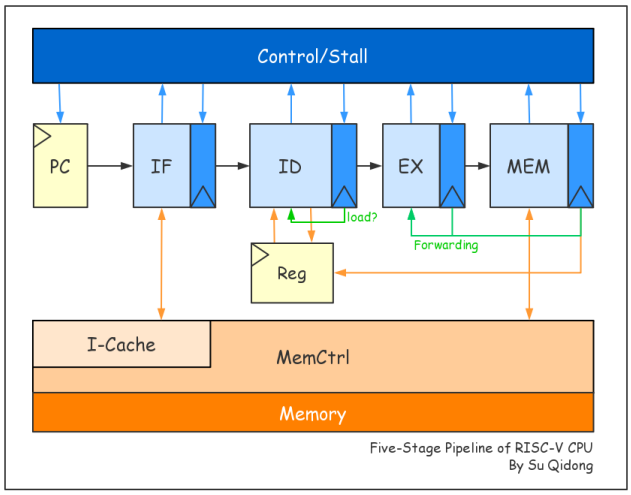
CPU 大作业报告 **苏起冬**
Computer System(I) 2018 Fall 517030910415
2018 年 12 月 5 日

1 整体架构

本项目是一个使用 Verilog HDL 语言实现的能在 Basys3 上运行 RISC-V 指令集的一个 CPU。最初的设计目标就是不仅要能通过仿真，而且必须可以在真实硬件上面运行，所以整体架构比较简单，主要目标是提高效率和可靠性。CPU 可以在 100MHz 的频率下工作。

整体结构是传统的五级流水架构，如右图所示。

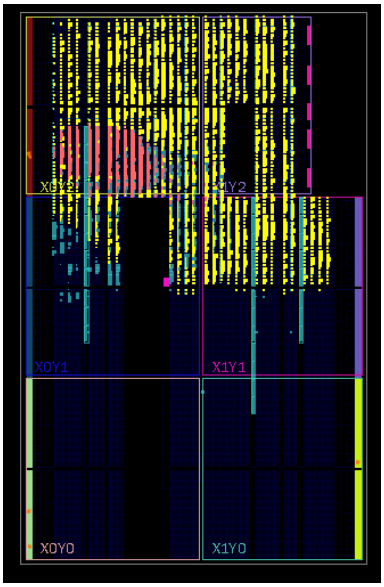


MemCtrl 是一个状态机，用于处理来自 IF 和 MEM 两阶段对于内存的访问需求，并作出仲裁。目前的做法是 MEM 的请求优先。写入一个字需要 5 周期，访问板载内存上的一个字需要 5 周期，通过 IO 访问的话需要 8 周期（不计 uart 通信时间）。

I-Cache 一个集成在 MemCtrl 中的指令缓存，大小为 256×4 字节。有了 I-Cache 之后，在命中的情况下可以实现一个时钟周期取指令，大大减少了读取指令的时间，使流水线可以真正工作起来。256 条指令基本上能容纳一个函数。

MemCtrl（包括 I-Cache）占据下图黄色部分。

IF 负责读取指令，是一个状态机。在读取完一条指令之后会进行预解码，如果是一条分支语句，则阻塞住流水线的之前阶段，直到这条分支语句计算出下一条语句的地址。（并没有做分支预测）



ID 负责解码。这个部分占用的面积（右图红色区域）比较大，看起来还有很多改进的空间。解码之后如果发现下一条指令是 load 指令并且与当前指令有依赖关系，则在当前指令后面插入一条空指令。

EX 负责执行。

MEM 负责进行内存操作，是一个状态机。如果目前处理的指令有内存操作，则阻塞流水线的之前阶段，直到 MemCtrl 返回 done 信号。

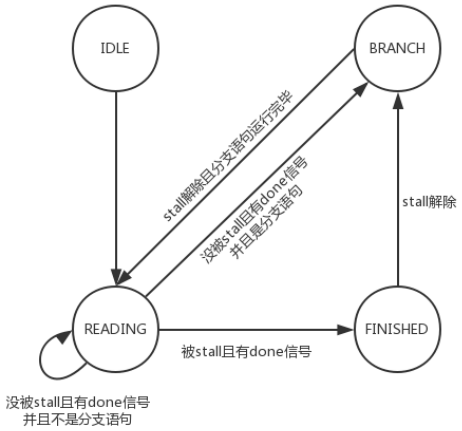
控制模块 控制模块主要做两件事情：控制 stall 和修改 pc 值。这是一个简单的组合逻辑，接收各个部件对于流水线 stall 的请求，计算后广播给各个部件。

***D-Cache** 我在 memctrl 里面做了一个简易的数据缓存，但是很难符合时序要求，很难在一个周期的时间里取到数。而且经过试验发现对于性能的改进并不是那么明显，占用面积又非常大，所以没有启用。但是可以通过宏定义在 memctrl 中启用 dcache 的功能。（功能上还不完善）

2 遇到的问题

2.1 时序设计

误区 我一开始以为在一个周期内做的事情越多越好，但是后来发现完全不是这么一回事。组合逻辑一复杂，逻辑门的延时就会变大，就容易无法满足时序要求，无法升高频率。设计的时候应该注意流水线的各个阶段之间所用的时间应该尽量均匀。



不同模块之间的配合 IF、MEM、MemCtrl、PC 等模块之间的时序关系比较复杂，如果要充分利用起每一个周期，需要仔细考虑它们之间的关系。我选择使用状态机来实现各个模块。左图是 IF 的状态转移图。

IF 在取完一条指令之后（此时 memctrl 的 busy 为 0,done 为 1）会立即请求下一条语句，也就是说这个 busy 为 0 的状态只会维持一个周期。而 MEM 阶段在完成一次读写操作之后（此时 memctrl 的 busy 为 0,done 为 1）这个周期是被空出来的，否则流水线仍然会被阻塞住（因为只要 MEM 有请求，流水线就会被阻塞住）。在这个空出来的周期里，可以完成 pc 的修改，IF 也可以抢先发起请求。

IF 发起请求之后，MemCtrl 会读完这条指令（无论这个过程中 MEM 阶段是否遇到读取指令）。在这个过程中，MEM 阶段遇到了读取指令，那么流水线一直处于 stall 状态。如果此时 IF 请求的指令读完了，并且当前流水线被 stall 住了，那 IF 就将读到的指令暂存起来，直到 stall 解除再恢复流水线的运行。

期待改进 现在这个 CPU 的设计应该说是非常简单，但是时钟频率最高也只能到达 110MHz 左右，无法继续升高频率，这和预想中的效率还有一些差距。

2.2 综合、实现

底层 = 黑箱 对于综合机制了解的不多。Vivado 中有许多 Systhesis、Implementation 的策略可供选择，但是我对它们并没有什么了解，也不知道修改那些参数会有什么后果，基本就是胡乱选择。现在产生的 bitstream 文件基本是在 Flow_PerOptimized_high 和 Performance_ExtraTimingOpt 策略下完成的。

2.3 调试

模拟和实际的区别 在模拟阶段发现代码有错误还是很容易改正的。最可怕的是可以过模拟，但是烧到 FPGA 上之后运行不正常。因为硬件上的调试手段并不多。在这种情况下，我选择使用 vivado 自带的调试核，在设定了一定的触发条件之后，它可以记录一段时间内信号的变化。对于调试来说非常实用。

参考文献

[1] 雷思磊. *自己动手写 CPU*, 电子工业出版社, 2014.

[2] 胡振波 *教你设计 CPU——RISC-V* 处理器人民邮电出版社，2018.