CPU设计verilog版

整体思路：

先参照P3的图封装好的模块在ise中也生成这么多的.v文件，根据端口定义好input与output端口，再在主.v文件中实例化各模块即可。注意哪些是组合电路，哪些是时序电路，用好assign和always语句。

EXT模块：

信号名	端口	描述
immediate_16[15:0]	I	16位的立即数
ExtOp	I	1:符号扩展,0:0扩展
immediate_32[31:0]	O	扩展成32位的立即数

ALU模块：

信号名	端口	描述
SrcA[31:0]	I	第一个操作数
SrcB[31:0]	I	第二个操作数
ALUOp[2:0]	I	不同ALUOp，不同运算操作
zero	O	输出SrcA-SrcB是否为0的信号，如果为0，zero=1，反之zero=0
ALUResult[31:0]	O	根据ALUOp得到SrcA与SrcB的运算结果

ALUOp	操作
000	and
001	or
010	add
011	lui
110	sub

GRF模块：

信号名	端口	描述
A1[4:0]	I	读到RD1的寄存器的地址
A2[4:0]	I	读到RD2的寄存器的地址
A3[4:0]	I	进行写操作的寄存器的地址
WD[31:0]	I	写入的数据
clk	I	时钟信号
reset	I	同步复位信号
WE	I	1：可写，0：无效
RD1[31:0]	O	从A1的寄存器读出的数据
RD2[31:0]	O	从A2的寄存器读出的数据

Instr_split模块

信号名	端口	描述
Instr[31:0]	I	从ROM中读出来的指令
opcode[5:0]	O	mips机器码
funct[5:0]	O	
Instr25_21[4:0]	O	
Instr20_16[4:0]	O	
Instr15_11[4:0]	O	
immediate_16	O	16位立即数
Instr25_0	O	未扩展的j型指令的立即数

IM模块：

IM容量为4096*32bit，故采用4096个32位的寄存器表示ROM。

信号名	端口	描述
PC[31:0]	I	程序计数器
Instr[31:0]	O	从ROM取得的指令

DM模块：

DM容量为3072*32bit，为实现统一性，故采用4096个32位的寄存器表示RAM，减少了logisim中RAM地址为12位的浪费。

信号名	端口	描述
address[31:0]	I	从RAM读出的地址
writedata	I	写入的数据
clk	I	时钟信号
WE	I	1：能写入，0：不能写入
reset	I	1：RAM清0，0：无效
DMOp[2:0]	I	000:无所谓，因为WE会恒为0，001:lw,sw指令,010:lh,sh指令,011:lb,sb指令
readdata	O	从RAM读出的数据

PC模块：

信号名	端口	描述
clk	I	时钟信号
reset	I	同步复位信号
PCSrc[2:0]	I	000:pc=pc+4,001:pc=pc+4+sign_extend(offset||0^2),010:pc=pc=pc31...28||instr||0^2,011:pc=ra
immediate_32	I	32位立即数
immediate_26	I	j指令的26位立即数
ra	I	返回的地址
pc_out	O	输出的pc

Control模块：

信号名	端口	描述
opcode[5:0]	I	机器码的一部分，R型指令为6'b0
funct[5:0]	I	机器码的一部分
zero	I	1：SrcA=SrcB，0：SrcA!=SrcB
MemtoReg[2:0]	O	000:从ALU运算单元取得的ALUResult，001：从内存中取得的数据，010：PC+4
MemWrite	O	1：对内存进行写操作，0：写操作无效

信号名	端口	描述
ALUOp[2:0]	O	决定了运算操作
ALUSrc	O	1：用立即数进行操作，0：用RD2进行操作
RegDst[2:0]	O	000：GRF写入的寄存器是rt寄存器，001：GRF写入的寄存器是rd寄存器，010：GRF写入的寄存器是31号（$ra）寄存器
RegWrite	O	1：可以进行写入操作，0：不能进行写入操作
ExtOp	O	1：对符号数进行符号扩展，0：0扩展
PCSrc[2:0]	O	决定PC的计算方式
DMOp[2:0]	O	决定DM的读写方式，是以字还是半字还是字节为单位进行读写。 000：字方式，lw,sw指令，001:半字方式，lh,sh指令，010:字节方式，lb,sb指令

建议：

上传cpu文档的时候记得把图附上去，不然太抽象了。

思考题

1.addr信号是从ALU运算单元得到的结果。因为存储单元是按字为空间，而mips系统是以字节为单位，按字节寻址，所以保证是4的倍数，故要从第2位开始而不是从第0位开始，故addr位数是[11:2]而不是[9:0]。

2.

```
assign ALUOp = (add|lw|sw)? `ADD:
    (sub)? `SUB:
    (ori)? `OR:
    (lui)? `LUI:
    0;
```

```
case(ALUOp)
    `ADD: ALUResult = SrcA + SrcB;
    `SUB: ALUResult = SrcA - SrcB;
    `LUI: ALUResult = SrcB<<16;
    `ORI: ALUResult = SrcA|SrcB;
```

3.

异步复位：reset信号优先级比clk信号高，无论是否是时钟的上升沿，只要reset信号有效，系统立即复位。

同步复位：reset信号有效当且仅当处在clk上升沿的阶段，如果clk不为上升沿，即使reset置1也发挥不了复位作用。

4.因为在考虑溢出的情况下，addiu与addi的区别就在于addiu是忽略溢出的，而addi是考虑溢出的，同样add指令考虑溢出，addu不考虑溢出，因此在忽略溢出的前提下，addi与addiu是等价的，add与addu是等价的。

注意：

判断一个数是否小于0，最好取最高位(imm[31]==1 -> 负数，immp[31]==0 ->正数)，看最高位是否为1来比较，避免(num<0)的写法。因为给出的数是否是有符号数题目已经给出要求了，我们要做的是简便比较。