

Final-project Report

R13921072 何家祥

整體 CPU 框架

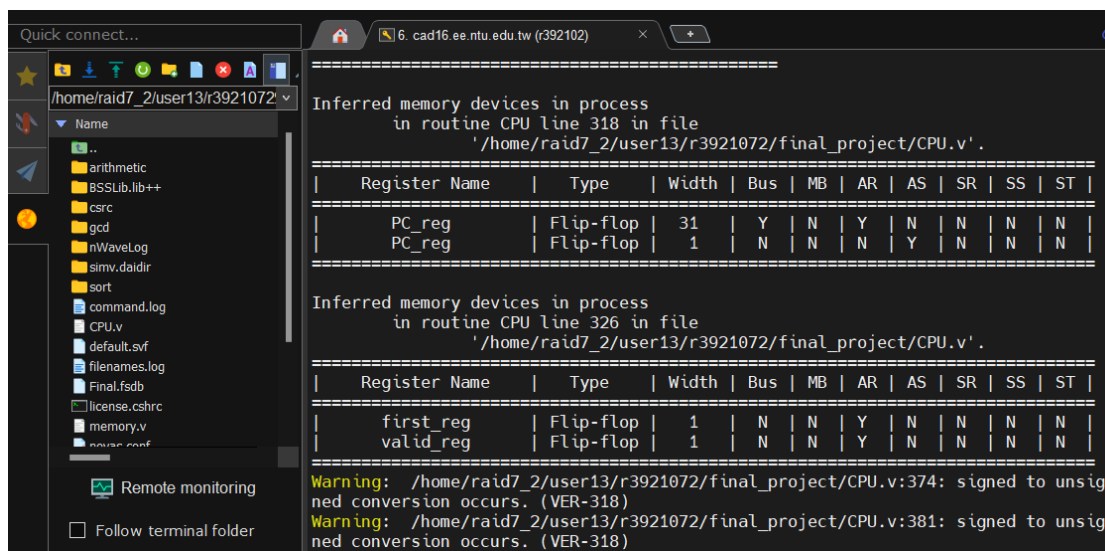
我是先提取 opcode 跟 funct3 來分出是那些指令，然後再根據 Greencard 的描述來對每個指令做它應該做的動作，值得注意的是在 R-type 指令中需要而外考慮 funct7 的部分才能順利分辨出到底是 add 還是 mul 等等。

對於一些比較特殊的指令，就要仔細看 Greencard 怎麼寫的，像是 jal、jalr 指令加的 offset 要另外換成有號數來拓展到 32bits，至於像 auipc 比較特殊，是把 imm 當作最高的 20 位，所以我是把資料抓下來後先把他邏輯左移 12 個位元，lui 指令也是類似的操作，只差在前者要另外加上 PC 位置而已。

Multi cycle 控制

由於規定在做乘除法時要利用多週期的運算方式，因此需要讓 PC 的位置滯留在原點直到運算結束，在這裡我利用 ready 來控制，當運算結束時讓乘法器回傳 ready = 1 才會進一步更新 PC_nxt 的位置，還要注意的是 valid 只在這個指令第一次出現時給他，滯留過程要令 valid = 0，所以我在主模型上另外設一個變數 first 來進行控制並判斷這個指令是不是第一次做運算。

暫存器合成結果



```
Quick connect...
/home/raid7_2/user13/r3921072
Name
arithmetic
BSSLib.lib++
csrc
gcd
nWaveLog
simv.daidir
sort
command.log
CPU.v
default.svf
filenames.log
Final.fsd
license.cshrc
memory.v
...
Remote monitoring
Follow terminal folder

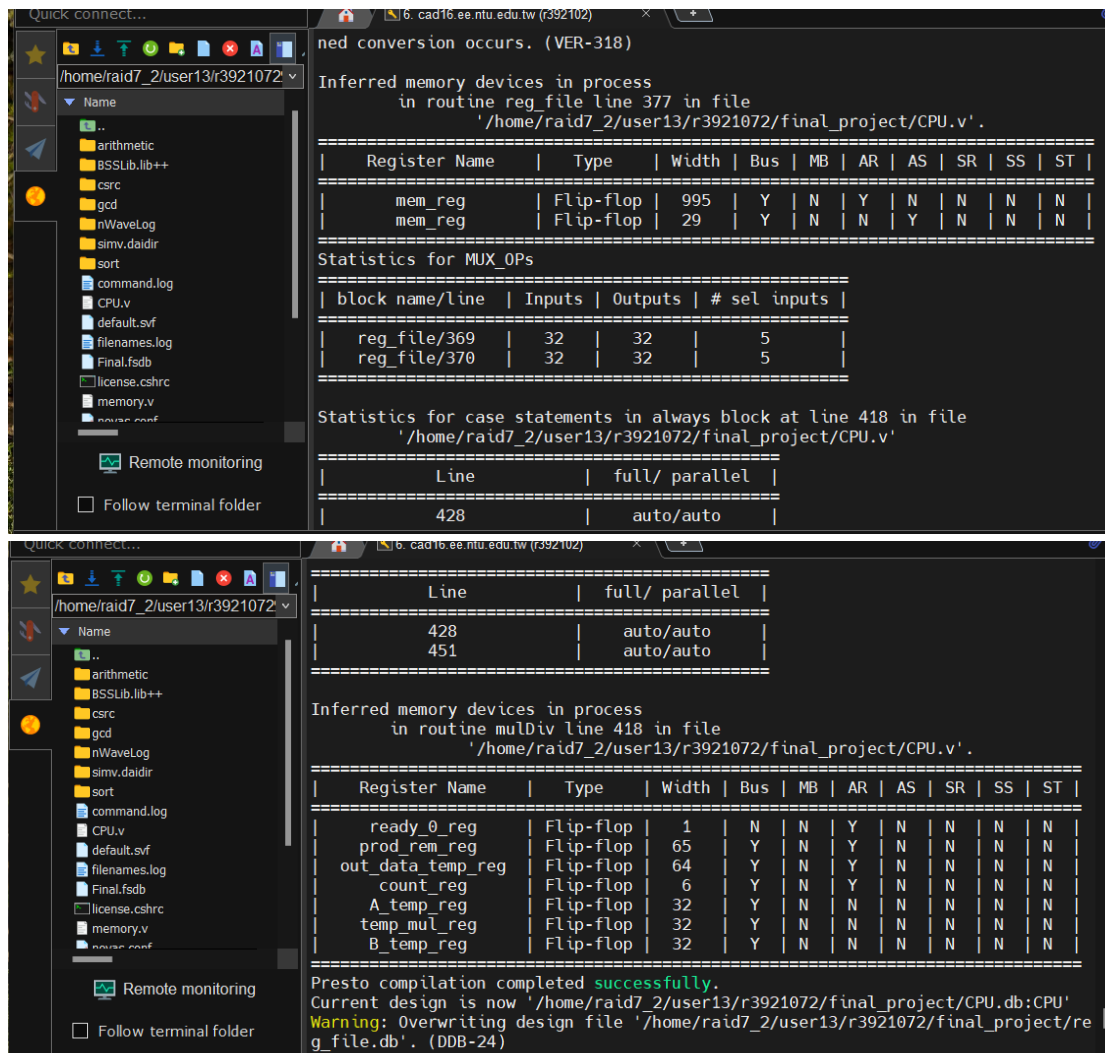
Inferred memory devices in process
in routine CPU line 318 in file
'/home/raid7_2/user13/r3921072/final_project/CPU.v'.

=====
| Register Name | Type | Width | Bus | MB | AR | AS | SR | SS | ST |
=====
| PC_reg | Flip-flop | 31 | Y | N | Y | N | N | N | N |
| PC_reg | Flip-flop | 1 | N | N | N | Y | N | N | N |
=====

Inferred memory devices in process
in routine CPU line 326 in file
'/home/raid7_2/user13/r3921072/final_project/CPU.v'.

=====
| Register Name | Type | Width | Bus | MB | AR | AS | SR | SS | ST |
=====
| first_reg | Flip-flop | 1 | N | N | Y | N | N | N | N |
| valid_reg | Flip-flop | 1 | N | N | Y | N | N | N | N |
=====

Warning: /home/raid7_2/user13/r3921072/final_project/CPU.v:374: signed to unsig
ned conversion occurs. (VER-318)
Warning: /home/raid7_2/user13/r3921072/final_project/CPU.v:381: signed to unsig
ned conversion occurs. (VER-318)
```



包含所有子模型的 Reg 都是 flip-flop 的型態

觀察與結論

這份作業整合了 CPU 內部的運作過程，可以觀察到 RISC-V 的小巧思，如何用少少的 32 位元同時儲存指令加上操作的暫存器是有哪些，發現它的一些規定，先利用 opcode 分別出指令，再給予需要的指令更大的 imm (ex. J-type, U-type) 達到更好的效果。

除此之外這次用到很多的布林數來控制一些關鍵的作用，像是需不需要寫入記憶體或是 `rd_data` 中，讓我對整個解碼流程有更多的瞭解。