2019 龙芯杯总结

范志鹏

目录

1. 萌生写 CPU 的念头

参加这个比赛以及种种之后发生的事情最终都追溯到一个最开始的念头——自己写一个 CPU。当时是大三上学期，我是信息与电子学院的，学业不是很紧张但也不是十分轻松，于是就想给自己找点事情做。在大二暑假参加过一个嵌入式的比赛，恰巧当时用的平台也是龙芯的开发板，当时是写了几个 linux 驱动，但是因为种种原因最后取得的成绩也不是特别好。

包括我之前参加的一些电子类比赛，我个人觉得其实要是真想取得非常好的成绩，是需要在除了编程以外花很大功夫的，要不然你的展示效果不好。比如机械模块的制作，PCB 画板等等。另外，参加这些比赛感觉自己学到了一些东西，但也没有太深入。比如写传感器的 Linux 驱动，我也是照葫芦画瓢，一个驱动洋洋洒洒几百行，但至于为什么要注册字符设备，为什么这里要有个 MODEULE_INIT，我却不太明白，只是看别其他的驱动有，我这里也有。

当时我也想搞明白，深入下去，但是这样比赛成绩就肯定不好看，所以我写完那几个 linux 驱动之后就想之后不要参加比赛了，安安静静钻研下知识。当时我也搜集了一些信息，最后确定的目标是是写一个 CPU，或许也还有 OS，编译器。后来是偶然看到上一届朱威浦和江学谦学长他们写的心路历程，于是就决定写 CPU 了，并且和陆老师发了邮件，表明我希望参加比赛的心愿。

翻了下邮箱，那是 2018 年的 11 月 25 号。或许和我之前表明不想参加比赛有些矛盾，但其实并没有。我从一开始就没有抱着获奖的念头来做事情，而是觉得我想写 CPU 正好有老师和其他同学一块更有干劲，于是就参加了。

2. 第一次组会

当我参加的时候，其他的同学应该已经有了一些基础都互相比较熟悉，而且我和他们不是同一个学院，这些参加比赛的同学我一个也不认识，所以第一次还有些紧张。当时我记得和朱威浦学长发微信，他告诉我每次都要进行汇报，吓得我赶紧把 patterson 那本书关于流水线旁路那块看完了，然后最后还好因为我是新来的，所以不需要汇报。

我之前用过 vivado，也写过一点 verilog，IP 核也用过，不过大部分是和通信有关的，比如调制变频同步解调等等，所以也不算有什么基础。而且那时候我学过关于计算机的课只有 C 语言，Python 和 C++与数据结构，其他涉及系统体系方面的知识完全不知，或者说靠自己看书学了可能连皮毛都不算的知识。

第一次开会，大概就是分配了一些任务，然后我和王赞还有几个延安大学的同学分到了一组，那一次大概就是熟悉一下要干什么，印象中任务主要就是写一个单周期的 CPU，只包含几条指令。

3. 期末考试进展缓慢

一开始我也不是太懂，一边看书看代码一边摸索，然后由于快要期末考试了，所以花在这上面的时间和精力也不是很多，不过磕磕碰碰还是完成了几条简单的指令，印象

中是 7 条，add, sub, or ,and, lw, sw,和 j，因为当时认为这几条就覆盖基本上所有的指令了，当然后来发现还远远不够。

单周期完成之后，大家也进入了紧张的复习阶段，最后开会确认的目标是希望寒假写完流水线，于是等考试之后放假了我才开始又把精力放在这上面。

4. 三天速成流水线

放假后，然后我又因为有别的事情，所以真正开始看流水线和动手写，大概也是放假 1 个星期了，大概 1 月 20 多号。我记得当时是反复看 patterson 那本书流水线那块的知识，我记得很清楚，有好多张架构图，这对帮助理解还是很好的。看明白之后，就自己开始写，第一次写没有借助太多的代码，基本就是看书，看图，写完之后发现关于流水线寄存器那块的 verilog 语句还是有点疑惑，看了《手把手教你写 CPU》那本书之后明白了得用时序逻辑（虽然当时可能也不是特别明白）。【简单来说就是 always@(posedge clk) 可以刷延时】

一气呵成，三天搞定，应该没有夸张，我记得当时写完整个工程也就不到 1k 行，当你把流水线，旁路，阻塞搞懂了，就其实没有那么难了。写完之后自己写了几个小测试仿真了实现的指令，基本对了之后感到很开心，于是就把这事搁置一边了。

5. 讨厌的延时

过了春节之后，想起学长说的搭 SOC，就是 CPU+ROM+RAM，我就开始了搭建这个简单的 SOC，因为用过 IP 核，coe 文件也知道怎么回事，所以搭建过程倒是没太费事，但最后搭建完之后，发现了一个问题，就是从 ram 取指令/取数(应该是 bram)，会有一个周期的延时，当时就觉得这个延时很糟心。

因为这个工程已经变得比较大了，1k 多行，如果要适配这个延时，就需要改动整个架构。但是当时的想法是如果为了这个延时改架构，之后可能又要再改，因为这个延时是固定一个周期的。改吧就感觉可能会做无用功之后还要再改，不改吧进度就只能停下。当时我问了下王赞，也可能是他找我，问一些问题，然后我们就交流了下，好像是他也卡在延时这块。我就想，那就先这样吧，等开学再问问学长怎么搞。

6. 解决——DRAM

开学之后第一轮汇报，发现大家的进度不太相同，我和王赞都开始做流水线，于是我们的交流也多了起来。当时向学长请教得到的答复是可以先用 DRAM，然后学长他们讨论还说可以直接上 AXI（江学谦），不过 AXI 我之前了解过，但是没太看懂，就想还是先把 DRAM 实现了吧，看看效果。

于是把 ram 改成 dram，发现果然延时没有了，结果正确无误，我借用老师的 fpga 板子，用内存接 led 灯测试，下板了几个小用例，发现没问题，很开心地把结果记录在 wiki 上了。

进展汇报 2019.3.11

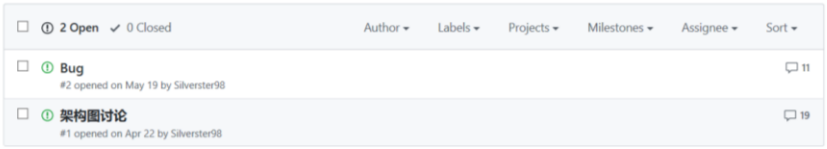
姓名	进展	下一步计划
范志鹏	使用内存连接led的方式，下板测试基础的SOC(五级流水CPU+指令ROM+数据DRAM)	修改架构完善更多指令如中断、自陷等

不过可惜的是，我没有把这个习惯坚持下去，因为我看到别人都没有记录，感觉自己把过程放在 wiki 上有点怪怪的（这或许只是个懒的借口，因为我本地也没有记录）。

7. 1 + 1 = ?

接下来的一段时间，再次陷入僵局，现在小组基本上就是我和王赞在讨论，进度也差不多，但这个时候我们还是各自为战，自己维护自己的 cpu。我内心是想一起写的，但是就感觉有点无从下手，因为各个模块各自有各自的想法，光是命名规则就很不统一。如果粒度细化到每一个寄存器名字，就有些太麻烦，如果粒度太粗，merge 的时候可能又有问题。

最后是王赞提出用 Process on 这个在线画图工具，把各模块接口都定义好，规范名字，都弄好之后这个时候已经是四月中旬了。



8. 拓展指令，完成 cp0，适配 sram 接口

这块记不太清了，所以统一写一下，看了下 github 上，第一次 commit 是 4 月底，这个时候大概就是在画 process on 和写流水线，我就写了 ID 和 cp0，其他的都是王赞写的。

我本来是想用学长的 cp0 的，但是王赞极力说服我自己写，我就照着上一届的 cp0 写了，不过江学谦写的很专业，而且那里面还有 tlb，我当时就把 cp0 部分挑出来，重构了下，印象比较深刻的是 function 的用法，感觉有点写 C 的味道，我挺喜欢，就学了过来。【江学谦写的 cp0，tlb 等很专业，不过看懂也比较麻烦】

完成 cp0 还是花了一些时间的，因为涉及到异常的处理，我自己写的 bug 也有一些，流水线也有一些。【主要就是异常那块，记录的 epc 得正确，尤其是有延时槽的情况】我们当时是用自己写的一个 cp0 的测试，虽然很不全面，但是那个用例过了之后，基本上 cp0 就没有大的问题。

于是我们开始适配 SRAM 接口，这个时候老问题又来了，还是那个多一个延时的问题，最后我们解决的办法是增加 prepc,premem，可以看作 7 级流水，可能是这样，但我记不太清了，也可能就是干脆多等一个周期。总之这不重要，因为最后跑 AXI 比较关键。

这部分主要都是王赞的功劳，我对流水线暂停那块至今仍然有点糊涂，跑通 sram 之后就开始适配 axi 接口了。

9. 一个星期搞定 AXI

Sram 跑完下板都 ok 之后，我就开始着手写 axi 的接口了，龙芯官方提供了一个类 SRAM 到 AXI 的桥，我们还需要做的工作是写一个 SRAM 到类 SRAM 的桥。

那个时候我比较闲，课很少，没有一门期末考试，所以很快就搞定了，当然也没那么快，因为我有点忘了状态机咋写了，临时复习了下，并且部分参考了学长的代码，然后把 Inferring latch 的警告消了【关于这个警告，朱威浦写的总结说明得很清楚，大致就是 always @(*) 这种组合逻辑没有给全所有情况，会综合出锁存器，布线的时候会有问题，警告一般不是很重要，但是这个不能有，需要留意一下】。

我记得跑完 SRAM 到跑完 AXI 就用了一周，因为当时我以为 AXI 很难搞定，结果发现龙芯给了桥了其实自己需要搞定的不多，AXI 下板全部通过是 6 月 6 号。

10. 忙碌的实习

由于我之后可能要准备出国，于是打算先实习一段时间，也算可以增加一点个人简历之类的。从 6 月 10 号开始，我就开始了忙碌的实习搬砖生活，每天早上 8 点出宿舍，吃完饭做一个小时公交车到上地上班，晚上平均 8 点回到学校。这段时间还是比较累的，而且实习工作也不轻松，所以其实之后这段时间我一周投入的时间可能也就不到 20 小时。不过我还是尽力最后在王赞的配合下调通了 cache。

刚才说了 6 月 6 号之后到七月出这段时间其实我们基本上什么事情也没有做，王赞在准备考试，我在实习，周末也就歇着没再继续完善 CPU 了，6.6 到下一次 commit 已经是 6.28 号了，等王赞考完试之后，我们商量还是要自己完成一个 cache。

11. cache cache cache

去年学长用的是 system cache 这个 IP 核，看他们的过程，中间也有写 cache 但最后没有成功，就觉得 cache 应该不好写，这个时候王赞身体也不太舒服，他回家进行了休息，我也只得是硬着头皮强行写 cache。

记得那段时间很痛苦，睡觉的时候做梦都是实习的工作和 CPU 的 cache，每天想调 cache 但又要实习，等下班回到学校又是身心俱疲，到实验室最早也是 7 点多，调一调就到了 9 10 点。当时的感觉就是很难受，写 cache 这个进程都很重要，但是我不得不分配这个进程很少的时间片，调度到它之后过一会就又要让他进入等待队列。

当时真的接近崩溃边缘，尤其是调数据 cache，我真的一度放弃了，然后拜托王赞帮我看看 bug，其实我那个时候对这个数据 cache 已经没有信心了，就开始想把系统 cache 调好。IP 核加的倒是很轻松，但是性能也只有 5.7，还是不太够进决赛感觉，不过好在王赞发现了些问题，他回老家休养同时也帮我改了改。虽然还有问题，但是起码让我有了些动力。

12. 肉眼 debug

这个时候其实 cache 加上接口大概还有不少 bug，我通过关 cache，关数据 cache，开指令 cache 的方法，把一个接口输入输出写反的 bug 排除了。【这种情况在龙芯官方给的文档也有写，现象就是就是仿真通过，下板全 0】然后开数据 cache 下板发现只有

一两个性能通过，其他的性能有问题。

每次下板都不停祈求，希望可以通过，但是每次都是铩羽而归，内心就还是有些动摇，加上王赞身体不太舒服，我也只能是继续孤军奋战。

那段时间有几天我都是我们宿舍第一个起床，最后一个回去，周末也经常不休息，不过可能也是由于身体比较疲惫，状态不太好等等各种原因。进展非常缓慢，一直没有发现问题。

转折出现在某一天在实习，那时我对着 2k 行的 data cache 肉眼 debug，当时我没有性能仿真，也没有单独进行下板，因为就算前一天晚上仿真完了，再分析又要下一天，我很不喜欢被打断然后再继续一件做很关键的事情。对于一个 OS 来讲可能这就是一个普通的 context switch，但可惜我不是计算机。

这个 data cache 虽然有两千来行，但其中一千多行都应该是王赞用脚本生成的，就是每一个 cacheline 的读写，都有一个模块，虽然可以更简化一下，但是我实在是不愿意重构了。于是就把精力放在其他的代码上，看是不是哪里写错了。

肉眼 debug 其实就相当于是我们考试时候的检查，这个检查相当于是反思，主动的反思。然后我灵光一现，突然发现了一个愚蠢的错误，大概就是某一个 case()里面，有四种情况,2'b00->2'b11 这四种情况对应四种类似地操作，好像是 find_set,那么理论上 2'b00 对应 find_set0，依次类推。然后结果 2'b10 和 2'b11 对应的也是 find_set0，大概就是复制粘贴的时候，忘记改了。这个毛病也是之前出现过，很低级，哎!

13. 山重水复疑无路

改完这个 bug 之后，我很激动，虽然在公司，但是已经迫不及带地想做公交车回实验室下板了，然而，现实再一次打击了我。

当我改完 bug，综合，实现，生成 bit 流一步步地进行下去的时候，内心既激动又忐忑，害怕再次失败，又期盼奇迹发生。没错，当时我内心就是觉得调好就是奇迹了。

下完板子，按了下 reset，发现全零，内心开始慌了，然后发现忘记调拨码开关了，拨码开关 0000_0000 没有对应的 test。0000_0001 才是第一个，我调整了下心态，内心先祷告了一番，然后开始试第一个，发现 ok，第二个，ok！… 一直到第五个，当我按下 reset 的时候，一红一绿的 LED 灯再次宣告了我的失败。

最后发现有三个没有过，内心很难受。不过想想已经从之前的错 9 个，到现在的错 3 个，可能也算是进步了吧。

14. 柳暗花明又一村

这个时候我决定开始仿真了，这个时候我发现，我单独下板第 8 个(好像是)没问题，但是联合下板的时候就是错的。最终我把问题定位到 quicksort 那个测试，因为只有他单独下板和联合下板都是错的。

然后我进行了仿真，至于为什么先下板后仿真，我是觉得，下板正确是仿真正确的充分条件，但是仿真正确不是下板正确的充分条件，所以我先下板，对于那些本来仿真和下板都正确的会减少依次仿真时间，但是对于那些下板错误仿真错误的会多一次仿真时间。选择先下板还是先仿真见仁见智了，倒也不是什么重点。

总之我发现 quicksort 是现在所有用例中唯一一个仿真错的，其他两个都是仿真 ok

单独下板 ok，联合下板错。那么我就想找出这个 quicksort 的问题所在，于是我就想搭建一个性能测试的 trace，但是当我搭建完之后，我跑了一个多小时之后发现生成的 trace 已经有 30 多 MB(纯文本)，但是仿真结束还是遥遥无期。我就想要不就先看前面这些，应该已经足够多了，不过恰好那天已经很晚了，我就想第二天再说了。

第二天的时候，我在实习趁着休息的时间，继续肉眼 debug，虽然说已经生成了 trace，但我心里就是痒痒，明明知道看代码调 bug 是自讨苦吃，但还是想赶快完成。结果，又是灵光一现，发现了问题所在。

这个 bug 价值还是比较大的，就是我们这个 cache 采用的是 burst 传输，并且实现了 dirty 位。Bug 在于：当 cache 执行一个内存读的操作时，这个时候假设这个 set 的 cacheline 满了，如果 dirty 位为 1，cache 会先把这行写回内存，再从内存读 lw 地址对应那 16 个字的数据。如果 dirty 为为 0，就直接从内存读 lw 地址对应的那 16 个字的数据。

目前为止没有问题，我的 cache 也没有问题。那么这个时候由于是读操作，所以 dirty 位是 0，但是问题是如果下一个 sw 指令，命中了 cache，也应该改变 dirty 位，但是我的 cache 没有改变，只是在 cache miss 的时候考虑到更新 dirty 位，在 cache hit 的时候没有更新 dirty 位。

而且这个 dirty 位实现的还有问题，就是他第一次被置 1 之后，之后就再也不会被置 0，虽然这不会带来问题，但性能上会下降，因为相当于你没有实现这个 dirty 位。这实际上也解释了为什么其他 7 个 test 通过，我这里就不解释了，因为深究也没必要了。

好吧，当我发现这个问题之后，心情虽然也很激动，但是也没有那么激动了，因为感觉可能还有问题，不过下板的时候发现竟然所有的都通过了。虽然性能第一次只有 7 点几，但也很开心了，毕竟自己的 cache 能用了。

15. 多拍延时

实现好这个 cache 之后，我和王赞就想法设法继续提高性能，现在这个 cache 在命中的时候，还是个多拍，准确的说是四拍，为什么有这么多拍，大概就是 cache，流水线设计的还是有比较大的缺陷。

经过优化我们改到了三拍，把频率提到了 88Mhz，这个时候是 14 分。经过这一系列的折腾，我确实内心不是很想再改 cache 了，最后几天就也没再做什么。

16. 另一组 115M 主频

另一组主频做到了 115MHz，让我们很感到惊讶，于是我们问他们是怎么做到的，他告诉我们关键路径是旁路那块逻辑太多了【这应该是一个比较共性的问题，主要在于 ID 级旁路需要 EX,MEM 的，然后比较旁路之后还要比较大小(比如 beq 这种)，总之最后 pc 的逻辑很差】，我当时看路径分析有点懵，他们这么一说，我发现还确实如此。这部分是他们队的贺隽文负责的，他的解决方法是遇到旁路的情况，就改成阻塞一个周期，我觉得这是一个方法，我也想过，但是觉得意义并不是很大。

为什么这么说呢，因为我们的 cache 是用 reg 写的，1KB 指令 cache，1KB 数据 cache，implementation 之后 FF LUT 都在 3W 左右，当我把频率调到 90Mhz 的时候，cache 这边的路径也有问题了，之前没加 cache 的时候，调到 100Mhz，其实 wns 为负的

也只有那么几条和旁路有关的。甚至我把指令 cache 加到 2KB，88Mhz 综合都有问题了。所以说我们没有再更改流水线了，这也是王赞的想法。

17. 遗憾落选

做到 14 分的时候，参考去年榜单我们组应该也有个 10 几名，我想进决赛应该问题不大，而且其实我内心对进决赛这个事情就没什么念头，我本意也就是体验这个过程，当然顺便拿个奖自然更好。不过最后发现其他组的实力都很强，从公布的榜单上可以看到我们和决赛的最后一名还有不小的差距，但我觉得也没有什么特别的失落，毕竟我们已经做得还可以了，实现了 cache 已经是一个不小的进步，投入了有限的时间，做到这个程度倒也还可以了。

18. 继续维护

没有进入决赛后，陆老师和向老师和我们两组同学召开了一个总结会，意思就是说希望我们可以不要比赛结束就不管这事了，如果可能继续做下去。

我其实和向老师想法有一定吻合，因为我接下来就打算开始写 OS 和编译器，为了降低难度，我打算在模拟器上实现。但向老师还是想让我们继续就这个平台做一做，王赞也想跑起 ucore，他已经完成了 tlb 和并通过了龙芯的 tlb 测试，我之后和王赞沟通，表示我也可以打打下手，硬件部分我不想过多参与(虽然现在看来还是参与了)，软件的话有问题可以一起商量商量。

19. 却顾所来径

回首这一路，我觉得收获还是很多的，其他参加这个比赛的同学也都有这个感觉，对于我来说，我最初的写 CPU 的目的已经达到了，并且还多完成了个 cache，虽然性能差一些，不过由于我个人能力有限，加上对处理器这块也没有特别的追求，所以结果这样也是意料之中。可能之后大四还会继续维护软件，现在已经跑起了 pmon90%多，但是还是有比较大的困难，希望可以慢慢解决。

特别感谢王赞同学，没有他的帮助，我们不可能最后完成这样一个 CPU；感谢朱威浦，江学谦两位学长，给我们提供了很多有价值的参考；感谢陆慧梅老师，向勇老师，王娟老师的教导!

2019.8.20