Lab 1: Hello µC/OS-II

How to create a SOPC system running µC/OS-II ?

Written by Geng You Chen
(gengyouchen@gmail.com)

Experiment Goal

- Design a SOPC system running two C/C++ functions at the same time:
  - Function 1: Print some messages on Character LCD every 3 seconds.
  - Function 2: Control Red LEDs according to Toggle Switches anytime.
- Traditional C/C++ program has only one task, that means you can only run one function at the same time and you need to wait until Function 1 has returned in order to call Function 2. In other word, you cannot run more than one functions at the same time.
- In fact, a CPU can only run one task at a time. Multi-task C/C++ programs need the help of OS (Operating System) to do Context Switching. For your convenience, Altera Corporation has integrated µC/OS-II into Nios II EDS.
- To run µC/OS-II, you need to add a Timer (designed by Altera Corporation) on the bus to interrupt CPU every 1 millisecond (this is the default value). When CPU is interrupted, it will stop the current task and run µC/OS-II. µC/OS-II will save the current task and load another task into CPU to run until next interrupt occurs. In other words, µC/OS-II do Context Switching every 1 millisecond. Because Context Switching is very fast, it “seems like” more than one task are running at the same time.

Experiment Equipment

- On-chip: FPGA Chip (Altera Cyclone II EP2C70F896C6N)
  - Everything (e.g. LEs, PLLs, M4Ks, Multiplier) is programmable.
  - After programming, these elements behave like a system including CPU, bus, controllers (for off-chip device), timers (for µC/OS-II) and so on.
- Off-chip: Development Board (Terasic DE2-70 Board)
  - Everything is ready-made and fixed on the board.
  - In Lab 1, you will only use some basic IO devices (e.g. LEDs, Switches, Characters LCD, JTAG) and memory devices (e.g. SSRAM, SDRAM, Flash).
  - Also, because the above devices all come from the original Altera board, Altera SOPC Builder can auto-generate all the device controllers for you. In other words, there's no IP design in Lab 1.
- EDA (Electronic Design Automation) Tools
  - Altera Quartus II (including SOPC Builder)
  - Altera Nios II EDS (including µC/OS-II)

On-chip…

- 68416 LEs (Logic Elements)
  - Most of the hardware design are implemented by using many LEs.
  - Each LE has a Programmable Register and a 4-input LUT (Look-Up Table) which can behave like any function of 4 variables.
- 4 PLLs (Phase-Locked Loops)
  - To write a Frequency Divider is simple. However, it's almost impossible to write a Frequency Multiplier unless you can precisely control the delay of each elements.
  - PLL can precisely multiply the frequency or shift the phase because it's analog.
  - The CPU designed by Altera runs 100MHz, but the oscillator in Terasic DE2-70 Board can only generate 50MHz clock. Therefore, you need a PLL to double the frequency.
- 250 M4K Memory Blocks
  - Compared with SSRAM, SDRAM and Flash, M4K is the fastest one because it's on-chip.
  - Each M4K has only 4096 bits for data and 512 parity bits for error detection.
  - Some IPs (e.g. CPU) need to be implemented by using some M4Ks.
  - Also, if your C/C++ program is small enough, you can use the rest of M4Ks to create an on-chip memory and burn your C/C++ program into this memory. Therefore, your SOPC system doesn't need any other memories such as SSRAM, SDRAM and Flash.
- 150 Embedded Multiplier Blocks
  - Each multiplier can be two 9-bit x 9-bit multipliers or one 18-bit x 18-bit multiplier.
  - Some IPs (e.g. CPU) need to be implemented by using some multipliers.

*Off-chip…

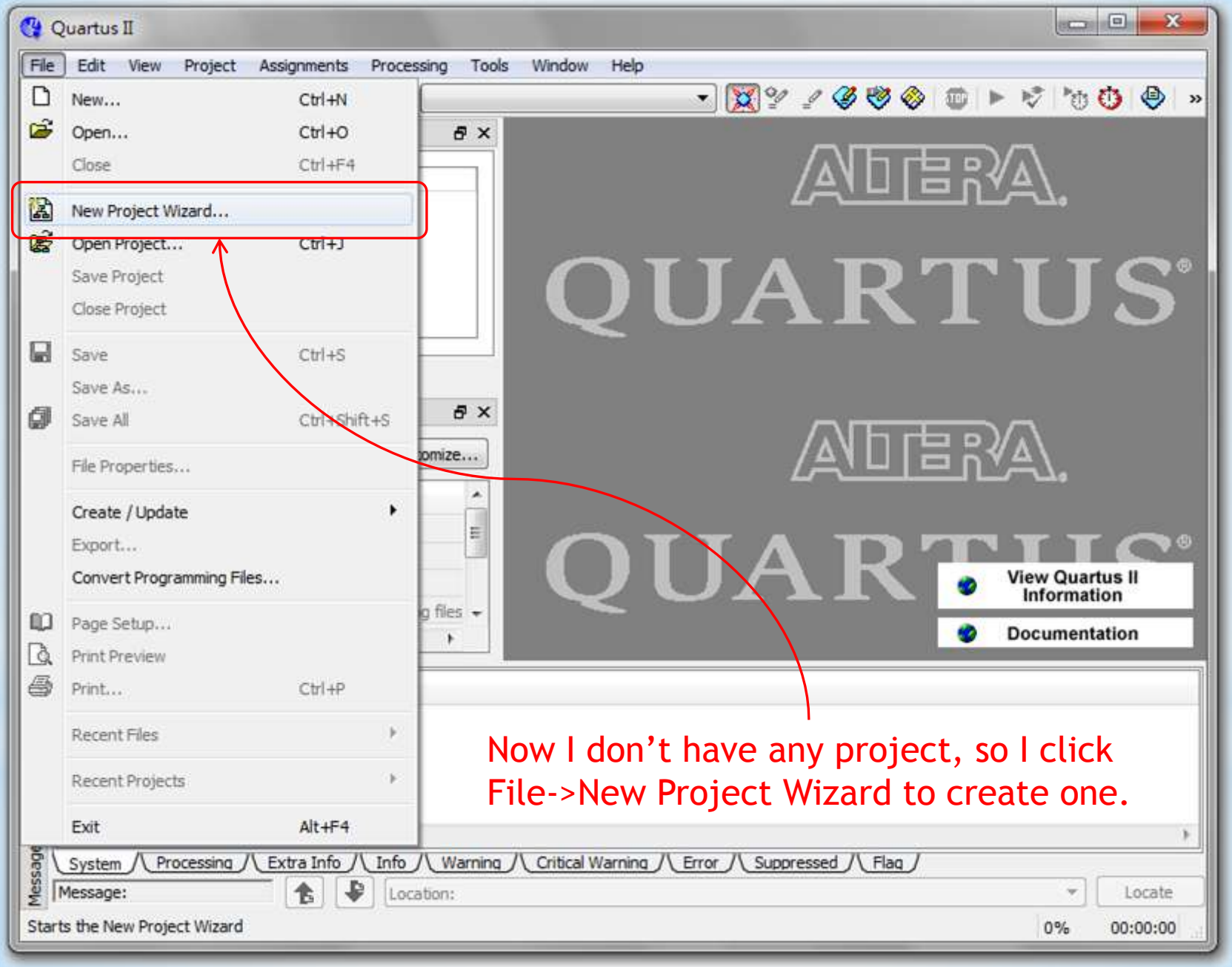
* 18 Red and 8 Green LEDs
* 18 Toggle and 4 Push-Button Switches
* Characters LCD
  * You can set Characters LCD as Standard C/C++ Output for CPU.
* JTAG (Joint Test Action Group)
  * Just like Characters LCD, JTAG uses Nios II EDS as Standard C/C++ Input/Output.
* 2MB SSRAM (Synchronous Static Random Access Memory)
  * Each memory cell use two inverters as a loop to latch your data, which means one of the node voltage enhances the other one and vice versa. Therefore, your data won't be lost unless you turn off the power.
  * 2MB is large enough for most of the C/C++ program. However, the SSRAM Controller designed by Altera has Tristate Interface. To attach it to the bus, you need to add a Tristate Bridge between the SSRAM Controller and the bus.
* 64MB SDRAM (Synchronous Dynamic Random Access Memory)
  * Each memory cell use a switch to cut-off your data in a capacitor, which means the electric charges stored in the capacitor will be lost during each reading. Therefore, in order to prevent data loss, SDRAM need to recharge capacitors frequently.
  * 64MB is large enough even if the C/C++ program is huge. However, for 100MHz CPU, the clock for the SDRAM Controller is the 100MHz but shifted -65 degree. You need to use PLLs to shift the degree. This -65 degree is just an experiment statistics of some clock skew issues in synchronous hardware, and it's not obvious at 50MHz.
* 8MB Flash Memory
  * Flash Memory is the only NVM (Non-Volatile Memory) in Terasic DE2-70 Board, that means the data won't be lost even if the power is off.
  * Just like SSRAM, you need a Tristate Bridge between Flash Controller and the bus.

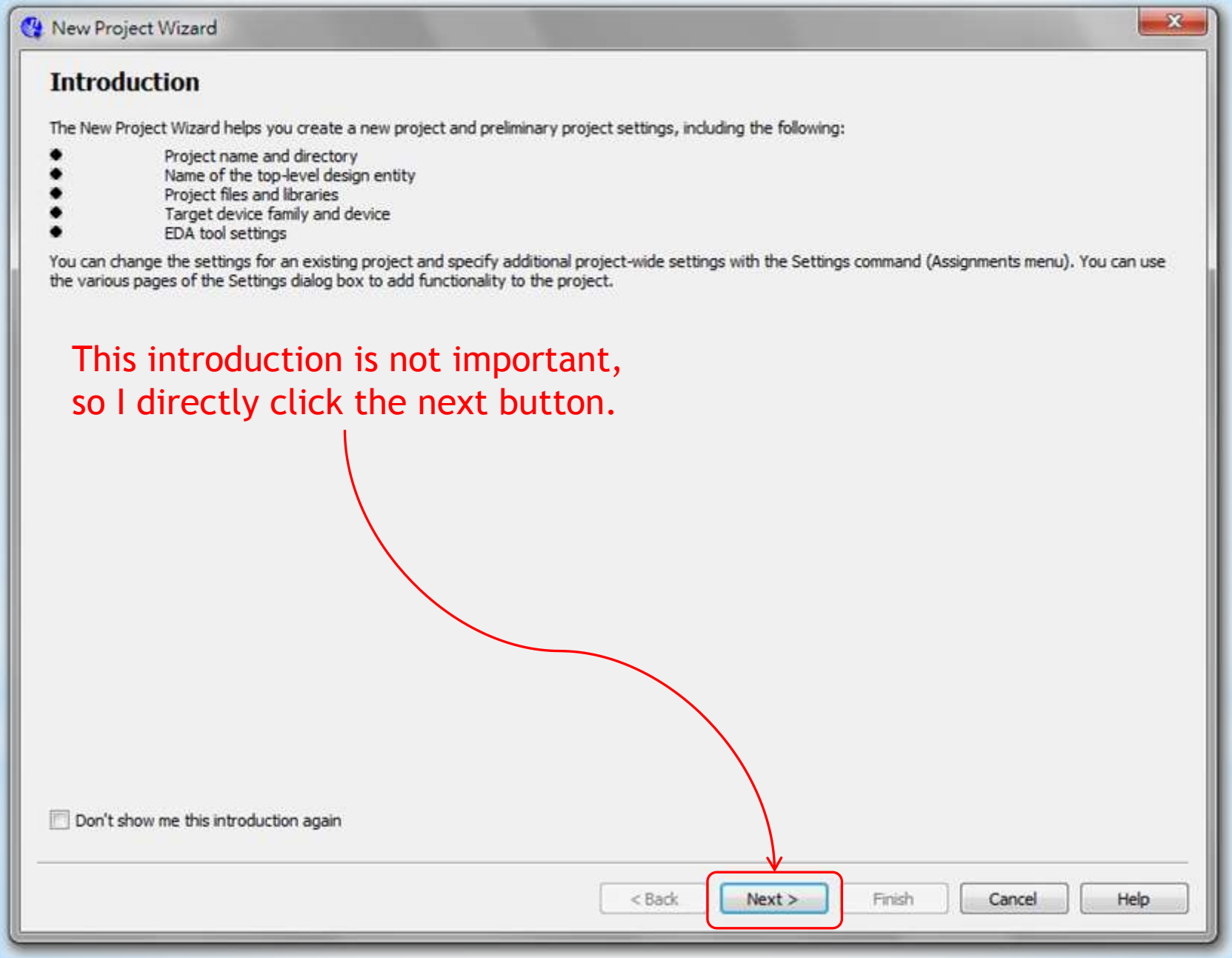
*EDA Tools…

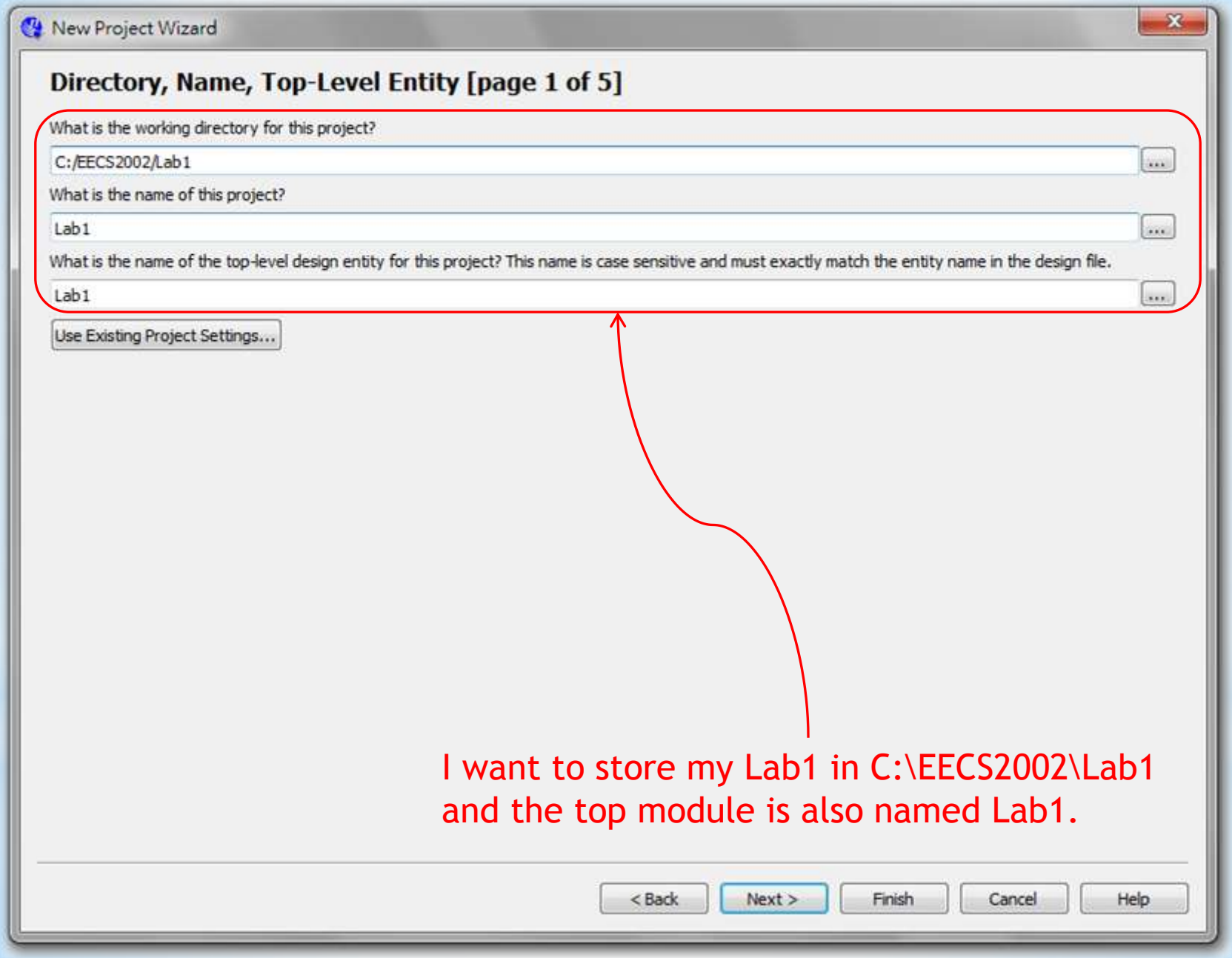
* Altera Quartus II
  * Use SOPC Builder to modify Nios II CPU.
    * Because Nios II CPU is soft-core CPU, which means it's Verilog/VHDL codes, you can use SOPC Builder to modify it. For convenience, SOPC Builder provide 3 default settings: standard, economy (small size) and fast (high speed).
    * After hardware reset, CPU will run from Reset Vector you set in SOPC Builder. Because on-chip memory, SSRAM and SDRAM will be cleared after reset, if you want to run your program immediately after you push Reset Button, you need to burn your program into Flash Memory and set Reset Vector to Flash Memory. If you don't care what will happen after you push Reset Button, Reset Vector can be anything because after you use Nios II EDS to burn your program into any memory, Nios II EDS will automatically set CPU to run from that memory.
    * After hardware interrupt, CPU will run from Exception Vector you set in SOPC Builder. If your system doesn't have interrupt mechanism, Exception Vector can be anything. However, you use interrupt mechanism for running µC/OS-II so Exception Vector must be set as the memory where the µC/OS-II is burned. This is almost the same memory where your C/C++ program is burned.
* Altera Nios II EDS
  * Use "Hello µC/OS-II" Project instead of "Hello World" Project.
    * All µC/OS-II system files will be auto-generated into your C/C++ Project.
    * Also, an example main function for using µC/OS-II will be auto-generated, too. This is a good example for anyone who want to learn using µC/OS-II.

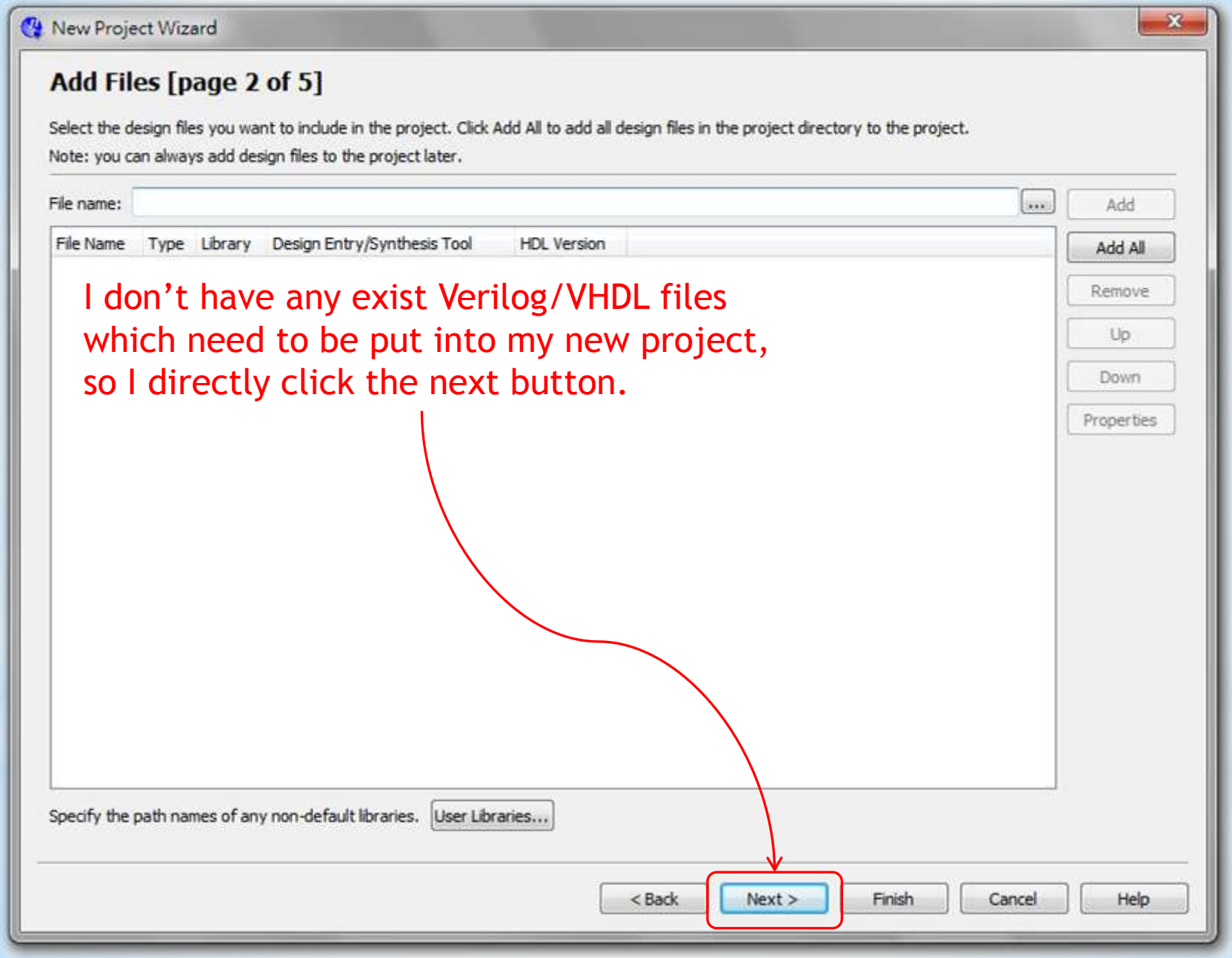
Experiment Method

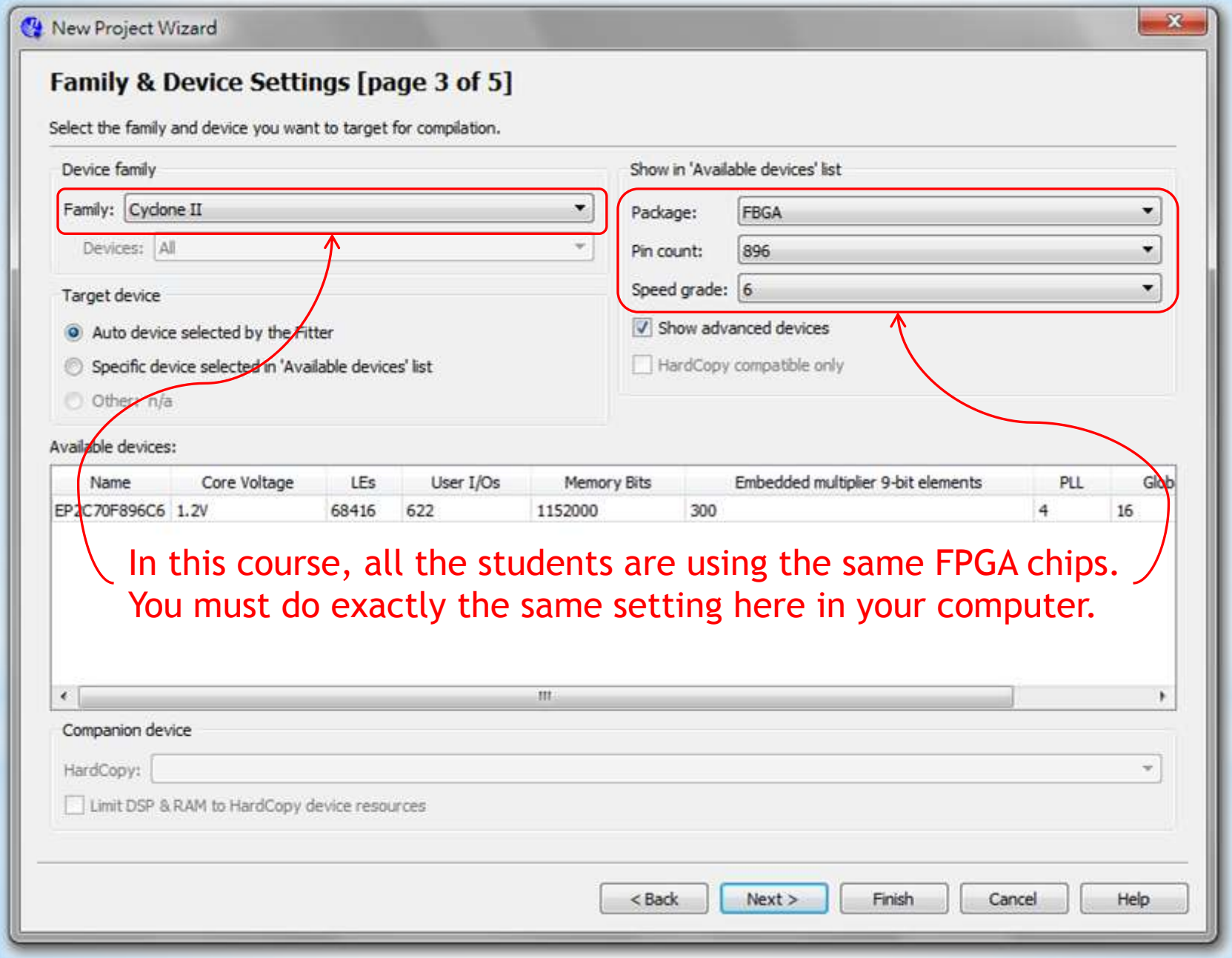
- Because this is the first lab in this course, I will demonstrate every steps for you. My demonstration environment are:
  - Microsoft Windows 7 SP1 Enterprise (x64)
  - Altera Quartus II Subscription Edition 10.1 SP1
  - Altera Nios EDS II 10.1 SP1
  - Terasic DE2-70 CD-ROM Version 1.4
- Before you look my demonstration, please make sure you've been "very familiar" with the following topics I've discussed:
  - Introduction to SOPC Lab, page 1 ~ 9, written by Geng You Chen
  - Lab 1: Hello µC/OS-II, page 1 ~ 6, written by Geng You Chen

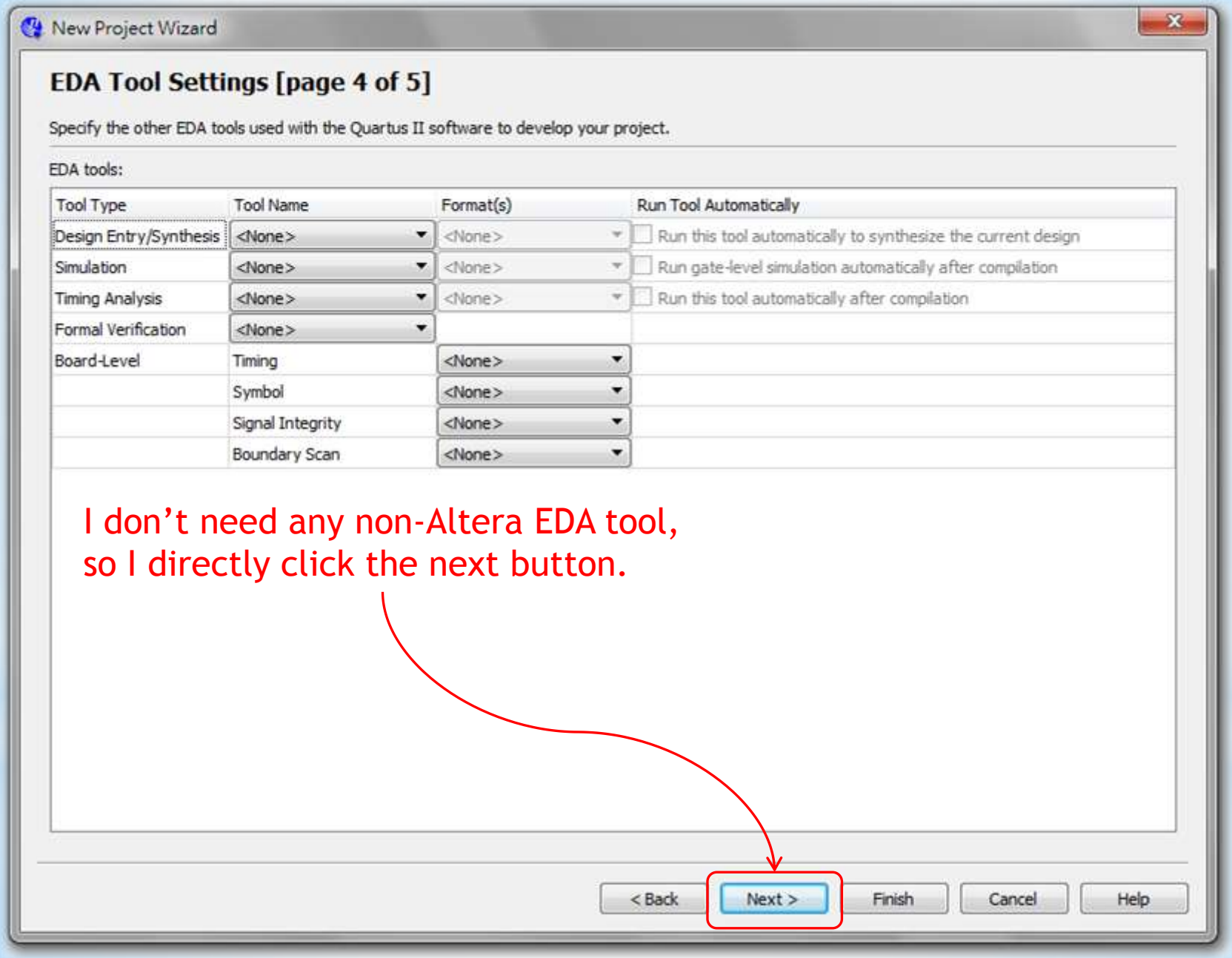

Quartus II
File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Save Project
Close Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
File Properties...
Create / Update
Export...
Convert Programming Files...
Page Setup...
Print Preview
Print... Ctrl+P
Recent Files
Recent Projects
Exit Alt+F4
ALTERA
QUARTUS
View Quartus II Information
Documentation
Now I don't have any project, so I click File->New Project Wizard to create one.
System
Processing
Extra Info
Info
Warning
Critical Warning
Error
Suppressed
Flag
Message:
Location:
Locate
Starts the New Project Wizard
0%
00:00:00

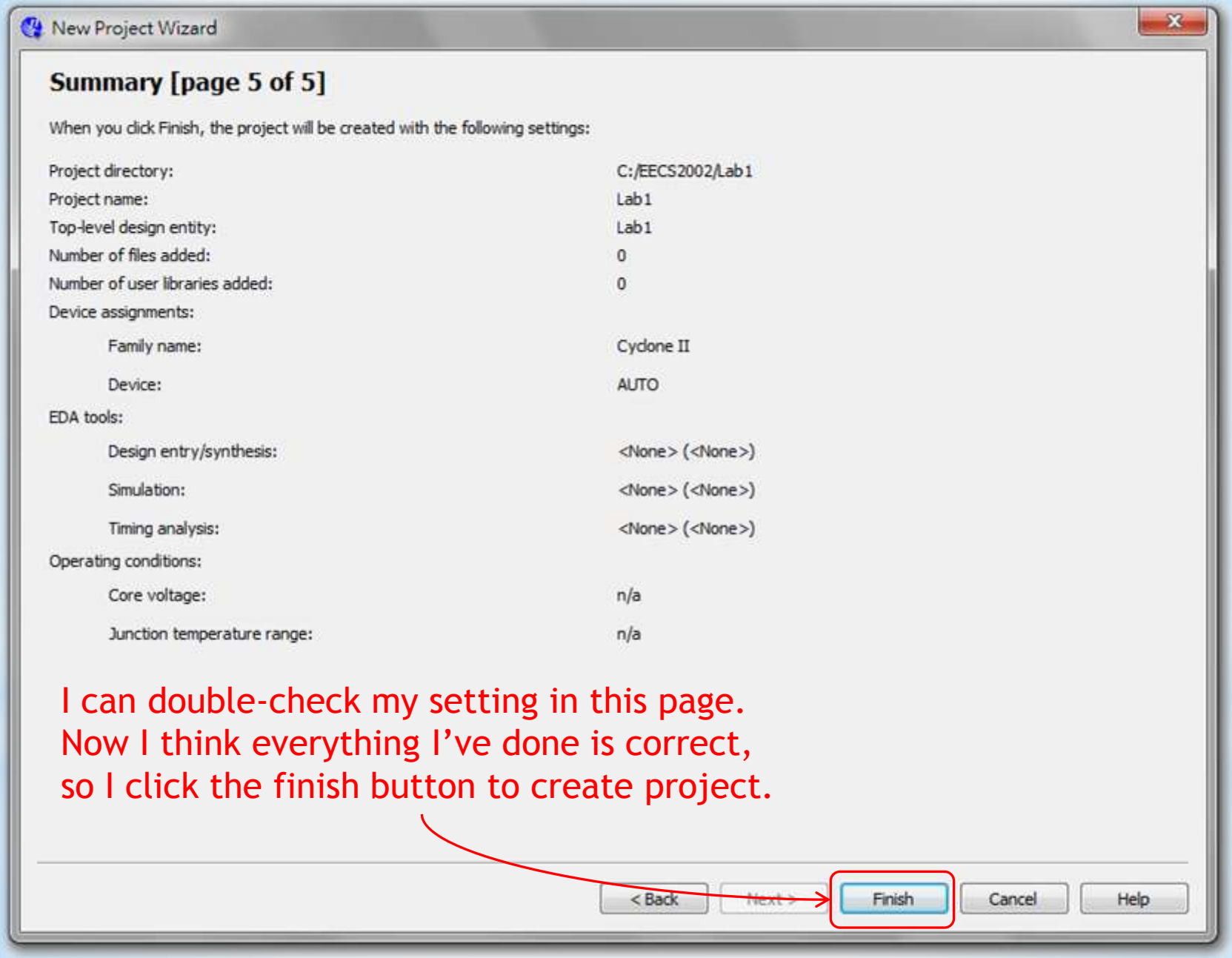

New Project Wizard
Introduction
The New Project Wizard helps you create a new project and preliminary project settings, including the following:
Project name and directory
Name of the top-level design entity
Project files and libraries
Target device family and device
EDA tool settings
You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.
This introduction is not important,
so I directly click the next button.
Don't show me this introduction again
< Back
Next >
Finish
Cancel
Help

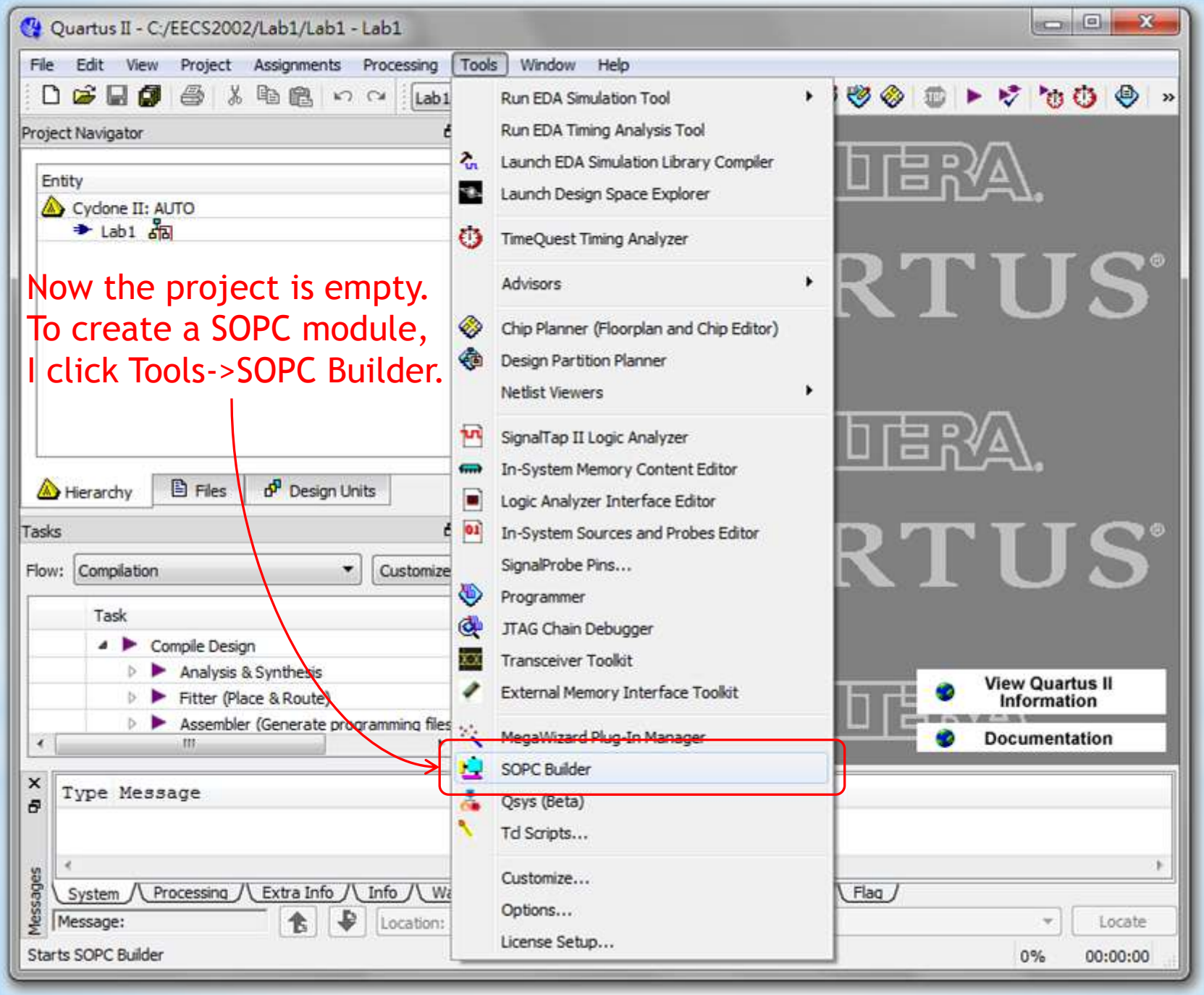

New Project Wizard
Directory, Name, Top-Level Entity [page 1 of 5]
What is the working directory for this project?
C:/EECS2002/Lab1
What is the name of this project?
Lab1
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
Lab1
Use Existing Project Settings...
I want to store my Lab1 in C:\EECS2002\Lab1
and the top module is also named Lab1.
< Back
Next >
Finish
Cancel
Help


New Project Wizard
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:
...
Add
Add All
Remove
Up
Down
Properties
File Name
Type
Library
Design Entry/Synthesis Tool
HDL Version
I don’t have any exist Verilog/VHDL files
which need to be put into my new project,
so I directly click the next button.
Specify the path names of any non-default libraries.
User Libraries...
< Back
Next >
Finish
Cancel
Help

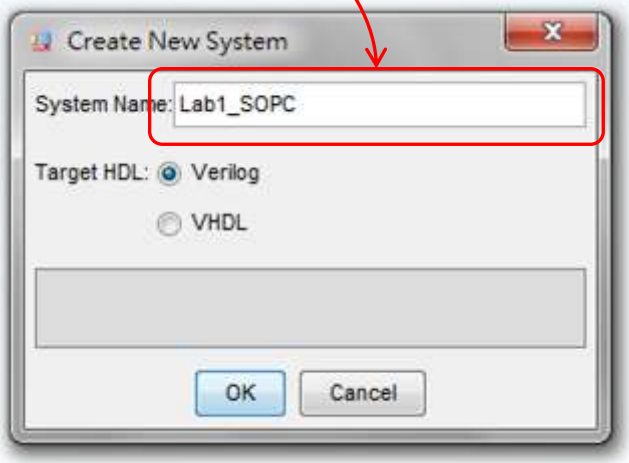

New Project Wizard
Family & Device Settings [page 3 of 5]
Select the family and device you want to target for compilation.
Device family
Family: Cyclone II
Devices: All
Target device
Auto device selected by the Fitter
Specific device selected in 'Available devices' list
Other: n/a
Show in 'Available devices' list
Package: FBGA
Pin count: 896
Speed grade: 6
Show advanced devices
HardCopy compatible only
Available devices:
Name
Core Voltage
LEs
User I/Os
Memory Bits
Embedded multiplier 9-bit elements
PLL
EP2C70F896C6
1.2V
68416
622
1152000
300
4
16
In this course, all the students are using the same FPGA chips.
You must do exactly the same setting here in your computer.
Companion device
HardCopy:
Limit DSP & RAM to HardCopy device resources
< Back
Next >
Finish
Cancel
Help

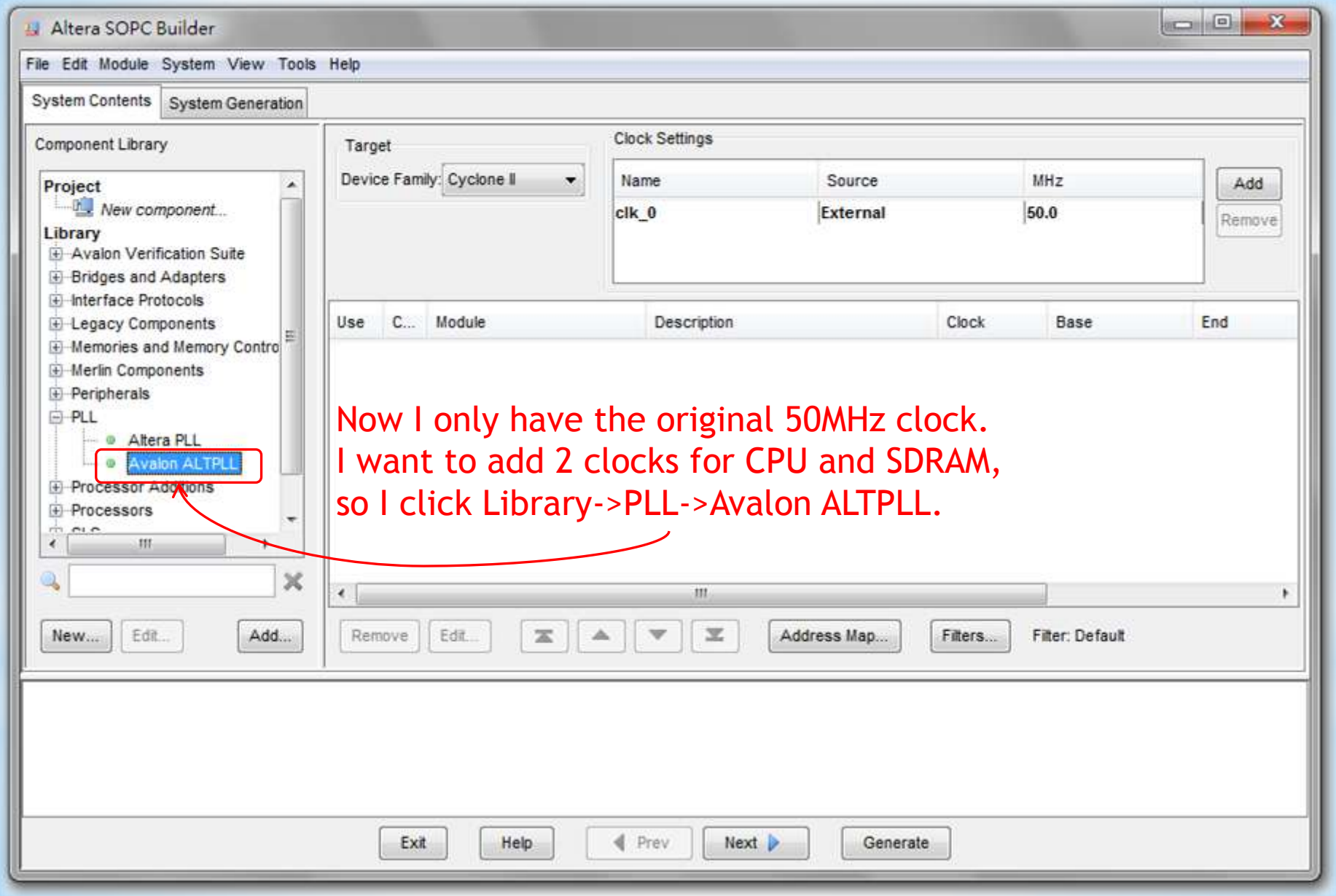

New Project Wizard
EDA Tool Settings [page 4 of 5]
Specify the other EDA tools used with the Quartus II software to develop your project.
EDA tools:
Tool Type
Tool Name
Format(s)
Run Tool Automatically
Design Entry/Synthesis
<None>
Run this tool automatically to synthesize the current design
Simulation
Run gate-level simulation automatically after compilation
Timing Analysis
Run this tool automatically after compilation
Formal Verification
Board-Level
Timing
Symbol
Signal Integrity
Boundary Scan
I don’t need any non-Altera EDA tool,
so I directly click the next button.
< Back
Next >
Finish
Cancel
Help

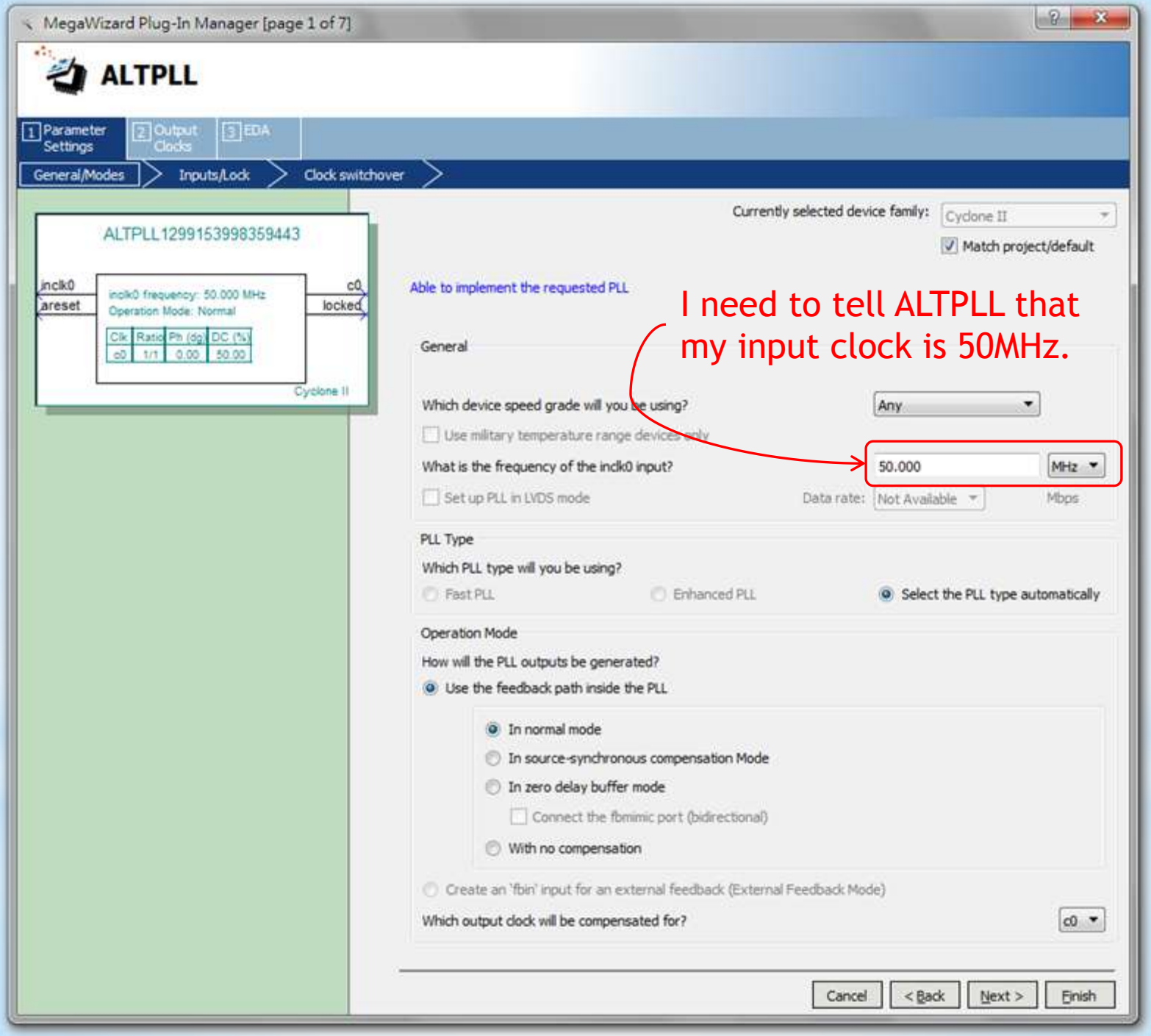

New Project Wizard
Summary [page 5 of 5]
When you click Finish, the project will be created with the following settings:
Project directory: C:/EECS2002/Lab1
Project name: Lab1
Top-level design entity: Lab1
Number of files added: 0
Number of user libraries added: 0
Device assignments:
Family name: Cyclone II
Device: AUTO
EDA tools:
Design entry/synthesis: <None> (<None>)
Simulation: <None> (<None>)
Timing analysis: <None> (<None>)
Operating conditions:
Core voltage: n/a
Junction temperature range: n/a
I can double-check my setting in this page.
Now I think everything I've done is correct,
so I click the finish button to create project.
< Back
Next >
Finish
Cancel
Help

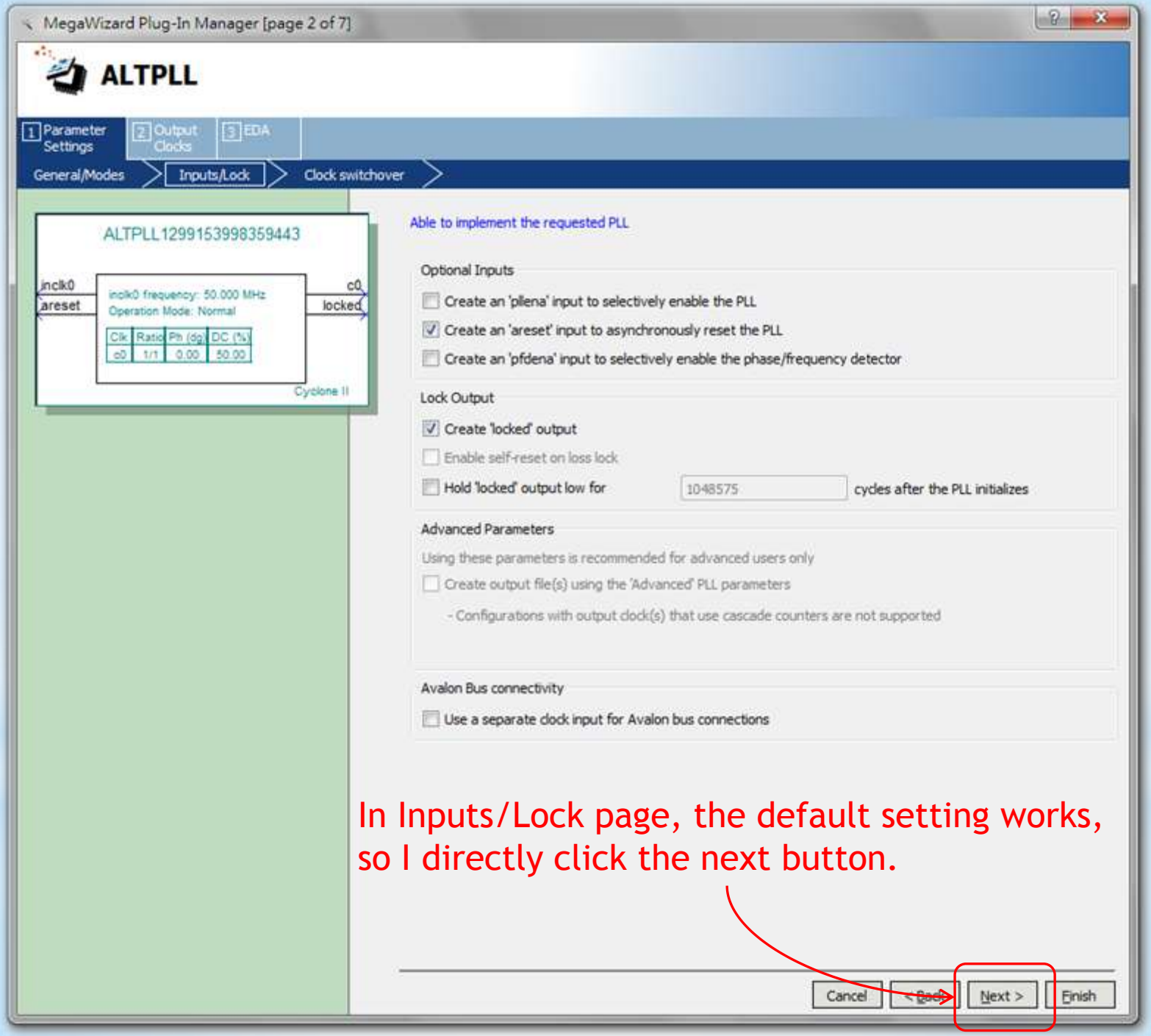

Quartus II - C:/EECS2002/Lab1/Lab1 - Lab1
File Edit View Project Assignments Processing Tools Window Help
Project Navigator
Entity
Cyclone II: AUTO
Lab1
Now the project is empty.
To create a SOPC module,
I click Tools->SOPC Builder.
Hierarchy Files Design Units
Tasks
Flow: Compilation
Customize
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files
Run EDA Simulation Tool
Run EDA Timing Analysis Tool
Launch EDA 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 (Beta)
Tcl Scripts...
Customize...
Options...
License Setup...
View Quartus II Information
Documentation
Type Message
System Processing Extra Info Info
Flag
Message:
Location:
Locate
Starts SOPC Builder
0%
00:00:00

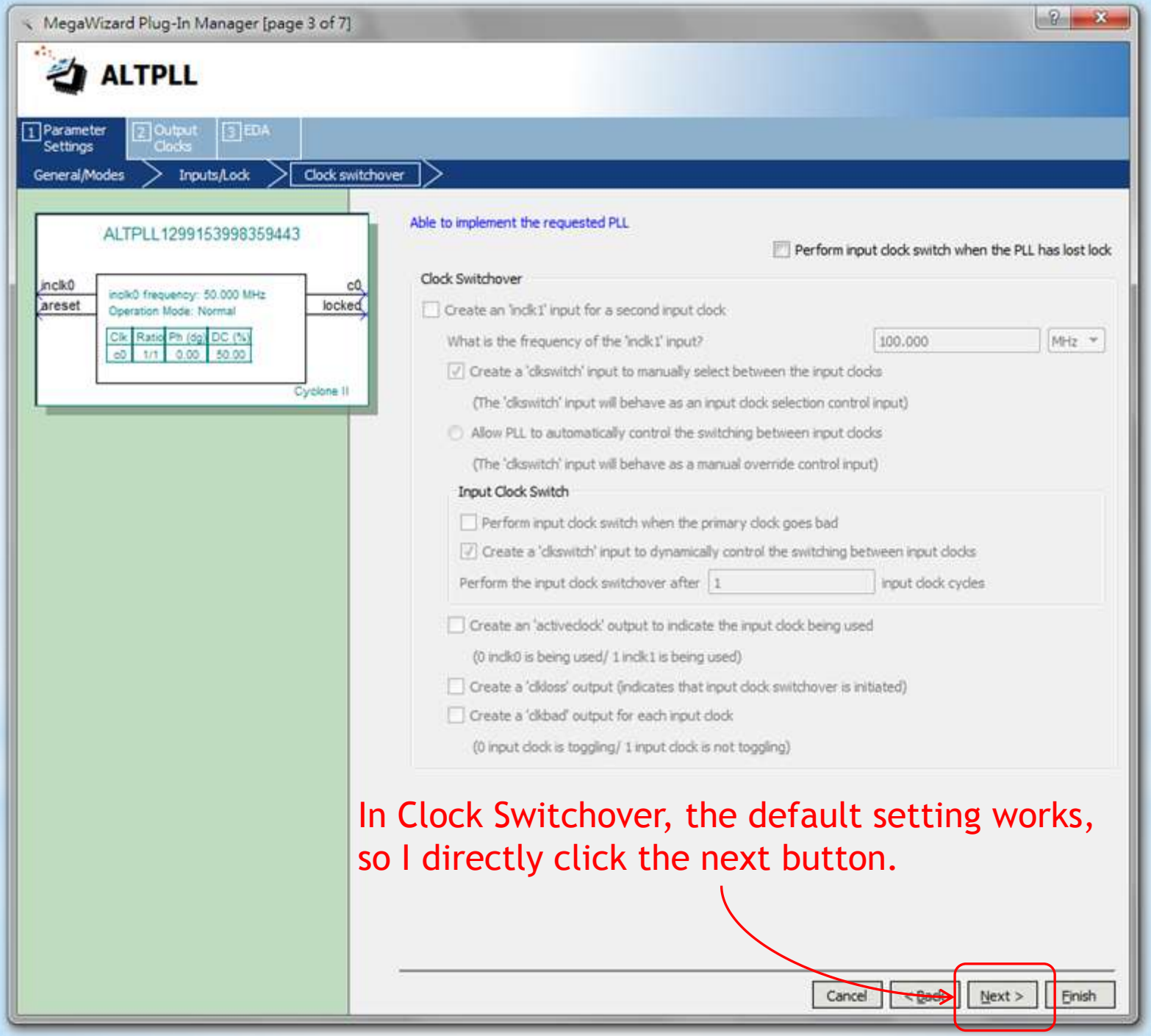
I want my SOPC module name be Lab1_SOPC.

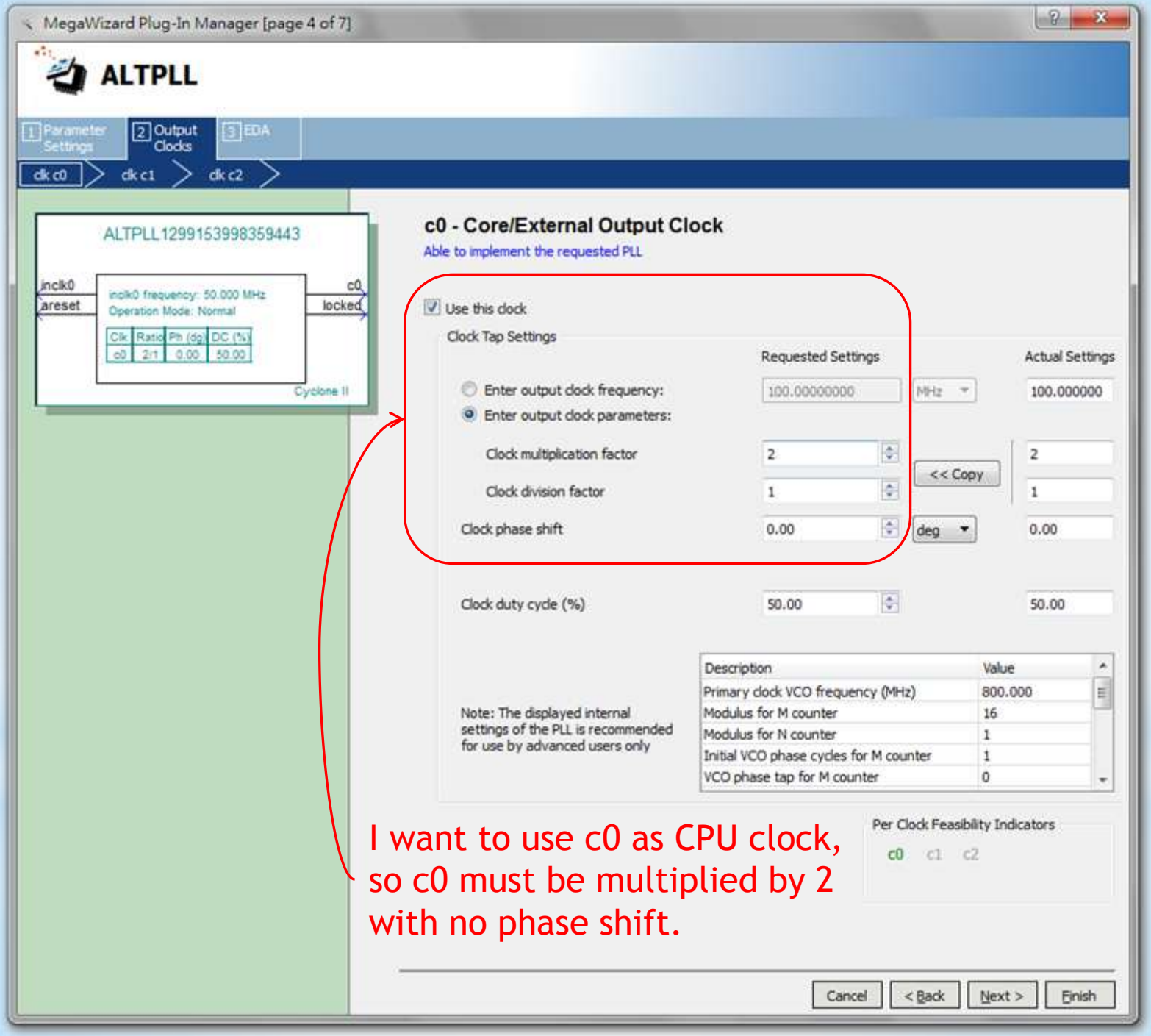


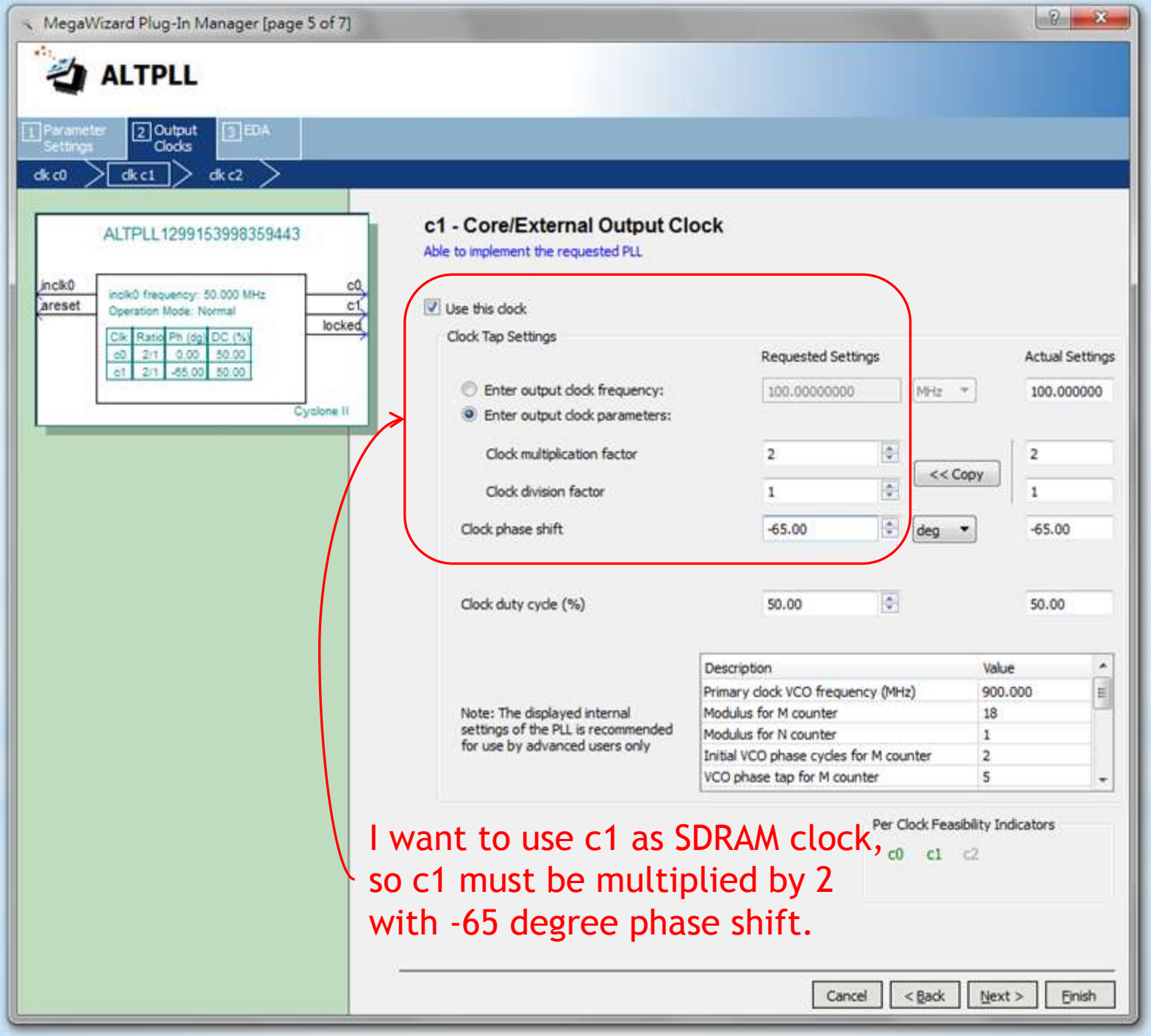

Altera SOPC Builder
File Edit Module System View Tools Help
System Contents
System Generation
Component Library
Project
New component...
Library
Avalon Verification Suite
Bridges and Adapters
Interface Protocols
Legacy Components
Memories and Memory Contro
Merlin Components
Peripherals
PLL
Altera PLL
Avalon ALTPLL
Processor Additions
Processors
New...
Edit...
Add...
Target
Device Family: Cyclone II
Clock Settings
Name
Source
MHz
clk_0
External
50.0
Add
Remove
Use
C...
Module
Description
Clock
Base
End
Remove
Edit...
Address Map...
Filters...
Filter: Default
Exit
Help
Prev
Next
Generate
Now I only have the original 50MHz clock.
I want to add 2 clocks for CPU and SDRAM,
so I click Library->PLL->Avalon ALTPLL.

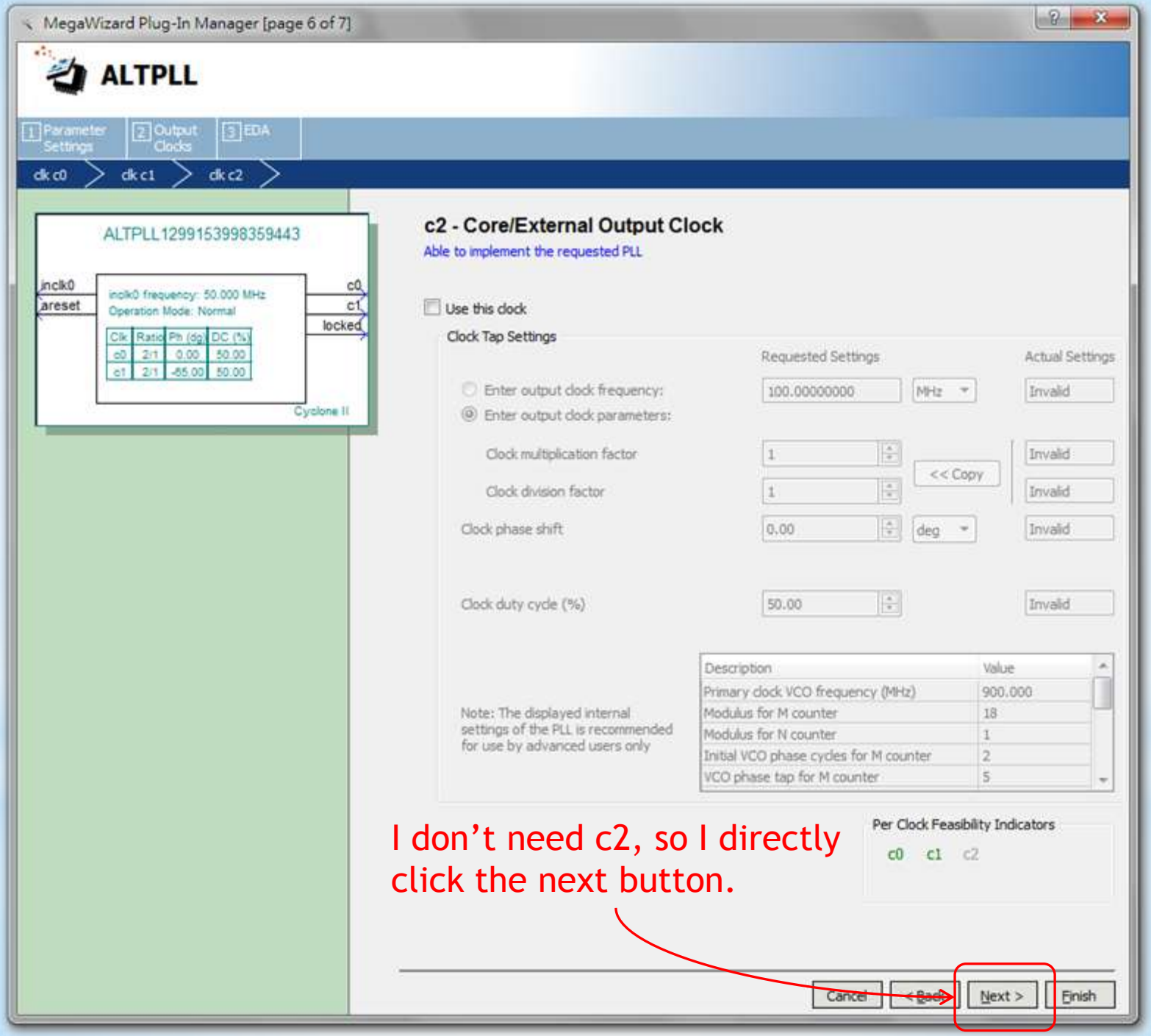

MegaWizard Plug-In Manager [page 1 of 7]
ALTPLL
1 Parameter Settings
2 Output Clocks
3 EDA
General/Modes
Inputs/Lock
Clock switchover
ALTPLL1299153998359443
inclk0
areset
c0
locked
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk
Ratio
Ph (dg)
DC (%)
c0
1/1
0.00
50.00
Cyclone II
Currently selected device family:
Cyclone II
Match project/default
Able to implement the requested PLL
I need to tell ALTPLL that my input clock is 50MHz.
General
Which device speed grade will you be using?
Any
Use military temperature range devices only
What is the frequency of the inclk0 input?
50.000
MHz
Set up PLL in LVDS mode
Data rate:
Not Available
Mbps
PLL Type
Which PLL type will you be using?
Fast PLL
Enhanced PLL
Select the PLL type automatically
Operation Mode
How will the PLL outputs be generated?
Use the feedback path inside the PLL
In normal mode
In source-synchronous compensation Mode
In zero delay buffer mode
Connect the fbmimic port (bidirectional)
With no compensation
Create an 'fbin' input for an external feedback (External Feedback Mode)
Which output clock will be compensated for?
c0
Cancel
< Back
Next >
Finish

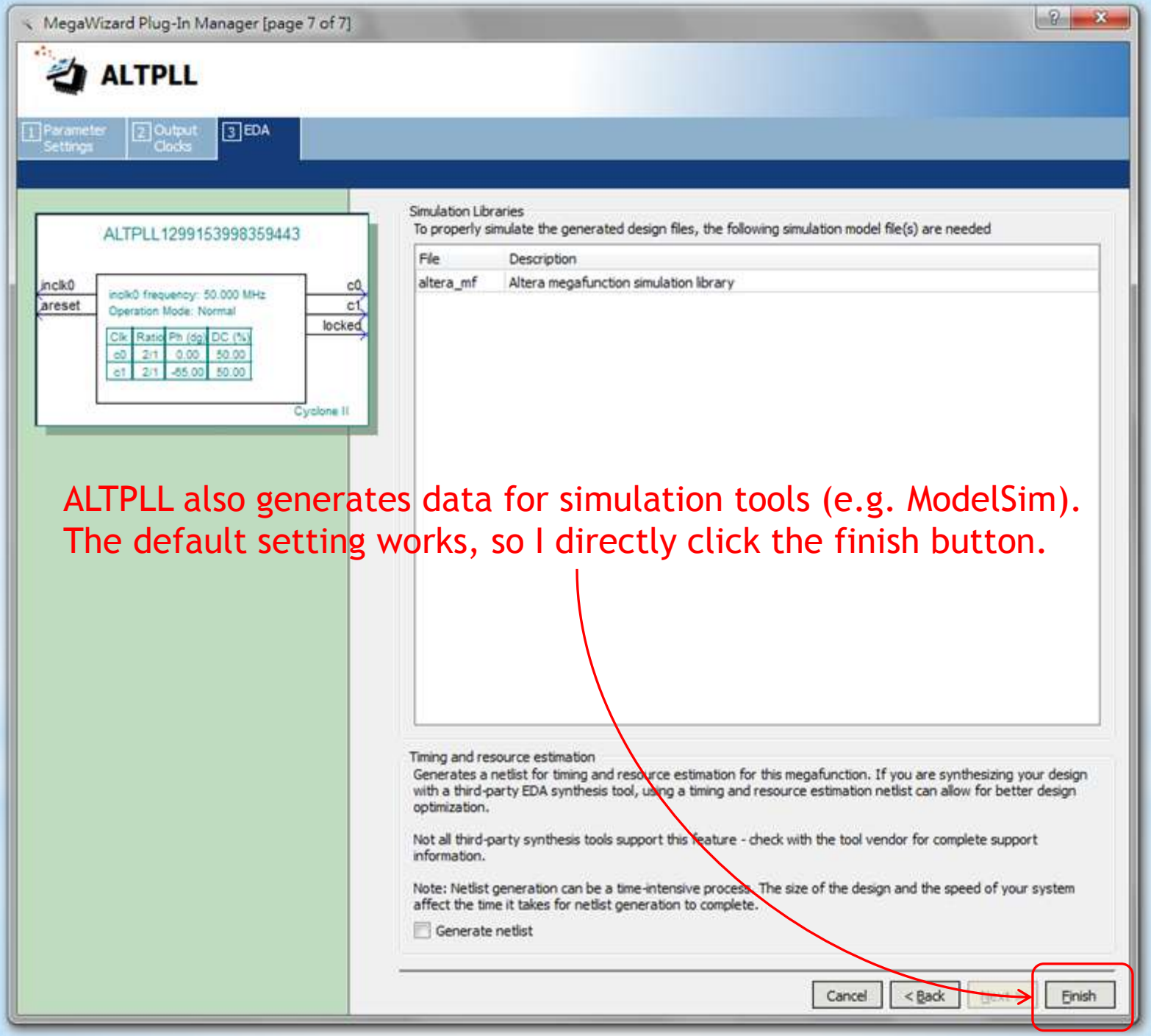

MegaWizard Plug-In Manager [page 2 of 7]
ALTPLL
1 Parameter Settings
2 Output Clocks
3 EDA
General/Modes
Inputs/Lock
Clock switchover
ALTPLL1299153998359443
inclk0
areset
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk
Ratio
Ph (dg)
DC (%)
c0
1/1
0.00
50.00
c0
locked
Cyclone II
Able to implement the requested PLL
Optional Inputs
Create an 'pllena' input to selectively enable the PLL
Create an 'areset' input to asynchronously reset the PLL
Create an 'pfdena' input to selectively enable the phase/frequency detector
Lock Output
Create 'locked' output
Enable self-reset on loss lock
Hold 'locked' output low for
1048575
cycles after the PLL initializes
Advanced Parameters
Using these parameters is recommended for advanced users only
Create output file(s) using the 'Advanced' PLL parameters
- Configurations with output clock(s) that use cascade counters are not supported
Avalon Bus connectivity
Use a separate clock input for Avalon bus connections
In Inputs/Lock page, the default setting works, so I directly click the next button.
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 3 of 7]
ALTPLL
1 Parameter Settings
2 Output Clocks
3 EDA
General/Modes
Inputs/Lock
Clock switchover
ALTPLL1299153998359443
inclk0
areset
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk
Ratio
Ph (dg)
DC (%)
c0
1/1
0.00
50.00
c0
locked
Cyclone II
Able to implement the requested PLL
Perform input clock switch when the PLL has lost lock
Clock Switchover
Create an 'inclk1' input for a second input clock
What is the frequency of the 'inclk1' input?
100.000
MHz
Create a 'clkswitch' input to manually select between the input clocks
(The 'clkswitch' input will behave as an input clock selection control input)
Allow PLL to automatically control the switching between input clocks
(The 'clkswitch' input will behave as a manual override control input)
Input Clock Switch
Perform input clock switch when the primary clock goes bad
Create a 'clkswitch' input to dynamically control the switching between input clocks
Perform the input clock switchover after
1
input clock cycles
Create an 'activeclock' output to indicate the input clock being used
(0 inclk0 is being used/ 1 inclk1 is being used)
Create a 'clkloss' output (indicates that input clock switchover is initiated)
Create a 'clkbad' output for each input clock
(0 input clock is toggling/ 1 input clock is not toggling)
In Clock Switchover, the default setting works, so I directly click the next button.
Cancel
< Back
Next >
Finish

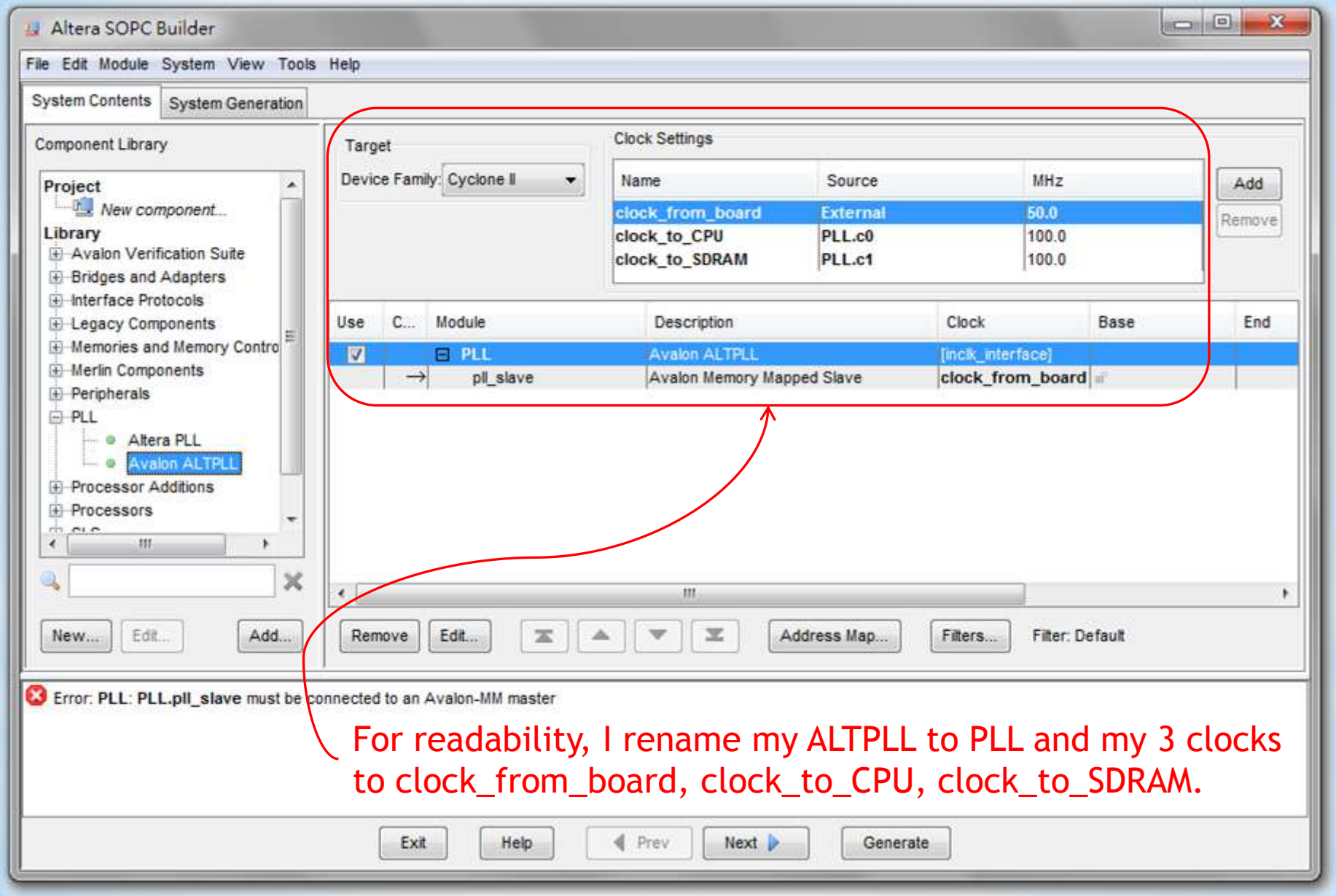

MegaWizard Plug-In Manager [page 4 of 7]
ALTPLL
1 Parameter Settings
2 Output Clocks
3 EDA
clk c0
clk c1
clk c2
ALTPLL1299153998359443
inclk0
areset
c0
locked
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk
Ratio
Ph (dg)
DC (%)
c0
2/1
0.00
50.00
Cyclone II
c0 - Core/External Output Clock
Able to implement the requested PLL
Use this clock
Clock Tap Settings
Requested Settings
Actual Settings
Enter output clock frequency:
100.00000000
MHz
100.000000
Enter output clock parameters:
Clock multiplication factor
2
2
<< Copy
Clock division factor
1
1
Clock phase shift
0.00
deg
0.00
Clock duty cycle (%)
50.00
50.00
Description
Value
Primary clock VCO frequency (MHz)
800.000
Modulus for M counter
16
Modulus for N counter
1
Initial VCO phase cycles for M counter
1
VCO phase tap for M counter
0
Note: The displayed internal settings of the PLL is recommended for use by advanced users only
Per Clock Feasibility Indicators
c0
c1
c2
I want to use c0 as CPU clock, so c0 must be multiplied by 2 with no phase shift.
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 5 of 7]
ALTPLL
1 Parameter Settings
2 Output Clocks
3 EDA
clk c0
clk c1
clk c2
ALTPLL1299153998359443
inclk0
areset
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk | Ratio | Ph (dg) | DC (%)
c0 | 2/1 | 0.00 | 50.00
c1 | 2/1 | -65.00 | 50.00
c0
c1
locked
Cyclone II
c1 - Core/External Output Clock
Able to implement the requested PLL
Use this clock
Clock Tap Settings
Requested Settings
Actual Settings
Enter output clock frequency:
100.00000000
MHz
100.000000
Enter output clock parameters:
Clock multiplication factor
2
<< Copy
2
Clock division factor
1
1
Clock phase shift
-65.00
deg
-65.00
Clock duty cycle (%)
50.00
50.00
Description
Value
Primary clock VCO frequency (MHz)
900.000
Modulus for M counter
18
Modulus for N counter
1
Initial VCO phase cycles for M counter
2
VCO phase tap for M counter
5
Note: The displayed internal settings of the PLL is recommended for use by advanced users only
Per Clock Feasibility Indicators
c0 c1 c2
I want to use c1 as SDRAM clock,
so c1 must be multiplied by 2
with -65 degree phase shift.
Cancel
< Back
Next >
Finish

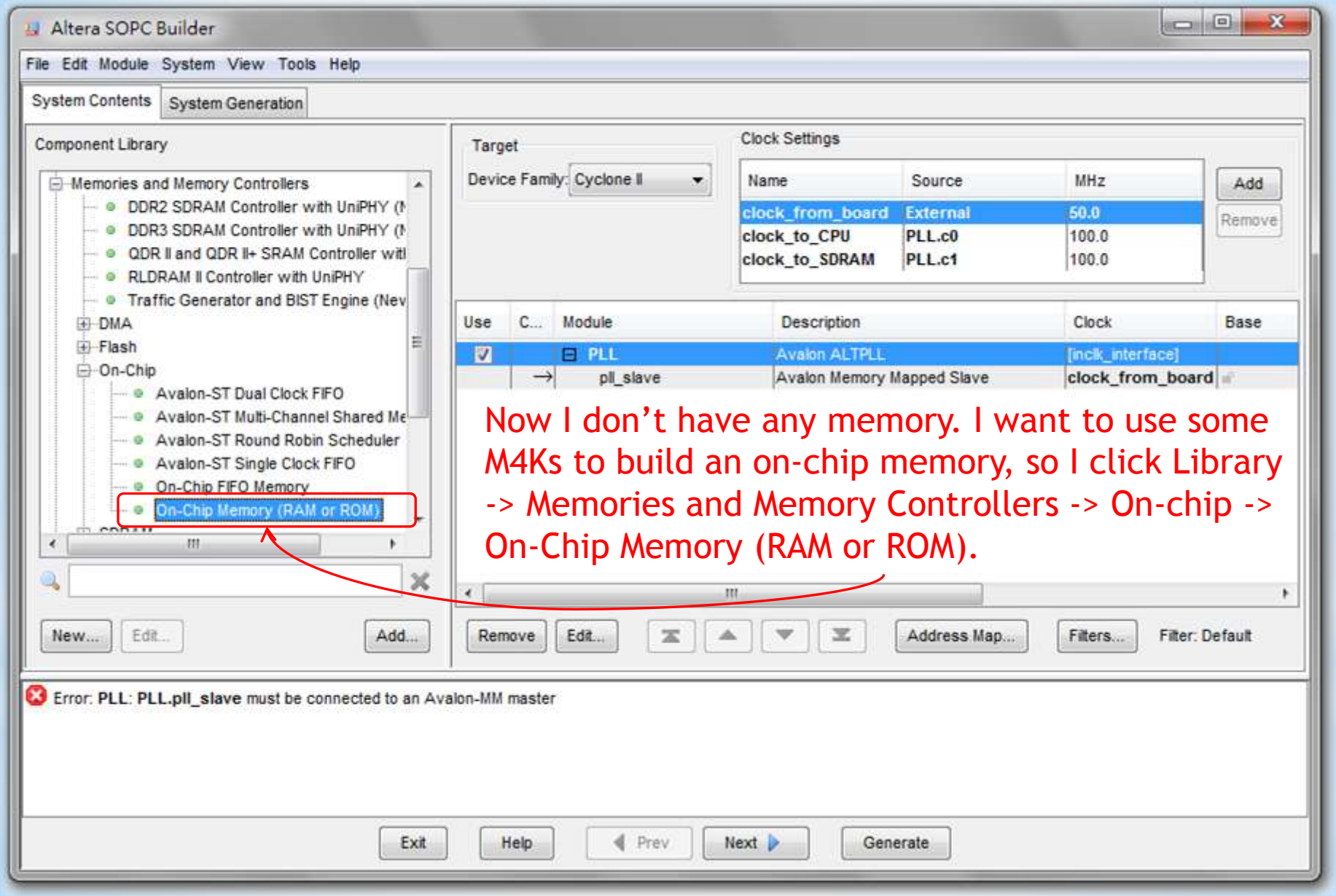

MegaWizard Plug-In Manager [page 6 of 7]
ALTPLL
1 Parameter Settings
2 Output Clocks
3 EDA
clk c0
clk c1
clk c2
ALTPLL1299153998359443
inclk0
areset
c0
c1
locked
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk | Ratio | Ph (dg) | DC (%)
c0 | 2/1 | 0.00 | 50.00
c1 | 2/1 | -65.00 | 50.00
Cyclone II
c2 - Core/External Output Clock
Able to implement the requested PLL
Use this clock
Clock Tap Settings
Requested Settings
Actual Settings
Enter output clock frequency:
100.00000000
MHz
Invalid
Enter output clock parameters:
Clock multiplication factor
1
Invalid
<< Copy
Clock division factor
1
Invalid
Clock phase shift
0.00
deg
Invalid
Clock duty cycle (%)
50.00
Invalid
Description
Value
Primary clock VCO frequency (MHz)
900.000
Modulus for M counter
18
Modulus for N counter
1
Initial VCO phase cycles for M counter
2
VCO phase tap for M counter
5
Note: The displayed internal settings of the PLL is recommended for use by advanced users only
Per Clock Feasibility Indicators
c0 c1 c2
I don’t need c2, so I directly click the next button.
Cancel
< Back
Next >
Finish

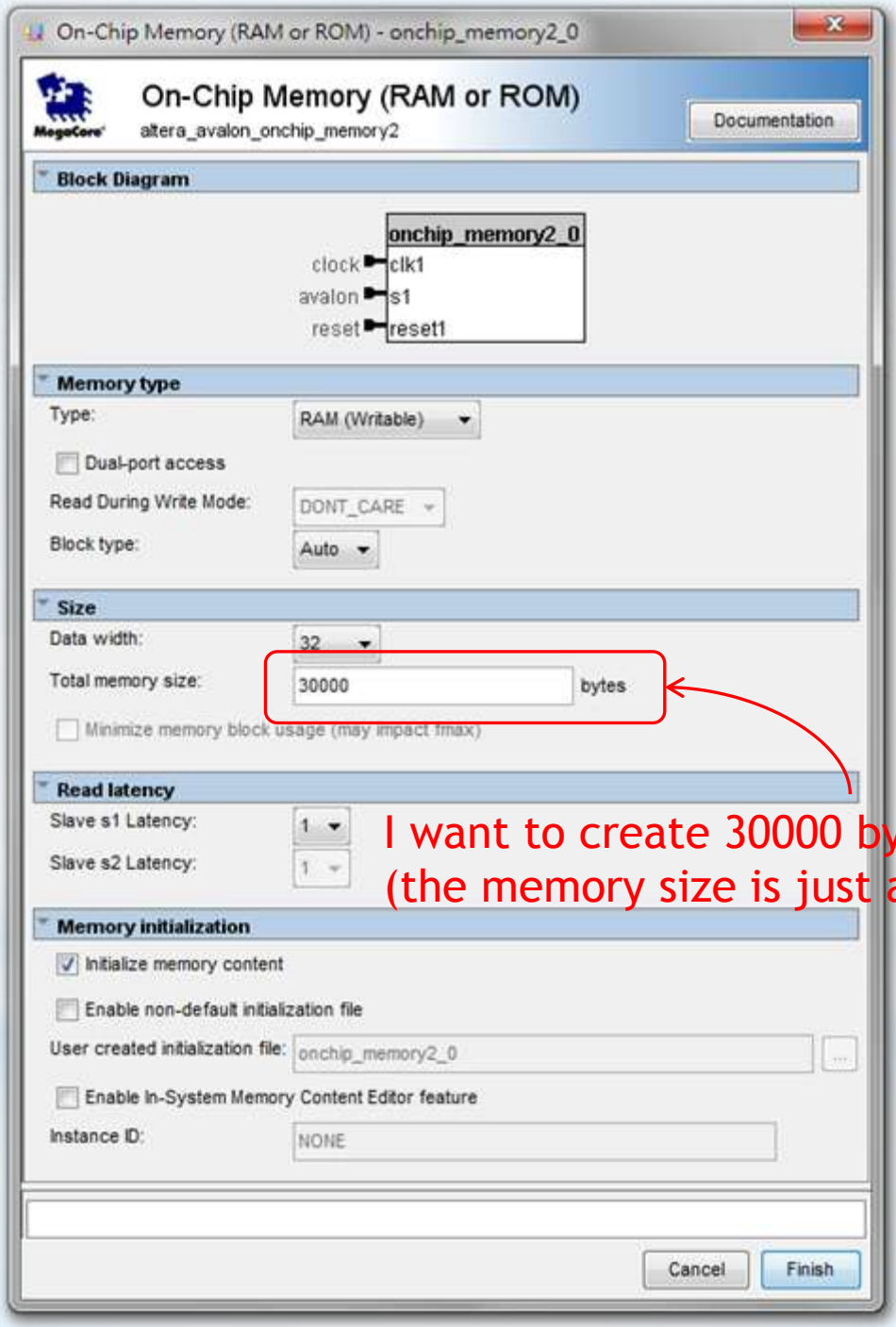


ALTPLL also generates data for simulation tools (e.g. ModelSim).
The default setting works, so I directly click the finish button.

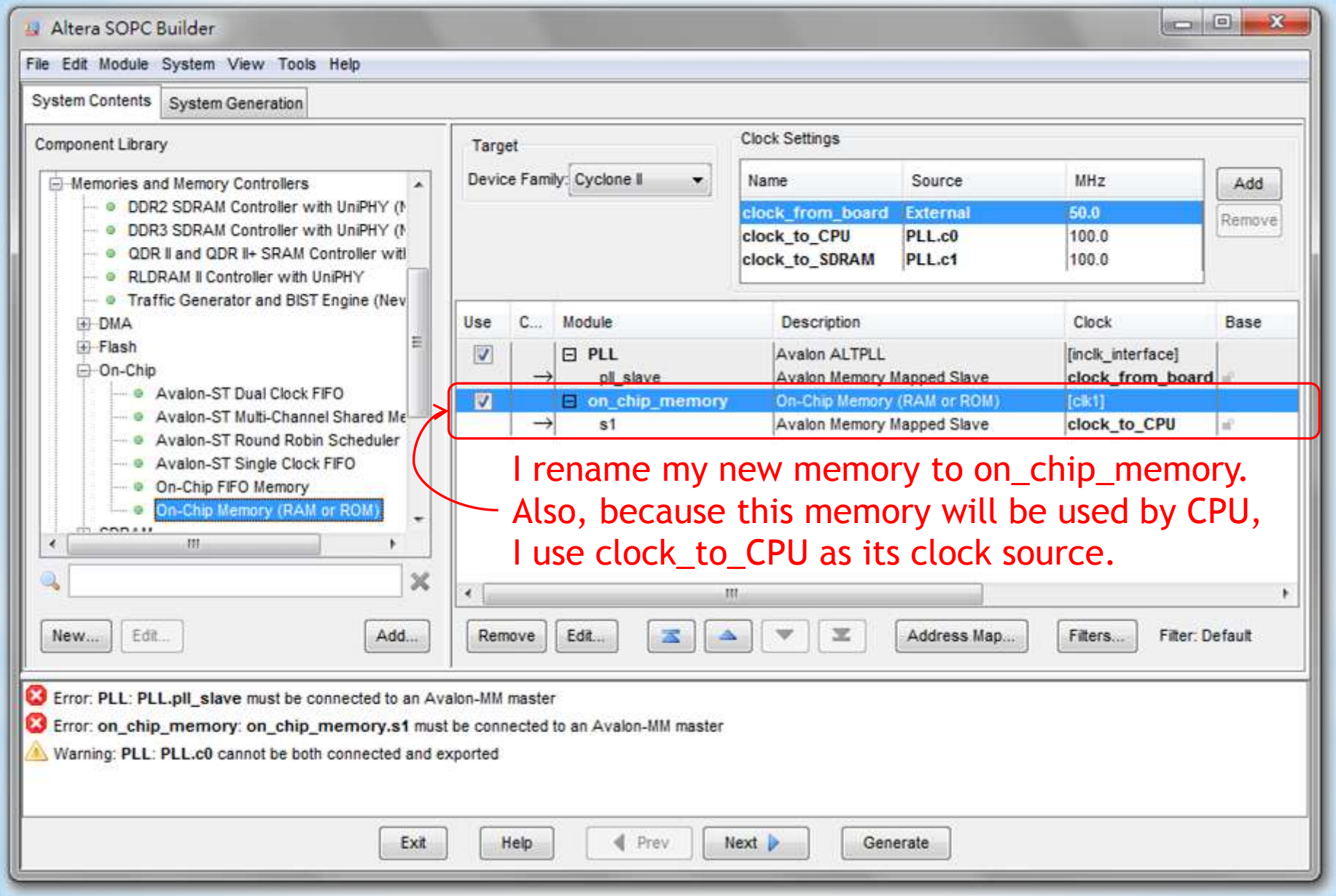


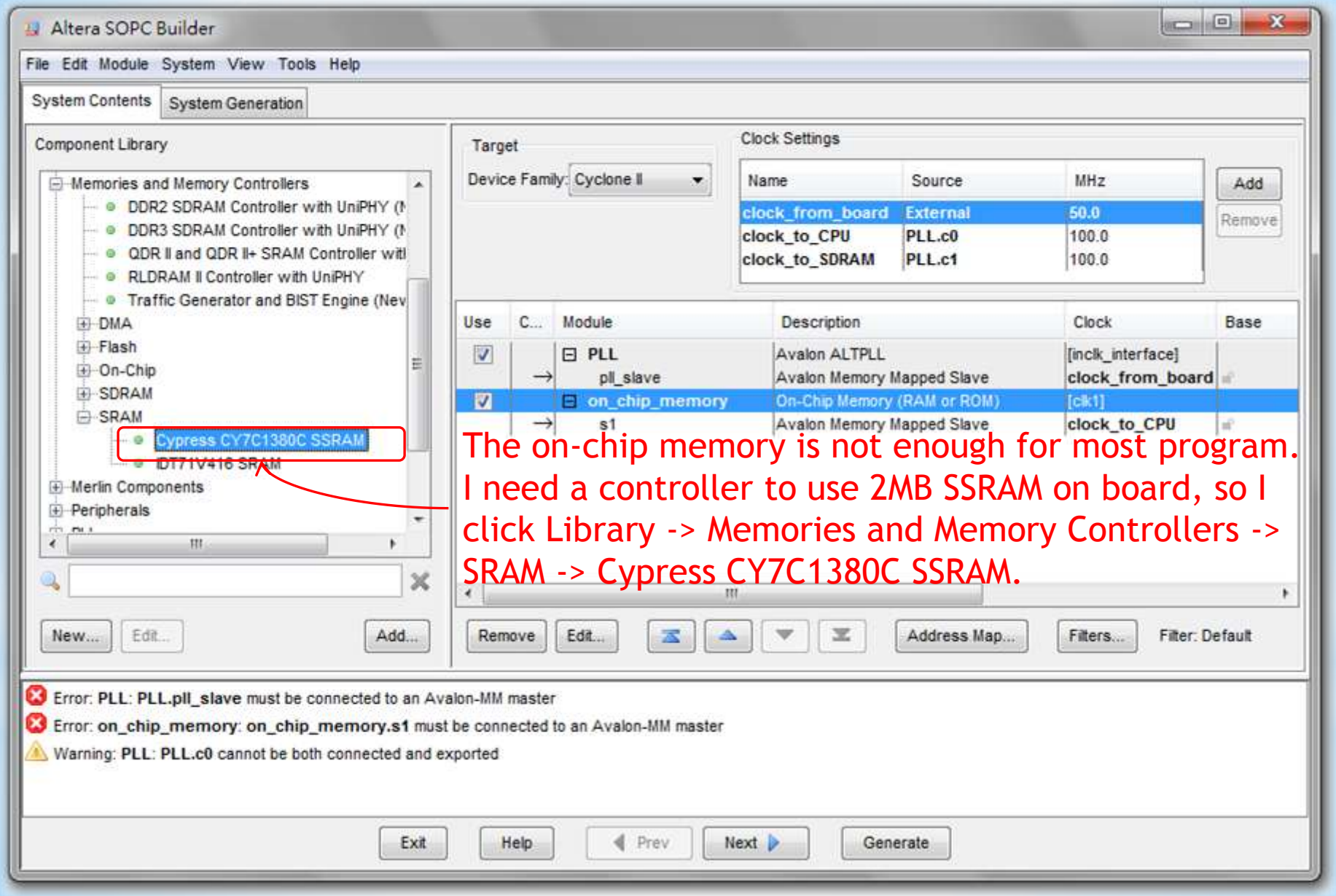
For readability, I rename my ALTPLL to PLL and my 3 clocks to clock_from_board, clock_to_CPU, clock_to_SDRAM.


Altera SOPC Builder
File Edit Module System View Tools Help
System Contents
System Generation
Component Library
Memories and Memory Controllers
DDR2 SDRAM Controller with UniPHY
DDR3 SDRAM Controller with UniPHY
QDR II and QDR II+ SRAM Controller
RLDRAM II Controller with UniPHY
Traffic Generator and BIST Engine
DMA
Flash
On-Chip
Avalon-ST Dual Clock FIFO
Avalon-ST Multi-Channel Shared
Avalon-ST Round Robin Scheduler
Avalon-ST Single Clock FIFO
On-Chip FIFO Memory
On-Chip Memory (RAM or ROM)
New...
Edit...
Add...
Target
Device Family: Cyclone II
Clock Settings
Name
Source
MHz
clock_from_board
External
50.0
clock_to_CPU
PLL.c0
100.0
clock_to_SDRAM
PLL.c1
100.0
Add
Remove
Use
C...
Module
Description
Clock
Base
PLL
Avalon ALTPLL
[inclk_interface]
pll_slave
Avalon Memory Mapped Slave
clock_from_board
Now I don't have any memory. I want to use some M4Ks to build an on-chip memory, so I click Library -> Memories and Memory Controllers -> On-chip -> On-Chip Memory (RAM or ROM).
Remove
Edit...
Address Map...
Filters...
Filter: Default
Error: PLL: PLL.pll_slave must be connected to an Avalon-MM master
Exit
Help
Prev
Next
Generate

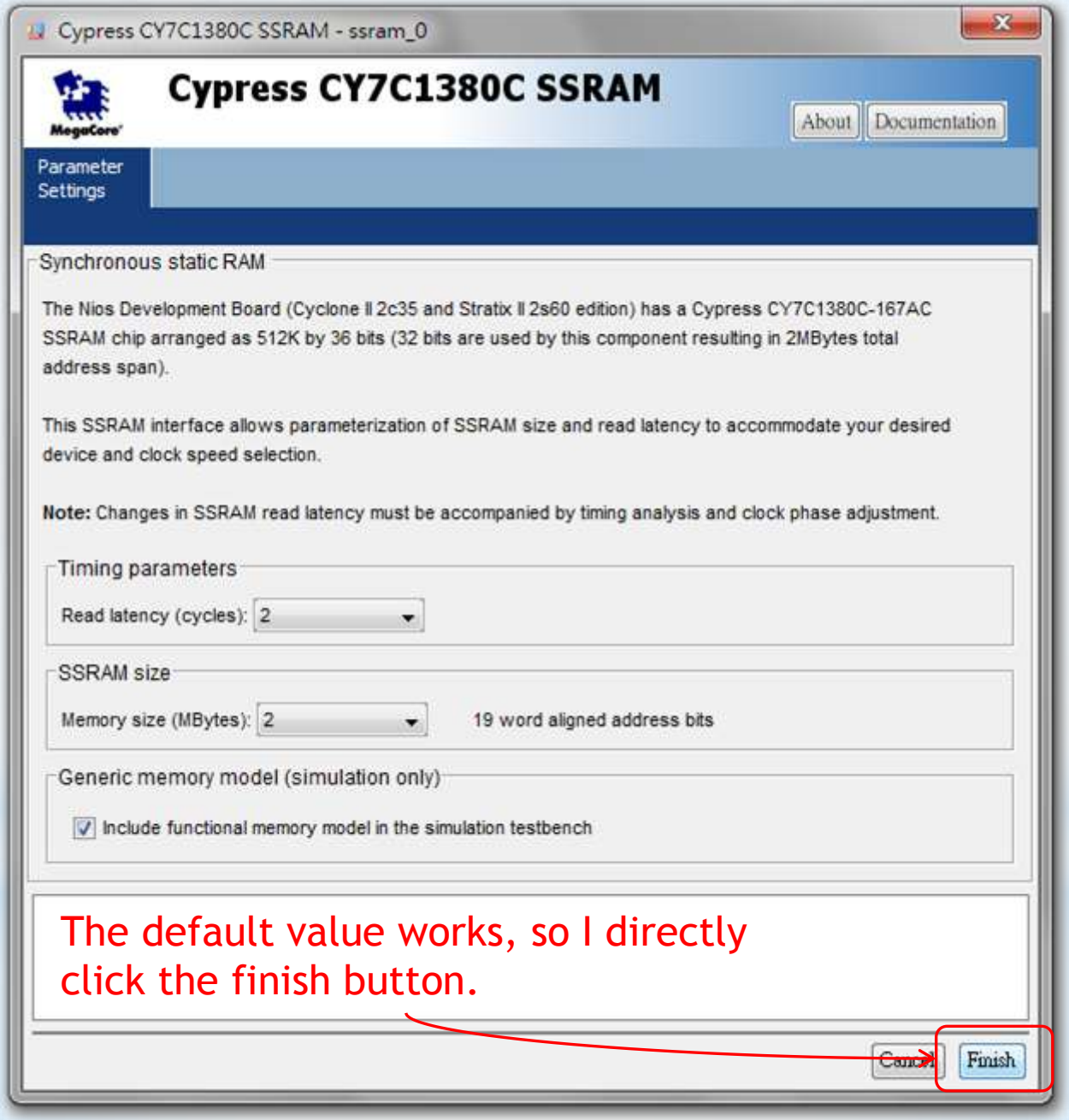


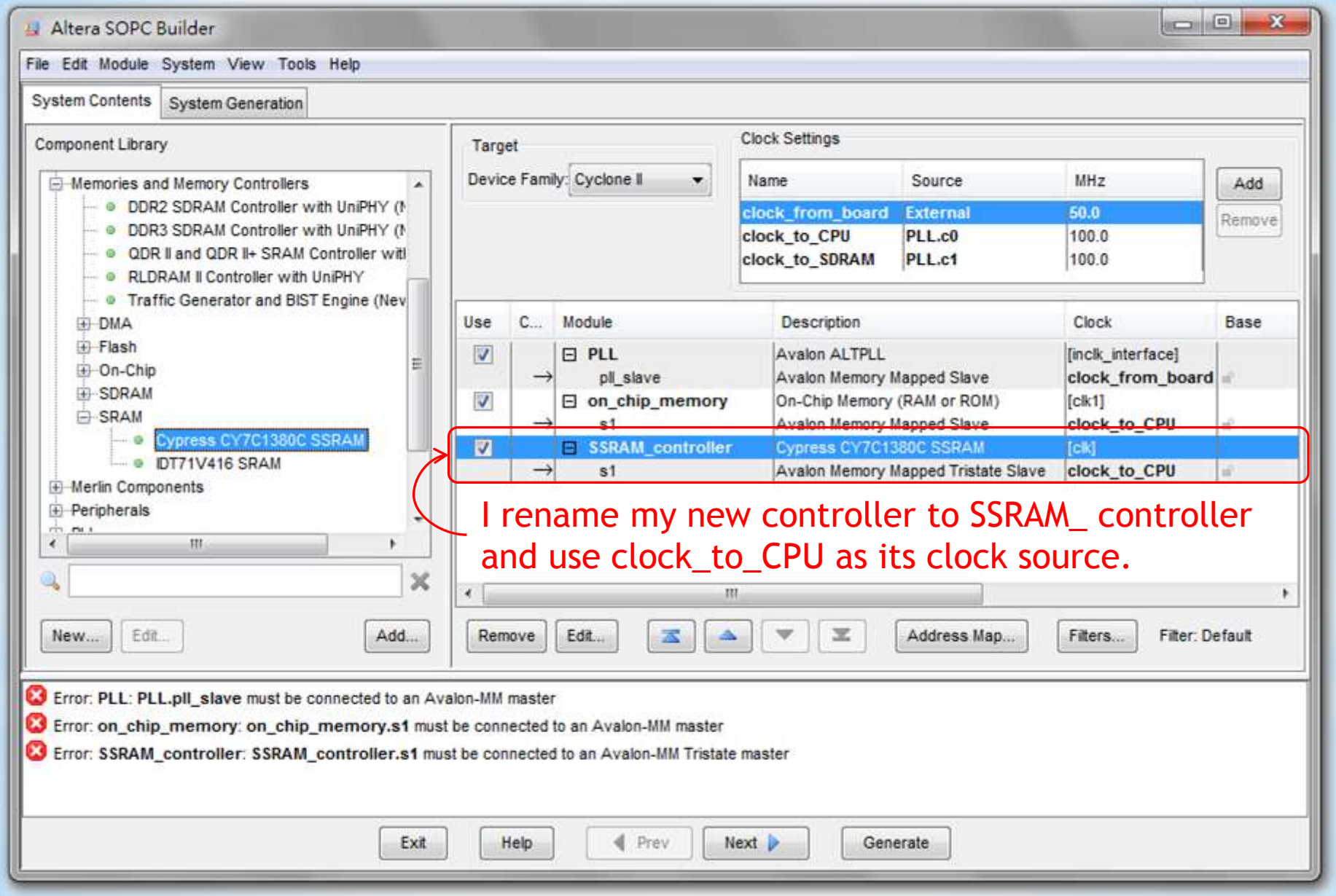
I want to create 30000 bytes on-chip memory.
(the memory size is just an example number)

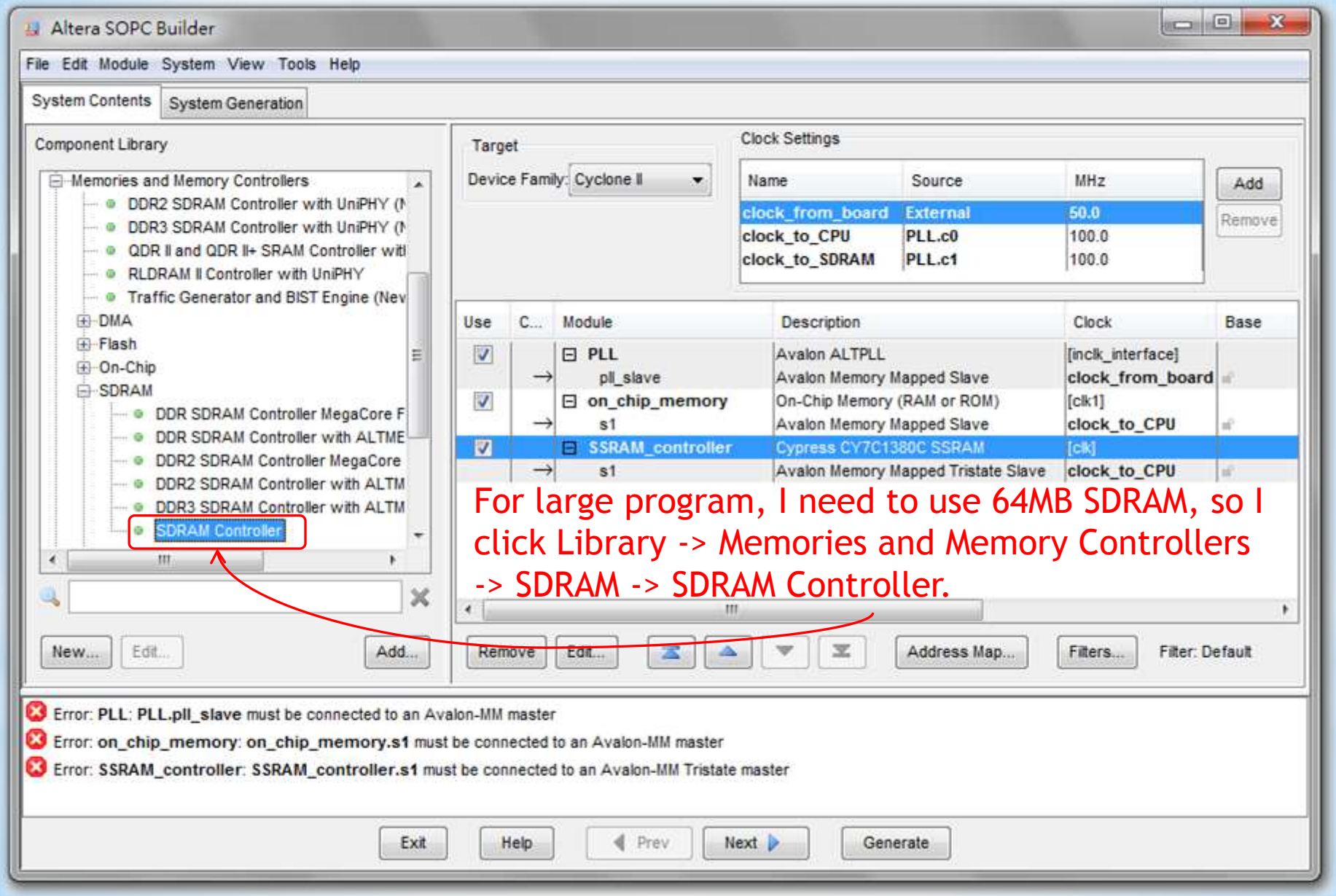

Altera SOPC Builder
File Edit Module System View Tools Help
System Contents
System Generation
Component Library
Memories and Memory Controllers
DDR2 SDRAM Controller with UniPHY
DDR3 SDRAM Controller with UniPHY
QDR II and QDR II+ SRAM Controller with
RLDRAM II Controller with UniPHY
Traffic Generator and BIST Engine
DMA
Flash
On-Chip
Avalon-ST Dual Clock FIFO
Avalon-ST Multi-Channel Shared Me
Avalon-ST Round Robin Scheduler
Avalon-ST Single Clock FIFO
On-Chip FIFO Memory
On-Chip Memory (RAM or ROM)
New...
Edit...
Add...
Target
Device Family: Cyclone II
Clock Settings
Name
Source
MHz
clock_from_board
External
50.0
clock_to_CPU
PLL.c0
100.0
clock_to_SDRAM
PLL.c1
100.0
Add
Remove
Use
C...
Module
Description
Clock
Base
PLL
Avalon ALTPLL
[inclk_interface]
pll_slave
Avalon Memory Mapped Slave
clock_from_board
on_chip_memory
On-Chip Memory (RAM or ROM)
[clk1]
s1
Avalon Memory Mapped Slave
clock_to_CPU
Remove
Edit...
Address Map...
Filters...
Filter: Default
I rename my new memory to on_chip_memory.
Also, because this memory will be used by CPU,
I use clock_to_CPU as its clock source.
Error: PLL: PLL.pll_slave must be connected to an Avalon-MM master
Error: on_chip_memory: on_chip_memory.s1 must be connected to an Avalon-MM master
Warning: PLL: PLL.c0 cannot be both connected and exported
Exit
Help
Prev
Next
Generate



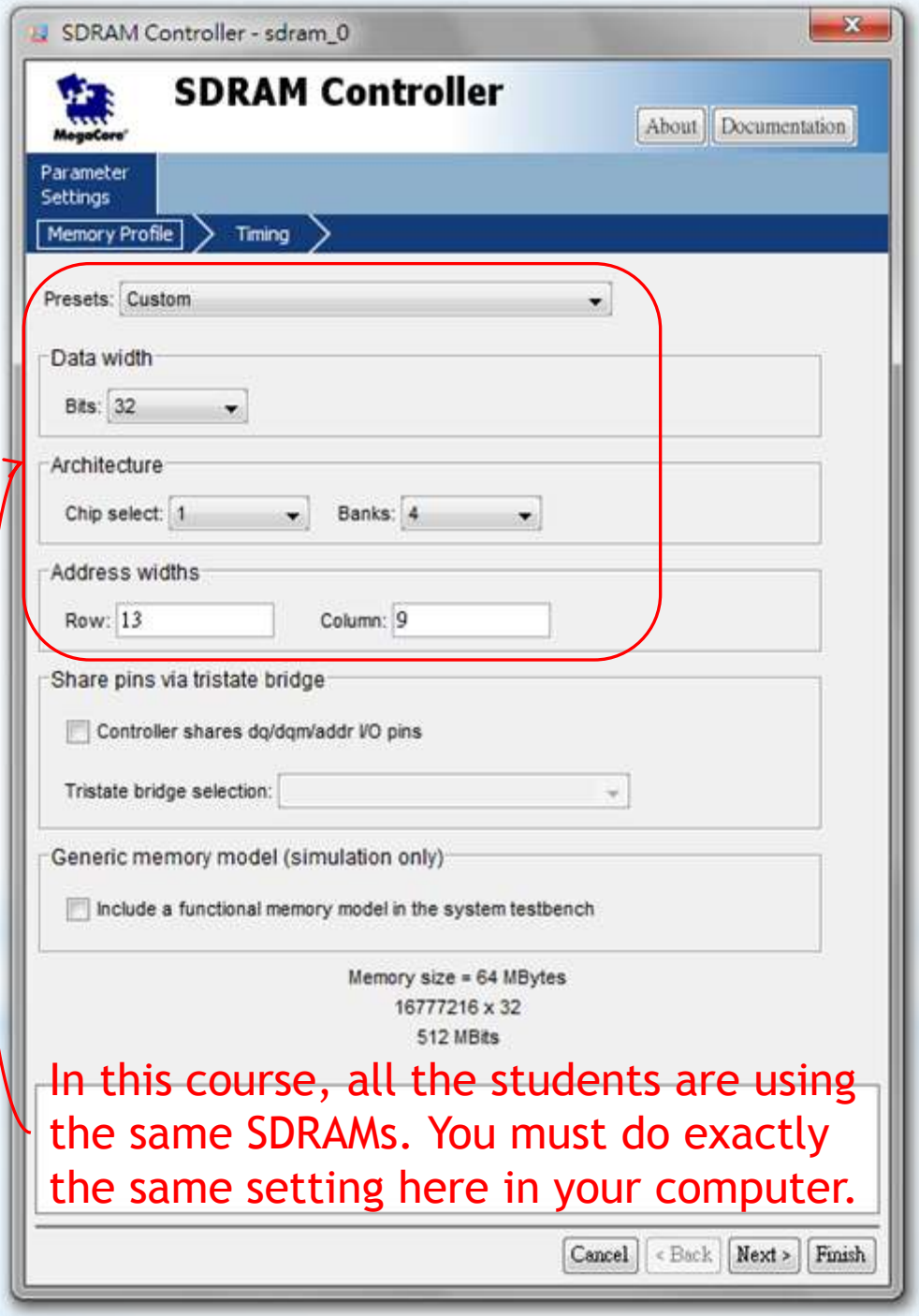
The on-chip memory is not enough for most program. I need a controller to use 2MB SSRAM on board, so I click Library -> Memories and Memory Controllers -> SRAM -> Cypress CY7C1380C SSRAM.

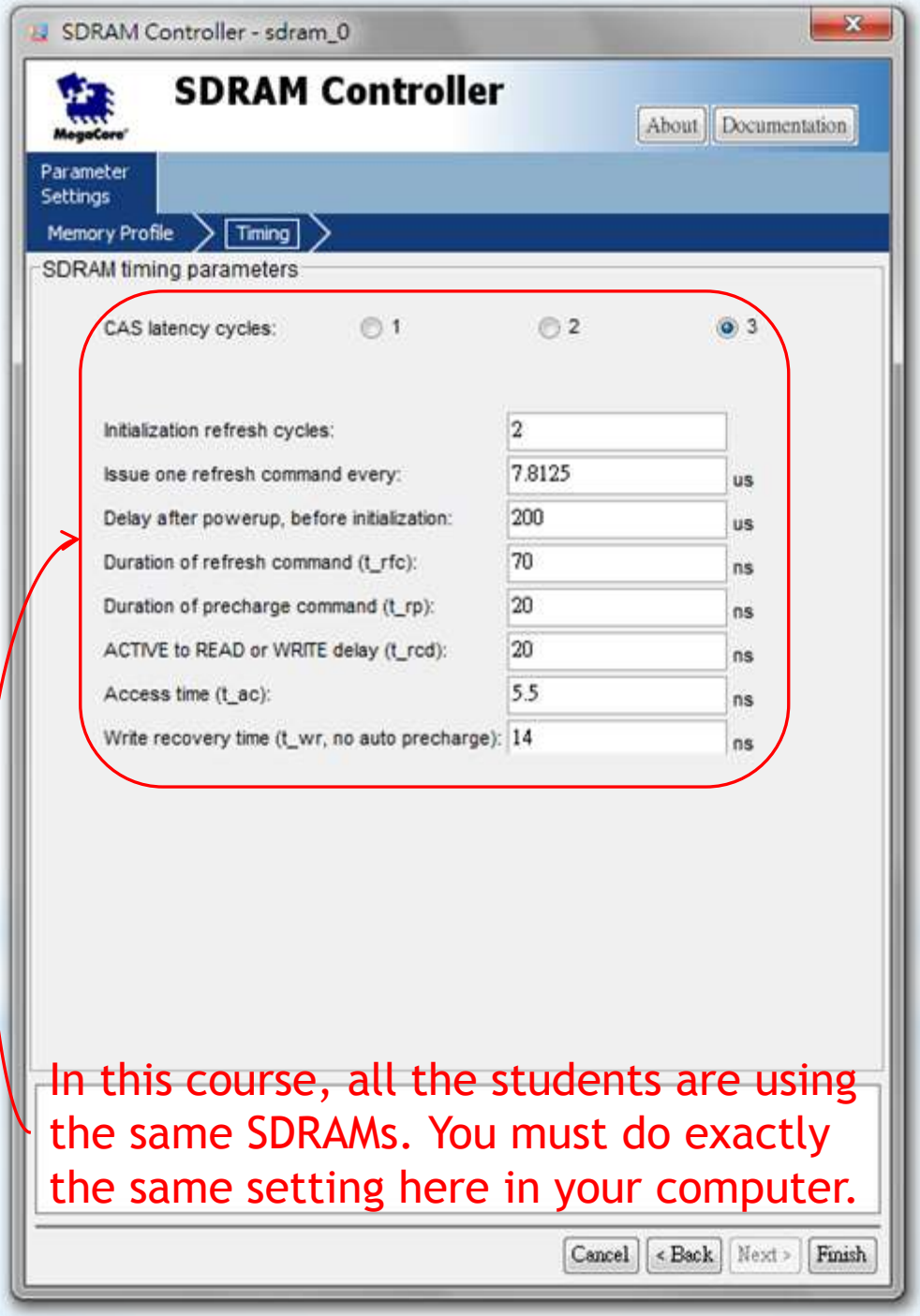

Cypress CY7C1380C SSRAM - ssram_0
Cypress CY7C1380C SSRAM
About
Documentation
Parameter Settings
Synchronous static RAM
The Nios Development Board (Cyclone II 2c35 and Stratix II 2s60 edition) has a Cypress CY7C1380C-167AC SSRAM chip arranged as 512K by 36 bits (32 bits are used by this component resulting in 2MBytes total address span).
This SSRAM interface allows parameterization of SSRAM size and read latency to accommodate your desired device and clock speed selection.
Note: Changes in SSRAM read latency must be accompanied by timing analysis and clock phase adjustment.
Timing parameters
Read latency (cycles): 2
SSRAM size
Memory size (MBytes): 2
19 word aligned address bits
Generic memory model (simulation only)
Include functional memory model in the simulation testbench
The default value works, so I directly click the finish button.
Cancel
Finish


Altera SOPC Builder
File Edit Module System View Tools Help
System Contents
System Generation
Component Library
Memories and Memory Controllers
DDR2 SDRAM Controller with UniPHY (
DDR3 SDRAM Controller with UniPHY (
QDR II and QDR II+ SRAM Controller with
RLDRAM II Controller with UniPHY
Traffic Generator and BIST Engine (Nev
DMA
Flash
On-Chip
SDRAM
SRAM
Cypress CY7C1380C SSRAM
IDT71V416 SRAM
Merlin Components
Peripherals
New...
Edit...
Add...
Target
Device Family: Cyclone II
Clock Settings
Name
Source
MHz
clock_from_board
External
50.0
clock_to_CPU
PLL.c0
100.0
clock_to_SDRAM
PLL.c1
100.0
Add
Remove
Use
C...
Module
Description
Clock
Base
PLL
Avalon ALTPLL
[inclk_interface]
pll_slave
Avalon Memory Mapped Slave
clock_from_board
on_chip_memory
On-Chip Memory (RAM or ROM)
[clk1]
s1
Avalon Memory Mapped Slave
clock_to_CPU
SSRAM_controller
Cypress CY7C1380C SSRAM
[clk]
s1
Avalon Memory Mapped Tristate Slave
clock_to_CPU
I rename my new controller to SSRAM_ controller
and use clock_to_CPU as its clock source.
Remove
Edit...
Address Map...
Filters...
Filter: Default
Error: PLL: PLL.pll_slave must be connected to an Avalon-MM master
Error: on_chip_memory: on_chip_memory.s1 must be connected to an Avalon-MM master
Error: SSRAM_controller: SSRAM_controller.s1 must be connected to an Avalon-MM Tristate master
Exit
Help
Prev
Next
Generate

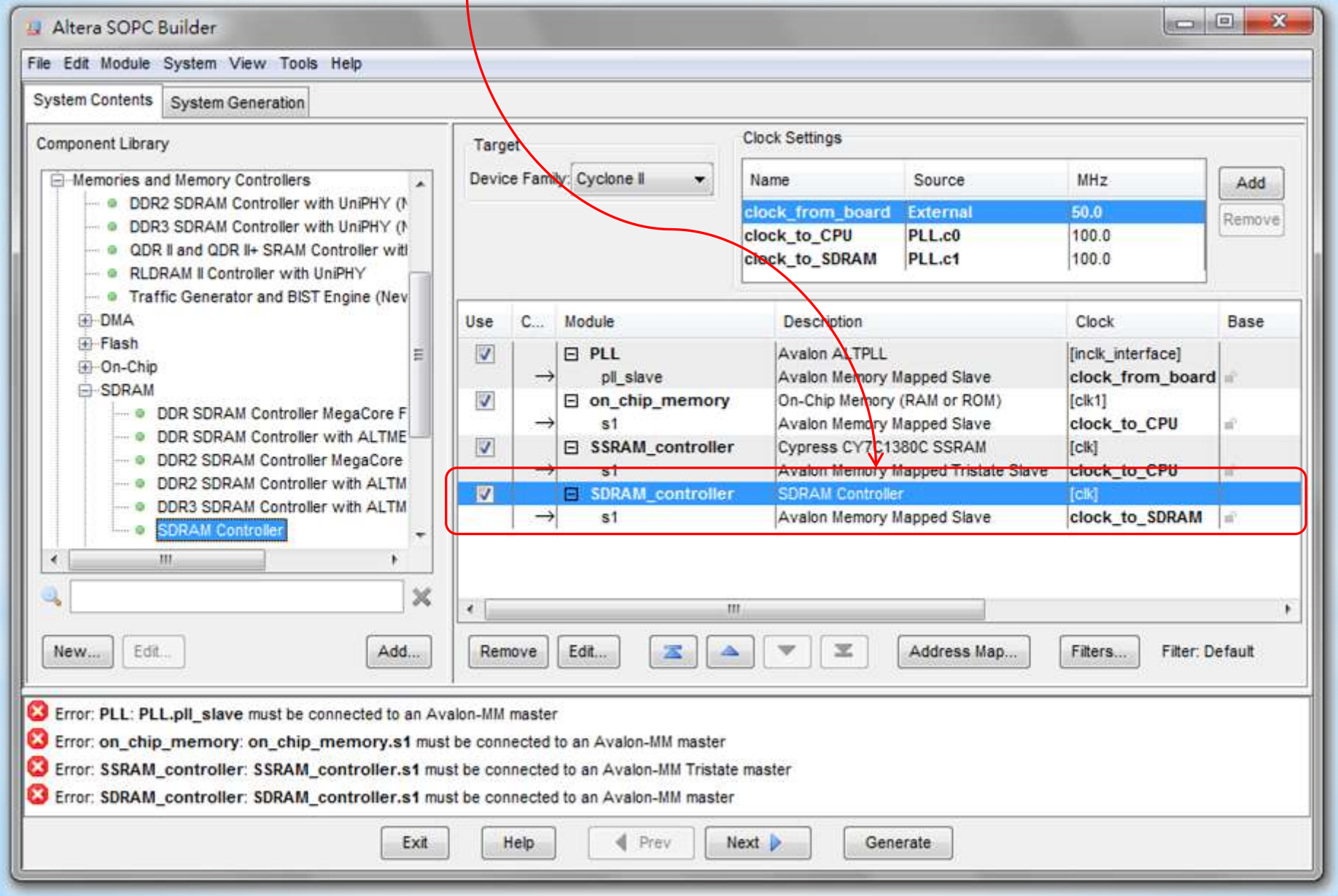


For large program, I need to use 64MB SDRAM, so I click Library -> Memories and Memory Controllers -> SDRAM -> SDRAM Controller.

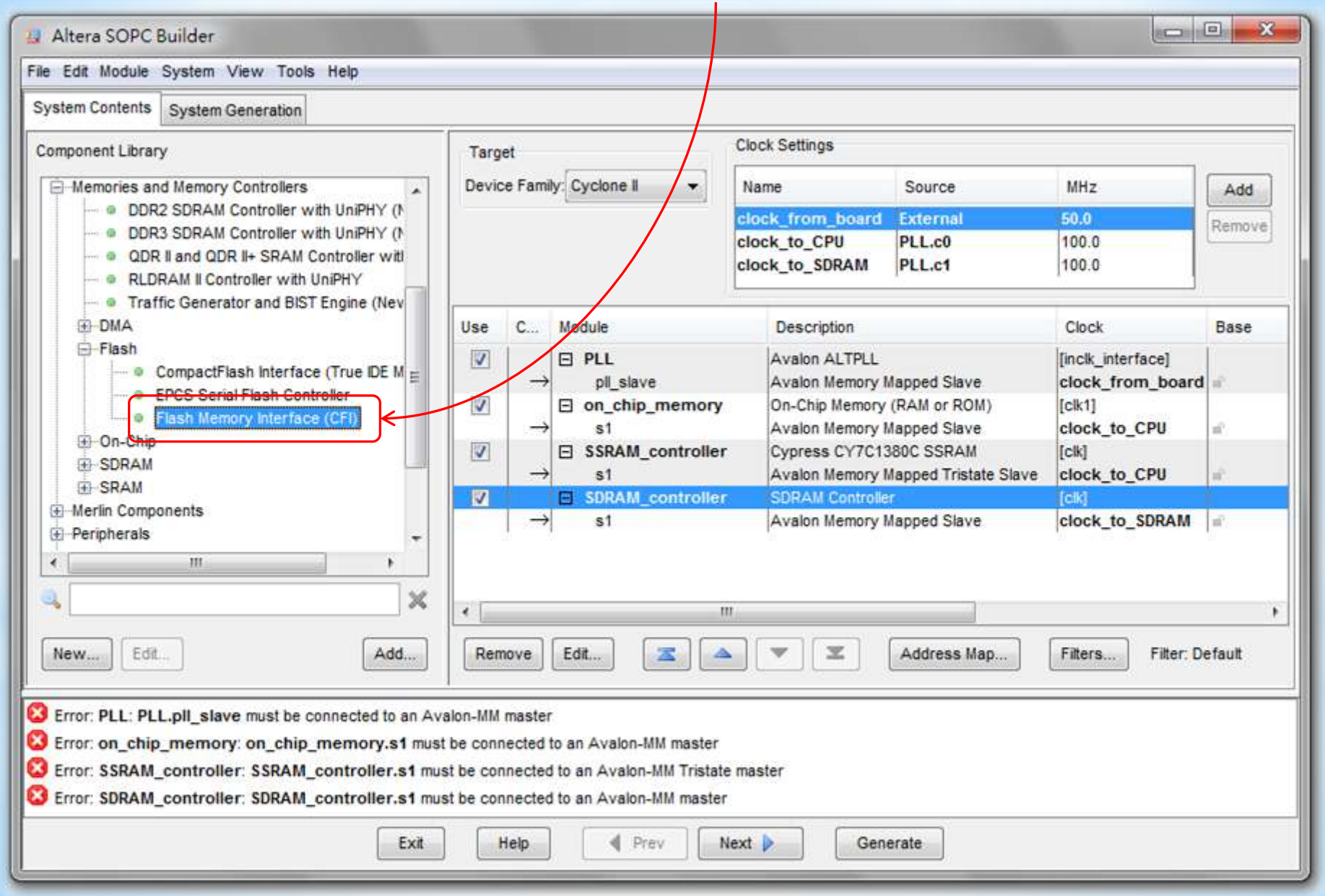

SDRAM Controller - sdram_0
SDRAM Controller
About
Documentation
Parameter Settings
Memory Profile
Timing
Presets: Custom
Data width
Bits: 32
Architecture
Chip select: 1
Banks: 4
Address widths
Row: 13
Column: 9
Share pins via tristate bridge
Controller shares dq/dqm/addr I/O pins
Tristate bridge selection:
Generic memory model (simulation only)
Include a functional memory model in the system testbench
Memory size = 64 MBytes
16777216 x 32
512 MBits
In this course, all the students are using the same SDRAMs. You must do exactly the same setting here in your computer.
Cancel
< Back
Next >
Finish


SDRAM Controller - sdram_0
SDRAM Controller
About
Documentation
Parameter Settings
Memory Profile
Timing
SDRAM timing parameters
CAS latency cycles: 1 2 3
Initialization refresh cycles: 2
Issue one refresh command every: 7.8125 us
Delay after powerup, before initialization: 200 us
Duration of refresh command (t_rfc): 70 ns
Duration of precharge command (t_rp): 20 ns
ACTIVE to READ or WRITE delay (t_rcd): 20 ns
Access time (t_ac): 5.5 ns
Write recovery time (t_wr, no auto precharge): 14 ns
In this course, all the students are using the same SDRAMs. You must do exactly the same setting here in your computer.
Cancel
< Back
Next >
Finish

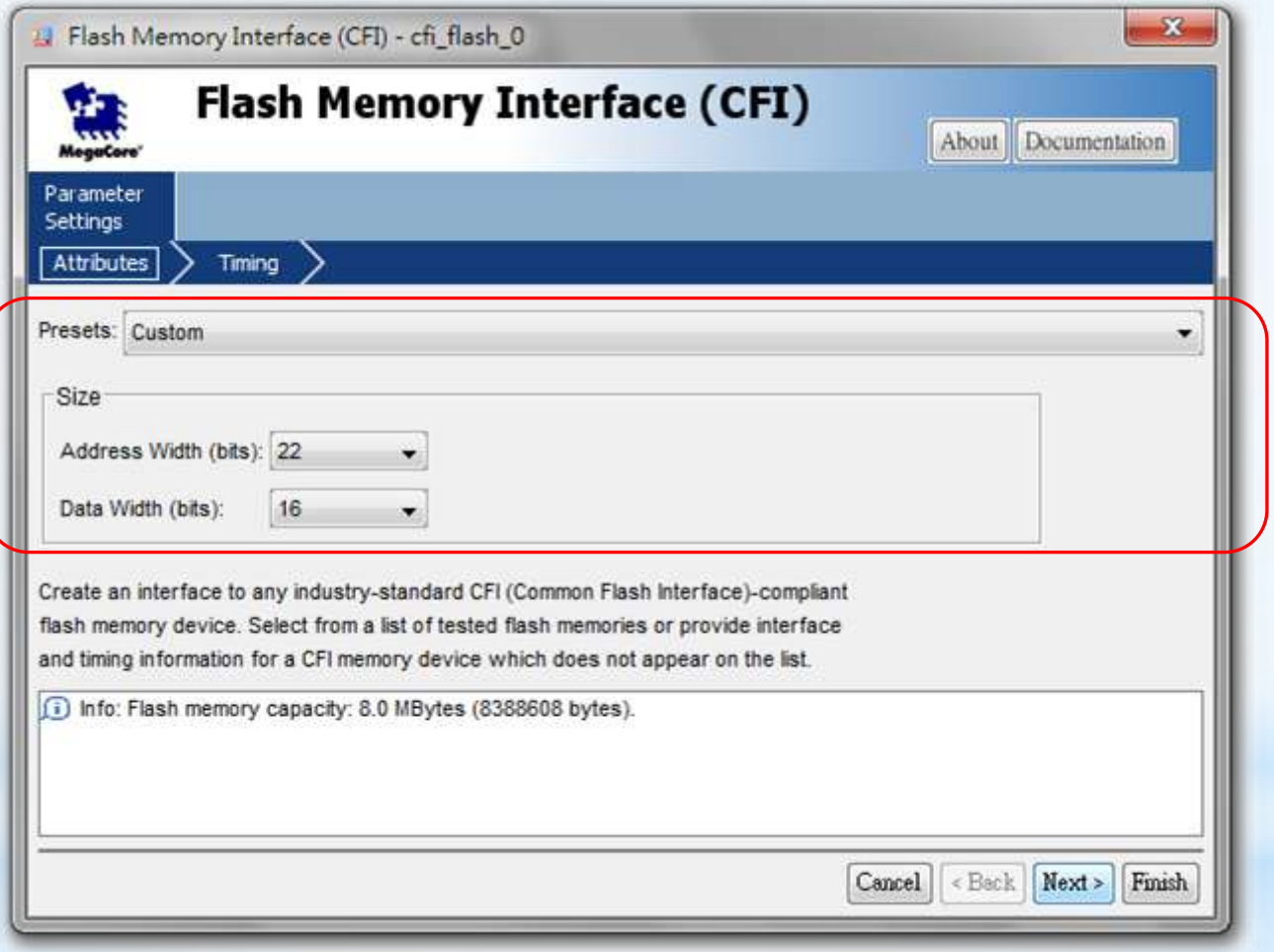
I rename my new controller to SDRAM_controller.
Remember to use clock_to_SDRAM, not clock_to_CPU.



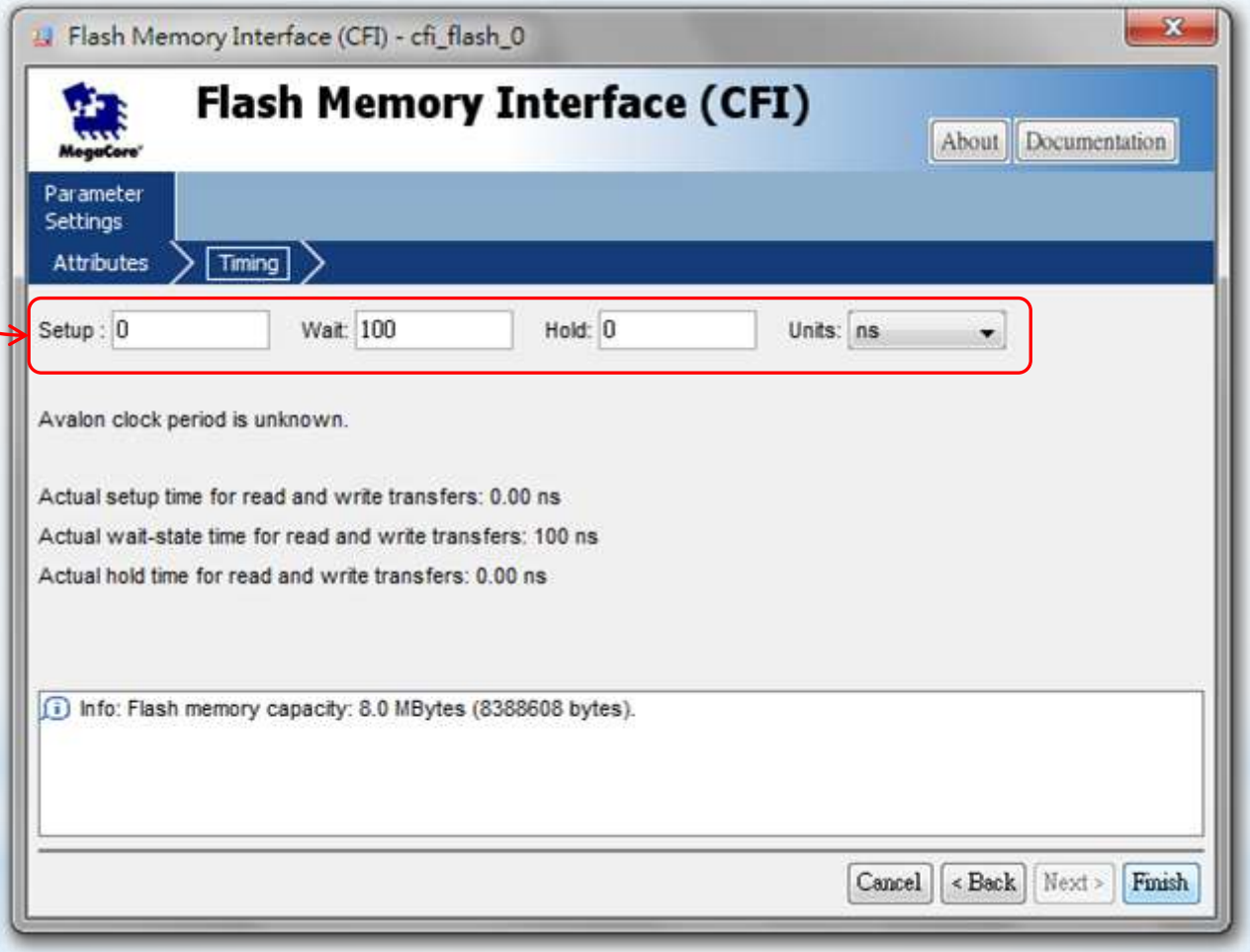
I want to use NVM, so I click Library -> Memories and Memory Controllers -> Flash -> Flash Memory Interface (CFI).



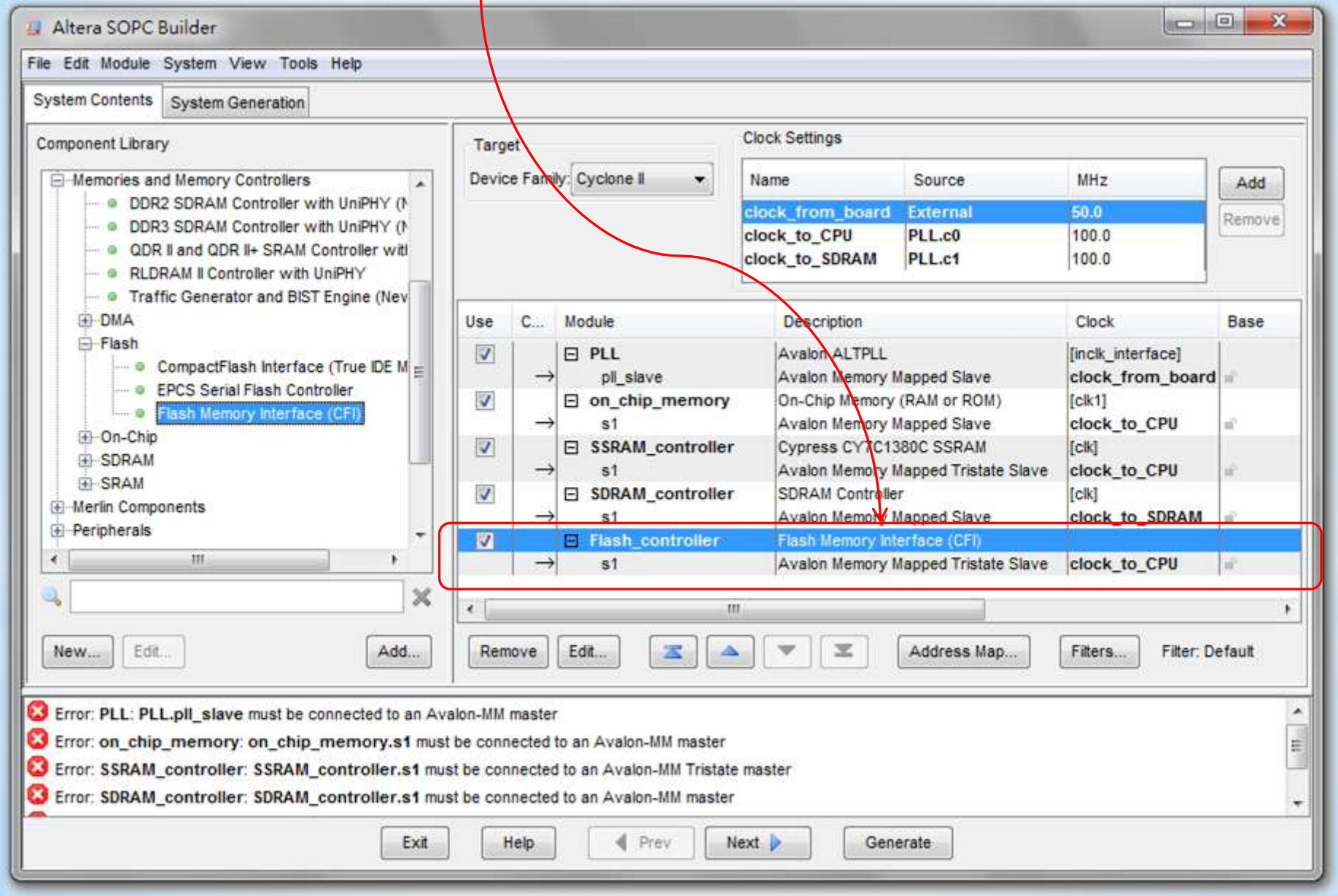
In this course, all the students are using the same Flashes.
You must do exactly the same setting here in your computer.



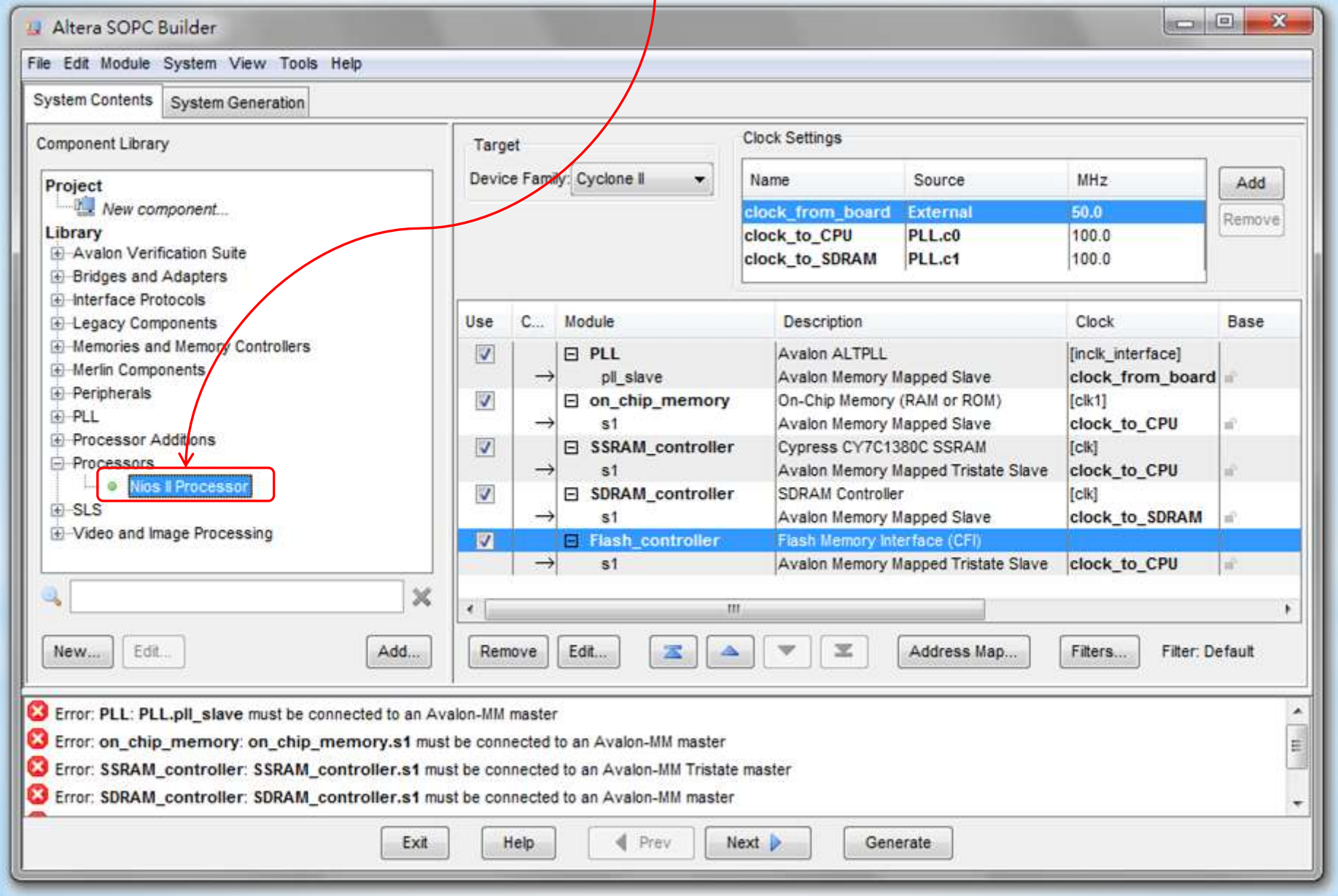
In this course, all the students are using the same Flashes.
You must do exactly the same setting here in your computer.

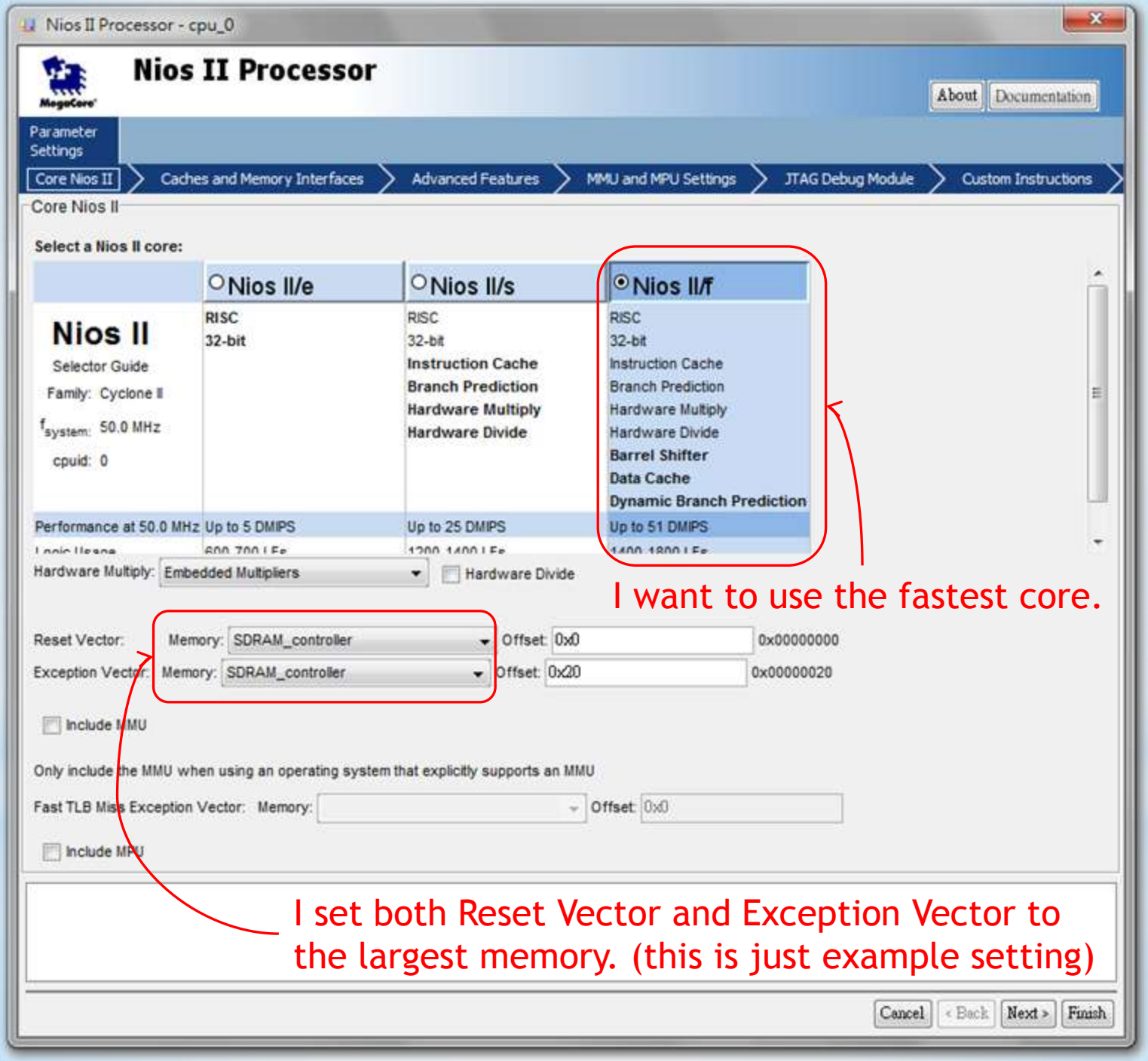


I rename my new controller to Flash_ controller and use clock_to_CPU as its clock source.

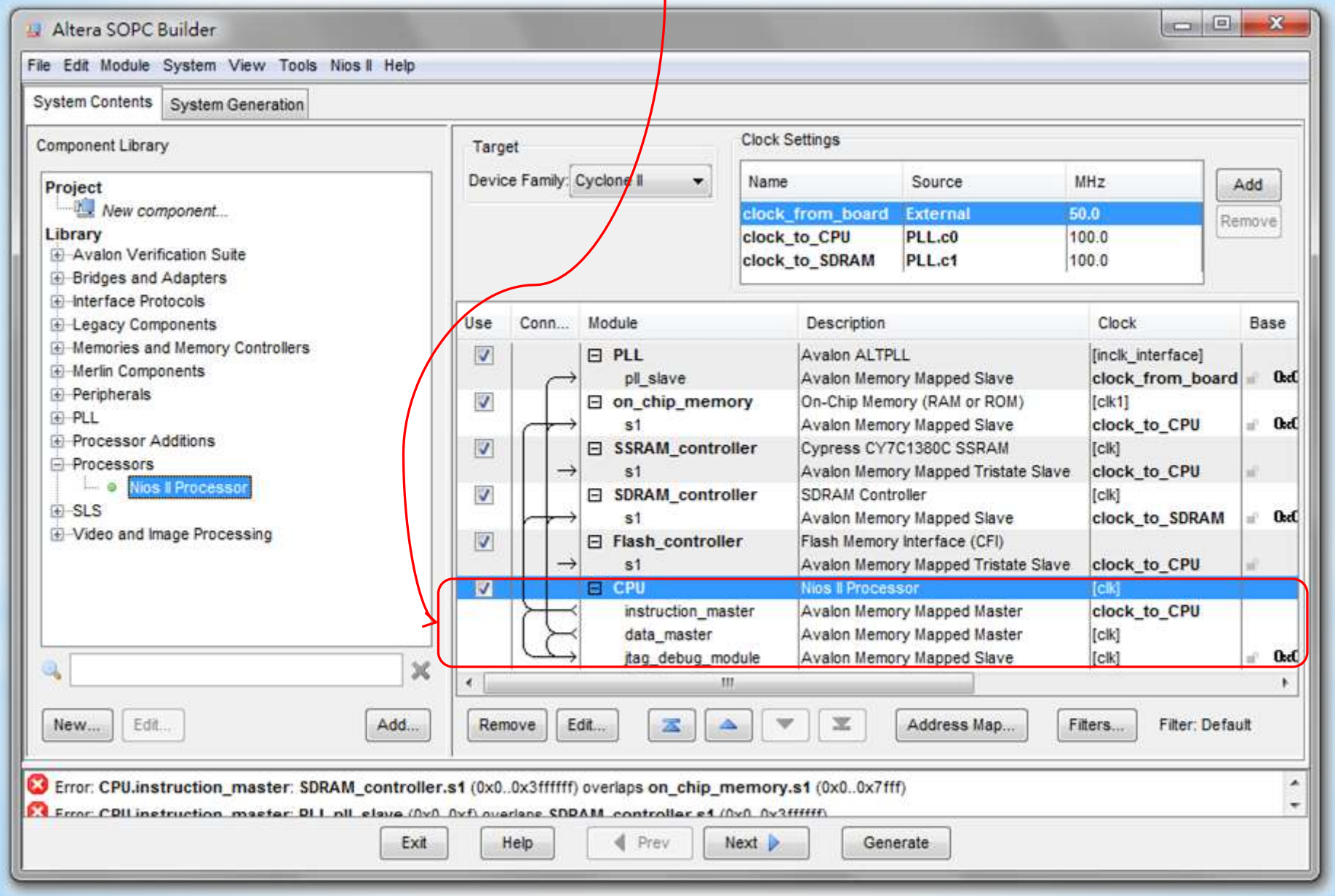


Now I have prepared all the memories for Nios II CPU to run my C/C++program.
It's Nios II CPU show time! Simply click Library -> Processors -> Nios II Processor.

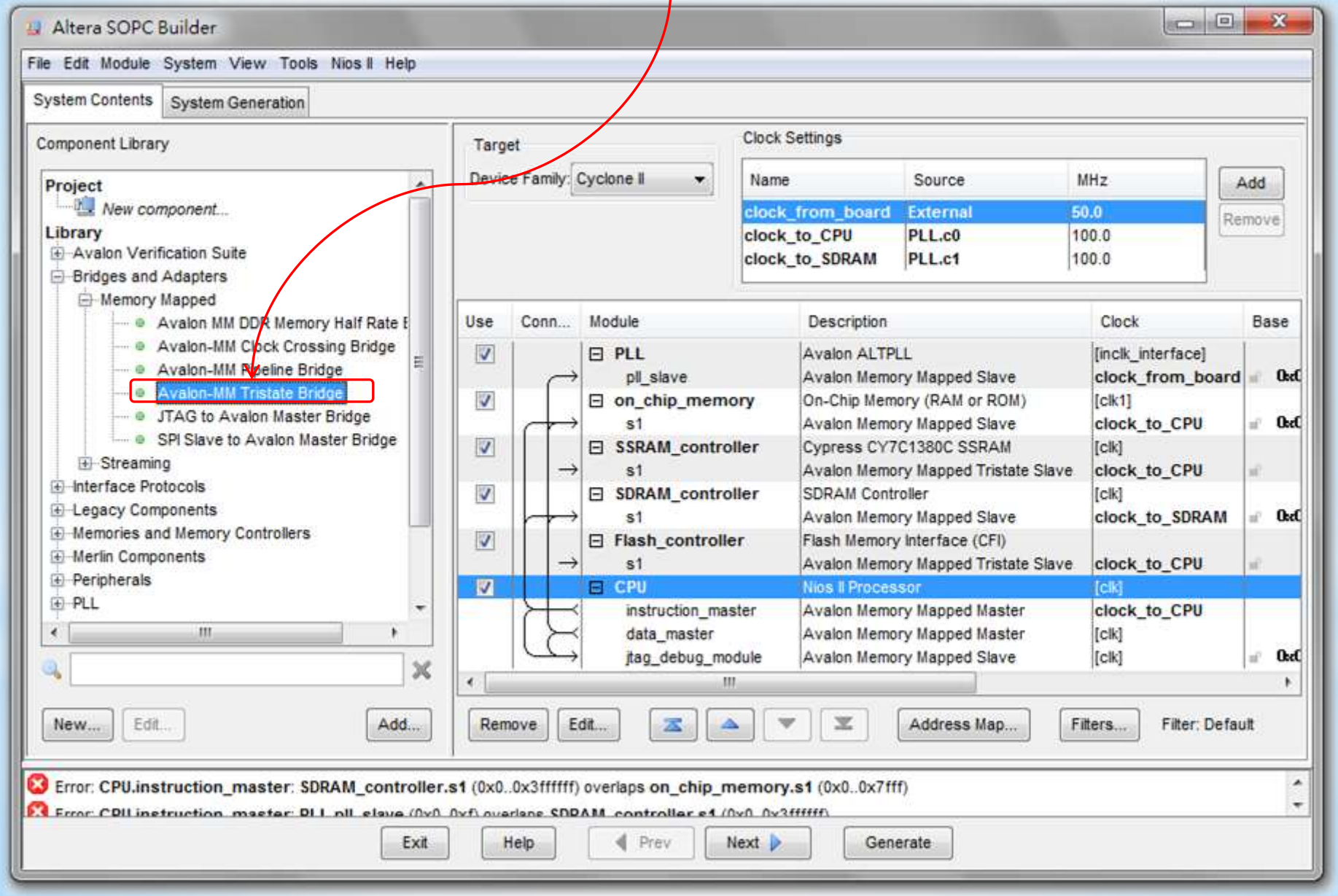


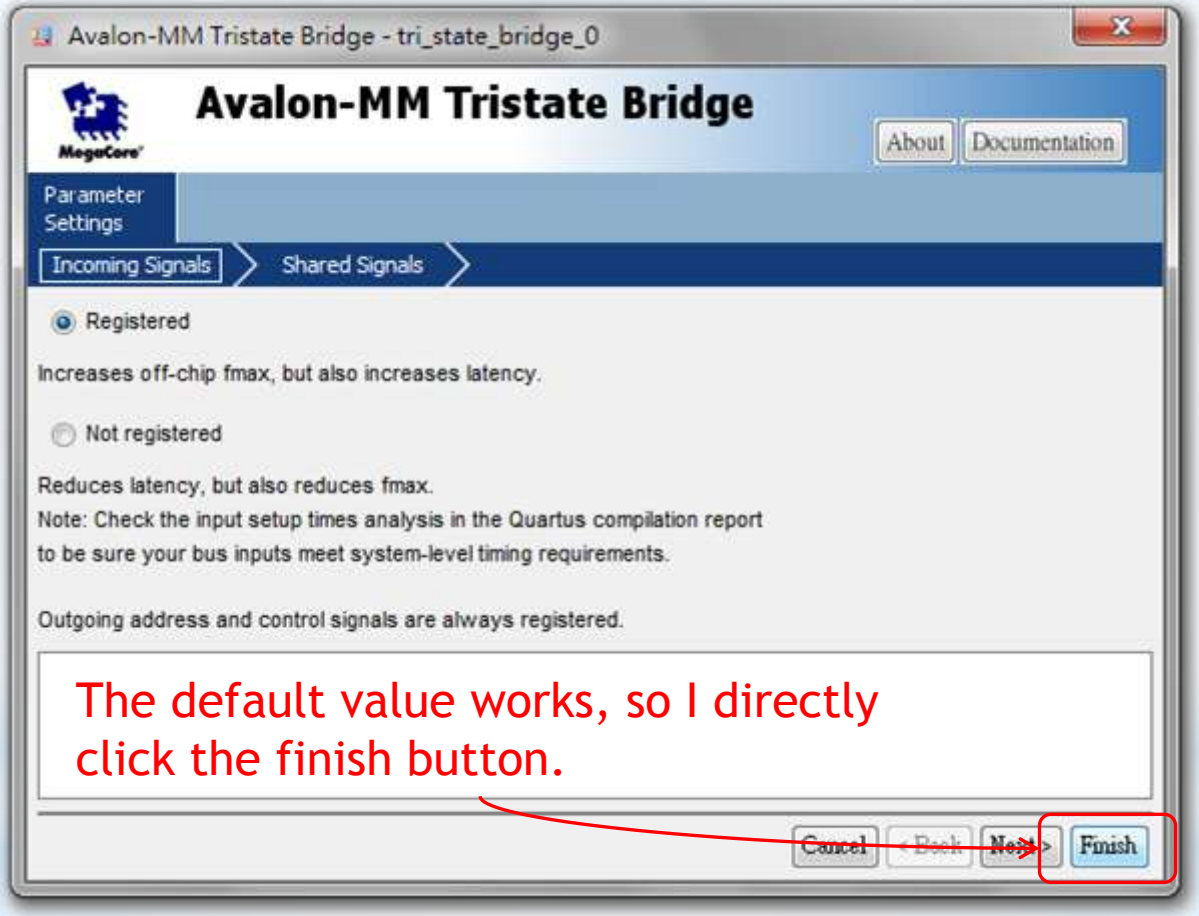

Nios II Processor - cpu_0
Nios II Processor
About
Documentation
Parameter Settings
Core Nios II
Caches and Memory Interfaces
Advanced Features
MMU and MPU Settings
JTAG Debug Module
Custom Instructions
Core Nios II
Select a Nios II core:
Nios II/e
Nios II/s
Nios II/f
Nios II
Selector Guide
Family: Cyclone II
f system: 50.0 MHz
cpuid: 0
RISC
32-bit
RISC
32-bit
Instruction Cache
Branch Prediction
Hardware Multiply
Hardware Divide
RISC
32-bit
Instruction Cache
Branch Prediction
Hardware Multiply
Hardware Divide
Barrel Shifter
Data Cache
Dynamic Branch Prediction
Performance at 50.0 MHz
Up to 5 DMIPS
Up to 25 DMIPS
Up to 51 DMIPS
Hardware Multiply: Embedded Multipliers
Hardware Divide
I want to use the fastest core.
Reset Vector: Memory: SDRAM_controller Offset: 0x0 0x00000000
Exception Vector: Memory: SDRAM_controller Offset: 0x20 0x00000020
Include MMU
Only include the MMU when using an operating system that explicitly supports an MMU
Fast TLB Miss Exception Vector: Memory: Offset: 0x0
Include MPU
I set both Reset Vector and Exception Vector to the largest memory. (this is just example setting)
Cancel
< Back
Next >
Finish

I rename the new processor to CPU with clock_to_CPU. However, you'll notice that SSRAM_controller and Flash_controller cannot be connected to CPU. Why?

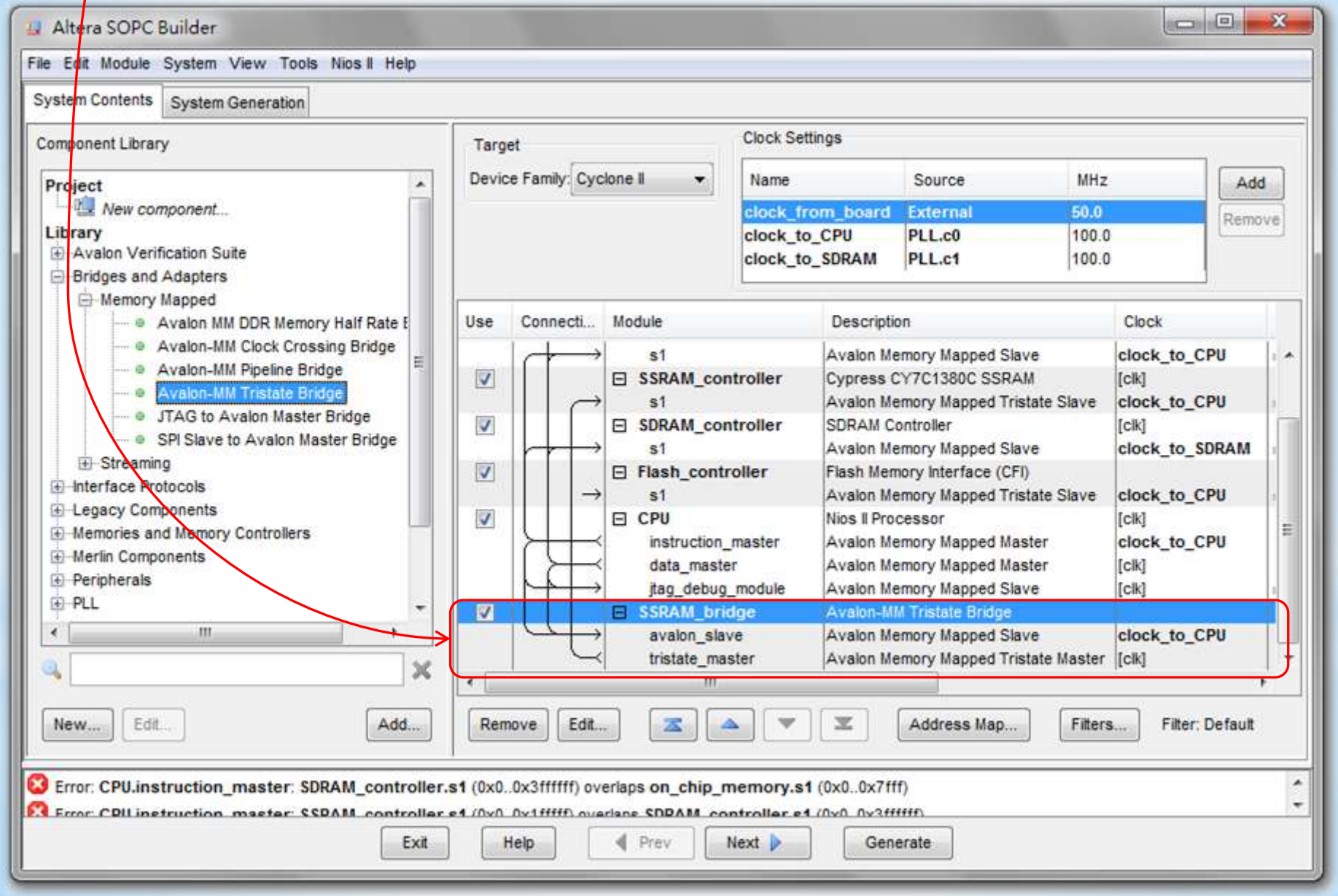


Because SSRAM_controller uses Tristate Interface, I need Tristate Bridge. Simply Click Library -> Bridges and Adapters -> Memory Mapped -> Avalon-MM Tristate Bridge.

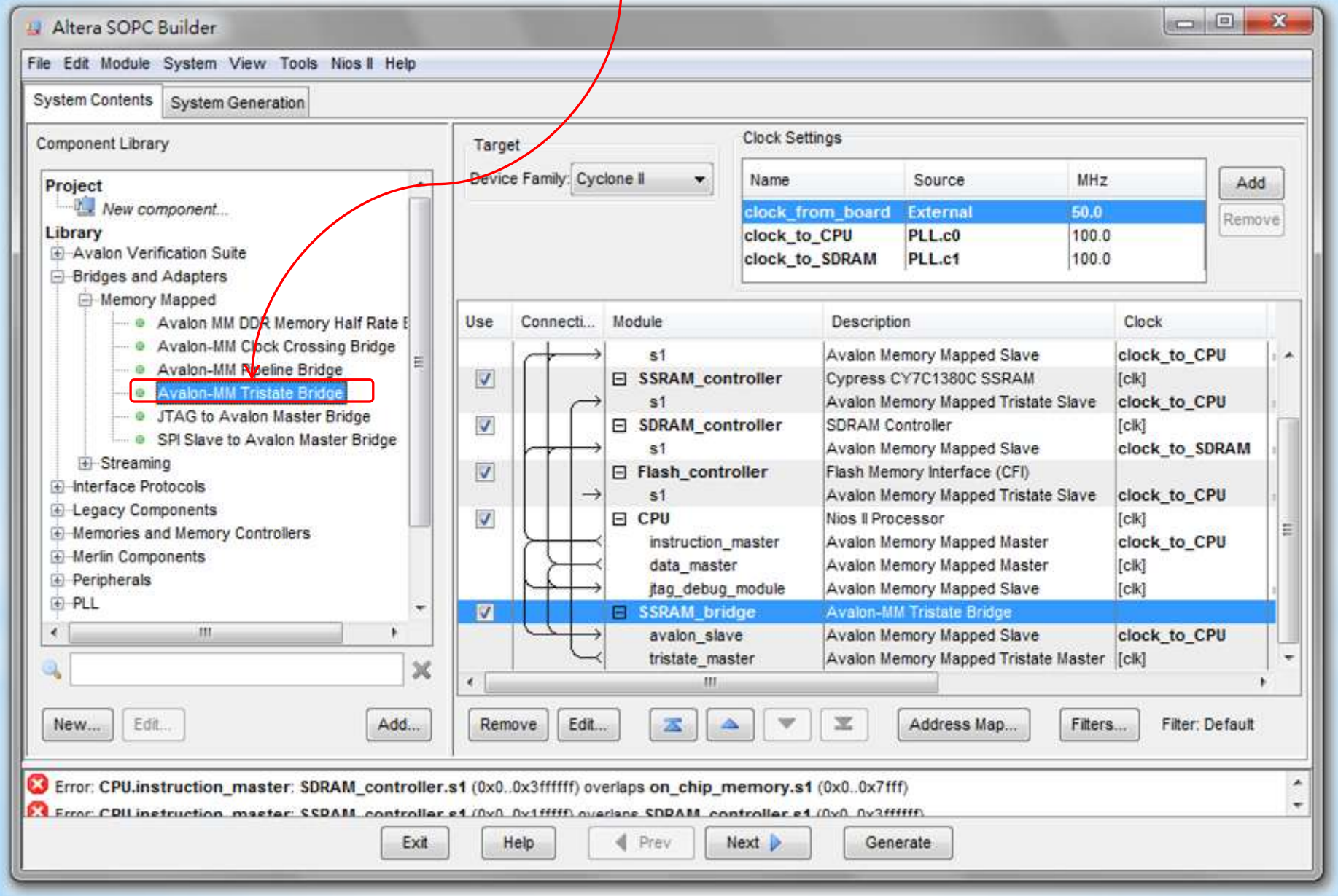


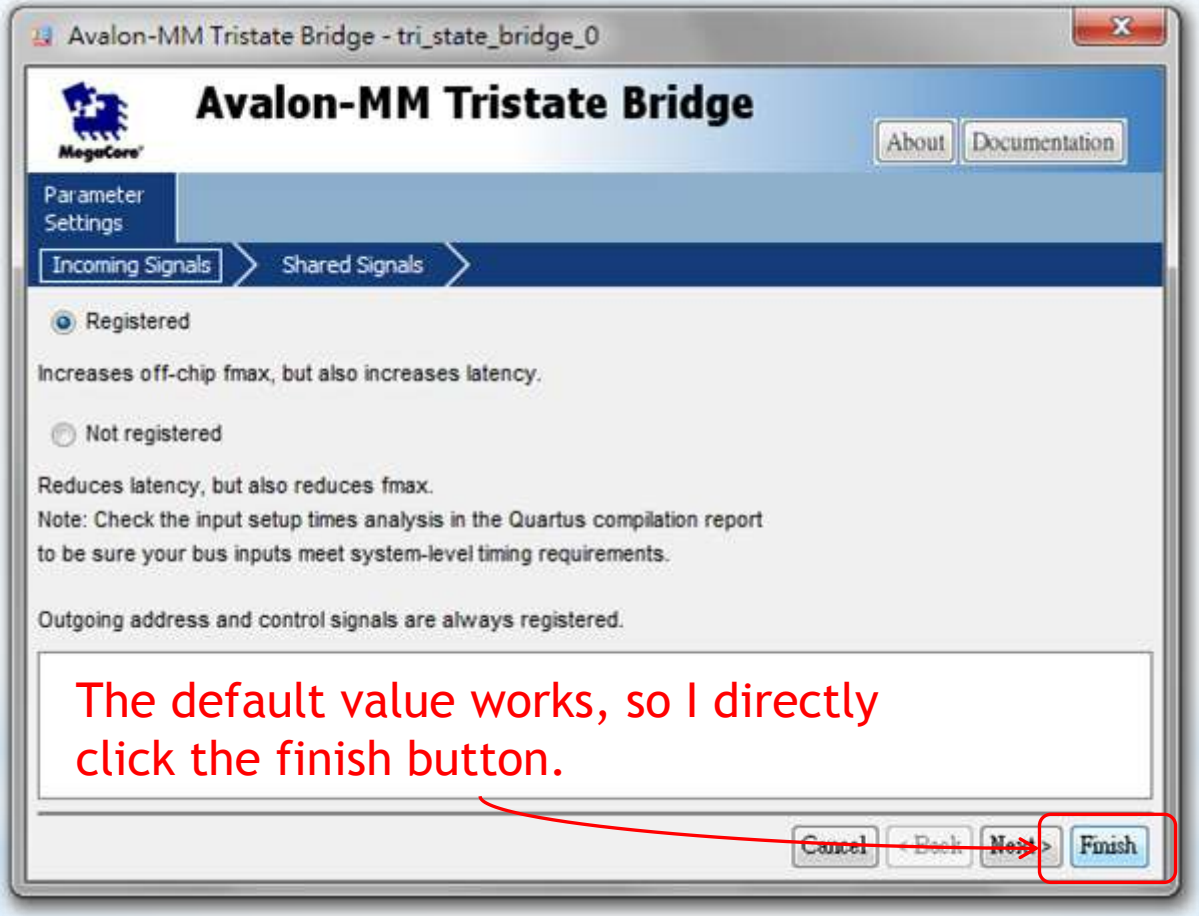

Avalon-MM Tristate Bridge - tri_state_bridge_0
Avalon-MM Tristate Bridge
MegaCore
About
Documentation
Parameter Settings
Incoming Signals
Shared Signals
Registered
Increases off-chip fmax, but also increases latency.
Not registered
Reduces latency, but also reduces fmax.
Note: Check the input setup times analysis in the Quartus compilation report
to be sure your bus inputs meet system-level timing requirements.
Outgoing address and control signals are always registered.
The default value works, so I directly click the finish button.
Cancel
< Back
Next >
Finish

I rename the new bridge to SSRAM_bridge and connect SSRAM_bridge to SSRAM_controller. Also, this bridge are using clock_to_CPU.

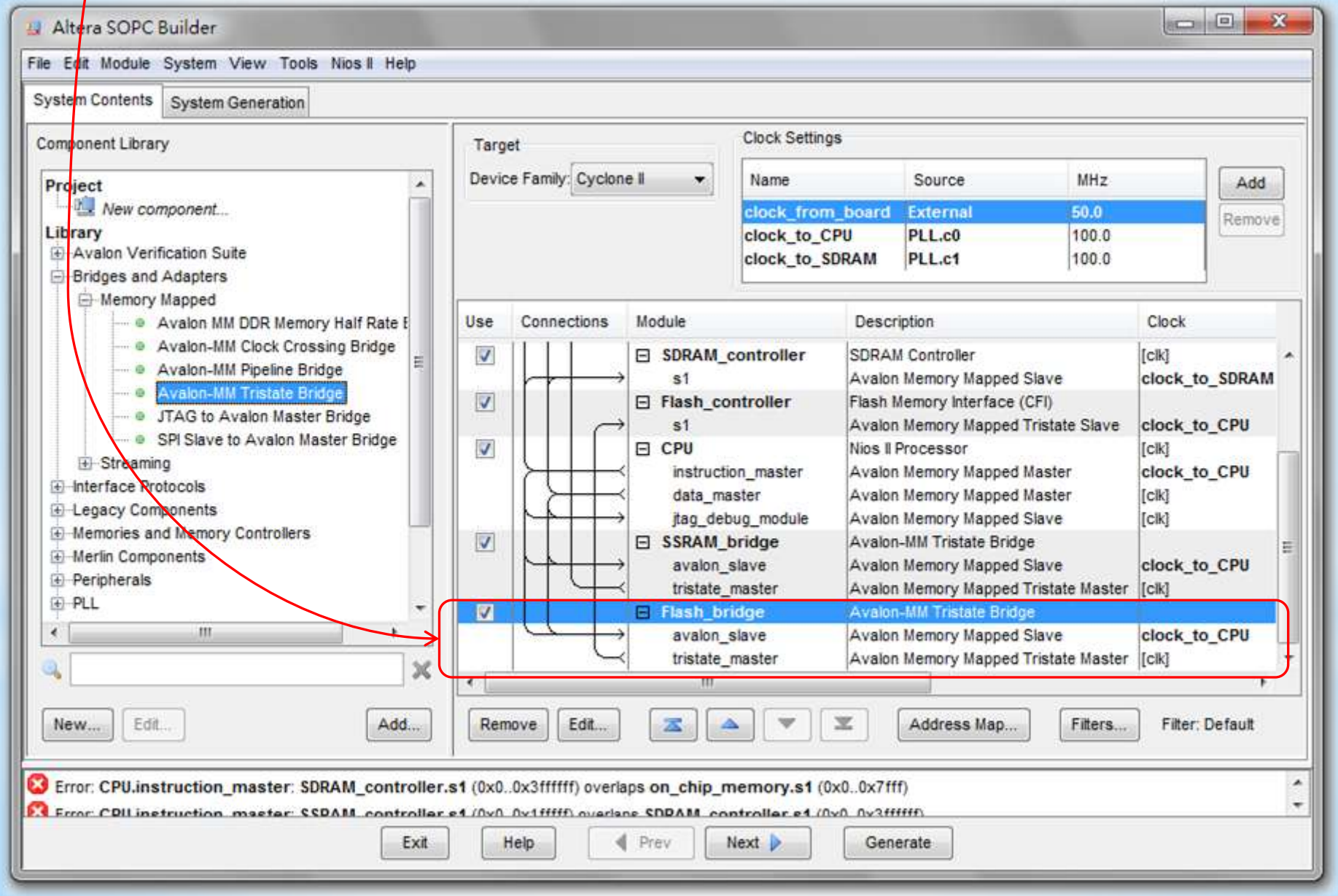


Similarly, Flash_controller also need Tristate Bridge, so I click Library -> Bridges and Adapters -> Memory Mapped -> Avalon-MM Tristate Bridge.

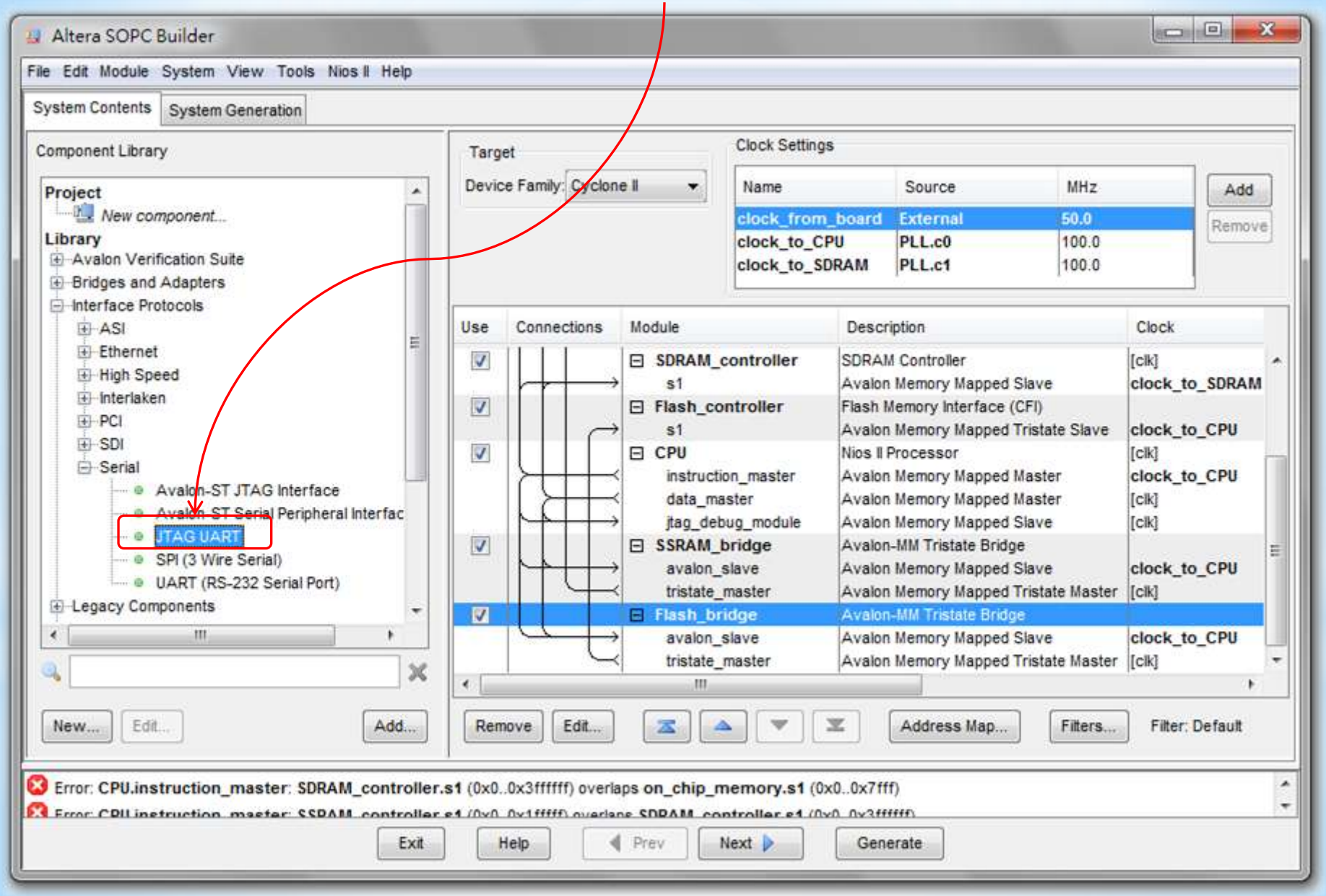


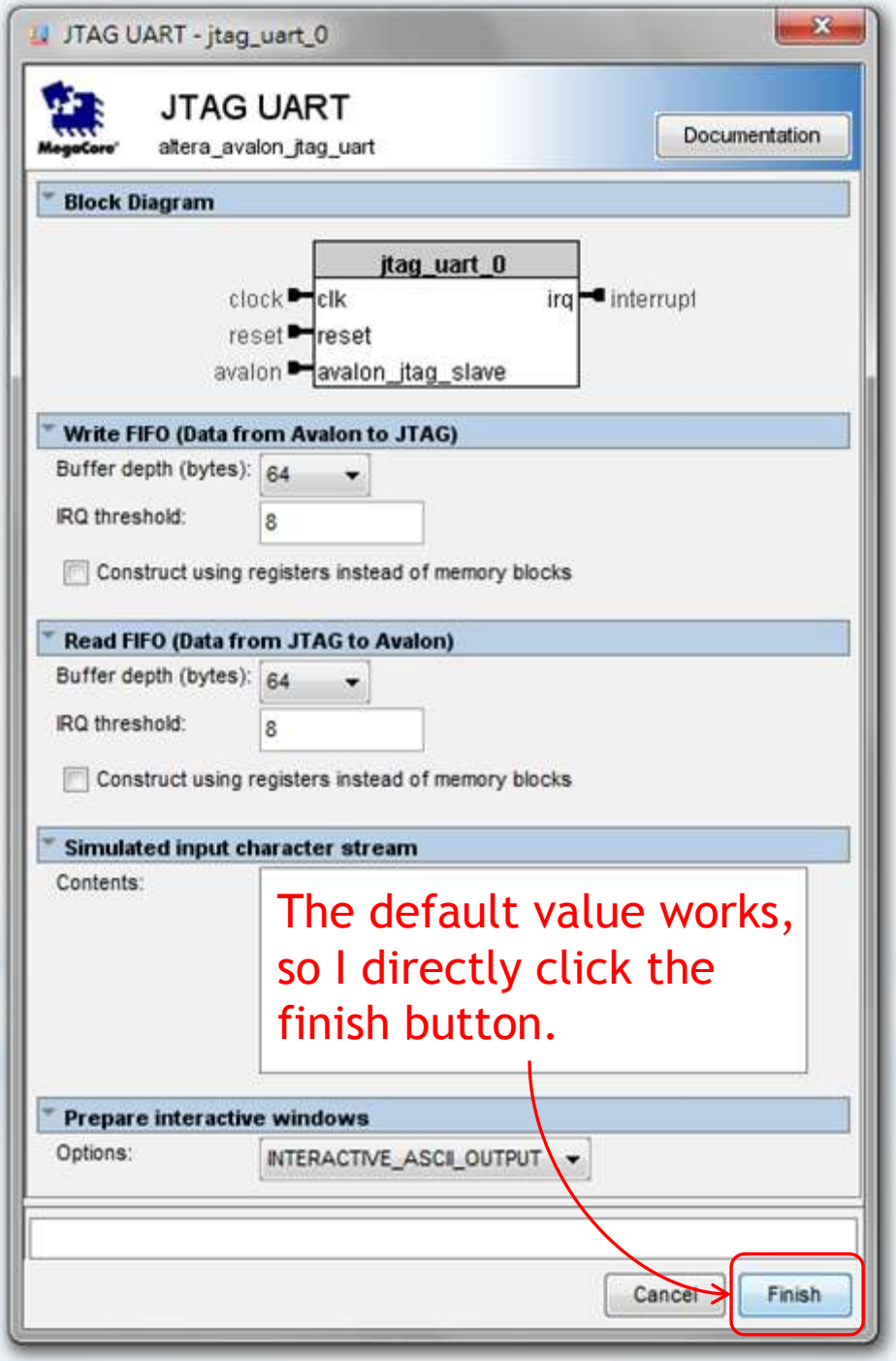

Avalon-MM Tristate Bridge - tri_state_bridge_0
Avalon-MM Tristate Bridge
MegaCore
About
Documentation
Parameter Settings
Incoming Signals
Shared Signals
Registered
Increases off-chip fmax, but also increases latency.
Not registered
Reduces latency, but also reduces fmax.
Note: Check the input setup times analysis in the Quartus compilation report
to be sure your bus inputs meet system-level timing requirements.
Outgoing address and control signals are always registered.
The default value works, so I directly click the finish button.
Cancel
< Back
Next >
Finish

I rename the new bridge to Flash_bridge and connect Flash_bridge to Flash_controller. Also, this bridge are using clock_to_CPU.

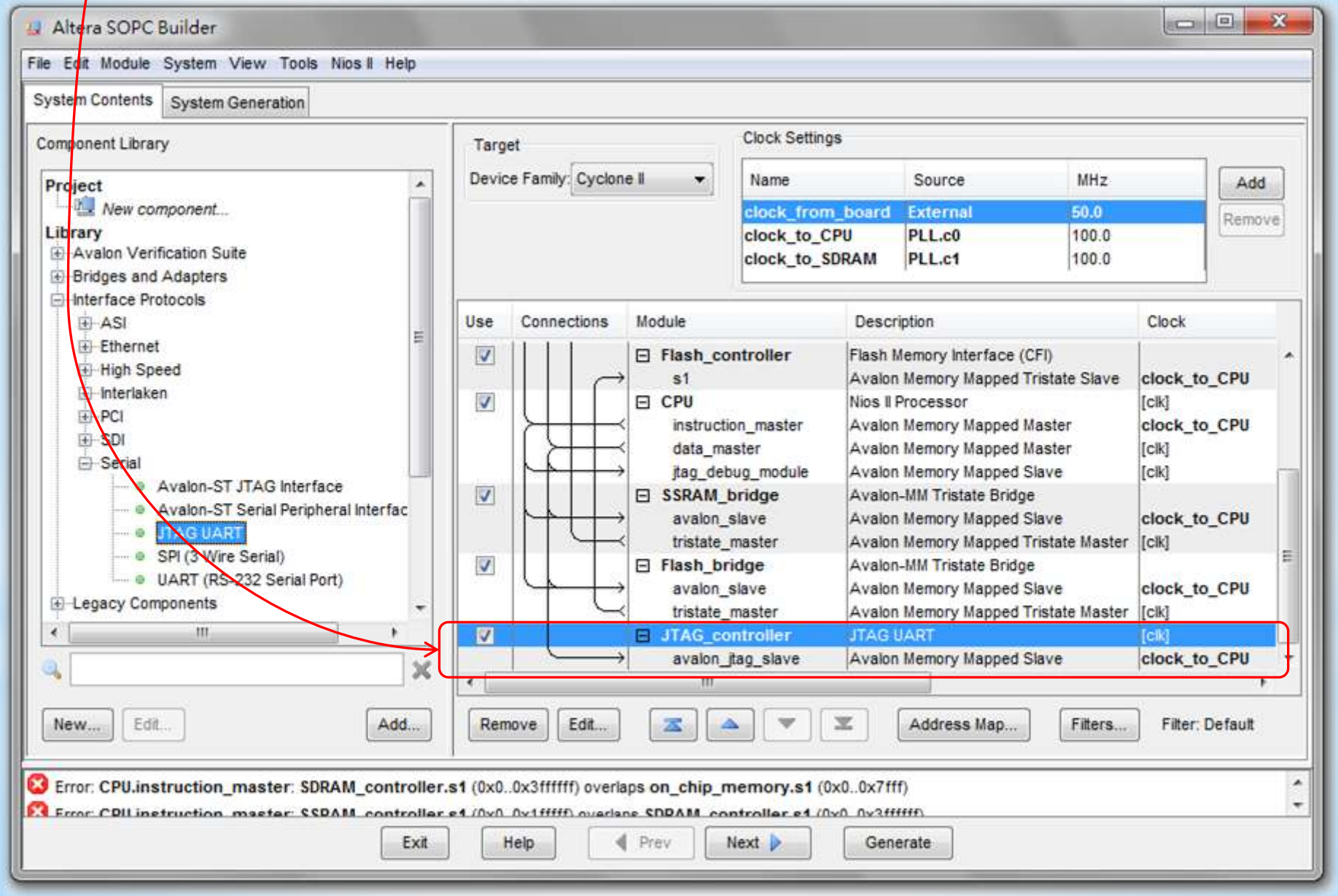


Now I have a system (CPU and memories) that can run programs. To communicate with Nios II EDS Console, I click Library -> Interface Protocols -> Serial -> JTAG UART.

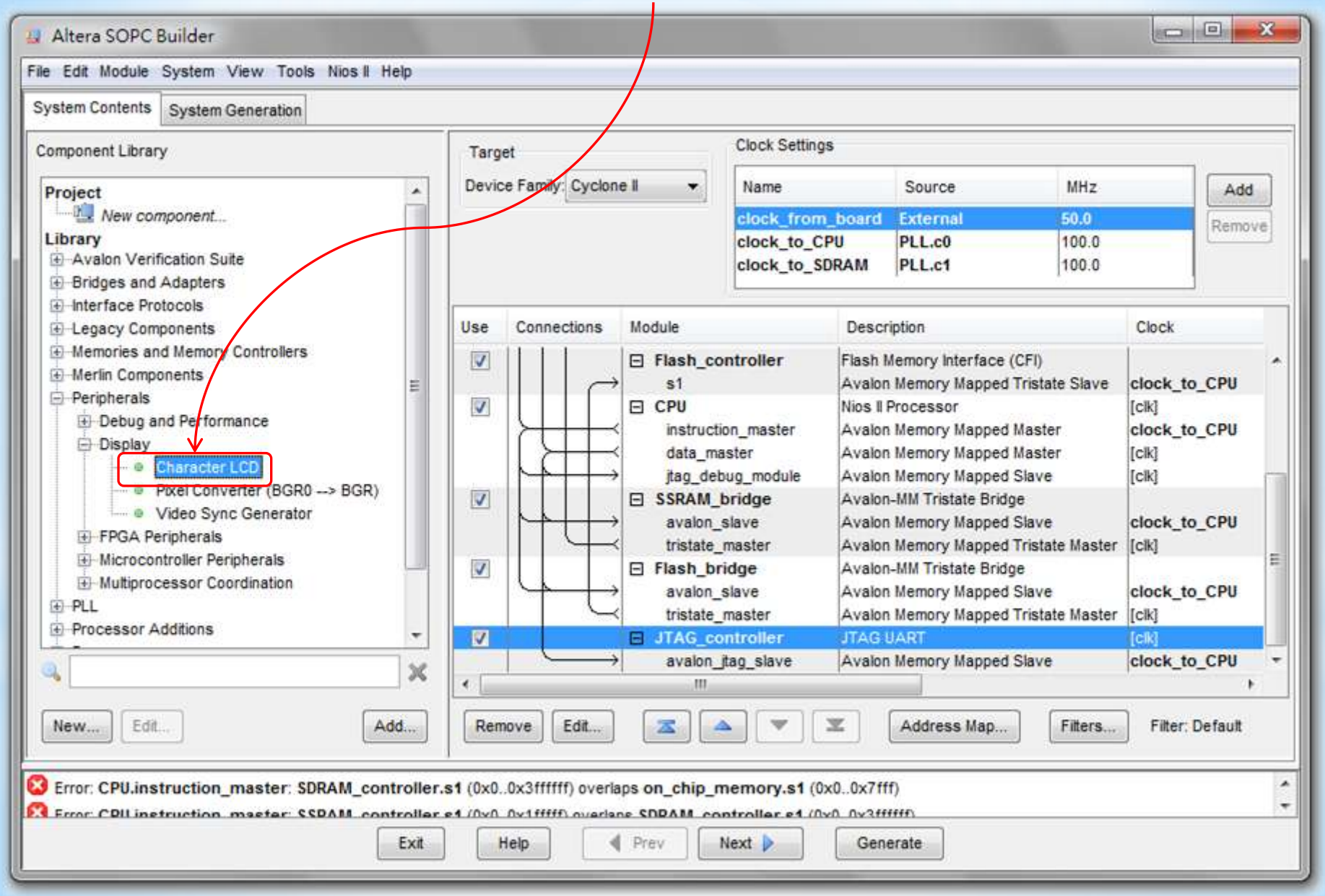


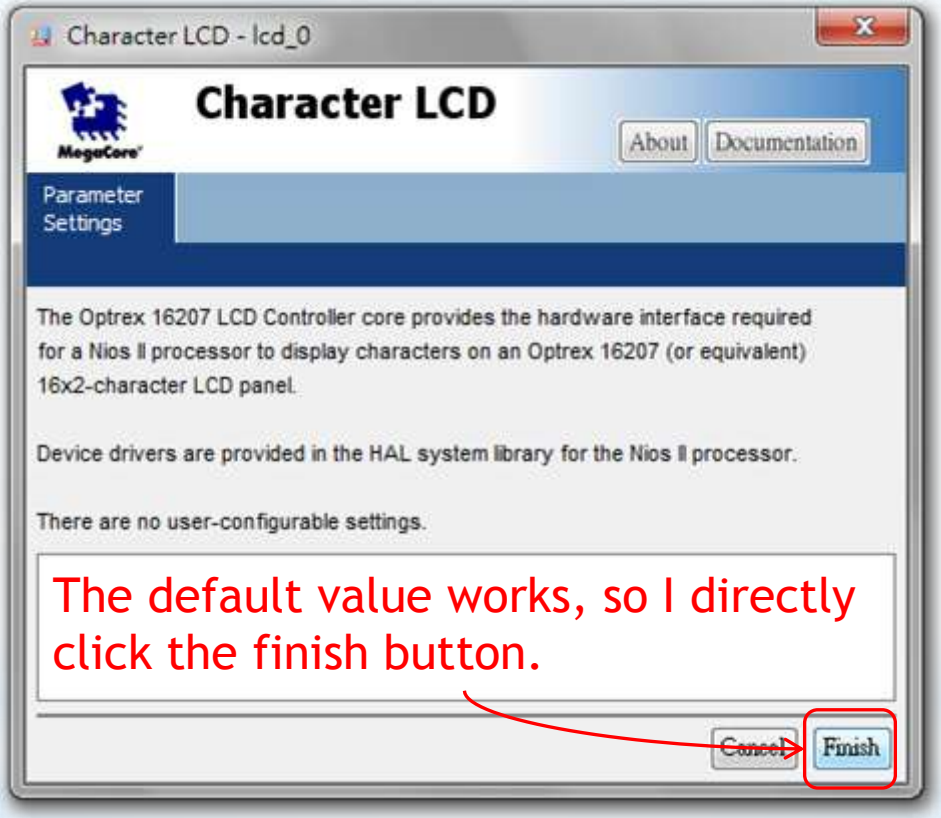

JTAG UART - jtag_uart_0
JTAG UART
altera_avalon_jtag_uart
Documentation
Block Diagram
jtag_uart_0
clock
clk
irq
interrupt
reset
reset
avalon
avalon_jtag_slave
Write FIFO (Data from Avalon to JTAG)
Buffer depth (bytes):
64
IRQ threshold:
8
Construct using registers instead of memory blocks
Read FIFO (Data from JTAG to Avalon)
Buffer depth (bytes):
64
IRQ threshold:
8
Construct using registers instead of memory blocks
Simulated input character stream
Contents:
The default value works, so I directly click the finish button.
Prepare interactive windows
Options:
INTERACTIVE_ASCII_OUTPUT
Cancel
Finish

I rename the new controller to JTAG_controller with clock_to_CPU.

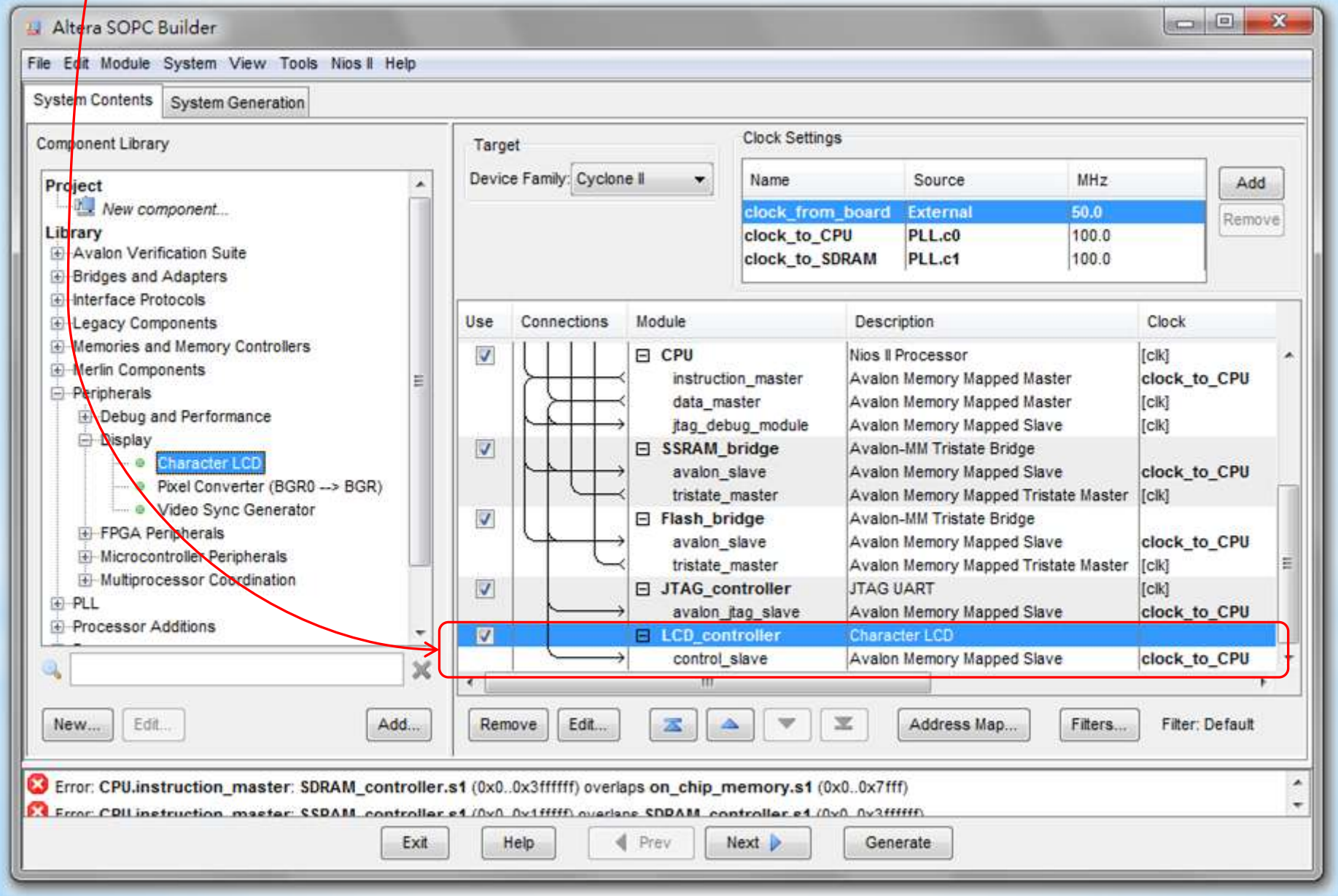


Similarly, I need the system can communicate with Character LCD, so I click Library -> Peripherals -> Display -> Character LCD.

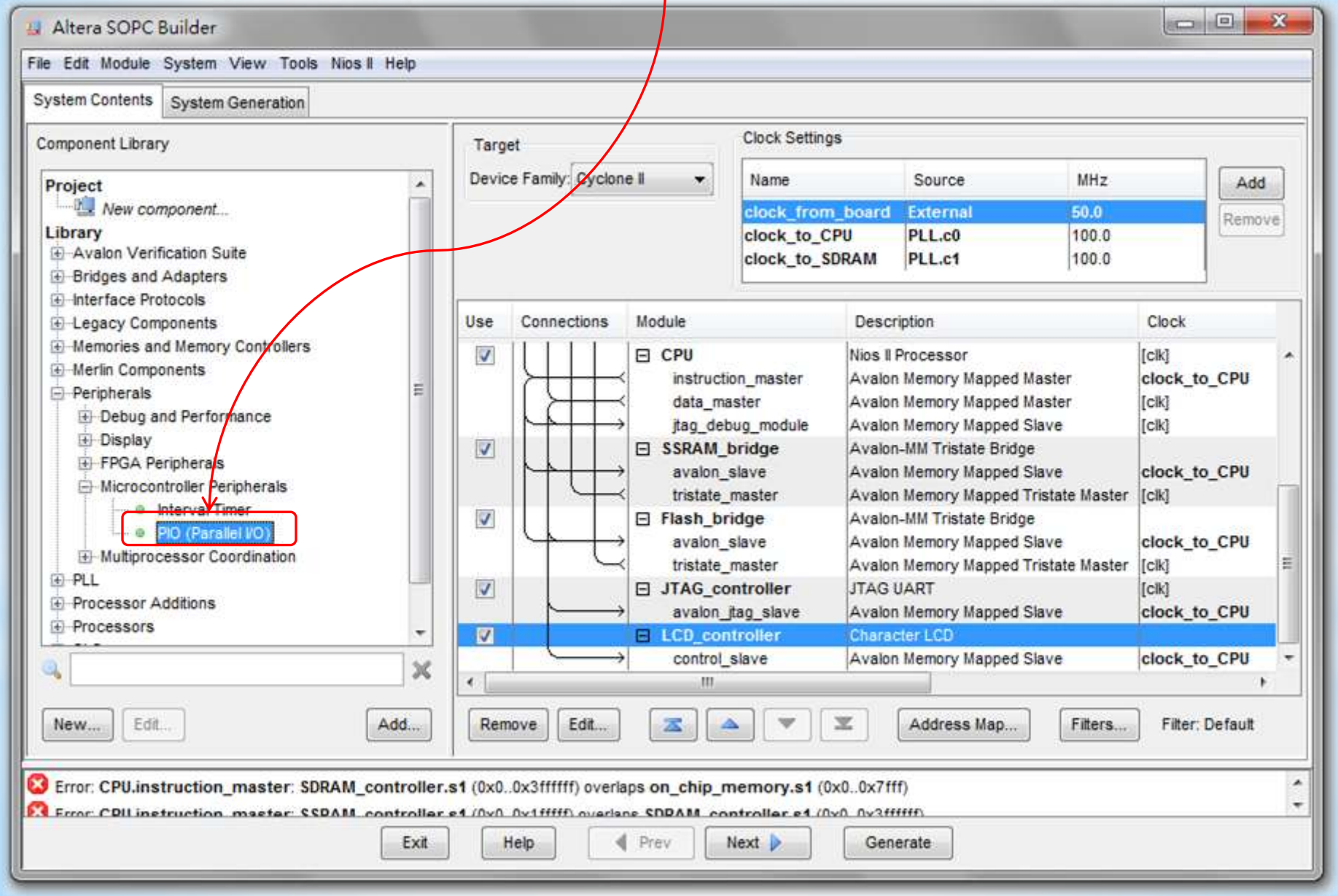



Character LCD - lcd_0
Character LCD
About
Documentation
Parameter Settings
The Optrex 16207 LCD Controller core provides the hardware interface required for a Nios II processor to display characters on an Optrex 16207 (or equivalent) 16x2-character LCD panel.
Device drivers are provided in the HAL system library for the Nios II processor.
There are no user-configurable settings.
The default value works, so I directly click the finish button.
Cancel
Finish

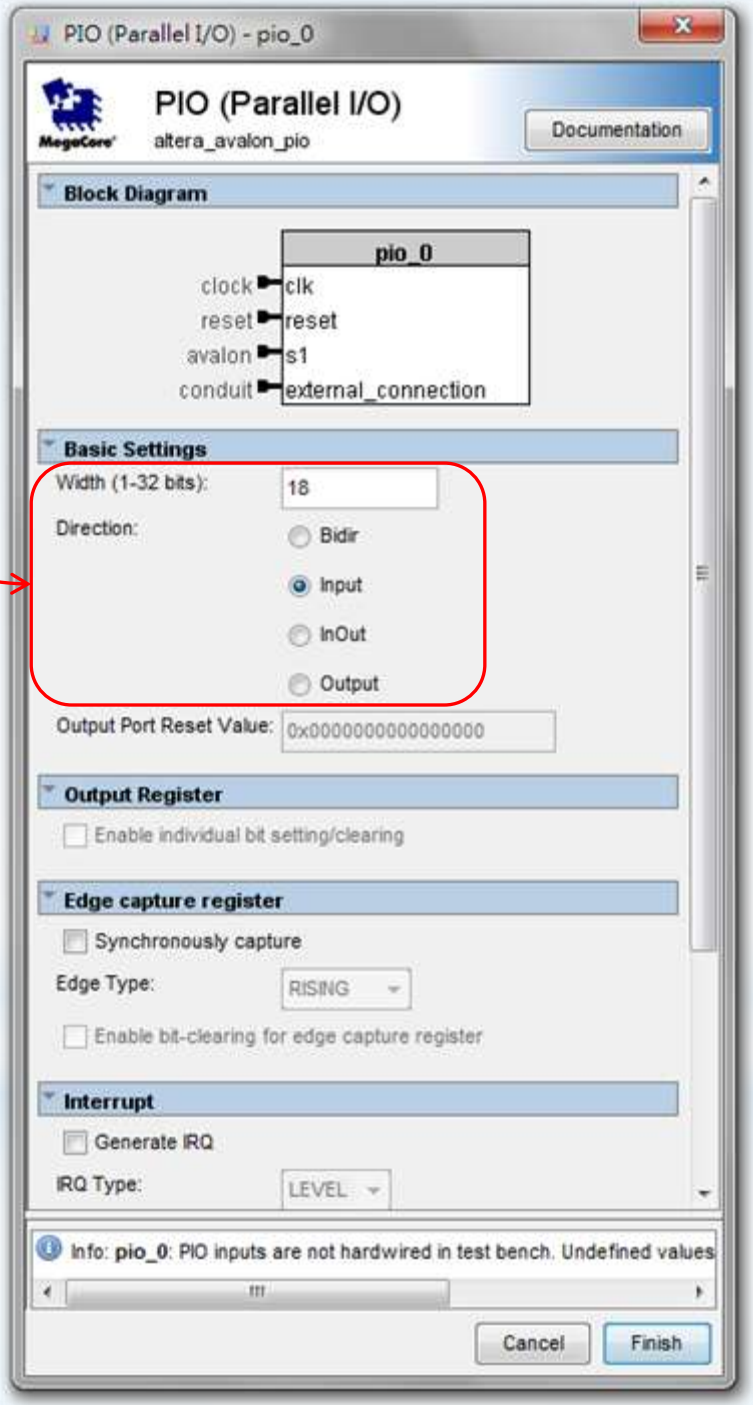
I rename the new controller to LCD_controller with clock_to_CPU.



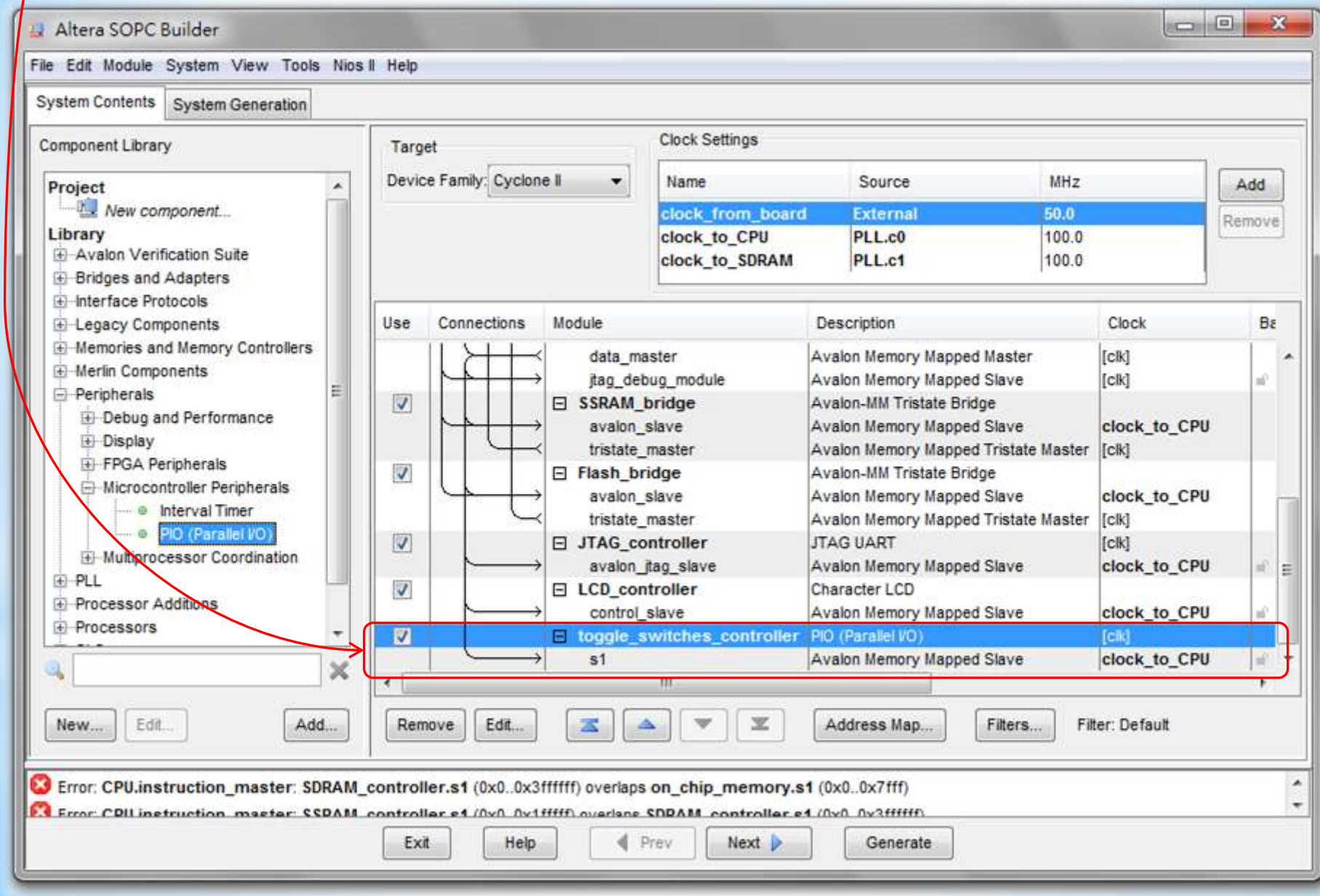
I need the system can sense the 18 Toggle Switches, so I click Library -> Peripherals -> Microcontroller Peripherals -> PIO (Parallel I/O).



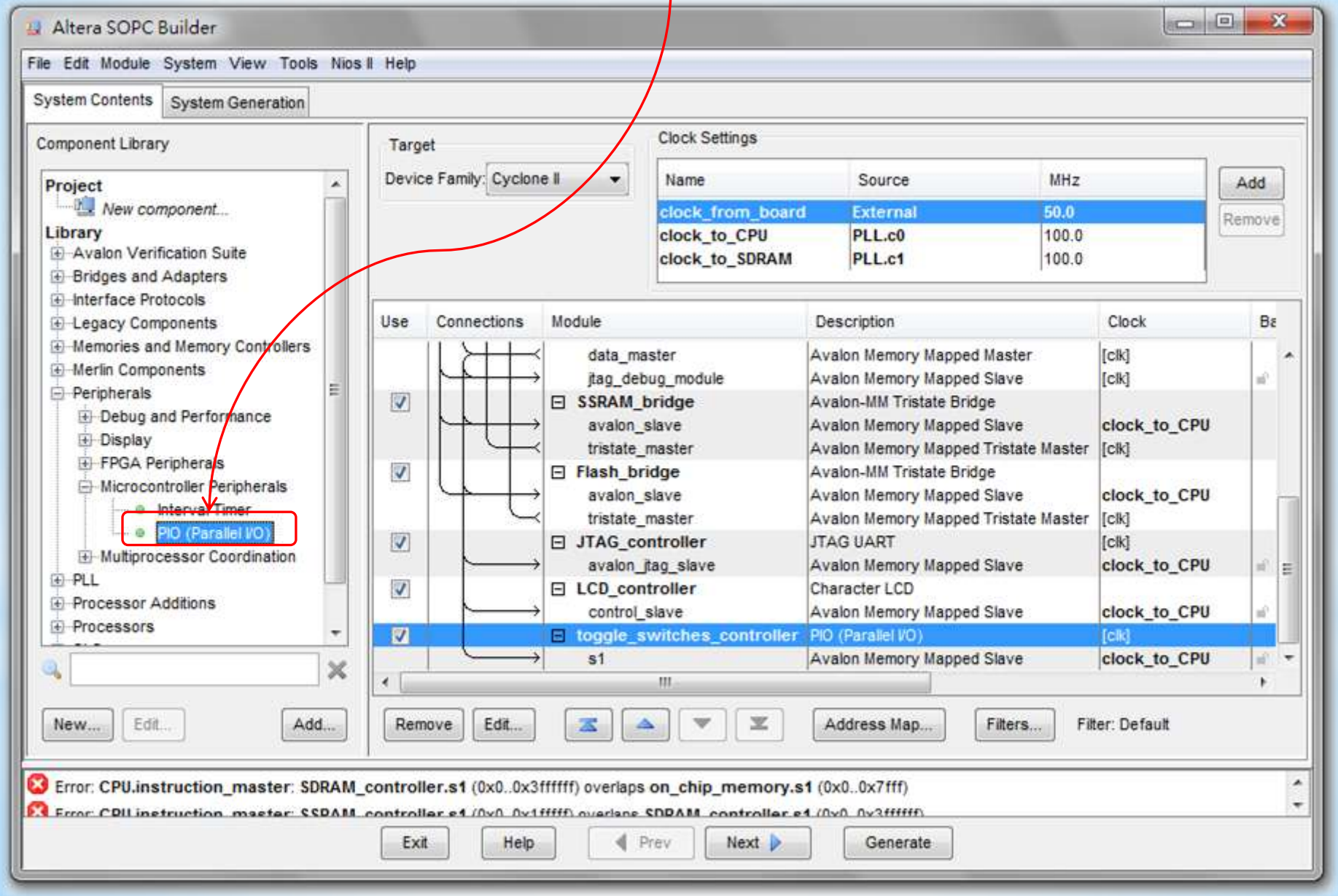
I need to sense 18 inputs for Toggle Switches.



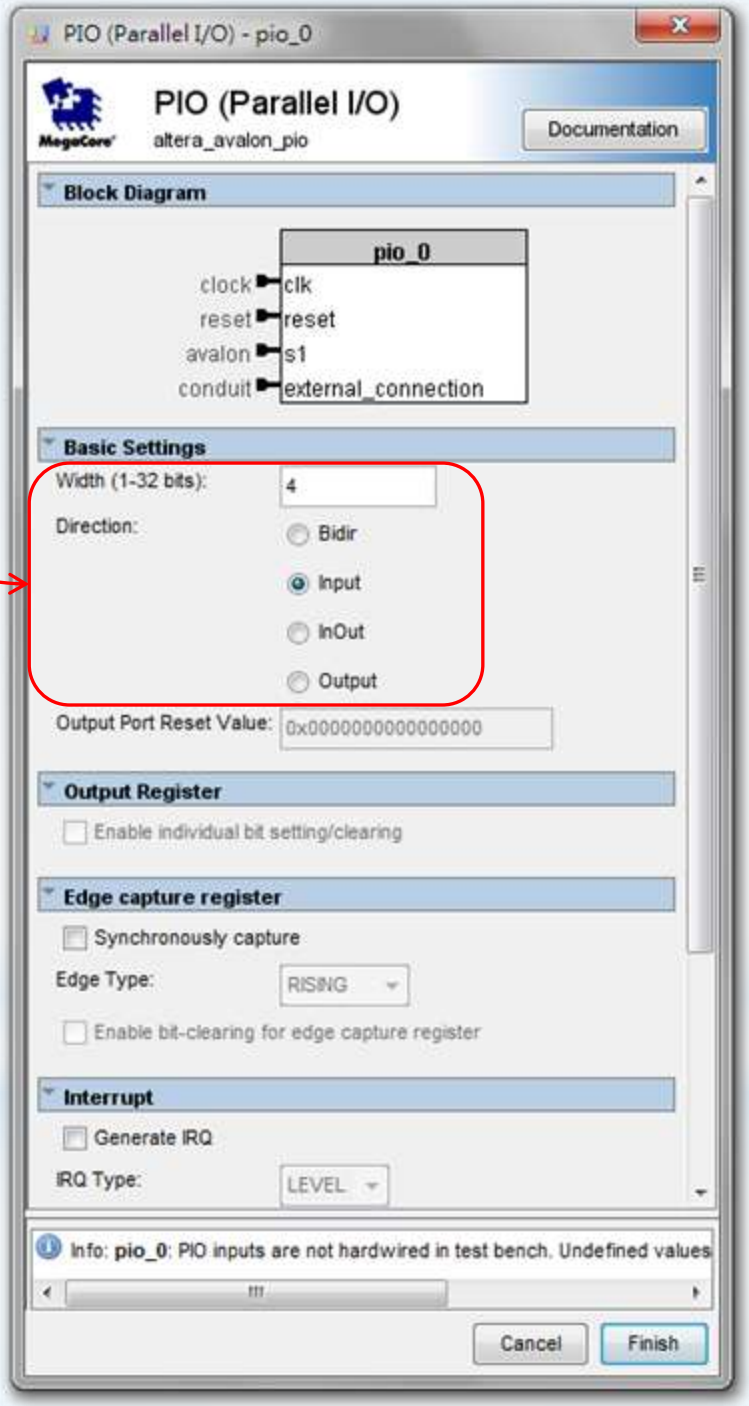
I rename the new controller to toggle_switches_controller with clock_to_CPU.



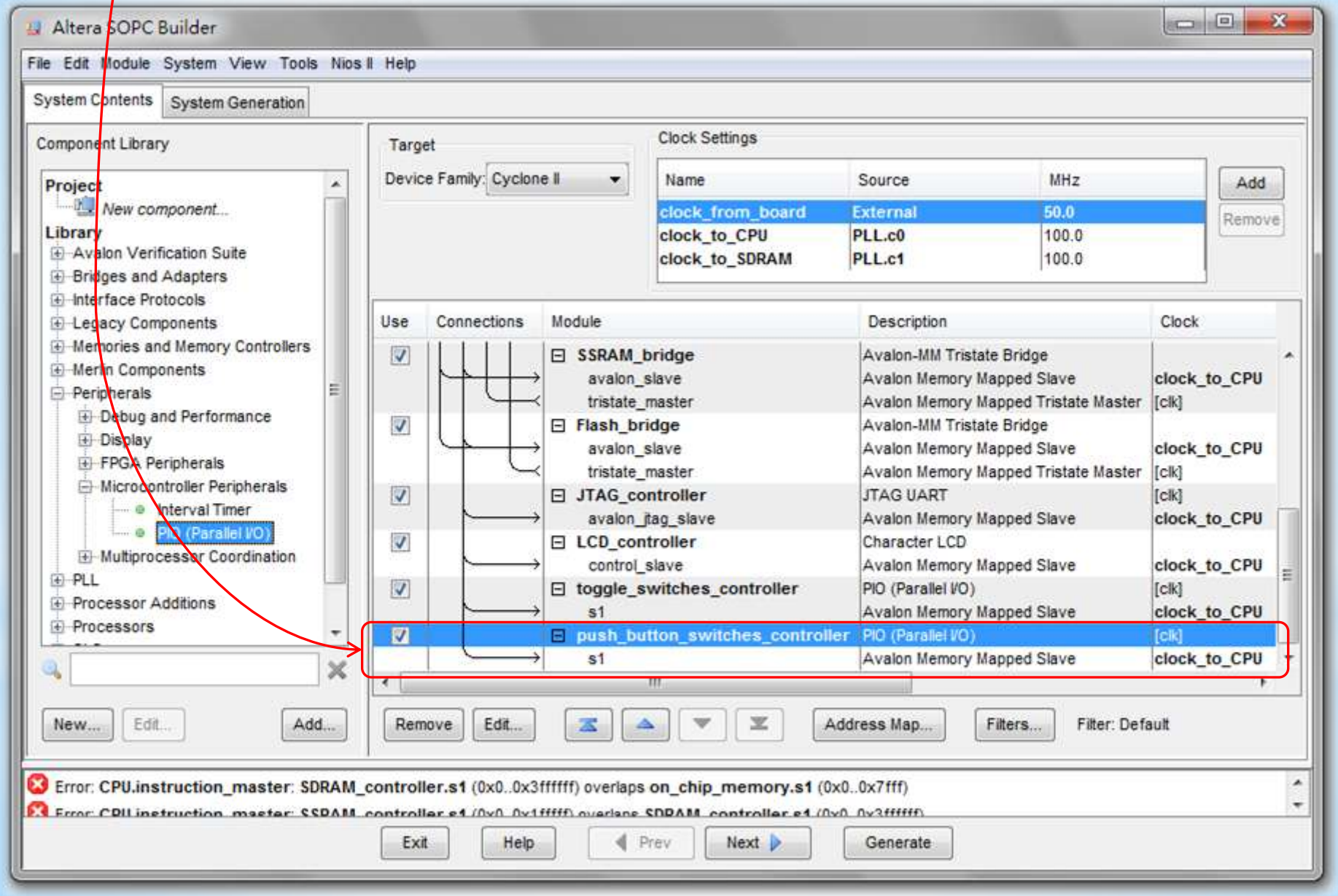
I need the system can sense the 4 Push-Button Switches, so I click Library -> Peripherals -> Microcontroller Peripherals -> PIO (Parallel I/O).



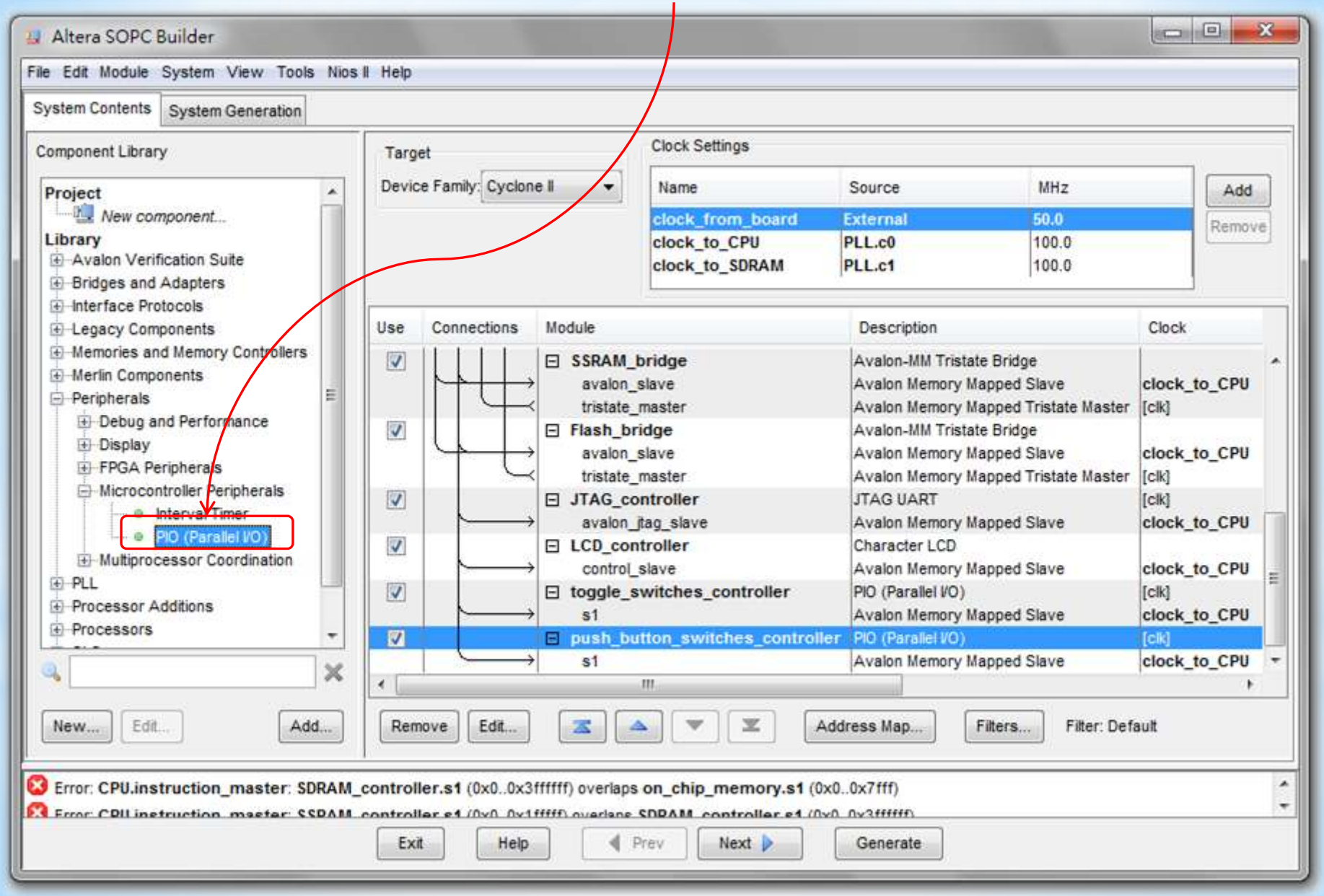
I need to sense 4 inputs for Push-Button Switches.



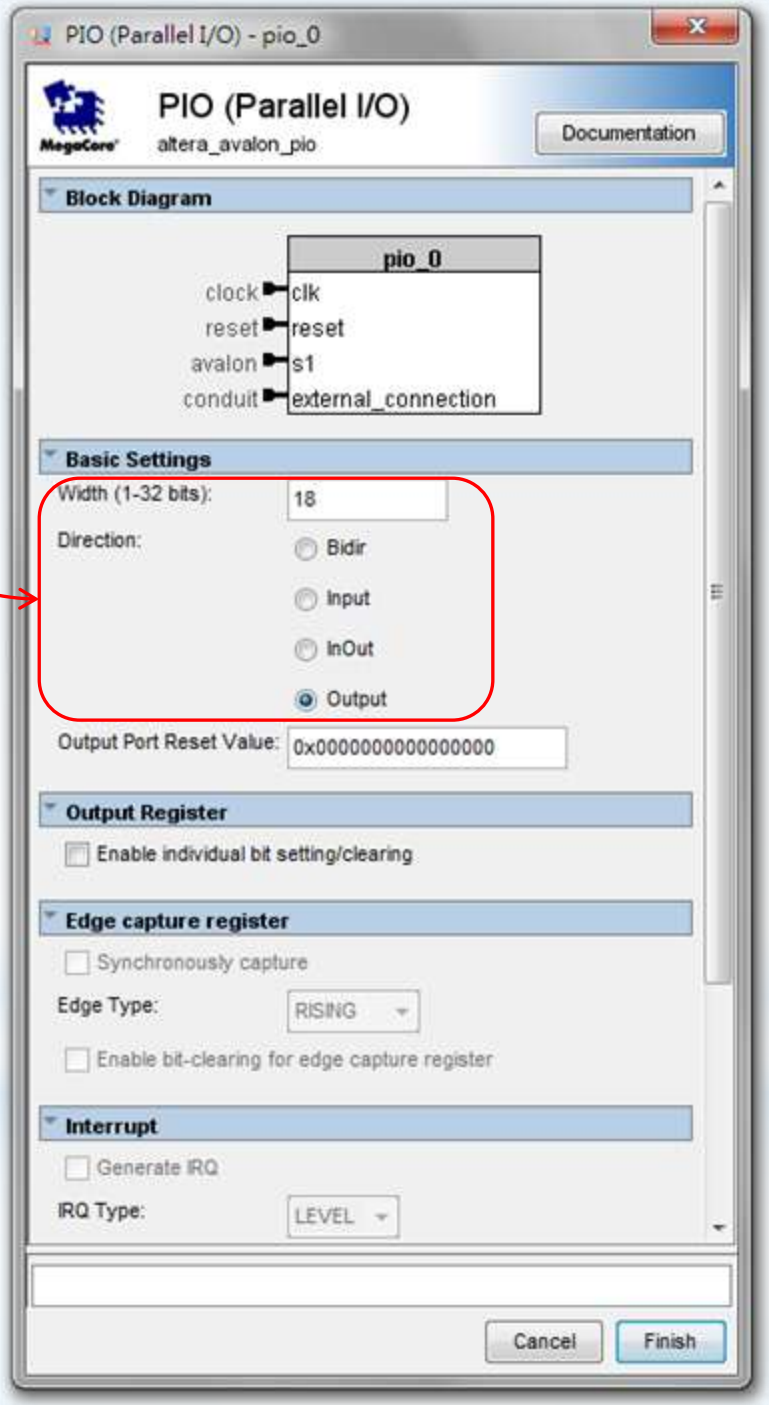
I rename the new controller to push_button_switches_controller with clock_to_CPU.



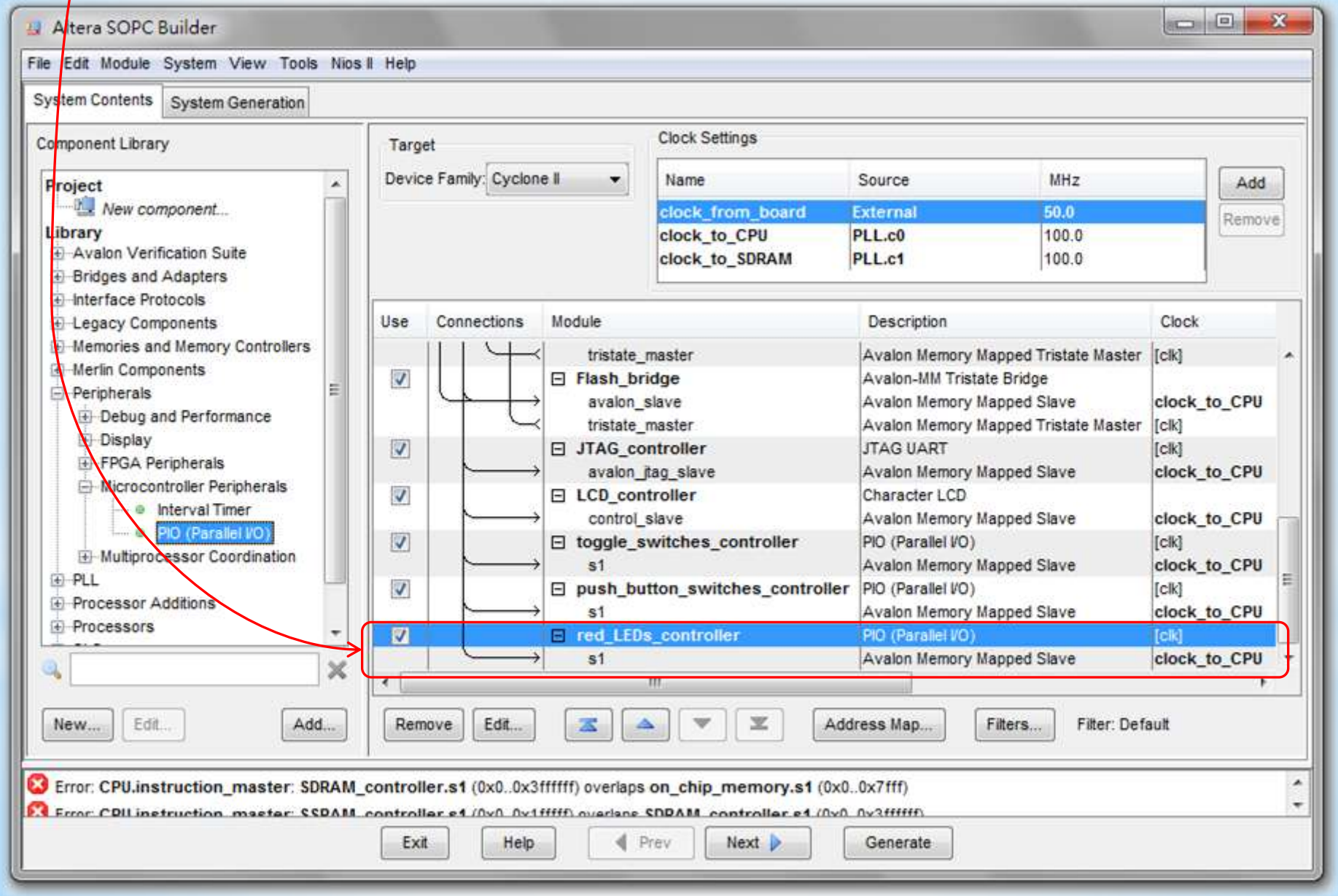
I need the system can control the 18 red LEDs, so I click Library -> Peripherals -> Microcontroller Peripherals -> PIO (Parallel I/O).



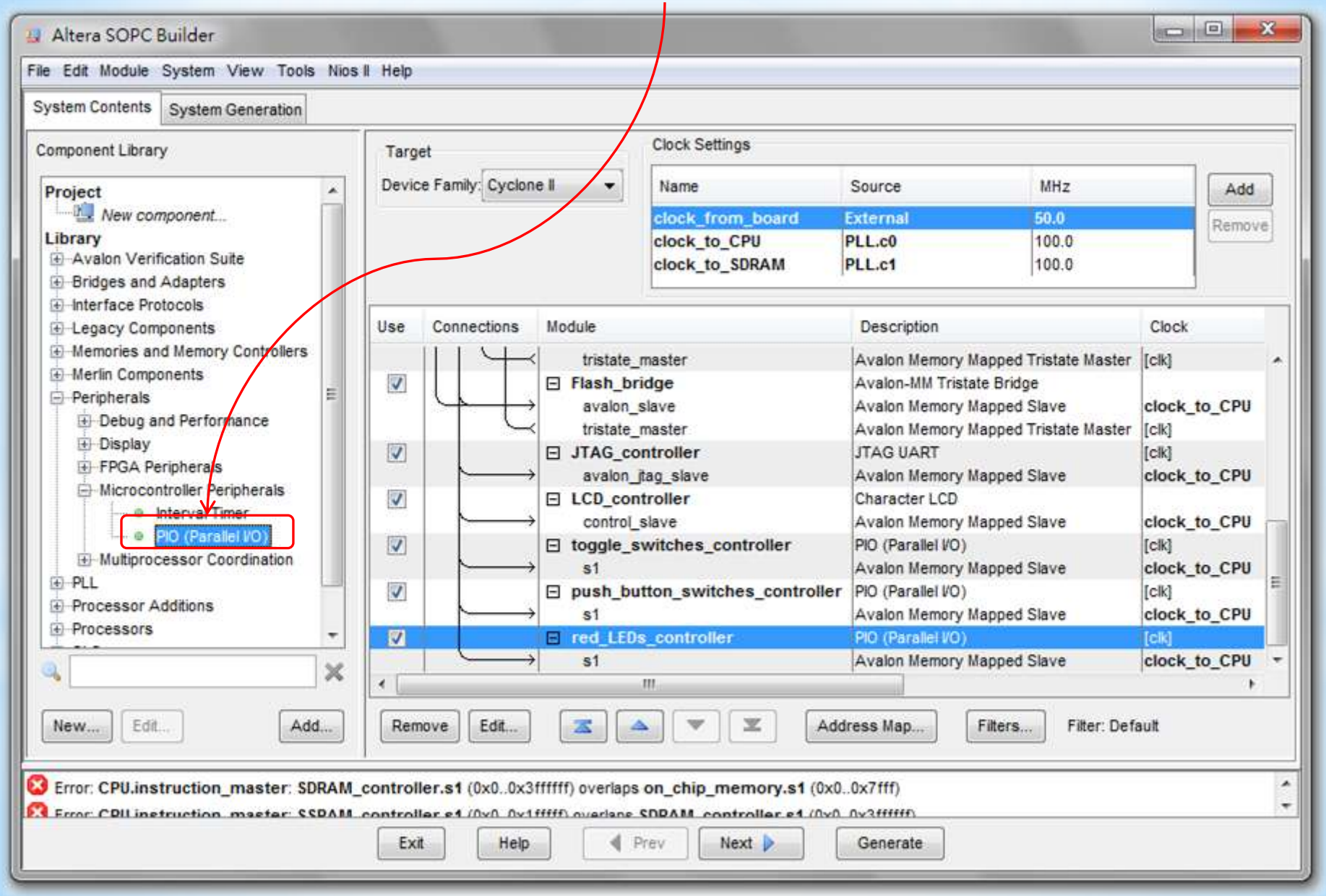
I need to control 18 outputs for red LEDs.



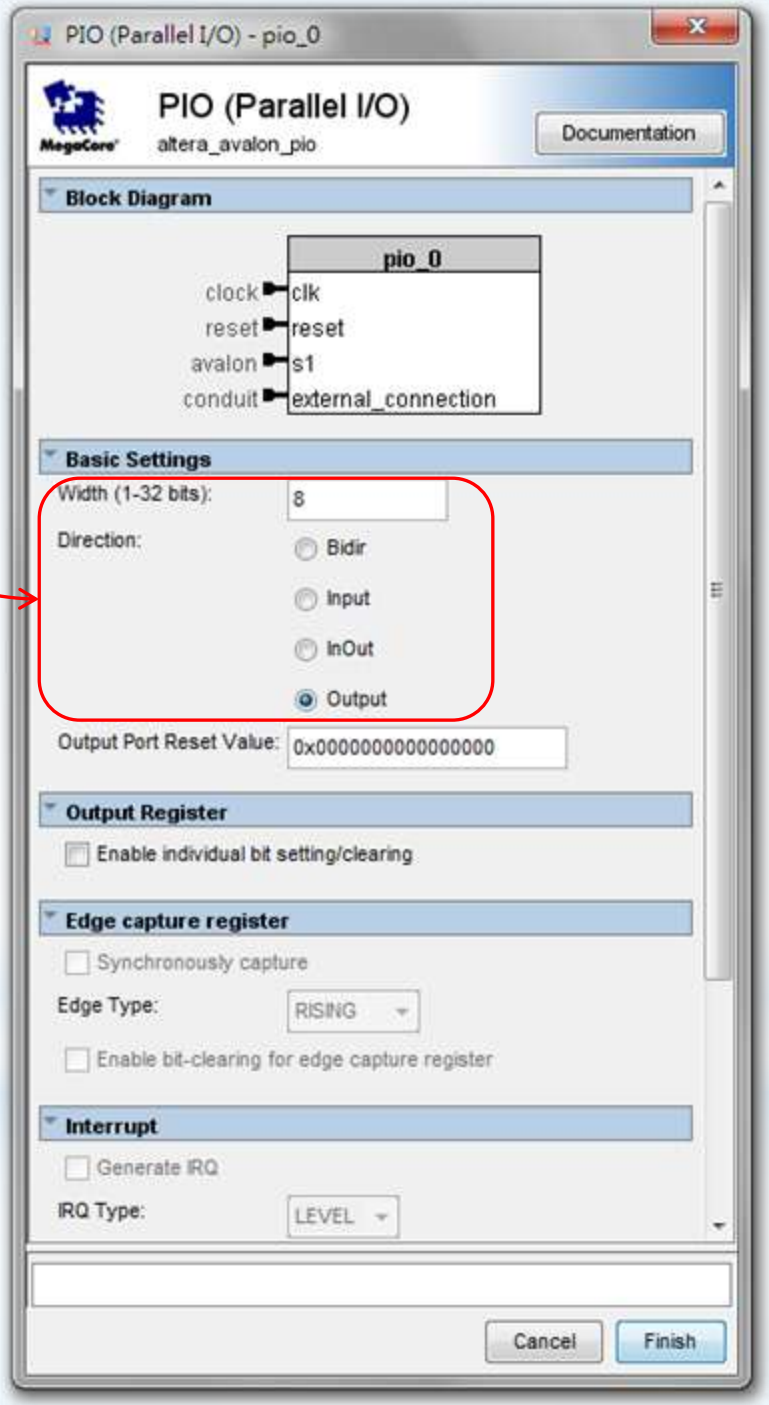
I rename the new controller to red_LEDs_controller with clock_to_CPU.



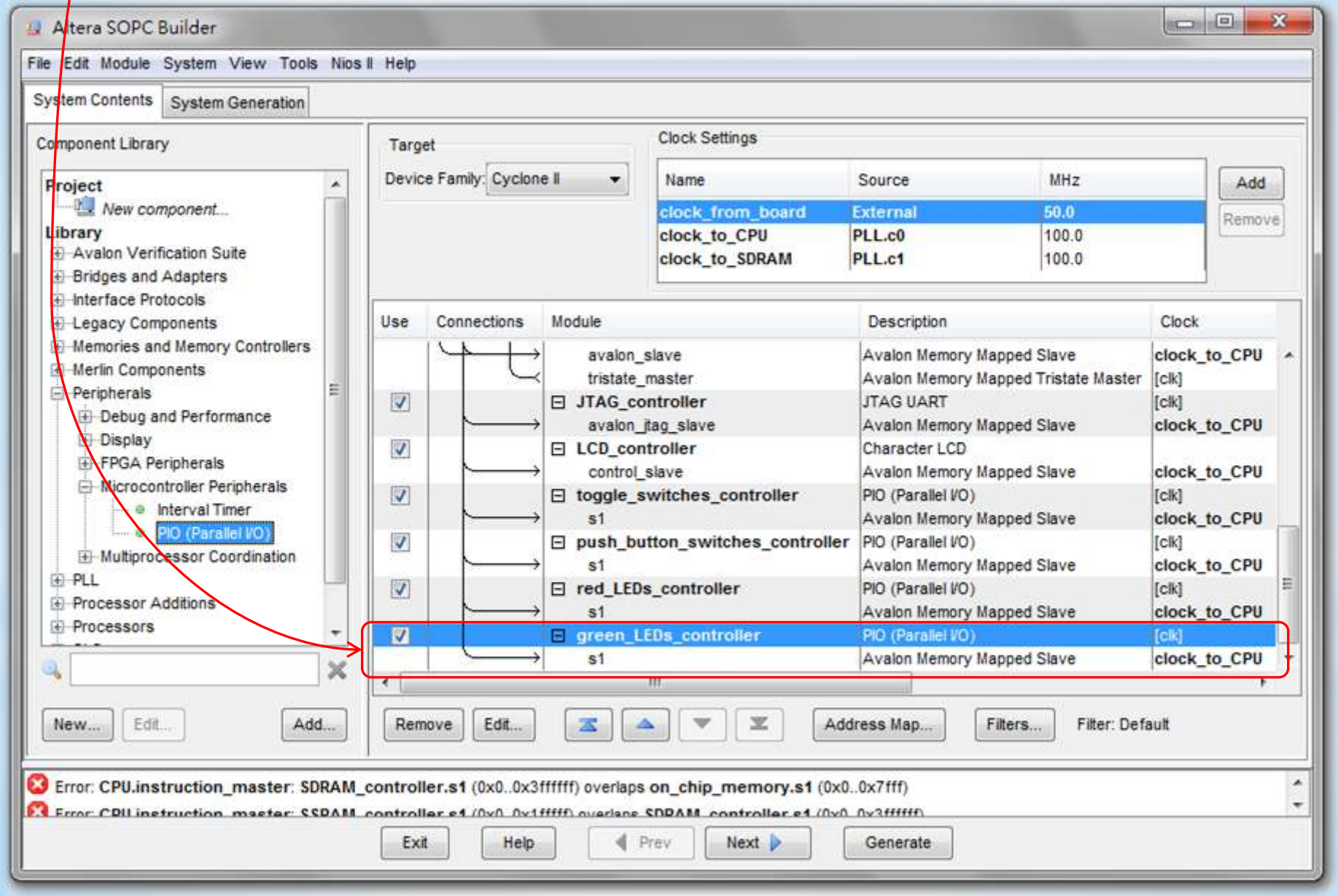
I need the system can control the 8 green LEDs, so I click Library -> Peripherals -> Microcontroller Peripherals -> PIO (Parallel I/O).



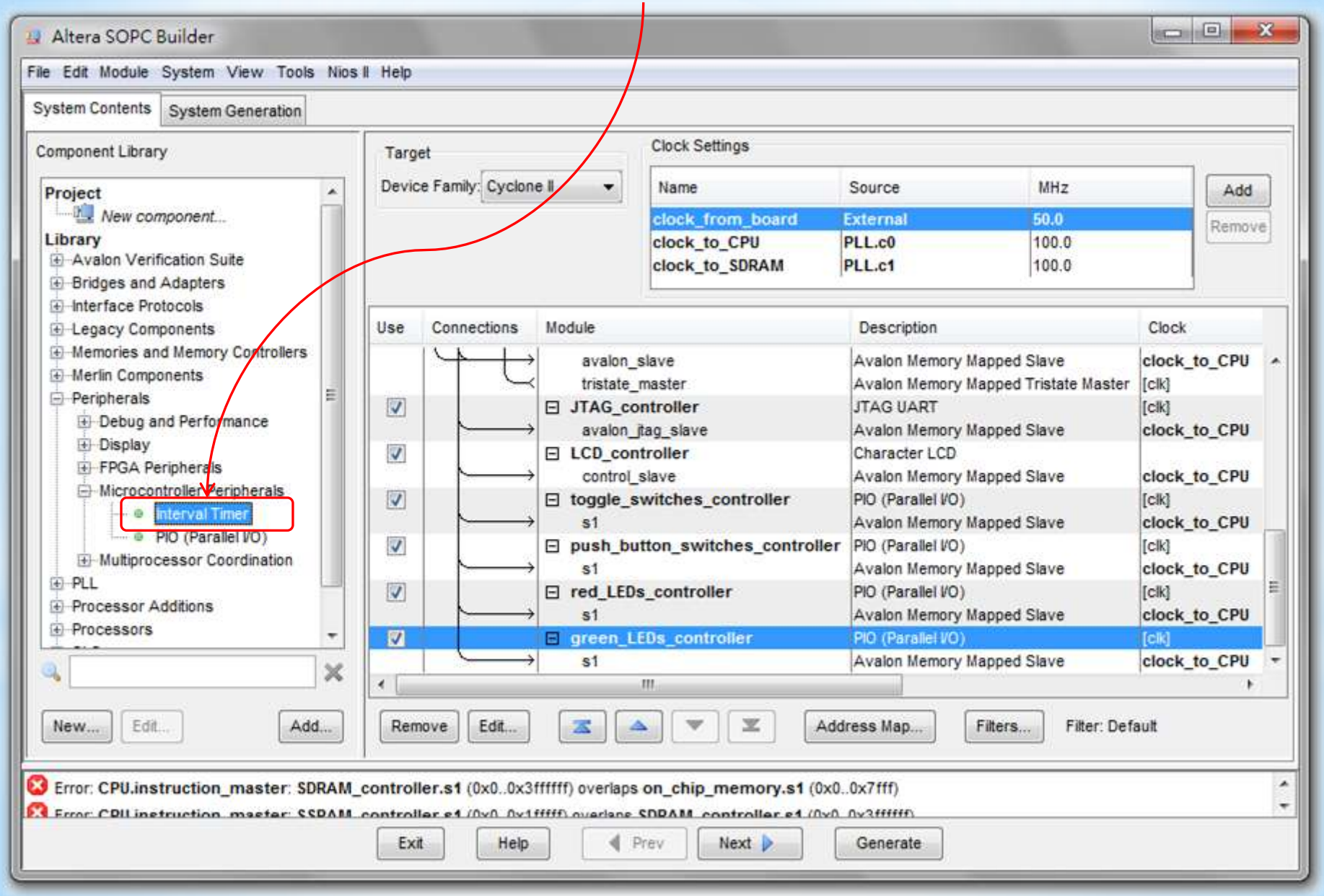
I need to control 8 outputs for green LEDs.



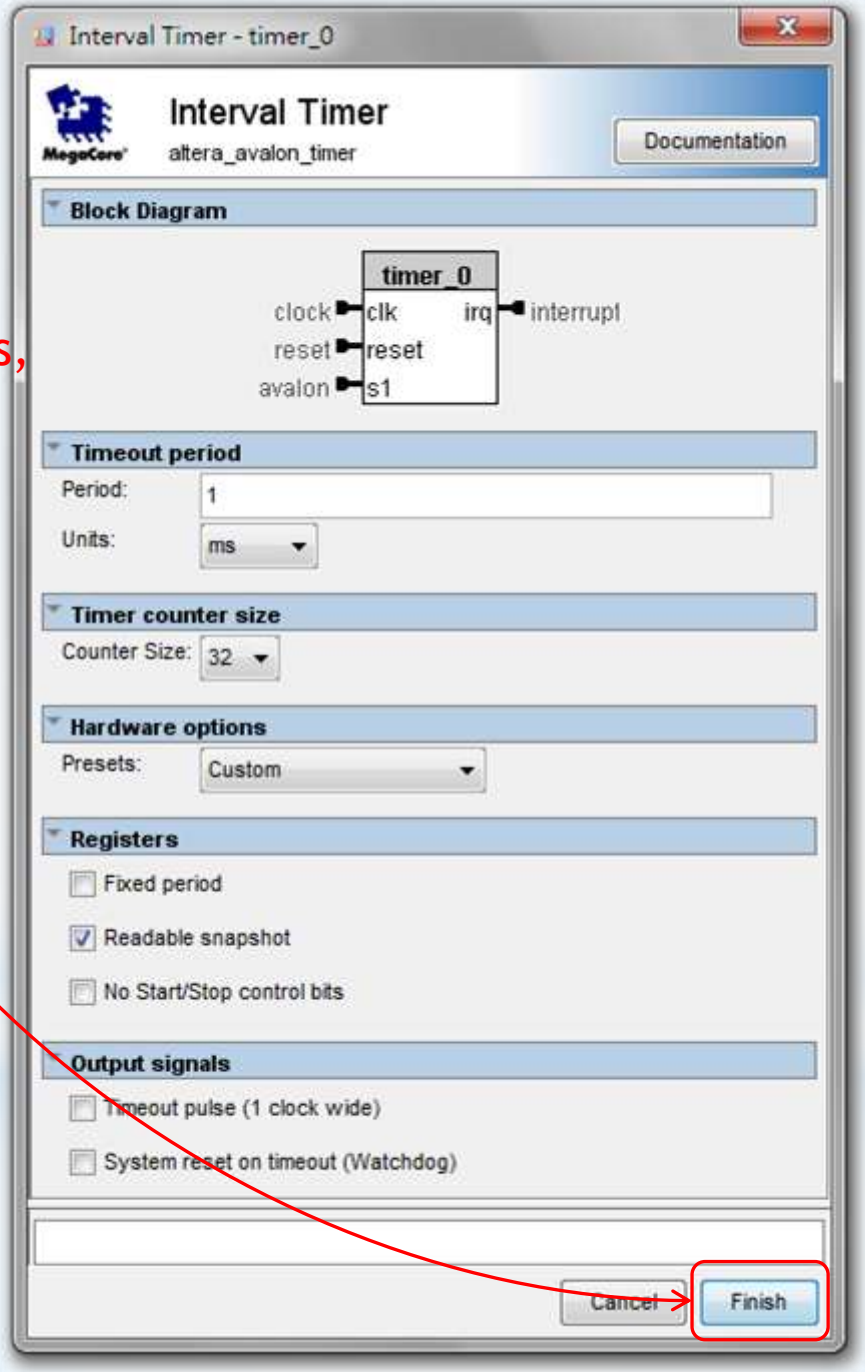
I rename the new controller to green_LEDs_controller with clock_to_CPU.



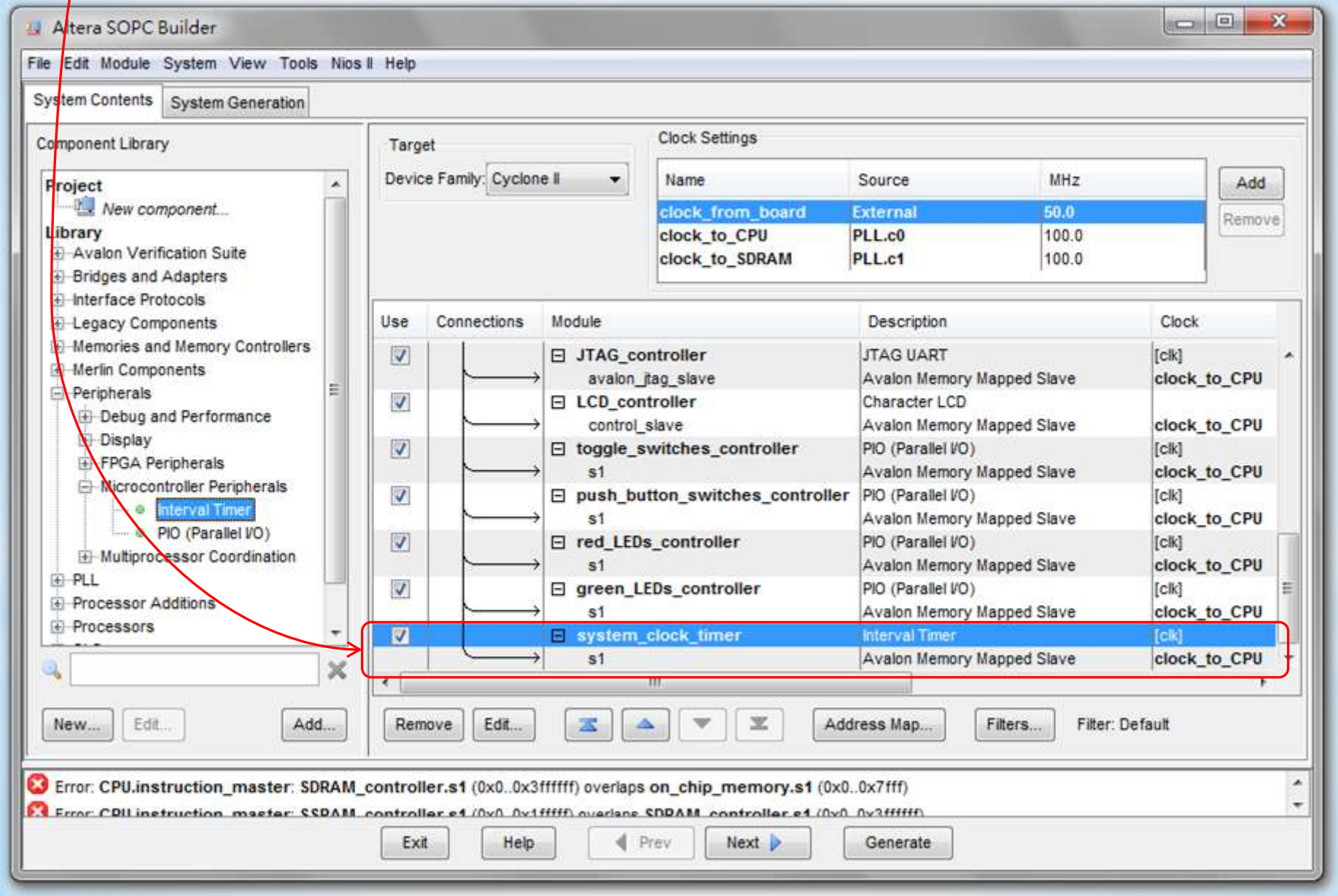
I need to run µC/OS-II, so I click Library -> Peripherals -> Microcontroller Peripherals -> Interval Timer.



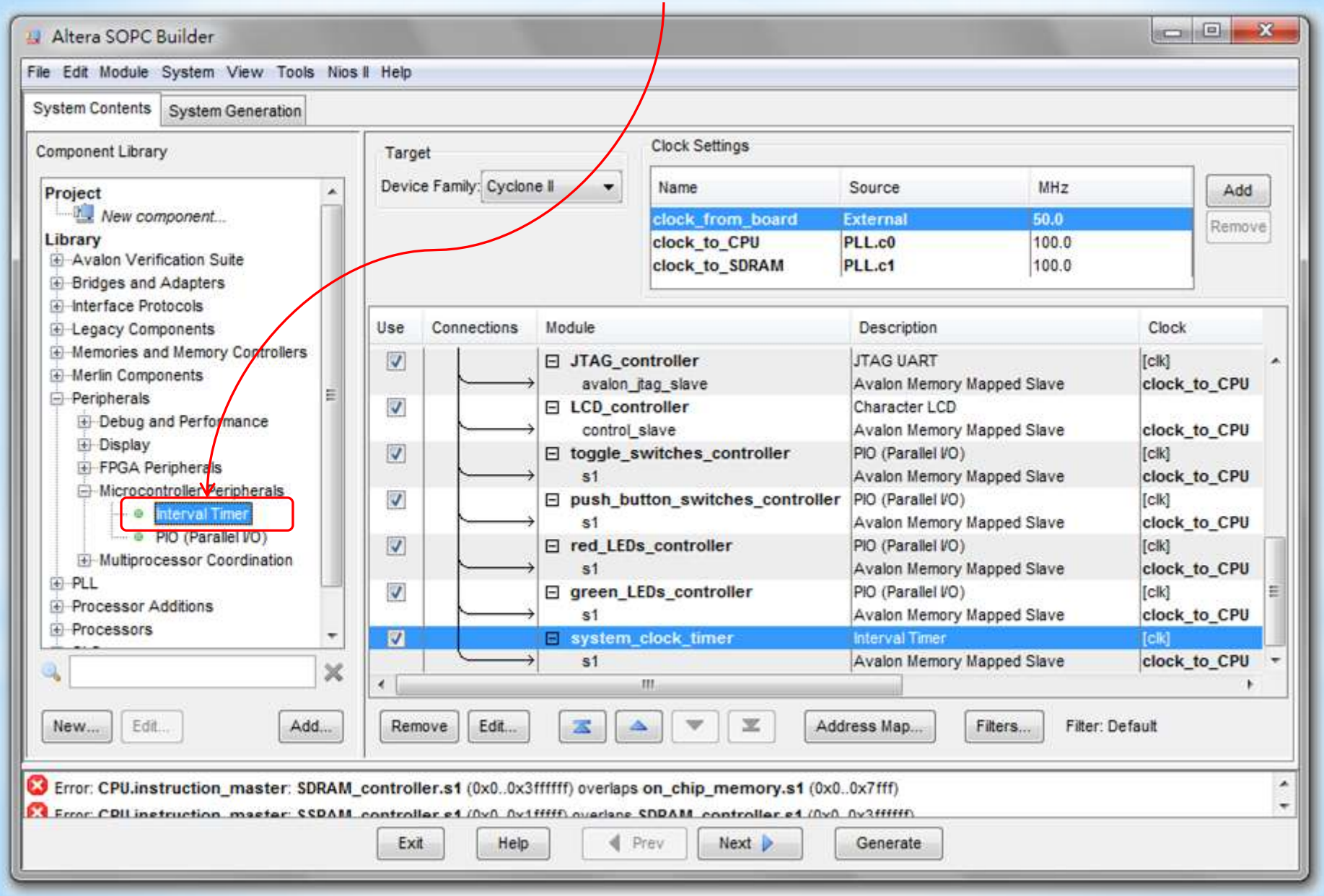
The default value works, so I directly click the finish button.



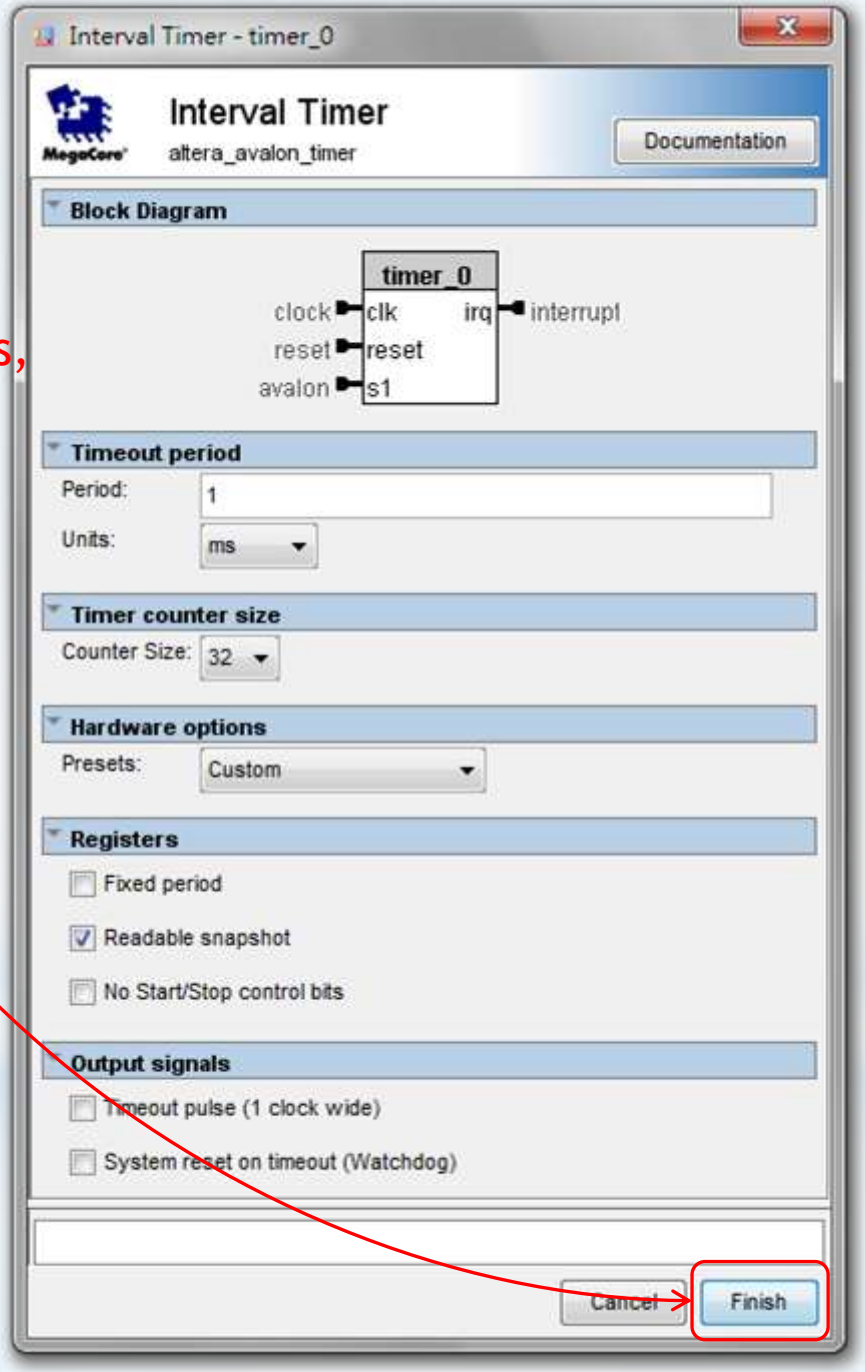
I rename the new timer to system_clock_timer with clock_to_CPU.



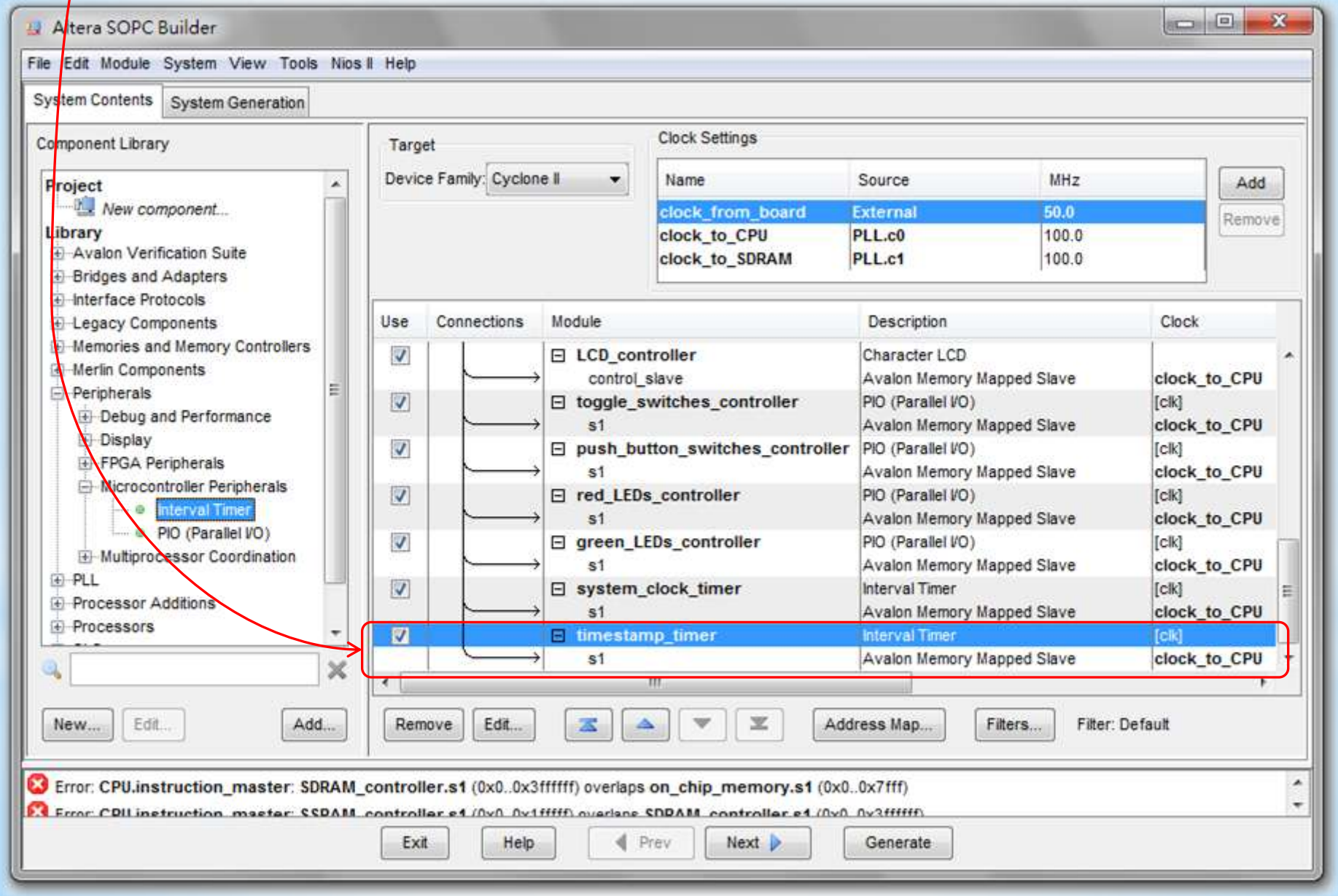
However, µC/OS-II need a timestamp service, so I click Library -> Peripherals -> Microcontroller Peripherals -> Interval Timer.



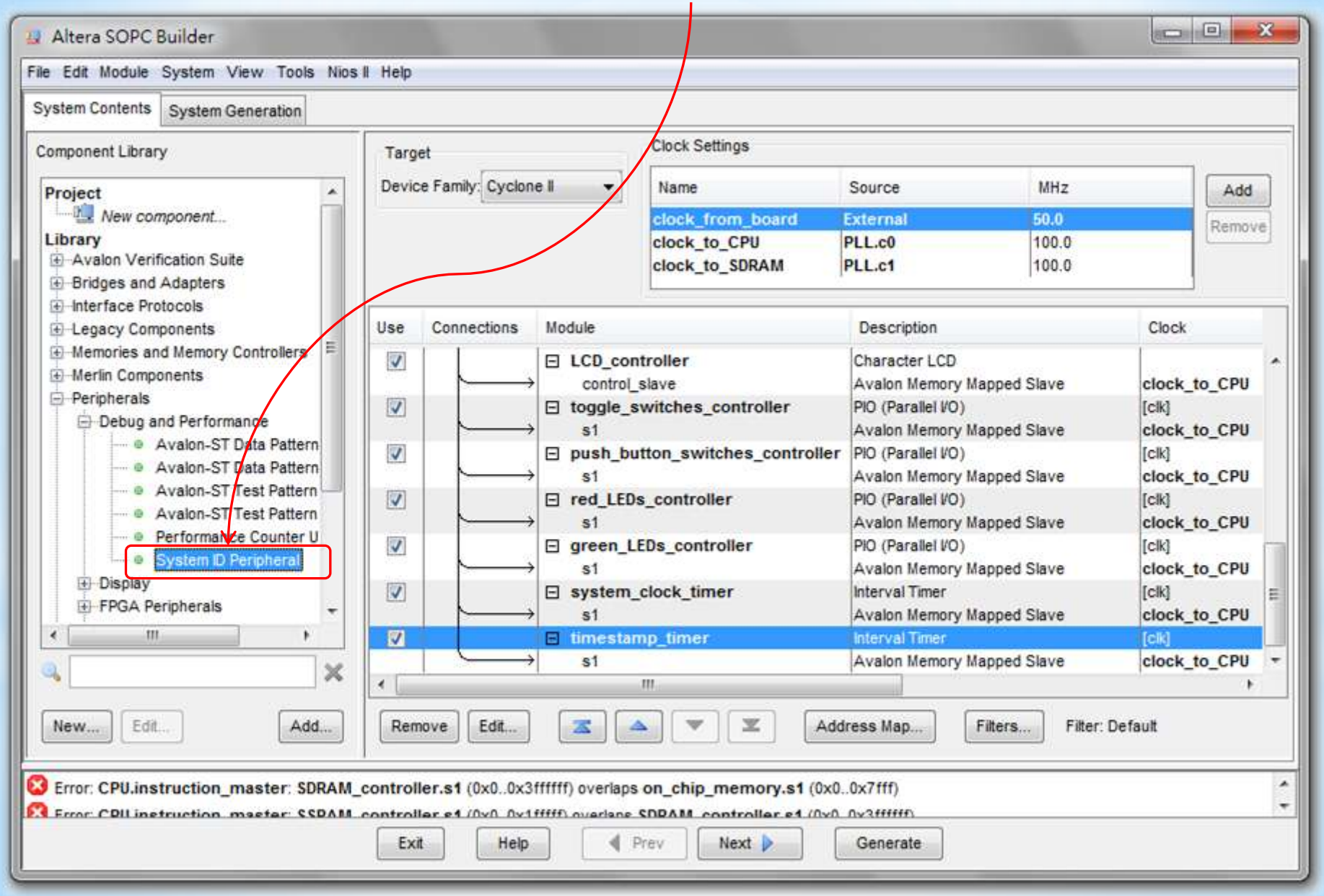
The default value works, so I directly click the finish button.



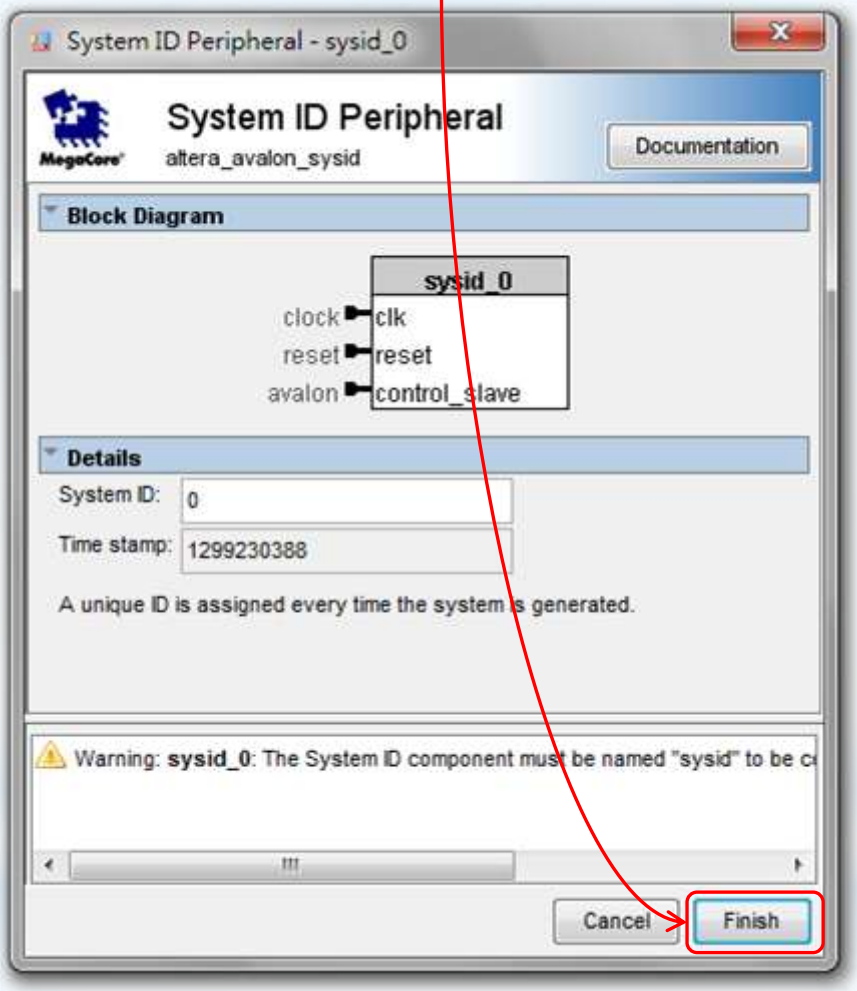
I rename the new timer to timestamp_timer with clock_to_CPU.



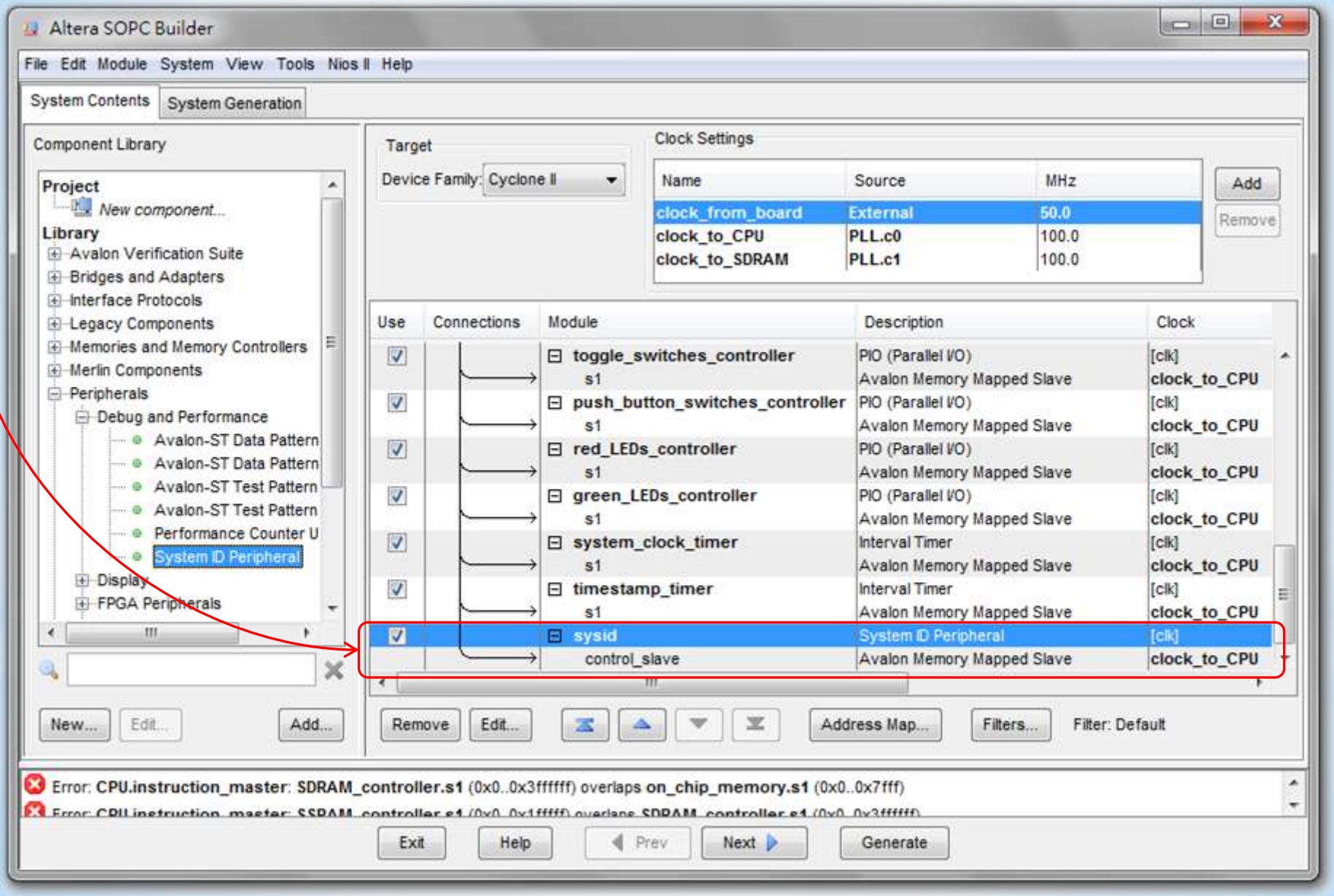
Finally, I need to give my system an unique ID by clicking Library -> Peripherals -> Debug and Performance -> System ID Peripheral.



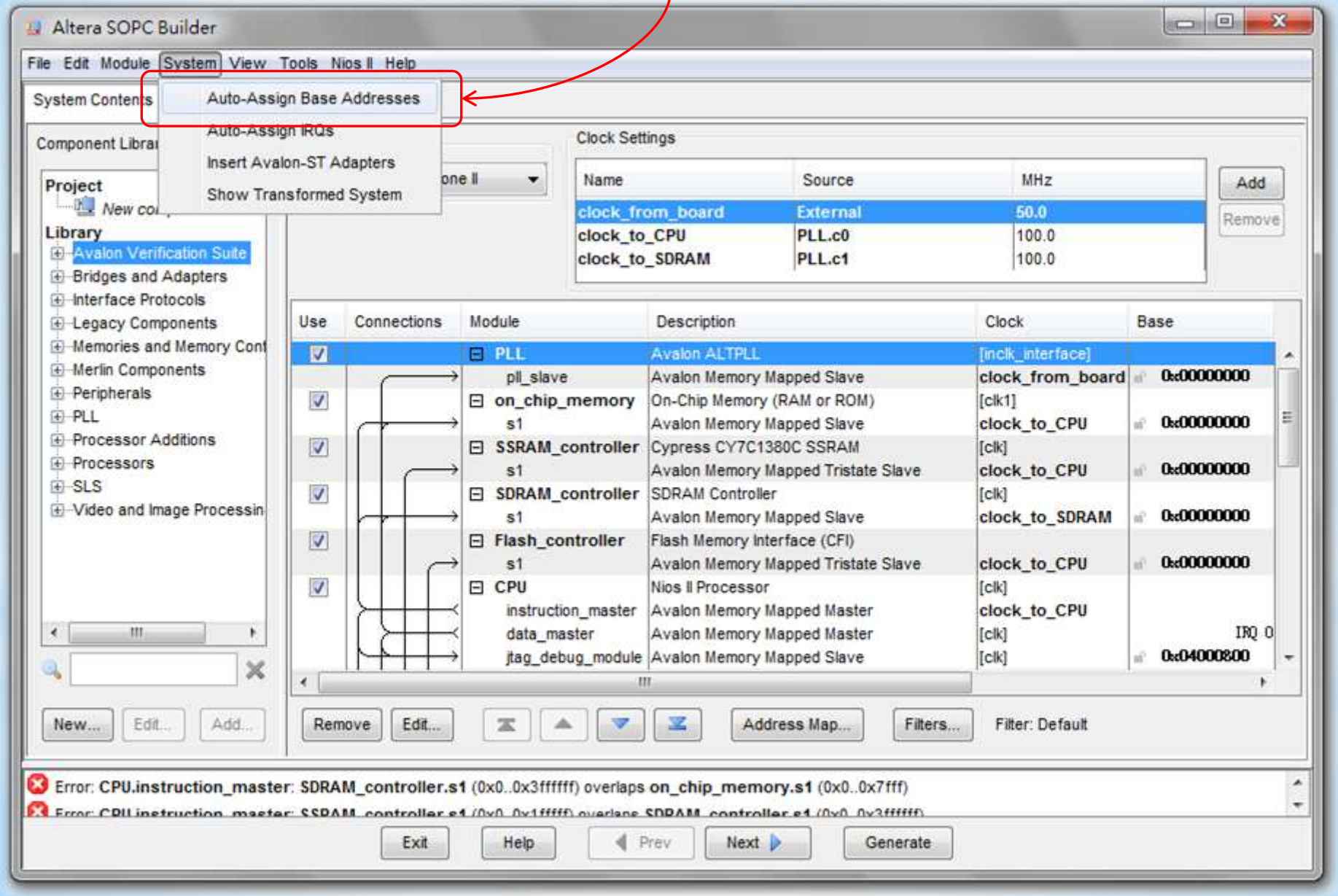
The default value works, so I directly click the finish button.



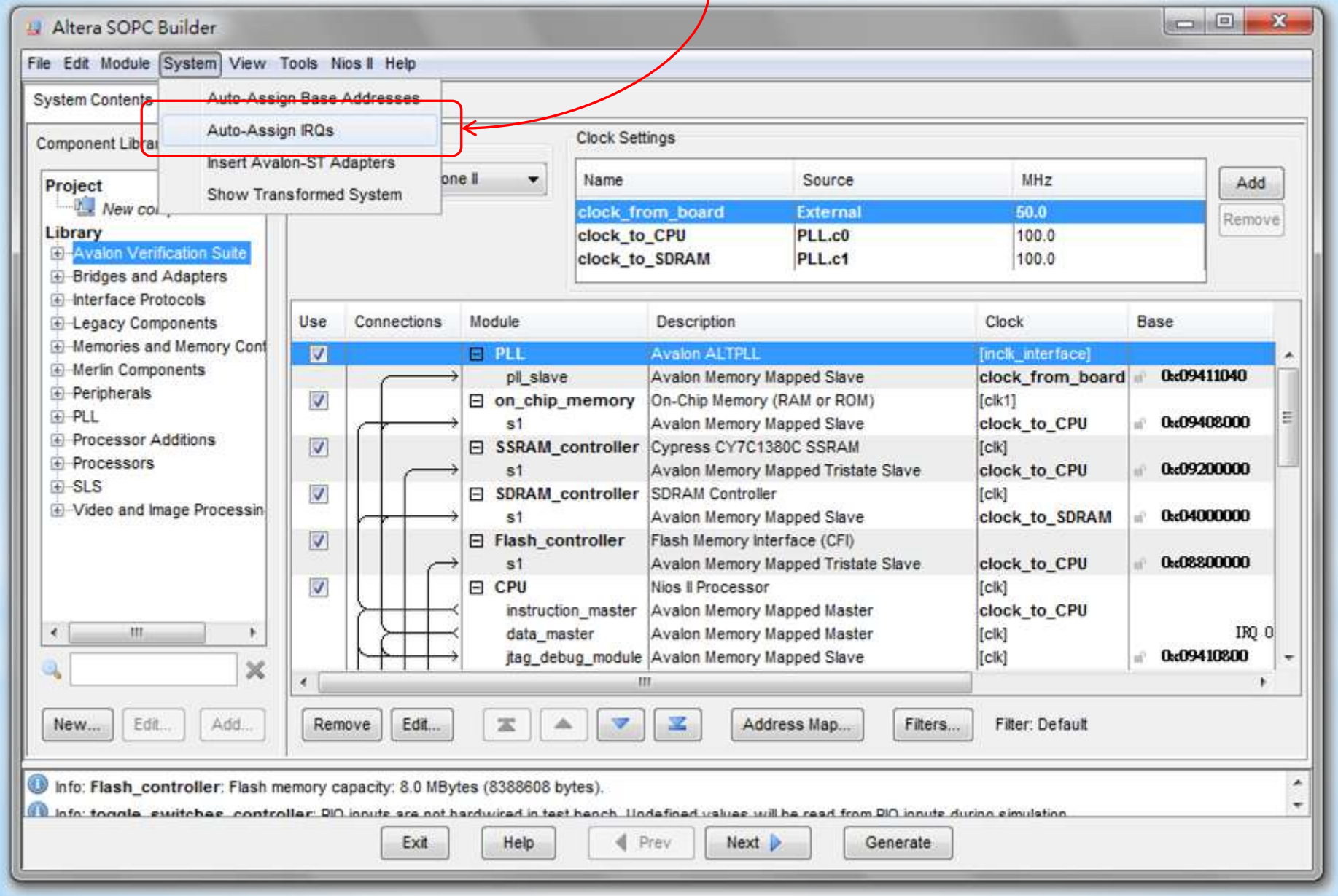
Be careful! Altera Corporation announces that System ID must be renamed to sysid.
You must do exactly the same setting here in your computer.



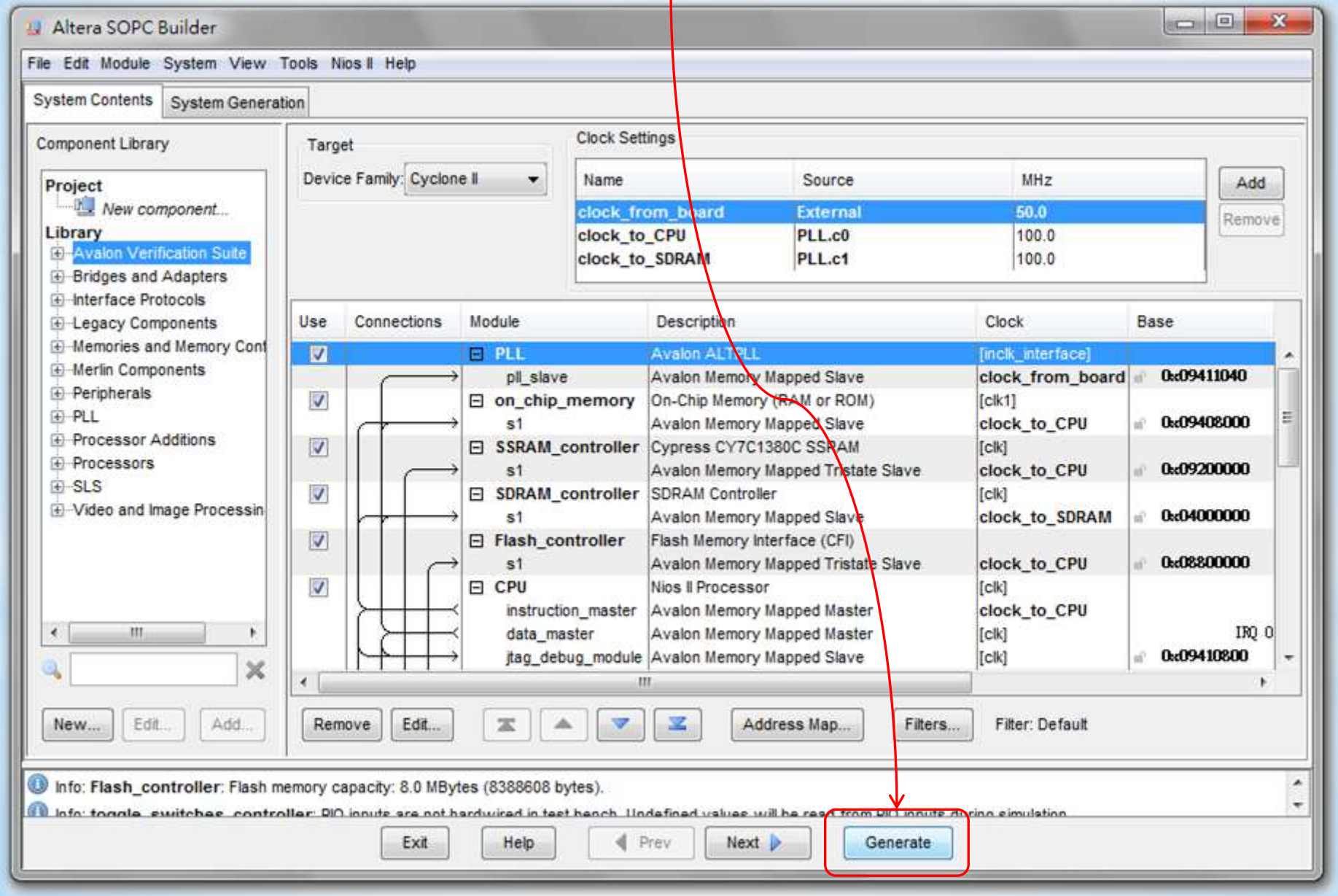
Now I have added all the components on the bus. I need to re-arrange the bus addresses of each components, so I click System -> Auto-Assign Base Addresses.



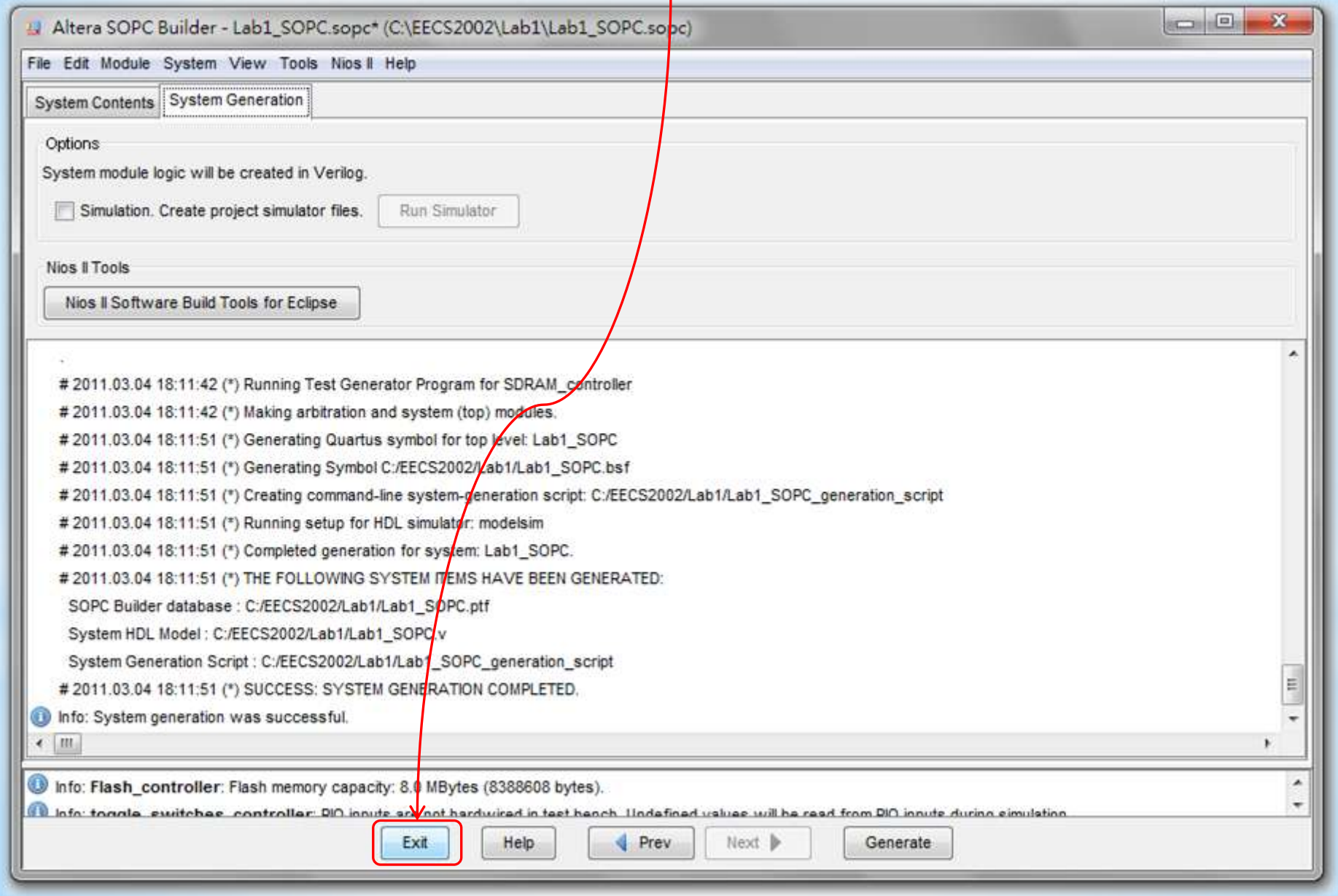
Similarly, the interrupt request (IRQ) number must be re-arranged, too, so I click System -> Auto-Assign IRQs.



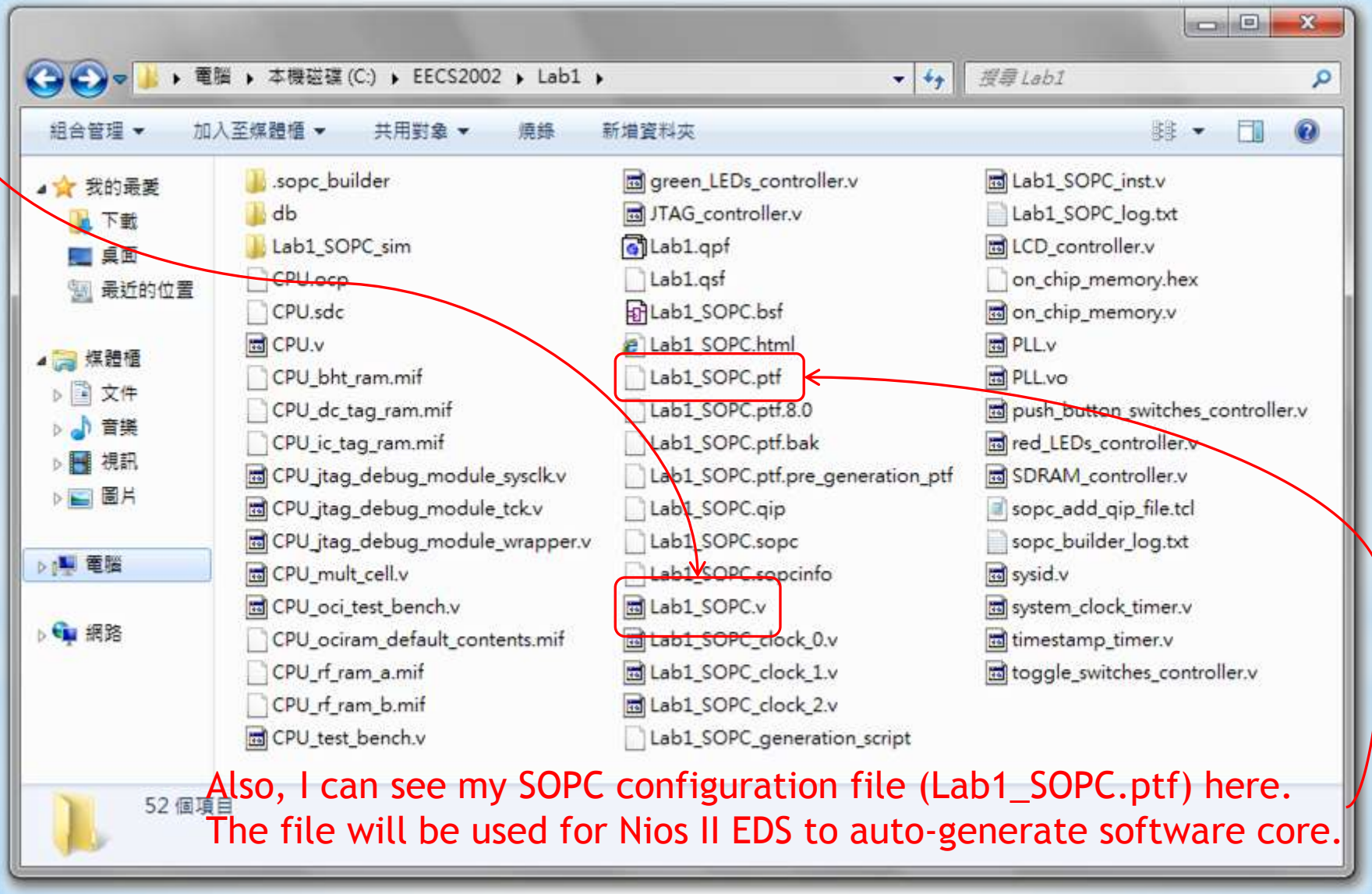
Finally, I've done all the design of SOPC module. I click the generate button to auto-generate all Verilog/VHDL files. (this process may take a few minutes)



Now SOPC Builder has generated all Verilog/VHDL files of my SOPC module.
I click the exit button to close SOPC Builder.

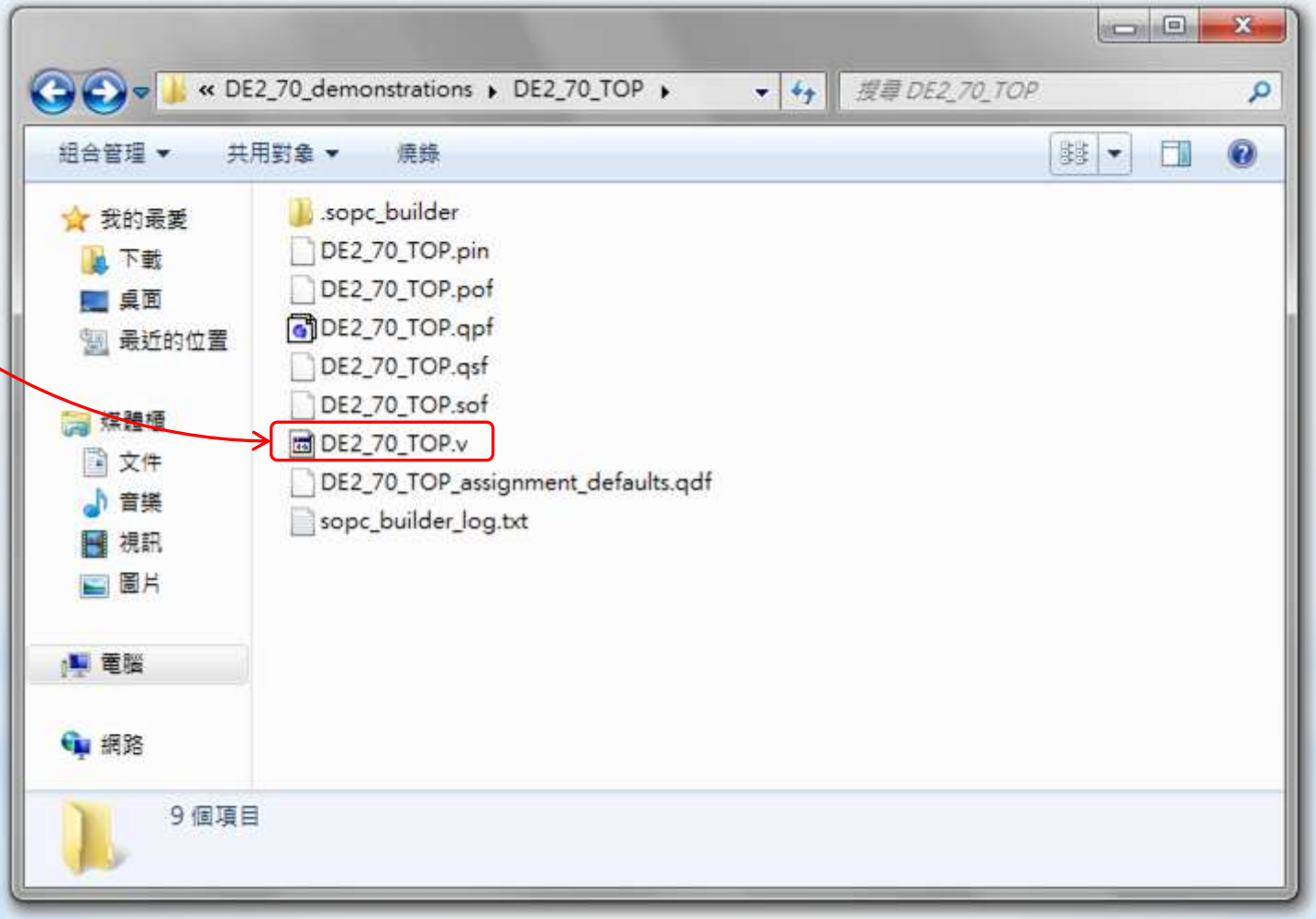


Now I can find my SOPC module (Lab1_SOPC.v) in my project (C:\EECS2002\Lab1\).
However, I can't directly use SOPC module (Lab1_SOPC.v) as top module (Lab1.v).
I need to manually write a top module to include my SOPC module .

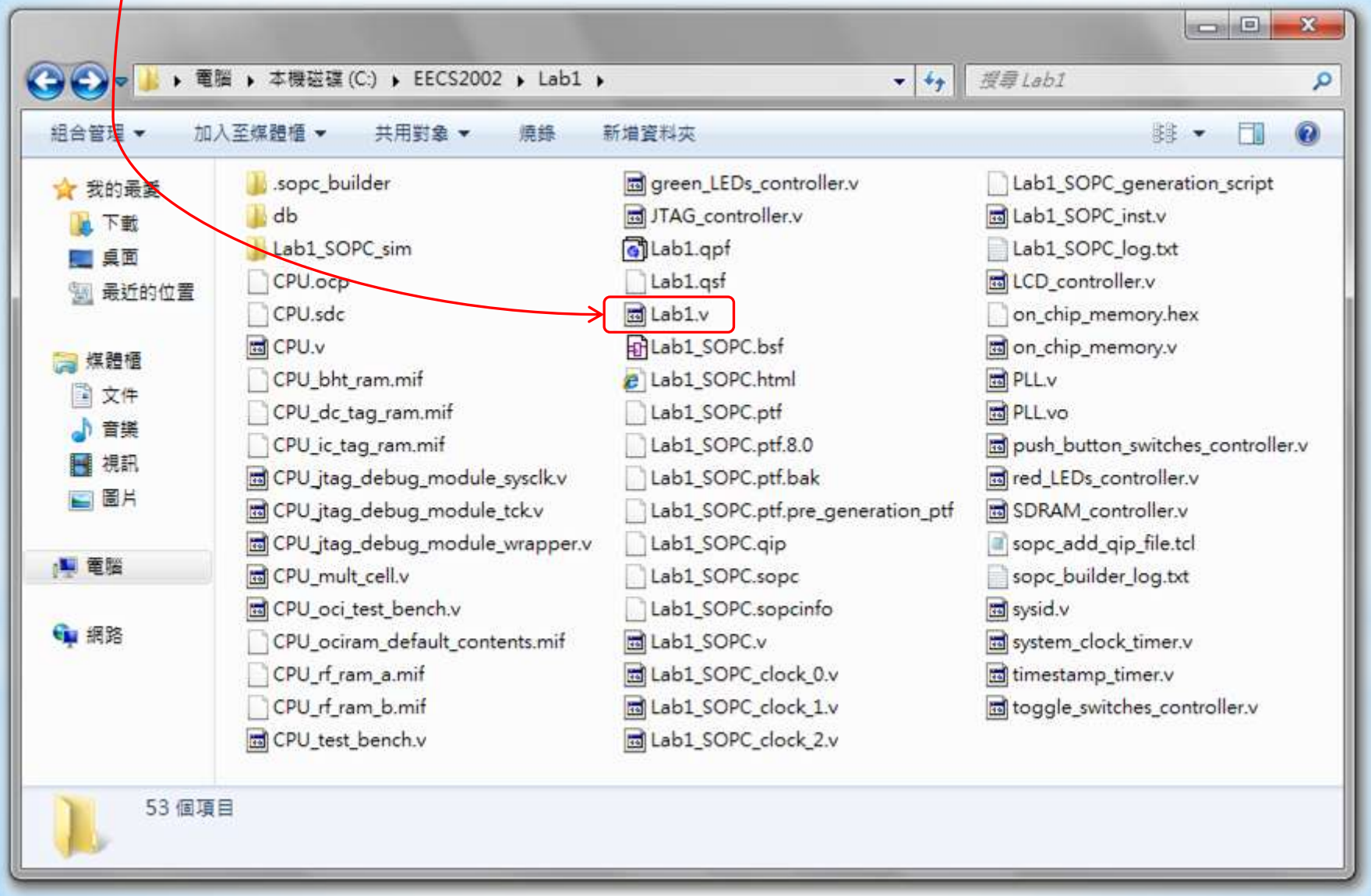


Also, I can see my SOPC configuration file (Lab1_SOPC.ptf) here.
The file will be used for Nios II EDS to auto-generate software core.

Because I don’t know how many I/O pins are used in my Terasic DE2-70 Board, I need to check DE2_70_TOP Example Project in my Terasic DE2-70 CD-ROM. This project contains an empty top module (DE2_70_TOP.v) with all I/O pins.



I copy this DE2_70_TOP.v to my project (C:\EECS2002\Lab1\)
and rename DE2_70_TOP.v to Lab1.v as my top module.



I open Lab1.v and rename the top module name to Lab1.

```
// Revision History :
// --------------------------------------------------------------------
//   Ver  :| Author            :| Mod. Date :| Changes Made:
//   V1.0 :| Johnny FAN        :| 07/07/09  :| Initial Revision
//   V1.1 :| Johnny FAN        :| 07/07/09  :| 1.change fpga device from
//                                              ep2c70896c8 to ep2c70896c6
//                                           2.change module name from DE2P_TOP
//                                              to DE2_70_TOP
//   V1.2 :| Johnny FAN        :| 07/07/09  :| 1. change for pcb v1.1
//                                           2. modify PS2 interface i/o
//                                           3. modify GPIO,VGA, and UART interface
//                                             pin assignment
//                                           4. change GPIO_CLKOUT from output to inout
//   V1.21 :| Johnny FAN       :| 07/07/09  :| 1.remove ssram address bus oSRAM_A[19]and oSRAM_A[20]
//                                           2.remove flash address bus oFLASH_A[25:22]
// --------------------------------------------------------------------

module Lab1
    (
      ////////////////////// Clock Input     //////////////////////
      iCLK_28,                    //  28.63636 MHz
      iCLK_50,                    // 50 MHz
      iCLK_50_2,                     // 50 MHz
      iCLK_50_3,                     // 50 MHz
      iCLK_50_4,                     // 50 MHz
      iEXT_CLOCK,                    // External Clock
```

I find the REG/WIRE declarations in Lab1.v and add the following codes.

```
///////////////////////////////////////////////////////////////////
//=================================================================
// REG/WIRE declarations
//=================================================================

// Define CPU clock pin from SOPC_Lab1 (100MHz, no phase delay)
wire cpu_clk;

// Define the first Push-Button Switch as Reset Button to SOPC_Lab1
reg cpu_reset_n;

// For safety, the length of reset signel is extended to 2 ^ 24 cycles
reg [23:0] counter;
always @ (posedge iCLK_50 or negedge iKEY[0]) begin
   if (~iKEY[0]) begin
      cpu_reset_n <= 1'b0;
      counter <= 24'h000000;
   end else if (counter != 24'hFFFFFF) begin
      cpu_reset_n <= 1'b0;
      counter <= counter + 24'h000001;
   end else begin
      cpu_reset_n <= 1'b1;
      counter <= counter;
   end
end

// Assign pins for real SSRAM I/O
wire [1:0] SRAM_DUMMY_ADDR;
assign oSRAM_ADSP_N = 1'b1;
assign oSRAM_ADV_N = 1'b1;
assign oSRAM_CE2 = ~oSRAM_CE1_N;
assign oSRAM_CE3_N = oSRAM_CE1_N;
assign oSRAM_GW_N = 1'b1;
assign oSRAM_CLK = cpu_clk;
```

I continue adding the following codes, which uses 1 x 64 MB SDRAM_controller pins for 2 x 32 MB real SDRAM pins on Terasic DE2-70 board.

```
// Define 1 * 64 MB SDRAM Pins for using SDRAM_controller
wire sdram_clk;
wire [12:0] dram_a;
wire [1:0] dram_ba;
wire dram_cas_n;
wire dram_cke;
wire dram_cs_n;
wire [3:0] dram_dqm;
wire dram_ras_n;
wire dram_we_n;

// Convert 1 * 64 MB to 2 * 32 MB SDRAM pins for real SDRAM I/O
assign oDRAM0_CLK = sdram_clk;
assign oDRAM1_CLK = sdram_clk;
assign oDRAM0_A = dram_a;
assign oDRAM1_A = dram_a;
assign oDRAM0_BA = dram_ba;
assign oDRAM1_BA = dram_ba;
assign oDRAM0_CAS_N = dram_cas_n;
assign oDRAM1_CAS_N = dram_cas_n;
assign oDRAM0_CKE = dram_cke;
assign oDRAM1_CKE = dram_cke;
assign oDRAM0_CS_N = dram_cs_n;
assign oDRAM1_CS_N = dram_cs_n;
assign oDRAM0_LDQM0 = dram_dqm[0];
assign oDRAM0_UDQM1 = dram_dqm[1];
assign oDRAM1_LDQM0 = dram_dqm[2];
assign oDRAM1_UDQM1 = dram_dqm[3];
assign oDRAM0_RAS_N = dram_ras_n;
assign oDRAM1_RAS_N = dram_ras_n;
assign oDRAM0_WE_N = dram_we_n;
assign oDRAM1_WE_N = dram_we_n;
```

I continue adding the following codes to finished the REG/WIRE declarations. Noticed that I've defined all unused pins to tri-state value (high impedance). This is a good coding style because if undefined pins are uncarefully connected to the ground by the wrong setting of Quartus II, massive current will go from the supply voltage to the ground via these ground-connected pins when these pins represents the signal 1 at some time. These massive current through the short path may damage FPGA chip or other components on development board.

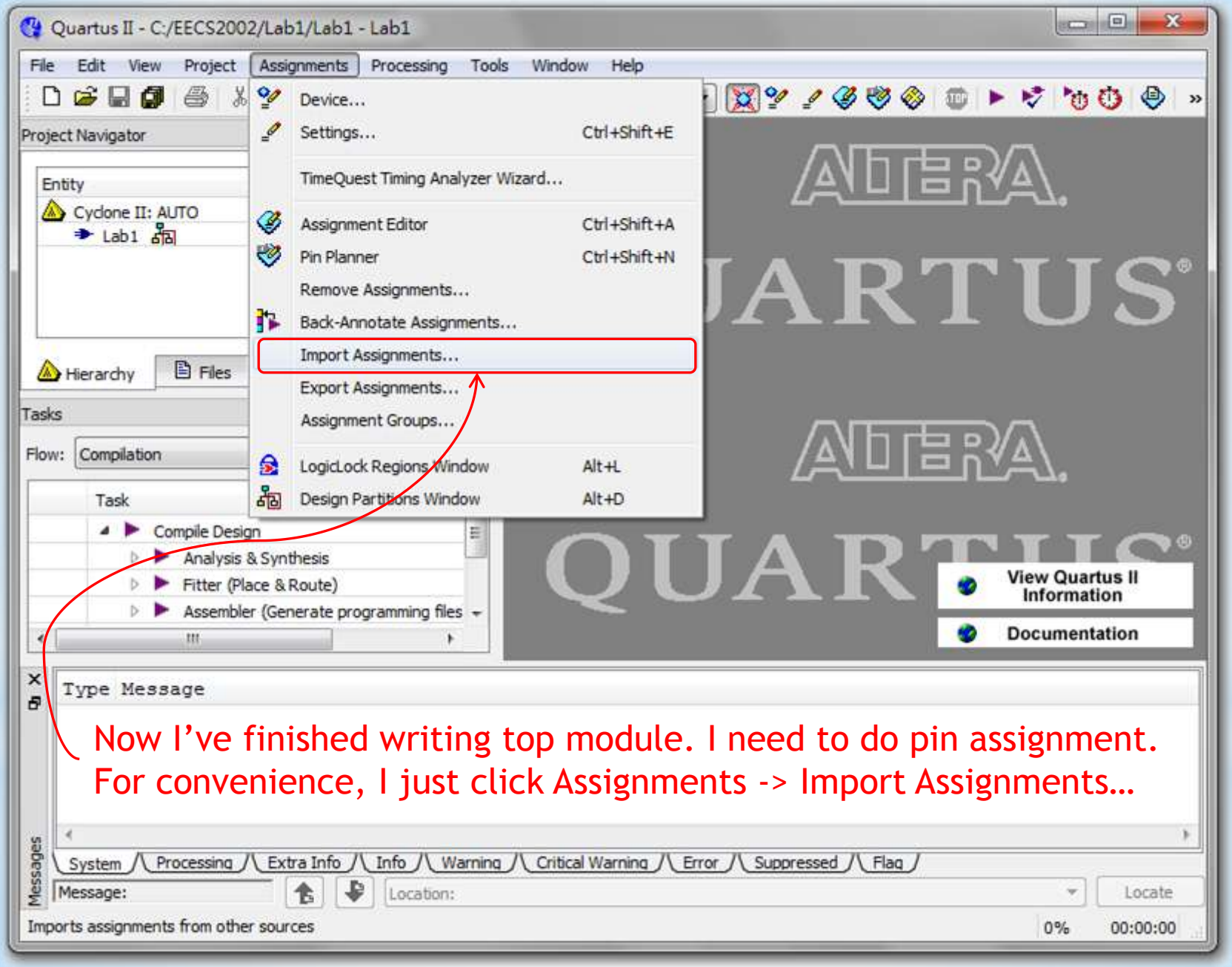
```
// Assign pins for real FLASH I/O
wire FLASH_16BIT_IP_A0;
assign oFLASH_BYTE_N = 1'b1;
assign oFLASH_RST_N = 1'b1;
assign oFLASH_WP_N = 1'b1;

// Assign pins for real character LCD
assign oLCD_ON = 1'b1;
assign oLCD_BLON = 1'b1;

// Assign other unused pins to tristate value
assign SD_DAT = 1'bz;
assign GPIO_0 = 32'hzzzzzzzz;
assign GPIO_1 = 32'hzzzzzzzz;
assign AUD_ADCLRCK = 1'bz;
assign oTD1_RESET_N = 1'bz;
assign oTD2_RESET_N = 1'bz;
assign OTG_FSPEED = 1'bz;
assign OTG_LSPEED = 1'bz;
```

I find the Structural coding in Lab1.v and add the following codes.

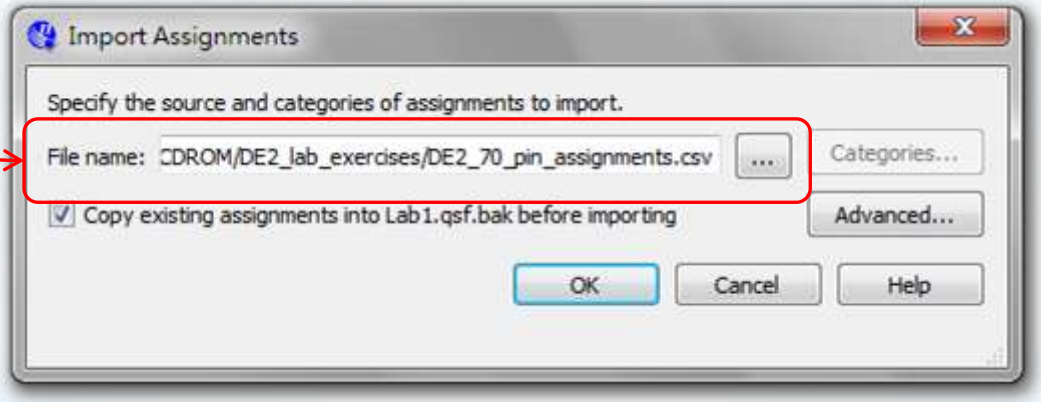
```
//=======================================================================================
//  Structural coding
//=======================================================================================

// Add sopc module (Lab1_SOPC) to top module (Lab1)
Lab1_SOPC sopc_module (

   // For PLL:
   .clock_from_board(iCLK_50),
   .clock_to_CPU(cpu_clk),
   .clock_to_SDRAM(sdram_clk),

   // For SSRAM_controller:
   .address_to_the_SSRAM_controller({oSRAM_A[17:0], SRAM_DUMMY_ADDR}),
   .adsc_n_to_the_SSRAM_controller(oSRAM_ADSC_N),
   .bw_n_to_the_SSRAM_controller(oSRAM_BE_N),
   .bwe_n_to_the_SSRAM_controller(oSRAM_WE_N),
   .chipenable1_n_to_the_SSRAM_controller(oSRAM_CE1_N),
   .data_to_and_from_the_SSRAM_controller(SRAM_DQ),
   .outputenable_n_to_the_SSRAM_controller(oSRAM_OE_N),

   // For SDRAM_controller:
   .zs_addr_from_the_SDRAM_controller(dram_a),
   .zs_ba_from_the_SDRAM_controller(dram_ba),
   .zs_cas_n_from_the_SDRAM_controller(dram_cas_n),
   .zs_cke_from_the_SDRAM_controller(dram_cke),
   .zs_cs_n_from_the_SDRAM_controller(dram_cs_n),
   .zs_dq_to_and_from_the_SDRAM_controller(DRAM_DQ),
   .zs_dqm_from_the_SDRAM_controller(dram_dqm),
   .zs_ras_n_from_the_SDRAM_controller(dram_ras_n),
   .zs_we_n_from_the_SDRAM_controller(dram_we_n),
```

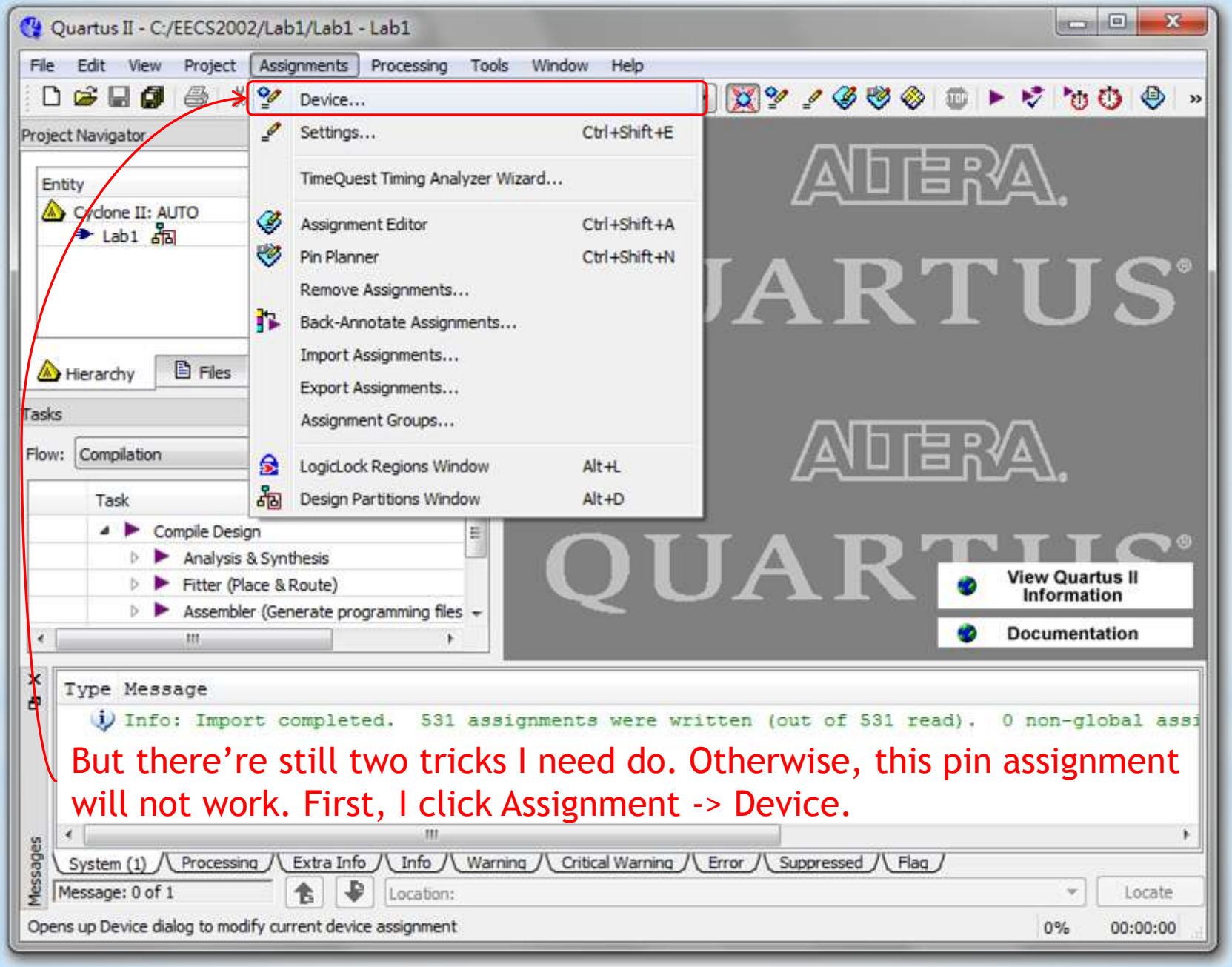
I continue adding the following codes to finished the Structural coding.
Remember that I use cpu_reset_n (extended signal of iKEY[0]) as reset signal.

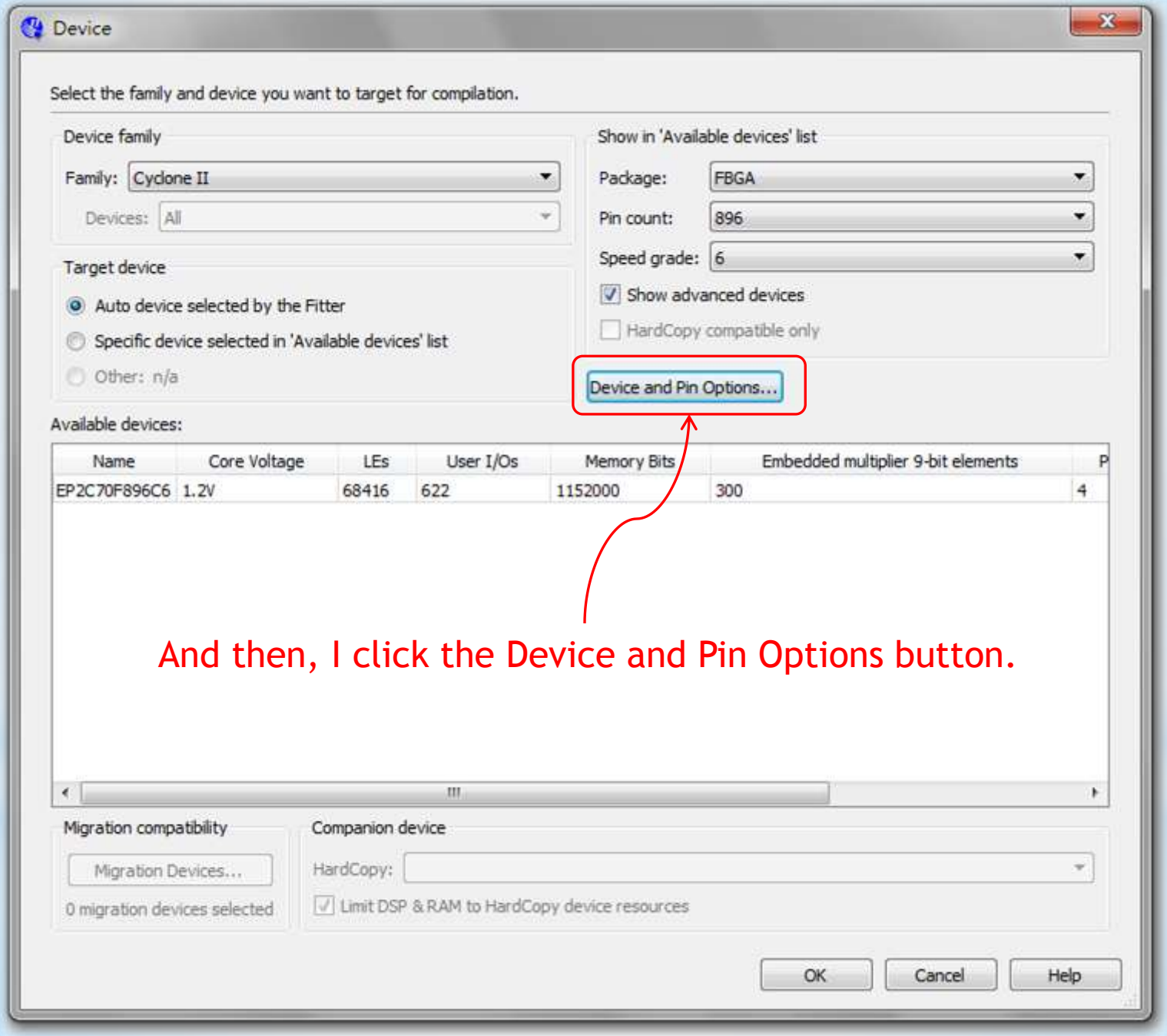
```
    // For Flash_controller:
    .address_to_the_Flash_controller({oFLASH_A[21:0], FLASH_16BIT_IP_A0}),
    .data_to_and_from_the_Flash_controller({FLASH_DQ15_AM1, FLASH_DQ}),
    .read_n_to_the_Flash_controller(oFLASH_OE_N),
    .select_n_to_the_Flash_controller(oFLASH_CE_N),
    .write_n_to_the_Flash_controller(oFLASH_WE_N),

    // For LCD_controller:
    .LCD_E_from_the_LCD_controller(oLCD_EN),
    .LCD_RS_from_the_LCD_controller(oLCD_RS),
    .LCD_RW_from_the_LCD_controller(oLCD_RW),
    .LCD_data_to_and_from_the_LCD_controller(LCD_D),

    // For toggle_switches_controller:
    .in_port_to_the_toggle_switches_controller(iSW),

    // For push_button_switches_controller:
    .in_port_to_the_push_button_switches_controller(iKEY),

    // For red_LEDs_controller:
    .out_port_from_the_red_LEDs_controller(oLEDR),

    // For green_LEDs_controller:
    .out_port_from_the_green_LEDs_controller(oLEDG),

    // System reset:
    .reset_n(cpu_reset_n)
);
```

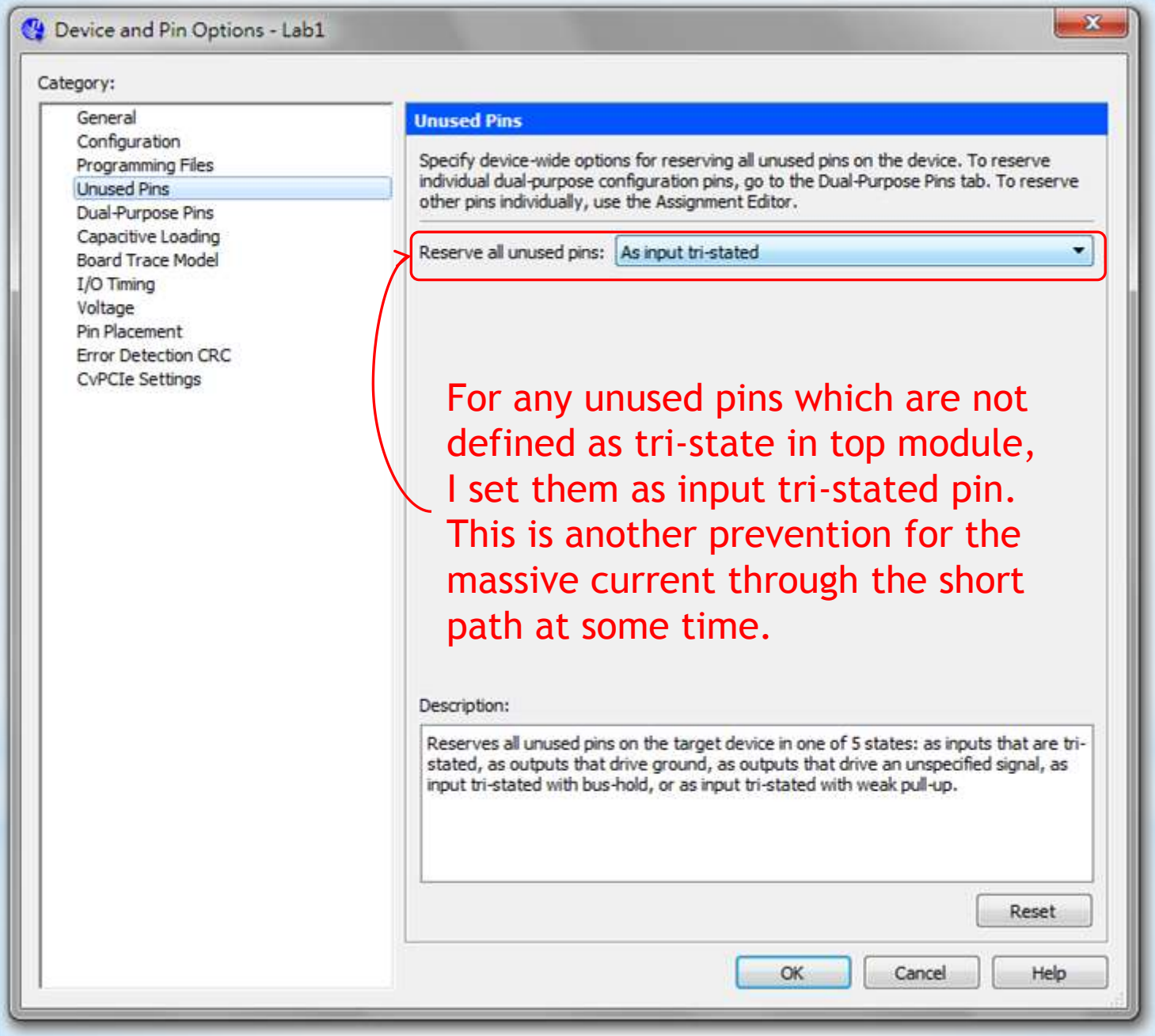

Quartus II - C:/EECS2002/Lab1/Lab1 - Lab1
File Edit View Project Assignments Processing Tools Window Help
Device...
Settings... Ctrl+Shift+E
TimeQuest Timing Analyzer Wizard...
Assignment Editor Ctrl+Shift+A
Pin Planner Ctrl+Shift+N
Remove Assignments...
Back-Annotate Assignments...
Import Assignments...
Export Assignments...
Assignment Groups...
LogicLock Regions Window Alt+L
Design Partitions Window Alt+D
Project Navigator
Entity
Cyclone II: AUTO
Lab1
Hierarchy
Files
Tasks
Flow: Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files
ALTERA
QUARTUS
View Quartus II Information
Documentation
Type Message
Now I've finished writing top module. I need to do pin assignment.
For convenience, I just click Assignments -> Import Assignments...
System Processing Extra Info Info Warning Critical Warning Error Suppressed Flag
Message:
Location:
Locate
Messages
Imports assignments from other sources
0%
00:00:00

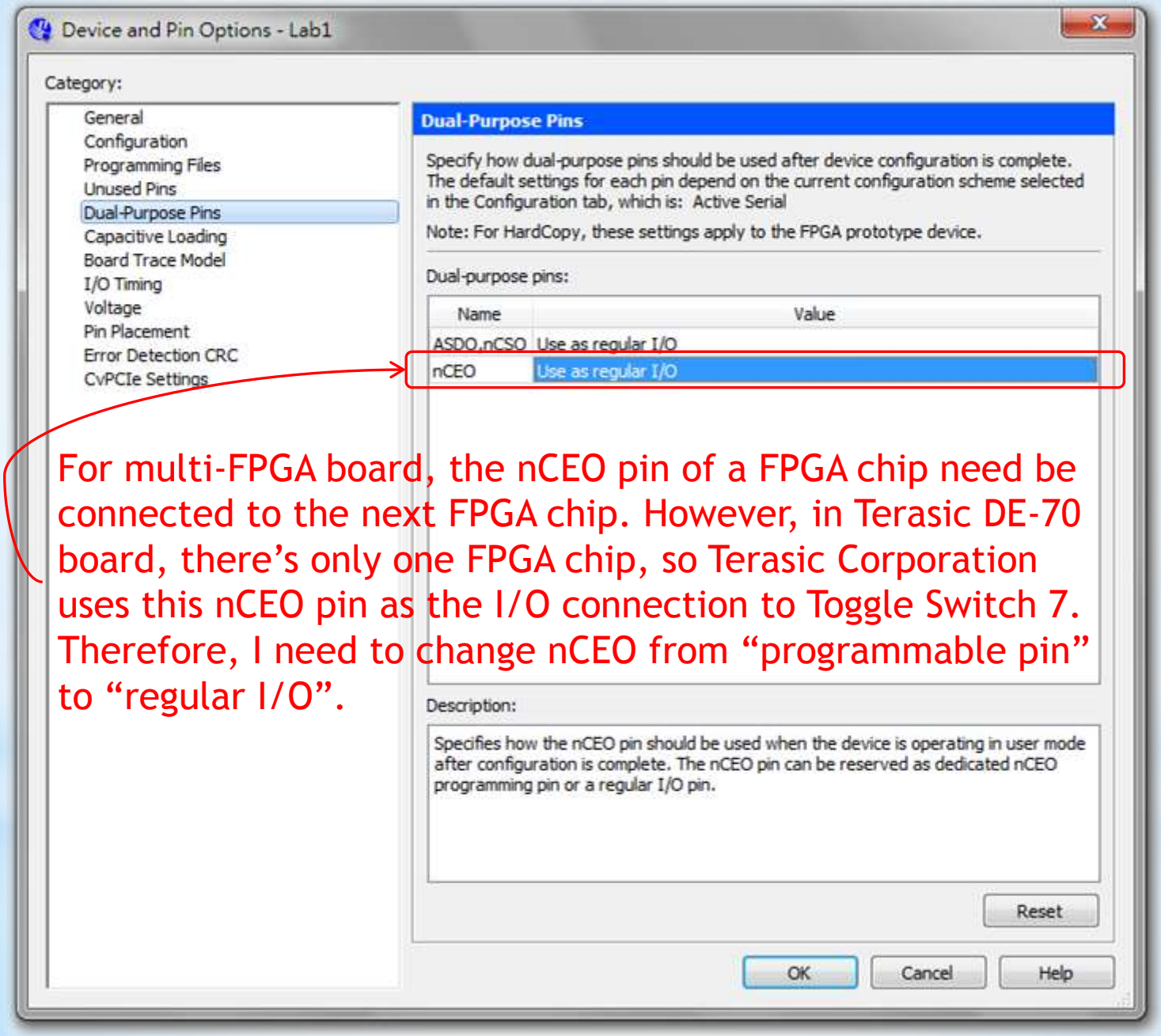
Because I don't know the positions of each I/O pins on FPGA chip, I need to check my Terasic DE2-70 CD-ROM again. There's also a pin table file named DE2_70_pin_assignments.csv (in DE2_lab_exercises folder) which maps all I/O pins positions according to the example top module (DE2_70_TOP.v). Luckily, my top module (Lab1.v) uses exactly the same I/O pins, so I simply import this DE2_70_pin_assignments.csv into my project.



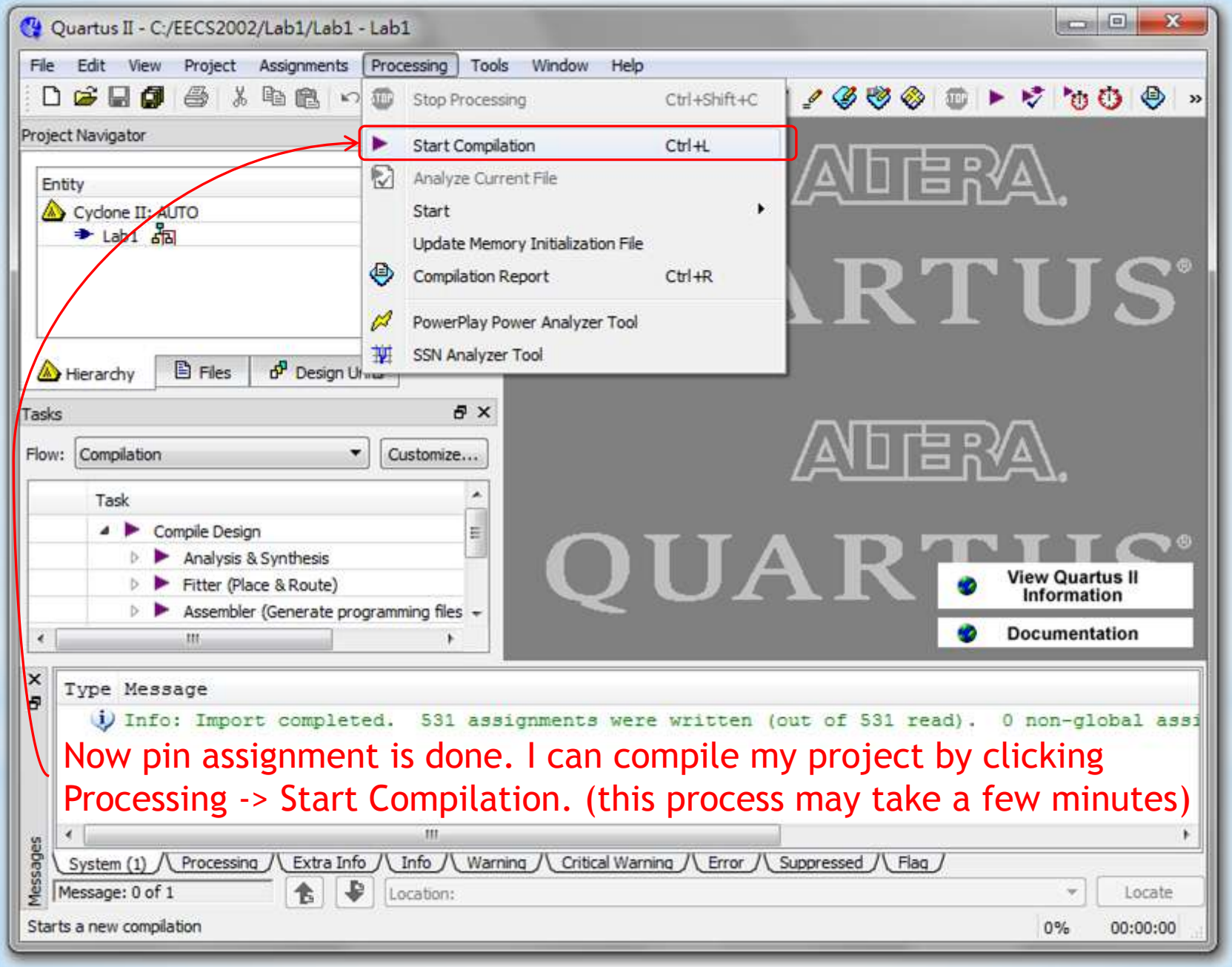

Quartus II - C:/EECS2002/Lab1/Lab1 - Lab1
File Edit View Project Assignments Processing Tools Window Help
Device...
Settings... Ctrl+Shift+E
TimeQuest Timing Analyzer Wizard...
Assignment Editor Ctrl+Shift+A
Pin Planner Ctrl+Shift+N
Remove Assignments...
Back-Annotate Assignments...
Import Assignments...
Export Assignments...
Assignment Groups...
LogicLock Regions Window Alt+L
Design Partitions Window Alt+D
Project Navigator
Entity
Cyclone II: AUTO
Lab1
Hierarchy
Files
Tasks
Flow: Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files
ALTERA
QUARTUS
View Quartus II Information
Documentation
Type Message
Info: Import completed. 531 assignments were written (out of 531 read). 0 non-global assi
But there're still two tricks I need do. Otherwise, this pin assignment will not work. First, I click Assignment -> Device.
Messages
System (1) Processing Extra Info Info Warning Critical Warning Error Suppressed Flag
Message: 0 of 1
Location:
Locate
Opens up Device dialog to modify current device assignment
0% 00:00:00


Device
Select the family and device you want to target for compilation.
Device family
Family: Cyclone II
Devices: All
Target device
Auto device selected by the Fitter
Specific device selected in 'Available devices' list
Other: n/a
Show in 'Available devices' list
Package: FBGA
Pin count: 896
Speed grade: 6
Show advanced devices
HardCopy compatible only
Device and Pin Options...
Available devices:
Name
Core Voltage
LEs
User I/Os
Memory Bits
Embedded multiplier 9-bit elements
P
EP2C70F896C6
1.2V
68416
622
1152000
300
4
And then, I click the Device and Pin Options button.
Migration compatibility
Migration Devices...
0 migration devices selected
Companion device
HardCopy:
Limit DSP & RAM to HardCopy device resources
OK
Cancel
Help

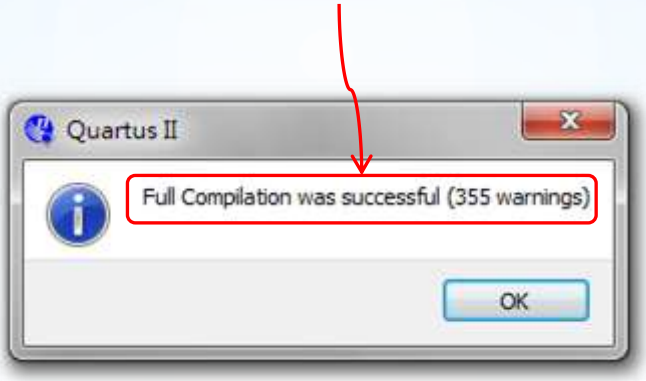

Device and Pin Options - Lab1
Category:
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC
CvPCIe Settings
Unused Pins
Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Editor.
Reserve all unused pins:
As input tri-stated
For any unused pins which are not defined as tri-state in top module, I set them as input tri-stated pin. This is another prevention for the massive current through the short path at some time.
Description:
Reserves all unused pins on the target device in one of 5 states: as inputs that are tri-stated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up.
Reset
OK
Cancel
Help



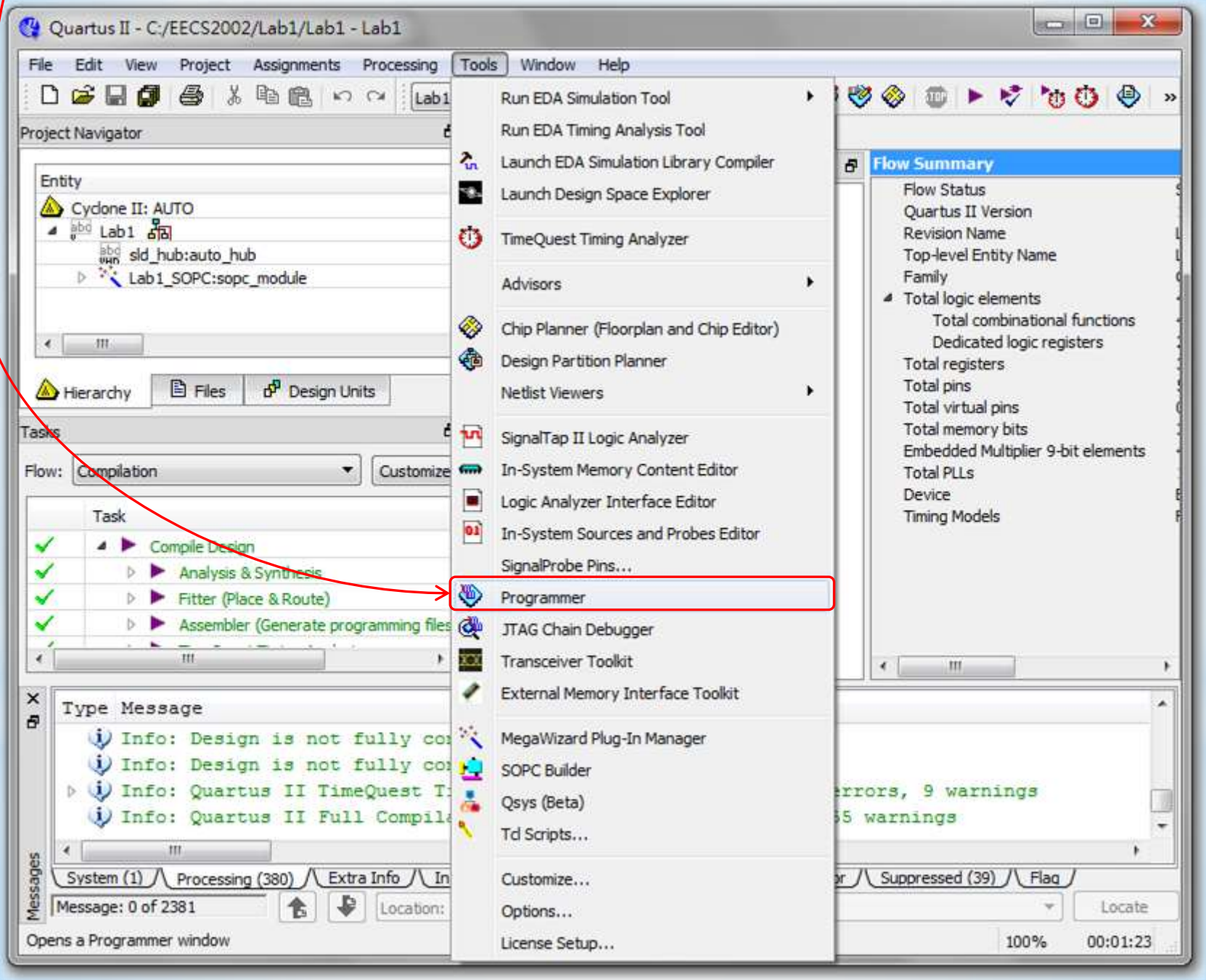
For multi-FPGA board, the nCEO pin of a FPGA chip need be connected to the next FPGA chip. However, in Terasic DE-70 board, there's only one FPGA chip, so Terasic Corporation uses this nCEO pin as the I/O connection to Toggle Switch 7. Therefore, I need to change nCEO from "programmable pin" to "regular I/O".


Quartus II - C:/EECS2002/Lab1/Lab1 - Lab1
File Edit View Project Assignments Processing Tools Window Help
Stop Processing Ctrl+Shift+C
Start Compilation Ctrl+L
Analyze Current File
Start
Update Memory Initialization File
Compilation Report Ctrl+R
PowerPlay Power Analyzer Tool
SSN Analyzer Tool
Project Navigator
Entity
Cyclone II: AUTO
Lab1
Hierarchy
Files
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files
ALTERA
QUARTUS
View Quartus II Information
Documentation
Type Message
Info: Import completed. 531 assignments were written (out of 531 read). 0 non-global assi
Now pin assignment is done. I can compile my project by clicking Processing -> Start Compilation. (this process may take a few minutes)
System (1) Processing Extra Info Info Warning Critical Warning Error Suppressed Flag
Message: 0 of 1
Location:
Locate
Messages
Starts a new compilation
0%
00:00:00

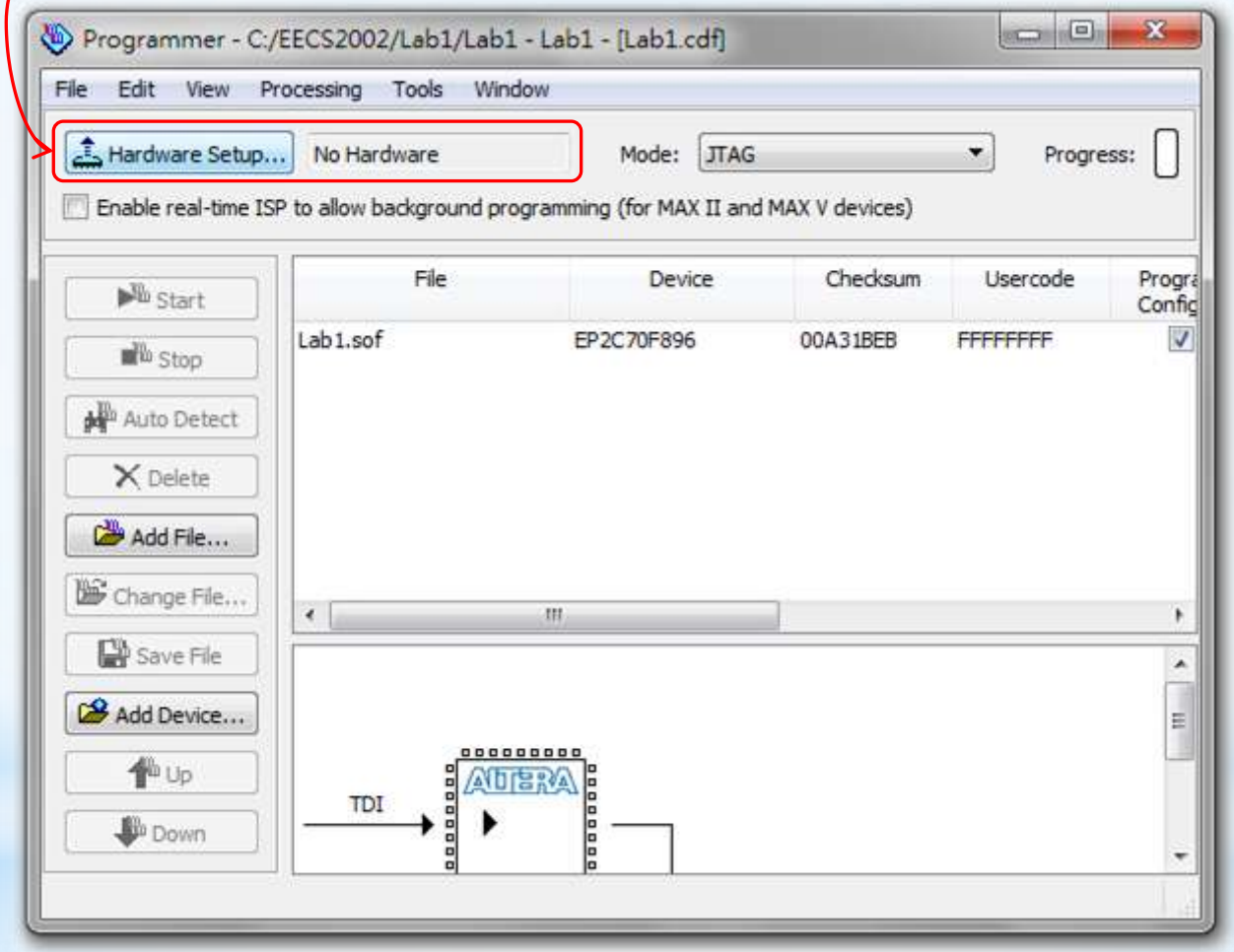
If there's no error (but just some warnings), the compilation will be successfully done. Because we're not designing real IC but just prototyping the design on FPGA, warnings can be ignored and it will still work.



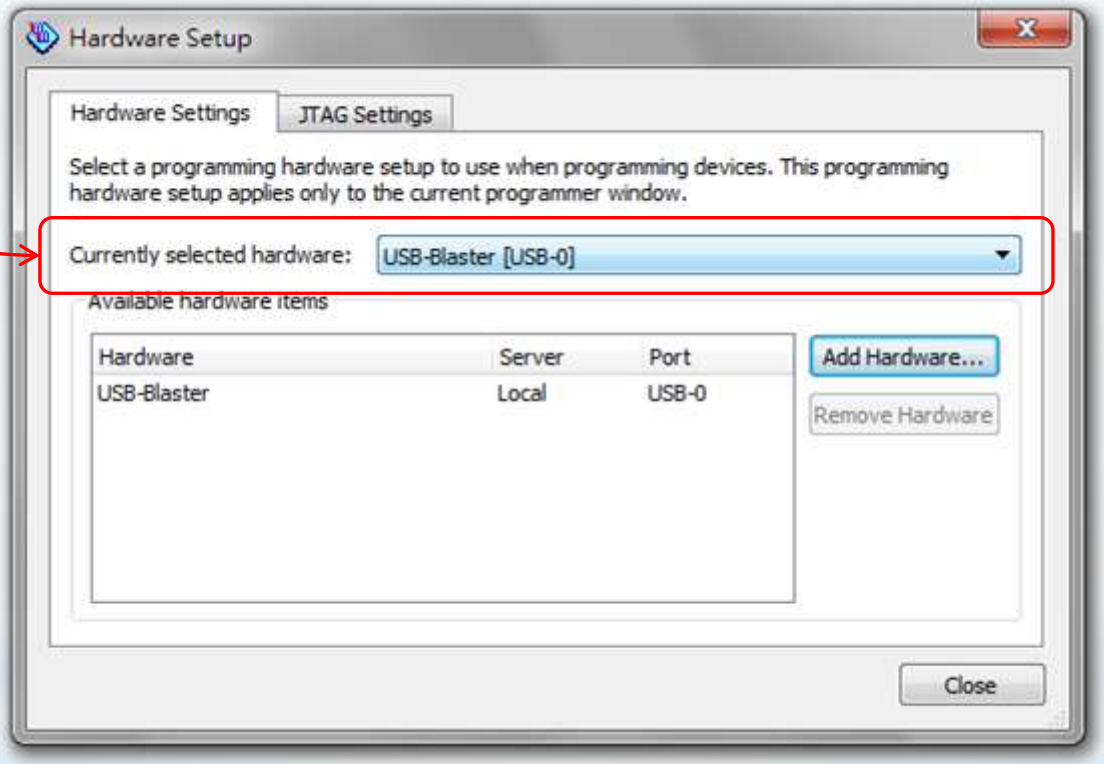
Now I can burn my design into FPGA chip by clicking Tools -> Programmer.



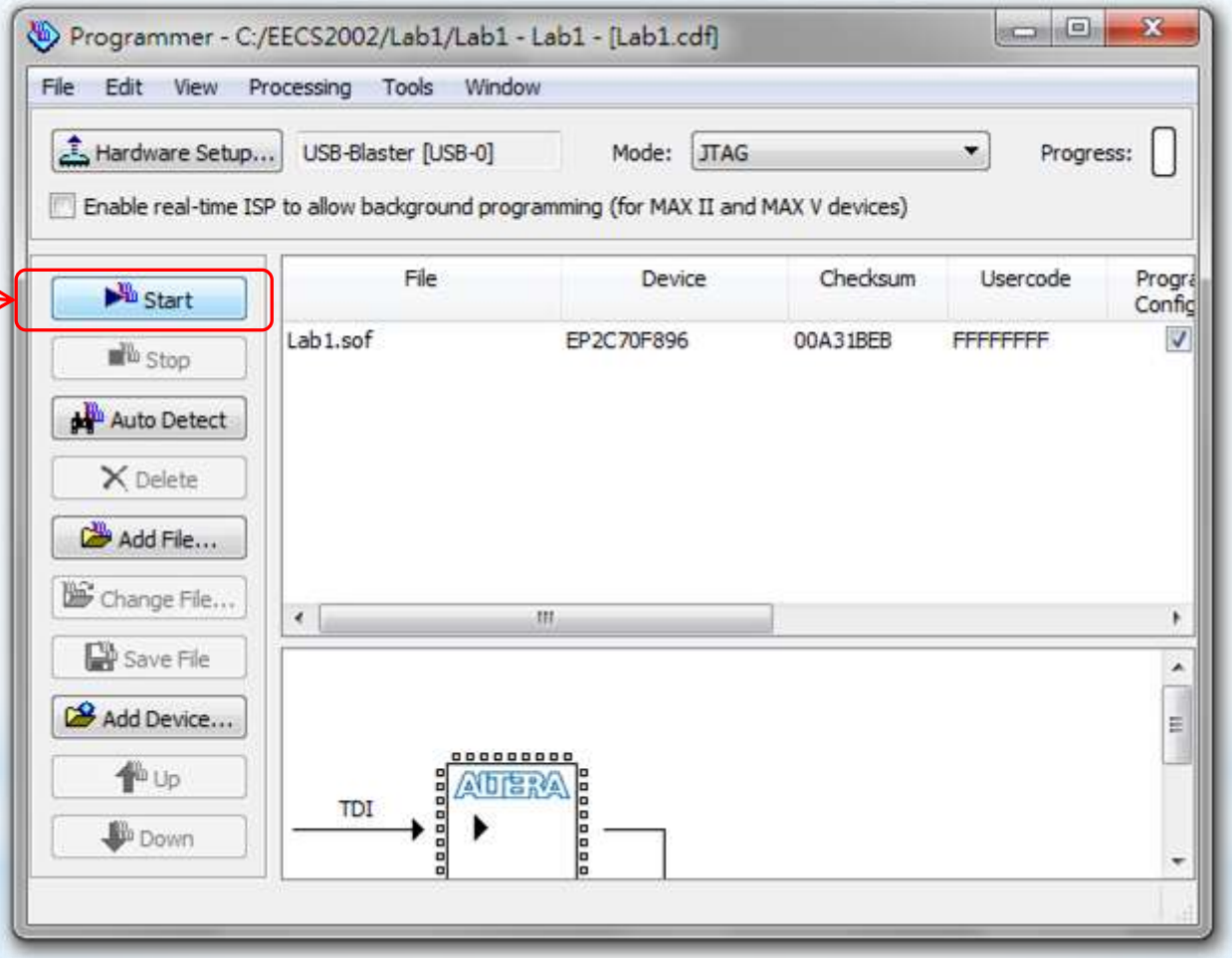
Because there's no hardware device selected,
I click the Hardware Setup button.



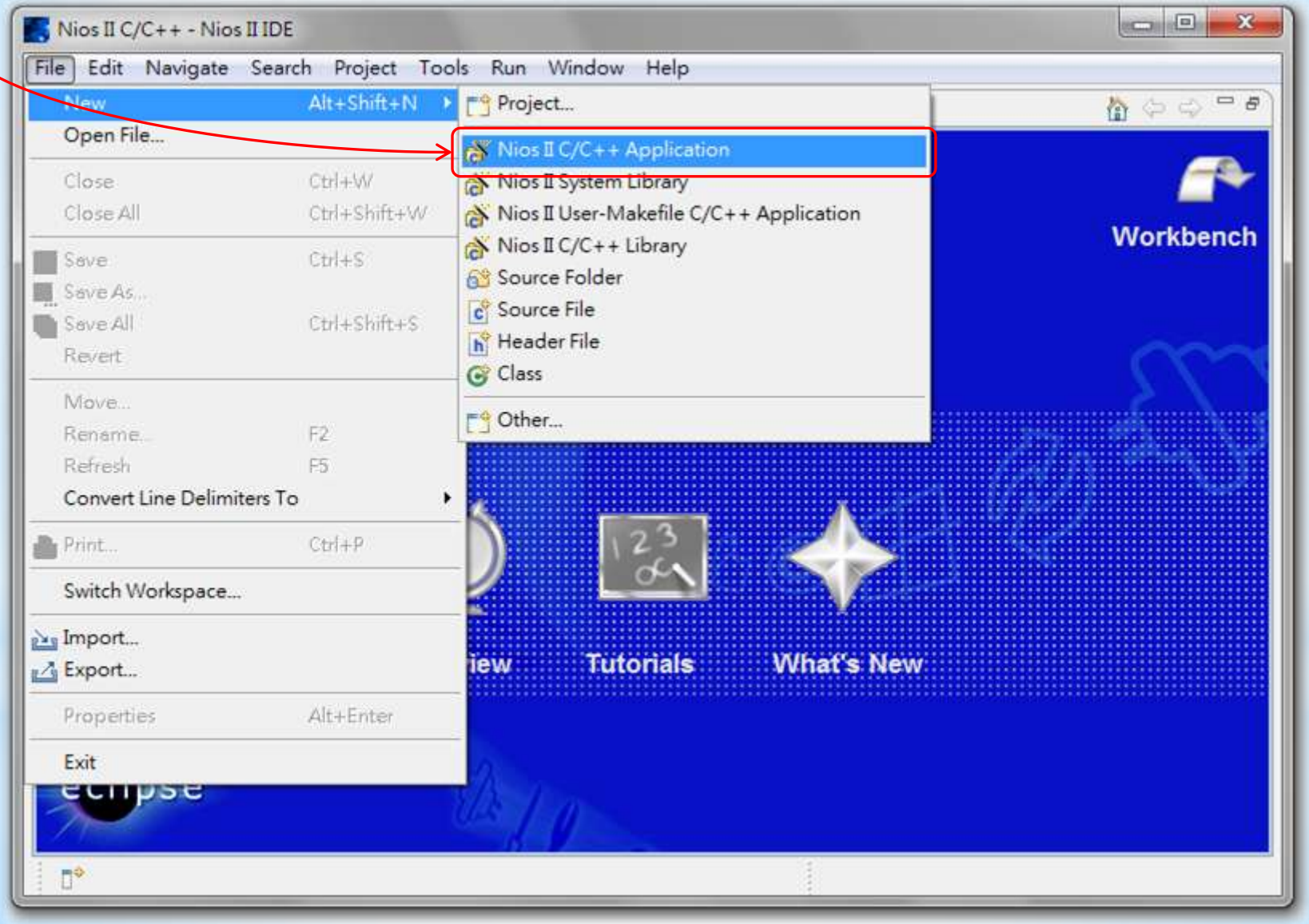
I select USB-Blaster [USB-0], where my Terasic DE2-70 Board is connected.



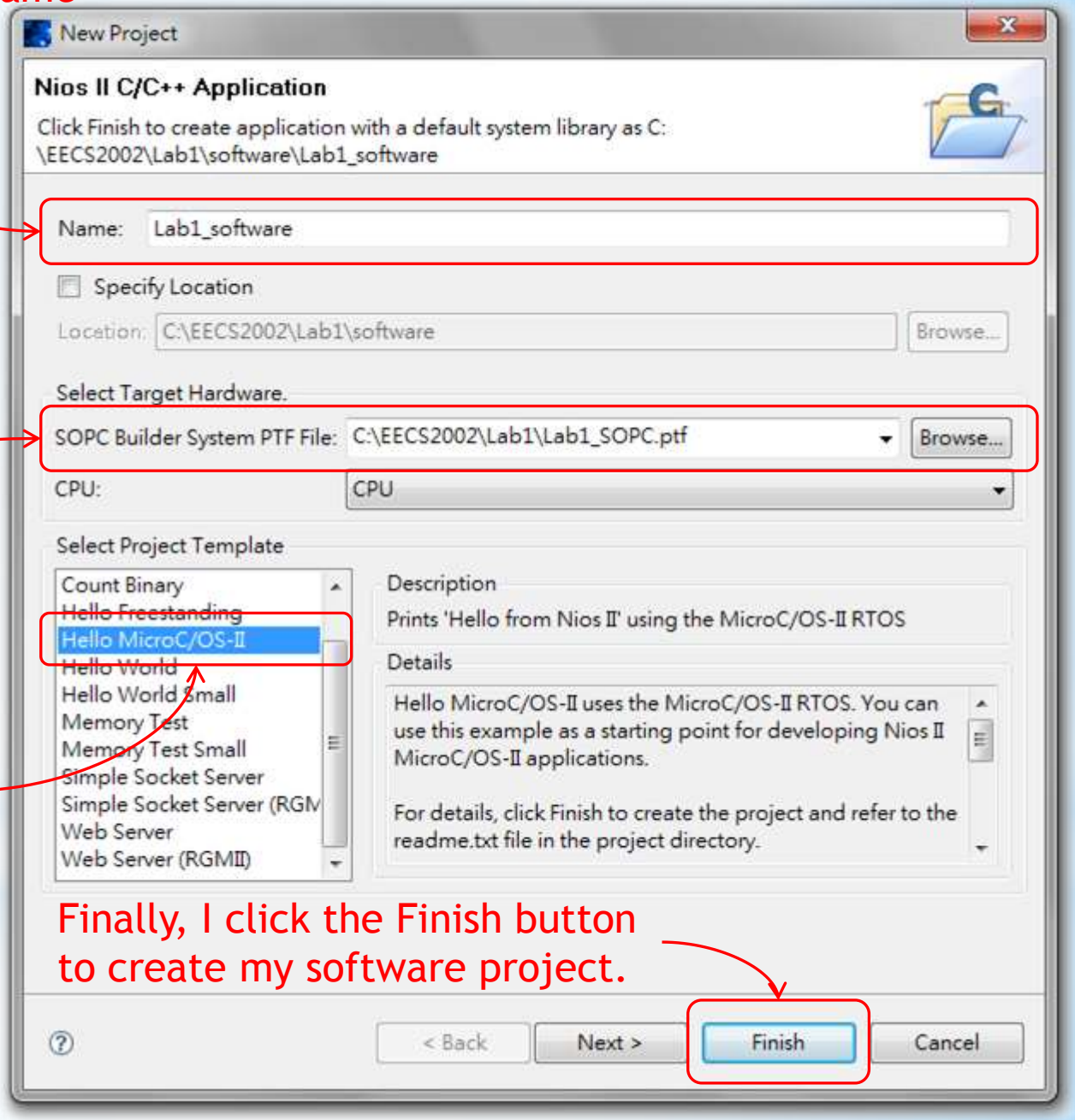
Now I can click the Start button to burn my project into FPGA chip and everything for hardware is done! Let's move on to the software part.



I don’t have any project so I click File -> New -> Nios II C/C++ Application.



I want my C/C++ program name is Lab1_software.

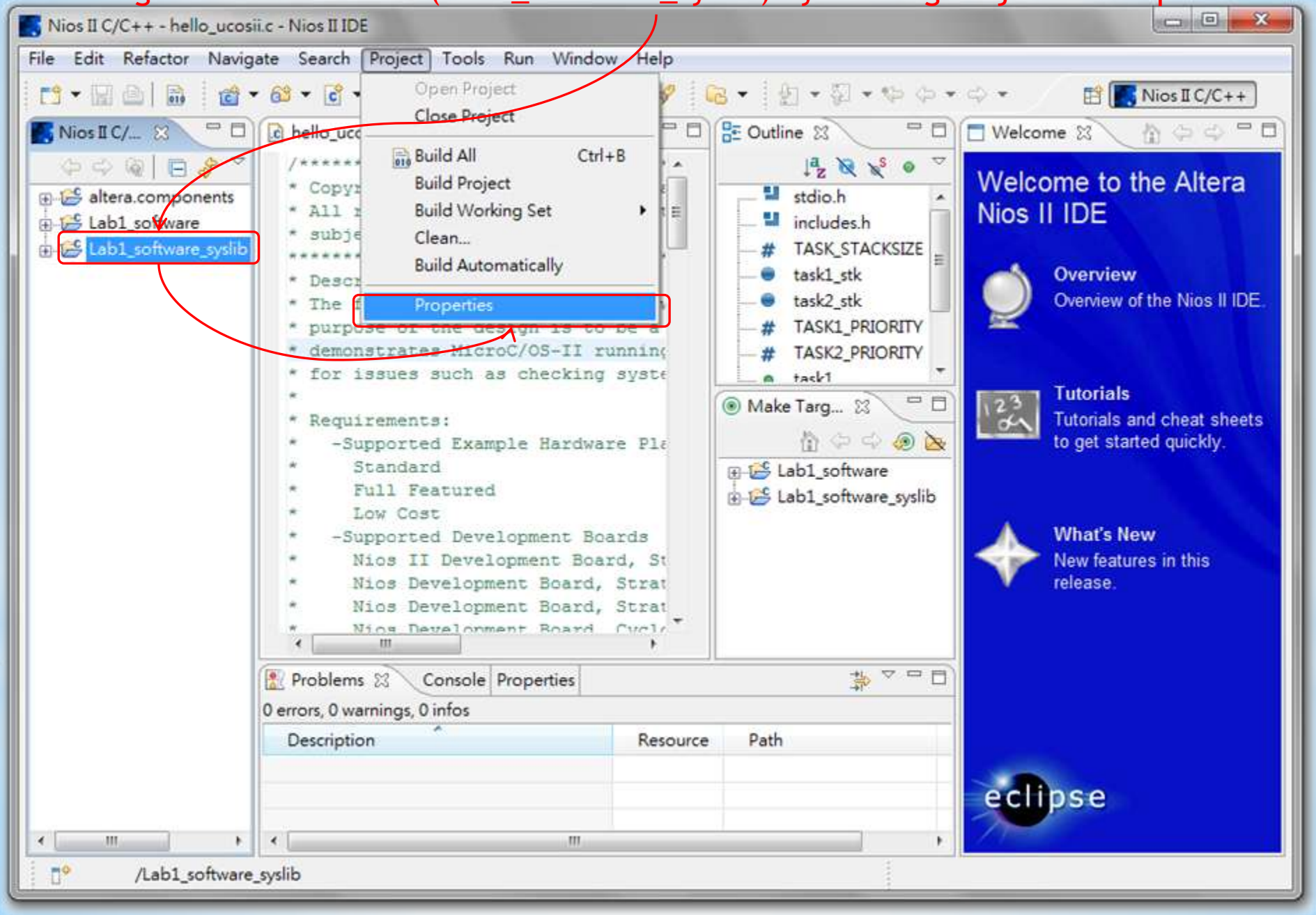


Also, I need to select my SOPC configuration file for Nios II EDS to auto-generate software core.

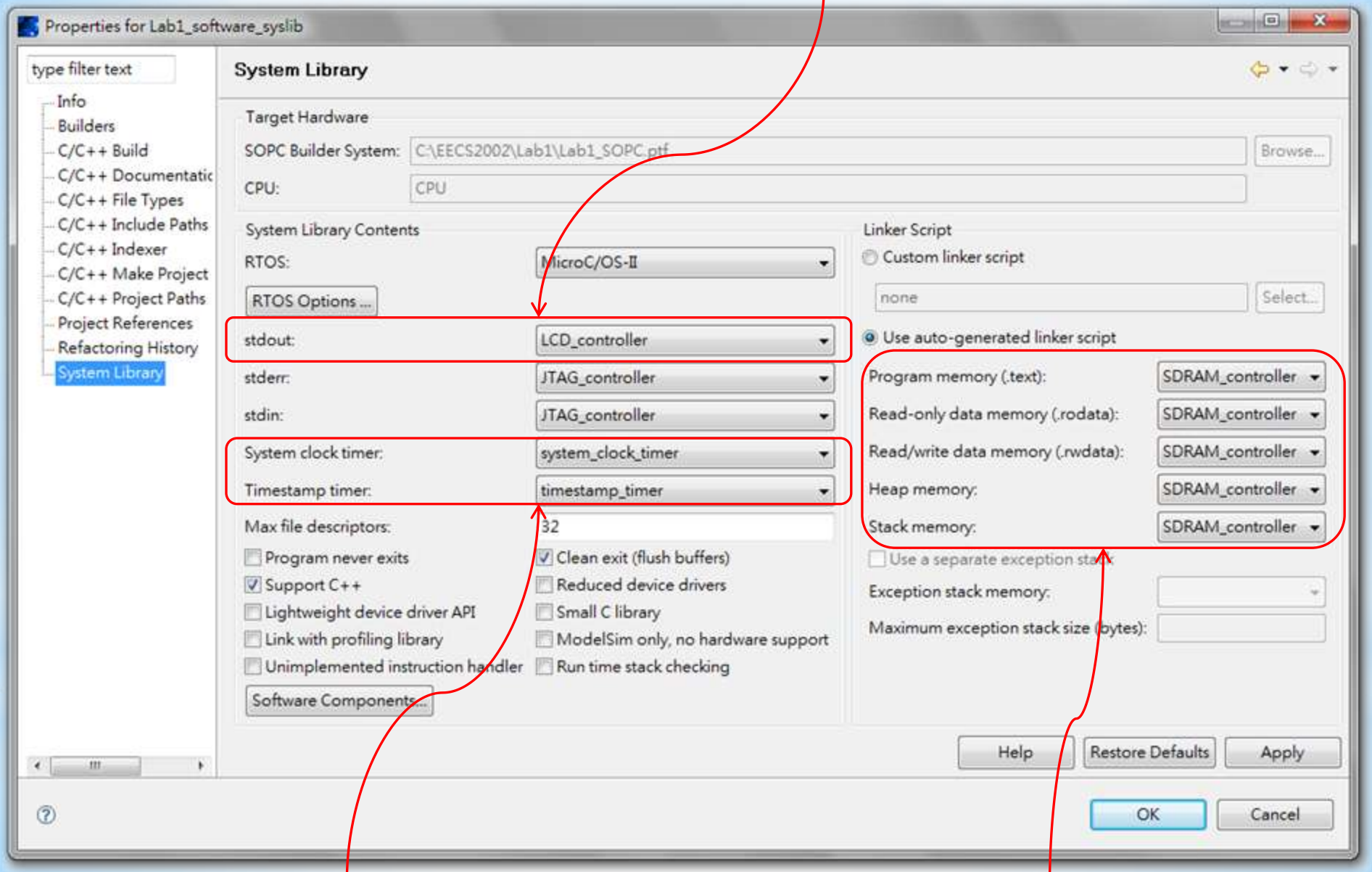
For convenience, I use "Hello MicroC/OS-II" as my project template.

Finally, I click the Finish button to create my software project.

Before I develop my C/C++ program (Lab1_software), I need to change some settings in software core (Lab1_software_syslib) by clicking Project -> Properties.



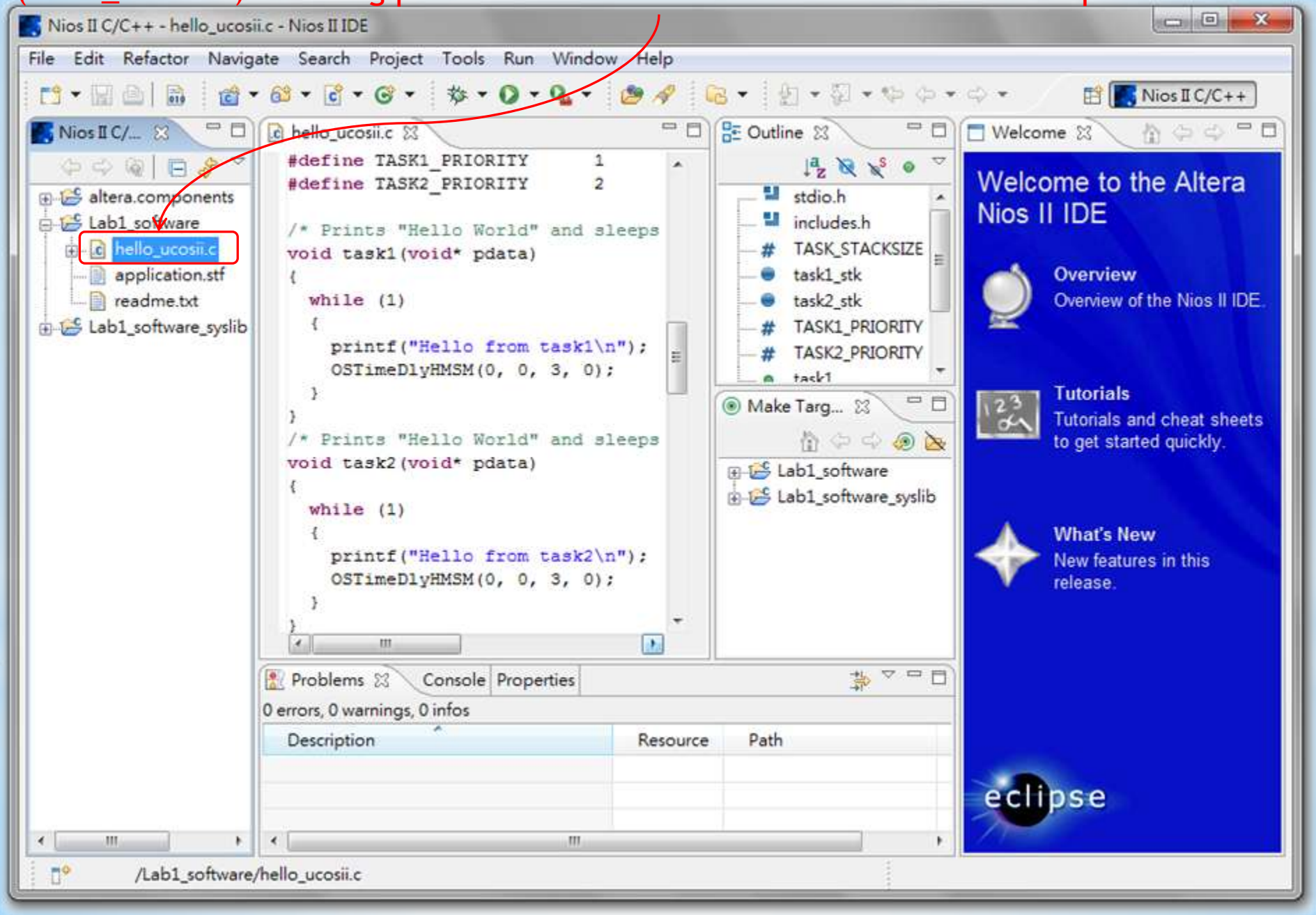
I want to use Character LCD as my Standard C/C++ Output.



I need to set timers for running µC/OS-II.

I want to burn my program into SDRAM.

Now I can develop my C/C++ program (Lab1_software). There's a good example file (hello_ucosii.c) for using µC/OS-II. I need to edit it to achieve Lab 1 Experiment Goal.



I change all codes in hello_ucosii.c to the following one:

```
#include <io.h>
#include <stdio.h>
#include "includes.h"
#include "system.h"
OS_STK s1[2048], s2[2048];
void f1(void *p) {
    while (1) {
        printf("EECS2002 Lab1\nName: 9760111\n");
        OSTimeDlyHMSM(0, 0, 3, 0);
        printf("Hello uC/OS-II\nNice to see you\n");
        OSTimeDlyHMSM(0, 0, 3, 0);
    }
}
void f2(void *p) {
    unsigned int buffer;
    while (1) {
        buffer = IORD(TOGGLE_SWITCHES_CONTROLLER_BASE, 0);
        IOWR(RED_LEDS_CONTROLLER_BASE, 0, buffer);
    }
}
int main() {
    OSTaskCreateExt(f1, 0, (void*) &s1[2047], 1, 1, s1, 2048, 0, 0);
    OSTaskCreateExt(f2, 0, (void*) &s2[2047], 2, 2, s2, 2048, 0, 0);
    OSStart();
    return 0;
}
```

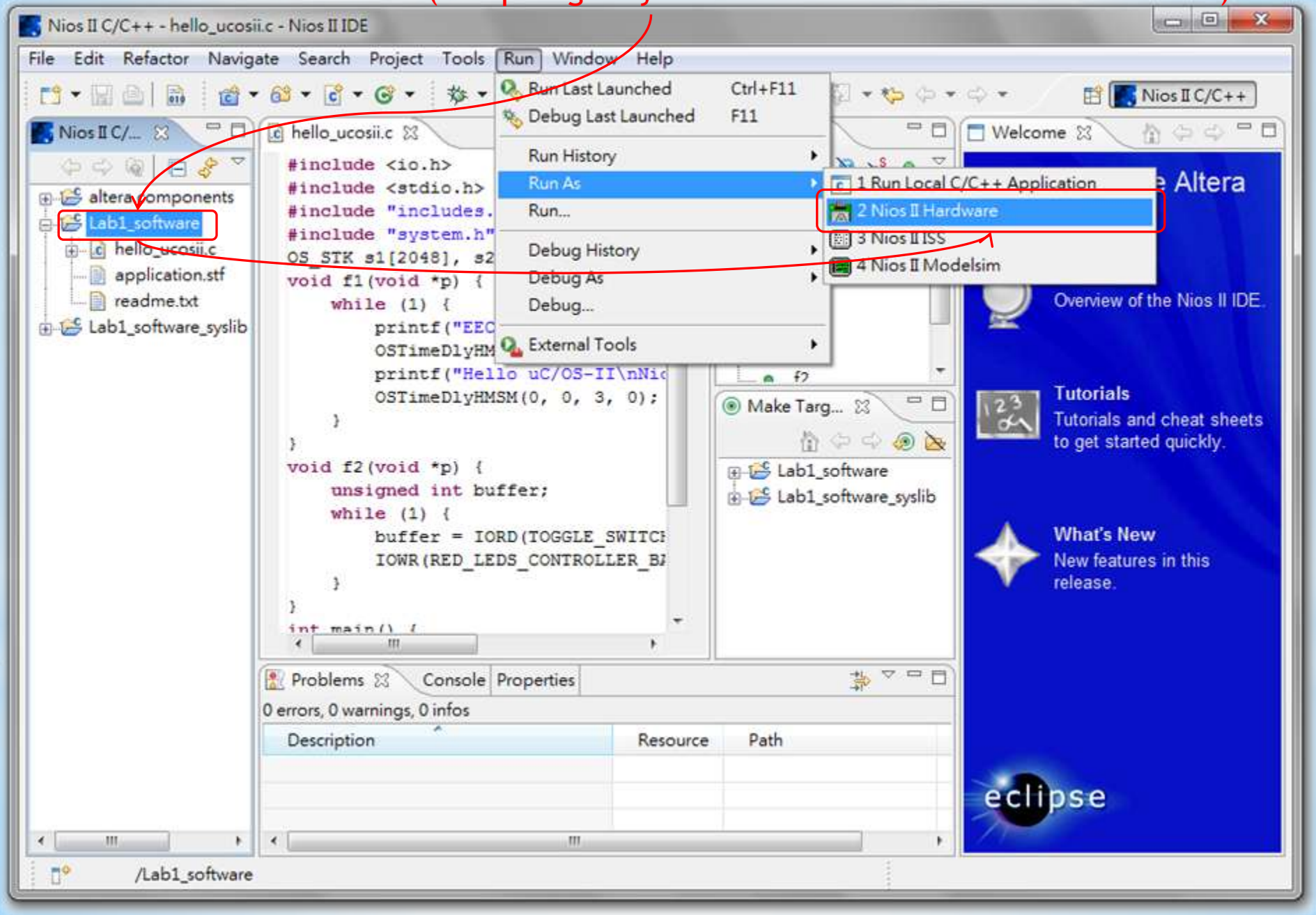
I want to use IORD and IOWR, so I need io.h header.

Every task needs to allocate its own stack memory.

The base addresses for device controllers are auto-defined in system.h by Nios II EDS.

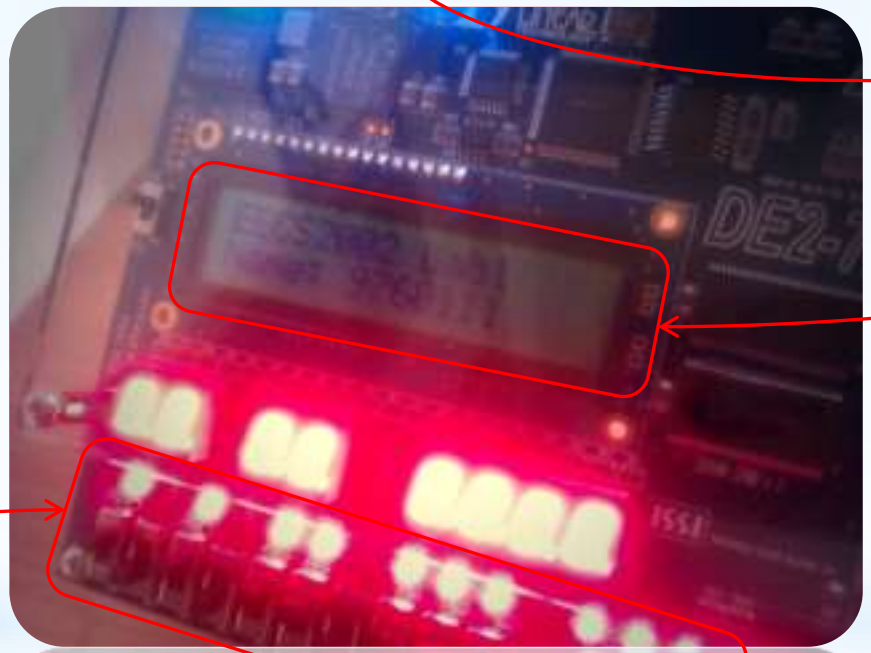
I need to create two tasks before I start running µC/OS-II. These functions are auto-defined in includes.h by Nios II EDS.

Now I can burn my C/C++ program (Lab1_software) into memory by clicking Run -> Run As -> Nios II Hardware (compiling may takes a few minutes at the first time)



*Experiment Result

Now I can see my messages in Character LCD and it changes every 3 seconds.



Also, I can open or close 18 Toggle Switches to control 18 Red LEDs.

*Review Questions

1. Why we need to use µC/OS-II in Lab 1? (Hint: Multi-tasking) How does it work? (Hint: Interrupt, Context Switching)
2. Assume we have two logic functions. Let f1(a, b, c, d) = abcd, f2(a, b, c, d) = ab + c'd'. Which logic function need more LEs to be implemented in FPGA? Why? (Hint: LUT)
3. Why we need to use PLL in Lab 1? (Hint: CPU, SDRAM)
4. Why CPU can access M4K much faster than SSRAM, SDRAM and Flash memory? (Hint: On-chip, Off-chip)
5. What is the basic difference of the hardware structure between a SSRAM cell and a SDRAM cell? (Hint: Latch, Capacitor)
6. In Lab 1, we've set CPU Reset Vector and Exception Vector to SDRAM and burn our program in it. Can our system normally run after we push Reset Button? Why? (Hint: Volatile Memory)
7. Why we need to use bridges to attached the SSRAM controller and the Flash controller to the bus? (Hint: Interface)
8. Why we need to define unused pin as tri-stated? (Hint: Damage)
9. Why we need to define nCEO as regular I/O? (Hint: Single FPGA)