STEP-MAX10 Nios II 搭建手册

小脚丫 STEPFPGA

STEP

2017/8/16

目录

1. Nios II 软核处理器概述

Nios Ⅱ嵌入式处理器是 ALTERA 公司推出的采用哈佛结构、具有 32 位指令集的第二代片上可编程的软核处理器, 其最大优势和特点是模块化的硬件结构, 以及由此带来的灵活性和可裁减性。

相对于传统的处理器, Nios Ⅱ系统可以在设计阶段根据实际的需求来增减外设的数量和种类。设计者可以使用 ALTERA 提供的开发工具 SOPC Builder, 在 FPGA(现场可编程逻辑门阵列)器件上创建软硬件开发的基础平台, 也可用 SOPC Builder 创建软核 CPU 和参数化的接口总线 Avalon。在此基础上, 可以很快地将硬件系统(包括处理器、存储器、外设接口和用户逻辑电路)与常规软件集成在单一可编程芯片中。而且, SOPC Builder 还提供了标准的接口方式,以便用户将自己的外围电路做成 Nios Ⅱ软核可以添加的外设模块。这种设计方式, 更加方便了各类系统的调试。

Nios II 系列包括 3 种产品，分别是：Nios II/f（快速）——最高的系统性能，中等 FPGA 使用量；Nios II/s（标准）——高性能，低 FPGA 使用量；Nios II/e(经济)——低性能，最低的 FPGA 使用量。这 3 种产品具有 32 位处理器的基本结构单元——32 位指令大小，32 位数据和地址路径，32 位通用寄存器和 32 个外部中断源；使用同样的指令集架构（ISA），100%二进制代码兼容，设计者可以根据系统需求的变化更改 CPU，选择满足性能和成本的最佳方案，而不会影响已有的软件投入。

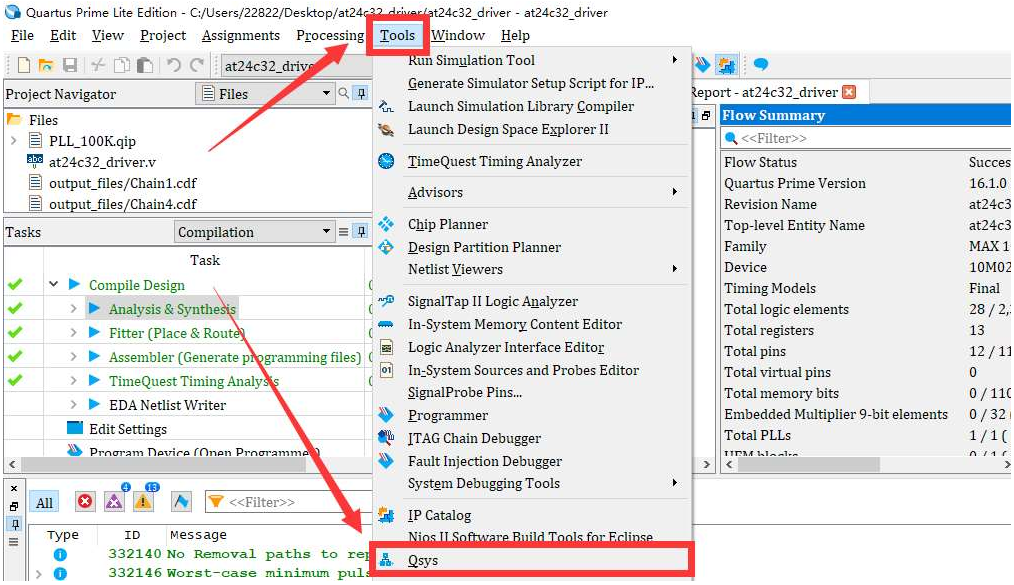
2. 利用 Qsys 构建软核平台

2.1 新建工程

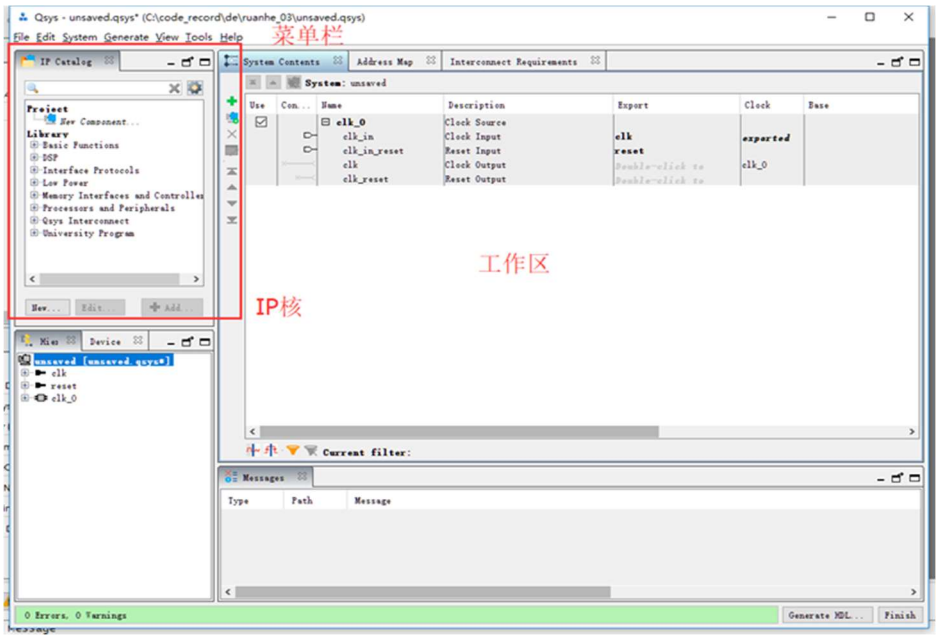
打开 Quartus Prime 并新建工程（注：工程须在纯英文路径下）。

2.2 在 Qsys 中搭建 Nios II

首先，点击界面内的 Qsys 图标，进入 Qsys 界面，你可以直接点击图标，也可以点击菜单栏中的“Tools”，在子菜单里找到该选项。



进入界面如下图所示：

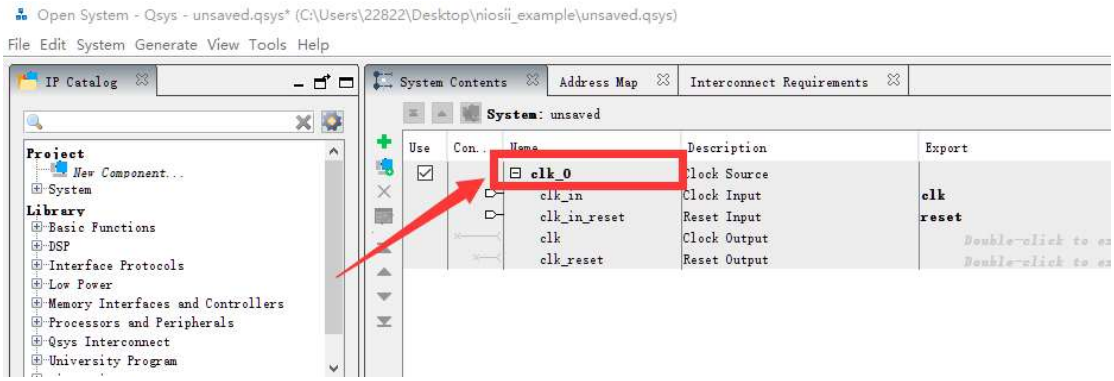


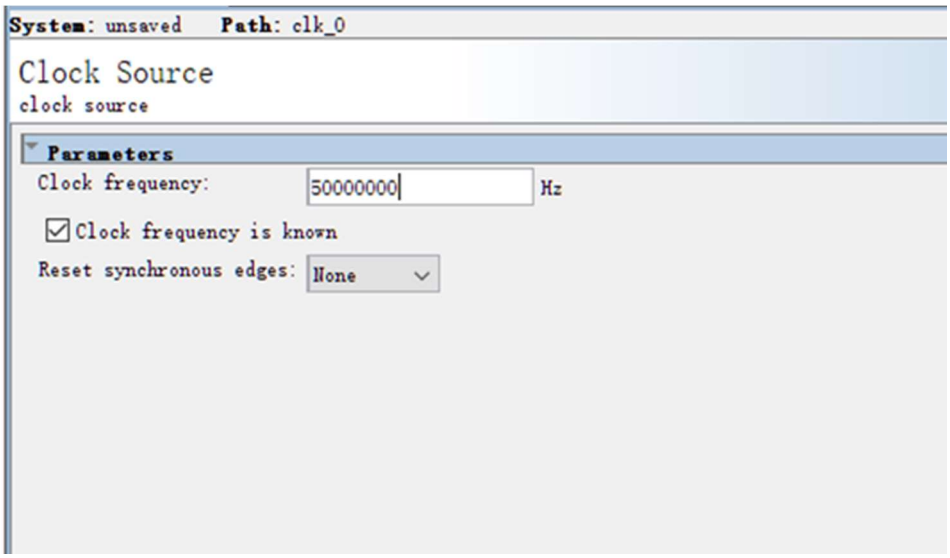
这个界面中，左上角是 IP 核，你也可以添加第三方的 IP 核，你也可以自己编写。中间的区域是工作区，在这里可以加入各种 IP 核并进行互联。

2.3 添加外设

2.3.1 更改时钟

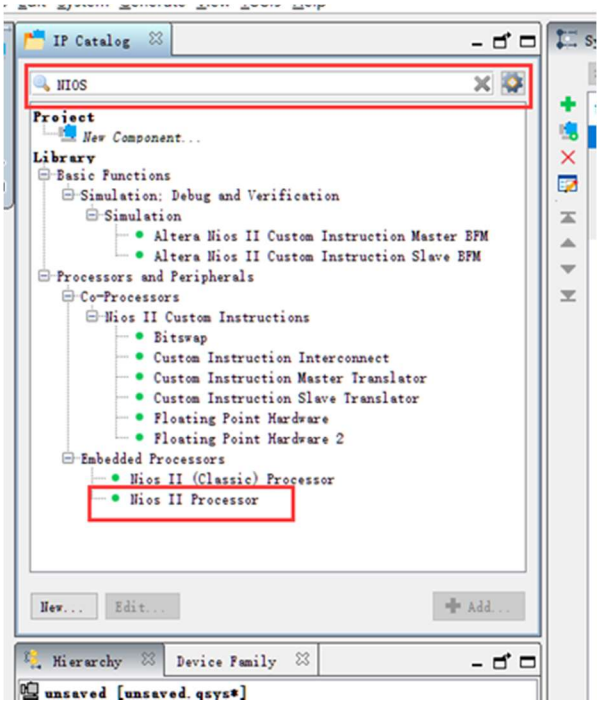
双击界面中的 clk_0，可以进行参数设置。一般来说，在这其中的时钟为外部输入时钟，频率是已知的，所以，勾选“Clock frequency is known”声明频率已知。此处输入的时钟频率应大于 20M。在这个选项下面，有一个名为“Reset synchronous edges”选项，代表是否需要同步复位，这里先暂时先就 None 就可以了。



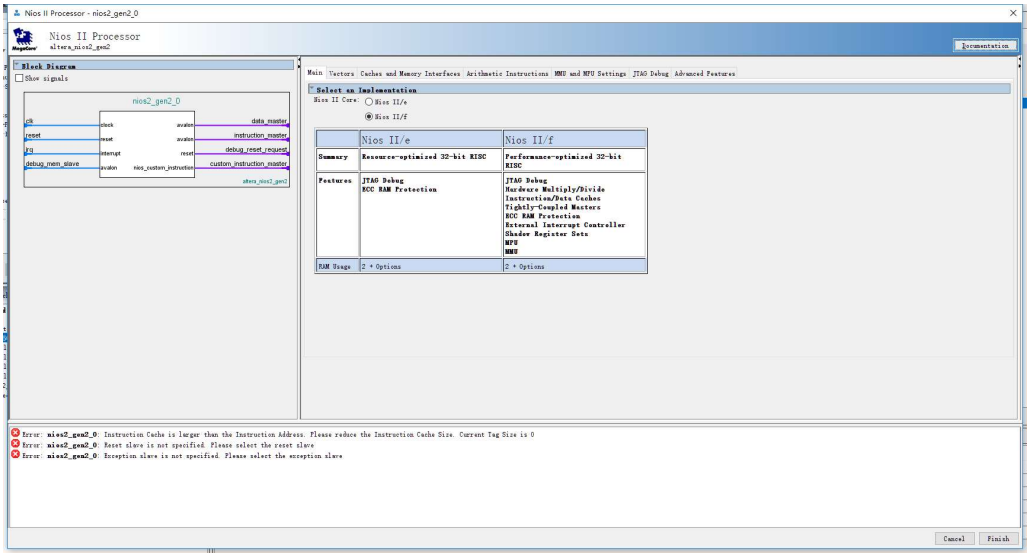


2.3.2 添加 CPU

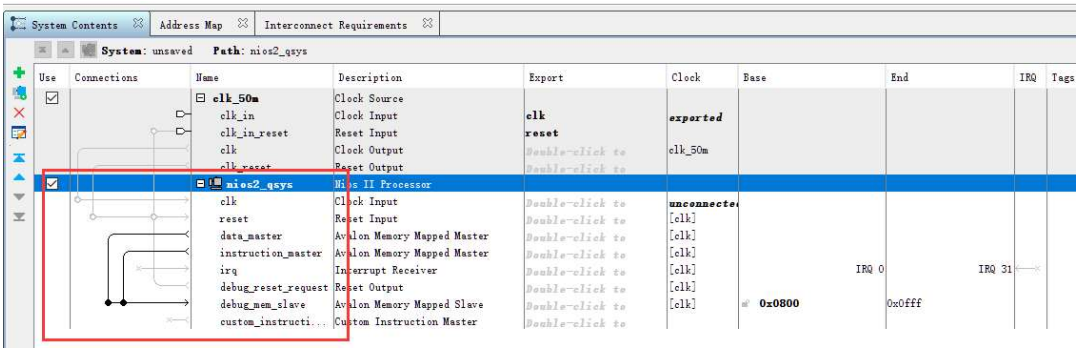
首先要找到 CPU 这个 IP 核，在 Altera 之中，当然就是 Nios II，在左上角这个搜索框中，搜索“NIOS”，找到下图中这个 IP：



找到这个 IP 核后，双击，弹出下面的对话框。

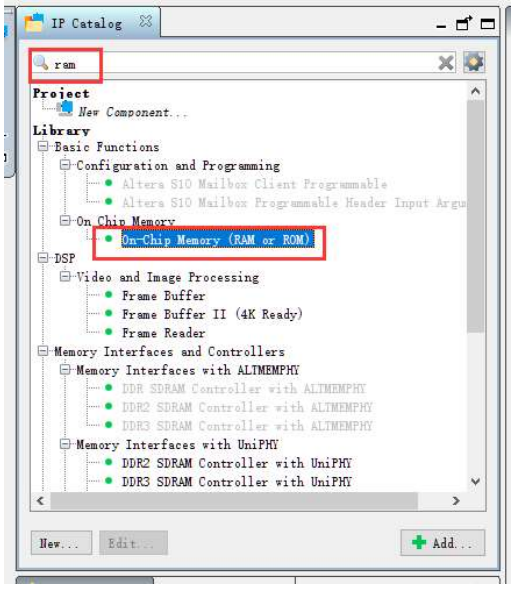


NIOSII-E/F 是指基础版本和高性能版本，选择 E 版本（基础版本）就可以。选择好型号之后，暂时不用更改其它设置，点击 Finish 添加并修改名字。添加完毕后如下所示：

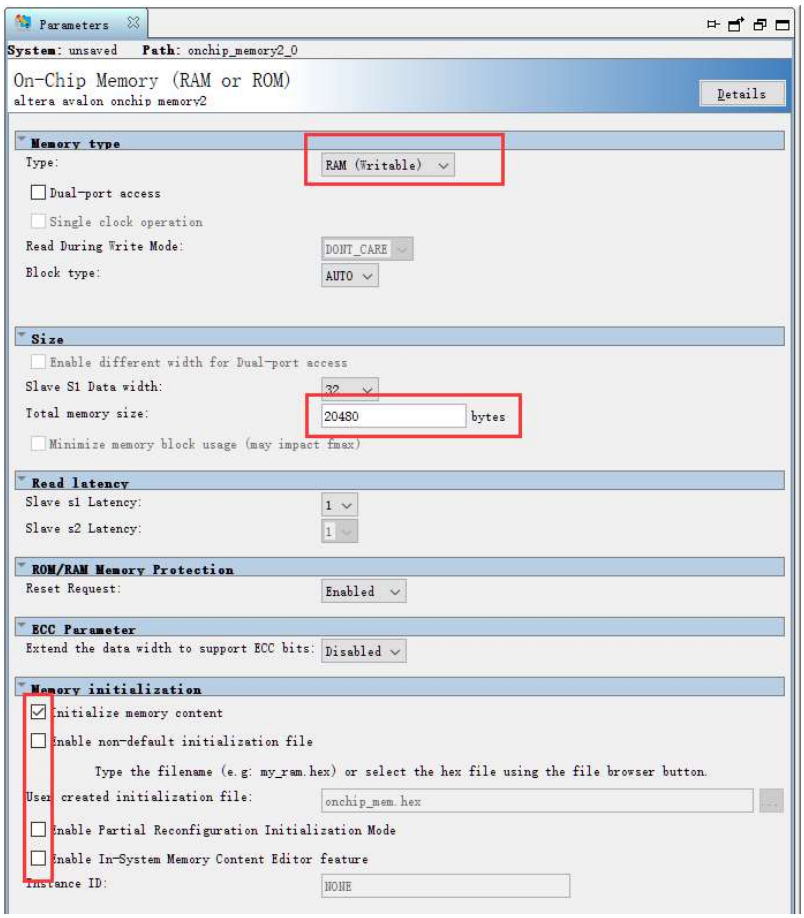


2.3.3 添加 RAM

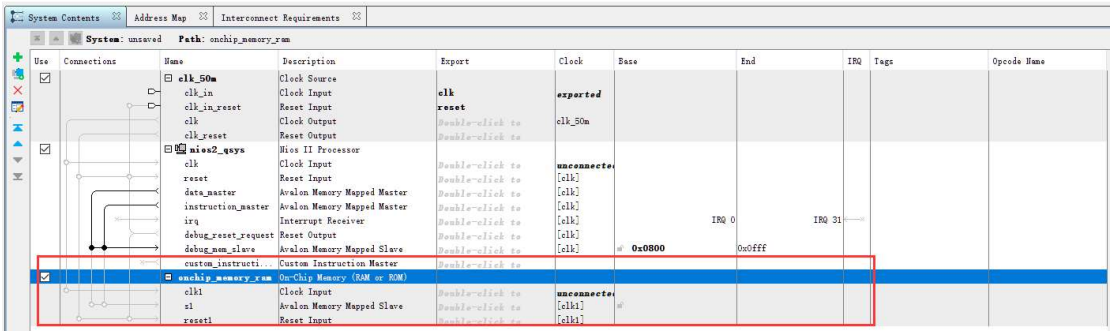
同样，在左侧搜索框内搜索 RAM，找到 On-Chip Memory(RAM or ROM)，双击。进入如下界面。



种类选为 RAM，大小这里填写 20480，初始化选择第一个即可。之后，更改 RAM 模块名字。

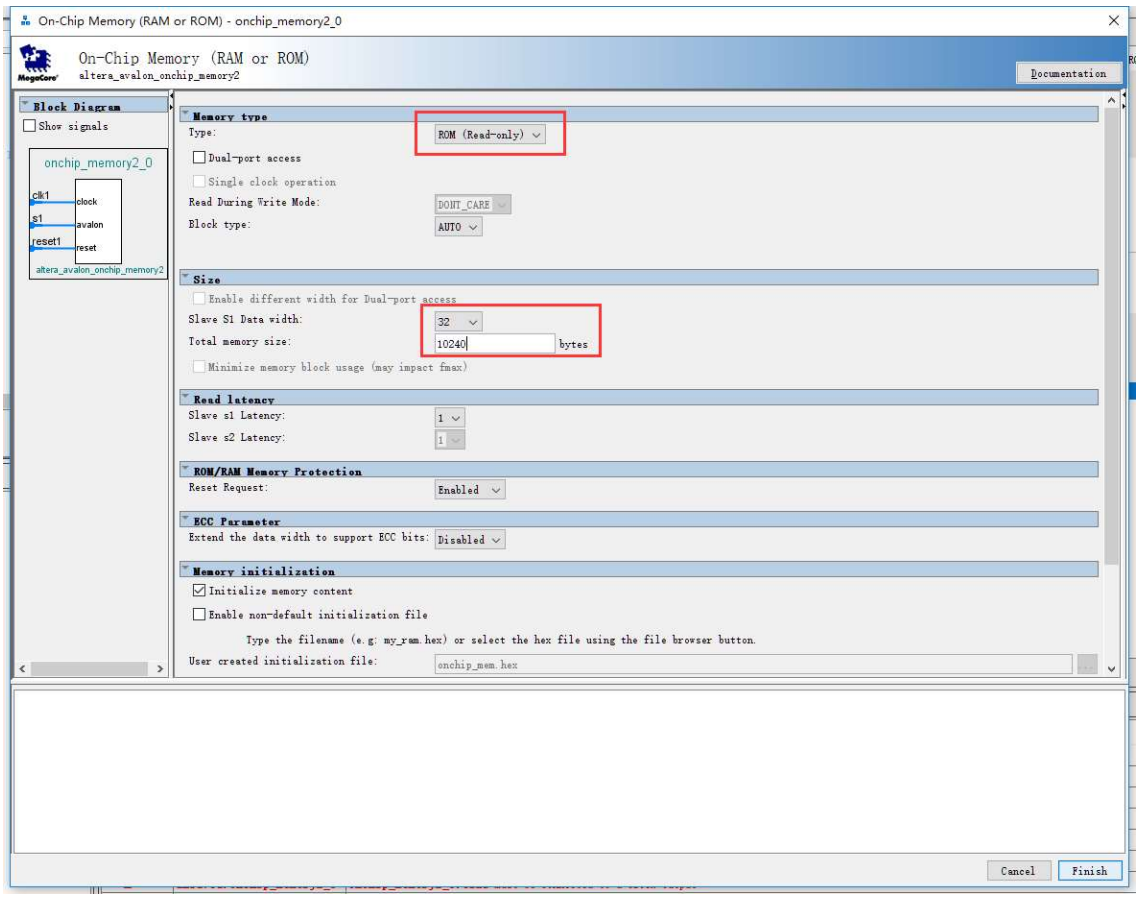


更改完毕如下：



2.3.4 添加 ROM

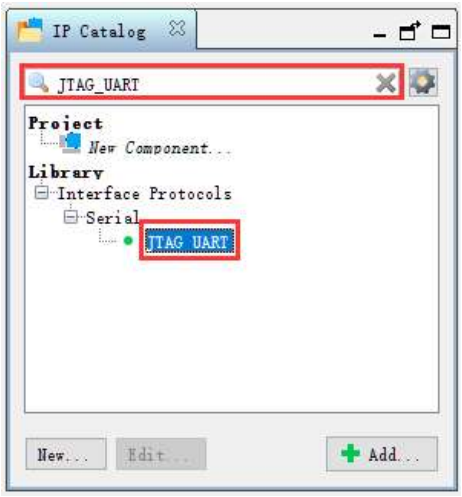
与添加 RAM 类似，添加 ROM，也是添加同一个 IP。Memory Type 选择为"ROM"，Totle memory size 输入为“10240”设置如下：

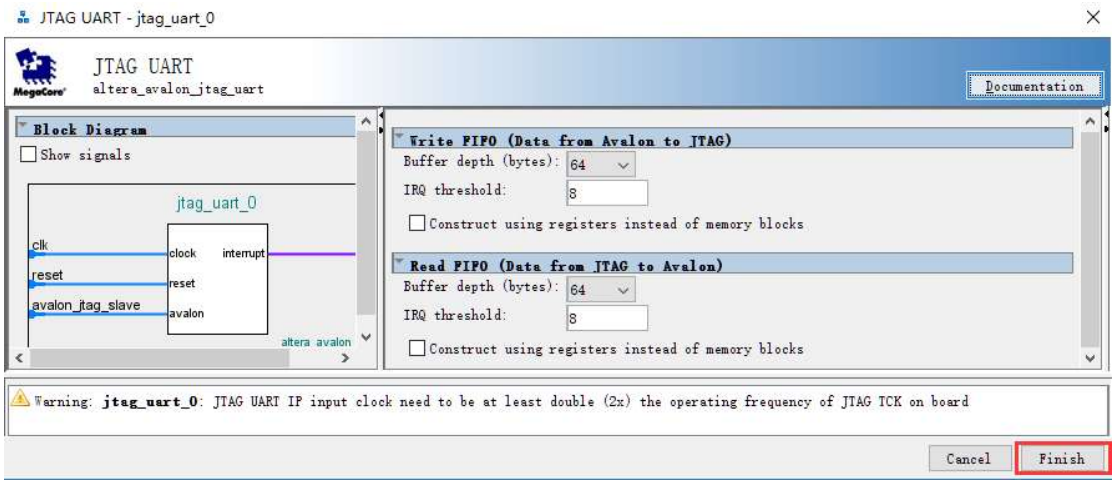


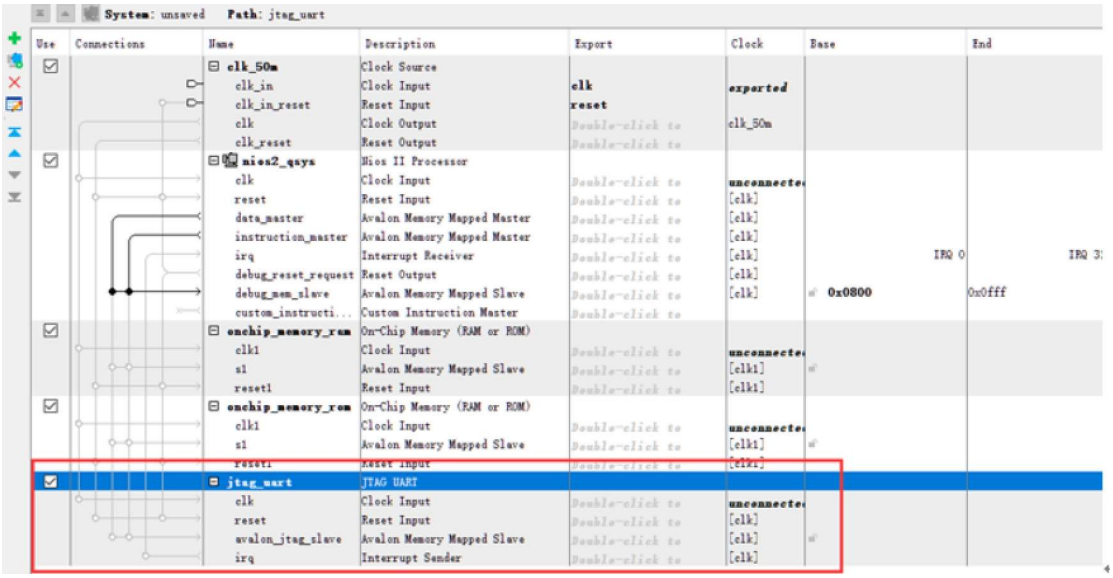
点击 Finish，完成添加，再次修改名字，下图为修改完毕图片：

2.3.5 添加 JTAG_UART

添加 JTEG_UART。搜索如下 IP，并双击进入，我们无需修改 JTAG_UART IP 核的参数，直接点击 FINISH：



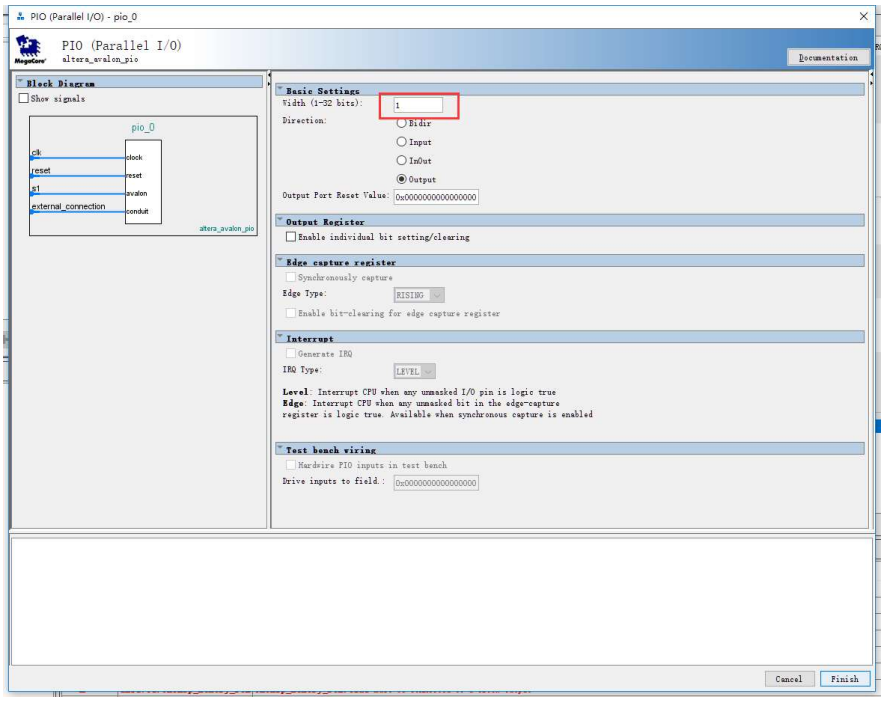




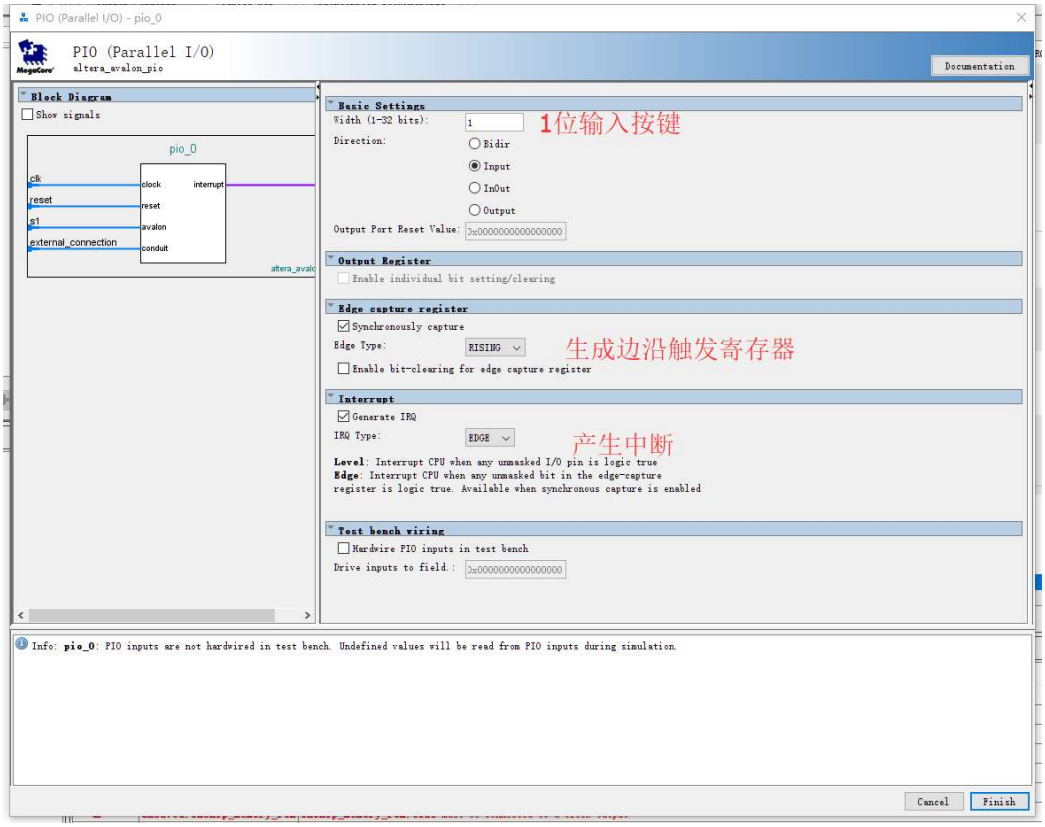
2.3.6 添加 PIO

添加 PIO。搜索 IP:PIO(Parallel I/O)，并双击进入，首先我们设置一个输出管脚 LED，

位宽设置成 8 位，如下图所示：



之后，设置一个输入按键，带有中断，该案件可作为系统复位按键，如下图进行设置：



生成完毕，如下图所示：


System: unsaved
Path: pio_led
Use
Connections
Name
Description
Export
Clock
Base
End
IRQ
Tags
Opcode Name
clk_reset
Reset Output
nios2_qsys
Nios II Processor
clk
Clock Input
reset
Reset Input
data_master
Avalon Memory Mapped Master
instruction_master
Avalon Memory Mapped Master
irq
Interrupt Receiver
IRQ 0
IRQ 31
debug_reset_request
Reset Output
debug_mem_slave
Avalon Memory Mapped Slave
0x0800
0x0fff
custom_instructi...
Custom Instruction Master
onchip_memory_ram
On-Chip Memory (RAM or ROM)
clk1
Clock Input
s1
Avalon Memory Mapped Slave
reset1
Reset Input
onchip_memory_rom
On-Chip Memory (RAM or ROM)
clk1
Clock Input
s1
Avalon Memory Mapped Slave
reset1
Reset Input
jtag_uart
JTAG UART
clk
Clock Input
reset
Reset Input
avalon_jtag_slave
Avalon Memory Mapped Slave
pio_led
PIO (Parallel I/O)
clk
Clock Input
reset
Reset Input
s1
Avalon Memory Mapped Slave
external_connection
Conduit
pio_key
PIO (Parallel I/O)
clk
Clock Input
reset
Reset Input
s1
Avalon Memory Mapped Slave
external_connection
Conduit

至此，外设添加完毕。

3. 配置外设

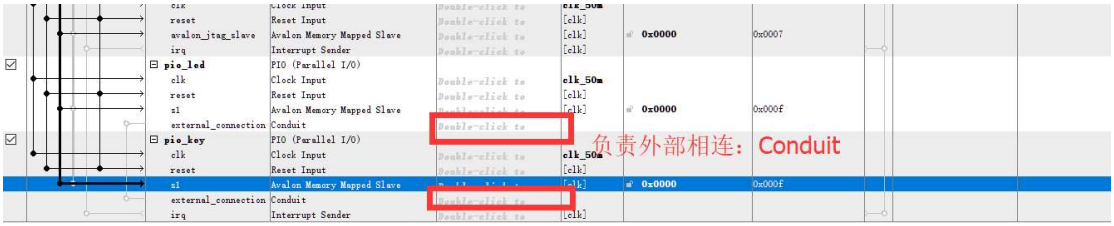
3.1 外设互联

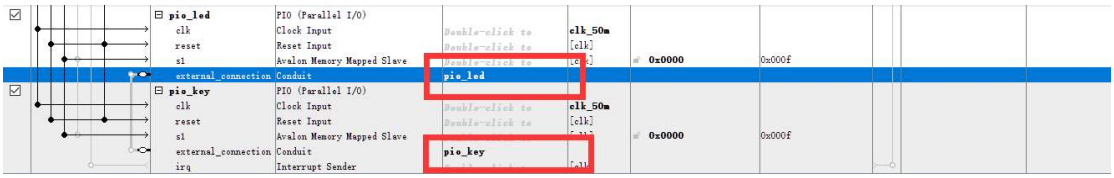
总线互联通用方式：

· 连接每个 IP 相关的时钟信号与复位信号；

· data_master（数据主端口）与 instruction_maste（命定主端口）的处理：如果 IP 模块为 Memory 类型的，这两个信号都链接上，如果不是，只连接数据线即可。

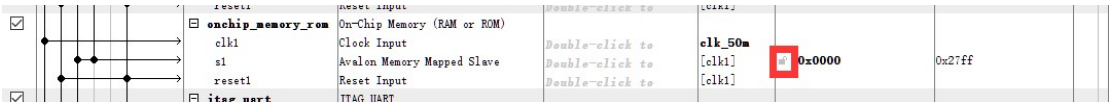
· 配置外部数据端口：双击想要链接外部的信号，找到对应外设的 Conduit 列，点击红框位置，输入信号名。

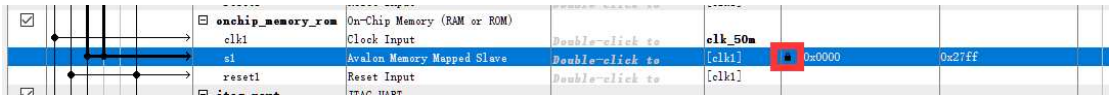




3.2 地址分配

首先，地址分配的最关键一点是将 ROM 的地址锁定：

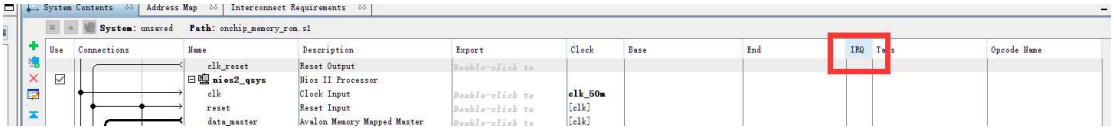




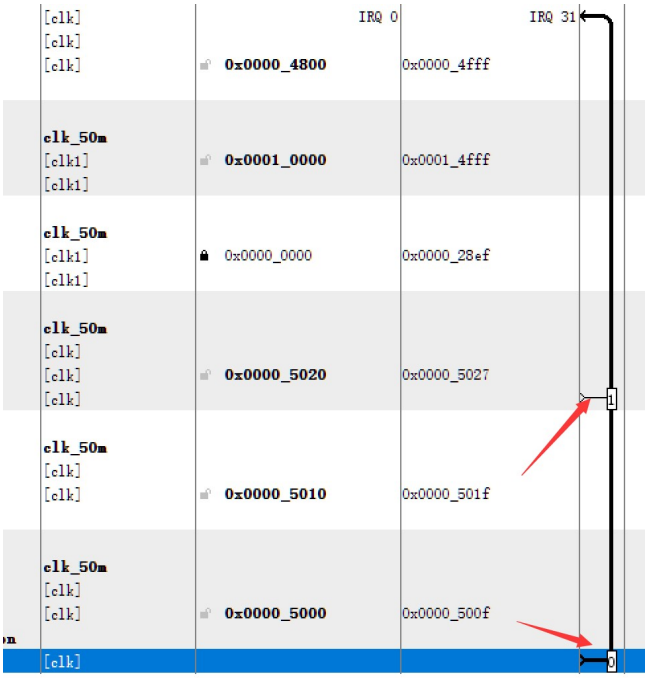
之后，点击 System-Assign Base Address,该工具会将所有外设的基地址分配完毕。

3.3 配置中断

点击”IRQ”列，在该列进行中断配置。

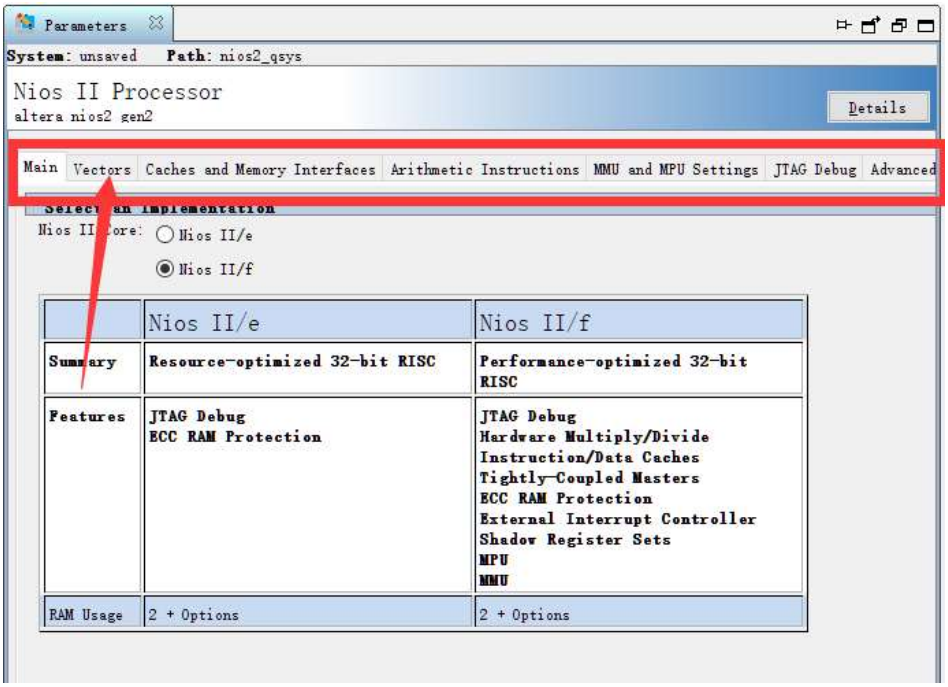


配置完毕如下图所示：

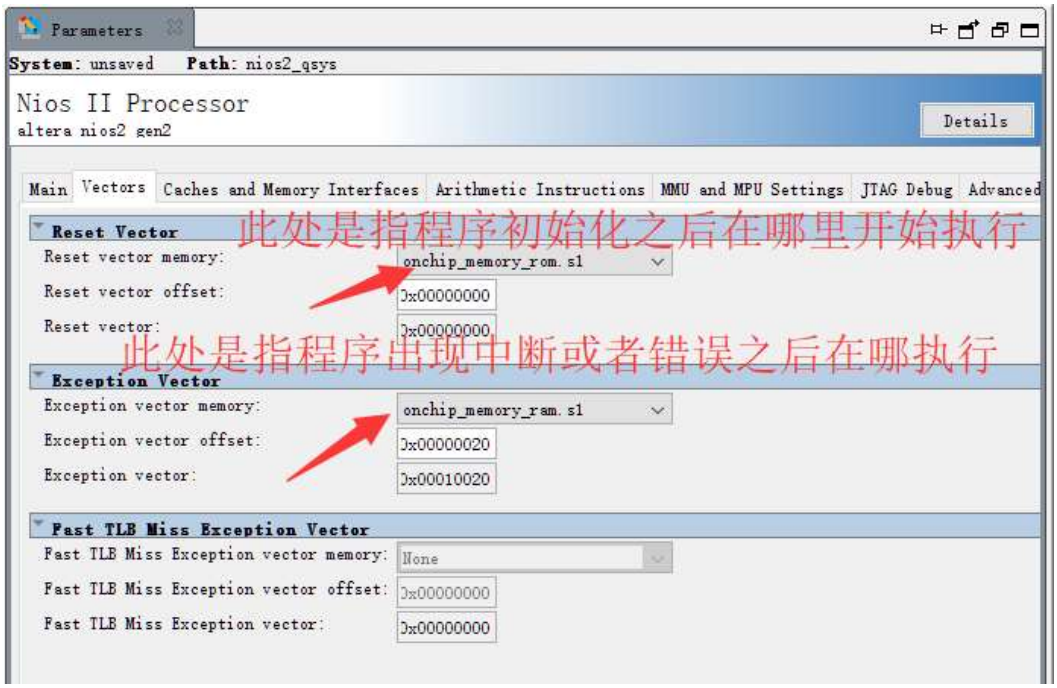


3.4 配置 Nios II 启动设置

双击 Nios II IP，进入配置界面：

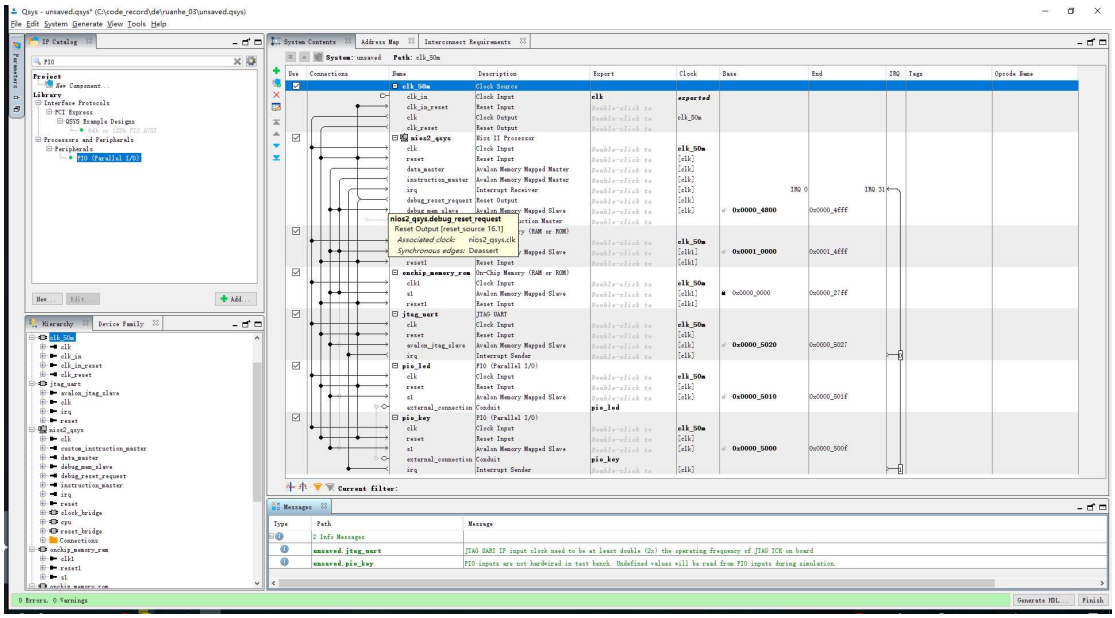


进入该子菜单，将 Reset Vector 选为之前配置的 Rom，将 Exception Vector 选为之前配置的 ram。



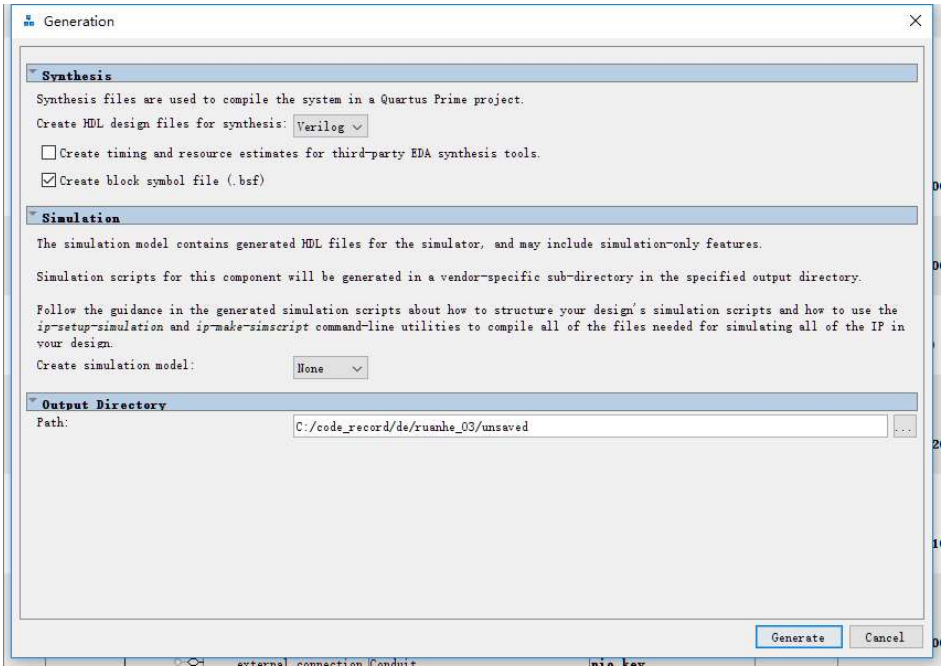
3.5 配置完毕

配置完毕如下图所示：

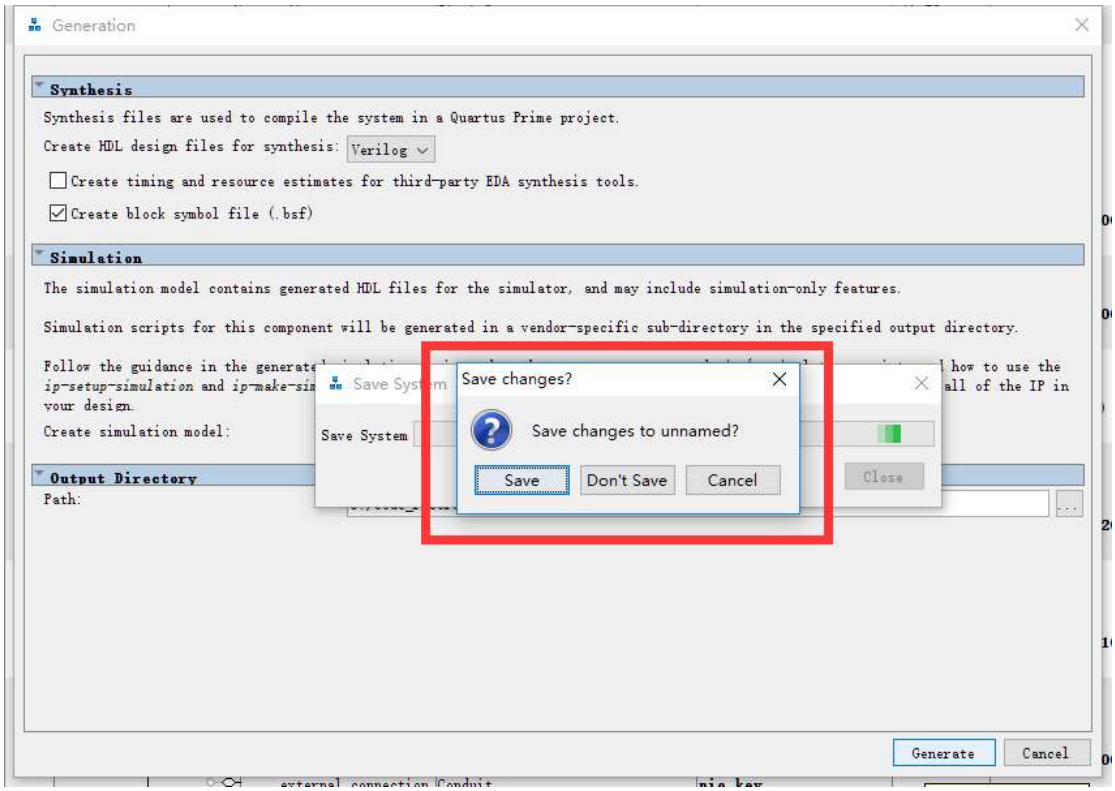


4. 生成 Nios II 软核处理器

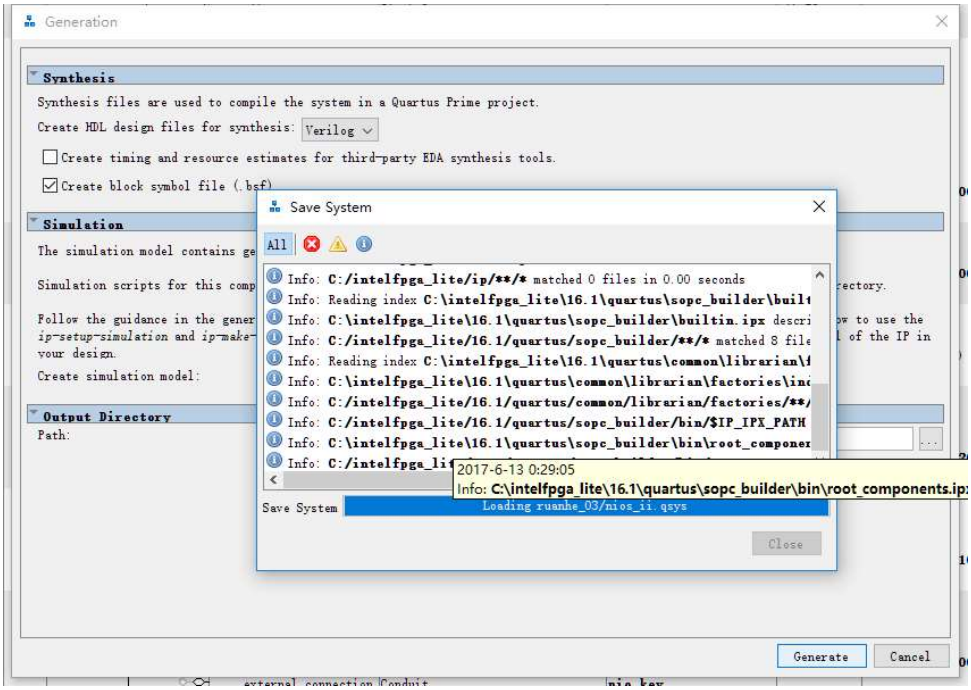
1. 点击 Generate-HDL，弹出如下对话框，点击生成：



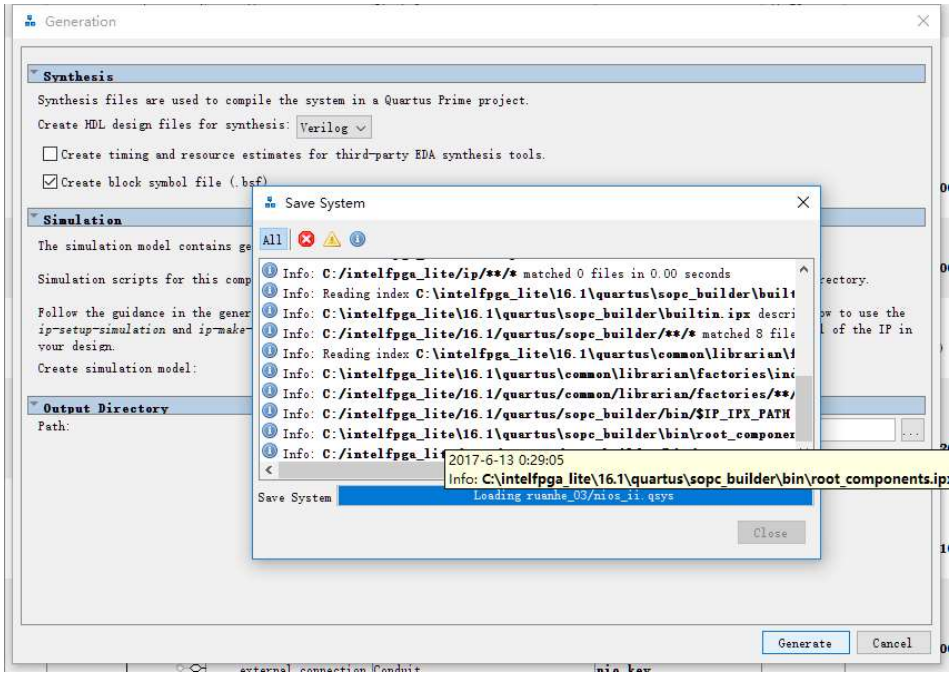
2. 保存工程



3. 生成中，稍等片刻。



4. 生成完成。



4.点击 Close,完成生成。

5. 认识 Qsys 为你生成的文件

进入工程目录，查阅 Qsys 为你生成的文件：

文件后缀与其含义如下所示：

.sopcinfo 文件：

包含了所有 Qsys 项目的描述，重要文件。ECSLIPES 关联会用到。

.qsys 文件:

包含了刚才所调用的所有 IP 核，也包含了核与核之间的连接与参数。

.bsf 文件:

模块标识文件

.html 文件:

网页版报表文件

.qip 文件:

包含了所有 IP 核的文件，需要被添加进工程中。

.v 文件:

Verilog 文件，此处为生成的例化 AlteraIP 核的文件

6. 整合工程

6.1 添加 Verilog 文件

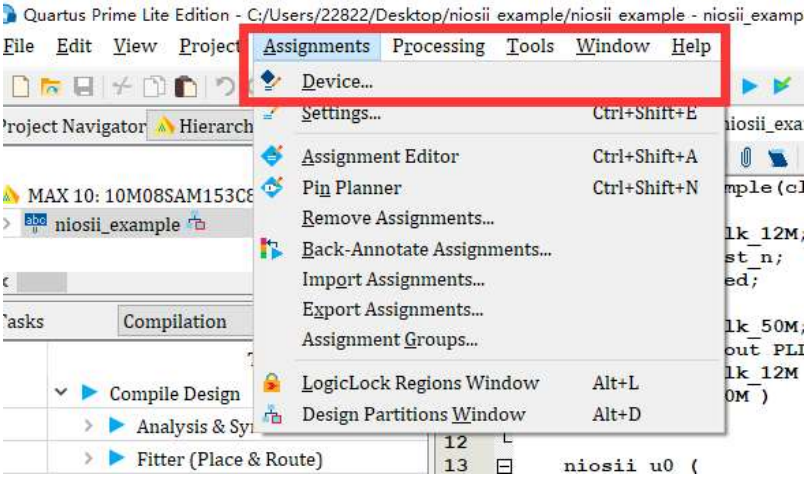
在 Quartus Prime 界面，新建 Verilog 文件，输入如下代码，保存：

```
    module niosii_example(clk_12M,rst_n,led);

        input       clk_12M;
        input       rst_n;
        output [7:0] led;

        wire            clk_50M;
         PLL_12Min_50Mout   PLL_12Min_50Mout_inst (
            .inclk0 ( clk_12M ),
            .c0 ( clk_50M )
        );

         niosii u0 (
            .clk_clk                            (clk_50M),
//                          clk.clk
            .pio_button_external_connection_export (rst_n), //
pio_button_external_connection.export
            .pio_led_external_connection_export    (led)     //
pio_led_external_connection.export
        );

    endmodule
```

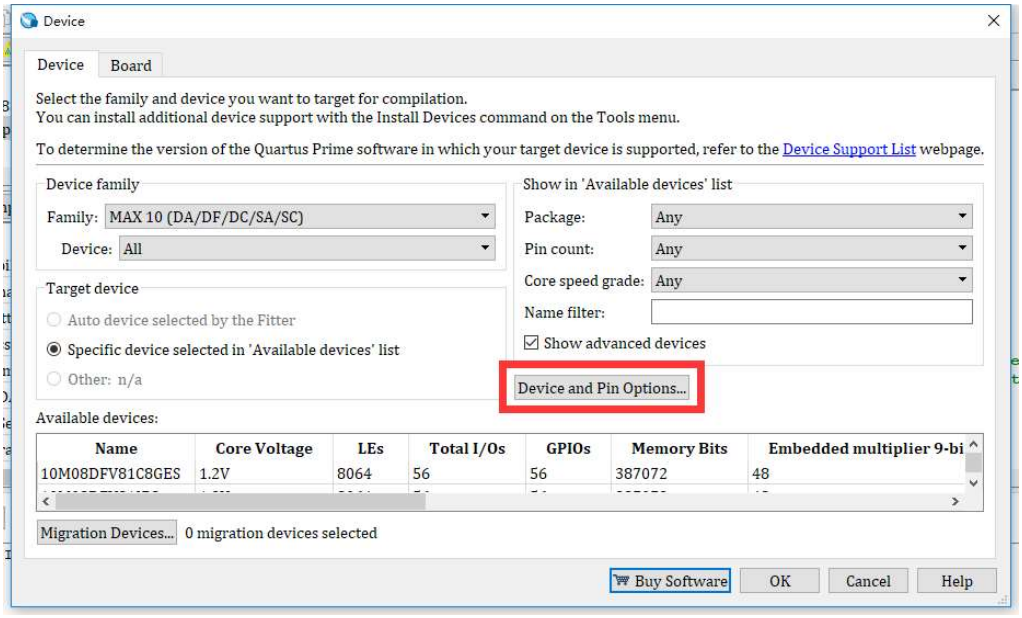
6.2 更改配置模式(Configuration Mode)

对于 MAX10 器件来说，由于其配置模式有多种，若需要使用 ROM IP 进行初始化或搭建 Nios II 软核时，我们需要将配置模式改为 **“Single Compressed Image with Memory Initialization”**。具体操作步骤如下：

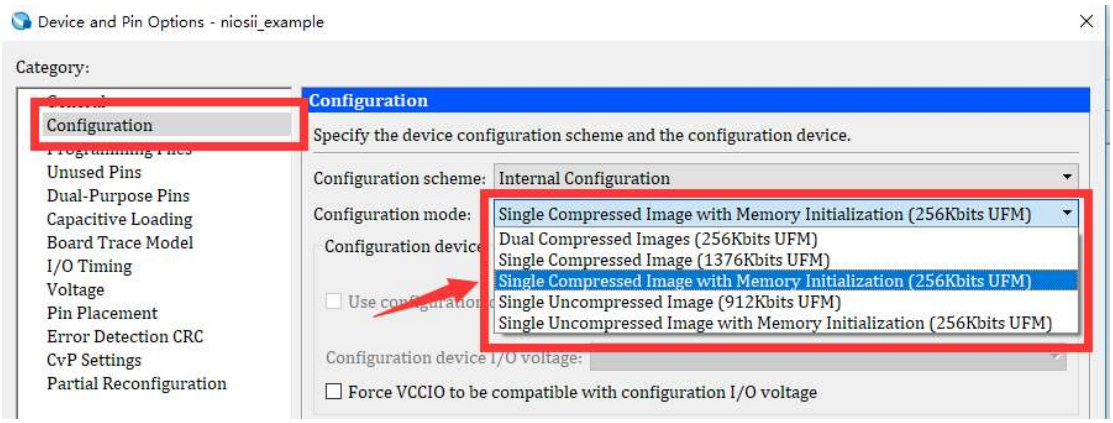
首先，点击菜单栏中的 Assignments 下的 Device，进入 Device 设置页面：



进入 Device 设置页面，点击“Device and Pin Options”，进行进一步的设置；



在左侧“Category”栏中，找到“Configuration”，单击，将 Configuration mode 改为 Single Compressed Image with Memory Initialization() 。

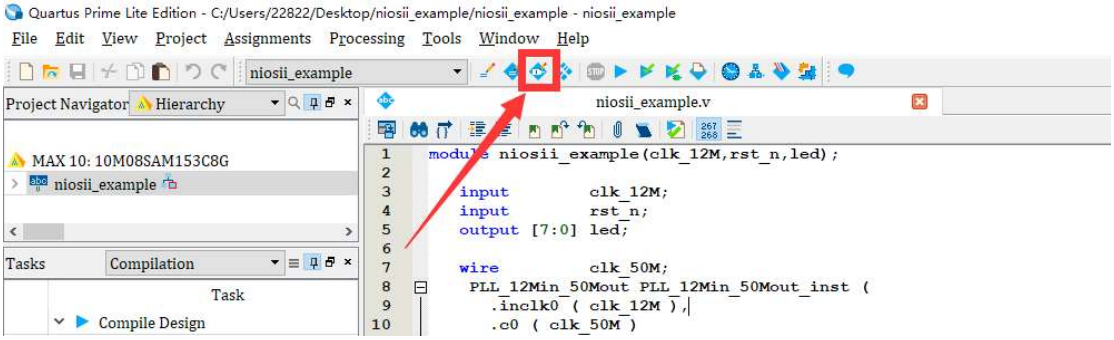


6.3 综合工程

完成上述配置后即可综合工程。

6.4 分配管脚

点击分配管脚按钮，进行管脚分配。

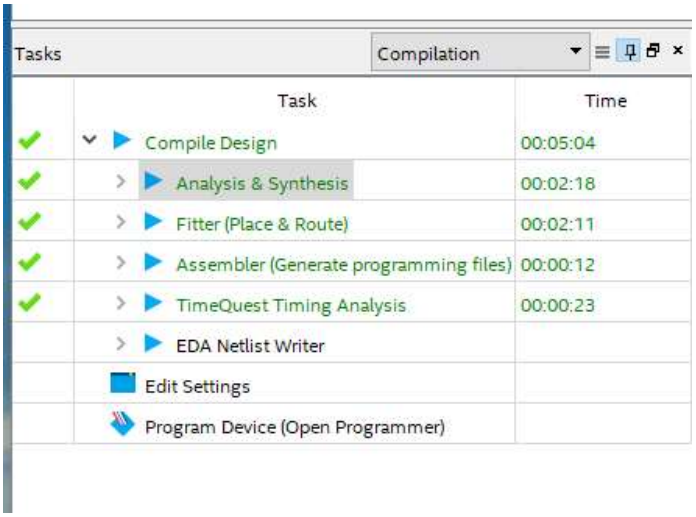


管脚分配如下所示：

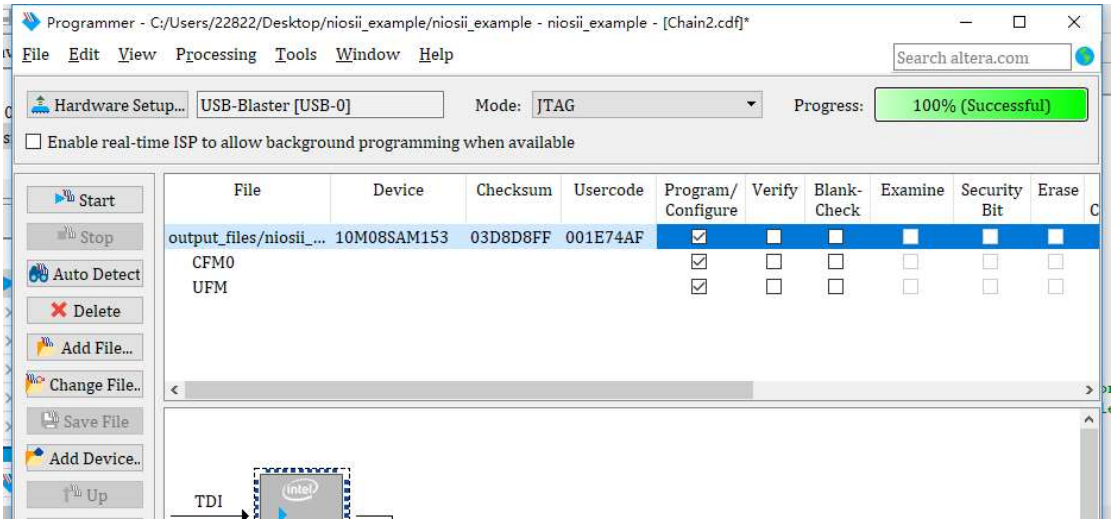
Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strengt	Slew Rate
altera_r...rved_tck	Input				PIN_J1	2.5 V (default)		12mA (...fault)	
altera_reserved_tdi	Input				PIN_H5	2.5 V (default)		12mA (...fault)	
altera_r...rved_tdo	Output				PIN_H4	2.5 V (default)		12mA (...fault)	2 (default)
altera_r...rved_tms	Input				PIN_G1	2.5 V (default)		12mA (...fault)	
clk_12M	Input	PIN_J5	2	B2_N0	PIN_J5	2.5 V		12mA (...fault)	
led[7]	Output	PIN_K11	5	B5_N0	PIN_K11	2.5 V		12mA (...fault)	2 (default)
led[6]	Output	PIN_L11	5	B5_N0	PIN_L11	2.5 V		12mA (...fault)	2 (default)
led[5]	Output	PIN_K12	5	B5_N0	PIN_K12	2.5 V		12mA (...fault)	2 (default)
led[4]	Output	PIN_L15	5	B5_N0	PIN_L15	2.5 V		12mA (...fault)	2 (default)
led[3]	Output	PIN_M12	5	B5_N0	PIN_M12	2.5 V		12mA (...fault)	2 (default)
led[2]	Output	PIN_M14	5	B5_N0	PIN_M14	2.5 V		12mA (...fault)	2 (default)
led[1]	Output	PIN_N14	5	B5_N0	PIN_N14	2.5 V		12mA (...fault)	2 (default)
led[0]	Output	PIN_N15	5	B5_N0	PIN_N15	2.5 V		12mA (...fault)	2 (default)
rst_n	Input	PIN_J9	5	B5_N0	PIN_J9	2.5 V		12mA (...fault)	
button	Unknown	PIN_H11	6	B6_N0		2.5 V (default)		12mA (...fault)	

6.5 编译文件并下载

双击 Compile Design 编译工程，编译完成如下：



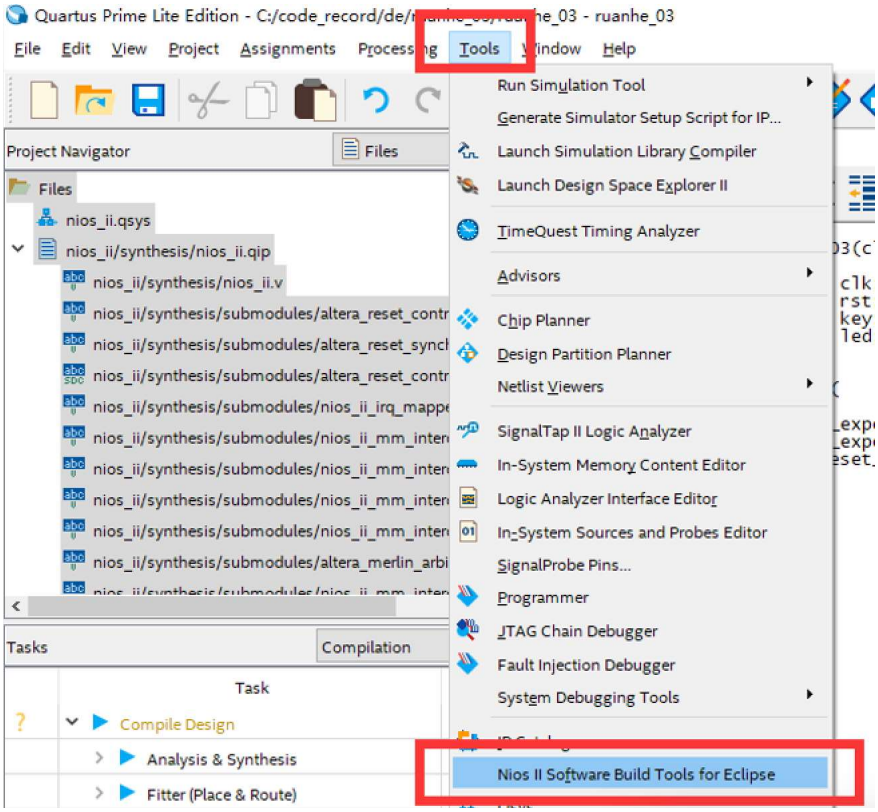
打开 Programmer，进行下载，下载完成如下所示。



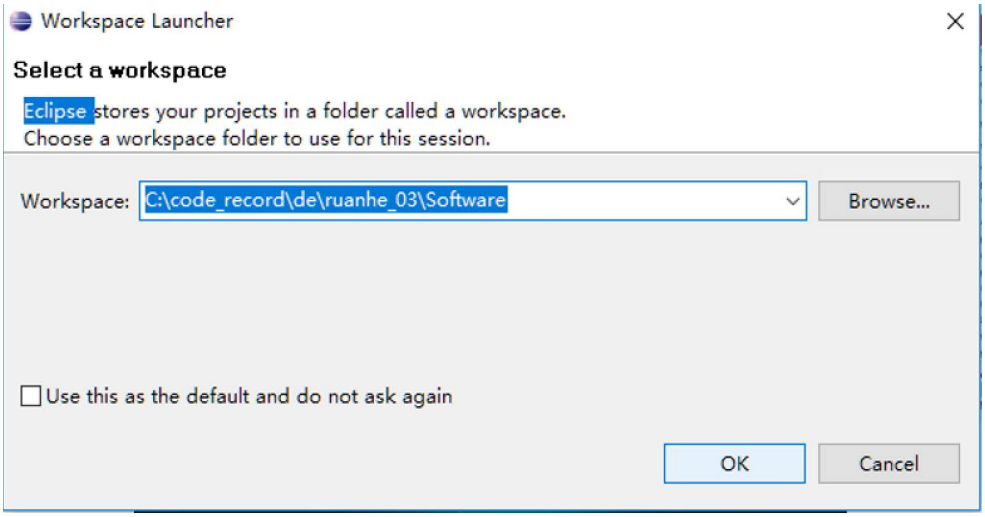
7. Eclipse

7.1 打开 Eclipse

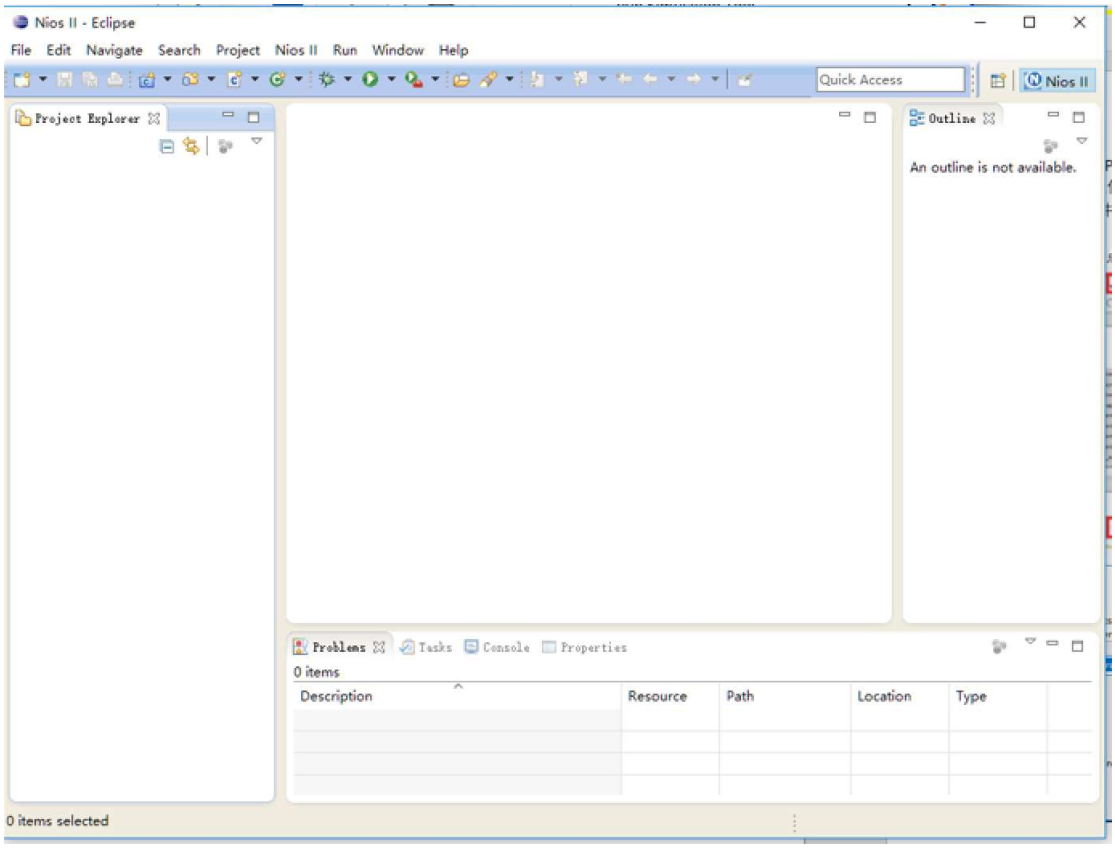
首先在 Quartus 界面点击 Tools-Nios II Software Build Tools Eclipse 进入 Eclipse。



进入软件之后，软件要求你选择一个工作空间，这个工作空间的意思是指你需要指定一个文件夹，这个文件夹会储存你关于你这个 Nios II 系统相关的软件设计方面的部分。若选择完毕，点击 OK。

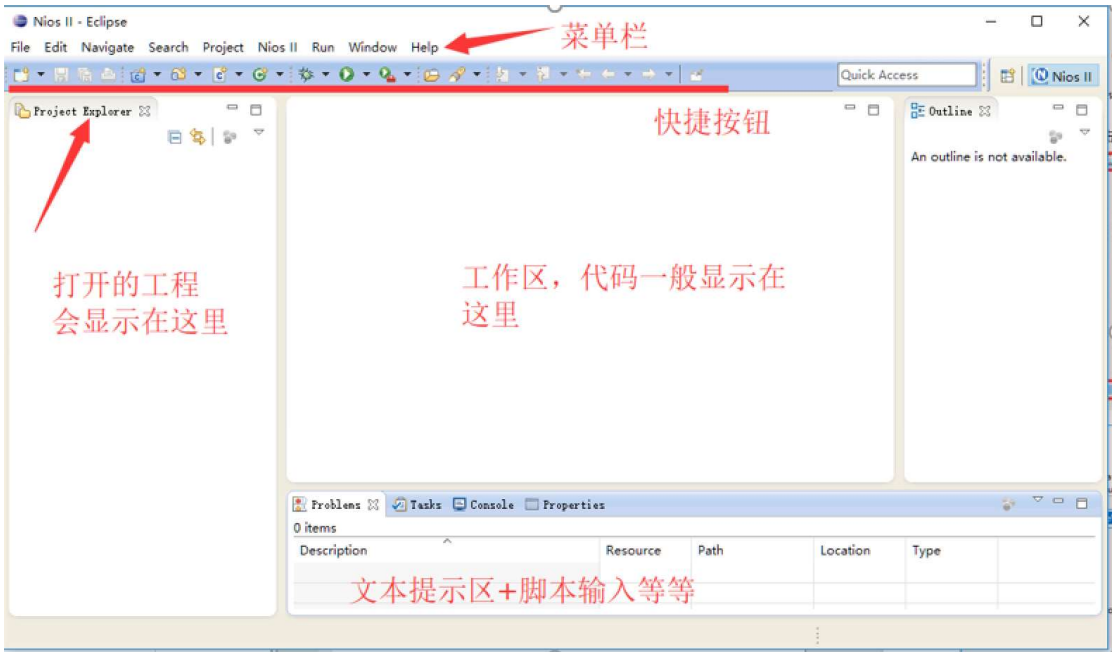


进入 Eclipse 如下所示：



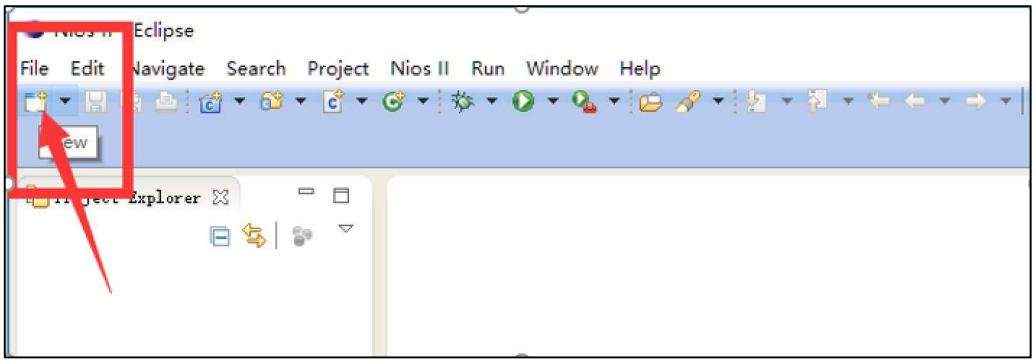
7.2 软件界面

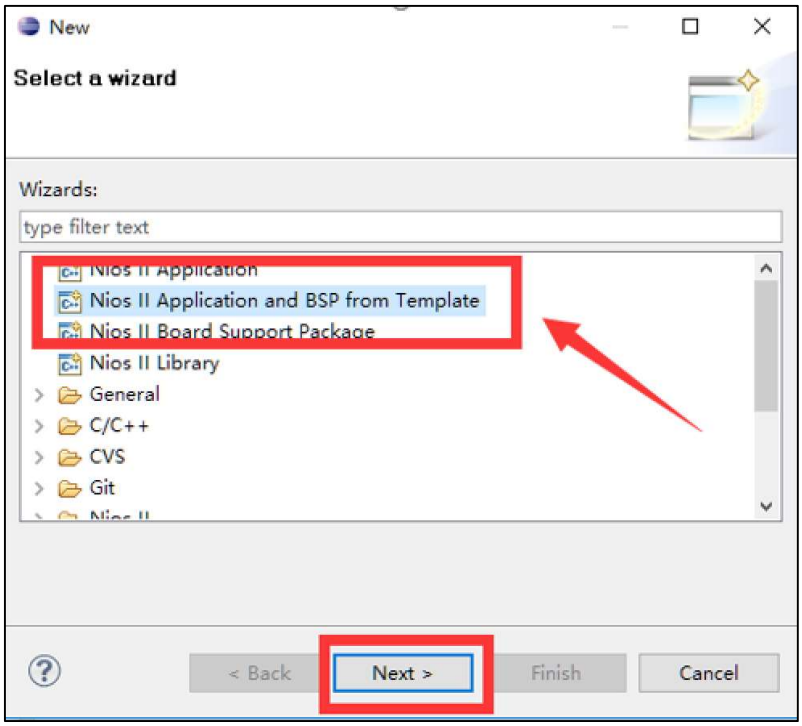
软件界面如下所示，若误删除功能框，迪阿尼 Window-Reset Perspective 即可恢复。



7.3 建立新的工程:板级支持包

点击新建按钮，也可以 File-New，选择 Nios II Application and BSP from Template 。

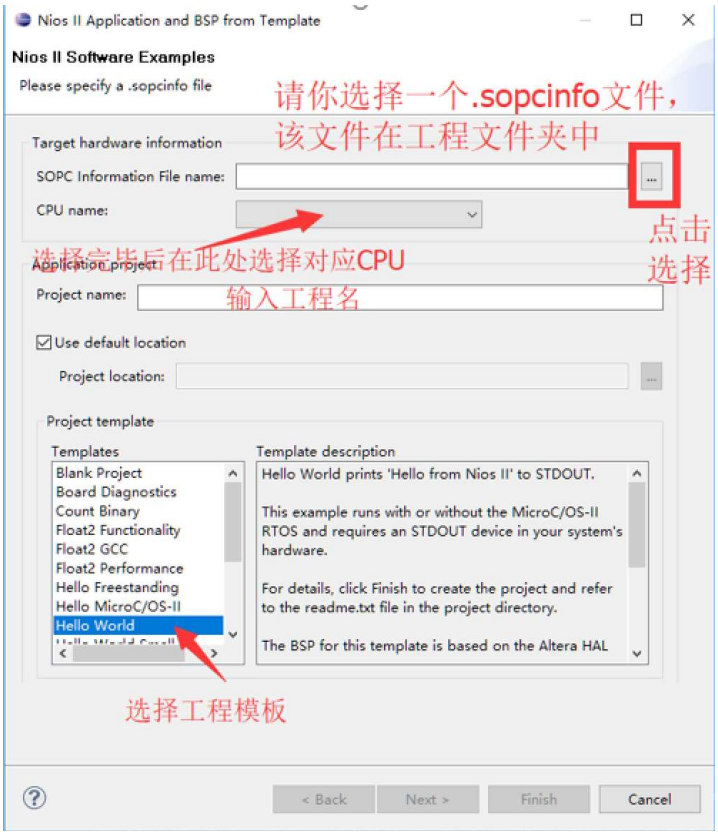


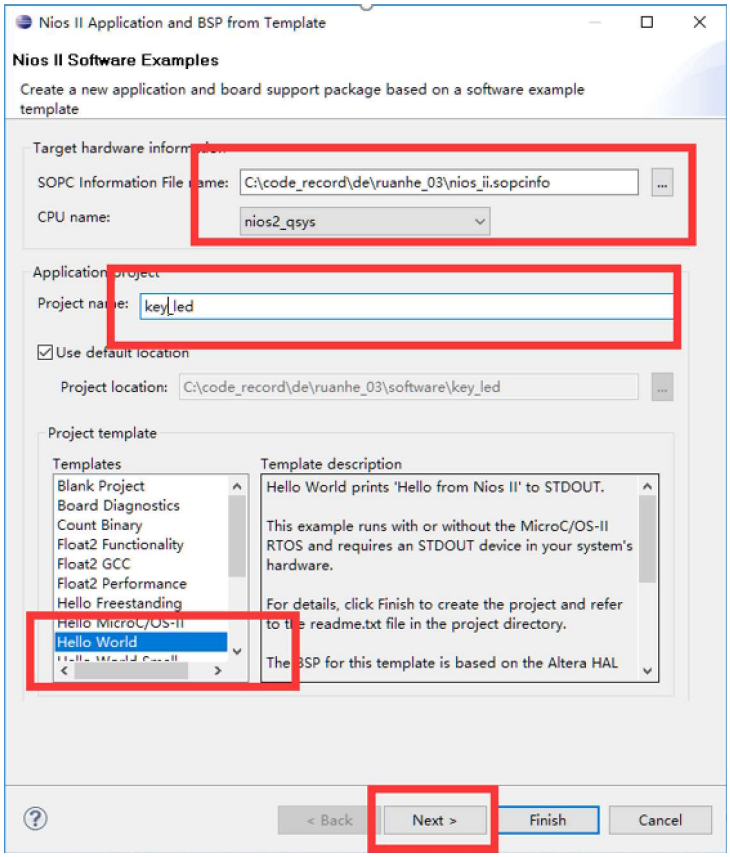


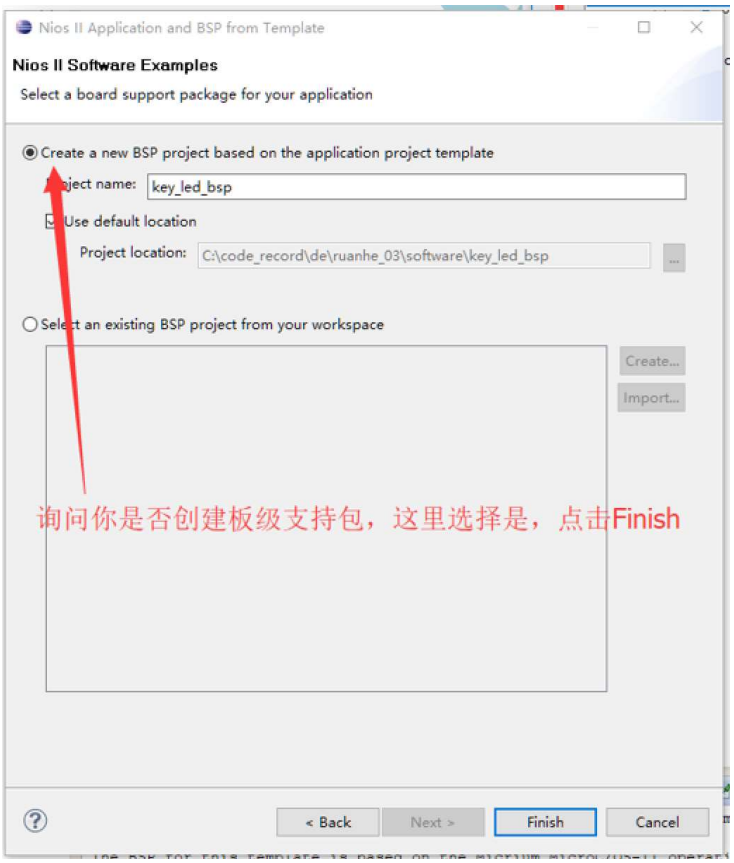
建立板级支持包的意思是我们从一个模板中创建一个工程以大大减少我们的开发周期与开发难度。并且可以直接利用其应用程序进行开发。

选择好了之后会弹出一个对话框，如图操作即可，选择完毕后点击“Next”。

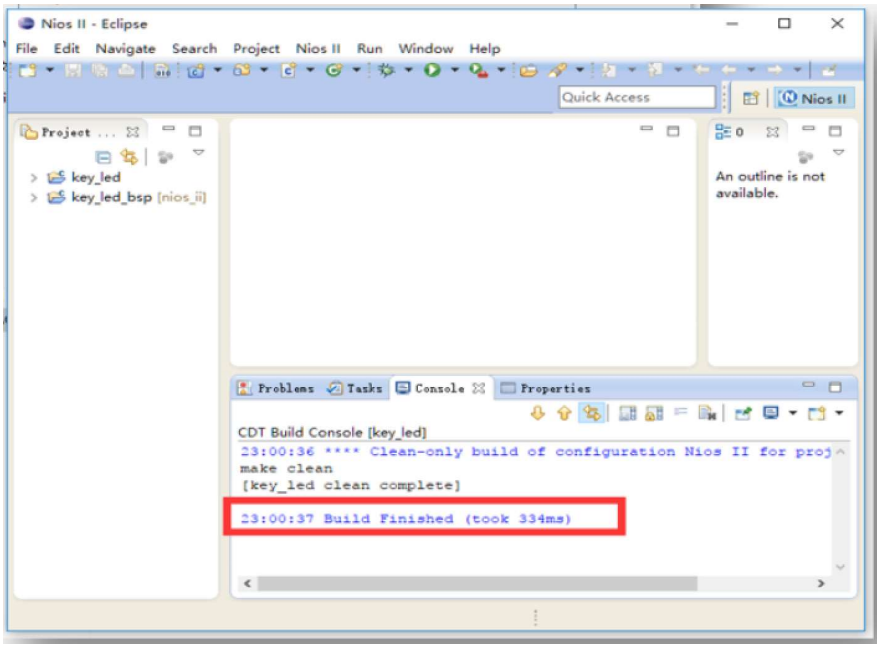
- 在 SOPC Information File Name 栏选择 Nios II 软核信息文件(文件后缀为.sopcinfo)；
- 输入 Project Name；
- Templates 选择为“Hello World ”。
- 点击 Next；
- 选中“Create a new BSP project.....”
- 点击 Finish。


Nios II Application and BSP from Template
Nios II Software Examples
Please specify a .sopcinfo file
请你选择一个.sopcinfo文件，
该文件在工程文件夹中
Target hardware information
SOPC Information File name:
CPU name:
点击
选择
选择完毕后在此处选择对应CPU
Application project
Project name:
输入工程名
Use default location
Project location:
Project template
Templates
Template description
Blank Project
Board Diagnostics
Count Binary
Float2 Functionality
Float2 GCC
Float2 Performance
Hello Freestanding
Hello MicroC/OS-II
Hello World
Hello World prints 'Hello from Nios II' to STDOUT.
This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware.
For details, click Finish to create the project and refer to the readme.txt file in the project directory.
The BSP for this template is based on the Altera HAL
选择工程模板
< Back
Next >
Finish
Cancel

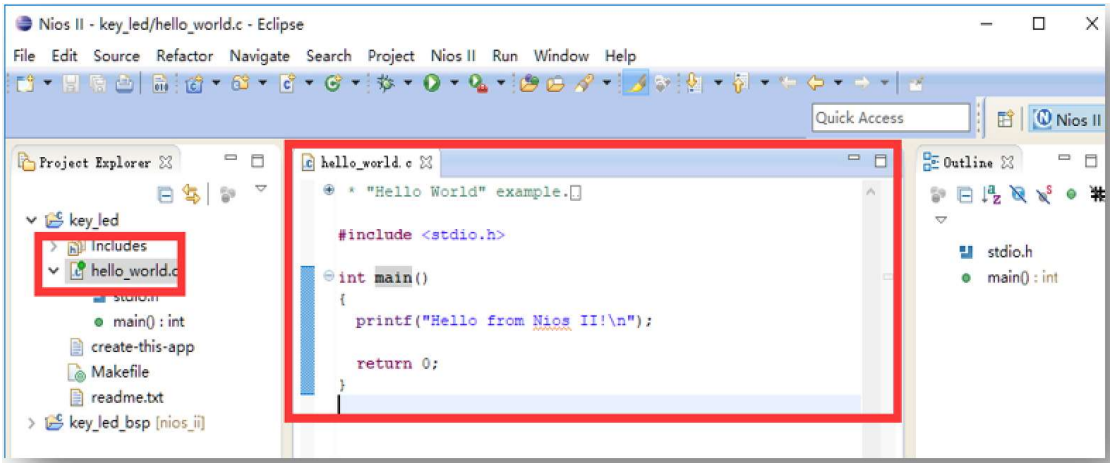

Nios II Application and BSP from Template
Nios II Software Examples
Create a new application and board support package based on a software example template
SOPC Information File name: C:\code_record\de\ruanhe_03\nios_ii.sopcinfo
CPU name: nios2_qsys
Project name: key_led
Use default location
Project location: C:\code_record\de\ruanhe_03\software\key_led
Project template
Templates
Template description
Blank Project
Board Diagnostics
Count Binary
Float2 Functionality
Float2 GCC
Float2 Performance
Hello Freestanding
Hello MicroC/OS-II
Hello World
Hello World prints 'Hello from Nios II' to STDOUT.
This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware.
For details, click Finish to create the project and refer to the readme.txt file in the project directory.
< Back
Next >
Finish
Cancel



建立完毕如下图所示。



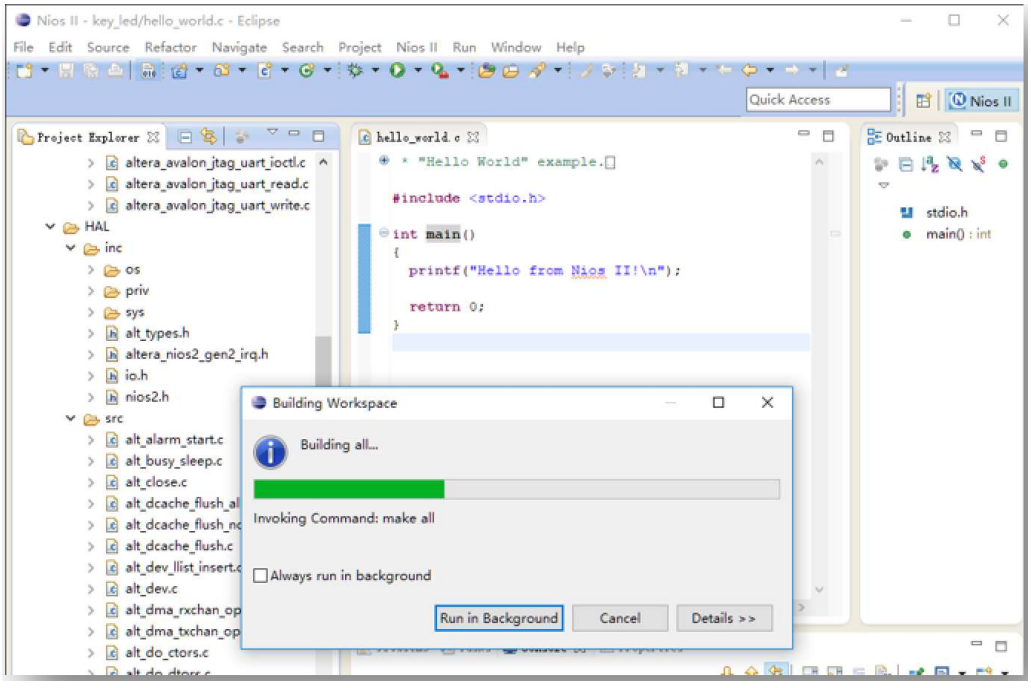
打开“key_led”，你可以发其中有一个.c 文件，打开它，里面就是模板内供你测试的程序。



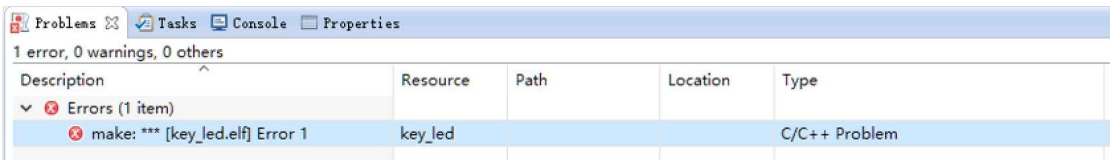
然后，我们再打开 bsp，里面有很多很多的文件:这里面包含了硬件驱动、硬件头文件、系统关键信息、输入输出、外围器件、设备地址等等信息。给你的硬件以最完善的支持，免除你写底层驱动的烦恼，很容易你就可以操作硬件。

7.4 编译工程:如何压缩板级支持包？

既然运行的是 C 程序，当然我们得对它进行编译，点击快捷键栏的全编译按钮，进行编译：

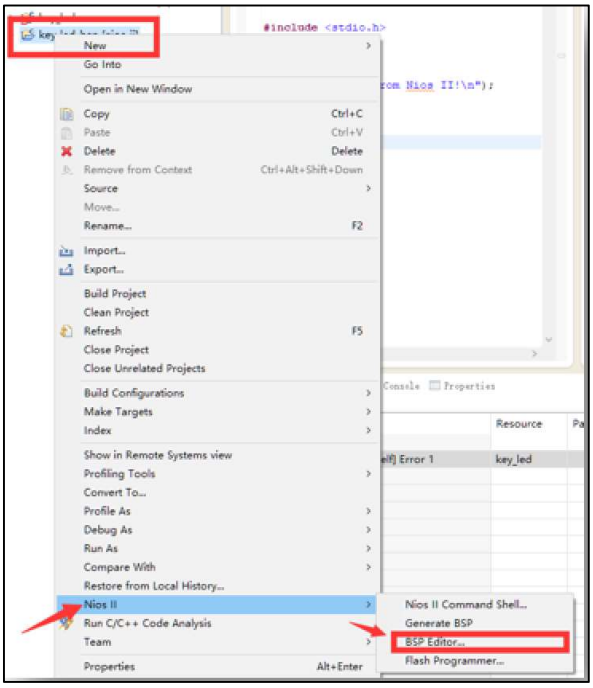


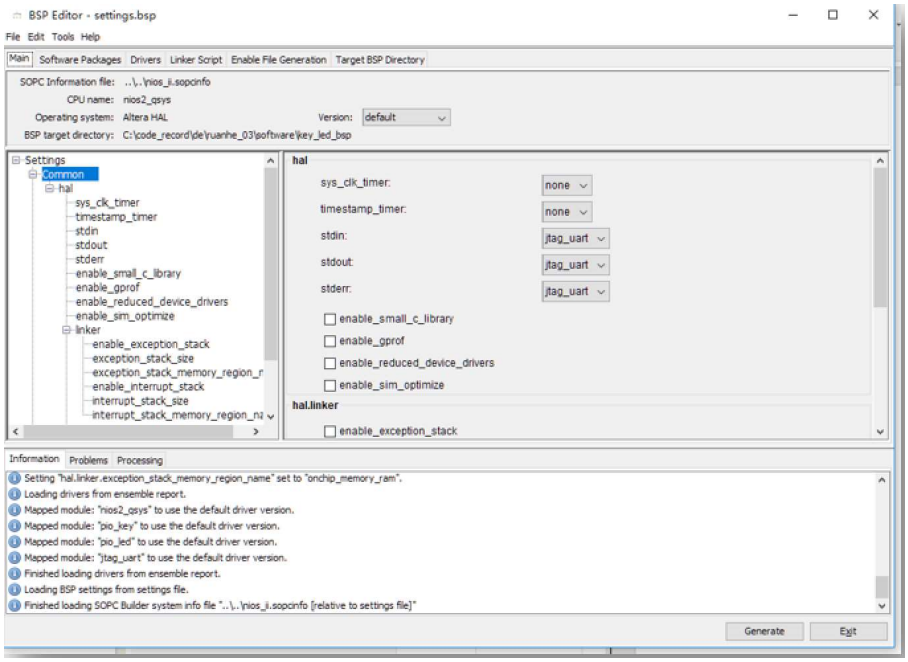
这里我们在 Problems 栏里面发现了一个错误：



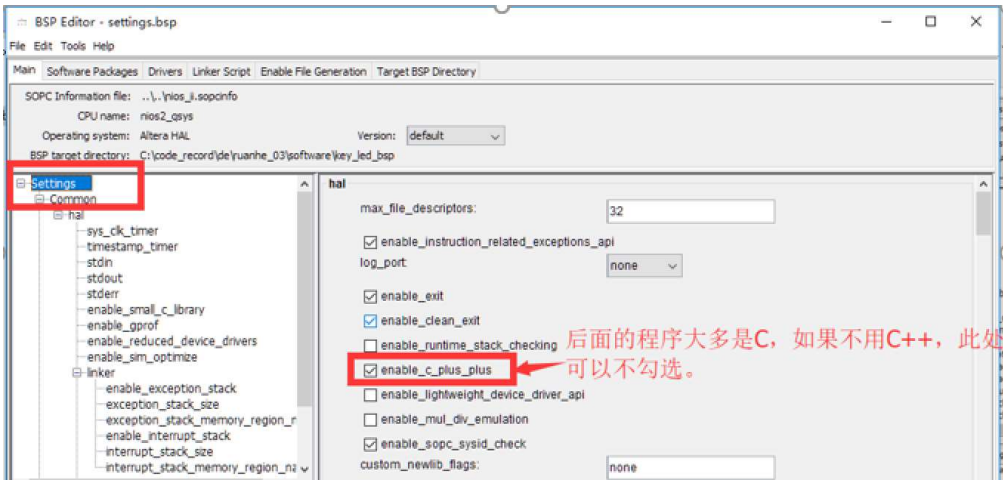
一般出现这个错误是因为软件过大，是因为我们的 RAM 或 ROM 过小导致的，在这里我们可以对 BSP 进行一些设置，使我们的软件变得更“轻巧”。

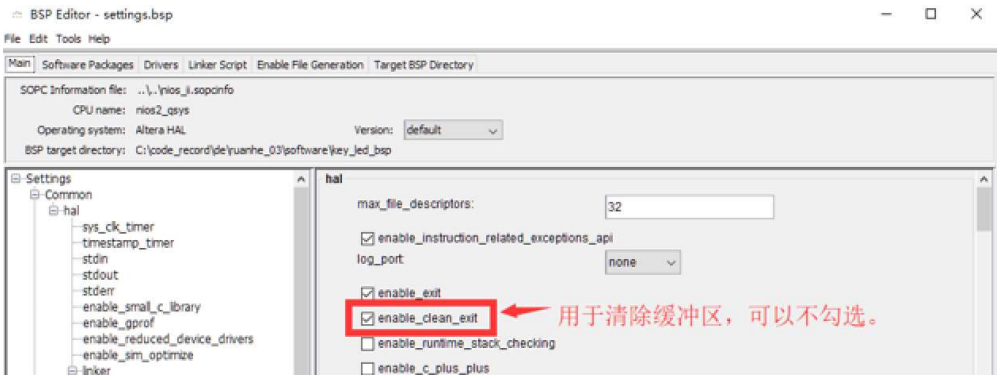
首先，右键点击“key_led_bsp”，选择 Nios II-BSP Eidtor...，弹出对话框如下图所示。这里面包含了对板级支持包详细的设置。



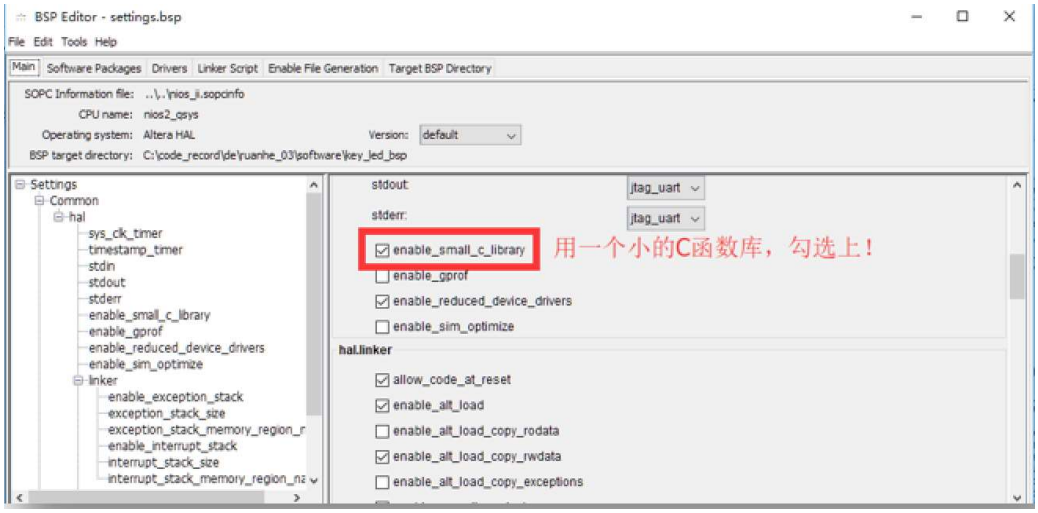


此处重点！若想减小程序，可更改以下几点

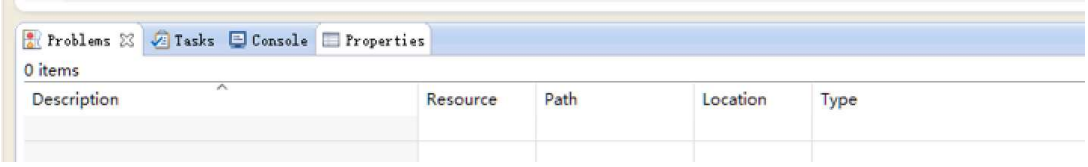




若可保证程序无冗余，多重循环的现象出现，“enable_clean_exit”可以不使用。如果程序比较简单，可使用精简 C 函数库，也可节省很多空间：

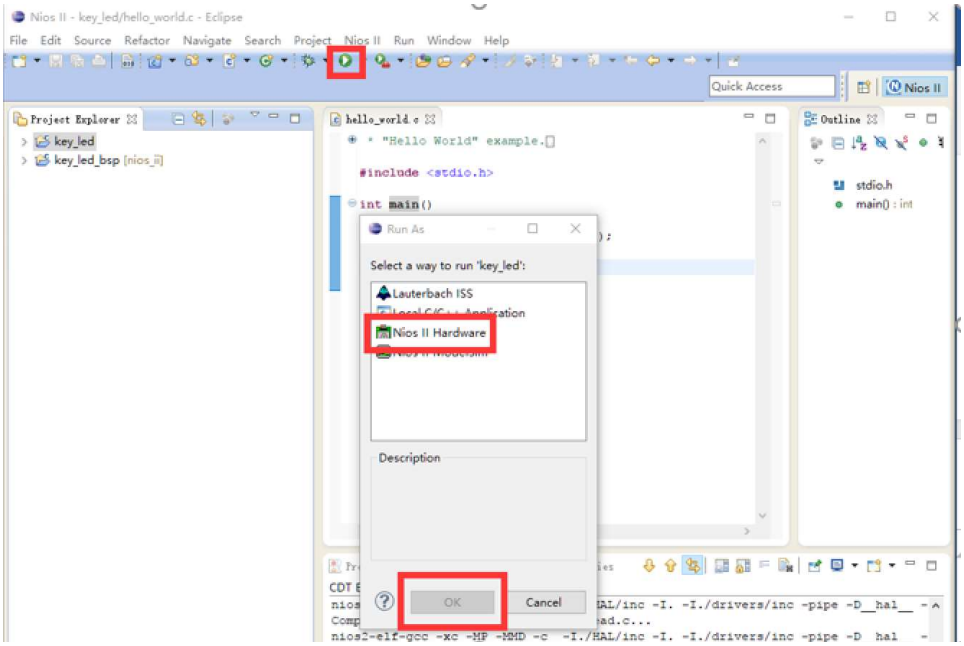


点击 Generate,完毕后点击 Exit,再次编译该工程，可以发现更改这四个选项的效果可谓是立竿见影的，报错信息已经不见了！

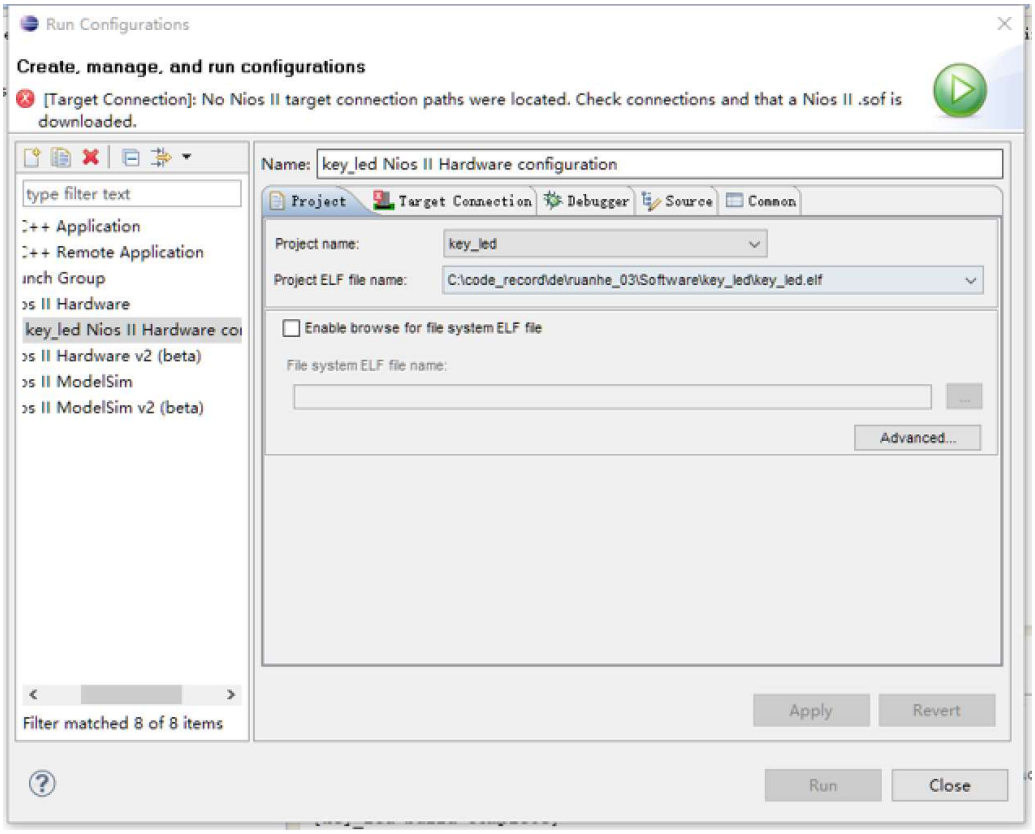


7.5 运行 Nios II：Hello World！

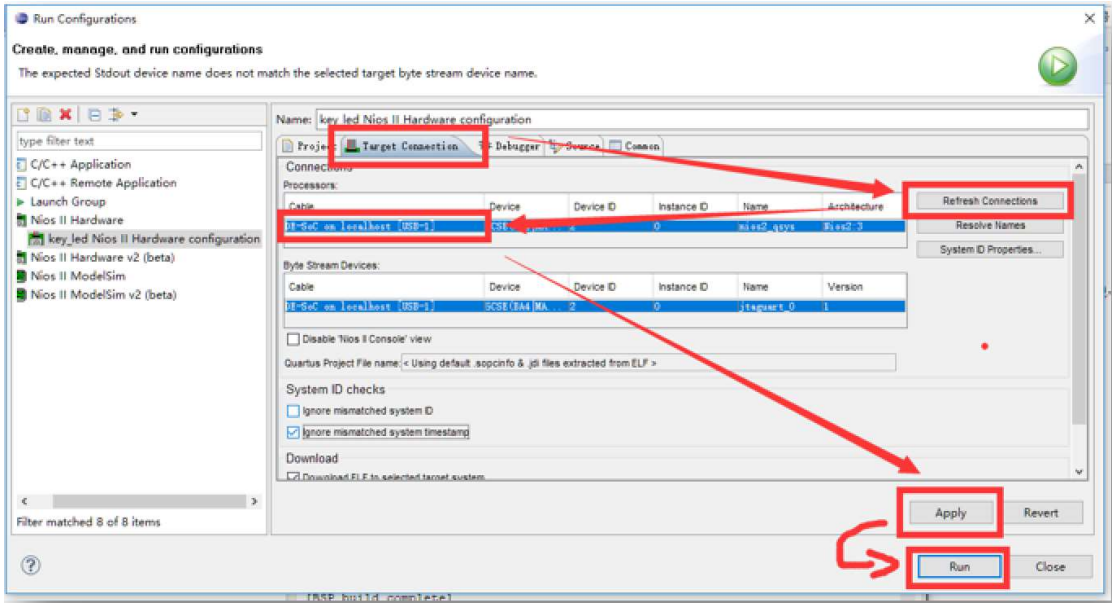
回到 Eclipse，点击”Run”按钮，在弹出的对话框中，选择 Nios II Hardware。



这时，会弹出一个窗口，用于配置 Nios II 的硬件配置。



点击 Target Conection，点击 Refresh Connections，选中对应设备，点击 Apply（如果不能点击 Run，更改 System ID checks 下面的选项）。

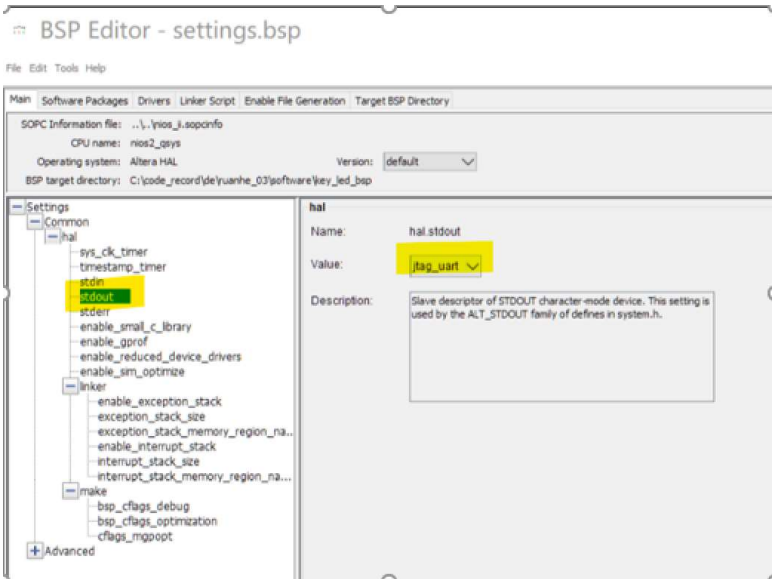


若运行成功，您会在 Console 栏内观察到**“Hello from Nios II!”**字符。

8. 基于 Nios II 处理器的流水灯

在更改程序之前，我们要先了解一下 Nios II 的工作原理。

上一小节，我们成功在搭建好的 Nios II 系统中成功打印了“Hello from Nios II!”，那么为什么会在信息栏中成功打印了“Hello from Nios II!”呢?实际上，我们可以回顾一下前面的内容，我们在系统的外围设备中添加了一个 JTAG_UART，我们可以打开 BSP Editor（右键 key_led_bsp-nios II- Bsp Editor...），找到下图选项：



可以发现，BSP 已经将 std error, srd in ,std out 都已经设置成 JTAG_UART，运行 C 语言中的这段 printf 正常来说是打印在控制台上，但是在 Eclipse 中，我们打印在了 Console 窗口，而这个窗口就是由 JTAG_UART 负责通讯的，你也可以添加另一个 UART IP 利用串口调试工具调试。虽然这只是简简单单打印了一句话，但是这句话，但是打印这句话的这个命令编译出的机器码是运行在 FPGA 内部的硬核（Nios II）上的。

我们本小节要完成的程序是流水灯，那么，我们应该如何做呢？

8.1 更改 C 代码

想要修改程序，当然得更改代码，打开“Hello world.c”，将下面代码覆盖粘贴：

注意，我们只需要修改应用层的程序（application），如无必要，不必修改板级支持包（bsp）。

```
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"  //PIO 操作
#include "alt_types.h"
#include "sys/alt_irq.h"
#include "unistd.h"              //定义 usleep 函数的头文件，可实现延时
#ifndef PIO_LED_BASE             //设置基地址
#define PIO_LED_BASE 0xffffffff
#endif

#if PIO_LED_BASE == 0xffffffff
#error "No definition of PIO_LED core .\n"
#endif

alt_u32 LED_Table[] =
{~0x01,~0x02,~0x04,~0x08,~0x10,~0x20,~0x40,~0x80};    //alt_u32 定义
数组变量

int main(void)
{
    alt_u8 i;       //定义变量 i
    printf("code Runing...");

    while(1)
    {
        for(i=0;i<=7;i++)
        {

IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE,LED_Table[i]);
            //↑该函数为向对应寄存器写数据，
            usleep(50000); //等待 100ms
        }
    }
    return 0;
}
```

8.2 编译并运行程序

编译并运行程序，如果报错，重新更改一下运行设置或者更改一下运行设备。如果运行成功，Console 结果如下图所示，观察开发板可发现程序已成功运行。运行效果为 8 个 LED 循环显示。