Some control signals might lack

General tag distribution, only store instructions do not need it (how many tags for each buffer)

Now we cannot load without previous store and store must be one by one (broadcasting/comparing address?)

Only one jalr can wait for execution, otherwise decoding will stop.

Branch Predictor

Input: logic [31:0] PC (PC of current instruction), logic [31:0] IR (Current Instruction), error\_b\_branch\_predict, pc\_b\_branch\_wrong\_predict, error\_jalr\_predict, 32 bit real\_jalr\_destination, pc\_jalr\_wrong\_predict

Output (With Instruction Queue): logic [31:0] PC\_next (PC of next/predicted instruction), logic br\_en

Function:

If opcode is not jal, jalr, branch (6 kinds), PC\_next = PC\_next + 4

If opcode is jal, PC\_next = PC + imm

If opcode is jalr or branch (6 kinds), predict PC\_next (always PC\_next + 4 in the first version)

br\_en indicates the branch prediction for B branch.

Timing Issue:

Returning PC\_next is combinational logic

Notice

May need other ports for branch predictor

Instruction Queue

Storage Line: PC (32 bits), Instruction (32 bits), PC\_next(32 bits), br\_en

Input: logic [31:0] PC\_predict (from branch predictor), logic br\_en (from branch predictor), logic [31:0] IR\_next (from memory hierarchy), logic memory\_resp, logic decode\_read, logic flush

Output: logic [31:0] PC\_decode, logic [31:0] IR\_decode [31:0], logic [31:0] PC\_next, logic br\_en, logic [31:0] PC\_fetch (to memory hierarchy), logic memory\_read, logic decode\_readable

Function and Timing Issue:

decode\_readable means there are valid instructions in the queue. PC\_decode, IR\_decode, PC\_next, br\_en are the instruction the decoder wants, and they are always accessible if the decoder wants, and they exist. PC\_next is specially used for jalr. br\_en is used for B branch prediction.

decode\_read indicates the decoder wants another instruction, and all instructions in the queue move by one position.

flush means branch misprediction and all instructions need to be omitted (new PC will be sent to branch predictor and then to instruction queue by PC\_next).

IR\_next, PC\_predict, memory\_resp, memory\_read are related with reading a new instruction and storing in the queue. PC\_predict is the same as the input PC\_next.

ALU Reservation Station and ALU

Storage Line: busy, tag, alu\_op, Qj, Qk, Vj, Vk

Input: Common Data Bus, flush, (from decoder) new\_line, tag, alu\_op, Qj, Qk, Vj, Vk

Output: Common Data Bus, line\_available

Function and Timing Issue:

Follow the Tomasulo algorithm inside the workstation and the common data bus.

line\_available means there are empty positions in the reservation station.

new\_line indicates the decoder has decoded a new instruction needs ALU, and the reservation station needs to find a position (busy) as well as store tag, alu\_op, Qj, Qk, Vj, Vk.

It can handle instructions R – add, sub, sll, xor, srl, sra, or, and, as well as I – addi, xori, ori, andi, slli, srli, srai, U- lui, aupic, J – jal

flush means branch misprediction and all lines need to be omitted

Qj, Qk are related with tag, 0 means using the value.

CMP Reservation Station and CMP

Storage Line: busy, tag, cmp\_op, Qj, Qk, Vj, Vk, branch\_or\_slt, pc, immediate, br\_en\_predict

Input: Common Data Bus, flush, (from decoder) new\_line, tag, cmp\_op, Qj, Qk, Vj, Vk, branch\_or\_slt, pc, immediate, br\_en\_predict

Output: Common Data Bus, line\_available, error\_b\_branch\_predict, pc\_b\_branch\_wrong\_predict

Function and Timing Issue:

It should handle two kinds of instructions, B - branch instructions (beq, bne, blt, bge, bltu, bgeu), and set-less-than instructions (R – slt, sltu, I – slti, sltiu), and branch\_or\_slt distinguishes them. They both use busy, tag, cmp\_op, Qj, Qk, Vj, Vk, branch\_or\_slt.

For set-less-than instructions, common data bus reports the results.

B branch instructions use additionally pc, immediate, br\_en\_predict, error\_b\_branch\_predict, pc\_b\_branch\_wrong\_predict, the common data bus reports the real PC\_next (reoder buffer will use it), and if br\_en\_predict is not the same as br\_en, indicates error\_b\_branch\_predict. pc\_b\_branch\_wrong\_predict will then be used by branch predictor for update.

line\_available means there are empty positions in the reservation station.

new\_line indicates the decoder has decoded a new instruction needs CMP.

flush means branch misprediction and all lines need to be omitted.

Jalr Reservation Station (only 1 line)

Storage Line: busy, tag, Qj, Vj, imm, pc\_prediction

Input: Common Data Bus, flush, (from decoder) new\_line, tag, Qj, Vj, imm, pc\_prediction, read\_jalr\_pc

Output: Common Data Bus, line\_available, logic error\_jalr\_predict, 32 bit real\_jalr\_destination, pc\_jalr\_wrong\_predict

Function and Timing Issue:

tag and bus are used for reg[rd] = PC + 4, and 32 bit real\_jalr\_destination is pc = (reg[rs1] + sext(offset)), which will be compared with pc\_prediction, if they are not the same, indicates logic error\_jalr\_predict, and pc\_jalr\_wrong\_predict will then be used by branch predictor for update. real\_jalr\_destination will be kept until receive read\_jalr\_pc.

line\_available means there are empty positions in the reservation station.

new\_line indicates the decoder has decoded a new instruction that is jalr.

flush means branch misprediction and all lines need to be omitted.

Load Buffer

Storage Line: busy, tag, Qj, Vj, imm, storage\_number

Input: Common Data Bus, flush, new\_store, (from decoder) new\_line, tag, Qj, Vj, imm, storage\_number, logic memory\_resp, logic [31:0] memory\_read\_data

Output: Common Data Bus, line\_available, logic memory\_read, logic [31:0] memory\_address

Function and Timing Issue:

storage\_number means the remaining number of storages before load, and new\_store indicates a new storage operation, which will decrease all storage\_number which is not 0.

line\_available means there are empty positions in the buffer.

new\_line indicates the decoder has decoded a new instruction that is load.

flush means branch misprediction and all lines need to be omitted.

Store Queue

Storage Line: Qj, Qk, Vj, Vk,

Input: Common Data Bus, flush, (from decoder) new\_line, Qj, Qk, Vj, Vk, logic memory\_resp, logic able\_to\_store

Output: storage\_number, new\_store, line\_available, logic [31:0] memory\_address, logic [31:0] memory\_write\_data

Function and Timing Issue:

Store one by one, able\_to\_store is used to to prevent storage after branch\_misprediction and storage before load, new\_store means a new store has just been applied

line\_available means there are empty positions in the buffer.

new\_line indicates the decoder has decoded a new instruction that is load.

flush means branch misprediction and all lines need to be omitted.

Regfile

Storage line: (RegNumber) Value, tag

Input: (Read, source register) Reg1, Reg2, (Store) RegS, VS, TS, (Decoder, destination register) RegD, TD, flush

Output: V1, T1, V2, T2

Function and Timing Issue:

Read can output results “immediately”. Write needs a rising in the clock. Decoder means a new value will be stored to RegD which values are indicated by TD when a new instruction is decoded, so TS need to update no matter the behavior of Store, but read still need to read “old” tag value.

When store, always store VS to value, but only clear tag if TS is the same as tag in the line, and only for this case, if reading the same register, modify V\_output, T\_output to no tag, V\_output is VS

flush clear all tag.

If T1, T2 are not 0, decoder also need to check reorder buffer.

Reorder buffer

A Cycle

Storage line: (tag), busy, is\_load, is\_store, store\_successful, is\_branch, branch\_has\_predict, branch\_correct, register\_number, value

Input: Common Data Bus, new\_store, error\_b\_branch\_predict, error\_jalr\_predict, T1, T2, (from decoder) new\_line, is\_load, is\_store, is\_branch, register\_number

Output: line\_available, line\_tag, flush, RegS, VS, TS, able\_to\_store, read\_jalr\_pc, V1\_valid, V1, V2\_valid, V2

Function and Timing Issue:

(A Finite State Machine might be needed)

is\_store, is\_branch are used for instruction bytes.

Finishing of all instructions except store are indicated by tag, error\_b\_branch\_predict, error\_jalr\_predict are used for branch prediction checking. Use flush to flush.

Store one by one, able\_to\_store is used to to prevent storage after branch\_misprediction and storage before load, new\_store means a new store has just been applied.

If T1, T2 are not 0 in the register file, decoder also need to check reorder buffer.

Decoder

Decode instructions to different stations/buffers/queue, obtain registers (with the help of reorder buffer), insert reorder buffer.