Myrmec: FPGA-Accelerated SmartNIC for Cost and Power Efficient IoT Sensor Networks

Jeffrey Chen and Sang-Woo Jun(✉)

University of California, Irvine, CA 92697, USA
jeffrc2@uci.edu, swjun@ics.uci.edu

Abstract. Battery-powered wireless sensor nodes are one of the fundamental components of IoT-style wide-scale data collection and processing, but their capabilities are often restricted by the limited wireless transmission bandwidth achievable under the stringent power envelope imposed by the battery or power harvesters. Extreme edge computing attempts to mitigate this issue by offloading some computation to the sensor nodes with the aim of reducing the wireless data transfer requirements, and it has shown great promise especially using application-specific hardware acceleration on reconfigurable fabrics such as FPGAs. However, simply attaching an FPGA accelerator as a peripheral to the embedded microcontroller requires microcontroller software to move data between the accelerator and network interface, which can quickly become the bottleneck for high-speed data collection and processing. In this work, we present Myrmec, a SmartNIC architecture which mitigates this burden by placing a low-power FPGA on the datapath between the microcontroller and NIC. We present a carefully optimized architecture for wireless data collection, and use three important application scenarios to show that it can improve effective bandwidth by up to almost 3× compared to a standalone accelerator, which is on top of the order of magnitude reduction in wireless data transfer thanks to extreme edge computing. Thanks to reduction of wireless data transfer, Myrmec can reduce the overall power consumption of the node, despite the addition of acceleration which significantly improves data collection performance.

Keywords: Wireless Sensor Network · FPGA · SmartNIC · IoT

1 Introduction

Driven by the popularity of the Internet-of-Things (IoT) or Cyber-Physical Systems (CPS) paradigms, we are expecting explosive sustained growth of ubiquitous data collection with wireless sensor nodes, as well as their processing for deep insight. The number of edge-enabled IoT devices is expected to grow to almost eight billion by the year 2030 [32], and the size of the IoT market is expected to grow to over $650 billion by the year 2023 [23]. Wireless sensing and

C. Silvano et al. (Eds.): SAMOS 2023, LNCS 14385, pp. 57–71, 2023.
https://doi.org/10.1007/978-3-031-46077-7_5

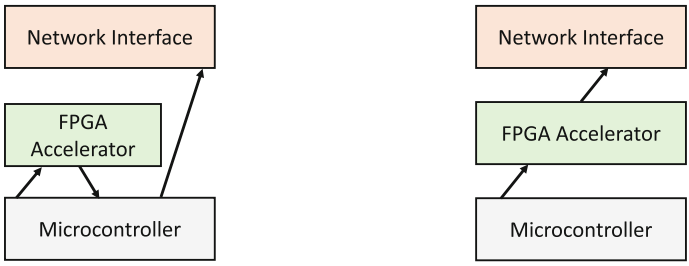
IoT technologies are being successfully deployed for a wide range of applications spanning personalized health, infrastructure monitoring, agriculture, security, and more [26,27,30].

One of the most prominent limitations to improving the scalability of such data collection is the limited computation and networking capabilities of the sensor node, imposed by its restricted power budget. Sensor node deployment often involve scenarios without access to a reliable power infrastructure, such as physical distribution across remote areas [1,10,15] or carried by people in unintrusive manner [2,17]. In such situations, the nodes are expected to operate for long amounts of time powered either by small batteries or by power harvesters. Such restricted power availability limits the rate of data collection because high-speed wireless data transmission is notoriously power-hungry, making it the primary performance bottleneck for battery-powered sensor nodes [4,6,20].

Edge processing, specifically *extreme edge* processing, is a prominent paradigm for reducing the network transmission requirements at the cost of more computation. Edge processing offloads some computation nearer to where data is collected, to distill data to smaller sizes before transmission. Extreme edge processing have shown to reduce wireless data transfer by over 95% percent [12]. This is done either at an intermediate *edge server*, on the path between the sensor nodes and the central server [31,35], or by placing more computation on the sensor devices themselves, on the so-called *extreme edge* [25]. Unfortunately, extreme-edge computing is not always useful in increasing data collection performance within the provided power budget. This is because while it can significantly reduce the data transmission requirements of the sensor node, it also increases its computation overhead, which comes with its own power budget requirements. On the other hand, edge processing while limiting computation within the original power budget may cause computation to become the bottleneck [8].

Application-specific hardware acceleration, especially using reconfigurable fabrics such as Field-Programmable Gate Arrays (FPGA), is a promising technology for low-power, low-cost edge processing, and there is great interest into how they should be integrated into the overall system. Hardware accelerators add an immense amount of computational capabilities to the system, allowing previously unavailable functionalities such as cryptography [9] or machine learning inference [8]. Furthermore, the sudden increase of computational throughput brought by FPGAs can often move the performance bottleneck to other system components, such as the communication between the microcontroller and FPGA [9]. For systems with sensor, accelerator, and network modules independently connected to the host microcontroller, the software overhead of moving data between these three components often became the prominent bottleneck instead of even the network bandwidth, since the FPGA accelerator can efficiency reduce the network bandwidth requirements.

In this paper, we address the performance issue of data movement for extreme edge acceleration with SmartNICs, where an FPGA accelerator is located on the datapath between the host microcontroller and the network module. We use the term *SmartNIC* (Smart Network Interface Card), even though our network is



(a) FPGA accelerator as a separate module has higher latency and higher bandwidth requirements.

(b) FPGA accelerator as a SmartNIC has lower latency and lower bandwidth requirements.

Fig. 1. The SmartNIC architecture results in much fewer data movement as well as lower bus bandwidth requirement.

not in a card format, since the term is commonly used in the datacenter context to describe a very similar architecture. Figure 1 illustrates the difference in data movement requirements between a conventional, independently connected accelerator and a SmartNIC-style accelerator. For the SmartNIC system, the host software is only responsible for transferring the sensor data to the SmartNIC accelerator, unlike the conventional system where it needs three separate data movement operations until it can be transmitted over the network.

We present the design and evaluation of our SmartNIC architecture, Myrmec, which is optimized for using very low-cost, low-power FPGAs in the extreme edge. To make the best use of FPGA resources with useful application-specific work, Myrmec's design does not use precious FPGA resource to handle network protocols, instead maintaining most of protocol handling in the software libraries just like non-accelerated systems. Instead, the accelerator interface transmits data by injecting accelerated output into the output stream generated by the software libraries. This way, the vast majority of FPGA resources can be allocated to application-specific acceleration.

To demonstrate that this architectural approach can effectively accelerate important applications, we construct a physical prototype to evaluate three prominent applications for edge processing. Our Myrmec prototype uses a low-cost ($< \$ 5$), low-power (¡ 20mW) Lattice iCE40 UP5K FPGA integrated into an embedded system consisting of an Arduino microcontroller and low-power LoRA communication, connected over SPI busses. We used three important applications for edge processing: Spike detection from time series information [7], Differential privacy [34], and Time series distance calculation using Dynamic Time Warping [33]. Using this prototype, we demonstrate two benefits of our system: First, the SmartNIC configuration can improve effective system throughput by almost $3\times$ for some applications compared to an independently installed accelerators. Second, aided by the effective resource usage of the Myrmec architecture, the low-power UP5K FPGA can effectively accelerate the important, select

benchmark applications, showing that this is a reasonable resource to allocate for real-world scenarios.

The rest of this paper is organized as follows: We present some relevant background and existing research in Sect. 2. Then, we present the design of the Myrmec architecture as well as the specifications of our prototype implementation in Sect. 3. We then present our evaluation results with important edge acceleration applications in Sect. 4. We conclude with discussions in Sect. 5.

2 Background and Related Works

Edge processing in the extreme edge, especially using power-efficient FPGA accelerators, is a popular paradigm for reducing the power-hungry wireless network transmission. Extreme-edge FPGAs have been used of offload many applications including video processing [19,36], posture and fall detection [22], time series mining [18], neural networks [3,8] and more. Many such accelerators result in over 95% reduction in wireless transmission, resulting in proportional power reductions [12].

In the datacenter, FPGA accelerators are often installed as a SmartNIC, where the FPGA is integrated into the network module itself [11,21]. This approach is gaining popularity since it can enable zero-latency acceleration in a bump-in-the-wire fashion as packets move over the network. It also supports low-latency access to remote accelerators over the network, enabling efficient distributed accelerator development [28].

Such accelerators, as well as conventional sensor nodes and wireless networking modules, are typically connected over a simple system bus such as I2C and SPI [13,16]. These buses are low power, and also inevitably low performance. But they are still the popular choice for even relatively high-performance controllers such as the Raspberry Pi [5]. As a result, there is significant research into faster, low-power system bus technologies as well [29].

A wide selection of wireless communication technologies exist for our sensor nodes [14,24]. There is typically a trade-off between bandwidth and power consumption, spanning from low-power, low-bandwidth WAN technologies like LoRa to faster ones with significantly higher power consumption such as MB-IoT. Slower LoRa can support tens of kbps of performance at tens of mW of power, compared to faster MB-IoT and LTE-M technologies which can support an order of magnitude higher bandwidth but also suffers proportionally larger power consumption.

3 SmartNIC Architecture for Embedded Systems

Myrmec places an FPGA accelerator on the data path between the microcontroller and the network interface, in order to remove the microcontroller overhead of having to transmit data to and from the accelerator. Figure 2 describes this configuration using a photo and architectural illustration of our prototype. Once the microcontroller transmits the collected sensor data to the SmartNIC,

consisting of the FPGA and the network interface, further data movement to the network interface (e.g. LoRA) is taken care of by the FPGA accelerator via a separate interconnect directly between the FPGA and the network. Neither microcontroller cycles nor its interconnect bandwidth needs to be spent on transmitting the accelerator-processed data to the network.

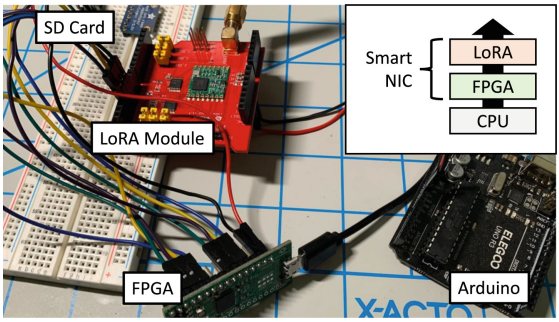


Fig. 2. A Myrmec prototype augmented with an FPGA on the wireless transmission datapath.

We note that our design is optimized for wireless sensor data collection, where the vast majority of data communication happens in a single direction, from the sensor nodes to the central host. We take advantage of this knowledge to optimize the hardware exclusively for data transmission.

3.1 Myrmec Accelerator Architecture

Figure 3 illustrates the internal architecture of the Myrmec system, and how the accelerator fits into the network datapath. Our Myrmec prototype facilitates communication between the microcontroller, network, and the FPGA over SPI links, but in principle other communication fabric such as I2C or UART can be used as well.

Like existing datacenter-scale SmartNIC designs, Myrmec also provides a *shell* on which user accelerators are programmed. By implementing user accelerators on top of the shell, Myrmec is able to provide a consistent interface to the host software. The interface between the shell and the user accelerator is carefully designed to minimize communication overhead, as well as minimize FPGA resource utilization in the shell. All input to the shell comes in through a single SPI link, and this stream is address-mapped to either a register map of parameters, or one of three queues: The data stream queue, command queue, and the bypass queue. The bypass queue is the normal path for software to communicate with the network interface. The software can also use the command and stream queues in concert with the parameter map to invoke accelerator kernels, and inject accelerator output into the bypass queue as payload.

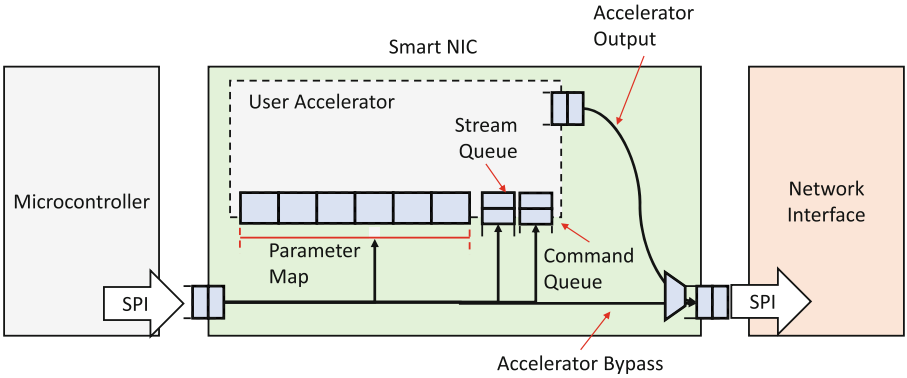


Fig. 3. Myrmec accelerator architecture exposes a common interface to the host software across all user accelerators.

- **Parameter map:** The lowest few addresses of the memory map points to the parameter register map. These registers can be random-accessed by the user accelerator to receive execution parameters from the host software, such as length of the incoming stream, window size, scale, and others. Because the parameter map is a randomly accessible register file, its size needs to be kept small to minimize the chance of timing closure issues in the user logic. The prototype shell reserves eight registers for the parameter map.
- **Command queue:** The command queue is used to initiate the user accelerator operations, as well as send commands to the other components of the shell. For example, the command queue is used to program the burst size parameters for the *Stream queue* described below, as well as program the MUX to interleave data from the accelerator output and the bypass queues for the single output queue to the network interface.
- **Stream queue:** Since all communication is now address-mapped, every byte of communication now incurs an additional byte of address overhead. To remove this overhead in typical cases, Myrmec also provides a DMA-like burst transfer over the stream queue. Once a stream burst is initiated via the command queue, the requested number of subsequent bits are sent to the stream queue without any additional address parameters. More details about the burst process is described in Sect. 3.2.
- **Bypass queue:** Input data is sent directly to the network interface without going through the accelerator. This stream can be merged and interleaved with the output stream from the accelerator according to the host software control. More details about the interleaving process is described in Sect. 3.2.

3.2 Software Interface

As described in Sect. 3.1, Myrmec provides an address-mapped interface into the SmartNIC shell, which enables different user accelerators to be invoked using a consistent software interface.

Figure 4 shows the stream of software commands involved in a very simple filtering accelerator. We note that while the input stream is illustrated as two separate streams for better visibility, they both arrive over the same SPI interface in sequence.

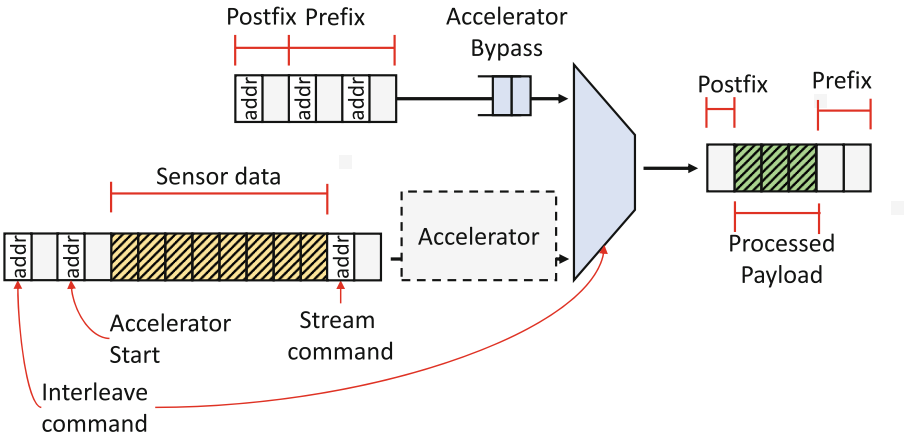


Fig. 4. Communication involved in a simple filtering accelerator call.

Instead of implementing the network controllers directly within the FPGA accelerator, Myrmec uses the accelerator bypass queue to use existing software libraries to control the network module. This approach bypasses the complexities of the network controller libraries in hardware design, as well as take advantage of the existing software libraries. Only the payload (if any) is emitted by the accelerator, which is then merged with the bypassed stream to construct the complete stream of control and data for the network device. In Fig. 4, such control data is depicted as prefix and postfix streams, which are sent to the accelerator bypass queue using its memory-mapped address.

The computational heavy lifting by the accelerator is initiated by a combination of commands over the command queue, as well as the burst of sensor data over the stream queue. First, a stream command is sent to the command queue to initiate a burst into the stream queue. Once the stream is ingested, the software can send the accelerator initiation command over the command queue, and then specify the interleaving pattern via a number of interleave commands. While Fig. 4 shows only one interleave command, in reality we need as many commands as there are bursts to merge. For the example in the figure, we would need three interleave commands for the prefix, payload, and postfix.

This simple example omits the use of parameters via the parameter map, but its addition should be a straightforward addition of more address-data pairs directed to the parameter map.

4 Evaluation

We construct a prototype Myrmec system using an Arduino Uno equipped with an ATMega328 microcontroller, a low-cost, low-power Lattice iCE40 UP5K FPGA, as well as a LoRA network module. All of these components were selected for their low cost and low power, meaning inevitably they are also low capability among their categories. However, we demonstrate using important real-world applications that even these devices are sufficient to maintain high rates of data collection.

We evaluate five different system configurations:

- **RPiZ**: Raspberry Pi Zero, one of the more power-efficient offerings of the Raspberry Pi single-board computers, equipped with a 1 GHz ARMv6 CPU.
- **Teensy4**: Arduino Teensy 4, one of the more powerful Arduino embedded systems, equipped with a 600 MHz ARM Cortex-M7.
- **Mega**: An arduino Mega, equipped with an embedded ATmega2560 microcontroller running at 16 MHz.
- **Ours**: Our Myrmec equipped with a low-performance ATmega328P running at 16 MHz, as well as a low-power Lattice UP5K FPGA. The FPGA can either be independent from the network, or integrated into a SmartNIC according to the Myrmec design.

4.1 Application Details

We evaluate three applications on the Myrmec prototype:

- **Time series spike detection:** This application detects when a spike exists in the time series data from a sensor. A spike is detected by calculating a running average and variance over a window of input data, and raising a flag whenever a data elements is larger than the average by a certain multiple of the variance [7].
  The user accelerator has a hard-coded window size, and the current window is stored in a BRAM FIFO. A running sum is maintained by adding every new input, and subtracting the old input being evicted from the window BRAM FIFO. This value is used to calculate the running average, which is fed into the variance calculation module which uses a similar strategy to calculate the variance.
  The accelerator consumes 21% of the UP5K LUT resources.
- **Differential privacy:** This application injects randomness into each sensor data in a principled fashion, in order to maintain statistical accuracy over many data samples, but hides personally identifiable information from the data stream by providing plausible deniability from randomness [34].
  The user accelerator implements a simple per-element Laplace mechanism, where a random noise is generated by subtracting two samples from the Laplace distribution. The user accelerator implements a linear congruential pseudo-random number generator seeded via a user-input parameter, as well as the integer logarithm modules required for the Laplace distribution.
  The accelerator consumes 23% of the UP5K LUT resources.

– **Time series distance calculation:** This application compares a window of input data against an application-provided reference series. For accurate comparisons we use the widely popular Dynamic Time Warping as the distance metric [33].
We only focus on calculating the distance value, without keeping track of point-by-point backtracking information. This simplifies the calculation for both software and hardware implementations since the whole dynamic programming matrix does not need to be store in memory. Instead, only two rows of the dynamic programming matrix at a time need to be maintained to calculate the next row based on the previous one.
The accelerator implements multiple Processing Elements (PE), and consumes 94% of the UP5K LUT resources.

Table 1 describes the changes in the data transmission for the three applications, as a result of edge acceleration. Spike detection and distance calculation both have a significant benefit in terms of data transmission reduction, since we no longer have to transmit the original collected data. On the other hand, differential privacy has no effect on the data transmission rate, because the amount of differentially privatized data is of same size compared to the original. While this means the differential privacy acceleration does not reduce network transmission, it is still an important application for the extreme edge since it can insure privacy before transmitting data from the sensor node into the untrusted network.

Table 1. Data filtering characteristics of the evaluated applications.

Application	Spike Detection	Differential Privacy	Distance Calculation
Data rate	↓	=	↓

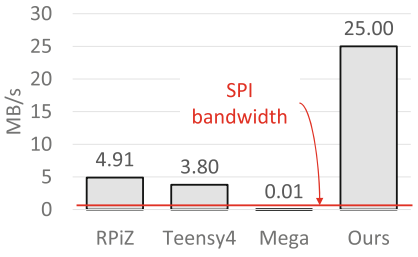
The three applications have differing data filtering characteristics, as well as widely varying computational requirements. For example, the spike detection algorithm has very low computational requirements, needing only a handful of arithmetic operations per input element, and drastically filtering the data for transmission. On the other hand, dynamic time warping has relatively compex computational requirements, needing $O(N)$ computation for each new sample, and also does not filter the data at all. Based on our evaluations on such varying applications, it should be possible to realistically assess the potential benefits of Myrmec on other applications of interest.

4.2 Performance Evaluation

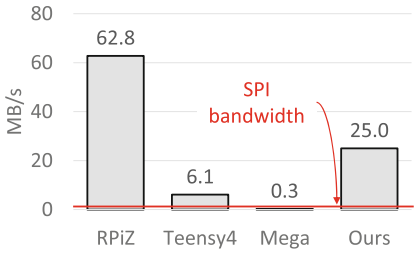
Figure 5 shows the performance evaluations of Myrmec using our prototype.

Streaming Accelerators: Figure 5a and 5b show the performance comparisons for the streaming algorithms, Differential privacy and Spike detection. Thanks to efficient pipelining, our accelerator is capable of maintaining a consistent throughput of 1 byte per cycle, resulting in a constantly high 25 MB/s bandwidth on a 25 MHz clock on both applications. The performance benefits of such effective pipelining is most prominent with the differential privacy results in Fig. 5a. The complex computation involved in the differential privacy application, including random number generation and the laplace distribution sampling, can be done in a completely pipelined manner. This is in contrast to software systems, which must expend multiple instructions to process each sample.

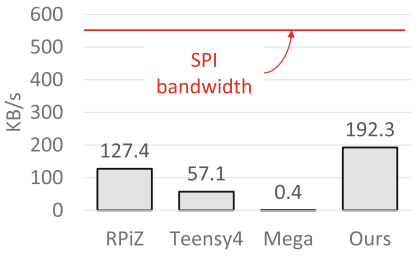
As a result, Myrmec is much faster than all other systems, except for Spike detection on the RPiZ. In this particular example, the computation requirements per sample byte is so low that even with the multiple cycles per byte of processing required by software, the high clock speed (1 GHz) of RPi Zero is able to achieve higher bandwidth than the fully pipelined FPGA running at 25 MHz. A single SPI link on our system is able to sustain 500 to 600 KB/s. Since all of the fastest systems for each application are much faster than what the SPI connection supports, both applications are bound by SPI performance, and communication bandwidth becomes valuable for performance.



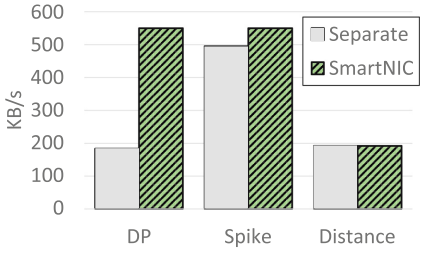
(a) Differential privacy performance.



(b) Spike detection performance.



(c) DTW performance.



(d) Performance benefits of SmartNIC architecture.

Fig. 5. Comparative performance evaluations.

This is emphasized in Fig. 5d, which compares the performance of Myrmec, the SmartNIC system, and a conventional node with an independently attached accelerator. Since there is no filtering for the differential privacy application, the three data movement operations (to FPGA, from FPGA, and to network) must all share a single SPI link as well as host software cycles. This results in an almost 3× gap in performance between the independent accelerator and Myrmec. On the other hand, Spike detection has significant amount of data filtering by the accelerator, so the overhead of the two latter data movement operations are minimal. As a result, the performance gap is much smaller compared to the differential privacy application.

Dynamic Time Warping Accelerator: Figure 5c shows the performance of the Dynamic Time Warping implementations. While Myrmec shows the best performance thanks to its multiple PEs, all measured systems are actually slower than the SPI, since the algorithm requires multiple passes over the whole data. Furthermore, since all systems are slower than the SPI bandwidth, bandwidth-saving SmartNIC approach has no benefits over the conventional systems. As a result, Fig. 5d shows that the Neighbor application (DTW) does not become faster on the SmartNIC device.

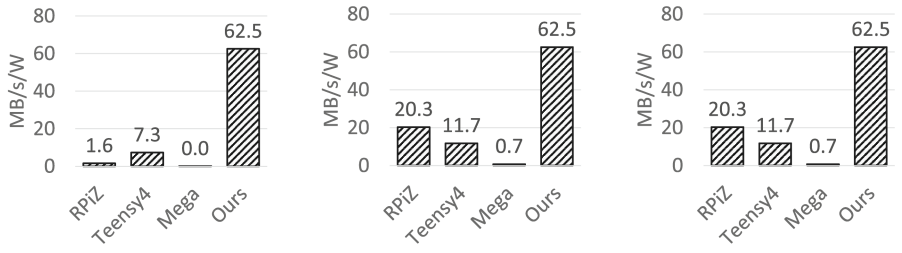
4.3 Power-Efficiency Evaluation

Figure 6 compares the power efficiency of different processing units on the three applications. Power consumption was measured via USB power monitors for each systems, and power efficiency was measured by dividing the bandwidth by the measured power. Note that the performance numbers we used were those not limited by SPI bandwidth, to illustrate the processing power efficiency of the computation units themselves.

Even when the raw performance was lower with Myrmec compared to more powerful systems like RPiZ, these figures show that thanks to the low power consumption of FPGAs, the power efficiency of Myrmec is much higher. The same is true for a non-SmartNIC FPGA-accelerated sensor node.

Since extreme edge acceleration can drastically reduce wireless transmission requirements, leading to reduction in network power consumption, the power consumption of the overall sensor node actually decreases despite the addition of an FPGA accelerator. In Fig. 7, we present the power consumption breakdown of the Spike detection application, which demonstrates end-to-end lower power consumption despite the addition of an FPGA accelerator. Since the maximum bandwidth of the LoRa module in our prototype is lower than the SPI bandwidth, we assume LoRa cannot sustain the necessary bandwidth without edge filtering. To reflect this, we use the published power consumption numbers for a higher-performance NB-IoT network interface for the conventional system. Other, even faster network fabrics exist (e.g., 5G, WiFi), but those also have proportionally larger power consumption.

We also note that this benefit is only available when edge processing has a filtering benefit. For non-filtering applications like differential privacy, the node



(a) Differential privacy power efficiency. (b) Spike detection power efficiency. (c) Dynamic time warping power efficiency.

Fig. 6. Power efficiency comparisons.

will consume more power compared to the original system, by the amount of power the FPGA consumes. However, this will likely be still valuable if differential privacy is required for the target deployment, since the SmartNIC-based system has much higher performance and power-efficiency compared to software implementations.

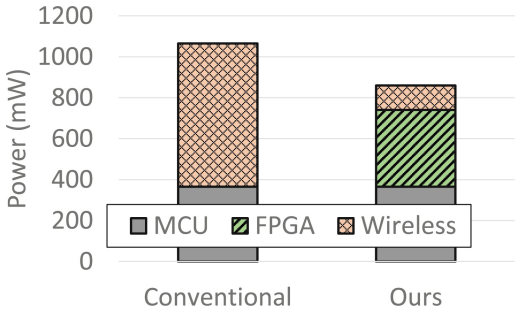


Fig. 7. Edge acceleration results in net reduction of power consumption.

5 Conclusion and Discussion

In this paper, we presented our work on Myrmec, which uses a SmartNIC-like FPGA-accelerated network interface to perform extreme edge acceleration without further stressing the system bus. Myrmec employed an interesting design choice of having the software still manage the network transmission protocol handling via widely available software libraries. This allowed Myrmec to minimize the FPGA resource overhead of networking, instead investing the vast majority of the precious reconfigurable logic for actually useful algorithm execution. We show that our prototype device with a low-power Lattice UP5K FPGA is actually capable of effectively offloading a select list of important benchmark

applications. We also show that the SmartNIC architecture can achieve higher bandwidth compared to the conventional, independently installed accelerator since the SPI system bus bandwidth, as well as the host microcontroller cycles, do not need to be shared for data movement between the accelerator and the network. Similarly to datacenter-scale SmartNICs, Myrmec is an effective tool for improving the performance and scalability of wide-scale data collection via wireless sensor nodes.

References

1. Ahmed, N., De, D., Hussain, I.: Internet of things (IoT) for smart precision agriculture and farming in rural areas. IEEE Internet Things J. **5**(6), 4890–4899 (2018)
2. Alam, M.M., Malik, H., Khan, M.I., Pardy, T., Kuusik, A., Le Moullec, Y.: A survey on the roles of communication technologies in IoT-based personalized healthcare applications. IEEE Access **6**, 36611–36631 (2018)
3. Anand, S., RK, K.M.: FPGA implementation of artificial neural network for forest fire detection in wireless sensor network. In: 2017 2nd International Conference on Computing and Communications Technologies (ICCCT), pp. 265–270. IEEE (2017)
4. Baddeley, M., Nejabati, R., Oikonomou, G., Sooriyabandara, M., Simeonidou, D.: Evolving SDN for low-power IoT networks. In: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pp. 71–79. IEEE (2018)
5. Calvo, I., Gil-García, J.M., Recio, I., López, A., Quesada, J.: Building IoT applications with raspberry pi and low power IQRF communication modules. Electronics **5**(3), 54 (2016)
6. Casals, L., Mir, B., Vidal, R., Gomez, C.: Modeling the energy performance of lorawan. Sensors **17**(10), 2364 (2017)
7. Chan, P.K., Mahoney, M.V.: Modeling multiple time series for anomaly detection. In: Fifth IEEE International Conference on Data Mining (ICDM 2005), pp. 8. IEEE (2005)
8. Chen, J., Hong, S., He, W., Moon, J., Jun, S.W.: Eciton: Very low-power LSTM neural network accelerator for predictive maintenance at the edge. In: 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), pp. 1–8. IEEE (2021)
9. Elnawawy, M., Farhan, A., Al Nabulsi, A., Al-Ali, A.R., Sagahyroon, A.: Role of FPGA in internet of things applications. In: 2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), pp. 1–6. IEEE (2019)
10. Farooq, M.S., Riaz, S., Abid, A., Umer, T., Zikria, Y.B.: Role of IoT technology in agriculture: a systematic literature review. Electronics **9**(2), 319 (2020)
11. Firestone, D., et al.: Azure accelerated networking: Smartnics in the public cloud. In: 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18), pp. 51–66 (2018)
12. Gaura, E.I., Brusey, J., Allen, M., Wilkins, R., Goldsmith, D., Rednic, R.: Edge mining the internet of things. IEEE Sens. J. **13**(10), 3816–3825 (2013)
13. Gia, T.N., et al.: IoT-based fall detection system with energy efficient sensor nodes. In: 2016 IEEE Nordic Circuits and Systems Conference (NORCAS), pp. 1–6. IEEE (2016)

14. Goudos, S.K., Dallas, P.I., Chatziefthymiou, S., Kyriazakos, S.: A survey of IoT key enabling and future technologies: 5G, mobile IoT, sematic web and applications. Wireless Pers. Commun. **97**, 1645–1675 (2017)
15. Heble, S., Kumar, A., Prasad, K.V.D., Samirana, S., Rajalakshmi, P., Desai, U.B.: A low power IoT network for smart agriculture. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), pp. 609–614. IEEE (2018)
16. Jafarzadeh, M., Brooks, S., Yu, S., Prabhakaran, B., Tadesse, Y.: A wearable sensor vest for social humanoid robots with GPGPU, IoT, and modular software architecture. Robot. Auton. Syst. **139**, 103536 (2021)
17. Kang, J.J., Yang, W., Dermody, G., Ghasemian, M., Adibi, S., Haskell-Dowland, P.: No soldiers left behind: an IoT-based low-power military mobile health system design. IEEE Access **8**, 201498–201515 (2020)
18. Kang, S., Moon, J., Jun, S.W.: FPGA-accelerated time series mining on low-power IoT devices. In: 2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 33–36. IEEE (2020)
19. Latha, P., Bhagyaveni, M.: Reconfigurable FPGA based architecture for surveillance systems in WSN. In: 2010 International Conference on Wireless Communication and Sensor Computing (ICWCSC), pp. 1–6. IEEE (2010)
20. Lauridsen, M., Krigslund, R., Rohr, M., Madueno, G.: An empirical NB-IoT power consumption model for battery lifetime estimation. In: 2018 IEEE 87th Vehicular Technology Conference (VTC Spring), pp. 1–5. IEEE (2018)
21. Li, J., Sun, Z., Yan, J., Yang, X., Jiang, Y., Quan, W.: DrawerPipe: a reconfigurable pipeline for network processing on FPGA-based SmartNIC. Electronics **9**(1), 59 (2019)
22. Mahdi, S.Q., Gharghan, S.K., Hasan, M.A.: FPGA-based neural network for accurate distance estimation of elderly falls using WSN in an indoor environment. Measurement **167**, 108276 (2021)
23. Marketsandmarksets: Internet of Things (IoT) Market Size, Global Growth Drivers amp; Opportunities. https://www.marketsandmarkets.com/Market-Reports/internet-of-things-market-573.html (2022). Accessed 30 Mar 2023
24. Mekki, K., Bajic, E., Chaxel, F., Meyer, F.: 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), pp. 197–202. IEEE (2018)
25. Merino, P., Mujica, G., Señor, J., Portilla, J.: A modular IoT hardware platform for distributed and secured extreme edge computing. Electronics **9**(3), 538 (2020)
26. Modieginyane, K.M., Letswamotse, B.B., Malekian, R., Abu-Mahfouz, A.M.: Software defined wireless sensor networks application opportunities for efficient network management: a survey. Comput. Electr. Eng. **66**, 274–287 (2018)
27. Mohamed, R.E., Saleh, A.I., Abdelrazzak, M., Samra, A.S.: Survey on wireless sensor network applications and energy efficient routing protocols. Wireless Pers. Commun. **101**, 1019–1055 (2018)
28. Ovtcharov, K., Ruwase, O., Kim, J.Y., Fowers, J., Strauss, K., Chung, E.S.: Accelerating deep convolutional neural networks using specialized hardware. Microsoft Res. Whitepaper **2**(11), 1–4 (2015)
29. Pahlevi, R.R., Abdurohman, M., et al.: Fast UART and SPI protocol for scalable IoT platform. In: 2018 6th International Conference on Information and Communication Technology (ICoICT), pp. 239–244. IEEE (2018)
30. Rashid, B., Rehmani, M.H.: Applications of wireless sensor networks for urban areas: a survey. J. Netw. Comput. Appl. **60**, 192–219 (2016)

31. Ray, P.P., Dash, D., De, D.: Edge computing for internet of things: a survey, e-healthcare case study and future direction. J. Netw. Comput. Appl. **140**, 1–22 (2019)
32. Statista: Number of edge enabled internet of things (IoT) devices worldwide from 2020 to 2030, by market. https://www.statista.com/statistics/1259878/edge-enabled-iot-device-market-worldwide/ (2021). Accessed 30 Mar 2023
33. Varatharajan, R., Manogaran, G., Priyan, M.K., Sundarasekar, R.: Wearable sensor devices for early detection of Alzheimer disease using dynamic time warping algorithm. Clust. Comput. **21**, 681–690 (2018)
34. Xu, C., Ren, J., Zhang, D., Zhang, Y.: Distilling at the edge: a local differential privacy obfuscation framework for IoT data analytics. IEEE Commun. Mag. **56**(8), 20–25 (2018)
35. Yu, W., et al.: A survey on the edge computing for the internet of things. IEEE Access **6**, 6900–6919 (2017)
36. Zhiyong, C.H., Pan, L.Y., Zeng, Z., Meng, M.Q.H.: A novel FPGA-based wireless vision sensor node. In: 2009 IEEE International Conference on Automation and Logistics, pp. 841–846. IEEE (2009)