BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

--------🕮--------

CHU VĂN THÀNH

THIẾT KẾ VÀ THỰC HIỆN HỆ THỐNG THU VÀ HIỂN THỊ ẢNH TRÊN NỀN FPGA

LUẬN VĂN THẠC SĨ KỸ THUẬT

CHUYÊN NGÀNH : KỸ THUẬT TRUYỀN THÔNG

NGƯỜI HƯỚNG DẪN KHOA HỌC : TS. VÕ LÊ CƯỜNG

HÀ NỘI – 2013

LỜI CAM ĐOAN

Tôi xin cam đoan:

Những kết quả nghiên cứu, các số liệu, hình vẽ, biểu bảng, kết quả tính toán được trình bày trong luận văn là hoàn toàn trung thực, không vi phạm bất cứ điều gì trong luật sở hữu trí tuệ và pháp luật Việt Nam.

TÁC GIẢ LUẬN VĂN

Chu Văn Thành

DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT

ASIC	:	Application Specific Integrated Circuit (vi mạch tích hợp chuyên dụng trong điện tử)
CCD	:	Charged Coupled Device (thiết bị tích điện kép)
CLB	:	Configurable Logic Blocks (khối cấu hình lôgic)
CMOS	:	Complementary Metal-Oxide Semiconductor (công nghệ dùng để chế tạo vi mạch tích hợp)
CPLD	:	Complex Programmable Logic Device
DSP	:	Digital signal processing (xử lý tín hiệu số)
FPGA	:	Field-programmable gate array (vi mạch dùng cấu trúc mảng phần tử logic mà người dùng có thể lập trình được)
GAL	:	Generic Array Logic devices
HDL	:	Hardware Description Language (Ngôn ngữ mô tả phần cứng)
LUT	:	Look-Up Table
MSI	:	Medium scale intergration (Tích hợp qui mô trung bình)
PAL	:	Programmable Array Logic devices
PC	:	Personal Computer (Máy tính cá nhân)
PDA	:	Personal Digital Assistant (Thiết bị kỹ thuật số hỗ trợ cá nhân)
PLD	:	Programmable Logic Device (Thiết bị logic lập trình được)
RAM	:	Random Access Memory (bộ nhớ truy xuất ngẫu nhiên)
ROM	:	Read Only Memory (phần bộ nhớ chỉ đọc)
SDRAM	:	Synchronous Dynamic RAM (DRAM đồng bộ)
SSI	:	Small scale integration (Tích hợp qui mô nhỏ)
TTL	:	Transistor transistor logic
VGA	:	Video Graphics Array
VHDL	:	VHSIC Hardware Description Language

DANG MỤC BẢNG

Trang

DANH MỤC HÌNH

Trang

MỤC LỤC

TÓM TẮT

Với sự phát triển và ứng dụng rộng rãi của hệ thống nhúng, nó đã được nghiên cứu ứng dụng trong thu thập và xử lý hình ảnh. Tuy nhiên do cấu trúc thiết kế của hệ thống nhúng hạn chế về tốc độ xử lý ảnh hưởng tới chất lượng hình ảnh thu được, bởi dữ liệu video có kích thước lớn, vì vậy việc thực hiện ảnh thời gian thực với độ tin cậy cao trên hệ thống nhúng là khó thực hiện. Đối với hệ thống thu thập hình ảnh tốc độ cao với quá trình thời gian thực, yêu cầu tốc độ xử lý cao vì một số lượng lớn dữ liệu hình ảnh cần được xử lý. Hệ thống thu thập hình ảnh được sử dụng rộng rãi trong công nghiệp, quân sự, y tế, an ninh, ví dụ như: trong điện thoại video, hội nghị truyền hình, hệ thống giám sát, điều khiển trong công nghiệp, giám sát từ xa.

Sự phát triển nhanh chóng của FPGA cung cấp một giải pháp mới cho hệ thống thu thập và xử lý hình ảnh. Bài luận văn đưa ra một phương án về thiết kế, thực hiện hệ thống thu thập và hiển thị hình ảnh với quá trình thời gian thực trên FPGA, với nội dung trình bày về các ứng dụng, tình trạng nghiên cứu của hệ thống thu thập hình ảnh, so sánh ưu điểm và nhược điểm của DSP, ASIC và FPGA trong hệ thống thu thập và xử lý hình ảnh, đề xuất thiết kế và thực hiện hệ thống thu thập hình ảnh trên FPGA. Trong thiết kế hệ thống được chia thành năm module chức năng chính, module thu thập hình ảnh, module lưu trữ hình ảnh, module hiển thị hình ảnh, module xử lý FPGA và module ngoại vi. Để thực hiện hệ thống, tác giả đã đưa ra sự lựa chọn chip và thiết kế mạch phần cứng cho các khối bao gồm: Mạch thu thập ảnh, mạch giao diện SDRAM, mạch giao diện VGA, Chip điều khiển logic và giao diện thiết bị ngoại vi. Trong đó khối FPGA điều khiển camera, nhận và xử lý thô dữ liệu hình ảnh thu được từ camera, dữ liệu được lưu tạm thời vào SDRAM sau đó đọc dữ liệu hình ảnh từ SDRAM gửi ra cổng VGA hiển thị lên màn hình LCD. Hệ thống được thực hiện bởi FPGA thuộc dòng CycloneII của Altera.

Bài luân văn thảo luận về kết quả mô phỏng các module trên phần mềm Quartus II và thực nghiệm trên Kit DE1 của hãng Altera, kết quả đó đã chứng minh tính đúng đắn và tính khả thi của quá trình thiết kế. Các module chính bao gồm: Module camera CMOS, module kiểm soát đọc ghi SDRAM và module xử lý FPGA. Hệ thống được thiết kế theo hướng nghiên cứu trên đã đạt được hiệu quả mong đợi bằng phương pháp thử nghiệm xác minh.

Từ khóa: Thu thập hình ảnh; Bộ cảm biến hình ảnh CMOS; Thời gian thực; FPGA

Abstract

Because of the development and wide application of embedded systems, they have been studied and applied in image acquisition and processing system. However, as the structural design of embedded systems is limited in term of processing speed, affecting the quality of the large size video image data, so the implementation of real-time image on the embedded systems is difficult. For the system which collects images at high speed in real time, requirement for high-speed processing is needed. The image acquition system is widely applied in industrial, military, medical and security purposes such as in the video phone, image recognition, video conferencing, monitoring system, industry control, remote monitoring.

Rapid development of FPGA provides a new solution to the system of collecting and processing image. This paper provides a solution for the high-speed acquisition and real-time processes of image data based on FPGA, with contents to present about the applications and research of image acquisition system, compare advantages and disadvantages of DSP, ASIC and FPGA in image acquisition and processing system, proposed a FPGA-based image acquisition and processing system. The whole system is divided into five major functional modules: image acquisition module, image storage module, image display module, FPGA core processing module and peripheral module. To complet the system design, the author selects chips and designs the hardware circuit including image acquisition circuit, SDRAM interface circuit, VGA interface circuit, the chip's logical control, peripheral interface logical control. FPGA is incharge of controlling camera, receiving and processing of raw data collected from the camera, the data is temporarily saved in SDRAM and then read image data from SDRAM to send out VGA display on the LCD screen. The system is implemented by FPGA under the Altera's CycloneII.

The dissertation discusses results of simulation by the Quartus II software modules and the experimentation on Kit DE1, simulation results prove the correctness and feasibility of the design system. These modules mainly includes: CMOS camera module, SDRAM literacy control module and image preprocessing module. The system designed by the above way achieves a satisfying effect by experimental verification.

Keywords: Image Acquisition; CMOS Image Sensor; Real-time; FPGA.

MỞ ĐẦU

1. Lý do chọn đề tài

Với sự phát triển của công nghệ xử lý ảnh, hệ thống thu và xử lý ảnh được ứng dụng ngày càng nhiều. Trong thị trường cảm biến hình ảnh hiện nay cảm biến CMOS được ứng dụng nhiều do có nhiều tiện ích và hơn nữa giá thành thấp. Trong nhiều ứng dụng của hệ thống thu và xử lý ảnh sử dụng DSP để điều khiển cảm biến ảnh và nhận dữ liệu hình ảnh, sau đó truyền tới PC qua cổng USB [4], trong hệ thống này, dữ liệu hình ảnh được đọc bằng phần mềm, do đó không đáp ứng được yêu cầu thu ảnh thời gian thực.

Đối với hệ thống thu thập hình ảnh tốc độ cao với quá trình xử lý thời gian thực, yêu cầu tốc độ xử lý cao vì một số lượng lớn dữ liệu hình ảnh cần được xử lý. Vì vậy, công nghệ xử lý song song là đặc biệt quan trọng. Việc nghiên cứu ứng dụng sản phẩm công nghệ xử lý song song vào hệ thống thu thập hình ảnh thời gian thực là cần thiết và đây cũng chính là lý do em lựa chọn làm đề tài nghiên cứu *"Thiết kế và thực hiện hệ thống thu và hiển thị ảnh trên nền FPGA"*.

2. Lịch sử nghiên cứu

Quá trình nghiên cứu trong và ngoài nước cho các hệ thống thu thập và hiển thị hình ảnh hiện nay đã có rất nhiều hướng nghiên cứu khác nhau và có những kết quả được công bố trên các trang báo như: "Design of a DSP-based CMOS Imaging System for Embedded Computer Vision" [4], "Design of an Imaging System based on FPGA Technology and CMOS Imager" [8], "Design of CMOS Image Acquisition System Based on FPGA" [1], FPGA – Based CMOS Image Acquisition System" [2], và một số kết quả nghiên cứu khác.

Đặc biệt với các chip xử lý DSP, FPGA được sử dụng làm phương pháp thu thập hình ảnh và đã trở thành một xu hướng trong lĩnh vực thu thập hình ảnh thời gian thực. So với trong nước, các nước phát triển trong lĩnh vực về việc thu thập hình ảnh và hệ thống xử lý, với sự phát triển nhanh chóng các sản phẩm có độ bền, độ tin cậy cao, phạm vi sử dụng rộng, nhưng cần chi phí cao cho việc thực hiên hệ thống. các sản phẩm trong nước có giá thấp hơn, với độ tin cậy và tính chính xác thấp. Vì

vậy, việc cải thiện chất lượng thu thập hình ảnh hiện có và phát triển công nghệ xử lý ảnh là một nhu cầu cần thiết.

3. Mục đích nghiên cứu, đối tượng và phạm vi nghiên cứu của đề tài

Mục đích chính của đề tài là thiết kế, thực hiện hệ thống thu và hiển thị ảnh tốc độ cao với quá trình thời gian thực của dữ liệu hình ảnh dựa trên FPGA [2]. Dựa vào kết quả nghiên cứu có thể phát triển hệ thống ứng dụng vào các thiết bị ghi hình, an ninh, giám sát và tự động điều khiển,...

Đối tượng nghiên cứu của đề tài là hệ thống thu ảnh từ cảm biến hình ảnh sử dụng công nghệ CMOS với bộ xử lý FPGA, dữ liệu ảnh được lưu tạm thời vào SDRAM và hiển thị ảnh trên giao diện VGA [2]. Trong đó việc đọc dữ liệu điểm ảnh từ cảm biến hình ảnh lưu vào SDRAM và hiển thị ảnh lên màn hình VGA được diễn ra liên tục. Đề tài đã tính toán tới tốc độ đọc ghi dữ liệu, chất lượng hình ảnh thu được và thiết lập chế độ đọc ảnh từ cảm biến.

Nghiên cứu nguyên lý làm việc của cảm biến hình ảnh OV9650 thuộc hãng OmniVision và phương pháp mã hóa tín hiệu điểm ảnh với ngõ ra của cảm biến.

Nghiên cứu về việc đọc ghi dữ liệu SDRAM và phương pháp hiển thị ảnh trên giao diện VGA.

Nghiên cứu các phương pháp tạo mã chương trình cấu hình phần cứng FPGA.

4. Phương pháp nghiên cứu

Để giải quyết vấn đề nêu trên có thể nhiều phương pháp nghiên cứu khác nhau như:

- Phương pháp nghiên cứu tài liệu và các bài báo
- Thiết kế và mô phỏng kết quả trên máy tính
- Phương pháp nghiên cứu thử nghiệm xác minh

Trong nội dung được trình bày luận văn, tác giả đã đưa ra lựa chọn phương pháp nghiên cứu thử nghiệm xác minh, việc phân tích tổng quan thiết kế hệ thống thu và hiển thị hình ảnh dựa trên nguyên lý quét điểm ảnh.

Tìm hiểu về phương pháp đọc, ghi dữ liệu với SDRAM và đọc dữ liệu điểm ảnh từ cảm biến OV9650. Từ đó lựa chọn giải pháp thực hiện thiết kế và tạo mã chương trình cho hệ thống.

Ứng dụng phương pháp quét xen dòng và xử lý tín hiệu màu nhằm tăng tốc độ thu và giảm thời gian hiển thị hình ảnh.

5. Tóm tắt cô đọng các luận điểm cơ bản và đóng góp mới của tác giả

Việc nghiên cứu ứng dụng cảm biến hình ảnh sử dụng công nghệ CMOS, bộ nhớ SDRAM và bộ xử lý FPGA cũng như việc sử dụng phần mềm Quartus II đã mang lại hiệu quả trong việc thiết kế, thực hiên hệ thống thu và hiển thị hình ảnh trên nền FPGA. Việc ứng dụng FPGA và cảm biến hình ảnh OV9650 trong hệ thống thu ảnh là giá trị khoa học của đề tài.

Trong lĩnh vực nghiên cứu khoa học, kết quả nghiên cứu của đề tài được sử dụng làm cơ sở nghiên cứu và phát triển các hệ thống thu thập hình ảnh có độ phức tạp và có ý nghĩa to lớn trong việc ứng dụng cho các hệ thống camera giám sát, hội nghị truyền hình,...

Đối với thực tiễn, hệ thống có thể được ứng dụng trong những hệ thống giám sát, chụp hình với yêu cầu độ chính xác thấp. Ngoài ra bài luận văn này cũng là một tài liệu cho việc thiết kế, thực hiện những hệ thống trên FPGA.

Việc thực hiện thành công mô hình hệ thống thu và hiển thi ảnh trên nền FPGA đã đáp ứng yêu cầu cho những hệ thống thu và hiển thị ảnh tốc độ cao với quá trình thời gian thực và những hệ thống không cần tới sự hỗ trợ của máy tính. Kết quả nghiên cứu này cũng là một thành công để tạo hướng nghiên cứu phát triển về sau cho những hệ thống thu và hiển thị ảnh.

6. Nội dung trình bày luận văn

Do thời gian nghiên cứu, thực hiện đề tài hạn hẹp, với kiến thức và việc tìm hiểu về hệ thống còn hạn chế, luận văn này thực hiện trong phạm vi "*Thiết kế và thực hiện hệ thống thu và hiển thị ảnh trên nền FPGA*" được thực hiện gồm các phần sau:

MỞ ĐẦU: Trong phần mở đầu tác giả trình bày tính lý do chọn đề tài, lịch sử nghiên cứu, mục đích, đối tượng và phạm vi nghiên cứu, tóm tắt cô đọng các luận điểm cơ bản và đóng góp mới của tác giả.

Chương 1: TỔNG QUAN**:** Nêu tính thời sự của đề tài, phân tích đánh giá các công trình nghiên cứu đã có liên quan đến đề tài và những vấn đề cần tập trung

nghiên cứu giải quyết. Trình bày ngắn gọn về chức năng các khối trong hệ thống thu thập và hiển thị hình ảnh trên nền FPGA.

Chương 2: PHÂN TÍCH LỰA CHỌN HỆ THỐNG: Trong chương phân tích và lựa chọn các thiết bị cho thiết kế mạch phần cứng cho các khối trong đó có chip FPGA và cảm biế hình ảnh, bộ nhớ lưu trữ dữ liệu tạm thời SDRAM và giao diện hiển thị hình ảnh VGA.

Chương 3: THIẾT KẾ HỆ THỐNG THU THẬP HÌNH ẢNH: Chương này tập trung vào thiết kế sơ đồ khối chức năng hệ thống thu thập hình ảnh tốc độ cao. Thiết kế mạch phần cứng và thiết kế mã chương trình điều khiển của mỗi khối chức năng, bao gồm: khối thu thập hình ảnh, bộ nhớ lưu dữ liệu hình ảnh SDRAM, màn hình hiển thị hình ảnh VGA và phần cứng ngoại vi.

Chương 4: MÔ PHỎNG VÀ THỰC HIỆN HỆ THỐNG TRÊN FPGA: trình bày về các kết quả mô phỏng trên phần mềm Quatus II và quá trình thực hiện hệ thống thu hình ảnh với các module chính bao gồm: module cảm biến ảnh CMOS, module điều khiển SDRAM đọc và viết dữ liệu hình ảnh, module điều khiển hiển thị ảnh.

KẾT LUẬN VÀ KIẾN NGHỊ: Đưa ra các nhận xét, đánh giá về hệ thống, những điểm đạt được và những hạn chế, điểm chưa đạt được của đề tài. Nêu kiến nghị bản thân và các đề xuất hoàn thiện hệ thống cũng như hướng phát triển của đề tài.

Chương 1: TỔNG QUAN

1.1 Tính thời sự của đề tài

Nhìn lại sự phát triển máy ghi hình cách đây gần 40 năm, tức là vào tháng 12 năm 1975, một thợ chụp ảnh tên Steven Sasson đã khai sinh ra kỷ nguyên máy ảnh kĩ thuật số bằng tấm hình đầu tiên chụp tại phòng kĩ thuật của công ty Kodak [25]. Tuy nhiên, cũng phải đợi cho đến đầu thập niên 90 mới thật sự phát triển và được ứng dụng rộng rãi. Ngày nay trong rất nhiều các hệ thống, thiết bị sử dụng hàng ngày gắn thiết bị ghi hình như trên máy ảnh, điện thoại, máy tính, thiết bị giám sát… ngoài ra thiết bị ghi hình được ứng dụng trong một số lĩnh vực nghiên cứu công nghệ cao như chế tạo Robot, thiết bị dò đường, … với sự phát triển khoa học công nghệ trong nước và ngoài nước trong việc ứng dụng thiết bị ghi lại hình ảnh thì công nghệ xử lý của camera cũng cần phải được thay đổi để phát triển cùng với các ngành khoa học khác.

Sự phát triển của các ngành khoa học công nghệ yêu cầu không chỉ công nghệ chế tạo các chip camera ngày càng nâng cao mà còn đòi hỏi công nghệ xử lý dữ liệu hình ảnh về chất lượng cũng như tốc độ ghi hình cũng cần được cải tiến để đáp ứng trong những hệ thống như: Camera giám sát, hệ thống tự đông điều khiển,… đối với những yêu cầu của cảm biến hình ảnh không những chỉ về độ trung thực của hình ảnh thu được mà tốc độ ghi hình và thời gian xử lý ảnh cũng cần được đáp ứng nhanh để hệ thống có những điều khiển phù hợp. Chính vì vậy, viêc thu và hiển thị hình ảnh tốc độ cao với quá trình thời gian thực là rất cần thiết.

Hiện nay trên thị trường đã ứng dụng chip DSP cho việc tích hợp, tính toán tốc độ đã được cải thiện đáng kể, chi phí giảm đáng kể [4]. DSP trở thành xu hướng chính của các hệ thống ghi và xử lý hình ảnh.

Việc sử dụng máy tính cho công nghệ xử lý hình ảnh kỹ thuật số có ý nghĩa thay đổi hướng đi cho công nghệ xử lý lưu trữ truyền thống tương tự trong thông tin kỹ thuật số. Thay vì số hóa, công nghệ ghi lại hình ảnh và xử lý dần chuyển sang hướng kỹ thuật số. Với sự phát triển của công nghệ vi điện tử, công nghệ xử lý ảnh được sử dụng trong các lĩnh vực khác nhau của thiết kế điện tử, công nghệ xử lý

hình ảnh kỹ thuật số đã đạt được những tiến bộ mang tính đột phá. Hệ thống thu thập hình ảnh thời gian thực đóng một vai trò cực kỳ quan trọng trong công nghệ đa phương tiện. Ngày nay, hầu hết các hệ thống thu thập hình ảnh với tốc độ thời gian thực và đã được sử dụng rộng rãi trong các điện thoại di động, PDA, điều khiển công nghiệp, thị giác máy, giám sát thời gian thực và các lĩnh vực khác. Trong thập kỷ qua, sự phát triển nhanh chóng của FPGA (Field Programmable Gate Array) đã tạo ra hướng nghiên cứu, thiết kế ứng dụng vào việc thu thập hình ảnh thời gian thực.

Hệ thống thu thập và hiển thị hình ảnh được thực hiện theo nguyên lý quét điểm ảnh từ cảm biến hình ảnh và xử lý tín hiệu rồi lưu tạm thời vào SDRAM, sau đó đọc dữ liệu từ SDRAM xuất ra màn hình. Quá trình đọc, ghi dữ liệu của SDRAM được diễn ra liên tục và hiển thị lên màn hình VGA tạo thành những video liên tục.

Để thực hiện được hệ thống thu và hiển thị ảnh trên nền FPGA cần tìm hiểu và xây dựng, thiết kế các khối sau:

– Khối xử lý trung tâm, sử dụng chip xử lý FPGA để thực hiện các lệnh điều khiển đọc tín hiệu từ cảm biến hình ảnh, đọc, ghi dữ liệu vào SDRAM và điều khiển hiển hình hình ảnh lên VGA.

– Khối thu thập dữ liệu hình ảnh, chính là khối cảm biến hình ảnh. Khối này có chức năng quét các điểm ảnh thông qua ống kính và mã hóa tín hiệu, gửi khung dữ liệu điểm ảnh tới khối xử lý trung tâm.

– Khối lưu trữ dữ liệu tạm thời, khối này có nhiệm vụ lưu lại những điểm ảnh thu được từ bộ cảm biến ảnh, trong trường hợp này ta sử dụng bộ nhớ SDRAM để lưu lại tạm thời những dữ liệu ảnh thu được.

– Khối hiển thị hình ảnh, khối này chỉ có nhiệm vụ hiển thị lại những hình ảnh thu được từ cảm biến, hình ảnh được hiển thị với kích thước khung hình theo chuẩn VGA 480x640 pixel.

Ngoài việc xây dựng, thiết kế các khối thực hiện chức năng cho hệ thống thu và hiển thị ảnh cần tìm hiểu về phần mềm lập trình cho phần cứng FPGA, Quartus II của hãng Altera.

1.2 Hướng nghiên cứu của đề tài

Hiện nay đã có một số hãng thiết kế chip xử lý đã hỗ trợ kỹ thuật điều khiển thiết bị bên ngoài bằng cách sử dụng các nguồn tài nguyên phần cứng FPGA, do đó giúp giảm sự phức tạp phần cứng trong thiết kế số. Hơn nữa, FPGA hỗ trợ thực hiện xử lý tín hiệu kỹ thuật số. Các tính năng quan trọng nhất của FPGA là hiệu suất tốc độ cao, kiểm soát tốc độ hoạt động và hoạt động của các hệ thống phần mềm theo cách thức rõ ràng. FPGA xử lý tín hiệu kỹ thuật số là một giải pháp tốt cho xử lý số lượng lớn dữ liệu với tốc độ cao. FPGA cho tính năng cấu hình, tạo thành hệ thống DSP dễ dàng để kiểm tra và sửa đổi. Ngày nay, việc sử dụng FPGA thay thế cho hệ thống DSP đã được phát triển phổ biến cho các hệ thống kỹ thuật số, xử lý với công nghệ mới phát triển được thể hiện như.

(1) Hiệu quả: nhà sản xuất thiết bị chính tiếp tục bổ sung các thư viện cốt lõi. Thiết kế có thể tận dụng lợi thế của việc thử nghiệm và tối ưu hóa để đảm bảo tính chính xác của hệ thống. Đây là giải pháp cho hoàn thành thiết kế các hệ thống phức tạp trên chip, để cải thiện tính chính xác và hiệu quả của việc thiết kế.

(2) Mức tiêu thụ điện năng thấp: ứng dụng phát triển sản phẩm di động, thiết bị lập trình yêu cầu tiêu thụ điện năng thấp. Việc thiết kế Chip đang chuyển theo hướng phát triển công suất thấp

(3) Hệ thống on-chip: sự phát triển của công nghệ làm cho các hệ thống có thể được tích hợp trên chip.

1.3 Tổng quan về hệ thống thu và hiển thị ảnh

1.3.1 Tổng quan về các thiết bị logic lập trình

Ngày nay khoa học kỹ thuật trên thế giới liên tục phát triển, mà lĩnh vực điện tử luôn chiếm vị trí hàng đầu. Bước khởi đầu mang một ý nghĩa quan trọng, đó là sự ra đời của linh kiện chất bán dẫn chính là tiền đề cho hướng phát triển công nghệ điện tử.

Với xu hướng phát triển đó thì việc tích hợp linh kiện bán dẫn trong một đơn thể (IC) ngày càng được chú trọng, nhằm đáp ứng sự phát triển ngày càng cao của khoa học kỹ thuật, cũng như những ứng dụng thực tế.

Khi xuất xưởng, các IC thường được tích hợp sẵn với những chức năng riêng biệt, khi đó người sử dụng phải chọn lựa linh kiện sao cho việc thiết kế mạch hiệu quả nhất. Nhưng do độ tích hợp của IC cũng có giới hạn, và để linh hoạt hơn trong việc thực hiện những chức năng của người thiết kế, cũng như mối quan hệ mật thiết giữa nhà sản xuất và người sử dụng, cụ thể là tối ưu hóa khả năng ứng dụng của IC, nhà sản xuất đã cho ra một loại linh kiện đặc biệt mà chức năng của nó sẽ được người thiết kế quy định chứ không phải là nhà sản xuất. Linh kiện đó được gọi chung là PLD (Programmable Logic Device - Thiết bị logic lập trình được).

Chúng ta sẽ khảo sát linh kiện PLD qua các IC cụ thể như PAL (Programmable Array Logic devices), GAL (Generic Array Logic devices) ... Các IC PAL, GAL với độ tích hợp rất cao nên có thể thay thế hầu hết các loại IC TTL. Điều quan trọng trong những IC này là chức năng của nó sẽ được người thiết kế quy định cho chính những ứng dụng sao cho kinh tế và hiệu quả nhất.

Để thực hiện được việc thiết kế những ứng dụng trên IC PAL GAL... đòi hỏi người sử dụng cần phải kết hợp kiến thức cả về kỹ thuật số lẫn các ngôn ngữ lập trình cho thiết bị.

PLD thuộc họ bộ nhớ hàm (Funtion Memory). PLD có dung lượng tương đối lớn, có kết cấu đơn giản nhất trong các linh kiện logic. Thông thường PLD cho phép người thiết kế tạo cho nó những chức năng riêng biệt, bởi khi xuất xưởng nhà sản xuất chưa tạo cho nó một ứng dụng nào.

Cấu trúc mạch bên trong của một PLD thường là một chuỗi hình chữ nhật gồm những phần tử giống nhau (Identical Cell - ô nhớ đồng dạng). Hai mảng AND - OR có thể lập trình được nhờ tập hợp ngẫu nhiên các cổng logic và phần tử nhớ (OLMC-Output Logic Macro Cell).

PLD là mạch tích hợp của “SSI and MSI’ nên tính năng hoạt động của PLD linh hoạt, dễ sử dụng, dễ thiết kế và diện tích mạch giảm đáng kể so với việc thiết kế mạch bằng các IC rời chứa các cổng logic.

Khi dùng PLD việc thiết kế dễ dàng nhanh chóng nhờ nó có những phần mềm chuyên dụng đảm nhiệm, làm cho công việc thiết kế logic đơn giản hơn. Ta cũng dễ dàng sửa lỗi chương trình, bổ sung, thay đổi cấu hình thiết kế bên trong để thực hiện

một chức năng ứng dụng khác. Công nghệ linh kiện PLD sản xuất bằng EECMOS (Electrically Erasable CMOS) tạo khả năng lập trình lại nhiều lần với tốc độ cao, công suất tiêu tán thấp, phương pháp lập trình đơn giản, giá thành thấp hơn mạch rời tương đương.

1.3.2 FPGA và các ưu, nhược điểm

1.3.2.1 Sơ lược về FPGA

Field Programmable Gate Array (FPGA) [7], [10], [11] là vi mạch dùng cấu trúc mảng phần tử logic mà người dùng có thể lập trình được. Vi mạch FPGA được cấu thành từ các bộ phận:

- Các khối logic cơ bản lập trình được (logic block)
- Hệ thống mạch liên kết lập trình được
- Khối vào/ra (IO Pads)
- Phần tử thiết kế sẵn khác như DSP slice, RAM, ROM, nhân vi xử lý...

FPGA cũng được xem như một loại vi mạch bán dẫn chuyên dụng ASIC [11], nhưng nếu so sánh FPGA với những ASIC thì FPGA không đạt được mức độ tối ưu như những loại này, và hạn chế trong khả năng thực hiện những tác vụ đặc biệt phức tạp, tuy vậy FPGA ưu việt hơn ở chỗ có thể tái cấu trúc lại khi đang sử dụng, công đoạn thiết kế đơn giản do vậy chi phí giảm, rút ngắn thời gian đưa sản phẩm vào sử dụng.

Còn nếu so sánh với các dạng vi mạch bán dẫn lập trình được dùng cấu trúc mảng phần tử logic như PLA, PAL, CPLD thì FPGA ưu việt hơn, thể hiện ở các đặc điểm: tác vụ tái lập trình của FPGA thực hiện đơn giản hơn, khả năng lập trình linh động hơn, và khác biệt quan trọng nhất là kiến trúc của FPGA cho phép nó có khả năng chứa khối lượng lớn cổng logic (logic gate), so với các vi mạch bán dẫn lập trình được trước nó.

Thiết kế hay lập trình cho FPGA được thực hiện chủ yếu bằng các ngôn ngữ mô tả phần cứng như HDL, VHDL, Verilog, AHDL [9], [13] các hãng sản xuất FPGA lớn như Xilinx, Altera thường cung cấp các gói phần mềm và thiết bị phụ trợ cho quá trình thiết kế, cũng có một số các hãng khác cung cấp các gói phần mềm kiểu này như Synopsys [22], Synplify [23], [24]... Các gói phần mềm này có khả năng

thực hiện tất cả các bước của toàn bộ quy trình thiết kế IC chuẩn, với đầu vào là mã thiết kế trên HDL (còn gọi là mã RTL).

Ứng dụng của FPGA bao gồm: xử lý tín hiệu số DSP, các hệ thống hàng không, vũ trụ, quốc phòng, tiền thiết kế mẫu ASIC (ASIC prototyping), các hệ thống điều khiển trực quan, phân tích nhận dạng ảnh, nhận dạng tiếng nói, mật mã học, mô hình phần cứng máy tính... Do tính linh động cao trong quá trình thiết kế cho phép FPGA giải quyết lớp những bài toán phức tạp mà trước kia chỉ thực hiện nhờ phần mềm máy tính, ngoài ra nhờ mật độ cổng logic lớn FPGA được ứng dụng cho những bài toán đòi hỏi khối lượng tính toán lớn và dùng trong các hệ thống làm việc theo thời gian thực.

1.3.2.2 So sánh FPGA/CPLD và ASIC[11]

FPGA (field programmable gate array) là một phần quan trọng của công nghệ điện tử, sự phát triển nhanh chóng của công nghệ điện tử không thể thiếu nó. Trong thực trạng hiện nay ứng dụng mạch kỹ thuật số nhiều hơn và rộng rãi hơn, mức độ phức tạp cũng cao hơn, yêu cầu chung về tích hợp vi mạch số lớn hơn dẫn đến sự gia tăng của khối lượng của hệ thống chậm lại, độ tin cậy là khó được đảm bảo. Ngoài ra, các sản phẩm kỹ thuật số với thời gian sản xuất ngắn, có thể thiết kế mạch khác nhau để thực hiện bất kỳ thay đổi nào trong một thời gian ngắn để đáp ứng yêu cầu chức năng mới, sử dụng một IC có ý nghĩa cho sự cần thiết việc thiết kế lại và tái đi dây. Vì vậy, các nhà thiết kế có thể thiết kế một chip vi mạch tích hợp chuyên dụng (ASIC) cho những ứng dụng riêng, với chu trình thiết kế càng ngắn càng tốt, và tính linh hoạt. Trong trường hợp này, lĩnh vực thiết bị logic lập trình đã được ra đời. Trong số đó, việc sử dụng các mảng cổng lập trình (FPGA) và thiết bị logic lập trình phức tạp (CPLD) được sử dụng rộng rãi nhất.

Trong ngành công nghiệp điện tử, với các yêu cầu về chi phí của sản phẩm, hiệu suất và thời gian sản xuất để quyết định xem có đáp ứng nhu cầu thị trường, để đạt được thị phần lớn với nhiều người dùng hơn. Để phát triển tiềm năng của FPGA và ASIC ta cần phân tích các khía cạnh trên.

1) Chức năng sản phẩm: FPGA với lợi thế có các đặc tính thiết yếu trong hội nhập và tính linh hoạt. Ngày nay, các nhà sản xuất sử dụng công nghệ ASIC phải

xem xét các vấn đề khả năng nâng cấp tương thích của sản phẩm, việc sử dụng FPGA cho giải quyết vấn đề này không còn là mối quan tâm cho các sản phẩm.

2) Chi phí: Trên thị trường hiện nay để có một ASIC 0.18mm cần chi phí từ 100-300USD. Trong khi các FPGA có giá thành thấp hơn rất nhiều so với một ASIC.

3) Hiệu suất: FPGA trong việc thực hiện, phát triển nhanh chóng FPGA có thể đáp ứng hầu hết các yêu cầu thiết kế, ở một số ứng dụng đã được thay thế bởi ASIC.

4) Quá trình phát triển: để ASIC phát triển một chip mất khoảng thời gian 12 – 18 tháng, so với FPGA quá trình phát triển sản phẩm thường là 3 – 6 tháng.

1.3.2.3 FPGA và CPLD

FPGA và CPLD thiết bị lập trình ASIC, chúng có rất nhiều điểm chung, nhưng cũng có những đặc điểm riêng vì sự khác biệt về cấu trúc:

1) FPGA sử dụng SRAM để cấu hình chức năng, có thể được lập trình lại, dữ liệu SRAM sẽ bị mất, sự cấu hình lại là cần thiết. Để lưu lại dữ liệu cấu hình FPGA cần có EPROM để ghi dữ liệu. Các CPLD sử dụng công nghệ EEPROM, hệ thống được hỗ trợ lưu dữ liệu, dữ liệu sẽ không bị mất, và áp dụng cho các dữ liệu được bảo mật.

2) FPGA có cấu tạo từ các logic-cell. Về cơ bản một logic-cell gồm một bảng tra (LUT), một Flip-Flop và một mux 2 sang 1 (có thể bỏ qua Flip-Flop nếu muốn). Một LUT giống như một RAM nhỏ có thể thực thi một chức năng logic nào đó và LUT có các ngõ vào (input). Các logic-cell được kết nối với nhau thông qua các "tài nguyên liên kết", là các dây nối và mux được đặt xung quanh logic-cell. Mỗi logic-cell có thể nhỏ nhưng có rất nhiều các kết nối đến chúng để có thể tạo ra các chức năng lôgic phức tạp.

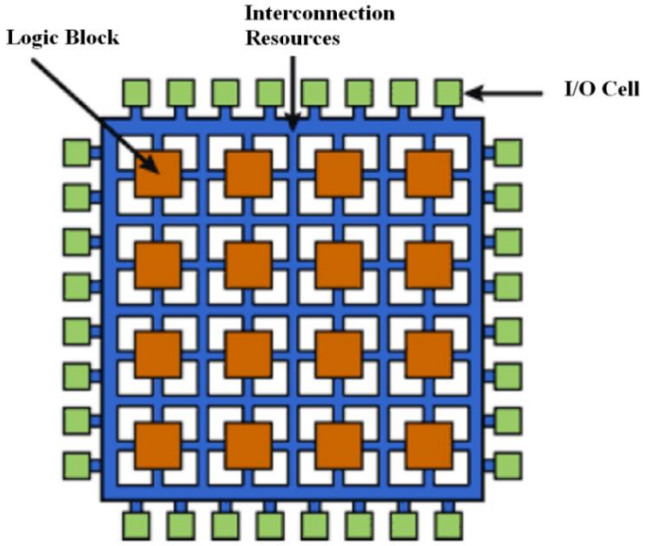
3) Bên cạnh các kết nối thông thường thì các "tài nguyên kết nối đa năng" cũng được thêm vào. Trong FPGA, các logic-cell liền kề nhau có các đường kết nối nhanh chuyên dụng (fast dedicated lines). Loại đường nhanh chuyên dụng phổ biến nhất là "chuỗi nhớ" (carry chains). Chuỗi nhớ cho phép tạo các chức năng toán học (như bộ đếm và bộ cộng) rất hiệu quả với tài nguyên logic thấp và tốc độ xử lý cao.

Với các công nghệ thấp hơn như PAL hay CPLD thì không có các "chuỗi nhớ" vì vậy tốc độ bị giới hạn khi các xử lý toán học được yêu cầu.

1.3.2.4 Kiến trúc FPGA

Cấu trúc tổng thể của FPGA bao gồm [7]:

- Các khối Logic
- Hệ thống liên kết mạch
- Các phần tử tích hợp sẵn

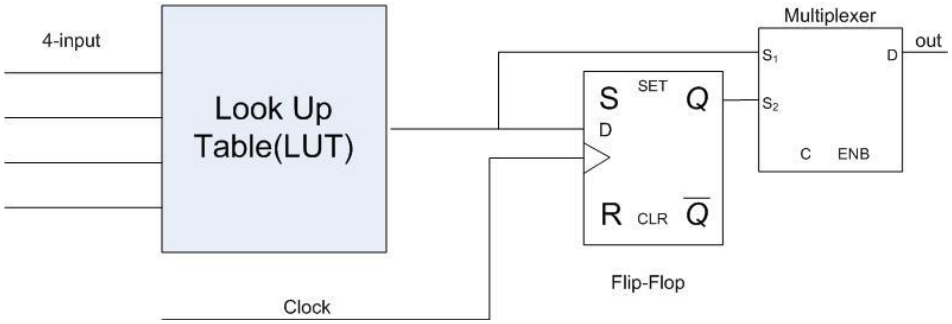


Hình 1.1: Cấu trúc tổng thể FPGA

Mỗi nhà sản xuất FPGA có cấu trúc riêng, nhưng nhìn chung cấu trúc được thể hiện giống như trong hình bên trên. Cấu trúc FPGA bao gồm có configuration logic blocks (CLBs), configurable I/O blocks (IOB), và programmable interconnect. Và tất nhiên, chúng có mạch clock để truyền tín hiệu clock tới các logic block, và thêm vào đó có các logic resources như ALUs, memory và có thể có cả decoders. Các phần tử lập trình được của FPGA có 2 dạng cơ bản là các RAM tĩnh (Static RAM) và anti-fuses.

Configurable Logic Blocks:

Configurable Logic Blocks (CLBs) bao gồm các Look-Up Tables (LUTs) rất linh động có chức năng thực thi các logic và các phần tử nhớ dùng như là các flip-flop hoặc các chốt (latch). CLB thực hiện phần lớn các chức năng logic như là lưu trữ dữ liệu,…

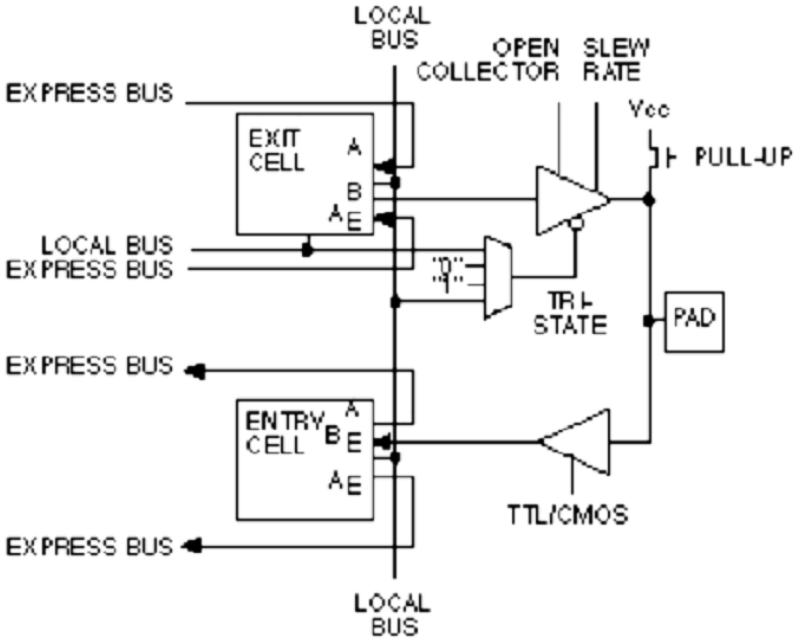


Hình 1.2 Khối Logic FPGA

FPGA chứa trong nó rất nhiều khối logic có thể tái cấu hình CLB Configurable Logic Blocks) được liên kết với nhau bằng các liên kết khả trình (Programmable Interconnect). Các khối vào ra được phân bố xung quanh chip tạo thành các liên kết với bên ngoài. Bên trong khối logic CLB có bảng LUT (Look – Up Table) và các phần tử nhớ (Flip Flop hoặc bộ chốt). LUT (Look up table) là khối logic có thể thực hiện bất kì hàm logic nào từ 4 đầu vào, kết quả của hàm này tùy vào mục đích mà gửi ra ngoài khối logic trực tiếp hay thông qua phần tử nhớ flip – flop.

Configurable I/O Blocks:

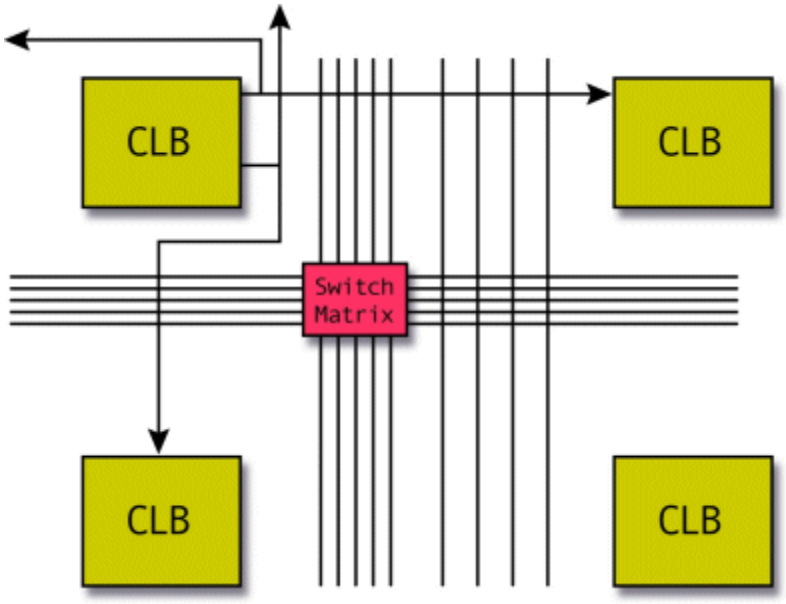
Input/Output Blocks (IOBs) điều khiển dòng dữ liệu giữa các chân vào ra I/O và các logic bên trong của FPGA. Nó bao gồm có các bộ đệm vào và ra với 3 trạng thái và điều khiển ngõ ra dạng open collector. Phần lớn là có điện trở kéo lên ở ngõ ra và thỉnh thoảng lại có trở kéo xuống. IOBs hỗ trợ luồng dữ liệu 2 chiều (bidirectional data flow) và hoạt động logic 3 trạng thái (3 state). Hỗ trợ phần lớn các chuẩn tín hiệu, bao gồm một vài chuẩn tốc độ cao, như Double Data-Rate (DDR).



Hình 1.3: Khối Configurable Logic FPGA

Programmable Interconnect:

Interconnect ở FPGA khác so với ở CPLD, tuy nhiên lại giống với gate array ASIC. Có một line dài được dùng để nối các CLBs quan trọng mà chúng lại ở cách xa nhau mà không gây ra quá nhiều trễ. Chúng có thể được dùng như là các bus ở trong chip. Có các line ngắn được dùng để liên kết các CLBs riêng rẽ nhưng đặt gần nhau. Và cũng thường có vài ma trận chuyển đổi (switch matrices), giống như trong CPLD, nối giữa các line dài và ngắn lại với nhau theo một số cách đặc biệt. Các chuyển đổi lập trình được (Programmable switches) bên trong chip cho phép kết nối giữa CLBs tới các interconnect line và giữa interconnect line với các line khác và với switch matrix. Các bộ đệm 3 trạng thái được dùng để kết nối phần lớn các CLBs với các line dài (long line), tạo nên các bus. Các long line đặc biệt, gọi là các line clock toàn cục (global clock lines), được thiết kế đặc biệt cho trở kháng thấp và nhờ đó mà thời gian lan truyền nhanh hơn. Chúng được kết nối với các bộ đệm clock và với mỗi phần tử được clock trong mỗi CLB. Đó là cách mà clock có thể phân phối bên trong FPGA.



Hình 1.4: Programmable Interconnect

Mạch đồng hồ (Clock Circuitry):

Các khối vào ra với bộ đệm clock high drive gọi là các clock driver, nằm rải rác xung quanh chip. Các bộ đệm này được nối với các chân clock vào và lái các tín hiệu clock vào các đường clock toàn cục (global clock line) như mô tả ở bên trên. Các đường clock được thiết kế sao cho thời gian thời gian lệch nhỏ nhất và thời gian lan truyền nhanh. Thiết kế đồng bộ là yêu cầu bắt buộc với FPGA, từ khi độ lệch tuyệt đối và trễ không được bảo đảm. Chỉ khi dùng các tín hiệu clock từ các bộ đệm clock thì thời gian trễ tương đối và thời gian lệch mới được đảm bảo.

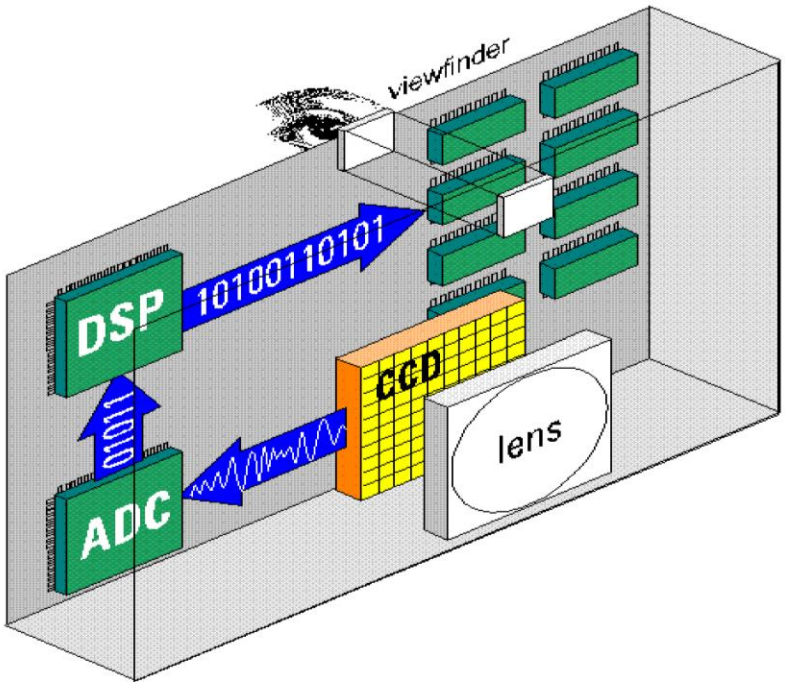
1.3.3 Kiến trúc cảm biến thu thập hình ảnh

Đây là bộ cảm biến ánh sáng nằm trong máy ảnh kỹ thuật số có tác dụng chuyển ánh sáng thu nhận từ môi trường bên ngoài sang tín hiệu điện. CCD (Charged Couple Device) [5] bao gồm hàng triệu tế bào quang điện, mỗi tế bào có tác dụng thu nhận thông tin về từng điểm ảnh (Pixel).

Để có thể thu được mầu sắc, máy ảnh kỹ thuật số sử dụng bộ lọc mầu (color filter) trên mỗi tế bào quang điện. Các tín hiệu điện tử thu được trên mỗi tế bào quang điện sẽ được chuyển đổi thành tín hiệu kỹ thuật số nhờ bộ chuyển đổi ADC (Analog to digital converter). Vào thời điểm hiện tại có hai loại bộ cảm biến ánh

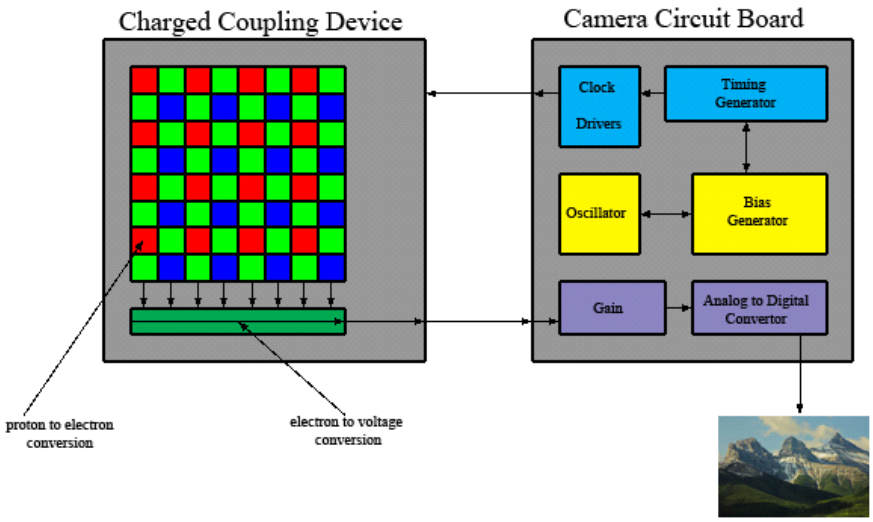
sáng: CCD (Charged Coupled Device) và CMOS (Complimentary Metal Oxide Semiconductor) [6].

Cả hai bộ phận cảm biến hình ảnh dùng công nghệ CCD và CMOS cùng có nhiệm vụ chuyển đổi tín hiệu ánh sáng sang tín hiệu điện. Một điều đơn giản có thể hiểu dùng trong máy ảnh kỹ thuật số là có một mảng 2D gồm hàng nghìn, hàng triệu những tế bào năng lượng mặt trời, mỗi một tế bào có nhiệm vụ chuyển ánh sáng từ một phần trên bức ảnh thành tín hiệu điện



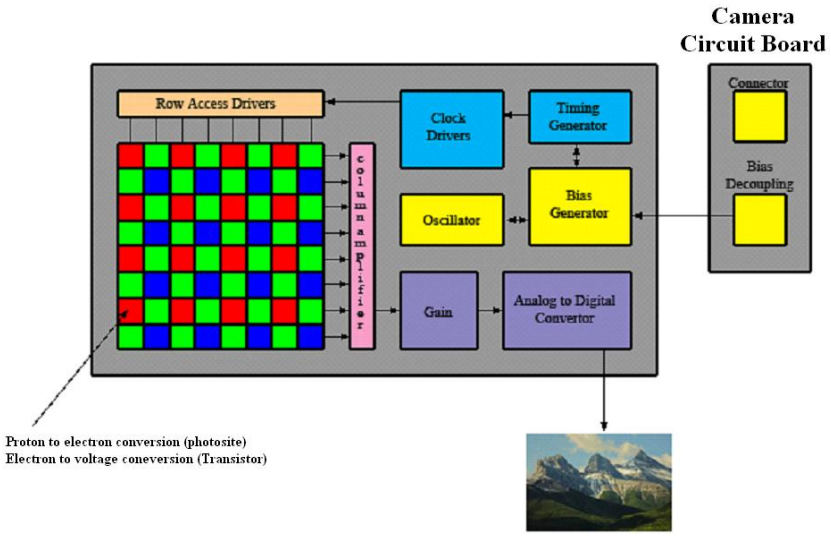
Hình 1.5: Sơ đồ tổng quan cảm biến hình ảnh CCD

Bước tiếp theo là đọc giá trị tín hiệu điện tại mỗi tế bào quang điện trong hình ảnh. Trong thiết bị CCD, điện áp nạp trên thực tế được qua một chip và được đọc ở góc của một mảng. Bộ chuyển đổi ADC (Analog – to – Digital Converter) sẽ biến giá trị mỗi Pixel thành giá trị số tương ứng [5].



Hình 1.6: Sơ đồ khôi cảm biến hình ảnh CCD

Trong hầu hết những thiết bị CMOS có vài Transitor cho mỗi một Pixel [6] và được khuếch đại và chuyển tín hiệu tới mạch nạp truyền thống. CMOS đạt được nhiều sự linh hoạt bởi vì mỗi Pixel được đọc giá trị riêng biệt.



Hình 1.7: Sơ đồ khôi cảm biến hình ảnh CMOS

Giá thành sản xuất CCD thường đắt hơn so với CMOS, nguyên nhân chủ yếu là do CCD đòi hỏi phải có dây chuyền sản xuất riêng trong khi có thể sử dụng dây chuyền sản xuất chip, bảng mạch thông thường để sản xuất CMOS.

Những CCD được chế tạo đặc biệt để có thể chuyển tín hiệu nạp tới chip mà không bị méo tín hiệu. Sự xử lý này để sản xuất những cảm biến với chất lượng cao với độ tin cậy cao và độ nhạy sáng cao.

Như một máy ảnh bình thường, một máy ảnh số có một thấu kính và một cửa trập cho phép ánh sáng đi qua. Nhưng có một điểm khác là ánh sáng tác dụng lên một mảng của những tế bào quang điện hoặc những ô cảm quang thay cho phim. Mảng tế bào quang điện là một chip khoảng 6 - 11mm chiều ngang. Mỗi bộ cảm biến hình ảnh là một thiết bị tích điện (Chargd Couple Device – CCD), nó chuyển đổi ánh sáng thành điện tích. Sự tích điện được lưu dưới dạng thông tin tương tự rồi được số hoá bởi một thiết bị khác gọi là bộ biến đổi tương tự - số (Analog to Digital Converter - ADC).

Mỗi phần tử quang điện trong mảng của hàng ngàn phần tử, tạo ra một pixel và mỗi pixel chứa một vài thông tin được lưu giữ. Một số máy ảnh số sử dụng bộ cảm biến hình ảnh bằng chip CMOS (Complementary Metal Oxide Semiconductor). Công nghệ CMOS liên quan tới quá trình thiết kế bộ cảm biến. Quá trình này cũng giống quá trình sản xuất hàng loại DRAM và những bộ vi xử lý nên bộ cảm biến CMOS rẻ hơn và dễ làm hơn nhiều so với bộ cảm biến CCD.

Những lợi thế khác của bộ cảm biến CMOS là chúng tiêu thụ ít điện hơn và có thể kết hợp những mạch khác trên cùng chip. Những tính năng bổ sung của loại chip này có thể bao gồm bộ chuyển đổi tương tự – số, tính năng điều khiển camera, nén hình ảnh hay chống rung.

Tuy nhiên, những mạch bổ sung này sử dụng không gian bình thường được sử dụng cho thiết bị đo sáng. Điều này làm cho bộ cảm biến kém nhậy sáng hơn, tạo ra những bức ảnh chất lượng thấp hơn khi chụp ở trong nhà hoặc trong những điều kiện thiếu sáng khác.

Những CMOS nói một cách khác được sản xuất xử lý một cách truyền thống, cùng phương pháp xử lý sản xuất các bộ vi xử lý. Bởi vì quá trình sản xuất khác nhau nên những cảm biến CCD và CMOS cũng có một số vấn đề khác nhau:

- Những cảm biến CCD, như đã đề cập ở trên được sản xuất với chất lượng cao để đạt được độ nhiễu thấp nhất. Những cảm biến CMOS được sản xuất bằng phương pháp truyền thống cho chất lượng hình ảnh thấp do bị ảnh hưởng nhiễu cao.
- Bởi vì mỗi Pixel trong cảm biến CMOS có vài Transistor do đó độ nhạy sáng của chip CMOS thấp hơn.

- Cảm biến hình ảnh CMOS tiêu thụ năng lượng thấp.
- Những cảm biến CCD tiêu thụ nhiều năng lượng trong quá trình xử lí, cảm biến CCD têu thụ điện gấp 100 lần so với cảm biến CMOS tương đương.

Dựa trên những sự khác nhau đó mà có thể xem như bộ cảm biến CCD trong máy ảnh ký thuật số cho chất lượng hình ảnh cao với nhiều Pixel với độ nhạy sáng cao. Những cảm biến dùng công nghệ CMOS tiêu thụ năng lượng thấp hơn, độ phân giải thấp hơn và độ nhạy sáng kém nhưng bên cạnh đó nó dùng được Pin lâu hơn vì tiêu thụ năng lượng thấp. Ngày nay những máy ảnh kỹ thuật số dùng công nghệ CMOS đã được cải tiến và chất lượng gần đạt được so với CCD.

1.4 Cấu trúc tổng quan hệ thống

1.4.1 Ý tưởng thiết kế hệ thống

Nội dung nghiên cứu nhằm mục đích thiết kế hệ thống thu thập hình ảnh thời gian thực trên nền FPGA. Sử dụng một chip FPGA để thiết kế hệ thống điều khiển thu thập hình ảnh.

Sự phát triển FPGA làm cho công nghệ chụp ảnh được cải thiện, bài luận văn này sẽ xây dựng trên cấu trúc cơ bản của hệ thống thu thập hình ảnh và thực hiện phần cứng của các khối chức năng khác nhau dựa trên nền FPGA. Nói chung, quá trình thiết kế hệ thống trên nền FPGA như sau:

(1) Định nghĩa chức năng hệ thống và phân chia khối chức năng logic, bao gồm việc xem xét định nghĩa cổng và tái sử dụng hệ thống.

(2) Thiết kế sơ bộ hệ thống.

(3) Thiết kế sơ bộ sơ đồ thời gian, bao gồm xung nhịp và cấu trúc đường truyền, ...

(4) Lựa chọn chip

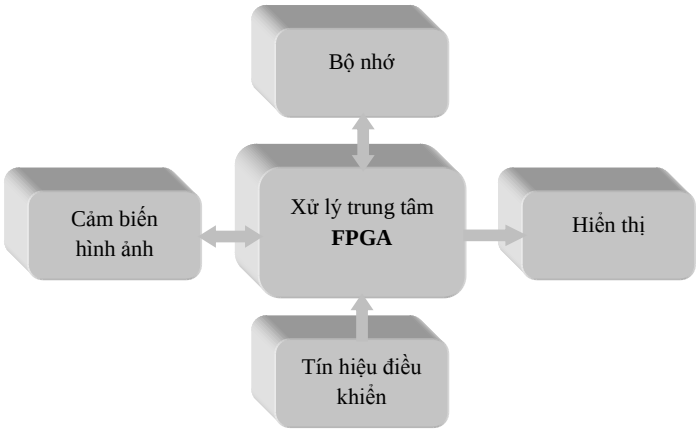
(5) Phân chia các khối chức năng logic với định nghĩa giao diện cho từng khối.

(6) Thiết kế phải tuân theo quá trình thiết kế cấu trúc hoàn chỉnh FPGA.

(7) Tổng hợp các khối và mô phỏng thử nghiệm, bao gồm cả thời gian mô phỏng và thử nghiệm chức năng của toàn bộ hệ thống.

1.4.2 Cấu trúc tổng quan hệ thống

Bài luận văn phân tích việc ghi lại hình ảnh truyền thống và kiến trúc hệ thống xử lý dựa trên các yêu cầu bằng cách sử dụng FPGA thiết kế hệ thống thu thập hình ảnh tốc độ cao với quá trình thời gian thực, dựa trên đề tài nghiên cứu. Về nguyên tắc, thu thập hình ảnh và hệ thống xử lý chủ yếu được chia thành bốn phần: thu thập, xử lý và hiển thị hình ảnh. Phần thu thập hình ảnh được hoàn thành bởi các cảm biến hình ảnh, xử lý hình ảnh và đầu ra được thực hiện bởi FPGA và khối chức năng bên ngoài. Trong việc ghi lại hình ảnh và xử lý nội bộ có thể được chia thành năm khối chức năng: khối cảm biến hình ảnh, khối xử lý trung tâm FPGA, khối lưu dữ liệu hình ảnh, khối hiển thị hình ảnh, khối tín hiệu điều khiển. Hình 1.8 dưới đây thể hiện sơ đồ tổng quan các khối trong hệ thống:



Hình 1.8: Sơ đồ khối tổng quan hệ thống

(1) Khối cảm biến hình ảnh: cảm biến hình ảnh chuyển đổi tín hiệu quang thành tín hiệu điện theo tín hiệu hình ảnh đầu ra, phân loại cảm biến này có thể được chia thành các cảm biến analog và kỹ thuật số.

(2) Khối xử lý trung tâm: khối này điều khiển toàn bộ hệ thống, chức năng xử lý tín hiệu hình ảnh đầu ra cảm biến, điều khiển lưu dữ liệu xuống bộ nhớ và điều khiển hiển thị hình ảnh ra màn hình.

(3) Khối lưu dữ liệu hình ảnh: khối có chức năng lưu dữ liệu hình ảnh tạm thời trong quá trình bộ xử lý trung tâm đọc dữ liệu ảnh từ bộ cảm biến.

(4) Khối tín hiệu điều khiển: khối tương tác với người dùng, cho phép người dùng kích hoạt các chế độ hoạt động của hệ thống.

(5) Khối giao diện VGA: khối này có nhiệm vụ hiển thị kết quả hình ảnh thu thập được từ bộ cảm biến ảnh

1.5 Tóm tắt chương

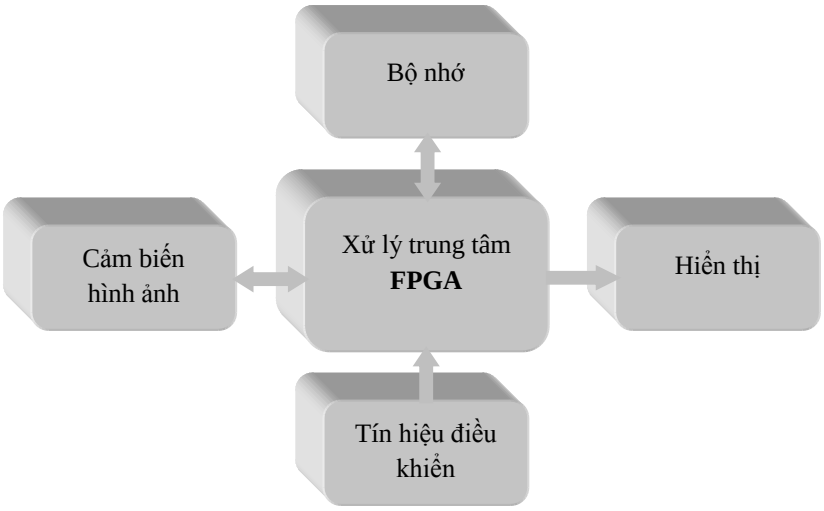
Trong chương trình bày về tính cấp thiết của đề tài và tình hình nghiên cứu các hệ thống thu và hiển thị ảnh ở trong và ngoài nước. Chương này giới thiệu tổng quan về đề tài nghiên cứu, mô tả các đặc tính và cách sử dụng FPGA, thiết bị logic lập trình được, FPGA, ASIC, CPLD và so sánh một số cấu trúc đơn giản đã thực hiện, cấu trúc tổng quan cảm biến hình ảnh. Sau đó, thiết kế kiến trúc tổng quan hệ thống thu thập hình ảnh, mỗi cấu trúc các khối thực hiện phần cứng làm nền tảng để thực hiện hệ thống.

Chương 2: PHÂN TÍCH LỰA CHỌN HỆ THỐNG

Như đã trình bày trong chương 1, thiết kế tổng thể hệ thống thu thập và hiển thị hình ảnh, phân tích lựa chọn thiết bị phần cứng và phương pháp thiết kế sơ đồ phần cứng các khối sẽ được thảo luận trong chương này. Trong chương đề cập đến sự lựa chọn thiết bị cho thiết kế mạch phần cứng và phân tích thiết bị trong các khối như: chip FPGA, cảm biến hình ảnh, bộ nhớ lưu trữ dữ liệu ảnh và giao diện hiển thị ảnh.

2.1 Cấu trúc phần cứng hệ thống

Các sơ đồ mạch phần cứng của các thiết kế hiện nay chủ yếu bao gồm các mạch thu thập hình ảnh, bộ nhớ SDRAM, VGA (video GraphicsArray) và nguồn cung cấp điện mạch.



Hình 2.1: Sơ đồ khối tổng quát hệ thống thu và hiển thị ảnh

Toàn bộ cấu trúc phần cứng của việc thu hình ảnh và hệ thống xử lý được thể hiện trong hình 2.1.

Trong đó, khối FPGA là khối điều khiển chính của hệ thống, chủ yếu là điều khiển đầu vào cảm biến thu hình ảnh và điều khiển đọc, ghi dữ liệu SDRAM, điều khiển hiển thị VGA, và phân tích và xử lý dữ liệu hình ảnh.

(1) Khối cảm biến thu hình ảnh, để thực hiện việc thu lại hình ảnh của mục tiêu và xử lý tín hiệu hình ảnh quang học.

(2) Bộ nhớ SDRAM để xử lý bộ đệm khung hình ảnh, dữ liệu trung gian và lưu trữ chương trình và làm giảm bớt tắc nghẽn dữ liệu.

(3) Một màn hình hiển thị hình ảnh chủ yếu bao gồm hai khối: khối chuyển đổi D/A và giao diện VGA.

(4) Khối tín hiệu điều khiển: khối tương tác người dùng, cho phép lựa chọn chế độ kích hệ thống.

(5) JTAG (Joint Test Action Group) là một giao thức kiểm tra tiêu chuẩn, chủ yếu được sử dụng cho lõi các chip thử nghiệm nội bộ. Mạch giao diện JTAG được sử dụng để hỗ trợ gỡ dối của module FPGA nội bộ.

(6) Giao diện nối tiếp để kết nối các thiết bị cấu hình nối tiếp được sử dụng để lưu dữ liệu cấu hình FPGA, làm cho FPGA khởi tạo một cách nhanh chóng có thể tải lại tất cả các dữ liệu.

(7) Nguồn cung cấp: cung cấp điện áp cần thiết cho toàn bộ thiết kế phần cứng.

Ở đây, các mạch giao diện nối tiếp, giao diện JTAG và mạch cung cấp điện là một phần của hệ thống giao diện ngoại vi. Các mạch thuộc hệ thống thu và hiển thị ảnh bao gồm các khối cảm biến hình ảnh, khối màn hình hiển thị hình ảnh giao diện VGA và chuyển đổi D/A, Bộ nhớ SDRAM lưu trữ dữ liệu FPGA xử lý. Hệ thống sẽ được phân tích chính chi tiết ở những phân tiếp theo.

2.2 Lựa chọn chip FPGA

FPGA module là phần cốt lõi của toàn bộ hệ thống, đây là khối xử lý, và điều khiển chính của hệ thống thông tin dữ liệu, xác định theo dõi các hình ảnh thu được và xử lý các hoạt động. Trên thị trường việt nam hiện nay có 2 hãng cung cấp chip FPGA chính đó là Xilinx và Altera. Thiết kế chip loại CycloneII của hãng Altera bao gồm năm mô hình của các sản phẩm chip: EP2C5 EP2C8 EP2C20, EP2C35, EP2C50 và EP2C70. Đặc điểm chip FPGA dòng CycloneII được hiển thị trong bảng 2.1.

Bảng 2.1: Thông số đặc trưng FPGA dòng CycloneII

Feature	EP2C5	EP2C8	EP2C20	EP2C35	EP2C50	EP2C70
M4K RAM blocks	26	36	52	105	129	250
Total RAM bits	119,808	165,888	239,616	483,840	594,432	1,152,000
Embedded multipliers	13	18	26	35	86	150
PLLs	2	2	4	4	4	4
Maximum user I/O pins	158	182	315	475	450	622
LEs	4,608	8,256	18,752	33,216	50,528	68,416

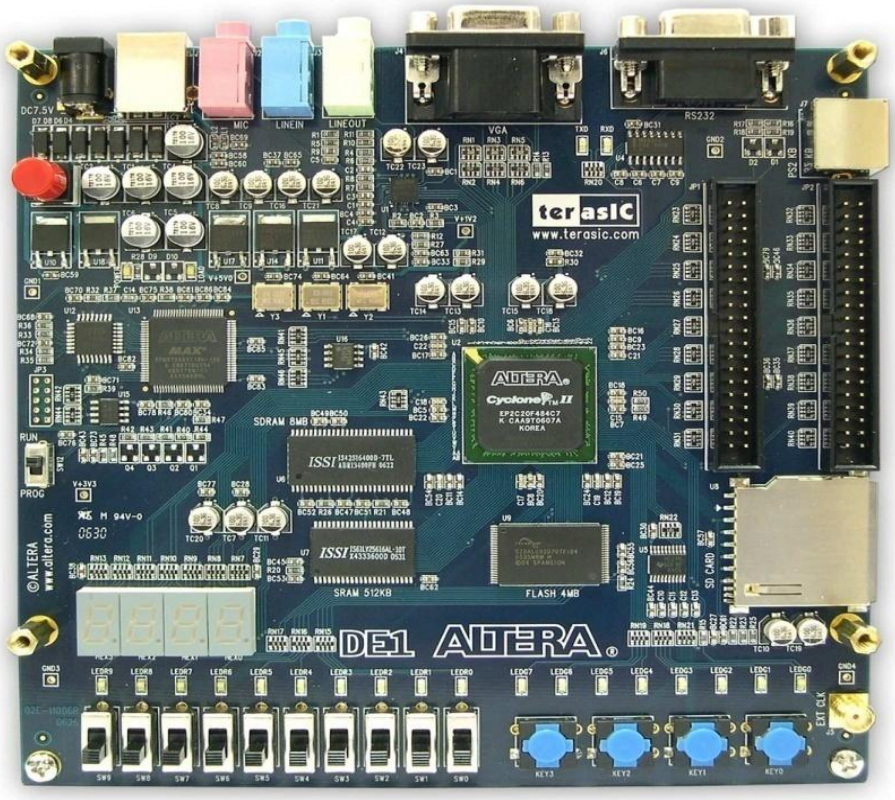
Dòng chip CycloneII bao gồm cả khối logic mảng (Logic Array Block, LAB), hệ số nhúng, và bộ nhớ, đồng hồ Phase-Locked Loop (PLL), số đơn vị đầu vào và đầu ra I/O của các logic. Cyclone II có tới 16 đồng hồ cho tốc độ xử lý mảng nhanh, hệ số nhúng, các khối bộ nhớ nhúng, và cung cấp đơn vị I/O, đồng hồ kiểm soát, FPGA cũng có thể được sử dụng như là việc sử dụng các tín hiệu đầu ra số tốc độ cao.

Xem xét các tài nguyên logic chip, dung lượng lưu trữ, tần số xung đồng hồ tối đa, số đầu vào ra I/O và kết hợp yếu tố giá thành, thiết kế hệ thống lựa chọn sử dụng chip FPGA EP2C20F484C7 [18]. Chip EP2C20F484C7 có các thông số đặc trưng như sau:

- 18,752 LEs
- 52 M4K RAM blocks
- 240K total RAM bits
- 26 embedded multipliers
- 4 PLLs
- 315 user I/O pins
- FineLine BGA 484 – pin package

Để thuận tiện cho việc nghiên cứu thực hiện hệ thống thu và hiển thị hình ảnh sử dụng Chip EP2C20F484C7 tác giả đã lựa chọn giải pháp sử dụng bộ Kit phát triển DE1 của hãng Altera để thực hiện việc nghiên cứu.

Board DE1 là board mạch phục vụ cho việc nghiên cứu và phát triển về các lĩnh vực logic số (digital logic), tổ chức máy tính (computer organization) và FPGA.



Hình 2.2: Board DE1

Board DE1 cung cấp khá nhiều tính năng hỗ trợ cho việc nghiên cứu và phát triển, dưới đây là thông tin chi tiết của một board DE1 [18]:

- FPGA:
  - Vi mạch FPGA Altera Cyclone II EP2C20F484C7.
  - Vi mạch Altera Serial Configuration – EPCS4
- Các thiết bị xuất nhập:
  - USB Blaster cho lập trình và điều khiển API của người dung; hỗ trợ cả 2 chế độlập trình JTAG và AS.
  - Cổng VGA-out.
  - Bộ điều khiển USB Host/Slave với cổng USB kiểu A và kiểu B.
  - Cổng nối PS/2 chuột/bàn phím.
  - Bộ giải mã/mã hóa âm thanh 24-bit chất lượng đĩa quang với jack cắm line-in, line-out, và microphone.

- 2 Header mở rộng 40-pin với lớp bảo vệ diode
- Cổng giao tiếp RS-232 và cổng nối 9-pin.

➢ Bộ nhớ:

- SRAM 512-Kbyte.
- SDRAM 8-Mbyte.
- Flash 4-MB
- Khe SD card.

➢ Switch, các đèn led, LCD, xung clock

- 4 nút nhấn, 10 nút gạt.
- 10 LED đỏ, 8 LED xanh, 4 Led 7 đoạn
- Bộ dao động 50-MHz và 27-MHz cho đồng hồ nguồn.

Thiết kế hệ thống FPGA bao gồm hai phương pháp: Thiết kế sơ đồ mạch và ngôn ngữ mô tả phần cứng. Thiết kế sơ đồ mạch thường được áp dụng cho các mạch nhỏ hơn và hệ thống trực quan, những hệ thống phức tạp hơn mạch được sử dụng ngôn ngữ lập trình để mô tả phần cứng các chế độ được thường được sử dụng để thể hiện các yêu cầu thiết kế. Ngôn ngữ mô tả hệ thống bao gồm 3 ngôn ngữ chính, Verilog HDL, VHDL và AHDL [9]. bằng cách sử dụng một loạt các module được mô tả các chế độ hoạt động của hệ thống, và sau đó sử dụng các ứng dụng các công cụ cho thiết kế và mô phỏng (EDA), sự kết hợp của module thành một netlist, sử dụng công cụ định tuyến các netlist vào cấu trúc mạch. Trong chương trình thiết kế và mô phỏng sử dụng phần mềm Quartus II với ngôn ngữ mô tả VHDL [18].

2.3 Lựa chọn cảm biến hình ảnh

Cảm biến hình ảnh là một phần quan trọng của các máy ảnh kỹ thuật số sử dụng chip CCD và CMOS, cảm biến hình ảnh được chia thành 2 loại khác nhau CCD và CMOS [5], [6].

Máy chụp ảnh sử dụng chip CCD nói chung có những ưu điểm sau [5]:

(1) Độ nhiễu thấp, độ nhạy cao: CCD có độ nhiễu rất thấp và độ nhiễu này có thể thay đổi bởi tỷ lệ tín hiệu trên nhiễu (SNR), CCD có đặc điểm độ nhạy cao, làm cho ứng dụng của nó ít bị tác động bởi môi trường.

(2) Độ phân giải cao: CCD có thể cảm nhận các đối tượng cực kỳ tốt, qua đó nâng cao chất lượng hình ảnh.

(3) Đặc tính tuyến tính tốt: Đặc tính tuyến tính tỷ lệ thuận với cường độ ánh sáng tới và tín hiệu đầu ra. Do đó, có thể phát hiện được ảnh trong bóng tối.

(4) Tín hiệu ít bị thay đổi trong quá trình xử lý hình ảnh, khôi phục lai ảnh ban đầu.

Cảm biến hình ảnh CMOS bắt đầu muộn hơn CDD, do những hạn chế về sự phát triển của quá trình sản xuất. Cho đến nay, với quá trình thương mại hóa và cải tiến liên tục của quá trình sản xuất, cảm biến CMOS không chỉ được cải thiện độ phân giải mà khã năng xử lý nhiễu cũng đã được cải thiện đáng kể. Ứng dụng cảm biến hình ảnh CMOS là rất rộng, bao gồm cả các máy ảnh kỹ thuật số, điện thoại di động, hội nghị truyền hình, hệ thống bảo mật thông minh, … So với CCD, CMOS có những đặc điểm sau đây [6]:

(1) COMS có kích thước nhỏ, độ tích hợp cao, tiêu thụ điện năng ít hơn so với CCD.

(2) Quá trình sản xuất tiêu chuẩn CMOS có thể sử dụng trực tiếp các thiết bị bán dẫn hiện có, không cần bổ sung đầu tư thêm thiết bị để cải thiện chất lượng của nó với sự phát triển của công nghệ bán dẫn.

(3) Giá thành thấp hơn so với CCD.

(4) CCD không thể chuyển đổi tín hiệu hình ảnh thành một tín hiệu kỹ thuật số trực tiếp, mà phải cần thêm chuyển đổi A/D. Mỗi phần tử quang của bộ cảm biến CMOS tích hợp bộ khuếch đại và ADC có thể được chuyển đổi trực tiếp tín hiệu analog thành một tín hiệu số tương ứng.

Dựa trên nhưng so sánh giữa hai lại cảm biến CCD và CMOS trong thiết kế hệ thống sử dụng chip cảm biến hình ảnh CMOS - OV9650 của hãng OmniVision để thiết kế cho khối thu thập hình ảnh [1], [2], [4].

OV9650 CAMERACHIP là một cảm biến hình ảnh CMOS điện áp thấp cung cấp đầy đủ chức năng của một máy ảnh SXGA đơn chip (1280x1024) và xử lý hình ảnh kích thước nhỏ. Các OV9650 cung cấp đầy đủ khung hình, định dạng hình ảnh

trong kiểu lấy mẫu hoặc cửa sổ 8-bit/10-bit [20], điều khiển thông qua giao diện điều khiển nối tiếp (SCCB) [19].

2.3.1 Giao diện cảm biến CMOS OV9650

OV9650 CAMERACHIP có một mảng hình ảnh, khả năng hoạt động lên đến 15 khung hình mỗi giây (fps) ở độ phân giải SXGA, người sử dụng có thể điều khiển các định dạng hình ảnh, chất lượng hình ảnh và chuyển giao dữ liệu đầu ra. Tất cả các yêu cầu xử lý hình ảnh chức năng, bao gồm cả điều khiển phơi ảnh, gamma, cân bằng trắng, độ bão hòa màu, điều khiển màu sắc, loại bỏ điểm ảnh màu trắng, loại bỏ tiếng ồn, … lập trình thông qua giao diện SCCB.

Ngoài ra, OmniVision CAMERACHIPS sử dụng công nghệ độc quyền để cải thiện chất lượng hình ảnh bằng cách giảm hoặc loại bỏ ánh sáng, nhiễu của nguồn điện ảnh hưởng đến chất lượng ảnh, chẳng hạn như nhòe mẫu hình cố định, nhiễu, vv… để tạo ra một hình ảnh màu sắc sạch, hình ảnh phải hoàn toàn ổn định.

2.3.2 Tính năng OV9650 [20]

Tính năng của cảm biến CMOS OV9650 được thể hiện với những đặc tính sau:

- Độ nhạy cao cho hoạt động ở chế độ ánh sáng thấp.
- Điện áp hoạt động thấp ứng dụng cho hệ thống nhúng.
- Sử dụng giao diện truyền dữ liệu nối tiếp SCCB.
- Hỗ trợ độ phân giải SXGA, VGA, QVGA, QQVGA, CIF, QCIF, QQCIF, và với định dạng cửa sổ đầu ra Raw RGB, RGB (GRB 4:2: 2), YUV (4:2:2) và YCbCr (4:2:2)
- VarioPixel phương pháp để định dạng lấy mẫu nhỏ (VGA, QVGA, QQVGA, CIF, QCIF, và QQCIF)
- Chức năng tự động điều khiển hình ảnh bao gồm: Tự động điều chỉnh độ sáng (AEC), tự động điều khiển hệ số khuếch đại (AGC), Hệ thống tự động cân bằng trắng (AWB), tự động lọc band (ABF), và tự động hiệu chỉnh mức đen (ABLC)
- Chất lượng hình ảnh điều khiển bao gồm cả độ bão hòa màu, màu sắc, gamma, độ sắc nét (cạnh tăng cường), điều chỉnh ống kính, loại bỏ điểm ảnh màu trắng, loại bỏ nhiễu và phát hiện độ sáng 50/60Hz

Ứng dụng:

- Điện thoại di động và hình ảnh
- Đồ chơi
- PC đa phương tiện
- Máy ảnh kỹ thuật số

2.3.3 Thông số kỹ thuật OV9650 [20]

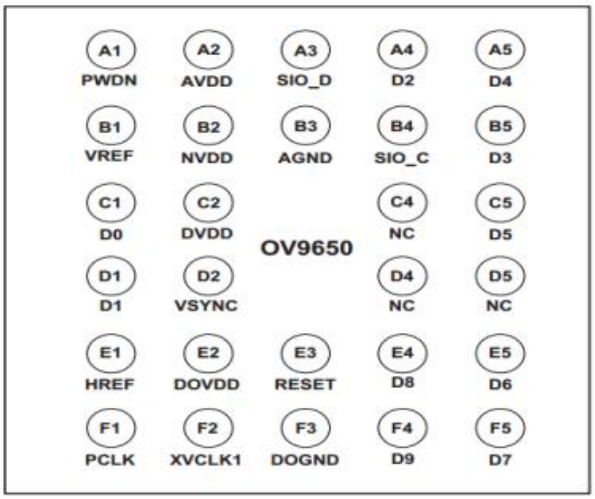
Các thông số kỹ thuật chính của cảm biến camera OV9650 được thể hiện trong bảng 2.2:

Bảng 2.2: Thông số kỹ thuật OV9650

Active Array Size		1300 x 1028
Power Supply	Core	1.8VDC +10%
	Analog	2.45 to 2.8 VDC
	I/O	2.5V to 3.3V
Power Requirements	Active	50 mW (15 fps, no I/O power)
	Standby	30 µW
Temperature Range	Operation	-20°C to 70°C
	Stable Image	0°C to 50°C
Output Formats (8-bit)		• YUV/YCbCr 4:2:2 • GRB 4:2:2 • Raw RGB Data
Lens Size		1/4"
Maximum Image Transfer Rate	SXGA	15 fps
	VGA	30 fps
	QVGA, QQVGA, CIF	60 fps
	QCIF, QQCIF	120 fps
Sensitivity		0.9 v/Lux-sec
S/N Ratio		40 dB

Dynamic Range	62 dB
Scan Mode	Progressive
Maximum Exposure Interval	1050 x t_{ROW}
Gamma Correction	Programmable
Pixel Size	3.18 µm x 3.18 µm
Dark Current	30 mV/s at 60°C
Well Capacity	28 K e
Fixed Pattern Noise	<0.03% of VPEAK-TO-PEAK
Image Area	4.13 mm x 3.28 mm
Package Dimensions	5095 x 5715 µm

2.3.4 Sơ đồ Pin OV9650



Hình 2.3: Sơ đồ Pin OV9650

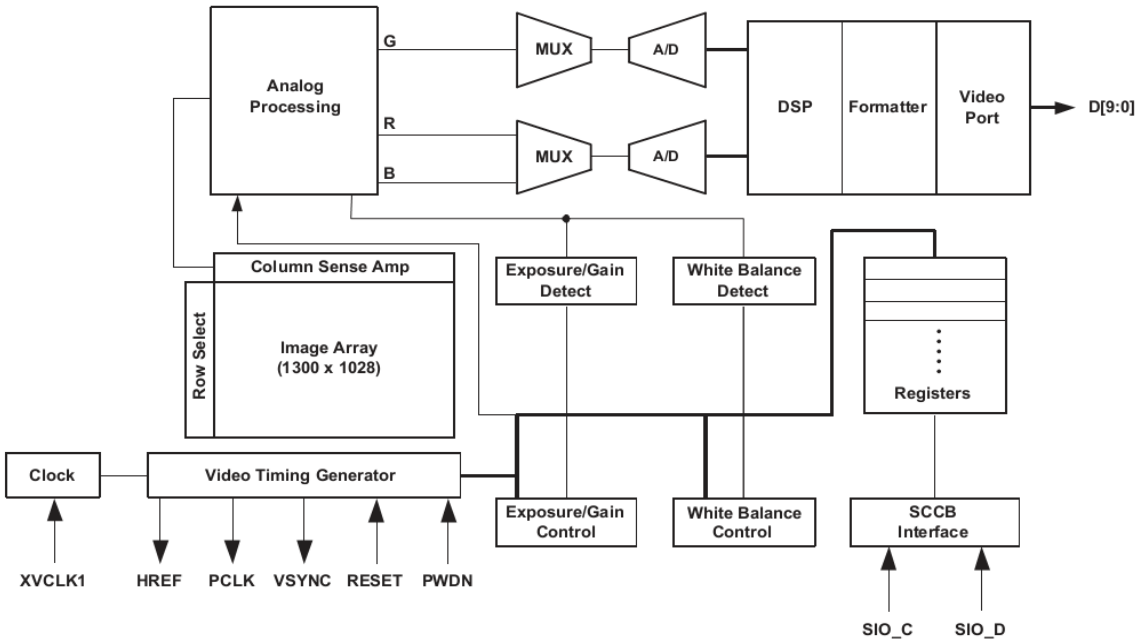
❖ *Mô tả chức năng*

Hình 2.4 cho thấy sơ đồ khối chức năng của các bộ cảm biến hình ảnh OV9650 bao gồm [20]:

- Bộ cảm biến hình ảnh Array (1300 x 1028 mảng hình ảnh hoạt động)
- Bộ xử lý tín hiệu Analog
- Bộ chuyển đổi A/D
- Xử lý tín hiệu số (DSP)
- Định dạng đầu ra
- Phát thời gian

- Giao diện SCCB
- Cổng Digital Video

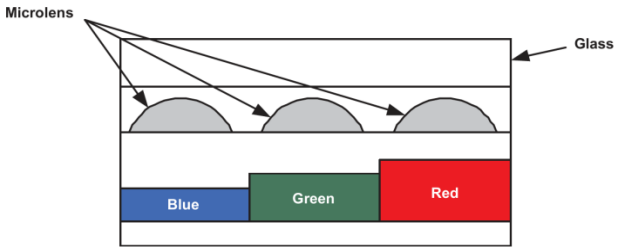
❖ *Sơ đồ khối chức năng*



Hình 2.4: Sơ đồ khối chức năng OV9650

(1) Bộ cảm biến hình ảnh mảng

Các cảm biến OV9650 có một mảng hình ảnh hoạt động 1300 cột 1028 hàng (1.336.400 điểm ảnh), Hình 2.5 cho thấy mặt cắt ngang của mảng cảm biến hình ảnh.



Hình 2.5: Mảng cảm biến hình ảnh

(2) Khối phát tín hiệu thời gian

Nói chung các khối phát tín hiệu thời gian kiểm soát các chức năng sau đây:

- Mảng kiểm soát và tạo khung (có thể định dạng 7 kết quả đầu ra khác nhau)
- Thời gian tín hiệu nội bộ và phân phối
- Tỷ lệ thời gian khung hình

- Tự động điều chỉnh độ sáng (AEC)
- Thời gian kết quả đầu ra (Vsync, HREF / HSYNC, và PCLK)

(3) Xử lý tín hiệu Analog

Khối này thực hiện tất cả các chức năng hình ảnh tương tự bao gồm:

- Tự động điều khiển hệ số khuếch đại (AGC)
- Cân bằng trắng tự động (AWB)

(4) Chuyển đổi A/D

Sau khi xử lý Analog, các mô hình bayer tín hiệu tương tự sang số (A/D) Raw 10-bit chuyển đổi thông qua hai ghép kênh, một cho kênh G và 1 chung bởi kênh B và R. Những chuyển đổi A/D hoạt động ở tốc độ lên đến 12MHz và hoàn toàn đồng bộ tỷ lệ điểm ảnh (tỷ lệ chuyển đổi thực tế có liên quan đến tỷ lệ khung hình)

Ngoài việc chuyển đổi A/D, khối này cũng có các chức năng:

- Hiệu chỉnh mức tối (BLC)
- Tùy chọn kênh trễ U/V
- Bổ xung phạm vi điều khiển A/D

Nói chung, sự kết hợp của phạm vi ghép kênh A/D và dải thiết lập điều khiển A/D và giá trị tối đa A/D để cho phép người dùng điều chỉnh độ sáng hình ảnh cuối cũng như là một chức năng của các ứng dụng cá nhân.

(5) Xử lý tín hiệu số (DSP)

Khối này điều khiển nội suy từ dữ liệu Raw RGB và điều khiển chất lượng hình ảnh:

- Cạnh tăng cường (lọc thông cao hai chiều).
- Chuyển đổi không gian màu (có thể thay đổi dữ liệu Raw RGB hoặc YUV / YCbCr).
- Ma trận RGB để loại bỏ sự trao đổi qua màu sắc.
- Điều khiển Hue và sự bão hòa.
- Lập trình điều khiển gamma.
- Chuyển dữ liệu 10-bit sang bit-8.
- Loại bỏ điểm ảnh trắng.

- De-noise.

(6) Định dạng đầu ra

Khối này kiểm soát tất cả các đầu ra và định dạng dữ liệu cần thiết trước khi gửi dữ liệu hình ảnh cho đầu ra.

(7) Cổng Video kỹ thuật số

Đăng ký bit COM2 [1:0] tăng I_{OL}/I_{OH} hiện thời

(8) Giao diện SCCB

Giao diện máy ảnh điều khiển nối tiếp Bus (SCCB) điều khiển hoạt động chip camera. Đặc điểm kỹ thuật điều khiển nối tiếp Camera (SCCB) Bus cho việc sử dụng chi tiết cổng điều khiển nối tiếp.

Bảng 2.3: Mô tả Pin

Pin Location	Name	Pin Type	Function/Description
A1	PWDN	Function (default = 0)	Power Down Mode Selection - active high, internal pull-down resistor. 0: Normal mode 1: Power down mode
A2	AVDD	Power	Analog power supply (VDD-A= 2.45 to 2.8 VDC)
A3	SIO_D	I/O	SCCB serial interface data I/O
A4	D2	Output	Output bit[2] - LSB for 8-bit YUV or RGB565/RGB555
A5	D4	Output	Output bit[4]
B1	VREF	VREF	Internal voltage reference - connect to ground through 1µF capacitor
B2	NVDD	VREF	Voltage reference
B3	AGND	Power	Analog ground
B4	SIO_C	Input	SCCB serial interface clock input
B5	D3	Output	Output bit[3]
C1	D0	Output	Output bit[0] - LSB for 10-bit Raw RGB data only
C2	DVDD	Power	Power supply (VDD-C= 1.8 VDC +10%) for digital core logic
C4	NC	—	No connection

C5	D5	Output	Output bit[5]
D1	D1	Output	Output bit[1] - for 10-bit RGB only
D2	VSYNC	Output	Vertical sync output
D4	NC	—	No connection
D5	NC	—	No connection
E1	HREF	Output	HREF output
E2	DOVDD	Power	Digital power supply (VDD-IO= 2.5 to 3.3 VDC) for I/O
E3	RESET	Function (default = 0)	Clears all registers and resets them to their default values. Active high, internal, pull-down resistor.
E4	D8	Output	Output bit[8]
E5	D6	Output	Output bit[6]
F1	PCLK	Output	Pixel clock output
F2	XVCLK1	Input	System clock input
F3	DOGND	Power	Digital ground
F4	D9	Output	Output bit[9] - MSB for 10-bit Raw RGB data and 8-bit YUV or RGB565/RGB555
F5	D7	Output	Output bit[7]

Chú ý:

D [9: 2] cho 8-bit YUV hoặc RGB565/RGB555 (D [9] MSB, D [2] LSB)

D [9:0] cho 10-bit dữ liệu Raw RGB (D [9] MSB, D [0] LSB)

Bảng 2.4: Giá trị cực đại OV9650

Ambient Storage Temperature		-40ºC to +95ºC
Supply Voltages (with respect to Ground)	VDD-A	4.5 V
	VDD-C	3 V
	VDD-IO	4.5 V
All Input/Output Voltages (with respect to Ground)		-0.3V to V_{DD-IO} ± 1V
Lead-free Temperature, Surface-mount process		245ºC
ESD Rating, Human Body model		2000V

Chú ý:

Giá trị cực đại trên có thể làm mất hiệu lực tất cả các thông số kỹ thuật điện AC và DC và có thể dẫn đến thiệt hại thiết bị vĩnh viễn

Bảng 2.5: Đặc điểm DC (-20 ° C < T_A < 70 ° C)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{DD-A}	DC supply voltage – Analog	–	2.45	2.5	2.8	V
V_{DD-C}	DC supply voltage – Core	–	1.62	1.8	1.98	V
V_{DD-IO}	DC supply voltage – I/O power	–	2.25	–	3.6	V
I_{DDA}	Active (Operating) Current	See Note[a]		20		mA
$I_{DDS-SCCB}$	Standby Current	See Note[b]		1		mA
$I_{DDS-PWDN}$	Standby Current			10		µA
V_{IH}	Input voltage HIGH	CMOS	0.7 xV_{DD-IO}			V
V_{IL}	Input voltage LOW				0.3 xV_{DD-IO}	V
V_{OH}	Output voltage HIGH	CMOS	0.9 xV_{DD-IO}			V
V_{OL}	Output voltage LOW				0.1 xV_{DD-IO}	V
I_{OH}	Output current HIGH	See Note[c]		8		mA
I_{OL}	Output current LOW		15	8		mA
I_L	Input/Output Leakage	GND to V_{DD-IO}			± 1	µA

a. V_{DD-A}= 2.5V, V_{DD-C}= 1.8V, V_{DD-IO}= 3.0V

$I_{DDA} = \sum \{I_{DD-IO} + I_{DD-C} + I_{DD-A}\}$, fCLK= 24MHz at 7.5 fps YUV output, no I/O loading

b. V_{DD-A} = 2.5V, V_{DD-C} = 1.8V, V_{DD-IO} = 3.0V

$I_{DDS:SCCB}$ refers to a SCCB-initiated Standby, while $I_{DDS:PWDN}$ refers to a PWDN pin-initiated Standby

c. Standard Output Loading = 25pF, 1.2KΩ

Bảng 2.6: Đặc điểm và chức năng AC (-20 ° C < T_A < 70 ° C)

Symbol	Parameter	Min	Typ	Max	Unit
Functional Characteristics					
	A/D Differential Non-Linearity		+1/2		LSB
	A/D Integral Non-Linearity		±1		LSB
	AGC Range		18		dB
	Red/Blue Adjustment Range		12		dB
Inputs (PWDN, CLK, RESET)					
f_{CLK}	Input Clock Frequency	10	24	48	MHz
t_{CLK}	Input Clock Period	21	42	100	ns
$t_{CLK:DC}$	Clock Duty Cycle	45	50	55	%
$t_{S:RESET}$	Setting time after software/hardware reset		1		ms
$t_{S:REG}$	Settling time for register change (10 frames required)		300		ms
SCCB Timing (see Figure 4)					
f_{SIO_C}	Clock Frequency		400		KHz
t_{LOW}	Clock Low Period		1.3		µs
t_{HIGH}	Clock High Period		600		ns
t_{AA}	SIO_C low to Data Out valid	100		900	ns
t_{BUF}	Bus free time before new START	1.3			µs
$t_{HD:STA}$	START condition Hold time	600			ns
$t_{SU:STA}$	START condition Setup time	600			ns
$t_{HD:DAT}$	Data-in Hold time	0			µs
$t_{SU:DAT}$	Data-in Setup time	100			ns
$t_{SU:STO}$	STOP condition Setup time	600			ns
t_R, t_F	SCCB Rise/Fall times			300	ns

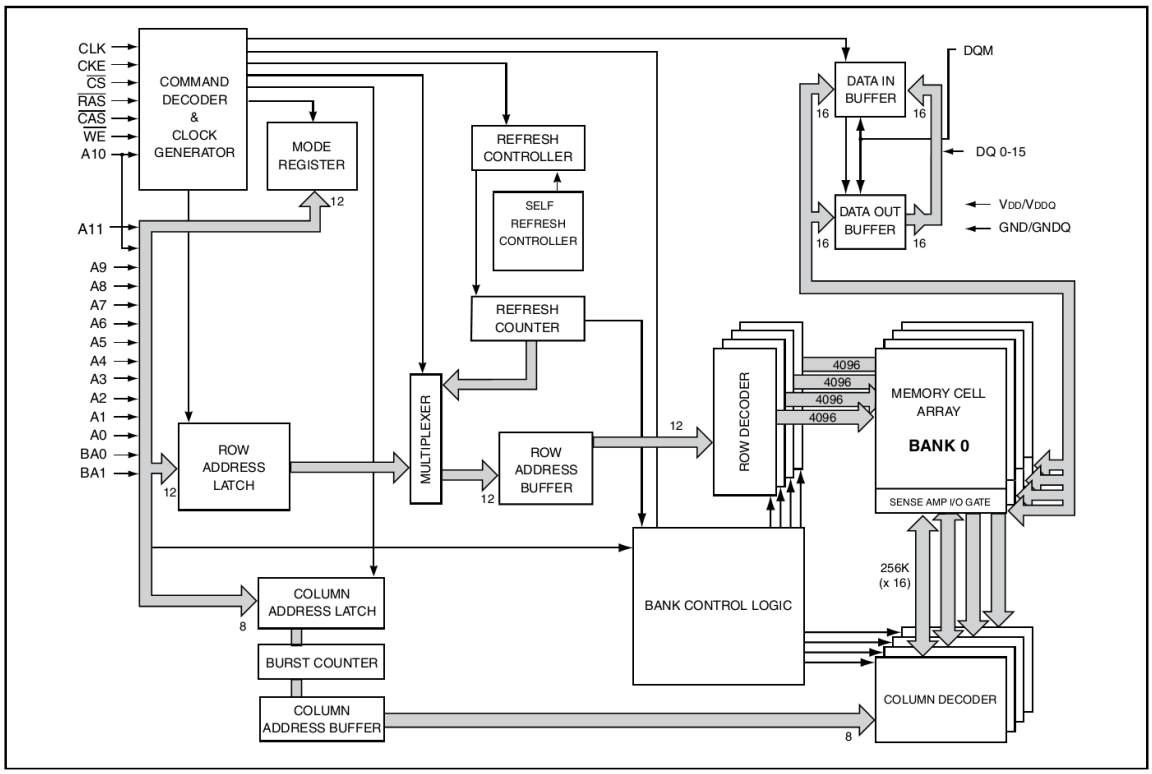
t_{DH}	Data-out Hold time	50			ns
Outputs (VSYNC, HREF, PCLK, and D[9:0] (see Figure 5, Figure 6, Figure 7, Figure 8, Figure 10, and Figure 11)					
t_{PDV}	PCLK[↓] to Data-out Valid			5	ns
t_{SU}	D[9:0] Setup time	15			ns
t_{HD}	D[9:0] Hold time	8			ns
t_{PHH}	PCLK[↓] to HREF[↑]	0		5	ns
t_{PHL}	PCLK[↓] to HREF[↓]	0		5	ns
AC Conditions:	• V_{DD}: V_{DD-C} = 1.8V, V_{DD-A} = 2.5V, V_{DD-IO} = 3.0V • Rise/Fall Times: I/O: 5ns, Maximum SCCB: 300ns, Maximum • Input Capacitance: 10pf • Output Loading: 25pF, 1.2KΩ to 3V • fCLK: 24MHz				

2.4 Lựa chọn khối bộ nhớ hệ thống

Trong quá trình thiết kế hệ thống thu thập hình ảnh, để lưu lại dữ liệu cho hình ảnh ta cần cấu hình thêm bộ nhớ bên ngoài. Để bộ nhớ bên ngoài có thể lưu trữ toàn bộ dữ liệu một hình ảnh thu được từ cảm biến OV9650 với khung hình SXGA (1280x1024) cần có một dung lượng bộ nhớ 1,3MB. Trong trường hợp này sử dụng bộ nhớ SDRAM (Synehronous Dynamie Random Aeeess Memory) dung lượng bộ nhớ 8-Mbyte trên Kit DE1 với chip SDRAM IS42S16400 [2].

SDRAM IS42S16400 có 67180864 bit SDRAM được tổ chức thành 4 dải (BANK) nhớ, mỗi dải có dung lượng 1024576 Words với tốc độ truyền dữ liệu có thể lên đến 133MHz.

SDRAM Thực hiện việc truyền dữ liệu qua các chân địa chỉ và dữ liệu dưới sự chi phối của các chân điều khiển:



Hình 2.6: Sơ đồ khối SDRAM

- CKE cho phép xung clock. Khi tín hiệu này ở mức thấp, chip xử lý giống như là xung clock hoàn toàn bị dừng lại.
- /CS lựa chọn chip: ở mức cao thì bỏ qua tất cả các đầu vào khác (ngoại trừ CKE), và hoạt động như một lệnh NOP.
- DQM mặt nạ dữ liệu: Khi ở mức cao, những tín hiệu này khống chế dữ liệu vào/ra. Khi đi kèm với việc ghi, dữ liệu không thật được ghi. Khi dữ liệu được giữ ở mức trong hai chu kỳ trước một chu kỳ đọc, việc đọc không được đưa ra từ chip. Trên một chip nhớ x16 hay DIMM, với 1 word 8 bit thì có một hàng DQM.
- /RAS Row Address Strobe là bit điều khiển cho qua địa chỉ hàng.
- /CAS Column Address Strobe bit điều khiển cho qua địa chỉ cột.
- /WE Write enable cho phép ghi.

Các tín hiệu /RAS, /CAS, /WE dùng để lựa chọn 1 trong 8 lệnh. Nói chung thì dùng để phân biệt các lệnh đọc, ghi.

SDRAM bên trong được chia thành 2 hay 4 dải (Bank) dữ liệu nội độc lập bên trong. Một hoặc hai địa chỉ vào của dải (Bank) BA0 và BA1 sẽ lựa chọn Bank mà lệnh tác động đến.

Phần lớn các lệnh đều sử dụng địa chỉ được đưa vào ngõ vào địa chỉ. Nhưng có một số lệnh lại không sử dụng chúng, hay chỉ biểu diễn một địa chỉ cột, vì vậy ta sử dụng A[10] để lựa chọn những phương án.

Bảng 2.7: Các chế độ truy cập SDRAM

/CS	/RAS	/CAS	/WE	BA	A10	A11 A9-A0	Lệnh
H	X	X	X	X	X	X	Hủy bỏ các lệnh
L	H	H	H	X	X	X	Không làm gì cả (NOP)
L	H	H	L	X	X	X	Dừng (hủy) truyền khối: dừng lệnh đọc khối hay ghi khối khi đang thực hiện.
L	H	L	H	Bank	L	Column	Read: đọc khối dữ liệu từ hàng kích hoạt hiện hành.
L	H	L	H	Bank	H	Column	Đọc với Precharge (nạp lại) tự động: khi thực hiện xong thì Precharge (tức là đóng hàng lại).
L	H	L	L	Bank	L	Column	Write: ghi khối dữ liệu từ hàng kích hoạt hiện hành.
L	H	L	L	Bank	H	Column	Ghi với sự nạp lại tự động: khi thực hiện xong thì nạp lại (Precharge) tức là đóng hàng lại.
L	L	H	H	Bank	Row		Active (kích hoạt): mở một hàng với lệnh Read và Write.
L	L	H	L	Bank	L	X	Precharge (nạp lại): Ngưng hoạt động hàng hiện hành của bank (dải) được chọn.
L	L	H	L	X	H	X	Precharge all (nạp lại toàn bộ): Ngưng hoạt động hàng hiện hành

							của tất cả các bank (dải).
L	L	L	H	X	X	X	Auto refresh (tự động làm tươi): làm tươi từng hàng của từng bank, sử dụng bộ đếm nội. Tất cả các dải phải được nạp lại.
L	L	L	L	X	Mode		Load mode register (chế độ nạp các thanh ghi): A[9:0] được nạp để cấu hình chip DRAM. Trong đó quan trọng nhất là ngầm định CAS (2 hoặc 3 chu kỳ) và chiều dài khối (1, 2, 4, hoặc 8 chu kỳ).

❖ *Sự tương tác các tín hiệu điều khiển SDRAM:*

Lệnh chế độ nạp các thanh ghi (load mode register command) yêu cầu tất cả các dải (Bank) ở trạng thái IDE, và phải trì hoãn về sau cho sự thay đổi để tác động.

Lệnh tự động làm tươi (auto refresh command) thì yêu cầu tất cả các dải (Bank) ở trạng thái IDE, và mất 1 khoảng thời gian làm tươi để đưa Chip về trạng thái IDE: thường là $t_{rcd} + t_{rp}$.

Chỉ có những lệnh khác thì cho phép trên một Bank IDE là các lệnh kích hoạt. Cần phải mất t_{rcd} trước khi hàng được mở hoàn toàn và chấp nhận một lệnh đọc hay ghi.

Khi một dải (Bank) được mở thì có 4 lệnh được cho phép: đọc, ghi, kết thúc truyền khối (Burst terminal), nạp lại (precharge). Lệnh đọc, ghi bắt đầu truyền khối và có thể bị ngắt bởi những ngắt sau:

❖ *Ngắt đọc khối dữ liệu:*

Sau một lệnh đọc thì bất cứ lúc nào cũng có thể có một trong các lệnh: đọc, kết thúc truyền khối, hoặc là nạp được phát ra. Và sẽ ngắt đọc khối này nếu có một ngầm định CAS được cấu hình. Nếu có 1 lệnh đọc ở thời điểm 0, 1 lệnh đọc khác ở chu kỳ 2, ngầm định CAS ở chu kỳ 3 thì lệnh đọc đầu tiên sẽ truyền khối dữ liệu ra ngoài ở chu kỳ 3 và 4, và kết quả của lệnh đọc thứ 2 sẽ bắt đầu xuất hiện ở chu kỳ 5.

Nếu lệnh ở chu kỳ 2 là kết thúc truyền khối hoặc là nạp lại Bank kích hoạt thì không có dữ liệu ra ở chu kỳ 5.

Mặc dù việc ngắt lệnh đọc có thể xuất hiện ở một Bank bất kỳ , nhưng lệnh nạp lại chỉ ngắt việc đọc khối nếu nó tác động trên cùng một Bank hoặc tất cả các Bank, nếu lệnh này hướng đến một Bank khác thì việc đọc khối vẫn tiếp tục.

Sự ngắt đọc tạo ra bởi một lệnh ghi thì cũng có thể nhưng sẽ khó khăn hơn. Thực hiên điều này nhờ vào một tín hiệu DQM để khống chế ngõ ra của SDRAM, vì vậy trong khoảng thời gian này,chíp điều khiển bộ nhớ có thể lái dữ liệu đi qua chân DQ để ghi vào SDRAM. Vì tác động của DQM trên lệnh đọc thì bị trì hoãn 2 chu kỳ trong khi đối với lệnh đọc thì ngay lập tức, nên DQM phải lên mức cao (raised) sớm hơn 2 chu kỳ trước khi có lệnh ghi.

Để thực hiện điều này trong 2 chu kỳ thì yêu cầu định vị thời điểm SDRAM tắt ngõ ra tại 1 cạnh lên xung Clock và thời điểm dữ liệu được cung cấp (cho lệnh ghi) như ngõ vào của SDRAM ở cạnh tiếp theo của Clock.

❖ *Một ngắt ghi khối dữ liệu:*

Bất kỳ lệnh đọc, ghi, hay kết thúc truyền tới một Bank bất kỳ sẽ kết thúc (dừng) việc ghi khối ngay lập tức, dữ liệu trên chân DQ khi lệnh thứ 2 được phát thì chỉ do lệnh này sử dụng.

Ngắt ghi khối với lệnh precharge (đến cùng một Bank) thì khá phức tạp. Đó là thời gian viết nhỏ nhất, t_{wr} phải được lướt qua giữa tác vụ ghi sau cùng tới 1 Bank (chu kỳ không bị che (unmasked) cuối cùng của ghi khối) với lệnh precharge kế tiếp, vì vậy một ghi khối sẽ bị dừng (hủy) bởi lệnh tích nạp (pre-charge) nếu có đủ chu kỳ kéo dài được che đi (dùng DQM) để tạo t_{wr} cần thiết. Một lệnh ghi với sự tích nạp tự động chứa đựng một trì hoãn tự động.

❖ *Ngắt một lệnh tích nạp tự động:*

Việc xử lý sự gián đoạn của thao tác đọc, ghi với chế độ tích nạp tự động là một đặc tính lựa chọn của SDRAM, và được hỗ trợ rất nhiều. Nếu được sử dụng, sự tích nạp hay thời gian chờ t_{wr} theo sau bởi sự tích nạp (sau khi đọc) bắt đầu cùng một chu kỳ như một lệnh ngắt.

❖ *Sắp xếp truyền khối SDRAM:*

Một bộ vi xử lý hiện đại có bộ đệm nói chung sẽ truy nhập bộ nhớ trong những đơn vị của line bộ đệm. Ví dụ để truyền 64byte, line bộ đệm yêu cầu 8 sự truy cập liên tiếp tới một DIMM(dual in-line memory module: module nhớ có hai hàng chân) 64bit, mà toàn bộ có thể được kích hoạt bởi một lệnh đơn đọc hay ghi tùy vào sự cấu hình các chíp SDRAM.

Sự truy cập line đệm điển hình được kích hoạt bởi một sự đọc từ một địa chỉ đặc biệt, và SDRAM cho phép " từ tính chất quyết định " của line đệm sẽ được truyền đầu tiên. ("từ" ở đây có nghĩa là chiều rộng chíp SDRAM hay DIMM, 64 bít với một DIMM tiêu biểu).

Chíp SDRAM hỗ trợ hai giao thức để sắp xếp các từ còn lại trong line đệm:

Chế độ truyền khối đan xen: làm cho các tính toán thêm phức tạp nhưng lại dễ dàng tổng hợp phần cứng hơn và được ưu tiên với các bộ vi xử lý Intel. Ta không sử dụng kiểu truyền này.

Chế độ truyền khối tuần tự: những từ trễ hơn được truy cập trong việc tăng dần địa chỉ, khi kết thúc thì quay trở lại điểm bắt đầu khối. Chẳng hạn, với một tuyền khối có chiều dài là 4, và địa chỉ cột được yêu cầu là 5, những từ sẽ truy cập theo thứ tự 5 – 6 – 7 – 4. Nếu chiều dài truyền khối là 8, thứ tự truy cập là 5 – 6 – 7 – 0 – 1 – 2 – 3 – 4. Điều này được thực hiện bởi việc thêm một bộ đếm địa chỉ cột, và bỏ qua số nhớ khi đi hết khối.

Ta có thể lựa chọn chiều dài khối và kiểu truy cập khối bằng cách sử dụng chế độ thanh ghi được mô tả phần tiếp theo.

❖ *Chế độ thanh ghi của SDRAM:*

Tốc độ dữ liệu đơn SDRAM có một chế độ thanh ghi 10 bít đơn lập trình được. Sau đó chuẩn SDRAM tốc độ dữ liệu kép SDRAM bổ sung thêm chế độ thanh ghi, định địa chỉ sử dụng những chân địa chỉ Bank. Với SDR SDRAM, chân địa chỉ Bank và địa chỉ hàng A[10] và cao hơn thì được lờ đi, nhưng phải là 0 trong khi ở chế độ ghi vào thanh ghi. Trong chu kỳ của chế độ thanh ghi thì các giá trị nạp vào M[9:0] chính là các bit địa chỉ.

M[9] chế độ ghi từng khối, ở mức 0 thì ghi sử dụng chế độ và chiều dài truyền khối ở chế độ đọc, ở mức 1 thì tất cả các ghi không phải là truyền khối(định vị đơn).

M[8:7] chế độ vận hành, muốn ở chế độ lưu trữ thì đặt giá trị 00.

M[6:4] ngầm định CAS chỉ với các giá trị hợp lệ là 010 (CL2) và 011 (CL3). Chỉ ra số chu kỳ giữa lệnh đọc và dữ liệu được gửi ra từ Chip. Chip sẽ hoàn thành một giới hạn cơ bản trong nanô-giây dựa trên giá trị này; khi khởi tạo, bộ điều khiển bộ nhớ phải sử dụng kiến thức của nó về tần số xung Clock và dịch giới hạn kia thành những chu trình.

M[3] kiểu truy cập các từ trong khối: 0 thì truy cập tuần tự, 1 thì truy cập đan xen.

M[2:0]: chiều dài khối: giá trị 000, 001, 010 và 011 chỉ ra kích thước khối tương ứng là 1, 2, 4 hay 8 từ. Mỗi đọc (và viết, nếu m [9] là 0) sẽ thực hiện nhiều sự truy cập, trừ phi được gián đoạn bởi một sự dừng (hủy) truyền khối hay các lệnh khác. Giá trị 111 đặc tả khối với đầy đủ hàng (full-row Burst hoặc còn gọi là full page Burst). Sự truyền khối với đầy đủ hàng chỉ được cho phép với kiểu tuần tự. Đối với SDRAM IS42S16400 thì chiều dài của 1 khối ở chế độ full page Burst là 256 từ. Sự truyền khối tiếp tục cho đến khi có ngắt.

❖ *Làm tươi tự động:*

Dùng để làm tươi lại Chip ram nhờ vào sự mở và đóng (kích hoạt và tích nạp) từng hàng trong từng Bank. Tuy nhiên, để đơn giản hóa chíp điều khiển bộ nhớ, Chip SDRAM hỗ trợ lệnh tự động làm tươi, tức là đồng thời thực hiện thao tác này tới một hàng trong từng Bank. SDRAM cũng duy trì một bộ đếm nội được lặp lại trên toàn bộ các hàng có thể. Chip điều khiển bộ nhớ thì đơn giản phải phát ra đủ số lượng các lệnh làm tươi tự động (1 lệnh đối với 1 hàng) với mỗi khoảng làm tươi (một giá trị chung là t_{ref} = 64 ms). Tất cả các Bank phải ở trạng thái IDE khi lệnh được phát.

❖ *Chế độ Lover Power:*

Như đã đề cập, ngõ vào cho phép xung Clock (CKE) có thể được dùng để dừng xung Clock tới SDRAM. Giá trị ngõ vào CKE được xét tại từng cạnh lên của xung

Clock, và nếu ở mức thấp, thì mọi cạnh lên của xung Clock tiếp theo sẽ bị bỏ qua mọi mục đích khác so với việc kiểm tra CKE.

Nếu CKE xuống thấp trong khi SDRAM đang thực hiện tác vụ, thì nó đơn giản chỉ là "đóng băng lại" tại chỗ cho đến khi CKE lên mức cao.

Nếu SDRAM ở trạng thái IDE (tất cả các Bank được tích nạp, không có lệnh nào đang hoạt động) khi CKE xuống thấp, SDRAM tự động chọn chế độ power-down (tiết kiệm năng lượng), giữ năng lượng ở cực tiểu cho tới khi có cạnh lên của CKE. Khoảng này thì không được dài hơn giá trị tối đa khoảng làm tươi t_{ref}, nếu không những gì bộ nhớ chứa đựng sẽ bị mất. Đây là phương pháp để dừng toàn bộ xung Clock trong khoảng thời gian này để tiết kiệm năng lượng.

Cuối cùng, nếu CKE ở mức thấp vào lúc một lệnh làm tươi tự động được gửi đến SDRAM, SDRAM chọn chế độ tự làm tươi (seft-refresh mode). Tương tự Power Down, nhưng SDRAM dùng một timer nội để phát ra các chu kỳ làm tươi nội khi cần thiết. Trong thời gian này thì dừng xung Clock. Chế độ tự làm tươi tiêu thụ ít năng lượng hơn so với chế độ Power Down, nhưng vẫn cho phép bộ điều khiển bộ nhớ disnable toàn bộ.

2.5 Lựa chọn giao diện hiển thị ảnh

Sau khi hình ảnh thu được và xử lý bởi FPGA, để hiển thị các tín hiệu này ra màn hình, các tín hiệu số phải được chuyển đổi thành tín hiệu hình ảnh tương tự. dữ liệu hình ảnh đầu ra FPGA được chuyển thành 3 màu R, G, B và được hiển thị theo chuẩn VGA. Cổng VGA là kiểu giao diện đồ họa được sử dụng rộng rãi, hầu hết các máy tính với một màn hình ngoài thông qua các thiết bị VGA [3].

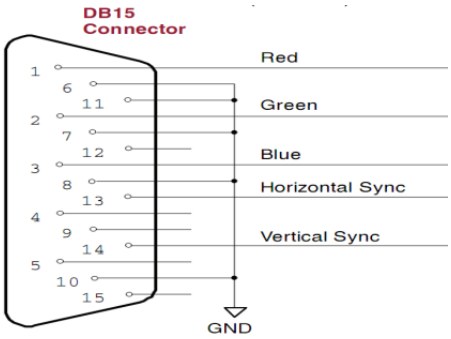
Hình ảnh được hiển thị theo khung hình VGA có độ phân giải 640x480 với 8 màu cơ bản của màn hình CRT (cathode ray tube).

Bảng 2.8: 8 màu cơ bản từ 3 bit của VGA :

Đỏ (R)	Xanh lục (G)	Xanh dương (B)	Màu
0	0	0	Đen
0	0	1	Xanh dương
0	1	0	Xanh
0	1	1	Lục lam
1	0	0	Đỏ

1	0	1	Đỏ tươi
1	1	0	Vàng
1	1	1	Trắng

3 tín hiệu màu là: đỏ (red), xanh lục (green) và xanh dương (blue) gửi tín hiệu màu sắc (color information) đến màn hình VGA. Mỗi một tín hiệu điều khiển một súng bắn điện tử (electron gun) để phóng các hạt electron vẽ lên một màu cơ bản tại một điểm trên màn hình. Dải của tín hiệu nằm từ từ 0V (tương ứng với màu tối hoàn toàn) và 0.7V (sáng hoàn toàn) điều khiển cường độ của mỗi thành phần màu và 3 thành phần màu kết hợp với nhau tạo lên màu của điểm ảnh (dot) hay phần tử ảnh (pixel) trên màn hình.



Hình 2.7: Sơ đồ kết nối VGA

Cổng VGA có 5 tín hiệu bao gồm các tín hiệu đồng bộ theo phương ngang và phương dọc , h_sync và v _sync và 3 tín hiệu hình ảnh cho 3 màu đỏ , xanh, xanh dương.

Hình ảnh là một tín hiệu tương tự , và bộ điều khiển video sử dụng một bộ chuyển đổi DAC để chuyển đổi tín hiệ u số đầu ra thành mức tương tự mong muốn . Nếu một tín hiệu hình ảnh N -bit thì tín hiệu này có thể được chuyển thành 2^N mức tương tự.

Trong phần thảo luận , chúng ta dùng tín hiệu hình ảnh màu 3 – bit nên đầu ra chúng ta sẽ thu được $2^3 = 8$ màu cơ bản như được liệt kê trong bảng trên .

Tùy vào độ rộng A bít của tín hiệu màu ngõ vào tín mà mỗi màu analog ở ngõ ra là một trong 2^A mức với bộ chuyển đổi digital to analog A bit, 3 tín hiệu analog kết hợp với nhau tạo nên phần tử ảnh (pixel) với $2^A \times 2^A \times 2^A = 2^{3A}$ màu khác nhau.

Xử lý đồ họa cuối cùng là được sản xuất màn hình, mà là chịu trách nhiệm cho các tín hiệu hình ảnh là đầu ra để hiển thị.

2.6 Tóm tắt chương

Trong chương 2, tác giả đã trình bày về phân tích và lựa chọn các thiết bị cho thiết kế mạch phần cứng cho các khối trong đó có chip FPGA EP2C20F484C7 và cảm biến hình ảnh OV9650, bộ nhớ lưu trữ dữ liệu tạm thời SDRAM IS42S16400 và giao diện hiển thị hình ảnh VGA. Việc thiết kế mạch chi tiết hần cứng các khối được thể trình bày trong chương 3.

Chương 3: THIẾT KẾ HỆ THỐNG THU THẬP HÌNH ẢNH

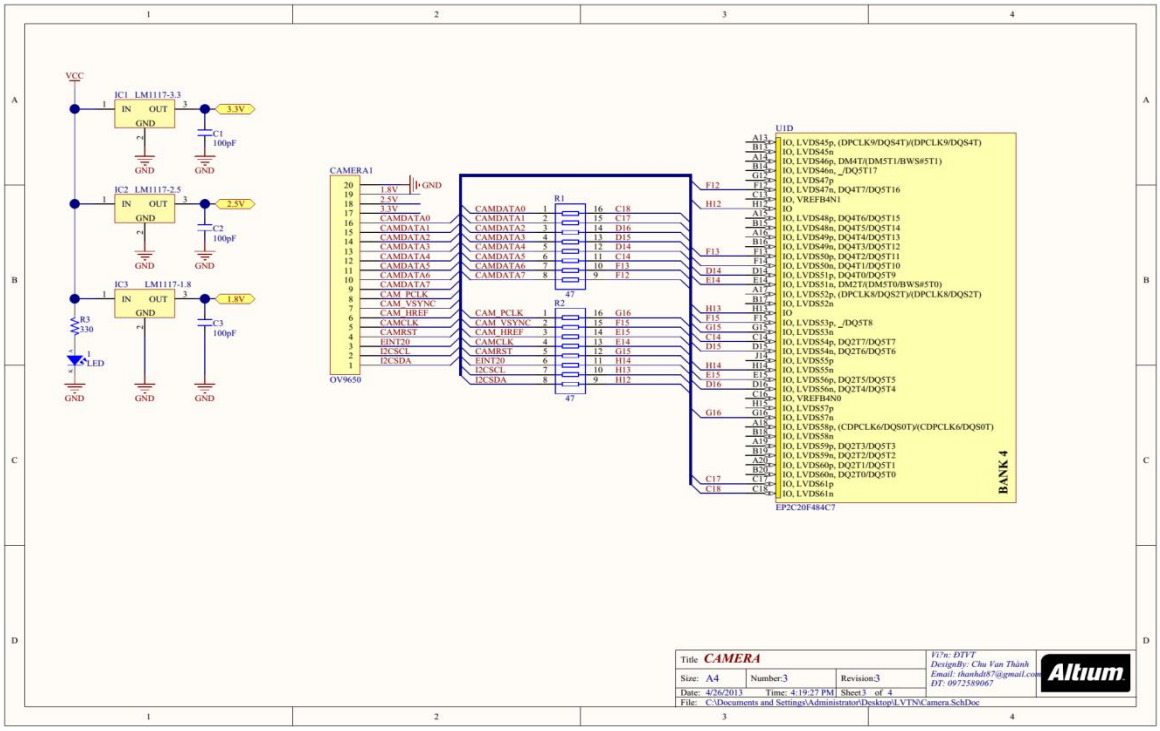
Thiết kế tổng thể hệ thống thu thập và hiển thị hình ảnh, sơ đồ phần cứng các khối sẽ được thảo luận chi tiết trong chương này. Dựa trên sơ đồ khối mạch phần cứng của các thiết kế hiện nay trong chủ yếu bao gồm các mạch thu thập hình ảnh, bộ nhớ SDRAM, VGA (video GraphicsArray) và mạch nguồn cung cấp điện. Trong chương cũng trình bày về thiết kế mã chương trình làm việc từng khối làm việc cho chip FPGA dựa trên nguyên lý, chức năng làm việc của từng khối.

3.1 Thiết kế khối Camera

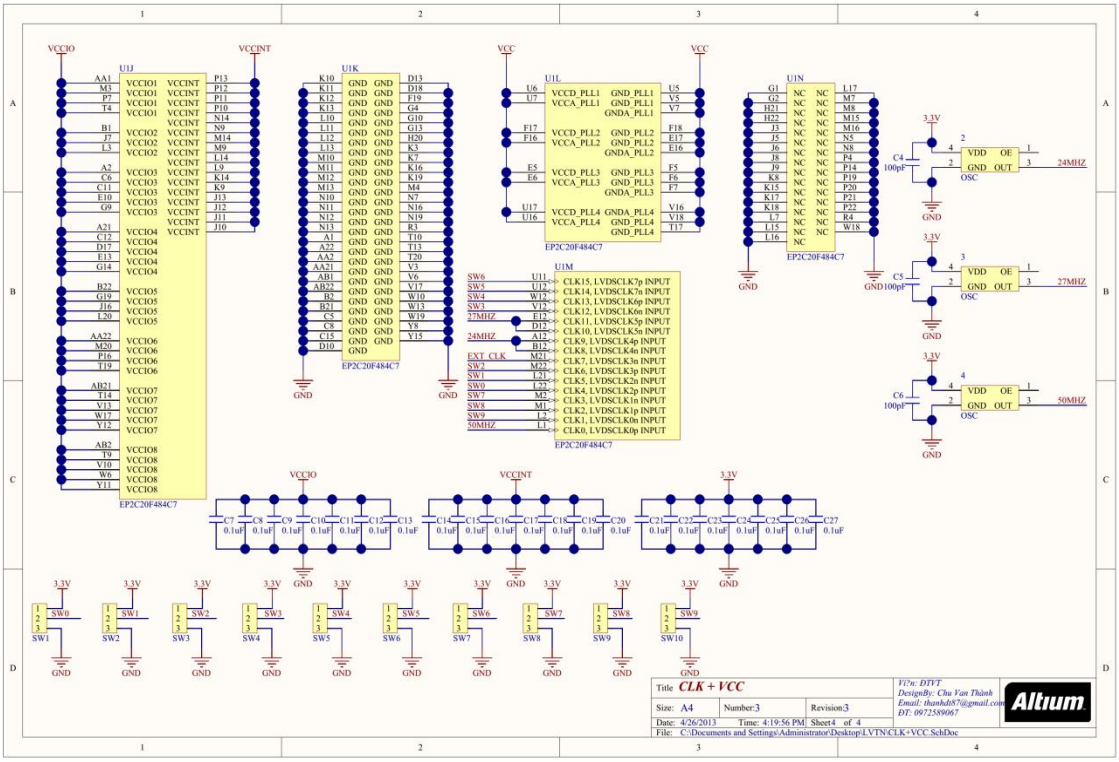
Trong khối Camera được chia ra làm 2 phần thiết kế chính: Thiết kế giao diện mạch phần cứng và thiết kế khối điều khiển trên FPGA.

3.1.1 Thiết kế mạch giao diện Camera

Sơ đồ nguyên lý mạch giao diện camera OV9650 được thể hiện trong hình 3.1. Trong sơ đồ thiết kế này tác giả đã thiết kế giao diện kết nối tín hiệu vào ra giữa FPGA và cảm biến hình ảnh OV9650 và nguồn cấp cho cảm biến. các tín hiệu vào ra của cảm biến OV9650 được thể hiển trên sơ đồ.



Hình 3.1: Sơ đồ nguyên lý khối Camera



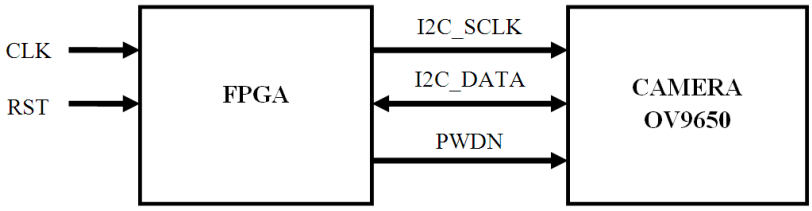
Hình 3.2: Sơ đồ nguyên lý khối cấp nguồn, ngõ vào ra xung và chuyển mạch

Trên hình 3.2 thể hiện thiết kế các chuyển mạch ngõ vào FPGA, tín hiệu xung clock và nguồn cung cấp cho FPGA

3.1.2 Thiết mã chương trình điều khiển camera

3.1.2.1 Thiết kế giao diện SCCB

Việc thực hiện truyền dữ liệu hai dây đòi hỏi phải có một phương pháp điều khiển tổng thể để tạo điều kiện truyền thông SCCB [19].



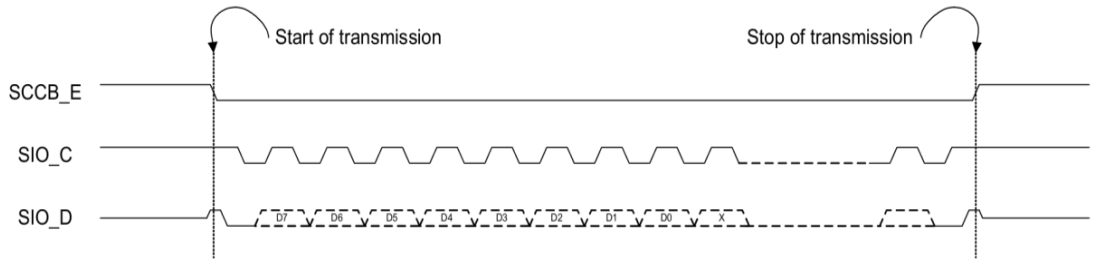
Hình 3.3: Sơ đồ khối điều khiển SCCB

Tên	Mô tả
CLK	Xung Clock 50MHz từ kit DE1
RST	Tín hiệu Reset hệ thống
I2C_SCLK	Ngõ ra xung Clock cho chế độ config

I2C_DATA	Ngõ truyền nhận dữ liệu
PWDN	Tín hiệu điều khiển chế độ hoạt động cho camera

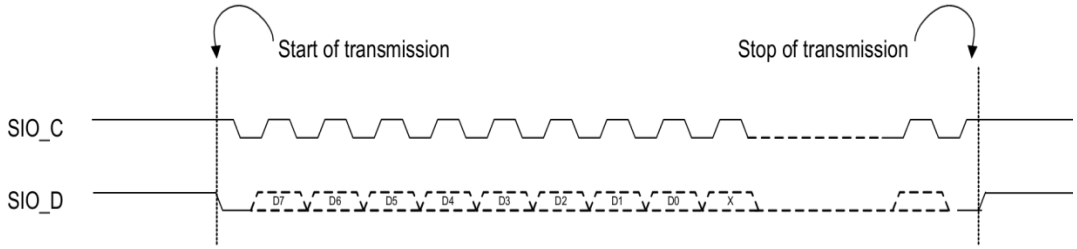
Vai trò của khối là ghi giá trị vào các thanh ghi của OV9650 nên có thể chọn xung clock làm việc của khối là 20KHz nhờ vào bộ chia tần số 50MHz. Địa chỉ Slaver của OV9650 là 60h [21] nên ta sử dụng cách gán dữ liệu cần truyền trên địa chỉ của thanh ghi.

❖ Sơ đồ thời gian truyền dữ liệu 3 dây



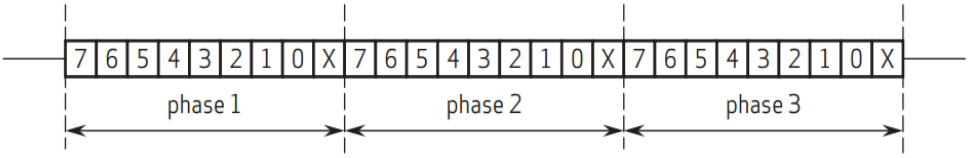
Hình 3.4: Sơ đồ thời gian truyền dữ liệu 3 dây

❖ Sơ đồ thời gian truyền dữ liệu 2 dây



Hình 3.5: Sơ đồ thời gian truyền dữ liệu 2 dây

❖ Các Phases truyền



Hình 3.6: Sơ đồ Phases truyền

Phase 1: ID Address

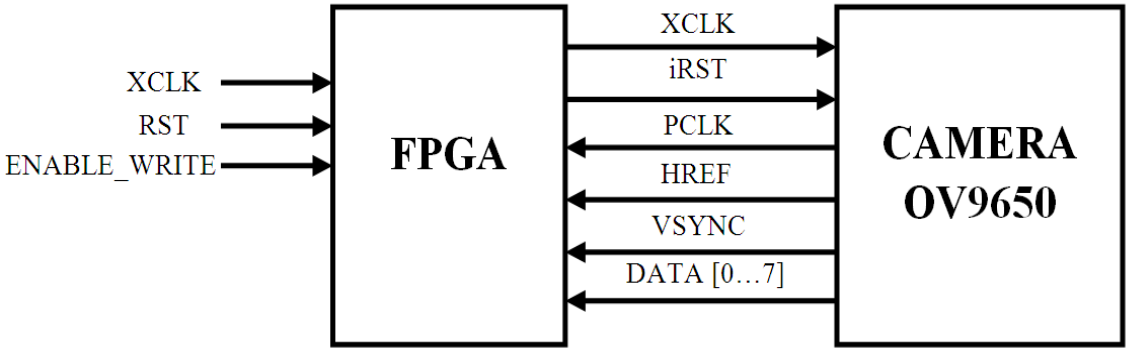
Phase 2: Sub-address / Read data

Phase 3: Write Data

3.1.2.2 Thiết kế giao diện thu thập hình ảnh

Hình ảnh thu được bằng cách sử dụng thiết kế khối cảm biến CMOS của OmniVision một thế hệ mới của chip cảm biến hình ảnh chuyên dụng OV9650. Các thông số hoạt động OV9650, thể hiện trong bảng 2.2 thông số kỹ thuật chính của cảm biến.

Việc kiểm soát một chip OV9650 hoạt động cần một xung đồng hồ 24Mhz cho đầu vào và một tín hiệu thiết lập lại chế độ hoạt động từ vi xử lý FPGA. Trong thời gian đồng bộ hóa tín hiệu PCLK, HREF, VSYNC, FPGA hoàn thành các thiết lập đăng ký, dữ liệu hình ảnh của bộ cảm biến hình ảnh CMOS thông qua giao diện SCCB, dữ liệu hình ảnh được gửi bởi OV9650 ở các đầu ra DATA[7. . 0] tới vi xử lý FPGA và vị trí các điểm ảnh được xác định bởi các tín hiệu PCLK, HREF, VSYNC. OV9650 có một đầu ra dữ liệu trong chế độ truyền song song, thiết kế được kết nối thể hiện như trong hình 3.7:



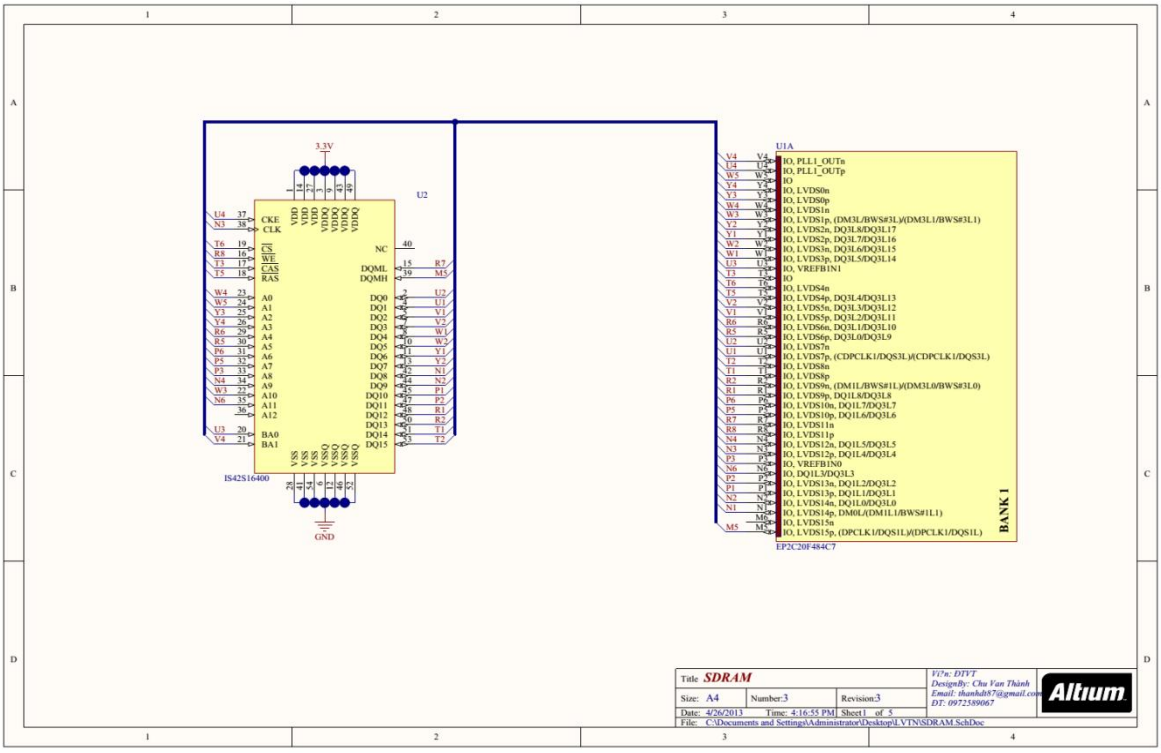
Hình 3.7: Sơ đồ khối thu thập hình ảnh

Tên	Mô tả
CLK	Xung Clock 50MHz từ kit DE1
RST	Tín hiệu Reset hệ thống
XCLK	Xung Clock 24 MHz cấp cho camera
iRST	Tín hiệu Reset camera
HREF	Tín hiệu HREF từ camera
VSYNC	Tín hiệu VSYNC từ camera
DATA [0…7]	Đường dữ liệu điểm ảnh thu được từ camera

3.2 Thiết kế khối bộ nhớ hệ thống

3.2.1 Thiết kế mạch giao diện SDRAM

Thiết kế mạch phần cứng cho khối bộ nhớ hệ thống, khối bộ nhớ được thể hiện ở hình 3.8



Hình 3.8: Sơ đồ nguyên lý khối SDRAM

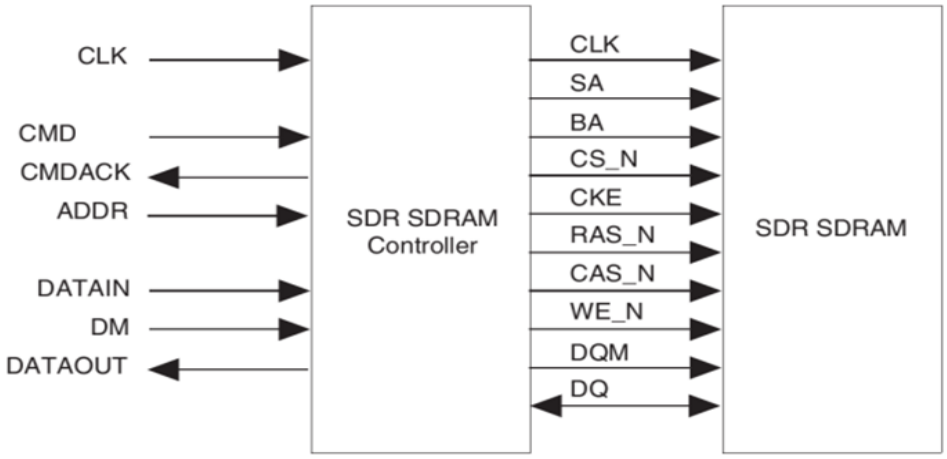
3.2.2 Thiết kế mã chương trình điều khiển SDRAM

Tốc độ dữ liệu của bộ nhớ đồng bộ truy cập ngẫu nhiên (SDRAM) cung cấp một giao diện điều khiển đơn giản làm tiêu chuẩn cho SDR SDRAM. Khối SDR SDRAM Controller được sử dụng ngôn ngữ mô tả Verilog HDL hoặc VHDL và được tối ưu hóa bởi Altera. SDR SDRAM điều khiển hỗ trợ các tính năng sau đây:

- Chiều dài Burst 1, 2, 4, hoặc 8 data words
- Độ trễ CAS 2 hoặc 3 chu kỳ đồng hồ
- 16-bit có thể lập trình được sử dụng để tự động làm mới
- Lựa chọn chip SDRAM
- Hỗ trợ lệnh NOP, READA, WRITEA, AUTO_REFRESH, Precharge, kích hoạt, BURST_STOP, và LOAD_MR

- Hỗ trợ cho hoạt động chế độ toàn trang
- PLL để tăng hiệu suất hệ thống
- Hỗ trợ đường dữ liệu 16, 32, và 64 bit

Hình 3.9 cho thấy một sơ đồ hệ thống cấp điều khiển SDR SDRAM [17].



Hình 3.9: Sơ đồ tổng quan hệ thống điều khiển SDRAM

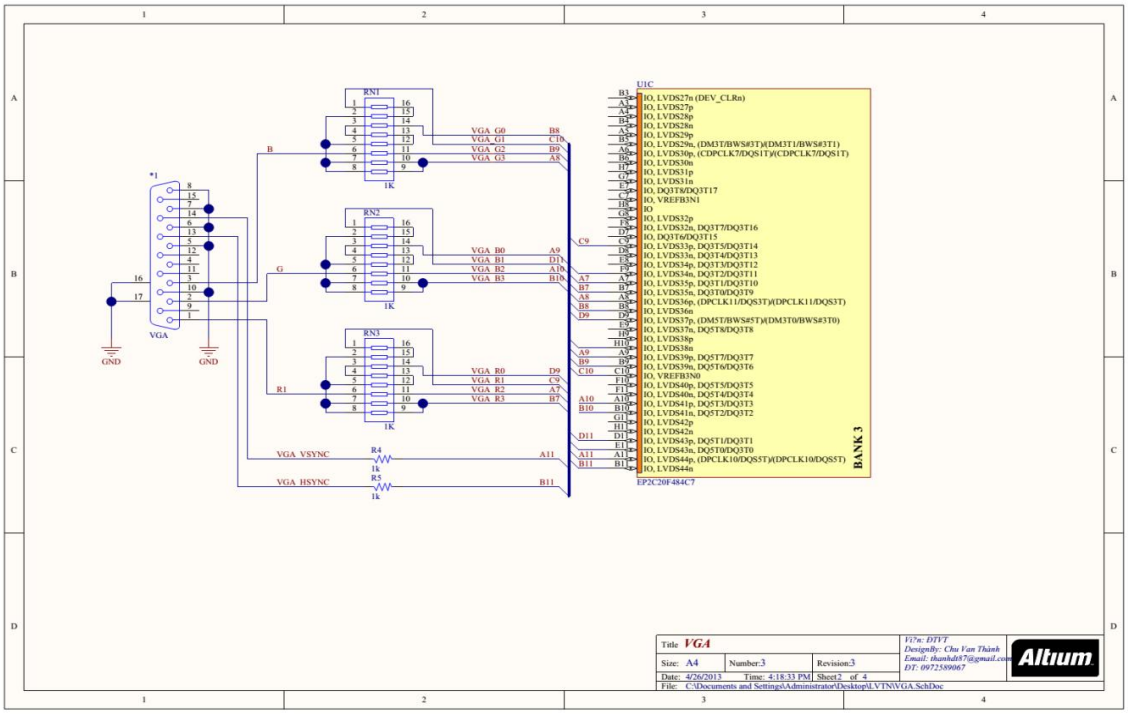
Tên	Mô tả
CLK	Đầu vào xung clock 50Mhz
RESET_N	Tín hiệu reset
ADDR [ASIZE-1:0]	Địa chỉ bộ nhớ read/write
CMD [2:0]	Đầu vào lệnh yêu cầu
CMDACK	Tín hiệu xác nhận lệnh
DATAIN [DSIZE-1:0]	Dữ liệu vào
DATAOUT [DSIZE-1:0]	Dữ liệu ra
DM [(DSIZE/8)-1:0]	Mặt nạ bytes dữ liệu
SA [11:0]	Địa chỉ hàng/cột của thanh ghi
BA[1:0]	Địa chỉ bank thanh ghi
CS_N [1:0]	Lựa chon chip
CKE	Tín hiệu cho phép xung clock SDRAM
RAS_N	Báo địa chỉ hàng
CAS_N	Báo địa chỉ cột

WE_N	Cho phép ghi data
DQ [DSIZE-1:0]	Bus dữ liệu
DQM [(DSIZE/8)-1:0]	Mặt nạ dữ liệu

3.3 Thiết kế khối hiển thị hình ảnh

3.3.1 Thiết kế mạch giao diện VGA

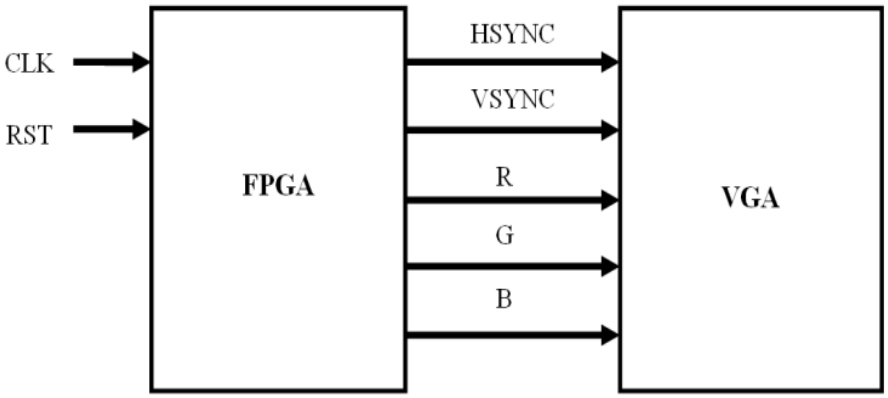
Trong thiết kế mạch giao diện VGA, tín hiệu màu R,G,B trên ngõ ra cổng VGA được tạo ra bởi 4 ngõ ra số trên FPGA và các điện trở để tạo tín hiệu điện áp thay đổi. Thiết kế được thể hiện trên hình 3.10



Hình 3.10: Sơ đồ nguyên lý khối VGA

3.3.2 Thiết kế mã chương trình điều khiển VGA

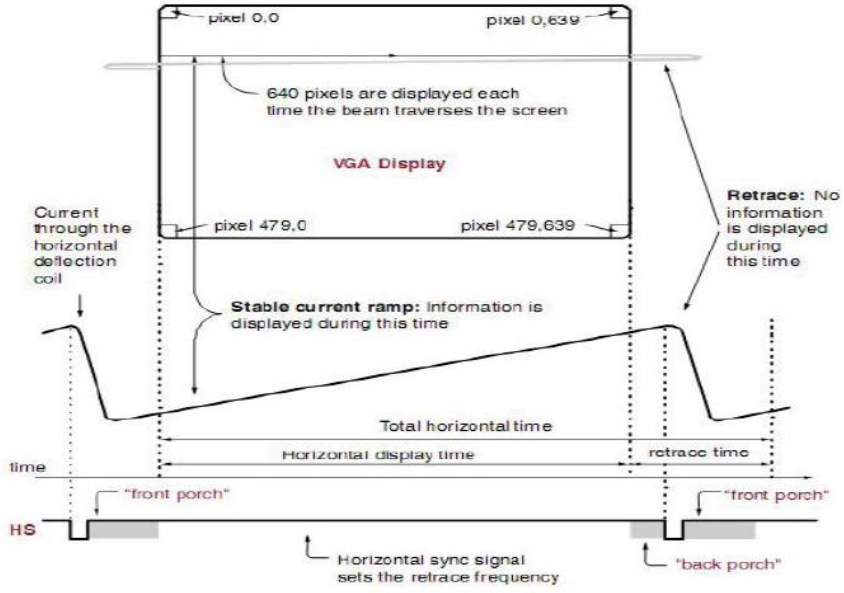
Để hiển thị hình ảnh với giao diện VGA cần có 5 tín hiệu bao gồm các tín hiệu đồng đbộ theo phươ ng ngang, phương dọc , h_sync và v _sync và 3 tín hiệu hình ảnh cho 3 màu đỏ , xanh, xanh dương.



Hình 3.11: Sơ đồ khối điều khiển VGA

Tên	Mô tả
CLK	Đầu vào xung clock 50Mhz
RESET_N	Tín hiệu reset
HSYNC	Tín hiệu đồng bộ ngang
VSYNC	Tín hiệu đồng bộ dọc
R	Tín hiệu màu đỏ
G	Tín hiệu màu xanh lục
B	Tín hiệu màu xanh dương

Mỗi một ảnh (hay frame) trên màn hình hiển thị là kết hợp của h dòng, mỗi dòng có w pixel. Kích thước của mỗi frame được biểu diễn w x h dưới dạng tiêu biểu gồm 640 x 480, 800 x 600, 1024 x 768 và 1280 x 1024.



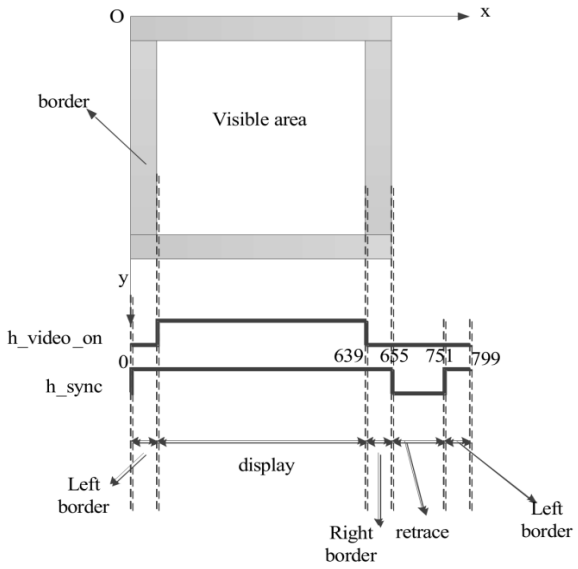
Hình 3.12: Sơ đồ thời gian hiển thị màn CRT

Để vẽ một khung hình, những mạch điện có trách nhiệm di chuyển dòng electron từ trái sang phải và từ trên xuống dưới dọc theo màn hình gọi là deflection circuit. Những mạch này yêu cầu phải có 2 tín hiệu đồng bộ để khởi động và dừng dòng electron tại đúng thời điểm để cho một dòng các điểm ảnh được vẽ dọc theo màn hình và mỗi dòng được điền theo cơ chế từ trên xuống dưới để tạo lên một ảnh.

3.3.2.1 Tín hiệu đồng bộ theo phương ngang

Sơ đồ thời gian quét của tín hiệu đồng bộ theo phương ngang như hình 3.13. Một chu kì của tín hiệu h _sync được chia làm 4 vùng :

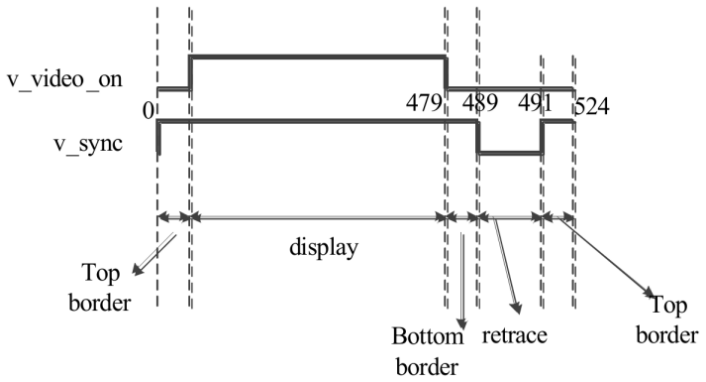
- Vùng hiển thị : là vùng mà các pixel được hiển thị trên màn hình .
- Vùng quét ngược : là vùng mà các tia điện tử quay ngược lại về phía cạnh bên trái.
- Vùng biên phải : là vùng màu đen bên phải , trong vùng này các tín hình ảnh bị vô hiệu hóa không được hiển thị .
- Vùng biên trái : là vùng màu đen bên trái và giống như vù ng biên bên phải , trong vùng này thì các tín hiệu hình ảnh không được hển thị



Hình 3.13: Sơ đồ thời gian quét theophương ngang

3.3.2.2 Tín hiệu đồng bộ theo phương dọc

Sơ đồ thời gian quét của tín hiệu đồng bộ theo phương dọc như s au:



Hình 3.14: Sơ đồ thời gian của tín hiệu quét theo phương dọc

Trong đó:

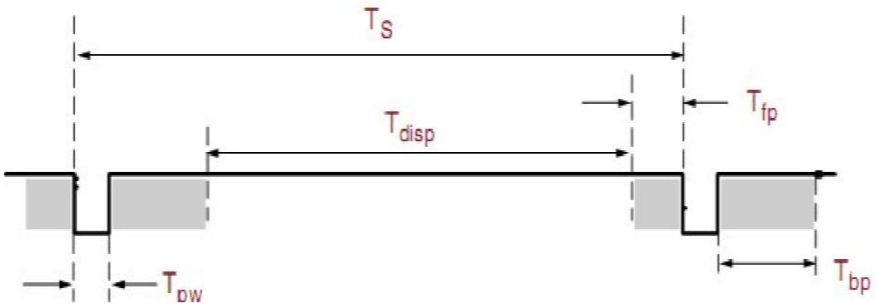
- Vùng hiển thị : là vùng mà các pixel được hiển thị trên màn hình .
- Vùng quét ngược : là vùng mà các tia điện tử quay ngược lại về phía cạnh bên trái.
- Vùng biên trên : là vùng màu đen bên trên , trong vùng này các tín hình ảnh bị vô hiệu hóa không được hiển thị .
- Vùng biên dưới : là vùng màu đen bên dưới và giống như vùng biên bên trên , trong vùng này thì các tín hiệu hình ảnh không được hển thị .

3.3.2.3 Tốc độ pixel

Gọi plà số pixel trong *l* hàng ngang , *l* là số đường ngang trong 1 màn hình và *s* là số màn hình hiển thị trong 1 s thì tốc độ pixel bằng *p*l**s.

Chú ý: để màn hình không bị nhấp nháy t hì s phải lớn hơn hoặc bằng 24.

Với màn hình có độ phân giải 640x480 và số màn hình hiển thị trong 1s là 60 *hình/s* thì *p =800 pixel/line, l=525 line/screen, s=60 screen/second* tốc độ pixel bằng *800x525x60 ≈25Mpixel/s.*



Hình 3.15: Thời gian điều khiển chế độ VGA 640 x 480

Bảng 3.1: Thời gian hiển thị với chế độ VGA 640 x 480

Symbol	Parameter	Vertical Sync			Horizontal Sync	
		Time	Clocks	Lines	Time	Clocks
T_s	Sync pulse time	16.7 ms	416,800	521	32µs	800
T_{DISP}	Display time	15.36 ms	384,000	480	25.6µs	640
T_{PW}	Pulse width	64µs	1,600	2	3.84µs	96
T_{FP}	Front porch	320µs	8,000	10	640ns	16
T_{BP}	Back porch	928µs	23,200	29	1.92µs	48

3.4 Kết luận chương

Chương này tập trung vào thiết kế sơ đồ khối chức năng hệ thống thu thập hình ảnh tốc độ cao. Việc lựa chọn chip, thiết kế mạch phần cứng và thiết kế mã chương trình điều khiển trên FPGA của mỗi khối chức năng, bao gồm: khối thu thập hình ảnh, bộ nhớ lưu dữ liệu hình ảnh SDRAM, màn hình hiển thị hình ảnh VGA và phần cứng ngoại vi.

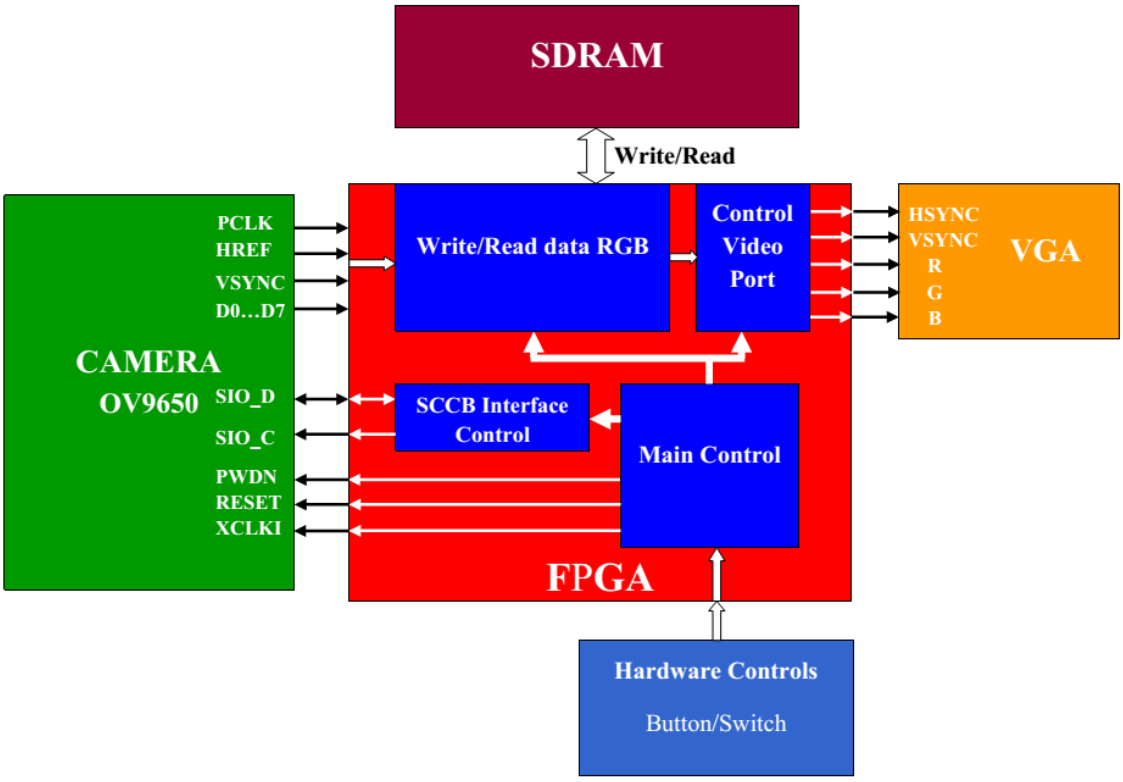
Chương 4: MÔ PHỎNG VÀ THỰC HIỆN HỆ THỐNG TRÊN FPGA

Trong các chương 1, 2 và 3 đã trình bày về các nội dung liên quan đến các thiết kế và phương pháp thực hiện thu và hiện thị hình ảnh. Chương 3 đưa ra thiết kế mạch phần cứng và mã chương trình điều khiển cho từng khối của hệ thống.

Kết quả mô phỏng từng chế độ, chức năng làm việc của từng khối trong hệ thống được thực hiện trên phần mềm Quatus II và quá trình thực hiện hệ thống trên Kit DE1 của hãng Altera cung cấp được trình bày trong chương 4.

4.1 Thực hiện hệ thống trên FPGA

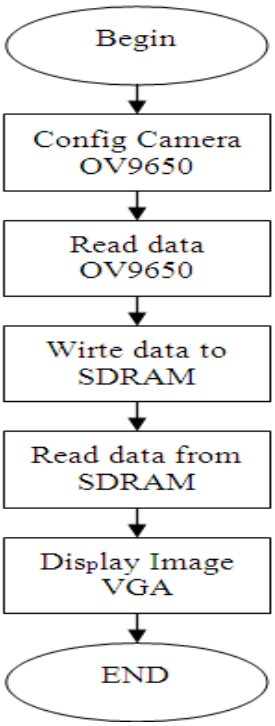
Toàn bộ hệ thống các thiết bị được điều khiển bởi khối xử lý trung tâm FPGA, FPGA điều khiển bao gồm các module bộ cảm biến CMOS, điều khiển đồng bộ, cấu hình và điều khiển truyền dữ liệu nối tiếp, module điều khiển SDRAM và module điều khiển hiển thị ảnh. Sơ đồ khối cụ thể của hệ thống được thể hiện trong hình 4.1 [2].



Hình 4.1: Sơ đồ khối thực hiện hệ thống

Sau khi thiết kế mạch hoàn tất, để kiểm tra các đầu vào là chính xác việc mô phỏng thiết kế FPGA là rất cần thiết, các phần mềm mô phỏng thiết kế chính sử dụng phần mềm Quartus II để có được những module FPGA nội bộ.

Chương trình mô tả hệ thống thu và hiển thị hình ảnh trên nền FPGA được thực hiện theo quy trình hoạt động thể hiện trên hình 4.2



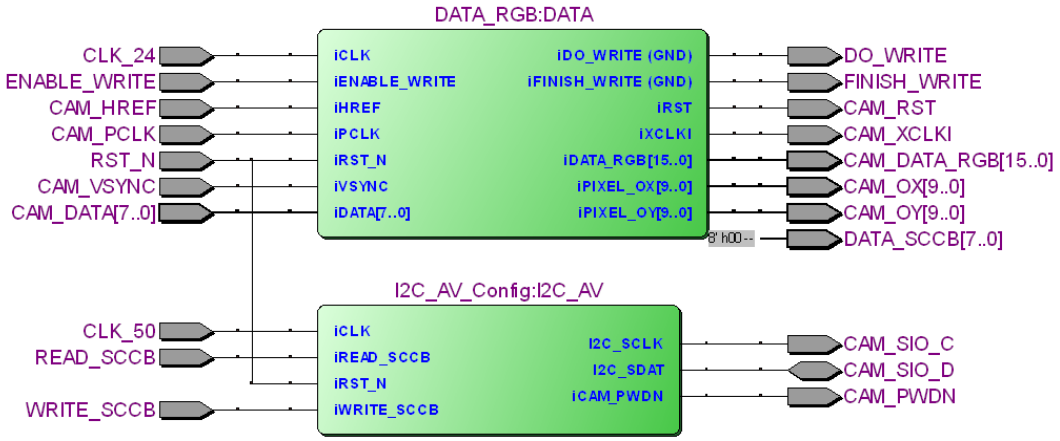
Hình 4.2: Quy trình hoạt động hệ thống

Để hệ thống có thể thu được những hình ảnh theo mong muốn thì cần phải cấu hình lại chế độ làm việc cho camera OV9650. Sau đó đọc tín hiệu dữ liệu hình ảnh thu được từ cảm biến ảnh, dữ liệu này thu được và được xử lý rồi lưu tạm thời vào SDRAM, sau đó dữ liệu hình ảnh trong SDRAM được đọc ra và hiển thị lên màn hình VGA.

4.1.1 Khối điều khiển cảm biến hình ảnh CMOS

Chức năng các module điều khiển hoạt động của cảm biến hình ảnh CMOS yêu cầu điều khiển đầu vào thiết kế, mô phỏng chức năng của khối điều khiển, phương pháp tiếp cận thiết kế truyền thống là khó khăn để đáp ứng yêu cầu. Hệ thống sử dụng sự kết hợp của ngôn ngữ mô tả phần cứng FPGA để mô tả cho chức năng cảm biến ảnh CMOS [20]. Các chức năng được phân chia theo giản đồ thời gian, để xác

định các tín hiệu đầu vào và đầu ra, và cho mỗi module thực hiện mô phỏng chức năng.

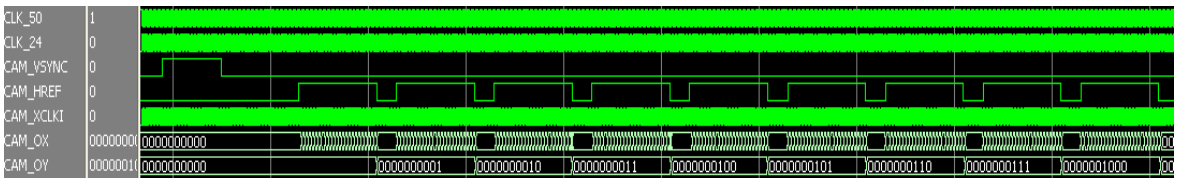


Hình 4.3: Sơ đồ khối điều khiển cảm biến OV9650

Chất lượng hình ảnh cảm biến được quyết định bởi các thanh ghi nội bộ, vì vậy cần phải cấu hình các thanh ghi bên trong để chọn chế độ hoạt động. việc cấu hình được thực hiện thông qua giao diện SCCB.

Giản đồ thời gian của bộ cảm biến hình ảnh có thể được chia thành hai phần chính. Liên quan đến thời gian của phần đọc dữ liệu các điểm ảnh pixel, tức là kiểm soát thời gian đồng bộ hóa dữ liệu đầu ra và sô điểm ảnh lấy mẫu của một khung hình. Một phần khác là thời gian đọc các điểm ảnh, thời gian các tín hiệu đồng bộ.

Hình 4.4 cho thấy kết quả mô phỏng với giản đồ thời gian đọc ảnh từ cảm biến OV9650 với khung hình 640x480.

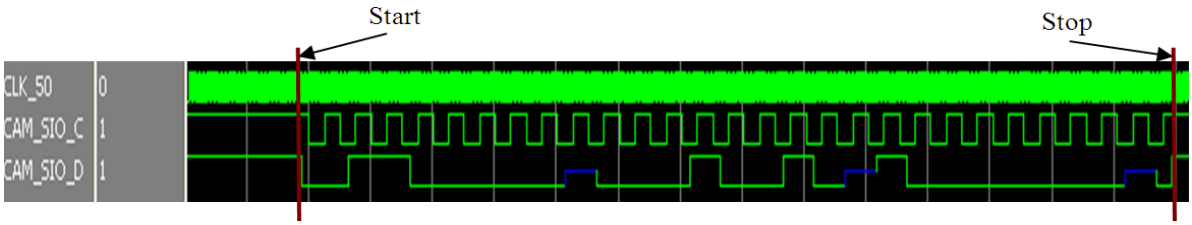


Hình 4.4: Kết quả mô phỏng thu ảnh từ OV9650

Để thiết lập chế độ hoạt động cho OV9650 ta cần cấu hình lại cho chip bằng cách thiết đặt giá trị trong các thanh ghi tương ứng của cảm biến ảnh. Dữ liệu được truyền trên giao diện điều khiển nối tiếp 2 dây, SIO_C và SIO_D.

Khi truyền chuỗi dữ liệu cần phải có các bit đồng bộ: 1 bit cho trạng thái IDE, 2 bit để thiết lặp cờ START, 3 bit để chờ ACK cho OV9650 xác nhận, 3 bit để thiết

lập cờ STOP và báo kết thúc chuỗi. Hình 4.5 là các kết quả mô phỏng quá trình điều khiển.



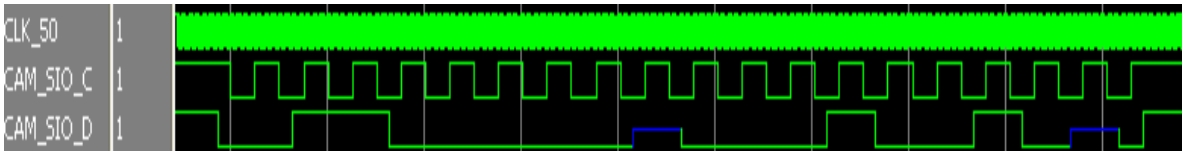
Hình 4.5: Kết quả mô phỏng ghi dữ liệu xuống OV9650

Dữ liệu được truyền mô phỏng với dòng bit như sau:

	IP address							R/W		Sub – address									Write data									
start	0	1	1	0	0	0	0	0	x	0	0	0	1	0	0	1	0	x	1	0	0	0	0	0	0	0	x	stop
	60H								x	12H								x	80H								x	

Để đọc dữ liệu của một thanh ghi trong bộ nhớ của OV9650 ta cần gửi địa chỉ thanh ghi cần đọc, sau đó gửi lệnh đọc đọc dữ liệu

- Gửi địa chỉ thanh ghi cần đọc dữ liệu



Hình 4.6: Kết quả mô phỏng truyền 2 Phase ghi

Dữ liệu được truyền với dòng bit như sau:

	IP address							R/W		Sub – address									
start	0	1	1	0	0	0	0	0	x	0	0	0	1	0	0	1	0	x	stop
	60H								x	12H								x	

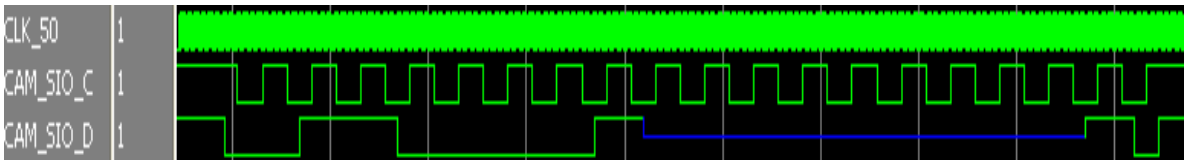
Để thiết lập chế độ hoạt động khung dữ liệu ảnh ngõ ra của bộ cảm biến OV9650 ta cần điều chỉnh các thông số các thanh ghi như bảng 4.1:

Bảng 4.1: Tham số cấu hình OV9650 [21]

VGA Preview, 25fps, 24 Mhz input clock	*Contrast 0*
SCCB_salve_Address=0x60;	SCCB_salve_Address=0x60;
write_SCCB(0x11, 0x80);	write_SCCB(0x7C ,0x04);
write_SCCB(0x6b, 0x0a);	write_SCCB(0x7D,0x07);

write_SCCB(0x39, 0x50);	write_SCCB(0x7E,0x10);
write_SCCB(0x38, 0x92);	write_SCCB(0x7F,0x28);
write_SCCB(0x35, 0x81);	write_SCCB(0x80 ,0x36);
write_SCCB(0x2a, 0x10);	write_SCCB(0x81 ,0x44);
write_SCCB(0x2b, 0x40);	write_SCCB(0x82 ,0x52);
write_SCCB(0x6a, 0x7d);	write_SCCB(0x83 ,0x60);
Light Mode Auto	write_SCCB(0x84 ,0x6C);
SCCB_salve_Address=0x60;	write_SCCB(0x85 ,0x78);
write_SCCB(0x13 ,0xe7);	write_SCCB(0x86 ,0x8C);
write_SCCB(0x11 ,0x81);	write_SCCB(0x87 ,0x9E);
write_SCCB(0x41 ,0x02);	write_SCCB(0x88 ,0xBB);
write_SCCB(0x3f ,0xa6);	write_SCCB(0x89 ,0xD2);
write_SCCB(0x3d ,0x92);	write_SCCB(0x8A,0xE6);
write_SCCB(0x66 ,0x01);	write_SCCB(0x6C ,0x40);
write_SCCB(0x14 ,0x1e);	write_SCCB(0x6D,0x30);
Color Saturation 0	write_SCCB(0x6E,0x48);
SCCB_salve_Address=0x60;	write_SCCB(0x6F,0x60);
write_SCCB(0x4f ,0x3a);	write_SCCB(0x70 ,0x70);
write_SCCB(0x50 ,0x3d);	write_SCCB(0x71 ,0x70);
write_SCCB(0x51 ,0x03);	write_SCCB(0x72 ,0x70);
write_SCCB(0x52 ,0x12);	write_SCCB(0x73 ,0x70);
write_SCCB(0x53 ,0x26);	write_SCCB(0x74 ,0x60);
write_SCCB(0x54 ,0x38);	write_SCCB(0x75 ,0x60);
Brightness 0	write_SCCB(0x76 ,0x50);
SCCB_salve_Address=0x60;	write_SCCB(0x77 ,0x48);
write_SCCB(0x24 ,0x70);	write_SCCB(0x78 ,0x3A);
write_SCCB(0x25 ,0x64);	write_SCCB(0x79 ,0x2E);
write_SCCB(0x26 ,0xc3);	write_SCCB(0x7A,0x28);
Special effects Normal	write_SCCB(0x7B ,0x22);
SCCB_salve_Address=0x60;	
write_SCCB(0x3a, 0x01);	
write_SCCB(0x67, 0xc0);	
write_SCCB(0x68, 0x80);	

Gửi lệnh đọc dữ liệu



Hình 4.7: Kết quả mô phỏng truyền 2 Phase đọc

Dữ liệu được truyền nhận với dòng bit như sau:

start	IP address							R/W		Read data									stop
start	0	1	1	0	0	0	0	1	x	D7	D6	D5	D4	D3	D2	D1	D0	x	stop
	61H								x	DATA								x	

Sau khi thiết lập chế độ hoạt động cũng như khung dữ liệu điểm ảnh đầu ra cho camera OV9650, lúc này hệ thống chỉ việc đọc dữ liệu từng điểm ảnh của khung hình ở ngõ ra dữ liệu điểm ảnh cùng tín hiệu đồng bộ khung hình VSYNC và tín hiệu xung clock PCLK.

Trong hệ thống thiết lập chế độ dữ liệu đầu ra khung hình là YUV422, dữ liệu đầu ra tương ứng U0 Y0 V0 Y1 U2 Y2 V2 Y3 U4 Y4 V4… với các dữ liệu từng điểm ảnh như sau:

Pixel Number	Pixel Values
0	U0 Y0 V0
1	U0 Y1 V0
2	U2 Y2 V2
3	U2 Y3 V2
4	U4 Y4 V4
…	…

Từ dữ liệu YUV nhận được, để hiển thị trên màn hình LCD cần chuyển tín hiệu sang RGB24 với hàm như sau [21]:

$R = Y + 1.371(Cr - 128)$

$G = Y - 0.698(Cr - 128) - 0.336(Cb - 128)$

$B = Y + 1.732(Cb - 128)$

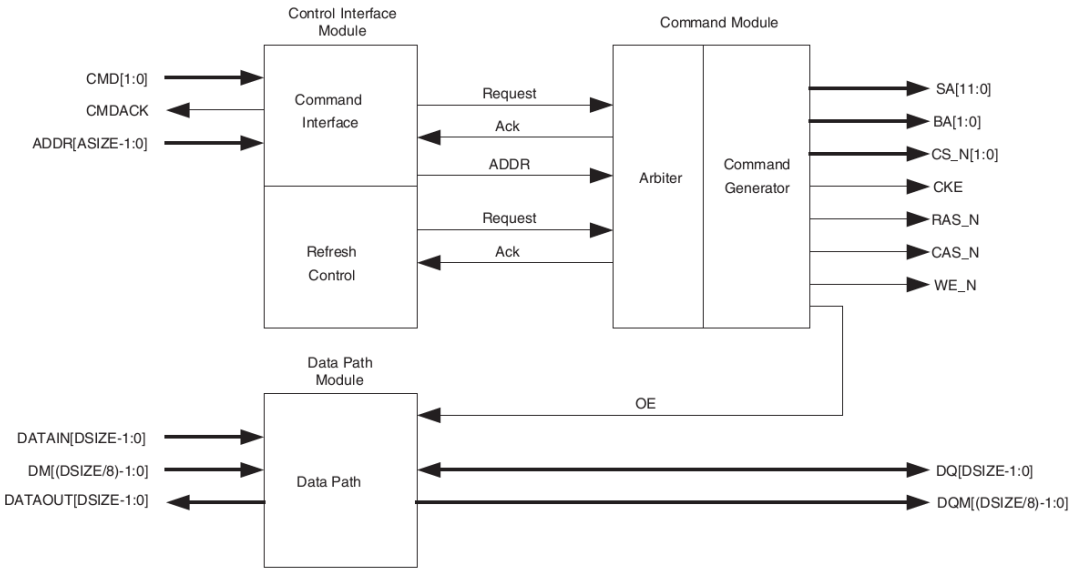
4.1.2 Khối điều khiển đọc, ghi dữ liệu SDRAM

Việc thực hiện lưu trữ dữ liệu tốc độ cao dung lượng lớn là một công nghệ quan trọng trong hệ thống thu thập dữ liệu. Phần này chủ yếu là thiết kế module lưu trữ dữ liệu giao diện lệnh điều khiển các hoạt động khác nhau của bộ nhớ. Trong hệ thống sử dụng FPGA điều khiển đọc và ghi dữ liệu cho SDRAM [17].

4.1.2.1 Thiết kế khối điều khiển DDR SDRAM

SDR SDRAM Control bao gồm 4 module chính [14], [15], [17]: Bộ điều khiển SDRAM (SDRAM controller), giao diện điều khiển (control interface), lệnh điều khiển (command), và đường truyền dữ liệu (data path). Module điều khiển SDRAM là module cấp cao khởi tạo ba module cấp thấp hơn và liên kết toàn bộ thiết kế với nhau. Module giao diện điều khiển nhận lệnh và địa chỉ bộ nhớ liên quan từ các thiết bị chủ, giải mã lệnh và truyền các yêu cầu cho module command. Module command nhận các lệnh và địa chỉ từ các module control interface, và tạo ra các lệnh thích hợp tới SDRAM. Các module data path xử lý các hoạt động đường dẫn dữ liệu trong lệnh WRITEA và READA. Các module điều khiển SDRAM cũng khởi tạo một PLL được sử dụng trong các chế độ CLOCK_LOCK để cải thiện thời gian I/O.

PLL không phải là điều cần thiết để hoạt động của SDR SDRAM Control và có thể không sử dụng. Hình 4.8 cho thấy sơ đồ khối SDR SDRAM Control.

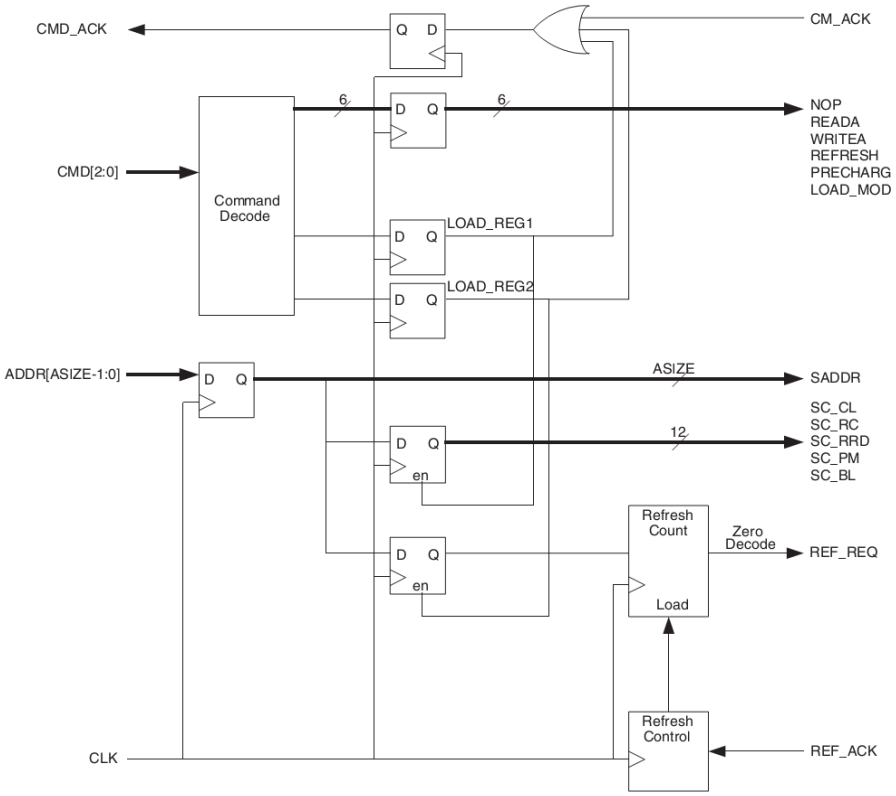


Hình 4.8: Sơ đồ khối điều khiển SDRAM

(1) Module Control Interface

Module giao diện điều khiển giải mã và đăng ký các lệnh từ máy chủ, và thông qua giải mã lệnh NOP, WRITEA, READA, REFRESH, Precharge, và LOAD_MODE, và Địa chỉ đến các module lệnh. Các lệnh LOAD_REG1 và LOAD_REG2 được giải mã và sử dụng trong nội bộ để tải REG1 và REG2 đăng ký với giá trị từ ADDR.

Module Control Interface bao gồm 16 bit truy cập và mạch điều khiển được sử dụng để tạo ra lệnh làm mới tuần hoàn cho module command. 16 bit được nạp với giá trị từ REG2 và đếm xuống 0. REFRESH_REQ ngõ ra được khẳng định khi truy cập đạt đến 0 và vẫn giữ cho đến khi module command xác nhận yêu cầu. Xác nhận từ module command tạo ra bộ đếm xuống được nạp lại với REG2 và lặp đi lặp lại quá trình. REG2 là một giá trị 16-bit cho khoảng thời gian giữa REFRESH với các vấn đề SDR SDRAM điều khiển. Giá trị được thiết lập bởi phương trình *int (refresh_period / clock_period)*.



Hình 4.9: Sơ đồ khối Module Control Interface

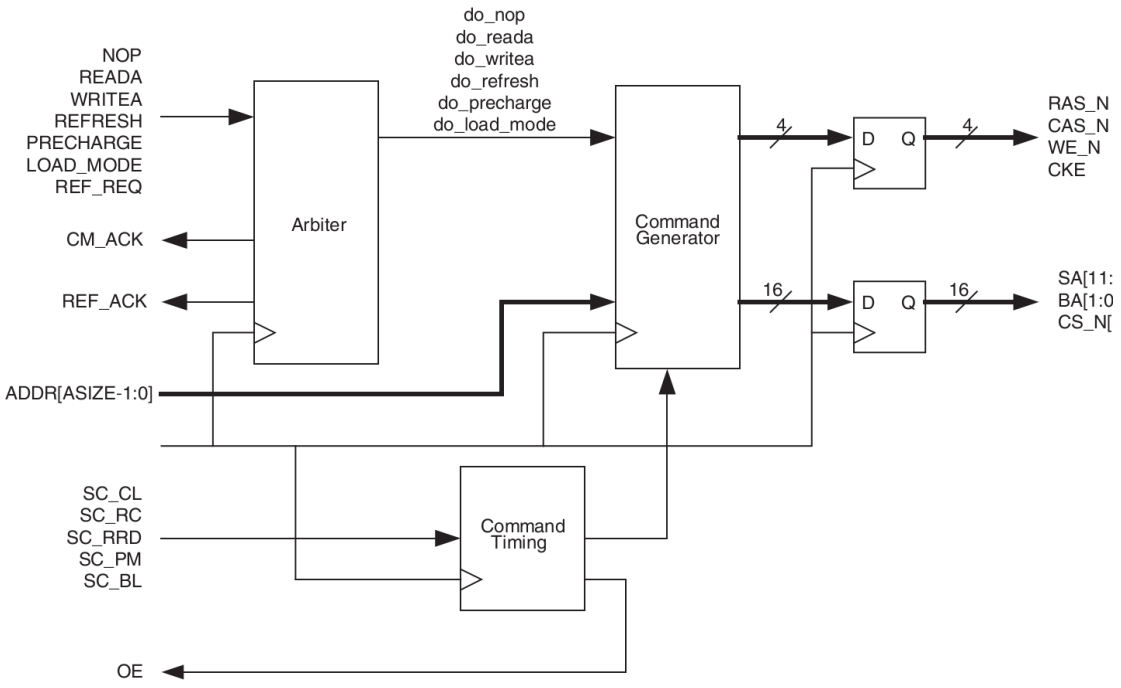
Ví dụ, nếu một thiết bị SDRAM được kết nối với bộ điều khiển SDR SDRAM có 64ms, yêu cầu 4096 chu kỳ làm mới, thiết bị phải có một lệnh REFRESH cấp cho nó ít nhất 64 ms/4096 = 15,625 µs.

Nếu SDRAM và SDR SDRAM điều khiển được bởi tốc độ 100MHz, giá trị tối đa của REG2 15,625 µs/0.01µs = 1562d.

(2) Module command

Module command nhận lệnh giải mã từ các module giao diện điều khiển, yêu cầu làm mới từ logic điều khiển làm mới, và tạo ra các lệnh thích hợp tới SDRAM. Module này bao gồm một arbiter điều chỉnh giữa các lệnh từ giao diện thiết bị chủ và yêu cầu refresh từ logic điều khiển làm mới.

Các yêu cầu refresh từ logic điều khiển làm mới có ưu tiên hơn các lệnh. Nếu một lệnh từ máy chủ đến cùng một lúc hoặc trong một hoạt động làm mới ẩn, Arbiter tắt thiết bị chủ bằng tín hiệu CMDACK cho đến khi hoạt động làm mới ẩn hoàn tất. Nếu nhận được một lệnh làm mới ẩn trong khi thiết bị chủ hoạt động trong tiến trình, làm mới ẩn được tổ chức cho đến khi hoạt động thiết bị chủ hoàn tất.



Hình 4.10: Sơ đồ khối Module Command

Sau khi Arbiter đã chấp nhận một lệnh thiết bị chủ, lệnh được thông qua tới Command generator của module Command. Module command sử dụng ba đăng ký để tạo ra thay đổi thời gian phù hợp giữa các lệnh được cấp cho SDRAM. Một đăng ký thay đổi được sử dụng để kiểm soát thời gian lệnh ACTIVATE, kiểm soát các vị trí của lệnh READA hoặc WRITEA, thời điểm lệnh thời gian, xác định hoạt động cuối cùng đã được hoàn thành.

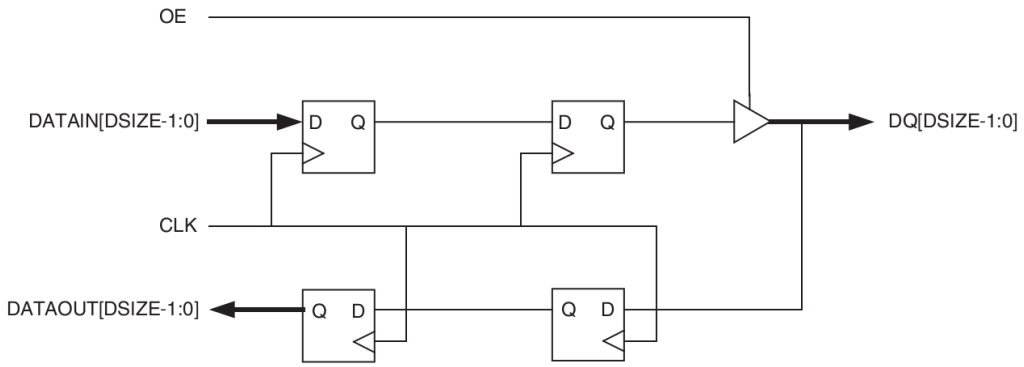
Module command cũng thực hiện ghép kênh của địa chỉ SDRAM. Phần hàng của địa chỉ được ghép với SDRAM đầu ra [11:0] trong lệnh ACTIVATE(RAS). Phần cột được ghép sau với đầu ra địa chỉ SDRAM trong một lệnh READA(CAS) hoặc WRITEA.

Tín hiệu đầu ra OE được tạo ra bởi module command để kiểm soát các vùng đệm trạng thái trong giai đoạn cuối của DATAIN đường dẫn trong các module data path.

(3) Module Data Path

Module Data Path cung cấp các dữ liệu giao diện SDRAM để lưu trữ. Dữ liệu thiết bị chủ được chấp nhận trên DATAIN cho Lệnh WRITEA và dữ liệu được cung cấp cho các thiết bị chủ trên DATAOUT trong khi lệnh READA. Hình 4.11 cho thấy sơ đồ khối module Data Path. DATAIN đường dẫn bao gồm một đường dẫn 2 giai đoạn để sắp xếp dữ liệu với CMDACK và các lệnh được cấp cho SDRAM.

DATAOUT bao gồm đường dẫn 2 giai đoạn đăng ký dữ liệu từ SDRAM trong một lệnh READA. Thời gian trễ DATAOUT đường dẫn có thể được giảm đến 1 hoặc 0, sự điều chỉnh đó ảnh hưởng đến mối quan hệ của CMDACK đến sự thay đổi DATAOUT.



Hình 4.11: Sơ đồ khối Module Data Path

4.1.2.2 Kết quả mô phỏng quá trình mô tả SDRAM

Khối điều khiển SDRAM cung cấp một giao diện đồng bộ lệnh với SDRAM và điều khiển một số đăng ký. Bảng 4.2 cho thấy các lệnh được mô tả trong các phần sau đây. Các lệnh áp dụng theo quy tắc sau [26]:

- Tất cả các lệnh, ngoại trừ NOP, được điều khiển bởi người dùng vào CMD [2:0]; ADDR và DATAIN được thiết lập phù hợp cho lệnh yêu cầu.
- Để CMDACK xác nhận các lệnh điều khiển cần một xung clock
- Đối với các lệnh READA hoặc WRITEA, người sử dụng đọc hoặc ghi dữ liệu trên DATAOUT và DATAIN
- Người sử dụng phải điều khiển lệnh NOP vào CMD [2:0] sau khi CMDACK xác nhận.

Bảng 4.2: Giao diện lệnh

Lệnh	Giá trị	Mô tả
NOP	000b	Không có hoạt động.
READA	001b	SDRAM đọc với Precharge tự động.
WRITEA	010b	SDRAM viết Precharge tự động.
REFRESH	011b	SDRAM tự động làm mới.
Precharge	100b	SDRAM Precharge tất cả các ngân hàng.
LOAD_MODE	101b	SDRAM tải chế độ đăng ký.
LOAD_REG1	110b	Cấu hình tải điều khiển đăng ký.
LOAD_REG2	111b	Tải điều khiển chu kỳ cập nhật đăng ký.

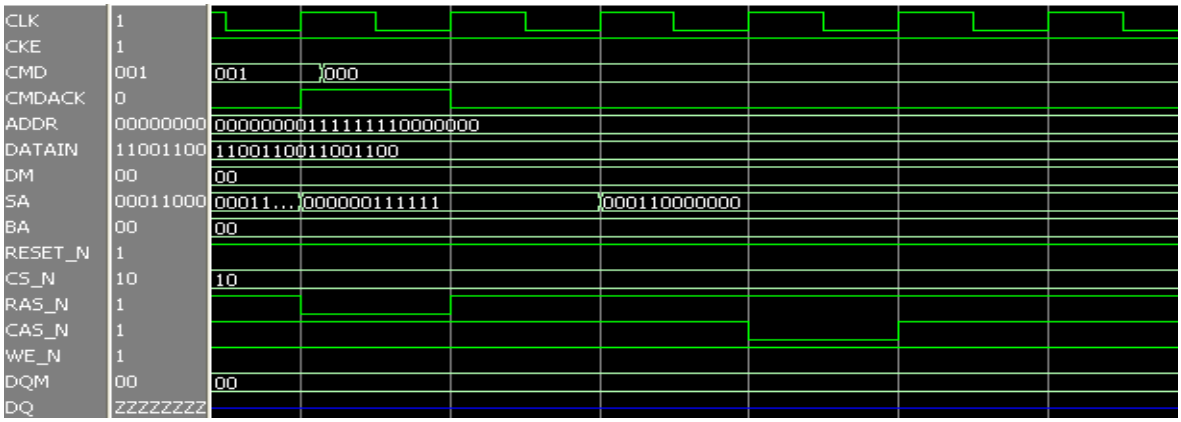
❖ Lệnh NOP

Lệnh NOP là lệnh không có hoạt động để điều khiển. Khi lệnh NOP được phát hiện bởi bộ điều khiển, nó thực hiện một lệnh NOP sau một xung clock. Một lệnh NOP phải được cấp chu kỳ xung clock sau khi bộ điều khiển đã xác nhận lệnh. Lệnh NOP không ảnh hưởng truy cập SDRAM.

❖ Lệnh READA

Lệnh READA để SDR SDRAM control thực hiện đọc với tự động Precharge SDRAM tại địa chỉ bộ nhớ được chỉ định bởi ADDR. SDR SDRAM điều khiển kích hoạt SDRAM bởi lệnh READA. Các dữ liệu đọc đầu tiên xuất hiện trên DATAOUT (RCD + CL + 2) sau khi SDR SDRAM control xác nhận CMDACK. Trong một lệnh READA người sử dụng phải giữ DM mức thấp. Khi bộ điều khiển được cấu hình cho chế độ full-page, lệnh READA thành READ (READ không có tự động Precharge). Hình 4.12 cho thấy sơ đồ thời gian cho một lệnh READA. Lệnh READA được mô tả các hoạt động theo trình tự sau đây:

- Xác nhận sử dụng READA, ADDR và DM
- SDR SDRAM Control xác nhận CMDACK để xác nhận lệnh và đồng thời bắt đầu gửi lệnh cho SDRAM
- Sử dụng một lệnh NOP sau khi CMD xác nhận
- CMDACK là giá trị đọc đầu tiên trên DATAOUT, phần sau là dữ liệu đọc theo mỗi chu kỳ xung clock kế tiếp.



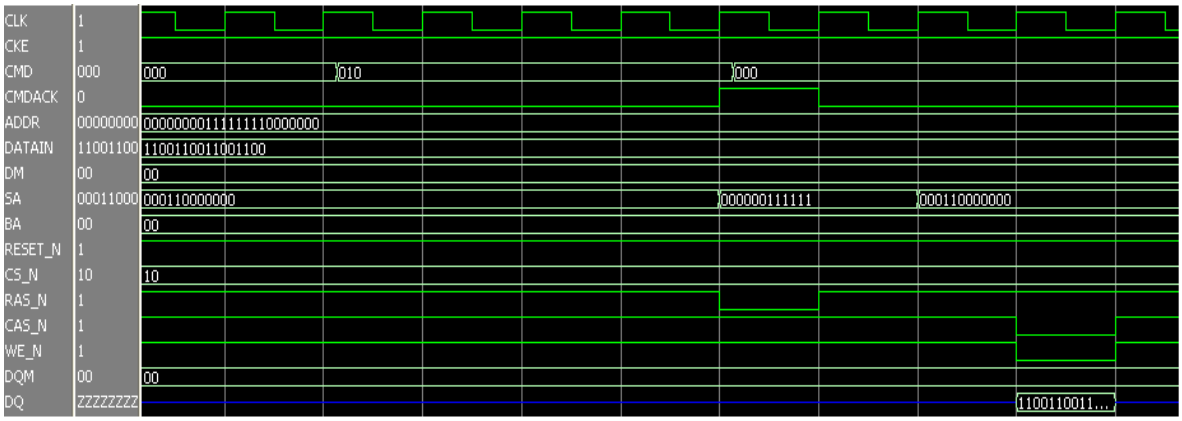
Hình 4.12: Sơ đồ thời gian SDRAM READA

❖ Lệnh WRITEA

Lệnh WRITEA điều khiển SDR SDRAM để thực hiện ghi với tự động precharge SDRAM tại địa chỉ bộ nhớ được chỉ định bởi ADDR. SDR SDRAM điều khiển sẽ phát hành một lệnh kích hoạt SDRAM theo sau một lệnh WRITEA. Dữ liệu đầu tiên trong chuỗi phải được trình bày với WRITEA và địa chỉ ADDR. Các xung clock đầu tiên phải bắt đầu cùng với dữ liệu DM mong muốn vào SDR SDRAM Controller (t_{RCD}-2) clock sau khi điều khiển SDR SDRAM đã xác nhận lệnh WRITEA.

Khi SDR SDRAM Controller ở chế độ full-page WRITEA trở thành WRITE (WRITE không tự động precharge). Hình 4.13 cho thấy một biểu đồ thời gian cho một lệnh WRITEA. Lệnh WRITEA được mô tả các hoạt động theo trình tự sau đây:

- Xác nhận sử dụng WRITEA, ADDR, dữ liệu giá trị đầu tiên ghi DATAIN và DM
- SDR SDRAM Control xác nhận CMDACK để xác nhận lệnh và đồng thời bắt đầu gửi lệnh cho SDRAM
- Sử dụng một lệnh NOP sau khi CMD xác nhận
- Sử dụng DATAIN and DM điều khiển SDRAM SDR



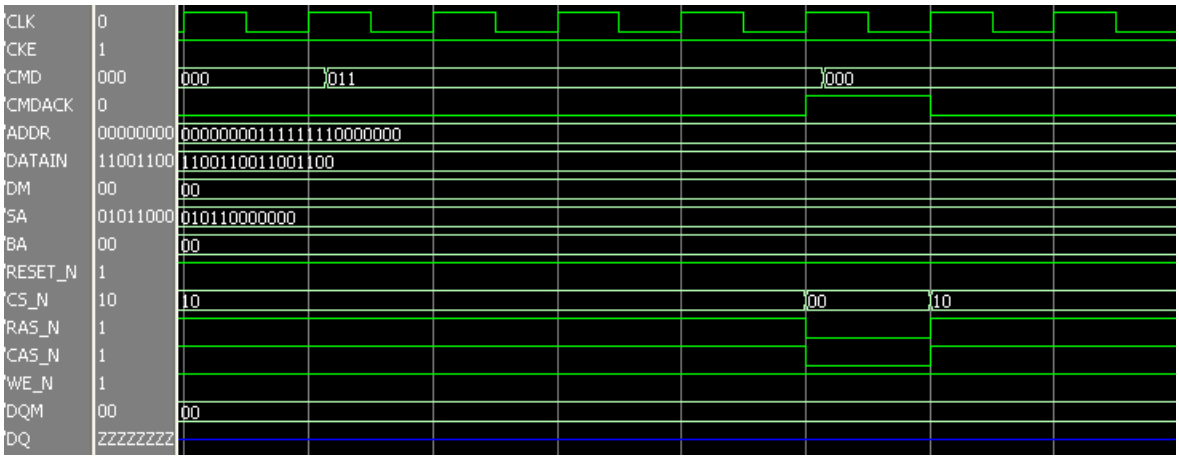
Hình 4.13: Sơ đồ thời gian SDRAM WRITEA

❖ Lệnh REFRESH

Lệnh REFRESH để SDR SDRAM control thực hiện một lệnh ARF tới SDRAM. SDR SDRAM control nhận lệnh REFRESH với CMDACK. Hình 4.14

cho thấy sơ đồ thời gian của lệnh REFRESH. Lệnh REFRESH được mô tả các hoạt động theo trình tự sau đây:

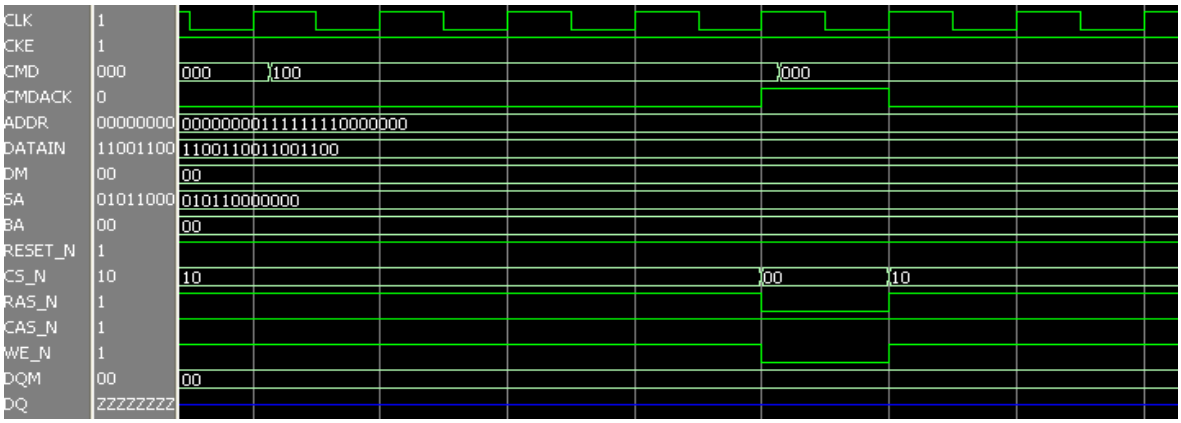
- Xác nhận sử dụng REFRESH trên đầu vào CMD
- SDR SDRAM Control xác nhận CMDACK để nhận lệnh và đồng thời bắt đầu gửi lệnh cho SDRAM
- Sử dụng một lệnh NOP sau khi CMD xác nhận



Hình 4.14: Sơ đồ thời gian SDRAM REFRESH

❖ Lệnh Precharge

Lệnh Precharge điều khiển SDR SDRAM để thực hiện một lệnh PCH SDRAM. SDR SDRAM Controller thừa nhận lệnh với CMDACK. Lệnh PCH cũng được dùng để SDRAM dừng lại. Sử dụng Precharge chỉ được hỗ trợ ở chế độ đầy đủ trang.

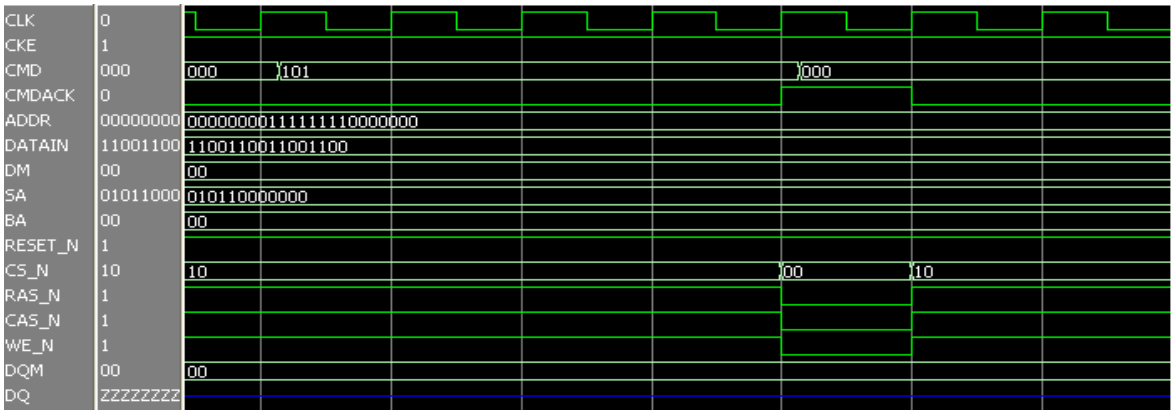


Hình 4.15: Sơ đồ thời gian SDRAM Precharge

❖ Lệnh LOAD MODE

Lệnh LOAD_MODE để SDR SDRAM control thực hiện một lệnh LMR SDRAM. Các giá trị đó được viết vào chế độ SDRAM đăng ký phải có ADDR[11:0] với lệnh LOAD_MODE. Giá trị ADDR [11:0] gán trực tiếp vào các chân A11-A0 SDRAM khi SDR SDRAM điều khiển LMR SDRAM. Hình 4.16 cho thấy ví dụ sơ đồ thời gian. Sau đây trình tự mô tả các hoạt động chung của một lệnh LOAD_MODE:

- Xác nhận sử dụng LOAD_MODE trên CMD
- SDR SDRAM Control xác nhận CMDACK để nhận các lệnh và đồng thời bắt đầu gửi lệnh cho SDRAM
- Sử dụng một lệnh NOP sau khi CMD xác nhận



Hình 4.16: Sơ đồ thời gian SDRAM LOAD MODE

Từ các kết quả mô phỏng cho hệ thống ta thấy việc điều khiển đọc, ghi dữ liêu với SDRAM đúng với nguyên tắc hoạt động của SDRAM.

4.1.3 Khối điều khiển hiển thị VGA

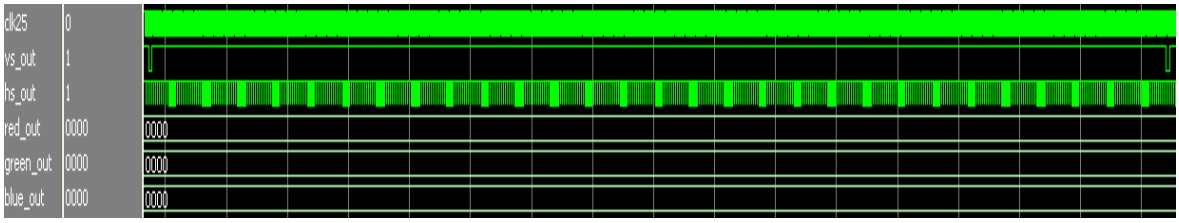
Sơ đồ cấu trúc của bộ hiển thị hình ảnh lên VGA như trên hình 4.17. Mỗi pixel có thể là 1, 2, 4, 8 hoặc 16 bit vì thế nên nội dung của thanh ghi pixel này được dịch sau mỗi xung clock để thay thế pixel hiện tại theo thứ tự các bít có trọng số thấp đến cao. Các bit này được gửi đến color map circuit để chuyển các pixel này sang các giá trị red, green và blue rồi gửi đến bộ DAC bên ngoài.

Hình 4.17: Sơ đồ khối cấu trúc của điều khiển VGA

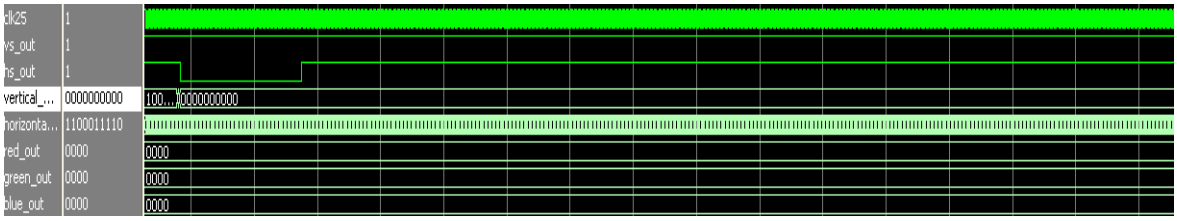
Hai mạch tạo xung đồng bộ (pulse generation circuit) được dùng để tạo các xung đồng bộ dọc và nằm ngang (horizontal và vertical). Các đầu ra là tín hiệu gate một chu kì trùng khớp với sườn lên của xung đồng bộ ngang (horizontal_sync pulse), tín hiệu gate này nối với tín hiệu clock – enable của vertical_sync vì thế nên clock – enable chỉ cập nhật bộ đếm thời gian sau mỗi dòng pixel (line of pixels). Tín hiệu gate của vertical sync được dùng như tín hiệu báo kết thúc một frame cho các khối dữ liệu pixel bên ngoài, đồng thời nó cũng reset và xóa toàn bộ nội dung của pixel buffer nên bộ điều khiển VGA luôn khởi động từ trạng thái xóa sạch hoàn toàn với mọi frame.

Bộ tạo tín hiệu đồng bộ cũng tạo ra các tín hiệu horizontal và vertical blanking. Khi dùng phép toán OR logic ta được tín hiệu blanking toàn cục. Các tín hiệu blanking được kết hợp với các bit có trọng số thấp hơn ở bộ đếm horizontal pixel counter để xác định khi nào đọc pixel từ bộ đệm. Ví dụ, nếu mỗi pixel có độ rộng 16 bit, thì từ 16 bit sẽ cần được đọc sau mỗi chu kì clock. Vì thế nên hoạt động đọc được khởi tạo bất cứ khi nào tín hiệu video không trống và 2 bit thấp của bộ đếm pixel đều bằng 0.

Dưới đây là kết quả mô phỏng quá trình suất dữ liệu ra mà hiển thị VGA



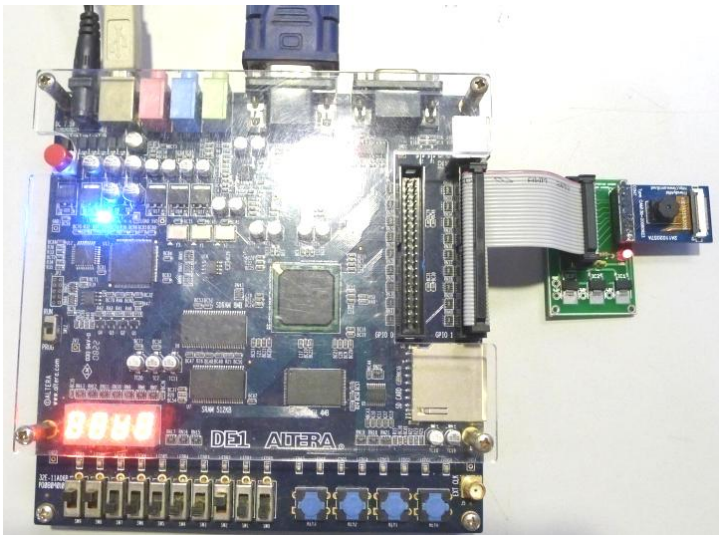
Hình 4.18: Sơ đồ thời gian điều khiển một khung hình 640 x480



Hình 4.19: Sơ đồ khối thời gian điều khiển một dòng

4.2 Kết quả thực hiện hệ thống

Quá trình nghiên cứu thực hiện hệ thống thu và hiển thu ảnh theo các hướng nghiên cứu và phương pháp thực hiện và mục tiêu đề đã đề ra. Việc nghiên cứu đã có được những thiết kế phần cứng hệ thống như hình ảnh dưới đây đây:



Hình 4.20: Phần cứng tổng quát hệ thống

Trong thiết kế phần cứng hệ thống có 3 khối phần cứng chính: khối phần cứng thu thập dữ liệu hình ảnh, khối phần cứng Kit phát triển DE1 và khối màn hình hiển thị VGA.

Để thực hiện hệ thống trên chip FPGA EP2C20F484C7 cần sử dụng tài nguyên trên chip nhất định:

- Tổng số phần tử Logic: 4,047 / 18,752 (22%)
    + Tổng số các chức năng tổ hợp: 3,866 / 18,752 (21 %)
    + Thanh ghi logic chuyên dụng: 527 / 18,752 (3 %)
- Tổng số thanh ghi: 527
- Tổng số chân: 81 / 315 (26 %)
- Tổng số chân ảo: 0
- Tổng số bit bộ nhớ: 0 / 239,616 (0 %)
- Hệ số nhúng thành phần 9-bit: 16 / 52 (31 %)
- Tổng số PLLs: 1 / 4 (25 %)

❖ Trong đó khối điều khiển camera sử dụng số tài nguyên như sau:

- Tổng số phần tử Logic: 3,708 / 18,752 (20%)
    + Tổng số các chức năng tổ hợp: 3,471/ 18,752 (18.5 %)
    + Thanh ghi logic chuyên dụng: 237 / 18,752 (1.5 %)
- Tổng số thanh ghi: 237
- Tổng số chân ảo: 0
- Tổng số bit bộ nhớ: 0 / 239,616 (0 %)
- Hệ số nhúng thành phần 9-bit: 16 / 52 (31 %)
- Tổng số PLLs: 0 / 4 (0 %)

❖ Khối điều khiển SDRAM tiêu tốn tài nguyên:

- Tổng số phần tử Logic: 201 / 18,752 (1.1%)
    + Tổng số các chức năng tổ hợp: 187/ 18,752 (1 %)
    + Thanh ghi logic chuyên dụng: 14 / 18,752 (0.1 %)
- Tổng số thanh ghi: 14
- Tổng số chân ảo: 0
- Tổng số bit bộ nhớ: 0 / 239,616 (0 %)
- Hệ số nhúng thành phần 9-bit: 0 / 52 (0 %)
- Tổng số PLLs: 1 / 4 (25 %)

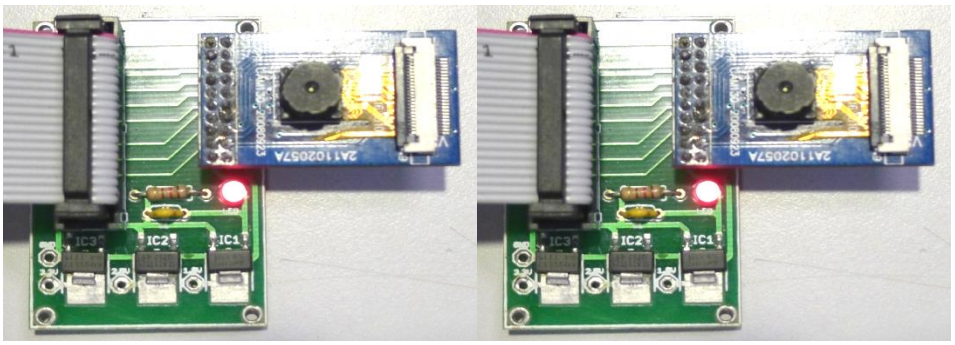
❖ Khối điều khiển VGA tiêu tốn tài nguyên:

- Tổng số phần tử Logic: 58 / 18,752 (0.3%)

+ Tổng số các chức năng tổ hợp: 37/ 18,752 (0.2 %)
+ Thanh ghi logic chuyên dụng: 21 / 18,752 (0.1 %)

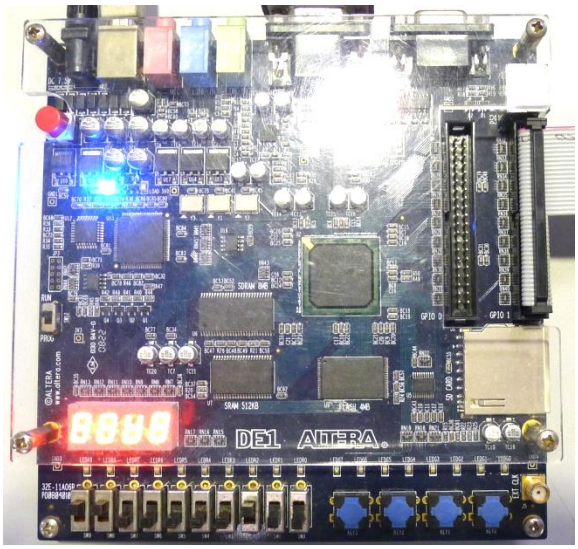
- Tổng số thanh ghi: 21
- Tổng số chân ảo: 0
- Tổng số bit bộ nhớ: 0 / 239,616 (0 %)
- Hệ số nhúng thành phần 9-bit: 0 / 52 (0 %)
- Tổng số PLLs: 0 / 4 (0 %)

Phần cứng giao tiếp khối thu thập hình ảnh từ camera CMOS OV9650 bao gồm module camera OV9650 và module giao diện kết nối giữa module camera và Kit DE1.



Hình 4.21: Phần cứng khối thu thập hình ảnh

Khối nhận và và xử lý dữ liệu hình ảnh FPGA và bộ lưu trữ dữ liệu SDRAM được tích hợp trên Kit phát triển FPGA DE1 thể hiển trên hình 4.22:



Hình 4.22: Kit phát triển FPGA DE1

Khối hiển thị hình ảnh được sử dụng trực tiếp màn hình hiển thị cho máy tính PC để hiện thị hình ảnh.

Hình 4.23: Màn hình hiển thi ảnh

Từ các thiết kế phần cứng và chương trình mô tả phần cứng cho FPGA viết dưới dạng ngôn ngữ VHDL bởi phần mềm Quartus II, hệ thống đã thực hiện được đúng chức năng của hệ thống thu và hiển thị ảnh trên nền FPGA, kết quả hình ảnh thu được từ camera được hiển thị như hình 4.24:

Hình 4.24: Hình ảnh thu được từ camera OV9650

Kết quả thu được từ thiết kế hệ thông thu và hiển thị ảnh có một số đặc điểm như sau:

- Hình ảnh thu được từ bộ cảm biến ảnh CMOS được xử lý bởi FPGA với độ chính xác cao.
- Ảnh video hiển thị liên tục, tốc độ hiển thị cao, video hình ảnh không bị rung.

- Tuy nhiên ảnh thu được có một số điểm ảnh có hệ số màu không chính xác do nhiễu gây ra.

4.3 Tóm tắt chương

Chương này thảo luận về quá trình thực hiện hệ thống thu hình ảnh với các module chính bao gồm: module cảm biến ảnh CMOS, module điều khiển SDRAM đọc và ghi dữ liệu hình ảnh, module điều khiển hiển thị ảnh. Những module này được phân tích chi tiết bởi phần mềm Quartus II để hoàn thành việc mô phỏng và và phân tích các kết quả mô phỏng. Sau đó thực hiện FPGA của hệ thống, đặc điểm hoạt động của hệ thống và kết quả của hệ thống thực hiện trên FPGA.

KẾT LUẬN VÀ KIẾN NGHỊ

1. Kết luận

Thiết kế của hệ thống thu thập và hiển thị hình ảnh thời gian thực trên nền FPGA, hệ thống bao gồm, khối nhận ảnh, khối bộ nhớ hệ thống, khối màn hình hiển thị hình ảnh, khối giao diện ngoại vi, khối FPGA. Bài luận văn viết về việc lựa chọn các chip xử lý cho các khối khác nhau, thiết kế mạch phần cứng và mã chương trình các khối hệ thống, bao gồm: khối nhận ảnh, lưu trữ video, màn hình hiển thị hình ảnh, khối phần cứng ngoại vi và FPGA. Bài viết này hoàn thành các công việc sau đây:

(1) Phân tích hệ thống thu thập hình ảnh và tình trạng các nghiên cứu phát triển trong và ngoài nước; so sánh FPGA, CPLD, ASIC, và đặc điểm khác của thiết bị, thảo luận về các thiết bị FPGA các lựa chọn như lợi thế của các chip cho hệ thống thu và xử lý dữ liệu ảnh.

(2) Phân tích sự phát triển của hệ thống thu thập hình ảnh, và đưa ra kiến trúc hệ thống thu thập hình ảnh, phân tích ưu và nhược điểm, và đề xuất cấu trúc tổng thể của hệ thống thu nhận ảnh tốc độ cao dựa trên FPGA.

(3) Chức năng của các khối và giới thiệu thiết kế phần cứng hệ thộng thu hình ảnh tốc độ cao.

(4) Mô tả quá trình thiết kế hệ thống trên phần mềm Quartus II.

(5) Mô phỏng và phân tích kết quả các khối chức năng trong hệ thống qua ngôn ngữ mô tả phần cứng FPGA.

(6) Sử dụng phần mềm cấu hình cho phần cứng thực hiện hệ thống thu và hiển thị ảnh và đánh giá kết quả đạt được.

Dựa trên những công việc đã nghiên cứu tác giả đã thu được những kết quả nhất định trong hệ thống thu thập và hiển thị hình ảnh như sau:

(1) Hệ thống thu và hiển thị hình ảnh trên nền FPGA hoạt động ổn định

(2) Hình ảnh thu được đạt được độ nét và tính trung thực của ảnh tương ứng với khả năng xử lý ảnh của cảm biến ảnh CMOS OV9650.

(3) Hệ thống thu và hiển thị ảnh đạt được với quá trình xử lý tốc độ cao.

2. Kiến nghị

Hệ thống thu thập và hiển thị hình ảnh trên nền FPGA đã đạt tốc độ cao, hiệu suất thu thập dữ liệu cao và khả năng xử lý tốt hơn. Để có hệ thống có tốc độ xử lý nhanh hơn, đáng tin cậy, hình ảnh thu được có chất lượng cao hơn cần có những cải tiến sau đây:

(1) Cải tiến thiết kế FPGA của hệ thống, để có một thiết bị FPGA mạnh mẽ hơn cải thiện tốc độ và chức năng của hệ thống đang chạy.

(2) Hệ thống có thể đơn giản hóa cấu trúc mạch phần cứng của hệ thống, để có một thiết kế tiên tiến hơn.

(3) Thiết kế thuật toán xử lý dữ liệu hình ảnh chính xác để đạt được chất lượng ảnh tốt hơn.

TÀI LIỆU THAM KHẢO

[1] Bei Yan, Yuefeng Sun, Fengfeng Ding, Haiwen Yuan (2011), *Design of CMOS Image Acquisition System Based on FPGA*. IEEE Conference on Industrial Electronics and Applications, pp.1726-1730

[2] Chao Li, Yu-lin Zhang, and Zhao-na Zheng (2009), *FPGA-Based CMOS Image Acquisition System*, FCC 2009, CCIS 34, pp. 122–127.

[3] D. Vanden Bout (2004), *VGA Generator for the XSA Boards*, XESS Corporation

[4] Fen Xu, Jian-Jun Zeng, Yun-Long Zhang (2008), *Design of a DSP-based CMOS Imaging System for Embedded Computer Vision*. Cybernetics and Intelligent Systems, IEEE conference on, pp. 430-433.

[5] Gerald C. Holst (1998), *CCD arrays, cameras, and displays*.

[6] Gerald C. Holst, Terrence S. Lomheim (2007), *CMOS/CCD Sensors and Camera Systems*.

[7] J. Rose, A. El Gamal, A. Sangiovanni – Vincentelli (1993), *Architecture of Field-Programmable Gate Arrays*, in Proceedings of the IEEE, Vol. 81, No. pp. 1013-1029.

[8] P. Chalimbaud F. Berry (2004), *Design of an Imaging System based on FPGA Technology and CMOS Imager*. IEEE Field Programmable Technology,.

[9] Pong P. Chu (2008), *FPGA Prototyping by VHDL Examples*. Xilinx Spartan-3 Version.

[10] S. Brown, R. Francis, J. Rose, Z (1992), Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers.

[11] Richard Munden (2004), *ASIC and FPGA Verification*.

[12] Volnei A. Pedroni (2004), *Circuit design with VHDL*.

[13] Volnei A. Pedroni (2010), *Circuit Design and Simulation with VHDL*, 2nd edition, Solutions to Selected Exercises.

[14] Vanden Bout (2005), *Dualport Module for the SDRAM Controller*, XESS Corporation.

[15] Vanden Bout (2005), *XSA Board SDRAM Controller, XESS Corporation*.

[16] Volnei A. Pedroni (2004), *Circuit Design with VHDL, MIT Press Cambridge*, Massachusetts London, England, 2004.

[17] Altera Corporation, *SDR SDRAM Controller*, August 2002 ver. 1.1

[18] *Development and Education Board DE1*, V1.1, 2006

[19] *OmniVision serial camera control bus*,version 2.2, 25 june 2007

[20] Omni Vision Technologies, Inc. *OV9620 Data Sheet* (Version 2.5). September

[21] *OV9650/OV9653 Camera Module Software Application Notes*, Revision:1.03, Dec 12th, 2007.

[22] *RTL – to – Gates Synthesis using Synopsys Design Compiler*, 2008

[23] *Synopsys FPGA Synthesis Synplify Pro for Lattice* – November 2012

[24] *Synplicity FPGA Synthesis*. Fpga User Guide, December 2005

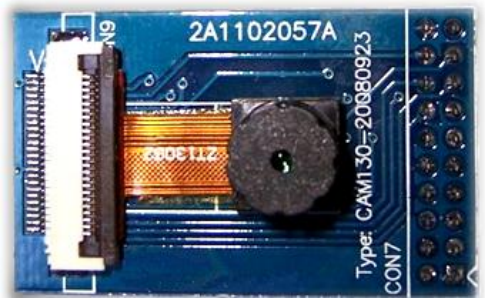
[25] *The Rediff Interview/Steven J Sasson, inventor of the digital camera*

[26] Zentel Electronics Corp, *A2V64S40CTP Datasheet*. Revision 2.1, Sep., 2008

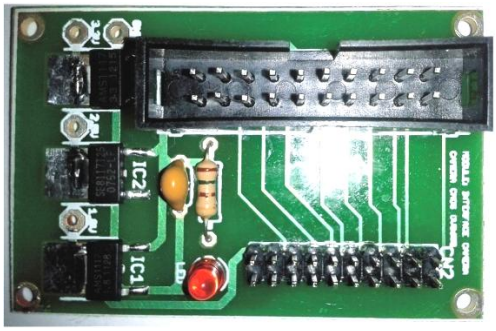
[27] Tống Văn On (2007), *Thiết Kế Mạch Số Với VHDL Và Verilog* - Tập 1, Tập 2. Nxb Lao động Xã hội

PHỤ LỤC 1: CÁC KHỐI THÀNH PHẦN

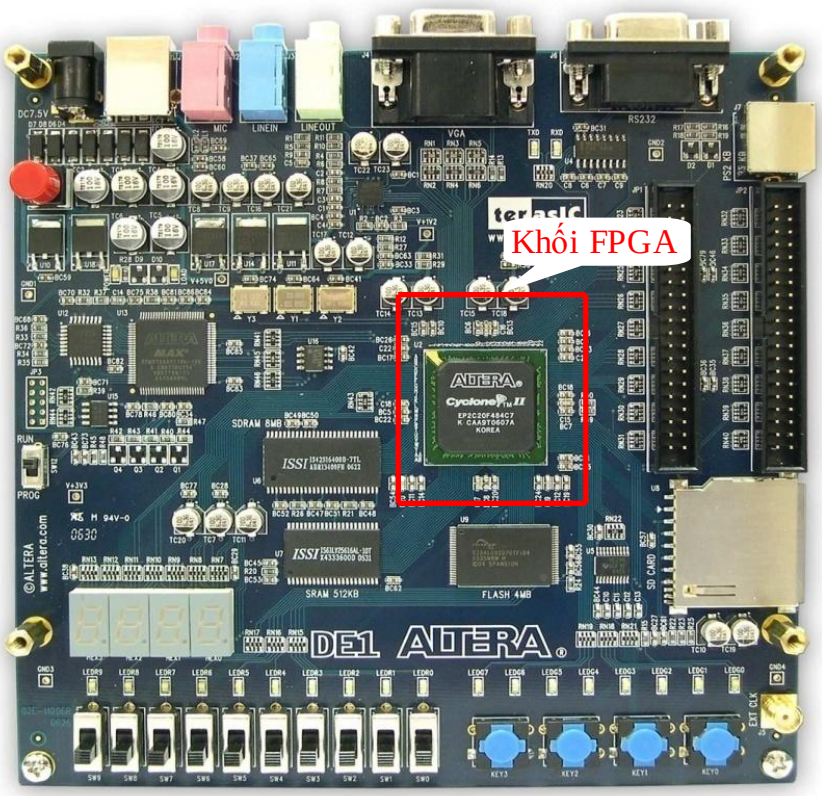
Khối cảm biến hình ảnh OV9650



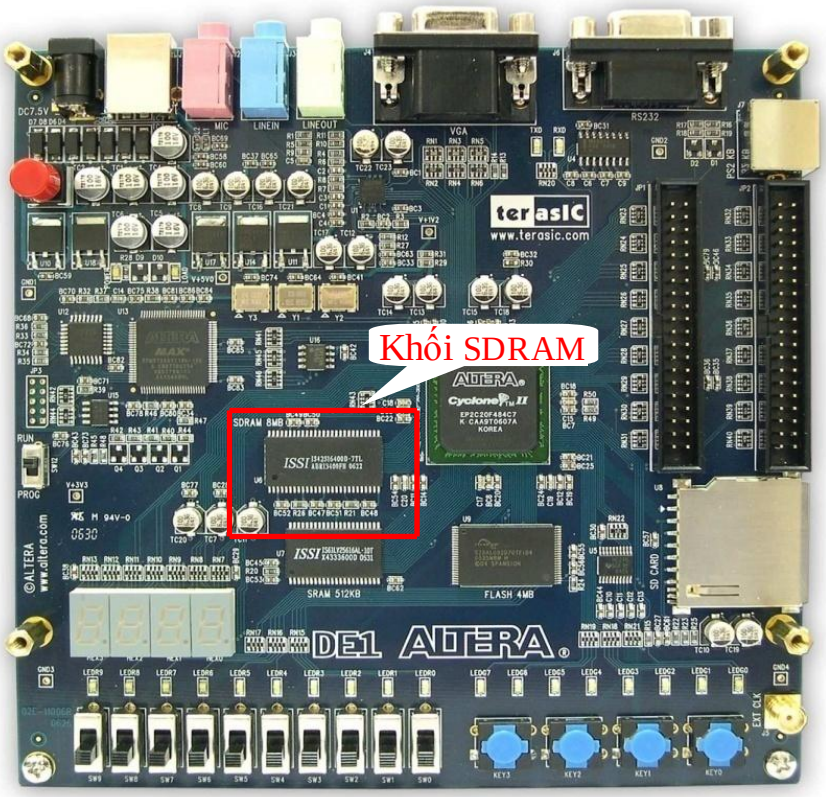
Khối giao diện kết nối cảm biến OV9650 với Kit DE1



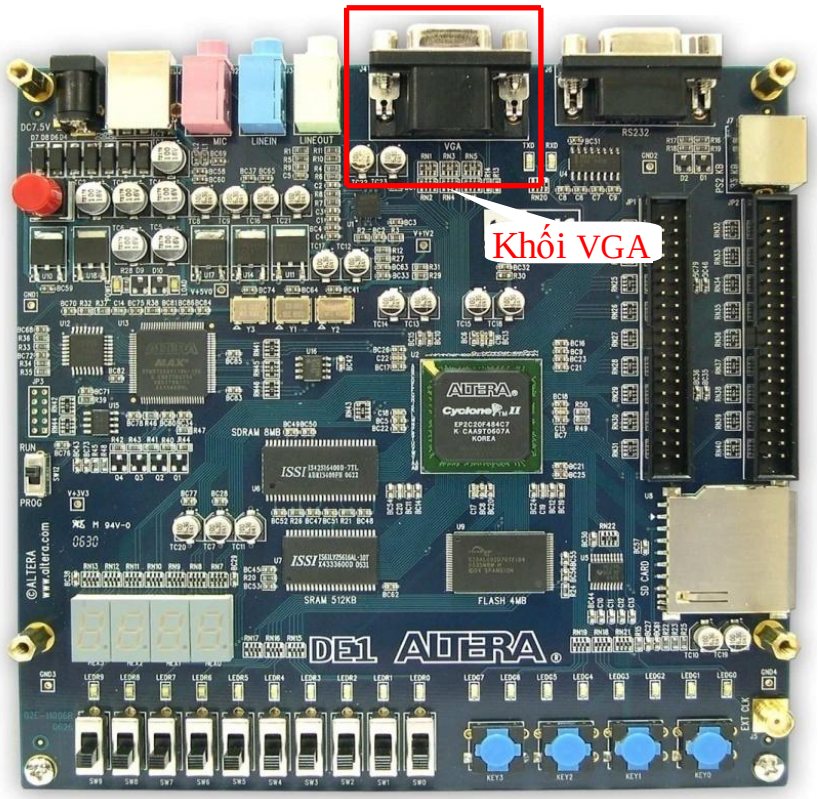
Khối xử lý trung tâm – Kit DE1



Khối lưu trữ dữ liệu SDRAM – Kit DE1



Khối giao diện VGA – Kit DE1



PHỤ LỤC 2: MÃ CHƯƠNG TRÌNH ĐIỀU KHIỂN HỆ THỐNG THU VÀ HIỂN THỊ ẢNH

Thành phần tín hiệu vào ra trong chương trình

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------------------------------------
ENTITY sdram_vga IS
   generic (
      ASIZE          : integer := 23;
      DSIZE          : integer := 16;
      ROWSIZE        : integer := 12;
      COLSIZE        : integer := 9;
      BANKSIZE       : integer := 2;
      ROWSTART       : integer := 9;
      COLSTART       : integer := 0;
      BANKSTART      : integer := 20
   );

PORT (
      CLK            : IN      STD_LOGIC;
      CLK_24         : IN      STD_LOGIC;
      SW             : IN      STD_LOGIC_VECTOR (1 DOWNTO 0);
      SW_AV          : IN      STD_LOGIC_VECTOR (2 DOWNTO 0);
      RST_N          : IN      STD_LOGIC;
      AUTO           : IN      STD_LOGIC;
-- Port in, out VGA ---
      VGA_HSYNC      : OUT     STD_LOGIC;
      VGA_VSYNC      : OUT     STD_LOGIC;
      VGA_DATA_R     : OUT     STD_LOGIC_VECTOR ( 3 DOWNTO 0);
      VGA_DATA_G     : OUT     STD_LOGIC_VECTOR ( 3 DOWNTO 0);
      VGA_DATA_B     : OUT     STD_LOGIC_VECTOR ( 3 DOWNTO 0);
-- Port in, out SDRAM ---
      SDRAM_CLK      : OUT     STD_LOGIC;
      SDRAM_SA       : OUT     STD_LOGIC_VECTOR (11 DOWNTO 0);
      SDRAM_BA       : OUT     STD_LOGIC_VECTOR (1 DOWNTO 0);
      SDRAM_CS_N     : OUT     STD_LOGIC_VECTOR (1 DOWNTO 0);
      SDRAM_CKE      : OUT     STD_LOGIC;
      SDRAM_RAS_N    : OUT     STD_LOGIC;
      SDRAM_CAS_N    : OUT     STD_LOGIC;
      SDRAM_WE_N     : OUT     STD_LOGIC;          --SDRAM write enable
      SDRAM_DQ       : INOUT   STD_LOGIC_VECTOR (DSIZE-1 DOWNTO 0);
      SDRAM_DQM      : OUT     STD_LOGIC_VECTOR (DSIZE/8-1 DOWNTO 0);
      SDRAM_CMD_ACK  : OUT     STD_LOGIC;
-- Port in, out CAMERA ---
      CAM_RW         : IN      STD_LOGIC_VECTOR (1 DOWNTO 0);
      CAM_PCLK       : IN      STD_LOGIC;
```

```
      CAM_HREF    : IN        STD_LOGIC;
      CAM_VSYNC   : IN        STD_LOGIC;
      CAM_DATA    : IN        STD_LOGIC_VECTOR (7 DOWNTO 0);
      CAM_RST     : OUT       STD_LOGIC;
      CAM_XCLKI   : OUT       STD_LOGIC;
      CAM_PWDN    : OUT       STD_LOGIC;
      CAM_SIO_C   : OUT       STD_LOGIC;
      CAM_SIO_D   : INOUT     STD_LOGIC
        );
END sdram_vga;
-----------------------------------------------------------------
ARCHITECTURE Behavior OF sdram_vga IS
 -- signal declarations
 -- signal sdram --
      signal    reset_sdr           :     std_logic;
      signal    ack_ddr             :     std_logic;
      signal    cmd_ddr             :     std_logic_vector (2 downto 0);
      signal    dm_ddr              :     std_logic_vector (DSIZE/8-1 downto 0);
      signal    add_ddr             :     std_logic_vector (ASIZE-1 downto 0);
      signal    dataout_ddr         :     std_logic_vector (DSIZE-1 downto 0);
-- signal VGA --
      signal    pixel_x_vga         :     std_logic_vector (9 downto 0);
      signal    pixel_y_vga         :     std_logic_vector (9 downto 0);
-- signal control sdram --
      signal    add_ex_sdram        :     std_logic_vector (2 downto 0);
       signal   data_write          :     std_logic_vector (DSIZE-1 downto 0);
      signal    DO_READ             :     std_logic;
      signal    DO_WRITE            :     std_logic;
      signal    FINISH_READ         :     std_logic;
      signal    FINISH_WRITE        :     std_logic;
      signal    ENABLE_READ         :     std_logic;
      signal    ENABLE_WRITE        :     std_logic;
 -- signal control camera --
      signal    data_cam            :     std_logic_vector (DSIZE-1 downto 0);
      signal    ox_cam              :     std_logic_vector (9 downto 0);
      signal    oy_cam              :     std_logic_vector (9 downto 0);
      signal    data_sccb           :     std_logic_vector (7 downto 0);
-----------------------------------------------------------------
COMPONENT control_VGA
        PORT (
            CLK_IN              : IN    STD_LOGIC;
            DO_READ             : out   std_logic;
            ENABLE_READ         : in    std_logic;
            FINISH_READ         : out   std_logic;
            DATA_GRB_VGA        : IN    STD_LOGIC_VECTOR (15 DOWNTO 0);
            PIXEL_X_VGA         : OUT   STD_LOGIC_VECTOR (9 DOWNTO 0);
            PIXEL_Y_VGA         : OUT   STD_LOGIC_VECTOR (9 DOWNTO 0);
            VGA_HSYNC           : OUT   STD_LOGIC;
```

```
            VGA_VSYNC           : OUT  STD_LOGIC;
            VGA_DATA_R          : OUT   STD_LOGIC_VECTOR ( 3 DOWNTO 0);
            VGA_DATA_G          : OUT  STD_LOGIC_VECTOR ( 3 DOWNTO 0);
            VGA_DATA_B          : OUT  STD_LOGIC_VECTOR ( 3 DOWNTO 0)
            );
END COMPONENT control_VGA;
-------------------------------------------------------------
COMPONENT sdr_sdram
        generic (
                ASIZE           : integer := 23;
                DSIZE           : integer := 16;
                ROWSIZE         : integer := 12;
                COLSIZE         : integer := 9;
                BANKSIZE        : integer := 2;
                ROWSTART        : integer := 9;
                COLSTART        : integer := 0;
                BANKSTART       : integer := 20
                );
        PORT (
            CLK             : in    std_logic;
            CLK_O           : out   std_logic;
            RESET_N         : in    std_logic;
            ADDR            : in    std_logic_vector(ASIZE-1 downto 0);
            CMD             : in    std_logic_vector(2 downto 0);
            CMDACK          : out   std_logic;
            DATAIN          : in    std_logic_vector(DSIZE-1 downto 0);
            DATAOUT         : out   std_logic_vector(DSIZE-1 downto 0);
            DM              : in    std_logic_vector(DSIZE/8-1 downto 0);
            SA              : out   std_logic_vector(11 downto 0);
            BA              : out   std_logic_vector(1 downto 0);
            CS_N            : out   std_logic_vector(1 downto 0);
            CKE             : out   std_logic;
            RAS_N           : out   std_logic;
            CAS_N           : out   std_logic;
            WE_N            : out   std_logic;
            DQ              : inout std_logic_vector(DSIZE-1 downto 0);
            DQM             : out   std_logic_vector(DSIZE/8-1 downto 0)
                );
END COMPONENT sdr_sdram;
-------------------------------------------------------------
COMPONENT control_RW_SDRAM
        generic (
                DSIZE       :       integer := 16;
                ASIZE       :       integer := 23
                );
        PORT (
             CLK            : in    std_logic;
             RST_N          : in    std_logic;
```

```
            AUTO            : in    std_logic;
            DO_READ         : in    std_logic;
            DO_WRITE        : in    std_logic;
            FINISH_READ     : in    std_logic;
            FINISH_WRITE    : in    std_logic;
            ENABLE_READ     : out   std_logic;
            ENABLE_WRITE    : out   std_logic;
            SW_AV           : in    std_logic_vector (2 downto 0);
            ADD_READ        : in    std_logic_vector (ASIZE-1 downto 0);
            ADD_Write       : in    std_logic_vector (ASIZE-1 downto 0);
            DATA_IN         : in    std_logic_vector (DSIZE-1 downto 0);
            ACK_DDR         : in    std_logic;
            CMD_DDR         : out   std_logic_vector (2 downto 0);
            DM_DDR          : out   std_logic_vector (1 downto 0);
            ADD_DDR         : out   std_logic_vector (ASIZE-1 downto 0);
            DATA_WRITE      : out   std_logic_vector(DSIZE-1 downto 0)
              );
END COMPONENT control_RW_SDRAM;
-------------------------------------------------------------
COMPONENT Camera
      PORT (
            RST_N               : IN    STD_LOGIC;
            CLK_50              : IN    STD_LOGIC;
            CLK_24              : IN    STD_LOGIC;
            DO_WRITE            : OUT   STD_LOGIC;
            FINISH_WRITE        : OUT   STD_LOGIC;
            ENABLE_WRITE        : IN    STD_LOGIC;
            WRITE_SCCB          : IN    STD_LOGIC;          -- Wrie SCCB
            READ_SCCB           : IN    STD_LOGIC;          -- Read SCCB
            DATA_SCCB           : OUT   STD_LOGIC_VECTOR (7 DOWNTO 0);
            CAM_PCLK            : IN    STD_LOGIC;
            CAM_HREF            : IN    STD_LOGIC;
            CAM_VSYNC           : IN    STD_LOGIC;
            CAM_DATA            : IN    STD_LOGIC_VECTOR (7 DOWNTO 0);
            CAM_RST             : OUT   STD_LOGIC;
            CAM_XCLKI           : OUT   STD_LOGIC;
            CAM_PWDN            : OUT   STD_LOGIC;
            CAM_SIO_C           : OUT   STD_LOGIC;
            CAM_SIO_D           : INOUT STD_LOGIC;
            CAM_DATA_RGB        : OUT   STD_LOGIC_VECTOR (15 DOWNTO 0);
            CAM_OX              : OUT   STD_LOGIC_VECTOR (9 DOWNTO 0);
            CAM_OY              : OUT   STD_LOGIC_VECTOR (9 DOWNTO 0)
              );
END COMPONENT Camera;
-------------------------------------------------------------
begin
-- instantiate the control interface module
ctl_VGA1 : control_VGA
```

```
        port map (
            CLK_IN              =>      CLK,
            DO_READ             =>      DO_READ,
            FINISH_READ         =>      FINISH_READ,
            ENABLE_READ         =>      ENABLE_READ,
            DATA_GRB_VGA        =>      dataout_ddr,
            PIXEL_X_VGA         =>      pixel_x_vga,
            PIXEL_Y_VGA         =>      pixel_y_vga,
            VGA_HSYNC           =>      VGA_HSYNC,
            VGA_VSYNC           =>      VGA_VSYNC,
            VGA_DATA_R          =>      VGA_DATA_R,
            VGA_DATA_G          =>      VGA_DATA_G,
            VGA_DATA_B          =>      VGA_DATA_B
               );
-------------------------------------------------------------------
sdr_sdram1 : sdr_sdram
        generic map (
            ASIZE           =>      ASIZE,
            DSIZE           =>      DSIZE,
            ROWSIZE         =>       ROWSIZE,
            COLSIZE         =>      COLSIZE,
            BANKSIZE        =>      BANKSIZE,
            ROWSTART        =>      ROWSTART,
            COLSTART        =>      COLSTART,
            BANKSTART       =>      BANKSTART
                    );
        port map (
            CLK             =>      CLK,
            CLK_O           =>      SDRAM_CLK,
            RESET_N         =>      RST_N,
            ADDR            =>      add_ddr,
            CMD             =>      cmd_ddr,
            DATAIN          =>      data_write,
            DATAOUT         =>      dataout_ddr,
            DM              =>      dm_ddr,
            CMDACK          =>      ack_ddr,
            SA              =>      SDRAM_SA,
            BA              =>      SDRAM_BA,
            CS_N            =>      SDRAM_CS_N,
            CKE             =>      SDRAM_CKE,
            RAS_N           =>      SDRAM_RAS_N,
            CAS_N           =>      SDRAM_CAS_N,
            WE_N            =>      SDRAM_WE_N,
            DQ              =>      SDRAM_DQ,
            DQM             =>      SDRAM_DQM
               );
-------------------------------------------------------------------
ctl_SDRAM : control_RW_SDRAM
```

```
        generic map (
                    DSIZE      => DSIZE,
                    ASIZE      => ASIZE
                    )
                  PORT map (
                CLK                         =>      CLK,
                RST_N                       =>      RST_N,
                AUTO                        =>      AUTO,
                DO_READ                     =>      DO_READ,
                DO_WRITE                    =>      DO_WRITE,
                FINISH_READ                 =>      FINISH_READ,
                FINISH_WRITE                =>      FINISH_WRITE,
                ENABLE_READ                 =>      ENABLE_READ,
                ENABLE_WRITE                =>      ENABLE_WRITE,
                SW_AV                       =>      SW_AV,
                ADD_READ(7 downto 0)        =>      pixel_x_vga(7 downto 0),
                ADD_READ(12 downto 11)      =>      pixel_x_vga(9 downto 8),
                ADD_READ(22 downto 13)      =>      pixel_y_vga,
                ADD_WRITE(7 downto 0)       =>      ox_cam(7 downto 0),
                ADD_WRITE(12 downto 11)     =>      ox_cam(9 downto 8),
                ADD_WRITE(22 downto 13)     =>      oy_cam,
                DATA_IN                     =>      data_cam,
                ACK_DDR                     =>      ack_ddr,
                CMD_DDR                     =>      cmd_ddr,
                DM_DDR                      =>      dm_ddr,
                ADD_DDR                     =>      add_ddr,
                DATA_WRITE                  =>      data_write
                  );
SDRAM_CMD_ACK <= ack_ddr;
Camera1 : Camera
        PORT map (
                CLK_50                      =>      CLK,
                CLK_24                      =>      CLK_24,
                RST_N                       =>      RST_N,
                DO_WRITE                    =>      DO_WRITE,
                FINISH_WRITE                =>      FINISH_WRITE,
                ENABLE_WRITE                =>      ENABLE_WRITE,
                WRITE_SCCB                  =>      CAM_RW(1),
                READ_SCCB                   =>      CAM_RW(0),
                DATA_SCCB                   =>      data_sccb,
                CAM_PCLK                    =>      CAM_PCLK,
                CAM_HREF                    =>      CAM_HREF,
                CAM_VSYNC                   =>      CAM_VSYNC,
                CAM_DATA                    =>      CAM_DATA,
                CAM_RST                     =>      CAM_RST,
                CAM_XCLKI                   =>      CAM_XCLKI,
                CAM_PWDN                    =>      CAM_PWDN ,
                CAM_SIO_C                   =>      CAM_SIO_C,
```

```
            CAM_SIO_D           =>  CAM_SIO_D,
            CAM_DATA_RGB        =>  data_cam,
            CAM_OX              =>  ox_cam,
            CAM_OY              =>  oy_cam
              );
end Behavior;
```

Chương trình điều khiển Camera

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------------------
ENTITY Camera IS
    PORT (
        RST_N       : IN    STD_LOGIC;      -- Reset system
        CLK_50      : IN    STD_LOGIC;      -- Clock 50Mhz
        CLK_24      : IN    STD_LOGIC;      -- Clock 24Mhz
        DO_WRITE    : out   std_logic;
        FINISH_WRITE    : out   std_logic;
        ENABLE_WRITE    : in    std_logic;
        WRITE_SCCB      : IN    STD_LOGIC; -- Wrie SCCB
        READ_SCCB       : IN    STD_LOGIC; -- Read SCCB
        DATA_SCCB   : OUT       STD_LOGIC_VECTOR (7 DOWNTO 0);
        CAM_PCLK    : IN        STD_LOGIC;      -- Camera Pixel clock
        CAM_HREF    : IN        STD_LOGIC;      -- Camera HREF
        CAM_VSYNC   : IN        STD_LOGIC;      -- Camera VSYNC
        CAM_DATA    : IN        STD_LOGIC_VECTOR (7 DOWNTO 0);
        CAM_RST     : OUT       STD_LOGIC;      -- Camera RST
        CAM_XCLKI   : OUT       STD_LOGIC;      -- Camera input clock
        CAM_PWDN    : OUT       STD_LOGIC;      -- Camera Power down
        CAM_SIO_C   : OUT       STD_LOGIC;      -- Camera SIO_C
        CAM_SIO_D   : INOUT     STD_LOGIC;      -- Camera SIO_D
        CAM_DATA_RGB    : OUT   STD_LOGIC_VECTOR (15 DOWNTO 0);
        CAM_OX      : OUT       STD_LOGIC_VECTOR (9 DOWNTO 0);
        CAM_OY      : OUT   STD_LOGIC_VECTOR (9 DOWNTO 0)
          );
END Camera;

ARCHITECTURE RTL OF Camera  IS
Signal io_I2C_END : STD_LOGIC;
    COMPONENT I2C_AV_Config
    Port (
        o_I2C_END       : out   STD_LOGIC;
        iCLK            : in    STD_LOGIC;
        iRST_N          : in    STD_LOGIC;
        iWRITE_SCCB     : in    STD_LOGIC;
        iREAD_SCCB      : in    STD_LOGIC;
        iCAM_PWDN       : out   STD_LOGIC;
        I2C_SCLK        : out   STD_LOGIC;
```

```
        I2C_SDAT            : inout  STD_LOGIC
          );
    END   COMPONENT;
    COMPONENT DATA_RGB
    Port    (
        iCLK                : in     STD_LOGIC;
        iRST_N              : in     STD_LOGIC;
        iDO_WRITE           : out    std_logic;
        iFINISH_WRITE       : out    std_logic;
        iENABLE_WRITE       : in     std_logic;
        iPCLK               : IN     STD_LOGIC;
        iHREF               : IN     STD_LOGIC;
        iVSYNC              : IN     STD_LOGIC;
        iDATA               : IN     STD_LOGIC_VECTOR (7 DOWNTO 0);
        iRST                : OUT    STD_LOGIC;
        iXCLKI              : OUT    STD_LOGIC;
        iDATA_RGB           : OUT    STD_LOGIC_VECTOR (15 DOWNTO 0);
        iPIXEL_OX           : OUT    STD_LOGIC_VECTOR (9 DOWNTO 0);
        iPIXEL_OY           : OUT    STD_LOGIC_VECTOR (9 DOWNTO 0)
          );
    END   COMPONENT;
BEGIN
    I2C_AV: I2C_AV_Config
    port map (
        o_I2C_END           =>    io_I2C_END,
        iCLK                =>    CLK_50      ,
        iRST_N              =>    RST_N               ,
        iWRITE_SCCB         =>    WRITE_SCCB,
        iREAD_SCCB          =>    READ_SCCB,
        iCAM_PWDN           =>    CAM_PWDN ,
        I2C_SCLK            =>    CAM_SIO_C,
        I2C_SDAT            =>    CAM_SIO_D
          );
    DATA : DATA_RGB
    port map (
        iCLK                =>    CLK_24,
        iRST_N              =>    RST_N,
        iDO_WRITE           =>    DO_WRITE ,
        iFINISH_WRITE       =>    FINISH_WRITE,
        iENABLE_WRITE       =>    ENABLE_WRITE,
        iPCLK               =>    CAM_PCLK,
        iHREF               =>    CAM_HREF,
        iVSYNC              =>    CAM_VSYNC,
        iDATA               =>    CAM_DATA,
        iRST                =>    CAM_RST ,
        iXCLKI              =>    CAM_XCLKI,
        iDATA_RGB           =>    CAM_DATA_RGB,
        iPIXEL_OX           =>    CAM_OX,
```

```
            iPIXEL_OY           =>      CAM_OY
              );
END RTL;
```

Chương trình điều khiển SDRAM

```
library ieee;
use ieee.std_logic_1164.all;
entity sdr_sdram is
   generic (
      ASIZE          : integer := 23;
      DSIZE          : integer := 16;
      ROWSIZE        : integer := 12;
      COLSIZE        : integer := 9;
      BANKSIZE       : integer := 2;
      ROWSTART       : integer := 9;
      COLSTART       : integer := 0;
      BANKSTART      : integer := 20
        );
   Port (
      CLK            : in     std_logic;
      CLK_O          : out    std_logic;
      RESET_N        : in     std_logic;
      ADDR           : in     std_logic_vector(ASIZE-1 downto 0);
      CMD            : in     std_logic_vector(2 downto 0);
      CMDACK         : out    std_logic;
      DATAIN         : in     std_logic_vector(DSIZE-1 downto 0);
      DATAOUT        : out    std_logic_vector(DSIZE-1 downto 0);
      DM             : in     std_logic_vector(DSIZE/8-1 downto 0);
      SA             : out    std_logic_vector(11 downto 0);
      BA             : out    std_logic_vector(1 downto 0);
      CS_N           : out    std_logic_vector(1 downto 0);
      CKE            : out    std_logic;
      RAS_N          : out    std_logic;
      CAS_N          : out    std_logic;
      WE_N           : out    std_logic;
      DQ             : inout  std_logic_vector(DSIZE-1 downto 0);
      DQM            : out    std_logic_vector(DSIZE/8-1 downto 0)
        );
End sdr_sdram;
Architecture RTL of sdr_sdram is
   Component command
      Generic  (
         ASIZE       : integer := 23;
         DSIZE       : integer := 16;
         ROWSIZE     : integer := 12;
         COLSIZE     : integer := 9;
         BANKSIZE    : integer := 2;
         ROWSTART: integer := 9;  -- Starting position of the row address within add
```

```
        COLSTART : integer := 0;  -- Starting position of the column address within add
        BANKSTART: integer := 20-- Starting position of the bank address within add
            );
    Port (
        CLK         : in    std_logic;
        RESET_N     : in    std_logic;
        SADDR       : in    std_logic_vector(ASIZE-1 downto 0);
        NOP         : in    std_logic;              -- Decoded NOP command
        READA       : in    std_logic;              -- Decoded READA command
        WRITEA      : in    std_logic;              -- Decoded WRITEA command
        REFRESH     : in    std_logic;              -- Decoded REFRESH command
        PRECHARGE: in   std_logic;              -- Decoded PRECHARGE command
        LOAD_MODE: in   std_logic;              -- Decoded LOAD_MODE command
        SC_CL       : in    std_logic_vector(1 downto 0);-- Programmed CAS latency
        SC_RC       : in    std_logic_vector(1 downto 0); -- Programmed RC delay
        SC_RRD      : in    std_logic_vector(3 downto 0); -- Programmed RRD delay
        SC_PM       : in    std_logic;              -- programmed Page Mode
        SC_BL       : in    std_logic_vector(3 downto 0); -- Programmed burst length
        REF_REQ     : in    std_logic;              -- Hidden refresh request
        REF_ACK     : out   std_logic;              -- Refresh request acknowledge
        CM_ACK      : out   std_logic;              -- Command acknowledge
        OE          : out   std_logic;              -- OE signal for data path module
        SA          : out   std_logic_vector(11 downto 0); -- SDRAM address
        BA          : out   std_logic_vector(1 downto 0);   -- SDRAM bank address
        CS_N        : out   std_logic_vector(1 downto 0);   -- SDRAM chip selects
        CKE         : out   std_logic;              -- SDRAM clock enable
        RAS_N       : out   std_logic;              -- SDRAM RAS
        CAS_N       : out   std_logic;              -- SDRAM CAS
        WE_N        : out   std_logic               -- SDRAM WE_N
      );
End component;
Component sdr_data_path
    Generic (
        DSIZE : integer := 16
            );
    Port (
      CLK           : in    std_logic;              -- System Clock
      RESET_N       : in    std_logic;              -- System Reset
      OE            : in    std_logic;              -- Data output(to the SDRAM) enable
      DATAIN        : in    std_logic_vector(DSIZE-1 downto 0);
      DM            : in    std_logic_vector(DSIZE/8-1 downto 0);
      DATAOUT       : out   std_logic_vector(DSIZE-1 downto 0);
      DQIN          : in    std_logic_vector(DSIZE-1 downto 0);
      DQOUT         : out   std_logic_vector(DSIZE-1 downto 0);
      DQM           : out   std_logic_vector(DSIZE/8-1 downto 0)
        );
  End component;
Component control_interface
```

```
	Generic (
	   ASIZE : integer := 32
		);
	Port (
	  CLK		: in	std_logic;		-- System Clock
	  RESET_N	: in	std_logic;		-- System Reset
	  CMD		: in	std_logic_vector(2 downto 0);		-- Command input
	  ADDR		: in	std_logic_vector(ASIZE-1 downto 0);-- Address
	  REF_ACK	: in	std_logic;		-- Refresh request acknowledge
	  CM_ACK	: in	std_logic;		-- Command acknowledge
	  NOP		: out	std_logic;		-- Decoded NOP command
	  READA		: out	std_logic;		-- Decoded READA command
	  WRITEA	: out	std_logic;		-- Decoded WRITEA command
	  REFRESH	: out	std_logic;		-- Decoded REFRESH command
	  PRECHARGE	: out	std_logic;		-- Decoded PRECHARGE command
	  LOAD_MODE	: out	std_logic;		-- Decoded LOAD_MODE command
	  SADDR		: out	std_logic_vector(ASIZE-1 downto 0); -- Registered version
of ADDR
	  SC_CL		: out	std_logic_vector(1 downto 0); -- Programmed CAS latency
	  SC_RC		: out	std_logic_vector(1 downto 0); -- Programmed RC delay
	  SC_RRD	: out	std_logic_vector(3 downto 0); -- Programmed RRD delay
	  SC_PM		: out	std_logic;		-- programmed Page Mode
	  SC_BL		: out	std_logic_vector(3 downto 0); -- Programmed burst length
	  REF_REQ	: out	std_logic;		-- Hidden refresh request
	  CMD_ACK	: out	std_logic		-- Command acknowledge
	    );
   End component;
   Attribute syn_black_box: boolean;
	Component pll1
	Port (
	inclk0		: in	std_logic;
	c0		: out	std_logic;
	locked		: out	std_logic
		);
End component;
 Attribute syn_black_box of pll1: component is true;
	-- signal declarations
	signal	ISA		:	std_logic_vector(11 downto 0); --SDRAM address output
	signal	IBA		:	std_logic_vector(1 downto 0);	--SDRAM bank address
	signal	ICS_N		:	std_logic_vector(1 downto 0);	--SDRAM Chip Selects
	signal	ICKE		:	std_logic;		--SDRAM clock enable
	signal	IRAS_N		:	std_logic;		--SDRAM Row address Strobe
	signal	ICAS_N		:	std_logic;		--SDRAM Column address Strobe
	signal	IWE_N		:	std_logic;
	signal	DQIN		:	std_logic_vector(DSIZE-1 downto 0);
	signal	IDATAOUT	:	std_logic_vector(DSIZE-1 downto 0);
	signal	DQOUT		:		std_logic_vector(DSIZE-1	downto	0);
--SDRAM write enable
```

```
    signal  sadd        :  std_logic_vector(ASIZE-1 downto 0);
    signal  sc_cl       :  std_logic_vector(1 downto 0);
    signal  sc_rc       :  std_logic_vector(1 downto 0);
    signal  sc_rrd      :  std_logic_vector(3 downto 0);
    signal  sc_pm       :  std_logic;
    signal  sc_bl       :  std_logic_vector(3 downto 0);
    signal  load_mode   :  std_logic;
    signal  nop         :  std_logic;
    signal  reada       :  std_logic;
    signal  writea      :  std_logic;
    signal  refresh     :  std_logic;
    signal  precharge   :  std_logic;
    signal  oe          :  std_logic;
    signal  ref_req     :  std_logic;
    signal  ref_ack     :  std_logic;
    signal  cm_ack      :  std_logic;
    signal  CLK133      :  std_logic;
    signal  CLK133B     :  std_logic;
    signal  clklocked   :  std_logic;

Begin
        -- instantiate the control interface module
   control1 : control_interface
      Generic map (
              ASIZE => ASIZE
                   );
      Port map  (
        CLK         =>    CLK133,
        RESET_N     =>    RESET_N,
        CMD         =>    CMD,
        ADDR        =>    ADDR,
        REF_ACK     =>    ref_ack,
        CM_ACK      =>    cm_ack,
        NOP         =>    nop,
        READA       =>    reada,
        WRITEA      =>    writea,
        REFRESH     =>    refresh,
        PRECHARGE =>      precharge,
        LOAD_MODE=>       load_mode,
        SADDR       =>    saddr,
        SC_CL       =>    sc_cl,
        SC_RC       =>    sc_rc,
        SC_RRD      =>    sc_rrd,
        SC_PM       =>    sc_pm,
        SC_BL       =>    sc_bl,
        REF_REQ     =>    ref_req,
        CMD_ACK     =>    CMDACK
              );
```

```
    -- instantiate the command module
command1 : command
  Generic map (
    ASIZE       =>   ASIZE,
    DSIZE       =>   DSIZE,
    ROWSIZE     =>   ROWSIZE,
    COLSIZE     =>   COLSIZE,
    BANKSIZE    =>   BANKSIZE,
    ROWSTART    =>   ROWSTART,
    COLSTART    =>   COLSTART,
    BANKSTART=>      BANKSTART
          );
  port map  (
        CLK          =>   CLK133,
        RESET_N      =>   RESET_N,
        SADDR        =>   saddr,
        NOP          =>   nop,
        READA        =>   reada,
        WRITEA       =>   writea,
        REFRESH      =>   refresh,
        PRECHARGE    =>   precharge,
        LOAD_MODE    =>   load_mode,
        SC_CL        =>   sc_cl,
        SC_RC        =>   sc_rc,
        SC_RRD       =>   sc_rrd,
        SC_PM        =>   sc_pm,
        SC_BL        =>   sc_bl,
        REF_REQ      =>   ref_req,
        REF_ACK      =>   ref_ack,
        CM_ACK       =>   cm_ack,
        OE           =>   oe,
        SA           =>   ISA,
        BA           =>   IBA,
        CS_N         =>   ICS_N,
        CKE          =>   ICKE,
        RAS_N        =>   IRAS_N,
        CAS_N        =>   ICAS_N,
        WE_N         =>   IWE_N
         );
    -- instantiate the data path module
data_path1 : sdr_data_path
    Generic map (
                DSIZE => DSIZE
                )
  Port map  (
    CLK     => CLK133,
    RESET_N  => RESET_N,
    OE      => oe,
```

```
        DATAIN    => DATAIN,
        DM        => DM,
        DATAOUT   => IDATAOUT,
        DQM       => DQM,
        DQIN      => DQIN,
        DQOUT     => DQOUT
           );

   pll : pll1
port map (
        inclk0 => CLK,
        locked  => clklocked,
        c0  => CLK133
        );
   -- Add a level flops to the sdram i/o that can be place
   -- by the router into the I/O cells
Process (CLK133)
Begin
        If (rising_edge(CLK133)) then
          SA            <=      ISA;
          BA            <=      IBA;
          CS_N          <=      ICS_N;
          CKE           <=      ICKE;
          RAS_N         <=      IRAS_N;
          CAS_N         <=      ICAS_N;
          WE_N          <=      IWE_N;
          DQIN          <=      DQ;
          DATAOUT       <=      IDATAOUT;
        End if;
End process;
DQ <= DQOUT when OE = '1' else (others => 'Z');
CLK_O <= CLK133;
End RTL;
```

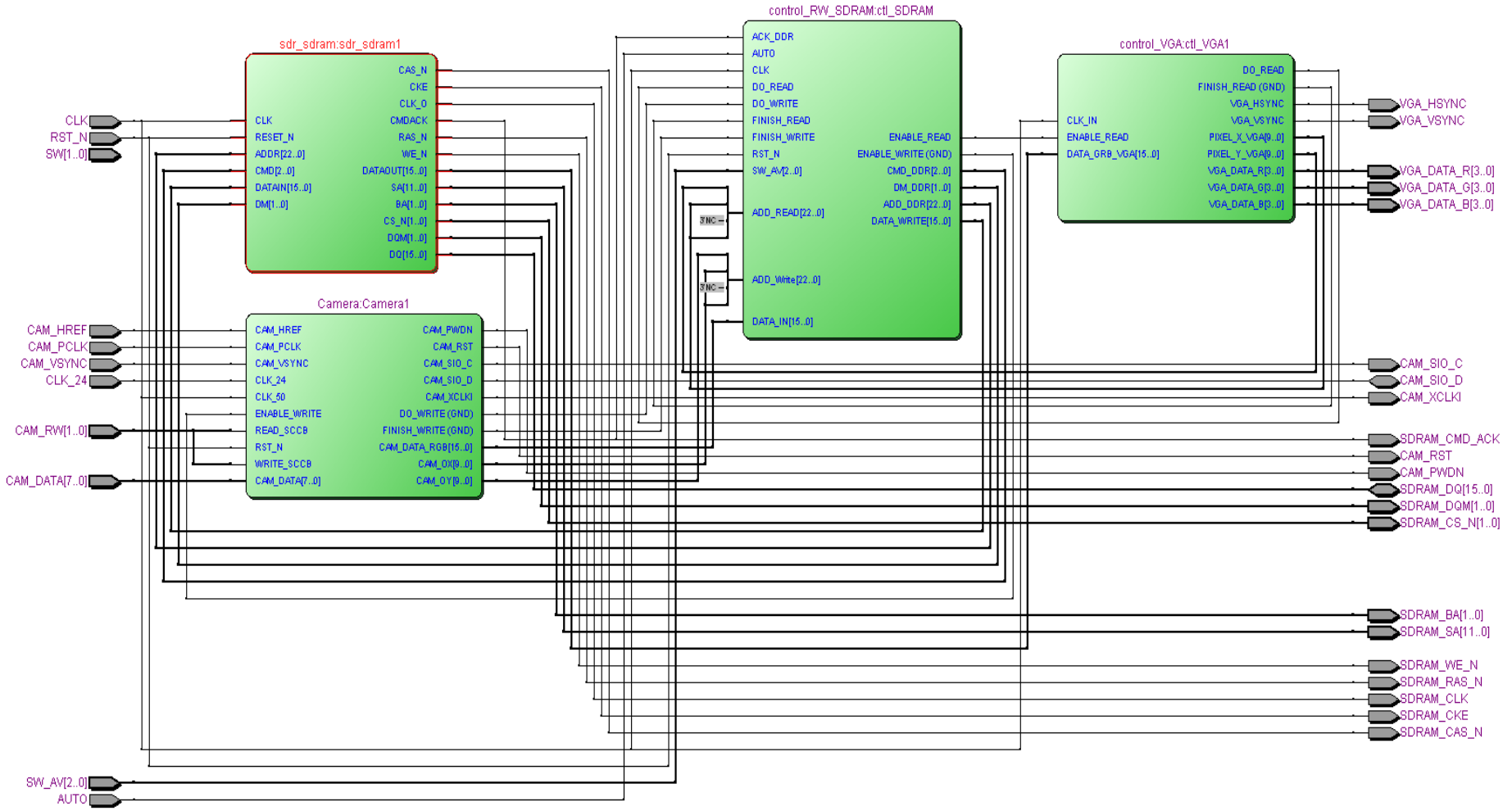
Chương trình hiển thị ảnh ra màn hình

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity control_VGA is
        port (
        CLK_IN              : IN    STD_LOGIC;
        DO_READ             : out   std_logic;
        ENABLE_READ         : in    std_logic;
        FINISH_READ         : out   std_logic;
        DATA_GRB_VGA        : IN    STD_LOGIC_VECTOR (15 DOWNTO 0);
        PIXEL_X_VGA         : OUT STD_LOGIC_VECTOR (9 DOWNTO 0);
        PIXEL_Y_VGA         : OUT  STD_LOGIC_VECTOR (9 DOWNTO 0);
```

```
        VGA_HSYNC       : OUT  STD_LOGIC;
        VGA_VSYNC       : OUT  STD_LOGIC;
        VGA_DATA_R      : OUT   STD_LOGIC_VECTOR ( 3 DOWNTO 0);
        VGA_DATA_G      : OUT  STD_LOGIC_VECTOR ( 3 DOWNTO 0);
        VGA_DATA_B      : OUT  STD_LOGIC_VECTOR ( 3 DOWNTO 0)
                );
end control_VGA;
architecture Behavioral of control_VGA is
    signal clk25                    : std_logic;
    signal horizontal_counter       : std_logic_vector (9 downto 0);
    signal vertical_counter         : std_logic_vector (9 downto 0);
    signal counter                  : std_logic_vector (3 downto 0);
    signal mode                     : std_logic;
    signal counter_frame            : std_logic_vector (2 downto 0);
begin
-- generate a 25Mhz clock
process (CLK_IN)
begin
        if (CLK_IN'event and CLK_IN='1') then
                if (clk25 = '0') then
                        clk25 <= '1';
                 else
                        clk25 <= '0';
                end if;
        end if;
end process;
process (clk25)
begin
        if (clk25'event and clk25 = '1') then
                if (horizontal_counter >= "0010000000")
                        and (horizontal_counter < "1100011111")
                        and (vertical_counter >=  "0000101000")
                        and (vertical_counter < "1000001000") then
                        DO_READ <= mode;
                else
                        DO_READ <= '0';
                end if;
                horizontal_counter <= horizontal_counter + "0000000001";
                if (horizontal_counter >= "0010010000")
                        and (horizontal_counter < "1100010000")
                        and (vertical_counter >=  "0000101000")
                        and (vertical_counter < "1000001000")  then
                        if      (ENABLE_READ = '1') then
                                VGA_DATA_R <= DATA_GRB_VGA(3 downto 0);
                                VGA_DATA_G <= DATA_GRB_VGA(7 downto 4);
                                VGA_DATA_B <= DATA_GRB_VGA(11 downto 8);
                        else
                                VGA_DATA_R <= "0100";
```

```
                        VGA_DATA_G <= "0100";
                        VGA_DATA_B <= "0100";
                end if;
        else
                VGA_DATA_R <= "0000";
                VGA_DATA_G <= "0000";
                VGA_DATA_B <= "0000";
        end if;
        if (horizontal_counter > "0000000000" )
        and (horizontal_counter < "0001100001") then
                VGA_HSYNC <= '0';
        else
                VGA_HSYNC <= '1';
        end if;
        if (vertical_counter > "0000000000" )
        and (vertical_counter < "0000000011" ) then
                VGA_VSYNC <= '0';
        else
                VGA_VSYNC <= '1';
        end if;
  --
        if (horizontal_counter="1100100000") then
                vertical_counter <= vertical_counter+"0000000001";
                horizontal_counter <= "0000000000";
                mode <= mode xor '1';
        end if;
        if (vertical_counter = "1000001001") then
                vertical_counter <= "0000000000";
        end if;
    end if;
End process;
PIXEL_X_VGA <= horizontal_counter - "0010001100";
PIXEL_Y_VGA <= vertical_counter - "0000101000";
End Behavioral;
```

PHỤ LỤC 3: SƠ ĐỒ CẤU TRÚC MÃ CHƯƠNG TRÌNH



PHỤ LỤC 4: BẢNG ĐỊA CHỈ VÀO RA FPGA

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard
AUTO	Input	PIN_M22	6	B6_N0	3.3-V LVTTL (default)
CAM_DATA[7]	Input	PIN_F12	4	B4_N1	3.3-V LVTTL (default)
CAM_DATA[6]	Input	PIN_F13	4	B4_N1	3.3-V LVTTL (default)
CAM_DATA[5]	Input	PIN_C14	4	B4_N0	3.3-V LVTTL (default)
CAM_DATA[4]	Input	PIN_D14	4	B4_N1	3.3-V LVTTL (default)
CAM_DATA[3]	Input	PIN_D15	4	B4_N0	3.3-V LVTTL (default)
CAM_DATA[2]	Input	PIN_D16	4	B4_N0	3.3-V LVTTL (default)
CAM_DATA[1]	Input	PIN_C17	4	B4_N0	3.3-V LVTTL (default)
CAM_DATA[0]	Input	PIN_C18	4	B4_N0	3.3-V LVTTL (default)
CAM_HREF	Input	PIN_E15	4	B4_N0	3.3-V LVTTL (default)
CAM_PCLK	Input	PIN_G16	4	B4_N0	3.3-V LVTTL (default)
CAM_PWDN	Output	PIN_H14	4	B4_N0	3.3-V LVTTL (default)
CAM_RST	Output	PIN_G15	4	B4_N0	3.3-V LVTTL (default)
CAM_RW[1]	Input	PIN_M1	1	B1_N0	3.3-V LVTTL (default)
CAM_RW[0]	Input	PIN_M2	1	B1_N0	3.3-V LVTTL (default)
CAM_SIO_C	Output	PIN_H13	4	B4_N1	3.3-V LVTTL (default)
CAM_SIO_D	Bidir	PIN_H12	4	B4_N1	3.3-V LVTTL (default)
CAM_VSYNC	Input	PIN_F15	4	B4_N0	3.3-V LVTTL (default)
CAM_XCLKI	Output	PIN_E14	4	B4_N1	3.3-V LVTTL (default)
CLK	Input	PIN_L1	2	B2_N1	3.3-V LVTTL (default)
CLK_24	Input	PIN_B12	4	B4_N1	3.3-V LVTTL (default)
RST_N	Input	PIN_L2	2	B2_N1	3.3-V LVTTL (default)
SDRAM_BA[1]	Output	PIN_V4	1	B1_N1	3.3-V LVTTL (default)
SDRAM_BA[0]	Output	PIN_U3	1	B1_N1	3.3-V LVTTL (default)
SDRAM_CAS_N	Output	PIN_T3	1	B1_N1	3.3-V LVTTL (default)
SDRAM_CKE	Output	PIN_N3	1	B1_N0	3.3-V LVTTL (default)

SDRAM_CLK	Output	PIN_U4	1	B1_N1	3.3-V LVTTL (default)
SDRAM_CMD_ACK	Output	PIN_Y21	6	B6_N1	3.3-V LVTTL (default)
SDRAM_CS_N[1]	Output	PIN_E1	2	B2_N1	3.3-V LVTTL (default)
SDRAM_CS_N[0]	Output	PIN_T6	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQ[15]	Bidir	PIN_T2	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQ[14]	Bidir	PIN_T1	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQ[13]	Bidir	PIN_R2	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQ[12]	Bidir	PIN_R1	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQ[11]	Bidir	PIN_P2	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQ[10]	Bidir	PIN_P1	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQ[9]	Bidir	PIN_N2	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQ[8]	Bidir	PIN_N1	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQ[7]	Bidir	PIN_Y2	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQ[6]	Bidir	PIN_Y1	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQ[5]	Bidir	PIN_W2	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQ[4]	Bidir	PIN_W1	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQ[3]	Bidir	PIN_V2	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQ[2]	Bidir	PIN_V1	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQ[1]	Bidir	PIN_U2	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQ[0]	Bidir	PIN_U1	1	B1_N1	3.3-V LVTTL (default)
SDRAM_DQM[1]	Output	PIN_M5	1	B1_N0	3.3-V LVTTL (default)
SDRAM_DQM[0]	Output	PIN_R7	1	B1_N0	3.3-V LVTTL (default)
SDRAM_RAS_N	Output	PIN_T5	1	B1_N1	3.3-V LVTTL (default)
SDRAM_SA[11]	Output	PIN_N6	1	B1_N0	3.3-V LVTTL (default)
SDRAM_SA[10]	Output	PIN_W3	1	B1_N1	3.3-V LVTTL (default)
SDRAM_SA[9]	Output	PIN_N4	1	B1_N0	3.3-V LVTTL (default)
SDRAM_SA[8]	Output	PIN_P3	1	B1_N0	3.3-V LVTTL (default)
SDRAM_SA[7]	Output	PIN_P5	1	B1_N0	3.3-V LVTTL (default)
SDRAM_SA[6]	Output	PIN_P6	1	B1_N0	3.3-V LVTTL (default)
SDRAM_SA[5]	Output	PIN_R5	1	B1_N1	3.3-V LVTTL (default)
SDRAM_SA[4]	Output	PIN_R6	1	B1_N1	3.3-V LVTTL (default)

SDRAM_SA[3]	Output	PIN_Y4	1	B1_N1	3.3-V LVTTL (default)
SDRAM_SA[2]	Output	PIN_Y3	1	B1_N1	3.3-V LVTTL (default)
SDRAM_SA[1]	Output	PIN_W5	1	B1_N1	3.3-V LVTTL (default)
SDRAM_SA[0]	Output	PIN_W4	1	B1_N1	3.3-V LVTTL (default)
SDRAM_WE_N	Output	PIN_R8	1	B1_N0	3.3-V LVTTL (default)
SW[1]	Input	PIN_L21	5	B5_N1	3.3-V LVTTL (default)
SW[0]	Input	PIN_L22	5	B5_N1	3.3-V LVTTL (default)
SW_AV[2]	Input	PIN_U12	8	B8_N0	3.3-V LVTTL (default)
SW_AV[1]	Input	PIN_W12	7	B7_N1	3.3-V LVTTL (default)
SW_AV[0]	Input	PIN_V12	7	B7_N1	3.3-V LVTTL (default)
VGA_DATA_B[3]	Output	PIN_B10	3	B3_N0	3.3-V LVTTL (default)
VGA_DATA_B[2]	Output	PIN_A10	3	B3_N0	3.3-V LVTTL (default)
VGA_DATA_B[1]	Output	PIN_D11	3	B3_N0	3.3-V LVTTL (default)
VGA_DATA_B[0]	Output	PIN_A9	3	B3_N0	3.3-V LVTTL (default)
VGA_DATA_G[3]	Output	PIN_A8	3	B3_N0	3.3-V LVTTL (default)
VGA_DATA_G[2]	Output	PIN_B9	3	B3_N0	3.3-V LVTTL (default)
VGA_DATA_G[1]	Output	PIN_C10	3	B3_N0	3.3-V LVTTL (default)
VGA_DATA_G[0]	Output	PIN_B8	3	B3_N0	3.3-V LVTTL (default)
VGA_DATA_R[3]	Output	PIN_B7	3	B3_N1	3.3-V LVTTL (default)
VGA_DATA_R[2]	Output	PIN_A7	3	B3_N1	3.3-V LVTTL (default)
VGA_DATA_R[1]	Output	PIN_C9	3	B3_N1	3.3-V LVTTL (default)
VGA_DATA_R[0]	Output	PIN_D9	3	B3_N0	3.3-V LVTTL (default)
VGA_HSYNC	Output	PIN_A11	3	B3_N0	3.3-V LVTTL (default)
VGA_VSYNC	Output	PIN_B11	3	B3_N0	3.3-V LVTTL (default)