ĐẠI HỌC QUỐC GIA, THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

ĐỒ ÁN MÔN HỌC THIẾT KẾ LUẬN LÝ

BÁO CÁO NHÓM 9

Image Rotation and Mirroring

GVHD:	Huỳnh Phúc Nghị	
SV thực hiện:	Lê Quyết Trung Hiếu	MSSV: 2310952
	Đặng Gia Hưng	MSSV: 2311318
	Lê Kim Ngân	MSSV: 2312218

TP.Hồ Chí Minh, Tháng 10 Năm 2025

Mục lục

1 GIỚI THIỆU

1.1 Tổng quan

- Xử lý ảnh là một lĩnh vực quan trọng trong thời đại số, với nhiều ứng dụng trong y học, an ninh và giải trí. Các phép biến đổi hình học cơ bản như xoay và lật ảnh thường xuyên được thực hiện, nhưng yêu cầu tốc độ xử lý nhanh và hiệu quả. Mặc dù các phương pháp phần mềm có thể thực hiện những tác vụ này, chúng thường không đáp ứng được yêu cầu về tốc độ xử lý trong các hệ thống thời gian thực.

- Đồ án này tập trung vào thiết kế một khối phần cứng chuyên dụng để xoay ảnh (90° theo chiều kim đồng hồ/ngược chiều kim đồng hồ) và lật ảnh (ngang/dọc). Giải pháp này nhằm tối ưu hóa tốc độ xử lý bằng cách thực hiện các phép tính song song, thay vì tuần tự như các phương pháp phần mềm truyền thống. Khối phần cứng này sẽ đọc các điểm ảnh từ bộ nhớ, tính toán lại tọa độ và ghi ảnh đã biến đổi trở lại bộ nhớ. Việc ứng dụng kiến thức về thiết kế luận lý và ngôn ngữ mô tả phần cứng (HDL) vào một dự án thực tế giúp sinh viên hiểu sâu hơn về kiến trúc phần cứng và cách tối ưu hóa hiệu năng bằng cách sử dụng các hệ thống nhúng tốc độ cao như FPGA.

1.2 Nhiệm vụ đề tài

Để hoàn thành đồ án, các nhiệm vụ chính được chia thành các giai đoạn sau:

- **Phân tích và Thiết kế:**
  - **Phân tích yêu cầu:** Tìm hiểu các yêu cầu về các chế độ biến đổi (xoay 90° CW/CCW, lật ngang/dọc) và định dạng dữ liệu (ảnh xám).
  - **Thiết kế thuật toán:** Xây dựng công thức biến đổi tọa độ $(x, y) \rightarrow (x', y')$ cho mỗi chế độ, bao gồm xoay 90° theo chiều kim đồng hồ, xoay 90° ngược chiều kim đồng hồ, lật ngang và lật dọc.
- **Thiết kế kiến trúc phần cứng:**
  - **Thiết kế Khối Điều khiển:** để điều khiển toàn bộ quá trình đọc/ghi bộ nhớ và chọn chế độ biến đổi.
  - **Thiết kế Khối Tính toán Tọa độ:** để thực hiện các phép tính biến đổi tọa độ.
  - **Thiết kế Giao tiếp Bộ nhớ:** để lưu trữ điểm ảnh từ bộ nhớ nguồn và ghi dữ liệu điểm ảnh biến đổi vào bộ nhớ đích.
- **Thực thi và Mô phỏng:**
  - **Viết mã HDL:** Triển khai các khối chức năng đã thiết kế bằng Verilog hoặc VHDL.
  - **Thiết kế Testbench:** Xây dựng Testbench để mô phỏng và kiểm tra hoạt động của hệ thống với dữ liệu đầu vào là ảnh xám mẫu.
- **Mô phỏng và Kiểm thử:**
  - **Mô phỏng tổng thể:** Kiểm tra lỗi để đảm bảo hoạt động chính xác.
  - **Mô phỏng hiển thị:** Mô phỏng hệ thống để xem các phép xoay và lật có được thực hiện đúng hay là không.

- **Báo cáo và Đánh giá:**
  - **Viết báo cáo:** Tổng hợp kết quả, trình bày thiết kế, thuật toán, kết quả mô phỏng và đánh giá hiệu năng.
  - **So sánh và Đánh giá:** Phân tích hiệu quả về tốc độ, tài nguyên sử dụng và khả năng mở rộng so với các phương pháp tương đương.
  - **Đánh giá kết quả:** Đánh giá mức độ bám sát yêu cầu đề tài, độ chính xác và hiệu năng thực hiện.

2 LÝ THUYẾT

Ảnh được xem là một ma trận điểm ảnh (pixel). Vị trí pixel được xác định bằng hệ tọa độ Descartes, gốc $(0, 0)$ ở góc trên cùng bên trái ảnh. Trục x hướng sang phải và trục y hướng xuống dưới. Các phép biến đổi hình học biến đổi tọa độ pixel (x, y) trong ảnh gốc thành tọa độ (x', y') trong ảnh mới. Cho chiều cao, chiều rộng của ảnh là Height, Width.

2.1 Phép lật (Mirroring)

- **Lật ngang (Horizontal Flip):** Ảnh đảo ngược theo trục dọc. Công thức biến đổi tọa độ:

$$x' = \text{Width} - x - 1$$
$$y' = y$$

- **Lật dọc (Vertical Flip):** Ảnh đảo ngược theo trục ngang. Công thức biến đổi tọa độ:

$$x' = x$$
$$y' = \text{Height} - y - 1$$

2.2 Phép xoay (Rotation)

Với phép xoay 90 độ, kích thước của ảnh mới sẽ thay đổi, chiều rộng mới bằng chiều cao cũ và ngược lại.

- **Xoay 90° theo chiều kim đồng hồ (CW):**

$$x' = \text{Height} - y - 1$$
$$y' = x$$

- **Xoay 90° ngược chiều kim đồng hồ (CCW):**

$$x' = y$$
$$y' = \text{Width} - x - 1$$

2.3 Ma trận biến đổi (khái niệm nâng cao)

Các phép biến đổi trên có thể biểu diễn bằng ma trận:

- **Ma trận lật ngang:**

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & \text{Width} - 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- **Ma trận xoay 90° CW:**

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & \text{Height} - 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

3 THIẾT KẾ VÀ THỰC HIỆN

3.1 Phần cứng

3.1.1 Kiến trúc tổng thể

Phần cứng của hệ thống được thiết kế theo kiến trúc mô-đun, phù hợp với sơ đồ khối đã cung cấp. Các khối chức năng này phối hợp với nhau để thực hiện các tác vụ xử lý ảnh.

- **Khối Điều khiển (Control Unit):** Đây là khối điều phối trung tâm của hệ thống. Nó nhận các tín hiệu đầu vào bên ngoài như `clk`, `rst_flag`, `start_flag` và `mode`. Khối này tạo ra các tọa độ pixel tuần tự `(x, y)` để xử lý ảnh, đồng thời phát ra các tín hiệu điều khiển `valid_flag` và `done_flag` để báo trạng thái hoạt động.
- **Khối Giao tiếp Bộ nhớ (Memory Interface):** Khối này quản lý việc đọc/ghi dữ liệu pixel từ bộ nhớ. Nó nhận các tín hiệu `clk`, `we_flag`, tọa độ đọc `(x, y)` từ Khối Điều khiển và tọa độ ghi `(x_new, y_new)` từ Khối Tính toán Tọa độ. Dựa vào các thông số này, khối này thực hiện việc chuyển dữ liệu pixel giữa các vùng nhớ.
- **Khối Tính toán Tọa độ (Coordinate Mapper):** Khối này thực hiện các phép biến đổi toán học. Nó nhận tọa độ gốc `(x, y)` từ Khối Điều khiển và chế độ biến đổi `mode` để tính toán ra tọa độ mới `(x_new, y_new)` theo các công thức đã định sẵn (ví dụ: xoay, lật).
- **Khối Ghi pixel (Pixel Writer):** Khối này điều khiển quá trình ghi dữ liệu pixel. Nó nhận các tín hiệu `clk` và `valid_flag` từ Khối Điều khiển, sau đó tạo ra tín hiệu `we_flag` để báo cho Khối Giao tiếp Bộ nhớ rằng có dữ liệu hợp lệ cần ghi vào bộ nhớ đích.

3.1.2 Luồng hoạt động của hệ thống

1. **Khởi động:** Khi tín hiệu `start_flag` được kích hoạt, Khối Điều khiển bắt đầu quá trình xử lý.
2. **Tạo tọa độ:** Khối Điều khiển tạo ra các tọa độ `(x, y)` tuần tự để duyệt qua tất cả các pixel của ảnh gốc.

3. **Tính toán tọa độ mới:** Tọa độ `(x, y)` được gửi đến Khối Giao tiếp Bộ nhớ để đọc dữ liệu pixel, và đồng thời gửi đến Khối Tính toán Tọa độ để tính toán tọa độ mới `(x_new, y_new)` dựa trên chế độ `mode` đã chọn.
4. **Điều khiển ghi:** Khi có tọa độ mới hợp lệ, Khối Điều khiển phát ra tín hiệu `valid_flag`. Khối Ghi pixel nhận tín hiệu này và tạo ra tín hiệu `we_flag` để cho phép ghi dữ liệu.
5. **Thực thi đọc/ghi:** Khối Giao tiếp Bộ nhớ sử dụng tọa độ `(x, y)` để đọc giá trị pixel từ bộ nhớ nguồn, và sử dụng `(x_new, y_new)` cùng tín hiệu `we_flag` để ghi giá trị pixel đó vào bộ nhớ đích.
6. **Hoàn thành:** Sau khi tất cả các pixel đã được xử lý, Khối Điều khiển phát ra tín hiệu `done_flag` để báo hiệu quá trình biến đổi đã hoàn tất.

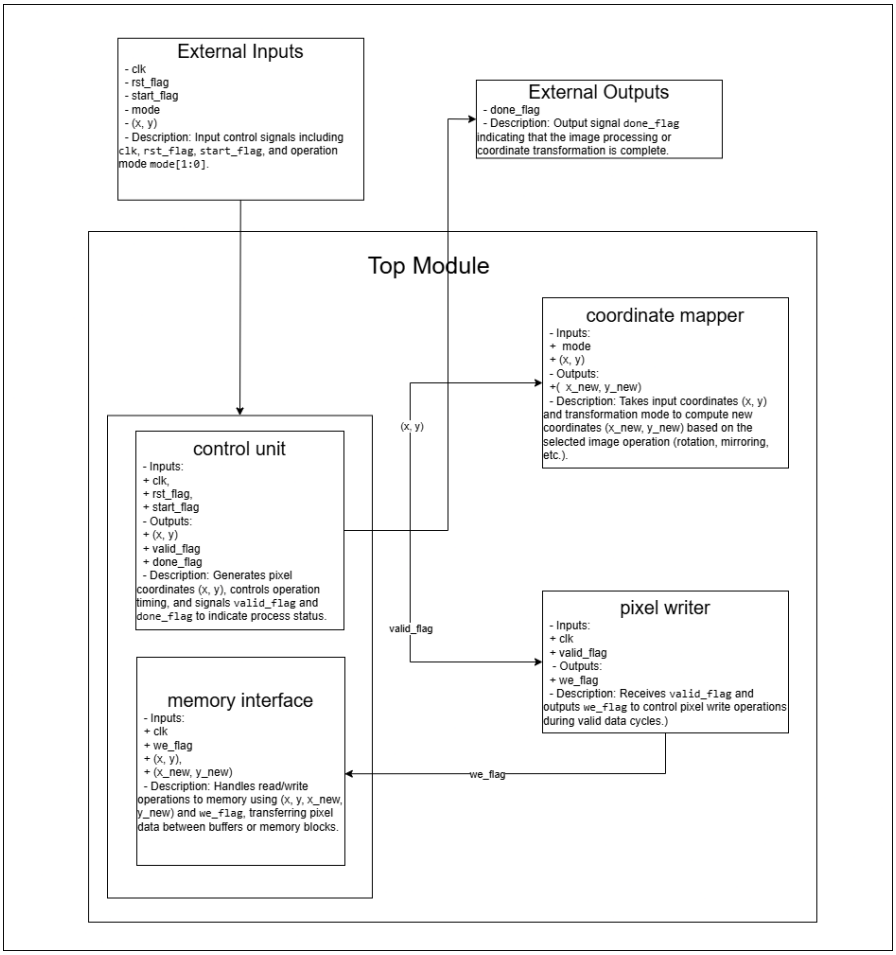

Figure 1: Sơ đồ khối của hệ thống thiết kế bằng Verilog

3.2 Phần mềm

3.2.1 Giới thiệu công cụ

Trong đồ án này, nhóm sử dụng phần mềm **Xilinx Vivado Design Suite** để thiết kế, mô phỏng và tổng hợp mạch số. Vivado là môi trường phát triển tích hợp (IDE) hỗ trợ đầy đủ các công đoạn của quá trình thiết kế phần cứng trên FPGA, bao gồm:

- Biên dịch, kiểm tra cú pháp và mô phỏng mã HDL (Verilog/VHDL).
- Thiết kế, kết nối và quản lý các khối IP.
- Tổng hợp (Synthesis) và ánh xạ (Implementation) thiết kế lên FPGA.
- Phân tích tài nguyên sử dụng, tần số hoạt động và kiểm tra thời gian trễ.
- Mô phỏng hành vi và dạng sóng bằng Vivado Simulator.

3.2.2 Cấu trúc dự án

Dự án được tổ chức trong thư mục `HDL_image` với các tệp và thư mục chính như sau:

- `HDL_image.xpr`: Tệp dự án chính của Vivado, chứa toàn bộ cấu hình và đường dẫn tệp mã nguồn, mô phỏng và IP.
- `HDL_image.srcs/`: Chứa mã nguồn HDL (Verilog/VHDL) của các khối chức năng và các tệp Testbench.
- `HDL_image.sim/`: Lưu kết quả mô phỏng và các dạng sóng (waveform) thu được từ Vivado Simulator.
- `HDL_image.hw/`: Lưu các tệp cấu hình phần cứng, dùng cho quá trình ánh xạ và nạp xuống FPGA.
- `HDL_image.cache/`: Thư mục tạm của Vivado, chứa dữ liệu tổng hợp và ánh xạ được tạo tự động.
- `HDL_image.ip_user_files/`: Chứa dữ liệu và cấu hình của các IP được sử dụng trong thiết kế.
- `.Xil/`: Thư mục hệ thống nội bộ của Vivado, chứa thông tin log và tiến trình tổng hợp.
- `vivado.jou` và `vivado.log`: Hai tệp nhật ký ghi lại quá trình biên dịch, tổng hợp và chạy mô phỏng trong Vivado.

3.2.3 Quy trình thiết kế và mô phỏng

Các bước thực hiện trong Vivado được nhóm tiến hành như sau:

1. **Tạo Project mới:** Chọn loại dự án RTL Project, thêm các tệp mã nguồn Verilog, cấu hình thiết bị FPGA (ví dụ: `xc7a35ticsg324-1L`).
2. **Thêm mã nguồn:** Nhập và tổ chức mã HDL cho các khối chức năng. Đảm bảo các module có khai báo cổng (port) và kết nối hợp lý.
3. **Tạo Testbench:** Xây dựng file Testbench để mô phỏng toàn bộ hệ thống, bao gồm việc phát tín hiệu clock, reset, start và thay đổi chế độ xoay/lật.

4. **Chạy mô phỏng:** Sử dụng Vivado Simulator để kiểm tra hoạt động của từng khối và toàn bộ hệ thống. Quan sát dạng sóng để xác nhận tọa độ và tín hiệu đọc/ghi hoạt động đúng.
5. **Phân tích kết quả:** Kiểm tra tín hiệu `valid_flag` và `done_flag`, đảm bảo ảnh được xử lý chính xác theo chế độ.
6. **Tổng hợp (Synthesis):** Sau khi mô phỏng thành công, tiến hành tổng hợp để chuyển mã HDL sang dạng netlist.
7. **Ánh xạ (Implementation):** Ánh xạ thiết kế lên kiến trúc FPGA thực tế, tối ưu hóa vị trí, đường dây và tài nguyên logic.
8. **Kiểm tra báo cáo tổng hợp:** Đánh giá kết quả sử dụng LUT, FF và tài nguyên bộ nhớ (BRAM) để đảm bảo thiết kế đáp ứng yêu cầu về hiệu năng.

3.2.4 Kết quả mô phỏng

Kết quả mô phỏng trên Vivado Simulator cho thấy:

- Các tín hiệu điều khiển như `clk`, `rst`, `start`, `done` hoạt động đúng chu kỳ.
- Các khối chức năng trao đổi dữ liệu chính xác theo thiết kế: pixel đầu vào được đọc, tính toán lại tọa độ và ghi ra bộ nhớ đích tương ứng.
- Hệ thống hoạt động ổn định trong tất cả các chế độ xoay và lật ảnh.

3.2.5 Đánh giá phần mềm

Phần mềm Vivado giúp nhóm triển khai toàn bộ quá trình thiết kế một cách trực quan và thuận tiện. Nhờ khả năng mô phỏng chính xác, nhóm có thể nhanh chóng phát hiện và khắc phục lỗi logic, tối ưu hóa tài nguyên FPGA, và đảm bảo kết quả hoạt động tương ứng với yêu cầu thiết kế phần cứng đã đề ra.

4 KẾT QUẢ THỰC HIỆN

Phần này trình bày kết quả mô phỏng hệ thống xoay và lật ảnh trên phần mềm **Vivado Simulator**. Kết quả được thể hiện qua các hình ảnh hiển thị trên *Tcl Console* và dạng sóng (*waveform*) cho từng chế độ biến đổi.

4.1 Xoay phải 90° (Rotate 90° Clockwise)

Mô phỏng hiển thị trên Tcl Console

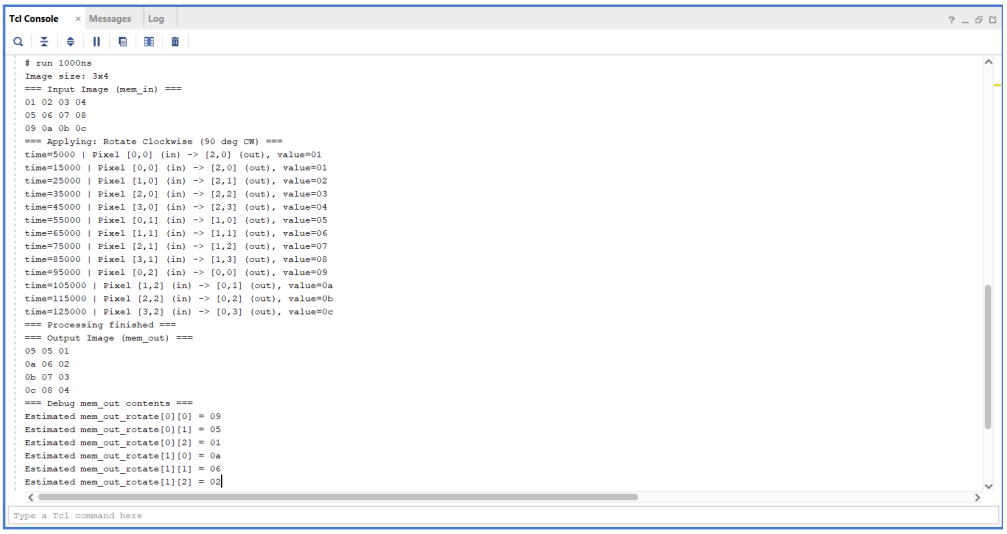

Figure 2: Mô phỏng hiển thị trên Tcl Console khi xoay phải 90°

Mô phỏng hiển thị trên waveform

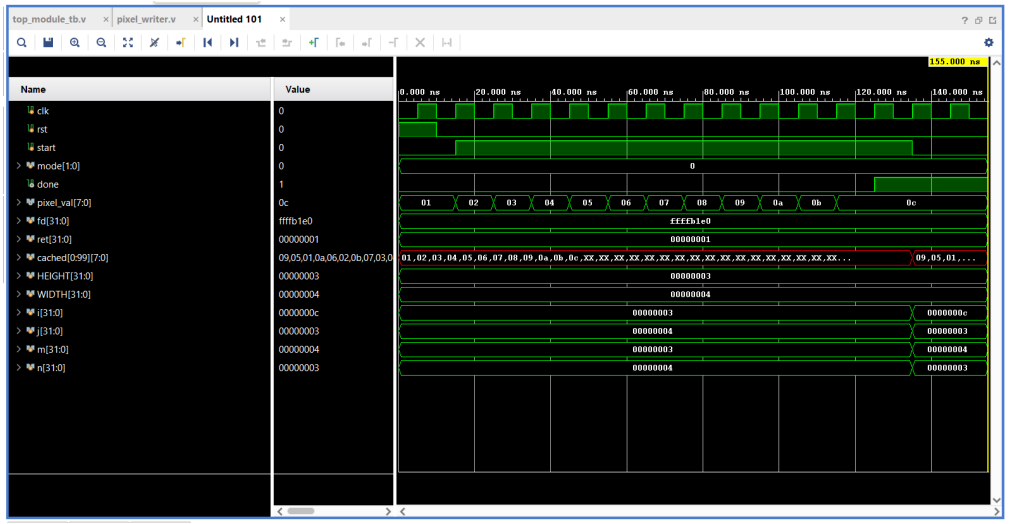

Figure 3: Dạng sóng mô phỏng khi xoay phải 90°

4.2 Xoay trái 90° (Rotate 90° Counter-Clockwise)

Mô phỏng hiển thị trên Tcl Console

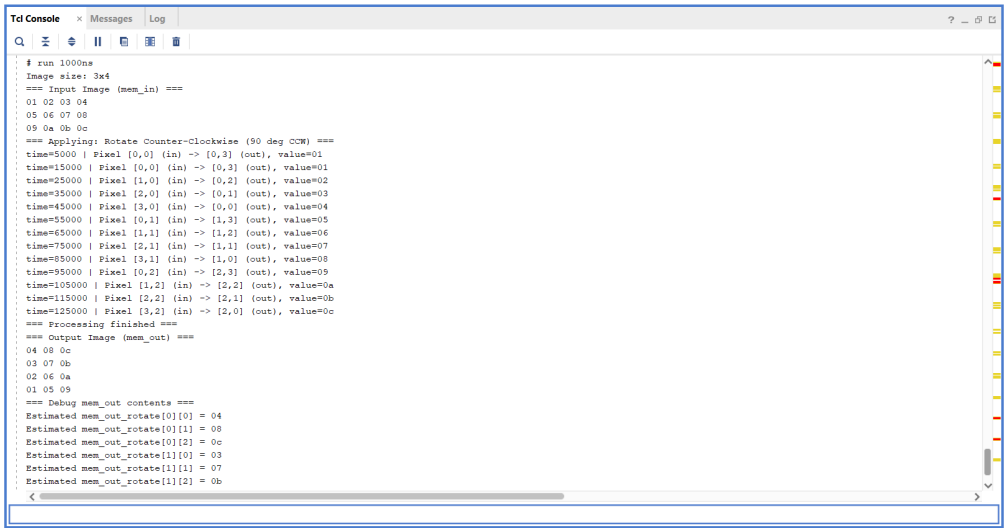

Figure 4: Mô phỏng hiển thị trên Tcl Console khi xoay trái 90°

Mô phỏng hiển thị trên waveform

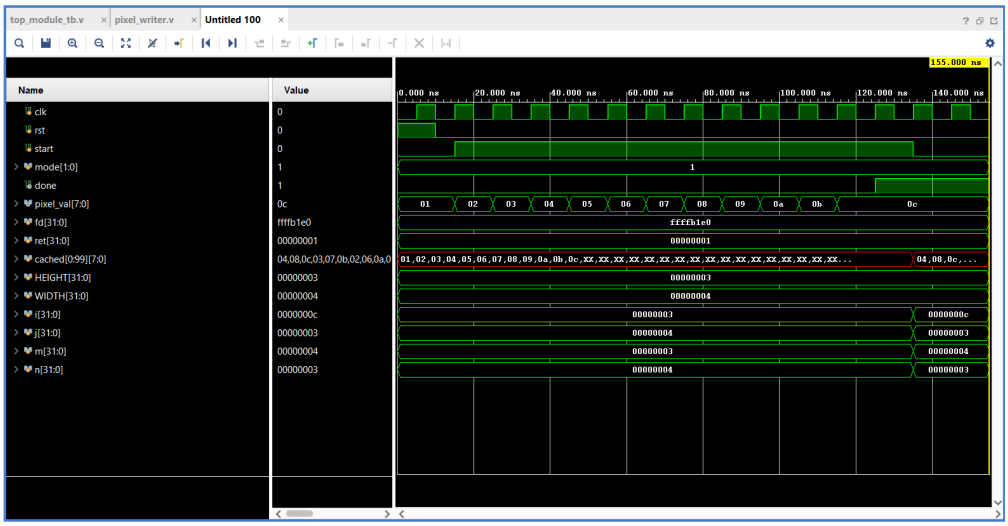

Figure 5: Dạng sóng mô phỏng khi xoay trái 90°

4.3 Lật ngang (Mirror Horizontal)

Mô phỏng hiển thị trên Tcl Console

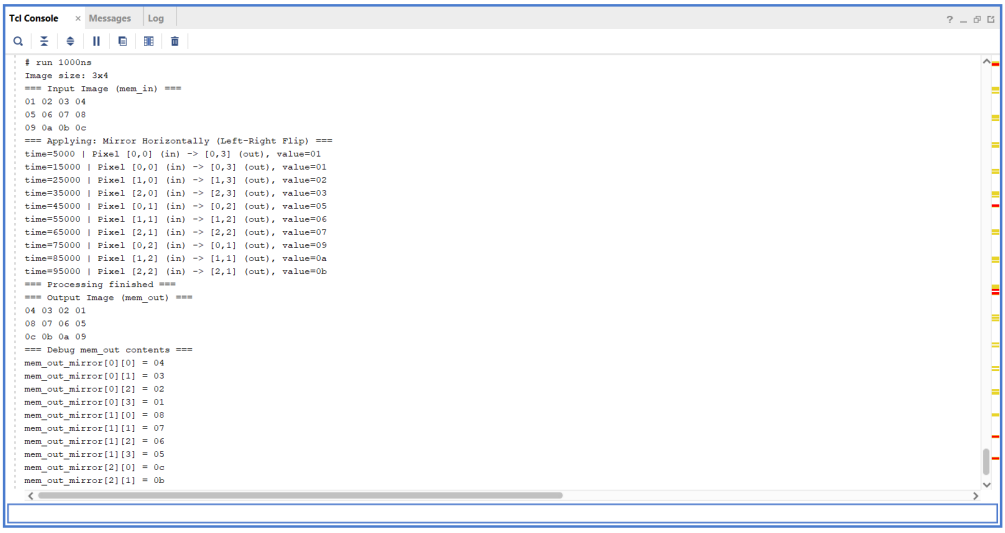

Figure 6: Mô phỏng hiển thị trên Tcl Console khi lật ngang

Mô phỏng hiển thị trên waveform

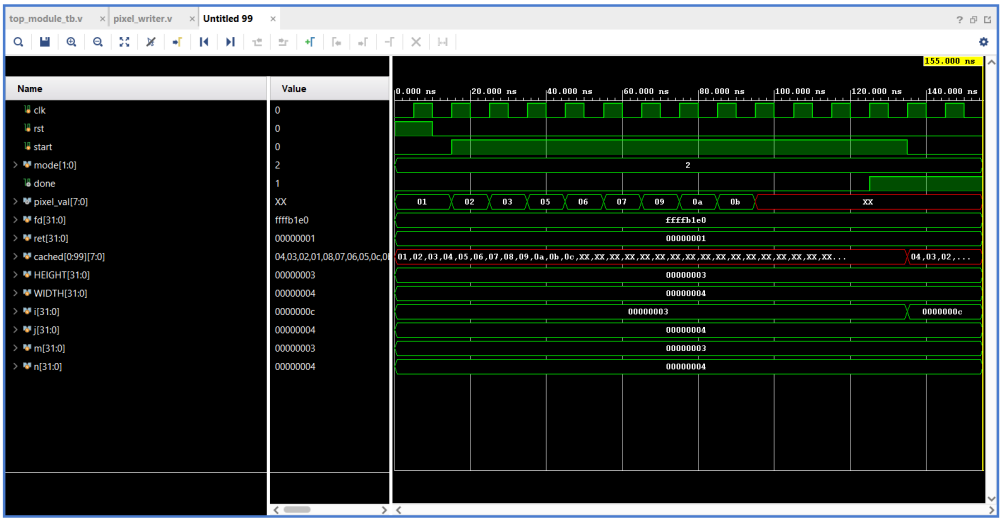

Figure 7: Dạng sóng mô phỏng khi lật ngang

4.4 Lật dọc (Mirror Vertical)

Mô phỏng hiển thị trên Tcl Console

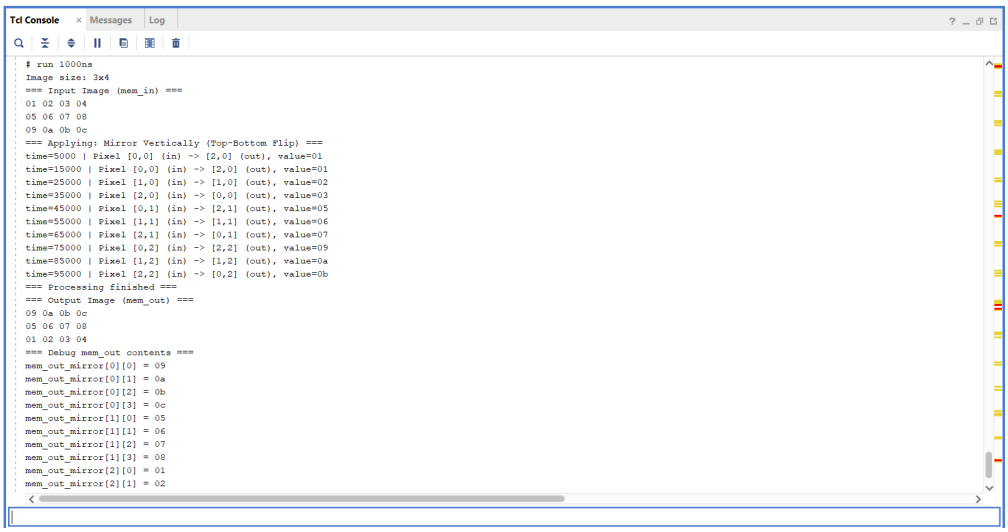

Figure 8: Mô phỏng hiển thị trên Tcl Console khi lật dọc

Mô phỏng hiển thị trên waveform

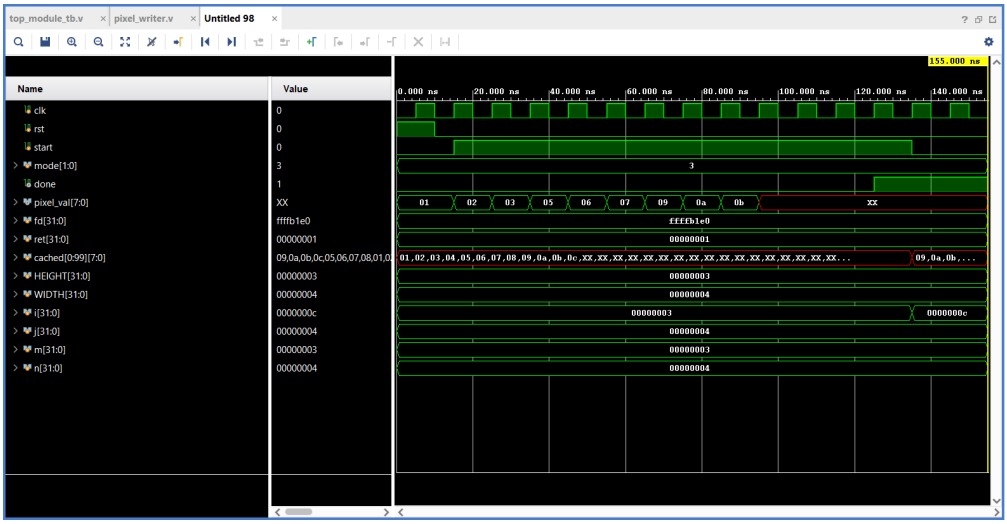

Figure 9: Dạng sóng mô phỏng khi lật dọc

5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Đồ án **"Image Rotation and Mirroring"** đã hoàn thành mục tiêu đề ra là thiết kế và mô phỏng thành công một hệ thống phần cứng có khả năng xoay và lật ảnh theo các chế độ khác nhau. Thông qua việc áp dụng kiến thức về thiết kế luận lý và ngôn ngữ mô tả phần cứng (HDL), nhóm đã:

- Xây dựng được các khối chức năng chính gồm Khối Điều khiển, Khối Giao tiếp Bộ nhớ, Khối Tính toán Tọa độ và Khối Ghi Pixel.
- Mô phỏng toàn bộ hệ thống trên phần mềm **Vivado Simulator**, xác nhận hoạt động đúng trong các chế độ: xoay 90° theo chiều kim đồng hồ, xoay 90° ngược chiều kim đồng hồ, lật ngang và lật dọc.
- Hiểu rõ hơn quy trình thiết kế mạch số, cách kết nối và kiểm thử các module trong môi trường FPGA.

Kết quả mô phỏng cho thấy hệ thống hoạt động ổn định, các tín hiệu điều khiển (`start`, `done`, `valid`) được đồng bộ chính xác, và quá trình đọc – ghi dữ liệu ảnh được thực hiện đúng theo yêu cầu. Đồ án giúp nhóm củng cố kỹ năng thiết kế luận lý, tư duy song song trong xử lý phần cứng và khả năng sử dụng công cụ Vivado trong thực tế.

5.2 Hướng phát triển

Trong tương lai, nhóm đề xuất một số hướng mở rộng và hoàn thiện hệ thống như sau:

- **Hỗ trợ ảnh màu (RGB):** Mở rộng thiết kế để xử lý các kênh màu, giúp hệ thống có thể áp dụng với hình ảnh thực tế.
- **Tăng độ phân giải:** Tối ưu hóa khối giao tiếp bộ nhớ và luồng xử lý để hỗ trợ ảnh có kích thước lớn hơn.
- **Tích hợp trên FPGA thực tế:** Thực hiện ánh xạ và kiểm thử trực tiếp trên kit FPGA để đánh giá hiệu năng thực tế.
- **Bổ sung giao diện người dùng:** Thiết kế giao diện điều khiển (UI) để người dùng có thể lựa chọn chế độ xoay/lật dễ dàng.
- **Tối ưu tài nguyên:** Giảm số lượng LUT/FF sử dụng bằng cách chia sẻ khối tính toán, cải thiện tốc độ xử lý và tiết kiệm phần cứng.

Tổng kết lại, đồ án đã đạt được mục tiêu đặt ra và là nền tảng vững chắc để nhóm tiếp tục phát triển các ứng dụng xử lý ảnh trên phần cứng tốc độ cao trong tương lai.

6 DEMO

https://github.com/DgHnG36/Logic-Design-Project-ImageRot-Mir-HCMUT-251

7 Tài liệu tham khảo

- R. Gonzalez and R. Woods, *Digital Image Processing*, Pearson, 4th Ed., 2018.
- Vladimir Kovalevsky and Steven Stoddart, *Modern Algorithms for Image Processing: Computer Imagery by Example*, Springer, 2020.
- Peter Wilson, *Design Recipes for FPGAs Using Verilog and VHDL*, Elsevier, 2nd Ed., 2016.