

（深圳）

# 课程报告

开课学期： 2024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： 8班

学生学号： 220110813

学生姓名： 罗中横

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） |
| R型指令8条   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | add | sub | and | or | xor | sll | srl | sra |   I型指令9条   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | addi | andi | ori | xori | slli | srli | srai | lw | jalr |   B型指令4条   |  |  |  |  | | --- | --- | --- | --- | | beq | bne | blt | bge |   U型指令1条  lui  S型指令1条  sw  J型指令1条  jal  时钟频率：单周期 CPU 频率为 25MHZ，流水线 CPU 频率为 100MH |
| 设计的主要特色（除基本要求以外的设计） |
| 1. 在alu计算单元中根据控制单元的alu\_op信号，能对两个操作数进行bge，bne，beq，blt四种大小判断，并通过f输出是否跳转成立信息。 2. 在多周期的写回数据选择实现，pc+4和imm分别在IF和ID阶段就获得了，但仍要等待alu和dram的结果进行四者的筛选，这样大大增加了流水线寄存器存储数据，因此我增加了一个多路选择器，让pc+4，imm，alu的结果先在EX阶段进行第一次的选择，将结果传递给EX/MEN流水线，到MEN阶段获得dram数据后，再进行一步的选择，这样就减少了两个32位寄存器存储空间。 |
| 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**。 |
| 模块说明：   1. 取指模块     1. PC：存储当前指令地址的 32 位寄存器，指令执行时负责更新当前 PC 值（时序逻辑）     2. NPC：得到下一条指令的 PC 值（组合逻辑）     3. IROM：指令存储器，根据 PC 值得到对应的指令     4. MUX：多路选择器，选择 NPC.pc 的值（组合逻辑）：pc和alu\_c。 2. 译码模块    1. RF：寄存器堆，可根据寄存器编号读出源寄存器数据或将数据写入目标寄存器（异步读（组合逻辑）、同步写（时序逻辑））     2. SEXT：立即数扩展单元，根据指令中立即数的格式进行扩展（组合逻辑）    3. MUX：多路选择器，选择写回寄存器堆的数据，来自alu\_c，pc4，ext，dram。 3. 执行模块     1. ALU：运算器，根据控制信号对输入数据执行加、减、与、或、异或、移位、比较 等运算（组合逻辑）     2. MUX：多路选择器，选择输入 ALU.A 端口的数据    3. MUX：多路选择器，选择输入 ALU.B 端口的数据 4. 存储模块     1. DRAM：数据存储器，可读可写，用于存储数据 5. 写回模块：将数据写入寄存器堆中 6. 控制模块     1. control：控制器，根据指令中 opcode、func3、func7 的值得到各个元件的控制信号 |

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**。 |
| 1. 取指模块：（IF）      1. PC单元：   指令正常执行时，时钟上升沿来临，pc自动更新为din的值；  复位信号出现时，pc更新为0   |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | clk | input | 1 | 时钟信号 | | rst | input | 1 | 复位信号 | | din | input | 32 | 下一条指令地址 | | pc | output | 32 | 当前指令地址 |  1. NPC单元   NPC的两个输出对应输出当前指令地址加四pc4的值和下一条指令地址npc的值。  NPC输出下条指令的地址由npc\_op决定，分别对应：   1. pc4：顺序执行 2. pc+offset：无条件跳转jal 3. rs1+offset：无条件跳转jalr 4. b？pc+offset：条件跳转B型指令  |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | op | input | 1 | NPC选择控制信号控制npc的结果 | | offset | input | 32 | 跳转偏移量 | | br | input | 1 | 是否发生跳转信号 | | pc | input | 32 | 当前指令地址 | | pc4 | output | 32 | 当前指令地址加四 | | npc | output | 32 | 下一条指令地址 | |
| 1. IROM   指令存储器，根据 PC 值对应地址得到对应的指令   |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | adr | input | 32 | 输入地址 | | inst | output | 32 | 输出指令 |   2. 译码模块：（ID）     1. RF单元    1. 对 32 位指令 inst 进行解析，得到读地址 rR1(inst [19:15])、rR2(inst [24:20]) 和写 地址 wR(inst[11:7])。寄存器的读没有时钟限制，因此可通过组合逻辑读出寄存器中数据rD1 = regfile[rR1]、rD2 = regfile[rR2]。寄存器的写与时钟上升沿同步，因此通过 时序逻辑实现：当 CPU 复位时，各寄存器中数据均置为0；如果写使能 we为1且写入寄存器号不为 0，将写寄存器数据 wD 写入目标寄存器；如果写入寄存器号为0， 则 x0 寄存器内数据始终为0。    2. 其中写寄存器数据 wD 由四选一多路选择器选择，其写入数据包括：运算器结果 ALU.C、存储器读出结果 DRAM.rdo、pc + 4、扩展后的立即数 SEXT.ext，并由 rf\_wsel 信号决定写入数据。    3. 其中RF后的是两个MUX，分别选择rD1和PC对执行单元的A操作数，rD2和imm对执行单元B的操作数。  |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | clk | input | 1 | 时钟信号 | | rst | input | 1 | 复位信号 | | rR1 | input | 5 | 读寄存器1 | | rR2 | input | 5 | 读寄存器2 | | wR | input | 5 | 写入寄存器 | | pc4 | input | 32 | pc4 | | alu\_c | input | 32 | alu计算结果 | | dram | input | 32 | dram的读取结果 | | opA | output | 32 | alu输入操作数A | | opB | output | 32 | alu输入操作B | | wD | output | 32 | 写入寄存器数据 |  1. SEXT单元    1. 对输入的inst[31:7]和选择的sext\_op进行拆解并输出为完整的立即数。  |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | din | input | 25 | 输入的数据 | | op | input | 3 | sext组合输出op | | ext | output | 32 | 根据op特定组合的立即数结果 |  1. 执行模块（EX）      1. ALU单元    1. 译码器负责接收控制单元发出的alu\_op信号，经过处理后对alu各个部分进行选择控制    2. 求补：由译码器控制是否执行求补，即减法运算    3. 逻辑运算单元执行译码器发出的逻辑运算    4. 桶型移位器执行译码器提供的移位运算    5. 最后在MUX中选择加法器、逻辑运算或移位器的结果    6. 在判断是否跳转f信号，通过加法器实现两个操作数的相减，译码器执行比较运算后输出f的结果。   执行单元   |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | opA | input | 32 | 操作数A | | opB | input | 32 | 操作数B | | alu\_op | input | 4 | 运算单元运算符 | | f | output | 1 | 是否跳转标识符 | | c | output | 32 | 运算结果 |  1. 访存模块      |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | clk | input | 1 | 时钟信号 | | we | input | 1 | 写使能信号 | | d | input | 32 | 写入数据信号 | | adr | input | 32 | 读地址信号 | | spo | output | 32 | 读数据信号 |  1. 控制模块 |
| 对32位inst拆解位opcode、fun3、fun7进行解析，并输出各个模块单元控制信号。   |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | inst | input | 32 | 指令信息 | | pc\_sel | output | 1 | 控制pc跳转信号 | | npc\_op | output | 2 | 控制npc下一条指令选择信号 | | rf\_wsel | output | 2 | 控制写入寄存器数据选择 | | rf\_we | output | 1 | 控制写入寄存器使能 | | op\_B\_sel | output | 1 | 控制执行阶段操作数B的选择 | | op\_A\_sel | output | 1 | 控制执行阶段操作数A的选择 | | sext\_op | output | 2 | 控制立即数扩展方式 | | dram\_we | output | 1 | 写入dram写使能信号 | | alu\_op | output | 4 | 执行阶段运算符 | |

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。 |
| I型指令：  上图是addi指令的波形，在pc=5c时，取出指令inst=00500193，识别opcode为I型指令；  故在NPC中设置npc\_op为0，将npc确定为pc4；  SEXT收到来自control的信号，输出I型指令的立即数扩展结果5  在alu中将分别设置操作数：alu\_A\_sel=rR1，alu\_B\_sel=imm，执行zero+imm的操作，并将结果写入wR=3，即regfile[3]的寄存器中；  同时，在RF寄存器堆的写入信号设置为可写入信号，写入数据信号为00，即写入alu计算结果。    B型指令：  上图是addi指令的波形，在pc=5c时，取出指令inst=00500193，识别opcode为I型指令；  故在NPC中根据alu计算得到标志位f==0，设置npc\_op为0，将npc确定为pc4；  SEXT收到control的信号sext\_op==B，根据B型指令要求扩展立即数，若执行跳转则将进行pc+imm的跳转。  在alu中将分别设置操作数：alu\_A\_sel=rR1，alu\_B\_sel=rR2，alu\_op==9，对应的运算为bne，根据两个运算数可以得知，两数相同，输出f==0，不执行跳转操作。  同时，在RF寄存器堆的写入信号设置为不可写入信号，因此不会影响寄存器堆的数据。    JALR指令:  pc==44时，识别opcode为jalr指令  此时npc\_op选择11，即从alu\_c中获取数值作为npc的值；  在SEXT立即数扩展单元中，要根据JALR的I型指令进行扩展；  此时alu需要计算rR1与立即数的运算，将op\_A\_sel==rR1，op\_B\_sel==imm，op==0，即加法运算。执行结果为4c，准备放入NPC中作为下一个指令地址；  RF负责将pc+4的值写入目的寄存器0D中，此时的we为1，写入数据pc+4为48；    LW  指令：pc==28时，inst==0040A703，此时的opcode==lw；  npc\_op设置为0，即pc+4  SEXT中收到control的信号，将执行I型立即数扩展，得到imm==4  alu确定运算数分别为opA==rR1，opB==imm，alu\_op==0，即加法操作，将结果连接到DRAM的写入地址（默认偏移两位）即2004->801  RF选择DRAM输出，即rf\_wesl==11，此时将DRAM地址为801的数据FF00FF00写入0  E寄存器堆中。 |

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**。 |
| pip_cycle (1)   1. 取指模块：    1. 根据当前 PC 值从指令存储器中读取一条指令，并更新下一条指令的 PC 值 2. 译码模块：    1. 分解指令，产生对应的控制信号    2. 得到对应的立即数，同时获得立即数和pc的加和结果    3. 从寄存器和立即数中取出对应将要运算的数据 3. 执行模块：    1. 进行对应数据运算    2. 对alu和pc+imm的结果进行选择，输出npc\_change的结果    3. 第一次进行rf\_wsel选择，选择alu\_c，pc+4，pc+imm的结果，将结果作为ex\_temp等待在存储模块进行最后一次选择。 4. 存储模块：    1. 在DRAM中进行读或写数据    2. MUX进行最后一次rf\_wsel选择，选择ex\_temp或DRAM读取数据，将作为wb\_wD等待写入寄存器堆中。 5. 写回模块：从存储模块获得的rf\_we和men\_wD传到RF中，当时钟上升沿到来将进行写回操作。 |

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，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。 |
| 1. 取指模块   修改了npc的实现方式：具体结合执行模块。    单周期：    多周期：      NPC模块：  将b和offset接口改为npc\_change接口，在ex阶段完成分支跳转的计算并返回给NPC，若发生跳转则npc选择npc\_change结果，否则将以pc+4作为npc的值。  NPC\_control模块：通过获得pc\_imm（jal），alu\_c（B型指令和jalr）的计算结果，根据control模块译码获得的npc\_sel决定当前指令类型并将对应的值赋予npc\_change和flag。   |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | npc\_sel | input | 2 | 对跳转指令的选取（j型or B型or jalr） | | pc\_imm | input | 32 | 在SEXT中计算得到的pc+imm的值（jal） | | f | input | 1 | 在alu运算单元得B型指令是否跳转的标志 | | alu\_c | input | 32 | alu计算结果 | | flag | output | 1 | 控制NPC跳转的标志位信号 | | npc\_change | output | 32 | NPC即将跳转的位置 |        1. 译码模块     control元件：  删去了op\_A\_sel的控制信号，转而在pc\_imm中进行pc+imm的计算。  RF元件：  删去了wD的多路选择器，转而在EX和MEN阶段对wD的内容进行选择控制。  SEXT模块：  在SEXT后加入一个pc+imm的加法器，在执行jal指令时，无需再通过alu，直接可以完成pc+imm的结果运算，在npc\_control中进行npc的选择控制。这种方式有益于在id阶段完成无条件跳转指令的跳转结果运算，在时钟下一周期来临前完成下一指令地址计算。     |  |  |  |  | | --- | --- | --- | --- | | Name | I/O | Width | description | | imm | input | 32 | 扩展后立即数 | | pc | input | 32 | 当前pc | | pc\_imm | input | 32 | pc+imm | |

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。 |
| 无冒险指令：       1. 在clk\_rst结束后IF已经取得第一个指令addi，当时钟上升沿到来，IF的指令将传递给ID中，pc，pc+4，inst的值随之带入到ID阶段。对指令进行分解并生成指令对应控制信号。 2. 由于是addi指令，运算器将进行寄存器和立即数的运算，这时候的op\_B\_sel将从默认的信号1转为0信号，alu\_op仍保持为默认的加法信号。      1. 在15ns时id中的控制信号以及数据信号被传入EX阶段，这时的opA是zero即0，opB是imm即1，进行了加法运算后得到的结果是1，即输出的结果是1。 2. 在EX阶段同时进行了npc的选择操作，由于是addi指令，无跳转，因此在npc\_change的输出为Z，flag为0，在npc中体现即选择pc+4作为npc的值。 3. 在EX阶段进行了写入寄存器的第一步筛选操作，wsel将选择alu作为第一个筛选结果放入ex\_temp中，将继续传递给MEN阶段进行最后一步筛选。即此时的ex\_temp的值为alu的值1。 4. EX阶段对应20ns的时间，在此时ID阶段的数据传入到EX中，20ns时ex\_we的值被拉高为1。      1. 接下来在25ns时，进入MEN阶段，MEN主要进行两个操作，即读写DRAM和选择写入寄存器数据。可见MEN阶段的rf\_we被拉高，最后进行写入寄存器数据的选择。此时的wsel默认为0，即选择从非寄存器数据，即写入结果为add计算结果1，在25ns时，RF的写入数据和写入使能信号同时被拉高，寄存器a1被写入。 |

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。 |
| 1. 单周期实现无条件跳转和条件跳转都在EX阶段进行计算，在无条件跳转时要在第三个阶段才能计算出npc的值，因此在多周期中，将pc+imm的计算放在了ID阶段，获得了control的控制信号后直接得到下一条指令的地址。 2. 在连线的过程中多次出现了未定义变量被使用的情况，在学习波形分析后充分发现了问题，认识到了代码编写规范的重要性。 |

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。 |
| 1. 个人收获：充分了解了CPU的内部结构以及各个部件的电路实现方式。以更为底层的视角了解了单周期流水线的设计方式。 2. 学会了使用画图来方便硬件设计，让设计更加系统，同时方便发现错误。 3. 强化了使用Verilog编程，了解了使用IP核来进行硬件设计。 |