一、Vivado 基本操作

1. 修改 project 属性（设置 FPGA 芯片类型），打开工程目录，找到设计文件所在目录。

2. 添加并设置 ip 核，或者打开已有的 ip 核, 对其关键设置（RAM, prgrom 选 1）做说明。

64 KB = 64 \* 1024 Bytes = 64 \* 1024 \* 8 bits, 64 \* 1024 \* 8 / 32 = 16384

3. 选择仿真对象（在多个testbench文件中选择其中一个作为当前的仿真对象），设置仿真时间。

4. 在已有的波形中添加指定的指定信号（设计模块的内部信号）进行观测。

二、Verilog 语法

1. 请在你的设计代码中找出组合逻辑模块、时序逻辑模块，说明该模块的敏感列表，以及敏 感列表与该模块执行的关系

2. 确定复位信号是高电平有效还是低电平有效，是异步复位还是同步复位

异步复位：在下一个时钟上升沿复位；同步复位：在rst跳边沿复位

3. 找出 阻塞赋值、非阻塞赋值对应的代码

阻塞赋值：=；非阻塞赋值：<=

4. 找出 CPU 的顶层模块文件，说明该文件中各子模块的例化中端口绑定的方式：是顺序绑 定还是以端口名进行绑定，二者各自的特点

顺序绑定：按照设计文件里面输入端口的顺序一一绑定；

端口名绑定：需要对每个输入指定端口名，方法是.a(a)；

三、对于单周期 CPU 的设计、实现的理解：（第一点必选 + (2，3，4 中) 3 选 1）

1. 单周期 CPU 在处理指令（lw，sw，jal，jr，subi，beq）时，内部操作（读写、运算）发 生的时刻点。（选择 2 条指令）

Lw: fetch(clk1\_pos) -> decoder(get address, sign-extend)(combinational logic) -> alu(address)(combinational logic)-> memory(read)(clk1\_neg) -> decoder(write)(clk2\_pos)

Sw: fetch(clk1\_pos) -> decoder(get address, sign-extend)(combinational logic) -> alu(address)(combinational logic)-> memory(read)(clk1\_neg)

Jal: fetch(clk1\_pos) -> decoder(pc+4 write to $31) (clk2\_pos)

-> (pc,instruction) -> fetch(update PC) (clk1\_neg)

Jr: fetch(clk1\_pos) -> decoder(read address)(combinational) -> fetch(update PC) (clk1\_neg)

Branch: fetch(read instruction) (clk1\_pos) -> decoder(signed-extend-immediate) (combinational) -> alu(pc+4+offset)(combinational) -> fetch(update PC)(clk1\_neg)

Subi: fetch(clk1\_pos) -> decoder(get address, sign-extend)(combinational logic) -> alu(address)(combinational logic)-> decoder(write back)(clk2\_pos)

2. MIPS32 指令系统中有哪些寻址方式，这些跳转的地址由哪些模块生成（jal，jr，bne）

Jal: {PC[31:28],Instruction[25:0],2'b00}

Jr: 地址来自寄存器

bne: pc\_plus\_4 + sign\_extend, by alu

3. 对于存储模块，地址端口的位宽如何确定（decoder 中访问寄存器组的地址端口，memory 中访问存储单元的地址端口）

有三十二个寄存器组，地址线为5位

Memory：64 KB = 64 \* 1024 Bytes = 64 \* 1024 \* 8 bits, 64 \* 1024 \* 8 / 32 =16384= 2^14

4. 单周期 CPU 内对于外设 IO 端口的处理方式（对应的指令以及具体操作）

1st step is to place the upper 20 bits of the port address in $28, 前22为1

lui $28,0xFFFF

ori $28,$28,0XF000

2nd step is to read the dial switch port

lw $s1,0XC70($28) sw $s1,0XC60($28)

lw $s1,0XC72($28) sw $s1,0XC60($28)

四、通过仿真波形文件对你的单周期 CPU 进行功能验证（两种方案选择 1 个）

1. 指出 cpu 的内部工作时钟，复位信号生效时 PC 寄存器的值，复位信号在何时失效

Clock, 0, 低电平

基于指令（jal，jr，beq，addi，lw，sw），观测：

2. 该指令处理时，取址模块的输出信号（机器指令的十六进制表示，对照 coe 文件）

3. 如果为跳转类的指令，该指令执行后的下一个周期，取址模块的输出信号（机器指令的 十六进制表示，对照 coe 文件）

4. 涉及到的子模块，以及该子模块与该指令处理相关的输出信号

1. Jal：controller 的 jal 信号. 给 decoder 的 opcplus4 信号（该值不同的实现有不同的结果， 以 jr 执行完毕后可以实现 callee 的返回为判断依据）

Jal = 1； pc\_plus\_4 >> 2

2. Jr：controller 的 jrn 输出信号, 以及 ifetch 从 decoder 获取的$ra 寄存器的值.

Jrn = 1;

3. beq: 从 decoder 中读取的数据，controller 的 branch 和 nbranch 信号，alu 的 zero 信号， alu 的 add\_result（计算得到的地址）

4. Addi：decoder 的读出信号（包括寄存器的读出以及立即数的扩展），alu 的 alu\_result， 写回操作完毕后在 decoder 中 rd 对应寄存器的值。

5. Lw 访问拨码开关：decoder 的读出信号（包括寄存器的读出以及立即数的扩展），alu 的 alu\_result，从拨码开关的读入信号，写回操作完毕后在 decoder 中 rt 对应寄存器的值。

6. Sw 访问 led：decoder 的读出信号（包括寄存器的读出以及立即数的扩展），alu 的 alu\_result，写入操作完毕后 led 上的信号。