國立中山大學電機工程學系

系統晶片實作專題

圖像卷積電路設計

Image Convolutional Circuit Design

專題生:

電機工程學系 劉浩崴 B093011055

電機工程學系 林祁穎 B103012040

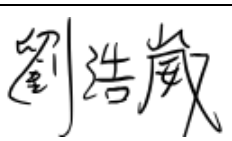
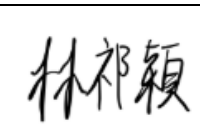
電機工程學系 鄞振祐 B103012042

指導教授: 謝東佑 教授

指導教授簽名:_____

中華民國 一百一十三 年 六 月 十一 日

專題分工說明與貢獻度百分比

專題生姓名	專題分工說明	貢獻度百分比	專題生簽名
劉浩崴	1. 專題電路設計 2. 高可靠電路設計 3. 專題報告撰寫	33.33%	
林祁穎	1. 專題電路設計 2. 演算方式設計 3. 專題報告撰寫	33.33%	
鄞振祐	1. 專題電路設計 2. 硬體資訊安全設計 3. 專題報告撰寫	33.33%	

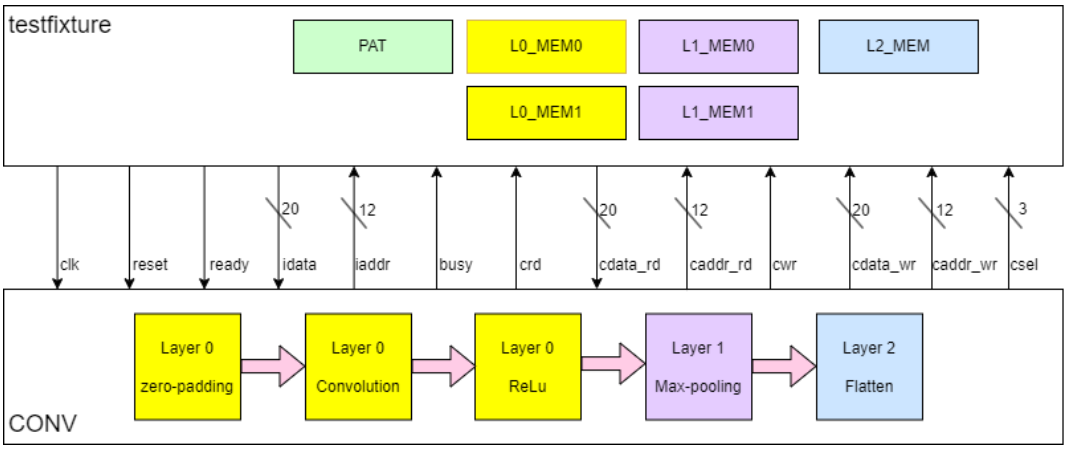
摘要

本專題需設計出一圖像卷積電路(Image Convolutional Circuit，CONV)，此電路須完成 3 層的運算流程。依序為 Convolution→Max-pooling→Flatten。

Layer0 為 Convolution，在此階段會先將從 PAT 讀取到的灰階資料進行 Zero-padding，而後會將此筆資料對 2 個不同的核(尺寸為 3x3)進行卷積運算以及 ReLU 運算，最後將計算完的兩筆資料分別送回 testfixture 中的 L0_MEM0 及 L0_MEM1 儲存。

Layer1 為 Max-pooling，此階段會讀取 L0_MEM0 以及 L0_MEM1 中的資料，以大小為 2x2 的視窗且步幅為 2 的方式取出範圍內的最大值，將資料轉為大小為 32x32 的圖像送回 testfixture 中的 L1_MEM0 及 L1_MEM1 儲存。

Layer2 為 Flatten，此階段會讀取 L1_MEM0 及 L1_MEM1 中的資料，將兩筆資料以交錯的方式依序寫入 testfixture 中的 L2_MEM。



電路腳位說明

輸入訊號	Width	說明
clk	1	系統時脈訊號。本系統為同步於時脈正緣之同步設計。
reset	1	高位準非同步之系統重置信號。
ready	1	灰階圖像準備完成指示訊號。當訊號為 High 時，表示灰階圖像準備完成，此時 CONV 才可以開始向 testfixture 發送輸入灰階圖像資料索取位址。
idata	20	輸入灰階圖像像素資料訊號，由 4 bits 整數(MSB)加上 16 bits 小數(LSB)組成，為有號數。testfixture 將 iaddr 所指示的位址之像素資料用此訊號送給 CONV。
cdata_rd	20	CONV 運算結果記憶體讀取訊號，由 4 bits 整數(MSB)加上 16 bits 小數(LSB)組成，為有號數。testfixture 將記憶體資料傳送至 CONV 電路。

輸出訊號	Width	說明
busy	1	系統忙碌指示訊號。當 CONV 接收到 ready 訊號為 High，且 CONV 準備開始動作時，需將此訊號設為 High，表示準備開始進行輸入灰階圖像資料索取；待所有運算處理完成且輸出結果寫回 testfixture 後，需再將訊號設為 Low 表示動作結束。
iaddr	12	輸入灰階圖像位址訊號。指示欲索取哪個灰階圖像像素(pixel)資料的位址。
crd	1	CONV 運算輸出記憶體讀取致能訊號。當時脈正緣觸發時，若此訊號為 High，表示要進行讀取動作。testfixture 會將 caddr_rd 位址指示之資料讀取到 cdata_rd 上。
caddr_rd	12	CONV 運算結果記憶體讀取位址。CONV 電路各層的運算結果利用此訊號指示將要讀取 testfixture 中所內建各層輸出結果之記憶體的哪個位址。
cwr	1	CONV 運算輸出記憶體寫入致能訊號。當時脈正緣觸發時，若此訊號為 High，表示要進行寫入動作。testfixture 會將 cdata_wr 內容寫到 caddr_wr 所指示之位址。
cdata_wr	20	CONV 運算結果記憶體寫出訊號，由 4 bits 整數(MSB)加上 16 bit 小數(LSB)組成，為有號數。CONV 電路各層的運算結果利用此訊號輸出至 testfixture。
caddr_wr	12	CONV 運算結果記憶體寫入位址。CONV 電路各層的運算結果利用此訊號指示將要寫入到 testfixture 中所內建各層輸出結果之記憶體的哪個位址。
csel	3	CONV 運算處理結果寫入/讀取記憶體選擇訊號。此訊號指示目前寫入/讀取資料為 CONV 電路中哪一層的運算結果。說明如下: 3'b000:讀取 Pat。 3'b001:寫入/讀取 Layer 0，Kernel 0 執行 Convolutional 的結果。 3'b010:寫入/讀取 Layer 0，Kernel 1 執行 Convolutional 的結果。 3'b011:寫入/讀取 Layer 1，將 Kernel 0 執行 Convolutional 後再進行 Max-pooling 運算的結果。 3'b100:寫入/讀取 Layer 1，將 Kernel 1 執行 Convolutional 後再進行 Max-pooling 運算的結果。 3'b101:表示寫入/讀取 Layer 2，Flatten 層的運算結果。

目錄

圖目錄

第一章　概論

1.1 研究動機與目的

卷積神經網路(Convolutional Neural Network, CNN) 為近年來一大熱門的研究項目，其主要的應用範圍包括圖像辨識及語音辨識等領域。由於在卷積層中需要將多筆資料與重複的權值做乘法運算，使得在這個階段會需要大量的乘法器以及儲存運算結果的暫存器，進而導致電路的功耗以及成本會有明顯的增加，因此如何在降低電路成本的同時，不能影響到運算的效率以及正確性是一個重要的研究目標。

在本專題中，由於只需要將一張圖片對兩種不同的核進行運算，因此在成本上的考量以電路面積為最優先次序，主要會朝減少乘法器以及暫存器的方向進行。本專題會以類似流水線(pipeline)的架構來實現卷積電路，將卷積層和池化層個別進行分割，並且使用計數器來控制資料的運算及傳送，此方法能夠避免將各層的所有步驟操作在同一段時間內，使得乘法器以及暫存器可以重複使用，有效地減少了電路面積。不過，由於會重複使用暫存器的關係，因此在各階段如何調控時脈訊號以避免資料傳輸錯誤也是需要考量的地方。

第二章　演算法說明

2.1 系統功能

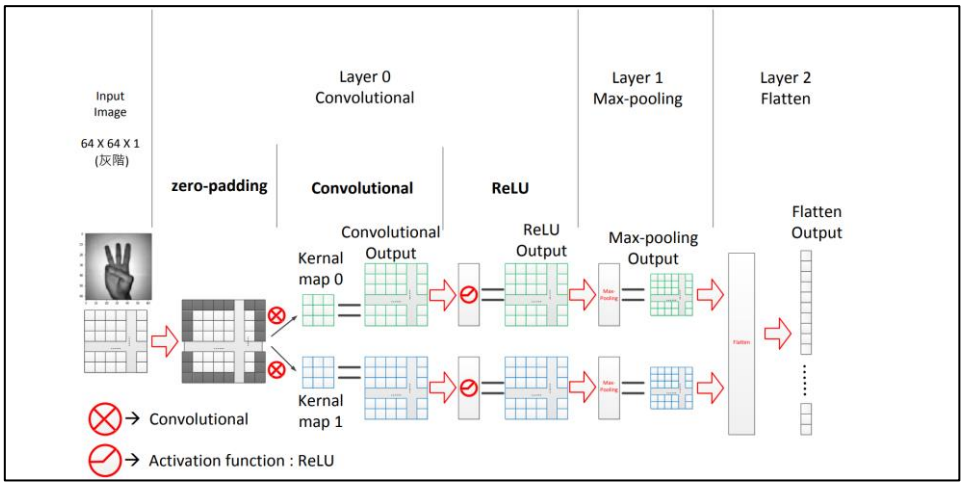


圖 2-1、CONV 電路系統流程圖

此卷積電路系統流程如圖一所述，會先將存放於 Testbench 其尺寸為 64 pixels(寬) x 64 pixels(高)的灰階圖像讀取出來，並依序進行 Convolutional (Layer 0)、Max-pooling (Layer 1)、Flatten (Layer 2)三層運算處理，其詳細功能如下。

2.1.1 Convolutional Layer (Layer 0)

(1) Zero-padding

在進行 Convolution 之前，需要先將原始圖片作 Zero-padding，其目的為避免圖片在 Convolution 後會縮小。方法為在原始圖片的周圍補上 1 pixel 的 0。

(2) Convolution

為將大小為 3x3 的 kernel 疊到經過 Zero-padding 的圖像上，而後將圖像上的每一個 pixel 與對應到 kernel 上的每一個 pixel 相乘，並且將 9 組相乘後的 pixel 值相加，最後再加上題目給定的 Bias，即完成第一次的卷積。做完後需將 kernel 向右移一格，開始新一輪的卷積，當 kernel 最右邊一欄對齊到圖像的最右邊一欄，則下一次的移動方式為將 kernel 移至最左邊並且向下一格，以此類推直到 kernel 對齊圖像右下角的 9 個 pixel，即完成整張圖像的 Convolution，而此電路有 kernel 0、kernel 1 兩組 kernel，因此會分別產生兩份不同的輸出。

(3) ReLU

為了避免 Convolution 後的 pixel 值為負值，需要將結果做調整，調整方式為當 pixel 值大於等於 0 時，值維持不變，當 pixel 值小於 0 時，將 Convolution 結果調整為 0。

2.1.2 Max-pooling Layer (Layer 1)

最大化池層(Max-pooling)是縮小水平及垂直空間的運算，而此電路會將 Layer0 的結果以 2 pixels(寬) x 2 pixels(高)的範圍為單位整合成一元素，並以步幅大小為 2 由左至右、上至下的方式移動，以縮小空間大小。因此經 Max-pooling 後的結果圖之尺寸只會有輸入圖尺寸的 1/2，也就是 32 pixels(寬) x 32 pixels(高)。

2.1.3 Flatten Layer (Layer 2)

平坦化(Flatten)的用意為將兩組二維的圖像轉為一組一維陣列儲存，其方法為將兩組圖像的 pixel 值交錯擺放，各圖像的取值方式為由左到右、由上至下，若為 kernel 0 產生的 pixel 值，擺放位址為偶數，若為 kernel 1 產生的 pixel 值，擺放位址為奇數。

2.2 系統 Finite State Machine (FSM)

圖 2-2、CONV 電路系統 FSM

2.2.1 狀態說明

(1) State_Initial: 電路初始狀態
(2) State_Conv: 電路執行 Convolutional (Layer0)
(3) State_MP: 電路執行 Max-pooling (Layer 1)
(4) State_Flat: 電路執行 Flatten (Layer 2)
(5) State_L0MEM: 電路將 Layer0 執行結果寫入記憶體
(6) State_L1MEM: 電路將 Layer1 執行結果寫入記憶體
(7) State_L2MEM: 電路將 Layer2 執行結果寫入記憶體
(8) State_Complete: 電路完成運算

2.2.2 電路控制訊號說明

(1) 計數器:
pixel_count: 控制 Layer0 pixel 運算和讀寫順序
mp_count: 控制 Layer1 pixel 運算和讀寫順序
(2) kernel、記憶體轉換:
kernel_switch_L0: 控制 Layer0 kernel 0、kernel 1 的轉換
mem_switch_L1: 控制 Layer1 記憶體的轉換
mem_switch_L2: 控制 Layer2 記憶體的轉換
(3) Layer 完成訊號:
L0_finish: Layer0 運算完成
L1_finish: Layer1 運算完成
(4) 圖片 pixel 控制:
x: 圖片 x 軸，用來表示圖片 pixel 的 x 軸位址
y: 圖片 y 軸，用來表示圖片 pixel 的 y 軸位址

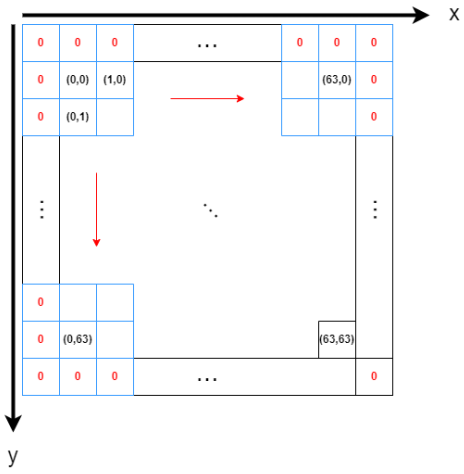


圖 2-3、Layer 0 pixel 座標表示

{y-1,x-1}	{y-1,x}	{y-1,x+1}
{y,x-1}	{y,x}	{y,x+1}
{y+1,x-1}	{y+1,x}	{y+1,x+1}

圖 2-4、Layer 0 kernel 對 x、y 座標

{y,x}	{y,x+1}
{y+1,x}	{y+1,x+1}

圖 2-5、Layer 1 Max-pooling 對 x、y 座標

由圖三得知經 Zero-padding 的圖片其大小為 66 pixels(寬)x 66 pixels(長)，然而整張圖片卷積次數僅需 64x64 次，因此我們可將 kernel 中間那格定為 x、y 座標的基準，如圖四所示，並根據 x、y 座標的值讀取記憶體對應的 pixel 數值，至於 Zero-padding 的部分，我們則訂定 x、y 任一值為 0 或 63 時為邊界條件來讓電路讀取該邊界 Zero-padding 的 0。

而在 Layer1 的 Max-pooling 也可像 Layer0 一樣，將 2 pixels(寬) x 2 pixels(高)的範圍設 x、y 座標，如圖五所示，並根據其位址讀取記憶體對應的數值來做 Max-pooling。

第三章　硬體架構分析

3.1 硬體架構分析流程說明

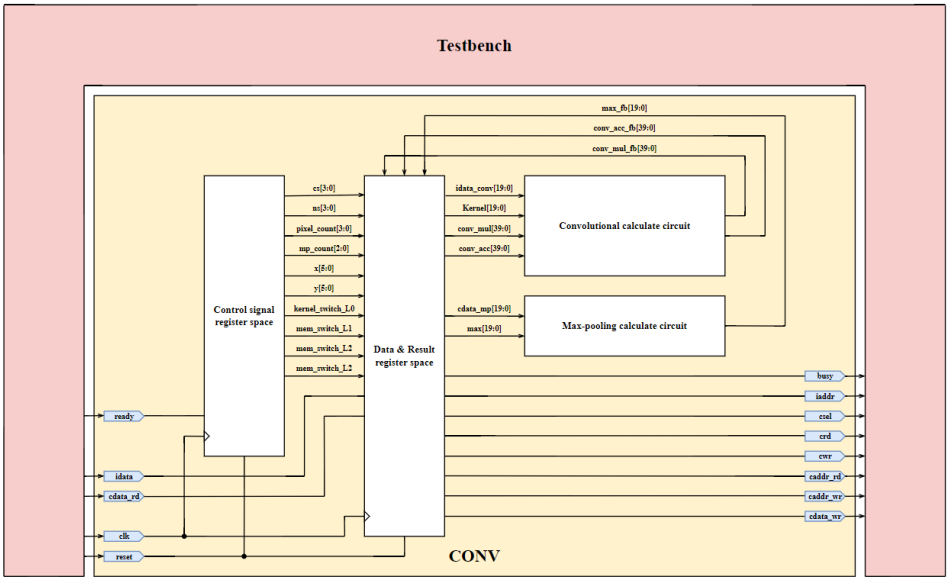


圖 3-1、硬體架構圖

Image Convolution Circuit 由三個 Layer 組成，分別為 Convolution Layer、Max-pooling Layer 以及 Flatten Layer。

3.1.1 Convolutional Layer (Layer 0)

Convolutional Layer 電路功能包含透過輸入影像資料的座標決定是否進行 zero-padding，並將輸入影像資料與 kernel 進行卷積運算後加上 bias 值，最後經過 ReLU 運算後得到 Layer 0 的結果。硬體架構會以有限狀態機、計數器、座標資訊暫存器、運算結果暫存器、時脈訊號與重置訊號實現 Layer 0 的設計。

3.1.2 Max-pooling Layer (Layer 1)

Max-pooling Layer 電路功能包含向外部索取 Layer 0 運算結果的資料、決定四個影像資料的最大值。硬體架構中會以有限狀態機、計數器、座標資訊暫存器、比較器、時脈訊號與重置訊號實現 Layer 1 的設計。

3.1.3 Flatten Layer (Layer 2)

Flatten Layer 電路功能包含向外部索取 Max-pooling Layer 運算結果的資料，並將由 kernel 0、kernel 1 進行 Layer 0、Layer 1 運算後的二維資料，依序交錯輸出成一維資料。硬體架構中會以有限狀態機、計數器、座標資料暫存器、時脈訊號與重置訊號實現 Layer 2 的設計。

第四章　Spyglass

4.1 Spyglass 檢查結果

在將 RTL 進行邏輯合成之前，需要先利用 Spyglass 當中的 lint 功能來檢查程式碼的品質是否良好，並且針對每一行報錯訊息進行修改。此步驟能夠幫助我們在下一階段的邏輯合成中，減少因為不良的 RTL code 導致電路出錯的機率，也能夠幫助我們提高在未來修改或維護程式碼的效率。

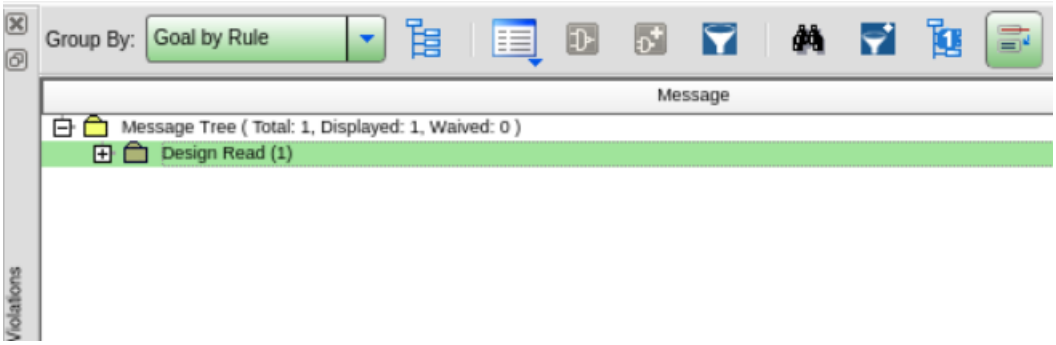


圖 4-1、Spyglass 檢查結果

經 Spyglass 的檢查後，電路中並沒有出現任何多餘的腳位或不良的 Verilog 語法等會影響合成電路品質的因子，可直接進行下一步的邏輯合成。

第五章　邏輯合成結果

5.1 製程選用與合成要求

本專題電路使用台積電 130nm 製程進行邏輯合成。而此次專題需達到競賽評分標準的最高等級，其要求如下。

(1) Layer 0~2 的 RTL 與 Gate-level simulation 結果完全正確。

(2) 面積小於 270,000 (um^2)。

此次專題我們以操作週期 10 (ns)來針對 CONV 以及在卷積層乘法器加入 TMR (三重模組冗餘)的 CONV_TMR 電路進行邏輯合成，其結果如下。

5.2 CONV

5.2.1 CONV 面積

```
Area
-----------------------------------
Combinational Area:    17897.385518
Noncombinational Area:  7234.318558
Buf/Inv Area:            841.910387
Total Buffer Area:           427.74
Total Inverter Area:         414.17
Macro/Black Box Area:      0.000000
Net Area:                  0.000000
-----------------------------------
Cell Area:             25131.704076
Design Area:           25131.704076
```

圖 5-1、CONV 面積

5.2.2 CONV 電路面積占比

```
Hierarchical area distribution
------------------------------

                                Global cell area           Local cell area
                                -------------------  ----------------------------
Hierarchical cell               Absolute    Percent  Combi-      Noncombi-  Black-
                                Total       Total    national    national   boxes   Design
------------------------------  ----------  -------  ----------  ---------  ------  -----------------
CONV                            25131.7041    100.0   6059.7179  7234.3186  0.0000  CONV
add_317                          1806.0336      7.2   1806.0336     0.0000  0.0000  CONV_DW01_add_2
add_327                           351.3618      1.4    351.3618     0.0000  0.0000  CONV_DW01_inc_1
mult_304                         9680.2722     38.5   9680.2722     0.0000  0.0000  CONV_DW_mult_tc_1
------------------------------  ----------  -------  ----------  ---------  ------  -----------------
Total                                                17897.3855  7234.3186  0.0000
```

圖 5-2、CONV 面積占比

5.2.3 CONV Setup-time & Hold-time slack

```
clock clk (rise edge)                    15.00      15.00
clock network delay (ideal)               0.00      15.00
conv_acc_reg_39_/CK (DFFRX1)              0.00      15.00 r
library setup time                       -0.23      14.77
data required time                                  14.77
-----------------------------------------------------------
data required time                                  14.77
data arrival time                                  -14.76
-----------------------------------------------------------
slack (MET)                                         0.01
```

圖 5-3、CONV Setup-time slack

```
clock clk (rise edge)                     5.00       5.00
clock network delay (ideal)               0.00       5.00
L1_result_reg_19_/CK (DFFRX1)             0.00       5.00 r
library hold time                        -0.03       4.97
data required time                                   4.97
-----------------------------------------------------------
data required time                                   4.97
data arrival time                                   -5.42
-----------------------------------------------------------
slack (MET)                                          0.44
```

圖 5-4、CONV Hold-time slack

5.2.4 CONV 功耗 (不含 Testbench)

```
                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register           0.4727        4.7378e-03        6.6503e+06             0.4841  (  76.37%)
sequential         0.0000           0.0000            0.0000             0.0000  (   0.00%)
combinational   8.7664e-02        4.3929e-02        1.8234e+07             0.1498  (  23.63%)
--------------------------------------------------------------------------------------------------
Total              0.5604 mW     4.8667e-02 mW     2.4884e+07 pW          0.6339 mW
```

圖 5-5、CONV 功耗

5.2.5 CONV Gate-level Simulation 結果

```
-----------------------------------------------------

START!!! Simulation Start .....

-----------------------------------------------------

Layer 0 (Convolutional Output) with Kernel 0 is correct !
Layer 0 (Convolutional Output) with Kernel 1 is correct!
Layer 1 (Max-pooling Output) with Kernel 0 is correct!
Layer 1 (Max-pooling Output) with Kernel 1 is correct!
Layer 2 (Flatten  Output) is correct!

-----------------------------------------------------

--------------------- S U M M A R Y -----------------

Congratulations! Layer 0 data have been generated successfully! The result is PASS!!

Congratulations! Layer 1 data have been generated successfully! The result is PASS!!

Congratulations! Layer 2 data have been generated successfully! The result is PASS!!

-----------------------------------------------------

Simulation complete via $finish(1) at time 1289030474 PS + 0
./testfixture.v:267       #(`CYCLE/2); $finish;
ncsim> exit
```

圖 5-6、CONV 電路模擬結果

5.3 CONV_TMR

5.3.1 CONV_TMR 面積

```
Area
-----------------------------------
Combinational Area:    39335.547467
Noncombinational Area:  8556.593124
Buf/Inv Area:           2052.156570
Total Buffer Area:          1201.76
Total Inverter Area:         850.40
Macro/Black Box Area:      0.000000
Net Area:                  0.000000
-----------------------------------
Cell Area:             47892.140591
Design Area:           47892.140591
```

圖 5-7、CONV_TMR 面積

5.3.2 CONV_TMR 電路面積占比

```
Hierarchical area distribution
------------------------------

                                Global cell area            Local cell area
                                ------------------  ----------------------------
Hierarchical cell               Absolute    Percent  Combi-      Noncombi-  Black-
                                Total       Total    national    national   boxes   Design
------------------------------  ----------  -------  ----------  ---------  ------  -----------------
CONV                            47892.1406    100.0   7561.9169  8556.5931  0.0000  CONV
add_368                          1778.8752      3.7   1778.8752     0.0000  0.0000  CONV_DW01_add_2
add_378                           351.3618      0.7    351.3618     0.0000  0.0000  CONV_DW01_inc_1
mult_340                         9880.5654     20.6   9880.5654     0.0000  0.0000  CONV_DW_mult_tc_4
mult_341                         9858.4992     20.6   9858.4992     0.0000  0.0000  CONV_DW_mult_tc_5
mult_342                         9904.3290     20.7   9904.3290     0.0000  0.0000  CONV_DW_mult_tc_3
------------------------------  ----------  -------  ----------  ---------  ------  -----------------
Total                                                39335.5475  8556.5931  0.0000
```

圖 5-8、CONV_TMR 面積占比

5.3.3 CONV_TMR Setup-time & Hold-time slack

```
clock clk (rise edge)                                  15.00      15.00
clock network delay (ideal)                             0.00      15.00
conv_mul_reg_5_/CK (DFFRX1)                             0.00      15.00 r
library setup time                                     -0.23      14.77
data required time                                                14.77
-----------------------------------------------------------------------
data required time                                                14.77
data arrival time                                                -14.76
-----------------------------------------------------------------------
slack (MET)                                                       0.00
```

圖 5-9、CONV_TMR Setup-time slack

```
clock clk (rise edge)                    5.00       5.00
clock network delay (ideal)              0.00       5.00
L1_result_reg_19_/CK (DFFRX1)            0.00       5.00 r
library hold time                       -0.03       4.97
data required time                                  4.97
-------------------------------------------------------------
data required time                                  4.97
data arrival time                                  -5.42
-------------------------------------------------------------
slack (MET)                                        0.44
```

圖 5-10、CONV_TMR Hold-time slack

5.3.4 CONV_TMR 功耗(不含 Testbench)

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)	
register	0.5658	5.4243e-03	7.8724e+06	0.5791	(62.15%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	0.2082	0.1027	4.1837e+07	0.3527	(37.85%)	
Total	0.7740 mW	0.1081 mW	4.9709e+07 pW	0.9318 mW		

圖 5-11、CONV_TMR 功耗

5.3.5 CONV_TMR Gate-level Simulation 結果

```
----------------------------------------------------

START!!! Simulation Start .....

----------------------------------------------------

Layer 0 (Convolutional Output) with Kernel 0 is correct !
Layer 0 (Convolutional Output) with Kernel 1 is correct!
Layer 1 (Max-pooling Output) with Kernel 0 is correct!
Layer 1 (Max-pooling Output) with Kernel 1 is correct!
Layer 2 (Flatten  Output) is correct!

----------------------------------------------------

--------------------- S U M M A R Y ----------------

Congratulations! Layer 0 data have been generated successfully! The result is PASS!!

Congratulations! Layer 1 data have been generated successfully! The result is PASS!!

Congratulations! Layer 2 data have been generated successfully! The result is PASS!!

----------------------------------------------------

Simulation complete via $finish(1) at time 1289030474 PS + 0
./testfixture.v:267       #(`CYCLE/2); $finish;
ncsim> exit
```

圖 5-12、CONV_TMR 電路模擬結果

由以上邏輯合成結果得知，不論 CONV 或 CONV_TMR 電路在操作週期 10 (ns)下皆有達成最高等級。

5.4 邏輯合成結果比較

5.4.1 CONV 電路效能

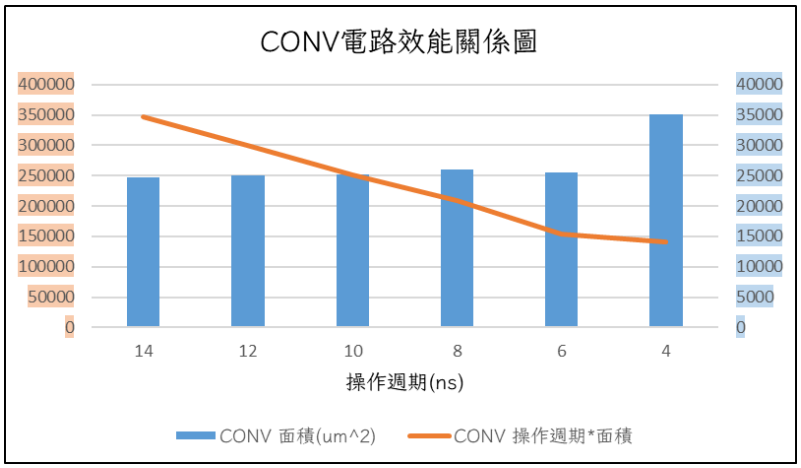


圖 5-13、CONV 電路效能關係圖

Cost function = 操作週期*面積

CONV 電路最小操作週期為 4(ns)，而根據所設定的 Cost function 可得到在操作週期為 4(ns)下會有最好的效能。

5.4.2 CONV_TMR 電路效能

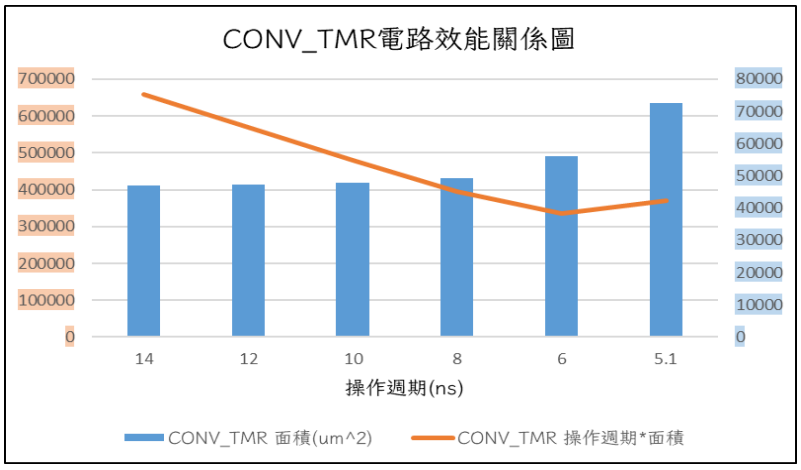


圖 5-14、CONV_TMR 電路效能關係圖

Cost function = 操作週期*面積

CONV_TMR 電路最小操作週期為 5.1(ns)，而根據所設定的 Cost function 可得到在操作週期為 6(ns)下會有最好的效能。

5.4.3 CONV 與 CONV_TMR 面積比較

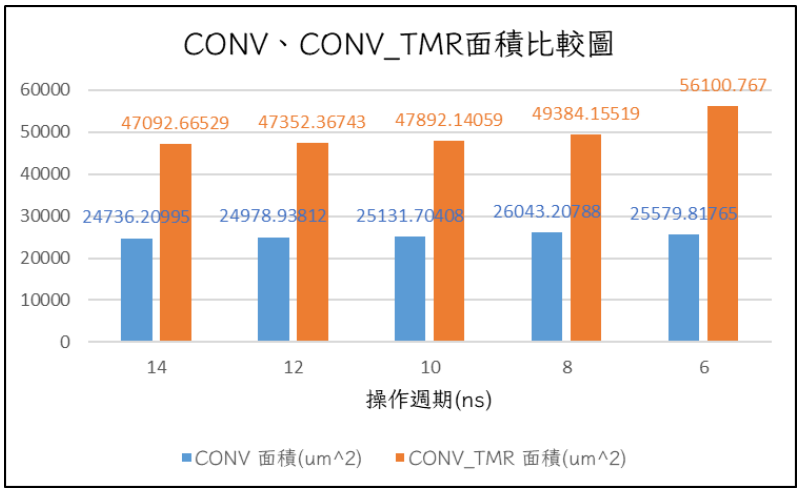


圖 5-15、CONV 與 CONV_TMR 面積比較圖

由上圖結果可發現對乘法器加入 TMR 會造成將近或甚至超過原先電路 2 倍的面積 overhead。

5.4.4 CONV 與 CONV_TMR 電路面積占比

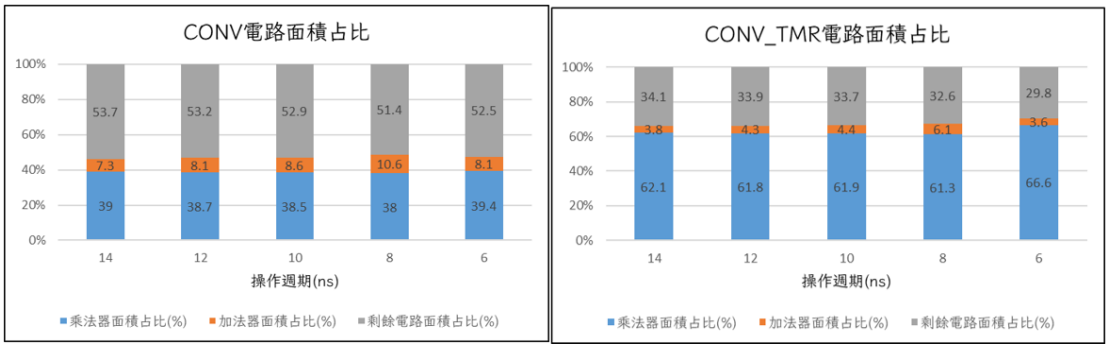


圖 5-16、CONV 與 CONV_TMR 電路面積占比

由上圖結果可發現在不同操作週期下 CONV 電路乘法器大約占了總面積的 40%，加入乘法器 TMR 後，面積占比則會高達電路總面積的 60%以上。而加法器的電路總面積占比相對乘法器則小了許多。

第六章　波形驗證與說明

6.1 波形驗證流程

CONV 電路波形驗證說明將根據 Convolutional Layer、Max-pooling Layer、Flatten Layer 三個階段分別對三層 Layer 的波形圖進行資料接收、電路運算結果、資料輸出與狀態轉換分析，以驗證電路功能是否與預期相符。此處為方便觀察，將訊號區分為 G1、G2、G3、G4 四類(G4 僅 Convolutional Layer 使用)。G1 包含 clk、reset、ready、busy；G2 包含 csel、crd、cwr；G3 包含 iaddr、idata、caddr_rd、cdata_rd、caddr_wr、cdata_wr；G4 包含 Kernel、pixel_count，數值皆以 16 進制表示。

6.1.1 Convolutional Layer 波形驗證

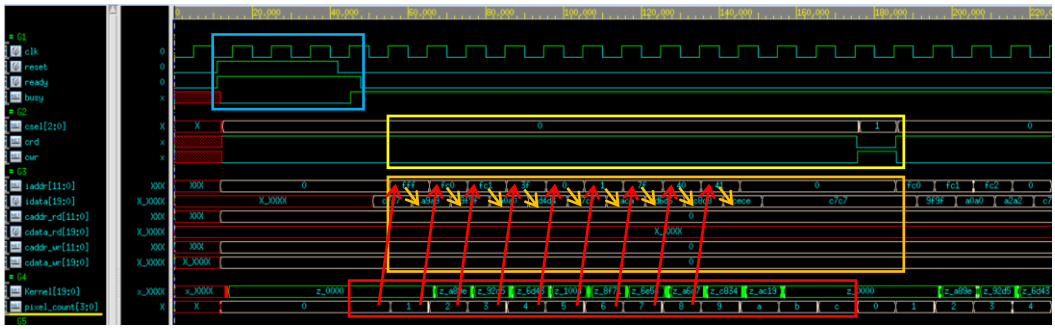


圖 6-1、Layer 0 波形驗證

在 G1 的 reset 與 ready 由邏輯 1 轉為邏輯 0 後，busy = 1 表示電路開始運作，如藍色方框處所示。G4 的 pixel_count = 0~8 時會持續在下個 clk(意即 pixel_count = 1~9 時)正緣處送出 iaddr 位址，如紅色箭頭所示。testbench 在接收到 iaddr 後於同一個週期的 clk 負緣處送入 idata 影像資料，如橙色箭頭所示。因處於讀取資料階段，故 G2 的 crd = 1、cwr = 0、csel = 0，如黃色方框處所示。G4 的 pixel_count = 2 時，第 1 筆資料送入暫存器 A 儲存，Kernel 更新成與第 1 筆資料相乘的數值；pixel_count = 3 時，將第 1 筆資料與 Kernel 數值相乘儲存於暫存器 B，同時釋放暫存器 A 空間供第 2 筆資料使用並更新 Kernel 數值。如此重複 9 個循環並將結果於 G4 的 pixel_count = 3~11 時，將每次運算結果累加至暫存器 C。最後於 G4 的 pixel_count = 12 時，加上 Bias 數值、四捨五入運算與進行 ReLU 運算後將數值儲存於暫存器 D。在下一個週期將暫存器 D 的資料透過 G3 的 caddr_wr 與 cdata_wr 送出，故此時 G2 的 crd = 0、cwr = 1、csel = 1(或 2)，並將 pixel_count 重置完成一次的資料讀取、運算與送出。

6.1.2 Max-pooling Layer 波形驗證

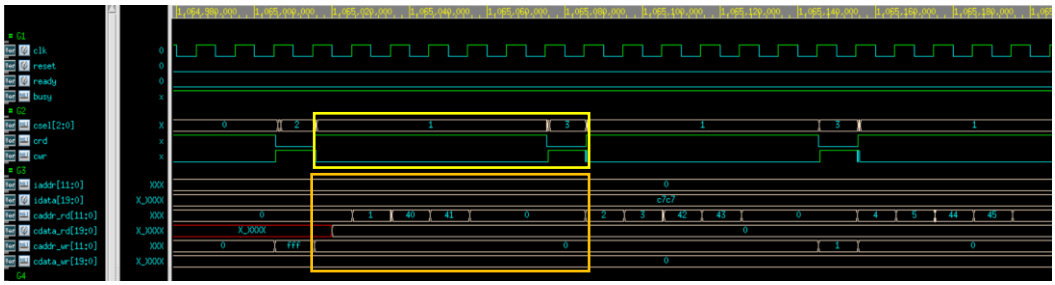


圖 6-2、Layer 1 波形驗證

在 G2 的 crd = 1、cwr = 0、csel = 1 (或 2)時，表示電路處於讀取資料階段，如黃色方框處所示。此時 testbench 會根據 G3 在 clk 正緣處輸出的 caddr_rd 位址資料，於相同 clk 週期的負緣處透過 cdata_rd 輸入所選記憶體中對應位址的資料，如橙色方框處所示。資料輸入後會儲存於暫存器 A，並與暫存器 B(初始值為 0)中的資料比較大小，透過回授電路將最大值更新儲存於暫存器 B，如此重複 4 個循環。於下個 clk 週期，G2 的 crd = 0、cwr = 1、csel = 3 (或 4)，表示電路處於輸出資料階段，如黃色方框處所示。此時 testbench 會根據 G3 在 clk 正緣處輸出的 caddr_wr 位址資料，將 cdata_wr 的資料輸入至所選記憶體中的對應位址，如橙色方框處所示，並將暫存器 B 的資料重置為 0，完成一次 Layer 1 的運算。

6.1.3 Flatten Layer 波形驗證

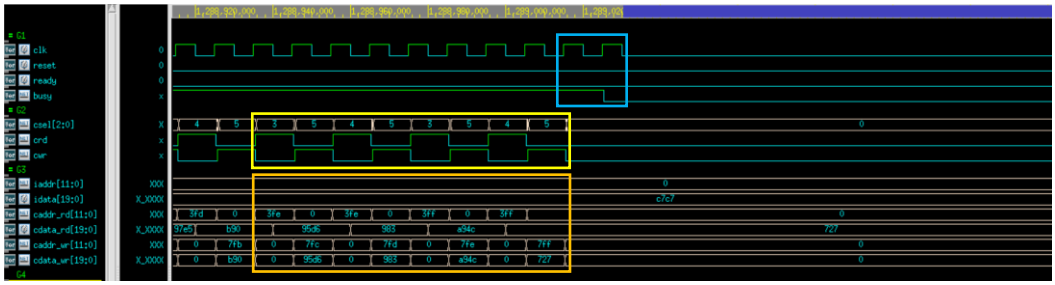


圖 6-3、Layer 2 波形驗證

G2 中 crd = 1、cwr = 0 與 crd = 0、cwr = 1 交替，同時 csel 數值以 3、5、4、5 的規律循環，表示 CONV 電路在 Layer 2 不斷重複讀取 Layer 1 中 Kernel 0 運算結果輸出至 Layer 2 記憶體和讀取 Kernel 1 運算結果輸出至 Layer 2 記憶體，如黃色方框處所示。輸入資料與其記憶體位址由 G3 的 cdata_rd 與 caddr_rd 控制；輸出資料與其記憶體位址則由 G3 的 cdata_wr 與 caddr_wr 控制，如橙色方框處所示。在將 Layer 1 的資料全數交替輸出至記憶體後，在下一個 clk 週期 G1 的 busy 會由 1 轉為 0，表示 CONV 電路運作結束，如藍色方框處所示。

第七章　電路資訊安全性設計

7.1 電路資訊安全性說明

Image Convolution Circuit 主要用於處理大型圖像的辨識，應用範圍涵蓋醫療、交通、娛樂等相當廣泛。然而，在這個過程中可能會受到惡意的第三方攻擊，例如 Man-in-the-middle Attack、Reverse Engineering 與 Fake Replica 等皆為晶片應用方與設計方所擔憂。對於數位 IC 設計工程師而言，在設計晶片功能的同時考慮到資訊安全相關問題是必要的。

7.2 預期目標

7.2.1 功能構想

目前未出現透過單一系統阻絕所有攻擊手段的方法，因此在設計時我們想到最常被使用的方式便是透過金鑰將硬體加密，唯有使用者輸入正確密碼電路方可執行正常的功能；反之則會使電路進入鎖死的狀態或是產生錯誤的輸出。而透過電路的功能敘述，我們認為可將硬體資訊安全設計安裝於 Layer 1，因為電路功能的關係可以確保最後輸出資料的正確性由密碼決定。

7.2.2 實作方式選擇

在電路資訊安全設計上我們欲採用 Logic Locking (或稱為 Key gate)的做法實現。Logic Locking 的電路原理是透過將原電路設計添加新的邏輯閘或多工器，透過外部輸入密碼或是讀取記憶體內的密碼資訊後，決定最後輸出結果的正確性，常見的 Logic Locking 種類分為以下三種。

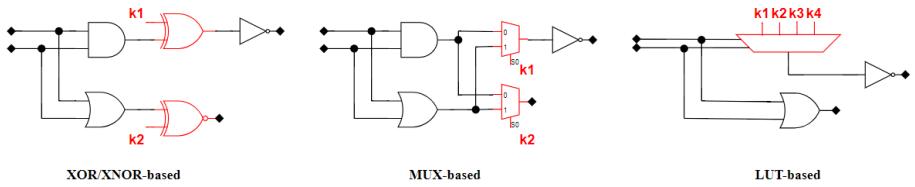


圖 7-1、常見 Logic Locking 種類

然而傳統的 Logic Locking 在實作上會遇到以下瓶頸，邏輯閘的擺放位置應置於何處才不會使其作用遭到後方電路覆蓋或是透過特殊的 pattern 關閉作用。過去常利用貪婪演算法(Greedy Algorithm) 進行初步的擺放位置選定，再使用 key gate insertion algorithm 繼續添加剩餘的邏輯閘。但是在本次專題是透過外部輸入密碼後儲存於暫存器中，因此最後選用一種稱為 Switch Box (SWB)的電路結構，SWB 屬於 MUX-based 的 key gate，同樣可對電路進行簡易加密。

7.3 電路設計

7.3.1 系統 Finite State Machine (FSM)修改

我們在原先的有限狀態機中，於 State_Initial 與 State_Conv 之間新增了一個新狀態 State_Pass 作為讀取外部輸入密碼所使用；另外於電路新增一個 output 腳位 pass，作為向外部傳達密碼已輸入完畢的訊號，新的 FSM 如下圖所示。

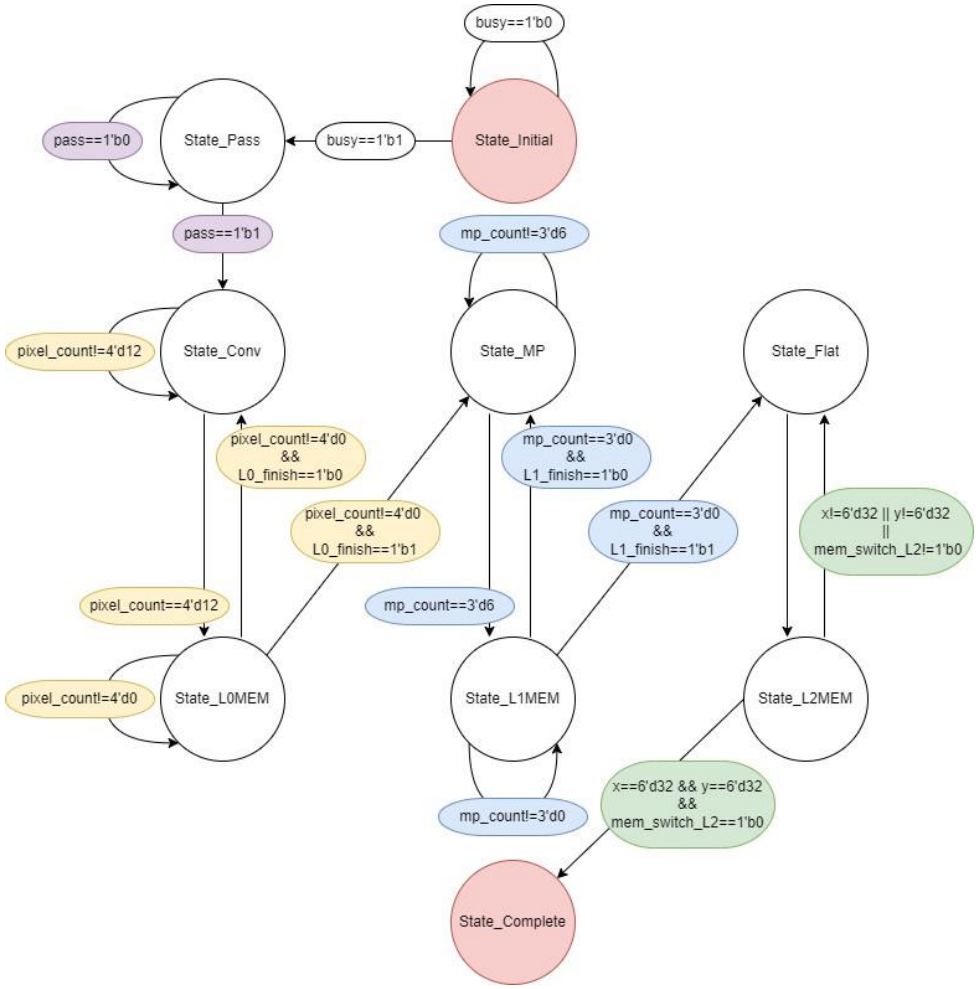


圖 7-2、CONV_SWB 電路系統 FSM

7.3.2 Switch Box (SWB)電路架構實作

SWB 的架構是透過將金鑰資料作為多工器的選擇線並搭配邏輯閘形成的電路單元，有著容易設計且同時具備保護電路的效果，透過將 SWB 電路單元組成陣列可進一步增加其安全性，本次專題主要目的在於驗證與學習 SWB 的基本功能，因此實作上設計了 5 種不同的 SWB，各取 1 個形成 1*5 的簡易陣列，輸入為 Max-pooling Layer 正確運算的結果，輸出為經過 SWB 運算的結果。

7.4 CONV_SWB 邏輯合成結果

7.4.1 CONV_SWB 面積

```
Library(s) Used:

    slow (File: /mnt4/CBDK_IC_Contest_v2.1/SynopsysDC/db/slow.db)

Number of ports:                          408
Number of nets:                          2673
Number of cells:                         2080
Number of combinational cells:           1825
Number of sequential cells:               246
Number of macros/black boxes:               0
Number of buf/inv:                        271
Number of references:                      69

Combinational area:              19279.069118
Buf/Inv area:                     1150.837182
Noncombinational area:            7933.647331
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 27212.716450
Total area:                 undefined
```

圖 7-3、CONV_SWB 電路面積

將電路合成後(使用台積電 130nm 製程)，透過與 CONV 電路互相比較發現安裝 SWB 前後在面積上會由 25131.704076 um^2 上升至 27212.716450 um^2，產生約 8.2804%的額外面積成本。

7.4.2 CONV_SWB 電路面積占比

```
Hierarchical area distribution
------------------------------

                              Global cell area           Local cell area
                              -----------------  ----------------------------
Hierarchical cell             Absolute    Percent  Combi-      Noncombi-  Black-
                              Total       Total    national    national   boxes   Design
------------------------------ ----------  -------  ----------  ---------  ------  -----------------
CONV                          27212.7164    100.0   6448.4225  7933.6473  0.0000  CONV
SWB0                            195.2010      0.7    195.2010     0.0000  0.0000  SWB0
SWB1                            195.2010      0.7    195.2010     0.0000  0.0000  SWB1
SWB2                            195.2010      0.7    195.2010     0.0000  0.0000  SWB2
SWB3                            191.8062      0.7    191.8062     0.0000  0.0000  SWB3
SWB4                            210.4776      0.8    210.4776     0.0000  0.0000  SWB4
add_346                        1806.0336      6.6   1806.0336     0.0000  0.0000  CONV_DW01_add_2
add_356                         351.3618      1.3    351.3618     0.0000  0.0000  CONV_DW01_inc_1
mult_333                       9685.3644     35.6   9685.3644     0.0000  0.0000  CONV_DW_mult_tc_1
------------------------------ ----------  -------  ----------  ---------  ------  -----------------
Total                                                19279.0691  7933.6473  0.0000
```

圖 7-4、CONV_SWB 電路面積占比

通過比較 CONV 與 CONV_SWB 電路面積與模組占比，電路總面積上升約 2081um^2，其中 5 個 SWB 使面積增加約 988um^2，其餘面積增加部分由儲存密碼用的暫存器所造成。

7.4.3 CONV_SWB 功耗(不含 Testbench)

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)	
register	0.0000	0.0000	0.0000	0.0000	(0.00%)	
sequential	0.2862	1.1527e-03	7.3021e+06	0.2946	(69.56%)	
combinational	3.2911e-02	7.7192e-02	1.8818e+07	0.1289	(30.44%)	
Total	0.3191 mW	7.8345e-02 mW	2.6120e+07 pW	0.4236 mW		

圖 7-5、CONV_SWB 功耗

7.4.4 CONV_SWB Gate-level Simulation 結果

在 testbench 的內容上搭配電路資訊安全設計，會在開始輸入影像資料前先向電路輸入一組 20 位元的密碼。同時為了在使用工作站進行電路驗證時能確保密碼成功輸入電路，會在發送完密碼後顯示「Password have been entered !」的字樣，以下將分別顯示安裝 SWB 後輸入正確密碼與輸入錯誤密碼的電路驗證結果。

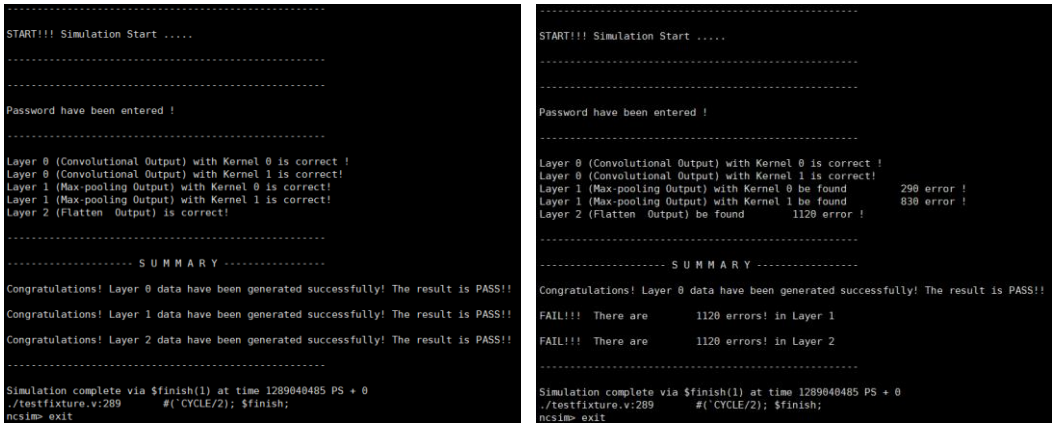


圖 7-6、CONV_SWB 電路模擬結果(左:正確密碼；右:錯誤密碼)

透過修改輸入的金鑰資訊，皆會顯示「Password have been entered !」並且在 10ns 的時脈週期下未發生 timing violation。發現在 Convolutional Layer 的運算結果是不受影響的，但是在輸入錯誤的密碼後的電路於 Max-pooling Layer 便無法產生完全正確的資料輸出以至於 Flatten Layer 亦產生錯誤，代表 SWB 電路實作發揮其預期效果。

第八章　可測試性分析

8.1 操作與設定說明

由於本專題的電路主要由循序電路構成，因此在進行可測試性分析時，會將所有的 flip-flop 換成 scan flip-flop，分別操作在 one scan chain insertion 和 two scan chains insertion 之下，並且將 abort limit 設定為 100000，比較這兩者的可測試性分析結果以及所需的成本。

8.2 可測試性分析結果

8.2.1 One Scan Chain 分析結果

```
Write pattern CPU time                   0.00
Total session CPU time                   3.16
-----------------------------------------------
    Uncollapsed Stuck Fault Summary Report
-----------------------------------------------
fault class                    code   #faults
------------------------------ ----  ---------
Detected                        DT       15257
Possibly detected               PT           0
Undetectable                    UD         111
ATPG untestable                 AU           0
Not detected                    ND           0
-----------------------------------------------
total faults                             15368
test coverage                           100.00%
fault coverage                           99.28%
-----------------------------------------------
Memory usage summary: total=126.42MB
          Pattern Summary Report
-----------------------------------------------
#internal patterns                         157
    #basic_scan patterns                   157
-----------------------------------------------
```

```
****************************************
Report : area
Design : CONV
Version: P-2019.03
Date   : Mon Jan 15 16:41:28 2024
****************************************

Library(s) Used:

    slow (File: /mnt4/CBDK_IC_Contest_v2.1/SynopsysDC/db/slow.db)

Number of ports:                          430
Number of nets:                          2499
Number of cells:                         1825
Number of combinational cells:           1567
Number of sequential cells:               246
Number of macros/black boxes:               0
Number of buf/inv:                        260
Number of references:                      57

Combinational area:              17868.529826
Buf/Inv area:                     1232.312381
Noncombinational area:           10021.449211
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 27889.979038
Total area:                 undefined

***** End Of Report *****
```

圖 8-1、One Scan Chain 分析結果

8.2.2 Two Scan Chain 分析結果

```
Write pattern CPU time                   0.00
Total session CPU time                  10.25
-----------------------------------------------
    Uncollapsed Stuck Fault Summary Report
-----------------------------------------------
fault class                    code   #faults
------------------------------ ----  ---------
Detected                        DT       15265
Possibly detected               PT           0
Undetectable                    UD         111
ATPG untestable                 AU           0
Not detected                    ND           0
-----------------------------------------------
total faults                             15376
test coverage                           100.00%
fault coverage                           99.28%
-----------------------------------------------
Memory usage summary: total=126.43MB
          Pattern Summary Report
-----------------------------------------------
#internal patterns                         160
    #basic_scan patterns                   160
-----------------------------------------------
```

```
****************************************
Report : area
Design : CONV
Version: P-2019.03
Date   : Mon Jan 15 16:31:40 2024
****************************************

Library(s) Used:

    slow (File: /mnt4/CBDK_IC_Contest_v2.1/SynopsysDC/db/slow.db)

Number of ports:                          432
Number of nets:                          2501
Number of cells:                         1826
Number of combinational cells:           1568
Number of sequential cells:               246
Number of macros/black boxes:               0
Number of buf/inv:                        261
Number of references:                      57

Combinational area:              17875.319426
Buf/Inv area:                     1239.101981
Noncombinational area:           10021.449211
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 27896.768637
Total area:                 undefined

***** End Of Report *****
```

圖 8-2、Two Scan Chain 分析結果

8.2.3 分析結果比較

經過 One scan chain insertion 和 Two scan chains insertion 的測試後，兩者的 test coverage 皆為 100%，fault coverage 皆為 99.28%，將面積與未加入 scan flip-flop 的電路相比，One scan chain insertion 的面積成長 2.49%，Two scan chains insertion 的面積成長 2.51%，在比較兩種版本的各項數據後，可以發現 Two scan chains insertion 所需的成本皆大於 One scan chain insertion，然而這些額外的成本並沒有使 fault coverage 上升或分析時間下降，因此在本專題中，適用的方式為 One scan chain insertion。

8.3 Undetectable Fault 分析與說明

```
sa1   UR   mult_333/U835/Y
sa0   --   mult_333/U835/B
sa0   --   mult_333/U835/A
sa1   --   mult_333/U862/A
sa1   UR   mult_333/U835/B
sa1   UR   mult_333/U834/B
sa1   UR   mult_333/U834/Y
sa0   --   mult_333/U834/B
sa0   --   mult_333/U834/A
sa1   --   mult_333/U779/A
sa1   UR   mult_333/U832/Y
sa0   --   mult_333/U832/B
sa0   --   mult_333/U832/A
sa1   --   mult_333/U858/A
sa1   UR   mult_333/U832/B
sa1   UR   mult_333/U833/B
sa1   UR   mult_333/U833/Y
sa0   --   mult_333/U833/B
sa0   --   mult_333/U833/A
```

```
always @(posedge clk or posedge reset) begin
    if(reset)
        conv_mul<=40'd0;
    else if(cs==State_Conv) begin
        if(pixel_count>=4'd2)
            conv_mul<=Kernel*idata_conv;
        else
            conv_mul<=40'd0;
    end
    else
        conv_mul<=40'd0;
end

always @(posedge clk or posedge reset) begin
    if(reset)
        conv_acc<=40'd0;
    else if(cs==State_Conv) begin
        if(pixel_count>=4'd3)
            conv_acc<=conv_acc+conv_mul;
        else
            conv_acc<=40'd0;
    end
    else
        conv_acc<=40'd0;
end

assign conv_bias=(kernel_switch_L0)?(conv_acc+{4'b0,Bias1,16'b0}):(conv_acc+{4'b0,Bias0,16'b0});

assign conv_result=(conv_bias[15])?conv_bias[35:16]+1:conv_bias[35:16];
```

圖 8-3、TetraMAX fault report 與 Verilog code 截圖

根據 TetraMAX 中的 fault report，大部分的 undetectable fault 都是 undetectable redundant，並且都集中在乘法器上，在回推到 RTL code 後，發現問題主要是由 Layer 0 所導致的。

在 Layer 0 中，由於 convolution 會將 20 bits 的圖像資料與 20 bits 的 kernel 相乘，因此需要一組 40 bits 的暫存器來儲存相乘完的結果。然而，在做四捨五入的同時，因為輸出資料的位元大小限制為 20 bits 的關係，需要將儲存在 40 bits 暫存器中的資料進行縮減，只會取其中的第 16 到第 35 個 bit 作為輸出，此動作會導致其餘位元無法傳送至任何的 primary output，使得 ATPG 無法模擬這些只用來計算這幾個位元的邏輯閘上的錯誤，進而讓 fault coverage 無法達到 100%。

第九章　APR

9.1 製程使用與規格說明

本次專題晶片下線製程選用聯電 180nm 製程，電路操作週期為 10(ns)。另外，由於晶片腳位規格限制，故將設計的內容與腳位進行以下微調。

(1) 由 64*64 像素改為 4*4 像素的灰階影像資料。

(2) 單筆資料由 20 位元改為 5 位元表示。

(3) 將 iaddr 與 idata 腳位與 caddr_rd 與 cdata_rd 合併。

9.2 晶片下線流程準備

晶片下線流程需經過許多步驟與 EDA tool 的協助，首先必須使用 Design Vision 準備合成相關檔案如 CONV_syn.v、CONV.sdf、CONV.sdc 與 CONV.dc。接下來在使用 Innovus 前，需要將 CONV_syn.v 檔案中新增一個 CONV_CHIP 模組並將其腳位與原 CONV_syn.v 檔案中的 top module 對接。

```
////////////////////////////////////////////////////////////////////
// Created by: Synopsys DC Expert(TM) in wire load mode
// Version   : P-2019.03
// Date      : Fri Sep  8 20:17:38 2023
////////////////////////////////////////////////////////////////////

module CONV_CHIP ( I_clk, I_reset, O_busy, O_cwr, O_caddr_wr, O_cdata_wr, O_crd, O_caddr_rd,
        I_cdata_rd, O_csel , O_none);

    output [3:0] O_caddr_wr;
    output [4:0] O_cdata_wr;
    output [3:0] O_caddr_rd;
    input [4:0] I_cdata_rd;
    output [2:0] O_csel;
    input I_clk, I_reset;
    output O_busy, O_cwr, O_crd;
    output [1:0] O_none;

    wire [3:0] caddr_wr;
    wire [4:0] cdata_wr;
    wire [3:0] caddr_rd;
    wire [4:0] cdata_rd;
    wire [2:0] csel;
    wire clk, reset;
    wire busy, cwr, crd;
    wire [1:0] none;

    CONV CONV_CHIP ( clk, reset, busy, cwr, caddr_wr, cdata_wr, crd, caddr_rd,
        cdata_rd, csel , none);
    XMD PAD_I_clk (.I(I_clk), .O(clk));
    XMD PAD_I_reset (.I(I_reset), .O(reset));
    XMD PAD_I_cdata_rd4 (.I(I_cdata_rd[4]), .O(cdata_rd[4]));
    XMD PAD_I_cdata_rd3 (.I(I_cdata_rd[3]), .O(cdata_rd[3]));
    XMD PAD_I_cdata_rd2 (.I(I_cdata_rd[2]), .O(cdata_rd[2]));
    XMD PAD_I_cdata_rd1 (.I(I_cdata_rd[1]), .O(cdata_rd[1]));
    XMD PAD_I_cdata_rd0 (.I(I_cdata_rd[0]), .O(cdata_rd[0]));

    YA2GSD PAD_O_busy (.O(O_busy), .I(busy), .E(1'b1));
    YA2GSD PAD_O_cwr (.O(O_cwr), .I(cwr), .E(1'b1));
    YA2GSD PAD_O_crd (.O(O_crd), .I(crd), .E(1'b1));
    YA2GSD PAD_O_csel2 (.O(O_csel[2]), .I(csel[2]), .E(1'b1));
    YA2GSD PAD_O_csel1 (.O(O_csel[1]), .I(csel[1]), .E(1'b1));
    YA2GSD PAD_O_csel0 (.O(O_csel[0]), .I(csel[0]), .E(1'b1));
    YA2GSD PAD_O_none1 (.O(O_none[1]), .I(none[1]), .E(1'b1));
    YA2GSD PAD_O_none0 (.O(O_none[0]), .I(none[0]), .E(1'b1));

    YA2GSD PAD_O_caddr_wr3 (.O(O_caddr_wr[3]), .I(caddr_wr[3]), .E(cwr));
    YA2GSD PAD_O_caddr_wr2 (.O(O_caddr_wr[2]), .I(caddr_wr[2]), .E(cwr));
    YA2GSD PAD_O_caddr_wr1 (.O(O_caddr_wr[1]), .I(caddr_wr[1]), .E(cwr));
    YA2GSD PAD_O_caddr_wr0 (.O(O_caddr_wr[0]), .I(caddr_wr[0]), .E(cwr));
    YA2GSD PAD_O_cdata_wr4 (.O(O_cdata_wr[4]), .I(cdata_wr[4]), .E(cwr));
    YA2GSD PAD_O_cdata_wr3 (.O(O_cdata_wr[3]), .I(cdata_wr[3]), .E(cwr));
    YA2GSD PAD_O_cdata_wr2 (.O(O_cdata_wr[2]), .I(cdata_wr[2]), .E(cwr));
    YA2GSD PAD_O_cdata_wr1 (.O(O_cdata_wr[1]), .I(cdata_wr[1]), .E(cwr));
    YA2GSD PAD_O_cdata_wr0 (.O(O_cdata_wr[0]), .I(cdata_wr[0]), .E(cwr));
    YA2GSD PAD_O_caddr_rd3 (.O(O_caddr_rd[3]), .I(caddr_rd[3]), .E(crd));
    YA2GSD PAD_O_caddr_rd2 (.O(O_caddr_rd[2]), .I(caddr_rd[2]), .E(crd));
    YA2GSD PAD_O_caddr_rd1 (.O(O_caddr_rd[1]), .I(caddr_rd[1]), .E(crd));
    YA2GSD PAD_O_caddr_rd0 (.O(O_caddr_rd[0]), .I(caddr_rd[0]), .E(crd));

endmodule
```

圖 9-1、CONV_CHIP 與原腳位對接內容

下一步須將 CONV.sdc 檔案中的時脈訊號名稱修改對應至 CONV_CHIP 的時脈訊號名稱，並添加 current_design CONV_CHIP 作為 top module 與更改下方所有腳位名稱，另外需準備一份晶片腳位圖排列檔案 CONV_CHIP.io。

```
(globals
   version = 3
   io_order = default
)
(iopad
   (top
   (inst   name="PAD_IOVDD1"          orientation=R180   cell="VCC3IOD"     place_status=placed)
   (inst   name="PAD_O_busy"          orientation=R180                      place_status=placed)
   (inst   name="PAD_O_cwr"           orientation=R180                      place_status=placed)
   (inst   name="PAD_O_crd"           orientation=R180                      place_status=placed)
   (inst   name="PAD_CoreVDD1"        orientation=R180   cell="VCCKD"       place_status=placed)
   (inst   name="PAD_CoreVSS1"        orientation=R180   cell="GNDKD"       place_status=placed)
   (inst   name="PAD_O_csel2"         orientation=R180                      place_status=placed)
   (inst   name="PAD_O_csel1"         orientation=R180                      place_status=placed)
   (inst   name="PAD_O_csel0"         orientation=R180                      place_status=placed)
   (inst   name="PAD_O_none1"         orientation=R180                      place_status=placed)
   )
   (left
   (inst   name="PAD_IOVDD4"          orientation=R270   cell="VCC3IOD"     place_status=placed)
   (inst   name="PAD_I_clk"           orientation=R270                      place_status=placed)
   (inst   name="PAD_I_reset"         orientation=R270                      place_status=placed)
   (inst   name="PAD_I_cdata_rd4"     orientation=R270                      place_status=placed)
   (inst   name="PAD_coreVSS4"        orientation=R270   cell="GNDKD"       place_status=placed)
   (inst   name="PAD_I_cdata_rd3"     orientation=R270                      place_status=placed)
   (inst   name="PAD_I_cdata_rd2"     orientation=R270                      place_status=placed)
   (inst   name="PAD_I_cdata_rd1"     orientation=R270                      place_status=placed)
   (inst   name="PAD_I_cdata_rd0"     orientation=R270                      place_status=placed)
   (inst   name="PAD_IOVSS4"          orientation=R270   cell="GNDIOD"      place_status=placed)
   )
   (bottom
   (inst   name="PAD_IOVSS3"          orientation=R0     cell="GNDIOD"      place_status=placed)
   (inst   name="PAD_O_caddr_wr3"     orientation=R0                        place_status=placed)
   (inst   name="PAD_O_caddr_wr2"     orientation=R0                        place_status=placed)
   (inst   name="PAD_O_caddr_wr1"     orientation=R0                        place_status=placed)
   (inst   name="PAD_O_caddr_wr0"     orientation=R0                        place_status=placed)
   (inst   name="PAD_CoreVDD3"        orientation=R0     cell="VCCKD"       place_status=placed)
   (inst   name="PAD_O_cdata_wr4"     orientation=R0                        place_status=placed)
   (inst   name="PAD_O_cdata_wr3"     orientation=R0                        place_status=placed)
   (inst   name="PAD_O_cdata_wr2"     orientation=R0                        place_status=placed)
   (inst   name="PAD_IOVDD3"          orientation=R0     cell="VCC3IOD"     place_status=placed)
   )
   (right
   (inst   name="PAD_O_none0"         orientation=R90                       place_status=placed)
   (inst   name="PAD_O_cdata_wr1"     orientation=R90                       place_status=placed)
   (inst   name="PAD_O_cdata_wr0"     orientation=R90                       place_status=placed)
   (inst   name="PAD_O_caddr_rd3"     orientation=R90                       place_status=placed)
   (inst   name="PAD_CoreVDD2"        orientation=R90    cell="VCCKD"       place_status=placed)
   (inst   name="PAD_CoreVSS2"        orientation=R90    cell="GNDKD"       place_status=placed)
   (inst   name="PAD_O_caddr_rd2"     orientation=R90                       place_status=placed)
   (inst   name="PAD_O_caddr_rd1"     orientation=R90                       place_status=placed)
   (inst   name="PAD_O_caddr_rd0"     orientation=R90                       place_status=placed)
   (inst   name="PAD_IOVSS2"          orientation=R90    cell="GNDIOD"      place_status=placed)
   )
   (topright
   (inst   name="CORNER0"             orientation=R0     cell="CORNERD"     place_status=placed)
   )
   (topleft
   (inst   name="CORNER1"             orientation=R90    cell="CORNERD"     place_status=placed)
   )
   (bottomright
   (inst   name="CORNER2"             orientation=R270   cell="CORNERD"     place_status=placed)
   )
   (bottomleft
   (inst   name="CORNER3"             orientation=R180   cell="CORNERD"     place_status=placed)
   )
```

圖 9-2、CONV_CHIP.io 檔案內容

將 CONV_CHIP.v 檔案引入 Innovus 後根據操作說明完成晶片繞線佈局圖生成 CONV_CHIP.gds 檔案。最後進行 DRC 與 LVS 驗證，DRC 驗證需確認報錯訊息是否為可接受的假錯，LVS 驗證後的 lvs_test.rep 檔案出現笑臉表示通過。

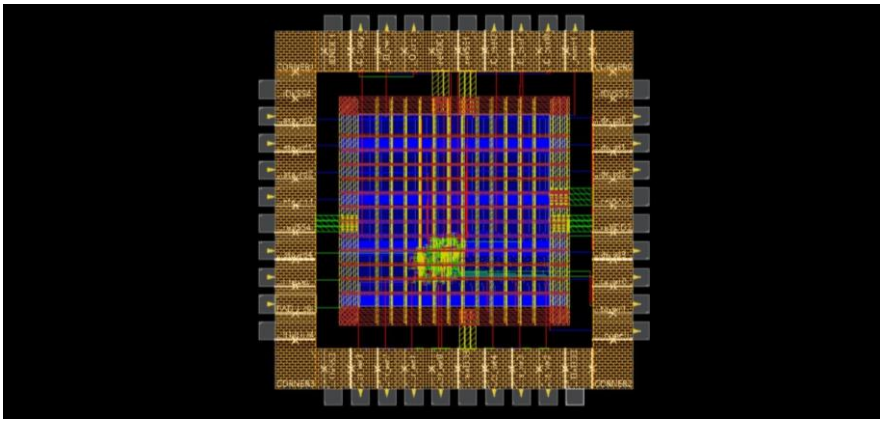


圖 9-3、晶片繞線圖

UMC18 可違反 DRC 假錯列表			
■ RECOMMEND_4.14L	■4.1M	■4.20G	■4.22G
■4.24G	■4.26G	■4.28G	■4.31F
□4.20C	□4.22C	□4.24C	□4.26C
□4.28C	■4.29NOTICE	□4.01Z.NO_IND_OD	■4.14Z.NO_IND_PO1
■4.20F.NO.IND_M1	■4.22F.NO_IND_M2	■4.24F.NO_IND_M3	□4.26F.NO_IND_M4
□4.28F.NO_IND_M5	□4.31E.NO_IND_M6	□6Bb.ME1	■Sanity_1
■IO5.1.W2	■IO5.1.R1	□IO5.2.1.W1.a	□IO5.2.1.W1.b
■IO5.2.2.L1.a	■IO5.2.2.L1.c	□IO5.5.4.Note	■Latch.4.1
■Latch.4.2	■Latch.4.4.pick	□Latch.4.5	■Latch.4.5.pick
■Latch.4.6.guard	■Latch.4.7	■Latch.4.7.guard	■Latch.4.10
■Latch.5.1	□Latch.5.3	□Latch.5.4	■Latch.5.5
■Latch.5.6	■Latch.4.8_Latch.4.9_Latch.5.2		
□5.2A_M3	□5.2B_M3	□Off_Grid	□SkewEdge
□ANT.3.1.1D.ME*	■ANT.3.1.2.NoTE2.VI*		

圖 9-4、DRC 驗證假錯統計圖

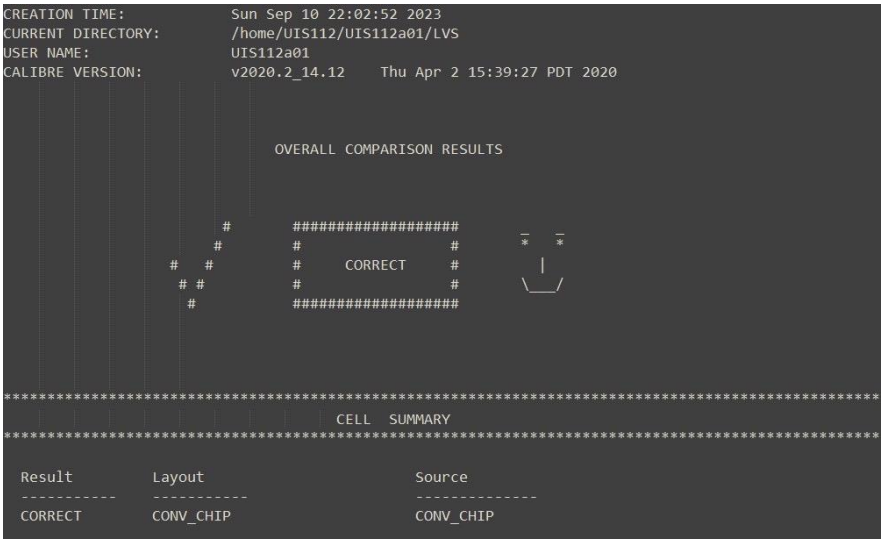


圖 9-5、LVS 驗證結果

9.3 CONV_CHIP_netlist Gate-level Simulation 結果

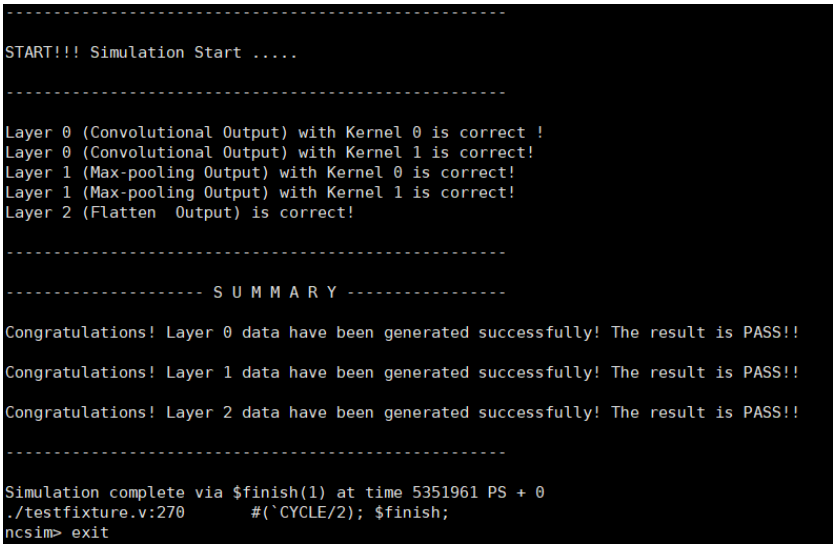


圖 9-6、CONV_CHIP_netlist 模擬結果

Gate-level Simulation 通過，至此完成所有晶片下線步驟與驗證流程。

第十章　專題實作心得

B093011055 劉浩崴

在這次 TARS 實驗室的專題中，我親自實作了數位 IC 從 Front-end 到 Back-end 的設計流程，除了基本的 RTL 設計、邏輯合成和 Gate-level 驗證外，在電路中加入高可靠設計、電路 APR、Fault-coverage 評估等以前僅在課堂中學到而自己從未實做過的內容實踐在自己的電路上則是讓我覺得收穫最大的地方。而在專題分工的部分，為了能讓每個人都能完整操作上述的流程，我們組決定各自設計一份自己的電路，雖然看似不是什麼有效率的分工，但也是因為這樣我才能檢視自己在大學期間設計一份完整電路的能力，以及在跟組員比較、討論的過程中發現自己設計時的盲點和可以再改進的地方，我認為這也是這次專題讓我有特別收穫的部分。最後，重要的還是得感謝振祐和祁穎兩位組員，尤其是在我同時準備研究所推甄和晶片下線的期間，他們倆扛下了絕大部份晶片下線報告的工作以及後續的相關作業，沒有他們的幫忙，我們組的晶片可能會來不及在下線截止日前完成，另外也要感謝鄭育玟助教在工作站以及 EDA 軟體上的指導和協助，讓我們組能夠順利地完成這份專題。

B103012040 林祁穎

藉由這次的專題，將設計晶片的流程從頭到尾跑了一次，在這個過程中，除了熟悉業界常用的 EDA tool 該如何使用之外，我覺得更重要的是讓我了解到自己在設計晶片上還有哪些不足的地方，像是一開始在構思整個電路架構時，還是會習慣使用軟體導向的方式來進行，然而這會使電路面積大幅增加，在重新看過題目說明後，才發現若能夠將資料傳輸跟計算同時運作，不僅可以減少大量面積，連操作時間也能夠減少許多，有了這次的經驗，使我更能夠體會到硬體導向的重要性以及如何利用正緣觸發這個特性，讓電路能夠以 Pipeline 的方式運作。

在這次的專題中，對我來說最困難的地方是 APR 的部分，由於有腳位數量的限制，以及需要學習一個新的 EDA tool，導致在實作時產生了許多問題，不過在解決這些問題的過程中，讓我學習到許多有關晶片下線的知識，也明白了原來晶片要下線還需要考慮到這麼多東西。在 DFT 的部分，原本我以為只需要跟著講義步驟操作就好，但沒有想到在做 DRC 檢查時出現了錯誤，上網查資料後才發現是因為在 TetraMAX 中，會將有用到正緣觸發或是負緣觸發的訊號都視為 clock，這才導致 error 產生，藉由這次的經驗，我才瞭解即使有 EDA tool 能夠幫助我們，DFT 也沒有我原先想的那麼簡單。

最後，謝謝教授在我們有提出問題時，都會很有耐心的解釋給我們聽，也謝謝助教在我們遇到實作過程遇到瓶頸時，都能夠在很短時間內協助我們解決問題。

B103012042 鄧振祐

在本次系統晶片實作專題中，我認為最大的收穫是在大學端能夠將晶片設計的所有流程親自操作一次，並學習到多種 EDA Tools 的使用方法如 Design Vision、Innovus、TetraMAX。在一開始進行電路架構設計時，我原本的想法可能會致使電路面積過大而無法達到競賽 S 等級標準，有賴兩位組員向我提出使用 pipeline 的運算模式，並透過 counter 調控運算才得以讓合成後的電路面積遠小於競賽標準。

整個專題過程中讓我覺得時間最緊湊的階段應該是處理晶片下線報告，由於是第一次接觸 APR，因此在檔案的準備與 EDA Tool 操作上仍顯得生疏。而且 Innovus 的操作量相當大，稍有不慎便是從頭來過，在最後 LVS 驗證出現笑臉的時候可以稍微體會到教授在 PDSD 課堂上說過 IC 設計工程師如釋重負的心情。下線報告書的撰寫我認為比較容易出錯的地方在 DRC 假錯列表的填寫，需要一邊對照 DRC 驗證生成的檔案與下線報告中的內容。

在進行 Test Coverage 與 Fault Coverage 評估時，一開始出現的問題和 reset 訊號有關，在 TetraMAX 這個 EDA Tool 中會自動將正緣觸發與負緣觸發的週期訊號皆視為時脈訊號。在將 one scan chain 與 two scan chain 分別注入原始電路後，透過 Design Vision 更可以清楚觀察 DFT 有無在電路架構上的差異。然而原本 Test Coverage 只有 99.99%，我原本以為是不是因為我加上了 Logic Locking 造成 Test Coverage 未達 100%，其中組員針對電路架構進行深度的探討最後找到出問題的位置，最後也透過助教的協助與建議使電路的 Test Coverage 和 Fault Coverage 分別達到 100%與 99.28%，亦能針對 Fault Coverage 未達 100%的狀況提出合理的解釋與說明。

進行報告書撰寫時，所有組員明確完成自己的分工並協助檢閱報告的正確性，在各階段的實體或線上討論時也都能為專題工作提出具有建設性的建議或是解決方法。最後，非常感謝教授與育玟助教在課餘時間和我們討論電路架構時能向我們講解一些較細微的概念，助教在整個專題流程中當我們遭遇問題遲遲無法解決時總能在短時間內提供有力的建議與協助，使我們的專題進度可以正常推進。

第十一章　參考文獻與資料

[1] Yasin, M., Rajendran, J. J., Sinanoglu, O., & Karri, R. (2015). On improving the security of logic locking. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 35(9), 1411-1424.

[2] Rajendran, J., Zhang, H., Zhang, C., Rose, G. S., Pino, Y., Sinanoglu, O., & Karri, R. (2013). Fault analysis-based logic encryption. IEEE Transactions on computers, 64(2), 410-424.

[3] Sweeney, J., Heule, M. J., & Pileggi, L. (2020, November). Modeling techniques for logic locking. In Proceedings of the 39th International Conference on Computer-Aided Design (pp. 1-9).