MANNIX

memory management

2020

Author: Simhi Gerner

Date: December 30, 2020

Document Information

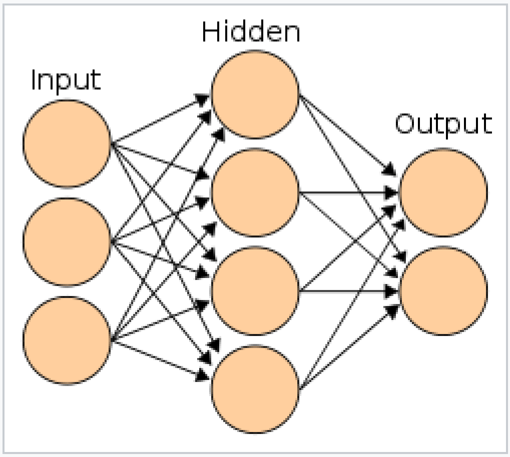
Document Title:	File Information:
MANNIX	**File Name:** Simhi_Gerner_preparatory_report **Last time saved:** December 31, 2020 **Saved by:** Simhi_Gerner
Keywords:	
Abstract:	
This document is the Specification of the memory management of the accelerators	

Contents

Introduction To Neural Network:

Neural Networks

A Neural Network is a mathematical model that developed in inspiration of process of biological neural network and used in machine learning. Network from this kind contain some of information units (i/o) that connect one to other, connection that sometimes pass through hidden layer. The form of the connection contain information on the connection strength, that imitate the connection form in the brain.



תרשים המדגים את אופן הפעולה של רשת עצבית מלאכותית.

Building neural network

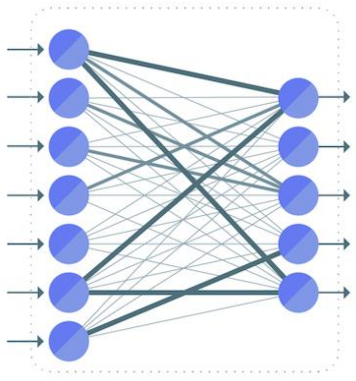
There is two stages:

1. Training-
   Each layer (fully connected and convolution) in neural network contain nodes and weights. For to determine the correct weights we need to enter inputs to the system and change the weights according to the write answer and the outputs of the network.
2. Inference -
   Once the Training complete we can enter inputs that propagate through the layers of the network and outputs the desired outcomes.

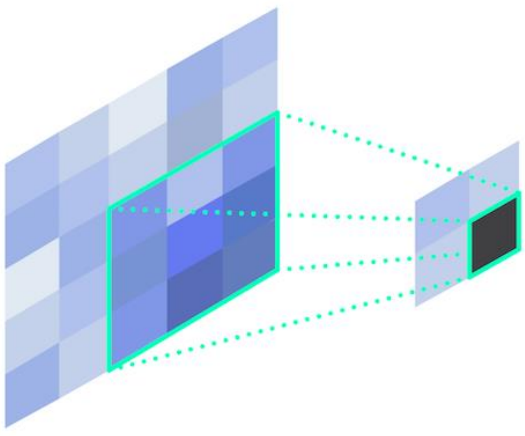
In this projects we focus on implementation of inference stage in hardware accelerators.

Four types of layers:

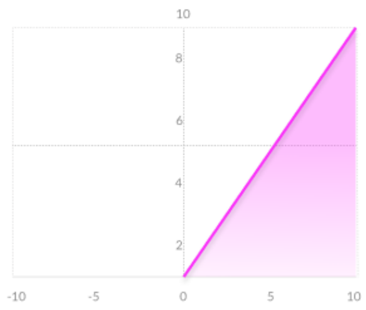
1. Fully connected layers connect every neuron in one layer to every neuron in the next layer. Fully connected layers are found in all different types of neural networks.



2. Convolution Layer is an important type of layer in a CNN. Its most common use is for detecting features in images, in which it uses a filter to scan an image, a few pixels at a time, and outputs a feature map that classifies each feature found.

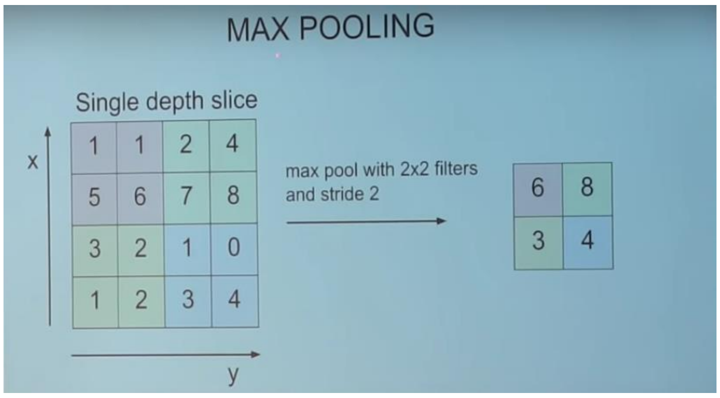


3. Activation function is mathematical equation that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated ("fired") or not, based on whether each neuron's input is relevant for the model's prediction. For simplify we use in ReLU function



This feature is implemented using predictive modeling – The AI start to process the information given to him by "guessing the result" (the computer only controls the weighs values) and by comparing the outcome of his calculation to the correct result provided by the user – the computer learns and adapt in order to be more precise in the next calculation.

4. pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently. The most common approach used in pooling is max pooling.



There is more types of layers, but in these four we use for our hardware accelerators.
There is a lot of information that enter and get out from each layers,
So, for do it effective in hardware implementation we need efficient memory management.

Memory management

We rely on the fact that the information in the memory is continuous. For example, all the information that fully connected layer need stored in continuous addresses.

Let's define: client = port that belong to one of the layer, that can can read/write from/to the memory.

So, instead of using in one big memory that only one client can perform read/write simultaneous.

We "break" the memory to smaller pieces, that every piece can perform read/write.

For one operation of inference each layers need a lot of cycles of getting information from the memory. We want also to work in pipeline mode which each layer busy all the time, and each client want to read form the memory, and we have some requests in simultaneous.

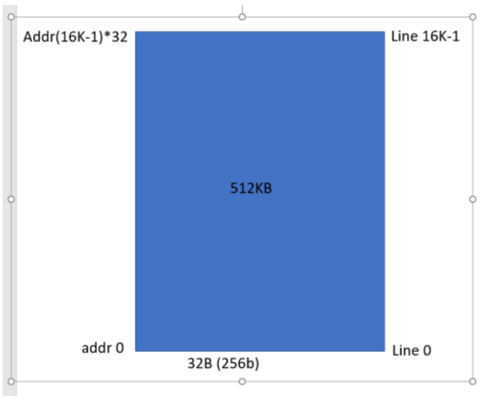
With "breaking" and with good management, we can perform achieve burst mode for each client

Burst mode = after some cycles, the client get its information cycle by cycle until the end of information.

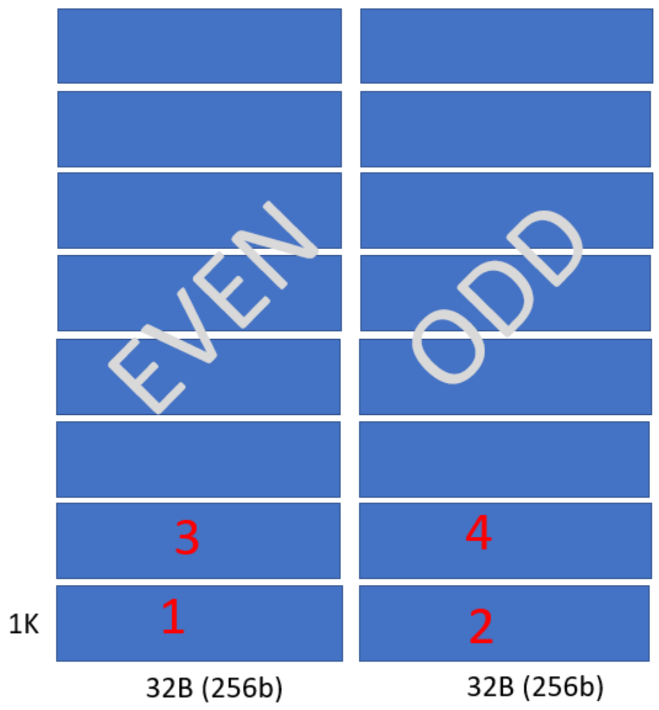
Memory dimensions

We have memory of total 512KB, all byte is address. i.e. 2^{19} addresses.

Each line is 32B (256 bit), so wee have 16K lines:



We "break" the memory into 16 pieces, each piece has 1K lines of 32B:



In this configuration the addresses don't continuous.

We have 8 regions that each region has even and odd piece

The addresses arranged in the following

In piece 1 – 0-31,64-95,...

In piece 2 – 32-63,96-127,...

In piece 3 – 32K-(32K+31),...

In piece 4 – (32K+32)-(32K+63),...

Etc..

This form allowing us to serve two clients simultaneous in the same region.

For example, clients1 and client2 want the same data from address 0 to 127

The clients get the information as the following:

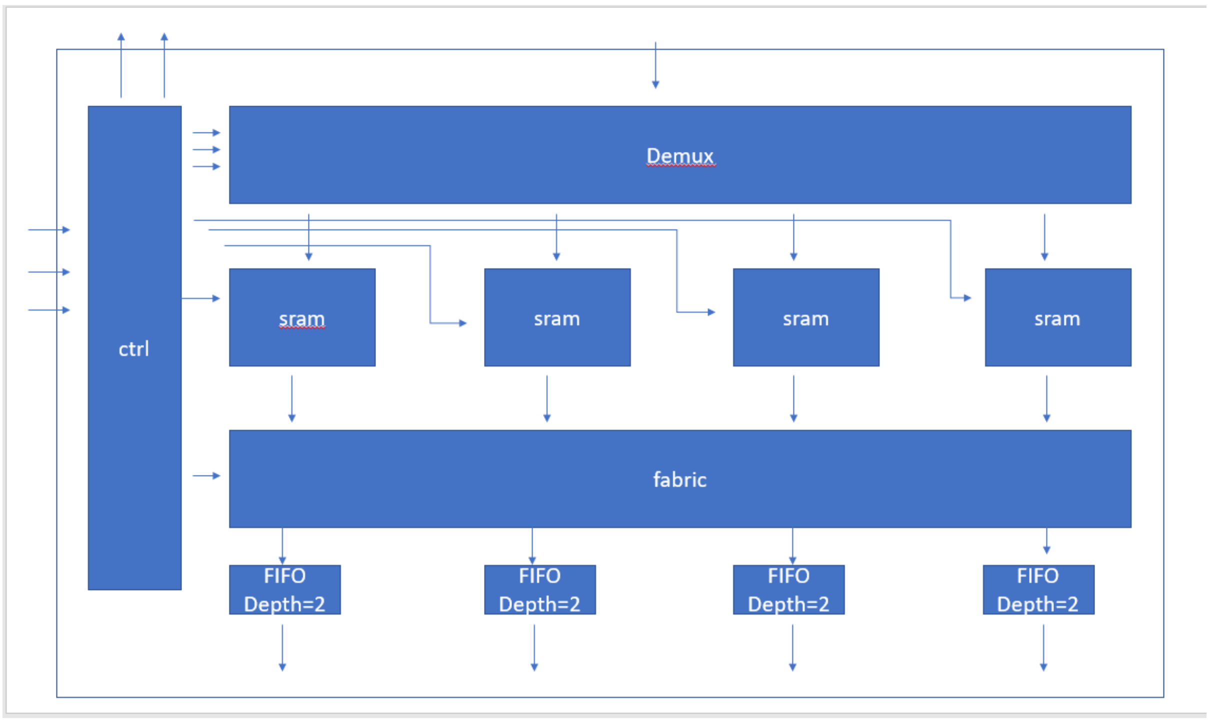
cycle	Client1	Client2
1	0-31	--
2	32-63	0-31
3	64-95	32-63
4	96-127	64-95
5	--	96-127

We can see that after one cycle the clients will get the information in burst mode.

Implementation - General Description:

The memory consists of two part, one for client reading and the second for client writing,
The parts almost symmetrical, so I will present here just for client reading.
We will use in interface component of System Verilog to create the same communication protocol between all the clients and the memory.

Block Diagram:



From above there is the ddr (dram).

From left there is the software.

From bottom there is the clients.

In the Diagram above there is just 4 sram (pieces of memory) this is for simplify, in the fact, there is 16 pieces. Same for the num of clients that more than 4.

Sub-blocks

Demux -Get the data from the ddr and by the control signals send it to the appropriate piece (sram).

Sram – store the data, the memory core.

Fabric – get the data from the srams and pass it to the appropriate client.

FIFO- align the data to full line (for case that the address in the middle of the line)

Ctrl – the controller of the units.

Interfaces:

Mem_intf_read :

Common for all the units

Name	I/O client	I/O memory	Comment
Mem_req	O	I	Request for read.
Mem_data_valid	I	O	The data valid in this cycle
Mem_start_addr	O	I	The address read
Mem_size_bytes	O	I	The number of bytes to be read
Mem_data	I	O	The data that read
last	I	O	The last cycle of the transaction
Mem_last_valid	I	O	The number of bytes that valid in the last cycle of the transaction

This interface instantiate in the memory and in the clients, so the I/O opposite between them.

Mem_farm:

Name	I/O	Comment
General		
clk	I	clock
rst_n	I	reset negative
Memory Interfaces		
mem_intf_read	IF	Read interfaces from memory as the number of clients
mem_intf_write	IF	Write interfaces from memory as the number of clients
Read_ddr	IF	Read interface from the ddr
Write_ddr	IF	Write interface from the ddr
Software Interface		
Read_addr_ddr	I	From which address to read from the ddr
Write_addr_ddr	I	From which address to write to the ddr
Client_pririty	I	Which client has priority

Schedule:

TASK	15-20-נוב	22-20-נוב	29-20-נוב	06-20-דצמ	13-20-דצמ	20-20-דצמ	27-20-דצמ	03-21-ינו	10-21-ינו	17-21-ינו	24-21-ינו	31-21-ינו	07-21-פבר	14-21-פבר	21-21-פבר	28-21-פבר	07-21-מרץ	14-21-מרץ	21-21-מרץ	28-21-מרץ	04-21-אפר	11-21-אפר	18-21-אפר	25-21-אפר	02-21-מאי	09-21-מאי	16-21-מאי	23-21-מאי	30-21-מאי	06-21-יוני	13-21-יוני	20-21-יוני	27-21-יוני
Comment										exams	exams	exams	exams																				
Initial Design Spec writing	▨	▨	▨																														
Initial skeleton implementation		▨	▨	▨																													
Base Version implementation				▨	▨	▨	▨	▨	▨																								
Final Design spec update									▨	🟧	🟧	🟧	🟧	▨	▨																		
Final version implementation										🟧	🟧	🟧	🟧	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨							
ASIC feasibility check (synthesis)																											▨	▨	▨	▨	▨		
Productizing and documenting																														▨	▨	▨	▨
In parallel to all - continues verification	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	░

References

1. https://towardsdatascience.com/four-common-types-of-neural-network-layers-c0d3bb2a966c
2. https://he.wikipedia.org/wiki/%D7%A8%D7%A9%D7%AA_%D7%A2%D7%A6%D7%91%D7%99%D7%AA_%D7%9E%D7%9C%D7%90%D7%9B%D7%95%D7%AA%D7%99%D7%AA
3. https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/
4. https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8