**Final-project Report**

**R13921072 何家祥**

**整體CPU框架**

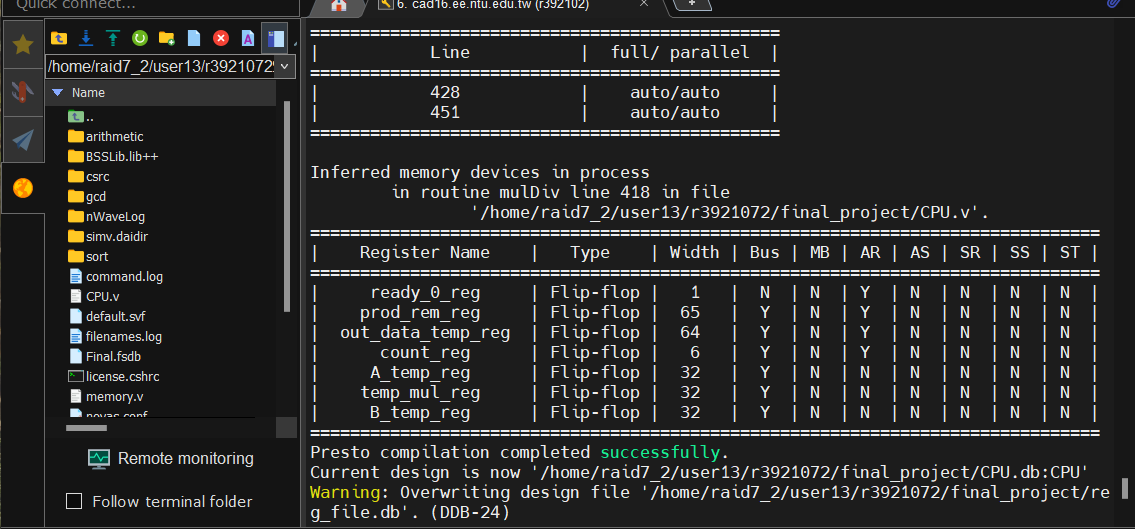
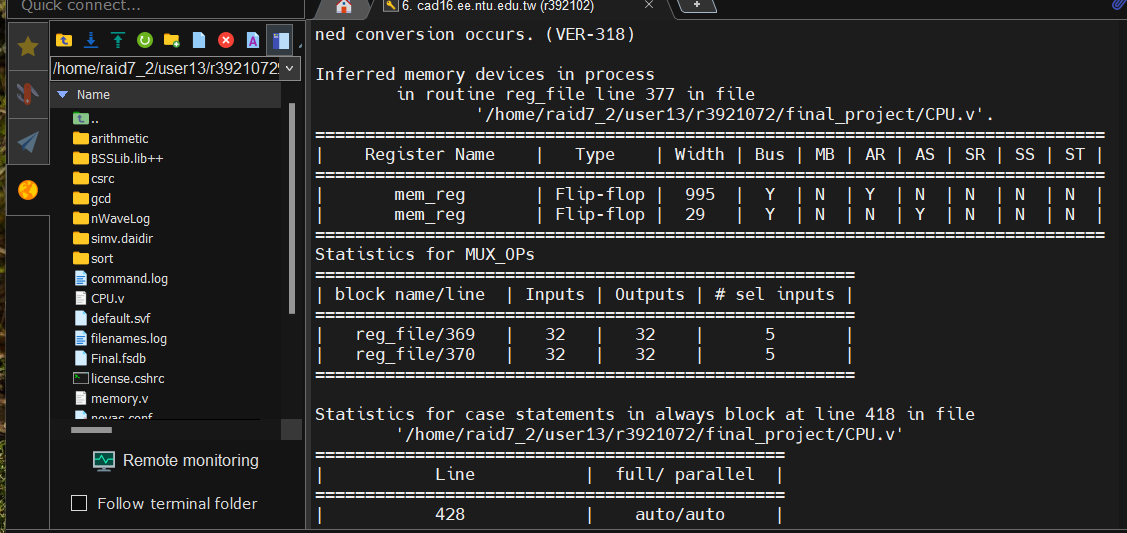
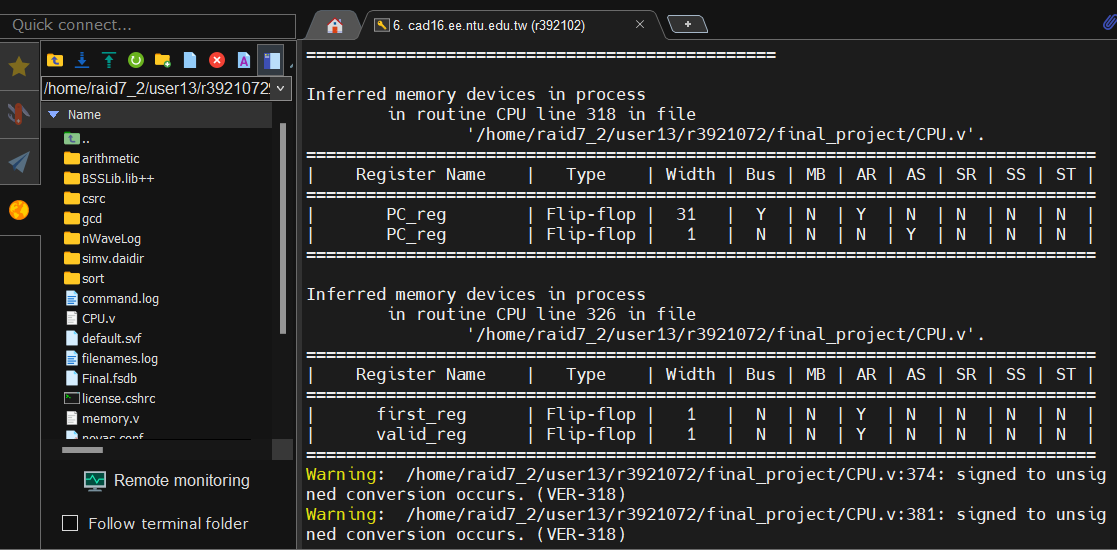
我是先提取opcode跟funct3來分出是那些指令，然後再根據Greencard的描述來對每個指令做它應該做的動作，值得注意的是在R-type指令中需要而外考慮funct7的部分才能順利分辨出到底是add還是mul等等。

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**

****

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。