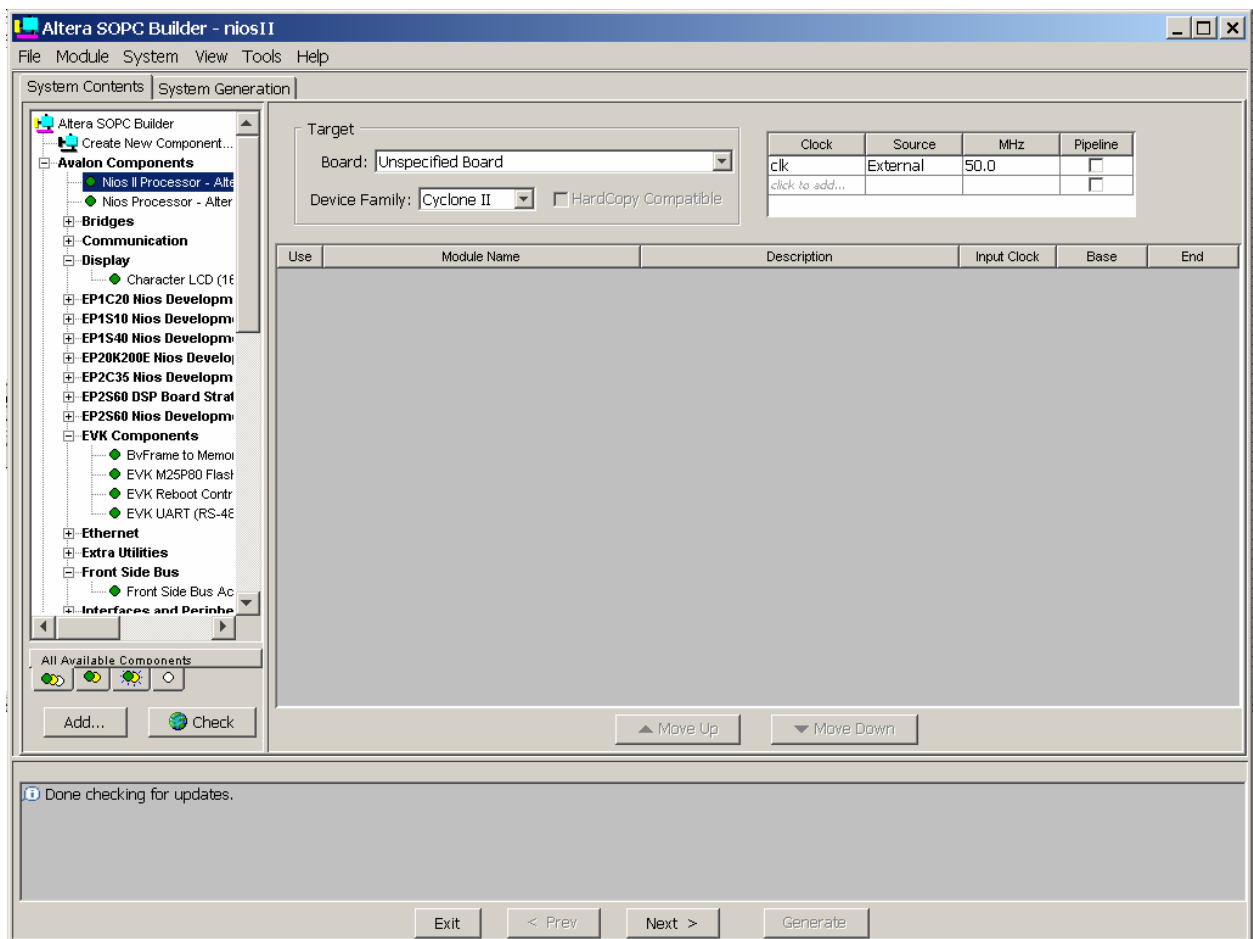


Nios II 设计练习

Lab 1

创建 Nios II 系统

1. 双击\Nios2\DE2Project.qpf. 开始 Quartus II 软件。
2. 在这个工程文件里器件族的选择和管脚分配已经是完成了，你可以在分配菜单（Assignments Menu）里看到
3. 现在开始建立我们的嵌入式系统，在 **Tools => SOPC Builder...**中启动 **SOPC Builder**，并且填写系统的名称“**NIOSii**”（也可以放其他名字），当下一个窗口弹出的时候，你可以选择 **VHDL or Verilog**（可以按照你的习惯来选择）作为执行语言。这空白的 **SOPC Builder** 窗口将会被打开，设置 **Target** 为你所用的特定的开发板（比如：**Nios Development Board,DE2**），还有设置 **Clock** 频率为 **50Mhz**



4. 从左窗口那里选择 **Nios II Processor**，点击 **Add**，选择 **Nios II/s** 作为这个处理器的内核，同时也选择 **Embedded multiplies** 作为硬件乘法器。

Altera Nios II - cpu_0

Nios II Core | Caches & Tightly Coupled Memories | JTAG Debug Module | Custom Instructions

Select a Nios II core:

| | <input type="radio"/> Nios II/e | <input checked="" type="radio"/> Nios II/s | <input type="radio"/> Nios II/f |
|--|---------------------------------|--|--|
| Nios II Selector Guide Family: Cyclone II f _{system} : 50 MHz | 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 MHz | Up to 5 DMIPS | Up to 26 DMIPS | Up to 56 DMIPS |
| Logic Usage | 600-700 LEs | 1200-1400 LEs | 1400-1800 LEs |
| Memory Usage | Two M4Ks | Two M4Ks + cache | Three M4Ks + cache |

Hardware Multiply: ☐ Hardware Divide

Altera Nios II - cpu_0

Nios II Core | Caches & Tightly Coupled Memories | JTAG Debug Module | Custom Instructions

Instruction Bus

Instruction Cache:

☐ Include tightly coupled instruction master port(s).

Number of ports:

You must connect each port to exactly one memory in the SOPC Builder connection panel.

Data Bus

Data Cache: ☐ Omit data master port

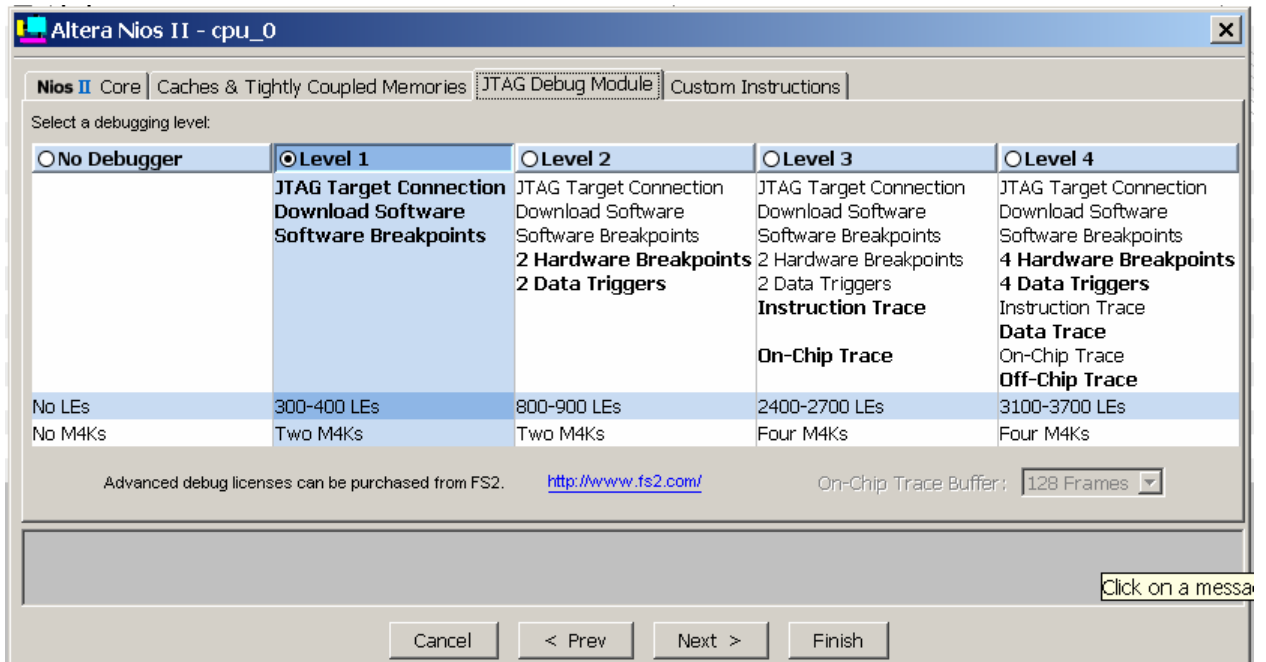
Data Cache Line Size:

☐ Include tightly coupled data master port(s).

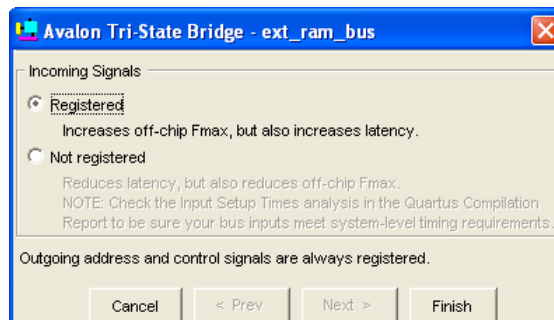
Number of ports:

You must connect each port to exactly one memory in the SOPC Builder connection panel.

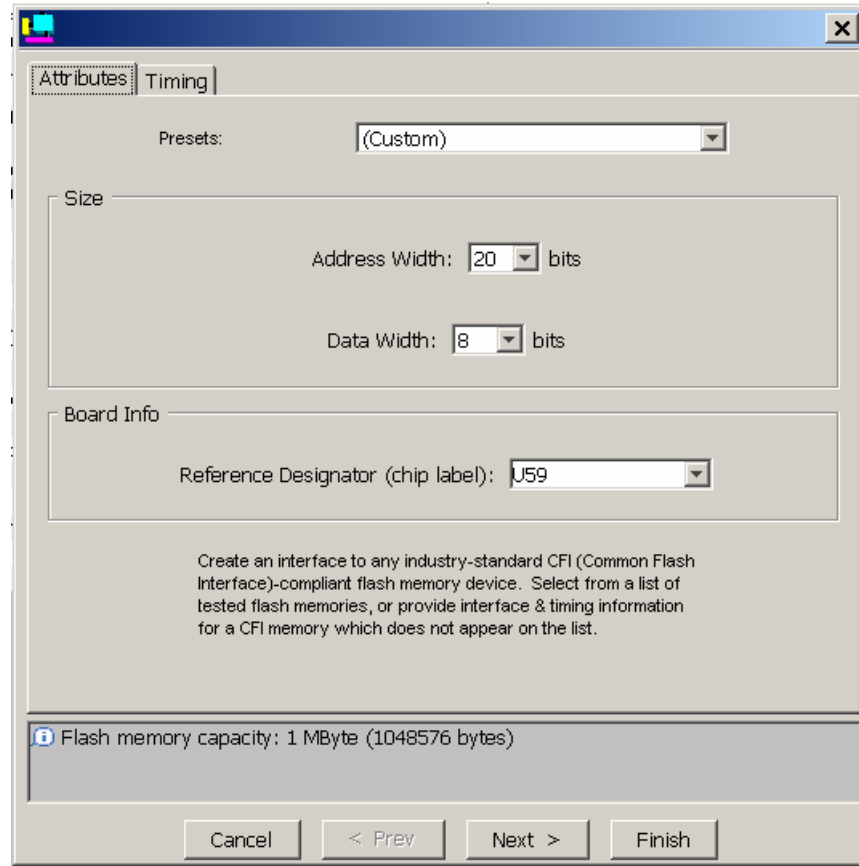
5. 现在，点击 **JTAG Debug Module**，选择 **JTAG Target Connection Download Level 1** 选项，在这个表上面提供了所有调试的选项。



6. 点击 **Finish.**，之后将它增加到 **SOPC Builder** 系统
7. 右键点击它，选择 **Rename** 可以改处理器的名字，打入“**cpu**”然后确认。
8. 从左窗口窗格 **Bridges** 里选择 **Avalon Tri-State Bridge**，点击 **Add**，选择 **Registered** 选项，并且点击 **Finish**，把这外围改名为“**ext_ram_bus.**”。

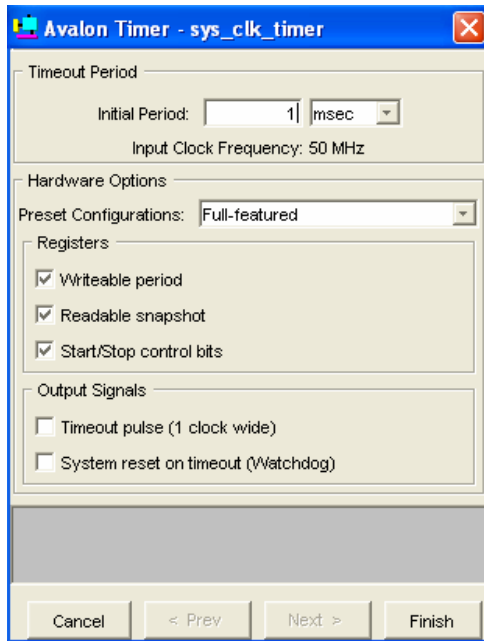


9. 从左窗口窗格 **Memory** 里选择 **Flash Memory (Common Flash Interface)**，点击 **Add**，选择 **DE2 boards**：prset 里选择 **custom**，在 **address width** 为 20，而 **data width** 为 8 之后点击 **Finish**，重新改名为 **ext_flash**

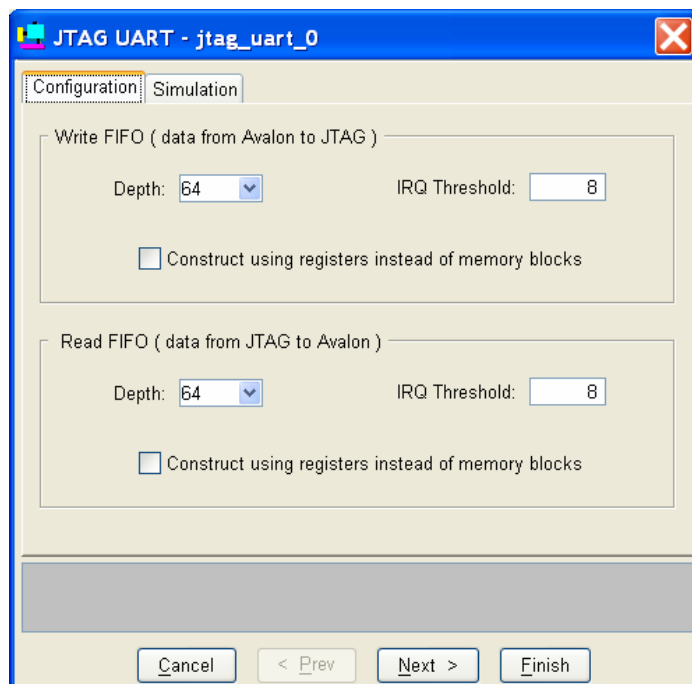


从左窗口窗格 **Memory** 里选择 **EPCS Serial Flash Controller**，点击 **Add**，其余的为默认值

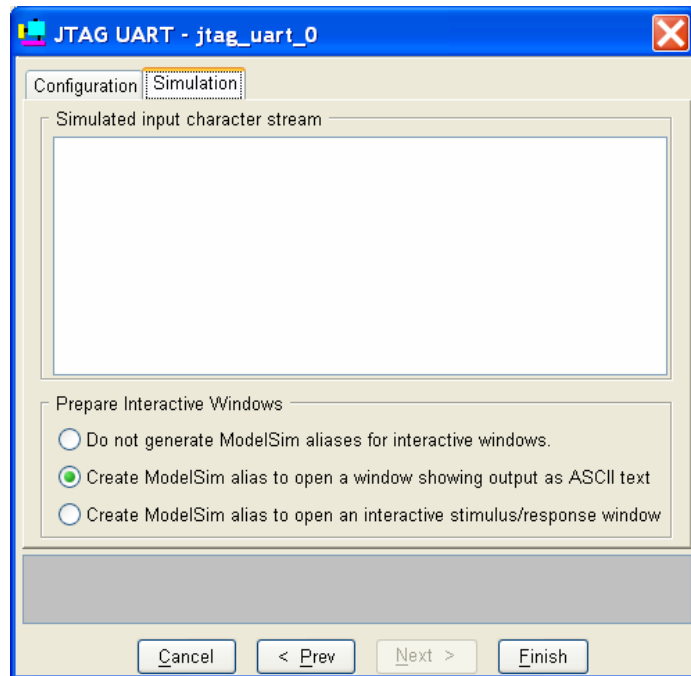
10. 从左窗口窗格 **Other** 选项里选择 **Interval Timer**，点击 **Add**，其余的为默认值，点击 **Finish**，重新把它改名为“**sys_clk_timer**”



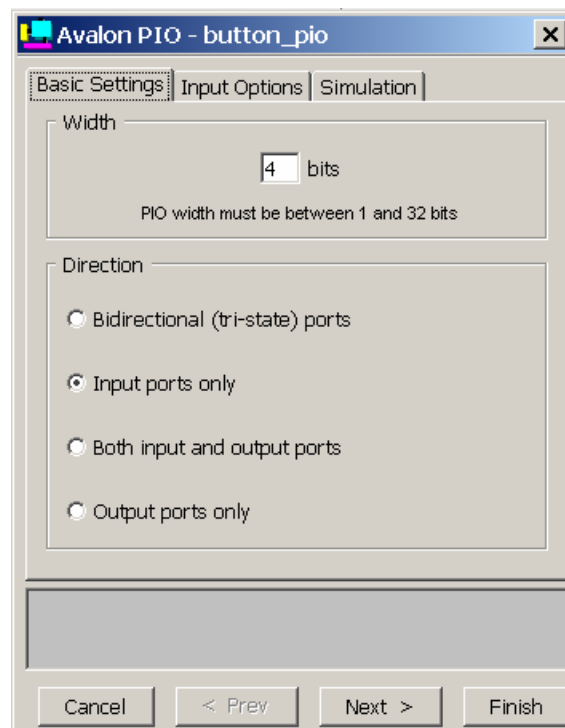
11. 从左窗口窗格 **Communication** 选项里选择 **JTAG UART**，点击 **Add**，其余的为默认值，点击 **Finish**，重新把它改名为” **sys_clk_timer** ”



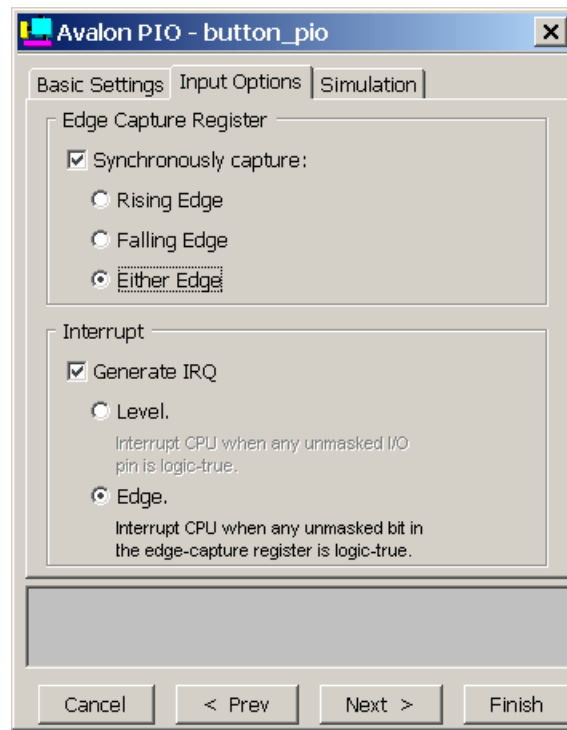
12. 点击 **Simulation** 标号，选择 **Create ModelSim alias to open a window showing output as ASCII text**，点击 **Finish**，然后把这外围改名为 **jtag_uart**。



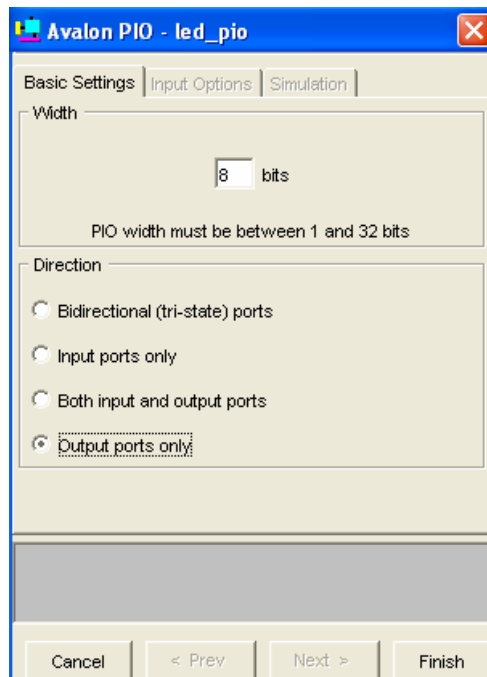
13. 从左窗口窗格选项里再次选择 **PIO (Parallel I/O)**，点击 **Add**，输入一个宽为 4 位，还有选上 **Input ports only**。



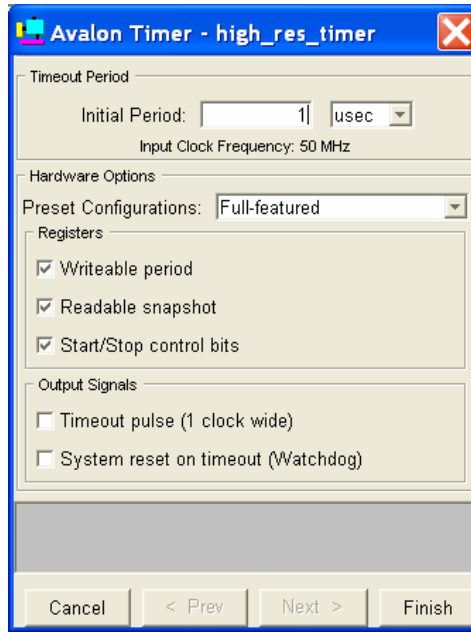
14. 点击 **Input Options** 标号，选择 **Edge Capture Register** 和 **Generate IRQ**，点击 **Finish**，然后把这外围改名为 **button_pio..**



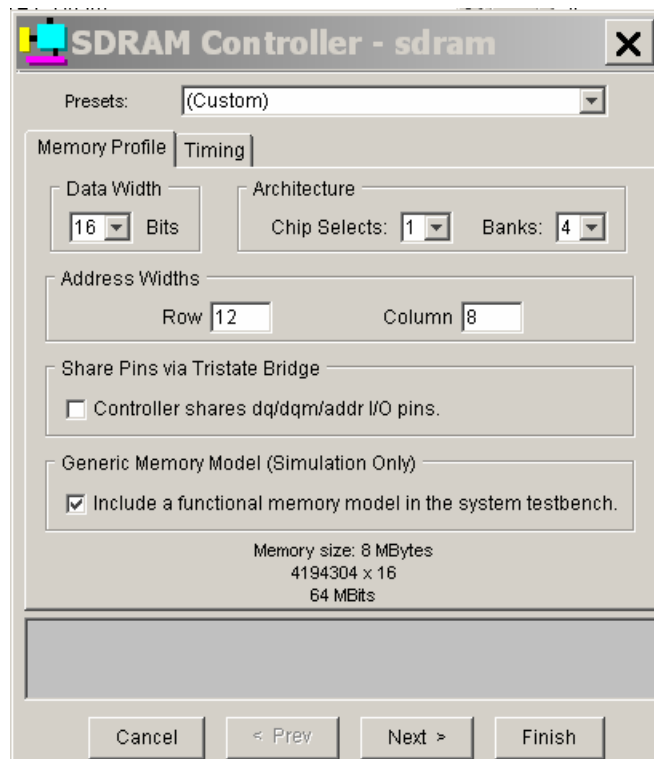
15. 从左窗口窗格选项里再次选择 **PIO (Parallel I/O)**，点击 **Add**，输入一个宽为 8 位，还有选上 **Input ports only.**，点击 **Finish**，然后把这外围改名为 **led_green...**



16. 从左窗口窗格 **Other** 选项里再次选择 **Interval Timer**，点击 **Add**，改变 **usec** 的周期，点击 **Finish**，然后把这外围改名为 **high_res_timer..**（当我们设定好我们一些软件功能的速度时，我们将会用到这个）



17. 从左窗口窗格 **Displa** 选项里选择 **Character LCD(16*2,Optrex 16207)**，点击 **Add**，其余的为默认值
18. 从左窗口窗格 **Other** 选项里选择 **System ID Peripheral**，点击 **Add**，其余的为默认值
19. 从左窗口窗格 **Communication** 选项里选择 **UART (RS-232 serial port)**，其余的为默认值，点击 **Finish**，重新把外围改名为” **uart1.**”
20. 从左窗口窗格 **memory** 选项里选择 **SDRAM Controller**，在 **preset** 里选 **custom**，而在 **data width** 里选上 **16** 位，其余为默认，点击 **Finish**，重新把它改名为” **sdram** ”。



21. 现在，从 **Other** 组件文件夹里增加 **Altera PLL** 组件到这个工程里，然后启动 **ALTPLL MegaWizard** 并且配置好 PLL，就如下图所示，每次都按照下一步来操作这 Wizard

PLL Parameters:

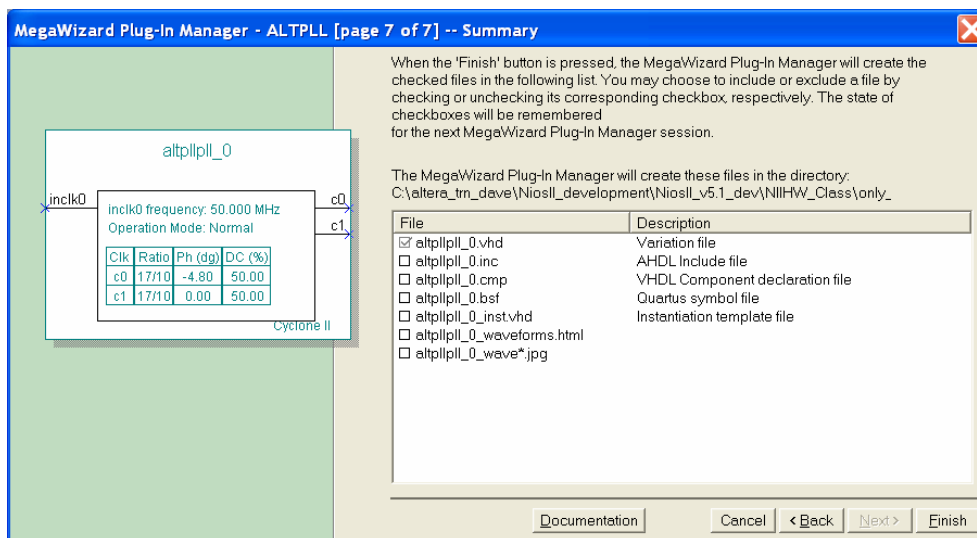
| | |
|--------------------|---------------|
| Input Clock Freq | 50 MHz |
| Auto PLL Selection | ON |
| Mode | Normal |

Jump to page for C0:

| | |
|-------------------|------------|
| C0: Mult Factor = | 1 |
| Div factor = | 1 |
| Phas shift = | 0 |
| Duty Cycle = | 50% |

Jump to page for C1:

PLL wiard 的最后一页将类似如下：（注意：根据你在开始为工程选择的语言，这里将会有小小的不同）

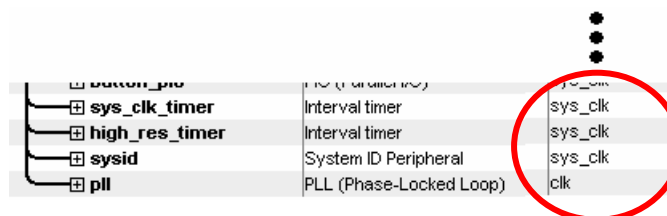


选择 **Finish** ，然后再一次 **Finish** 去增加组件到 **SOPC Builder** 系统，然后改名为 **“PLL”** ；

双击这 **clock** 窗口，并把他 **“clk_1”** 改名为 **sys_clk**，把 **clk_2** 改名为 **“sdrclk”**，这系统时钟窗口将会显示为如下：

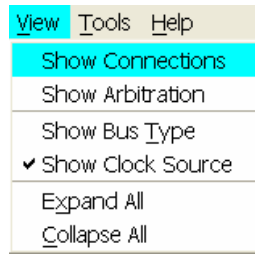
| Clock | Source | MHz | Pipeline | |
|---------|-------------|------|--------------------------|---|
| clk | External | 50.0 | <input type="checkbox"/> | ▲ |
| sys_clk | c0 from pll | 50.0 | <input type="checkbox"/> | |
| sdrclk | c1 from pll | 50.0 | <input type="checkbox"/> | ▼ |

现在点击除了 **PLL** 外的所有外围的 **Input Clock** 专栏，并且改这 **input clock** 为 **“sys_clk”**

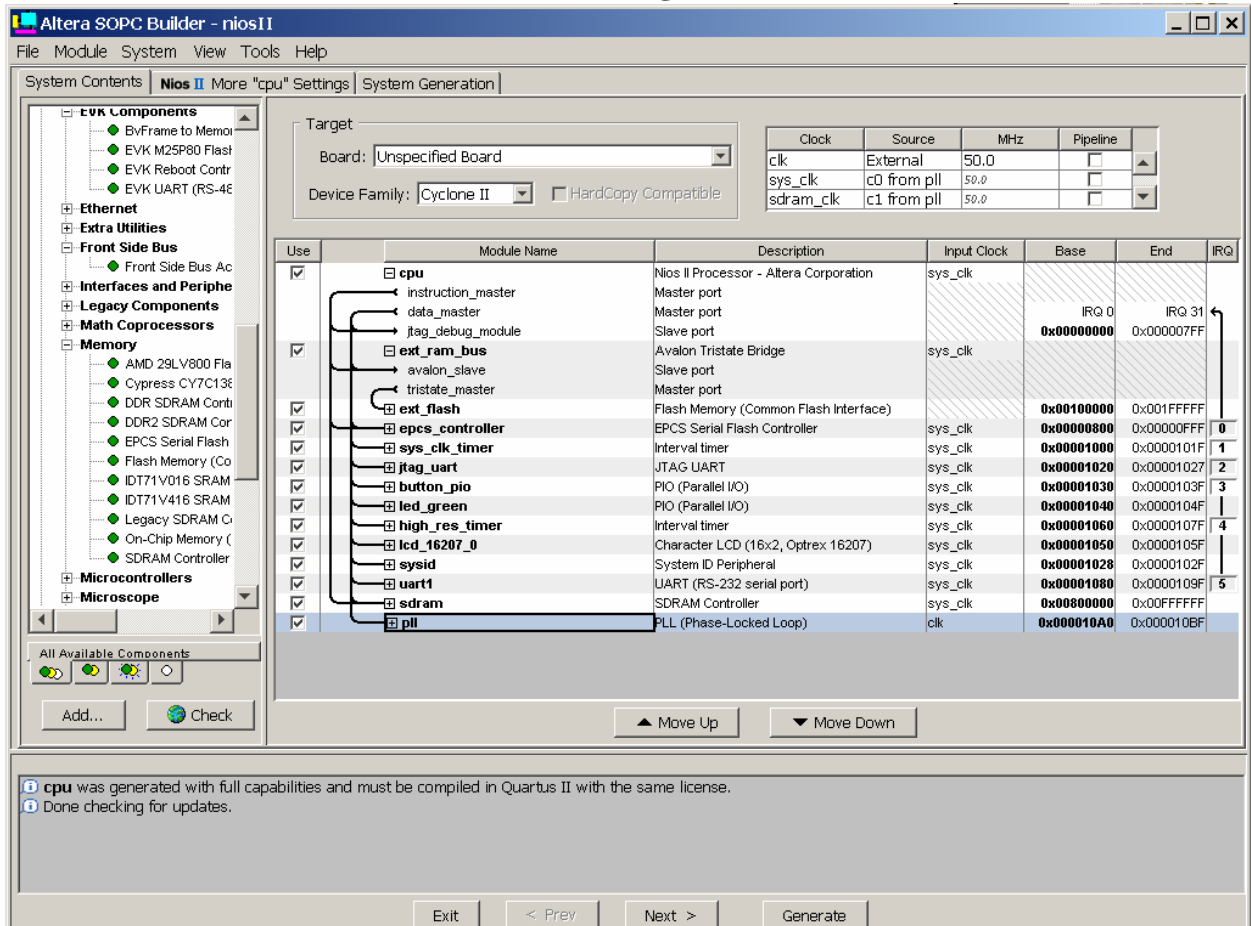


22. 保证所有的基地址是有效的，在这表上右击任何一个基地址，并且选择 **Auto-Assign Base Addresses**.

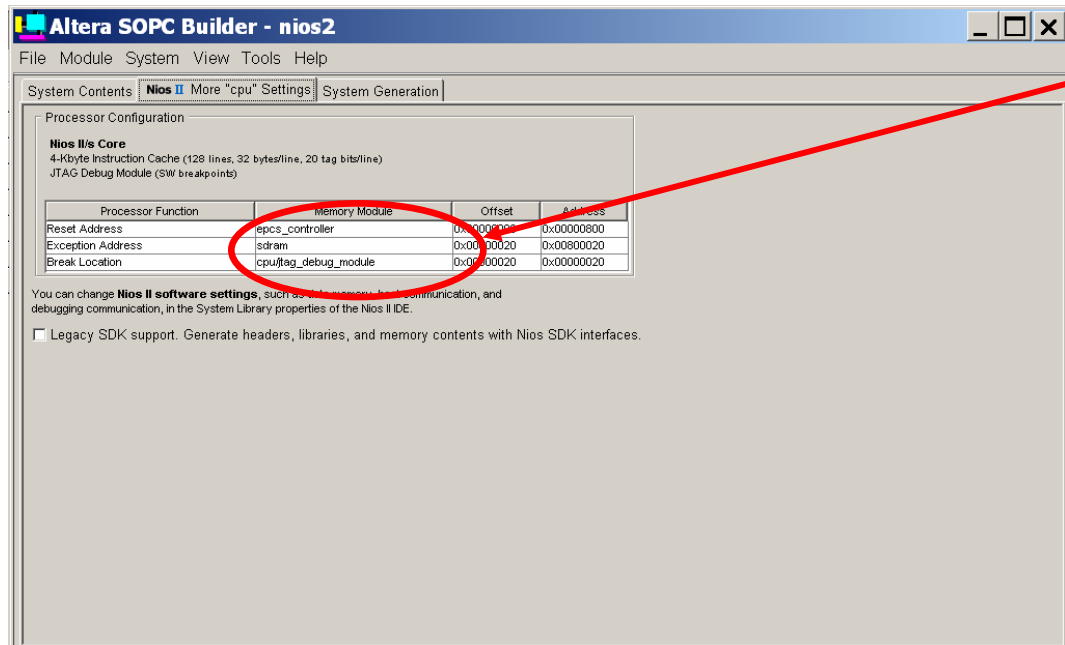
23. 现在，我们必须保证，有适当的主从连接，如果还没有连接好，打开 **View** 菜单，接着选择 **Show Connections**（你的选择旁边将会出现一个核对的标号）



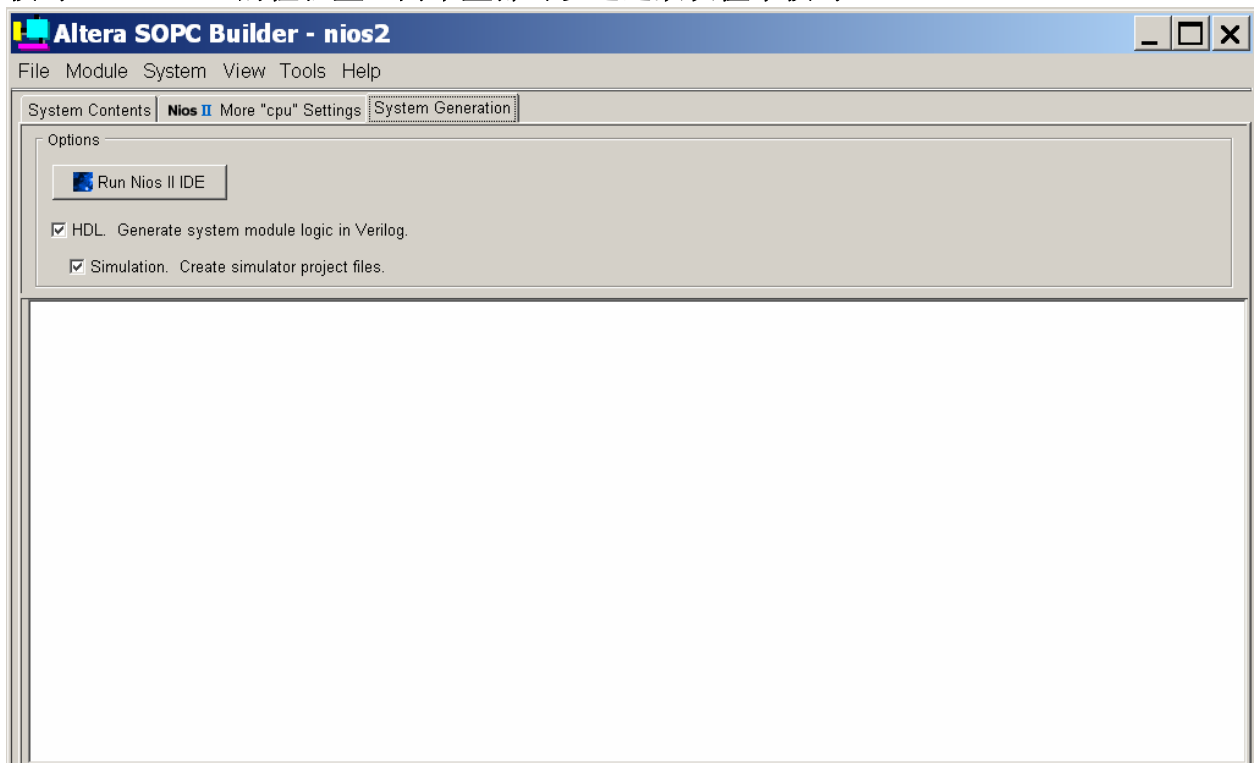
你的系统可以在如下页里看到：（对于 Target 设置为你所用的特定的板子）



24. 现在，点击 Nios II More “cpu” Settings 方格，这页是用来建立 CPU 的复位和异常地址的，为 Reset address 选择 epcs_controller，为 Exception Address 选择 sdram，然后点击 next。

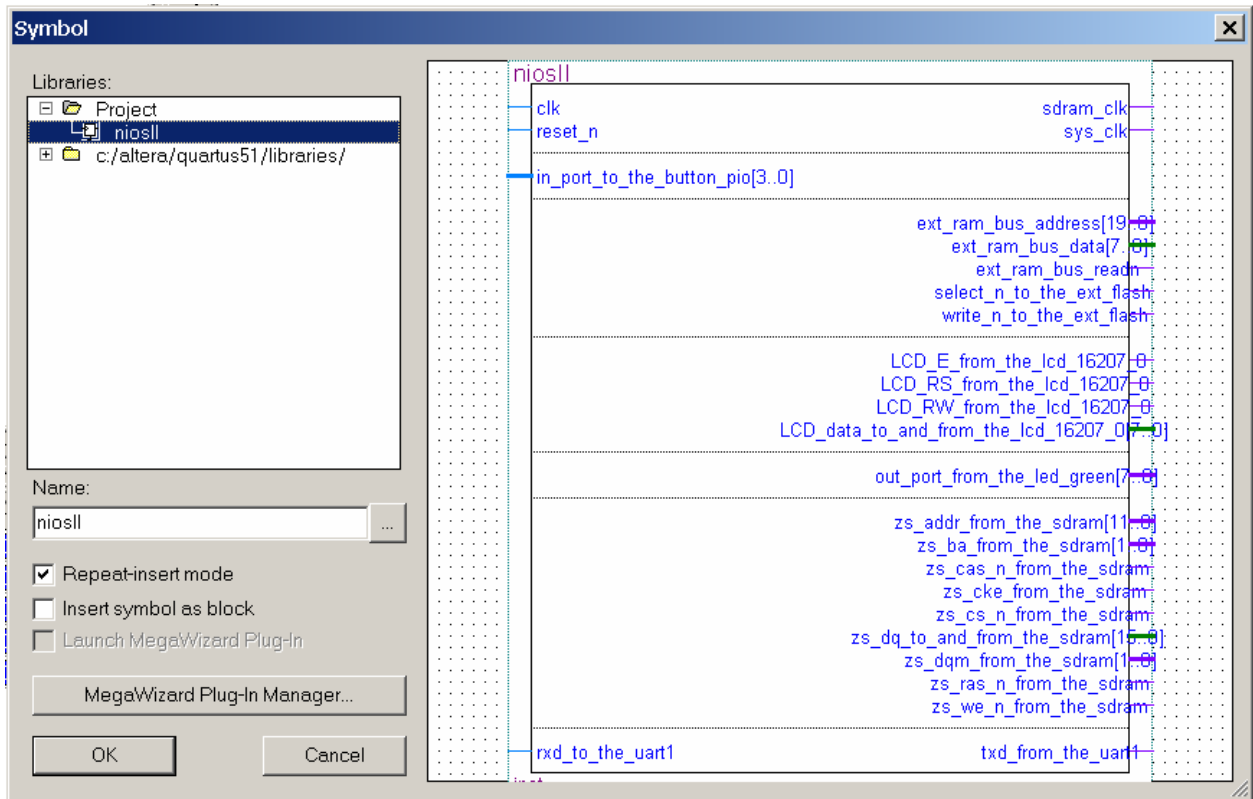


25. 核对 **Simulation** 的检验盒，两个盒都可以通过默认值来核对

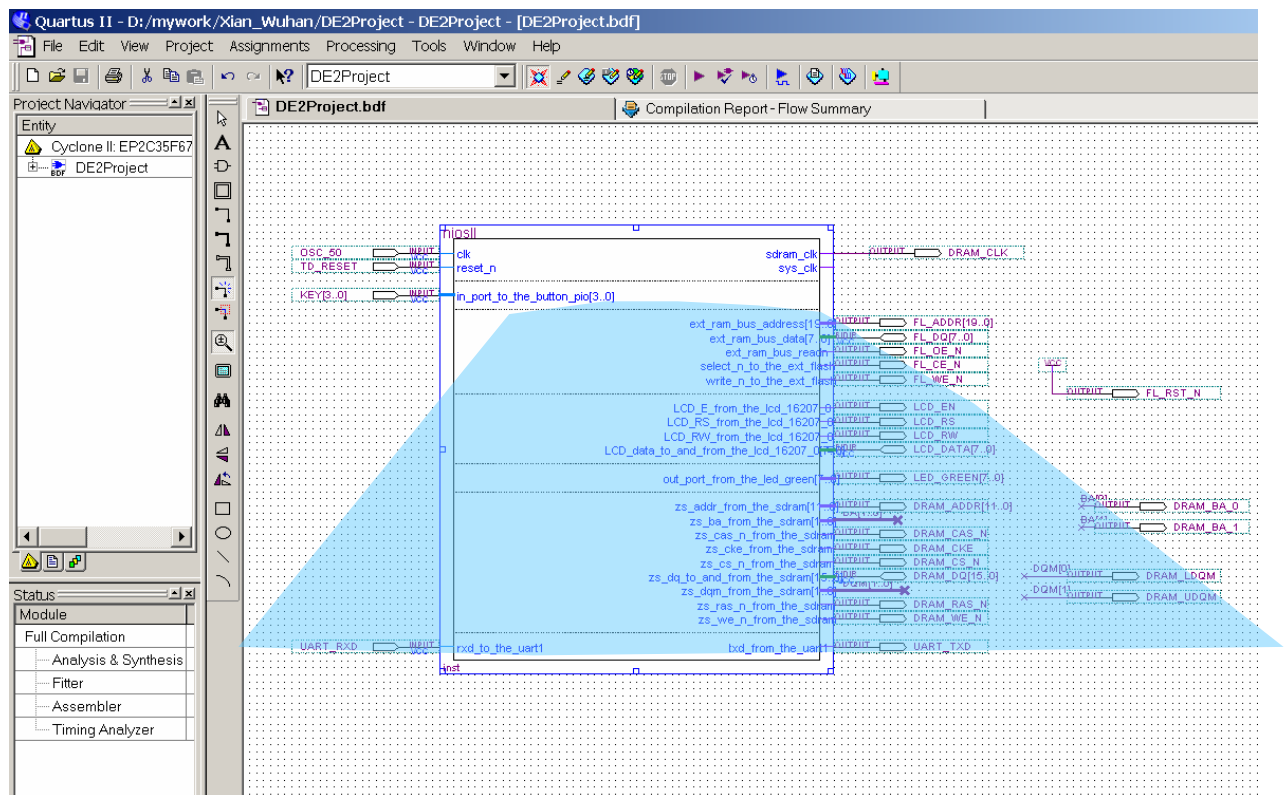


26. 点击 **Generate**，SOPC Builder 将会生成参数化的 Nios II processor system.处理器系统

27. 当 SOPC Builder 生成系统后，QuartusII 工程里，用 File->Open 打开这已经完成的部分 DE2Project.bdf 的原理图，我们可以增加 SOPC 系统到我们的 QuartusII 工程里，在原理图框的任意地方双击，打开 Symbol 视图，然后，在 symbol viewer 里打开这工程的文件夹，选择 NIOSII





28. 按 OK，放置这 symbol 在原理图框里，以致可以像如下图一样把管脚整体排列布置，管脚应该是整体准确排列，如果没有，很可能你需要回到 SOPC Builder 里重新核对你所做的

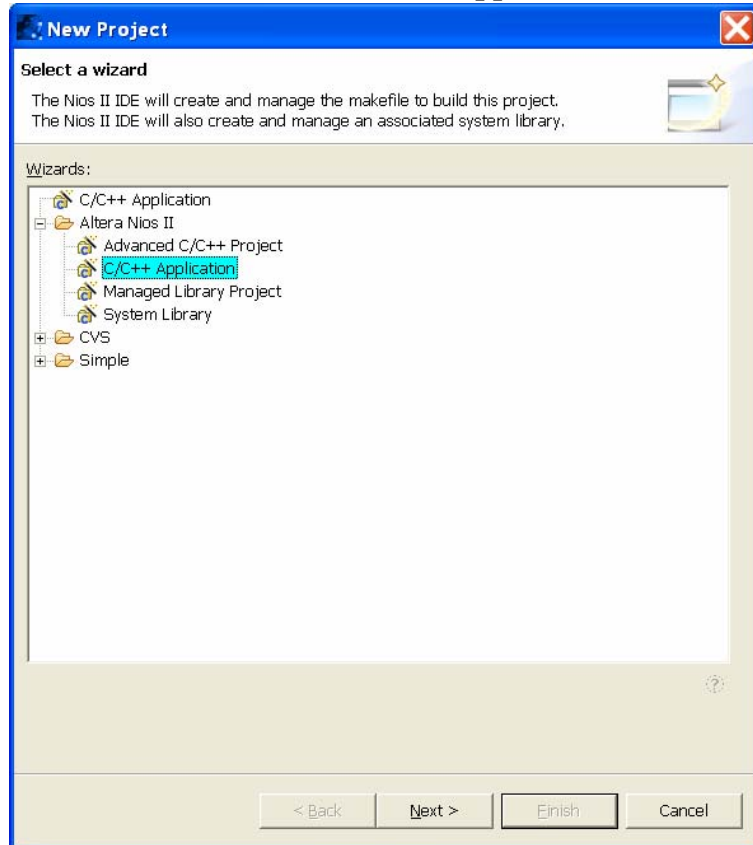


29. 保存原理图，在 Quartus Processing 里选择 Start Compilation 开始编译。

Lab 2

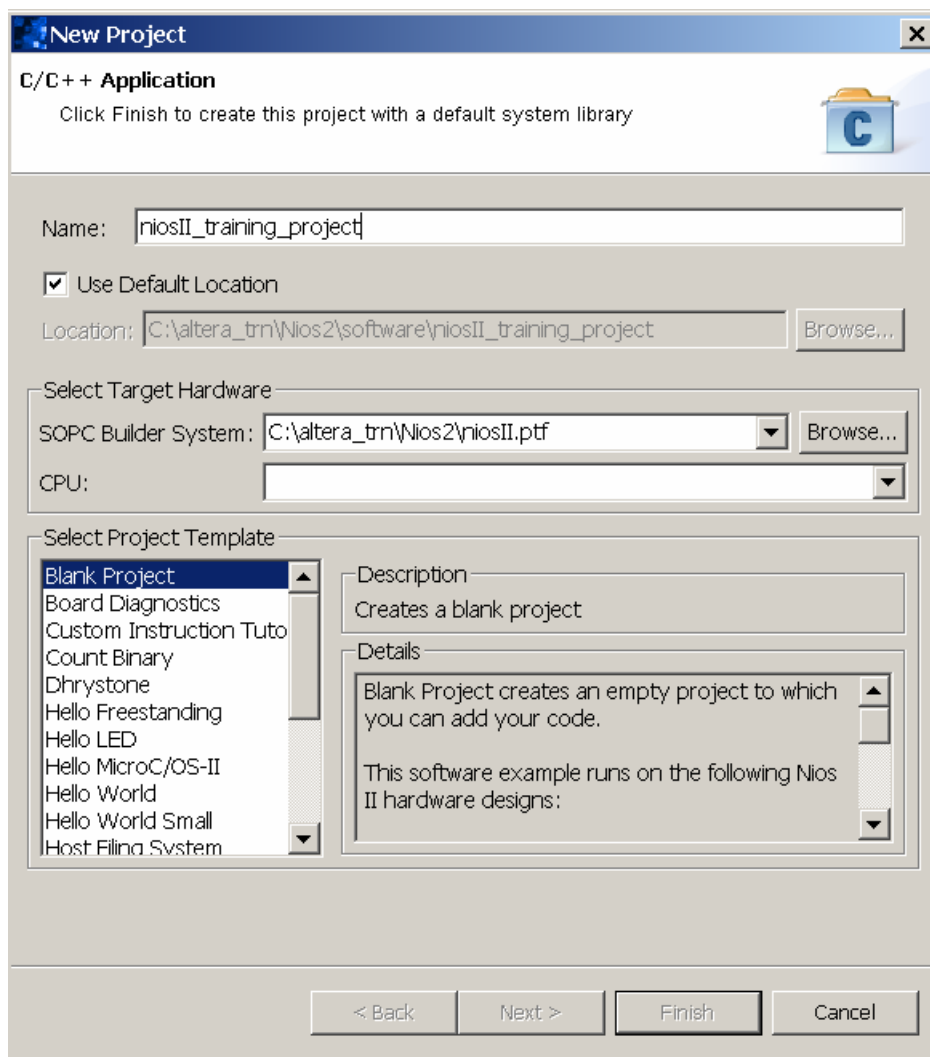
软件设计流程

1. 现在把上面所建立的 Nios II FPGA 设计下载到 Nios 开发板，在 QuartusII 里，从 Tools 菜单中选择 Programmer，假如你工程里的.sof 文件没有装上，点击 Add File 按钮 ，然后选择 DE2Project.sof 并且点击 Open。
2. 为.sof 文件标记 Program/Configure 检测盒，然后点击开始编程的图标 
3. 从 SOPC Builder 中启动 Nios II IDE，从 Tools > Nios II IDE，当 Workspace Launcher 对话框出现时，选择 OK
4. 在 NiosII IDE 的工作平台上，从 File 菜单上选择 New->Project，创建一个新的软件工程，在这 Altera NiosII 子目录中，选择 C/C++ Application。按 Next。

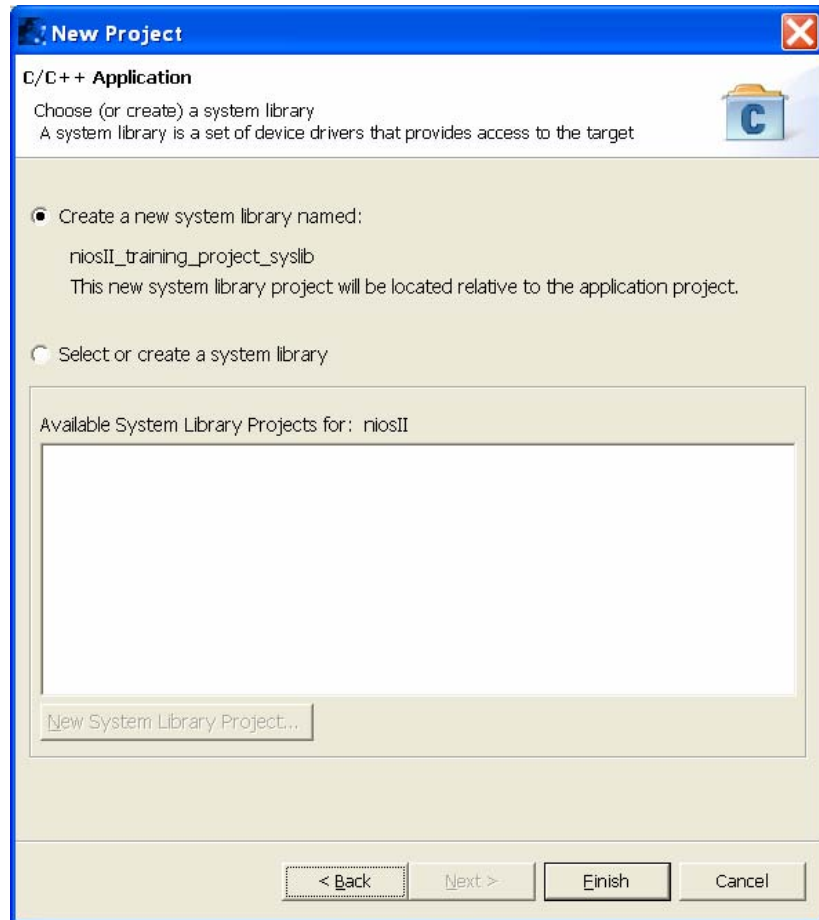


5. 在出现的窗口中，niosII_training_project 作为应用程序的名字被建立，然后，检验 SOPC Builder 中的已经在 Select Target Hardware 框中自动装上 SOPC Builder 三原色 tem 文件的 Ptf 文件，现在从窗口的左边框中的 Select Project Template 选择 blank project 模板，这新的工程窗口类似如下窗口：

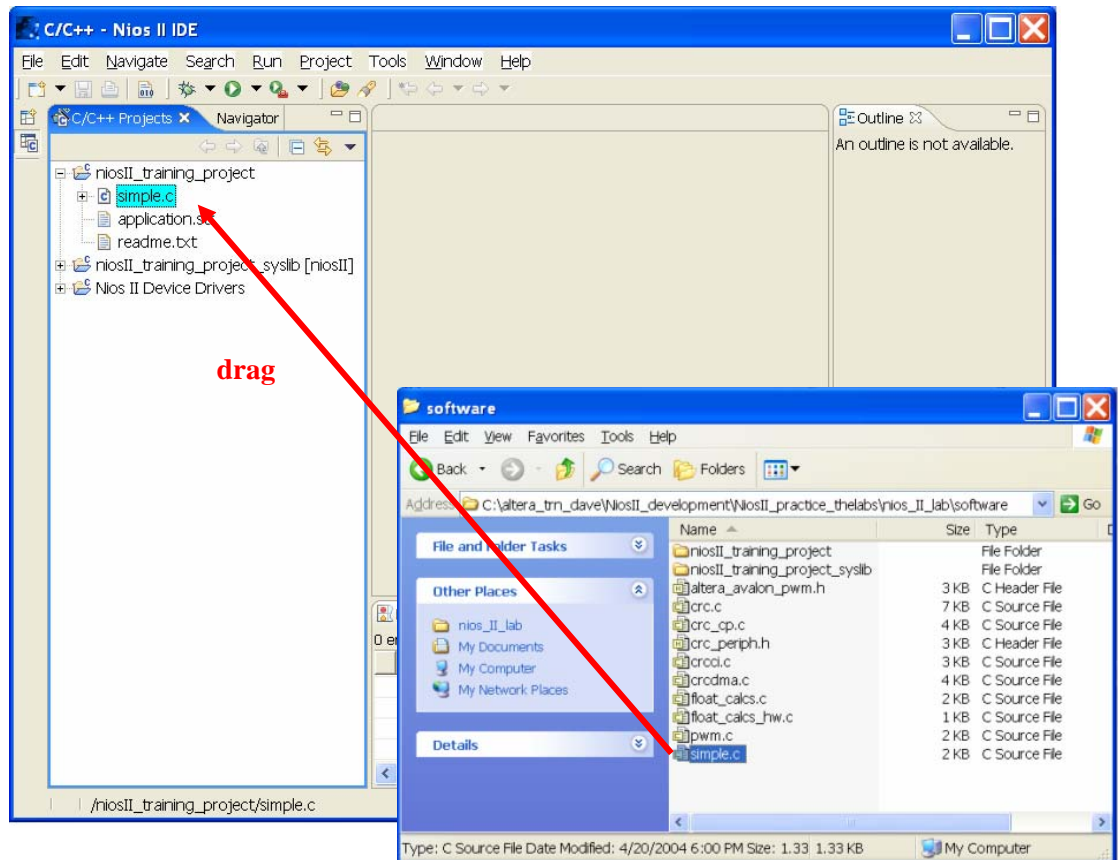
Note: As an alternative, for Step 2 we could have also launched the Nios II IDE from the Windows Start menu >All Programs > Altera > Nios II Development Kit 1.0.



6. 按 Next，指示选择 **Creating a new system library named: niosII_training_project** 检验盒。为你的创建一个新的系统库。点击 **Finish**



7. 现在，你已经在 **NiosII IDE** 中创建了两个新的工程项目，一个 **C/C++应用工程** 和一个 **系统库工程**，但，这 **C/C++应用工程** 是空白的，需要我们去增加添加源代码。
为了添加 **simple.c** 文件去工程，首先要找到并打开 **niosII_training_project** 文件夹，然后找到 **simple.c** (浏览 **C:\altera_trn\Nios2\SoftwareExample.**)现在在 **NiosII IDE** 中把 **simple.c** 拖去 **niosII_training_project** 文件夹，如下所示：

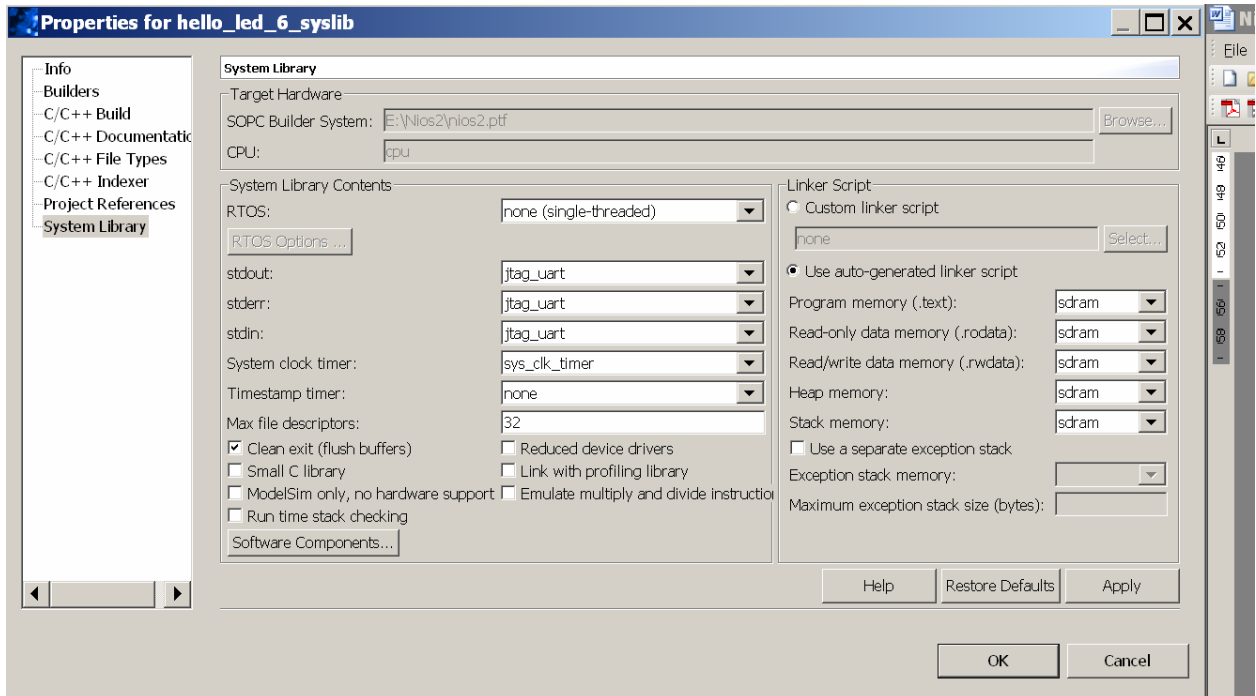


8. 现在，选中 `niosII_training_project_syslib` 文件，点击右键，选择 **properties**，然后从它的左窗口中选择 **system Library**，保证 `stdout`, `stderr` 和 `stdin` 设备被设置为 `jtag_uart`，并且这 **System clock timer** 是设置为 `sys_clk_timer`，设置 **Read-only data**, **Read/write memory** 和 **Program Memory** 为 `sdram`

Please refer to the Figure on the next page.

Note:

With this step, you have created a software project that uses the JTAG UART as the stdio device. You have also placed different parts of the compiled program into different physical memories in the system.



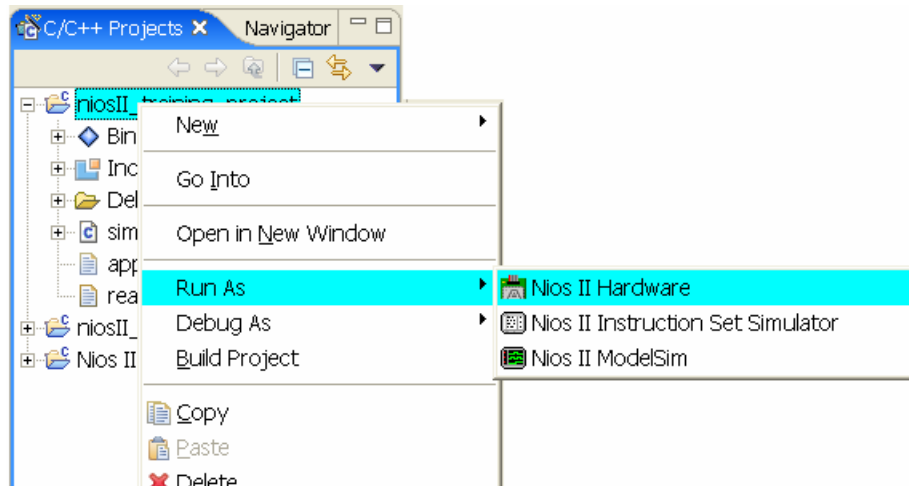
Click OK.

9. 在 C/C++工程窗口里选中 **niosII_training_project** 文件编译程序，然后，点击右键，选择 **Build Project**。
10. 当编译完后，选中 **niosII_training_project**、然后右键点击，选择 **Run As-> Nios II Hardware**。把程序下载到开发板上

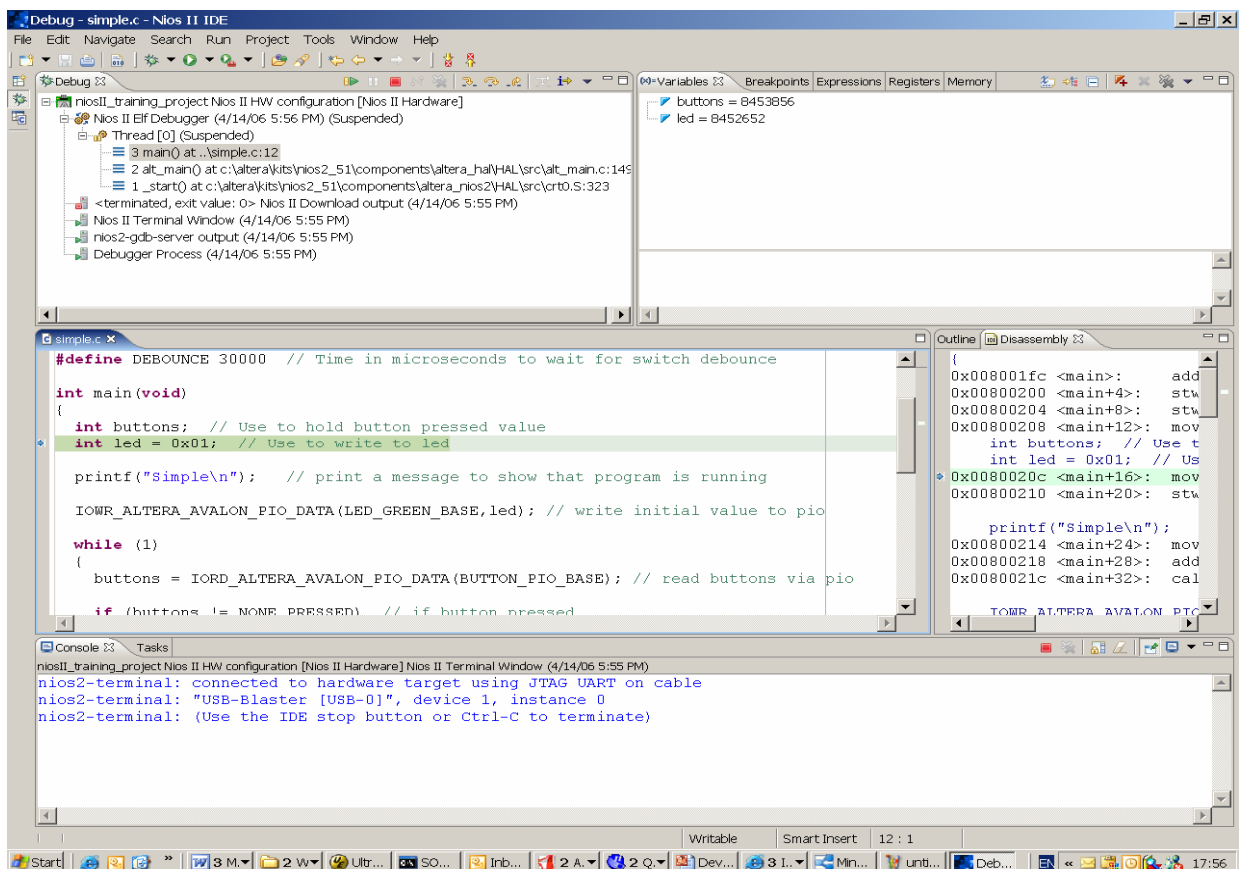
***Note 1:** If Run terminates before the code downloads to your board, and you get a message pertaining to the JTAG download cable, then select **Run > Run...** and from the **Target Connection** tab choose the appropriate download cable that you are using and then press **Apply** and **Run**.*

***Note 2:** The Nios II IDE will actually Build the project automatically for you if you just click **Run As -> Nios II Hardware** without you having to explicitly go through the Build step. You can enable or disable this option in the **Window > Preferences**.*


This will download the program to the development board. You should observe that the console window in the Nios II IDE displays the printf statement from the simple.c file. Now, press of any of the buttons on the board, and the lit LED should shift to the right.



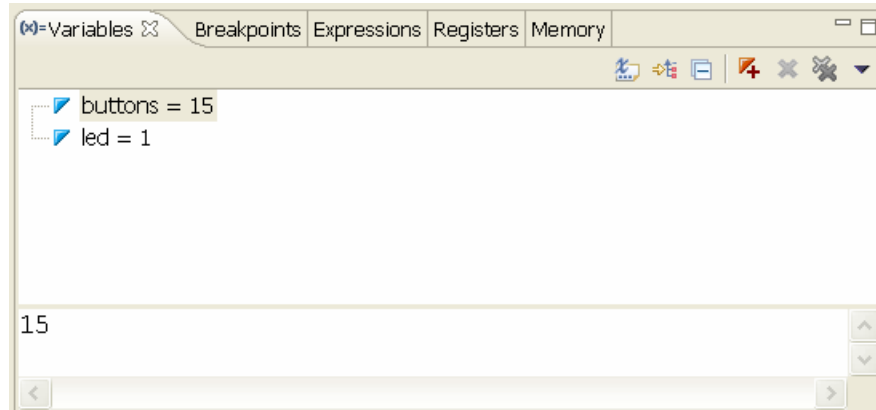
11. 现在，我们开始在这个设计中运行调试器，并按步从头到尾调试某些代码，启动这调试器：选 **niosII_training_project** 软件工程文件 然后右键点击，选择 **Debug As-> Nios II Hardware**，这调试器将会启动，连接目标板，并且下载程序为调试作准备。




12. 在第 22 行和第 37 行设立断点，（假如编制文件里没有显示行 的数，我们需要去设置 window 里的 preferences，打开 C/C++ 文件夹，选择 Editor，核对 Show line numbers 盒，并且按 OK）

To set a breakpoint, simply place the cursor in the grey area to the left of the line number and double-click. A circle should now appear next to the line number indicating that a breakpoint is set. Click on the **resume** button .

13. 现在，选择 **variables** 窗口，看按键变量的内容，注意按键的值已经被读，显示在变量窗口里，通过右击按键的值，选择 **Format =>Hexadecimal.**，改换为十六进制。



14. 再次点击 **resume** ，由于 **if** 的表达式是假的（没有按键按下），在波形括号里的句子没有被执行，我们将不能在 37 行上断点。而是在 22 行已经断点了。
15. 保持板上 SW3 的开关为按下，点击 **resume**，一个新的值被存储在变量 **buttons** 上。
16. 连续去保持 SW3 为按下状态，再次点击 **resume**，在第 37 行的断点当按键被按下的时候被获取，所以这 **IF** 的条件是真的，注意，这 **LED** 的灯亮位置已经被改变。
17. 点击 **resume**
18. 在这 **variables** 窗口里双击 **Buttons**，现在改值为 **0xe**，后按 **OK**，点击 **resume**，因为我们改了按键的值，所以 **IF** 语句被执行。
This is useful for emulating external hardware events or other conditions that are difficult to replicate!
19. 终止这进程。在 **Debug** 的子菜单中看，然后右击正在运行的软件工程，并且选择 **Terminate and Remove.**

