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

Status Apr 2017

Lee B. Baker

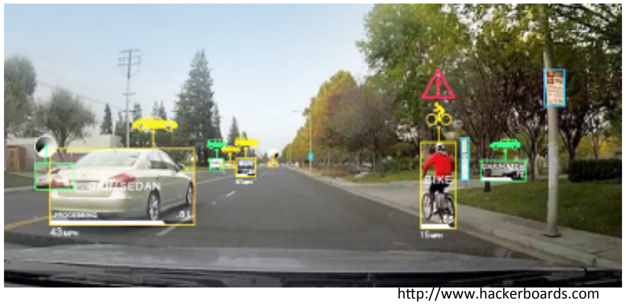
Advisor: Paul Franzon

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

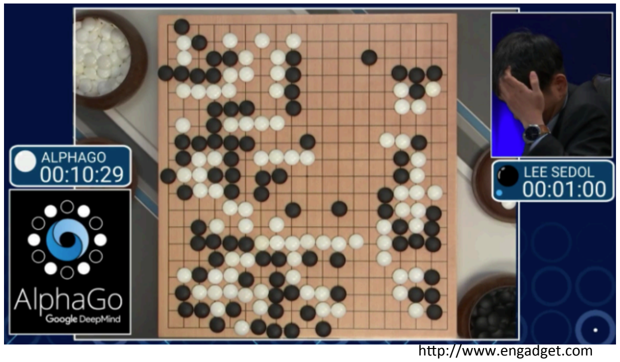


http://www.hackerboards.com

http://www.BGR.com

Reinforcement learning is gaining popularity

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



http://www.engadget.com

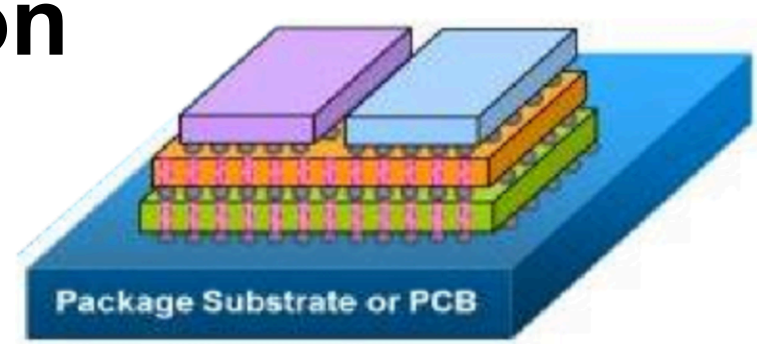
Introduction

- 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
  - locally and fully connected networks are bandwidth limited
- 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
  - ASIC's often employ SRAM which consumes a high percentage of silicon
  - 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 to allow available silicon to be focused on processing
  - eliminate NN size restrictions imposed by local SRAM
- Data Structures
  - data structures to ensure neural network data can be accessed efficiently from primary storage (DRAM)
  - 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
  - avoid local memory imposing NN size limitations
  - focus available silicon on processing requirements

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
  - employ data duplication
- 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. **<u>Proposed Architecture</u>**
   - 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 ~1700GFLOPS[1]
  - Required Memory bandwidth ~54Tbps
- Current 3D-DRAM memory technology bandwidth
  - **HMC – SERDES~2Tbps**
  - **HBM – wide DDR ~2Tbps**
  - **Tezzaron DiRAM4 – wide DDR – 4Tbps**

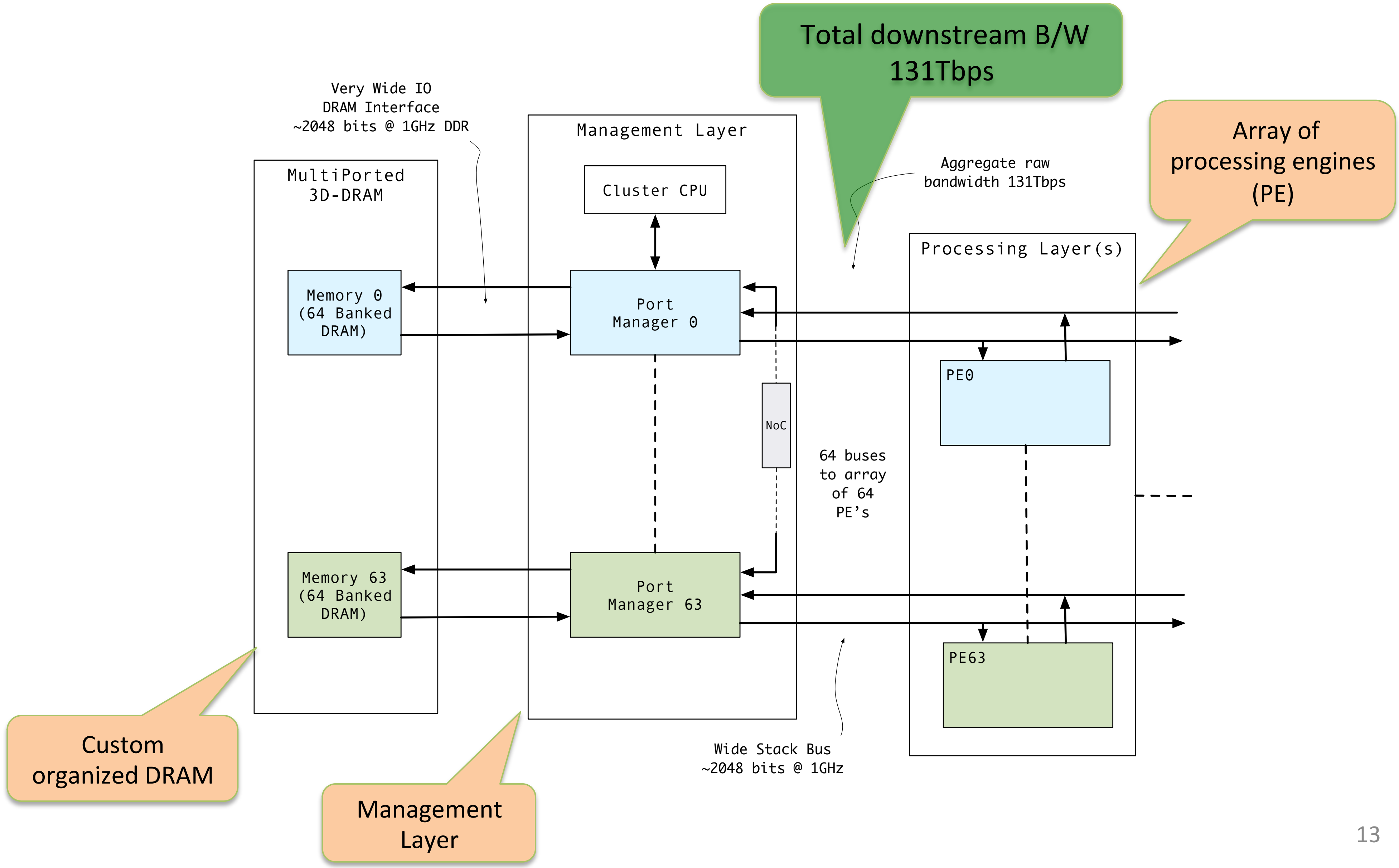
Solution

- Utilize DRAM as the primary processing memory
  - allows support of locally and fully connected NNs
  - improves NN size support with near deterministic performance
  - not optimal for NNs that take advantage of locality
- Processing Layer can be customized
  - PE layer designed to support application specific accuracy, algorithm etc.

Solution

- Architecture includes:
  - Custom organized 3D-DRAM specification
    - allows constant streaming of data to PE avoiding local storage
  - 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 includes small SIMD unit to process non hot-spot functions and provide some generality
- 3D solution contained within the footprint of available DRAM
  - allows use of TSV's for memory interface
  - PE provides 32 execution lanes within the footprint
  - minimal use of local SRAM allows PE and manager logic to fit within footprint

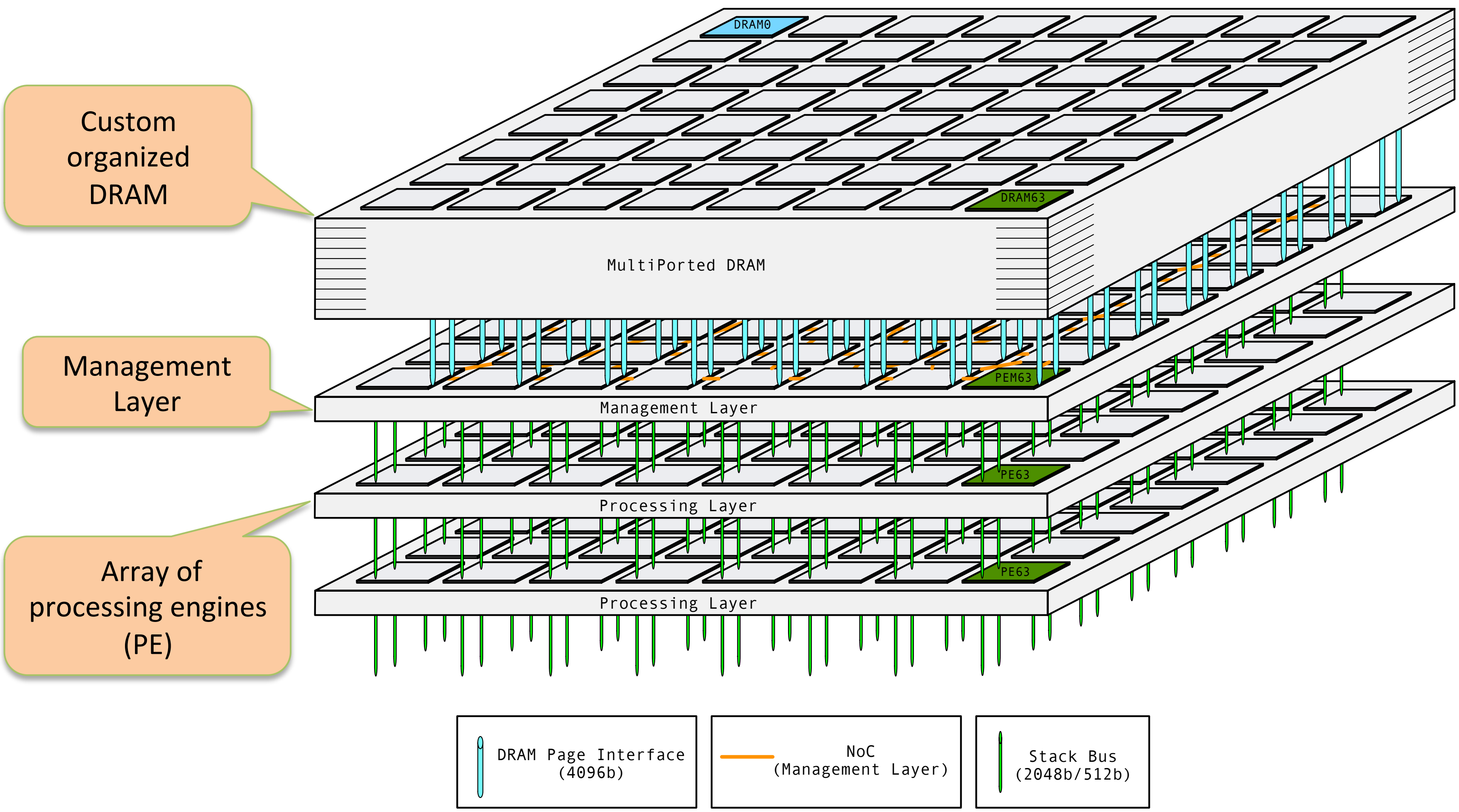
Solution Block Diagram



Potential Customers

- Google current have their own solution in the datacenter space
  - Tensor processing unit is a custom ASIC
    - large local SRAM and a large array (64K) of 8-bit MACs
    - lower DRAM bandwidth
  - High percentage of their NNs are MLP and LSTM and smaller percentage (5%) are CNNs
    - CNNs take advantage of locality and are suited to using large local SRAM
    - it may be the case that as MLP and LSTM have many fully/locally connected layers, they are impacted by a low main memory bandwidth
- 3D DRAM System could provide performance improvement in accelerating MLPs and LSTM
  - provide the bandwidth to support the bulk of the NNs processed in their datacenters
  - a custom 3D solution could provide a PE layer to support the target resolution
    - perhaps work with google to integrate 8/16-bit PE similar to their TPU

3D Configuration

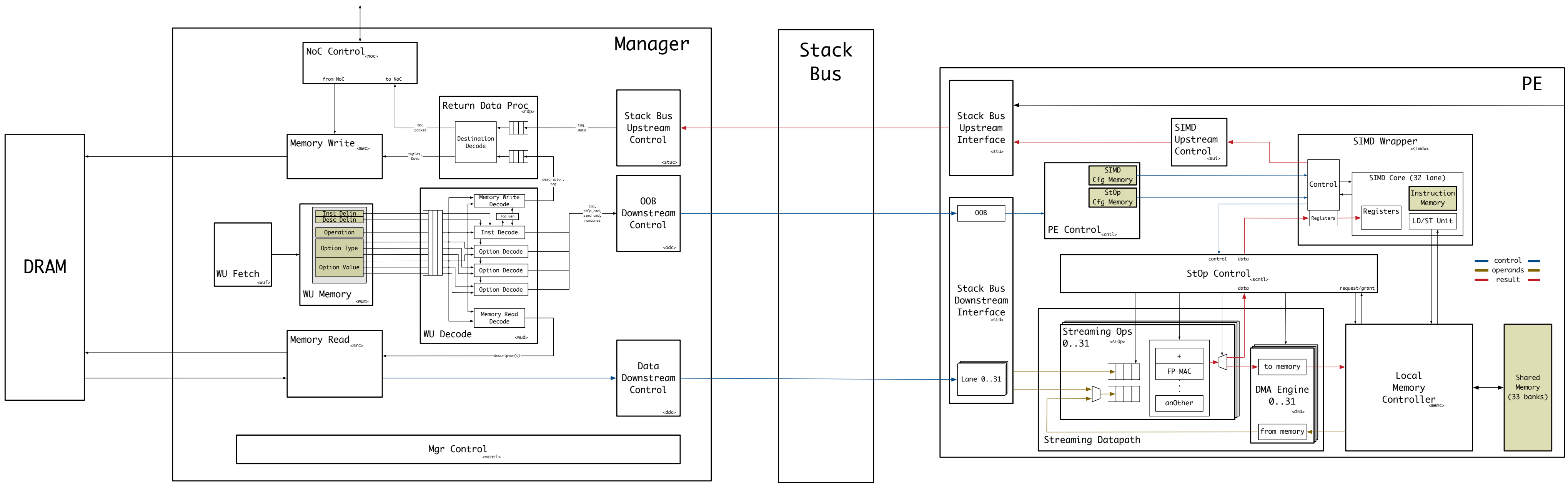


System Column

- Each column includes:
  - port into 3D-DRAM
  - Manager for configuration, DRAM reading, writing and data duplication
  - PE
- Idea is to read and write mainly from DRAM
  - no assumptions on data reuse (kernels or input)
  - input or weight reuse would be more optimal but we are providing a more flexible high bandwidth solution
- Solution includes instructions to describe operations required for a group of neurons
  - operations performed by PE
    - PE operates on two streams from manager e.g. weight/input
  - where input and weights reside in DRAM
  - where to write results including duplication to other columns

System Column

- A processing column includes:
  - DRAM port
  - Manager
    - process set of instructions which implement a NN
    - manage configuration of PE and communication of results
  - PE
    - take data from Manager/DRAM via stack bus
    - operate on data at line rate to generate neuron activations
    - silicon focused on processing required not storage

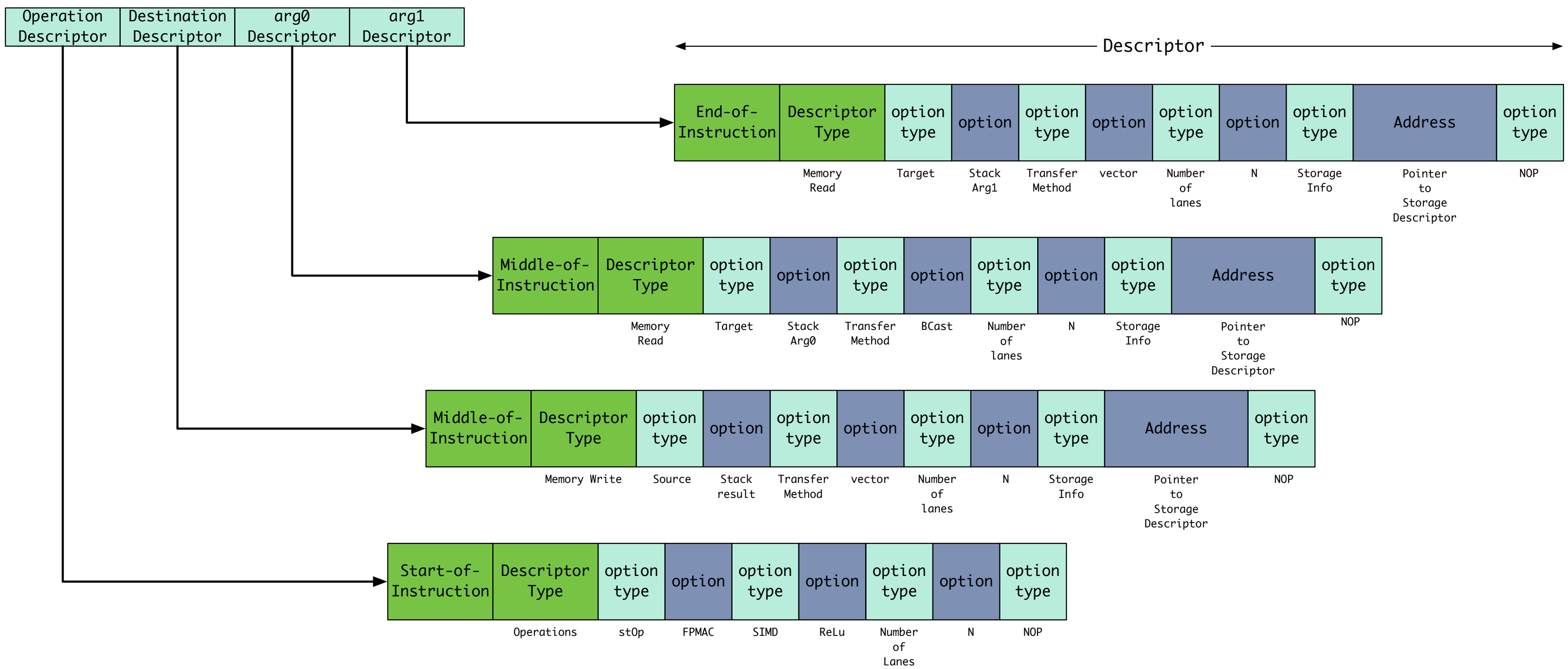


Instruction Format

- Instruction must describe operations for processing a group of neurons
  - convolution of weights and input
  - Relu or Sigmoid to generate activation
  - where and how to read weights
  - where and how to read prior activations
  - where and how to save neuron activations
- Instruction formed from four descriptors
  - PE operation
  - Weight read operation
  - Input read operation
  - Write-Back operation
- Instruction decoder generates control messages from descriptors:
  - Operation descriptors sent via OOB configuration packets to PE to configure SIMD and stOp
  - Write descriptor sent to a Return Data processor describing where result from PE should be sent
  - Read descriptors sent to memory modules to stream data to PE

Instruction

- Includes four descriptors
  – operation
  – memory read (2)
  – result write



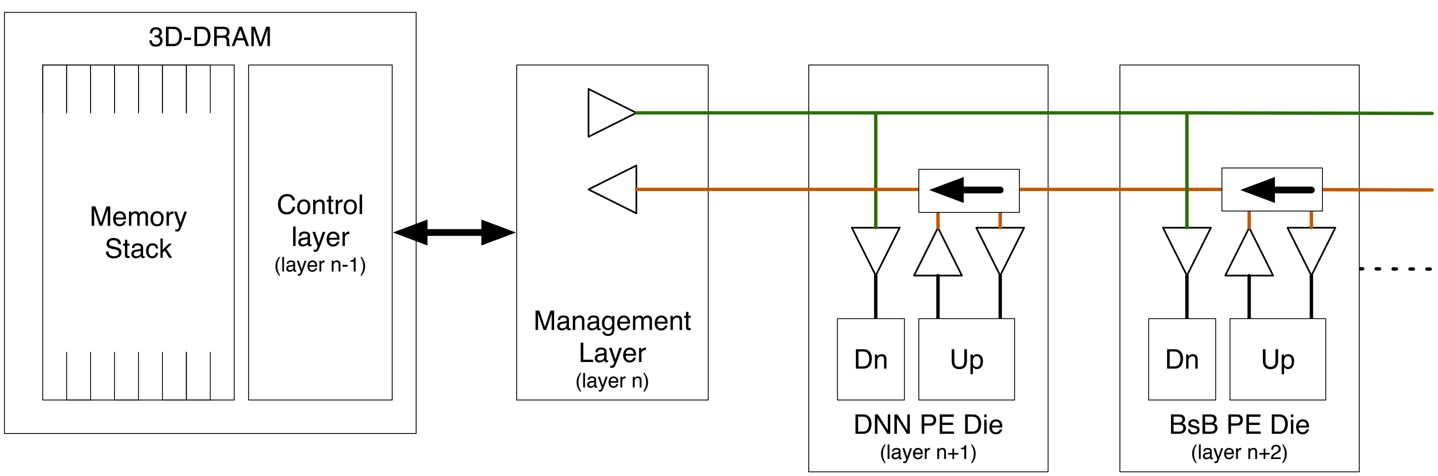
Instruction Format

- 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

3D Stack Bus

Stack Bus

- Each processing column has a downstream and upstream stack bus
  - Downstream 2048[1] bit running at 1GHz providing131Tbps raw system bandwidth
  - Upstream bus 512[2] bits running at 1GHz
    - result bandwidth ~1/100th of operand bandwidth



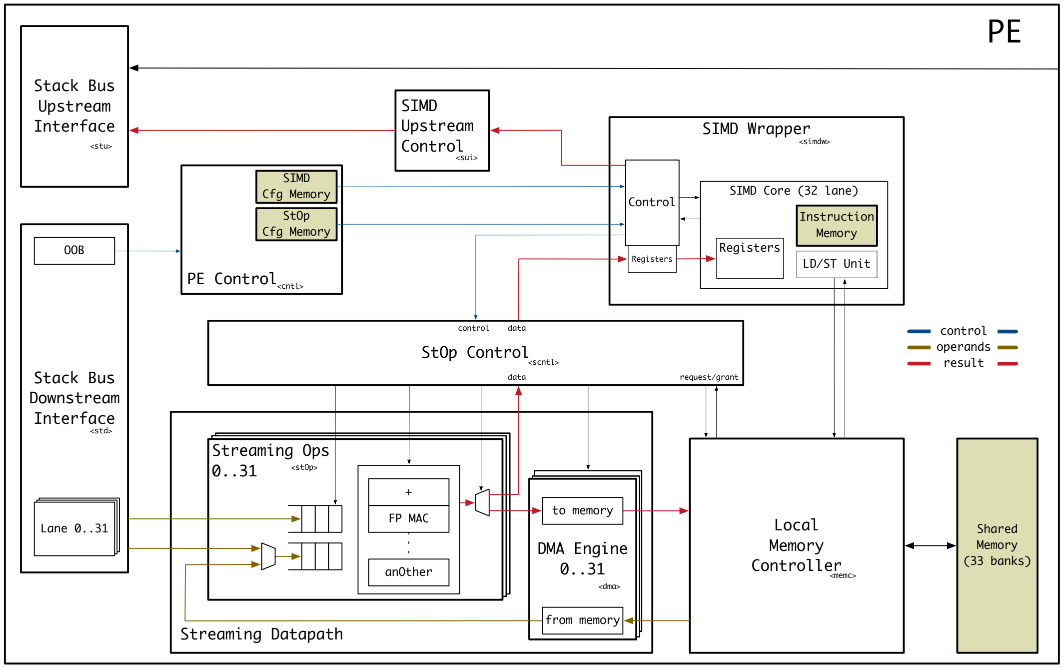
Stack Bus

- Downstrean 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



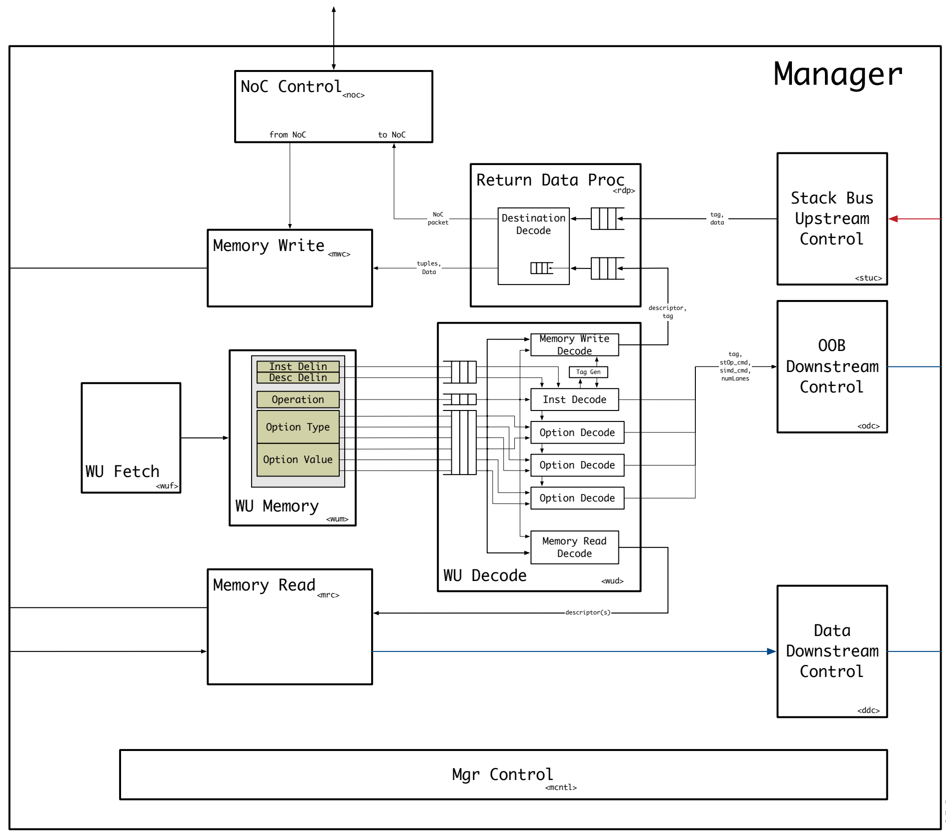
PE Status

- 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

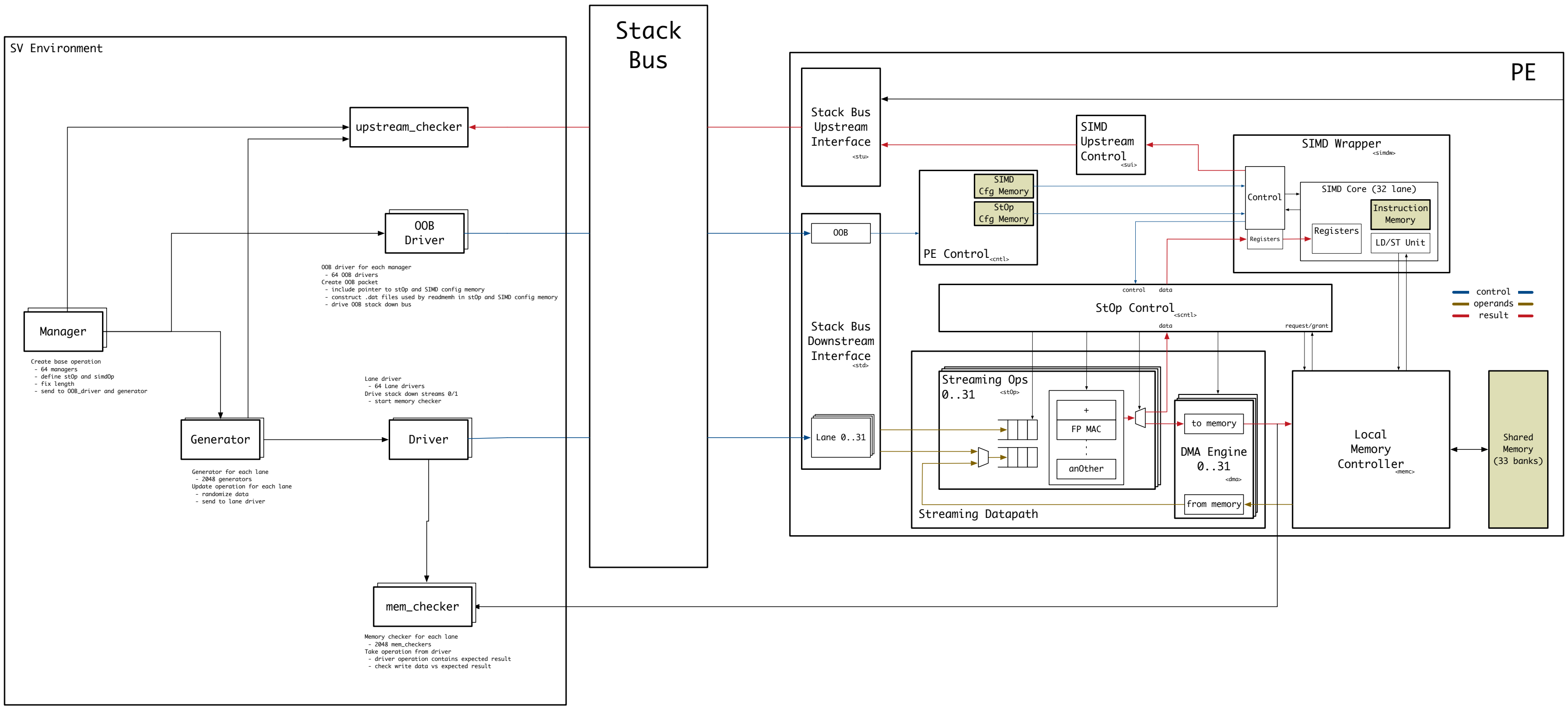
- Ensure data can be streamed efficiently from DRAM
- Operates on a custom instruction set
  - instructions describe all that is necessary to process a group of neurons
  - avoid need for large local SRAM in PE
- System requires data duplication
  - NoC designed to multicast neuron activations to other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



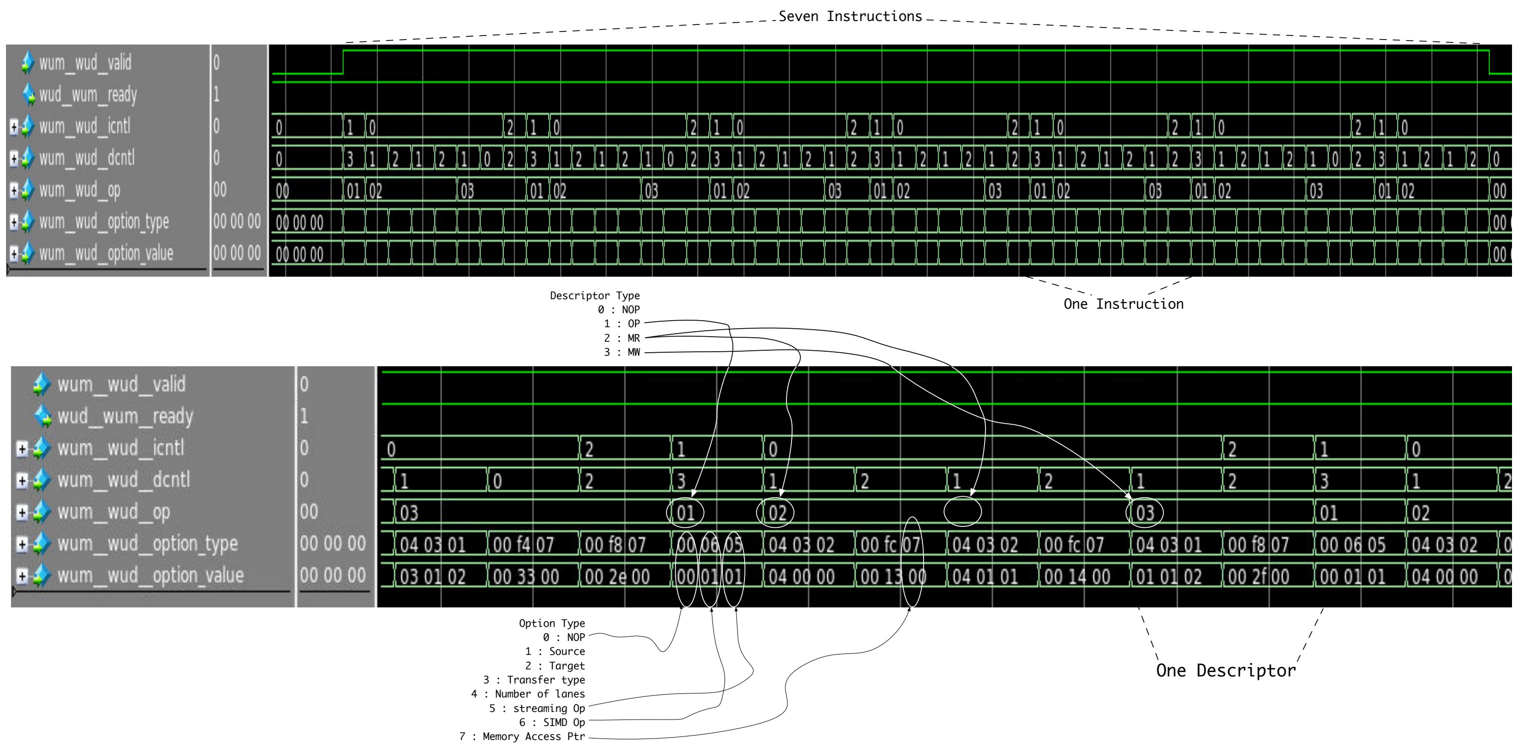
Manager Status

- 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



Instruction Communication



NoC Signaling