312510190 張家瑋

HW1 Report

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

```
        SystemC 2.3.3-Accellera --- Mar  2 2024 23:27:20
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
||=========================================================||
||   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.
22:53 mlchip004@ee21[~/hw1]$ make
g++ -I . -I  /RAID2/COURSE/mlchip/mlchipTA01/systemc-2.3.3/include -L
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
||=========================================================||
||   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.
23:04 mlchip004@ee21[~/hw1]$
```

implementation approach:

I construct one module called alexnet to include all convolutions layers and fc-layers:

```
SC_MODULE(alexnet){
    sc_out <float> output_final[1000];
    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][13][13];
    float fc_6st[4096];
    float fc_7st[4096];
    float fc_8st[1000];
```

In the module, all the layers are written as functions, separately:

```
    SC_CTOR(alexnet) {
        SC_METHOD(conv1);
        SC_METHOD(conv2);
        SC_METHOD(conv3);
        SC_METHOD(conv4);
        SC_METHOD(conv5);
        SC_METHOD(fc6);
        SC_METHOD(fc7);
        SC_METHOD(fc8);
    }

};
```

One function complete all the calculate operations, including padding, convolution, bias addition, activation function, max-pooling:

```cpp
void conv1(){
    int INPUT_CHANNELS = 3;
    int INPUT_HEIGHT = 224;
    int INPUT_WIDTH = 224;
    int OUTPUT_CHANNELS = 64;
    int OUTPUT_HEIGHT = 55;
    int OUTPUT_WIDTH = 55;
    int KERNEL_SIZE = 11;
    int ZERO_PADDING = 1;
    int STRIDE_SIZE = 4;
    int pool_size = 3;
    int pool_stride = 2;
    int pool_output_height = 27;
    int pool_output_width = 27;

    float output[OUTPUT_CHANNELS][OUTPUT_HEIGHT][OUTPUT_WIDTH];
    float bias[OUTPUT_CHANNELS];
    // Create convolution module
    //Convolution convolution("convolution");

    // Load input data from file
    std::ifstream input_file("dog.txt");
    if (input_file.is_open()) {
        float temp;
        for (int ic = 0; ic < INPUT_CHANNELS; ic++) {
            for (int ih = 0; ih < INPUT_HEIGHT; ih++) {
                for (int iw = 0; iw < INPUT_WIDTH; iw++) {
                    input_file >> temp;
                    input[ic][ih][iw] = temp;
                }
            }
        }
        input_file.close();
```

```cpp
    // Perform convolution
    for (int oc = 0; oc < OUTPUT_CHANNELS; oc++) {
        for (int oh = 0; oh < OUTPUT_HEIGHT; oh++) {
            for (int ow = 0; ow < OUTPUT_WIDTH; ow++) {
                float sum = 0;

                for (int ic = 0; ic < INPUT_CHANNELS; ic++) {
                    for (int kh = 0; kh < KERNEL_SIZE; kh++) {
                        for (int kw = 0; kw < KERNEL_SIZE; kw++) {
                            if(oh == 0 && ow == 0){
                                weight_file >> weight_1st[ic][kh][kw];
                            }
                            int ih = oh * STRIDE_SIZE + kh - ZERO_PADDING;
                            int iw = ow * STRIDE_SIZE + kw - ZERO_PADDING;

                            if (ih >= 0 && ih < INPUT_HEIGHT && iw >= 0 && iw < INPUT_WIDTH) {
                                sum += input[ic][ih][iw] * weight_1st[ic][kh][kw];
                            }
                        }
                    }
                }

               output[oc][oh][ow] = sum;
            }
        }
    }
    weight_file.close();
```

After the computations, I construct a monitor module to print out the outputs from alexnet:

```cpp
SC_MODULE( Monitor ) {
    sc_in < bool > rst;
    sc_in < float > fc_8st [1000];

    void print_output(){
        float f1 = 0, f2 = 0, f3 = 0, f4 = 0, f5 = 0;
        int idx1 = 0, idx2 = 0, idx3 = 0, idx4 = 0, idx5 = 0;
        string s1, s2, s3, s4, s5, temp_str;

        ifstream class_name_file("imagenet_classes.txt");
        if(!class_name_file){
            cout << "Can't open imagenet_classes.txt for reading\n";
        }
        cout<<"||=========================================================||\n";
        cout<<left<<"||  "<<setw(3)<<"idx"<<"  |    "<<setw(7)<<"value"<<"   |      "<<setw(20)<<"class name"<<"        ||\n";
        cout<<"||=========================================================||\n";
```

challenges faced:

At first, I use lots of float-type arrays to store values of weights, bias, and the output results, it comes out a “core dump” problem:

```
522 int sc_main(int argc, char* argv[]) {
523     // Create signals
524     sc_signal<bool> reset, clock;
525     float input[3][224][224];
526     float weight_1st[64][3][11][11];
527     float output_1st[64][27][27];
528     float weight_2st[192][64][5][5];
529     float output_2st[192][13][13];
530     float weight_3st[384][192][3][3];
531     float output_3st[384][13][13];
532     //float weight_4st[256][384][3][3];------->core dumped!!
533     //float output_4st[256][13][13];
534
```

Then I start to deal with the weight arrays, trying not to identify the OUTPUT_CHANNEL index in the weight arrays, it means: take the weight value from .txt file only when the computations need it, so it becomes:

```
89      for (int oc = 0; oc < OUTPUT_CHANNELS; oc++) {
90          for (int oh = 0; oh < OUTPUT_HEIGHT; oh++) {
91              for (int ow = 0; ow < OUTPUT_WIDTH; ow++) {
92                  float sum = 0;
93
94                  for (int ic = 0; ic < INPUT_CHANNELS; ic++) {
95                      for (int kh = 0; kh < KERNEL_SIZE; kh++) {
96                          for (int kw = 0; kw < KERNEL_SIZE; kw++) {
97                              if(oh == 0 && ow == 0){
98                                  weight_file >> weight_1st[ic][kh][kw];
99                              }
```

other observations or insights gained:

I’ve learned some in creating modules in systemC. In SystemC, a module represents the smallest container of functionality with state, behavior, and structure for hierarchical connectivity. Understanding the concept and usage of modules in SystemC is fundamental for building complex hardware and software systems.

Modules in SystemC facilitate hierarchical design, allowing designers to break down complex systems into smaller, more manageable components. This modular approach promotes reusability, scalability, and maintainability of designs.

While port communication is the preferred mechanism for inter-module communication, SystemC allows other communication mechanisms for debugging or diagnostic purposes. For example, direct method calls between modules or accessing internal module variables may be used temporarily for debugging, but such practices should be limited to diagnostic purposes and avoided in production designs.

```
750     alexnet m_alexnet("alexnet");
751     Reset m_Reset( "m_Reset", 10 );
752     Clock m_clock( "m_clock", 5, 40 );
753     Monitor m_Monitor( "m_Monitor" );
754     m_Reset( reset );
755     m_clock( clock );
756     m_Monitor.rst( reset );
757     for(int i = 0; i < 1000; i++){
758         m_alexnet.output_final[i](output_final[i]);
759         m_Monitor.fc_8st[i](output_final[i]);
760     }
761
```