2023 Digital IC Design Homework 4

NAME	何坤霖
Student ID	N26112445

Simulation Result

Functional simulation	60

```
VSIM 18> run -all
# ------------------------------------------------------
#
# START!!! Simulation Start .....
#
# ------------------------------------------------------
#
# Layer 0 output is correct !
# Layer 1 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!!
#
# terminate at      83981 cycle
# ------------------------------------------------------
#
# ** Note: $finish    : D:/Master shit life/111-2/dic2023/HWs/HW4/testfixture.v(178)
#    Time: 4199050 ns  Iteration: 0  Instance: /testfixture
```

Gate-level simulation	20

```
ModelSim> vsim -gui -L D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/altera -L D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/cycloneive work.testfixtu
# vsim -gui -L D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/altera -L D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/cycloneive work.testfixture
# Start time: 09:32:18 on May 17,2023
# Loading work.testfixture
# Loading work.ATCONV
# Loading work.hard_block
# Loading D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/cycloneive.cycloneive_io_obuf
# Loading D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/cycloneive.cycloneive_io_ibuf
# Loading D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/cycloneive.cycloneive_clkctrl
# Loading D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/cycloneive.cycloneive_mux41
# Loading D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/cycloneive.cycloneive_ena_reg
# Loading D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/cycloneive.cycloneive_lcell_comb
# Loading D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/altera.dffeas
# Loading instances from ATCONV_v.sdo
# Loading D:/dic2023_tool/intelFPGA/20.1/modelsim_ase/altera/verilog/altera.PRIM_GDFF_LOW
# Loading timing data from ATCONV_v.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /testfixture File: D:/Master shit life/111-2/dic2023/HWs/HW4/testfixture.v
VSIM 6> run -all
# ------------------------------------------------------
#
# START!!! Simulation Start .....
#
# ------------------------------------------------------
#
# Layer 0 output is correct !
# Layer 1 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!!
#
# terminate at      83981 cycle
# ------------------------------------------------------
#
# ** Note: $finish    : D:/Master shit life/111-2/dic2023/HWs/HW4/testfixture.v(178)
#    Time: 4199057659 ps  Iteration: 0  Instance: /testfixture
```

Synthesis Result

Total logic elements	508
Total memory bits	0
Embedded multiplier 9-bit elements	0
Total cycle used	83981

Flow Status	Successful - Wed May 17 09:17:21 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	ATCONV
Top-level Entity Name	ATCONV
Family	Cyclone IV E
Device	EP4CE55F23A7
Timing Models	Final
Total logic elements	508 / 55,856 (< 1 %)
Total registers	117
Total pins	82 / 325 (25 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	0 / 308 (0 %)
Total PLLs	0 / 4 (0 %)

Description of your design

這次 lab 大概我寫了 3 個版本，分別是一開始最暴力直接開一個空間存 padding 完的 image，再來是一次 9 個存取完再去 convolution，但是都會超出要求的 logic element 個數，最後定在一次讀一個 pixel 去累扣的方式，或許 lab 圖片也是走這個方式。

State 的設計上面總共使用了 9 個 states，IDLE、BUSY_UP、DATA_IN、CONV、RELU、W_L0，以上是針對 layer0，RADDR_L0_GEN、MAXPOOL、W_L1，這些是針對 layer1(Maxpooling)

最早存去的位置是要特別花一個 state 下去計算，後來發現可以拆出去 combinational 做，所以速度 cycle 數就可以再壓下來 (本來 16xxxx，算位置也是這次難點之一)。

計算位置的 combinational 電路設計，是透過右移達到除法的計算，判斷目前 kernel 中心 (x1 那個, src_addr)位置，是在原始輸入 image 的那個列，行的判斷是透過 & 12'b111111 運算去得到，交叉組合就可以得到，在左上角，上邊緣，右上角，左邊源，右邊源 …etc，由這些位置訊號，去判斷下一個(conv_idx)要取得位置是在原本輸入的哪個位置。

p.s. conv_idx 是用來 iterate 計算一次 conv，要的 9 個 elements 位置以及對應要乘上的 kernel 數值 ; new_src_addr 是用來判斷是不是 conv_idx 0 是不是第一個進來的，因為想要把 conv_idx 壓在 3bits，結果這個訊號也還是多花 1 bit (笑)。

演算法其實很單純，就是設計一個 conv_sum 去累計每次讀取完 pixel (by src_addr and conv_idx) 再計算 conv 的數值，跟 kernel 數值相乘的部分的點

子來自於實驗室同學，因為這題題目設計 fixed floating point 以及數值是 0.0625、0.125、0.25，透過右移就可以不用乘法器達到數值計算。
透過 conv_idx 也可以判斷是不是要扣 bias，進行 relu。
每次算完一組 conv，就把資料寫回對應位置(src_addr，caddr_wr)，以上就算是完成 layer 0。
Maxpooling 這段沿用上面的點子也就相對簡單，透過一個 caddr_rd_coor，記錄目前的 maxpooling 的原點(coordinate)，也可判斷是不是到 62 行(stride = 2, pooling size 2*2)，如果是下個位置就是 caddr_rd_coor + 66。
再透過一個 max_idx，去 layer0_mem 取得剛剛存的資料來 iterate 比大小，類似 sorting 的概念只留最大的數值(max_res)，最後再寫回 layer1_mem 就可以。

*Scoring = (Total logic elements + Total memory bits + 9*Embedded multipliers 9-bit elements) X Total cycle used*

*** Total logic elements must not exceed 1000.**