# NeuroCube: Low-Power Scalable Neuromorphic Architecture Using Hybrid Memory Cube


## Abstract

*This document serves as a sample for submissions to MICRO 2014. We provide some guidelines that authors should follow when submitting papers to the conference.*


## 1. Introduction

THIS PART IS FOR INTRODUCTION

1. Machine Learning is important application in RMS (Recognition/Mining/Synthesis)

2. Since its application scope increases and handle the big data, input domain and complexity increases dramatically

3. NN is one of the well known algorithms for ML, and large scale of NN is required to solve complex ML problem.

4. Large scale NN means more neurons, more layers, and more weight. Therefore, it requires massive computations and data storage

5. For given hardware, there are two methods to solve this problem.

5-1. First method is algorithm multiplexing, which divides the problem into sub-problems, and conquer them. It's hardware overhead/requirement is low, but it can't solve the problem requiring global optimization, for example, hole filling.

5-2. Another method, which we adapt, storing all neural network in 'external' memory, which can have high capacity with long latency and share some computational units in given hardware.

6. Interesting part is that operation of neural network is determined by data access sequence of memory since most of digital NNs are based on similar operations such as multiply-and-accumulate or sigmoid function. Therefore changing the data access sequency can implements different neural networks with given hardware.

7. Since entire neural network is stored in memory, accessing memory is the key operation in neural network.

8. Based on given neural network, data sequence is not random sequence, but it is predetermined by algorithm. Therefore, dedicated memory controller generating data request and address can be implemented for each neural entwork. This controller is the key element for neural network function.

9. This controller will make memory deliver the data to multiple processing engines without request to operate multiple PEs in parallel without stall due to memory access (full utilization).

10. For the application requiring sharing the data between PEs, network-on-chip is required to connect multiple PEs and multiple Memory. As data sequence is determined by neural network type, network traffic is also determined by neural network.

11. In this paper, we will investigate architecture for digital neuromorphic hardware including multiple processing engines, cache memory, network-on-chip, external memory, and memory controller.

12. Under the given external memory and processing engines, optimal number of cache memory and processing engines will be studied to maintail full utilization of PEs.

13. For different NNs, memory controller is implemented, and its perforamnce will be analyzed for different applications.

14. With given architecture, network traffic pattern on NoC will be studied and performance of entier system will be discussed.

15. To achieve high external memory bandwidth, hybrid memory cube could be appropriate system for neuromorphic architecture. Its advantage will be analyzed for different NNs.

## 2. Previous Work

In this section, we will introduce recent machine learning techniques using different types of neural network and hardware implementation based on ASIC or FPGA.

### 2.1. Neural network for machine learning

Artificial Neural network (neural network) which mimics biological neural networks is the most interested topic in machine learning. It is composed of set of neurons and a single neuron may connect to set of other neuron or all neurons in the network. This network could be composed of multiple layers –> Deep NN?

Main benefits of Neural network are the characteristic of computation process in biological neural network, such as parallelism, nonlinearity, error-robustness, and learning (plasticity). Therefore neural networks can be classified based on these different characteristics [1]. According to Section 6 in [1], neural network can be classified based on different characteristics such as target application, connectivity, flow of network. In terms of hardware design for neural network, we should note that connectivity and flow of network are most important factors of neural network since these factors are directly related to the interface in the system (for example, bus-interface or network-on-chip (NoC)).

**Connectivity**

A single neuron can connect with all neurons in the network or set of neurons. Hopfiled network is the network with fully connected two layers [5]. Full connectivity means outputs of all neurons (including itself) are fed into as input to a single

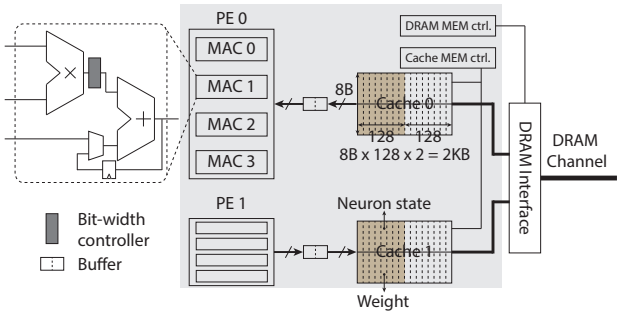


**Figure 1: Full system MLP ver2 from 15-DAC.**

neuron. Due to its complexity of network, it is impractical to implement in real hardware.

In stead of fully connection, cellular neural network has local connection; a single neuron connects to the set of neurons in the neighborhood [3]. A single neuron is connected with $(2r+1)^2$ neurons when radius of neighborhood is $r$. For simple image processing (such as edge detection[2], garbor filter [4]), $r = 1$ is enough and $r = 3$ is still enough for associative memory operation (reference is required).

**Network flow**

1. Why do we need different NNs other than Convolution NNs? –> Deep learning requires 'different' type of neural network layers

2. What is the characteristic of diff. NNs (pro - cons) or appropriate application

Can we compute number of basic computations such as multiplications or additions for some applications?

Maybe table to compare diff NNs will be good.

### 2.2. Algorithm multiplexing

Describe algorithm multiplexing

### 2.3. Hardware sharing?

Describe hardware sharing and compare with algorithm multiplexing

### 2.4. Neuromorphic hardware implementation

Following hardware sharing

## 3. Reconfigurable Neuromorphic Architecture

text

### 3.1. External memory

text

### 3.2. Cache memory

cache memory

### 3.3. Processing elements

As we explained before, main basic operations for NNs are multiplication, addition, activation function, sampling ... So we will prepare all functions to cover different neural networks. Then controlling the data from memory to processing elements can implement different neural networks.

### 3.4. Network-on-chip

text

### 3.5. Memory architecture

As stated in Section 2.4, we design the digital MLP hardware fully utilizing MAC units. To make the MLP hardware scalable to larger network size, the on-chip cache memories should interface with an external DRAM memory (Fig. 1). However, the DRAM access latency sacrifices the parallelism provided by MLP algorithm. We have to choose the maximum number of MAC units in each PE at design time depending on the memory architecture to achieve full utilization. To begin with, the read latency of DRAM can be defined as (MLP operation is read-intensive, thus, we focus on the read latency):

$$DRAM_{lat} = tRCD + tCAS + tRP + \left\lfloor \left( \frac{page\_size}{bit\_width_{bus}} - 1 \right) / 2 \right\rfloor, \quad (1)$$

where $tRCD$ is the number of clocks required to access each row, $tCAS$ is the column access latency, and $tRP$ is the required clocks between row precharge and activate. Each parameter in (1) is determined by the specification of the external memory. Also, we assume that the burst length can be modified to read the whole page after a single row access (since MLP has a memory access pattern with strong spatial coherence) with low column access latency. Let's consider we have an external memory which can be accessed at $f_{DRAM}$ (= $1/CLK_{DRAM}$). Here, $CLK_{DRAM}$ represents the DRAM I/O bus clock cycle. We then choose PE to operate at $f_{PE}$ and SRAM memories to operate at $f_{SRAM}$. Then, $DRAM_{lat}$ can be rewritten by using $f_{PE}$.

$$DRAM_{latPE} = DRAM_{lat} \times \frac{f_{PE}}{f_{DRAM}}. \quad (2)$$

As shown in Fig. 1, the half of the cache (equivalent to the page size of DRAM) has to be initially filled up to fetch in the data into each PE (a group of MAC units) in parallel. The 8Byte is the data size required at each MAC unit for matrix-vector multiplication. After initialization, each cache is updated with round-robin scheduling in parallel with the data read from the cache to PE (Fig. 1). To achieve the full utilization, the following condition has to be satisfied.

$$\frac{\#Lines}{m} = DRAM_{latPE} \times n, \quad (3)$$

where m is the number of MAC units in a PE, n is the number of PEs, and #Lines is the (page_size/word_size). The left-hand side of (3) defines the clock cycles to use all data stored in the half portion (shaded) of the cache. The right-hand side of (3) represents the clock cycles to prepare data for

all PEs connected to each channel (filling up the other half). *Equation (3) determines the total number of MAC units* $(n \times m)$ *that can be designed without sacrificing the full utilization.* The question then arises how to set the number of MAC units per PE (= '$m$'). This is determined by the clock frequency ratio between the cache and the PE ($f_{SRAM}/f_{PE}$). At each $CLK_{SRAM}$, the cache provides 8B data to the buffer. This may be continued for $k \times CLK_{SRAM}$, which allows total 8kB for a PE to use at one $CLK_{PE}$ (If the frequency ratio is 4, then the data size the cache can provide within one $CLK_{PE}$ becomes 32B). Thus, the number of MAC units per PE is constrained by

$$m = \frac{f_{SRAM}}{f_{PE}} = \frac{CLK_{PE}}{CLK_{SRAM}}, \tag{4}$$

This relation naturally leads to the decision of the number of PEs, $n$. Since the total number of MAC units is determined by (3), sets of $(m,n)$ can be obtained. Here, as reducing the number of cache memories results in smaller area and lower power, selecting smaller $n$ is a better design option with a given memory architecture. This is for one channel case so the total number of MAC units $(NUM_{MAC})$ with multi-channel memory becomes:

$$NUM_{MAC} = \#Channel \times (n \times m). \tag{5}$$

## 4. Traffic Analysis on Network-on-Chip

For different neural networks, our system described in previous sesction is simulated to operate given testcase for each neural network. For this system, standard DDR3-SDRAM (low bandwidth specification) is assumed as external DRAM.

## 5. Hybrid Memory Cube

hmc introduce

### 5.1. Architecture of HMC

hmc introduce

### 5.2. Intranetwork of HMC through logic die

hmc introduce

### 5.3. Internetwork of HMC through high speed links

hmc introduce

### 5.4. Neurocube using Intranetwork of HMC

Processor-in-Memory (PIM) design hmc introduce Area/power/thermal limitation

## 6. Traffic Analysis of NeuroCube

Introduce improvement of NeuroCube with HMC with high bandwidth

## 7. Conclusions

conclusions bla bla bla [1].

## References


[1] I. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[2] L. O. Chua and T. Roska, *Cellular neural networks and visual computing: foundations and applications*. Cambridge University Press, 2002.

[3] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *Circuits and Systems, IEEE Transactions on*, vol. 35, no. 10, pp. 1273–1290, 1988.

[4] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks—a review," *Pattern recognition*, vol. 35, no. 10, pp. 2279–2301, 2002.

[5] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.