RISCV-CPU

一个基本实现了RISCV 32位整数指令集的CPU。

概要

该项目是 System 2018 的大作业，使用 Verilog 语言，实现一个简单的 CPU，并在 FPGA 上进行烧写测试。该报告介绍了一些该项目的设计细节和经验总结。

一些细节

五周期的访存操作

因为这次作业的内存设计在板子上，一次读写只能走8 bit（1字节），所以取指令和大量访存操作都要4次内存读写（至少要5周期），所以我设计了一个5周期访存的内存控制器，用状态机实现，控制IF和MEM阶段的内存操作。以下是其中的一些要点：

- IF和MEM同时有访存需求时，优先处理MEM；
- 第五周期取到数据的最后一部分时，立刻用组合逻辑将数据送到流水线上，以保证下一周期的IF和ID顺利接班。

I-Cache

为了让五级流水真正地留起来，加入了I-Cache。这是一个非常简单的直接映射Cache，用pc的第2-7位作为索引，因为内存中指令部分不会被更改，所以该Cache不用考虑写操作的问题。

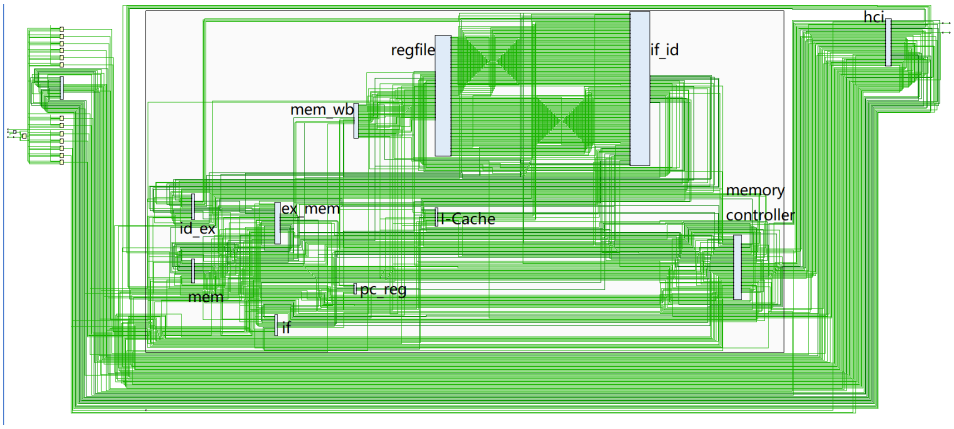
当然，加入了I-Cache，内存控制器就必须要谨慎处理各类读取操作和是否miss的关系了。在此期间，很多原本没发现的流水错误也都暴露出来了，例如无法用forwarding解决的数据Hazard，让我调了好久。

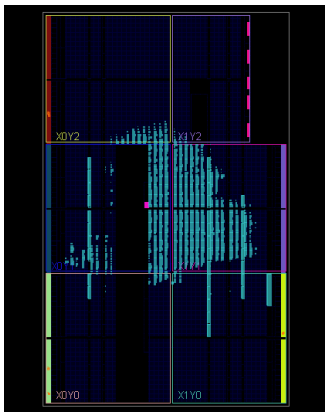
分支预测？

原本是计划写一个分支预测的，动态分支预测太难写而且板上接线容易乱掉，就设计了一个静态分支预测。写的过程中发现了一个问题：若没有BTB，那么是否实现静态分支预测就只有一个周期的差距，而且只是针对于六种branch最后不跳转的情况。预测失败时还要在一个周期内打断已经取指了一部分的（假）下一条指令，并立刻开始准备新指令的读取，否则预测成功的优势就会因为预测失败的延迟而失去了。

经过一番思考与尝试，我还是放弃了实现分支预测，因为我觉得若强行照着上面的思路实现并不影响原有的设计的话，线路会又乱又杂很可能只有降频才能满足时序了。最后，我的策略是在IF阶段就判断出当前指令是不是分支指令，如果是，在当前指令的ID阶段结束前不要再IF了，一般来讲，这样只会耽误一个周期，在其他地方流水很充足的情况下并无太大的性能影响。

接线与板上元件的使用状况

- 

- 

一些测试结果

- pi.c 1.5-1.6s 升频到**125MHz**后为**1.1-1.2s**
- qsort.c 5.2-6.8s
- queens.c 2.1-2.2s
- bulgarian.c 0.6-0.7s
- hanoi.c 2.5-2.6s

```
50 9751 9752 9753 9754 9755 9756 9757 9758 9759 9760 9761 9762 9763
74 9775 9776 9777 9778 9779 9780 9781 9782 9783 9784 9785 9786 9787
98 9799 9800 9801 9802 9803 9804 9805 9806 9807 9808 9809 9810 9811
22 9823 9824 9825 9826 9827 9828 9829 9830 9831 9832 9833 9834 9835
46 9847 9848 9849 9850 9851 9852 9853 9854 9855 9856 9857 9858 9859
70 9871 9872 9873 9874 9875 9876 9877 9878 9879 9880 9881 9882 9883
94 9895 9896 9897 9898 9899 9900 9901 9902 9903 9904 9905 9906 9907
18 9919 9920 9921 9922 9923 9924 9925 9926 9927 9928 9929 9930 9931
42 9943 9944 9945 9946 9947 9948 9949 9950 9951 9952 9953 9954 9955
66 9967 9968 9969 9970 9971 9972 9973 9974 9975 9976 9977 9978 9979
90 9991 9992 9993 9994 9995 9996 9997 9998 9999 10000

CPU returned with running time: 5.296875
```

```
Enter r to run, q to quit, p to get cpu PC(demo)
r
CPU start
31415926535897932384626433832795288419716939937510582097494
93844695558223172535940812848111745028412701938521155596444
19914564856692346348610454326648213393607260249141273724587
88204665213841469519415116094330572736575959195309218611738
01194912983367336244065664386021394946395224737190702179860
56082778577134275778960917363717872146844090122495343014654
77477130996051870721134999999837297804995105973173281609638

CPU returned with running time: 1.156250
wangnick@LAPTOP-Q44N2JRM:/mnt/d/documents/projects/riscv$
```

写在最后

就像我在presentation中说的，这个CPU经历了反复的修补，也让我发现了自己遗漏的各种知识点和没有推敲过的细节。对着波形图查错的过程真令人头皮发麻，但是纠出了错误的那一刻真的好开心！

非常感谢各位助教和同学们在此期间提供的帮助！！

2018年12月30日