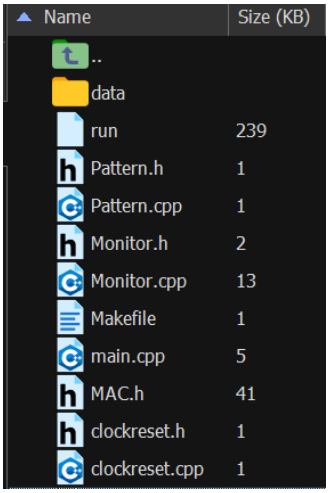
HW1 Implementation of AlexNet in SystemC

I. Code Structure

data folder	Include weight, bias, and input image matrix
run	Executable files for SystemC program
Pattern.h	Declare the internal variable, input/output signal , functions, method.
Pattern.cpp	Implement the methods that will provide our AlexNet input image. It will read the text file (image) from data folder when rst signal is on. After turning off rst signal, Pattern will output 2 image in 2 following continuous cycles.
Monitor.h	Declare the input/output signal, functions and method for monitoring our AlexNet. Also, the output layer (Softmax layer and sorting) is implemented here. After getting the last linear layer (fc8), it will trigger the output layer and show the inference result according to the Softmax result and class name list.
Monitor.cpp	Implement the method for monitoring our AlexNet. Also, the output layer (Softmax layer and sorting) is implemented here. After getting the last linear layer (fc8), it will trigger the output layer and show the inference result according to the Softmax result and class name list.
Makefile	Makefile script for compile systemC program.
main.cpp	Declare the main function, create the module instances, mapping the signals. It includes all operation units, pattern module, clockreset modules, and monitor module.
MAC.h	Implement all operation units.
clockreset.h	Declare the clock module and reset modules.
clockreset.cpp	Implement the clock module and reset modules.



II. Design Architecture and Implementation

A. AlexNet Inference

I separate AlexNet into 11 modules. The following sheet shows the module and their corresponding layer.

My AlexNet Module	AlexNet Layer Name
conv2d_0	Conv2d_1
	ReLU_1
MAX_POOLING_0	MaxPooling_1
conv2d_1	Conv2d_2
	ReLU_2
MAX_POOLING_1	MaxPooling_2
conv2d_2	Conv2d_3
	ReLU_3
conv2d_3	Conv2d_4
	ReLU_4
conv2d_4	Conv2d_5
	ReLU_5
MAX_POOLING_2	MaxPooling_3
No need to Implement	AdaptiveAvgPool2d
No need to Implement	Dropout_1
LINEAR0_RELU	Linear_1
	ReLU_6
No need to Implement	Dropout_2
LINEAR1_RELU	Linear_2
	ReLU_7
LINEAR_2	Linear_3

In all of conv2d and linear module, there exists both in_valid and out_valid. The former represents the input signal for this block is valid for now; the later means the output result is valid. This design make my AlexNet inference become a pipelined architecture.

Generally, all of ReLU layer are combined with their last layer because it is just filter the negative result and replace with zero element.

Since both the input feature map and output feature map of AdaptiveAvgPool2d have the same shape, it's no need to implement that layer. Also, dropout is a trick in training stage. In inference stage, we can just skip this kind of layer.

B. Output Layer

Because it is not include in the model architecture shown in problem pdf. I implement it in the monitor. When the monitor get the out_valid signal from LINEAR_2, the last layer, it will apply softmax to the result from LINEAR_2 sort it in descending order and list the top-5 result.

C. Pattern

When rst signal is high, it will read two images (dog.txt, cat.txt) from data folder. Then, pattern will output these two image in **sc_vector<sc_out<sc_fixed_fast<46,17>>> image** dataformat in the following 2 cycle.

When it outputs images, it will also pull the in_valid_pat signal high, which represents the current image signal is valid.

III. Observations and Optimization

A. sc_fixed_fast

Originally, I use sc_fixed, but it run very slowly. After searching some information in the forum, I found a sc_dt data type called sc_fixed_fast, which implement the fixed point based on C++ double data type. Thus, it will simulate faster and pass the signal to next module quicker.

To the internal calculate in the module, I implement the calculation in C++ double and vector<double>. That will also speed up the operation in the module.

B. Bitwidth

I set all of sc_fixed_fast data type in bit width <46,17>, which means a fixed number will use 46 bits to represent and 17 bits will represent its integer part. Because pyTorch usually use 16 floating point to run the calculation, which is roughly equal to sc_fixed_fast<40,17>. But I still add more bits for higher precision.

C. pipelined architecture

I use in_valid and out_valid signal to make AlexNet become a piplined architecture. Thus, it will just enable the needed part. Unused part will remain the former result and lower the out_valid. This will also decrease the calculation for simulation,

and speed up the simulation.

D. Optimization for compile

```
all:
	g++ -I . -I  $(INC_DIR) -L . -L $(LIB_DIR) -o $(O) $(C) $(LIB) $(RPATH) -O3
	./run
```

Because systemC is a C++ library, it still use g++ to compile. Therefore, I try to use -O3 optimization. As I expected, it run faster than before.
Before I use this, it takes almost 10 minutes to run 1 image. After applying this, I only use 40 seconds to process 2 images.

IV. Demo Results

dog.txt

```
****************************************************************************************************
-                        Top-5 Results
****************************************************************************************************
Index          Val              Possibility            ClassName
207     16.594305               38.626756%             golden retriever
175     15.569432               13.860900%             otterhound
220     15.361640               11.260262%             Sussex spaniel
163     15.002478               7.862605%              bloodhound
219     14.593001               5.220751%              cocker spaniel
====================================================================================================
```

cat.txt

```
****************************************************************************************************
-                        Top-5 Results
****************************************************************************************************
Index          Val              Possibility            ClassName
285     20.206511               96.381130%             Egyptian cat
281     16.136685               1.646226%              tabby
282     15.733710               1.100220%              tiger cat
287     14.790746               0.428505%              lynx
728     14.411773               0.293339%              plastic bag
```