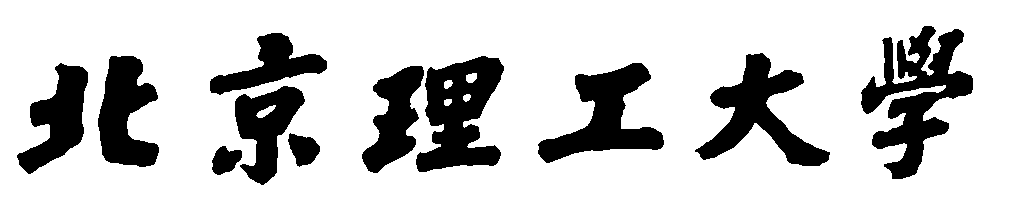
****

*计算机硬件系统设计*

实验一 SOPC流水灯与HPS,Yocto项目

姓 名：沈翰文

学 号：1120151745

组 员：朱威浦，沈翰文，肖子原，廖汉龙

学 院：计算机学院

邮 箱：[liamliaohl@gmail.com](mailto:liamliaohl@gmail.com)

目录

[摘要 4](#_Toc516333911)

[第1章 5](#_Toc516333912)

[1.1 实验名称:： 5](#_Toc516333913)

[1.2 实验目的:： 5](#_Toc516333914)

[1.3 实验原理 5](#_Toc516333915)

[1.4 DE2i150实验步骤 6](#_Toc516333916)

[1.4.1 新建工程 6](#_Toc516333917)

[1.4.2 下载NOIS II 到DE2i150开发板 6](#_Toc516333918)

[1.4.3 编写并运行流水灯代码 8](#_Toc516333919)

[1.5 DE2i150实验结果及拓展功能 12](#_Toc516333920)

[1.5.1 实验结果 12](#_Toc516333921)

[1.5.2 拓展功能 12](#_Toc516333922)

[1.6 Sockit实验步骤 13](#_Toc516333923)

[1.6.1 新建工程 13](#_Toc516333924)

[1.6.2 下载noisII到开发板 13](#_Toc516333925)

[1.6.3 编写并运行流水灯代码 15](#_Toc516333926)

[1.7 Sockit实验结果及拓展功能 16](#_Toc516333927)

[1.7.1 实验结果 16](#_Toc516333928)

[1.7.2 拓展功能 17](#_Toc516333929)

[1.8 问题及解决思路 17](#_Toc516333930)

[1.8.1配置过程问题 17](#_Toc516333931)

[1.8.2 下载过程问题 17](#_Toc516333932)

[第2章 18](#_Toc516333933)

[2.1 实验名称： 18](#_Toc516333934)

[2.2 实验目的 18](#_Toc516333935)

[2.3 实验原理 18](#_Toc516333936)

[2.4 实验步骤 19](#_Toc516333937)

[2.4.1创建并编译hello, world工程文件 19](#_Toc516333938)

[2.4.2 连接Sockit开发板 21](#_Toc516333939)

[2.4.3使用scp命令传送可执行文件 23](#_Toc516333940)

[2.5 实验结果 24](#_Toc516333941)

[2.6 问题及解决思路 24](#_Toc516333942)

[2.6.1 putty无法正常使用 24](#_Toc516333943)

[2.6.2开发板无法识别 24](#_Toc516333944)

[第3章 25](#_Toc516333945)

[3.1实验名称： 25](#_Toc516333946)

[3.2 实验目的 25](#_Toc516333947)

[3.3实验步骤 26](#_Toc516333948)

[3.3.1在DE2i-150板上安装Yocto系统 26](#_Toc516333949)

[3.3.2配置FPGA 26](#_Toc516333950)

[3.3.3复制演示文件到板子 27](#_Toc516333951)

[3.3.4启动PCIE 驱动程序 27](#_Toc516333952)

[3.3.5启动演示相应的应用软件 27](#_Toc516333953)

[3.4实验结果及拓展功能 28](#_Toc516333954)

[3.4.1 PCIE基础演示 28](#_Toc516333955)

[3.4.2 PCIE显示演示 31](#_Toc516333956)

[3.4.3拓展功能 33](#_Toc516333957)

[3.5 问题及解决思路 34](#_Toc516333958)

[3.5.1 启动Yocto故障 34](#_Toc516333959)

[3.5.2 PCIE基础演示时一些功能无法出现预期结果 35](#_Toc516333960)

[第4章 35](#_Toc516333961)

[4.1个人工作和学习过程 36](#_Toc516333962)

[4.2 心得体会 36](#_Toc516333963)

# 摘要

本文主要介绍硬件系统设计课程中实验的过程及心得体会。本小组所有成员通过相互合作，共同学习的方法掌握了在DE2i-150和Sockit实验板下的SOPC流水灯设计，HPS和Yocto下的PCIE实验。其中第一章介绍了SOPC流水灯设计流程和实验结果展示以及实验中遇到的问题和解决思路，包括DE2i-150和Sockit实验板。第二章介绍了HPS使用流程和实验结果展示以及实验中遇到的问题和解决思路。第三章介绍了在Yocto下的PCIE传输实验过程和实验结果展示以及实验中遇到的问题和解决思路。第四章介绍个人在实验过程中的工作及学习步骤，并说明参与实验的心得体会。

# 第1章

1.1 实验名称:：SOPC流水灯设计实验

1.2 实验目的:：通过对SOPC流水灯的设计与实现，了解SOPC的设计理念和SOPC的设计过程，并掌握NIOS软核系统的使用方法。

## 1.3 实验原理

可编程片上系统（SOPC）是一种特殊的嵌入式系统，由单个芯片完成整个系统的主要逻辑功能，具有可裁减、可扩充、可升级和软硬件在系统可编程的功能。

NIOS是一个32位SOPC，实现了在FPGA上构建CPU，RAM等基本内容，可以在上面直接执行C语言程序。通过编写基于NIOS II的程序来控制LED灯，作为NIOS II最基本的应用。

若要在FPGA开发板中运行基于NIOS II的C语言程序，需要建立一个Quartus工程完成NIOS的基本配置，并将 NIOS II软核下载到开发板中。完成NIOS软核下板后，需要自行在集成Eclipse环境中编写C语言程序，最终在开发板上执行C程序，来控制LED灯，实现基于NIOS II的流水灯。

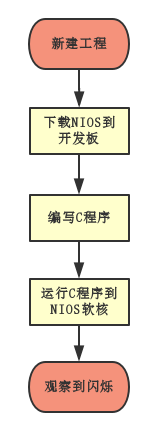
 

图1-1 实验流程图 图1-2 NIOS框架图

## 1.4 DE2i150实验步骤

### 1.4.1 新建工程

新建名为My\_First\_NiosII的Quartus工程，并依据实验指导手册的流程对工程文件进行配置。

### 1.4.2 下载NOIS II 到DE2i150开发板

通过USB下载电缆将电路板连接到主机, 启动Eclipse的Nios II软件构建工具，如图1-3。

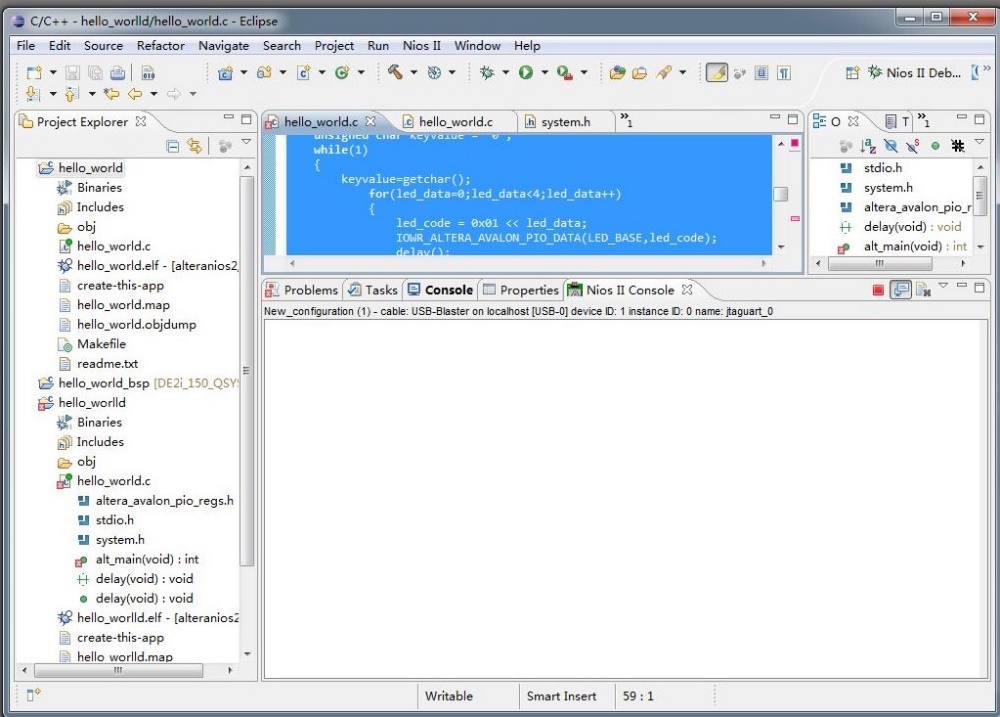


图1-3

选择Nios II-> Quartus II Programmer，如图1-4。

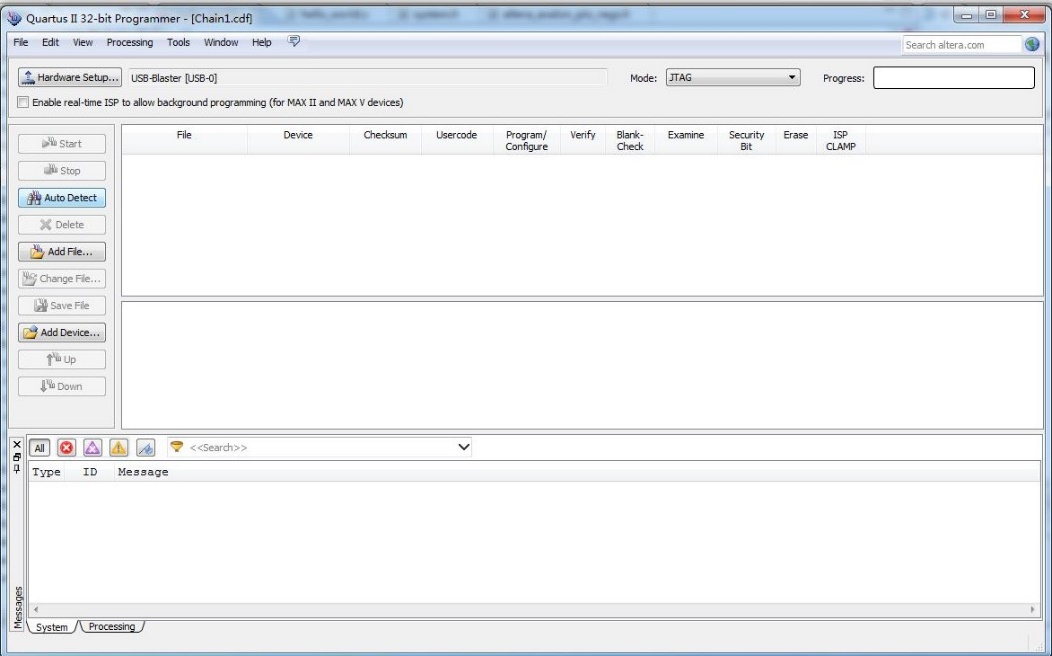


图1-4

配置Nios II系统，如图1-5。

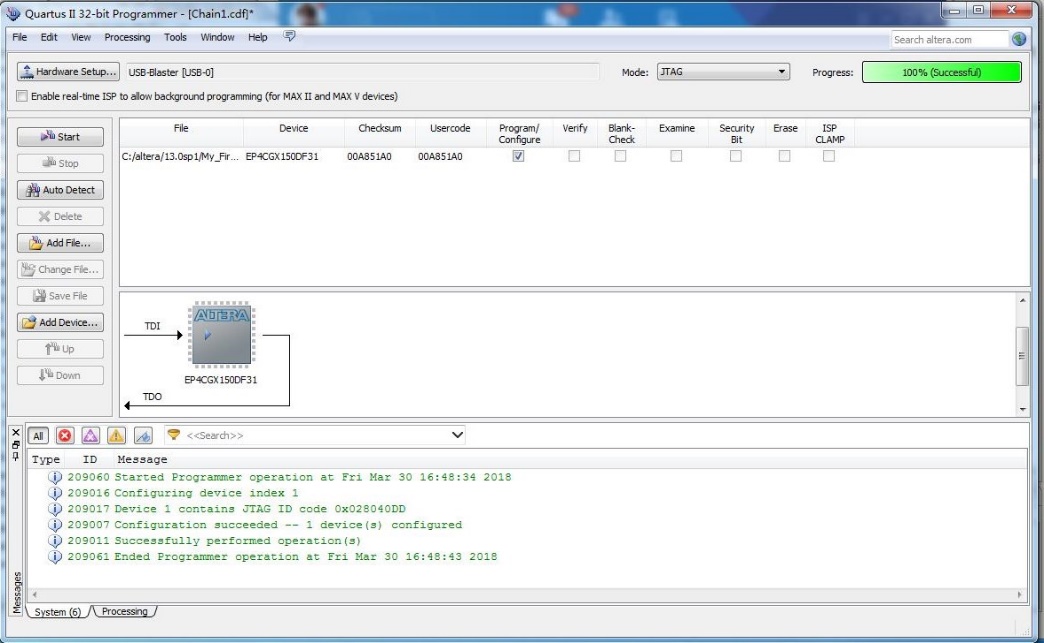


图1-5

### 1.4.3 编写并运行流水灯代码

通过查阅官方文档我们了解到，Nios II系统描述头文件system.h包含Nios II硬件系统中所有组件的软件定义，名称，位置，基址和设置， system.h文件位于My\_First\_NiosII \_bsp目录中。

如果想要控制LED灯的闪烁与否，则需要对相应的PIO管脚进行操作，例如板载LED灯的PIO常量是LED\_BASE，使用独热码进行操作，LED灯对应的码为1时，该LED灯亮，LED灯对应的码为0时，该LED灯灭。使用IOWR\_ALTERA\_AVALON\_PIO\_DATA( )函数对PIO管脚进行操作，IOWR\_ALTERA\_AVALON\_PIO\_DATA( )函数每次调用时都会写入PIO管脚的新值，这个值将会维持一段时间，直到再次调用为止。所以设计LED流水灯的思路在于周期性的改变PIO管脚的值，比如当前值为0001，保持了一段时间之后就变为0010，从而让LED一个接一个地点亮，实现流水灯操作。当然NIOS II软核系统的执行速度是很快的，所以在每个动作之间必须要有一定的延时，为了简单起见，我使用while( )循环来空操作一段时间，从而来控制流水灯的周期。

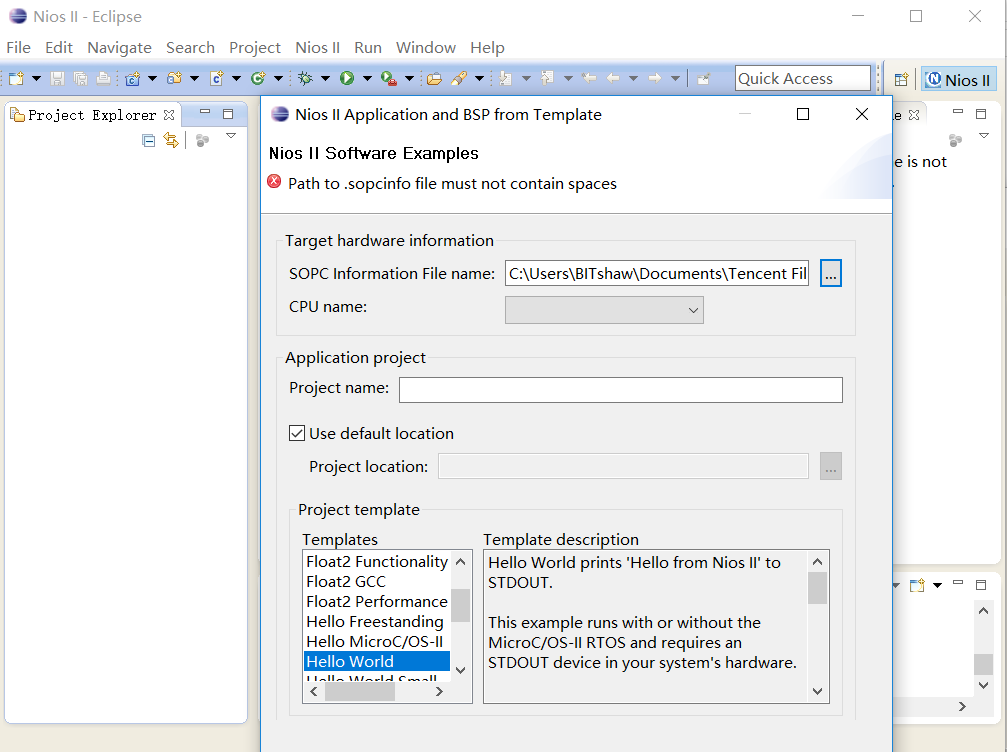
流水灯的C语言代码如代码1 所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #include <stdio.h>  #include "system.h"  #include "altera\_avalon\_pio\_regs.h"  int main(){  while(1)  {       for(i=0;i<4;i++)       {  IOWR\_ALTERA\_AVALON\_PIO\_DATA(LED\_BASE, 1<<i);           delay = 0;           while(delay < 1000000){delay++;}       }  }  return 0;  } |

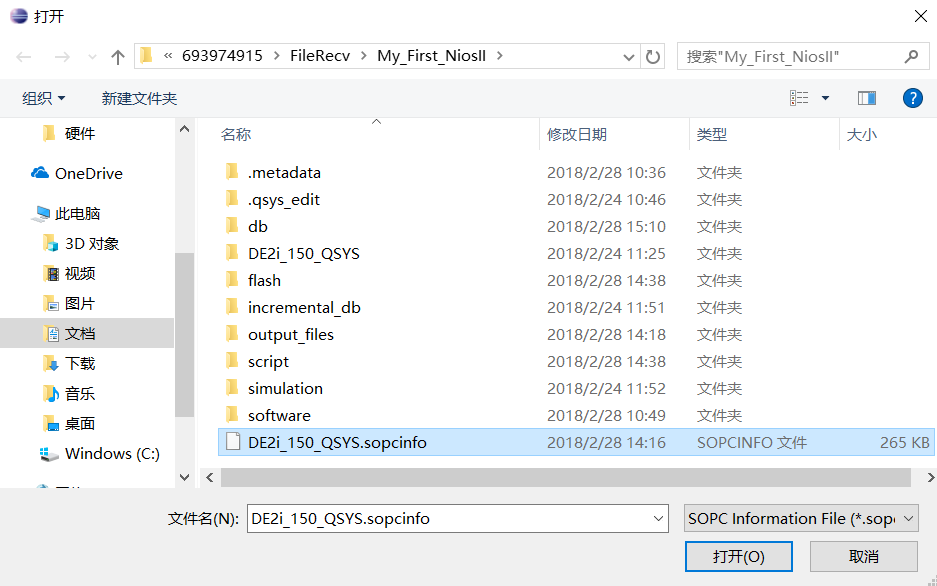
代码1 C语言流水灯代码

新建Eclipse C++项目

Eclipse是一个完整的IDE包含了编辑，运行，debug等功能，为了让刚刚编写的C程序能够正确地下板执行，必须要在Eclipse工程中编译并运行它。在新建Eclipse工程时，我们选择了使用Hello World模板，为了将SOPC系统的配置信息导入到工程，在Target hardware information栏中，选择相应的.sopcinfo文件，如图1-6。

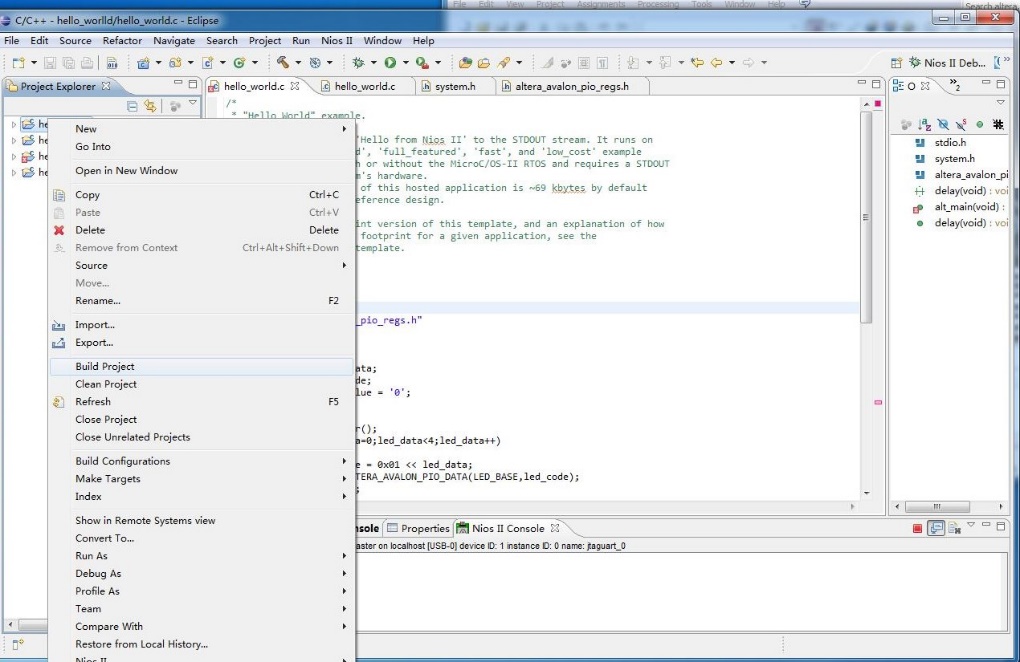
图1-6

在老师提供的压缩包中，选择对应的.sopcinfo文件，输入项目名称就完成了Eclipse工程的配置，将代码1所示的C语言程序写入到工程中，如图1-7。

图1-7

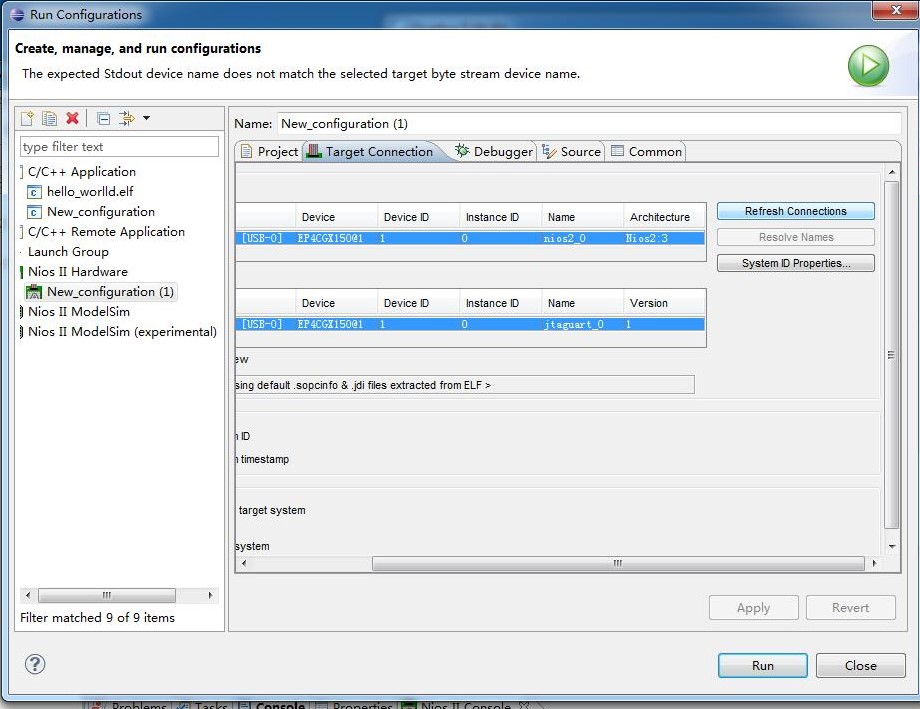
Build C语言流水灯项目

在Eclipse中使用build project可以完成项目的编译和链接操作。但如果不是第一次build，在进行build操作之前要先进行Clean Project操作，以清除上一次build操作的结果。除了选择build project操作还可以选择build all操作，将所有工作空间中的项目都编译链接一次，如图1-8。

图1-8

C程序下板运行

使用Run as> Nios II Hardware可以将C程序下载到开发板中，如图1-9。

图1-9

## 1.5 DE2i150实验结果及拓展功能

### 1.5.1 实验结果

LED灯对应的码为1时，该LED灯亮，LED灯对应的码为0时，该LED灯灭，如图1-10。

图1-10

### 1.5.2 拓展功能

如果想要控制LED灯的闪烁与否，则需要对相应的PIO管脚进行操作，例如板载LED灯的PIO常量是LED\_BASE，使用独热码进行操作，LED灯对应的码为1时，该LED灯亮，LED灯对应的码为0时，该LED灯灭。使用IOWR\_ALTERA\_AVALON\_PIO\_DATA( )函数对PIO管脚进行操作，IOWR\_ALTERA\_AVALON\_PIO\_DATA( )函数每次调用时都会写入PIO管脚的新值，这个值将会维持一段时间，直到再次调用为止。所以设计LED流水灯的思路在于周期性的改变PIO管脚的值，比如当前值为0001，保持了一段时间之后就变为0010，从而让LED一个接一个地点亮，实现流水灯操作。

最后实现的拓展功能为设计的流水灯动作先从左往右运行一次，再从右往左运行一次，最后在从左依次点亮并保持，完成一个周期循环。

## 1.6 Sockit实验步骤

Sockit开发板的下板流程与De2开发板的下板流程大体上相同，所以重点阐述不同的部分。

### 1.6.1 新建工程

新建名为My\_First\_NiosII的Quartus工程，并依据实验指导手册的流程对工程文件进行配置。

### 1.6.2 下载noisII到开发板

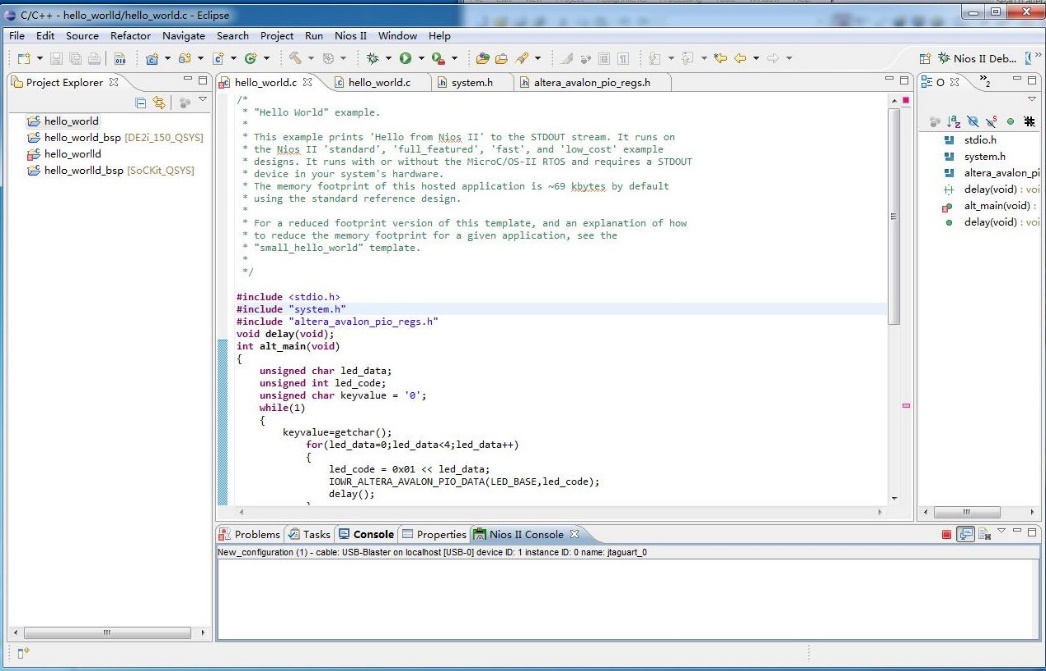
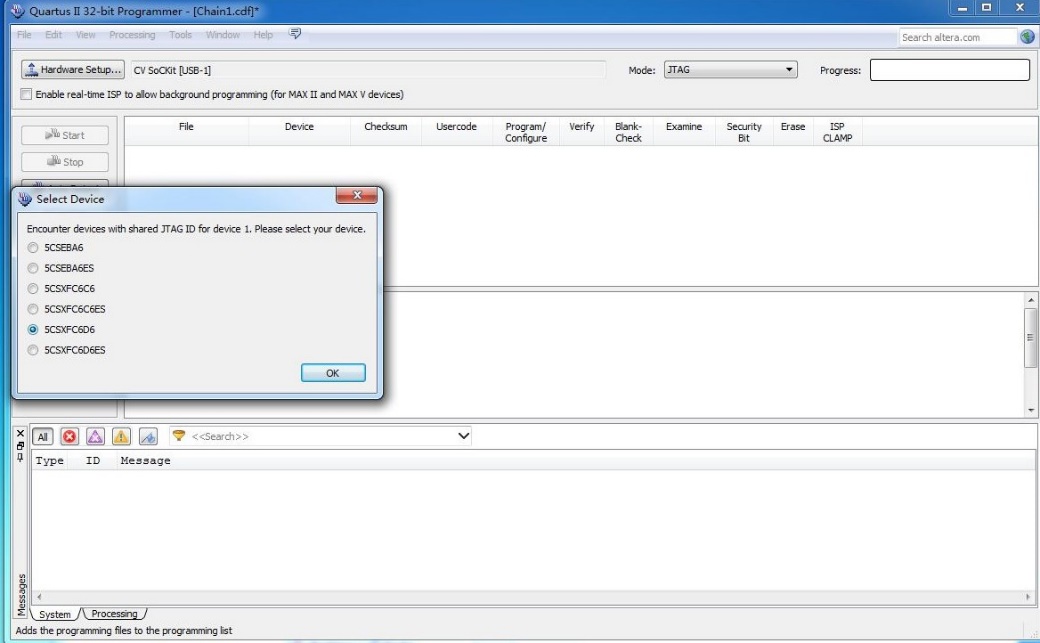
通过USB下载电缆将电路板连接到主机, 启动Eclipse的Nios II软件构建工具，如图1-11。

图1-11

配置Nios II系统

观察板子上的FPGA芯片型号，点击自动匹配芯片。建立Quartus工程时生成的.sof文件必须要与板子的物理特性相匹配，否则会造成下板失败，如图1-12。

图1-12

加载.sof文件

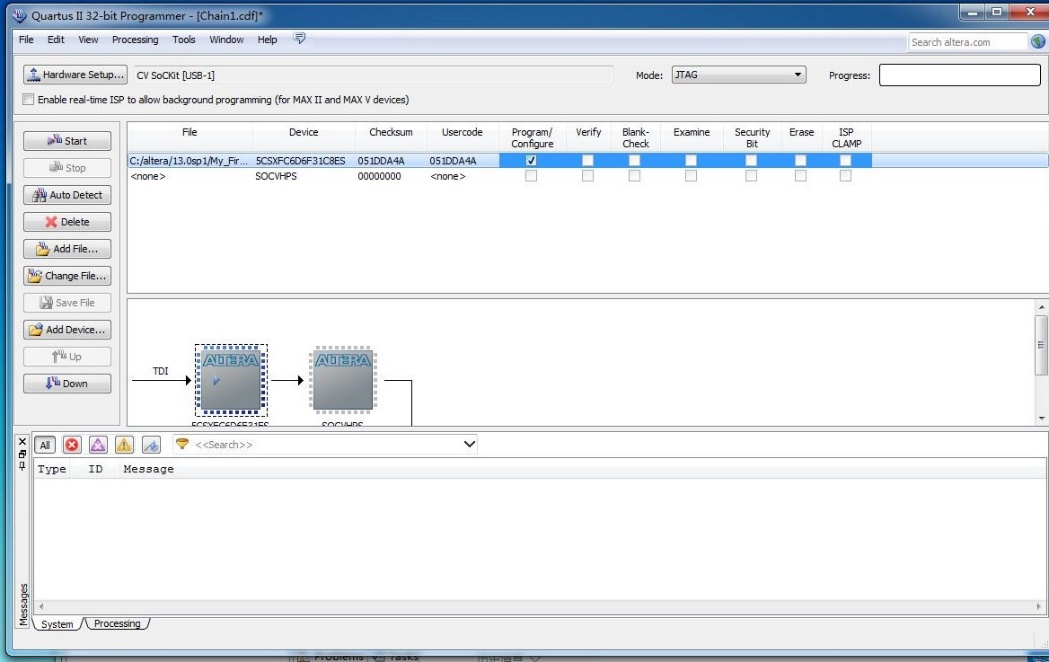
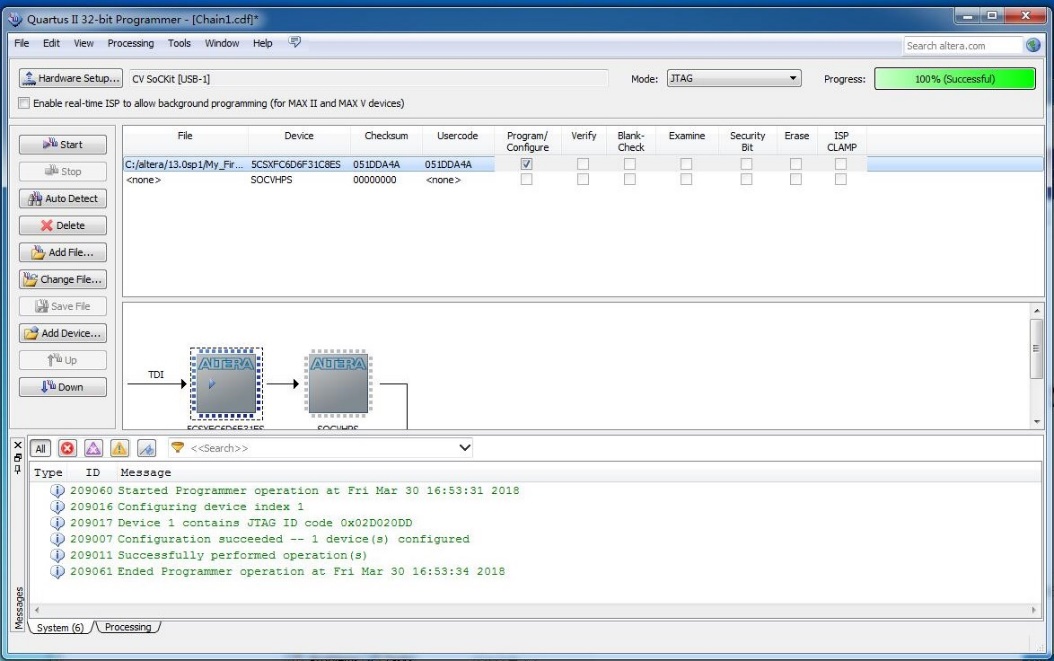
直接单击芯片对应的File栏，就会出现文件选择框，在文件选择框中选择建立Quartus工程时生成的.sof文件，如图1-13。

图1-13

完成NIOS II系统的烧录，如图1-14。

图1-14

### 1.6.3 编写并运行流水灯代码

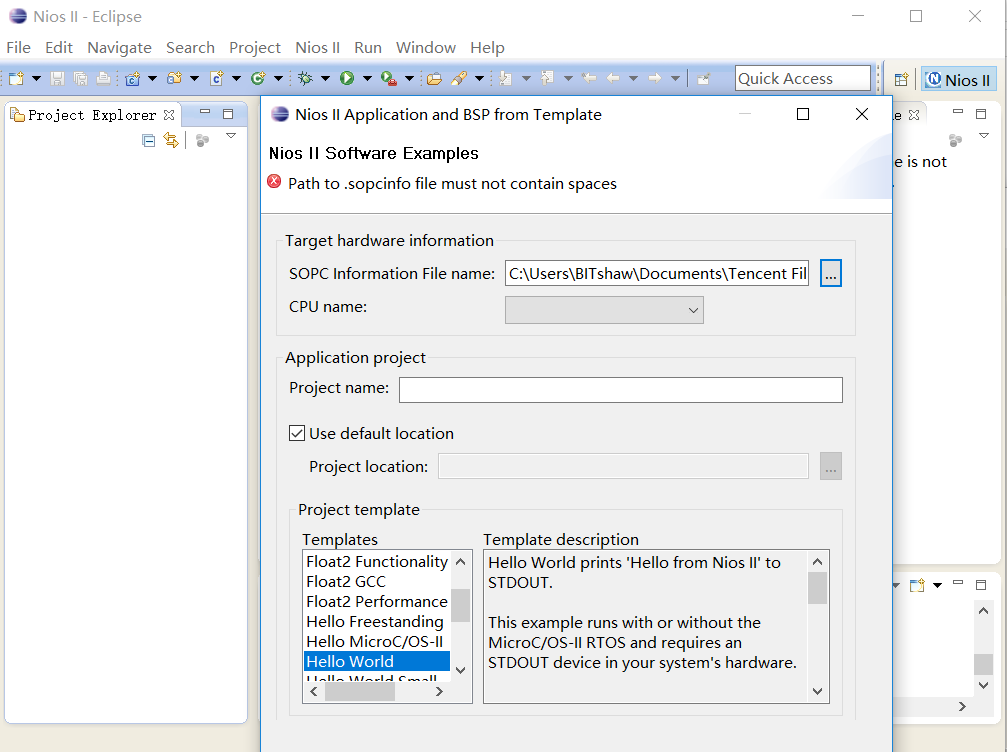
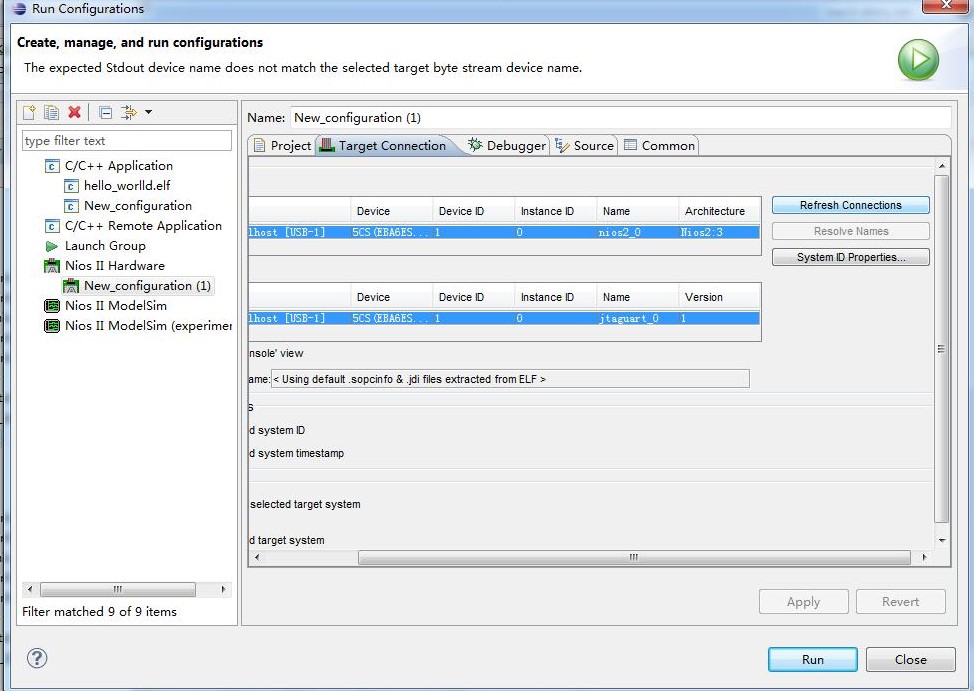
新建Eclipse C++项目，如图1-16。

图1-16

Build C语言流水灯项目，如图1-17。

图1-17

## 1.7 Sockit实验结果及拓展功能

### 1.7.1 实验结果

LED灯对应的码为1时，该LED灯亮，LED灯对应的码为0时，该LED灯灭，如图1-18。



图1-18

### 1.7.2 拓展功能

实现的拓展功能为设计的流水灯动作先从左往右运行一次，再从右往左运行一次，最后在从左依次点亮并保持，完成一个周期循环。

## 1.8 问题及解决思路

### 1.8.1配置过程问题

在配置Nios II系统时，需要注意一些细节。建立Quartus工程时生成的.sof文件必须要与板子的物理特性相匹配，否则会造成下板失败。为了让Programmer正确地识别开发板的物理型号，避免手动输入造成错误，将开发与PC链接并安装驱动之后，就可以使用Auto选项来识别FPGA芯片。

在选择完成后，FPGA就会出现下方的示意框中，但此时，如果将.sof文件写入到板子会造成错误。经过多次摸索尝试之后，我们发现了一个可行方法，直接单击芯片对应的File栏，就会出现文件选择框，在文件选择框中选择建立Quartus工程时生成的.sof文件，就可以完成.sof文件和硬件的匹配，选择开始烧录，则完成了NIOS II软核系统的下板。

### 1.8.2 下载过程问题

使用Run as> Nios II Hardware可以将C程序下载到开发板中。但这一步要特别注意一点，Eclipse不能直接识别出开发板，所以直接右键单击工程不会出现Run as> Nios II Hardware选项。为了解决这个问题，我们尝试了许多种方法，最终找到了一种可行方案：选择运行配置选项，点击NIOS II Hardware 可以生成一个新的配置，在New configuration的Target Connection选项中有一个刷新连接选项，在尝试几次刷新之后，可以在Device栏中找到相应的设备，此时点击运行即可。但有些时候点击运行并不会马上观察到结果，根据之前的经验我们猜测可能是Eclipse对开发板的识别存在滞后，这时，再次点击刷新即可观察到LED流水灯效果。

# 第2章

2.1 实验名称：Sockit开发板下的HPS(硬核处理器系统)实验

2.2 实验目的

通过使用Sockit开发板，对HPS 有一定的初步了解，掌握基于Sockit的HPS开发的基本操作。了解如何使用Putty 连接Sockit开发板中的Linux系统。了解在Liunx下的基本的C程序的编译与运行过程。

## 2.3 实验原理

SoC EDS是Altera SoC器件上的软件开发。它包含开发工具实用程序，运行时软件，和应用实例来跳转启动固件和应用软件开发。我们可以使用SoC EDS开发固件和应用软件。

Putty是一款集成虚拟终端、系统控制台和网络文件传输为一体的免费且开源的程序。它支持多种网络协议，包括SCP，SSH，Telnet，rlogin和原始的套接字连接。它也可以连接到串行端口。

本实验中主要是使用Putty来连接当前的windows系统与开发板，从而使用Linux的终端进程项目的编译与结果显示。

实验流程如图2-1。

图2-1

实验装置连接如图2-2

|  |
| --- |
|  |

图2-2

## 2.4 实验步骤

### 2.4.1创建并编译hello, world工程文件

在 “my\_first\_hps”文件夹下创建一个名为“main.c”的空文件。然后，在文件中输入以下代码并保存,将此文件夹存放在embedded文件夹的根目录下，当然这是为了后续操作的方便，也可以放在任意文件夹下，但是后续需要输入绝对路径来查找文件。

|  |
| --- |
|  |

由于Altera SoC EDS需要makefile才能知道如何编译和链接项目，所以需要在“my\_first\_hps”文件夹下创建一个名为“Makefile”的空文件。 然后，输入以下内容并保存。 在makefile中，“TARGET”变量定义了输出文件名。

|  |
| --- |
|  |

编译项目如下：

|  |
| --- |
|  |

图2-3

要编译一个项目，需要启动Altera嵌入式Command Shell，即上图所示的Embedded\_Command\_Shell.bat, 打开以后，所有的命令和Linux下是一致的，需要的操作是make在第一二步中已经写好的工程项目，操作如下：

|  |
| --- |
|  |

图2-4

可以发现，上图中，由于已经是make过的文件，所以出现的提示是未build, 已经make成功的标志是出现了“my\_first\_hps“ 文件。

在后续的步骤中，是通过Putty，在当前系统下操作Linux 运行my\_first\_hps，显示结果。

### 2.4.2 连接Sockit开发板

将开发板，路由器与计算机按照下图所示相连

|  |
| --- |
|  |

图2-5

使用Putty 连接开发板

打开putty.exe，点击Serial,配置如下，可以发现其实这个板子的连接条件是苛刻的，再有些计算机下边需要把Flow control设置成None才行。

|  |
| --- |
|  |

图2-6

再点击Session，进行连接即可。

连接上之后的操作如下

使用root 账户登录，直接再出现的终端方框中输入root回车

输入“udhcpc”从DHCP服务器查询IP。

|  |
| --- |
|  |

图2-7

输入“ifconfig”以检查SoCKit板的以太网IP。 在本教程中，将IP“192.168.1.113”分配给SoCKit板

|  |
| --- |
|  |

图2-8

通过以上操作，可以得知，分配给开发板的IP地址为10.1.1.143

“scp”命令需要密码，即使用Soc EDS 将可执行文件传送到开发板的操作需要密码，由于做到这一步尚未定义root帐户的密码，输入Linux命令“passwd”以创建密码。

|  |
| --- |
|  |

图2-9

### 2.4.3使用scp命令传送可执行文件

使用“scp”命令将可执行文件“my\_first\_hps”复制到SD卡中。 在Altera SoC命令行程序中，键入“scp my\_first\_hps root@10.1.1.143:/home/root”将文件复制到文件夹“/home/root”中。 其中，“10.1.1.143”IP地址是在上一步中获得的。 当看到提示信息“您确定要继续连接（是/否）？”时，请输入“yes”并按ENTER键回复yes。 接下来，输入密码。

|  |
| --- |
|  |

图2-10

## 2.5 实验结果

使用Putty 运行可执行文件

由于可执行文件已经成功的传送到SD中，所以，最有只需要和平时再Liunx下编译项目文件时一样，找到文件所在的目录下（当前靠在了home/root目录下），执行“./my\_first\_hps“,可得到编译结果，出现hello,world!。

|  |
| --- |
|  |

图2-11

## 2.6 问题及解决思路

### 2.6.1 putty无法正常使用

putty并不要求一定要是有线的方式，通过路由器来连接，实际上我们在做实验的时候，由于路由器是有问题的（小米路由器），路由器内部可能对网线连接有限制，所以即使换了计算机进行操作也一直没有成功，最后，直接使计算机连接无线校园网，Sockit开发板插入到实验室网口，也可连接校园网，这样操作使完全没有问题的。

### 2.6.2开发板无法识别

调整开发板的模式，将背面可调按钮调节成1向下，其他向上。

并且此串口的驱动程序已经在光盘文件中给出

|  |
| --- |
|  |

图2-12

# 第3章

3.1实验名称：PCIE DEMO

## 3.2 实验目的

安装了一枚Intel的ATOM N2600微处理器是Terasic旗下DE2i-150板的特色。在这个微处理器上可以跑简单的Linux系统。但如何利用好这个微处理器？一个很重要的方面就是将微处理器与FPGA进行联动。在我看来，本实验的实验目的主要有以下几项：

1. 增强对DE2i-150内部结构的理解，尤其是微处理器和FPGA之间的PCIE连接
2. 了解DE2i-150内微处理器Linux系统安装及该系统下的相关操作
3. 实现简单的PCIE演示功能

## 3.3实验步骤

### 3.3.1在DE2i-150板上安装Yocto系统

本实验首先要在DE2i-150板上安装Yocto系统，该系统将被安装在微处理器上。在这个地方我们一开始遇到了一些问题，详见后面的“3.5问题及解决思路”的部分。

### 3.3.2配置FPGA

PCIE DEMO也需要对FPGA进行配置，这里配置的方法相对比较简单。只需用USB电缆将板子和主机（这里用的是惠普笔记本电脑）连接，然后在DE2i-150系统盘的Demonstrations/FPGA路径下选取对应的PCIE文件夹。注意此处两种PCIE DEMO用到的文件夹不同，此处若不进行正确的配置将有可能导致之后PCIE DEMO失败，比如无法进行按钮状态演示等。我们组先做“PCIE基础演示”，则此处应选择的文件夹即为“PCIE\_Fundamental”，见图3-1。该文件夹的“demo\_batch”目录下有个叫“sof\_download.bat”的批处理文件，双击运行，即可完成配置工作。此时，还应重启系统以使下板的内容生效。

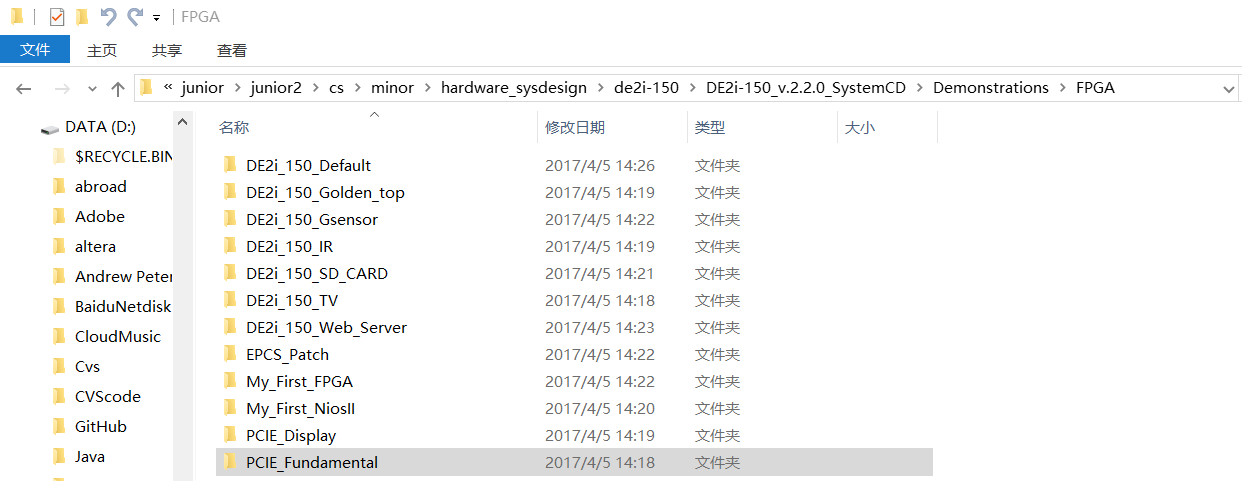


图3-1 配置FPGA时需选用正确的文件夹

### 3.3.3复制演示文件到板子

下一步就是把演示要用到的文件夹拷到板子上。此处的演示文件为“DE2i-150\_v.2.2.0\_SystemCD\Demonstrations\PCIe\_SW\_KIT\linux\PCIe\_DriverInstall”和“DE2i-150\_v.2.2.0\_SystemCD\Demonstrations\FPGA\PCIE\_Fundamental\linux\_app”（同理若是做PCIE显示演示则需要选择不同的文件夹）。在板子的yocto系统下的“/media”目录下可以找到自己的USB存储器，把相关的文件移动到“/home/root”目录下，供之后使用。重启yocto系统。

### 3.3.4启动PCIE 驱动程序

用终端打开/home/root目录，打开“PCIe\_DriverInstall”文件夹下的“yocto-3.10.11”目录，输入“sh ./load\_terasic\_qsys\_pcie\_driver.sh”命令。返回“找到匹配设备”信息则说明PCIE已启动。

### 3.3.5启动演示相应的应用软件

用终端打开/home/root目录，打开“linux\_app”文件夹输入“./app”命令，下面进行相应的操作即可，详见第五部分“实验结果和分析”。

以上是PCIE基础演示的基本步骤，对于PCIE显示演示，步骤基本相同，此处略去不表。

## 3.4实验结果及拓展功能

### 3.4.1 PCIE基础演示

在按上面第四部分的实验步骤走完后，会出现如图3-2所示的菜单。

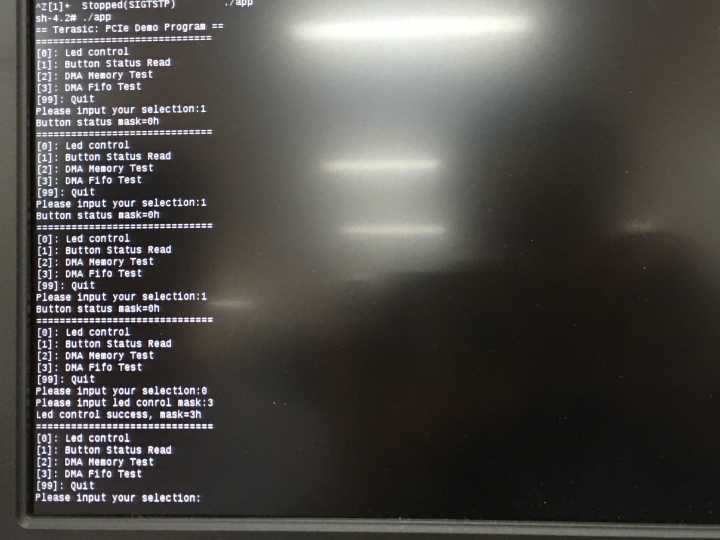


图3-2 PCIE基础演示在终端出现的菜单

从图中可以看出，该菜单一共包括“Led control”、“Button Status Read”、“DMA Memory Test”、“DMA Fifo Test”、“Quit”五项内容。这5项内容由其前面的数字序号指定。比如我们要测试“Led control”即对板上前4个LED灯的控制情况，则需先在终端输入“0”，表示选用该功能。下面我们分开演示一下这4个功能的实验结果。

（1）Led control

先在终端输入“0”，表示选用该功能，然后再输入相应的控制掩码，比如输入的是2，其转成二进制就是“0010”，所以按道理LED1应该亮，然后LED0、LED2、LED3都不亮。实验结果如下图3-3。

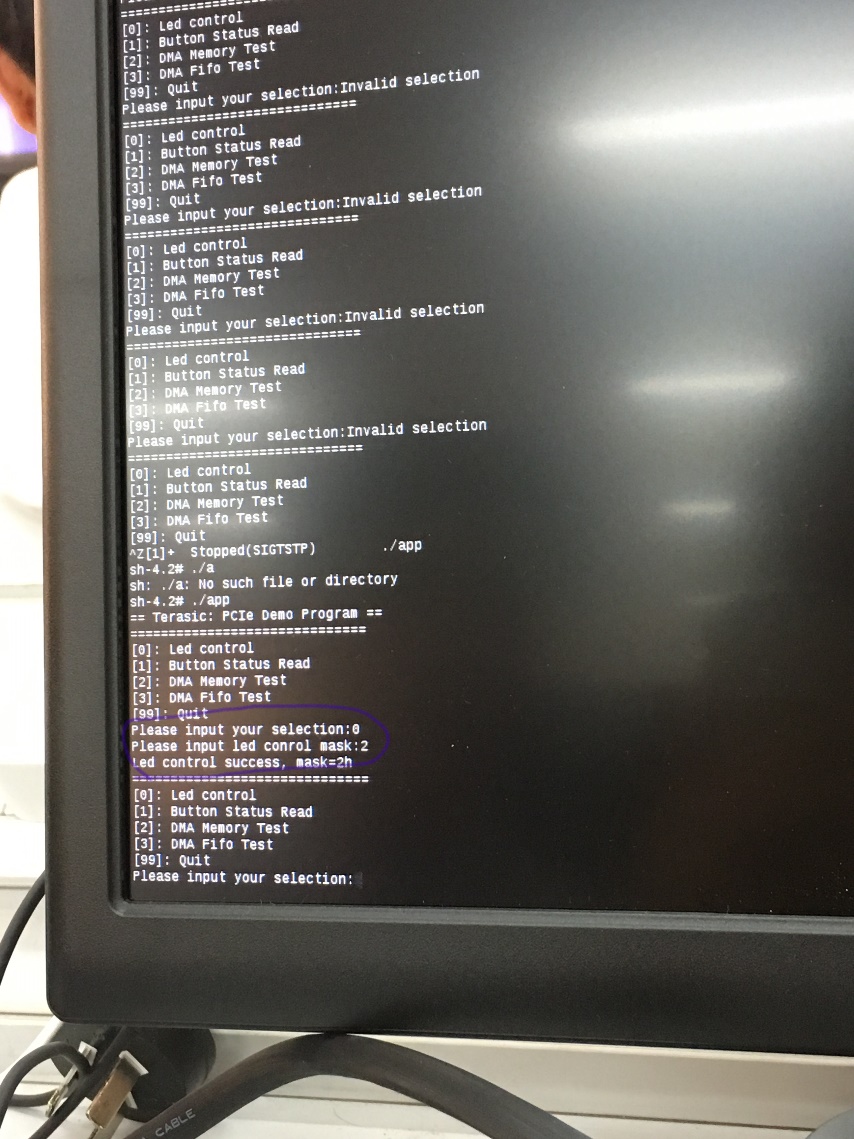


图3-3 LED控制测试，输入掩码为2，显示控制成功

从图中可以看到，终端输出了“Led control success, mask = 2h”的信息。而亮灯的情况如图3-4。

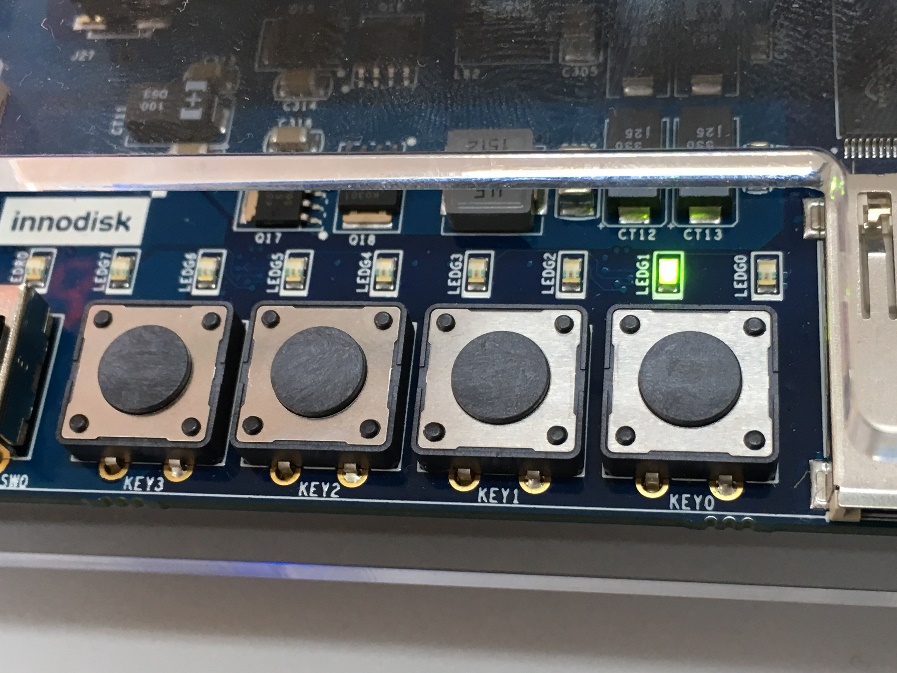


图3-4 输入控制掩码2时的LED灯点亮情况

从图中可以看到LED1亮了，其他LED灯都处于熄灭状态，刚好表示了0010b=2，这与前期的预测相符。

（2）Button Status Read

按动实验板上的任意按键，Terminal界面中以十六进制显示按键的编码，实验结果如图3-5。

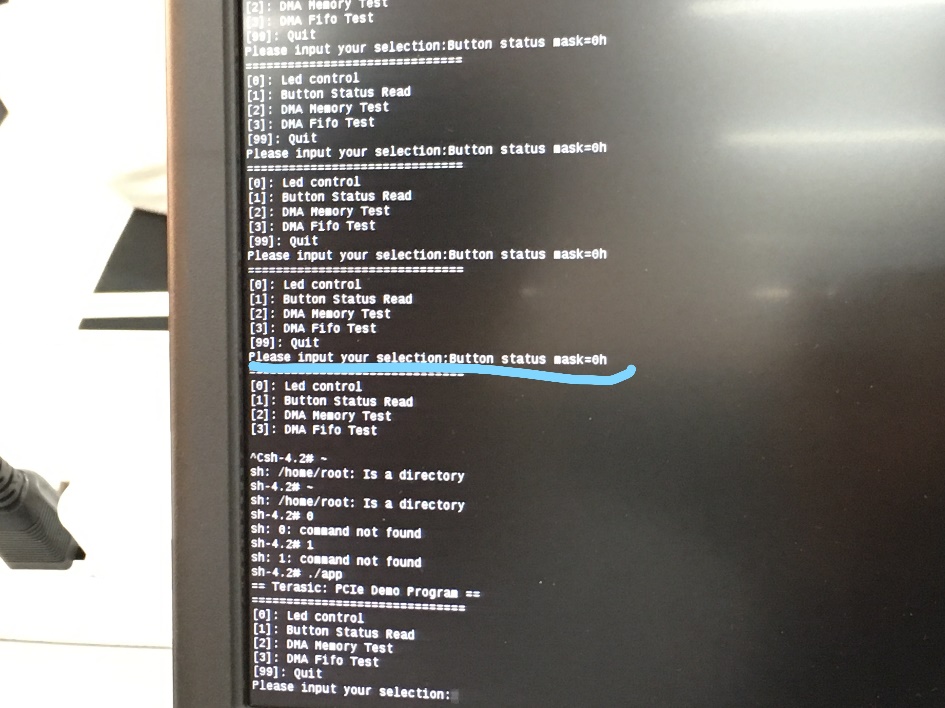


图3-5

（3）DMA存储测试

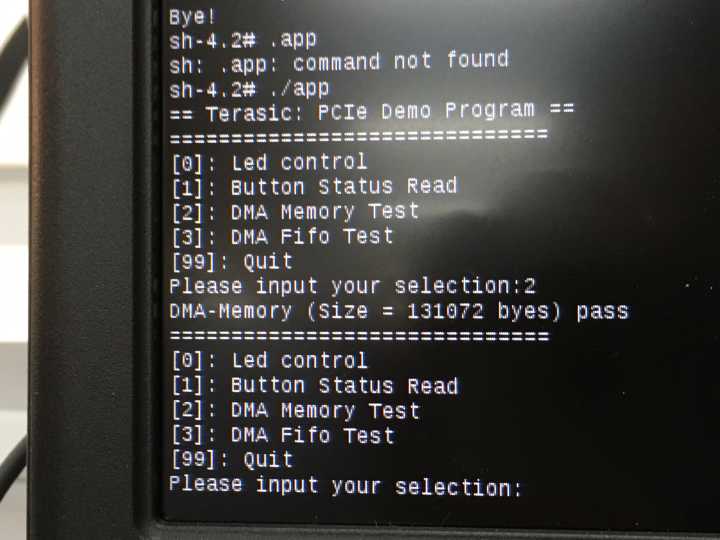


图3-6

（4）DMA FIFO

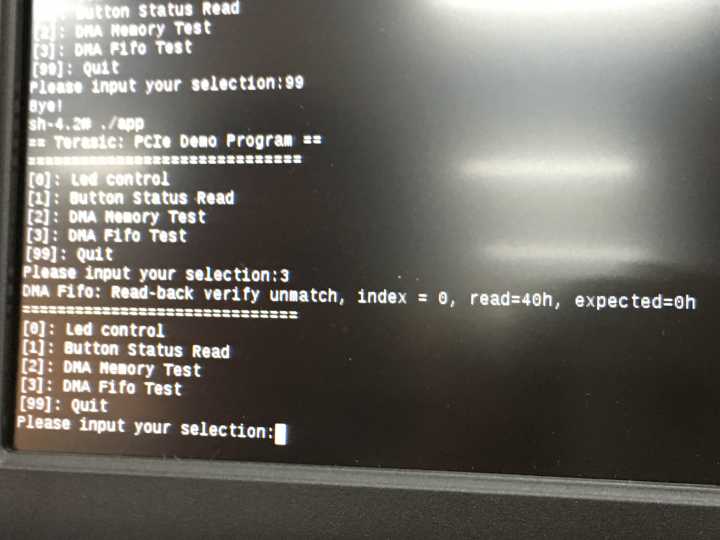


图3-7

### 3.4.2 PCIE显示演示

PCIE显示演示的步骤如第四部分里描述的那样。在终端进入指定目录输入“./app”命令，出现如下图3-8的数据传输字样，此时把显示器直接转接到板子的J8口，则会出现图3-9的显示结果。至此，实验结果陈述完毕。

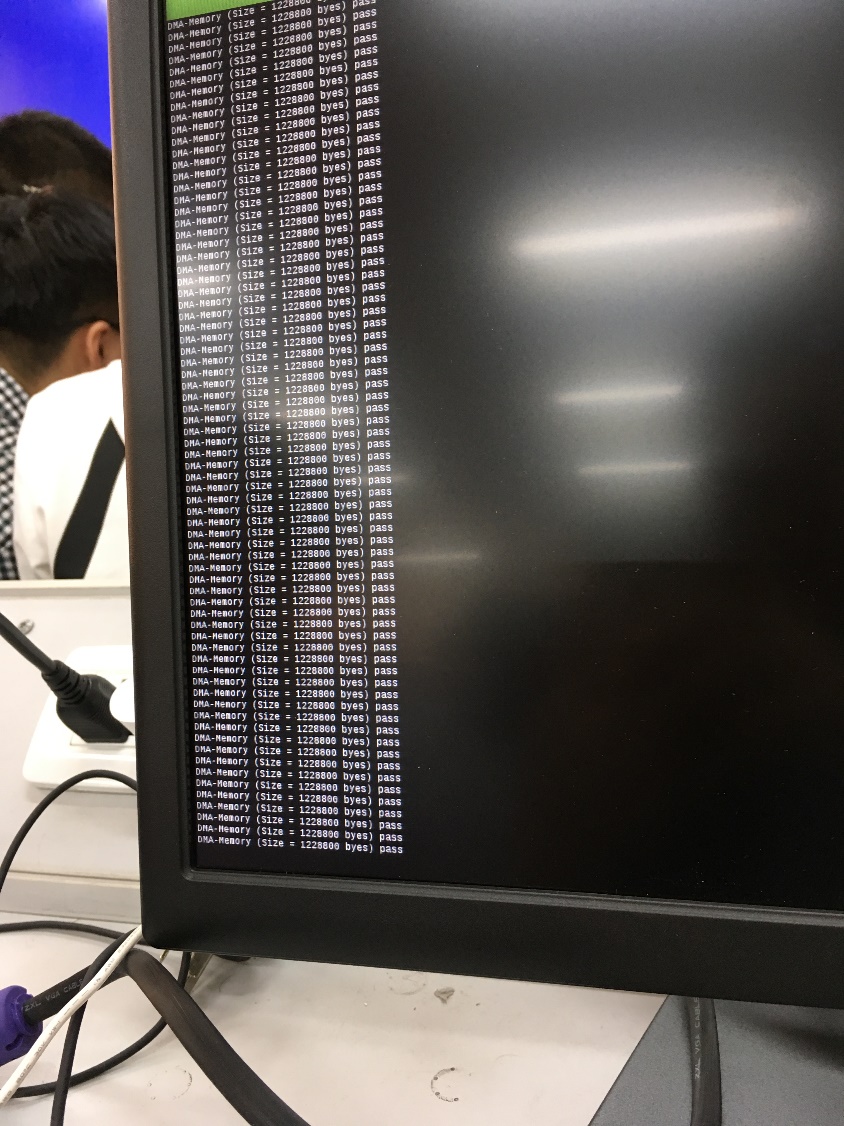


图3-8 PCIE显示演示未接显示器时终端传输的数据

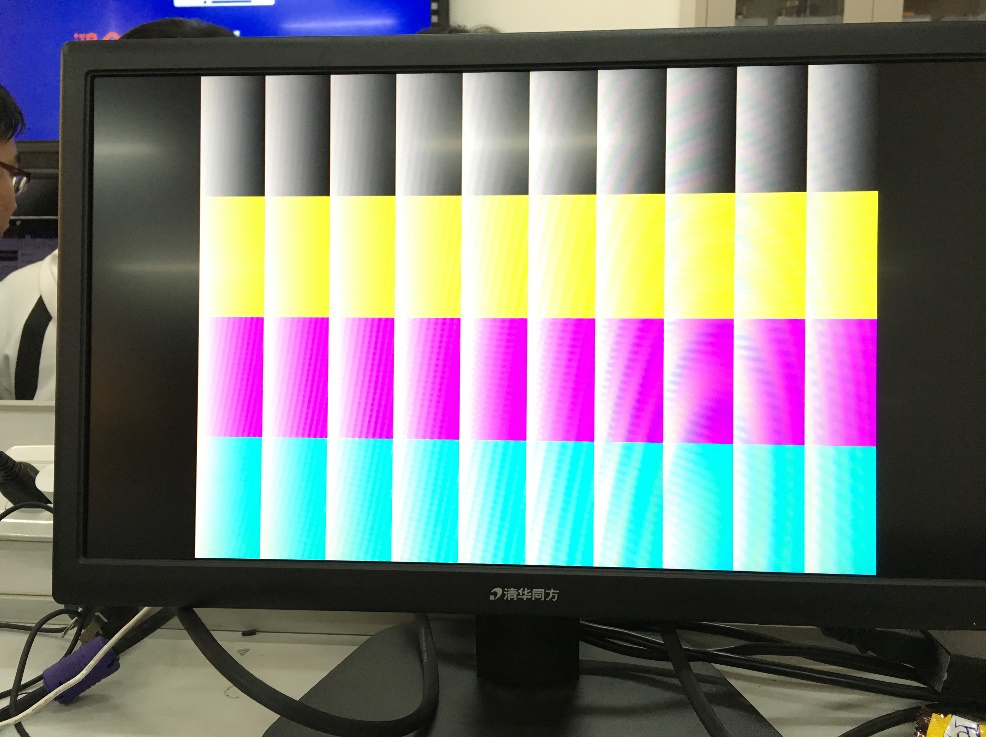


图3-9 PCIE显示演示接显示器的演示结果图

### 3.4.3拓展功能

1. 编写自己的PCIE演示app

为了编写自己的PCIE演示app，我们需要找到“linux\_app”文件夹里对应的晕源文件。



图3-10

这里我打开的是“PCIE\_Fundamntal”目录下的“linux\_app”文件，其中的app.c文件可以预见是之前在终端生成功能菜单的源文件。

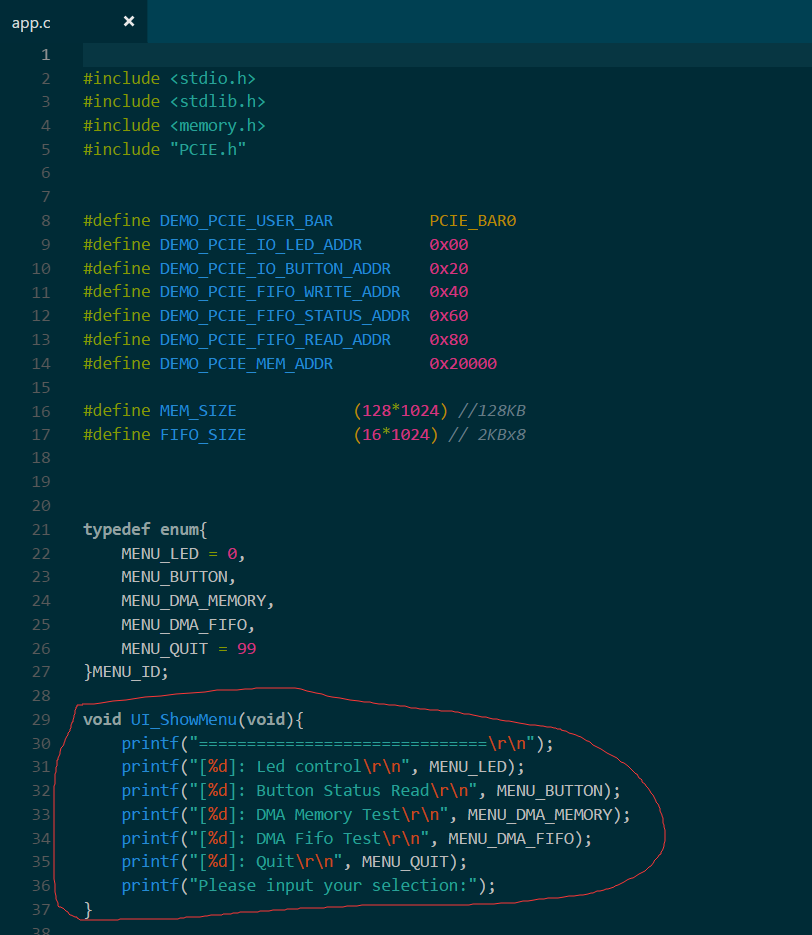


图3-11 app.c文件部分代码截图

从图中我们可以看到“UI\_ShowMenu”函数显然是显示菜单界面的函数，与我们的猜想一致。这样，我们只需要修改app.c源文件，就可以实现我们自己想要的PCIE演示功能。必要时可以修改Makefile文件以便对源文件进行编译。

## 3.5 问题及解决思路

### 3.5.1 启动Yocto故障

我们小组在最初给DE2i-150板子装Linux系统的时候按照老师给出的文档进行一步步操作，包括设置U盘启动等。但尝试了若干次重启用U盘进行系统install的时候，总是会出错，如图3-12。

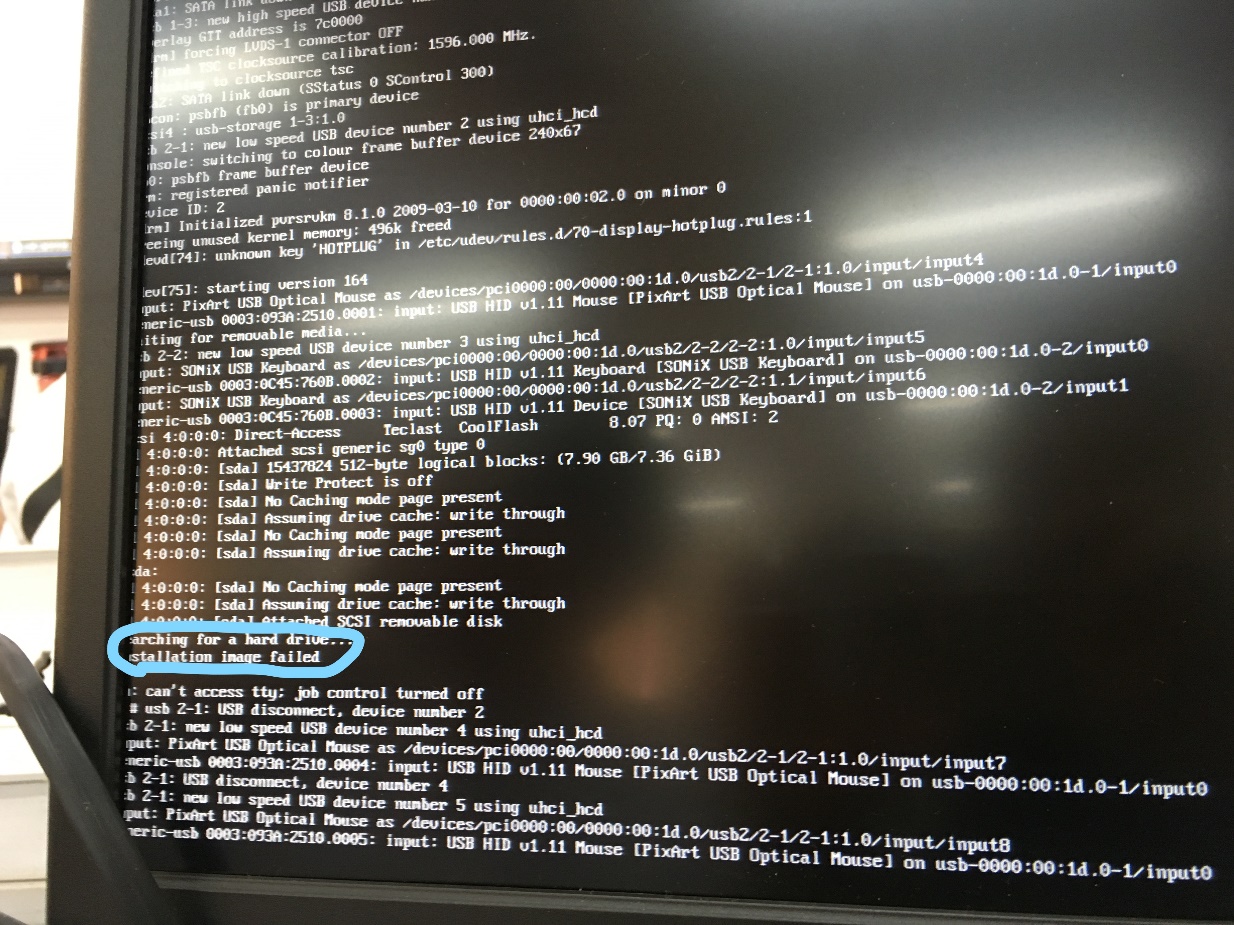


图3-12 U盘启动时出现的“searching for a hard drive”问题

从图中可以看出问题是没有找到空闲硬盘。

于是，我们怀疑DE2i-150板子没装硬盘。经过和其他组的讨论交流，发现板子的包装盒里果然有自带的硬盘，不过需要手装。装上后我们还惊喜地发现该硬盘上本身就带有yocto系统，我们不用再用U盘进行启动了。

### 3.5.2 PCIE基础演示时一些功能无法出现预期结果

在测试功能2即按钮状态测试时我们一度遇到一些问题，就是无论按钮怎么按，最终显示的状态都是0h。最终发现是下板时相关管脚文件并没有烧进去，即sof.bat执行完后没有进行reboot。

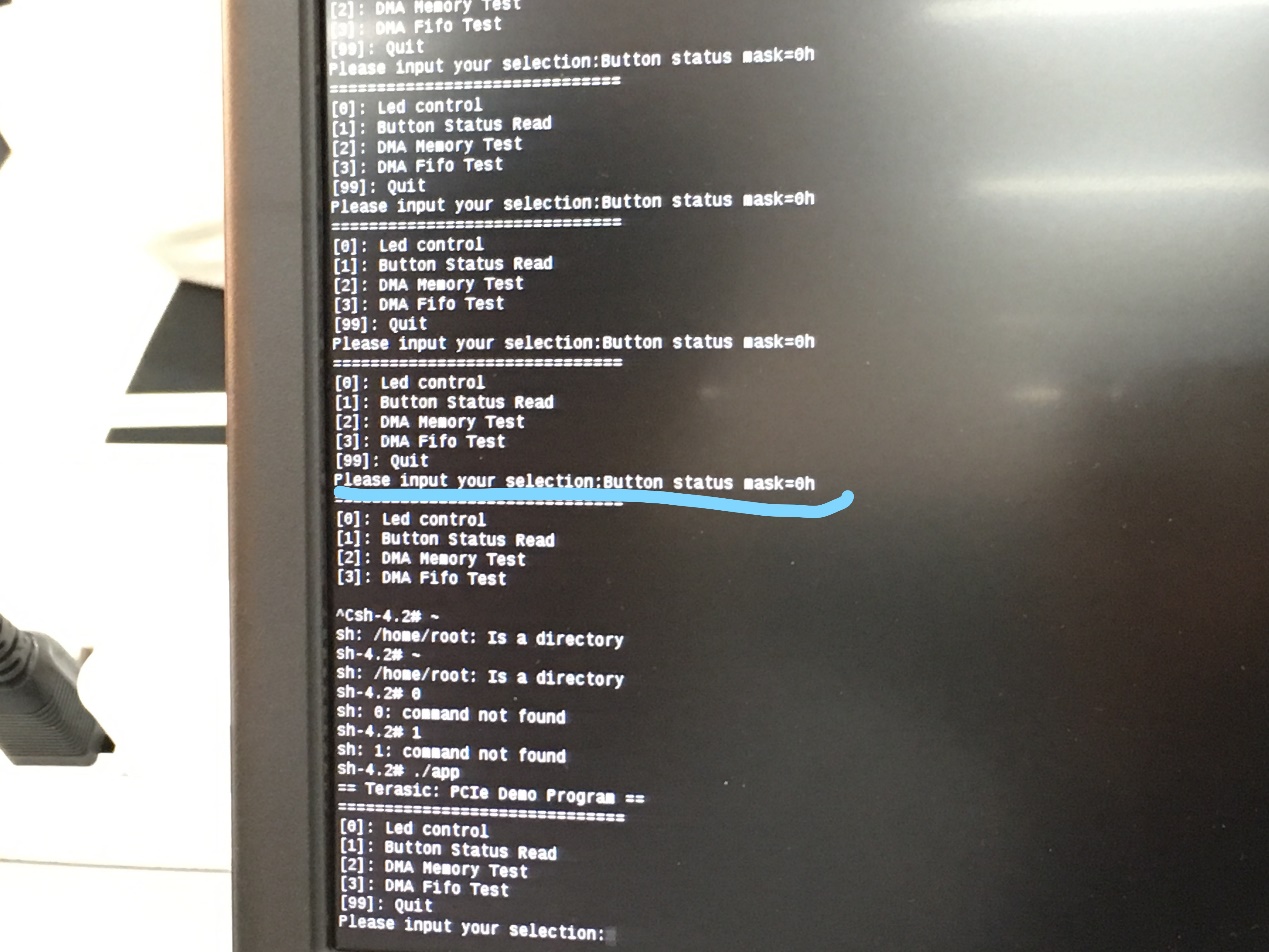


图3-13 测试PCIE基础演示功能2“按钮状态读取”时按钮状态一直是0h

# 第4章

## 4.1个人工作和学习过程

在实验过程中，我主要负责与朱威浦同学合作完成在DE2i-150实验板下的实验，包括PCIEDEMO实验和DE2i150平台下SOPC流水灯实验。最终PCIE DEMO部分的实验报告主体由我来完成，朱威浦同学做了一些修改和润色。

## 4.2 心得体会

1. DE2i-150的内部结构

回到DE2i-150板子的内部结构上来，我从网上找到的一张图清晰地说明了板子内部微处理器和FPGA与外设接口的连接关系。

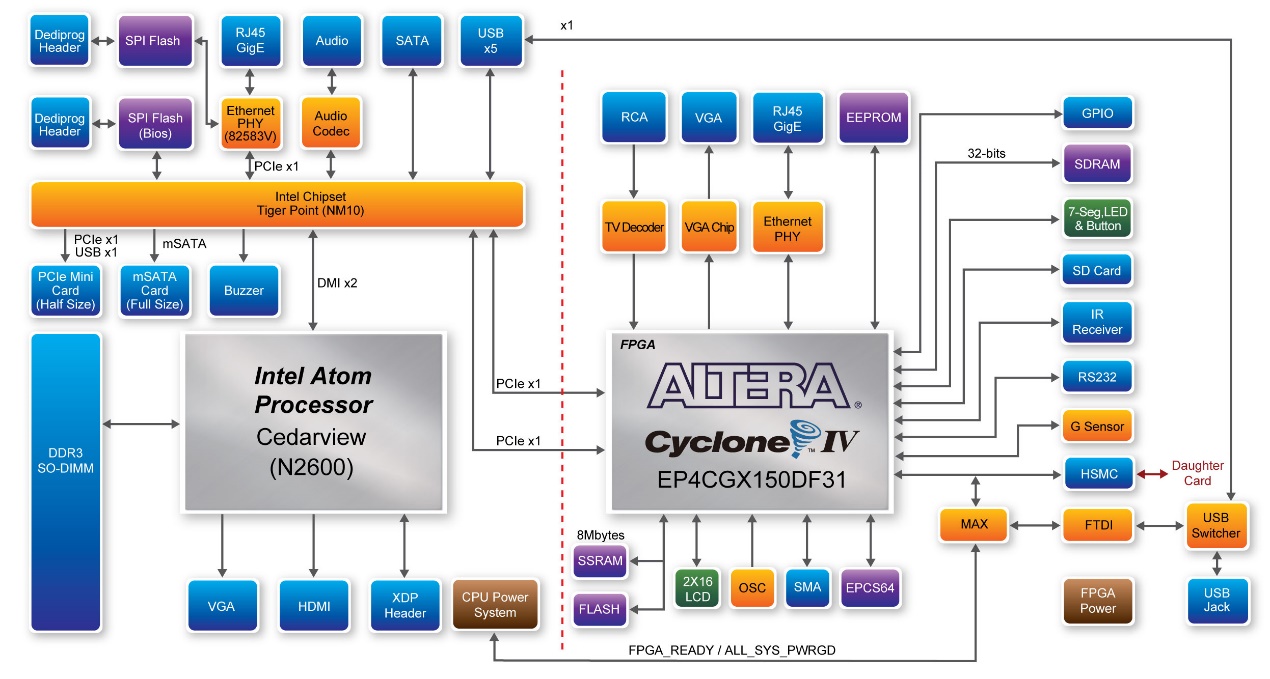


图3-12 DE2i-150板子的内部结构（来自google）

从图中我们可以看到FPGA连接了LED灯和按钮，而微处理器和FPGA之间则通过PCIE总线进行交互。我们的PCIE演示实验就是证明这个PCIE总线工作良好。

2. 细节是关键

通过PCIE实验，我明白了细节是关键，我们在往板子复制文件时没有注意是“PCIE显示演示”还是“PCIE基础演示”，另外在配置FPGA时主机也没有运行正确的批处理文件，即使运行了正确的批处理文件，也没有进行reboot，这直接导致了“PCIE基础演示”时一些功能测试失败。我们能从中吸取的教训就是细节是关键，即便是一个文件名错误也可能造成很大的影响。

1. 排除所有似是而非的出错可能，剩下的一定是真正的错因。

在进行PCIE演示实验时我们在安装yocto系统时遇到的问题一直无法解决，一开始我们推断是用来做启动盘的U盘里面有其他文件，影响了安装，于是我们就重新格式化了U盘，然而还是出现了同样的“searching for a hard drive”的问题。与此同时，别的组在装系统时也遇到了相同的问题。在上网查看了各种资料后，我们排除了一些其他的错误可能，在与其他做出来的小组的讨论过程中我们了解到可能是板子缺少硬盘的问题，虽然这个问题有些荒谬，但事实证明我们的板子确实没装硬盘，而这个硬盘里自带了一个yocto系统。这也再次印证了“真相只有一个”，逐一排除假错因后剩下的一定是真错因。

1. 进一步理解了在DE2i-150板上运行C程序的方法。

通过SOPC流水灯实验，我初步了解了用软核在开发板运行C程序的方法。如此一来，一个C程序既可以用硬核（yocto）跑也可以用软核跑，可以说十分灵活了。以后可以利用这一点开发出更多实用的程序。