Design, Implementation and Evaluation of MLPs in Hardware

Peng Zhang, Sarah Saleem, Marko Savić, David Ramón Alamán

January 15, 2024

Abstract

In this project a multi-layer NN with multiple parallel-MAC neurons per layer was implemented in hardware (SystemVerilog). A single code simulates both a one-neuron and an N-neuron MLP design by changing the parameters within.

1 One-neuron MLP design

The design is has been created as and FSMD (Finite State Machine and Datapath) and it is composed of three modules. The first one, the datapath implements the logic related with the operation of the data, including the neurons. It has been developed using parameters so that it can be generic. The second module, also related to the datapath is a memory module where the different weights are stored. The weights needed in each step of the operation are selected by the FSM which generates the corresponding addresses. Also, the FSM module is in charge of controlling the datapath through implementing the state machine shown in Figure 1.

The state machine has 4 states:

- `IDLE`: Is the default state of the design, it waits for the `init` signal to perform a transition to the
- `FETCH`: The `FETCH` state enables the input register so that the input data can be loaded into the design. And transitions to the `OP` state.
- `OP`: In this state, the addresses for the selection of the weights for each operation. Also, this state generates a pulse in the `start` signal to indicate to the neuron when to start a new operation.
- `COMPLETE`: Whenever all the operations have been perform the FSM performs a transition to the `COMPLETE` state, in which the signal `ready` is asserted to indicate that the final result of the MLP is available.

Figures 2 to 5 show the schematics obtained for the design.

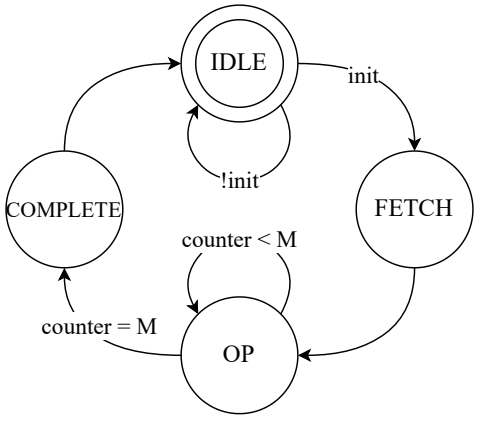

Figure 1: One-neuron MLP FSM diagram

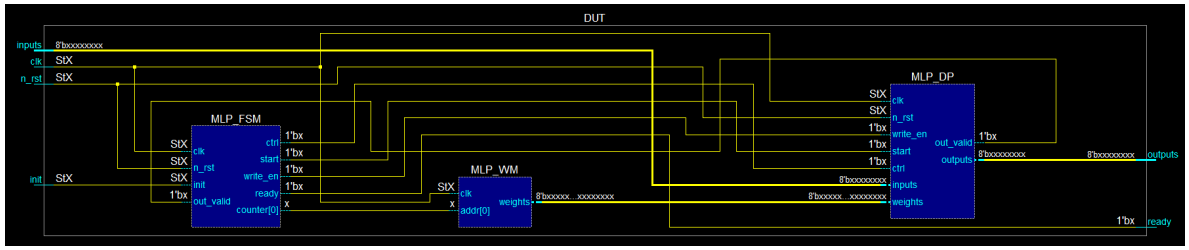

Figure 2: One-neuron MLP top module schematic

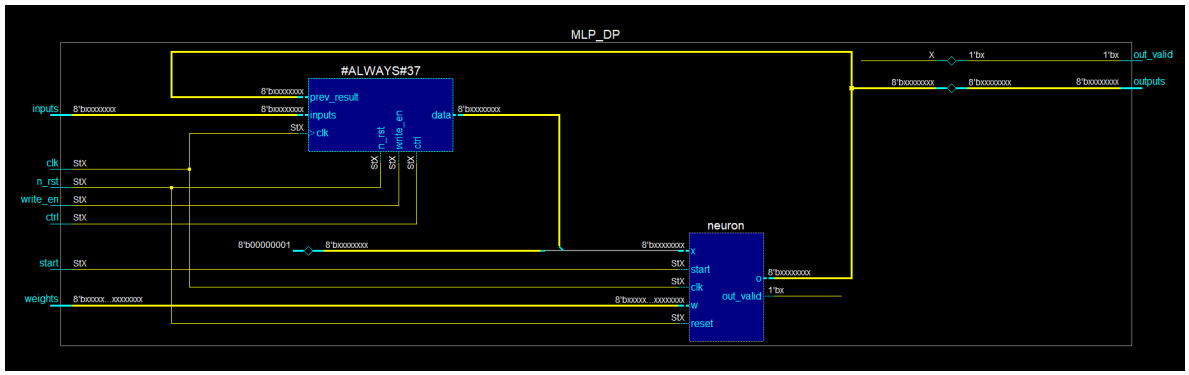

Figure 3: One-neuron MLP datapath schematic

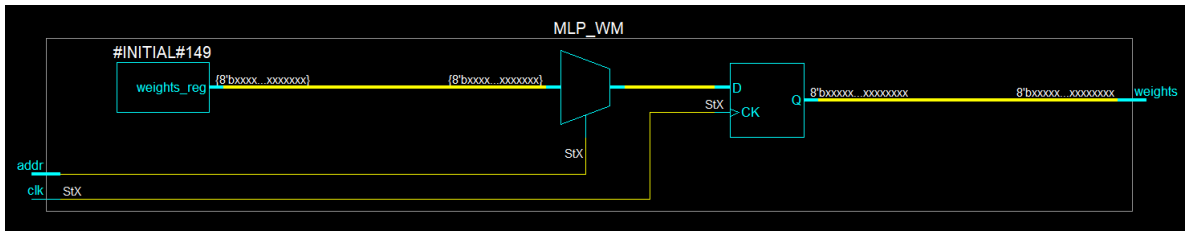

Figure 4: One-neuron MLP weight memory schematic

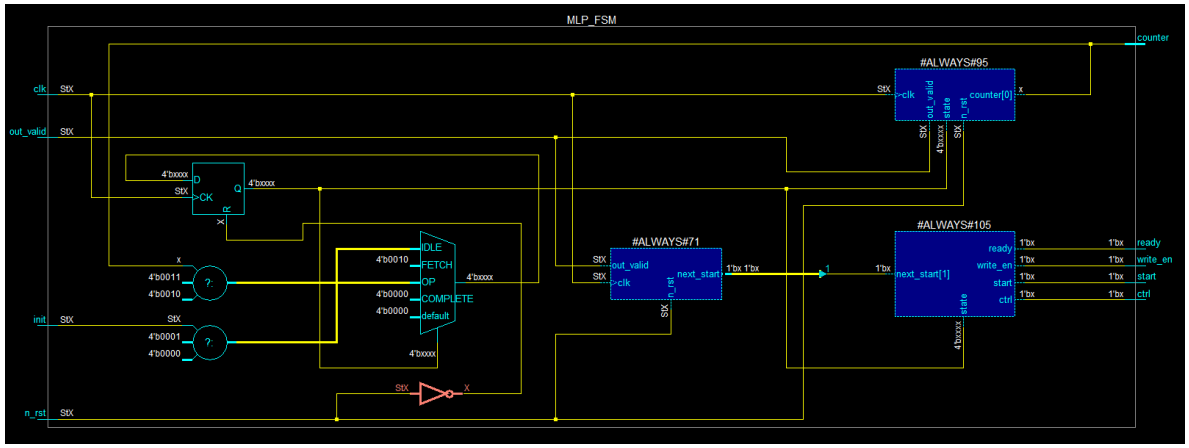

Figure 5: One-neuron MLP FSM schematic

2 N-neuron MLP design

This design allows to reuse the neurons inside a layer to implement an MLP with a semiparallel implementation. The code has been design to be generic by being able to configure all the connections just by modifying the parameters `N` and `M`.

In this implementation, the inputs are loaded into a register to be forwarded into the neuron along with the weights. After the operation, the results are stored into this register again. The model employs the same state machine as in the previous section (Figure 1).

Figures 6 to 9 show the schematics obtained for the design.

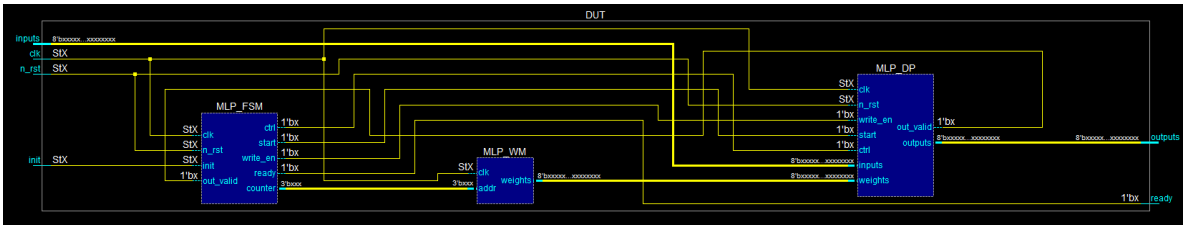

Figure 6: N-neuron MLP top module schematic

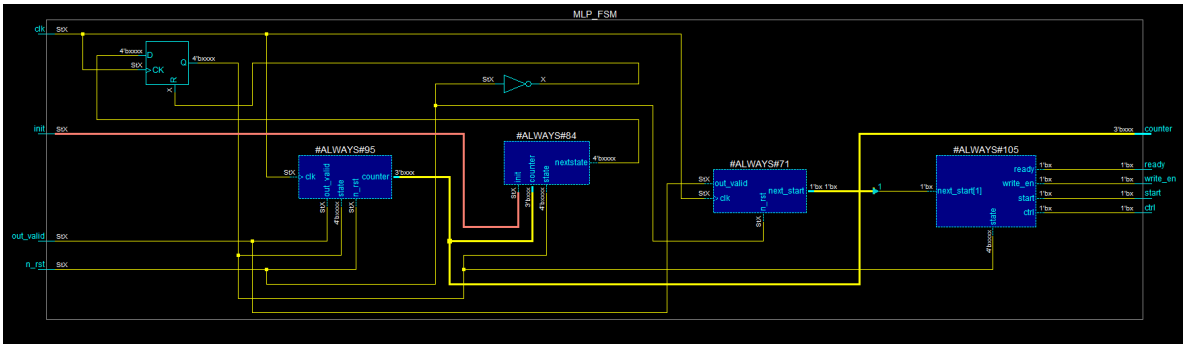

Figure 7: N-neuron MLP FSM schematic

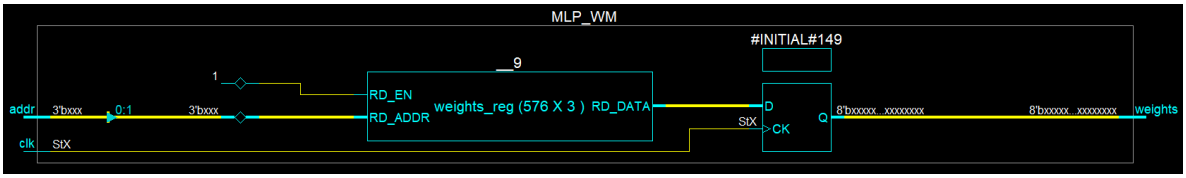

Figure 8: N-neuron MLP weight memory schematic

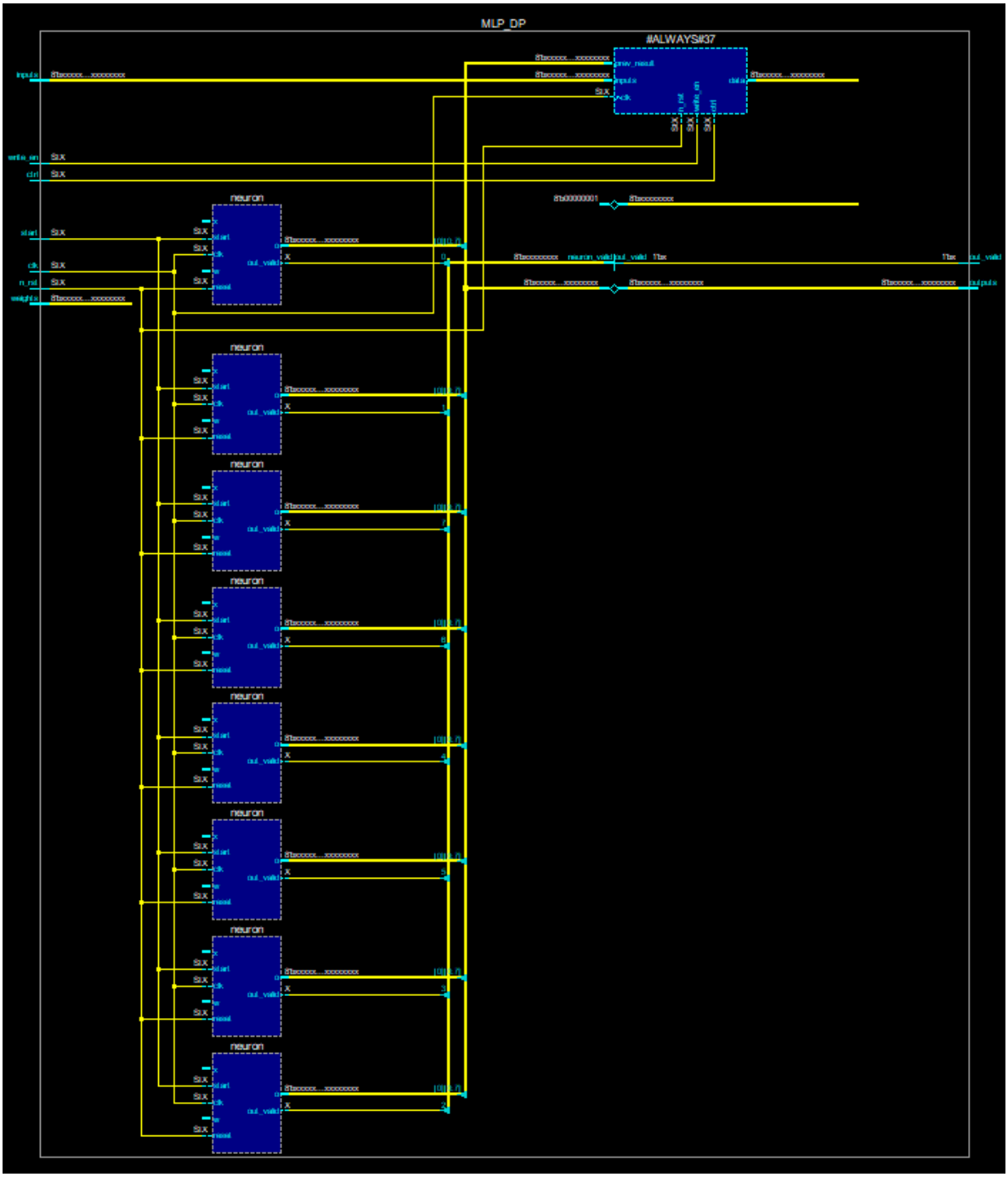

Figure 9: N-neuron MLP datapath schematic

3 Results

Area, power, and frequency for: (number of layers, number of neurons) = (3, 3), (3, 8), (3, 16) and (8, 3), (8, 8), (16, 16)

M number of layers	3	3	3	8	8	16
N number of neurons	3	8	16	3	8	16
datawidth	8	8	8	8	8	8
area [number of units]	785	5828	22595	866	6266	24214
power [W]	0.133	0.15	0.212	0.133	0.142	0.221
minimum period [ns]	3.528	6.837	7.43	3.656	7.044	8.945

Figure 10: Results

3.1 Discussion 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.

We used 8 bit as our data precision and Relu as our acitiviation function, from Figure 7, we can see that more number of layers and more number of neurons lead to more area and power, at the same time the frequency it needed to match the timing become larger.

The more the neurons there are, the more the links and the larger the area. The total number of links with the number of neurons N and number of layers is $(M-1)\cdot N^2$. This is exponential growth of links with more neurons. The area is estimated to grow exponentially in the worst case, but in reality is likely linear growth.

3.2 Discussion 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?

memory which used to store weights increases when number of layers and neurons increases, if only half or 25% of weights can be stored in one clock cycle, there will need more clock cycles to store the whole weights and the layers and neurons increase, which will affect the performance of the network.

3.3 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?

Overlapped layer processing in an N-neuron design, where computations for layer m and layer m-1 occur simultaneously, is theoretically possible but presents significant challenges. This requires a sophisticated approach to pipelining, where multiplication, accumulation, and data fetching are managed in a way that allows for concurrent processing across layers. Implementing such a system would need careful coordination to ensure that the data dependencies between layers are properly

managed and that the output of one layer is available as soon as needed by the next. The feasibility and efficiency of this approach would depend on the specific architecture of the MLP and the hardware capabilities available. It could lead to faster computation times but would increase the complexity of the hardware design and the potential for synchronization issues.

3.4 Example of operation (N, M) = (2, 2)

The model has been simulated using Questa. In the example, a MLP has been implemented with 2 neurons per layer and 2 inner layers. As shown in Figure 11. The results can be compared in Figures 12 & 13.

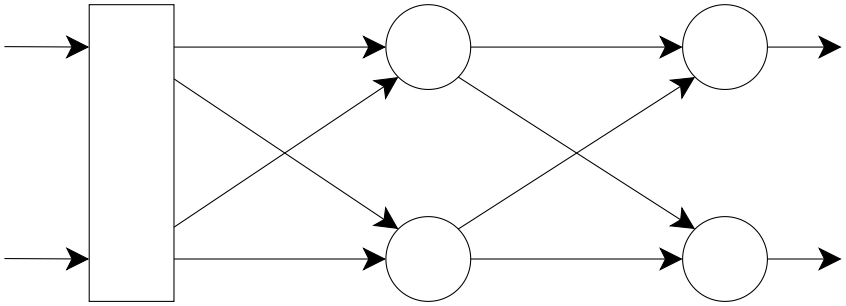

Figure 11: MLP model

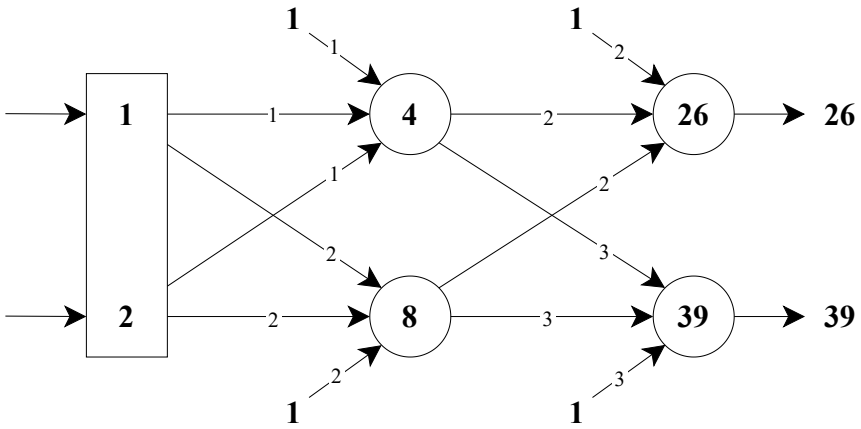

Figure 12: MLP with values

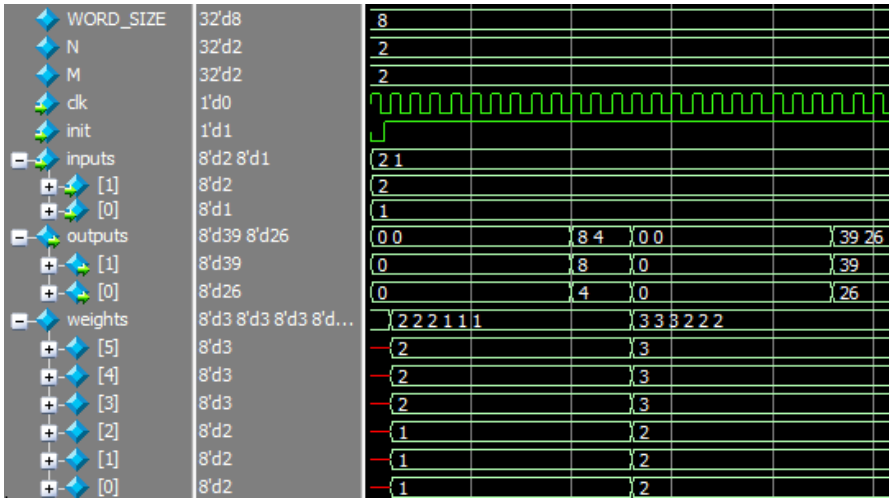

Figure 13: Waveforms