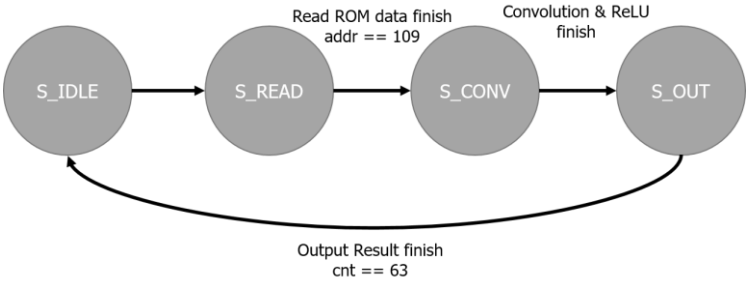
2023 Spring NYCU-EE Machine Learning Intelligent Chip Design – Homework3

IEE 陳冠瑋 310510221 / May. 28, 2023

1. Design Ideas

(1) Finite State Machine

總共設計 4 個 State，分別為 S_IDLE, S_READ, S_CONV 與 S_OUT。




```
if( rst ){
    rom_rd->nb_write( false );
    rom_addr->nb_write( 0 );
    data_out_signal->nb_write( false );
    data_out->nb_write( 0 );
    state = S_IDLE;
    cnt = 0;
    addr = 0;
    row = 0;
    col = 0;
}
else{
    if(state == S_IDLE){
    // Read state
    else if(state == S_READ){
    // Convolution + ReLU
    else if(state == S_CONV){
    // Output
    else if(state == S_OUT){
}
```

(2) Read ROM (S_READ state)

使用 cnt 作為計數器來當作 addr 的參考依據，在實作時發現 nb_read()會 delay 3 個 cycle 從 ROM 讀出 data，因此進入到 S_READ state 的第 3 個 cycle(offset = 3)才能開始將值暫存在 array。

```
// Read state
else if(state == S_READ){
    rom_rd->nb_write( true );
    rom_addr->nb_write( cnt );
    data_out_signal->nb_write( false );
    data_out->nb_write( 0 );

    // Read image
    if( cnt >= offset && cnt < IMG_SIZE*IMG_SIZE + offset ){
        addr = cnt - offset;
        row = addr / IMG_SIZE;
        col = addr % IMG_SIZE;
        data_in->nb_read( img[row][col] );
        if(debug)
            cout << "addr: " << addr << ", pixel["<< row << "][" << col << "]: " << img[row][col] << endl;

    }
    // Read kernel
    else if ( (cnt >= IMG_SIZE*IMG_SIZE + offset) && (cnt < IMG_SIZE*IMG_SIZE + KER_SIZE*KER_SIZE + offset) ){
        addr = cnt - offset;
        row = ( addr - IMG_SIZE*IMG_SIZE ) / KER_SIZE;
        col = ( addr - IMG_SIZE*IMG_SIZE ) % KER_SIZE;
        data_in->nb_read( ker[row][col] );
        if(debug)
            cout << "addr: " << addr << ", weight["<< row << "][" << col << "]: " << ker[row][col] << endl;
    }
    // Read bias
    else if( cnt == IMG_SIZE*IMG_SIZE + KER_SIZE*KER_SIZE + offset ){
        addr = cnt - offset;
        row = 0;
        col = 0;
        data_in->nb_read( bias );
        state =  S_CONV;
        if(debug)
            cout << "addr: " << addr << ", bias: " << bias << endl;
    }
    cnt++;
}
```

(3) Convolution & ReLU (S_CONV state)

由於是使用 SystemC 和 PA 模擬 Convolution + ReLU 的功能，這裡使用較接近軟體的寫法來快速實現 Convolution 與 ReLU 運算。外面使用雙層 for 迴圈控制 output feature map 的 elements，內部用雙層 for 迴圈控制 kernel 每個 weights。累加 partial sum 與加上 bias 後，直接計算 ReLU output 的結果，判斷值是否大於 0，如果大於 0 就為原本的值，如果不是則將值設為 0。

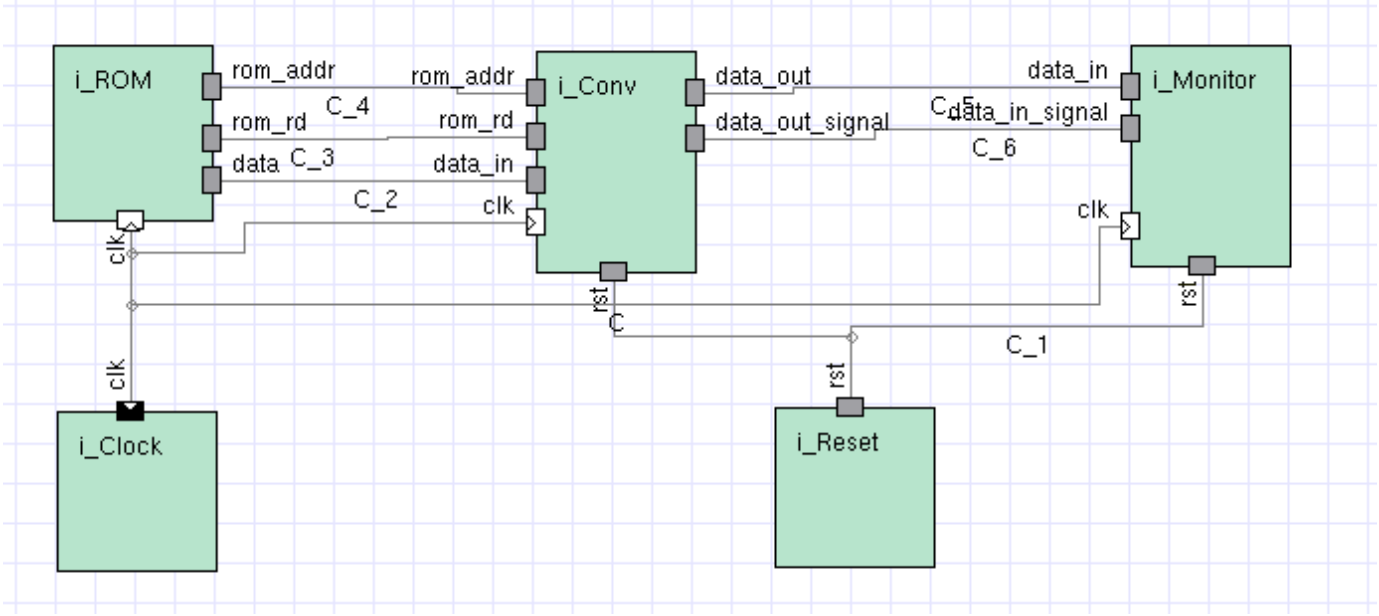
```
// Convolution + ReLU
else if(state == S_CONV){
    rom_rd->nb_write( false );
    cnt = 0;
    addr = 0;
    // Covolution
    for(auto i = 0; i < FMAP_SIZE; ++i){
        for(auto j = 0; j < FMAP_SIZE; ++j){
            sum = 0;
            for(auto ki = 0; ki < KER_SIZE; ++ki){
                for(auto kj = 0; kj < KER_SIZE; ++kj){
                    sum += img[i+ki][j+kj] * ker[ki][kj];
                }
            }
            // ReLU
            fmap[i][j] = ( (sum + bias) > 0 ) ? sum + bias : 0;
        }
    }
    state = S_OUT;
}
```

(4) Output (S_OUT state)

使用計數器來當作 array index 的參考，連續 48 個 cycle 輸出 feature map 的值。

```
// Output
else if(state == S_OUT){
    rom_rd->nb_write( false );
    row =  cnt / FMAP_SIZE;
    col =  cnt % FMAP_SIZE;
    data_out_signal->nb_write( true );
    data_out->nb_write( fmap[row][col] );
    cnt++;
}
```

2. Block Diagram



3. Simulation Result

(1) 透過 Make 編譯後，執行./LAB 模擬的結果

```
ML310510221@ZEUS /home/ML310510221/310510221_HW3/main% ./LAB

        SystemC 2.3.3-Accellera --- Nov 16 2021 18:55:23
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED

------------------------
Result:
Result:
0       177     0       107     0       93      24      102
24      0       0       64      0       17      0       0
10      109     38      0       74      14      69      53
0       21      0       0       24      0       70      0
44      10      96      95      13      76      0       0
0       11      97      7       0       0       0       200
56      173     0       0       81      50      82      0
111     0       96      0       144     0       69      0

Info: /OSCI/SystemC: Simulation stopped by user.
```

(2) 使用 Platform Architect 的模擬結果