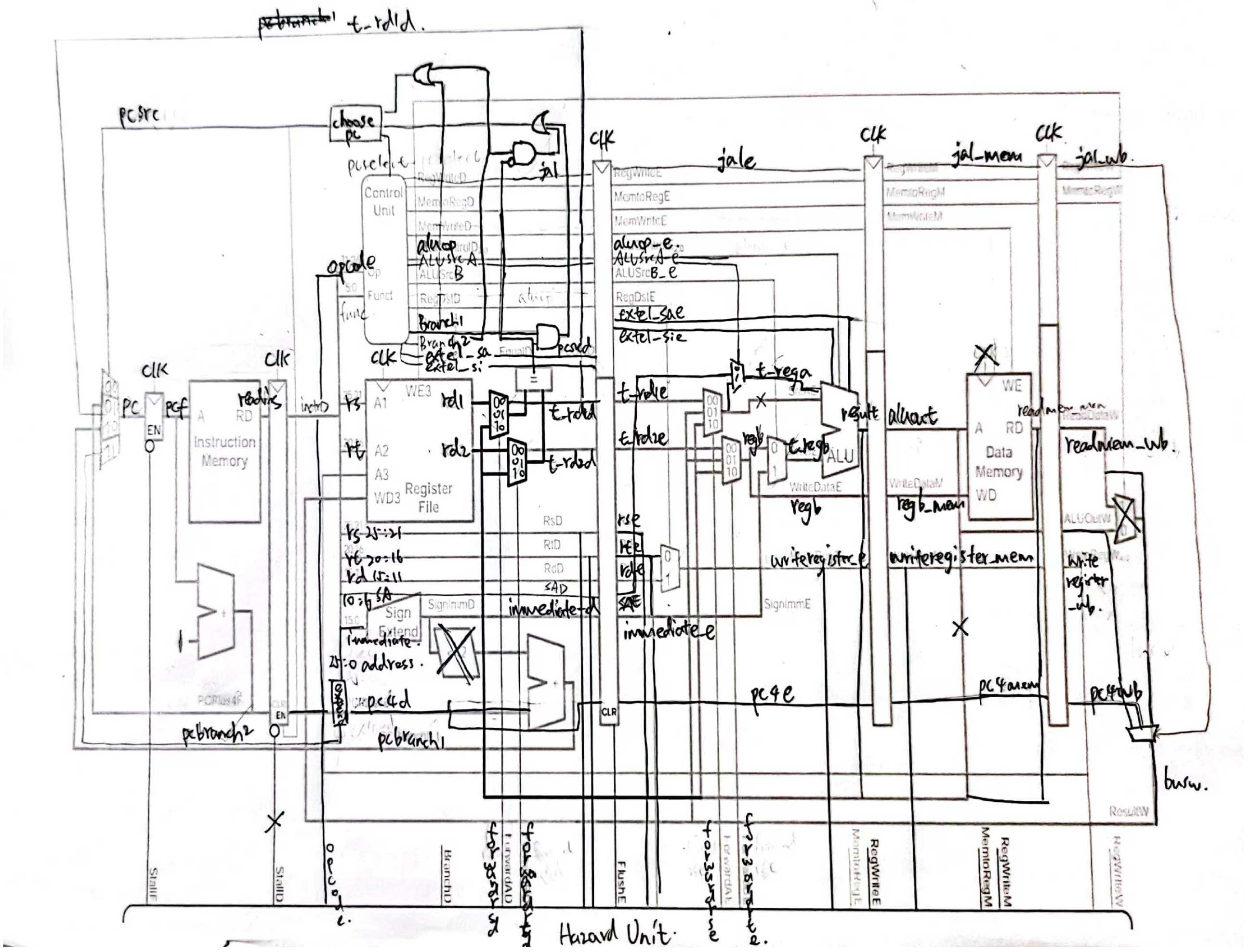
CSC3050 Assignment 4 Report (120090211)

Overview:

This is a pipeline CPU based on Verilog HDL. The whole CPU will be controlled by one CLK signal. Once the program starts, all needed instructions will be loaded into the instruction memory, and then it will execute in the way defined by the selected machine codes. If the picked machine code is FFFF FFFF, then the CPU will end up executing. It will display how many cycles it takes and dump the data memory (output.txt) and the waveform file (test_CPU.vcd).

Processing Logic:

To implement this pipeline CPU, I use the following data flow chart.



As shown in the picture, I divide the whole CPU into several modules, including (1) Choosepc, (2) PC (3) IFIDRegister, (4) Control_Unit, (5) Registerfile, (6)Chooseinput, (7) IDEXRegister (8) ALU (9) EXMEMRegister (10) MainMemory (11) MEMWBRegister (12) HazardUnit (13) InstructionRAM

(1) Choosepc

This module is a 4-1 multiplexer. In this module, the address of the next instruction will be chosen by the pcsrc.

If pcsrc=00 (without branch), the output will be the past address + 1.

If pcsrc=01, the output will be the address of the next instruction which is calculated by the beq/bne instruction

If pcsrc=10, the output will be the address of the next instruction which is stored in Registerfile[rs]. (jr)

If pcsrc=11, the output will be the address of the next instruction which is calculated by the j/jal instruction

(2) PC

This module will keep loading the updated pc address in its inner register and output the address of the next instruction at every clock cycle if there is no stall.

(3) IFIDRegister (7) IDEXRegister (9) EXMEMRegister (11) MEMWBRegister

These pipeline registers store the last stage's information at the rising edge and output the next stage with information stored inside the registers.

(4) Control_Unit

This control unit generates controlling signals of this CPU according to the opcode and function part of the instruction.

(5) Registerfile

This module contains 32 general registers. It keeps sending out the content stored in rs and rt. At every rising edge of the clock cycle, the data inside the register file will update.

(6) Chooseinput

This module is a 3-1 multiplexer. This module is designed for data hazards, and it will help to choose the updated inputs.

If the control signal is 00, the output will be the present stage data.

If the control signal is 01, the output will be the next stage data.

If the control signal is 10, the output will be the second next stage data.

(8) ALU

This module will do the simple logical or arithmetic calculation based on the control signals (aluop) and two inputs.

(10) MainMemory

This module acts as the cache of the computer. The CPU can dynamically store and access the data inside. It consists of 512 32-bit registers. The index starts from 0.

(12) HazardUnit

This module will recognize whether the CPU will do forwarding or stall.

(13) InstructionRAM

It will store the machine codes once the program starts. It consists of 512 32-bit registers. The index starts from 0.

Only the PC, Registerfile, IF/ID Register, ID/EX Register, EX/MEM Register, MEM/WB Register are sequential circuits, the rest of the modules are combinational.

Implementation details:

1. The meaning of some important control signals.
   RegWrite: Whether the register file will update.
   MemtoReg: Whether the data read from the memory will be written back to the register file.
   Memwrite: Whether the data memory will update.
   RegDst: Whether the writeback register will be rt or rd.
   alu_srcA: Whether the first input of the ALU will be sa part of the instruction or Registerfile[rs]
   alu_srcB: Whether the second input of the ALU will be Registerfile[rt] or sign-extend immediate.
   extsel_im: Whether the immediate will be sign-extend or zero-extend.
   extsel_sa: Whether the shifting operation in CPU will be logical or arithmetic.
   extsel_si: Whether the ALU will do sign operation or unsigned operation.
   aluop: The type of ALU operation.
   0010 denotes addition
   0000 denotes AND operation
   0001 denotes OR operation
   1001 denotes XOR operation

1100 denotes NOR operation
0110 denotes subtraction
0111 denotes comparison
1010 denotes shift left
1011 denotes shift right

branch1: Whether the instruction is beq
branch2: Whether the instruction is bne
pcselect: The type of pc address.
00 (default value)
01 denotes beq/bne
10 denotes jr
11 denotes j/jal

jal: Whether the write back data will be a pc address.

2. The technique used to determine hazards.
   Hazard may occur if one of the following conditions is met.

   a. Data hazard in ID stage or EX stage
   1) The rs/rt in the ID stage is the same as the write-back register in EX or MEM stage
   2) The next or the second next stage will write back data to the register file.
   3) The write-back register is not $zero
   b. lw Stall
   1) The instruction in the EX stage is an lw instruction
   2) The rs/rt in the ID stage is the same as the write-back register in EX stage.

   If lw stall happens, pc will not update and will remain at the address of the last instruction.

3. The technique used to achieve write before reading in the register file.
   At the rising edge, the Register file will update if needed. At the falling edge, the Register file will read data from the register file. As a result, it will always write before reading.

4. The technique used to achieve flushing in branch hazards.
   Once the hazard unit recognizes the occurrence of the branch hazard, the address for the next instruction will be sent to the wire pc simultaneously and wait for the next cycle to be stored in the IF/ID register. Therefore, there is no need to flush the IF/ID register since it will update in the next cycle.