109062320 朱季葳
Discuss with 109062101, 109062223, 109062233, 109062302, 109062323

1. (15 points) A common defect in silicon chips is for one signal wire to always get a constant logic value 0 (1, respectively); it is called a stuck-at-0 (stuck-at-1, respectively) fault. Consider the single-cycle processor for the following instructions: `ld`, `sd`, `add`, `sub` and `beq`.

(a)(5 points) Which instruction(s) could fail to operate correctly if the ALUSrc wire is stuck at 0?

ld and sd.
ALUSrc wire is the control signal used to choose rs2 or immediate to be one of the input of ALU.
"0" means "choose rs2", while "1"means "choose immediate", and since "R-type", and "SB-type"instructions need rs2,
"ld and sd"could fail to operate correctly if the ALUSrc wire is stuck at 0.

(b) (5 points) Which instruction(s) could fail to operate correctly if the MemtoReg wire is stuck at 1?

add and sub.
MemtoReg wire is the control signal used to choose "read data" or "ALU result".
"0" means "choose ALU result", while "1"means "choose Read Data", and since "Load" instructions need to read data, R-type instructions need to read ALU result, and store and conditional branch instructions don't need to write back anything; as a result , we "don't care" the value of MemtoReg when we executing store and conditional branch instructions. Therefore, "R-type-->add and sub" could fail to operate correctly if the MemtoReg wire is stuck at 1.

(c) (5 points) Which instruction(s) could fail to operate correctly if the ALUop0 wire is stuck at 0?

Beq.
"ld and sd' s ALUop0" is 0 , while beq is 1, R-type instructions is don't care.
As a result, "beq" could fail to operate correctly if the ALUop0 **wire is stuck at 0.**

2. (10 points) Consider the execution of the machine instruction $00CE3623_{hex}$ on the single-cycle processor.

(a)(7 points) What are the values of the signals: Branch, MemRead, MemtoReg, ALUOp, MemWrite, ALUSrc and RegWrite?

00CE3623 --> 0000 0000 1100 1110 0011 0110 0010 0011

→ opcode: 010 0011 --> S-type

→ 0000000 01100 11100 011 01100 0100011

→ imm[11:5] rs2 rs1 fun3 imm[4:0] opcode

→ immediate:000000001100 → 12

→ rs2 = x12, rs1 = x28

→ sd x12,12(x28)

Branch	MemRead	MemtoReg	ALUOp1	ALUOp0	MemWrite	ALUSrc	RegWrite
0	**0**	**x**	**0**	**0**	**1**	**1**	**0**

(b) (3 points) What are the input values of the ALU? You can use Reg[x] to denote the value of register x.

Reg[28] + 12_{dec} (Reg[rs1] + immed)

3. (10 points) Assume that the positive edge-triggered clocking methodology is adopted and the logic blocks used to implement the single-cycle processor have the following delay values:

I-Mem/ D-Mem	Register File	Mux	ALU	Adder	Single Gate	PC Read	Register Setup	Sign Extend	Control
230 ps	130 ps	20 ps	180 ps	130 ps	5 ps	20 ps	10 ps	30 ps	30 ps

"I-Mem/D-Mem" is the amount of time to access the Instruction or Data Memory.
"Register File" is the amount of time to read rs1 and rs2.
"PC Read" is the amount of time for the new PC value to appear on the output after a rising clock edge; this delay value applies to the PC only.
"Register Setup" is the amount of time a register's data input must be stable before a rising clock edge; this delay value applies to both the PC and Register File.

(a) (4 points) What is the latency of an `add` instruction (i.e., what is the minimum clock period to ensure that this instruction works correctly)?

In add instruction, we need
PC Read + I-Mem +
(instruction fetch and PC is incremented)
Register File +
(Two registers are read and since control is set simultaneously and the time cost of the former is longer than the latter, we only count the time cost of the former)
Mux + ALU +
(ALU operates on the data read from the register file)
Mux + Register Setup
(the result from ALU is written into the destination register)
= 20 + 230 + 130 + 20 + 180 + 20 + 10
= 610ps

(b) (3 points) What is the latency of an `ld` instruction (i.e., what is the minimum clock period to ensure that this instruction works correctly)?

PC Read + I-Mem +
(instruction fetch and PC is incremented)
Register File +
(Two registers are read and since 12-bit offset is extended is set simultaneously and the time cost of the former is longer than the latter, we only count the time cost of the former)
Mux + ALU + D-Mem +
(ALU operates on the data read from the register file)
Mux + Register Setup
(the result from ALU is written into the destination register)
= 20 + 230 + 130 + 20 + 180 + 230 + 20 + 10
= 840ps

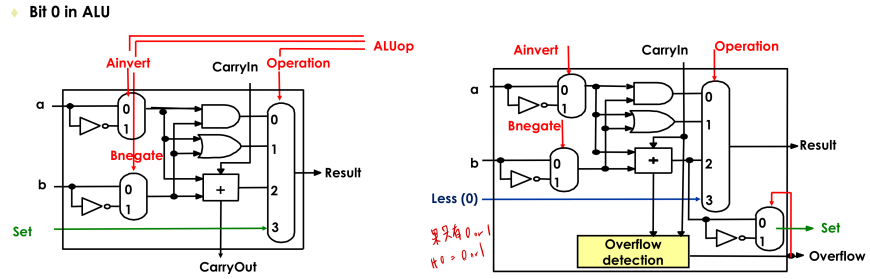
(c) (3 points) What is the latency of an `sd` instruction (i.e., what is the minimum clock period to ensure that this instruction works correctly)?

PC Read + I-Mem +
(instruction fetch and PC is incremented)
Register File +
(Two registers are read and since 12-bit offset is extended is set simultaneously and the time cost of the former is longer than the latter, we only count the time cost of the former)
Mux + ALU + D-Mem
(ALU operates on the data read from the register file)
= 20 + 230 + 130 + 20 + 180 + 230
= 810ps

4. (10 points) Recall that the ALU introduced in Chapter 3 can support the set-less-than (slt) operation by setting the ALU operation to 0111. Now consider how to modify the single-cycle processor such that it can also execute the R-type instruction: slt rd, rs1, rs2.

(a) (3 points) Are additional logic blocks or wires needed? Justify your answer.

Yes.
In order to justify rs1 is less than rs2 or not, we do rs1-rs2 in order to check that the result is positive or negative, and we can do it by subtraction, which can be done in the ALU we have.
But the result have to show the outcome that rs1 < rs2(result = 1) or rs1 >= rs2(result = 0), which is the reason why slt operation's result is different from subtraction even though both of them do the subtraction of rs1 and rs2.
Therefore, we need to add additional logic blocks or wires since for bit 0, we have to check the sign bit of rs1-rs2(if sign bit >=0, then rs1 >= rs2.Else, rs1 < rs2), which is bit 63 in ALU, and except bit 0, all the other bits have to be 0 for any condition.
What's more, we also have to add overflow detection if we don't need it, and a mux to choose the result which is used to set bit 0's less wire, just like the images below.



(b) (3 points) Does the "Control" unit need any change? Justify your answer.

No.

Since we have already known that slt is R-type, and for R-type instructions, all the control signals are the same; therefore, we don't have to change it anymore.

(c) (4 points) Does the "ALU control" unit need any change? Justify your answer.

Yes.

Since the ALUoperation of slt is 0111, we need to change ALUcontrol's output in order to generate "0111" when encountering the slt instruction.

Input: ALUOp, Funct7, Funct3 of slt instruction

Output:0111

5. (10 points) Assume that individual stages of a datapath have the following latencies:

IF	ID	EX	MEM	WB
300 ps	400 ps	450 ps	350 ps	250

(a) (4 points) What are the clock periods in a five-stage pipelined processor and a single-cycle processor, respectively?

Five-stage Pipelined:

Since EX stage 's latency is the biggest (450), the clock period in a five-stage pipelined processor is 450ps.

Single-cycle processor:

Since we have to do all the stages in a single-cycle for the single-cycle processor, the clock period is

300 + 400 + 450 + 350 + 250 = 1750ps.

(b) (4 points) What are the latencies of an sd instruction in a five-stage pipelined processor and a single-cycle processor, respectively?

Five-stage Pipelined 's total latency = 450*5 = 2250ps

Single-cycle processor 's total latency = 1750 – 250 = 1500ps
Since the sd instruction is done when MEM stage, we don't have to add the latency for WB stage.

(c) (2 points) If you can split one stage of the five-stage pipelined processor into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock period? Note that the new clock period must be minimum among all possible ways of splitting.

I will split EX stage 's latency because the clock period in a five-stage pipelined processor is determined by the stage whose latency is the longest. Therefore, only when we split EX stage 's latency will the clock period in a five-stage pipelined processor change, and the new clock period is 400, which is the longest latency among all the stages after splitting EX stage 's latency

(450→ 450/2=225).

6. (10 points) Answer the following questions by assuming the clock rate is 500 MHz and no pipeline stalls occur.

(a) (5 points) If a 4-stage pipelined processor takes 50 ns to execute S instructions, how long will it take to execute 4S instructions?

To execute S instructions, we need 4+(S-1) cycles, and since
[4+(S-1)]/(500*10^6) = 50*10^(-9)
S+3 = 25000*10^(-3) = 25
→ S = 22

Then for 4S instructions, we need [4 + (4S-1)]/(500*10^6) seconds.
(3+4*22)/(500*10^6) = (3+88)/(500*10^6) = 91/(500*10^6)
= 18.2*10^(-8) = 182*10^(-9) = 182ns

(b) (5 points) Consider a pipelined processor which has N stages. If it takes 100 ns to execute S instructions and 340 ns to execute 4S instructions. What are N and S, respectively?

[N+(S-1)]/(500*10^6) = 100*10^(-9)
→ N + S – 1 = 50000*10^(-3) = 50
→ N + S = 51 –> equation 1
[N+(4S-1)]/(500*10^6) = 340*10^(-9)
→ N + 4S – 1 = 170000*10^(-3) = 170
→ N + 4S = 171 –> equation 2
By equation 1 and equation 2:
3S = 120 → S = 40
N + 40 = 51 → N = 11

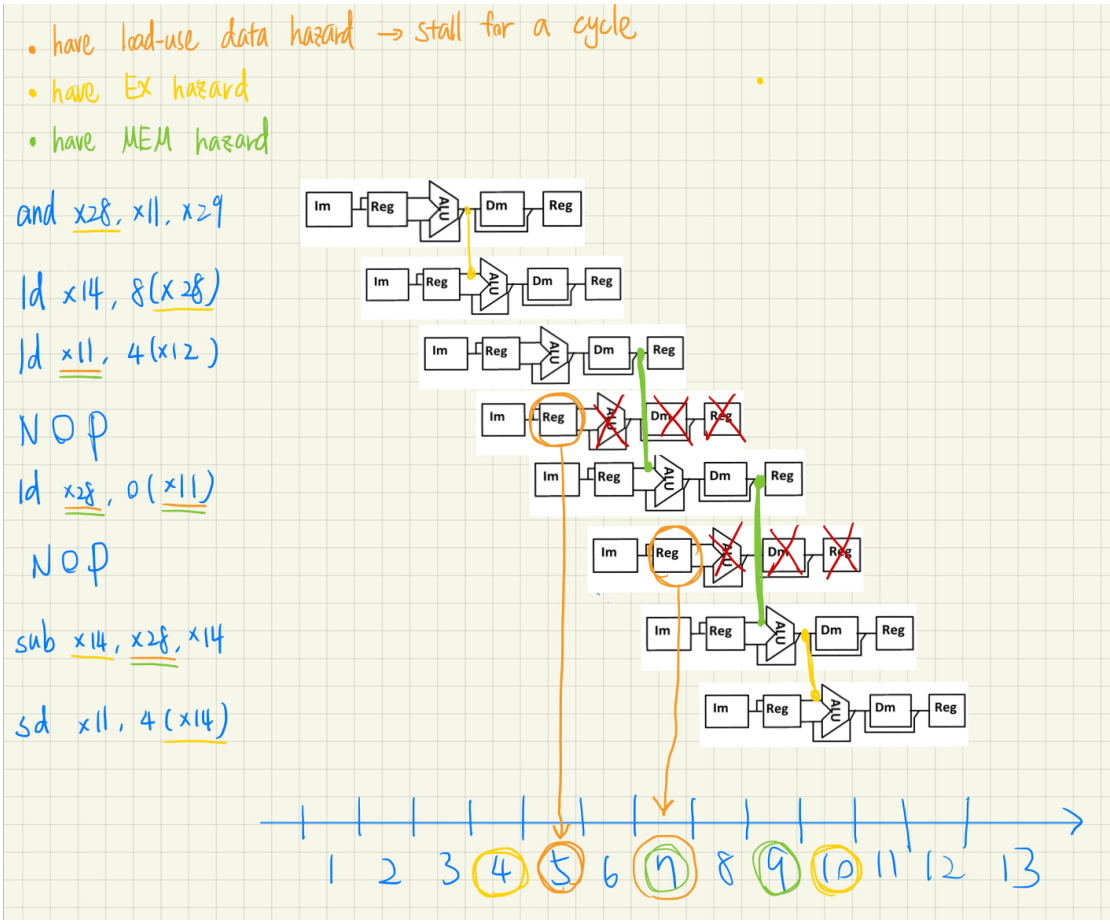
7. (20 points) Consider the following sequence of instructions executed on the five-stage pipelined processor. Assume that the execution starts in clock cycle 1.

```
and x28, x11, x29
ld x14, 8(x28)
ld x11, 4(x12)
ld x28, 0(x11)
sub x14, x28, x14
sd x11, 4(x14)
```

(a) (4 points) Assume both forwarding unit and hazard detection unit are not present in the processor. Insert a minimum number of `NOP` (no operation) instructions to ensure correct execution.

and x28, x11, x29
NOP
NOP
ld x14, 8(x28) → x28 is used in EX stage
ld x11, 4(x12)
NOP
NOP
ld x28, 0(x11) → x11 is used in EX stage
NOP
NOP
sub x14, x28, x14 → x28 is used in EX stage
NOP
NOP
sd x11, 4(x14) → x14 is used in EX stage

(b) (12 points) Assume the processor has the forwarding unit and hazard detection unit. For each of the following three conditions, indicate in which clock cycle, it is true for the processor.



(i) (4 points) All control signals to be stored in the ID/EX register are set to 0.

We set control values in ID/EX register to 0 in order to stall for a cycle, which means that there is a "load-use data hazard" because we can solve other data hazards such as EX hazard and MEM hazard by forwarding.
As a result, there is a load-use data hazard for the instruction "ld x28, 0(x11)" and "sub x14, x28, x14",and by the above graph, we can see their clock cycle 5 and 7, respectively.

(ii) (4 points) The correct data is forwarded from the EX/MEM register to one ALU input.

This situation happens in "EX hazard", and the condition of it is EX/MEM.RegisterRd = (ID/EX.RegisterRs1 or ID/EX.RegisterRs2) under the condition that EX/MEM.RegWrite =1 and EX/MEM.RegisterRd != 0
Consequently, there is a " EX hazard" for the instruction "ld x14, 8(x28) " and "sd x11, 4(x14)", and by the above graph, we can see their clock cycle is 4 and 10, respectively.

(iii) (4 points) The correct data is forwarded from the MEM/WB register to one ALU input.

This situation happens in "MEM hazard", and the condition of it is MEM/WB.RegisterRd = (ID/EX.RegisterRs1 or ID/EX.RegisterRs2) under the condition that MEM/WB.RegWrite =1 and MEM/WB.RegisterRd != 0
Consequently, there is a " MEM hazard" for the instruction "ld x28, 0(x11)" and "sub x14, x28, x14", and by the above graph, we can see their clock cycle is 7 and 9, respectively.

(c) (4 points) Assume the processor has the forwarding unit and hazard detection unit. How many clock cycles does the processor take to complete the execution of the code?

1+1+1+2(stall for a cycle)+2(stall for a cycle)+1+ 4 (the last instruction's ID stage to WB stage) = 12

8. (10 points) Consider the sequence of branch outcomes: T, T, T, NT, NT, NT, NT, for a branch instruction that has been executed 7 times in a program, where T denotes taken and NT denotes not- taken.

(a) (4 points) What are the accuracy rates of the always-taken and always-not-taken predictors for this sequence of branch outcomes, respectively?

Always-taken:
3/7 ~= 42.86%
Always-not-taken:
4/7 ~= 57.14%

(b) (3 points) Consider a 1-bit dynamic predictor which starts at the NT state. What is the accuracy rate of the predictor for this sequence of branch outcomes?

Let "predict taken" be 1, "predict not-taken" be 0.
"Start at the NT state" means that we predict not taken(0) at first.
T, T, T, NT, NT, NT, NT
0, 1, 1 , 1 , 0 , 0 , 0
Accuracy rate: (7-2)/7 = 5/7 ~= 71.43%

(c) (3 points) Consider a 2-bit dynamic predictor which starts at the "strongly predict taken" state. What is the accuracy rate of the predictor for this sequence of branch outcomes?

Let "strongly predict taken" be 11, "strongly predict not-taken" be 00,
"weakly predict taken" be 10, "weakly predict not-taken" be 01.
T , T , T , NT, NT, NT, NT
11,11,11, 11 , 10, 01, 00
Accuracy rate: (7-2)/7 = 5/7 ~= 71.43%

9. (5 points) Consider the execution of the following sequence of instructions on the five-stage pipelined processor:

```
add x10, x28, x29
sub x6, x31, x28
beq x28, x29, LABEL
sd x28, 0(x29)
```

Suppose the third instruction is detected to have an invalid target address and cause an exception in the ID stage (i.e., in clock cycle 4).
What instructions will appear in the IF, ID, EX, MEM, and WB stages, respectively, in clock cycle 5?
Note that each instruction in your answer should be one chosen from the given instructions, the NOP instruction (or bubble), and the first instruction of the exception handler.

IF: **first instruction of the exception handler**
ID: **bubble(NOP)**
EX: **bubble**
MEM: `sub x6, x31, x28`
WB: `add x10, x28, x29`