計算機組織與 RISC-V CPU 設計 · 第 25 章

五級管線 CPU: 危障、前遞與停頓

IF/ID/EX/MEM/WB 管線暫存器 / 前遞單元 / 載入使用停頓 / 分支沖除 / 完整 Verilog 實作

摘要

本章把第 24 章的單週期 CPU 改造成 五級管線 (IF-ID-EX-MEM-WB)。理論基礎來自第 12 章: 管線化以指令級重疊換取吞吐量, 但引入資料危障與控制危障。本章的重點是把課本的抽象規則落成可綜合的 **Verilog**: 四組管線暫存器怎麼切、前遞單元的優先序怎麼寫、載入使用危障為何必須停頓一拍、分支錯誤時要沖除哪兩級, 以及一個容易被忽略的距離 **3** 危障——它的解法藏在暫存器檔的寫穿 (write-through) 裡。所有程式碼都通過了專門設計的危障壓力測試。

目錄

1　管線切割: 四組暫存器

把單週期資料路徑沿「五個階段」切開, 相鄰階段之間放入管線暫存器, 保存該指令後續所需的全部資訊:

暫存器	攜帶的資料	攜帶的控制	寬度 (約)
IF/ID	pc, pc+4, instr	(無,ID 才解碼)	96 b
ID/EX	pc, rs1_data, rs2_data, imm, rs1/rs2/rd 編號	ALU 控制、alu_src、mem_*、wb_*	200+ b
EX/MEM	alu_y, rs2_data(供 store), rd, pc+4	mem_read/write、wb_*	140 b
MEM/WB	alu_y, dmem_rdata, rd, pc+4, imm	reg_write、wb_sel	140 b

切割原則

- 控制訊號跟著指令走:ID 一次解碼完, 之後隨管線暫存器逐級傳遞、逐級「用掉」。
- 任何後級要用的值 (如 store 的 rs2、jal 的 pc+4) 都必須一路帶下去, 不能回頭讀。
- 每級只做自己的事:IF 取指、ID 解碼讀暫存器、EX 運算、MEM 存取、WB 寫回。

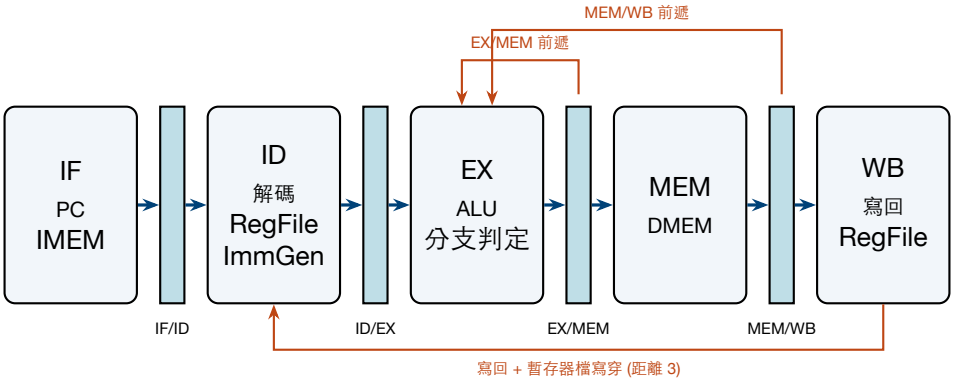

圖 1　五級管線與三條「補救路徑」

2　資料危障與前遞

2.1　三種距離的 RAW 危障

設指令 i 寫 rd, 指令 j 讀同一暫存器, 兩者相距 $d = j - i$ 條指令:

d	j 在 **EX** 時 i 在	解法	代價
1	MEM(結果在 EX/MEM)	EX/MEM → EX 前遞	0 週期
2	WB(結果在 MEM/WB)	MEM/WB → EX 前遞	0 週期
3	已寫回/正在寫回	暫存器檔寫穿 (同週期寫讀直通)	0 週期
1(load)	MEM(資料還沒讀出)	停頓 **1** 拍再前遞	1 週期

2.2　前遞單元

```
// EX/MEM 優先於 MEM/WB: 較新的結果才是正確值
wire fwd_a_exmem = exmem_reg_write && (exmem_rd != 0) &&
                   (exmem_rd == idex_rs1);
wire fwd_a_memwb = memwb_reg_write && (memwb_rd != 0) &&
                   (memwb_rd == idex_rs1) && !fwd_a_exmem;

wire [31:0] ex_rs1 = fwd_a_exmem ? exmem_fwd_data :
                     fwd_a_memwb ? wb_data        :
                                   idex_rs1_data;
// rs2 對稱; store 的寫入資料同樣要吃前遞後的 ex_rs2
```

程式 1 前遞邏輯 (riscv_pipeline.v)

兩個細節:(1) `rd != 0`——x0 永遠不構成相依;(2) `exmem_fwd_data` 不能直接用 `alu_y`: 若 EX/MEM 那條是 jal, 要前遞的是 pc+4; 是 lui, 要前遞的是 imm。因此前遞值本身也要過一個小多工器。

2.3 距離 3: 寫穿暫存器檔

WB 級在週期 t 的上升緣寫入;ID 級的指令在週期 t 用組合邏輯讀。若讀寫同一暫存器, 非同步讀會拿到*舊值*——這是我們在模擬中實際踩到的錯誤: 迴圈加總 $1+2+\cdots+10$ 回傳 1 而非 55。解法是在暫存器檔內部做寫穿:

```
wire bypass1 = (BYPASS != 0) && we && (wa != 5'd0) && (wa == ra1);
assign rd1 = (ra1 == 5'd0) ? 32'b0
           : (bypass1      ? wd : regs[ra1]);
```

程式 2 regfile.v 的 BYPASS 參數

參數化的原因

單週期 CPU 的寫回資料在同一週期內組合地依賴讀出資料; 若也開啟寫穿, 會形成組合迴路, 模擬器直接卡死 (我們實測如此)。因此 `BYPASS` 預設 0, 只有管線版例化時設 1。

3 載入使用危障: 唯一必須停頓的情況

載入的資料在 MEM 級*結束*才存在, 而下一條指令的 EX 級與該 MEM 級同週期——物理上無法前遞。偵測與處置:

```
wire load_use = idex_mem_read && (idex_rd != 0) &&
                ((idex_rd == id_rs1) || (idex_rd == id_rs2));

// 處置: PC 與 IF/ID 凍結一拍, ID/EX 插入氣泡(控制訊號清零)
wire stall  = load_use;
wire bubble = load_use;
```

程式 3 載入使用偵測與停頓

停頓一拍後, 載入結果落在 MEM/WB, 由既有的前遞路徑送達——不需要額外硬體。

4 控制危障: 分支沖除

本設計在 **EX** 級判定分支 (branch_unit 吃前遞後的運算元, 確保比較值正確)。判定錯誤時,IF、ID 兩級已裝入錯誤路徑的指令, 必須沖除:

```
wire redirect = idex_jump | idex_jalr | (idex_branch & ex_br_taken);
wire [31:0] redirect_pc = idex_jalr ? ((ex_rs1 + idex_imm) & ~32'b1)
                                    : (idex_pc + idex_imm);
// PC 選擇: redirect ? redirect_pc : pc+4 (含 stall 凍結)
// 沖除: redirect 時 IF/ID 與 ID/EX 同步清為 NOP(控制訊號歸零)
```

程式 4 重導向與沖除

代價: 每次 taken 分支 / jal / jalr 損失 **2** 個週期。這等效於「預測不採取」的靜態策略 (第 12 章); 練習 4 探討移到 ID 級判定把代價降到 1 拍的取捨。

5 驗證: 危障壓力測試

`verilog/sw/test_hazard.s` 專門建構最壞情境:

```
    addi x5, x0, 10      # 連續 RAW: 距離 1
    addi x6, x5, 1       #   需要 EX/MEM 前遞
    add  x7, x5, x6      #   x5 距離 2, x6 距離 1
    lw   x9, 0(x8)       # 載入使用:
    addi x10, x9, 1      #   必須停頓 1 拍
    lw   x11, 4(x8)      # 載入後緊接分支
    beq  x11, x12, target
    jalr x0, 0(x13)      # jalr 基址來自前遞
```

程式 5 test_hazard.s(節錄)

實測結果

兩套測試 (基本 81 條指令、危障壓力) 在單週期與管線版上全部通過, 寫入魔數 `0xC0DE600D`。管線版基本測試耗時 **133** 週期 (單週期 98): 多出的 35 拍來自載入使用停頓與每次 taken 分支/跳躍的 2 拍沖除 (81 條指令中呼叫與分支密集); 但管線版的時脈可以快約 3 倍 (第 24 章的關鍵路徑分析), 總體時間仍大幅領先。

6 本章重點回顧

本章要點

- 管線暫存器攜帶「資料 + 控制」, 控制訊號在 ID 一次解碼、逐級消耗。
- RAW 危障按距離分治:1、2 用前遞,3 用暫存器檔寫穿, 載入使用必須停頓 1 拍。
- 前遞要點:EX/MEM 優先、x0 排除、前遞值本身要選對 (ALU/pc+4/imm)。
- EX 級判定分支, 錯誤時沖除 IF、ID 兩級, 代價 2 拍。
- 寫穿旁路必須參數化——單週期版開啟會造成組合迴路。

7 練習

1. 畫出 `lw x9,0(x8); addi x10,x9,1; add x11,x10,x9` 的逐週期管線圖 (含停頓與前遞箭頭)。
2. 為什麼 store 指令的 rs2 也需要前遞? 寫出一段會因缺少該前遞而算錯的三行組語。
3. 若把前遞優先序寫反 (MEM/WB 優先), 構造一段程式使 CPU 算錯, 並解釋錯在哪。
4. 把分支判定移到 ID 級: 需要哪些額外前遞路徑? 哪些情況反而要多停頓? 比較兩種設計的 CPI。
5. 在 EX/MEM 前遞路徑上,jal 指令應前遞什麼值? 在程式碼中找出對應的多工器並說明。

參考資料

[1] D. Patterson and J. Hennessy, *Computer Organization and Design, RISC-V Edition*, Ch. 4.6–4.9.

[2] W. Stallings, *Computer Organization and Architecture*, Ch. 12(管線化理論).

[3] 本教材原始碼:`verilog/rtl/riscv_pipeline.v`、`verilog/sw/test_hazard.s`。