**Neural Network Calculator (nncalc)**

**A project for EE272**

**Phase 1 Version 0.0000**

Your first assignment is to design a neural network calculator using synthesizable system verilog. This is the portion of a neural network used for computing outputs from inputs, not the part used to find the weights and train the network. A neural network consists of a number of inputs, with each input connected to a ‘neuron’ with a multiplying weight. The neuron performs math on the weighted inputs, and then computes an output. This project builds a simple neural network. It consists of layers, and the neuron is approximated by a third order piece wise linear interpolation. This allows more than simple linear output results. The math operator is limited to summing the weighted values. There is no voting, majority, or top n type neuron operations. (These are hard to calculate in back propagation anyways, so this works for many networks) The design allows for up to 1024 inputs to a neuron, and up to 1024 neuron outputs. Each neuron can take a weighted value from each of the inputs, and produce an output. In many problems, not all inputs need be connected to each neuron. These types of networks are often called convolutional neural networks. These are most commonly associated with image processing where a region of the image results in a value. These regional values are passed to further layers for processing.

This design allows for 1024 inputs, and 1024 outputs. Commonly, there is another input of ‘1.0’ used to provide a bias or intercept to the results. Thus there may only be 1023 inputs per layer. The architecture allows for more complex layers by using two 1023 input layers on one physical layer at the expense of the number of layers processed and the number of inputs to each neuron.

The design consists of several memories. The first memory is a configuration memory. This memory is used by the engine to decide how to process the inputs. The second is a data memory used to hold the data inputs, outputs, and intermediate terms. The data memory is 64K 24 bit values.

The configuration memory limits the neural network complexity. There are 128K words of configuration memory, and the number of storage locations per neuron is approximated by:

Some sample calculations on configuration storage show that 1024 neurons are not practical with the current system, but 20 layers/Levels with 100 neurons each will fit in the system.



The configuration memory is 32 bits wide, and contains 217 locations. Within the configuration memory is the configuration block at a location pointed to by a hardware register. The configuration block is defined as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Word** | **Bits** | **Name** | **Function** |
| 0 | 11 | Clayers | The number of neuron layers |
| 0 | 17 | ClayerLoc | The table of neuron layers (see below) |
| 0 | 4 | Reserved |  |

The neuron layer table indicates how many neurons are in the layer. It points to a linear list of neuron blocks. The format is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Word** | **Bits** | **Name** | **Function** |
| 0 | 11 | Nneurons | The number of neurons on this layer |
| 0 | 17 | NtableLoc | The table of neurons on this layer (see below) |
| 0 | 4 | Reserved | Future expansion |

A neuron table configuration consists of 4 adjacent 32 bit little endian words in the configuration memory. This results in 128 bits allocated as follows:

| **Bits** | **Function** | **Comment** |
| --- | --- | --- |
| 10 | Ninputs | The number of inputs for this neuron. |
| 17 | Lbase | The base offset for inputs |
| 17 | Oloc | The result location |
| 17 | Nimap | The base for neuron input specifications |
| 17 | Wimap | The base for neuron weights |
| 17 | NeuronTable | The table for the second order interpolation |
| *5* | NeuronShift | Amount to shift the neuron value right for calculation |
| 5 | PostShift | Amount to shift the neuron table result left after calculation |
| 23 | Flags | Only bits 1 and 0 used  The low order bit (0) indicates calculation is complete after this neuron.  Bit 1 indicates that this is the end of a layer, and multiple units should wait until this has completed to begin the next layer calculations.  All other bits are reserved |
| 128 | Total bits |  |

The Nimap is a list of neuron locations. Each location is 10 bits, and there are 3 locations per map word. The location value is added to the Lbase to become the location in data memory for the neuron input. The upper two bits are reserved, and currently not used.

Each neuron can have up to 1024 inputs. Each input address is calculated by taking the 10 bit value in the Nimap, and adding that to Lbase. You are expected to process 8 inputs each clock. You will have to read the 8 ports from the configuration memory, fetch from the data memory, and multiply by the weights. The Ninputs tells you how many to process each clock.

The weights are 16 bit 2s complement signed values stored as 4.12. They are stored 2 per word little endian style. You will need to process 8 per clock.

A brief configuration bandwidth is:

8 inputs = 2.33 reads for the addresses

8 weights= 4 reads for the weights

8 Total = 6.33 reads per clock.

The memory can perform 8 reads per clock cycle. This allows reading 3 locations for addresses, and 4 locations for weights every clock cycle. The location data will require some shifting and/or aligning as reading 3 locations fetches 9 addresses, and you only need 8. Don’t re-fetch a location again as the 8th port is needed to fetch the next neuron information, and for table look ups on the neuron result.

Every 3 clocks, only two reads ports are required. The third one should be available for other reads as needed by other functions such as output interpolation. You will need an arbitrator on at least two of the read ports.

Consider a sequence as follows:

Residual Fetched Total in system

0 3\*3 = 9 9

1 3\*3 = 9 10

2 2\*3 = 6 8

0 3\*3 = 9 9

.

.

.

All internal calculations result in 24 bits. This is 8.16 format. This will require rounding. Round away from zero. The weights are 4.12. An 8.16 value multiplied by a 4.12 value results in 12.28 This should be summed to 20.28, or 48 bits.

The neural calculation first shifts the data right arithmetically (sign copied) by the amount specified for this neuron. if the shifted value is positive then it is saturated to 4.28 bits (if positive, then 7fffffe, if negative, then it is saturated to -7fffffe. The results of the saturation are then applied to the lookup table. The bits representing 4.4 (In the middle of the 32 bits) are used to find values in the table. The last half of the table are negative numbers. There are 3 words in the lookup table, W0, W1, W2, W3. They are stored as 2s complement 4.20 values. The remaining fraction from the shifted result is used (be careful with negative numbers) to perform the final value calculation. You will need to compute fraction squared and fraction cubed.

R=W0+f\*W1+f2\*W2+f3\*W3

R is computed to 16.48 bits, post shifted left, and then saturated to 24 bits for the final value. The binary point of the 8.16 is from the 16.48 .

Note that the neural calculation never writes to the configuration memory.

You will probably need a 2 flag model between the neuron weighted summing, and the interpolated output modules.

The data memory supports 8 reads and one write each clock. The data memory is typically written at the end of a neuron computation, and happens infrequently.

The first assignment will have one neural calculator. Subsequent assignments will add additional calculators and memories, and place nncalc on various busses.

The module has a simple register model:

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Name** | **Function** | **Other** |
| 0 | Config | The address of the configuration block in configuration memory. | Only the lower 17 bits used for address |
| 1 | CalcTime | The time to calculate the last nn result. Updated at the end of the nn calculation |  |
| 2 | Start | Any write to this data of 32’h00000ACE starts a neural calculation. | A read always returns the status as all 1’s for busy, and all 0’s for idle. |
| 0x20000 | Config | The configuration memory | A memory mapped register. There are 217 locations |
| 0x40000 | DataMem | The data memory | A memory mapped register. There are 216 24 bit values. The upper 8 should be sign extended on read. |

The block has a simple interface:

| **Name** | **Decl** | **Dir** | **Comment** |
| --- | --- | --- | --- |
| clk | Reg | In | A rising edge clock |
| reset | Reg | In | An active high reset |
| RW | Reg | In | Indicates if a read or write is taking place |
| sel | Reg | In | Indicates the block is selected for Read or Write |
| addr | [18:0] Reg | In | The address to read or write (19 bits) |
| din | [31:0] reg | In | Data into the block for register writes |
| dout | [31:0] reg | Out | Data out of the block for register reads |
| bus\_stop | reg | out | The data set is not taken this cycle. The read/write will need to be continued another cycle. |
| pushout | reg | out | The result is ready. This signal need only be high for one clock cycle. This can alert the CPU, or bus master circuity to start moving the output from the data memory. |
| dm | memIntf.mem | Both | The interface to a 9 ported data memory (1W/8R) |
| cm | cmemIntf.mem | Both | The interface to a 9 ported configuration memory |

In subsequent phases, (HW assignments), you will get more memory interfaces, and the engine will grow more complex.