

THỰC HÀNH THIẾT KẾ IC

GVHD: NGUYỄN ĐỨC TIẾN | IT4251 | V1.2

Tài liệu này chứa các bài thực hành bổ sung, để sinh viên làm quen với kit DE2-150i, sau đó triển khai các thiết kế với FPGA Cyclone trên kit này.

HỌ TÊN SINH VIÊN: _____

NHIỆM VỤ	DUE DATE	DONE	INITIALS
Tìm hiểu công cụ thực hành	[Date]	<input type="checkbox"/>	
Cài đặt các công cụ thiết kế và lập trình	[Date]	<input type="checkbox"/>	
Bài 1: Nạp thiết kế lên kit DE2i-150	[Date]	<input type="checkbox"/>	
Bài 2: Thiết kế đơn giản, nối dây, gắn chân pin với Atera Quatus	[Date]	<input type="checkbox"/>	
Bài 3 : Bộ giải mã led 7 đoạn	[Date]	<input type="checkbox"/>	
Bài 4 : Đồng hồ đếm thời gian và đếm số lần bấm nút	[Date]	<input type="checkbox"/>	

BÀI TẬP LỚN	DUE DATE	DONE	INITIALS
Bài tập lớn 1. IC tạo hiệu ứng ánh sáng cho led dải	[Date]	<input type="checkbox"/>	
Bài tập lớn 2. Đồng hồ đếm ngược	[Date]	<input type="checkbox"/>	
Bài tập lớn 3. Bộ ALU	[Date]	<input type="checkbox"/>	
Bài tập lớn 4. Hộp nhạc	[Date]	<input type="checkbox"/>	
Bài tập lớn 5. Hiệu ứng màu RGB	[Date]	<input type="checkbox"/>	
Bài tập lớn 6. Trò chơi quả bóng đập tường	[Date]	<input type="checkbox"/>	

Tìm hiểu công cụ thực hành

Vài dòng sơ lược về kit DE2i-150

- Các kit DE2i-150 mà sinh viên đang thực tập là **DE2i-150 Rev.C**, đính kèm **module Wireless type A (Version A)**.
- Trang chủ và các tài liệu liên quan:

<http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=163&No=529&PartNo=5>

DE2i-150

Overview Specification Layout Kit Contents Resources Order Now

DE2i-150 FPGA Development Kit

Like 100 people like this. Be the first of your friends.

[How to distinguish Version A and Version B board ?](#)

BIOS

Title	Linux Kernel	Size(KB)	Date Added	Download
DE2i-150 BIOS	58	-	2013-07-24	
Alternative DE2i-150 BIOS by WinZent	1.0.0	-	2015-03-17	

Documents

Title	Version	Size(KB)	Date Added	Download
DE2i-150 FPGA System User Manual	1.2	10415	2014-11-13	
DE2i-150 Win7 User Manual	1.0	2643	2014-05-09	
DE2i-150 Getting Started Guide	1.0	1397	2013-06-14	
DE2i-150 Quick Start Guide	1.2	1301	2013-06-14	
My First FPGA	1.0	2114	2013-06-14	
My First NiosII	1.0	5452	2013-06-14	



Download:

- CD Altera Complete Design Suite DVD 1 of 1, 7.8G
- DE2i-150_SystemCD, 140MiB, đã có tất cả các file pdf như trong ảnh trên

Hướng dẫn đọc tài liệu

Tải về các tài liệu theo đường link trên. Cách đọc tài liệu như sau

DE2i-150_QSG.pdf, tài liệu cần đọc đầu tiên

Đọc kỹ để nhớ vị trí các nút bấm, các chức năng trên kit.

- 2 trang brochure giới thiệu kit (có bản cứng kèm theo khi mua kit)

DE2i-150_Getting_Started_Guide.pdf, tài liệu cần đọc thứ 2

- Cách bật kit lên để chạy
- Cài đặt phần mềm Quartus lên máy tính của SV để làm việc. Trong bộ 5 DVD cài đặt đính kèm kit, SV chỉ cần cài đĩa Altera Complete Design Suite FREE PACKAGE DVD 1 of 1. (4 đĩa còn lại dành cho phiên bản Quartus có phí thuê bao, và các thư viện của họ FPGA không có trên kit).
- Cài đặt USB-Blaster driver trên máy SV để thông qua dây, điều khiển hoạt động của FPGA, hoặc là để nạp cấu hình mới cho FPGA. Lưu ý rằng USB Blaster là module để giao tiếp FPGA, đã được tích hợp sẵn trên kit DE2i-150, nên không cần phải mua hay cắm thêm bất cứ thiết bị nào bên ngoài
- Hướng dẫn dùng Quartus kết nối với kit DE2i-150 qua usb (với USB-Blaster driver) để nạp cấu hình cho FPGA.
Lưu ý khi nạp thử chương trình My_First_FPGA, ở bước 10 trong tài liệu, trang 21. Khi chạy trên kit thực tế, ghi nhận rằng chương trình Programmer khi chạy auto detect sẽ phát hiện thiết bị FPGA EP4CGX150; sau khi bổ sung file sof sẽ tìm thêm thiết bị EP4CGX150DF31 nữa. Để chạy đúng, hãy xóa thiết bị EP4CGX150, để giao diện còn lại giống hệt như trong ảnh ở bước 10.

DE2i-150_Win7_User_Manual.pdf

Hướng dẫn cài đặt Windows7 lên Kit DE2i-150. Nội dung tài liệu gồm có

- Hướng dẫn tạo file iso từ đĩa DVD cài Windows7
- Hướng dẫn cài tool và tạo USB Boot với bộ cài Windows 7
- Cài đặt Windows 7 từ USB
- Đường link download các driver của DE2i-150 cho Windows 7
http://www.terasic.com/downloads/cd-rom/de2i-150/Drivers_for_CPU_System/
hoặc lấy trong thư mục DE2i-150_driver, đã được download sẵn
- Hướng dẫn giao tiếp giữa CPU và FPGA thông qua PCIe với ngôn ngữ C++ với 2 project demo

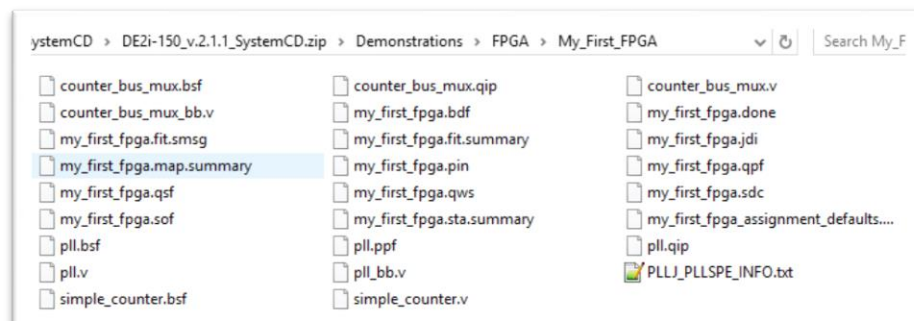
Thư mục DE2i-150_driver

Để cài đặt driver cho OS chạy trên chip Atom trên DE2i-150

- Hỗ trợ driver cho 3 loại OS: Linux, Windows7, Windows XP
- Các driver gồm: Audio, Ethernet, Graphic Driver, Wifi

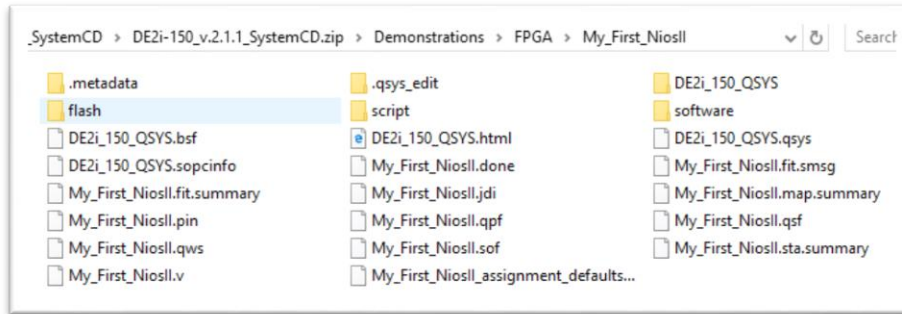
My_First_Fpga.pdf, dự án demo

- Mã nguồn của project có sẵn trong đĩa CD đính kèm kit, DE2i-150_v.2.1.1_SystemCD, trong mục \Demonstrations\FPGA\



My_First_NiosII.pdf, dự án demo

- Mã nguồn của project có sẵn trong đĩa CD đính kèm kit, DE2i-150_v.2.1.1_SystemCD, trong mục \Demonstrations\FPGA\



DE2i-150_FPGA_System_manual.pdf

Tài liệu kỹ thuật chi tiết: pin, ghép nối màn hình, ghép nối PCIe...

Đĩa DE2i-150_SystemCD

THƯ MỤC	Ý NGHĨA
D: Datasheet\	<p>Đặc tả chi tiết thông số để ghép nối IC (<i>tương đương với mô tả Entity trong VHDL</i>)</p> <p>Bao gồm: Intel Atom-d2000, FPGA Cyclone 4, Gigabit Ethernet Transceiver, Cảm biến gia tốc số ADXL345, Cảm biến thu nhận hồng ngoại IRM-V538N7/TR1, màn LCD 16x2, bộ nhớ EEPROM, bộ nhớ Flash, bộ nhớ động SDRAM, bộ nhớ tĩnh SSRAM, bộ giải mã tín hiệu Video (phục vụ ghi hình), Chuyển đổi DAC 3 kênh 10-bit tích hợp (phục vụ xuất hình)</p>
D: Demonstrations\FPGA\DE2i_150_Default\	
D: Demonstrations\FPGA\DE2i_150_Golden_top\	
D: Demonstrations\FPGA\DE2i_150_Gsensor\	
D: Demonstrations\FPGA\DE2i_150_IR\	
D: Demonstrations\FPGA\DE2i_150_SD_CARD\	
D: Demonstrations\FPGA\DE2i_150_TV\	
D: Demonstrations\FPGA\DE2i_150_Web_Server\	
D: Demonstrations\FPGA\EPCS_Patch\	
D: Demonstrations\FPGA\My_First_FPGA\	

D: Demonstrations\FPGA\My_First_NiosII\	
D: Demonstrations\FPGA\PCIE_Display\	
D: Demonstrations\FPGA\PCIE_Fundamental\	
D: Demonstrations\PCIE_SW_Kit\	Các driver, thư viện cài đặt trên Windows, Linux để các phần mềm trên đó có thể giao tiếp được với các module vào ra được thiết kế ở nửa board FPGA Terasic.
F: Schematic\DE2i_150.pdf	Sơ đồ nguyên lý thiết kế chi tiết kit DE2i-150
D:Tools\ControlPanel\	
D:Tools\SystemBuilder\	
D: User manual\	Chứa các tài liệu, giống như tải về từ trang chủ của kit

Cài đặt các công cụ thiết kế và lập trình

Altera Complete Design Suite

Đây là bộ công cụ hoàn chỉnh gồm:

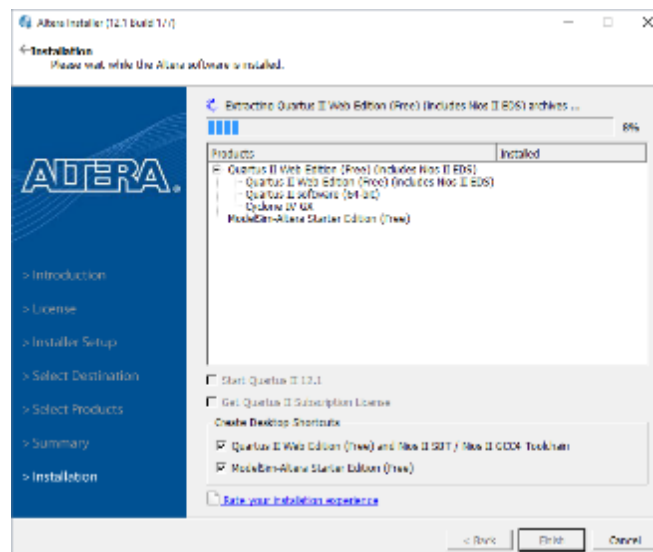
1. Phần mềm thiết kế và lập trình VHDL/Verilog của công ty Altera.
2. MegaCore IP Library – OpenCore Plus evaluation:
 - a. IP Base Suite (full license): các thư viện thiết kế sẵn, chỉ việc kéo thả và sử dụng luôn
 - b. Nios II Processor (evaluation license): thiết kế của bộ xử lý Nios II chỉ tương thích với FPGA của Altera. Bao gồm cả trình biên dịch, bộ giả lập để viết phần mềm cho Nios II. Thiết kế của Nios II có thể tùy biến như thay đổi tập lệnh, bổ sung các module..nhưng bị giới hạn bởi license.
 - c. Model SIM – Altera Starter Edition: bộ giả lập của hãng ModelSIM, mạnh hơn so với trình giả lập sẵn có trên Quartus.

Các bước cài đặt xem trong tài liệu DE2i-150_Getting_Started_Guide.pdf, Chapter 2. Trong đó lưu ý một số bước là

- Khởi động từ đĩa <DVD 1 of 1>



- Với các dialog tiếp theo, và bấm <Next>. Chú ý ở phần tùy chọn các mục cài đặt hãy chọn **thư viện Cyclone IV GX**.



Driver để kết nối với module nạp FPGA nằm sẵn trên kit

Các bước cài đặt xem trong tài liệu DE2i-150_Getting_Started_Guide.pdf, Chapter 4.

Lưu ý: đã test thành công trên windows 10.

Trợ giúp và mã nguồn

Mã nguồn

Lưu ý: Mã nguồn từ GitHub có thể lấy về bằng công cụ sẵn có của GitHub, hoặc sử dụng TortoiseHG (với extension hgit), hoặc download cả project về rồi sử dụng độc lập.

https://github.com/neittien0110/lab_quartus_01.git

https://github.com/neittien0110/lab_quartus_02.git

https://github.com/neittien0110/lab_quartus_03.git

https://github.com/neittien0110/lab_quartus_04.git

Bài 1: Nạp thiết kế lên kit DE2i-150

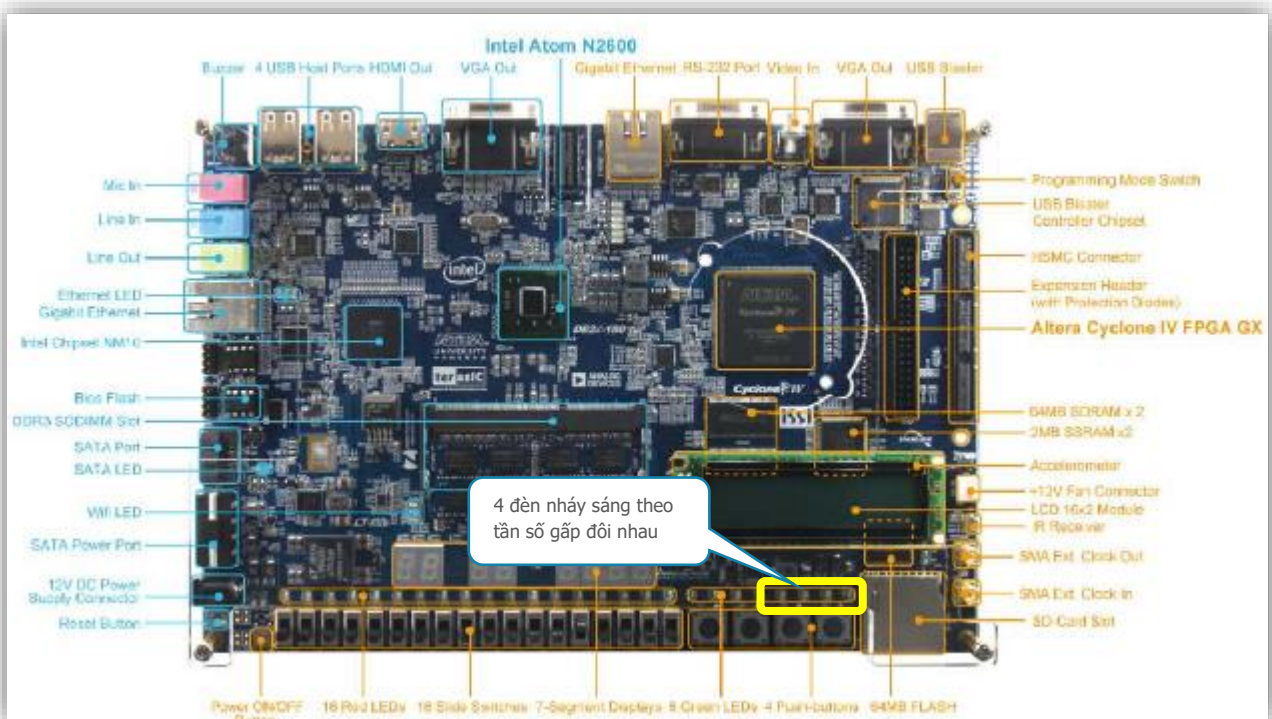
Mô tả: Nạp một file thiết kế có sẵn ở dạng nhị phân lên kit FPGA.

Ý nghĩa: Sinh viên nắm được các bước để nạp một file đặc tả cấu hình phần cứng lên chip FPGA, để FPGA hoạt động theo mong đợi.

Kiến thức cần nắm vững:

- File sof là file đặc tả cấu hình phần cứng, dạng nhị phân. File đã được biên dịch để phù hợp duy nhất với loại FPGA đang sử dụng của Altera, và phù hợp cả với main board DE2i-150. Có thể so sánh file sof giống như là file exe vậy.
Công thức : (<thiết kế HDL> + <chọn FPGA> + <chọn chân phù hợp mainboard>) ----- biên dịch bằng Quartus -----> file sof
- Thiết kế khi được nạp lên FPGA sẽ mất mỗi khi ngắt điện, vì thuộc loại RAM-based.

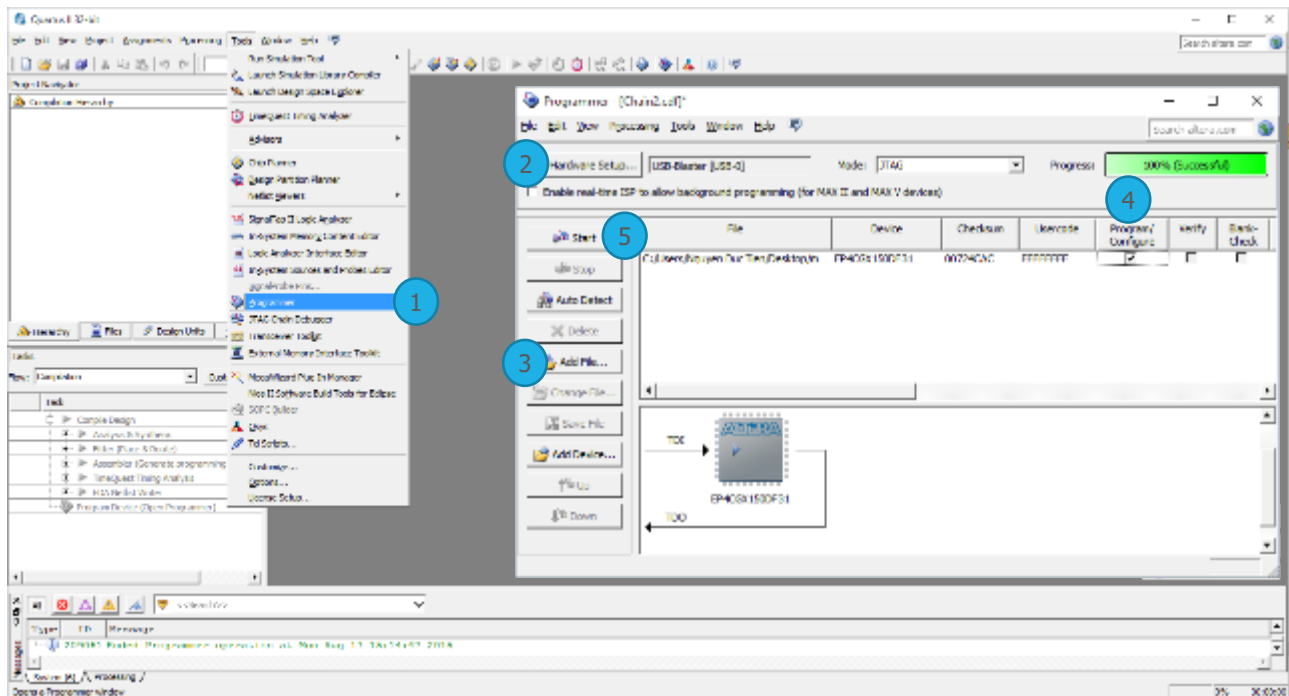
Kết quả mong đợi:



Các bước tiến hành:

- Xem trong tài liệu DE2i-150_Getting_Started_Guide.pdf, Chapter 5.

Ảnh dưới là mô tả nhanh các bước tiến hành



Bài 2: Thiết kế đơn giản, nối dây, gắn chân pin với Atera Quatus



Mô tả: Nạp một file thiết kế có sẵn lên kit FPGA.

Ý nghĩa:

- Sinh viên nắm được nguyên tắc hoạt động của các linh kiện vào/ra đơn giản như công tắc, đèn led, led 7-đoạn
- Sinh viên tạo được một dự án đơn giản với thiết kế chỉ toàn phép gán (hoặc nối dây nếu thiết kế bằng block diagram)
- Sinh viên nắm được cách đọc datasheet của kit DE2i-150 để tìm ra các thông tin cần thiết để biết đặc tính của các linh kiện vào/ra trên kit như thế nào, từ đó kiểm tra thiết kế vừa làm.
- Sinh viên nắm được cách ánh xạ, gắn chân IC ảo (thiết kế bởi HDL) với chip FPGA thật.

Yêu cầu công việc:

Hãy lập trình điều khiển các

- công tắc dip switch SW17, SW16 
- dải đèn led LEDR17..LEDR14, LEDR5..LEDR0. 
- đèn led 7 đoạn HEX0 

để sao cho:

- dải đèn led LEDR5... LEDR0 hiển thị giá trị cố định ☼ ○ ○ ○ ○ ☼
- khi SW16 ở trạng thái ON, gạt lên trên, thì đèn LEDR15, LEDR14 ở trạng thái sáng ☼ ☼
khi SW16 ở trạng thái OFF, gạt xuống, thì đèn LEDR15, LEDR14 ở trạng thái tắt ○ ○
- khi SW17 ở trạng thái ON, gạt lên trên, thì đèn LEDR17, LEDR16 ở trạng thái tắt ○ ○
khi SW17 ở trạng thái OFF, gạt xuống, thì đèn LEDR17, LEDR16 ở trạng thái sáng ☼ ☼
- khi SW6~0 thay đổi, thì 7 đoạn – 7 đèn led của đèn led 7-đoạn HEX0 sẽ sáng/tắt tương ứng.

Kiến thức cần nắm vững:

- Kit DE2i-150 sử dụng FPGA thuộc họ Cyclone 4 với tên đầy đủ là **EP4CGX150DF31N** (Ghi rõ trên vỏ IC). Khi tạo một Project trong Quatus, hãy chọn đúng **EP4CGX150DF31C7, hoặc EP4CGX150DF31C8**
- Cần biết vị trí của các công tắc dip switch SW17~0 được gán với chân pin nào của FPGA.
Xem tài liệu DE2i-150_FPGA_System_manual.pdf, Table 3-2, trang 32.
- Cần biết vị trí của các đèn LEDR17~0 được gán với chân pin nào của FPGA.
Xem tài liệu DE2i-150_FPGA_System_manual.pdf, Table 3-4, trang 33.
- Phải hiểu rõ led7 đoạn trong kit DE2 là loại Anode chung - tích cực mức thấp (đèn sáng nếu chân pin ở logic 0); hay là loại Cathode chung – tích cực mức cao (đèn sáng nếu chân pin ở logic 1).
Xem tài liệu DE2i-150_FPGA_System_manual.pdf, trang 33

Cần biết vị trí của các đèn led 7 đoạn HEX7~0 được gán với chân pin nào của FPGA.

Xem tài liệu DE2i-150_FPGA_System_manual.pdf, Table 3-5, trang 34.

Các bước tiến hành:**Giai đoạn 1: Thiết kế**

- Tạo một project mới bằng phần mềm Quartus. Ở Page 1/5, hãy điền tên của project

← Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

What is the name of this project?

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

[Use Existing Project Settings...](#)

- Page 2/5 hỏi các file mã nguồn đã có để đưa vào project mới. Không có file nào cần thêm cả. Bấm Next

← Add Files [page 2 of 5]

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.
 Note: you can always add design files to the project later.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version

[Add](#) [Add All](#) [Remove](#) [Up](#) [Down](#) [Properties](#)

Specify the path names of any non-default libraries. [User Libraries...](#)

< Back Next > Finish Cancel Help

- Page 3/5 để khai báo loại FPGA sẽ sử dụng. Hãy chọn đúng **Family = Cyclone IV GX**, và **Available devices = EP4CGX150DF31C**. Bấm Finish

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Device family

Family: **Cyclone IV GX**

Devices: All

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Available devices:

Name	Core Voltage	LEs	User I/Os	GXB Transmitter Channel PMA	GXB Receiver Channel PMA
EP4CGX150DF27C8	1.2V	149760	426	8	8
EP4CGX150DF27I7	1.2V	149760	426	8	8
EP4CGX150DF27I7AF	1.2V	149760	426	8	8
EP4CGX150DF31C7	1.2V	149760	508	8	8
EP4CGX150DF31C8	1.2V	149760	508	8	8
EP4CGX150DF31I7	1.2V	149760	508	8	8
EP4CGX150DF31I7AD	1.2V	149760	508	8	8

Companion device

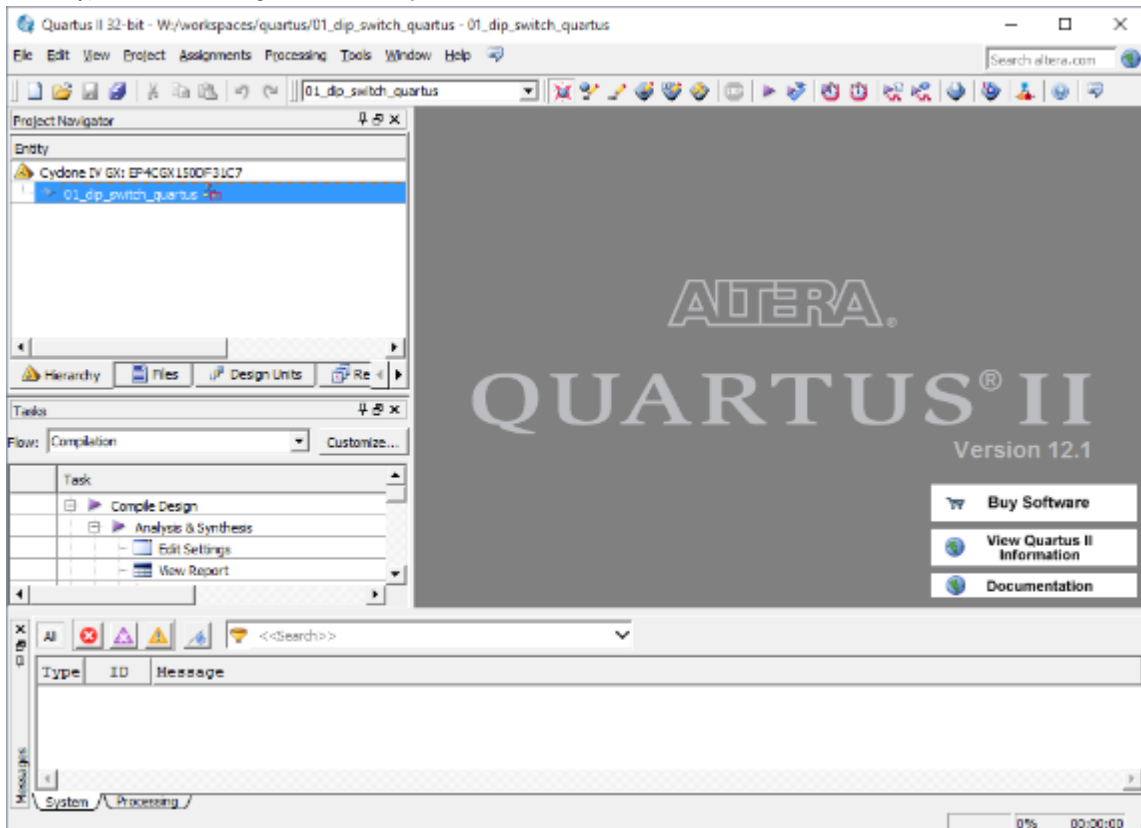
HardCopy: [Empty]

☐ Limit DSP & RAM to HardCopy device resources

☒ Show advanced devices ☐ HardCopy compatible only

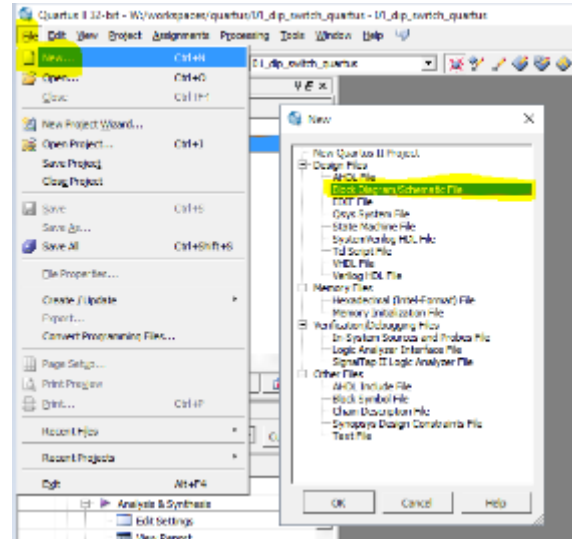
< Back Next > Finish Cancel Help

- Lúc này, cửa sổ của Project mới đã hiện ra như hình vẽ.

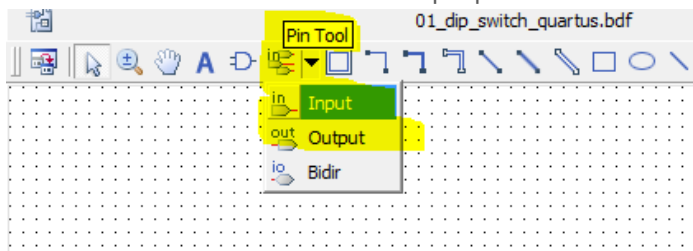


- Trong thanh menu, chọn Files/New. Sau đó, khi cửa sổ New hiện ra, hãy chọn Block Diagram/Schematic File.

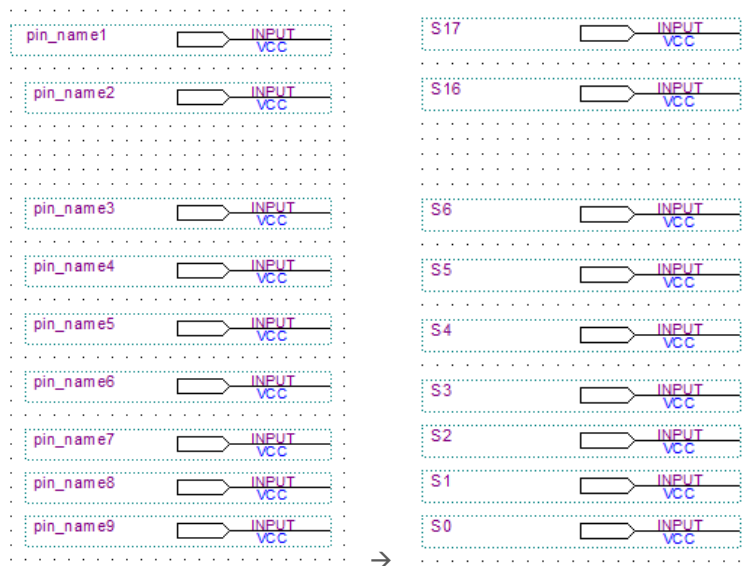
Như vậy, ta sẽ tạo một file thiết kế bằng phương pháp vẽ sơ đồ nguyên lý, sau đó Quartus sẽ tự dịch ra mã nguồn VHDL, rồi biên dịch lần nữa thành file nhị phân .so.



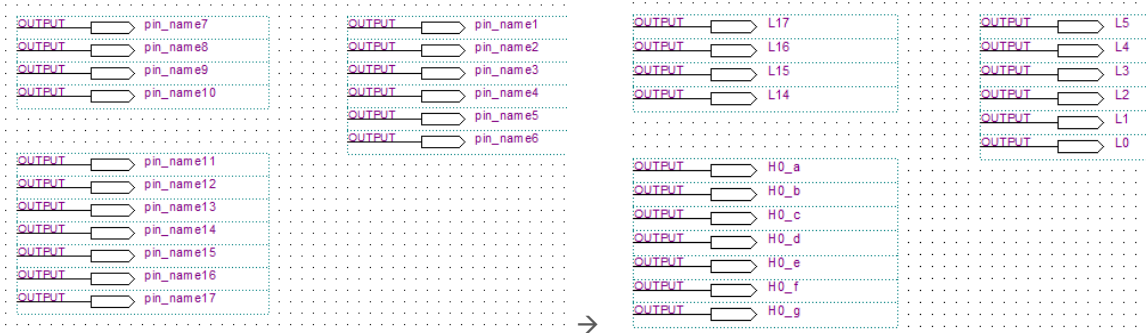
- Trước tiên, lưu file thiết kế mới tạo đã cho chắc chắn. Hãy vào menu, chọn File/Save As và nhập tên file .bdf.
- Ở cửa sổ file bdf, trong thanh công cụ Block Editor, click vào hộp combo của Pin Tool (🔌), chọn Input. Ta sẽ tiến hành mô tả các chân vào – input pin – của IC mới.



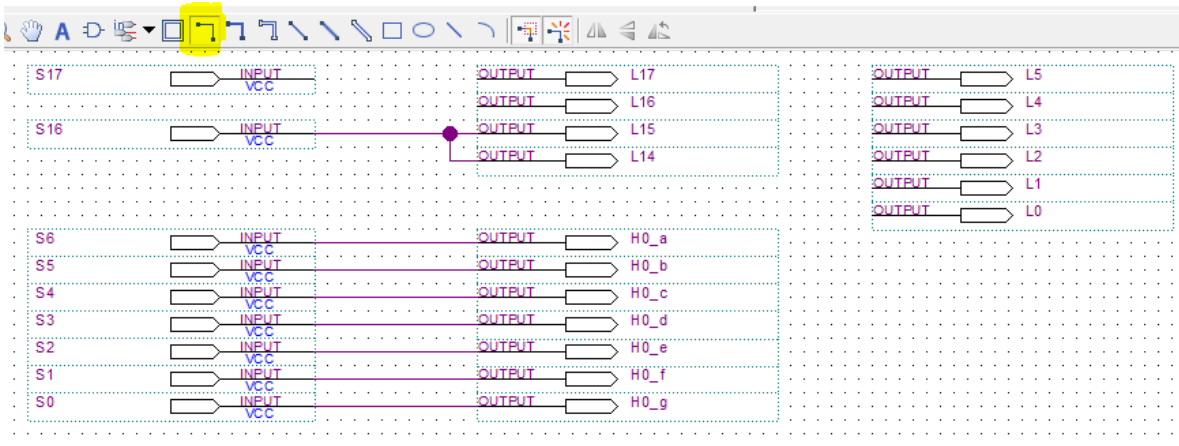
- Tạo ra 9 input pin cho IC mới (tương ứng với các công tắc Dip switch sau này), sau đó đổi tên gọi nhớ. Double click vào pin_name để đổi tên.



- Tương tự như vậy, tạo ra 17 output pin cho IC mới (tương ứng với các đèn led, led 7-đoạn sau này), sau đó đổi tên gọi nhớ.



- Theo yêu cầu thiết kế, tương ứng 1-1 giữa một công tắc với một/một vài đèn, nên thiết kế chỉ toàn các đường nối dây. Trên thanh Block Editor, sử dụng công cụ Orthogonal Node Tool



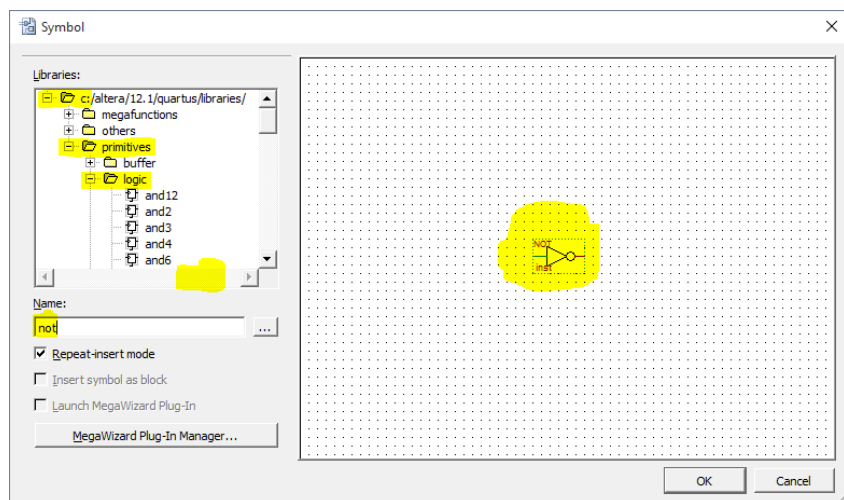
- Duy nhất có một yêu cầu về đảo chiều cách sáng của đèn nên phải có cổng NOT. Trên thanh Block Editor, bấm vào công cụ Symbol.



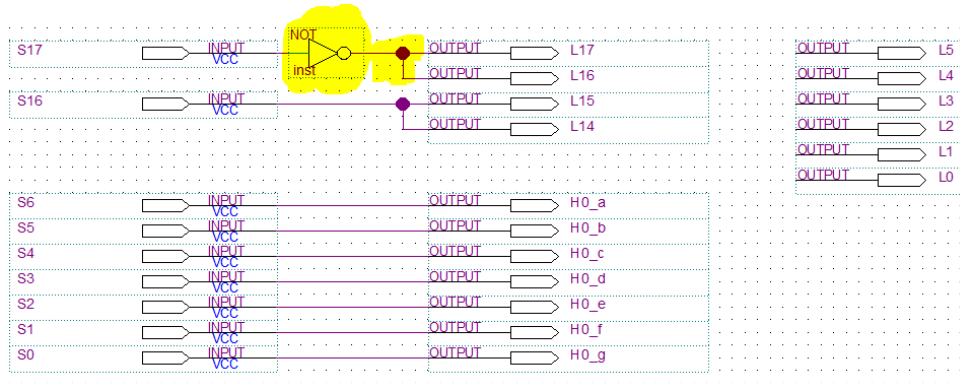
- Trong cửa sổ Symbol, ở mục Libraries, chọn ../primitives/logic/not

hoặc có thể gõ trực tiếp từ "not" vào textbox Name để thực hiện tìm kiếm.

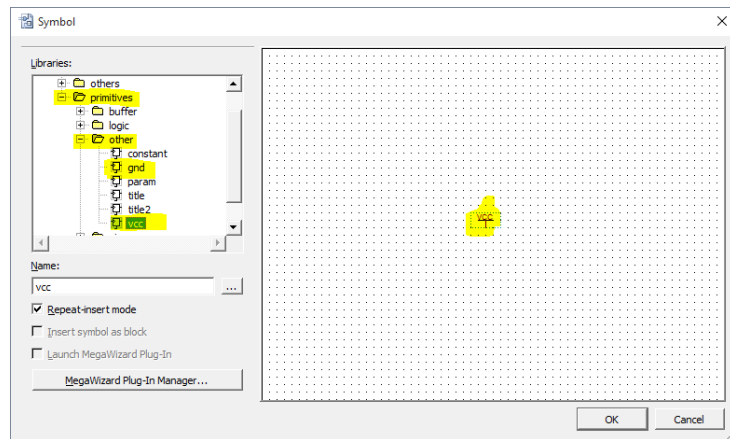
Bấm nút OK.



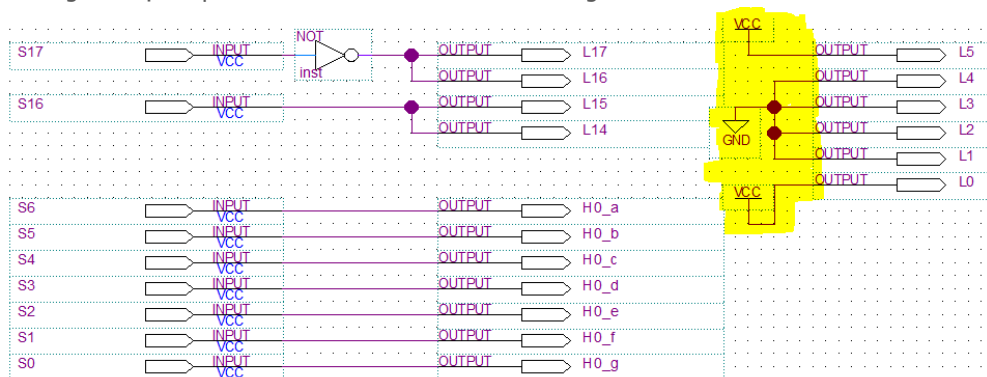
- Đưa cổng NOT vào thiết kế như trong hình vẽ



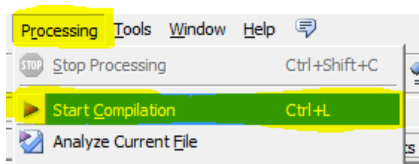
- Một số đèn led có trạng thái sáng/tắt không thay đổi, hằng số. Điều đó có nghĩa rằng, các đèn này được nối trực tiếp với nguồn điện để làm sáng/tắt, chứ không qua nút bấm nào cả. Trên thanh Block Editor, bấm vào công cụ Symbol.
- Trong cửa sổ Symbol, ở mục Libraries, chọn ../primitives/other/vcc. Bấm nút OK.
Lặp lại như vậy, chọn ../primitives/other/gnd. Bấm nút OK.



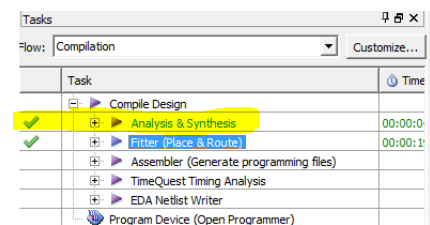
- Đưa nguồn trực tiếp VCC và GND vào thiết kế như trong hình vẽ



- Kiểm tra xem thiết kế có đúng không bằng cách biên dịch thử. Trong thanh menu, chọn Processing / Start Compilation.



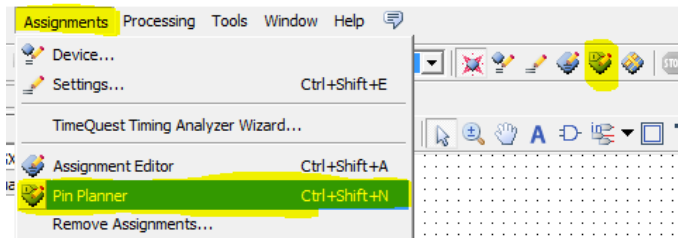
-- (biên dịch thành công) -->



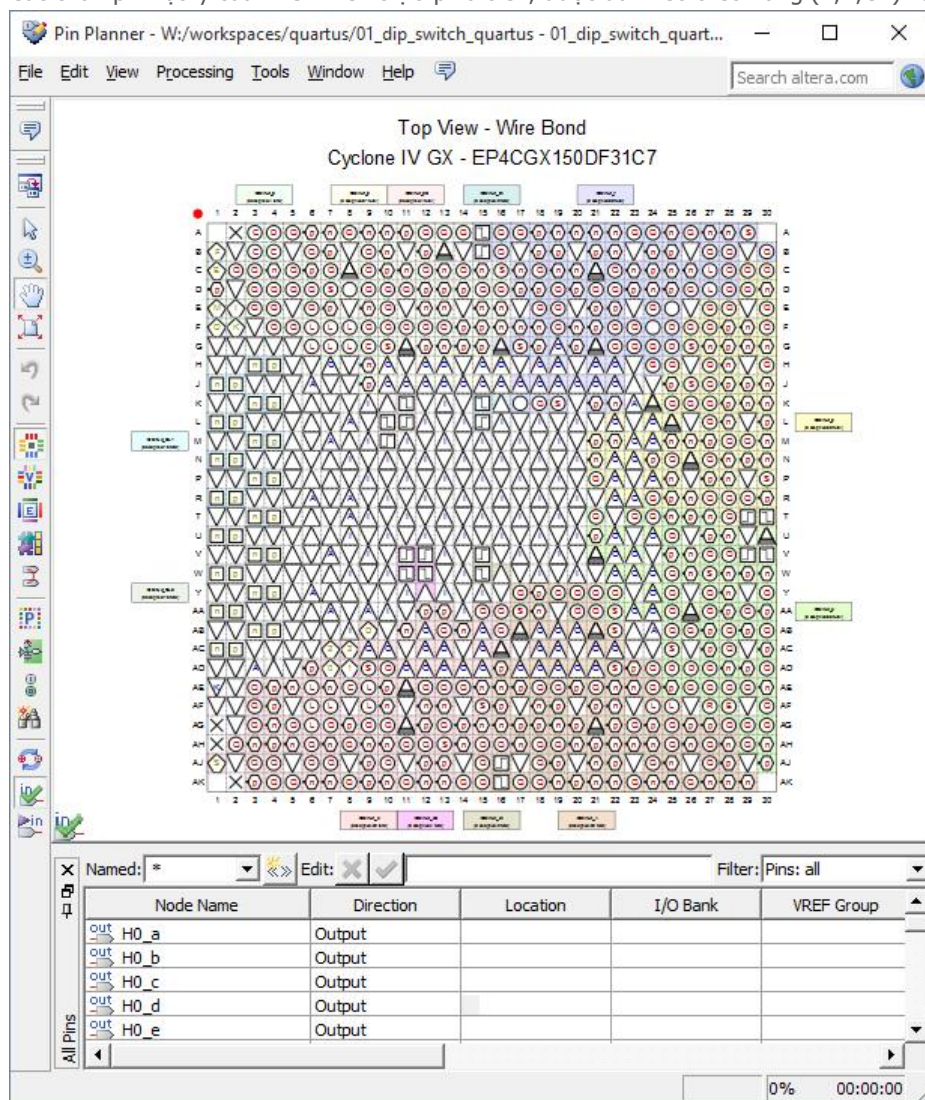
- Quá trình thiết kế IC mới tới đây hoàn tất. Chúng ta đã có một IC ảo.

Giai đoạn 2: Gán chân pin để ánh xạ giữa chân pin của IC ảo với chip FPGA thật

- Trong thanh menu, chọn Assignments / Pin Planner. Hoặc có thể bấm biểu tượng của công cụ Pin Planner trên thanh Block Editor.

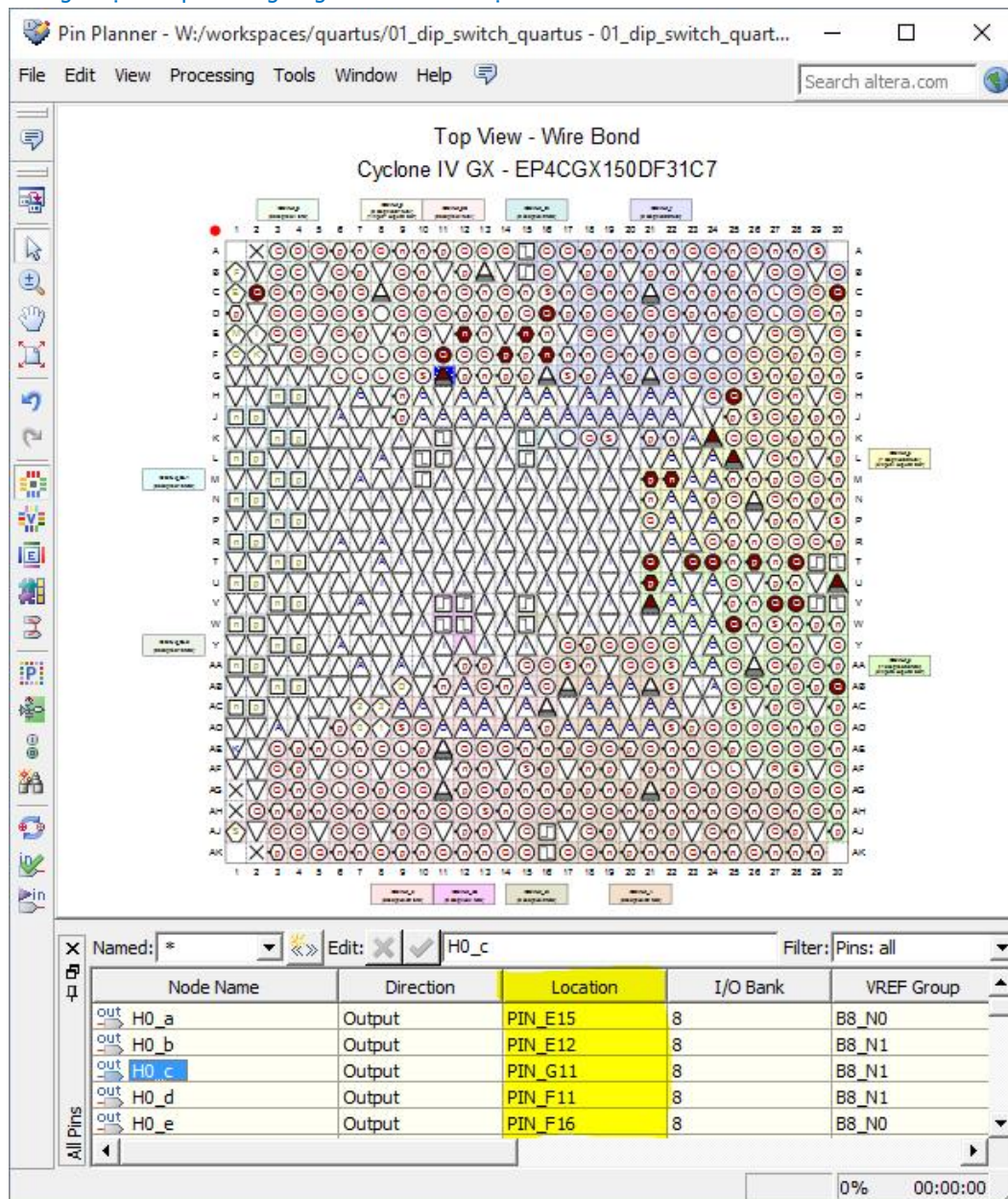


- Cửa sổ Pin Planner hiển thị như hình dưới.
 - Danh sách các chân pin của IC ảo vừa được thiết kế xong, như là H0_a, H0_b.. hiển thị ở bên dưới.
 - Các chân pin vật lý của FPGA hiển thị ở phía trên, được đánh số theo hàng (A,B,C..) và cột (1,2,3...)



- Tìm hiểu theo để biết các chân pin của FPGA được nối với đèn led, nút bấm nào trên board mạch, dựa theo các thông tin trong phần **Kiến thức cần nắm vững**. Sau đó gán các chân pin của IC ảo với FPGA thật bằng cách:

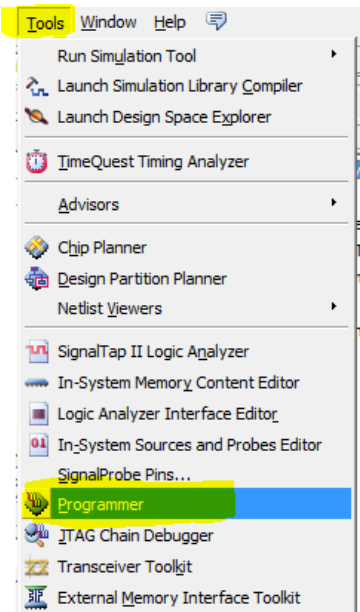
điền giá trị chân pin tương ứng của FPGA vào cột Location



- Đóng cửa sổ Pin Planner. Đã gán chân pin thành công

Giai đoạn 3: Nạp thiết kế lên FPGA

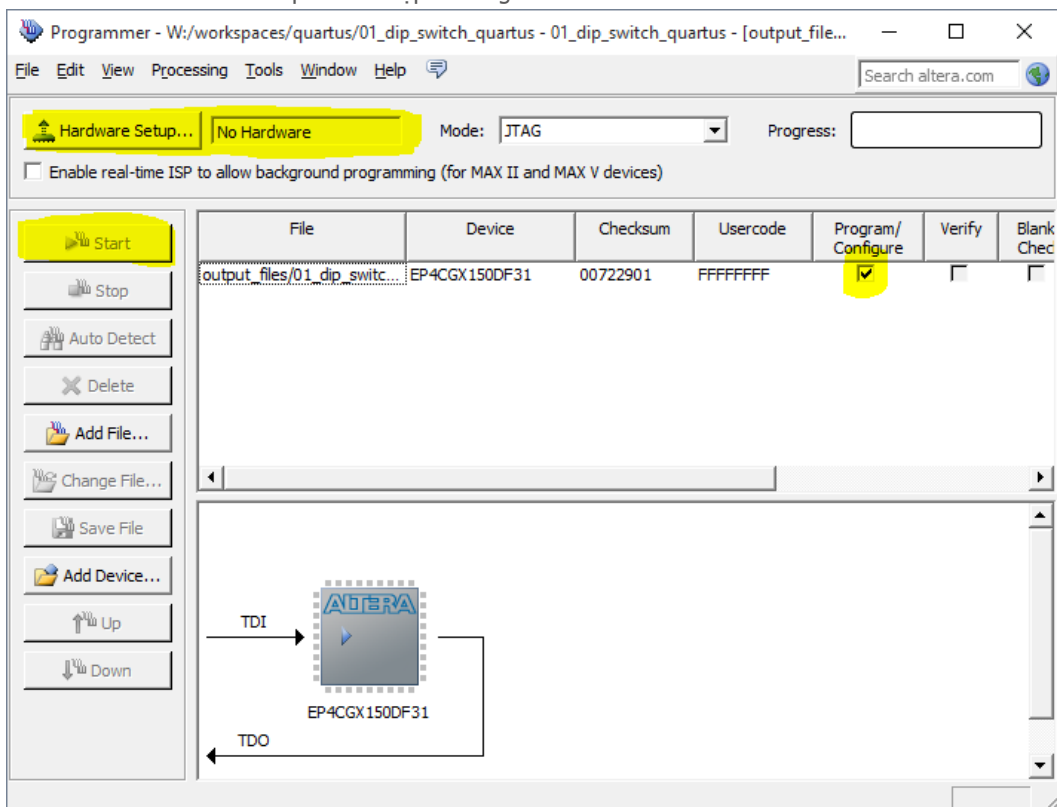
- Mở cửa sổ Programmer bằng cách vào thanh menu, chọn Tools / Programmer.



- Ở cửa sổ Programmer, file thiết kế đã biên dịch, loại chip FPGA, sẽ được tự động đưa vào cửa sổ. Không phải chỉnh sửa gì thêm.

Cần lưu ý vị trí Hardware Setup ghi giá trị *No Hardware* thì không đúng, mà cần chọn lại là **Usb Blaster** (bởi vì ta sử dụng module này để nạp).

Bấm nút Start để bắt đầu quá trình nạp là xong.



Bài 3 : Bộ giải mã led 7 đoạn

Mô tả: Sinh viên tự thiết kế bộ giải mã led 7 đoạn, đồng thời sử dụng thư viện ic giải mã led 7 đoạn có sẵn

Ý nghĩa:

- Sinh viên tự thiết kế được một mạch tổ hợp đơn giản, bộ giải mã led 7 đoạn
- Sinh viên nắm được các bước để đưa một thiết kế có sẵn (tự code hoặc sử dụng thư viện), nhúng vào một Block Diagram trên Quartus

Yêu cầu công việc:

Hãy lập trình điều khiển các

- công tắc dip switch SW7.. SW0
- đèn led 7 đoạn HEX0, Hex4, Hex6

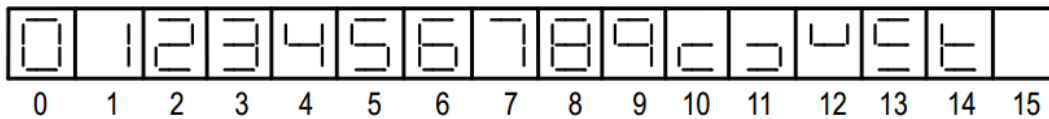


để sao cho:

- Khi đầu vào là các công tắc SW7..SW0 được gạt ở các vị trí "0000" → ... → "1001" → ... → "1111" thì đầu ra là các đèn led 7 đoạn sẽ hiển thị số 0 → ... → 9 → F.

Kiến thức cần nắm vững:

- Nhớ qui tắc giải mã, led 7 đoạn dạng anode chung hoặc cathode chung (xem slide bài giảng)

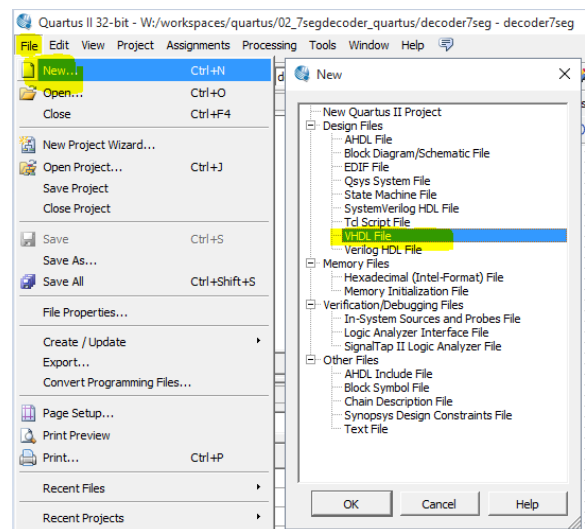


- Cần biết vị trí của các công tắc và đèn led (giống như bài 2)
- Loại đèn led mà kit DE2i-150 là loại anode chung, hay là cathode chung?
- IC 74LS247 là bộ giải mã led 7 đoạn có anode chung, IC 74LS248 là bộ giải mã led 7 đoạn có cathode chung.

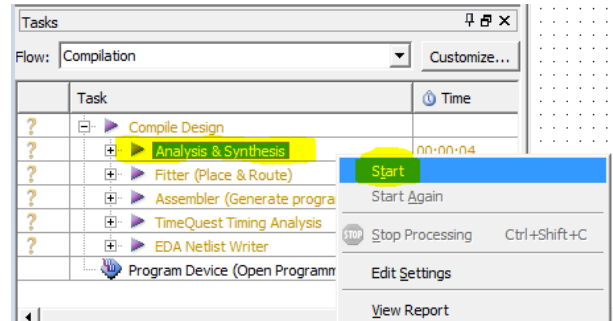
Các bước tiến hành:

Giai đoạn 1: Thiết kế và biên dịch bộ giải mã (tự code)

1. Tạo một dự án mới trên Quartus
2. Tự thiết kế bộ giải mã led 7 đoạn với (4 bit đầu vào và 7 bit đầu ra) bằng code VHDL. Tham khảo slide bài giảng. Cách bổ sung một thiết kế VHDL mới như trong ảnh.
Phần tiếp theo của tài liệu giả định rằng file tự thiết kế bộ giải mã sẽ lưu với tên decoder7seg.vhd

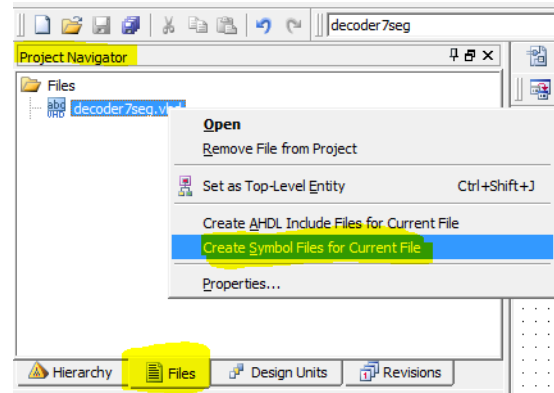


- Trong cửa sổ task, bấm vào Compile Design / Analysis & Synthesis / chọn Start. Quá trình biên dịch file VHDL sẽ được tiến hành.
Hãy bảo đảm không có lỗi xảy ra. Fix lỗi nếu có.



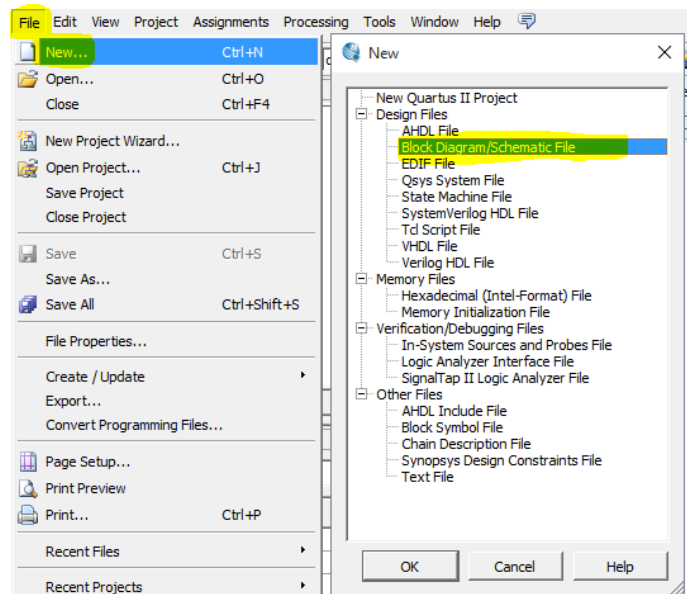
- Trong cửa sổ Project Navigator, chọn tab Files. Bấm chuột phải vào tên file VHDL vừa thiết kế, sau đó chọn **Create Symbol Files for Current File**

Mục đích của bước này là Quartus sẽ tạo ra một biểu tượng của thiết kế ở dạng Block Diagram, mà sau đó khi ta có thể nhúng biểu tượng này vào trong các Block Diagram khác (tạo component). Nếu bước này bỏ qua, sẽ không tìm thấy entity tương ứng khi mở library ở giai đoạn 2.

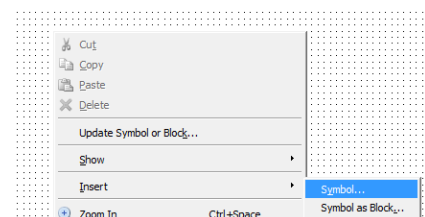


Giai đoạn 2: Tạo thiết kế mức cao hơn và nhúng các thiết kế đã làm vào bên trong (giống hàm main)

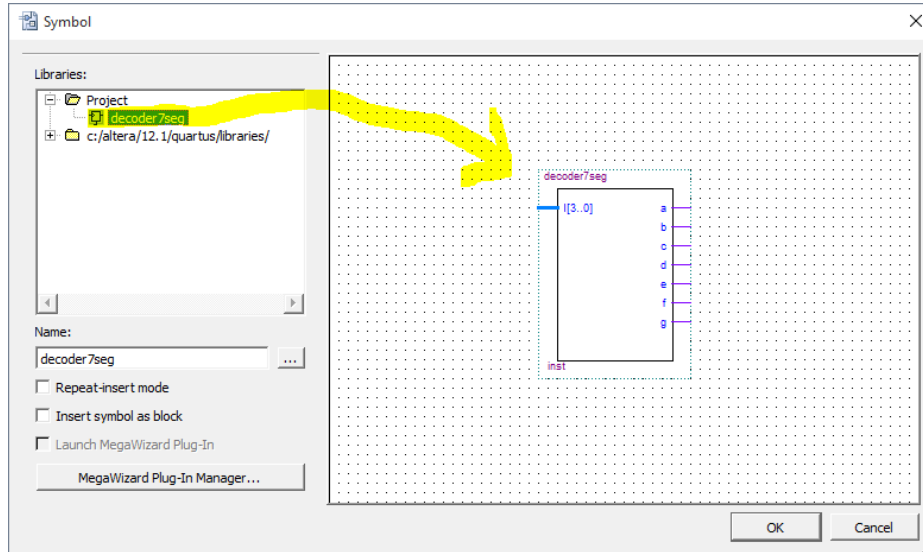
- Tạo một file thiết kế mới ở dạng Block Diagram.
Trong tài liệu này file thiết kế mới được đặt tên là topdesign.bdf



- Ở cửa sổ thiết kế Block Diagram, bấm chuột phải vào phần trống / chọn Insert / chọn Symbol...

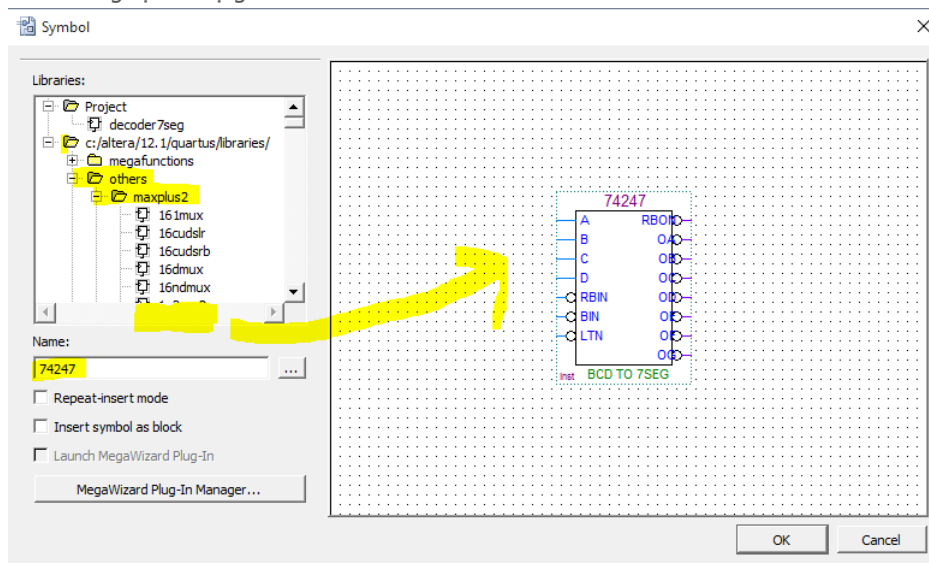


7. **Để nhúng thiết kế tự code:** Ở cửa sổ Symbol, chọn Project và chọn biểu tượng của thiết kế mà ta vừa làm xong. Sau đó, bấm chuột lên giao diện Block Diagram để đặt biểu tượng của bộ giải mã lên topdesign.bdf.

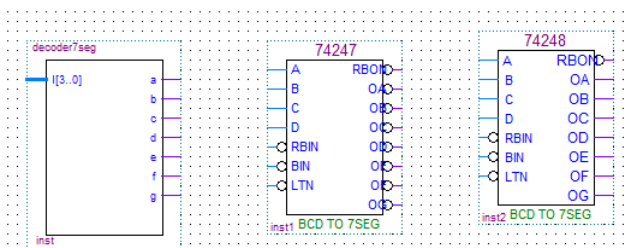


8. **Để nhúng thiết kế bộ giải mã 74LS247, 74LS248 có sẵn:** Ở cửa sổ Symbol, chọn others/maxplus2 và tìm tới tên 74247. Sau đó, bấm chuột lên giao diện Block Diagram để đặt biểu tượng của bộ giải mã led 7 đoạn lên topdesign.bdf.

Làm tương tự với bộ giải mã 74LS248.

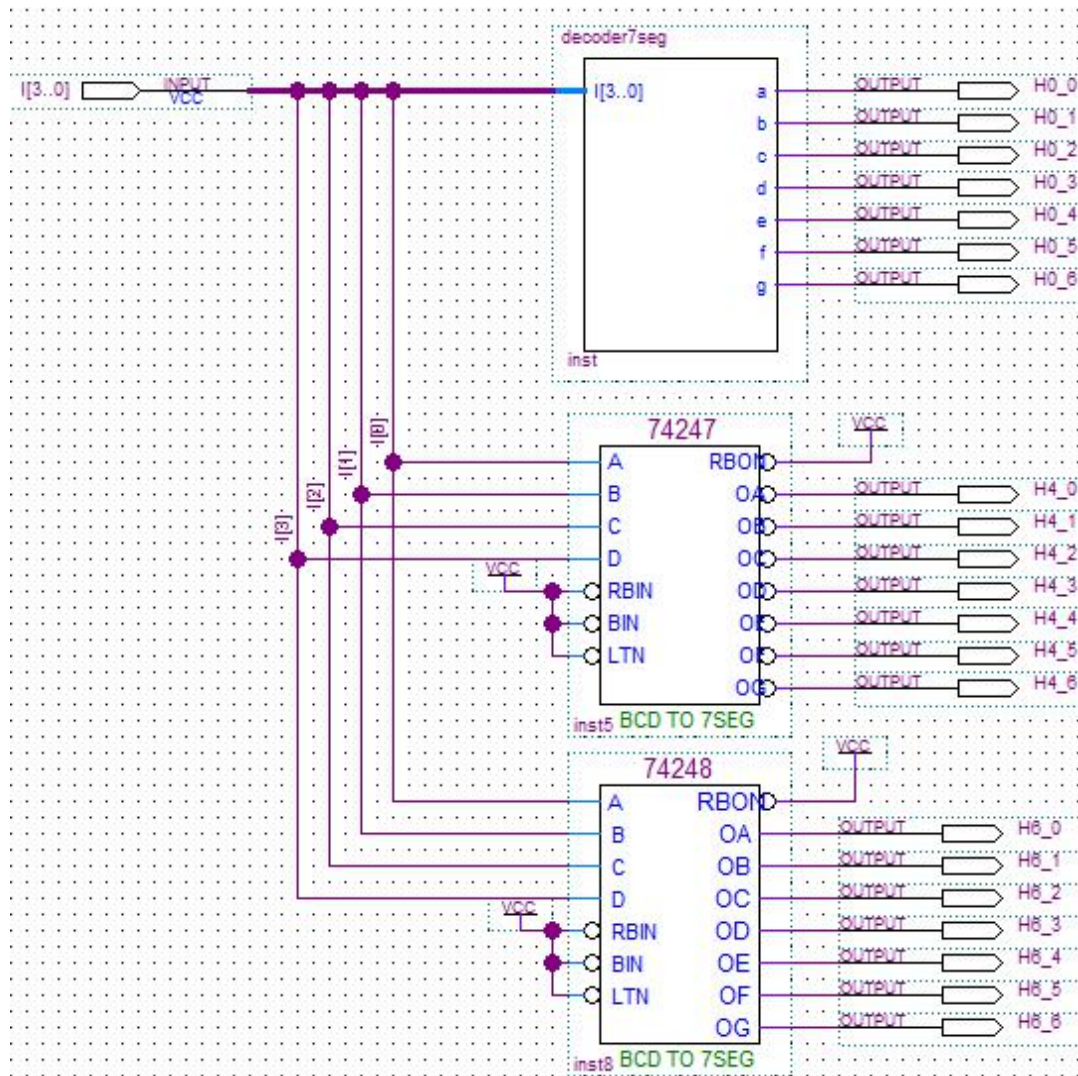


9. Lúc này file thiết kế topdesign.bdf sẽ như sau.

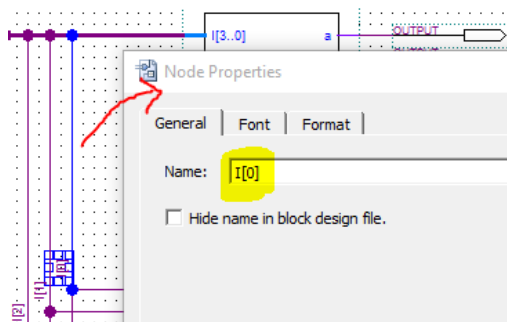


10. Hoàn chỉnh thiết kế của topdesign

- Thêm các pin vào/ra. Chú ý cách đặt tên đa tín hiệu đầu vào XYZ ở Quartus là XYZ[3..0]
- Nối dây. Chú ý cách đặt trích một tín hiệu đơn ra khỏi đa tín hiệu XYZ ở Quartus là XYZ[2]



Mình họa cách trích một đơn tín hiệu từ đường bus đa tín hiệu bằng cách đặt tên với chỉ số trong ngoặc []



Giai đoạn 3: Thực hiện gán chân pin vào SW7..0 và đèn led Hex0, Hex4, Hex6

Tương tự như các bài thực hành trước.

Bài 4 : Đồng hồ đếm thời gian và đếm số lần bấm nút

Mô tả: Sinh viên thiết kế mạch đếm thời gian tăng dần (time-up counter) và hiển thị ra đèn led 7 đoạn theo định dạng: mm:ss:d với

- mm là 2 số bcd, hiển thị số phút, có giá trị từ 00 → 59
- ss là 2 số bcd, hiển thị số giây, có giá trị từ 00 → 59
- d là 1 số bcd, hiển thị số 1/10 giây, có giá trị từ 0 → 9

IC có nút tạm dừng, để mỗi khi người dùng bấm và giữ nút này thì mạch thời gian ngừng đếm.

Đồng thời IC được thiết kế có tính năng phụ là đếm số lần người dùng bấm một nút nào đó và hiển thị số lượng bấm ra đèn led 7 đoạn với phạm vi đếm từ 0→F.

Ý nghĩa:

- Sinh viên nắm được nguyên tắc chia tần số
- Tính toán chia tần hợp lý để chuyển đổi từ MHz sang đơn vị giây, phút.
- Tính toán chia tần với sai số hợp lý để chuyển đổi từ đơn vị MHz sang đơn vị 1/10 giây.

Yêu cầu công việc:

Hãy lập trình điều khiển các

- Nút bấm Key 2 đóng vai trò nút Pause mạch đếm thời gian
- Nút bấm Key 1 đóng vai trò nút bấm để đếm tăng mỗi khi người dùng nhấn.
- Nút bấm Key 0 đóng vai trò nút Reset giá trị bộ đếm.
- Đèn led 7 đoạn HEX7, Hex6 hiển thị số phút đếm được
- Đèn led 7 đoạn HEX5, Hex4 hiển thị số giây đếm được
- Đèn led 7 đoạn HEX3 hiển thị số 1/10 giây đếm được
- Đèn led 7 đoạn HEX0 hiển thị số lần người dùng bấm nút với giá trị từ 0→F

để sao cho:

- Khi Key 0 (reset) được bấm, tất cả đèn led đều hiển thị số 0
- Khi key 1 được bấm, đèn Hex0 tăng dần và quay vòng giá trị từ 0 → F
- Khi không có phím nào được bấm, giá trị thời gian cứ tăng dần trên 5 đèn led HEX7, HEX6, HEX5, HEX4, HEX3 theo dạng mm:ss:d
- Khi Key 2 được bấm và giữ, giá trị thời gian trên 5 đèn led sẽ dừng lại. Khi nhả Key 2, bộ đếm tiếp tục đếm từ giá trị hiện tại.

Kiến thức cần nắm vững:

- Các kỹ thuật để chia tần số
- Chấp nhận sai số khi chia tần số ở dạng tín hiệu số khi tần số xung nhịp không chia hết với số chia.

Các bước tiến hành:

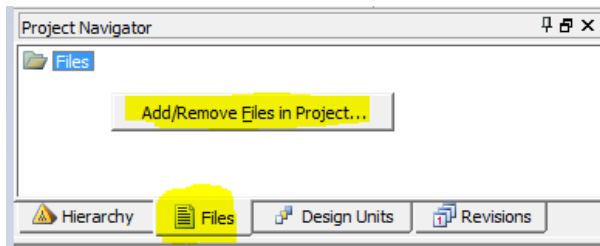
Giai đoạn 1: Sử dụng Active-HDL thiết kế các module cơ bản

1. Tạo một dự án mới trên Active-HDL
2. Sử dụng VHDL code để tạo ra các module nhỏ sau
 - a. Bộ đếm tràn 50 lần, để chia tần số 50 lần
 - b. Bộ đếm tràn 1Mi lần (2^{20} lần): để chia tần số 1Mi lần
 - c. Bộ đếm 4 bit, để đếm số lần người dùng bấm nút
 - d. Bộ đếm 0 → 5, để đếm hàng chục của phút và giây
 - e. Bộ đếm 0 → 9, để đếm hàng đơn vị của phút và giây
 - f. Bộ đếm 0 → 127, để chia nhỏ 1 giây thành 10 phần, 10 deci second

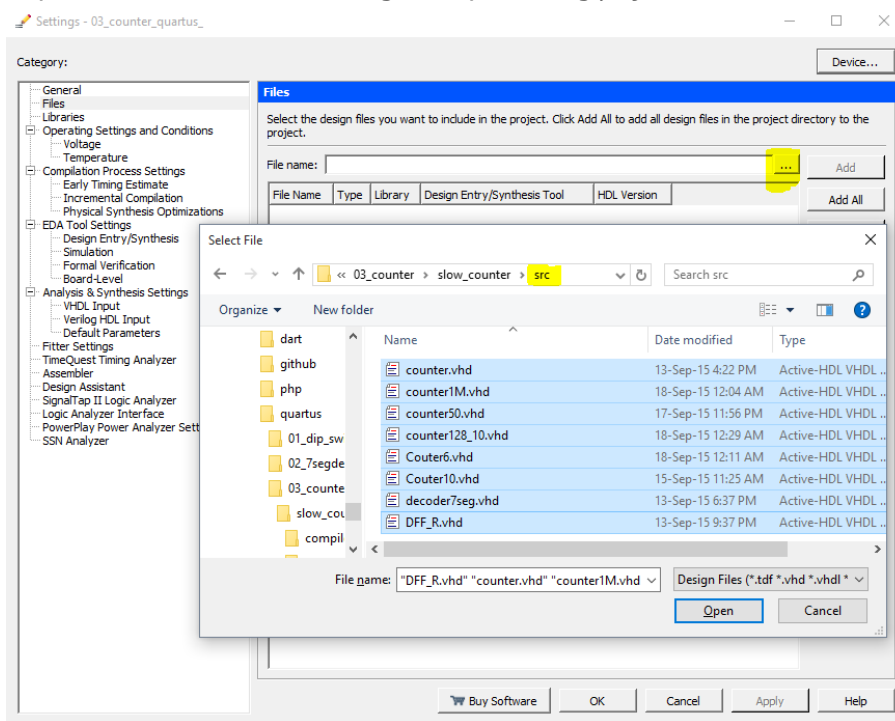
- g. Bộ giải mã led 7-đoạn
3. Sử dụng Block Diagram để tạo ra thiết kế mức Top Level. Xem ảnh bên dưới
4. Tiến hành biên dịch toàn bộ thiết kế mức Top Level thành VHDL, bằng cách bấm phím F11. File VHDL được sinh ra sẽ nằm trong thư mục *compile* của Project.

Giai đoạn 2: Tạo dự án mới trên Quartus và link tới các file VHDL tạo bởi Active-HDL

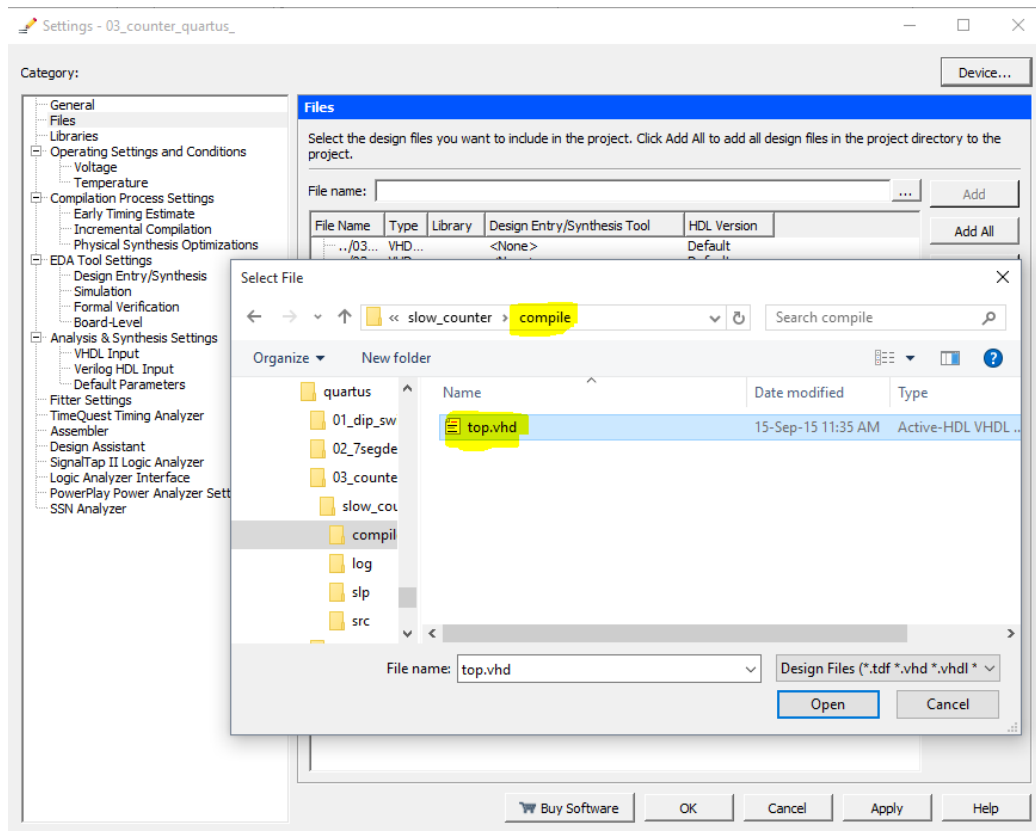
5. Tạo Project Quartus mới. Ví dụ: lấy tên là *03_counter_quartus*
6. Trong cửa sổ Project Navigator, chọn tab Files, bấm chuột phải, chọn Add/Remove Files in Projects...



7. Khi cửa sổ Setting xuất hiện, bấm vào nút để bổ sung file.
Chọn các file VHDL thuần nằm trong thư mục *src* trong project của Active-HDL

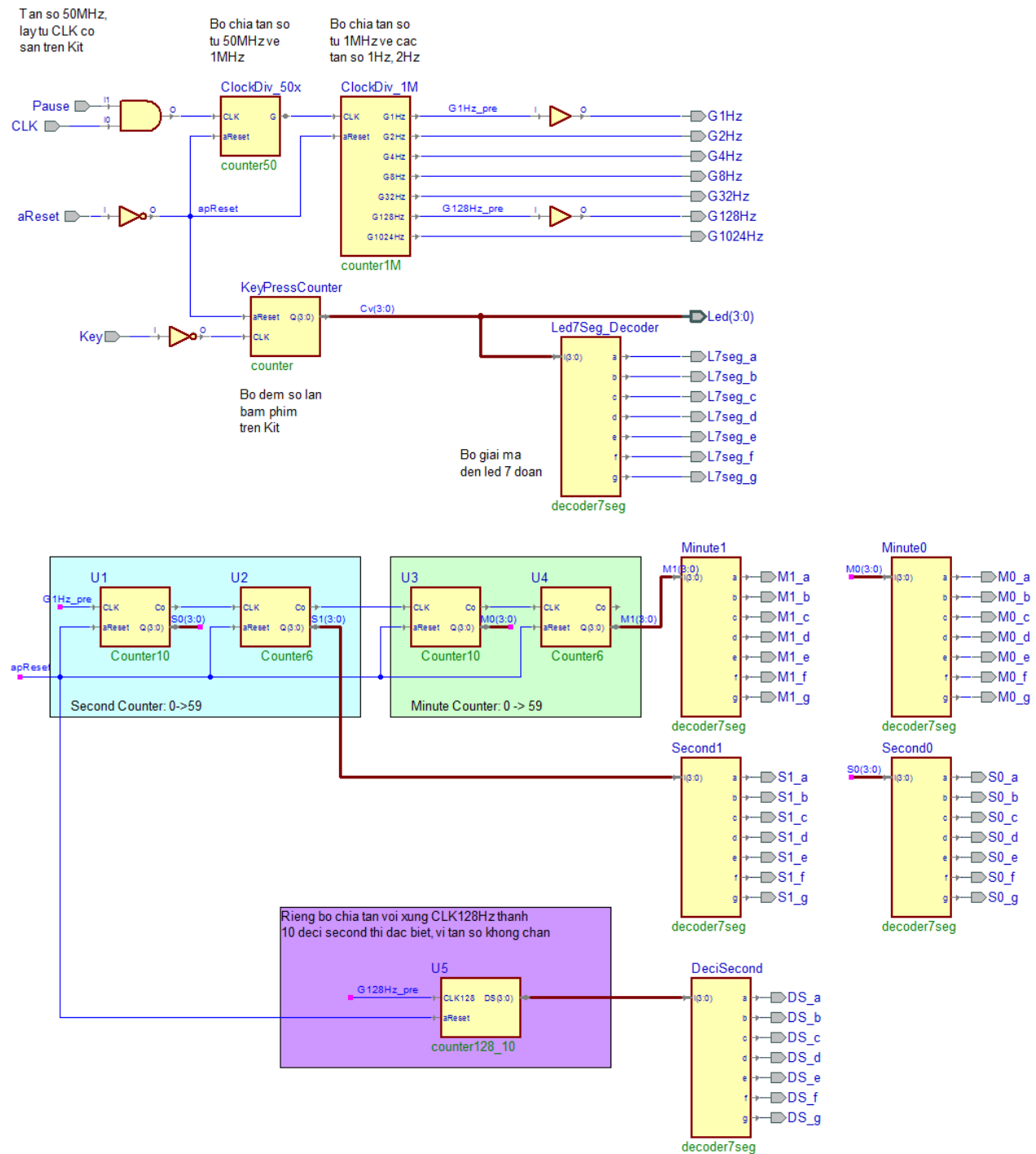


8. Vẫn trong cửa sổ Setting, bấm vào nút để bổ sung file.
Chọn các file VHDL được sinh ra từ Block Diagram, nằm trong thư mục *compile* trong project của Active-HDL



9. Quá trình tạo dự án mới trên Quartus và link tới các file mã nguồn trong Active-HDL đã xong. Mọi sự thay đổi ở các file VHDL trong dự án của Active-HDL sẽ tự động thay đổi trong dự án của Quartus và ngược lại.
10. Tiến hành biên dịch và nạp FPGA giống như các bài thực hành trước.

Thiết kế mức Top Level



Bộ đếm tràn 50 lần

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
```

```

--- @brief Bo dem tran 50, de chia tan so 50MHz thanh 1MHz
--- @details Nguyen tac dem: 39, 40, 41.. 63, 39
---      Chi dem cho 25 boi vi sau khi cong 25 lan thi Carry Flag se thay doi tu 0 -> 1
---      Sau do, bo cong tai dem tiep 25 lan nua thi Carry Flag se thay doi tu 1 -> 0
--      Do do, tinh ca 2 lan la 50 chu ki CLK dau vao moi tao ra du 1 chu ki dau ra
entity counter50 is
    port(
        CLK : in STD_LOGIC;      --- Tin hieu dong ho dau vao. Kit DE2-150i, CLK=50MHz co san tren
mainboard
        aReset: in STD_LOGIC;    --- Tin hieu reset a/p.
        G : buffer STD_LOGIC     --- Tin hieu sau khi tran bo cong,
    );
end counter50;

architecture behavior of counter50 is
    signal adder_value: STD_LOGIC_VECTOR(6 downto 0);    --- Thanh ghi chua gia tri hien thoi cua bo dem
    signal load_value: STD_LOGIC_VECTOR(5 downto 0) := "100111"; --- Gia tri nap khi tran = 2**6-25=39
begin
    process (CLK, aReset)
    begin
        if aReset = '1' then
            adder_value <= '0' & load_value;
        elsif rising_edge(CLK) then
            adder_value <= adder_value + 1;
            if (adder_value(5 downto 0) = "111111") then
                adder_value(5 downto 0) <= load_value;
            end if;
        end if;
    end process;

    --- Dua gia tri bit carry, bit tran cua bo dem ra ngoai, thanh tin hieu G 1MHz.
    G <= adder_value(6);
end behavior;

```

Bộ đếm tràn 1 Mi lần

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

--- @brief Bo dem tran 1MHz, de chia tan so 1MHz thanh nhieu tan so nho hon
entity counter1M is
    port(
        CLK      : in STD_LOGIC;      --- Tin hieu dong ho dau vao.
        aReset: in STD_LOGIC;      --- Tin hieu reset a/p.
        G1Hz     : out STD_LOGIC;    --- Tin hieu sau khi tran bo cong, voi tan so 1Hz
        G2Hz     : out STD_LOGIC;    --- Tin hieu sau khi tran bo cong, voi tan so 2Hz
        G4Hz     : out STD_LOGIC;    --- Tin hieu sau khi tran bo cong, voi tan so 5Hz
        G8Hz     : out STD_LOGIC;    --- Tin hieu sau khi tran bo cong, voi tan so 10Hz
        G32Hz    : out STD_LOGIC;    --- Tin hieu sau khi tran bo cong, voi tan so 32Hz
        G128Hz   : out STD_LOGIC;    --- Tin hieu sau khi tran bo cong, voi tan so 128Hz
        G1024Hz  : out STD_LOGIC;    --- Tin hieu sau khi tran bo cong, voi tan so 1024Hz
    );
end counter1M;

architecture behavior of counter1M is
    signal adder_value: STD_LOGIC_VECTOR(19 downto 0);    --- Thanh ghi chua gia tri hien thoi cua bo dem
begin
    --- Thuc hien bo dem 19 bit
    process (CLK, aReset)
    begin
        if aReset = '1' then
            adder_value <= (others => '0');
        elsif rising_edge(CLK) then
            adder_value <= adder_value + 1;
        end if;
    end process;
    --- Dua gia tri bit carry, bit tran cua bo dem ra ngoai, thanh tin hieu chia tan so
    G1Hz <= adder_value(19);
end behavior;

```

```

G2Hz    <= adder_value(18);
G4Hz    <= adder_value(17);
G8Hz    <= adder_value(16);
G32Hz   <= adder_value(14);
G128Hz  <= adder_value(12);
G1024Hz <= adder_value(9);
        -- G500KHz <= <= adder_value(0);
end behavior;

```

Bộ đếm 4 bit

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity counter is
    generic ( BIT_NUM : integer := 4 );
    port(
        aReset : in STD_LOGIC;
        CLK : in STD_LOGIC;
        Q : out STD_LOGIC_VECTOR(BIT_NUM-1 downto 0)
    );
end counter;

architecture behavior of counter is
    signal adder_value: STD_LOGIC_VECTOR(BIT_NUM-1 downto 0);
begin
    process (CLK, aReset)
    begin
        if aReset = '1' then
            adder_value <= (others => '0');
        elsif rising_edge(CLK) then
            adder_value <= adder_value + 1;
        end if;
    end process;
    Q <= adder_value;
end behavior;

```

Bộ đếm 0 → 5

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity Counter6 is
    port(
        CLK : in STD_LOGIC;
        aReset : in STD_LOGIC;
        Co : out STD_LOGIC;
        Q : buffer STD_LOGIC_VECTOR(3 downto 0)
    );
end Counter6;

architecture Counter6 of Counter6 is
begin
    process (CLK, aReset)
    begin
        if aReset = '1' then
            Q <= "0000";
            Co <= '0';
        elsif rising_edge(CLK) then
            Q <= Q + 1;
            if (Q = "0101") then
                Q <= "0000";
                Co <= '1';
            end if;
            if (Q = "011") then
                Co <= '0';
            end if;
        end if;
    end process;
end Counter6;

```

-- Tao suon len khi bo dem nhan gia tri 5 -> 0

-- Tao suon xuong khi bo dem nhan gia tri 3 -> 4

```

        end if;
    end process;
end Counter6;

```

Bộ đếm 0 → 9

```
-- Sinh viên tự thực hiện
```

Bộ đếm 0 → 127

```

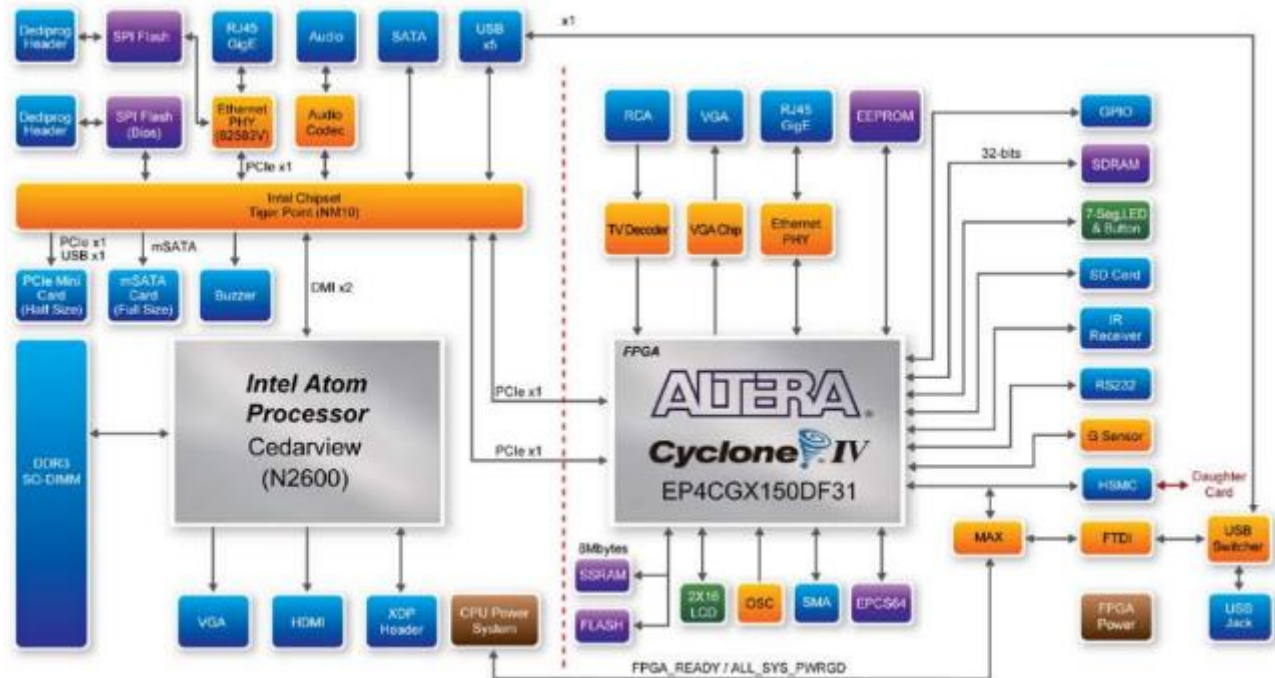
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

-- Boi vi sau khi chia tan so, xung dong ho khong chan 10 vi vay
-- phai su dung giai thuat gan dung de chuyen tu 128Hz sang don vi centi-second
entity counter128_10 is
    port(
        CLK128 : in STD_LOGIC;    --- Tin hieu dong ho dau vao. Kit DE2-150i, CLK=50MHz,
                                   --- sau khi chia tan, con lai 128Hz
        aReset: in STD_LOGIC;    --- Tin hieu reset a/p.
        DS     : buffer STD_LOGIC_VECTOR(3 downto 0) -- Gia tri Deci-second don vi
    );
end counter128_10;

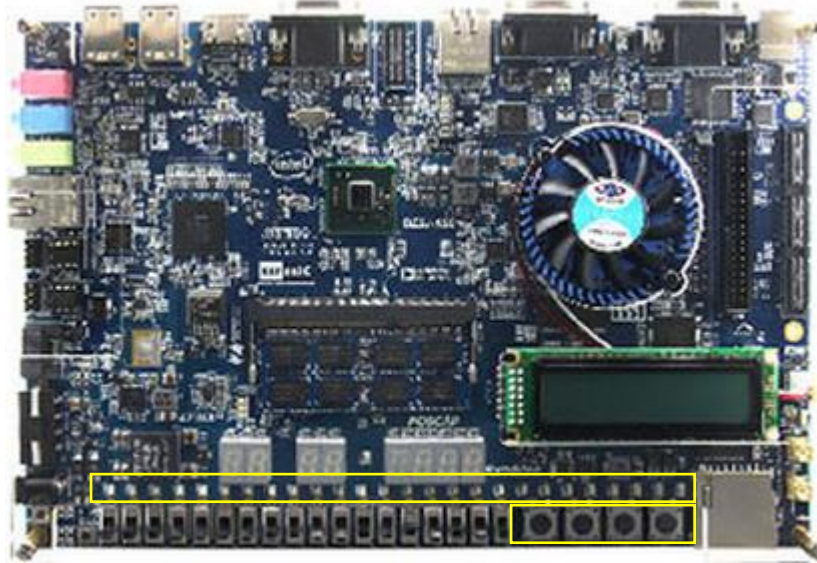
--}} End of automatically maintained section

architecture behavior of counter128_10 is
    signal adder_value: STD_LOGIC_VECTOR(6 downto 0);    --- Thanh ghi chua gia tri hien thoi cua bo dem
    7 bit, tu 0-> 127
begin
    process (CLK128, aReset)
        variable deci: integer;
    begin
        if aReset = '1' then
            -- boi vi moi khi bam reset thi chi con 1 nua chu ki la sang giay tiep theo roi
            adder_value <= CONV_STD_LOGIC_VECTOR(65,7);
            DS <= (others => '0');
        elsif rising_edge(CLK128) then
            adder_value <= adder_value + 1;
            --adder_value= 0 --> 12, 13 -> 25, 26 -> 38, 39 -> 51, 52 -> 64, 65 -> 76, 77-> 89, 90
            --tmp
            = 0 --> 0, 1 -> 1, 2 -> 2, 3 -> 3, 4 -> 4, 5 --> 5, 6 -> 6, 7
            if adder_value = CONV_STD_LOGIC_VECTOR(0,7) then
                DS <= CONV_STD_LOGIC_VECTOR(0,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(13,7) then
                DS <= CONV_STD_LOGIC_VECTOR(1,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(26,7) then
                DS <= CONV_STD_LOGIC_VECTOR(2,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(39,7) then
                DS <= CONV_STD_LOGIC_VECTOR(3,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(52,7) then
                DS <= CONV_STD_LOGIC_VECTOR(4,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(65,7) then
                DS <= CONV_STD_LOGIC_VECTOR(5,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(77,7) then
                DS <= CONV_STD_LOGIC_VECTOR(6,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(90,7) then
                DS <= CONV_STD_LOGIC_VECTOR(7,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(103,7) then
                DS <= CONV_STD_LOGIC_VECTOR(8,4);
            elsif adder_value = CONV_STD_LOGIC_VECTOR(116,7) then
                DS <= CONV_STD_LOGIC_VECTOR(9,4);
            end if;
        end if;
    end process;
end behavior;

```



Bài tập lớn 1. IC tạo hiệu ứng ánh sáng cho led dải



Mô tả: Tạo một IC điều khiển để tạo ra 3 hiệu ứng ánh sáng khác nhau cho 18 led đỏ và 8 led xanh như sau:

1. Chớp sáng: tắt cả các đèn đều sáng, tắt liên tục
 ○○○○○○○○ → ○○○○○○○○ → ○○○○○○○○
2. Con rắn: nhóm 5 đèn sáng chạy dần từ trái sang phải, từ phải sang trái
 ○○○○○○ ○○○○○○ → ○○○○○○ ○○○○○○ → ○○○○○○ ○○○○○○
3. Va chạm: 2 đèn led sáng chạy từ 2 đầu và gặp nhau ở giữa
 ○○○○○○○○ → ○○○○○○○○ → ○○○○○○○○ → ○○○○○○○○ → ○○○○○○○○ →
 ○○○○○○○○ → ○○○○○○○○ → ○○○○○○○○ → ○○○○○○○○ → ○○○○○○○○

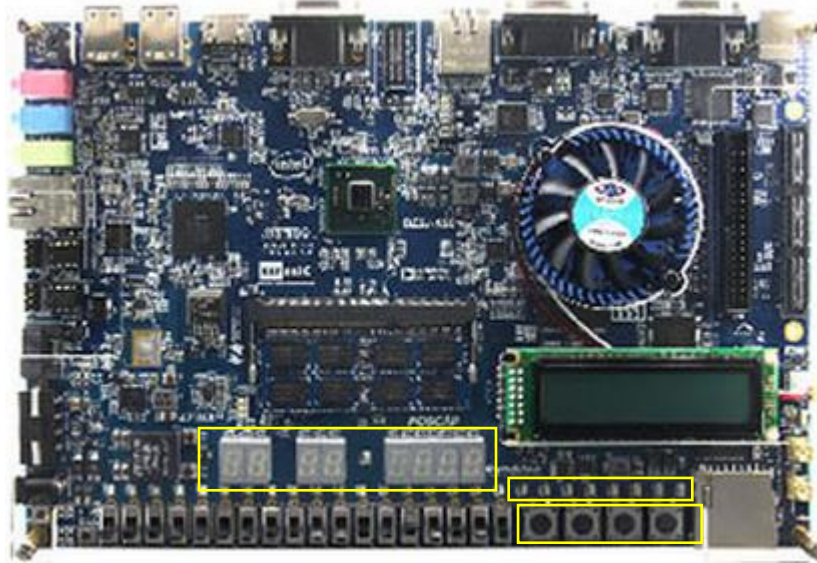
Đồng thời:

- Các hiệu ứng này chuyển động không ngừng, lặp đi lặp lại.
- Thời gian các đèn led chuyển từ hoạt cảnh này sang hoạt cảnh khác sẽ do nhóm tự quyết định, chỉ cần mắt thường nhìn được.

IC cũng đồng thời xử lý 2 nút bấm Key1 và Key 2 sao cho:

1. Phím Key0 để reset mạch dãy.
2. Khi bấm Key1, sẽ chuyển hiệu ứng này sang hiệu ứng khác. (hiệu ứng 1 → 2 → 3 → 1 → 2...)
3. Khi bấm và giữ Key2, hiệu ứng sẽ tạm dừng. Nhả Key2, hiệu ứng lại tiếp tục

Bài tập lớn 2. Đồng hồ đếm ngược



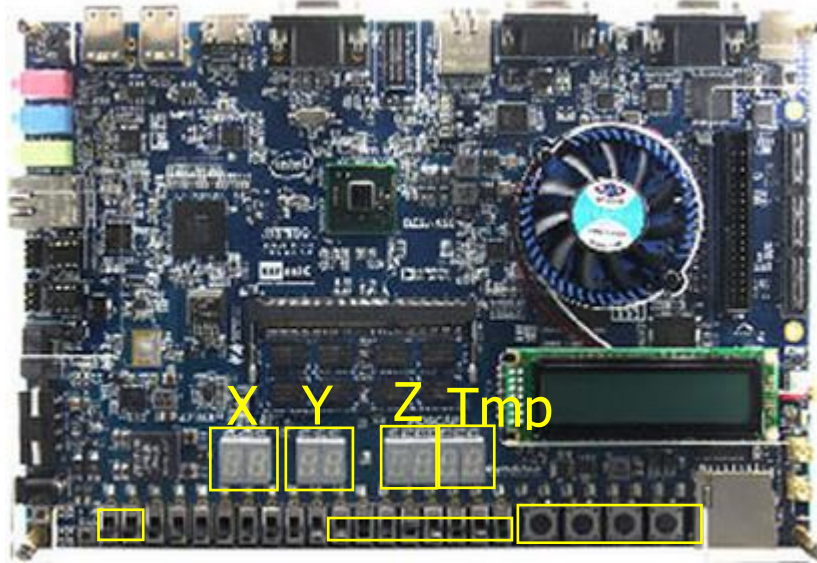
Mô tả: Tạo một IC điều khiển đồng hồ thời gian đếm xuôi và ngược, với độ chính xác 1/10 giây.

1. Hiển thị thời gian: giây (từ 0→59) và 1/10 giây lên các đèn led 7-đoạn
2. IC cho phép đếm thời gian xuôi hoặc ngược
3. Khi đếm ngược, giá trị thời gian ban đầu sẽ được thiết lập. Ví dụ 17 giây.
4. Khi đếm ngược về tới 0, quá trình đếm phải kết thúc luôn, không được tiếp tục đếm xuôi/ngược nữa.

Đồng thời

5. Phím Key0 để reset mạch dãy.
6. Người dùng bấm phím Key 1 để tăng số giây thiết lập lúc ban đầu, tăng từ 0→...→59→0→... Giá trị lúc ban đầu đó cũng được hiển thị lên đèn led 7-đoạn ngay lập tức.
7. Người dùng bấm phím Key2 để bắt đầu quá trình đếm ngược, và bấm Key 2 lần nữa để tạm dừng quá trình đếm ngược. Bấm key 2 lần thứ 3 thì quá trình đếm ngược lại tiếp tục....
8. Người dùng bấm và giữ phím Key3 thì quá trình đếm ngược chuyển thành đếm xuôi.

Bài tập lớn 3. Bộ ALU



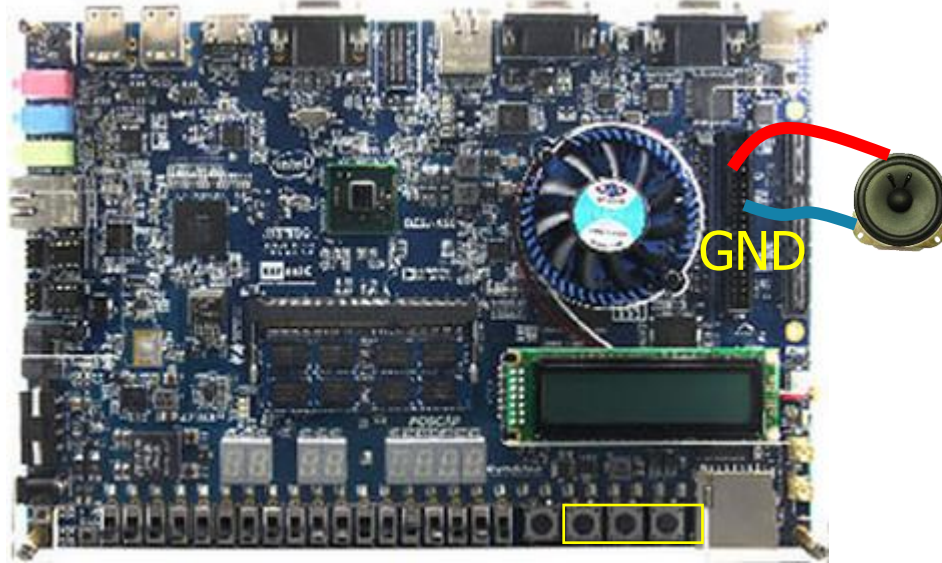
Mô tả: Tạo bộ ALU với 4 phép toán

1. ALU thực hiện được phép toán cộng/trừ/and/or tương ứng với các mã phép toán là 0,1,2,3.
2. ALU thực hiện phép toán trên 2 số 8 bit không dấu và được kết quả là số 8 bit không dấu.
 $X + Y = Z$ $X - Y = Z$ $X \text{ and } Y = Z$ $X \text{ or } Y = Z$
3. Giá trị của các toán hạng X, Y, Z được hiển thị lên đèn led 7-đoạn ở hệ 16.

Đầu vào như sau

4. Sử dụng 8 công tắc DIP switch để thiết lập giá trị ban đầu, gọi là Tmp. Mỗi thay đổi switch, giá trị tương ứng (ở hệ 16) sẽ được hiển thị lên led 7-đoạn ở vị trí như trong ảnh.
5. Giá trị Tmp đó sẽ được gán cho toán hạng X, nếu như người dùng bấm nút Key 1.
6. Giá trị Tmp đó sẽ được gán cho toán hạng Y, nếu như người dùng bấm nút Key 2.
7. Người dùng gạt 2 switch số 16 và 17 để chọn mã phép toán
8. Nếu người dùng bấm nút Key 3, giá trị của kết quả Z hiện thời sẽ được gán cho toán hạng Y.

Bài tập lớn 4. Hộp nhạc



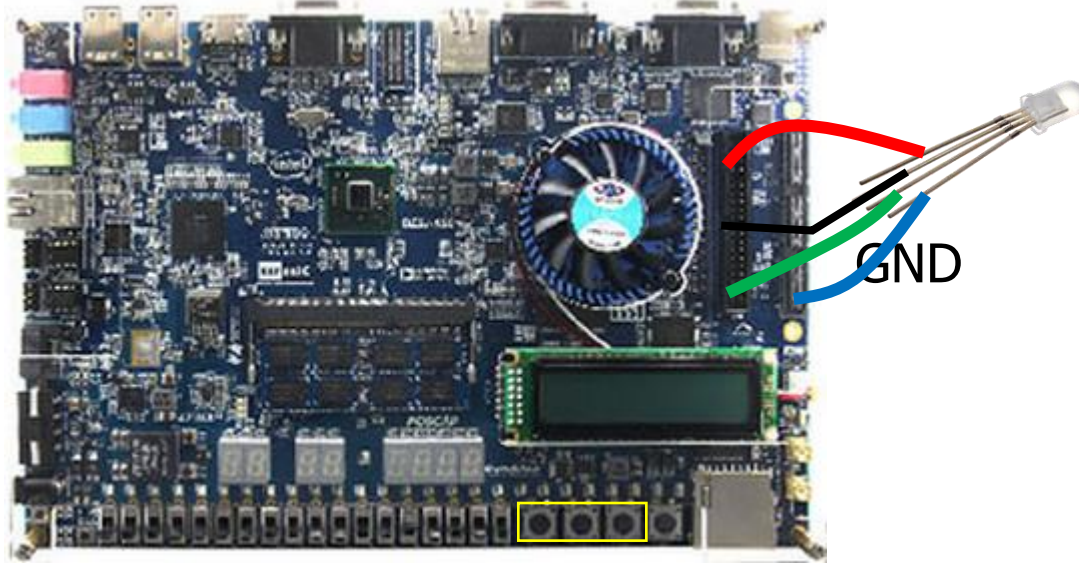
Mô tả: Hãy thiết kế một IC điều khiển hộp nhạc lưu trữ 3 bản nhạc khác nhau. Nhạc sẽ được phát ra khi người dùng chọn bài hát.

1. Lưu trữ 3 bản nhạc cố định trong IC.
2. Mỗi bản nhạc chỉ cần có sự khác biệt với nhau là được, với các nốt nhạc khác nhau. Ví dụ, bài "Chắc ai đó sẽ học lại".
3. Sau khi chọn bài, âm thanh sẽ được phát ra ở loa ngoài.
4. Loa có 2 pin, nối trực tiếp với Kit DE2i-150 như hình vẽ. Lưu ý rằng
 - a. 1 chân của loa ngoài luôn nối với chân GND,
 - b. 1 chân còn lại của loa ngoài nối với chân tín hiệu cần cấu hình của FPGA.

Đầu vào như sau

5. Người dùng bấm Key0 để Reset mạch dãy.
 6. Người dùng bấm Key 1 để chuyển sang bài hát tiếp theo theo thứ tự tăng dần
 7. Người dùng bấm Key 2 để Start/Pause bài hát.
 8. Bảng tần số của các note nhạc tra cứu ở đây: <http://www.phy.mtu.edu/~suits/notefreqs.html>
- Giới hạn của bài tập: tần số của nốt nhạc chỉ cần gần đúng, và hỗ trợ 3 nốt nhạc là được. Nhiều hơn càng tốt.

Bài tập lớn 5. Hiệu ứng màu RGB



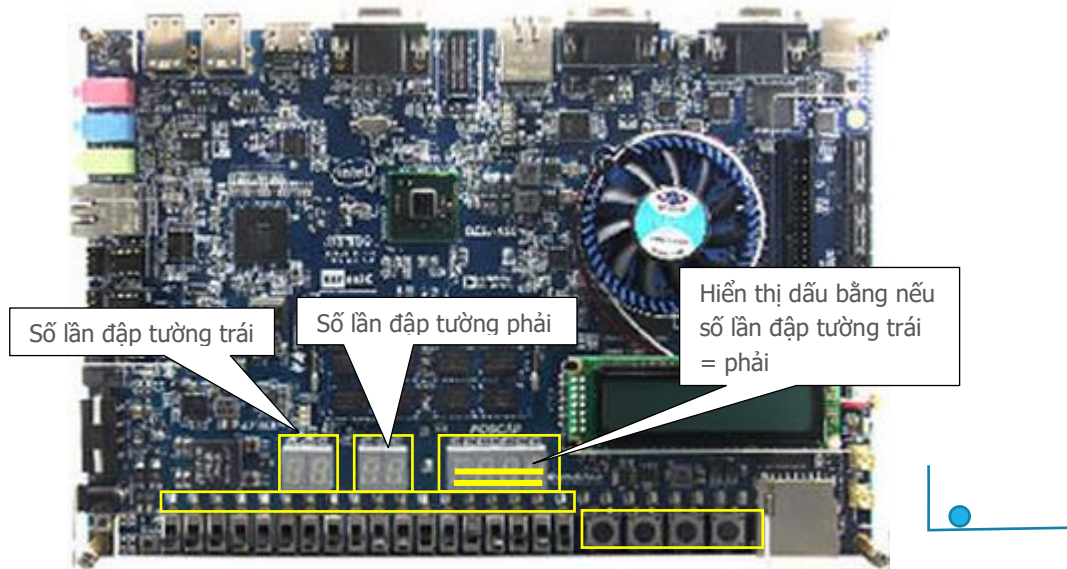
Mô tả: hãy tạo ra IC điều khiển màu của led RGB với nhiều màu sắc khác nhau.

1. Hãy tạo ra 8 màu sắc khác nhau với đèn led RGB
2. Có thể điều khiển được độ sáng của từng màu R, G, B bằng cách dùng độ rộng xung.

Với các đầu vào là:

3. Khi người dùng bấm Key 1, đèn led rơi vào trạng thái Test. Đèn sẽ sáng tất cả các màu, rồi tắt, nhấp nháy liên tục.
4. Khi người dùng bấm phím Key2, các hiệu ứng màu lần lượt chuyển

Bài tập lớn 6. Trò chơi quả bóng đập tường



Mô tả: một quả bóng được điều khiển bởi người chơi, sẽ va đập vào 2 bên cạnh tường. Mỗi khi va đập sang bên trái, hoặc phải thì số lần va chạm đó sẽ được hiển thị lên đèn led 7-seg.

1. Toàn bộ dải 18 đèn led sẽ tắt, chỉ trừ 1 đèn sáng, tương trưng cho một quả bóng
2. Người dùng bấm Key0 để Reset mạch dãy.
3. Khi người dùng bấm Key 1 (RIGHT), quả bóng dịch sang phải 1 vị trí
 ○○○○○○ ○○○○ → ○○○○○○○○ ○○○○
4. Nếu bóng đang ở vị trí led 1, khi người dùng bấm Key 0 (RIGHT), bóng sẽ chuyển ra led 0. Người dùng bấm tiếp Key 0 (RIGHT), bóng vẫn ở vị trí led 0, nhưng IC ghi nhận đã có thêm 1 lần bóng đập tường phải.
 Tuy nhiên, nếu người dùng tiếp tục bấm Key , IC sẽ không ghi nhận đập tường nữa
 → ○○○○○○○○ ○○ → ○○○○○○○○○○ ○ → ○○○○○○○○○○ ○ (ghi nhận) → ○○○○○○○○○○ ○ (không ghi nhận)
5. Tương tự như vậy với nút Key 3 (LEFT)
6. Khi số lần đập bóng trái và phải bằng nhau, dấu = sẽ hiển thị ở các đèn led 7-đoạn còn lại.
7. Mỗi khi người dùng bấm Key 2 (BRIGHTNESS), độ sáng của quả bóng sẽ thay đổi luân phiên
 - a. Sáng chói
 - b. Sáng 50%