32 位 CPU 计软联合实验组

总结文档

Totoro 组

杨乐 琚锡廷 董原良 李林涵

2019 年 1 月

1 迭代开发计划

我们组的开发计划参考了学长的一些报告，并提取了其中的步骤要点，加以总结而成：

A. **Sprint1**/需求分析：分析 ucore 操作系统及需求（MIPS、流水线、操统）

B. **Sprint2**/流水线搭建、异常处理、访存实现、仿真测试：流水线搭建，异常与中断的支持，MMU 与 TLB，SRAM 控制器

C. **Sprint3**/联合测试、上板子：流水线、异常处理、访存联合调试，并上板子通过测试测例

D. **Sprint4**/操作系统测试、外设支持：根据操作系统进行调试，添加对 Flash、VGA 的支持

E. **Sprint5**/额外功能：改写 CPU 与操作系统使其支持额外功能

开发工程中遇到的问题

在开发计划中，我们做的不好一点是错误估计了阶段的时间。我们在阶段 D 浪费了大量的时间来 debug，直到最后 3 天才把操作系统跑通。直接导致了最后一个阶段十分匆忙，只弄了一个 VGA 和一个小游戏。

问题主要是由于前期的计划进度比较平缓，顺利，所以到最后错误估计了硬件调试的工作量。软件 debug 和硬件 debug 是十分不同的，我们应更加细分的划 ddl。由于这个项目是两个星期一次验收，所以也不可避免的造成了项目的拖延，前一个星期不干活，后一个星期只能刷夜的困难。

其余方面我们回头反思，其实在一开始的话没有知识也没有精力搞懂整个项目的计划，也和我们需求调研不充分有关。直到项目的后期，我们才算是把整个项目要做什么比较清楚了。

制定与执行注意要点

每个 Sprint 的时间：在定每个 Sprint 的时间时应该更加具体、更加细化；相对于一个笼统的大概目标，细化的、可执行的小目标会更助于计划的稳步推进。

工作量的预估和调整：在预估工作量时，需要动态的进行调整。硬件编程所遇到的问题不同于软件编程，我们也缺乏这方面的经验，所以最好能够抓紧时间进行测试；在遇到考试和大作业冲突时，应放松时间的期限；而在大家都有空的时候，最好能够进行集中编程。组长在其中的作用是重大的，应保证同学能够利用适当的时间与精力完成此项目。

2 分工

小组的分工如下所示：

杨乐（组长）：前期需求分析，操作系统解读与修改，操作系统测试，文档撰写，进度掌控

李林涵：前期需求分析，文档撰写

琚锡廷：流水线，异常与中断支持，CPU 测试

董原良：串口，SRAM，MMU(TLB)，Flash，VGA，硬件调试

我们分工的思路是，由琚锡廷同学负责 CPU 除访存模块，董原良同学负责外设部分以及与访存模块与 CPU 衔接，杨乐同学负责操作系统；前期的需求和设计文档由李林涵和杨乐两位同学负责；项目的整体进度则由杨乐同学控制。

3 测试要点

3.1 使用在线平台进行评测

课程项目提供了在线板子平台，大大降低了调试的成本。与本地板子相比，在线平台的优点是：在断电后、或是在图书馆等不方便携带板子的场所可以进行调试、步骤较为简单、并且有直观的串口交互界面。缺点则是在网络不稳定、或是同时有多组使用的时候会有卡顿的情况；十分感谢助教提供了这个平台，让后期调试的负担大大降低。

同时，对于远程 JTAG 调试看波形这个功能也没有尝试，也希望这个能够在后面被派上用场（我们 JTAG 调试还是依赖于本地的 JTAG 线）。

3.2 本地 JTAG 调试

由于电脑的一些奇怪问题导致无法连接，我们迟迟没有启用 JTAG 调试，现在回想也是浪费了大量时间。我们组内一台机子可以连 JTAG 但综合速度十分慢，一台机子速度快但无法连接 JTAG。最后张宇翔助教为我们提供了一种方法，可以让慢机子连 JTAG 并充当服务器，让快的机子去远程连接慢的机子。这种方法最后实现成功了，也让我们一下子顺利地找到了困扰我们好几天的 bug。

3.3 为何不用 CI

根据软件工程的理念，我们应该采用 CI 来做持续测试。但最终我们没有使用的原因为：

A）在流水线阶段，实现结果比对较为复杂。由于需要对波形进行比较，涉及到了 Vivado 的一些功能；由于之前没有考虑这方面的测试，并且了解到其他小组的方法后已经过了这个阶段，故 CI 测试也搁置了。

B）在联合测试阶段，助教提供的功能测例很好的完成了测试的功能，我们也无需开发出一套新的测试工具。

C）CI 服务器不稳定，当我们前期配置好了 CI 后，过了一个月后期想要使用的时候发现挂了，而且原因也十分奇怪；并且由于开发多为串行开发，在本地机子上进行编译会更加快，故在服务器上编译对我们小组是一个伪需求。

D）开发迭代少，在后期联合测试时，bug 已经很少，对 CPU 也不会轻易修改，有时一天只调一个 bug，那么顺便在本地跑一次功能测例的时间是可以忽略的，并且在本地调试也更为直观，故 CI 服务器我们并没有充分应用。

当然此项目作为软件工程联合项目，最好能够使用到软件工程的开发思路；我们也希望日后 CI 能够用一种更科学、更方便的方式为本项目提供帮助。

3.4 折衷的测试方法

在 Flash 模块没有完成时，我们采取的方式是通过 bin 文件直接写入 sram 进行测试。由于在 ucore 编译时最后得到的是 elf 文件，没有编译出 bin 文件，我们仿照的是功能测例中的 makefile 中的命令，利用 objdump 从 elf 文件得到 bin 文件完成测试。

3.5 测试过程问题的解决方式

以下是测试过程中遇到的问题，以及最终的解决方式：

综合时的 Time Loop

问题：综合时主要有两处地方出现的 Time Loop，即信号的影响形成了环。一处是 EX 与 Div 模块的交互，一处是异常处理与访存的交互。

解决方案：解决方法为调整了时序使之更加清晰。

EX 与 DIV 处的时序为：EX 模块向 Div 模块提供除法开始信号，除法运算的数据，Div 模块向 EX 模块提供除法运算的结果以及运算何时结果。

MEM 段的时序为：先检测访存地址是否对齐和流水线之前阶段的其他异常中断，若已有异常则直接不进行访存指令（不向 MMU 模块传递使能信号），无误则正常执行访存，然后再处理 MMU 模块返回的异常。

指令第一条被修改为 0x7fffffff

问题：sram 在写入.bit 文件的时候起始地址的数据被改写。

解决方案：在 sram 控制器中初始化的部分改写成根据 rst 信号判断而不是根据状态机的状态判断

IF 段与 MEM 段的信号传递问题

问题：当 IF 段访存出现错误时并未记录，IF 段访存时 CPU 的运行模式传递不正确。

解决：增加 IF 段访存时的 CPU 运行模式传递（从 cp0_status 读取），若 IF 段访存出错，则返回空指令，同时将错误信号从 IF 段传入流水线，沿流水线传至 MEM 段处理。

气泡中触发中断

问题：中断信号直接传给 MEM 段，而此时 MEM 段是插入的气泡，导致 EPC 被记录为 0

解决：中断信号也从流水线 IF 段传入，沿流水线传至 MEM 段处理，而不直接把中断信号给 MEM 处理，此时若有气泡，则中断信号的传递也会被暂停，不会出现以上情况。

总线使能问题

问题：操作系统调试时进入用户态后执行用户程序失败，经常触发缺页异常。

原因：tlb 模块产生异常时未及时关闭总线，导致改写了非法的物理地址。

4 总结与感想

选取此项目的最初原因主要是学长的推荐，“不难”，至少他们是这么说的。

回顾整个过程，作为组长的我没能调动起大家全部的积极性和动力，也留下了许多遗憾，如果回到开始造机的那个时候，我能够更加全面来制定计划并执行。其中我最严重的一个失误就是最后的时候，一个 Bug 卡了我们差不多一整个星期，而我迟迟没有想到使用 JTAG 来调试，直到找过助教后才顺利解决，那时候距离展示只有两天了。

在那天晚上调通 Ucore 的时候，“终于可以睡个好觉了”。整个项目的代码量和难度可能不算太大，但硬件编程与调试如果真的是全靠自己钻研琢磨的话，还是会花费大量的时间和精力。纵然最后的成果没有达到预期的结果，从这个项目还是学习到了十分多关于 CPU、关于外设和操作系统、软件工程的知识。也希望下一年联合项目能够让同学学到更多的知识、考察更全面的能力。

最后感谢我的组员的辛勤劳动，也感谢张宇翔助教的帮助和提供的种种建议，让我们小组顺利地跨过了种种困难和障碍。