2023 Digital IC Design Homework 4

NAME	劉兆軒
Student ID	N26112437

Simulation Result

Functional simulation	pass	Gate-level simulation	pass

```
---------- 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    179205 cycle
-----------------------------------------------------
```

```
---------- 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    179205 cycle
-----------------------------------------------------
```

Flow Status	Successful - Fri May 19 23:52:14 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	532 / 55,856 (< 1 %)
Total registers	269
Total pins	82 / 325 (25 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	2 / 308 (< 1 %)
Total PLLs	0 / 4 (0 %)

Synthesis Result

Total logic elements	532
Total memory bits	0
Embedded multiplier 9-bit elements	2
Total cycle used	179205

主要分成兩部分，第一部分處理 layer0，分成 Padding、Row_major Readmem1、Atrous_conv、ReLU、Writemem 等階段，進行 padding、Atrous_conv 和 ReLU 的 function；第二部分處理 layer1，分成 Maxpooling_addr、Maxpooling、Readmem2、 Maxpooling、Max、Writemem2、Stop 等階段，進行 maxpooling 的 function。

Padding:處理 64*64 轉成 68*68 對應的問題
Row_major:將 2 維矩陣轉成 1 維矩陣
Atrous_conv:進行 conv 運算
ReLU:大於 0 為原值，小於 0 為 0
Maxpooling_addr:找到 pooling 對應 4 個值的 addr
Maxpooling:4 個取出前兩個大
Max:取出最大

Description of your design

這次作業主要麻煩的地方是要去處理 padding 的部分，因為有 logic element 限制，所以無法將整個 memory 的值存到 reg 內，這時就要去想辦法利用有限的硬體資源去完成，我的想法是設一個 padding68*68 矩陣，用 i,j 走遍所有位置，其中當 i<2 時，則表示該位置的值是對應原 img64*64 矩陣的 ii=0 位置；若 i>=65，則表示該位置是對應原 img64*64 矩陣的 ii=65 位置。而 j 同理，這樣就可以解決遇到 padding 的問題了，只需要使用 9 個 reg 去 memory 存目前走到位置的值即可，不用將整個 memory 存到 reg 內。

```
Padding:begin
    cwr <= 0;
    csel <= 0;
    //i,j = padded row/col
    //ii,jj = img row/col
    if(down < 3)begin
        if(right < 3)begin
            //if (ii < 0)
            if (i < 2)
                ii <= 0;

            //else if (ii >= img_size)
            else if (i-2 >= 63)
                ii <= 63;

            else
            // ii = i - kernel_size
                ii <= i - 2;

            if (j < 2)
                jj <= 0;

            else if (j-2 >= 63)
                jj <= 63;

            else
                jj <= j-2;

            right <= right + 1;
            j <= j + 2;
        end

        else begin
            down <= down + 1;
            right <= 0;
            i <= i + 2;
            j <= j_begin;
        end
    end
    else begin
        right <= 0;
        down <= 0;
    end

end
```

如上述，原本使用的是二維矩陣去對應目前走訪到的位置，所以要去 memory 找對應位置時，要先轉成 row_major 的形式。轉完後，我是使用 temp 累加 9 次 field 和 kernel 相乘值的方式，最後再進行 relu。

```
Row_major:begin
    //iaddr要等下個clock才有值
    iaddr <= ii * 64 + jj;
end

Readmem1:begin
    data <= idata>>4;

end

Atrous_conv:begin
    if(k == 8) begin
        temp <= temp + data * kernel[k] + bias;
        k <= 0;
    end
    else begin
        temp <= temp + data * kernel[k];
        k <= k + 1;
    end
end
```

```
ReLU:begin
    //走到最右
    if(j_begin == 63) begin
        i_begin <= i_begin + 1;
        j_begin <= 0;
        end
    else
        j_begin <= j_begin + 1;

    caddr_wr <= mem_addr;
    mem_addr <= mem_addr + 1;

    down <= 0;
    right <= 0;
    //ReLU function
    if(temp >= 4096)
        cdata_wr <= 0;
    else
        cdata_wr <= temp;
end
```

Maxpooling:

```
Maxpooling:begin
    if(maxpooling_matrix[maxpooling_index]<maxpooling_matrix[maxpooling_index+1])
        out1 <= maxpooling_matrix[maxpooling_index+1];
    else
        out1 <= maxpooling_matrix[maxpooling_index];

    if(maxpooling_matrix[maxpooling_index+2]<maxpooling_matrix[maxpooling_index+3])
        out2 <= maxpooling_matrix[maxpooling_index+3];
    else
        out2 <= maxpooling_matrix[maxpooling_index+2];
    caddr_wr <=  mem_addr2;
    mem_addr2 <= mem_addr2 + 1;
end
```

這裡主要是最後存值的時候，要注意無條件進位，把剩餘小數點透過 shift 給清掉，無條件進位的寫法要注意。

```
Max:begin
    cwr <= 1;
    csel <= 1;
    //$display("out1 %b",out1);
    //$display("out2 %b",out2);
    if(out1<out2) begin
        //$display("out2 %b",out2);
        if(out2<<9 >= 13'b0001000000000)begin
            //無條件進位
            cdata_wr <=((out2>>4)+1)<<4;
            //$display("out2>>4+1",(out2>>4)+1);
        end
        else begin
            cdata_wr <= (out2>>4)<<4;
            //$display("out2>>4",out2>>4);
        end
    end
    else begin
        //$display("out1 %b",out1);
        if(out1<<9 >= 13'b0001000000000)begin
            cdata_wr <= ((out1>>4)+1)<<4;
            //$display("out1>>4+1 %b",(out1>>4)+1);
        end
        else begin
            cdata_wr <= (out1>>4)<<4;
            //$display(" out1>>4", out1>>4);
        end
    end
end
```

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

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