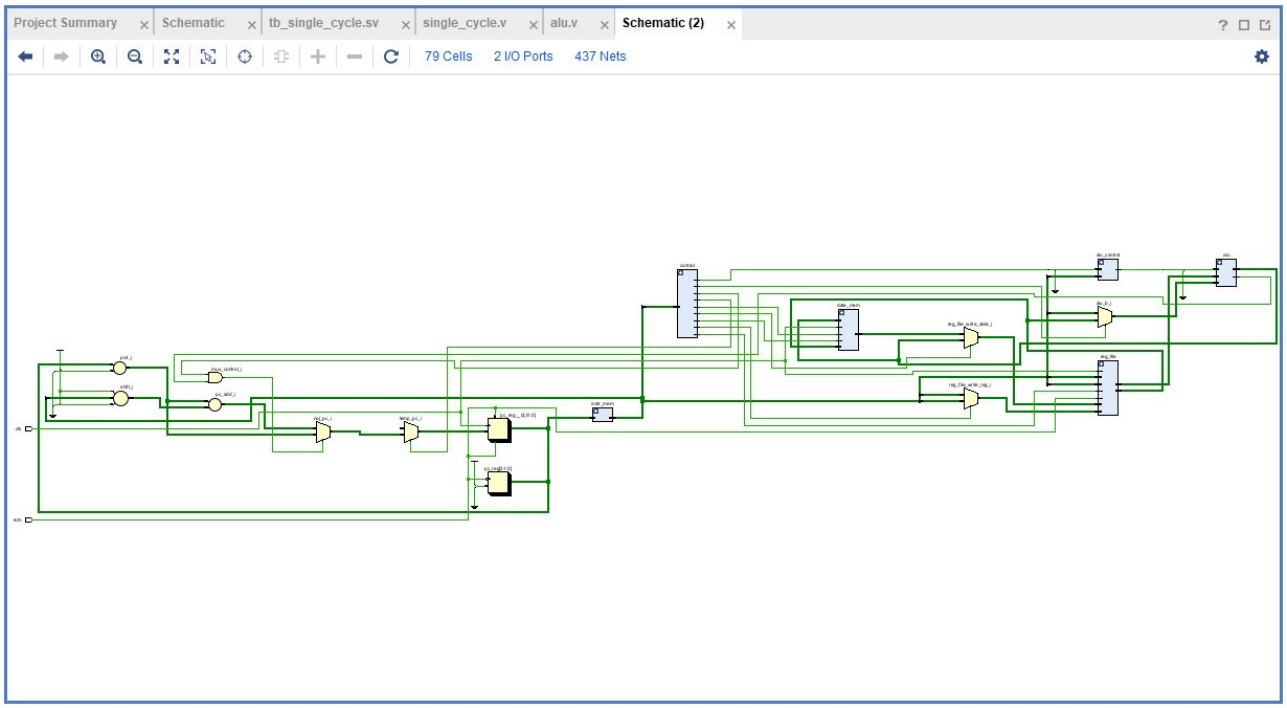
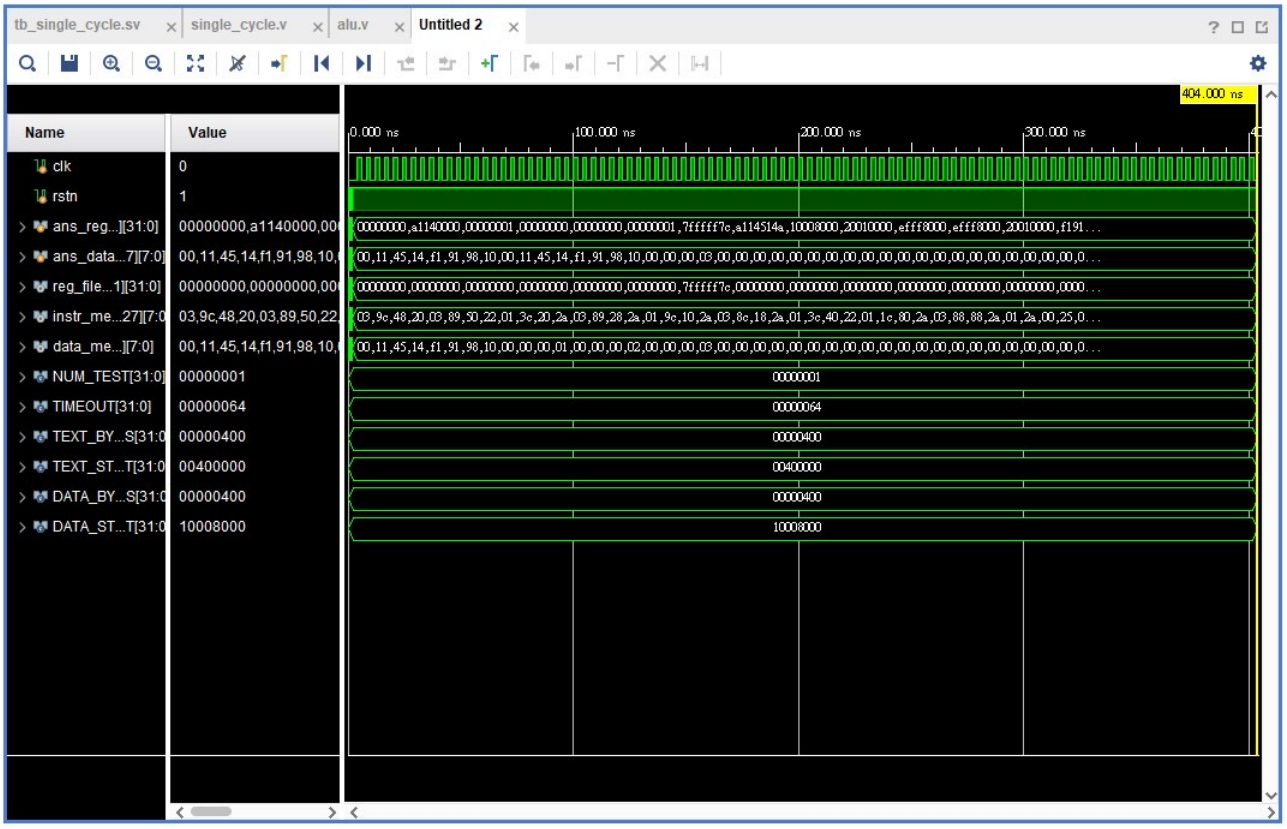
Lab2 Report

1. **Architecture Diagrams** –> 參考老師上課講義Ch4的p24,29的電路圖形進行拉線



2. **Experimental Result**
   1. **Screen shot of test bench**

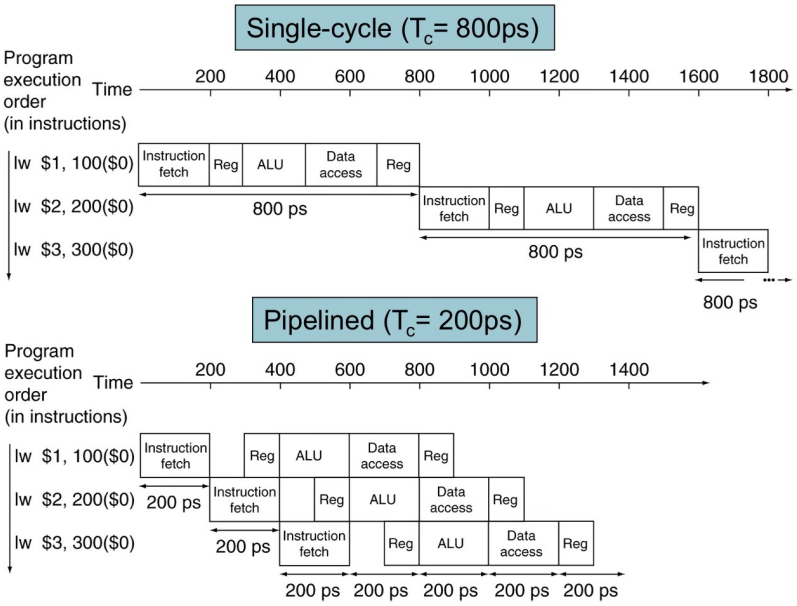


2. 因為第一次的Alu實作中我的slt有出現問題，所以有另外將ALU拉出來測試是否訂正正確，測試slt與slt overflow時的情形。

3. **Answer the following questions**
   1. **When does write to register/memory happen during the clock cycle? How about read?**
      ① Write –> 通常發生在clock cycle的 rising edge 或 falling edge。
      ② Read –> 可以在時鐘周期的任何階段發生，通常在寫入之後的下一個時鐘周期。
   2. **Translate the "branch" pseudo instructions (blt, bgt, ble, bge) in the Green Card into real instructions. Only at register can be modified, and other common registers should not be modified.**
      ① blt $s, $t, label → slt $at, $s, $t; bne $at, $0, label
      ② bgt $s, $t, label → sub $at, $s, $t; slt $0, $at, label
      ③ ble $s, $t, label → slt $at, $t, $s; beq $at, $0, label; beq $t, $s, label
      ④ bge $s, $t, label → slt $at, $t, $s; bne $at, $0, label
   3. **Give a single beq assembly instruction that causes infinite loop. (consider that there's no delay slot)**
      –> CHECK: beq $0, $0, CHECK
   4. **The j instruction can only jump to instructions within the "block" defined by "(PC+4)[31:28]*. Design a method to allow j to jump to the next block (block number + 1) using another j.**
      –> original_block_address <= (A) * 2^28 –4
      j original_block_address

      original_block_address:
      next_block_address <= (A+1) * 2^28 –4
      j next_block_address

      next_block_address:
      執行指令
   5. **Why a Single-Cycle Implementation Is Not Used Today?**
      效率較低，就算一個指令不需要一個clock cycle就可以完成，在single-cycle中還是需要一個clock cycle去實現它，且通常需要大量的硬體資源來實現，每個指令都需要完整的控制單元、運算單元和存儲單元，這會導致資源的浪費。我們可以簡單地從single-cycle與pipelined的program execution order看出來。



4. Problems Encountered

因為這次的Lab是要沿用上次的ALU，但上次的ALU中我的slt是錯誤的，也因此在一開始的時候花了很長的一段時間debug，甚至還call out同學跟我一起研究。

5. Feedback

希望在每次的Lab之後都可以公佈當次Lab的正確解答，讓我們參考不一樣的思考邏輯，同時也讓我們知道自己的錯誤可能是因為沒有考量到什麼東西，避免真的需要使用到上一次的Lab code但卻找不出自己上次的錯誤的問題。

另外上次Lab的slt overflow的評分，如果是想要同學實現overflow只會在輸入訊號要求我們實現add/sub的時候才考慮overflow的情況的話，希望能夠在slt overflow的部測試分只要overflow都=0就可以從寬給分，可以讓有考慮到只在某些情況下檢測overflow但slt寫錯的同學也獲得部分分數。

總而言之辛苦助教了，每次在teams上發問都可以很快的得到助教的解答！謝謝助教！