Tsinghua University

Computer Organization Experimentation

Project Summary Document

Author:
Shizhi Tang
Xihang Liu
Zixi Cai

Supervisor:
Yuxiang Zhang
Prof. Weidong Liu

January 21, 2018

Contents

1 前言

本文档作为 BnG 组 32-bits MIPS CPU 的项目总结。对整个迭代开发的流程和我们在其中得到的经验进行总结。

2 迭代开发计划

2.1 计划内容

我们组的迭代开发计划如下：

- **Sprint 1** 代码格式的约定，操作系统解读，分析初步的需求进而编写初步的需求文档，流水线通路和基本测试环境的搭建。
- **Sprint 2** 对应于《自己动手写 CPU》这本书中前 11 章的内容，完成 datapath 和异常处理的工作。
- **Sprint 3** 串口控制器、Flash 访问模块，能够在硬件测试中通过功能测例。
- **Sprint 4** 能够在硬件上通过监控程序和 u-core。
- **Sprint 5** 完成 USB、以太网、HDMI 的外设工作。
- **Sprint 6** 完成除法等操作，最终可以运行 u-boot。

其中需要说明的是，前四个 Sprint 是项目的基本要求。而第五个和第六个 Sprint 则是因为我们的进度较好，因此将需求从“能够正常运行 ucore”更改为“能够正常运行全部的外设和通过 u-boot 启动 u-core”。可以看到，在这一过程里，需求是动态变动的。与其他组不同的是，我们组的特点是：

- 首先对项目的需求进行了分析，并对代码格式和报告的格式进行了约定。
- 测试环境的搭建先于代码的编写，我们认为，这也是我们组能够在最后进度最快的重要原因。
- 硬件测试为多个人同时进行，一方面是由于代码编译很慢，两个人可以在同一时间里验证更多的假设，另一方面，多个人 debug 也可以显著提高完成硬件测试的速度。

在整个项目的进行过程中，我们最注重的是测试的充分性。对每一个新加入的功能，都需要加入对应的单元测试。此外，我们在设计测试框架时，也特别注意了“可以在之后加入新功能之后，通过测例对功能进行充分的验证”。基于这一思想，我们也对功能测例进行了定制化。定制化之后的功能测例甚至可以直接在 Vivado 的仿真上运行，这也避免了实现所需要花费的巨大时间成本，提高了 debug 的速度。

2.2 计划制定

在一开始拿到项目之后，我们并没有划分对应的 Sprint。在学期初，我们对 CPU 的整个运行也十分不了解。但是我们组每隔一周都会开一次会，开会的主要内容包含两个方面：

- 分配下一周的任务，在每一次开会结束之后，我们会在 gitlab 上发布三个 issue，这三个 issue 分别对应我们组三名同学在下一周所需要完成的工作。
- 讨论整个 CPU 中各种难点的设计，如访存时的状态机等。CPU 相关代码的编写工作，在整个项目所占的只是非常小的一部分，而合理的设计对项目的进行有着非常好的帮助，好的架构可以让我们进展更快。此外，通过讨论，我们也可以将单人设计犯错的可能性降低。

在第一次小组会议上，我们主要完成了代码格式的约定，之后分工，由蔡子熙同学负责基本流水线的搭建。唐适之同学负责测试环境的搭建，而刘熙航则负责需求的分析和初步需求报告的撰写。在这一阶段结束之后，我们首先按照《自己动手写 CPU》这本书的章节顺序进行任务的分配（每周三章左右的进度，一名同学负责一章）。虽然这一任务在很多时候并不能做到完全的并行开发，但是这是由于项目的特点所致。个人认为，我们在并行上做的是很好的。在按照《自己动手写 CPU》的章节目录进行项目的过程中，我们对整个项目的进度有了更为清晰的认识。因此也就制定了前文中的 Sprint 2, 3, 4。按照这样的划分，Sprint 1 虽然任务量较少，但其对于项目的进行有着至关重要的作用。有了好的测试环境，我们就可以更好的进行开发。

由此也可以看到，在项目的一开始，我们是知道最终目标的。但是对于如何达成这一最终目标的认识十分不明确。而随着项目的进行，我们对最终目标的认识一步步的加深。并在这一过程中对项目的 Sprint 进行了划分。由此我们也意识到，很多任务和我们搭建 mips-32 cpu 都有相似之处：在项目的一开始，最终的目标虽然很明确，但是具体的需求是模糊的，而随着项目的进行，我们才能完全的明白项目最终的需求，并按照项目的需求进行变动。在这一任务的完成中，敏捷开发的思想是十分重要的。只有有了敏捷开发的思想。我们才能够做得很好。

3 分工

3.1 Sprint 1

对于 Sprint 1，我们的分工为：

- 三个人一起编写代码的用例规范。
- 蔡子熙负责搭建基本的数据通路。
- 唐适之负责搭建基本的测试框架。
- 刘熙航负责分析需求，撰写初步的需求报告。

3.2 Sprint 2

在这一部分，分工为：

- 蔡子熙负责实现了移动指令和算数指令。
- 唐适之负责实现加载存储，并对代码进行相关的优化。
- 刘熙航负责实现了逻辑、移位、转移、CP0 和异常指令。

3.3 Sprint 3

在这一部分，分工为：

- 蔡子熙负责了串口控制器，搭建了功能侧例的仿真框架，并负责了一部分功能测例的仿真通过。
- 唐适之负责实现了 MMU 和 TLB，并在硬件上跑通了全部的功能测例。
- 刘熙航负责对代码进行调试，并在仿真中通过了全部的功能测例。

3.4 Sprint 4

在这一部分，分工为：

- 蔡子熙和唐适之完成了监控程序的硬件通过。
- 刘熙航和唐适之完成了 u-core 在硬件上的通过（至此，基本要求已经完成，我们的扩展要求分成外设和跑通 u-boot 两部分并行完成，即 Sprint5 和 Sprint6 是平行关系）。

3.5 Sprint 5

这一部分由蔡子熙完成。

3.6 Sprint 6

这一部分的分工为：

- 蔡子熙实现了除法指令，并在硬件上跑通了全部的功能测例。
- 唐适之同学实现了额外的访存指令，并负责了 u-boot 相关的一些 debug。
- 刘熙航同学实现了与 cp0 相关的额外指令。

3.7 文档

这一部分的分工为：唐适之负责设计文档，蔡子熙负责测试文档，刘熙航负责完善之前的需求文档，并补充项目的总结文档。

4 项目感想和总结

我们的项目获得了巨大的成功，在计原最终展示时，我们组是所有的组里唯一一组跑通 u-boot 和全部功能测例的组，并且也实现了全部的可能实现的外设，可以说，我们完美的完成了学期初的需求。以下，我们将项目开发中的一些经验和对项目的感想、建议总结如下：

4.1 项目总结

我们将在项目中收获的经验和整个开发流程的感受总结如下：

虽然在前期，我们组的进度与其他组很接近，但是到了后期，从跑通监控程序开始，我们组的进度就已经领先了所有其他组。不得不说，整个实验的环境也有一些坑，这些坑中的很多也都是要靠我们组慢慢的填平。如功能测例中的两个 bug:访存破坏自身数据区、将 Cache 指令作为 reserved instruction，包括 u-core 会在不会被执行到的函数中使用 break 和 div 这两条指令，这些问题也都是我们组第一个遇到的。可以说，我们在这些坑上花费了很多时间，但这些时间也是这个项目的一部分。就工作量而言，我们认为，自己的 32 位 CPU 并不比 16 位的 CPU 复杂，需要的可能仅仅是将三周的时间平摊到一个学期，大多数时间的工作量也反而会更少。所以，32 位 CPU 的实验，虽然有一些待完善的地方，但也是非常值得去选的。

关于测试：

我们的测试分成三个阶段：在一开始，采用我们自行编写的测试框架进行单元测试，之后通过功能测例进行集成测试，然后通过监控程序进行测试和 debug，之后通过 ucore 和 uboot 进行 debug。实际上，监控程序、ucore、uboot 除了需要支持的指令和机制有少许的差别外，更大的难度来自于代码的规模。因为很多的 bug 会在一些概率极低的情况出现（如我们曾经发现，sw 在 jalr 后三条时会出 bug，这就是一个很小概率的事件，但是却直到 uboot 阶段才触发）。

测试一定要尽量早做，尽量全面地去做，我们的测试分为单元测试和集成测试两个部分，集成测试又分为软件和硬件。在跑 u-core 时，我们耗时最长的一个 bug 花费了大约 70 个小时才被解决，而原因是在单元测试时，由于测试过于复杂而忽视了 ASID 段的相关内容。在完成这个项目的过程里，我们真真切切感受到了，debug 的难度和项目的复杂程度成指数级的关系。所以，单元测试一定要尽量全面的覆盖所有情况。这样可以为后期减少很多的麻烦。

在不同的测试阶段，应当关注不同的问题，在单元测试时，我们只需要去观察对应的单元执行结果是否正确，而对于集成测试，我们更关注的是多条语句的结合。在硬件测试时，问题往往是由访存等耗时较长的操作产生的，而在最终跑 u-core 和 u-boot 的阶段，大多数的 bug 都是和 cp0 相关的机制导致的，这些机制之前很难通过单元测试和功能测例找到。特别是，很多情况下 bug 的发生会有条件，比如我们之前曾经发现，sw 在 jalr 之后三

条的位置会出错。这些也很难通过集成测试来查找。

我们认为，功能测例并不足以算是集成测试，其更偏向于单元测试（这是因为其测试的重点在于指令是否按照期望被执行，而不是能通过指令的组合起到某些效果）。因为对于硬件而言，集成测试的环境其实是很难编写的，其难度可能比 debug 还高，所以我们并没有花时间去编写相应的框架。

关于设计：

虽然我们的编程是基于 VHDL 和 Verilog HDL 的，这也意味着和传统的设计模式没有大的关联，但是设计仍然很重要。这里所指的设计是整个流水线通路的设计。例如，在《自己动手写 CPU》这本书中，其异常所采用的方法是一个向量的每一个 bit 代表一种异常。而我们就将其修改为了一个 5bit 的数据来记录异常。这些与 mips 标准不太符合的修改，是基于整个项目的特征所进行的。**因此，对需求进行更全面、更充分的分析，可以让我们在设计通路时少走很多弯路，进而达到提高效率的目的**

关于分工：

大多数时候，我们的项目是串行工作的，这也是项目的性质所致：流水线通路的搭建，往往是一个功能需要建立在前一个功能已经完整实现的基础之上。针对这一情况，我们采用了以下几种方式来缓解这一问题：

- 我们将每个人每周的空余时间分开，例如 A 在周三到周五完成功能 A，B 在周六开始完成依赖于功能 A 的功能 B，以此类推。
- 在硬件测试时，由多个人同时找 bug，通过交流来提高效率。

4.2 项目感想

32 位 mips CPU 的项目，可以让我们更熟悉计算机的相关知识，熟悉 c++ 编译器可能生成的所有指令的工作原理。虽然我们以后可能不会在 mips 架构上工作，但是熟悉底层仍然可以帮助我们写出更好、更快的代码。而这些，也正是我们作为计算机科班学生的核心竞争力所在。而且，32 位的 CPU，让我们对底层的架构有了更为清晰的认知。

在下一学期里，我们将继续这一项目，与操作系统课联合，希望可以在 2018 年 9 月龙芯的比赛里取得好成绩。

4.3 对学弟学妹们的建议

我们的建议有以下几点：

- 要让团队中的每个成员都熟悉别人做的事情，熟悉所需要用到的全部工具链。
- 要尽早开始对项目进行测试。
- 开发流程中，要约定代码规范，实现功能时要考虑到后续被扩展的可能。
- 在项目中的一些难点上，要多与队友进行讨论。

BnG, better and greater.