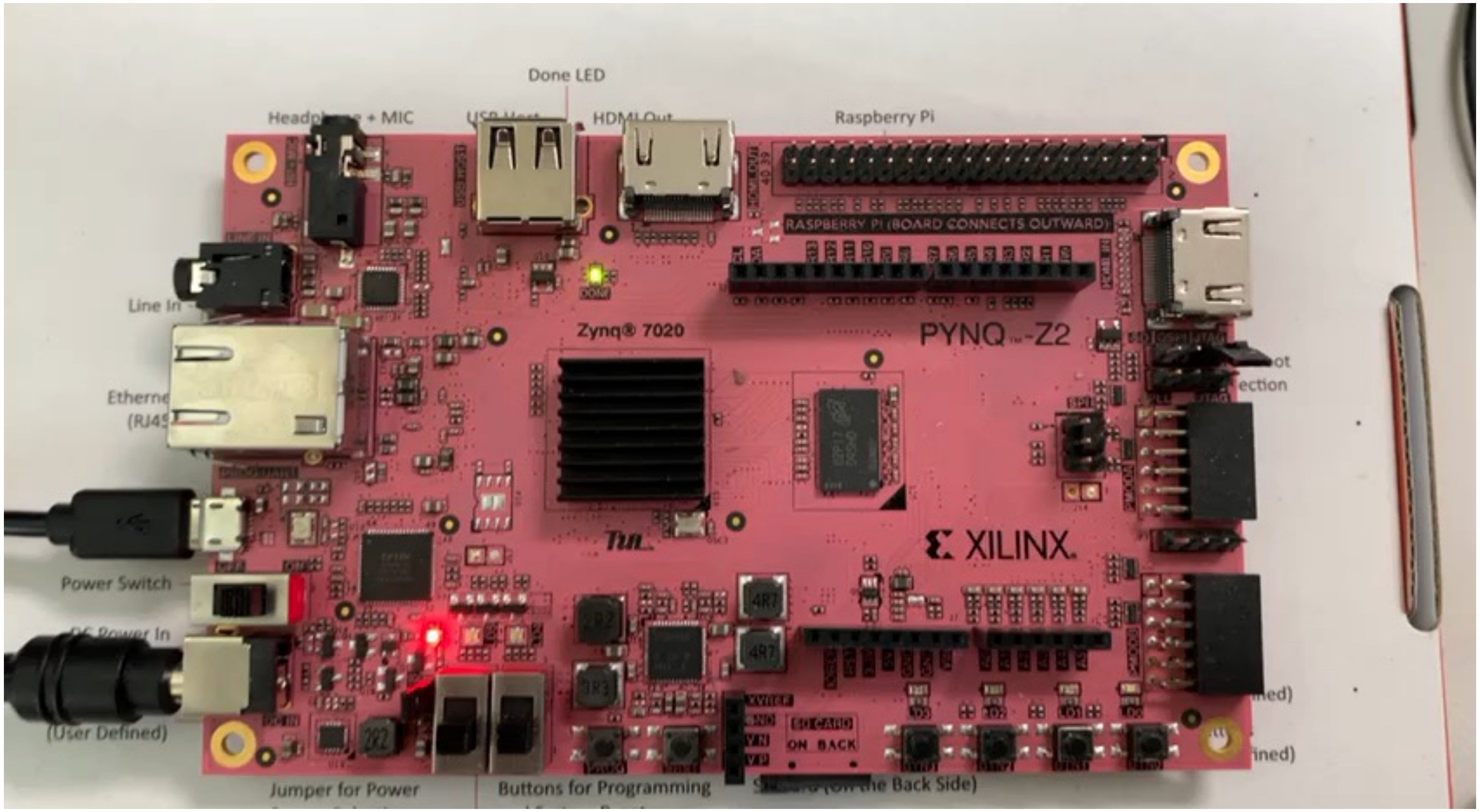
PWM呼吸燈

國立成功大學 電機系

2022

呼吸燈

1. 白光呼吸燈
2. 混色光呼吸燈

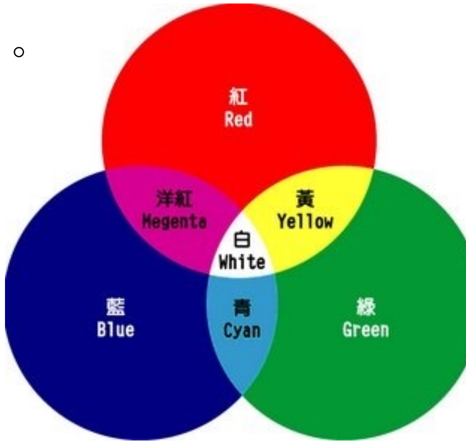


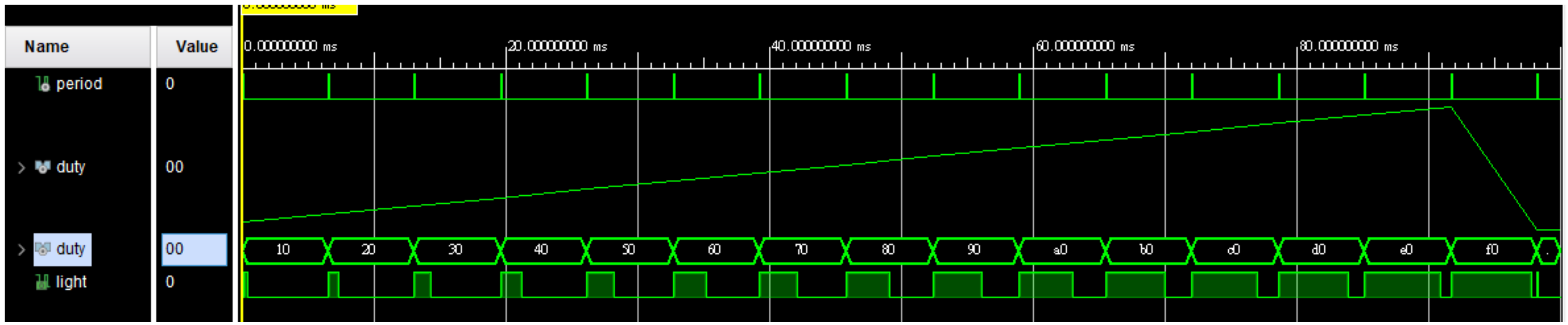
視覺暫留

1. 人眼的圖像滯留時間約為40ms(1/24幀畫面)。在此其間的變化無法被察覺。

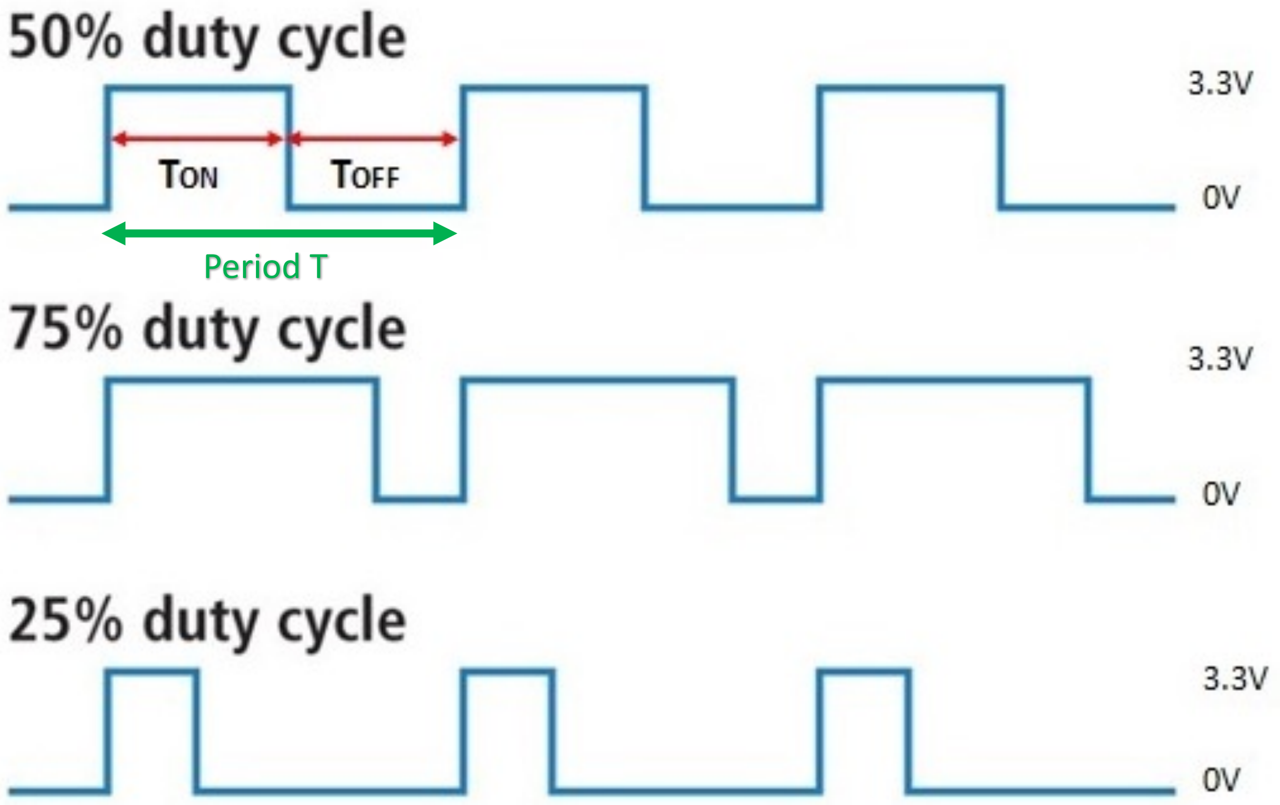
2. 40ms內LED on和off所占的百分比在視覺上會出現亮與暗的區別。

3. 紅、綠、藍光可獨立刺激眼睛的受光體，人眼基於。
   (ex, 紅色與綠色混合可以產生黃色或橙色，
   三種原色以等比例疊加在一起時則會呈現白色。)





Duty cycle



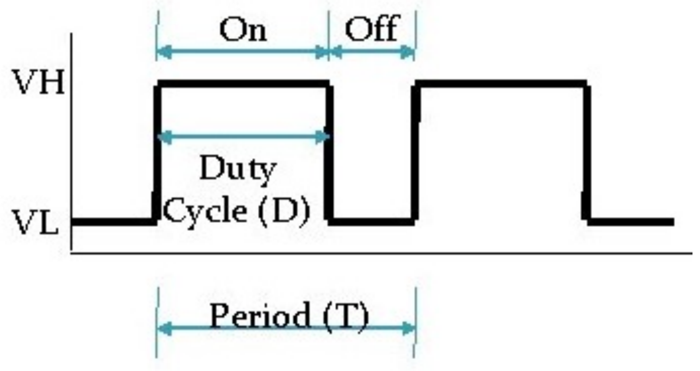
$$\text{Duty} = \frac{T_{ON}}{Period\ T}$$

Pulse Wave Modulation

- Pulse-wave Modulation (PWM) Uses a rectangular pulse wave whose pulse width is modulated resulting in the variation of the average value of the waveform.
- Average value of a pulse waveform f(t) with period T and low value V_L high V_H can be found as

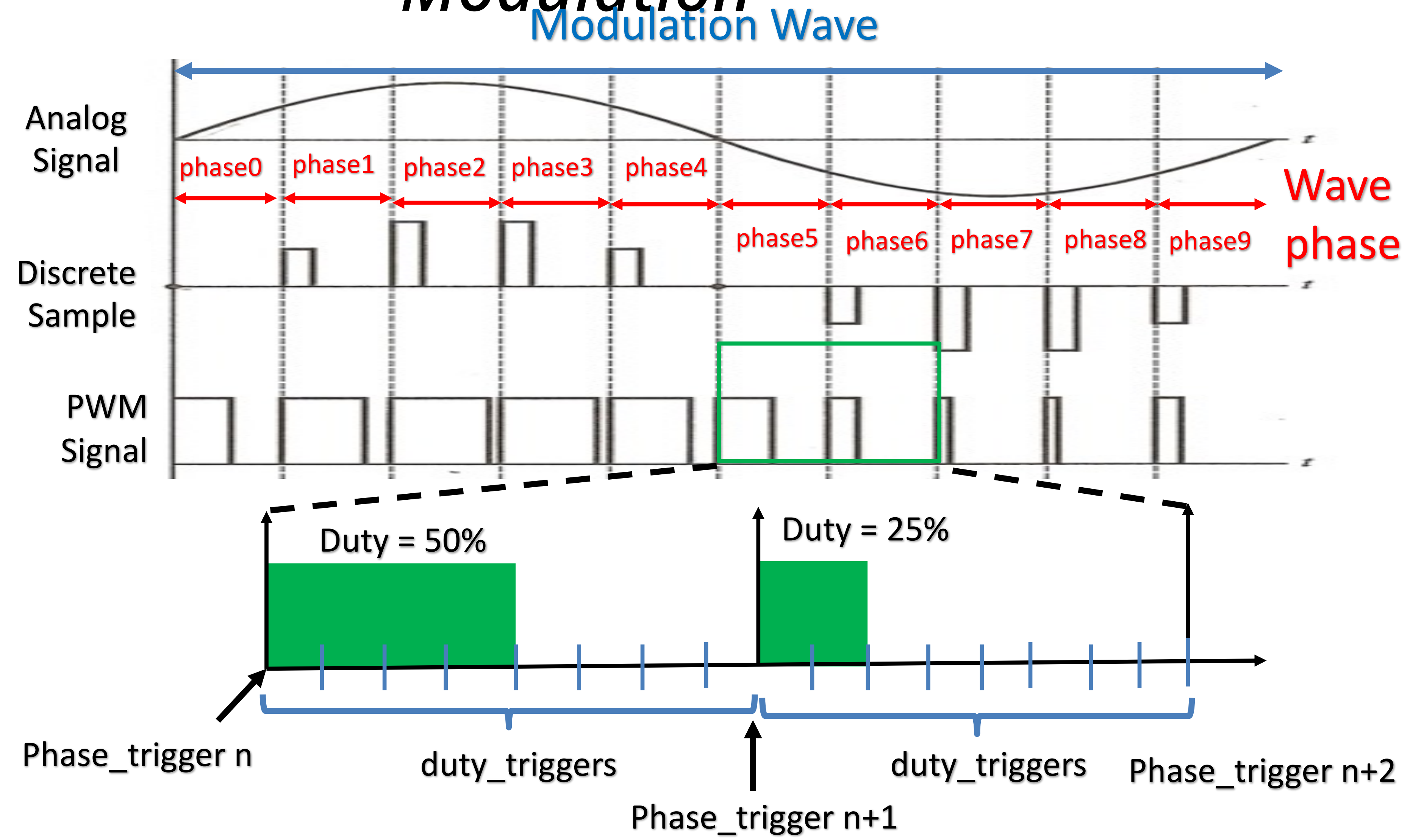
$$\bar{y} = \frac{1}{T}\int_0^T f(t)dt$$

$$V_{avg} = D \cdot V_H + (1 - D) \cdot V_L$$

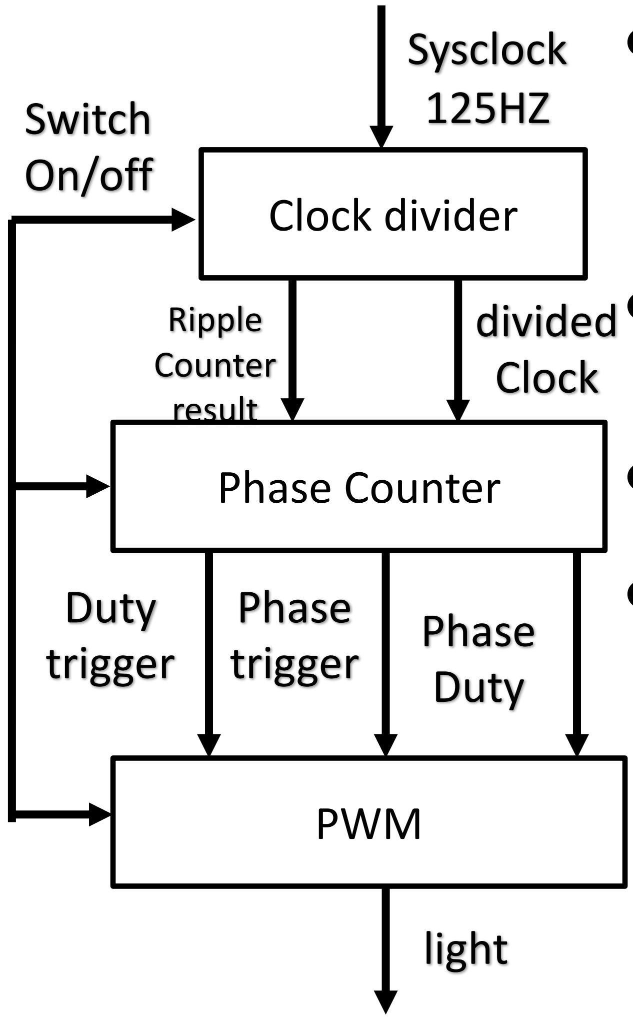


- In general VL is zero; D is the duty; $V_{avg} = D \times V_H$
- The average value of the signal is directly dependent on the duty cycle D.
- Output signal alternates between on and off within a specified period.

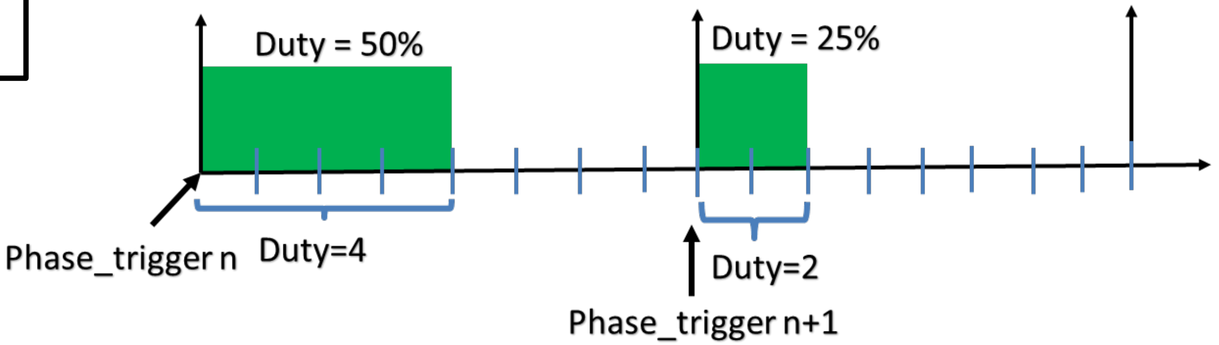
Lead Edge Modulation



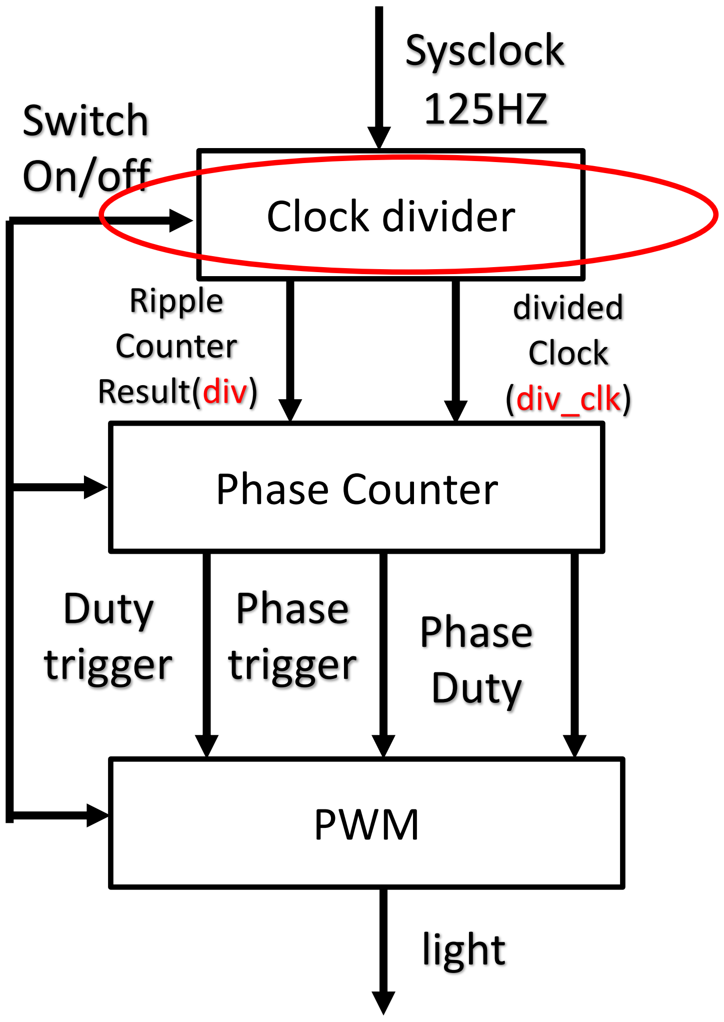
Simple Lead Edge Modulation



- 由於125MHZ的Sysclk頻率太高，請嘗試以 Clock Divider 將頻率降至適合操作的頻率。
  (Hint : divided clock frequency = $\frac{125MHZ}{2^n}$)
- 當Phase trigger的週期小於眼睛圖像滯留時間(約為40ms)，可減少目視閃爍的現象。
- Phase trigger週期須為Duty trigger週期的倍數。
- Duty 為每次phase trigger後維持高位的cycle
  假設Phase trigger週期為8 cycle, phase duty = 4 cycle
  那PWM 的 Duty = 50%。

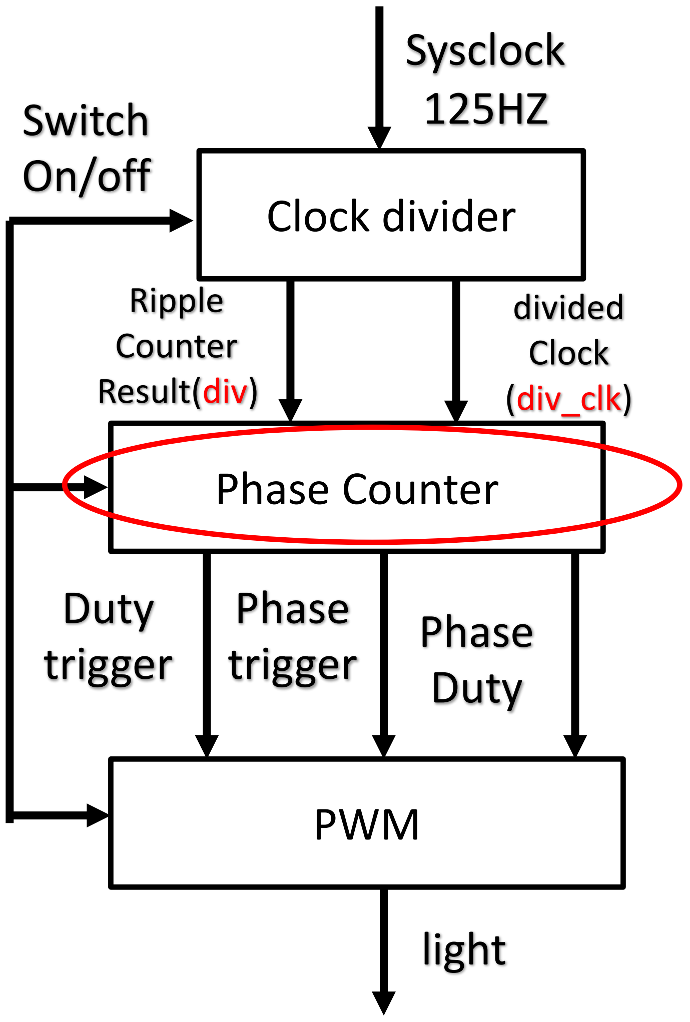


Clock Divider



```
module ripple_div(
    input rst,
    input sysclk,
    output wire [31:0] div,
    output wire div_clk
    );
//31
dff_div d0( .rst(rst),  .trigger(sysclk), .div(div[0]));
dff_div d1( .rst(rst),  .trigger(div[0]), .div(div[1]));
…
…
…
dff_div d31( .rst(rst),  .trigger(div[30]), .div(div[31]));
endmodule
```

Phase Counter

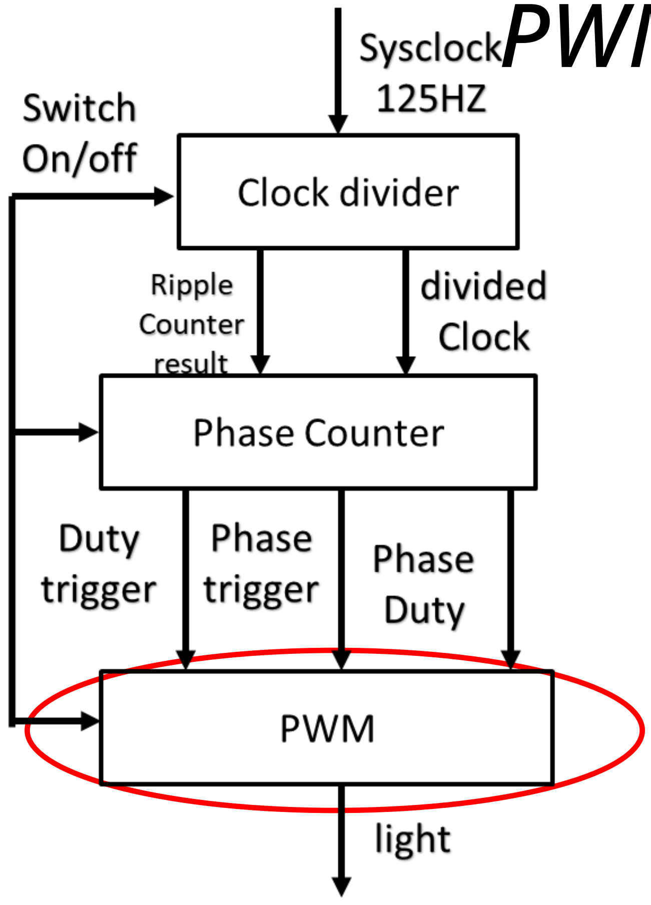


```
module phase_counter(
    input rst,
    input enable,
    input [7:0] phase_shift,
    input[31:0] div,
    input  div_clk,
    output wire duty_trig,
    output wire phase_trig,
    output wire [7:0] phase
);

reg [7:0] duty;
assign duty_trig  = (div[7:0] == 8'hff) & enable;
assign phase_trig = (div[15:0] == 16'hff) & enable;
assign phase      =  div[27:20] + phase_shift;

endmodule
```

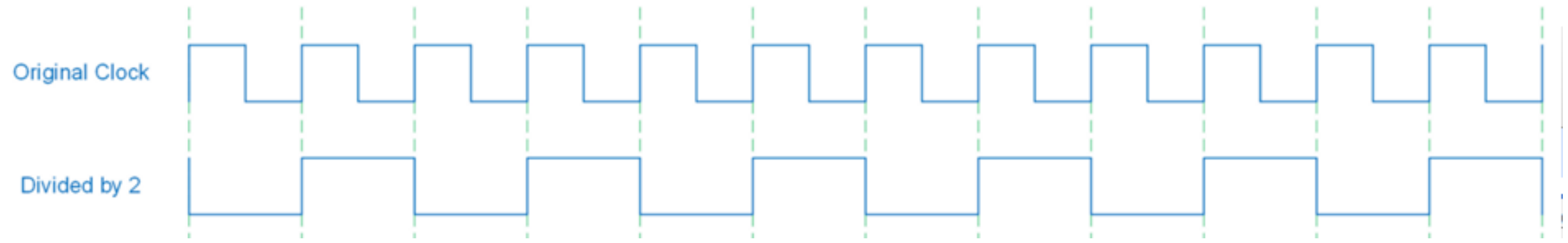
8 bit phase PWM



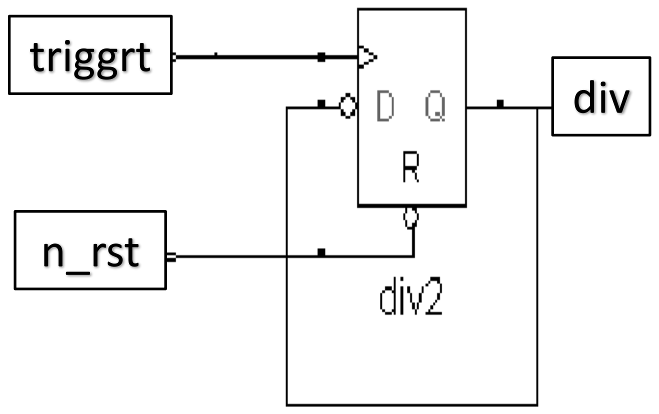
256 duty cycles between Phase_triggers

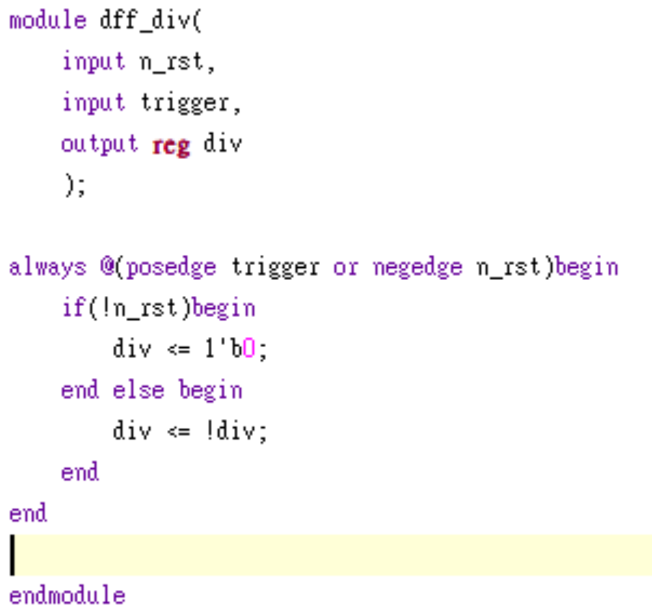
```
module pwm(
    input rst,
    input enable,
    input duty_trig,
    input phase_trig,
    input [7:0] phase,
    output wire out
);
reg[7:0] count;
assign out = ((count < phase) | phase_trig) & enable;
always@(posedge duty_trig or posedge rst)begin
    if(rst)begin
        count <= 8'b0;
    end else if(phase_trig)begin
        count <= 8'b0;
    end else if(duty_trig)begin
        count <= count + 1;
    end else begin
        count <= count;
    end
end
endmodule
```

Clock Divider1



*Divide clock **frequency** by 2*





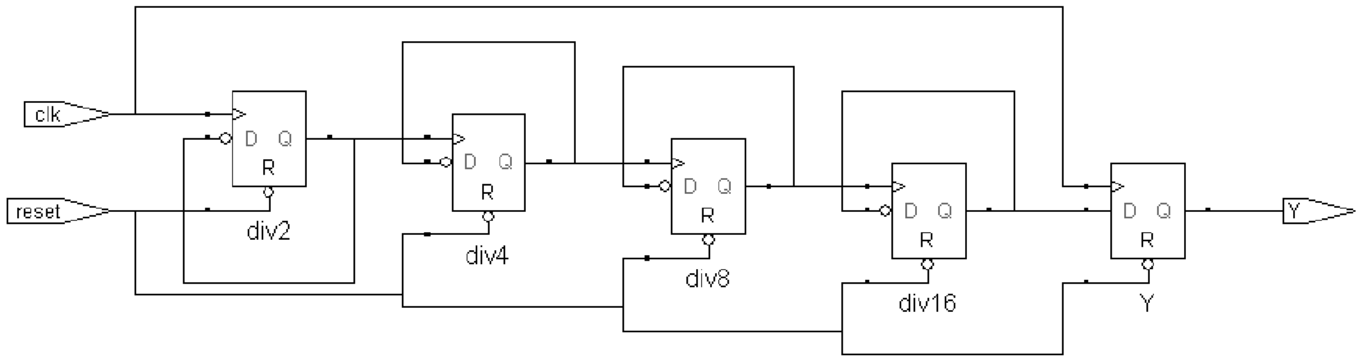
```
module dff_div(
    input n_rst,
    input trigger,
    output reg div
    );

always @(posedge trigger or negedge n_rst)begin
    if(!n_rst)begin
        div <= 1'b0;
    end else begin
        div <= !div;
    end
end

endmodule
```

Clock Divider2

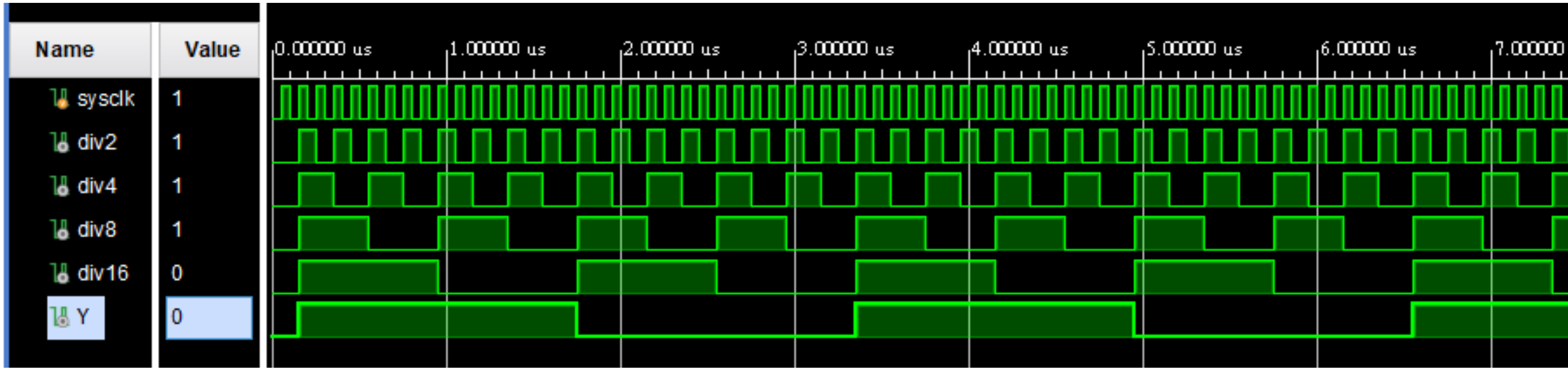
Clock divide by 2^n (n DFF個數)



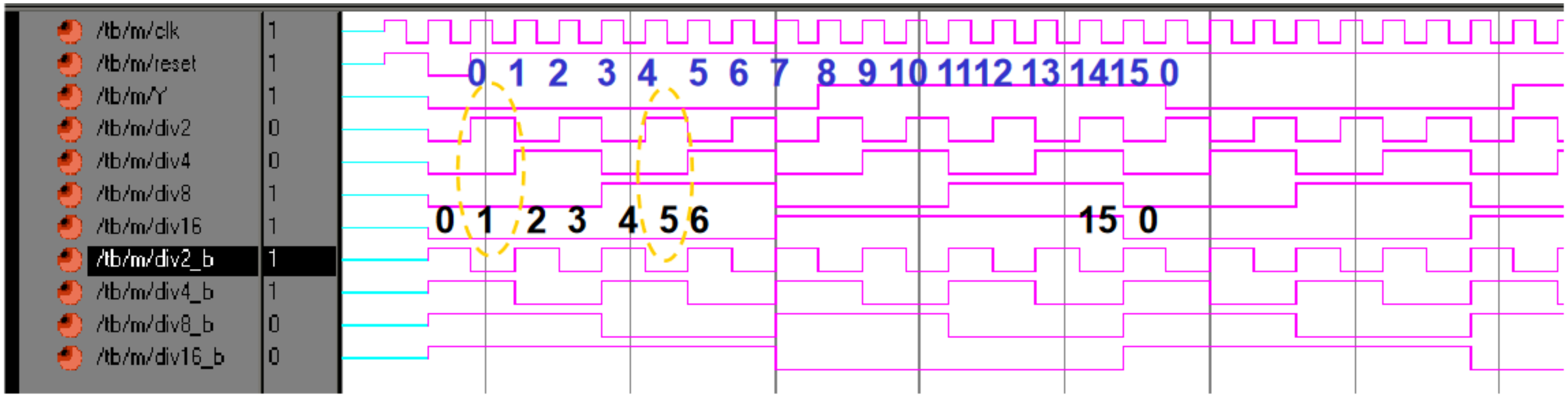
Divided clock frequency

$$freqY = \frac{125MHZ}{2^n}$$

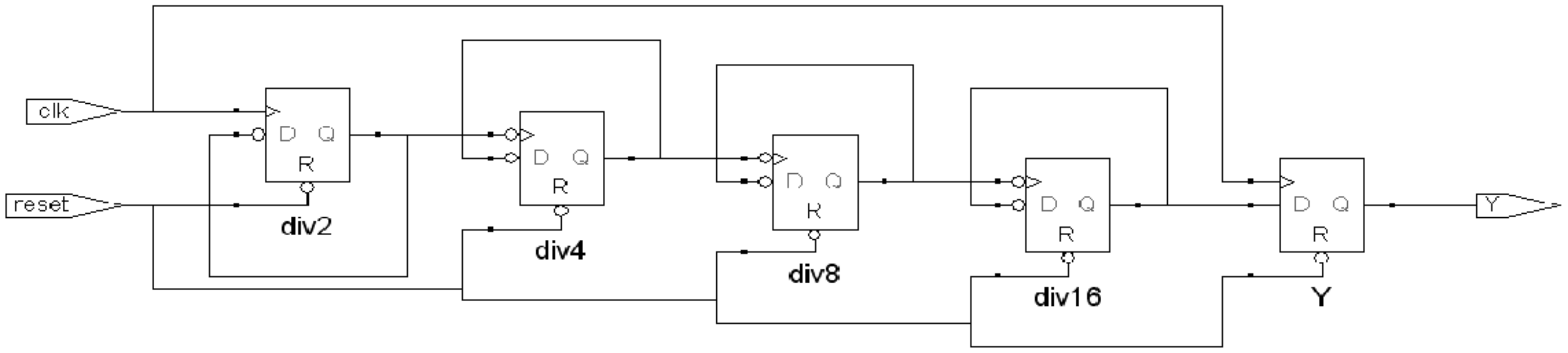
(n=#DFF)



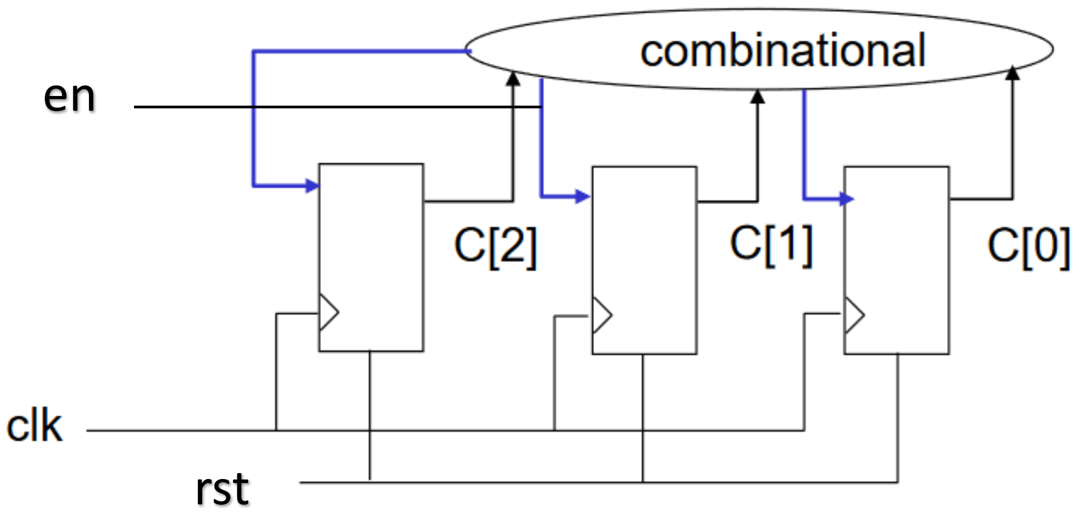
Ripple Counter



Count-up counter 0➔1➔2➔…….15➔0➔1➔2➔…..



Synchronous counter

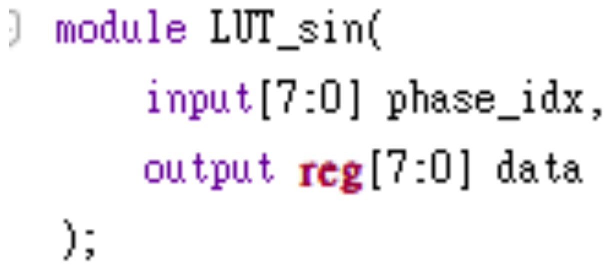


C(old)	C(new)
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
….	
1 1 0	1 1 1
1 1 1	0 0 0

```
always@(posedge clk or posedge rst)begin
    if(rst)begin
        C <= 3'b0;
    end else if(en)begin
        C <= C + 3'b1;
    end else begin
        C <= C;
    end
end
```

Digital Sinusoidal wave

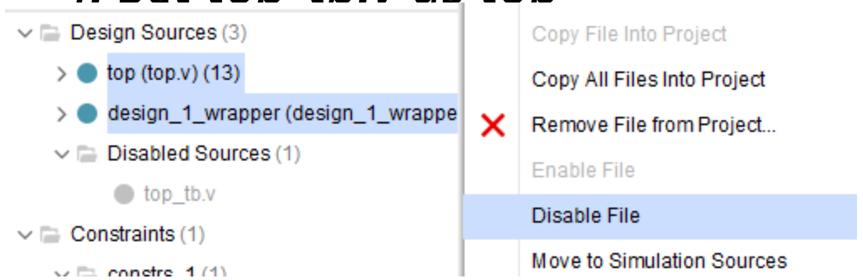
/*NOTE
LUT_sin table is a wave look up table with 8 bit index from 0~255
the range of output data 0 ~ 255
*/

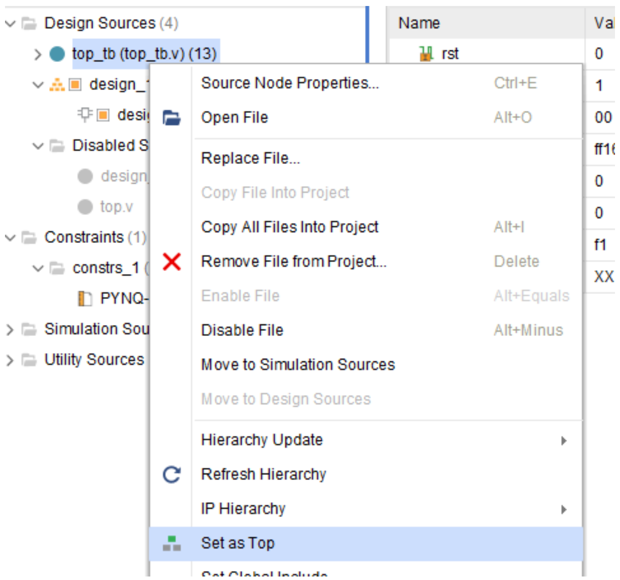


```
module LUT_sin(
    input[7:0] phase_idx,
    output reg[7:0] data
);
```

Simulation

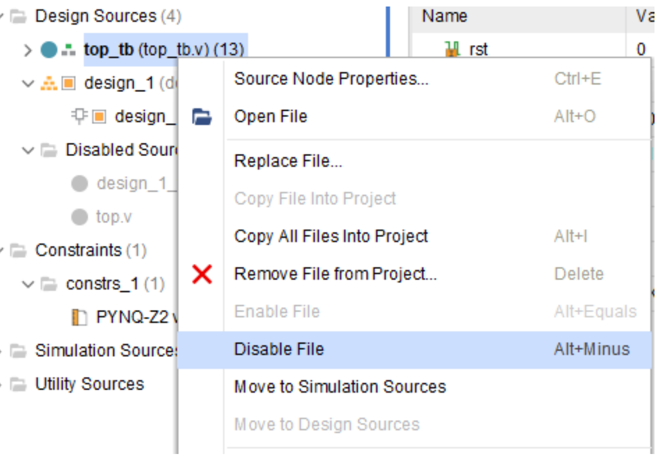
1. Disable top.v
2. Disable design_1_wrapper.v
3. Enable top_tb.v
4. Set top_tb.v as top

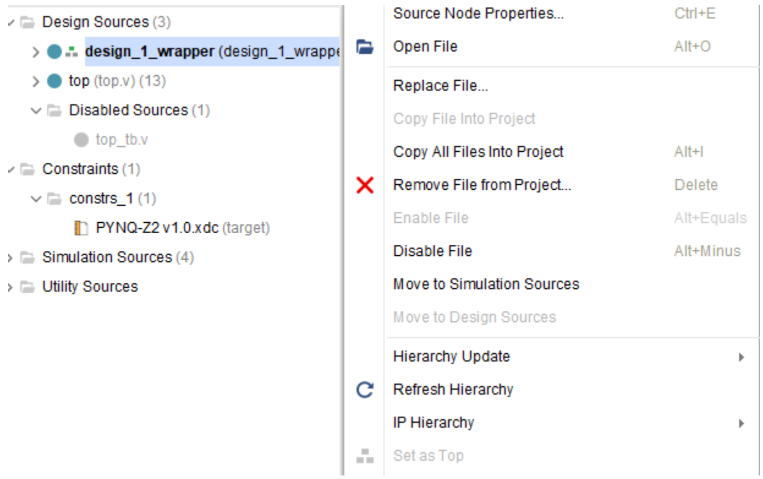




FPFA synthesis

1. Enable top_tb.v
2. Enable top.v
3. Enable design_1_wrapper.v
4. Set design_1_wrapper.v as top

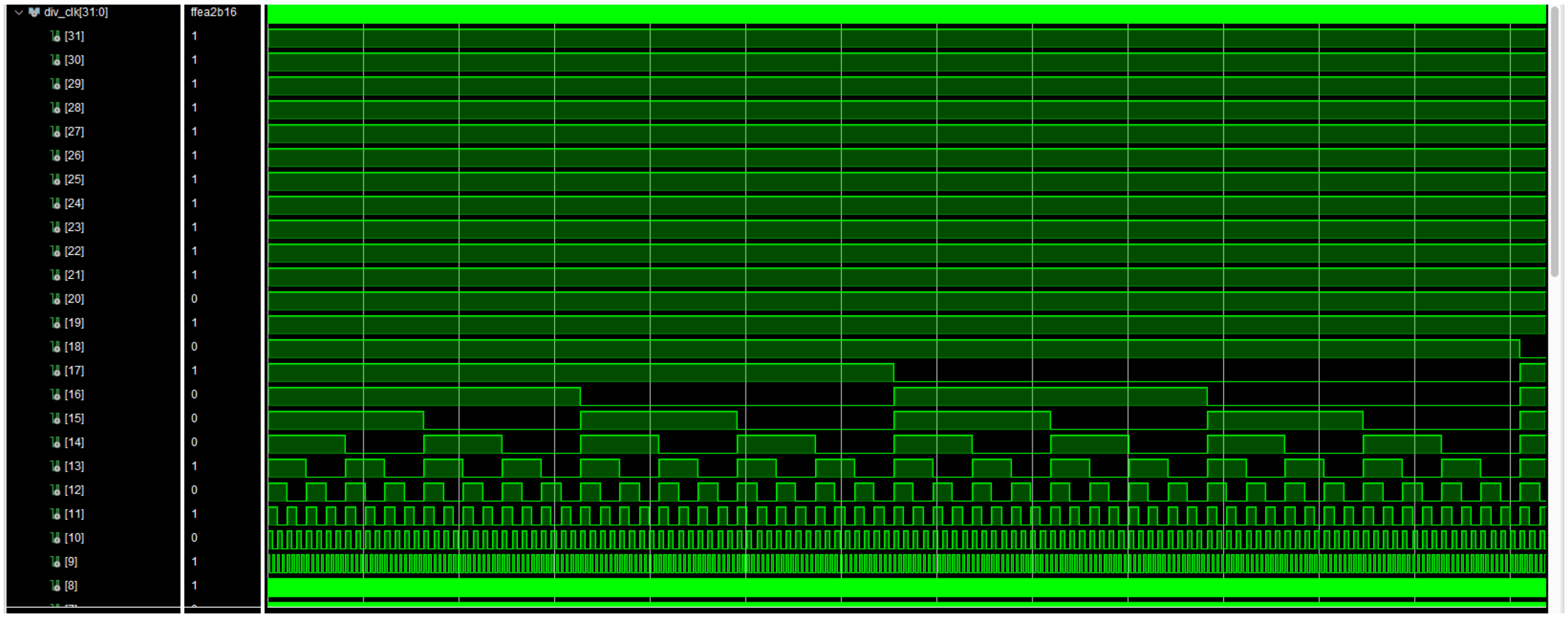




基礎題

產生一組clock divider之波型，結報需解釋波型的產生方法與波型圖擷圖(如下圖)。

以及開發版LED4及LED5之成果照片(白燈，及紅綠藍色燈)



```
div_clk[31:0]   ffea2b16
  [31]          1
  [30]          1
  [29]          1
  [28]          1
  [27]          1
  [26]          1
  [25]          1
  [24]          1
  [23]          1
  [22]          1
  [21]          1
  [20]          0
  [19]          1
  [18]          0
  [17]          1
  [16]          0
  [15]          0
  [14]          0
  [13]          1
  [12]          0
  [11]          1
  [10]          0
  [9]           1
  [8]           1
```

加分題

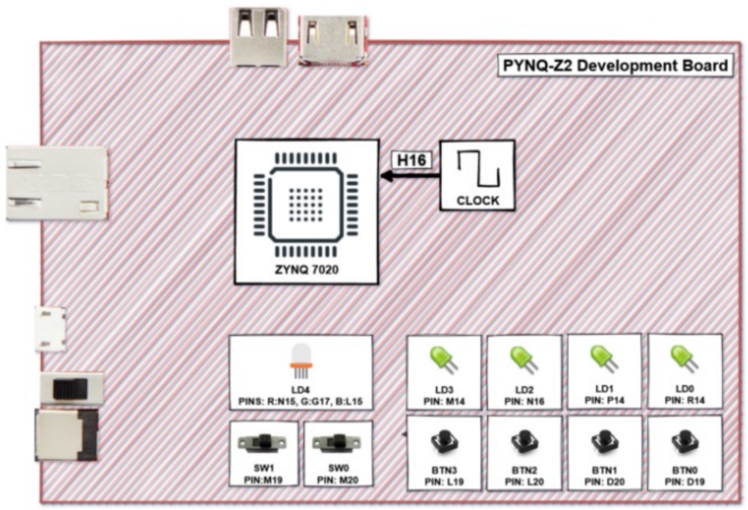
請解釋TOP.v所呼叫的各函式之程式碼

越詳細分數越高

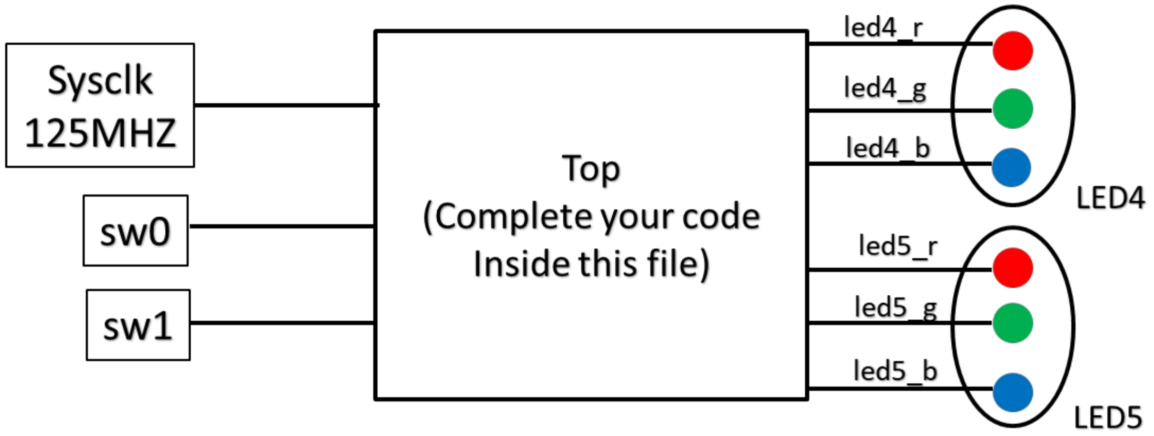
沒寫不扣分

FPGA(60%)

同學需在lab9_Breathing_LED的專案內的top.v檔實作呼吸燈電路。同學可依需求修改top檔案內非port name的部分(切記不可修改top.v 的port name，不然會合成失敗)。接著合成Bitstream檔並燒入到FPGA內。



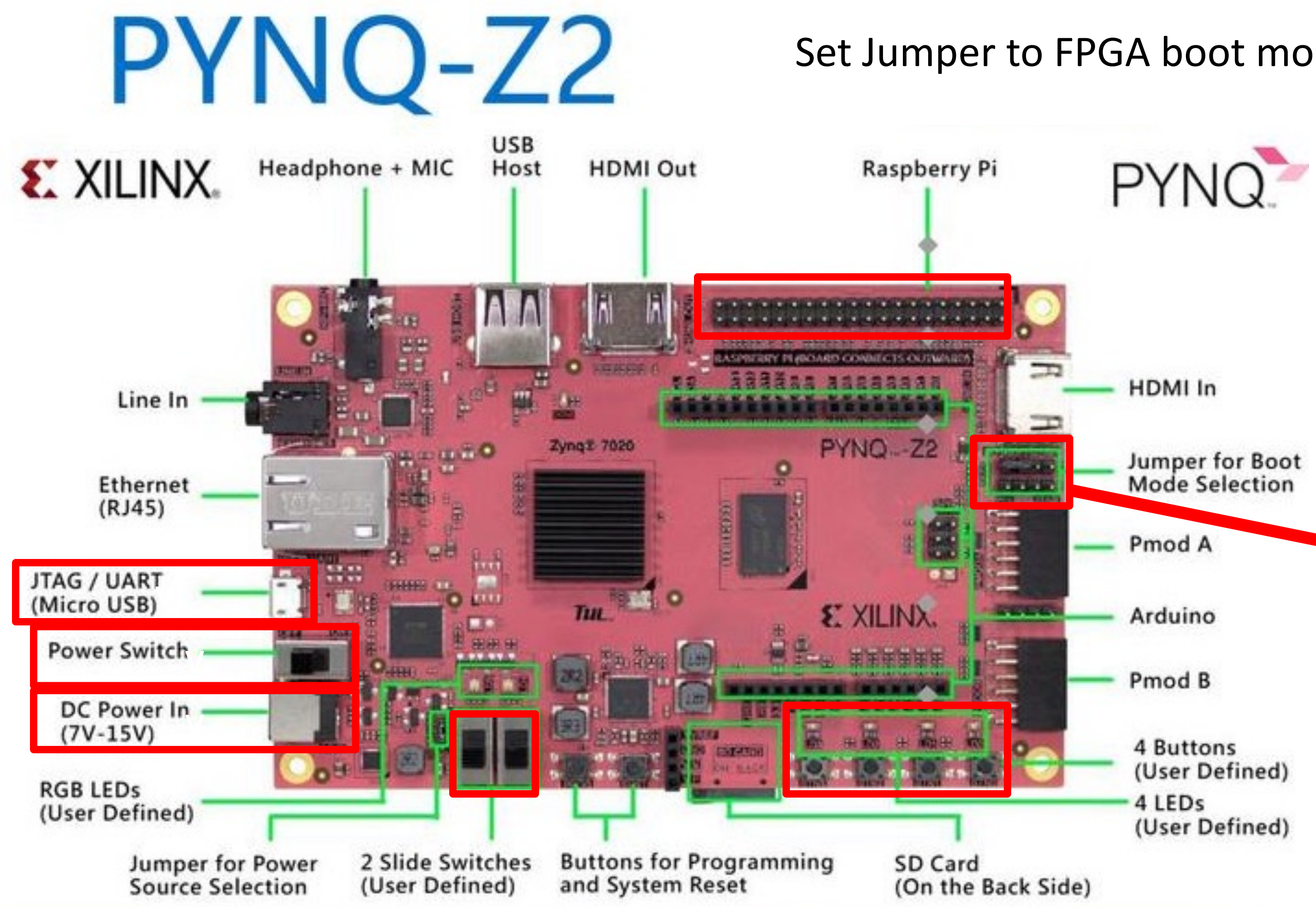
- Note : FPGA內LED4、LED5內有三種獨立RGB LED燈。同學可自行挑選。沒有使用的output線路記得要在top module內接地。(Assign to 0)。

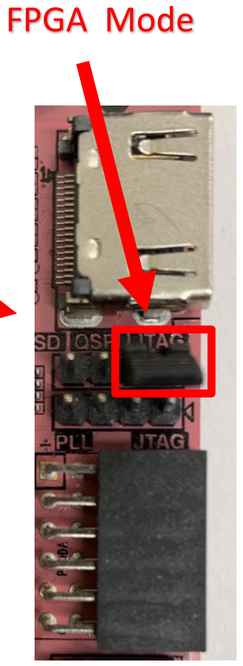


```
module top(
    input sysclk,
    input [1:0] sw,
    input [3:0] btn,
    output wire [3:0] led,
    output wire led4_b,
    output wire led4_g,
    output wire led4_r,
    output wire led5_b,
    output wire led5_g,
    output wire led5_r
);
```

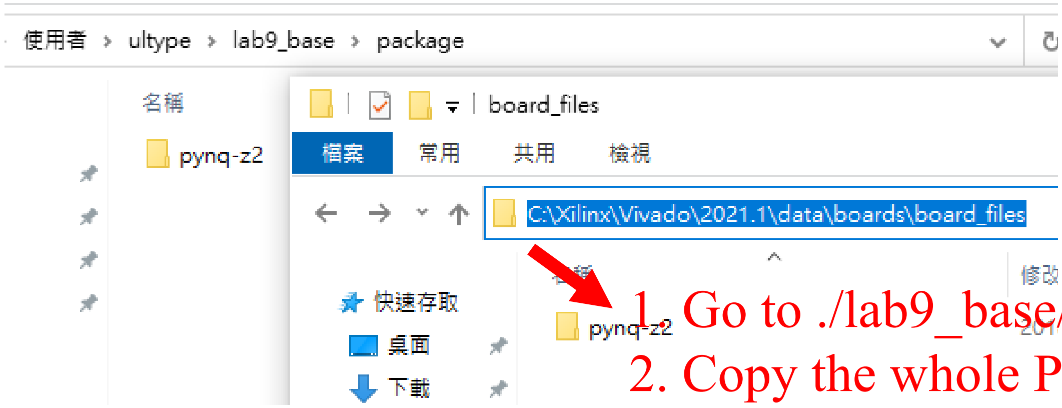
PYNQ-Z2 Board Hardware Introduction

Set Jumper to FPGA boot mode



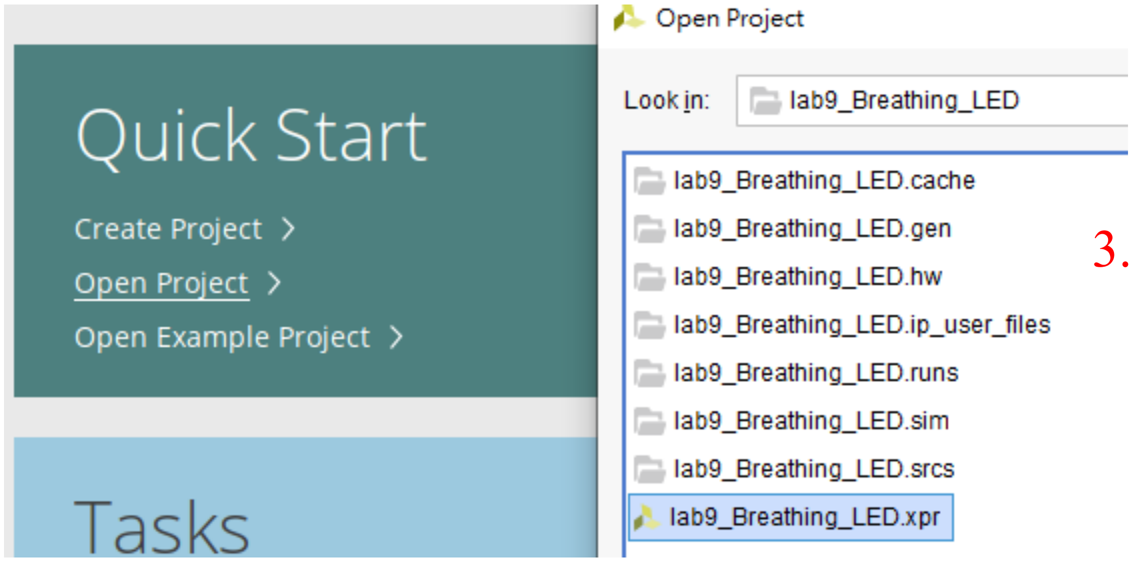


Install PYNQ-Z2 board file package



C:\Xilinx\Vivado\2021.1\data\boards\board_files

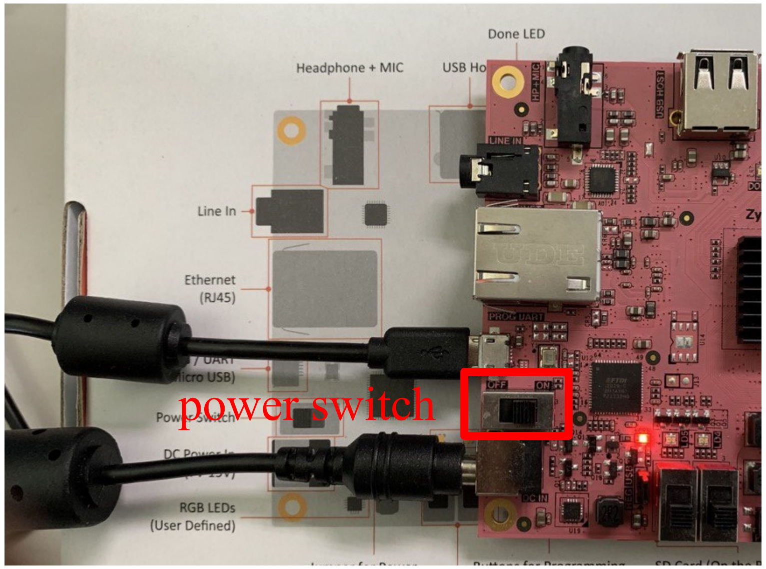
1. Go to ./lab9_base/package/
2. Copy the whole PYNQ-Z2 folder
   to Vivado board files:
   PATH%/Xilinx/Vivado/2021.2/data/boards/board_files



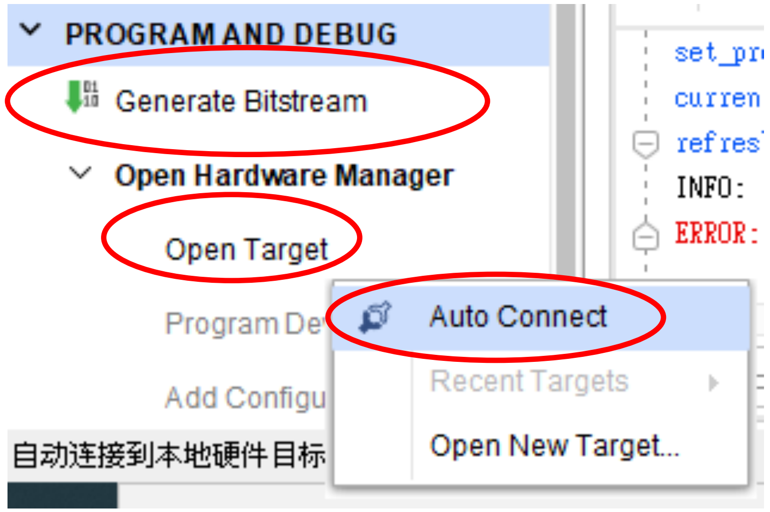
3. Open lab9 project template

Board Connection Checking

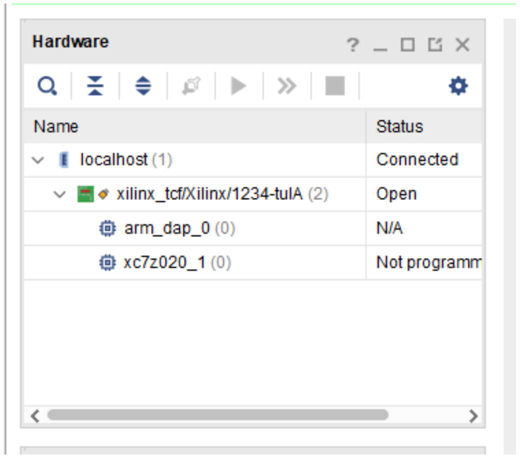
1. plug micro USB and power port
2. Turn on power switch
3. Connect to device
4. Check hardware panel



.Connect to device(Left down manual)

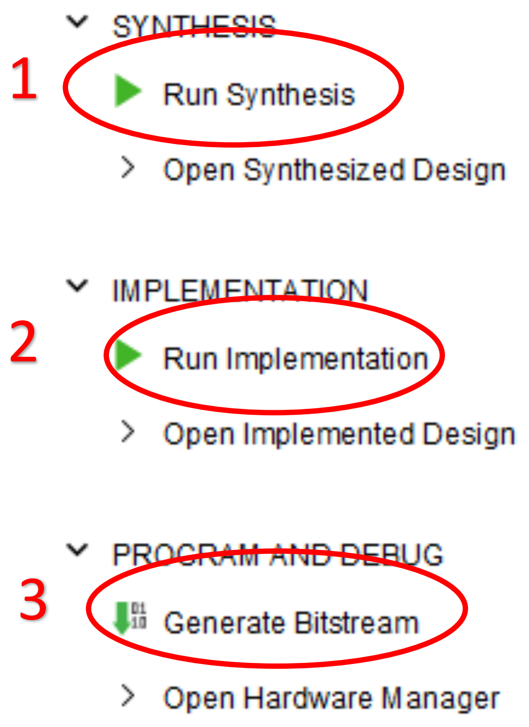


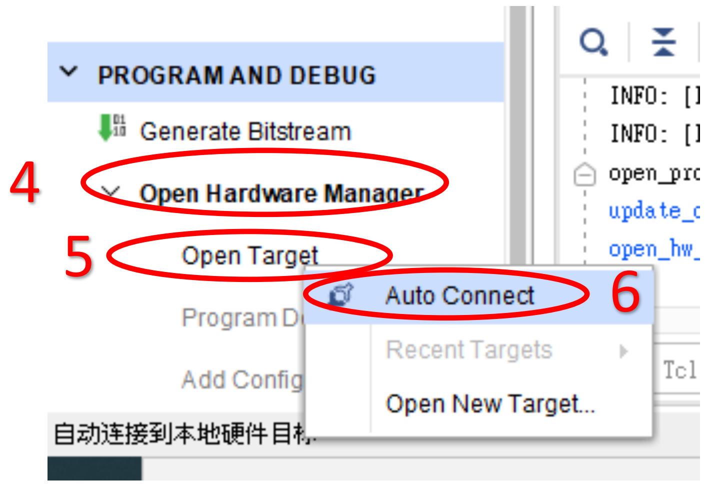
Hardware panel

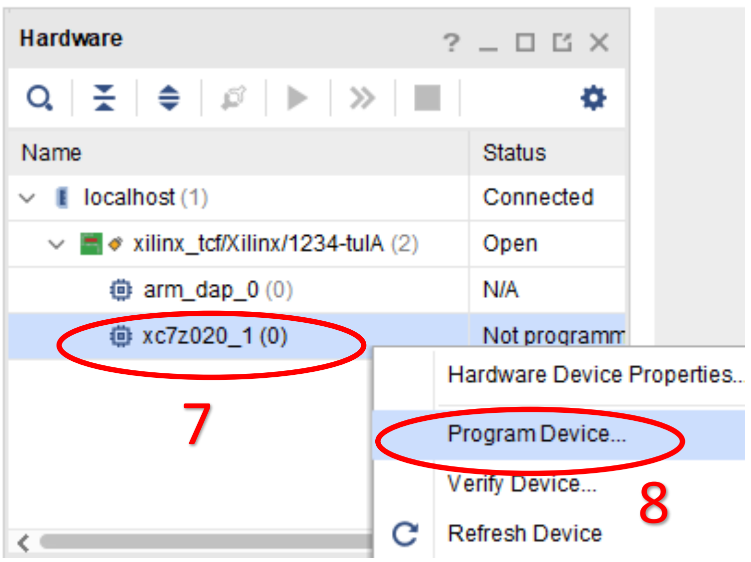

Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Xilinx/1234-tulA (2)	Open
arm_dap_0 (0)	N/A
xc7z020_1 (0)	Not programm

FPGA 合成&燒錄


SYNTHESIS
1 Run Synthesis
Open Synthesized Design
IMPLEMENTATION
2 Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
3 Generate Bitstream
Open Hardware Manager

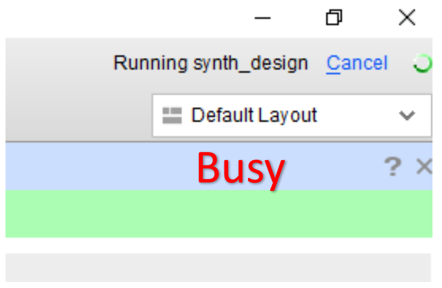

PROGRAM AND DEBUG
Generate Bitstream
4 Open Hardware Manager
5 Open Target
Program D
Add Config
6 Auto Connect
Recent Targets
Open New Target...
自动连接到本地硬件目标
INFO: [I
INFO: [I
open_pro
update_c
open_hw_
Tcl

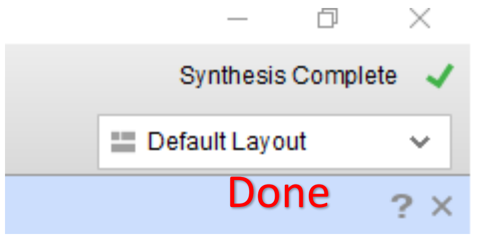

Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Xilinx/1234-tulA (2)	Open
arm_dap_0 (0)	N/A
xc7z020_1 (0)	Not programm

7
Hardware Device Properties...
Program Device...
Verify Device...
Refresh Device
8

Note up-right side


Running synth_design Cancel
Default Layout
Busy


Synthesis Complete ✓
Default Layout
Done