電子系統層級設計與驗證

Homework 1

Implement the RGB to YUV circuit
with three multipliers, three adders,
and initiation interval = 3
by using structural (FSM+ Datapath) Verilog code

學號：B103090046

姓名：宋易柔

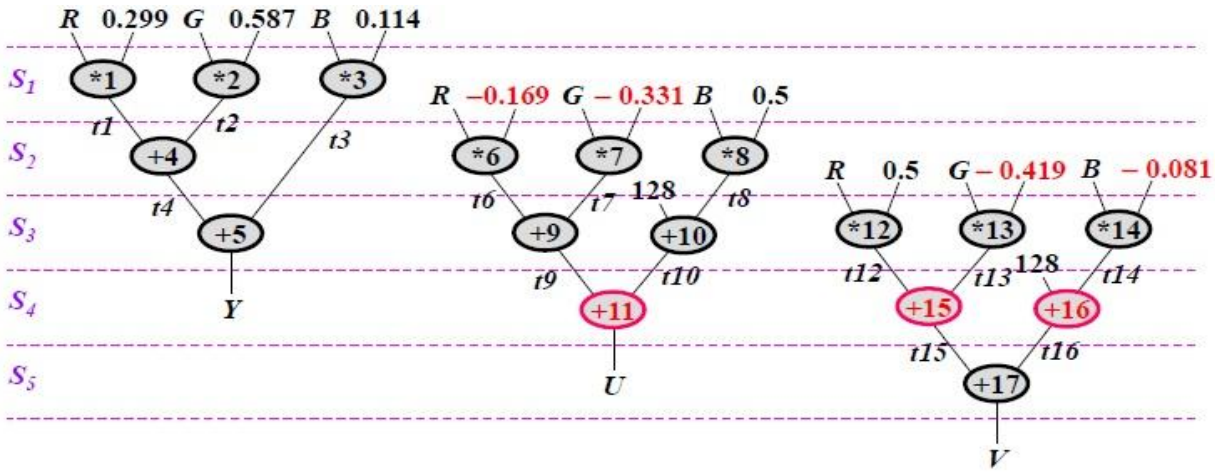
日期：113/04/08

(一) 實驗主題

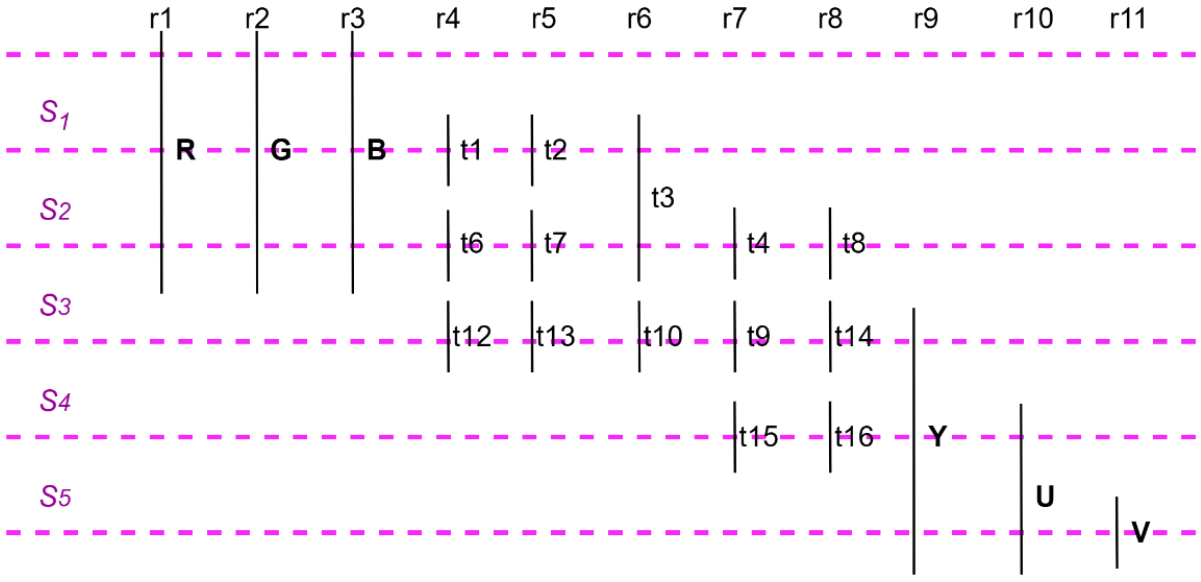
在設計三個乘法器和三個加法器的硬體時，首先對所有運算結果和對應暫存器進行排程，接著設計出相應的 Datapath，最後調整 Controller 來控制各個多工器、暫存器及 ROM address 的輸入輸出訊號。

(二) 實驗過程

(1) 排程

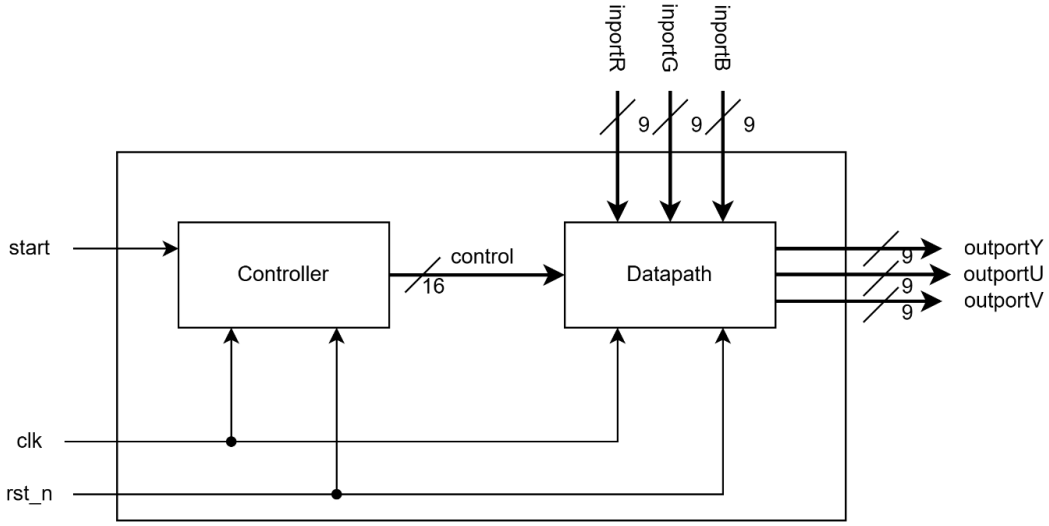


排程表

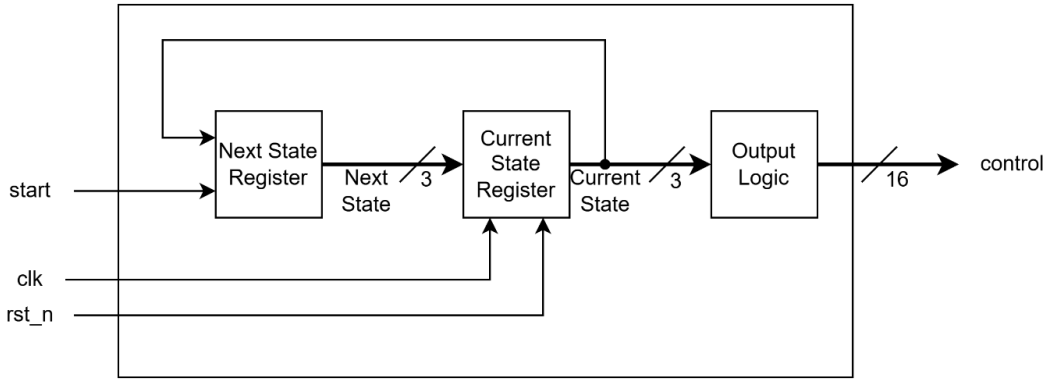


(2) 架構圖

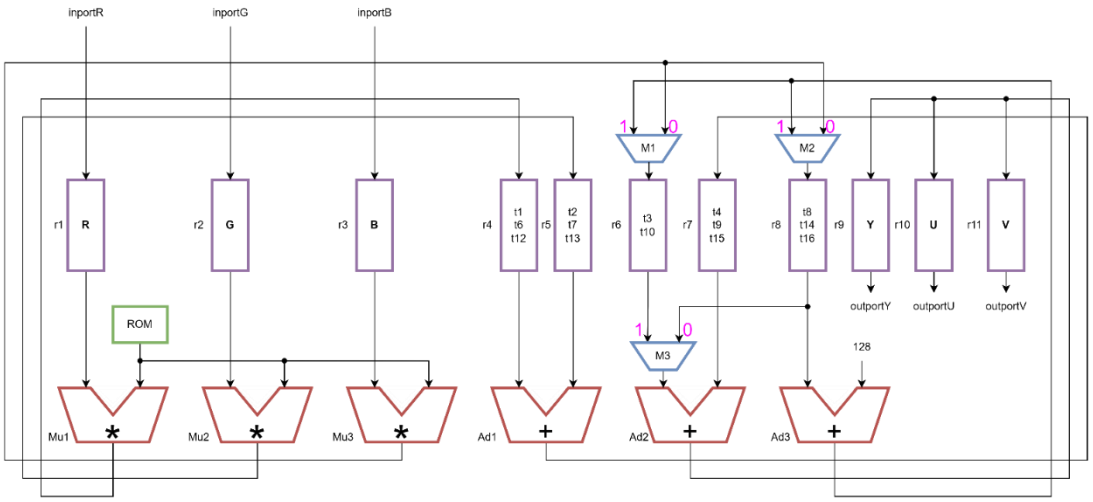
RGB2YUV(整體架構)：



Controller：



Datapath：



ROM address：

Address(binary)	Value (decimal)	Value (binary)
00	0_0_0	000000000_000000000_000000000
01	0.299_0.587_0.114	001001100_010010110_000011101
10	-0.169_-0.331_0.5	111010101_110101100_010000000
11	0.5_-0.419_-0.081	010000000_110010101_111101100

一個Address放同一個狀態會需要使用到所有的值(Value)。並將18~26位元指給Mu1乘法器使用、9~17位元指給Mu2乘法器使用、0~8位元指給Mu3乘法器使用

如：當S1時，使用Address=01的值；

當S2時，使用Address=10的值；

當S3時，使用Address=11的值。

(3) Controller的FSM：

暫存器與ROM：

Present State	Input	Next State	Control Signals											
			r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	ROM
S0	Start=0	S0	1	1	1	0	0	0	0	0	0	0	0	-
	Start=1	S1												
S1	-	S2	0	0	0	1	1	1	0	0	0	0	0	01
S2	-	S3	0	0	0	1	1	0	1	1	0	0	0	10
S3	-	S4	0	0	0	1	1	1	1	1	1	0	0	11
S4	-	S5	0	0	0	0	0	0	1	1	0	1	0	-
S5	-	S0	0	0	0	0	0	0	0	0	0	0	1	-

設計共有五個狀態。

暫存器Control Signals為1時，代表該狀態需要使用到該暫存器，反之，則無。

而ROM僅在S1、S2、S3時須參與運算。

多工器：

Present State	Input	Next State	Control Signals		
			M1	M2	M3
S0	Start=0	S0	-	-	-
	Start=1	S1			
S1	-	S2	0	-	-
S2	-	S3	-	0	-
S3	-	S4	1	0	1
S4	-	S5	-	1	1
S5	-	S0	-	-	0

本設計之三個多工器皆為二對一。

M1決定r6的輸入為Mu3(M1=0)或是Ad3的輸出(M1=1)；

M2決定r8的輸入為Mu3(M2=0)或是Ad3的輸出(M2=1)；

M3決定Ad2的輸入為r8(M3=0)或是r6的輸出(M3=1)。

p.s.
1. Control Signals = 16'br1_r2_r3_r4_r5_r6_r7_r8_r9_r10_r11_ROM_M1_M2_M3
2. 表格中「-」(Don't care)皆設為0。

(三)實驗結果

(1) 圖片

1. 原圖

2.Y

3. U

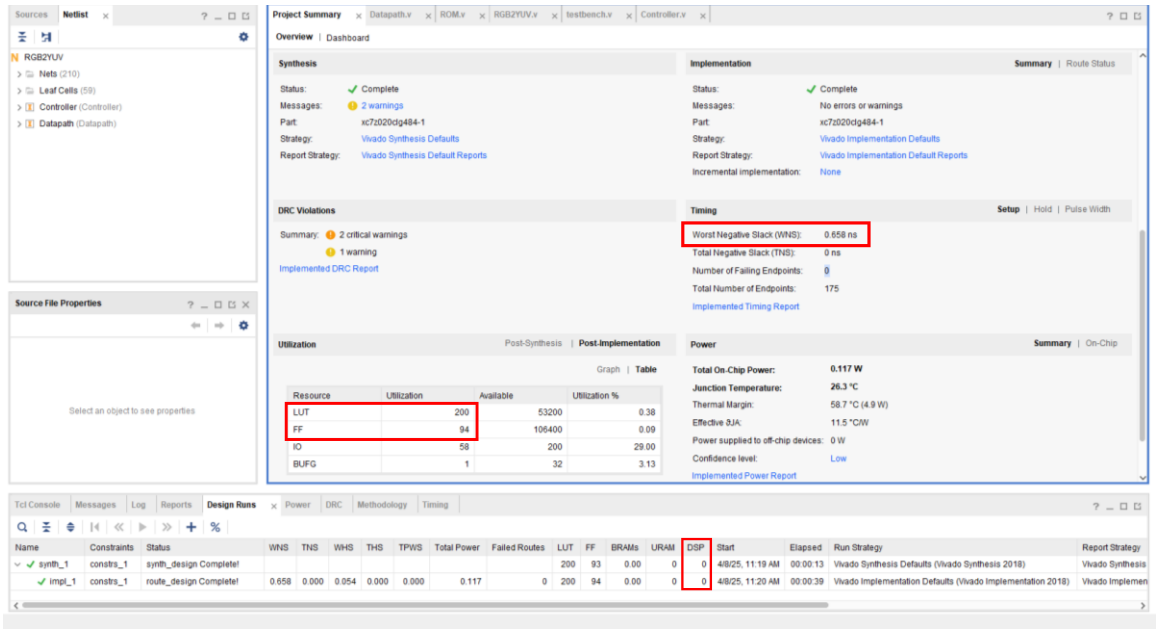
4.V

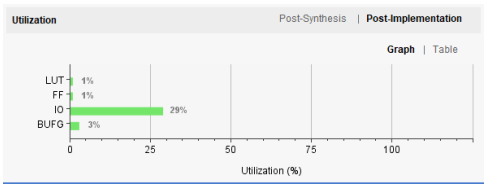
(2) Vivado 合成結果

RGB2YUV	DSP	LUT	FF	Clock Freq.	Cycle
Version 1	0	200	93	151.9MHz	3145960
Version 2	0	235	105	45.8 MHz	1573096
Version 3	0	277	132	50.5MHz	786672

Version 1

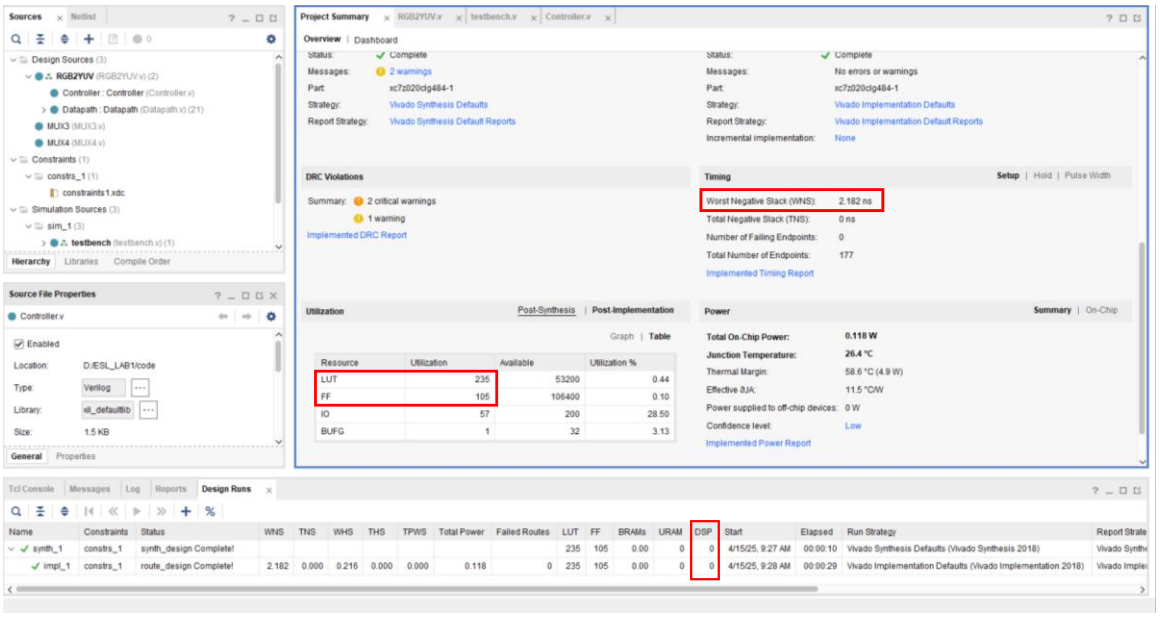
```
$finish called at time : 31459600 ns : File "D:/ESL_LAB1/testbench.v" Line 63
```

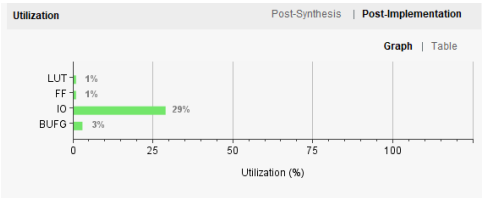




Version 2

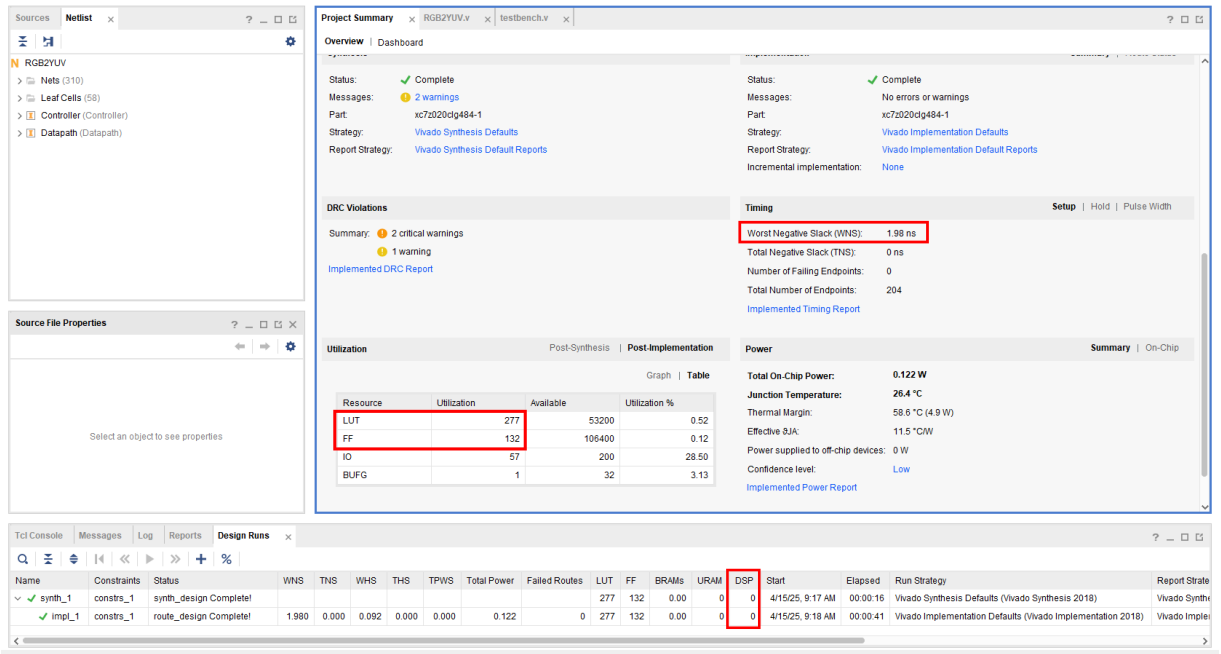
```
$finish called at time : 15730960 ns : File "C:/Users/cheng/OneDrive/文件/version2/testbench.v" Line 63
```

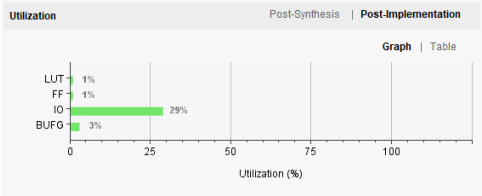




Version 3

```
$finish called at time : 7866720 ns : File "C:/Users/cheng/OneDrive/文件/version3_v2/testbench.v" Line 64
```





(四)實驗心得

這次作業用了三種方式來實現 RGB 轉 YUV，從中我發現如果可以使用的資源愈多，就會愈有可能加速完成算法，但是同時也必須犧牲面積、功耗等等代價。還有排程對於電路設計亦為相當重要，了解每個訊號需要存在多久，就可以讓出不需要的位置給別的訊號使用。指令管線化的部分苦思了一陣子要如何實作，了解其中訊號的意義，就發現其實只要讓控制訊號推遲三個 cycle，再跟當下的控制訊號做 bit-wise or 就可以完成管線，確實讓第二種版本的電路使用一樣多的資源(三個乘法器與三個加法器)，但卻可以更快速的完成得到一樣的 YUV 圖，而且平均三個 clock 就可以完成一個 pixel。

另外，我也好奇 Version 1 的 V 圖為何與我做的 Version 2 和 3 有些微差異，反覆從頭仔細檢查 Controller 與 Datapath 是否正確，覺得都沒問題，公式也正確，不過這個星期用 SystemC 做出來的圖與我的 Version 2 和 3 一樣，就比較放心無誤了。