

（深圳）

# 课程报告

开课学期： 2023夏季

课程名称： 计算机设计与实践

项目名称： 基于miniRV的SoC设计

项目类型： 综合设计型

课程学时： 56 地点： T2615

学生班级：

学生学号：

学生姓名：

评阅教师：

报告成绩：

实验与创新实践教育中心制

2023年7月

注：本设计报告中各个部分如果页数不够，请同学们自行扩页。原则上一定要把报告写详细，能说明设计的成果、特色和过程。报告应该详细叙述整体设计，以及设计中的每个模块。设计报告将是评定每个人成绩的重要组成部分（**设计内容及报告写作**都作为评分依据）。

|  |  |
| --- | --- |
| 设计概述（罗列出所有实现的指令，以及单周期/流水线CPU频率） | |
| 共实现24条miniRV指令：  移位指令： sll, sllli, srl, srli, sra, srai  算数运算指令：add, addi, sub, lui  逻辑运算指令：xor, xori, or, ori, and, andi  跳转链接指令：jal, jalr  加载指令： lw  存储指令： sw  分支指令： beq, bne, blt, bge  单周期CPU频率：50 MHz  流水线CPU频率：50 MHz | |
| 设计的主要特色（除基本要求以外的设计） | |
| 单周期CPU实现了50 MHz的频率 | |
| 资源使用、功耗数据截图（Post Implementation；含单周期、流水线2个截图） | |
| 单周期：  流水线： |

1 单周期CPU设计与实现

1.1 单周期CPU数据通路设计

|  |
| --- |
| 要求：贴出完整的单周期数据通路图，无需画出模块内的具体逻辑，但要标出模块的接口信号名、模块之间信号线的信号名和位宽，并用文字阐述各模块的功能。 |
| NPC模块：   * 根据当前PC值、ALU运算值、偏移量、跳转标识、控制信号决定下一条指令的地址； * 产生pc4信号（pc4 = pc + 4），用于写回寄存器。   PC模块：   * 产生新一条指令地址。   IROM模块：   * 根据PC模块的地址获取指令。   Controller模块：   * 根据指令产生控制信号，包括立即数扩展控制信号sext\_op、下一条指令控制信号npc\_op，数据存储器写入控制信号ram\_we，ALU控制信号alu\_op，ALU输入选择信号alub\_sel，寄存器堆写入控制信号rf\_we，寄存器堆写入选择信号rf\_wsel。   SEXT模块：   * 按照Controller模块产生的立即数扩展控制信号sext\_op将操作数扩展至32位。   RF模块：   * 存取寄存器的值。   ALU模块：   * 根据Controller模块产生的ALU控制信号alu\_op，进行对应的运算。   DRAM模块:   * 根据数据存储器写入控制信号ram\_we，读/写数据存储器。 |

1.2 单周期CPU模块详细设计

|  |
| --- |
| 要求：以表格的形式列出各个部件的接口信号、位宽、功能描述等，并结合图、表、核心代码等形象化工具和手段，详细描述各个部件的关键实现。 |
| NPC模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | NPC | in | PC | 32 | 获取当前PC值 | | NPC | in | aluin | 32 | 获取ALU计算值 | | NPC | in | offset | 32 | 获取跳转偏移量 | | NPC | in | br | 1 | 判断是否跳转 | | NPC | in | op | 2 | 选择NPC所执行的操作 | | NPC | out | npc | 32 | 下一条指令的地址 | | NPC | out | pc4 | 32 | PC+4，用于写回 |   */\* NPC \*/*  assign NPC\_pc4 = NPC\_pc + 4;  always @(\*) begin  case (NPC\_op)  *// NPC\_PC4*  2'b00: NPC\_npc = NPC\_pc4;  *// NPC\_B*  2'b01: NPC\_npc = NPC\_br ? NPC\_pc + NPC\_offset : NPC\_pc4;  *// NPC\_JMP*  2'b10: NPC\_npc = NPC\_pc + NPC\_offset;  *// NPC\_JALR*  default: NPC\_npc = NPC\_aluin;  endcase  end  NPC关键：获取下一条指令的地址  PC模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | PC | in | rst | 1 | 复位 | | PC | in | clk | 1 | 时钟 | | PC | in | din | 32 | 获取npc的值 | | PC | out | pc | 32 | 当前PC地址 |   */\* PC \*/*  always @(posedge clk or posedge rst) begin  if (rst)  PC\_pc <= 32'hFFFF\_FFFC;  else  PC\_pc <= PC\_din;  end  PC关键：产生指令的地址  IROM模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | IROM | in | adr | 32 | 指令的地址 | | IROM | out | inst | 32 | 实际指令 |   IROM关键：获取实际指令  Controller模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | Controller | in | opcode | 7 | 获取指令的opcode | | Controller | in | funct3 | 3 | 获取指令的funct3 | | Controller | in | funct7 | 7 | 获取指令的funct7 | | Controller | out | sext\_op | 3 | 产生SEXT部件的控制信号 | | Controller | out | npc\_op | 2 | 产生NPC部件的控制信号 | | Controller | out | ram\_we | 1 | 控制DRAM部件的写入使能 | | Controller | out | alu\_op | 4 | 产生ALU部件的控制信号 | | Controller | out | alub\_sel | 1 | 产生ALU部件的输入选择信号 | | Controller | out | rf\_we | 1 | 控制RF部件的写入使能 | | Controller | out | rf\_wsel | 2 | 产生RF部件的输入选择信号 |   always @(\*) begin  case (CU\_opcode)  *// R-type*  7'b0110011:略  *// I-type*  7'b0010011: 略  *// lw*  7'b0000011: 略  *// jalr*  7'b1100111: 略  *// S-type*  7'b0100011: 略  *// B-type*  7'b1100011: 略  *// U-type*  7'b0110111: 略  *// J-type*  default: 略  endcase  end  Controller关键：产生各个控制信号  SEXT部件：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | SEXT | in | op | 3 | 获取控制信号 | | SEXT | in | din | 25 | 获取未扩展的立即数 | | SEXT | out | ext | 32 | 输出已扩展的立即数 |   */\* SEXT \*/*  always @(\*) begin  case (SEXT\_op)  *// EXT\_I*  3'b000: SEXT\_ext = $signed({SEXT\_din[24:13]});  *// EXT\_S*  3'b001: SEXT\_ext = $signed({SEXT\_din[24:18],SEXT\_din[4:0]});  *// EXT\_B*  3'b010: SEXT\_ext = $signed({SEXT\_din[24],SEXT\_din[0],  SEXT\_din[23:18],SEXT\_din[4:1], 1'b0});  *// EXT\_U*  3'b011: SEXT\_ext = {SEXT\_din[24:5], 12'b0};  *// EXT\_J*  default: SEXT\_ext = $signed({SEXT\_din[24], SEXT\_din[12:5],  SEXT\_din[13], SEXT\_din[23:14], 1'b0});  endcase  end  SEXT关键：产生立即数  RF模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | RF | in | clk | 1 | 时钟信号 | | RF | in | we | 1 | 获取写入使能信号 | | RF | in | rR1 | 5 | 获取输入信号1 | | RF | in | rR2 | 5 | 获取输入信号2 | | RF | in | wR | 5 | 获取待写入寄存器号 | | RF | in | wD | 32 | 获取待写入数 | | RF | out | rD1 | 32 | 输出读取的数据1 | | RF | out | rD2 | 32 | 输出读取的数据2 |   */\* RF \*/*  *// 同步写*  always @(posedge clk) begin  if (RF\_we) begin  for (i = 1; i < 32; i = i + 1) begin  if (RF\_wR == i)  RF\_reg[i] <= RF\_wD;  end  end  end  *// 异步读*  always @(\*) begin  for (ii = 1; ii < 32; ii = ii + 1) begin  if (RF\_rR1 == ii)  RF\_rD1 = RF\_reg[ii];  end  for (jj = 1; jj < 32; jj = jj + 1) begin  if (RF\_rR2 == jj)  RF\_rD2 = RF\_reg[jj];  end  if (RF\_rR1 == 32'b0) RF\_rD1 = 32'b0;  if (RF\_rR2 == 32'b0) RF\_rD2 = 32'b0;  end  RF关键：正确读写寄存器堆  ALU模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | ALU | in | op | 4 | 获取控制信号 | | ALU | in | A | 32 | 获取待操作数1 | | ALU | in | B | 32 | 获取待操作数2 | | ALU | out | C | 32 | 输出计算结果 | | ALU | out | f | 1 | 判断是否跳转 |   */\* ALU \*/*  always @(\*) begin  case (ALU\_op)  *// ALU\_ADD*  4'b0001: 略  *// ALU\_SUB*  4'b0010: 略  *// ALU\_AND*  4'b0011: 略  *// ALU\_OR*  4'b0100: 略  *// ALU\_XOR*  4'b0101: 略  *// ALU\_SLL*  4'b1000: 略  *// ALU\_SRL*  4'b1001: 略  ……  endcase  end  ALU关键：进行逻辑运算  DRAM模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | DRAM | in | clk | 1 | 时钟信号 | | DRAM | in | adr | 32 | 获取待操作地址 | | DRAM | in | we | 1 | 获取写入使能信号 | | DRAM | in | wdin | 32 | 获取写入数据 | | DRAM | out | rdo | 32 | 输出访存结果 |   DRAM关键：正确读写数据存储器 |

1.3 单周期CPU仿真及结果分析

|  |
| --- |
| 要求：包含逻辑运算、访存、分支跳转三类指令的仿真截图及波形分析；每类指令的截图和分析中，至少包含1条具体指令；截图需包含信号名和关键信号。 |
| 1. 5020ns时，置rst信号为1，各信号成功初始化，符合预期； 2. 5045ns时，进入第一个周期，设置输入为000037。pc值（NPC\_pc）为0x0000\_0000，符合预期； 3. 【访存指令lw】5065ns时，指令地址为0x0000\_0004，指令为0x0704a403，即`lw x8,70(x9)`，由图可知，RF写回的数据RF\_wD为0x0000\_0037，符合预期； 4. 【逻辑运算指令andi】5205ns时，指令地址为0x0000\_0020，指令为0x0ff4\_7613，即`andi x12,x8,0xff`，由图可知，RF写回的数据RF\_wD为0x0000\_0037，符合预期； 5. 【分支跳转指令beq】5325ns时，指令地址为0x0000\_0038，指令为0x255\_0a63，即`beq x10,x5,0x34`，而x10与x5均为0，应该跳转。由图可知，在5345ns时，指令地址为0x0000\_006c，成功跳转，符合预期。   综上所述，由仿真结果可知，本程序正确实现了单周期CPU的功能。 |

2 流水线CPU设计与实现

2.1 流水线CPU数据通路

|  |
| --- |
| 要求：贴出完整的流水线数据通路图，无需画出模块内的具体逻辑，但要标出模块的接口信号名、模块之间信号线的信号名和位宽，并用文字阐述各模块的功能。  此外，数据通路图应当能体现出流水线是如何划分的，并用文字阐述每个流水级具备什么功能、需要完成哪些操作。 |
| 注：数据通路图画了暂停的部分，实际代码中暂停功能尚未实现。  CPU分为5个流水级，分别为IF、ID、EX、MEM、WB。  IF流水级：取址，判断是否有跳转指令（是否需要暂停）。  PC模块：   * 产生新一条指令地址。   IROM模块：   * 根据PC模块的地址获取指令。   JMP\_detector模块：   * 判断是否有跳转指令。   ID流水级：译码，产生控制信号，生成立即数。  Controller模块：   * 根据指令产生控制信号，包括立即数扩展控制信号sext\_op、下一条指令控制信号npc\_op，数据存储器写入控制信号ram\_we，ALU控制信号alu\_op，ALU输入选择信号alub\_sel，寄存器堆写入控制信号rf\_we，寄存器堆写入选择信号rf\_wsel，PC选择信号PC\_sel，寄存器读信号reg\_RE1和reg\_RE2。   SEXT模块：   * 按照Controller模块产生的立即数扩展控制信号sext\_op将操作数扩展至32位。   RF模块：   * 存储寄存器的值。   EX流水级：逻辑运算，计算下一条指令的地址。  ALU模块：   * 根据Controller模块产生的ALU控制信号alu\_op，进行对应的运算。   NPC模块：   * 根据PC值、ALU运算值、偏移量、跳转标识、控制信号决定下一条指令的地址；   MEM流水级：读写数据存储器。  DRAM模块：   * 根据数据存储器写入控制信号ram\_we，读/写数据存储器。   WB流水级：写回寄存器。  数据冒险检测模块Data\_Hazard\_Detector用于检测数据冒险：   * 检测将要写入寄存器堆的数据是否存在冒险，若存在，输出暂停的周期数。   流水线寄存器IF\_ID、ID\_EX、EX\_MEM、MEM\_WB：   * 用于暂存各流水级之间的信息；输入流水线暂停信号，控制流水线的通断。 |

2.2 流水线CPU模块详细设计

|  |
| --- |
| 要求：以表格的形式列出所有与单周期不同的部件的接口信号、位宽、功能描述等，并结合图、表、核心代码等形象化工具和手段，详细描述这些部件的关键实现。此外，如果实现了冒险控制，必须结合数据通路图，详细说明数据冒险、控制冒险的解决方法。 |
| Controller模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | Controller | out | reg\_RE1 | 1 | RF数据1的读信号 | | Controller | out | reg\_RE2 | 1 | RF数据2的读信号 | | Controller | out | PC\_sel | 1 | PC选择信号 |   // 表格中为新增的  以下模块已设计，但未跑通Trace。  JMP\_detector模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | JMP\_detector | in | inst | 32 | 获取信号 | | JMP\_detector | out | JMP\_suspend | 1 | 是否需要暂停 |   Data\_Hazard\_Detector模块：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **部件** | in/out | 信号名 | 位宽 | 功能描述 | | DHD | in | rs1 | 5 | 写入数据1的寄存器号 | | DHD | in | rs2 | 5 | 写入数据2的寄存器号 | | DHD | in | JMP\_suspend | 1 | 是否因为跳转指令而停止 | | DHD | in | ID\_reg\_re1 | 1 | RF数据1的读信号（ID阶段） | | DHD | in | ID\_reg\_re2 | 1 | RF数据2的读信号（ID阶段） | | DHD | in | ID\_EX\_rf\_we | 1 | RF的写使能（EX阶段） | | DHD | in | ID\_EX\_rd | 5 | RF的写回寄存器号（EX阶段） | | DHD | in | EX\_MEM\_rd | 5 | RF的写回寄存器号（MEM阶段） | | DHD | in | EX\_MEM\_rf\_we | 1 | RF的写使能（MEM阶段） | | DHD | in | MEM\_WB\_rd | 5 | RF的写回寄存器号（WB阶段） | | DHD | in | MEM\_WB\_rf\_we | 1 | RF的写使能（WB阶段） |   解决数据冒险：检测写回的寄存器是否和要读取的寄存器存在冲突，若冲突，则暂停对应情况的周期数。 |

2.3 流水线CPU仿真及结果分析

|  |
| --- |
| 要求：包含控制冒险和数据冒险三种情形的仿真截图，以及波形分析。若仅实现了理想流水，则此处贴上理想流水的仿真截图及详细的波形分析。 |
| 代码部分数据冒险暂未实现，以下是理想流水线的仿真分析：   1. 25ns时，开始复位后的第一个时钟周期，PC的值（PC\_pc）为0x0000\_0000，符合预期 2. 65ns时，即开始复位后的第五个时钟周期，此时处于第一条指令的写回阶段，第一条指令为`addi x1, x0, 1`。图中，MEM\_WB\_pc为0x0000\_0000，符合预期；寄存器写使能（MEM\_WB\_CU\_rf\_we）为1，符合预期；写回的寄存器（MEM\_WB\_rd）为1，写回的数据（MEM\_WB\_rf\_wD）为1，符合预期； 3. 75ns时，即开始复位后的第六个时钟周期，此时处于第二条指令的写回阶段，第二条指令为`addi x2, x0, 2`图中，MEM\_WB\_pc为0x0000\_0004，符合预期；寄存器写使能（MEM\_WB\_CU\_rf\_we）为2，符合预期；写回的寄存器（MEM\_WB\_rd）为2，写回的数据（MEM\_WB\_rf\_wD）为1，符合预期； 4. 125ns时，此时处于第七条指令的写回阶段，第七条指令为` sub x7, x2, x1`。图中，MEM\_WB\_pc为0x0000\_0018，符合预期；寄存器写使能（MEM\_WB\_CU\_rf\_we）为1，符合预期；写回的寄存器（MEM\_WB\_rd）为7，写回的数据（MEM\_WB\_rf\_wD）为1，符合预期；   综上所述，由仿真结果可知，本程序正确实现了理想流水线CPU的功能。  附：理想汇编指令：  addi x1, x0, 1  addi x2, x0, 2  ori x3, x0, 3  addi x4, x0, 4  addi x5, x1, 1  add x6, x1, x2  sub x7, x2, x1 |

3 设计过程中遇到的问题及解决方法

|  |
| --- |
| 要求：包括设计过程中遇到的有价值的错误，或测试过程中遇到的有价值的问题。所谓有价值，指的是解决该错误或问题后，能够学到新的知识和技巧，或加深对已有知识的理解和运用。 |
| 汇编程序：  问题：不知道存储指令与外设如何交互。  解决：运行示例代码，知道不同外设使用不同的基地址。  单周期：  问题：BLT和BGE指令两数大小比较时负数比正数大。  解决：使用有符号数比较： if ($signed(ALU\_A) < $signed(ALU\_B))  问题：不知道数据存储器如何使用。  解决：阅读总线桥代码（Bridge.v），了解具体内容。  问题：IROM导入机器码存在问题。  解决：删除中文注释。  问题：Trace第一条指令报错。  解决：PC从FFFF\_FFFC开始。  问题：Trace时经常在寄存器堆处报错。  解决：不给x0赋值，取x0值时直接取出0。  问题：上板时无复位键。  解决：将一个button设置为复位键。  问题：七段数码管显示出错。  解决：多次修改刷新频率，使其合理。  流水线：  问题：将单周期代码改为流水线时无从下手。  解决：先修改数据通路图  问题：NPC不能放在IF阶段。  解决：将NPC放在EX阶段。 |

4 总结

|  |
| --- |
| 要求：谈谈学完本课程后的个人收获以及对本课程的建议和意见。请在认真总结和思考后填写总结。 |
| 个人收获：  通过夏季学期的课程，我完成了RISC-V汇编语言的程序设计，并用verilog完成了单周期和理想流水线的CPU设计。本次课程让我收获颇丰，我熟悉了RARS、Logisim等汇编和模拟仿真工具的使用；熟练掌握了RISC-V汇编语言，熟悉并理解了RISC-V指令系统；了解程序在单周期RISC-V CPU搭建的SoC中的运行过程；理解了单周期CPU工作过程；理解指令存储器和数据存储器的哈佛结构存储；熟悉miniRV指令集；掌握单周期CPU设计与实现方法；理解流水线CPU工作过程；理解流水线冲突产生的原因及其解决方法；掌握流水线CPU设计与实现方法。  建议：  可以多整理一些易错点，列成共享文档，防止在实验过程中被一些小问题卡住。  总结：  在实验一中，通过实现RISCV指令的汇编程序，我理解了指令集和汇编语言的运作方式。这帮助我更好地理解计算机程序的执行过程。实验二和实验三中，通过使用Verilog完成单周期和流水线的CPU设计，我提高了自己的CPU设计能力。在完成CPU设计的过程中，我也学会了问题解决和调试的技巧。硬件设计中常常会遇到各种问题，如逻辑错误、时序问题等，通过仿真等方法不断调试和排查，我学会了如何更快地找到问题并解决它们。 |