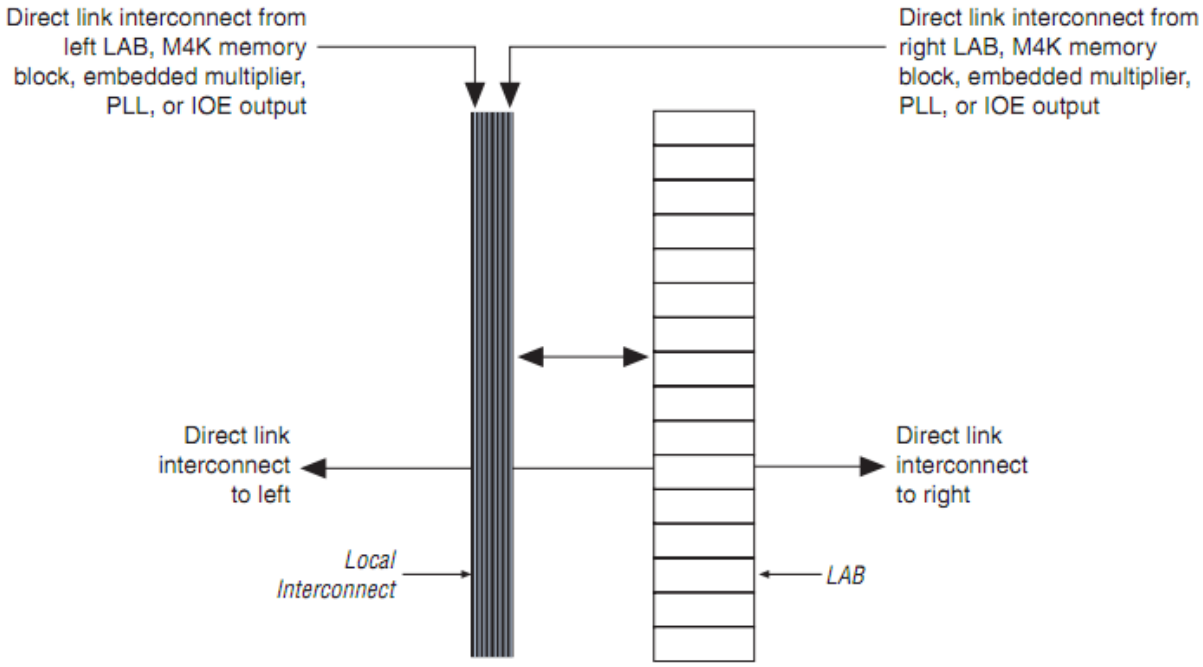
Kết nối cục bộ: kết nối cục bộ sẽ truyền tín hiệu giữa các LE trong cùng LAB.



Hình 2.3: Cấu trúc LAB trong Cyclone II [5]

b. Liên kết trong LAB:

Liên kết cục bộ LAB dùng để kết nối các LE trong cùng một LAB. Chúng được điều khiển bằng các liên kết hàng, liên kết cột, và các ngõ ra LE trong cùng một LAB.



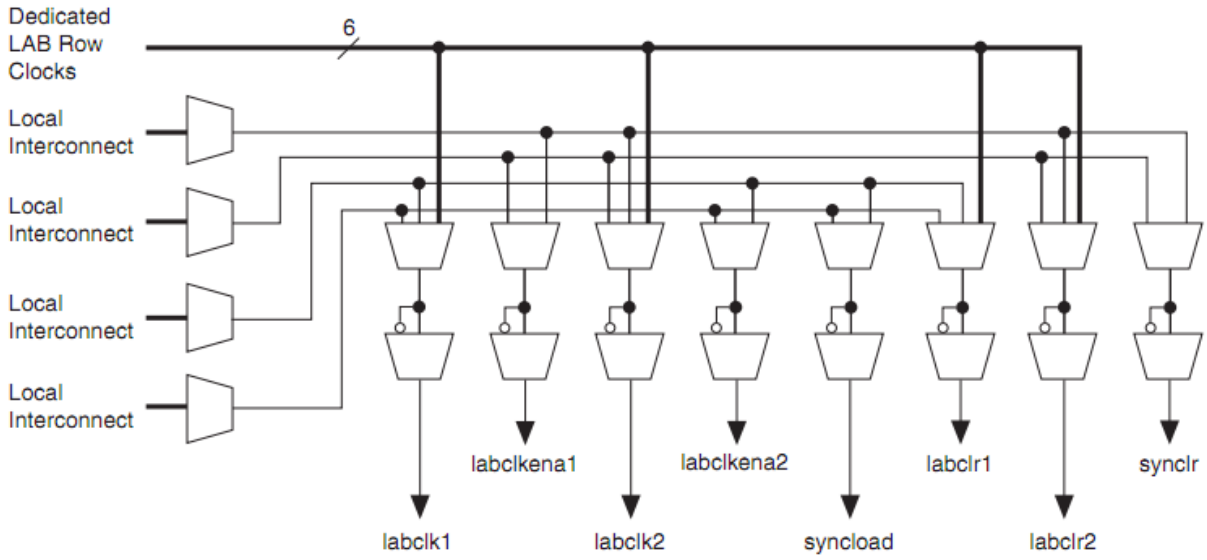
Hình 2.4: Minh họa cách kết nối trực tiếp [5]

Ngoài ra, các khối PLL, M4K, các bộ nhân nhúng cũng có thể điều khiển liên kết cục bộ của một LAB thông qua các kết nối trực tiếp. Kết nối trực tiếp giúp giảm số lượng liên kết hàng và cột.

c. Tín hiệu điều khiển:

Bao gồm:

- 2 clock và 2 clock enable: mỗi clock và clock enable được liên kết với nhau.
- 2 clear bất đồng bộ.
- 1 clear đồng bộ và 1 load đồng bộ: thực hiện việc đếm và các công việc khác, 2 tín hiệu này ảnh hưởng đến toàn bộ các thanh ghi trong LAB.



Hình 2.5: Mô tả các tín hiệu khối LAB [5]

2.1.2.3 Liên kết đa kênh:

Dùng để liên kết các LE, các khối M4K, bộ nhân nhúng và các chân IO.

Là các đường dẫn liên tục có tốc độ khác nhau dùng cho sự kết nối của các khối trong và ngoài, bao gồm các liên kết hàng và cột liên kết qua các khoảng cách cố định.

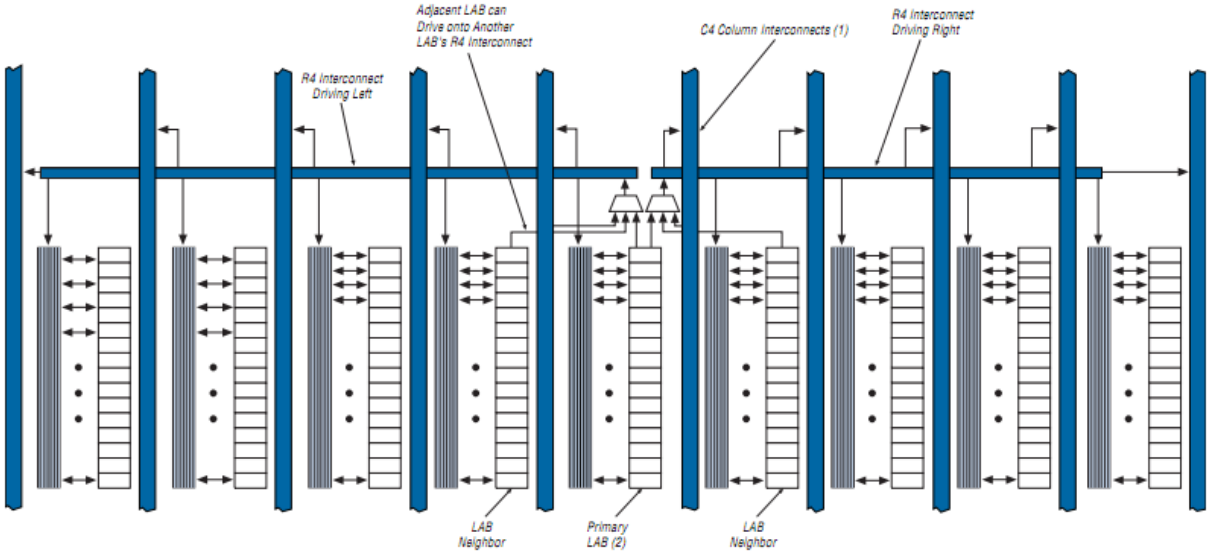
a. Kết nối theo hàng:

Dùng để kết nối các khối trên cùng một hàng, bao gồm:

- Liên kết trực tiếp giữa LAB và các khối kế cận.
- Liên kết R4.
- Liên kết R24 dành cho các ứng dụng truy xuất tốc độ cao.

Liên kết trực tiếp cho phép các khối có thể điều khiển trực tiếp các liên kết cục bộ bên trong các khối xung quanh.

Liên kết R4 mở rộng liên kết ra 4 khối nằm ngang bên trái hoặc phải của khối LAB ban đầu, có thể là 4 khối LAB, 3 khối LAB và một khối M4K hay một khối nhân. R4 tạo nên kết nối hàng tốc độ cao cho các vùng gồm 4 khối LAB.



Hình 2.6: Mô tả kết nối nhiều LAB [5]

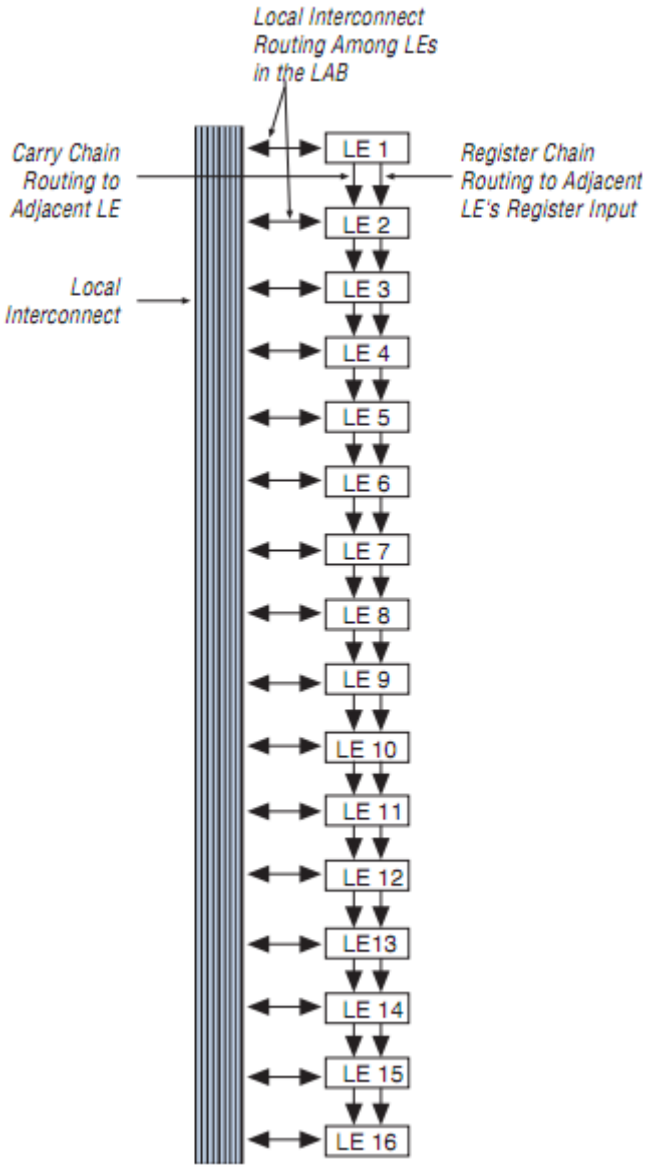
Liên kết R24 mở rộng liên kết ra tới 24 LAB và cung cấp kết nối nhanh nhất cho các khoảng cách dài và cho các khối không kề cận nhau. R24 điều khiển các kết nối hàng hay cột khác ở mỗi LAB thứ 4. R24 sẽ điều khiển các liên kết cục bộ LAB gián tiếp thông qua các liên kết R4 và C4. R24 có thể điều khiển các kết nối R4, C16, C4.

b. Kết nối theo cột:

Hoạt động tương tự như các liên kết hàng, bao gồm:

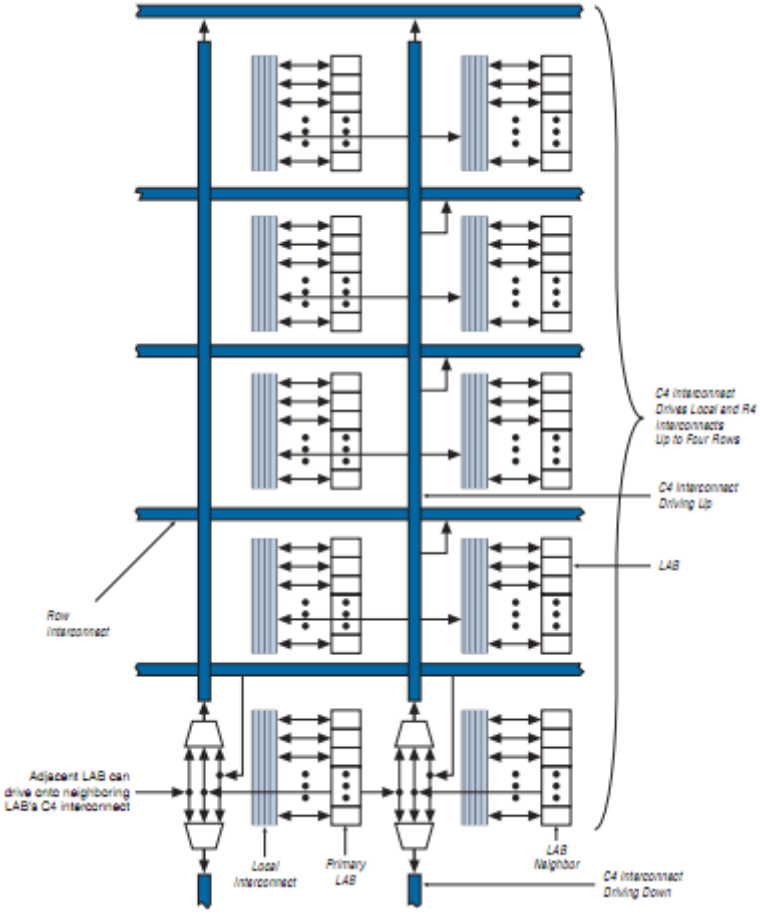
- Kết nối chuỗi thanh ghi bên trong một LAB.
- Liên kết C4.
- Liên kết C16.

Cyclone cung cấp một cấu trúc liên kết nâng cao bên trong mỗi LAB để dẫn ngõ ra thanh ghi của LE này đến ngõ vào thanh ghi của LE kế một cách nhanh chóng bằng cách sử dụng các kết nối chuỗi thanh ghi theo cách dịch thanh ghi.



Hình 2.7: Sơ đồ liên kết chuỗi thanh ghi [5]

Liên kết C4 mở rộng ra 4 khối trên hoặc dưới khối LAB nguồn. Mỗi LAB có một bộ liên kết C4 để điều khiển lên hoặc xuống.



Hình 2.8: Sơ đồ liên kết C4 [5]

C16 mở rộng ra 16 khối và cung cấp liên kết nhanh nhất cho các khoảng cách xa giữa các khối. C16 có thể điều khiển các liên kết hàng hoặc cột khác tại mỗi LAB thứ tư. C16 sẽ kết nối với các liên kết cục bộ của LAB thông qua các liên kết R4 và C4 và sẽ không điều khiển liên kết cục bộ LAB một cách trực tiếp. C16 có thể điều khiển R24, R4, C16 và C4.

2.1.2.4 Mạng lưới clock toàn cục và PLL:

Mạng lưới clock toàn cục bao gồm 16 tín hiệu clock toàn cục, 4PLL, lựa chọn nguồn clock động, cho phép hay không cho phép clock động.

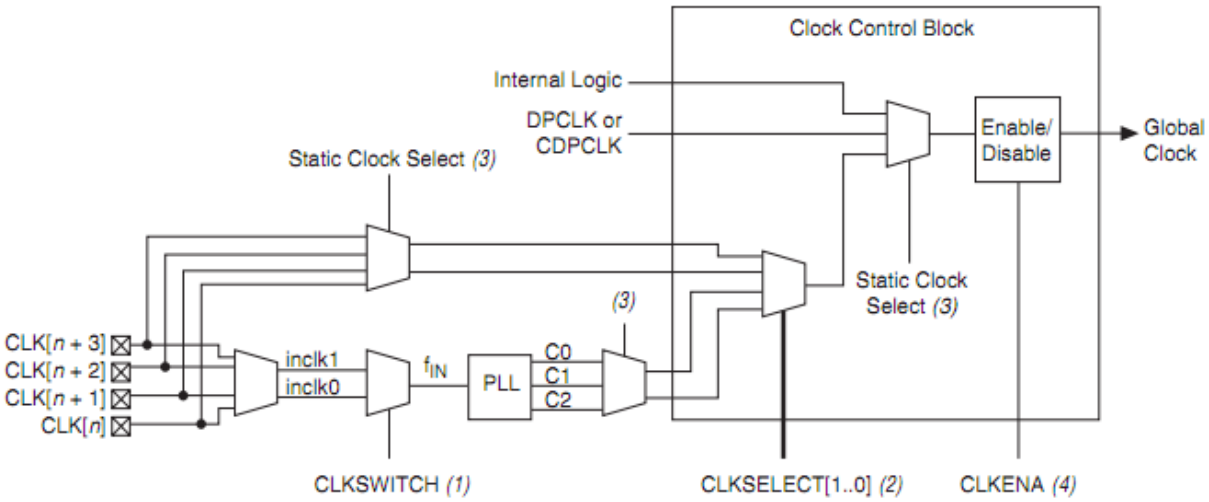
Mỗi mạng lưới clock toàn cục có một khối điều khiển block để chọn một trong các nguồn clock như PLL, các chân CLK (clock), DPCLK (clock dual-purpose), hay từ các nguồn logic.

a. Mạng lưới clock toàn cục:

Khối điều khiển clock:

Tùy theo họ Cyclone, có thể có 8 hay 16 khối điều khiển được sắp xếp theo chu vi của chip. Chức năng của khối này:

- Lựa chọn nguồn clock.
- Cho phép hoặc ngắt nguồn clock.



Hình 2.9: Sơ đồ khối điều khiển clock [5]

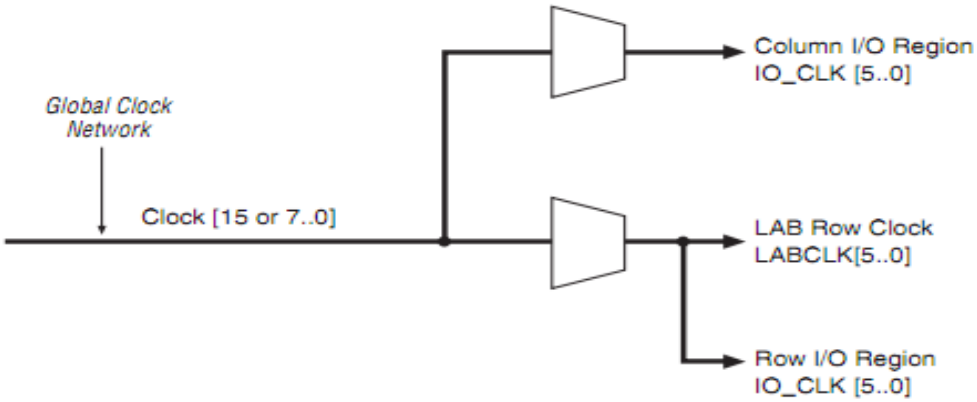
Các nguồn clock có thể là:

- Chân clock cùng phía với khối điều khiển clock.
- 3 ngõ ra clock từ các PLL.
- 4 chân DPCLK cùng phía đối với khối điều khiển clock.
- 4 tín hiệu được tạo bên trong.

Chỉ có 2 chân clock, 2 ngõ ra PLL, một chân DPCLK, một tín hiệu được tạo ra bên trong được chọn để cung cấp cho mạng lưới clock.

Mạng lưới phân phối clock:

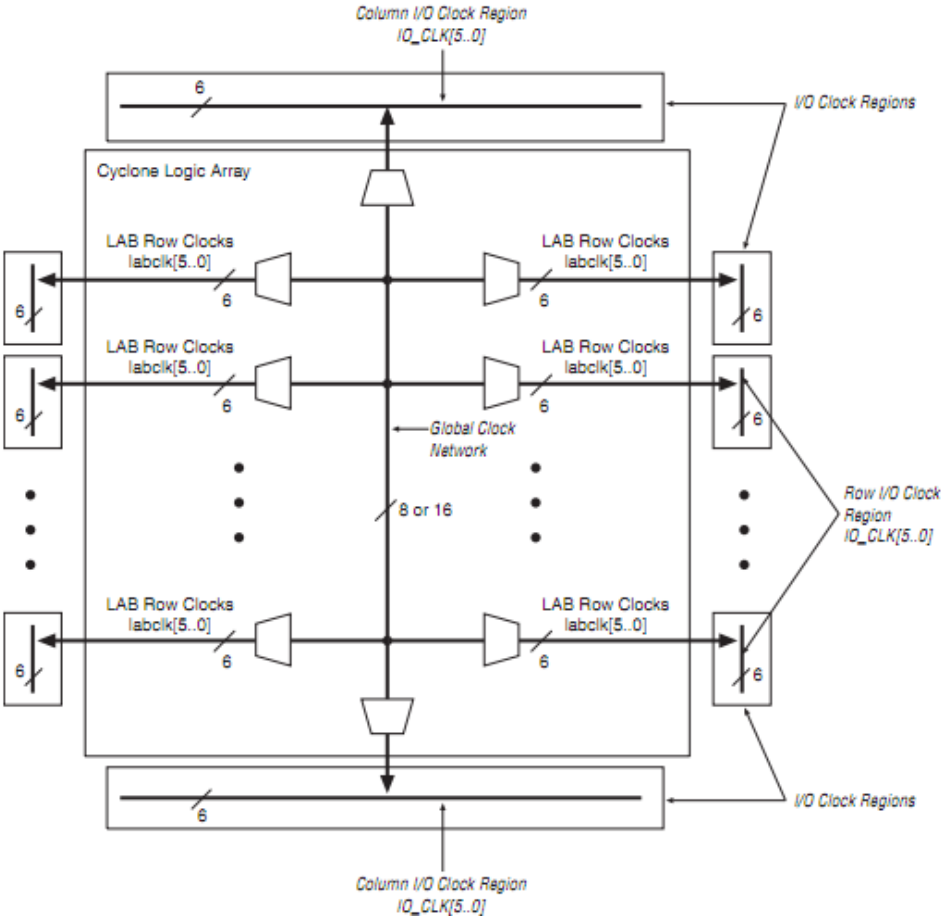
16 mạng lưới clock cùng với các bộ phân kênh sẽ tạo ra các búi 6 bit để điều khiển các clock IOE cột, clock LAB hàng, hay clock cho IOE hàng. Một bộ MUX khác trong LAB để chọn 2 trong số 6 clock hàng LAB để cung cấp cho các LE trong cùng LAB.



Hình 2.10: Bộ phân kênh xung clock [5]

Clock hàng LAB có thể cung cấp cho các LE, M4K, và các khối nhân nhúng và mở rộng ra các vùng clock IO hàng.

Clock IOE liên kết tới các vùng khối hàng và cột. Chỉ 6 nguồn clock toàn cục cung cấp cho các vùng cột và hàng.

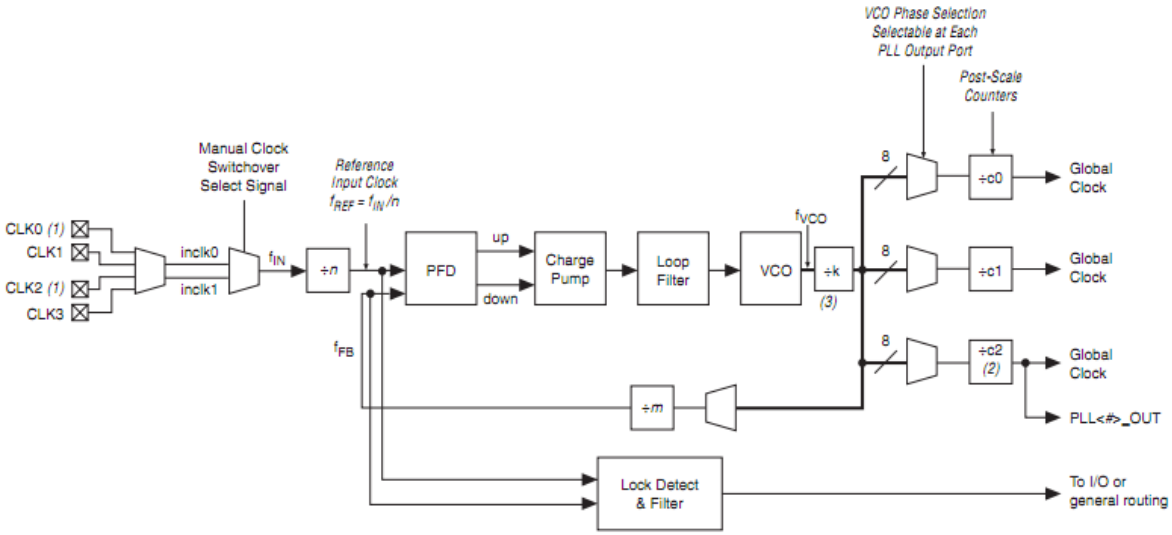


Hình 2.11: Các vùng clock I/O và LAB [5]

b. PLL:

PLL cung cấp các clock đa dụng cũng như cung cấp các chức năng như:

- Nhân và chia clock.
- Dịch pha.
- Chu kỳ nhiệm vụ (duty cycle) lập trình được.
- Tạo ra 3 clock nội khác nhau.
- Tạo ra 1 clock ngoại.
- Các ngõ ra clock cho các hỗ trợ IO khác nhau.
- Các tín hiệu điều khiển…



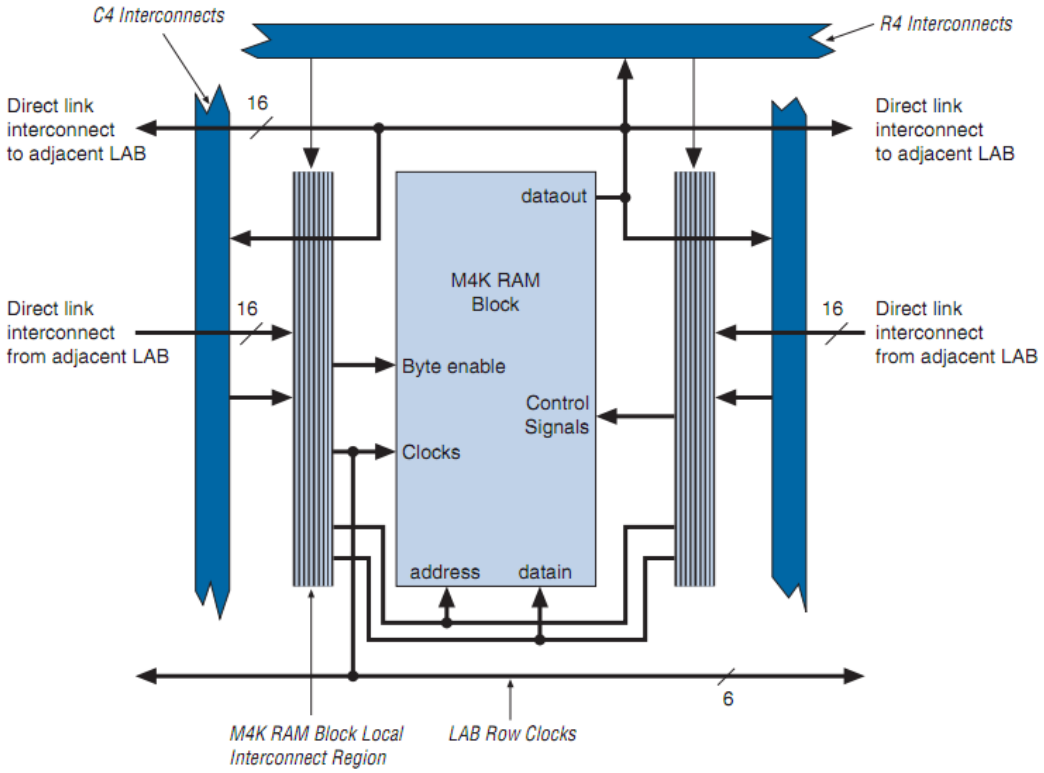
Hình 2.12: Sơ đồ khối của PLL [5]

2.1.2.5 Bộ nhớ nhúng:

Bộ nhớ nhúng bao gồm các cột M4K. Mỗi khối M4K có thể thực hiện nhiều loại bộ nhớ khác nhau, có các tính năng:

- 4068 bit RAM.
- Tốc độ 250 Mhz.
- Bộ nhớ dual-port thực.
- Bộ nhớ dual-port đơn giản.
- Bộ nhớ single-port.
- Các bit Parity.
- Thanh ghi dịch.

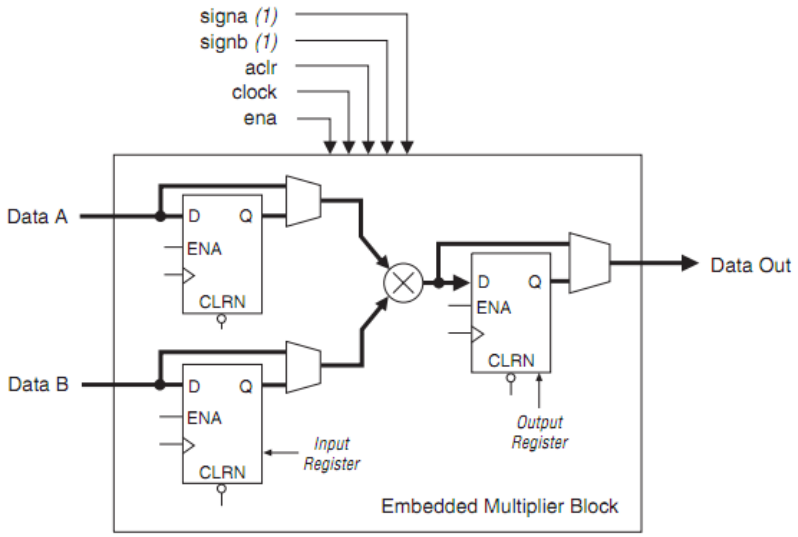
- Đệm FIFO.
- Bộ nhớ ROM…



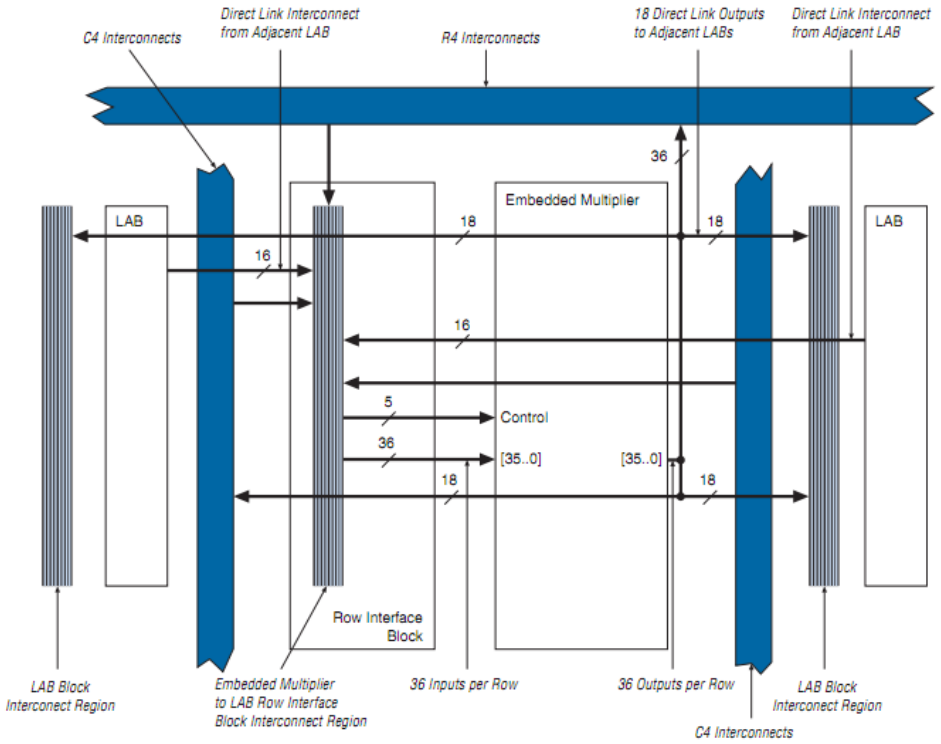
Hình 2.13: Mô tả kết nối giữa LAB với M4K [5]

2.1.2.6 Bộ nhân nhúng :

Cyclone cung cấp bộ nhân nhúng cho các ứng dụng DSP, có thể thực hiện một bộ nhân 18 bit hay 2 bộ nhân 9 bit.



Hình 2.14: Cấu trúc khối nhân [5]

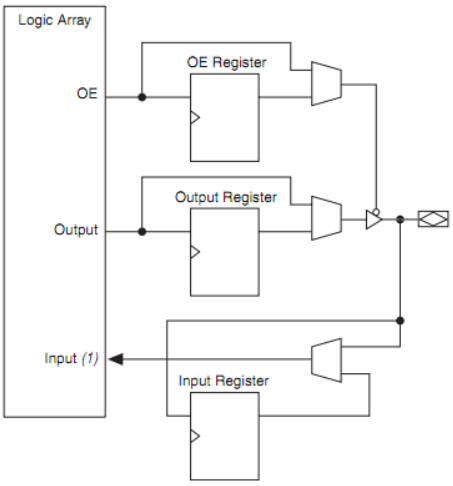


Hình 2.15 : Giao tiếp giữa khối nhân với LAB [5]

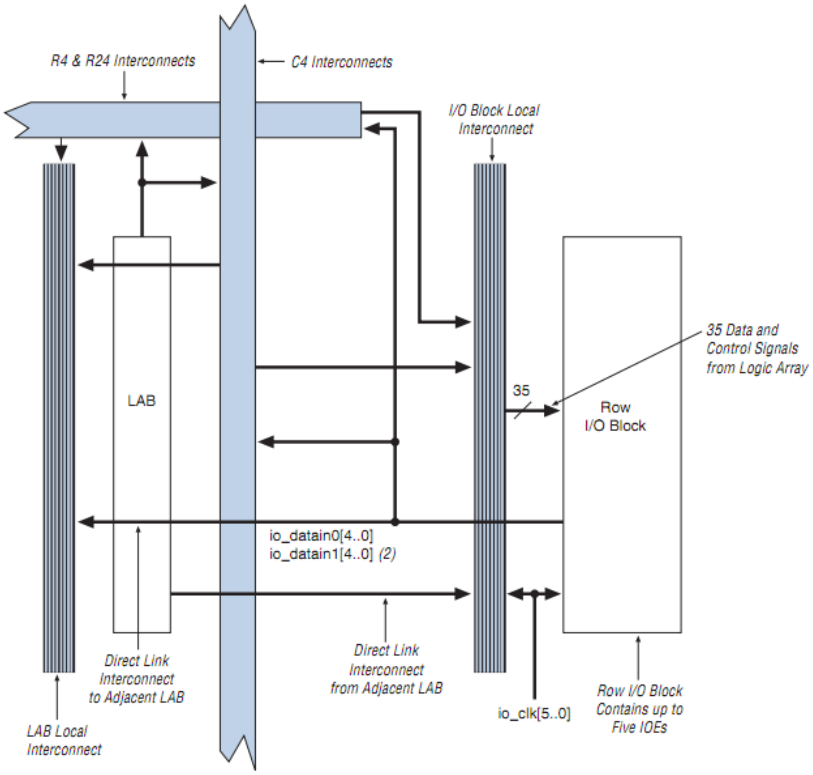
2.1.2.7 Cấu trúc IO :

Các IOE chứa bộ đệm IO hai chiều và 3 thanh ghi hỗ trợ: 1 thanh ghi nhập, 1 thanh ghi xuất, 1 thanh ghi cho phép xuất.

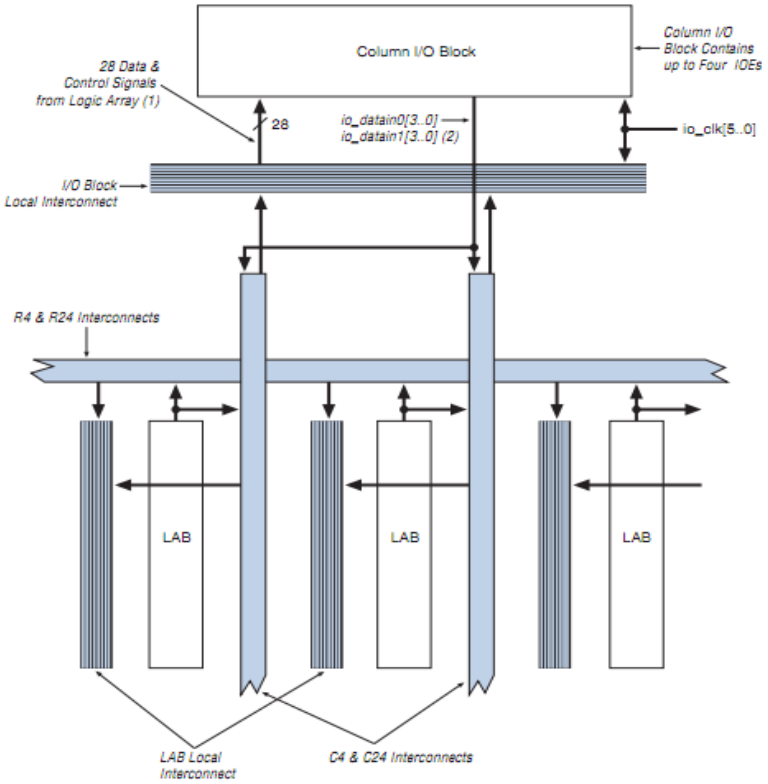
Các IOE nằm trong các khối IO dọc theo chu vi của chip. Có 5 IOE trong mỗi hàng và 4 IOE cho mỗi khối cột. Các khối hàng sẽ điều khiển các liên kết hàng, cột (chỉ cho kết nối C4) hay kết nối trực tiếp.



Hình 2.16: Cấu trúc IOE [5]

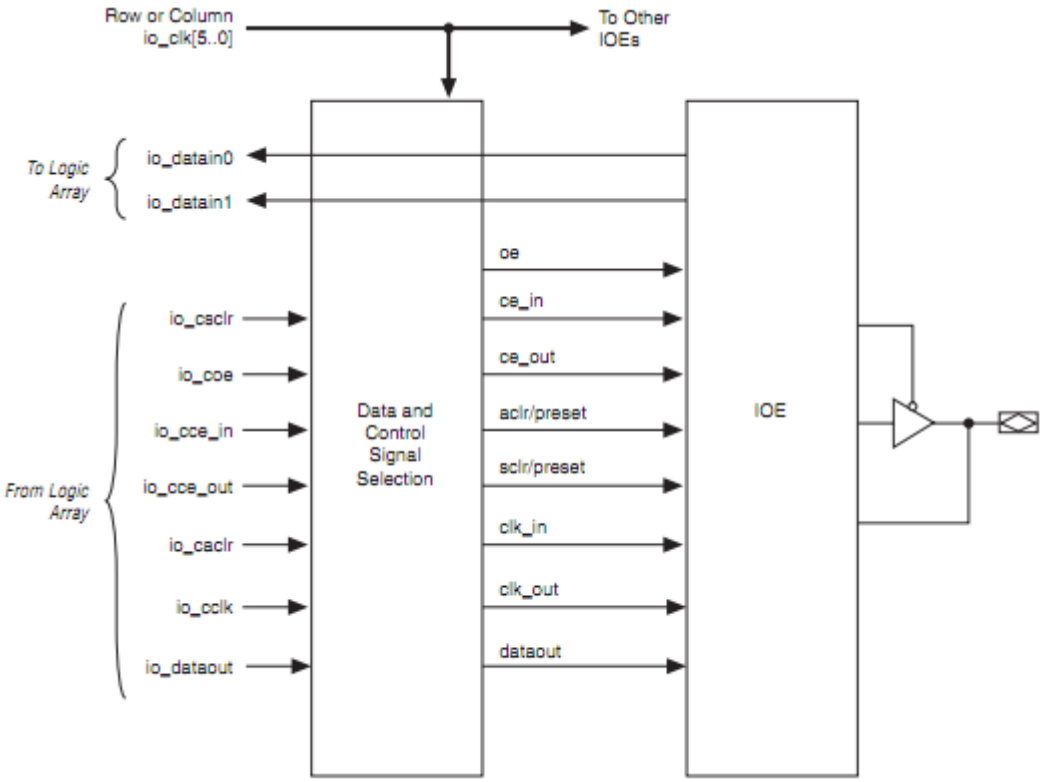


Hình 2.17: Sơ đồ kết nối khối IO hàng vào mạng lưới liên kết [5]



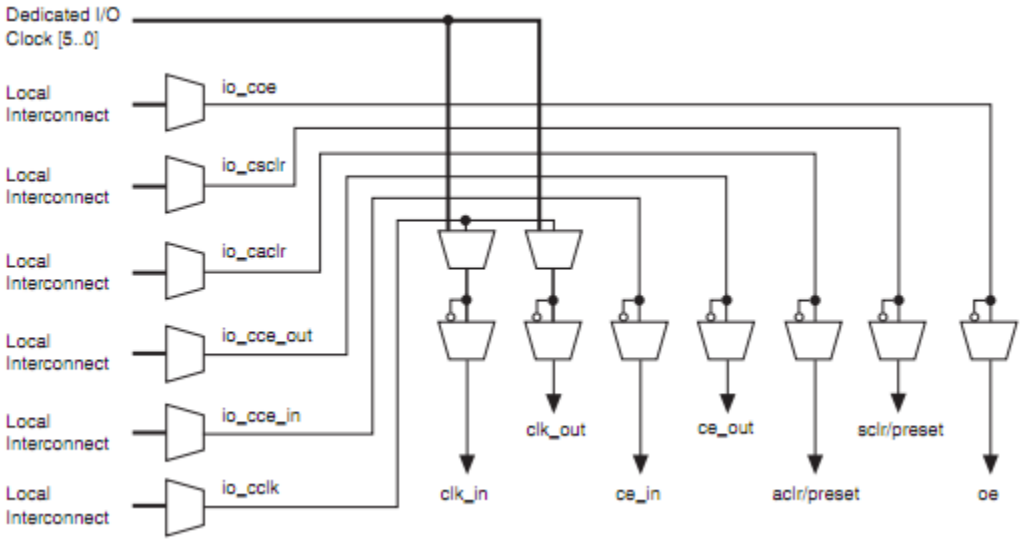
Hình 2.18: Sơ đồ kết nối khối IO cột vào mạng lưới liên kết [5]

Mạng lưới clock toàn cục sẽ tạo ra các clock IOE để cung cấp cho các vùng IO hàng và cột.



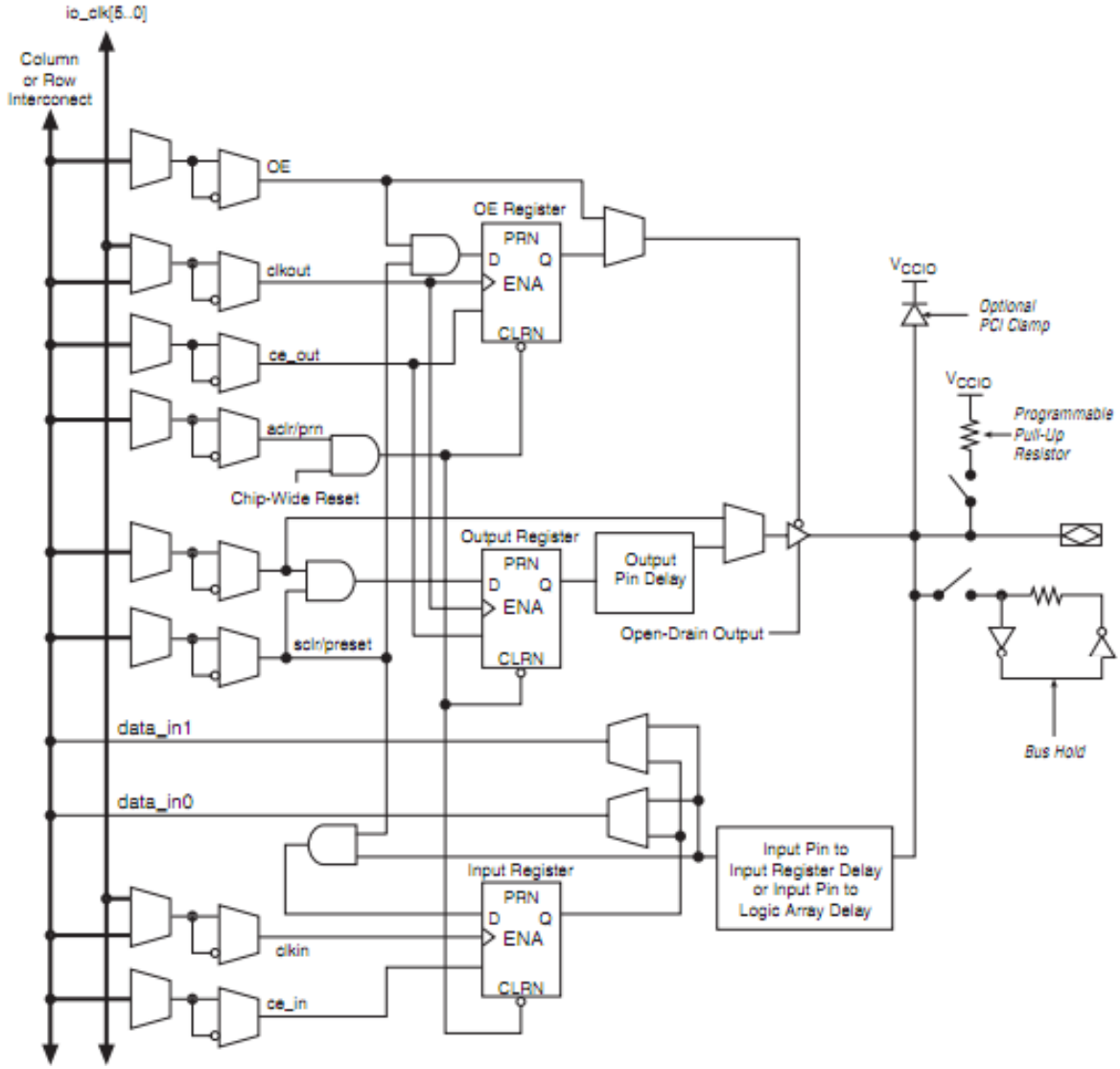
Hình 2.19: Đường tín hiệu đi qua khối IO [5]

Mỗi IOE sẽ có các chọn lựa tín hiệu điều khiển cho riêng mình.



Hình 2.20: Lựa chọn tín hiệu điều khiển cho IOE [5]

Ở chế độ 2 chiều thông thường, ta có thể sử dụng các thanh ghi nhập cho việc nhập dữ liệu tốc độ cao. Thanh ghi nhập có ngõ vào clock và chọn clock riêng biệt với các thanh ghi OE và thanh ghi xuất. Thanh ghi OE và thanh ghi xuất chia sẻ cùng một nguồn xung clock và cùng nguồn cho phép clock từ liên kết cục bộ.



Hình 2.21: Cấu hình 2 chiều của IOE [5]

2.2. Tổng quan về Verilog [1]

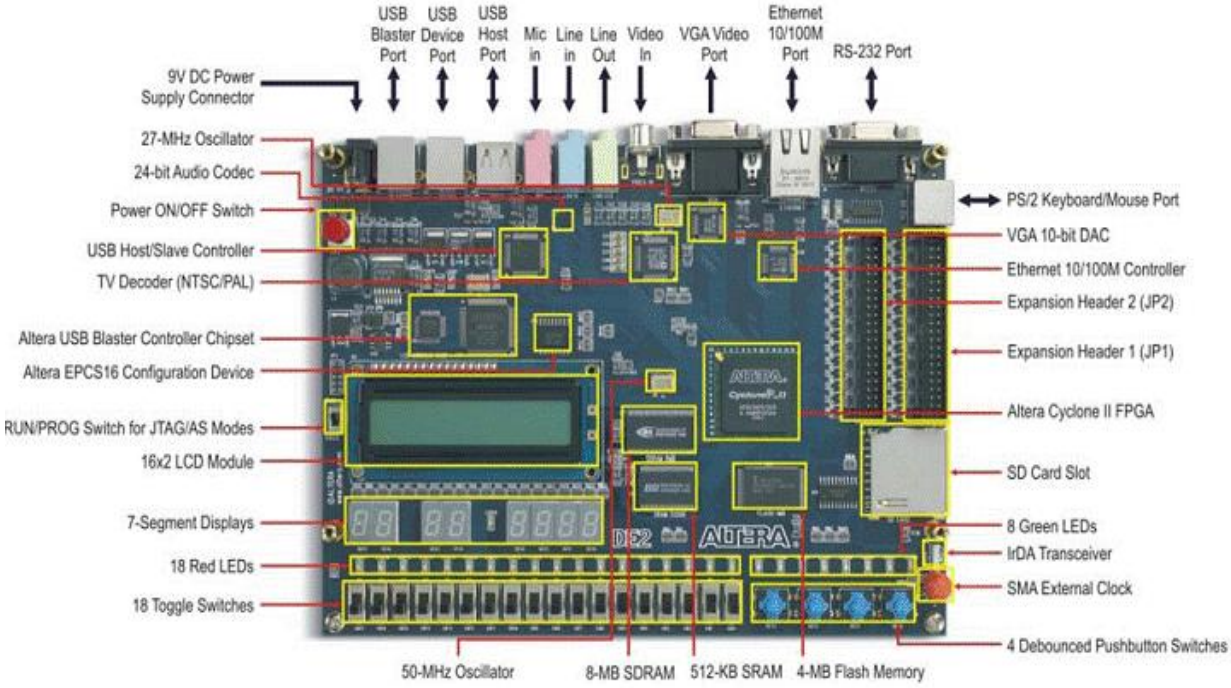
Verilog là một trong hai ngôn ngữ mô tả phần cứng chính (gồm VHDL và Verilog HDL) được người thiết kế phần cứng sử dụng để mô tả, thiết kế các hệ thống số, ví dụ như máy tính hay linh kiện điện tử.

Verilog dễ học và dễ sử dụng hơn VHDL. Verilog được chuẩn hoá theo chuẩn IEEE vào năm 1995 và 2001. Verilog rất giống ngôn ngữ C và được giới chuyên môn nghiên cứu, và được sử dụng nhiều.

Verilog HDL có thể được sử dụng để thiết kế hệ thống số ở nhiều mức khác nhau, ví dụ ở mức cao như các mô hình đặc trưng đến các mức thấp như mô hình bố trí dây, điện trở, transistor trên một mạch tích hợp; mô tả các cổng logic, flip_flop trong hệ thống số; mô tả thanh ghi và sự di chuyển dữ liệu giữa các thanh ghi (RTL - Register Transfer Level).

2.3 Giới thiệu kit DE2

Trong luận văn này tác giả sử dụng phần mềm Quatus II để tổng hợp chương trình sau đó dịch ra mã hex và lập trình hệ thống lên chip FPGA Cyclone II trên kit DE2 của Altera.



Hình 2.22: Kit DE2 [5]

Hình 2.23: Các bộ phận kết nối kit DE2 [5]

Kit DE2 có rất nhiều tính năng cho phép các nhà thiết kế thực hiện một khối
lượng lớn các hệ thống, mạch chức năng từ đơn giản đến phức tạp.

Dưới đây là các tính năng được cung cấp sẵn trên kit DE2:

- Altera Cyclone II 2C35 FPGA
- Altera Serial Configuration device – EPCS16
- USB Blaster dùng để lập trình hệ thống từ PC, hỗ trợ các chế độ JTAG cũng như điều khiển trực tiếp khi sử dụng NIOS II
- 512 – Kbyte SRAM
- 8 – Mbyte SDRAM
- 4-Mbyte Flash memory
- Khe cắm SD Card
- 4 phím nhấn PushButton KEY [3:0]
- 18 Switch (cấp mức ‘0’ hay mức ‘1’)
- 18 đèn LEDR và 9 đèn LEDG
- Có hai nguồn clock là 50 MHz và 27 MHz
- Chip giải mã âm thanh 24 bits với các jack cắm line-in, line-out và microphone

- VGA DAC (10-bit high-speed triple DACs) với cổng VGA
- TV Decoder (NTSC/ PAL) với TV-in (Video-in)
- Cổng 10/ 100 Ethernet
- Bộ điều khiển USB host/slave
- RS – 232 với cổng kết nối 9 chân
- Cổng PS/ 2 giao tiếp với chuột và keyboard
- Cổng hồng ngoại
- 40 chân để mở rộng

Khi tổng hợp chương trình rồi gán chân cho các hệ thống, ta chỉ cần khai báo theo đúng chân. Chương trình Quartus II sẽ tự động gán chân theo đúng bảng chân trên.

Trong luận văn này tác giả sử dụng các cổng USB Blaster để nhận dữ liệu từ PC, cổng VGA để truyền ảnh lên màn hình CRT.

2.4 Khái quát về các loại nhiễu thường gặp trong ảnh siêu âm

Nhiễu là vấn đề hay gặp trong kỹ thuật ảnh, nó làm giảm chất lượng hình ảnh dẫn đến sai lệch thông tin hoặc sai chẩn đoán. Trong thực tế có nhiều loại nhiễu, tuy nhiên chúng tác giả xét 3 loại nhiễu chính : nhiễu cộng, nhiễu nhân, và nhiễu xung.

- Nhiễu cộng : nhiễu cộng thường phân bố khắp ảnh và thường được biểu diễn như sau :

$$X_n = X_g + \eta \qquad (2.1)$$

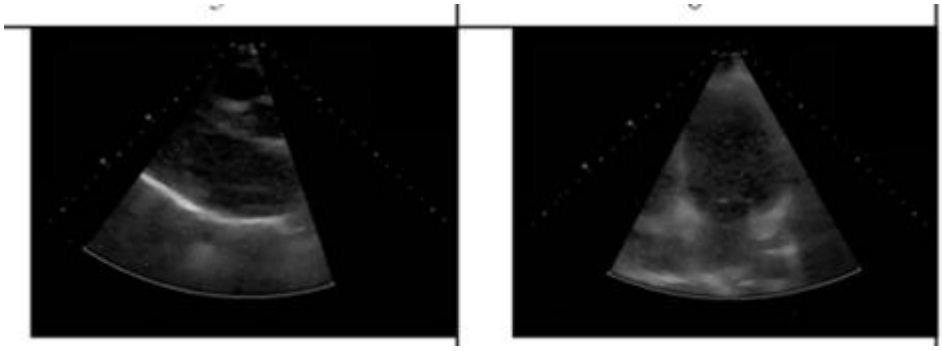
- Nhiễu nhân : cũng thường phân bố khắp ảnh và được biểu diễn như sau :

$$X_n = X_g . \eta \qquad (2.2)$$

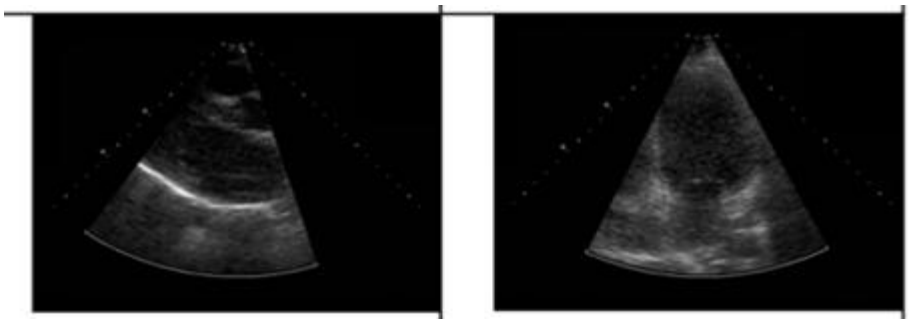
- Nhiễu xung: nhiễu xung thường gây đột biến tại một số điểm ảnh.

Trong đó: X_n : ảnh bị nhiễu, X_g : ảnh gốc, η: nhiễu

Một số nhiễu điển hình như: nhiễu xung, nhiễu muối tiêu, nhiễu gauss, nhiễu đốm,… [3]



Hình 2.24: Ảnh nhiễu muối tiêu [16]



Hình 2.25: Ảnh nhiễu đốm [16]

Trong luận văn này tác giả xét nhiễu đốm trong hình ảnh siêu âm. Trong y tế, siêu âm là phương pháp phổ biến dùng để chuẩn đoán, vì nội tạng và mô mềm không bị ion hóa và chi phí của siêu âm thấp. Nhưng vấn đề lớn của nó là gặp nhiễu đốm. Nhiễu đốm trong hình ảnh siêu âm là do sự can thiệp của năng lượng từ tán xạ phân phối ngẫu nhiên gây ra. Ngoài ra nhiễu đốm có thể do sự tiếp xúc không tốt hay khoảng cách không khí giữa đầu dò và cơ thể con người. Cũng có thể là do mất mát thông tin trong quá trình hình thành chùm tia siêu âm hay trong quá trình chuyển đổi âm và tín hiệu điện. Nó gây ra sự khó khăn trong chuẩn đoán có khi dẫn đến sai lệch thông tin.

Và mô hình nhiễu đốm như sau:

$$X_n = X_g . \eta + \alpha \quad (2.3)$$

Trong đó ŋ và α là nhiễu nhân và nhiễu cộng.

Trong thực tế nhiễu cộng nhỏ hơn nhiều so với nhiễu nhân nên ta có thể viết : $X_n = X_g . \eta$ (2.4)

Chương 3
THỰC HIỆN LỌC NHIỄU ĐỐM SỬ DỤNG MATLAB

Trong chương này tác giả nêu các phương pháp lọc nhiễu đốm, sử dụng phần mềm Matlab để lọc ảnh nhiễu đốm, nhiễu muối tiêu (để so sánh) bằng cách sử dụng hai bộ lọc Median và Modify Hybrid Median

3.1 Giới thiệu về Matlab và các phương pháp lọc nhiễu sử dụng Matlab

MATLAB – phần mềm nổi tiếng của công ty MathWorks. Nó tích hợp tính toán, hiện thị và lập trình trong một môi trường dễ sử dụng. Các ứng dụng tiêu biểu của MATLAB bao gồm:

- Hỗ trợ toán học và tính toán.
- Phát triển thuật toán.
- Mô hình, mô phỏng.
- Phân tích, khảo sát và hiển thị số liệu.
- Đồ họa khoa học và kỹ thuật.
- Phát triển ứng dụng với các giao diện đồ họa.

Ngoài ra phần mềm MATLAB còn được trang bị thêm các ToolBox – các gói chương trình (thư viện) cho các lĩnh vực ứng dụng rất đa dạng như xử lý tín hiệu, nhận dạng hệ thống, xử lý ảnh, mạng nơ ron, logic mờ, tài chính, tối ưu hóa, phương trình đạo hàm riêng,... Đây là các tập hợp mã nguồn viết bằng chính MATLAB dựa theo các thuật toán mới, hữu hiệu mà người dùng có thể chỉnh sửa hoặc bổ sung thêm các hàm mới.

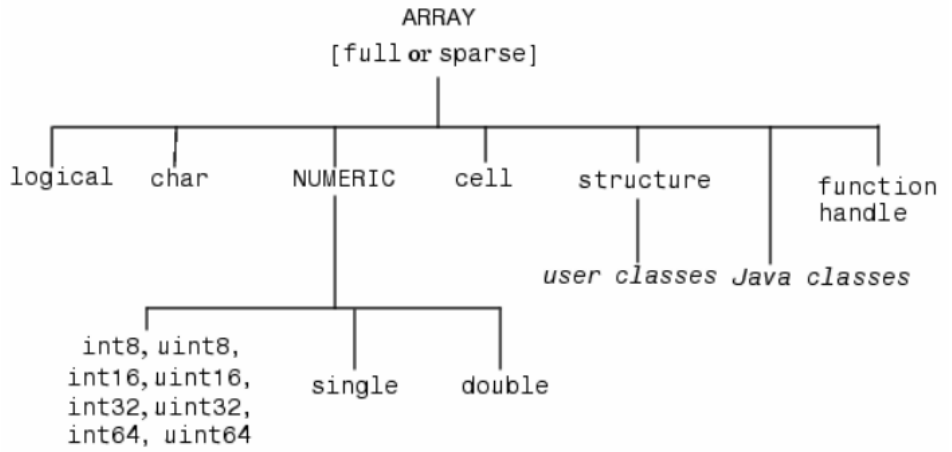
MATLAB được thiết kế để giải các bài toán bằng số chứ không nhằm mục đích chính là tính toán ký hiệu như MATHEMATICA và MAPLE. Tuy nhiên, trong MATLAB cũng có thể tính toán ký hiệu được nhờ các hàm trong Symbolic Math ToolBox [7]

3.1.1 Dữ liệu

Dữ liệu của Matlab thể hiện dưới dạng ma trận (hoặc mảng – tổng quát), có các kiểu dữ liệu được liệt kê sau đây:

- Kiểu Single: kiểu này có lợi về bộ nhớ dữ liệu vì nó đòi hỏi ít byte nhớ hơn, kiểu dữ liệu này ít được sử dụng trong các phép tính toán học vì độ chính xác không cao.
- Kiểu Double: kiểu này kiểu thông dụng nhất của các biến trong Matlab.
- Kiểu int8, int16 int32 int64; uint8, uint16, uint32, uint64
- Kiểu char
- Kiểu cell
- Kiểu Structure

Trong Matlab kiểu dữ liệu double là kiểu mặc định sử dụng trong các phép số học.



Hình 3.1: Các kiểu dữ liệu trong Matlab [7]

3.1.2 Ứng dụng

Matlab được ứng dụng trong hầu hết các lĩnh vực trong khoa học tự nhiên cũng như về nhu cầu tính toán trong các ngành ngân hàng, địa lý, sinh học, y học,…

Matlab cung cấp các công cụ rất hữu ích trong các lĩnh vực gần với thực tế như mô phỏng hay các bài toán về xác xuất, đặc biệt là trong lĩnh vực xử lý tín

hiệu số, xử lý ảnh số, xử lý âm thanh, hay về mã hóa, điều chế, đánh giá kênh truyền và tính toán các bộ cân bằng, …

3.1.3 Hệ thống giao diện

Môi trường phát triển: Đây là nơi đặt các thanh công cụ, các phương tiện giúp ta sử dụng các lệnh, các file như :

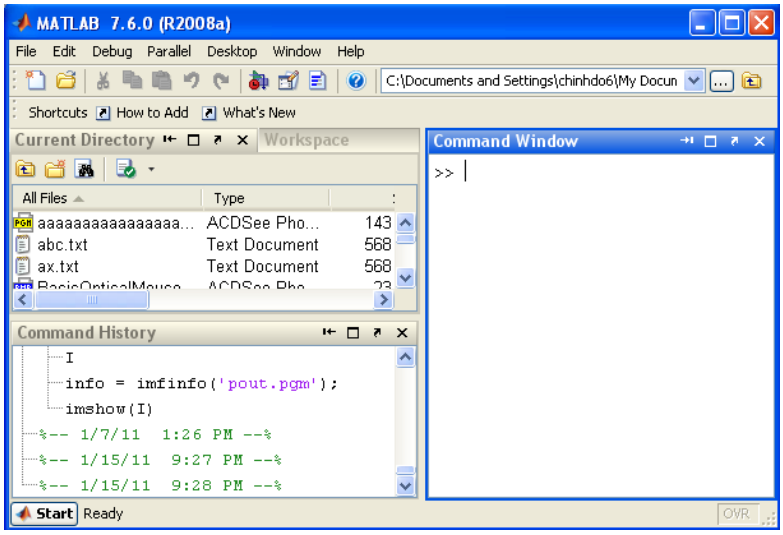
- Desktop
- Command Window
- Command History
- Browsers for viewinghelp

Thư viện, các hàm toán học: Bao gồm các cấu trúc như tính tổng, sin, cosin, tan, cotan, … Các phép tính từ đơn giản đến phức tạp như tính các ma trận nghịch đảo, trị riêng, chuyển đổi fourrier, laplace, …

Ngôn ngữ Matlab: Đó là ngôn ngữ cao về ma trận và mảng, với các dòng lệnh, các hàm, cấu trúc dữ liệu vào, có thể lập trình hướng đối tượng.

Đồ họa trong Matlab: Bao gồm các câu lệnh thể hiện đồ họa trong môi trường 2D và 3D, tạo các giao diện tương tác giữa người sử dụng và máy tính.

Giao tiếp với các ngôn ngữ khác: Cho phép tương tác với các ngôn ngữ C hay Fortran.



Hình 3.2: Giao diện của Matlab [7]

3.1.4 Hộp công cụ xử lý ảnh:

Do Matlab cung cấp rất nhiều Toolbox cho các lĩnh vực khác nhau với các ứng dụng riêng biệt nên ta chỉ giới thiệu về Image Processing Toolbox với các lệnh được ứng dụng trong luận văn này.

3.1.4.1 Các loại ảnh trong Matlab:

Image Processing Toolbox hỗ trợ 4 dạng ảnh cơ bản:

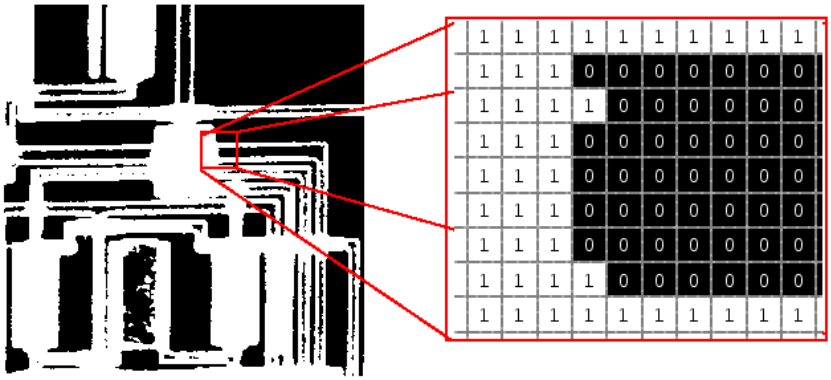
- Ảnh nhị phân (Binary Image)
- Ảnh Indexed (Indexed Image)
- Ảnh mức xám (Grayscale Image)
- Ảnh màu (Truecolor Image)

Chúng ta sẽ đi vào chi tiết từng dạng ảnh bên dưới.

a. Ảnh nhị phân (Binary Image):

Với ảnh nhị phân (Binary image) mỗi pixel mang một và chỉ một trong 2 giá trị 0 hoặc 1. Mỗi ảnh nhị phân được lưu trữ trong một mảng logic. Để dễ dàng chúng ta dùng tên BW (Black & White) để chỉ rằng đây là ảnh Binary image.

Ảnh minh họa sau là ảnh BW với một vùng được phóng to để chúng ta có thể thấy được giá trị các pixel.



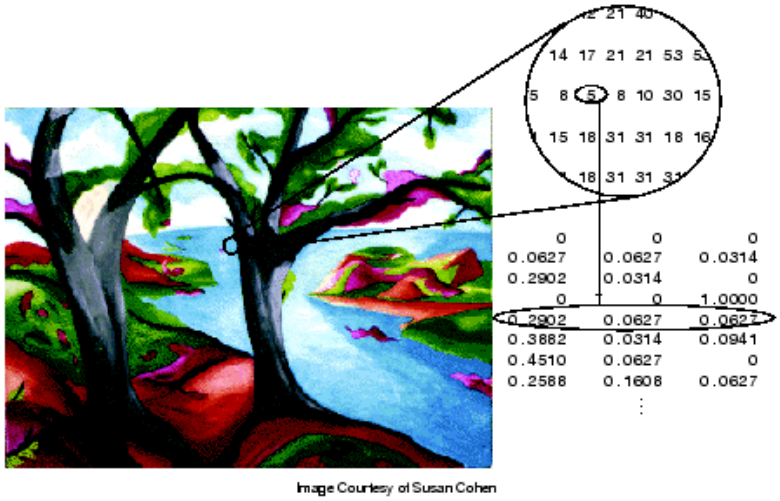
Hình 3.3: Minh họa ảnh nhị phân [6]

b. Ảnh Indexed (Indexed Image):

Một ảnh Indexed Image bao gồm một mảng và một ma trận màu. Mỗi giá trị pixel trong mảng chỉ trực tiếp đến một bảng màu. Để cho thuận tiện trong đề

tài này dùng biến tên X để chỉ rằng mảng và ma trận đang trỏ tới bảng màu (color map).

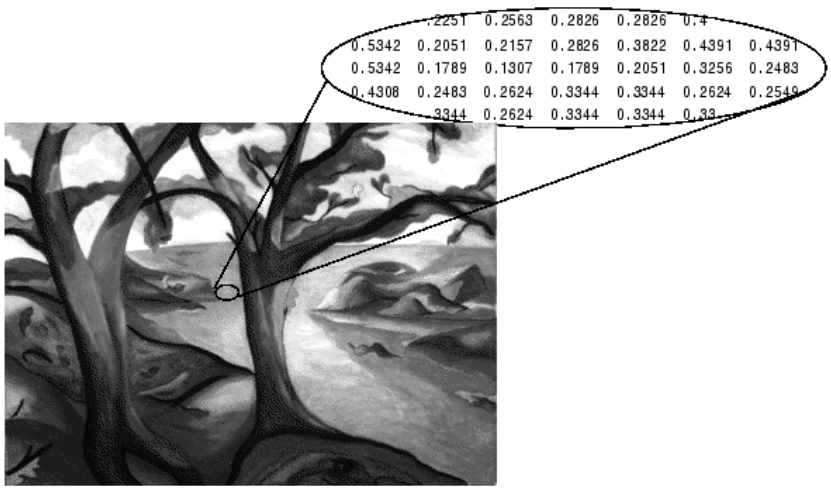
Ma trận bảng màu (colormap matrix) là một mảng số thực m x 3 bao gồm các giá trị nằm trong dải [0,1]. Mỗi hàng của bảng chỉ ra các thành phần màu Red, Green và Blue của một màu đơn. Một ảnh indexed image trỏ trực tiếp giá trị pixel tới các giá trị trong bảng màu. Màu của mỗi pixel của ảnh được quyết định bởi việc dùng giá trị tương ứng của X như chỉ danh trong bảng màu.



Hình 3.4: Minh họa ảnh indexed [6]

c. Grayscale Image:

Là một ma trận số, mà giá trị của nó đại diện cho độ sáng của các pixel ảnh, dữ liệu trong ma trận có thể được lưu ở double, uint8, or uint16.



Hình 3.5: Minh họa ảnh mức xám [6]

d. Truecolor Image (RBG Image)

Là ma trận ảnh được lưu dưới dạng m x n x3 (m-by-n-by-3 data array), đại diện cho ba màu đỏ, xanh lá, xanh dương. Màu của mỗi pixel ảnh do sự kết hợp giá trị của 3 giá trị của mỗi ma trận ở pixel tương ứng, và giá trị của pixel trong mỗi ma trận được mặc định 8 bits tức là ảnh RGB sẽ có độ phân giải 24 bits.

Hình 3.6: Minh họa ảnh RGB [6]

3.1.4.2 Các khái niệm cơ bản:

Cấu trúc cơ bản trong Matlab là mảng (array) – một tập hợp có thứ tự của các phần tử là số thực hay số phức. Đối tượng này tất nhiên rất phù hợp cho miêu tả image – một tập màu sắc hoặc dữ liệu cường độ theo thứ tự giá trị thực.

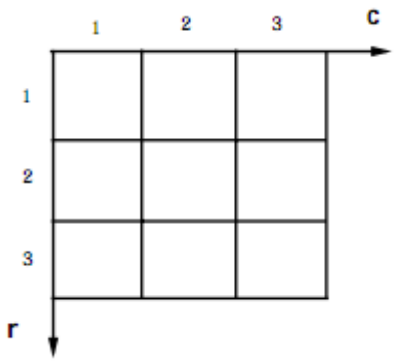
Pixel (điểm ảnh) được lấy được từ phần tử ảnh và thường biểu thị một điểm trong hiển thị máy tính.

Matlab lưu trữ phần lớn dưới dạng mảng 2 chiều (hay gọi là ma trận), trong đó mỗi phần tử của ma trận tương ứng với một điểm ảnh trong image.

Hệ tọa độ (Coordinate system):

Nhìn chung, cách phổ biến nhất để biểu diễn vị trí trong một image là dùng toạ độ điểm ảnh. Trong hệ tọa độ pixel này, mỗi ảnh được xem như một

mạng lưới các phần tử rời rạc, thứ tự từ trên xuống và từ trái sang phải, như hình minh họa:



Hình 3.7: Tọa độ điểm ảnh [6]

Đối với mỗi tọa độ pixel, thành phần đầu tiên r (viết tắt của row – hàng) tăng từ trên xuống dưới, trong khi phần tử thứ hai là c (viết tắt của column) tăng từ trái sang phải. Các tọa độ pixel là các giá trị nguyên và thay đổi trong dải từ 1 đến chiều dài của cột hoặc hàng.

Có sự tương ứng 1 – 1 giữa tọa độ trong Matlab dùng cho các ma trận. Sự tương ứng này làm cho mối quan hệ giữa ma trận dữ liệu của ảnh image và cách mà image được hiển thị dễ hiểu hơn nhiều.

Ví dụ: dữ liệu cho điểm ảnh ở hàng thứ 15, cột thứ 2 được lưu trong phần tử (15,2) của ma trận. Do đó, I (2,15) trả về giá trị của điểm ảnh ở hàng thứ 2 và cột thứ 15 của ảnh I

Chuyển đổi giữa các kiểu ảnh:

Danh sách sau sẽ liệt kê các hàm được sử dụng trong việc chuyển dạng ảnh :

- Dither: Tạo một ảnh nhị phân từ một ảnh cường độ đen trắng bằng cách trộn, tạo một ảnh chỉ số từ một ảnh RGB bằng cách trộn (Dither)
- Gray2id: Tạo một ảnh chỉ số từ một ảnh cường độ đen trắng .
- Grayslice: Tạo một ảnh chỉ số từ một ảnh cường độ đen trắng bằng cách đặt ngưỡng

- Im2bw: Tạo một ảnh nhị phân từ một ảnh cường độ, ảnh chỉ số hay ảnh RGB trên cơ sở của ngưỡng ánh sáng .
- Ind2gray: Tạo một ảnh cường độ đen trắng từ một ảnh chỉ số
- Ind2rgb: Tạo một ảnh RGB từ một ảnh chỉ số
- Mat2gray: Tạo một ảnh cường độ đen trắng từ dữ liệu trong một ma trận bằng cách lấy tỉ lệ giữ liệu
- Rgb2gray: Tạo một ảnh cường độ đen trắng từ một ảnh RGB
- Rgb2ind: Tạo một ảnh chỉ số từ một ảnh RGB

Chuyển đổi giữa các dữ liệu ảnh:

Chúng ta có thể chuyển đổi dữ liệu ảnh kiểu uint8 và uint16 sang kiểu double bằng cách dùng các hàm double của Matlab. Thế nhưng, chuyển dổi giữa các kiểu ảnh này tất nhiên sẽ thay đổi cách thức của Matlab để hiểu được các dữ liệu ảnh. Nếu ta muốn mảng kết quả có thể được xử lý như là các dữ liệu ảnh, ta cần phải cân chỉnh lại hoặc bù vào dữ liệu khi ta chuyển đổi chúng.

Để dễ dàng chuyển đổi giữa các loại ảnh, ta dùng một trong các hàm của toolbox dưới đây: im2uint8, im2uint16, im2single, hoặc im2double. Các hàm này sẽ tự động căn chỉnh lại hoặc bù lại các dữ liệu gốc của bất kì ảnh nào.

Ví dụ, command này sẽ chuyển đổi ảnh RGB kiểu double với dữ liệu nằm trong dải [0,1] thành kiểu ảnh RGB uint8 với dữ liệu nằm trong dải [0,255].

RGB2 = im2uint8 (RGB1);

Mất dữ liệu khi chuyển đổi:

Khi ta chuyển từ một lớp ảnh được mã hóa bằng số bit lớn hơn số bit của định dạng ảnh sẽ chuyển đổi thì nhìn chung bạn sẽ mất đi một số thông tin trong ảnh đó. Ví dụ: một ảnh grayscale uint16 có khả năng lưu trữ tối đa 65536 dạng riêng của màu xám, nhưng ảnh grayscale uint8 chỉ lưu trữ tối đa 256 dạng khác nhau của màu xám. Khi ta chuyển đổi từ ảnh uint16 sang ảnh uint8, thì hàm im2uint8 lượng tử hóa các dạng xám về mạng ảnh gốc. Nói cách khác, tất cả giá trị từ 0 tới 127 trong ảnh gốc sẽ trở thành 0 trong ảnh uint8, giá trị từ 128 tới 385 sẽ trở thành 1, và tiếp tục.

Một số lệnh cơ bản:

Đầu tiên hãy xóa hết các biến trong workspace và các figure đang mở bằng lệnh:

Close all

Để biết thêm chi tiết về lệnh nào đó:

Help tên lệnh

Ví dụ: help imread

Để đọc một file ảnh dung lệnh *imread*. Để dễ nhớ thì nghĩ như sau: *im* là viết tắt của image. Còn tiếp theo sau là các từ miêu tả ngắn gọn lệnh.

Imread(filename,fmt)

Filename là tên file (ví dụ : ex1.jpg). Nếu *filename* không ở trong thư mục hiện tại hoặc trong *matlab path* thì *filename* là đường dẫn đến file ảnh. Khi đó *fmt* là tên mở rộng của file.

Sau khi đọc xong một ảnh, ta sẽ hiển thị ảnh bằng lệnh :

Imshow(X, map)

Với *X* là dữ liệu vừa đọc ảnh mà ta muốn hiển thị, '*map*' là bản đồ màu tương ứng mà ta muốn hiển thị.

3.1.5 Phương pháp lọc nhiễu:

Ta xem xét và so sánh hai bộ lọc nhiễu sau :

3.1.5.1. Bộ lọc Median:

Trung vị x_a của một chuỗi n phần tử $\{x_n\}$ được định nghĩa :

- Nếu n lẻ: có (n-1)/ 2 phần tử lớn hơn x_a và (n-1)/ 2 phần tử nhỏ hơn hay bằng x_a.

- Nếu n chẵn: x_a là trung bình cộng của 2 phần tử x_i và x_j ∈ $\{x_n\}$ sao cho có (n-2)/2 phần tử nhỏ hơn hay bằng x_i (n-2)/ 2 phần tử lớn hơn hay bằng x_j.

Thuật toán lọc trung vị được dùng để lọc nhiễu bằng cách trượt trên mặt phẳng ảnh, mội lần trượt di chuyển một cột điểm. Những phần tử trong cửa sổ

được coi như là một chuỗi $\{x_n\}$ và điểm quan tâm được thay thế bởi giá trị x_a của chuỗi.

Một cách tổng quát, ta có thuật toán như sau:

B_1: Lấy các phần tử trong cửa sổ ra mảng một chiều (L phần tử).

B_2: Tìm Min của lần lượt các chuỗi con rồi lấy Max: gọi m_1 là giá trị này.

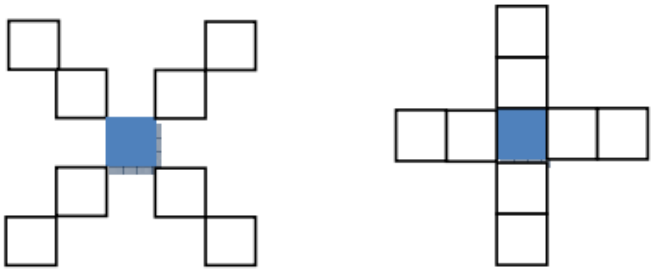
B_3: Tìm Max của lần lượt các chuỗi con rồi lấy Min: gọi m_2 là giá trị tìm được.

B_4: Gán giá trị điểm đang xét là trung bình cộng của m_1 và m_2.

Với lọc trung vị, số lượng tính toán khá lớn *(có thể bằng số mũ của kích thước cửa sổ lọc).*Vì vậy, để khắc phục nhược điểm này tác giả đã dùng phương pháp lọc Modified hybrid median [6]

3.1.5.2 Bộ lọc Modified hybrid median [8]:

Bộ lọc Modified hybrid median tạo ra ma trận cửa sổ hình chữ " X" và chữ " +" trong ma trận ảnh được chia thành 2 nhóm : một nhóm các giá trị 45° tạo thành chữ " X" và một nhóm các giá trị 90° tạo thành chữ thập " + ". Ví dụ ma trận ảnh 5*5 sẽ được biểu diễn như hình sau :



Và kết hợp chúng ta được :

$$\begin{pmatrix} D & * & R & * & D \\ * & D & R & D & * \\ R & R & C & R & R \\ * & D & R & D & * \\ D & * & R & * & D \end{pmatrix}$$

Hình 3.8: Ma trận bộ lọc Modifed hybrid median [8]

Giải thuật xử lý như sau :

- Tìm median MR của các pixels R và pixel trung tâm C
- Tìm maximun MXD của các pixels D và pixel trung tâm C
- Pixel cuối cùng sau xử lý là : M = median{ MR, MXD, C}

3.1.5.3 So sánh các bộ lọc [8]:

a. Tiêu chuẩn SNR: Tỉ lệ tín hiệu gốc so với nhiễu

$$SNR = 10.\log_{10}\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{i,j}^2+y_{i,j}^2)}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(x_{i,j}-y_{i,j}\right)^2} \quad (3.1)$$

b. Tiêu chuẩn RMSE: Căn bậc hai của sai số bình phương trung bình:

$$RMSE = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(x_{i,j}-y_{i,j}\right)^2} \quad (3.2)$$

c. Tiêu chuẩn PSNR: Tỉ lệ tín hiệu đỉnh trên nhiễu

$$PSNR = 20.\log_{10}(g_{max}^2/RMSE) \quad (3.3)$$

g_{max} : cường độ lớn nhất của ảnh trước khi lọc

d. QI: Chỉ số chất lượng phổ :

$$QI = \frac{\sigma_{xy}}{\sigma_x\sigma_y}\frac{2\bar{y}\bar{x}}{(\bar{y})^2+(\bar{x})^2}\frac{2\sigma_y\sigma_x}{\sigma_x^2+\sigma_y^2} \quad (3.4)$$

$\bar{x}, \bar{y}$ là trung bình ảnh gốc và ảnh sau khi lọc

σ_x, σ_y độ lệch tiêu chuẩn của ảnh gốc và ảnh sau khi xử lý

σ_{xy} phương sai giữa ảnh gốc và sau khi lọc

e. SSI: Chỉ số kết cấu tương tự giữa hai ảnh :

$$SSI = \frac{(2\bar{x}\bar{y}+2.55)(\sigma_{xy}+7.65)}{(\bar{x}^2+\bar{y}^2+2.55)(\sigma_x^2+\sigma_y^2+7.65)} \quad (3.5)$$

SSI nằm giữa 0 và 1 là xấu, và bằng 1 là tốt.

f. EPF: Khả năng bảo quản cạnh hình ảnh

Trong ảnh siêu âm bị nhiễu đốm, ngoài việc quan tâm đến hiệu suất, chúng ta còn quan tâm đến khả năng bảo quản cạnh của hình ảnh

$$EPF = \frac{\sum(\Delta x - \overline{\Delta x})(\Delta y - \overline{\Delta y})}{\sqrt{\sum(\Delta x - \overline{\Delta x})^2 \sum(\Delta y - \overline{\Delta y})^2}} \quad (3.6)$$

Δx, Δy : là phiên bản qua bộ lọc thông cao của ảnh x, y theo tiêu chuẩn ước lượng toán tử Laplace.

EPF càng lớn thì khả năng bảo quản cạnh của ảnh càng cao

g. Kết quả so sánh các bộ lọc:

Tác giả mô phỏng các bộ lọc trên bằng Matlab với σ^2 = 0,2, ta được kết quả sau :

Bảng 3.1: Kết quả so sánh hai bộ lọc Median và Modified hybrid median với σ^2 = 0,2

Filter	RMSE	SNR	PSNR	EPF	SSI	QI
Median	12,1956	17,2011	29,4170	0,2728	0,8185	0,9249
Modified hybrid median	6,1556	25,8382	37,7046	0,9279	1,0000	0,9983

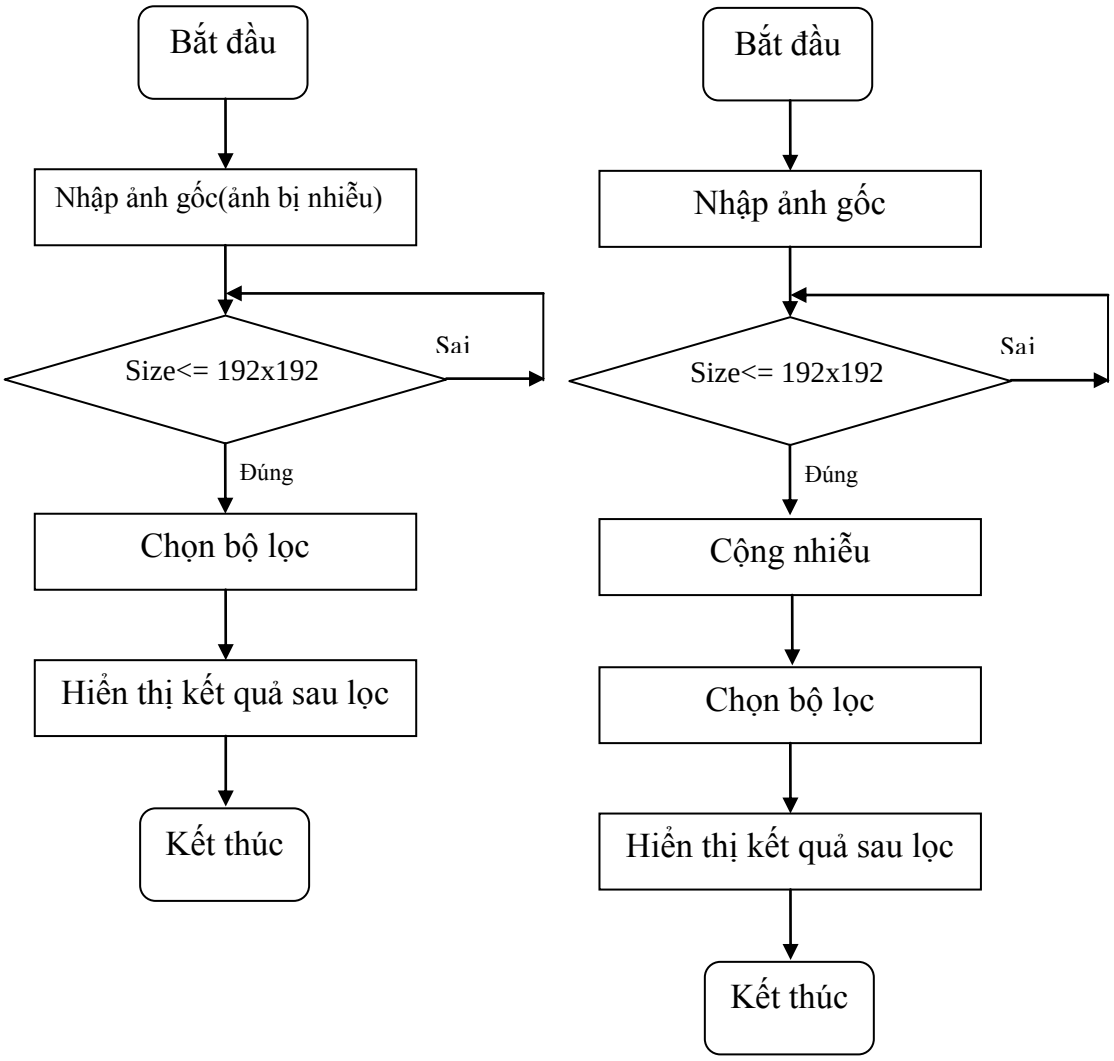
Kết quả cho thấy, bộ lọc Modified hybrid median hiệu quả hơn và được lựa chọn để thực hiện lọc nhiễu đốm cho ảnh siêu âm trong luận văn này.

3.2 Thiết kế và lọc nhiễu đốm sử dụng Matlab

3.2.1 Yêu cầu:

- Tìm hiểu kỹ thuật lọc nhiễu.
- Nguyên cứu giải thuật lọc Median và Modified hybrid median.
- Thực hiện lọc nhiễu sử dụng Matlab

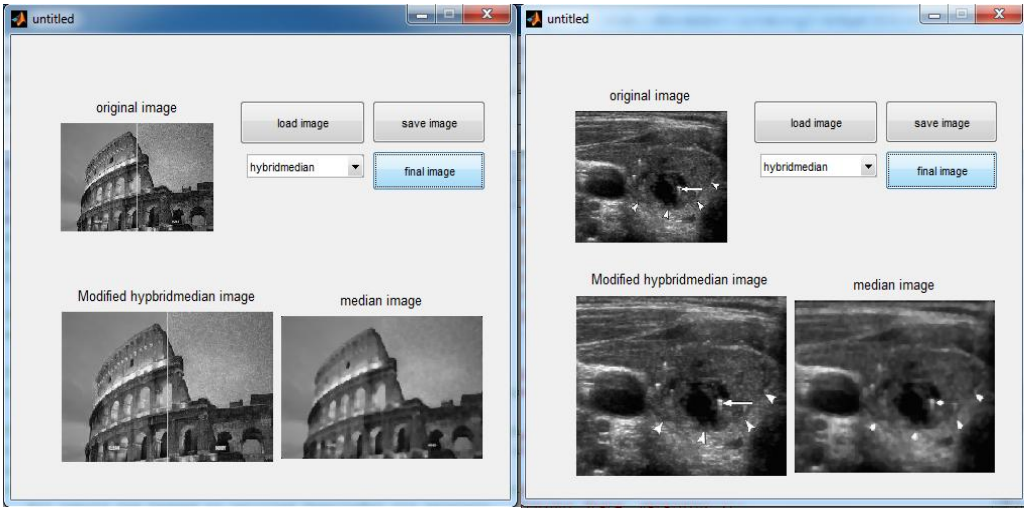
3.2.2 Lưu đồ giải thuật lọc ảnh:



(a): lọc ảnh nhiễu thật (b): lọc ảnh nhiễu giả

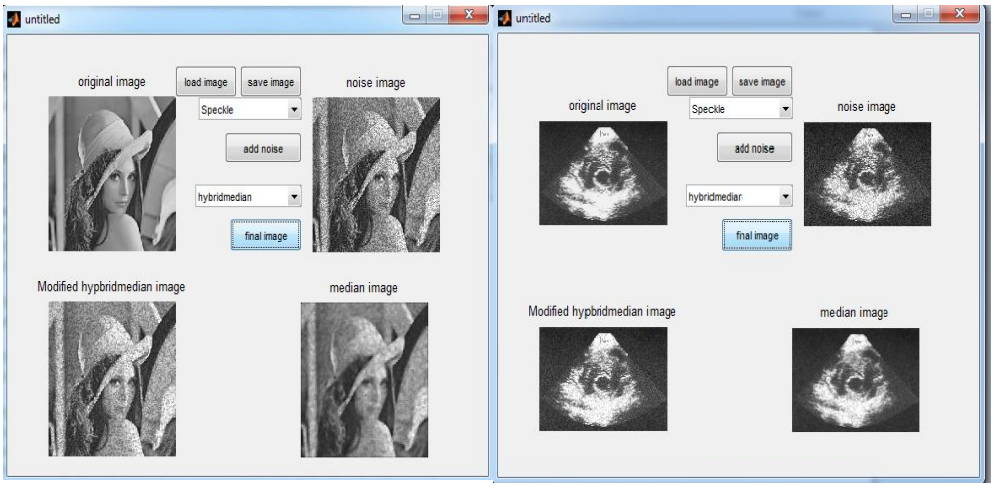
Hình 3.9: Lưu đồ lọc ảnh sử dụng Matlab

Trên hình 3.9a tác giả tiến hành load ảnh gốc cần lọc; sau đó tiến hành kiểm tra kích thước ảnh nhỏ hơn hoặc bằng 192x192; tiếp theo tiến hành chọn bộ lọc và hiển thị kết quả sử dụng một trong hai bộ lọc Median và Modified hybrid median; kết thúc quá trình lọc nhiễu. Và quá trình được lặp lại khi ta chọn bộ lọc khác.



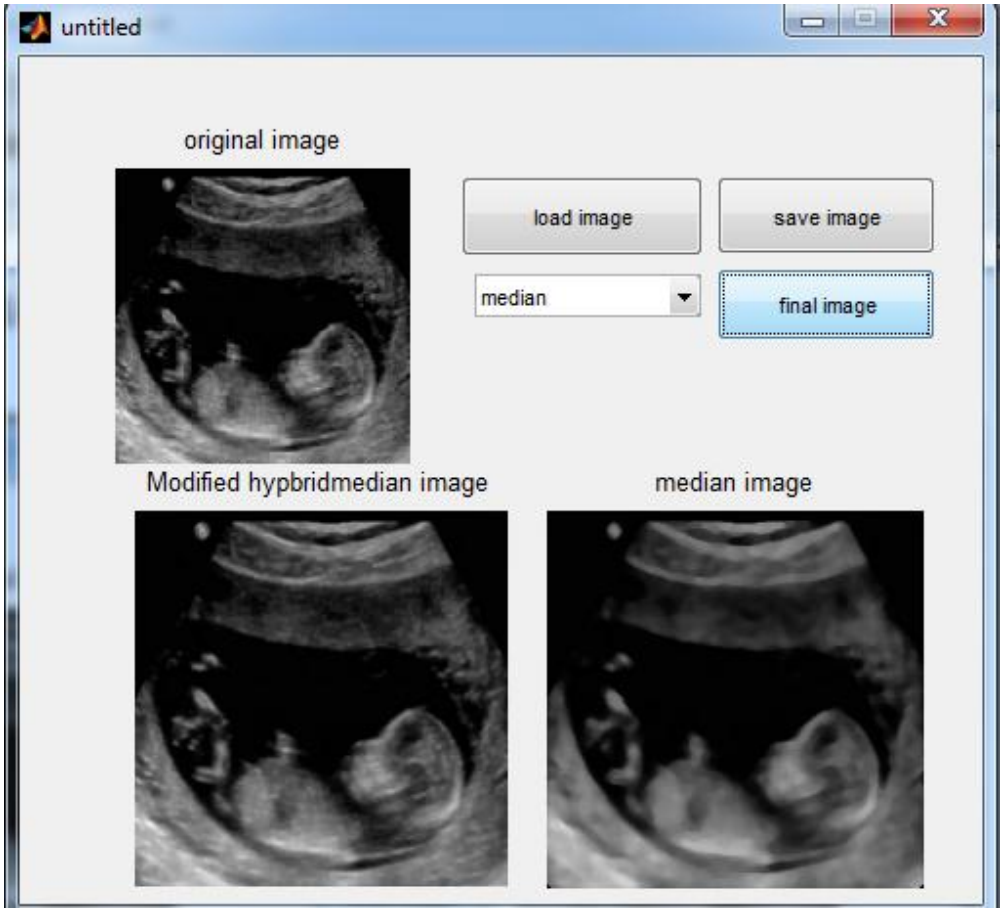
Hình 3.10: Lọc ảnh thông thường và ảnh siêu âm bị nhiễu đốm thật dùng Matlab

Trên hình 3.9b tác giả tiến hành load ảnh gốc cần lọc; sau đó tiến hành kiểm tra kích thước ảnh nhỏ hơn hoặc bằng 192x192 và cộng nhiễu giả định vào, nhiễu này ta thực hiện trên Matlab trong hai trường hợp nhiễu đốm (và nhiễu muối dùng để so sánh); cuối cùng tiến hành lọc là hiển thị kết quả sử dụng một trong hai bộ lọc Median và Modified hybrid median; kết thúc quá trình lọc nhiễu. Và quá trình được lặp lại khi ta chọn loại nhiễu khác.



Hình 3.11: Lọc ảnh thông thường và ảnh siêu âm bị nhiễu đốm giả dùng Matlab

3.3 Kết quả và thảo luận



Hình 3.12: Dùng Matlab lọc nhiễu đốm cho ảnh siêu âm

Bảng 3.2: Kết quả so sánh hai bộ lọc Median và Modified hybrid median cho ảnh dùng trong luận văn.

Filter	RMSE	SNR	PSNR	EPF
Median	5,8197	7,0333	79,9827	0,2728
Modified hybrid median	4,0404	10,2056	83,1522	0,9279

Ta dùng một số tiêu chí để đánh giá các bộ lọc:

SRN: Tỉ lệ tín hiệu gốc so với nhiễu

$$SNR = 10.\log_{10}\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{i,j}^2+y_{i,j}^2)}{\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{i,j}-y_{i,j})^2} \qquad (3.7 \equiv 3.1)$$

SRN : Median (7,0333), Modified hybrid median(10,2056)

SRN càng lớn thì chất lượng ảnh sau lọc càng tốt

RMSE: Căn bậc hai của sai số trung bình bình phương

$$RMSE = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(x_{i,j} - y_{i,j}\right)^2} \qquad (3.8 \equiv 3.2)$$

RMSE : Median(5,8197), Modified hybrid median(4,0404)

RMSE càng nhỏ thì chất lượng ảnh sau lọc càng tốt

PSNR: Tỉ lệ tín hiệu đỉnh trên nhiễu

$$PSNR = 20.\log_{10}(g_{max}^2 / RMSE) \qquad (3.9 \equiv 3.3)$$

PSNR : Median (79,9827), Modified hybrid median (83,1522)

PSNR càng cao thì kết quả lọc càng hiệu quả.

EPF : khả năng bảo quản cảnh hình ảnh

$$EPF = \frac{\sum(\Delta x - \overline{\Delta x})(\Delta y - \overline{\Delta y})}{\sqrt{\sum(\Delta x - \overline{\Delta x})^2 \sum(\Delta y - \overline{\Delta y})^2}} \qquad (3.10 \equiv 3.6)$$

EPF : Median (0,2728), Modified hybrid median (0,9279).

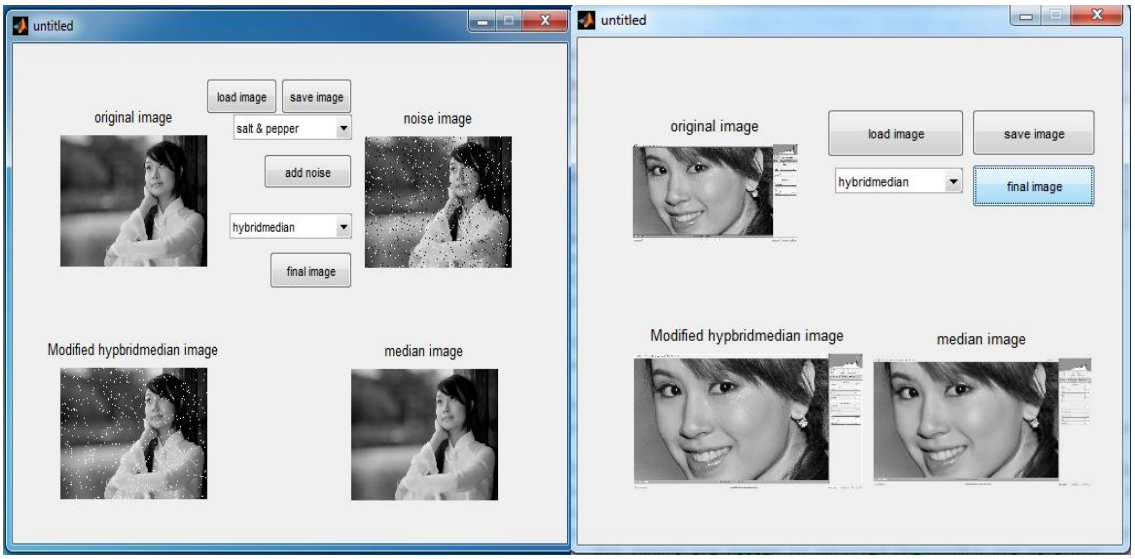
EPF càng lớn thì khả năng bảo quản cạnh cho ảnh càng tốt.

Như vậy với một số tiêu chuẩn so sánh tác giả nhận thấy bộ lọc Modified hybrid median cho kết quả tốt hơn bộ lọc Median khi lọc nhiễu đốm. Vì vậy, khi ảnh siêu âm gặp nhiễu đốm thì bộ lọc Modified hybrid median là một đề xuất tốt nhất vì loại bỏ được nhiễu, khả năng bảo quản cạnh của ảnh tốt, chất lượng ảnh giống với ảnh gốc: SNR : Median (7,0333), Modified hybrid median (10,2056) ; RMSE : Median (5,8197), Modified hybrid median (4,0404) ; PSNR : Median (79,9827), Modified hybrid median (83,1522)

Một vài kết quả so sánh hai bộ lọc Median và Modified Hybrid Median lọc nhiễu muối tiêu, nhiễu đốm với ảnh thông thường và ảnh siêu âm sử dụng Matlab:

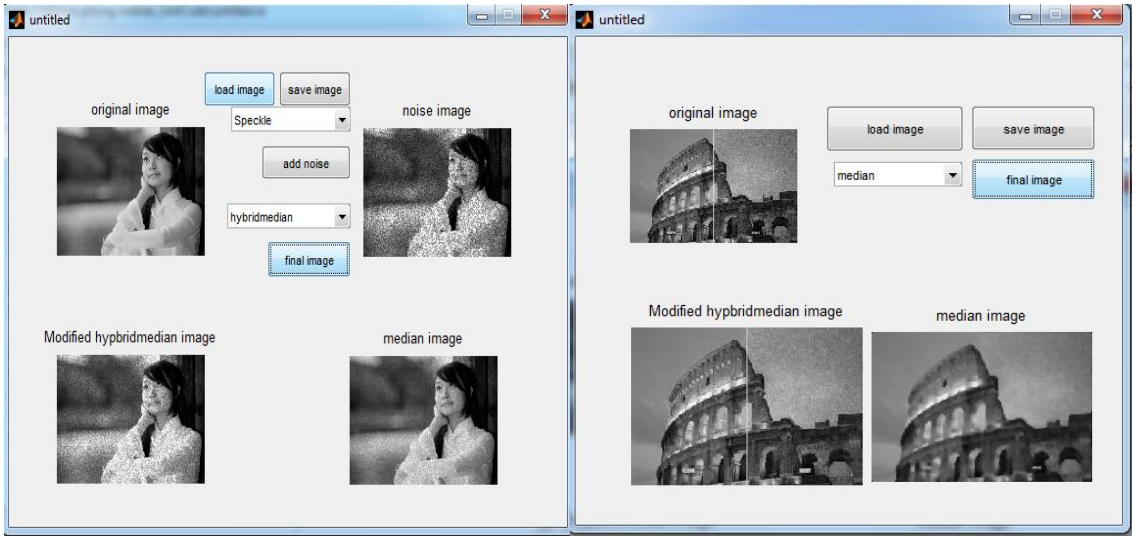
a. Lọc ảnh thông thường sử dụng Matlab: Kết quả cho thấy với nhiễu muối tiêu thì bộ lọc Median lọc mịn hơn – loại bỏ gần như hoàn toàn các nhiễu có trên ảnh (RMSE = 6.7612), khả năng bảo quản cạnh ảnh cũng tốt hơn (EPF = 0.9235), cho kết quả giống với ảnh gốc (SSI = 1.0000) hơn so với bộ lọc

Modified hybrid median (RMSE = 12,0251; EPF = 0,2729; SSI = 0,7627) ⇒ Đối với ảnh thông thường bị nhiễu muối tiêu chúng ta nên dùng bộ lọc Median.



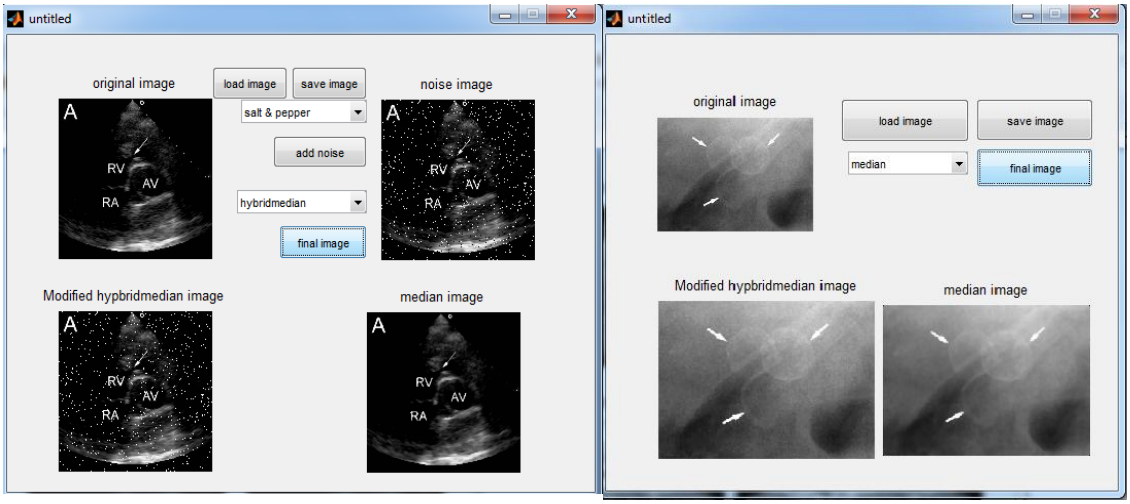
Hình 3.13: Dùng Matlab lọc nhiễu muối tiêu cho ảnh thông thường (ảnh nhiễu giả và ảnh nhiễu thật)

Còn đối với ảnh bị nhiễu đốm bộ lọc Modified hybrid median bảo quản cạnh tốt (EPF = 0,9351), loại bỏ nhiễu khá rõ (RMSE = 9,3736), ảnh có kết cấu hình ảnh giống với ảnh gốc hơn SSI = 1,0000) so với bộ lọc Median (RMSE= 12,2093; EPF= 0,3728; SSI = 0,8349) ⇒ Đối với ảnh thông thường khi bị nhiễu đốm chúng ta nên dùng bộ lọc Modified hybrid median sẽ lọc hiệu quả hơn.



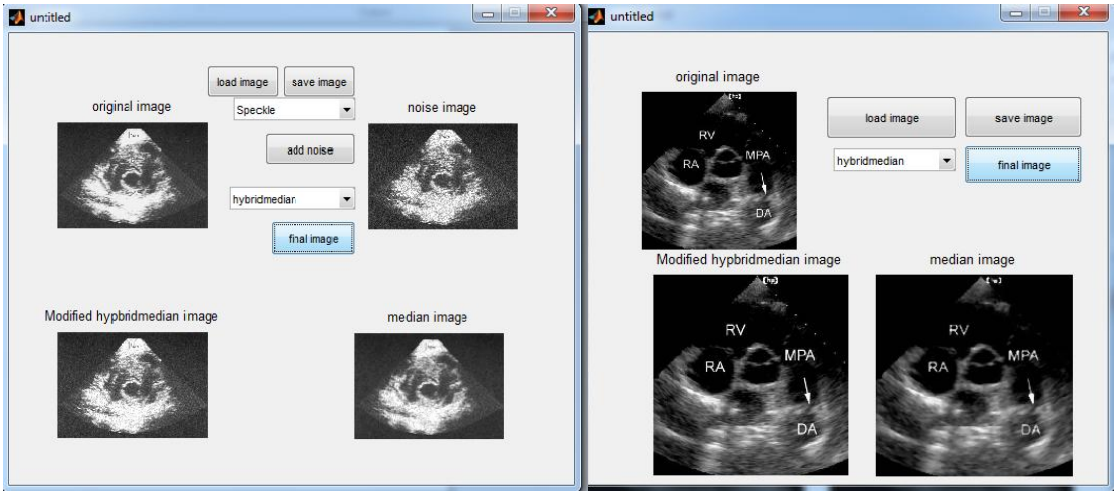
Hình 3.14: Dùng Matlab lọc nhiễu đốm cho ảnh thông thường
(ảnh nhiễu giả và ảnh nhiễu thật)

b. Lọc ảnh siêu âm sử dụng Matlab: Kết quả cho thấy với nhiễu muối tiêu thì bộ lọc Median lọc mịn hơn – loại bỏ gần như hoàn toàn các nhiễu có trên ảnh (RMSE = 6,2431), khả năng bảo quản cạnh ảnh tốt (EPF = 0,9374), cho kết quả giống với kết cấu ảnh gốc (SSI = 0,9998) hơn so với bộ lọc Modified hybrid median (RMSE = 11,0573; EPF = 0,6321; SSI = 0,8794) ⇒ Đối với ảnh siêu âm bị nhiễu muối tiêu chúng ta nên dùng bộ lọc Median.



Hình 3.15: Dùng Matlab lọc nhiễu muối tiêu cho ảnh siêu âm
(ảnh nhiễu giả và ảnh nhiễu thật)

Còn đối với ảnh siêu âm bị nhiễu đốm bộ lọc Median lọc được các đốm nhiễu (RMSE = 11,9273), song khả năng bảo quản cạnh của ảnh (EPF = 0,5254) và kết cấu hình ảnh không được tốt (SSI = 0,8936); còn bộ lọc Modified hybrid median bảo quản cạnh tốt hơn (EPF = 0,9279), loại bỏ hoàn toàn nhiễu đốm (RMSE = 6,1556), cho kết quả giống với ảnh gốc (SSI = 1,0000) hơn so với bộ lọc Median ⇒ Đối với ảnh siêu âm khi bị nhiễu đốm chúng ta nên dùng bộ lọc Modified hydrid median sẽ lọc hiệu quả hơn.



Hình 3.16: Dùng Matlab lọc nhiễu đốm cho ảnh siêu âm
(ảnh nhiễu giả và ảnh nhiễu thật)

Tóm lại qua cuộc khảo sát này tác giả nhận thấy rằng đối với ảnh siêu âm bị nhiễu đốm nên dùng bộ lọc Modified hybrid median lọc sẽ hiệu quả hơn (Hình 3.12 : SRN = 10,2056 ; RMSE = 4,0404 ; PSNR = 83,1522 ; EPF = 0,9279), giảm nhiễu đáng kể mang lại kết quả chính xác cao trong khi chẩn đoán kết quả của các Bác sĩ. Tuy nhiên, quá trình xử lý trên Matlab chỉ là mô phỏng chưa đưa thành chương trình sử dụng trong thực tế, việc thay đổi các thông số các bộ lọc chưa linh hoạt, …để khắc phục một số nhược điểm trên tác giả tiến hành thực hiện lọc nhiễu đốm cho ảnh siêu trên FPGA.

Chương 4:

THỰC HIỆN LỌC NHIỄU ĐỐM CHO ẢNH SIÊU ÂM TRÊN FPGA

Trong chương này tác giả dùng phần Quartus II 12.0sp2 Web Edition (32 – bit) để lập trình ngôn ngữ Verilog HDL lọc nhiễu đốm trên FPGA, giao tiếp SRAM, VGA với kit DE2.

4.1 Yêu cầu

Tìm hiểu kỹ thuật lọc nhiễu đốm cho ảnh siêu âm.

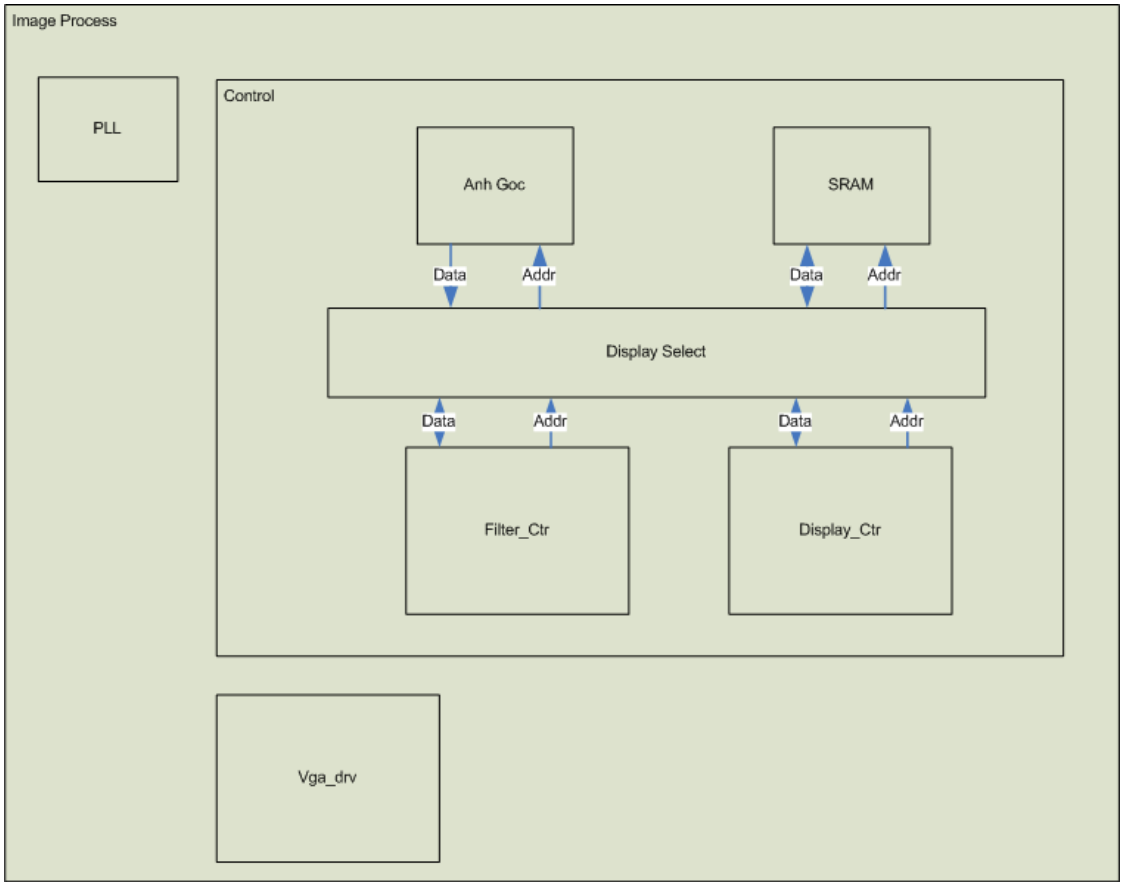
Nghiên cứu giải thuật lọc Median và Modified hybrid median.

Thực hiện phần cứng trên FPGA.

Nội dung thực hiện:

- Truyền ảnh xuống kit DE2 của Altera qua cổng Blaster, đồng thời lập trình xử lý ảnh để có kết quả so sánh với phần cứng.
- Sử dụng phần mềm Quartus II của Altera để viết chương trình bằng ngôn ngữ Verilog HDL và giao tiếp với kit DE2.
- Giao tiếp với VGA, SRAM trên kit DE2.
- Hiển thị kết quả lên monitor: ảnh gốc, ảnh sau khi lọc sử dụng bộ lọc Median và Modified hybrid median trên cùng màn hình

4.2 Sơ đồ thiết kế



Hình 4.1: Sơ đồ khối Image Process

- PLL cung cấp các clock đa dụng cũng như cung cấp các chức năng như:

  + Nhân và chia clock.

  + Dịch pha.

  + Chu kỳ nhiệm vụ (duty cycle) lập trình được.

  + Tạo ra 3 clock nội khác nhau.

  + Tạo ra 1 clock ngoại.

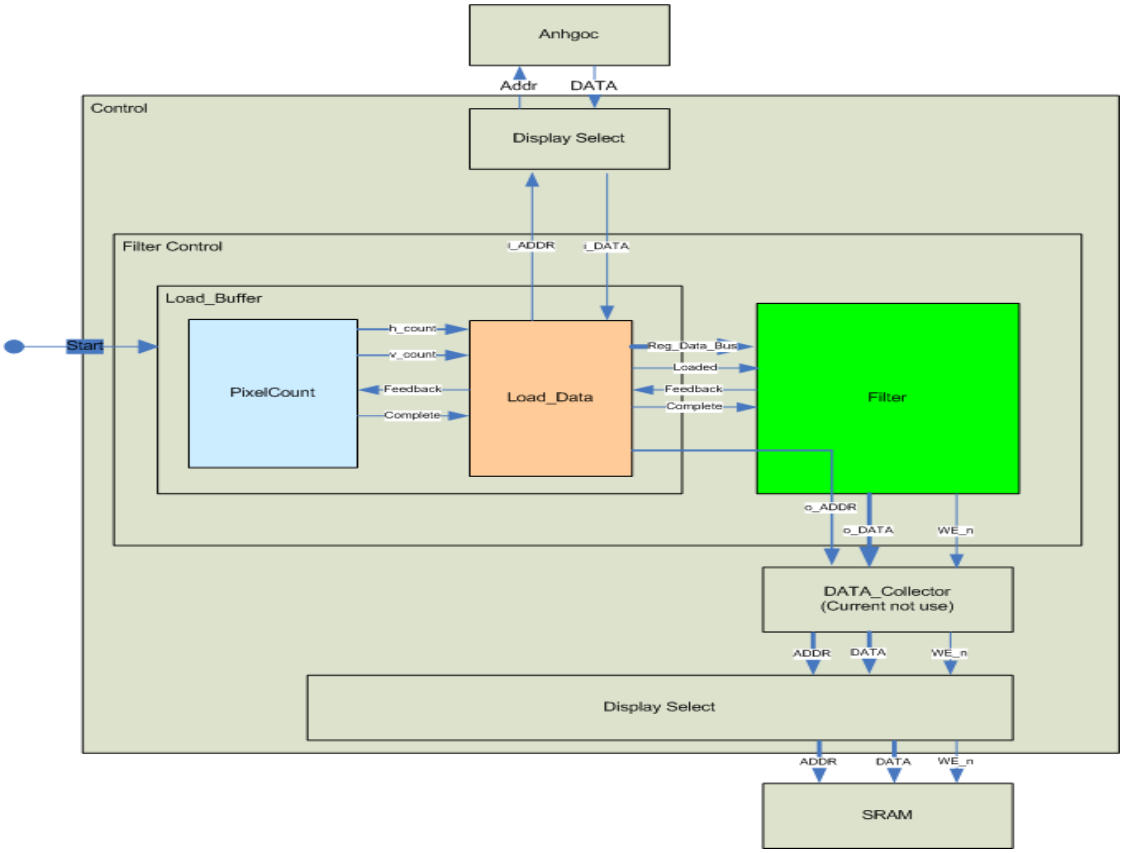
  + Các ngõ ra clock cho các hỗ trợ IO khác nhau.

  + Các tín hiệu điều khiển…

- Ảnh gốc sẽ được lưu trong RAM nội của FPGA

- Ảnh được xử lý bởi "Filtter_Ctr"

- Ảnh sau xử lý sẽ được lưu trong SRAM
- "Display_Ctr" tính toán địa chỉ để lấy ảnh gốc và ảnh đã xử lý để hiển thị lên VGA.
- Vì cả RAM và SRAM đều được dùng trong quá trình xử lý và hiển thị. Nên ta cần tín hiệu Display_Select để điều khiển luồng của bus dữ liệu
- VGA_Ctr để đồng bộ dữ liệu giữa khối Display_Ctr với chip VGA (Điều khiển các tín hiệu xuất ra VGA chip)



Hình 4.2: Sơ đồ khối Control

4.2.1 Nhiệm vụ khối PixelCount:

Đưa ra vị trí điểm ảnh cần xử lý

```
for (v_count = 0; v_count <192; v_count++)
{
```

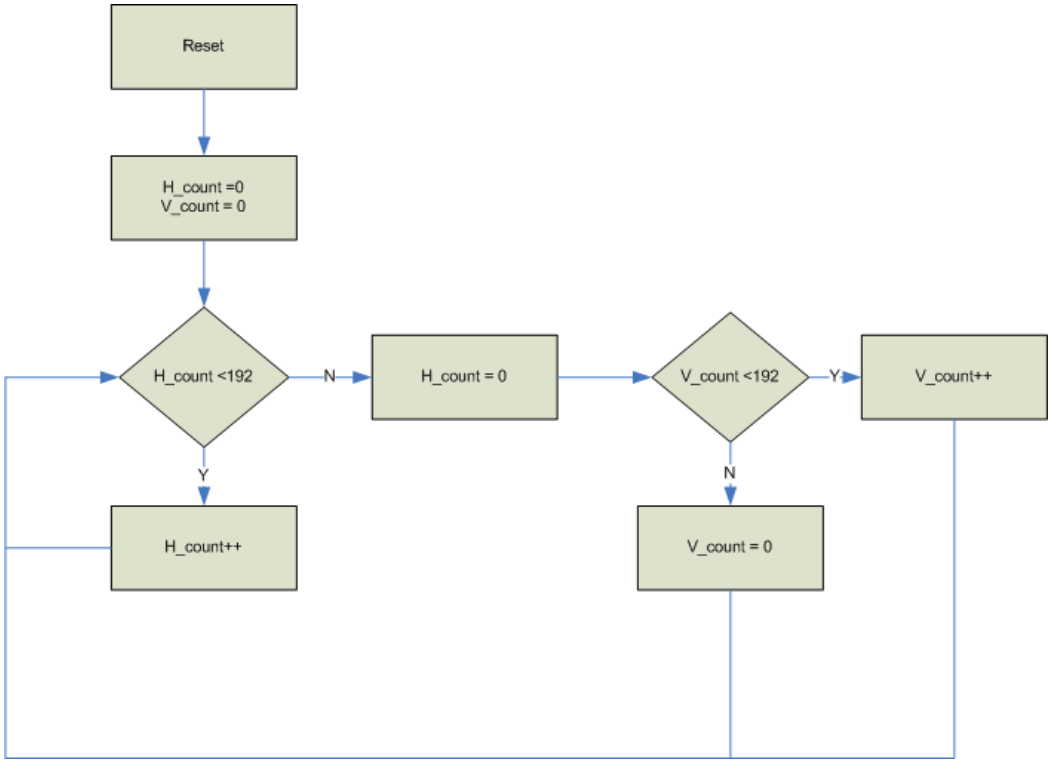
```
 for(h_count = 0; h_count <192; h_count++)
{
 }
}
```

Sau khi Reset, khối PixelCount được đợi tín hiệu "Start" để bắt đầu đếm

Mỗi lần cập nhật h_count, v_count, tín hiệu "CountDone" lên 1 để thông báo cho khối LoadBuffer bắt đầu hoạt động. Và sẽ chờ tín hiệu Feedback.

Mỗi khi có tín hiệu "Feedback = 1" sẽ kiểm tra và tăng h_count. Đồng thời kiểm tra và tăng v_count.

Tín hiệu "Feedback" do khối "LoadBufffer" báo về để báo và đã xử lý xong cho điểm ảnh cần xử lý.



Hình 4.3: Lưu đồ giải thuật khối PixelCount

4.2.2 Nhiệm vụ khối Load Data:

- Tính địa chỉ của các điểm ảnh cần xử lý và 8 điểm ảnh xung quanh

Với mỗi giá trị (h_count, v_count) được đưa ra bởi khối PixcelCount. Ta sẽ tính toán ra các giá trị của địa chỉ của các điểm ảnh xung quanh.

(h_count -1,v_count -1) (h_count ,v_count -1) (h_count +1,v_count -1)

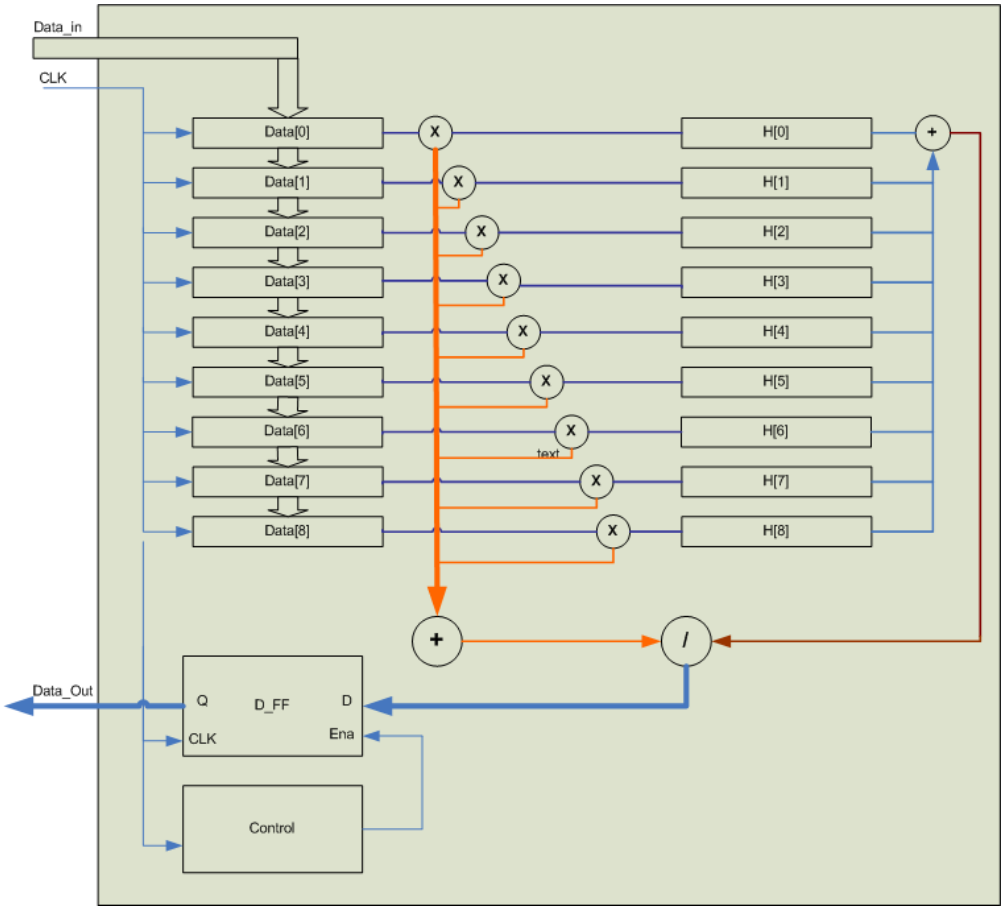
(h_count -1,v_count) (h_count,v_count) (h_count +1,v_count)

(h_count -1,v_count +1) (h_count,v_count +1) (h_count +1,v_count -1)

- Đưa địa chỉ cho khối "RAM".
- Nhận dữ liệu từ khối RAM store vào tập thanh ghi dịch
- Sau khi đã store đủ các giá trị sẽ thông báo tín hiệu "Loaded" đến khối "Filter".
- Sau khi khối "Filter" xử lý xong sẽ báo về "Done"
- Khi nhận được "Done" LoadBuffer sẽ báo "Feedback" về cho PixelCount

4.2.3 Nhiệm vụ khối Filter:

- Tính toán giá trị điểm ảnh mới sau xử lý bằng cách nhân data với bộ thông số của mạch lọc.
- Ghi dữ liệu vào SRAM vào địa chỉ cần ghi



Hình 4.4: Khối Filter

- Data được lưu vào thanh ghi dịch Data [0] → Data [8]
- Nhân data với bộ thông số H
- Tính tổng kết quả thu được
- Tính tổng của bộ thông số H
- Chia kết quả cho tổng của bộ thông số H
- Khối control dùng để điều khiển khi nào enable D-FF để đưa kết quả ra ngoài.
- Để có được mạch lọc Median hoặc Modified Hybrid Median. Ta chỉ cần thay đổi bộ thông số H [0→8]

4.2.4 Nhiệm vụ khối Data_Collector:

Vì dữ liệu sau Filter là 8bit và dữ liệu trong SRAM 16 bit nên khối Data_Collector được dùng để kết hợp 2 dữ liệu Filter (8bit) liên tiếp thành dữ liệu 16 bit. Khi có được dữ liệu 16 bit sẽ ghi vào SRAM

4.2.5 Nhiệm vụ khối Display_Select:

Đây chỉ là một tín hiệu. Sau khi xử lý xong pixel cuối cùng Display_Select sẽ được set lên 1

Khi Display_Select = 1, Khối VGA_Ctr và Dispay_Ctr sẽ chiếm bus data để giao tiếp với RAM và SRAM, hiển thị hình ảnh lên VGA

4.2.6 Nhiệm vụ khối VGA_Ctr:

Tạo các tín hiệu để đồng bộ data với VGA Chip, thông qua h_count, v_count

VGA_Ctr sẽ cung cấp h_count, v_count và nhận được giá trị RGB của điểm ảnh tương ứng

4.2.7 Nhiệm vụ khối Display_Ctr:

Nhận giá trị h_count, v_count từ VGA_Ctr

Tính toán để chọn lấy dữ liệu từ RAM hay SRAM

Tính toán giá trị địa chỉ

Đưa dữ liệu RGB cho VGA_Ctr

4.3 Giao tiếp với VGA và SRAM trên kit DE2

4.3.1 Chức năng và cách thiết kế khối VGA:

Tạo tín hiệu giao tiếp và điều khiển VGA chip. Tạo các tín hiệu đồng bộ cho VGA Chip (VGA_HS, VGA_VS). Cung cấp địa chỉ tương ứng của pixel

(VGA_ADDR) cho các khối bộ đệm dữ liệu (SRAM, RAM) để có thể đưa đúng dữ liệu R, G, B cho VGA chip.

Trong luận văn này tác giả sẽ hiển thị ảnh 24bit màu với kích thước 192*192. Do đó kích thước tối thiểu mà hệ thống cần để chứa dữ liệu hình ảnh hiển thị lên màn ảnh là:

24*192*192 = 884736bit = 110592 byte (1byte = 8bit) (4.1)

Bộ nhớ trên còn có một yêu cầu là trong 1 xung clock phải đưa đủ 24 bit cấp cho VGA. Để làm được như vậy ta cần phải sử dụng đồng thời 2 bộ nhớ.

Bộ nhớ có sẵn trong FPGA: M4K:Kích thước 40.000 * 8bit data out. Bộ nhớ này được dùng để chứa dữ liệu Red của hình ảnh

Và SRAM chip trên kit DE2: Kích thước 512 KByte = 256K * 16bit data. Được dùng để lưu dữ liệu Green và Blue của ảnh.

Cả hai bộ nhớ này được cung cấp cùng địa chỉ bởi VGA. Data out tổng cộng 16+8 là 24 bit.

Cách tính địa chỉ cho VGA_addr:

VGA_ADDR = (b – (V_count – y))*a + (H_count – x) (4.2)

Trong đó:

H_count: Biến đếm theo phương ngang. $0 \leq$ H_count $\leq 640 + 160$. Với 640 là kích thước theo phương ngang của toàn màn hình. Và 160 là độ trễ cần thiết để tín hiệu sóng quét ngang có thể trở về vị trí đầu hàng mới.

V_count: Biến theo phương dọc. $0 \leq$ V_count $\leq 480 + 45$. Với 480 là kích thước theo chiều dọc của toàn màn hình. Và 45 là độ trễ cần thiết để tín hiệu sóng quét ngang quay về vị trí ban đầu.

a: Kích thước ngang của ảnh.

b: Kích thước dọc của ảnh

x, y: tọa độ của ảnh được hiển thị

Trong đề tài này các thông số trên là:

a = b = 192;

x = 100;

y = 100;

Do đó ta có công thức sau:

VGA_addr = (192 - (v_count - 100))*192 + (h_count- 100) (4.3)

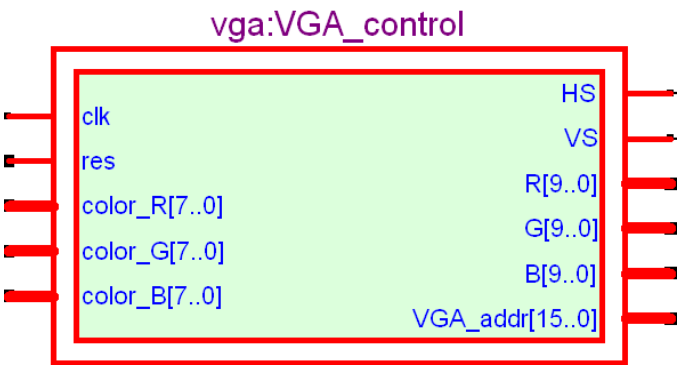
Vì ta chỉ cho hiển thị hình ảnh 192*192, một phần của màn hình 640*480. Nên ta cần tạo ra tín hiệu khi nào nhận dữ liệu từ các bộ đệm và khi nào không nhận tín hiệu từ bộ đệm.

```
if(h_count>=100 & h_count<292 & v_count >= 100 & v_count < 292)
    pic_en <= 1;
else pic_en <= 0;
assign color_R_t = pic_en ? color_R : 8'h00;
assign color_G_t = pic_en ? color_G : 8'h00;
assign color_B_t = pic_en ? color_B : 8'h00;
```

Các tín hiệu bắt đồng bộ HS và VS được tạo theo hướng dẫn của Altera:

```
if(h_count>=1318 & h_count<=1510) HS<=0;
                    else HS<=1;
if(v_count<=494 & v_count>=493) VS<=0;
                    else VS<=1;
```

4.3.2 Sơ đồ tổng hợp khối VGA:

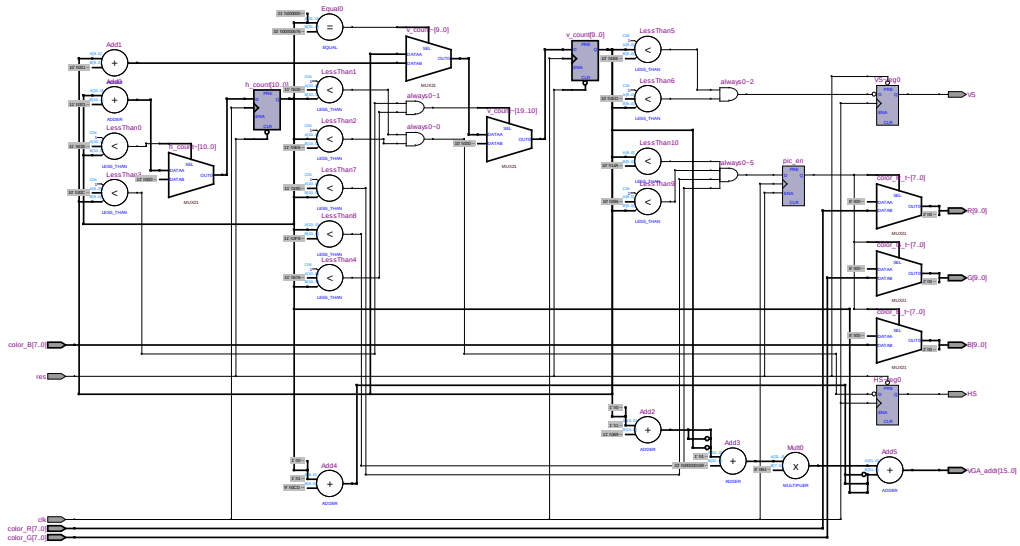


Hình 4.5: Khối VGA_control và các port giao tiếp

Dựa vào tín hiệu đếm theo hàng và cột để đồng bộ tín hiệu quét (sử dụng code trên trang web của Altera).

Tần số sử dụng trong khối VGA: 50Mhz

Tạo ra tín hiệu pic_en để quan sát khi nào VGA_Chip đang quét trong vùng hiển thị hình ảnh.



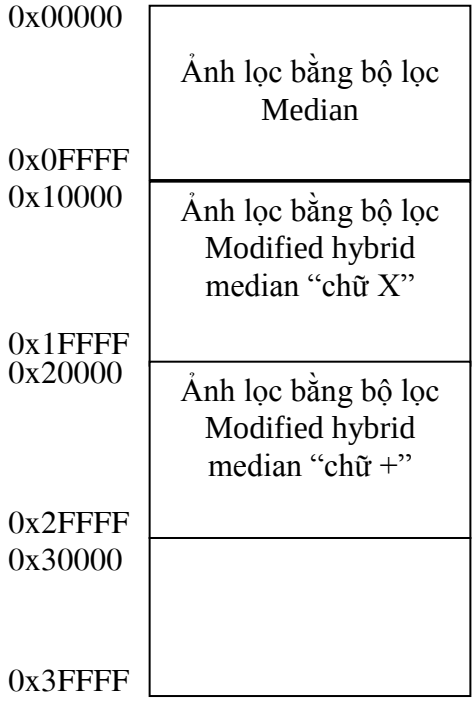
Hình 4.6: Sơ đồ khối VGA được tổng hợp bởi Quartus II

4.3.3 Nhiệm vụ của SRAM:

Nhiệm vụ của SRAM là chứa ảnh sau khi lọc được hiển thị.

SRAM phải được đọc ghi bởi CPU. VGA_control cũng phải đọc được SRAM. Để làm được điều này ta cần thực hiện ghép các tín hiệu điều khiển bằng các bộ chọn kênh.

Cấu trúc và phương pháp thực hiện được trình bày ở các mục dưới đây.



Hình 4.7: Các vùng trong bộ nhớ của SRAM

Bộ nhớ SRAM được chia làm 3 phần chính:

Phần 1: 0x00000 → 0x0FFFF

Có nhiệm vụ chứa hình ảnh sau khi lọc bằng bộ lọc Median được hiển thị.

Phần 2: 0x10000 → 0x1FFFF.

Có nhiệm vụ lưu trữ hình ảnh sau khi lọc bằng bộ lọc Modified hybrid median "chữ X".

Phần 3: 0x20000 → 0x3FFFF Có nhiệm vụ lưu trữ hình ảnh sau khi lọc bằng bộ lọc Modified hybrid median "chữ X".

Để hiển thị ảnh đã xử lý ta chỉ cần chép toàn bộ nội dung của phần 0x20000 → 0x2FFFF vào vùng 0x00000 → 0x0FFFF. Và chép dữ liệu của vùng 0x30000 → 0x3FFFF vào RAM. Sau đó cho DISPLAY = 1.

Để giao tiếp với SRAM chúng ta phải hiểu biết được các port tín hiệu, và các thức lập trình điều khiển các port này để phù hợp với giao thức giao tiếp của SRAM.

Các tín hiệu cần thiết để giao tiếp với SRAM:

SRAM_ADDR 17−0 – 18-bits, input.

SRAM_CE_N 1-bit, input.

SRAM_WE_N 1-bit, input.

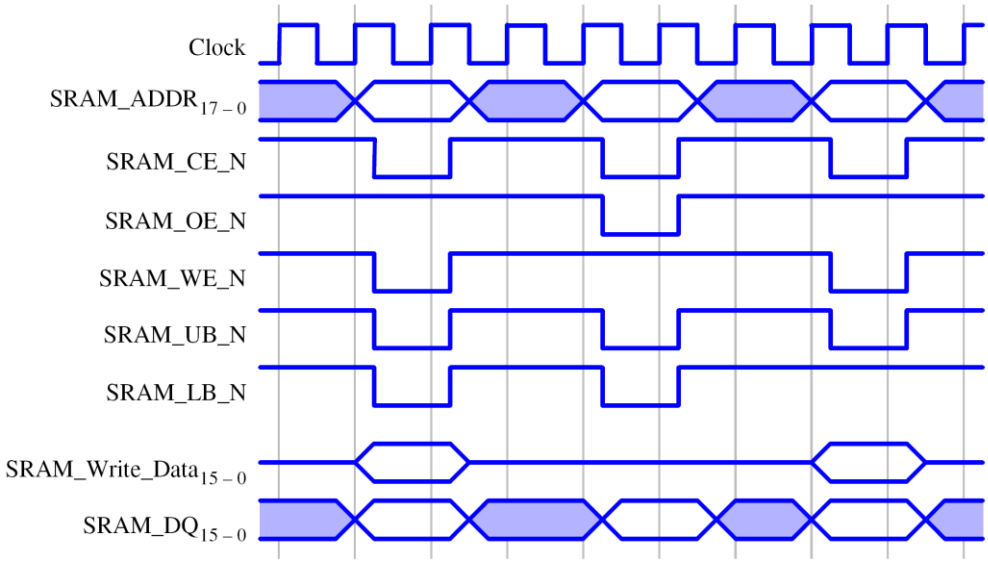
SRAM_OE_N 1-bit, input.

SRAM_UB_N 1-bit, input.

SRAM_LB_N 1-bit, input.

SRAM_DQ 16-bits, in_output

Hình dưới đây mô tả dạng sóng của tín các tín hiệu trong quá trình đọc và ghi SRAM:



Hình 4.8: Dạng sóng giao tiếp với SRAM

Để giao tiếp với SRAM ta phải tạo thêm các port PIO dưới đây:

output PIO	:	SRAM_ADDR	[17:0]
inout PIO	:	SRAM_DQ	[15:0]
output PIO	:	SRAM_WE_N	1 bit

Các tín hiệu còn lại sẽ cho tích cực, tức là được gán mức 0.

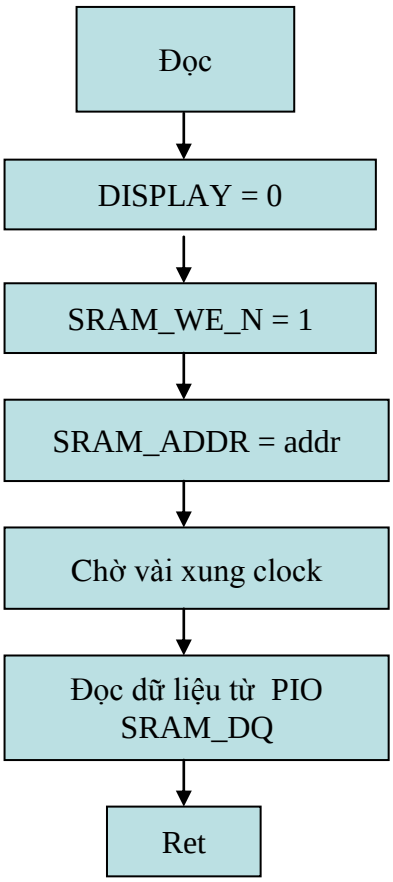
Cần chú ý một điều khi khởi tạo PIO SRAM_DQ để giao tiếp với SRAM_DQ của SRAM, ta khởi tạo là in_out port. Nhưng hệ thống thu được bao gồm một port in và một port out.

4.3.4. Lưu đồ giải thuật đọc và ghi SRAM:

Trước khi thực hiện giao tiếp với SRAM tác giả phải sử dụng lệnh IOWR(DISPLAY_BASE,0,0)

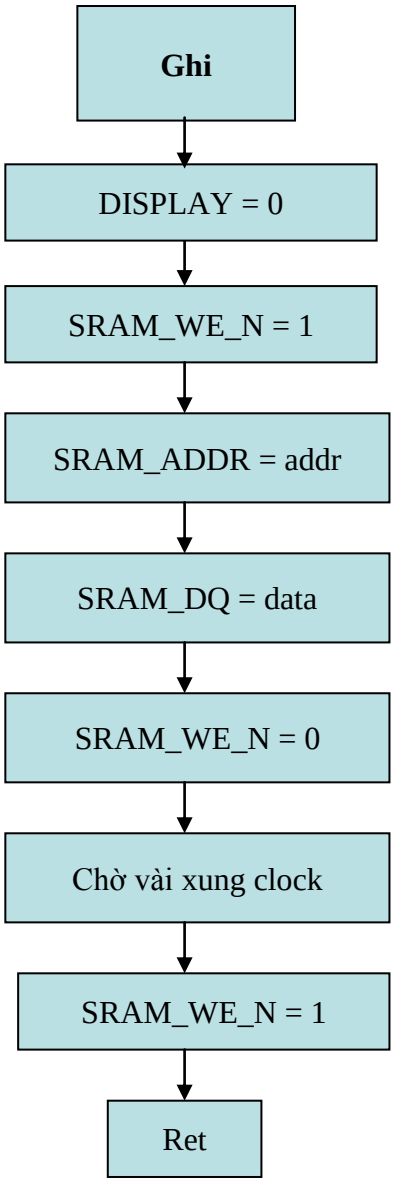
Để đưa tín hiệu DISPLAY lên 0. Điều này đã được giải thích ở phần trên.

4.3.4.1. Đọc dữ liệu từ SRAM:



Hình 4.9: Đọc SRAM

4.3.4.2. Ghi dữ liệu vào SRAM:



Hình 4.10: Ghi SRAM

4.4 Kết quả và thảo luận

4.4.1 Truyền ảnh từ PC xuống kit DE2:

Dùng phần mềm "Advanced Serial Port Terminal" với tốc độ baud 115200. Trong luận văn đã truyền được dữ liệu ảnh dưới dạng file Hex từ PC xuống kit DE2 và được lưu vào SRAM.

Dùng DE2 Control panel tác giả load dữ liệu sau khi truyền dữ liệu từ SRAM lên PC để đối chiếu với dữ liệu gốc.

goc

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000000000	01	00	7C	00	75	00	70	00	7A	00	80	00	93	00	8E	00
0000000010	92	00	8C	00	9D	00	A7	00	86	00	96	00	B6	00	9C	00
0000000020	C5	00	B2	00	A6	00	B0	00	AB	00	D1	00	BE	00	DE	00
0000000030	CF	00	AB	00	B0	00	AF	00	A6	00	B7	00	B4	00	99	00
0000000040	B3	00	B2	00	BC	00	9E	00	C9	00	BE	00	BD	00	BA	00
0000000050	D2	00	C9	00	D7	00	DD	00	B4	00	C5	00	B8	00	DA	00
0000000060	DD	00	BA	00	AB	00	A7	00	A0	00	A3	00	94	00	96	00
0000000070	83	00	90	00	85	00	58	00	59	00	60	00	69	00	60	00
0000000080	71	00	84	00	8A	00	85	00	65	00	68	00	6F	00	76	00
0000000090	79	00	87	00	76	00	74	00	6B	00	67	00	73	00	66	00
00000000A0	6D	00	5F	00	63	00	7F	00	7C	00	77	00	73	00	7E	00
00000000B0	71	00	65	00	72	00	99	00	86	00	71	00	71	00	69	00
00000000C0	72	00	74	00	8A	00	82	00	95	00	84	00	8D	00	9A	00
00000000D0	AB	00	A7	00	B0	00	8E	00	B4	00	A5	00	9F	00	A1	00
00000000E0	B3	00	9A	00	88	00	93	00	8A	00	83	00	81	00	7B	00
00000000F0	7F	00	6D	00	7C	00	73	00	8A	00	85	00	76	00	77	00
0000000100	74	00	5B	00	63	00	66	00	5D	00	61	00	5B	00	58	00
0000000110	72	00	67	00	5A	00	7B	00	6B	00	69	00	62	00	61	00
0000000120	5D	00	5F	00	64	00	70	00	5E	00	66	00	69	00	70	00
0000000130	6E	00	6B	00	62	00	65	00	61	00	54	00	63	00	5C	00
0000000140	59	00	57	00	5A	00	6E	00	6C	00	66	00	73	00	6C	00
0000000150	6A	00	76	00	87	00	8B	00	7F	00	73	00	86	00	86	00
0000000160	68	00	84	00	93	00	6E	00	8C	00	8D	00	8D	00	B5	00

Hình 4.11: Dữ liệu ảnh gốc

tu_sram

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000000000	01	00	7C	00	75	00	70	00	7A	00	80	00	93	00	8E	00
0000000010	92	00	8C	00	9D	00	A7	00	86	00	96	00	B6	00	9C	00
0000000020	C5	00	B2	00	A6	00	B0	00	AB	00	D1	00	BE	00	DE	00
0000000030	CF	00	AB	00	B0	00	AF	00	A6	00	B7	00	B4	00	99	00
0000000040	B3	00	B2	00	BC	00	9E	00	C9	00	BE	00	BD	00	BA	00
0000000050	D2	00	C9	00	D7	00	DD	00	B4	00	C5	00	B8	00	DA	00
0000000060	DD	00	BA	00	AB	00	A7	00	A0	00	A3	00	94	00	96	00
0000000070	83	00	90	00	85	00	58	00	59	00	60	00	69	00	60	00
0000000080	71	00	84	00	8A	00	85	00	65	00	68	00	6F	00	76	00
0000000090	79	00	87	00	76	00	74	00	6B	00	67	00	73	00	66	00
00000000A0	6D	00	5F	00	63	00	7F	00	7C	00	77	00	73	00	7E	00
00000000B0	71	00	65	00	72	00	99	00	86	00	71	00	71	00	69	00
00000000C0	72	00	74	00	8A	00	82	00	95	00	84	00	8D	00	9A	00
00000000D0	AB	00	A7	00	B0	00	8E	00	B4	00	A5	00	9F	00	A1	00
00000000E0	B3	00	9A	00	88	00	93	00	8A	00	83	00	81	00	7B	00
00000000F0	7F	00	6D	00	7C	00	73	00	8A	00	85	00	76	00	77	00
0000000100	74	00	5B	00	63	00	66	00	5D	00	61	00	5B	00	58	00
0000000110	72	00	67	00	5A	00	7B	00	6B	00	69	00	62	00	61	00
0000000120	5D	00	5F	00	64	00	70	00	5E	00	66	00	69	00	70	00
0000000130	6E	00	6B	00	62	00	65	00	61	00	54	00	63	00	5C	00
0000000140	59	00	57	00	5A	00	6E	00	6C	00	66	00	73	00	6C	00
0000000150	6A	00	76	00	87	00	8B	00	7F	00	73	00	86	00	86	00
0000000160	68	00	84	00	93	00	6E	00	8C	00	8D	00	8D	00	B5	00

Hình 4.12: Dữ liệu lấy từ SRAM

Sau khi kiểm tra đối chiếu 2 bảng dữ liệu tác giả nhận thấy là giống nhau, từ đó chứng minh được là quá trình truyền và lưu dữ liệu là đúng.

4.4.2 Hiển thị ảnh gốc lên monitor thông qua cổng VGA của kit DE2:

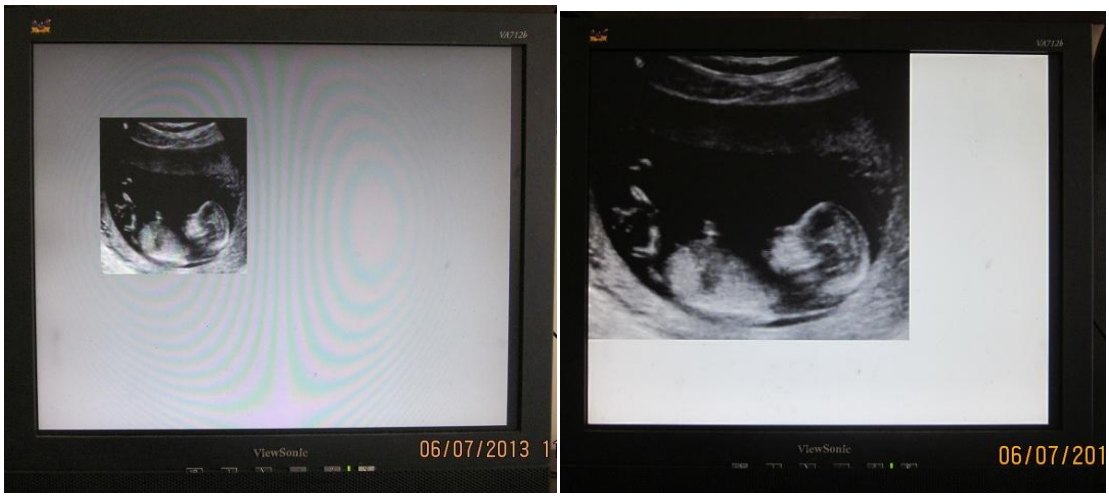
Việc giao tiếp đọc, ghi SRAM có thể dùng chương trình của Altera thực hiện và xuất ảnh từ SRAM ra VGA.

Với ảnh 24 bit màu thì phần dữ liệu bắt đầu từ địa chỉ 0x36. Để tách phần dữ liệu này ra tác giả sử dụng chương trình Note_pad++ để cắt phần header bắt đầu từ địa chỉ 0x00 → 0x35. Và lưu lại file này với đuôi *.BIN:

Lưu ý để xem file ở dạng binary ta phải add thêm hex_editor plugin cho Note_pad++. Dưới đây là các hướng dẫn cắt header:

Chuẩn bị file ảnh có kích thước 192*192* 24bit

Right_click vào file ảnh và chọn edit with note_pad++.



(a) (b)

Hình 4.13: Ảnh gốc được xuất ra từ cổng VGA của kit DE2 lên màn hình CRT

Từ hình 4.13 cho thấy đã xuất được ảnh lên màn hình CRT thông qua cổng VGA của kit DE2 với hai kích thước khác nhau: 192x192 (Hình 4.13a) và 640x480 (Hình 4.13b).

4.4.3 Thiết kế bộ lọc:

Thiết kế bộ lọc ma trận ảnh bằng Verilog, sau đó với mô phỏng Simulation kiểm tra bộ lọc vừa tạo được bằng cách ta cho một số dữ liệu đầu vào của bộ lọc sau đó cho chạy mô phỏng kiểm tra kết quả bằng tính toán cụ thể với

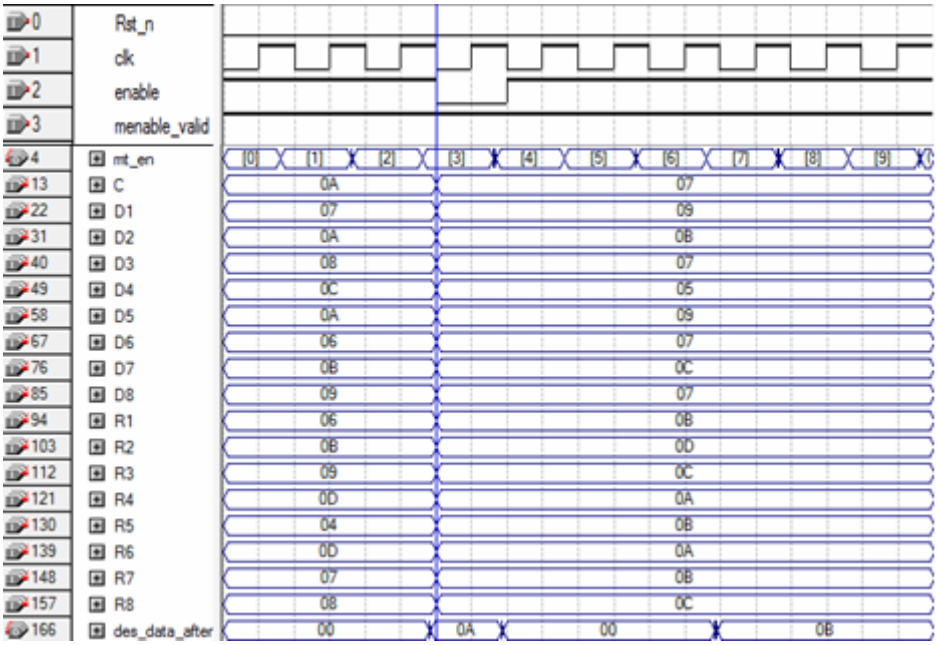
kết quả thu được từ mô phỏng để kiểm nghiệm lại bộ lọc có thực hiện đúng giải thuật hay không.

Với giải thuật bộ lọc:

$$\begin{pmatrix} D & * & R & * & D \\ * & D & R & D & * \\ R & R & C & R & R \\ * & D & R & D & * \\ D & * & R & * & D \end{pmatrix}$$

- MR= {các pixels R, C}
- MXD= {các pixels D, C}
- M= {MR, MXD, C}

Ta cho các giá trị cụ thể như : D1 có giá trị : 07, D2 (0A), D3 (08), D4 0C), D5 (0A), D6 (06), D7 (0B), D8 (09), R1 (06), R2 (0B), R3 (09), R4 (0D), R5 (04), R6 (0D), R7 (07), R8 (08), C (0A). Ta tính toán kết quả được MR = 08, MXD= 0D, M = {MR, MXD, C} = 0A. So sánh với kết quả từ mô phỏng như trong hình tác giả thấy kết quả giống nhau. Ta có thể tiến hành kiểm tra thêm một vài giá trị khác, từ đó ta thấy giải thuật viết là đúng.



Hình 4.14: Dạng sóng mô tả bộ lọc Modified hybrid median

Với mô phỏng kiểm tra trên hình 4.14 cho thấy sau khoảng 4 clock thì pixel trung tâm mới được xử lý xong. Với ảnh 192*192 ta có 36864 pixels, vậy sẽ mất 147456 clock.

Vậy mất khoảng 3ms để xử lý xong ảnh → nhanh.

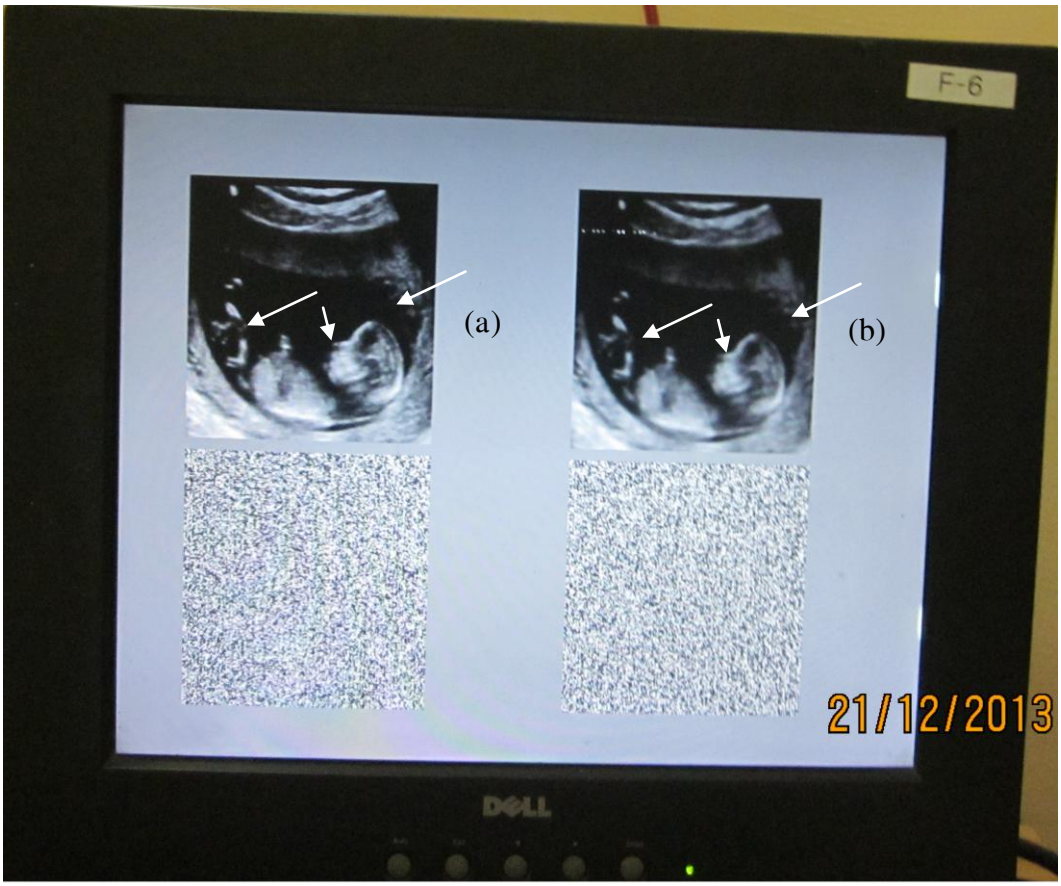
4.4.4 Lọc ảnh và xuất lên màn hình thông qua cổng VGA của kit DE2:

Trên FPGA tác giả chia SRAM thành bốn vùng riêng biệt kết hợp với các switch (SW) trên kit DE2: vùng thứ nhất lưu ảnh lọc sử dụng bộ lọc Median (SW ở vị trí 00), vùng thứ hai lưu ảnh lọc sử dụng bộ lọc Modified hybrid median hình "chữ X" (SW ở vị trí 10) và vùng thứ ba lưu ảnh lọc sử dụng bộ lọc Modified hybrid median hình "chữ +" (SW ở vị trí 01).

Hình 4.15b ảnh lọc sử dụng bộ lọc Median ta thấy nhiễu đốm vẫn còn xuất hiện ở một vài điểm ảnh, do bộ lọc khi thực hiện chỉ quét tìm pixel trung tâm còn các pixel xung quanh rất dễ bị bỏ sót; khả năng bảo quản cạnh đối với trường hợp này cũng thấp, biên không rõ ràng. Do thông số của bộ lọc được lấy trung bình giữa các pixel ảnh theo:

```
filter_matrix[0] <= 8'h01;
```

```
filter_matrix[1] <= 8'h01;
filter_matrix[2] <= 8'h01;
filter_matrix[3] <= 8'h01;
filter_matrix[4] <= 8'h01;
filter_matrix[5] <= 8'h01;
filter_matrix[6] <= 8'h01;
filter_matrix[7] <= 8'h01;
filter_matrix[8] <= 8'h01;
div_var <= 9;
```



Hình 4.15: Ảnh gốc (a) và ảnh sau lọc bằng bộ lọc Median (b); các mũi tên chỉ các điểm nhiễu đốm

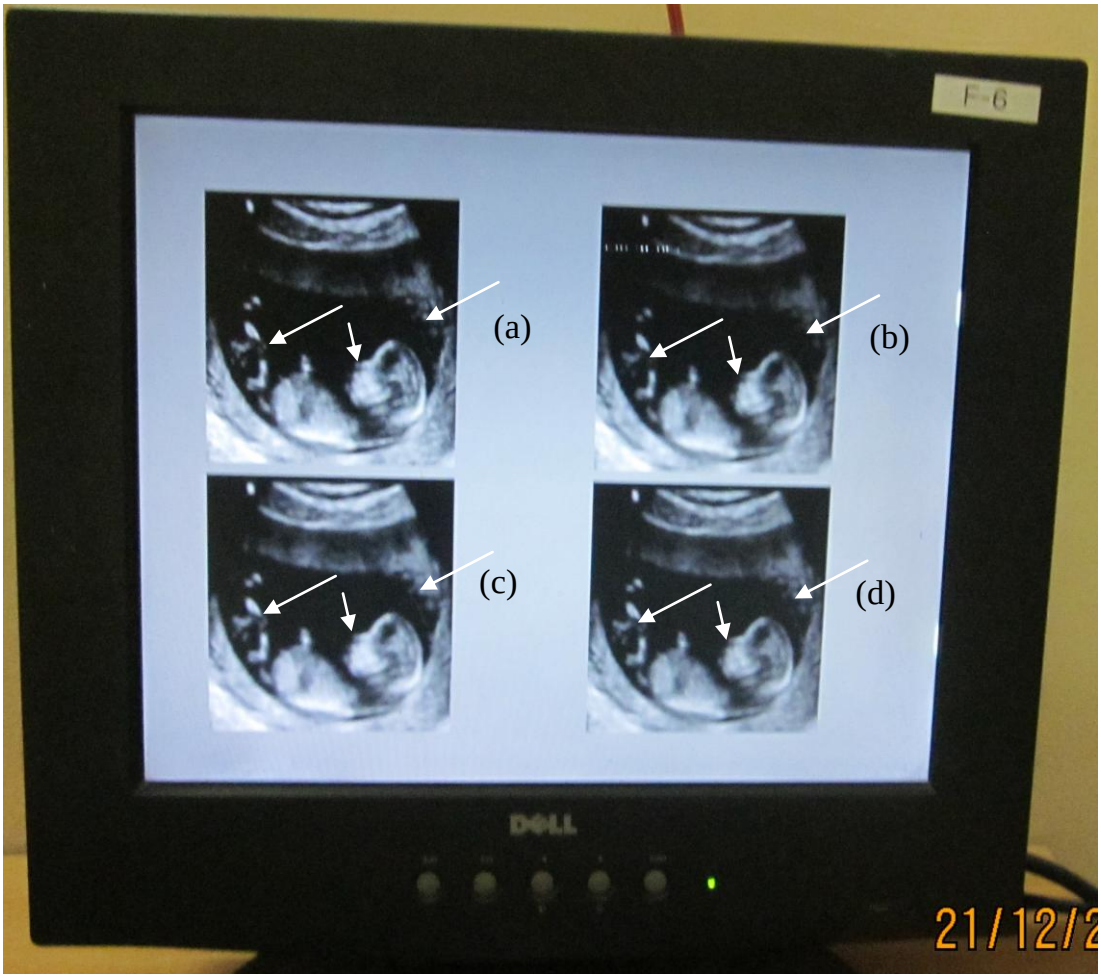
Để khắc phục tình trạng này tác giả tiến hành lọc với bộ lọc Modified hybrid median (Hình 4.16).

Hình 4.16c và hình 4.16d là ảnh lọc sử dụng bộ lọc Modified hybrid median ta thấy nhiễu đốm đã được loại bỏ, do bộ lọc khi thực hiện quét theo hình chữ X hoặc chữ thập nên khả năng tìm nhiễu chính xác hơn; khả năng bảo quản cạnh đối với trường hợp này cũng tốt hơn, các đường biên của chi tiết ảnh rõ ràng hơn. Do thông số của bộ lọc được lấy đối xứng của ma trận ảnh theo:

```
filter_matrix[0] <= 8'h01;
filter_matrix[1] <= 8'h00;
filter_matrix[2] <= 8'h01;
filter_matrix[3] <= 8'h00;
filter_matrix[4] <= 8'h01;
filter_matrix[5] <= 8'h00;
filter_matrix[6] <= 8'h01;
filter_matrix[7] <= 8'h00;
filter_matrix[8] <= 8'h01;
div_var <= 5;
```

Hoặc:

```
filter_matrix[0] <= 8'h00;
filter_matrix[1] <= 8'h01;
filter_matrix[2] <= 8'h00;
filter_matrix[3] <= 8'h01;
filter_matrix[4] <= 8'h01;
filter_matrix[5] <= 8'h01;
filter_matrix[6] <= 8'h00;
filter_matrix[7] <= 8'h01;
filter_matrix[8] <= 8'h00;
div_var <= 5;
```



Hình 4.16: Ảnh gốc (a) và ảnh sau lọc bằng bộ lọc bằng Median (b), bộ lọc Modified hybrid median hình "chữ X" (c) và bộ lọc Modified hybrid median hình "chữ +" (d); các mũi tên chỉ các điểm nhiễu đốm

Qua quá trình thực hiện lọc nhiễu đốm cho ảnh siêu âm trên FPGA tác giả nhận thấy rằng kết quả thu được khá tốt, đáp ứng được việc chúng ta có thể thay đổi các thông số bộ lọc một cách linh hoạt (các thông số trên bộ lọc có thể thay đổi tùy ý, nhưng cũng trong phạm vi thông số từng bộ lọc và tùy vào yêu cầu của người dùng), chương trình ứng dụng thời gian thực (realtime) có thể ứng dụng được trong thực tế hơn so với lọc trên Matlab chỉ có thể mô phỏng.

Chương 5
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI

Mục đích của luận văn này là tác giả tiến hành thực hiện lọc nhiễu đốm cho ảnh siêu âm trên FPGA thông qua việc tìm hiểu kỹ thuật lọc nhiễu đốm cho ảnh siêu âm, nghiên cứu giải thuật lọc Median và Modified hybrid median, đồng thời thực hiện phần cứng trên FPGA: Truyền ảnh xuống kit DE2 của Altera qua cổng Blaster, đồng thời sử dụng phần mềm Quartus II của Altera để viết chương trình bằng ngôn ngữ Verilog HDL và giao tiếp với kit DE2, giao tiếp với VGA, SRAM trên kit DE2. Hiển thị kết quả lên monitor: ảnh gốc, ảnh sau khi lọc sử dụng bộ lọc Median và Modified hybrid median trên cùng màn hình.

5.1 Kết quả

Tác giả tập trung nghiên cứu các kỹ thuật lọc nhiễu, đặc biệt là nhiễu đốm. Đã tiến hành lọc và loại bỏ nhiễu đốm cho ảnh siêu âm bằng hai bộ lọc Median và Modified hybrid median sử dụng Matlab (có lọc ảnh bị nhiễu muối tiêu để so sánh) và lọc nhiễu đốm cho ảnh siêu âm trên FPGA. Cụ thể đạt được như sau:

- Nghiên cứu giải thuật và thiết kế bộ lọc Median và Modified hybrid median.

- Thực hiện lọc nhiễu đốm và nhiễu muối tiêu cho ảnh thông thường và ảnh siêu âm sử dụng Matlab trong hai trường hợp có nhiễu thật và ảnh nhiễu giả, so sánh các kết quả thu được: tác giả nhận thấy rằng đối với ảnh siêu âm bị nhiễu đốm nên dùng bộ lọc Modified hybrid median lọc sẽ hiệu quả hơn vì nhiễu giảm rất nhiều so với ảnh gốc, kết cấu ảnh giống với ảnh gốc hơn, khả năng bảo quản cạnh tốt (SRN = 10,2056 ; RMSE = 4,0404 ; PSNR = 83,1522 ; EPF = 0,9279) so với bộ lọc Median; Do đó, sẽ mang lại kết quả chính xác khi các Bác sĩ chẩn đoán trong siêu âm.

- Hiển thị được ảnh siêu âm gốc với hai kích thước 192x192 và 640x480.

- Hiển thị được ảnh gốc và ảnh sau khi lọc bằng hai bộ lọc Median và Modified hybrid median trên cùng màn hình: trên FPGA ta chia SRAM thành bốn vùng riêng biệt kết hợp với các switch (SW) trên kit DE2: vùng thứ nhất lưu ảnh lọc sử dụng bộ lọc Median (SW ở vị trí 00), vùng thứ hai lưu ảnh lọc sử dụng bộ lọc Modified hybrid median hình "chữ X" (SW ở vị trí 10)và vùng thứ ba lưu ảnh lọc sử dụng bộ lọc Modified hybrid median hình "chữ +" (SW ở vị trí 01). Cuối cùng tác giả cũng nhận thấy với ảnh siêu âm bị nhiễu đốm nên dùng bộ lọc Modified hybrid median khi thực hiện trên FPGA vì nó mang lại kết quả thực nghiệm, chúng ta có thể thay đổi chương trình dễ dàng, …

5.2 Hạn chế

Tuy nhiên vì nguồn ảnh siêu âm bị nhiễu đốm tìm được có giới hạn nên khi thực hiện lọc nhiễu đốm cho ảnh siêu trên FPGA học viên chỉ thực hiện trên một vài ảnh; và chỉ thực hiện trên ảnh tĩnh, ảnh đen trắng, kích thước ảnh lọc được còn nhỏ 192x192, …

5.3 Hướng phát triển

Trong thời gian tới tác giả sẽ tiến hành nghiên cứu thêm một số kỹ thuật lọc nhiễu, hợp tác đầy đủ và toàn diện với một bệnh viện để có được sự hỗ trợ cần thiết về hình ảnh nghiên cứu nhiễu, đặc biệt là nhiễu đốm và các máy siêu âm hiện giờ để mở rộng :

- Xử lý ảnh siêu âm màu, sử dụng bộ nhớ SDRAM.

- Xử lý ảnh siêu âm động (video), …

TÀI LIỆU THAM KHẢO

[1] Giáo sư Nguyễn Văn Ngọ, *Giáo trình Xử lý ảnh*

[2] Lương Mạnh Bá, Nguyễn Thanh Thủy, *Nhập môn Xử lý ảnh*, Nhà xuất bản Trường đại học Bách Khoa Hà Nội. 2006

[3] Nguyễn Quang Hoan, *Xử lý ảnh*, Học viện bưu chính viễn thông. 2006

[4] Nguyễn Kim Sách, Hàn Trung Định, *Khôi phục ảnh động bị nhiễu - Speckle, Blotch, Scratch*

[5] DE2 Reference Manual, Altera. 2005

[6] Alasdair McAndrew, *An Introduction to Digital Image Processing with Matlab*, Victoria University of Technology. 2004

[7] N. K. Ragesh1, A. R. Anil, Dr. R. RajeshDigital, *Image Denoising in Medical Ultrasound Images: A Survey*, ICGST AIML - 11 Conference, Dubai, UAE. 12-14 April 2011

[8] R.Vanithamani, G. Umamaheswari , M. Ezhilarasi, *Modified Hybrid Median Filter for Effective Speckle Reduction in Ultrasound Images*, Senior Lecturer, Department of ECE, PSG College of Technology, Coimbatore, Tamilnadu, India

[9] Pong P.Chu. *FPGA Prototyping by verilog examples*

[10] D.Dhanasekaran, K. Boopathy Bagan, *High Speed Pipelined Architecture for Adaptive Median Filter*, EuroJournals Publishing, Inc. 2009

[11] KazimierzWiatr, *Median and Morphological Specialized Processorsfor a Real-Time Image Data Processing*. 2011

[12] DE2Manual, Altera. 2007

[13] Muthukumar Venkatesan, Daggu Venkateshwar Rao, *Image Processing Algorithms on Reconfigurable Architecture Using handel-C*, University of Nevada Las Vegas. 2006

[14] Milindkumar V. Sarode, Prashant R. Deshmukh, *Reduction of Speckle Noise and Image Enhancement of Images Using Filtering Technique*

[15] MR. M. D. Singh, *Filter Selection for Speckle Noise Reduction*, Thapar University. 2008

[16] Kalpana Saini, *A Modified hybrid Filter for Echocardiographic Image*. June, 2012. Vol. 5, No. 2

[17] Shruthi G Department of ECE, *A Novel Approach for Speckle eduction andEnhancement of Ultrasound Images*. May 2012. Vol. 45, No. 20

[18] R.Vanithamani, *Modified Hybrid Median Filter for Effective Speckle Reduction in Ultrasound Images*

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

www.hcmute.edu.vn