

A 3DIC system to aid in the acceleration of systems that employ multiple instances of artificial neural networks

Status Apr 2017

Lee B. Baker

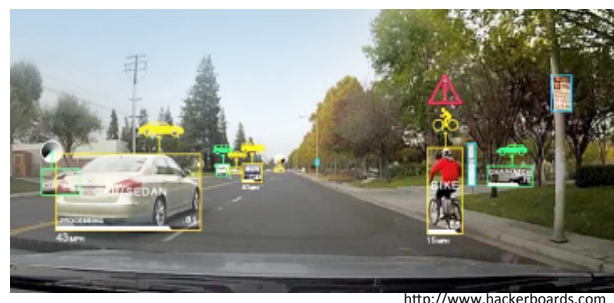
Advisor: Paul Franzon

Introduction

- Recently Artificial Neural networks (NN) have demonstrated superior performance in classification and function approximation

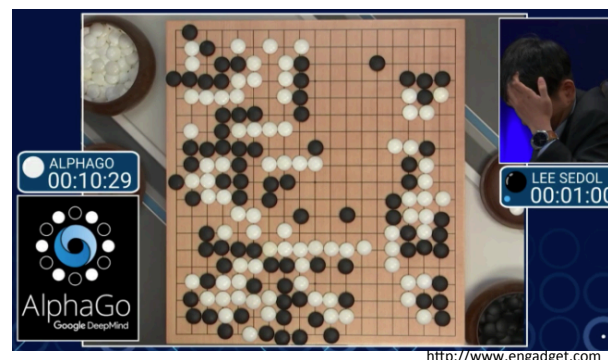
Deep Neural Networks (DNN) have been very successful in image processing applications [Kri12]

- it is anticipated that DNNs will be employed more and more in self-driving cars



Reinforcement learning is gaining popularity

- alphaGo employed reinforcement learning with deep neural networks [Mad14]



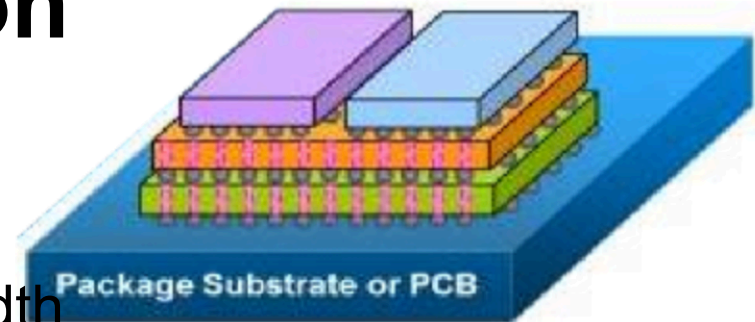
Introduction

- Many applications will and do require multiple NN engines running at or near real-time
 - car or drone scanning in many directions during navigation
 - cloud server processing multiple images when searching
 - reinforcement learning requires multiple state-action value calculations during optimal policy searches
 - text recognition system processing multiple pages of text

Problem

- Many applications will/do have power, space and weight limitations
 - drone
 - server rack has building cooling limitations
- Useful Neural Networks are big
 - lots of memory and computations → bandwidth
 - local SRAM consumes too much silicon
- To achieve near or at real-time processing, current solutions require high power and high real-estate
 - GPU solutions have large real-estate and power requirements (~100-200W)
 - ASIC's target smaller network sizes and/or specific NN's
 - Both have memory bandwidth limitations

Solution



<http://www.semiwiki.com>

- 3DIC Architecture
 - reduces energy and area
 - increase connectivity and bandwidth
- 3D-DRAM
 - provide high bandwidth and large storage
 - minimize dependency on SRAM
- Data Structures
 - data structures to ensure neural network data can be accessed efficiently
 - maintain constant access to DRAM to avoid dependency on local SRAM
- Specialized processing layers
 - provide special functions to aid in acceleration of target neural networks

Contribution

- Research uniquely organized 3DIC DRAM
 - research bank and page organization for a family of neural networks
 - maintain efficient read and writing
- Research and propose Data Structures to exploit DRAM capacity
 - in conjunction with DRAM organization
 - data structures organized to allow efficient streaming of data directly to processing layers
- Propose 3D architecture to take advantage of DRAM bandwidth
 - memory management to manage configuration and algorithm operations and coalesced accesses
 - processing layer contains special functions for a family of NN's

Outline

1. What are artificial Neural networks
- 2. Target Neural networks**
3. State-of-the-art
4. Proposed Architecture
 - Problem
 - Solution
5. Preliminary Work
 - 3DIC DRAM and data structures
 - 3D Stack Bus
 - Processing Layer
6. Research Plan
7. Summary

Target ANN Type's

- This work will focus on the following ANNs :
 - Deep Neural Network Classifiers
 - ANN in Reinforcement Learning
 - Cogent Confabulation
 - Brain-state-in-a-Box
- Our research will involve analyzing memory access and processing requirements

Outline

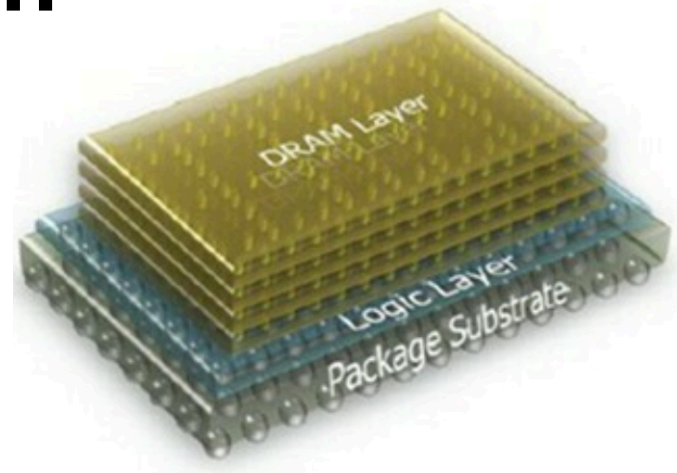
1. What are artificial Neural networks
2. Target Neural networks
3. State-of-the-art
4. **Proposed Architecture**
 - Problem
 - Solution
5. Preliminary Work
 - 3DIC DRAM and data structures
 - 3D Stack Bus
 - Processing Layer
6. Research Plan
7. Summary

Problem

- System solution requiring multiple NN's
 - Consider a system that requires ~8 DNNs^[Boj16]
 - ~16Gb of memory
 - ~100W per GPU suggests >800W total power
- Real-Time
 - Image needs to be processed ~16mS (60 frames/sec)
 - Allowing ~10mS for classification requires ~1700GFLOPS¹
 - Required Memory bandwidth ~54Tbps
- Current 3D-DRAM memory technology bandwidth
 - **HMC – SERDES~2Tbps**
 - **HBM – wide DDR ~2Tbps**
 - **Tezzaron DiRAM4 – wide DDR – 4Tbps**

Solution

- Utilize 3DIC technology



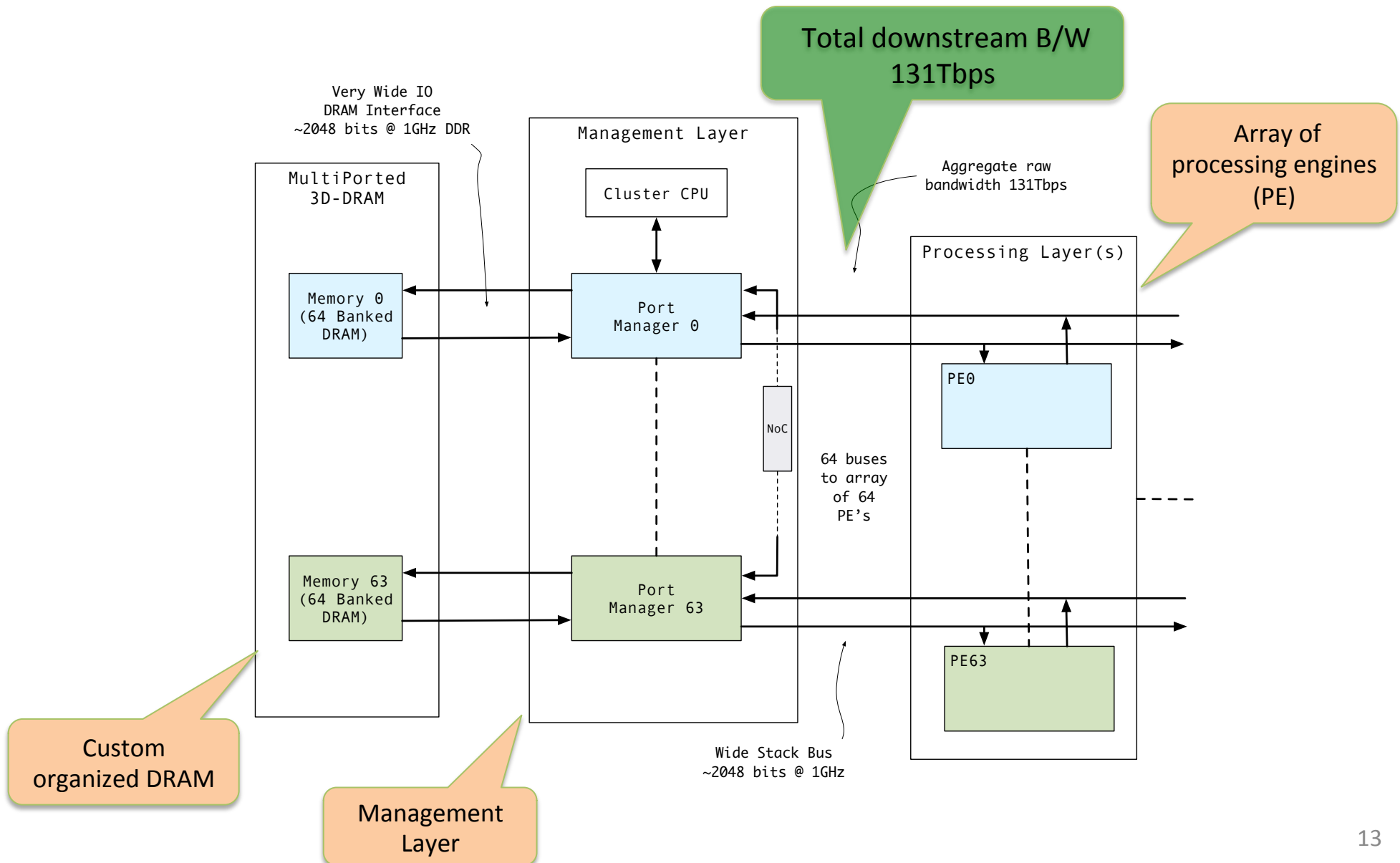
<http://www.electroiq.com/>

- Architecture includes:
 - Custom organized 3D-DRAM specification
 - Data Structures specific to each ANN
 - Management Layer for configuration and control
 - Processing layer(s) targeted toward a family of ANNs
 - array of processing engines (PE) with special streaming functions
 - PE will include small CPU to process non hot-spot functions and provide some generality

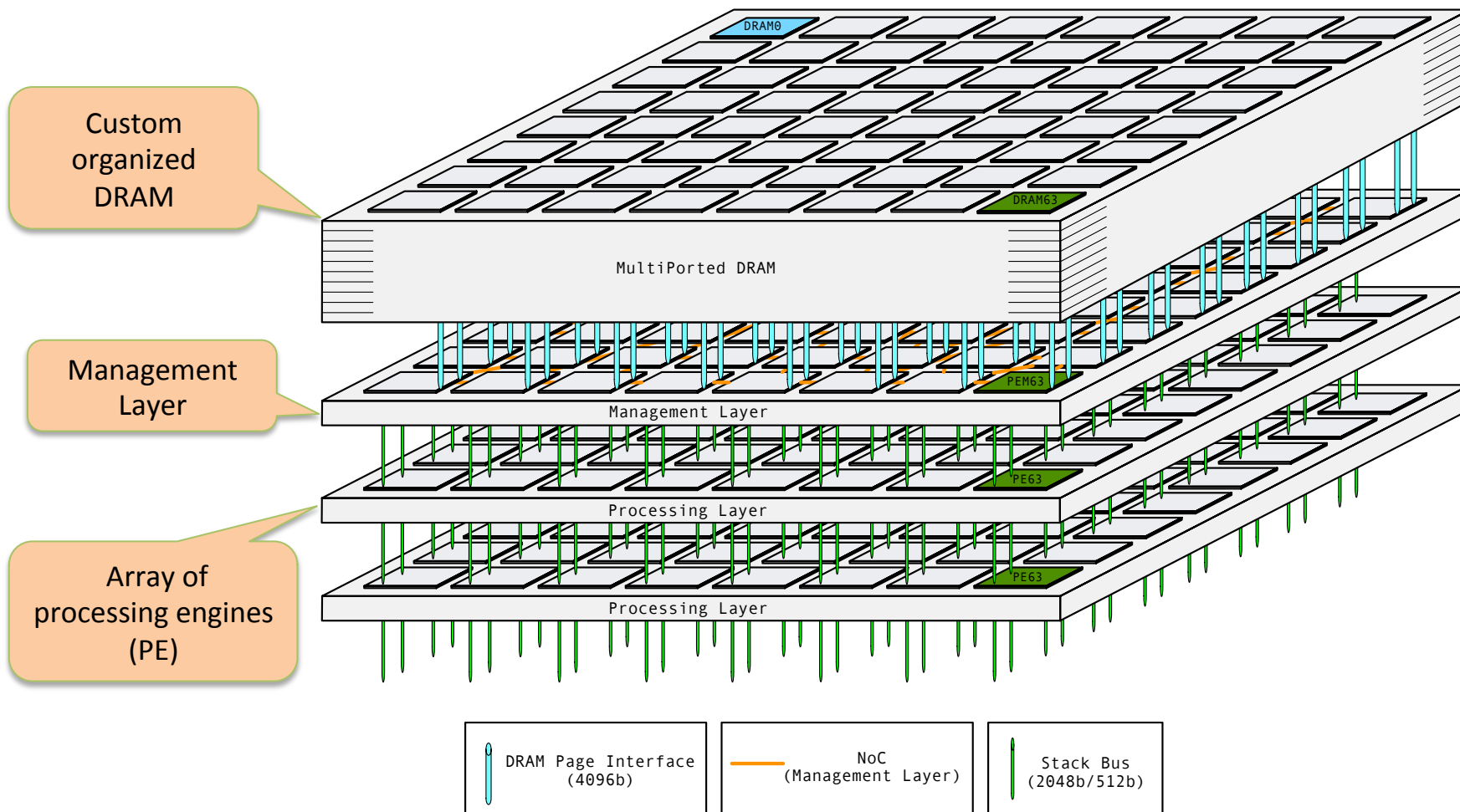
Outline

1. What are artificial Neural networks
2. Target Neural networks
3. State-of-the-art
4. Proposed Architecture
 - Problem
 - Solution
5. **Preliminary Work**
 - 3DIC DRAM and data structures
 - 3D Stack Bus
 - Processing Layer
6. Research Plan
7. Summary

Solution Block Diagram

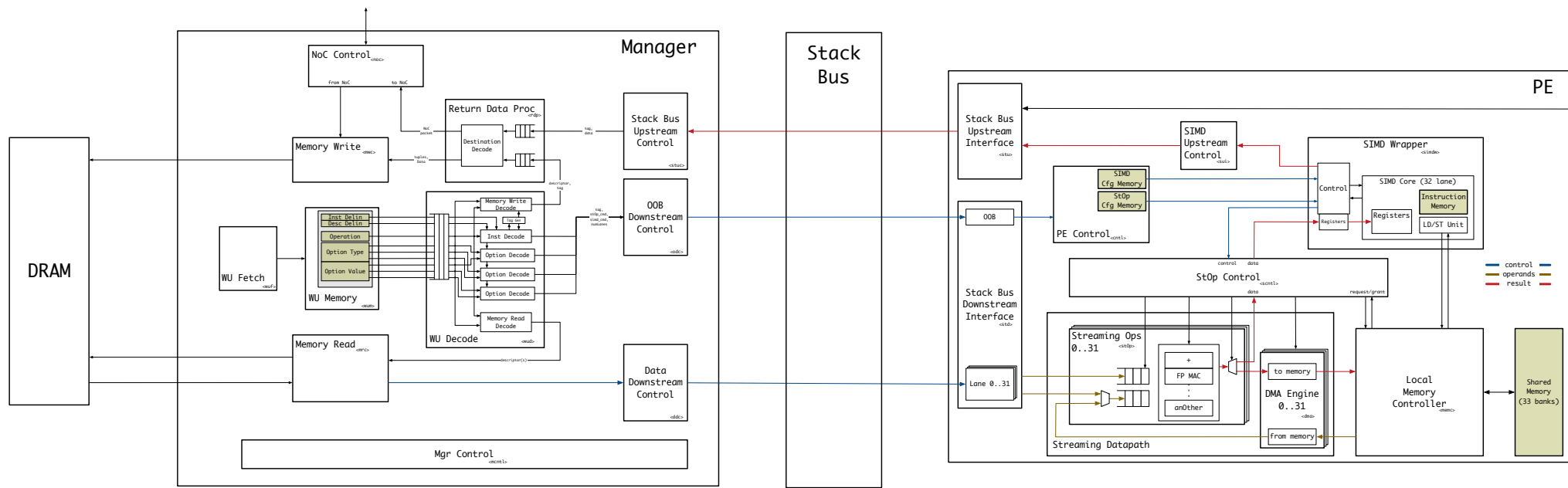


3D Configuration



System Column

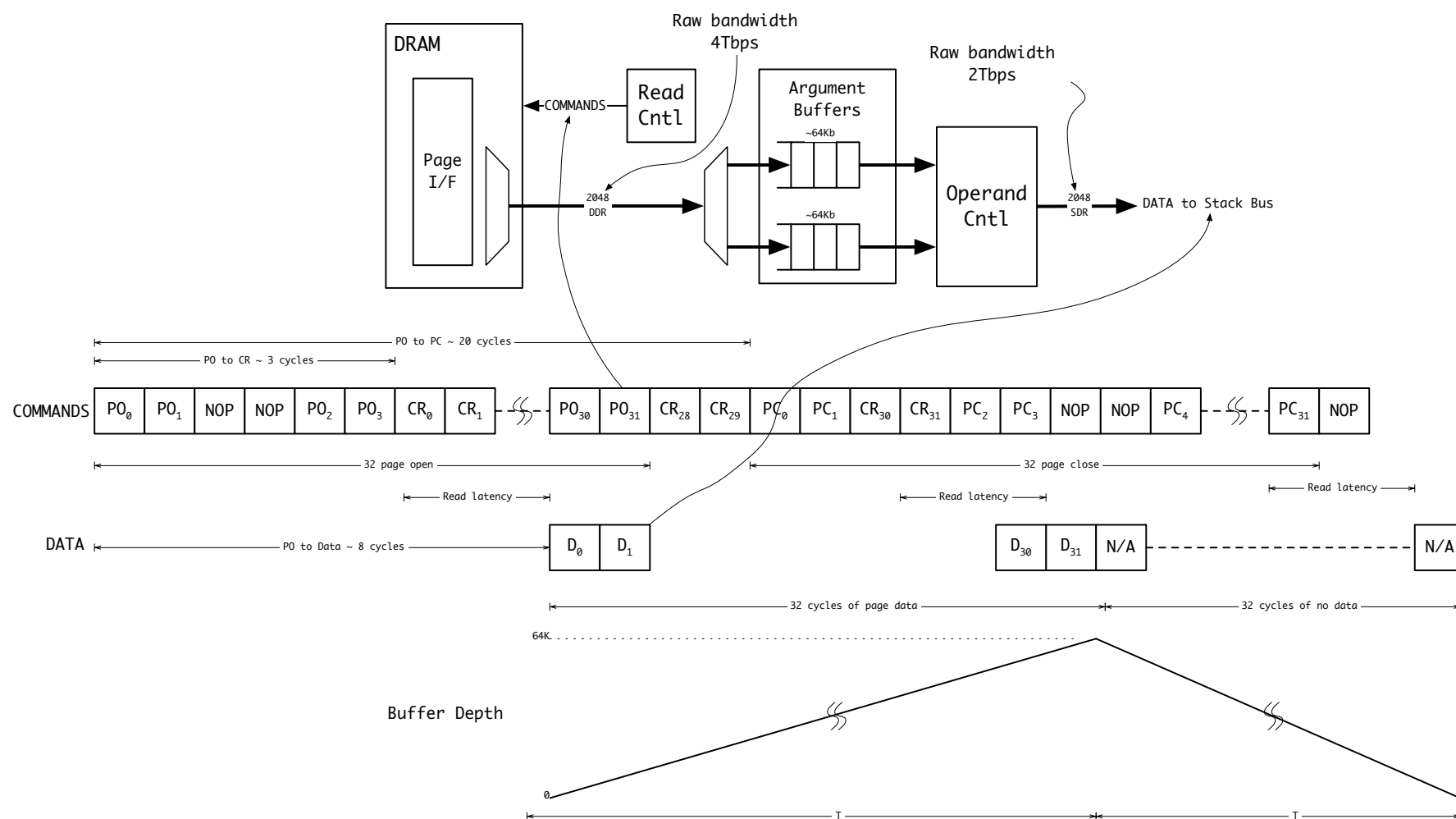
- A processing column includes:
 - DRAM port
 - Manager
 - process set of instructions which implement a NN
 - PE
 - take data from DRAM via stack bus
 - operate on data without local storage
 - silicon focused on processing required not storage



DRAM Ports, Bus Widths and PE Array Size

- Determined how many PE's required to support bandwidth needs
 - based on available GOPS, current DRAM technology and reasonable operating frequency
- Size data width to provide enough operands to each execution lanes (EL)
- Estimated 64-port memory each supporting a processing engine with 32 execution lanes
- Review DRAM operations to stream data for power dissipation

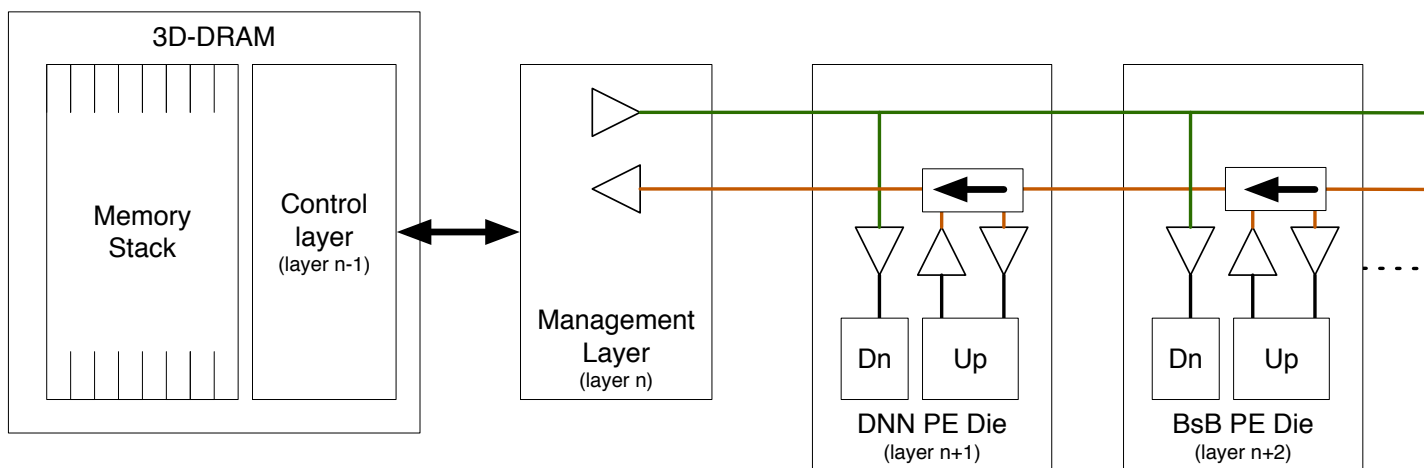
Example DRAM Access Sequence



3D Stack Bus

Stack Bus

- Each processing column has a downstream and upstream stack bus
 - Downstream 2048^1 bit running at 1GHz providing 131Tbps raw system bandwidth
 - Upstream bus 512^2 bits running at 1GHz
 - result bandwidth $\sim 1/100^{\text{th}}$ of operand bandwidth



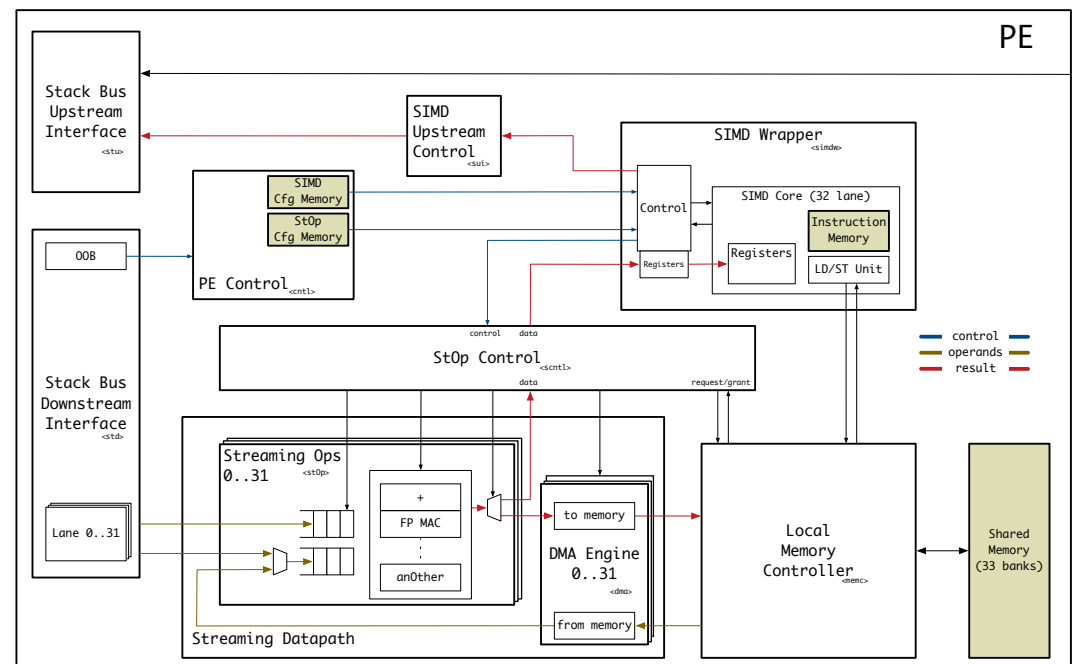
Stack Bus

- Downstream and upstream buses
- Buses carry control packets and data packets
 - packets will be multiple clock cycles
- Control packets used to configure PE
 - special function and SIMD operations
 - result information
- Data Packets will carry arguments for special functions
 - data to will be multiple cycles of two 32-bit words

Processing Layer

Special Functions in PE

- Special Functions are designed to operate on data from the Stack bus
 - operate at line rate from downstream bus
 - perform computations directly as data is read from memory
 - avoid need for large local SRAM in PE
 - result is passed to “small” local memory or registers for later processing by SIMD
 - result sent to upstream bus



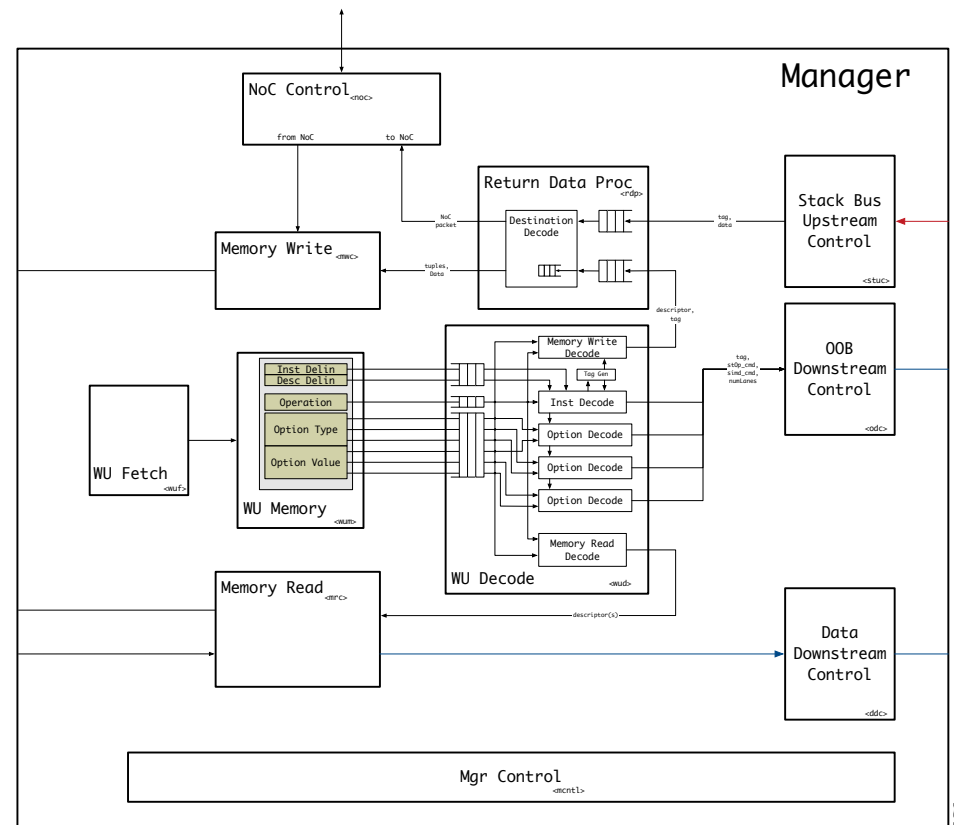
PE Status

- FMA streaming operation functionality coded
- SIMD has yet to be integrated
 - a wrapper around SIMD allows system to operate normally except SIMD doesn't process neuron outputs
- SV environment generates PE configuration packets and streams neuron weights and inputs to PE
 - Result verified by examining result packet on upstream stack bus

Management Layer

Manager

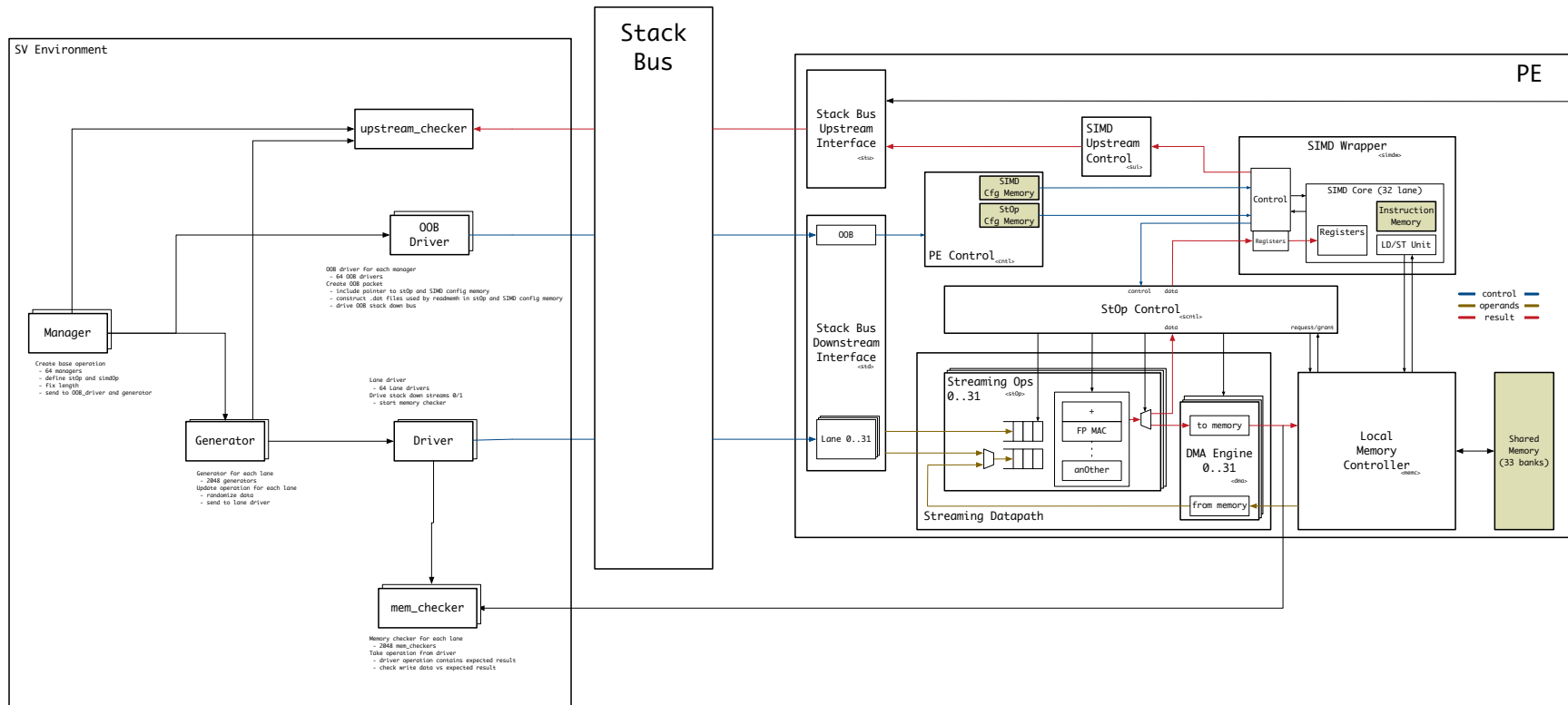
- Ensure data can be streamed efficiently from DRAM
- Operates on a custom instruction set
 - instructions describe all that is necessary to process a group of neurons
 - avoid need for large local SRAM in PE
- System requires data duplication
 - NoC designed to multicast neuron activations to other managers



Manager Status

- Basic functionality except Memory read and Write are coded
- Manager instructions generated from python script
 - instructions include a group of descriptors that specify:
 - PE operation
 - source of weights and inputs
 - where to write results
 - python script generates readmem files
- SV
 - environment still generates neuron weights and inputs to PE
 - Manager code generates configuration packets from instructions
 - Manager code takes results from upstream stack bus and constructs NoC packets for writes to local memory and other manager memory

Verification Block diagram



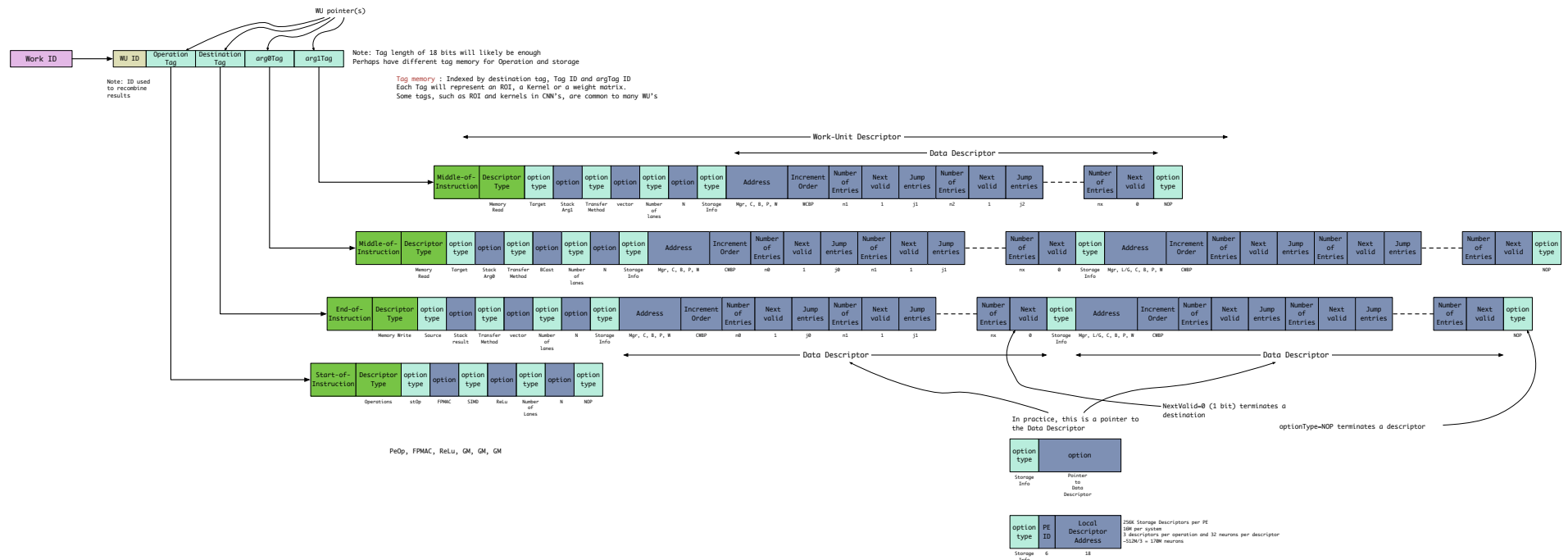
WU/Instruction Format

- Instruction must describe operations for processing a group of neurons
- Describe operations performed for a group of neurons
 - convolution on weights and input
 - Relu or Sigmoid to generate activation
 - how to read weights
 - how to read inputs
 - how to save neuron activations
- Instruction decoder generates:
 - OOB configuration packets to PE to configure SIMD and PIM stOp
 - Information to return Data processor describing where result should be sent
 - RDP sends result data along with write descriptor pointers to local manager or other managers over NoC
 - Read descriptors pointers to memory read modules to send operation arguments down stack bus to PE

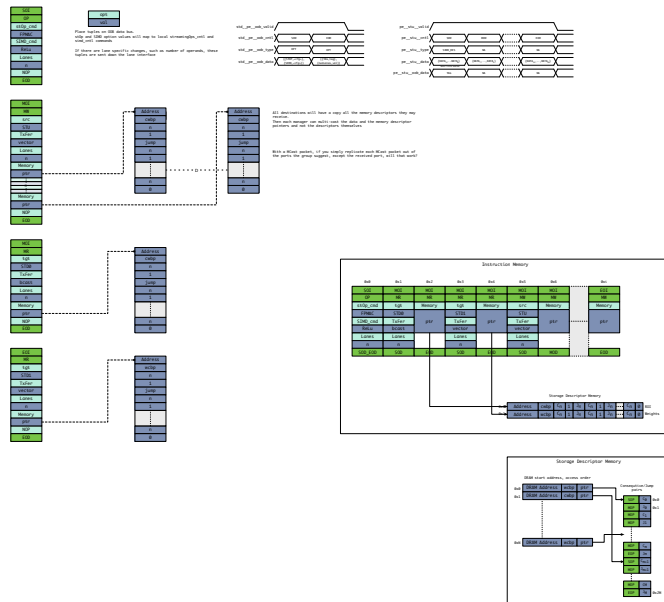
WU/Instruction Format

- Instructions formed from a set of descriptors
- Descriptors for:
 - Operations
 - memory Read
 - memory Write
 - access method
- Operation descriptor
 - defines SIMD and stOp operations
 - SIMD and StOp operation stored in PE instruction memory
 - descriptors point to instructions in PE IM
- Memory read and write Descriptor
 - sets start address
 - describes how address is accessed
 - bank/page/word increment order
 - how data is transferred to stack bus and on to PE
 - memory descriptor points to an access descriptor that is stored in manager memory
 - expect commonality amongst access methods so many descriptors may point to same access descriptor
- Access descriptor
 - not used in the actual instruction
 - expect some commonality between memory read and writes so avoid replicating access descriptor

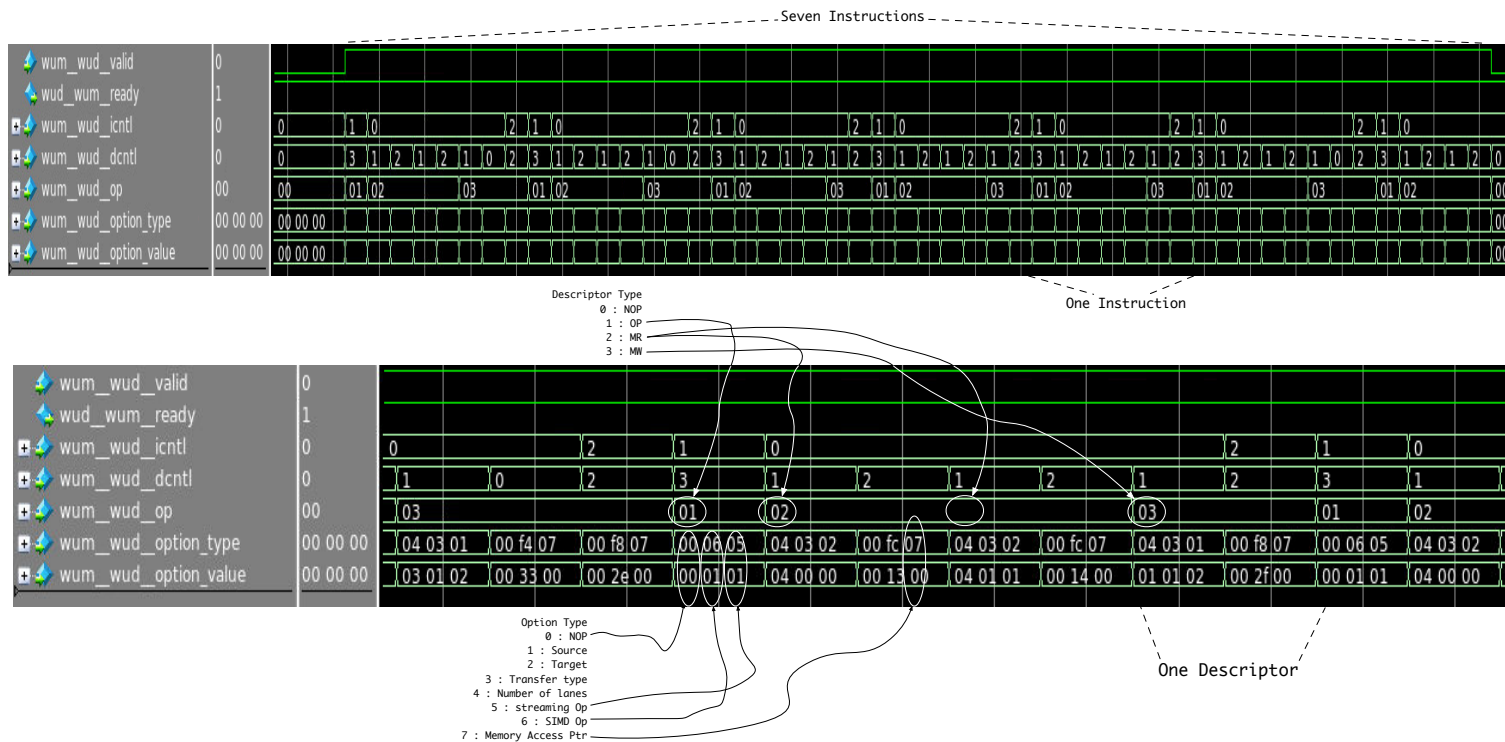
WU/Instruction Format



WU/Instruction Processing



Instruction Communication



NoC Signaling

