BỘ GIÁO DỤC VÀ ĐÀO TẠO

HUTECH
Đại học Công nghệ Tp.HCM

THỰC HÀNH
THIẾT KẾ HỆ THỐNG SỐ

Biên soạn: | **TS. Võ Đình Tùng**

THỰC HÀNH THIẾT KẾ HỆ THỐNG SỐ

Ấn bản 2015

MỤC LỤC

MỤC LỤC I
HƯỚNG DẪN III

BÀI 1: ABEL 1

1.1 GIỚI THIỆU 1
1.1.1 Ngôn ngữ lập trình ABEL 1
1.1.2 Các bước chuẩn bị 2
1.2 SOẠN THẢO CHƯƠNG TRÌNH & BIÊN DỊCH 3
1.2.1 Soạn thảo chương trình 3
1.2.2 Biên dịch chương trình ABEL 5
1.3 NẠP CHƯƠNG TRÌNH 5
1.4 CÁC BÀI THÍ NGHIỆM 7
TÓM TẮT 12
BÀI TẬP 12

BÀI 2: BOARD DE2 & PHẦN MỀM THIẾT KẾ TRÊN FPGA 14

2.1 GIỚI THIỆU BOARD DE2 14
2.1.1 Các thành phần boardard DE2 14
2.1.2 Một số ứng dụng tiêu biểu của boardard DE2 15
2.2 CÁC PHẦN MỀM THIẾT KẾ TRÊN FPGA 16
2.3 SOẠN THẢO CHƯƠNG TRÌNH BIÊN DỊCH 17
2.3.1 Tạo Project 17
2.3.2 Soạn thảo chương trình & biên dịch 18
2.4 NẠP CHƯƠNG TRÌNH LÊN BOARD DE2 20
2.4.1 Cấu hình chân 20
2.4.2 Nạp lên board DE2 23
2.5 MÔ PHỎNG CHƯƠNG TRÌNH 26
2.5.1 Tạo file mô phỏng 26
2.5.2 Chọn chế độ mô phỏng 26
2.5.3 Chọn file mô phỏng 27
2.5.4 Tiến hành mô phỏng 27
TÓM TẮT 28
CÂU HỎI ÔN TẬP 29

BÀI 3: NGÔN NGỮ LẬP TRÌNH VERILOG 30

3.1 GIỚI THIỆU 30
3.2 NGÔN NGỮ LẬP TRÌNH VERILOG 31
3.2.1 Cấu trúc chương trình Verilog 31
3.2.2 Các bài thí nghiệm verilog 32
TÓM TẮT 36
CÂU HỎI ÔN TẬP 37

BÀI 4: NGÔN NGỮ LẬP TRÌNH VHDL 39

4.1 GIỚI THIỆU 39
4.2 NGÔN NGỮ LẬP TRÌNH VHDL 39
4.2.1 Cấu trúc chương trình VHDL 39
4.2.2 Các bài thí nghiệm VHDL 40
TÓM TẮT 44
CÂU HỎI ÔN TẬP 45

BÀI 5: THIẾT KẾ FPGA THÔNG QUA SƠ ĐỒ KHỐI 47

5.1 GIỚI THIỆU 47
5.2 THIẾT KẾ FPGA THÔNG QUA SƠ ĐỒ KHỐI (SCHEMATIC) 47
5.2.1 Tạo sơ đồ mạch 47
5.2.2 Mô phỏng chương trình 50
5.2.3 Gán chân và nạp chương trình 50
TÓM TẮT 51
CÂU HỎI ÔN TẬP 52

BÀI 6: MÁY TRẠNG THÁI 53

6.1 MÁY TRẠNG THÁI 53
6.1.1 Máy trạng thái kiểu MOORE 53
6.1.2 Máy trạng thái kiểu MEALY 54
6.2 GIẢN ĐỒ TRẠNG THÁI 54
6.2.1 Giản đồ trạng thái kiểu MOORE 54
6.2.2 Giản đồ trạng thái kiểu MEALY 55
6.3 LẬP TRÌNH FPGA THÔNG QUA MÁY TRẠNG THÁI 55
6.3.1 Giới thiệu 55
6.3.2 Mở của sổ soạn thảo máy trạng thái 56
6.3.3 Qui trình tạo máy trạng thái 56
6.3.4 Tạo file ngôn ngữ mô tả phần cứng từ máy trạng thái 60
TÓM TẮT 61

CÂU HỎI ÔN TẬP 62
TÀI LIỆU THAM KHẢO 63

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Thí nghiệm thiết kế hệ thống số nhằm mục đích hỗ trợ các bạn sinh viên trong việc tiếp xúc với ngôn ngữ đặc tả phần cứng.

Thực hiện việc lập trình bằng các ngôn ngữ mô tả phần cứng khác nhau: ABEL, VHDL, VERILOG, SHEMATIC, State Machine ….

Thực thi các chương trình ngay trên thiết bị phần cứng.

NỘI DUNG MÔN HỌC

- Bài 1. Ngôn ngữ lập trình ABEL.
- Bài 2: Boardard DE2 và phần mềm thiết kế trên FPGA.
- Bài 3: Ngôn ngữ lập trình Verilog và VHDL.
- Bài 4: Thiết kế FPGA thông qua sơ đồ khối.
- Bài 5: Thiết kế FPGA thông qua máy trạng thái.

KIẾN THỨC TIỀN ĐỀ

Môn học thí nghiệm thiết kế hệ thống số đòi hỏi sinh viên có nền tảng về kỹ thuật số cơ bản và đã học các ngôn ngữ đặc tả phần cứng.

YÊU CẦU MÔN HỌC

Người học phải dự học đầy đủ các buổi thí nghiệm và làm bài tập đầy đủ ở nhà.

CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Để học tốt môn này, người học cần ôn tập các bài đã học, làm đầy đủ bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học.

Đối với mỗi bài học, người học đọc trước mục tiêu và tóm tắt bài học, sau đó tiến hành thực hiện các bài thí nghiệm.

Sau mỗi bài học tiến hành làm các bài tập.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

Môn học được đánh giá: Thực hiện kiểm tra sau khi học mỗi bài. Điểm của môn học là trung bình cộng của các bài đã học.

BÀI 1: ABEL

Sau khi học xong bài này, học viên có thể:

- *Hiểu được công nghệ PLD;*
- *Cách viết chương trình ABEL cho PLD;*
- *Cách lập trình PLD bằng ABEL.*

1.1 GIỚI THIỆU

1.1.1 Ngôn ngữ lập trình ABEL

ABEL(Advanced Boardolean Equation Language) là một trong những ngôn ngữ lập trình rất mạnh cho PLD. Trình biên dịch của ABEL có khả năng mô phỏng và tạo file cầu chì (fuse map) cho PLD

Khung chương trình ngôn ngữ ABEL

Module Module_name
Title 'String'
Device_indentifier **Device** device_type
@alternate " có thể có hoặc không
Pin **declarations** "comment
Other declarations
Equations

Test vectors
End module_name

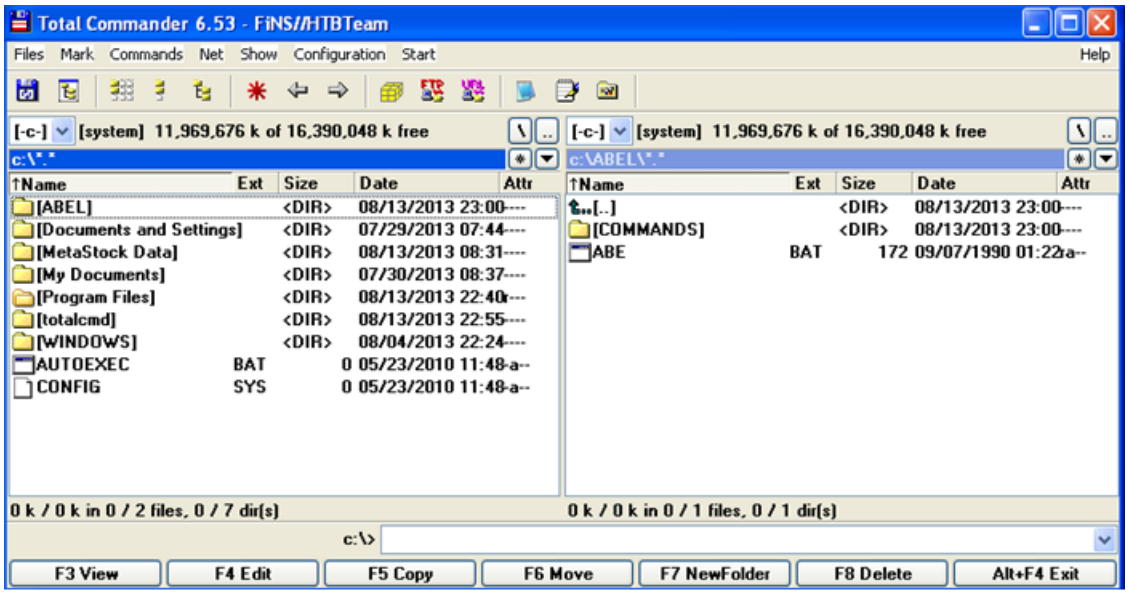
Ngôn ngữ lập trình ABEL hỗ trợ cho nhiều hình thức ngõ vào:

- Mô tả bằng phương trình đại số BOARDOLEAN (equations)
- Mô tả bằng bảng sự thật (true table)
- Mô tả bằng sơ đồ trạng thái (state diagram)

1.1.2 Các bước chuẩn bị

Để viết chương trình cho PLD dùng ngôn ngữ ABEL cần chuẩn bị:

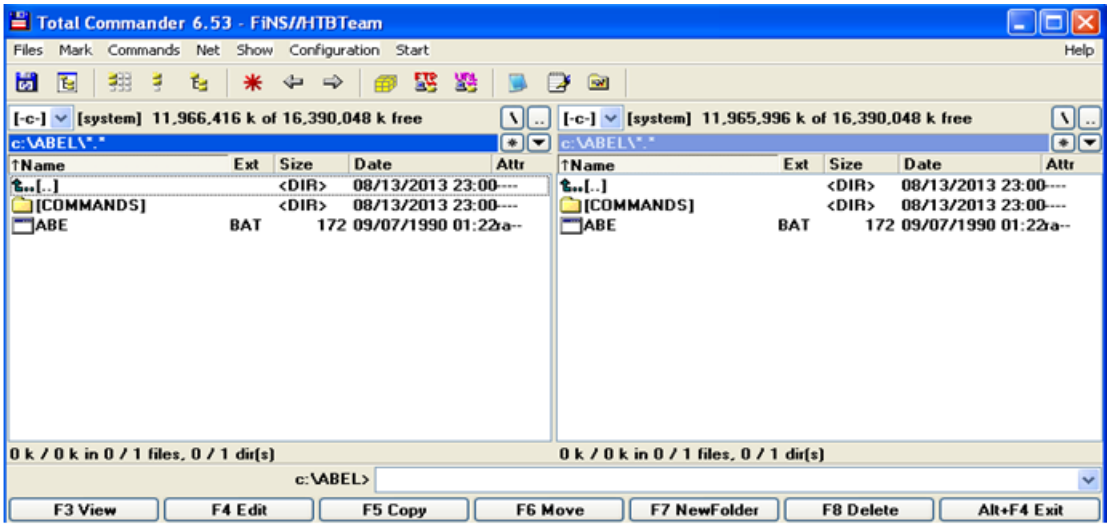
Bước 1: Cài đặt Total commander.



Hình 1.1: Cửa sổ Total commander

Bước 2: Tạo thư mục ABEL tại thư mục gốc ổ đĩa C.

Bước 3: Chép 2 file COMMANDS và ABE.BAT vào thư mục ABEL.

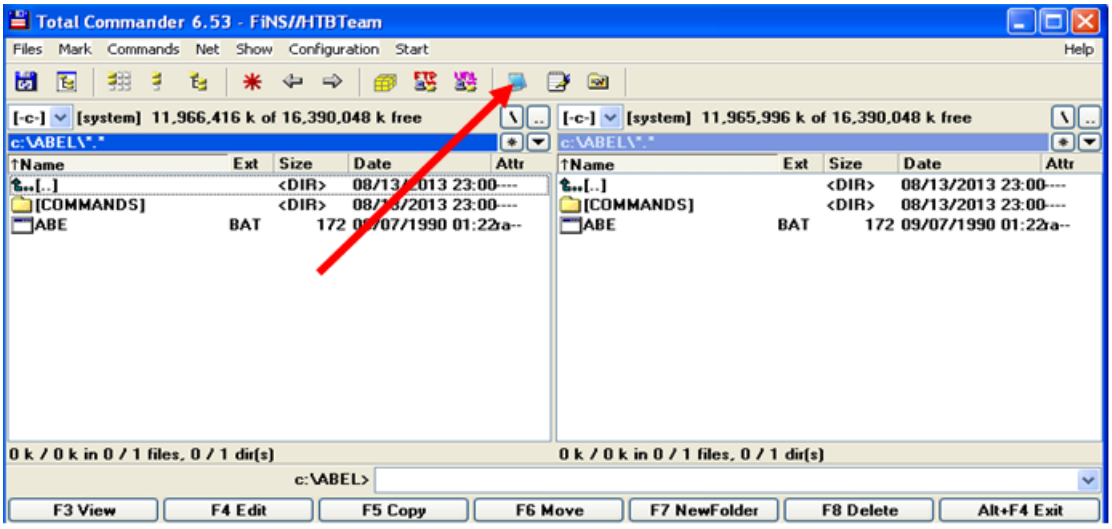


Hình 1.2: Thư mục ABEl được tạo

1.2 SOẠN THẢO CHƯƠNG TRÌNH & BIÊN DỊCH

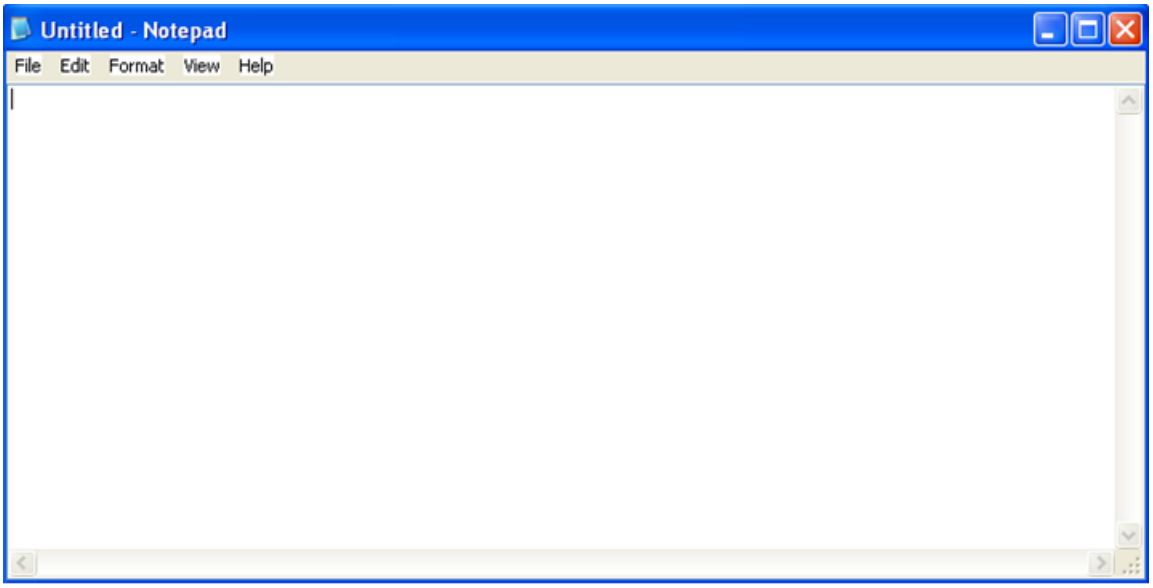
1.2.1 Soạn thảo chương trình

Bước 1: Trong cửa sổ Total commander nhấn vào biểu tượng notepad



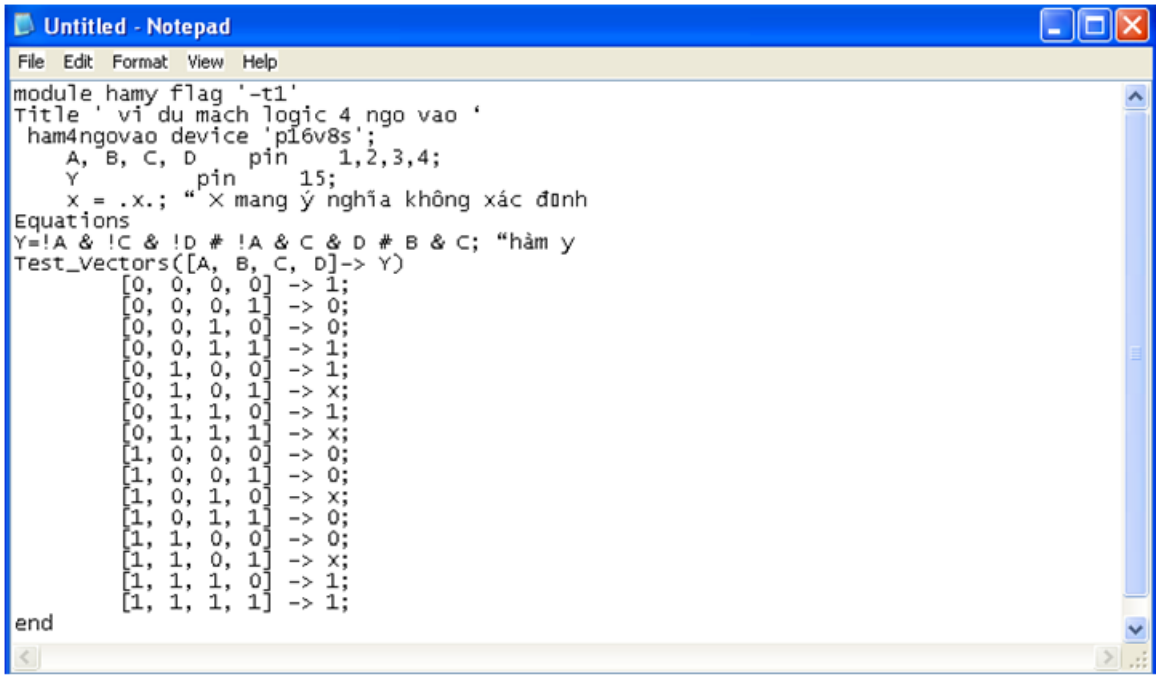
Hình 1.3: Mở cửa sổ soạn thảo chương trình ABEL

Cửa sổ soạn thảo chương trình xuất hiện



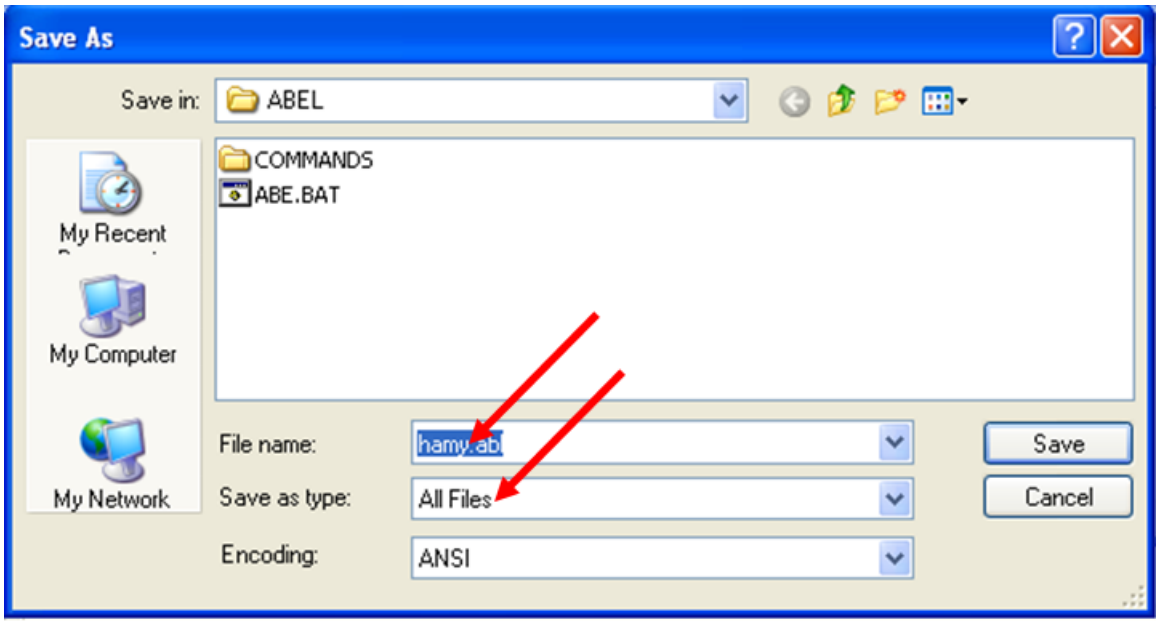
Hình 1.4: Cửa sổ soạn thảo chương trình ABEL

Bước 2: Nhập chương trình. Dưới dây là nội dung chương trình viết cho hàm Y với 4 biến vào



Hình 1.5: Chương trình ABEL điển hình

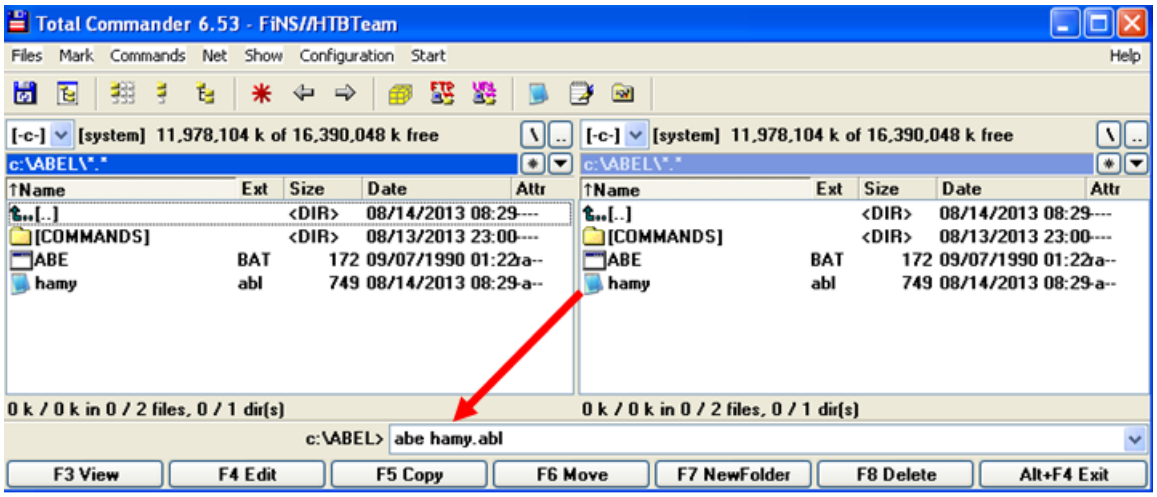
Bước 3: Sau khi viết xong ta phải lưu lại file có tên trùng với module_name, file có phần mở rộng là .ABL vào trong thu mục ABEL.



Hình 1.6: Hộp thoại lưu file

1.2.2 Biên dịch chương trình ABEL

Tại dấu nhắc đánh lệnh ABE tên file rồi enter.



Hình 1.7: Biên dịch chương trình

Chương trình sẽ được biên dịch. Nếu chương trình không bị lỗi thì sau khi biên dịch sẽ tạo ra các file: hamy.lst, hamy.doc, hamy.sim, hamy.jed

+ hamy.lst : Chứa lỗi
+ hamy.doc : mô tả chân linh kiện
+ hamy.sim : mô phỏng
+ hamy.jed : dùng để nạp vào PLD

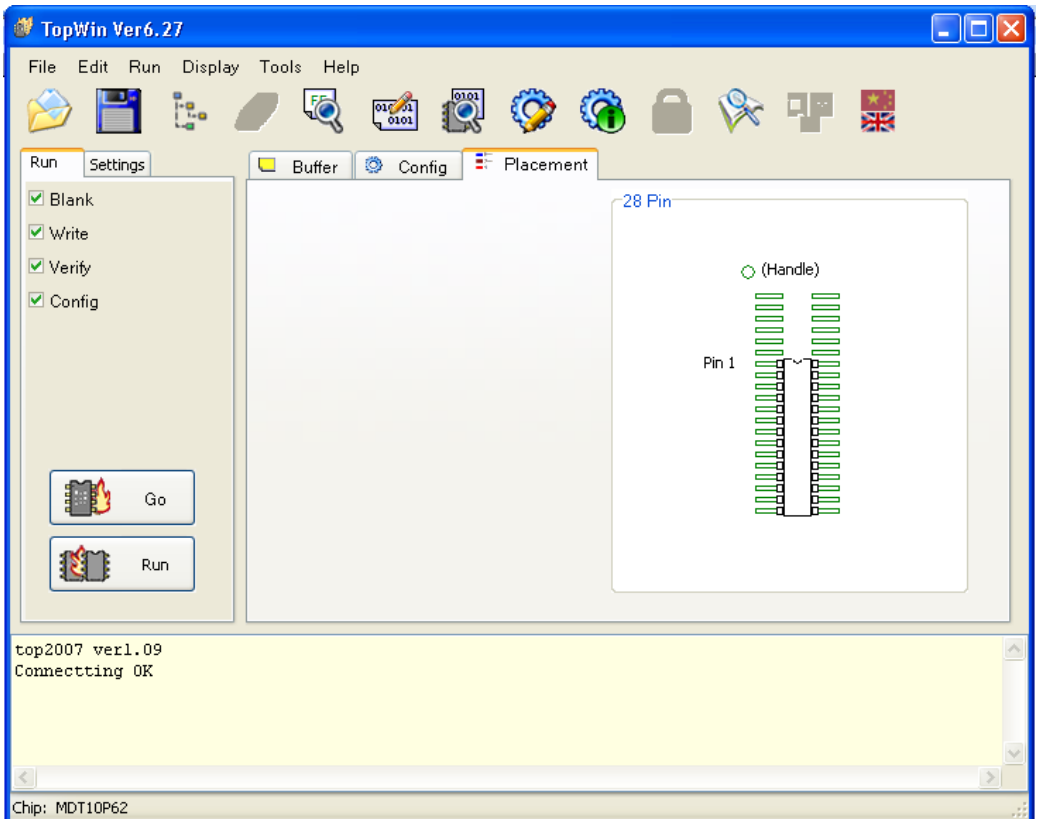
Sau khi biên dịch, ở file có phần mở rộng .LST để xem có lỗi hay không. Nếu có thì tiến hành sửa lỗi trên file gốc (.ABL) rồi thực hiện biên dịch lại. Lập lại quá trình này cho đến khi nào không còn lỗi trong file.list.

1.3 NẠP CHƯƠNG TRÌNH

Dùng kit nạp Top Programmer để lập trình cho PLD:

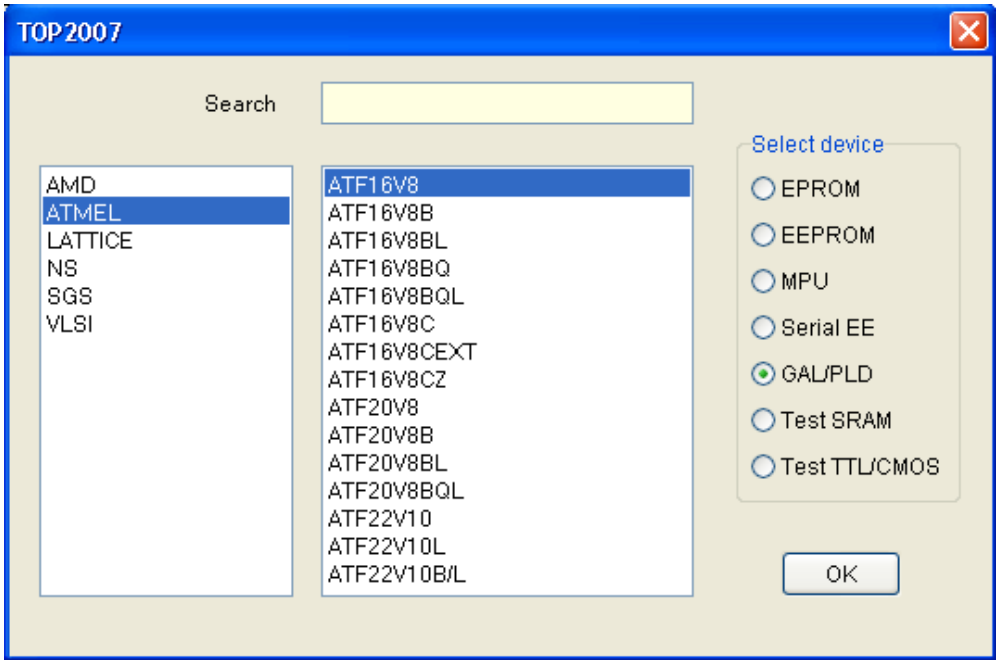
Bước 1: Kết nối kit nạp Top Programmer với máy tính qua cổng USB.

Bước 2: Mở chương trình Topwin6: bằng cách nhấp Double vào shortcut Topwin6. Cửa sổ lập trình xuất hiện. Chọn tùy chọn Placement



Hình 1.8: Cửa sổ lập trình PLD

Bước 3: Chọn chip: Vào Run → Select chip → Cửa sổ chọn chip xuất hiện



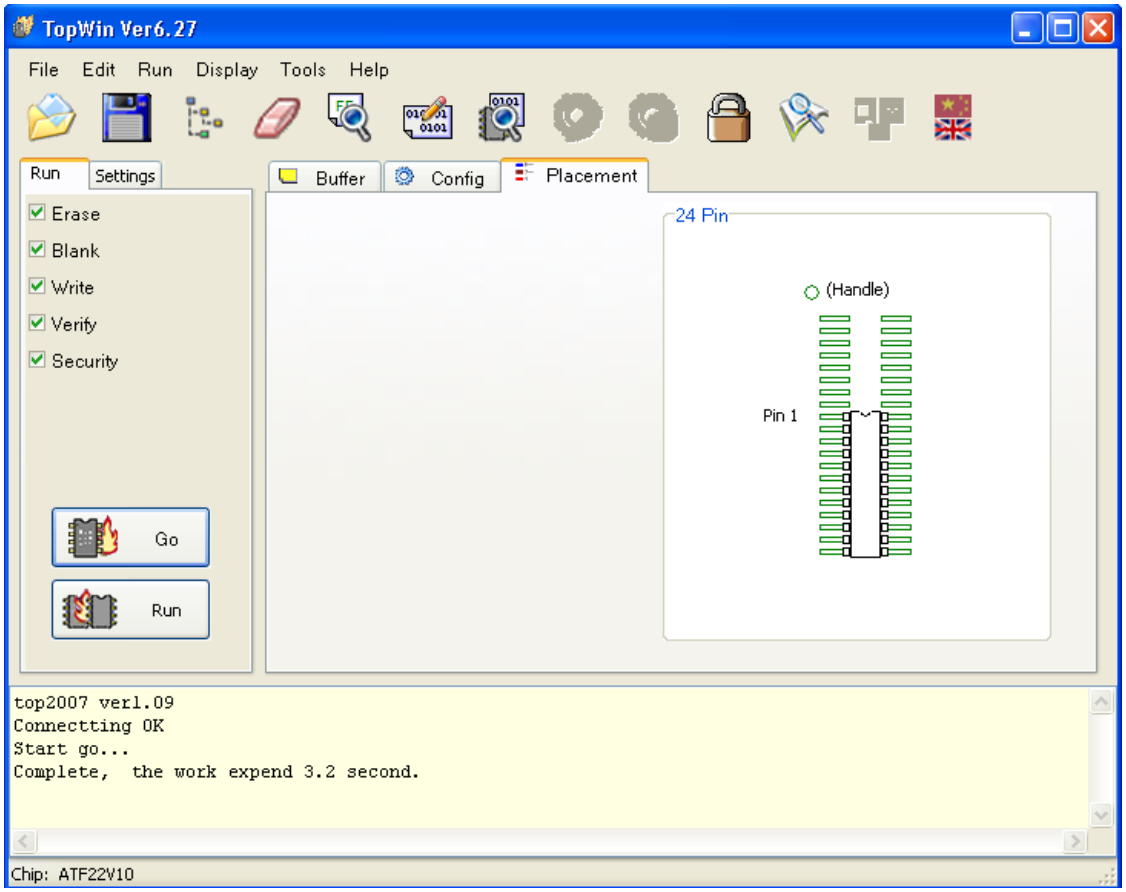
Hình 1.9: Cửa sổ chọn chip

Trong cửa sổ chọn chip: Chọn GAL/PLD trong khung Select device. Chọn đúng chip cần nạp → OK.

Lắp chip cần nạp vào kit nạp Top Programmer theo vị trí chỉ dẫn trong cửa sổ.

Bước 4: Chọn tập tin cần nạp: vào **file** → **Open**, chọn chỉ vị trí file cần nạp (tập tin có hần mở rộng .JED → Open.

Bước 5: Để lập trình chọn **Go**. Quá trình lập trình diễn ra. Sau khi lập trình xong sẽ báo là complete.



Hình 1.10: Lập trình hoàn tất

1.4 CÁC BÀI THÍ NGHIỆM

Bài 1: Viết chương trình ABEL thực hiện bảng sự thật sau sử dụng GAL 16v8s.

A	B	C	Y
0	0	0	1
0	0	1	x
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	x
1	1	0	0
1	1	1	1

Chương trình ABEL

```
module table flag '-t1'
title ' ham y 3 ngo vao dung bang su that'
trutable device 'p16v8s';
    A, B, C pin 1, 2, 3;
    Y pin 15;
    X = .X.;
truth_table ([A, B, C] -> Y)
    [0, 0, 0] -> 1;
    [0, 0, 1] -> X;
    [0, 1, 0] -> 0;
    [0, 1, 1] -> 1;
    [1, 0, 0] -> 1;
    [1, 0, 1] -> X;
    [1, 1, 0] -> 0;
    [1, 1, 1] -> 1;
```

```
test_vectors ([A, B, C] -> Y)
    [0, 0, 0] -> 1;
    [0, 0, 1] -> X;
    [0, 1, 0] -> 0;
    [0, 1, 1] -> 1;
    [1, 0, 0] -> 1;
    [1, 0, 1] -> X;
    [1, 1, 0] -> 0;
    [1, 1, 1] -> 1;
end
```

Bài 2: Viết chương trình ABEL thực hiện bảng mạch giải mã 74138 sử dụng GAL 16v8.

```
module OL74138 flag '-t1'
Title 'giai ma 3 sang 8 '
L74138a device 'P16V8S';
"Input pins
A2, A1, A0 pin 3, 2, 1;
G1, !G2A, !G2B pin 4, 5, 6;
"Output pins
!Y7,!Y6,!Y5,!Y4        pin 12,13,14,15;
!Y3,!Y2,!Y1,!Y0        pin 16,17,18,19;
"Constant expression
ENB = G1 & !G2A & !G2B;
  select = [A2,A1,A0];
  DATAOUT = [!Y7,!Y6,!Y5,!Y4,!Y3,!Y2,!Y1,!Y0];
Equations
```

```
  Y0 = (select == 0) & ENB;
  Y1 = (select == 1) & ENB;
  Y2 = (select == 2) & ENB;
  Y3 = (select == 3) & ENB;
  Y4 = (select == 4) & ENB;
  Y5 = (select == 5) & ENB;
  Y6 = (select == 6) & ENB;
  Y7 = (select == 7) & ENB;
test_vectors
  ([G1,G2A,G2B,select]->DATAOUT)
  [0,.x.,.x.,.x.] -> ^B11111111;
  [.x.,.x.,1,.x.] -> ^B11111111;
  [.x.,1,.x.,.x.] -> ^B11111111;
  [1,0,0,0] -> ^B11111110;
  [1,0,0,1] -> ^B11111101;
  [1,0,0,2] -> ^B11111011;
  [1,0,0,3] -> ^B11110111;
  [1,0,0,4] -> ^B11101111;
  [1,0,0,5] -> ^B11011111;
  [1,0,0,6] -> ^B10111111;
  [1,0,0,7] -> ^B01111111;
  [1,0,0,0] -> ^B11111110;
end
```

Bài 3: Viết chương trình ABEL thực hiện việc đếm lên đồng bộ mod16 dùng GAL16V8. Với CK tác động cạnh lên, CL tác động mức thấp

```
module count8db flag '-t1'
title 'mach dem dong board mod8'
Count8 device 'P16V8R';
    CK,!CL pin 1,2 ;
    Q2,Q1,Q0 pin 18,17,16;
    Count = [Q2,Q1,Q0];
    C,H,L = .C.,1,0 ; "Xung CK tac dong canh len
Equations
    Q0 := !Q0 & CL;
    Q1 := (Q1$Q0) & CL;
    Q2 := (Q2$(Q1&Q0))&CL;
Test_vectors([CK,CL] -> Count)
    [C,L] -> 0 ;
    [C,H] -> 1 ;
    [C,H] -> 2 ;
    [C,H] -> 3 ;
    [C,H] -> 4 ;
    [C,H] -> 5 ;
    [C,H] -> 6 ;
    [C,H] -> 7 ;
    [C,H] -> 0 ;
END
```

TÓM TẮT

Trong bài này, học viên làm quen với ngôn ngữ ABEL lập trình cho PLD:

Ngôn ngữ ABEL hổ trợ nhiều hình thức ngõ vào: Phương trình đại số BOARDOL, bảng sự thật, sơ đồ trạng thái.

Cách soạn thảo, biên dịch chương trình viết bằng ngôn ngữ *ABEL*

Cách lập trình cho PLD.

CÂU HỎI ÔN TẬP

Câu 1: Viết chương trình ABEL thực hiện cổng logic cơ bản

Câu 2: Viết chương trình ABEL thực hiện yêu cầu sau: Một hàm Boardol Y của 3 biến vào A, B, C. Ngõ ra Y = 1 khi A = B ≠ C.

Câu 3: Viết chương trình ABEL thực hiện các mạch mã hóa, giải mã, dồn kênh, phân kênh.

Câu 4: Viết chương trình ABEL thực hiện mạch đếm đồng bộ với số MOD bất kỳ.

Câu 5: Cho hàm Boardolean $Y = \sum_{ABCD}(0,1,3,5,12,14)$

Viết chương trình Abel mô tả hàm Y.

BÀI 2: BOARD DE2 & PHẦN MỀM THIẾT KẾ TRÊN FPGA

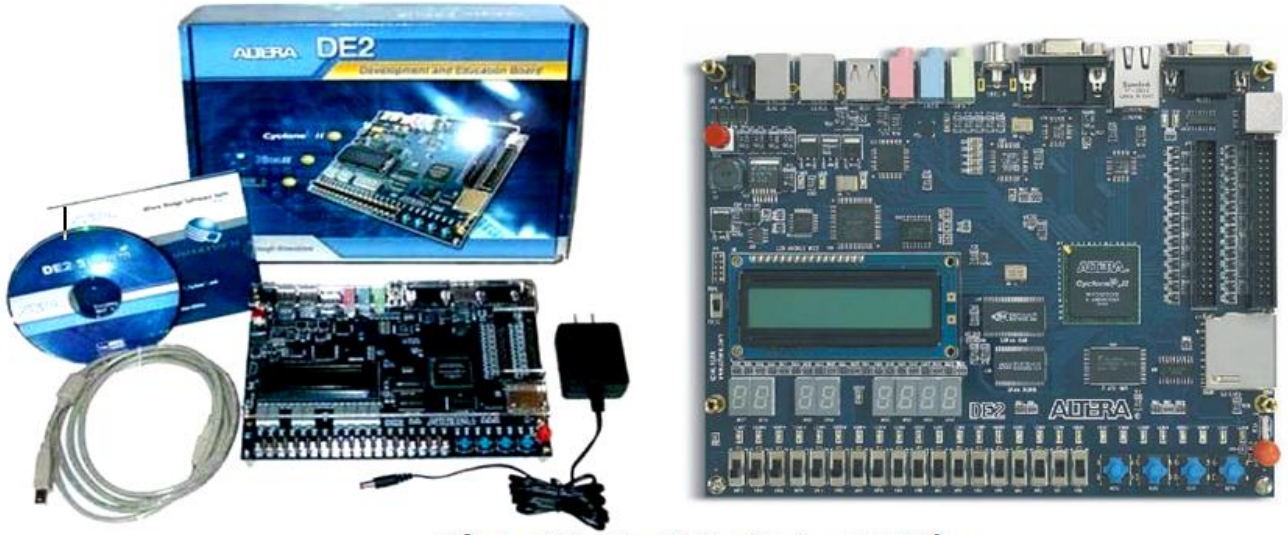
Sau khi học xong bài này, học viên có thể:

- *Hiểu được công nghệ FPGA;*
- *Sử dụng được phần mềm Quartus II để lập trình cho FPGA;*
- *Sử dụng được board thí nghiệm DE2.*

2.1 GIỚI THIỆU BOARD DE2

Bộ công cụ DE2 được thiết kế để nhằm áp dụng nhu cầu học tập, giảng dạy và nghiên cứu các lĩnh vực thiết kế luận lý, thiết kế hệ thống số, và FPGA. Board mạch DE2 là board mạch thử nghiệm hoạt động riêng.

2.1.1 Các thành phần boardard DE2

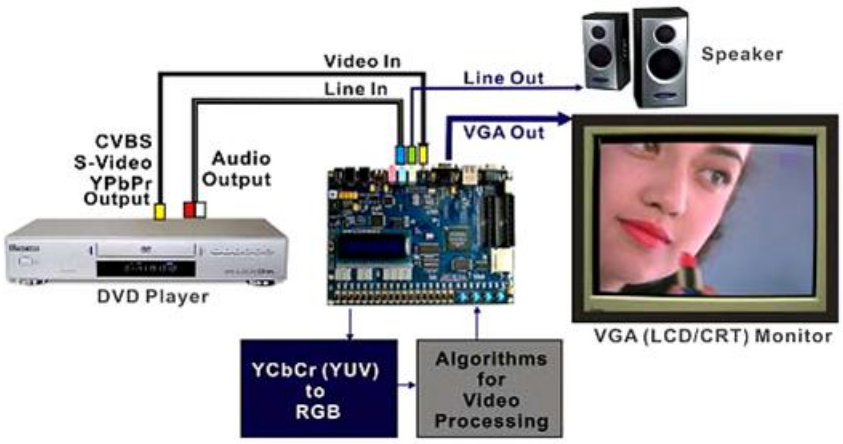


Hình 2.1: Bo DE2 và các phụ kiện

- FPGA:
    - Vi mạch FPGA Altera Cyclone II 2C35.
    - Vi mạch Altera Serial Configuration – EPCS16.
- Các thiết bị xuất nhập:
    - USB Blaster cho lập trình và điểu khiển API của người dung; hỗ trợ cả 2 chế độ lập trình JTAG và AS.
    - Bộ điều khiển Cổng 10/100 Ethernet.
    - Cổng VGA-out.
    - Bộ giải mã TV và cổng nối TV-in.
    - 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.
    - Cổng giao tiếp hồng ngoại.
- Bộ nhớ:
    - SRAM 512-Kbyte.
    - SDRAM 8-Mbyte.
    - Bộ nhớ cực nhanh 4-Mbyte (1 số mạch là 1-Mbyte).
    - Khe SD card.
- Switch, các đèn led, LCD, xung clock
    - 4 nút nhấn, 18 nút gạt.
    - 18 LED đỏ, 9 LED xanh, 8 Led 7 đoạn
    - LCD 16x2
    - Bộ dao động 50-MHz và 27-MHz cho đồng hồ nguồn.

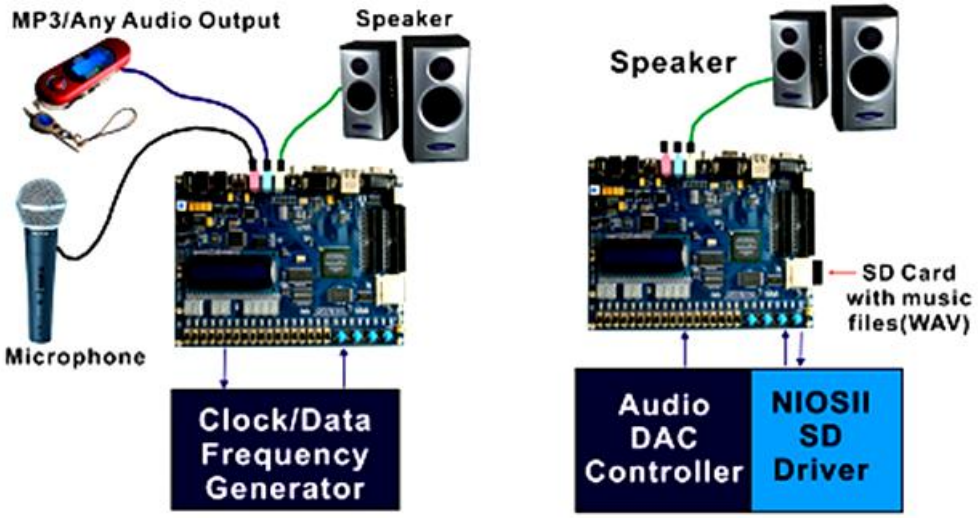
2.1.2 Một số ứng dụng tiêu biểu của boardard DE2

- Ứng dụng làm Tivi boardx



Hình 2.2: Tivi box

- Máy hát Karaoke & máy chơi nhạc SD



Hình 2.3: Máy hát Karaoke & máy chơi nhạc từ card SD

2.2 CÁC PHẦN MỀM THIẾT KẾ TRÊN FPGA

Bộ phần mềm thiết kế đi kèm với boardard DE2 bao gồm 2 đĩa: Quartus 2 và Nios 2 Integrated Development Environment (IDE).

Nios II, môi trường phát triển tích hợp của họ Nios II (IDE), nó là công cụ phát triển chủ yếu của họ vi xử lý Nios II. Phần mềm sẽ là môi trường cung cấp khả năng chỉnh sửa, xây dựng, debug và mô tả sơ lược về chương trình. IDE còn cho

phép tạo các chương trình từ đơn nhiệm đến các chương trình phức tạp dựa trên một hệ điều hành thời gian thực và các thư viện middleware.

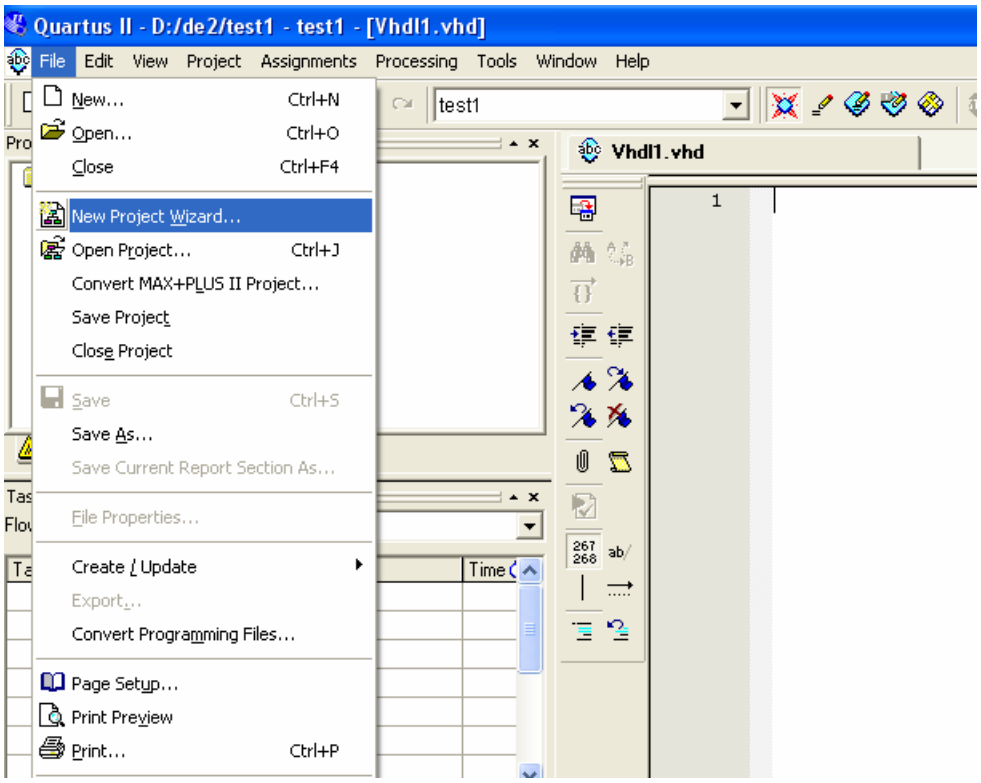
Quartus II là phần mềm hổ trợ tất cả mọi quá trình thiết kế một mạch logic, bao gồm quá trình soạn thảo chương trình, tổng hợp, placement và routing (sắp xếp và chạy dây), mô phỏng, và lập trình lên board DE2. Việc soạn thảo chương trình bằng quartus có thể tiến hành:

- Viết bằng ngôn ngữ Verilog hoặc VHDL
- Vẽ sơ đồ Schematic
- Máy trạng thái

2.3 SOẠN THẢO CHƯƠNG TRÌNH BIÊN DỊCH

2.3.1 Tạo Project

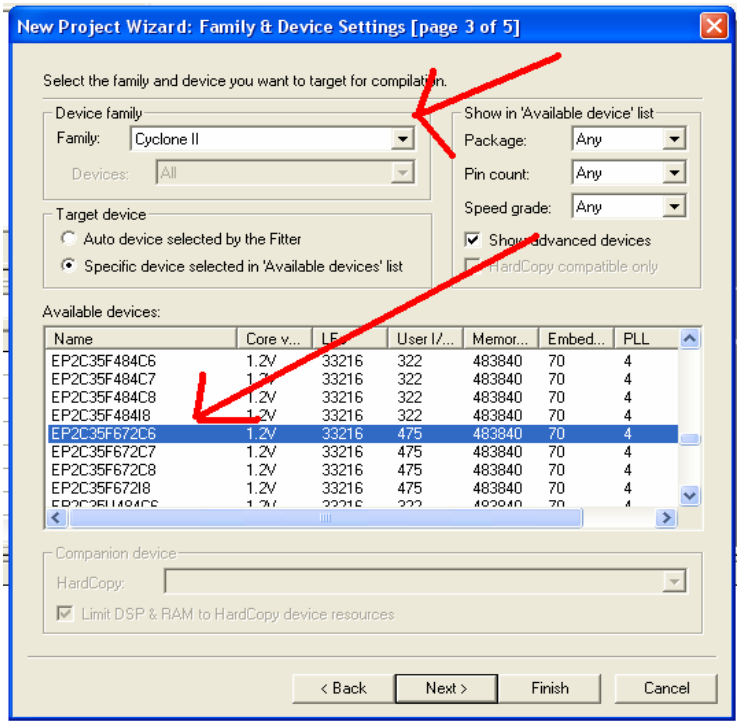
Bước 1. Vào ***Menu > file*** chọn ***New Project Wizard***



Hình 2.4: Tạo một Project mới

Bước 2. Chọn thư mục để chứa project và đặt tên cho project

Bước 3. Chọn hãng sản xuất chip và tên loại chip trên mạch



Hình 2.5: Cửa sổ chọn chip

+ Family : cyclone II

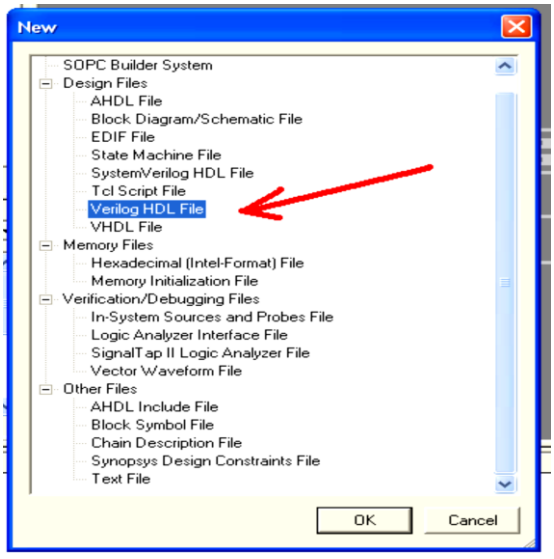
+ Check: show advanced device, specific device selected in …

+ Available device: Ep2c35F672C6

Bước 4. Chọn ***Finish*** để hoàn tất

2.3.2 Soạn thảo chương trình & biên dịch

Bước 1. Vào ***Menu > file*** chọn ***New.*** Hộp thoại New xuất hiện



Hình 2.6: Lựa chọn loại file cần tạo

Bước 2. Chọn loại file mà chúng ta muốn viết chương trình như hình 2.6.

+ Verilog
+ VHDL
+ State machine
+ Block Diagram/Schematic

Bước 3. Nhập file vào cửa sổ viết chương trình.

Ví dụ: viết chương trình bằng Verilog thực hiện hàm XOR

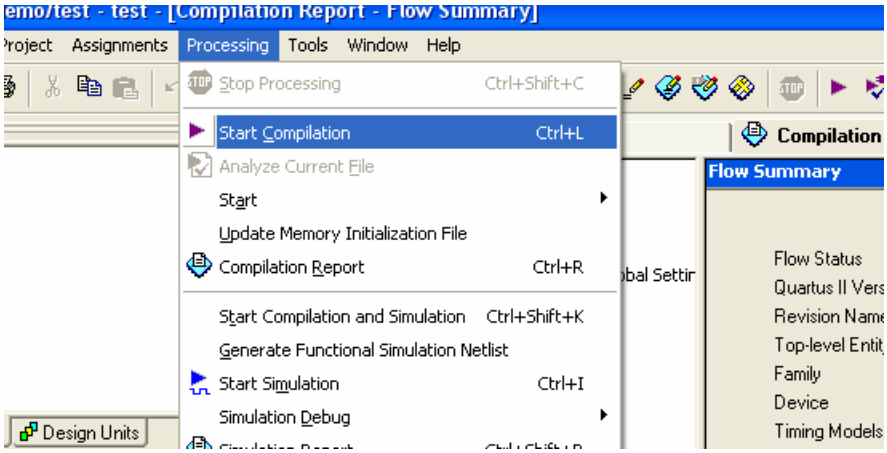
```
Module light(x1, x2, f);
Input x1, x2;
Output f;
Assign f=(x1&~x2) | (~x1&x2);
endmodule
```

Hình 2.7: Cửa sổ viết chương trình

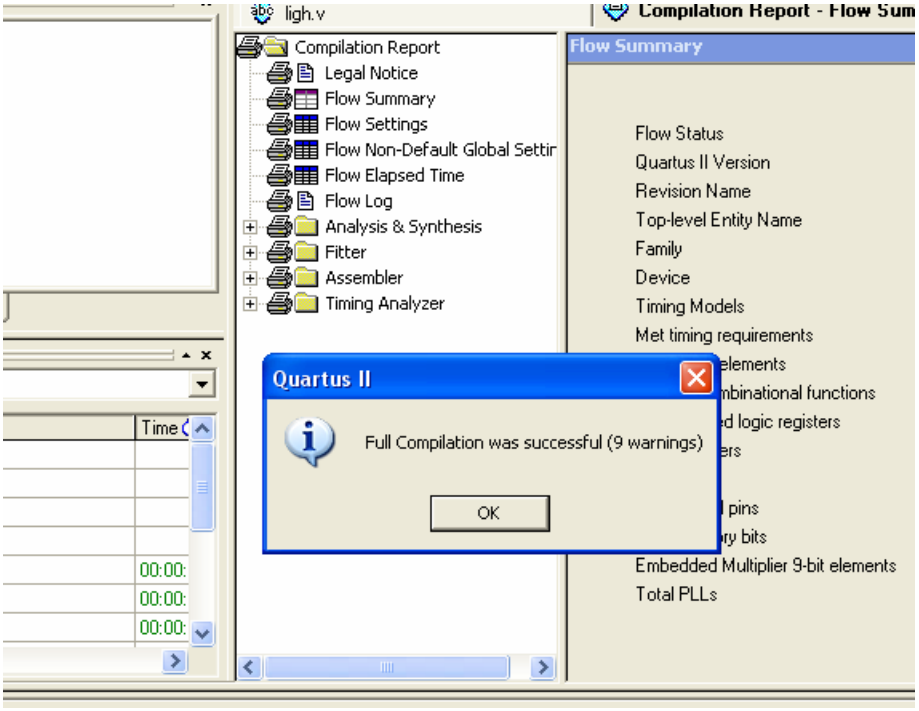
+ Lưu *file vừa nhập vào File > save As*
+ File name phải trùng với tên trong module hoặc là tên trong Entity
+ Save type chọn VHDL hoặc Verilog tùy thuộc vào bước 2.
+ Check hộp ***Add file to current project***

Bước 4. Sau khi lưu file xong phải thiết lập cho file là ***top-level*** thì mới biên dịch được.

Bước 5. Biên dịch chương trình: Vào **processing** > **Start compilation**



Hình 2.8: Biên dịch chương trình



Hình 2.9: Quá trình biên dịch thành công

Bước 6. Nhấn OK

Đến đây ta có thể nạp chương trình vào chip FPGA trên board DE2

2.4 NẠP CHƯƠNG TRÌNH LÊN BOARD DE2

2.4.1 Cấu hình chân

Trong quá trình biên dịch ở ***bước 5*** phần 2.3.2. Trình biên dịch tự do chọn lựa bất kỳ các chân trên FPGA làm ngõ vào và ngõ ra. Tuy nhiên, trên board DE2 đã thực hiện kết nối cứng các chân FPGA với các thành phần khác trên board. Chính vì

vậy chúng ta phải tiến hành gán chân theo kết nối cứng này khi dùng các thành phần trên board DE2.

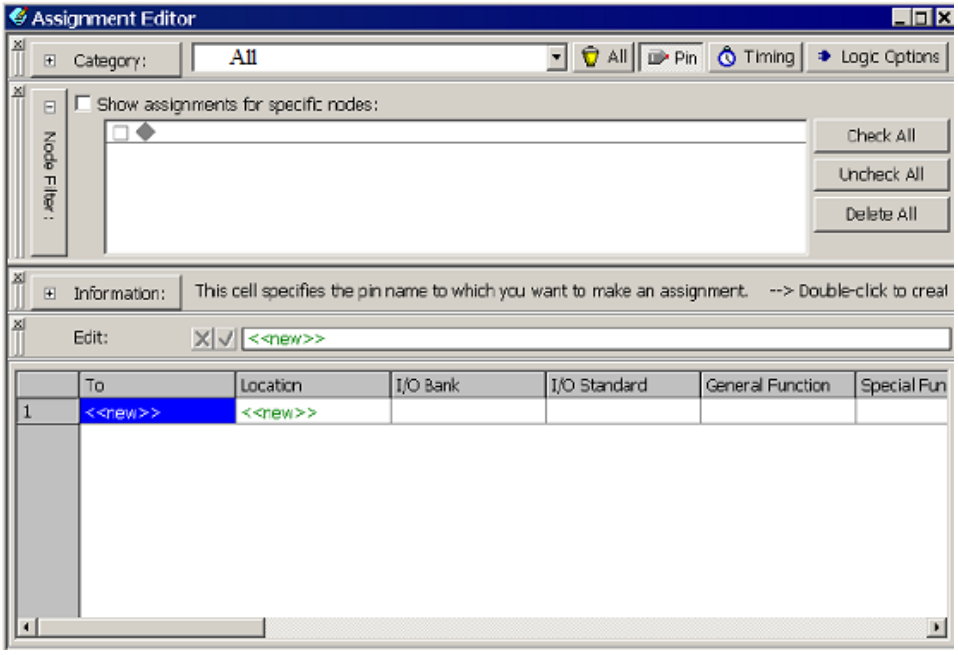
DE2 cung cấp một file cấu hình chuẩn, **DE2_pin_assigments.csv**. File cấu hình chuẩn này sẽ kết nối tất cả các chân của DE2. Khi đó chỉ cần đặt tên tín hiệu trùng tên với tín hiệu mà DE2 quy ước. Ví dụ: các nút nhấn sẽ có tên là SW, các đèn led sẽ có tên là LEDG hay LEDR, …

Thực hiện việc gán chân như sau cho ví dụ trên:

Component	DE2
SW_0	PIN_N25
SW_1	PIN_N26
$LEDG_0$	PIN_AE22

Hình 2.10: Gán chân cho chip FPGA

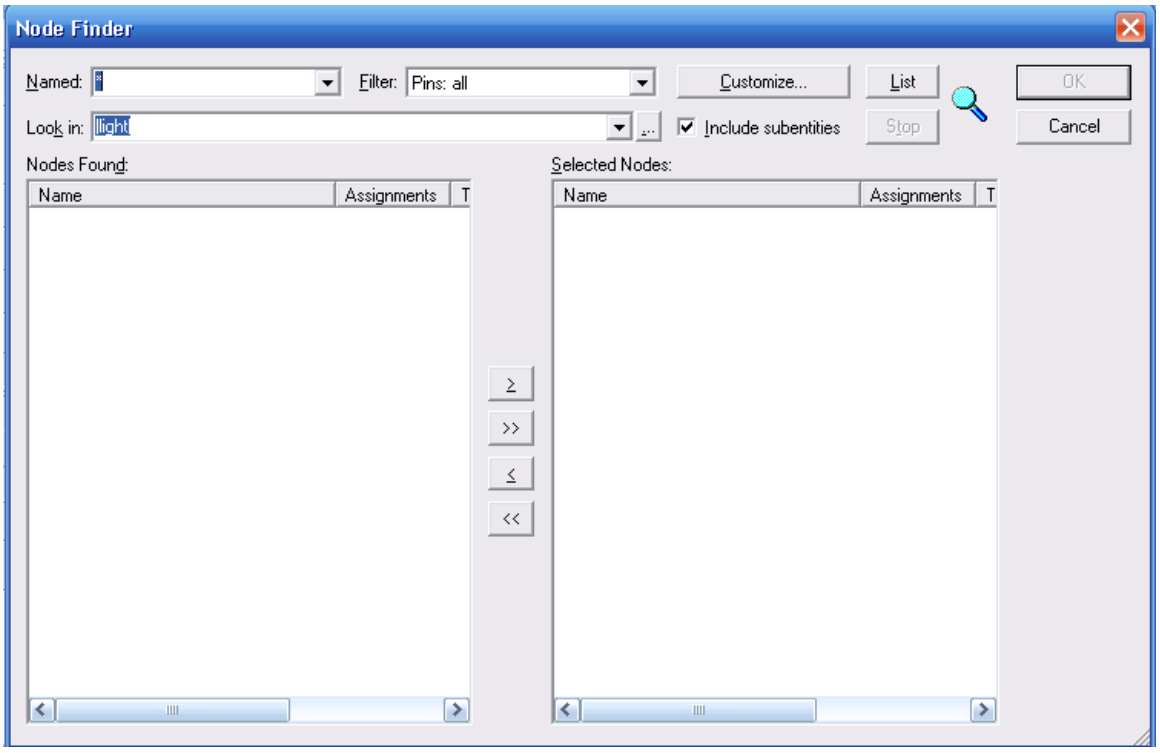
Bước 1. Chọn Assignments-> Assigment Editor cửa sổ lựa chọn chân xuất hiện



Hình 2.11: Cửa sổ gán chân

Trong tùy chọn Category chọn all.

Bước 2. Nhấp đúp vào ô <<new>> ở cột **To**. Chọn Node Finder. Cửa sổ Node Finder xuất hiện.



Hình 2.12: Cửa sổ chọn chân để gán

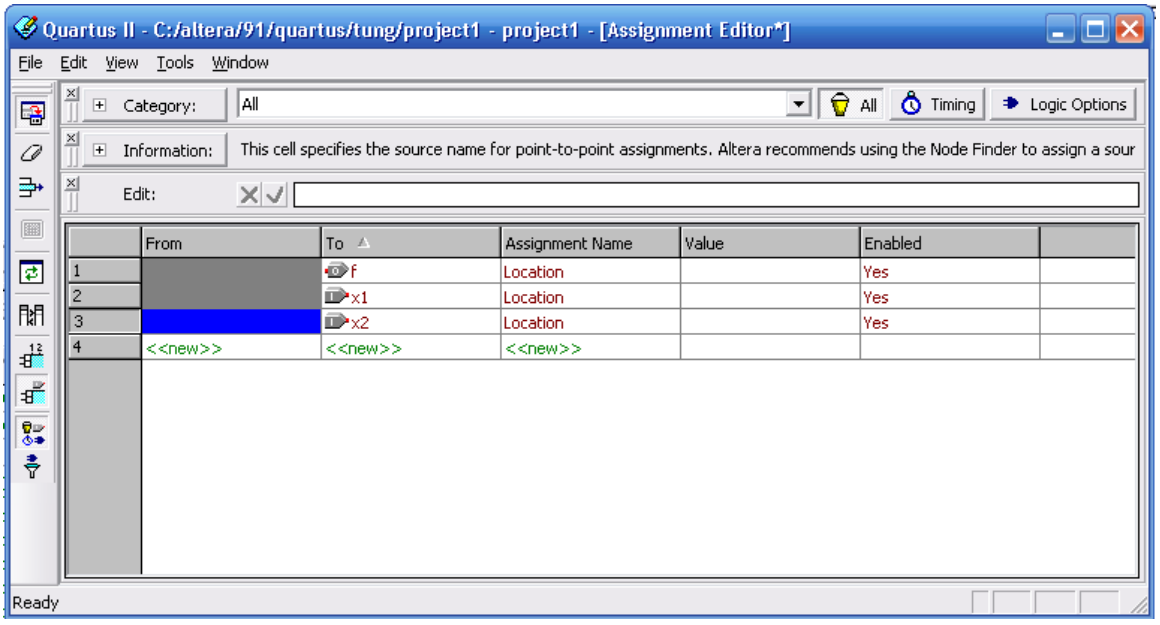
Trong cửa sổ Node finder:

+ Filter: Pins: all

+ Nhấn vào List

+ Nhấn vào nút 

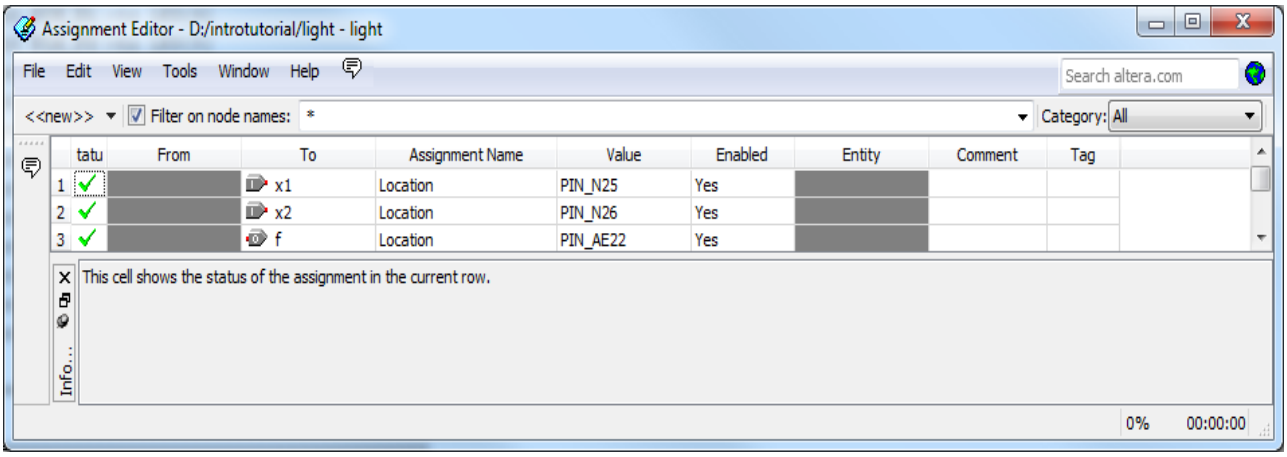
+ Nhấn OK



Hình 2.13: Hoàn tất chọn chân

Bước 3. Tiếp theo nhấn vào ô <<new>> ở cột Assigment Name chọn Location (Accepts wildcards/groups) cho tất cả các tín hiệu vào ra.

Bước 4. Nhấn đúp vào hộp trong cột Value nhập gán chân tương ứng như hình 2.10.



Hình 2.14: Quá trình gán chân hoàn tất

Bước 5. Để lưu lại file cấu hình chân bạn chọn File -> save, sau đó nhập tên file cần lưu.

Bước 6. Thực hiện biên dịch lại. Lần này trình biên dịch thực hiện biên dịch đúng với các chân đã gán.

2.4.2 Nạp lên board DE2

DE2 hỗ trợ 2 cách nạp lên boardard: **JTAG (**Joint Test Action Group) và **AS** (Active Serial). Để chuyển đổi giữa 2 chế độ nạp này thì trên boardard DE2 cung cấp nút **RUN/PROG**. **RUN** tương ứng với chế độ nạp **JTAG**, trong khi **PROG** là chế độ **AS**.

2.4.2.1 Lập trình theo chế độ JTAG

Trong chế độ **JTAG** (Joint Test Action Group) thì dữ liệu cấu hình sẽ được nạp trực tiếp lên FPGA. Trong chế độ này thì thông tin cấu hình sẽ bị mất khi tắt nguồn.

Bước 1. Gạt nút **RUN/PROG** trên boardard DE2 sang **RUN**, sau đó chọn **Tools->Programmer**

Bước 2. Trên cửa sổ lập trình, trong khung chế độ lập trình (Mode) chọ lựa chọn **JTAG**. Tiếp đó nếu **USB-Blaster** không được chọn như trên hình, thì bạn nhấn **Hardware Setup**, cửa sổ **Hardware Setup** sẽ xuất hiện khi đó bạn chọn **USB-Blaster** để lựa chọn cổng kết nối với boardard DE2.

Bước 3. Trở lại cửa sổ lập trình bạn chọn vào file lập trình (light.sof). Nếu file này chưa có thì bạn có thể nhấn nút **Add** File để thêm file này vào. Tiếp đó bạn nhấp vào lựa chọn **Program/configure**

Bước 4. Nhấn nút **Start** để bắt đầu việc lập trình. Trong khi lập trình thì các đèn led trên boardard DE2 sẽ sáng mờ đi. Trên cửa sổ lập trình, thanh **Progess** sẽ cho thấy tiến trình nạp lên boardard DE2.

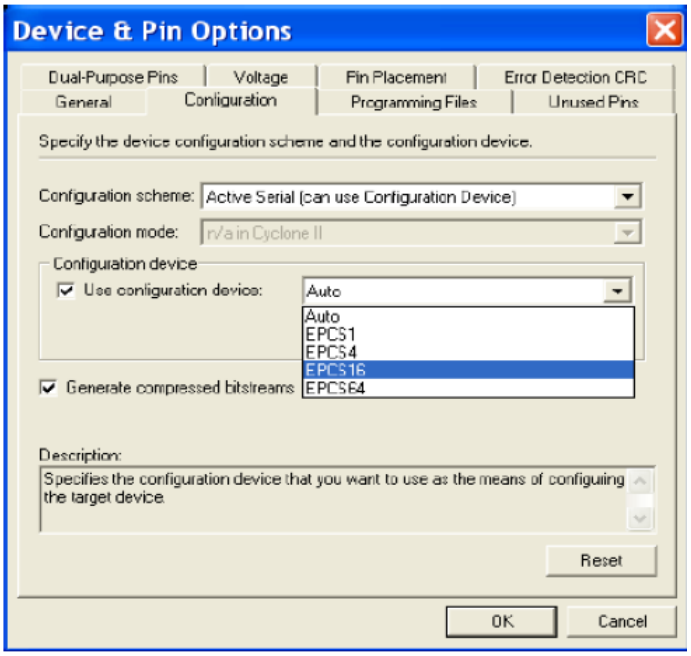
2.4.2.2 Lập trình theo chế độ As

Ở chế độ **AS** (Active Serial), thì dữ liệu cấu hình sẽ được nạp lên bộ nhớ flash. Mỗi khi mở nguồn (reset) thì thông tin cấu hình ở đây sẽ được load lên FPGA, do đó thông tin cấu hình FPGA sẽ không bị mất mỗi khi tắt nguồn.

Bước 1. Gạt nút **RUN/PROG** trên boardard DE2 sang **PROG**.

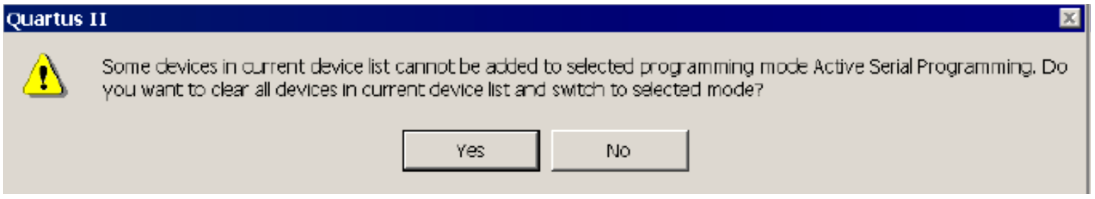
Bước 2. Vào **Assignment -> Device**, chọn **Device** và sau đó chọn thiết bị là **EP2C35F672C6**.

Tiếp đó nhấp vào **Device & Pin Options**, cửa sổ **Device & Pin Opions** sẽ xuất hiện, bạn chọn tab **Configuration**, trong khung **Configuration device** chọn **EPCS16**. Nhấn **OK** để ấn định sau đó dịch lại chương trình.



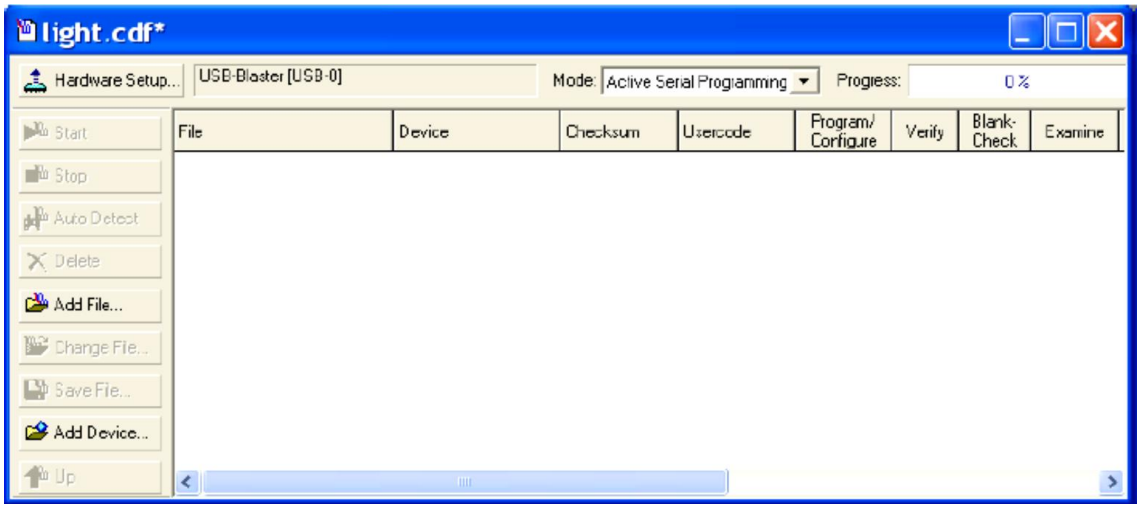
Hình 2.15: Cửa sổ device & device options

Bước 3. Tools->Programmer, cửa sổ lập trình sẽ xuất hiện (như hình cửa sổ lập trình). Tiếp đó trong khung **Mode** bạn chọn **Active Serial Programming**. Một thông báo sẽ hiện lên bạn chọn **Yes**.



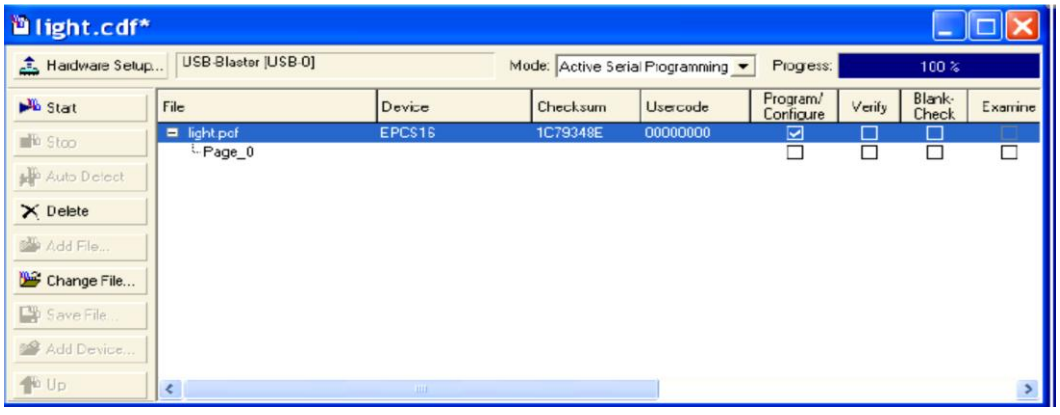
Hình 2.16: Thông báo thay đổi chế độ lập trình

Bước 4. Cửa sổ lập trình ở chế độ **AS** sẽ hiện ra như hình bên dưới. Bạn chọn nút **Add File** để thêm file cần nạp vào chương trình (**light.pof**, file để cấu hình cho chế độ **AS** sẽ có dạng ***.pof** và file cấu hình cho chế độ **JTAG** sẽ có dạng ***.sof**).



Hình 2.16: Cửa sổ lập trình chế độ AS

Bước 5. Nhấp vào lựa chọn **Program/Configure**. Tiếp đó bạn nhấn nút **Start** để nạp chương trình cấu hình lên boardard DE2.



Hình 2.17: Quá trình lập trình hoàn tất

2.5 MÔ PHỎNG CHƯƠNG TRÌNH

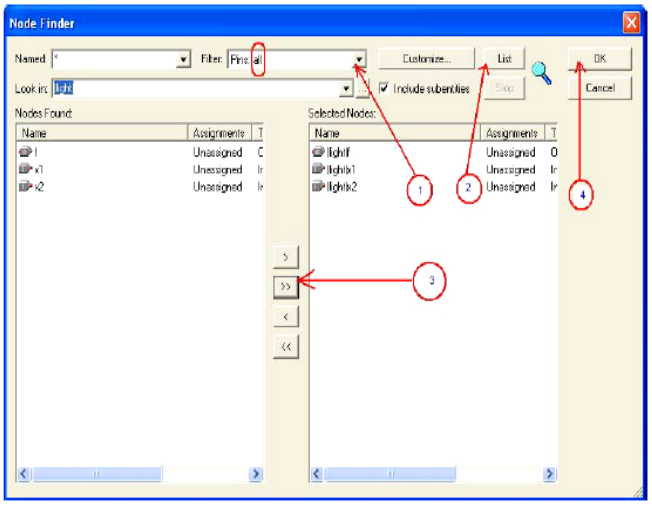
2.5.1 Tạo file mô phỏng

Bước 1. Vào ***Menu > file*** chọn ***New***, sau đó chọn ***Vector Waveform File*** →**Ok**

Bước 2. Sau đó nhấp chuột phải chọn ***Insert***, chọn ***Insert Node or bus***

Bước 3. Chọn ***Node Finder***

Bước 4. Cửa sổ lựa chọn tín hiệu sẽ xuất hiện



- Để hiển thị ra tất cả các chân :
  - Chọn ***Pins: all***.
  - Sau đó chọn nút ***List*** để hiện tất cả các chân.
- Nút [>] : Chọn từng tín hiệu
- Nút [>>] : Chọn tất cả các tín hiệu
- Nút [<] : Bỏ từng tín hiệu
- Nút [<<] : Bỏ tất cả các tín hiệu
- Ví dụ muốn chọn 3 tín hiệu f, x1, x2 ta có 2 cách:
  - Chọn từng tín hiệu theo nút [>]
  - Nhấn nút [>>] để chọn tất cả cá tín hiệu
- Nhấn ***OK*** để hoàn tất việc chọn tín hiệu

Hình 2.15: Cửa sổ chọn tín hiệu

Bước 5. Thiết lập giá trị các tín hiệu:

- Ta rê chuột để tô khối chúng lại
- Sau đó sử dụng các nút 0, 1 để thiết lập giá trị cho chúng

Bước 6. Sau khi thiết lập giá trị của các chân xong ta Save lại.

2.5.2 Chọn chế độ mô phỏng

Bước 1. Nhấp vào nút  để lựa chọn chế độ simulate

Bước 2. Trong cửa sổ setting test :

+ Trong khung **categocy** chọn **simulation settings**

+ Trong khung ***Simulate mode*** chọn chức năng ***Functional***

+ Ok

Bước 3. Vào ***Processing > Generate Functional Simulation Netlist*** để tiến hành quá trình phân tích và tổng hợp

Bước 4. Hộp thoại thông báo quá trình phân tích và tổng hợp thành công, chọn ***OK*** để ấn định

2.5.3 Chọn file mô phỏng

Bước 1. Lựa chọn file testbench cho chương trình cần mô phỏng

Vào Assigment > Settings

Bước 2. Nhấn vào nút  trong cửa sổ setting test để lựa chọn đường dẫn lưu trữ file testbench.

Bước 3. Lựa chọn vị trí lưu trữ file testbench trong hộp thoại ***Select File*** → **Ok**

2.5.4 Tiến hành mô phỏng

Chọn ***Processing > Start Simulation*** hoặc nhấp vào nút  để tiến hành quá trình simulation

Đến đây, nếu mô phỏng thành công ta nhận được thông báo:

"simulator was successful"

TÓM TẮT

Trong bài này học viên làm quen với board thí nghiệm DE2:

+ Nắm được tổ chức phần cứng board DE2

+ Các ứng dụng có thể triển khai trên board DE2

Sử dụng phần mềm Quartus:

+ Soạn thảo chương trình, biên dịch chương trình, tạo file mô phỏng, thực hiện việc mô phỏng

+ Thực hiện việc gán chân cho FPGA

+ Nạp chương trình lên board DE2.

CÂU HỎI ÔN TẬP

Câu 1: Dùng Quartus thực hiện chương trình mạch dồn kênh 2 $\rightarrow$ 1 bằng ngôn ngữ VERILOG. Thực hiện việc soạn thảo chương trình, biên dịch, mô phỏng chương trình trên.

Câu 2: Thực hiện cấu hình chân và nạp chương trình ở câu 1 theo chế độ JTAG

Câu 3: Thực hiện cấu hình chân và nạp chương trình ở câu 1 theo chế độ AS.

Câu 4: Thực hiện câu 1, 2, 3 với ngôn ngữ lập trình VHDL

Câu 5: Sử dụng các Switch SW_{17-0} như là các ngõ vào. Các $LEDR_{17-0}$ như là các ngõ ra.

a) Viết chương trình thực hiện kết nối các SW với các LEDR tương ứng bằng VERILOG, thực hiện chương trình lên FPGA

b) Thực hiện việc mô phỏng trên Quartus

c) Yêu cầu giống như câu a và b nhưng thực hiện bằng ngôn ngữ VHDL

BÀI 3: NGÔN NGỮ LẬP TRÌNH VERILOG

Sau khi học xong bài này, học viên có thể:

- *Dùng ngôn ngữ đặc tả phần cứng để mô tả mạch số*
- *Viết chương trình cho FPGA bằng ngôn ngữ VERILOG;*

3.1 GIỚI THIỆU

Một số phương pháp mô tả mạch số:

+ Mô tả bằng sơ đồ khối

+ Mô tả bằng ngôn ngữ mô tả phần cứng

+ Mô tả bằng máy trạng thái

Trong bài thí nghiệm này giới thiệu phương pháp mô tả mạch số bằng ngôn ngữ phần cứng.

Phương pháp này cho phép mô tả vi mạch bằng các cú pháp tương tự như cú pháp của ngôn ngữ lập trình. Một số ngôn ngữ mô tả phần cứng phổ biến hiện nay là:

+ *Verilog*

+ VHDL

+ AHDL

+ System verilog

3.2 NGÔN NGỮ LẬP TRÌNH VERILOG

Đây là một ngôn ngữ mô tả phần cứng có cấu trúc và cú pháp gần giống với ngôn ngữ lập trình C, ngoài khả năng hỗ trợ thiết kế thì Verilog rất mạnh trong việc hỗ trợ cho quá trình kiểm tra thiết kế.

3.2.1 Cấu trúc chương trình Verilog

Ngôn ngữ Verilog mô tả hệ thống số như là thiết lập một module. Cấu trúc một module như sau :

---

module module_name (danh sách các port);

input;

output;

inout;

reg;

wire;

parameter ;

.

.

.

//các câu lệnh

Initial statement

Always statement

Module Instantiation

Endmodule

3.2.2 Các bài thí nghiệm verilog

Bài 1: Viết chương trình Verilog thực hiện 8 bộ dồn kênh 2 sang 1 (eight-bit wide 2-1 multiplexer):

a) Tạo 1 project mới trên quartus

b) Viết file verilog sử dụng SW_{17} là ngõ vào chọn (s). SW_{7-0} ngõ vào X. SW_{15-8} ngõ vào Y. Ngõ ra M nối đến các $LEDG_{7-0}$. Nối các SW đến LEDR. Đoạn code chương trình 8-bit wide 2-1 multiplexer

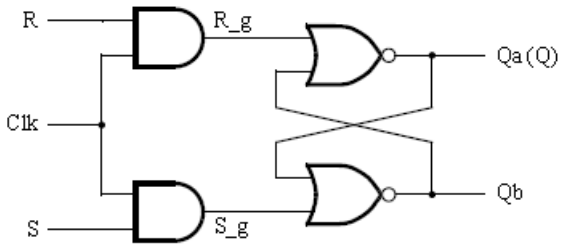
```
module part3(SW,s,LEDR, LEDG);
input [15:0]SW; //X =7-0, Y=15-8
input s;
output [7:0]LEDG; //M
output [15:0]LEDR; //15-8=X, 7-0=Y
assign LEDG[0]=(~s&SW[0])|(s&SW[8]);
assign LEDG[1]=(~s&SW[1])|(s&SW[9]);
assign LEDG[2]=(~s&SW[2])|(s&SW[10]);
assign LEDG[3]=(~s&SW[3])|(s&SW[11]);
assign LEDG[4]=(~s&SW[4])|(s&SW[12]);
assign LEDG[5]=(~s&SW[5])|(s&SW[13]);
assign LEDG[6]=(~s&SW[6])|(s&SW[14]);
assign LEDG[7]=(~s&SW[7])|(s&SW[15]);
assign LEDR=SW;
endmodule
```

c) Thực hiện việc gán chân

d) Biên dịch chương trình

e) Nạp chương trình lên board DE2, kiểm tra.

Bài 2: Cho mạch JK-FF như sau:



Hình 3.1: Sơ đồ khối RS-FF

a) Tạo 1 project mới trên Quartus

b) Viết File Verilog

```
module part1 (Clk, R, S, Q);
input Clk, R, S;
output Q;
wire R_g, S_g, Qa, Qb /* synthesis keep */ ;
assign R_g = R & Clk;
assign S_g = S & Clk;
assign Qa = ~(R_g | Qb);
assign Qb = ~(S_g | Qa);
assign Q = Qa;

endmodule
```

c) Biên dịch chương trình, sử dụng công cụ RTL Viewr của Quartus để xem mạch được vẽ ở mức cổng

d) Tiến hành mô phỏng bằng Quartus.

Bài 3: Viết file Verilog thiết kế mạch đếm lên không đồng bộ 4 bit có ngõ vào xóa (clr) tích cực mức thấp. Thực hiện việc gán chân như sau: Key_0 ngõ vào Clk, SW_0 ngõ vào Clr. HEX0 hiển thị giá trị bộ đếm.

Đoạn code chương trình:

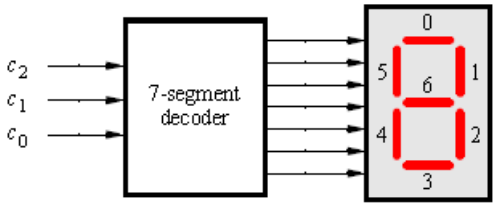
```
module counter (clk, clr, q);
input   clk, clr;
```

```
output [3:0] q;
reg    [3:0] tmp;
always @(posedge clk or posedge clr)
      begin
          if (clr)
             tmp <= 4'b0000;
          else
             tmp <= tmp + 1'b1;
       end
          assign q = tmp;
        endmodule
```

a) Mô phỏng chương trình trên Quartus.

Thực hiện chương trình trên board DE2

Bài 4: Một mạch giải mã 7 đoạn có 3 ngõ vào $C_2C_1C_0$.



$c_2c_1c_0$	Character
000	H
001	E
010	L
011	O
100	
101	
110	
111	

Hình 3.2: Bộ giải mã LED 7 đoạn và bảng sự thật

a) Tạo 1 project trên quartus

b) Viết file Verilog sử dụng SW_{2-0} cho các ngõ vào $C_2C_1C_0$. Nối các ngõ ra mạch giải mã tới các đoạn của HEX0. Đoạn code verilog

```
module part4(C, display);
input [2:0]C;
output [0:6]display;
```

```
assign display[0]=~C[0];
assign display[1]=(~C[1]&C[0])|(C[1]&~C[0]);
assign display[2]=(~C[1]&C[0])|(C[1]&~C[0]);
assign display[3]=(~C[1]&~C[0]);
assign display[4]=0;
assign display[5]=0;
assign display[6]=C[1];
endmodule
```

c) Thực hiện việc gán chân

d) Biên dịch

e) Thực thi chương trình trên board DE2

TÓM TẮT

Trong bài này, học viên làm quen với các ngôn ngữ lập trình đặc tả phần cứng để lập trình FPGA; Viết chương trình mô tả mạch số bằng ngôn ngữ VERILOG và thực hiện trên FPGA

CÂU HỎI ÔN TẬP

Câu 1: Thiết kế mạch tổ hợp thực hiện việc chuyển mã từ BCD sang LED 7 đoạn, hiển thị lên các LED từ HEX3 đến HEX0, các giá trị được thiết lập bởi các SW_{15-0} tương ứng như sau: SW_{15-12}, SW_{11-8}, SW_{7-4}, SW_{3-0} hiển thị lên các LED HEX3, HEX2, HEX1 và HEX0. Các LED hiển thị từ 0 đến 9, các tổ hợp 1010 đến 1111 xem là don't care.

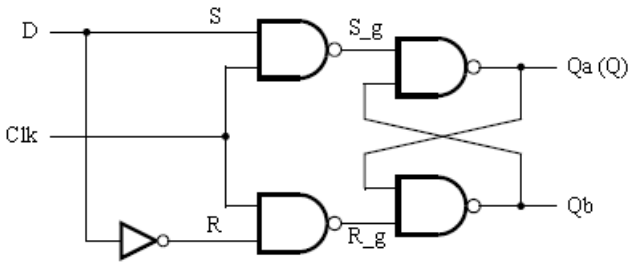
a) Tạo 1 project mới trên Quartus

b) Viết file Verilog

c) Thực hiện việc gán chân

d) Thực thi chương trình trên board DE2

Câu 2: Cho mạch D-FF



a) Tạo 1 project mới trên Quartus

b) Viết File Verilog

c) Mô phỏng chương trình

d) Thực hiện việc cấu hình chân: SW_0 = D; SW_1 = Clk; LEDR = Q.

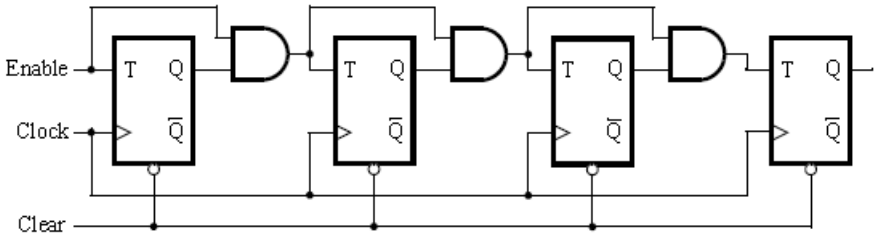
e) Biên dịch lại chương trình và thực thi trên board DE2.

Câu 3: Viết file verilog thực hiện mạch đếm xuống không đồng bộ 8 bit có ngõ vào clear tích cực mức thấp. Thực hiện việc gán chân như sau: Key_0 ngõ vào Clk, SW_0 ngõ vào Clr. HEX1 và HEX0 hiển thị giá trị bộ đếm

b) Mô phỏng chương trình trên Quartus.

c) Thực hiện chương trình trên board DE2

Câu 4: Cho mạch đếm



a) Tạo 1 project mới trên Quartus

b) Viết File Verilog cho mạch đếm. KEY_0 = clock, SW_0 = Clear, SW_1 = Enable. HEX0 hiển giá trị mạch đếm

c) Thực hiện việc gán chân, biên dịch chương trình và thực thi trên board DE2.

BÀI 4: NGÔN NGỮ LẬP TRÌNH VHDL

Sau khi học xong bài này, học viên có thể:

Dùng ngôn ngữ đặc tả phần cứng để mô tả mạch số. Viết chương trình cho FPGA bằng ngôn ngữ VHDL.

4.1 GIỚI THIỆU

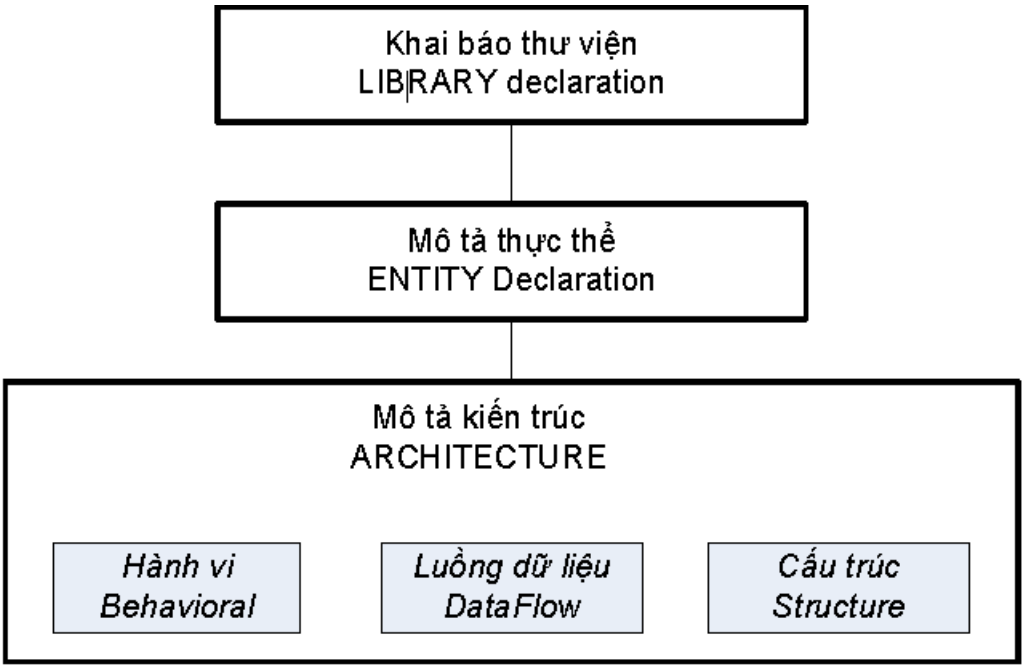
Một số phương pháp mô tả mạch số đã được đề cập ở bài 3. Trong bài thí nghiệm này giới thiệu phương pháp mô tả mạch số bằng ngôn ngữ phần cứng bằng ngôn ngữ VHDL.

4.2 NGÔN NGỮ LẬP TRÌNH VHDL

VHDL được phát triển dựa trên cấu trúc của ngôn ngữ lập trình Ada. Cấu trúc của mô tả VHDL tuy phức tạp hơn Verilog nhưng mang tính logic chặt chẽ và gần với phần cứng hơn.

4.2.1 Cấu trúc chương trình VHDL

Cấu trúc tổng thể của một module VHDL gồm ba phần, phần khai báo thư viện, phần mô tả thực thể và phần mô tả kiến trúc.



Hình 4.1: Cấu trúc một thiết kế VHDL

4.2.2 Các bài thí nghiệm VHDL

Bài 1: Viết chương trình thực hiện D-FF trên board DE2

a) Tạo 1 projcet mới

b) Viết file VHDL như sau

```
library ieee ;
use ieee.std_logic_1164.all;
entity dff is
port(  data_in: in std_logic;
  clock:    in std_logic;
  data_out:     out std_logic);
end dff;
architecture behv of dff is
begin
```

```
    process(data_in, clock)
    begin
        if (clock='1' and clock'event) then
      data_out <= data_in;
  end if;
    end process;
end behv;
```

c) Biên dịch chương trình

d) Tạo file mô phỏng, tiến hành mô phỏng

e) Thực hiện việc gán chân: data_n = $SW_{0;}$ data_out = $LEDG_0$. Clock = KEY_0. Biên dịch lại chương trình và thực thi trên board DE2.

Bài 2: Viết chương trình thực hiện mạch cộng 2 số 4 bit trên board DE2

a) Tạo 1 projcet mới

b) Viết file VHDL như sau

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity adder4 is
port(
Cin : in std_logic;
A : in std_logic_vector(3 downto 0);
B : in std_logic_vector(3 downto 0);
SUM : out std_logic_vector(3 downto 0);
Cout: out std_logic
);
end adder4;
architecture behavioral of adder4 is
```

```
signal A_temp : std_logic_vector(4 downto 0);
signal B_temp : std_logic_vector(4 downto 0);
signal Sum_temp : std_logic_vector(4 downto 0);
begin
    A_temp <= '0'& A;
    B_temp <= '0'& B;
    plus: process (A_temp, B_temp, Cin)
    begin
        sum_temp <= a_temp + b_temp + Cin;
end process plus;
    SUM <= sum_temp(3 downto 0);
    Cout <= sum_temp(4);
end behavioral;
```

c) Biên dịch chương trình

d) Tạo file mô phỏng, tiến hành mô phỏng. Thực hiện việc gán chân: A = SW_{3-0}; B = SW_{7-4}; Cin = SW_8; SUM =$LEDG_{3-0}$; Cout = $LEDR_0$

e) Biên dịch lại chương trình và thực thi trên board DE2.

Bài 3: Viết chương trình thực hiện mạch đếm n bit

a) Tạo 1 projcet mới

b) Viết file VHDL như sau

```
ibrary ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter is
generic(n: natural :=2);
port(   clock:      in std_logic;
  clear:      in std_logic;
```

```
  count:     in std_logic;
  Q:    out std_logic_vector(n-1 downto 0));
end counter;
architecture behv of counter is
signal Pre_Q: std_logic_vector(n-1 downto 0);
begin
    -- behavior describe the counter
    process(clock, count, clear)
    begin
  if clear = '1' then
      Pre_Q <= Pre_Q - Pre_Q;
  elsif (clock='1' and clock'event) then
      if count = '1' then
         Pre_Q <= Pre_Q + 1;
      end if;
  end if;
    end process;
    -- concurrent assignment statement
    Q <= Pre_Q;
end behv;
```

c) Biên dịch chương trình

d) Tạo file mô phỏng, tiến hành mô phỏng. Thực hiện việc gán chân: clock = KEY_0; Clear = SW_0; cout = SW_1; Q = $LEDG_{n-1}$ đến $LEDG_0$.

e) Biên dịch lại chương trình và thực thi trên board DE2.

TÓM TẮT

Trong bài này, học viên làm quen với các ngôn ngữ lập trình đặc tả phần cứng để lập trình FPGA. Viết chương trình mô tả mạch số bằng ngôn ngữ VHDL và thực hiện trên FPGA

CÂU HỎI ÔN TẬP

Câu 1: Thiết kế mạch tổ hợp thực hiện việc chuyển mã từ BCD sang LED 7 đoạn, hiển thị lên các LED từ HEX3 đến HEX0, các giá trị được thiết lập bởi các SW_{15-0} tương ứng như sau: SW_{15-12}, SW_{11-8}, SW_{7-4}, SW_{3-0} hiển thị lên các LED HEX3, HEX2, HEX1 và HEX0. Các LED hiển thị từ 0 đến 9, các tổ hợp 1010 đến 1111 xem là don't care.

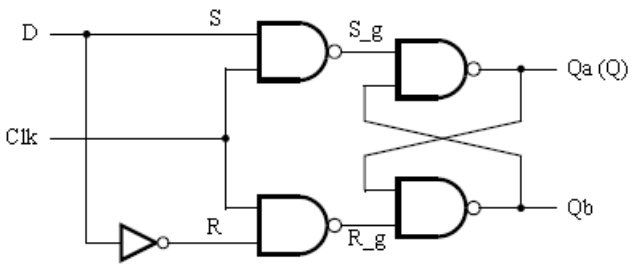
a) Tạo 1 project mới trên Quartus

b) Viết file VHDL

c) Thực hiện việc gán chân

d) Thực thi chương trình trên board DE2

Câu 2: Cho mạch D-FF



a) Tạo 1 project mới trên Quartus

b) Viết File VHDL

c) Mô phỏng chương trình

d) Thực hiện việc cấu hình chân: SW_0 = D; SW_1 = Clk; LEDR = Q.

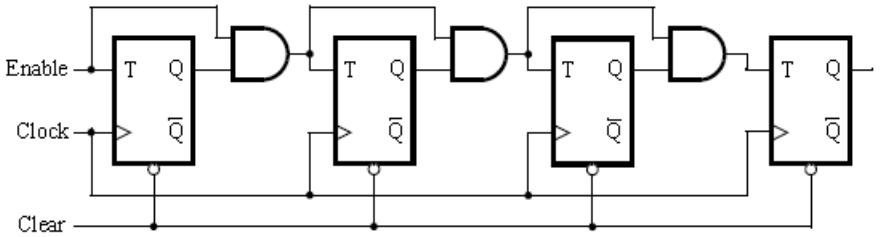
e) Biên dịch lại chương trình và thực thi trên board DE2.

Câu 3: Viết file VHDL thực hiện mạch đếm xuống không đồng bộ 8 bit có ngõ vào clear tích cực mức thấp. Thực hiện việc gán chân như sau: Key_0 ngõ vào Clk, SW_0 ngõ vào Clr. HEX1 và HEX0 hiển thị giá trị bộ đếm

a) Mô phỏng chương trình trên Quartus.

b) Thực hiện chương trình trên board DE2

Câu 4: Cho mạch đếm



d) Tạo 1 project mới trên Quartus

e) Viết File VHDL cho mạch đếm. KEY_0 = clock, SW_0 = Clear, SW_1 = Enable. HEX0 hiển giá trị mạch đếm

f) Thực hiện việc gán chân, biên dịch chương trình và thực thi trên board DE2.

BÀI 5: THIẾT KẾ FPGA THÔNG QUA SƠ ĐỒ KHỐI

Sau khi học xong bài này, học viên có thể nắm được:

- *Các bước thiết kế mạch tổ hợp, mạch tuần tự;*
- *Các phương pháp rút gọn biểu thức đại số BOARDOLEAN;*
- *Thiết kế FPGA thông qua sơ đồ khối Schematic.*

5.1 GIỚI THIỆU

Bài thí nghiệm này trình bày phương pháp mô tả mạch số sử dụng sơ đồ khối (Schematic), sử dụng trực tiếp các khối luận lý là một trong những cách khá thông dụng.

Mạch số được mô tả trực quan bằng cách ghép nối các phần tử logic khác nhau một cách trực tiếp. Thông thường các phần tử không đơn thuần là các đối tượng đồ họa mà còn có các đặc tính vật lý gồm chức năng logic, tải vào ra, thời gian trễ… Những thông tin này được lưu trữ trong thư viện logic thiết kế. Mạch vẽ ra có thể được mô phỏng để kiểm tra chức năng và phát hiện và sửa lỗi một cách trực tiếp.

5.2 THIẾT KẾ FPGA THÔNG QUA SƠ ĐỒ KHỐI (Schematic)

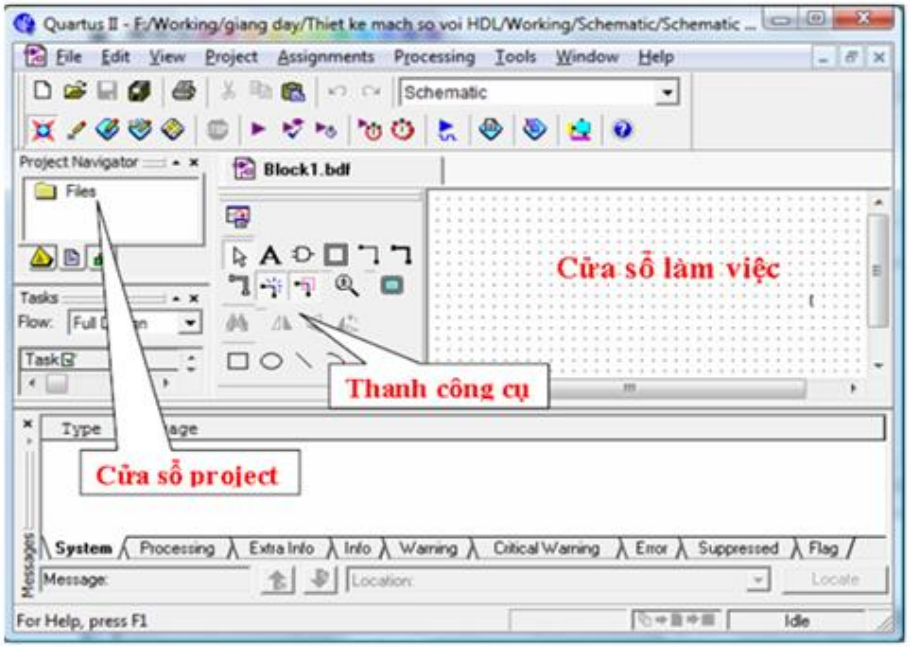
5.2.1 Tạo sơ đồ mạch

Để nắm rõ quá trình thiết kế này, chúng ta thực hiện một ví dụ hàm logic thực phép NOR 2 ngõ vào.

Bước 1. Mở Quartus II và tạo một project mới (thực hiện như đã trình bày ở bài 2)

Bước 2. Vào **File -> New**. Cửa sổ lựa chọn loại file sẽ xuất hiện chọn **Block Diagram/Schematic File** sau đó nhấn **OK**.

Bước 3. Cửa sổ soạn thảo sẽ hiện ra.



Hình 5.1: Cửa sổ soạn thảo sơ đồ khối

Chức năng một số nút trên thanh công cụ:

a. (*Select tool*): con trỏ là công cụ giúp chọn lựa các thành phần trên cửa sổ làm việc

b. Nút (*Text tool*) : công cụ tạo các ra các dòng văn bản trên cửa sổ làm việc

c. (*Symbol tool*): chứa linh kiện sử dụng cho quá trình soạn thảo (các cổng luận lý, các megafunction, và các chứa năng khác)

d. (*Block tool*): công cụ hỗ trợ việc tạo ra các khối chức năng. Giúp cho việc thiết kế nhiều cấp chức năng.

e. (*orthogonal node tool*): nối dây tín hiệu

f. (*orthogonal bus tool*): nối bus cho các tín hiệu

g. (*Zoom tool*): phóng to, thu nhỏ

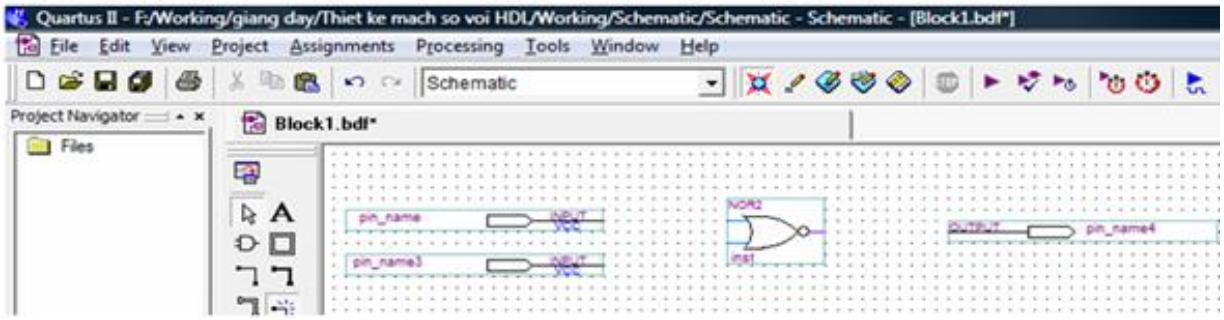
h. (*Full Screen*): Lựa chọn chế độ cửa sổ làm việc là Full Screen hay không.

i. (Find): công cụ tìm kiếm trên cửa sổ làm việc

Bước 4. Tiếp đến chọn nút (Symboardl tool) cửa sổ symboardl xuất hiện

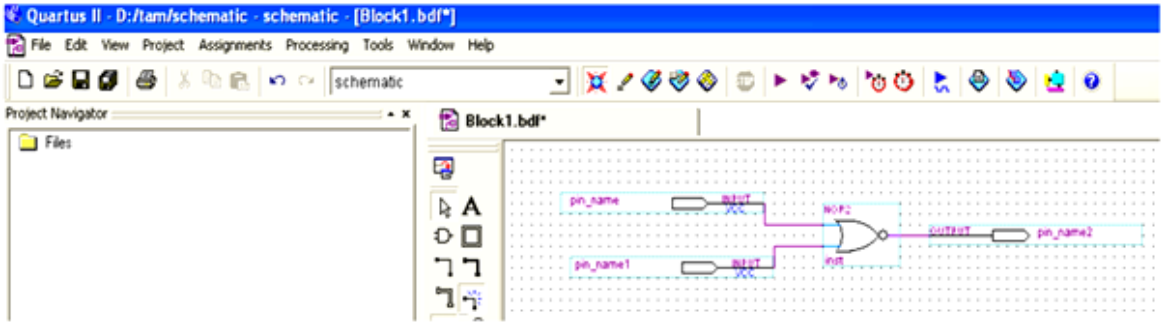
Bước 5. Trên cửa sổ lựa chọn linh kiện bạn chọn primitives -> logic -> NOR2 sau đó nhấn OK. Tiếp đó bạn nhấn chuột trái lên cửa sổ làm việc để thực hiện việc đặc một linh kiện cổng NOR (2 ngõ nhập). Để kết thúc việc chọn cổng NOR2, bạn nhấp chuột phải vào cửa sổ làm việc và chọn Cancel.

Bước 6. Tiếp tục chọn (Symbol tool), trong cửa sổ lựa chọn linh kiện bạn chọn đường dẫn primitives -> pin -> input, sau đó bạn đặt 2 tín hiệu ngõ nhập. Lặp lại quá trình này để có thêm một tín hiệu output như hình bên dưới



Hình 5.2: Cửa sổ bố trí các phần tử trong sơ đồ khối

Bước 7. Bước kế tiếp là kết nối chân các linh kiện. Bạn nhấp vào biểu tượng (orthogonal node tool) trên thanh công cụ, sau đó drag chuột từ vị trí muốn nối đến vị trí đích. Sau khi kết nối các ngõ vào ngõ ra ta được sơ đồ khối hoàn chỉnh.



Hình 5.3: Thực hiện nối dây các phần tử

Bước 8. Lưu sơ đồ khối *Vào File > save As*

Check hộp ***Add file to current project***

Bước 9. Sau khi lưu file xong phải thiết lập cho file là ***top-level*** thì mới biên dịch được.

Bước 10. Biên dịch chương trình: Vào **processing** > **Start compilation**

Bước 11. Nhấn OK

5.2.2 Mô phỏng chương trình

Bước 1. Tạo file mô phỏng: Thực hiện như phần 2.5.1 bài 2

Bước 2. Chọn chế độ mô phỏng: Thực hiện như phần 2.5.2 bài 2

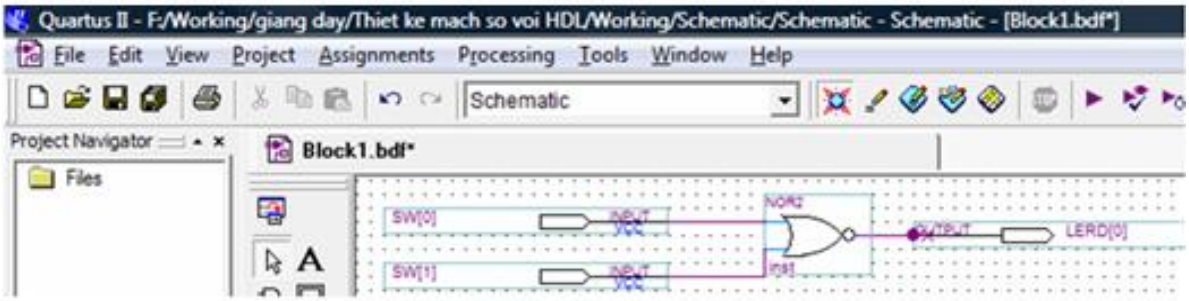
Bước 3. Chọn file mô phỏng: Thực hiện như phần 2.5.3 bài 2

Bước 4. Tiến hành mô phỏng: Thực hiện như phần 2.5.4 bài 2

5.2.3 Gán chân và nạp chương trình

Bước 1. Nhấp đúp vào tín hiệu nhập để gán chân cho mạch thiết kế. Nhập giá trị SW[0] vào ô pin name sau đó nhấn OK.

Bước 2. Tương tự như vậy bạn thiết lập cho tín hiệu nhập còn lại là SW[1] và tín hiệu output là LEDR[0].



Hình 5.4: Thiết lập tín hiệu

Bước 3. Chọn Assignments -> Import Assignments, cửa sổ lựa chọn file cấu hình chân cho FPGA xuất hiện, bạn chọn đường dẫn cho file DE2_pin_assignments.csv rồi nhấn OK.

Bước 4. Nhấn Ctrl + S hoặc nhấp vào biểu tượng trên thanh công cụ. Khi hộp thoại Save As xuất hiện bạn chọn đúng thư mục chứa project hiện tại của mình và lưu với tên trùng với tên project của mình, đồng thời chọn Add file to current project sau đó nhấn OK

Bước 5. Nhấp nút trên thanh công cụ hoặc chọn Processing -> Start Compilation để biên dịch chương trình

Bước 6. Sau khi quá trình compilation hoàn tất nhấn OK

Bước 7. Tiến hành nạp lên boardard DE2. Quy trình nạp này hoàn toàn giống phần 2.4.2 bài 2.

TÓM TẮT

Trong bài này, học viên làm quen với việc lập trình FPGA thông qua sơ đồ khối:

+ Tạo sơ đồ mạch trên phần mềm Quartus.

+ Tạo file mô phỏng cho sơ đồ vừa tạo.

+ Tiến hành mô phỏng.

+ Thực hiện việc gán chân & lập trình cho FPGA.

CÂU HỎI ÔN TẬP

Câu 1: Thực hiện hàm logic $Y = \sum_{ABCD}(1,3,5,7,9,10) + d(8)$ bằng sơ đồ Schematic trên Quartus và thực hiện lên FPGA

Câu 2: Thực hiện hàm logic $Y = \prod_{ABCD}(0,2,4,6,8,12) + d(9)$ bằng sơ đồ Schematic trên Quartus và thực hiện lên FPGA.

Câu 3: Thực hiện các mạch mã hóa, giải mã, dồn kênh, phân kênh bằng sơ đồ Schematic trên Quartus và thực hiện lên FPGA.

Câu 4: Hãy hiện thực hàm F (không rút gọn) lên FPGA bằng phương pháp thiết kế sơ đồ khối, tín hiệu nhập được đưa vào từ các SW, và tín hiệu xuất là các LEDR.

$$F = \overline{A(B + \overline{C}) + \overline{D + BC}}$$

Câu 5: Thiết kế mạch chốt RS, JK, D, T dùng các cổng logic và hiện thực chúng lên FPGA.

BÀI 6: MÁY TRẠNG THÁI

Sau khi học xong bài này, học viên có thể:

- *Nắm được mô hình máy trạng thái kiểu Moore & Mealy*
- *Thiết kế sử dụng mô hình máy trạng thái*
- *Tạo file ngôn ngữ mô tả phần cứng từ máy trạng thái.*

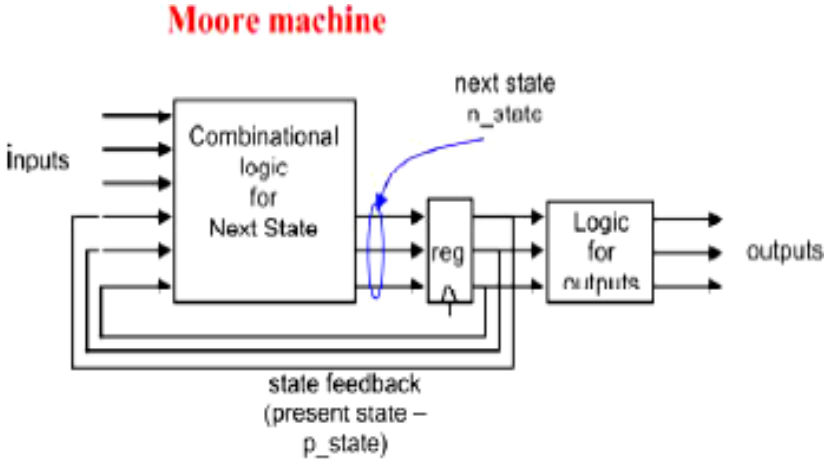
6.1 MÁY TRẠNG THÁI

Máy trạng thái là mạch tuần tự. Gồm mạch tổ hợp cộng với phần tử nhớ. Giá trị ngõ ra của máy trạng thái đối với giá trị ngõ vào phụ thuộc vào trạng thái hiện tại của nó.

Máy trạng thái có 2 kiểu: Kiểu MOORE và MEALY

6.1.1 Máy trạng thái kiểu MOORE

Ngõ ra chỉ phụ thuộc vào trạng thái hiện tại. Ngõ ra FSM chỉ thay đổi khi trạng thái của FSM thay đổi. Vì ngõ ra "*chẳng liên quan*" gì đến ngõ vào.

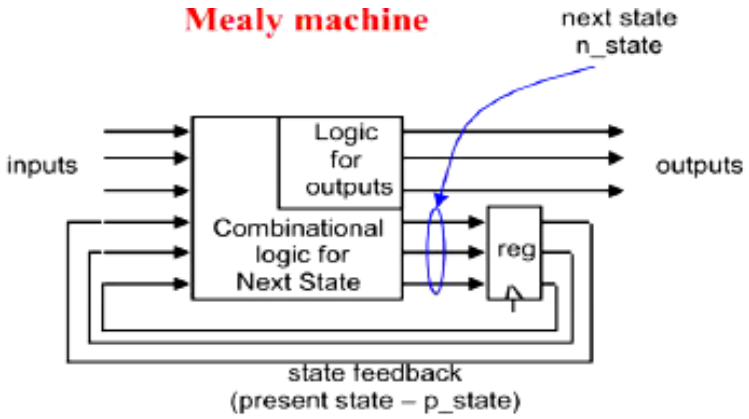


Hình 6.1: Máy trạng thái kiểu MOORE

Khi các giá trị ngõ vào thay đổi thì các giá trị của các ngõ ra chỉ có thể thay đổi ở xung đồng hồ kế tiếp (luôn trễ hơn 1 xung đồng hồ)

6.1.2 Máy trạng thái kiểu MEALY

Ngõ ra phụ thuộc vào trạng thái hiện tại và giá trị ngõ vào. Nghĩa là, ngõ ra FSM thay đổi khi trạng thái của FSM thay đổi hoặc giá trị ngõ vào thay đổi.



Hình 6.2: Máy trạng thái kiểu MEALY

Khi giá trị của các ngõ vào thay đổi thì giá trị của các ngõ ra có thể thay đổi trong cùng chu kỳ xung đồng hồ

6.2 GIẢN ĐỒ TRẠNG THÁI

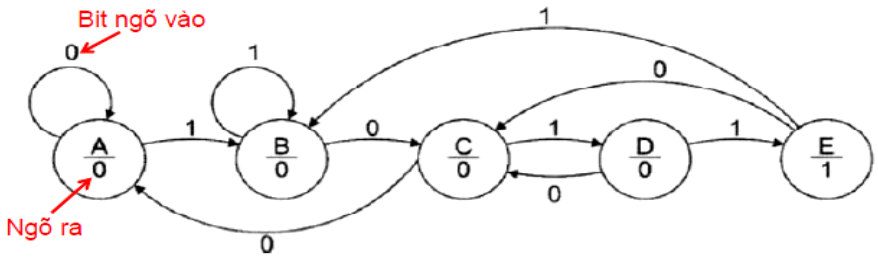
Giản đồ trạng thái cung cấp một phương pháp mô tả đáp ứng của mạch tuần tự. Các trạng thái được biểu diễn bởi các vòng tròn hoặc các nút, và quá trình chuyển là các mũi tên nối giữa các nút.

Có 2 loại giản đồ trạng thái được chọn và biểu diễn cho 2 loại máy trạng thái khác nhau: Loại MOORE và loại MEALY

6.2.1 Giản đồ trạng thái kiểu MOORE

Các trạng thái ra phối hợp với các trạng thái đặt biệt. Vì vậy, xuất hiện các đánh số kèm theo tên trạng thái hoặc số ở các nút.

Các chuyển tiếp chỉ đánh số với tổ hợp vào tạo ra chuyển tiếp.



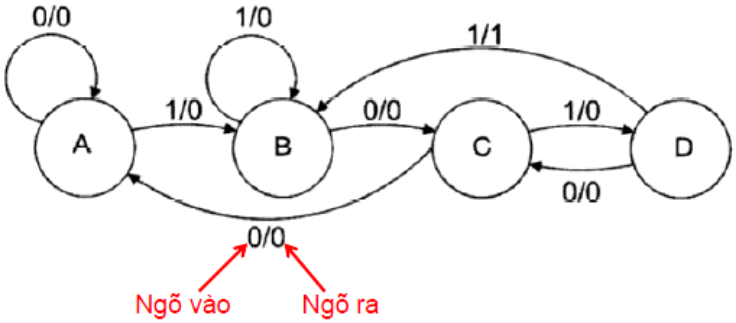
Hình 6.3: Giản đồ trạng thái kiểu Moore

6.2.2 Giản đồ trạng thái kiểu MEALY

Các trạng thái ra được vẽ như là các hàm của các trạng thái vào và của các biến trạng thái nội.

Các đường dẫn chuyển tiếp được đánh số với tổ hợp các giá trị vào làm chuyển tiếp và tổ hợp các giá trị ra được tạo ra từ phép chuyển tiếp.

Ở một trạng thái xác định giá trị của bit ngõ vào x quyết định giá trị ngõ ra → Số trạng thái ít hơn 1 trạng thái so với máy Moore



Hình 6.4: Giản đồ trạng thái kiểu Mealy

6.3 LẬP TRÌNH FPGA THÔNG QUA MÁY TRẠNG THÁI

6.3.1 Giới thiệu

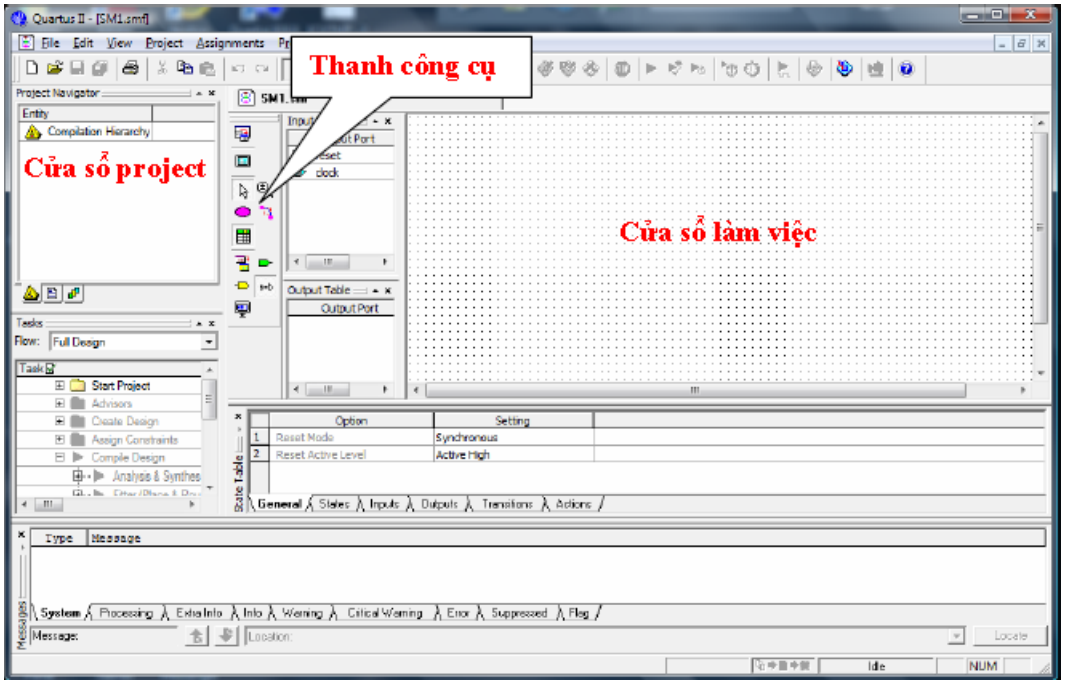
Quy trình thiết kế FPGA thông thường khá phức tạp đòi hỏi người thiết kế phải thông qua khá nhiều bước phức tạp. Máy trạng thái là một trong những bước trung gian giúp cho quá trình thiết kế và hiện thực code verilog trở nên dễ dàng hơn đối với người lập trình. Thông thường có 2 dạng máy trạng thái: Moore và Mealy. Tuy nhiên trong Quartus chỉ hổ trợ máy trạng thái Moore.

6.3.2 Mở của sổ soạn thảo máy trạng thái

Bước 1. Mở Quartus và tạo mới một project (cách tạo như đã trình bày trong bài 2)

Bước 2. Kế tiếp bạn vào **File -> New** để tạo mới một file

Bước 3. Trong cửa sổ tạo mới file bạn chọn loại State Machine file, sau đó nhấn OK. Cửa sổ soạn thảo sẽ xuất hiện như hình vẽ

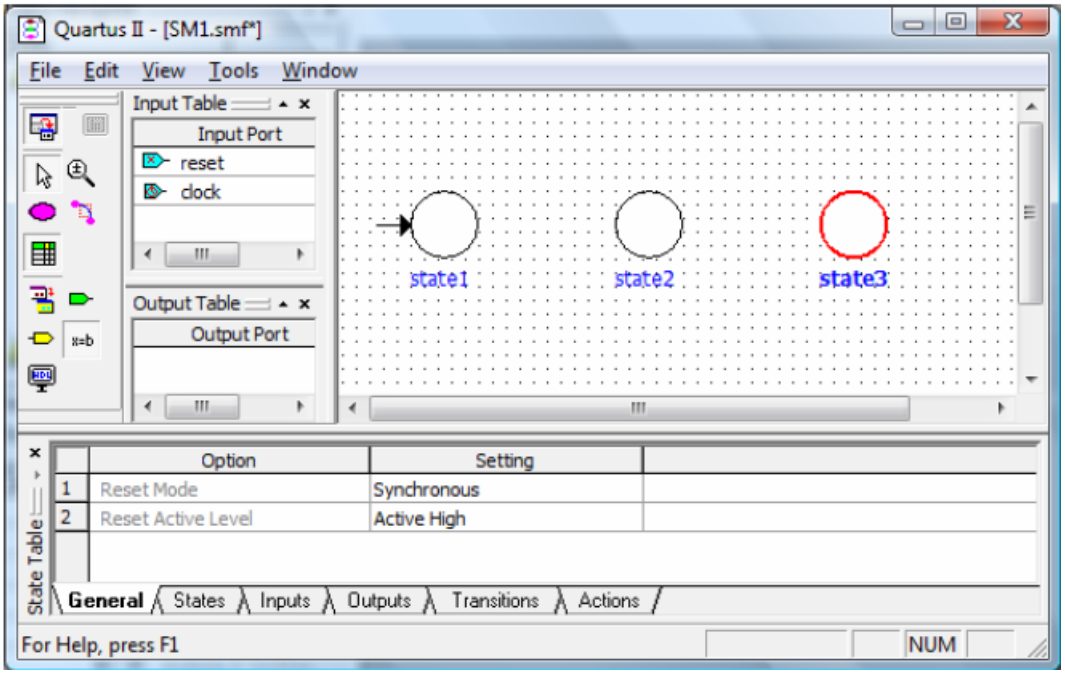


Hình 6.5: Cửa sổ soạn thảo State machine file

6.3.3 Qui trình tạo máy trạng thái

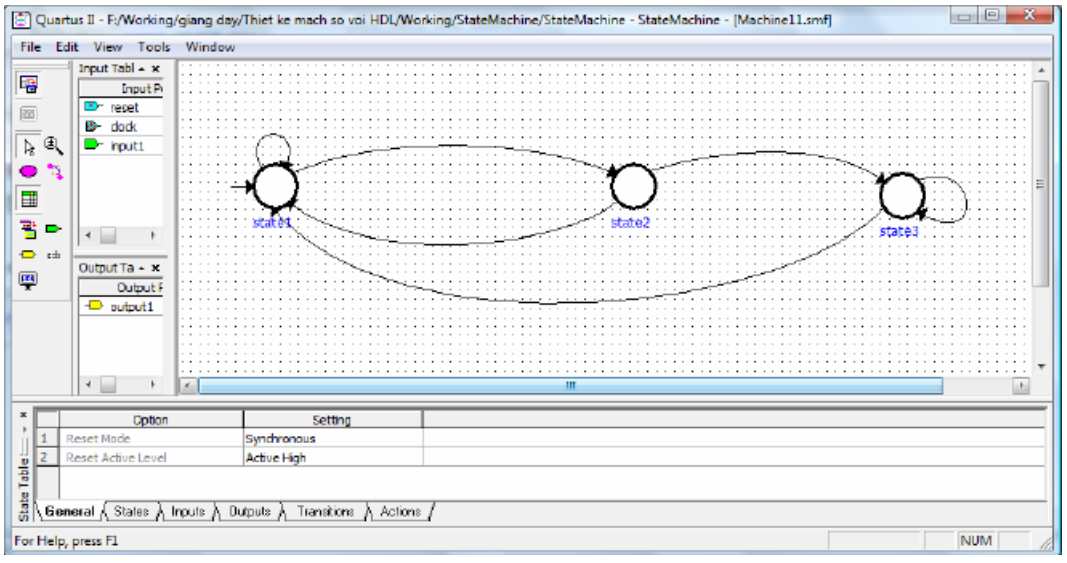
Để cụ thể sau đây là qui trình tạo ra một máy trạng thái có khả năng nhận diện chuỗi 2 bit 1 liên tiếp. Sau đó chuyển file này thành code verilog.

Bước 1. Bạn nhấp vào công cụ (State tool) sau đó vẽ 3 trạng thái như hình vẽ



Hình 6.6: Tạo các trạng thái

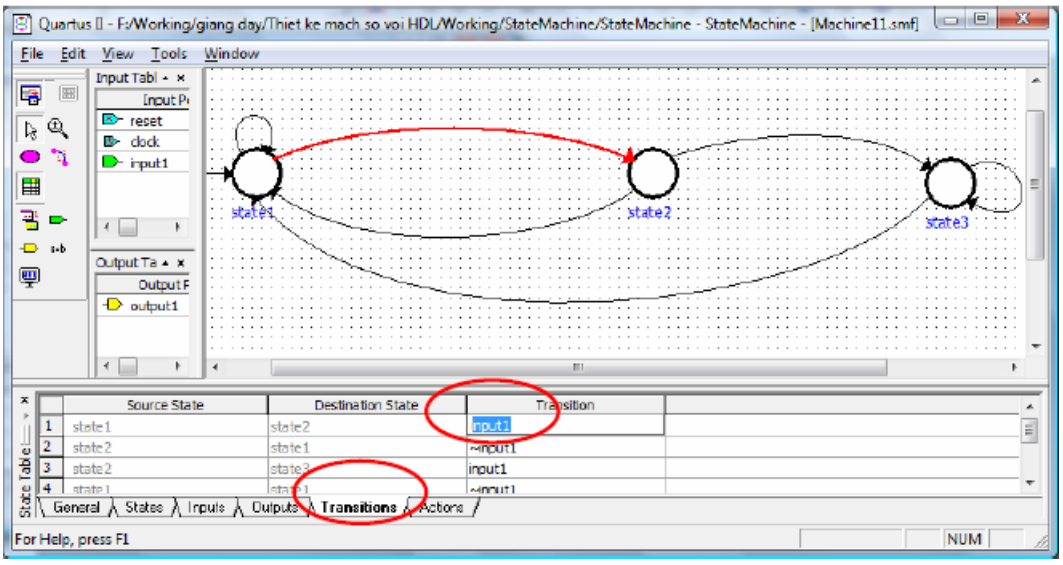
Bước 2. Tiếp theo bạn nhấp vào công cụ (Transition tool) và drag chuột từ trạng thái state1 đến trạng thái state2 để hình thành đường chuyển trạng thái từ trạng thái state1 sang trạng thái state2. Tương tự ta vẽ được máy trạng thái như hình vẽ (trường hợp nếu vẽ đường chuyển trạng thái vào chính nó thì chỉ cần chọn công cụ transition tool, sau đó nhấp chuột trái vào trạng thái đó).



Hình 7.7: Tạo các đường chuyển trạng thái

Bước 3. Tiếp theo bạn nhấp vào biểu tượng (Insert input port) để thêm một tín hiệu nhập và click vào biểu tượng (Insert output port) để thêm một tín hiệu xuất

Bước 4. Nhấp vào biểu tượng (State table) để hiển thị bảng trạng thái (nếu bảng này đã bị ẩn đi). Trong cửa sổ trạng thái này chọn tab Transition (xem hình)



Hình6.8: Thiết lập biểu thức chuyển trạng thái

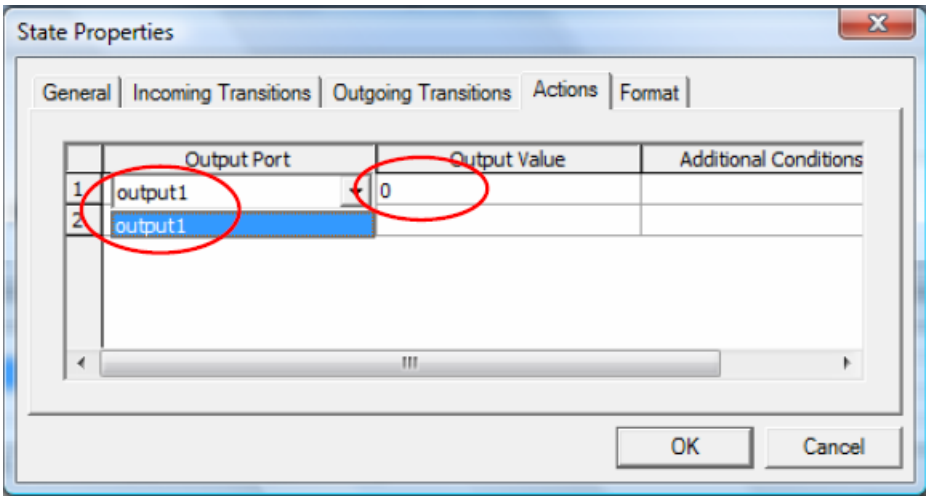
Bước 5. Nhập tín hiệu tên tín hiệu nhập vào (input1). Trường hợp nếu đường chuyển trạng thái này chỉ bị ảnh hưởng bởi duy nhất một tín hiệu nhập (input1) thì:

a. Nếu chuyển trạng thái xảy ra khi gặp một tín hiệu 0 thì nhập tên tín hiệu đó vào (input1)

b. Nếu chuyển trạng thái xảy ra khi gặp tín hiệu nhập là 1 thì nhập tên tín hiệu đó và phía trước có dấu ~ *(~input1*, đảo tín hiệu *input1*) Trường hợp đường chuyển trạng thái bị ảnh hưởng bởi nhiều tín hiệu thì thêm dấu & giữa các tín hiệu đó (ví dụ: *input1 & input2*).

Bước 6. Tương tự như vậy đối với các đường chuyển trạng thái khác.

Bước 7. Kế tiếp bạn chọn trạng thái state1 sau đó nhấp chuột phải chọn properties. Cửa sổ properties sẽ xuất hiện, bạn chọn tab Action (xem hình)



Hình 6.9: Thiết lập tín hiệu xuất & Action của mỗi trạng thái

Bước 8. Trong cột Output Port bạn chọn tín hiệu xuất, ở cột Output Value bạn thiết lập giá trị xuất ra cho trạng thái đó. Sau đó nhấn OK

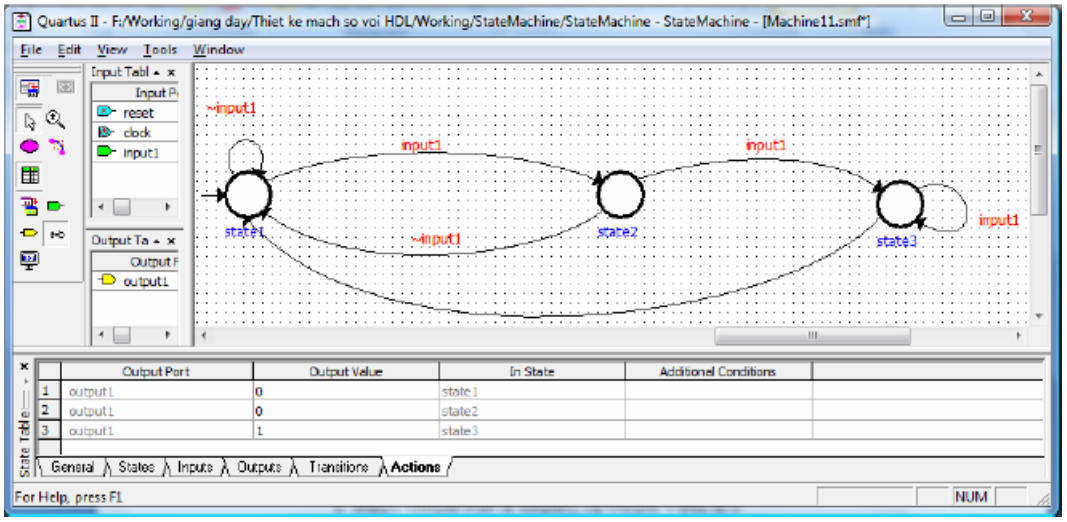
Bước 9. Lặp lại thao tác này cho 2 trạng thái còn lại.

a. *State1*: Output Port là output1, và Output Value là 0

b. *State2*: Output Port là ouput1, và Output Value là 0

c. *State3*: Output Port là output1, và Output Value là 1

Bước 10. Cuối cùng bạn sẽ được máy trạng thái như hình vẽ:

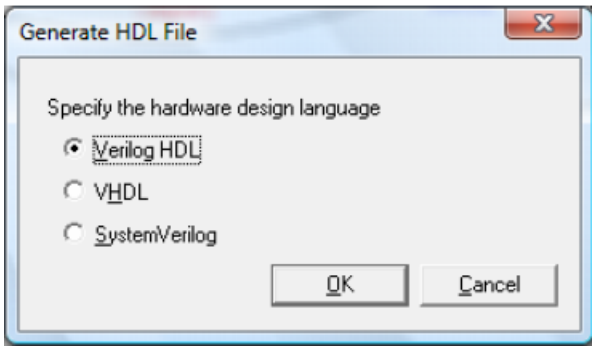


Hình 6.10: Máy trạng thái phát hiện chuỗi 2 bit 1

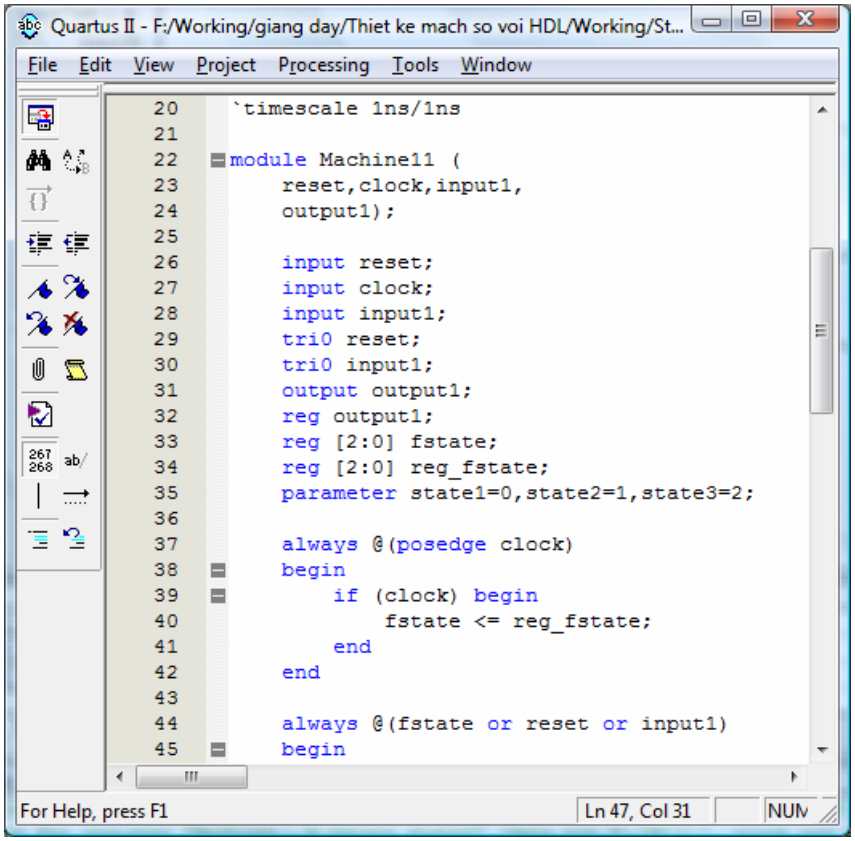
Bước 11. Nhấn Ctrl + S, hoặc nhấp vào biểu tượng trên thanh công cụ để lưu lại máy trạng thái này với tên là Machine11.smf.

6.3.4 Tạo file ngôn ngữ mô tả phần cứng từ máy trạng thái

Nhấn vào biểu tượng (Generate HDL file) trên thanh công cụ. Khi bản thông báo sinh file HDL xuất hiện bạn nhấp vào lựa chọn Verilog HDL sau đó nhấn OK, Quartus sẽ sinh ra một file Verilog (Machine11.v) từ sơ đồ máy trạng thái này.



Hình 6.11: Chuyển máy trạg thái sang ngôn ngữ mô tả phần cứng



Hình 6.12: Code Verilog được tạo từ máy trạng thái

TÓM TẮT

Trong bài này, học viên nắm được:

+ Máy trạng thái và các loại máy trạng thái

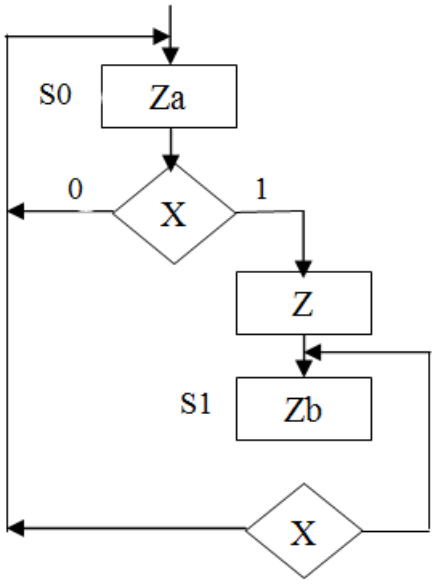
+ Giản đồ trạng thái và các loại giản đồ trạng thái

+ Các soạn thảo một thiết kế thông qua máy trạng thái trên Quartus

Tạo mã ngôn ngữ đặc tả phần cứng từ máy trạng thái.

CÂU HỎI ÔN TẬP

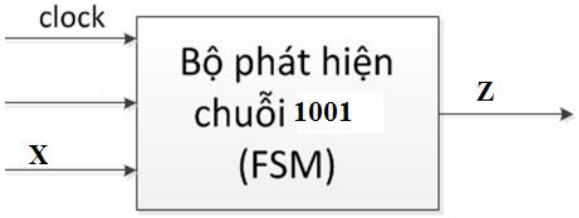
Câu 1: Cho lưu đồ trạng thái như hình vẽ:



a) Vẽ giản đồ trạng thái của lưu đồ trạng thái trên

b) Thực hiện máy trạng thái này trên FPGA

Câu 2: Cho hệ thống số sau:



X ngõ vào nối tiếp. Z = 1 khi nhận được chuỗi bit 1001 (bit MSB đi trước).

a) Vẽ giản đồ trạng thái

b) Thực hiện máy trạng thái này trên FPGA

TÀI LIỆU THAM KHẢO

1. Altera (2007). *DE2 Development and Education Boardard User Manual*. Altera Corporation.
2. Richard B. Alcorn (1997). *A digital circuit design implementation using ABEL-HDL and programmable logic device*. Youngstown State University.
3. Nguyễn Trọng Hải (2006). *Bài giảng kỹ thuật số.* ĐH công nghệ TP.HCM.
4. ĐH Bách khoa TP.HCM (2009). *Thực hành thiết kế mạch số với HDL.* Khoa khoa học và kỹ thuật máy tính.
5. Zainalabedin Navabi (2006). *Verilog Digital System Design*. McGraw-Hill.
6. Douglas L. Perry (2002). VHDL: Programming by Example. McGraw-Hill.