BÀI THỰC HÀNH SỐ 3

I. Mục tiêu

- Trong bài thực hành này, sinh viên sẽ dùng procedural assignment để thiết kế các mạch đếm (Counter) và mạch định thời (Timer).
- Thực hành sử dụng **LPM** (Library of Parameterized Modules) của Altera

  http://quartushelp.altera.com/14.1/master.htm#mergedProjects/reference/glossary/def_lpm.htm

  http://quartushelp.altera.com/14.1/master.htm#mergedProjects/hdl/mega/mega_list_mega_lpm.htm

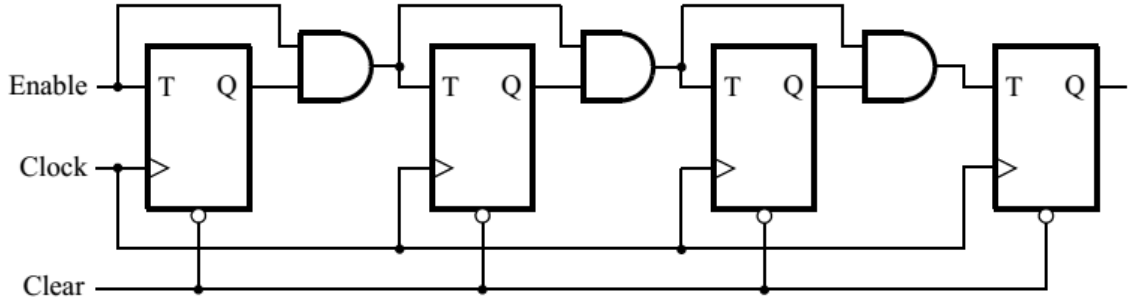
II. Chuẩn bị thực hành

- Sinh viên phải chuẩn bị code Verilog cho tất cả các câu trong phần nội dung thực hành và nộp cho GVHD vào đầu buổi học.
- Sinh viên nào không có bài chuẩn bị được xem là vắng buổi học hôm đó.
- Bài chuẩn bị được tính vào điểm bài báo cáo của Lab.

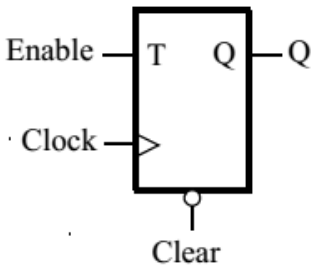
III. Nội dung thực hành

Câu 1.

1.1. Thiết kế bộ đếm 4-bit như hình dưới:



Bảng sự thật của T-FF



Clear	Enable	Clock	Q+
0	x	x	0
1	0	↓	Q
1	1	↓	Q’

Hướng dẫn:

- Tín hiệu Clear trong T-FF trên là Clear bất đồng bộ và tích cực mức thấp

Verilog code:

```
always @(posedge Clock or negedge Clear)
        if (Clear == 1'b0)
        ….
```

- Clock = ~KEY[0]; Clear = SW[1];
- Enable = SW[0]; LEDR[0] = Enable;
- LEDG [3:0] = Q[3:0]

Yêu cầu: SV thực hiện mô phỏng VectorWaveform và nạp KIT demo cho mạch đếm.

1.2. Thiết kế bộ đếm 4-bit như trong câu 1.1, nhưng mô tả ở mức Behavior

Q <= Q + 1

Thực hiện gán chân như câu 1.1

Sử dụng RTL Viewer để xem mạch sau khi Synthesis so với mạch trong câu 1.1

So sánh tần số Fmax của mạch trong câu 1.1 và câu 1.2

Câu 2.

2.1. Sửa lại thiết kế bộ đếm 4-bit trong câu 1.2. bằng cách dùng Parameter và Clear đồng bộ.

Verilog code cho mạch có Clear đồng bộ, tích cực mức 0:

```
always @(posedge Clock)
        if (Clear == 1'b0)
```

2.2. Thiết kế bộ đếm 8-bit bằng cách gọi bộ đếm 4-bit trong câu 2.1. và hiệu chỉnh giá trị Parameter.

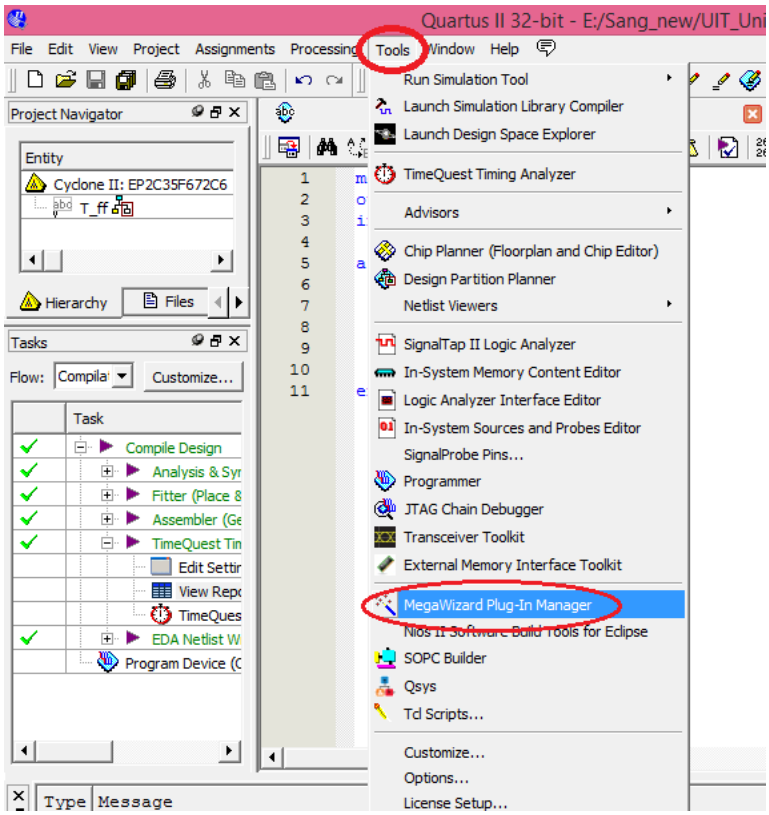
Giá trị ngõ ra bộ đếm Q[7:0] được gán cho LEDG[7:0]

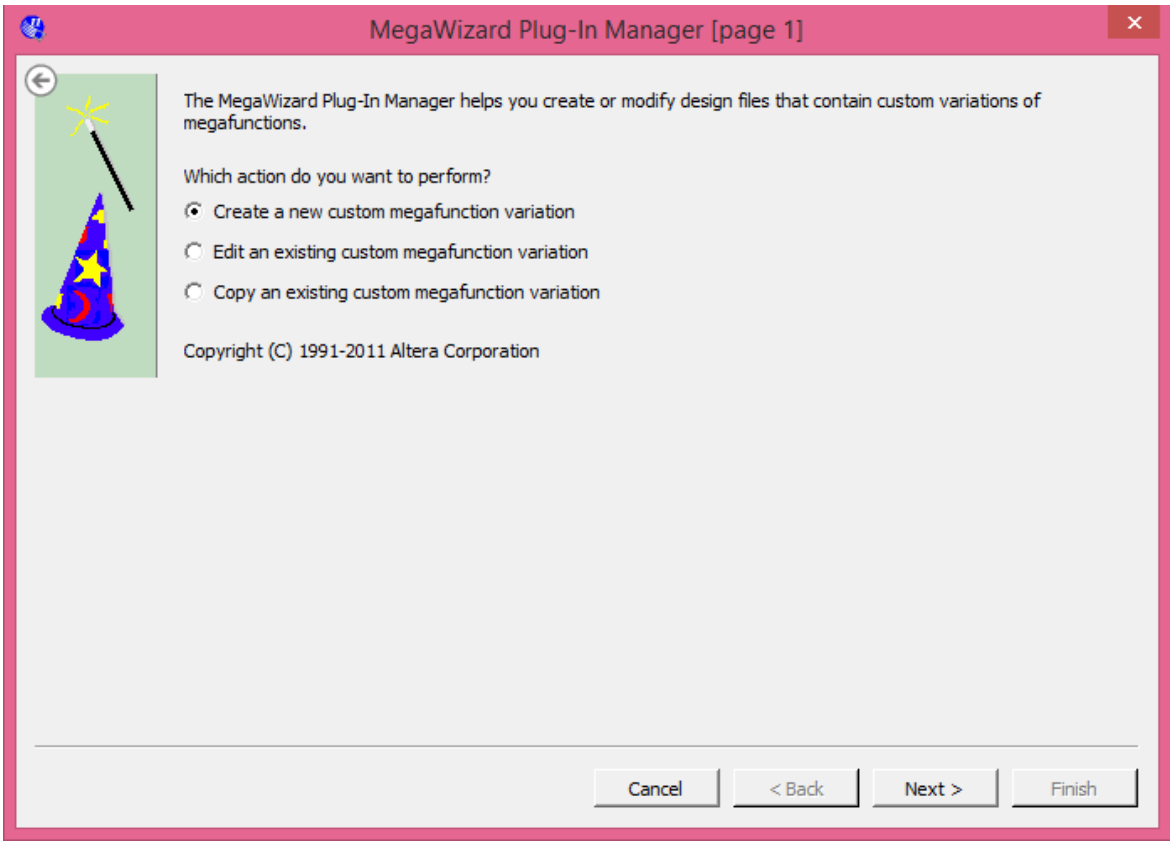
Gán chân cho Clock, Enable, Clear giống câu 1.1

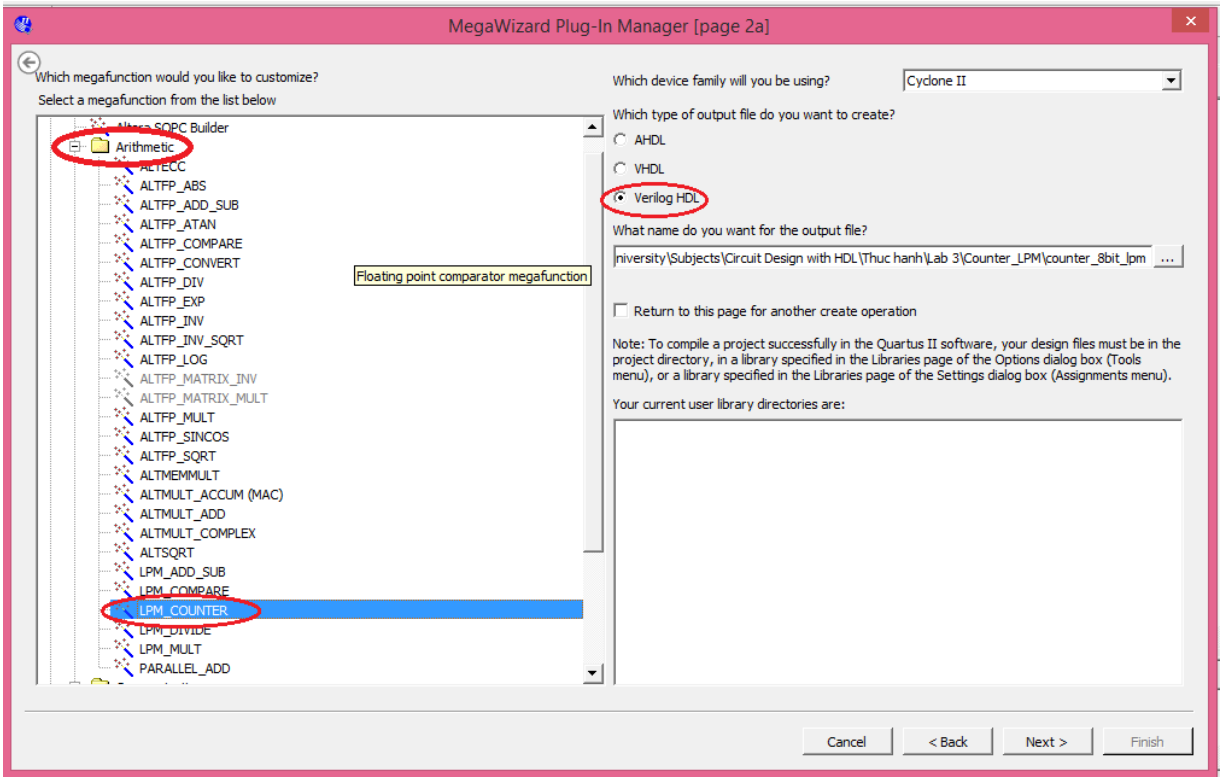
2.3. Thiết kế bộ đếm 8-bit bằng các sử dụng **LPM** (Library of Parameterized Modules) của Altera.

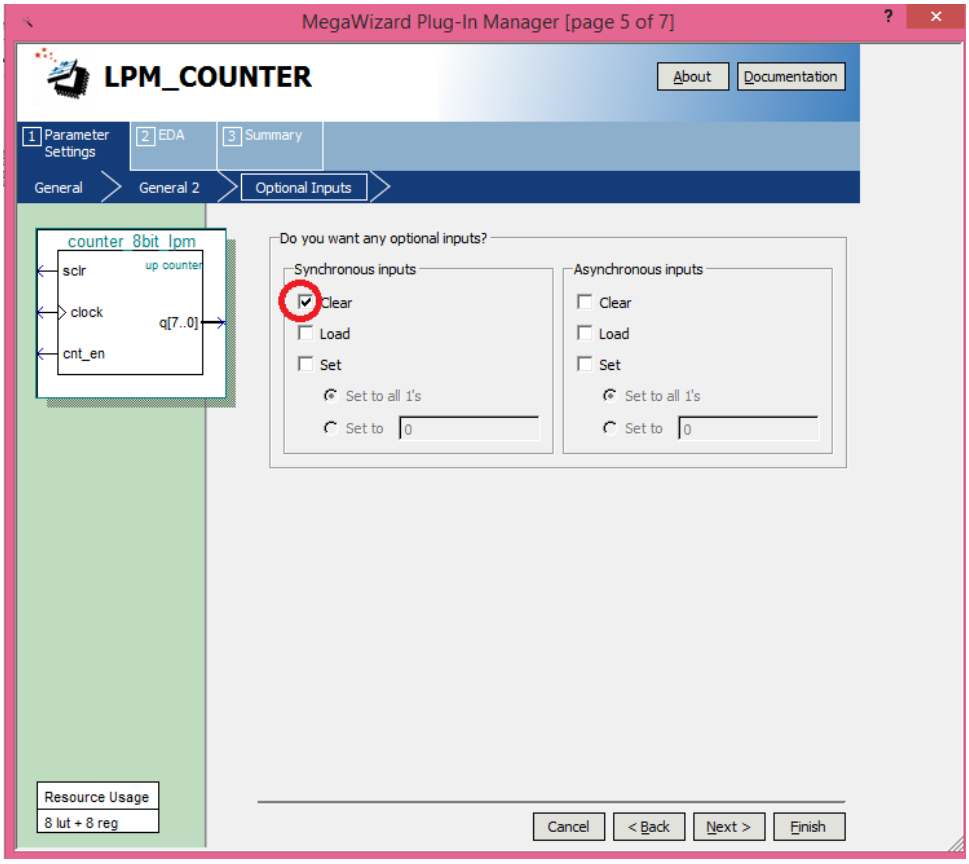
Để sử dụng LPM của Altera:

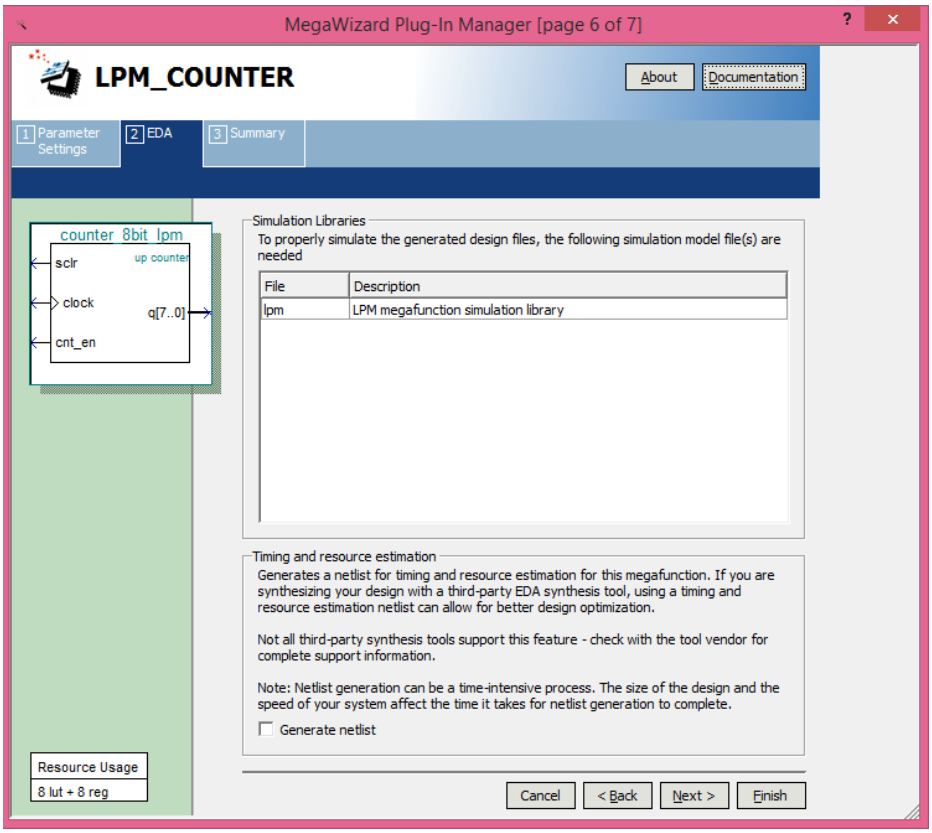
Mở phần mềm Quartus II → Chọn **Tool → MegaWizard Plug-in Manager**

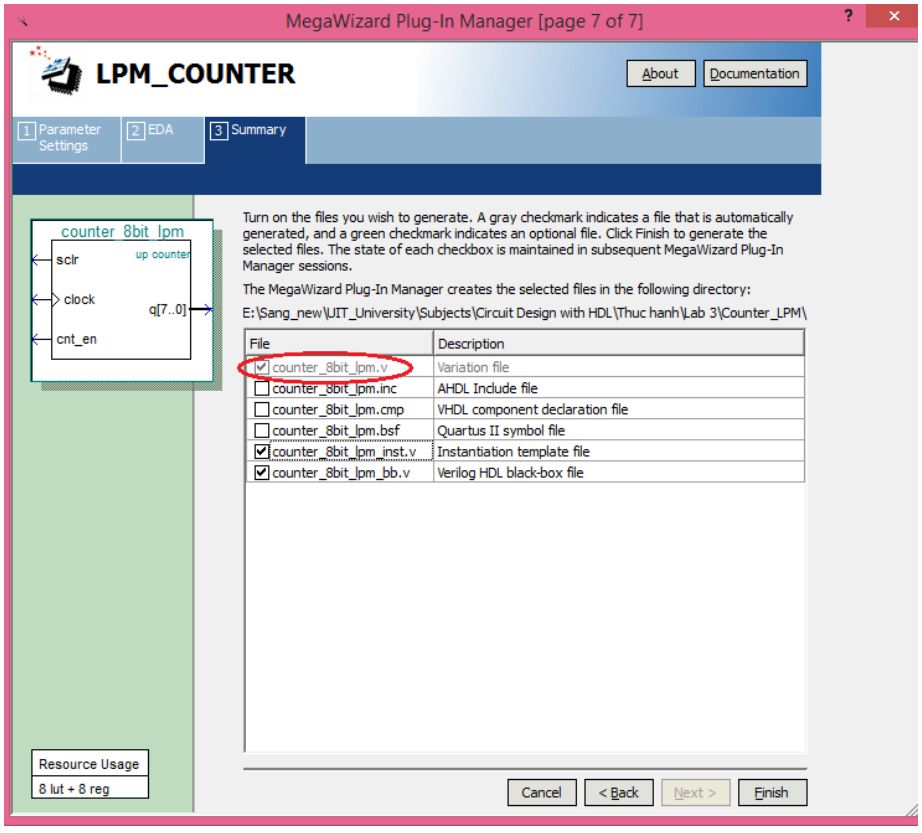

Quartus II 32-bit - E:/Sang_new/UIT_Uni
File Edit View Project Assignments Processing Tools Window Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer
TimeQuest Timing Analyzer
Advisors
Chip Planner (Floorplan and Chip Editor)
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
SignalProbe Pins...
Programmer
JTAG Chain Debugger
Transceiver Toolkit
External Memory Interface Toolkit
MegaWizard Plug-In Manager
SOPC Builder
Qsys
Tcl Scripts...
Customize...
Options...
License Setup...
Project Navigator
Entity
Cyclone II: EP2C35F672C6
T_ff
Hierarchy
Files
Tasks
Flow:
Customize...
Task
Compile Design
Analysis & Syn
Fitter (Place &
Assembler (Ge
TimeQuest Tin
Edit Settir
View Repo
TimeQues
EDA Netlist W
Program Device (C
Type
Message

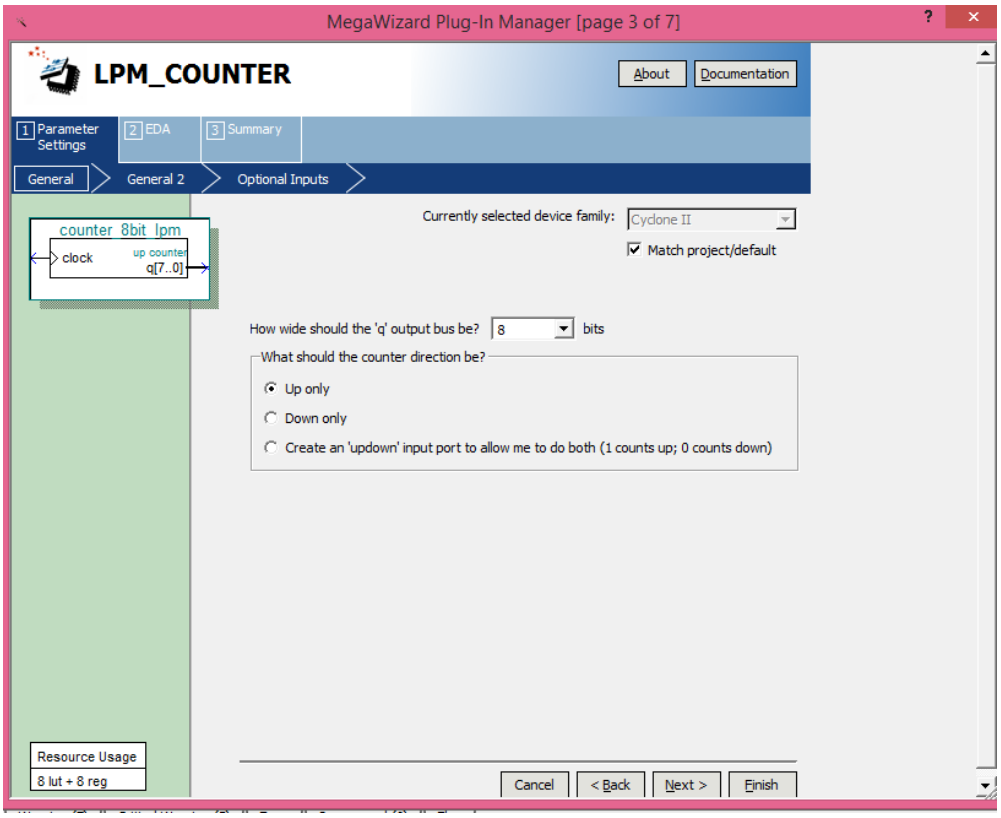

MegaWizard Plug-In Manager [page 1]
The MegaWizard Plug-In Manager helps you create or modify design files that contain custom variations of megafunctions.
Which action do you want to perform?
Create a new custom megafunction variation
Edit an existing custom megafunction variation
Copy an existing custom megafunction variation
Copyright (C) 1991-2011 Altera Corporation
Cancel
< Back
Next >
Finish

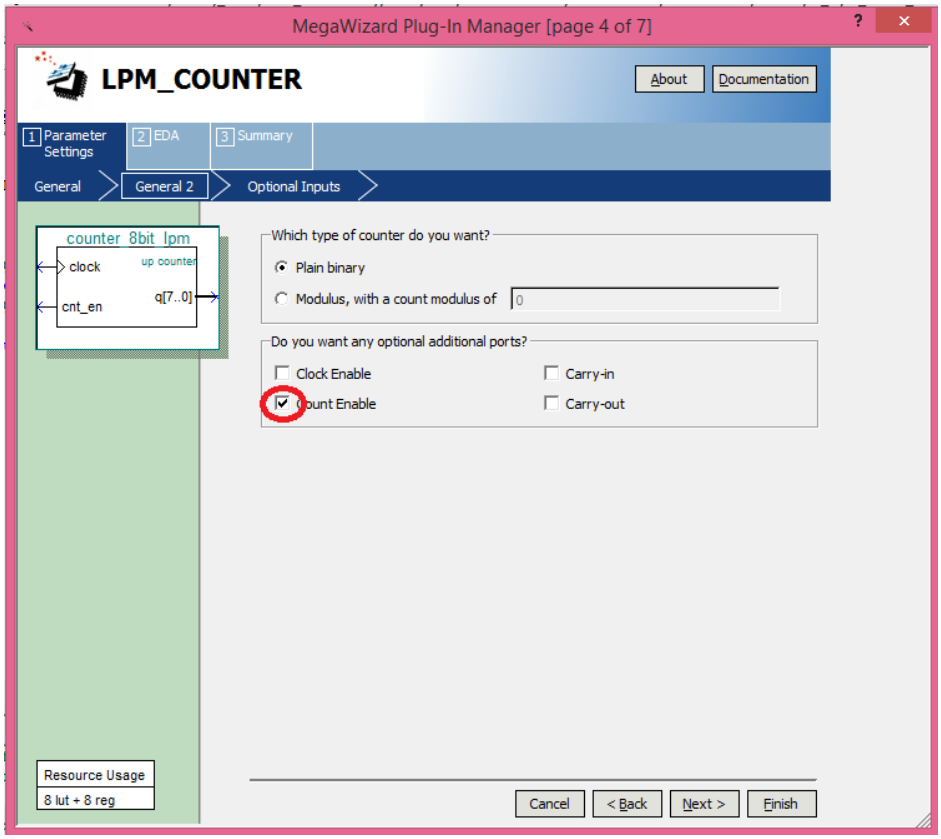

MegaWizard Plug-In Manager [page 2a]
Which megafunction would you like to customize?
Select a megafunction from the list below
Arithmetic
ALTECC
ALTFP_ABS
ALTFP_ADD_SUB
ALTFP_ATAN
ALTFP_COMPARE
ALTFP_CONVERT
ALTFP_DIV
ALTFP_EXP
ALTFP_INV
ALTFP_INV_SQRT
ALTFP_LOG
ALTFP_MATRIX_INV
ALTFP_MATRIX_MULT
ALTFP_MULT
ALTFP_SINCOS
ALTFP_SQRT
ALTMEMMULT
ALTMULT_ACCUM (MAC)
ALTMULT_ADD
ALTMULT_COMPLEX
ALTSQRT
LPM_ADD_SUB
LPM_COMPARE
LPM_COUNTER
LPM_DIVIDE
LPM_MULT
PARALLEL_ADD
Floating point comparator megafunction
Which device family will you be using?
Cyclone II
Which type of output file do you want to create?
AHDL
VHDL
Verilog HDL
What name do you want for the output file?
niversity\Subjects\Circuit Design with HDL\Thuc hanh\Lab 3\Counter_LPM\counter_8bit_lpm
Return to this page for another create operation
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in a library specified in the Libraries page of the Options dialog box (Tools menu), or a library specified in the Libraries page of the Settings dialog box (Assignments menu).
Your current user library directories are:
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 5 of 7]
LPM_COUNTER
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
General 2
Optional Inputs
counter_8bit_lpm
sclr
clock
cnt_en
up counter
q[7..0]
Do you want any optional inputs?
Synchronous inputs
Clear
Load
Set
Set to all 1's
Set to
0
Asynchronous inputs
Clear
Load
Set
Set to all 1's
Set to
0
Resource Usage
8 lut + 8 reg
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 6 of 7]
LPM_COUNTER
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
counter_8bit_lpm
sclr
clock
cnt_en
up counter
q[7..0]
Simulation Libraries
To properly simulate the generated design files, the following simulation model file(s) are needed
File
Description
lpm
LPM megafunction simulation library
Timing and resource estimation
Generates a netlist for timing and resource estimation for this megafunction. If you are synthesizing your design with a third-party EDA synthesis tool, using a timing and resource estimation netlist can allow for better design optimization.
Not all third-party synthesis tools support this feature - check with the tool vendor for complete support information.
Note: Netlist generation can be a time-intensive process. The size of the design and the speed of your system affect the time it takes for netlist generation to complete.
Generate netlist
Resource Usage
8 lut + 8 reg
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 7 of 7]
LPM_COUNTER
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
counter_8bit_lpm
sclr
clock
cnt_en
up counter
q[7..0]
Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a green checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.
The MegaWizard Plug-In Manager creates the selected files in the following directory:
E:\Sang_new\UIT_University\Subjects\Circuit Design with HDL\Thuc hanh\Lab 3\Counter_LPM\
File
Description
counter_8bit_lpm.v
Variation file
counter_8bit_lpm.inc
AHDL Include file
counter_8bit_lpm.cmp
VHDL component declaration file
counter_8bit_lpm.bsf
Quartus II symbol file
counter_8bit_lpm_inst.v
Instantiation template file
counter_8bit_lpm_bb.v
Verilog HDL black-box file
Resource Usage
8 lut + 8 reg
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 3 of 7]
LPM_COUNTER
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
General 2
Optional Inputs
counter_8bit_lpm
clock
up counter
q[7..0]
Currently selected device family:
Cyclone II
Match project/default
How wide should the 'q' output bus be?
8
bits
What should the counter direction be?
Up only
Down only
Create an 'updown' input port to allow me to do both (1 counts up; 0 counts down)
Resource Usage
8 lut + 8 reg
Cancel
< Back
Next >
Finish

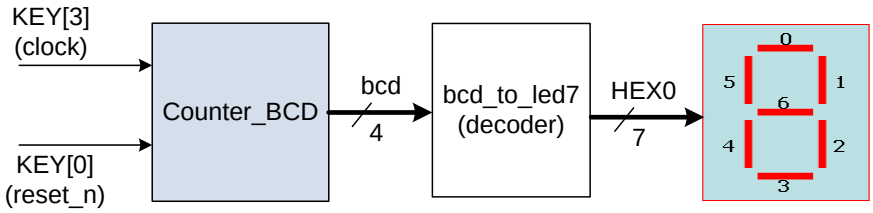

MegaWizard Plug-In Manager [page 4 of 7]
LPM_COUNTER
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
General 2
Optional Inputs
counter_8bit_lpm
clock
cnt_en
up counter
q[7..0]
Which type of counter do you want?
Plain binary
Modulus, with a count modulus of
0
Do you want any optional additional ports?
Clock Enable
Carry-in
Count Enable
Carry-out
Resource Usage
8 lut + 8 reg
Cancel
< Back
Next >
Finish

- Thực hiện gán chân giống câu 2.2.
- So sánh tần số Fmax so với bộ đếm trong câu 2.2

Lưu ý: Sinh viên tham khảo thêm cách sử dụng các LPM trong file "tut_lpms_verilog.pdf"

Câu 3.

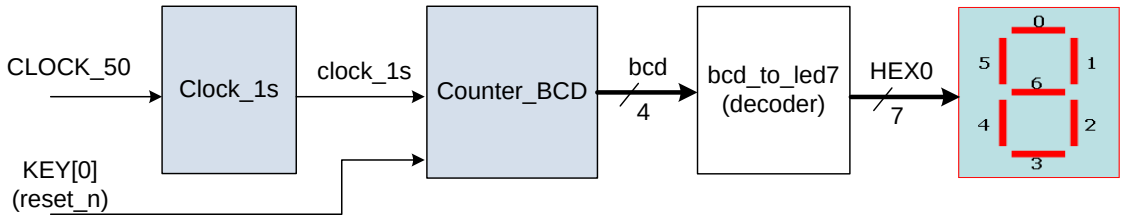
3.1.Thiết kế mạch đếm BCD 0 đến 9 sao cho giá trị bộ đếm tăng lên 1 khi có cạnh lên của xung clock KEY[3].



Tín hiệu **reset_n** là tín hiệu reset bất đồng bộ và tích cực mức thấp

Yêu cầu: Sinh viên thực hiện bộ **decoder** bằng lệnh "**case** … **endcase**"

3.2.Thiết kế mạch đếm BCD 0 đến 9 sao cho giá trị bộ đếm tăng lên 1 sau mỗi 1s



Sử dụng xung CLOCK_50 (50 MHz) để tạo xung clock_1s (1Hz)

Yêu cầu: SV vẽ lưu đồ giải thuật cho khối tạo xung "Clock_1s"

Câu 4.

4.1. Thực hiện bảng chạy chữ như mô tả ở hình dưới. Sử dụng HEX7-0 để hiển thị kí tự.

Sử dụng:

- KEY[0] làm xung clock để điều khiển sự dịch chuyển.
- KEY[1] làm tín hiệu reset của mạch (khi reset tích cực, thì các HEX hiển thị dòng chữ tại Clock cycle 0)

Yêu cầu: SV thiết kế SPEC (sơ đồ khối) để thực hiện mạch trên

Clock cycle	Character pattern							
	H7	H6	H5	H4	H3	H2	H1	H0
0				H	E	L	L	O
1			H	E	L	L	O	
2		H	E	L	L	O		
3	H	E	L	L	O			
4	E	L	L	O				H
5	L	L	O				H	E
6	L	O				H	E	L
7	O				H	E	L	L
8				H	E	L	L	O
. . .	and so on							

4.2. Dùng CLOCK_50 để thay KEY[0] trong câu a và điều khiển mạch sao cho các HEX sẽ tự động dịch chuyển sau khoảng thời gian 1s

Câu 5.

Thiết kế bộ đếm 2-digit BCD counter đếm các giá trị từ 00 đến 20. Giá trị đếm được hiển thị lên hai Led 7 đoạn (HEX0 và HEX1). Bộ đếm có thể được Reset (bất đồng bộ) về 0 khi KEY[0] được nhấn. Mỗi giá trị đếm được hiển thị trong 1s, sử dụng CLOCK_50 là giá trị xung clock tham khảo. Kiểm tra thiết kế trên board DE2.

Câu 6.

Hiện thực một **đồng hồ hiển thị giờ, phút, giây trong ngày**. Đồng hồ sẽ thể hiện giá trị "giờ" (từ 0 đến 23) lên các led 7-đoạn HEX7-6, giá trị "phút" (từ 0 đến 60) lên các led HEX5-4, và giá trị "giây" (từ 0 đến 60) lên các led HEX3-2. Sử dụng các SW15-0 để reset lại giá trị "giờ" và "phút" cho đồng hồ.

Mạch hiện thực phải có khả năng báo lỗi hoặc không cho thiết lập các giá trị giờ, phút, giây bất hợp lý. Kiểm tra thiết kế trên board DE2.

TÀI LIỆU THAM KHẢO

Digital_Logic lab của **Altera.**