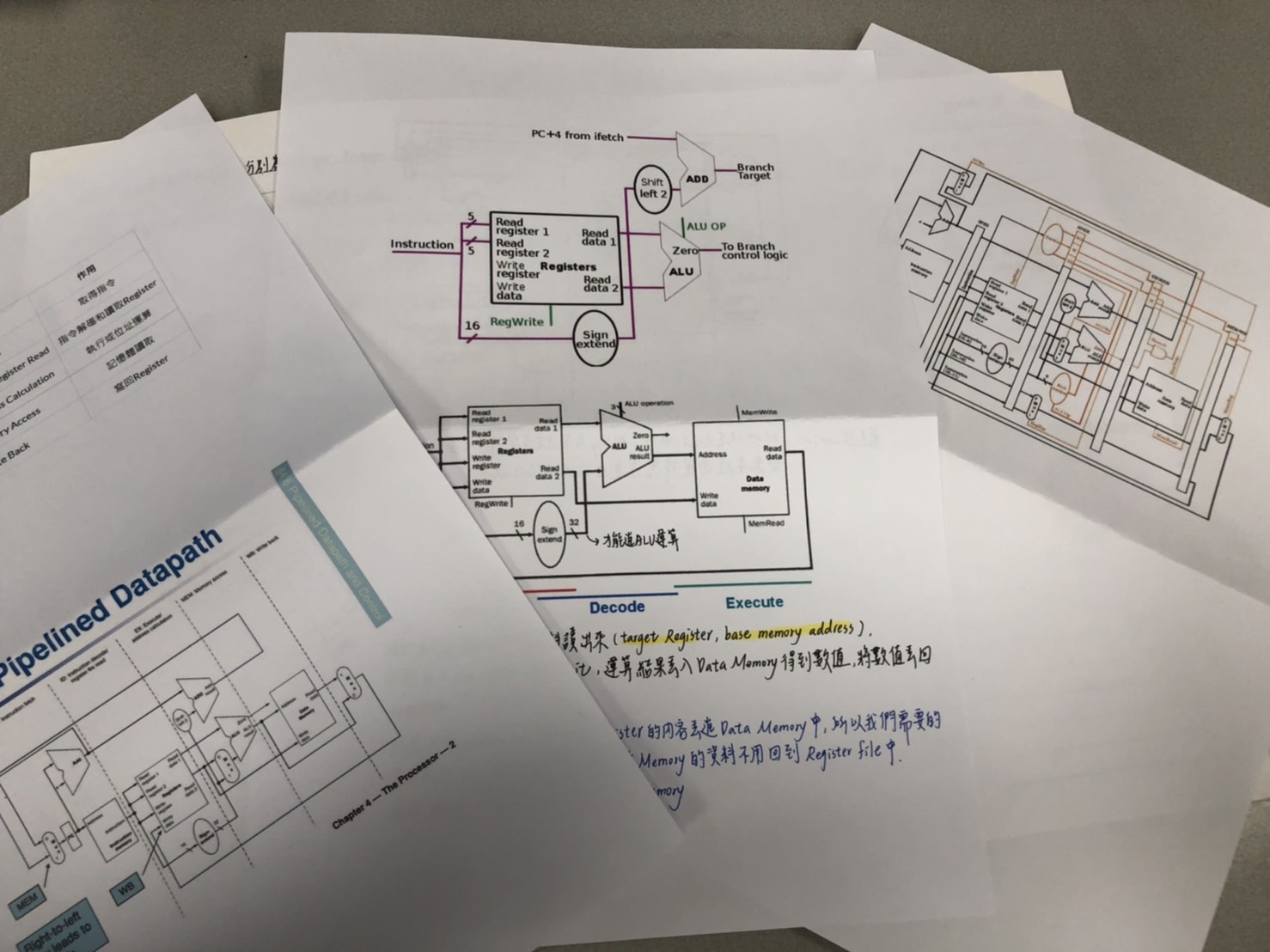
FPGA 系統設計實務

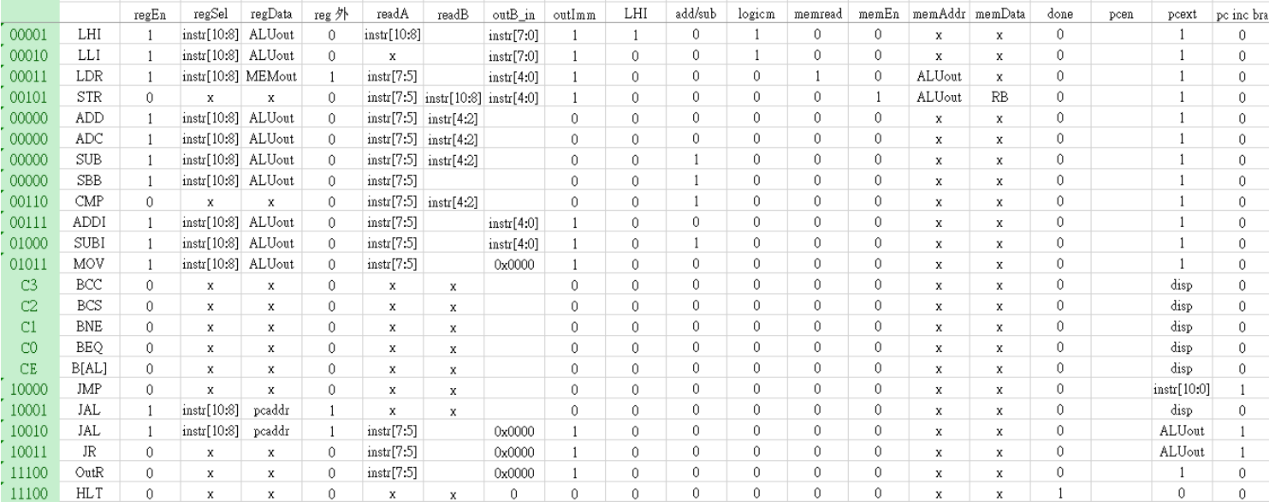
期中報告

M11102150

劉家瑜

Design

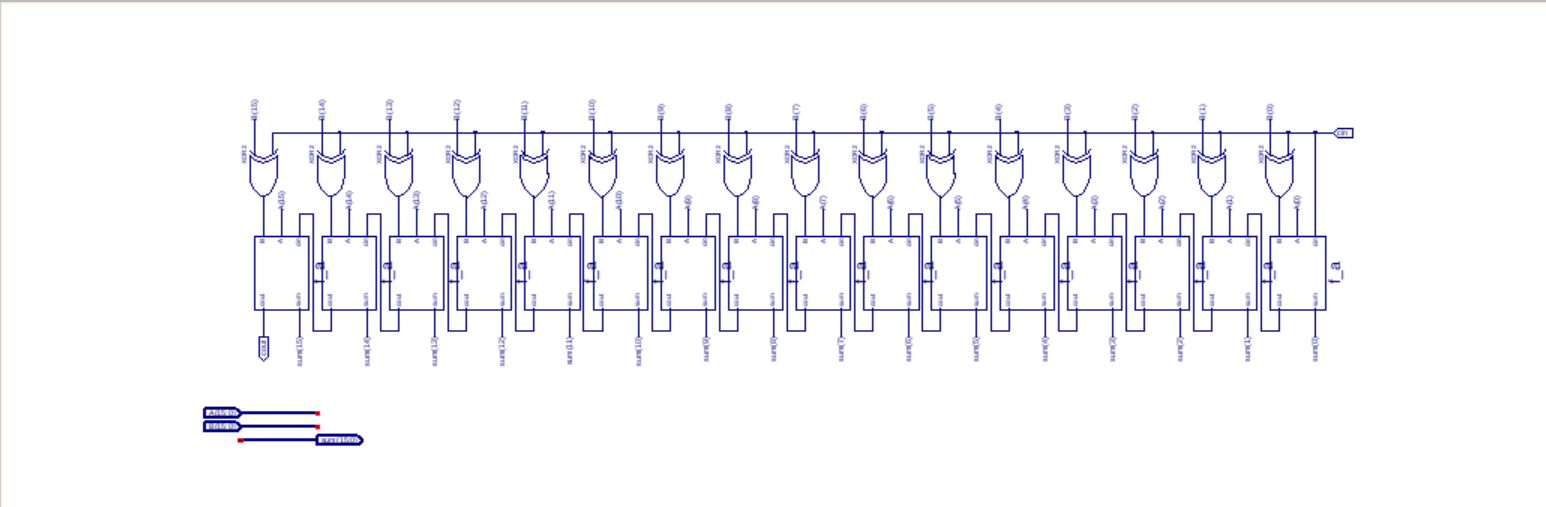




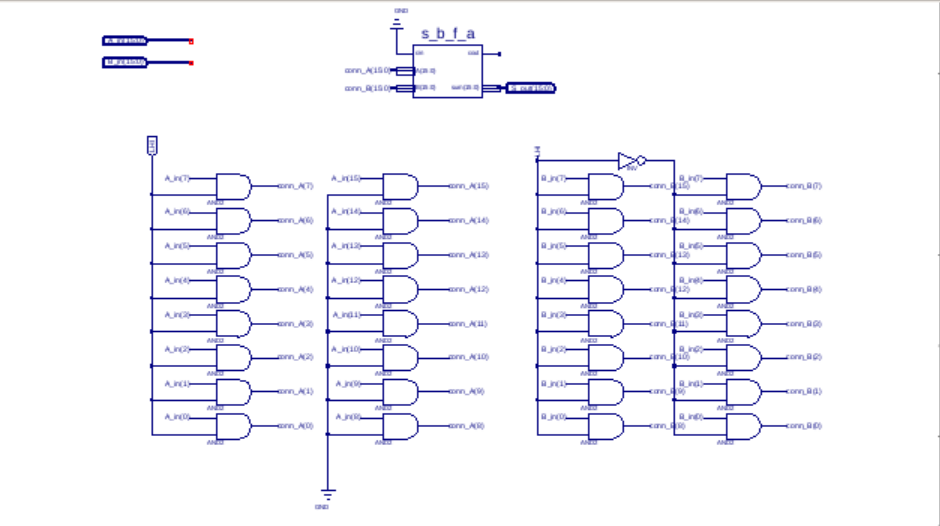
設計部分，一開始選這堂課的時候，就知道這堂課需要具備計算機組織與數位邏輯還有寫硬體描述語言的能力，因此架構的部分除了上網查之外，也上了清大OCW的課程，所以在開始拉電路之前，資料該怎麼流與怎麼控制，已有初步的想法與規劃。

Schematic

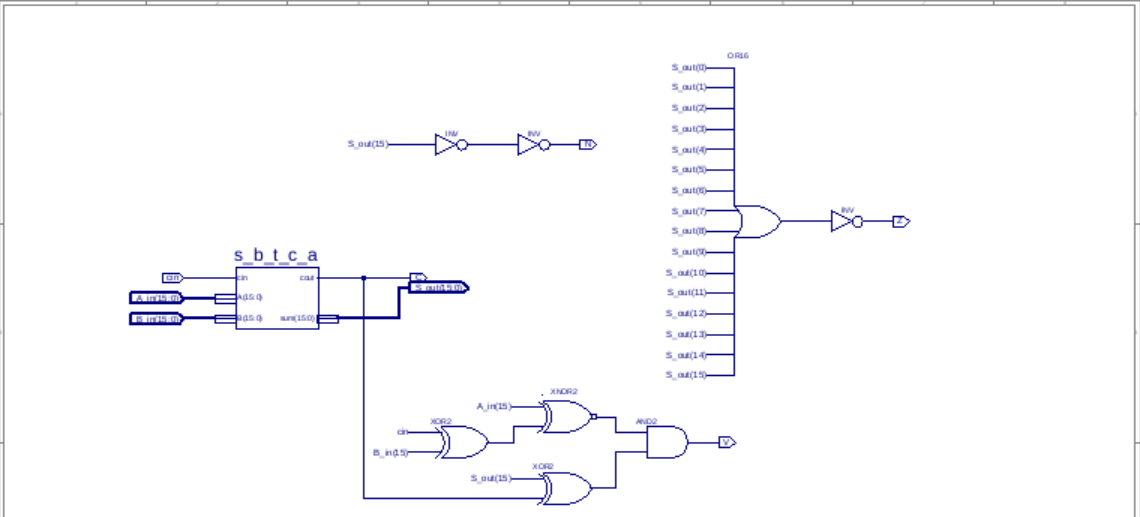
1.16-Bit Complement Adder



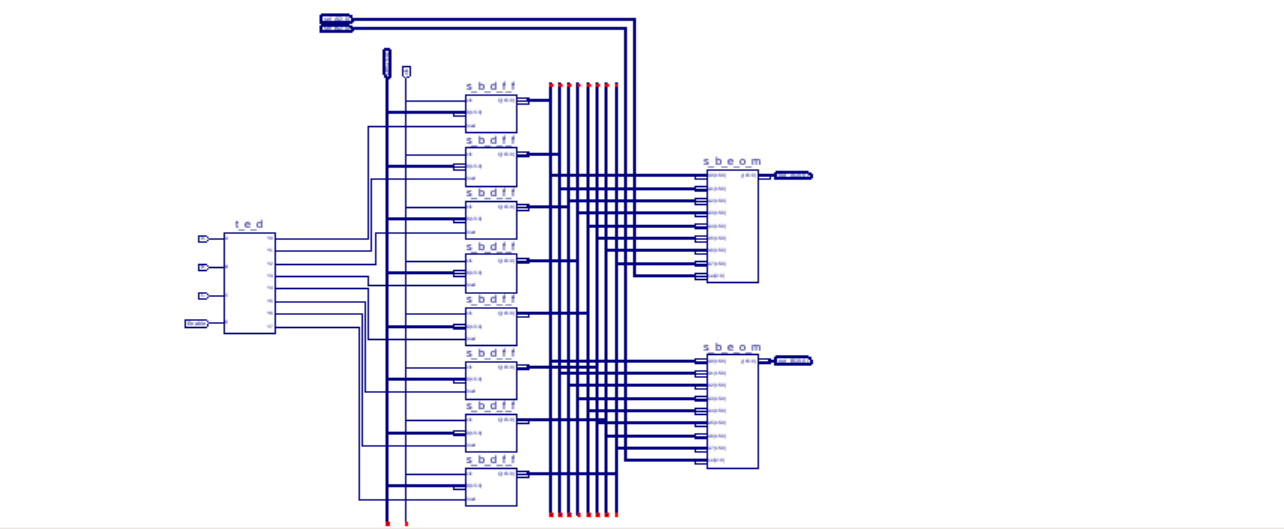
2.LHI\_LLI(用來處理LHI和LLI指令)



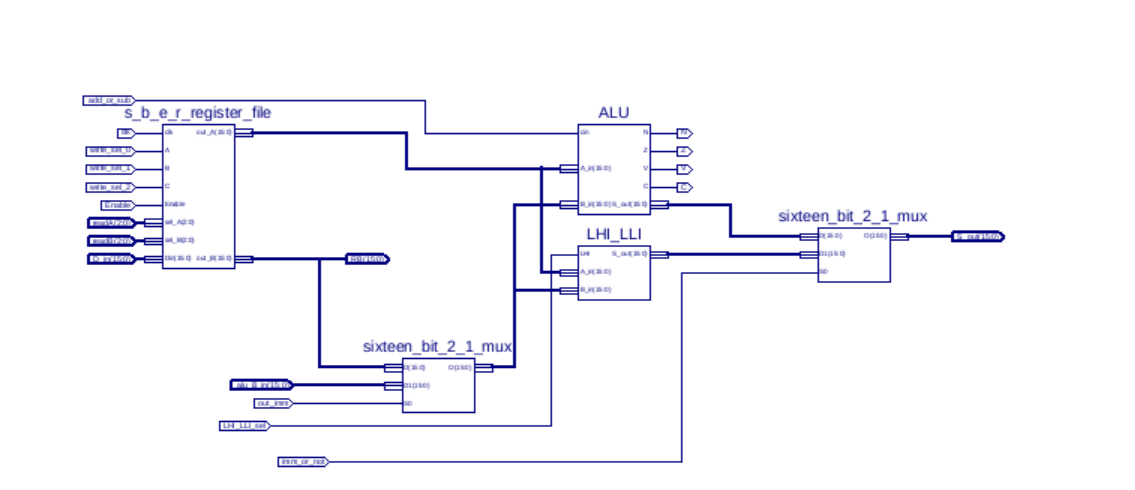
3.ALU



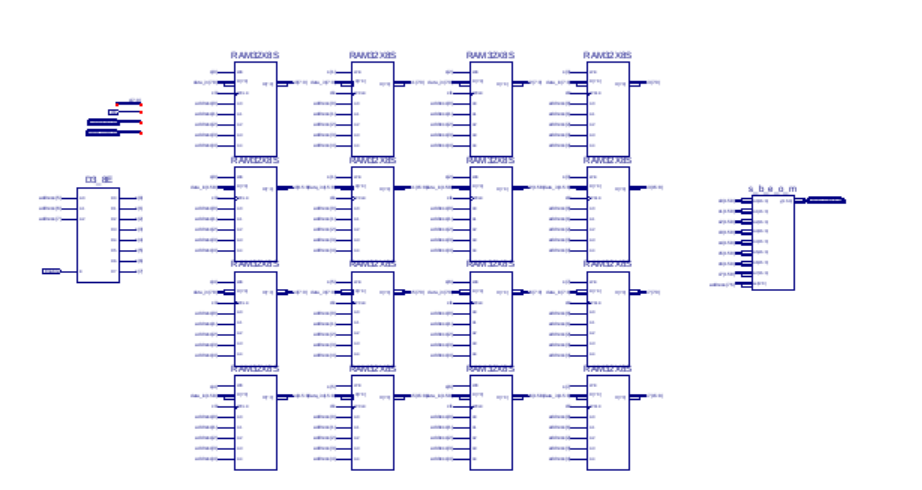
4.16-Bit Eight-Register Register File



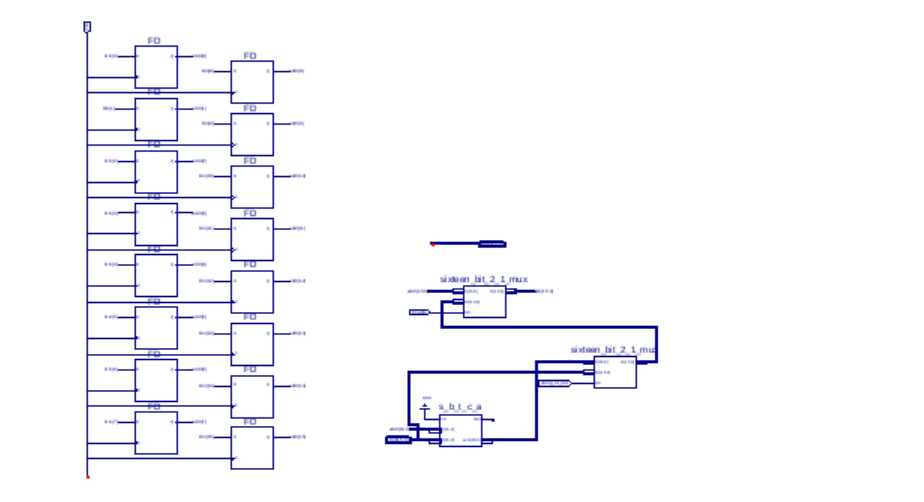
5.16-Bit RF-plus-ALU



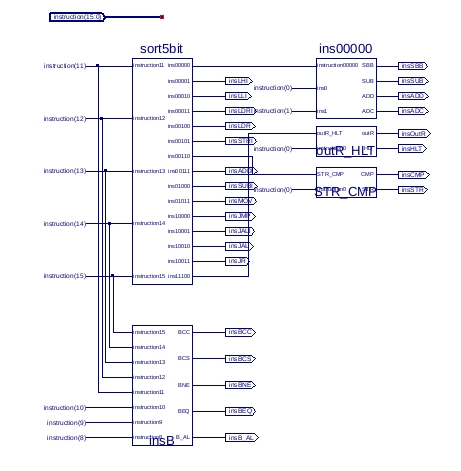
6.256\*16 Memory Module



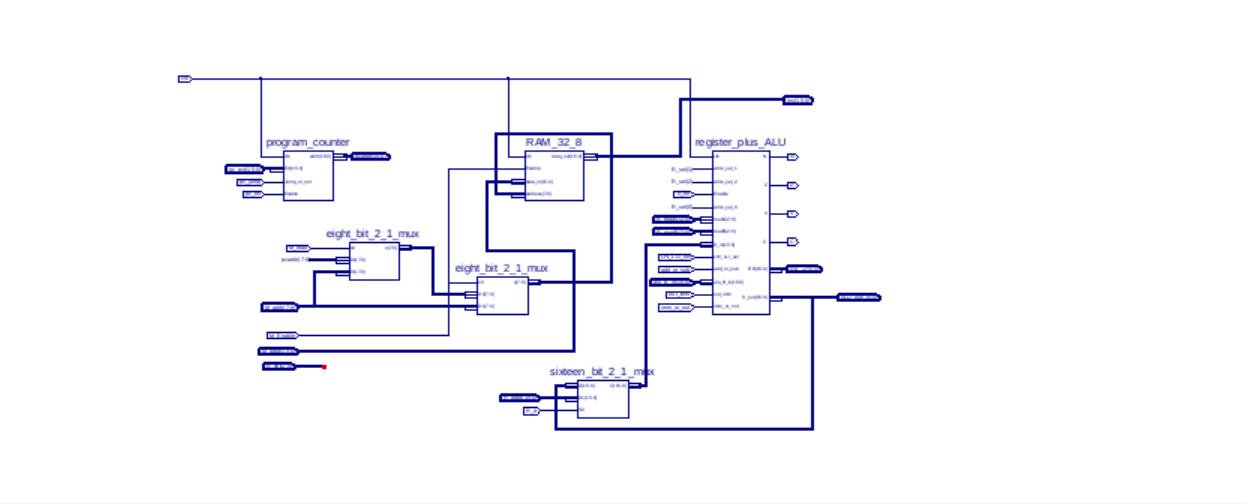
7.Program Counter



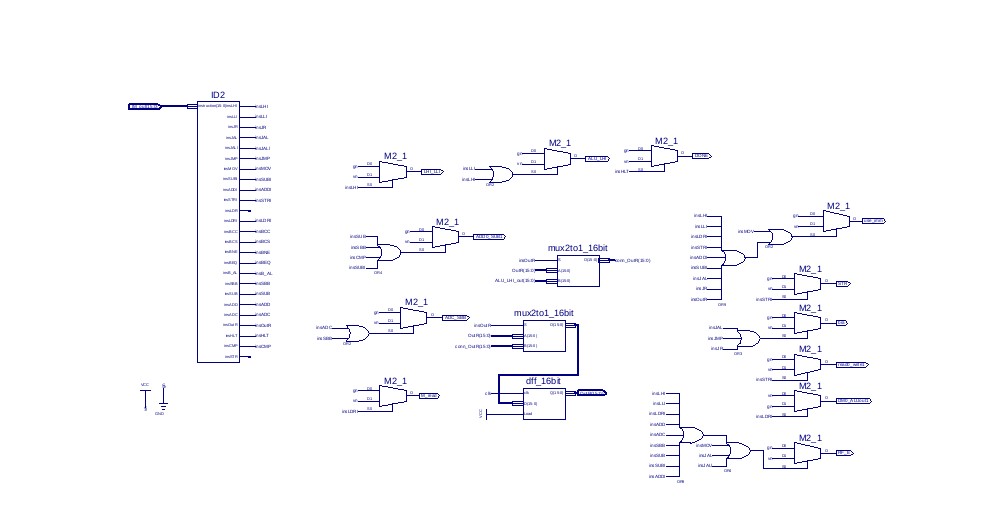
8.Instruction Decoder



9.Datapath



10.Controller(partial)

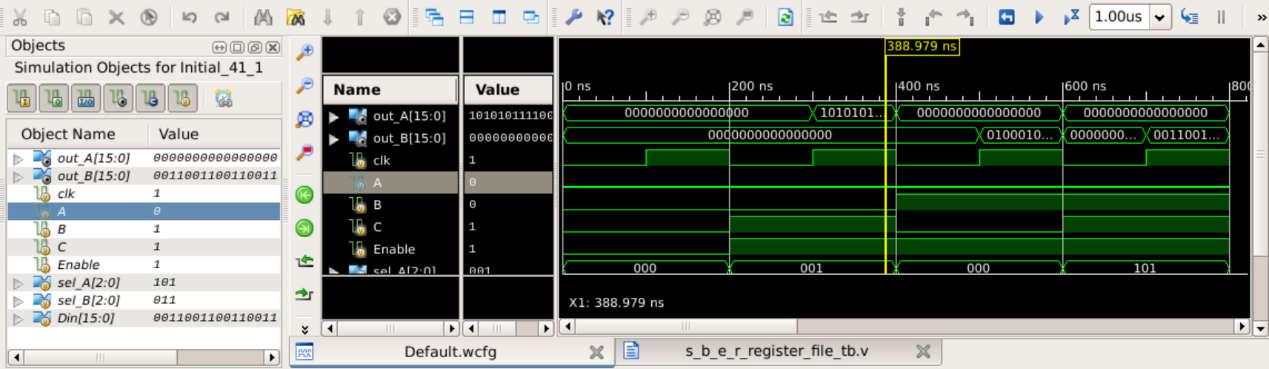


11.Complete Computer

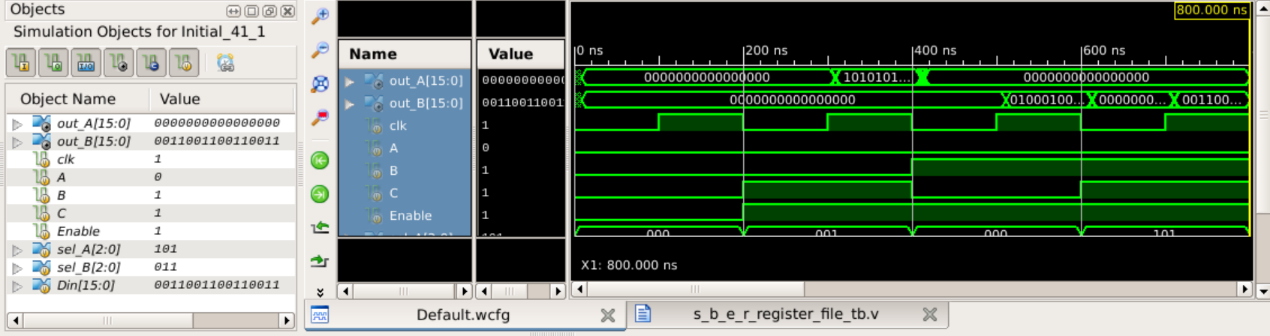


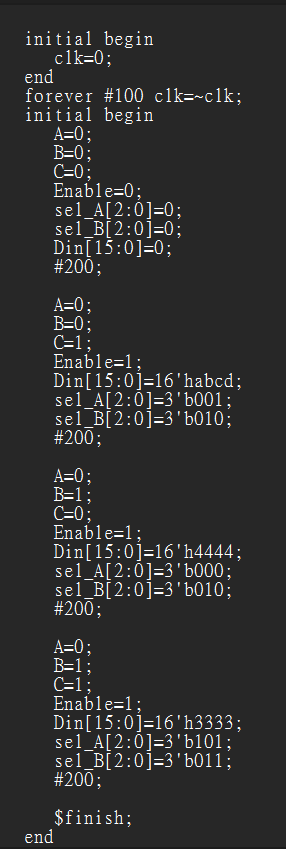
Verification

1.16-Bit Eight-Register Register File Test Bench

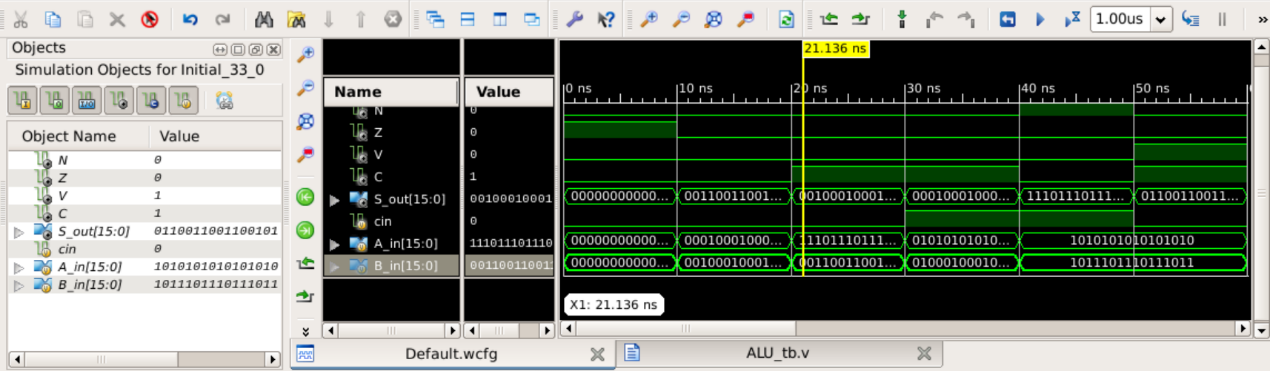


Post-Route

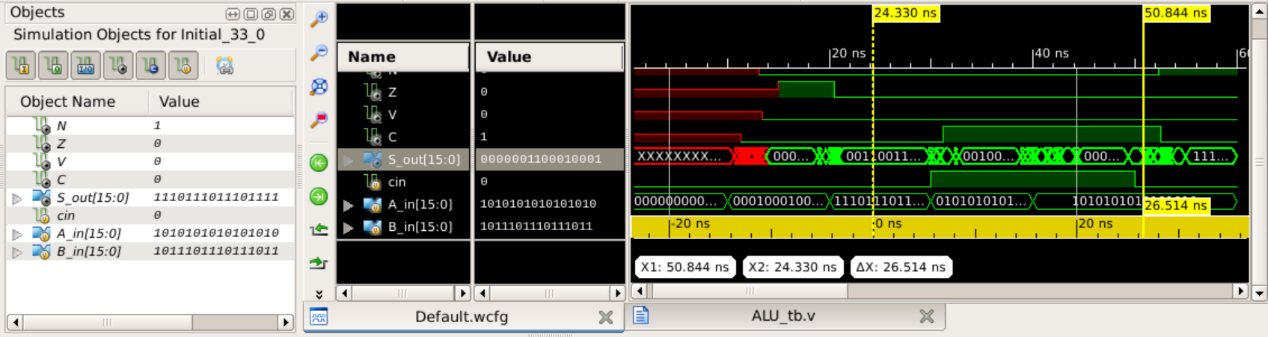


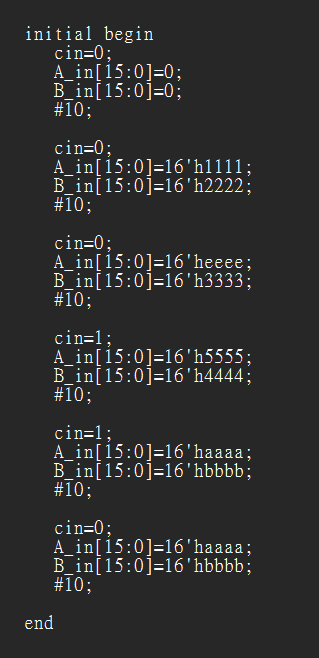


2.16-bit ALU Test Bench

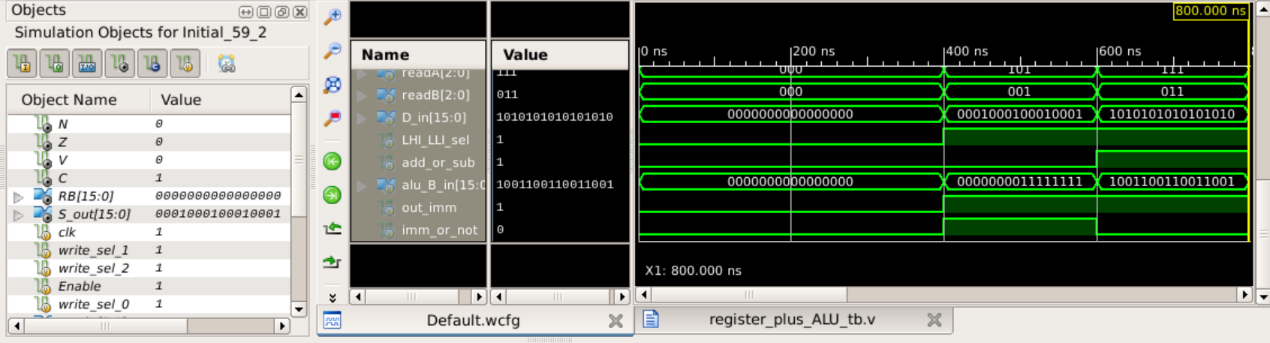


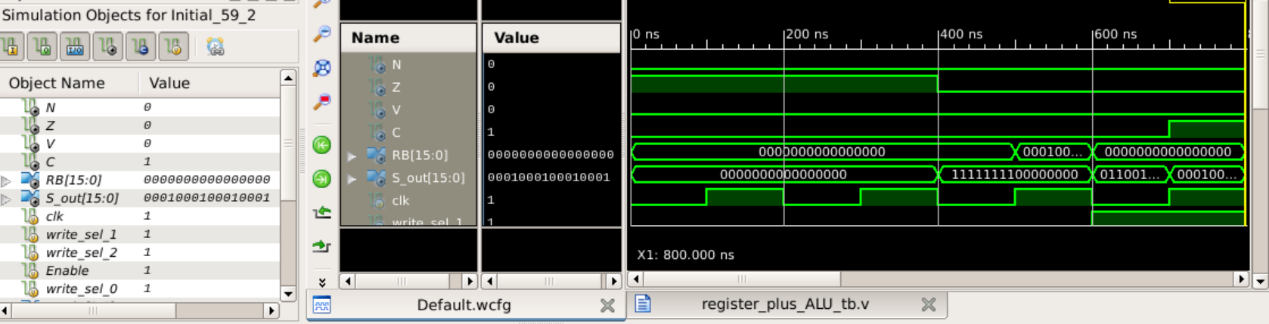
Post-Route



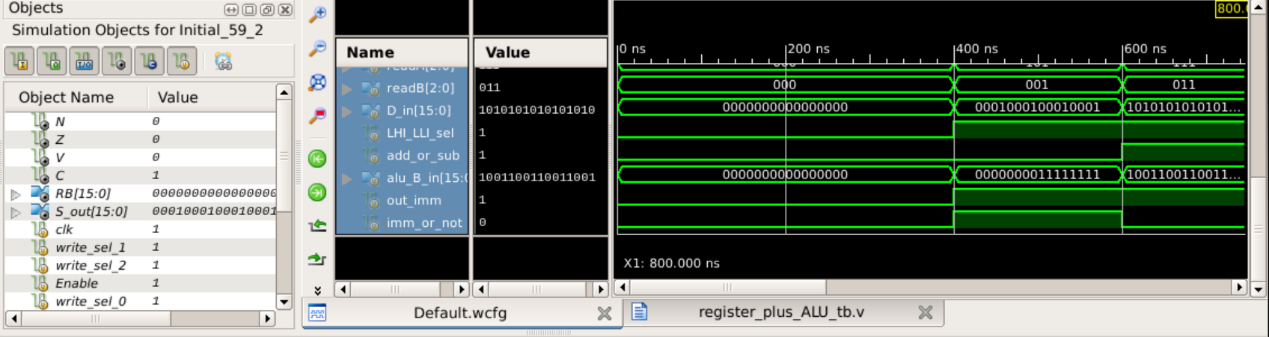


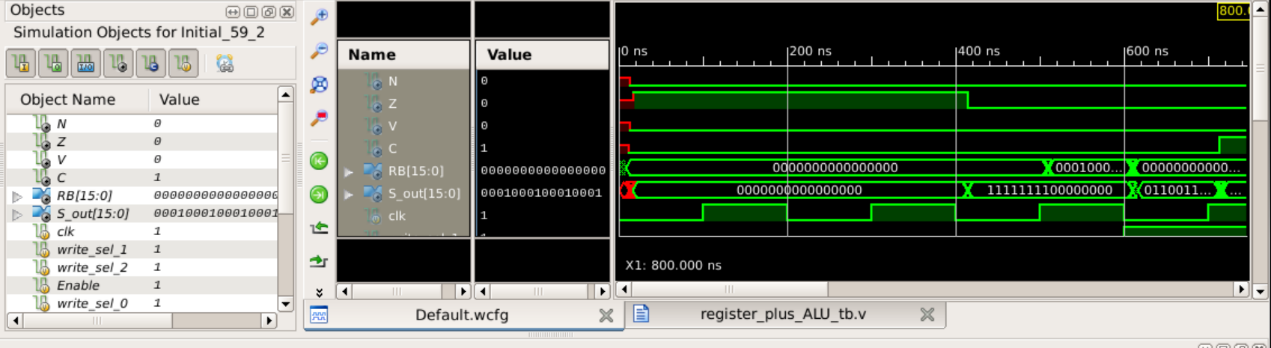
3.16-Bit RF-plus-ALU Test Bench



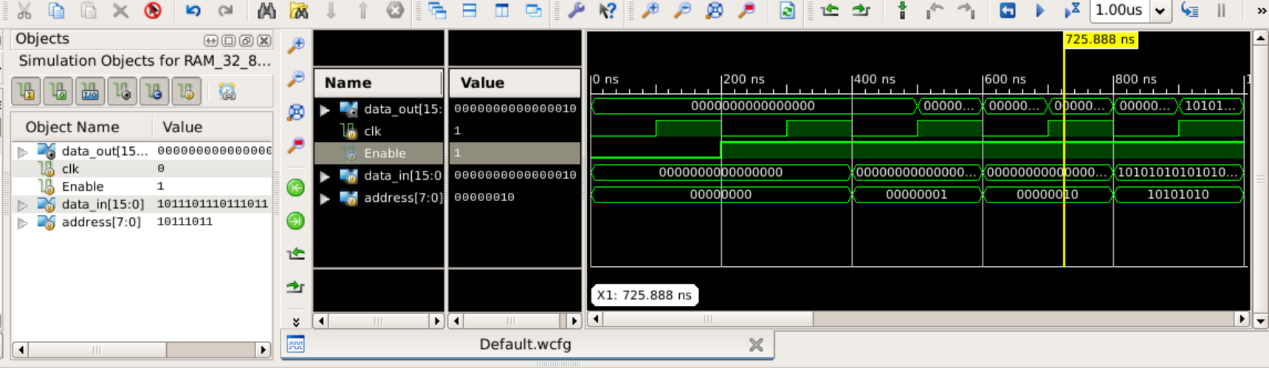


Post-Route

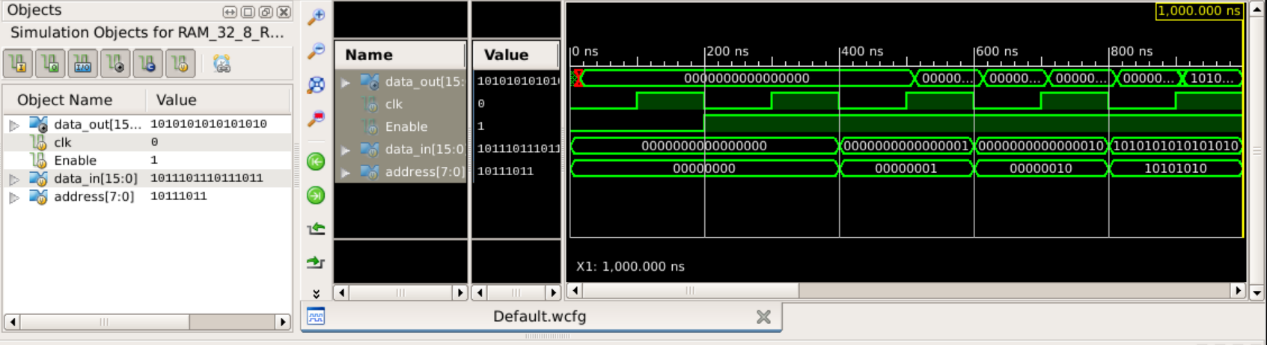




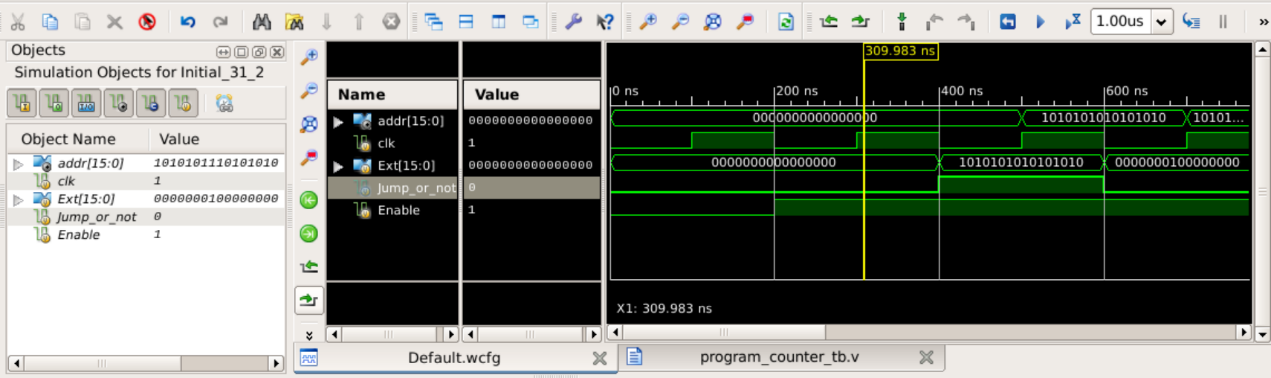
4.256\*16 Memory Module Test Bench



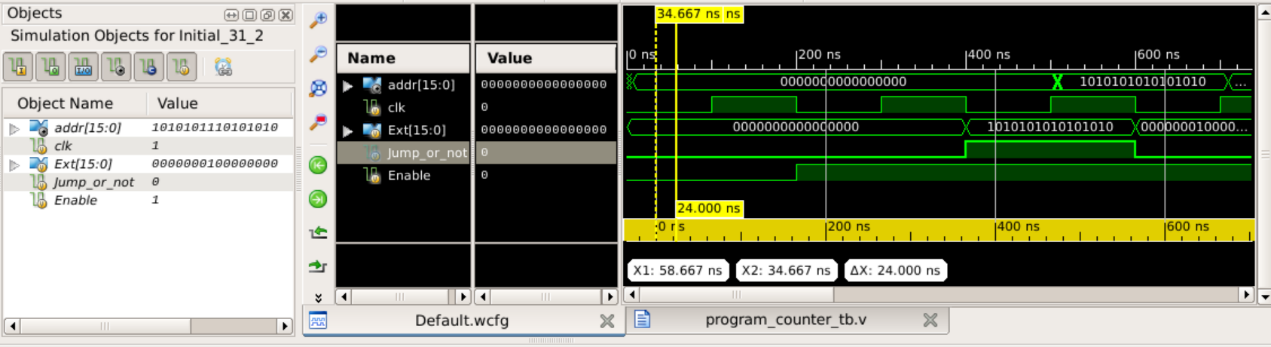
Post-Route



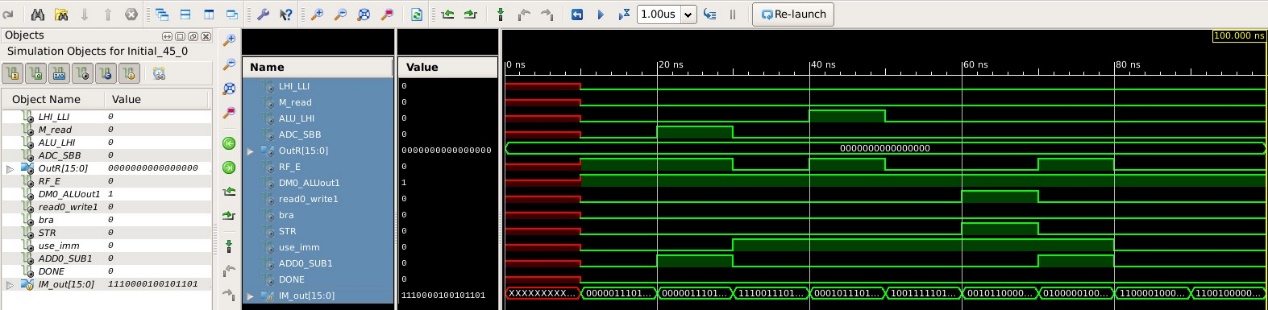
5.Program Counter Test Bench



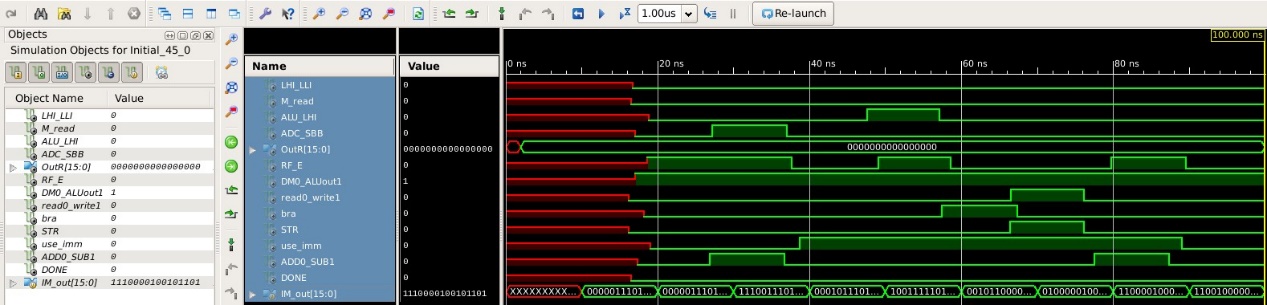
Post-Route



6.Controller Test Bench



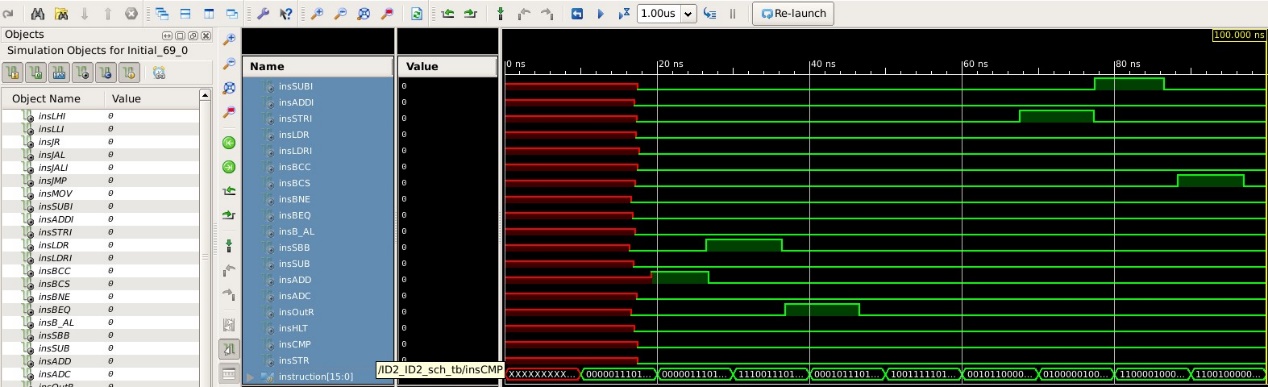
Post-Route



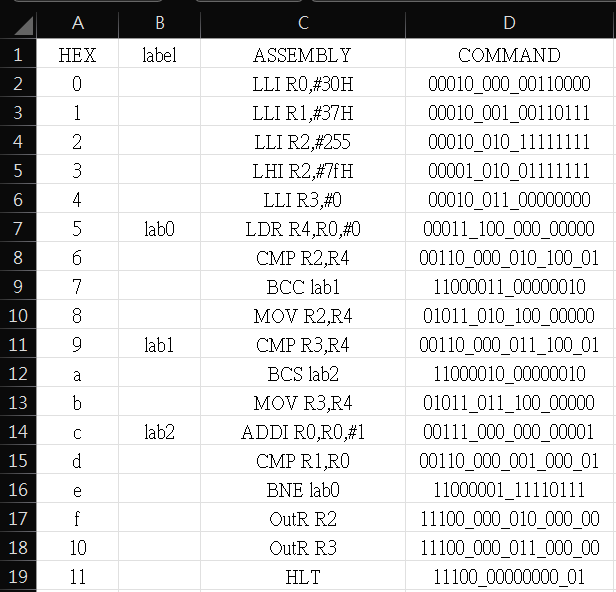
7.Instruction Decoder Test Bench



Post-Route



1. Find the minimum and maximum from two numbers in memory



write\_mem(16'h0,16'b00010\_000\_00110000 ) ;//LLI R0,#30H

write\_mem(16'h1,16'b00010\_001\_00110111 ) ; //LLI R1,#37H

write\_mem(16'h3,16'b00010\_010\_11111111 ) ; //LLI R2,#255

write\_mem(16'h4,16'b00001\_010\_01111111 ) ; //LHI R2,#7fH

write\_mem(16'h5,16'b00010\_011\_00000000 ) ; //LLI R3,#0

write\_mem(16'h6,16'b00011\_100\_000\_00000 ) ; //LDR R4,R0,#0

write\_mem(16'h7,16'b00110\_000\_010\_100\_01 ) ; //CMP R2,R4

write\_mem(16'h8,16'b11000011\_00000010 ) ; //BCC labl

write\_mem(16'h9,16'b01011\_010\_100\_00000 ) ; //MOV R2,R4

write\_mem(16'hA,16'b00110\_000\_011\_100\_01 ) ; //CMP R3,R4 write\_mem(16'hA,16'b11000010\_00000010 ) ; //BCS lab2 write\_mem(16'hB,16'b01011\_011\_100\_00000 ) ; //MOV R3,R4

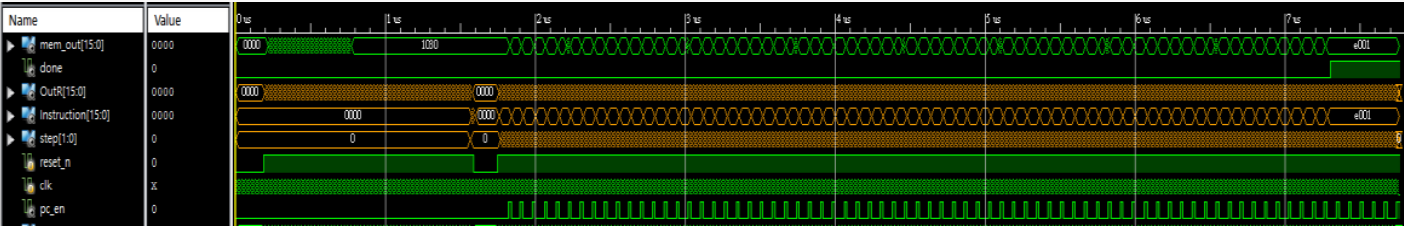
write\_mem(16'hB,16'b00111\_000\_000\_00001 ) ; //ADDI R0,R0,#1 write\_mem(16'hB,16'b00110\_000\_001\_000\_01 ) ; //CMP R1,R0

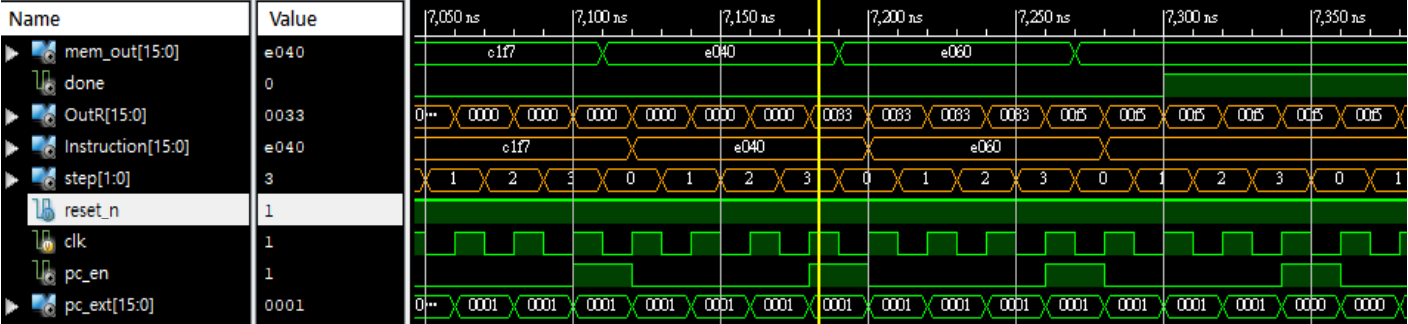
write\_mem(16'hB,16'b11000001\_11110111 ) ; //BNE lab0

write\_mem(16'hB,16'b11100\_000\_010\_000\_00 ) ; //OutR R2

write\_mem(16'hB,16'b11100\_000\_011\_000\_00 ) ; //OutR R3

write\_mem(16'hB,16'b11100\_000000000\_01 ) ; //HLT





1. Add two numbers in memory and store the result in memory location

write\_mem(16'h0,16'b00010\_000\_00110000 ) ;//LLI R0,#30H

write\_mem(16'h1,16'b00011\_011\_000\_00000 ) ; //LDR R3,R0,#0

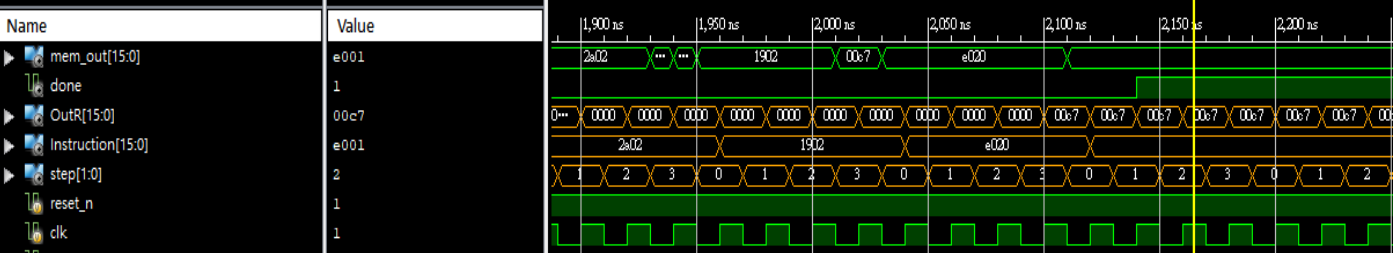
write\_mem(16'h2,16'b00011\_100\_000\_00001 ) ; //LDR R4,R0,#1

write\_mem(16'h3,16’b00000\_010\_011\_100\_00 ) ; //ADD R2,R3,R4 write\_mem(16'h4,16'b00101\_010\_000\_00010 ) ; //STR R4,R0,#0

write\_mem(16'h5,16'b00011\_001\_000\_00010 ) ; //LDR R1,R0,#2

write\_mem(16'h6,16'b11100\_000\_001\_000\_00 ) ; //OutR R1

write\_mem(16'h7,16'b11100\_000000000\_01 ) ; //HLT



1. Add ten numbers in consecutive memory locations

write\_mem(16'h0,16'b00010\_000\_00110000 ) ;//LLI R0,#30H

write\_mem(16'h1,16'b00010\_010\_00111001 ) ;//LLI R2,#39H

write\_mem(16'h2,16'b00011\_001\_000\_00000 ) ; //LDR R1,R0,#0

write\_mem(16'h3,16’b00111\_001\_001\_011\_00 ) ; //ADDI R0,R0,#1

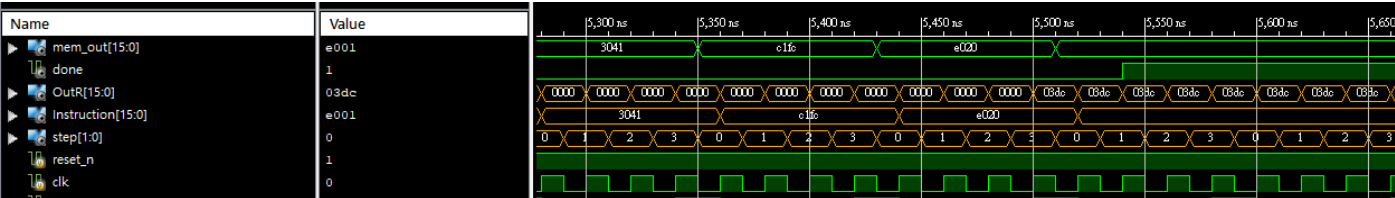
write\_mem(16'h4,16'b00011\_011\_000\_00000 ) ; //LDR R3,R0,#0 write\_mem(16'h5,16’b00000\_001\_001\_011\_00 ) ; //ADD R1,R1,R3

write\_mem(16'h6,16'b00110\_000\_010\_000\_01 ) ; //CMP R2,R0

write\_mem(16'h7,16'b11000001\_11110111 ) ; //BNE lab0

write\_mem(16'h8,16'b11100\_000\_001\_000\_00 ) ; //OutR R1

write\_mem(16'h9,16'b11100\_000000000\_01 ) ; //HLT



4. Mov a memory block of N words from one place to another

write\_mem(16'h0,16'b00010\_000\_00110000 ) ;//LLI R0,#30H

write\_mem(16'h1,16'b00010\_010\_00111001 ) ;//LLI R2,#39H

write\_mem(16'h2,16'b00011\_001\_000\_00000 ) ; //LDR R1,R0,#0

write\_mem(16'h3,16’b00111\_001\_001\_011\_00 ) ; //ADDI R0,R0,#1

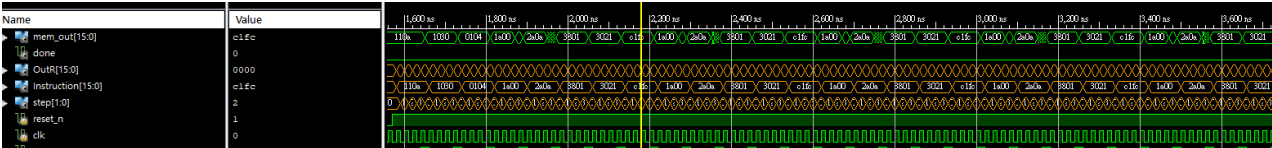
write\_mem(16'h4,16'b00011\_011\_000\_00000 ) ; //LDR R3,R0,#0 write\_mem(16'h5,16’b00000\_001\_001\_011\_00 ) ; //ADD R1,R1,R3

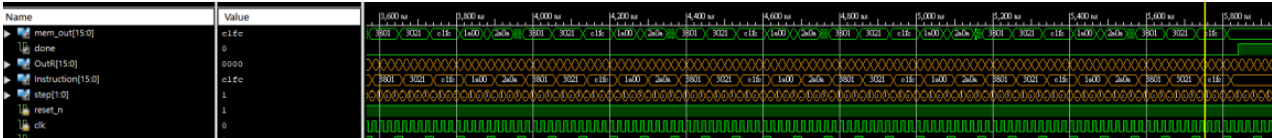
write\_mem(16'h6,16'b00110\_000\_010\_000\_01 ) ; //CMP R2,R0

write\_mem(16'h7,16'b11000001\_11110111 ) ; //BNE lab0

write\_mem(16'h8,16'b11100\_000\_001\_000\_00 ) ; //OutR R1

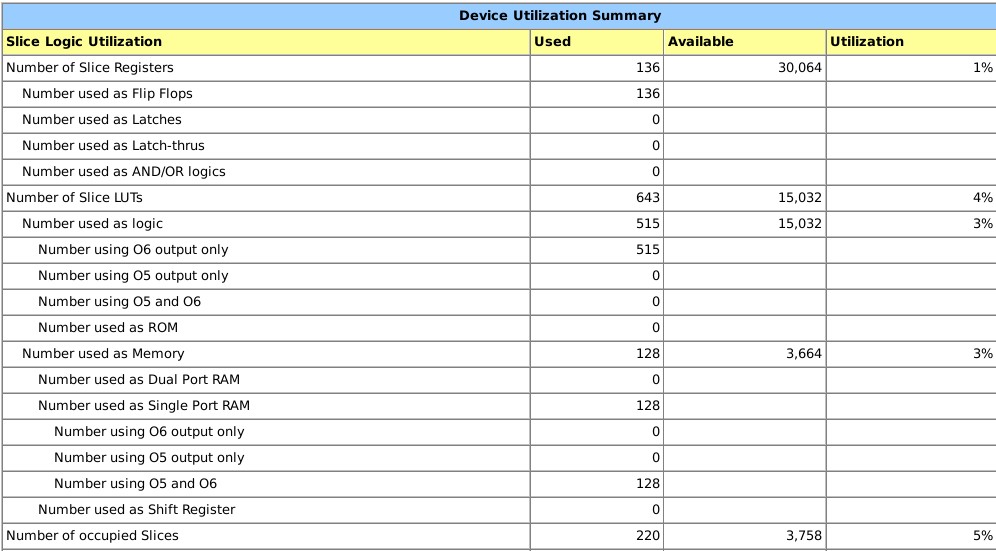
write\_mem(16'h9,16'b11100\_000000000\_01 ) ; //HLT





Hardware Cost

16-bits RISC CPU hardware cost



Total number of LUTs 643

Total number of FFs 136

Discussion

1. Design Entry

一開始我以為設計一個RISC CPU應該不難，因為網路上其實有很多類似的架構圖，但是再按圖施工的時候才發現，其實大部分有些東西都精簡過，所以如果沒先想好資料流動方式，其實是很容易卡關的。我花了很多時間在修改錯誤與理解每塊module的運作方式與功能，此外我大學的時候有學過數位邏輯和稍微寫過verilog，但是電路是第一次拉，一開始蠻好玩的，可是遇到挫折的時候，其實蠻需要與人討論的，不然真的要花超多時間研究。最重要的地方是我認為我對每塊module的功能與溝通還不夠熟悉，Term Project中也只有幾塊有很確切的解釋裡面該有哪些小模塊，其他就需要靠你對CPU架構的理解了。

1. Timing Simulation Error

另一方面就是我對時序的方面可能還不夠了解，我認為我的behavior simulation沒有太大的問題，但是在timing simulation 上好像有些許錯誤，但是我可能無法如期找出問題把它修改好，接下來還要準備期中考，希望我的期末Project可以把它改進並做得更好。