

（深圳）

# 实验报告

开课学期：

课程名称： 计算机设计与实践

实验名称： CPU设计

实验性质： 综合设计型

实验学时： 52 地点：

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

2022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）。

|  |
| --- |
| 设计的功能描述（含所有实现的指令描述，以及单周期/流水线CPU频率） |
| 指令描述（24条）：  R-型指令  （1） add  指令名：加法指令  功能描述：rd ← rs + rt; PC ← NPC (PC + 4)。 32 位整数加法，源操作数分  别在 rs, rt 两个通用寄存器中，结果放在 rd 寄存器。由于本设计无溢出检  测，因此该指令功能同 addu。  （2） sub  指令名：减法指令  功能描述：rd ← rs - rt; PC ← NPC (PC + 4)。 32 位整数减法，源操作数分别  在 rs, rt 两个通用寄存器中，结果放在 rd 寄存器。  （3） and  指令名： 逻辑与  功能描述：rd ← rs and rt; PC ← NPC (PC + 4)。 32 位数按位逻辑与，源操作  数分别在 rs, rt 中，结果放在 rd 寄存器。  （4） or  指令名： 逻辑或  功能描述：rd ← rs or rt; PC ← NPC (PC + 4)。 32 位数按位逻辑或，源操作  数分别在 rs, rt 中，结果放在 rd 寄存器。  （5） xor  指令名： 逻辑异或  功能描述：rd ← rs xor rt; PC ← NPC (PC + 4)。 32 位数按位逻辑异或，源操  作数分别在 rs, rt 中，结果放在 rd 寄存器。  （6） sll  指令名: 逻辑左移  功能描述： (rd)←(rt)<<shamt; PC ← NPC (PC + 4)。逻辑左移，将 rt 寄存器  中的 32 位数逻辑左移后赋给 rd，低位用 0 填充，移位的位数是 shamt。  （7） srl  指令名: 逻辑右移  功能描述： (rd)←(rt)>>shamt; PC ← NPC (PC + 4)。逻辑右移，将 rt 寄存器  中的 32 位数逻辑右移后赋给 rd，移位的位数是 shamt。  （8） sra  指令名: 算术右移  功能描述： (rd)←(rt)>>>shamt; PC ← NPC (PC + 4)。算术右移，将 rt 寄存  器中的 32 位数算术右移后赋给 rd，移位的位数是 shamt。算术右移时，符  号位不仅要参与移位，还要保留。  I-型指令  （9） addi  指令名：有符号立即数加法指令  功能描述：(rt)←(rs) + (Sign-Extend)immediate; PC ← NPC (PC + 4)。首先将 有符号立即数扩展到 32 位，然后加上 rs 中的数，结果给 rt 寄存器。  （10） andi  指令名：立即数逻辑与指令  功能描述：(rt)←(rs) AND (Zero-Extend)immediate; PC ← NPC (PC + 4)。首先  将立即数零扩展到 32 位，然后同 rs 中的数按位逻辑与，结果给 rt 寄存器。  （11） ori  指令名：立即数逻辑或指令  功能描述：(rt)←(rs) ORI (Zero-Extend)immediate; PC ← NPC (PC + 4)。首先  将立即数零扩展到 32 位，然后同 rs 中的数按位逻辑或，结果给 rt 寄存器。  （12） xori  指令名：立即数逻辑异或指令  功能描述：(rt)←(rs) XORI (Zero-Extend)immediate; PC ← NPC (PC + 4)。首先  将立即数零扩展到 32 位，然后同 rs 中的数按位逻辑异或，结果给 rt 寄存器。  （13） slli  指令名：立即数逻辑左移指令  功能描述： (rd)←(rt)<<immediate; PC ← NPC (PC + 4)。逻辑左移，将 rt 寄存器中的 32 位数逻辑左移后赋给 rd，低位用 0 填充，移位的位数是immediate。  （14） srli  指令名：立即数逻辑右移指令  功能描述： (rd)←(rt)>>immediate; PC ← NPC (PC + 4)。逻辑右移，将 rt 寄存器中的 32 位数逻辑右移后赋给 rd，移位的位数是immediate。  （15） srai  指令名：立即数算术右移指令  功能描述： (rd)←(rt)>>>immediate; PC ← NPC (PC + 4)。算术右移，将 rt 寄存器中的 32 位数算术右移后赋给 rd，移位的位数是immediate。算术右移时，符号位不仅要参与移位，还要保留。  （16） lw  指令名：存储器读（字操作）  功能描述：(rt)←Memory[(rs)+(sign\_extend)offset]; PC ← NPC (PC + 4)。以 rs  寄存器的内容为基地址，offset 通过符号扩展后形成 32 位的偏移，将基地  址加上偏移形成一个 32 位的地址，以此地址从 RAM 中读出一个字赋给 rt 寄存器。  （17） jalr  指令名：跳转与连接寄存器指令  功能描述：(rd)←(PC)+4; (PC)←( (rs1)+immediate)。先将下条指令的地址（(PC)+4）保存在rd作为过程的返回地址，然后无条件转移到新的地址。新地址是寄存器rs1中的值加上立即数。  S-型指令  （18） sw  指令名：存储器写（字操作）  功能描述：Memory[(rs)+(sign\_extend)offset]←(rt); PC ← NPC (PC + 4)。以 rs  寄存器的内容为基地址，offset 通过符号扩展后形成 32 位的偏移，将基地  址 加上偏移形成一个 32 位的地址，将 rt 寄存器的内容写入到 RAM 中该  地址开始的一个字单元。  B-型指令  （19） beq  指令名：相等则转移指令  功能描述：if ((rt)=(rs)) then (PC)←(PC)+4+( (Sign-Extend) offset<<2); else PC ← NPC (PC + 4)。如果 rt 和 rs 的值相等，则转移到新的地址。新地址是当前指令的下一条指令地址 （PC+4）加上一个 32 位偏移量。  （20） bne  指令名：不相等则转移指令  功能描述：if ((rt)≠(rs)) then (PC)←(PC)+4+( (Sign-Extend) offset<<2); else PC ← NPC (PC + 4)。如果 rt 和 rs 的值不等，则转移到新的地址。新地址是当前指令的下一条指令地址 （PC+4）加上一个 32 位偏移量。  （21） blt  指令名：小于则转移指令  功能描述：if ((rt)<(rs)) then (PC)←(PC)+4+( (Sign-Extend) offset<<2); else PC ← NPC (PC + 4)。如果 rt 和 rs 的值不等，则转移到新的地址。新地址是当前指令的下一条指令地址 （PC+4）加上一个 32 位偏移量。  （22） bge  指令名：大于或等于则转移指令  功能描述：if ((rt)>(rs)||(rt)=(rs)) then (PC)←(PC)+4+( (Sign-Extend) offset<<2); else PC ← NPC (PC + 4)。如果 rt 和 rs 的值不等，则转移到新的地址。新地址是当前指令的下一条指令地址 （PC+4）加上一个 32 位偏移量。  U-型指令  （23） lui  指令名：立即数赋值指令  功能描述：(rt)←immediate<<12 & FFFFF000H ; PC ← NPC (PC + 4)。首先 20 位立即数赋给 rt 寄存器的高 20 位，低 12 位用0 填充。  （24） jal  指令名：跳转指令  功能描述：(rd)←(PC)+4; (PC)←( PC+immediate(zero\_extend))。先将下条指令  的地址（(PC)+4）保存在rd作为过程的返回地址，然后无条件转移到新的地址。新地址是原地址加上20位立即数拓展到32位。  单周期CPU频率分析：  (1)根据指导书要求，CPU频率为25MHz：  如图，并未出现违约 |
| 设计的主要特色（除基本要求以外的设计） |
| (1)模块划分清楚，结构清晰，高内聚、低耦合。如，在Bridge模块中整合了所有外部设备，直接利用CPU中ALU生成的地址寻址，并返回对应的读出数据，避免其他复杂逻辑 |
| 资源使用、功耗数据截图（Post Implementation；含单周期、流水线2个截图） |
| 以下是示例，请贴自己的图。 |

1 单周期CPU设计与实现

1.1 单周期CPU整体框图

|  |
| --- |
| 要求：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，以及说明每个模块的功能含义。 |
| ①NPC:组合逻辑，根据控制信号输出CPU下一条要执行的指令地址；  ②PC:在时钟上升沿输出CPU要执行的指令地址；  ③IROM:指令只读存储器，输入指令地址输出对应地址的指令；  ④SEXT:位拓展器，根据控制信号输出对应输入din的拓展数ext；  ⑤RF:寄存器堆，rD1输出rR1号寄存器内的数据，rD2输出rR2号寄存器内的数据，当写入控制信号有效时在时钟上升沿把wD写入wR；  ⑥BC:比较器，比较rD1与rD2，输出对应BrEq与BrLt信号，当BrUn为一时做无符号比较否则做有符号比较；  ⑦ALU:根据控制信号对ab两个输入数据做对应运算输出结果c；  ⑧Bridge:根据地址与使能信号等对外部存储器等外设进行读或写操作；  ⑨Control:根据指令产生对应的各种控制信号。 |

1.2 单周期CPU模块详细设计

|  |
| --- |
| 要求：画出各个模块的详细设计图，包含内部的子模块，以及关键性逻辑；标出子模块接口信号名、各信号线的信号名和位宽，并有详细的解释说明。 |
| (1)ALU:  根据控制信号，对指定的数据进行算术运算、逻辑运算与移位操作，输出所需要的数据。   1. Bridge:   根据地址与控制信号，判断CPU要对哪个器件进行读或写操作，输出CPU所需要的数据，写操作时向对应器件输出CPU所需要传送的数据。 |

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以及波形分析；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。 |
| ①and：  指令为0020F733: and x14,x1,x2  故alu\_op=2，即做and运算，alu\_a=x1中的值F00FF00F,alu\_b=x2中的值F0F0F0F0，得到结果F000F000并写回x14中。  ②sw：  指令为0020A423：sw x2，(8)x1  故alu\_op=0，即做add运算，alu\_a=x1中的值00002000，alu\_b=拓展后的立即数00000008，结果为00002008，取其[15:2]段即0802作为地址向其存入x2中的值0AA00AA0  ③jal：  指令为010026F： jal x2,16  故alu\_op=0，即做add运算，alu\_a=当前指令地址PC=0000000C，alu\_b=拓展后的立即数00000010，结果为0000001C作为下一条指令地址npc |

2 流水线CPU设计与实现

2.1 流水线的划分

|  |
| --- |
| 要求：画出流水线如何划分，说明每个流水级具备什么功能、需要完成哪些操作。 |
| ①IF:  功能：取指  操作：IROM根据PC输出的指令地址输出对应指令；  ②ID:  功能：译指  操作：Control、RF、SEXT根据IF/ID寄存器输出的指令得到对应的控制信号、对应寄存器的值与拓展数；  ③EX:  功能：执行指令  操作：ALU、BC根据ID/EX寄存器输出的控制信号、操作数进行运算；  ④MEM:  功能：访存  操作：根据EX/MEM寄存器中的地址对DRAM进行读/写操作；  ⑤WB:  功能：写回  操作：根据MEM/WB寄存器中的地址对RF中对应寄存器进行写操作； |

2.2 流水线CPU整体框图

|  |
| --- |
| 要求：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，以及说明每个模块的功能含义。 |
| ①NPC:组合逻辑，根据控制信号输出CPU下一条要执行的指令地址；  ②PC:在时钟上升沿输出CPU要执行的指令地址；  ③IROM:指令只读存储器，输入指令地址输出对应地址的指令；  ④SEXT:位拓展器，根据控制信号输出对应输入din的拓展数ext；  ⑤RF:寄存器堆，rD1输出rR1号寄存器内的数据，rD2输出rR2号寄存器内的数据，当写入控制信号有效时在时钟上升沿把wD写入wR；  ⑥BC:比较器，比较rD1与rD2，输出对应BrEq与BrLt信号，当BrUn为一时做无符号比较否则做有符号比较；  ⑦ALU:根据控制信号对ab两个输入数据做对应运算输出结果c；  ⑧Bridge:根据地址与使能信号等对外部存储器等外设进行读或写操作；  ⑨Control:根据指令产生对应的各种控制信号。  ⑩hazard\_ctrl:根据IF/ID、ID/EX、EX/MEM、MEM/WB寄存器中的数据判断是否存在数据冒险与控制冒险，并根据具体的冒险种类向PC、IF/ID、ID/EX输出前递信号与数值、停顿信号、清空信号。 |

2.3 流水线CPU模块详细设计

|  |
| --- |
| 要求：画出各个模块的详细设计图，包含内部的子模块，以及关键性逻辑；标出子模块接口信号名、各信号线的信号名和位宽，并有详细的解释说明；此外，必须结合模块图，详细说明数据冒险、控制冒险的解决方法。 |
| hazard\_ctrl:根据IF/ID、ID/EX、EX/MEM、MEM/WB寄存器中的数据判断是否存在数据冒险与控制冒险，并根据具体的冒险种类向PC、IF/ID、ID/EX输出前递信号与数值、停顿信号、清空信号。  ①数据冒险：  检测：IF/ID中要从中取值的寄存器与ID/EX、EX/MEM、MEM/WB中要写回的寄存器相同。  解决：从ID/EX、EX/MEM、MEM/WB中取对应数值传给ID/EX进行前递。  加载类数据冒险检测：IF/ID中要从中取值的寄存器与ID/EX中要写回的寄存器相同，且ID/EX中写回的数值为访存得到的读数据。  解决：IF/ID、PC进行停顿，等待下一个周期从内存中取出数据后前递处理。  ②控制冒险：  检测：EX阶段检测ID/EX中的npc\_op信号，为1时代表要跳转发生了控制冒险。  解决：清空IF/ID、ID/EX，PC进行停顿，在下一时钟周期更新正确的npc，流水线重新启动。 |

2.4 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。 |
|  |

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. 问题/错误：因指令、变量繁多，在疏忽大意下出现不少如位宽错误、接线错误等问题；解决方法：通过分析波形图发现问题并追溯问题、定位错误，修改错误代码。 2. 问题/错误：时序混乱，即使是较简单的单周期时序设计，对于PC更新时机、RF写入时机、DRAM写入时机难以安排；解决方法：通过研读指导书内容对时序逻辑进行合理安排。 3. 问题/错误：verilog的使用，在实现算术移位等部分功能时不了解verilog的相关语法；解决方法：查询资料后学习。 4. 问题/错误：冒险控制模块hazard\_ctrl的设计与实现，在实现流水线CPU的冒险控制时，该模块复责检测各类冒险的发生并发出相应的控制信号以应对冒险，但课程结束时仍在debug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。 |
| 在本次课程中，我从零开始实现一颗CPU，从第一次上课时的迷茫无助，到最后完成时的自豪，在这整个过程中，我一遍遍复习理论课的知识，深刻理解了什么叫“纸上得来终觉浅，绝知此事要躬行”，终于通过指导书和老师、同学们的帮助、自己的不断摸索，一步一个脚印，从数据通路图与控制信号表，到各个模块的编写，最后实现了整颗CPU的实现。  在一次次的debug中，我深刻意识到了初期对整个项目建立整体的认识、建立一个完善的、易改进的整体架构有多么重要，由于初期没有这样的意识，在后面的编写中我不断重构原有代码，造成了很多重复劳动，大大增加了开发负担。  而由于水平和时间所限，最终的CPU仍然称不上令我满意，对于流水线CPU我没能完成可以对冒险进行处理的版本，而对于单周期CPU，我认为也有许多地方值得改进优化。如：数据通路或许可以划分得更合理、时钟周期安排可以通过进一步分析原理而安排得更合理等等。  总的来说，这次课程让我领略到了硬件的魅力，也让我对自己的工程能力、时间能力进行了锻炼，我在此次课程中受益匪浅。 |