ICLab Final Project Checklist:

Checklist

Design Stage	Description	Fin	Comment
Software (/SW)	Your program (*.exe, *.m…)	y	1.檔案請放在/SW 資料夾內 2.與硬體對應的軟體實作 3.使用 ICcontest 者可忽略這部分
	Test pattern	y	測試資料，檔案請放在/SW/TP 內
RTL (/RTL)	Source file <Verilog file (*.v) >	y	1.檔案請放在/RTL/hdl 內 2.你所寫的電路檔、不會用來合成的 model… 所有.v 檔案
	Simulation file <testbench (*.v) ncverilog.log, mist.txt>	y	1.檔案請放在/RTL/sim 內 2.請在 misc.txt 描述哪些檔案需要加入模擬(或是自己寫好 filelist)，並且解釋 testbench 如何模擬你的功能
	Spyglass report <spyglass.rep>	y	檔案請放在/RTL 內
Synthesis (/SYN)	DC < netlist (*.v), synthesis scripts(*.tcl), synthesis log(*.log), reports(*.rep), misc.txt>	y	1.netlist 檔案請放在/SYN/netlist 內 2.其餘檔案請放在/SYN 內 3.請記得附上 timing, power, area 的 report 4. 請在 misc.txt 中描述設定 constrain 的原因以及合成的策略
P&R (/APR)	Innovus <netlist (*.v), delay (*.sdf) P&R log (*.log), reports (*.rep), DRC&LVS results (*.rep)>	y	1.netlist 檔案請放在/APR/netlist 內 2.其餘檔案請放在/APR 內 3.請記得附上 timing, area 的 report 4. 請在 misc.txt 中描述設定 constrain 的原因以及 P&R 的策略
	Power Analysis <pt_power.rep, scripts(*.tcl)>	y	1.檔案請放在/APR/PT 內 2. 不同階段的 power estimation (可參考 Lab13)
LEC (/LEC)	LEC <script (*.do, *.sh, *.f), log (*.log), misc.txt >	y	1. 請在 misc.txt 中說明如何跑 LEC 以及結果
pdf document. (/)	1.放入這份 Gxx_Checklist.pdf, Gxx_finalproject.pdf, 2.簡述如何使用你的 SW 驗證 3.請附上你們 project 資料夾的路徑，並公開權限讓助教可以抓到檔案	y	

Software Usage

Test Pattern 中可分為 instruction 與 golden 兩份不同的 txt file，instruction 指的是 cpu 能讀懂的 binary code，也就是從 riscv instruction set decode 出來的，而 golden 是從 riscv interpreter 得到 register 和 memory 的值，用來與 cpu 運算完的結果做比較。

Test Pattern 的兩個部分分別用三個部份的 python script 做處理。第一個是產生可閱讀的 instruction set。這個 instruction set 會方便我們在 debug 時做閱讀，也會在後面的步驟轉為 cpu 使用的 binary code。首先將 RV32I 的 instruction 做分類，並在分類後根據會產生 data hazard 的組合產生 pattern。主要會產生 hazard 的組合如下：

3 forward : ALU->ALU, ALU->LW, ALU->SW, AUIPC->LUI, LUI->AUIPC, LW->LW, LW->SW

2 forward+1 stall : ALU->BRANCH, ALU->JAL, LW->BRANCH, LW->ALU

A->B 指在 A operation 後接 B operation 會產生 data hazard。每一個 data hazard 都可以藉由塞三個 NOP operation，所以 3 bypass 是指這三個 NOP 都可以用 forward unit 來做跳過，而 2 forward 1 stall 則是指兩個需要 forward unit 一個需要 stall unit 才能跳過。由於我們有將不同 hazard 種類做分類，所以在實作時我們可以依據進度來調整我們生成的 test pattern。在未 implement forward, stall 時 test pattern 先加 NOP，implement 後則可以測不需要 NOP 的。第二個部分則是將可閱讀的 insrutcion set 轉為 cpu 可使用的 binary code。我們利用網路上的 riscv instruction decoder (https://luplab.gitlab.io/rvcodecjs/)，將剛剛第一部份產生的 instruction set 逐行的寫入該網站，並讀出我們需要的 binary code 寫入 intruction.txt。這部分藉由 python selenium 的 library 來達成。第三個部分則是將可閱讀的 instruction set 寫入網路上的 riscv interpreter (https://www.cs.cornell.edu/courses/cs3410/2019sp/riscv/interpreter/)，等該網頁執行完所有 instruction 後，讀出 registers 與 memory 的值並寫入 golden.txt。這三個部分都有利用 shell script 來使他們可以連續產生 100 或以上個不同 pattern。

另外在會產生 hazard 的 pattern 外，我們也另外做了 QuickSort 的 instruction set 進行測試，產生 binary code 與 golden 的方法一樣利用第二部分與第三部分的 script。

Project path

~u109022210/ICLAB/FinalProject/

Filetree

./RTL/hdl: 放置 RTL 檔案

top_riscv_core.v 最後完成電路的 top module，其餘的 module 皆為該 submodule。

./RTL/sim: 放置模擬的檔案

run_sim.sh：跑 test_top.v 與 test_top_quicksort.v 的 presim。

run_gatesim.sh：跑 test_top_quicksort.v 的 gatesim。

run_postsim.sh：跑 test_top_quicksort.v 的 postsim。

./RTL/spyglass.rpt：spyglass report PASS。

./SYN/run_syn.sh：跑合成

./LEC/pre_layout/run_lec.sh：跑 RTL 跟合成 netlist 的 LEC

./LEC/post_layout/run_lec.sh：跑 APR netlist 跟合成 netlist 的 LEC

./APR/innovus.log：包含 DRC、LVS 跟 ANTENNA 的 LOG 檔

./APR/report_APRarea.png：core area 截圖。

./APR/report_area.png：APR area 截圖。

./SW/run_recur.bat：直接執行整套 random pattern generation。