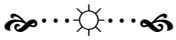
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA



ĐỒ ÁN MÔN HỌC THIẾT KẾ LUẬN LÝ (CO3091)

BÁO CÁO GIAI ĐOẠN 3 (BÁO CÁO TỔNG KẾT)

Lớp L08 - Nhóm 3 - HK221

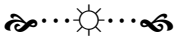
Giảng viên hướng dẫn: Huỳnh Phúc Nghị

STT	Sinh viên thực hiện	Mã số sinh viên	Đánh giá
1	Phạm Duy Quang	2011899	100%
2	Hà Vĩnh Nguyên	2011698	55%
3	Đỗ Thành Minh	2011610	55%
4	Phạm Đức Thắng	2012080	30%

Thành phố Hồ Chí Minh, tháng 12 năm 2022

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA



ĐỀ TÀI: ỨNG DỤNG PHÁT HIỆN VẠCH KẺ ĐƯỜNG, VẠCH KẺ Ô ĐỖ XE, VẠCH TRẮNG CHỈ ĐƯỜNG,... THÔNG QUA HÌNH ẢNH THU ĐƯỢC TỪ CAMERA

STT	Sinh viên thực hiện	Khối lượng công việc
1	Phạm Duy Quang	Tìm hiểu và đề xuất các giải thuật; tìm hiểu các thiết bị phần cứng phù hợp với đề tài; hiện thực, hoàn thiện và kiểm thử các giải thuật trên phần mềm và phần cứng; viết báo cáo về các bước hiện thực và tổng hợp báo cáo
2	Hà Vĩnh Nguyên	Hỗ trợ hiện thực, chỉnh sửa và hoàn thiện giải thuật trên phần mềm và phần cứng
3	Đỗ Thành Minh	Hỗ trợ tìm hiểu và viết báo cáo về giải thuật; tìm hiểu các thiết bị phần cứng cho đề tài
4	Phạm Đức Thắng	Hỗ trợ tìm hiểu và viết báo cáo về giải thuật; tìm hiểu các thiết bị phần cứng cho đề tài

Thành phố Hồ Chí Minh, tháng 12 năm 2022

MỤC LỤC

I. Báo cáo giai đoạn 3 (tổng kết): Hoàn thiện và kiểm thử **2**

1. Nội dung nhóm đã làm được trong các giai đoạn vừa rồi 2

2. Nội dung nhóm dự định làm trong tương lai 8

3. Khó khăn mà nhóm gặp phải trong các giai đoạn vừa rồi 9

II. Nội dung báo cáo **9**

1. Đánh giá thực nghiệm 9

1.1. Phần mềm (C++ với thư viện OpenCV trên CPU) 9

1.2. Tiền xử lý trên phần cứng FPGA (thư viện xfOpenCV) kết hợp với C++ trên bộ xử lý ARM (thư viện OpenCV) 10

2. So sánh hiệu năng 11

2.1. Trên phần mềm 11

2.2. Trên nền tảng SDSoC (kết hợp hiện thực trên phần cứng và phần mềm FPGA) 11

2.3. Nhận xét 11

3. Hoàn thiện báo cáo 12

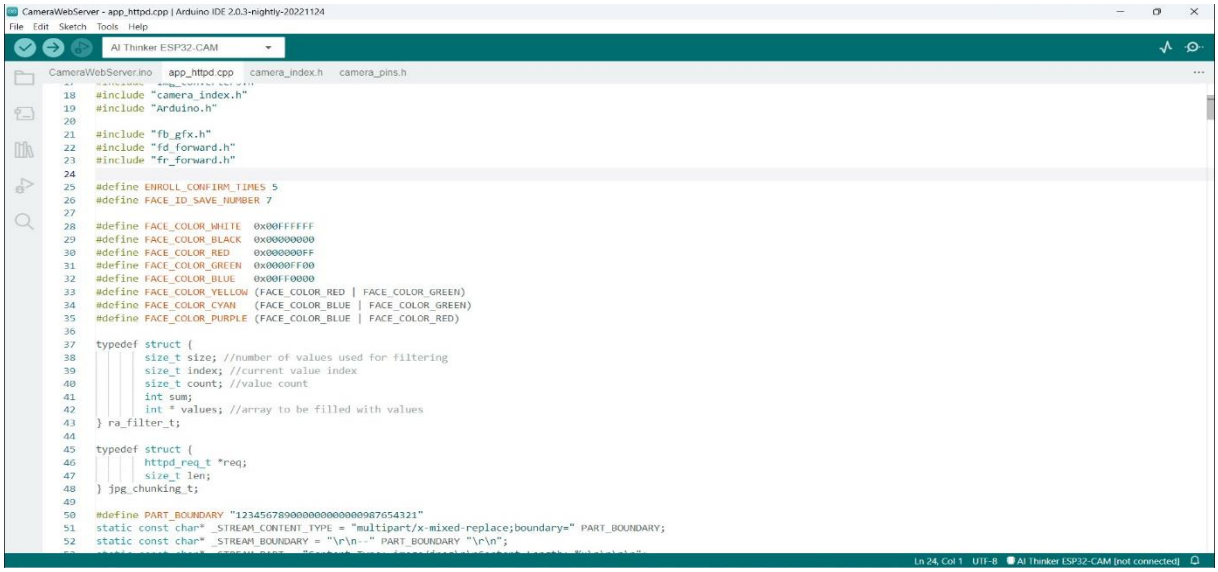
III. Tài liệu tham khảo **13**

I. Báo cáo giai đoạn 3 (tổng kết): Hoàn thiện và kiểm thử

Link Github về Project của nhóm 3 trong giai đoạn 3 (tổng kết): https://github.com/ULTIMATE-Mystery/Logic-Design-Project-Group-3-Semester-221-HCMUT.

1. Nội dung nhóm đã làm được trong các giai đoạn vừa rồi

- Sau rất nhiều cố gắng, cuối cùng nhóm cũng đã có thể hiện thực các giải thuật phát hiện cạnh/góc trực tiếp trên ESP32-CAM với Arduino IDE và hiển thị kết quả lên máy với sự hỗ trợ từ Python:

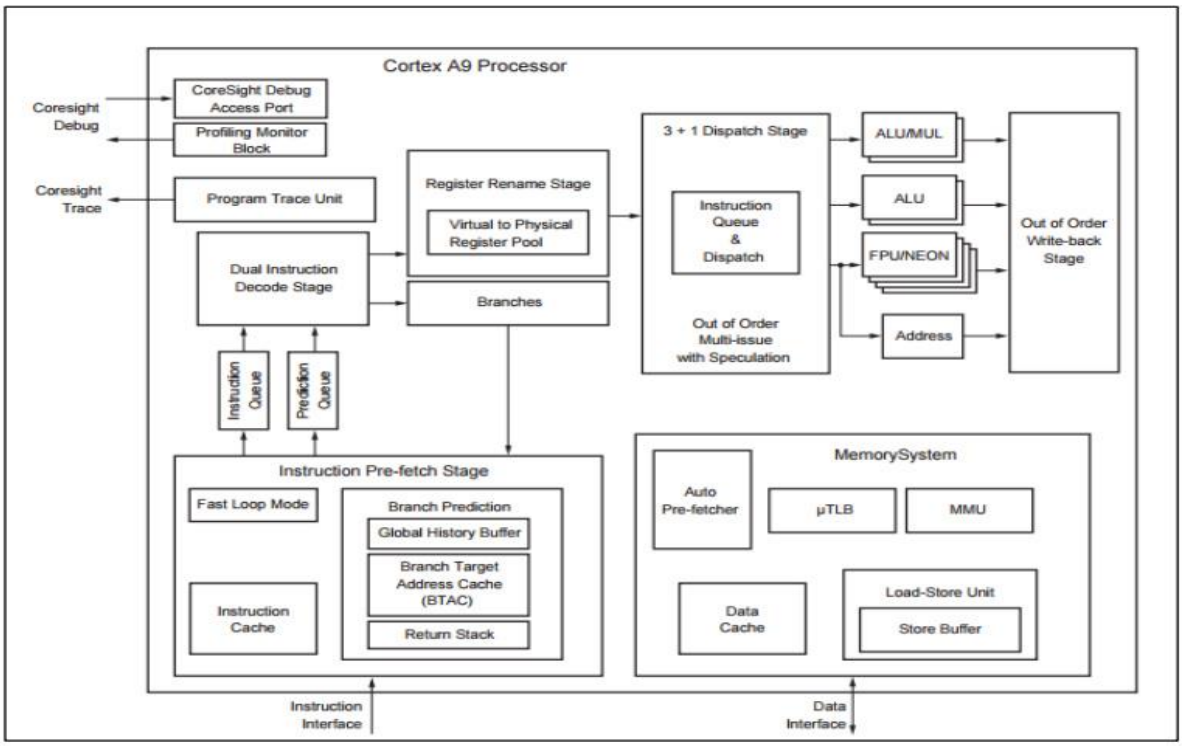


Tuy nhiên điều này khó có thể áp dụng ra thực tiễn. ESP32-CAM với bộ nhớ giới hạn (520 KB SRAM và 4 MB PSRAM) và khả năng xử lý hạn chế cũng như không có sự hỗ trợ từ thư viện OpenCV, nhóm chỉ có thể xử lý một ảnh (kích thước nhỏ) trong mỗi lần xử lý và không thể thực hiện các giải thuật phát hiện làn đường trên thời gian thực. Đối với những ảnh có kích thước lớn và mật độ điểm ảnh dày, việc xử lý ảnh trên ESP32-CAM còn gây ra quá tải và bắt buộc phải reset thiết bị.

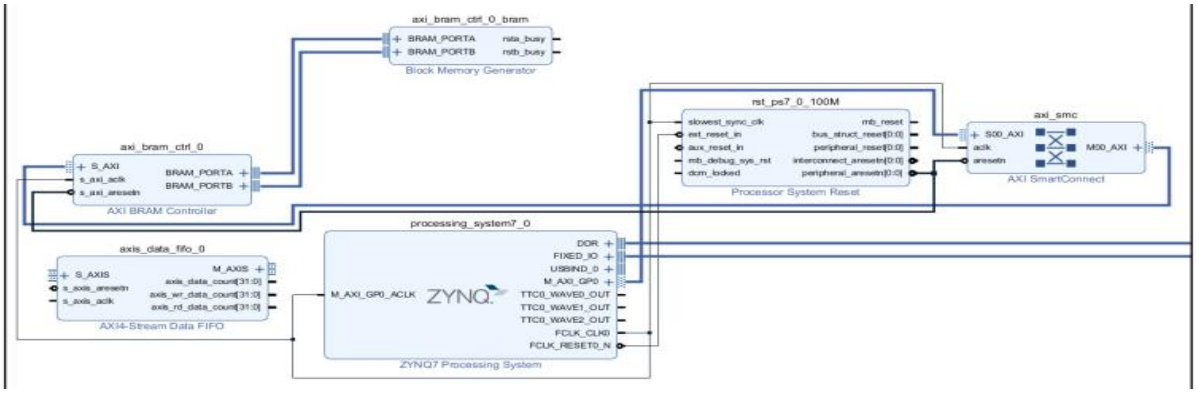
- Nhóm cũng đã hoàn thiện các giải thuật phát hiện làn đường hỗ trợ cho hệ thống xe tự lái trên FPGA ZedBoard Zynq-7000 với Module Camera OV7670:

- Bo mạch phát triển FPGA ZedBoard Zynq-7000 của hãng Xilinx:

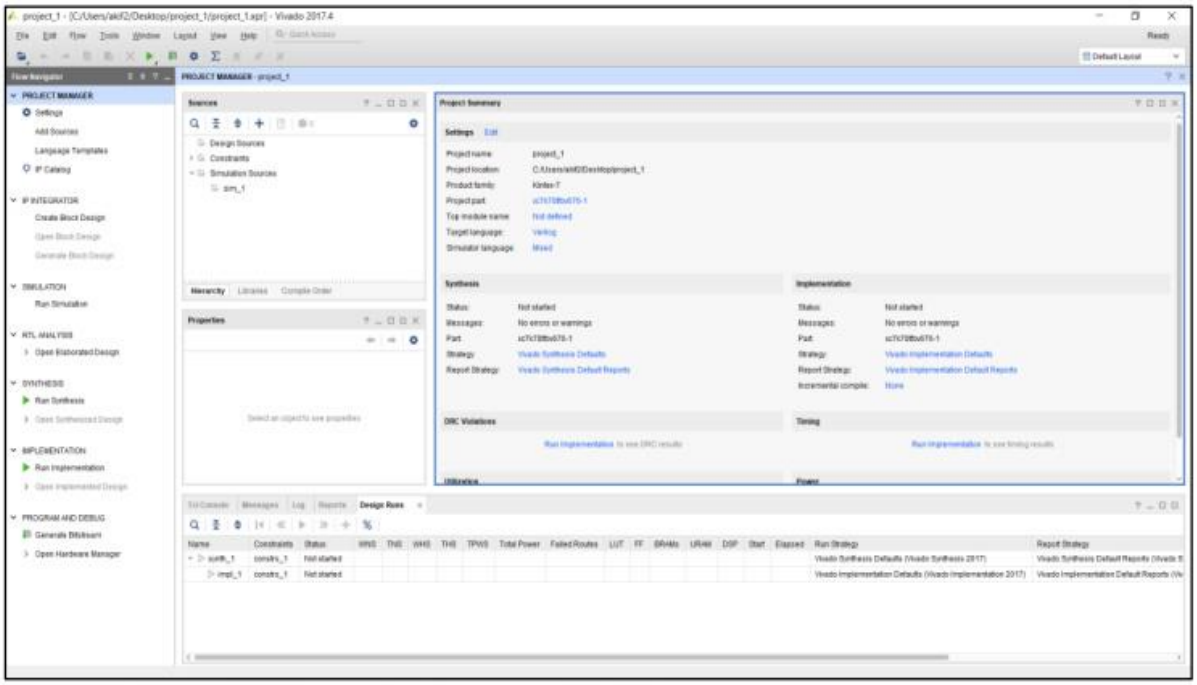
- Cấu trúc ARM Cortex A9:



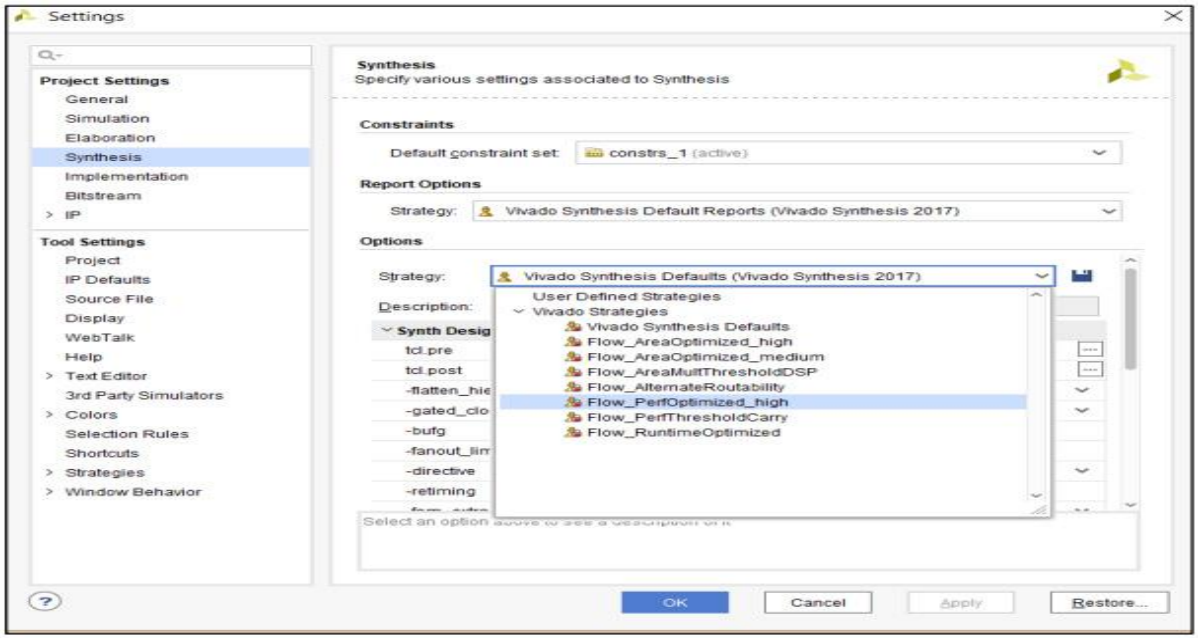
- Thiết kế các khối trên nền tảng VIVADO (phần cứng FPGA):



- Lập trình trên nền tảng VIVADO:

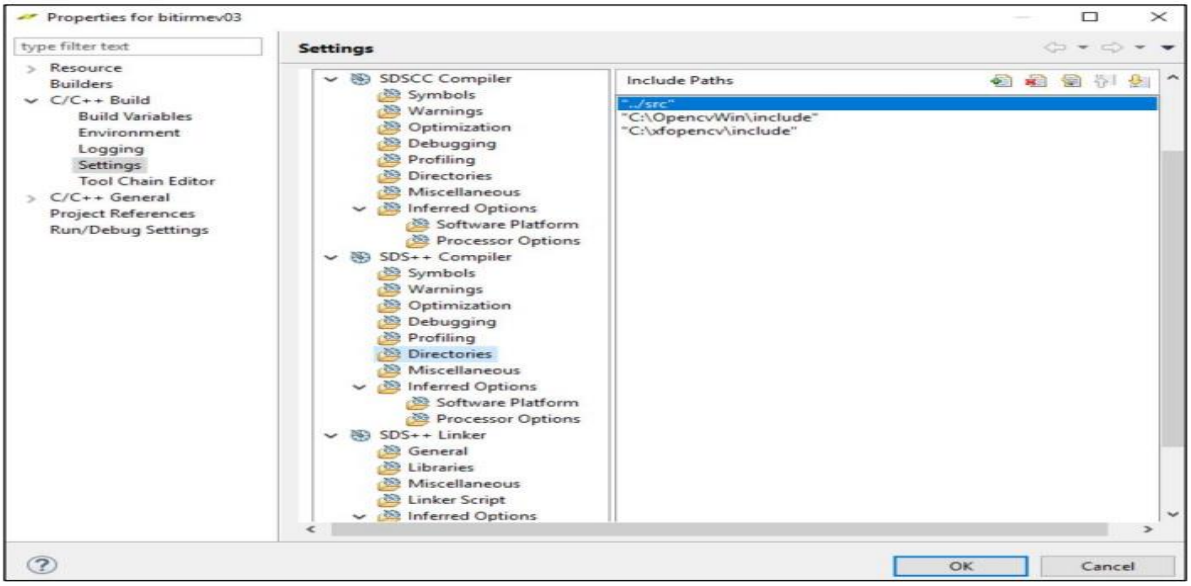


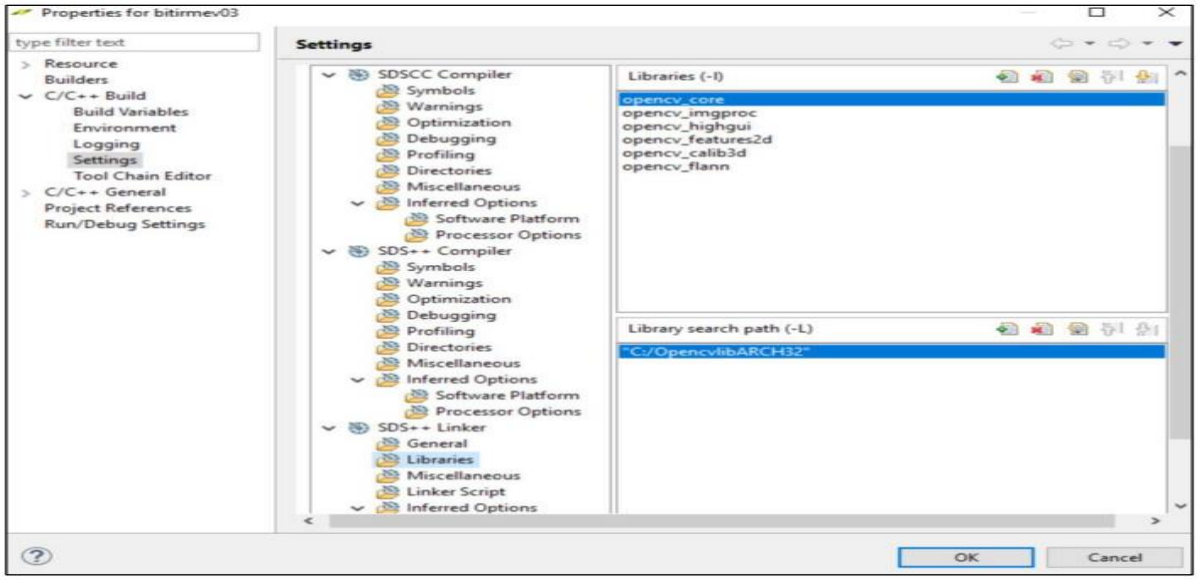
- Thiết lập cài đặt trên nền tảng VIVADO:



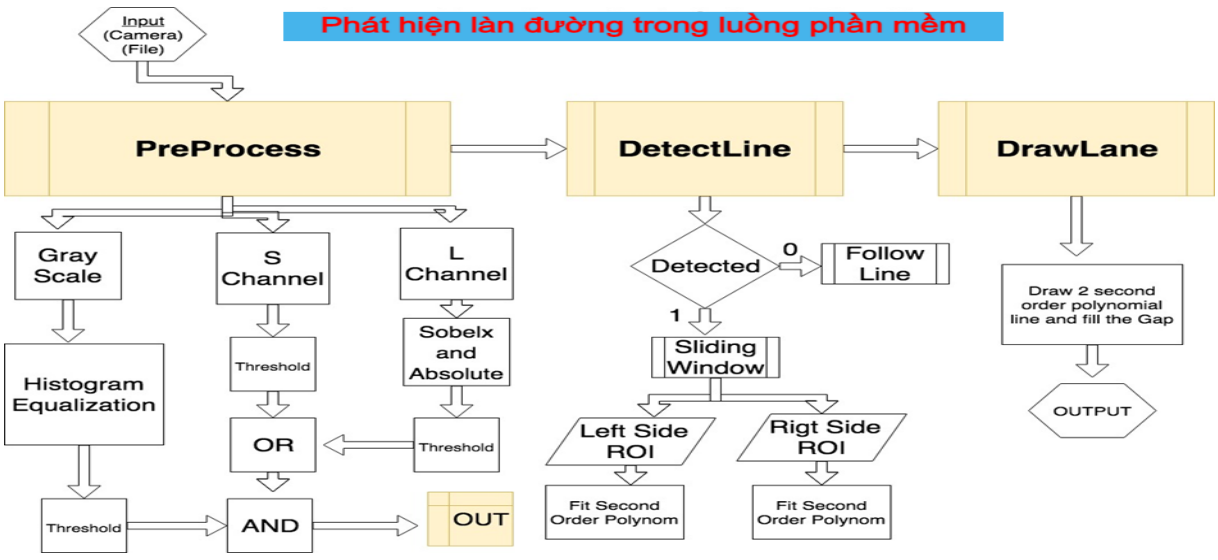
- Lập trình trên nền tảng SDSoC (kết hợp hiện thực trên phần cứng và phần mềm FPGA):

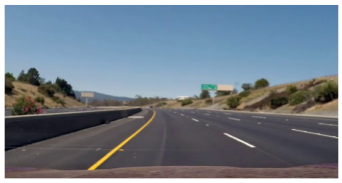
- Liên kết thư viện OpenCV và xfOpenCV:





- Các bước xử lý giải thuật trên phần mềm (C++ với thư viện OpenCV trên CPU):





a-) Original Frame (RGB)

b-) “L” Channel (HLS Color Space)

e-) “S” Channel (HLS Color Space)

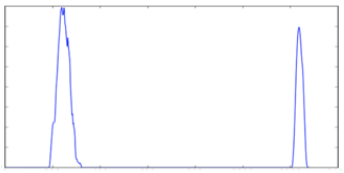
c-) Horizontal derivation

f-) Threshold

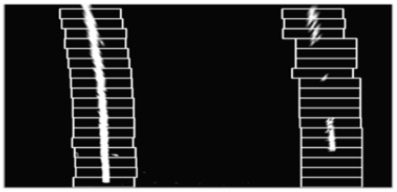
d-) Threshold

g-) Bitwise OR

h-) After Perspective Transform

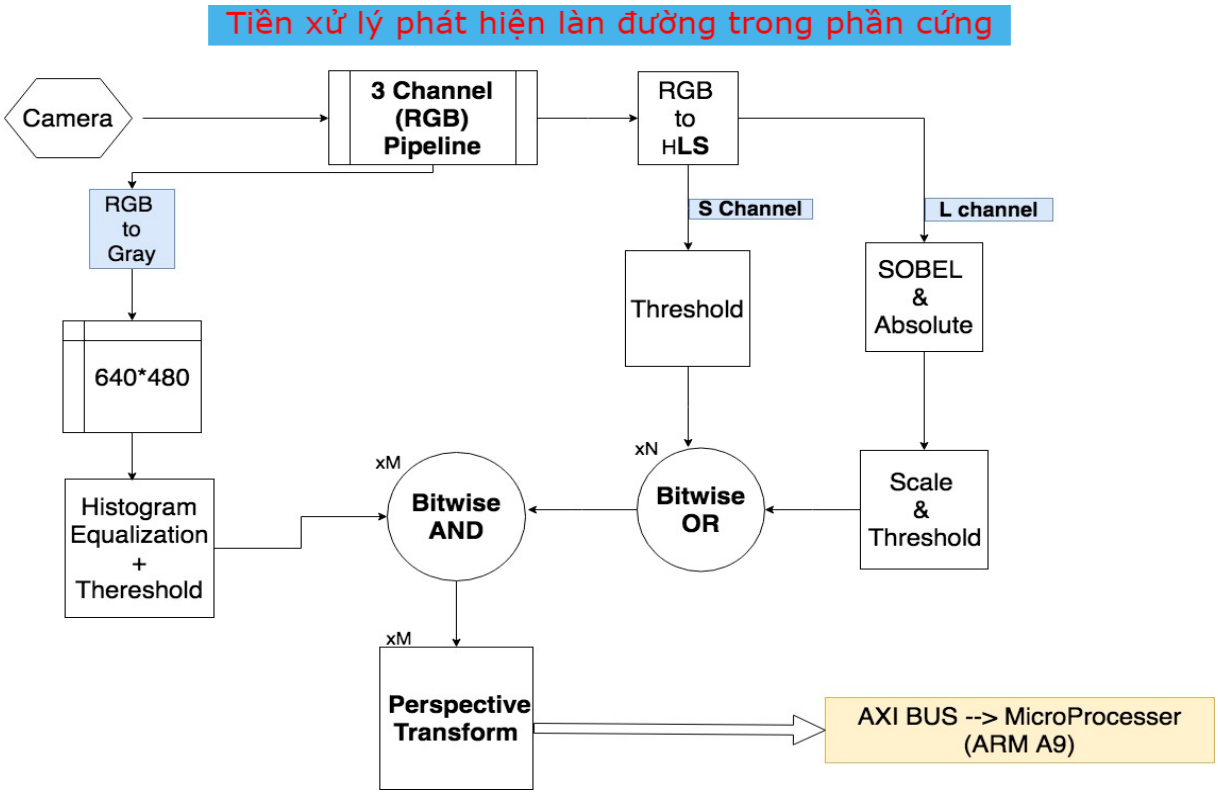


i-) Histogram



j-) Sliding Windows

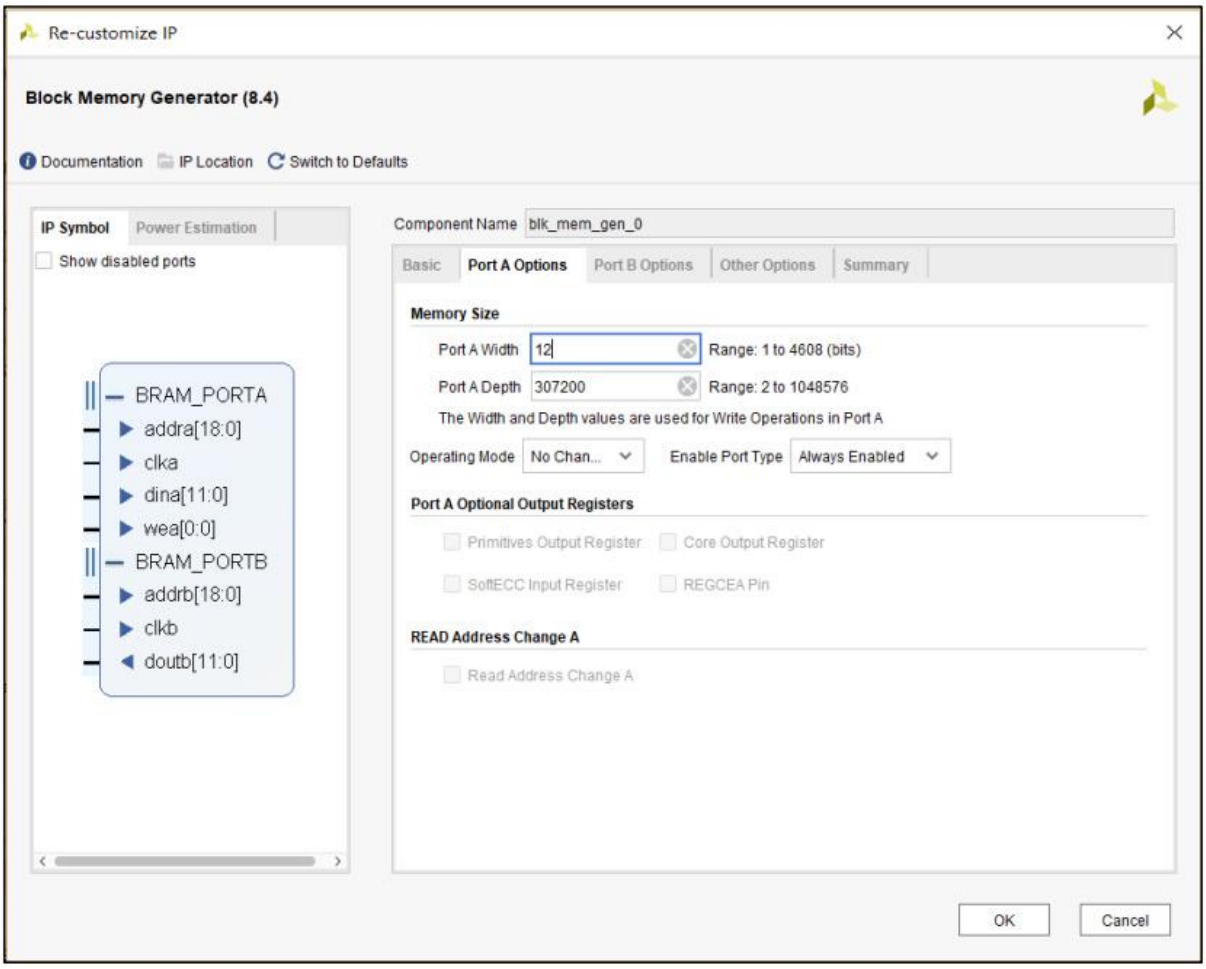
- Tiền xử lý phát hiện làn đường trong phần cứng FPGA:



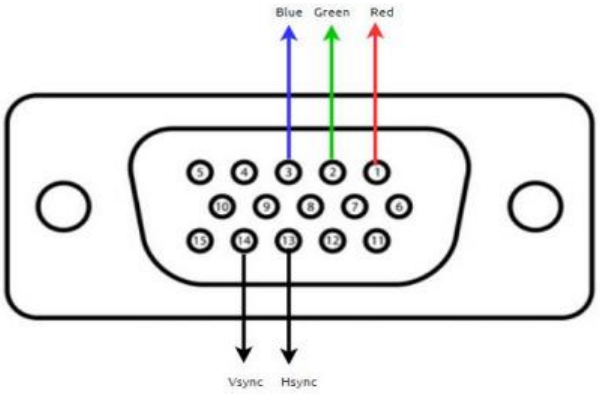
- Module Camera OV7670:

- Kết nối Module Camera OV7670 với FPGA ZedBoard Zynq-7000:

- Điều chỉnh kích thước của khối RAM:



- Kết nối VGA để hiển thị ra màn hình:



2. Nội dung nhóm dự định làm trong tương lai

- Hoàn thiện và cải tiến hơn nữa để các giải thuật phát hiện làn đường càng trở nên chính xác, và nếu có thể sẽ tiến tới đưa vào áp dụng trong thực tiễn.
- Áp dụng các kỹ thuật xử lý ảnh trong thị giác máy tính mà nhóm đã học được không chỉ trong nội dung đồ án môn học mà còn trong các dự án tương lai.

3. Khó khăn mà nhóm gặp phải trong các giai đoạn vừa rồi

- Nhóm gặp nhiều khó khăn khi tiếp cận về nội dung mới lạ, phức tạp của các giải thuật phát hiện cạnh/góc từ hình ảnh. Nhóm cũng gặp khó khăn trong việc bàn bạc, quyết định chọn lựa phần cứng để từ đó lựa chọn phương pháp, nền tảng hiện thực các giải thuật.

- Với nguồn lực, kinh tế và thời gian giới hạn, nhóm khó tiếp cận được các thiết bị phần cứng với hiệu năng xử lý cao như FPGA. Việc hiện thực các tác vụ xử lý hình ảnh mà không có thư viện OpenCV (ESP32-CAM không hỗ trợ) khiến cho nhóm gặp rất nhiều khó khăn.

II. Nội dung báo cáo

1. Đánh giá thực nghiệm

Với việc các làn đường đã được biểu thị theo các phương trình bậc 2, thuật toán phát hiện làn đường của nhóm đã có thể phát hiện các làn đường ở các khúc cua. Nhưng khi tới những khúc cua gắt, chẳng hạn như đường núi, thì thuật toán phát hiện làn đường của nhóm cũng có thể phát hiện một phần làn đường. Thuật toán cũng phát hiện được làn đường trong các tình huống ngoại cảnh khác nhau như trời râm, trời mưa, các làn đường được vẽ bằng các màu khác nhau,...

1.1. Phần mềm (C++ với thư viện OpenCV trên CPU)

- Hình ảnh trước khi xử lý trên phần mềm:

- Hình ảnh sau khi xử lý trên phần mềm:

1.2. Tiền xử lý trên phần cứng FPGA (thư viện xfOpenCV) kết hợp với C++ trên bộ xử lý ARM (thư viện OpenCV)

- Hình ảnh trước khi đưa vào xử lý trên FPGA:

- Hình ảnh sau khi đưa vào xử lý trên FPGA:

2. So sánh hiệu năng

Ta tiến hành so sánh hiệu năng của thuật toán phát hiện làn đường trên các nền tảng khác nhau cho Video lái xe trên đường cao tốc với độ phân giải 640x480.

2.1. Trên phần mềm

Chức năng	Số lượng khung hình (frame) trong video	Thời gian xử lý chậm nhất (trên một khung hình)	Thời gian xử lý nhanh nhất (trên một khung hình)	Thời gian xử lý trung bình (trên một khung hình)
Tiền xử lý	1260	81,71 ms	8,21 ms	36,99 ms
Tất cả các hàm	1260	153,31 ms	25,47 ms	77,84 ms
Tiền xử lý / Tất cả các hàm (%)	-	53,3%	32,2%	47,5%

2.2. Trên nền tảng SDSoC (kết hợp hiện thực trên phần cứng và phần mềm FPGA)

Chức năng	Số khung hình (frame) xử lý	Thời gian xử lý (phần mềm)	Thời gian xử lý (phần cứng)	Tỉ lệ giảm thời gian xử lý (hiệu suất)
Tiền xử lý (chuyển đổi phối cảnh)	1	157,07 ms	113,76 ms	Giảm khoảng 28%
Tốc độ khung hình (FPS)	1	Khoảng 6,37 FPS	Khoảng 8,79 FPS	Hiệu suất tăng khoảng 1,38 lần

2.3. Nhận xét

Sử dụng thư viện "xfOpenCV" với ngôn ngữ lập trình C++ thông qua nền tảng SDSoC (tổng hợp phần cứng mức cao) trên FPGA Zedboard đạt được tốc độ khoảng

8,78 khung hình/giây (FPS) cho chức năng tiền xử lý (không bao gồm chuyển đổi phối cảnh). Nếu khối phần cứng không được sử dụng và triển khai (không bao gồm chuyển đổi phối cảnh) trong bộ xử lý, hiệu suất là khoảng 6,37 khung hình/giây (FPS). Kết quả là cấu trúc SoC được triển khai trên Zedboard với nền tảng SDSoC và thư viện "xfOpenCV" sử dụng ngôn ngữ lập trình C++ đạt hiệu suất tốt hơn khoảng 1,38 lần so với giải pháp phần mềm thuần túy trên bộ xử lý.

Trên nền tảng VIVADO kết nối một Module Camera OV7670 đầu ra 25 MHz với ngôn ngữ đặc tả phần cứng Verilog. Chức năng tiền xử lý của thuật toán phát hiện làn đường được hiện thực trên bo mạch phát triển FPGA Zedboard và xử lý trong khoảng 12,3 ms tức là với tốc độ khung hình khoảng 81,3 khung hình/giây. Kết quả này tốt hơn 3,01 lần so với giải pháp trên phần mềm và 18,4 lần so với giải pháp trên ARM Cortex A9 (667 MHz).

3. Hoàn thiện báo cáo

Thuật toán phát hiện làn đường bao gồm ba phần chính: Tiền xử lý, phát hiện làn đường, vẽ làn đường. FPGA Zedboard dòng ZYNQ-7000 của hãng Xilinx được sử dụng làm “Hệ thống trên Chip” (SoC) để giải quyết vấn đề này. Thuật toán phát hiện làn đường được triển khai trên FPGA sau khi thuật toán được chia ra thực hiện trên phần mềm và phần cứng.

Theo phân tích thời gian của thuật toán phát hiện làn đường, phần tiền xử lý đã được thực hiện trong phần cứng. Các nền tảng SDSoC và VIVADO đã được sử dụng để thực hiện thuật toán phát hiện làn đường lập trình trên FPGA. Ngôn ngữ đặc tả phần cứng (Verilog) được sử dụng trên nền tảng VIVADO trong khi ngôn ngữ lập trình bậc cao (C++) được sử dụng trên nền tảng SDSoC. Trên nền tảng SDSoC, thư viện "xfOpenCV" cũng được sử dụng. Thư viện “xfOpenCV” là một bộ gồm hơn 50 nhân (kernels), được tối ưu hóa cho FPGA và SoC của Xilinx, dựa trên thư viện thị giác máy tính OpenCV. Các nhân trong thư viện xfOpenCV được tối ưu hóa và hỗ trợ trong bộ công cụ Xilinx SDSoC.

Một số phần của thuật toán thuộc hệ thống được triển khai trong phần cứng để hiện thực “Hệ thống phát hiện làn đường” nhanh hơn. Zedboard FPGA dòng ZYNQ-7000 của hãng Xilinx phù hợp với hệ thống vì thuật toán phát hiện làn đường có thể được tách biệt trên đó dưới dạng khối phần cứng và phần mềm. Khối phần mềm có sẵn trên bộ xử

lý ARM Cortex-A9 trong Zedboard; khối phần cứng sẽ được triển khai trong các ô logic của Zedboard.

Thuật toán được chia thành 2 phần: Chức năng tiền xử lý được thực hiện trong phần cứng (logic lập trình) và các chức năng khác được thực hiện trong phần mềm (phần mềm lập trình – Arm-A9). Chức năng tiền xử lý được triển khai trong phần cứng do khả năng xử lý mạnh mẽ của nó.

Kết luận: Khi thiết kế phần cứng được thực hiện, ngôn ngữ lập trình bậc cao có thể được sử dụng cho các giải pháp tham chiếu hoặc cho các giải pháp ngắn hạn. Mặt khác, ngôn ngữ lập trình bậc thấp phù hợp với các giải pháp dự kiến sẽ nâng cao hiệu suất sau này.

III. Tài liệu tham khảo

1. *OpenCV | Real Time Road Lane Detection*, truy cập từ: https://www.geeksforgeeks.org/opencv-real-time-road-lane-detection.
2. *Edge Detection with Gaussian Blur*, truy cập từ: https://www.projectrhea.org/rhea/index.php/Edge_Detection_with_Gaussian_Blur.
3. *OpenCV - Image Thresholding*, truy cập từ: https://docs.opencv.org/5.x/d7/d4d/tutorial_py_thresholding.html.
4. *OpenCV - Canny Edge Detection*, truy cập từ: https://docs.opencv.org/5.x/da/d22/tutorial_py_canny.html.
5. *OpenCV - Hough Line Transform*, truy cập từ: https://docs.opencv.org/5.x/d3/de6/tutorial_js_houghlines.html.
6. *[Tìm hiểu] ESP32-CAM AI-Thinker Board là gì?*, truy cập từ: https://blog.mecsu.vn/esp32-cam-ai-thinker-board-la-gi.
7. *Image processing on FPGA using Verilog HDL*, truy cập từ: https://www.fpga4student.com/2016/11/image-processing-on-fpga-verilog.html.