FPGA Homework1 第八組

組員一：F44071128 李其祐 組員二：C14074021 張柏彥

組員三: F14051041 陳祺

Problem1- RGB LEB

一、設計規格：

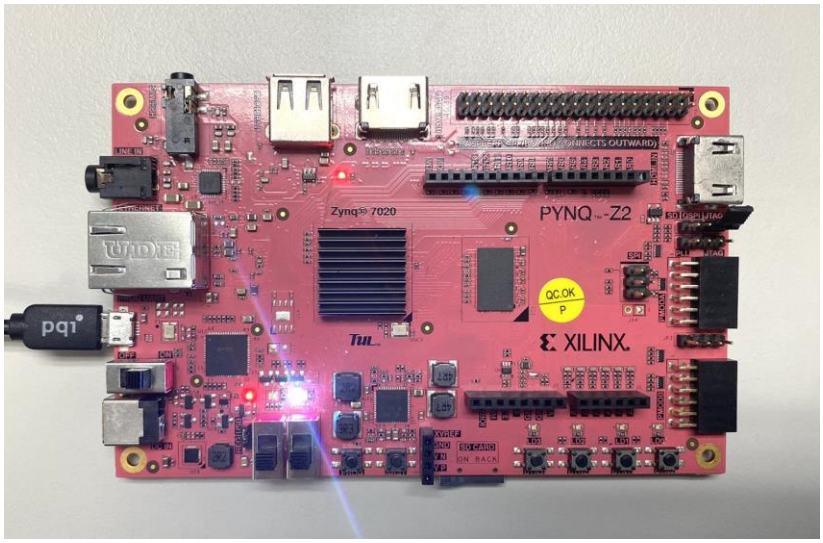
Module : LED			
Signal Name	Direction	Width(bit)	Description
rst	input	1	非同步系統重置訊號，當此訊號為 1 時表示系統重置。
clk	Input	1	系統時脈訊號。
sw	Input	2	開關控制的輸入信號， sw[0]為第一個開關信號 sw[1]為第二個開關信號
rgb[2:0]	Output reg	3	控制 RGB_LED 的輸出信號 rgb[0]為藍色 led 控制信號 rgb[1]為綠色 led 控制信號 rgb[2]為紅色 led 控制信號

二、RTL Code 設計說明：

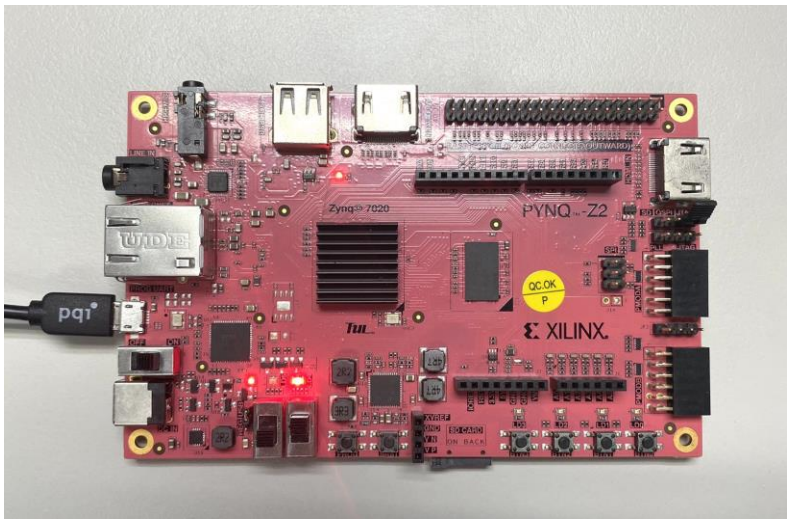
此電路為 combinational circuit，當 sw 為 2’b00 時，rgb 輸出 3’b111，RGB_LED 會亮白色 ； 當 sw 為 2’b01 時，rgb 輸出 3’b100，RGB_LED 會亮紅色; 當 sw 為 2’b10 時，rgb 輸出 3’b010，RGB_LED 會亮綠色 ； 當 sw 為 2’b11 時，rgb 輸出 3’b110，RGB_LED 會亮黃色(由綠色+紅色組成)。

三、FPGA 驗證結果

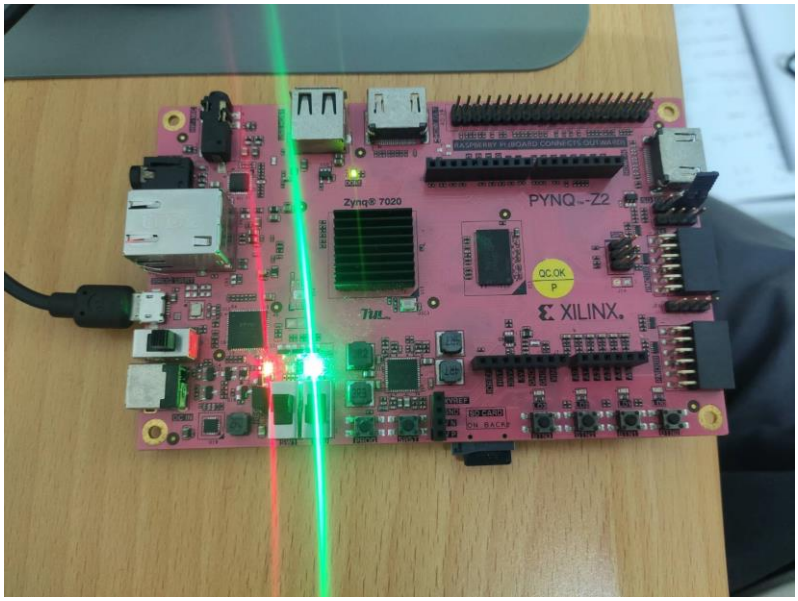
sw == 2’b00，RGB_LED 亮白色燈



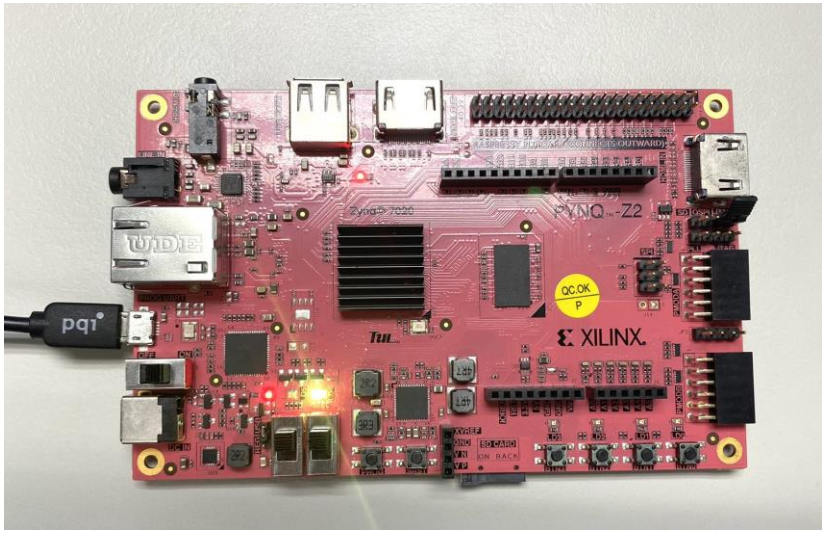
sw == 2’b01，RGB_LED 亮紅色燈



sw == 2’b10，RGB_LED 亮綠色燈

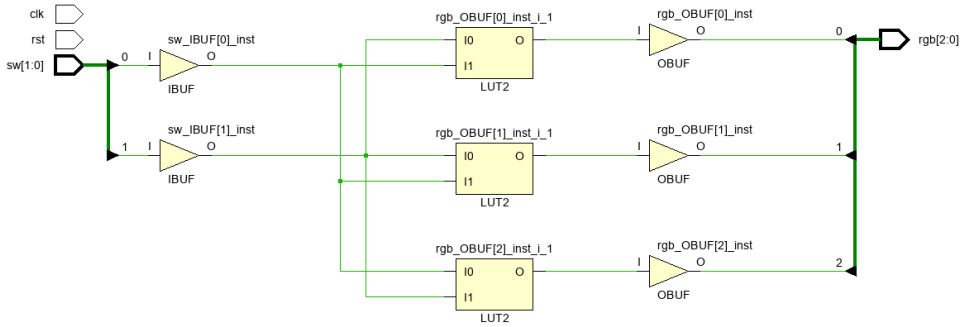


sw == 2’b11，RGB_LED 亮黃色燈

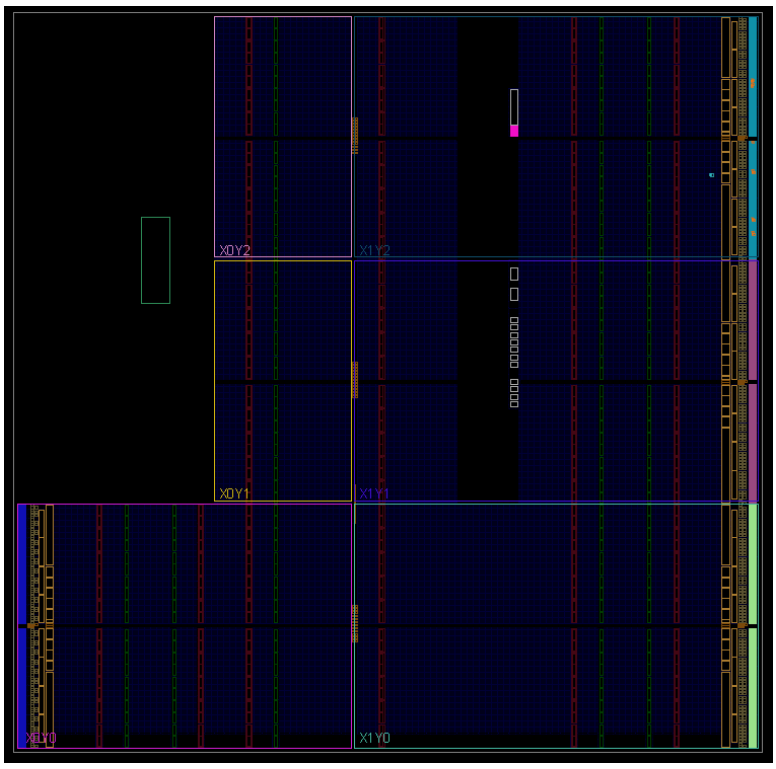


四、Synthesize / Implememtation 結果

Netlist



Layout



Power

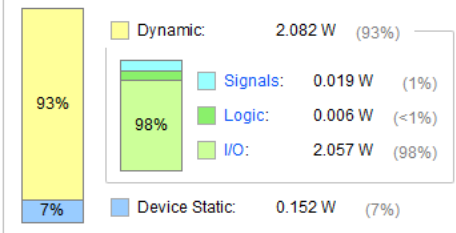
Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**2.235 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**50.8°C**
Thermal Margin:	34.2°C (2.8 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power



Problem2-Traffic Light

一、設計規格：

Module : traffic_light			
Signal Name	Direction	Width(bit)	Description
rst	input	1	非同步系統重置訊號，當此訊號為 1 時表示系統重置。
clk	Input	1	系統時脈訊號。
sw	Input	2	開關控制的輸入信號， sw[0]為第一個開關信號 sw[1]為第二個開關信號
BTN	Input	3	按鈕控制的輸入信號， BTN[0]為第一個按鈕信號，用於 RESET 秒數 BTN[1]為第二個按鈕信號，用於增加秒數 BTN[2]為第三個按鈕信號，用於減少秒數
led4	Output reg	3	控制第一個 RGB_LED 的輸出信號 led4[0]為藍色 led 控制信號 led4[1]為綠色 led 控制信號 led4[2]為紅色 led 控制信號
led5	Output reg	3	控制第二個 RGB_LED 的輸出信號 led5[0]為藍色 led 控制信號 led5[1]為綠色 led 控制信號 led5[2]為紅色 led 控制信號
led	Output	4	控制 LED 的輸出信號，代表 led[0]為第 1 個 led 控制信號，代表 LSB led[1]為第 2 個 led 控制信號 led[2]為第 3 個 led 控制信號 led[2]為第 4 個 led 控制信號，代表 MSB

我們自定義的訊號:

Signal Name	Type	Width(bit)	Description
state	reg	3	儲存 FSM 目前狀態的暫存器
n_state	reg	3	儲存 FSM 下一個狀態的暫存器
red5	reg	4	儲存目前一紅燈一綠燈時長的暫存器， default 值為 5
red1	reg	4	儲存目前同時紅燈時長的暫存器， default 值為 1
yellow	reg	4	儲存目前一紅燈一黃燈時長的暫存器， default 值為 1
max	reg	4	儲存目前狀態下的最大時長

led_	reg	4	儲存目前狀態下經過的時長
sw_flag	reg	1	目的是判斷我們是否有將 sw 從其他非零值切回 0，才能使得 sw 切回 0 時，紅綠燈能從 state=GR 開始運作

Signal Name	Type	Width(bit)	Description
flag	wire	1	當 flag=1 時，代表狀態要在下一個 cycle 改變，使 led_從 0 開始計數

Parameter	Value	Description
GR	0	代表 T1 亮綠燈；T2 亮紅燈的的狀態
YR	1	代表 T1 亮黃燈；T2 亮紅燈的的狀態
RR1	2	代表第 1 次 T1、T2 同時亮紅燈的狀態
RG	3	代表 T1 亮紅燈；T2 亮綠燈的的狀態
RY	4	代表 T1 亮紅燈；T2 亮黃燈的的狀態
RR2	5	代表第 2 次 T1、T2 同時亮紅燈的狀態

二、RTL Code 設計說明：

1. state transition:

當 sw 為 2'b00 時，紅綠燈開始正常運作，我們預設 state 為 GR 並依照兩個號誌燈狀態的變化，將 n_state 給定下一個狀態。在 state 為 GR 時，若 led_ == red5 且 sw_flag == 0 時，代表我們將 sw 從其他非零值切回 0 且 led_已經數到一紅燈一綠燈狀態下的最大時長，因此 n_state 會變成 YR；在 state 為 YR 時，若 led_ == yellow 時，led_已經數到一黃燈一紅燈狀態下的最大時長，因此 n_state 會變成 RR1；在 state 為 RR1 時，若 led_ == red1 時，led_已經數到同時紅燈狀態下的最大時長，因此 n_state 會變成 RG；在 state 為 RG 時，若 led_ == red5 時，led_已經數到一紅燈一綠燈狀態下的最大時長，因此 n_state 會變成 RR2；在 state 為 RY 時，若 led_ == yellow 時，led_已經數到一黃燈一紅燈狀態下的最大時長，因此 n_state 會變成 RR2；在 state 為 RR2 時，若 led_ == red1 時，led_已經數到同時紅燈狀態下的最大時長，因此 n_state 會變成 GR。

2. red5, yellow, red1 控制電路:

當 rst=1 時分別將 red5 預設為 5、yellow 預設為 1、red1 預設為 1，且當 sw 分別為 2'b01、2'b10、2'b11 時，可藉由 BTN 調整 red5、yellow、red1 的時長。而 BTN 又分別藉由 BTN[0]、BTN[1]、BTN[2]執行 RESET、加一秒、減一秒的功能。

3.

我們利用 flag 訊號控制 led_在 sw 為 2’b00（即兩個號誌正常運作）時的變化，若 (state == GR or RG 且 led_==red5) 或 (state == YR or RY 且 led_==yellow) 或 (state == RR1 or RR2 且 led_==red1) 或 sw_flag==1（即代表我們將 sw 從其他非零值切回 0）時，flag 變為 1。

4. led_控制電路:
當 rst=1 時，將 led_初始化成 0。當 sw==2’b00，且當 flag 為 1 時，代表 led_為各狀態時的最大時長，因此在下個 cycle 時需將 led_變為 0 重新計數，若 flag 為 0 則 led_ = led_ + 1。當 sw 為 2’b01、2’b10、2’b11，led_則顯示各狀態下的最大時長，即分別為 red5、yellow、red1。

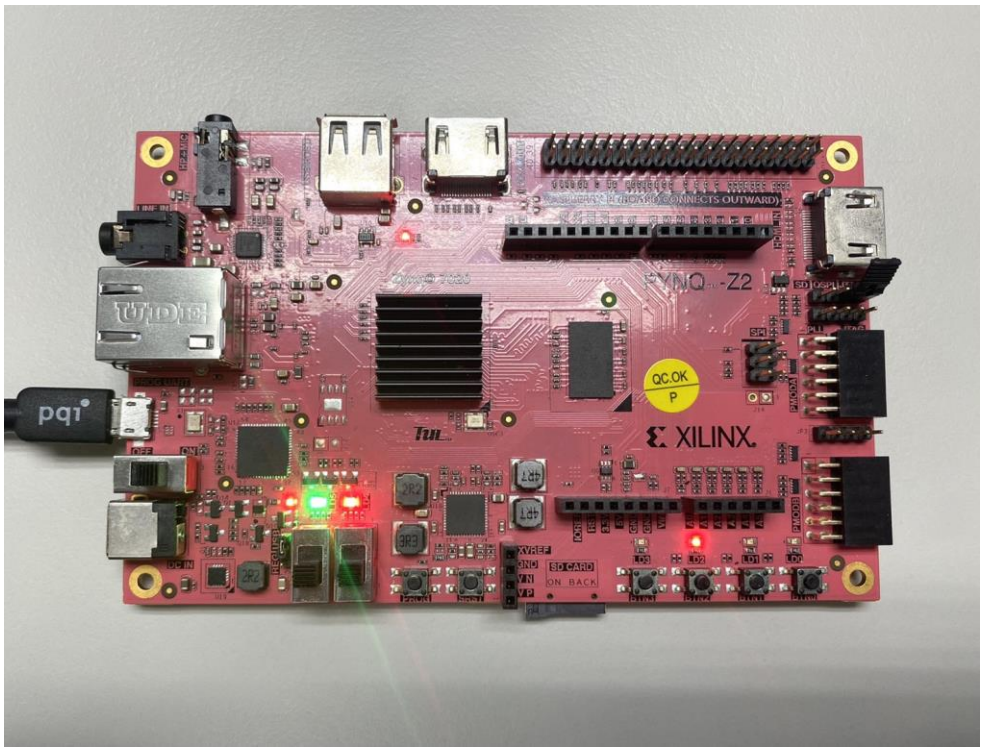
5. sw_flag 控制電路:
當 rst=1 時，將 sw_flag 初始化成 0，當 sw 從非零值變成 0 時，sw_flag=1 維持一個 cycle 後變為 0。

6. max 控制電路:
我們設計此電路的目的是因為紅綠燈顯示必須是倒數的，但從最大值往下扣會造成控制訊號複雜，所以我們一樣採用正數的方式，但透過 max 值來相減就可以得到倒數的值。由於我們預設當 sw 從非零值變回 0 時會回到 state GR，因此我們將 max 預設為 red5，當 state == GR or RG，max 為 red5 ; state == YR or RY，max 為 yellow ; state == RR1 or RR2，max 為 red1。接著我們還會再搭配下方的 led 訊號來選擇我們的輸出訊號，若 sw == 0，就要輸出 max-led_的值，反之若為其他 sw，就輸出 led_，也就是當下每個燈號的時長。

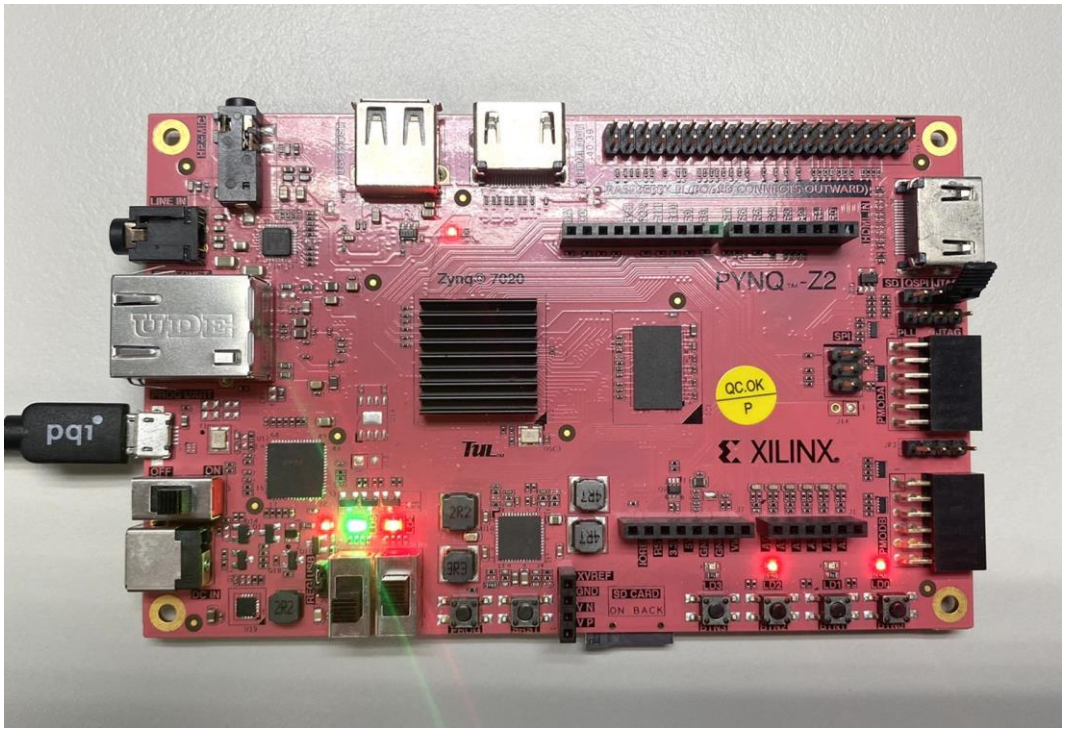
7. led4, led5 控制電路:
led4、led5 分別代表兩個不同 RGB_LED（即兩個不同的的交通號誌 T1, T2），在不同的狀態下分別給予不同的訊號。當 state 為 GR 時，led4 為 3’b010，led5 為 3’b100，即 led4 為綠燈、led5 為紅燈 ； 當 state 為 YR 時，led4 為 3’b110，led5 為 3’b100，即 led4 為黃燈、led5 為紅燈 ； 當 state 為 RR1 時，led4 為 3’b100，led5 為 3’b100，即 led4、led5 皆紅燈= red1 ; 當 state 為 RG 時，led4 為 3’b100，led5 為 3’b010，即 led4 為紅燈、led5 為綠燈 ； 當 state 為 RY 時，led4 為 3’b100，led5 為 3’b110，即 led4 為紅燈、led5 為黃燈 ； 當 state 為 RR2 時，led4 為 3’b100，led5 為 3’b100，即 led4、led5 皆為紅燈。

三、FPGA 驗證結果

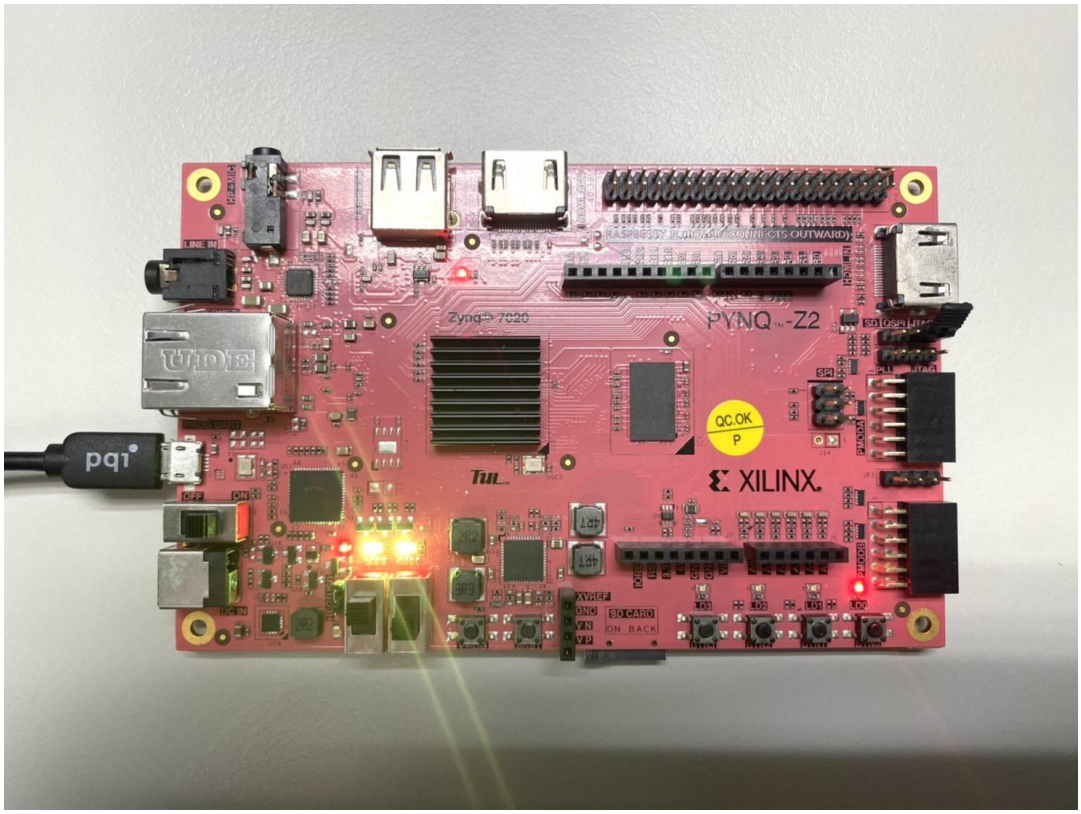
1. sw == 2’b00，兩個號誌正常運作，default 狀態為 GR



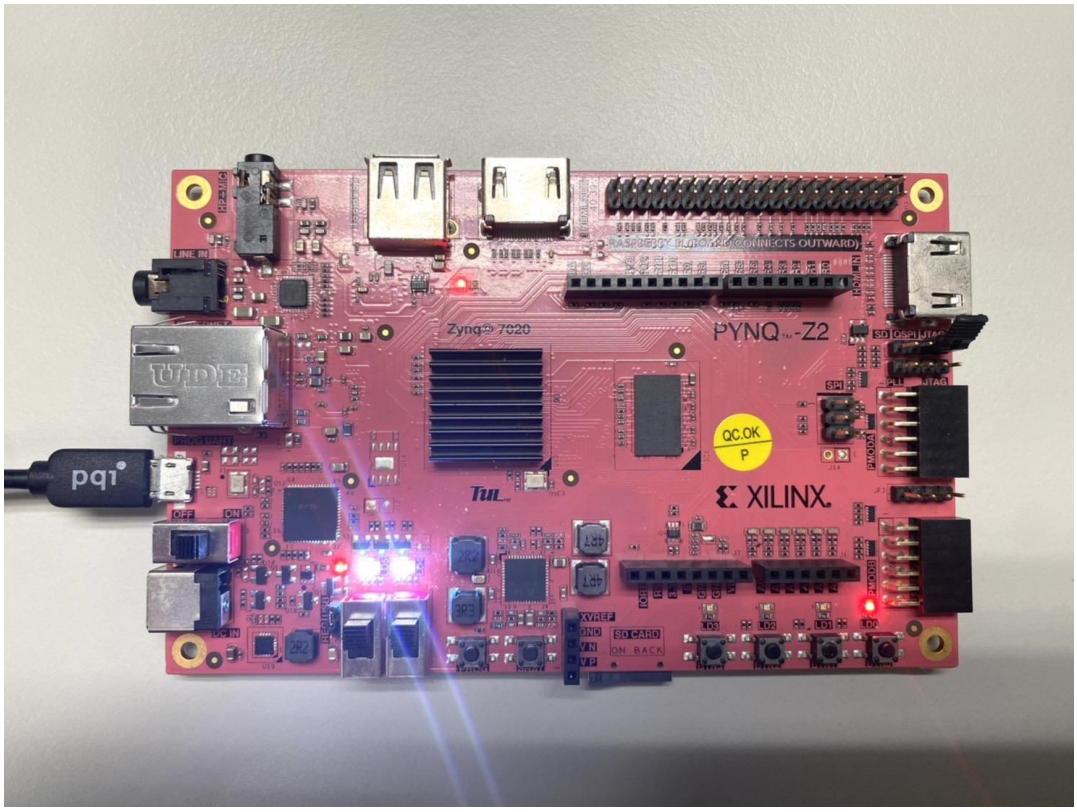
2. sw == 2’b01，RGB_LED 亮一紅一綠，並顯示一紅一綠燈狀態下的目前時長，default 秒數為 5 秒



3. sw == 2’b10，RGB_LED 皆亮黃燈，並顯示黃燈狀態下的時長，default 秒數為 1 秒

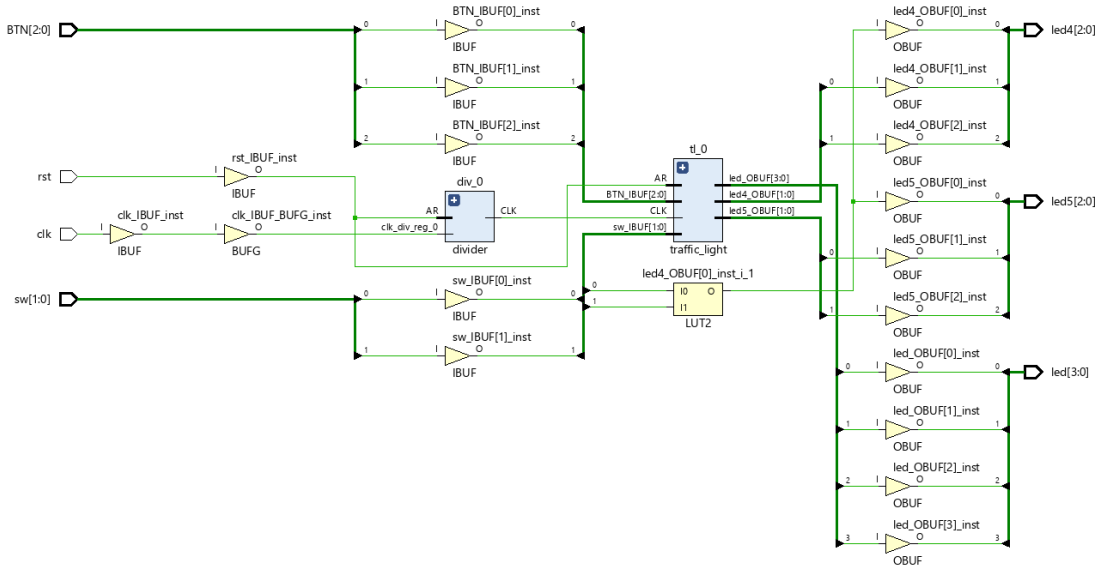


4. sw == 2’b11，RGB_LED 皆亮白燈，並顯示同時紅燈狀態下的時長，default 秒數為 1 秒

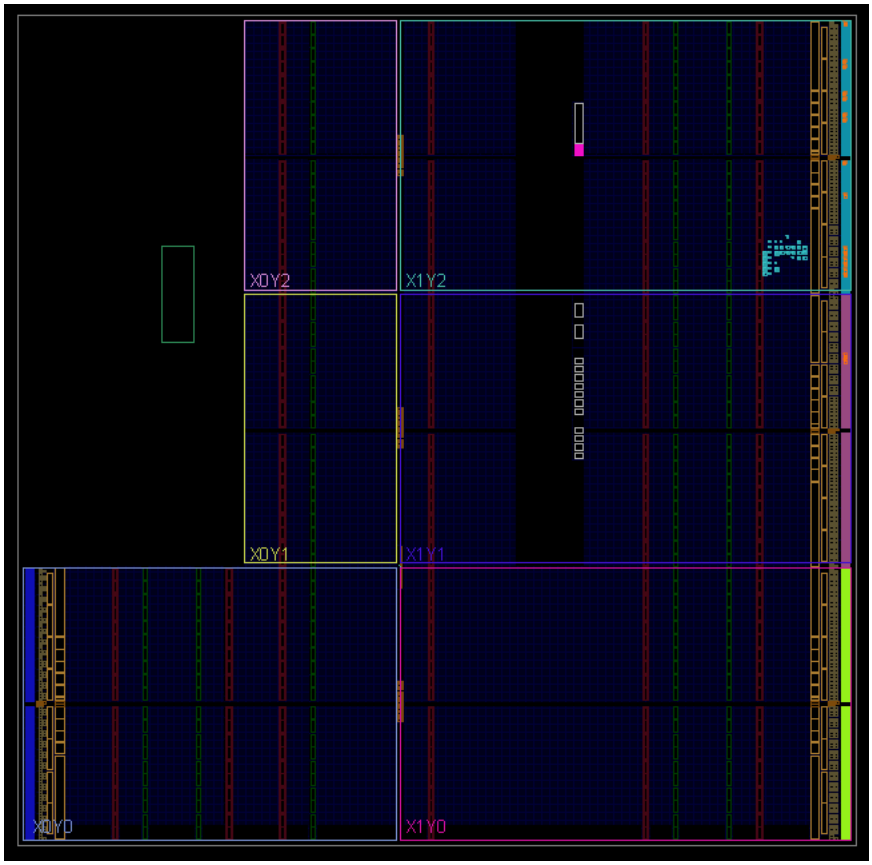


四、Synthesize / Implememtation 結果

Netlist



Layout



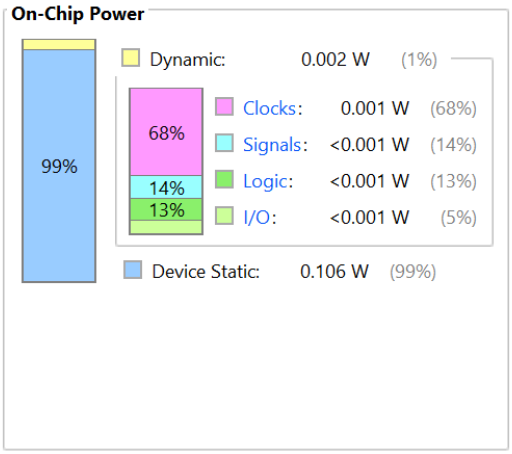
Power

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.107 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.2°C**
Thermal Margin:	58.8°C (4.9 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



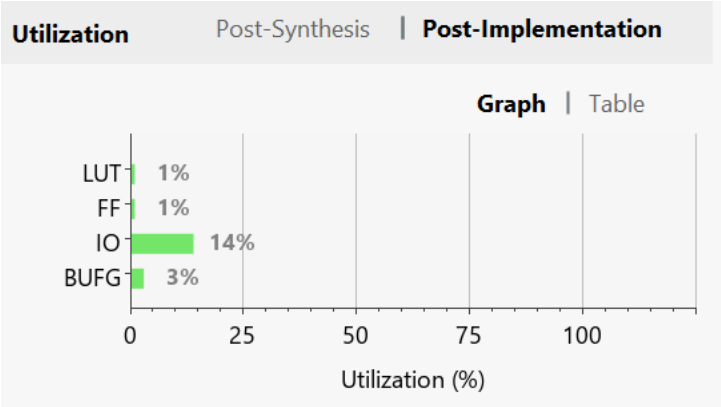
Timing

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.601 ns	Worst Hold Slack (WHS):	0.112 ns	Worst Pulse Width Slack (WPWS):	3.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	46	Total Number of Endpoints:	46	Total Number of Endpoints:	48

All user specified timing constraints are met.

Utilization



Problem

1. 為什麼要加入”blinky.sdc”?

產生 sys_clk 讓 vivado 知道時序上的要求

因為電路內有除頻後的 clk 驅動 DFF，需要設定 constraint 讓 vivado 能夠追蹤該 clk 經過的路徑並計算延遲。

2. Vivado 的開發流程中，Synthesis 和 Implementation 的結果差異在哪?

Synthesis 僅是將 RTL 轉譯成 gate level 層級的電路。

Implementation 則是依照 constraint 將轉譯出來的電路實際擺放到 FPGA 上面，並完成線路的連接。