20200919 组会总结

林成渊 PB18051113

RPU-llxx

版本:1.0

更新:2020 年 9 月 19 日

1 一些问题及其解答

基本就是在看文档。一个文档是 RISC-V 官网上给出的非特权指令集部分的文档,重点关照 RV32I 的部分。另一个文档就是 cv32e40p 这个核,参考了三个部分的内容,一者是 github 页面上有的关于该核的说明书,二者是源码本身,三者是往期师兄做过的一部分工作所写成的文档。不过尚未对这个核做到能够在 vivado 上编译综合出来。关于文档阅读本身提了些问题。

1.1 为什么 RISC-V 的指令集里面不需要延迟槽?

首先是组内互相解释对于延迟槽的理解和看法。延迟槽本身是一种针对控制冒险设立的机制,把一些不管控制出现与否都会执行的语句放在延迟槽里,另外一种把分支指令尽量放在流水线靠前的流水段的做法正是一种削短延迟槽的努力。延迟槽本身在编译中可能被直接插入一个 nop。对于 RISC-V 舍弃延迟槽,可能的解释有三

- **预测准度**:当前的 CPU 设计中分支预测已经做得非常准确,其预测成功的概率往往高于九成,如此一来原本为预测失败情况而专门准备的分支预测槽就没有很大的存在必要;

- **成本**：设立分支预测槽本身会在软件编译和硬件实现层面增加不小的复杂度和开销，舍弃延迟槽本身是一种精简设计、节约成本的做法；
- **收益比重**：RISC-V 等现代处理器系统的流水段通常都是装满的，而且预测失败本身的代价在整个运行中的占比已经很小，所以再去准备延迟槽是一种高开销低收益的做法。

1.2 opcode 为什么放在低位（以此作为对 immdata 位数分配问题的前置）？

主要有两点可能的解释。

- **拓展与兼容**：opcode 放在低位，有利于这个本身是 32 位的指令能很好地兼容 16 位或者扩展到 64 位的指令，由于任何指令小端都是有着公共的位次，不必大张旗鼓地移动关键字位置；
- **读指令**：在只能读单个字节的系统里，opcode 放在低位，那么我先读到的就会是 opcode 所在的那个字节，然后根据这上面的 opcode 信息，我们就可以分析到底需要往后读多少字节。

1.3 SBJU 型指令的 imm 字段为什么要做如此复杂的划分？

这个与前一个问题同理，重复部分不再赘述，仔细观察还能有如下可能的解释：

- **移位**：这个划分实际上是根据移位来的，因为我们参考的指令集里面有 compressed 指令（半字）等，因此不同于之前课程实验里指令左移两位的做法，这里仅移动 1 位;
- **重要的位**：仔细观察会发现一些重要的位依然保留在相应的位置。比如指令中最高位始终保存着 imm 所能给出的位里面的最高位，这有利于符号扩展;
- **简化硬件实现**：这个设计本身有一定 tradeoff 在里面，参考 BSUJ 的区分，在保证得到需要的位数的前提下，我们尽量让指令字中固定位的信息表示 imm 值中固定位的值。这一做法尽管会为软件和硬件描述代码的编写带来麻烦，但实际上对于底层硬件反倒是一种简化，我将

可以直接对指令的固定位设定固定的操作而省下一个 mux 的部分。

1.4 为什么 J 指令的 imm 不左移了，那多保存的一位里面到底存了什么信息？

J 一类指令常见于函数调用等。一个可能的解释是这个多保存的一位用于标记这里是否为有效地址。

2 关于后续工作如何处理这个核

谈了环境的搭建、对于这个核做哪些处理、后续工作的安排。

2.1 环境

[运行环境] 直接用 Vivado 就可以，结合课程配的 Nexys4DDR 开发板。尽量争取能把这个核先编译综合起来。

[测试环境] 关于如何测试这个核，一个思路是参考龙芯给的 Soc 环境做一个 trace 比对，但是移植难度有点大，一方面我们需要一个好用的交叉编译环境，但现有的工具链可能会编译出不需要的指令，另一方面代码语言都不一样，接口和 ISA 等都差异巨大，龙芯的框架没法简单地搬过来用。故此思路不太可行，但仍保留。

目前有了一个基于汇编的仿真器，可以根据这个自己编写测试用的汇编代码。

2.2 怎么处理核

大致来讲，就是先对核做一些裁剪工作，仅保留 RV32I 有的指令内容。而且 RV32I 里面最后三条指令不需要实现（FENCE、ECALL、EBREAK 都不用），还有中断相关也不需要实现。

因此大量的模块都可以直接砍掉（包括浮点运算部分）。这里我们讨

论了一下这个工作怎么进行，有两条路可走，一者是根据原核进行裁剪，二者是直接重写一套新的。

裁剪的原因可能是为了能够保留大体的结构和接口，方便后面直接调用人家成熟的框架和工具。而重写则是基于这个指令集本身足够简单，与其读那些有不少冗余功能的代码慢慢裁剪（可能裁得支离破碎又不成体系）不如重写。具体抉择在后面又有讨论。

2.3 后续工作

尽量一周内把前面两点提到的工作干掉。关于处理核的两条路，原来是希望分两组，最好每个二人组跟别做裁剪和重写。但考虑到效率和时间问题，组内 4 个人基本都是觉得裁剪更好，而且最好每个两人组做一个裁剪。