HOMEWORK #4
MLP Implementation By SystemC

Ali Alipour Fraydani
810101233
ali.alipour98@yahoo.com

Ali
Alipour
(Fraydani)

In this homework we try to implemented MLP by VHDL. in this project we design Datapath and Controller for every Neuron.

VHDL, Datapath, Controller, State Diagram

I. INTRODUCTION

In this homework first we want describe Dathapath and Controller of Single Neuron in next step connected them to hidden layer, in output compare them and identify classes of iris (Setosa, Versicolor, Virginica).

II. DATAPATH AND CONTROLLER

A. *Datapath*

Figure 2.1 show datapath of single neuron

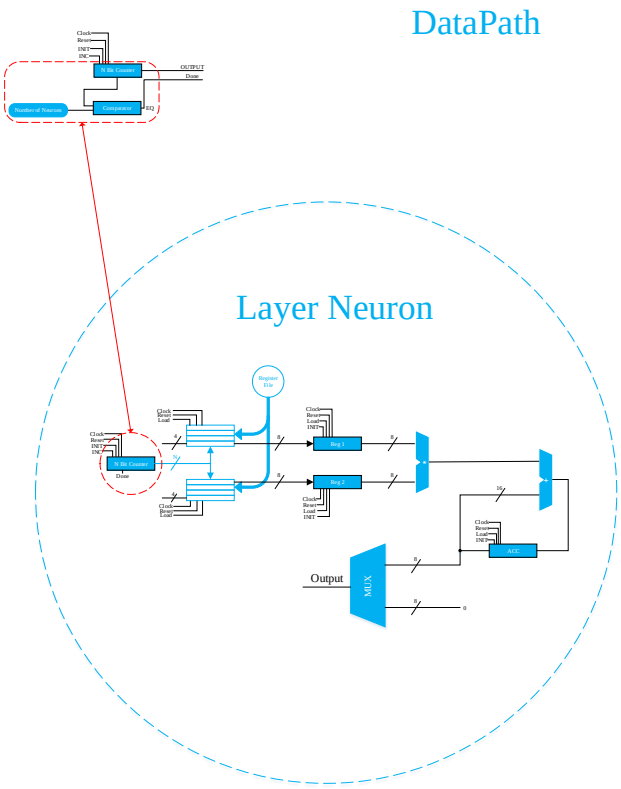

Figure II:I

Datapath has two register file for store inputs and weights, next one we have two register that we called Partial register that every data that need to process goes there next one data multiply and add with bias, first one bias initial in Accumulator and after than that add with before data. Output multiplexer describe ReLu function.

B. *Controller*

In this level describe a Controller Figure 2.2 shown the controller of single neuron.

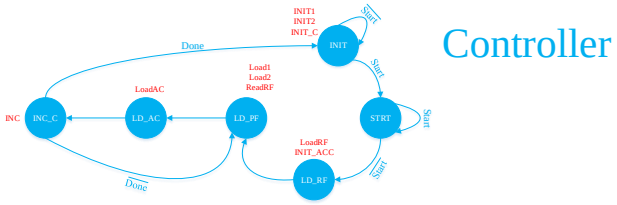

Figure II:II

Controller start with Start signal, this signal have to on and off for start control, next state load data to register files and Accumulator initial with bias data first one, after than load data to Partial registers and load to Accumulator after than, increment counter if neuron doing itself duty controller issued Done, else load new value to partial register.

In this section we describe the MLP by systemC, we describe modules in VHDL into c++ file that shown in Figure 2.3.

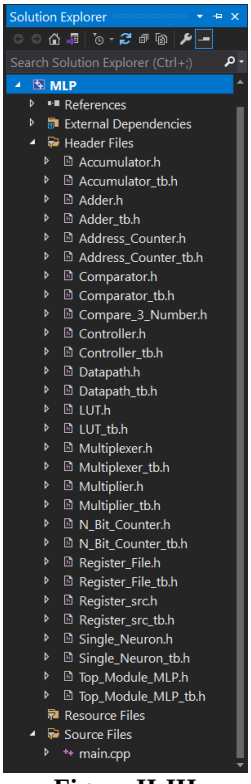

Figure II:III

This module explain in represent, that capturing from desktop. Single Neuron included from datapath & controller that shown in Figure 2.4.

```
Datapath<4, Comp_Value>* DP_Neuron;
Controller * CTRL_Neuron;

SC_CTOR(Single_Neuron)
{
    DP_Neuron = new Datapath<4, Comp_Value>("Dathpath");
    DP_Neuron->Clock(Clock);
    DP_Neuron->Reset(Reset);
    DP_Neuron->ReadRF(ReadRF);
    DP_Neuron->INC(INC);
    DP_Neuron->Load1(Load1);
    DP_Neuron->Load2(Load2);
    DP_Neuron->LoadRF(LoadRF);
    DP_Neuron->LoadACC(LoadACC);
    DP_Neuron->INIT1(INIT1);
    DP_Neuron->INIT2(INIT2);
    DP_Neuron->INIT_C(INIT_C);
    DP_Neuron->INIT_ACC(INIT_ACC);
    DP_Neuron->Done(Done_Int);
    DP_Neuron->INPUT_0(INPUT_0);
    DP_Neuron->INPUT_1(INPUT_1);
    DP_Neuron->INPUT_2(INPUT_2);
    DP_Neuron->INPUT_3(INPUT_3);
    DP_Neuron->INPUT_4(INPUT_4);
    DP_Neuron->INPUT_5(INPUT_5);
    DP_Neuron->INPUT_6(INPUT_6);
    DP_Neuron->INPUT_7(INPUT_7);
    DP_Neuron->INPUT_8(INPUT_8);
    DP_Neuron->INPUT_9(INPUT_9);
    DP_Neuron->Weight_0(Weight_0);
    DP_Neuron->Weight_1(Weight_1);
    DP_Neuron->Weight_2(Weight_2);
    DP_Neuron->Weight_3(Weight_3);
    DP_Neuron->Weight_4(Weight_4);
    DP_Neuron->Weight_5(Weight_5);
    DP_Neuron->Weight_6(Weight_6);
    DP_Neuron->Weight_7(Weight_7);
    DP_Neuron->Weight_8(Weight_8);
    DP_Neuron->Weight_9(Weight_9);
    DP_Neuron->Bias(Bias);
    DP_Neuron->OUTPUT_I(OUTPUT_I);
    DP_Neuron->OUTPUT_II(OUTPUT_II);

    CTRL_Neuron = new Controller("Controller");
    CTRL_Neuron->Clock(Clock);
    CTRL_Neuron->Reset(Reset);
    CTRL_Neuron->ReadRF(ReadRF);
    CTRL_Neuron->INC(INC);
    CTRL_Neuron->Load1(Load1);
    CTRL_Neuron->Load2(Load2);
    CTRL_Neuron->LoadRF(LoadRF);
    CTRL_Neuron->LoadACC(LoadACC);
    CTRL_Neuron->INIT1(INIT1);
    CTRL_Neuron->INIT2(INIT2);
    CTRL_Neuron->INIT_C(INIT_C);
    CTRL_Neuron->INIT_ACC(INIT_ACC);
    CTRL_Neuron->Start(Start);
    CTRL_Neuron->Done(Done_Int);

    SC_METHOD(assignment);
    sensitive << Done_Int;
}
```

Figure II:IV

Top_module_MLP include 21 Neuron, 10 Neuron for first layer, 8 Neuron for second layer and 3 Neuron for output layer, number of inputs in first layer is 4 that fully connected to Neurons of layer, therefore counter of this Neurons must be count until 4, as the same way, layer 2 count until 10 and third layer count until 3, by use of this information set parameter of Single Neurons in Top_module_mlp that first parameter shown number of bit from counter and second parameter shown maximum counting by counter. Figure 2.5 shown this.

```
Single_Neuron<4, 4>*    Neuron_I[10];
Single_Neuron<4, 10>*   Neuron_II[8];
Single_Neuron<4, 8>*    Neuron_III[3];
```

Figure II:V

We use for statement in c++ for describe Neuron of layer that shown in Figure 2.6.

```
// Layer I
for (int i = 0; i < 10; i++) { ... }

// Layer II
for (int j = 0; j < 8; j++) { ... }

// Layer III
for (int k = 0; k < 3; k++) { ... }

SC_METHOD(assignment);
sensitive << Done_III[2] << OUTPUT_II_I[0] << OUTPUT_II_II[0] << mem[40] << Done_I[0];

SC_METHOD(serialGeneration);
```

Figure II:VI

First layer shown in Figure 2.6 that shown how describe it by for statement.

```
// Layer I
for (int i = 0; i < 10; i++)
{
    Neuron_I[i] = new Single_Neuron<4, 4>(("Neuron I_" + std::to_string(i)).c_str());
    Neuron_I[i]->Clock(Clock);
    Neuron_I[i]->Reset(Reset);
    Neuron_I[i]->Start(Start);
    Neuron_I[i]->Done_OUT(Done_I[i]);
    Neuron_I[i]->INPUT_0(INPUT_0);
    Neuron_I[i]->INPUT_1(INPUT_1);
    Neuron_I[i]->INPUT_2(INPUT_2);
    Neuron_I[i]->INPUT_3(INPUT_3);
    Neuron_I[i]->INPUT_4(Zero);
    Neuron_I[i]->INPUT_5(Zero);
    Neuron_I[i]->INPUT_6(Zero);
    Neuron_I[i]->INPUT_7(Zero);
    Neuron_I[i]->INPUT_8(Zero);
    Neuron_I[i]->INPUT_9(Zero);
    Neuron_I[i]->Weight_0(mem[4 * i]);
    Neuron_I[i]->Weight_1(mem[4 * i + 1]);
    Neuron_I[i]->Weight_2(mem[4 * i + 2]);
    Neuron_I[i]->Weight_3(mem[4 * i + 3]);
    Neuron_I[i]->Weight_4(Zero);
    Neuron_I[i]->Weight_5(Zero);
    Neuron_I[i]->Weight_6(Zero);
    Neuron_I[i]->Weight_7(Zero);
    Neuron_I[i]->Weight_8(Zero);
    Neuron_I[i]->Weight_9(Zero);
    Neuron_I[i]->Bias(mem[i + 144]);
    Neuron_I[i]->OUTPUT_I(OUTPUT_I_I[i]);
    Neuron_I[i]->OUTPUT_II(OUTPUT_II_I[i]);
}
```

Figure II:VII

mem array take by file by format text from zero time that shown in Figure 2.8.

```
void MLP::serialGeneration()
{
    int temp;
    std::ifstream fp;
    fp.open("ROM_WB.txt");
    for (int i = 0; i < 165; i++)
    {
        fp >> temp;
        mem[i] = sc_lv<8>(temp);
    }

}
```

Figure II:VIII

serialGeneration has a SC_METHOD member function that run from zero time after running.

In test bench of top_module_mlp give test data from file that named by X_Test_Data.txt and apply from MLP and compare by targets that exist in Y_Test_Data.txt end of run show accuracy. Inputting member function shown in Figure 2.9.

Figure II:IX

Result of command line shown in Figure 2.10, accuracy get same of VHDL Homework.

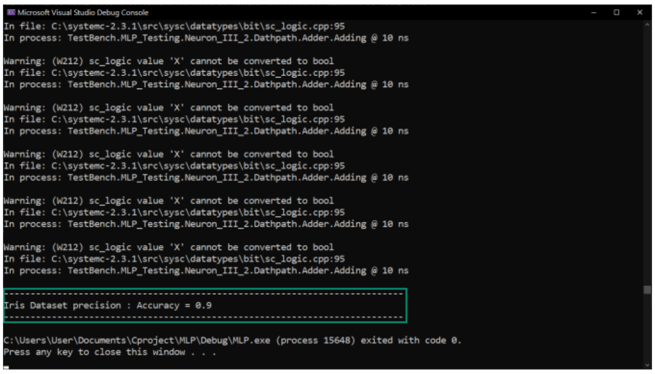

Figure II:X

Waveform of this Homework shown in Figure 2.11.

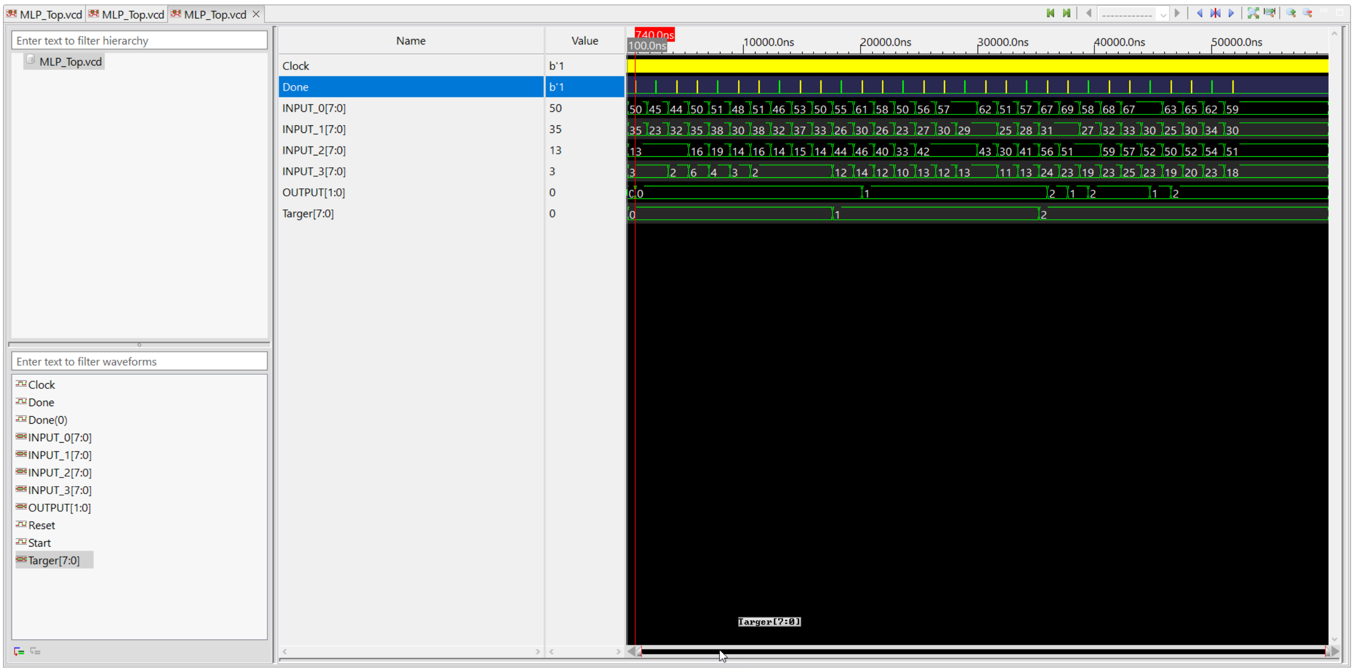

Figure II:XI