计算机组成原理 108组实验报告

1 项目简介

本项目是2018-2019秋季学期清华大学刘卫东老师计算机组成原理课程挑战组的实验项目。项目的设计要求、功能测例、监控程序和项目代码可以在同文件夹下找到。

1.1 项目成果

我们最终设计并实现了一个支持MIPS指令集的32位五级流水线CPU。具体功能包括：

- 支持共？条MIPS指令系统的基本指令
- 支持与CPU有关的异常、中断及TLB表查询
- 通过全部功能测例，并能运行监控程序，CPU主频达到50Mhz
- 支持相关的扩展外设，包括RAM读写、flash读写、串口读写、VGA图像输出等
- 编写了一个汇编语言游戏程序（贪吃蛇），并在cpu上运行

1.2 开发过程简介

与普通组的“奋战三星期，造台计算机”不同，挑战组项目贯穿整个2018-2019秋季学期。我们在此过程中研精覃思，夜以继日，最终的设计目标才得以完成，可谓“千淘万漉虽辛苦，吹尽狂沙始到金”。

1.2.1 组队及分工

本组（108组）的三名组员及相应分工如下：

姓名	班级	学号	分工
丁相允	计65	2016011361	?
尹一帆	计65	?	?
阎静雅	计65	?	?

1.2.2 开发流程

1. 第1周至第4周：组队，项目前期调研（向往届学长学姐咨询），学习计算机系统基本知识，确定开发语言，确定实验目标，初步模块划分及任务分工。
2. 第5周至第7周：编写CPU的基本架构，支持MIPS基本指令，ram读写，相关代码调试。
3. 第8周至第10周：编写异常处理部分、中断相关部分、TLB，相关代码调试。
4. 第11周至第14周：编写各外设模块，各部分独立调试。

5. 第15周：各部分协同联调，编写贪吃蛇汇编程序，项目展示。

1.2.3 开发环境

开发语言

我们使用Verilog语言完成本项目的开发。

我们曾经试图使用Chisel（由伯克利大学开发的基于Scala的硬件开发框架）进行开发。经过最开始的尝试，我们总结了chisel的如下优缺点：

优点：提供抽象的数据类型和接口，使用面向对象的方法，能够批量端口连接，简化开发过程，能直接通过sbt（Scala Build Tools）编译成Verilog语言。

缺点：学习成本高（需要学习Scala和Chisel并配置相应环境），仍需要手动编写逻辑电路，在小型项目上优势不明显，不支持高阻态等（可以通过内嵌Verilog代码解决），编译后的代码注入thinpad_top框架后难以调试。

综合以上分析，我们最终还是选择了Verilog。

开发工具

我们使用Vivado 2018.1作为IDE。Vivado提供了便捷的verilog编写和仿真工具，并能生成bitstream直接导入FPGA芯片中，且与thinpad_top平台兼容性好。

我们使用清华大学计算机系的32位开发板进行实验。同时我们也使用了更加方便的在线实验平台。

2 项目设计

2.1 指令系统

我们完成的cpu共支持？条指令。各指令的格式、功能如下：

（一个表格）

2.2 CPU设计图

2.3 贪吃蛇游戏设计

我们采用CPU直接控制的方式写显存，即在汇编代码中使用`store`指令到特定的地址。

3 CPU基本模块

我们的CPU采用单一时钟的设计，各模块均引入了clk（50Mhz时钟）和rst（1表示复位，0表示运行），在以后的各个模块中不再赘述。

3.1 PC

3.2 IF_ID

3.3 ID

3.4 registers

3.5 ID_EX

3.6 EX

3.7 EX_MEM

3.8 MEM

3.9 MEM_WB

3.10 HILO

3.11 div

4 CPU扩展模块

4.1 MMU

4.2 sram_control

4.3 uart_control

4.4 vga_control

4.5 flash

5 其他相关代码

5.1 对实验框架的修改

最终的实验框架结构如下图：

对原有框架的修改主要是在thinpad_top中加入我们编写的CPU：

```
CPU CPU0(
    .clk(clk_50M),
    .clk_11M(clk_11M0592),
    .rst(reset_btn),
    .btn(touch_btn),

    .dip_sw(dip_sw),

    // 两块ram的连线
    .instAddr_o(ext_ram_addr),
    .dataAddr_o(base_ram_addr),

    .inst_WE_n_o(ext_ram_we_n),
    .inst_OE_n_o(ext_ram_oe_n),
    .inst_CE_n_o(ext_ram_ce_n),
    .inst_be_n_o(ext_ram_be_n),

    .data_WE_n_o(base_ram_we_n),
    .data_OE_n_o(base_ram_oe_n),
    .data_CE_n_o(base_ram_ce_n),
    .data_be_n_o(base_ram_be_n),

    .data_io(base_ram_data),
    .inst_io(ext_ram_data),

```

```
    // 实验板上的LED和数码管连线
    .led_o(leds),
    .dpy0_o(tmp0),
    .dpy1_o(tmp1),

    .rxd(rxd),
    .txd(txd),

    // VGA相关连线
    .video_red(video_red),
    .video_green(video_green),
    .video_blue(video_blue),
    .video_hsync(video_hsync),
    .video_vsync(video_vsync),
    .video_clk(video_clk),
    .video_de(video_de),

    // flash相关连线
    .flash_a(flash_a),
    .flash_d(flash_d),
    .flash_rp_n(flash_rp_n),
    .flash_vpen(flash_vpen),
    .flash_ce_n(flash_ce_n),
    .flash_oe_n(flash_oe_n),
    .flash_we_n(flash_we_n),
    .flash_byte_n(flash_byte_n)
);
```

另外，我们还修改了`tb.sv`中的`BASE_RAM_INIT_FILE`、`EXT_RAM_INIT_FILE`、`FLASH_INIT_FILE`参数，便于在仿真时进行调试。另外我们增加了ip核bram作为显存（上文中提到过）。除此之外，我们没有对thinpad_top原有代码的其他部分做修改。

5.2 图片压缩

因为实验板的VGA输出是3位Red，3位Green，2位Blue，普通的图片没有这种格式的，故需要我们进行转换。我们使用python将图片转换为二进制，然后存储到flash中。转换核心代码如下：

```
filename = sys.argv[1]
img = Image.open(filename)
img = img.resize((800, 600),Image.ANTIALIAS)
img.save(filename+'_resize.png','png')

file = open(filename[0:-4]+'.bin','wb')
for i in range(600):
    for j in range(800):
        color=img.getpixel((j,i))
```

```
        R = color[0] // 32
        G = color[1] // 32
        B = color[2] // 64
        s = struct.pack('B', R*32+G*4+B)
        file.write(s)
file.close()
```

首先将图片转为800*600大小，然后对于每一个像素的RGB值，其范围本来为[0,255]，通过除以32或64即可将其转换为相应的二进制表示。最后通过`struct.pack`函数写到bin文件中。最后将bin文件导入flash，运行cpu，就可以在显示器中显示图片。

由于最后阶段时间紧，我们没有做视频相关的处理，但我们对视频输出做了基本的设计构想，即采用一些视频压缩算法将视频存入flash，然后使用硬件将视频解压，通过与显示图片相同的方法即可在显示器中播放。

5.3 贪吃蛇汇编程序

（？？？）

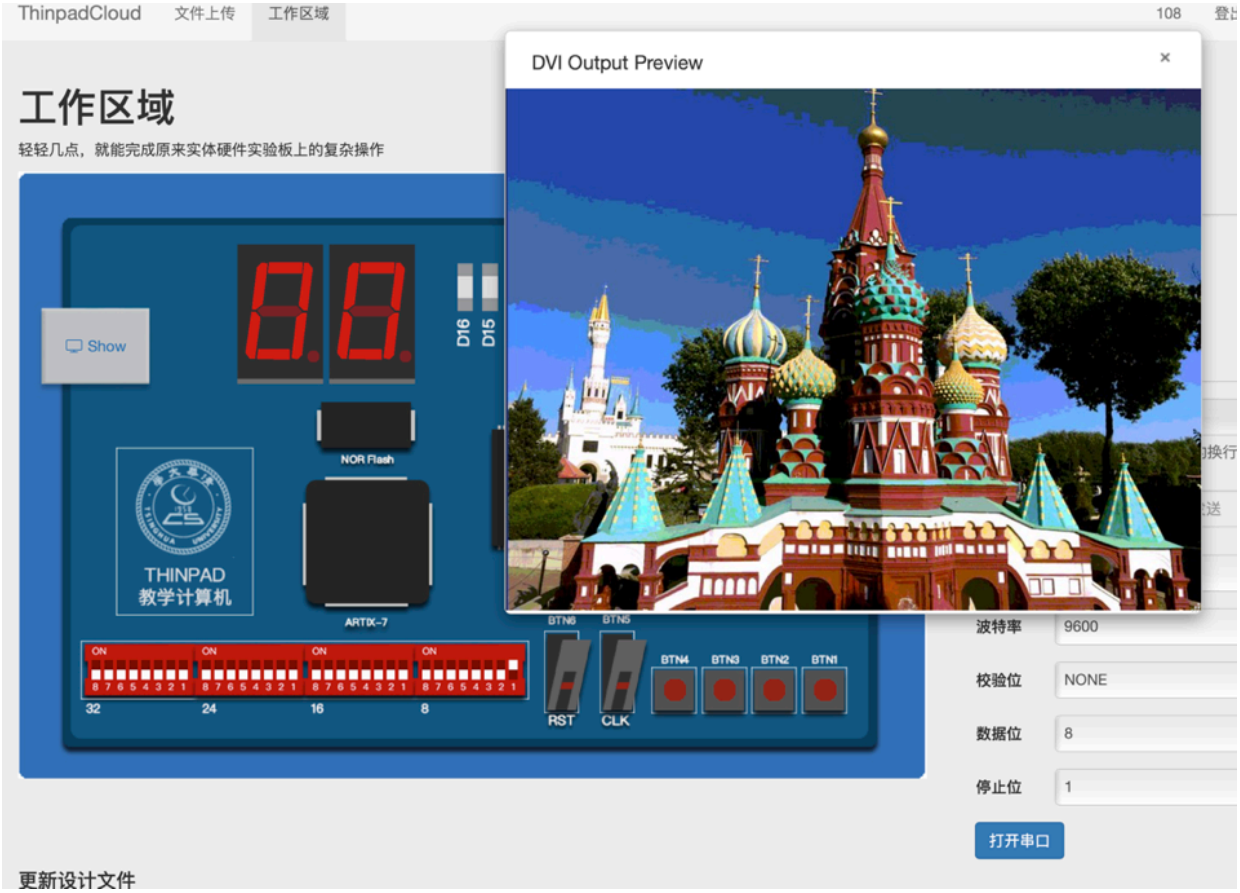
6 实验成果展示

6.1 通过功能测例

6.2 运行监控程序

6.3 从flash读取并展示图片

在线实验平台上的效果如图：



6.4 贪吃蛇游戏

7 实验总结

7.1 CPU设计总结

在实验的设计阶段，我们咨询了许多造过计算机的先辈们，并参考了众多教科书和网上资料，中途不断改正，才形成了最终的CPU设计版本。在此过程中遇到的问题和总结如下：

- 开始时可以先不确定各个模块的接口，可以在开发的过程中随功能的变化添加。据我们了解，普通组的开发过程是先设计好了数据通路，然后再开发各个模块。但我们组借鉴了《自己动手写CPU》中的方法，开始时只是大致画出了CPU设计图，在随后的开发过程中再明确各个接口。当然，设计阶段必须对每个模块的功能认识明确，对整个CPU如何工作有总体的理解。
- 在明确指令系统的过程中，我们组员对其中有些指令的理解有误，对后续的开发过程中造成了麻烦。遇到这种问题时不建议上网搜索，因为网上的内容错综复杂，还有可能涉及MIPS的版本等问题。最好的办法就是直接阅读MIPS文档，其中对各条指令的形式、功能都有极为详细的描述。

7.2 开发过程总结

大三上学期是计算机系课业最为繁重的一个学期，除造计算机外，我们还有软件工程课的大作业、编译原理课的编译器、网络原理课的路由器等等需要开发的工作。开发过程中我们遇到的问题及总结如下：

- 确定好开发时间，拒绝拖延。在第一次检查之前，我们面临许多项作业的deadline，本项目只进行了设计分工工作。于是在检查前的几天里，我们通宵达旦、夜以继日，终于支持了MIPS的基本指令。但我们也进行了反思：欲速则不达，一定要合理规划开发时间。在之后的几周，我们一般采用每周3天，每天若干小时的方式开发，既提高了开发效率，又保证了身体健康。
- 循序渐进，敏捷开发。虽然我们不是计原软工联合项目组，但软件工程课中所学到的诸多知识如使用git管理代码，迭代开发，敏捷开发等对我们此项目有莫大的帮助。使用git进行版本控制，使我们能够多人同时开发不同的部分。而敏捷开发所提倡的沟通交流也十分重要，包括小组成员之间的沟通，以及与老师和助教的沟通。我们的项目最终有近万行代码，中间过程中修改、重写、丢弃的代码更是不计其数，但总量虽多，只要制定好开发计划，在git的帮助下最终也取得了成功。借用《自己动手写CPU》中的一段话：

  > 在实现OpenMIPS处理器的过程中，笔者深刻体会到“罗马非一日建成”这句话，外表看起来巨大、庞杂的罗马，也是通过人们一步一步、一天一天、一点一点建成的，处理器也是如此，读者首先不要被处理器的神秘吓到，从最简单的地方入手，逐渐增加功能、完善设计，一行代码一行代码地书写，不仅要有实现处理器的远大目标，还要确立切实可操作的短期目标，比如本周实现除法指令，下周实现转移指令，诸如此类，等有一天你突然回头，会发现，原来已经走了那么远，实现了那么多功能。李白有诗云：两岸猿声啼不住，轻舟已过万重山。当是此意。

7.3 心得体会

世界上第一台计算机ENIAC每秒只能进行5000次计算，而当今小小的手机计算能力都远超它。从图灵、冯诺依曼的时代开始，一代代的计算机科学家们呕心沥血，继往开来，才让今天的计算机硬件系统如此的精妙，功能如此的强大。在本项目里，我们用自己的双手实现了一个简单的32位CPU。虽然这在历史上早已被实现过，也不是什么困难的事情，但在这其中我们深刻体会到了计算机系统结构的博大精深，也让我们对计算机先贤们肃然起敬。科技的进步，在未来或许也需要我们的一份力量。

光阴似箭，日月如梭，一学期的课程终于结束了。造计算机，这所有贵系人都要经历的一道坎，我们也终于平安度过。虽然我们离”深入理解计算机体系结构“还有很远，但一个简单的CPU，便也是我们努力的见证。

最后借用《自己动手写CPU》中的一段话作为本项目的总结：

> 一个人的旅行是孤单的
>
> 一个人的冬季是寒冷的
>
> 但是
>
> 一个人的CPU是骄傲的
>
> 让我们骄傲一次

8 附录

8.1 上交文件列表

```
.
├── code
├── image
|   └── CPU.png
├── report.md
└── 大实验报告要求.png
```

8.2 参考资料

刘卫东老师计算机组成原理课件

《自己动手写CPU》雷思磊著，电子工业出版社

挑战组网站http://os.cs.tsinghua.edu.cn/oscourse/CCP2017上的相关资料，包括功能测例资料、监控程序资料、32位实验板说明、RAM文档、flash文档、VGA文档

往届软工文档（李北辰和谢磊组文档，逆光组文档）

MIPS32 Architecture For Programmers（共三卷）

8.3 鸣谢

感谢刘卫东老师为我们生动地讲解计算机结构知识，为CPU的实现打下了坚实的基础。

感谢李山山老师、杨松涛助教、张宇翔助教为本课程的付出，尤其是最后一位帮助我们解决了实验中遇到的问题。

感谢清华大学计算机系为我们提供这样一门有趣并极具挑战性的课程。