

序

其实关于写总结的这件事从知道比赛结果后一直搁在我心里，我却迟迟没有开始下手，原因有很多，一是想起这个比赛心情总归是有点郁闷，二是为这个比赛确实是倾注了很多心血，最后的结果却是不尽人意，三是怕是让陆老师和向老师失望了，连决赛都没有闯进去，别的学校怕是一年比一年好，我们可好，还不如两位学长。所以想起来这件事情就很伤心，就先索性不去想他，专心致志先忙着自己的事情。但晚上睡觉的时候，心思总会不自觉的溜出来，毕竟这是我参加的时间跨度最长，难度也最大的一个比赛。而且确实像向勇老师老早就提到过，假如这个比赛没有拿奖，你们又有能够从中得到哪些收获呢？（当时听这句话的时候，确实心里没什么感触，现在却感触良多）这个比赛我确实是有很多的收获，所以就想拿出四五天的时候，好好回顾一下整个的准备比赛的过程，把路上的一些坑，一些经验总结一下，希望接下来参加这个比赛的学弟学妹们，能够有所警惕，有所收获。

甄淇深
2019.09.09

序 -----1

前半年的准备过程 -----3

 2018年十月份.....3

 2018年十一月份.....4

 2018年十二月份.....4

 2019年一月份.....5

 2019年三月份.....5

 2019年四月份.....7

 2019年五月份.....7

一个月的准备过程 -----8

 7月1日到7月3日.....8

 7月4日到7月6日.....8

 7月7日到7月12日.....8

 7月13日到14日.....8

 7月17日到7月20日.....9

 7月21日到7月22日.....10

 7月23日到7月24日.....10

 7月25日到7月29日.....10

 7月30日到8月1日.....11

 8月4日到8月6日.....11

总结 -----13

思考 -----13

尾 -----13

附录 -----14

前半年的准备过程

2018年十月份



第一次会议

在学院里面的大群里面，惠导发了一个通知，说系统能力培养大赛要招新了，因为本来我就对计算机是怎么工作的问题就很感兴趣，再说前两年一直在学习基础课程，这个问题依旧还不是很清楚，所以就抱着试一试的态度去参加了这第一次会议，记得当时我迟到了大约五分钟，所以进去的时候还挺紧张的，依稀记得陆老师是满脸笑容看着我，我也就慢慢放松了下来。陆老师主持会议，朱威浦和江学谦两位学长介绍经验。期间向老师有问，有哪几位同学是看过朱威浦学长写过的总结经验过来，零星几个人举了手，我也举了手，当时还站了起来说了说看过之后的感想，现在回过头看，当时也是年轻，太naive了。然后每个人也发言了自己的感想，说了说参加这个比赛的动力来源是什么。当时20多个人，说了些是因为兴趣，也有说是为了想参加个竞赛，或有的只是想有个经历，回想起最后一次2019年8月份的总结会议，自始至终坚持下来了也就三个人。不过毕竟每个人的追求是不同的，每个人愿意在哪个方向所花费的时间自然也是不同的。追寻自己所热爱的事业当然是无可厚非的。


本次会议之后的主要内容是安装虚拟机，搭建vivado环境，学习Verilog语法知识，同时也推荐了几本书，目标是树立起基本的设计cpu的概念，

第一周（2018.10.15-2018.10.21）：实验指导





一. 实验环境配置

1. 安装Vmware虚拟机
2. 在虚拟机下安装Ubuntu16.04
3. 在Ubuntu16.04下安装Vivado2018.1
 -  [软件安装包](#)
 -  [安装参考资料](#)

二. 实验知识储备

1. 复习Verilog语法知识
2. 了解MIPS指令集架构
3. 推荐资料《自己动手写CPU》第一、二章*  [博客版](#)

三. 进阶资料

1.  [The MIPS32 Instruction Set Manual.pdf](#)
2.  [MIPS32_microMIPS32_Privileged_Resource_Architecture.pdf](#)
3. *  [Verilog.pdf](#)
4. *  [SystemVerilog.pdf](#)
5. 《计算机组成与设计 硬件/软件接口》第二章
6. 《数字设计与计算机体系结构》第六章

当时也是一脸的懵，什么都不清楚，当时唯一学过有关的课程就是数字逻辑了，即使是学过的数字逻辑还相对更底层一些，和设计cpu关系也不是很大，所以也是对着学长推荐的书本进行学习，推荐的几本书中，因为《自己动手写cpu》语言比较简单易懂，语言也挺生动，所以作为入门读物还是很合适的，适合新手从零起步直接去做一个CPU，学妹学弟们如果去借图书馆的那本《自己动手写cpu》，里面都留存着学长留下的各种笔记。我也对这本书里面的openMIPS十分熟悉，从一开始搭建一个基本流水线大框架，到逐渐的添加各类指令，这本书还是很不错的，其中对数据冲突的处理，对我们后来提升cpu主频有一定的借鉴，他的那些设计规范也给了我们深远的影响，有好也

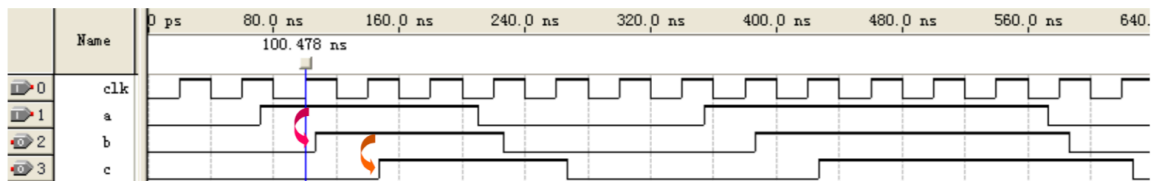
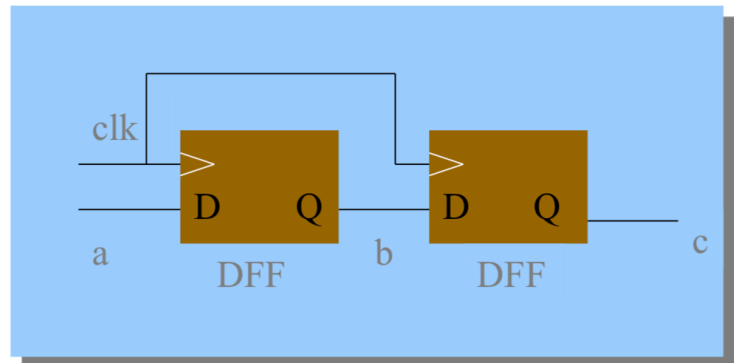
有坏，比如always*的这种设计语言其实是不提倡的，组合逻辑用assign来表达就好，而且他的设计方式即对各模块设计添加各类指令时，是在各模块中通过添加接口，这对各组员分工时十分不友好，（我们是通过添加功能模块的方式进行的）。这本书也有些局限性，对一些包括怎么debug，波形图怎么看，怎么测试，这些很重要的能力也并没有好好得阐释清楚。

总结延伸：因为无论是《自己动手写CPU》也好，上一届的开源项目也好，我们的项目也好，你所看到的都是一个成品，其中肯定是有版本迭代过的，期间所遇到的bug都是调过的，所以一些很细节的坑如果只是看成品你是没有办法遇到的，就比如一个很细节的问题，verilog中阻塞和非阻塞赋值的区别在哪里，具体显示到波形图中有是个怎么样子。

(1) 非阻塞赋值方式

```
always @(posedge clk)
begin
    b <= a;
    c <= b;
end
```

非阻塞赋值在块结束时才完成赋值操作！



- ◆ c的值比b的值落后一个时钟周期！
- ◆ 若块内有多条赋值语句，则在块结束时同时赋值。
- ◆ 多条非阻塞赋值语句并行执行！

71

所以说挺建议大家早早开始动手写，无论基础知识掌握的怎么样，先动起了手来，当时我就是，觉得自己基础啥的都不清楚，总想有个万全的准备再开始，所以也就耽搁了很长时间。

2018年十一月份

第二次会议的时候，王娟老师给我们上了一节大四小学期时讲解写单周期cpu的ppt。里面阐述了基本的MIPS指令格式和基本的概念，但当时我记得我因为还不怎么了解这方面的知识，所以感触不是很深，听听就过去了。

第三次会议记得是两位学长给我们复现他们比赛作品，对cpu设计一窍不通的我也只是听个新鲜，什么也听不懂。不过这个时候人就开始逐渐流失了，不过我记得是这次过后，我们开始分成了两个小组，不过虽说是分成了小组，小组成员之间依旧是各忙各的，不怎么联系。此次会议过后，学长开始给我们提出要求，自己先尝试着写一个单周期的cpu出来。试着按照每类指令（逻辑、算数、移位、跳转）都写出来一条来。

2018年十二月份

第四次会议的时候，差距开始显现出来了，赞哥已经把基本的cpu的数据通路和控制通路都已经十分熟悉了，对每条指令的执行过程以及各个模块都已经捋得很清楚了，我还处于一种云里雾里的一种状态，到网上也搜寻了很多如何写单周期cpu资料，学长推荐的几本书也翻了翻，回想当初，一点经验就是，感觉资料太多了，抓不到重点，不知道从哪里入手，那个时候还花了很长一段时间去复习数字逻辑，还专门看了《数字设计》那本书前几章，什么D触发器啊，锁存器啊什么的，其实并没有多大的用处，看了一些大赛的培训视频，看到SOC那边我就不怎么懂了，也耐不住性子去接着看下去了。而且那个时候对vivado的还不怎么会用，学了学verilog语法（不过这个还是很有用的）。

延伸：包括我去学习ucore的时候也是有同样的感触，提供的资料很多，学堂在线上的向勇老师的视频课程，实验指导书、github上提供的各种专业性资料、PIZZA的问题广场、

总结一下就是，没有一个明确的侧重点，想要每个都看，然后每个都是浅尝辄止，这样很难树立起一个系统的，现在反思一下，学习就先抓住主干，然后遇到不懂的地方再去查阅，记得是查阅，不用去通读，只关注自己需要的地方。

推荐一个学习的顺序

- 1、王娟老师推荐的一个MOOC 东南大学的《计算机系统综合设计》（课程里面不仅有教如何设计cpu的还有教怎么进行外设设计的）
- 2、《数字设计》第七章，前面单周期部分
- 3、小学期课程里有个单周期参考。

看过上面的资料之后，应该就可以写出来一个单周期cpu了，这个时候对一些概念应该就有体会了，什么，比如说cpu的功能有哪些：指令控制、操作控制、时序控制、数据加工、端口访问、中断处理，cpu的流程是什么。等等。这个时候可以利用我们的精工版进行尝试下板了。

2019年一月份

在本学期的最后一次会议，每个人是按照的想法都上黑板汇报了一下自己设计的单周期的数据通路，每个人的数据通路也是大同小异，因为单周期还是比较简单的，但真正能够把单周期写出来的却是寥寥几人。大家还只是停留在理论上。甚至理论上也没有建立好，我记得当时只有赞哥是把单周期写的差不多了，包括我的其他人也只是开始了几条指令，更别提下板了。这个时候就反映出了执行力的差距，此外，人也基本上走的差不多了，当时剩下的应该是不到八个人了，于是就按照人数组成了两队。

回顾上半年的概念，算是只是基本建立起来了概念，但实际上做的工作却基本上没有，而且那个时候对比赛也没有紧迫感，更缺乏一个节奏，属于走一步看一步，而且大三上事情的确还是蛮多的，而且寒假又参加了美赛，所以上半年的准备工作也就到这了

对比于学长上一年的准备过程，发现我们和他们之间的准备方式还是有蛮大区别的，学长上一年这个时候把精力放在了复现实验上，而且主要是操作系统这个级别上，复现清华naiveMIPS在上面跑Supervisor 和ucore，而我们主要把精力发在了cpu上，不知道这算是一个进步还是一个退步，（因为我们连复赛都没有进去，就何谈操作系统了），不过考虑到学长上一年是第一次参赛，没有什么经验，手里面也不像我们资料这么充足，光是复现实验就耗费了很多的功夫。所以也算是帮我们把一些坑都踩过了。

2019年三月份

开学伊始

新学期的第一次会议

在第一次会议上，老师们问我们经过一个寒假的进度怎么样，大家就支支吾吾，因为我寒假精力都放在了美赛上，对这个比赛又没有怎么上心，进度停滞不前，就只是把那些培训视频看了看，其他的也没干什么了。这个时候赞哥说了一下，我流水线指令已经写到了20多条了，我们都暗自惊叹，这也太快了吧，这个时候，我连流水线的概念是什么都还不知道，更别提怎么写了。经过了此次会议之后，感叹别人的进度之快，我也开始正式把自己的主要精力放在了这门比赛上。

怎么开始写流水线，下一步的计划是什么？无论是《软硬件接口》还是《数字设计》还有一些其他的书，开篇都只是阐述流水线的概念，流水线的原理，流水线里面会有数据冲突，这些就像是我们上课一样，上课时讲些原理和理论，但真正动起手来又是一番天地，vivado你会不会用，包括仿真界面怎么添加波形，verilog语法你熟不熟，这些在现在的我们看来是非常简单的一件事，但当时真的像拦路虎一样，阻碍你前进，所以《自己动手写cpu》这本书虽然说这样那样的缺点，但真的就只有这本书是在代码层次上，一行一行的教你怎么先用一个ori指令构建一个基本的流水线框架出来，然后再一步一步往上面添加逻辑、算数、跳转、加载等指令的，最后再加CPO设计和异常处理设计。逻辑很清晰，所以三月份和四月份我基本上都在啃这个书，同时也把学长上一年的cpu初赛提交拿出来，对比着来看，同时结合学长的设计文档把数据通路给画了出来。

同时因为这两个月因为陆老师有要求操作系统实验课换成了清华大学的ucore。前期已经提到过了，没有抓重点的问题。我光lab0实验准备就花了大约一个月的时候（当然还有其他的事情）配置环境和预备知识等等，

我的建议是做ucore实验的时候，以实验指导书为主要，其他的包括学堂在线上的视频以及配套的ppt为辅，光lab0的内容有100多页，里面讲到的内容不是特别懂也没有关系。先往下看，比如lab1，搞清楚boot文件下一个.S文件.c文件，然后跳转到init下的init.c，看如何开始初始化一系列配置。然后搞清楚具体每个初始化函数的含义，跳转到每个函数的定义去理解其具体实现过程，后面就顺理成章啦。

```
void
kern_init(void){
    extern char edata[], end[];
    memset(edata, 0, end - edata);

    cons_init();                // init the console

    const char *message = "(THU.CST) os is loading ...";
    cprintf("%s\n\n", message);

    print_kerninfo();

    grade_backtrace();

    pmm_init();                 // init physical memory management

    pic_init();                 // init interrupt controller
    idt_init();                 // init interrupt descriptor table

    clock_init();               // init clock interrupt
    intr_enable();              // enable irq interrupt

    //LAB1: CAHLLenge 1 If you try to do it, uncomment lab1_switch_test()
    // user/kernel mode switch test
    lab1_switch_test();

    /* do nothing */
    while (1);
}
```

在第二次会议上，学长给我们演示了如何利用龙芯的soc_up通过pemo来启动linux，依旧是云里雾里，抚额，叹气。

不过这次会议中加入了三名新同学，分别是李航、吴杰和范志鹏，同时这个时候人都走的差不多了，老一批的人到现在也就只剩下了我、贺隽文、王赞三个，我们六个人组成了两队。甄、李、贺、吴四人一队，王、范两人一队，（为什么没有三三分呢？我也有点莫名其妙），由于这几位同学都还没有接触过相关知识，我就给他们说先花两三天看完《数字设计》第七章十几页的内容，同时结合《自己动手写cpu》了解经典五级流水线的数据通路和控制通路的构建。

这个时候学长开始给我们这一组布置任务，在接下来的一个月里，也就是四月末，能够写出来一个基本的流水线，不包括cp0和异常处理，只实现逻辑、移位、跳转、算数、加载等指令是我cpu，同时学长嘱托我们，一定要对测试这项工作特别上心！如何测试呢，利用《自己动手写cpu》里面的每章后面提供的测试用例来进行测试。

2019年四月份

报名参赛

比赛报名截止日期是4月30日，我们组也恰巧在这个日期完成了基本的cpu设计，设计过程和《自己动手写cpu》一致，首先通过ori指令构建出一条基本的流水线，证明基本的数据通路和控制通路是没有问题的，然后我们四人按类分指令往上面添加，（逻辑算数、移位、跳转、加载）验证的话，就通过《自己动手写cpu》上面提供的测试用例，观察波形。但我们采用的数据通路是《数字设计》里面的图，前面也提到过，《cpu》那本书的设计文件是按五大阶段，取值、译码、执行、访存、回写直接对应的就是if.v id.v ex.v mem.v wb.v 外加上几个控制模块等等、这样就导致了一个问题，控制通路不是很明确，控制信号和数据传输混杂在一起，所以如果按照这种模式来写的话，思路清晰还好，思路不清晰的话就特别容易乱，所以我们换种思路，按照功能来划分模块，通过在控制信号上添加后缀F、D、E、M、W来表明阶段，学弟学妹们看我们的模块命名也大约知道个所以然来，例如get_pc,get_instr,get_SrcA等等。

2019年五月份

到了后期，我们也从两周开一次组会变成了一月一次组会，转眼到了5月末，这次会议陆老师邀请了向勇老师来参加，向勇老师还是和第一次会议见到的那样，精神好，说话中气十足，我们汇报了各自组的进度，我们组还停留在利用《cpu》上面的测例来进行测试，还停留在仿真阶段。而隔壁组已经开始下板跑指令了，89条指令，好像已经过了80个左右，只剩下后面几个cp0异常延迟槽的没有过了，当时觉得贼啦慌，毕竟我还对怎么下板的流程都不清楚，怎么分配引脚（其实不用分配、龙芯已经给你准备好了，添加一个提供的xdc约束文件就好了）综合、实现、生成bit流这一套流程也不会（其实很简单，就直接综合，实现，生成bit流就好了，）怎么样去适配龙芯提供的框架，sram框架是个什么意思，axi框架又是个什么意思，更何况我们当前版本的cpu还只是一个阉割版本的，最难的异常处理和cp0都还没实现（当初觉得最难，现在也就觉得那么回事），trace是个什么玩意，怎么样用trace来调bug，一概不知，这就是当前我们五月份末的进度。不过在这一个月中，我们小组内部的交流是越来越多了，并且我们之间交流的主题是，框架到底是什么，我们当时对这个很疑惑，cpu虽然是有带欠缺，但我们知道是怎么回事，但sram框架是个什么东西，怎么写出来，cache又要加在哪里，这是我们不解的地方。（其实根本不用操心这么多，因为这些龙芯都已经帮大家做好，你要做的恰恰就是设计cpu内部就可以了，只要加入到func_sram下的的Mycpu文件就好了，打开vivado项目就OK了）

so, 这就是我五月份的状态, 比较焦虑, 包括6月份也是, 准备期末考试, 基本上没准备这个比赛, 6月26号期末考试后, ucore实验要验收了, 需要验收前四个实验, 当时我应该是做到了lab3, 还要作lab4, 不过做完lab4, 做嗨了, 感觉还是挺有意思的, 就继续做了lab5, lab6, 不过这是后话了。

在6月31日, 应该是这一天, 陆老师又召集我们开了一次会议, 同时这也是真正的最后一次赛前会议了, 在等陆老师来的过程中, 我们六个人也分别讨论了以后各自想怎么发展啊, 也吐吐槽, 说哪个老师给分低啊巴拉巴拉的。(对了, 说个题外话, 其实还有一组女生组, 这几位女生都是从延安大学交换过来的, 代表延安大学参赛, 不过从三月份开始后, 会议上也不见他们的踪影了。最后好像还是没有参加。)那么这次会议所以这次也算是一次赛前动员会, 陆老师还给我们带了西瓜吃, 我们也有自信在接下来的一个月里面, 投入所有的时间和精力去准备比赛, 取得一个好的成绩。

一个月的准备过程

7月1日到7月3日

这四天内, 首先是创建了我们自己组内的GitHub, 然后把之前的版本cpu回顾一下(毕竟6月光顾着考试了, 一个月都没有看了)

7月4日到7月6日

转机出现在了4号晚上, 贺隽文这几天一直在看 A11_Trace比对机制使用说明_v1.pdf, 成功适配出了sram的框架, 简单的来说就是get_pc怎么样按照sram的规则去取指令, 适配了框架就意味着可以使用trace, 可以跑89个测例, 这个一个从0到1的过程, 然后, 贺隽文就开始根据trace仿真返回的结果, 开始对我们的cpu进行debug, 89个测例中, 前10个左右测例都是基本指令, 算数逻辑指令等等, 也不涉及到数据冲突。

7月7日到7月12日

这个时间段李航去参加了夏令营, 吴杰在写他的大作业, 我回了趟家, 所以这个时间段, 基本上是贺隽文一个人在debug, 这个时候, 出现了一个问题, lw指令的load相关数据冲突处理, 这个问题大约困扰了我们两天的时间, 贺隽文分别和我和李航进行了讨论, 最后解决方案是暂停流水线, 不进行数据前处理, 这个时候我们就对怎么根据trace返回的结果进行debug炉火纯青了起来, 1、首先根据trace的结果找到仿真波形中出错的指令在哪, 因为这个地方出现错误并不代表就是这个地方出现了bug, 也有可能是前面吃了bug导致了这个地方的bug, 所以要往前推几步来看, 然后打开test.s文件, 这是正确的cpu132的反汇编文件, 对照着这个文件搜索出现错误地方的pc, 对照指令来看我们的cpu哪里出现了问题, 是跳转没有跳转对, 还是改跳转没有跳转等等, 然后修改代码。所以等到10号晚上的时候, 我们算是过到了第37个功能点, 第38个中的BGTZ中指令跳转目标不对。(翻了翻聊天记录, 发现在10号下午的时候, 我们指令还停留在在第6条, 所以前期一个小bug被de掉, 就能往前前进一大步, 不过也是, 一旦掌握住方法, 进展还是很快的)这个时候, 同样也关注到了一个地址映射的问题, 需要做下地址转换。

7月13日到14日

7月13日，继续昨天的工作，进度到了46条，还剩下乘除法和异常处理，这一块我们还没写呢，所以当然通不过了。。。。，乘法我们直接是星号*来解决，除法一开始是想在网上找开源代码，也没有找到好的，所以直接拿《cpu》上面的除法拿来用，ok，除法解决了，进度应该是到了60几条，开始进行有关异常处理了。

7月15日到7月19日

在7月15号，李航结束了他的三个夏令营之旅，重新回归到了比赛当中，考虑到当前版本的cpu已经和《数字设计》那本书上的图增添了很多模块以及很多的控制信号，同时也为了捋一捋整个cpu的框架和架构，也为了下面为写异常处理做个准备，15号晚上忙活到凌晨三四点我和李航在A4纸上把原图打印了下来，一起用铅笔把新增的信号和模块给划了上去，重新画了一份数据通路图。

7月16日

贺隽文这边除法器借用了《自己动手写cpu》的试商法，进度我们这时是提到了64。剩下的65到75是异常处理，75到89同样均是异常处理，不过只是在异常处理的基础上要实现延迟槽。我开始接手设计。

7月17日到7月20日

关于异常处理，首先是要实现cp0和mtc0和mfco，以及解决有关的数据冲突（解决方式和mthi和mtlo类似）。cp0寄存器的实现根据A03文档中去实现要求的6个寄存器。准备工作做好之后，开始进行异常处理的工作，其实异常处理非常明确，一旦在流水线的五个阶段中，有异常出现，就跳转到异常处理例程入口地址，意思就是判断一下有异常发生时就把入口地址给PC就好了，同时就将该异常的pc保存下来，（以便处理完之后返回到这个地方继续往下执行）。

为了实现精确异常，就是将各个阶段发生的异常统一送到访存阶段来进行处理，怎么收集异常，通过给10位信号，设置标识位，发生了那个异常，就自己定义哪一位由零置为一，最后到了访存阶段，就包含了前面几个阶段所有出现过的异常，统一处理即可。

这三天的时间，我们是通过sram架构下的89条的所有指令，同时在这期间，李航把所有的always (*) 以及有关的reg信号，全部改成了assign赋值和wire信号来进行表达，我们的control_unit模块直接从1200多行缩减到了200多行，ALU等其他模块也大大简化了。也是龙芯所提倡的，cpu设计只需要掌握住三种语法就可以了，1、always (posedge clock) 2、assign 3、() ? () : () 选择

myCPU中

- **禁止initial语句**
- **禁止使用always @ (*)**
- **禁止使用always @ (a)**
- **禁止使用always @ (posedge clk or negedge resetn)**
- **只允许always @(posedge clk) ——时序电路**
- **assign ——组合电路**
- **建议：一个always只对一个变量赋值**
- **请多多使用中间变量**
- **常量加宽度： 1'b0**

这时我开始研究下板，贺隽文和李航开始适配axi框架。

7月21日到7月22日

下板流程很简单，添加个约束文件，直接综合、实现、生成bit流，烧到板子里，（后期我们很关注每次综合后的综合设计以及实现后的实现设计，不过这是后话了）按下reset键位下，拨下拨码开关后，59000059就显示在了数码管上面了，这也是我们第一次真正的下板。当时感觉还是非常有成就感的。

axi框架这一块，直接是用了比较简单的类sram转axi来进行适配的。贺隽文应该是花了一晚上的时间就搞定了类sram，毕竟从零到一的sram已经很顺溜了，然后用了龙芯提供的转axi的转接桥转换成了axi，然后花了一下午的时间重新在axi框架debug跑通了89条功能测试指令。

7月23日到7月24日

这个时间段，我们就完全抛弃了sram框架，开始在axi框架下进行测试，既然实现了axi框架，必然是为了开始准备做性能测试了，我们熟悉了性能测试说明文档，按照文档步骤一步一步添加coe文件，进行仿真，着手下板，把下板的结果输进excel表格中，记得我们第一次没有经过任何优化的性能分是0.989，当时能做到这一步，成就感满满，按照当时大赛群里面的讨论，说是在裸cpu的（不加cache）的性能能够上1分，说明性能还是很不错的。这个时候算一算，离大赛提交还剩两周多一点的时间，这个时候我们很有信心利用这段时间去优化和改良我们的cpu。

7月25日到7月29日

其实优化的思路很明确，一是提高主频，二是增添cache，怎么调高主频，优化数据通路，怎么优化数据通路，找到数据通路最长的那一条，把它改短，然后接着找下一条，怎么找到最长那一条呢，这个时候就要关注vivado提供的open synthesize design 和 open implementation design了，这个时候需要熟悉vivado这方面的操作。怎么分析最长路径，怎么改短，我们改短的思路主要有两个，在我们最长路径中，乘法模块占据了半壁江山，原因如下：FPGA设计中虽然可以直接使用乘法*，这是因为vivado会帮你默认调用乘法的ip核，（类比考虑一下除法，除法就只能一位一位的去除），所以相类比，我们自己添加了一个乘法器，不直接使用“*”。

修改完这个问题之后，继续优化，这时候问题接着就出现了，在处理load相关这一条路径上，我们的解决方法是，在阻塞方式上，为了数据避免从访存到执行再到译码此类连续重定向而时序过长，以及多个指令同时被依赖时丢失数据的情况，我们设计了双优先级的综合阻塞方式。其中以取值、取数的等待暂停为高优先级，而译码阶段指令依赖于尚无法得到结果的执行阶段指令（如访存、cp0、hilo）的阻塞为低优先级。

与此同时，还有几个指标需要关注，在看实现设计报告的时候，关注一下，实现的比如LUT等逻辑门这些元器件的数量，我们LUT大约是2000多个？，还是3000个？具体数值忘了。

最关键的一步来了，在进行优化的过程中，什么指标代表着你是在往好的方向发展，而不是越优化越差呢，看实现后的wns，（不是综合后的wns，不过可以作为参考），wns不能为负数，这是大赛要求，为负的话就代表不符合大赛提供的约束文件的时序要求。

我们在修改完一步后，就会进行实现一次，看wns是正是负，然后调主频，提升主频，一直提升到wns为负的情况下，这个过程中会有很多次的综合与实现，综合实现一次大约要半个小时，不过不需要干什么活，只需要每实现一次后，看wns正负，调下主频，然后重复。

原始cpu，大赛统一的为50Mhz，我们的优化过程是，修改完乘法后，主频提到了60Mhz，修改一下乱七八糟的代码，整合整合呀，主频提到了80Mhz，修改完load相关这一条数据通路后，主频提到了120Mhz。

此时，我们的主频这一方面上的优化也就搞了一段落，在120Mhz的主频下，我们进行了一次性能测试，性能分为1.288 对于这个结果，我们觉得还是挺OK的。

7月30日到8月1日

原计划是在接下来的一周时间内，我们自己写出来一个cache，因为自己写的cache要比vivado提供的ip核cache好上太多了，但不知道怎么回事，可能是大家太累了，在一个月的时间，的确大家很不容易，一直神经都绷着，而且隔壁组在暑假期间也都回家了，贺同学说他要回家过两天，然后吴同学也回家了，这样一看，我和李同学也动了回家的心思，哎，很是挺遗憾的。如果不回家的话，是不是cache就能自己写出来了？是不是性能分就能够挺进决赛啦？不知道。然后贺同学和吴同学就先走了，剩下我和李同学在回家之前开始准备往我们的cpu上面加cache。

加ip核很简单，直接在顶层文件把cache的ip核提供的接口给对上就可以了。然而这个因为我粗心花了我们一天的时间才接上，终于这个cache在我们各回各家、各找各妈前给接上了。但最后的结果，加上cache的性能分是多少呢？我们这时还没有测试。

8月4日到8月6日

我在家呆了三天的时间，回到了学校，这个时候离初赛提交满打满算还有3天的时间，这三天时间，写design设计报告，整理要上交提交包。不过此时下他们都没回来，此时就只有我一人了。

首先是看下加上cache的性能分是多少，结果出来是8.826，这也是我们最终的提交时的性能分数了。当时按照这个分数，觉得还可以，应该能进决赛，但还是事与愿违。

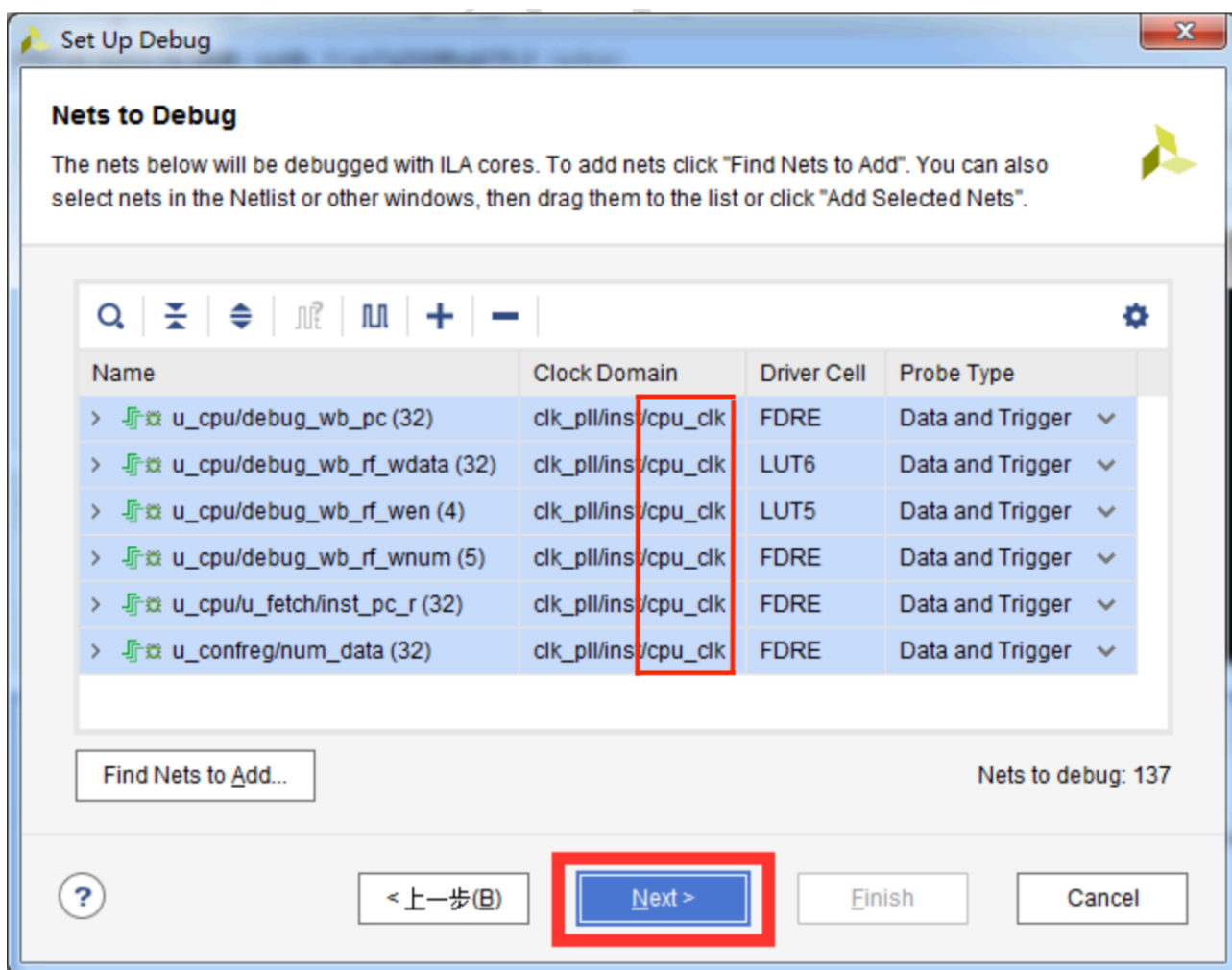
设计报告，我们通过小组合作的方式完成了，每人写一部分，然后我来整合。

整理提交包，这个过程花费了我一天的时间，因为在经过很多次版本迭代之后，有很多乱七八糟的文件在我们的文件夹中，这还好说，按照大赛文档中要求，自己添加到的ip核要放在myCpu文件夹的模块下，但大家也知道，vivado默认IP核路径肯定不是这，所以我的方法是把先工程中的ip核删掉，然后在重新添加（因为只能在重新添加的时候才能设置路径）。这个过程不复杂，但是很繁琐，到后面我心态还有点小崩。

这个时候就不得不提一个从我们适配好了axi框架后就一直困扰我们的一个bug。axi功能测试框架下，89个功能点仿真没有问题全部通过，但在下板的时候，却有一个功能点没有过，在数码管上显示的是59000058，真的是，我现在看到这串数字还是感到一阵反胃。仿真通过，下板没有通过，只能通过ILA逻辑分析仪来解决，具体环境的配置看A10_FPGA在线调试说明_v1.00，这里有一点需要提醒一下：

红框中，我们在自己设置中会出现sys_clk，注意要全部是cpu_clk，否则会出现了两个调试core。就不是正确的ila调试环境了。

我们处理的方法是，我们在进行板上ila板上调试的时候，通过观察波形，我们的cpu 给出一个正确的pc，一路一路追查下去，最后返回给我们的指令却是一个乱码的指令，而且整个inst_coe文件中也只有这一处有问题出现，所以最终我们是做了一个特判处理。在这个pc下，不采用sram传给指令，而是根据.test文件反汇编文件特判使用正确的指令。加入特判后，这个bug就被de掉了。但是功能测试和性能测试的coe文件是不一样的，所以在性能测试下就把这行特判注释掉了，所以这个问题很奇怪的地方就在于在功能测试下89个功能点仿真能过，在性能测试下仿真下板



也均没有问题，偏偏只有功能测试下板时会有问题。我们也是很苦恼，也不知道怎么从根源上去解决掉这个bug。

所以问题来了，在我已经把提交包都已经整理好了，把多余的生成的工程文件都删除掉了，这个时候记得要重新打开xpr文件，看vivado会不会报error和warning，此时只剩下了一个7M左右的压缩包，模拟上交的提交包再重新跑一遍，性能测试没有问题，很顺利跑成成功了，然而功能测试又出现了59000058，这个时候我就超级慌了，原因分析可能是加了cache之后，cache会改变pc，所以以前的那个特判就没有了。需要找到新的特判。离提交截止时间也没差几个小时了，又出现了这个问题，而且当时情况是只剩我一人了，我的组员们都在家，当时情况是两种解决方案，一是如实提交上去，一个功能点没有过，按理是功能分是可以得99.5分，二是在剩下的几个小时里面重新配置ila调试环境，可以光配置ila调试环境就要一个多小时，先不说找特判值的时间，即使找到之后，还需要再综合实现生成bit流验证是否正确，加了cache之后的综合时间达长一个小时。我很是担心时间的问题。而且此时还有一个顾虑，说明文档里面明确说明，功能测试和性能测试的内容应该是完全一致的。很显然，我们的功能测试和性能测试代码虽然只相差那一行特判，但还是不一致，很担心会给我们的分数带来一定的影响。

但是，更严重的是，如果功能测试都没有到100分的话，我怕龙芯的人连性能测试都不会给你复核了，而且，没有100分了，那决赛是铁定进不去，连一丝希望都没有。所以最后还是我和李航两人通过视频，远程交流，我来配置环境，在我这边看ila波形debug，在他的机子上跑综合实现生成bit流，把生成的bit流发过来在我这烧到板子里面验证。同步进行，争分夺秒，最后成功交了上去。

总结

为什么像记流水账似的去写一篇这样的总结的呢？前边还好，时间单位还是月份，特别是到了准备比赛的一个月中，时间单位都具体到了几天几天了。

因为我在准备这个比赛的时候，心中最大的惴惴不安是未知，一是未知我做到哪一步是完成了比赛的要求，二是我未知哪个时间段完成是到哪一步是合适的，我的心中是没有底的。前者还好，可以参考学长，明白要先实现sram，后跑89条指令，在适配axi，重新跑89条指令，然后运行性能测试。然而感觉前面学长和我们走的路我觉得差距还是蛮多的，包括我现在以过来人的身份去看他们的总结，我还是有些懵。我希望学弟学妹们在看过我的总结之后，能在时间上的把控上有一个底，在哪个时间段能够做到哪一步，或者以我们为反面教材，因为我们花了很多时间浪费到了没有用的地方上。

所以我们说我们真正开始把事情做起来也就是7月份这一个月，另一组在这一个月的时间也就写出来一个IP核，所以希望学弟学妹们能够抓紧时间，不要懈怠。但今年做出的进步还是有些的，我们组把主频提了上去到了120Mhz，另一组是写出了ip核，学弟学妹可以吸收各自的长处。

不过，这只是最基本的，比赛的水平是水涨船高，而且形势是非常严峻的，从上一届的性能分是0.8几就能进决赛到今年的最后一位决赛名额的性能分数是24.5分，差距是非常大的。

建议的时间安排是：在大赛建议上有个维护的github上，有对时间安排的是十分合理的，能在暑假假期前出一个性能分，多低都无所谓，然后花一个月的时间去提主频，加cache，考虑双发射、分支预测、等进阶提高性能的方法。

一点建议是：尽早熟悉大赛框架，但在初期写一个cpu的时候是没有必要的，基本上在57条指令写的差不多了，再加在大赛框架下，但这个过程还是要尽早，这样就不会出现我们花了很长时间去想soc是什么呀，怎么写soc呀，sram接口是什么呀，其实这些你都不用操心，大赛这些都给大家提供好了，搞清楚大赛提供的文档基本上就完全ok了。完完全全需要自己做的也就是写个cpu罢了，也就是几十个.v文件，其他的都不需要你担心。

还有就是：doc文件夹下的每个文档都是十分重要的，一句话一个字一张图都不能漏过去，每个图每段话都有被用到的地方，一定要搞透彻和想明白。

思考

其实最近一个问题我一直在思考，就是优秀的人和平庸的人的区别究竟是在哪里，不知道在哪里看过一句话，或许可以解释这个问题，一眼可以看到本质的人过的人生和花了半辈子才能看到本质的人过的人生肯定是截然不同的两种人生。这里本质泛指一切，包括任何领域，学习呀，工作呀，爱情啊，但我觉得能够有一项就很厉害了，比如学习上，高中前我们可能只是麻木的去执行老师给的命令，（当然如果完不成就更算了，这又牵扯到了执行力的问题了）却不曾想过命令背后是什么样的，厉害的人就会想这个是不是适合我。我该怎么去调整这些来适合自己。

但如果看清了，但又没有执行力，也就谈不上优秀，所以想要优秀还是很不容易的，但又该怎么去训练，怎么去成为，这又是一个要摸索的过程。又很可怕的一点是，优秀的人会觉得这种事情理所应当啊，本来不就应该这样嘛，为什么要去训练？

尾

写完这篇总结，有一点写后感吧，就是写是整个创作过程中最简单的一件事，因为在写这个过程中，你是有东西的，是有激情的，是想把自己说的话娓娓道来的，而且觉得有些地方写的不好是可以略过去的，但是写完之后的重新组织、排版、修改错别字，最后呈现出给一个井井有条的样子的这个过程是枯燥的，是你不愿意做的。也是最乏味的。

像极了我们写代码，码的过程是最舒服的一个过程，大脑里面有思路和架构就可以码了，但码完之后的debug，各种测试过不了的烦躁，是很难受点。所以就像老舍先生说的，文章不是写出来的，是改出来的。套用一下，代码不是写出来的，是de出来的。

同理，这个道理也适合本次大赛呦，反汇编文件在（nscsc2019/nscsc2019_release_v0.01/func_test_v0.01/soft/func/obj / test.s）

附录

觉得特别有用的网站：

https://www.eg.bucknell.edu/~csci320/mips_web/ MIPS Convert 超级好用的转mips指令为16进制 或者是16进制转mips指令，在对着test.s debug有很大用处

<https://www.jianshu.com/p/8514c7998227> 与我们架构图类似的一个简单版本的cpu，可做参考

<https://blog.csdn.net/tangyuanzong/article/details/78595854> 一个ucore的实验报告参考

<https://mubu.com> 幕布app，大纲自动生成思维导图。