HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA ĐIỆN TỬ-VIỄN THÔNG

ĐỒ ÁN 1

ĐỀ TÀI: ĐIỀU KHIỂN MÀN HÌNH LCD 1602 BẰNG FPGA

Sinh viên thực hiện: **1. Nguyễn Quốc Tuấn DT010243**

2. Hoàng Trung Kiên DT010221

3.Nguyễn Văn Hiền DT010213

Giảng viên hướng dẫn: **Th.s Dương Phúc Phần**

Hà Nội, 2020

NHIỆM VỤ ĐỒ ÁN

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đồ án 1

.....

.....

.....

.....

.....

.....

Các số liệu cần thiết để thiết kế

.....

.....

.....

.....

.....

.....

GIẢNG VIÊN HƯỚNG DẪN ĐỒ ÁN

Họ và tên: Dương Phúc Phần

Học hàm, học vị: Thạc sĩ (Trưởng khoa bộ môn điện tử máy tính)

Cơ quan công tác: Học viện kỹ thuật Mật Mã

Nội dung hướng dẫn: Điều khiển màn hình LCD 1602 bằng FPGA

Đồ án 1 được giao ngày 1 tháng 8 năm 2020

Yêu cầu phải hoàn thành xong trước ngày 26 tháng 8 năm 2020

Giảng viên hướng dẫn đồ án

.....

PHẦN NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. **Tinh thần của các sinh viên trong quá trình làm đồ án:**

.....

.....

.....

.....

2. **Đánh giá chất lượng của khóa luận(so với yêu cầu đã đề ra trong nhiệm vụ đồ án trên các mặt lý luận thực tiễn, tính toán,...):**

.....

.....

.....

.....

.....

3. **Cho điểm của cán bộ hướng dẫn(ghi cả số và chữ):**

.....

.....

.....

Hà Nội, ngày 26 tháng 8 năm 2020

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

PHẦN NHẬN XÉT TÓM TẮT CỦA NGƯỜI CHẤM PHẢN BIỆN

1. **Đánh giá chất lượng của khóa luận(so với yêu cầu đã đề ra trong nhiệm vụ đồ án trên các mặt lý luận thực tiễn, tính toán,...):**

   .....

   .....

   .....

   .....

   .....

   .....

   .....

2. **Cho điểm của cán bộ hướng dẫn(ghi cả số và chữ):**

   .....

   .....

   .....

   .....

Hà Nội, ngày 26 tháng 8 năm 2020

Người chấm phản biện

LỜI CẢM ƠN

Trong thời gian làm đồ án 1, nhóm chúng em đã nhận được sự giúp đỡ nhiệt tình và đóng góp của thầy cô và các bạn.

Em xin gửi lời cảm ơn chân thành nhất đến Th.S Dương Phúc Phần trưởng khoa điện tử máy tính học viện kỹ thuật Mật Mã. Đã là người hướng dẫn, chỉ bảo cho nhóm chúng em trong suốt thời gian tìm hiểu về đồ án 1 lần nay.

Nhóm chúng em cũng cảm ơn chân thành đến các thầy cô giáo trong trường Học viện kỹ thuật Mật Mã nói chung và các thầy cô giáo trong khoa DTVT nói riêng đã cho chúng em một nền tảng vững về kiến thức đại cương cũng như nền móng về chuyên ngành và cũng tạo điều kiện giúp đỡ chúng em trong suốt quá trình học tập và quá trình làm đồ án này .

Cuối cùng em xin cảm ơn đến gia đình, bạn bè , đã tạo điều kiện quan tâm, giúp đỡ, động viên chúng em trong suốt quá trình hoàn thành làm đồ án 1.

Hà Nôi, ngày 26 tháng 8 năm 2020

MỤC LỤC

LỜI NÓI ĐẦU

Ngày nay với sự phát triển hết sức mạnh mẽ về khoa học, công nghệ ngày càng được đổi mới và sự tối ưu hóa nhằm nâng cao tính hiệu quả của nó. Chính vì sự phát triển mạnh mẽ của công nghệ phần mền, công nghệ phát triển phần cứng cũng ngày càng đòi hỏi cao để có thể đáp ứng và xử lý nhanh được công việc. Các mạch lọc tương tự trước đây không còn đủ khả năng để đáp ứng những yêu cầu đó nữa. Cho nên FPGA đã ra đời, nó như một giải pháp cung cấp môi trường làm việc hiểu quả cho các ứng dụng thực tế. Tính linh động cao trong quá trình thiết kế cho phép FPGA giải quyết được những bài toán khó mà trước đây chỉ thực hiện được nhờ phần mền máy tính. FPGA còn có mật độ các cổng logic nhiều, đáp ứng được những bài toán đòi hỏi khối lượng tính toán lớn và dùng trong các hệ thông làm việc với thời gian thực. Những áp dụng của FPGA nó thể hiện rất rõ trong các ngành: Hàng không, vũ trụ, quốc phòng,… Đặt biệt với khả năng tái lập trình, người sử dụng có thể thay đổi lại thiết kế của mình chỉ trong vài giờ.

Chính vì những điểm mạnh mẽ và những ứng dụng thực tiễn của FPGA vào đời sống, nên nhóm chúng em đã chọn đề tài:” Điều khiển màn hình LCD 1602 bằng FPGA”, giúp bọn em một cái nhìn tổng quan hơn về FPGA, và cách giao tiếp với LCD để từ đó có thể chọn FPGA là một hướng đi phát tiển học tập và công việc về sau.

Bố cục đồ án chia làm 4 chương:

- Chương I. Tổng quan về FPGA
- Chương II. Giới thiệu về KIT SPARTAN-3E và môi trường lập trình ISE 14.7
- Chương II. Tổng quan về LCD
- Chương IV. Điều khiển màn hình LCD 1602 bằng FPGA (KIT SPARTAN-3E).

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

CHƯƠNG I. TỔNG QUAN VỀ FPGA

1.1 Khái niệm về FPGA

Field-programmable gate array (FPGA) là một loại mạch tích hợp cỡ lớn 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,nhưng nếu so sánh FPGA với những ASIC đặc chế hoàn toàn hay ASIC thiết kế trên thư viện logic 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 các điểm:

- Tác vụ tái lập trình của FPGA thực hiện đơn giản hơn
- Khả năng lập trình linh động hơn
- 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 có trước nó

Thiết kế hay lập trình cho FPGA được thực hiện chủ yếu bằng các ngôn ngữ mô tả phần cứng HDL như VHDL, Verilog, AHDL, 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 thứ ba cung cấp các gói phần mềm kiểu này như Synopsys, Synplify... 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).

1.2 Lịch sự ra đời FPGA

FPGA được thiết kế đầu tiên bởi Ross Freeman, người sáng lập công ty Xilinx vào năm 1984, kiến trúc mới của FPGA cho phép tích hợp số lượng tương đối lớn các phần tử bán dẫn vào 1 vi mạch so với kiến trúc trước đó là CPLD. FPGA có khả năng chứa tới từ 100.000 đến hàng vài tỷ cổng logic, trong khi CPLD chỉ chứa từ 10.000 đến 100.000 cổng logic; con số này đối với PAL, PLA còn thấp hơn nữa chỉ đạt vài nghìn đến 10.000.

CPLD được cấu trúc từ số lượng nhất định các khối SPLD (Simple programmable devices, thuật ngữ chung chỉ PAL, PLA). SPLD thường là một mảng logic AND/OR lập trình được có kích thước xác định và chứa một số lượng hạn chế các phần tử nhớ đồng bộ (clocked register). Cấu trúc này hạn chế khả năng thực hiện những hàm phức tạp và thông thường hiệu suất làm việc của vi mạch phụ thuộc vào cấu trúc cụ thể của vi mạch hơn là vào yêu cầu bài toán.

Kiến trúc của FPGA là kiến trúc mảng các khối logic, khối logic, nhỏ hơn nhiều nếu đem so sánh với một khối SPLD, ưu điểm này giúp FPGA có thể chứa nhiều hơn các phần tử logic và phát huy tối đa khả năng lập trình của các phần tử logic và hệ thống mạch kết nối, để đạt được mục đích này thì kiến trúc của FPGA phức tạp hơn nhiều so với CPLD.

Một điểm khác biệt với CPLD là trong những FPGA hiện đại được tích hợp nhiều những bộ logic số học đã sơ bộ tối ưu hóa, hỗ trợ RAM, ROM, tốc độ cao, hay các bộ nhân cộng (multiplication and accumulation, MAC), thuật ngữ tiếng Anh là DSP slice dùng cho những ứng dụng xử lý tín hiệu số DSP.

Ngoài khả năng tái cấu trúc vi mạch toàn cục, một số FPGA hiện đại còn hỗ trợ tái cấu trúc cục bộ, tức là khả năng tái cấu trúc một bộ phận riêng lẻ trong khi vẫn đảm bảo hoạt động bình thường cho các bộ phận khác.

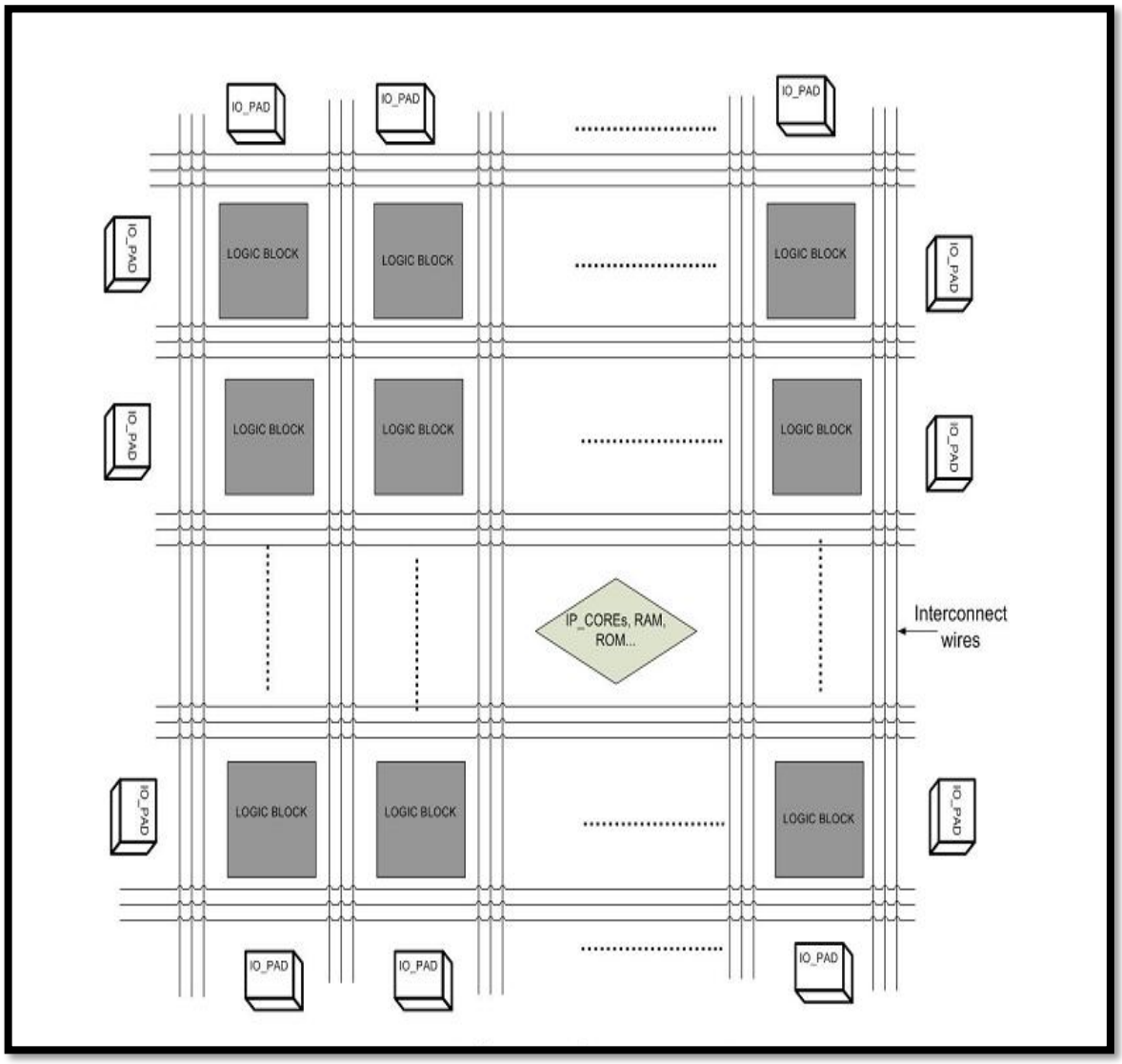
1.3 Ứng dụng

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, máy đánh cờ (Máy đánh cờ Hydra có 32 bộ vi xử lý cộng thêm FPGA đã chiến thắng kiện tướng quốc tế Michael Adams trong năm 2005.)...

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.

1.4 Cấu trúc của FPGA

FPGA (Field-programmable gate array) là một thiết bị bán dẫn bao gồm các khối logic lập trình được gọi là “Logic Block”, và các kết nối khả trình. Các khối logic có thể được lập trình để thực hiện các chức năng của các khối logic cơ bản như AND, XOR, hoặc các chức năng kết hợp phức tạp như decoder hoặc các phép tính toán học. Trong hầu hết các kiến trúc FPGA, các khối logic cũng bao gồm các phần tử nhớ. Đó có thể là các Flip-Flop hoặc những bộ nhớ hoàn chỉnh hơn.



Hình 1. 1 Kiến trúc tổng quan FPGA

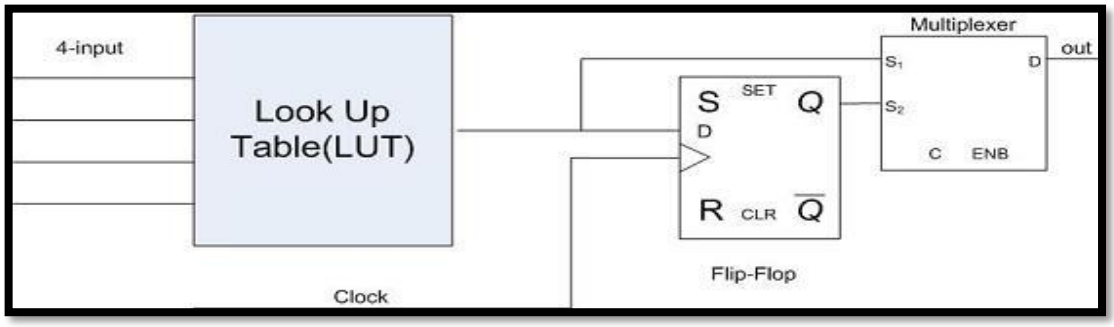
1.4.1 Khối logic FPGA

Phần tử chính của FPGA là các khối logic (logic block). Khối logic được cấu thành từ LUT và một phần tử nhớ đồng bộ flip-flop.

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.

Nếu nhìn cấu trúc tổng thể của mảng LUT thì ngoài 4 đầu vào kể trên còn hỗ trợ thêm 2 đầu vào bổ sung từ các khối logic phân bố trước và sau nó nâng tổng số đầu vào của LUT lên 6 chân. Cấu trúc này là nhằm tăng tốc các bộ số học logic.

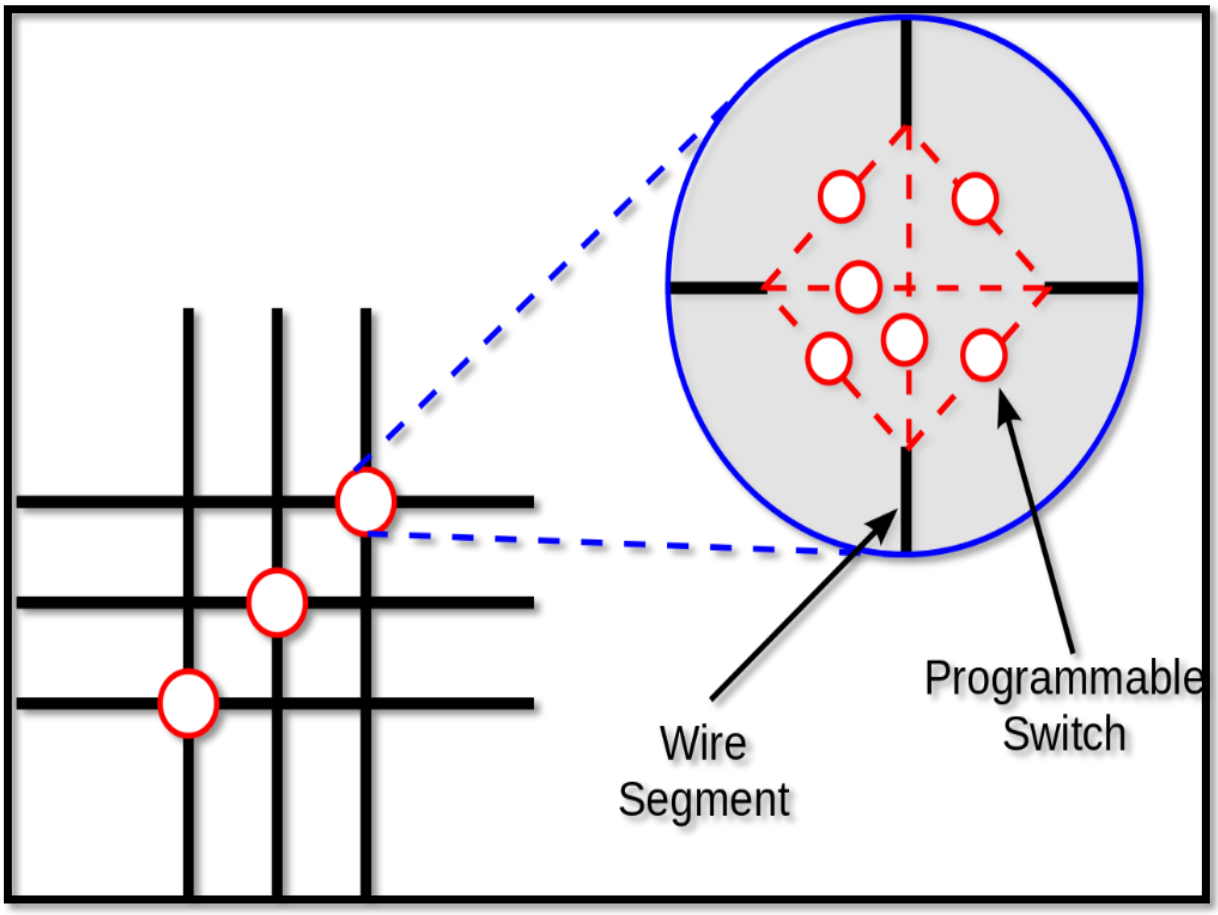
Mạng liên kết trong FPGA được cấu thành từ các đường kết nối theo hai phương ngang và đứng, tùy theo từng loại FPGA mà các đường kết nối được chia thành các nhóm khác nhau, ví dụ trong XC4000 của Xilinx có ba loại kết nối: ngắn, dài và rất dài. Các đường kết nối được nối với nhau thông qua các khối chuyển mạch lập trình được (programmable switch), trong một khối chuyển mạch chứa một số lượng nút chuyển lập trình được đảm bảo cho các dạng liên kết phức tạp khác nhau.



Hình 1. 2 Khối Logic FPGA

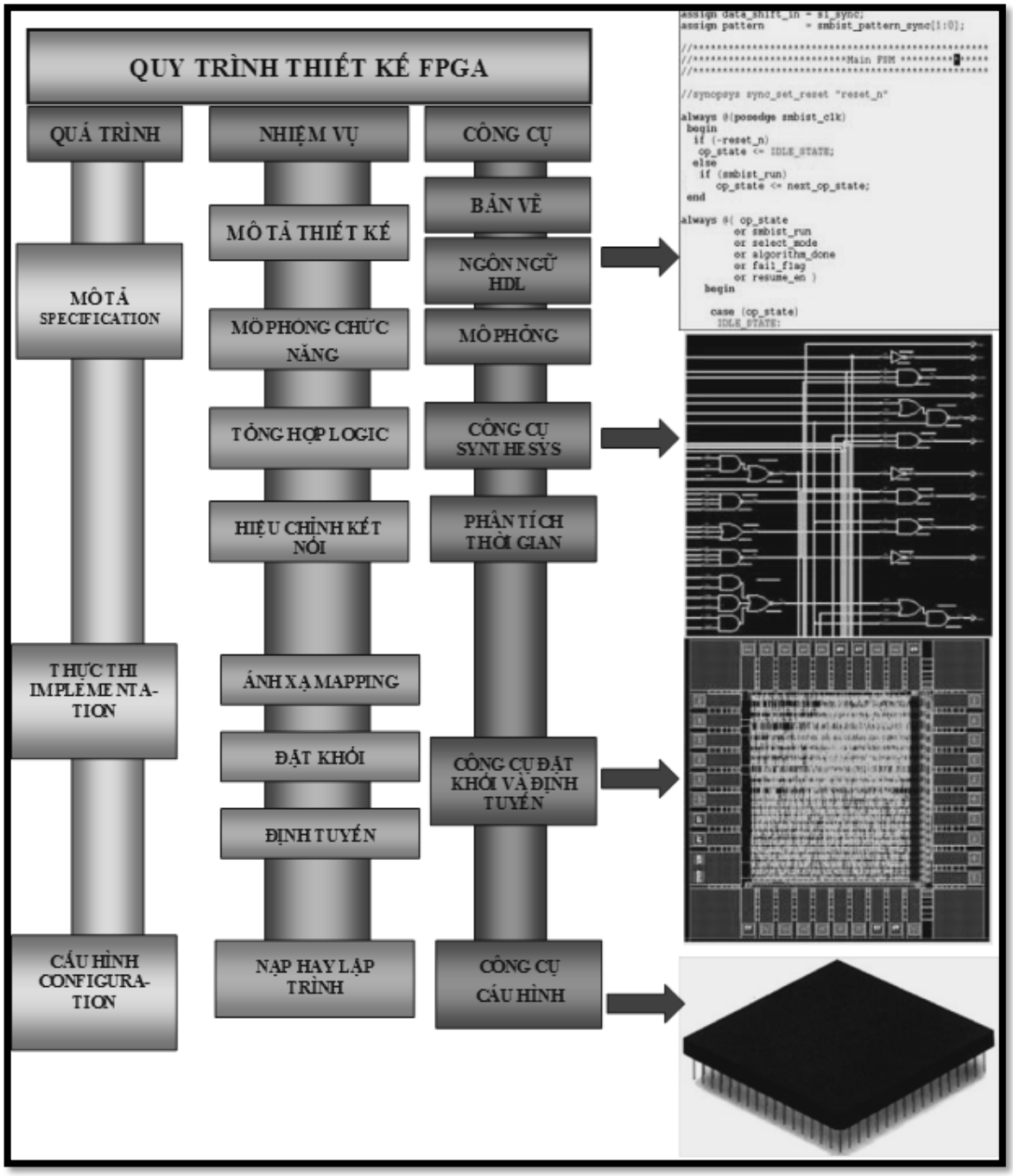
1.4.2 Các thành phần tích hợp

Ngoài các khối logic tùy theo các loại FPGA khác nhau mà có các phần tử tích hợp thêm khác nhau, ví dụ để thiết kế những ứng dụng SoC, trong dòng Virtex 4,5 của Xilinx có chứa nhân xử lý PowerPC, hay trong Atmel FPSLIC tích hợp nhân AVR…, hay cho những ứng dụng xử lý tín hiệu số DSP trong FPGA được tích hợp các DSP Slide là bộ nhân cộng tốc độ cao, thực hiện hàm A*B+C, ví dụ dòng Virtex của Xilinx chứa từ vài chục đến hàng trăm DSP slices với A, B, C 18-bit.



Hình 1. 3 Khối chuyển mạch của FPGA

1.4.3 Quy trình thiết kế tổng quát FPGA



Hình 1. 4 Quy trình thiết kế tổng quan FPGA

1.4.3.1 Mô tả ban đầu và thiết kế

Mô tả ban đầu về thiết kế. (Specification)

- Khi xây dựng một chip khả trình (FPGA) với ý nghĩa dành cho một ứng dụng riêng biệt, vì xuất phát từ mỗi ứng dụng trong thực tiễn cuộc sống, sẽ đặt ra yêu cầu phải thiết kế IC thực hiện tối ưu nhất những ứng dụng

đó. Bước đầu tiên của quy trình thiết kế này có nhiệm vụ tiếp nhận các yêu cầu của thiết kế và xây dựng nên kiến trúc tổng quát của thiết kế.

Mô tả thiết kế(Design Specification)

- Trong bước này, từ những yêu cầu của thiết kế và dựa trên khả năng của công nghệ hiện có, người thiết kế kiến trúc sẽ xây dựng nên toàn bộ kiến trúc tổng quan cho thiết kế. Nghĩa là trong bước này người thiết kế kiến trúc phải mô tả được những vấn đề sau:
  - Thiết kế có những khối nào?
  - Mỗi khối có chức năng gì?
  - Hoạt động của thiết kế và của mỗi khối ra sao ?
  - Phân tích các kỹ thuật sử dụng trong thiết kế và các công cụ, phần mềm hỗ trợ thiết kế.
- Một thiết kế có thể được mô tả sử dụng ngôn ngữ mô tả phần cứng, như VHDL hay Verilog HDL hoặc có thể mô tả qua bản vẽ mạch (schematic capture). Một thiết kế có thể vừa bao gồm bản vẽ mạch mô tả sơ đồ khối chung, vừa có thể dùng ngôn ngữ HDL để mô tả chi tiết cho các khối trong sơ đồ.

Mô phỏng chức năng (Function simulation).

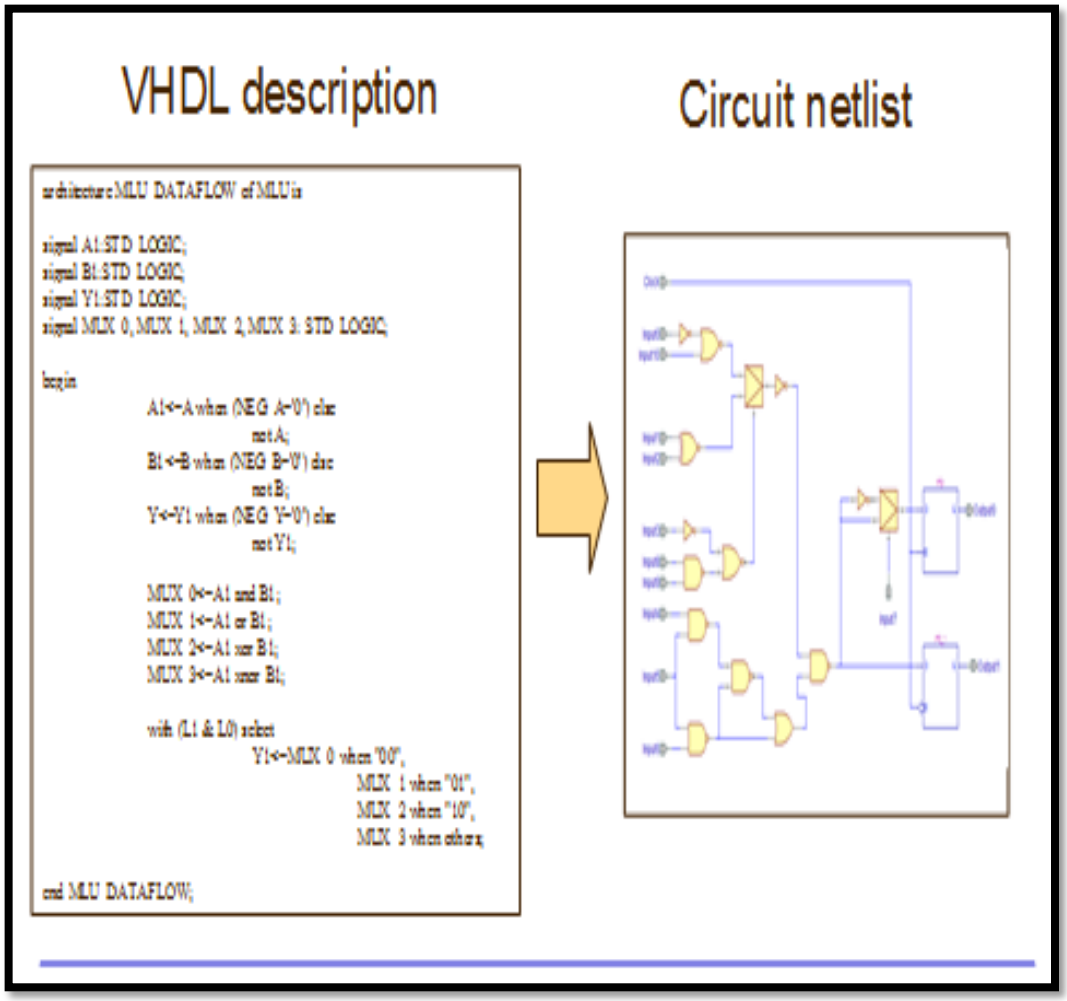
- Sau khi mô tả thiết kế, người thiết kế cần mô phỏng tổng thể thiết kế về mặt chức năng để kiểm tra thiết kế có hoạt động đúng với các chức năng yêu cầu.

Tổng hợp logic (Logic Synthesis).

- Tổng hợp logic là quá trình tổng hợp các mô tả thiết kế thành sơ đồ bố trí mạch (netlist). Quá trình chia thành 2 bước: chuyển đổi các mã RTL, mã HDL thành mô tả dưới dạng các biểu thức đại số Boolean và dựa trên các biểu thức này kết hợp với thư viện tế bào chuẩn sẵn có để tổng hợp nên một thiết kế tối ưu.

Hiệu chỉnh các kết nối (Datapath Schematic).

- Nhập netlist và các ràng buộc về thời gian vào một công cụ phân tích thời gian (timing analysic). Công cụ phân tích này sẽ tách rời tất cả các kết nối của thiết kế, tính thời gian trễ của các kết nối dựa trên các ràng buộc. Dựa trên kết quả phân tích (report) của công cụ phân tích, xác định các kết nối không thỏa mãn về thời gian. Tùy theo nguyên nhân dẫn đến không thỏa mãn mà ta có thể viết lại mã và tiến hành lại tổng hợp logic hoặc hiệu chỉnh lại các ràng buộc.



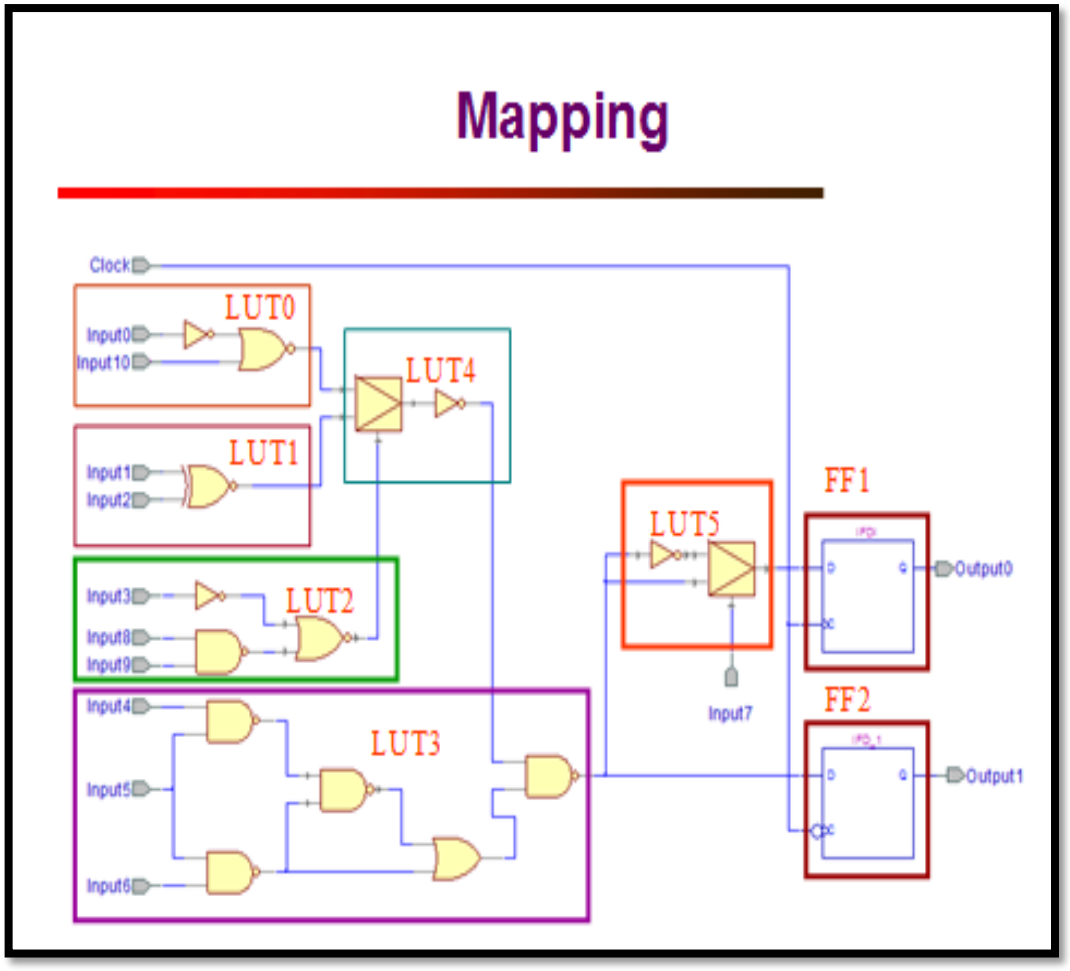
Hình 1. 5 Logic Synthesis

1.4.3.2 Thực thi

Ta đã có sơ đồ bố trí netlist mô tả tổng thể thiết kế tại mức cổng (chỉ gồm các cổng logic cơ bản và các mạch logic khác như: MUX). Quá trình này sẽ đặt sơ đồ netlist này lên chip, gọi là quá trình thực thi (Device Implementation).

Quá trình gồm các bước:

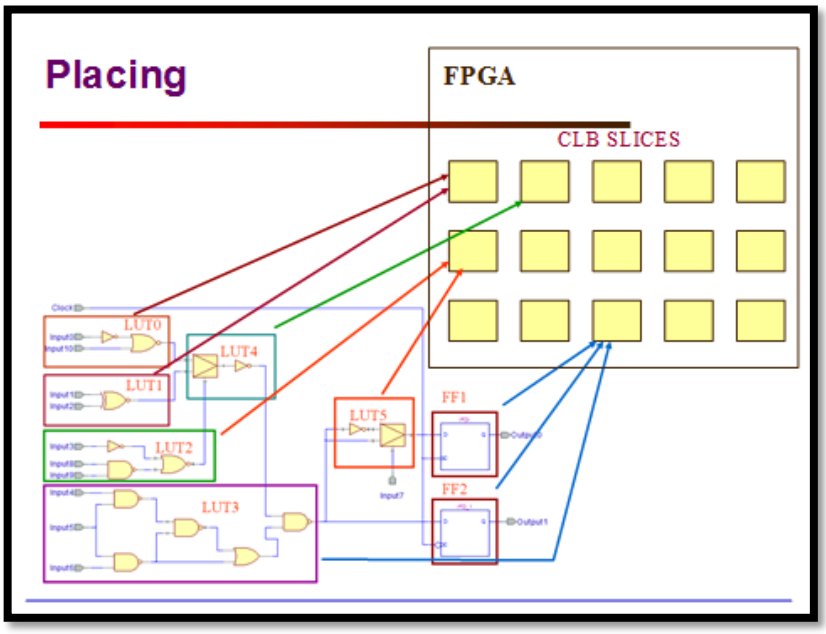
- **Ánh xạ** (mapping hay còn gọi fitting - ăn khớp) : chuẩn bị dữ liệu đầu vào, xác định kích thước các khối. Các khối này sẽ phải phù hợp với cấu trúc của 1 tế bào cơ bản của FPGA. (gồm nhiều cổng logic) và đặt chúng vào các vị trí tối ưu cho việc chạy dây.



Hình 1. 6 Sơ đồ gán chân

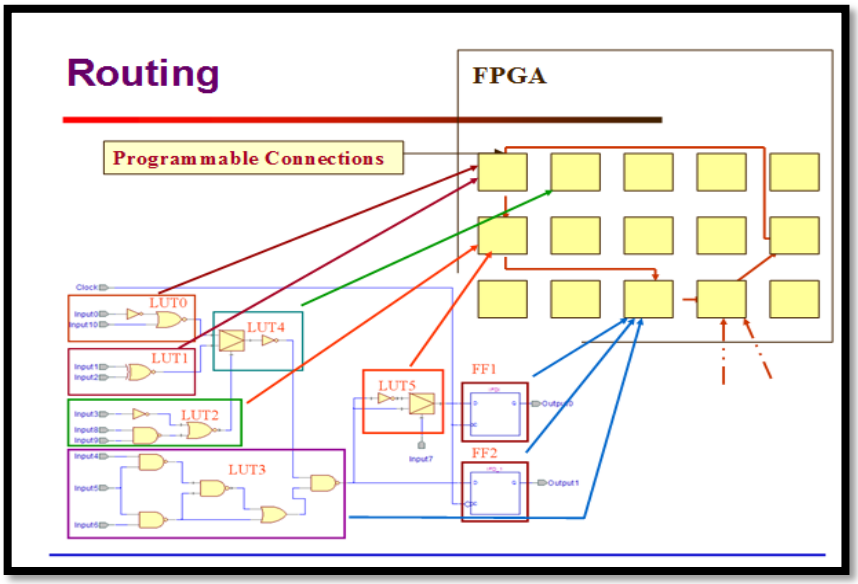
-Đặt khối và định tuyến (Place & Route):

+ Đặt khối: đặt các khối ánh xạ vào các tế bào (cell) ở vị trí tối ưu cho việc chạy dây**.**



Hình 1. 7 Sơ đồ không gian gán bên trong FPGA

+Định tuyến: Bước này thực hiện việc nối dây các tế bào.



Hình 1. 8 Sơ đồ định tuyến

Để thực hiện việc này, chúng ta cần có các thông tin sau:

- Các thông tin vật lý về thư viện tế bào, ví dụ kích thước tế bào, các điểm để kết nối, định thời, các trở ngại trong khi đi dây.
- Một netlist được tổng hợp sẽ chỉ ra chi tiết các instance và mối quan hệ kết nối bao gồm cả các đường dẫn bị hạn chế trong thiết kế.
- Tất cả các yêu cầu của tiến trình cho các lớp kết nối, bao gồm các luật thiết kế cho các lớp chạy dây, trở kháng và điện dung, tiêu thụ năng lượng, các luật về sự dẫn điện trong mỗi lớp.

1.4.3.3 Quá trình nạp code và lập trình

- Sau quá trình thực hiện, thiết kế cần được nạp vào FPGA dưới dạng dòng bit (bit stream).

- Quá trình nạp thiết kế (download) vào FPGA thường nạp vào bộ nhớ bay hơi, ví dụ như SRAM. Thông tin cấu hình sẽ được nạp vào bộ nhớ. Dòng bit được truyền lúc này sẽ mang thông tin định nghĩa các khối logic cũng như kết nối của thiết kế. Tuy nhiên, lưu ý rằng, SRAM sẽ mất dữ liệu khi mất nguồn nên thiết kế sẽ không lưu được đến phiên làm việc kế tiếp.

- Lập trình (program) là thuật ngữ để mô tả quá trình nạp chương trình cho các bộ nhớ không bay hơi, ví dụ như PROM. Như vậy, thông tin cấu hình vẫn sẽ được lưu trữ khi mất nguồn.

1.5 Tổng quan về Verilog

1.5.1 Giới thiệu về ngôn ngữ mô tả phần cứng Verilog

Verilog là ngôn ngữ mô tả phần cứng (Hardware Description Language) được sử dụng trong việc thiết kế các các hệ thống số, các mạch tích hợp... Cùng với ngôn ngữ VHDL, Verilog là một trong hai ngôn ngữ mô tả phần cứng phổ biến nhất hiện nay.

Verilog cũng có các đặc điểm như tính độc lập về công nghệ, dễ dàng trong thiết kế và debug, cũng như tính đơn giản so với các thiết kế bằng sơ đồ khối (schematics), đặc biệt là trong việc thiết kế các hệ thống phức tạp.

Verilog lần đầu được giới thiệu vào năm 1984 bởi công ty Gateway Design Automatic. Verilog không được chuẩn hóa và đều được chỉnh sửa ở hầu hết các phiên bản sau từ năm 1984 đến năm 1990. Năm 1995 Verilog chính thức được chuẩn hóa bởi tổ chức IEEE.

Nhiều người cho rằng Verilog dễ học và sử dụng hơn VHDL nhờ cú pháp khá giống với ngôn ngữ C (ngôn ngữ được dạy trong hầu hết các trường đại học, cao đẳng). Tuy cả hai ngôn ngữ Verilog và VHDL đều được sinh ra tại Mỹ nhưng Verilog lại được dùng phổ biến hơn tại đây.

Verilog được dùng để xây dựng các ứng dụng trên nền các công nghệ như FPGA, CPLDs…Code Verilog dùng để mô tả các hệ thống số được xây dựng trong các thiết bị lập trình được của các hãng như Xilinx, Altera, hay Amtel…

Một số ưu điểm của ngôn ngữ Verilog:

Nền tảng mạnh : chuẩn hóa năm 1995 bởi IEEE, hỗ trợ công nghiệp, phổ biến cho các nhà ASIC vì dễ học, cho phép mô phỏng và tổng hợp hiệu quả.

Tính đa năng: cho phép quá trình thiết kế thực thể thực hiện trong môi trường thiết kế cả phân tích và kiểm tra. Tuy nhiên Verilog không thích hợp lắm cho các thiết kế mức hệ thống phức tạp, đây là trở ngại chính của Verilog.

Hỗ trợ công nghiệp: phổ biến cho các nhà thiết kế ASIC vì dễ học , cho phép mô phỏng nhanh và tổng hợp hiệu quả

Có khả năng mở rộng IEEE Std 1364 chứa định nghĩa của PLI Verilog (Programming Language Interface) cho phép mở rộng khả năng của Verilog. Nó là một tập hợp các bộ định tuyến cho phép các chức năng bên ngoài truy nhập thông tin chức năng thiết kế Verilog.

1.5.2 Cấu trúc chương trình Verilog

Cấu trúc

Module < tên>

(

Input < tên các đầu vào>(reg , wire...)

Output < tên các đầu ra>>(reg , wire...)

);

(parameter) // tham số có thể khai báo hoặc không tùy vào bài toán

// các câu lệnh

endmodule

Các phép gán biến

Phép gán =(gán theo tuần tự): Phép gán Blocking thực hiện các câu lệnh một cách tuần tự. Phép gán này phụ thuộc vào giá trị của câu lệnh dưới.

Phép gán <=(phép gán song song): Phép gán Non-blocking thực hiện các câu lệnh một cách song song. Tất cả các phép gán sẽ thực hiện đồng thời ngay tại 1 thời điểm mà không quan tâm đến thứ tự.

Phép gán assign(gán liên tục): Phép gán assign: phía bên trái của phép gán là một biến còn phía bên phải là một biểu thức. Kiểu dữ liệu bên trái bắt buộc phải là net và bên phải có thể là kiểu net, reg… Phía bên trái tách biệt với phía bên phải bằng kí tự "=". Giá trị của phía bên trái thay đổi khi giá trị biểu thức bên phải thay đổi.

Thông thường có 3 loại:

Cấu trúc: Thực hiện mô phỏng ở mức thấp(mức cổng logic), mạch tổng hợp chính xác

Luồng dữ liệu: Đầu ra phụ thuộc đầu vào (thường dùng từ khóa assign). Mô tả chính xác hầu như mọi trường hợp

Hành vi: Chỉ mô tả chức năng của mạnh. Mức hành vi mô tả một hệ thống số bằng những thuật toán (một số lệnh giống ngôn ngữ C như: if, case, of, while,..). Mạch khố tổng hợp chính xác (dùng từ khóa always @)

Khối Always, Initial

Khối always: Được dử dụng để mô tả sự kiện xảy ra trong các điều kiện. nhất định ví dụ if, case….

Là cấu trúc chính trong khuôn mẫu RTL (Register Transfer Level).Khối always có thể được dùng trong chốt, flip flop hay các kết nối logic.Tất cả các khối always trong một module thực thi một cách liên tục. Nếu các lệnh của khối always nằm trong phạm vi khối begin… end thì được thực thi liên tục, nếu nằm trong khối fork… join, chúng được thực thi đồng thời (chỉ trong mô phỏng). Khối always thực hiện bằng mức, cạnh lên/xuống của một or nhiều tín hiệu (các tín hiệu cách nhau bởi từ khóa OR).

Cú pháp:

always @(các đầu vào)

begin

// Các câu lệnh

end

Hoặc thay always @(các đầu vào) bằng always @*

Khối Initial: Tương tự khối always nhưng khối initial chỉ thực thi một lần từ lúc bắt đầu(không bắt buộc phải là đầu tiên) của quá trình mô phỏng. Khối này là tiêu biểu để biến khởi chạy và chỉ định dạng sóng tín hiệu trong lúc mô phỏng. Khối initial chỉ dành cho thực hiện mã testbench. Khi chương trình tổng hợp khối initial sẽ được bỏ qua.

CHƯƠNG II. GIỚI THIỆU VỀ KIT SPARTAN-3E BOARD VÀ MÔI TRƯỜNG LẬP TRÌNH ISE 14.7

2.1 Spartan-3E kit board

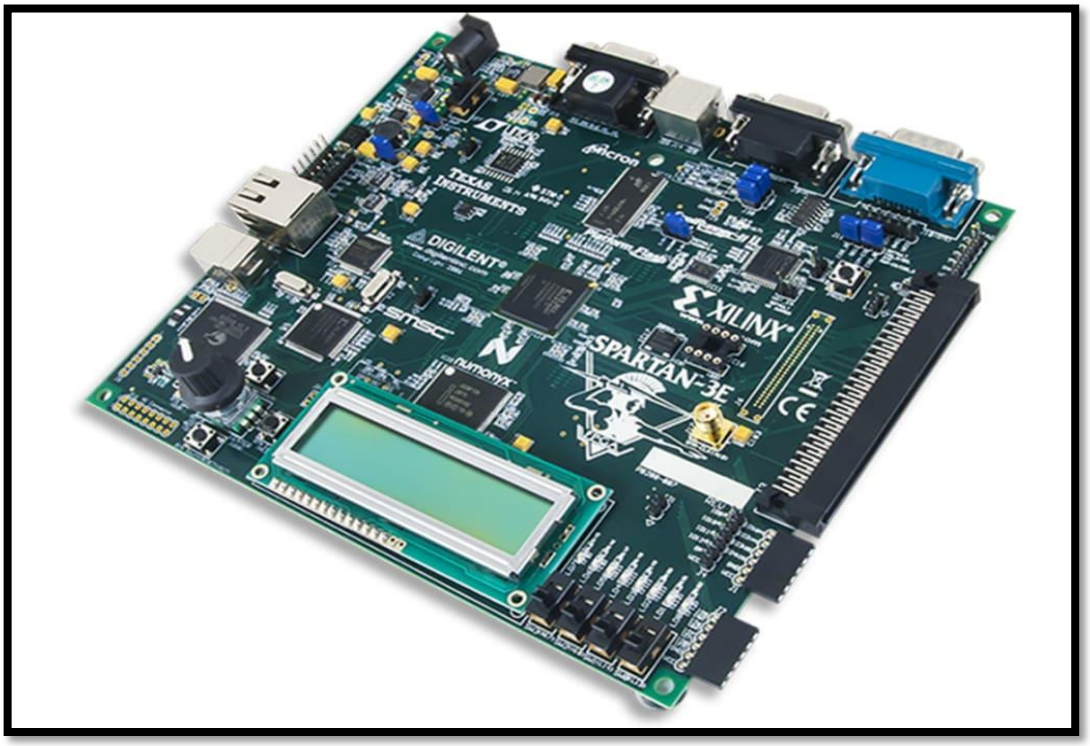
2.1.1 Các thành phần của kit Spartan-3E

- Xilinx XC3S500E Spartan-3E FPGA: con chính của KIT
- Cấu hình Flash nền tảng Xilinx 4 Mbit PROM
- Xilinx 64-macrocell XC2C64A CPLD CoolRunner
- 64 MByte (512 Mbit) DDR SDRAM, giao diện dữ liệu x16, 100+ MHz
- 16 MByte (128 Mbit) của NOR Flash song song (Intel StrataFlash)
- 16 Mbit của Flash nối tiếp SPI (STMicro)
- Màn hình LCD 2 dòng, 16 ký tự
- Cổng PS / 2 chuột hoặc bàn phím
- Cổng hiển thị VGA
- 10/100 Ethernet PHY (yêu cầu Ethernet MAC trong FPGA)
- Hai cổng RS-232 9 chân (kiểu DTE- và DCE)
- Giao diện tải xuống / gỡ lỗi FPGA / CPLD dựa trên USB trên bo mạch
- Bộ dao động xung nhịp 50 MHz
- SHA-1 EEPROM nối tiếp 1 dây để bảo vệ sao chép dòng bit
- Đầu nối mở rộng Hirose FX2
- 3 đầu nối mở rộng 6 chân Digilent
- Bộ chuyển đổi Digital-to-Analog (DAC) bốn đầu ra, dựa trên SPI
- Bộ chuyển đổi tín hiệu tương tự sang tín hiệu số (ADC) hai đầu vào, dựa trên giao thức SPI
- Cổng gỡ lỗi ChipScope ™ SoftTouch
- -Bộ mã hóa vòng quay với trục nút nhấn
- 8 đèn LED rời
- 4 công tắc trượt

2.1.2 Các thông số kĩ thuật và một số hình ảnh của kit Spartan_3E.

Spartan-3E là họ FPGA mới nhất của Xilinx với nhiều ưu điểm nổi bật. Đầu tiên phải kể đến là khả năng tích hợp của spartan-3E từ 100,000 gates đến1,6 triệu gates. Ngoài ra, còn một số đặc điểm chính của Spartan-3E là:

- Dễ sử dụng , giá thành thấp, tiêu thụ điện ít.
- Mật độ tích hợp nhiều phần tử logic(Đây là ưu điểm so với họ Spartan 3).
- Tốc độ xung nhịp hệ thống từ 5-300 Mhz.
- Năm mức tiêu thụ điện năng (3.3V;2.5V;1.8V;1.5V;1.2V)
- Tích hợp tới 376 chân I/O hay 156 cặp tín hiệu khác nhau.
- Truyền dữ liệu với tốc độ khá cao.



Hình 2. 1 Spartan-3E Starter Kit Board

Cấu trúc của Spartan-3e

Các thành phần:

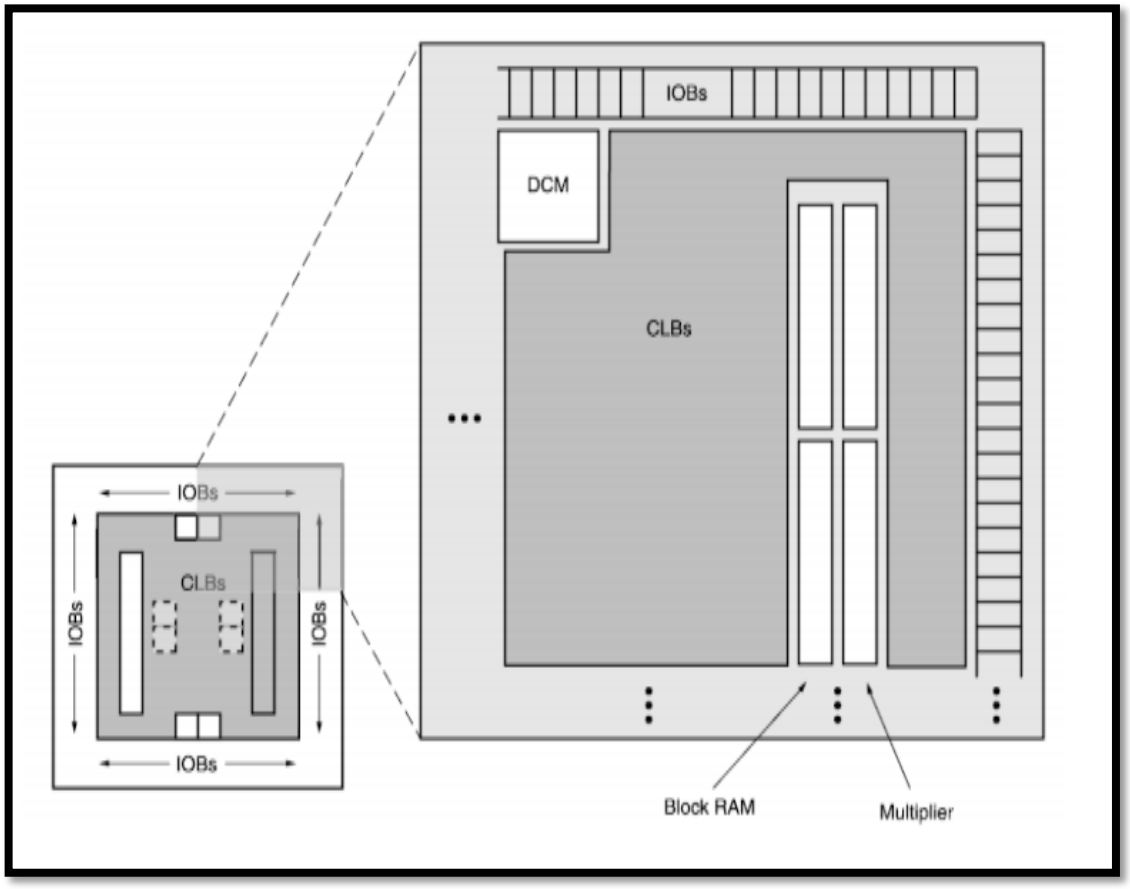
Input/Output Blook (Ios): các khối vào ra

Configurable Logic Blocks (CLBs): được cấu tạo từ look-Up Table(LUTs).

Block RAM: Hỗ trợ 16 Kb RAM trên mỗi Block RAM, số lượng các Block RAM tùy thuộc vào mỗi chip, với XC3S500E có 20 Block 18 bit.

Digital Clock Manager(DCM) Blocks: khối điều khiển xung clk.

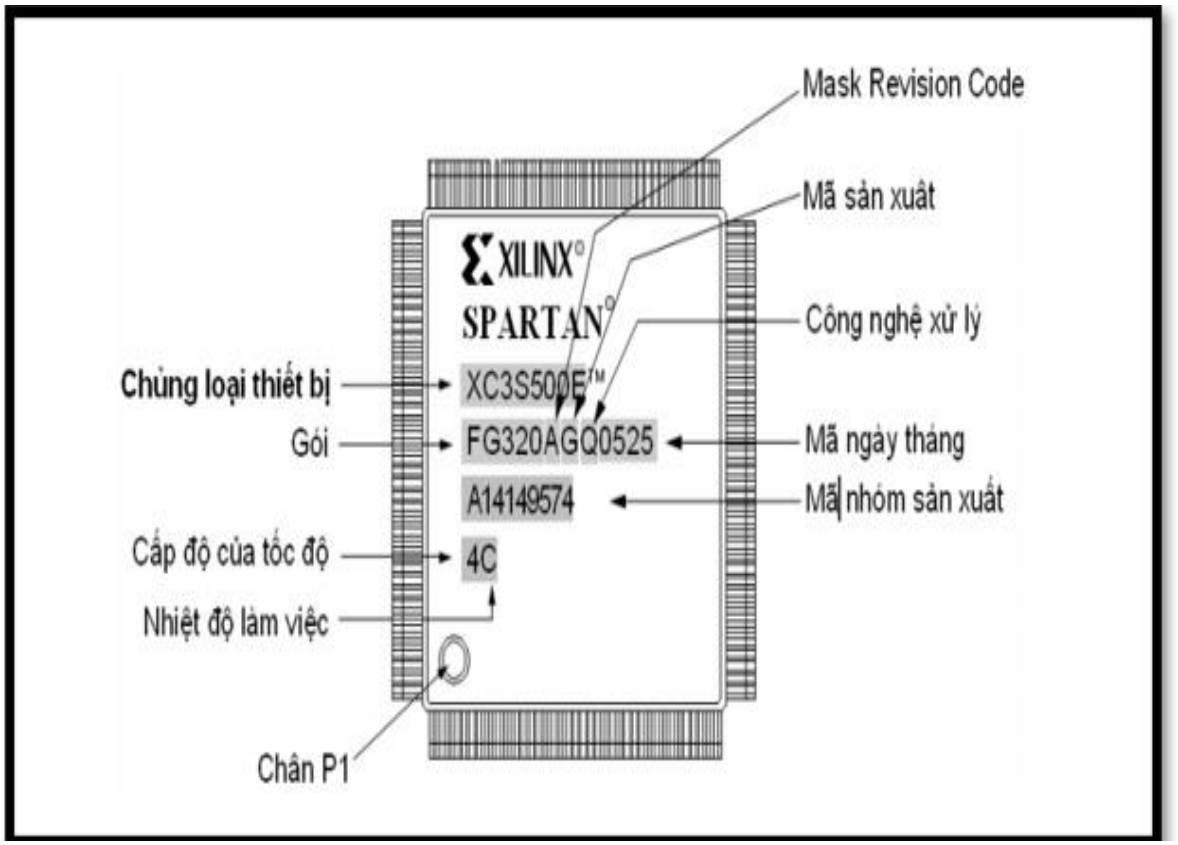
- Interconnect: các kết nối.



Hình 2. 2 Cấu trúc các thành phần của Spartan 3E

Spartan-3E Starter kit board là một công cụ hữu hiệu cho bất kỳ ai đang có ý định thiết kế các sản phẩm dựa trên công nghệ FPGA. Đây là một giải pháp cơ bản cho nhằm tối ưu thời gian và chi phí ban đầu. Nó cho phép chế tạo ngay với giá thành sản phẩm thấp. Bộ kit này là một thiết bị cấu trúc logic có thể người sử dùng lập trình trực tiếp mà không sử dụng bất kỳ một công cụ chế tạo mạch tích hợp nào.

2.1.3 Thông số chíp và ý nghĩa

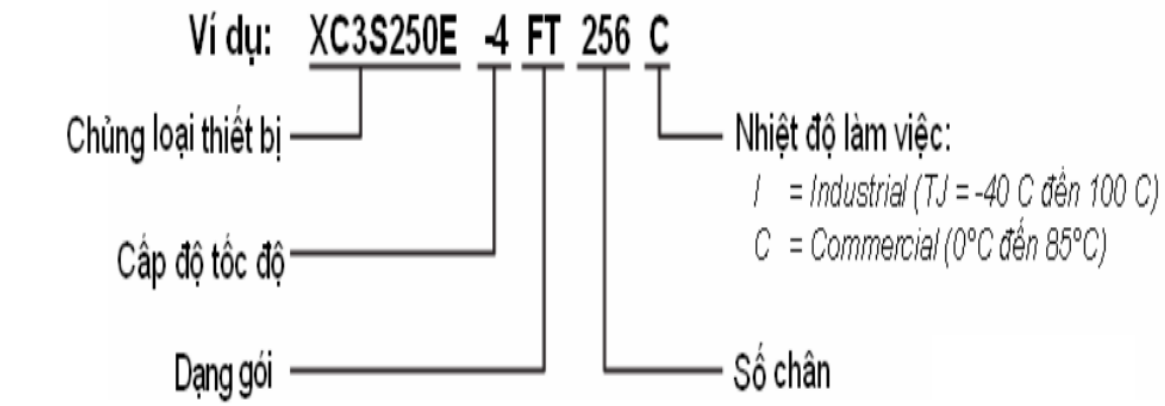


Hình 2. 3 Chíp Spartan-3E Xilink với các thông số

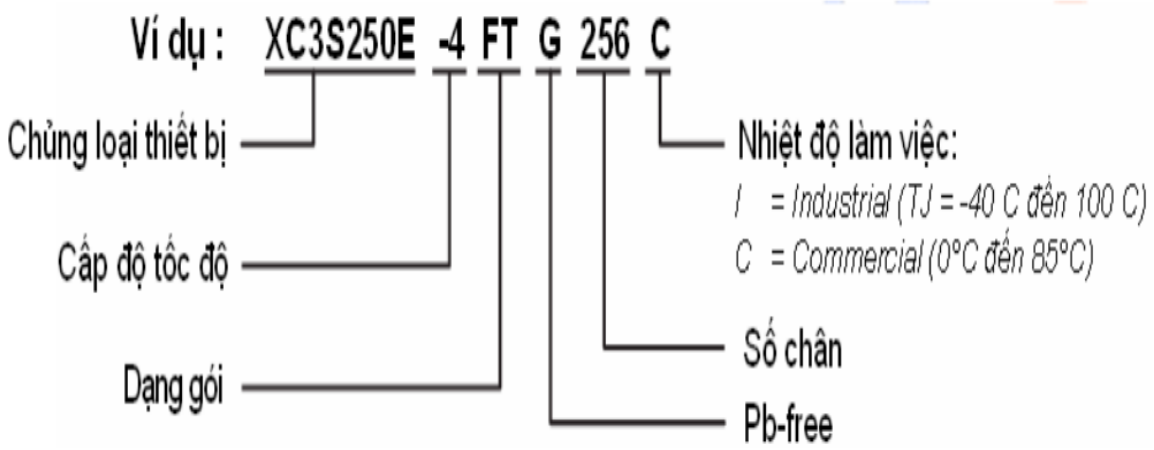
Trên bề mặt chíp được in các mã số, dựa vào các mã số này, người thiết kế mạch có thể biết được khả năng làm việc của bo mạch và lựa chọn để mua thiết bị phù hợp với nhu cầu sử dụng.

Các bo Kit phát triển Spartan-3E được sản xuất ở hai dạng gói cả tiêu chuẩn và Pb-free cho tất cả các thiết bị sản xuất. Các gói Pb-free có chứa thêm ký tự ‘G’ trong mã gói

Standard Packaging



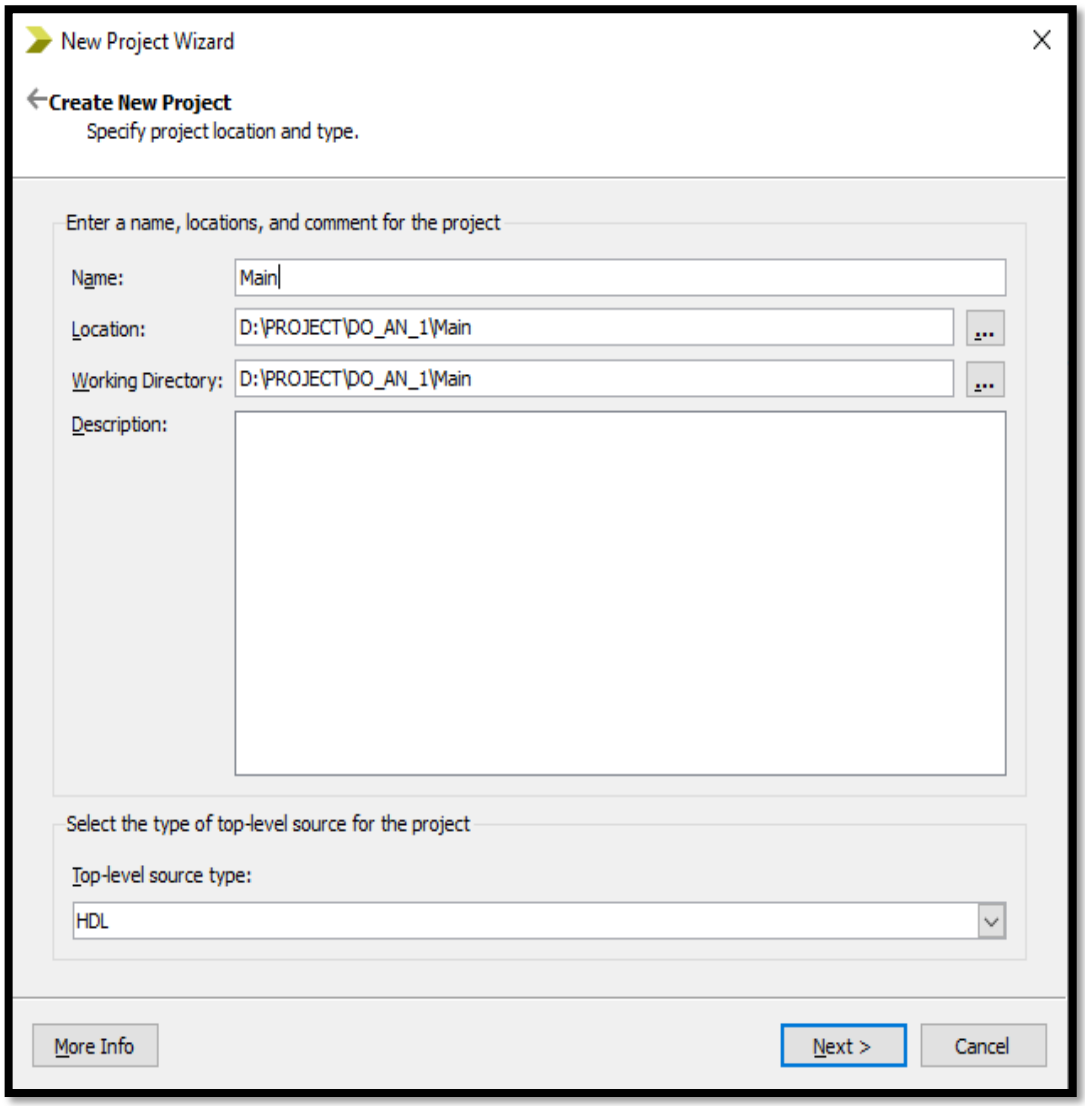
Pb-Free Packaging



2.2 Sơ lược về ISE 14.7

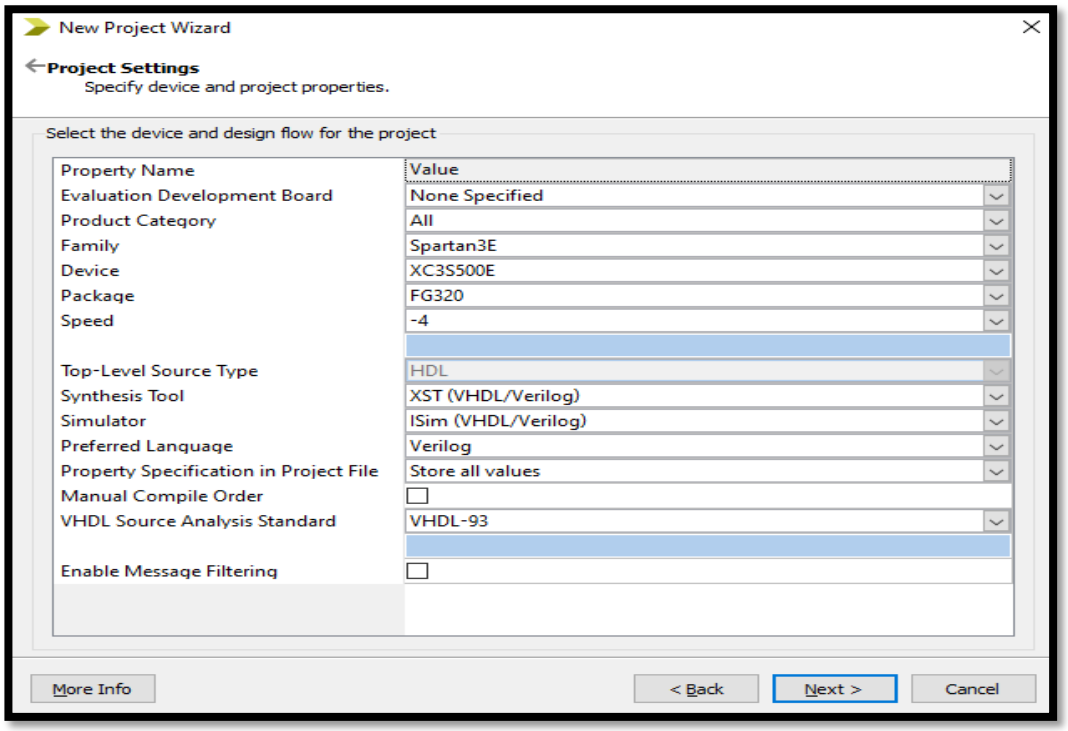
2.2.1 Tạo một Project

Vào Start > All Programs > Xilinx ISE 14.7 > Project Navigator để khởi động chương trình. Vào File > New Project cửa sổ hướng dẫn hiện ra như hình 2.4 bên dưới:



Hình 2. 4 Tạo mới project

Project Name: Đặt tên project. Project location : Nơi chứa project. Click Next, cửa sổ mới hiện ra như hình 2.5 ở bên dưới:



Hình 2. 5 Lựa chọn thiết bị cho chương trình

Ô Family : chọn Spartan3E .

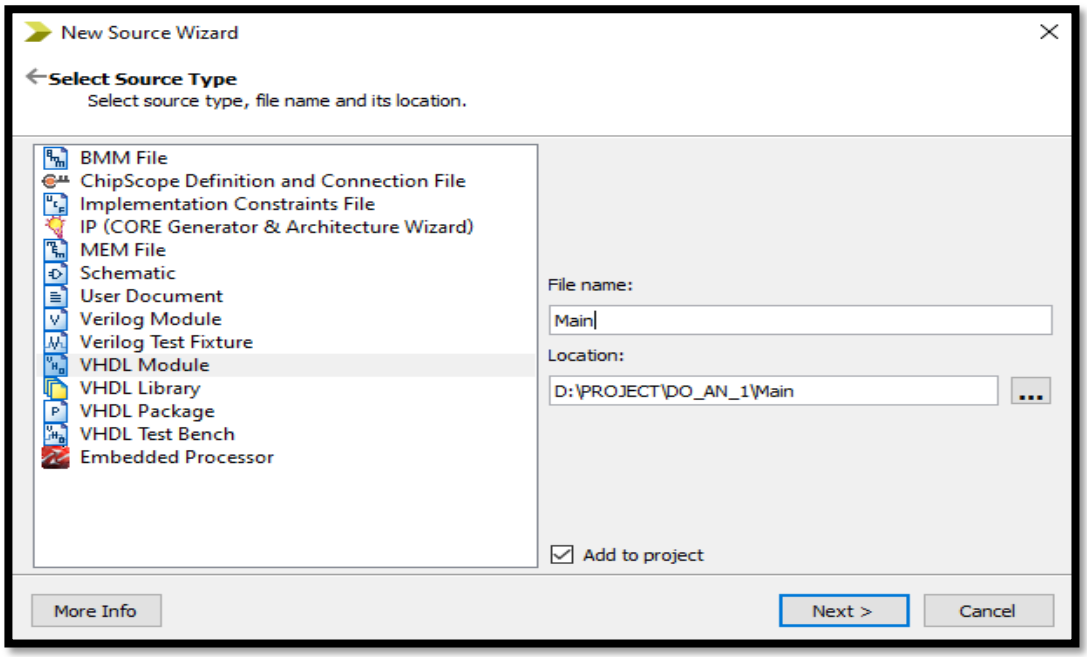
Ô Device : chọn XC3S500E.

Ô Package : chọn FG320.

Tiếp tục click Next , Next cửa mới hiện ra, chọn thanh : New Source.

Cửa sổ mới hiện ra, chọn Verilog Module để viết code verilog, nếu viết bằng

VHDL thì chọn : VHDL Module. cửa sổ hướng dẫn hiện ra như hình 2.6 ở bên dưới:



Hình 2. 6 Thêm module cho chương trình

Chọn tên file vhdl ở ô File name. (ở đây ta đang tạo bộ đếm nên chọn tên làcounter).

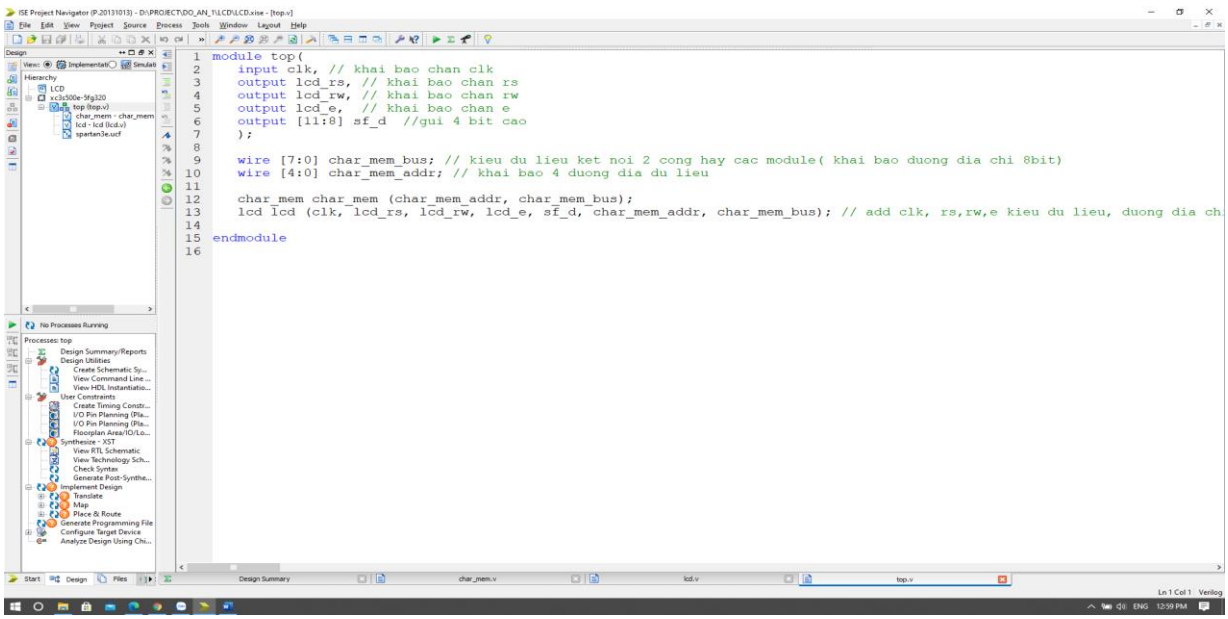
Tiếp tục click Next . Cửa sổ mới hiện ra , ta sẽ chọn giao diện cho vào ra cho khối counter:

Cột Port Name để chọn tên cổng

Cột Direction để chọn chân là lối vào, lối ra hay cả hai vào/ra

Cột Bus : nếu dùng bus thì tréo vào ô này. Ở đây, bộ đếm của ta có ngõ ra là một port 4 bit nên ta chéo ô này.

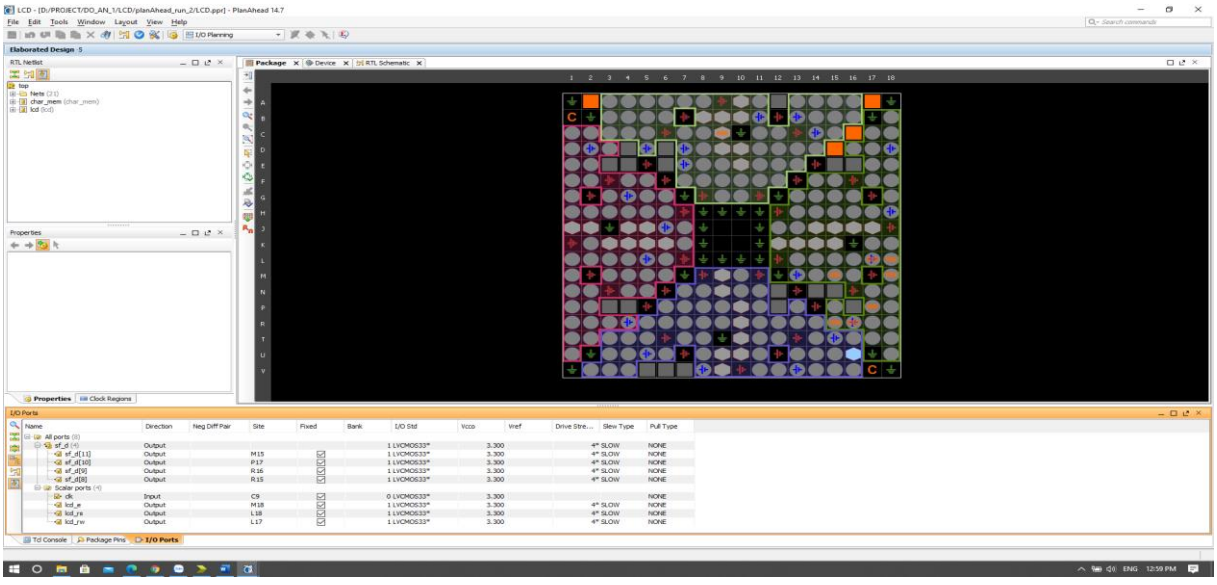
Tiếp tục ấn Next -> Next -> Finish , cuối cùng ta được kết quả như hình 2.7 bên dưới và sẽ viết code ở đây.



```
module top(
    input clk, // khai bao chan clk
    output lcd_rs, // khai bao chan rs
    output lcd_rw, // khai bao chan rw
    output lcd_e,  // khai bao chan e
    output [11:0] sf_d  //gui 4 bit cao
    );

    wire [7:0] char_mem_bus; // kieu du lieu ket noi 2 cong hay cac module( khai bao duong dia chi 8bit)
    wire [4:0] char_mem_addr; // khai bao 4 duong dia du lieu

    char_mem char_mem (char_mem_addr, char_mem_bus);
    lcd lcd (clk, lcd_rs, lcd_rw, lcd_e, sf_d, char_mem_addr, char_mem_bus); // add clk, rs,rw,e kieu du lieu, duong dia ch

endmodule
```

Hình 2. 7 Khung chương trình

Việc gắn chân : Chọn User Constraints -> IO/ Pin Planning (PlaAhead) – Post-Synthesis -> ra một hộp thoại và chọn YES -> Thực hiện việc gán chân giống như hình 2.8 bên dưới:



Hình 2. 8 Khung chương trình

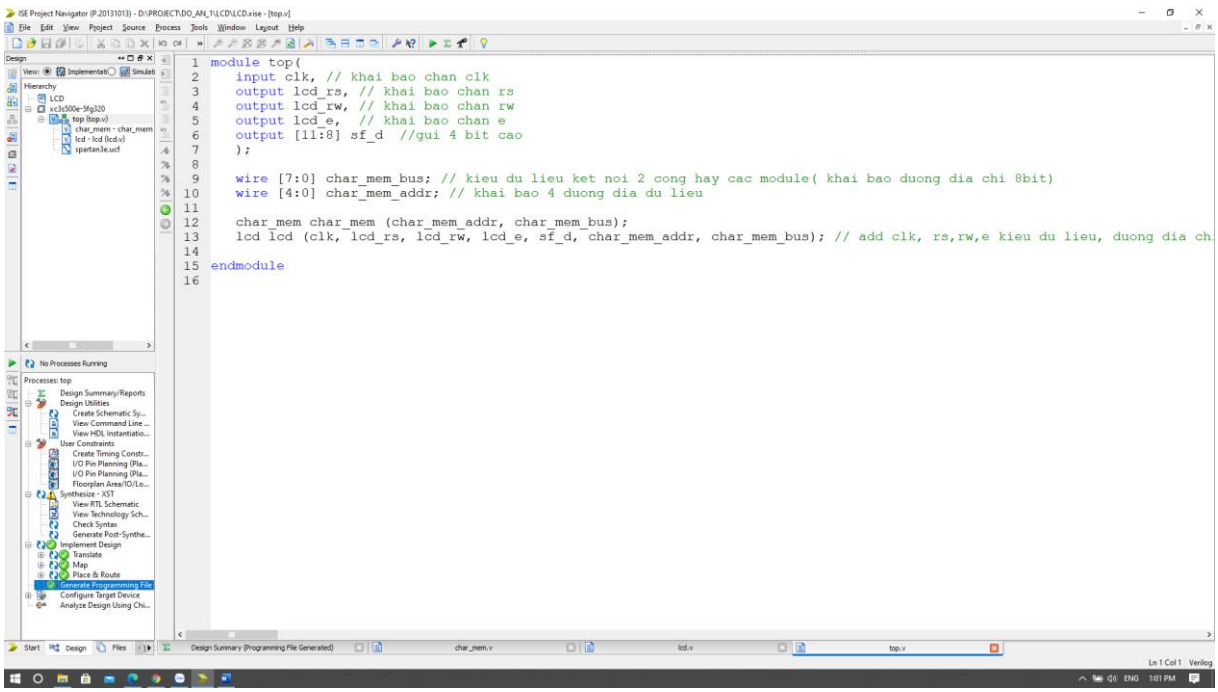
Kiểm tra mã nguồn : Tại cửa sổ process: Symthesis_XST-> Check Syntax

Kết nối với FPGA: Chọn Implement Design

Nạp vào FPGA: Tại Generate Programming File Configure Deveice

Xuất hiện của sổ ISE iMPACT gán vào khối hình ROM đầu tiên và quan sát kếtquả đàu ra trên Kit Spartan 3E

Kết quả là hình 2.9 bên dưới:



Hình 2. 9 Kiểm ra mã nguồn, kiểm tra gán chân và nạp code và chương trình

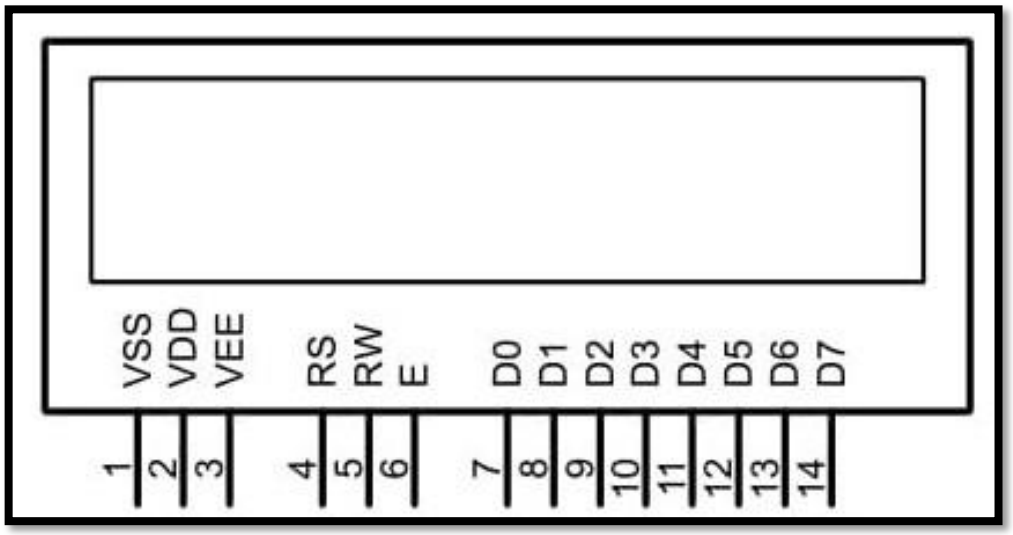
CHƯƠNG III. TỔNG QUAN VỀ LCD

3.1 Giới thiệu về LCD

LCD có rất nhiều dạng phân biệt theo kích thước từ vài kí tự đến hàng chục kí tự.

3.2 Các chân của LCD

LCD có nhiều loại nhưng có 2 loại phổ biến là loại 14 chân và 16 chân



Hình 3. 1 Hình nh LCD loại 14 chân

Các chân cấp nguồn:

+ Chân 1 nối với mass (OV), chân 2 nối với nguồn +5V.

+ Chân 3 chỉnh contrast thường nối với biến trở, chỉnh cho đến khi thấy được ký tự thì ngừng.

Các chân điều khiển:

+ Chân RS dung để điều khiển lựa chọn thanh ghi.

+ Chân R/W dung để điều khiển quá trình đọc và ghi.

+ Chân E là chân cho phép dạng xung chốt.

+ Các chân dữ liệu D7 đến D0: dùng để trao đổi dữ liệu giữa thiết bị điều khiển và LCD.

Chân số	Tên	Chức năng
1	VSS	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển
2	VDD	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với Vcc = 5V của mạch điều khiển
3	VEE	Chân này dùng để điều chỉnh độ tương phản của LCD
4	RS	Chân chọn thanh ghi (Register select). + Logic "0": Bus DB0-DB7 sẽ nối với thanh ghi lệnh IR của LCD (ở chế độ "ghi" - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ "đọc" - read) + Logic "1": Bus DB0-DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD.
5	RW	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic "0" để LCD hoạt động ở chế độ ghi, hoặc nối với logic "1" để LCD ở chế độ đọc.
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E. + Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào(chấp nhận) thanh ghi bên trong nó khi phát hiện một xung (high-to-low transition) của tín hiệu chân E. + Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện cạnh lên (low- to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.
7-14	DB0-DB7	Tám đường của bus dữ liệu dùng để trao đổi thông tin với MCU. Có 2 chế độ sử dụng 8 đường bus này : + Chế độ 8 bit : Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7.

		+ Chế độ 4 bit : Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là bit DB7.
15-16	A,K	Đèn của LCD

Bảng 1 Chức năng các chân của LCD 1602

3.3 Bộ điều khiển LCD và các vùng nhớ

Để điều khiển LCD thì có các IC chuyên dùng được tính hợp bên dưới LCD có mã số 447801 đến IC 447809

Bộ điều khiển có ba vùng nhớ nội, mỗi vùng có chức năng riêng:

- Vùng nhớ DDRAM:

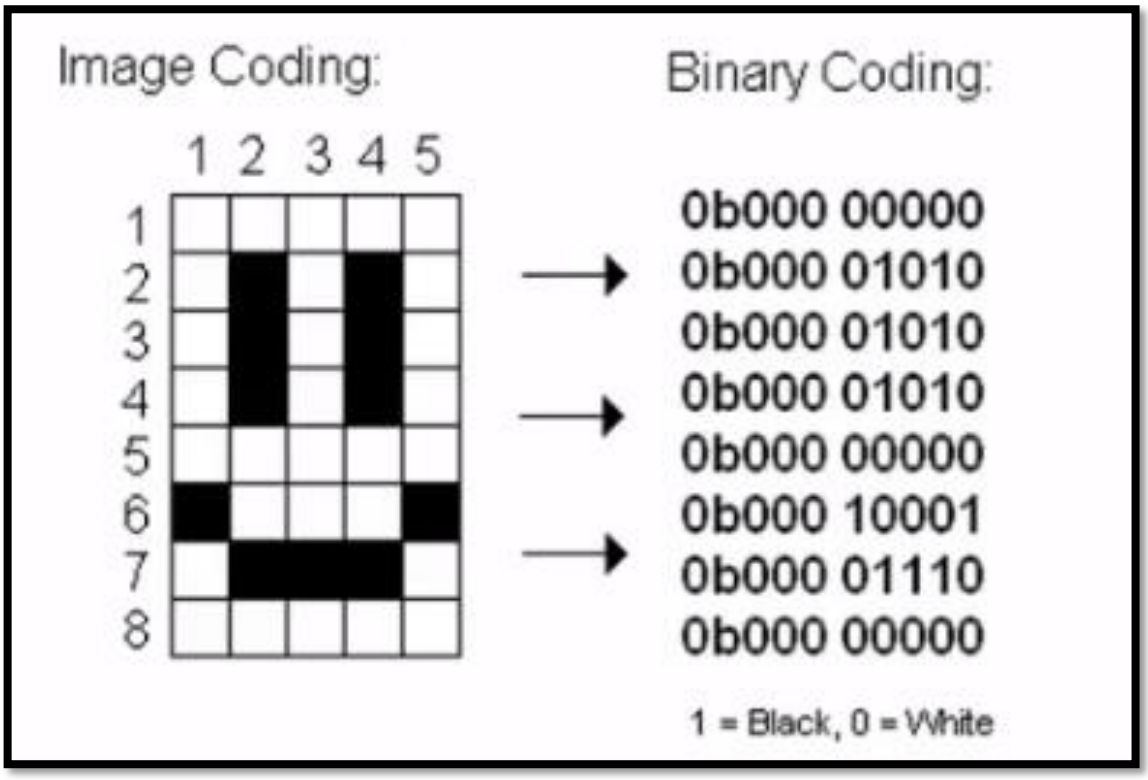
+ Bộ nhớ chứa dữ liệu để hiện thị (Display Data RAM: DDRAM) lưu trữ những mã ký tự để hiện thị lên màn hình

+ Mã ký tự lưu trữ trong vùng DDRAM sẽ tham chiếu với từng bipmap kí tự được lưu trữ trong CGROM đã được xác định trước hoạt đặt trong vùng do người sử dụng định nghĩa

- Vùng nhớ CGRAM: Bộ phát kí tự RAM- CGRAM (Character Generator RAM)

+ Cung cấp vùng nhớ để tao ra 8 kí tự tùy ý.

+ Mỗi kí tự gồm 5 cột và 8 hàng



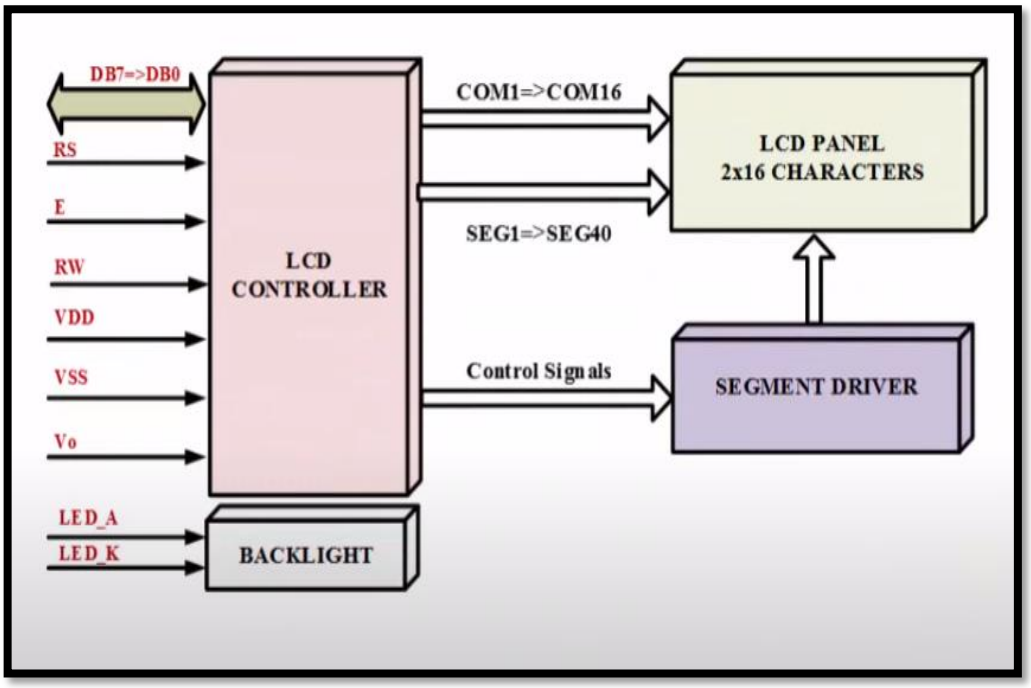
Hình 3. 2 .A Custom 5x8 Pixel Character

- Vùng nhớ CGROM : Bộ phát kí tự ROM- CGROM (Character Generator ROM)

+ Chứa các kiểu bitmap cho mỗi kí tự được xác định mà LCD có thể hiện thị, nhưng trong bảng mã ASCII

+ Mã kí tự lưu trong DDRAM sẽ được tham chiếu đến vị trí trong CGROM.

Sơ đồ khối bộ điều khiển LCD:



Hình 3. 3 Sơ đồ khối bộ điều khiển LCD

Gồm có 4 phần:

- Bộ điều khiển LCD
- Bảng ký tự LCD
- Bộ thúc tín hiệu của đoạn
- Đèn nền

3.4 Các lệnh điều khiển của LCD

Các IC chuyên dùng sẽ nhận được các lệnh điều khiển từ đối tượng giao tiếp với LCD để thực hiện cấu hình và hiển thị thông tin.

Các lệnh cấu hình LCD và hiện thị:

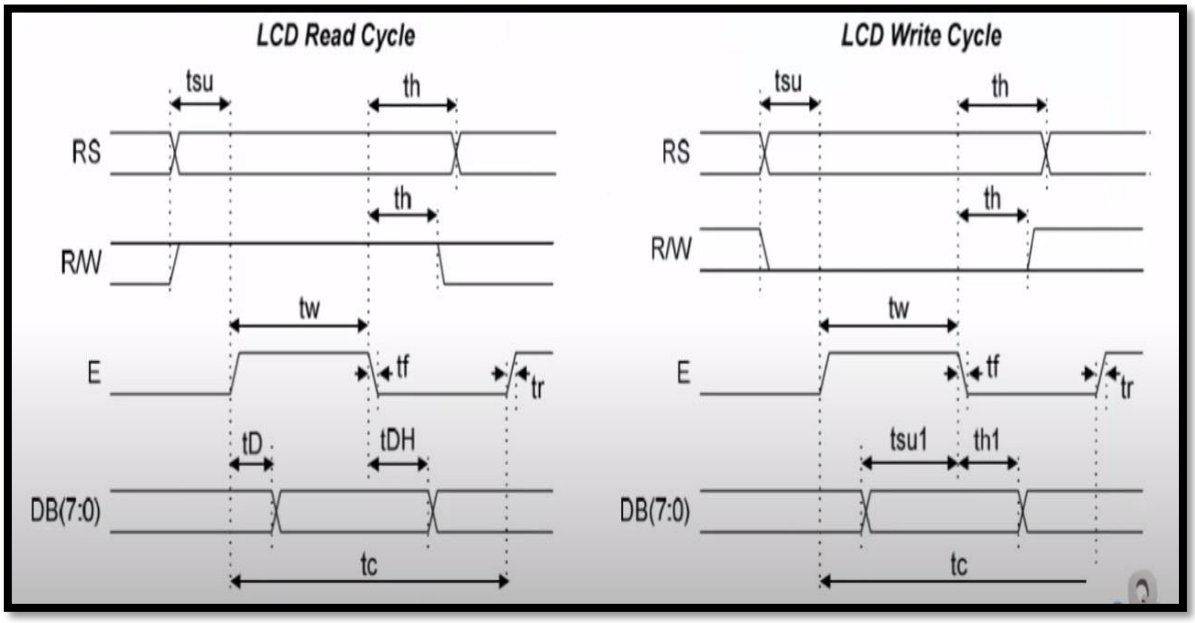
Mã HEX	**Chức năng**
0x01	Xóa toàn bộ nội dung đang hiển thì trên màn
0x02	Di chuyển con trỏ về vị trí đầu màn hình

0x06	Tự động di chuyển con trỏ đến vị trí tiếp theo mỗi khi xuất ra LCD một kí tự
0x0C	Bật hiển thị và tắt con trỏ
0x0E	Bật hiển thị và bật con trỏ
0x80	Di chuyển con trỏ về đầu dòng 1
0xC0	Di chuyển con trỏ về đầu dòng 2
0x38	Giao tiếp 8 bit hiển thị 2 dòng
0x28	Giao tiếp 4 bit, hiển thị 2 dòng

Bảng 2 Các lệnh cấu hình LCD và hiện thị

3.5 Hoạt động đọc ghi của LCD

Có hai hoạt động là đọc và ghi LCD, hai dạng song tương ứng:



Hình 3. 4 Hoạt động đọc ghi LCD

Hoạt động đọc:

Điều k hiển tín hiệu RS.

Điều khiển tín hiệu R/W lên mức 1.

Điều khiển tín hiệu E lên mức cao để cho phép.

Đọc dữ liệu từ bus dữ liệu DB7 đến DB0(data bus).

Điều khiển tín hiệu E về mức thấp.

Hoạt động ghi:

Điều khiển tín hiệu RS.

Điều khiển tín hiệu R/W về mức 0.

Điều khiển tín hiệu E lên mức cao để cho phép

Xuất dư liệu ra bus dữ liệu DB7 đến DB0 (data bus).

Parameter	Symbol	Min	Max	Unit	Test Pin
Enable cycle time	tc	500		ns	E
Enable High pulse width	tw	220		ns	E
Enable rise/fall time	tr, tf		25	ns	E
RS, R/W setup time	tsu	40		ns	RS, R/W
RS, R/W hold time	th	10		ns	RS, R/W
Read data output delay	tD	60	120	ns	DB0-DB7
Read data hold time	tDH	20		ns	DB0-DB7
Write data setup time	tsu1	40		ns	DB0-DB7
Write data hold time	th1	10		ns	DB0-DB7

Bảng 3 Các thông số thời gian của LCD

3.6 Mã ASCII

LCD sử dụng mã ACSII để hiện thị các kí tự , các số, các ký hiệu, … có tổng ký tự là 256 ký tự.

Upper 4 Bits / Lower 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	p				ー	タ	ミ	α	p
xxxx0001	(2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q
xxxx0010	(3)		"	2	B	R	b	r			「	イ	ツ	メ	ß	θ
xxxx0011	(4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
xxxx0100	(5)		$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
xxxx0110	(7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w			ァ	キ	ヌ	ラ	q	π
xxxx1000	(1)		(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
xxxx1001	(2)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	⁻¹	y
xxxx1010	(3)		*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
xxxx1011	(4)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	ˣ	万
xxxx1100	(5)		,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
xxxx1101	(6)		−	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
xxxx1110	(7)		.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
xxxx1111	(8)		/	?	O	_	o	←			ッ	ソ	マ	°	ö	■

Bảng 4 Bảng mã ASCII

Trong các bài hiển thị các ký tự và các bài đếm thì hiển thị các con số từ 0 đến 9.

Tra bảng thì ta thấy các con số sẽ có mã từ 0x30 đến 0x39.

Để hiện thị số trên LCD từ số cần hiện thị thì chuyển sang số BCD rồi sau đó cộng thêm với 0x30 mới hiện thị được trên LCD

3.7 Vùng nhớ hiển thị DDRAM

Chip điều khiển LCD thì đã tích hợp đầy đủ bảng mã ASCII.

Muốn hiện thị ký tự nào thì tiến hành gửi mã của ký tự đó vào vùng nhớ DDRAM, thì vi điều khiển của LCD sẽ lấy mã ma trận của ký tự đó rồi đem ra hiện thị trên màn hình

Lệnh thiết lập vùng nhớ để hiện thị là set DDRAM addr, trong lệnh có 7 bit địa chỉ từ AC6 đến AC0

Vùng nhớ này chia ra làm 2:

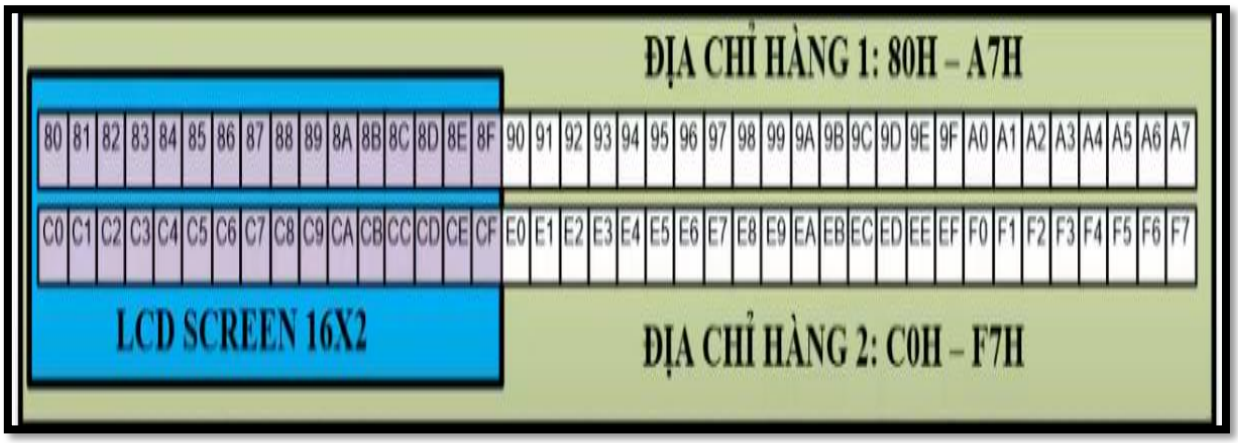
- Vùng 1: các ô nhớ có địa chỉ từ 0x80 đến 0xBF sẽ lữu trữ các ký tự hiện thị hang 1 của LCD.

- Vùng 2: Các ô nhớ có địa chỉ từ 0xC0 đến 0xFF sẽ lưu các ký tự hiện thị hang 2của LCD.

***Chi tiết: loại LCD 16x2**

- Hàng 1 chỉ hiện thị 16 kí tự có địa chỉ từ 0x80 đến x08F, từ 0x90 đến 0xBF ẩn bên trong , muốn hiện thị thì phải dùng lệnh dịch.

- Hàng 2 chỉ hiện thị 16 ký tự có địa chỉ từ 0xC0 đến 0xCF, từ 0xB0 đến 0xFF ẩn bên trong , muốn hiện thị thì phải dùng lệnh dịch.



Hình 3. 4 DDRAM MEMORY

Khi sử dụng LCD để hiện thị các ký tự thì ta phải khởi tạo LCD

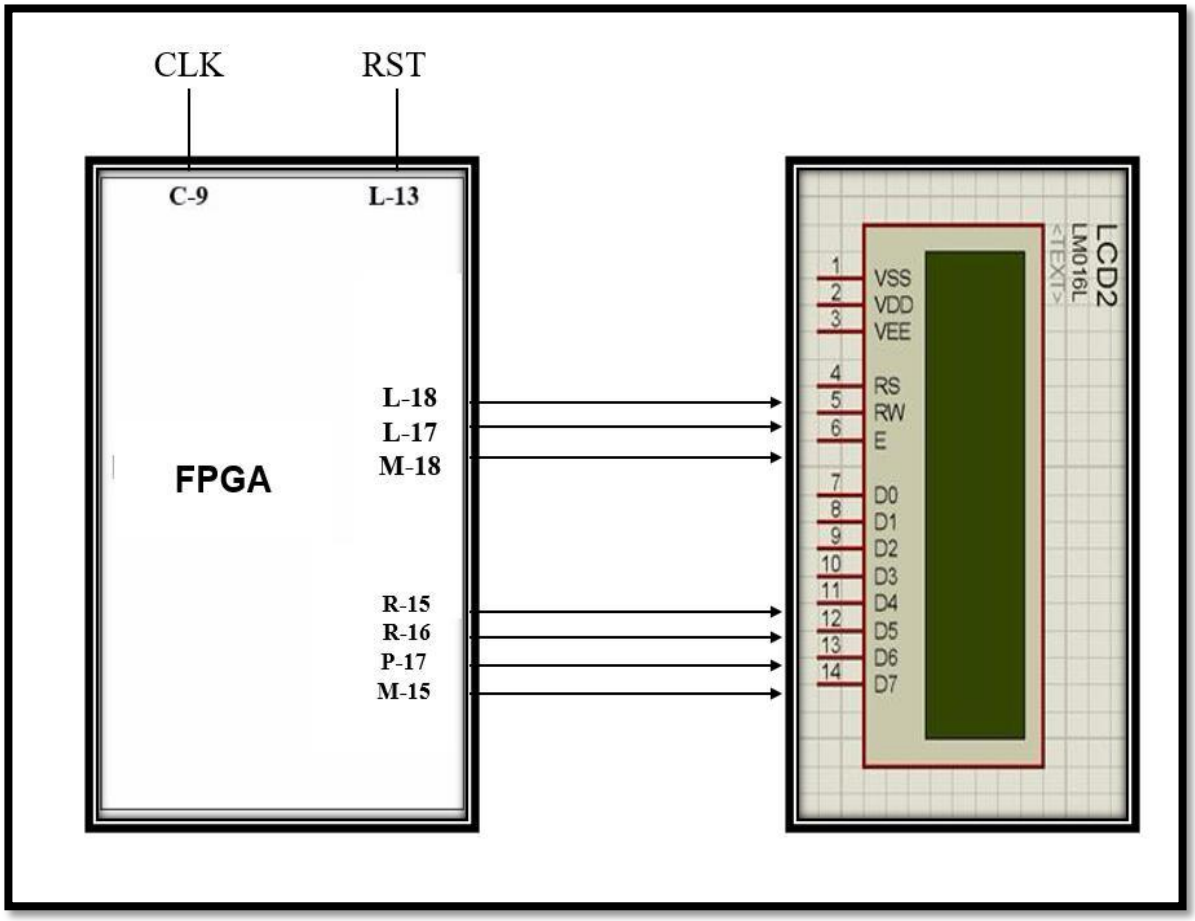
Trình tự khởi tạo LCD cơ bản theo chuẩn giao tiếp 8bit dữ liệu.

CHƯƠNG IV. ĐIỀU KHIỂN MÀN HÌNH LCD 1602 BẰNG FPGA (KIT SPARTAN-3E)

4.1 Thiết kế chương trình

4.1.1 Sơ đồ thiết kế khối điều khiển LCD1602A bằng FPGA

4.1.1.1 Sơ đồ giao diện điều khiển LCD ở chế độ 4 bit

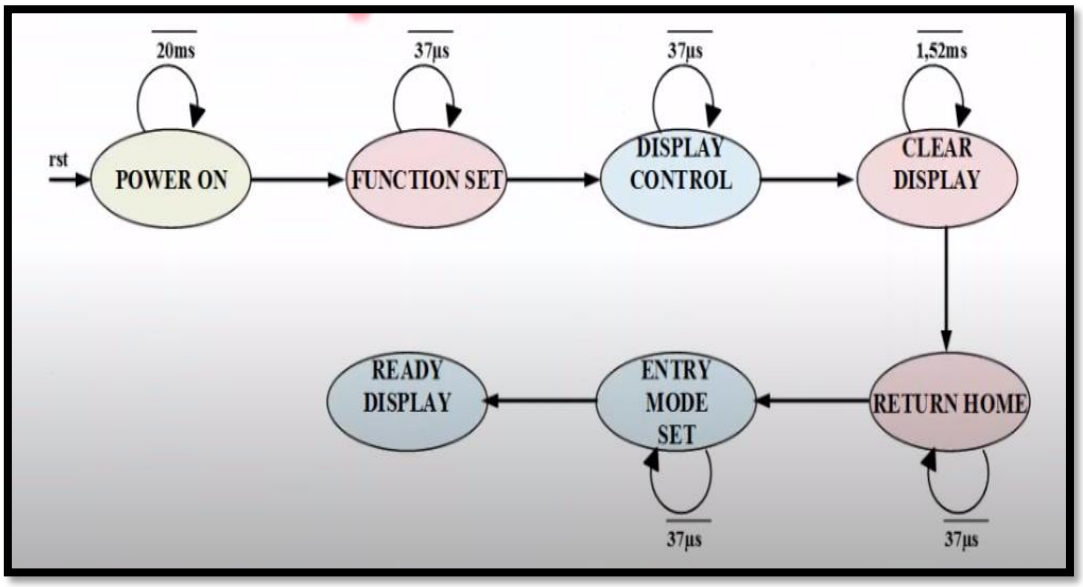


Hình 4. 1 Sơ đồ giao diện điều khiển LCD (4bit)

4.1.1.2 Sơ đồ khởi tạo LCD

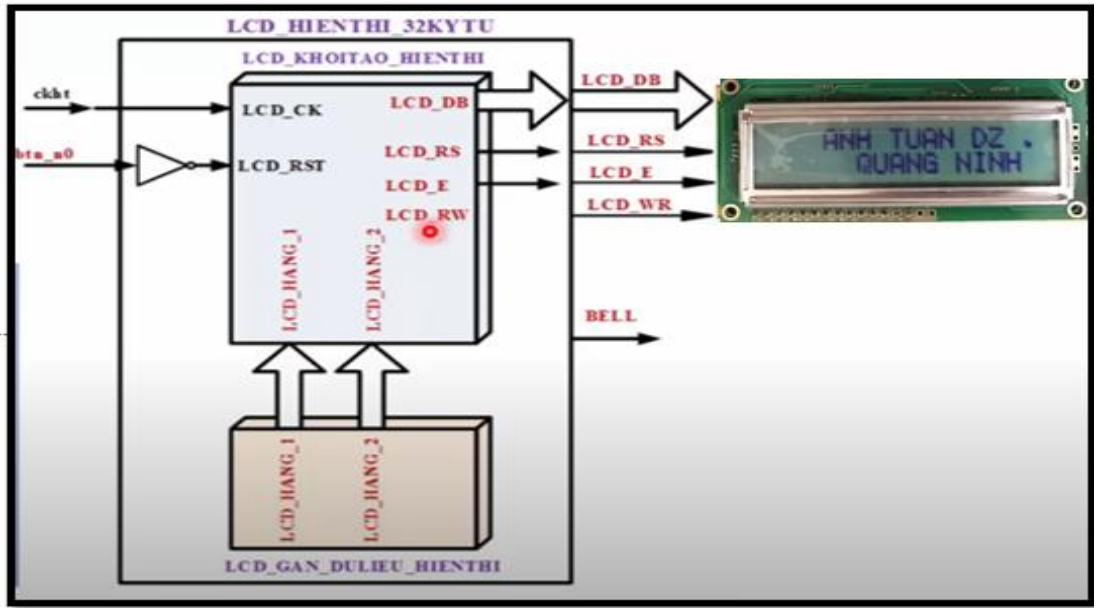
Khi sử dụng LCD để hiện thị các ký tự thì ta phải khởi tạo LCD

Trình tự khởi tạo LCD cơ bản theo chuẩn giao tiếp 8bit dữ liệu.



Hình 4. 2 Mô hình trạng thái

4.1.1.3 Sơ đồ khối hiển thị trên FPGA



Hình 4. 3 Sơ đồ khối hiển thị LCD trên FPGA

4.1.2 Chương trình điều khiển LCD hiển thị 32 kí tự(gán trực tiếp)

```
`timescale 1ns / 1ps
module TestLCD( clk, sf_e, e, rs, rw, d, c, b, a );
  (* LOC = "C9" *) input clk;
  (* LOC = "D16" *) output reg sf_e; // 1 LCD access (0 StrataFlash access)
  (* LOC = "M18" *) output reg e;
  (* LOC = "L18" *) output reg rs;
  (* LOC = "L17" *) output reg rw;
  (* LOC = "M15" *) output reg d;
  (* LOC = "P17" *) output reg c;
  (* LOC = "R16" *) output reg b;
  (* LOC = "R15" *) output reg a;

  reg [ 26 : 0 ] count = 0;    // 27-bit count, 0-(128M-1), over 2 secs
  reg [ 5 : 0 ] code;                  // 6-bit different signals to give out
  reg refresh;                                 // refresh LCD rate @ about 25Hz

  always @ (posedge clk) begin
        count <= count +1;

        case ( count[ 26 : 21 ] )    // as top 6 bits change
              0: code <= 6'h03;   // bat nguon init
              1: code <= 6'h03;   // bat lai mot lan
              2: code <= 6'h03;   // bat  lai nguon khi ban man LCD bat
```

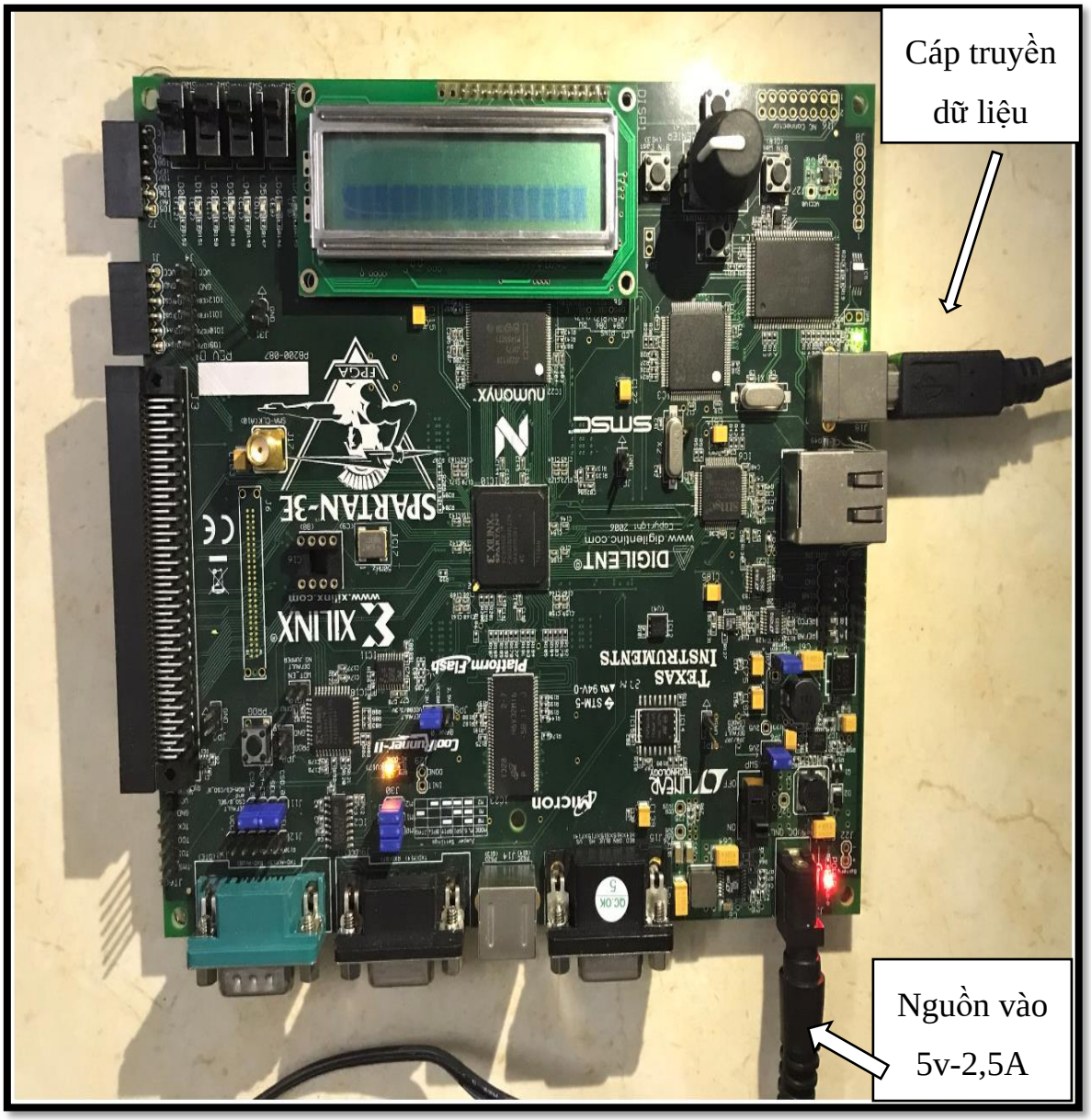
```
3: code <= 6'h02;                    // nhap nhay hien thi ki tu


4: code <= 6'h02;              // Function Set, upper nibble 0010
5: code <= 6'h08;   // lower nibble 1000 (10xx)// Entry Mode
6: code <= 6'h00;
7: code <= 6'h06;                    // Display On/Off
8: code <= 6'h00;
9: code <= 6'h0C;                    // Clear Display,
10: code <= 6'h00;
11: code <= 6'h01;            //nhap char
12: code <= 6'h22;           // khoang trang
13: code <= 6'h20;
14: code <= 6'h24;            // 'H' high nibble
15: code <= 6'h28;            // 'H' low nibble
16: code <= 6'h25;            // V
17: code <= 6'h26;
18: code <= 6'h24;            // K
19: code <= 6'h2B;
20: code <= 6'h25;            // T
21: code <= 6'h24;
22: code <= 6'h22;        // khoang trang
23: code <= 6'h20;
24: code <= 6'h24;            // M
25: code <= 6'h2D;
```

```
26: code <= 6'h26;      // a
27: code <= 6'h21;
28 : code <= 6'h27;             // chu T
29: code <= 6'h24;
30: code <= 6'h22;      // khoang trang
31: code <= 6'h20;
32: code <= 6'h24;          // M
33: code <= 6'h2D;
34: code <= 6'h26;      // a
35: code <= 6'h21;
36: code <= 6'h22;      // !
37: code <= 6'h21;
//  Set DD RAM (DDR) Address
// dua con tro xuong dong 2
38: code <= 6'b001100;
39: code <= 6'b000000;
40: code <= 6'h24;      // B
41: code <= 6'h22;
42: code <= 6'h26;      // a
43: code <= 6'h21;
44: code <= 6'h26;      // o
45: code <= 6'h2F;
46: code <= 6'h22;      // khoang trang
47: code <= 6'h20;
```

```
                48: code <= 6'h24;      // C
                49: code <= 6'h23;
                50: code <= 6'h26;      // a
                51: code <= 6'h21;
                52: code <= 6'h26;      // o
                53: code <= 6'h2F;
                54: code <= 6'h22;      // khoang trang
                55: code <= 6'h20;
                56: code <= 6'h24;          // D
                57: code <= 6'h24;
                58: code <= 6'h26;          // o
                59: code <= 6'h2F;
                60: code <= 6'h24;      // A
                61: code <= 6'h21;
                62: code <= 6'h26;      // n
                63: code <= 6'h2E;
    default: code <= 6'h10;    // the rest un-used time
        endcase
        refresh <= count[ 20 ]; // flip rate almost 25 (50Mhz / 2^21-2M)
            sf_e <= 1;
            { e, rs, rw, d, c, b, a } <= { refresh, code };
                end
endmodule
```

4.2 Mô phỏng hiện thị LCD trên SPARTAN-3E và hướng phát triển đề tài

4.2.1 Mạch thực tế



Hình 4. 4 KIT Spartan 3E khi đã khởi động

4.2.2 Kết quả chạy thử mạch

Hình 4. 5 Hiện thị kết quả trên KIT spartan 3e

KẾT LUẬN

Sau một quá trình nghiên cứu học hỏi , được sự giúp đỡ tận tình của thầy cô trong khoa Điện Tử -Viễn Thông nói chung , thầy Dương Phúc Phần nói riêng trong việc thực hiển đồ án của em và sau đây là kết quả em đã đạt được trong quá trình làm đồ án :

- ✓ Hiểu rõ về tổng quan FPGA và ngôn ngữ Verilog,Kit Spartan _3E, sử dụng thành thạo phần mềm ISE14.7.
- ✓ Chúng em có thể hiểu biết rõ hơn về cách giao tiếp của LCD kết nối với Spartan _3E .

Từ những kết quả đạt được chúng em mong muốn sẽ phát triển và khai thác thêm các đề tài lớn hơn đối với kit Spartan _3E, để có thể giải quyết những bài toán, và ứng dụng thực tế , thiết thực hơn trong đời sống ví dụ như giao tiếp bàn phím , kết nối với ,Mouse , VGA kết nối với Spartan_3E,UART,màn hình LED, OLED, xử lí ảnh…

Mặc dù em đã nỗ lực và cố gắng để hoàn thiện đồ án một cách tốt nhất, nhưng em vẫn không thể tránh khỏi những sai sót trong quá trình làm đồ án, rất mong nhận được sự góp ý của các thầy cô trong hội đồng và các bạn trong khoa để hoàn thiện đồ án tốt hơn… Cuối cũng, chúng em xin chân thành cảm ơn các thầy cô đã dạy bảo nhóm trong suốt quá trình học tập tại trường, đặc biệt là thầy Dương Phúc Phần và các thành viên ĐT1- khoa Điện tử Viễn thông đã giúp em hoàn thành tốt đồ án này.

Nhóm chúng em xin chân thành cảm ơn !

TÀI LIỆU THAM KHẢO

[1] Trịnh Quang Kiên, Lê Xuân Bằng (HĐ: PGS TS Đỗ Xuân Tiến) Thiết kế logic số - HVKTQS 2011

[2] IEEE Standard for Binary Floating-Point Arithmetic. ANSI/IEEE StandardNo. 754. American National Standards Institute - Washington, DC - 1985.

[3] Douglas L.Perry, Verilog Programming by Example McGraw - Hill, Fourth Edition -2008

[4] Vũ Đức Lương - Giáo trình ngôn ngữ mô tả phần cứng Verilog - ĐHQG TPHCM - Năm 2012.

[5] ThS Nguyễn Trọng Hải - Bài giảng Verilog - ĐHKTCN TPHCM - Năm 2005.

PHỤ LỤC

CHƯƠNG TRÌNH MỞ RỘNG HIỂN THỊ LCD 1602

lcd.v

```
module lcd (
input          clk,
output reg     lcd_rs,
output reg     lcd_rw,
output reg     lcd_e,
output reg [7:4] lcd_d,
output    [4:0] mem_addr,
input     [7:0] mem_bus
);
parameter      n = 24;
parameter      j = 17;
parameter      k = 11;
parameter      noop = 6'b010000;
reg       [n:0] count = 0;
reg       [5:0] lcd_state = noop;
reg             init = 1;
reg             row = 0;
assign mem_addr = {row, count[k+6:k+3
        always @ (posedge clk) begin
        count <= count + 1;
        if (init) begin
```

```
                case (count[j+7:j+2])
                        1: lcd_state <= 6'b000010;    // HAM FUNCTION SET
                        2: lcd_state <= 6'b000010;
                        3: lcd_state <= 6'b001000;
                        4: lcd_state <= 6'b000000;    // display on/off control
                        5: lcd_state <= 6'b001100;
                        6: lcd_state <= 6'b000000;    // display clear
                        7: lcd_state <= 6'b000001;
                        8: lcd_state <= 6'b000000;    // entry mode set
                        9: lcd_state <= 6'b000110;
                        10: begin init <= ~init; count <= 0; end
                endcase
        {lcd_e,lcd_rs,lcd_rw,lcd_d[7:4]}<={^count[j+1:j+0]&~lcd_rw,lcd_state;
                                        end else begin
                case (count[k+7:k+2])
                        32: lcd_state <= {3'b001,~row,2'b00
                        33: lcd_state <= 6'b000000;
                        34: begin count <= 0; row <= ~row; end

        default:lcd_state<={2'b10,~count[k+2]?mem_bus[7:4]:    mem_bus[3:0]};
endcase
        {lcd_e,lcd_rs,lcd_rw,lcd_d[7:4]}<={^count[k+1:k+0]&~lcd_rw,lcd_state
};
        end
        end
```

```
endmodule
```

top.v

```
module top(
  input clk, // khai bao chan clk
  output lcd_rs, // khai bao chan rs
  output lcd_rw, // khai bao chan rw
  output lcd_e,  // khai bao chan e
  output [11:8] sf_d  //gui 4 bit cao
  );
  wire [7:0] char_mem_bus; // kieu du lieu ket noi 2 cong hay cac module( khai bao duong dia chi 8bit)
  wire [4:0] char_mem_addr; // khai bao 4 duong dia du lieu
  char_mem char_mem (char_mem_addr, char_mem_bus);
  lcd lcd (clk, lcd_rs, lcd_rw, lcd_e, sf_d, char_mem_addr, char_mem_bus);
endmodule
```

char_mem.v

```
module char_mem(
   input  [4:0] addr, // dau vao 4
   output [7:0] bus  // dau ra 7 duong dia chi
   );
  parameter LINES = 2;   // so dong
  parameter CHARS_PER_LINE = 16; // ki tu 1 dong
  parameter BITS_PER_CHAR = 8; // so bit 1  ki tu
  parameter STR_SIZE=LINES*CHARS_PER_LINE* BITS_PER_CHAR;
```

```
parameter [0:STR_SIZE-1] str = "HVKT-Mat Ma!    Bao cao do an1!";

assign bus = str[{addr[4:0], 3'b000}+:8];

  endmodule
```

FILE GÁN CHÂN CHO KIT SPARTAN 3E

spartan3e.ucf

```
NET "LCD_E"    LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

NET "LCD_RS"   LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

NET "LCD_RW"   LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

NET "SF_D<8>"  LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

NET "SF_D<9>"  LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

NET "CLK"      LOC = "C9"  | IOSTANDARD = LVCMOS33 ;
```