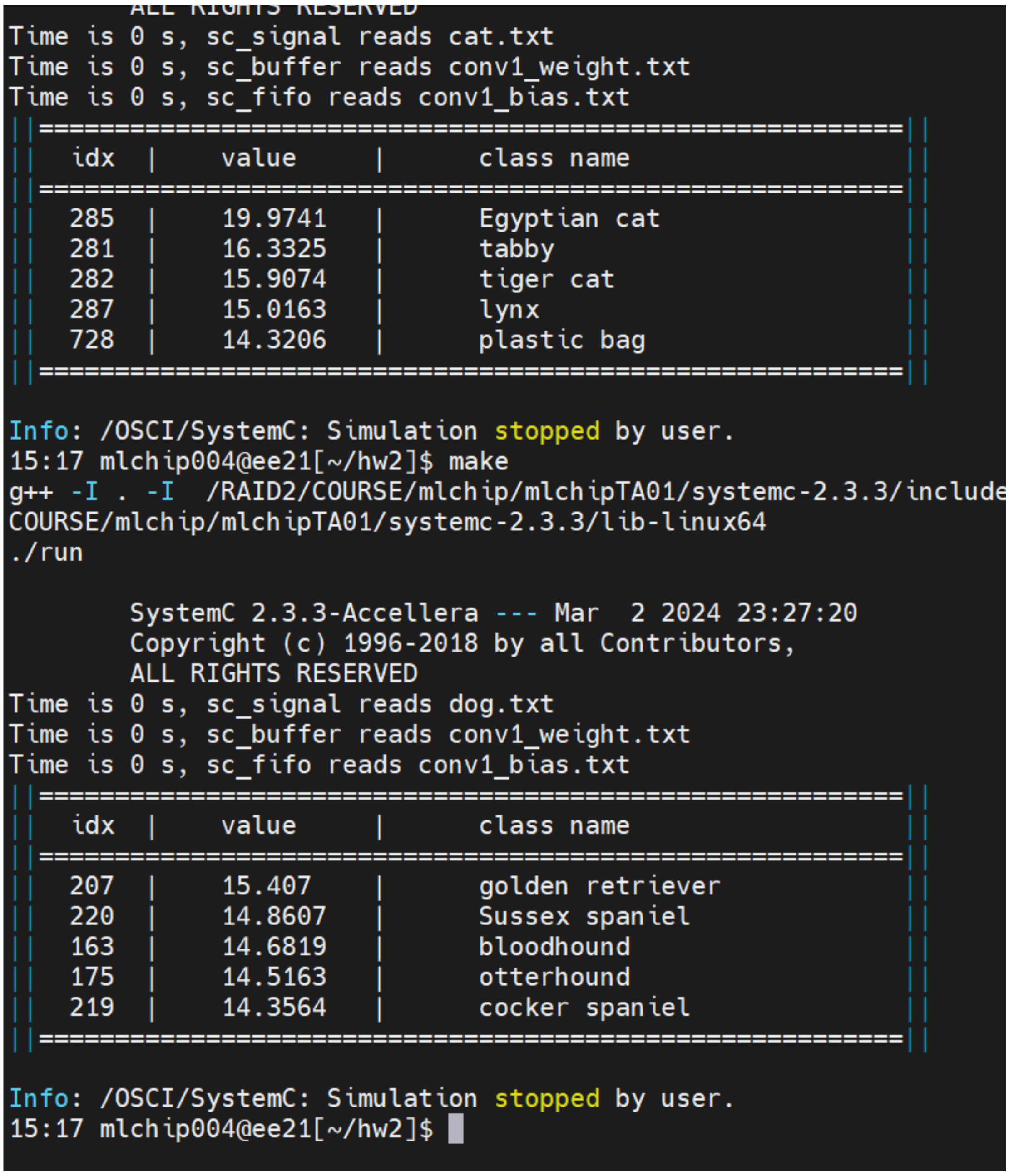
312510190 張家瑋

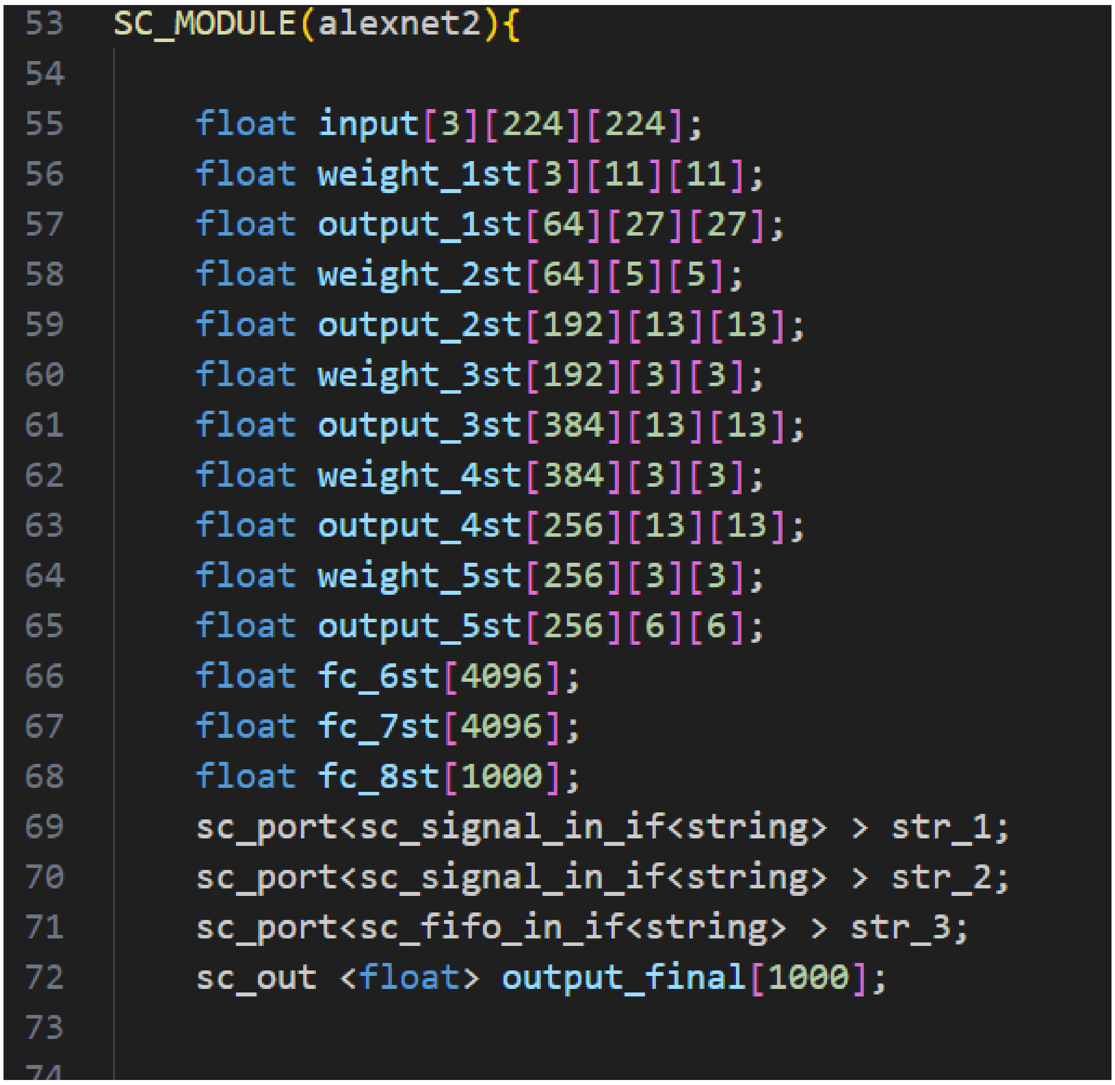
HW2 Report

Simulation results demonstrate the predicted output for the provided input data:


```
        ALL RIGHTS RESERVED
Time is 0 s, sc_signal reads cat.txt
Time is 0 s, sc_buffer reads conv1_weight.txt
Time is 0 s, sc_fifo reads conv1_bias.txt
||=========================================================||
||  idx  |     value      |       class name               ||
||=========================================================||
||  285  |    19.9741     |       Egyptian cat             ||
||  281  |    16.3325     |       tabby                    ||
||  282  |    15.9074     |       tiger cat                ||
||  287  |    15.0163     |       lynx                     ||
||  728  |    14.3206     |       plastic bag              ||
||=========================================================||

Info: /OSCI/SystemC: Simulation stopped by user.
15:17 mlchip004@ee21[~/hw2]$ make
g++ -I . -I  /RAID2/COURSE/mlchip/mlchipTA01/systemc-2.3.3/include
COURSE/mlchip/mlchipTA01/systemc-2.3.3/lib-linux64
./run

        SystemC 2.3.3-Accellera --- Mar  2 2024 23:27:20
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
Time is 0 s, sc_signal reads dog.txt
Time is 0 s, sc_buffer reads conv1_weight.txt
Time is 0 s, sc_fifo reads conv1_bias.txt
||=========================================================||
||  idx  |     value      |       class name               ||
||=========================================================||
||  207  |    15.407      |       golden retriever         ||
||  220  |    14.8607     |       Sussex spaniel           ||
||  163  |    14.6819     |       bloodhound               ||
||  175  |    14.5163     |       otterhound               ||
||  219  |    14.3564     |       cocker spaniel           ||
||=========================================================||

Info: /OSCI/SystemC: Simulation stopped by user.
15:17 mlchip004@ee21[~/hw2]$
```

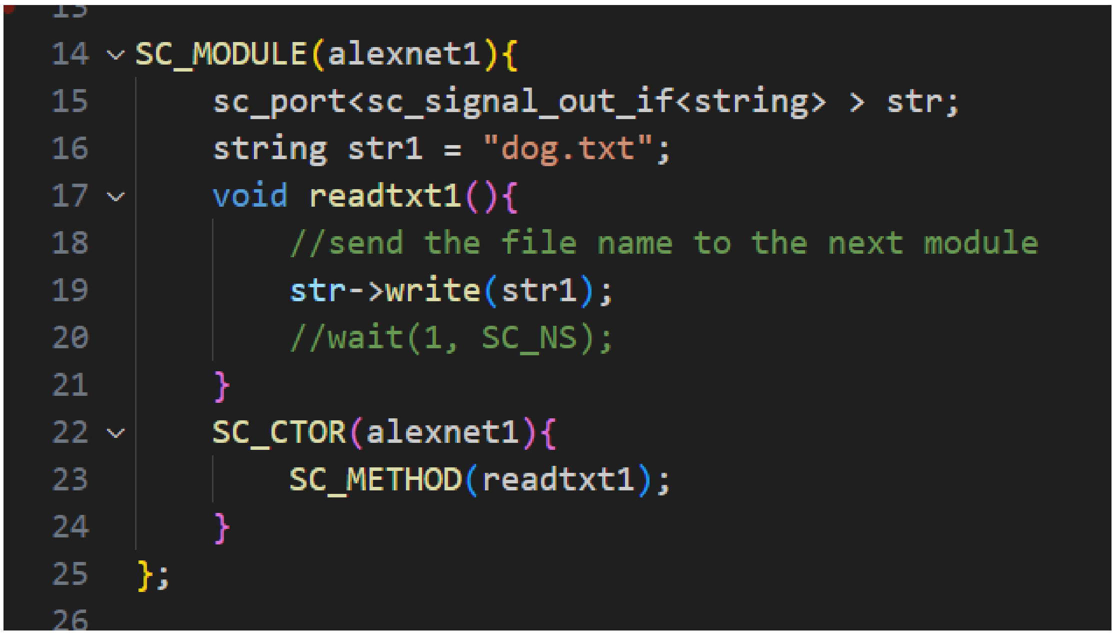
implementation approach:

I divide alexnet module into four submodules, using channel sc_signal, sc_fifo, sc_buffer to connect submodules



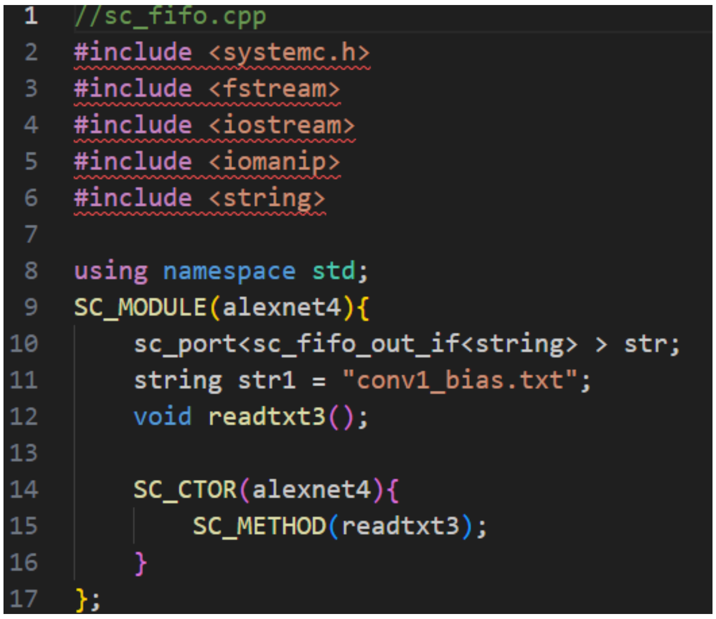
```
SC_MODULE(alexnet2){

    float input[3][224][224];
    float weight_1st[3][11][11];
    float output_1st[64][27][27];
    float weight_2st[64][5][5];
    float output_2st[192][13][13];
    float weight_3st[192][3][3];
    float output_3st[384][13][13];
    float weight_4st[384][3][3];
    float output_4st[256][13][13];
    float weight_5st[256][3][3];
    float output_5st[256][6][6];
    float fc_6st[4096];
    float fc_7st[4096];
    float fc_8st[1000];
    sc_port<sc_signal_in_if<string> > str_1;
    sc_port<sc_signal_in_if<string> > str_2;
    sc_port<sc_fifo_in_if<string> > str_3;
    sc_out <float> output_final[1000];

```

The functionality of each submodule is transmit file address string successively:

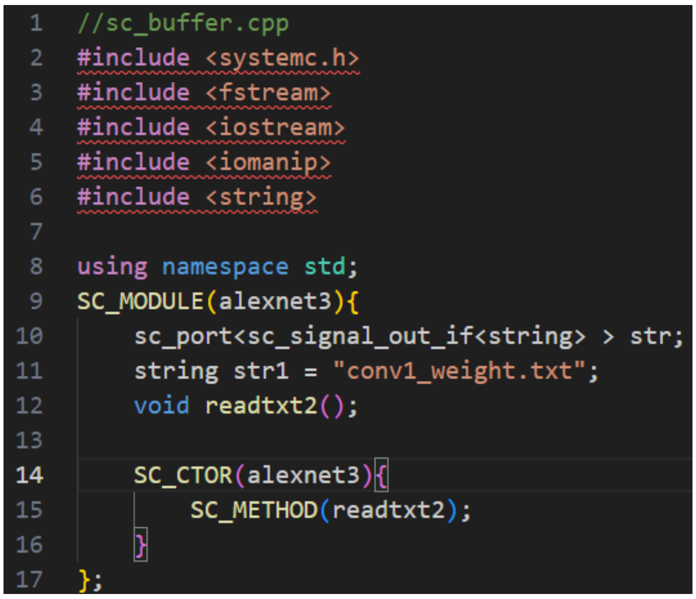


```
SC_MODULE(alexnet1){
    sc_port<sc_signal_out_if<string> > str;
    string str1 = "dog.txt";
    void readtxt1(){
        //send the file name to the next module
        str->write(str1);
        //wait(1, SC_NS);
    }
    SC_CTOR(alexnet1){
        SC_METHOD(readtxt1);
    }
};
```

To compile main.cpp of each channels, I write two header file to compile the whole design, the goal of sc_signal is passing address of image file, where sc_fifp is passing address of weight file, and sc_buffer is passing address of bias file:



```
//sc_fifo.cpp
#include <systemc.h>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;
SC_MODULE(alexnet4){
    sc_port<sc_fifo_out_if<string> > str;
    string str1 = "conv1_bias.txt";
    void readtxt3();

    SC_CTOR(alexnet4){
        SC_METHOD(readtxt3);
    }
};
```

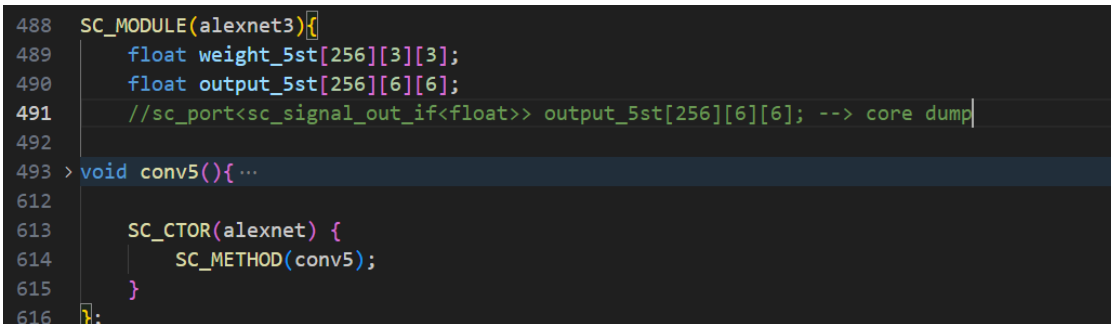


```
//sc_buffer.cpp
#include <systemc.h>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;
SC_MODULE(alexnet3){
    sc_port<sc_signal_out_if<string> > str;
    string str1 = "conv1_weight.txt";
    void readtxt2();

    SC_CTOR(alexnet3){
        SC_METHOD(readtxt2);
    }
};
```

challenges faced:

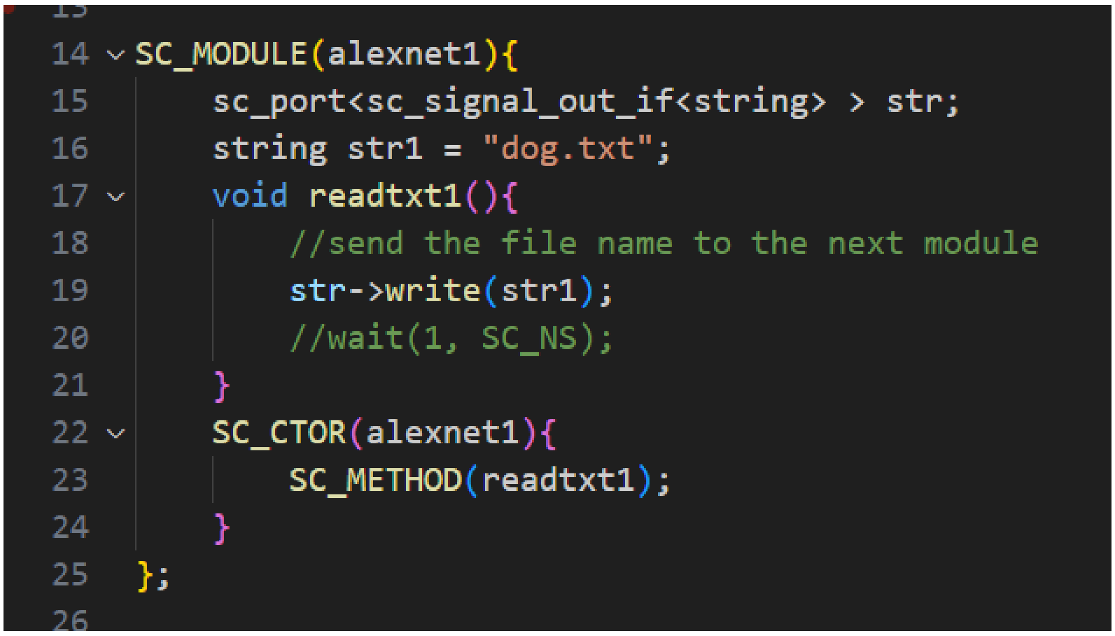
I tried to pass matrix with size of 256*6*6 by sc_signal, but came out a core dump issue.

According to the stack storage limitation in sc_core, it's hard to pass large matrix through submodules, which cause "core dump" issue:



```
SC_MODULE(alexnet3){
    float weight_5st[256][3][3];
    float output_5st[256][6][6];
    //sc_port<sc_signal_out_if<float>> output_5st[256][6][6]; --> core dump

> void conv5(){…

    SC_CTOR(alexnet) {
        SC_METHOD(conv5);
    }
};
```

Instead, I pass file address using channels, which is more proper and also usable:

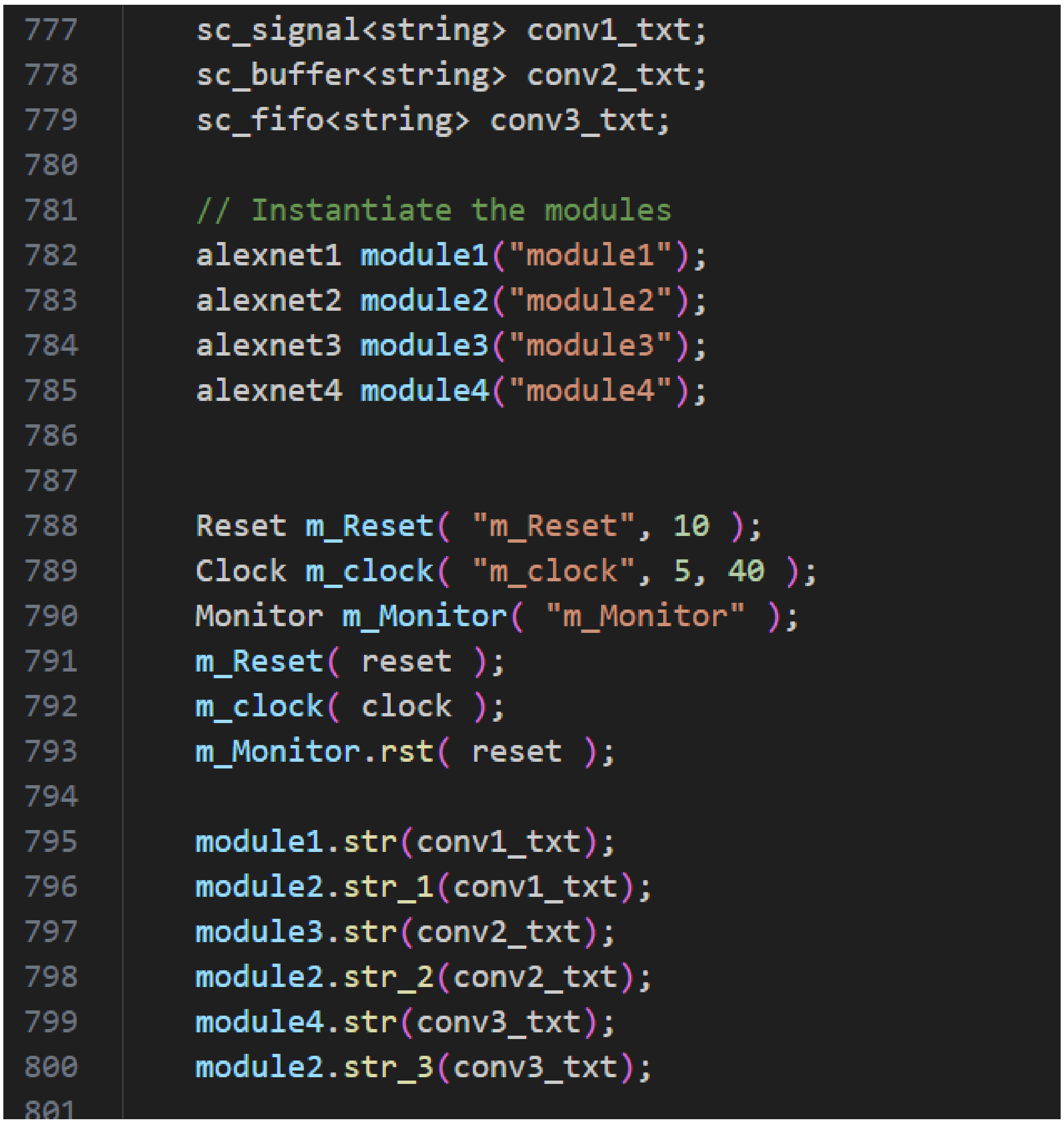


```
SC_MODULE(alexnet1){
        sc_port<sc_signal_out_if<string> > str;
        string str1 = "dog.txt";
        void readtxt1(){
            //send the file name to the next module
            str->write(str1);
            //wait(1, SC_NS);
        }
        SC_CTOR(alexnet1){
            SC_METHOD(readtxt1);
        }
};
```

other observations or insights gained:

The concepts of channels and interfaces in SystemC play a crucial role in facilitating communication between different modules or components of a system.This separation of interface and implementation allows for clear abstraction and encapsulation of functionality, enabling modular design and reusability.

The use of ports, interfaces, and channels facilitates hierarchical design in SystemC, where complex systems are built by composing simpler modules. Modules communicate with each other through well-defined interfaces, promoting modularity, scalability, and maintainability of the overall system.



```
    sc_signal<string> conv1_txt;
    sc_buffer<string> conv2_txt;
    sc_fifo<string> conv3_txt;

    // Instantiate the modules
    alexnet1 module1("module1");
    alexnet2 module2("module2");
    alexnet3 module3("module3");
    alexnet4 module4("module4");


    Reset m_Reset( "m_Reset", 10 );
    Clock m_clock( "m_clock", 5, 40 );
    Monitor m_Monitor( "m_Monitor" );
    m_Reset( reset );
    m_clock( clock );
    m_Monitor.rst( reset );

    module1.str(conv1_txt);
    module2.str_1(conv1_txt);
    module3.str(conv2_txt);
    module2.str_2(conv2_txt);
    module4.str(conv3_txt);
    module2.str_3(conv3_txt);
```