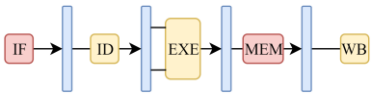
數位 IC Project: 基於四種 Hazard 解決方案之 MIPS Pipeline CPU

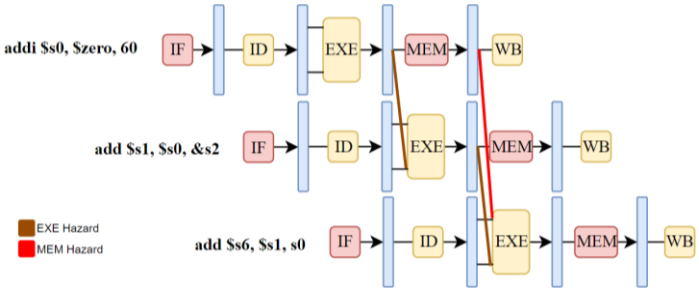
指導教授：蔡宗漢 | 學生: 徐松廷

本專題利用 Verilog 實作 MIPS pipeline CPU，分成五個 stage 以提升硬體利用率並增加吞吐量，同時解決 CPU 的四種 Hazard，分別是 EXE/MEM Data Hazard, load-use Hazard, Read after Write Hazard, Branch Hazard，本架構以 Vivado 執行功能性驗證，再以開源的 synthesis tool Yosys 確認本架構可以成功在 FPGA 上合成，未來會繼續以 FPGA 進行更完整的功能驗證。

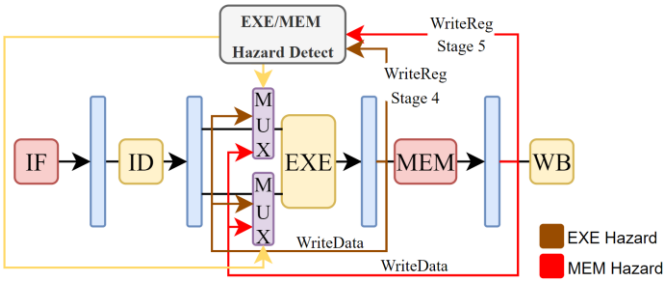


1. EXE/MEM Data Hazard Solution

本專題以 Stage 4, Stage 5 的 Write Data 對 Stage 3 的 Read Data 1,2 進行 forwarding，示意圖如下:

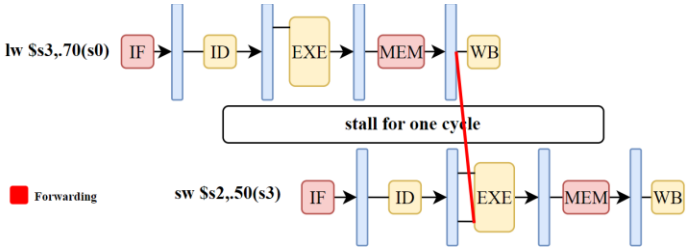


為解決 EXE/MEM Data Hazard 之電路架構如下:

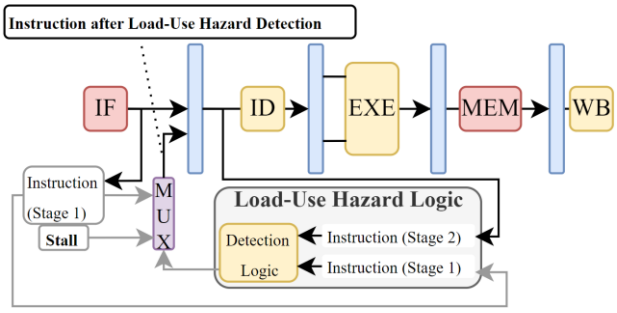


2. Load-Use Data Hazard Solution

當 Hazard Detection Logic 偵測到 lw 的$rt 會在下個指令被用到時代表產生 Load-Use Data Hazard，會在下個 cycle 讓 Stage 2 的 instruction 變成 Stall，讓 Stage2-5 都停擺一個週期，Load-Use Data Hazard 就會自動變成 MEM Data Hazard，即可使用 Forwarding 架構解決，示意圖如下:

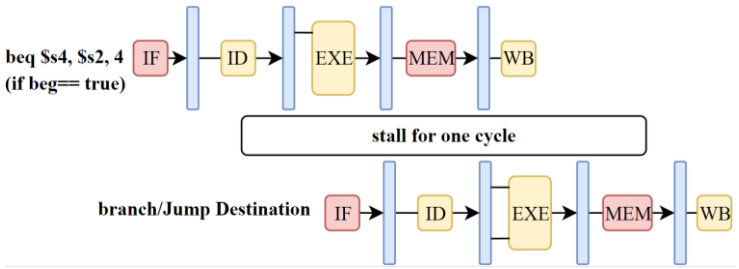


為解決 Load-Use Data Hazard 之電路架構如下:

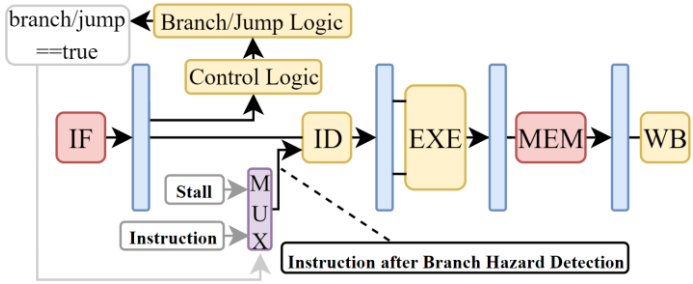


3. Branch Hazard Solution

在 MIPS 架構中 branch 的成立與否在 Stage 3 判定，我們特別將 branch 與 jump 的成立與否都提前到 Stage 2 判定，缺點是會使 Stage 2 的 critical path 變長進而降低最大操作頻率，優點是當 branch/jump 成立時，只需要 stall 1 cycle。

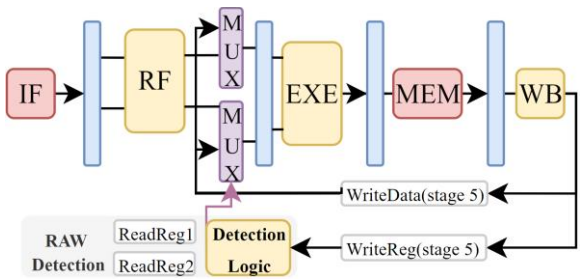


為解決 Branch Hazard 之電路架構如下:



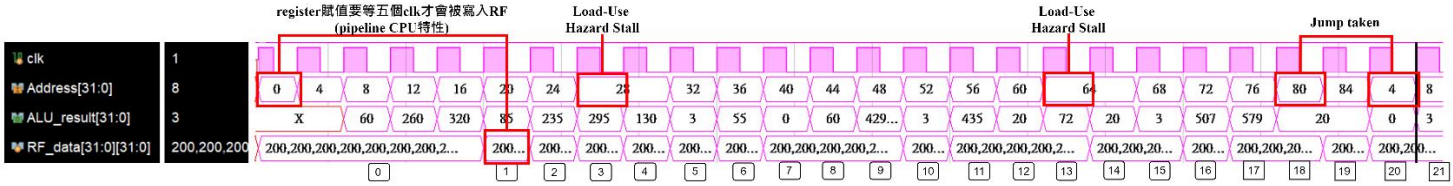
4. Read After Write Hazard Solution

stage 2 的 register file 的 write logic 為 non-blocking，所以要等一個周期才會寫入 RF，如果該 cycle 正在對特定變數進行寫入但該變數又被 read，就會產生 RAW Hazard，我們以 forwarding 解決。



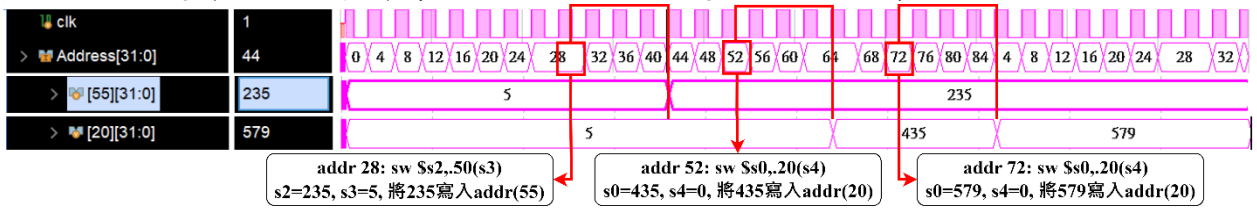
5. Simulation and Verification Result on Vivado

本專題以 24 個指令驗證功能正確性，包含 add, sub, addi, lw, sw, beq, j, or, and， test case 涵蓋 EXE/MEM Data Hazard, Load-Use Data Hazard, RAW Hazard, Branch Hazard 等以證明電路正確性與可靠性，test result 如下:



Instruction Memory	Register File Content After instructions exectuted	Verification
initial state	0 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200	
addr 0: addi $s0, $zero, 60	1 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,60,200	addr 0: |s0=60|
addr 4: add $s1, $s0, &s2	2 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,260,200	addr 4: |s1=260|
addr 8: add $s6, $s1, s0	3 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,320,200,200,200,200,260,60,200,200,200,200,200,200	addr 8: |s6=320|
addr 12: addi $s7, $zero, 85	4 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,320,200,200,200,200,260,60,200,200,200,200,200,200,200,200	addr 12:|s7=85|
addr 16: sub $s2, $s6, $s7	5 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,320,200,200,200,235,260,60,200,200,200,200,200,200,200,0	addr 16:|s2=235|
addr 20: add $s6, $s2,s0	6 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,200,5,235,260,60,200,200,200,200,200,200,200,0	addr 20:|s6=295|
addr 24: lw $s3,.70(s0)	7 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,200,200,235,260,60,200,200,200,200,200,200,200,0	addr 24:|s3=5|
addr 28: sw $s2,.50(s3)	8 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,200,200,235,260,60,200,200,200,200,200,200,200,0	addr 28:將235寫入addr(55)
Load-Use Hazard stall 1 cycle	9 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,200,200,235,260,60,200,200,200,200,200,200,200,0	//stall for 1 cycle
addr 32: slt $s4, $s1, $s2	10 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,0,5,235,260,60,200,200,200,200,200,200,200,0	addr 32:|s4=0|
addr 36: beq $s4, s2, 1	11 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,0,60,235,260,60,200,200,200,200,200,200,200,0	addr 36:|s2!=s0| no beq
addr 40: beq $s4, $zero, 1	12 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,0,60,235,260,60,200,200,200,200,200,200,200,0	addr 40:|s1=260| beq taken
Branch Hazard stall 1 cycle	13 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,0,60,235,260,60,200,200,200,200,200,200,200,0	//stall for 1 cycle
addr 44: and $s5, $s2, $s3	因為branch所以不會被執行,branch destination為address(48)	
addr 48: add $s0, $s5, $s2	14 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,0,60,235,260,435,200,200,200,200,200,200,200,0	addr 48:|s0=435|
addr 52: sw $s0,.20(s4)	15 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,0,60,235,260,435,200,200,200,200,200,200,200,0	addr 52:將435寫入addr(20)
addr 56: addi $s3, $zero, 72	16 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,295,200,0,72,235,260,435,200,200,200,200,200,200,200,0	addr 56:|s3=72|
addr 60: lw $s6,.20(s4)	17 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,435,200,0,72,235,260,435,200,200,200,200,200,200,200,0	addr 60:|s6=435|
Load-Use Hazard stall 1 cycle	18 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,435,200,0,72,235,260,435,200,200,200,200,200,200,200,0	//stall for 1 cycle
addr 64: or $s6, $s6, $s3	19 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,507,200,0,72,235,260,435,200,200,200,200,200,200,200,0	addr 64:|s6=507|
addr 68: addi $s0, $s6, 72	20 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,507,200,0,72,235,260,579,200,200,200,200,200,200,200,0	addr 68:|s0=579|
addr 72: sw $s0,.20(s4)	21 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,507,200,0,72,235,260,579,200,200,200,200,200,200,200,0	addr 72:將579寫入addr(20)
addr 76: lw $s5,.20(s4)	22 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,507,579,0,72,235,260,579,200,200,200,200,200,200,200,0	addr 76:|s5=579|
addr 80: j 5(address=20)	23 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,507,579,0,72,235,260,579,200,200,200,200,200,200,200,0	addr 80:jump to addr(4)
Jump to addr(4), stall 1 cycle	24 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,507,579,0,72,235,260,579,200,200,200,200,200,200,200,0	//jump stall
addr 4: add $s1, $s0, &s2	25 200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,85,507,579,0,72,235,814,579,200,200,200,200,200,200,200,0	addr 4: |s1=s0+s2=814|

sw 指令會間隔 3 個 clk 後對 DRAM 執行寫入，DRAM 256 個變數的初始值為 32'd5，DRAM 的模擬結果如下



6. 本專題實作之 MIPS Pipeline CPU with 4 Hazard Solutions 之電路架構如下