

（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： T2612

学生班级： 5班

学生学号： 210110503

学生姓名： 陈睿玮

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） |
| 单周期CPU和流水线CPU都实现了全部37条指令。  单周期CPU频率为25MHz，流水线CPU频率为75MHz。 |
| 设计的主要特色（除基本要求以外的设计） |
| 实现了包括选做指令在内的全部37条指令。  流水线CPU在处理非LOAD类型指令产生的数据冒险时，使用前递机制，高效解决了数据冒险。在处理LOAD类型指令产生的ID/EX数据冒险时，将IF/ID、ID/EX寄存器暂停一个时钟周期，并在MEM阶段将数据前递回来。在处理控制冒险时，使用暂停法，将IF/ID寄存器暂停3个时钟周期，然后根据ALU传回的结果进行是否跳转的操作。 |
| 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PC信号和跳转信号产生下一条PC信号。   * **PC模块**   接受NPC模块的npc信号，在时钟上升沿更新PC信号。   * **IROM模块**   将输入的PC信号作为地址，从ROM中读出相应的指令。   * **Controller模块**   根据指令的opcode，funct3，funct7信号，产生一系列控制信号。   * **SEXT模块**   根据sext\_op信号，将指令对应部分的立即数扩展为32位。   * **RF模块**   读出指令中对应寄存器的数，在上升时钟沿向寄存器堆写入写回数据。其中，读操作为组合逻辑，写操作为时序逻辑。   * **ALU模块**   将两个操作数进行算术运算、逻辑运算、移位运算或比较操作，产生对应的结果信号。   * **DRAM模块**   存储数据，根据地址和读写操作信号进行读取或写入操作，针对不同的LOAD类型指令和S类型指令需要对数据进行不同的加工。其中，读操作为组合逻辑，写操作为时序逻辑。 |

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 | | | | | 信号 | 位宽 | I/O | 功能描述 | | pc | 32 | I | 接受PC.pc信号，结合跳转信号，用于产生下一条pc信号 | | op | 2 | I | NPC模块的控制信号，用于控制NPC模块产生npc | | offset | 32 | I | 接受SEXT.ext信号，是基于PC.pc信号跳转的偏移量 | | br | 1 | I | 接受ALU.comp信号，用于判断B型指令是否发生跳转 | | next | 32 | I | 接受ALU.c信号，用于JALR指令这种直接给定npc的跳转指令 | | npc | 32 | O | 产生下一条pc信号 | | pc4 | 32 | O | 产生pc+4信号 |   **NPC模块**  **PC模块**   |  |  |  |  | | --- | --- | --- | --- | | PC | | | | | 信号 | 位宽 | I/O | 功能描述 | | clk | 1 | I | 时钟信号 | | rst | 1 | I | 复位信号 | | din | 32 | I | 接受NPC.npc信号 | | pc | 32 | O | 根据din信号在每个上升时钟沿更新pc信号 |   **IROM模块**   |  |  |  |  | | --- | --- | --- | --- | | IROM | | | | | 信号 | 位宽 | I/O | 功能描述 | | addr | 32 | I | 接受PC.pc信号作为指令地址 | | inst | 32 | O | 从对应指令地址中取出指令 |   **SEXT模块**   |  |  |  |  | | --- | --- | --- | --- | | SEXT | | | | | 信号 | 位宽 | I/O | 功能描述 | | op | 3 | I | SEXT模块的控制信号，用于立即数扩展 | | din | 25 | I | 接受IROM.inst中立即数的信号 | | ext | 32 | O | 根据sext\_op信号对立即数进行扩展 |   **RF模块**   |  |  |  |  | | --- | --- | --- | --- | | RF | | | | | 信号 | 位宽 | I/O | 功能描述 | | clk | 1 | I | 时钟信号 | | rst | 1 | I | 复位信号，用于清空寄存器堆 | | rR1 | 5 | I | 接受IROM.inst[19:15]，用于指定指令需要读取的寄存器号 | | rR2 | 5 | I | 接受IROM.inst[24:20]，用于指定指令需要读取的寄存器号 | | wR | 5 | I | 接受IROM.inst[11:7]，用于指定指令需要写入的寄存器号 | | we | 1 | I | 写入使能信号 | | wD | 32 | I | 接受由多路选择器选择信号作为写入数据 | | rD1 | 32 | O | 从寄存器堆中对应rR1的寄存器读出的数据 | | rD2 | 32 | O | 从寄存器堆中对应rR2的寄存器读出的数据 |   **ALU模块**   |  |  |  |  | | --- | --- | --- | --- | | ALU | | | | | 信号 | 位宽 | I/O | 功能描述 | | op | 4 | I | ALU模块的控制信号，用于控制ALU对操作数进行运算 | | a | 32 | I | 接受多路选择器选择的PC.pc或RF.rD1信号，作为第一个操作数 | | b | 32 | I | 接受多路选择器选择的SEXT.ext或RF.rD2信号，作为第二个操作数 | | c | 32 | O | 根据控制信号，将a和b进行运算后得到的结果 | | comp | 1 | O | 根据控制信号，将a和b进行比较操作后得到的结果 |   **Controller模块**   |  |  |  |  | | --- | --- | --- | --- | | Controller | | | | | 信号 | 位宽 | I/O | 功能描述 | | opcode | 7 | I | 接受IROM.inst[6:0]，作为指令的opcode | | funct3 | 3 | I | 接受IROM.inst[14:12]，作为指令的funct3 | | funct7 | 7 | I | 接受IROM.inst[31:25]，作为指令的funct7 | | sext\_op | 3 | O | SEXT模块的控制信号 | | npc\_op | 2 | O | NPC模块的控制信号 | | ram\_we | 1 | O | DRAM模块的写使能信号 | | ram\_r\_op | 3 | O | DRAM模块的读操作控制信号，用于针对不同LOAD类型的指令读出的数据进行加工 | | ram\_w\_op | 2 | O | DRAM模块的写操作控制信号，用于针对不同S类型的指令对写入的数据进行加工 | | alu\_op | 4 | O | ALU模块的控制信号 | | alu\_asel | 1 | O | ALU模块a信号的选择信号 | | alu\_bsel | 1 | O | ALU模块b信号的选择信号 | | rf\_we | 1 | O | RF模块的写使能信号 | | rf\_wsel | 2 | O | RF模块的wD信号的选择信号 |   **DRAM模块**   |  |  |  |  | | --- | --- | --- | --- | | DRAM | | | | | 信号 | 位宽 | I/O | 功能描述 | | r\_op | 3 | I | DRAM模块的读操作控制信号，用于针对不同LOAD类型的指令读出的数据进行加工 | | w\_op | 2 | I | DRAM模块的写操作控制信号，用于针对不同S类型的指令对写入的数据进行加工 | | we | 1 | I | DRAM模块的写使能信号 | | addr | 32 | I | 接受ALU.c，作为读地址或写地址 | | wdin | 32 | I | 接受RF.rD2，作为写入的数据 | | rdo | 32 | O | 从addr地址处读出的数据 | |

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。 |
| * **逻辑运算（AND）**   在260ps时，指令为and x14,x1,x2，wb\_pc为14，rf\_rD1为FF00FF00，rf\_rD2为0F0F0F0F。alu\_asel为0，在rf\_rD1和pc之间选择了rf\_rD1作为ALU的a操作数。alu\_bsel为0，在rf\_rD2和SEXT.ext之间选择了rf\_rD2作为b操作数。alu\_op为0010，说明alu进行and运算。alu\_c为0F000F00，结果正确。wb\_value为0F000F00，wb\_reg为14，wb\_ena为1，写回结果也正确。   * **访存**   在220ps时，指令为lui x1,0x2，wb\_value为00002000，wb\_reg为1，wb\_ena为1，说明向x1寄存器写入值正确。  在230ps时，指令为addi x1,x1,0，写入相关的信号保持不变。  在240ps时，指令为lw x14,0(x1)，Bus\_addr为00002000，与reg[x1]+0的值一致，Bus\_rdata为00FF00FF，与wb\_value一致，wb\_reg为14，wb\_ena为1，说明向x14寄存器写入了正确的数据。   * **分支跳转**   在1030ps时，指令为addi x4,x4,1，写入数据说明向x4寄存器写入了1。  在1040ps时，指令为addi x5,x0,2，写入数据说明向x5寄存器写入了2。  在1050ps时，指令为bne x4,x5,12c，为伪指令，这时alu\_comp为1，说明对应指令的比较结果为真，即x4不等于x5，在NPC中npc\_op为10，说明执行分支判断和跳转操作，br为1，说明需要执行分支跳转，offset为-36，说明需要跳转到pc+offset处，npc值为0000012c，说明下一条pc值更新正确。 |

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。 |
| * **HazardDetection模块**   接收不同流水级的信号，进行数据冒险和控制冒险的判断，向不同段寄存器发出暂停信号，生成前递数据。   * **IF流水级**   接受MEM流水级的信号实现跳转，产生pc信号，根据对应pc信号取出相应指令。   * **ID流水级**   根据指令生成对应的控制信号，生成扩展后的立即数，从RF中读出对应寄存器的数据，同时接收前递数据。   * **EX流水级**   根据传递来的数据进行ALU数据的选择，根据控制信号进行相应的运算。   * **MEM流水级**   从EX流水级中保持跳转信号，发送给IF流水级。从DRAM中读数据或写数据。   * **WB流水级**   选择出向RF的写入数据，并向RF对应寄存器写入。 |

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。 |
| |  |  |  |  | | --- | --- | --- | --- | | NPC | | | | | 信号 | 位宽 | I/O | 功能描述 | | pc | 32 | I | 接受PC.pc信号，结合跳转信号，用于产生下一条pc信号 | | op | 2 | I | NPC模块的控制信号，用于控制NPC模块产生npc | | offset | 32 | I | 接受EX/MEM.ext信号，是基于EX/MEM.pc信号跳转的偏移量 | | br | 1 | I | 接受EX/MEM.comp信号，用于判断B型指令是否发生跳转 | | next | 32 | I | 接受EX/MEM.c信号，用于JALR指令这种直接给定npc的跳转指令 | | base\_pc | 32 | I | 接受EX/MEM.pc信号，是跳转指令的pc | | pc\_update | 1 | I | 是否需要进行跳转指令的判断和更新npc，若为0，则npc=pc+4 | | npc | 32 | O | 产生下一条pc信号 | | pc4 | 32 | O | 产生pc+4信号 |   **NPC模块**  **Controller模块**   |  |  |  |  | | --- | --- | --- | --- | | Controller | | | | | 信号 | 位宽 | I/O | 功能描述 | | ... | ... | ... | ... | | rf\_rf1 | 1 | O | 判断是否需要读取RF模块rR1数据的信号 | | rf\_rf2 | 1 | O | 判断是否需要读取RF模块rR2数据的信号 |   **HazardDetection模块**   |  |  |  |  | | --- | --- | --- | --- | | HazardDetection | | | | | 信号 | 位宽 | I/O | 功能描述 | | clk | 1 | I | 时钟信号 | | rst | 1 | I | 复位信号 | | id\_opcode | 7 | I | ID级指令的opcode | | id\_rR1 | 5 | I | ID级指令的rR1 | | id\_rR2 | 5 | I | ID级指令的rR2 | | id\_rf\_rf1 | 1 | I | ID级指令rR1是否需要读取的标志位 | | id\_rf\_rf2 | 1 | I | ID级指令rR2是否需要读取的标志位 | | ex\_opcode | 7 | I | EX级指令的opcode | | ex\_wR | 5 | I | EX级指令的wR | | ex\_rf\_we | 1 | I | EX级指令RF的写使能信号 | | ex\_rf\_wsel | 2 | I | EX级指令的向RF写数据的选择信号 | | ex\_c | 32 | I | EX级指令的ALU.c | | ex\_ext | 32 | I | EX级指令的的SEXT.ext | | ex\_pc4 | 32 | I | EX级指令的的NPC.pc4 | | mem\_wR | 5 | I | MEM级指令的wR | | mem\_rf\_we | 1 | I | MEM级指令RF的写使能信号 | | mem\_rf\_wsel | 2 | I | MEM级指令的向RF写数据的选择信号 | | mem\_c | 32 | I | MEM级指令的ALU.c | | mem\_rdo | 32 | I | MEM级指令的的DRAM.rdo | | mem\_ext | 32 | I | MEM级指令的的SEXT.ext | | mem\_pc4 | 32 | I | MEM级指令的的NPC.pc4 | | wb\_wR | 5 | I | WB级指令的wR | | wb\_rf\_we | 1 | I | WB级指令RF的写使能信号 | | wb\_rf\_wsel | 2 | I | WB级指令的向RF写数据的选择信号 | | wb\_c | 32 | I | WB级指令的ALU.c | | wb\_rdo | 32 | I | WB级指令的的DRAM.rdo | | wb\_ext | 32 | I | WB级指令的的SEXT.ext | | wb\_pc4 | 32 | I | WB级指令的的NPC.pc4 | | pc\_update | 1 | O | NPC模块是否需要进行跳转指令的判断和更新npc，若为0，则npc=pc+4 | | rs1\_id\_data\_hazard | 1 | O | rR1寄存器发生了除LOAD型指令的数据冒险，需要将rD1替换为前递数据 | | rs2\_id\_data\_hazard | 1 | O | rR2寄存器发生了除LOAD型指令的数据冒险，需要将rD2替换为前递数据 | | forward\_rD1 | 32 | O | rD1的前递数据 | | forward\_rD2 | 32 | O | rD1的前递数据 | | if\_id\_pipeline\_stop | 1 | O | IF/ID段寄存器暂停信号 | | id\_ex\_pipeline\_stop | 1 | O | ID/EX段寄存器暂停信号 | | ex\_mem\_pipeline\_stop | 1 | O | EX/MEM段寄存器暂停信号 | | mem\_wb\_pipeline\_stop | 1 | O | MEM/WB段寄存器暂停信号 |   **数据冒险的解决方法：**  除了LOAD类型指令引起的ID\_EX的数据冒险，对其他类型的数据冒险，采用前递机制，将所需要前递的数据和控制信号先传给HazardDetection模块，然后由该模块负责生成前递数据和前递选择信号用于替换rD1和rD2。  核心代码如下  对于LOAD类型指令引起的ID\_EX类型的数据冒险，对IF/ID和ID/EX模块都暂停一个时钟周期，然后在下一个时钟周期，冒险就变成了ID\_MEM类型的数据冒险，这时LOAD类型的指令也能在MEM流水级中读出数据，可以使用前递机制把数据前递过去。  **控制冒险的解决方法：**  HazardDetection模块接收ID流水级传来的指令信息，对于所有B型指令、JAL指令和JALR指令，HazardDetection将其判断为控制冒险，将IF/ID段寄存器暂停3个时钟周期，这样控制指令便能通过这3个时钟周期实现EX模块执行完毕，将EX模块执行的结果传给IF模块，和更新pc这三个操作，实现了指令跳转。 |

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。 |
| **数据冒险：**  观察上述汇编代码，我们发现38和3c发生了ID\_EX类型的数据冒险，34和3c发生了ID\_MEM类型的数据冒险。  在450ps时，ID流水级正在执行3c，EX流水级正在执行38，MEM流水级正在执行34，此时同时发生了ID\_EX数据冒险和ID\_MEM数据冒险，于是rs1\_id\_mem\_hazard为1且rs2\_id\_ex\_hazard为1，HazardDetection需要产生前递数据forward\_rD1和forward\_rD2，发现数据正确，于是3c的结果正确。  **LOAD类型指令产生的ID\_EX数据冒险：**  在1320ps时，mem\_pc为130，ex\_pc为134，id\_pc为138，id\_ex\_ld\_hazard为1，说明发生了LOAD类型指令引起的ID\_EX数据冒险，此时if\_id\_pipeline\_stop和id\_ex\_pipeline\_stop为1，暂停IF/ID和ID/EX段寄存器一个时钟周期。  在1330ps时，rs1\_id\_mem\_hazard为1，说明发生了ID\_MEM数据冒险，此时将数据前递回去，看到forward\_rD1正确，于是138指令执行的结果正确。  **控制冒险：**  在270ps时，id\_pc为c，此时id\_control\_hazard为1，因此if\_id\_pipeline\_stop为1，在下一个时钟沿开始时暂停3个时钟周期，第一个周期用于ID流水级，第二个周期用于EX流水级得出跳转的结果和更新npc，第三个周期用于pc的更新。  在290ps时，即第二个周期，update\_pc为1，说明npc需要更新，npc更新成功为1c。  在300pc时，pc成功更新为1c。 |

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。 |
| 在CPU下板时，我发现板子的数码管不停地闪烁，起初我以为是因为自己写的数码管刷新频率有问题，但是后面检查过后发现没有问题。后来经过反复的思考和检查，发现是由于我对总线的理解不到位。数码管显示的数据，不应该是直接接到总线上，这样的话，总线的数据变化了，数码管显示也会发生变化，举个例子，CPU向RAM写入数据时需要占用总线，而此时数码管要显示的数据是直接接到总线上的，这时数码管显示的数据就会变成CPU向RAM写入的数据，而由于总线是经常被占用的，总线上的数据也经常变化，因此数码管也就不停闪烁。  解决方法是，将数码管连上总线的写使能信号，这样，只有总线对数码管进行写入时，数码管要显示的数据才需要进行更新。 |

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：**  学完本课程，我不仅亲自动手实现了单周期的CPU，也更进一步，实现了流水线的CPU，并实现了前递机制和暂停机制，对CPU的架构和流水线有了更深入的认识，对计算机的底层有了进一步的认识。  在上计组课时，我听着老师上课讲的CPU数据通路和流水线，如同听天书。老师课上讲数据通路从零开始讲起，从最基本的add指令讲起，遇到一个新指令就加一个硬件，最后即使老师只讲了一小部分指令，老师的数据通路图也看起来复杂无比，控制信号也非常多。在老师讲完数据通路图后，我仍然不是很清楚为什么要有这些控制信号和为什么数据通路图上要有这个硬件，这实际上是因为我对指令到硬件的实现不够熟悉，没有深入地去思考每条指令需要哪些硬件，需要哪些信号。  而在这门课程，在最开始设计那个信号表的时候，我才认真地去思考每条指令需要哪些硬件和哪些信号，才发现很多指令在硬件实现上基本都是类似的，才发现原来CPU实现不是原来想象中的那么难，这全归功于这门课将具体的指令简化为抽象的指令（通过数据通路表和控制信号取值表发现不同指令对硬件使用的相似性），然后基于抽象指令并使用模块化的思路来实现相应的硬件电路，通过这样，大大简化了设计，同时也实现了具体指令与硬件电路的联系，我发现单周期的CPU的代码量少的惊人，完全不是我原来所预想的那么困难，也是这样我更觉得通过抽象简化设计这一思想的伟大之处。  不过流水线CPU的实现确实相当困难，信号量相比单周期CPU多了数倍不止，并且需要深入思考每条指令的暂停机制和前递机制，当然，利用抽象的方法，其实流水线也就需要处理数据冒险和控制冒险的几种情况，不过主要还是需要对时序理解非常深刻。  **建议：**  CPU下板的指导弱了一点，指导书可以在这方面写的更详细一点，我在单周期CPU通过trace之后，面对CPU下板有点一头雾水的感觉，课程组提供了CPU下板的代码框架和总线的框架是好的，但是指导书关于这方面写的不够详细。并且对于CPU下板，这次实验应该是只需要实现数码管和拨码开关就行了，LED和按键开关既没有用到也没有考查，实验步骤需要更新。 |