**NSCSCC2023龙芯杯个人赛设计报告**

学校：河南理工大学

姓名：包子旭

一、设计简介

BZX\_MIPS是专为“龙芯杯”计算机系统能力培养大赛而设计的一款MIPS32、无中断、无异常的处理器核。它采用了经典的五级流水线结构，使用单发射机制，共实现了57条指令，包括15条算术运算指令、8条逻辑运算指令、6条移位指令、12条分支跳转指令和12条访存指令。由于采用的是经典的五级流水线结构，只用考虑写后读（Read After Write，简称RAW）的数据相关情况，为解决RAW数据相关问题，该处理器核实现了数据前递和流水线暂停技术，即通过在流水线的不同阶段将计算结果提前传递给后续阶段，避免数据相关冒险，从而提高指令的执行效率。当遇到数据相关或其他冲突时，暂停流水线的进程，等待冲突解决后再继续执行，避免错误结果的产生。另外，利用延迟槽技术解决了分支跳转问题。另外，利用延迟槽技术解决了分支跳转问题。这些优化措施使得处理器的性能得到了显著提升，提高了指令执行的效率和流畅性。目前，本项目已经可以通过全部的三级测试和性能测试，时钟频率最高为55MHz。

二、设计方案

（一）总体设计思路

本项目主要由BZX\_MIPS处理器和thinpad\_top串口控制两个顶层模块组成。通过在thinpad\_top中调用BZX\_MIPS的接口，实现流水线与BaseRAM和ExtRAM的交互，并且在thinpad\_top文件中设计了内存映射关系根据地址的不同，分别映射到BaseRAM和ExtRAM中，经过串口的发送与接收，将访存结果传回BZX\_MIPS处理器中。BZX\_MIPS处理器采用经典的五级流水线结构，包括IF\_stage（取指）、ID\_stage（译码）、EXE\_stage（执行）、MEM\_stage（访存）、WB\_stage（写回），通过在mycpu\_top调用各级流水线模块。为了进行性能测试，并实现MUL乘法指令，需要添加乘法器。乘法器采用both两位乘和华莱士树结构，放置在执行阶段进行调用。值得注意的是，乘法器需要执行两个时钟周期才能得到正确的结果。所以需要把乘法结果通过EXE\_stage的顶层模块传递到MEM\_stage，然后在MEM\_stage取得乘法的正确结果的低32位，并随着WB\_stage写入rd寄存器。这样的设计能够满足处理器核的乘法需求，并在性能测试中提供准确的运算结果。

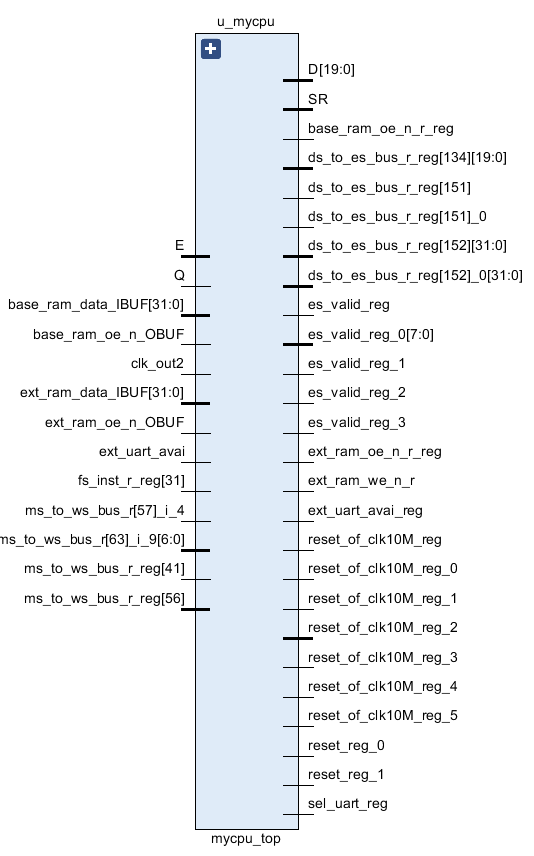


图1 BZX\_MIPS顶层模块接口图

（二）IF\_stage模块设计

IF阶段模块的主要功能是从指令存储器中取出指令，并根据控制信号和分支预测结果更新程序计数器。并将正确的PC以及对应的指令码传递到下一级流水线。在实现过程中考虑了流水线暂停、分支预测、指令存储器的读取等因素，以确保指令的正确取出和流水线的顺利运行。

不过在我加上串口后，发现会出现“吞指令”的情况，即出现load指令的时候，会导致IF级阻塞两个节拍，这样会造成同一条指令在取指阶段inst\_sram\_rdata会从存储器中读取三条指令的数据，会出现三个指令码。导致正确的指令码被覆盖掉，从而出现“吞指令”的情况。为了解决这个问题在IF阶段增加了一个寄存器fs\_inst\_r用来存储被覆盖掉的正确的PC对应的指令码。

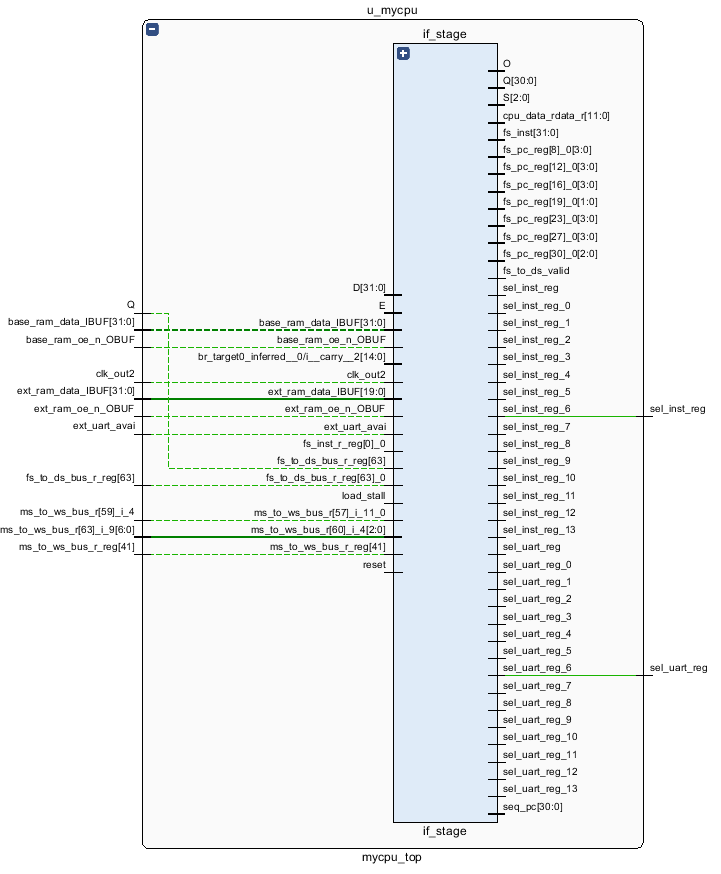


图2 取指模块顶层模块接口图

（三）ID\_stage模块设计

译码阶段是处理器中的重要组成部分，负责将取指阶段获取的指令进行解码，并生成相应的控制信号，以保证指令正确执行。我所设计的译码阶段顶层模块（id\_stage）包括以下主要功能：

* 数据传递：模块接收来自译码与取址阶段的输出数据（fs\_to\_ds\_valid和fs\_to\_ds\_bus），并将数据传递给执行阶段（ds\_to\_es\_valid和ds\_to\_es\_bus），确保数据的正确传递和处理。
* 数据前递：模块接收来自执行、译码、写回阶段的前递过来的相关信息，包括阻塞信号、前递使能、前递的数据地址，以及前递过来的数据。并根据需要产生阻塞信号（forward\_stall），用于防止数据相关冲突。
* 分支信号：模块接收来自执行阶段的分支完成信号（IF\_over），用于处理分支指令。
* 加载阻塞：模块接收来自执行阶段的加载阻塞信号（load\_stall），并根据需要产生阻塞信号，防止数据相关问题。
* 分支跳转：模块产生来自译码与取址阶段的分支跳转信号（br\_bus），用于执行阶段进行分支跳转操作。

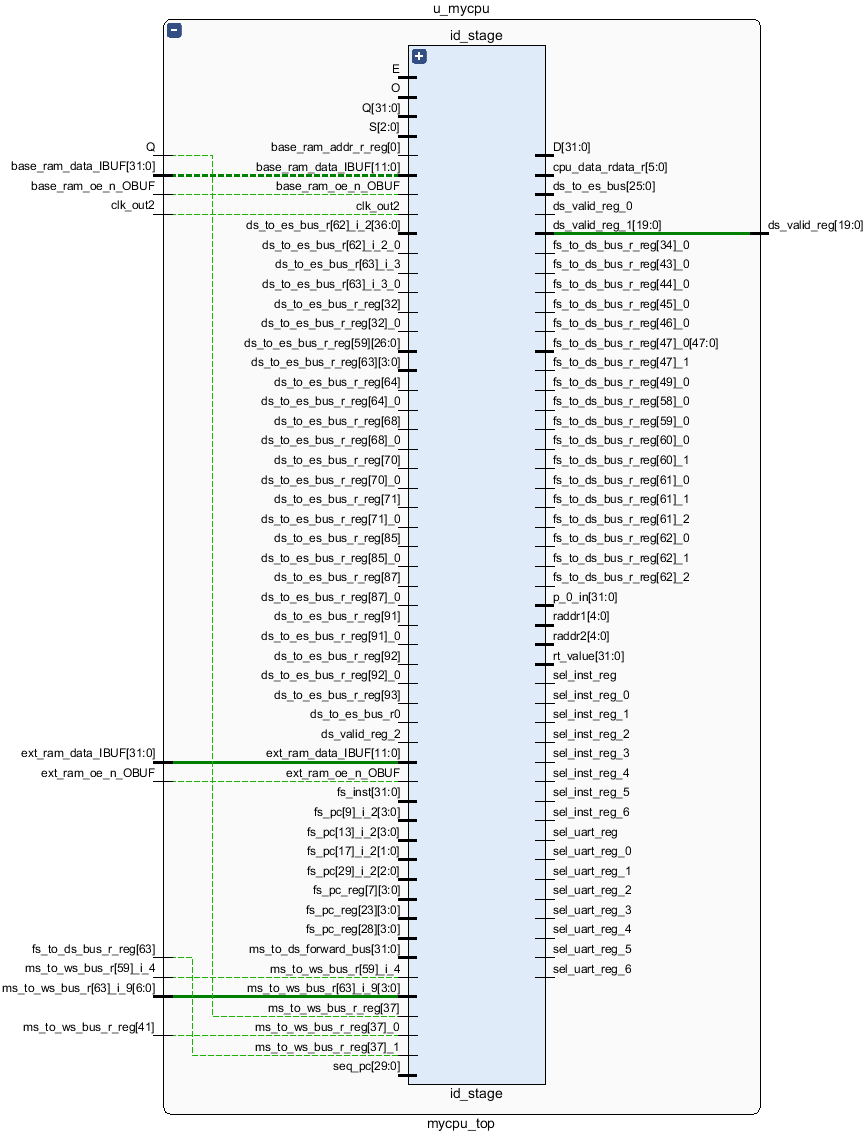


图3 译码阶段顶层模块接口图

1. EXE\_stage模块设计

执行阶段是计算机处理器中的关键组成部分，主要负责对已译码的指令进行特定计算，得出运算结果，并确定写入的寄存器地址或访存地址。在执行阶段，各类指令的运算被逐步完成，包括算术运算、逻辑运算、移位操作、分支跳转等，并将获得的数据前递到译码阶段，并生成执行结果。

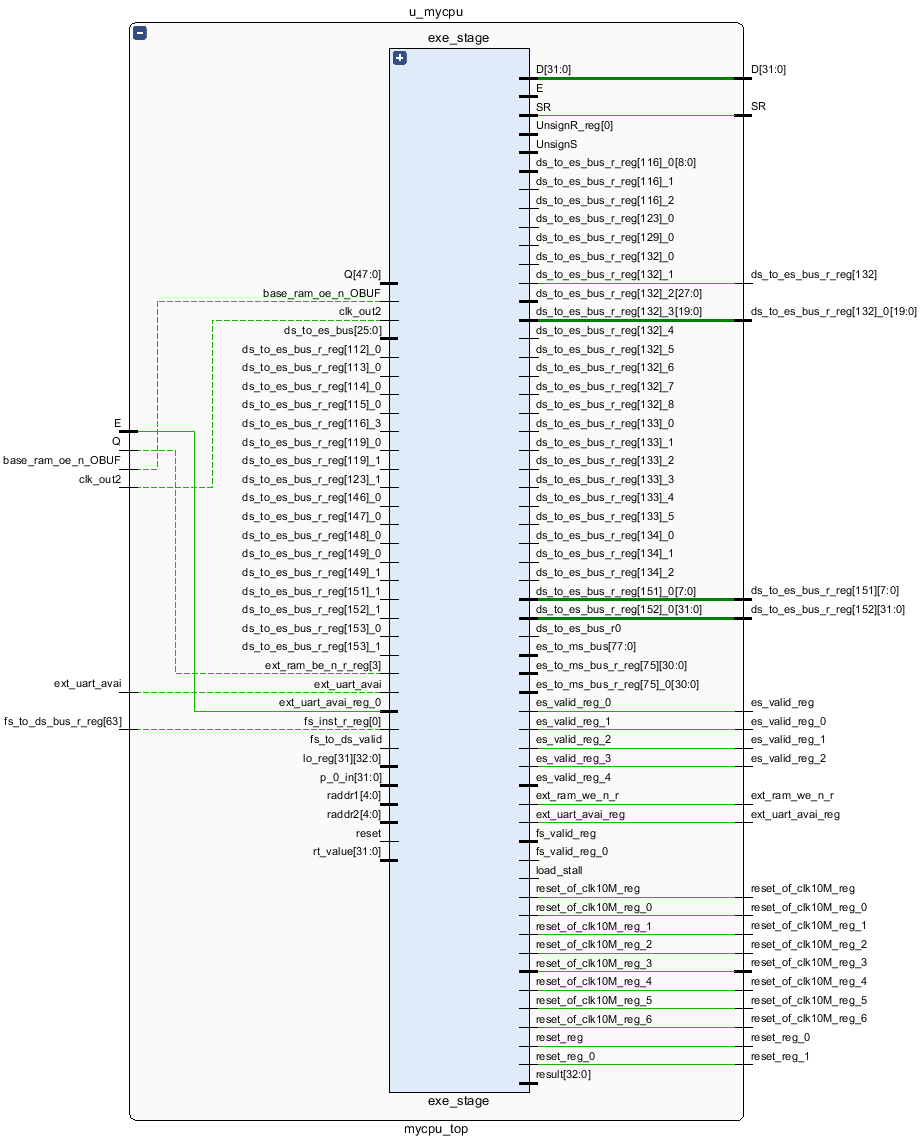


图4 执行阶段顶层模块接口图

1. MEM\_stage模块设计

访存阶段主要处理与内存访问相关的数据传递和操作，通过读取数据RAM，得到访存数据，并设计前递通路将数据前递到译码阶段。

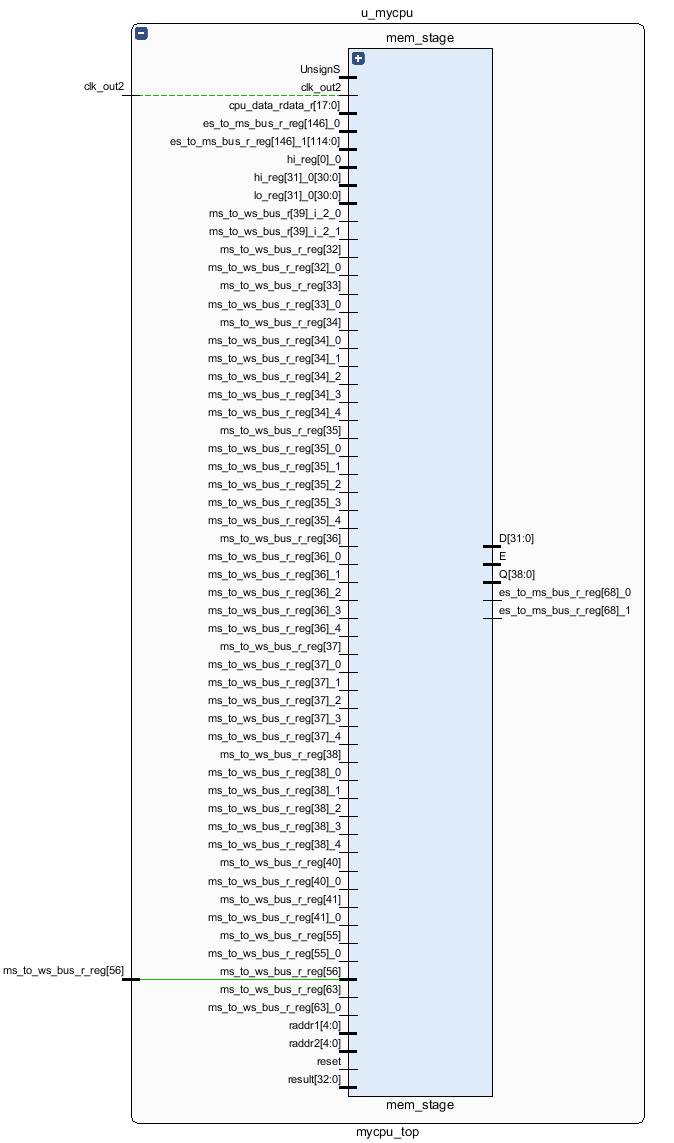


图5 访存阶段顶层模块接口图

（六）WB\_stage模块设计

要作用是将执行阶段计算得出的结果写回到寄存器或内存中，以完成指令的最终执行和数据的更新。并且设计了写回阶段到译码阶段的前递数据通路，实现了数据的前递。在指令执行过程中，各种运算指令会产生结果，这些结果需要保存在相应的寄存器或内存中，以供后续指令使用或读取。

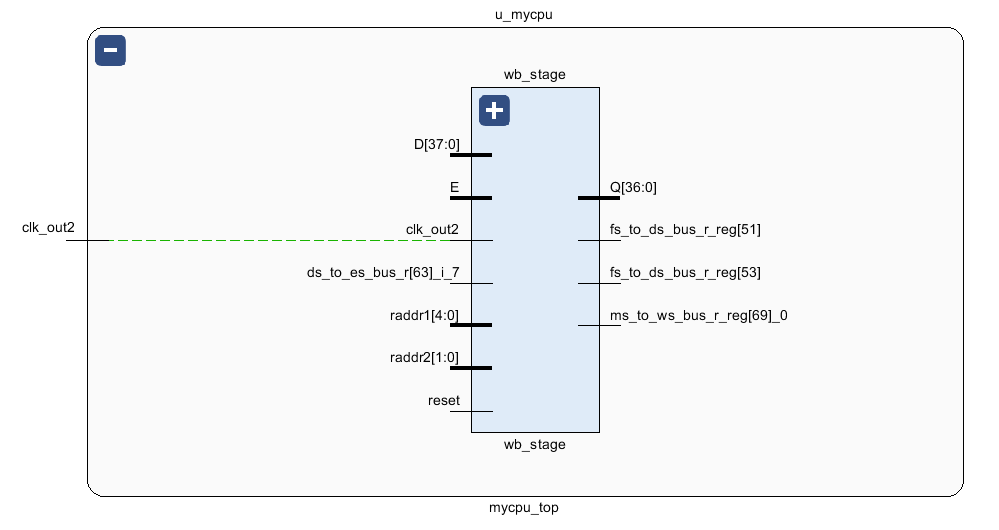


图6 写回阶段顶层模块接口图

三、设计结果

（一）设计交付物说明

thinpad\_top.srcs

├─ sources\_1

│ ├─ new

│ │ ├─ async.v

│ │ ├─ SEG7\_LUT.v

│ │ ├─ thinpad\_top.v **//顶层模块**

│ │ ├─ vga.v

│ │ └─ mycpu **//设计文件**

│ │ ├─ IF\_stage.v **//取指阶段**

│ │ ├─ ID\_stage.v **//译码阶段**

│ │ ├─ EXE\_stage.v **//执行阶段**

│ │ ├─ MEM\_stage.v **//访存阶段**

│ │ ├─ WB\_stage.v  **//写回阶段**

│ │ ├─ alu.v **//alu模块**

│ │ ├─ IF\_stage.v **//取指阶段**

│ │ ├─ mul.v  **//乘法器**

│ │ ├─ mycpu.h  **//宏定义**

│ │ ├─ mycpu\_top.v **//mycpu顶层模块**

│ │ ├─ regfile.v **//寄存器堆**

│ │ └─ tools.v

│ ├─ testbin

│ │ ├─ xingneng

│ │ │ ├─ inst\_ram.coe

│ │ │ ├─ kernel.bin  **//性能测试初始化数据寄存器**

│ │ │ └─ test.s

│ │ ├─ lab3

│ │ │ ├─ inst\_ram.coe

│ │ │ ├─ kernel.bin  **//lab3初始化数据寄存器**

│ │ │ ├─ kernel.elf

│ │ │ └─ test.s

│ │ ├─ lab2

│ │ │ ├─ inst\_ram.coe

│ │ │ ├─ lab2.bin

│ │ │ ├─ lab2.elf

│ │ │ └─ test.s

│ │ └─ lab1

│ │ ├─ lab1.bin

│ │ └─ lab1.S

│ └─ ip **//所使用的的ip核**

│ └─ pll\_example **//时钟分频模块**

│ ├─ mmcm\_pll\_drp\_func\_7s\_mmcm.vh

│ ├─ mmcm\_pll\_drp\_func\_7s\_pll.vh

│ ├─ mmcm\_pll\_drp\_func\_us\_mmcm.vh

│ ├─ mmcm\_pll\_drp\_func\_us\_pll.vh

│ ├─ mmcm\_pll\_drp\_func\_us\_plus\_mmcm.vh

│ ├─ mmcm\_pll\_drp\_func\_us\_plus\_pll.vh

│ ├─ pll\_example.v

│ ├─ pll\_example.veo

│ ├─ pll\_example.xci

│ ├─ pll\_example.xdc

│ ├─ pll\_example.xml

│ ├─ pll\_example\_board.xdc

│ ├─ pll\_example\_clk\_wiz.v

│ ├─ pll\_example\_ooc.xdc

│ └─ doc

│ └─ clk\_wiz\_v6\_0\_changelog.txt

├─ sim\_1

│ ├─ new

│ │ ├─ 28F640P30.v

│ │ ├─ clock.v

│ │ ├─ cpld\_model.v

│ │ ├─ flag\_sync\_cpld.v

│ │ ├─ sram\_model.v

│ │ ├─ tb.sv **//功能仿真**

│ │ └─ include

│ │ ├─ BankLib.h

│ │ ├─ CUIcommandData.h

│ │ ├─ data.h

│ │ ├─ def.h

│ │ ├─ TimingData.h

│ │ └─ UserData.h

│ └─ imports

│ ├─ CFImemory64Mb\_bottom.mem

│ └─ CFImemory64Mb\_top.mem

└─ constrs\_1

└─ new

└─ thinpad\_top.xdc

1. 注意事项

1.请使用Vivado2019.2版本打开本项目，使用其他版本打开可能有兼容问题。

2.使用前请仔细查看上述项目目录，本人已将需要了解的文件加以注释，本地仿真前请修改tb.sv文件中初始化baseram（指令存储器）的文件地址。

（三）设计演示结果

表1功能测试

|  |  |
| --- | --- |
| **测试类型** | **得分** |
| 一级测评 | 100 |
| 二级测评 | 100 |
| 三级测评 | 100 |
| 性能测试 | 100 |

表2 性能测试

|  |  |
| --- | --- |
| **运行程序** | **运行时间** |
| STREM | 0.100s |
| MATRIX | 0.211s |
| CRYPTONIGHT | 0.477s |

四、参考设计说明

①汪文祥、邢金璋.《CPU设计实战》中mycpu的接口定义以及流水线框架的设计。

②龙芯杯NSCSCC2020个人赛开源代码中关于串口通信控制的设计。

链接如下：<https://github.com/fluctlight001/nscscc2022_single_tools>

③乘法器参考了龙芯开源的chiplab项目中的乘法器。

链接如下：<https://gitee.com/loongson-edu/chiplab>

1. 参考文献
2. 汪文祥,邢金璋.CPU设计实战.北京:机械工业出版社,2021.
3. 胡伟武等著.计算机体系结构基础.北京:机械工业出版社,2021.