Exploring the Design Space of Specialized Multicore Neural Processors

Tarek M. Taha, Raqibul Hasan, Chris Yakopcic, and Mark R. McLean

Abstract—This study examines the design of several novel specialized multicore neural processors. Systems based on SRAM cores and memristor devices were examined. Detailed circuit simulations were used to ensure that the systems could be compared accurately. Two types of memristor cores were examined: digital and analog cores. Novel circuits were designed for both of these memristor systems. Additionally full system evaluation of multicore processors based on these cores and specialized routing circuits were developed. Our results show that the memristor systems yield the highest throughput and lowest power.

We compared these specialized systems to more traditional HPC systems. Two commodity high performance processors were examined: a six core Intel Xeon processor, and an NVIDIA Tesla M2070 GPGPU. Care was taken to ensure the code on each platform was very efficient (multi-threaded on the Xeon processor, and a high device utilization CUDA program on the GPGPU). Our results indicate that the specialized systems can be between two to five orders more energy efficient compared to the traditional HPC systems. Additionally the specialized cores take up much less die area – allowing in some cases a reduction from 179 Xeon six-core processor chips to 1 memristor based multicore chip and a corresponding reduction in power from 17 kW down to 0.07 W.

I. INTRODUCTION

INTEREST in neuromorphic computing has been increasing significantly over the past few years. Several groups have been exploring the simulation of large scale neural networks on high performance compute clusters in order to achieve powerful data processing capabilities [1, 2, 3]. These large compute clusters consume massive amounts of electrical power, take up significant space, and are expensive. This makes these clusters unsuitable for portable applications, thus significantly limiting the applications of the large neural algorithms.

The design of small and efficient specialized neural processing chips can solve these problems. They will have a large variety of applications, including acting as accelerators for more traditional computing platforms. A recent study has shown that the class of recognition, mining, and synthesis (RMS) applications can be simulated through neural networks [4]. Intel has identified RMS applications as a key driver of future computing systems [5]. Esmaeilzadeh et al. [6, 7] show that several key application kernels (such as FFT and JPEG) can be approximated using neural networks. They make the case for specialized neural network accelerators on general purpose CPUs.

Existing work on specialized neural systems have typically focused on accurate modeling of biological neurons [8, 9], with less emphasis on power consumption and area. Recent, more efficient, specialized neural processing platforms include [8, 10, 11, 12]. The SpiNNaker project [10] developed a package with 18 ARM processor cores on a chip and 128 MB of SDRAM in a flip chip module. This system is more efficient than traditional computing platforms and is highly flexible at simulating multiple neural algorithms. However, less flexible but more efficient specialized systems could be designed. Other studies have examined specialized architectureS for convolutional neural networks [13, 14]. These systems have been implemented on FPGAs, but their power and area using custom designed chips has not been studied.

The IBM SRAM based neural core [11, 12] is an example of a more specialized and power efficient system. The core can process a limited set of neural algorithms and is designed to be built as a multicore system. Many circuit-level optimizations were utilized to reduce the active power per synapse down to 45 pJ. These include the use of high threshold devices, near threshold operations, SOI processes, low operating voltages, and asynchronous circuits. Additionally, they examined the use of low power states while the cores were inactive.

In this paper, we examined the design of novel specialized multicore neural processors, where the processing cores would be similar to the architecture in Fig. 1. We studied the impact of using two memory technologies for the synaptic array in Fig. 1: SRAM and memristors. Memristors [15, 16, 17] are a new class of devices that can offer lower energy and higher throughput per chip. Two types of memristor cores were examined: digital and analog cores. Novel circuits were designed for both of these memristor systems. Very detailed circuit simulations were used to ensure that the systems could be compared accurately. The memristor circuits were simulated using an accurate memristor SPICE model we developed recently [18].

For each form of the core in Fig. 1, the impact of scaling to a multicore chip, as shown in Fig. 2, was studied. To ensure accurate modeling of the multicore system, we
designed specialized routing circuits between the cores and accounted for their area, power, and delays. Our results show that the top two systems with highest throughput and lowest power are the analog and then digital memristor systems. The SRAM based digital system is third best. It has higher energy cost due to leakage energy – the memristors do not have this, thus making them more energy efficient.

We compared these specialized systems to more traditional HPC systems. Two commodity high performance processors were examined: a six core Intel Xeon processor, and an NVIDIA Tesla M2070 GPGPU. Care was taken to ensure the code on each platform was very efficient: multi-threaded on the Xeon processor to utilize all six cores, and a highly parallel CUDA program on the GPGPU. Our results indicate that the specialized systems can be between two to five orders more energy efficient compared to the traditional HPC systems. The energy efficiency depends on the options utilized within the core (such as SRAM or memristor memory and data bit width). The specialized cores take up much less die area — allowing, in some cases, a reduction from 179 Xeon six-core processor chips down to 1 memristor based multicore chip and a corresponding reduction in power from 17 kW down to 0.07 W.

In this study we examine the properties of several neural cores for implementing a multi-layered neural network. Each neuron in the neural network performed two types of operations: it summed its weighted inputs and evaluated a non-linear function. If the axonal inputs to a neuron are given by \( x_i \) then the corresponding neuron output was evaluated as:

\[
  v_j = \sum W_{ij} x_i \quad (1)
\]

\[
  y_j = f(v_j) \quad (2)
\]

Here, \( W \) is a weight matrix in which \( W_{ij} \) is the synaptic weight of axon \( i \) for neuron \( j \) and \( f \) is a nonlinear function (usually a sigmoid function).

The novel contributions of this work are:

1) A detailed design of two memristor neural cores: an analog and a digital core. The timing and power data for both cores are based on detailed SPICE simulations using a novel SPICE model we developed [18]. To the best of our knowledge, this is the first detailed design and SPICE simulation of a memristor core in the literature.

2) Novel circuit designs were developed for the memristor cores to reduce their overall power consumption and enable fast processing. These techniques cut down the power consumption at the expense of an increase in the area of the core.

3) The area, power, and performance of a full multicore chip based on our design were estimated through detailed modeling and simulations. Area and power of an efficient on-chip routing system was examined to obtain a more accurate evaluation of the full chip.

4) A comparison of the specialized cores against current high end processing platforms was carried out. This gives an idea of the power and area savings that can be achieved through the use of specialized cores. A comparison of this form for memristive systems has not been carried out before.

II. ARCHITECTURE OVERVIEW

Fig. 1 shows a block diagram of our proposed digital core for processing the feed forward neural network described in equation (1). Each core processes a collection of \( N \) neurons, with each neuron having up to \( M \) input axons. The input synaptic weights \( (W_{ij}) \) are stored in a memory array. These synaptic values are multiplied with the pre-synaptic input values \( (x_i) \) and are summed into an accumulator. Once the final output neural values are generated, they are sent to other cores for processing. Input and output buffers store the presynaptic inputs and post synaptic outputs respectively.

We assumed a multicore architecture as shown in Fig. 2. A collection of neural processing cores are grouped together and connected to other groups of processing cores through an on-chip routing network. A large neural network would be distributed across multiple cores, with each core processing at most \( N \) neurons from the network.

The memory array shown in Fig. 1 can be developed using several different memory technologies. The most
common memory technology integrated with processing cores today is SRAM. In SRAM technology, each memory element utilizes six transistors and stores a binary value (either 0 or 1). Resistive memory technologies, such as STT-MRAM, PCRAM, and memristors, are increasingly being studied as replacements for SRAM. This is primarily because of their higher densities and non-volatility. Of these technologies, memristors have the highest range of resistance values and have the potential for the highest memory density. The high resistance range in memristor devices allows them to model synapses in analog form. In this form, multiply and add operations can be carried out using memristor resistance values and current summation respectively, thus reducing circuit area and power consumption.

In this study we examine the design of the neural core in Fig. 1 with both SRAM and with memristor devices. Detailed analysis is used to determine the area, power, and timing of each core. Two classes of memristor based neural cores are examined: a digital case identical to the design in Fig. 1, and an analog case that was a slight variant of the design in Fig. 1. For all three cases (SRAM, memristor digital, and memristor analog), we examine system properties with 1 bit per neuron and with 4 bits per neuron. For the 1 bit per neuron systems, we assumed that the synapses are 2 bits wide. For the 4 bits per neuron systems, we assumed that the synapses are 4 bits wide.

### III. SRAM Neural Core

The SRAM based neural core studied is identical to the architecture in Fig. 1 (somewhat similar to the IBM study [11, 12]). We assumed that each core would process 256 neurons and have 1024 axons for each neuron. This would require the memory array to hold the data for 1024×256 synapses. Two specific cases of the core were examined: one with 2 bit synapses and one with 4 bit synapses. The 2 bit synapse system had a 1024×512 bit SRAM array and used 12 bit adders to accumulate weights. The 4 bit synapse system had a 1024×1024 bit SRAM array and used 4 bit multipliers and 18 bit adders.

We assumed all systems were to be developed using a 45nm process for this study and operated at a 200MHz clock frequency. The area, power, and timing of the SRAM array was calculated using the CACTI cache modeling tool [19] with the low operating power transistor option utilized. Components of a typical cache that would not be needed in the neural core were not included in the area and power calculations. These include the tag array and tag comparator power and area. The power and timing of the multipliers, adders, and all registers were calculated using SPICE. The area, power, and timing of the routing core were calculated using the Orion [20] interconnection network tool (assuming 8 bits per link). The area, power, and timing of the input and output buffers were calculated based on the buffer calculations done in Orion. The control logic within the neural core was assumed to be equivalent to the router control logic area and power (as calculated by Orion). Although this area was simply an approximation, we believe this does not affect the results of this study significantly as the SRAM array dominates the area and power of the core.

### IV. Memristor Neural Core

Two types of memristor based neural cores were examined: a digital type identical to the design in Fig. 1, and an analog type with some design optimizations compared to Fig. 1. A crossbar structure was utilized for the memristor memory circuits in this study. We developed several novel approaches for designing the memristor cores. Detailed SPICE level simulations of these circuits were utilized to examine their power and timing properties.
A. Memristor Model

To perform a device level analysis of a memristor crossbar memory system, a SPICE equivalent of the memristor model first proposed in [18] was utilized. Previous work [18] shows that this model correlates well with many published devices. This model was set to match the characterization data of one of the memristor devices published in [21] (see Fig. 3). This device was chosen because it had a large \( R_{\text{OFF}}/R_{\text{ON}} \) ratio \( \left(10^6\right) \) while still retaining a relatively low switching time (about 10 ns). It also has a large on state resistance of about 125k\( \Omega \) (determined by the 8\( \mu \)A current from a 1V read pulse).

The simulation result in Fig. 3 shows the minimum and maximum resistances of the model to be 124.95k\( \Omega \) and 125.79\times10^6\( \Omega \) respectively, which correlates very closely to the characterization [21]. These strong simulation results show that a reliable device model has been developed, and this will lead to more accurate results when simulating the crossbar tiles.

B. Memristor Crossbar Digital Core

Table 1 lists the read energy consumption of different sized memristor crossbar arrays. These power values were generated through detailed SPICE simulations taking wire resistance into account. It is seen that as the crossbar size grows, the read energy increases. This is due to extra sneak paths in a larger memory array. To constrain the read energy of a large memristor memory, we developed a tiled crossbar memory architecture as shown in Fig. 4. In this case, a single transistor is placed on each input and each output of the crossbar to allow a tile to be isolated from other tiles. Only one row of tiles is accessed at a time and this limits the leakage current within the memristor grid. The leakage current present during an operation is now limited to that of a 4\times4 crossbar. This tiling approach reduces the number of leakage paths and thus reduces the overall dynamic power consumption of the memory compared to an untiled crossbar array (see Table 1). In the digital memristor memory based neural core, the SRAM array was replaced with the tiled crossbar array seen in Fig. 4. All other components were assumed to remain the same.

C. Neural Computations in an Analog Crossbar Array

Given that memristors have a large range of resistance values \( R_{\text{OFF}}/R_{\text{ON}} \) ratio of \( 10^6 \) for the device we simulated, it is possible for a single memristor to encode a large range of synaptic weights in analog form. Thus a crossbar of memristors could potentially simulate a neural network inherently. To demonstrate this, we trained a 4\times4 crossbar array (see Fig. 5) to classify a set of 4\times4 binary patterns. Each row of a pattern was converted to an analog voltage and supplied as input to the crossbar. A detailed SPICE simulation of the crossbar that considered wire resistance showed that the circuit was able to correctly classify the linearly separable set of patterns (see Fig. 6). Each row in each of the input images is represented by a 16 value input pulse that is applied to one of the 4 inputs in the circuit in Fig. 5. The 4\times4 crossbar is inherently carrying out multiply operations by the flow of current through a memristor resistance, and doing add operations by the summing of currents in an output column. The conductivity of the each of the memristors was trained iteratively so correct classification could be performed. This classification experiment is explained in more detail in [22]. At each input and output of the crossbar, an isolator circuit consisting of two transistors and a diode is placed as shown in Fig. 5. This enables a large array of these crossbars to be read simultaneously. This is discussed further towards the end of the next section.

Fig. 5. Schematic for the crossbar designed for a 2 layer neural network simulation.

Fig. 6. Simulation results for the 2 layer memristor based linear classifier.
D. Analog Memristor Neural Core

The analog memristor core also utilized a tiled memristor crossbar array as shown in Fig. 7. A tiled design was used as this would reduce the number of sneak paths and thus the overall energy consumption. This tiled configuration was designed to allow all tiles to be read simultaneously. Thus several components in the digital circuit could be eliminated, including the memory array decoder and the multiply-add-accumulate circuits. This parallel read operation allows the large crossbar to be much faster than the digital circuit.

As with the digital case, we consider systems with 1 bit and 4 bit neural values. In the system with 4 bit values, a D-to-A converter is needed for each input axonal value. At the output of the crossbar array, the current summed from all tiled crossbar segments is converted to a 4 bit digital value using an A-to-D converter. If we assume each synapse can only hold a limited range of values, a small A-to-D circuit would be needed. In this study we optimized the design of the A-to-D and D-to-A circuits and examined their function, power, and timing using SPICE. Since it was decided that a maximum of 4 bits of precision was needed for the D-to-A and A-to-D circuits, their design was simplified. They are each based on a set of 16 transistors that switch according to the analog input voltage level.

To avoid the large energy requirement and noise associated with a high density memristor crossbar [23], the memristor crossbar used in these simulation had been partitioned into smaller 8×8 tiles. This limits the energy consumption and the unwanted current paths in the system. To ensure that unwanted current paths do not flow between crossbars, the isolator circuit in Fig. 8 is used at each input and output of the crossbars. When the write switch is enabled, current can flow in either direction. During the training phase, these transistors are used to only turn on one row of 8×8 crossbars at a time to ensure no unwanted current paths are present. Within a row of crossbars, a training operation is applied to a single row of a crossbar similarly to the digital system.

When the read mode is enabled, current can only flow in one direction since a read pulse will always be positive. The diodes stop current from flowing incorrectly to a different 8×8 crossbar, so the entire memristor grid can be read in parallel using this approach. Alternate current paths are limited to the circuit within a tile and cannot occur between tiles.

V. SIMULATION ON COMMODITY PROCESSORS

To assess the performance of the neural cores designed, we compared their performance against current commodity high performance processors. Two processor platforms were examined: an Intel Xeon X5650 processor and an NVIDIA Tesla M2070 GPGPU. The Xeon processor had six cores, ran at 2.66 GHz, and consumed 95W maximum power. The GPGPU had 448 cores, ran at 575 MHz and consumed 225W maximum power. We developed multi-layered feed-forward neural networks codes to be run on these processors with care taken to ensure the highest throughput was obtained from each processor.

On the Xeon processor, a two layer network with 1024 neurons/layer was simulated for 10 iterations. Each output neuron had 1024 synapses. This network configuration was chosen as it would fit fully within the on-chip 12MB level 3 cache and thus performance was not hampered by off-chip data access latency. This gives a more reasonable comparison with our neural cores where all synaptic data is stored on-chip as well. The neural network program was written in C and utilized the POSIX thread library to enable multi-threading. Thus all six cores on the chip were utilized. A throughput of 3.6 million neurons/s was achieved.
Although our C program did not utilize the SIMD capability of the processors, we multiplied the neurons/s throughput by four to mimic the best possible performance that could have been achieved with the Intel Xeon processor.

GPGPUs provide significantly better performances compared to traditional multicore processors for data parallel applications. Several groups have studied the simulation of neural networks on these systems [1, 2]. We examined the simulation of a 10 layer neural network with 1024 neurons/layer for 20 iterations on an NVIDIA Tesla M2070 GPGPU. The program was written in CUDA. It was parallelized to enable each thread to model one synapse, allowing a maximum of 1024×1024 threads to exist at a time. This high parallelism is needed to take advantage of the large number of cores on the GPGPU (448 cores). The program was designed to use shared memory to optimize performance. Profiling of the code using the CUDA profiler showed 85% utilization of the GPGPU cores. The throughput of the GPGPU was about 215 million neurons per second. This is a 60 times speedup over the Xeon processor. However the higher power of the GPGPU makes the GPGPU about 6 times more power efficient per neuron processed compared to the Intel Xeon processor (assuming SIMD operations on the Xeon processor).

VI. RESULTS

A. Core Properties

We developed detailed area, timing, and power analysis of the neural cores studied. Two versions of each core were studied with neuron values being either 1 bit or 4 bit. Tables 2 and 3 show the three configurations examined for the 1 bit and 4 bit cases respectively. The energy per neuron listed in the tables is based on a comprehensive analysis of each core.

Table 2. 1 bit per neuron core configurations and performance (multiplier is not needed because of 1 bit input)

<table>
<thead>
<tr>
<th>Config.</th>
<th>Synaptic memory device</th>
<th>Memory cells per synapse</th>
<th>Bits per synapse</th>
<th>Adder</th>
<th>Core area (mm²)</th>
<th>Energy per neuron (nJ)</th>
<th>Core throughput (neurons/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Memristor</td>
<td>1</td>
<td>Analog</td>
<td>Current add</td>
<td>0.037</td>
<td>0.02</td>
<td>263.9×10⁶</td>
</tr>
<tr>
<td>2</td>
<td>Memristor</td>
<td>2</td>
<td>2 bits</td>
<td>12 bit adder</td>
<td>0.098</td>
<td>0.24</td>
<td>42.1×10⁶</td>
</tr>
<tr>
<td>3</td>
<td>SRAM</td>
<td>2</td>
<td>2 bits</td>
<td>12 bit adder</td>
<td>0.288</td>
<td>0.38</td>
<td>42.1×10⁶</td>
</tr>
</tbody>
</table>

Table 3. 4 bit per neuron core configurations and performance

<table>
<thead>
<tr>
<th>Config.</th>
<th>Synaptic memory device</th>
<th>Memory cells per synapse</th>
<th>Bits per synapse</th>
<th>Multiplier</th>
<th>Adder</th>
<th>Core area (mm²)</th>
<th>Energy per neuron (nJ)</th>
<th>Core throughput (neurons/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Memristor</td>
<td>1</td>
<td>Analog</td>
<td>Memr. resist.</td>
<td>Current add</td>
<td>0.058</td>
<td>0.03</td>
<td>66.3×10⁶</td>
</tr>
<tr>
<td>5</td>
<td>Memristor</td>
<td>4</td>
<td>4 bits</td>
<td>4 bit multiplier</td>
<td>18 bit adder</td>
<td>0.179</td>
<td>0.39</td>
<td>28.6×10⁶</td>
</tr>
<tr>
<td>6</td>
<td>SRAM</td>
<td>4</td>
<td>4 bits</td>
<td>4 bit multiplier</td>
<td>18 bit adder</td>
<td>0.513</td>
<td>0.78</td>
<td>26.6×10⁶</td>
</tr>
</tbody>
</table>

Table 4. 1 bits per neuron systems. Both examples are processing 2.56 billions neurons/s.

<table>
<thead>
<tr>
<th>Configuration</th>
<th># of chips</th>
<th>Chip area (mm²)</th>
<th>% Active</th>
<th>Power (W)</th>
<th>Power eff. over Xeon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memristor Analog (config 1)</td>
<td>1</td>
<td>3.7</td>
<td>9.7%</td>
<td>0.07</td>
<td>253,489</td>
</tr>
<tr>
<td>Memristor Digital (config 2)</td>
<td>1</td>
<td>9.7</td>
<td>60.8%</td>
<td>0.62</td>
<td>27,546</td>
</tr>
<tr>
<td>SRAM (config 3)</td>
<td>1</td>
<td>35.2</td>
<td>60.8%</td>
<td>1.13</td>
<td>15,099</td>
</tr>
<tr>
<td>NVIDIA M2070</td>
<td>12</td>
<td>529.0</td>
<td>99.2%</td>
<td>2700.00</td>
<td>6</td>
</tr>
<tr>
<td>Intel Xeon X5650</td>
<td>179</td>
<td>240.0</td>
<td>99.9%</td>
<td>17005.00</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5. 4 bits per neuron systems. Both examples are processing 2.56 billions neurons/s.

<table>
<thead>
<tr>
<th>Configuration</th>
<th># of chips</th>
<th>Chip area (mm²)</th>
<th>% Active</th>
<th>Power (W)</th>
<th>Power eff. over Xeon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memristor Analog (config 4)</td>
<td>1</td>
<td>5.9</td>
<td>38.6%</td>
<td>0.07</td>
<td>234,859</td>
</tr>
<tr>
<td>Memristor Digital (config 5)</td>
<td>1</td>
<td>18.2</td>
<td>89.6%</td>
<td>0.62</td>
<td>16,968</td>
</tr>
<tr>
<td>SRAM (config 6)</td>
<td>1</td>
<td>29.1</td>
<td>89.6%</td>
<td>1.13</td>
<td>8,215</td>
</tr>
<tr>
<td>NVIDIA M2070</td>
<td>12</td>
<td>529.0</td>
<td>99.2%</td>
<td>2700.00</td>
<td>6</td>
</tr>
<tr>
<td>Intel Xeon X5650</td>
<td>179</td>
<td>240.0</td>
<td>99.9%</td>
<td>17005.00</td>
<td>1</td>
</tr>
</tbody>
</table>
They are based on both the active and leakage energy for all the components within each core and also the energy needed to transfer data into and out of the cores.

The 1 bit and 4 bit per neuron cases were designed to use 2 bit and 4 bits per synapse respectively. Thus the digital systems needed multiple memory cells to store a synaptic value. In the analog cores, it was assumed that the resistance of one memristor could be varied to accommodate these synaptic values. With 1 bit per neuron, a multiplier is not needed to calculate neural outputs. With 4 bits per neuron, multipliers are needed in the digital cores, while the analog cores use the memristor conductance to carry out multiplications in the analog domain. In all the digital cores, one synapse is read per cycle for all the neurons in parallel, thus requiring an adder to accumulate the weighted synaptic outputs over multiple cycles. In the analog cores, all the synapses of all the neurons operate in parallel, thus allowing current summations to perform addition of the synaptic outputs.

B. System Properties

This section examines the system level impact of the core architectures described in Tables 2 and 3. We calculated the performance of the multicore systems based on the cores described earlier. Each system was assumed to have a routing network to allow multiple cores to communicate with each other. We assumed that eight cores shared a routing core.

Two examples neural network processing scenarios were studied. In the first example, all systems processed a network with 25,000 neurons at a rate of 150,000 iterations per second. This lead to a throughput of 2.56 billion neurons processed per second. In the second example, we kept the neurons per second throughput the same as the first example, but processed a much larger network at a slower rate of 1500 iterations/s. In addition, we examined what the runtime would be on a cluster of current high performance compute systems. The systems examined include the NVIDIA Tesla M2070 GPGPU and the Intel Xeon X5650 six-core processor. To ensure a fair comparison the GPGPU and the Xeon processors also evaluated 1024 synapses per neuron.

As shown under example 1 in Table 4, the specialized neural systems consumed much less power and area compared to the current commercial systems. About 12 NVIDIA GPGPUs or 179 Xeon processors were needed to process the network, leading to a power consumption of 2.7kW and 17kW respectively. All the specialized neural systems had a power consumption of less than 2W and a chip area of less than 36mm². The digital memristor system was 2 times more power efficient and 28% the area of the SRAM based system. The analog memristor system was 16 times more power efficient and 11% the area of the SRAM based system.

Each specialized neural core processed only 256 specific neurons. Thus each core was active only 100,000 times a second in this example, and the network was processed at the same rate. This leads to the cores being inactive for part of the time if they can process their 256 neurons at a higher rate. A downside of the specialized cores is that they cannot be multiplexed to process other neurons while they are inactive. Thus a larger network processed at a slower rate may have the same overall neurons/s throughput, but would require more specialized cores sitting idle for a longer time between successive iterations.

To test the impact of this, the second example processes a larger network at a slower rate, but it has the same overall neurons/s throughput. Since the NVIDIA GPGPUs and Xeon processors can multiplex among different neurons, the same number of GPGPUs and Xeon processors would be needed as in the first example (see example 2 in Table 4).

More specialized neural cores are needed to accommodate the larger number of neurons in example 2, leading to larger chip area and power compared to example 1. As expected, the active times of the specialized cores were very low due to the low iteration rate. With this low activity rate, it is seen that the memristor based systems are far more power efficient than the SRAM based neural cores. Since memristors can retain their data when powered off, we assumed that the memristor based cores could be powered down during their inactive periods. However the routers on these chips were assumed to be active throughout, thus leading to higher power consumption for the larger chips in example 2 compared to example 1. The SRAM neural cores had higher energy consumptions due to the leakage energy of the SRAM arrays. The SRAM cores could potentially be set to a lower power data retentive mode to increase their power efficiencies (as described in [11, 12]). Further study is needed to examine the impact of this mode on our cores.

Table 5 lists the data for the same examples as in Table 4, but for systems with 4 bits per neuron and input. In this case the chip areas and power were larger because the cores were bigger and slower than in the 1 bit case. The power efficiency trends however were the same as in example 1.

VII. CONCLUSION

This study examined the design of several high performance, low power specialized neural cores. These included both memristor and SRAM based cores. Two cases of the memristor cores were studied – digital and analog. Full system performance prediction of multicore processors with these cores and specialized on-chip routers were developed.

The results indicate that specialized multicore processors can provide significant area, power, and speed efficiencies over current commodity high performance compute platforms. These efficiencies come at the cost of generalizability – however a large body of studies has shown that neural processing can be applied a broad range of applications [4, 6, 7]. Memristor based cores provide the highest efficiencies, while in several cases the SRAM based specialized cores were very efficient as well. The SRAM based cores lost efficiency when they were idle for longer periods of time due to leakage energy loss. Specialized low power circuit techniques as utilized by IBM in their SRAM based neural core [11, 12] can drastically cut down on this power. Some of these techniques could be applied to the memristor cores and possibly the on-chip routers examined.
Future work will be to examine these low power circuit approaches and develop more accurate system level simulation tools for these neural cores. In this study we assumed a 100% switching activity factor within all the core components during neural processing. A more detailed simulation with actual pattern recognition applications would likely result in much lower switching activity factors and thus lower power.

REFERENCES


