

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机组成 |
| 姓 名： | 应周骏 |
| 学 院： | 计算机科学与技术学院 |
| 系： | 计算机科学与技术系 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3200103894 |
| 指导教师： | 马德 |

2022 年 6月 20 日

**浙江大学实验报告**

课程名称： 计算机组成 实验类型： 综合

实验项目名称： CPU设计之流水线

学生姓名： 应周骏 专业：计算机科学与技术 学号：3200103894

同组学生姓名： 无 指导老师： 马德

实验地点： 东4-509实验日期：2022 年 5 月 25 日

**一、实验目的和要求**

1.理解流水线CPU的基本原理和组织结构；

2.掌握五级流水线的工作过程和设计方法；

3.理解流水线CPU停机的原理；

4.设计流水线测试程序；

**二、实验内容和原理**

**目标：**

熟悉RISC-V 五级流水线的工作特点，了解流水线处理器的原理，掌握IP核的使用方法，集成并测试CPU。

**内容：**

集成设计流水线CPU，在Exp04的基础上完成利用五级流水线各级封装模块集成CPU替换 Exp04的单周期CPU为本实验集成的五级流水线CPU，并设计流水线测试方案并完成测试。

**原理：**

1. 流水线优势

一个时钟周期完成一条指令所有操作，结构简单，但面对复杂指令集，其电路最长路径严重影响CPU工作频率，则效率太低。

一个时钟周期完成一条指令一个操作，面对单条指令会花费更多时间；但从全局看，各个操作阶段的延时比整个CPU操作时钟延时短，时钟周期有效缩短。

在多周期基础上，利用不同阶段用不同时钟周期，功能部件可复用的特点，将不同指令的不同阶段重叠执行，则效率更高。

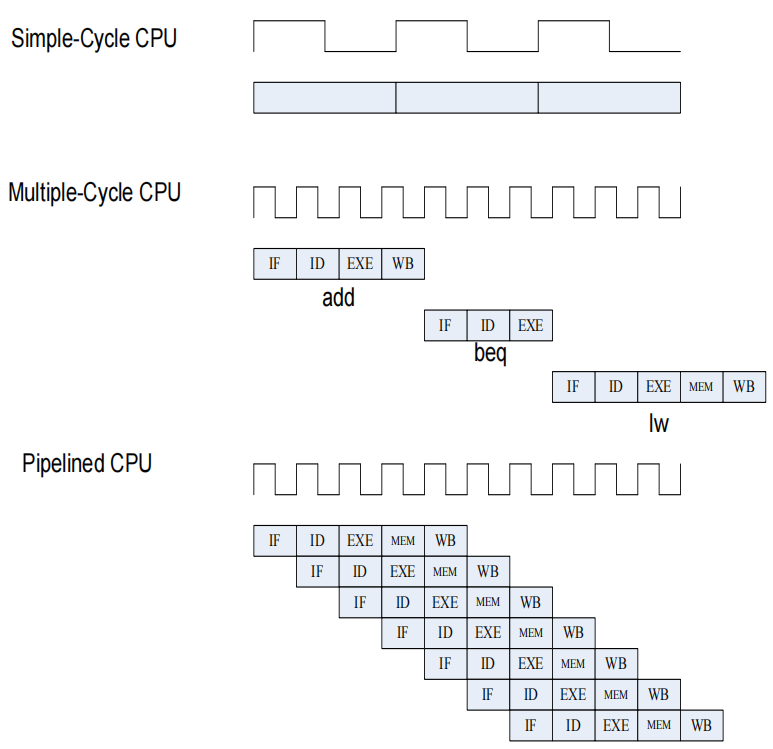


图2.1 不同CPU策略

2. 流水线各阶段

取指：取指阶段涉及程序计数器（PC）和指令存储器（I\_Mem）；程序计数器输出作为地址从指令存储器中读取指令；

IF\_reg\_ID：暂存指令和PC值，以待下一级使用；

译码：译码阶段涉及寄存器堆（RegisterFiles）和译码器、立即数生成单元（ImmGen）；从寄存器堆可以读取操作数，译码器对指令进行解析产生各种各种控制信号，立即数生成单元根据控制信号和输入指令生成各种类型的立即数。

ID\_reg\_Ex：暂存PC值，寄存器读取的数据，立即数和控制信号以待下一级使用；

执行：执行阶段涉及运算单元（ALU）它获取操作数并完成指定的算数运算或逻辑运算；

Ex\_reg\_Mem：暂存运算结果和控制信号，以待下一级使用；

存储器访问：存储器访问阶段涉及数 据 存 储 器 （ D\_Mem ）；Load\Store指令对数据存储器进行读或写；

Mem\_reg\_WB：暂存存储器结果和控制信号，以待下一级使用；

写回：写回阶段涉及寄存器堆（RegisterFiles）；将ALU的运算结果、存储器输出结果、PC+4写回到寄存器堆。

写回阶段结束，一次完整的五级流水操作完成；此时下一次操作进行到存储器访问阶段（如果有）。由于在各级流水线之间插入了寄存器作为数据及控制信号的暂存，从而实现多条指令的重叠而不受影响。

3. 流水线控制

取指控制：取指阶段，读指令存储器和写PC值永远有效，无需控制信号。

译码控制：译码阶段，立即数生成单元需要根据指令类型产生对应输出，ImmSel信号输出；其他信号暂存寄存器。

执行控制：执行阶段，ALU的操作和第二个操作数Src\_B需要选择，ALU\_ctrl、ALUSrc\_B信号输出；其他信号暂存寄存器。

存储器访问控制：存储器访问阶段，需要读写存储器以及根据分支跳转指令判定选择PC转移值，MemRW、Branch、BranchN、Jump输出；其他信号暂存。

写回控制：写回阶段， ALU运算结果、存储器输出等需要选择写回寄存器堆，同时寄存器堆的写使能需要设置；MemToReg、RegWrite信号输出。

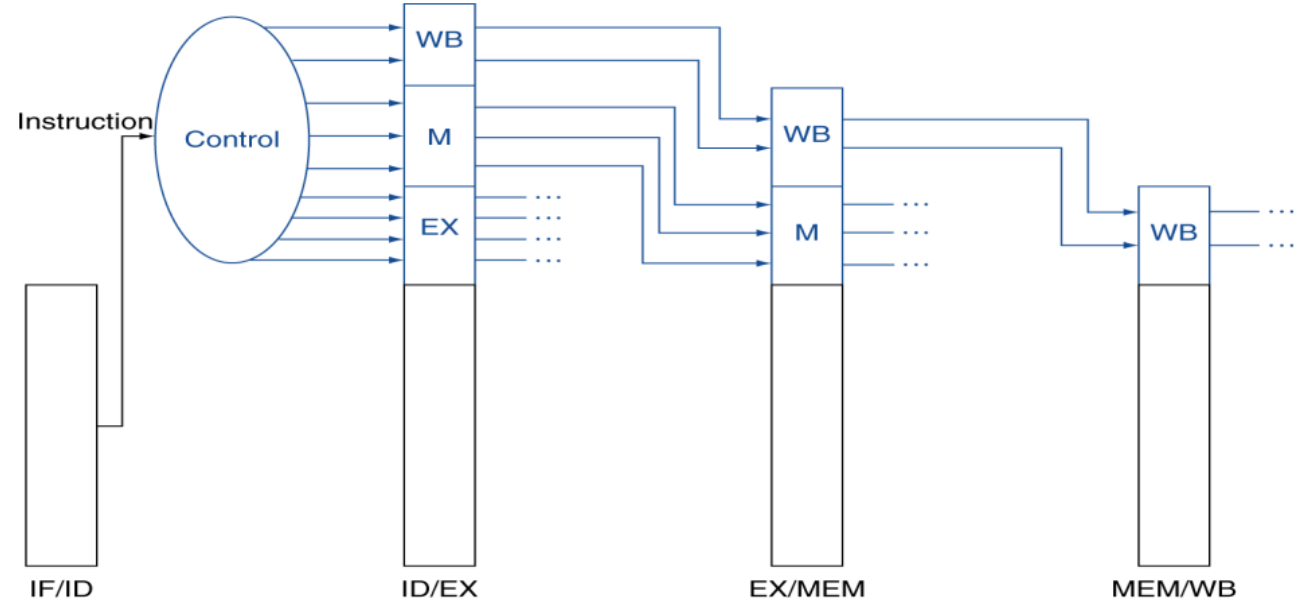


图2.2 流水线控制示意图

**三、实验过程和数据记录**

**1. 工程文件建立**

新建工程文件，命名为“OxExp05\_pipeline”。

**2. CSSTE模块更新**

在实验二的“CSSTE. v”文件基础上，修改部分代码，主要集中在VGA和pipeline\_CPU接口（详见附件）。

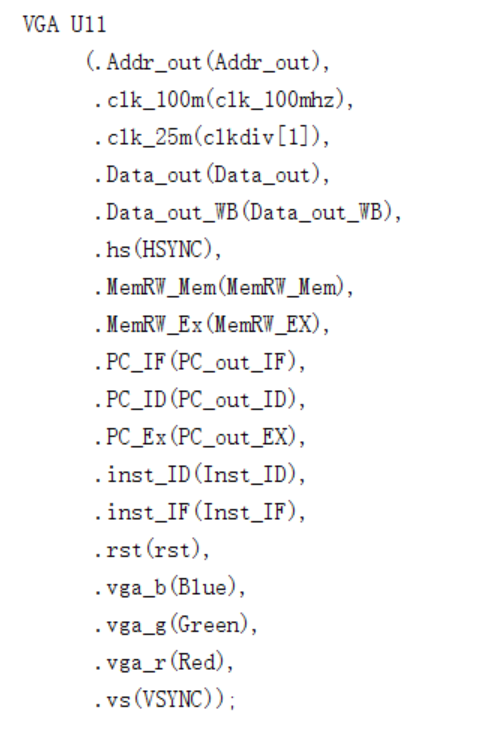


图3.2.1 VGA更新

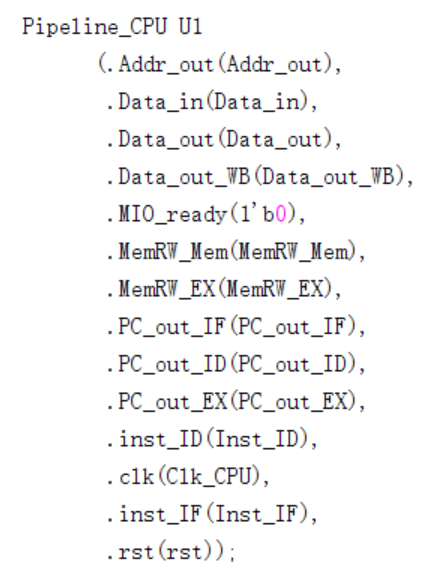


图3.2.2 Pipeline\_CPU接口

**3. Pipeline\_CPU模块设计**

新建“Pipeline\_CPU.v”，依据给出的原理图，连接各给出模块。

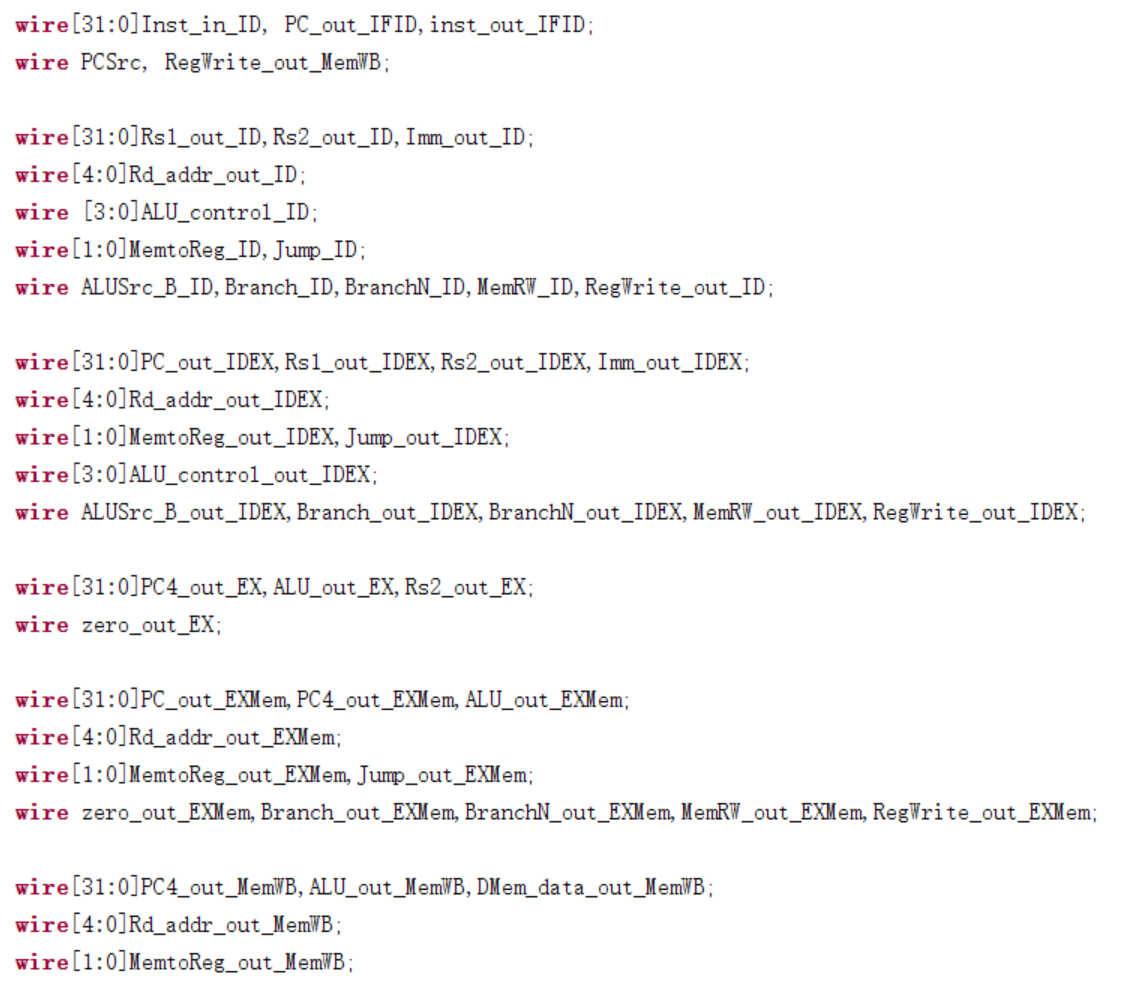
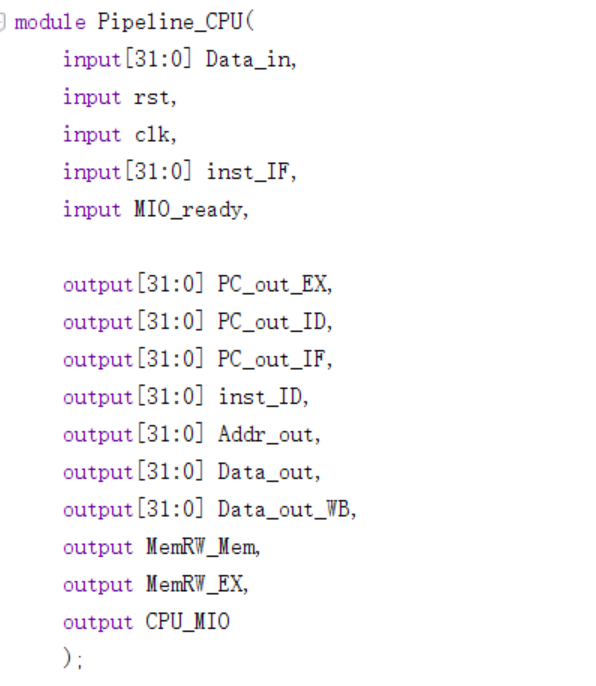


图3.3.1 流水线CPU接口

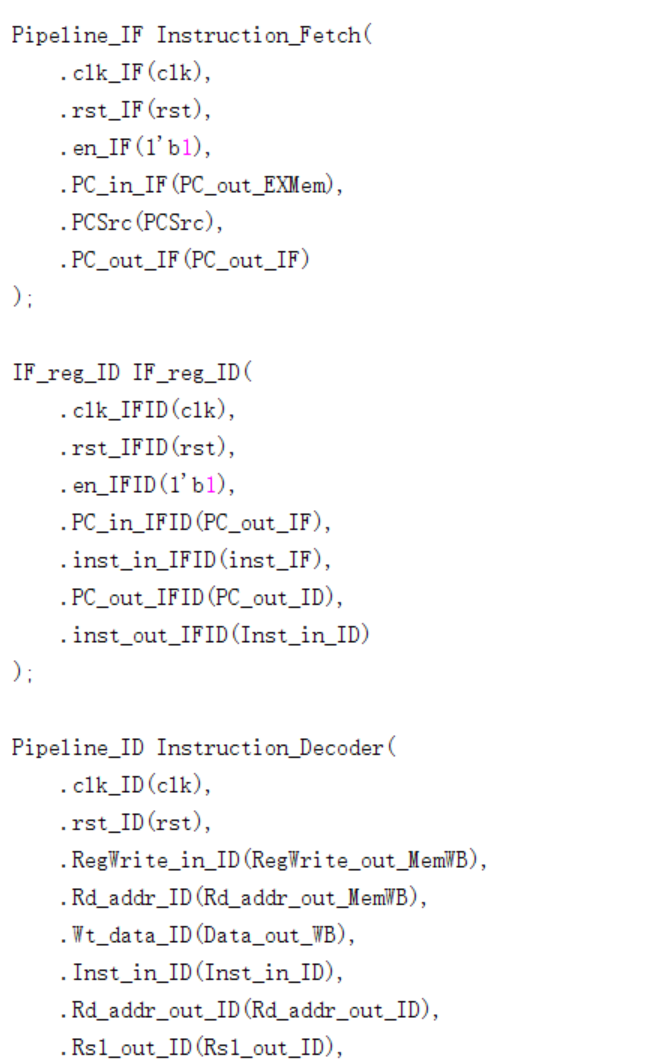


图3.3.2 连接各模块（详细代码见附件）

**4. 五级流水线模块及寄存器设计**

新建“Pipeline\_IF.v”模块，实现取指模块。

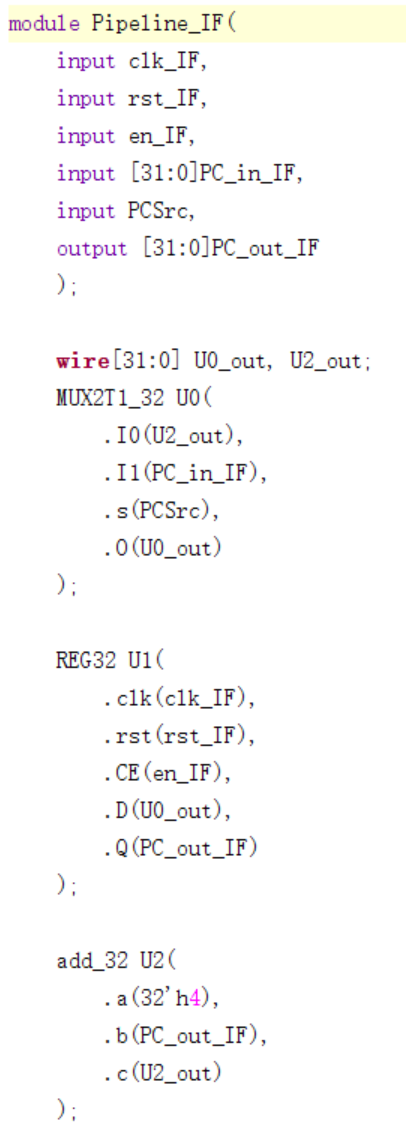


图3.4.1 取指模块代码

新建“IF\_reg\_ID.v”模块，实现取指/译码寄存器模块。

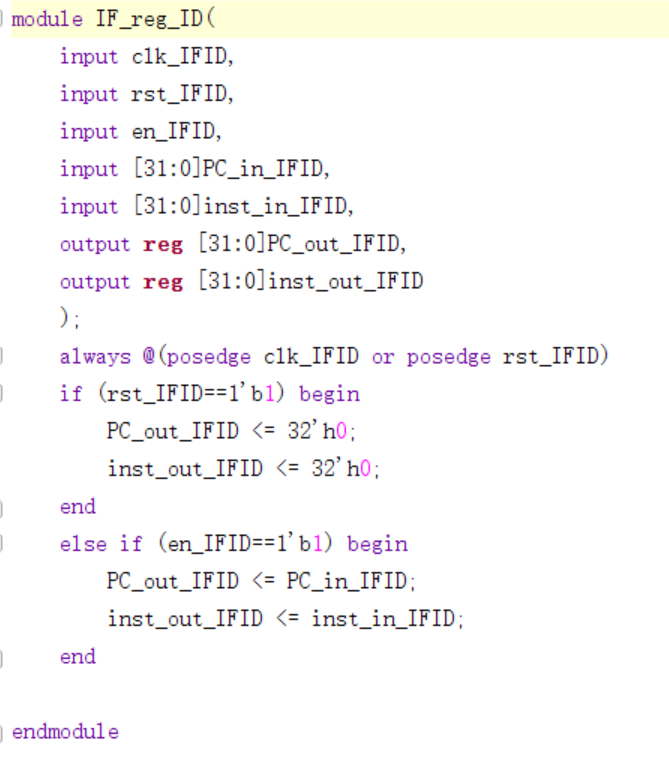


图3.4.2 IF/ID.reg实现

新建“Pipeline\_ID.v”模块，实现译码模块，并修改相应的译码模块，支持该实验需要，但是扩展指令支持，而基础指令不支持的功能。

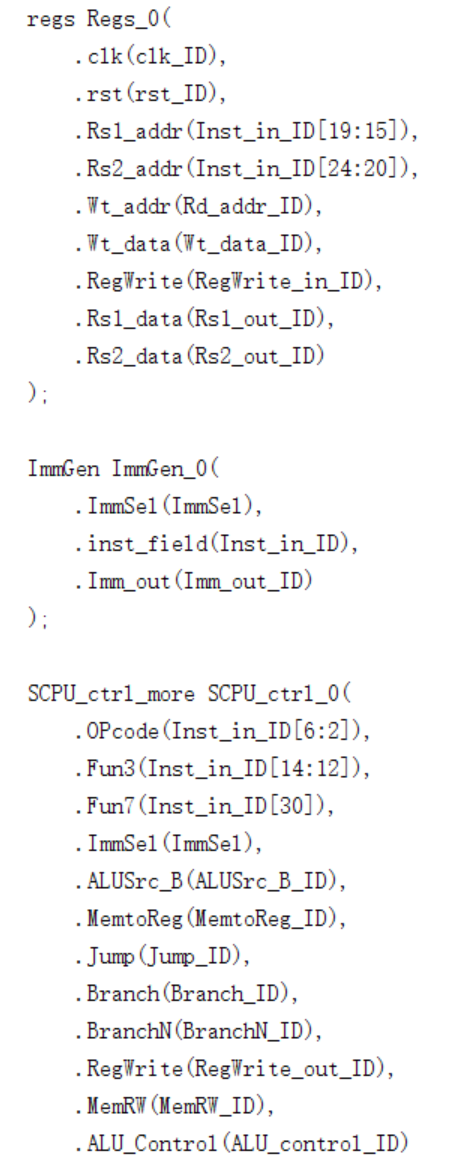
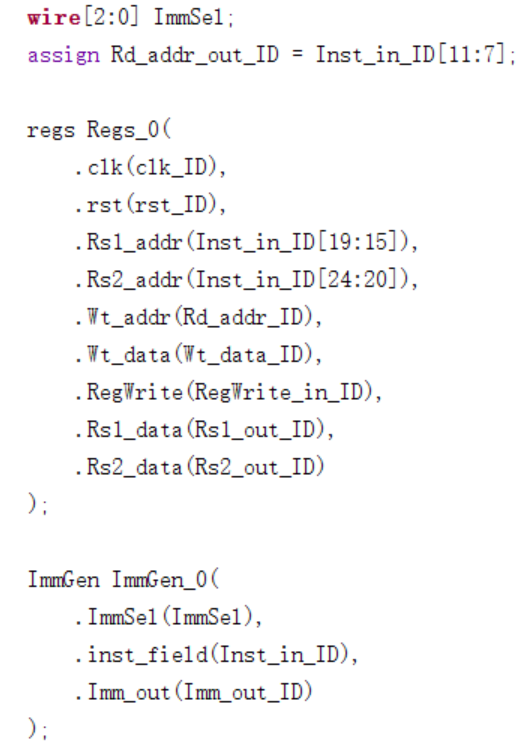


图3.4.3 译码模块代码

新建“ID\_reg\_EX.v”模块，实现译码/执行寄存器模块。

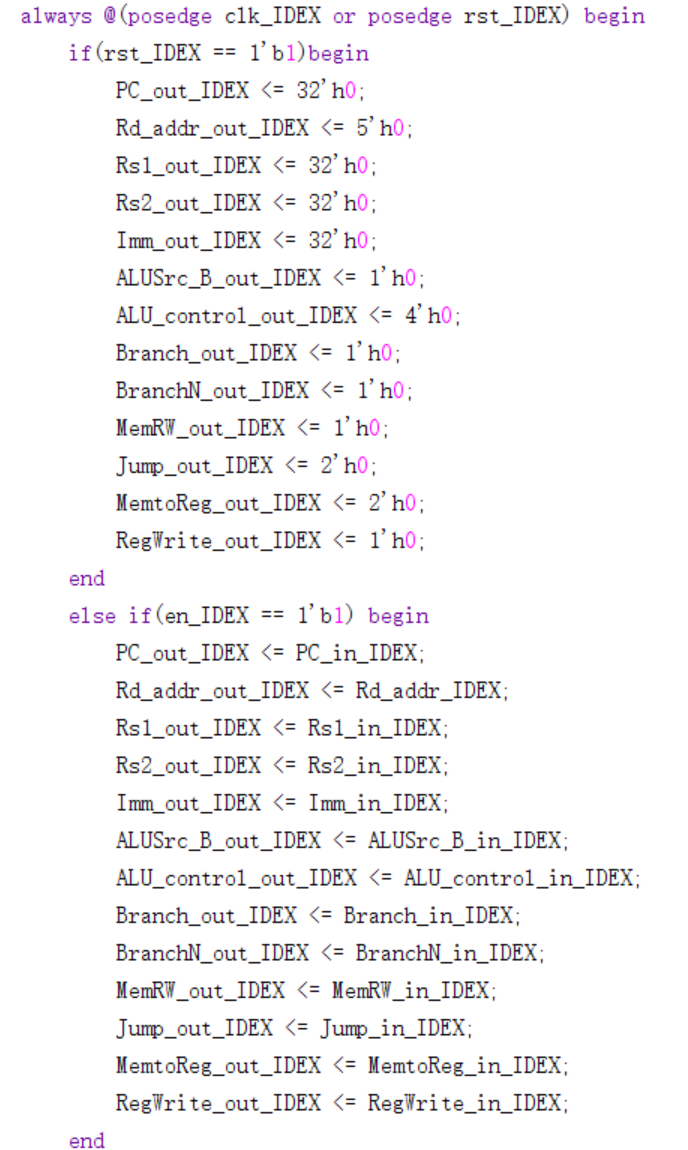


图3.4.4 ID/EX.reg实现

新建“Pipeline\_EX.v”模块，实现执行模块。

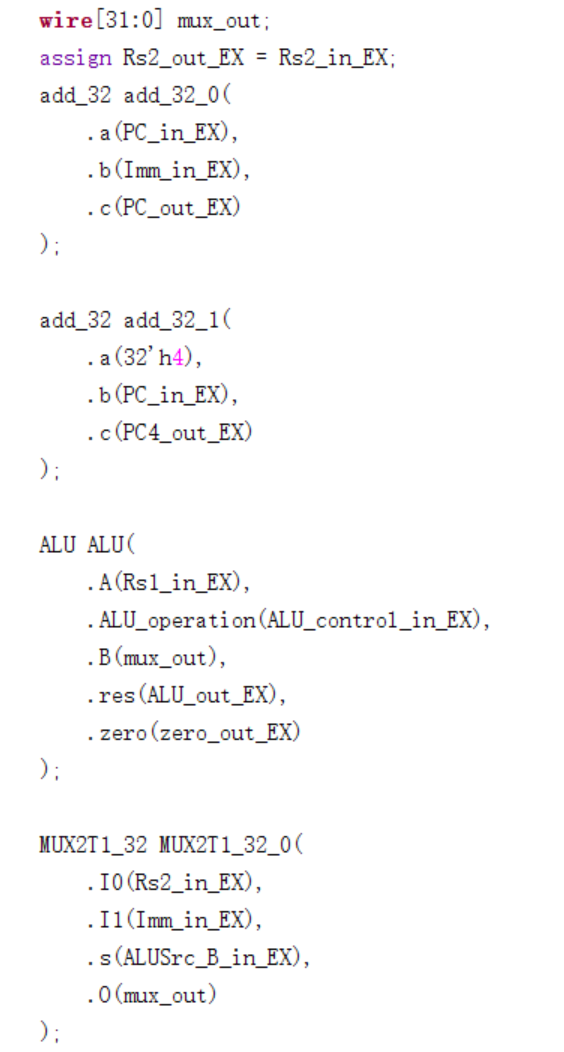


图3.4.5 执行模块代码

新建“EX\_reg\_MEM.v”模块，实现执行/存储器访问寄存器模块。

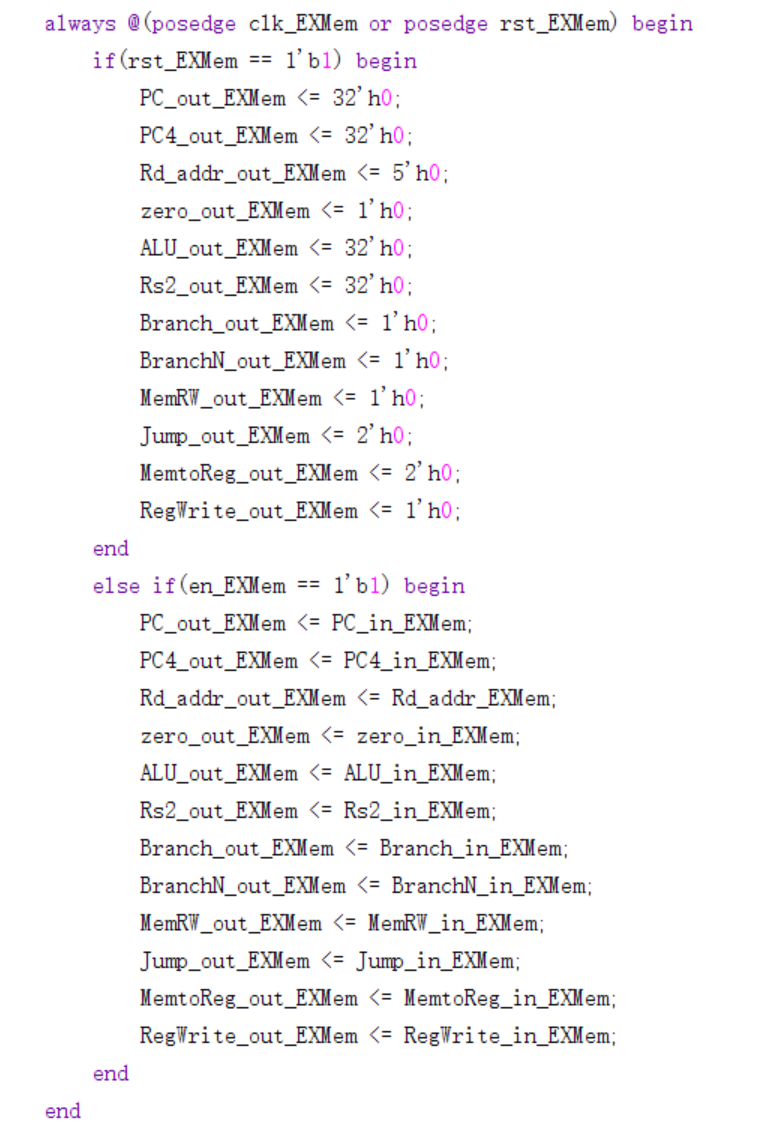


图3.4.6 EX/MEM.reg实现

新建“Pipeline\_MEM.v”模块，实现存储器访问模块。

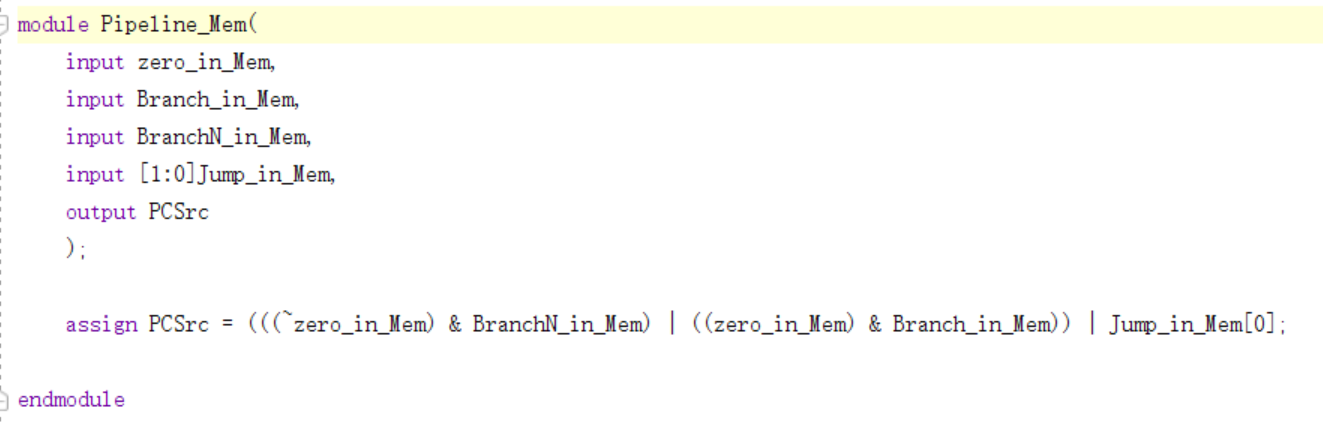


图3.4.7 存储器访问模块代码

新建“MEM\_reg\_WB.v”模块，实现存储器访问/写回寄存器模块。

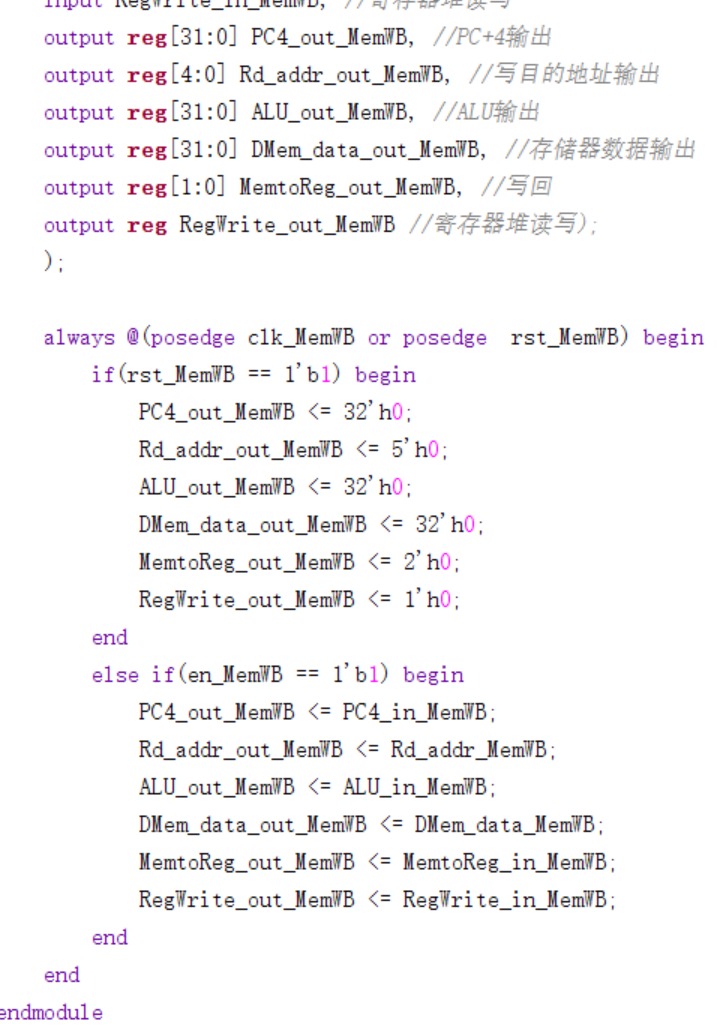


图3.4.8 MEM/WB.reg文件

新建“Pipeline\_WB.v”模块，实现写回模块。

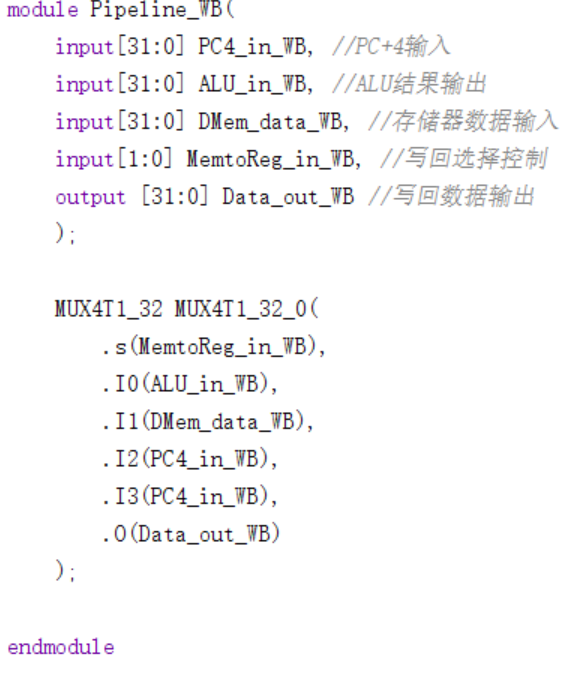


图3.4.9 写回模块代码

**5. 模块集成**

在工程中加入实验二中相关的模块，以及新的VGA模块，修改VgaDisplay模块的相关文件地址为本机实际存储地址。同时，依照此前实验，生成流水线测试指令/冒险测试指令的ROM文件，本实验将不含冒险的指令作为测试。

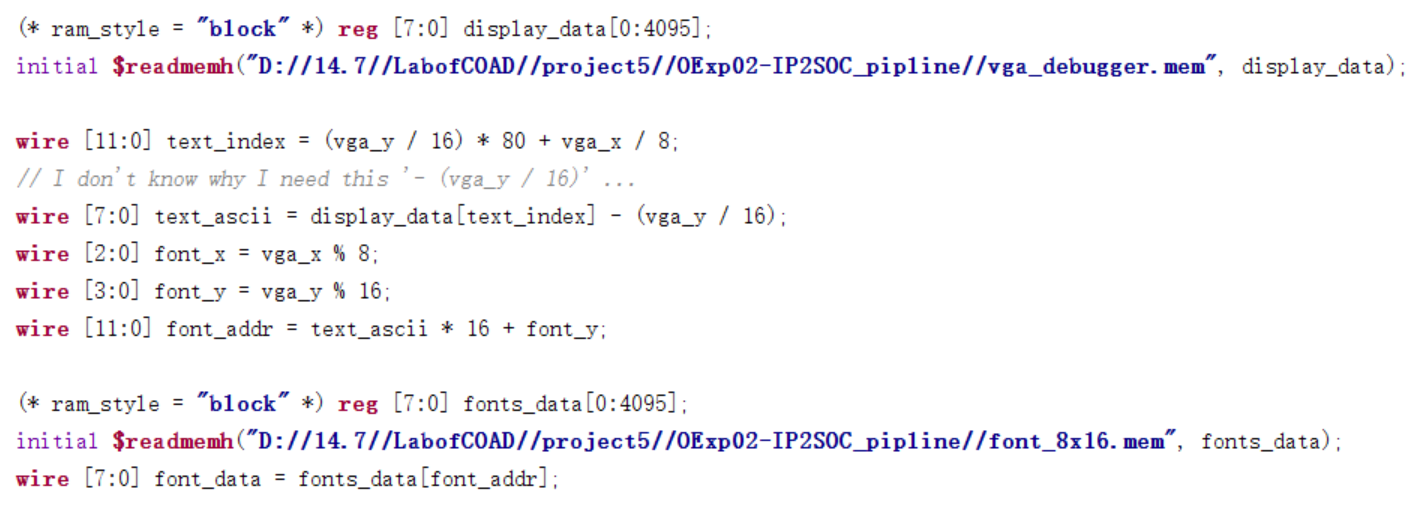


图3.5.1 VGA模块更新

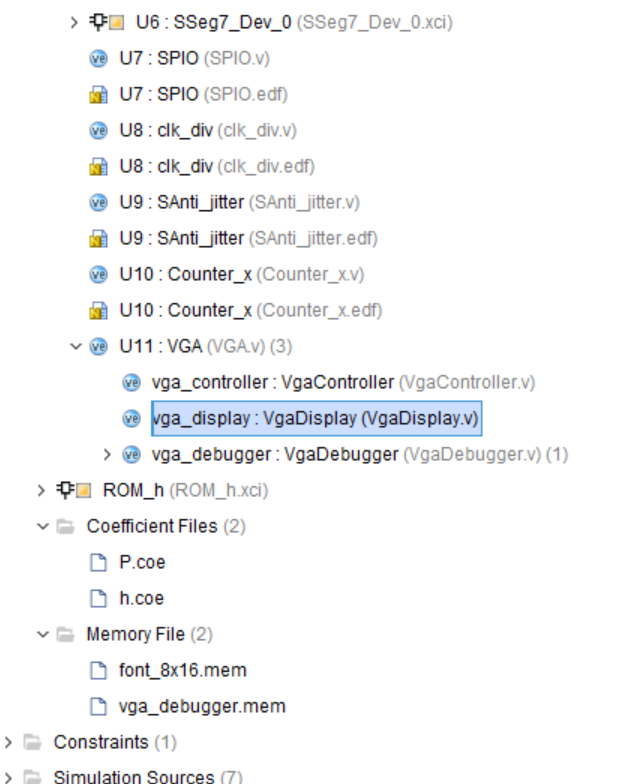
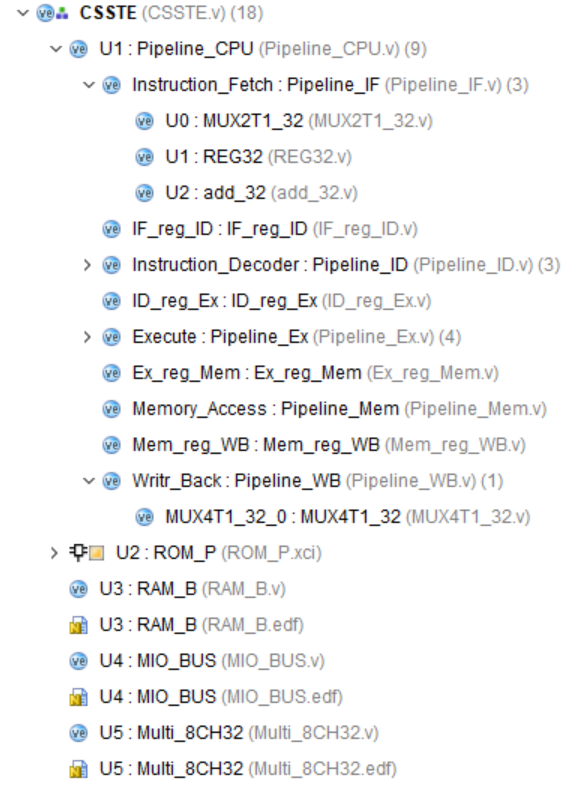


图3.5.2 整体模块架构

**6. 物理验证**

对上述流水线CPU生成bit文件，在SWORD实验板上进行验证。

**四、实验结果分析**

**1. 各模块设计思路**

对于五级流水线的具体模块，按照PPT给出的原理图进行设计。其中，由于PPT给出的整体接口位宽与拓展指令/基础指令均不兼容，故本实验采取了扩展指令模块，并对接口做了相应调整，以实现BranchN指令。

对于寄存器，本实验采用时序逻辑对相应控制信号和数值进行保持/清零。

**2. 物理验证结果**

物理验证结果能够通过验收。同时，由于本实验设计的控制通路模块，对无需跳转的PC值的ALU结果仍然输出有效值，及有值但不用，所以输出PC时在最后一级的结果存在不符合PC+4的条件，属于合理情况。

**五、讨论与心得**

1. 通过本次实验，我掌握了流水线CPU的基本运作原理，基本了解了五级流水线CPU的工作流程。

2. 本次实验中，由于模块之间信号位宽与此前的扩展/基础实验均不一致，因此在调整位宽时花了较多时间，不过整体流程相对顺利，没有遇到很多问题。

3. 本次实验由于时间限制没有进一步实现冒险，希望在暑假有空时进行完善和学习，为未来进一步的硬件学习打下基础。