

2024/03/20

實驗二

序向邏輯練習

姓名：黃文祺 學號：01057013

班級：資工 3A

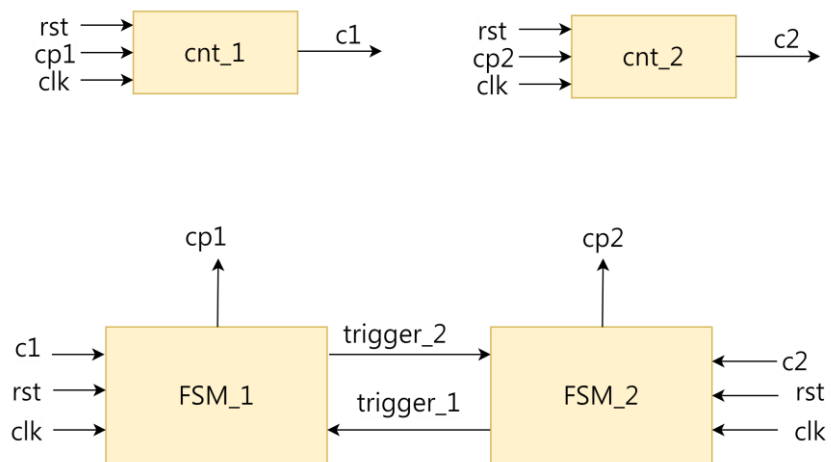
E-mail：OOOOOOOOO

注意

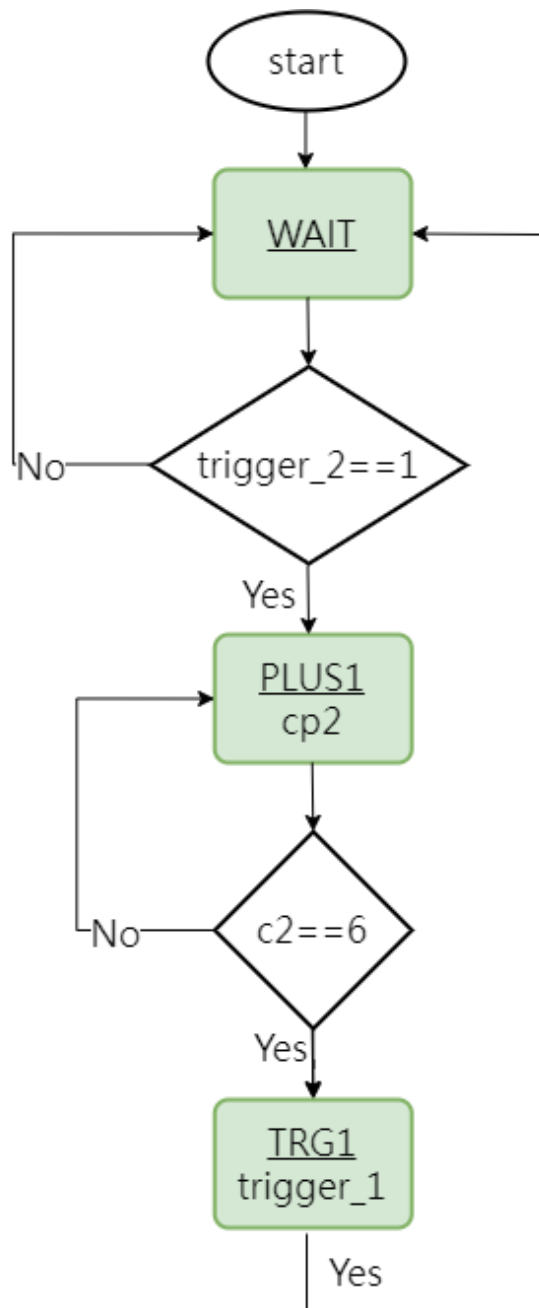
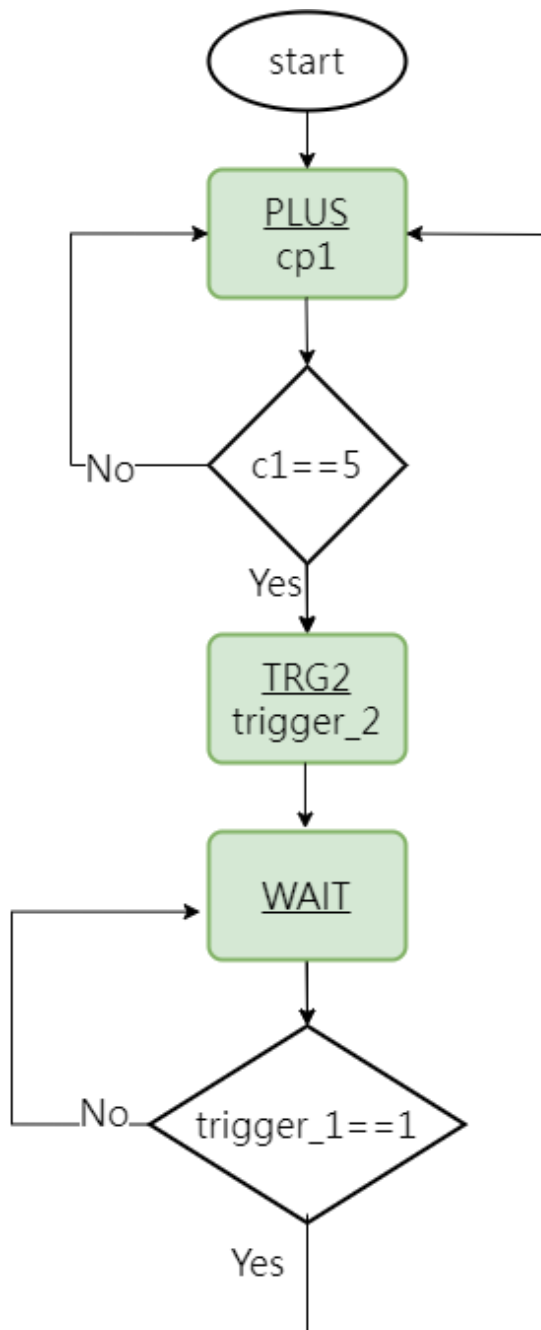
1. 一律將此檔轉成 PDF 檔
繳交
2. 繳交期限為
隔週三上午九點
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
檔名請按照作業檔名格式進行填寫
未依照格式不予批改

一、Handshaking

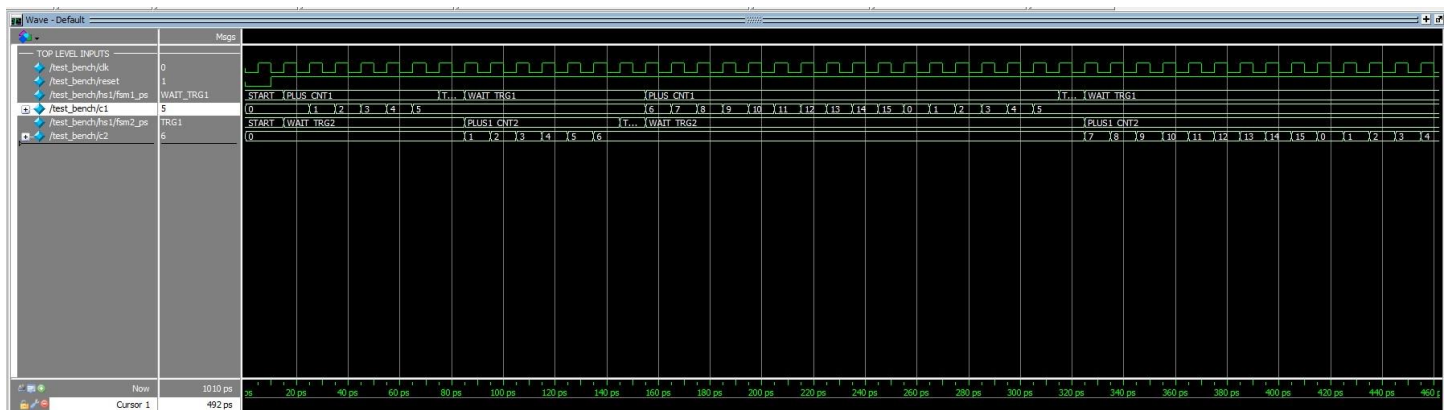
- 如下圖所示，2 個 4 bits counter (cnt_1, cnt_2)和 2 個 FSM。



- 請用 systemverilog 設計此電路，及控制之 FSM，FSM_1, FSM_2 之流程圖如下所示：



STATE
OUT



■ 系統架構程式碼、測試資料程式碼與程式碼說明

截圖請善用 win+shift+S

hs.sv handshaking 的主程式

```
C:\Users\user\Desktop\handshaking\DE0_CV\design\hs.sv - Notepad++
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 編碼(N) 語言(L) 設定(T) 工具(O) 巨集(M) 執行(R) 外掛(P) 視窗(W) ?
testbench.sv hs.sv
1 module hs(
2     input clk,
3     input reset
4 );
5
6     logic cp1;
7     logic [3:0] c1;
8     logic cp2;
9     logic [3:0] c2;
10    logic trigger_1;
11    logic trigger_2;
12
13    always_ff @(posedge clk) //cnt1
14        if (reset) c1 <= 0;
15        else if (cp1) c1 <= c1 + 1;
16
17    always_ff @(posedge clk) //cnt2
18        if (reset) c2 <= 0;
19        else if (cp2) c2 <= c2 + 1;
20
21
22
23    //----- FSM1
24    typedef enum {START1, PLUS_CNT1, TRG2, WAIT_TRG1} state1_t; //FSM state
25    state1_t fsm1_ps, fsm1_ns;
26
27    always_ff @(posedge clk) //ps ns
28        if (reset) fsm1_ps <= START1;
29        else fsm1_ps <= fsm1_ns;
30
31    always_comb begin
32        cp1 = 0;
33        trigger_2 = 0;
34        fsm1_ns = START1;
35
36        case (fsm1_ps)
37
38            START1:begin
39                fsm1_ns = PLUS_CNT1;
40            end
41
42            PLUS_CNT1:begin
43                if (c1 == 5) begin
44                    fsm1_ns = TRG2;
45                end
46
47                else begin
48                    cp1 = 1;
49                    fsm1_ns = PLUS_CNT1;
50                end
51            end
52
53            TRG2:begin
54                trigger_2 = 1;
55                fsm1_ns = WAIT_TRG1;
56            end
57
58            WAIT_TRG1:begin
59
60                if (trigger_1) begin
61                    fsm1_ns = PLUS_CNT1;
62                    cp1 = 1;
63                end
64
65                else fsm1_ns = WAIT_TRG1;
66            end
67
68        endcase
69    end
70    //-----//
71
```

```

73 //-----// FSM2
74 typedef enum {START2, PLUS_CNT2, TRG1, WAIT_TRG2} state2_t;
75 state2_t fsm2_ps, fsm2_ns;
76
77 always_ff @(posedge clk)
78     if (reset) fsm2_ps <= START2;
79     else fsm2_ps <= fsm2_ns;
80
81
82 always_comb begin
83     cp2 = 0;
84     trigger_1 = 0;
85     fsm2_ns = START2;
86
87     case (fsm2_ps)
88
89     START2:begin
90         fsm2_ns = WAIT_TRG2;
91     end
92
93     PLUS_CNT2:begin
94
95         if (c2 == 6) begin
96             fsm2_ns = TRG1;
97         end
98
99         else begin
100             cp2 = 1;
101             fsm2_ns = PLUS_CNT2;
102         end
103     end
104
105     TRG1:begin
106         trigger_1 = 1;
107         fsm2_ns = WAIT_TRG2;
108     end
109
110     WAIT_TRG2:begin
111         if (trigger_2) begin
112             fsm2_ns = PLUS_CNT2;
113             cp2 = 1;
114         end
115         else fsm2_ns = WAIT_TRG2;
116     end
117
118 endcase
119
120 end
121 //-----//
122 endmodule

```

testbench.sv

```
C:\Users\user\Desktop\handshaking\DE0_CV\simulation\tb\testbench.v - Notepad++
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 編碼(N) 語言(L) 設定(T) 工具(O) 巨集(M) 執行(R) 外掛(P) 視窗(W) ?
testbench.v hs.v
1 module testbench;
2
3     logic clk;
4     logic reset;
5
6     hs hsl(
7         .clk(clk),
8         .reset(reset)
9     );
10
11     always #5 clk = ~clk;
12
13     initial begin
14         #0 clk = 0; reset = 1;
15         #10 reset = 0;
16         #1000 $stop;
17     end
18
19 endmodule
```

■ 模擬結果與結果說明：

[illegible]

二、PLL

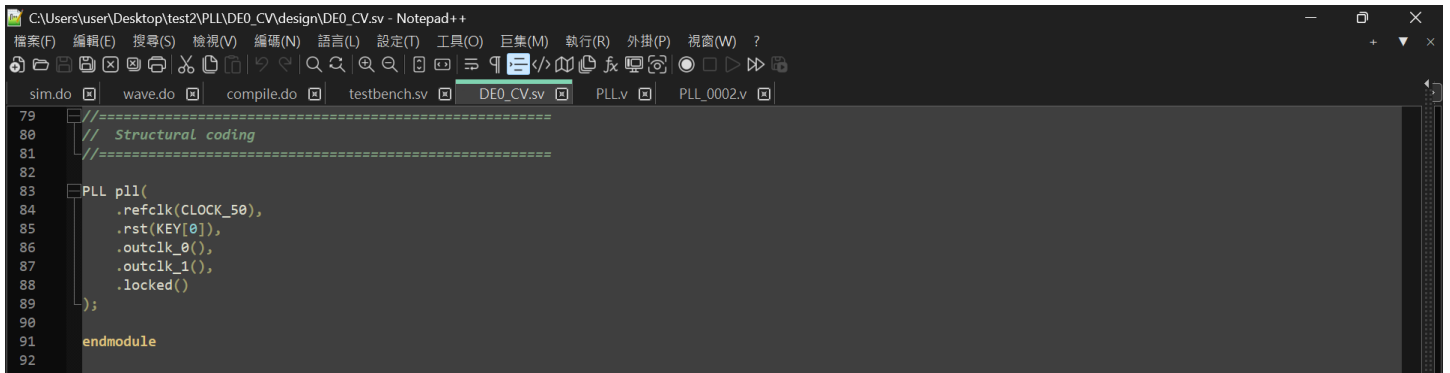
■ 實驗說明：

- 使用內建的 PLL IP 及系統 CLK 製作出兩組 CLK，並透過 ModelSim 及 Signal Tap 觀察。
- 第一個 CLK 為 100MHz。
- 第二個 CLK 為 10MHz。

■ 系統架構程式碼、測試資料程式碼與程式碼說明

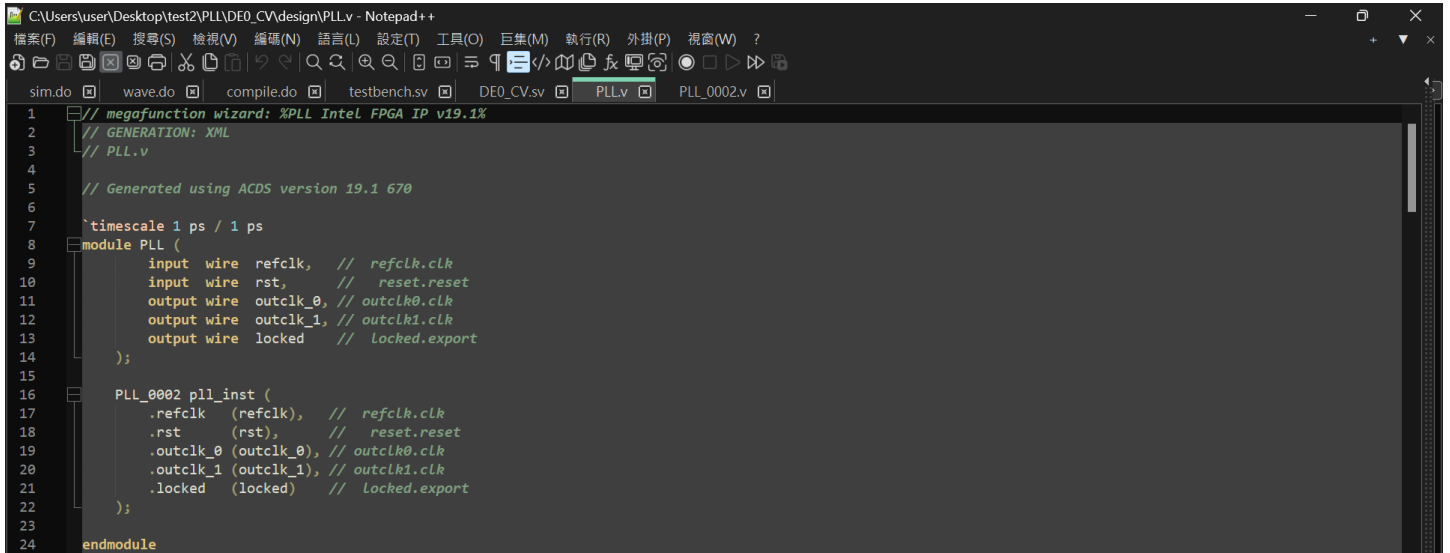
截圖請善用 win+shift+S

DE0_CV.sv 有改的地方



```
79 //=====
80 // Structural coding
81 //=====
82
83 PLL pll(
84     .refclk(CLOCK_50),
85     .rst(KEY[0]),
86     .outclk_0(),
87     .outclk_1(),
88     .locked()
89 );
90
91 endmodule
92
```

產生的 PLL.v



```
1 // megafunction wizard: %PLL Intel FPGA IP v19.1%
2 // GENERATION: XML
3 // PLL.v
4
5 // Generated using ACDS version 19.1 670
6
7 `timescale 1 ps / 1 ps
8 module PLL (
9     input wire refclk, // refclk.clk
10    input wire rst, // reset.reset
11    output wire outclk_0, // outclk0.clk
12    output wire outclk_1, // outclk1.clk
13    output wire locked // Locked.export
14 );
15
16 PLL_0002 pll_inst (
17     .refclk (refclk), // refclk.clk
18     .rst (rst), // reset.reset
19     .outclk_0 (outclk_0), // outclk0.clk
20     .outclk_1 (outclk_1), // outclk1.clk
21     .locked (locked) // Locked.export
22 );
23
24 endmodule
```

產生的 PLL_0002.v

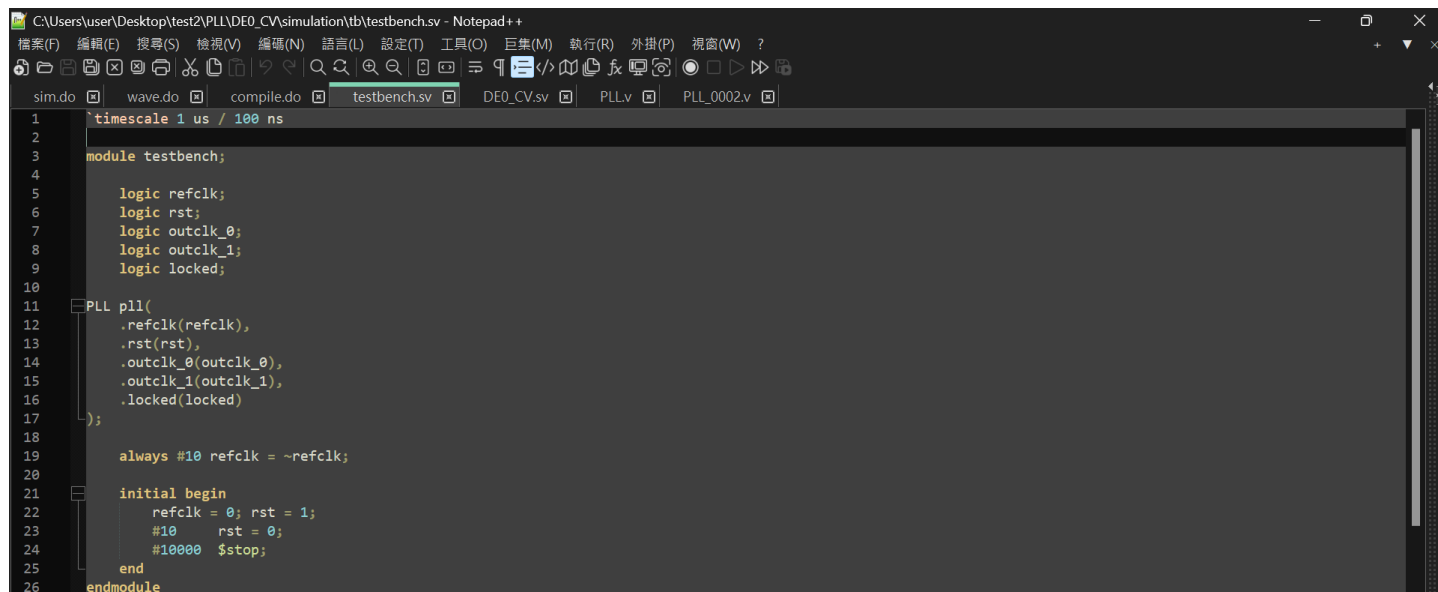
```
C:\Users\user\Desktop\test2\PLL\DE0_CV\design\PLL\PLL_0002.v - Notepad++
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 編碼(N) 語言(L) 設定(T) 工具(O) 巨集(M) 執行(R) 外掛(P) 視窗(W) ?
sim.do  wave.do  compile.do  testbench.sv  DE0_CV.sv  PLL.v  PLL_0002.v

1  timescale 1ns/10ps
2  module PLL_0002(
3
4      // interface 'refclk'
5      input wire refclk,
6
7      // interface 'reset'
8      input wire rst,
9
10     // interface 'outclk0'
11     output wire outclk_0,
12
13     // interface 'outclk1'
14     output wire outclk_1,
15
16     // interface 'locked'
17     output wire locked
18 );
19
20 altera_pll #(
21     .fractional_vco_multiplier("false"),
22     .reference_clock_frequency("50.0 MHz"),
23     .operation_mode("direct"),
24     .number_of_clocks(2),
25     .output_clock_frequency0("100.000000 MHz"),
26     .phase_shift0("0 ps"),
27     .duty_cycle0(50),
28     .output_clock_frequency1("10.000000 MHz"),
29     .phase_shift1("0 ps"),
30     .duty_cycle1(50),
31     .output_clock_frequency2("0 MHz"),
32     .phase_shift2("0 ps"),
33     .duty_cycle2(50),
34     .output_clock_frequency3("0 MHz"),
35     .phase_shift3("0 ps"),
36     .duty_cycle3(50),
37     .output_clock_frequency4("0 MHz"),
38     .phase_shift4("0 ps"),
39     .duty_cycle4(50),
40     .output_clock_frequency5("0 MHz"),
41     .phase_shift5("0 ps"),
42     .duty_cycle5(50),
43     .output_clock_frequency6("0 MHz"),
44     .phase_shift6("0 ps"),
45     .duty_cycle6(50),
46     .output_clock_frequency7("0 MHz"),
47     .phase_shift7("0 ps"),
48     .duty_cycle7(50),
49     .output_clock_frequency8("0 MHz"),
50     .phase_shift8("0 ps"),
51     .duty_cycle8(50),
52     .output_clock_frequency9("0 MHz"),
53     .phase_shift9("0 ps"),
54     .duty_cycle9(50),
55     .output_clock_frequency10("0 MHz"),
56     .phase_shift10("0 ps"),
57     .duty_cycle10(50),
58     .output_clock_frequency11("0 MHz"),
59     .phase_shift11("0 ps"),
60     .duty_cycle11(50),
61     .output_clock_frequency12("0 MHz"),
62     .phase_shift12("0 ps"),
63     .duty_cycle12(50),
64     .output_clock_frequency13("0 MHz"),
65     .phase_shift13("0 ps"),
66     .duty_cycle13(50),
67     .output_clock_frequency14("0 MHz"),
68     .phase_shift14("0 ps"),
69     .duty_cycle14(50),
70     .output_clock_frequency15("0 MHz"),
71     .phase_shift15("0 ps"),
72     .duty_cycle15(50),
73     .output_clock_frequency16("0 MHz"),
74     .phase_shift16("0 ps"),
75     .duty_cycle16(50),
76     .output_clock_frequency17("0 MHz"),
77     .phase_shift17("0 ps"),
78     .duty_cycle17(50),
79     .pll_type("General"),
80     .pll_subtype("General")
81 ) altera_pll_i (
82     .rst      (rst),
83     .outclk   ({outclk_1, outclk_0}),
84     .locked   (locked),
85     .fboutclk ( ),
86     .fbclk    (1'b0),
87     .refclk   (refclk)
88 );
89 endmodule
```

testbench.v:

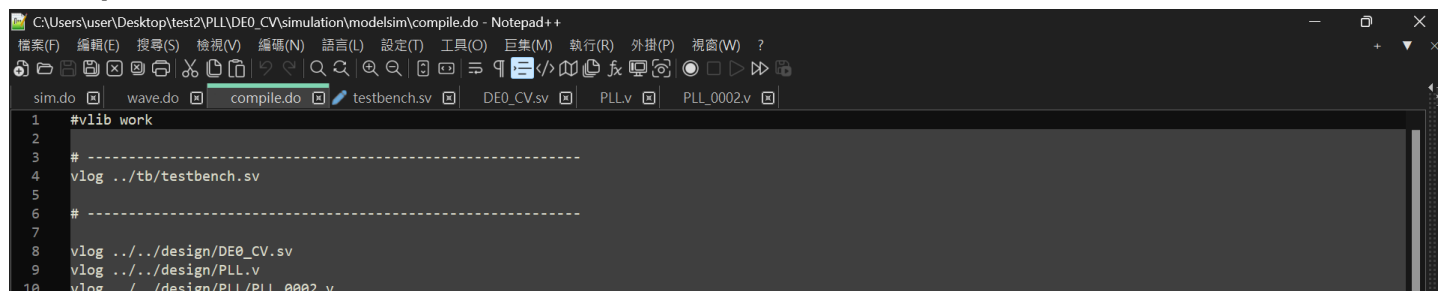
一開始沒加到`timescale 1ps / 1ps，wave 就出不來

** Error (suppressible): (vsim-3009) [TSCALE] - Module 'testbench' does not have a timeunit/timeprecision specification in effect, but other modules do.

A screenshot of a Notepad++ window titled "C:\Users\user\Desktop\test2\PLL\DE0_CV\simulation\tb\testbench.v - Notepad++". The code defines a Verilog testbench module. It starts with a time scale of 1 us / 100 ns. It declares logic signals for refclk, rst, outclk_0, outclk_1, and locked. It instantiates a PLL block with these signals. It includes a clock divider logic: always #10 refclk = ~refclk;. It has an initial block that sets refclk to 0, rst to 1, and then waits for 10000 time units before setting rst to 0 and stopping the simulation. The module ends with endmodule.

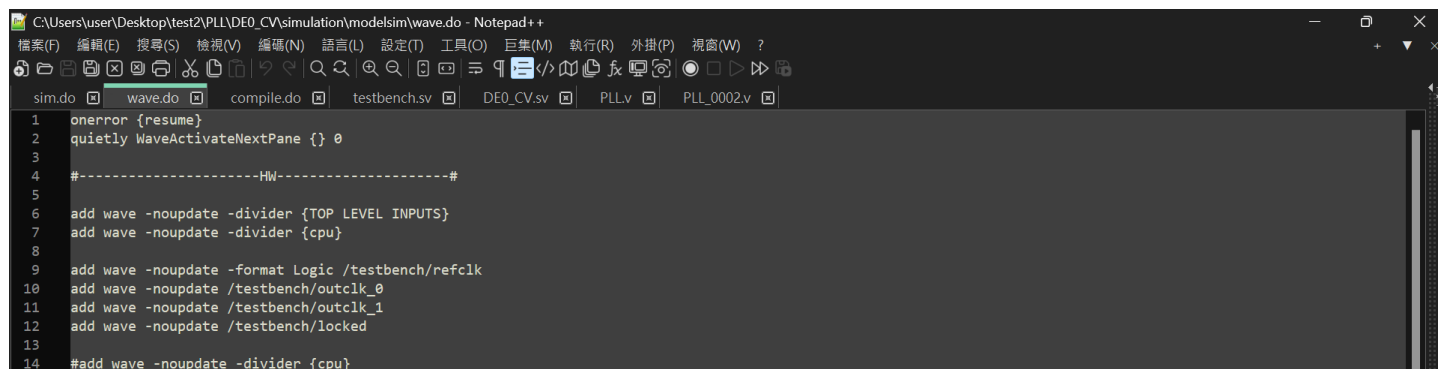
```
1 `timescale 1 us / 100 ns
2
3 module testbench;
4
5     logic refclk;
6     logic rst;
7     logic outclk_0;
8     logic outclk_1;
9     logic locked;
10
11     PLL pll(
12         .refclk(refclk),
13         .rst(rst),
14         .outclk_0(outclk_0),
15         .outclk_1(outclk_1),
16         .locked(locked)
17     );
18
19     always #10 refclk = ~refclk;
20
21     initial begin
22         refclk = 0; rst = 1;
23         #10 rst = 0;
24         #10000 $stop;
25     end
26 endmodule
```

compile.do

A screenshot of a Notepad++ window titled "C:\Users\user\Desktop\test2\PLL\DE0_CV\simulation\modelsim\compile.do - Notepad++". The script sets the work directory to the current directory and uses vlog to compile the testbench.v file and the design files DE0_CV.v, PLL.v, and PLL_0002.v.

```
1 #vlib work
2
3 # -----
4 vlog ../tb/testbench.v
5 # -----
6
7
8 vlog ../../design/DE0_CV.v
9 vlog ../../design/PLL.v
10 vlog ../../design/PLL/PLL_0002.v
```

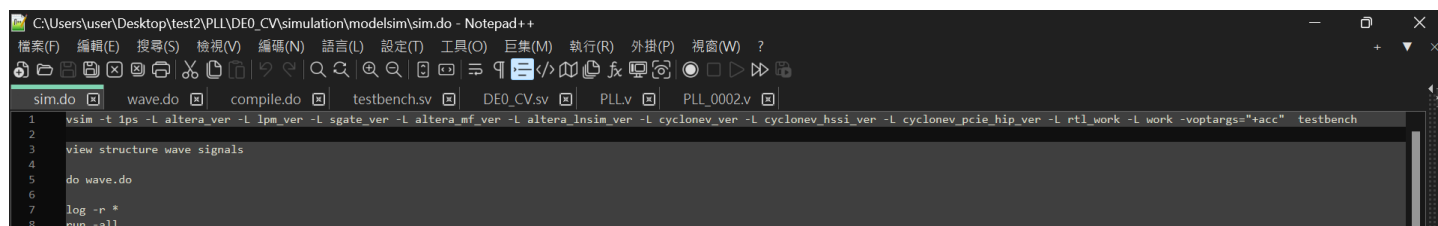
wave.do

A screenshot of a Notepad++ window titled "C:\Users\user\Desktop\test2\PLL\DE0_CV\simulation\modelsim\wave.do - Notepad++". The script sets onerror to resume, quietly activates the next pane, and adds wave signals for the testbench signals. It also adds a divider for the CPU.

```
1 onerror {resume}
2 quietly WaveActivateNextPane {} 0
3
4 #-----HW-----#
5
6 add wave -noupdate -divider {TOP LEVEL INPUTS}
7 add wave -noupdate -divider {cpu}
8
9 add wave -noupdate -format Logic /testbench/refclk
10 add wave -noupdate /testbench/outclk_0
11 add wave -noupdate /testbench/outclk_1
12 add wave -noupdate /testbench/locked
13
14 #add wave -noupdate -divider {cpu}
```

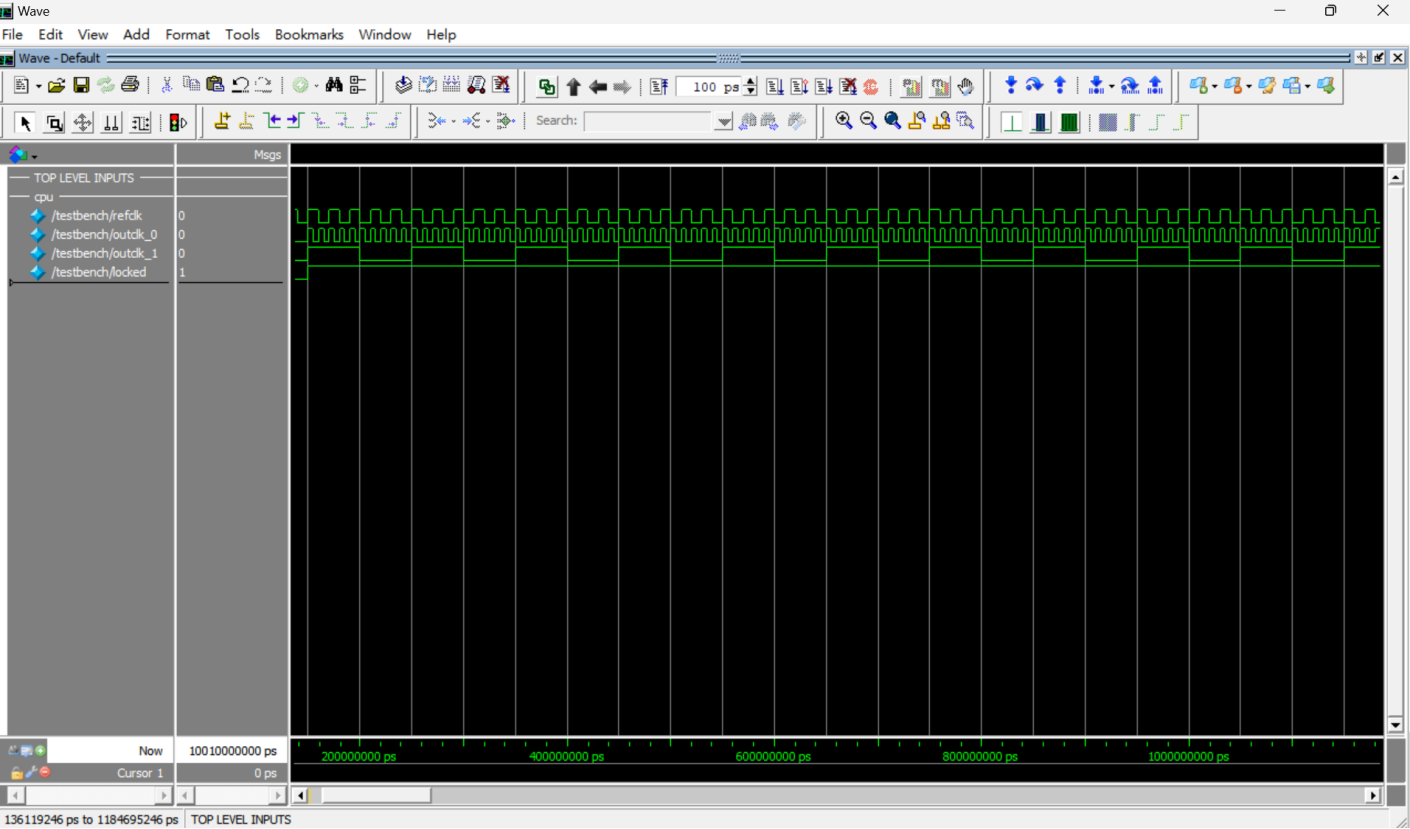
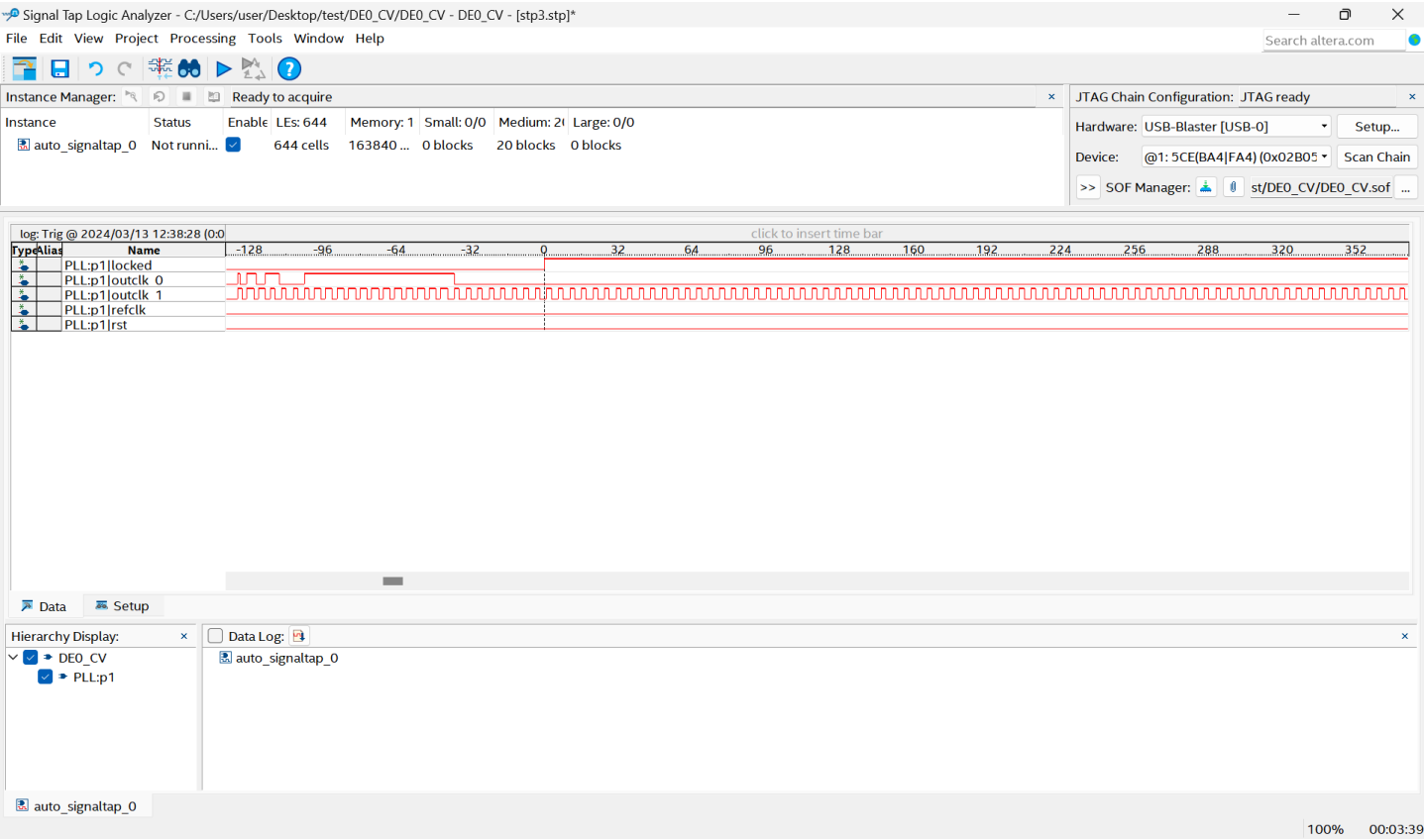
sim.do

一開始我以為講義是把第一行那串分成四行(大誤會 XD)

A screenshot of a Notepad++ window titled "C:\Users\user\Desktop\test2\PLL\DE0_CV\simulation\modelsim\sim.do - Notepad++". The script runs a vsim command with various library paths and the testbench module. It then sets the view structure to wave signals, runs the wave.do script, logs the results, and finally runs the simulation.

```
1 vsim -t 1ps -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_lnsim_ver -L cyclonev_ver -L cyclonev_hssi_ver -L cyclonev_pcie_hip_ver -L rtl_work -L work -voptargs="+acc" testbench
2
3 view structure wave signals
4
5 do wave.do
6
7 log -r *
8 run -all
```


■ 模擬結果與結果說明：



■ 結論與心得：

第一個實驗按照圖做各塊要做的事寫好後再把他們連接起來就好了~

除了一開始有點忘記怎麼給狀態用文字定義之外，其他都還好

第二個實驗主要是`timescale 1ps / 1ps 沒加到以及 sim.do 那個第一行要自己處理一下而已。