A VLSI Design for Neuromorphic Computing

Mark Dean and Christopher Daffron
University of Tennessee College of Engineering
Electrical Engineering and Computer Science Dept.
Knoxville, TN
Research website: http://neuromorphic.eecs.utk.edu/

ISVLSI 2016 Workshop
July 11, 2016
Outline

• Why Neuromorphic Computing
• Related and Prior Work
• VLSI Design and Implementation
  – Approach
  – Full element layout
  – Array layout
  – Supporting component layout
  – Chip floorplan
• Analysis
• Future Work
Challenges with the Existing Computing Paradigm

• von Neumann / Turing computers “data bottlenecks”.
  – The heart of the problem: the more data there are, the harder it is to access it efficiently. Computation and storage are distinct, distant and mismatched.

• Deterioration of Moore’s Law.
  – Single core performance improvements ended in 2004
  – Power density and device feature size limits (manufacturing and atomic-scale/noise) have significantly slowed system scaling.
  – Industry relies on increasing parallelism to meet current computational demands, but success is limited.

• Programming High-Performance Computers is HARD.
  – Some problems don’t parallelize; and for those that do, parallelization is difficult.
  – Software complexity is a major constraint - due to development costs & lack of tools and skills.

• Data and problem sizes continue to grow.
  – Growth in data volume is increasing exponentially.
In 2015, Global Data Volume

By 2015 80% of all data will be uncertain.

By 2015 the number of networked devices will be double the entire global population. All sensor data has uncertainty.

The total number of social media accounts exceeds the entire global population. This data is highly uncertain in both its expression and content.

Data quality solutions exist for enterprise data like customer, product, and address data, but this is only a fraction of the total enterprise data.

Multiple sources: IDC, Cisco

Global Data Volume in Exabytes

Aggregate Uncertainty %

2005 2010 2015

Global Data Volume — Enterprise Data, Social Media, Sensors (Internet of Things), VoIP

The University of Tennessee at Knoxville
Growth of Global Data

Big data growth

Big data market is estimated to grow 45% annually to reach $25 billion by 2015

*greater than
Sources: Nasscom -Crisil G&A analysis

Mobile Traffic


The Promise of Neuromorphic Computing

Leveraging Neuroscience for Computing Architecture

• Potentially orders of magnitude improvement in **performance/watt**.
  – Use of: low-voltage analog/digital circuits, low-power non-volatile devices (e.g. memristors) and “spike” based computing.
• Significant increase in processing and storage capacities.
  – Changes in **information structure** and processing accuracy.
• Significant reduction in physical system size for target applications.
• Adaptable and Dynamic: Can adjust to changes in: environments, inputs, thresholds, critical conditions, ... - (**dynamic learning**)
• “training” and/or **“learning” paradigm** vs programming paradigm
• Primary target – “**complex dynamic real-time**” applications
  – Classification - Anomaly Detection
  – Controls - Analytics
Related Work

HBP Neuromorphic Many-Core System²
(SpiNNaker)

HBP Neuromorphic Physical Model¹
(BrainScaleS / FACETS)

Memristors⁶
(Picture from UCSB ECE)

FPGAs⁵
(RT-Spike, Associative Memory
Neural Networks, Hopfield Classifier)

IBM TrueNorth³

Stanford Neurogrid⁴

HBP: Human Brain Project
Architectures Inspired by Neuroscience

• There have been, and are, big efforts to develop computational methods based upon neuroscience.
  – **USA**: SyNAPSE (DARPA) - IBM True North
  – **EU**: Human Brain Project - SpiNNaker

• These efforts utilize artificial neural hardware and massive interconnections. The Challenge:
  – **Highly complex neuron models** that attempt to mimic biological behaviors make general purpose design methods hard.
  – Current efforts usually use **conventional computing structures** – increasing size, power requirements, cost.
  – The massive complexity of **the interconnections nominates** the system and could limit scalability.
  – **Application Development Tools** still need to be developed.
Our Approach – NIDA & DANNA

• **NIDA = Neuroscience-Inspired Dynamic Architecture**
  – Two simple elements: neurons & synapses
  – Both element types incorporate memory & dynamics.
    • Neurons hold “charge”.
    • Synapses contain short histories of signals in transit
  – Spike / event-based computing; distributed memory & dynamics
  – Placement, number, and connections among neurons are not constrained: network topology and function determined by evolutionary optimization & real-time adaptation.

• **DANNA = Dynamic Adaptive Neural Network Array**
  – 2D adaptation of NIDA
  – Off-the-shelf hardware (to date) minimizes cost/risk: using FPGAs.
  – Programmable array of elements (neurons, synapses, pass-thru)
  – Real-time execution (currently MHz clock rates).
  – Nearest neighbor connectivity with configurable long-range capability.
  – Mixed-signal implementations (lower power & higher element density).

• **Design tools based upon genetic algorithms support “learning” paradigm**
  – Evolutionary optimization and hardware matched simulator
Neuroscience-Inspired Dynamic Architecture (NIDA)
Dynamic Adaptive Neural Network Array (DANNA)

• Generic neuromorphic elements
  – Neuron, synapse, or pass-through
  – Connects to nearest neighbors (16)

• DANNA Element Functions
  – Input Sampling
  – Accumulate and Fire
  – Long Term Potentiation / Long Term Depression
  – Synapse Delay

• DANNA Chip Functions
  – Configuration/Command Interface
  – Input/Output FIFOs
  – Array Monitoring
  – Clocking
Present DANNA Implementation

- DANNA arrays up to 75x75
- Implemented on FPGAs
  - Xilinx 690T & 2000T
- External Interface using PCIe or USB
- Clock speeds
  - 0.5 MHz Event Clock
  - 8 MHz Element Clock
  - 16 MHz Accumulator Clock
- JTAG-like monitoring functionality
DANNA FPGA Constraints/Challenges

- Placement affecting functionality
  - Fixed by disabling some optimizations in Xilinx Vivado
- Clocking delays / load / skew
  - Fixed by manually inserting clock buffers
- Optimized out components
  - Fixed by restricting modification of certain components
- Routing congestion
  - High utilization causes routing difficulties
- Maximum clock freq.
  - But still faster than other neural network hardware implementations
VLSI Design - Approach

• Full-custom design of a single element
• IBM-8RF 130 nm process, 390 mm² chip area (max.)
• Utilize ASIC design flow
  – Create the array and route signals between elements
  – Programming Interface
• Utilizes flops for memory due to unavailability of SRAM cells
• Element was divided into seven parts (to reduce complex & simplify layout)
  – Input Sampling
  – Register File
  – LTP/LTD
  – Output Control
  – Accumulator
  – Synapse Distance
  – Monitoring
• Target 10x clocking freq. of FPGA design (e.g. 5MHz Event Clock, 160MHz Accum. Clock)
• Inverters and Buffers block much more space than they actually occupy
• Due to constant track height
• Could be solved by having multiple track heights or defining elements as black boxes independent of tracks
Due to all rows having CTS run separately, a custom clock tree needed to be created to tie them together.

A script needed to be created to tie all rows together.
VLSI Implementation – Array Layout

- 75x75 was able to be implemented
- Single DRC violation (not impacting functionality) after routing
- Clock distribution drives unique array shape
  - Number of buffers between tree root and load is approximately equal for several tested signals.
- Array takes up vast majority of space
- All supporting components at bottom of chip occupy fraction of available space.
- Array size could be increased if non-rectangular shape was allowed
- Number of Instances: 177,183
- Density: 53.75%
Analysis/Modeling – Placement Analysis

- Nearest neighbors are placed close to each other.
- Array layout should allow for connections across edges of the array.
VLSI Design – Comparison of FPGA and VLSI densities

- Improved density achieved through elimination of unnecessary transistors, logic optimization, and hand layout optimization

<table>
<thead>
<tr>
<th>Category</th>
<th>FPGA</th>
<th>VLSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Size</td>
<td>28 nm</td>
<td>130 nm</td>
</tr>
<tr>
<td>Die Size</td>
<td>600 mm²</td>
<td>390 mm²</td>
</tr>
<tr>
<td>Element Count</td>
<td>4,900</td>
<td>5,625</td>
</tr>
</tbody>
</table>

- Keeping element count constant, the VLSI design would occupy $\frac{1}{33}$rd of the FPGA die size in the same process.
- With FPGA die size/process, VLSI design could have 186,543 elements
- With 14nm process / FPGA die size, VLSI design could have 746,173 elements
With improved density and smaller processes, very large arrays can be implemented in the same die size.

<table>
<thead>
<tr>
<th>Density</th>
<th>Process</th>
<th>Elements</th>
<th>Array Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>53.75%</td>
<td>130 nm</td>
<td>5,625</td>
<td>75×75</td>
</tr>
<tr>
<td>90%</td>
<td>130 nm</td>
<td>9,419</td>
<td>97×97</td>
</tr>
<tr>
<td>90%</td>
<td>65 nm</td>
<td>37,674</td>
<td>194×194</td>
</tr>
<tr>
<td>90%</td>
<td>28 nm</td>
<td>203,028</td>
<td>450×450</td>
</tr>
<tr>
<td>90%</td>
<td>14 nm</td>
<td>812,114</td>
<td>901×901</td>
</tr>
<tr>
<td>90%</td>
<td>12 nm</td>
<td>1,156,656</td>
<td>1075×1075</td>
</tr>
</tbody>
</table>
The logic is designed to interface with the following external chips:

- Silicon Labs Si5334K-B04797-GM
  - Provides all needed clocks
- TI SN74V3670-7PEU
  - 8Kx36 FIFO Chip
Future Work

• Perfect Verilog model, possibly create a Verilog-A model
• Simulate only the clock trees (individually)
  – Create SKILL script to extract only clock tree layout
• Maximize the array size
  – Somewhere between 75 and 80 with current method
  – Approaching 95 with multi-height tracks and increased density
• Additional element optimization
• Conversion to smaller process sizes
• Larger array sizes with full-custom layout
• Analog circuits / memristors
Thank you!

Questions?
Moore’s Law Ended in 2004

Historical growth in single-processor performance and a forecast of processor performance to 2020, based on the ITRS roadmap.


VLSI Implementation – Full Element

• All individual pieces combined to create the full element.
• Emphasis placed on the layout being compact and square, keeping signal paths as short as possible.