FPGA Neural Networks

Hardware Neural Net Entertainment System (HNES)

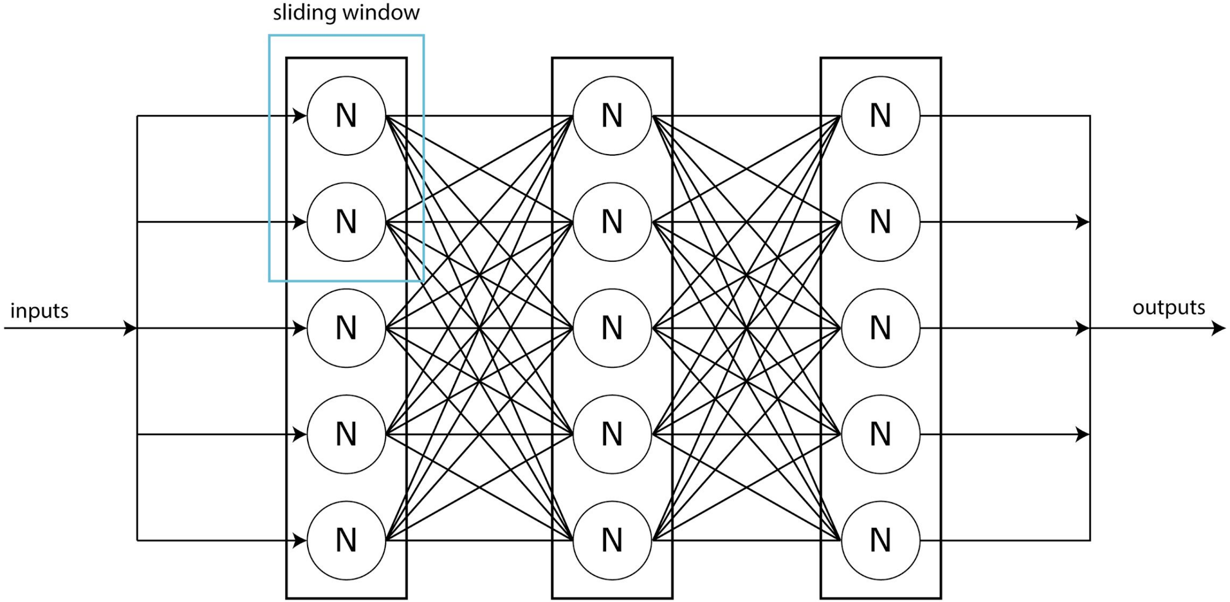
Sebastian Bartlett, Josh Noel

Project Overview

Motivation: Applications of ML on FPGA is an active area of research

Inference on Feed-Forward network topology

Interface to PC for network inputs & outputs



Network Architecture

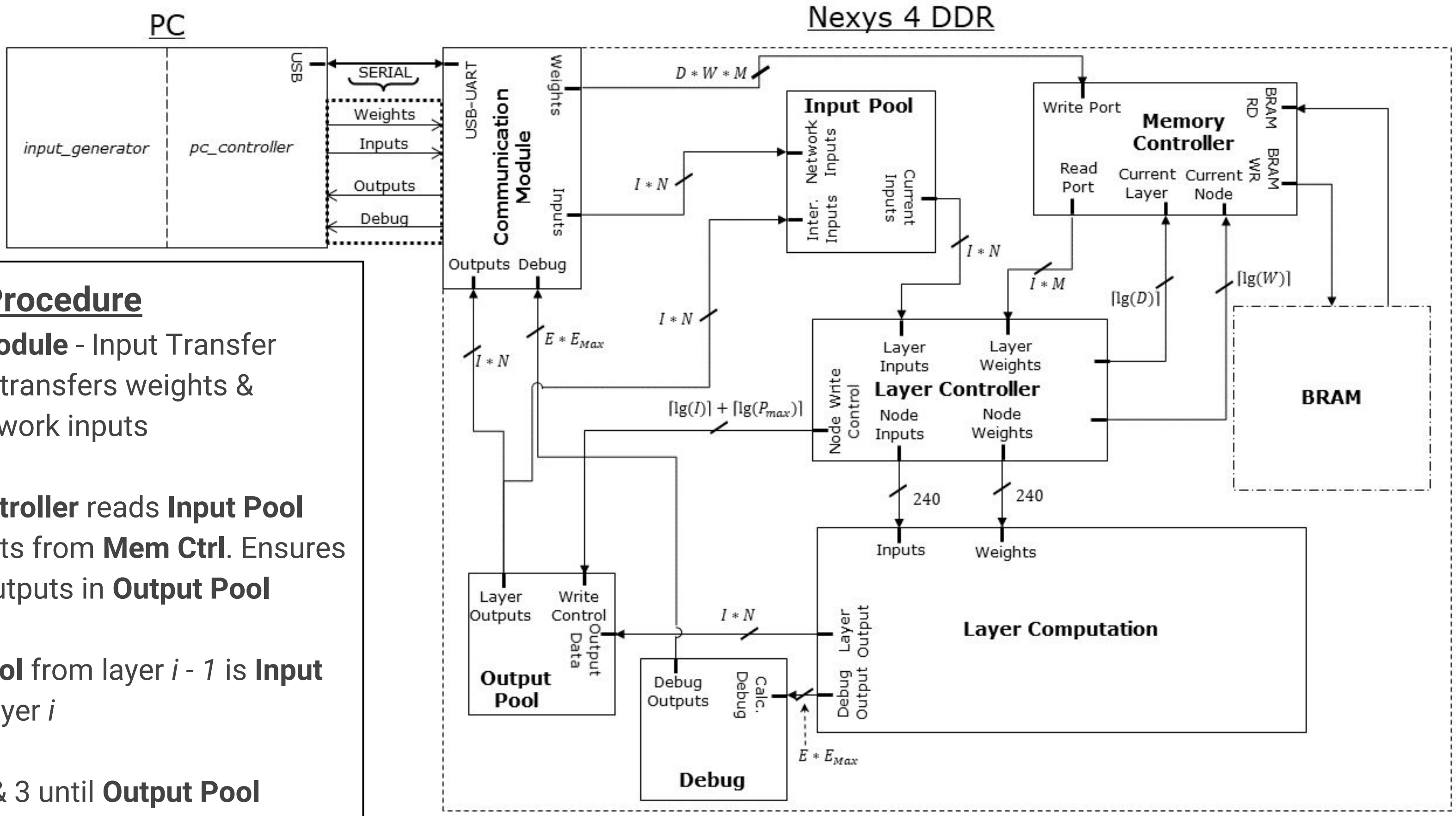
(conceptual, not module)

Parallelism => P : Length (Sliding window)

Network Depth => D : Length (Layers)

Network Width => W : Length (Nodes Per Layer)

Implementation Overview



Procedure

1. **Comm. Module** - Input Transfer
    a. PC transfers weights & network inputs
2. **Layer Controller** reads **Input Pool** and weights from **Mem Ctrl**. Ensures all layer outputs in **Output Pool**
3. **Output Pool** from layer *i - 1* is **Input Pool** for layer *i*
4. Repeat 2 & 3 until **Output Pool** contains network outputs
5. **Comm. Module -** Output Transfer

Communication Module

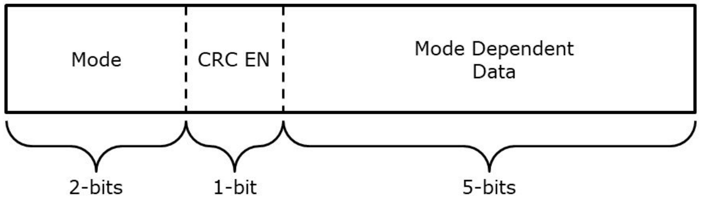
Communication over USB-UART - 10 bit data packets

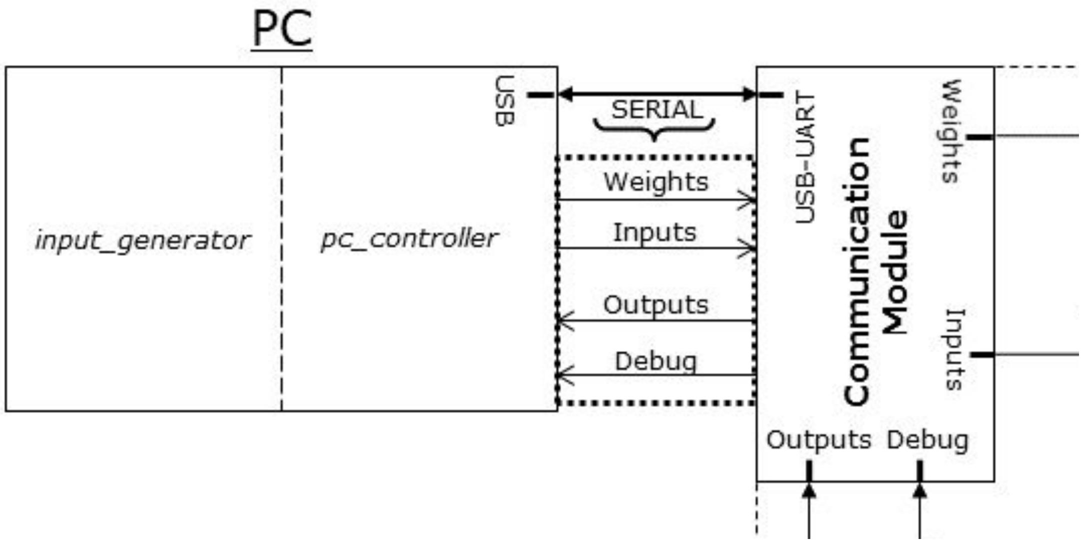
Packet Stream: [Header, 9 data packets]

Header defines data packet type

Must divide data across multiple packets

Packet Stream Header



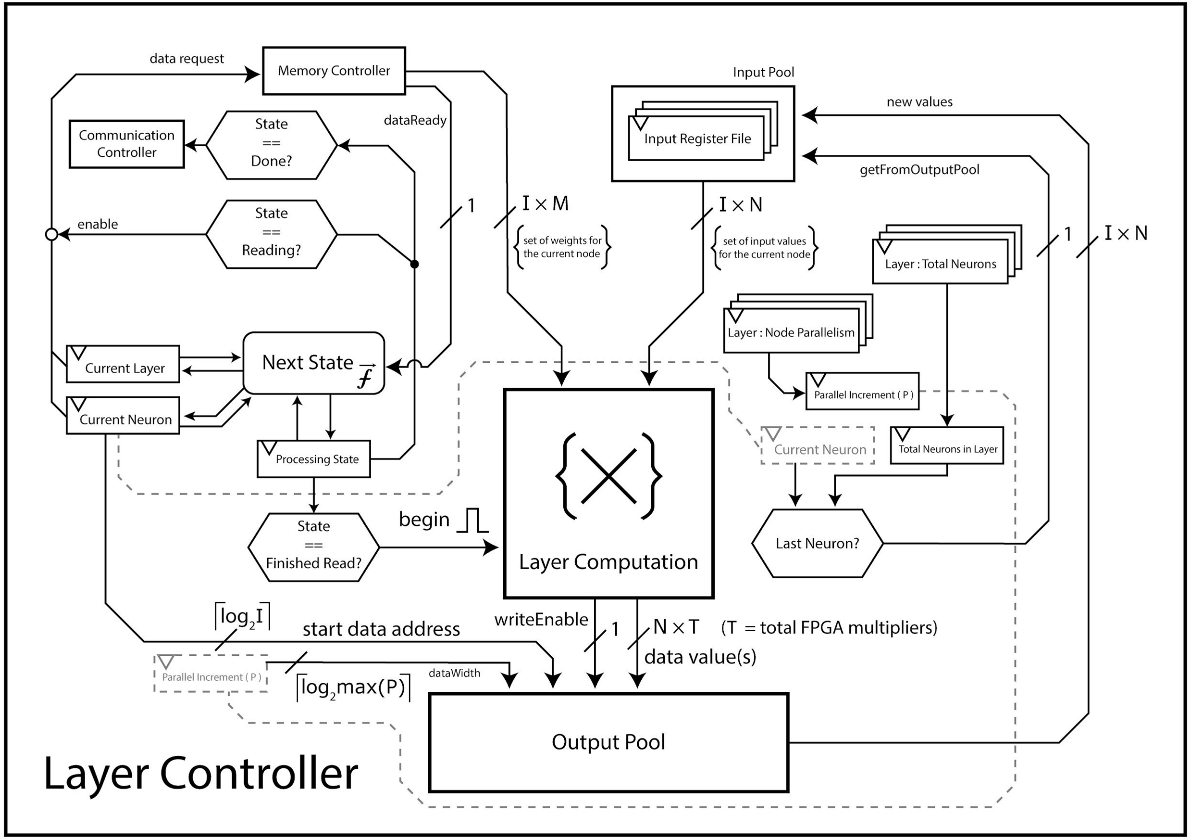


Layer Controller

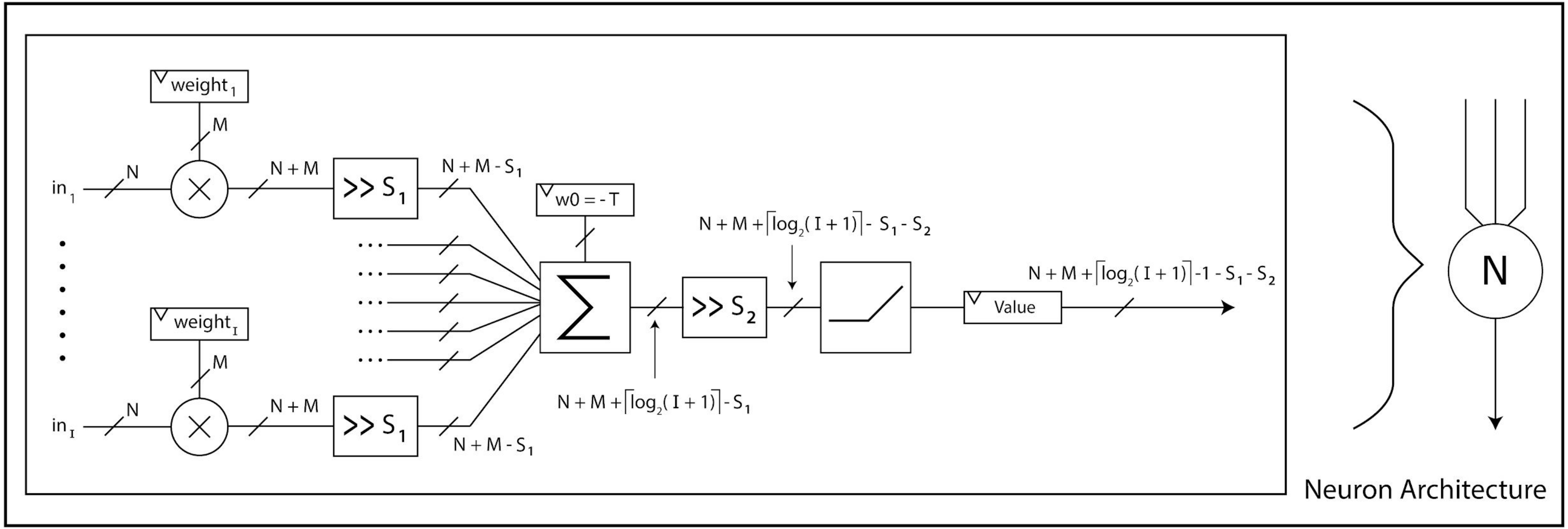
Controls all state transfers

Routes input pool & weight memory reads to **Layer Computation**

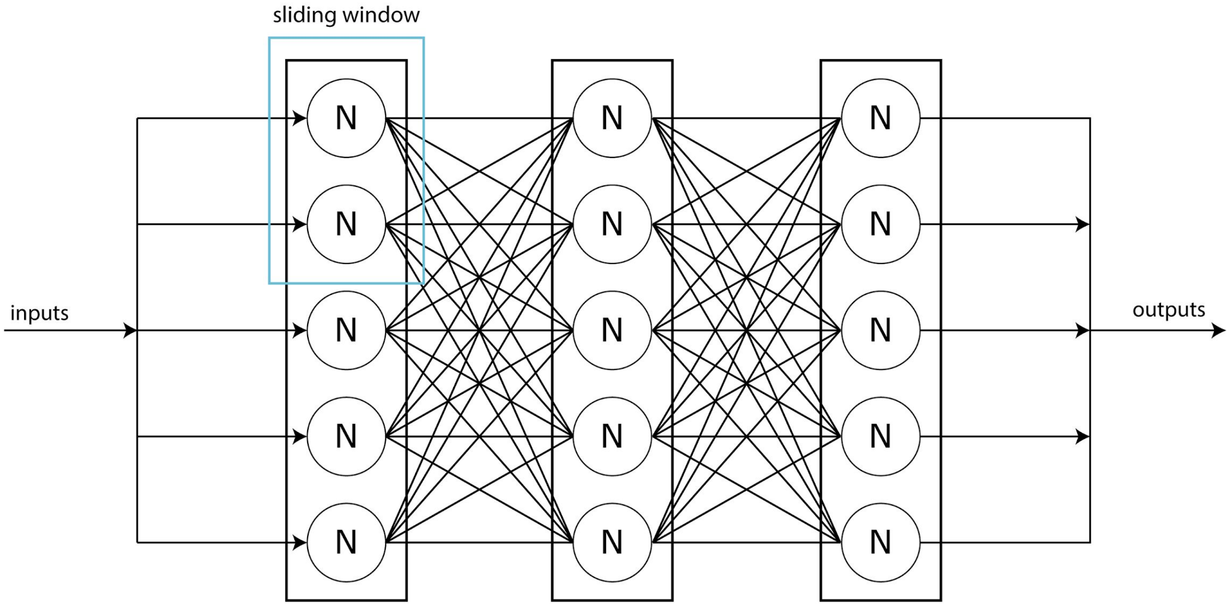
Routes **Layer Computation** output to correct addresses in **Output Pool**



Layer Computation Module



# of neurons calculated at once (parallelism) is bounded by DSP slices



Network Architecture

(conceptual, not module)

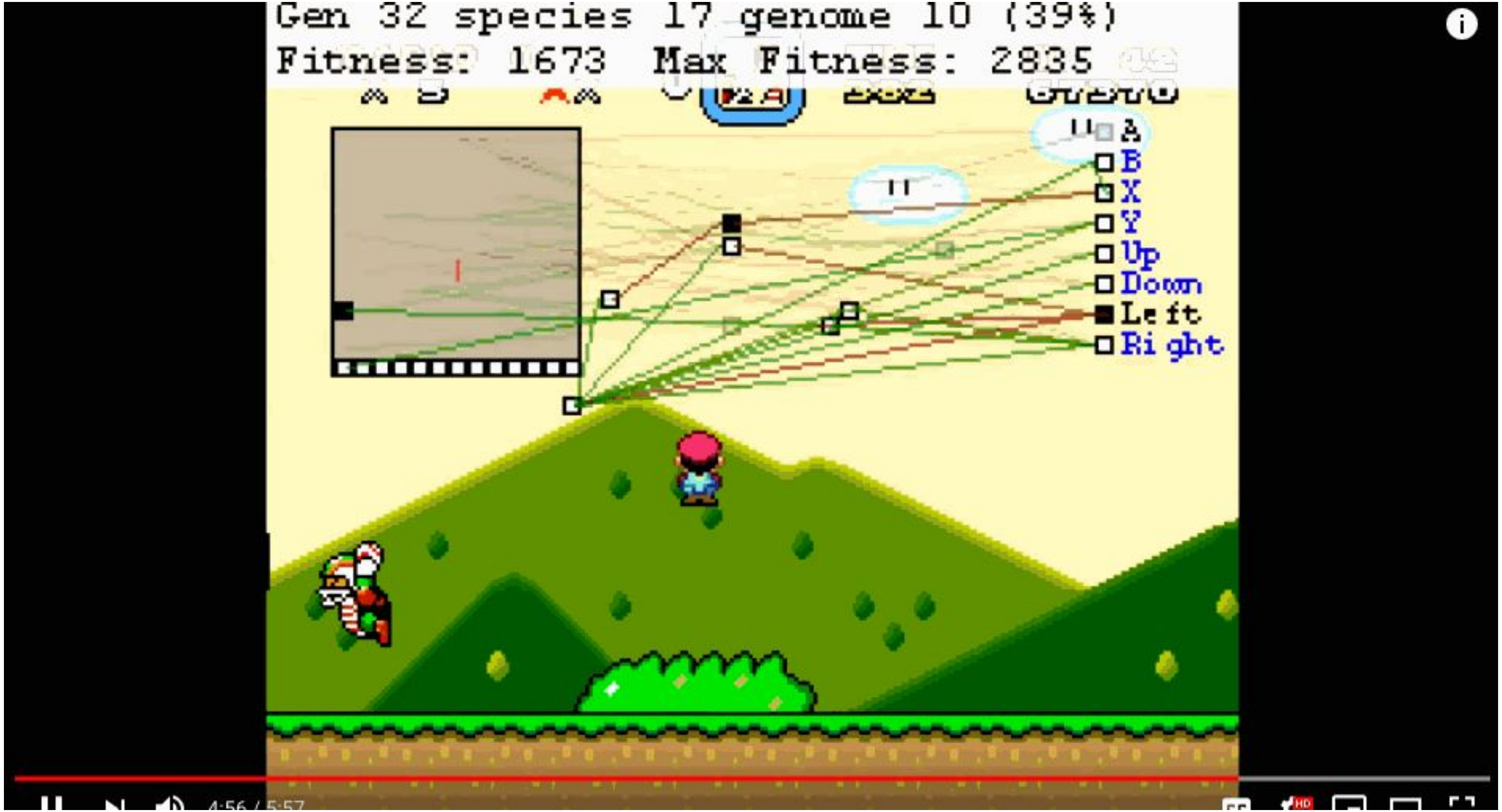
Parallelism => P : Length (Sliding window)

Network Depth => D : Length (Layers)

Network Width => W : Length (Nodes Per Layer)

Timeline

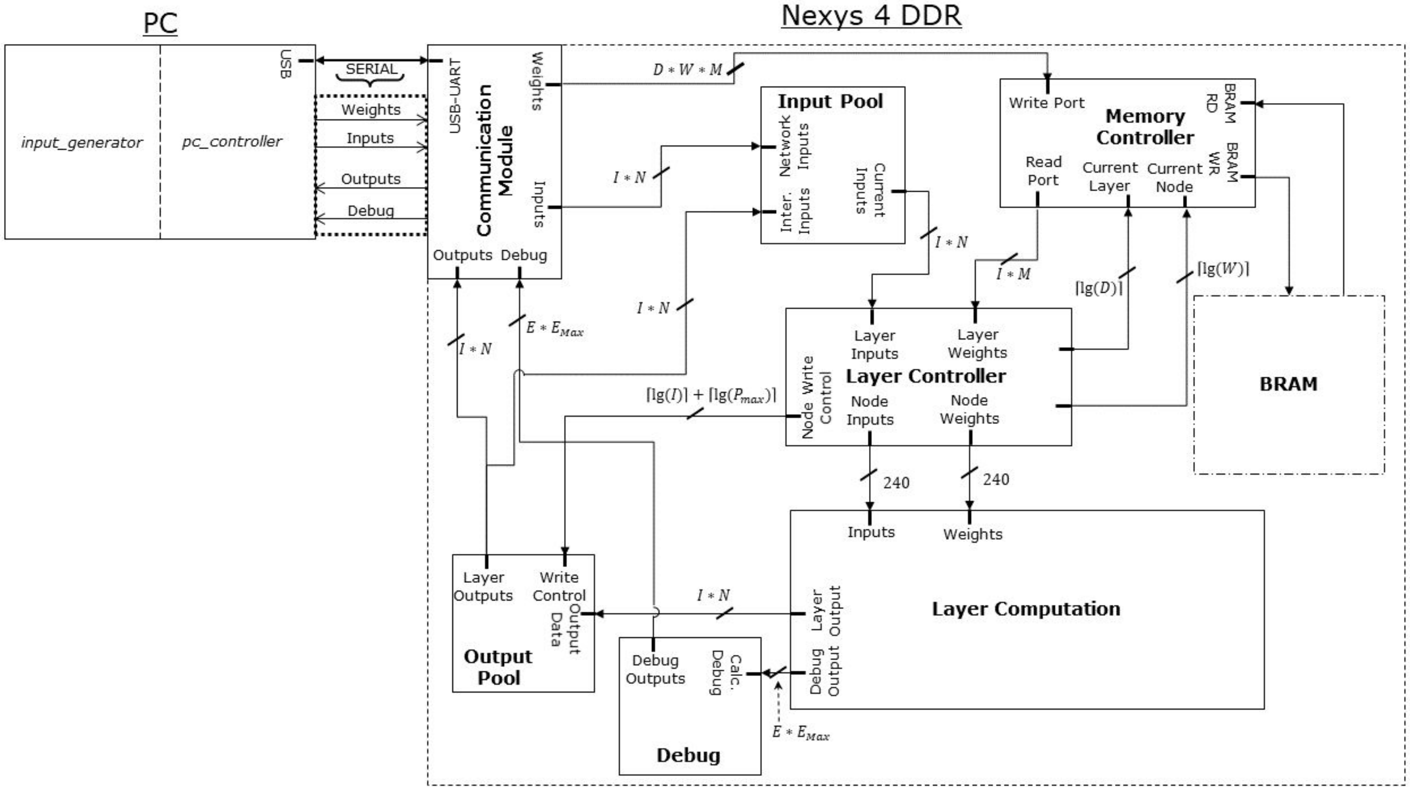
- 11/16 - Module Implementation
- 11/30 - Module Debugging, Integration Testing
- 12/10 - Neural Network Application and Implementation
    - Base Goals
        - Calculate single layer outputs based off PC inputs
            - FPGA behaves as a hardware matrix multiplier
        - Calculate output layer of multilayer feed-forward network
            - This requires calculating intermediate (hidden) layers
            - This enables inference to be performed
            - More complex topologies can be interfaced on PC-side
    - Stretch Goal
        - Interface with NES emulator to play Super Mario



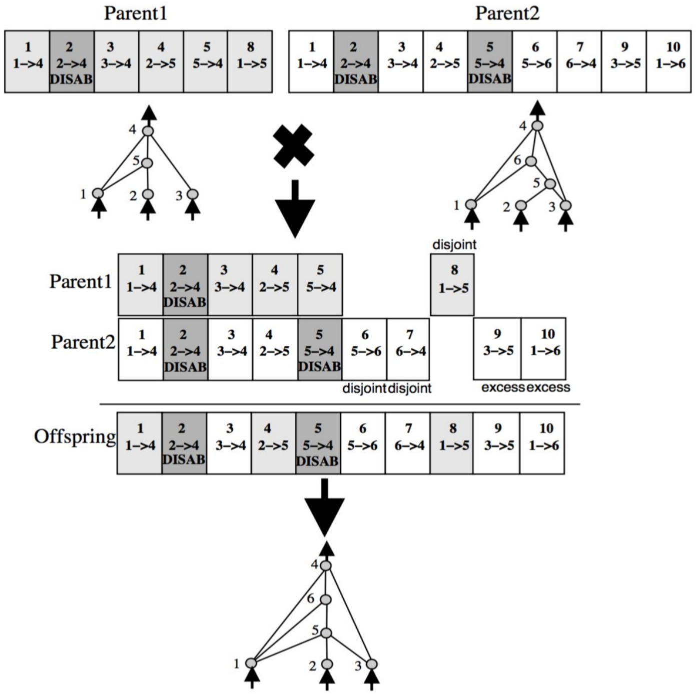
```
Gen 32 species 17 genome 10 (39%)
Fitness: 1673  Max Fitness: 2835
```

Mar I/O by “SethBling” (YouTube)

Questions?


PC
Nexys 4 DDR
input_generator
pc_controller
USB
SERIAL
Weights
Inputs
Outputs
Debug
USB-UART
Communication Module
Weights
Inputs
Outputs
Debug
$D * W * M$
Input Pool
Network Inputs
Inter. Inputs
Current Inputs
$I * N$
Write Port
Memory Controller
BRAM RD
BRAM WR
Read Port
Current Layer
Current Node
$I * N$
$I * M$
$\lceil \lg(D) \rceil$
$\lceil \lg(W) \rceil$
$I * N$
$E * E_{Max}$
$I * N$
Layer Inputs
Layer Weights
Layer Controller
Node Write Control
Node Inputs
Node Weights
$\lceil \lg(I) \rceil + \lceil \lg(P_{max}) \rceil$
BRAM
240
240
Inputs
Weights
Layer Computation
Layer Output
Debug Output
$I * N$
Layer Outputs
Write Control
Output Data
Output Pool
Debug Outputs
Calc. Debug
Debug
$E * E_{Max}$

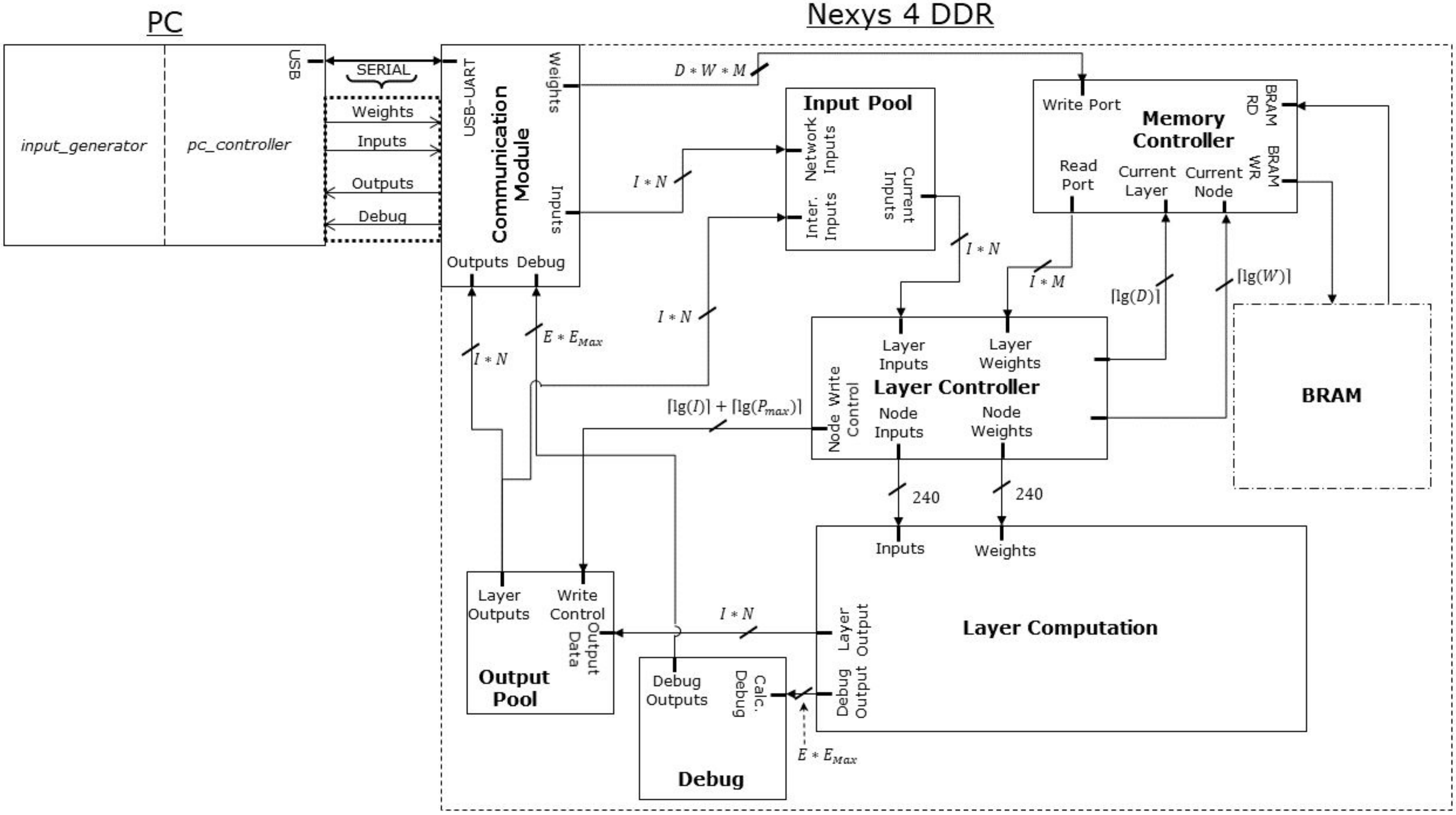
(Reference Slides)

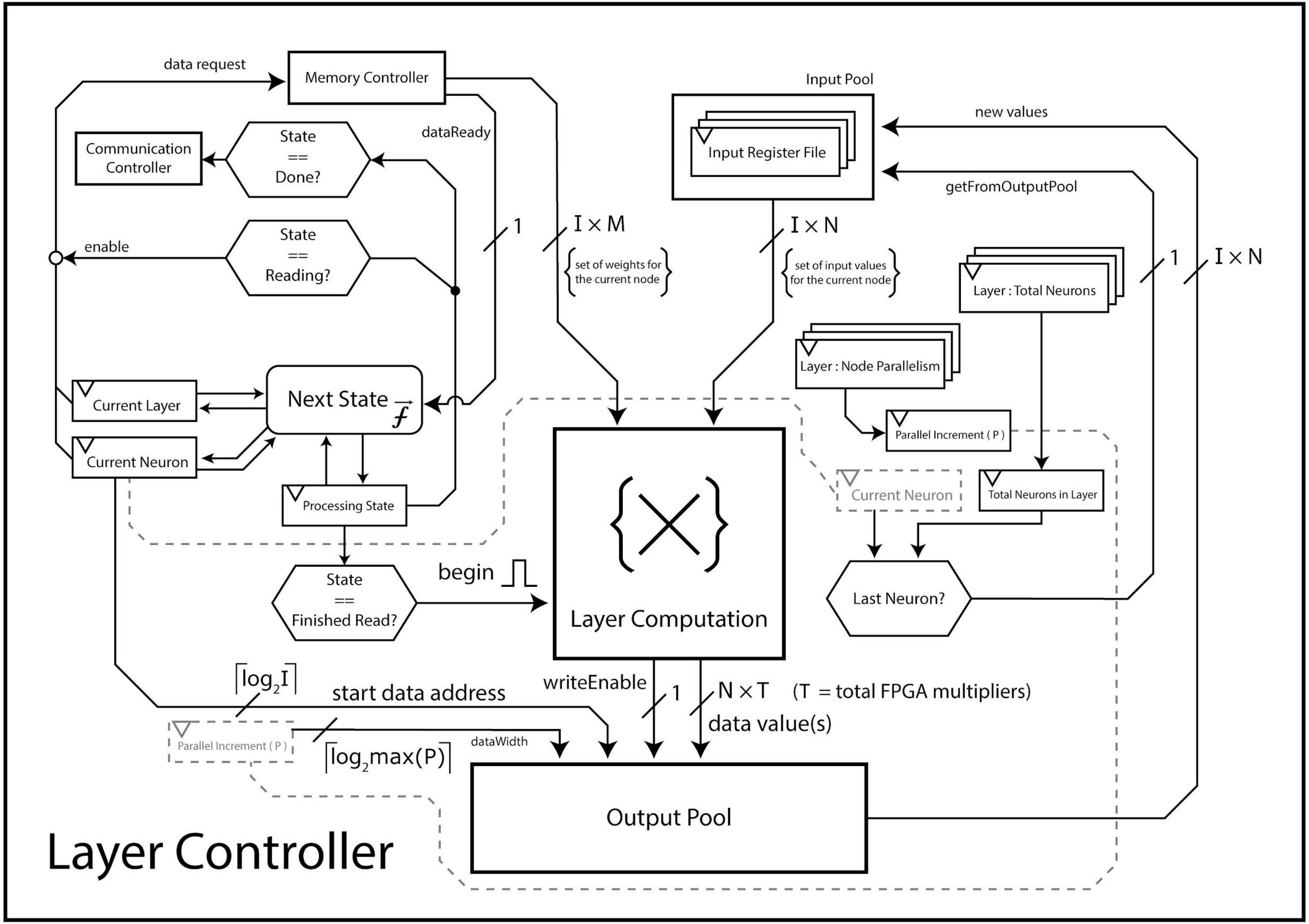


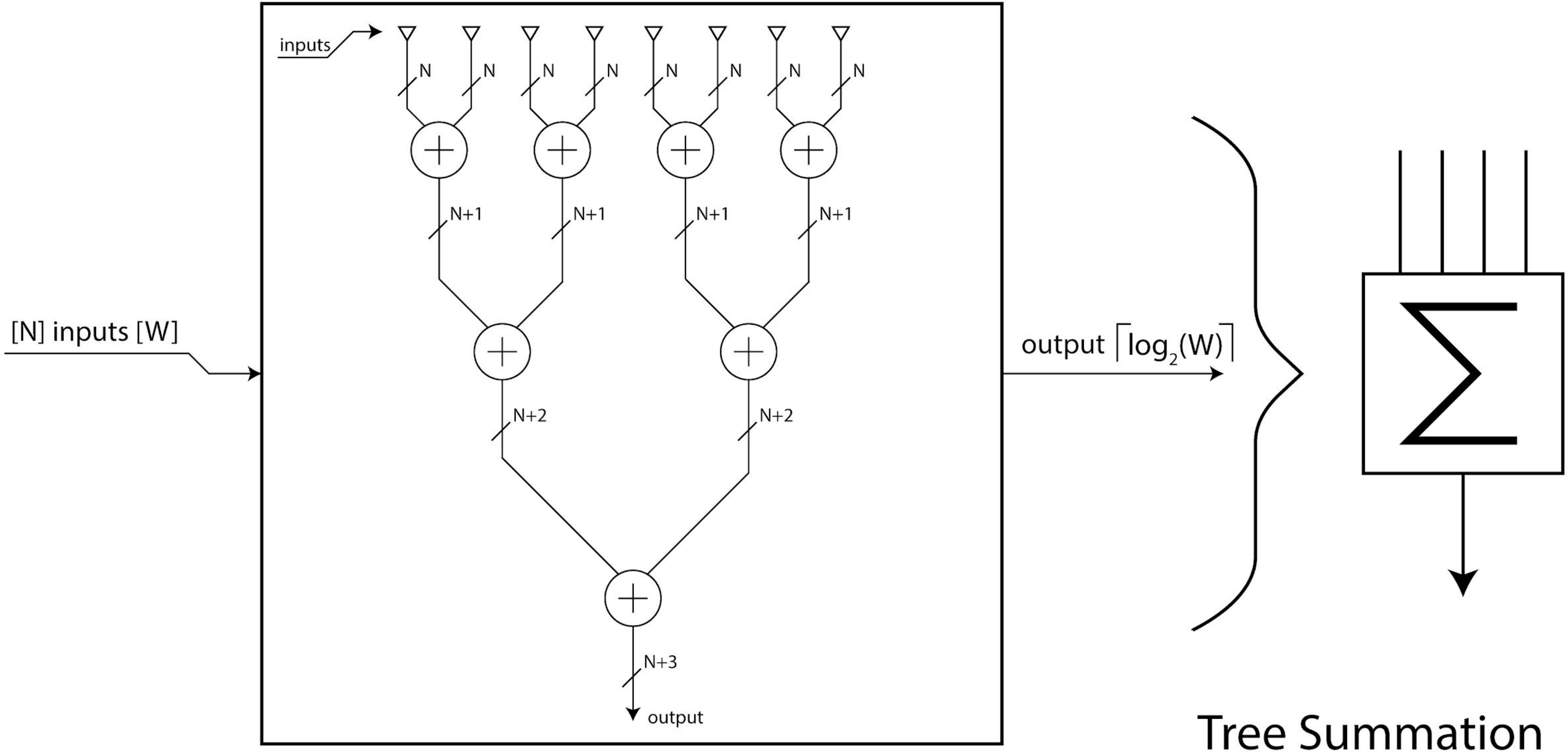
Source: Evolving Neural Networks through Augmenting Topologies

Kenneth O. Stanley kstanley@cs.utexas.edu Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712, USA

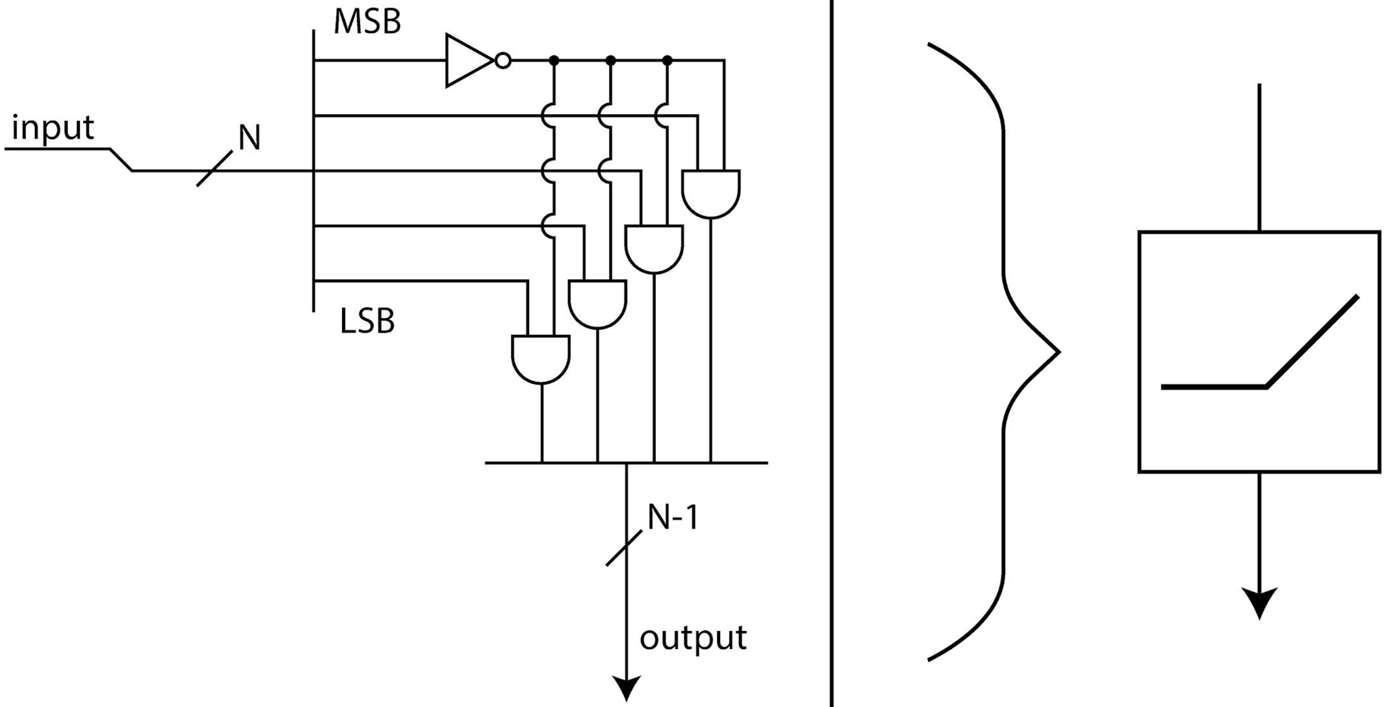
Risto Miikkulainen risto@cs.utexas.edu Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712, USA


PC
Nexys 4 DDR
input_generator
pc_controller
USB
SERIAL
Weights
Inputs
Outputs
Debug
USB-UART
Communication Module
Weights
Inputs
Outputs
Debug
$D * W * M$
$I * N$
$I * N$
$I * N$
$I * N$
$E * E_{Max}$
Input Pool
Network Inputs
Inter. Inputs
Current Inputs
$I * N$
Memory Controller
Write Port
Read Port
Current Layer
Current Node
BRAM RD
BRAM WR
$I * M$
$\lceil \lg(D) \rceil$
$\lceil \lg(W) \rceil$
BRAM
Layer Controller
Layer Inputs
Layer Weights
Node Write Control
Node Inputs
Node Weights
$\lceil \lg(I) \rceil + \lceil \lg(P_{max}) \rceil$
240
240
Layer Computation
Inputs
Weights
Layer Output
Debug Output
Output Pool
Layer Outputs
Write Control
Output Data
Debug
Debug Outputs
Calc. Debug
$E * E_{Max}$

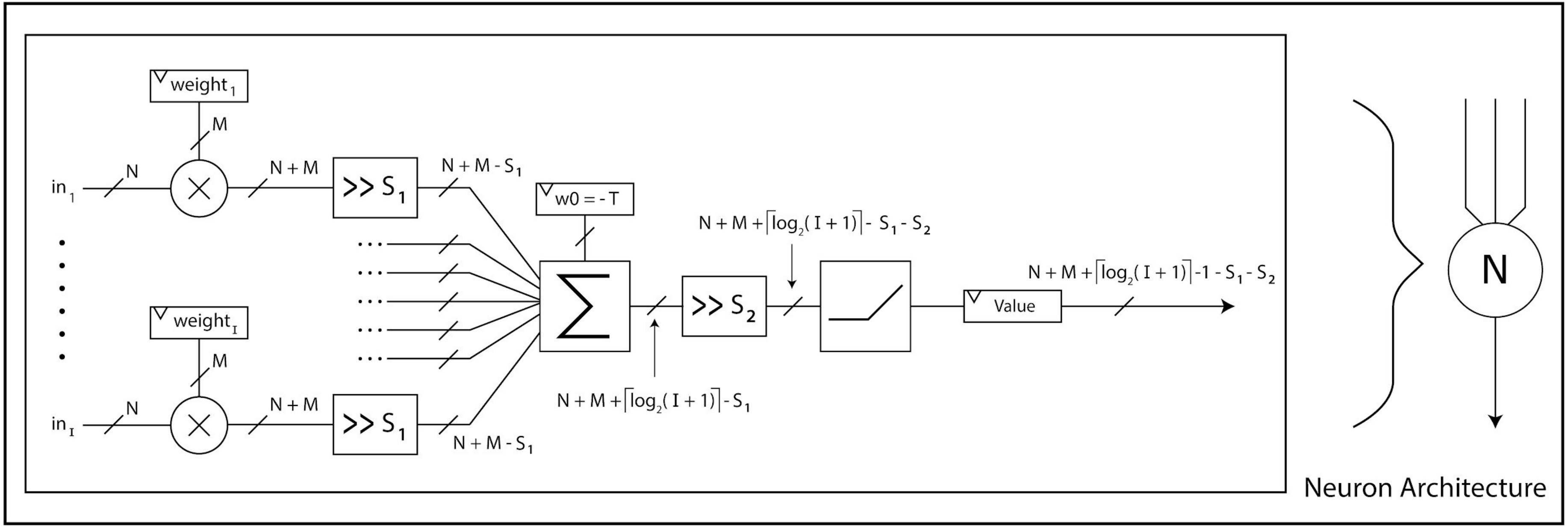

data request
Memory Controller
dataReady
Communication Controller
State == Done?
State == Reading?
enable
Current Layer
Current Neuron
Next State $\vec{f}$
Processing State
State == Finished Read?
begin
1
I × M
set of weights for the current node
Input Pool
Input Register File
I × N
set of input values for the current node
new values
getFromOutputPool
1
I × N
Layer : Total Neurons
Layer : Node Parallelism
Parallel Increment (P)
Current Neuron
Total Neurons in Layer
Last Neuron?
Layer Computation
writeEnable
1
N × T
(T = total FPGA multipliers)
data value(s)
$\lceil \log_2 I \rceil$
start data address
Parallel Increment (P)
$\lceil \log_2 \max(P) \rceil$
dataWidth
Output Pool
Layer Controller



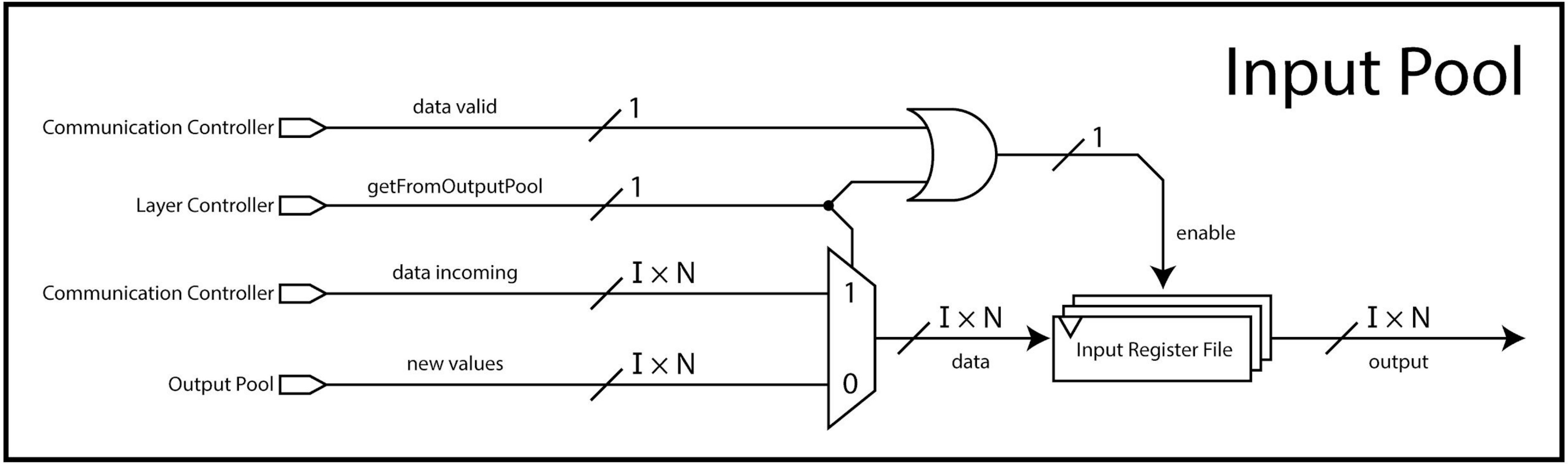
Tree Summation

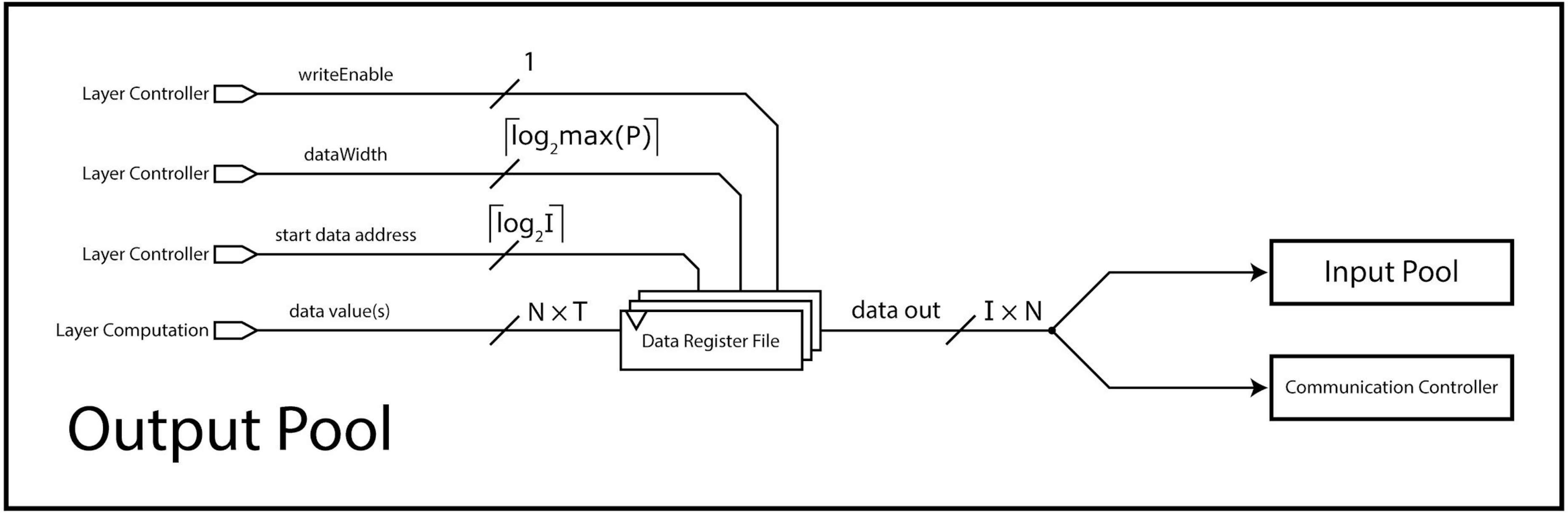


Rectified Linear Unit


weight_1
M
N
in_1
N + M
$>> S_1$
$N + M - S_1$
w0 = - T
$N + M + \lceil \log_2(I+1) \rceil - S_1 - S_2$
$N + M + \lceil \log_2(I+1) \rceil - 1 - S_1 - S_2$
$\sum$
$>> S_2$
Value
weight_I
M
N
in_I
N + M
$>> S_1$
$N + M - S_1$
$N + M + \lceil \log_2(I+1) \rceil - S_1$
N
Neuron Architecture

Input Pool


Communication Controller
data valid
1
Layer Controller
getFromOutputPool
1
Communication Controller
data incoming
I × N
Output Pool
new values
I × N
1
0
1
enable
I × N
data
Input Register File
I × N
output


Layer Controller
writeEnable
1
Layer Controller
dataWidth
$\lceil \log_2 \max(P) \rceil$
Layer Controller
start data address
$\lceil \log_2 I \rceil$
Layer Computation
data value(s)
N × T
Data Register File
data out
I × N
Input Pool
Communication Controller
Output Pool