IL2230 Lab 3A

Jinghan Yang (yjinghan@kth.se)
Ruolan Wang (ruolanw@kth.se)
Hui Feng (huif@kth.se)
Xinyi Zhang (xinyz@kth.se)

December 12th 2023

1 Instruction to Design

1.1 FSM design

- As for the design of finite state machine, we set four states in total. The first one is 'idle', the system will initialize the register and counter, then waiting for the next clock cycle. The second one is 'calculate', the system will start to calculate until repeat for K layers.The third one is 'shift-data', the system will shift the output of this layer to the input data register. The output of last layer will become the input of this layer. The final one is 'finish', after finishing all the layer and get the final output, the system has finished its mission and send the output.
- The most important thing in this session is the state shifting. We use three counter as the signal of state shifting. Assuming there is a MLP that has K layers with N neurons per layer, and every neuron has N input.
- We set MAC counter, neuron counter and layer counter to record the process of the calculation. When MAC counter is equal to N-1, the neuron counter add one. That means one neuron finish its calculation. Then, when neuron counter is equal to N-1, the layer counter add one and set neuron counter equal to 0. That means one layer has finished calculation. Then the state of system is shifted to data shift. If the layer counter is less than K-2 (input layer doesn't need to be counted), then the state turns back to 'calculate' state, or the state will turns to 'finish' state and give the final result.

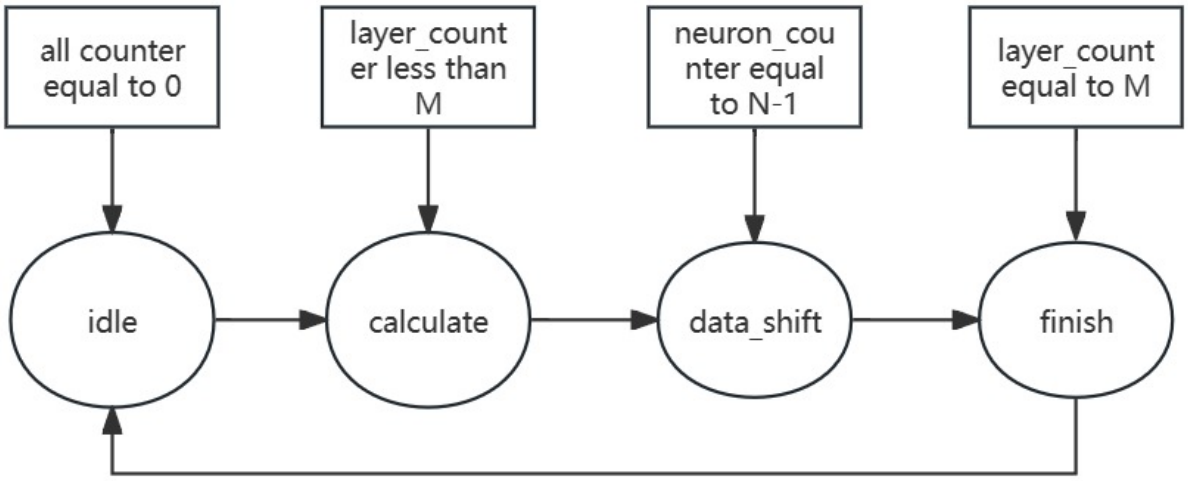


Figure 1: FSM design

1.2 One neuron design - task1

- In this task, we are asked to design a one-neuron design MLP. With what we have down in Lab2, we decide to use the module that we designed in Lab2. The module is actually a N-input neuron, so we just

need to call the neuron for N times, so that we can get a layer with N neurons. What's more, we store the output of the neuron in a register. In this way, the register can become the memory for the input of next layer.

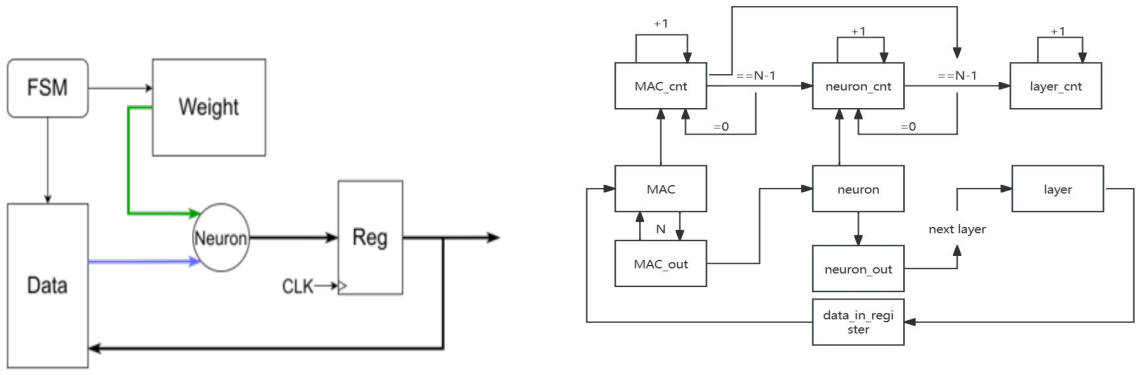


Figure 2: One neuron design

1.3 N neuron design - task2

- In this task, we are asked to design a N-neuron design MLP. The difference between task2 and task1 is the parallel design inside the layer. N neurons can handle the calculation mission at the same time. So this design just need one clock cycle to finish one layer, and shift to the next layer.
- What's more, because of the increased efficiency, the calculation in every layer just need one clock cycle. The MAC counter and neuron counter are always set as N-1, so that only if layer counter is equal to K-2, the layer switching will stop.

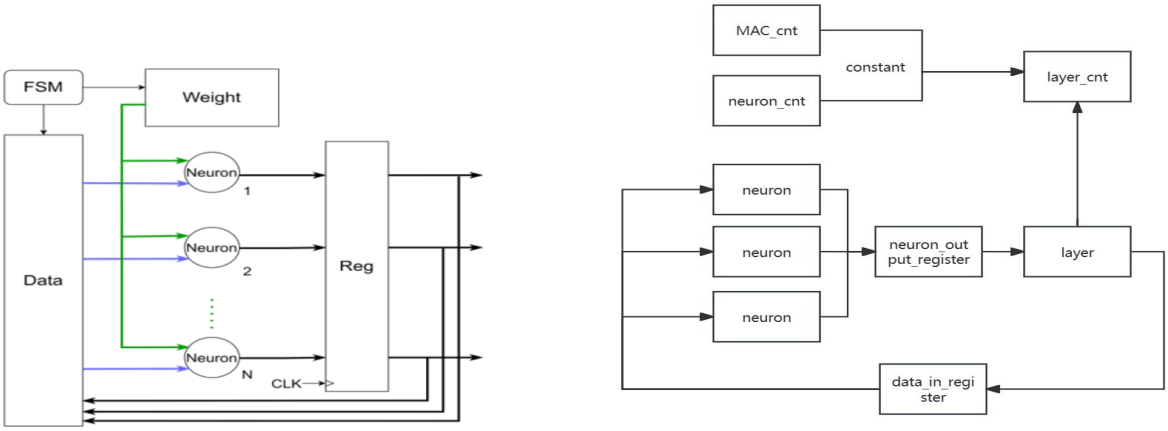


Figure 3: N neuron design

2 Questions

2.1 Q1

- Discuss the scalability of your designs for the given choices of non-linear function and data precision.Report your synthesis results of area, power, and frequency. What conclusions can be drawn from the comparative and scalability evaluations? For example, the performance of design for small/large values of N and M, one neuron/N-neuron, etc.
  - As for the results that we have got, we can say that it's clear that the utilization and the power of the system is rising when the number of layer and neuron in per layer increase.
    But here is a problem when the number of layer and neuron increase to 3 and 8. The slack of setup check is negative. We think that's because the increasing computation content leads to more

computation time are needed during the calculation state. So when the clock signal comes to the next positive edge, the calculation are not totally finished.
We can solve it by simplifying the process of calculation. Using pipeline structure instead of serial structure or extending the time period of the clock.

2.2 Q2

- Discuss on how your memory subsystem (for storing weights, activations) impacts your designs? For example, what if only half or 25% of the weights can be loaded in one cycle for each layer computation?
  - As the chart about data flow we showed above, you can see there is a register for the input data for every layer. There is also a register for input weight, which should be a third dimensional array. But in system verilog, we can't define a array like that, so we extend the content of a second dimensional array, and select the proper input according to which layer we are processing now. If there is something wrong with weight loading, we can add a weight_load signal to check if the weights are selected properly. The calculation state will start only if the weights are totally input.

2.3 Q3

- In your N-neuron design, you have considered to parallelize the computations within each layer. This is a natural layer-after-layer processing, meaning that the next layer computation starts only after all the previous layer computation is complete. Is it possible to realize overlapped layer processing? This means that computation for the next layer m is performed simultaneously with computation for the previous layer m-1. If this is possible, discuss how? If not, why?
  - In our design, we implemented the structure in task2 through calling the layer module for M times and updating the input when the calculation is finished. So, we can finish a M layer design finally. If we want to implement overlapping layer processing, we need to design a pipeline structure. And generate M layers instead of just calling the same layer with different input. What's more, every layer will be in different calculating stage. For example, when the first layer is calculating for M times, the second layer is calculating for M-1 times, so that we can make the data flow between different layers.

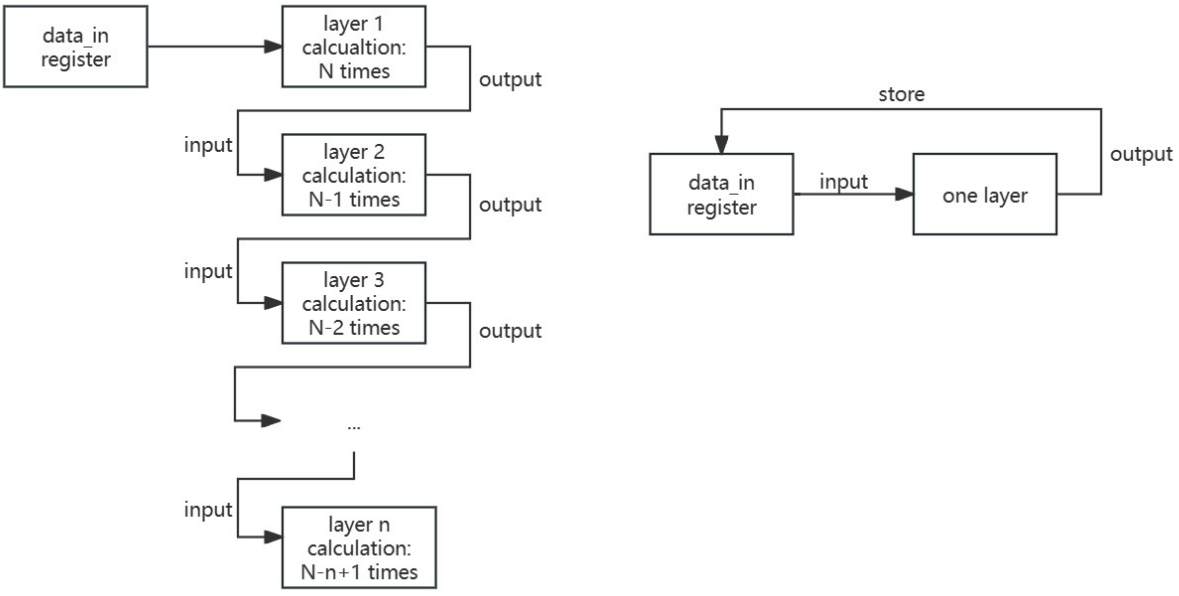
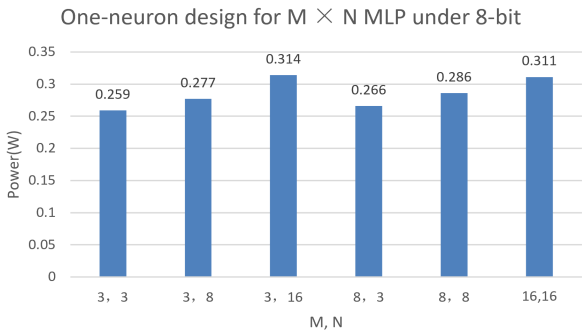


Figure 4: parallel design and original design

3 Results

3.1 One neuron design

Figures below show the Area and power consumption of One neuron design under 8bits. From the figure, we can see that the more neurons in each layer, the more power and area are consumed.



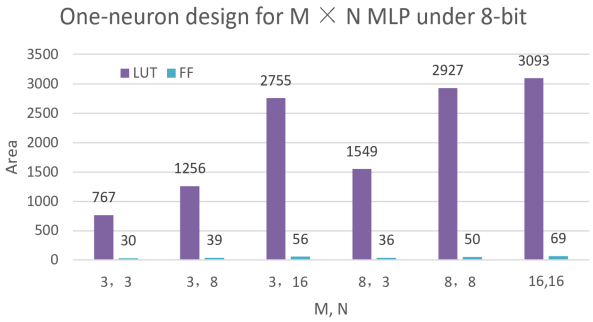


Figure 5: Area and power of One neuron design

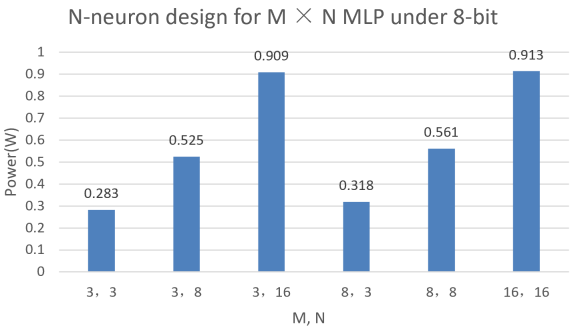
The table below shows the slack of One neuron design under 8bits. From the figure, we can see that Setup time becomes more negative when number of layers and neurons in each layer increase. That's because more layers and more neurons lead to more computation time.

Table 1: One neuron design's Time slacks for M × N MLP under 8-bit.

M,N	Setup[ns]	Hold [ns]
3,3	0.919	0.216
3,8	-0.795	0.271
3,16	-8.498	0.165
8,3	0.478	0.206
8,8	-1.581	0.19
16,16	-8.504	0.186

3.2 N neuron design

Figures below show the Area and power consumption of N neuron design under 8bits. From the figure, we can see that the more neurons in each layer, more layers, the more power and area are consumed.



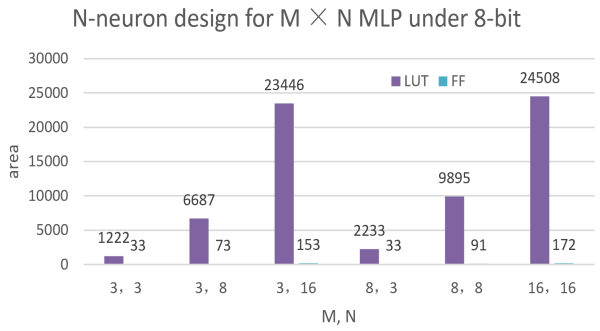


Figure 6: Area and power of N neuron design

The table below shows the slack of N neuron design under 8bits. From the figure, we can see that Setup time becomes more negative when number of layers and neurons in each layer increase. That's because more layers and more neurons lead to more computation time.

Table 2: N neuron design's Time slacks for M × N MLP under 8-bit.

M,N	Setup[ns]	Hold [ns]
3,3	0.626	0.147
3,8	-0.373	0.16
3,16	-6.637	0.148
8,3	0.499	0.171
8,8	-1.889	0.21
16,16	-5.7	0.177

3.3 comparison

The figure below shows the comparison of One neuron design and N neuron design. From the figure, we can see that parallelism consumes more resources (area, power) than serial.

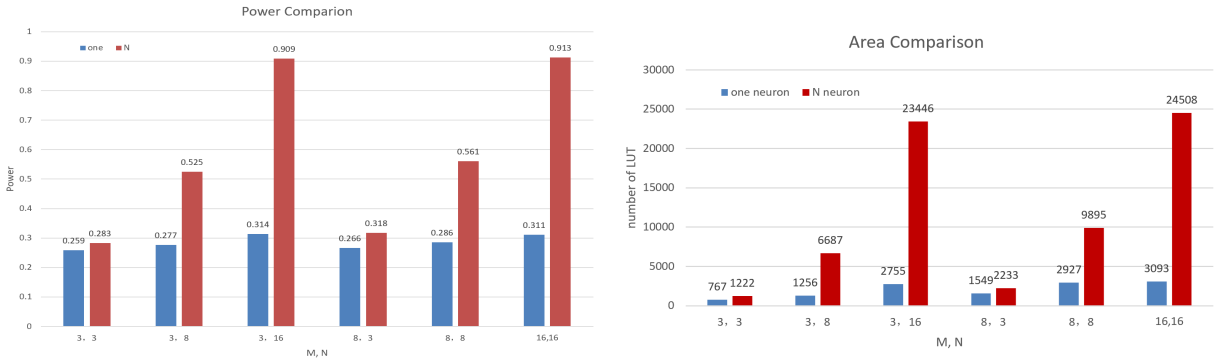


Figure 7: Power and area comparison of two design.