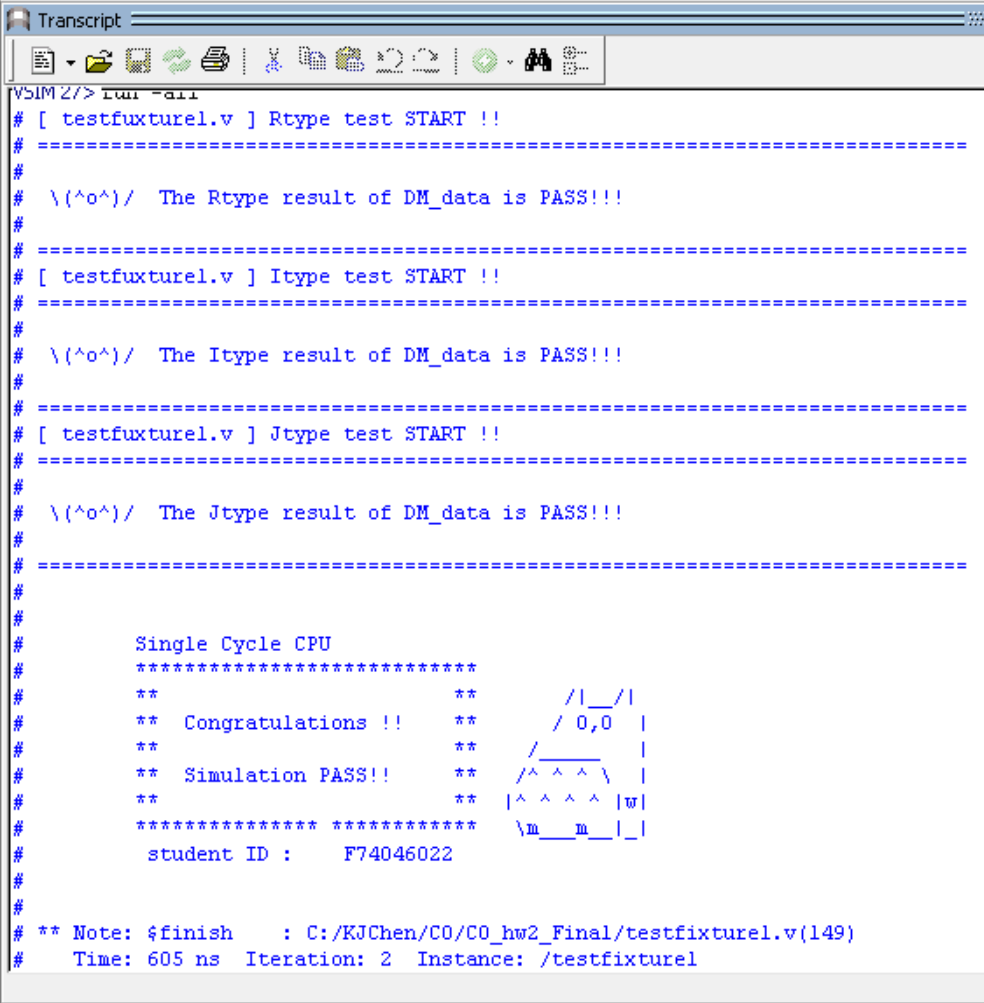
**Computer Organization 2017**

**HOMEWORK II**

系級: 資訊108 學號: F74046022 姓名: 陳冠仁

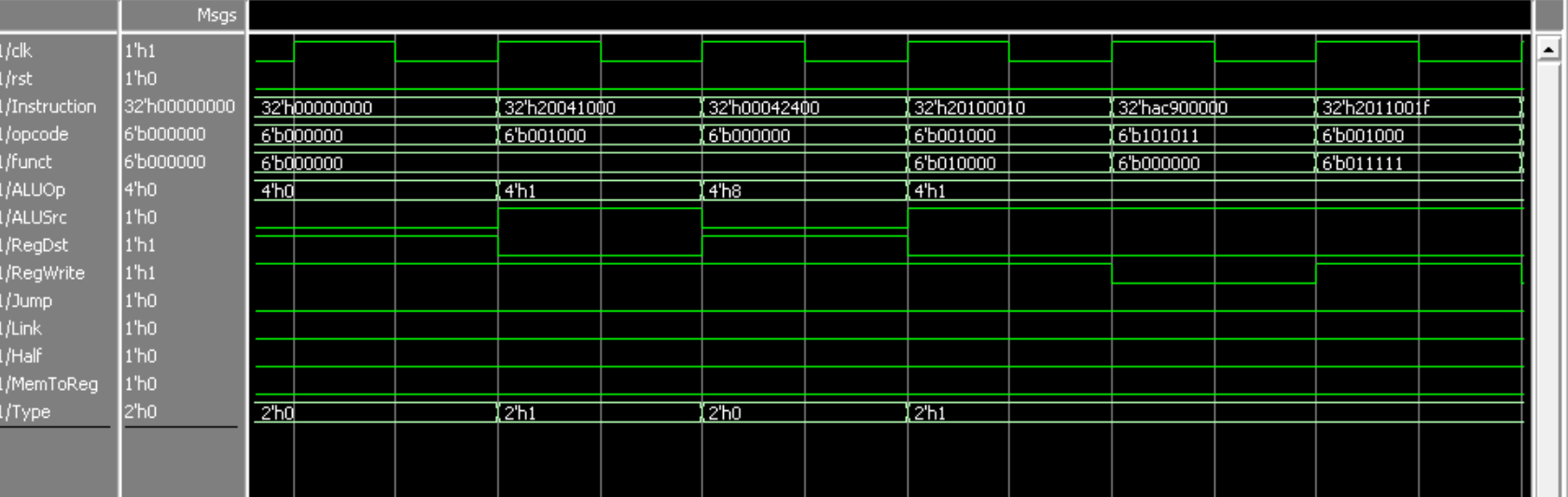
**實驗結果圖(snapshot of result)**

****

**指令波型圖( Instructions waveform )**

(Please explain why your snapshot is correct, including the wires, signals.)

Controller:



I-type : addi

I-type : addi

Reason:

**I-type addi** :

ALUOp=1 (ALU\_result = src1 + src2)

ALUSrc=1 (ALU uses imm instead of rt)

RegDst=0 (RegFile uses rt instead of rd)

RegWrite=1 (need to write register)

Jump=0 (no need to jump)

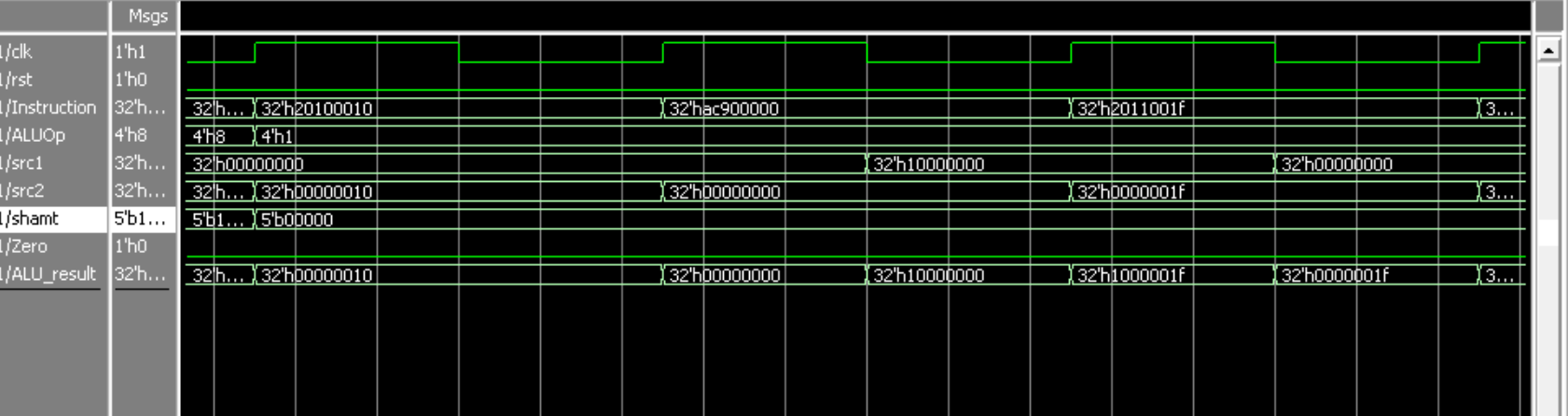
Link=0 (no need to link)

Half=0 (no need to sign-extend)

MemToReg=0 (no need to write data from Mem to Reg)

Type=1 (0=>R-type 1=>I-type 2=> J-type)

ALU



I-type : sw

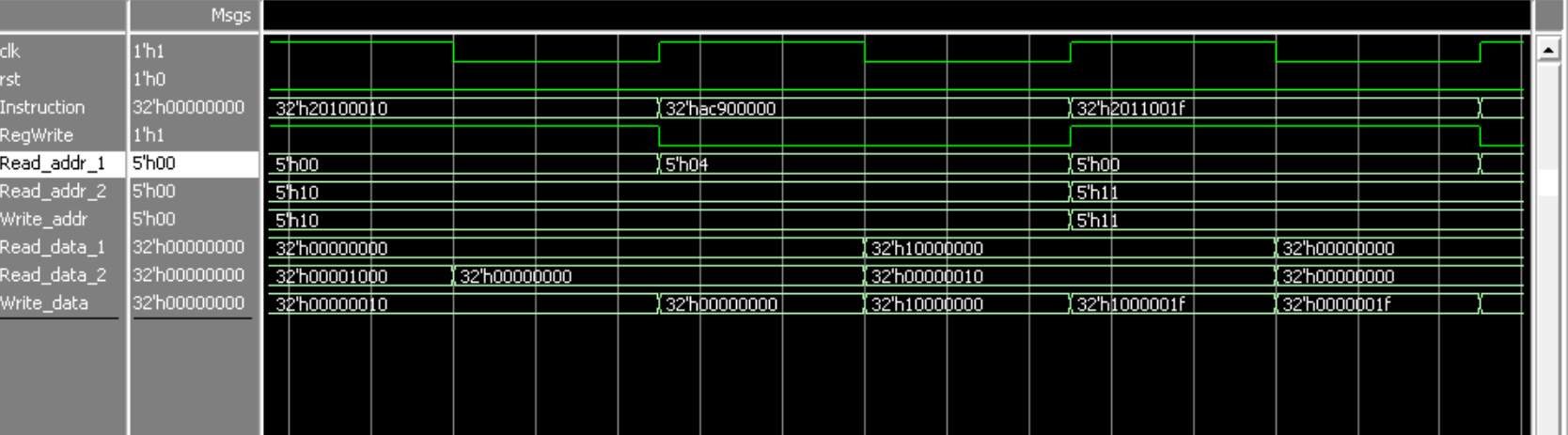
Reason:

ALUOp decides action performed (add, sub, and..).

Zero stores the result of (beq/bne).

Zero=0 when not in use.

Regfile



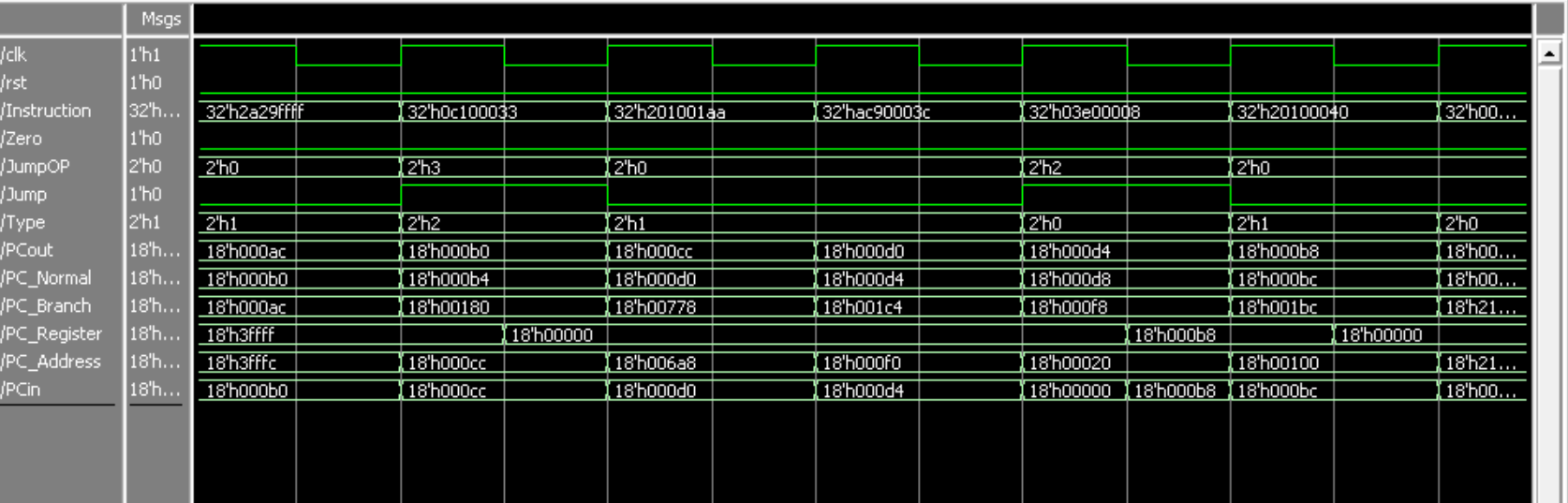
R-type : sw

Reason:

Writes data into register if RegWrite=1.

Output register data (rs and rt) with each clock cycle.

Jump\_Ctrl



jr / jalr

j / jal

Reason:

(PC related wires are not in Jump\_Ctrl, displayed here to prove Jump\_Ctrl is effective)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **JumpOP** | 0(Normal) | 1(beq/bne) | 2(jr/jalr) | 3(j/jal) |
| **Target** | PC\_Normal | PC\_Branch | PC\_Register | PC\_Address |

branch is decided by Zero (jump if Zero=1)

CPU datapath (If your CPU datapath is different from HW2 Fig.2.)

Reason:

**心得(Report)**

(請寫下完成本次作業的心得、學到哪些東西、困難點的部分。大約   
 100~200字 )

(Please write your learned lesson and conclusion, and difficult point. About   
 100~200 words)

一開始完全不懂要怎地做，只能照著MIPS的指令集打，之後開始除錯之後開始了解CPU的設計。因為不熟Verilog所以剛開始花了些時間看語法和練習，而且因為和一般程式語言有些差距，所以在理解邏輯和用法上面有些困難。不過寫完這次作業後對CPU和Verilog有更深入的理解，算是蠻有收穫的。