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） |
| 莫丰源 | 12311805 | Monday56\_WangQing | DataMem和RAM的IP核设计与测试 |
| 施皓天 | 12312214 | Monday56\_WangQing | 整体代码框架及除了ALU和IFetch和内存之外的模块编写与测试 |
| 郭珈伊 | 12311851 | Monday56\_WangQing | ALU和IFetch模块和测试 |

1. 代码规范：结构化设计（是\否）

命名规范：是

注释要求：是

符号化常量的定义及使用：是

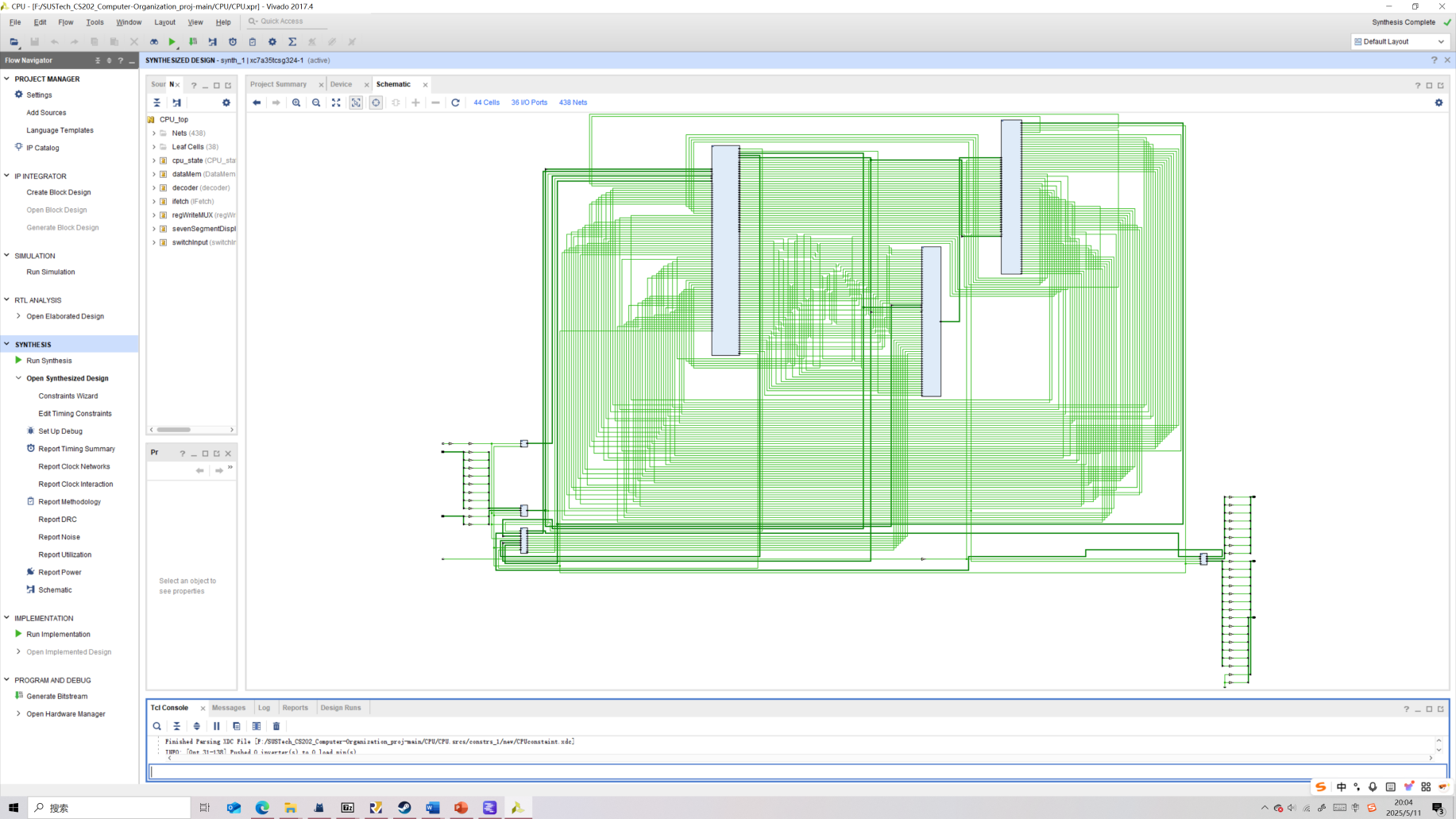
1. CPU特性 (单周期 \ ~~pipeline~~), CPU时钟( 50 MHz) , **ISA**( RISC-V \ ~~LA~~ \ ~~ARM~~)

**存储方案**(~~冯诺伊曼~~ \ 哈佛), **IO方案**( ~~中断~~ \ 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)：

lb、lh、lw、lbu、lhu、sb、sh、sw、addi、slli、slti、sltiu、xori、srli、srai、ori、andi、add、sub、sll、slt、sltu、xor、srl、sra、or、and、beq、bne、blt、bge、bltu、bgeu、jal、jalr、auipc、lui

1. CPU架构设计（内部模块及连线关系）/接口设计说明



（1）IFetch

|  |  |  |
| --- | --- | --- |
| 端口名称 | 方向 | 来源/去向 |
| clk | 输入 | 顶层输入 clk。 |
| rst | 输入 | 顶层输入 rst。 |
| branch | 输入 | control 模块的输出 branch。 |
| zero | 输入 | ALU 的输出 zero。 |
| imm32 | 输出 | 连接到顶层信号 imm，并作为指令立即数提供给 decoder 和 ALU。 |
| pc | 输出 | 为 ALU 提供 PC 值（通过 PCtoALU 控制是否选中）。 |
| instruction | 输出 | 指令存储器中的当前指令，连接到 decoder、control、ALU 的输入。 |
| IFen | 输入 | 来自 CPU\_state 的输出 IFen（控制是否使能指令获取阶段）。 |

（2）decoder

|  |  |  |
| --- | --- | --- |
| 端口名称 | 方向 | 来源/去向 |
| clk | 输入 | 顶层输入 clk。 |
| rst | 输入 | 顶层输入 rst。 |
| instruction | 输入 | IFetch 的输出 instruction。 |
| writeData | 输出 | 连接到 regWriteMUX 的选择结果（regWriteData），最终写入寄存器文件。 |
| immediate | 输出 | 顶层信号 imm，作为立即数提供给 ALU 和控制单元。 |
| regWrite | 输出 | 控制是否写入寄存器的使能信号。 |
| readData1 | 输出 | 从寄存器读取的第一个数据，发送到ALU |
| readData2 | 输出 | 从寄存器读取的第二个数据，发送到ALU |
| WBen | 输入 | 来自 CPU\_state 的输出 WBen，标志是否处于写回阶段。 |

（3）control

|  |  |  |
| --- | --- | --- |
| 端口名称 | 方向 | 来源/去向 |
| opcode | 输入 | 指令的 instruction[6:0]（来自 decoder）。 |
| branch | 输出 | 控制是否跳转分支，发送给 IFetch 和 ALU。 |
| memRead | 未使用 |  |
| memtoReg | 输出 | 发送给 regWriteMUX，选择写入寄存器的数据源（ALU 或内存）。 |
| ALUop | 输出 | 控制 ALU 运算类型，发送给 ALU。 |
| memWrite | 输出 | 决定是否执行写内存操作，发送给 DataMem。 |
| ALUsrc | 输出 | 控制 ALU 的第二个操作数来源（寄存器或立即数）。 |
| regWrite | 输出 | 发送给 decoder 以控制寄存器写入使能。 |
| PCtoALU | 输出 | 决定 ALU 是否使用 PC 值作为第一个操作数（如跳转指令）。 |

（4）regWriteMUX

|  |  |  |
| --- | --- | --- |
| 端口名称 | 方向 | 来源/去向 |
| memtoReg | 输入 | 来自 control 的输出。 |
| ALUResult | 输入 | ALU 的运算结果。 |
| DataRead | 输入 | 来自DataMem 的读数据 |
| regWriteData | 输出 | 连接到 decoder 和寄存器文件的写入数据端口。 |

（5）DataMem

|  |  |  |
| --- | --- | --- |
| 端口名称 | 方向 | 来源/去向 |
| address | 输入 | ALU 的结果 ALUResult。 |
| memOp | 输入 | 指令的 funct3 即instruction [14:12]，决定内存操作类型（如 LW, SW）。 |
| readData | 输出 | 内存读取数据，连接到 regWriteMUX。 |
| memWrite | 输入 | 来自 control 的 memWrite，控制是否执行 Store 操作。 |
| writeData | 输入 | 寄存器的第二个读出值 readData2（作为存储数据）。 |
| clk | 输入 | 顶层输入。 |
| rst | 输入 | 顶层复位信号。 |
| IOin | 输入 | 来自 switchInput 的 IOin，用于 I/O 写入操作。 |
| IOout | 输出 | IO输出，发送给七段显示模块。 |
| MEMen | 输出 | 连接到 CPU\_state 的输入，控制内存阶段使能。 |

（6）ALU

|  |  |  |
| --- | --- | --- |
| 端口名称 | 方向 | 来源/去向 |
| ReadData1 | 输入 | 寄存器文件的第一个读数据（来自 decoder）。 |
| ReadData2 | 输入 | 寄存器文件的第二个读数据（来自 decoder）。 |
| imm32 | 输入 | 立即数值，来自 IFetch 的输出。 |
| ALUSrc | 输入 | control 决定是否选择立即数作为操作数。 |
| PCtoALU | 输入 | 是否使用 PC 作为第一个操作数（如J类型指令）。 |
| ALUOp | 输入 | control 定义的运算类型 |
| funct3 | 输入 | 指令的 funct3[14:12] |
| funct7 | 输入 | 指令的 funct7[31:25] |
| pc | 输入 | 当前 PC 值，来自IFetch。 |
| ALUResult | 输出 | 运算结果，连接到dataMem、regWriteMUX 和 DataMem。 |
| zero | 输出 | 标志位，用于分支条件判断（发送给 IFetch）。 |

（7）sevenSegmentDisplay

|  |  |  |
| --- | --- | --- |
| 端口名称 | 方向 | 来源/去向 |
| clk | 输入 | 顶层时钟。 |
| rst | 输入 | 异步复位信号。 |
| IOout | 输入 | DataMem 的输出 |
| seg, seg1 | 输出 | 驱动七段数码管的段选线。 |
| an | 输出 | 控制不同位点亮。 |

（8）switchInput

|  |  |  |
| --- | --- | --- |
| 端口名称 | 方向 | 来源/去向 |
| clk | 输入 | 顶层时钟，触发开关读取。 |
| rst | 输入 | 复位信号（清空输入缓冲）。 |
| sw | 输入 | 用户通过开关提供的8位输入值。 |
| choose | 输入 | 选择 sw 的目标字节段。 |
| IOin | 输出 | 将开关数据转换为32位总线信号，发送给 DataMem 的 I/O 接口。 |

1. CPU指令于控制信号的关系

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | Branch | ALUOp | ALUsrc | MemRead | MemWrite | MemtoReg | RegWrite |
| lb | 0 | 00 | 1 | 1 | 0 | 1 | 1 |
| lh | 0 | 00 | 1 | 1 | 0 | 1 | 1 |
| lw | 0 | 00 | 1 | 1 | 0 | 1 | 1 |
| lbu | 0 | 00 | 1 | 1 | 0 | 1 | 1 |
| lhu | 0 | 00 | 1 | 1 | 0 | 1 | 1 |
| sb | 0 | 00 | 1 | 0 | 1 | 0 | 0 |
| sh | 0 | 00 | 1 | 0 | 1 | 0 | 0 |
| sw | 0 | 00 | 1 | 0 | 1 | 0 | 0 |
| addi | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| slli | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| slti | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| sltiu | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| xori | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| srli | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| srai | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| ori | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| andi | 0 | 10 | 1 | 0 | 0 | 0 | 1 |
| add | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| sub | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| sll | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| slt | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| sltu | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| xor | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| srl | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| sra | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| or | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| and | 0 | 10 | 0 | 0 | 0 | 0 | 1 |
| beq | 1 | 01 | 0 | 0 | 0 | 0 | 0 |
| bne | 1 | 01 | 0 | 0 | 0 | 0 | 0 |
| blt | 1 | 01 | 0 | 0 | 0 | 0 | 0 |
| bge | 1 | 01 | 0 | 0 | 0 | 0 | 0 |
| bltu | 1 | 01 | 0 | 0 | 0 | 0 | 0 |
| bgeu | 1 | 01 | 0 | 0 | 0 | 0 | 0 |
| jal | 1 | 11 | 1 | 0 | 0 | 0 | 1 |
| jalr | 1 | 11 | 1 | 0 | 0 | 0 | 1 |
| auipc | 0 | 11 | 1 | 0 | 0 | 0 | 1 |
| lui | 0 | 11 | 1 | 0 | 0 | 0 | 1 |

注：表格数据基于 control.v 和 ALU.v 模块的逻辑，ALUOp 的值分别为 00(加载/存储)、01(分支)、10(R/I型)、11(跳转/U型)。

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 | 0 |  |
| CPU模块设计 | 100 |  |
| CPU顶层设计 | 100 |  |
| CPU 单元测试 | 50 |  |
| CPU 集成测试 | 100 |  |
| IO 设计 | 100 |  |
| IO 测试 | 0 |  |
| 程序下发方式 | 0 |  |
| 工具链 | 0 |  |
| 基本测试场景1 | 0 |  |
| 基本测试场景2 | 0 |  |
| 附加测试场景 | 0 |  |

1. 计划使用/开发的工具链  
   目前无
2. 项目整体进度（\_\_70\_\_%）当前困难或问题、预计最终答辩时间、后续计划