Integrated Circuit Design Final Design
Final Project
Image Convolution Circuit Design

電機三 B05901030 陳欽安
電機三 B05901144 王皓仁

Register and Wire

```
Inferred memory devices in process
        in routine CONV line 1032 in file
                '/home/raid7_2/userb05/b05030/ICD_Project/CONV_opt.v'.
=================================================================================
|     Register Name       |    Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=================================================================================
|   convolution3_1_reg    | Flip-flop |  360  |  Y  | N  | Y  | N  | N  | N  | N  |
|   convolution3_0_reg    | Flip-flop |  360  |  Y  | N  | Y  | N  | N  | N  | N  |
|        busy_reg         | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|       iaddr_reg         | Flip-flop |  12   |  Y  | N  | Y  | N  | N  | N  | N  |
|        cwr_reg          | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|       caddr_wr_reg      | Flip-flop |  12   |  Y  | N  | Y  | N  | N  | N  | N  |
|       cdata_wr_reg      | Flip-flop |  20   |  Y  | N  | Y  | N  | N  | N  | N  |
|        csel_reg         | Flip-flop |   3   |  Y  | N  | Y  | N  | N  | N  | N  |
|   state_process_reg     | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|     state_row_reg       | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
|     state_read_reg      | Flip-flop |   3   |  Y  | N  | Y  | N  | N  | N  | N  |
|     state_conv_reg      | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
|   state_maxpool_reg     | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
|    state_column_reg     | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
|  state_calculate_reg    | Flip-flop |   3   |  Y  | N  | Y  | N  | N  | N  | N  |
|    state_output_reg     | Flip-flop |   3   |  Y  | N  | Y  | N  | N  | N  | N  |
|   state_counter_reg     | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
|   convolution1_1_reg    | Flip-flop |  360  |  Y  | N  | Y  | N  | N  | N  | N  |
|       TABLE_reg         | Flip-flop | 5120  |  Y  | N  | Y  | N  | N  | N  | N  |
| out_column_counter_reg  | Flip-flop |   5   |  Y  | N  | Y  | N  | N  | N  | N  |
|        Grid_reg         | Flip-flop |  320  |  Y  | N  | Y  | N  | N  | N  | N  |
|    cal_counter_reg      | Flip-flop |   4   |  Y  | N  | Y  | N  | N  | N  | N  |
|   column_counter_reg    | Flip-flop |   6   |  Y  | N  | Y  | N  | N  | N  | N  |
|    row_counter_reg      | Flip-flop |   6   |  Y  | N  | Y  | N  | N  | N  | N  |
|   output_counter_reg    | Flip-flop |  12   |  Y  | N  | Y  | N  | N  | N  | N  |
|  maxpool_counter_reg    | Flip-flop |  10   |  Y  | N  | Y  | N  | N  | N  | N  |
|  flatten_counter_reg    | Flip-flop |  11   |  Y  | N  | Y  | N  | N  | N  | N  |
|       kernel0_reg       | Flip-flop |  80   |  Y  | N  | Y  | N  | N  | N  | N  |
|       kernel1_reg       | Flip-flop |  80   |  Y  | N  | Y  | N  | N  | N  | N  |
|   interPooling0_reg     | Flip-flop |  40   |  Y  | N  | Y  | N  | N  | N  | N  |
|    maxPooling1_reg      | Flip-flop |  20   |  Y  | N  | Y  | N  | N  | N  | N  |
|   interPooling1_reg     | Flip-flop |  40   |  Y  | N  | Y  | N  | N  | N  | N  |
|    maxPooling0_reg      | Flip-flop |  20   |  Y  | N  | Y  | N  | N  | N  | N  |
|   convolution4_0_reg    | Flip-flop |  360  |  Y  | N  | Y  | N  | N  | N  | N  |
|   convolution4_1_reg    | Flip-flop |  360  |  Y  | N  | Y  | N  | N  | N  | N  |
|   convolution1_0_reg    | Flip-flop |  360  |  Y  | N  | Y  | N  | N  | N  | N  |
|   convolution2_1_reg    | Flip-flop |  360  |  Y  | N  | Y  | N  | N  | N  | N  |
|   convolution2_0_reg    | Flip-flop |  360  |  Y  | N  | Y  | N  | N  | N  | N  |
=================================================================================
```

reg TABLE 用來暫存讀進來的data
reg Grid 一次從TABLE copy 4組convolution所需要的data（16筆）
reg convolution 用來存取一次convolution所需的data （9筆）
reg kernel 用來存取四組convolution的結果 （4筆）
reg maxPooling 用來存取一次maxPooling的結果（1筆）

wire kx_x 用來計算9筆convolution與bias的乘加結果

Finite States Machine and Algorithm

<u>Overall states</u>

IDLE: 當ready=1時，進入到WORK，並且把busy拉高
WORK: 直到輸出最後一筆資料後，把busy拉低，回到IDLE

<u>Read Data states</u>

READ_IDLE:
　　當ready為1時，開始進入READ_INIT read data

READ_INIT:
　　前三列的讀檔方式為直式，也就是addr從0->64->128->1->65->129...，當讀完第三行時就換到READ_UP

READ_UP:
　　第三列之後讀檔方式為橫排四個四個讀，也就是addr從192->193->194->195，就會換到READ_DOWN，等addr 259讀完後又會回到READ_UP繼續讀196->197...

READ_DOWN:
　　第三行之後讀檔方式為橫排四個四個讀，也就是256->257->258->259後，就會換到READ_UP讀196->197...；等到讀完最後一行後，也就是 addr[5:0] 跑到 6’d63後就換到READ_WAIT states，先暫時停止讀檔

READ_WAIT:

為了以防Grid的移動速度被讀檔速度追上，所以等到Grid讀到最後第二步時（column_counter == 6'd60）時，才能回到READ_UP繼續下一列的讀值；直到全部讀完後就換回READ_IDLE

States for Grid's data loading

因為我們只用4x64的TABLE size來暫存data 所以在用Grid計算convolution時需要有三種不同的copy data的方式

FIRST_TYPE:

為第一列的Grid load TABLE data的方式，讀完第一列後換到second_type，同時row_counter+2

SECOND_TYPE:

就是GRID的第一列去copy TABLE的第二列；
就是GRID的第二列去copy TABLE的第三列；
就是GRID的第三列去copy TABLE的第四列；
就是GRID的第四列去copy TABLE的第一列

跑完整列之後 跳到FOURTH_TYPE

FOURTH_TYPE:

就是GRID的第一列去copy TABLE的第四列；
就是GRID的第二列去copy TABLE的第一列；
就是GRID的第三列去copy TABLE的第二列；
就是GRID的第四列去copy TABLE的第三列

跑完整列之後 跳到SECOND_TYPE

Calculation States

CAL_IDLE:

因為在Load 前九筆data 時都還不能做convolution所以先待在CAL_IDLE，當讀到第九筆data時，換到CAL_LOAD

CAL_LOAD:

把TABLE中暫存的data copy到Grid中

CAL_0:

同時將Grid中4組convolution data與kernel 0, kernel 1做乘法，進入CAL_1

CAL_1:

將4組做完乘法的9個值與bias相加並做ReLU，存入reg kernel中，進入CAL_WAIT

CAL_WAIT:

等待output完目前算好的layer 1的結果後，回到CAL_LOAD，準備下一輪的計算

MaxPooling states

MAXPOOL_IDLE:

當CAL_1結束後，就會開始進行計算maxpooling，所以當Calculation state == CAL_1，就換到MAXPOOL_FIRST_STEP

MAXPOOL_FIRST_STEP

kernel上下兩列兩兩比較值的大小，將較大的存入reg interPooling中，接著跳入MAXPOOL_LAST_STEP

MAXPOOL_LAST_STEP:

interPooling中的值兩兩比較大小，得到maxpooling的結果，回到MAXPOOL_IDLE

Output states

OUT_IDLE:

一旦進入到CAL_WAIT，就是可以準備輸出convolution的結果了，所以換到OUT_LAYER_0_KERNEL_0/OUT_LAYER_0_KERNEL_1進行輸出

OUT_LAYER_0_KERNEL_0/OUT_LAYER_0_KERNEL_1:

輸出第0 layer kernel0/第0 layer kernel1 的convolution結果

OUT_LAYER_1_KERNEL_0/OUT_LAYER_1_KERNEL_1:

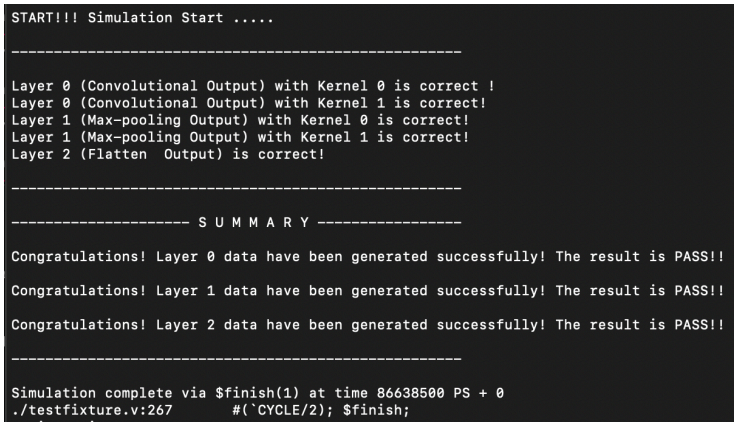
輸出第1 layer kernel0/第0 layer kernel1 的convolution結果

OUT_LAYER_2_KERNEL_0/OUT_LAYER_2_KERNEL_1:

輸出第2 layer kernel0/第2 layer kernel1 的convolution結果

Result 截圖

RTL simulation:



Gate-Level Simulation:

```
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 106640546 PS + 0
./testfixture.v:267       #(`CYCLE/2); $finish;
```

Transistor-Level Simulation:

```
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 133292001 PS + 0
./testfixture.v:267       #(`CYCLE/2); $finish;
```