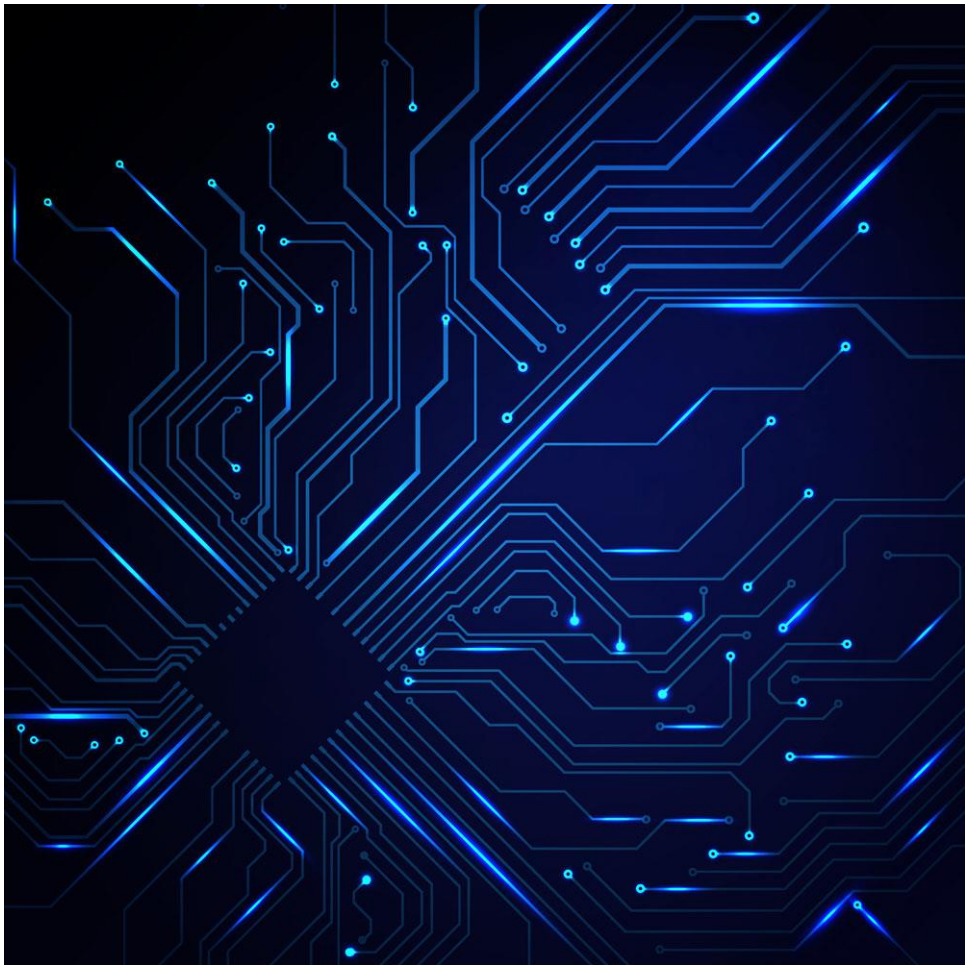


注意

1. 繳交時一律轉 PDF 檔
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

2023/03/15

實驗三

SIGNAL TAP

姓名：吳秉宸　　學號：00957202

班級：資工 4A

E-mail：evan20010126@gmail.com

一、 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

pll.v

```
// megafunction wizard: %PLL Intel FPGA IP v19.1%
// GENERATION: XML
// pll.v

// Generated using ACDS version 19.1 670

`timescale 1 ps / 1 ps
module pll (
        input  wire  refclk,   //  refclk.clk
        input  wire  rst,      //   reset.reset
        output wire  outclk_0, // outclk0.clk
        output wire  outclk_1, // outclk1.clk
        output wire  locked    //  locked.export
    );

    pll_0002 pll_inst (
        .refclk   (refclk),   //  refclk.clk
        .rst      (rst),      //   reset.reset
        .outclk_0 (outclk_0), // outclk0.clk
        .outclk_1 (outclk_1), // outclk1.clk
        .locked   (locked)    //  locked.export
    );

endmodule
// Retrieval info: <?xml version="1.0"?>
//<!--
//  Generated by Altera MegaWizard Launcher Utility version 1.0
//  ************************************************************
//  THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
//  ************************************************************
//  Copyright (C) 1991-2023 Altera Corporation
//  Any megafunction design, and related net list (encrypted or decrypted),
//  support information, device programming or simulation file, and any other
//  associated documentation or information provided by Altera or a partner
//  under Altera's Megafunction Partnership Program may be used only to
//  program PLD devices (but not masked PLD devices) from Altera.  Any other
//  use of such megafunction design, net list, support information, device
```

```
//  programming or simulation file, or any other related documentation or
//  information is prohibited for any other purpose, including, but not
//  limited to modification, reverse engineering, de-compiling, or use with
//  any other silicon devices, unless such use is explicitly licensed under
//  a separate agreement with Altera or a megafunction partner.  Title to
//  the intellectual property, including patents, copyrights, trademarks,
//  trade secrets, or maskworks, embodied in any such megafunction design,
//  net list, support information, device programming or simulation file, or
//  any other related documentation or information provided by Altera or a
//  megafunction partner, remains with Altera, the megafunction partner, or
//  their respective licensors.  No other licenses, including any licenses
//  needed under any third party's intellectual property, are provided herein.
//-->
// Retrieval info: <instance entity-name="altera_pll" version="19.1" >
// Retrieval info:  <generic name="debug_print_output" value="false" />
// Retrieval info:  <generic name="debug_use_rbc_taf_method" value="false" />
// Retrieval info:  <generic name="device_family" value="Cyclone V" />
// Retrieval info:  <generic name="device" value="5CEBA2F17A7" />
// Retrieval info:  <generic name="gui_device_speed_grade" value="1" />
// Retrieval info:  <generic name="gui_pll_mode" value="Integer-N PLL" />
// Retrieval info:  <generic name="gui_reference_clock_frequency" value="50.0" />
// Retrieval info:  <generic name="gui_channel_spacing" value="0.0" />
// Retrieval info:  <generic name="gui_operation_mode" value="direct" />
// Retrieval info:  <generic name="gui_feedback_clock" value="Global Clock" />
// Retrieval info:  <generic name="gui_fractional_cout" value="32" />
// Retrieval info:  <generic name="gui_dsm_out_sel" value="1st_order" />
// Retrieval info:  <generic name="gui_use_locked" value="true" />
// Retrieval info:  <generic name="gui_en_adv_params" value="false" />
// Retrieval info:  <generic name="gui_number_of_clocks" value="2" />
// Retrieval info:  <generic name="gui_multiply_factor" value="1" />
// Retrieval info:  <generic name="gui_frac_multiply_factor" value="1" />
// Retrieval info:  <generic name="gui_divide_factor_n" value="1" />
// Retrieval info:  <generic name="gui_cascade_counter0" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency0" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c0" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency0" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units0" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift0" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg0" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift0" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle0" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter1" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency1" value="10.0" />
// Retrieval info:  <generic name="gui_divide_factor_c1" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency1" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units1" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift1" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg1" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift1" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle1" value="50" />
```

```
// Retrieval info:  <generic name="gui_cascade_counter2" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency2" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c2" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency2" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units2" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift2" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg2" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift2" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle2" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter3" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency3" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c3" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency3" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units3" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift3" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg3" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift3" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle3" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter4" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency4" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c4" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency4" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units4" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift4" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg4" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift4" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle4" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter5" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency5" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c5" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency5" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units5" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift5" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg5" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift5" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle5" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter6" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency6" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c6" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency6" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units6" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift6" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg6" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift6" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle6" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter7" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency7" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c7" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency7" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units7" value="ps" />
```

```
// Retrieval info:  <generic name="gui_phase_shift7" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg7" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift7" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle7" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter8" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency8" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c8" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency8" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units8" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift8" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg8" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift8" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle8" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter9" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency9" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c9" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency9" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units9" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift9" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg9" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift9" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle9" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter10" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency10" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c10" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency10" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units10" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift10" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg10" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift10" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle10" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter11" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency11" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c11" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency11" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units11" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift11" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg11" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift11" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle11" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter12" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency12" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c12" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency12" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units12" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift12" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg12" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift12" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle12" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter13" value="false" />
```

```
// Retrieval info:  <generic name="gui_output_clock_frequency13" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c13" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency13" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units13" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift13" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg13" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift13" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle13" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter14" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency14" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c14" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency14" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units14" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift14" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg14" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift14" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle14" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter15" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency15" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c15" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency15" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units15" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift15" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg15" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift15" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle15" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter16" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency16" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c16" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency16" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units16" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift16" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg16" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift16" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle16" value="50" />
// Retrieval info:  <generic name="gui_cascade_counter17" value="false" />
// Retrieval info:  <generic name="gui_output_clock_frequency17" value="100.0" />
// Retrieval info:  <generic name="gui_divide_factor_c17" value="1" />
// Retrieval info:  <generic name="gui_actual_output_clock_frequency17" value="0 MHz" />
// Retrieval info:  <generic name="gui_ps_units17" value="ps" />
// Retrieval info:  <generic name="gui_phase_shift17" value="0" />
// Retrieval info:  <generic name="gui_phase_shift_deg17" value="0.0" />
// Retrieval info:  <generic name="gui_actual_phase_shift17" value="0" />
// Retrieval info:  <generic name="gui_duty_cycle17" value="50" />
// Retrieval info:  <generic name="gui_pll_auto_reset" value="Off" />
// Retrieval info:  <generic name="gui_pll_bandwidth_preset" value="Auto" />
// Retrieval info:  <generic name="gui_en_reconf" value="false" />
// Retrieval info:  <generic name="gui_en_dps_ports" value="false" />
// Retrieval info:  <generic name="gui_en_phout_ports" value="false" />
// Retrieval info:  <generic name="gui_phout_division" value="1" />
```

```
// Retrieval info:  <generic name="gui_mif_generate" value="false" />
// Retrieval info:  <generic name="gui_enable_mif_dps" value="false" />
// Retrieval info:  <generic name="gui_dps_cntr" value="C0" />
// Retrieval info:  <generic name="gui_dps_num" value="1" />
// Retrieval info:  <generic name="gui_dps_dir" value="Positive" />
// Retrieval info:  <generic name="gui_refclk_switch" value="false" />
// Retrieval info:  <generic name="gui_refclk1_frequency" value="100.0" />
// Retrieval info:  <generic name="gui_switchover_mode" value="Automatic Switchover" />
// Retrieval info:  <generic name="gui_switchover_delay" value="0" />
// Retrieval info:  <generic name="gui_active_clk" value="false" />
// Retrieval info:  <generic name="gui_clk_bad" value="false" />
// Retrieval info:  <generic name="gui_enable_cascade_out" value="false" />
// Retrieval info:  <generic name="gui_cascade_outclk_index" value="0" />
// Retrieval info:  <generic name="gui_enable_cascade_in" value="false" />
// Retrieval info:  <generic name="gui_pll_cascading_mode" value="Create an adjpllin signal
to connect with an upstream PLL" />
// Retrieval info: </instance>
// IPFS_FILES : pll.vo
// RELATED_FILES: pll.v, pll_0002.v
```

參照上課投影片的流程產生出 pll.v 檔案。

Pll_0002.v

```
`timescale 1ns/10ps
module  pll_0002(

    // interface 'refclk'
    input wire refclk,

    // interface 'reset'
    input wire rst,

    // interface 'outclk0'
    output wire outclk_0,

    // interface 'outclk1'
    output wire outclk_1,

    // interface 'locked'
    output wire locked
);

    altera_pll #(
        .fractional_vco_multiplier("false"),
        .reference_clock_frequency("50.0 MHz"),
        .operation_mode("direct"),
        .number_of_clocks(2),
        .output_clock_frequency0("100.000000 MHz"),
```

```
        .phase_shift0("0 ps"),
        .duty_cycle0(50),
        .output_clock_frequency1("10.000000 MHz"),
        .phase_shift1("0 ps"),
        .duty_cycle1(50),
        .output_clock_frequency2("0 MHz"),
        .phase_shift2("0 ps"),
        .duty_cycle2(50),
        .output_clock_frequency3("0 MHz"),
        .phase_shift3("0 ps"),
        .duty_cycle3(50),
        .output_clock_frequency4("0 MHz"),
        .phase_shift4("0 ps"),
        .duty_cycle4(50),
        .output_clock_frequency5("0 MHz"),
        .phase_shift5("0 ps"),
        .duty_cycle5(50),
        .output_clock_frequency6("0 MHz"),
        .phase_shift6("0 ps"),
        .duty_cycle6(50),
        .output_clock_frequency7("0 MHz"),
        .phase_shift7("0 ps"),
        .duty_cycle7(50),
        .output_clock_frequency8("0 MHz"),
        .phase_shift8("0 ps"),
        .duty_cycle8(50),
        .output_clock_frequency9("0 MHz"),
        .phase_shift9("0 ps"),
        .duty_cycle9(50),
        .output_clock_frequency10("0 MHz"),
        .phase_shift10("0 ps"),
        .duty_cycle10(50),
        .output_clock_frequency11("0 MHz"),
        .phase_shift11("0 ps"),
        .duty_cycle11(50),
        .output_clock_frequency12("0 MHz"),
        .phase_shift12("0 ps"),
        .duty_cycle12(50),
        .output_clock_frequency13("0 MHz"),
        .phase_shift13("0 ps"),
        .duty_cycle13(50),
        .output_clock_frequency14("0 MHz"),
        .phase_shift14("0 ps"),
        .duty_cycle14(50),
        .output_clock_frequency15("0 MHz"),
        .phase_shift15("0 ps"),
        .duty_cycle15(50),
        .output_clock_frequency16("0 MHz"),
        .phase_shift16("0 ps"),
        .duty_cycle16(50),
```

```
        .output_clock_frequency17("0 MHz"),
        .phase_shift17("0 ps"),
        .duty_cycle17(50),
        .pll_type("General"),
        .pll_subtype("General")
    ) altera_pll_i (
        .rst    (rst),
        .outclk ({outclk_1, outclk_0}),
        .locked (locked),
        .fboutclk   ( ),
        .fbclk  (1'b0),
        .refclk (refclk)
    );
endmodule
```

參照上課投影片的流程產生出 pll_0002.v 檔案

compile.do

```
#vlib work



#
-----------------------------------------
------------------
vlog ../tb/testbench.sv
vlog ../../pll.v
vlog ../../pll/pll_0002.v
vlog ../../design/sub_4bit.sv
vlog ../../design/DE0_CV.sv
vlog ../../design/handshaking.sv
```

編譯 testbench.sv、pll.v、pll_0002.v 檔案。

sim.do

```
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

# vsim -voptargs=+acc work.testbench
view structure wave signals

do wave.do

log -r *
run -all
```

依照上課投影片引入相關 library。

Wave.do

```
onerror {resume}      You, 19 hours ago • lab 1~3
quietly WaveActivateNextPane {} 0
add wave -noupdate -divider {TOP LEVEL INPUTS}
add wave -noupdate -divider cpu
add wave -noupdate /testbench/clk
add wave -noupdate /testbench/rst
add wave -noupdate /testbench/outclk_0
add wave -noupdate /testbench/outclk_1
add wave -noupdate /testbench/locked
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {23116 ps} 0}
quietly wave cursor active 1
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
configure wave -timelineunits ns
update
WaveRestoreZoom {0 ps} {1065750 ps}
```

輸出相關信號

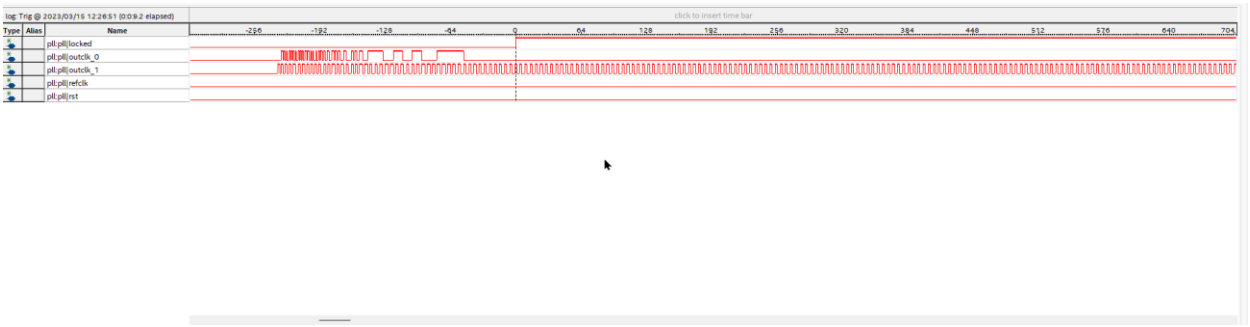
testbench.sv

```
`timescale 1ns/10ps
module testbench;

    logic clk;
    logic rst;

    logic outclk_0;
    logic outclk_1;
    logic locked;
    pll pl_test (
        .refclk(clk),
        .rst(rst), // .rst(KEY[0]),
        .outclk_0(outclk_0),
        .outclk_1(outclk_1),
        .locked(locked)
    );

    always #10 clk = ~clk;
    initial begin
        rst = 1; clk = 0;
        #15 rst = 0;
        #1000 $stop;
    end
endmodule
```

使用 pll 模組，並輸出相關信號。

■ 模擬結果與結果說明：

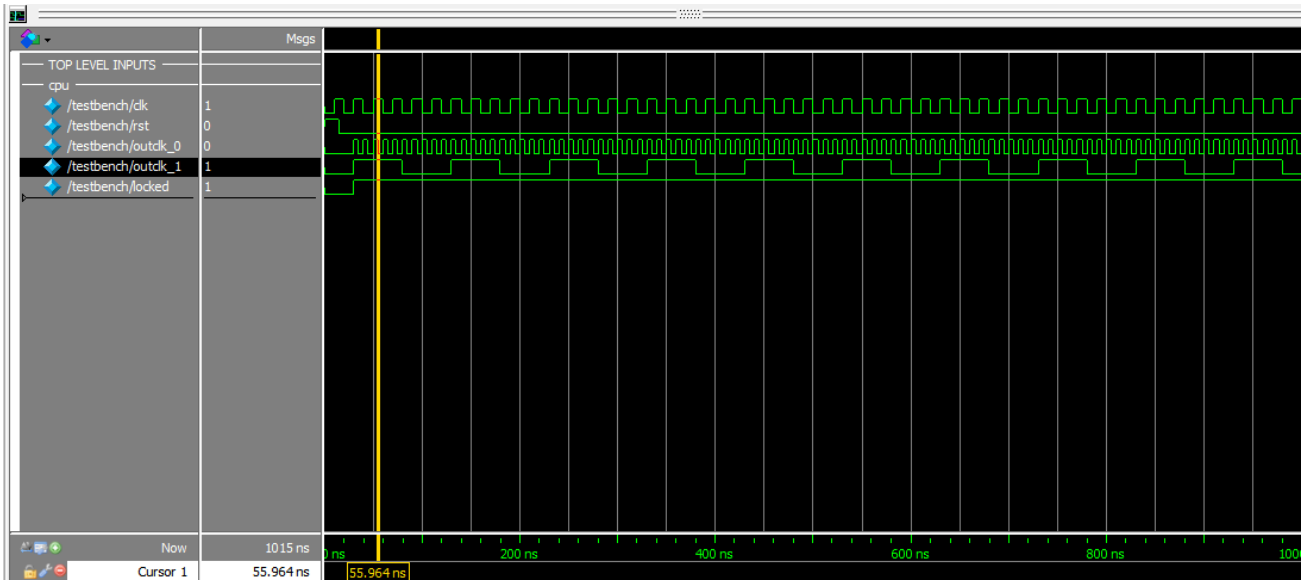
Signal Tap 觀察



可以看出在觸發事件(locked)前，outclk_0、outclk_1 不穩定。

觸發後，因為取的頻率不對 outclk_0 會沒有信號，而 outclk_1 的信號會很漂亮。

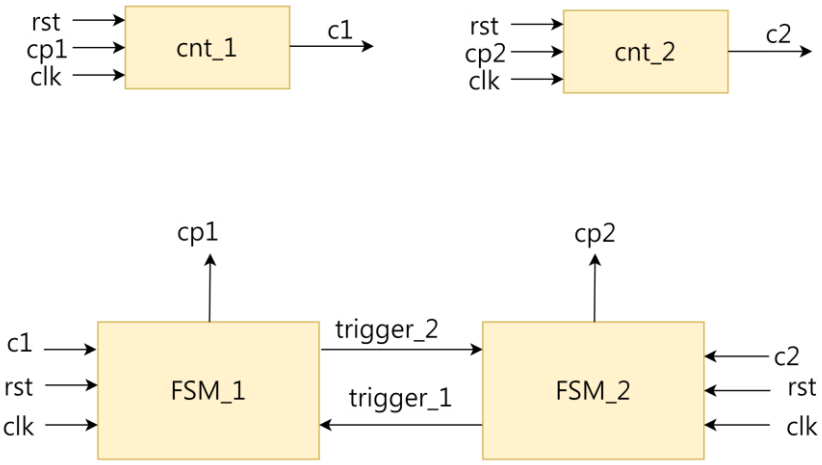
Modelsim 觀察



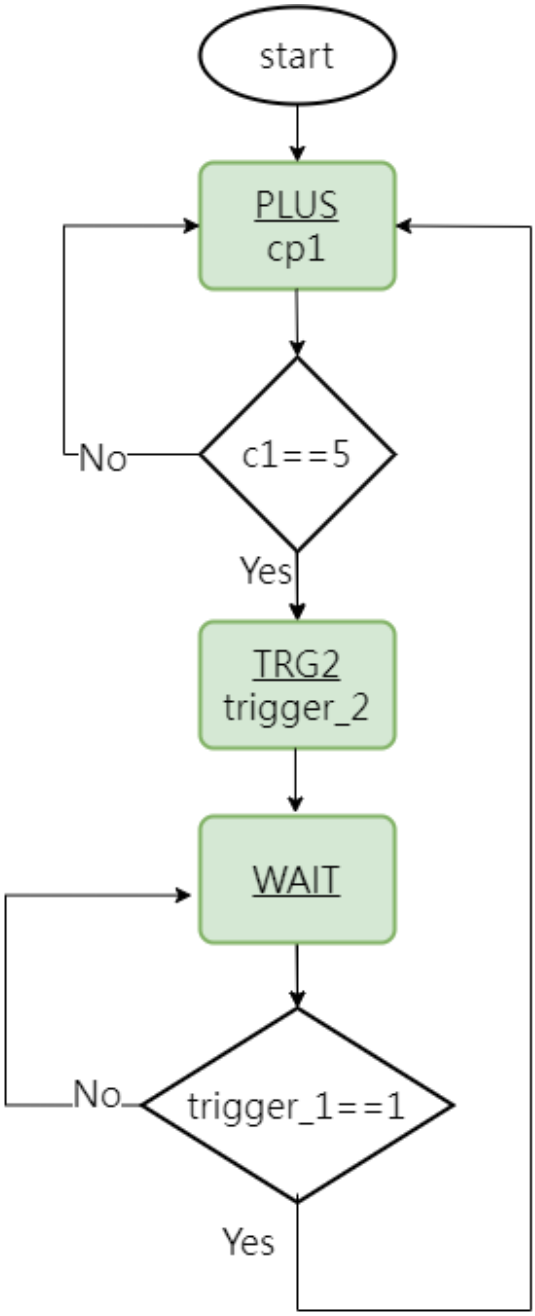
從波形可以看到不同頻率的 clk。

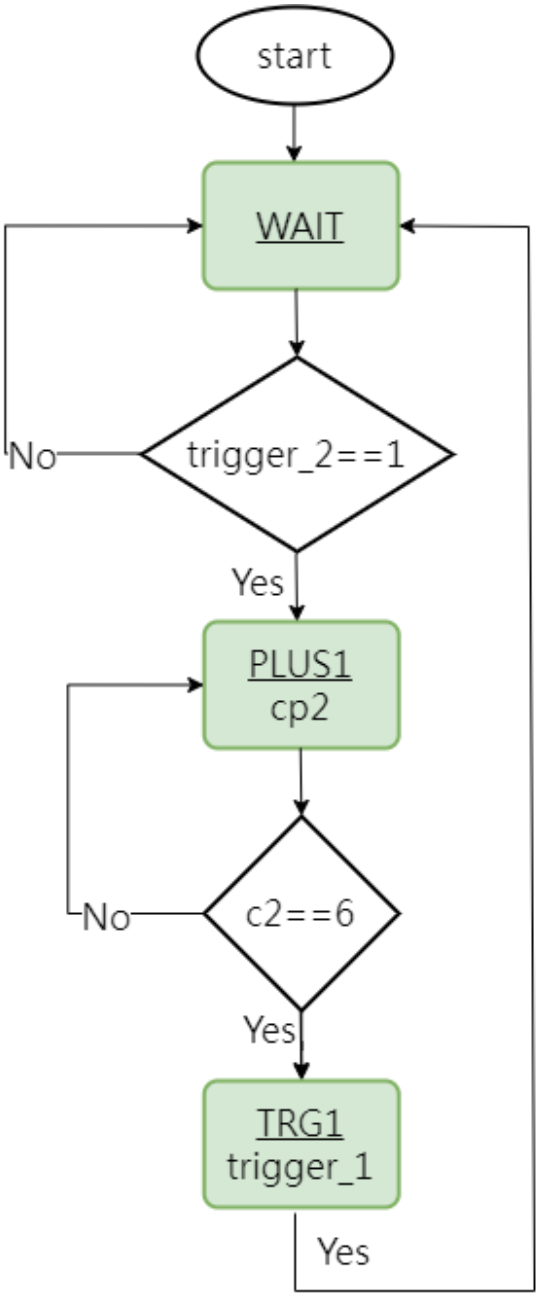
二、 Handshaking

- 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：


start
PLUS
cp1
c1==5
No
Yes
TRG2
trigger_2
WAIT
trigger_1==1
No
Yes


start
WAIT
trigger_2==1
No
Yes
PLUS1
cp2
c2==6
No
Yes
TRG1
trigger_1
Yes

STATE
OUT

■ 系統架構程式碼、測試資料程式碼與程式碼說明

Handshaking.sv

```
module handshaking (
    input rst,
    input clk,
    output logic [3:0] c1,
    output logic [3:0] c2
);

logic cp1, cp2, trigger_1, trigger_2;

typedef enum { START, PLUS_CNT1, PLUS_CNT2, TRG1, TRG2, WAIT_TRG1, WAIT_TRG2 } fsm_state;

fsm_state fsm1_ps, fsm1_ns, fsm2_ns, fsm2_ps;

always_ff @ (posedge clk) begin
    if (rst) fsm1_ps <= START;
    else fsm1_ps <= fsm1_ns;
end

always_comb
begin
    fsm1_ns = fsm1_ps;
    cp1 = 0;
    trigger_2 = 0;

    case (fsm1_ps)
        START: begin
            fsm1_ns = PLUS_CNT1;
        end
        PLUS_CNT1: begin
            if (c1 == 5) fsm1_ns = TRG2;
            else cp1 = 1;
        end
        TRG2: begin
            trigger_2 = 1;
            fsm1_ns = WAIT_TRG1;
        end
        WAIT_TRG1: begin
            if (trigger_1 == 1) begin
                cp1 = 1; // let counter1 countinue;
                fsm1_ns = PLUS_CNT1;
            end
        end
    endcase
end

always_ff @ (posedge clk) begin
```

```
    if (rst) fsm2_ps <= START;
    else fsm2_ps <= fsm2_ns;
end

always_comb
begin
    fsm2_ns = fsm2_ps;
    cp2 = 0;
    trigger_1 = 0;

    case (fsm2_ps)
        START: begin
            fsm2_ns = WAIT_TRG2;
        end
        WAIT_TRG2: begin
            if (trigger_2 == 1) begin
                cp2 = 1; // let counter2 countinue;
                fsm2_ns = PLUS_CNT2;
            end
        end
        PLUS_CNT2: begin
            if (c2 == 6) fsm2_ns = TRG1;
            else cp2 = 1;
        end
        TRG1: begin
            trigger_1 = 1;
            fsm2_ns = WAIT_TRG2;
        end
    endcase
end

always_ff @ (posedge clk) begin
    if (rst) c1 <= 0;
    else if (cp1) c1 <= c1 + 1;
end

always_ff @ (posedge clk) begin
    if (rst) c2 <= 0;
    else if (cp2) c2 <= c2 + 1;
end


endmodule
```

共宣告 4 個 flip-flop，其中兩個轉換 fsm 的 present state 與 next state，其中兩個為 counter，依照 cp1、cp2 信號來累加 c1,c2。

依照題目所述完成兩個 FSM。其中值得注意的是 WAIT_TRG1 與 WAIT_TRG2，cp1 與 cp2 在條件成立時信號為 1，原因是要讓兩個 FSM 數完一輪之後還可以繼續數，否則會卡在 PLUS_CNT1 與 PLUS_CNT2。

compile.do

```
#vlib work



# ----------------------------------------------------------
vlog ../tb/testbench.sv
vlog ../../design/handshaking.sv
vlog ../../design/sub_4bit.sv
vlog ../../design/DE0_CV.sv









# ----------------------------------------------------------
```

編譯 handshaking.sv 與 testbench.sv 檔案。

Wave.do

```
onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate /testbench/rst
add wave -noupdate /testbench/clk
add wave -noupdate -radix hexadecimal /testbench/c1
add wave -noupdate -radix hexadecimal /testbench/c2
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {121 ps} 0}
quietly wave cursor active 1
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
configure wave -timelineunits ps
update
WaveRestoreZoom {0 ps} {1058 ps}
```

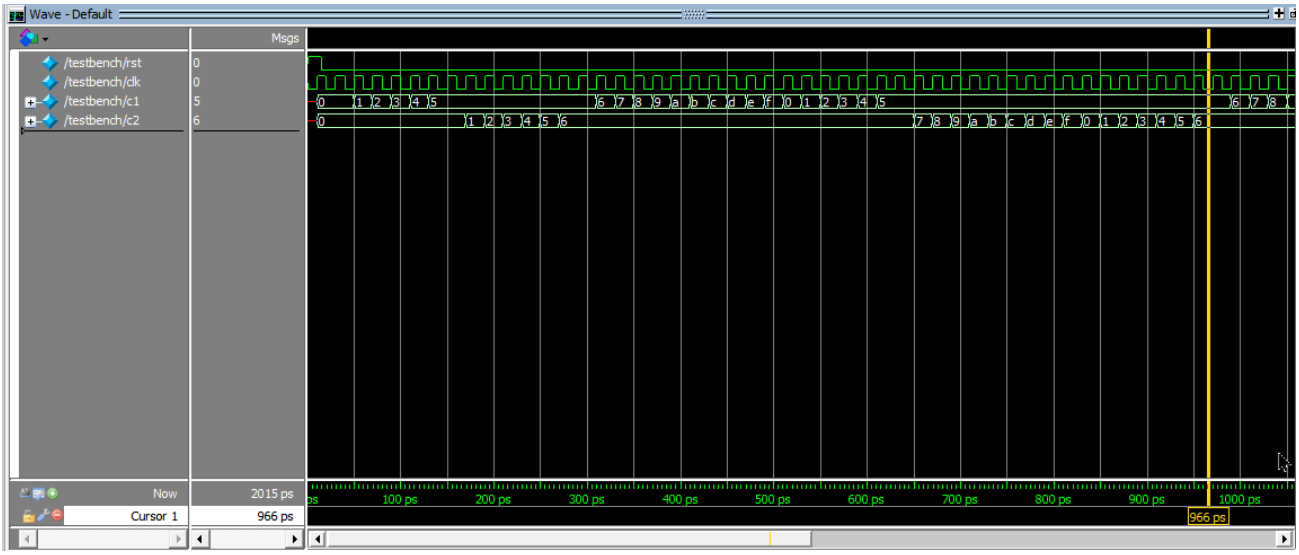
將 clk、rst、c1、c2 的接角輸出。

Testbench.sv

```
module testbench;

    logic rst;
    logic clk;
    logic [3:0] c1;
    logic [3:0] c2;

    handshaking u1 (
        .rst(rst),
        .clk(clk),
        .c1(c1),
        .c2(c2)
    );

    always #10 clk = ~clk;

    initial begin
        rst = 1; clk = 0;
        #15 rst = 0;
        #2000 $stop;
    end
endmodule
```

每 10 單位時間反向 clk 一次，一開始 rst = 1 用來做初始化，15 個單位時間後 rst 設回 0。

■ 模擬結果與結果說明：

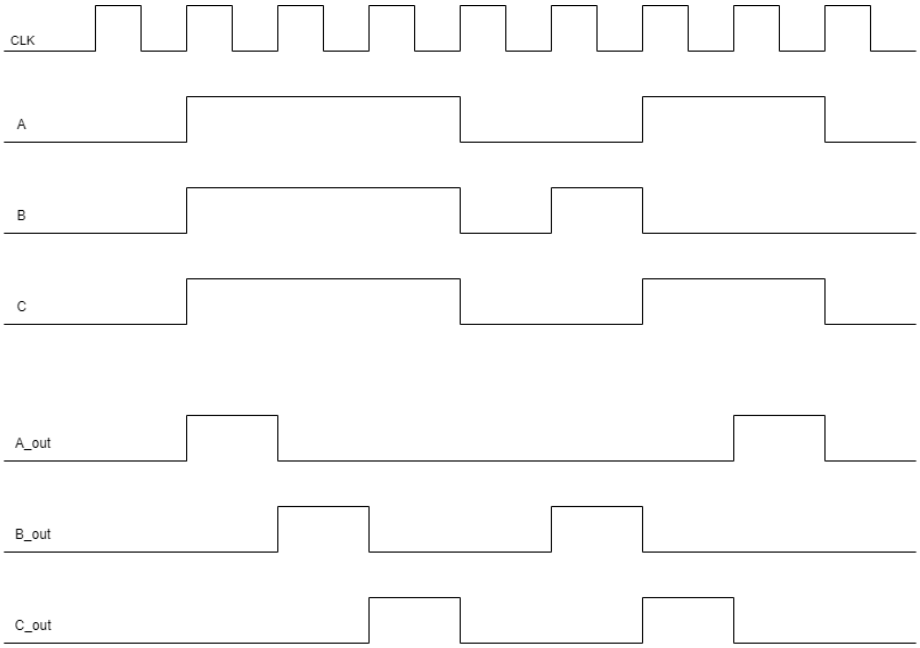


從波型可以看出 c1 從 0 數到 5 觸發 trigger 使 c2 開始數，c2 數到 6 觸發 trigger 使 c1 繼續數。

三、 仲裁

■ 實驗說明：

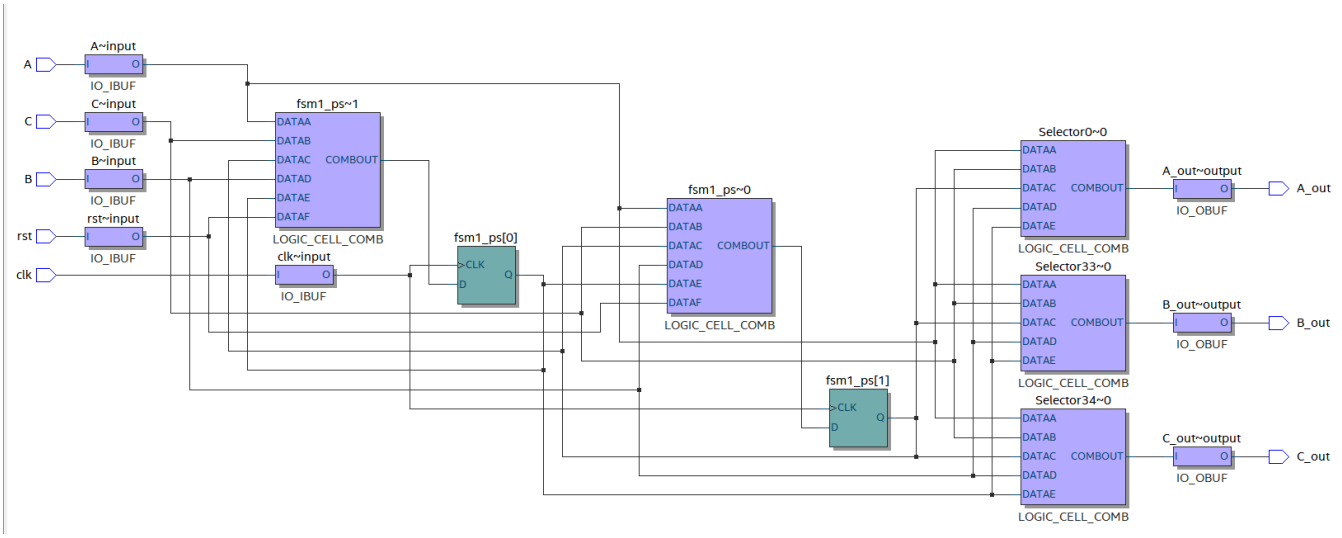
1. 除了 CLK 外有 3 個 1bit 輸入：A、B、C 和 3 個 1bit 輸出：A_out、B_out、C_out。
2. 當 A=1 時 A_out 輸出會 1 個 CLK 的 1（B、C 同理)，但同時只會有一個輸出為 1，所以必須考慮優先順序。
3. 輸出的優先順序為 A_out > B_out > C_out 。



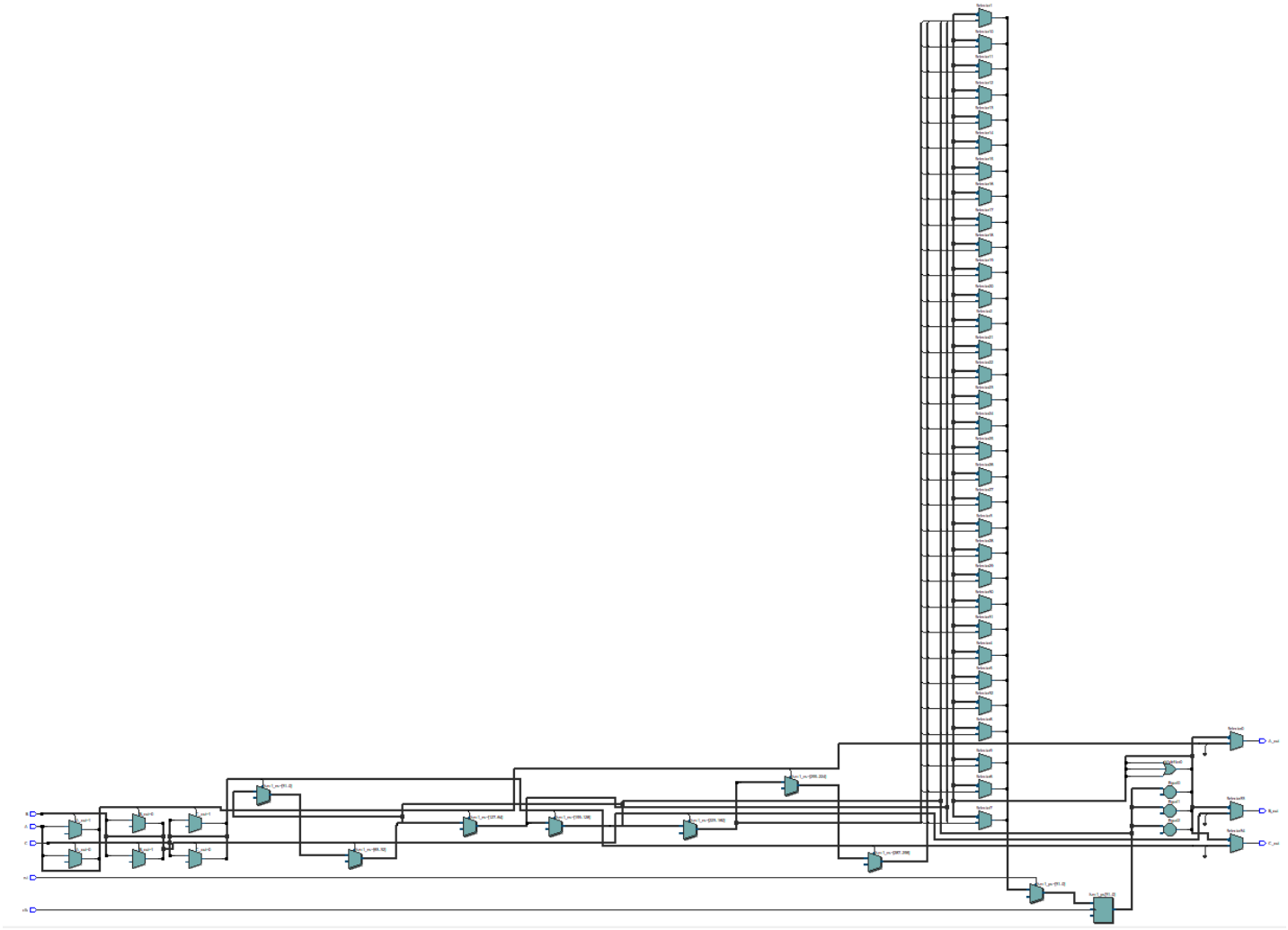
注意 B_out 第二次輸出 1 的地方，因這時的狀態已經判斷到 B，所以下一個輸出是 C_out 而不是 A_out。

■ 系統硬體架構方塊圖（接線圖）：

Technology map viewer(port-mapping)



RTL Viewer



■ 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

Lab3_3.sv

```
module lab3_3 (
    input clk,
    input rst,
    input A,
    input B,
    input C,
    output logic A_out,
    output logic B_out,
    output logic C_out
);

typedef enum { T0, T1, T2 } fsm1_state_enum;

fsm1_state_enum fsm1_ns, fsm1_ps;

always_ff @(posedge clk) begin
    if (rst)
        fsm1_ps <= T0;
    else
        fsm1_ps <= fsm1_ns;
end

always_comb begin
    fsm1_ns = fsm1_ps;
    A_out = 0;
    B_out = 0;
    C_out = 0;
    case (fsm1_ps)
        /*
        T0 ~ T2 are different priorities.
        T0: A > B > C;
        T1: B > C > A;
        T2: C > A > B;
        */
        T0: begin
            if (A) begin
                A_out = 1;
                fsm1_ns = T1;
            end else if (B) begin
                B_out = 1;
                fsm1_ns = T2;
            end else if (C) begin
                C_out = 1;
                fsm1_ns = T0;
```

```
            end
        end

        T1: begin
            if (B) begin
                B_out = 1;
                fsm1_ns = T2;
            end else if (C) begin
                C_out = 1;
                fsm1_ns = T0;
            end else if (A) begin
                A_out = 1;
                fsm1_ns = T1;
            end
        end
        T2: begin
            if (C) begin
                C_out = 1;
                fsm1_ns = T0;
            end else if (A) begin
                A_out = 1;
                fsm1_ns = T1;
            end else if (B) begin
                B_out = 1;
                fsm1_ns = T2;
            end
        end

    endcase
end

endmodule
```

依照題目要求宣告 input、output 接角，之後定義三個 state，T0、T1、T2，為不同優先序的狀態，T0 為 A->B->C；T1 為 B->C->A；T2 為 C->A->B，並在 fsm 中以 if elseif else 優先序的概念實作。且宣告 flip flop 以 clk 控制 state。

compile.do

```
#vlib work



# ----------------------------------------------------------------
vlog ../tb/testbench.sv
vlog ../../design/lab3_3.sv
vlog ../../design/sub_4bit.sv
vlog ../../design/DE0_CV.sv

```

編譯 lab3_3.sv 與 testbench.sv 檔案。

Wave.do

```
onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -divider {TOP LEVEL INPUTS}
add wave -noupdate -divider cpu
add wave -noupdate /testbench/clk
add wave -noupdate /testbench/rst
add wave -noupdate /testbench/A
add wave -noupdate /testbench/B
add wave -noupdate /testbench/C
add wave -noupdate /testbench/A_out
add wave -noupdate /testbench/B_out
add wave -noupdate /testbench/C_out
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {42 ps} 0}
quietly wave cursor active 1
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
configure wave -timelineunits ns
update
WaveRestoreZoom {0 ps} {626 ps}

```

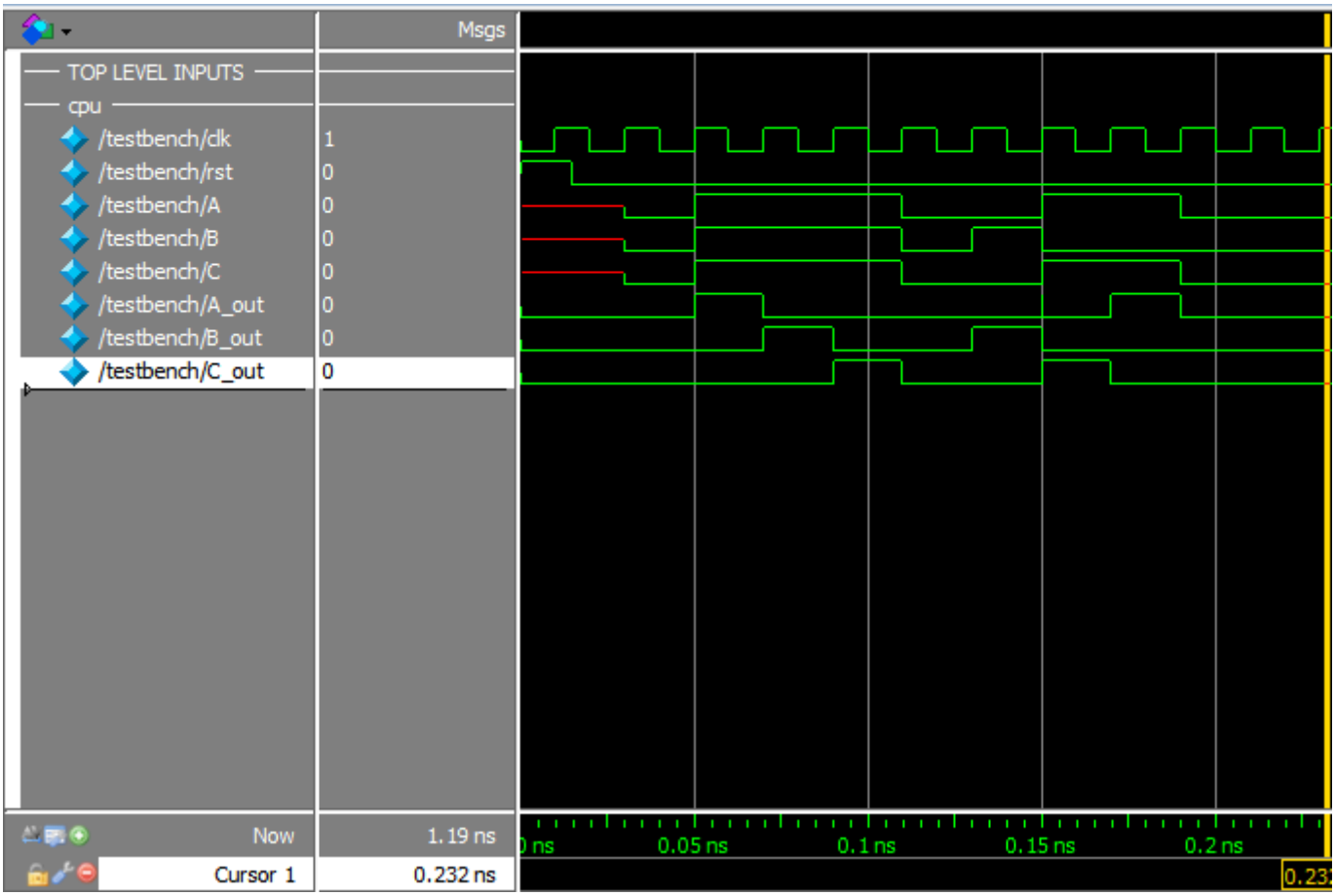
將 clk、rst、A、B、C、A_out、B_out、C_out 的接角輸出。

Testbench.sv

```
module testbench;

    logic clk;
    logic rst;
    logic A;
    logic B;
    logic C;
    logic A_out;
    logic B_out;
    logic C_out;

    lab3_3 u1(
        .clk(clk),
        .rst(rst),
        .A(A),
        .B(B),
        .C(C),
        .A_out(A_out),
        .B_out(B_out),
        .C_out(C_out)
    );

    always #10 clk = ~clk;

    initial begin
        rst = 1; clk = 0;
        #15 rst = 0;
        #15 A = 0; B = 0; C = 0;

        #20 A = 1; B = 1; C = 1;
        #60 A = 0; B = 0; C = 0;
        #20 B = 1;
        #20 A = 1; B = 0; C = 1;
        #40 A = 1; B = 1; C = 1;

        A = 0; B = 0; C = 0;

        #1000 $stop;
    end
endmodule
```

每 20 單位時間為 clk 的一個週期，依照題目將 A、B、C 賦值。

■ 模擬結果與結果說明：



由上圖可看出輸出波形與 LAB 中給的波型相同。

■ 結論與心得：

這次上課學到很多以後應該知道的常識，像是 FPGA 內部調節頻率的 PLL 與燒入在 FPGA 內部的 single tab。尤其是 single tab，利用板子給的 usb 線回傳真實的信號，覺得很新奇！使用 single tab 偵測實際信號時很順利，但在使用 Modelsim 模擬的時候遇到很多問題，希望下次的實驗可 已更加順利的完成！