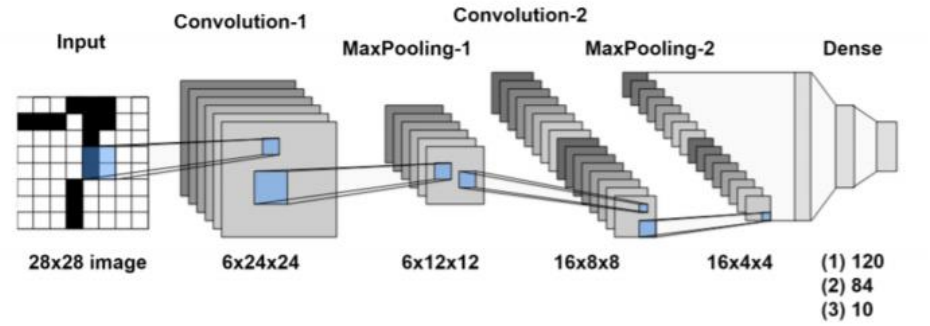
變數宣告:




```
4  DATA_TYPE all_array[45210];
5  DATA_TYPE Conv_1_weight[6][26];      //156=6*(1*5*5+1)
6  DATA_TYPE Conv_2_weight[16][151];    //2416=16*(6*5*5+1)
7  DATA_TYPE Dense_1_weight[120][257]; //30840=120*(256+1)
8  DATA_TYPE Dense_2_weight[84][121];   //10614=84*(120+1)
9  DATA_TYPE Dense_3_weight[10][85];    //850=10*(84+1)
10
```

array 標準化:

all_array : 已知開檔讀檔最為耗時，所以選擇一次性讀取
Conv的Weight : [輸出片數] [輸入片數*權重行*權重列+bias]
Dense的Weight : [輸出長度] [輸入權重長度+bias]

```
11 DATA_TYPE input[1][784];                 //1*28*28
12 DATA_TYPE Output_Conv_1[6][576]={0};     //6*24*24
13 DATA_TYPE Output_Conv_2[16][64]={0};     //16*8*8
14 DATA_TYPE Output_Max_1[6][144]={0};      //6*12*12
15 DATA_TYPE Output_Max_2[16][16]={0};      //16*4*4
16 DATA_TYPE Output_Dense_1[120]={0};
17 DATA_TYPE Output_Dense_2[84]={0};
18 DATA_TYPE Output_Dense_3[10]={0};
19 DATA_TYPE flatten[256];
```

輸入輸出的array也嚴格標準化

Input、Conv、Maxpool 中:[輸出片數][面積]

flatten、Dense : [輸出長度]

array 標準化:

```
21 template<size_t _Array_1_,size_t _Array_2_>
22
23 void Set_Weight_Array(DATA_TYPE (&output_array)[_Array_1_][_Array_2_],int _First_)
24 {
25   for(int FOR_i=0;FOR_i<_Array_1_;FOR_i++)
26   {
27     for(int FOR_j=0;FOR_j<_Array_2_;FOR_j++)
28     {
29       output_array[FOR_i][FOR_j]=all_array[FOR_i*_Array_2_+FOR_j+_First_];
30     }
31   }
32 }
```

用副程式來處理相似度極高的程式碼，不僅能節省空間，最重要的是節省腦袋的算力，不需要一直判斷。在改錯方面，只要改一個地方就好，不會漏改

```
110   else if(i==45214)  //set weight
111   {
112     Set_Weight_Array(Conv_1_weight, 0);
113     Set_Weight_Array(Conv_2_weight, 156);
114     Set_Weight_Array(Dense_1_weight, 2572);
115     Set_Weight_Array(Dense_2_weight, 33412);
116     Set_Weight_Array(Dense_3_weight, 43576);
117     Set_Weight_Array(input, 44426);
118     i++;
119   }
```

主要執行的module變得簡單易懂

計算-Convolution：

Output_Conv_DEF(output_array, output 片數, output 寬/高,

input_array, input 片數, input 寬/高, weight_ array)

```
#define Output_Conv_DEF(output_name, _output_1_, _output_23_,input_name, _input_1_, _input_23_,  weight_name) \
{ \
  for(int FOR_i=0;FOR_i<_output_1_;FOR_i++)  \
  { \
    for(int FOR_j=0;FOR_j<_output_23_;FOR_j++) \
    { \
      for(int FOR_k=0;FOR_k<_output_23_;FOR_k++) \
      { \
        DATA_TYPE Xtemp = 0; \
        for(int FOR_l=0;FOR_l<_input_1_;FOR_l++)   \
        { \
          for(int FOR_m=0;FOR_m<25;FOR_m++) \
          { \
            Xtemp=Xtemp+input_name[FOR_l][(FOR_j+FOR_m/5)*_input_23_+(FOR_k+FOR_m%5)]*weight_name[FOR_i][FOR_l*25+FOR_m]; \
          } \
        } \
        output_name[FOR_i][FOR_j*_output_23_+FOR_k]=ReLU(Xtemp,weight_name[FOR_i][_input_1_*25]); \
      } \
    } \
  } \
}
```

這裡太多變數array不好寫void，
選擇#define一樣可以縮減程式碼。

#define Output_Conv_DEF 就是計算並輸出Conv

```
Output_Conv_DEF(Output_Conv_1, 6, 24, input, 1, 28, Conv_1_weight)
Output_Conv_DEF(Output_Conv_2, 16, 8, Output_Max_1, 6, 12, Conv_2_weight)
```

Output_Conv_DEF(output_array, output片數, output寬/高,
input_array, input片數, input寬/高, weight_ array)

Conv輸出前要先ReLU，下一頁說明

ReLU 判斷 ：

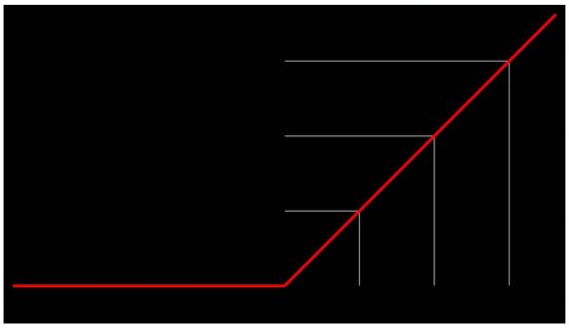
(我把ReLU和bias放在一起計算)

```
87 DATA_TYPE ReLU(DATA_TYPE temp_test,DATA_TYPE bias)
88 {
89   if(temp_test+bias>=0)
90     return temp_test+bias;
91   else
92     return 0;
93 }
```

ReLU :

if (x <= 0) y=0;

else y = x;



計算-Maxpooling :

2*2的方格取最大

```
61 #define MAXpooling_DEF(output_name, _output_1_, _output_23_, input_name) \
62 { \
63   for(int FOR_i=0;FOR_i<_output_1_;FOR_i++) \
64   { \
65     for(int FOR_j=0;FOR_j<_output_23_;FOR_j++) \
66     { \
67       for(int FOR_k=0;FOR_k<_output_23_;FOR_k++) \
68       { \
69         DATA_TYPE temp1 = input_name[FOR_i][FOR_j*2*(2*_output_23_)+FOR_k*2]; \
70         DATA_TYPE temp2 = input_name[FOR_i][FOR_j*2*(2*_output_23_)+FOR_k*2+1]; \
71         DATA_TYPE temp3 = input_name[FOR_i][(FOR_j*2+1)*(2*_output_23_)+FOR_k*2]; \
72         DATA_TYPE temp4 = input_name[FOR_i][(FOR_j*2+1)*(2*_output_23_)+FOR_k*2+1]; \
73         if(temp1>=temp2&&temp1>=temp3&&temp1>=temp4) \
74           output_name[FOR_i][FOR_j*_output_23_+FOR_k] = temp1; \
75         else if(temp2>=temp1&&temp2>=temp3&&temp2>=temp4) \
76           output_name[FOR_i][FOR_j*_output_23_+FOR_k] = temp2; \
77         else if(temp3>=temp1&&temp3>=temp2&&temp3>=temp4) \
78           output_name[FOR_i][FOR_j*_output_23_+FOR_k] = temp3; \
79         else if(temp4>=temp1&&temp4>=temp2&&temp4>=temp3) \
80           output_name[FOR_i][FOR_j*_output_23_+FOR_k] = temp4; \
81       } \
82     } \
83   } \
84 }
```

計算-Dense：

```
126     for(int FOR_i=0;FOR_i<16;FOR_i++)
127     {
128       for(int FOR_j=0;FOR_j<16;FOR_j++)
129       {
130         flatten[FOR_i*16+FOR_j]=Output_Max_2[FOR_j][FOR_i];
131       }
132     }
```

先把Output_Max_2轉成一維陣列(flatten)，要注意排序

```
85 #define Output_Dense_DEF(isReLU, output_name, _output_len_,  input_name, _input_len_, weight_name) \
86 { \
87   for(int FOR_i=0; FOR_i < _output_len_; FOR_i++){ \
88     DATA_TYPE Xtemp = 0; \
89     for(int FOR_j=0; FOR_j < _input_len_;FOR_j++){ \
90       Xtemp=Xtemp+ input_name [FOR_j]* weight_name [FOR_i][FOR_j]; \
91     } \
92     if (isReLU)output_name [FOR_i]= ReLU ( Xtemp , weight_name [FOR_i][_input_len_]); \
93     if (!isReLU)output_name [FOR_i]= Xtemp + weight_name [FOR_i][_input_len_]; \
94   } \
95 }
Output_Dense_DEF(1, Output_Dense_1, 120, flatten, 256, Dense_1_weight);
Output_Dense_DEF(1, Output_Dense_2, 84, Output_Dense_1, 120, Dense_2_weight);
Output_Dense_DEF(0, Output_Dense_3, 10, Output_Dense_2, 84, Dense_3_weight);
```

資料讀取 ：

```
95 void LeNet::run() {
96   ram_wr.write(1);
97   if(i<=45213)  //set input array
98   {
99     if(i>=2){
100      rom_rd.write(1);
101      rom_addr.write(i-2);
102      all_array[i-4]=rom_data_out.read();
103    }
104
105    i++;
106    output_valid.write(0);
107    }
```

array 標準化 ：

```
110   else if(i==45214)  //set weight
111   {
112     Set_Weight_Array(Conv_1_weight, 0);
113     Set_Weight_Array(Conv_2_weight, 156);
114     Set_Weight_Array(Dense_1_weight, 2572);
115     Set_Weight_Array(Dense_2_weight, 33412);
116     Set_Weight_Array(Dense_3_weight, 43576);
117     Set_Weight_Array(input, 44426);
118     i++;
119   }
```

開始計算

```
120 else if(i==45215)  //set Output array
121 {
122   Output_Conv_DEF(Output_Conv_1, 6, 24, input, 1, 28, Conv_1_weight)
123   MAXpooling_DEF(Output_Max_1, 6, 12, Output_Conv_1);
124   Output_Conv_DEF(Output_Conv_2, 16, 8, Output_Max_1, 6, 12, Conv_2_weight)
125   MAXpooling_DEF(Output_Max_2, 16, 4, Output_Conv_2);
126   for(int FOR_i=0;FOR_i<16;FOR_i++)
127   {
128     for(int FOR_j=0;FOR_j<16;FOR_j++)
129     {
130       flatten[FOR_i*16+FOR_j]=Output_Max_2[FOR_j][FOR_i];
131     }
132   }
133   Output_Dense_DEF(1, Output_Dense_1, 120, flatten, 256, Dense_1_weight);
134   Output_Dense_DEF(1, Output_Dense_2, 84, Output_Dense_1, 120, Dense_2_weight);
135   Output_Dense_DEF(0, Output_Dense_3, 10, Output_Dense_2, 84, Dense_3_weight);
136   i++;
137 }
```

輸出：

```
138   else if(i>45215) //output
139   {
140     output_valid.write(true);
141     result.write(Output_Dense_3[i-45216]);
142     i++;
143   }
```

ALL source code: (LeNet.cpp)

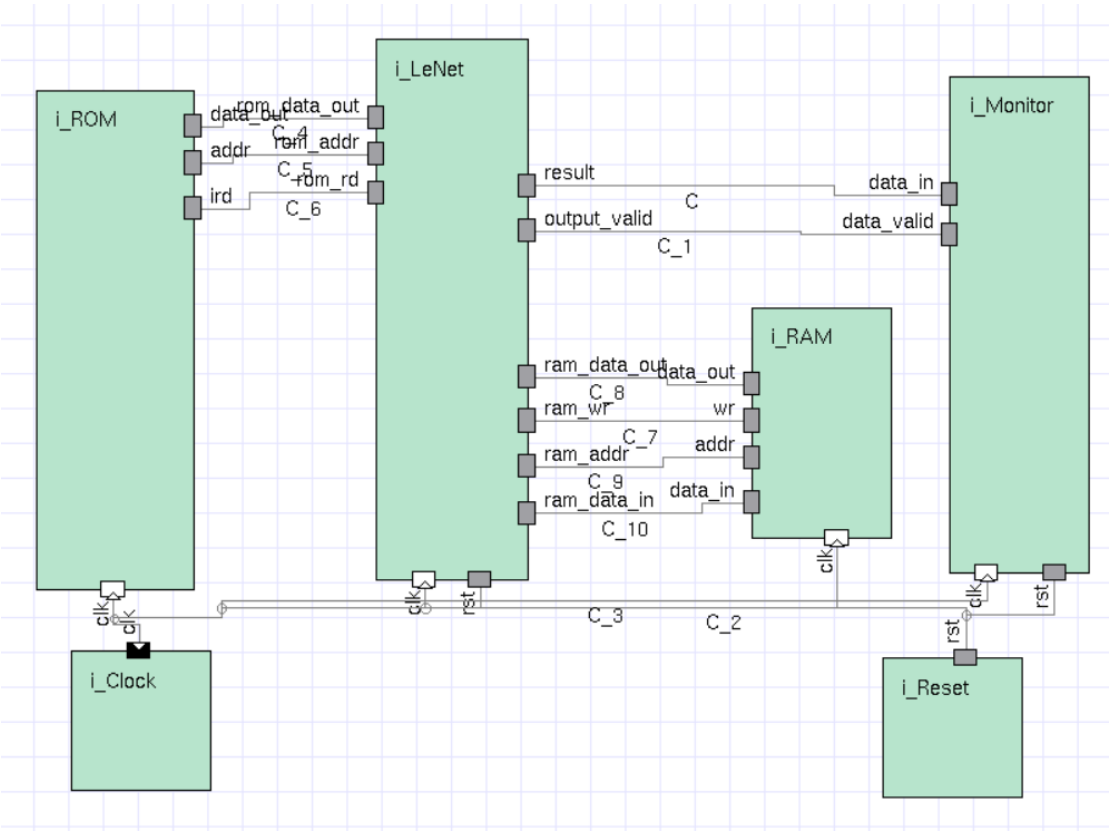
```cpp
#include "LeNet.h"
// vvvvv put your code here vvvvv

DATA_TYPE all_array[45210];
DATA_TYPE Conv_1_weight[6][26];     //156=6*(1*5*5+1)
DATA_TYPE Conv_2_weight[16][151];   //2416=16*(6*5*5+1)
DATA_TYPE Dense_1_weight[120][257]; //30840=120*(256+1)
DATA_TYPE Dense_2_weight[84][121];  //10614=84*(120+1)
DATA_TYPE Dense_3_weight[10][85];   //850=10*(84+1)

DATA_TYPE input[1][784];              //1*28*28
DATA_TYPE Output_Conv_1[6][576]={0};  //6*24*24
DATA_TYPE Output_Conv_2[16][64]={0};  //16*8*8
DATA_TYPE Output_Max_1[6][144]={0};   //6*12*12
DATA_TYPE Output_Max_2[16][16]={0};   //16*4*4
DATA_TYPE Output_Dense_1[120]={0};
DATA_TYPE Output_Dense_2[84]={0};
DATA_TYPE Output_Dense_3[10]={0};
DATA_TYPE flatten[256];
int i=0;
template<size_t _Array_1_,size_t _Array_2_>

void Set_Weight_Array(DATA_TYPE (&output_array)[_Array_1_][_Array_2_],int _First_)
{
  for(int FOR_i=0;FOR_i<_Array_1_;FOR_i++)
  {
    for(int FOR_j=0;FOR_j<_Array_2_;FOR_j++)
    {
      output_array[FOR_i][FOR_j]=all_array[FOR_i*_Array_2_+FOR_j+_First_];
    }
  }
}
DATA_TYPE ReLU(DATA_TYPE temp_test,DATA_TYPE bias)
{
  if(temp_test+bias>=0)
    return temp_test+bias;
  else
    return 0;
}
#define Output_Conv_DEF(output_name, _output_1_, _output_23_,input_name, _input_1_, _input_23_, weight_name) \
{ \
  for(int FOR_i=0;FOR_i<_output_1_;FOR_i++)  \
  { \
    for(int FOR_j=0;FOR_j< _output_23_;FOR_j++) \
    { \
      for(int FOR_k=0;FOR_k< _output_23_;FOR_k++) \
      { \
        DATA_TYPE Xtemp = 0; \
        for(int FOR_l=0;FOR_l<_input_1_;FOR_l++)   \
        { \
          for(int FOR_m=0;FOR_m<25;FOR_m++) \
          { \
            Xtemp=Xtemp+input_name[FOR_l][(FOR_j+FOR_m/5)*_input_23_+(FOR_k+FOR_m%5)]*weight_name[FOR_i][FOR_l*25+FOR_m]; \
          } \
        } \
        output_name[FOR_i][FOR_j* _output_23_+FOR_k]=ReLU(Xtemp,weight_name[FOR_i][_input_1_*25]); \
      } \
    } \
  } \
}
#define MAXpooling_DEF(output_name, _output_1_, _output_23_, input_name) \
{ \
  for(int FOR_i=0;FOR_i<_output_1_;FOR_i++) \
  { \
    for(int FOR_j=0;FOR_j<_output_23_;FOR_j++) \
    { \
      for(int FOR_k=0;FOR_k<_output_23_;FOR_k++) \
      { \
        DATA_TYPE temp1 = input_name[FOR_i][FOR_j*2*(2*_output_23_)+FOR_k*2]; \
        DATA_TYPE temp2 = input_name[FOR_i][FOR_j*2*(2*_output_23_)+FOR_k*2+1]; \
        DATA_TYPE temp3 = input_name[FOR_i][(FOR_j*2+1)*(2*_output_23_)+FOR_k*2]; \
        DATA_TYPE temp4 = input_name[FOR_i][(FOR_j*2+1)*(2*_output_23_)+FOR_k*2+1]; \
        if(temp1>=temp2&&temp1>=temp3&&temp1>=temp4) \
          output_name[FOR_i][FOR_j*_output_23_+FOR_k] = temp1; \
        else if(temp2>=temp1&&temp2>=temp3&&temp2>=temp4) \
          output_name[FOR_i][FOR_j*_output_23_+FOR_k] = temp2; \
        else if(temp3>=temp1&&temp3>=temp2&&temp3>=temp4) \
          output_name[FOR_i][FOR_j*_output_23_+FOR_k] = temp3; \
        else if(temp4>=temp1&&temp4>=temp2&&temp4>=temp3) \
          output_name[FOR_i][FOR_j*_output_23_+FOR_k] = temp4; \
      } \
    } \
  } \
}
#define Output_Dense_DEF(isReLU, output_name, _output_len_,  input_name, _input_len_, weight_name) \
{ \
  for(int FOR_i=0; FOR_i < _output_len_; FOR_i++){ \
    DATA_TYPE Xtemp = 0; \
    for(int FOR_j=0; FOR_j < _input_len_;FOR_j++){ \
      Xtemp=Xtemp+ input_name [FOR_j]* weight_name [FOR_i][FOR_j]; \
    } \
    if (isReLU)output_name [FOR_i]= ReLU ( Xtemp , weight_name [FOR_i][_input_len_]); \
    if (!isReLU)output_name [FOR_i]= Xtemp + weight_name [FOR_i][_input_len_]; \
  } \
}
```

```
void LeNet::run() {
  ram_wr.write(1);
  if(i<=45213)  //set input array
  {
    if(i>=2){
      rom_rd.write(1);
      rom_addr.write(i-2);
      if(i>=4)
        all_array[i-4]=rom_data_out.read();
    }
    i++;
    output_valid.write(0);
    }
  else if(i==45214)  //set weight
  {
    Set_Weight_Array(Conv_1_weight, 0);
    Set_Weight_Array(Conv_2_weight, 156);
    Set_Weight_Array(Dense_1_weight, 2572);
    Set_Weight_Array(Dense_2_weight, 33412);
    Set_Weight_Array(Dense_3_weight, 43576);
    Set_Weight_Array(input, 44426);
    i++;
  }
  else if(i==45215)  //set Output array
  {
    Output_Conv_DEF(Output_Conv_1, 6, 24, input, 1, 28, Conv_1_weight)
    MAXpooling_DEF(Output_Max_1, 6, 12, Output_Conv_1);
    Output_Conv_DEF(Output_Conv_2, 16, 8, Output_Max_1, 6, 12, Conv_2_weight)
    MAXpooling_DEF(Output_Max_2, 16, 4, Output_Conv_2);
    for(int FOR_i=0;FOR_i<16;FOR_i++)
    {
      for(int FOR_j=0;FOR_j<16;FOR_j++)
      {
        flatten[FOR_i*16+FOR_j]=Output_Max_2[FOR_j][FOR_i];
      }
    }
    Output_Dense_DEF(1, Output_Dense_1, 120, flatten, 256, Dense_1_weight);
    Output_Dense_DEF(1, Output_Dense_2, 84, Output_Dense_1, 120, Dense_2_weight);
    Output_Dense_DEF(0, Output_Dense_3, 10, Output_Dense_2, 84, Dense_3_weight);
    i++;
  }
  else if(i>45215) //output
  {
    output_valid.write(true);
    result.write(Output_Dense_3[i-45216]);
    i++;
  }
}
```

ALL source code: (LeNet.h)

```
#include "systemc.h"
#include "define.h"
#include <iostream>

using namespace std;

// vvvvv put your code here vvvvv
SC_MODULE( LeNet ) {
  sc_in_clk clk;
  sc_in < bool > rst;

  sc_out < bool > rom_rd;
  sc_out < sc_uint<16> > rom_addr;
  sc_in < DATA_TYPE > rom_data_out;

  sc_out < bool > ram_wr;
  sc_out < sc_uint<16> > ram_addr;
  sc_in < DATA_TYPE > ram_data_out;
  sc_out < DATA_TYPE > ram_data_in;

  sc_out < DATA_TYPE > result;
  sc_out < bool > output_valid;
  void run();

  SC_CTOR( LeNet )
  {
      SC_METHOD( run );
      sensitive << clk.pos();
  }
};
```

PA



Result

```
done!
0: -6.13672!
1: 1.44434!
2: -4.41699!
3: 1.85547!
4: -9.71387!
5: -0.980469!
6: -10.2959!
7: 14.8623!
8: -4.5459!
9: -1.62402!
```

defined

```
done!
0: -6.47721!
1: 1.60004!
2: -4.60503!
3: 2.01056!
4: -10.0545!
5: -1.03759!
6: -10.7092!
7: 15.4823!
8: -4.84496!
9: -1.56151!
```

undefined