



现代计算机组成原理

潘明 潘松 编著

科学出版社



第 10 章

NiosII嵌入式系统 软硬件设计

10.1 NiosII基本硬件系统构建

10.1.1 设计模型准备

Nios II Hardware Tutorial - Cyclone FPGA
GW48-SOPC
Mode : No.5

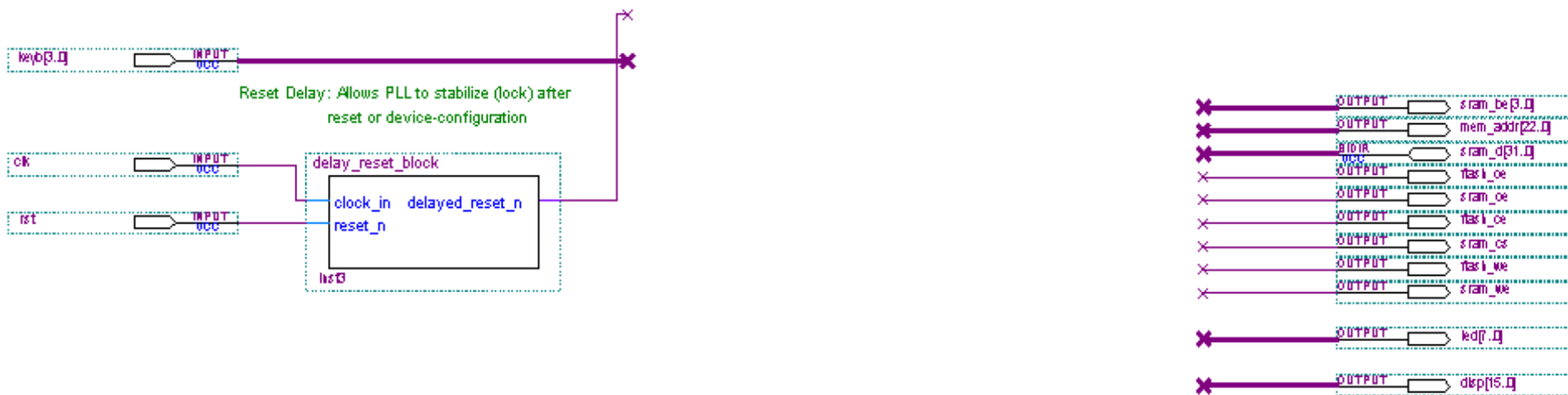


图10-1 NiosII系统模型

10.1 NiosII基本硬件系统构建

10.1.1 设计模型准备

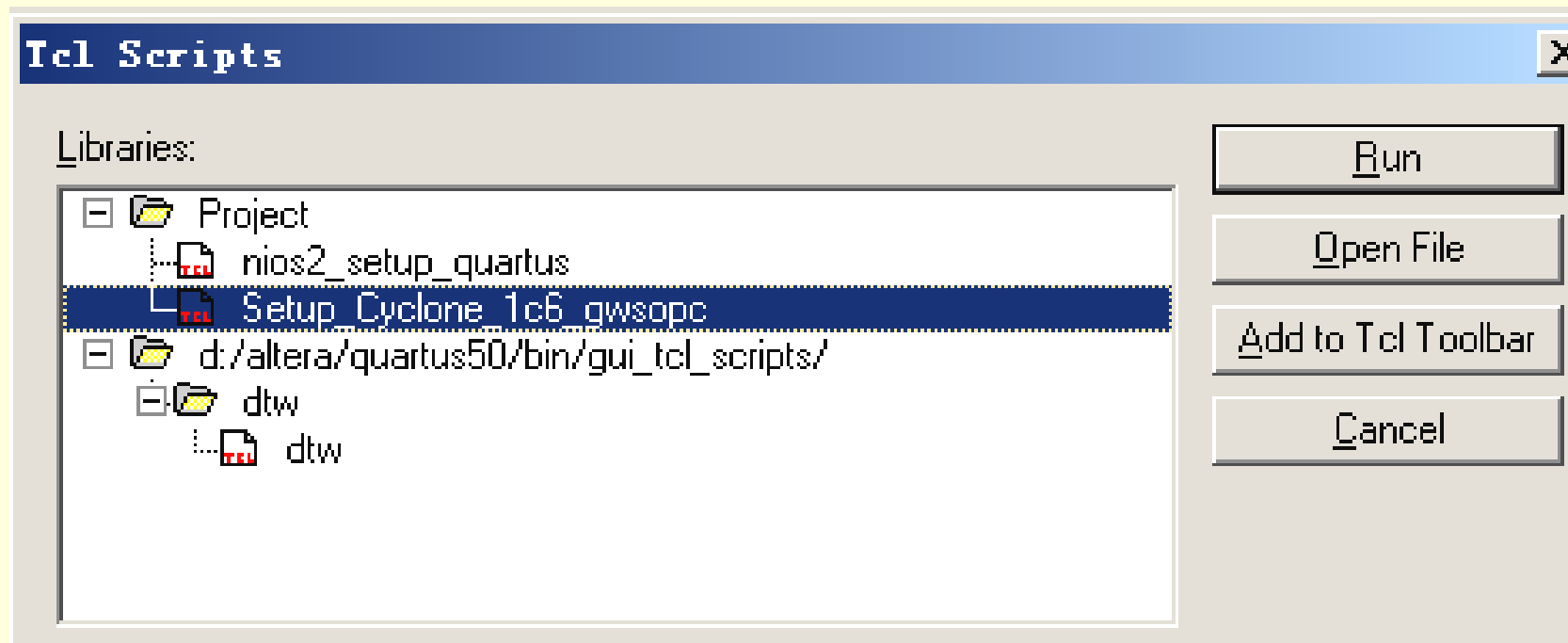


图10-2 选择Tcl文件

10.1 NiosII基本硬件系统构建

【例10-1】 `setup_cyclone_lc6_gwsopc.tcl`

```
# Load Quartus II Tcl Project package
package require ::quartus::project
set_global_assignment -name FAMILY Cyclone
set_global_assignment -name DEVICE EP1C6Q240C8
set_global_assignment -name RESERVE_ALL_UNUSED_PINS "AS INPUT
TRI-STATE"

set_location_assignment PIN_1 -to led[0]
set_location_assignment PIN_2 -to led[1]
set_location_assignment PIN_3 -to led[2]
set_location_assignment PIN_4 -to led[3]
set_location_assignment PIN_6 -to led[4]
. . .
set_location_assignment PIN_164 -to disp[27]
set_location_assignment PIN_165 -to disp[28]
set_location_assignment PIN_166 -to disp[29]
set_location_assignment PIN_167 -to disp[30]
set_location_assignment PIN_168 -to disp[31]
```

10.1 NiosII基本硬件系统构建

10.1.1 设计模型准备

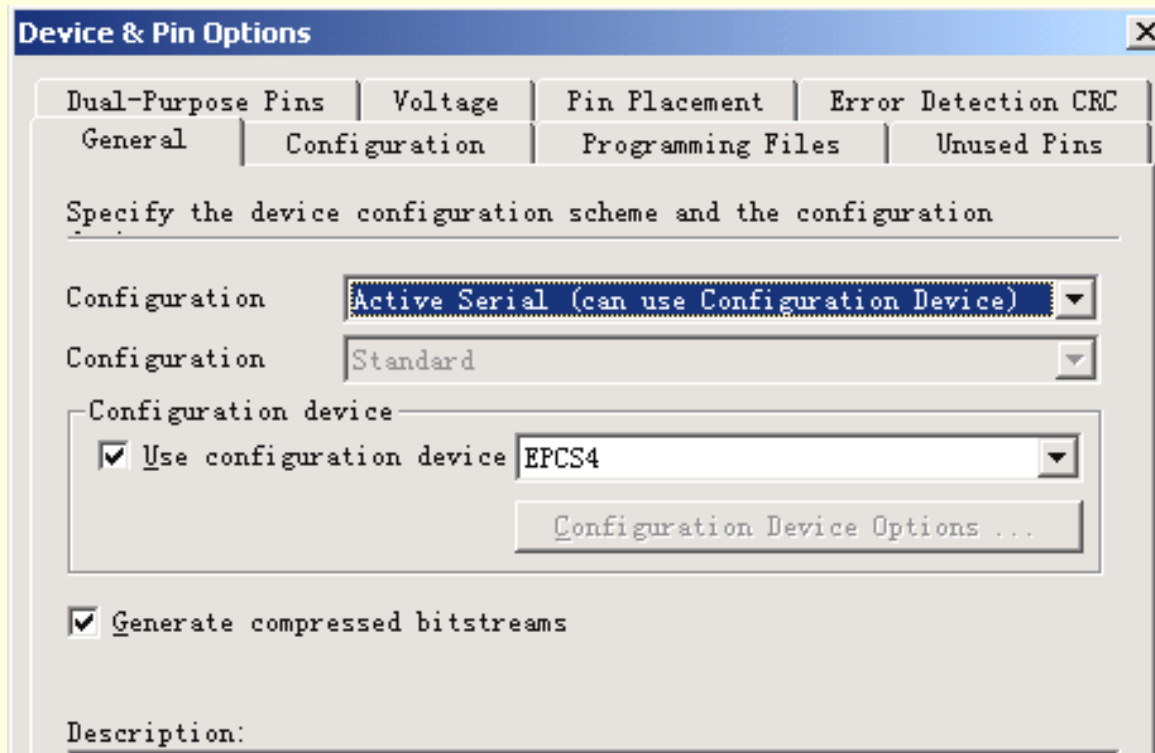


图10-3 确定配置器件EPCS4

10.1 NiosII基本硬件系统构建

10.1.1 设计模型准备

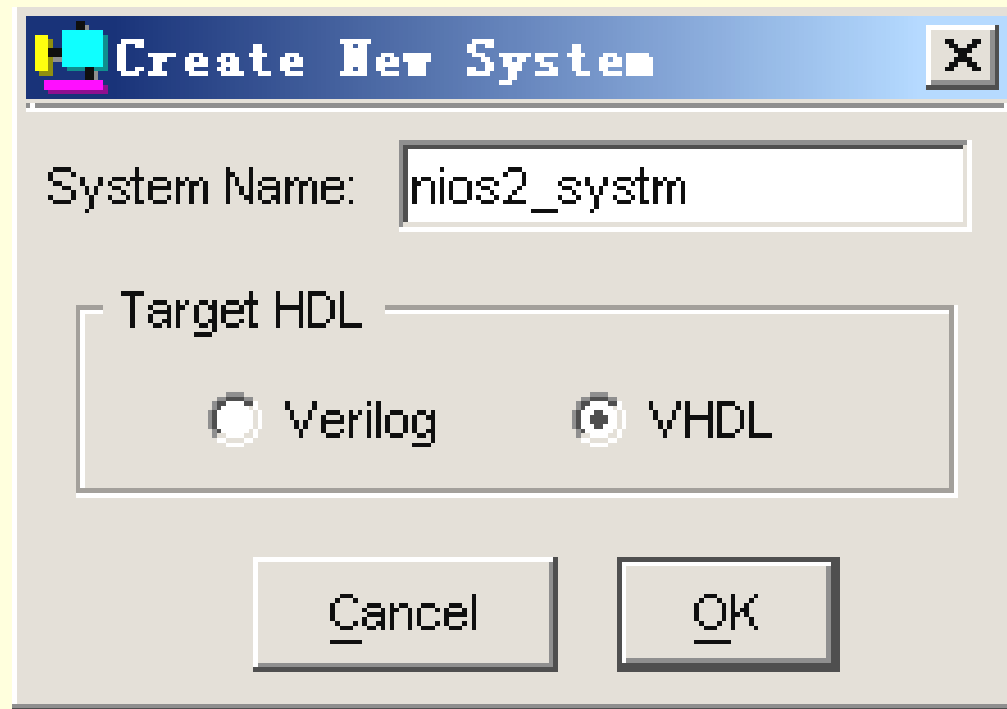


图10-4 建立一个新的系统

10.1 NiosII基本硬件系统构建

10.1.1 设计模型准备

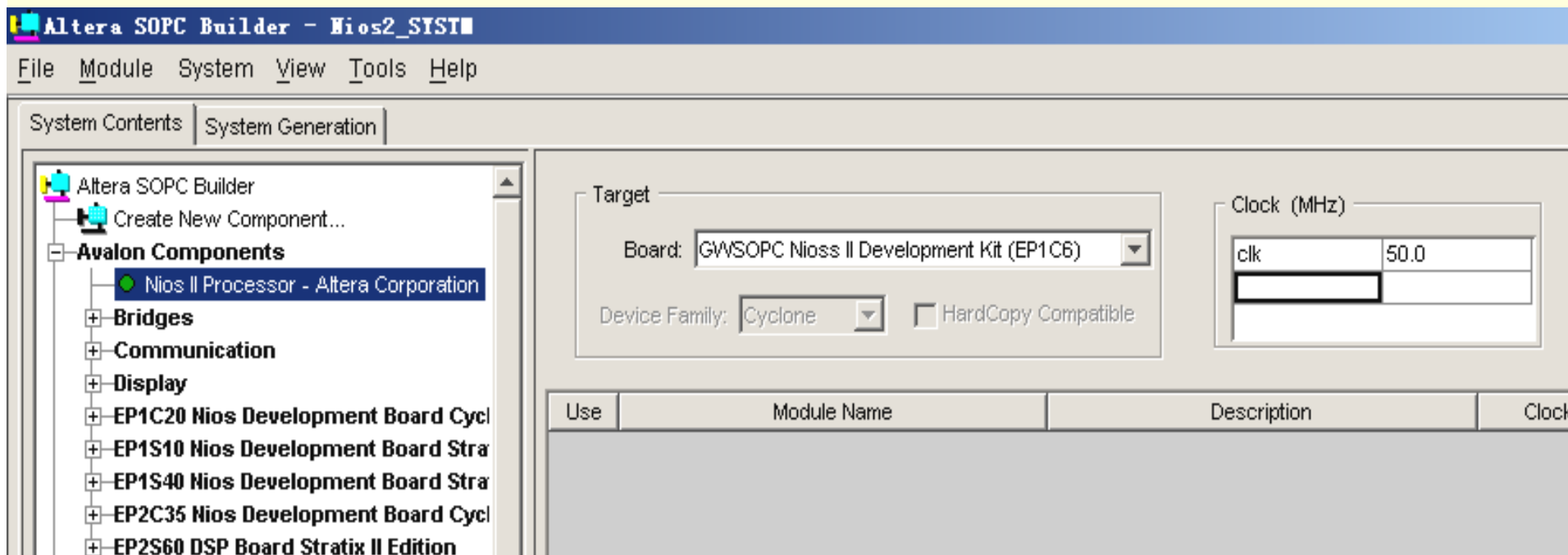


图10-5建立一个SOPC系统模块

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

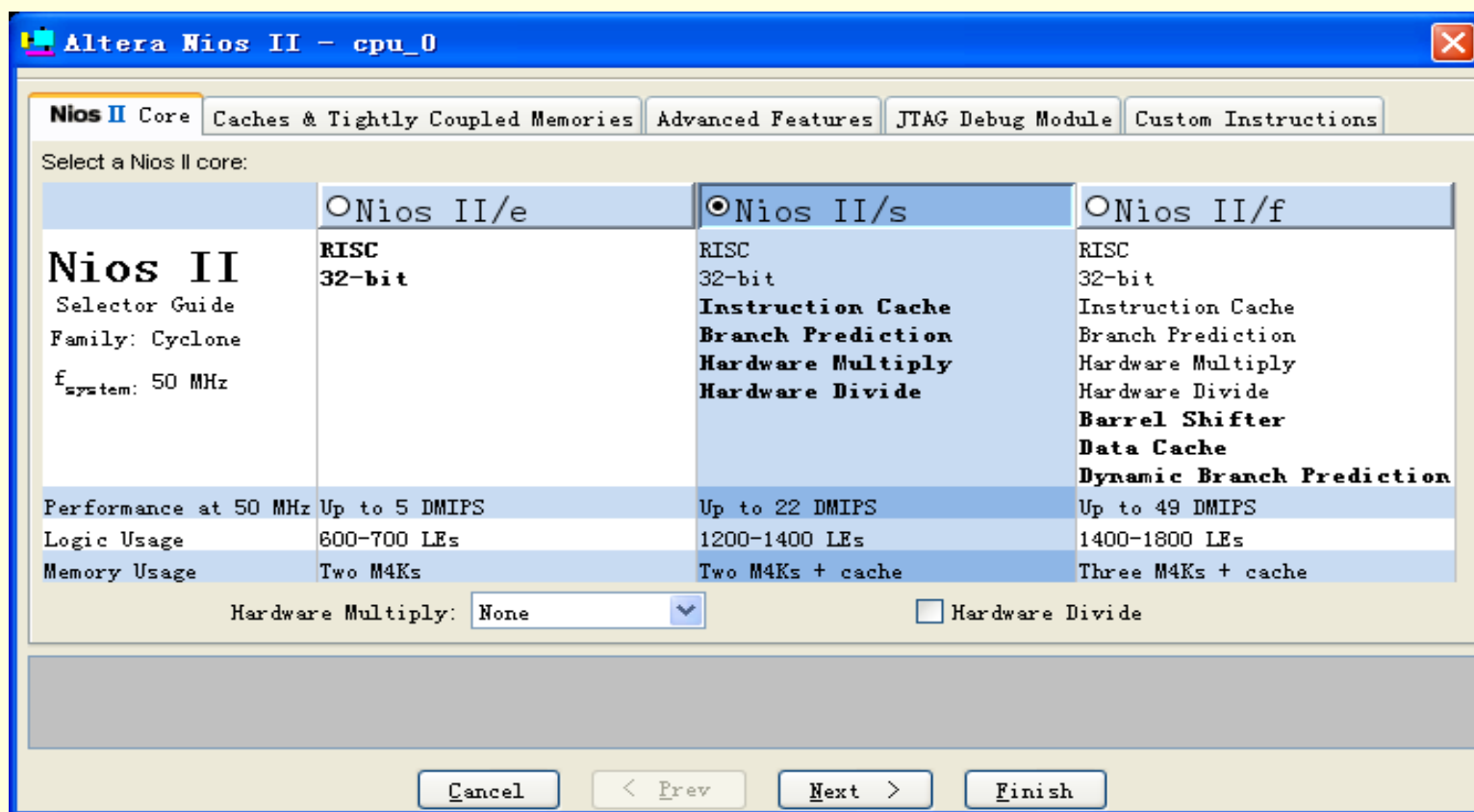


图10-6 Nios II/s CPU模式

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

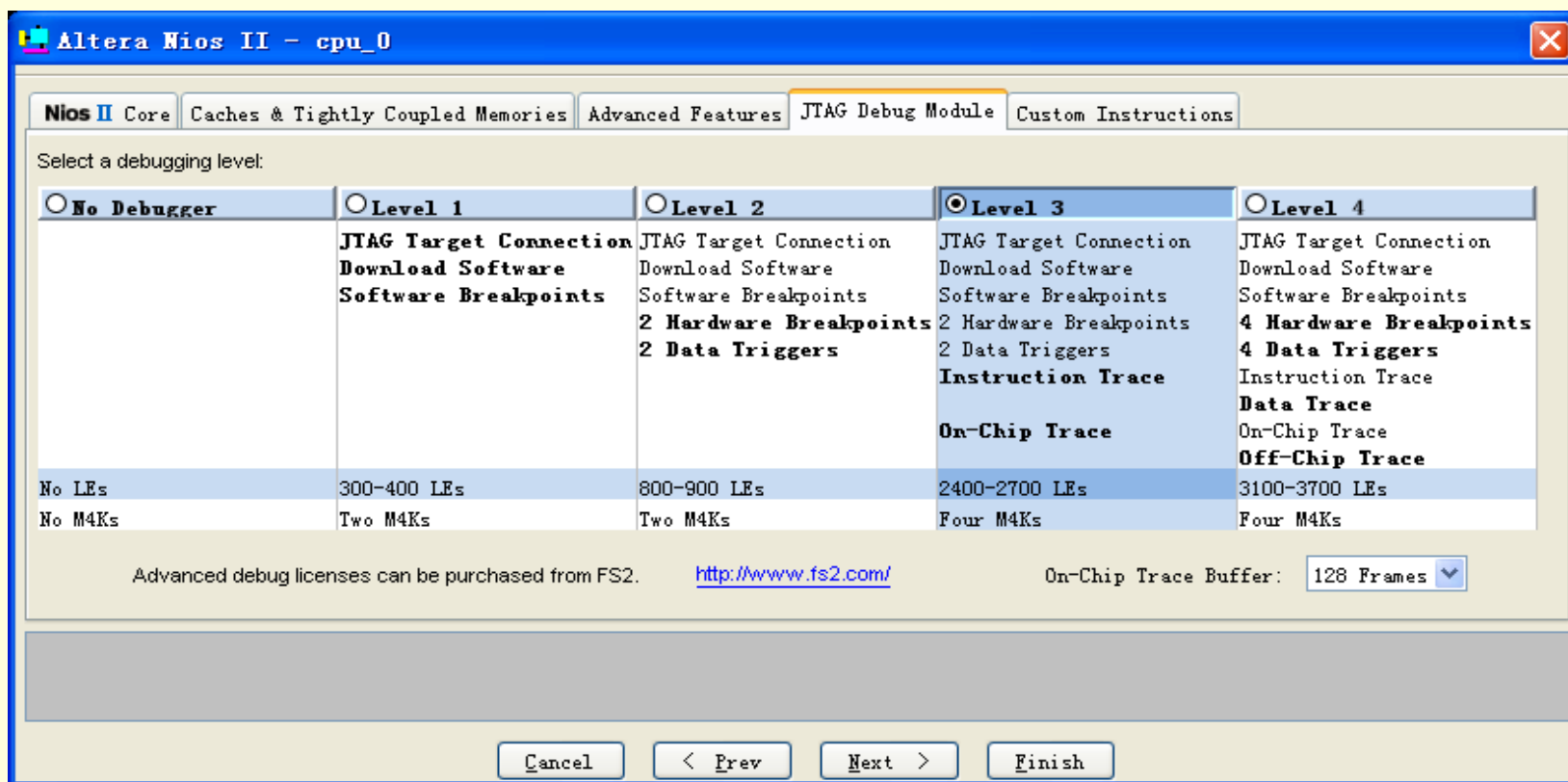


图10-7 选择JTAG的调试模式

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

The screenshot shows the configuration window for a NiosII Sopc Builder. It includes a 'Target' section with a 'Board' dropdown set to 'GWSOPC Nios II Development Kit (EP1C6)', a 'Device Family' dropdown set to 'Cyclone', and an unchecked 'HardCopy Compatible' checkbox. A 'Clock (MHz)' table shows 'clk' at 50.0 MHz. Below is a table of components:

Use	Module Name	Description	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>	<input type="checkbox"/> CPU	Nios II Processor - Altera Corporation	clk			
	instruction_master	Master port				
	data_master	Master port		IRQ 0	IRQ 31	
	jtag_debug_module	Slave port		0x00000000	0x000007FF	

图10-8 加入了NiosII的SOPC窗口

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

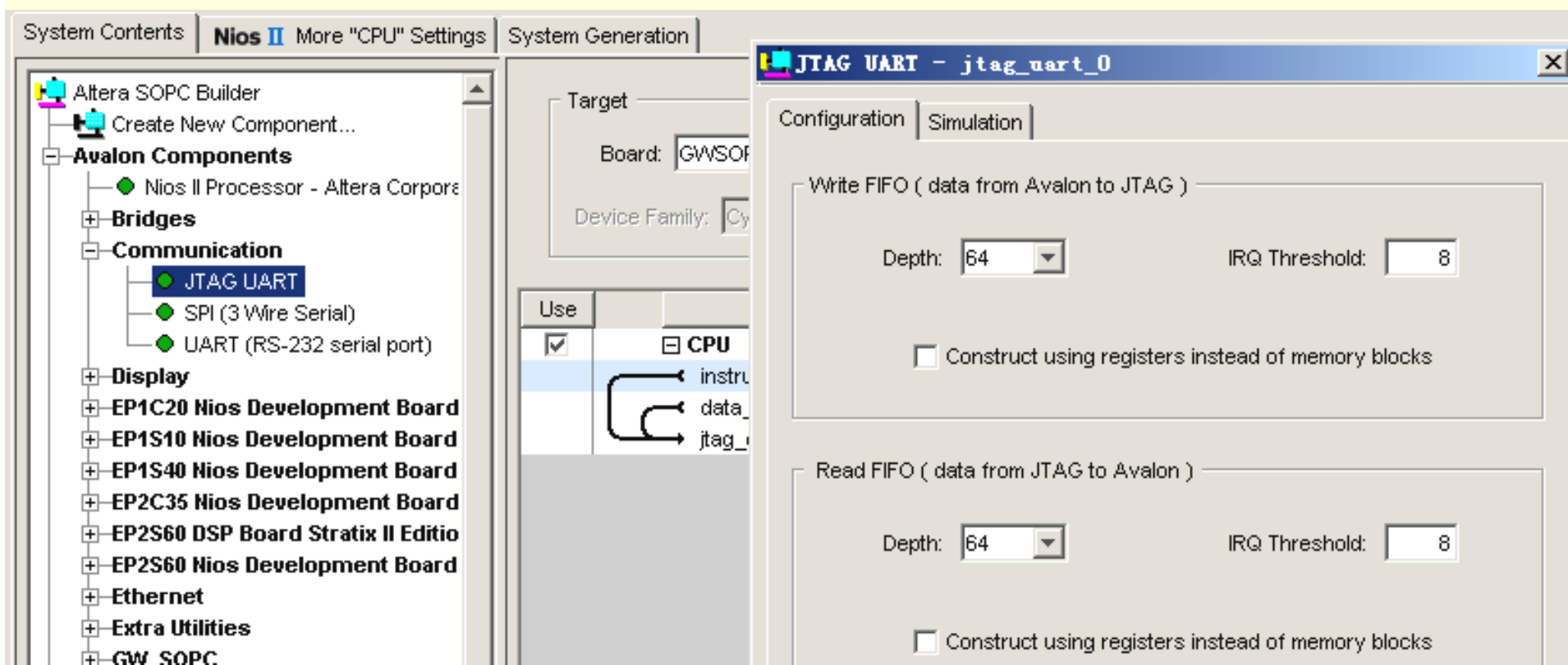


图10-9 组件JTAG UART设置

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

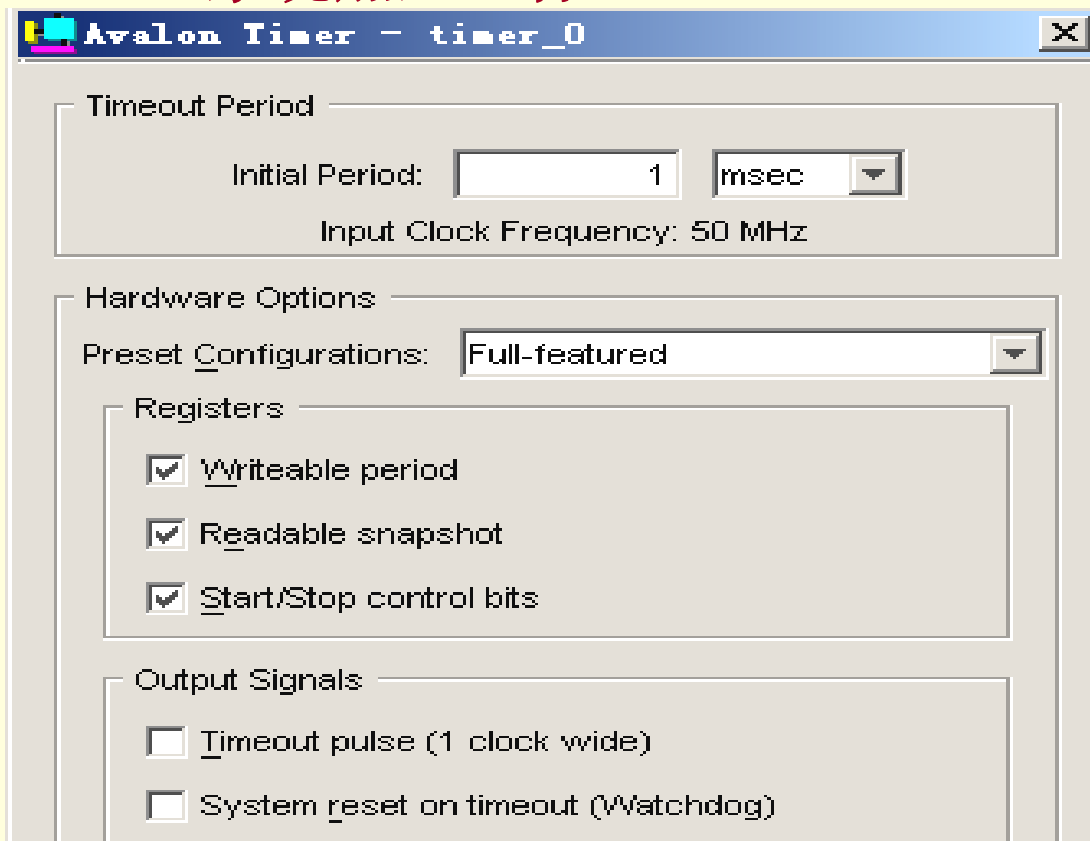


图10-10 组件Timer设置窗

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

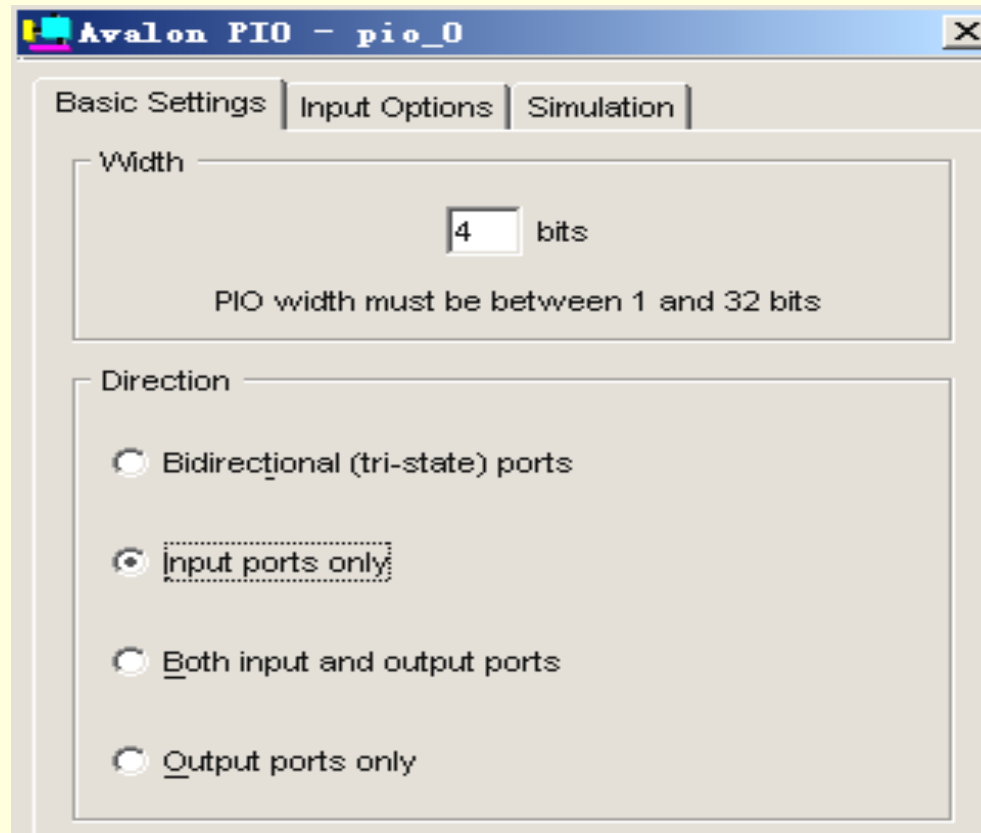


图10-11 组件PIO输入口设置窗

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

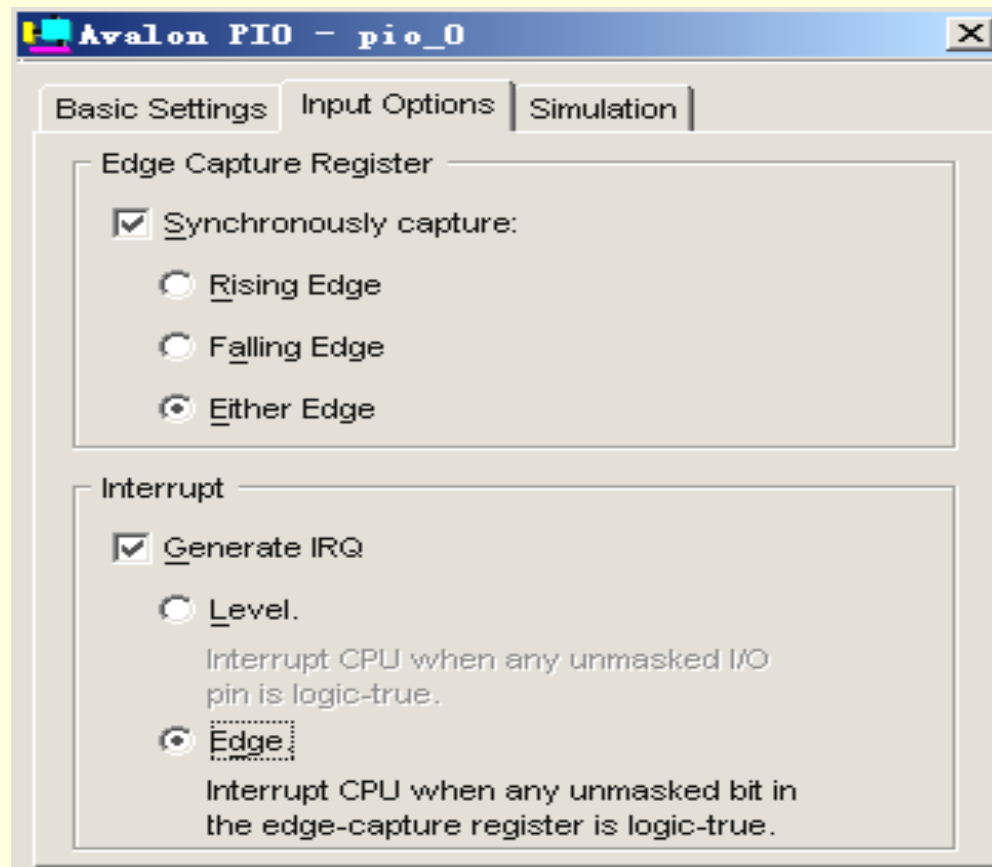


图10-12 输入方式设置窗

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

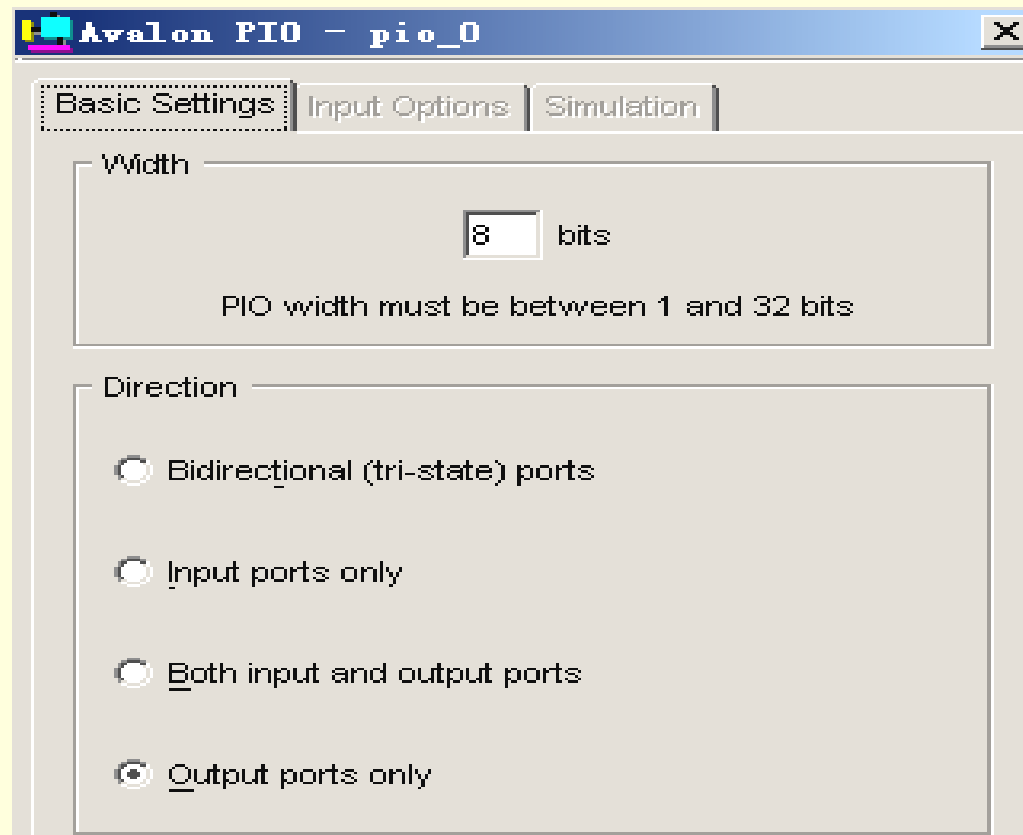


图10-13 加入8个输出PIO口

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

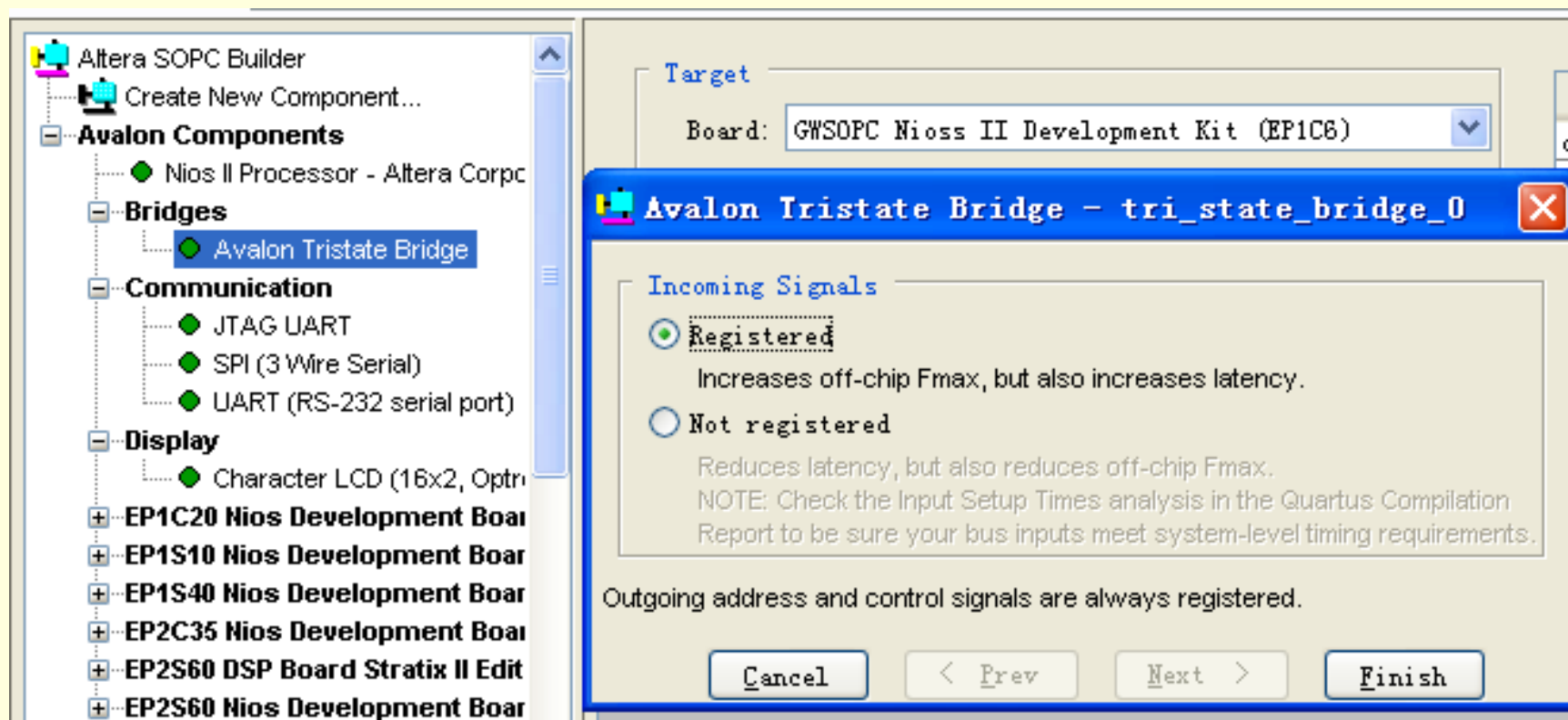


图10-14 加入Avalon总线3态桥设置

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

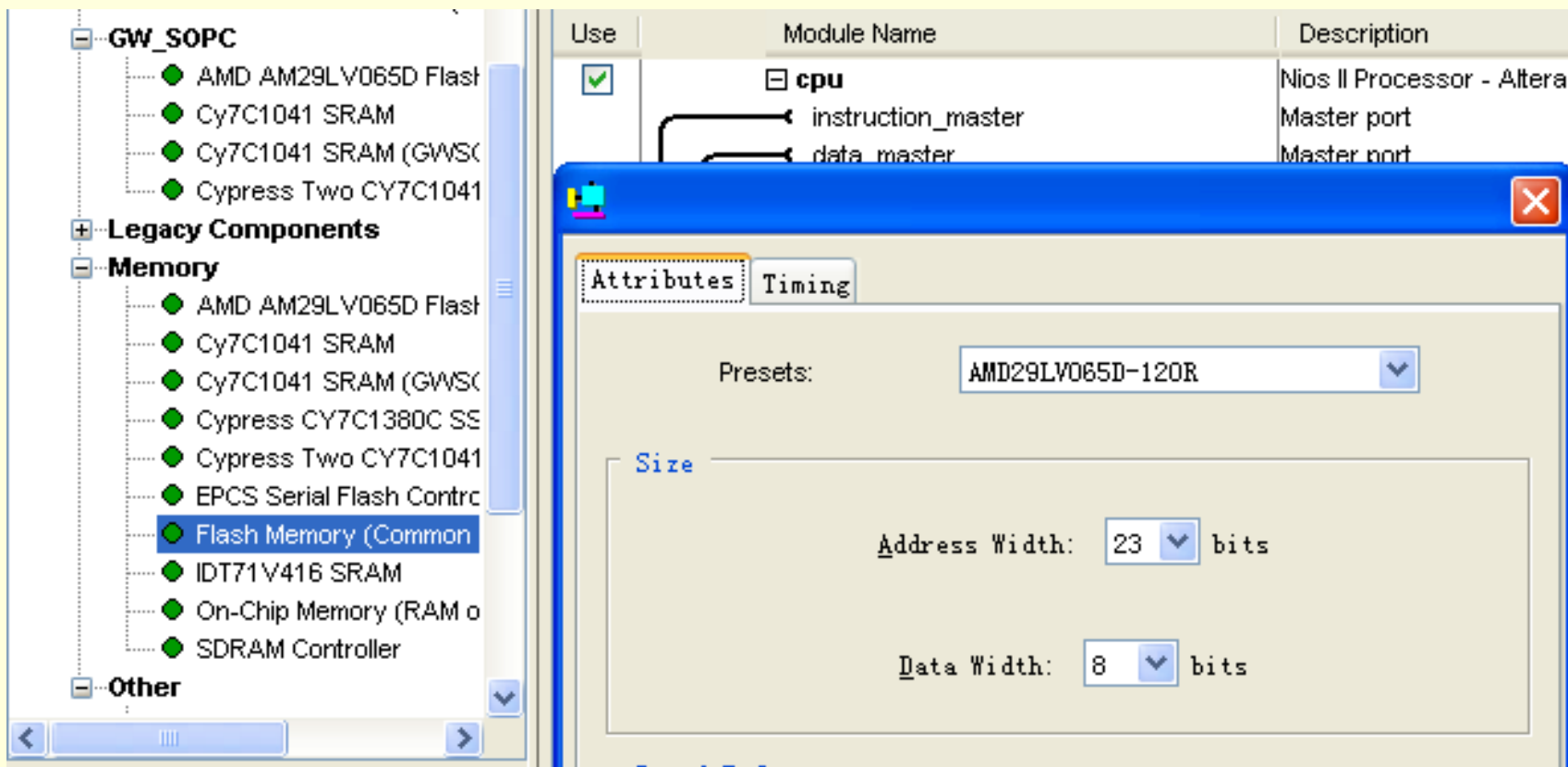


图10-15 加入外部Flash组件的设置窗

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

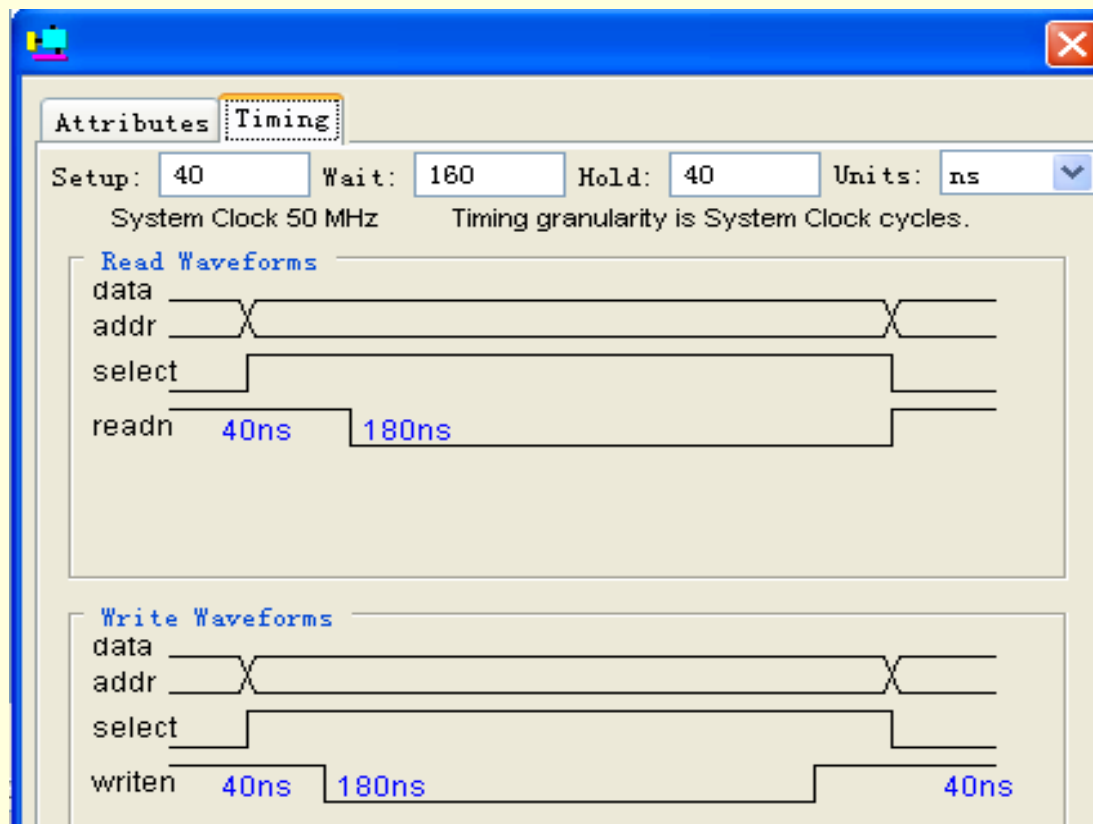


图10-16 设置对外部Flash读写时序

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

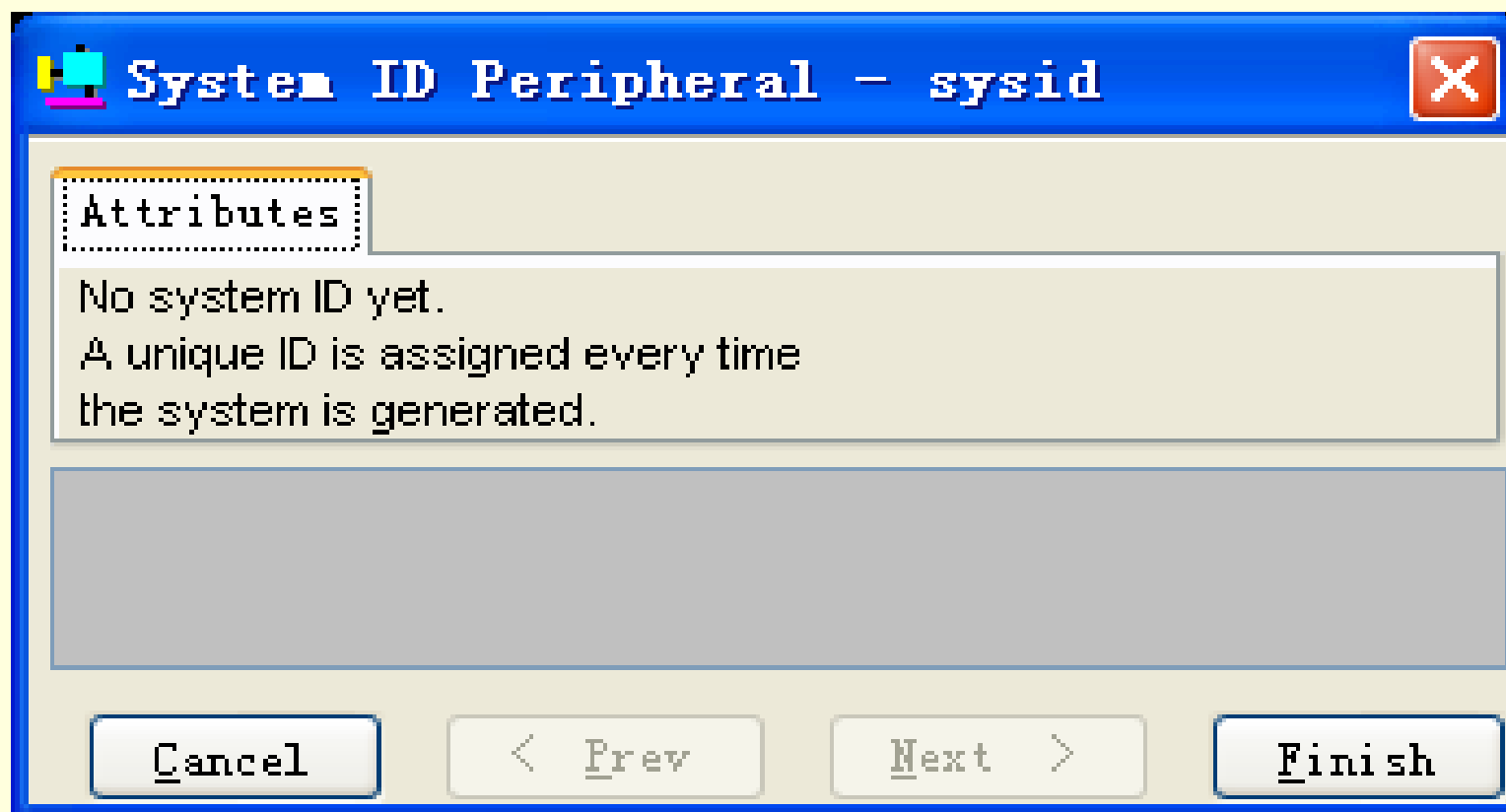


图10-17 加入系统ID组件

10.1 NiosII基本硬件系统构建

10.1.2 NiosII系统加入组件

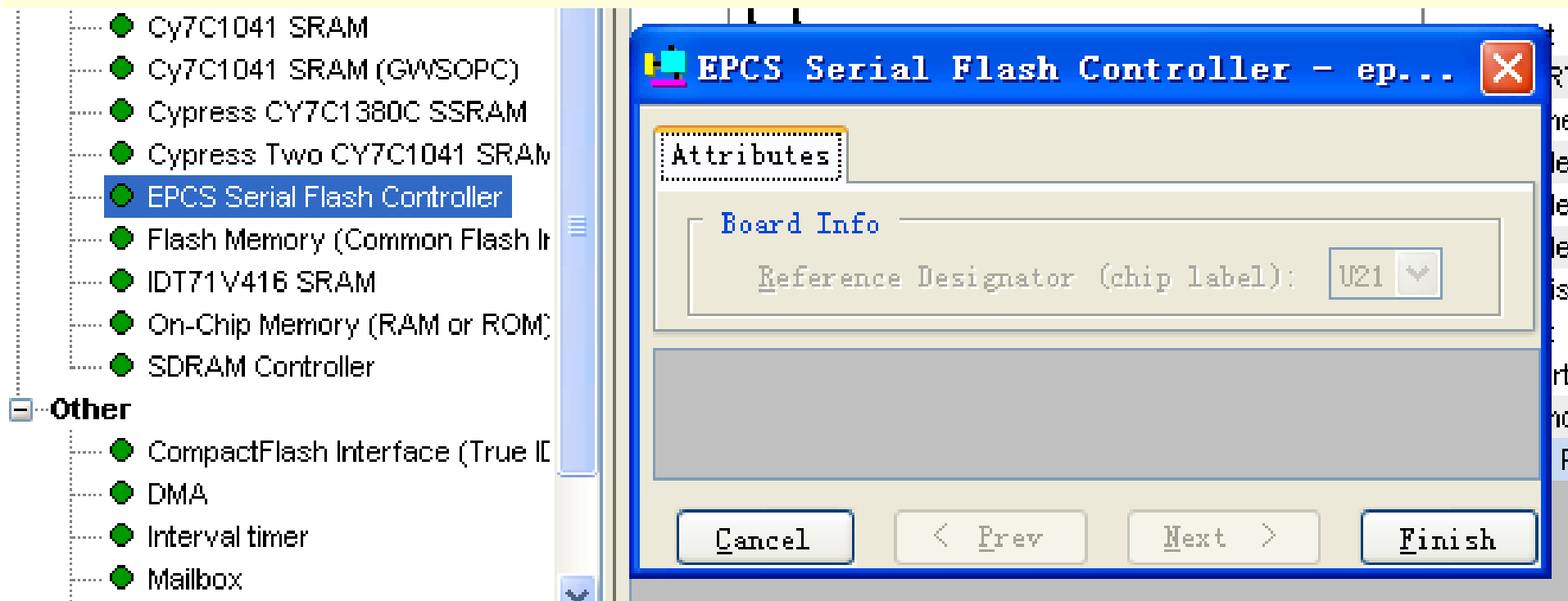


图10-18 EPCS Serial Flash Controller组件

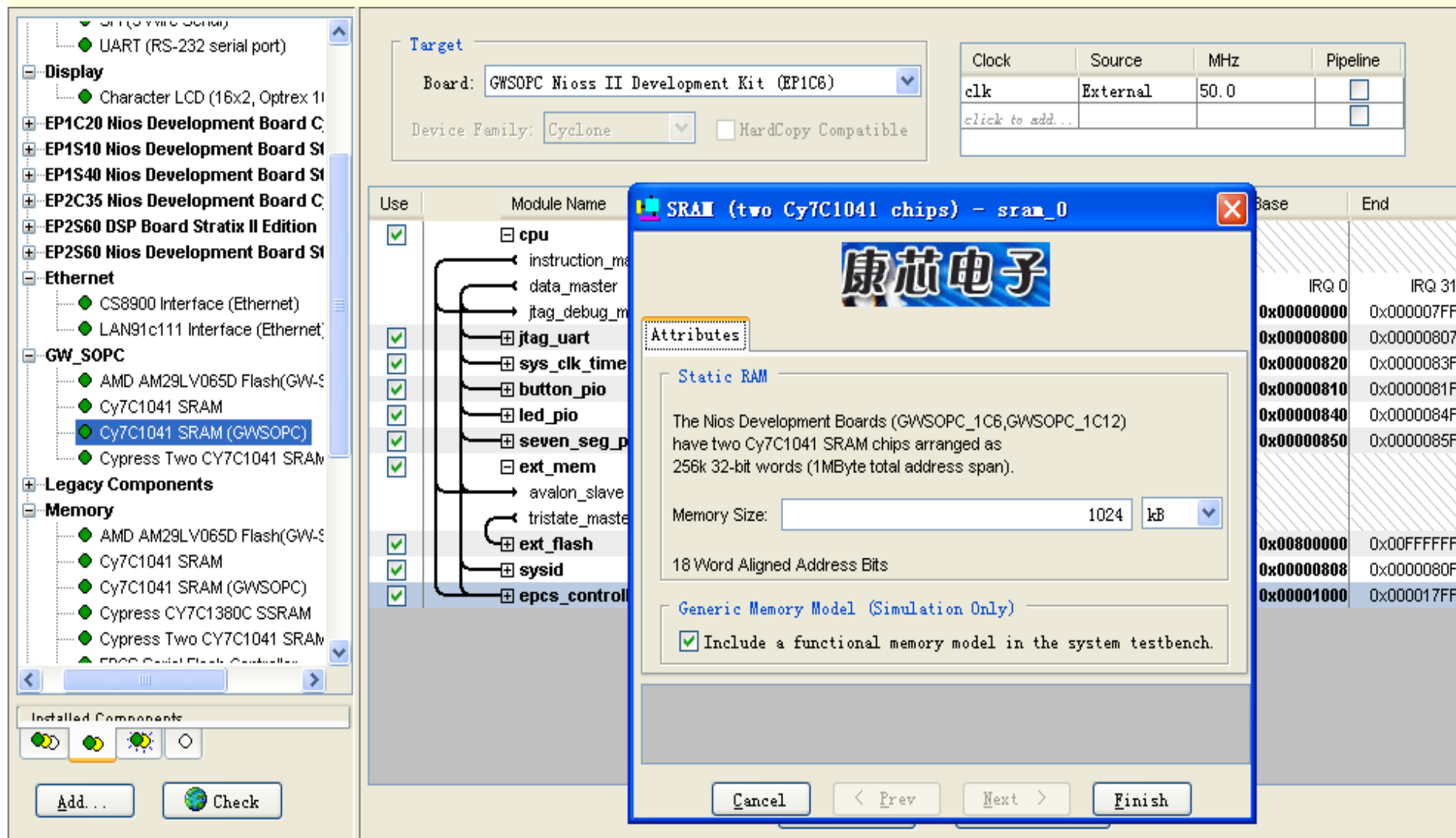


图10-19 加入SRAM组件

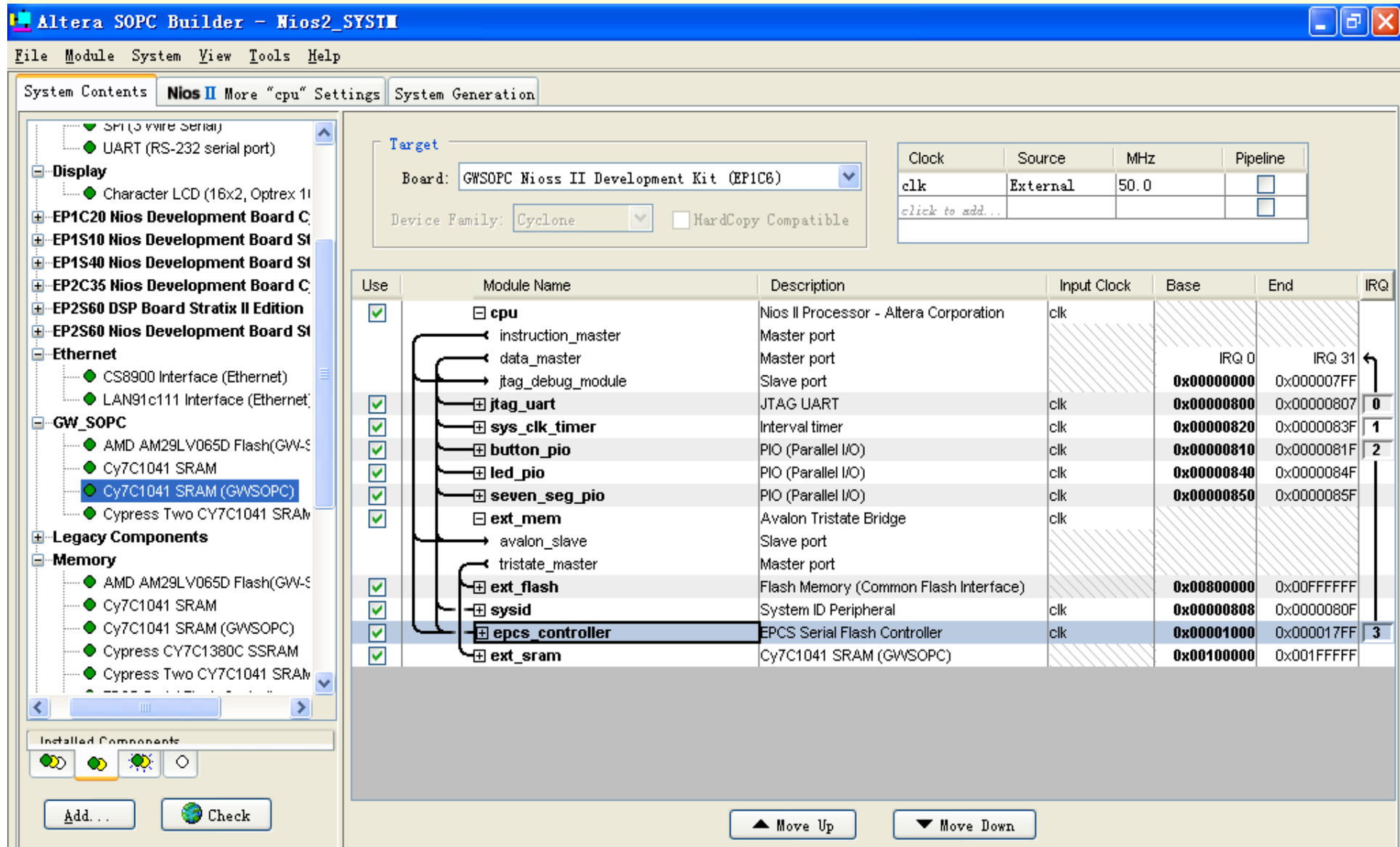


图10-20 本项设计NiosII完整组件窗

10.1 NiosII基本硬件系统构建

10.1.3 NiosII系统生成前设置与系统生成

1、地址自动分配设置

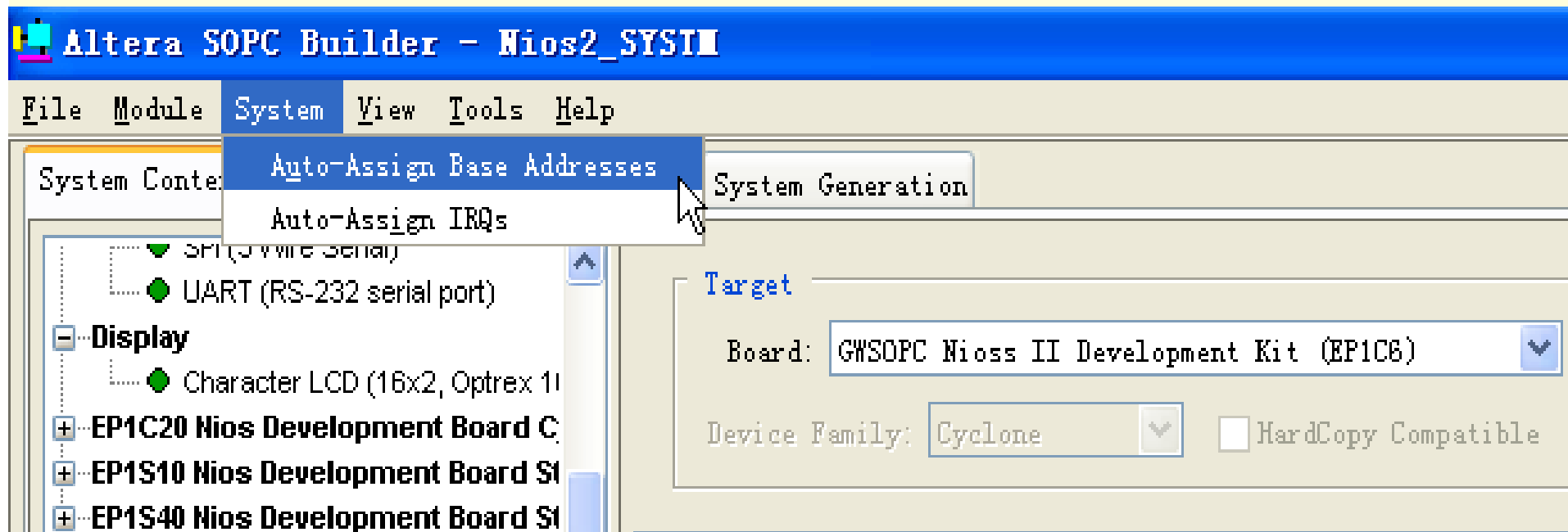


图10-21地址自动分配设置

10.1 NiosII基本硬件系统构建

10.1.3 NiosII系统生成前设置与系统生成

2、复位地址和程序运行区域置

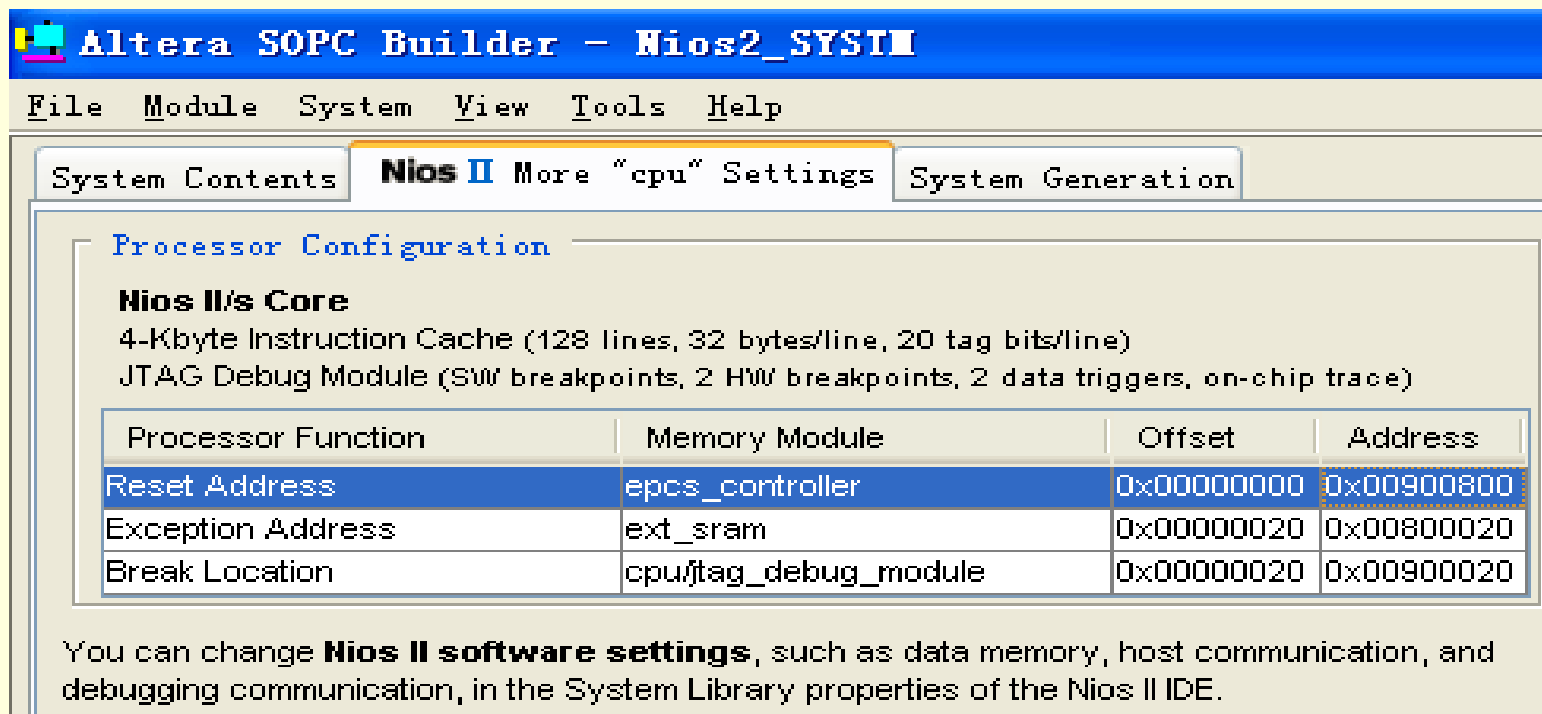


图10-22 NiosII处理器配置窗

10.1 NiosII基本硬件系统构建

10.1.3 NiosII系统生成前设置与系统生成

3、系统文件生成

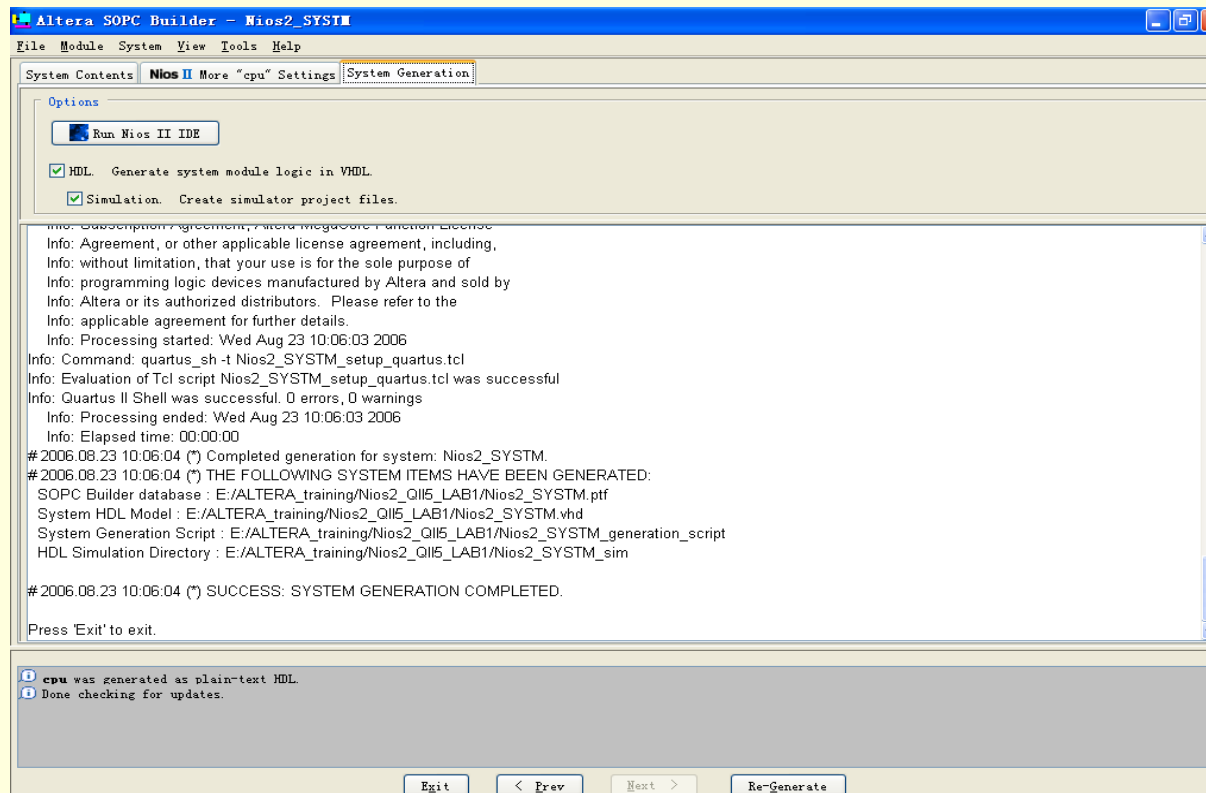


图10-23 Nios II系统生成窗

10.1 NiosII基本硬件系统构建

10.1.4 NiosII硬件系统生成

1、加入原理图元件模块

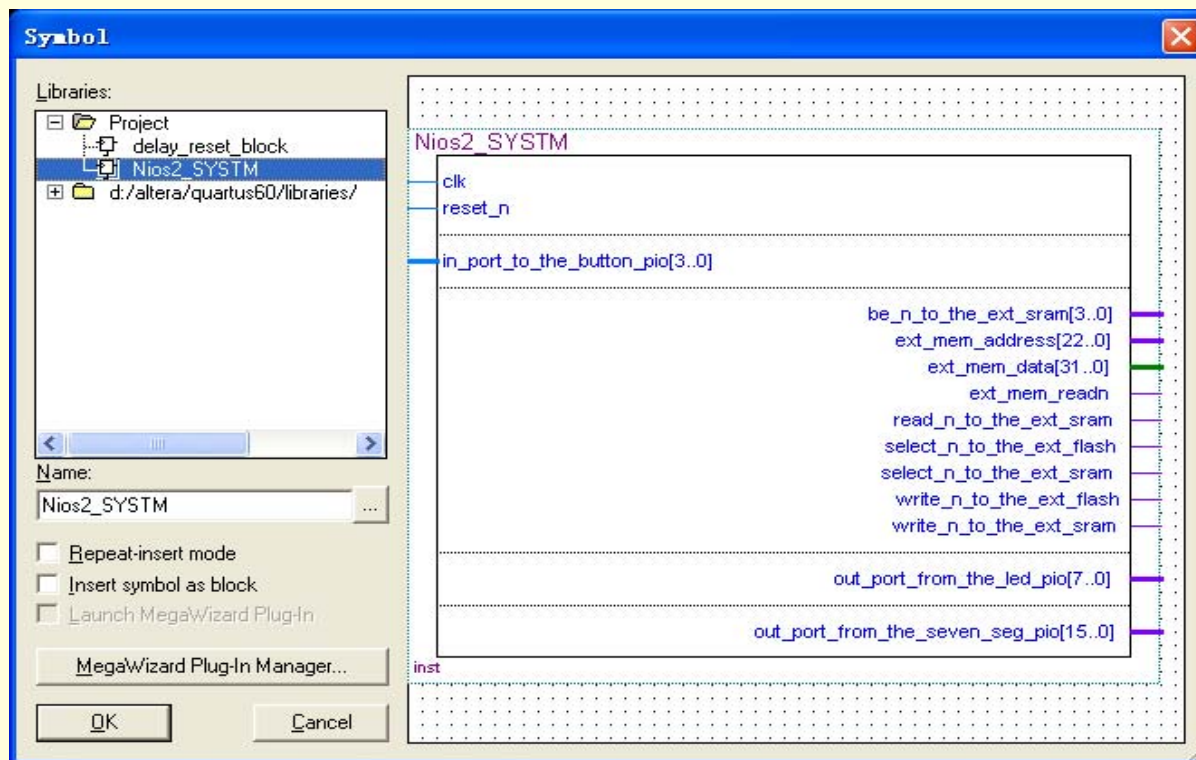


图10-24 Nios II系统生成窗

10.1 NiosII基本硬件系统构建

10.1.4 NiosII硬件系统生成

1、加入原理图元件模块

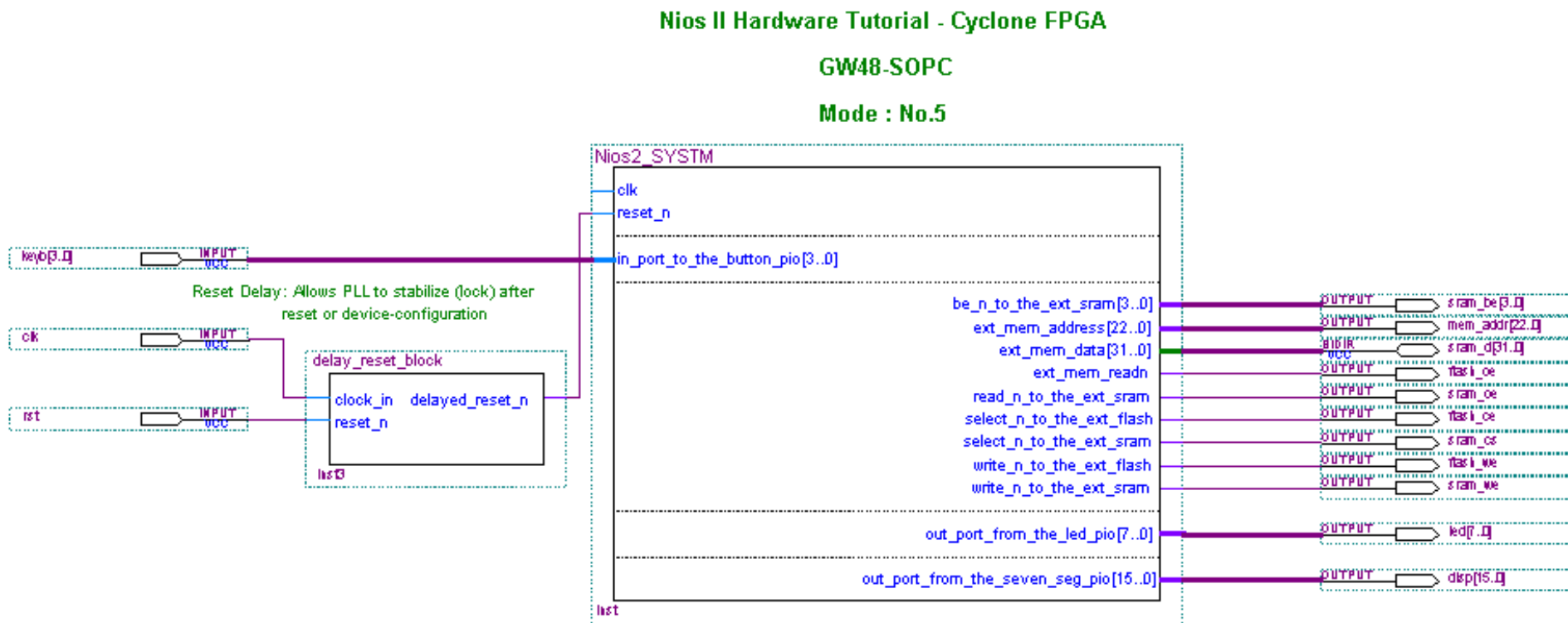


图10-25 连上Nios2_System模块

10.1 NiosII基本硬件系统构建

10.1.4 NiosII硬件系统生成

2、加入锁相环

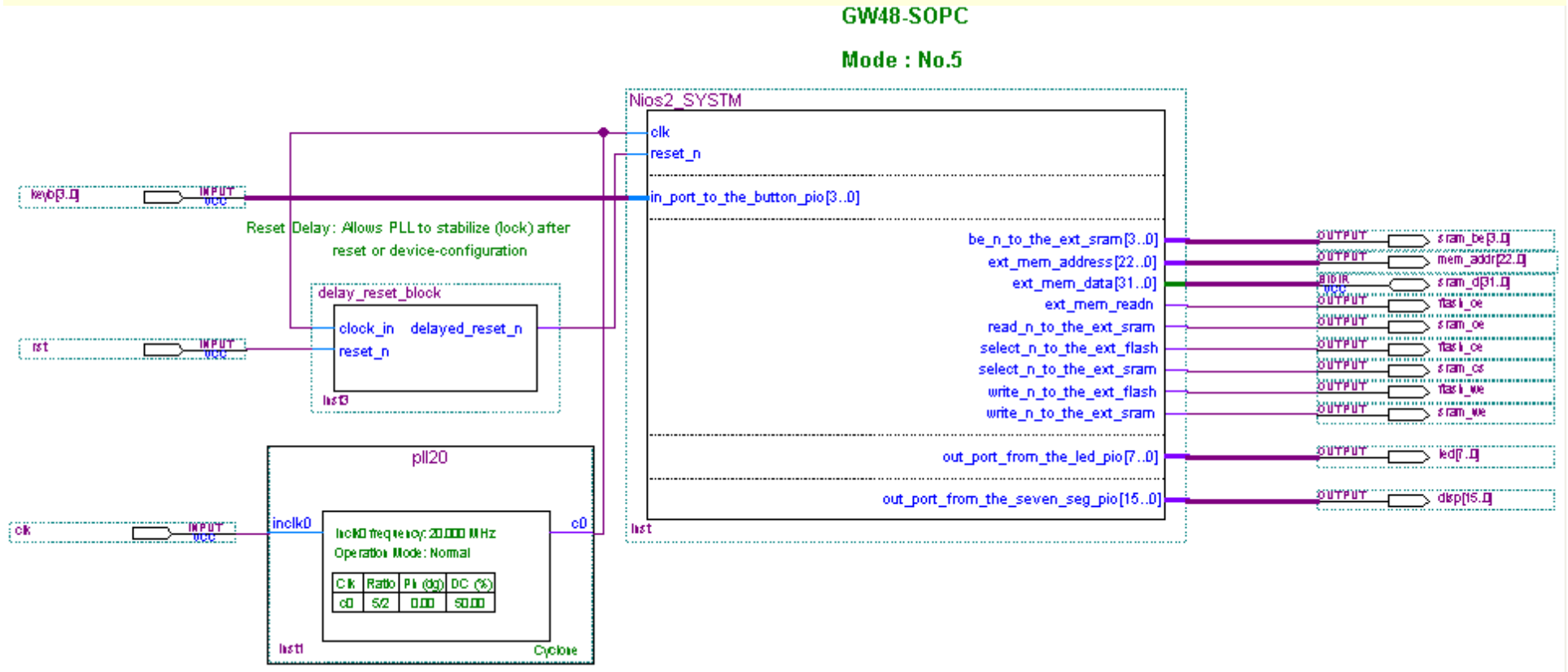


图10-26 将锁相环PLL20连接到时钟输入端

10.1 NiosII基本硬件系统构建

10.1.4 NiosII硬件系统生成

3、编译

The screenshot displays the Quartus II interface during the compilation phase. On the left, the Project Navigator shows the 'niosII_lab' project with 3117 logic cells. The Status window indicates that all compilation steps, including Full Compilation, Analysis & Synthesis, Fitter, Assembler, and Timing Analyzer, are 100% complete. The main window shows the 'Compilation Report - Flow Summary' with a 'Flow Summary' tab selected. A 'Quartus II' dialog box is overlaid on the report, displaying the message 'Full Compilation was successful (117 warnings)' and a '确定' (OK) button.

Module	Progress %	Time
Full Compilation	100 %	00:00
Analysis & Synthesis	100 %	00:00
Fitter	100 %	00:00
Assembler	100 %	00:00
Timing Analyzer	100 %	00:00

Flow Summary

Flow Status: Successful - Wed Aug 23 10:36:54 2006
Quartus II Version: 6.0 Build 202 06/20/2006 SP 1 SJ Full Version
Revision Name: niosII_lab
Top-level Entity Name: niosII_lab

Quartus II
Full Compilation was successful (117 warnings)

确定

Total memory bits: 55,296 / 92,160 (60 %)
Total PLLs: 1 / 2 (50 %)

图10-27 全程编译完成

10.2 NiosII软件设计与运行流程

1、向FPGA下载配置文件

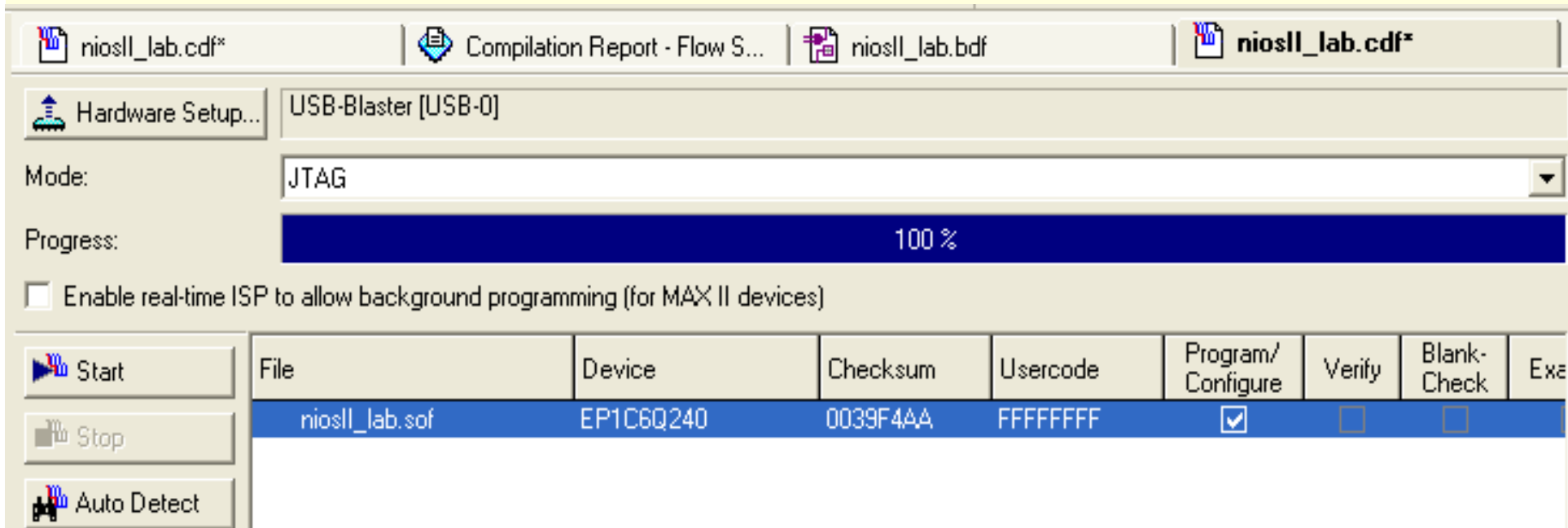


图10-28 下载niosII_lab.sof配置文件

10.2 NiosII软件设计与运行流程

2、进入集成开发环境IDE

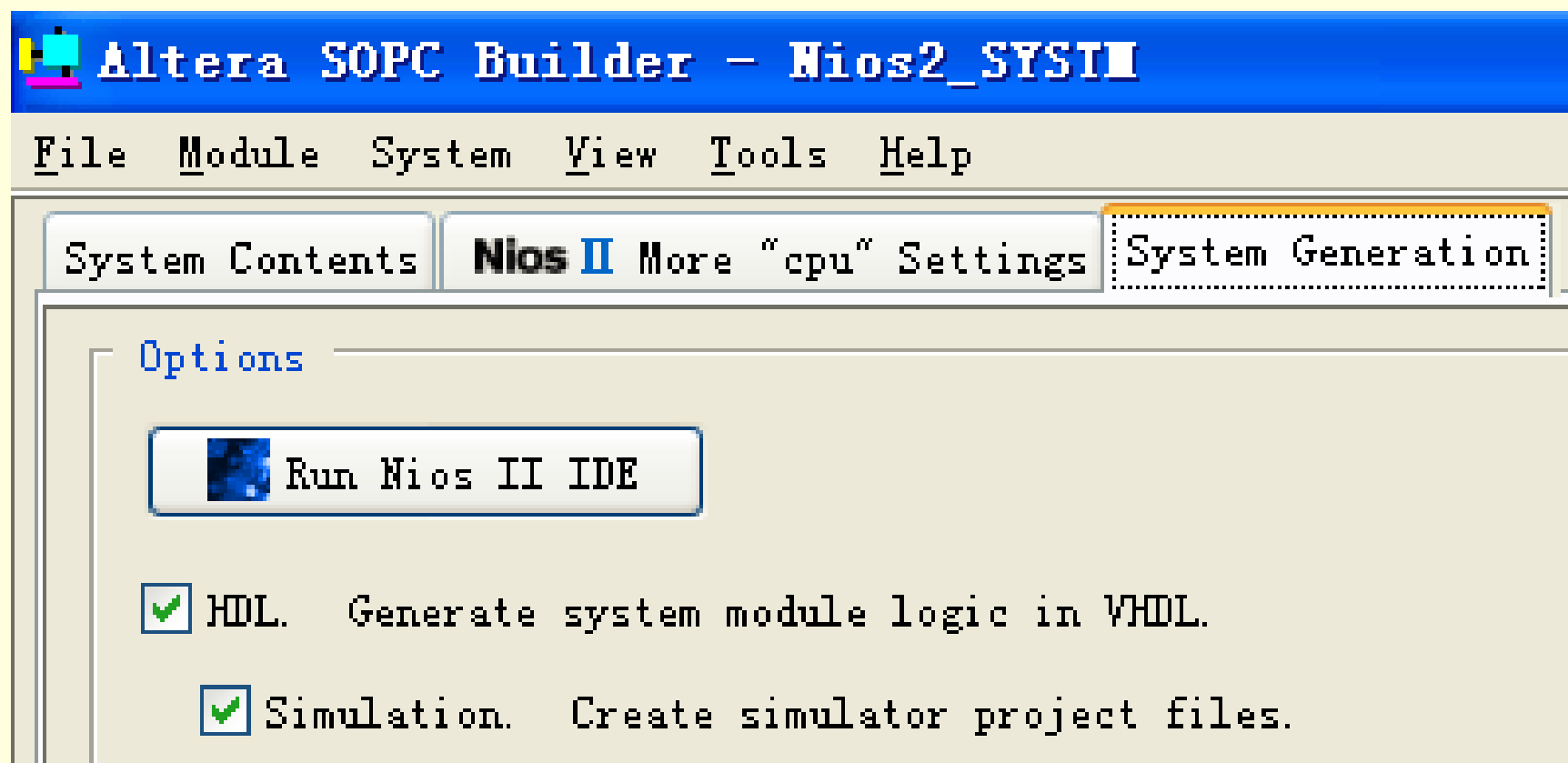


图10-29 点击Run NiosII IDE按钮，进入集成开发环境

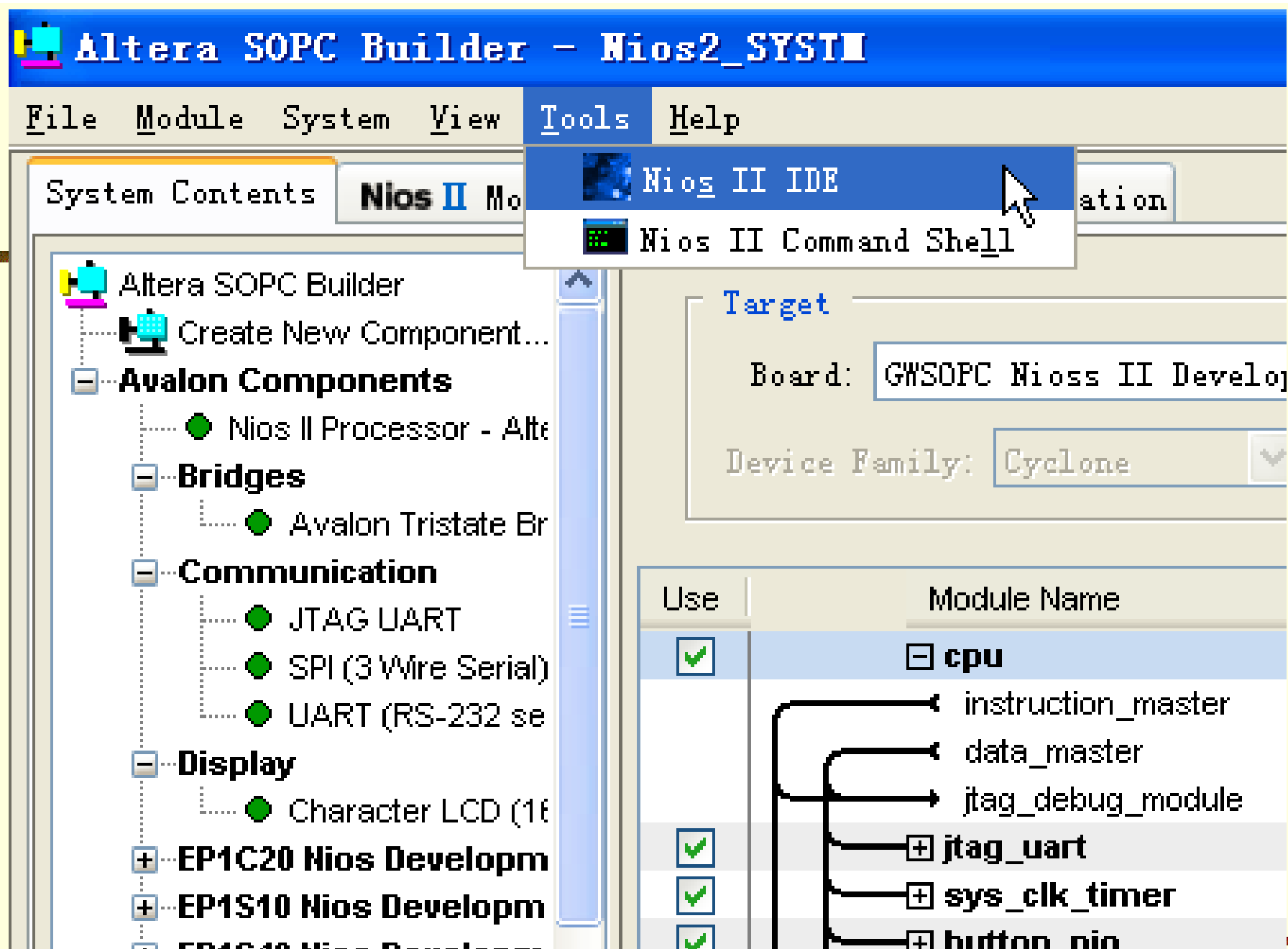


图10-30 选择NiosII IDE选项，进入集成开发环境

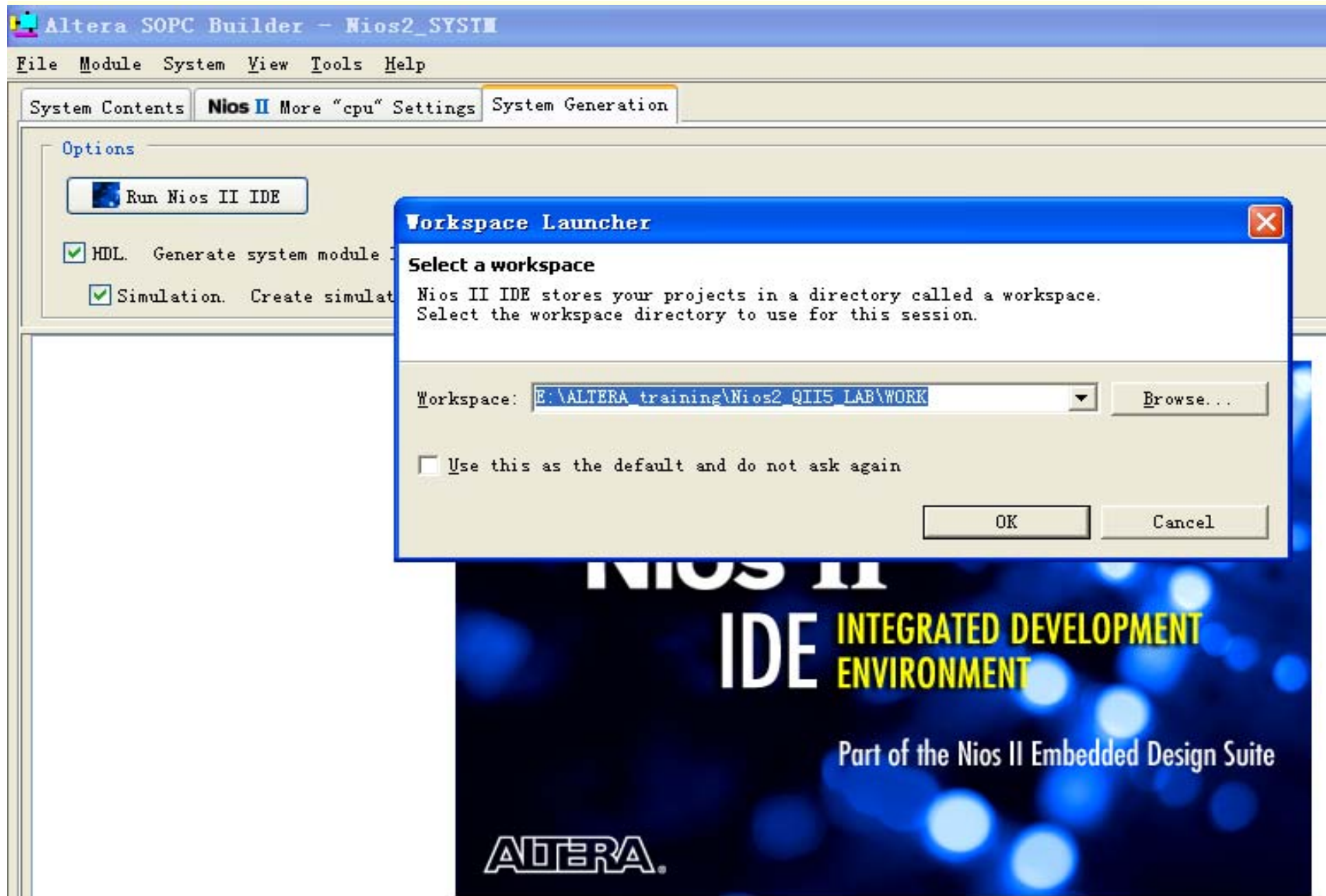


图10-31 选择软件工程库

10.2 NiosII软件设计与运行流程

2、进入集成开发环境IDE

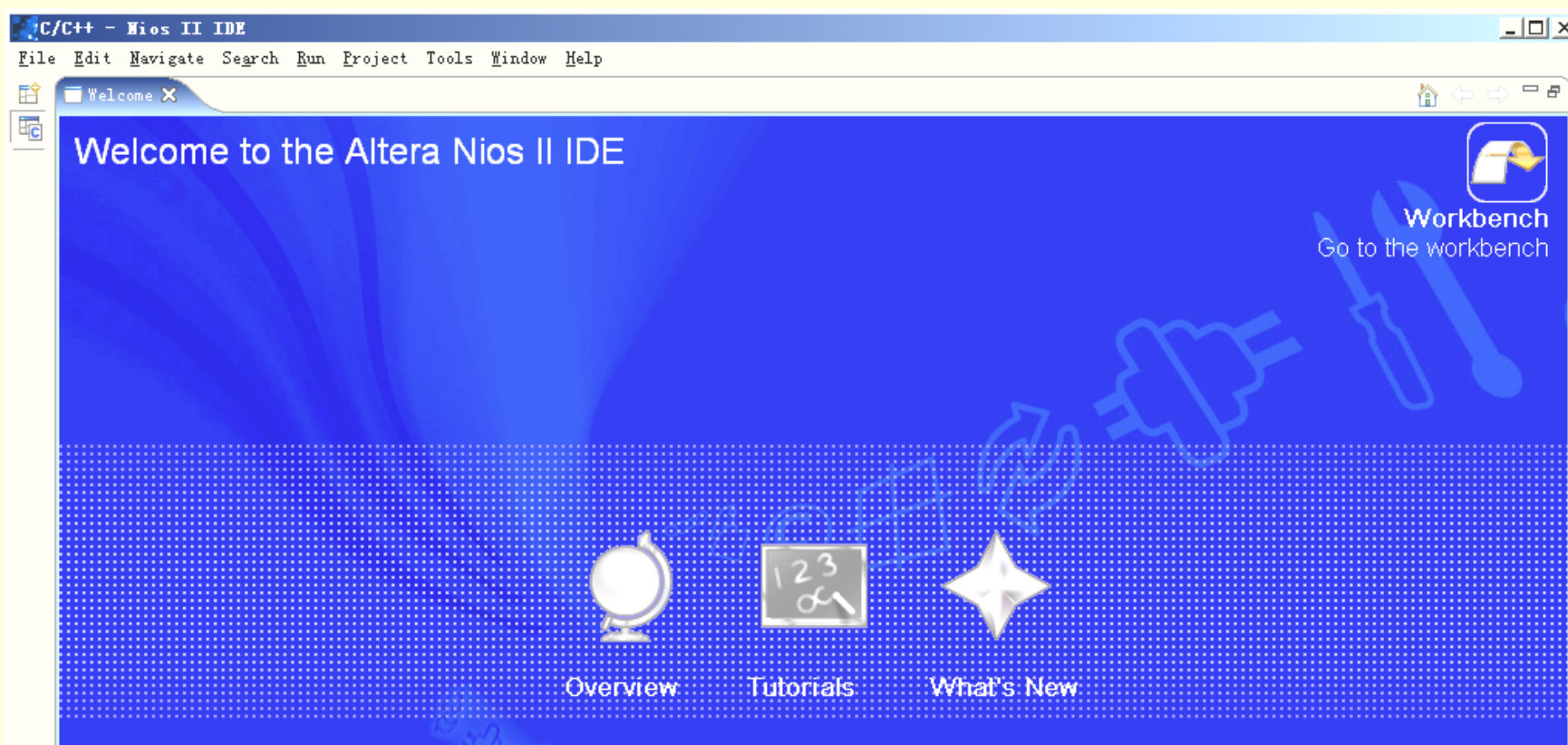


图10-32 选择进入IDE软件设计/调试平台

10.2 NiosII软件设计与运行流程

3、建立C软件开发工程

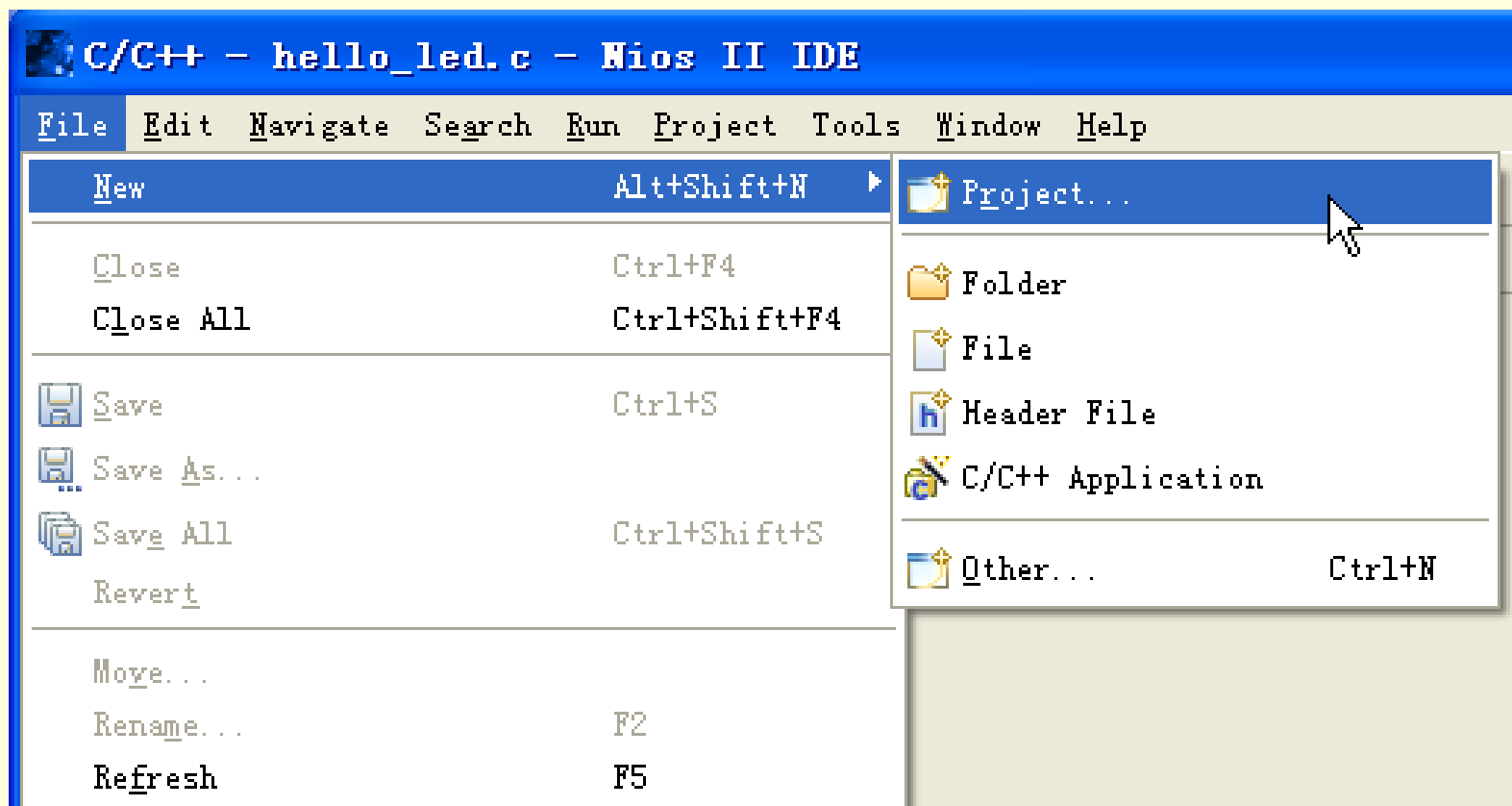


图10-33 建立一个软件实例工程

10.2 NiosII软件设计与运行流程

3、建立C软件开发工程

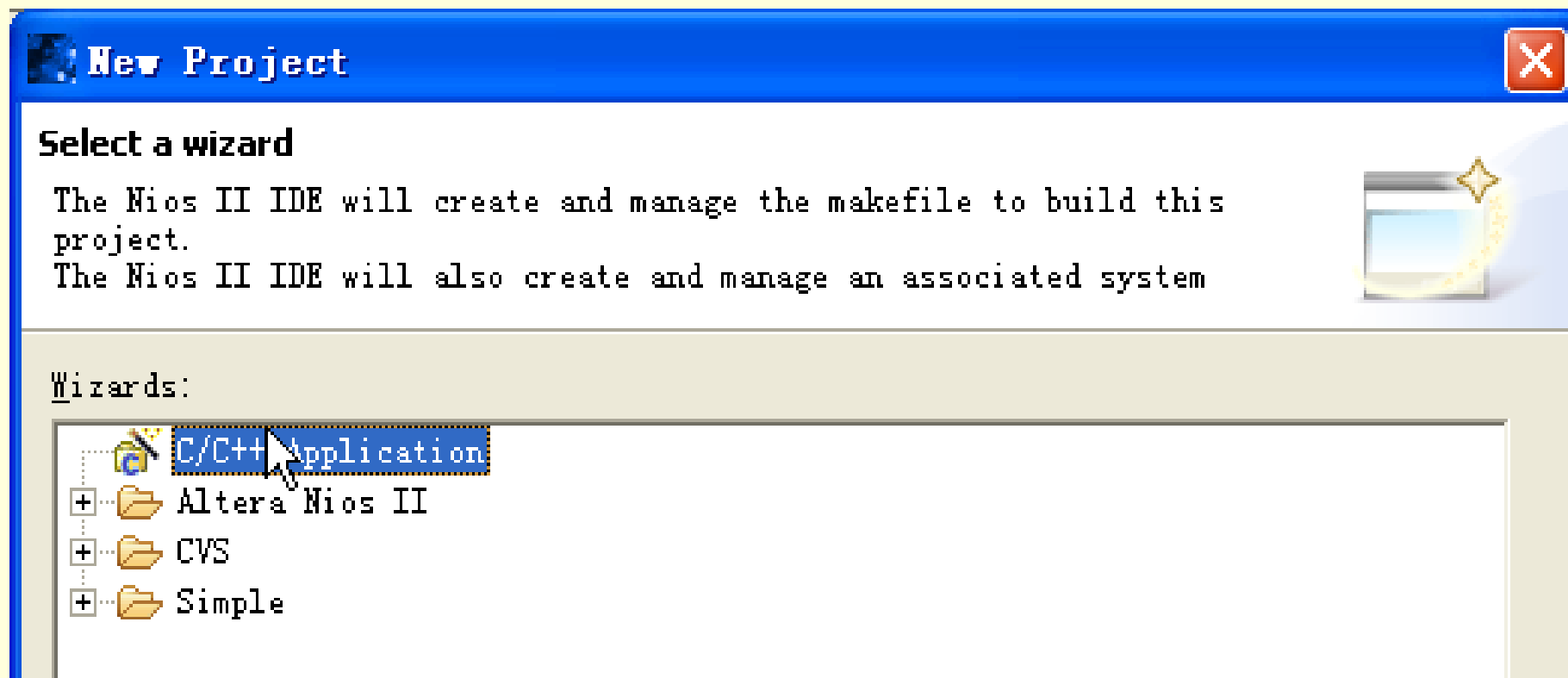


图10-34 选择C/C++应用

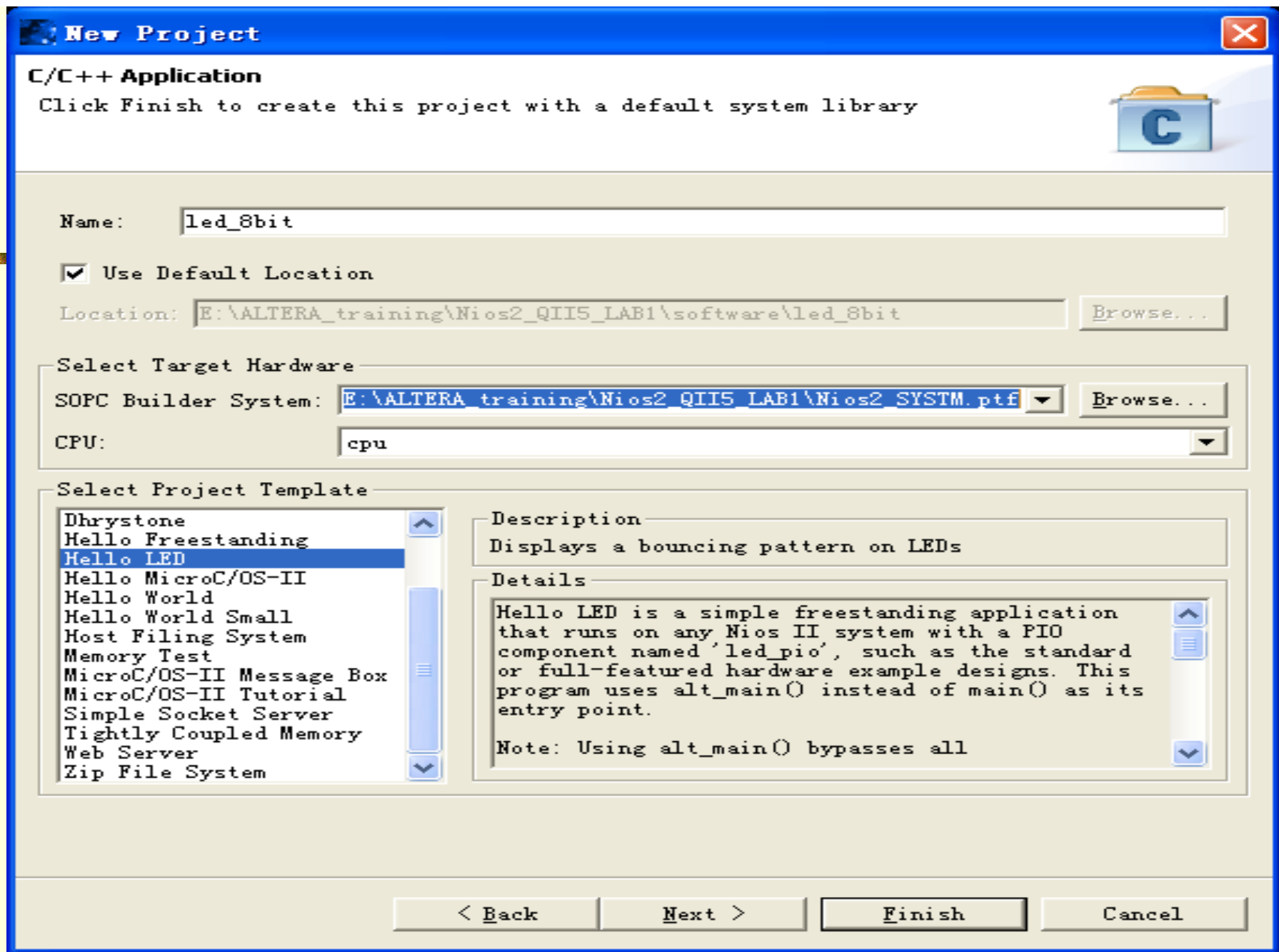


图10-35 在示例库中选择一个C程序实例

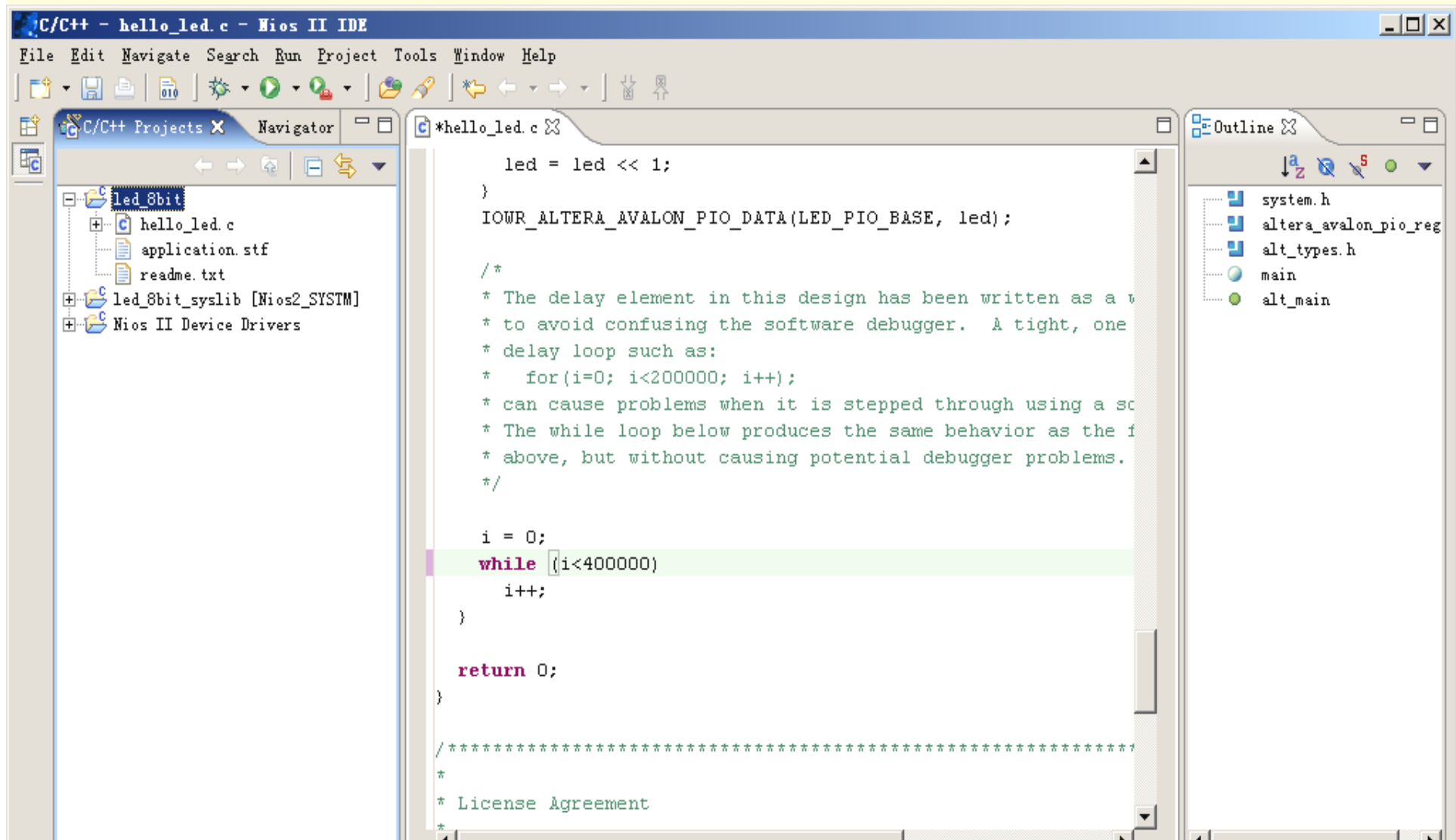


图10-36 进入NiosII IDE窗口

10.2 NiosII软件设计与运行流程

4、编译运行C程序

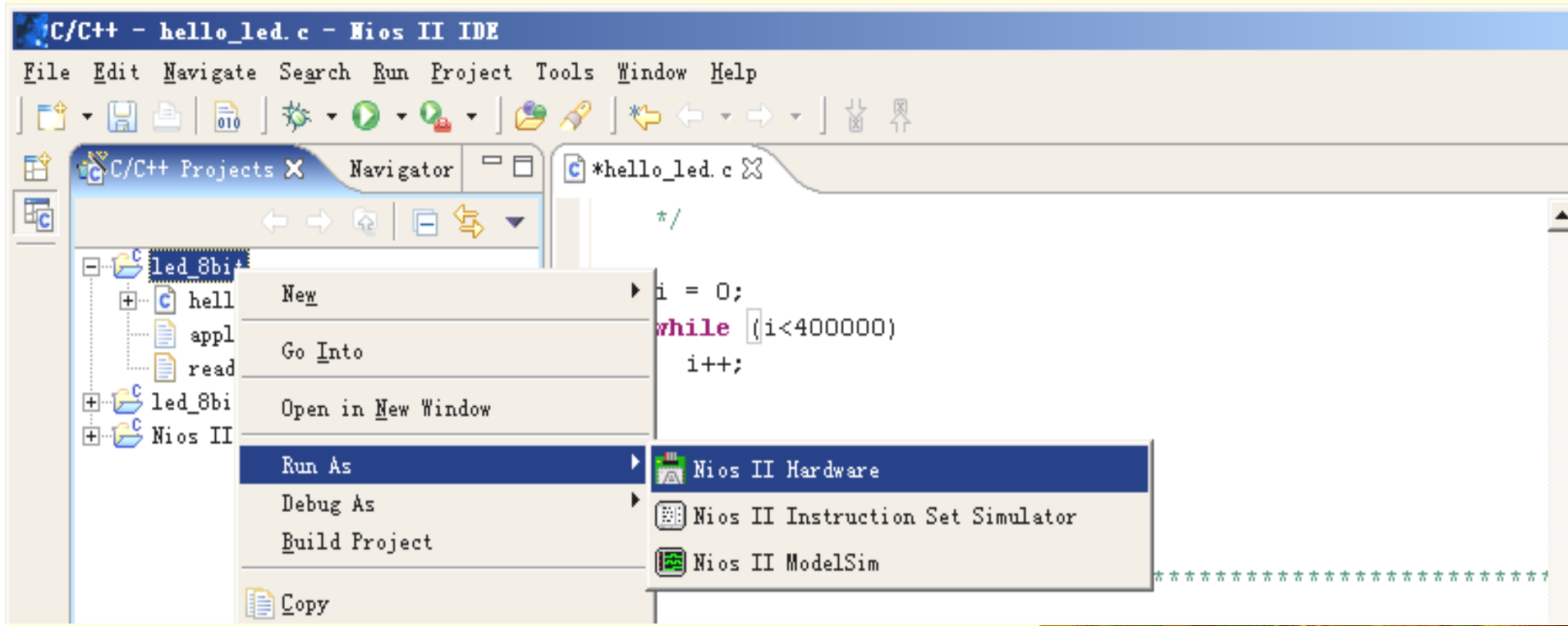


图10-37 编译、下载并在NiosII CPU中全速运行该示例

10.2 NiosII软件设计与运行流程

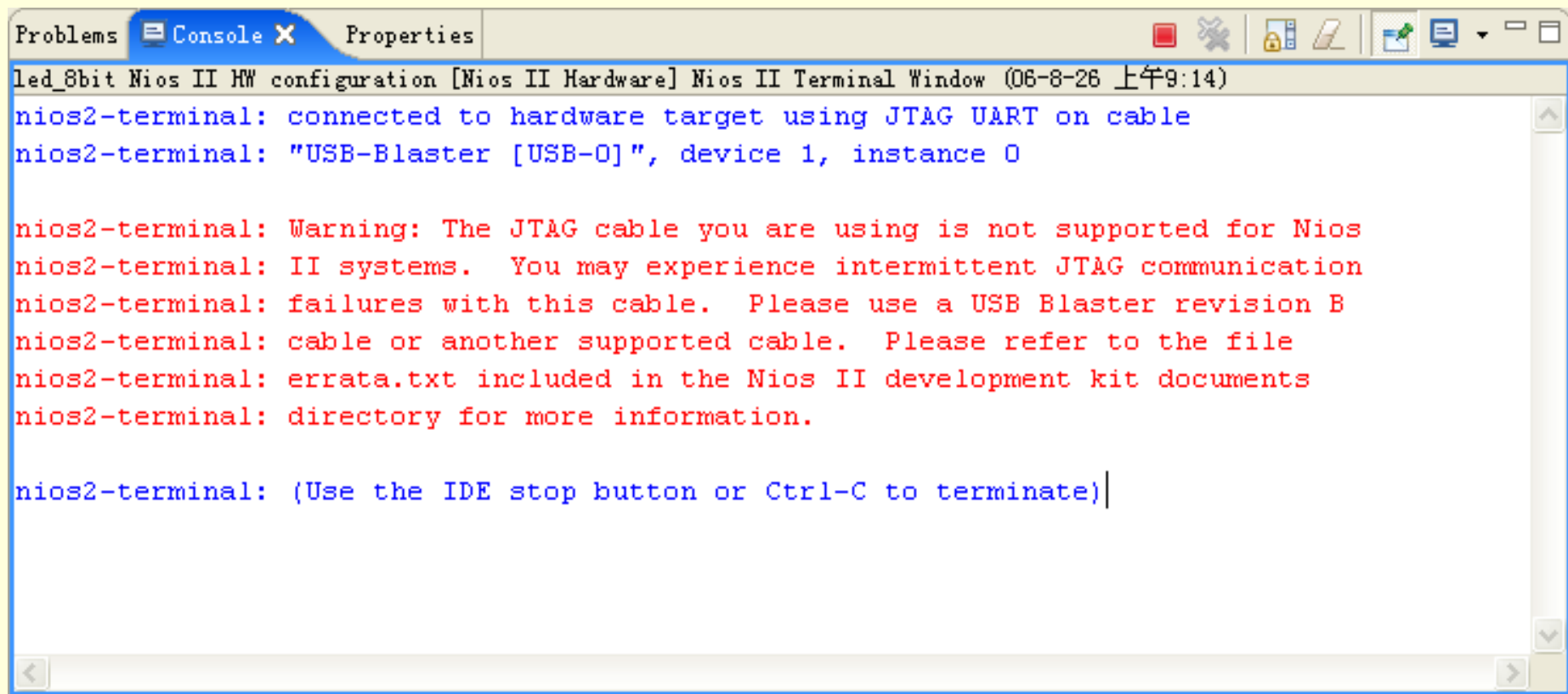
5、观察运行结果



图10-38 存盘已修改的C源程序

10.2 NiosII软件设计与运行流程

6、单步/跟踪调试运行



```
led_8bit Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (06-8-26 上午9:14)
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0

nios2-terminal: Warning: The JTAG cable you are using is not supported for Nios
nios2-terminal: II systems. You may experience intermittent JTAG communication
nios2-terminal: failures with this cable. Please use a USB Blaster revision B
nios2-terminal: cable or another supported cable. Please refer to the file
nios2-terminal: errata.txt included in the Nios II development kit documents
nios2-terminal: directory for more information.

nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)|
```

图10-39 C程序下载成功，启动运行

10.2 NiosII软件设计与运行流程

6、单步/跟踪调试运行

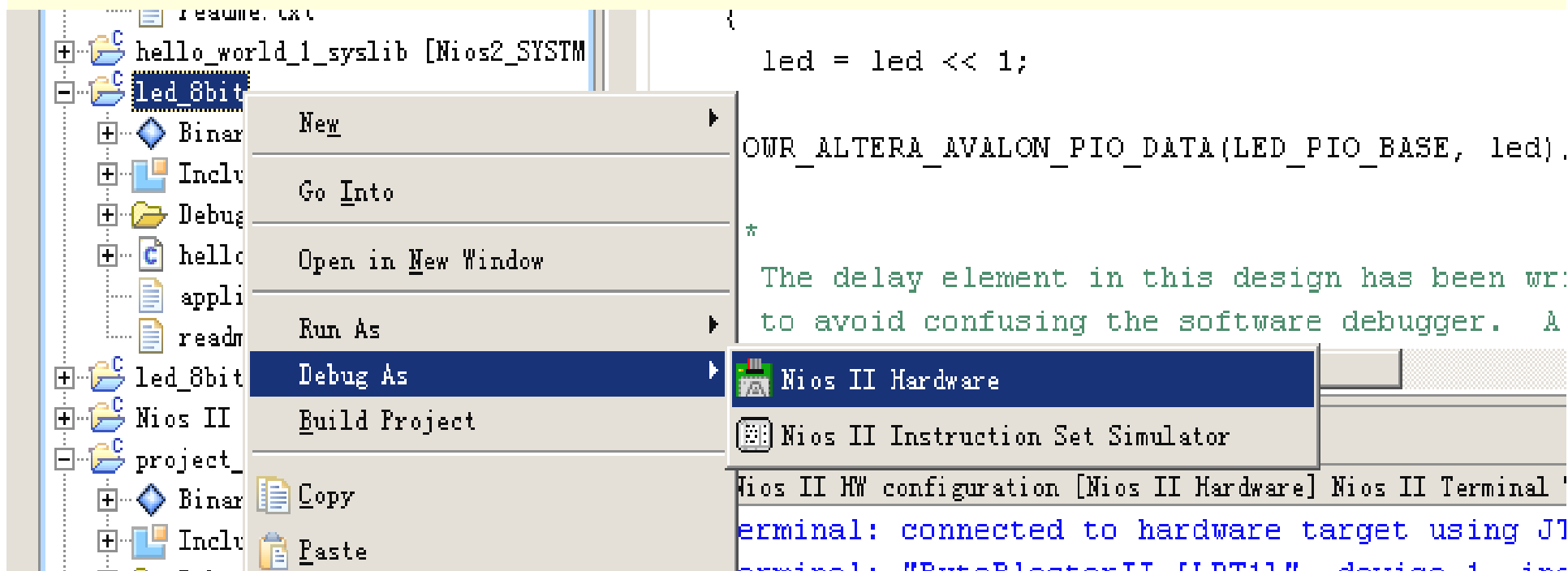


图10-40 选择单步/跟踪调试模式运C程序

10.2 NiosII软件设计与运行流程

6、单步/跟踪调试运行

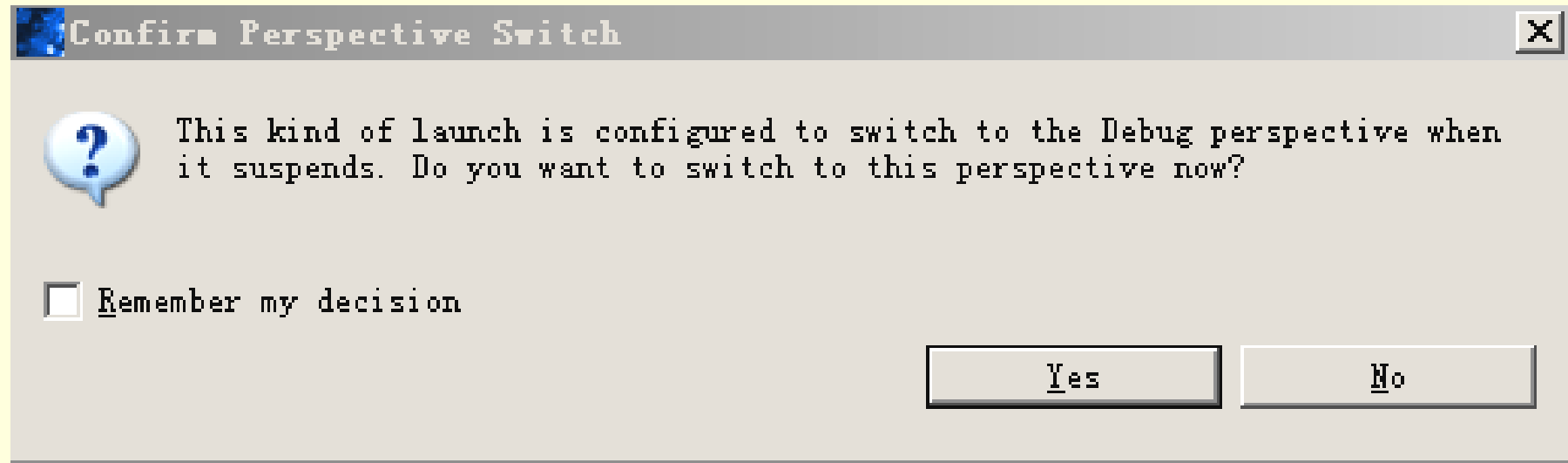


图10-41 选择单步/跟踪调试模式运C程序

10.2 NiosII软件设计与运行流程

6、单步/跟踪调试运行

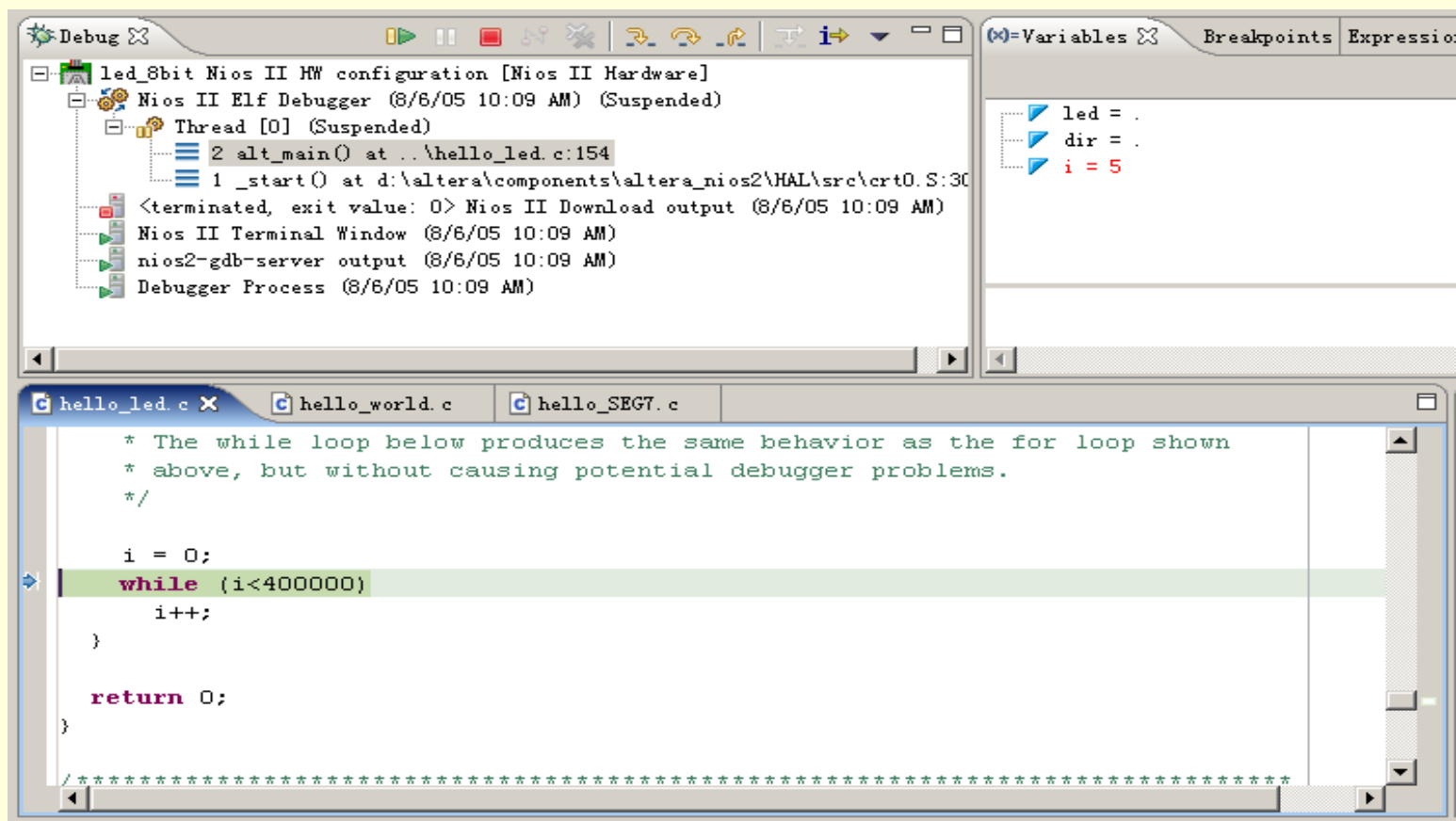


图10-42 单步/跟踪调试

10.2 NiosII软件设计与运行流程

6、单步/跟踪调试运行

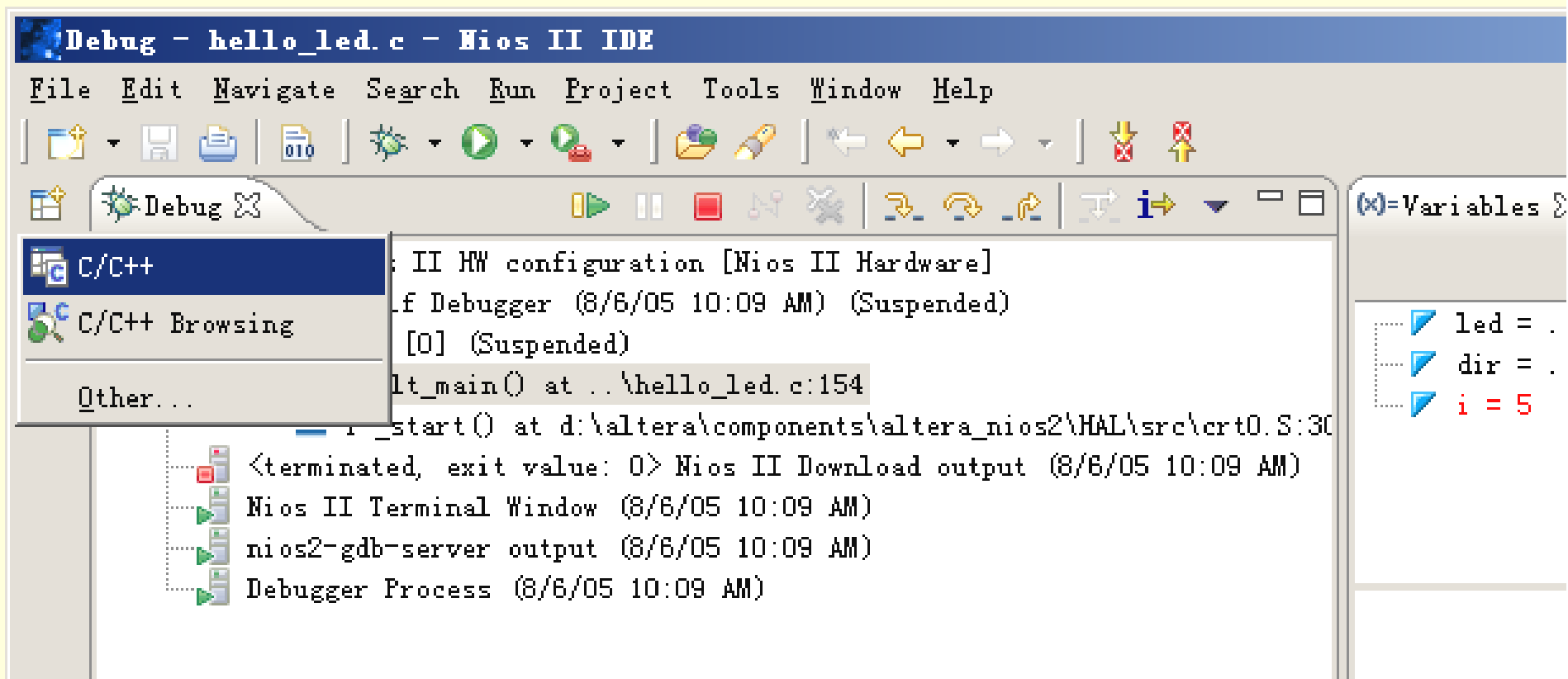


图10-43 返回IDE主控窗

10.2 NiosII软件设计与运行流程

6、单步/跟踪调试运行

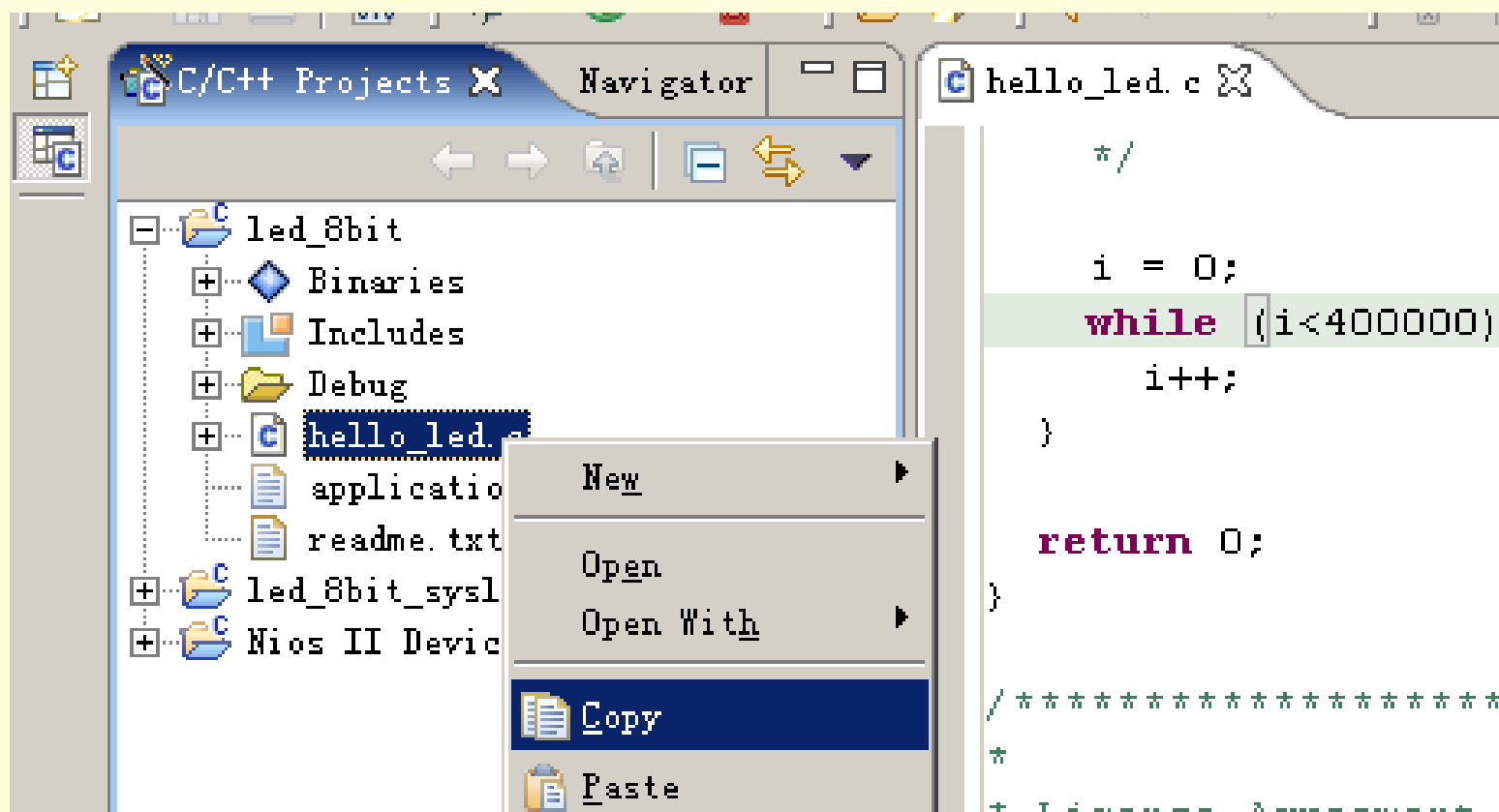


图10-44 将已调试好的C程序COPY到其他文件夹存盘

10.2 NiosII软件设计与运行流程

7、运行另一个示例程序

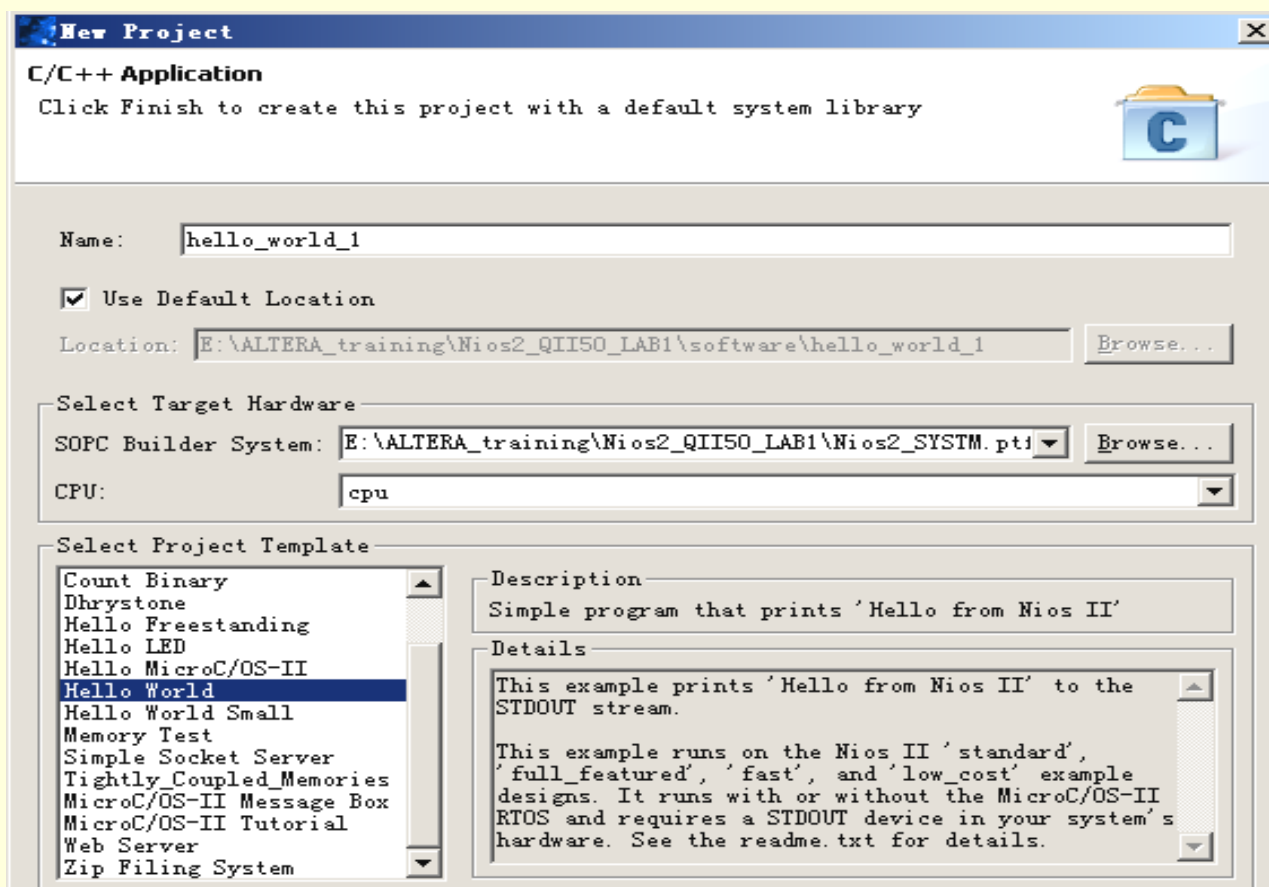


图10-45 为测试运行另一个示例程序建立一个新工程

10.2 NiosII软件设计与运行流程

8、运行用户程序

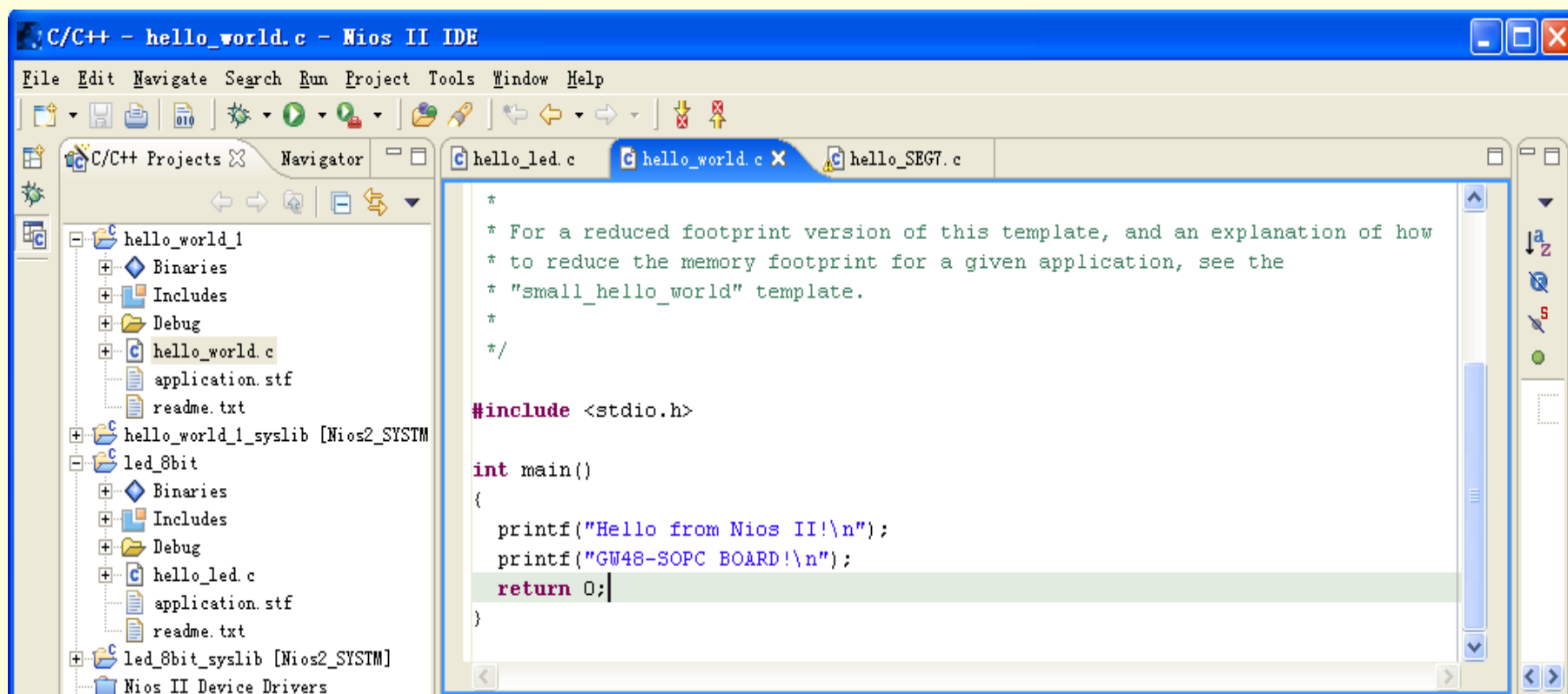
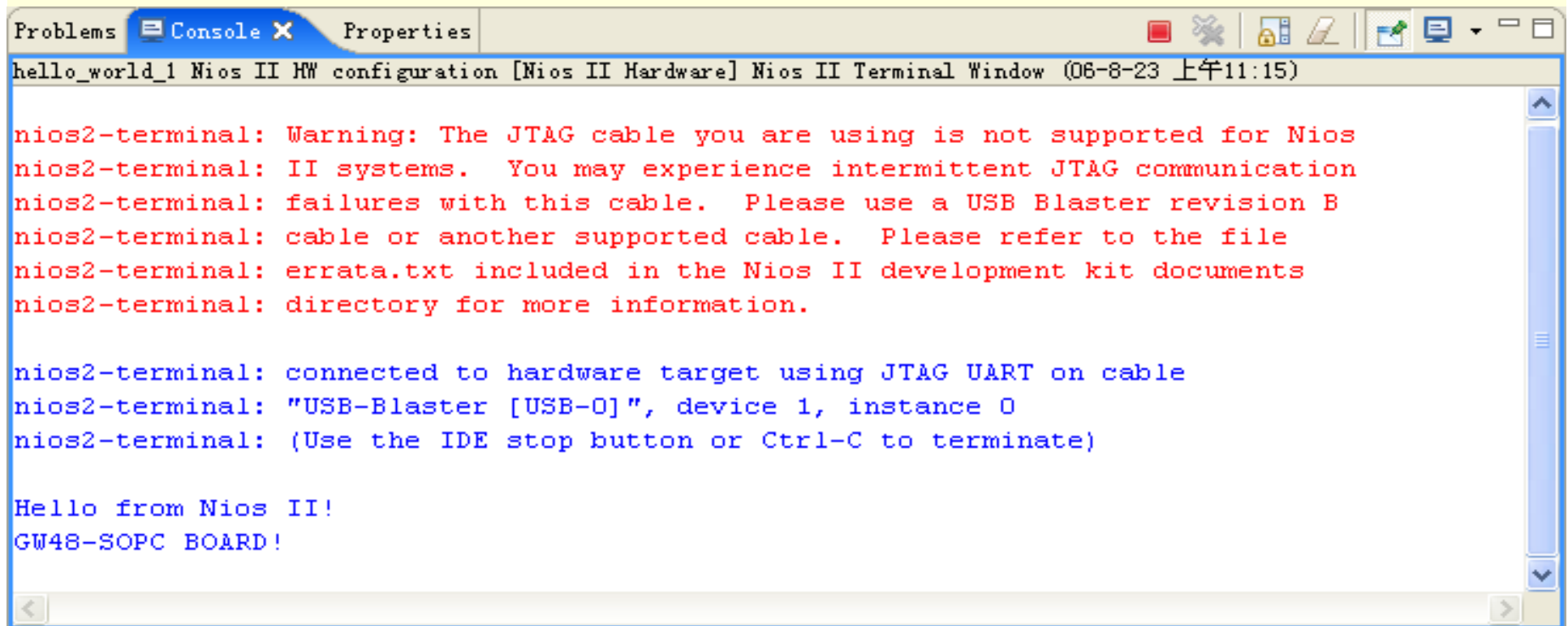


图10-46 修改hello_world.c源程序

10.2 NiosII软件设计与运行流程

8、运行用户程序



The screenshot shows a terminal window titled "hello_world_1 Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (06-8-23 上午11:15)". The window contains the following text:

```
nios2-terminal: Warning: The JTAG cable you are using is not supported for Nios
nios2-terminal: II systems. You may experience intermittent JTAG communication
nios2-terminal: failures with this cable. Please use a USB Blaster revision B
nios2-terminal: cable or another supported cable. Please refer to the file
nios2-terminal: errata.txt included in the Nios II development kit documents
nios2-terminal: directory for more information.

nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Hello from Nios II!
GW48-SOPC BOARD!
```

图10-47 hello_world.c程序下载运行成功，并通过JTAG_UART口输出执行结果

10.2 NiosII软件设计与运行流程

8、运行用户程序

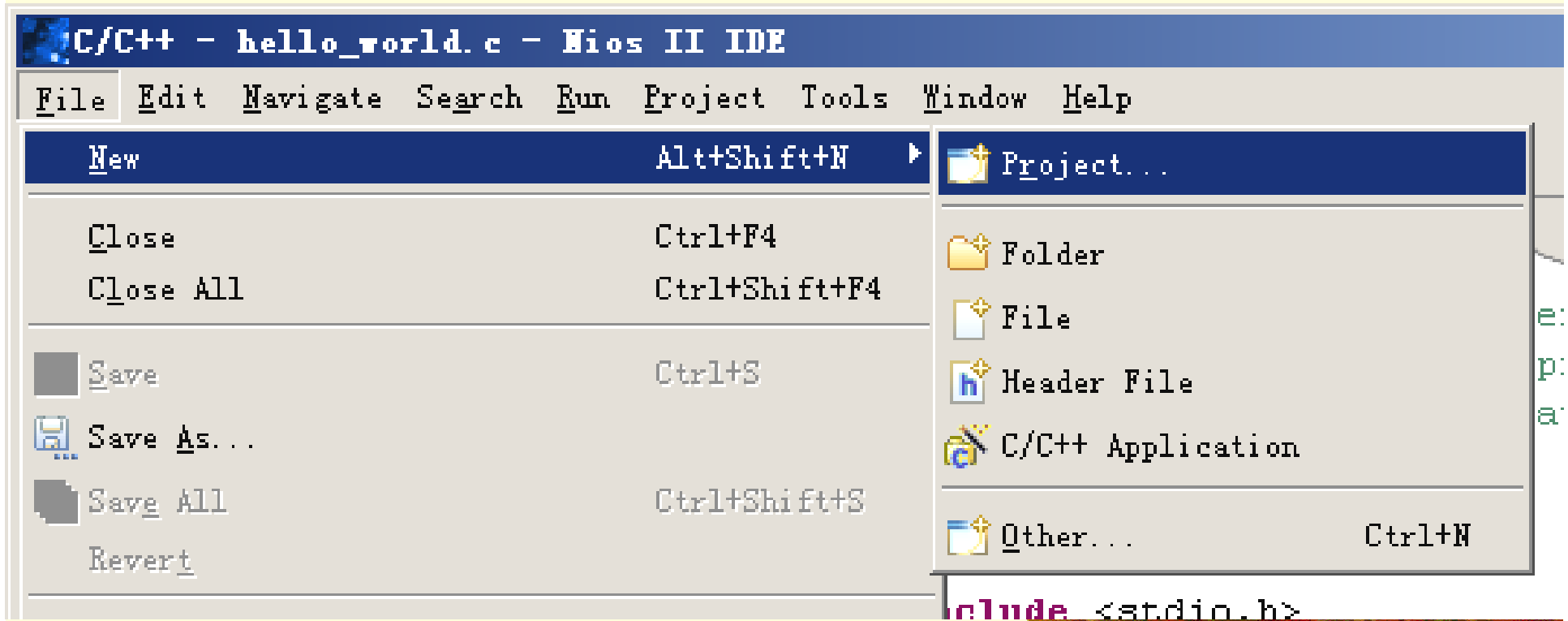


图10-48 为调试一个用户程序建立一个空白工程

10.2 NiosII软件设计与运行流程

8、运行用户程序

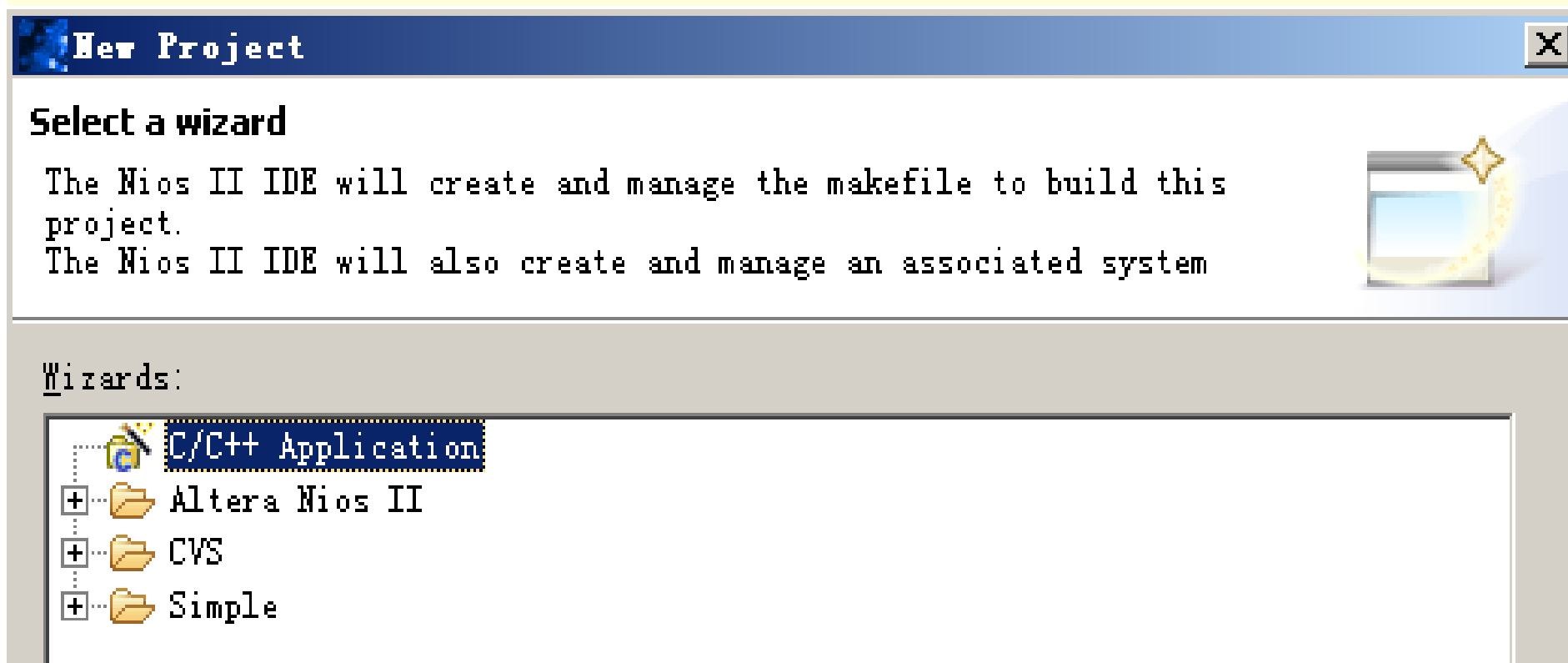


图10-49 同样选择C/C++应用

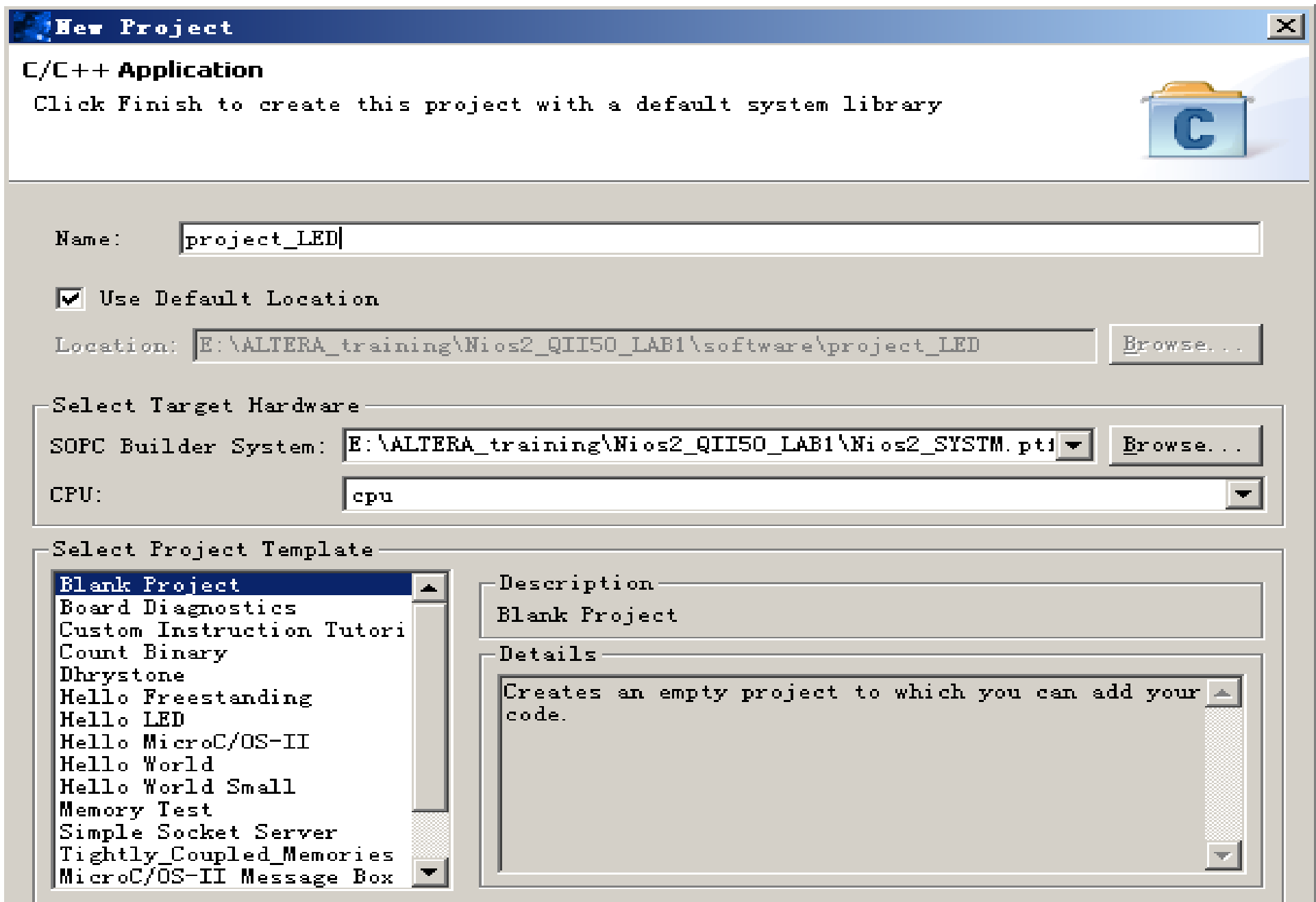


图10-50 选择空白工程，并取名为：project_LED

10.2 NiosII软件设计与运行流程

8、运行用户程序

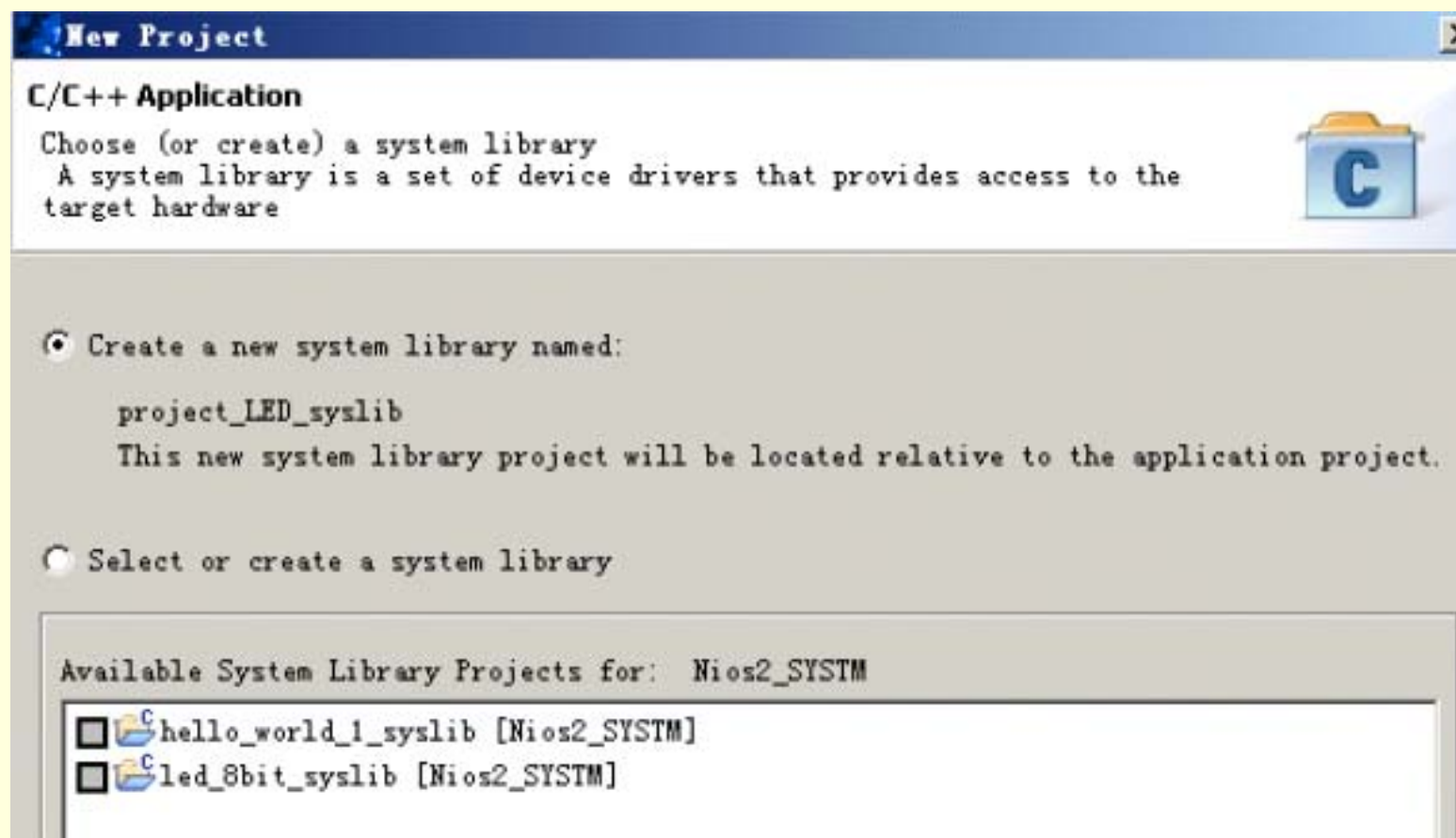


图10-51 选择默认设置

10.2 NiosII软件设计与运行流程

8、运行用户程序

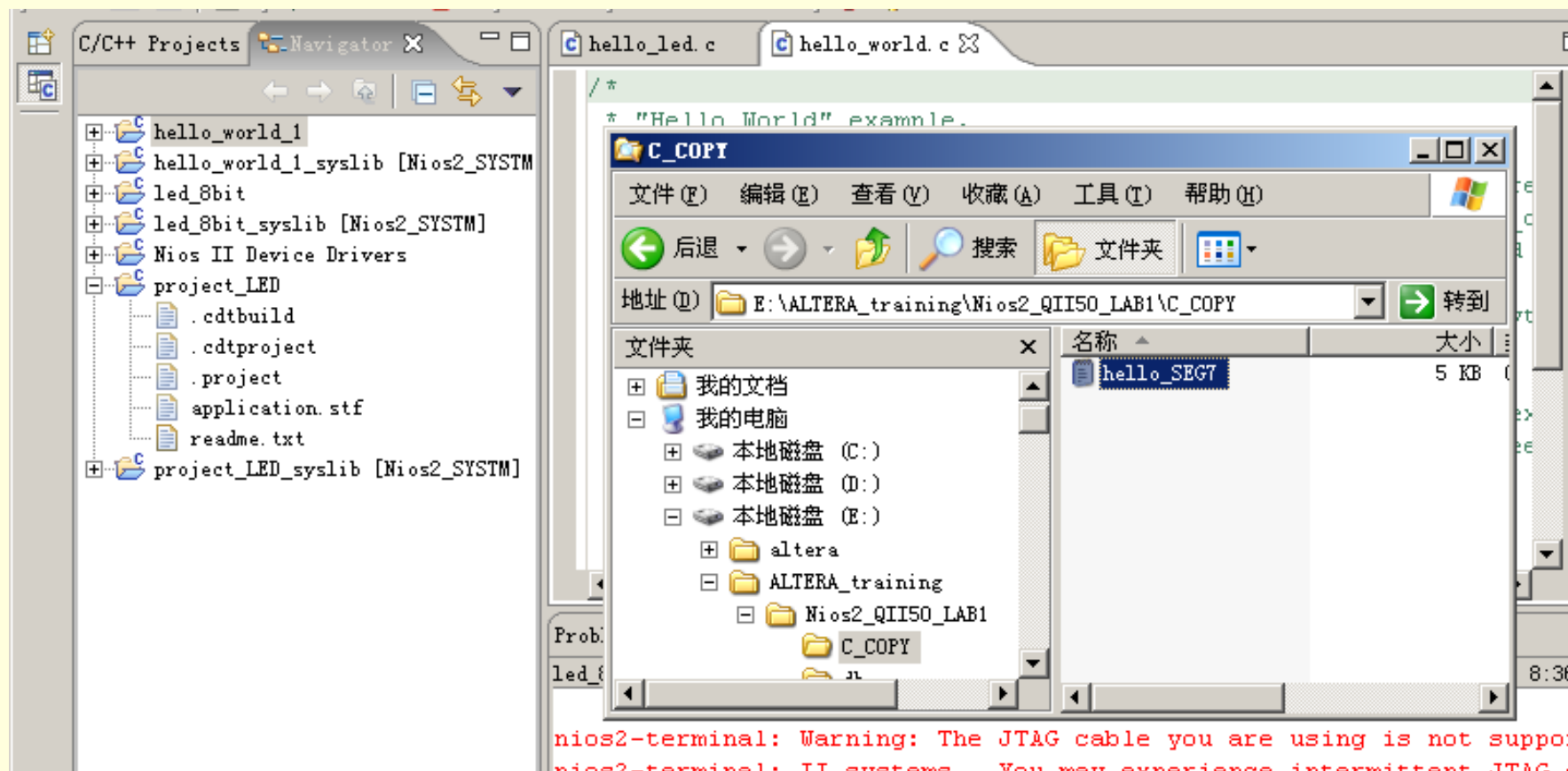


图10-52 将hello_SEG7.c拖入Navigator栏中的用户C程序工程

10.2 NiosII软件设计与运行流程

8、运行用户程序

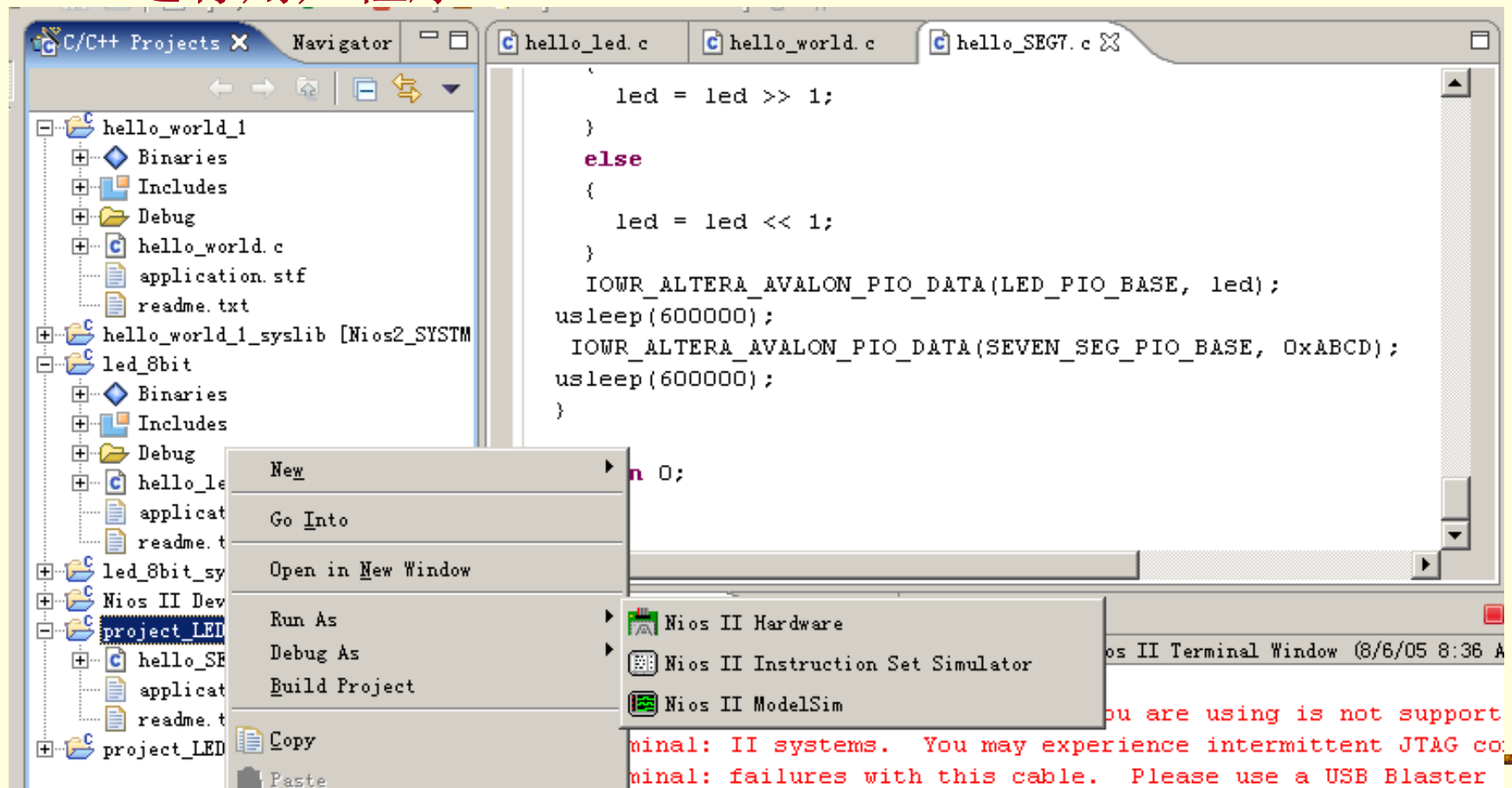


图10-53 返回C/C++ Projects窗，并全速运行该用户程序

10.3加入用户自定义组件设计

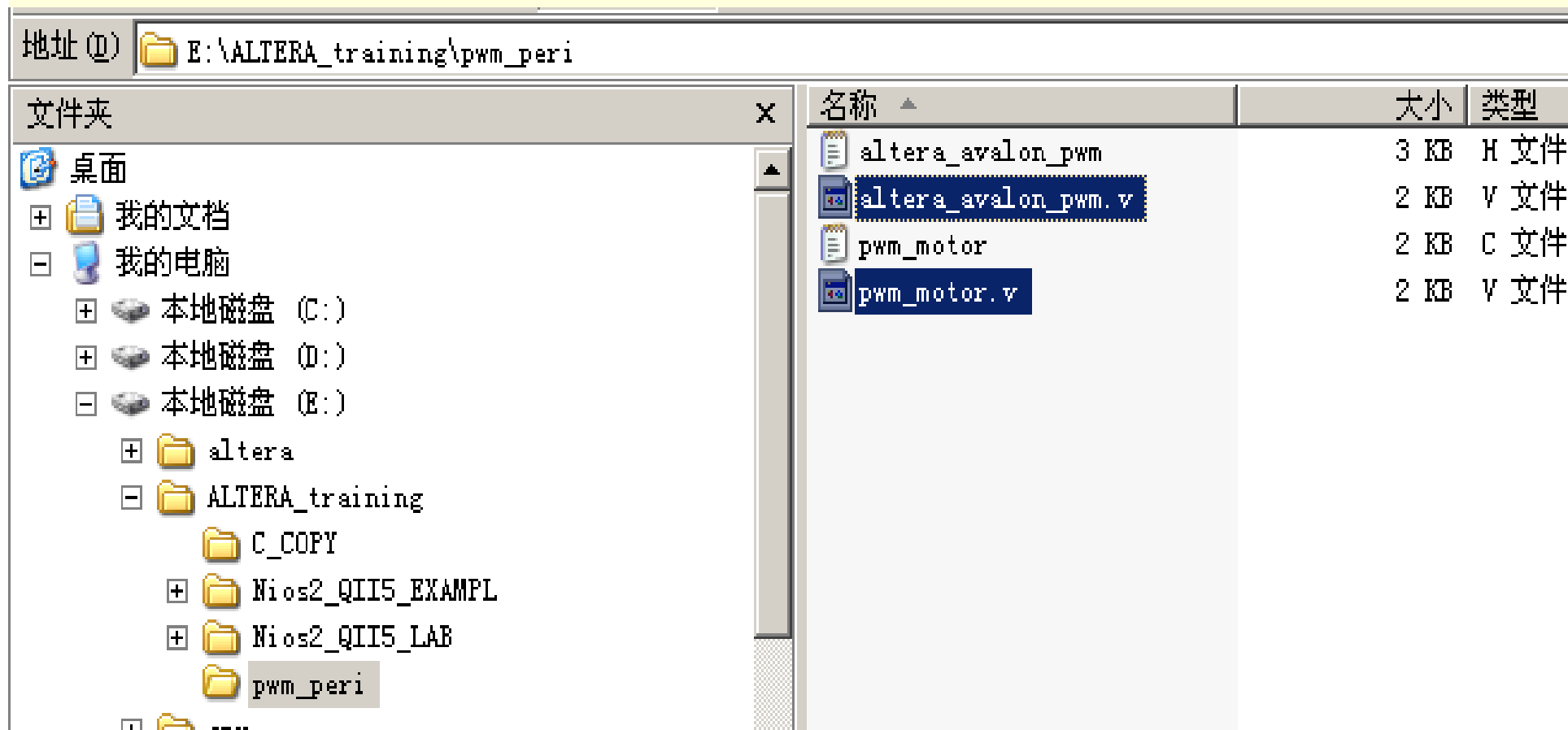


图10-54 PWM用户自定义逻辑模块的Verilog文件和C文件存放路径

1、入用户逻辑模块

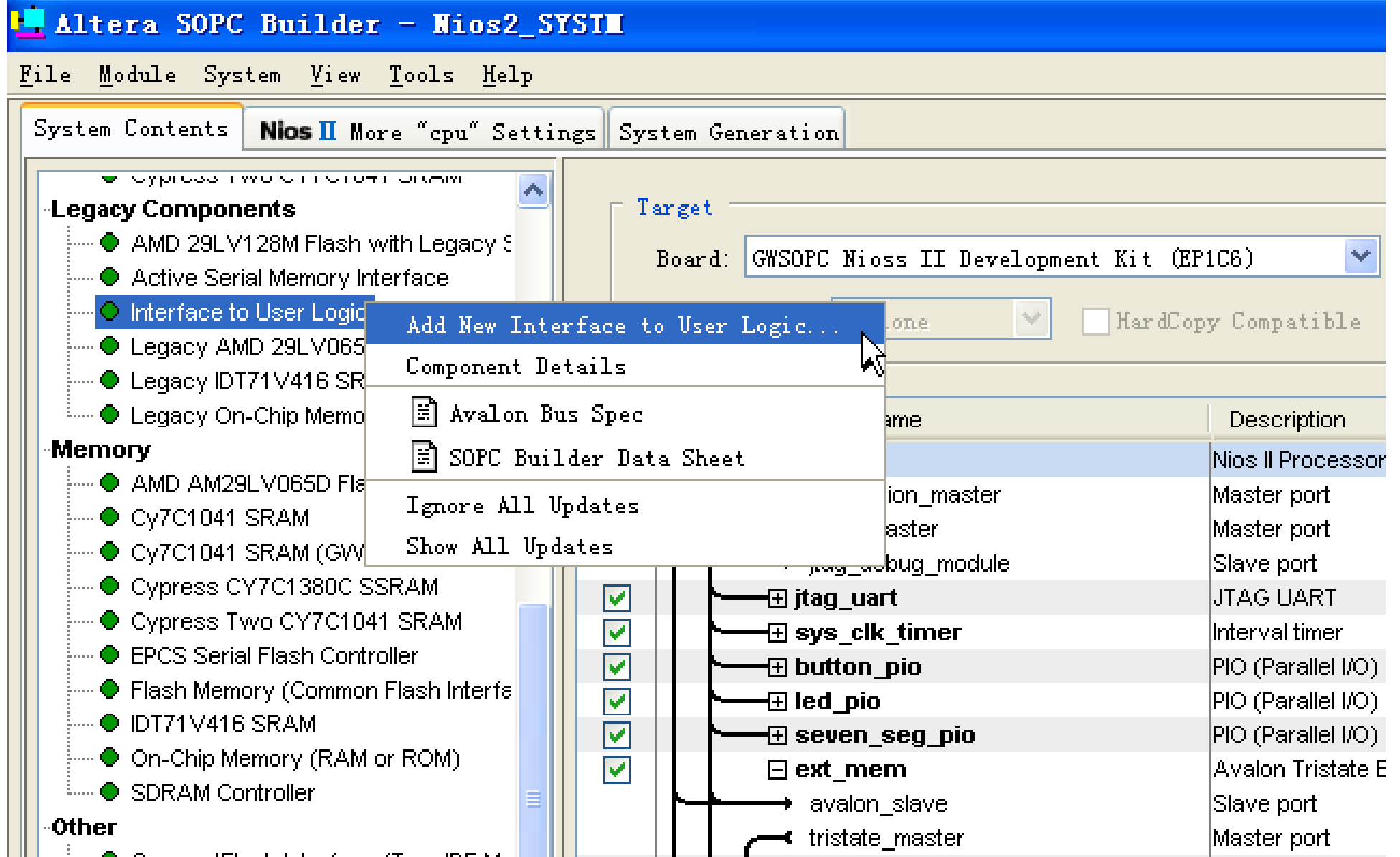


图10-55 将用户自定义逻辑模块加入进NiosII中

1、入用户逻辑模块

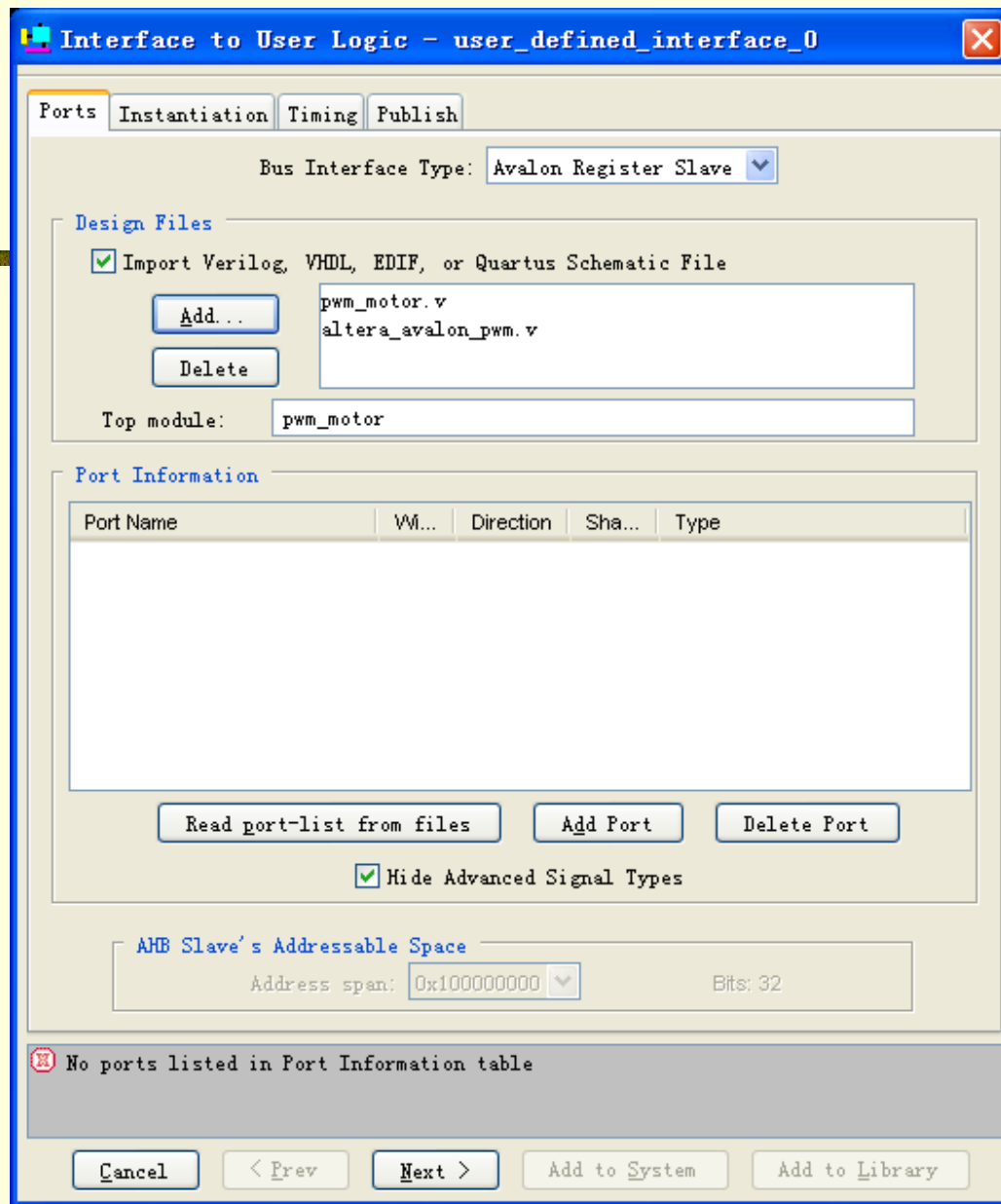


图10-56 加入用户自定义逻辑模块文件

1、入用户逻辑模块

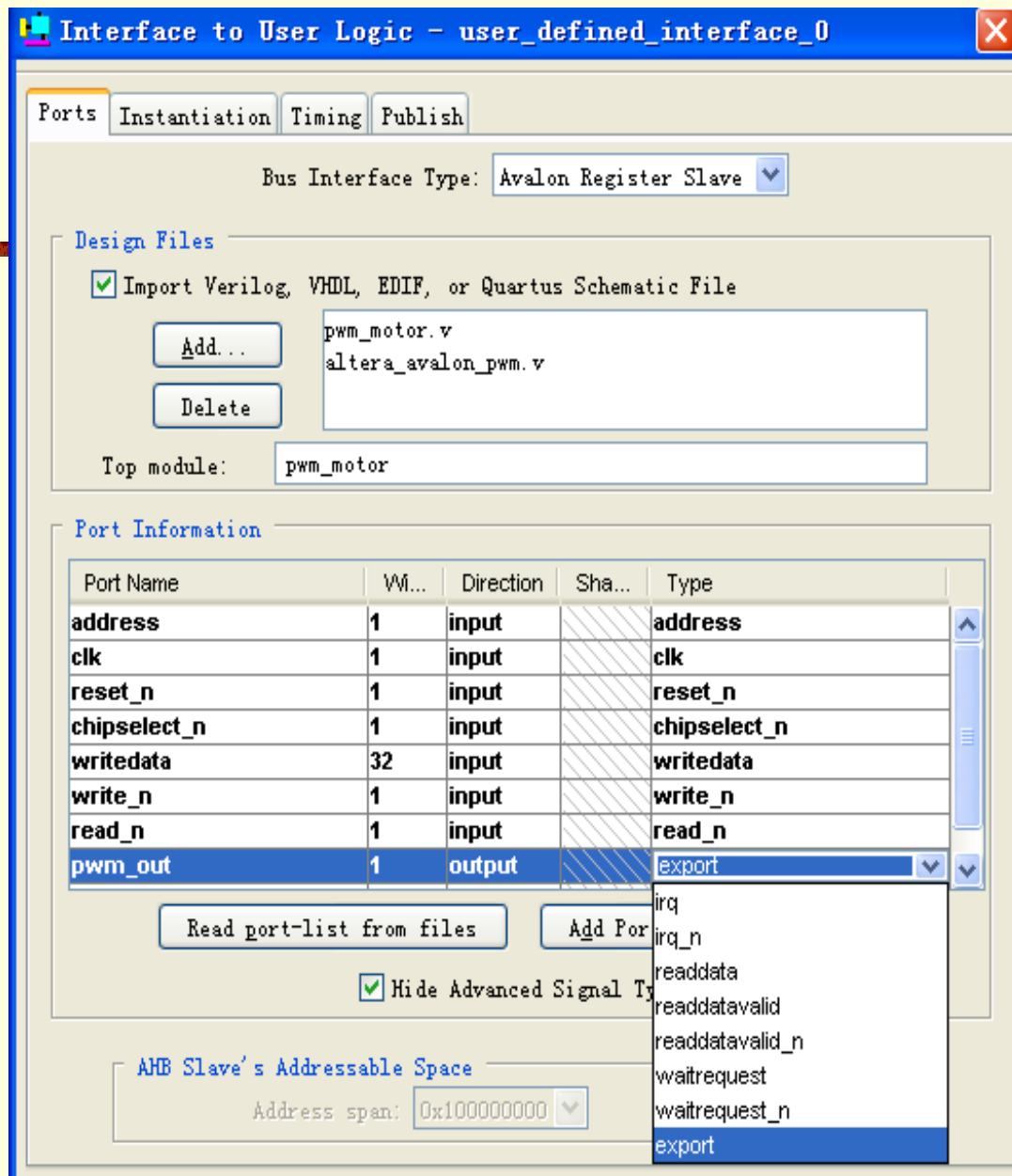


图10-57 读入模块的端口表，并补充端口名

10.3 加入用户自定义组件设计

1、入用户逻辑模块

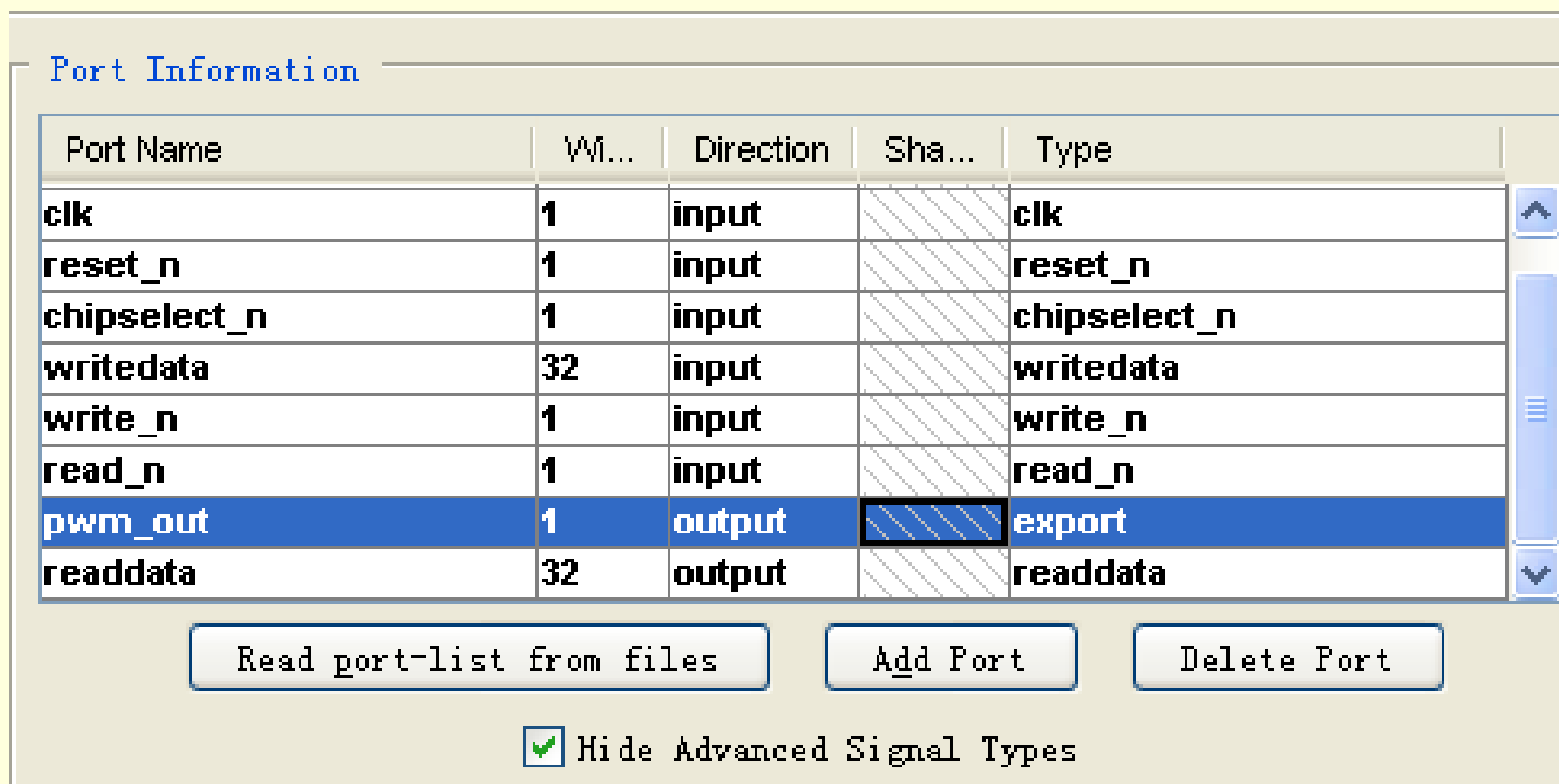


图10-58 最后正确的端口名表

10.3 加入用户自定义组件设计

1、加入用户逻辑模块


Use	Module Name	Description	Input Clock	Base	End	IRQ
<input checked="" type="checkbox"/>	<input type="checkbox"/> cpu	Nios II Processor - Altera Corporation	clk			
	instruction_master	Master port				
	data_master	Master port				
	jtag_debug_module	Slave port				
<input checked="" type="checkbox"/>	<input type="checkbox"/> jtag_uart	JTAG UART	clk	0x00901050	0x00901057	0
<input checked="" type="checkbox"/>	<input type="checkbox"/> sys_clk_timer	Interval timer	clk	0x00901000	0x0090101F	1
<input checked="" type="checkbox"/>	<input type="checkbox"/> button_pio	PIO (Parallel I/O)	clk	0x00901020	0x0090102F	2
<input checked="" type="checkbox"/>	<input type="checkbox"/> led_pio	PIO (Parallel I/O)	clk	0x00901030	0x0090103F	
<input checked="" type="checkbox"/>	<input type="checkbox"/> seven_seg_pio	PIO (Parallel I/O)	clk	0x00901040	0x0090104F	
<input checked="" type="checkbox"/>	<input type="checkbox"/> ext_mem	Avalon Tristate Bridge	clk			
	avalon_slave	Slave port				
	tristate_master	Master port				
<input checked="" type="checkbox"/>	<input type="checkbox"/> ext_flash	Flash Memory (Common Flash Interface)		0x00000000	0x007FFFFFFF	
<input checked="" type="checkbox"/>	<input type="checkbox"/> sysid	System ID Peripheral	clk	0x00901058	0x0090105F	
<input checked="" type="checkbox"/>	<input type="checkbox"/> epcs_controller	EPCS Serial Flash Controller	clk	0x00900800	0x00900FFF	3
<input checked="" type="checkbox"/>	<input type="checkbox"/> ext_sram	Cy7C1041 SRAM (GWSOPC)		0x00800000	0x008FFFFFFF	
<input checked="" type="checkbox"/>	<input type="checkbox"/> my_pwm	Interface to User Logic	clk	0x00901060	0x00901067	

图10-59 加入了用户定制模块的NiosII组件列表

10.3 加入用户自定义组件设计

2、生成和编译

```
# 2006.08.26 09:42:53 (*) Completed generation for system: Nios2_SYSTM.  
# 2006.08.26 09:42:53 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:  
SOPC Builder database : E:/ALTERA_training/Nios2_QII5_LAB1/Nios2_SYSTM.ptf  
System HDL Model : E:/ALTERA_training/Nios2_QII5_LAB1/Nios2_SYSTM.vhd  
System Generation Script : E:/ALTERA_training/Nios2_QII5_LAB1/Nios2_SYSTM_generation_script  
HDL Simulation Directory : E:/ALTERA_training/Nios2_QII5_LAB1/Nios2_SYSTM_sim  
  
# 2006.08.26 09:42:53 (*) SUCCESS: SYSTEM GENERATION COMPLETED.  
  
Press 'Exit' to exit.
```

 **cpu** was generated as plain-text HDL.

Exit

< Prev

Next >

Re-Generate

图10-60 NiosII组件生成运行完成

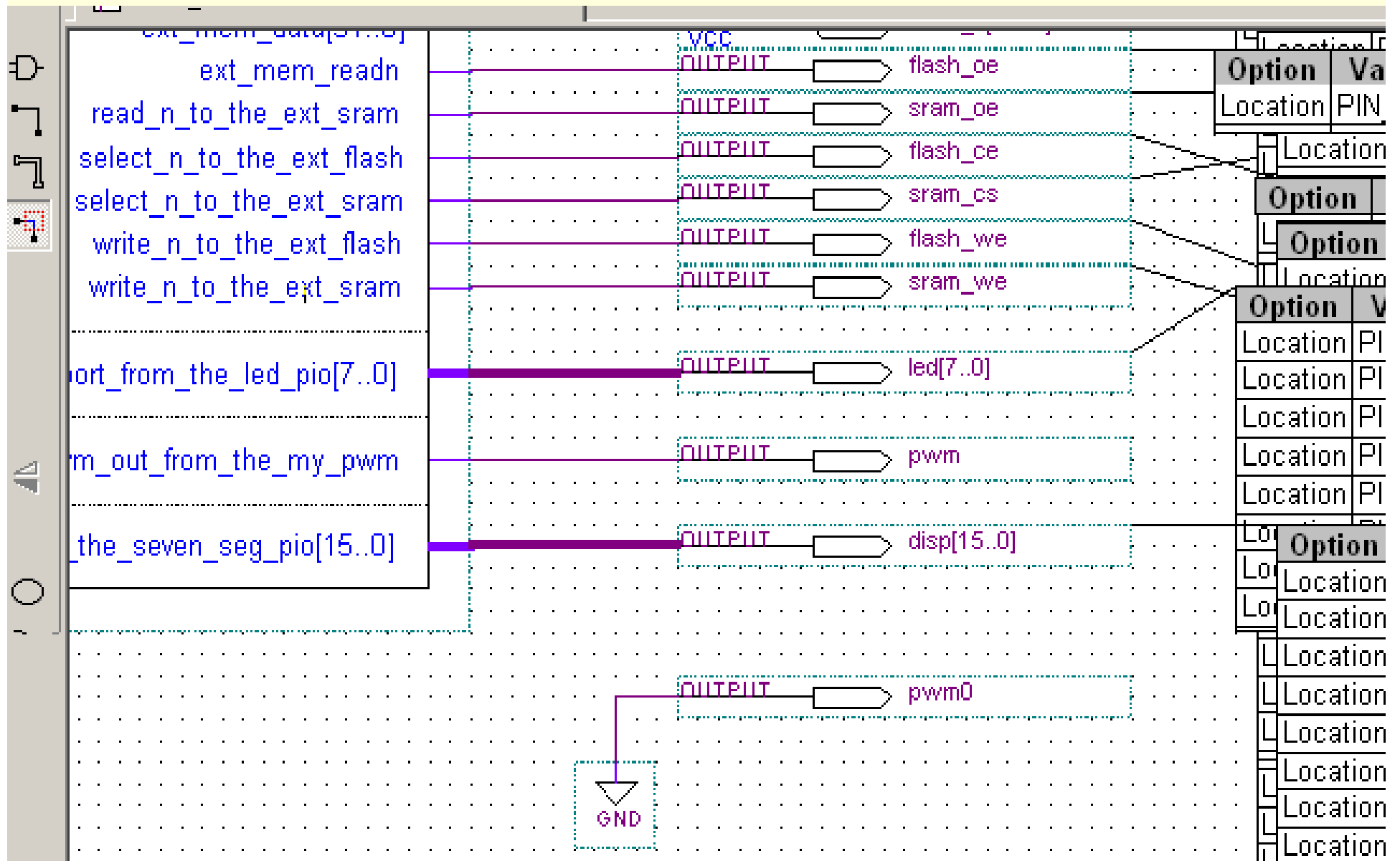


图10-61 在更新的NiosII模块中加入端口信号

10.3 加入用户自定义组件设计

2、生成和编译

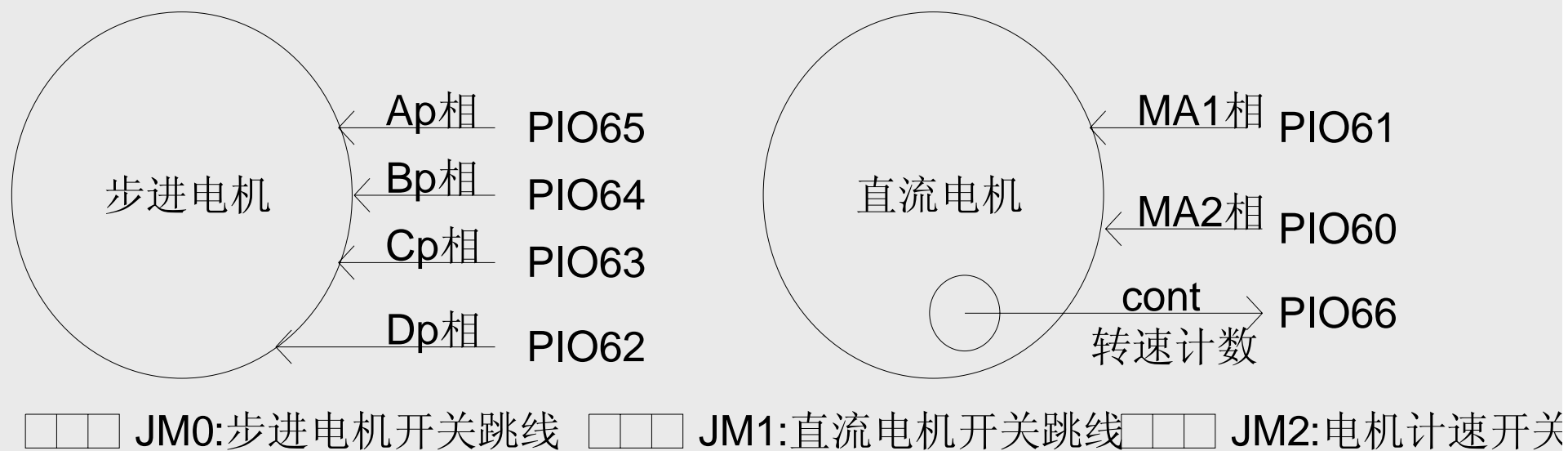


图10-62 电机引脚原理图

10.3 加入用户自定义组件设计

- 3、锁定FPGA控制电机的引脚
- 4、建立软件工程

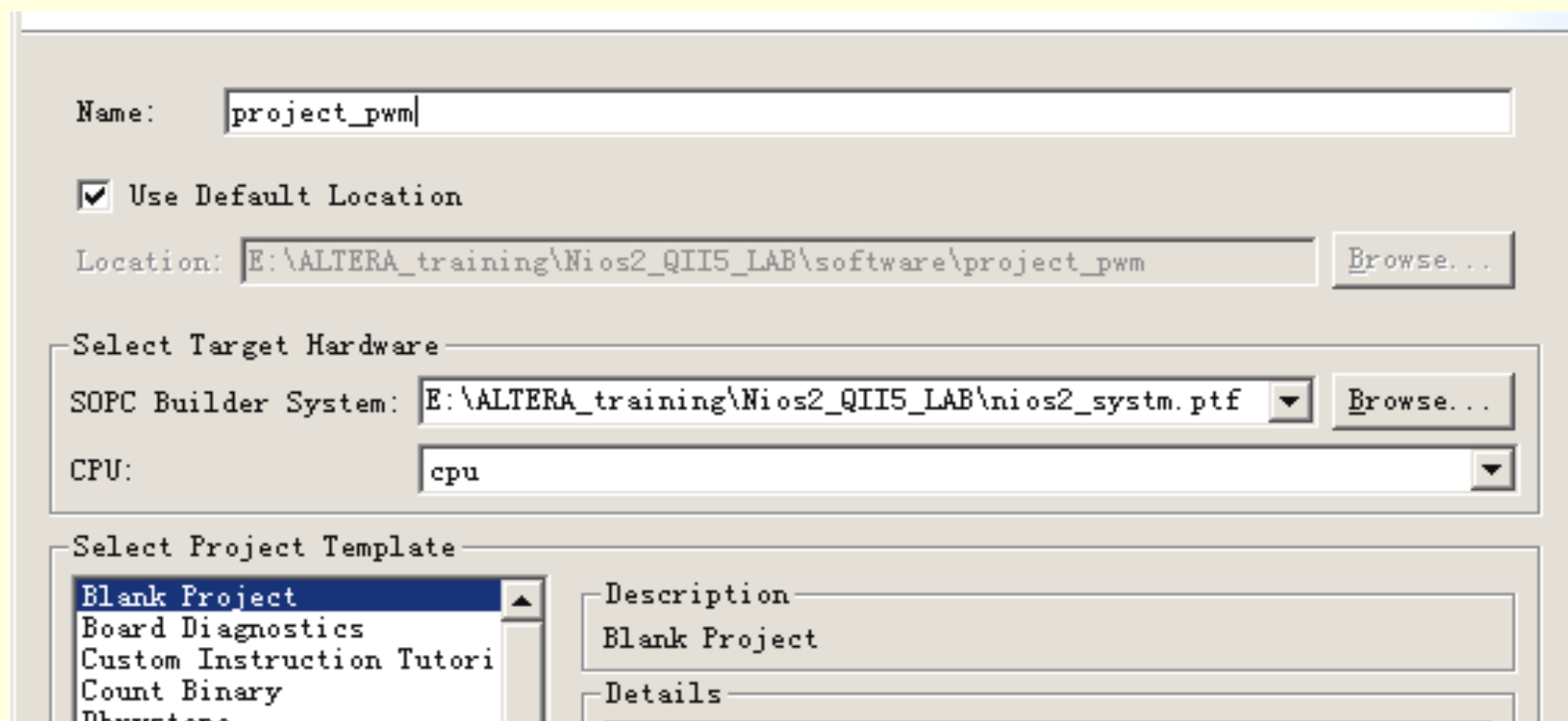


图10-63 建立一个空的软件工程project_pwm

10.3加入用户自定义组件设计

4、建立软件工程

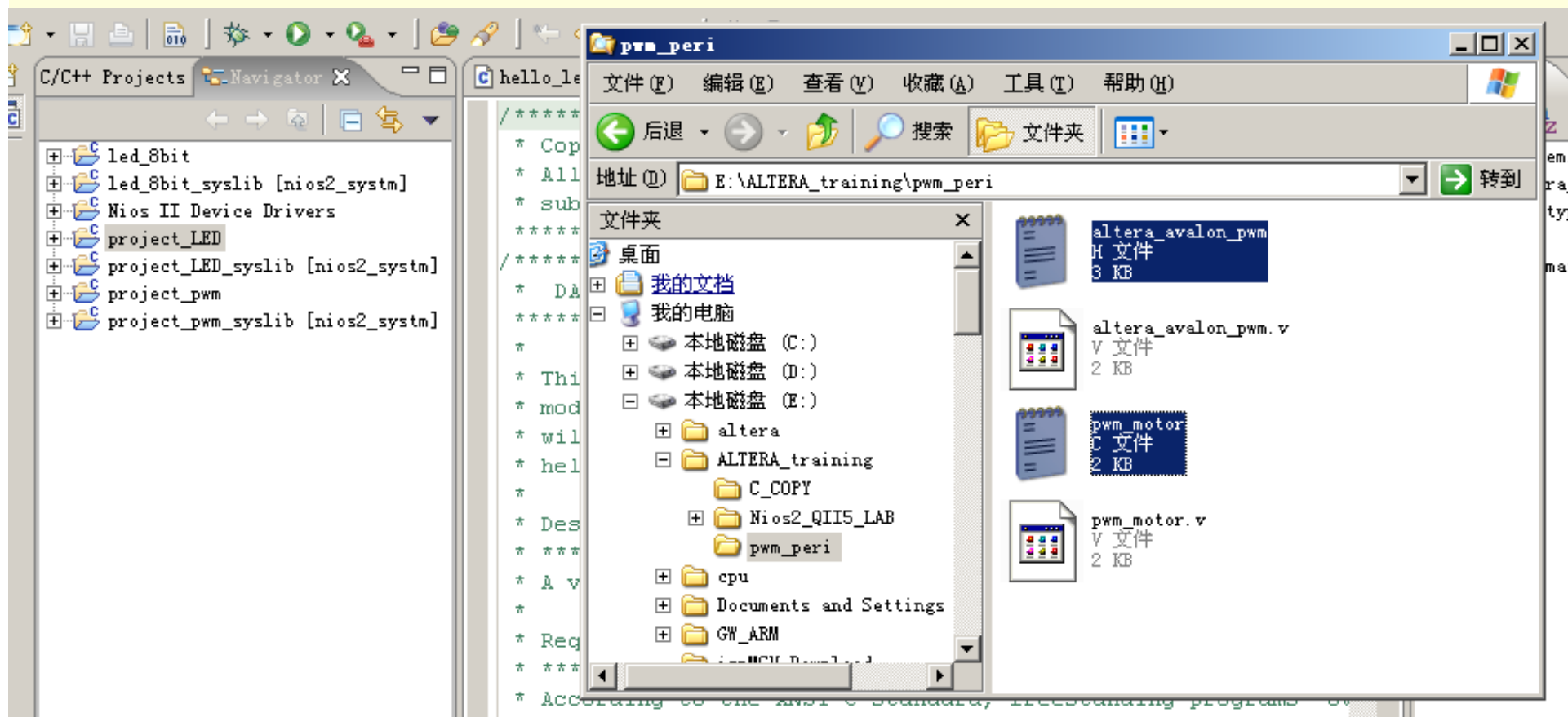


图10-64 将2工作软件拖入Navigator栏的空工程中

10.3 加入用户自定义组件设计

4、建立软件工程

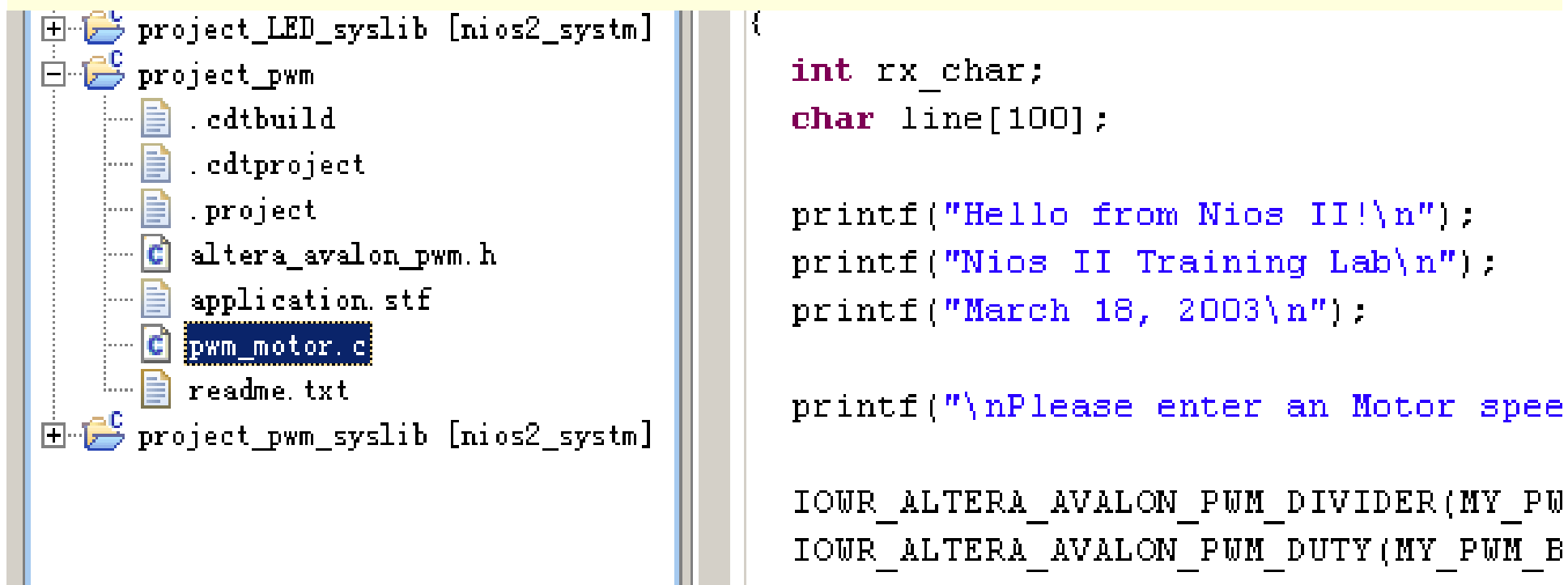


图10-65 观察C/C++ Projects栏中被加入的源程序pwm_motor.c

10.3 加入用户自定义组件设计

5、运行和调试软件

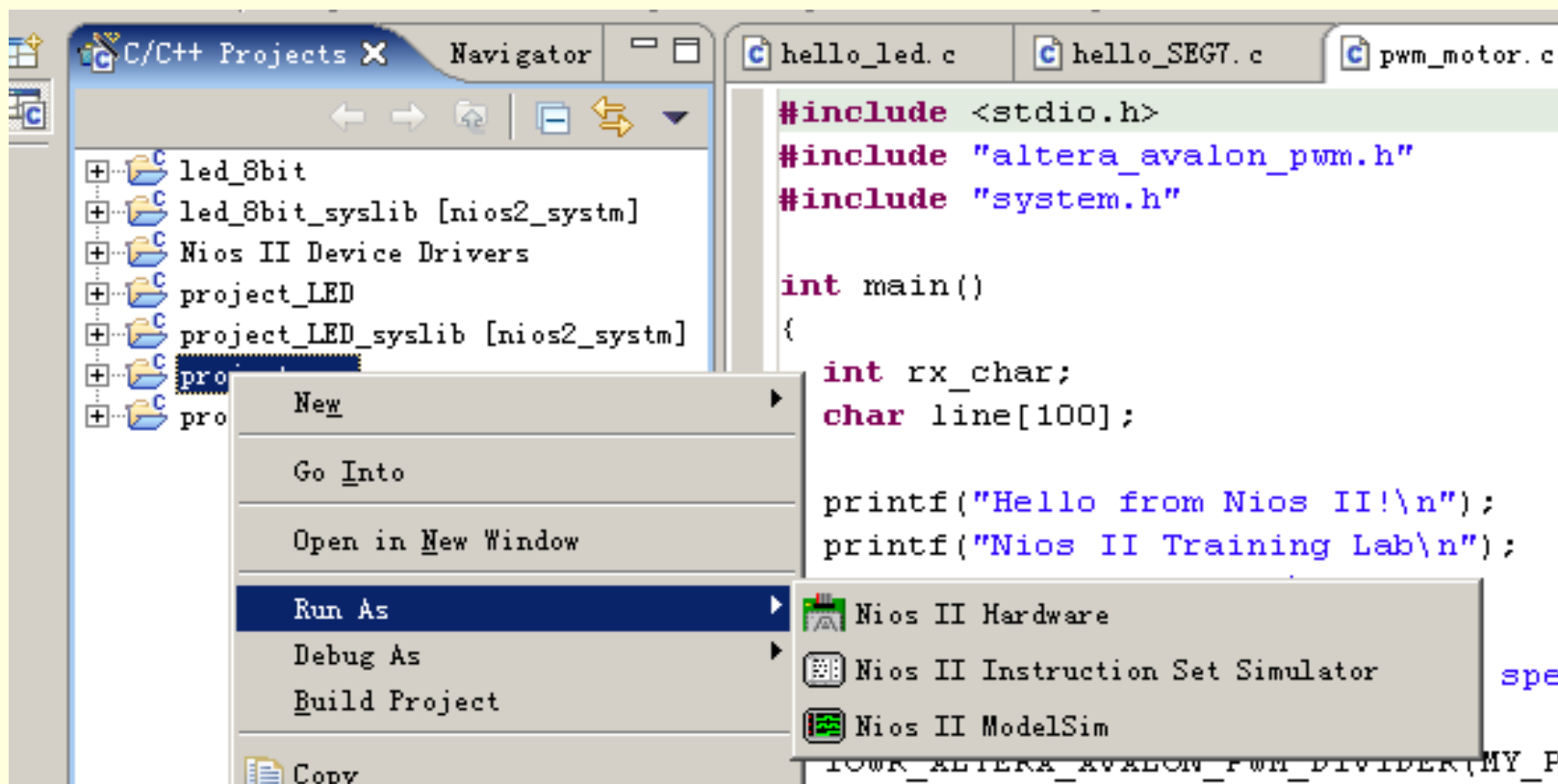
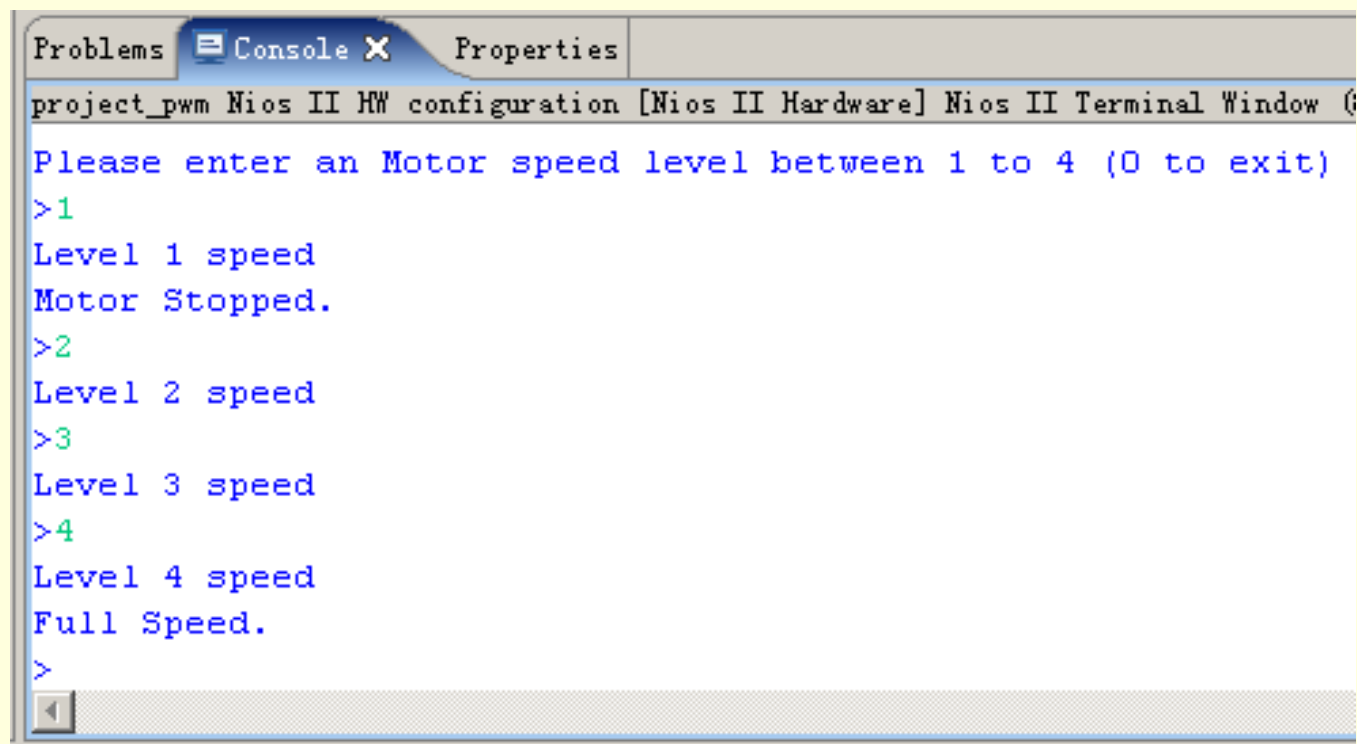


图10-66 编译并全速运行该程序

10.3 加入用户自定义组件设计

5、运行和调试软件



```
Problems Console Properties
project_pwm Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (
Please enter an Motor speed level between 1 to 4 (0 to exit)
>1
Level 1 speed
Motor Stopped.
>2
Level 2 speed
>3
Level 3 speed
>4
Level 4 speed
Full Speed.
>
```

图10-67 软件对电机运行和操作过程

10.3加入用户自定义组件设计

6、加入电机测速电路

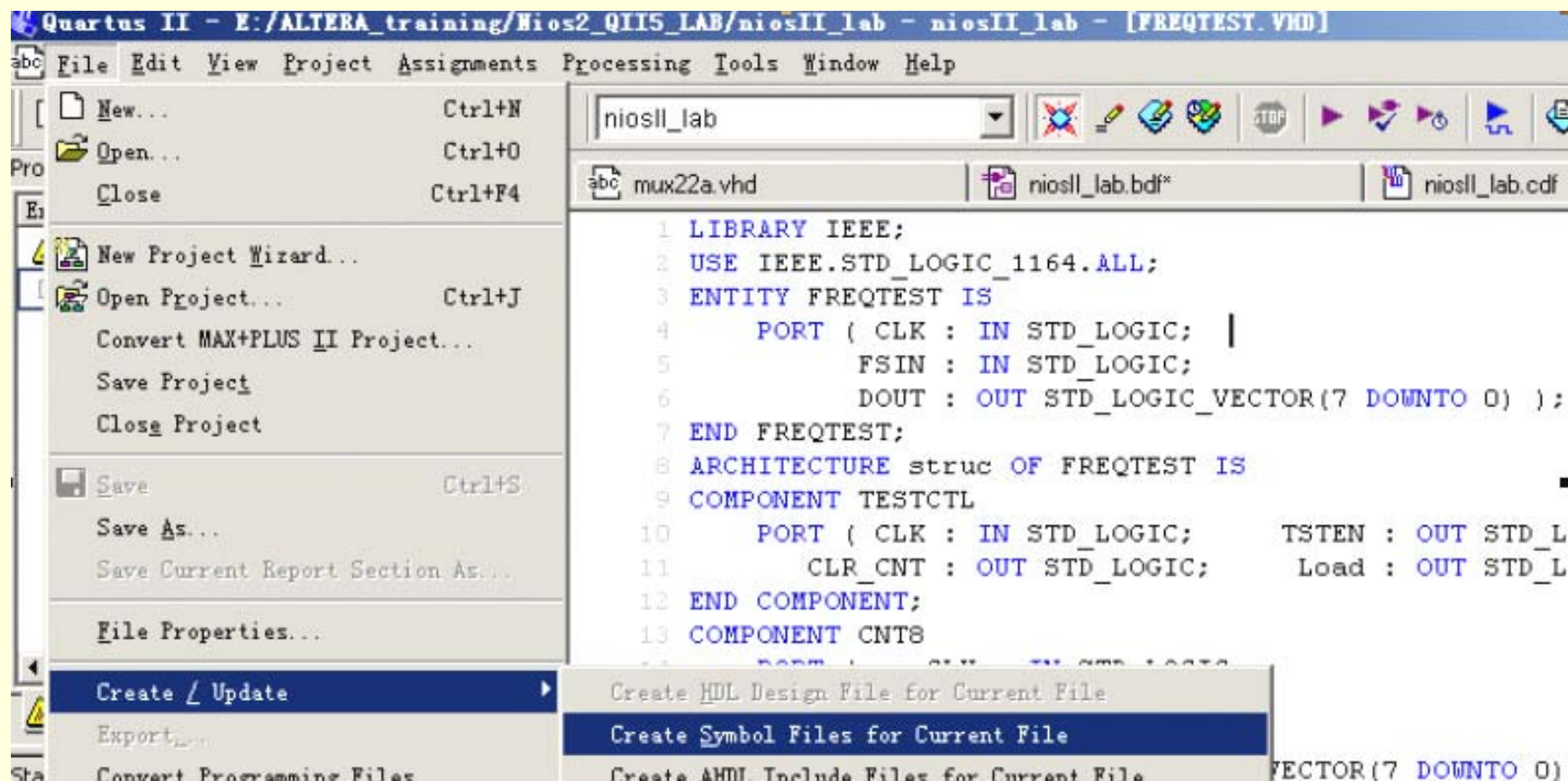


图10-68 将频率计VHDL顶层文件生成一个原理图元件

10.3 加入用户自定义组件设计

6、加入电机测速电路

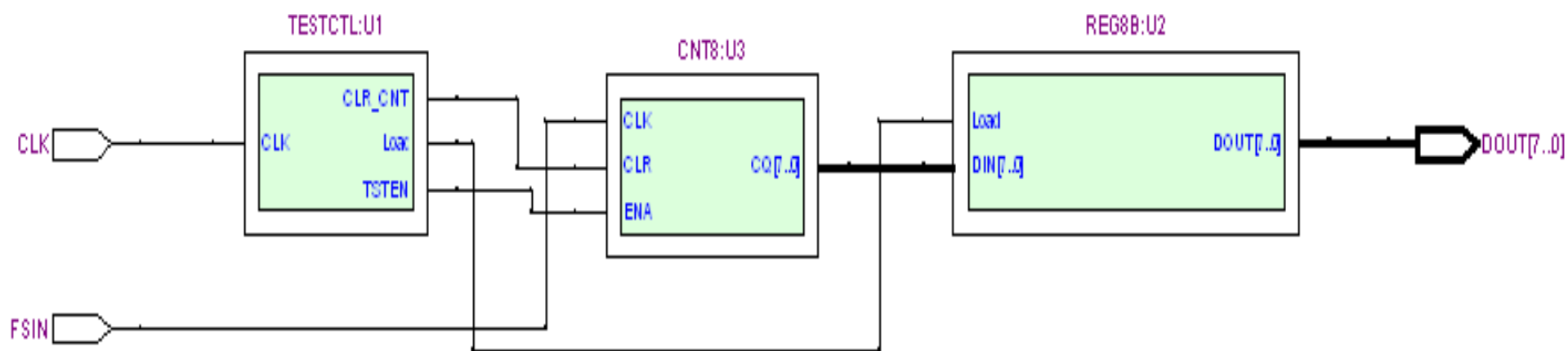


图10-69 频率计FREQTEST的RTL

10.3 加入用户自定义组件设计

6、加入电机测速电路

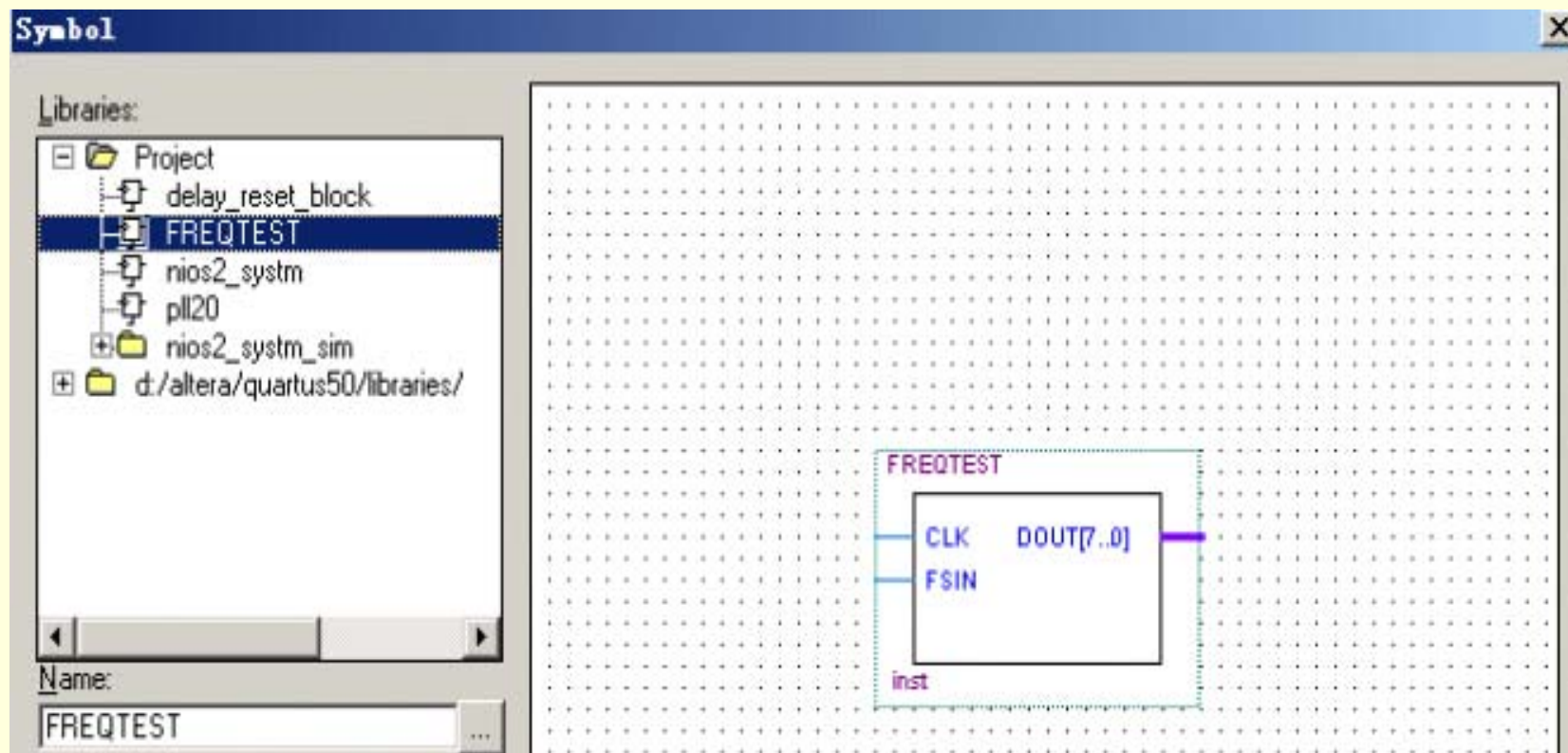


图10-70 向主系统原理图调入频率计原理图元件

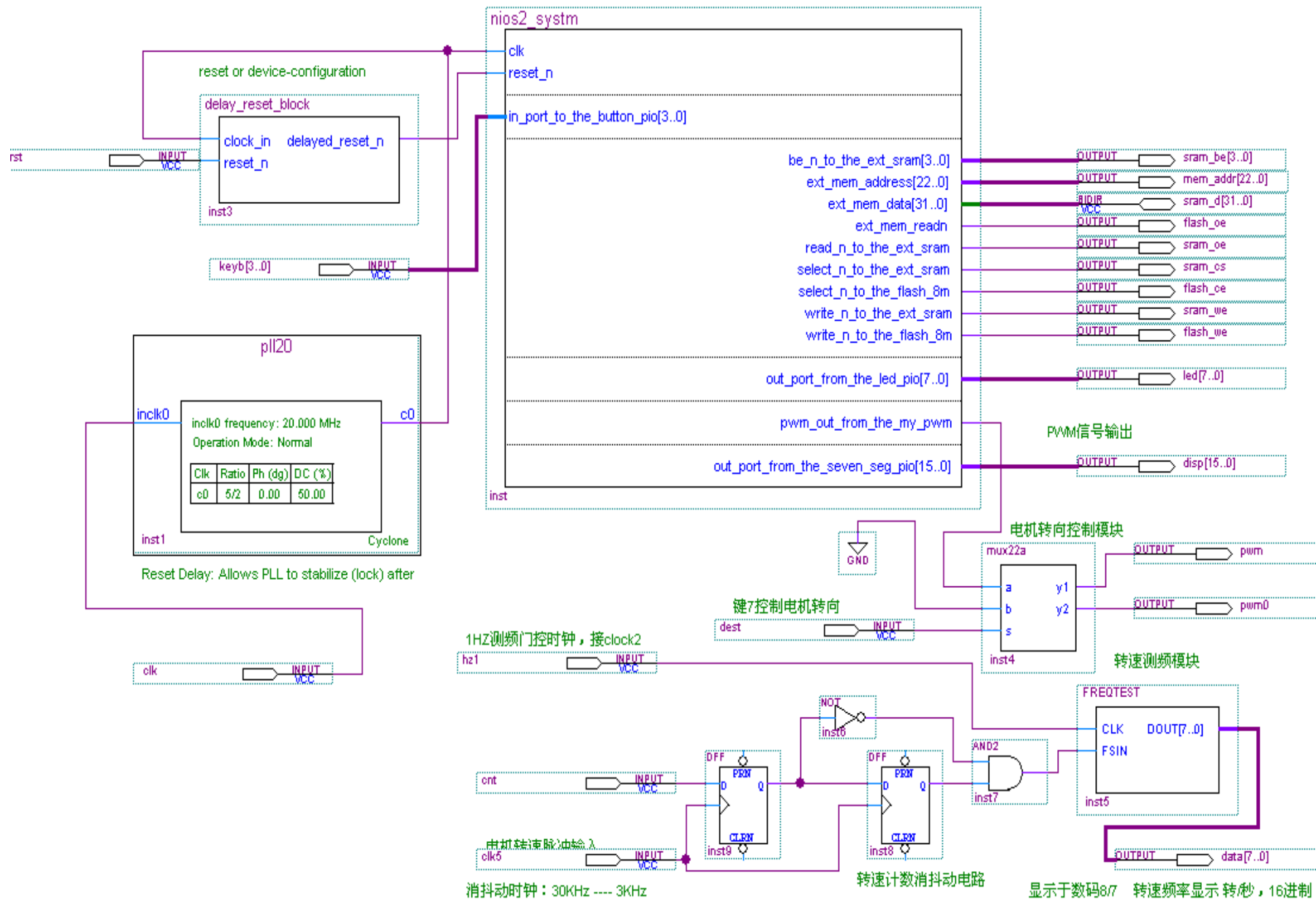


图10-71 NiosII系统完整的原理图

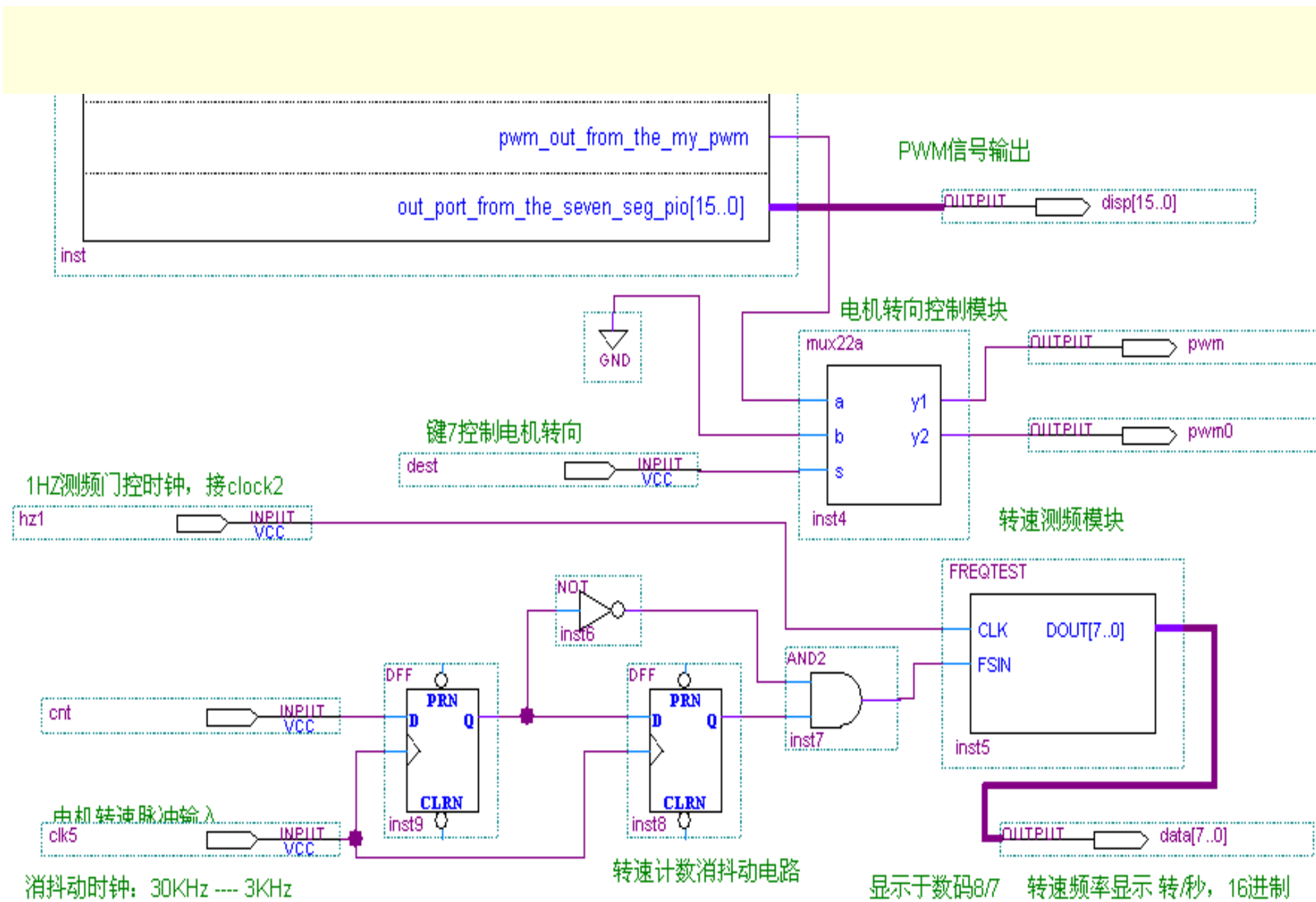


图10-72 NiosII系统电机控制与转速测定/显示电路

10.3加入用户自定义组件设计

7、运行软件

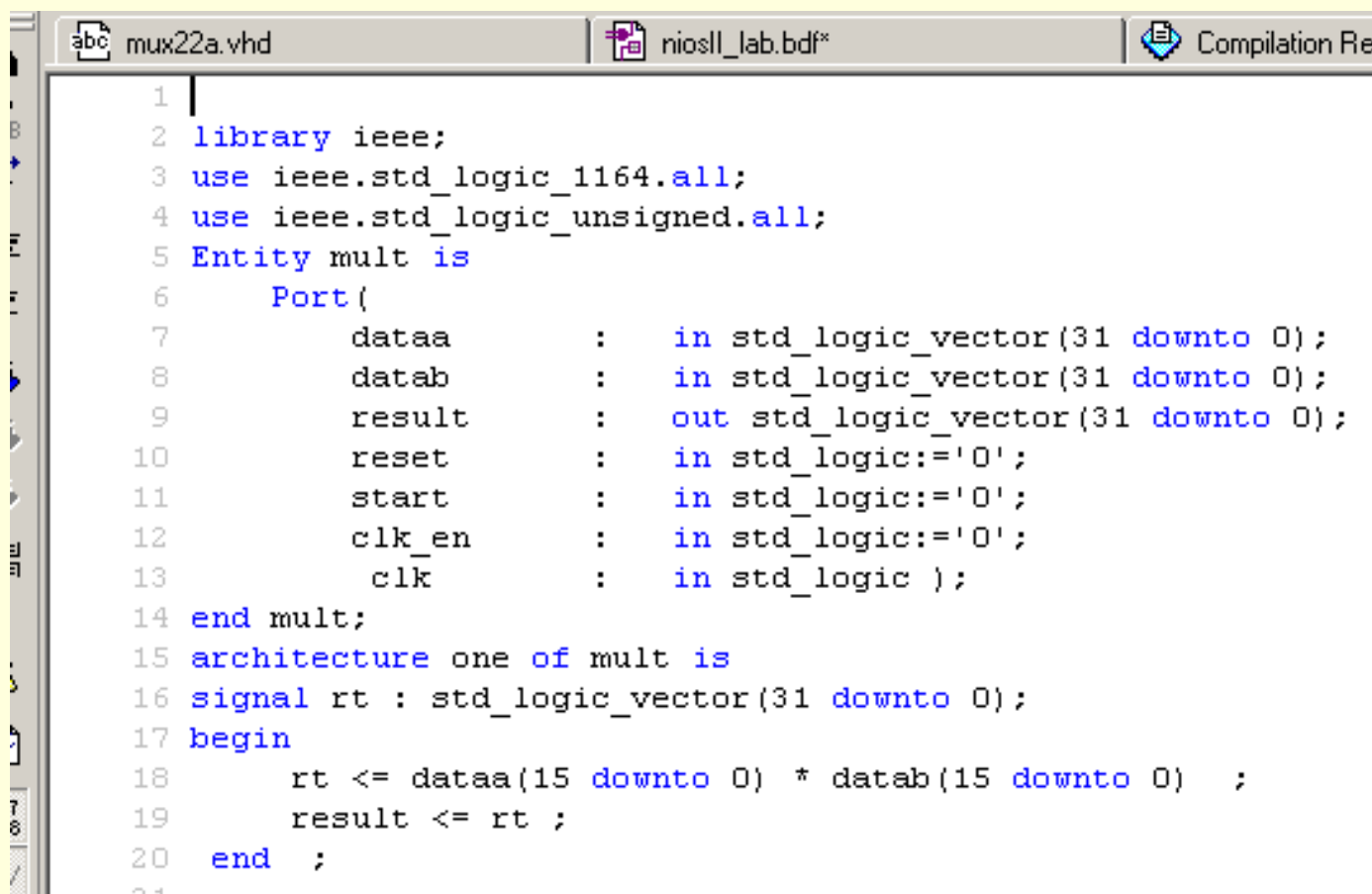
首先将此电路系统从新全程编译一次，下载后，再将以上的软件程序下载运行，观察转速控制和变化情况；同时了解电机旋转方向的控制情况。

10.4 加入用户自定义指令设计

【例10-2】 mult.VHD

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
Entity mult is
    Port(
        dataa      :      in std_logic_vector(31 downto 0);
        datab     :      in std_logic_vector(31 downto 0);
        result     :      out std_logic_vector(31 downto 0);
        reset      :      in std_logic:='0';
        start      :      in std_logic:='0';
        clk_en     :      in std_logic:='0';
        clk        :      in std_logic );
end mult;
architecture one of mult is
    signal rt : std_logic_vector(31 downto 0);
begin
    rt <= dataa(15 downto 0) * datab(15 downto 0) ; result <= rt ;
end ;
```

10.4 加入用户自定义指令设计



The image shows a screenshot of a VHDL editor window. The window title bar contains three tabs: 'abc mux22a.vhd', 'niosll_lab.bdf*', and 'Compilation Re'. The main text area displays the following VHDL code:

```
1 |
2 library ieee;
3 use ieee.std_logic_1164.all;
4 use ieee.std_logic_unsigned.all;
5 Entity mult is
6     Port(
7         dataa      : in std_logic_vector(31 downto 0);
8         datab      : in std_logic_vector(31 downto 0);
9         result      : out std_logic_vector(31 downto 0);
10        reset       : in std_logic:= '0';
11        start        : in std_logic:= '0';
12        clk_en       : in std_logic:= '0';
13        clk          : in std_logic );
14 end mult;
15 architecture one of mult is
16 signal rt : std_logic_vector(31 downto 0);
17 begin
18     rt <= dataa(15 downto 0) * datab(15 downto 0) ;
19     result <= rt ;
20 end ;
21
```

图10-75 乘法指令VHDL程序

10.4 加入用户自定义指令设计

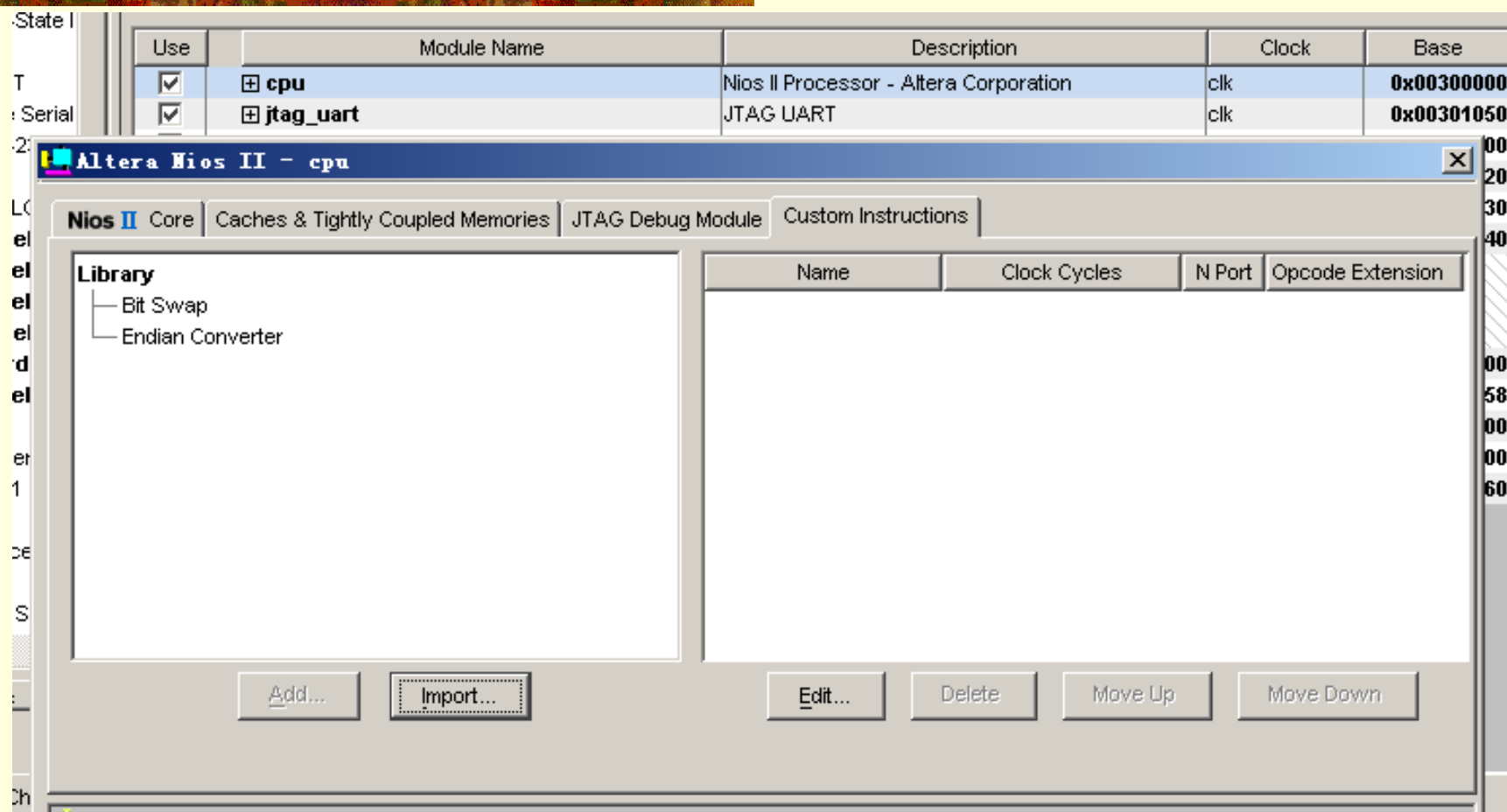


图10-76 打开CPU的自定义指令对话框

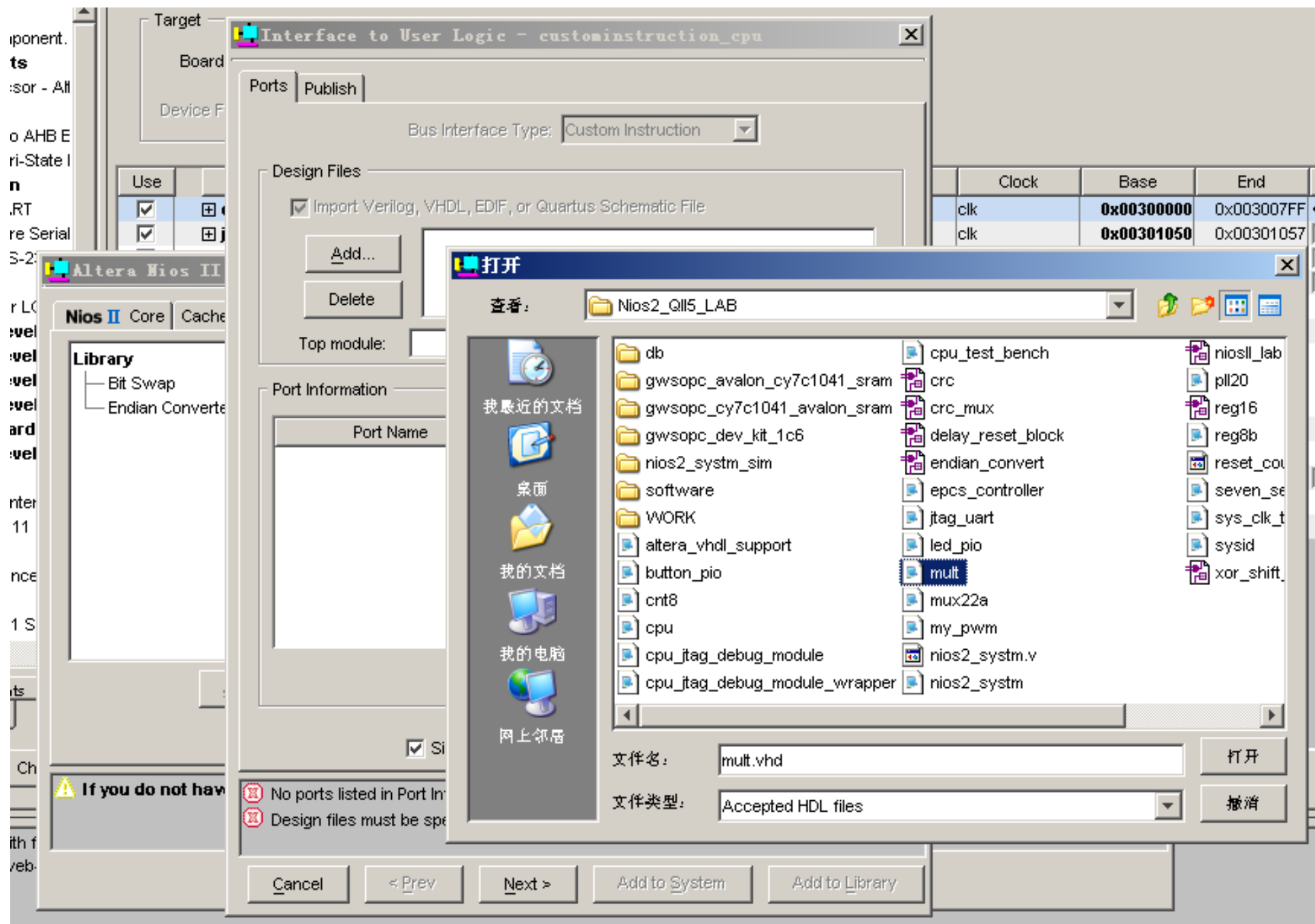


图10-77 加入已设计好的乘法指令VHDL程序

10.4 加入用户自定义指令设计

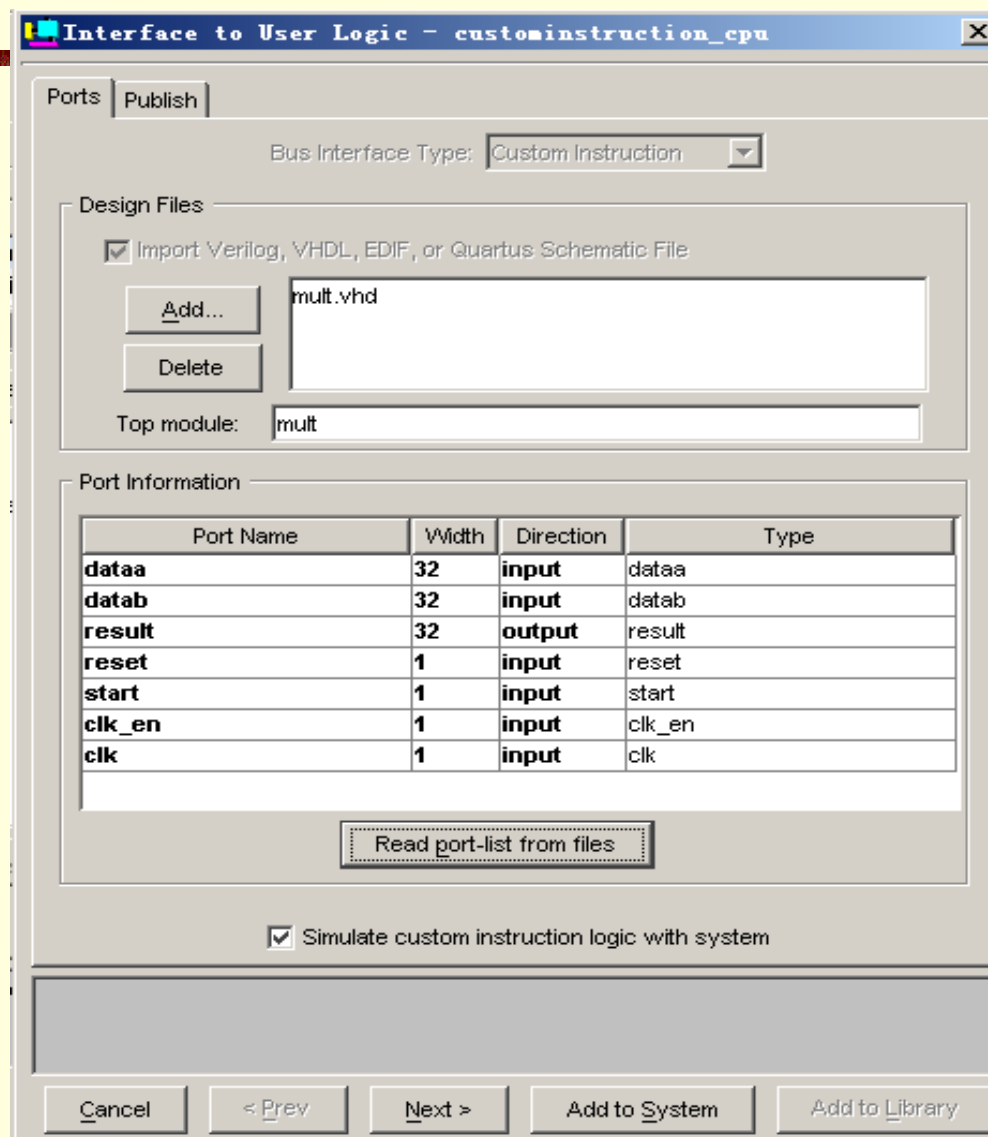


图10-78 读入程序的端口信号

10.4 加入用户自定义指令设计

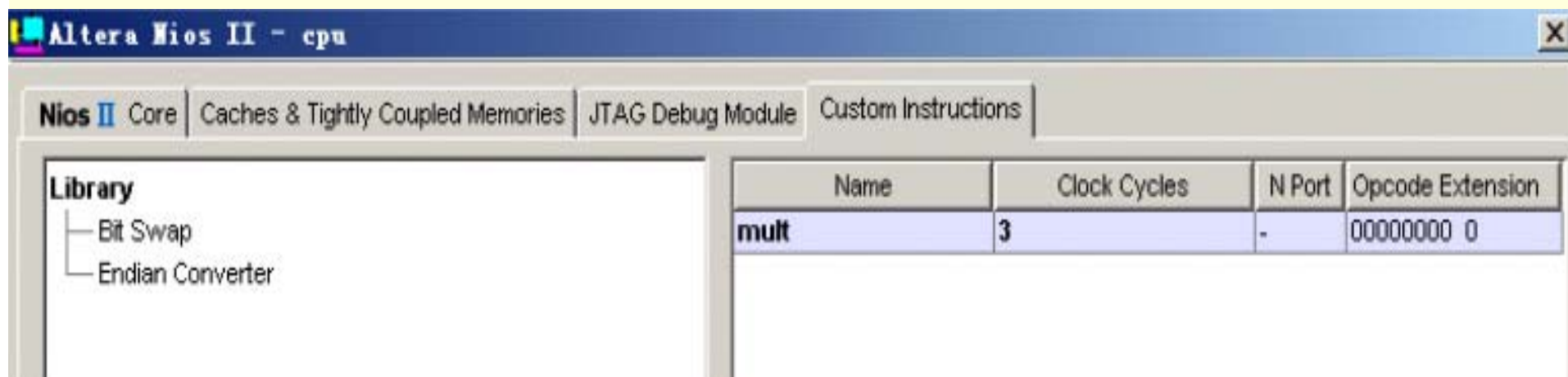


图10-79 已加入一条名为mult的乘法指令

10.4 加入用户自定义指令设计

【例10-3】

```
#include "system.h"
int main (void)
{
    int x,y,z,f ;
    xr=569;
    yr=1923;;
    x=xr;
    y=yr;
    printf("\n%08x %08x \n",x,y);
    z=ALT_CI_MULT(x,y);    //使用ALT_CI_MULT(x,y)调用了乘法
    自定义指令，注意指令//要大写
    printf("\n%08x %08x %08x\n",x,y,z);
}
```

10.4 加入用户自定义指令设计

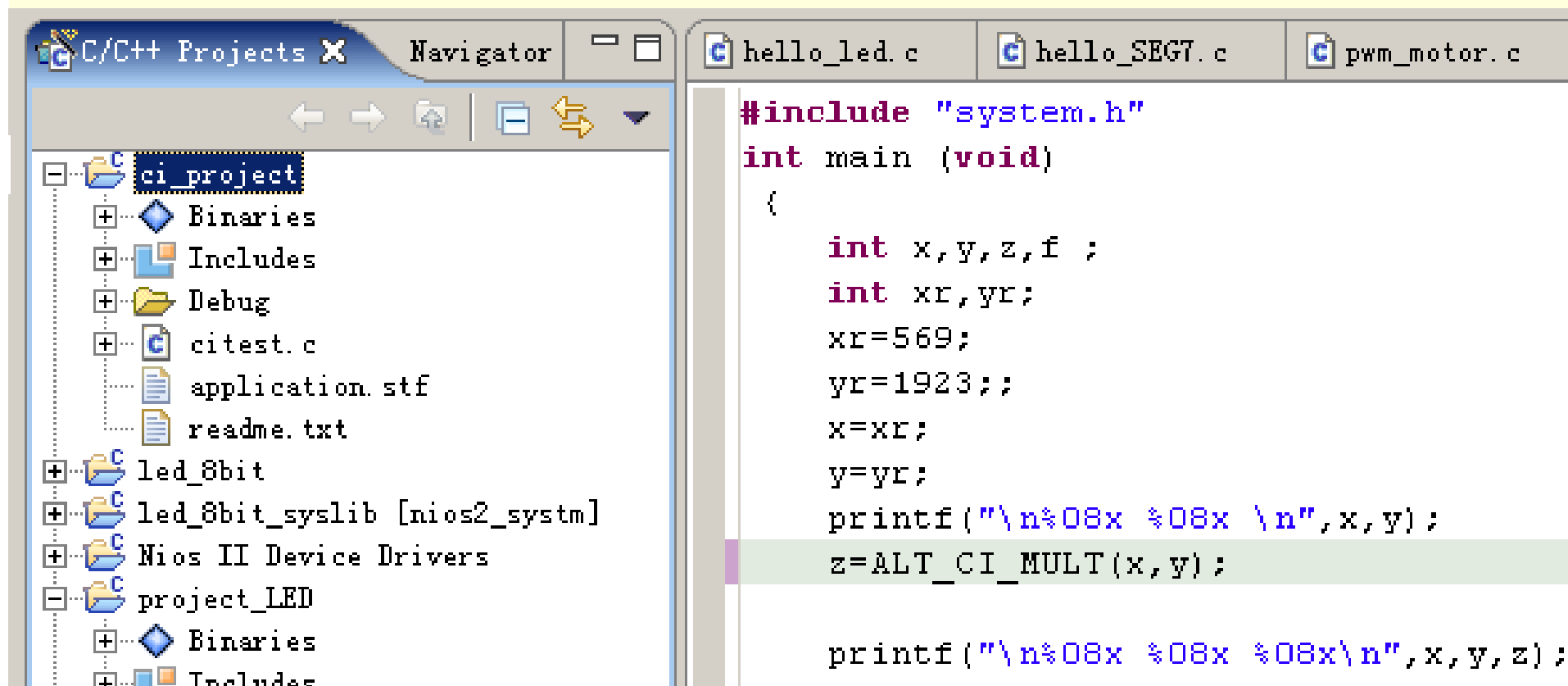
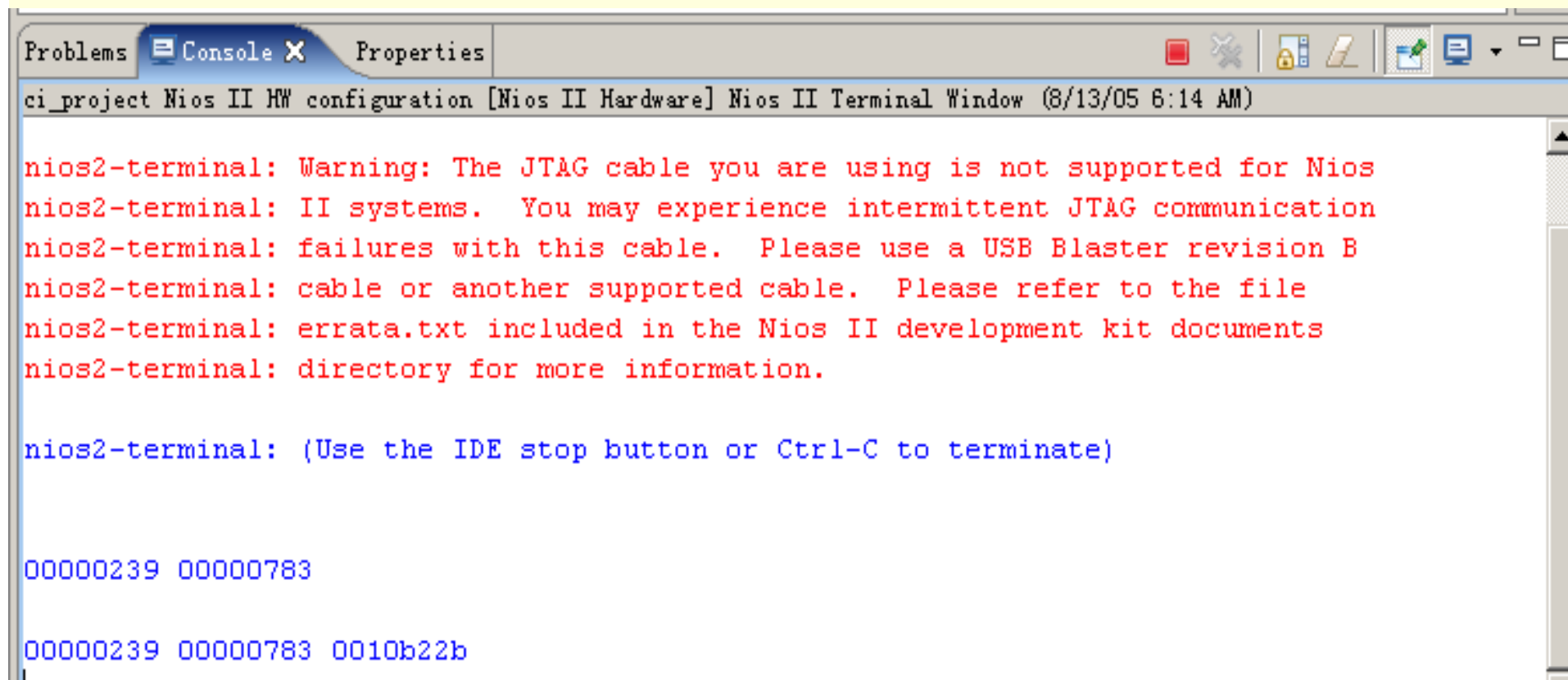


图10-80 自定义指令C工程: ci_project

10.4 加入用户自定义指令设计



```
ci_project Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (8/13/05 6:14 AM)

nios2-terminal: Warning: The JTAG cable you are using is not supported for Nios
nios2-terminal: II systems. You may experience intermittent JTAG communication
nios2-terminal: failures with this cable. Please use a USB Blaster revision B
nios2-terminal: cable or another supported cable. Please refer to the file
nios2-terminal: errata.txt included in the Nios II development kit documents
nios2-terminal: directory for more information.

nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

00000239 00000783

00000239 00000783 0010b22b
```

图10-81 自定制指令mult执行结果

10.5 Flash编程下载

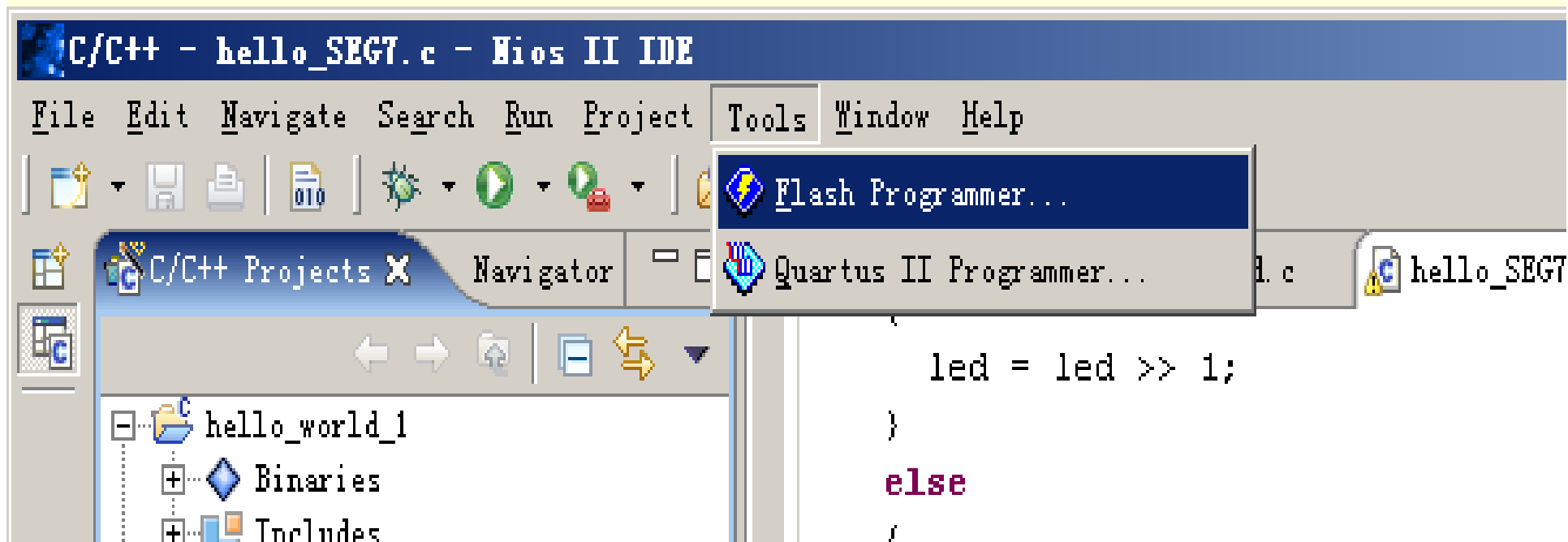


图10-82 打开Flash编程对话框

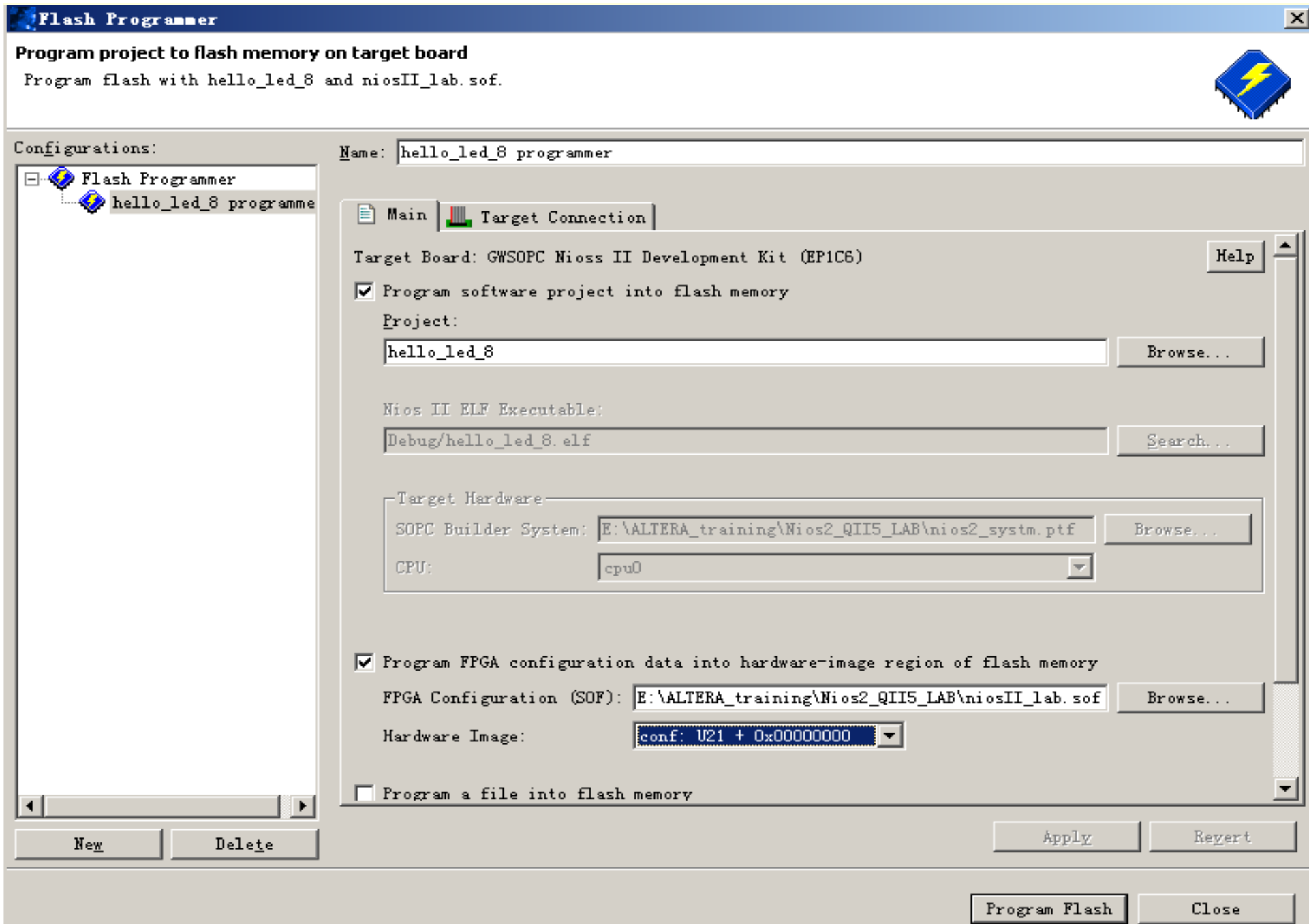


图10-83 在Flash编程对话框中完成必要设置

```
Problems Console X Properties
<terminated> led_8bit programmer [Flash Programmer] led_8bit_programmer.sh (06-8-30 上午9:12)

00010000 ( 0%): Erasing

Erased 64kB in 1.0s (64.0kB/s)

00010000 ( 0%): Programming

Programmed 1KB +63KB in 1.6s (40.0kB/s)
Did not attempt to verify device contents
Leaving target processor paused

# Creating .flash file for the project
$SOPC_KIT_NIOS2/bin/elf2flash --base=0x00000000 --end=0x7fffff --reset=0x900800
--input=led_8bit.elf --output=ext_flash.flash --boot=$SOPC_KIT_NIOS2/components/
altera_nios2/boot_loader_cfi.srec

# Programming flash with the project
$SOPC_KIT_NIOS2/bin/nios2-flash-programmer --cable='USB-Blaster [USB-0]' --devic
e=1 --sidp=0x00901058 --id=3315650538 --timestamp=1156899436 --base=0x00000000 e
xt_flash.flash
Using cable "USB-Blaster [USB-0]", device 1, instance 0x00
Resetting and pausing target processor: OK
Reading System ID at address 0x00901058: verified
Leaving target processor paused
```

图10-84 对Flash烧写编程的消息

10.5 Flash编程下载

Processor Configuration

Nios II/s Core

4-Kbyte Instruction Cache (128 lines, 32 bytes/line, 20 tag bits/line)

JTAG Debug Module (SW breakpoints, 2 HW breakpoints, 2 data triggers, on-chip trace)

Processor Function	Memory Module	Offset	Address
Reset Address	ext_flash	0x00000000	0x00000000
Exception Address	ext_sram	0x00000020	0x00800020
Break Location	cpu/jtag_debug_module	0x00000020	0x00900020

You can change **Nios II software settings**, such as data memory, host communication, and debugging communication, in the System Library properties of the Nios II IDE.

图10-85 将复位地址设置于外部Flash



习题

- 10-1. 请简要叙述基于NiosII的嵌入式系统的设计流程。与通常的嵌入式系统（如ARM）的开发在设计流程上有什么区别？
- 10-2. 请详细叙述基于NiosII的嵌入式系统的硬件开发流程。
- 10-3. 请详细叙述基于NiosII的嵌入式系统的软件开发流程。
- 10-4. 试在NiosII嵌入式系统上编写一个程序，通过JTAG-UART输出字符串“This is a test!”。



实验与设计

实验10-1.设计一个简单的NiosII系统

- (1) **实验目的** 掌握基于NiosII系统软硬件设计流程。熟悉SOPC Builder、QuartusII的使用。
- (2) **实验任务1** 按照本章所述的硬件设计流程，设计一个简单的基于NiosII的嵌入式系统，并下载到GW48实验系统中，进行软硬件测试。
- (3) **实验任务2** 修改NiosII系统，增加一个串口命名为UART2（波特率为9600bps），然后生成系统，在QuartusII中添加相应得UART引脚，以对应GW48实验系统的串口。使用QuartusII进行编译并下载到实验系统上，使UART2连接PC机。

接着，编写C程序使JTAG-UART输出下列字符串：

```
This is a test!
```

```
Test OK!
```

试使用IDE完成上面的编编译下载工作，并进行调试。



实验与设计

实验10-2. 秒表程序设计

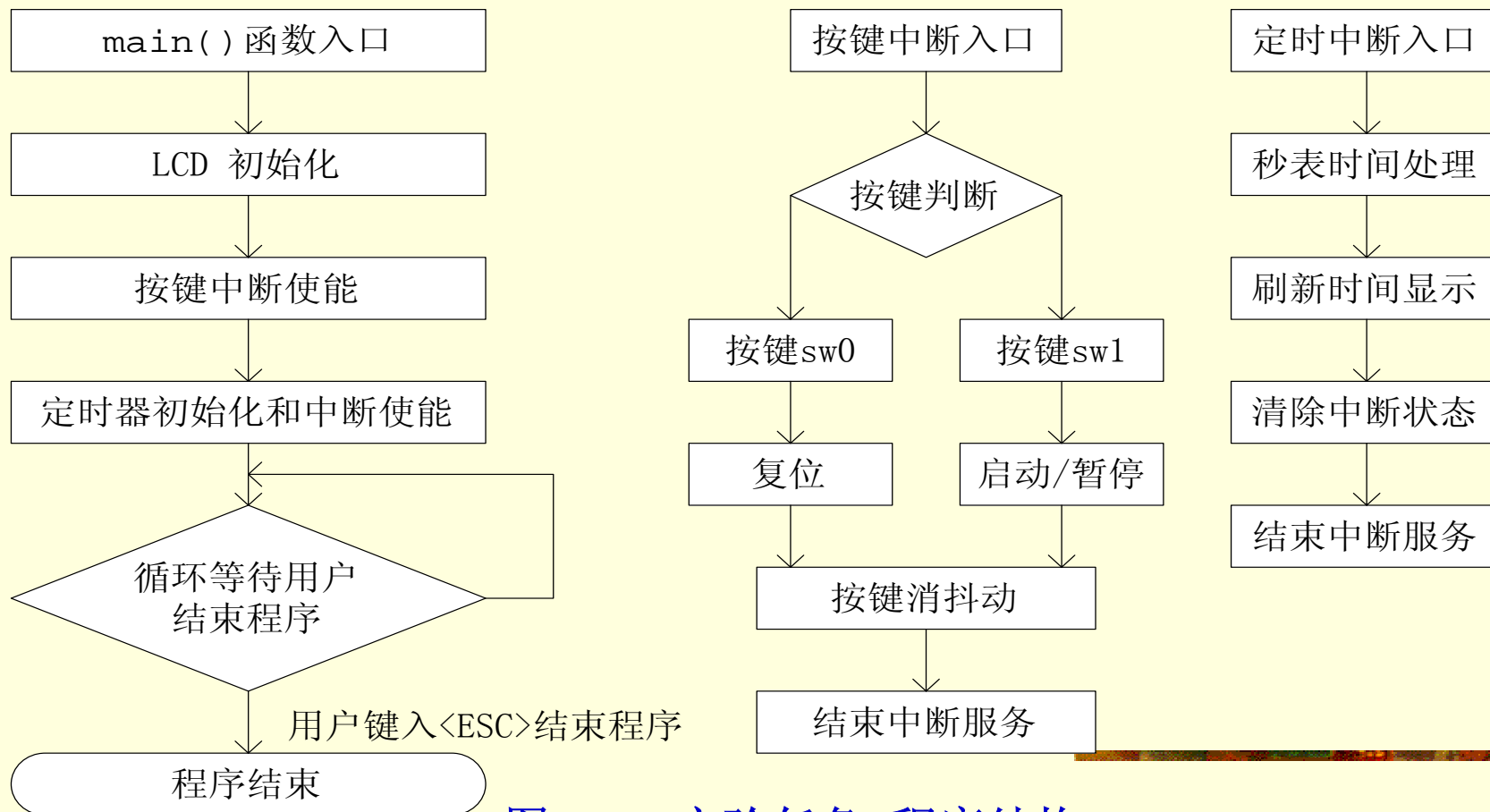


图10-86 实验任务1程序结构

实验10-3. 为NiosII系统定制复数乘法器硬件加速指令

【例10-4】

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Entity comp is
Port(
    dataa,datab : in std_logic_vector(31 downto 0);
    clk,clk_en,reset,start : in std_logic;
    result : out std_logic_vector ( 31 downto 0)
);
End entity comp;
Architecture bhv of comp is
    signal xr,xi : std_logic_vector(15 downto 0);
    signal yr,yi : std_logic_vector(15 downto 0);
    signal zr,zi : std_logic_vector(31 downto 0);
Begin
    xr <= dataa(15 downto 0);
    xi <= dataa(31 downto 16);
    yr <= datab(15 downto 0);
    yi <= datab(31 downto 16);
    zr <= xr * yr - xi * yi;
    zi <= xr * yi + xi * yr;
    result<= zi(15 downto 0) & zr(15 downto 0);
end;
```



实验与设计

实验10-3. 为NiosII系统定制复数乘法器硬件加速指令

【例10-5】

```
#include <nios.h>
int main(void)
{
    int x,y,z;
    int xr,xi,yr,yi;
    xr=11;xi=28;
    yr=23;yi=17;
    printf("\nx=%di+%d y=%di+%d\n",xi,xr,yi,yr);
    x=xi*0x10000+xr;
    y=yi*0x10000+yr;
    printf("x=%08x y=%08x\n",x,y);
    z=ALT_CT_comp(x,y);    //使用nm_comp()调用了自定义指令
    printf("\n%08x %08x %08x\n",x,y,z);
}
```

//数据验证结果: $(28i+11)x(17i+23)=i831-223=i33FH + FF21H$,
RESULT=033FFF21



实验与设计

实验10-3. 为NiosII系统定制复数乘法器硬件加速指令

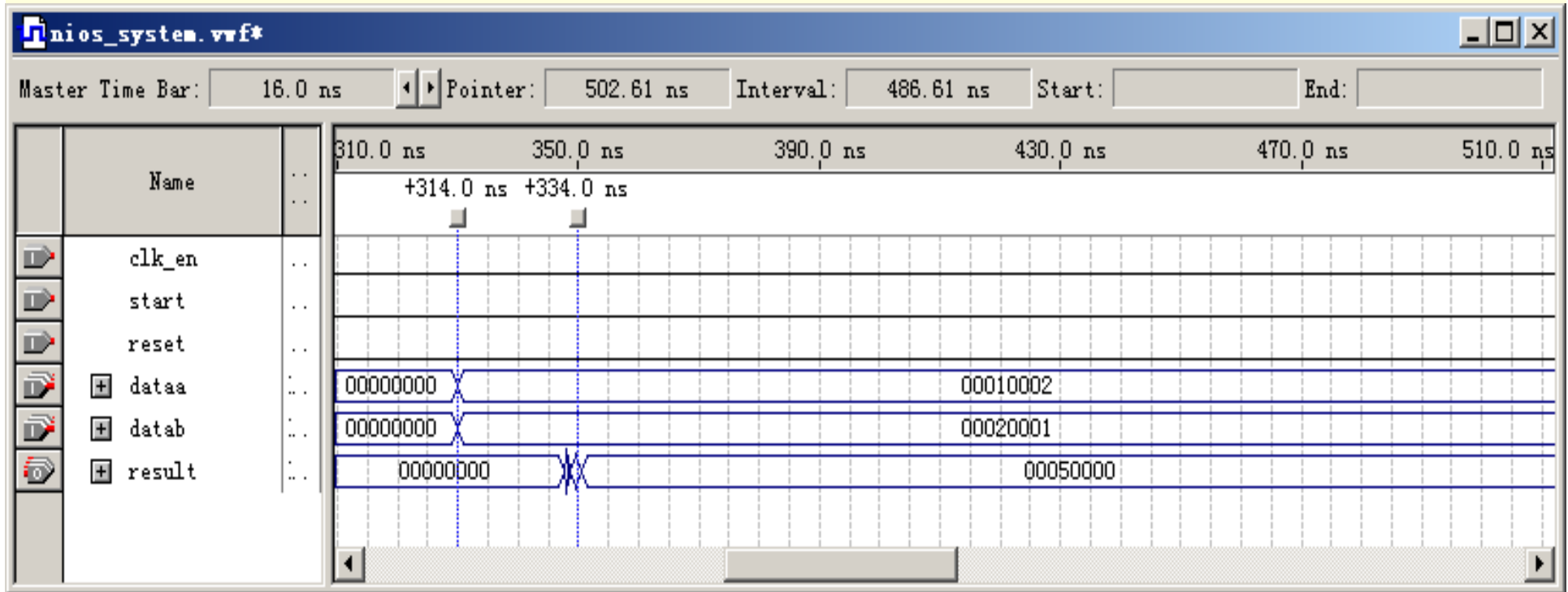


图10-87 comp.vhd的仿真结果



实验与设计

实验10-4. 利用NiosII系统完成简单计算器程序设计与功能实现

实验10-5. Avalon Slave外设（数码管动态扫描显示模块）设计
