实验报告格式说明：

1. 标题层次建议不超过四级，从第一级开始标号格式依次采用：一、二、三；（一）、（二）、（三）；1、2、3；（1）、（2）、（3）。
   1. 一级标题用小三号字，中文为黑体，英文为Times New Roman，单倍行距，段前段后各0.5行。
   2. 二级标题用四号字，中文为黑体，英文为Times New Roman，单倍行距，段前段后各0.25行。
   3. 三级标题用小四号字，中文为黑体，英文为Times New Roman，单倍行距。
   4. 四级标题用五号字，中文为黑体，英文为Times New Roman，单倍行距。
2. 正文用五号字，中文为宋体，英文为Times New Roman，1.5倍行距。
3. 所有图统一顺序标号，图标题紧挨在图的下方，居中，用小五号字，中文为宋体，英文为Times New Roman，单倍行距，段后0.5行。
4. 所有表统一顺序标号，图标题放在表的上方，居中，用小五号字，中文为宋体，英文为Times New Roman，单倍行距，段前0.5行。
5. **报告形成后删除本模板中所有红色文字！**

**实验七报告**

学号 2017K8009929032 2017K8009929034

姓名 杨程远 杨宇恒

箱子号 15

一、实验任务

在实验六的 CPU 代码基础上添加更多的指令：转移类指令BGEZ、 BGTZ、 BLEZ、 BLTZ、 J、BLTZAL、 BGEZAL、 JALR以及访存指令 LB、 LBU、LH、 LHU、 LWL、 LWR、 SB、 SH、 SWL、 SWR。检验方法：运行func\_lab7，要求成功通过仿真和上板验证。

二、实验设计

**针对Lab3实验**，请参考以下格式描述我们提供的参考CPU的设计。（也就是完成子任务一）

（一）总体设计思路

阐明总体设计思路，即从系统顶层角度出发，概要性地描述整个系统的工作机制，所需要进行哪些实验设计、完成哪些功能。

在进行本章节描述时，推荐以结构设计图的形式阐述硬件部分，以流程图的形式阐述软件部分。

**需要给出参考设计的结构设计图！**

1. 顶层模块的通路调整

数据通路无需调整。

控制通路中分支指令增加无需调整，因为都在ID级完成，而访存指令需要在流水线中添加三组控制信号。第一，ID级译出的指令信号族inst\_load（由inst\_lw、inst\_lb、inst\_lbu等7个信号组成，store类似）和inst\_store，它们向之后流水级的传播，直到这些信号结合计算得到的地址译码成控制信号ld\_rshift\_op和st\_rshift\_op（四种循环移位，具体含义见重要模块4和5），并且进一步向后传播直到被使用；第二，ID级译出的ld\_extd\_op控制信号，向后传播直到被使用；第三，对mem\_wen和gr\_wen的位宽调整，其中，gr\_wen在访存级由1位变为4位。

前递通路中在原有1位gr\_wen的基础上再增加4位gr\_wen。

1. ID模块内部调整

添加新的指令所需的译码逻辑；将分支判断逻辑整合成br\_comp模块（见重要模块1设计）。

1. EXE模块内部调整

对inst\_store调用st\_decode模块，结合计算得到的访存地址，译码出st\_rshift\_op和data\_sram\_wen。对st\_rshift\_op调用st\_select模块，多选出data\_sram\_data。

1. MEM模块内部调整

对inst\_load调用ld\_decode模块，结合从EXE级传播过来的访存地址，译码出ld\_rshift\_op。对ld\_rshift\_op和ld\_extd\_op调用ld\_select模块，两级多选出mem\_result。

（二）重要模块1设计：br\_comp模块

1. 工作原理

根据branch指令的比较方式输出分支是否应该跳转的结果。这让译码级的代码更简洁。

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| br\_op | IN | 6 | 右六种比较的情况 |
| br\_src1 | IN | 32 | 比较操作数1 |
| br\_src2 | IN | 32 | 比较操作数2 |
| br\_happen | OUT | 1 | 比较结果是否成立 |

1. 功能描述

该模块模仿ALU，用6位独热码br\_op分别表示6种不同的分支条件（BEQ、BNE、BGEZ、BGTZ、BLEZ、BLTZ），并结合输入的两个源操作数判断分支是否发生。需要注意的是源操作数必须取前递通路的多选器之后的数值。

（三）重要模块2设计：ld\_decode模块

1. 工作原理

在访存级对ld指令译码，从指令类型和地址后两位译出循环移位多选器控制信号和拓展方式多选器控制信号（具体含义见重要模块4）。

1. 接口定义

每部分的接口是什么。如果写报告的时间充裕，可以以表格形式列出；如果时间仓促，该节可以一笔带过。

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| inst\_load | IN | 7 | load指令的类型（7种），采用独热码 |
| addr | IN | 2 | 访存地址的最低2位 |
| gr\_we | IN | 1 | 访存级流水线中的1位寄存器写使能信号，用于产生非load指令的寄存器写使能 |
| ld\_rshift\_op | OUT | 4 | RAM数据需要循环右移的字节数，独热码，作多选器的信号 |
| gr\_we | OUT | 4 | 寄存器堆的字节写使能信号 |

1. 功能描述

根据load指令类型计算出数据RAM返回的数据需要循环右移几个字节，并且得到对应的寄存器堆字节写使能信号。若非load指令（inst\_load全0），则将流水线中的1位写使能信号gr\_we\_1复制成4位。

1. 设计权衡

我们选择将ld\_decode模块放在MEM级而非WB级，这样更好。这是因为译码和访存可以并行执行，而访在写回级需要先译码后多选，延长了数据通路，虽然写回级显然不是关键路径。

（四）重要模块3设计：st\_decode模块

1. 工作原理

在执行级对st指令译码，从指令类型核地址后两位译出循环移位多选器控制信号（具体含义见重要模块5）。由于执行级才算出地址，并且执行级就要进行多选，只能在这里译码。

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| inst\_store | IN | 5 | store指令的类型（5种），采用独热码 |
| addr | IN | 2 | 访存地址的最低2位 |
| st\_rshift\_op | OUT | 4 | RT源操作数需要循环右移的字节数，独热码，作多选器的信号 |
| mem\_we | OUT | 4 | 数据RAM的字节写使能信号 |

1. 功能描述

根据store指令的类型计算出数据RAM的字节写使能信号，并算出应当把RT源操作数循环右移多少个字节。

1. 设计权衡

我们选择将st\_decode模块放在EX级而非ID级，但在译码级进行地址后两位的运算并译码也是一个可选方案。这以增加译码级长度为代价减少执行级长度，哪个方案更好需要根据时序分析结果。

（五）重要模块4设计：ld\_select模块

1. 工作原理

在访存级根据多选控制信号，先后从4种循环移位和5种拓展方式（包括不拓展）中进行多选一。这可以让访存级的代码更简洁。

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| ld\_rshift\_op | IN | 4 | 4位对应0、8、16、24位四种向右循环移位的选择 |
| ld\_extd\_op | IN | 5 | 5位对应LB、LBU、LH、LHU、不拓展（LWL、LWR）五种拓展方式的选择 |
| data\_sram\_rdata | IN | 32 | 数据ram返回的数据 |
| mem\_result | OUT | 32 | 选择要写回的结果 |

1. 功能描述

该模块由两个多选器组成：先根据ld\_rshift\_op信号选择出循环右移后的数据mem\_result\_unextd，再根据ld\_extd\_op选择出将mem\_result\_unextd通过适当方式扩展后的结果。

1. 设计权衡

我们选择将ld\_select访在MEM级而非WB级，但放在WB级也是一个可选方案。后者可能可以减少关键路径长度，但会导致MEM级无法前递ld数据。

（六）重要模块5设计：st\_select模块

1. 工作原理

在执行级根据多选控制信号，从4种循环移位中多选一。这可以让执行级代码略微简洁，并和ld\_select模块相一致。

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| st\_rshift\_op | IN | 4 | 4位对应0、8、16、24位四种向右循环移位的选择 |
| data\_from\_reg | IN | 32 | 寄存器读出结果 |
| data\_sram\_wdata | OUT | 32 | 选择要存入数据ram的结果 |

1. 功能描述

该模块为一个四选一多选器：根据st\_rshift\_op选择出循环右移后的数据并输出。

（七）重要模块6设计：forward\_merge模块

1. 工作原理

在译码级，把前递数据和寄存器独读出的数据，按照前递源流水级的4位寄存器写使能进行合并。这个模块可以调用两次，并且让译码级代码简洁。

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| forward | IN | 3 | 3位对应执行、访存、写回级是否前递，输入保障3位中最多只有一个为1 |
| forward\_en | IN | 12 | 执行、访存、写回级的4位寄存器写使能 |
| forward\_data | IN | 96 | 执行、访存、写回级的前递数据 |
| rf\_rdata | IN | 32 | 寄存器堆读出的数据 |
| merge\_value | OUT | 32 | 合并后的数据，之后用于分支是否跳转判断 |

1. 功能描述

由于输入的forward信号已经保证了优先级，模块内的多选器采用与或式，其中default项的选择信号简单地通过其他选择信号都为0的逻辑来产生。

1. 设计权衡

我们不嫌麻烦地用与或式实现多选器，因为这条路径很可能是关键路径。这是因为多选器的输入分别是执行、访存、写回级末端的输出。而由于使用与或式，我们需要增加一个default项的选择信号，首先这信号的产生延时一定比优先级多选器短，其次这个信号的产生很可能可以和前递数据的产生并行。

三、实验过程（50%）

（一）实验流水账

记录哪一天，几点到几点，做了什么事，结果如何。事情不要展开来写。

10月15日23:00-10月16日2:00：完成重要模块的工作原理设计和接口定义。

10月21日17:00-10月22日1:30：完成全部实验设计和代码。

以下错误记录 也就是记录 子任务二 的完成过程。

（二）错误记录

重点记录调试过程和机理分析。请以**图文结合**的方式进行描述，如有波形图应当**分组（Group）分明、分割（Divider）清晰、有标志线（Marker）指示关键时刻**。

1、错误1：前递数据选择错误导致写回的计算结果错误

（1）错误现象

写回值错误，如图1。

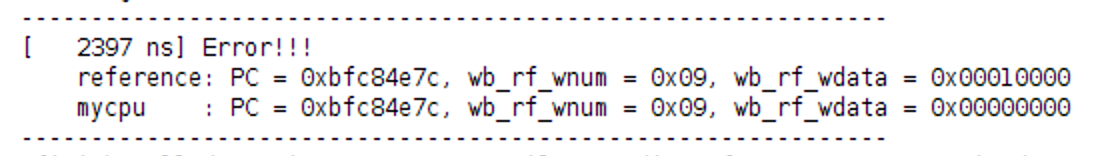


图1 错误1的写回值错误

（2）分析定位过程

先从写回级倒推回译码级，发现没有把寄存器读出来的值作为之后的alu输入，如图2，rf\_rdata2没有作为rt\_value。

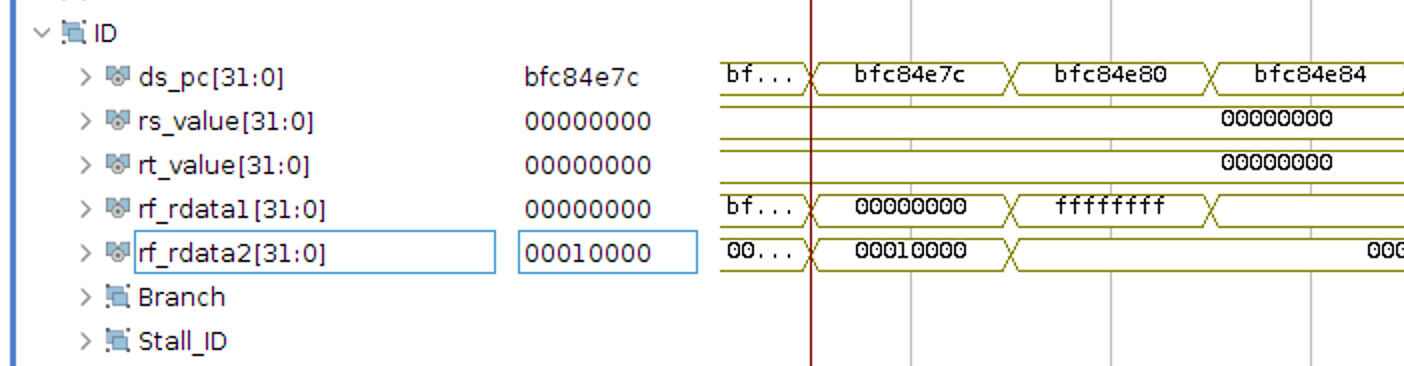


图2 rf\_rdata2和rt\_value不同。

于是，查看rf\_rdata2生成rt\_value的forward\_merge模块，如图3，发现多选器default控制信号no\_forward错误，由于只有最低位正确，考虑可能是位数问题。

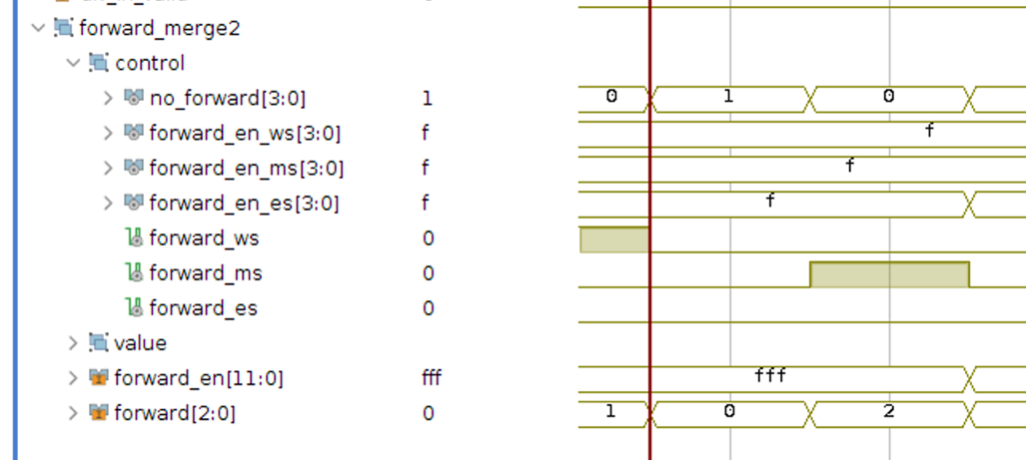


图3 forward\_merge模块控制信号错误

之后检查no\_forward信号的assign代码，发现图4，其中，取非操作应使用~但使用了错误的!。

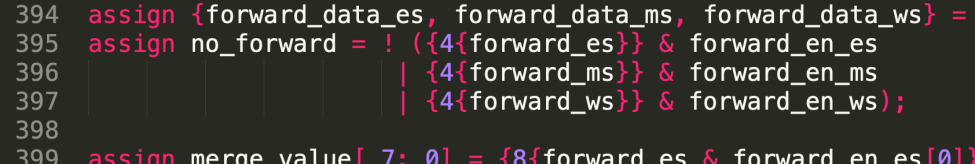


图4 no\_forward取非操作错误

（3）错误原因

no\_forward取非操作错误。

（4）修正效果

用~取非，之后这部分仿真通过。

2、错误2：数据RAM使能信号计算时出错

（1）错误现象

寄存器堆的写回值错误，如图5。

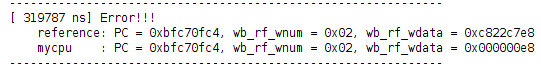


图5 错误2的写回值错误

（2）分析定位过程

根据发生错误处的PC寻找对应的汇编代码片段，发现产生错误的是一个LW指令，如图6。

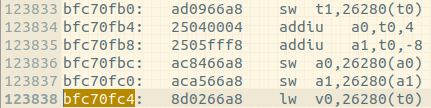


图6 发生错误的汇编码片段

在波形图中寻找对应的位置，如图7所示。

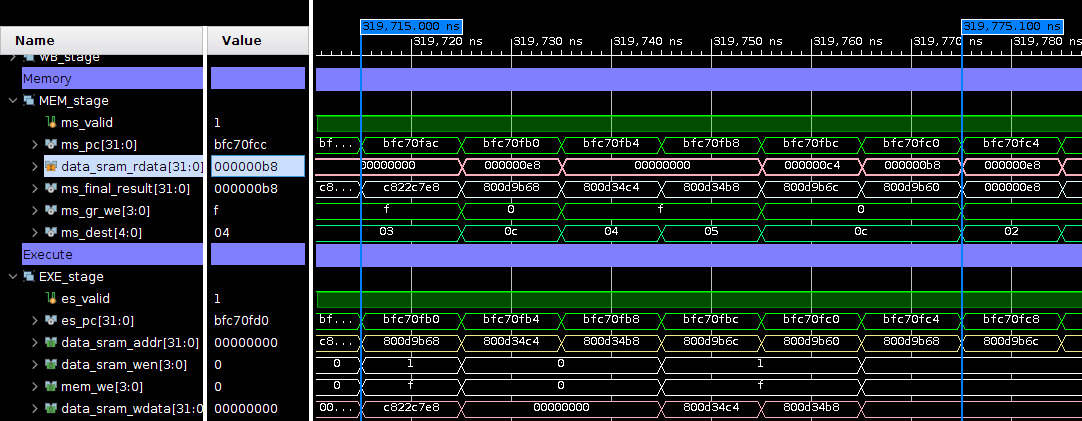


图7 发生错误的波形图示例

发生写回错误的LW指令在第二个Marker时到达访存级。从波形中可以发现，数据RAM返回的值就是错误的0x000000e8，说明LW指令本身没有错误，而是在此之前写数据RAM的指令出错了。通过搜索地址值，可以在波形图中找到对应的指令，它在第一个Marker处进入了执行级；在图6中可以发现它就是地址位于0xbfc70fb0的这条SW指令。这里的写数据data\_sram\_wdata也是正确值0xc822c7e8，所以只能是写使能出错了：它应该是四位全高的0xf，却只拉高了最低位。再将st\_decode模块输出的mem\_we抓取出来，发现它是正确的0xf，所以一定是data\_sram\_wen的计算式出错了。

（3）错误原因

bug2_reason

图8 1位的es\_valid使得写使能信号高3位恒为0

如图8，计算数据RAM写使能data\_sram\_wen时需要和执行流水级的有效信号es\_valid相与，可是es\_valid只有1位，像这样直接用按位与会导致计算结果的高3位恒为0，所以写数据RAM时永远只能写入最低字节。

（4）修正效果

bug2_correction

图9 将es\_valid复制成4位即可修正错误

如图9，将es\_valid信号复制成4位再进行按位与就能解决此问题。

（5）归纳总结

在不同的信号进行按位逻辑操作时一定要注意位宽的问题，用1位信号与多位信号进行与或操作时需要对其进行复制操作。

1、错误1：错误简介命名

（1）错误现象

描述这个错误产生时的现象。

（2）分析定位过程

说清楚你碰到这个问题是如何分析定位出错原因的。可能你分析定位过程中经历了多轮尝试，把它们都记录下来。

（3）错误原因

给出一个出错原因的正式说明。

（4）修正效果

说明你修正这个错误的方法，并说明它是否有效。

（5）归纳总结（可选）

说说你觉得这个错误是哪种类型的，今后如何提前规避。

2、错误2：错误简介命名

……

四、实验总结（可选）

细致的模块抽象和明确的功能描述可以方便小组合作。