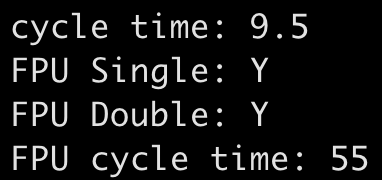
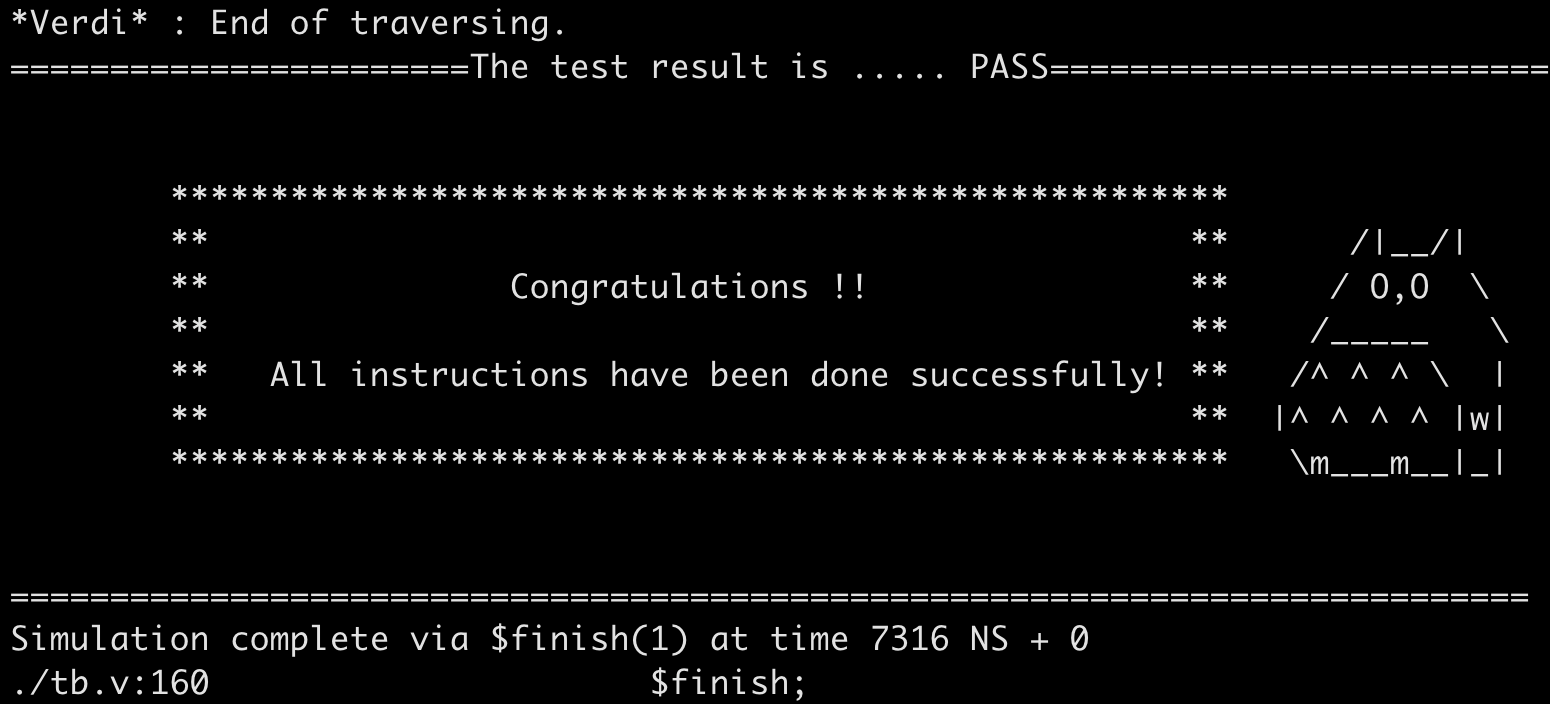
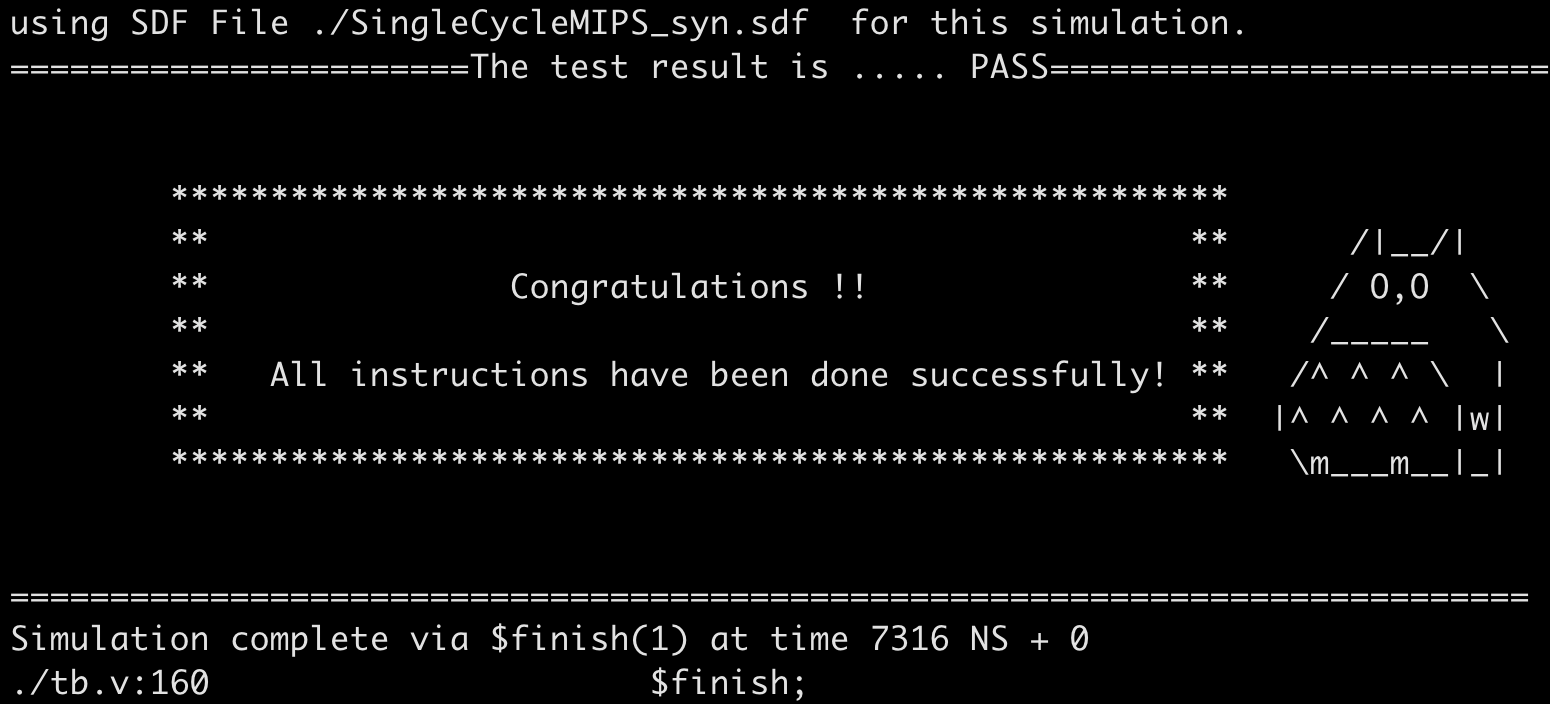
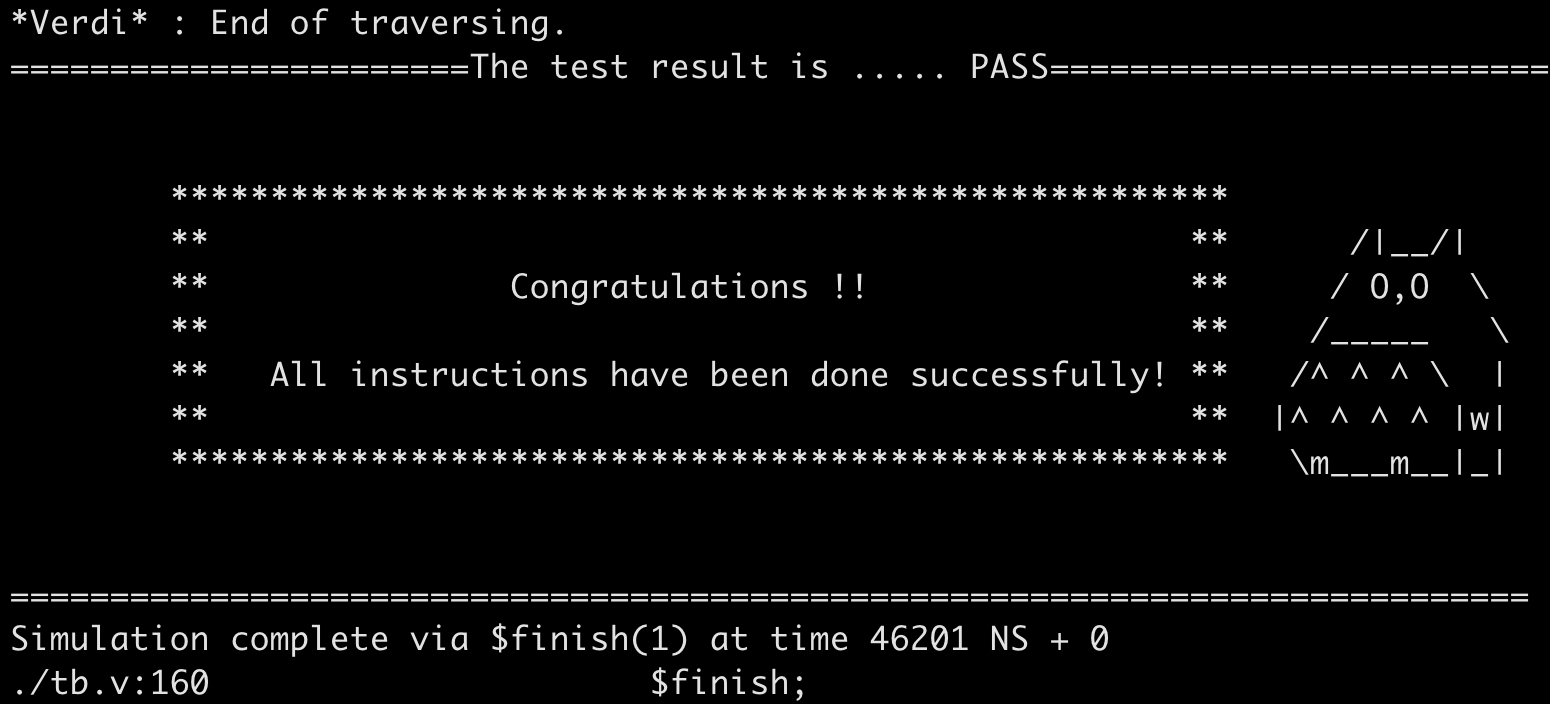
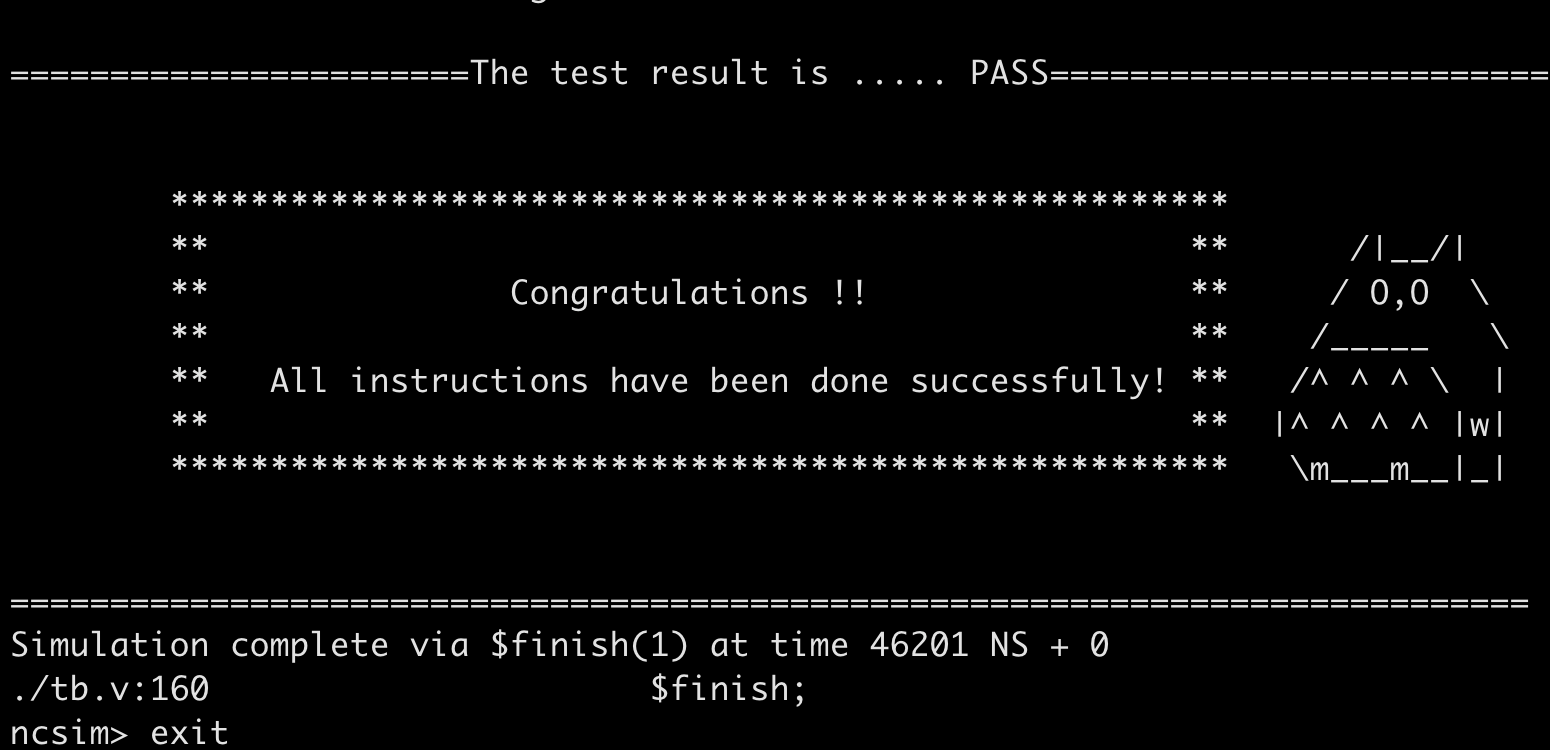
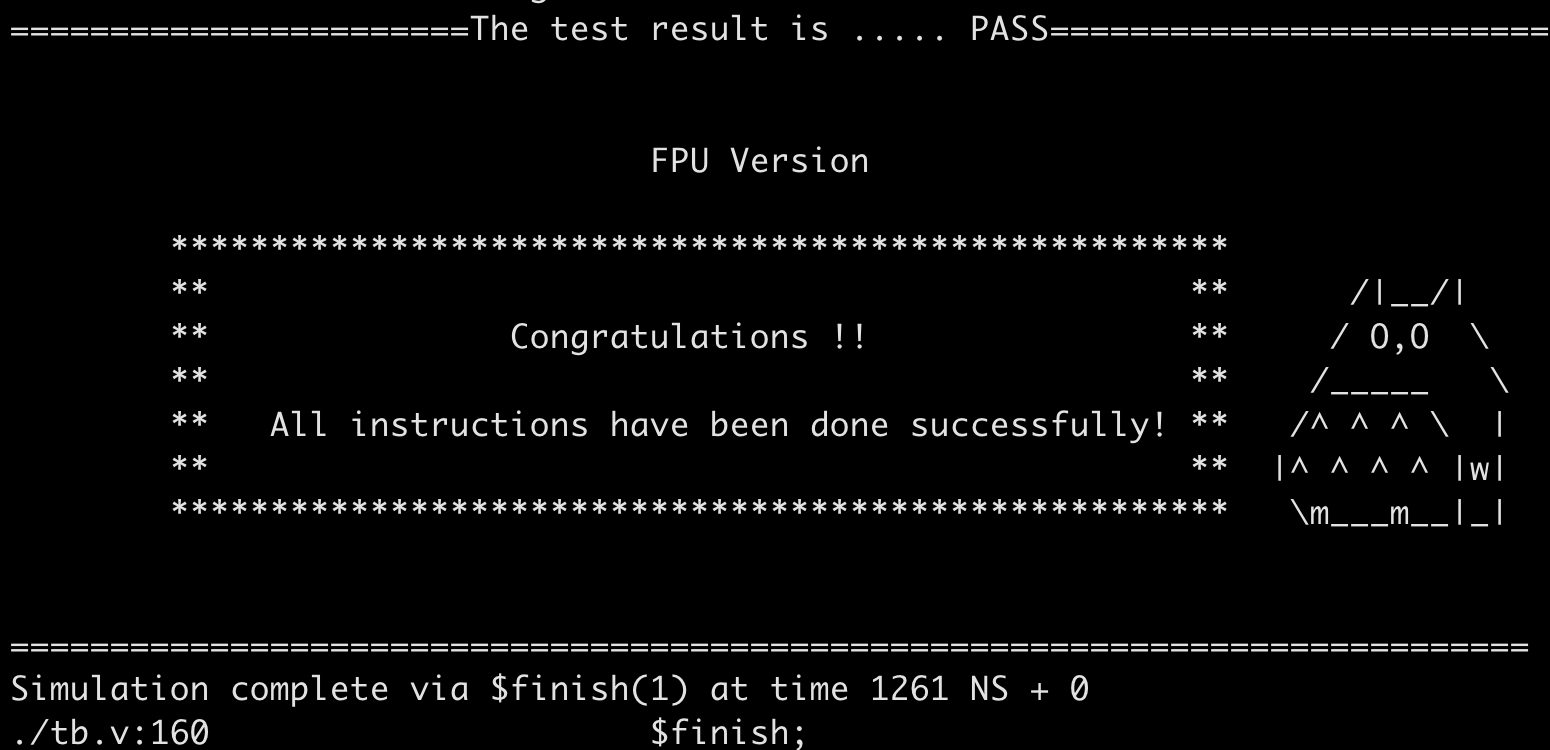
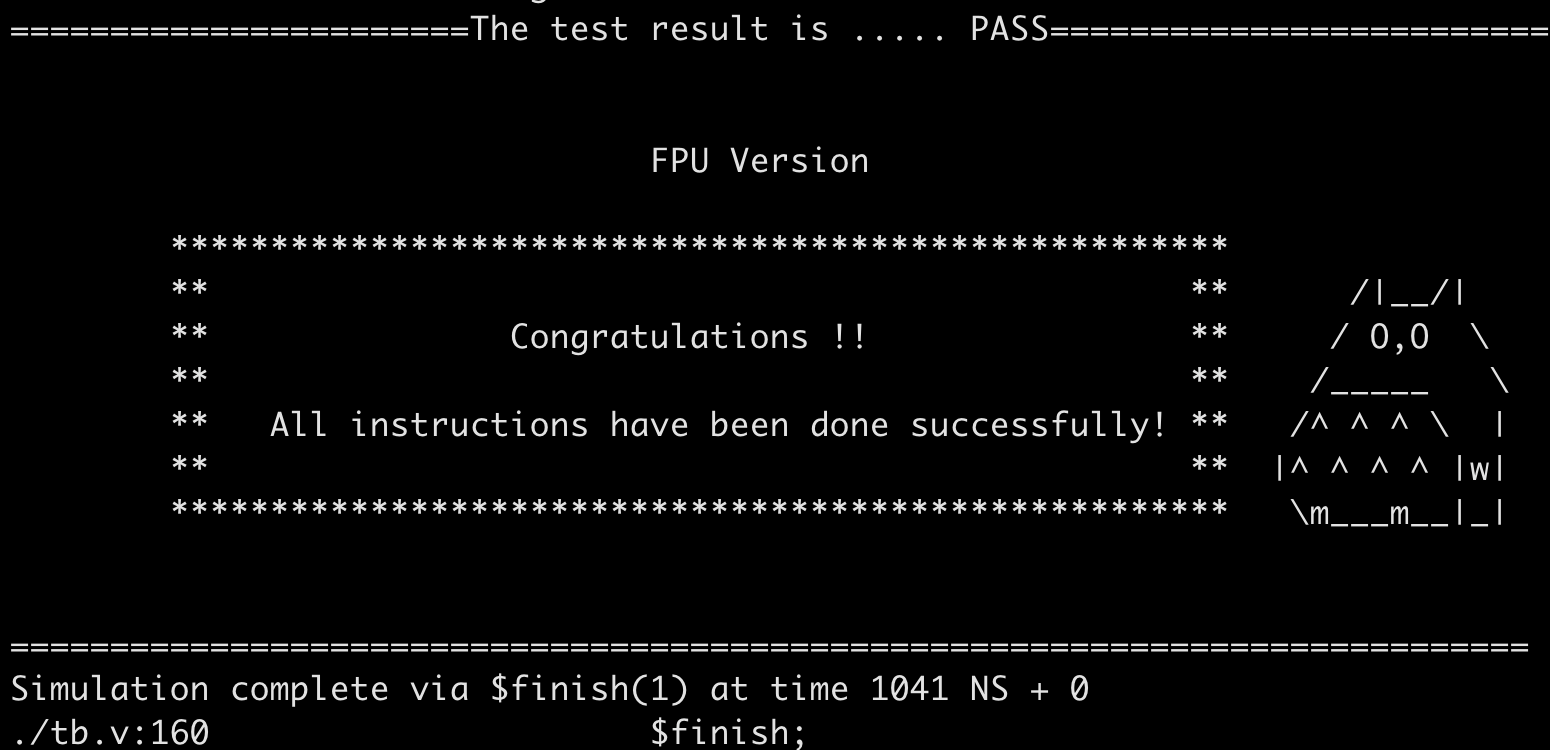
計算機結構 HW4 G15 Report

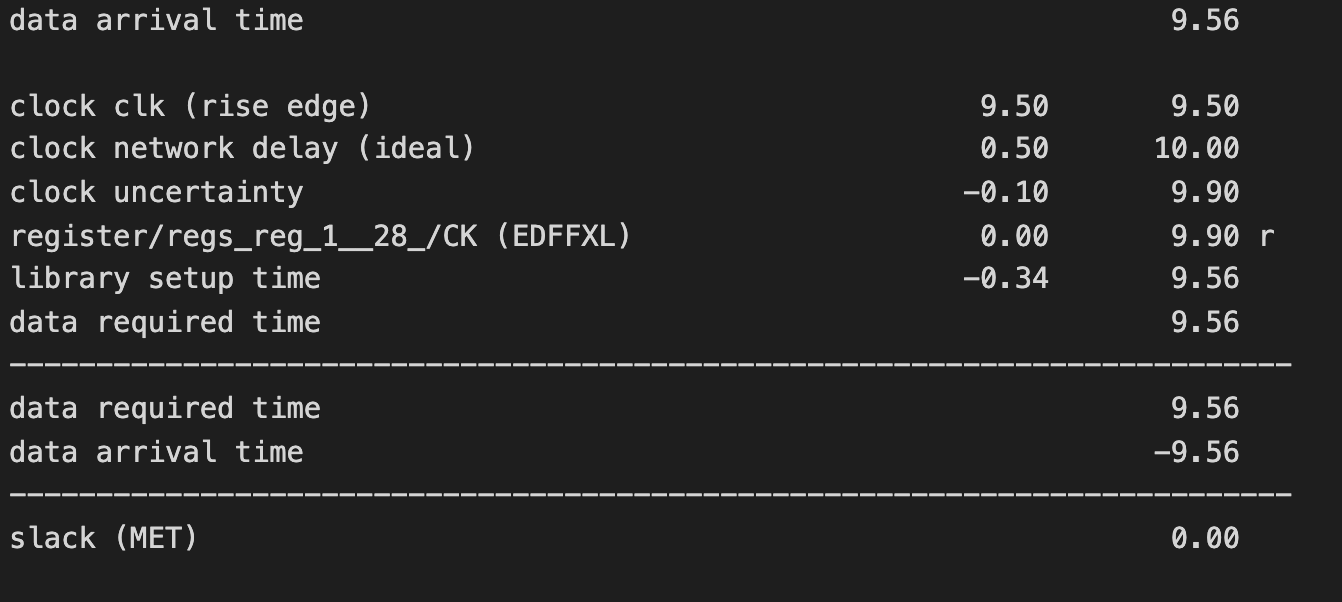
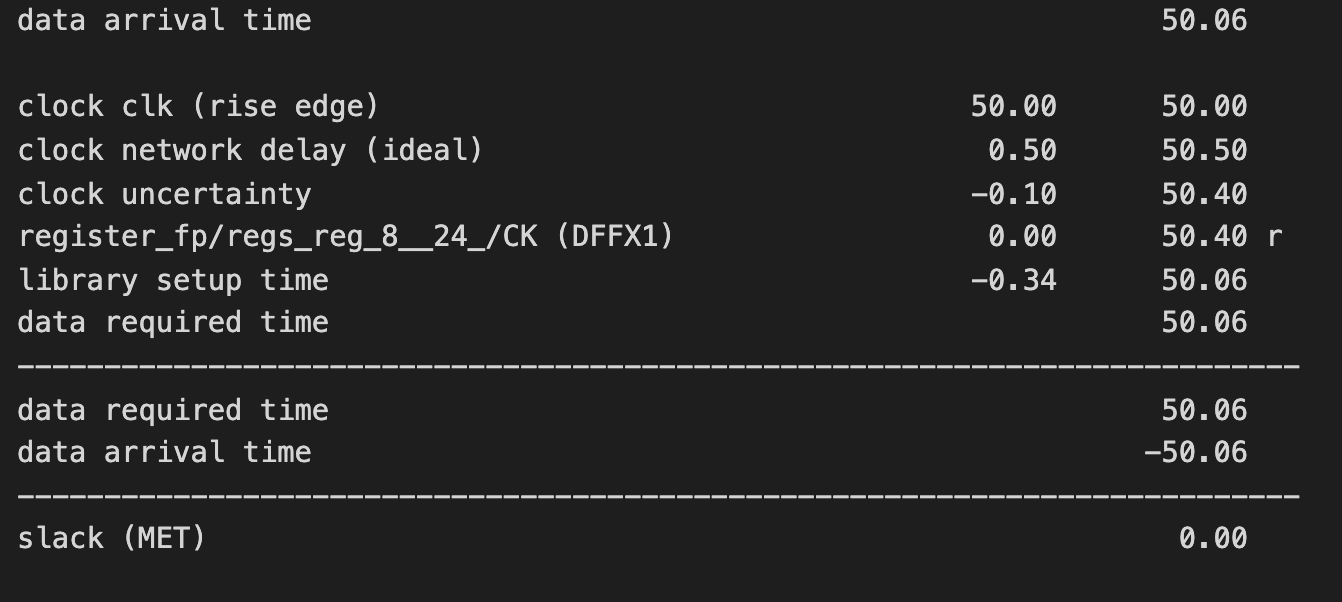
施力維 b04502031

蔡承佑 b06901051

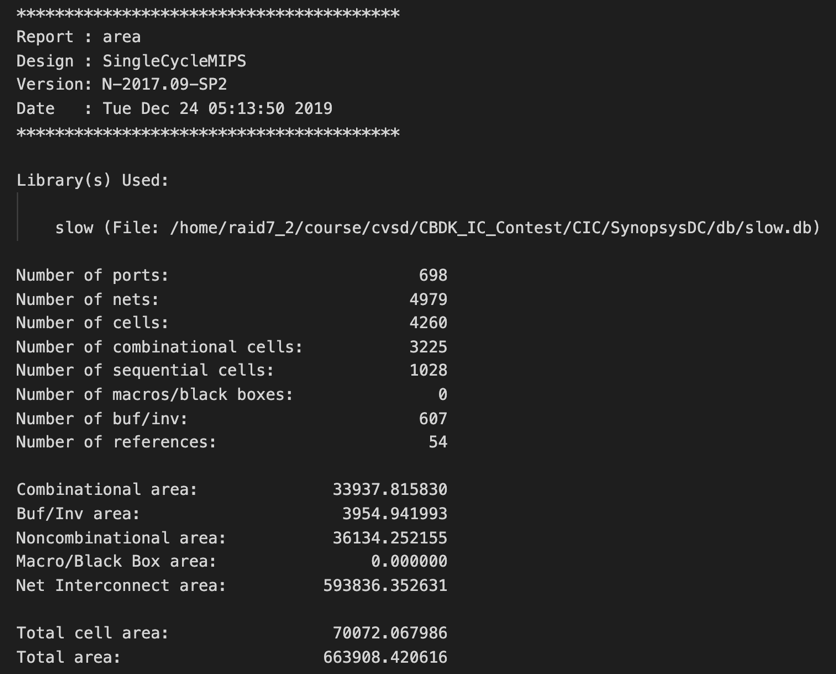
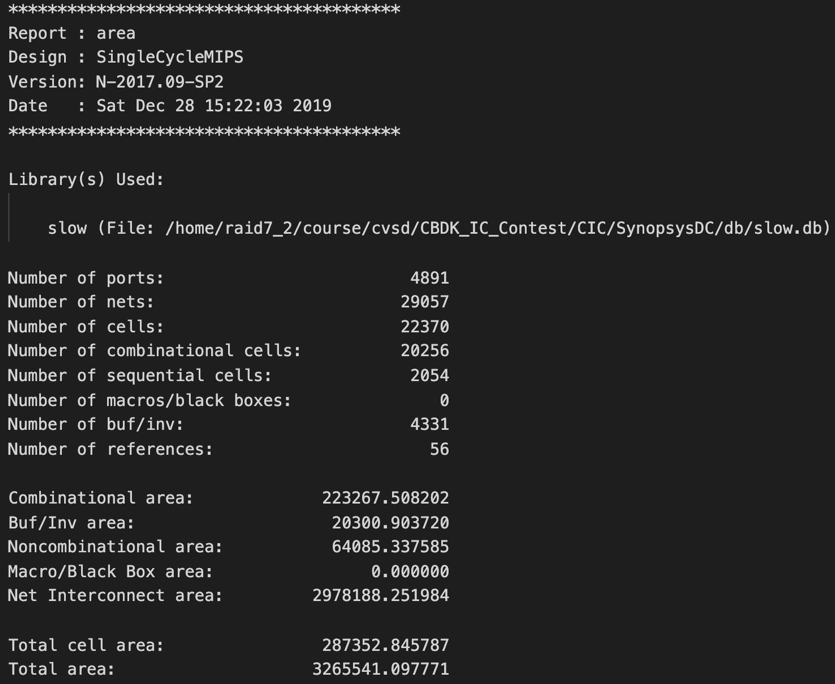
1. Screenshots
2. Readme.txt Content
3. Baseline (RTL)
4. Baseline (Gate-level)



1. FPU (Baseline RTL)
2. FPU (Single RTL)
3. FPU (Double RTL)
4. FPU (Baseline Gate-level)
5. FPU (Single Gate-level)
6. FPU (Double Gate-level)
7. Timing

* Baseline
* FPU

1. Area

* Baseline
* FPU

1. A\*T Value

A = 70072.067986, T = 9.5

→ A\*T = 665684.645867

1. Design & Difficulties
   1. Design

* Baseline:

將原有結構加入對其他指令的支援，例如：

PC Branch: (beq & zero) | (bne & ~zero)

Write Reg: jal ? 5’d31 : (RegDst ? IR[15:11] : IR[20:16])

PC input: jr ? read\_data1 : original logic with modified branching

Write data: jal ? PC\_add4 : original wire

ALU、ALU Control: 新增運算模式 (shift, less than,…)

* FPU
  1. Difficulties

Debug時間花較久，RTL較容易，而Gate-level則需要從波形中找出計算較慢的部分加以修改，但經常照著我們以為較好的方法改實際上卻差強人意。理論上將submodule盡量整併會獲得更好的A\*T，但是將原有的wire去除之後反而較差，需要多次調整。

此外，sdc的cycle time在每次修改設計後多次測試才能找出最佳解，亦花費許多時間，而compile的順序經過多次測試後發現先將原有map effort調高為high，執行後再以compile\_ultra進行incremental mapping，A\*T減少的效果顯著。

1. Work Distribution

Baseline：collaborate

FPU：力維

Optimization：承佑

1. A\*T Improvement
2. ALU adder: 將一個32-bit adder改為8個4-bit adder
3. Output A: 將原本從ALU計算的答案拉出改為單獨計算(兩個和(a)共用的4-bit adder + XOR for MSB)
4. Register: 由 assign read\_data = regs[read\_reg];

調整為:

always @(\*) begin

case(read\_reg)

5’dxx : read\_data = regs[xx];

1. ALU Control: 合併至ALU內
2. ALU: 將beq, bne的運算從 ”-“ 改為 ”!=”
3. PC: 使用30bit計算
4. Control: 各個output對應的opcode做K-map化簡
5. run.tcl: 新增以下指令 (effect: A: 73k→70k; T: 10→9.5)

set\_structure true

group\_path -name clk -weight 15

report\_path\_group

compile -map\_effort high #改自compile

compile\_ultra -increment