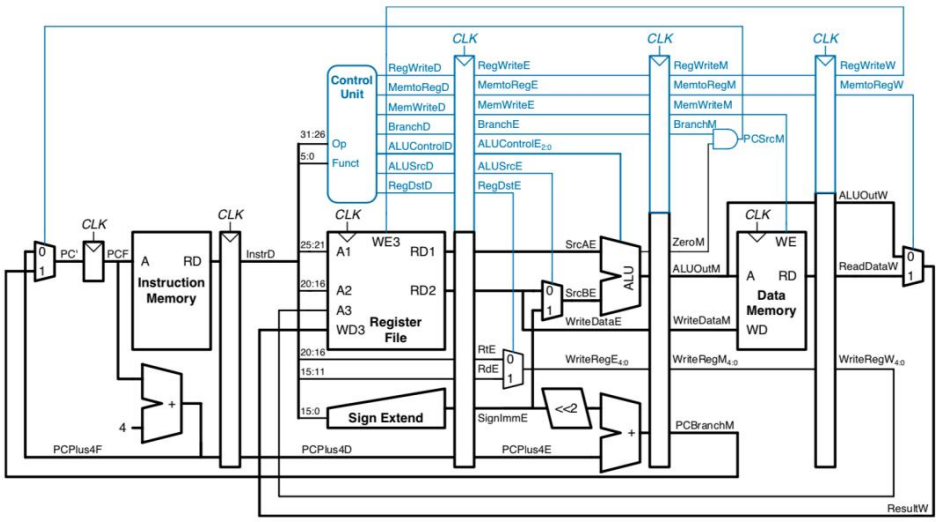
Overview

In project3,we are required to implement a 5-stage pipeline cCPU using Verilog language which can execute the MIPS instructions We are provided the instruction memory module and data memory module. We are expected to implement the register file, ALU module, and the control unit. For testing, we have totally 8 test file, which have different types of hazard existing in each test file.

In my project, i design the CPU with 7 files:clr_alu.v, parsing_inst.v, alu.v, cpu.v, regFile.v , Instruction.v , and MainMemory.v. In alu.v, i implement all the 27 arithmetic methods including add, addi sub slt, etc. In RegisterFile.v, 32 32-bit register are provided. The Instruction.v and the MainMemory are provided, we can use them to get the series of instruction and operate the main memory to modify their value according to the instruction. Finally, i use the cpu.v file as the top-level file of the whole pipelined cpu, which organized other .vfiles to achieve all the functions of the cpu. In addition , i solve all the data hazard and control hazards using the methods of fowarding and stalling to make my pipeline cpu run as save as possible.

2 Main part

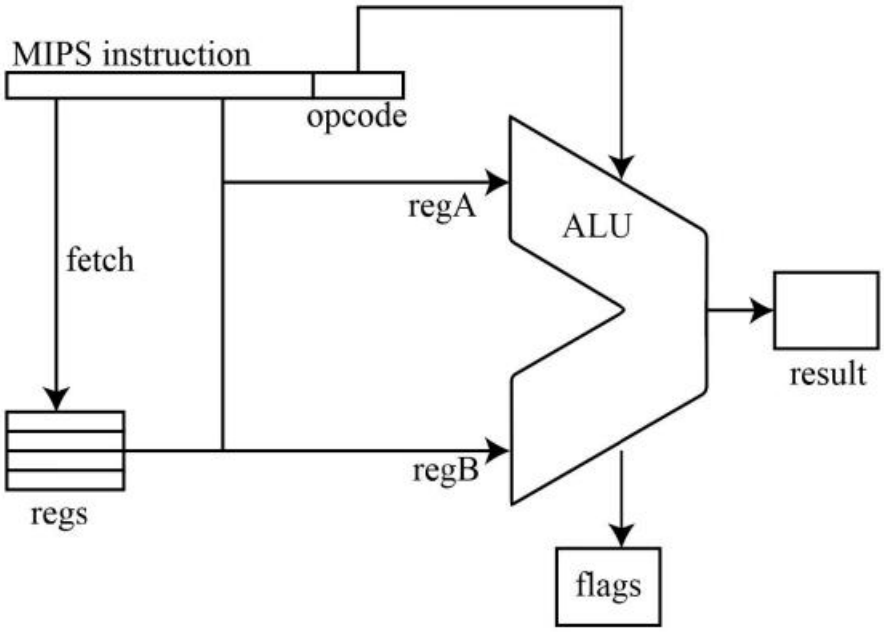


2.1 alu.v

In this file, i have implemented computing functions of,

including DATA transfer instruction : lw, sw,

Arithmetic instructions: - add, addu, addi, addiu, sub, subu
Logical instructions - and, andi, nor, or, ori, xor, xori
Shifting instructions - sll, sllv, srl, srlv, sra, srav
Branch/Jump instructions:
- beq, bne, slt,j, jr, jal.
The alu module receives 3 inputs and give 2 outputs.

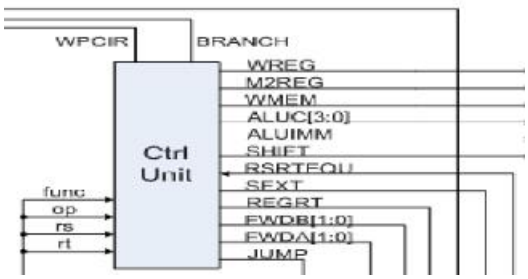


Alu.v that i implement receives the operate types signal(opcode) from ALU_control.v output and two regs. And according to the instruction opcode operate regA and regB.

2.2 parsing_inst.v

In this module, i decode the instructions and transform them into opcode, and set corresponding signals.

2.3 ctrl_alu.v



In this module, it receives MIPS instruction code and parse it and then transmit opcode(ALu_control)to alu part.What's more it output many various signals to handle all the five stages check signals and handle three kinds of hazard.

2.4regfile.v

This file implement simulation of all the 32-bitsregister groups used in the real computer. They store temporarily data to make sure cpu can run 5-stage at the same time.

2.5InstructionRAM.v

This element module is provided by bb, which receive the address of the instruction.

2.6MainMemory.v

The same as 2.5, address of the memory is input , stored the value in the address.

3.7cpu.v

I make it as a top-level file of the 5-stage pipelined cpu.

The following are the 5-stage:

(1)IF

(2)ID

(3)EXE

(4)MEM

(5)WB

4 About hazards

Personally, I think It is the most important and difficult part in my project.Let's suppose that A and B instruction a occur hazard and B after A.One central solution is to make the B step that using the same register postponed to A step finish store its value.

4.1 forward

The forward hazard has three cases. We represent them as 00 01 10. The "00" means that no forward is needed. The "10" means that operand is forward from prior alu result. The "01" means that the operand is forwarded from data memory or an earlier alu result. BTW, sometimes we will meet continuous hazard which need to handle two or more forward. The trick to handle this is select the most recent to forward.

```
add x1,x1,x2
add x1,x1,x3
add x1,x1,x4
```

4.2 stalling

The central thought of stalling is to insert some bubbles(NOP) to keep two instruction in the same stage.We must stop the behind for some clock to wait the MEM update the register value.

4.2.1 branch

We use prediction to handle jump instruction hazard.When the jump instruction is not met, cpu don’t need to stall. When the jump instruction meet the condition to stall, the jump instruction needs to stall a clock to jump.

About running :

Since i wrote all my code by modelsim, so i have no makefile. And all the steps to run my code are included in the explanation.doc which is implemented by Chinese.