ABSTRACT

BAKER, LEE B. Design of a 3DIC system to aid in the acceleration of edge systems that employ multiple instances of disparate artificial neural networks. (Under the direction of Paul Franzon.)

Although Artificial Neural Networks (ANN) have been known about for many decades, it hasnt been until the last few years that they have demonstrated efficacy in applications such as image recognition and voice recognition. These ANNs have demonstrated significant improvements over what was considered to be state-of-the-art algorithms.

In most cases, ANNs must be "trained" to perform their function and these recent breakthoughs can, in a large part be attributed to the availability of "big" training data.

This data takes the form of images, recordings etc. and this along with the available storage capacity means that the required training data is now available to train these ANNs.

Artificial Neurons (AN) take their inspiration from neuron behavior observed in the mammalian brain, although implementations are simplifications of what actually exists in the brain. These simplifications range from attempts to emulate the actual spiking behavior of real neurons to ANs that simply encode the spiking behavior in the form of a number or rate.

Surprisingly, the ANNs that have demonstrated the most efficacy are those that employ the more simpler rate-based ANs. Now this may be in large part because these simpler rate-based ANs are easier to process or because large NNs employing the more complex spiking ANs have yet to demonsrate improved efficacy in the various applications. There is a belief that the more complex spiking neurons have the ability to outperform the simpler rate-based neurons but this work focuses on the proven efficacy of ANNs formed from rate-based ANs.

The ANNs that have demonstrated to be most effective are a family of neural networks that can be described as Deep Neural Networks. These DNNs are created by cascading layers of ANs to form a large, layered ANN. These ANNs are in most cases generate outputs in the form of a classification, such as the probability of an image containing a certain object or the output of some approximated function, which might be the expected cost from making a stock trade.

Now researchers have experimented with various sized NNs for various applcations, but those that have demonstrated the most efficacy employ tens of thousands of ANs.

In many implementations of these useful sized NNs, the performance is impacted by the memory bandwidth of the system. Much of the NN application specific (ASIC/ASIP) research has focused on taking advantage of the performance and ease of use of Static Random Access Memory or SRAM. These implementations can be shown to be effective with specific NN architectures, such as Convolutional NNs but in reality, these implementations do not provide the flexibility, storage capacity and deterministic performance required to implement all useful sized NNs.

In addition, it is this works belief that that real-world applications will employ multiple instances of these useful sized ANNs and current implementations will not meet the demands of these multi NN systems.

One area of integrated circuit technology that hasnt been widely used in ANNs is 3-D integrated circuits (3DIC). 3DIC has the potential to increase connectivity, and thus bandwidth and keep power dissipation to within acceptable levels.

This work combines ANNs with 3DIC technology to demonstrate how a 3DIC dynamic random access memory (DRAM) memory can be combined with customized IC layers to produce a system providing an acceptable level of performance in systems with multiple instances of various types of DNNs.

This work includes utilizing a customized 3D DRAM along with a system which includes a management layer which coordinates and executes ANN operations in the form of unique instructions and a processing layer able to process data from the DRAM, via the manager at a bandwidth that meets the demands of a system employing multiple ANNs.

Design of a 3DIC system to aid in the acceleration of edge systems that employ multiple instances of disparate artificial neural networks

by
Lee B. Baker

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2017

APPROVED BY:

Winser Alexander	Gregory Byrd
Richard Warr	Paul Franzon Chair of Advisory Committee

DEDICATION

To my wife Mandy, my children Adam, Rachel and Paul and my parents Joan and Barry.

BIOGRAPHY

The author was born in a the United Kingdom. After performing poorly in high school he took a job in a local electronic engineering firm under a vocational program. After seeing the white coated "engineers" being called down from upstairs to solve the "big" problems, he decided he wanted to wear one of those white coats (his dress sense was wanting). The journey took him to Brighton Polytechnic, now Brighton University and a First Class Honours Degree in Electrical Engineering. After working in the UK for a couple of years, he moved to the United States. The journey included a family with a daughter and two son's. The education continued with a Masters in Engineering from Villanova University and a Masters in Business Administration from North Carolina State University.

With the family now being somewhat independent, he decided to make a career change which would hopefully include teaching.

That career change included enrolling in the Electrical Engineering PhD program at North Carolina State University. This stage of the education journey has resulted in this dissertation.

And remember:

"do not stand still."

"do not let your past dictate your future."

ACKNOWLEDGEMENTS

At a personal level, I would like to thank my wife Mandy and my children Adam, Rachel and Paul for their encouragement.

I would like to thank my advisor, Paul Franzon for his help in making this possible.

I would also like to thank my fellow students, especially Jong Beom, Josh, Sumon and Weifu for their healthy discussions and, being an older student, referring to me as Lee and not Sir or Mr. Baker.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

Recently, there has been much interest in the use of artificial neural networks in systems that employ tasks such as image recognition[Kri12], text recognition[Qiu13] and game playing[Mad14]. In particular, in the field of image recognition these artificial neural network models have demonstrated superior performance over other state-of-the-art technology[Kri12]. These artificial neural networks will continue to be applied to numerous other areas such as voice recognition, text recognition, face recognition and autonomous control.

Artificial neural networks (ANN) take their inspiration from neuron behavior observed in the mammalian brain, although implementations are simplifications of what actually exists in the brain.

The mammalian neuron is a cell that receives input and generates output in the form of electrical and chemical processes. The neuron has a cell body (or soma), a group of dendrites which provide

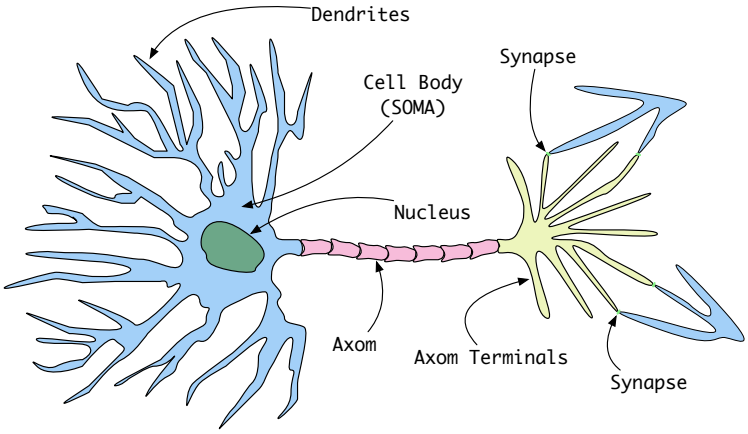


Figure 1.1 Neuron

the inputs from other cells, a cell body, an axom which generates the output signals, and the axom terminals which are the outputs of the cell. The connection from a cells output, or axom terminal to another cells input, or dendrite is known as a synapse. The connection in the synapse is a chemical process stimulated by electrical impulses. The neuron can be seen in figure 1.1.

The connection from one cell to another has both an associated delay and a strength. The strength of the connection can be influenced by the size of the pre-synaptic neuron spike or by the pre-synaptic neuron generating a series of spikes rather than a single spike.

So it is known that mammalian neurons generate "spikes" in response to inputs which for humans include sight, touch, sound etc.. This spiking behavior is often referred to as the neuron being activated. When these neurons are activated, their spikes propagate to other neurons. Under certain conditions, the combination of the various inputs to a neuron cause it to activate. A particular neuron may have many hundreds, perhaps thousands of other neurons connected to its "input". These input neurons are referred to as pre-synaptic neurons. These pre-synaptic neurons may provide input to many neurons which are referred to as post-synaptic neurons. A particular neuron can get activated by a particular arrival pattern of pre-synaptic neuron spikes or simply by the intensity of the pre-synaptic spikes.

The spiking behavior of a neuron also varies and many spiking profiles have been observed, including single spikes, groups of spikes and repetitive spiking. It is believed that information is carried in the delay and strength of the connections and how pre-synaptic neurons combine to cause a neuron to activate. In simple terms, if a neuron is activated by its pre-synaptic neurons, then the activation of the neuron means a pattern has been detected which will influence a reaction. In mammalian terms, that might be the detection of a threat from both smell and sight neurons and the reaction is to control muscles resulting in flight.

The various chemical and electrical processes that result in the generation and propagation of these neuron spikes is beyond the scope of this dissertation, but how neurons and networks of neurons are artificially emulated is what we will duscuss next.

Artificial Neural Networks

When modeling these neurons in artificial neural networks, the neuron models either generate actual spikes similar to actual neurons or produce a value which is proportional to the rate at which spikes occur. These artificial neural networks can be categorized as rate-based coded or spike time coded neurons.

When used in networks of neurons, both model types employ a connection weight between the pre and post-synaptic neuron, however, the spiking neuron network also introduces a time delay associated with the connection.

The spiking neuron model is characterized by:

- Connections between neurons have both a strength and a delay
  - The pre-synaptic neuron output is multiplied by the connection weight and delayed
- The weighted inputs from all pre-synaptic neurons are accumulated
- The accumulated inputs drives an activation function

  - the activation function $f(x)$ is a spiking model is based on differential equations
  - many models have been proposed with varying levels of complexity

    examples are:

    · Leaky integrate and fire

    · Izhikevich [**Iz2005**] (see Fig. 1.4a)

The Rate-based neuron model is characterized by:

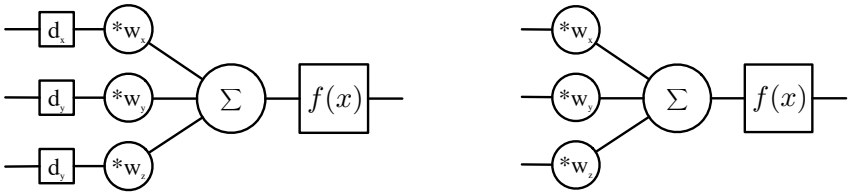
- Connections between neurons have only a strength
  - The pre-synaptic neuron output is multiplied by the connection weight
- The weighted inputs from all pre-synaptic neurons are accumulated
- The accumulated inputs drives an activation function
  - the activation function $f(x)$ is a non-linear function
  - early models used binary functions although in practice the function needs to be differentiable

    examples are (see Fig. 1.3):

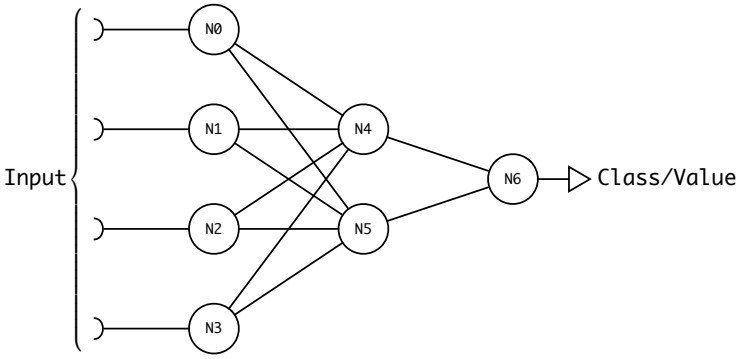
    · sigmoid

    · rectified linear unit

To emulate complex behavior, the artifical neurons are connected in networks, typically with layers of sub-networks which are in affect separated by the non-linear activation function. Examples of both rate-based and spiking artificial neural networks can be seen in Fig. 1.5a and Fig. 1.5b respectively. Typically neural networks process in a feed-forward fashion. Considering Fig. 1.2, this means the input arrives on the left, the inputs propagate to neurons N0 through N3. WHen N0 through N3 are processed, their values propagate forward to neurons N4 and N5 etc.. SOmetime

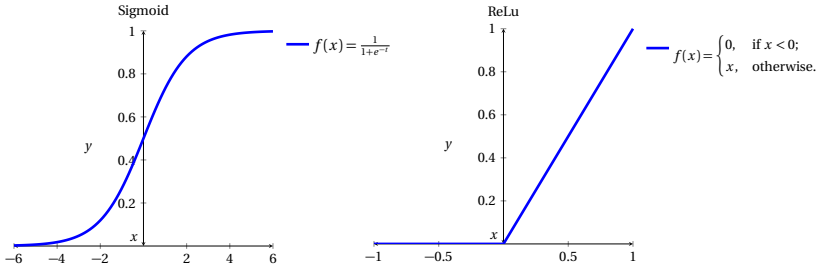


(a) Spiking Model **(b)** Rate-Based Model



(c) Network of Artificial Neurons

Figure 1.2 Artificial Neural Network

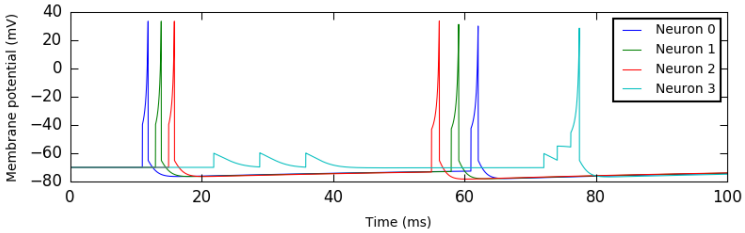


(a) Sigmoid **(b)** Relu

Figure 1.3 Example Rated-Based Model Activation functions

$$v' = 0.04v^2 + 5v + 140 - u - I$$
$$u' = a(bv - u)$$
$$\text{if } v \geq 30\,\text{mV, then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases}$$

(a) Izhikevich Model



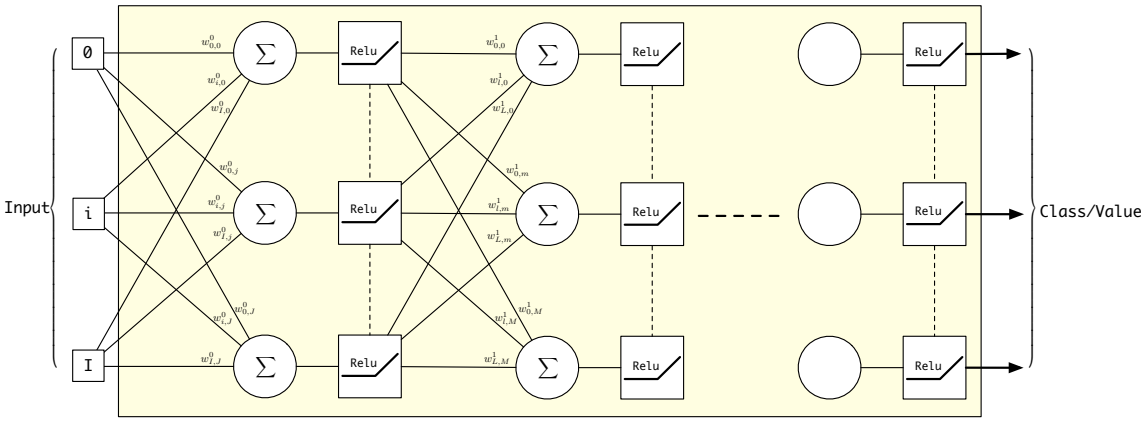
(b) Izhikevich [**Iz2005**] Model Simulation

Figure 1.4 Example Spiking Activation Function Model

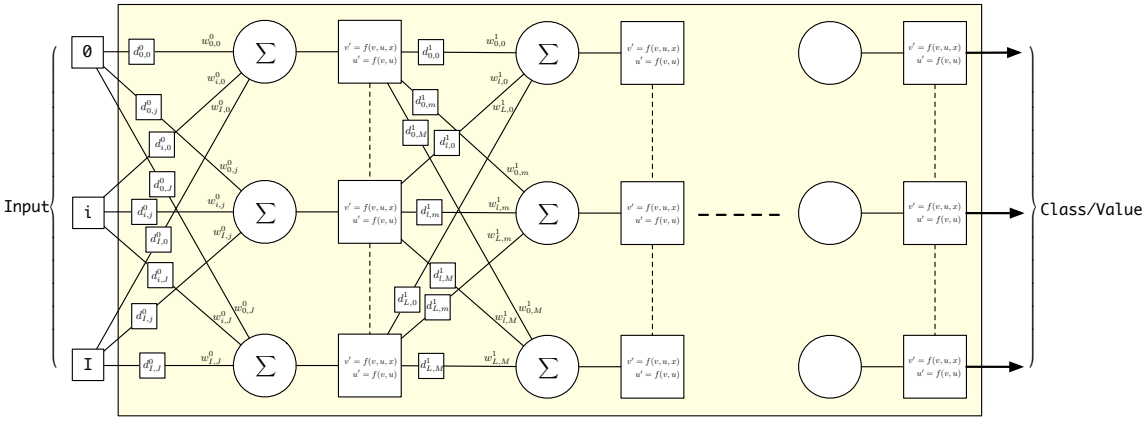
ANNs also include recursion where for example neurons N0 through N4 are not only influenced by the input, but also by themselves. Many ANNs operate only in feed-forward fashion but some popular ANNs, such as Long short-term memory (LSTM), employ recursion.

Another popular ANN known as Deep Neural Networks (DNN) have proved very popular over the last few years. They get good press in applications such as image recognition and speech recognition. Deep Neural Networks are often formed from tens of layers of ANs with each layer containing many ANs. DNNs are also processed in a feed-forward manner with one layer being the inputs to the next layer. As mentioned [Kri12], these useful DNNs often require hundreds of thousands of ANs and within the network, each AN can have hundreds, even thousands of feeder or pre-synaptic ANs. There have been implementations that use different number formats from double precision floating point to eight bit integers, but in all cases these useful ANNs require a significant amount of memory to store the connection weights (parameters).

Although the spiking neural network more closely models the behavior of real neurons, over the



(a) Rate-based Model Artificial Neural Network (with ReLu activation function))



(b) Spiking-based Model Artificial Neural Network

Figure 1.5 Example Artificial Neural Networks

last 20 years there have been breakthroughs in the configuring of rate-based models especially with the introduction of the back-propagation algorithm and stochastic gradient descent. Along with the abundance of data now available in the form of voice, images etc. to "teach" these networks using back-propagation, most of the effective applications of artificial neural networks have employed these rate-based models.

The Problem

To approach the capabilities observed in human behavior, such as object recognition these ANNs become very large. They often utilize hundreds of thousands of neurons to implement what a human would consider a relatively straightforward task. For example, a "useful" ANN similar to that described in [Kri12] that is used to recognize up to 1000 different object classes has a network size of approximately 650,000 neurons and 630 million synaptic connections [Kri].

The increased performance of ANNs over classical methods in image recognition and voice recognition might suggest that ANNs might out-perform other existing systems in other applications. There is reason to believe that these ANNs will replace existing control and monitoring functions in existing systems.

If ANNs fulfill their potential, systems employing ANNs will utilize them for various functions, such as engine monitoring, anomaly detection, navigation etc. all within the same system. Considering the various functions a complex customer facing or edge application system performs, it is likely that many real-world applications will employ multiple disparate instances of these useful sized ANNs. Assuming these complex functions will require ANNs similar in size to [Kri12], these implementations will be processing multiple large ANNs at or near real-time.

Considering the storage required for the input, the AN states and most significantly the weights for each of the ANs, the storage requirements results in gigabytes of memory. When these ANNs are required to be solved in fractions of a second, the processing and memory bandwidth becomes prohibitive.

As a metric, this work asumes that any useful DNN will employ 100's of thousands of ANs. Although there is a lot of debate regarding number formats for ANNs, this work also assumes single-precision floating point. Assuming an ANN with 250K neurons and an average fanin to each AN of 2000, a system employing 10 ANNs for various disparate functions and an average processing time of 10 ms suggests a average bandwidth of 16 Tbit/s (see equation 1.1).

$$
\begin{aligned}
\text{Average Bandwidth} &= \sum_{\mathbf{n}=0}^{\bar{\mathbf{N}}_\mathbf{n}} \left(\frac{\bar{\mathbf{N}}_\mathbf{a} \cdot \bar{\mathbf{C}}_\mathbf{p} \cdot \bar{\mathbf{b}}_\mathbf{w}}{\bar{\mathbf{T}}_\mathbf{p}} \right) \\
&= \sum_{\mathbf{n}=0}^{9} \left(\frac{250 \times 10^4 \cdot 2 \times 10^3 \cdot 32}{10 \times 10^{-3}} \right) \\
&= 16\,\text{Tbit/s} \qquad (1.1)
\end{aligned}
$$

where $\mathbf{N}_n$ is the number of ANNs

$\mathbf{N}_a$ is the average number of ANs

$\mathbf{C_p}$ is the average number of connections

and $\mathbf{T_p}$ is the processing time

Therefore, when imlementing ANNs, the memory requirements are significant. The storage is required for the input, the AN state and most significantly the weights for each of the ANs. This storage requirement often results in gigabytes of memory.

In addtion, in edge applications, it is anticipated that these ANNs are required to be solved in fractions of a second, in which case the processing and memory bandwidth becomes prohibitive.

The problem becomes "**to provide deterministic at or near real-time performance within tolerable power and space constraints for edge systems employing inference on multiple disparate useful-sized neural networks.**"

In most cases, Graphics processing Units (GPU) are used to implement large ANNs. In many ANN architectures, such as Convolutional NNs (CNN), they are quite effective. However, we should not forget they are not optimized purely for ANN processing and are restricted by available SRAM and they are power hungry. These limitations will limit the effectiveness of GPUs regardless of what we might hear from the GPU community.

Much of the ANN application specific (ASIC/ASIP) research has focused on taking advantage of

the performance and ease of use of Static Random Access Memory or SRAM. These implementations can be shown to be effective with specific ANN architectures (CNN), server applications or the "toy examples" but when a system requires multiple disparate ANNs in an edge application, these implementations do not provide the required flexibility, storage capacity and deterministic performance.

Considering that DRAM is required to store the NN parameters, why use SRAM as an intermediate store? Well, in practice there are benefits if you can operate solely out of SRAM. Certainly good performance and potentially low power. But use of SRAM makes assumptions on the NNs that can be supported. The primary requirement of the NN to allow effective use of SRAM is "reuse". Once parameters are stored in SRAM, can they be reused such that the SRAM isn't simply an intermediate memory but something akin to a cache.

In some ANNs there are reuse opportunities. Specifially, with CNNs, the weights are reused. A convolutional filter is passed across an input to form the next layer. These filter "kernels" can be held in memory and the input is read from DRAM thus reducing the DRAM bandwidth. Even with DNNs where weights may not be reused, when implementing multiple DNNs, there is opportunity to hold the input in memory. If the system is being employed in cloud applications or in training, there is opportunity to reuse inputs whilst performing batch processing.

But SRAM comes at a price, its big. Often when we see physical layouts of NN processors, they are dominated by the silicon area of the SRAM. The area required for SRAM has been understood for quite some time and companies attempt to create custom SRAMs to minimize the area impact.

So the question becomes, can a system employ DRAM with minimal SRAM and still provide a high performance system within acceptable area constraints?

Even in cloud applications, there are limitations on reuse. We paraphrase a quote from a Google paper [Jou17] on their Tensor Processing Unit ASIC (TPU):

"the architecture research community is paying attention to NNs, but of all the papers at ISCA 2016 on hardware accelerators for NNs, alas, all nine papers looked at CNNs, and only two mentioned

other NNs. Unfortunately CNNs represent only about 5% of our datacenter NN workload”

The applications targeted by the google TPU [Jou17] assume multiple requests, so reuse in the form of batch processing is still of great benefit, but the bulk of the requests in [Jou17] are fully-connected DNNs and in these cases weight reuse is not as benefitial and the performance of the TPU is degraded when implementing these fully-connected DNNs.

Therefore, implementations that focus on CNNs can suffer from severe degradation in performance when targeting generic types of ANN, such as locally and fully connected DNNs and LSTMs.

This work focuses on edge applications employing disparate ANNs and assumes both weight reuse and batch processing do not apply. Considering systems will want to perform multiple DNNs simultaneously suggests that these edge systems will require usable memory bandwidth of the order of 10’s of Tbit/s. In these cases, **DRAM bandwidth is the bottleneck**.

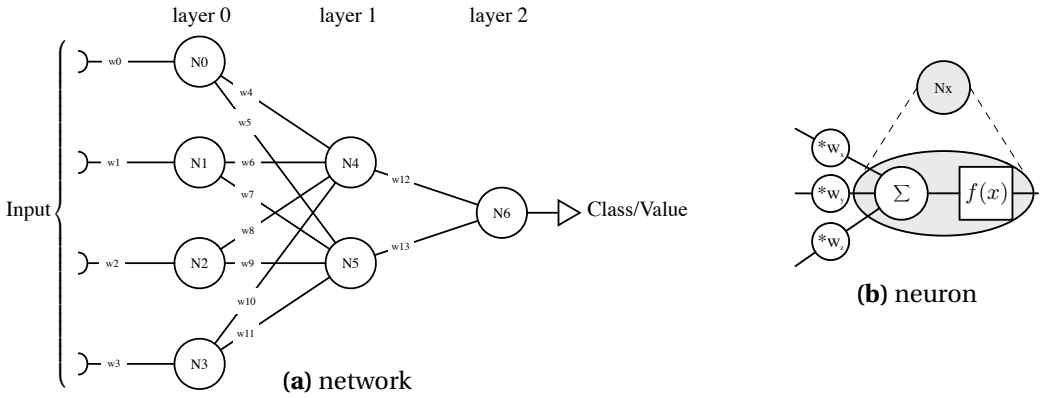
The Solution

The requirements of these applications would currently be satisified by employing multiple graphics processor units(GPU). These multiple GPU systems have high real-estate and power requirements that may exceed the target applications requirements.

This research explores a 3DIC solution using a custom organized 3DIC memory in conjunction with unique data structures and custom processing modules to significantly reduce the area and power footprint of an application that needs to support the processing associated with multiple ANNs.

Artificial Neural Networks are a network of inter-connected processing elements inspired by the connectivity and processing observed in the brain.

They are characterized by the connections between a pre-synaptic neuron and a post-synaptic neuron having a weight which determines the impact the pre-synaptic neuron has on the "activation" of a post-synaptic neuron. Typically the processing portion of the neuron performs a



(a) network

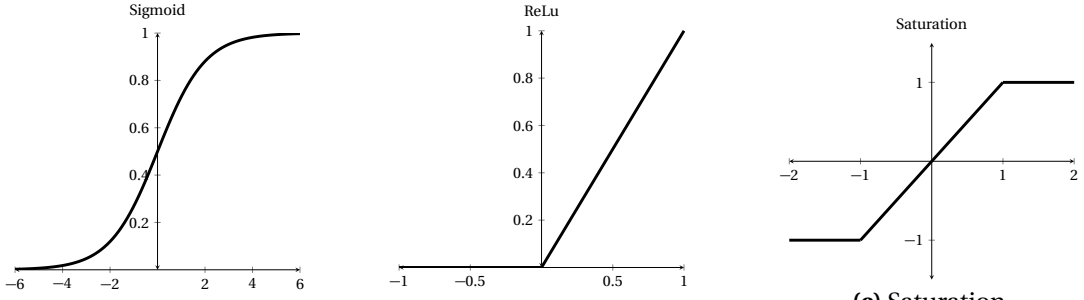
(b) neuron

Figure 1.6 Simple Artificial Neural Network

multiplication and accumulation on the outputs from all the pre-synaptic neurons followed by a non-linear "activation" function, such as a rectified linear unit or sigmoid function to create the neuron output.

It is well understood a single layer of these neurons can be configured to perform a linear function approximation. A collection of these neurons, when connected in layers with one layers neurons connected to the next layers neurons, are known to be able to act as non-linear function approximators. The layering not only allows the ANN to replicate non-linear functions, but also means the ANN layers can be processed in disjoint sections that make it suitable for sequential processing. As well as being effective function approximators, these neurons can be configured to act as classifiers, attractor networks or associative memories. A simple example of an ANN can be seen in Fig. 1.6a and the contents of an artificial neuron with the multiplication and summation stages and the activation function $f(x)$ can be seen in Fig. 1.6b.

For the most part, different ANNs are characterized by the function contained in the processing element, how the neurons are interconnected and how the weights of the inter-connections are configured or learnt. Most of the effective applications of ANNs require 10's of thousands of neurons and millions of synaptic connections. When the entire network must generate a result in near real-



(a) Sigmoid **(b)** ReLu **(c)** Saturation

Figure 1.7 Typical neuron activation functions

time, the result is a system requiring billions of floating point operations per second. When multiple ANNs need to be processed in parallel, the power and area requirements may not always be satisifed using current technology.

Some typical examples of the activation function can be seen in Fig. 1.7.

Deep Neural Networks

A single layer of neurons is a linear classifier and therefore cannot be used to classify unless the classes can be separated using a linear function. Even some simple cases cannot be linearly separated, an example often used is an exclusive-OR gate. To allow classes in these cases to be linearly separated, the input space needs to be transformed. Deep neural networks are ANNs that incorporate many layers of neurons which are often up to 10 layers deep. These additional layers are incorporated to translate the space of the input so the various classes being identified can be separated using linear classifiers in the later layers. A very successful early example known as a convolutional neural network (CNN) is which in [Kri12] was used to classify objects in images and shown in Fig. 1.8. These CNN's use the early layers to identify low-level features and later layers are used combine these features into yet more higher-level features. Finally, the combination of high-level features are used to classify. This layering is shown in Fig. 1.9. Finally, a fully connected linear classify is used to classify the object using this translated input. These deep neural networks can be used as classifiers or as

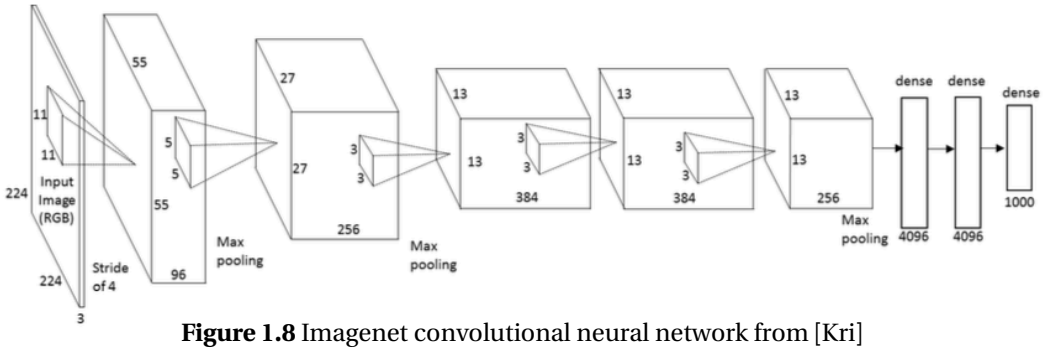


Figure 1.8 Imagenet convolutional neural network from [Kri]

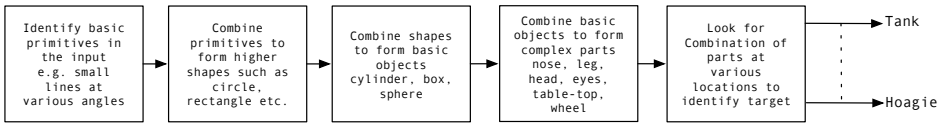


Figure 1.9 Deep network showing feature layers

function approximators as in our implementation of reinforcement learning. In practice, useful deep neural networks are very large and often exceed the capacity of single GPU's. Therefore, there is opportunity for a solution that provides real-time acceleration and storage of of these systems.

Reinforcement Learning

In the context of this work, reinforcement learning is the process by which a system identifies the actions to take to maximize reward. For example, what actions should an aircraft take to avoid crashing (low reward) versus landing (high reward). In this context of this work, we are discussing solving the action-reward function associated with a complex Markov-Decision-Process (MDP). The action-reward function tells an agent operating in an environment the expected accumulated reward if an action is taken in a particular state. The idea being the agent should always take the action that provides maximum expected reward. The action-value function is defined for any state an agent may find itself in. There are various dynamic programming methods employed to "learn" this action-value function and in most cases through repeated trials. However, in most useful environments, the size of the state-space is too large to learn and/or store each states action-value

table, therefore action-value function approximators are used and deep neural networks are used as these approximators [Mni13]. The focus of this work will be on accelerating the application of deep NNs to action-value function approximators.

CHAPTER

2

STATE OF THE ART

For the most part, large scale ANNs have been implemented using GPU's. In some cases, these GPU's have demonstrated efficacy in the highly parallel processing required for some specific NNs such as Convolutional NNs. There are also custom implementations that have targeted specific ANNs[Che16][Far11], again such as convolutional neural networks (CNN). In most cases, these implementations focus on solving specific "hot-spots" inherent in the processing of the network[Che16]. Almost all ASIC solutions employ arrays of processing elements (PE) each with local processing capability and local memory. For most of these, the size of the network supported is limited by the size of the local memory and this large local memory limits the number of available processing functions. In some cases , as seen in Fig. 2.1the area consumed for local memory can exceed of the order of 65% of the processing element die [Kim16][Che14]. Those that employ external DRAM,

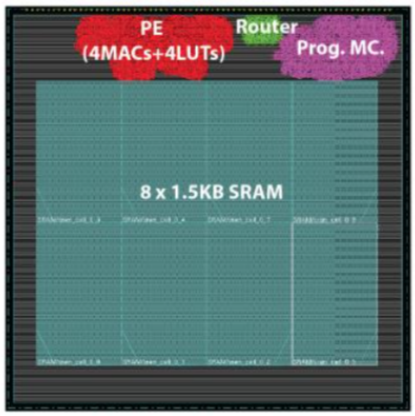
such as NnSP[Esm05] and NeuroCube[Kim16] still load weights and inputs to local SRAM prior to processing. In the case of NnSP[Esm05], the paper discusses caching data to bridge the speed gap between external memory and the PE but does not provide details on how to ensure data locality when reading a DRAM cacheline and how to minimize the impact of the DRAM protocol. NnSP does not provide any detail regarding network size and supported types. Neuflow[Far11] is limited to CNNs and the external memory is QDR SRAM and thus will be limited by the network size. NeuroCube uses a 3D stack along with HMC 3D-DRAM and data is transferred from the DRAM to the PE's via a network-on-chip (NoC). The combination of limited HMC interface bandwidth and the NoC limits the processing performance. Eyeriss[Che16] focuses on CNNs and specifically on the convolution "hot-spots". It does not support the pooling operations although these can be supported by a local CPU but it does not support the memory intensive classifier stage. Eyeriss can not be effectively applied to locally-connected type ANNs such as Deepface [Tai14].

Unlike the current state-of-the-art, this work focuses on processing data directly as that data is read out of the DRAM thus avoiding requiring excessive SRAM in the PE's thus allowing optimum logic assignment to the processing functions.

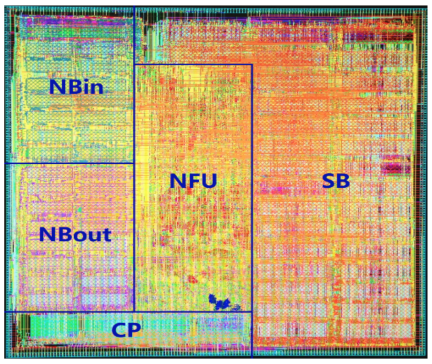
To reiterate, state-of-the-art does not:

- currently propose streaming data directly from 3D-DRAM through the processing functions to avoid the use of large local memory
- propose data structures to support continuous streaming from DRAM
- propose a 3D-stack supporting more than one processing layer

NeuroCube[Kim16] is the only solution to explicitly discuss 3D-memory but will again be limited by its focus on utilizing local SRAM where our focus is on providing silicon for processing and not local storage. Neuflow[Far11] has processing elements that are limited to CNNs. The Tensor processing Unit (TPU) [Jou17] employs an 256x256 array of eight bit processing elements but specifically states it is bandwidth limited with fully connected (MLP) networks NnSP[Esm05] transfers weights and inputs to local PE memory prior to processing. nn-X[Gok14] limits the number format to integer,



(a) NeuroCube



(b) Diannao (SB and NBx are SRAM)

Figure 2.1 Example state-of-the-art die

and although there is some acceptance that similar results can be obtained compared to floating point, this will likely limit the applcation space. nn-X is also limited to performing the convolution operation and SRAM size will limit its capability when performing the classifier stage.

Below is a summary of current technology.

2.1 NnSP[Esm05]

Uses external DRAM with a cache. Does not address the main issue of DRAM access limitations and providing data structures. It simply says "use a cache" but doesnt justify locality issues.

Transfers weights and inputs to local PE prior to processing Similar "Streaming" terminlogy but streams from local memory. Neuron outputs are kept locally and transferred to other neurons thru NoC

Implemented in FPGA

No detail on processing element but likely limited to "standard" feedforward NN's

2.2 NeuroCube[Kim16]

NeuroCube[Kim16] transfers data to PE prior to processing which can limit classifier size although most convolution kernels can be supported. The SRAM on the PE is significant (see Fig. 2.1a). NeuroCube uses HMC 3D-DRAM which has limited bandwidth and this will limit network processing performance. The NeuroCube implements a processing layer but does not address adding additional layers.

3D uses HMC - limited by HMC bandwidth - is serdes a good choice for die-to-die communication (typically used for die to chip/board?) - doesnt really discuss adding additional layers

Transfers data to PE prior to processing

2.3 IMAPCAR [KO11]

IMAPCAR uses external SRAM so will be limited by network size. The architecture of IMAPCAR is optimized for processing video.

In general will not scale to ANNs of interest.

2.4 NeuFlow [Far11]

Neuflow[Far11] has processing elements that are "highly optimized" to CNNs. Neuflow is designed around external QDR memory and does not address the issues associated with supporting large networks.

2.5 nn-X [Gok14]

nn-X[Gok14] limits the number format to integer, and although there is some acceptance that similar results can be obtained, this will likely limit the applcation space. nn-X is also limited to performing

the convolution operation and SRAM size will limit its capability when performing the classifier stage. nn-X was implemented in an fpga and does not address SRAM size when scaling to larger networks.

2.6 Diannao [Che14]

Diannao[Che14] acknowledges the need for large amounts of storage which will exceed the capability of local memory. The kernel sizes are limited by SRAM size although it does state kernels can be "broken up". The SRAM use on the PE is significant (see Fig. 2.1b).

2.7 Eyeriss [Che16]

Eyeris[Che16] focuses on CNNs and specifically the convolution stage. Eyeriss ignores the pooling and classifier stages which although not as compute intensive they are memory intensive. Therefore Eyeriss limits network size and does not implement the network.

2.8 TPU [Jou17]

The TPU utilizes an array of 64K eight bit fixed point processing elements and demonstrates significant (30X) performance improvements over GPUs and CPUs. It does employ DRAM which provides the required storage capacity but employs SRAM as an intermediate store between the DRAM and the PEs. This Google designed solution acknowledges that it is bandwidth limited when implementing the fully connected MLP type NNs but they still achieve significant perfroamce because of the reliance on very high levels of batch processing. The DRAM bandwidth is ~60 Tbit/s, which is relatively low for a 2D solution. The paper also states that their experience of implementing NNs in the Google server farms suggests that these fully connected NNs represent the bulk of their processing requirements. The simpler CNNs only represent 5% of the servers NN processing requirements. It

should also be stated that this paper also believes that GPU solutions cannot reach the performance targets even though the GPU community might state otherwise.

CHAPTER

3

RESEARCH OBJECTIVE AND ANTICIPATED RESULTS

The goals of this work is to research and design an ASIC based solution implemented using 3DIC technology able to support at or near real-time simultaneous processing of multiple ANNs.

This research will target a family of ANNs that have demonstrated efficacy in a family of popular applications. These applications have large neural networks requiring a large amount of memory to support weight storage and the requirement of processing multiple ANNs at or near real-time.

Targeted Applications

This work will target:

1. Brain-state-in-a-Box (BsB) and Cogent Confabulation (CC) because of their demonstrated

efficacy in text recognition [Qiu13].

- The system described in [Qiu13] utilized 78 clusters with each cluster incorporating 22 Sony PS3 game stations and two NVidia GPU cards. The text recognition system was segmented into character recognition performed by the BsB networks and the word and sentence construction was performed by Cogent Confabulation. There is opportunity to accelerate BsB and potential to accelerate portions of the Cogent Confabulation networks.

2. Deep neural network classifiers [Kri12] because of their demonstrated efficacy in image recognition
    - The application space includes autonomous navigation of automobiles [Boj16] with a need to process multiple classifiers simultaneously at real-time.
3. Reinforcement learning in the context of ANNs
    - The reinforcement learning algorithm employs deep neural networks in action-value function approximation [Mni13]. Considering most environments will have very large state spaces, environment exploration will result in large numbers of action-value calculations and therefore acceleration will improve processing times. It should be noted that supporting reinforcement learning may require provisions for acceleration of back-propagation.

The result of this research will be:

1. a custom organized DRAM specification with optimal bank, page and IO organization to support the target systems
2. a set of data structures for storage of inputs, weights and matrices to ensure high bandwidth utilization
3. a 3DIC architecture including a communication bus proposal to carry inputs and results between processing layers and the memory management layer
    - will include a packet structure with data and configuration formats

4. a processing layer with a group of streaming operations customized for each of the target ANNs
    - the PE will include a SIMD engine to perform additional processing not suited to hardware acceleration
    - a separate processing layer(s) may be provided for each target ANN

This work expects to demonstrate a power/performance improvement over a GPU solution while significantly reducing the physical footprint. The work also expects to demonstrate capacity and processing improvements compared to state-of-the-art ASIC solutions which in most cases are not able to support ANNs of this type, number and size.

Current GPU's are to some extent general purpose processors and therefore have excess silicon and power not directly involved in solving any given application. It is not uncommon for a GPU, because of memory limitations to support a single effective ANN whilst consuming the order of $> 100\,\mathrm{W}$. This work expects to demonstrate a power/performance improvement over current GPU solutions by benefiting from the power and performance advantages of 3DIC architecture whilst simultaneouly supporting multiple ANNs.

CHAPTER

4

TECHNICAL APPROACH

The goals of this work is to research and design an ASIC based solution implemented using 3DIC technology able to support at or near real-time smultaneous processing of multiple ANNs. To achieve a power/performance improvement over GPU's and potentially other solutions employing 3D DRAM, this work needs to provide both a power and bandwidth improvement. This will be achieved by keeping the entire solution within the physical footprint of a 3DIC die stack [Tsa08]. To achieve this goal, the solution will employ a custom organized 3D-DRAM to provide the required capacity along with custom management and processing layers to perform all the computations associated with the multi-ANN application within the 3DIC die stack. To provide the required bandwidth, this work will research a custom organized DRAM and data structures which will allow the processing layers to perform all the ANN computations with direct access to the DRAM. The combination of custom

organized DRAM along with custom data structures will eliminate the requirement to store or cache the ANN data in local SRAM thus providing additional silicon to be applied to the computational units. By keeping the solution within the 3DIC footprint, this work avoids the bandwidth loss due to input/output (IO) pin density reduction associated with communicating off chip. This work will employ high density TSV technology to provide the necessary communication bandwidth between main memory and the computational units. These low capacitance TSV connections will reduce power dissipation compared to an off-chip IO solution whilst keeping the power dissipation within manageable limits. By keeping within the 3DIC footprint, this work anticipates that even when including the usual system infrastructure this work will provide a solution to applications that have significant size and power constraints.

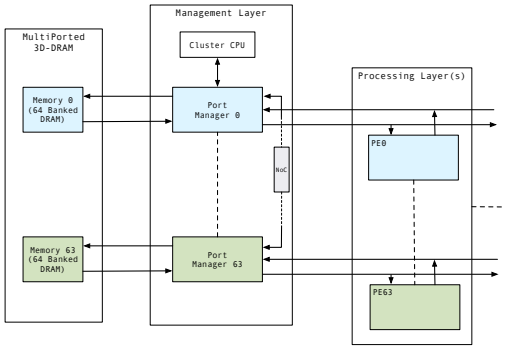
Our proposed solution shown in Fig. 4.1 includes a 3D-DRAM on top of a stack of custom die which includes a DRAM management layer and additional processing layers. The die will be connected using through silicon vias (TSV). The custom DRAM is sub-divided into smaller DRAM's accessed through individual ports. These ports include multiple banks and pages. Each port is controlled by a port manager module. The port manager reads operation units known as work units from a local instruction memory. The work units identify the function, the memory locations to be streamed and the destination processing layer. The port manager first transmits a configuration packet to the processing layer identifying the functions to be performed and how the operands will be presented on the stack communication bus. The port manager then streams the operands to the processing layer. The processing layer sends the result back to the management layer where the result is directed to the appropriate DRAM port for storage. The management layer may include:

- a cluster CPU which is responsible for uploading the work unit instruction memory and to perform any operations more suited to a CPU
- port managers each controlling configuring the processing layer PE and streaming the operands and coalescing results back to DRAM

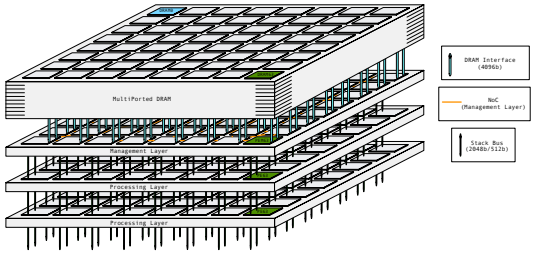
- a network-on-chip for communicating results to other port managers within the management layer

The target neural networks will be analyzed for storage requirements and access patterns. To achieve this, the target applications will be sub-divided into specific ANNs, such as BsB and cogent confabulation. These ANNs will be further sub-divided into primitive operations, such as convolutions on an image region-of-interest or BsB processing of a character image etc.. These primitive operations will be analyzed to determine the optimum DRAM organization and data structures. Of particular interest will be access patterns associated with cogent confabulation. The accessing of the cogent confabulation knowledge base matrices is not deterministic and it is this works intention to analyze access patterns and research options for sparce matrix storage and access. It is anticipated that the access patterns will strongly influence the design of the sparse matrix storage method.

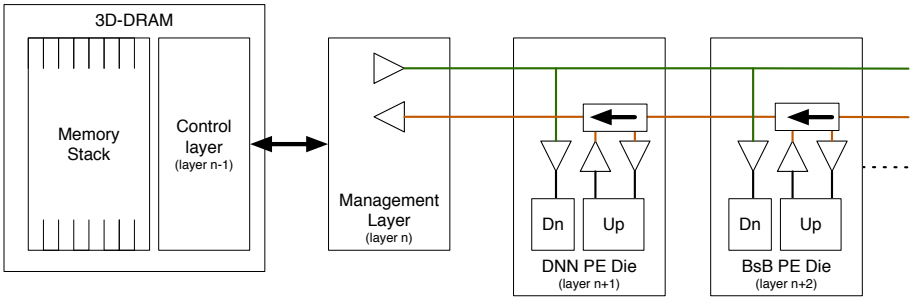
This combination of DRAM organization and data structures will be researched to ensure an acceptable level of bandwidth utilization is obtained when accessing data in the DRAM. The system will be further analyzed for power dissipation under the target applications. This combination of specific DRAM organization and data structures makes this work unique. The target applications will then be simulated to verify the system performance meets our anticipated results.



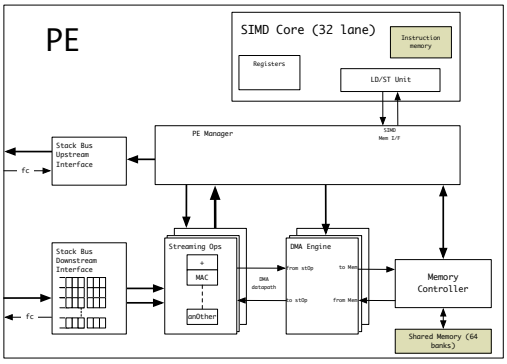
(a) System block diagram



(b) Die stack



(c) Stack Bus



(d) PE block diagram

Figure 4.1 Example 3DIC solution

CHAPTER

5

PRELIMINARY WORK

A small team within the NCSU ECE department, known as the DARPA Cortical team worked on accelerating two types of ANN known as Sparsey and HTM. The work[Fra16] focused on comparing GPU implementations to one solution implemented using a pure ASIC and another solution using an ASIP. Both Sparsey and HTM were implemented on a GPU and then power and speed were compared against the custom ASIC and ASIP solutions. The DARPA Cortical team were able to demonstrate a overall performance improvement of better than two orders of magnitude over a GPU solution. Although these designs employed only SRAM, and the size of the networks were constrained, the Cortical team demonstrated the potential for significant performance gains accelerating ANNs using ASIC or ASIP solutions.

This works research will propose a custom organized 3D-DRAM and special processing functions

optimized for a family of popular ANNs. The solution is contained within a 3DIC footprint which inherently reduces power and increases bandwidth by keeping communication largely "on silicon".

To achieve the performance improvement over current generation GPU's, this work is targeting a 3DIC stack with ten's of thousands of connections (TSV's) between die with logic running at approximately 1GHz. The preliminary work carried out to date was a feasibility study to ensure the various research goals were achievable. This work was based on a proposed partially re-organized version of the DRAM described in [Tez]. The initial re-organization proposes gaining access to an entire page within the DRAM. This will provide a raw DRAM bandwidth of 131Tbps spread over an array of 64 processing engines. The feasibility study involved:

- a baseline application with eight CNNs [Kri]
- a custom organized 3D-DRAM DiRAM4 [Tez]
- a power estimate of the primary energy consumers
- a real-estate study of the processing layer

During the DARPA Cortical project, a rudimentary design was completed that managed and performed floating point streaming operations. The design included a streaming operation module, a circuit-switched Network-on-chip (NoC) module, a DMA engine, a memory controller and general control logic. The synthesis results from that work were used in the feasibility study. In addition, [GH11] was used to estimate fused multiply-accumulate (FMA) function power and size, [Liu12] was used to estimate energy in TSV's in a die stack, [Pat14] was used to estimate real-estate for TSV's and [Tez] was used to estimate energy associated with 3D-DRAM accesses.

For much of the feasibility study, this work used the Tezzaron DiRAM4 [Tez]. The area of the Tezzaron DiRAM4 die is approximately 12.5 mm by 14.0 mm or 175 mm^2. This area was used as the baseline for the 3D stack die size.

For comparison, a typical GPU [Tec] has a silicon area of 550 mm^2 using a 28 nm technology node and consumes in excess of 100 W.

Assuming these ANNs are processing images at a frame rate of 60fps, we will allocate 10 ms to complete computations associated with the ANNs. We will also assume each NN is of a similar size to the CNN described in [Kri] which we estimate to have 2.1 billion floating point operations. The floating point operations per second (FLOPS) estimate for eight NN's is therefore 1.7 TFLOPS which correlates to a bandwidth of 54 Tbits/sec. The design target summary is shown in table 5.1.

Table 5.1 Design targets

Parameter	Target
Power	40 W
Bandwidth	54 Tbit/s
Die Area	175 mm^2

In the feasibility study, we assume the DiRAM4 port can be re-organized to provide access to the 4096 bit cacheline. The DRAM page open and close constraints are such that we can only gain access to a page every two clocks. Therefore, the port manager transfers 2048 bits down the stack bus every clock period of 1 ns.

With 64-ports this provides a raw bandwidth of 131 Tbits/sec. This work anticipates that the data structure resulting from the research will result in the solution being able to meet or exceed the 42% bandwidth utilization of the baseline.

The port manager shown in Fig. 4.1a will transmit computation operands to a processing element (PE) on the processing layer. The PE will contain a number of executions lanes and each execution lane can receive up to two operands every clock cycle from the port manager. A "float" is 32 bits, therefore the stack bus width of 2048 bits will support 32 execution lanes on the corresponding PE. The DiRAM4 has 64-ports, therefore the processing layer will support 64 PE's with each PE supporting 32 execution lanes. In practice, each execution lane will be performing a multiply-accumulate on the weight/input tuple from the stack bus. Therefore, for the PE design we assume 32 FMA's, a small amount of SRAM, a 32-lane NCSU SIMD core and dma/control logic whose size is estimated from [Fra16].

The overall power estimate is based on estimates of the PE FMA's, the SIMD, the DRAM, the TSV interconnects and an estimate for the general control logic. The main concern with respect to area is supporting the number of computational units in each PE within the processing layer. The area of each PE is based on the SIMD, FMA, SRAM, DMA and control logic.

The SIMD and control logic is based on the work from [Fra16]. The area and power of the FMA is based on [GH11]. The DRAM power is based on [Tez]. The overall power estimate is shown in Table 5.2a. The available area is based on the size of the DiRAM4. The area of the PE is estimated by scaling the area of the individual units. The area was estimated using nations of best through worst case scaling numbers from [Sch14] The PE area for the various scaling can be seen in Fig. 5.2b. Only the estimated area using worst case scaling exceeds the area available.

	Technology		
Unit	130 nm	65 nm	45 nm
SIMD (32-lane)	3200000		
DMA		368000	
Memory Controller		294000	
Control		1032000	
SRAM		740098	
FMA			405440

(a) Original area estimates

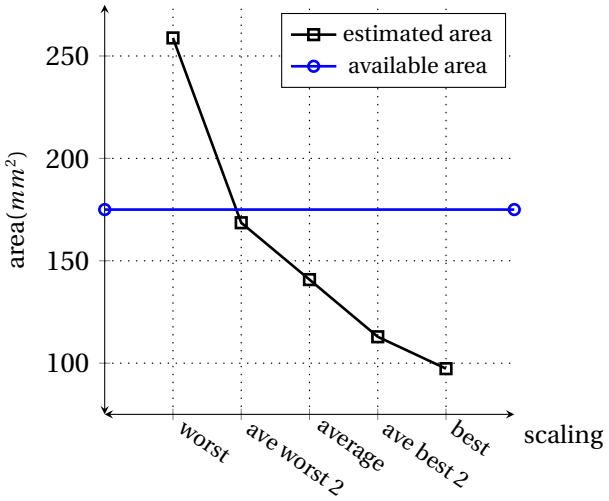
	Technology			
Source	130 nm	65 nm	45 nm	32 nm
ITRS	8.00	2.00	1.00	0.66
PTM	8.00	2.00	1.00	0.50
Intel	2.37	1.33	1.00	0.75

(b) Scaling numbers (from [Sch14])

Figure 5.1 Original area estimates and scaling numbers

Function	Power
FMA	4.7 W
General	16 W
SIMD	1.6 W
DRAM	3.4 W
TSV's	13.2 W
Total	38.9 W

(a) Power estimates



(b) PE layer area estimate after scaling (using [Sch14])

Figure 5.2 Power and area estimates

CHAPTER

6

RESEARCH PLAN

This work will identify and emulate systems employing the ANNs targeted by this work. The systems complexity will be representative of the identified systems to ensure memory access patterns, computations and algorithmic procedures, such as back-propagation can be analyzed for performance.

The systems recreated will be similar to those described in [Kri12][Qiu13][Mni13]. These systems will be simulated in either matlab, C++ or Python and the code will be instrumented to extract memory access behavior and processing "hot-spots". Where possible, this work will attempt to gain access to current work such as that described in [Qiu13] and [Aba15]. This data will be used to research designs to support the objectives outlined in section 3.

1. any research options maintain high bandwidth utilization
    - research data structures and memory organizations to maintain high bandwidth utiliza-

tion during all algorithm phases

  - for example, consider "caching" mechanisms in management layer to ensure coalesced accesses

2. cases where low memory bandwidth utilization is encountered ensure the performance exceeds that of a similar CPU or GPU implementation
    - in cogent confabulation knowledge base access, examine access patterns to optimize matrix layout, storage methods and hash functions

Most of the digital design will be implemented in Verilog HDL and synthesized and routed in an appropriate technology node. A system verilog verification environment will be used to verify the functionality of the design and measure the system performance. If necessary, a literature review will be performed to ensure the feasibility of the stack bus performance requirements. The design will be analyzed for area and power dissipation and if necessary scaled for comparison to state-of-the-art. If necessary, the DRAM analysis may be supported using the work from [Par15].

Where necessary, subsets of the simulation environments will be ported to GPU systems for performance comparisons.

Future Work

1. Consider issues associated with connecting more than one of these 3D modules into a larger network. This work may consider the impact of partitioning a network across multiple devices and issues associated with communicating neuron state information between modules.
2. Translate each ANNs data structures to a C++ library

BIBLIOGRAPHY

[Aba15] Abadi, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[Boj16] Bojarski, M. et al. "End to End Learning for Self-Driving Cars". *arXiv preprint arXiv:1604.07316* (2016).

[Che14] Chen, T. et al. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning". *ACM Sigplan Notices*. Vol. 49. 4. ACM. 2014, pp. 269–284.

[Che16] Chen, Y.-H. et al. "14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks". *2016 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE. 2016, pp. 262–263.

[Tez] *DiRAM4-64Cxx Cached Memory Subsystem*. Rev. 0.04. Tezzaron Semiconductor. 2015.

[Esm05] Esmaeilzadeh, H. et al. "NnSP: embedded neural networks stream processor". *48th Midwest Symposium on Circuits and Systems, 2005.* IEEE. 2005, pp. 223–226.

[Far11] Farabet, C. et al. "Neuflow: A runtime reconfigurable dataflow processor for vision". *Cvpr 2011 Workshops*. IEEE. 2011, pp. 109–116.

[Fra16] Franzon, P. D. et al. *Hardware Acceleration of Sparse Cognitive Algorithms*. FA8650-15-7518. AFRL. North Carolina State University. 2016.

[GH11] Galal, S. & Horowitz, M. "Energy-efficient floating-point unit design". *IEEE Transactions on Computers* **60**.7 (2011), pp. 913–922.

[Gok14] Gokhale, V. et al. "A 240 g-ops/s mobile coprocessor for deep neural networks". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 682–687.

[Jou17] Jouppi, N. P. et al. "In-datacenter performance analysis of a tensor processing unit". *arXiv preprint arXiv:1704.04760* (2017).

[Kim16] Kim, D. et al. "Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory". *Proceedings of ISCA*. Vol. 43. 2016.

[Kri] Krizhevsky, A. et al. *ImageNet Classification with Deep Convolutional Neural Networks*. `http://image-net.org/challenges/LSVRC/2012/supervision.pdf`. Accessed: 2016-08-30.

[Kri12] Krizhevsky, A. et al. "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[KO11] Kyo, S. & Okazaki, S. “IMAPCAR: A 100 GOPS in-vehicle vision processor based on 128 ring connected four-way VLIW processing elements”. *Journal of Signal Processing Systems* **62**.1 (2011), pp. 5–16.

[Liu12] Liu, Y. et al. “A compact low-power 3D I/O in 45nm CMOS”. *2012 IEEE International Solid-State Circuits Conference*. IEEE. 2012, pp. 142–144.

[Mad14] Maddison, C. J. et al. “Move evaluation in go using deep convolutional neural networks”. *arXiv preprint arXiv:1412.6564* (2014).

[Mni13] Mnih, V. et al. “Playing atari with deep reinforcement learning”. *arXiv preprint arXiv:1312.5602* (2013).

[Par15] Park, J. B. *Three-Dimensional DRAM Area, Timing and Energy Model*. ECE Dept., North Carolina State University, Box 7911, Raleigh. Box 7911, Raleigh, NC, 27695-7911, 2015.

[Pat14] Patti, R. “2.5 D and 3D Integration Technology Update”. *Additional Papers and Presentations* **2014**.DPC (2014), pp. 1–35.

[Qiu13] Qiu, Q. et al. “A parallel neuromorphic text recognition system and its implementation on a heterogeneous high-performance computing cluster”. *IEEE Transactions on Computers* **62**.5 (2013), pp. 886–899.

[Sch14] Schabel, J. et al. *Predictive energy-Per-Op scaling for exploring the design space*. ECE Dept., North Carolina State University, Box 7911, Raleigh. Box 7911, Raleigh, NC, 27695-7911, 2014.

[Tai14] Taigman, Y. et al. “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[Tec] Techpowerup. *NVidia Tesla K20c*. `https://www.techpowerup.com/gpudb/564/tesla-k20c`. Accessed: 2016-09-08.

[Tsa08] Tsai, Y.-F. et al. “Design space exploration for 3-D cache”. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **16**.4 (2008), pp. 444–455.

APPENDICES