Minitaur, an Event-Driven FPGA-Based Spiking Network Accelerator

Daniel Neil and Shih-Chii Liu, *Senior Member, IEEE*

***Abstract*— Current neural networks are accumulating accolades for their performance on a variety of real-world computational tasks including recognition, classification, regression, and prediction, yet there are few scalable architectures that have emerged to address the challenges posed by their computation. This paper introduces Minitaur, an event-driven neural network accelerator, which is designed for low power and high performance. As an field-programmable gate array-based system, it can be integrated into existing robotics or it can offload computationally expensive neural network tasks from the CPU. The version presented here implements a spiking deep network which achieves 19 million postsynaptic currents per second on 1.5 W of power and supports up to 65 K neurons per board. The system records 92% accuracy on the MNIST handwritten digit classification and 71% accuracy on the 20 newsgroups classification data set. Due to its event-driven nature, it allows for trading off between accuracy and latency.**

***Index Terms*— Deep belief networks, field programmable arrays, machine learning, neural networks, restricted Boltzmann machines, spiking neural networks**

I. Introduction

RECENT advances in neural networks and machine learning have demonstrated their marked usefulness for real-world tasks. However, neural network computation suffers because the calculations required are not ideally suited to modern computer architectures. In standard feedforward networks, the fundamental computation can be thought of as a matrix multiplication between weights of a layer and their activations; this can be done efficiently on GPUs, but matrix multiplication scales poorly (with computation requirements greater than $O(n^2)$), ultimately requiring a very large number of computations. An alternative approach would be ideal, one that is significantly lower power (for robotics and mobile computation) and can support much larger network sizes than a full matrix multiplication can allow.

Minimizing wasted computation is necessary to dramatically reduce computational load, and event-driven computation is one approach that can achieve this goal. Computational vision typically presents a static image in a completed frame, but significant speedups can be achieved if visual information is processed in a frame-free manner [1], [2], enabling extremely rapid computation. If every frame was to be processed, a significant amount of computation would be duplicated (and wasted) in computing static backgrounds or uninteresting features. This computational style works exceptionally well for neuromorphic event-driven sensors, as shown in [1] and [3]–[5]. These event-driven devices send out spikes that show events that need processing resulting in much more efficient computation.

Critically, the system performance in an event-based system is proportional to network activity and not network size. This allows for a dramatically larger parameter space to obtain more accurate results, and does so while speeding up runtime execution and requiring fewer computations (thus requiring lower power as well). Furthermore, the decreased computational load can allow the processing to be done in the real-time domain, making a platform ideal for real-world applications and platforms with real-time interaction.

Current computing architectures are not ideally suited for network architectures like neural networks. The inherent massive parallelism of neurons, in which each performs a similar computation simultaneously, implies a more parallel architecture than what CPUs currently provide. GPUs can capitalize on the parallelism of the network but they are not suited to event-driven computation. Current GPU programming paradigms use a kernel-launch approach in which a large chunk of computation is offloaded onto the GPU with a batch of data instead of continuously run. In addition, the power consumption of GPUs precludes most of the embodied robotics applications. Because of the above reasons, none of these platforms is suited for network architectures, such as spiking deep belief networks (DBNs), especially event-driven DBNs, which combine the dramatic advances achieved with DBNs [6] with the performance of event-based processing [7].

Field-programmable gate array (FPGA) architectures, however, can address these issues. They are low-cost devices, available off-the-shelf in a variety of configurations, and are reconfigurable to allow updating and the reconfiguration of the source design as necessary. They inherently support parallel processing and contain enough local memory to cache many weights present in a typical deep network. They are low power compared with CPUs and GPUs, and a successful design can be turned into a lower power, higher performance application-specified integrated circuit in the future. In addition, the design code can be shared electronically to allow researchers to collaborate and upgrade their physical hardware without additional cost or physical adjustment.

This paper introduces Minitaur, an event-driven FPGA-based spiking neural network accelerator. This accelerator is used to study the ability of an FPGA platform to implement

Manuscript received May 18, 2013; revised October 6, 2013; accepted November 21, 2013. Date of publication January 9, 2014; date of current version November 20, 2014. This work was supported in part by the University of Zürich, ETH Zürich, and in part by Samsung Electronics Corporation.

The authors are with the Institute of Neuroinformatics, University of Zürich, ETH Zürich, Zürich CH-8057, Switzerland (e-mail: dneil@ini.phys.ethz.ch; shih@ini.phys.ethz.ch).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2013.2294916

a real-time, event-driven deep spiking network. Section II introduces prior work on FPGAs and spike-based neural network accelerators, followed by the theory of the event-based processing in Section III. Section IV discusses the specific design of Minitaur. In Section V, Minitaur's performance characteristics as well as real-world performance on the MNIST and newsgroup classification tasks will be evaluated. Finally, Section VI reviews future challenges and raises questions for further investigation.

II. Background

Significant work has been invested into software algorithms allowing the acceleration of artificial neural networks, and event-driven methods have emerged as one way of speeding up the simulation time of these networks. An efficient event-driven software implementation was described in [8], and the time complexity of scaling these networks was investigated in [9]. Event-driven optimizations have been considered for advanced neuron model implementations as well, notably including the Hodgkin–Huxley model in [10]. A comprehensive review of the performance of both event-driven and time-stepped software algorithms and implementations of spiking neural networks can be found in [11].

Hardware systems, which accelerate these spiking networks, including FPGA-based designs, are reviewed in [12] and [13]. These FPGA systems are predominantly time-stepped hardware accelerators as in [14]–[16], achieving high speeds but with performance proportional to the size of the network. Event-driven, sparser-computation hardware implementations, such as in [17]–[19] are rare, typically focusing on using biologically descriptive neuron models, such as the Izhikevich model [20] and biological network topologies.

In recent years, machine learning approaches have validated alternative network topologies, such as those used in DBNs [6], a multilayered probabilistic generative model. The individual layers consist of undirected graphical models called restricted Boltzmann machines (RBMs) with a bottom layer of visible sigmoidal units and a top layer of hidden sigmoidal units, bidirectionally connected with symmetric weights. When RBMs are stacked to form a DBN, the hidden layer of the lower RBM becomes the visible layer of the next higher RBM. DBNs have proved effective in a variety of domains, with notable successes in areas, such as machine vision [21] and machine audition [22], [23]. In [7], spiking DBNs are constructed by replacing these sigmoidal units with spiking leaky integrate-and-fire (LIF) neurons.

Minitaur extends this prior work on event-driven systems, FPGA implementations, and DBNs to accelerate spiking neural network implementations. It is a low-power, compact, event-driven system with a strong focus on spiking deep networks as an application domain. The system supports the loading of arbitrary spiking neural networks at runtime of up to 65 536 neurons and millions of synapses (Table II). With its focus on optimized memory fetches and simplified neuron models as in [24], as well as on low power dissipation, it eschews the high-power, high-performance memory elements used in [15] and [18]. Its performance is validated on two common machine learning tasks using a DBN composed of spiking neurons, as described in Section V.

TABLE I
Minitaur Parameters

Parameter	Size	Format
Simulation Parameters		
τ_m	16 bits	Fixed-Point (5.11)
t_{ref}	16 bits	Fixed-Point (5.11)
V_{thr}	15 bits	Fixed-Point (4.11)
Neuron State Parameters		
V_{mem}	16 bits	Signed Fixed-Point (5.11)
Timestamp	24 bits	Integer
t_{re}	16 bits	Integer
Address	16 bits	Integer
Refractory State	1 bit	Boolean
Connection Weight	16 bits	Signed Fixed-Point (5.11)

TABLE II
System Performance

Benchmark	Benchmark Results
Clock Rate	75 MHz
Power Consumption	1.1W (idle) - 1.5W (peak)
Average USB-to-USB Latency	236μs
Supported Number of Neurons	65536
Supported Number of Synapses	16.78 Million
Peak Performance	18.73 Million PSC/sec
MNIST Accuracy	92.00%
MNIST Performance	4.88 Million PSC/sec
20 Newsgroups Accuracy	71.07%
Newsgroups Performance	76 Thousand PSC/sec
Core 2 Duo P7350 2.0GHz CPU	
Power Consumption	20W (peak)
MNIST Performance	33.6 Million PSC/sec
Newsgroups Performance	173 Thousand PSC/sec

III. Algorithm and Theory

A. Event-Driven Neural Model

The neural model used here is a common spiking model containing three submodels: 1) a soma described by the LIF model; 2) an instantaneous synapse for input current; and 3) a fixed-delay axon for spike generation. The LIF model is both mathematically and intuitively simple; the cell membrane is modeled as a capacitor with a leak [25]. This simple circuit forms an exponentially decaying RC system with decay time constant τ_m. In a time-stepped model, the cell membrane voltage V_{mem} on the $n+1$th step can be calculated as follows:

$$V_{\text{mem}}(n+1) = V_{\text{mem}}(n) \cdot e^{-\Delta t/\tau_m} \tag{1}$$

where Δt is the time step.

The synapse model has instantaneous dynamics and a step increase of $W^{i,j}$ is added to the membrane potential of neuron i when it receives a spike from input neuron j. The model implements three discontinuities to more accurately model biological systems: 1) a threshold (V_{thr}) where a neuron makes a spike once $V_{\text{mem}} > V_{\text{thr}}$; 2) a reset potential for the membrane after a spike (V_{reset}); and 3) a refractory period (t_{ref}) during which a neuron cannot make a new spike after it spikes.

Algorithm 1 Time-Stepped Updating of an LIF Network

Given: N input neurons connected to M neurons
Given: $S^t \in R^{N \times M}$, a binary matrix where the elements indicate the presence or absence of a spike between a particular input neuron j to neuron i at time t
Given: set of all weights $W \in R^{N \times M}$
$t_{re}^{1..M} \leftarrow 0$ ▷ Previous refractory end time
for $t \leftarrow 0 : \Delta t : t_{final}$ **do** ▷ Loop through all time
  for $i \leftarrow 1$ to M **do** ▷ Loop through all neurons
    if $t > t_{re}^i$ **then** ▷ Ensure not refractory
      $V_{mem}^i \leftarrow V_{mem}^i \cdot e^{-\Delta t/\tau_m}$ ▷ Decay membrane
      $V_{mem}^i \leftarrow W^{i,1..N} \cdot S^{t,i,1..N} + V_{mem}^i$ ▷ Input
    end if
    if $V_{mem}^i > V_{thr}$ **then**
      DoSpiking()
      $V_{mem}^i \leftarrow V_{reset}$ ▷ Reset membrane potential
      $t_{re}^i \leftarrow t + t_{ref}$ ▷ Set refractory end
    end if
  end for
  $t \leftarrow t + \Delta t$
end for

Algorithm 2 Event-Driven Updating of an LIF Network

Given: set of all sorted input spike times Q_t
Given: set of corresponding destination neuron indices Q_{dest}
Given: set of corresponding source neuron indices Q_{src}
Given: set of all weights $W \in R^{N \times M}$
$t_{re}^{1..M} \leftarrow 0$ ▷ Reset refractory end time
$t_{prev}^{1..M} \leftarrow 0$ ▷ Reset previous input time
for k in length(Q_t) **do**
  $t \leftarrow Q_t^k$ ▷ Obtain spike time
  $i \leftarrow Q_{dest}^k$ ▷ Obtain index of neuron to update
  $V_{mem}^i \leftarrow V_{mem}^i \cdot e^{-(t - t_{prev}^i)/\tau_m}$ ▷ Decay membrane
  if $t > t_{re}^i$ **then**
    $V_{mem}^i \leftarrow V_{mem}^i + W^{i,Q_{src}^k}$ ▷ Add impulse
  end if
  if $V_{mem}^i > V_{thr}$ **then**
    DoSpiking()
    $V_{mem}^i \leftarrow V_{reset}^i$ ▷ Reset membrane potential
    $t_{re}^i \leftarrow t + t_{ref}$ ▷ Set refractory end
  end if
  $t_{prev}^i \leftarrow t$
end for

The complete time-stepped model can be found in Algorithm 1.

This algorithm can easily be transformed into an event-driven equivalent. Since the input current is instantaneous and the membrane potential decays away exponentially, it is only necessary to check for firing after the membrane potential has been updated when there is an input spike. The time of the previous update is stored; when the next spike arrives, the neuron membrane is decayed according to the time difference, and then summed with the instantaneous input current. This yields the neuron model used in the Minitaur system. The neuron only updates on input spikes, so the computation speed is now proportional to network activity, not numbers of neurons. The complete event-driven execution is described by Algorithm 2.

B. Simulation

A model of the hardware was created in MATLAB to ensure the viability of the design and quickly prototype the effects of parameter adjustment. This implementation was primarily used for prototyping caching strategies since memory bandwidth, rather than compute time, fundamentally limits the performance of the hardware. For more details on these strategies, see Section IV-B.

IV. Minitaur Architecture

A. Spartan-6 FPGA Architecture

Minitaur was designed using the low-cost Xilinx Spartan-6 platform. The full implementation was done on an ZTEX USB 1.15 board, which contains 128 MB of DDR2 RAM, a microSD card slot for storage, 128-kB flash memory for a bootloader, and an FX2 chip for USB interfacing. The commercially available complete board (http://www.ztex.de) is low cost and ideal for off-the-shelf interfacing and computation. The Xilinx Spartan-6 LX150 contains 150k logic cells, the largest of the Spartan-6 family.

In addition to the large number of logic cells, the Spartan-6 contains 180 DSP units for low-power parallel math processing. These DSPs support two 18-bit operands for fixed-point multiplication and addition. Importantly, the Spartan-6 has a total of 549 kB of memory in 268 individually addressable low-latency block RAMs (BRAMs). These BRAMs require one cycle for fetch and optionally one cycle to register the output of the fetch, making them ideal for core-specific caching.

The maximum supported clock speed of the Xilinx Spartan-6 is 400 MHz, though as on all FPGA devices, this number is heavily dependent on clock load and routing.

B. Minitaur Design Principles

The performance-limiting step in an event-based neural network system occurs during spike generation. When a neuron spikes, it performs two very memory intense operations: 1) determining the recipient neurons of the spike (between 10^2 and 10^4 destinations, typically) and 2) determining the weights of each of these connections. Accomplishing these two tasks quickly is of paramount importance in optimizing the system's performance.

To minimize the impact of connection lookups, rule-based connections are stored. Although true biological networks are typically recurrent and difficult to simplify, artificial neural networks tend to follow specific connectivity patterns. DBNs, autoencoders, single-layer RBMs, and multilayer perceptrons all have a very stereotyped structure. Namely, there is a layer of neurons receiving projections from the previous layer and projecting connections to the following layer, typically in an all-to-all fashion. A connection rule can be stored very efficiently by stating connections in a ranged-rule format, for

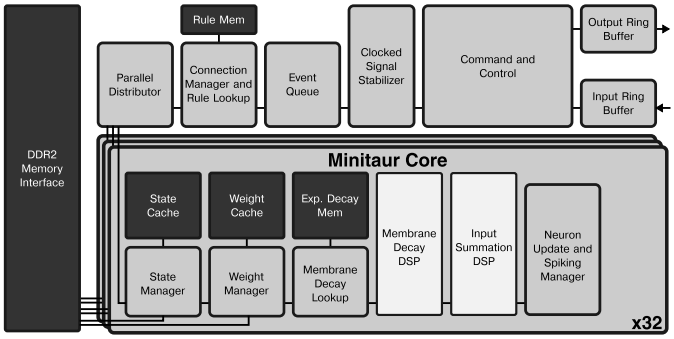

Fig. 1. Simplified architecture of the Minitaur system. It contains 32 parallel cores and 128 MB of DDR2 for main memory. Each core has 2048 kB of state cache, 8192 kB of weight cache, and two DSPs for performing fixed-point math (one multiplying the decay and one for summation of the input current). The exponential decay lookup uses 2048 kB of RAM, preloaded from the design and used as a ROM.

example, all neurons in layer 1 project to all neurons in layer 2, requiring only to store the SRC start, SRC end, DEST start, and DEST end addresses to map the connection for an entire layer.

Managing the multitude of outgoing postsynaptic currents (PSC) is also a challenge for a neural network accelerator system as neurons typically have a very large number of output connections. When storing spikes in a spike queue, instead of storing the addresses of neurons that are receiving spikes, it is vastly more efficient to store the addresses of neurons from which spikes are coming. In this way, the spike queue stores the SRC addresses, not the DEST addresses, and only performs the rule lookup to obtain the DEST address when evaluating the postsynaptic update after axonal delay.

Finally, cache locality is critical in optimizing neuron weight and state lookups. In an event-driven system that interacts with the real world, there are also likely to be significant patterns in the input data that can be exploited by the system. In the DVS event-driven retina system [3], for example, a pixel that sees an on event (on contrast change) is likely to have an off (off contrast change) event soon after because of the movement of a spatially extended object. These inherent correlations offer a major advantage over time-stepped systems, and there is no wasted computation as every input spike represents a change in the world, which necessitates an update of the output. For spatially similar input events, the weight and state values can be cached and fetched much more quickly. Ensuring locality was done by striping the neurons across the computational cores. The last five bits of the neuron ID assigns each neuron to a core, allowing a given core to store state and weights for frequent and recently active neurons without contention.

C. Minitaur Implementation

The simplified architecture diagram is shown in Fig. 1, and a list of the parameters and their formats can be found in Table 1. The system is designed to exploit commonalities in modern artificial neural networks to allow for greatly reduced computational load. Though Minitaur is a digital, clock-based system, no processing occurs except upon input events. This hybrid approach takes advantage of the ease-of-use of digital tools as well as the dramatic computation reduction of event-driven processing.

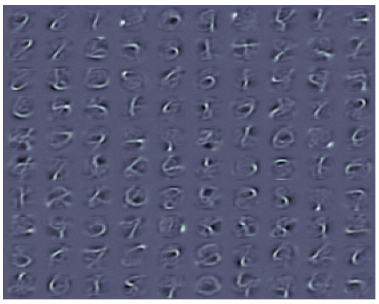

Fig. 2. Visualization of the weights between the first and second layers of the MNIST network [7] trained using rate neurons. This figure shows a sample of 100 neurons in the second layer, in which the incoming 784 weights are reshaped into a 28 × 28 pixel image. The weights shown are typical of MNIST-trained networks.

In this implementation, spikes (events) arrive over USB in packets, stamped with a four-byte timestamp, a one-byte layer indicator, and a two-byte neuron address. After passing through a ring buffer to allow for rapid bursts of spikes, the spikes are dispatched to the event queue where they are sorted by timestamp and layer. Time sorting is presumed only necessary with axonal delays and mixing spikes from external sources (e.g., spike-based neuromorphic devices) with on-Minitaur computation layers.

To support recurrent connections and axonal delays, a time delay must exist between when a neuron spikes and when that spike is delivered to its receiving neuron. The event queue maintains a sorted list of incoming spikes as a priority queue, which allows for $O(\log(n))$ operations of insertion and root-node extraction. The event queue uses a five-byte index key comprised of a four-byte timestamp and an one-byte layer index to store the neuron address. In this way, all spikes from the same layer simultaneously (a common occurrence with all-to-all connectivity and identical delays) will be sorted together and separated from the spikes of another layer simultaneously. Seven 2048-byte BRAMs are used to store up to 2048 events simultaneously. A flag ensures that spikes cannot be emitted from a layer until all inputs to that layer at that time have been evaluated.

Simultaneously, the parallel distributor block, which connects all the cores together in Fig. 1, checks the event queue to extract the first available spike for processing. Spikes are stored according to spike origin, not spike destination, so a connection lookup is necessary to determine which neurons will have membrane potential updates.

Each neuron is assigned an ID number and neurons in a given layer are assigned consecutive IDs. In this way, a connection can be very compactly represented: 4 bytes for the start and end of each of the SRC and DEST addresses yields a 16-byte range rule. To support multiple layer fanout, all possible rules are matched. Note that extremely complex, nonlayer-based networks can still be represented in this form using point-to-point connectivity. The number of comparisons is usually very small when describing a typical DBN for Minitaur's intended use; for example, five rules are sufficient to describe the MNIST handwritten digit identification network shown in Fig. 3 (one for each layer and one to map output to the computer). During a connection lookup, the source address is iteratively compared with all connection

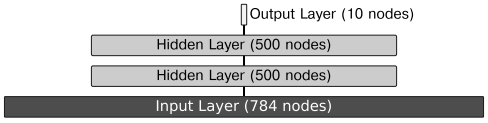

Fig. 3. Visualization of the network configuration used for the MNIST task.

rules in the rule memory, and if a rule is found with a source range containing the input spike address, the output range is passed as the range to compute.

The bottom five bits of the address of the output range are used to assign a particular neuron update to a particular core: for example, a neuron with address ID 1025 will always be assigned to core 1. This allows for cache locality and obviates any issues causing stale caches from other cores updating a given neuron. The parallel distributor assigns a batch of 32 neurons to be updated; waits until they are completed; and increments to assign the next batch of 32. This continues until the entire destination range of neurons has been addressed.

The event-driven algorithm (Algorithm 2) is executed by the core once a neuron is assigned. State fetching is done simultaneously with weight fetching to minimize latency, and as soon as the state is fetched it is passed to the DSP for computation. The state is stored as an eight-byte chunk: two bytes for the refractory end time, three bytes for the last-update timestamp, two bytes for state, and one byte for assorted information including the cache tag, an initialization bit, and a refractory bit. Because the DSP does not support exponentiation, a ROM lookup table is generated with exponential decay factors. Obtaining the time delta and dividing by the membrane time constant yields an integer lookup index in this memory, and the value at the address j, of 1024 addresses, is $e^{-j/128}$, approximated to 16 bits of fixed-point accuracy (five integer bits and 11 fractional bits). This yields an accurate decay range from $1/128\tau$ to 8τ in steps of $1/128\tau$. The weight is stored as two bytes in the same fixed-point format. During neuron updates, the membrane is decayed, followed by an impulse according to the weight of the input neuron, and the potential is compared with the threshold. If the threshold is exceeded, then a flag is raised by the core. After arbitration and depending on system parameters, that spike is either assembled and sent to the computer or it is added to the spike queue with an axonal delay, where it will be sorted according to its time and layer.

Fast local memory is a key for optimizing neural network computations by minimizing the impact of weight lookups. This cache is implemented with a variant of the direct-mapped cache algorithm: each neuron or weight lookup has only one location that it can be mapped to, and at each location, a reference counter tracks the number of consecutive misses. The penalty for a DDR2 swap is not as severe as a hard disk, so occasional contention for a specific memory location is an acceptable tradeoff for very fast lookup.

For neuron state cache lookups, the eight-bit cache address is formed using the middle part of the neuron's address. The lower five bits are common to all the neurons at a given core due to the computational partitioning; the upper three bits are used as a tag. Combined together, the core-specified five bits, the cache address's eight bits, and the tag's three bits recovers the entire 16-bit neuron address. In the case of a cache hit (i.e., a value was successfully retrieved from cache), a reference bit is set to one; in the case of a miss (i.e., the value is not cached and must be fetched from main memory), the bit is cleared, and if already cleared, the entry is swapped out. The neuron state is stored in eight bytes including the tag, allowing for 256 entries per core using one 2-kB BRAM.

For neuron weight lookups, a two-stage approach is used. Optimized for fanout, the local weight cache is separated into blocks of 16 entries. Each block of 16 has a single SRC neuron address and up to 15 DEST neuron addresses. On a cache lookup, the lower seven bits of the SRC address are used to index the cache; if the SRC matches, then the four lower bits of the DEST addresses are used to calculate a jump offset from the initial SRC address. If the DEST address matches, then the weight is retrieved from the cache. Here, the reference bit for the SRC address is two bits since up to 15 DEST entries could be swapped out at once; thus, four consecutive misses are required to swap an address out. The DEST address uses only one reference bit, requiring two misses to swap out.

V. Result

The completed Minitaur design uses 22k logic slices, 23% of the capacity of the Spartan-6 LX150 FPGA. The power consumption of the FPGA is 1.5 W, of which 400 mW supports electronics external to the FPGA. Of the on-chip power budget, 10.0% supports logic and signals, 16.2% supports the 200 (of 268) block memories heavily used as cache to speed up the system, and 73.8% is due to the PLLs, clock distributors, IOs, and leakage. The system makes use of primarily three clocks: the DDR2 clock at 132 MHz, the USB I/O clock at 48 MHz, and the logic clock operating at 75 MHz. Minitaur is an early stage device, and many further optimizations can be made to significantly increase its performance. A summary of the results can be found in Table II. Following [12], the connections per second, or more specifically, the PSCs/second, was chosen as the primary performance metric.

The performance statistics were gathered on an Intel Core 2 Duo P7350 clocked at 2.00-GHz running Ubuntu Linux 12.04. Minitaur was connected via USB using the Java libusb device wrapper and received input spikes from benchmarking software on the computer for performance benchmarking, the MNIST task, and newsgroup classification task. In addition, the CPU PSC/second performance statistics were calculated using a high-performance parallel MATLAB implementation of LIF neuron networks, rather than a block-wise model of Minitaur, to ensure a fair peak performance comparison.

A. Data Sets

1) MNIST Handwritten Digits: The system was tested extensively with the well-studied MNIST benchmark of handwritten digits [26]. Where not otherwise specified, the performance results were obtained using the full test set of 10 000 handwritten digits after training on the full 60 000 digit training set. To convert the static images into events, the 28×28 images were vectorized into 784 neuron addresses and spikes were sent with probability proportional to the intensity of the pixel.

The final feedforward network was 784–500–500–10 units in size (Fig. 3). Since each layer is connected in the standard all-to-all fashion, this yields 647 000 synapses in this task and 1785 neurons. Using weights previously trained to achieve

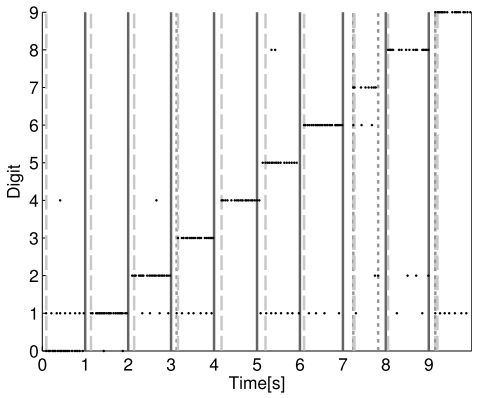

Fig. 4. Example real-time run of digit identification, with an output spike represented by a black dot. Each digit was presented in order, from zero to nine, for a duration of 1 s during which 1000 spikes were presented; the probability of an input spike for a given pixel is proportional to the pixel's intensity. The winning digit is chosen according to an exponentially decaying histogram ($\tau = 0.11$ s). Dark dotted line: a transition to a selection of an incorrect winning digit. Lighter dashed lines: a transition to the correct choice for that digit. For the trial shown here, the average time to transition to a newly selected digit after a change in the input digit was 0.152 s.

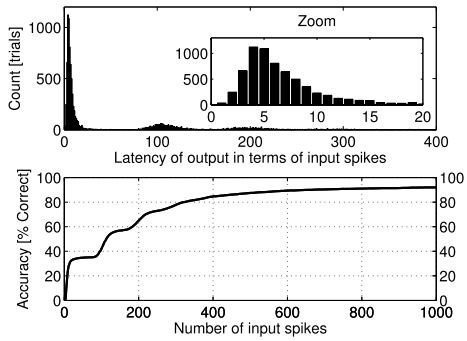

Fig. 5. Increasing accuracy with additional information, using the complete 10 000 digits in the MNIST test set. For an event-based system, the natural unit of time is number of input events, not seconds. Each input event refines the answer estimate in the same way a long exposure time or multiple frames accumulate evidence in a time-stepped model. Moreover, latency is measured using input events because the system cannot produce an answer without accumulated information added to the system. The top plot shows a histogram of latency until the first output spike. Most of the trials produce a result spike after four input spikes (as seen in the zoomed-in inset), but some trials can take hundreds of inputs to produce their first output spike. The bottom plot shows the effect of adding more events in the MNIST task. Additional spikes cause the accuracy to asymptotically approach a 92% value.

94.2% accuracy on the MNIST task with LIF neurons in software [7] and shown in Fig. 2, Minitaur achieved 92% accuracy with 1000 spikes per image; the effects of different input volume per image are discussed later in this paper (Fig. 5). The loss of accuracy is likely due to the fewer bits for representation of the weights, as the computer performs the task with doubles (eight bytes) while Minitaur uses just two bytes for a neuron's weight. To ameliorate this, future versions can have a reduced-accuracy training paradigm to train weights that balance each other more accurately using less-precise representations, rather than simply truncating the more accurate representations.

The output behaviour of an example real-time execution of the Minitaur system on the MNIST classification task can be found in Fig. 4.

2) Newsgroups Data Set Classification Performance: Furthermore, to demonstrate the runtime configurability of Minitaur, a nonvisual large data set was selected in addition to the standard MNIST task. The 20 newsgroups data set is a collection of approximately 20 000 documents evenly partitioned across 20 newsgroup forums, collected in 1995 [27] and commonly used in text classification and clustering. As shown in Table III, several of the document types are very closely related, while others are more distant. The documents are presented in a bag-of-words model: each document is represented with a sparse vector of word counts.

Typically, the word counts of these documents are transformed by applying the term frequency-inverse document frequency (TF-IDF) transform to control for commonly used words and identify the more salient usage of infrequent words. This preprocessing step, however, requires both additional transformations and knowledge of the complete data set. Minitaur is designed to save computation time and the system should operate with as a little preprocessing and unnecessary computation as possible. Furthermore, word counts must be transformed into events to be used in the system.

In the approach used here, an event corresponding to each word is emitted whenever that word is encountered in text. Time no longer has a concrete meaning in this domain so it was chosen to assign random, small time steps to each additional word spike. In this way, the network represents the semantic context of the document, which gradually either decays away or accumulates according to the types of words that are presented.

TABLE III
NEWSGROUP CLASSIFICATION

Newsgroup	Top Positive Terms	Perf
talk.religion.misc	order, bull, brian	44.22%
talk.politics.misc	kaldis, cramer, tax	46.45%
sci.electronics	circuit, electronics, hc	56.74%
comp.sys.ibm.pc.hardware	gateway, dx, bus	61.73%
comp.windows.x	motif, server, widget	61.79%
rec.autos	car, cars, dealer	64.30%
comp.os.ms-windows.misc	windows, win, cica	65.47%
comp.sys.mac.hardware	mac, apple, powerbook	66.58%
comp.graphics	graphics, image, polygon	66.84%
soc.religion.christian	rutgers, athos, christ	70.10%
talk.politics.guns	gun, guns, weapons	70.33%
sci.med	disease, doctor, msg	70.48%
alt.atheism	keith, atheism, mathew	72.64%
talk.politics.mideast	israel, israeli, turkish	74.47%
sci.space	space, orbit, launch	76.02%
misc.forsale	sale, offer, shipping	78.80%
rec.sport.baseball	baseball, phillies, runs	82.62%
rec.motorcycles	dod, bike, motorcycle	86.90%
rec.sport.hockey	hockey, nhl, playoff	87.47%
sci.crypt	encryption, clipper, key	87.59%

The network used in this case is a simple two-layer network of size 10 000–20, yielding a network of 10 020 neurons and 200 000 synapses. The number of neurons in the input layer is equal to the number of words used (10 000 in this experiment), and the number of neurons in the output layer equals the number of distinct classes (20 distinct newsgroups). The network was trained using standard backpropagation combined with dropout [28].

Without transforming the input using TF-IDF, word counts were emitted as word spike counts, and using the standard 60% train, 40% test split on these documents, the Minitaur system achieved 71% classification accuracy using the 10 000 most frequently used words in the data set. The breakdown by category with the most common words is shown in Table III.

The PSC/second performance of both the CPU and the Minitaur system decreased with the newsgroup classification task. A given news item to be classified may only have 100 noncommon words, and the fanout for each of these words is only 20. This means that Minitaur cannot fully use all 32 cores during computation of neuron updates. In addition, the system has suboptimal caching from the significant weight convergence (10 000 nodes to 20), which limits the number of SRC neurons that can have cached weights. The CPU-based approach suffers as well because the parallelism between trials is low; a given news item may only have 100 noncommon words while another might have 10 000, so the CPU is not able to parallelize as many trials simultaneously as in the MNIST task.

B. System Performance

The current design has a benchmarked USB-to-USB latency of 236 μs (averaged over 10 000 trials), which is primarily dominated by the latency of the operating system issuing USB read and writes. Minitaur was benchmarked at processing 585 kevts/s or one input spike every 1.71 μs with all memory fetches drawing from local cache. With each input spike causing 32 PSC (fully using the parallel cores), Minitaur processed 18.73 million PSC/second at its peak speed.

C. Initial Response and Additional Accuracy

Minitaur can be used to abort a computation early when sufficient accuracy is reached. When operating on a fixed input, event-based computation is a process of refinement rather than a static computation. Sequential input events add information to the system, and the system accumulates evidence over time to arrive at a more accurate answer.

This implies that an embodied robotic platform using Minitaur could use Minitaur's initial output after a very low response time to achieve a low-quality guess, or pay a small time cost to allow subsequent processing to increase the accuracy of that guess. In the MNIST task, 59.2% of the first output spikes (not shown) showed a correct answer occurring after the delay represented in the top half of Fig. 5. This would allow the system to make a low-accuracy guess after a very short delay. The bottom of Fig. 5 shows the increased accuracy of the system as the number of input events increases.

Both early abort and longer refinement use cases have obvious applications in robotics, and the freedom to choose at runtime is a major advantage of the Minitaur system.

D. Noise Robustness and Indecision

With the fallibility of sensors in general, and the likelihood of unexpected events in real-world data sets, robustness to noise is a significant part of designing a real-time event-driven system. To test the robustness of the system to noise, the MNIST data set was employed with varying noise levels.

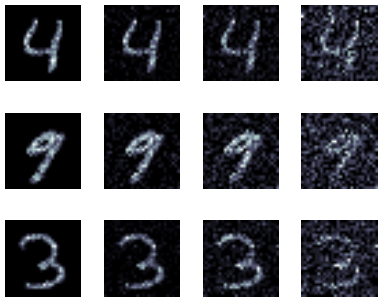

Fig. 6. Visualization of three digits from the MNIST data set with noise added. From left to right: 0% noise, 30% noise, 55% noise, and 80% noise for example handwritten digits four, nine, and three. Noise spikes were drawn uniformly from the pixel space and used to replace informative spikes.

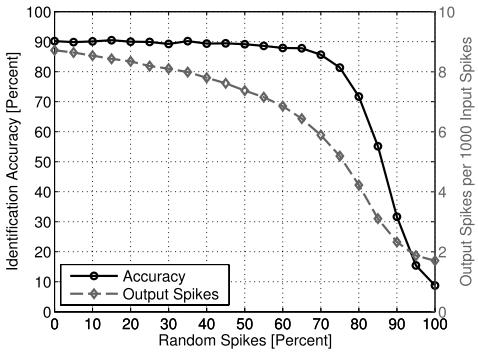

Fig. 7. System performance is robust to noise. Even when the input is only 20% signal and 80% noise, the event-driven system still correctly classifies the digits with more than a 70% success rate. This is largely due to the robustness of RBMs to uniform noise. Since no particular distribution is favored by uniform noise, it does not strongly affect the result. The increased noise of the data does create more indecision in the result. The number of output spikes drops dramatically with increased noise and accounts for the falling accuracy.

As before, spikes are drawn from the image with probability proportional to pixel intensity. Then, a percentage of spikes is subsequently replaced with spikes from random pixels, drawn uniformly from the pixel space, and the accuracy of the system is calculated (Fig. 6). As shown in Fig. 7, the system is very robust to noise due to the weights of the RBM, which act to denoise the input by keeping only significant features as events propagate through the layers. Interestingly, the number of output spikes drops dramatically with increased noise; when receiving 90% noise, the system will average just over two output spikes for 1000 input spikes. The decrease of spikes has practical advantages as well; a downstream system using the output of Minitaur will be signaled with fewer spikes since Minitaur is less confident of its result.

VI. Conclusion

In this paper, the authors have introduced the Minitaur spiking network accelerator. In addition to the system's performance of 18.73 million PSCs/second, it consumes just 1.5 W of power, enabling it to be used in embedded robotics applications. The system records 92% accuracy on the MNIST handwritten digit classification and 71% accuracy on the 20 newsgroups classification data set. With proper weights, the system is remarkably robust to noise. In addition, the knowledge about the output spikes can be used to determine how difficult a task is, and to weigh the confidence of the output accordingly.

A significant challenge of using this system right now is the dearth of effective training methods for LIF-spike-based systems. Various approaches for learning the weights for spike-based LIF networks, in particular DBNs, are being explored [7] especially where such networks prevent traditional training regimens of real-valued sigmoidal activation functions and backpropagation. Further work on event-based learning is needed to improve training and runtime accuracy significantly.

The system raises questions for further research about event-based processing in the real world. Systems with constrained resources to acquire information, process it, and subsequently act on it can benefit from event-based processing. Progressive refinement can allow the system to gather as much information as it needs to perform an action, determine a confidence of its analysis, and act according to that level of confidence. In closed-loop systems interacting in the real world, this can greatly optimize resource usage, and Minitaur's robustness to noise bolsters confidence that this can be used in real-time systems.

ACKNOWLEDGMENT

The authors would like to thank P. O'Connor, M. Pfeiffer, and T. Delbruck for discussions and inspiration.

REFERENCES

[1] T. Delbruck, "Frame-free dynamic digital vision," in *Proc. Int. Symp. Secure-Life Electron., Adv. Electron. Quality Life Soc.*, 2008, pp. 21–26.

[2] C. Farabet, Y. LeCun, K. Kavukcuoglu, E. Culurciello, B. Martini, P. Akselrod, *et al.*, *Large-Scale FPGA-Based Convolutional Networks*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[3] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128 × 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[4] S.-C. Liu, A. van Schaik, B. Minch, and T. Delbrück, "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 2027–2030.

[5] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion Neurobiol.*, vol. 20, no. 3, pp. 288–295, 2010.

[6] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[7] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers Neurosci.*, vol. 7, no. 178, p. 1, Oct. 2013.

[8] A. Delorme and S. Thorpe, "SpikeNET: An event-driven simulation package for modeling large networks of spiking neurons," *Netw., Comput. Neural Syst.*, vol. 14, no. 4, pp. 613–627, 2003.

[9] I. Marian, R. Reilly, and D. Mackey, "Efficient event-driven simulation of spiking neural networks," in *Proc. 3rd WSES Int. Conf., Neural Netw. Appl.*, 2002, pp. 1–6.

[10] C. Lobb, Z. Chao, R. Fujimoto, and S. Potter, "Parallel event-driven neural network simulations using the Hodgkin-Huxley neuron model," in *Proc. Workshop PADS*, 2005, pp. 16–25.

[11] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, *et al.*, "Simulation of networks of spiking neurons: A review of tools and strategies," *J. Comput. Neurosci.*, vol. 23, no. 3, pp. 349–398, 2007.

[12] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1, pp. 239–255, 2010.

[13] L. Maguire, T. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin, "Challenges for large-scale implementations of spiking neural networks on FPGAs," *Neurocomputing*, vol. 71, no. 1, pp. 13–29, 2007.

[14] D. Thomas and W. Luk, "FPGA accelerated simulation of biologically plausible spiking neural networks," in *Proc. 17th IEEE Symp. FCCM*, Apr. 2009, pp. 45–52.

[15] A. Cassidy, A. Andreou, and J. Georgiou, "Design of a one million neuron single FPGA neuromorphic system for real-time multimodal scene analysis," in *Proc. 45th Annu. CISS*, 2011, pp. 1–6.

[16] B. Leung, Y. Pan, C. Schroeder, S. O. Memik, G. Memik, and M. Hartmann, "Towards an 'early neural circuit simulator': A FPGA implementation of processing in the rat whisker system," in *Proc. Int. Conf. FPL*, Sep. 2008, pp. 191–196.

[17] K. Cheung, S. Schultz, and P. Leong, "A parallel spiking neural network simulator," in *Proc. Int. Conf. FPT*, 2009, pp. 247–254.

[18] K. Cheung, S. R. Schultz, and W. Luk, "A large-scale spiking neural network accelerator for FPGA systems," in *Proc. 22nd ICANN*, 2012, pp. 113–120.

[19] R. Agis, E. Ros, J. Diaz, R. Carrillo, and E. M. Ortigosa, "Hardware event-driven simulation engine for spiking neural networks," *Int. J. Electron.*, vol. 94, no. 5, pp. 469–480, 2007.

[20] E. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[21] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, 2010.

[22] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 14–22, Jan. 2012.

[23] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. INTERSPEECH*, 2011, pp. 437–440.

[24] T. Schoenauer, N. Mehrtash, A. Jahnke, and H. Klar, "MASPINN: Novel concepts for a neuroaccelerator for spiking neural networks," *Proc. SPIE*, vol. 3728, pp. 87–96, Mar. 1999.

[25] N. Brunel and M. C. W. van Rossum, "Lapicques̀ 1907 paper: From frogs to integrate-and-fire," *Biol. Cybern.*, vol. 97, no. 5, pp. 337–339, 2007.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[27] K. Lang, "NewsWeeder: Learning to filter netnews," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 331–339.

[28] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," Ph.D. dissertation, Dept. Comput Sci., Univ. Toronto, Toronto, ON, Canada, Jul. 2012.

Daniel Neil received the B.S. degree in biomedical computation from Stanford University, Stanford, CA, USA, and the master's degree in the neural systems and computation program from the Institute of Neuroinformatics, Zurich, Switzerland. He is currently pursuing his Doctoral degree at the Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich.

He was formerly a Research Assistant in Kwabena Boahen's Brains in Silicon Laboratory, Stanford University. He also worked as a technical consultant in the San Francisco Bay Area. His current research interests include scalable computing architectures for machine learning, with an emphasis on inspiration from biology.

Shih-Chii Liu (M'02–SM'07) received the B. S. degree in electrical engineering from MIT and the Ph.D. degree in the computation and neural systems program from the California Institute of Technology, Pasadena, CA, USA.

She worked at various companies, including Gould American Microsystems, San Jose, CA, USA, LSI Logic, Sherman Oaks, CA, USA, and Rockwell International Research Laboratories, Thousand Oaks, CA, USA. She is currently an Oberassistentin with the Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland. Her current research interests include neuromorphic visual and auditory sensors, cortical processing circuits, and event-based circuits and algorithms.

Dr. Liu is a Past Chair of the IEEE CAS Sensory Systems and Neural Systems and Applications Technical Committees. She is currently a Chair of the IEEE Swiss CAS Society, and an Associate Editor of the IEEE TRANSACTIONS OF BIOMEDICAL CIRCUITS AND SYSTEMS and the *Neural Networks Journal*.