***2020***



**系统能力培养综合实验 蓝牙小车方向 实验报告**

j0242087[1]

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1705 |
| 学 号： | U201714715 |
| 姓 名： | 吉伍子吉 |
| 组 名： | AE86 |
| 同组成员： | 蔡泽宇、陈郁峰、熊逸钦、 |
|  | 王明明、王乐寅 |
| 邮 件： | 2509322516@qq.com |
| 完成日期： | 2020年11月7日 |



目 录

[1 课程实验概述 3](#_Toc55493553)

[1.1 课程目标 3](#_Toc55493554)

[1.2 实验任务 4](#_Toc55493555)

[1.3 实验要求 5](#_Toc55493556)

[2 基于MIPSfpga的硬件平台的设计与实现 6](#_Toc55493557)

[2.1 设计目标 6](#_Toc55493558)

[2.2 环境配置等准备工作 6](#_Toc55493559)

[2.3 基于MIPSfpga的硬件平台搭建 7](#_Toc55493560)

[2.4 自定制接口模块的设计 10](#_Toc55493561)

[2.5 MIPSfpga硬件平台的中断实现 12](#_Toc55493562)

[2.6 遇到的问题及解决方法 14](#_Toc55493563)

[2.7 结论及改进设想 15](#_Toc55493564)

[3 操作系统的设计与实现 17](#_Toc55493565)

[3.1 设计目标 17](#_Toc55493566)

[3.2 HOS-MIPS操作系统的构建与运行 17](#_Toc55493567)

[3.3 HOS-MIPS集成开发调试环境安装 19](#_Toc55493568)

[3.4 从内核到应用 20](#_Toc55493569)

[3.5 遇到的问题及解决方法 25](#_Toc55493570)

[3.6 结论及改进设想 26](#_Toc55493571)

[4 应用部分的设计与实现 27](#_Toc55493572)

[4.1 设计目标 27](#_Toc55493573)

[4.2 蓝牙模块设计与实现 27](#_Toc55493574)

[4.3 遇到的问题 28](#_Toc55493575)

[5 总结与心得 29](#_Toc55493576)

[5.1 个人总结 29](#_Toc55493577)

[5.2 实验心得 29](#_Toc55493578)

[5.3 意见和建议 29](#_Toc55493579)

[致谢 30](#_Toc55493580)

[参考文献 31](#_Toc55493581)

# 课程实验概述

## 课程目标

计算机系统能力是指对软硬件系统深刻理解，具有系统层面认知和设计能力，能对软硬件功能进行合理划分、对系统不同层次进行抽象封装、对系统整体进行性能分析和调优、调试，以及根据不同的应用合理构建系统框架等能力。

本课程服务于信息产业自主可控的国家战略需求。我国正在实施“中国制造2025”、“互联网+”、“网络强国”等重大战略，目的在于突破核心关键技术，构筑先发优势，使我国在未来全球创新生态系统中能占据战略制高点。对计算机专业人才的知识结构与能力结构的深度、广度以及信息技术与其他产业融合创新能力都提出了更高要求。

具体而言，课程的目标如下几个方面：

1. 基于CPU+FPGA的SoC技术，采用IP集成的方法，搭建一个基于FPGA开发板的蓝牙小车，蓝牙小车运行自主的Hos操作系统，在此基础上开发具有可展示度、具有新颖性和创新性的蓝牙小车应用系统；
2. 根据蓝牙小车的设计需求，通过查阅资料和文献阅读，认真调研、分析，然后确定蓝牙小车的总体目标和设计方案，最终的设计方案除了能够展示蓝牙小车的基本功能外，还要具有可展示度、新颖性和创新性；
3. 能根据蓝牙小车的总体目标和设计方案，选择正确的设计与实现路线，设计实验方案对实现过程的正确性进行验证；
4. 能对蓝牙小车设计与实现过程中的测试结果进行理论分析，对实验现象进行解释，通过信息综合得到合理有效的结论；
5. 熟悉硬件、软件设计主流工具（EDA软件、FPGA开发平台、操作系统和软件开发调试环境等）的功能、特点及使用方法，掌握基于主流设计工具进行软硬件协同设计开发的流程和方法；能根据功能验证的需要，设计仿真脚本，掌握FPGA仿真波形、EDA软件资源、操作系统、专用硬件模块等数据收集、分析与处理的方法，通过数据分析验证所设计硬件、软件及系统；同时理解硬件仿真、操作系统和软件测试的局限性；
6. 能根据设计任务要求组建团队，成员分工明确、合理。能理解多学科背景下的团队中各角色的定位与责任，能够胜任个人承担的角色任务，并能与团队其他成员有效沟通；
7. 掌握计算机工程项目管理的基本方法和技术，在软硬件开发过程中对各个功能模块和系统集成的版本进行管理和控制，并运用项目管理工具完成项目决策和管理；
8. 对蓝牙小车的开发、设计过程进行科学化管理，除考察任务的完成度外，还要进行工程化考核，即考核系统的性能、成本、对资源的消耗、稳定性、功耗，以及开发的应用系统的创新性和新颖性等；
9. 能设计开发全新的软硬件功能模块并集成到系统中去，这需要通过不断学习掌握新技术和新方法才能做，从而锻炼自主学习的能力。

## 实验任务

整个实验由三大部分组成，分别是硬件部分的设计与实现、操作系统部分的设计与实现和应用部分的设计与实现。

1. 硬件设计

* Lab1：基于MIPSfpga的硬件平台搭建及其测试

1. 硬件平台搭建的实践准备
2. 基于MIPSfpga的硬件平台搭建
3. 自定制接口模块的设计
4. MIPSfpga硬件平台的中断
5. 操作系统设计

* Lab2\_1：HOS-MIPS操作系统的构建与运行
* Lab2\_2：HOS-MIPS集成开发调试环境安装
* Lab2\_3：从内核到应用

1. 应用设计

* Lab3\_1：基于MIPSfpga的蓝牙小车硬件设计与实现
* Lab3\_2：蓝牙小车操作系统、驱动及应用软件设计开发
* Lab3\_3：蓝牙小车应用扩展和创新

## 实验要求

综合实践步骤如下：

1. MIPSfpga处理器硬件平台的搭建
2. 完成Hos操作系统的实践
3. 手机蓝牙控制小车但没有操作系统
4. 基于操作系统的手机蓝牙控制小车
5. 蓝牙小车扩展功能

具体而言，首先需要搭建硬件平台的开发环境和调试环境，然后在硬件平台设计工具Vivado中对基于MIPSfpga CPU的硬件系统进行设计，并烧入开发板中运行测试程序进行验证。

然后在该硬件平台上设计实现自己定制的IP接口模块。

之后为硬件平台增加中断机制的实现，硬件方面添加IP模块和修改接线，软件方面编写中断测试程序。

接下来进入HOS-MIPS操作系统设计阶段。先在模拟Linux环境下使用gcc工具编译HOS-MIPS操作系统，生成操作系统镜像，然后将其加载到之前设计实现的MIPSfpga硬件平台上运行。

再在VSCode中为HOS-MIPS源码在硬件平台上运行时的调试过程搭建集成开发调试环境。

然后掌握在HOS-MIPS中添加应用和添加系统调用的方法，在HOS-MIPS中添加“Hello World”应用程序、“Hello World”系统调用以及“Free Pages”系统调用。

然后进入应用设计阶段。先在之前的软硬件平台基础上添加蓝牙模块和马达驱动模块。

然后在HOS-MIPS操作系统中编写程序来实现小车的控制。

最后自行扩展小车功能。

# 基于MIPSfpga的硬件平台的设计与实现

## 设计目标

（1）安装并测试相关的开发软件和工具，并初步熟悉搭建基于MIPSfpga处理器的硬件平台的开发环境。

（2）利用Vivado IP集成方法搭建一个基于MIPSfpga处理器，并通过AXI4总线连接处理器各个部件的简单嵌入式硬件平台系统。该系统主要包括MIPSfpga 处理器、AXI4总线、BRAM内存以及GPIO、UART等基于AXI4总线接口规范的外设模块。然后通过编写MIPS汇编程序或C语言程序对外设进行操作演示。

（3）在深入理解MIPSfpga处理器和 AXI 总线协议、以及掌握了基于Vivado IP集成方法搭建MIPSfpga处理器系统的流程的基础上，自己动手实现一个基于 AXI 总线接口标准的外设模块，并将该模块添加到MIPSfpga处理器系统中。

（4）通过实现中断功能，对MIPSfpga处理器的中断机制以及MIPSfpga处理器硬件平台的中断功能实现进行实践。

## 环境配置等准备工作

环境配置部分需要安装的工具包括Xilinx Vivado、OpenOCD嵌入式调试工具和MIPS MTI交叉编译开发环境。

### 安装Xilinx Vivado 2015.2

这一步按照指导书的内容，在网络上找到Vivado 2015.2版本的安装包下载之后解压，并运行xsetup.exe开始安装过程。

安装完成后找到桌面上的Vivado图标双击打开发现有Visual C++ 2012文件缺失的情况，使用DirectX修复工具DirectX Repair修复后重启即可正常进入Vivado 2015.2的主界面。

### 安装OpenOCD嵌入式调试工具

这里使用老师提供的资料包中的OpenOCD-0.9.2-Installer.exe进行安装，按照实验指导书的步骤完成安装流程。

安装完毕后在其安装目录找到zadig\_2.1.2.exe，然后将MIPSfpga的调试器通过USB连接到主机，并先选择BUSBLASTERv3c（Interface0）进行调试器驱动程序安装，完成后再选择Interface1进行相同的操作。

### 安装MIPS MTI交叉编译开发环境

使用在线安装程序进行Codescape MIPS SDK的安装，按步骤完成安装流程。

安装完毕后，将老师提供的资料包中的工具链复制到OpenOCD安装目录下的子文件夹，命名为Toolchains。

然后将工具链添加到系统环境变量，添加完毕后系统环境变量如下图2.1所示：



图2.1 配置环境后的系统环境变量

## 基于MIPSfpga的硬件平台搭建

### 搭建过程

首先按照指导书的内容搭建一个基于MIPSfpga处理器的最简系统。需要注意由于我在本次实验使用的是2015.2版本的Vivado，因此先要更新Vivado的AHB-Lite\_to\_AXI桥IP，将该IP的vhdl文件夹用网盘下载的ahblite\_axi\_bridge\_v3\_0目录中的ahblite\_axi\_bridge\_v3\_0/hdl/src/vhdl文件夹替换。

搭建最简系统的过程主要包括新建工程、新建Block Design、添加MIPSfpga处理器IP、添加AHB-Lite to AXIBridge模块、逐个添加AXI Interconnect、AXI GPIO、 AXI BRAM Controller和Block Memory Generator等IP模块、添加Utility Buffer IP模块、配置IP模块并连线、添加ram\_init.coe初始化文件、添加时钟IP模块、分配CPU总线设备的地址。完成上述过程之后就实现了一个最简的系统。

然后在最简系统的基础上进行扩展。这里添加了一个AXI UART16550 IP模块，用于串口通信，还添加了一个DDR2内存控制器。DDR2内存控制器的IP模块设置需要按照指导书的内容逐步实现，对于后续开发板支持HOS-MIPS操作系统的运行是很重要的。

搭建完成后的Block Design连接图如下图2.2所示：

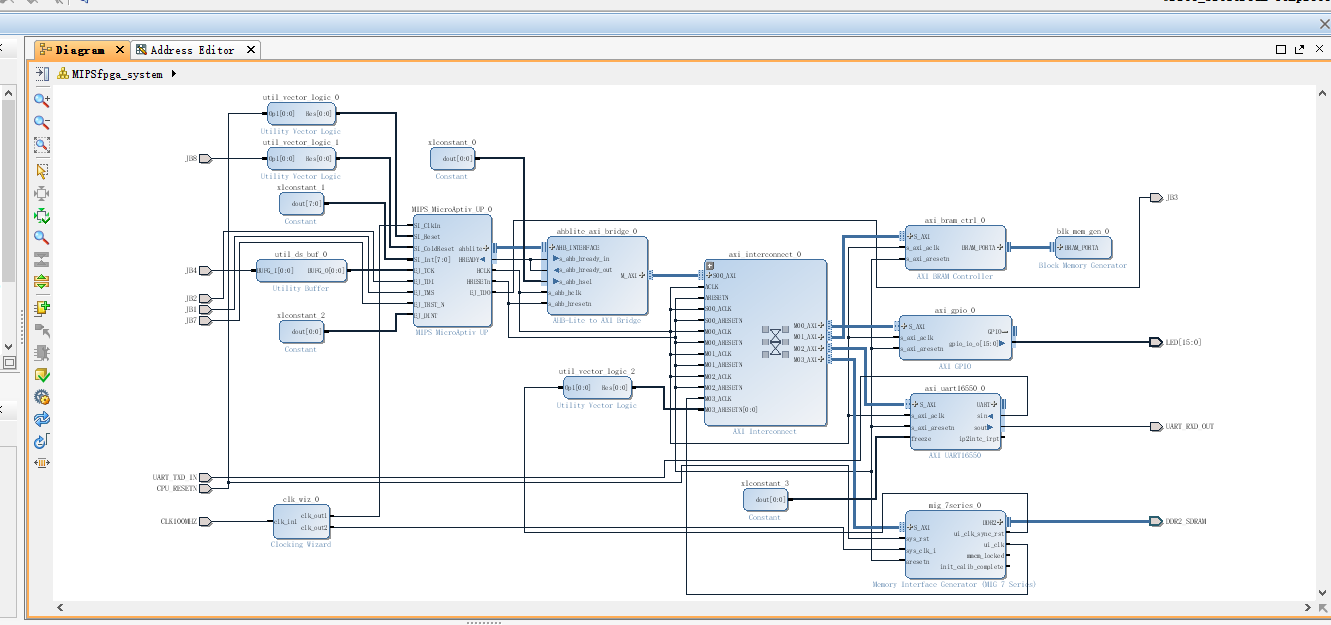


图2.2 搭建完成后的硬件平台

### 运行和调试

成功生成比特流之后，在主机上进入\_C目录下使用make命令对程序进行编译，生成elf可执行文件。

之后用USB线连接开发板和JTAG调试器，将比特流烧写到开发板中，然后用loadMIPSfpga.bat脚本在开发板上运行程序。同时打开Putty终端，用串口（在设备管理器查看COM编号，波特率设置为115200）连接，查看串口终端的输出情况。

测试结果如下图2.3所示：

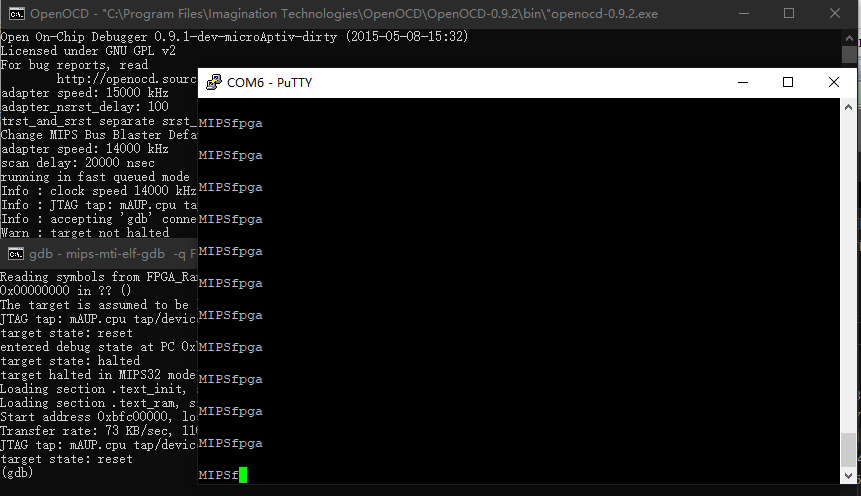


图2.3 硬件平台测试结果

开发板的状态如下图2.4所示：

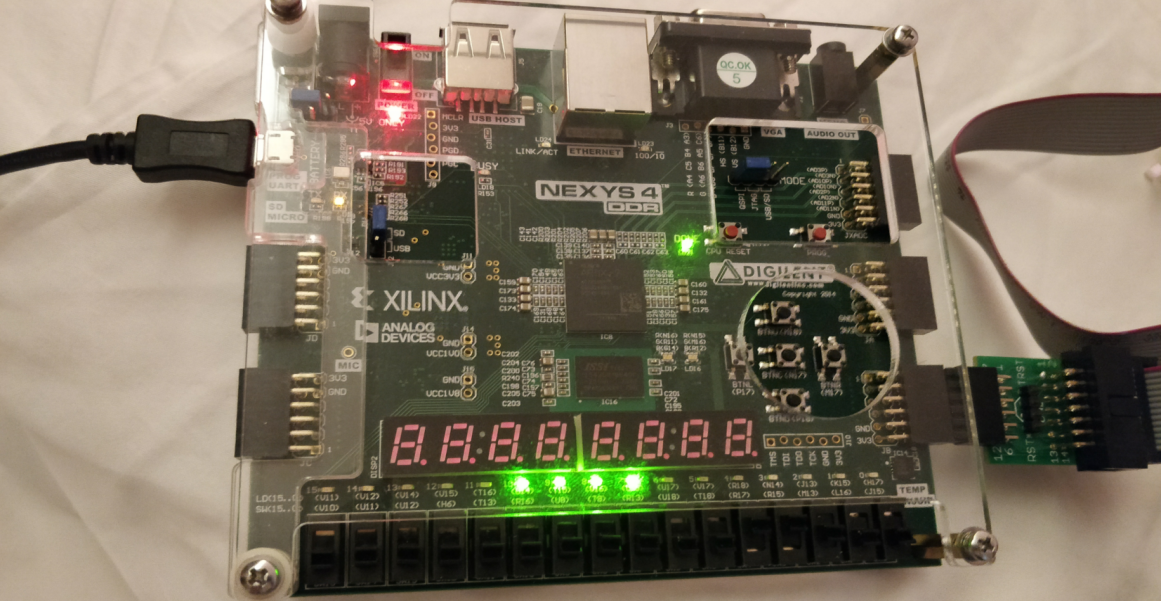


图2.4 搭建硬件平台后开发板状态

可以看到，Putty终端窗口中能正确从串口收到循环的打印信息，而且开发板的LED灯光也能正确点亮。说明正确搭建了MIPSfpga硬件平台。

## 自定制接口模块的设计

### 搭建过程

首先复制先前搭建好的硬件平台工程，然后使用Vivado打开，并打开Create and Package IP菜单构建一个新的自定义IP。

在构建新IP的工程中，使用Verilog语言编写该IP的设计文件，用来控制该IP模块的行为和功能。由于篇幅限制，此处不粘贴Verilog源码，可在工程目录中找到。

添加完设计文件后可以进行综合和仿真，确定无误之后对该IP模块使用Package IP进行封装。首先检查File Groups中的内容，需要把设计文件改为相对路径。然后在Ports and Interfaces选项中检查输入输出引脚是否正确设置。最后选择Click Review and Package选项，确认后选择Re-Package IP进行封装。

回到原工程后就可以在Block Design中使用Add IP添加自己自定义的IP模块到MIPSfpga处理器系统了，最后给该模块分配物理地址。

添加完自定义IP模块之后的Block Design如下图2.5所示：

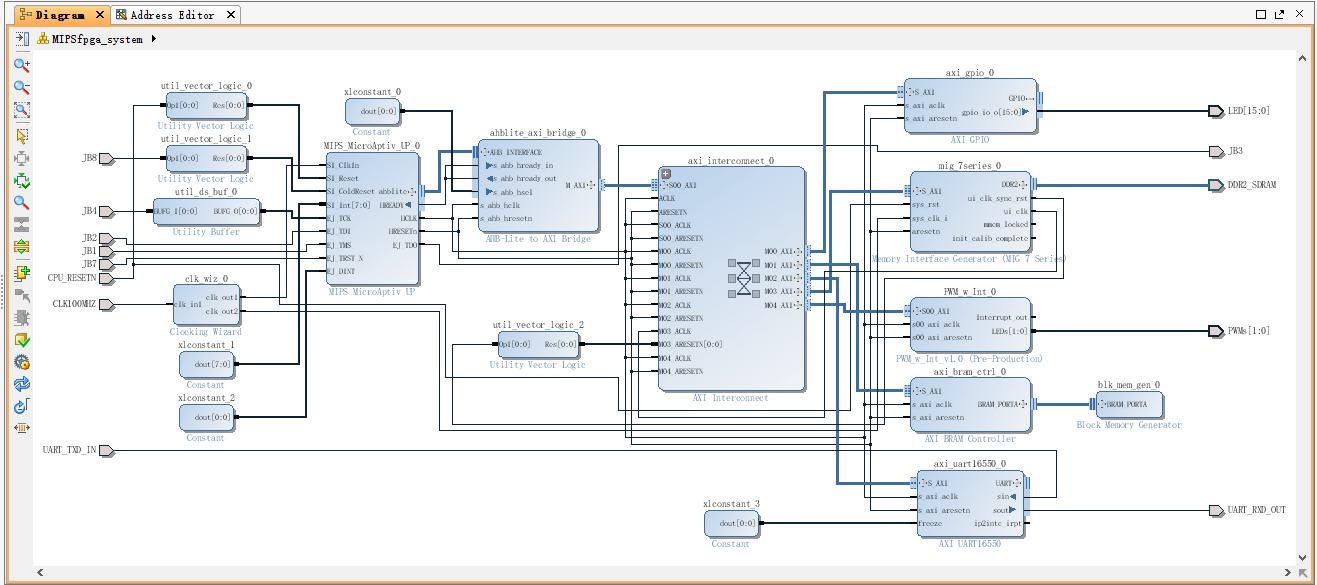


图2.5 添加自定义模块后的硬件平台

### 运行和调试

验证设计正确性并生成比特流之后，使用对应的\_C文件夹中的程序对硬件平台进行测试。

测试结果如下图2.6所示：

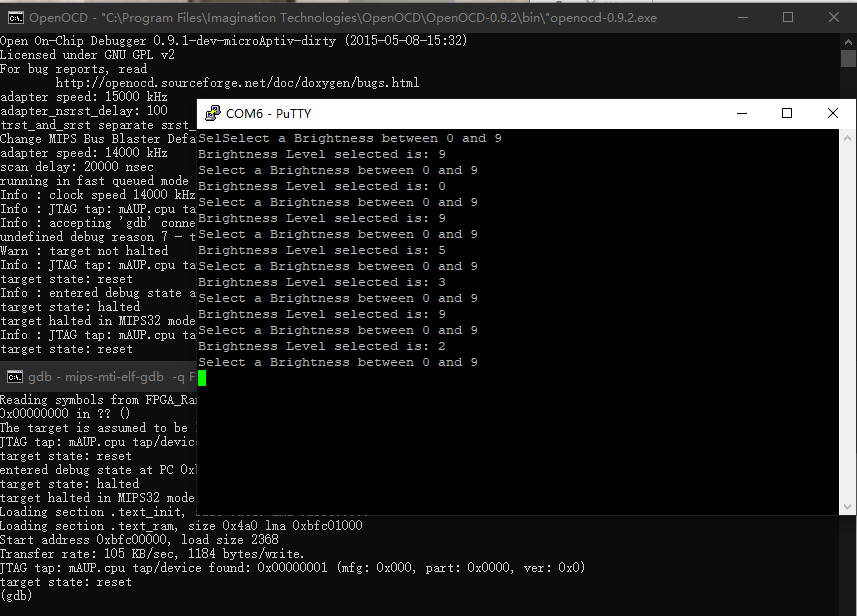


图2.6 添加自定义IP后的硬件平台测试结果

开发板的状态如下图2.7所示：

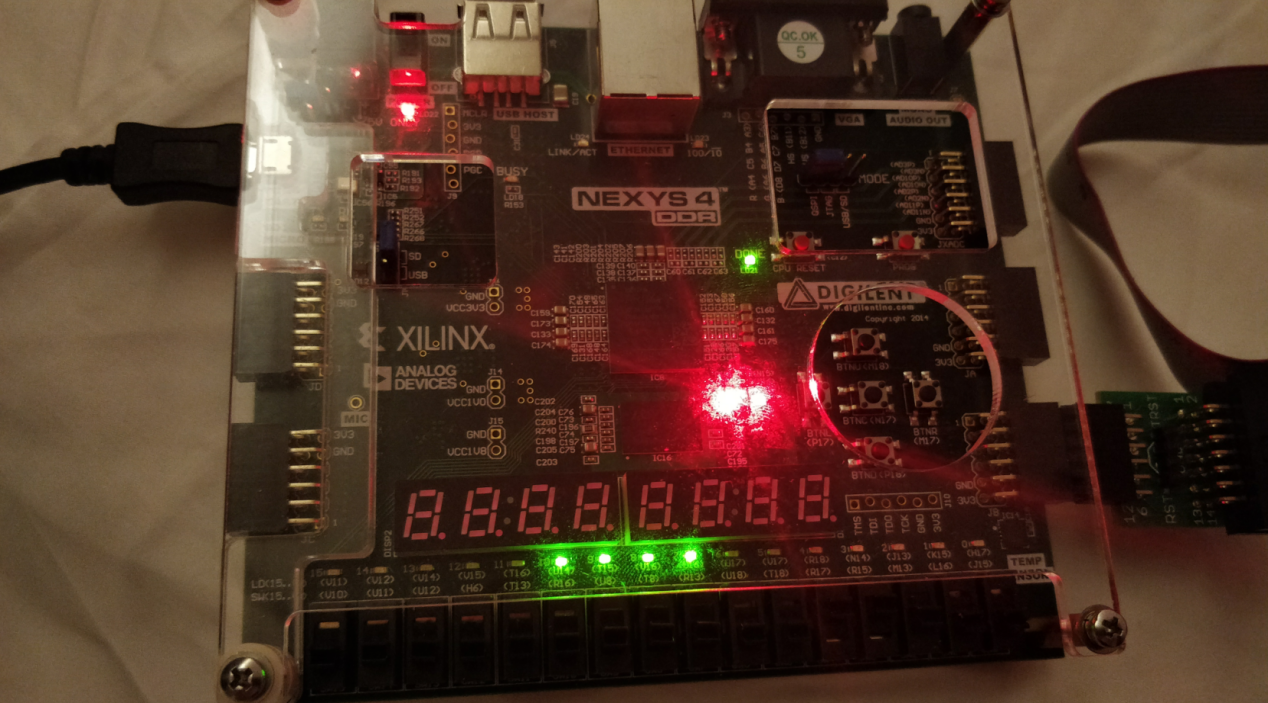


图2.7 搭建添加自定义IP后的硬件平台后的开发板状态

根据上述结果可以得出结论，开发板能正确给出询问信息，并能从串口读取亮度选择信息。说明正确实现了自定义IP模块的搭建。

## MIPSfpga硬件平台的中断实现

### 搭建过程

首先复制2.4节中搭建好的硬件平台工程，然后使用Vivado打开，将连接 MIPSfpga处理器SI\_INT[7:0]引脚的Constant模块断开，双击Constant模块并将其 ConstWidth改为1，再添加一个8位的Concat模块。然后将Concat模块的dout[7:0]引脚连接到MIPSfpga处理器的SI\_INT[7:0]引脚。最后将Concat模块中In4、In5 引脚外的其它引脚全部连接到前面修改的Constant模块。

完成后将PWM\_w\_Int模块的Interrupt\_out引脚连接到Concat模块的In4引脚，UART16550模块的ip2intc\_irpt引脚连接到In5引脚。完成连接后的Block Design如下图2.8所示：

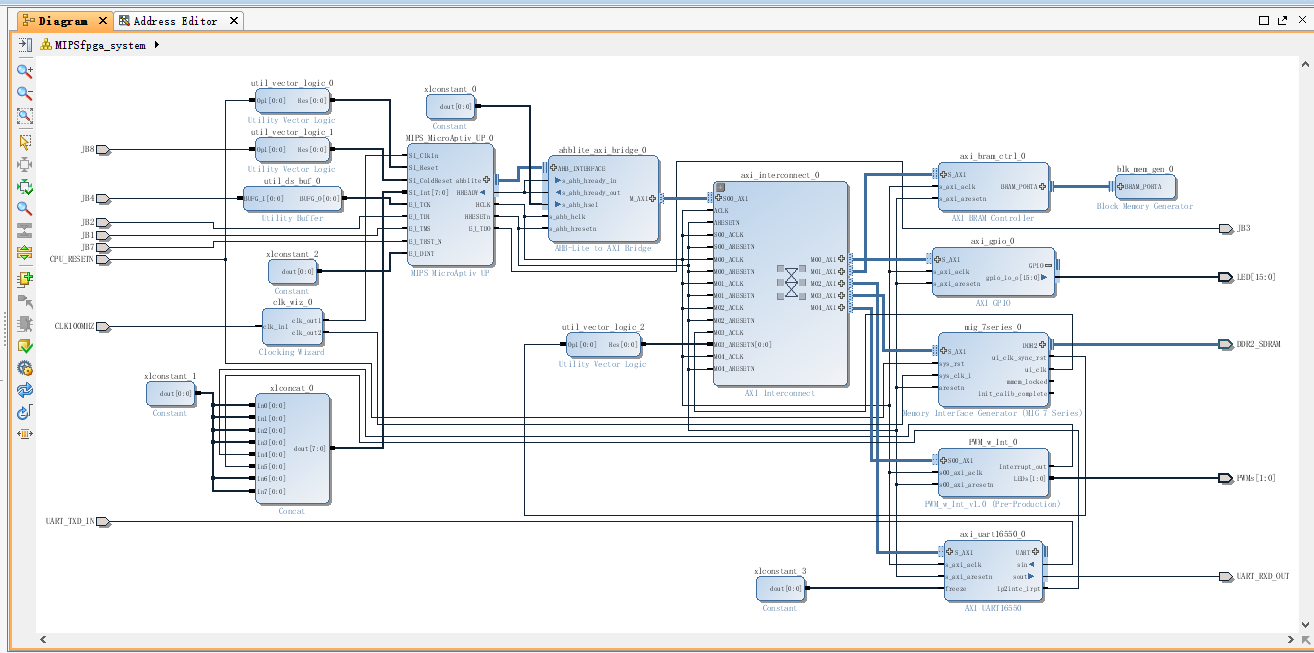


图2.8 支持中断的硬件平台

按照指导书的内容，继续使用外部中断控制器模式（EIC）实现中断。复制上述工程，然后添加AXI Interrupt Controller这一个IP模块作为中断控制器，将其连接到硬件平台。

然后根据指导书的内容，复制Constant模块和Concat模块，并修改连线。最后给中断控制器分配物理地址。

完成连接后的Block Design如下图2.9所示：

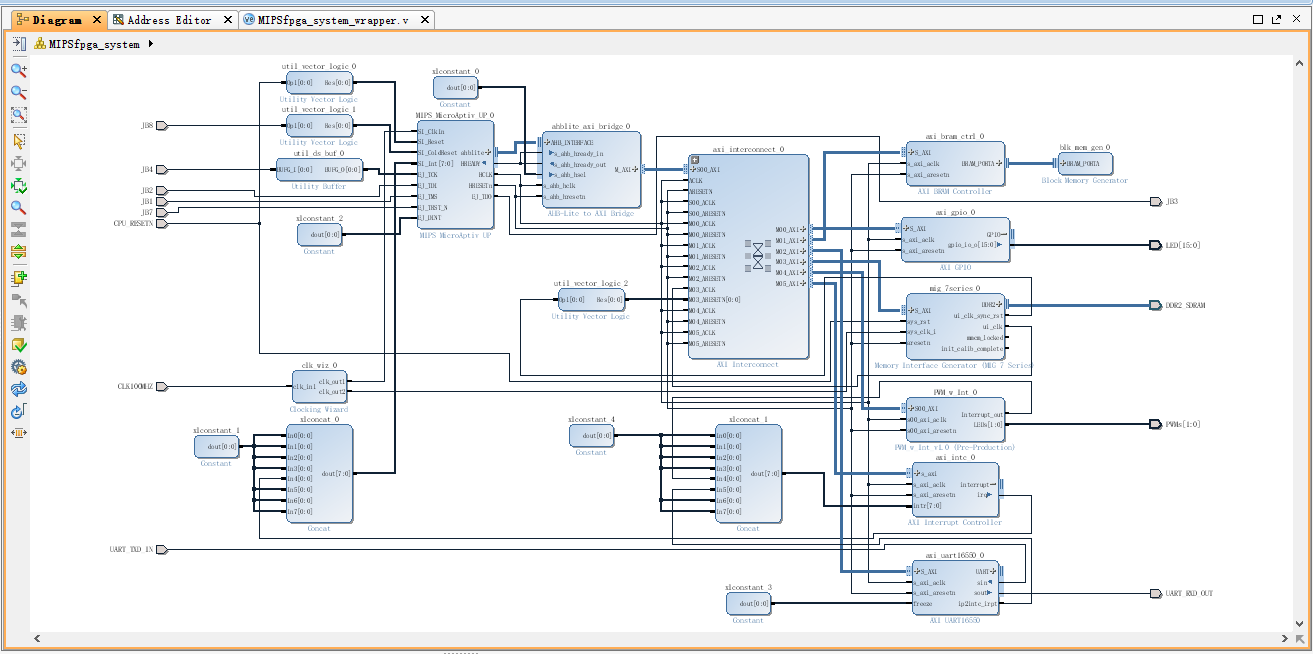


图2.9 支持外部中断控制器中断的硬件平台

### 运行和调试

验证设计正确性并生成比特流之后，使用对应的\_C文件夹中的程序对硬件平台进行测试。

测试结果如下图2.10所示：

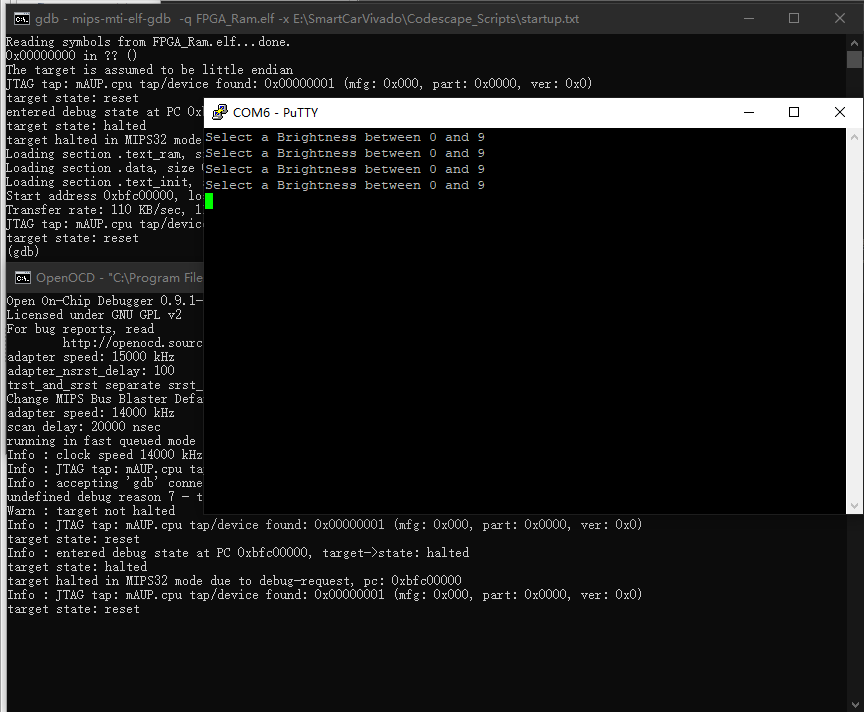


图2.10 支持中断的硬件平台测试结果

开发板的状态如下图2.11所示：

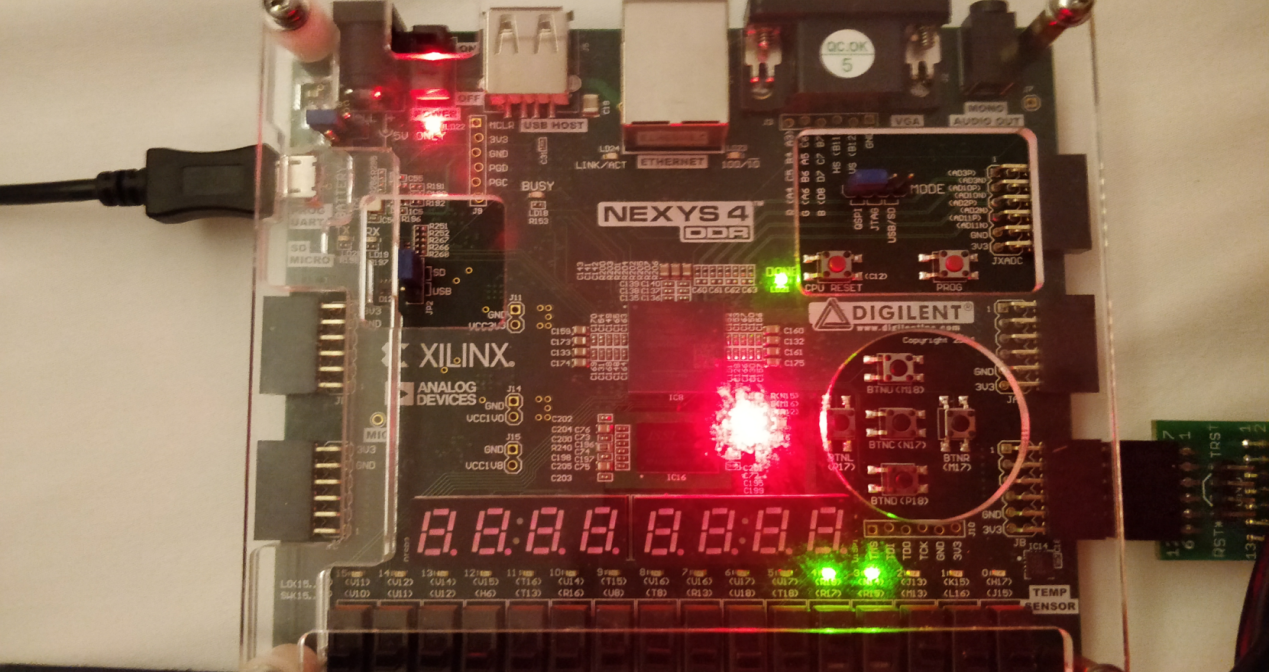


图2.11 搭建支持中断的硬件平台后开发板状态

根据上述结果可以得出结论，开发板能进入中断并正确给出询问信息，且能从串口读取亮度选择信息。说明正确实现了支持中断的硬件平台的搭建。

## 遇到的问题及解决方法

### Vivado无法启动的Visual C++ 2012问题

**问题描述：**

安装完成后找到桌面上的Vivado图标双击打开，发现有Visual C++ 2012文件缺失的情况，等待后弹窗提示程序响应超时。

**解决方法：**

使用DirectX修复工具DirectX Repair修复后重启主机，即可正常进入Vivado 2015.2的主界面。

### MIG内存控制器配置问题

**问题描述：**

在DDR2内存控制器IP的配置过程中，MIG 的配置界面读取Nexys4DDR\_for\_MIG.ucf文件时无法正常弹出选择文件路径的窗口，会弹窗提示报错，然后需要重新进入IP模块配置界面。

**解决方法：**

只能使用文本编辑器查看老师在资料包中提供的Nexys4DDR\_for\_MIG.ucf文件内容并手动输入到配置界面。需要注意的是，经过尝试后我发现只需要输入PinNumber列，不需要手动填写其他列的内容，其余内容都会由Vivado自动填充。输入完毕后点Validate验证输入正确性即可。

### 自定义IP时模块封装的问题

**问题描述：**

封装模块时在Package IP界面发现添加的PWM\_Controller\_Int.v文件是绝对路径，需要手动删除并重新添加。

此外，还在Ports and Interfaces页面中发现最终封装的IP会多出两个内部引脚。

**解决方法：**

具体是先右键将PWM\_Controller\_Int.v文件从File Group中remove，然后将该文件移到ip\_repo/hdl/目录下。

在Ports and Interfaces页面中需要手工加入引脚，就可以防止最终封装的IP会多出两个内部引脚。

## 结论及改进设想

### 结论

本次基于MIPSfpga的硬件平台的设计和实现部分实验中，我使用Vivado设计工具，按照实验指导书的内容从无到有搭建了自己的MIPSfpga处理器系统，并实践了该系统的扩展、自定义IP模块的添加以及中断机制的实现。整个实验使我理论上的知识和实践上的能力都得到了提高。

总而言之，本次实验对于MIPSfpga的硬件平台的设计和实现，对我来说是一个不小的挑战，完成的过程中也使我在理论和实践方面都有了非常大的收获。

### 改进设想

本次实验中实现的支持中断的MIPSfpga硬件平台在测试时发现一段时间过后会陷入串行通信接口之外的中断。后来查阅资料发现这是开发板陷入了定时器中断。

进一步的设想是修改\_C文件夹下的源代码，使得程序运行时能够规避掉定时器中断。

# 操作系统的设计与实现

## 设计目标

（1）学习在自己的个人电脑上安装构建Hos-mips 操作系统的环境，以及将所生成的镜像下载到 MIPSfpga 硬件平台、并将其运行起来的方法。

（2）安装Hos-mips的集成开发环境VSCode，使用VSCode打开 Hos-mips 源代码，并在该环境中构建内核、执行并调试。

（3）回顾操作系统特权级知识，掌握Hos-mips操作系统中从应用层到系统内核层的完整调用路径，在Hos-mips内核中添加系统调用，使得应用层能够触发操作系统特权级的动作。

## HOS-MIPS操作系统的构建与运行

### 环境配置

首先需要安装Cygwin这样一个类Linux环境，可以利用它使用Linux下的make、gcc、perl这些工具。安装后将Cygwin加入系统环境变量。

完成后下载Hos-mips源码，这里直接使用老师提供的资料包中的源码压缩包解压到我主机的D:\Hos\hos-mips目录下。

完成上述步骤后就完成了环境的配置。

### 构建并运行

首先打开Cygwin，使用cd /cygdrive/d/Hos/hos-mips进入Hos-mips源代码目录，然后输入make命令对系统进行build。由于之前的准备工作比较细致地完成了，因此这个过程中没有出现任何报错，顺利地完成了Hos-mips系统的build过程。

构建完成Hos-mips之后会产生一个系统镜像，成功生成后如下图3.1所示：

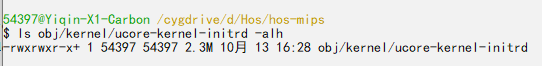


图3.1 Hos-mips系统镜像成功生成

成功生成系统镜像文件之后，就可以在硬件平台上运行Hos-mips操作系统了。

首先需要将开发板连接到主机上，然后打开Vivado，进入Hardware Manager，再使用Auto Connect连接之后就可选择之前生成的比特流文件烧到开发板中。

然后需要修改Hos-mips根目录下的run.bat文件，修改完后如下图3.2所示：

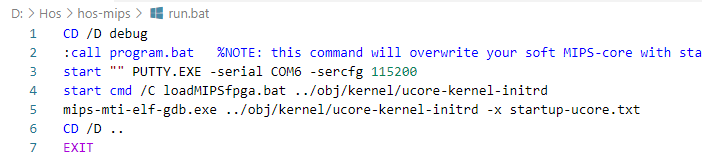


图3.2 修改完成后的run.bat文件内容

这里主要修改了调用Putty读取串口信息时，Putty所连接的设备端口号。通过在设备管理器中查看串行设备端口号可知开发板在我的主机上所分配的端口号为COM6，因此将run.bat中的参数修改为COM6。

修改完毕后，打开Cygwin，进入Hos-mips的根目录，直接运行run.bat，就可以将Hos-mips加载到开发板上运行了。

进入系统后运行时的效果如下图3.3所示：

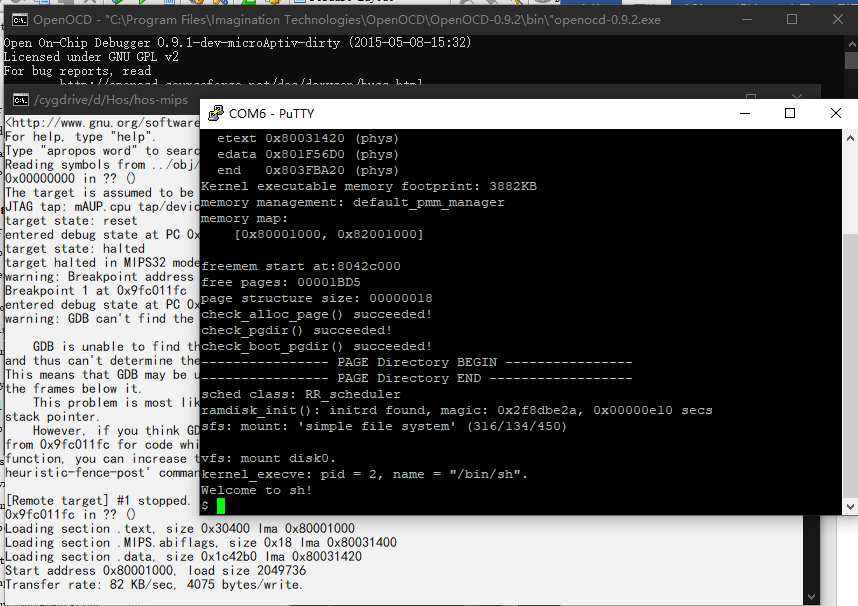


图3.3 在开发板上运行Hos-mips时效果

到这里就完成了Hos-mips系统的构建和运行过程。

## HOS-MIPS集成开发调试环境安装

这一部分需要使用VSCode安装Native Debug插件实现。

由于我的主机上已经安装VSCode，因此只需要在插件安装界面安装Native Debug即可。

在VSCode中使用“打开文件夹”按钮打开Hos-mips系统源代码的根目录，然后随便打开一个C源代码文件（这里和指导书中保持一致，打开了kern-ucore子目录下的init.c文件）并设置断点。需要注意的是在进行调试之前需要关闭之前打开的所有窗口，包括Cygwin、Putty和OpenOCD的窗口。

最后需要修改一下之前提到的run.bat文件，把文件的第五行注释掉，如下图3.4所示：

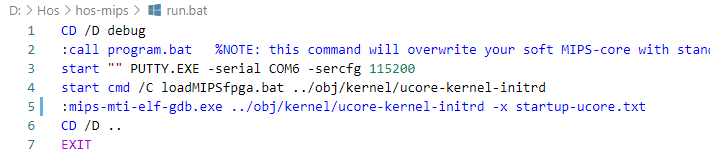


图3.4 修改后的run.bat文件

这一步关闭了run.bat对mips-mti-elf-gdb.exe的调用，因为VSCode将会对它进行调用。

下面回到VSCode，按F5进行调试。需要注意的是在这个过程中开发板需要始终连接到主机保持上电状态，且比特流已经成功加载进开发板中。

在加载完成后，VSCode将会尝试将Hos-mips镜像加载到MIPSfpga系统的内存中，并加载和启动操作系统。由于之前设置了断点，所以会在断点处停止。

需要说明的是，在我的主机环境中按指导书中的内容直接进行调试会报错，需要修改一下launch.json的内容，具体请见本章3.5节“遇到的问题及解决方法”这一部分的内容。

调试过程中在断点处停止的状态如下图3.5所示：

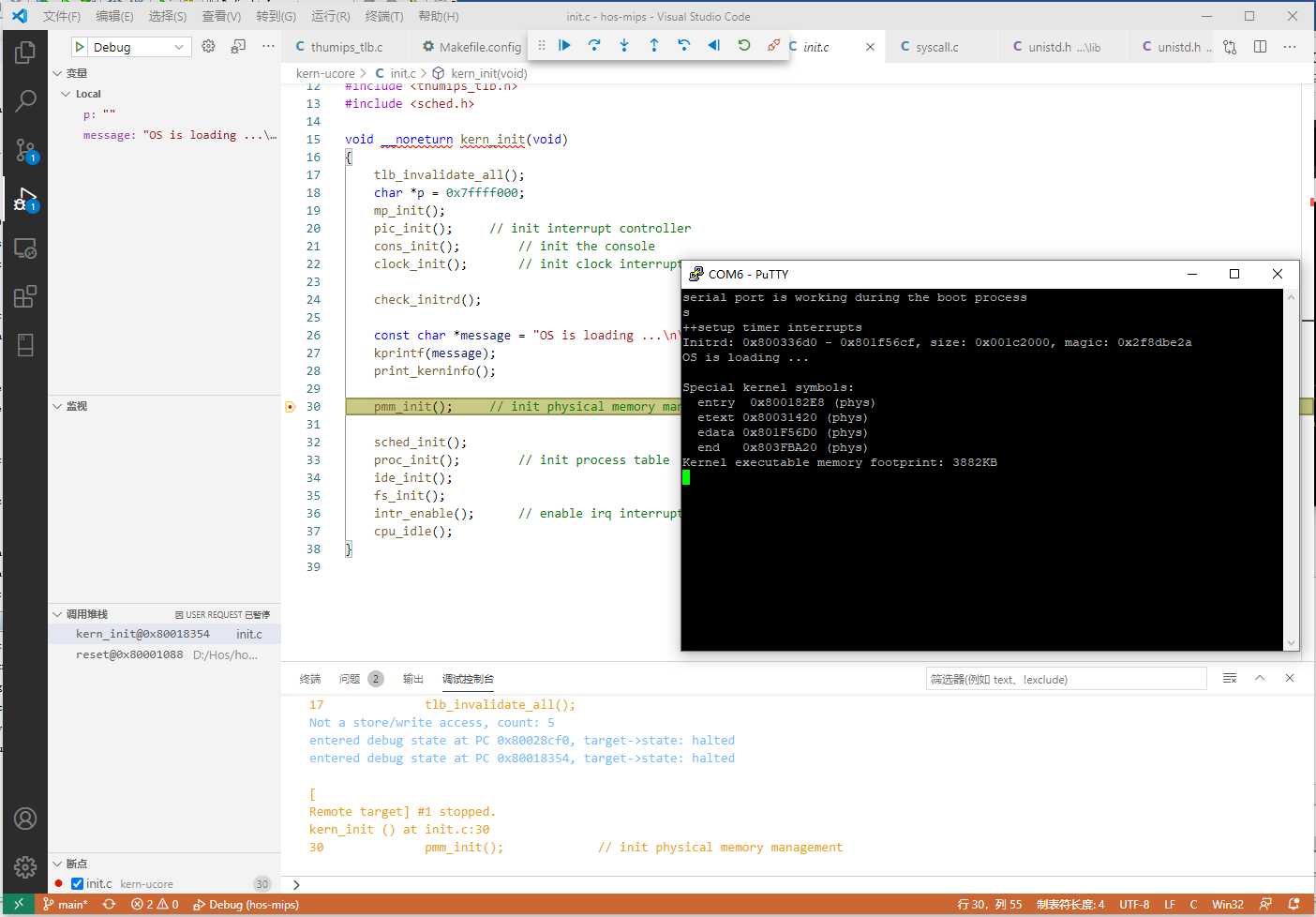


图3.5 使用VSCode断点调试

由上图可见，能够正常使用VSCode对Hos-mips进行断点调试。

## 从内核到应用

这一部分实践了在Hos-mips中添加应用、添加系统调用等多种方式来在Hos-mips上运行自己编写的程序。首先需要添加一个“Hello World”应用，然后尝试添加一个系统调用，最后尝试综合应用和系统调用的知识，打印出系统的空闲页面数量。

### 添加“Hello World”应用

阅读指导书后我明白了，Hos-mips操作系统的构建过程中会进入user/user-ucore目录下，用来构建obj/user-ucore/bin下面的一系列应用程序。

我还阅读到这些应用程序名字的列表在Hos-mips根目录下的Makefile.config这个文件中定义。

根据上面的知识，添加一个应用只需要在user/user-ucore目录下添加自己的C语言代码源文件，然后修改Makefile.config的内容，将自己定义的应用程序名加入到列表中。

具体就是在user/user-ucore下添加一个hello.c源文件，在其中编写自己的源程序，如下图3.6所示：

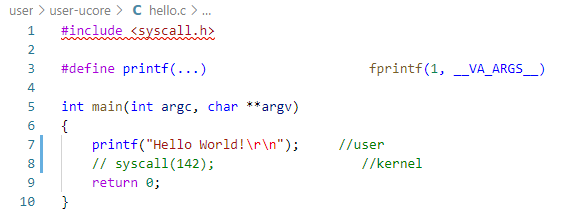


图3.6 hello.c源文件内容

然后修改Makefile.config，将应用名字hello添加到里面去，如下图3.7所示：

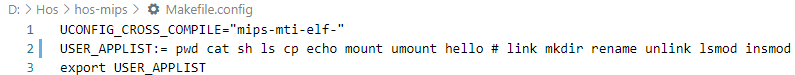


图3.7 在Makefile.config中添加hello字段

最后重新编译Hos-mips并加载上板，运行hello应用，结果如下图3.8所示：

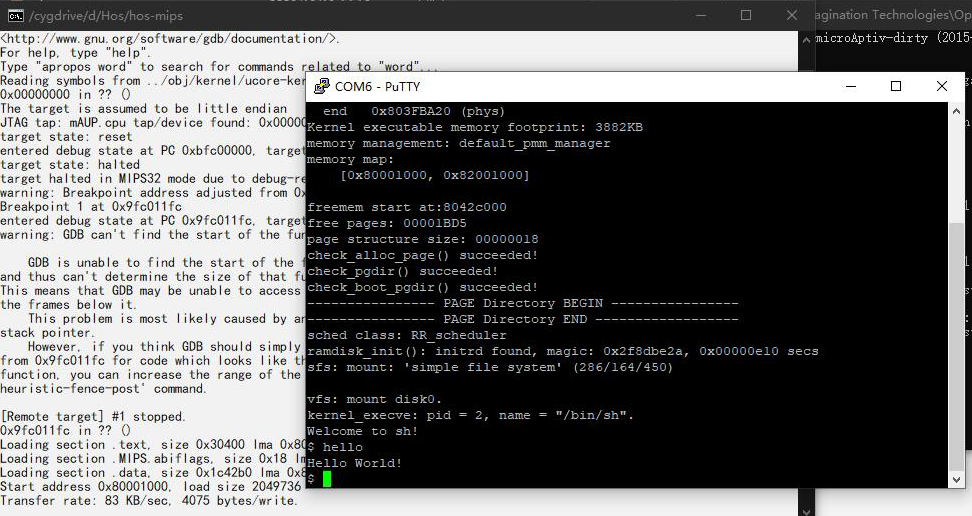


图3.8 添加hello应用程序后的效果

可以看到上图中在Hos-mips系统输入hello指令后，在窗口中打印出了“Hello World！”字样，实现了hello应用的添加。

### 添加系统调用

按照指导书中的内容，先修改文件kern-ucore/syscall.c，在其中仿照其他系统调用的形式添加一个名为khello的系统调用，如下图3.9所示：

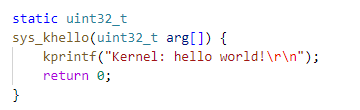


图3.9 修改syscall.c源文件

除此之外，还需要在syscall.c源文件中syscalls这个函数数组中添加成员SYS\_khello来对应sys\_khello这个函数。

然后修改kern-ucore/include/lib/unistd.h和user/include/unistd.h以及user/include/syscall.h这三个文件中的内容，具体修改内容如下：

* kern-ucore/include/lib/unistd.h



图3.10 修改kern-ucore/include/lib/unistd.h文件

* user/include/unistd.h



图3.11 修改user/include/unistd.h文件

* user/include/syscall.h



图3.12 修改user/include/syscall.h文件

然后修改user/user-ucore/hello.c的内容，把应用程序调用printf改为一个系统调用，调用号使用上述设置的142，如下图3.13所示：

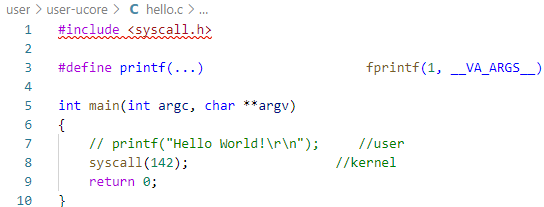


图3.13 修改hello.c源文件的内容

重新编译Hos-mips并上板运行，结果如下图3.14所示：

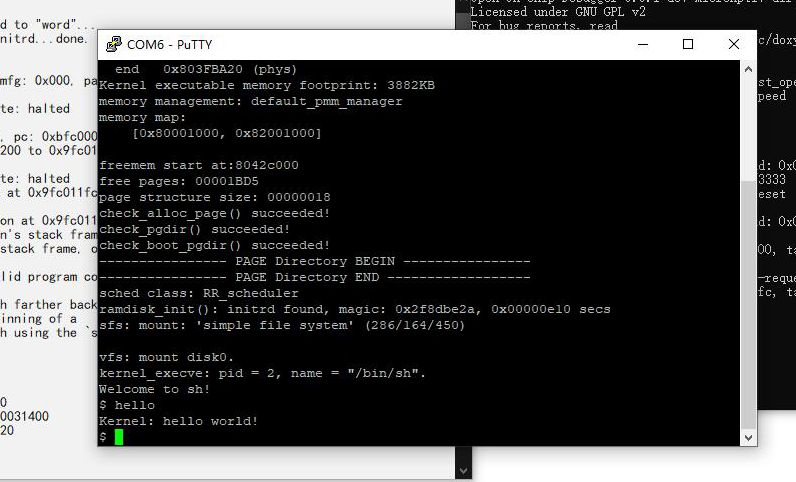


图3.14 添加系统调用后的运行效果

可以看到上图中在Hos-mips系统输入hello指令后，在窗口中打印出了“Kernel：hello world！”字样，说明成功使用系统调用的方式跨越了用户态和内核态。

### 打印空闲页面数量

最后需要综合上面的方法，实现打印出操作系统中空闲页面的数量这一功能。

这一步骤使用增加系统调用和增加用户应用两种方式结合的办法进行实现。

根据指导书的提示，阅读kern-ucore/pmm.c的源码之后发现有一个nr\_free\_pages的函数如下图3.15所示：

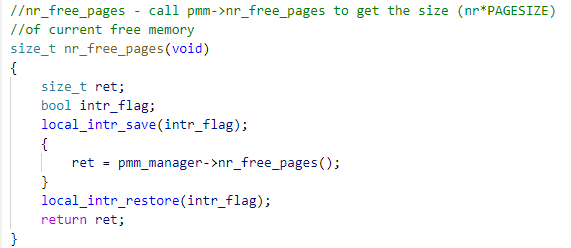


图3.15 源码中的nr\_free\_pages函数

阅读源码注释之后可以发现，这个函数是内核函数，功能是获取当前空闲内存的大小，符合程序的功能设计要求，因此可以新增一个系统调用来调用这个函数在特权态获取空闲页面数量并打印。

增加系统调用的方法与之前相同，将其命名为kfree，系统调用号设置为143。

再用编写用户程序的相同方法增加一个名为free的用户程序，user/user-ucore/free.c的内容如下图3.16所示：

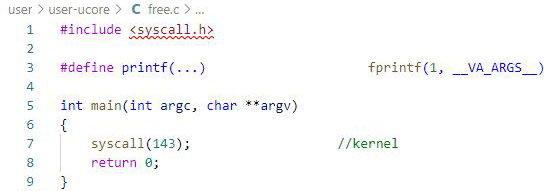


图3.16 user/user-ucore/free.c的内容

修改完毕之后重新编译Hos-mips并上板运行，结果如下图3.17所示：

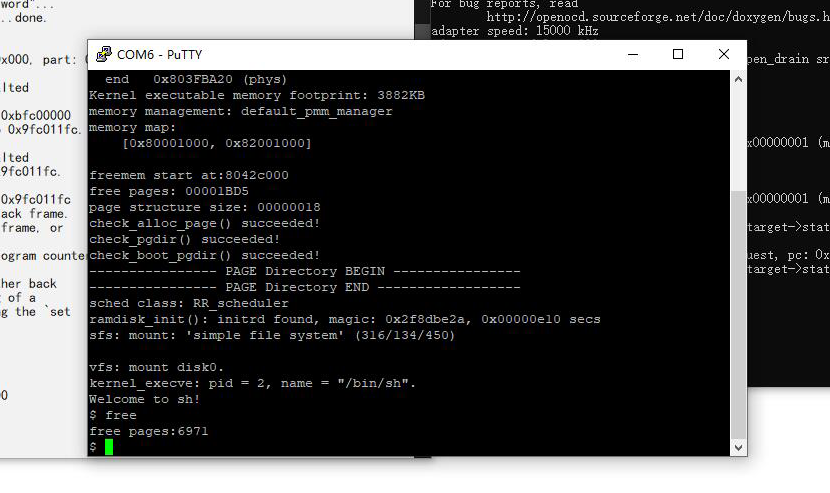


图3.17 实现打印空闲页面数量功能后的运行效果

根据上图可知，在系统的终端输入free命令后会调用free这个应用程序运行，而这个应用程序的功能代码中使用syscall函数实现了第143号系统调用（也就是新增的kfree系统调用），成功在特权态打印了空闲页面数量信息。

## 遇到的问题及解决方法

### 搭建调试环境时报“无法找到文件”错误

**问题描述：**

在VSCode中安装Native Debug并按下F5进行调试时，报错提示无法找到mips-mti-gde-gdb.exe文件，根据提示找到Hos-mips系统根目录下的launch.json文件中gdbpath这个键的值被设置为mips-mti-gde-gdb.exe，但我在工具链Path的C:\Program Files\Imagination Technologies\Toolchains\mips-mti-elf\2017.10-08\bin目录下并未找到mips-mti-gde-gdb.exe可执行文件，所以报错。

**解决方法：**

结合之前的经验，猜测这里需要使用的gdb工具是mips-mti-elf-gdb.exe，且在Path中的确能够找到mips-mti-elf-gdb.exe，因此修改launch.json内容，如下图3.18所示：



图3.18 修改后的launch.json内容

修改之后再按F5进行VSCode辅助调试就可以正常加载系统并停止在断点处。

## 结论及改进设想

### 结论

本次实现Hos-mips操作系统的搭建和运行、调试环境的安装以及用户应用和系统调用的添加过程中，我仔细阅读了指导书的内容以及Hos-mips的源码结构，发现它主要由用户态和内核态两部分代码进行组织，且系统调用的运行机制和其他Linux系统类似。在之前的操作系统课设中曾经在Ubuntu这个Linux发行版上实现了系统调用的添加，为本次实验打下了一定的基础，本次实验也让我复习了操作系统相关的知识。

总而言之，本次实验主要围绕Hos-mips操作系统展开，在实践过程中感受到了软硬件结合的魅力，学习新知识的同时也使得之前学习的内容融会贯通，收获颇丰。另外，实验中对Hos-mips操作系统的熟悉也为之后的实验打下了基础。

### 改进设想

在实现Hos-mips用户应用程序添加时，使用了比较简单的做法，将应用视为Hos-mips的一个命令来实现。

进一步可以将自己编写的应用加入到Hos-mips系统镜像中，作为文件系统中的一个可执行文件存在，就像Windows操作系统的.exe文件一样。

# 应用部分的设计与实现

## 设计目标

（1）对蓝牙小车的硬件模块中基于AXI4总线接口的蓝牙模块和马达驱动模块进行设计和实现。同时将通过相应的测试程序对其进行必要的测试和验证，为后续在Hos-mips操作系统上实现蓝牙小车应用打下硬件基础。

（2）在Hos-mips操作系统中实现蓝牙模块和马达驱动模块的驱动程序，并以蓝牙模块和马达驱动模块驱动程序为基础，对蓝牙小车前进、倒车、转向等基本功能进行测试。

（3）采用设备驱动方式实现一个简单的蓝牙小车应用系统。

（4）学习如何实现蓝牙小车的自启动功能。

## 蓝牙模块设计与实现

查阅指导书后了解到蓝牙模块是基于串口方式工作的，因此在MIPSfpga硬件平台中增加一个UART串口模块就可以对蓝牙模块进行控制。

打开之前的Vivado工程，在Block Design中添加一个AXI UART16550串口模块，如下图4.1所示：

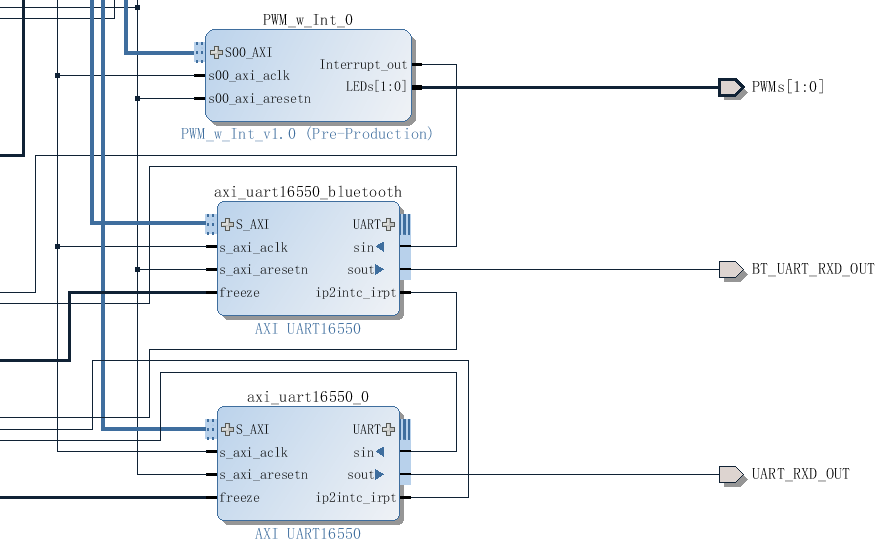


图4.1 添加蓝牙控制模块

需要注意该串口模块的sout引脚需要引出一个外部引脚，并在约束文件中修改为相对应的引脚名称。

添加模块之后分配物理地址，然后经过综合过程生成比特流，就可以编写程序进行测试，编写程序可以参考之前\_C文件夹中的内容和指导书附录中的无线蓝牙测试程序源代码。

## 遇到的问题

**问题描述：**

使用hos-mips源码的debug目录下面的比特流能够正常加载hos-mips系统，但使用自己生成的比特流会卡在断点，如下图4.2所示：

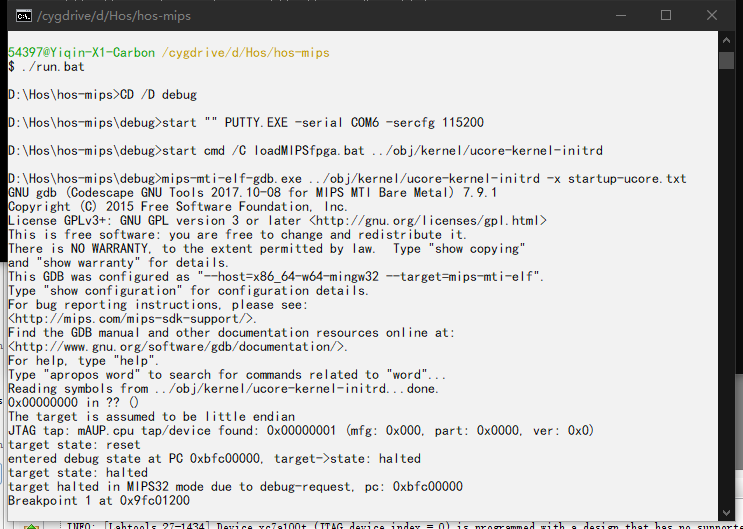


图4.2 加载Hos-mips卡在断点

硬件平台在第一部分的测试都通过了，生成比特流过程中也未报错，但仍存在上图4.2中的问题。

卡在断点时，开发板的16个LED全闪烁，且频率应该是从左到右越来越高的。怀疑是一种报错提示的方式。

由于该问题没能在短时间内解决，无法继续进行后面的实验，感到非常遗憾。

# 总结与心得

## 个人总结

通过这次实验，我主要有以下收获：

1. 进一步巩固了蓝牙通信编程能力；

2.结合了以前所学的操作系统、组成原理、软件工程、计算机系统结构等课程 的知识并建立了一个完整的计算机系统的概念；

3.掌握了基于AXI4总线标准的接口设计与实现；

4.体会到了团队合作的重要性。

## 实验心得

经过此次的项目经验，我们学会了学以致用，能够将自己在课堂上学习到的东西能够在项目中加以利用,正所谓知行统一。首先我们复习了组成原理、计算机接口技术、操作系统、软件工程等方面的知识，为最终实现智能小车的相关功能奠定基础。由于小车上有装了蓝牙模板等，我们又能摸索学习蓝牙模块的一系列知识，对于我们是一次长知识的经历同时也是一次能够接受的挑战，还好大一的时候我们在工程训练这门课种接触过蓝牙小车的相关操作，也算是熟悉了小车的蓝牙连接操作过程，所以这部分并没有花费我们组的太多时间。

在实验中，我们也遇到了许多的困难和问题。刚开始拿到小车的时候都比较兴奋，但看了大纲以后小组间的分工确定不下来，而且都没学过接口技术这门课，在总线的接口设计与实现这块大家都觉得很棘手，而且小车的接口很多，对应的线也比较多，驱动板的相关说明却很少，很多想法在实际操作时都行不通，于是我们就加强了同学之间的合作，几个同学公关一个模块，同时私下经常去请教其他组的同学，借鉴他们的想法，有时候问题也就迎刃而解。每个同学的特长也不一样，每个模块都有擅长的同学带头解决问题。但是在后期的小车总体设计中，由于正处于同学们在升学和秋招找工作的时间段，我们组几乎所有的同学都在备考以及笔试面试的进行中，因此我们组只完成了前面比较重要的几个模块，没能投入全部的精力去实现一个更为完善的智能小车。但是在这种小组内部成员的合作中，我体会到了团队合作和的效率和团队沟通的重要性，如果下学期还能接触到类似的课程，我们一定会花费更多时间做出更多的成果。最后在这里也感谢两位老师对于我在本次实验中问题的耐心解答，也感谢本组所有成员在实验中对于我的帮助和建议。我相信这次综合能力培养实验中我的收获将会对我今后的学习有很大帮助。

## 意见和建议

因此课的上课时间与找工作和备考研有所冲突，建议开课时间可以提前至大三下学期末，暑假实习的任务并没有那么重，可以在大三暑假一并完成，返校后直接上机房操作；可加大对智能小车的宣传力度，吸引更多同学参选此课，间接增加智能小车的创新；上课时同学都比较松散，同学之间的监督力度欠缺，完成任务的效率较低，建议老师可酌量安排每次上课的任务量和到勤上课纪律。

# 致谢

特别感谢Imagination Community对本次实验中的支持和贡献，正因为有这些业界公司无私的教学支持才能让我们这些学生在学习中走的更远。

在实验过程中，尤其感谢老师在线上对我提出的问题提出耐心细致的解答，为我打开了新的思路。尽管最终没能完整地实现智能小车的功能，但我仍然非常感谢老师的努力和付出，让我受益匪浅。

# 参考文献

1. 胡迪青，邵志远，李若时. 系统能力培养综合实践--蓝牙小车. 清华大学出版社，2020年7月.
2. Vivado中MIG核中DDR的读写控制

URL: https://blog.csdn.net/wordwarwordwar/article/details/79539049

|  |
| --- |
| 一、原创性声明 |
| 本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。  特此声明！  **作者签字:** 吉伍子吉 |
| 二、对课程实验的学术评语（教师填写） |
|  |
| 三、对课程实验的评分（教师填写） |
| |  |  |  |  | | --- | --- | --- | --- | | 评分项目  （分值） | 报告撰写  （20分） | 实验过程  （80分） | 最终评定  （100分） | | 得分 |  |  |  | |
| **指导教师签字:** |