Title

Eciton: Very Low-Power LSTM Neural Network Accelerator for Predictive Maintenance at the Edge

Authors

Chen, Jeffrey
Hong, Sehwan
He, Warrick
et al.

Publication Date

2021-01-03

DOI

10.1109/fpl53798.2021.00009

Eciton: Very Low-Power LSTM Neural Network Accelerator for Predictive Maintenance at the Edge

Jeffrey Chen*, Sehwan Hong*, Warrick He†, Jinyeong Moon‡, Sang-Woo Jun*
*Department of Computer Science, University of California, Irvine
†Diamond Bar High School
‡Department of Electrical and Computer Engineering, Florida State University
Email: *{jeffrc2, sehwanh, swjun}@uci.edu
†warrickhe@gmail.com ‡j.moon@fsu.edu

***Abstract*—This paper presents Eciton, a very low-power LSTM neural network accelerator for low-power edge sensor nodes, demonstrating real-time processing on predictive maintenance applications with a power consumption of 17 mW under load. Eciton reduces memory and chip resource requirements via 8-bit quantization and hard sigmoid activation, allowing the accelerator as well as the LSTM model parameters to fit in a low-cost, low-power Lattice iCE40 UP5K FPGA. Eciton demonstrates real-time processing at a very low power consumption with minimal loss of accuracy on two predictive maintenance scenarios with differing characteristics, while achieving competitive power efficiency against the state-of-the-art of similar scale. We also show that the addition of this accelerator actually reduces the power budget of the sensor node by reducing power-hungry wireless transmission.The resulting power budget of the sensor node is small enough to be powered by a power harvester, potentially allowing it to run indefinitely without a battery or periodic maintenance.**

I. Introduction

Cyber-Physical Systems (CPS), coupled with the Internet-of-Things (IoT), enable deeper, data-driven analytics and insight into our physical world by employing swarms of small, low-power *sensor nodes* communicating over a wireless network. The real-world impact of CPS/IoT is rapidly growing, and are impacting a wide range of areas including manufacturing [32], agriculture [18], and healthcare [45], [44].

Predictive maintenance is a prominent emerging application of CPS, which takes advantage of fine-grained data collection to predict future failures of mechanical devices. It has been quickly gaining adoption as accurate predictions can significantly reduce downtime and maintenance cost [40]. A popular method is to use statistical and machine learning to mine information from streams of sensor data, such as vibration, acceleration, and temperature, collected from various locations of a mechanical device [17], [31], [40], [51], [20]. Long Short Term Memory (LSTM) recurrent neural networks have proven especially effective [7], [4], [47].

Low power and energy consumption is one of the key goals in the design of a CPS deployment, as deployment scale or non-intrusive sensing requirements can limit access to power infrastructure [18], [31]. As a result, nodes are often expected to function on a scale of years powered by small batteries [35] or power harvesting modules [31], [27], [2]. These constraints limit not only the overall energy consumption, but also the power consumption at any given time. This limits the amount of computation available on a node, as well as the communication bandwidth over power-hungry wireless communication. In fact, the primary limitation of data collection capacities for low-power nodes is often the power consumption of wireless data transmission [33], [5], [9], [25], [13].

Edge Computing attempts to reduce the pressure on the wireless network by moving some computation to the nodes to distill data to smaller sizes. On the one hand, many real-world applications have shown that filtering at the edge can reduce the network data transmission requirement by over 95% [11]. However, provisioning enough computation performance on end nodes can be costly in terms of power consumption and hardware, defeating the purpose of edge computing.

Hardware accelerators can often remedy this problem by achieving high performance on a low power budget [23], [54], [37], [21]. But while existing FPGA-based LSTM accelerators achieve high performance and power efficiency, even they regularly require hundreds of mW to multiple Watts of power, straining the limits of the stringent power budgets of edge nodes [14], [8], [28], [26], [6].

In this paper, we present **Eciton**, an extremely power-efficient FPGA-based accelerator for time-series mining using neural networks at the edge. We show that Eciton can perform real-time inference of Long Short Term Memory (LSTM) neural network models of realistic size, at a power budget of 17 mW. Eciton demonstrates competitive performance compared to more power-hungry FPGA implementations, as well as competitive power efficiency against the state-of-the-art. To the best of our knowledge, no prior FPGA LSTM accelerator has demonstrated lower power consumption numbers, especially while maintaining performance on models of similar scale.

Eciton employs 8-bit fixed-point quantization of weights, hard sigmoid activation functions, as well as a carefully optimized microarchitecture to reduce the chip resource and memory requirements. These quantization approaches results in a modest accuracy loss of ˜5% when evaluated on real-world predictive maintenance LSTM models with 3- to 4-layers. Meanwhile, the low resource requirements means Eciton can fit in a low-power, low-cost, but resource-constrained Lattice iCE40 UP5K FPGA. The UP5K was uniquely positioned for

978-1-6654-3759-2/21/$31.00 ©2021 IEEE
DOI 10.1109/FPL53798.2021.00009

this application, thanks not only to its low cost (~$5) and μW-scale static power consumption, but also relatively large on-chip SRAM resources (128 KiB), which are unavailable in similarly positioned low-power FPGAs such as the Microsemi IGLOO or other Lattice iCE40 chips.

This paper claims the following contributions:

- Demonstrates an LSTM accelerator design which fits in a low-power FPGA and a 17 mW power budget.
- Demonstrates performance and accuracy on predictive maintenance models of realistic scale.

The rest of this paper is organized as follows: Background and related works are explored in Section II. We present the architecture of Eciton and our neural network compression methods in Section III. We provide in-depth evaluation of Eciton in the context of a CPS deployment of predictive maintenance in Section IV. We conclude with future work in Section V.

II. Background and Related Works

A. Predictive Maintenance

Predictive maintenance, or condition-based maintenance, aims to monitor mechanical systems for signs of imminent failure in order to preemptively perform maintenance operations. Compared to conventional, periodic maintenance, accurate predictive maintenance can significantly reduce downtime and cost [17]. Modern predictive maintenance approaches integrate pervasive data collection of CPS and IoT into target mechanical devices to achieve more accurate prediction compared to manual inspection [17]. Depending on the mechanical properties of the target, information including vibration, humidity, temperature, pressure, sound, electrical current, and inductance can be collected from various locations in the device, which are analyzed using various statistical and machine learning-based approaches [17], [31], [40], [51], [20].

B. LSTM Recurrent Neural Network

Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture that can better handle long-term dependencies in the input data. Classical RNNs suffer from the so-called *vanishing gradient* problem wherein the effects of an earlier input quickly become too small to be useful. LSTMs handle this problem by controlling gradient degradation using three *gates*. This adds three more trainable weights per cell, which can learn the best rate of gradient decay. As a result, LSTMs can learn relationships between events happening millions of time steps apart [39], making them useful for many real-world time series data mining applications [48], [49].

Figure 1 shows the internal architecture of a single LSTM cell. Each weight ($f, i, \widetilde{c}$, and o) represents a multiplication and accumulation operation against both input and recurrent values. LSTMs also use two classes of activation functions: kernel and recurrent activation functions. As seen in Figure 1, typically used activation functions are sigmoid (σ) and tanh, respectively.

LSTMs are typically designed with multiple layers, as well as a *dense* layer at the top layer. The dense layer is a conventional fully-connected neural network layer, usually with the same number of units as the number of classes to detect. A softmax of the output of the dense layer can be used to determine the predicted class of the neural network.

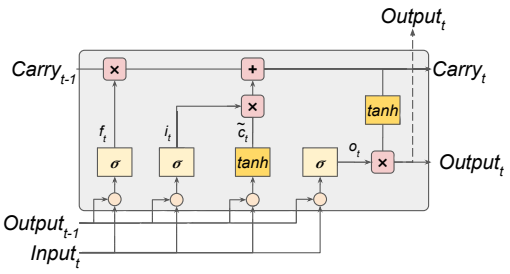

Fig. 1. Long-short term memory cell architecture

C. Neural Network Compression

Compressing neural networks, either by reducing the number of weights (*pruning*) or the bit width of each weight values (*quantizing*), is an important tool for reducing the computation and memory overhead of neural networks, often with negligible loss of accuracy [15]. Pruning involves removing weights with less impact on the final result, and then re-training the model without these weights. Quantization maps the range of original values to a smaller set of values using either a linear or non-linear mapping function. A variance-based cutoff may also be used to limit the range of the source set in order to account for outliers. A popular instance of quantization is to map 4 byte floating-point to 1 byte fixed-point. This not only reduces memory requirements, but also substitutes costly floating point operations with cheaper fixed-point arithmetic [43], [22], [52].

D. Accelerators on CPS End Nodes

Augmenting CPS/IoT end nodes with reconfigurable hardware accelerators such as FPGAs can help reduce the power consumption of required computation, as well as power-efficiently implement edge computing on end nodes [10], [54]. Existing research has explored offloading end node operating system functions [36], encryption for security [42], and communication protocols [12], as well as application-specific accelerators for video processing [23], [54], person detection [37], and statistical time series mining [21] into FPGAs. Much research has gone into optimal management of FPGAs on end nodes as well, including dynamic partial reconfiguration and power gating [46]. Industry offerings also exist for low-power convolutional neural networks at the edge, including the Lattice SensAI platform [41].

Power-efficient neural network implementation on FPGAs for the edge has also been a popular research topic [53], [16], [19], including LSTMs [28], [26]. However, performance and model size requirements have prevented realistic LSTM models from fitting in the low-power class FPGAs such as Lattice iCE40 or Microsemi IGLOO series, limiting them to relatively performance-oriented FPGAs with 100s of mW of static power. This may limit their use in resource-constrained CPS/IoT scenarios with mW-level power budgets [31], [2].

E. Wireless Communication For CPS/IoT

There is a wide range of available wireless communication technologies for CPS and IoT systems [13], [29]. They typically represent a trade-off between communication bandwidth, range, and cost/power consumption, spanning from LTE or GSM-based technologies like LTE-M and MB-IoT with hundreds of kbps of performance as well as hundreds of mW of power consumption [25], to Low-Power WAN (LPWAN) technologies like LoRa with an order of magnitude bandwidth and power consumption [9], [1]. Reducing network transmission requirements with effective edge computing may allow use of low-power technologies while still maintaining effective data collection bandwidth.

III. Eciton Node Architecture

Eciton augments a CPS/IoT end node with a flexible LSTM accelerator implemented on a low-cost, low-power FPGA. Collected sensor data is first evaluated by the LSTM accelerator, and the sensor data is sent wirelessly to a central server only when potential failure is detected locally. At the central server, a more complex software system can then make higher-level decisions based on data from multiple nodes.

Figure 2 shows the overall architecture of an Eciton end node. Sensor data is first filtered by the FPGA, reducing the data traffic and workload of the microcontroller unit (MCU). Due to the complexities of managing the network, the MCU software is currently responsible for the network interface.

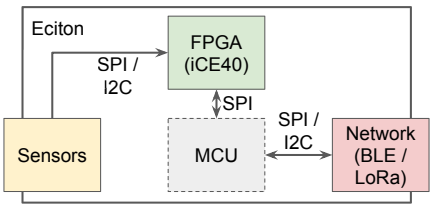

Fig. 2. Eciton end node architecture

Internally, the LSTM accelerator consists of two separate cores for LSTM and dense layers, as seen in Figure 3. The MCU is responsible for loading the LSTM and dense layer cores with weights, after which sensor data can stream through both cores for inference.

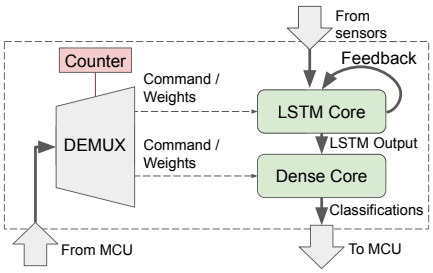

Fig. 3. The MCU provides weights to LSTM and Dense cores, after which the accelerator can process input from sensors

A. FPGA Resource Restrictions

Our target platform is the Lattice iCE40 UP5K FPGA, which is a low-cost (~$5), low-power ($\mu W$-scale static power) FPGA with stringent resource restrictions. This choice was driven by the existence of 1024 kbits of on-chip SRAM provided in the form of four Single-Port RAM (SPRAM) blocks, in addition to usual embedded Block RAM. Most other FPGA offerings of similar class, including the Microsemi IGLOO [30] or other Lattice iCE40 FPGAs [24], do not include this, limiting their on-chip memory to mere 32 to 128 kbits of block RAM. However, even this additional memory is not enough for neural network models of realistic size. Since off-chip memory access will be harmful for both performance and power efficiency, we need to employ neural network compression to make the model fit entirely on-chip.

Furthermore, the UP5K provides 8 16-bit Digital Signal Processing (DSP) blocks for fast arithmetic. While this is a larger number compared to similar offerings, it is nowhere enough to implement fast floating point operations, or even wide integer division. Our selection of quantization and hard sigmoid activation function parameters was the result of optimizing performance and accuracy within these constraints.

B. Quantizing LSTM Models

Eciton performs post-training linear quantization of weights from 4-byte floating point to 8-bit dynamic fixed point, reducing memory footprint by $1/4$. We discovered 8-bit quantization resulted in the best size-accuracy trade-off, which will be presented in more detail in Section IV-B.

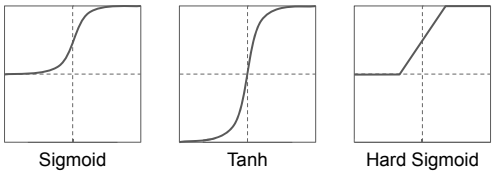

Fig. 4. Comparison between three activation functions

In order to reduce chip resource requirements, Eciton uses the linear *hard sigmoid* activation function for both kernel and recurrent activation functions of LSTM, instead of non-linear functions such as sigmoid or tanh. Figure 4 shows the three activation functions. While non-linear functions were too complex to fit on our target FPGA, we discovered that hard sigmoid can be implemented efficiently while providing a more accurate approximation of the non-linear functions, compared to simpler functions such as the Rectified Linear Unit (ReLU).

C. LSTM Core Architecture

Figure 5 shows the LSTM core architecture. Input and output is done in units of 8-bit words. All weights are stored across the four available on-chip SPRAM blocks combined into a single address space, exposing a single 16-bit interface. The computation pipeline consists conceptually of two sub-pipelines: The Multiply-Accumulator (MAC) pipeline, and the

State pipeline for calculating carry and hidden states. Both pipelines share a single quantized ALU Module.

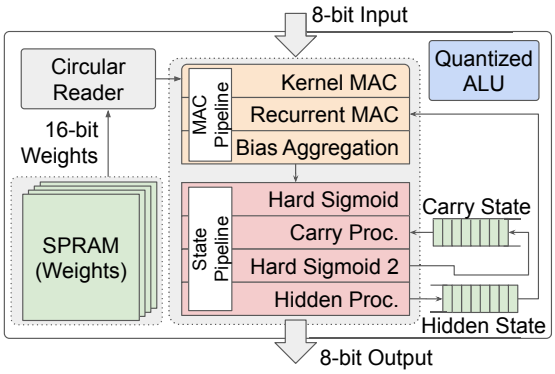

Fig. 5. Microarchitecture of the LSTM Core

1) Weight Memory Map: Weights are stored in a layer-major order, so for each input tuple to propagate through all layers, the whole weight memory is linearly scanned exactly once by a very simple circular reader module.

Figure 6 shows an example memory layout of weights for a model with two LSTM layers. The weights are stored in the order of kernel weights, recurrent weights, and bias weights, for each LSTM layer. Each weight is a four-byte tuple, corresponding to the four 1-byte weights of each LSTM cell. The microarchitecture orders computation in the same order, so that weights can simply be linearly scanned for correct operation.

2) MAC Pipeline: The MAC pipeline is where the majority of the arithmetic for LSTM is done. While computation is conceptually organized as a pipeline, Eciton actually processes each step in non-overlapping sequence to efficiently reuse a small number of quantized arithmetic units. Computation is organized in the order of weight storage, i.e., *kernel*, *recurrent*, and *bias* weights.

For kernel weights, each 8-bit input can be independently multiply-accumulated against each of the four kernel weights of all LSTM cells in this layer. This results in $Unit$ number of four-byte tuples in a BRAM FIFO, where $Unit$ is the number of LSTM cells in this layer. By the time kernel MACs are done, the recurrent weights are already loaded and ready in the read queue. The same MAC process is performed, reusing the same ALU hardware, using the *hidden state* data of the previous timestep, if any. Once both stages are done, the results

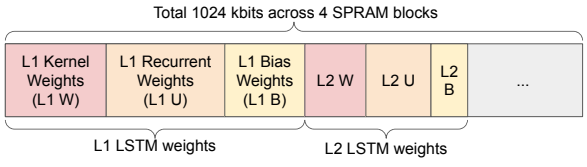

Fig. 6. LSTM weights are stored in layer-major sequence

of both are forwarded to the bias aggregation stage, where the two streams are added with the bias weights conveniently ready in the read pipeline. At the end of this stage, we have a $Unit$ number of four value tuples stored in a BRAM FIFO.

3) Carry and Hidden State Pipeline: Once the MAC operations are done and the $Unit$ number of 4-byte tuples are generated, we can calculate the carry states for the next time step, and the hidden states for the next layer according to the LSTM algorithm. This process also re-uses the same quantized ALU module to perform hard sigmoid, quantized multiplication and addition.

Another notable feature of the layer-major weight map and computation organization is that the memory for carry state and hidden state results can each be stored in a single large FIFO implemented in BRAM, instead of a complex memory map for each layer or unit. Because the computation is done sequentially in a layer-major order, the hidden state FIFO will always hold the values for the next layer. By the time all layer processing is done and computation comes back to the first layer for the next time step, the carry state for all of the layers will be waiting in-order in the carry state FIFO.

4) Quantized ALU Architecture: Our quantized ALU provides three functions: quantized addition, multiplication, and hard sigmoid. While addition between two quantized values is straightforward, multiplication is slightly more complicated. Because linear quantization involves a scale factor mapping one set of values to another, multiplying two scaled values results in multiplication of the scale factor as well, and the multiplication results need to be divided with the scale factor again to maintain correctness.

While division in hardware is resource-intensive, fortunately the scale factor is a static value determined during quantization. We pre-calculate a normalized division `(1≪16)/scale` during quantization. This value can simply be multiplied to the multiplication results using the UP5K DSP block, which supports 16-bit multiplication to a 32-bit value. The resulting value can be shifted down 16 bits to get the correct division results. Since the original multiplication results are 16 bits wide, and because weights and input are all 8 bits, we do not lose any correctness with this approach. As a result, quantized multiplication requires two DSP blocks, one for multiplication, and one for division by the scale factor.

The hard sigmoid module also requires a multiplier, but since our computation organization never requires multiplication and sigmoid calculation to happen simultaneously, our ALU is designed to share one internal multiplier.

Since the SPRAM memory interface is 16 bits wide, two weights are delivered per cycle. Since the four weights in a cell are processed independently from each other, our quantized ALU internally implements two separate modules to operate in parallel, consuming 4 of the 8 available DSP blocks. We note that although memory bandwidth could be easily increased by using SPRAM modules in parallel, our experiments showed that this would not lead to high performance as it was difficult to fit more ALU modules in the limited chip space.

D. Dense Core Architecture

The dense core is a separate entity from the LSTM core. Its architecture is similar to the architecture of the LSTM cell, but much simpler. The dense core only has three computation stages: kernel MAC, bias aggregation, and a final hard sigmoid. Since the amount of dense layer computation is nearly negligible compared to the LSTM layer, the quantized ALU of the dense block ALU is designed to use only one MAC unit, using only two DSP blocks and less supporting logic.

IV. Evaluation

The accelerators of Eciton were implemented using open-source toolchains Icestorm [50] and Bluespec [34], on a low-cost, low-power Lattice iCE40 UP5K FPGA, coupled with an Arduino Nano as the MCU [3].

The following sections present evaluation results of Eciton, including the neural network models we have trained and used for predictive maintenance datasets, as well as the performance and efficiency of our FPGA accelerator implementation.

A. Neural Network Models and Datasets

We demonstrate Eciton on two predictive maintenance scenarios: Turbofan engine maintenance dataset from NASA [38], as well as electrical motor maintenance using vibration and humidity [31]. Two predictive maintenance models were first trained using Tensorflow, quantized post-training, and finally loaded on the Eciton system. This section presents the original floating-point trained models, and provide more detailed analysis of quantization efficiency in Section IV-B.

1) Turbofan Engine Maintenance: The NASA Turbofan Engine Maintenance Model is a predictive maintenance model that simulates the failure of a turbofan engine, with the dataset being an engine degradation simulation using C-MAPSS. The dataset includes input from 25 different sensors. Our model uses two LSTM layers with 100 and 50 nodes, as well as a single-cell dense layer. The total number of weights adds up to 80,651. The model before quantization reached a competitive accuracy of 97%.

2) Electrical Motor Maintenance: The Electrical Motor Maintenance model is trained on four input data streams, 3 accelerometer axes and a humidity sensor, collected from an electrical motor in the process of spinning down. The accelerometers are sampled at 2 KHz, while the humidity sensor is sampled at 50 Hz. This scenario has interesting real-time and energy budget requirements because the system is powered by a power harvester generating energy from the magnetic field as the motor spins. Because power harvesting becomes unavailable after the motor has shut down, the collected data must be either processed or transmitted after detecting motor shutdown, within a budget of 0.6 Joules of harvested energy. The model uses three LSTM layers each with 128, 32, and 16 nodes, and a single-cell dense layer adding up to a total of 91,873 weight values. The model before quantization has a competitive trained accuracy of 90%.

B. Quantized Model Accuracy

Figure 7 shows the effect of various quantization approaches on accuracy on the turbofan model. While use of hard sigmoid, 16-bit and 8-bit fixed-point quantizations have gradual loss of accuracy, 4-bit quantization results in a sharp, 24% drop. While the 16-bit quantized model is sufficiently small to fit in the on-chip SPRAM, we decided on 8-bit quantization to make better use of the limited DSP blocks, performing more 8-bit operations per cycle. The electric motor model behaved similarly, and the 8-bit quantized model achieved an accuracy of 84%, a modest 6% accuracy drop.

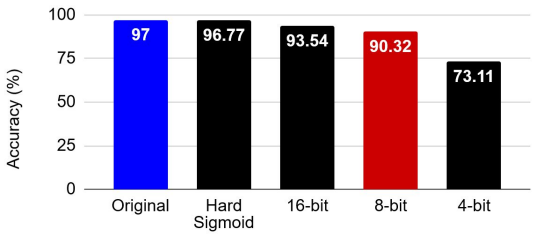

Fig. 7. Accuracy of the turbofan model drops sharply at 4-bit quantization

C. FPGA Resource Utilization

The resource utilization for Eciton is presented in Table I. All 4 available SPRAM blocks, as well as 6 of the 8 available DSP blocks are used. Overall, the accelerator uses 94.5% of the board's logic cells, while achieving a clock frequency of 17.0 MHz.

TABLE I
Eciton on-chip resource utilization

Resource	Resource Utilization	Percent Utilization (%)
LC	4987 / 5280	94.5
SPRAM	4 / 4	100
BRAM	22 / 30	73.3
DSP	6 / 8	75

The total on-chip capacity of the four SPRAM blocks is 256 KB, which can comfortably fit either of our trained models, and potentially much larger ones as well. The memory footprint of the turbofan model is 80,651 bytes, and the electrical motor model is 91,873 bytes.

D. Performance Evaluation

We evaluate the performance and power efficiency of Eciton against multiple system configurations, including Keras and our best-effort software implementation on an Intel core i7-8700K CPU, as well as a Raspberry Pi Zero, an Arduino Uno (ATmega) and an Arduino Due (ARM Cortex-M0). We have implemented both quantized and non-quantized models in software, and evaluated both on possible platforms.

The Raspberry Pi Zero was chosen as it regularly demonstrated good power-efficiency among similar Linux-enabled embedded systems offerings. We emphasize that the both

Arduino systems do not have enough on-chip SRAM to hold even the quantized model. To obtain an upper-limit estimate of performance on the Arduino boards, we measured the performance of the software modified to re-use a much smaller set of weights, resulting in extremely poor accuracy.

Figure 8 and Figure 9 presents the performance and power efficiency evaluations of the turbofan and motor models, respectively. We see that both keras and our hand-optimized software on the i7 vastly outperforms all embedded systems. But its high power consumption and low power efficiency make it a poor fit for the edge, and is presented here just as a point of reference. Compared with embedded systems such as the Raspberry Pi and the Arduinos, Eciton delivered over 2× the performance of even the fastest system. We also note that on the i7 and RPi, the floating point implementations outperform quantized ones due to the added overhead of quantized arithmetic operations, specificially the added normalization requirement after each operation. An exception is the motor model on the RPi, where the overhead of more computation was offset by the reduced size of the model fitting better on the small on-chip cache.

E. Power Efficiency Evaluation

For power efficiency, we measured the total power consumption during execution of the turbofan neural network on all systems *except the i7 system*. For the i7 system, instead of measuring total system power consumption, we only measured the difference in power consumption between idle and load states, added with the idle power consumption divided by the number of cores. This was an attempt to make a fair comparison in the favor of the i7 system, regarding economy of scale in the datacenter.

Eciton's FPGA accelerator measured a power consumption around 17 mW under full load. Coupled with the Arduino Nano emulating sensor data input via USB serial, the average power consumption of the Eciton system under load measured around 290 mW. The power efficiency of Eciton is an order of magnitude better than other high performance and low power options. The systems with the next-best power efficiencies were the i7 and the Raspberry Pi running floating point, but even they were orders of magnitude lower.

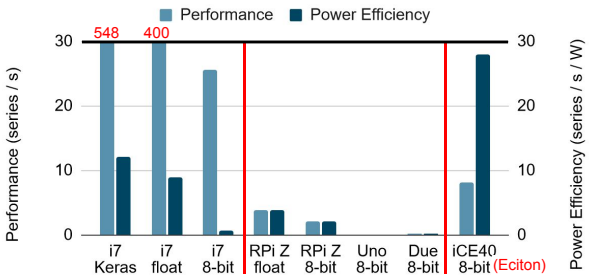

Fig. 8. Performance and power efficiency for turbofan

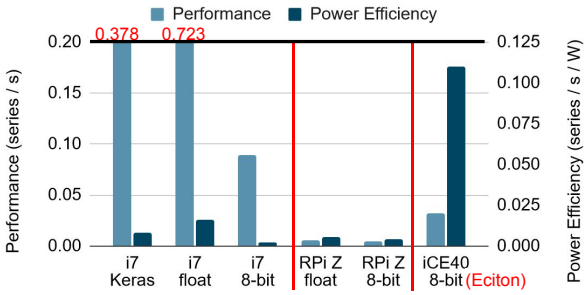

Fig. 9. Performance and power efficiency for motor

Eciton was also able to achieve the energy limit requirements of the electric motor scenario. It took Eciton 31.3 seconds to fully process the 25 seconds of 1/4 sampled data stream as per the model parameters, at a steady power consumption of 17 mW by the FPGA. Assuming the rest of the system can be put to sleep during this time, our Eciton's FPGA accelerator finishes the job within the 0.6 Joule energy limit imposed by the power harvester. No other system configuration was able to achieve this milestone.

F. Comparisons Against State-of-The Art

Table II shows performance and power efficiency comparisons against state-of-the-art FPGA implementations of LSTMs, aross various hardware platforms with different scale and capabilities. Despite the resource constraints of the UP5K platform, Eciton achieves competent performance and power efficiency compared to other low-power implementations such as those on Xilinx Artix 7 FPGAs. While Eciton achieves lower performance and power efficiency compared to much larger, power-hungry FPGAs that benefit from better power economies at scale, Eciton still occupies an important niche in the FPGA accelerator catalog due to its very lower power consumption as well as real-time processing capabilities. While some bigger chips can deliver better performance per watt, resource-constrained edge deployments may not be able to use them at all due to the power budget.

TABLE II
COMPARISONS AGAINST STATE-OF-THE-ART

	Low-power		High-power	
	Eciton	[26]	[6]	[8]
Platform	**iCE40**	Artix-7	Zynq-7000	Arria 10
mW	**17**	109	280	19,100
GOP/s	**0.067**	0.055	7.51	304.1
GOP/s/W	**3.9**	0.5	26.84	15.9

G. Total System Power With Networking

Figure 10 shows the sustained, under-load total system power consumption breakdown including network transmission, for the top two most power efficient embedded systems evaluated: The Raspberry Pi Zero and Eciton, as well as an Arduino Nano system transmitting all data collected with no edge computing. The Raspberry Pi and Eciton systems

are configured with a low-power LoRa module, while the Arduino Nano is configured with a faster NB-IoT module due to the higher data rate required without edge filtering. The power consumption numbers and performance of NB-IoT and LoRaWAN was taken from existing work [25], [9], [1].

We can see that the power consumption of Eciton is significantly lower than other systems despite the FPGA addition. Furthermore, the power consumption is low enough that it can be sustainably powered by many proposed non-intrusive power harvesting units [31], [27], [2], meaning it can continuously operate indefinitely by harvesting power from the ambient environment.

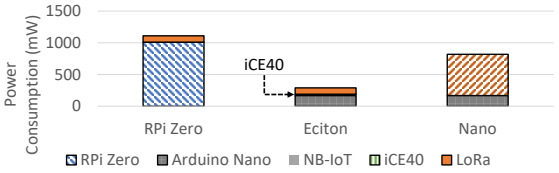

Fig. 10. Total system power consumption of nodes under load

V. Conclusion

We have presented the design and evaluation of Eciton, a neural network accelerator for real-time predictive maintenance at the edge. The reduced power and networking requirements enabled by Eciton will allow wider CPS/IoT deployments coupled with low-power wide-area networking and power harvesting technologies. We also plan to explore many other CPS/IoT domains where such a platform can be beneficial.

References

[1] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. Understanding the limits of lorawan. *IEEE Communications magazine*, 55(9):34–40, 2017.

[2] Afghan Syeda Adila, Almusawi Husam, and Géza Husi. Towards the self-powered internet of things (iot) by energy harvesting: Trends and technologies for green iot. In *2018 2nd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)*, pages 1–5. IEEE, 2018.

[3] Arduino. Arduino nano. https://store.arduino.cc/usa/arduino-nano.

[4] Olgun Aydin and Seren Guldamlasioglu. Using lstm networks to predict engine condition on large scale data processing framework. In *2017 4th International Conference on Electrical and Electronic Engineering (ICEEE)*, pages 281–285. IEEE, 2017.

[5] Michael Baddeley, Reza Nejabati, George Oikonomou, Mahesh Sooriyabandara, and Dimitra Simeonidou. Evolving sdn for low-power iot networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 71–79. IEEE, 2018.

[6] Erfan Bank-Tavakoli, Seyed Abolfazl Ghasemzadeh, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram. Polar: A pipelined/overlapped fpga-based lstm accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(3):838–842, 2019.

[7] Dario Bruneo and Fabrizio De Vita. On the use of lstm networks for predictive maintenance in smart industries. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 241–248. IEEE, 2019.

[8] Shijie Cao, Chen Zhang, Zhuliang Yao, Wencong Xiao, Lanshun Nie, Dechen Zhan, Yunxin Liu, Ming Wu, and Lintao Zhang. Efficient and effective sparse lstm on fpga with bank-balanced sparsity. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 63–72, 2019.

[9] Lluís Casals, Bernat Mir, Rafael Vidal, and Carles Gomez. Modeling the energy performance of lorawan. *Sensors*, 17(10):2364, 2017.

[10] Antonio De La Piedra, An Braeken, and Abdellah Touhafi. Sensor systems based on fpgas and their applications: A survey. *Sensors*, 12(9):12235–12264, 2012.

[11] Elena I Gaura, James Brusey, Michael Allen, Ross Wilkins, Dan Goldsmith, and Ramona Rednic. Edge mining the internet of things. *IEEE Sensors Journal*, 13(10):3816–3825, 2013.

[12] Tiago Gomes, Sandro Pinto, Adriano Tavares, and Jorge Cabral. Towards an fpga-based edge device for the internet of things. In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–4. IEEE, 2015.

[13] Sotirios K Goudos, Panagiotis I Dallas, Stella Chatziefthymiou, and Sofoklis Kyriazakos. A survey of iot key enabling and future technologies: 5g, mobile iot, sematic web and applications. *Wireless Personal Communications*, 97(2):1645–1675, 2017.

[14] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al. Ese: Efficient speech recognition engine with sparse lstm on fpga. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 75–84, 2017.

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[16] Cong Hao, Xiaofan Zhang, Yuhong Li, Sitao Huang, Jinjun Xiong, Kyle Rupnow, Wen-mei Hwu, and Deming Chen. Fpga/dnn co-design: An efficient design methodology for iot intelligence on the edge. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019.

[17] Hashem M Hashemian. State-of-the-art predictive maintenance techniques. *IEEE Transactions on Instrumentation and measurement*, 60(1):226–236, 2010.

[18] Soumil Heble, Ajay Kumar, KV V Durga Prasad, Soumya Samirana, Pachamuthu Rajalakshmi, and Uday B Desai. A low power iot network for smart agriculture. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 609–614. IEEE, 2018.

[19] Shuang Jiang, Dong He, Chenxi Yang, Chenren Xu, Guojie Luo, Yang Chen, Yunlu Liu, and Jiangwei Jiang. Accelerating mobile applications at the network edge with software-programmable fpgas. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 55–62. IEEE, 2018.

[20] Deokwoo Jung, Zhenjie Zhang, and Marianne Winslett. Vibration analysis for iot enabled predictive maintenance. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1271–1282. IEEE, 2017.

[21] Seongyoung Kang, Jinyeong Moon, and Sang-Woo Jun. Fpga-accelerated time series mining on low-power iot devices. In *2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 33–36. IEEE, 2020.

[22] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[23] P Latha and MA Bhagyaveni. Reconfigurable fpga based architecture for surveillance systems in wsn. In *2010 International Conference on Wireless Communication and Sensor Computing (ICWCSC)*, pages 1–6. IEEE, 2010.

[24] Lattice. ice40 lp/hx/lm. http://www.latticesemi.com/iCE40.

[25] Mads Lauridsen, Rasmus Krigslund, Marek Rohr, and Germán Madueno. An empirical nb-iot power consumption model for battery lifetime estimation. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2018.

[26] Nitheesh Kumar Manjunath, Hirenkumar Paneliya, Morteza Hosseini, W David Hairston, Tinoosh Mohsenin, et al. A low-power lstm processor for multi-channel brain eeg artifact detection. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pages 105–110. IEEE, 2020.

[27] Philipp Mayer, Michele Magno, and Luca Benini. Smart power unit-mw-to-nw power management and control for self-sustainable iot devices. *IEEE Transactions on Power Electronics*, 2020.

[28] Arnab Neelim Mazumder, Hasib-Al Rashid, and Tinoosh Mohsenin. An energy-efficient low power lstm processor for human activity monitoring. In *33rd International System-on-Chip Conference (SOCC)*. IEEE, 2020.

[29] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. Overview of cellular lpwan technologies for iot deployment: Sigfox, lorawan, and

nb-iot. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 197–202. IEEE, 2018.

[30] Microsemi. Igloo. https://www.microsemi.com/product-directory/fpgas/1689-igloo.

[31] Jinyeong Moon, Peter Lindahl, John Donnal, Steven Leeb, Lt Ryan Zachar, Lt William Cotta, and Christopher Schantz. A nonintrusive magnetically self-powered vibration sensor for automated condition monitoring of electromechanical machines. In *2016 IEEE AUTOTESTCON*, pages 1–7. IEEE, 2016.

[32] D Mourtzis, E Vlachou, and NJPC Milas. Industrial big data as a result of iot adoption in manufacturing. *Procedia cirp*, 55:290–295, 2016.

[33] Karan Nair, Janhavi Kulkarni, Mansi Warde, Zalak Dave, Vedashree Rawalgaonkar, Ganesh Gore, and Jonathan Joshi. Optimizing power consumption in iot based wireless sensor networks using bluetooth low energy. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 589–593. IEEE, 2015.

[34] Rishiyur Nikhil. Bluespec system verilog: efficient, correct rtl from high level specifications. In *Proceedings. Second ACM and IEEE International Conference on Formal Methods and Models for Co-Design, 2004. MEMOCODE'04.*, pages 69–70. IEEE, 2004.

[35] Keith E Nolan, Wael Guibene, and Mark Y Kelly. An evaluation of low power wide area network technologies for the internet of things. In *2016 international wireless communications and mobile computing conference (IWCMC)*, pages 439–444. IEEE, 2016.

[36] Vilabha S Patil, Yashwant B Mane, and Shraddha Deshpande. Fpga based power saving technique for sensor node in wireless sensor network (wsn). In *Computational Intelligence in Sensor Networks*, pages 385–404. Springer, 2019.

[37] Marwen Roukhami, Mihai Teodor Lazarescu, Francesco Gregoretti, Younes Lahbib, and Abdelkader Mami. Very low power neural network fpga accelerators for tag-less remote person identification using capacitive sensors. *IEEE Access*, 7:102217–102231, 2019.

[38] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE, 2008.

[39] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[40] Sule Selcuk. Predictive maintenance, its implementation and latest trends. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 231(9):1670–1679, 2017.

[41] Lattice Semiconductor. Lattice sensai stack. https://www.latticesemi.com/sensAI.

[42] Björn Stelte. Toward development of high secure sensor network nodes

[49] Daqian Wei, Bo Wang, Gang Lin, Dichen Liu, Zhaoyang Dong, Hesen Liu, and Yilu Liu. Research on unstructured text data mining and using an fpga-based architecture. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pages 539–543, 2010.

[43] Xiao Sun, Jungwook Choi, Chia-Yu Chen, Naigang Wang, Swagath Venkataramani, Vijayalakshmi Viji Srinivasan, Xiaodong Cui, Wei Zhang, and Kailash Gopalakrishnan. Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. *Advances in neural information processing systems*, 32:4900–4909, 2019.

[44] Sapna Tyagi, Amit Agarwal, and Piyush Maheshwari. A conceptual framework for iot-based healthcare system using cloud computing. In *2016 6th International Conference-Cloud System and Big Data Engineering (Confluence)*, pages 503–507. IEEE, 2016.

[45] Arijit Ukil, Soma Bandyoapdhyay, Chetanya Puri, and Arpan Pal. Iot healthcare analytics: The importance of anomaly detection. In *2016 IEEE 30th international conference on advanced information networking and applications (AINA)*, pages 994–997. IEEE, 2016.

[46] Juan Valverde, Andres Otero, Miguel Lopez, Jorge Portilla, Eduardo De la Torre, and Teresa Riesgo. Using sram based fpgas for power-aware high performance wireless sensor networks. *Sensors*, 12(3):2667–2692, 2012.

[47] Qi Wang, Siqi Bu, and Zhengyou He. Achieving predictive and proactive maintenance for high-speed railway power equipment with lstm-rnn. *IEEE Transactions on Industrial Informatics*, 16(10):6509–6517, 2020.

[48] Qianlong Wang, Yifan Guo, Lixing Yu, and Pan Li. Earthquake prediction based on spatio-temporal data mining: an lstm network approach. *IEEE Transactions on Emerging Topics in Computing*, 2017. fault classification based on rnn-lstm with malfunction inspection report. *Energies*, 10(3):406, 2017.

[50] Clifford Wolf and Mathias Lasser. Project icestorm. http://www.clifford.at/icestorm/.

[51] Yoji Yamato, Yoshifumi Fukumoto, and Hiroki Kumazaki. Predictive maintenance platform with sound stream analysis in edges. *Journal of Information processing*, 25:317–320, 2017.

[52] Yukuan Yang, Lei Deng, Shuang Wu, Tianyi Yan, Yuan Xie, and Guoqi Li. Training high-performance and large-scale deep neural networks with full 8-bit integers. *Neural Networks*, 125:70–82, 2020.

[53] Xiaofan Zhang, Anand Ramachandran, Chuanhao Zhuge, Di He, Wei Zuo, Zuofu Cheng, Kyle Rupnow, and Deming Chen. Machine learning on fpgas to face the iot revolution. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 894–901. IEEE, 2017.

[54] Chao Hu Zhiyong, Liu Yingzi Pan, Zhenxing Zeng, and Max Q-H Meng. A novel fpga-based wireless vision sensor node. In *2009 IEEE International Conference on Automation and Logistics*, pages 841–846. IEEE, 2009.