中期答辩自述文档

1. 项目分工

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 学号 | 实验课时间 | 负责的工作（不是扣分项，但建议写，方便问答） |
| 娄毅彬 | 12310513 | 周一下午 5-6节 王晴老师 | 1、实现Dmem模块  2、支持字节（sb/lb/lbu）、半字（sh/lh/lhu）和字（sw/lw）访问。内存读写通过 Data\_Memory\_ip 完成。  3、支持内存映射 IO：写地址 0xFFFF\_FC60 对应 LED 输出，读地址 0xFFFF\_FC70 和 0xFFFF\_FC71 分别对应读取开关的低 8 位和高 8 位。根据地址判断是否是 IO 操作，并据此决定数据来源和写入目标。  4、支持对写数据的对齐与掩码处理，保证不同类型存储的正确性 |
| 方酉城 | 12310519 | 周一下午 5-6节 王晴老师 | 1.设计TOP模块整合Decoder,ALU,Controller等子模块  2.添加slt,sltu,blt,bltu,jar等测试场景需要的新指令加入指令集，使得CPU允许支持新指令 |
| 芮煜涵 | 12310520 | 周一下午 5-6节 王晴老师 | 1、负责 ALU 模块设计与实现 完成算术/逻辑/位移/比较等操作  2、完成 slt、sltu、sll、srl 等新指令的 testbench 编写与仿真 验证功能正确性，保证 CPU 能通过场景测试中的相关功能点。 |

1. 代码规范：结构化设计（是）：将 CPU 分为多个子模块（ IFetch、ALU、Decoder、Controller、Dmem、CPU\_TOP 等）

命名规范：使用驼峰命名或下划线分隔，如 ReadData1、switch\_in、io\_output 等，风格统一

注释要求：关键逻辑部分添加了注释，尤其是在控制信号生成、ALU 运算控制等部分

符号化常量的定义及使用：使用了 parameter 或 localparam 来统一管理常量，如地址映射和操作码值，避免 magic number

1. CPU特性 (单周期，后续计划实现pipeline), CPU时钟( 23 MHz) , **ISA**( RISC-V)

**存储方案**(哈佛), **IO方案**(MMIO) (不是评分点，已确定好建议做标记)

本次大作业中要实现的指令集(请罗列)：

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

（后续会拓展更多指令）

1. CPU架构设计（内部模块及连线关系）/接口设计说明

当前的 CPU 采用模块化结构设计，顶层模块命名为 CPU\_TOP，其内部包括以下子模块：

* IFetch：负责指令获取、PC管理；
* Controller：根据指令字段生成控制信号；
* Decoder：完成寄存器堆读写及立即数生成；
* ALU：执行算术和逻辑运算；
* Dmem：访问内存/IO设备（支持 MMIO 地址映射）；

这些模块在顶层按如下方式连接（主要是数据通路）：

 IFetch → Controller\Decoder → ALU → Dmem → Decoder（写回）

* **程序计数器 PC** 是在 IFetch 模块中管理的。在每个周期内，根据控制器提供的跳转信号（Branch）和 ALU 计算的 zero 标志来判断是否进行分支跳转（pc\_next = pc + imm32 或 pc + 4），并更新下一条指令地址。
* 控制信号的生成在 Controller 中完成，结合 opcode 和 funct3 输出 ALU 操作类型、寄存器写使能、内存读写等信号。
* **拨码开关、LED由 Dmem 模块统一管理，使用MMIO（Memory-Mapped IO）方式**接入。具体来说：
  + 0xFFFF\_FC60：LED 输出地址（对应0(x31)）；
  + 0xFFFF\_FC70 / FC71：拨码开关（低/高8位）输入地址；

Dmem 模块会根据地址判断是否为 IO 访问，自动与实际硬件进行数据交换，同时兼容 load/store 指令格式。

1. CPU指令于控制信号的关系 (可参考以下表格)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | Branch | ALUOp | ALUsrc | MemRead | MemWrite | MemtoReg | RegWrite |
| lw | 0 | 00 | 1 | 1 | 0 | 1 | 1 |
| sw | 0 | 00 | 1 | 0 | 1 | 0 | 0 |
| add | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| beq | 1 | 01 | 0 | 0 | 0 | 0 | 0 |
| jal | 0 | 00 | X | 0 | 0 | 0 | 1 |
| jalr | 0 | 00 | 1 | 0 | 0 | 0 | 1 |
| lui | 0 | 00 | 1 | 0 | 0 | 0 | 1 |

1. 项目实施的情况，已完成的项目代码

|  |  |  |
| --- | --- | --- |
| 单项 | 完成百分比 | 备注（待完成/待解决） |
| 文档 | 70% | 后续补充外设部分说明 |
| CPU模块设计 | 90% | |  | | --- | |  | |
| CPU顶层设计 | 90% | 顶层模块 CPU\_TOP 构建完成 |
| CPU 单元测试 | 90% | 仍需通过上板测试 |
| CPU 集成测试 | 90% | 仍需通过上板测试 |
| IO 设计 | 50% | 仍需通过上板测试 |
| IO 测试 | 50% | 后续加入更多外设验证 |
| 程序下发方式 | 30% | 待整合程序烧写流程 |
| 工具链 | 30% | 待支持多平台集成 |
| 基本测试场景1 | 80% |  |
| 基本测试场景2 | 10% |  |
| 附加测试场景 | 0% | 准备扩展外设场景，如串口、中断等 |

1. 计划使用/开发的工具链

主要使用 Vivado 进行代码编写、综合和仿真，测试部分编写 Verilog testbench 来完成。目前使用的 coe 文件生成器和指令转码器是由老师提供的工具，后续尝试使用 UART 实现多个测试场景的动态切换。

1. 项目整体进度（\_\_50\_\_%）当前困难或问题、预计最终答辩时间、后续计划

主要问题是还未进行上板测试，尚未验证在实际硬件环境下的执行稳定性。我们预计在13周上板调试，争取在第15周完成提前答辩。

后续计划包括继续完善指令集的支持；同时计划在单周期的基础上拓展出流水线版本，用于实现 bonus。尝试使用 UART 实现多个测试场景的动态切换。