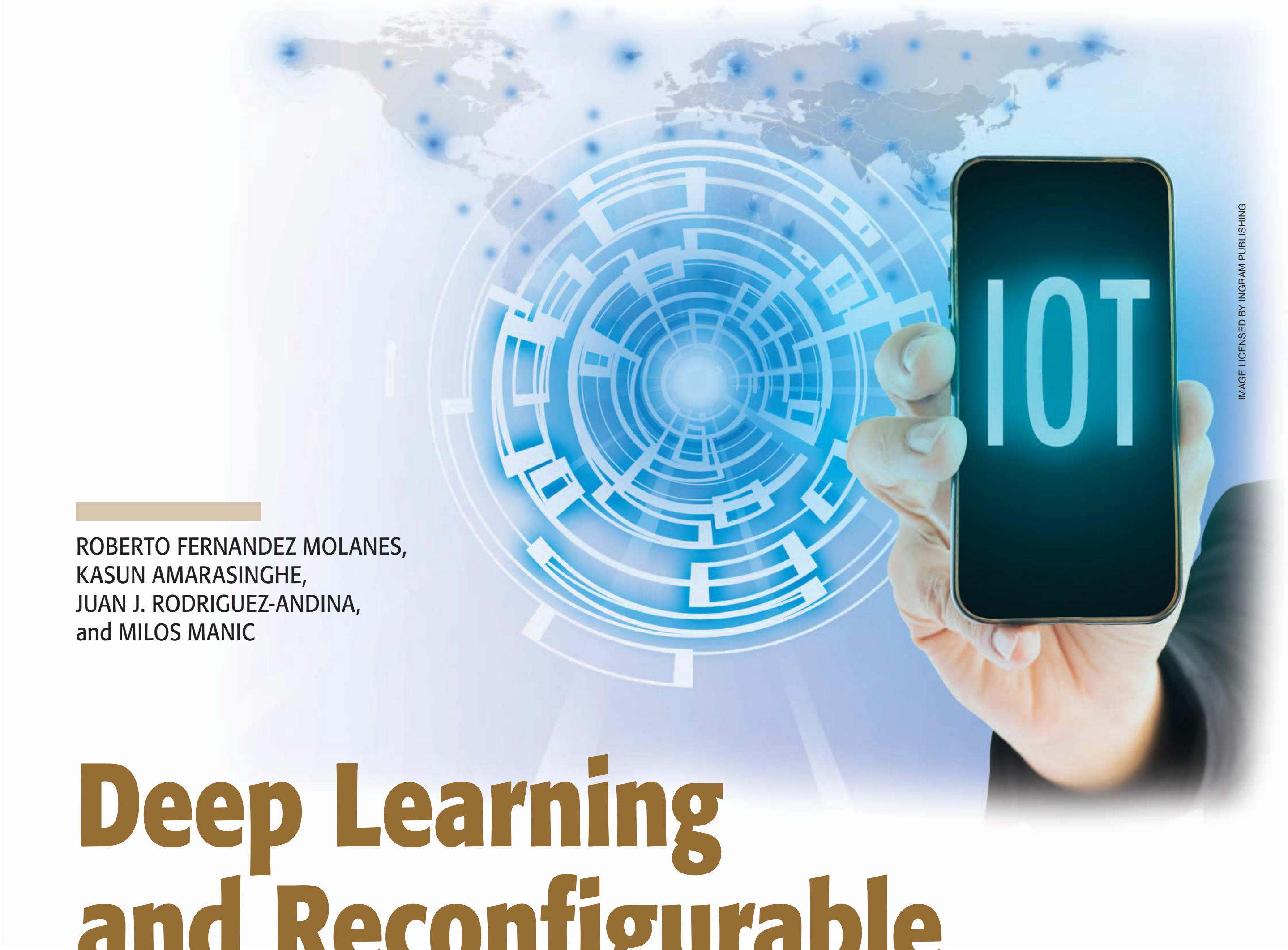


ROBERTO FERNANDEZ MOLANES, KASUN AMARASINGHE, JUAN J. RODRIGUEZ-ANDINA, and MILOS MANIC

Deep Learning and Reconfigurable Platforms in the Internet of Things

Challenges and Opportunities in Algorithms and Hardware

As the Internet of Things (IoT) continues its run as one of the most popular technology buzzwords of today, the discussion really turns from how the massive data sets are collected to how value can be derived from them, i.e., how to extract knowledge out of such (big) data. IoT devices are used in an ever-growing number of application domains (see Figure 1), ranging from sports gadgets (e.g., Fitbits and Apple Watches) or more serious medical devices (e.g., pacemakers and biochips) to smart homes, cities, and self-driving cars, to predictive maintenance in mission-critical systems (e.g., in nuclear power plants or airplanes). Such applications introduce endless possibilities for better understanding, learning, and informedly acting (i.e., *situational awareness* and *actionable information* in government lingo). Although rapid expansion of devices and sensors brings terrific opportunities for taking advantage of terabytes of machine data, the mind-boggling task of understanding growth of data remains

Digital Object Identifier 10.1109/MIE.2018.2824843
Date of publication: 25 June 2018

1932-4529/18©2018IEEE

and heavily relies on artificial intelligence and machine learning [1], [2].

Where traditional approaches do not scale well, artificial intelligence techniques have evidenced great success in applications of machine and cognitive intelligence (such as image classification, face recognition, or language translation). We recognize the widespread usage of various well-known machine-learning algorithms in the IoT (such as fuzzy systems, support vector machines, Bayesian networks, reinforcement learning, and others), but we focus here on the most recent and highly advantageous type of machine learning in the IoT: deep learning.

The success of deep learning and, in particular, deep neural networks greatly coincides with the advent of highly specialized, powerful parallel-computing devices, i.e., graphics processing units (GPUs) [4]. Although the overwhelming processing and memory requirements can be met with high-performance computing hardware, the resulting sheer size, cost, and power consumption would make the goal of deep neural network-enabled IoT and embedded devices unattainable.

In this scenario, field-programmable system-on-chip (FPSoC) platforms, which combine in a single chip one or more powerful processors and reconfigurable logic [in the form of field-programmable gate array (FPGA) fabric], are emerging as a very suitable implementation alternative for the next generation of IoT devices. The fine-grained structure of FPGAs has proven to provide powerful implementations of machine-learning algorithms with less power consumption than comparable platforms (in terms of cost or size) [5], making them ideal for machine and cognitive intelligence in strict resource-limited applications, like many in the IoT (while GPUs remain as the dominant platforms for other IoT scenarios).

Moreover, FPSoCs allow the processing load to be balanced between processors and reconfigurable logic, the most suitable implementation (hardware or software) being used for each specific functional building block to be optimized, and functionality to be easily reconfigured on site. In addition, reconfigurable platforms dramatically ease system scalability and upgrading. Hence, they provide high levels of flexibility, as demanded by the IoT market.

In this regard, this article identifies hardware implementation challenges and thoroughly analyzes the aforementioned suitability of FPSoCs for a broad range of IoT applications involving machine-learning and artificial intelligence algorithms, which is demonstrated in two case studies, one related to deep learning and the other to the more classical evolutionary computing techniques.

Deep Learning for the IoT

In the era of the IoT, the number of sensing devices that are deployed in every facet of our day-to-day life is enormous. In recent years, many IoT applications have arisen in various domains, such as health, transportation, smart homes, and smart cities [6]. It is predicted by the U.S. National Intelligence Council that, by 2025, Internet nodes will reside in everyday things, such as food packages, furniture, and documents [7]. This expansion of IoT devices, together with cloud computing, has led to creation of an unprecedented amount of data [8], [9]. With this rapid development of the IoT, cloud computing, and the explosion of big data, the most fundamental challenge is to store and explore these volumes of data and extract useful information for future actions [9].

The main element of most IoT applications is an intelligent learning methodology that senses and understands its environment [6]. Traditionally, many machine-learning algorithms were proposed to provide intelligence to IoT devices [10]. However, in recent years, with the popularity of deep neural networks/deep learning, using deep neural networks in the domain of the IoT has received increased attention [6], [11]. Deep learning and the IoT were among the top three technology trends for 2017 announced at Gartner Symposium/ITxpo [12]. This increased interest in deep learning in the IoT domain is because traditional machine-learning algorithms have failed to address the analytic needs of IoT systems [6], which produce data at such a rapid rate and volume that they demand artificial intelligence algorithms with modern data analysis approaches. Depending on the predominant factor, volume or rate, data analytics for IoT applications can be viewed in two main categories: 1) big data analysis and 2) data stream analysis.

When focusing on data volume, the IoT is one of the major sources of big data. Analytics of the generated massive data sets directly benefit the performance and enhance capabilities of IoT systems. Extracting knowledge from such big data is not a straightforward task. It requires capabilities that go beyond the traditional inference and learning techniques [13], generally expressed with the six Vs [14], [15]:

- volume, which refers to the ability to ingest, process, and store large data sets (petabytes or even exabytes)
- velocity, which refers to the speed of data generation and frequency of delivery (sampling)

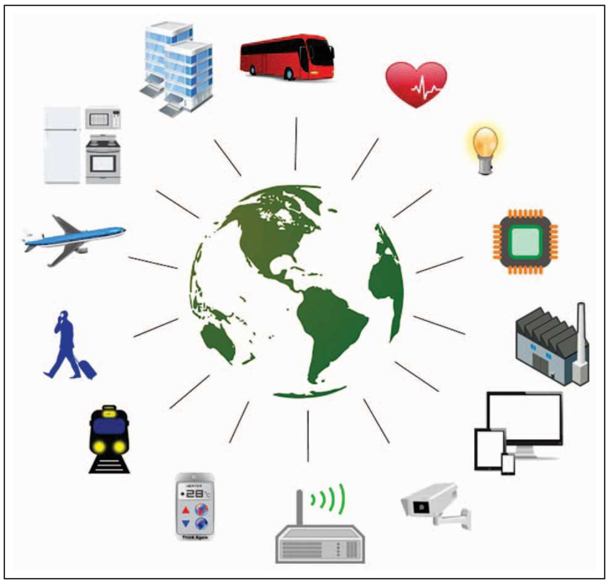


FIGURE 1 – IoT devices (adapted from [3]).

- variety, which refers to the data from different sources and types (structured or unstructured); even the types of data have been growing fast
- variability, which refers to the need for getting meaningful data considering scenarios of extreme unpredictability
- veracity, which refers to bias, noise, and abnormality in data (only the relevant, usable data within analytic models is to be stored)
- value, which refers to the purpose the solution has to address.

Figure 2 shows the six Vs of big data and how the advantages of deep-learning techniques can be used to meet these challenges in big data. More specific applications of deep-learning techniques in big data in the IoT are presented in the next section. The latest considerations add three additional Vs to the mix: vulnerability (of data), volatility (relevance of data before becoming obsolete), and visualization (ways of meaningful visualization).

As mentioned, in addition to performing data mining on massive collections of data produced by IoT systems, another important aspect is dealing with real-time data streams that require fast-learning algorithms. IoT applications, such as traffic management systems and supply chain logistics of supermarkets, involve large data sets that have to be analyzed in near real time [16]. Mining fast-generated data streams requires the algorithms to be adaptable to the change of data distributions as the environment changes around the devices [17]. This context/concept drift occurs due to the changes in factors, such as location, time, and activity. In addition to the requirement of speed adaptability, the lack of labeled data in IoT data streams adds to the difficulty because it makes supervised learning methods inadequate for analysis [17], [18]. Therefore, highly adaptable unsupervised and semisupervised deep-learning techniques are required for mining the fast-changing data streams in IoT devices.

Applications of Deep Learning in the IoT

Deep neural networks have revolutionized a multitude of fields because of their ability for learning through multiple layers of abstraction [19], [20]. This enables learning of complex patterns that are hidden in complex data sets, a capability ideal for mining massive heterogeneous data sets. Different deep neural network algorithms have been used to good effect in a range of areas that were very difficult to tackle in the past. Long short-term memory algorithms, e.g., have been shown to be extremely useful in speech recognition and natural language processing [21]–[23], and convolutional neural networks have been used to produce state-of-the-art performance in many vision applications, such as image classification [24], [25]. Therefore, deep learning is applied extensively in a range of IoT devices for human interaction.

One of the most important derivatives of the IoT is the concept of smart cities. Improving cities is becoming a global need with the rising and urbanization of the population [26]. The concept of smart cities has been around since the early 2000s. Smart cities claim to contain thousands of sensing devices, which generate massive amounts of data that can be harnessed to optimize and improve the operations of these cities [27]. Smart cities try to accomplish goals, e.g., reducing pollution and energy consumption or optimizing transportation [28]. IoT devices can help collect data about how people use cities, and machine-learning algorithms can be used to understand that data [26]. Adding further intelligence to the embedded sensing nodes allows local storage needs and network congestion to be reduced.

One of the most important aspects of smart cities powered by the IoT is smarter energy management. With the advent of smart meters, there are massive amounts of data being collected on energy consumption. This enables research on energy consumption prediction, which can lead to optimizing energy usage and the way energy is generated in smart cities and smart grids. Machine-learning algorithms are indispensable in this area, and deep-learning algorithms, such as long short-term memory algorithms, restricted Boltzmann machines, and convolutional neural networks, have been proposed to perform data-driven predictions of energy usage at both

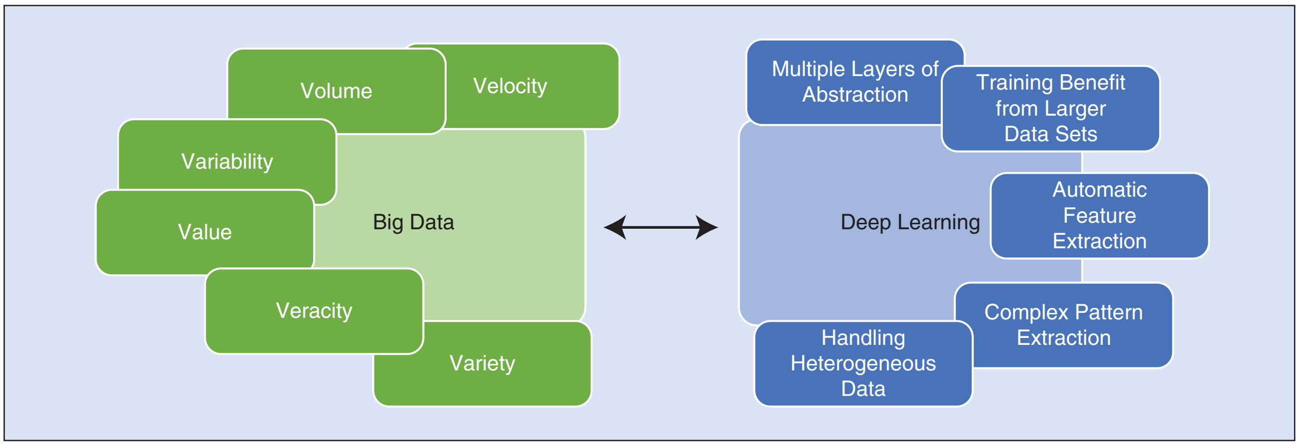

Volume
Velocity
Variability
Value
Big Data
Veracity
Variety
Multiple Layers of Abstraction
Training Benefit from Larger Data Sets
Deep Learning
Automatic Feature Extraction
Complex Pattern Extraction
Handling Heterogeneous Data

FIGURE 2 – The big data six Vs and their connection with deep learning.

the individual consumer and aggregate levels [29]–[32].

Another important aspect of smart cities is using machine learning and the IoT for traffic management. Optimized traffic management targets reducing congestion, long queues, delays, and even the carbon footprint of cities [33]. To that end, driverless or self-driving cars have become a much-discussed topic recently, with major car companies, such as Tesla, BMW, and Ford, and tech giants, such as Google and Apple, stepping up to the plate to develop truly intelligent autonomous cars.

Self-driving cars have a plethora of devices continuously sensing their environment and a suit of machine-learning algorithms for understanding and fusing the various data sources, such as LIDAR depth maps and images. Deep neural networks have been extensively explored in this domain, as they have the capability of automatically learning features to pick out obvious ones, such as lane marking and road edges, as well as other subtle ones that exist on the roads [34].

Computer vision is a highly sought-after application in many use cases in the IoT domain. Smart cameras, especially in smart security systems, play an important role in smart homes [35], and vision applications, such as face recognition, are very crucial [36]. Machine-learning algorithms have been used extensively in image-processing applications and, in that, convolutional neural networks have been deemed the gold standard since the advent of LeNet [37]. Ko et al. presented a framework for energy-efficient neural networks to be used in IoT edge devices [38]. The authors claim that in deploying deep neural networks-based image processing, energy efficiency can be the performance bottleneck, and hence, they present the recent technological advantages for making deep neural networks, such as convolutional and recurrent neural networks, more energy efficient.

Another area in which machine-learning-driven vision applications is used in the IoT is human activity recognition in smart homes. Fang and Hu proposed a deep-learning-based framework for human activity recognition in smart homes used especially for helping people with diseases [39]. Context awareness is another important aspect of the IoT, closely tied with mining data streams. Machine learning has a very crucial role to play in understanding the environment and the context of the device from the data.

In recent years, we have seen commercial IoT devices or edge devices emerging in the market, such as Nest Thermostat [40] and Amazon devices powered by Alexa [41], that have the ability of sensing their environment and using machine learning to understand data. Context-aware devices or things have the ability of understanding the environment and adapting their reasoning capabilities [10]. Further, machine-learning algorithms are extremely crucial for some areas, such as intelligent health trackers for medicine, e.g., intelligent pacemakers or photoplethysmography systems [42], [43] that can monitor the heartbeat of a patient.

Adding intelligence to these devices is very important, as it permits improved and faster preventive detection of pathologies. Compared with the option to send data via the Internet to remote sensors for analysis or saving data for postprocessing, this option enables a dramatic reduction of data transmission and storage (with the respective reduction of energy consumption) and the possibility to work offline (very useful for remote or rural areas).

Safety and Security in the IoT

In addition to enabling and facilitating IoT applications, deep learning plays a crucial role in keeping the highly connected devices safe. Due to its ubiquity in the modern technological ecosystem, the IoT is a very attractive target for cyberattackers. Therefore, cybersecurity is one of the most important research areas in the field of the IoT [44], [45]. It is known that a large number of zero-day attacks are emerging continuously due to the various protocols added to the IoT [46]. The multiple-level feature-learning capabilities of deep learning have been exploited in this domain to good effect.

Diro and Chilamkurti presented a deep neural networks-based distributed methodology for cyberattack detection in the IoT [46]. They compared their distributed deep model with a shallow neural network and a centralized deep model, and they concluded that the distributed deep model outperforms the others significantly.

Another area of cybersecurity is malware detection. Pajouh et al. presented a deep recurrent neural network-based malware detection methodology for the IoT [47]. The authors implemented three different long short-term memory configurations and showed that their algorithm can achieve 98.18% accuracy in malware detection for the tested data set. In all aspects of cybersecurity, when taking a data-driven approach, anomaly detection algorithms are very useful tools. Canedo and Skjellum presented an artificial neural network-based anomaly detection methodology tailored for IoT cybersecurity [48]. They recognized that the main challenges for anomaly detection in IoT data are quantity and heterogeneity. They showed that the artificial neural network-based methodology was able to overcome those challenges in detecting anomalies in the data sent from edge devices.

Hardware Implementation Challenges

The implementation of machine-learning algorithms has been a hot topic in research for several years but recently boomed, mainly thanks to the opportunities created by the advancements in chip fabrication technologies, which enabled solving design problems at a cost and with a time-to-market that were unthinkable just a few years ago. The resolution of Google Challenge by AlexNet using an eight-layer deep neural network [24] is usually cited as an inflexion point that boosted the research on new chips and applications of machine-learning algorithms, especially in the field of neural networks. This explosion coincides with the deceleration of Moore's law (even Gordon Moore himself predicted the end of his Moore's law [92]), which now makes it economically reasonable to work on optimized software and hardware structures, as opposed to the trend of the last 30 years, where waiting for the next generation of devices was more profitable than investing in optimization. All of these facts combined make it more difficult than ever for designers to decide the best possible architecture for their applications.

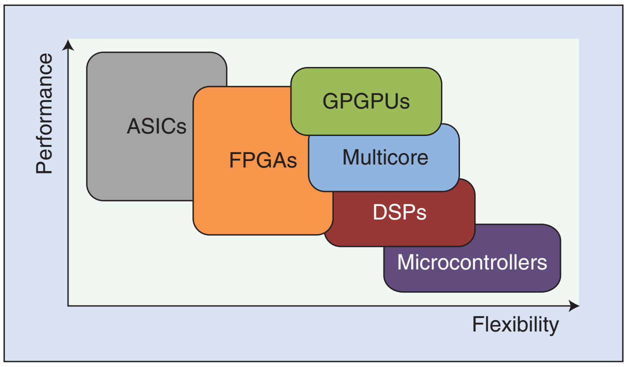


FIGURE 3 – The performance versus flexibility of digital processing platforms (adapted from [52]).

The digital processing platforms currently available in the market are summarized in Figure 3, where they can be compared in terms of performance and flexibility. Flexibility refers here to ease of development, portability, and possibility for adapting to changes in specifications. For high-end deep neural network applications, where performance is the most important parameter, general-purpose GPUs (GPGPUs) are the dominant solution. Their parallel structure, the latest efforts by manufacturers to compete for machine-learning applications (e.g., adding specific instructions for fast neuron inference), and their reduced cost due to the mass production for personal computers made them ideal for training and inference of deep neural networks.

The latest NVIDIA Volta GV100 GPU platform, including 21.1 billion transistors within a die size of 815 mm^2, is capable of doing inference 100 times faster than the fastest current central processing unit (CPU) on the market [49]. This unparalleled brute power force comes at a price: high power consumption, the need for custom data types (not necessarily float), irregular parallelism (alternating sequential and parallel processing), and divergence (not all cores executing the same code simultaneously). That is why some companies are investing in neural network application-specific integrated circuits (ASICs) for improved performance at the expense of losing flexibility. Examples are the first and second generation (optimized for inference and both inference and training, respectively) of the Google tensor processing unit (TPU), slowly stealing high-performance computing applications from GPUs.

While this is the pace for high-performance computing, the lack of flexibility in ASICs and the high power consumed by GPUs do not fit in wide areas of the IoT world that demand power-efficient, flexible embedded systems. This explains why many IoT devices are currently based on microcontrollers, digital signal processors (DSPs), and multicore CPUs. However, as the IoT market grows, both manufacturers and designers face a problem due to the diversification of applications and increasing demand for computing power (particularly for machine-learning algorithms), leading a transformation from sense making to decision making [50].

Offering a wider portfolio of devices to cover the different applications means less market share per device, increasing manufacturing costs. However, offering complex heterogeneous devices that can be used in several applications implies higher integration of functionality and a waste of silicon, also increasing the overall cost [51]. In this scenario, FPGAs, located in the middle of Figure 3, appear as a balanced solution to add flexibility and efficient computing power for machine-learning algorithms to the next generation of IoT devices. Combining processors and FPGAs in a single package results in the FPSoC concept. In the following sections, FPSoC architecture is presented along with an analysis of the usefulness of its hardware resources for implementing machine-learning algorithms in IoT devices.

FPSoC Architecture

FPSoCs feature a hard processing system (HPS) and FPGA fabric on the same chip. Both parts are connected by means of high-throughput bridges, which provide faster communications and power savings compared to multichip solutions [53]. The HPS in first-generation FPSoCs featured single- or dual-core ARM application processors and some widely used peripherals, such as timers and controllers for different types of communication protocols, i.e., Ethernet, universal serial bus (USB), interintegrated circuit (I2C), universal asynchronous receiver-transmitter (UART), and controller area network (CAN).

Pushed by increasing application requirements, some devices in the newest FPSoC families include quad-core ARM processors, GPUs, and real-time processors in the HPS, with FPSoCs becoming complex heterogeneous computing platforms. Resources in the FPGA fabric also evolved from the basic structure consisting of standard logic resources and relatively simple specialized hardware blocks (e.g., fixed-point DSP multipliers, memory blocks, and transceivers). Current devices include much more complex blocks, e.g., DSP blocks with floating-point capabilities, video codecs for video compression, soft-decision forward error recovery (SD-FEC) units to speed up encoding/decoding in wireless applications, or analog-to-digital converters (ADCs). Figure 4 shows the generic block diagram of a modern FPSoC device, where the location and connection of the aforementioned elements is depicted.

All computing elements (processors and GPU) have their own cache memory and share common synchronous dynamic random access memory (SDRAM) external memory, usually controlled by a single multiport controller. A main switch interconnects masters and slaves in the HPS. The FPGA fabric can be accessed as any other memory-mapped peripheral from the HPS through the HPS-to-FPGA bridges. There are also several options to access the HPS from the FPGA fabric: FPGA-to-HPS bridges to access HPS peripherals, the accelerator coherency port (ACP) to coherently access processor cache, and FPGA-to-SDRAM bridges to access main memory in a noncoherent way.

Not all FPSoCs include all blocks in Figure 4. Table 1 shows a summary of characteristics of the most relevant currently available FPSoC families. Intel FPGA and Xilinx offer powerful devices with application processors and large FPGA fabrics, focused on higher-end applications, such as fifth-generation communications, artificial intelligence, data centers, or video processing. Microsemi and Quicklogic offer simpler devices with real-time processors, focusing on data acquisition, wearables, and smartphones.

Despite the additional components that manufacturers provide in some

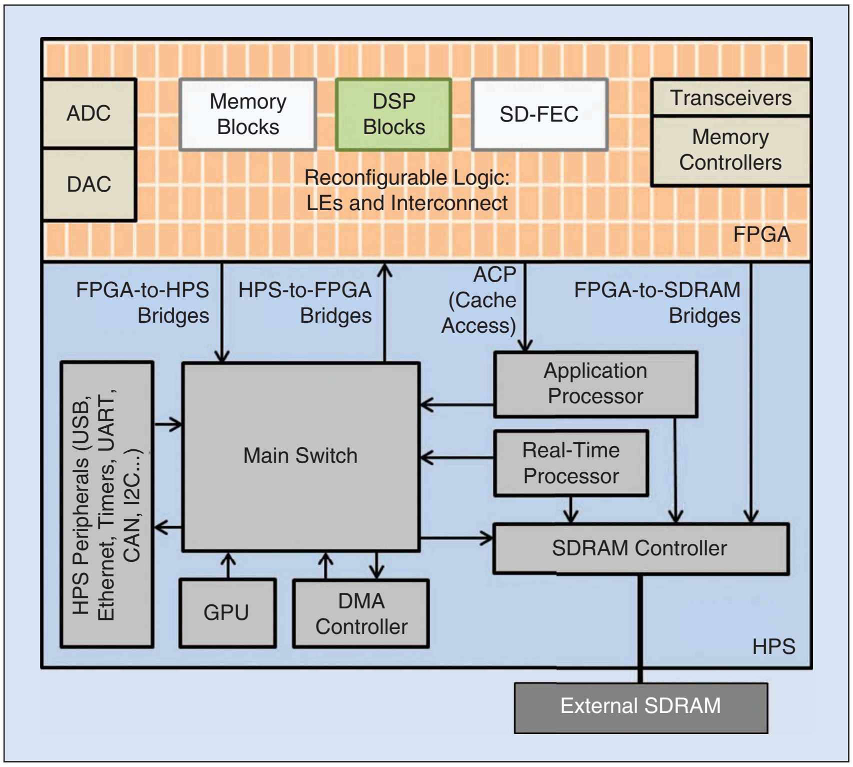


FIGURE 4 – The block diagram of a modern FPSoC. DAC: digital-to-analog converter.

TABLE 1 – THE CHARACTERISTICS OF MODERN FPSoC FAMILIES.

			APPLICATION PROCESSOR		REAL-TIME PROCESSOR		FPGA		
COMPANY	FAMILY	TRANSISTOR SIZE	TYPE	MAXIMUM F (GHz)	TYPE	MAXIMUM F (MHz)	MAXIMUM SIZE	MAXIMUM F (MHz)	OTHER
Intel FPGA	Cyclone V SoC	28 nm	Single/dual 32-bit ARM Cortex-A9	0.925	–	–	301 K LEs	200	
	Arria V SoC	28 nm	Single/dual 32-bit ARM Cortex-A9	1.05	–	–	462 K LEs	300	
	Arria 10 SoC	20 nm	Dual 32-bit ARM Cortex-A9	1.5	–	–	1.15 M LEs	500	Floating-point DSP blocks in FPGA
	Stratix 10 SoC	14 nm tri-gate	Quad 64-bit ARM Cortex-A53	1.5	–	–	5.5 M LEs	1000	Floating-point DSP blocks in FPGA
Xilinx	Zynq-7000 Artix	28 nm	Single/dual 32-bit ARM Cortex-A9	0.866	–	–	85 K LCs	–	ADC
	Zynq-7000 Kintex	28 nm	Dual 32-bit ARM Cortex-A9	1	–	–	444 K LCs	–	ADC
	Ultrascale+ Kintex	20 nm	Dual/quad 64-bit ARM Cortex-A53	1.5	Dual-cortex-R5	600	1143 K LCs	–	Option to GPU, video codec, ADC, DAC, SD-FEC
Microsemi	SmartFusion	130 nm	–	–	Single-cortex-M3	100	6 K LEs	350	ADC, nonvolatile FPGA
	SmartFusion 2	130 nm	–	–	Single-cortex-M3	166	150 K LEs	350	ADC, nonvolatile FPGA
QuickLogic	S3	–	–	–	Single-cortex-M4-F	80	–	–	DSP, power management unit

devices targeting specific applications, the most important in an FPSoC are still the HPS processors and the FPGA fabric. To successfully deploy an application taking the greatest possible advantage of these devices, processors and FPGA should smoothly cooperate with each other, executing the parts of the functionality that best fit their respective architectures, sharing data between them when needed.

A designer typically starts with a software implementation in HPS and moves to the FPGA those parts of the code that need acceleration. Communication between HPS and FPGA is not a trivial task and depends on several factors, such as data size, operating system (OS), or FPGA operating frequency. It is very important to choose the best possible mechanism for HPS–FPGA data exchange, otherwise it can impair the acceleration achieved by moving portions of the algorithms to hardware. In [54]–[56], different analyses of the influence of these factors in the transfer rate are carried out. In [56], the results of the analysis are elaborated into design guidelines to maximize the performance of FPSoC implementations.

FPGA design is typically based on hardware description languages (HDLs), which require from designers good knowledge of digital hardware. Fortunately, nowadays it is also possible to automatically compile code for both the FPGA and the HPS from high-level languages, namely C/C++ (using high-level synthesis tools, either commercial or open-source, like LegUp [57]), OpenCL, MATLAB, and LabVIEW. This gives designers with limited or no experience in digital design access to the excellent characteristics of FPSoCs. Code generated by these tools is not as optimized as that resulting from HDL workflows, but they allow design time to be dramatically reduced [58].

FPSoCs and the IoT

FPSoC characteristics make them very suitable for many IoT applications. The availability of HPS peripherals for the most popular communication protocols enables interoperability among a broad range of devices [59]. The HPS, e.g., can simultaneously connect with sensors using I2C and with other devices via Ethernet or Wi-Fi. The FPGA fabric adds great flexibility, enabling the implementation of communication protocols not included in HPS as well as specific functionalities that achieve higher performance in hardware than in software, such as pulsewidth modulation, capture and compare, or frequency measurement units.

Connectivity of IoT devices raises serious security and privacy concerns. At the hardware level, one possible way to address them is with ARM's TrustZone Technology [60], which defines some peripheral slaves as secure, so only trusted masters can access them. A secure interrupt controller, e.g., may be used to create a noninterruptible task that monitors the system, and a secure keyboard may ensure secure password entries. This concept has also been extended to software, as shown in Figure 5. A trusted firmware layer controls context switching of the processor from trusted OS and apps to regular OS and apps, which may run malicious software completely isolated from trusted software and secure hardware.

To protect intellectual property, current FPSoCs also allow the FPGA configuration bitstream as well as the boot image for the HPS to be encrypted [61]. In addition to the solutions provided by manufacturers, extra functionalities can be implemented to prevent hacker attacks. These include physically unclonable functions, useful for unique network identification, traceability, and access control [62].

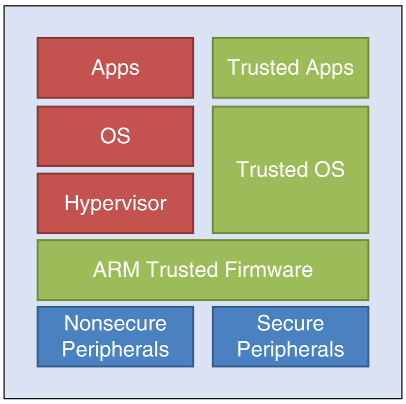


FIGURE 5 – The ARM TrustZone security (adapted from [63]).

FPSoCs enable the design of embedded systems with very small size, low power consumption, and performance sometimes even equal or higher than that of desktop platforms [64]. Regarding energy, FPSoCs largely outperform computer systems in terms of operations per second and watt [65]. FPSoCs are also more power efficient than GPU-based SoC designs [66], particularly for neural network implementations [67], [68]. However, poor usage of the available FPGA resources may result in some cases in CPUs and GPUs outperforming them [69]. With this concern in mind, FPSoCs are the best option for implementing machine learning in battery-powered systems with strict size limitations, like drones [70] or wireless sensor networks [71].

Regarding economic and marketing issues, FPSoCs are inexpensive because they are mass-produced components. Time to market is short and, thanks to the new high-level synthesis tools (like OpenCL and C/C++ compilers), similar to that of pure software solutions. Because of its reconfigurable nature, functionality can be upgraded without the need for changing the hardware platform, improving postsale support compared to nonconfigurable devices like ASICs.

FPSoCs and Machine Learning

FPGAs exhibit some unique features for efficiently implementing portions of machine-learning algorithms in hardware.

- *Parallelism*: Most machine-learning algorithms include parallelizable portions of the code that can take advantage of this property of the hardware. Each neuron in a neural network layer can be computed in parallel, e.g. In evolutionary computing, fit functions can also be concurrently executed for the whole population of genes/particles.
- *Pipelining*: Although this technique is also used in processors and GPUs to fetch and execute instructions, greater advantage of it can be taken in FPGAs, where the output of an operation can directly feed

the input of the next one, avoiding the extra clock cycles required to compute the same operations in the arithmetic/floating-point units of processors and GPUs.

- *Scalability and upgrading*: It is common for machine-learning algorithms to change structure or size (e.g., adding layers or inputs to a neural network) to improve performance from knowledge gained during test or normal operation. In a hardware/software coprocessing implementation, this may mean to port more (or new) parts of the algorithm to hardware. The same may happen in the context of the IoT when new functionality, whether related to the target machine-learning algorithm or not (such as a web server or an encryption algorithm), needs to be added to the system. The abundance of standard logic resources and specialized hardware blocks in FPGAs, together with their reconfiguration capabilities, facilitates system scalability and upgrading.

Current FPGAs include tens to hundreds of DSP blocks usually equipped with fixed-point multipliers and adders. Other operations, e.g., floating point, are implemented by a combination of these blocks and standard FPGA logic elements (LEs). FPGAs are very powerful for fixed-point operations [72] but achieve lesser performance in number of floating-point operations per second than GPUs for most machine-learning implementations [73]. However, in some cases the configurable FPGA architecture compensates this drawback and achieves faster execution times [74].

In an effort to make FPSoCs more competitive, newer devices from Intel FPGA (Arria 10 and Stratix 10 families) include DSP blocks with single floating-point capabilities in the FPGA fabric. Table 2 summarizes the size (LE and DSP block usage) and performance (latency and maximum operating frequency, f_{MAX}) of floating-point operators in Arria V and Arria 10 FPGAs for some usual floating-point operations in machine-learning algorithms. Double-precision operations require more than twice the resources and have almost twice the latency of single-precision ones. Addition, subtraction, and multiplication make low usage of resources, whereas other operators are less efficiently implemented. Using floating-point DSP blocks results in improvements in terms of either significant reduction of logic resource usage or increase of maximum operating frequency. The exception is the exponential operation, because it does not suit the fixed structure of floating-point DSP blocks well.

In low-level design with HDLs, it is easy to estimate the performance of a given algorithm implementation in a given device from the information regarding available hardware resources and latency of the different operations. This is not the case when using high-level synthesis tools, where the compiler can make inefficient use of hardware resources. To achieve acceptable performance when using these tools, it is a must to consider all of the available options to help the tool efficiently fit the design in the FPGA fabric [76].

The aforementioned hardware features are complemented in FPSoCs with those provided by the application processors in HPS. Those range from real-time processors with fixed-point arithmetic capabilities available in simpler devices to DSP-like processors for speeding up signal processing tasks, or to dedicated floating-point units or single-instruction multiple data coprocessors for vector arithmetic in more advanced devices.

Case Study 1: Implementation of Deep Neural Networks in FPSoC

Neural network algorithms and, in particular, deep neural networks are

TABLE 2 – THE RESOURCE USAGE AND LATENCY FOR USUAL FLOATING-POINT OPERATIONS IN ARRIA FPSoCS [75].

		ARRIA V (FIXED-POINT DSP BLOCKS)				ARRIA 10 (FLOATING-POINT DSP BLOCKS)			
OPERATION	FLOATING-POINT PRECISION	LATENCY (CLOCK CYCLES)	LEs	DSP BLOCKS	F_{MAX} (MHz)	LATENCY (CLOCK CYCLES)	LEs	DSP BLOCKS	F_{MAX} (MHz)
Addition/subtraction	Single	Nine	1,193	Zero	250	Five	1,208	Zero	319
	Double	12	2,903	Zero	252	Seven	2,765	Zero	290
Multiplication	Single	Five	390	One	281	Three	123	One	289
	Double	Seven	848	Four	186	Five	780	Four	289
Division	Single	18	1,140	Four	249	16	985	Four	347
	Double	35	3,523	15	185	30	3,020	15	258
Exponential base e	Single	14	1,795	Two	217	26	745	Six	365
	Double	28	5,335	Ten	185	28	5,390	Ten	260
Sine	Single	12	1,463	Three	240	11	1,463	Three	280
	Double	29	4,370	14	185	29	4,795	14	260

executed in two phases: training (where network weights are adapted to achieve the desired functionality) and inference (deployment operation of the network). Training is highly computationally demanding, so it is typically implemented by processing batches of data (several patterns at the same time) offline, for which GPUs are very suitable. The inference phase is suitable for FPGA implementation, because it typically has to be implemented over single patterns in real time and, as shown in Figure 6, the neurons in one layer can be executed in parallel. Moreover, the operations to be performed by each neuron can be very efficiently implemented using DSP blocks. These operations are

$$a_x(y) = \sigma\left(\sum_{i=0}^{n-1} a_i(y-1) * w_{ix}\right), \quad (1)$$

where $a_x(y)$ is the output of neuron x in layer y, w_{ix} is the weight between neuron i in layer y-1 and neuron x in layer y, and σ is the so-called activation function of the neuron. The classical neuron activation functions are $\mathrm{Sigmoid}(x) = 1/(1+e^{-x})$ and $\mathrm{Tanh}(x) = (e^x - e^{-x})/(e^x + e^{-x})$.

These operations involve divisions and exponentials so, according to Table 2, their FPGA implementation is not particularly efficient. Because of that, some works addressed their efficient hardware implementation using linear approximations. The use of Taylor approximations and reuse of the multipliers and adders for the linear part of the neuron is proposed in [77], reducing the additional hardware needed for the activation function to almost none. The solution in [78] incurs just 0.03% error with regard to an implementation using true exponential and division cores. However, the activation function $ReLu(x)$ = max (0, x) has recently been shown to provide better classification results and shorter training times than the former ones for deep neural networks [79], simplifying their implementation in all platforms.

Although most implementations use floating-point operations, recent works have shown that fixed-point approximations provide equal performance in some cases [80]. Moreover, for some applications it is possible to aggressively scale down (what is called *quantization*) the number of bits in fixed-point representations. In [81], e.g., it is reported that with only five-bit integer resolution for the weighting coefficients, performance degradation is negligible compared with the original 32-bit floating-point resolution.

Other operations that can be used to reduce FPGA logic resource usage are network pruning (removing non-important connections) [81], network clustering (fusing neurons) [82], and retraining (adding a penalty term in the training cost function to maximize not only the network fitting to inputs and outputs but also the bit depth needed for the network weights) [83]. These techniques, together with the use of simpler activation functions like ReLu, will surely boost the number of implementations in FPGA-based devices in the near future.

FPSoC platforms have already been used to improve pure FPGA implementation. In [84], a Zynq-7000 is used to implement an image classifier based on a deep convolutional neural network. The network layers (convolutional, pooling, and fully connected) are executed in the FPGA, whereas the HPS is responsible for synchronization [controlling direct memory access (DMA) in the FPGA] and the final steps of the classification process. A set of configurable processing elements (PEs) performs all network operations (see Figure 7). This implementation is compared against others using an Intel Xeon CPU at 2.9 GHz, an NVIDIA TK1 mobile GPU with 192 CUDA cores, and an NVIDIA K40 GPU with 2,880 CUDA cores. Results show that the FPSoC is 1.4 times faster than the CPU, with 14 times less power consumption; two times faster than the mobile GPU, with the same power consumption; and 13 times slower than the GPU, but consuming 26 times less power. This shows that FPSoCs achieve excellent performance–power consumption tradeoffs.

In [85], a Zynq-7000 is used to implement a Deep-Q network (Figure 8) that learns how to play a board game called *Trax*. Starting from a pure C/C++ software implementation and using high-level synthesis, the most time-consuming parts of the algorithm, in this case matrix multiplication of the convolutional layers, were moved to hardware. Each layer has its own matrix multiplication core that uses a double-precision floating-point multiply accumulate module to perform operations and two FPGA-SDRAM ports to share data with the processor in the HPS.

One port is used to read operands from the processor and the other to write results back. The processor executes the rest of the algorithm. Results show a 26 times acceleration with respect to the pure software implementation. Design time was very short, because hardware was directly compiled from C/C++ code using high-level synthesis, and only the most time-consuming parts of the algorithm were migrated to hardware. This example shows that

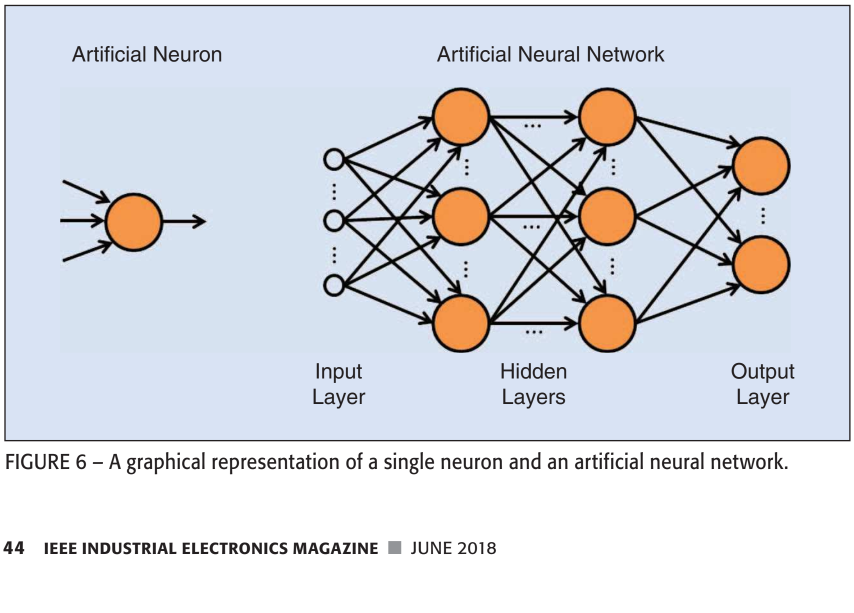


FIGURE 6 – A graphical representation of a single neuron and an artificial neural network.

high-level synthesis tools may allow impressive performance improvements to be achieved by migrating software implementations to hardware ones with little programming effort.

Artificial neural network implementation in FPGA-based devices is becoming so popular that a neural network compiler, which generates HDL code from high-level specifications, has recently been created [86]. Designers only have to select the structure, activation function, and other parameters of the artificial neural network, and the compiler automatically generates the HDL code, applying the most suitable optimization options in each case. This reduces the design time compared to using high-level synthesis, where a deep analysis of the network and the FPGA is needed to optimize the implementation.

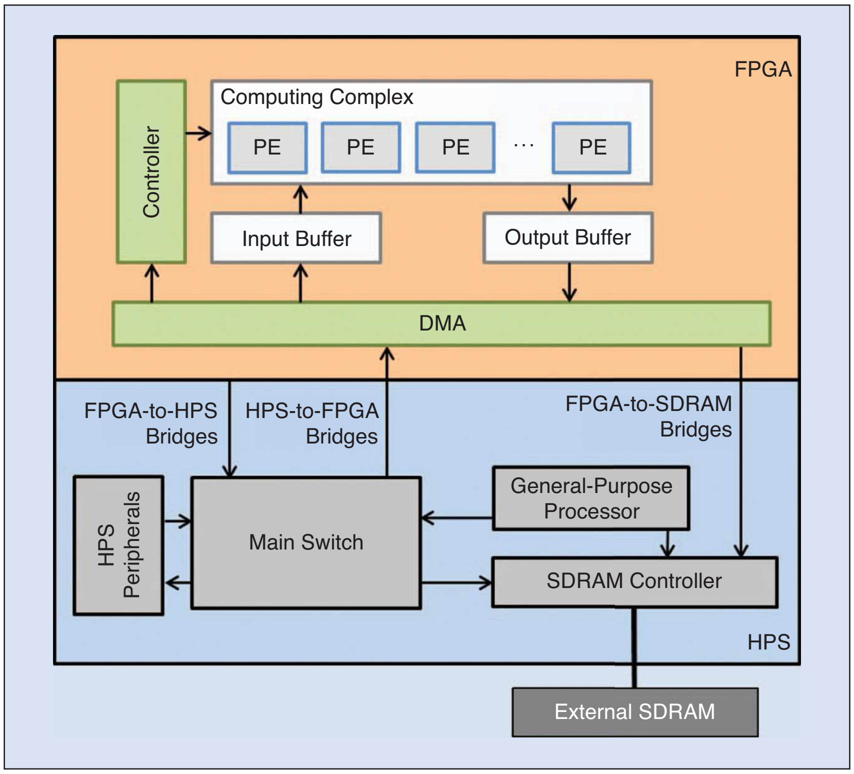


FIGURE 7 – The implementation of a deep convolutional neural network on Zynq-7000.

Case Study 2: Implementation of Evolutionary Computing in FPSoC

FPSoCs are suitable implementation platforms not only for deep-learning algorithms, such as deep neural networks, but also for other machine-learning algorithms (such as evolutionary computing ones) used in a wide range of IoT applications. Evolutionary computing algorithms are used for complex optimization problems. In them, a population of individuals (e.g., particles or genes) is spread through the solution space, and a fit function is evaluated for them, the goal being to minimize or maximize it. Depending on the values of the fit function for the different individuals in the current and past iterations, these move toward a possible solution.

After some iterations, the algorithm should converge to the global solution. Several families of such algorithms exist. They are characterized by the search policy of the individuals: ant colony optimization (which emulates ant colony food search), particle swarm optimization (which emulates the movement of a flock of birds where the distance between individuals is important), or genetic algorithms (where particles experience gene evolution through, e.g., mutation and crossover), to name just the most popular ones.

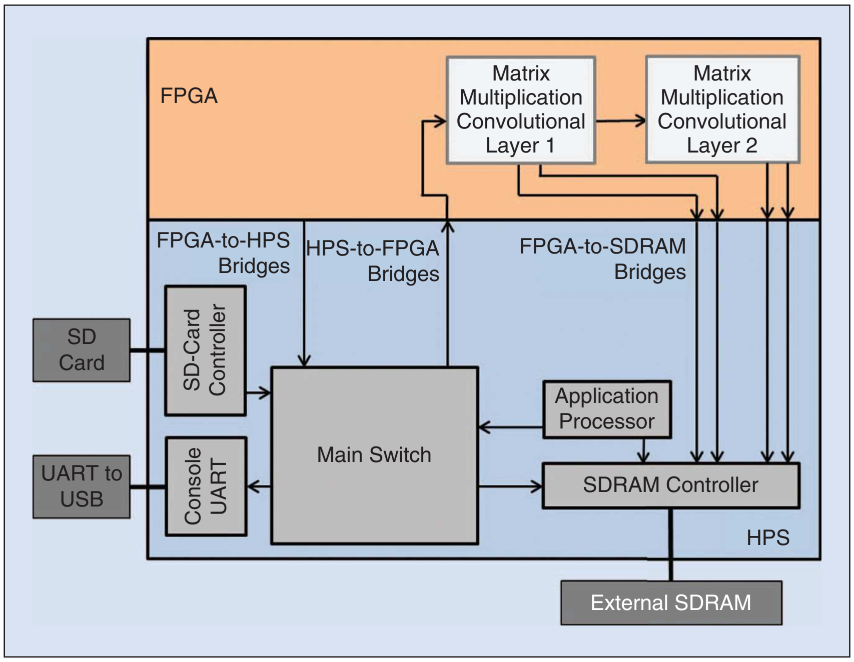


FIGURE 8 – The implementation of a Deep-Q network on Zynq-7000. SD: secure digital.

Although the fit function can be evaluated in parallel for each individual, evolutionary computing algorithms are not always as suitable for FPGA implementation as artificial neural networks because their arithmetic operations are completely dependent on the application and the algorithm used. The application defines the fit function and, depending on the operations involved, it will be more or less appropriate for FPGA implementation. Generally speaking, the more pipelineable and parallelizable the fit

FPSoCs are suitable implementation platforms not only for deep-learning algorithms but also for other machine-learning algorithms.

function, the better. Also, according to Table 2, fit functions involving multiplications and additions are more suitable for FPGA implementation than those using exponentials and divisions. The operations involved in particle movement in the aforementioned evolutionary computing algorithms are

- *ant colony*: addition, multiplication, division, exponential, square root, and random number generation [87], hence, these algorithms are not particularly suitable for FPGA implementation
- *particle swarm optimization*: multiplication, addition, and random number generation [88], which can be efficiently implemented in FPGA
- *genetic algorithms*: random number generation and movement or modifications of chromosomes [89]; processing of chromosomes perfectly fits in FPGA hardware, to the extent that it can be concurrently executed for all individuals in a single clock cycle.

Until recently, when considering the use of configurable platforms for implementing evolutionary computing algorithms, both the algorithm itself (particle movement) and the evaluation of the fit function were typically executed in hardware [88], [90]. In some cases where simple fit functions can be used, a soft processor (i.e., a processor implemented using standard FPGA logic resources) may be in charge of evaluating the fit function in software, as reported, e.g., in [91]. However, in real-life problems it is very usual that fit function evaluation takes most of the execution time, and soft processors are not fast enough to justify a software implementation, therefore most designers opted for pure hardware implementations.

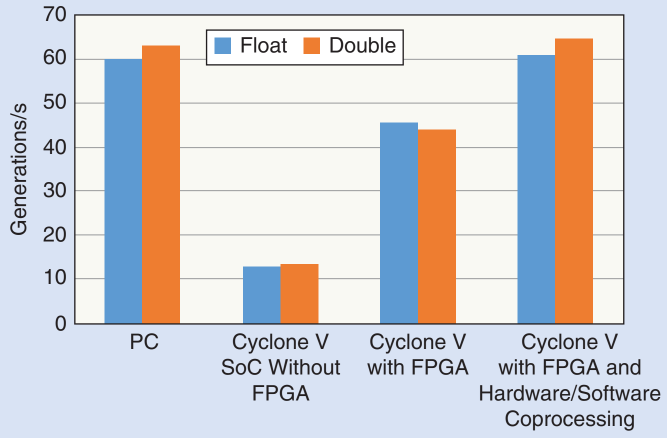
Today, the situation is different with the availability of powerful FPSoC devices, whose embedded hard processors work much faster than soft ones and have in many cases floating-point capabilities. In this scenario, the most efficient solution is to implement the evaluation of the fit function in hardware and execute the algorithm in software.

In [64], a particle swarm optimization algorithm is proposed for evaluating the state of health of solar panels located in remote areas, where human intervention is difficult. In a pure software implementation, the evaluation of the fit function takes 83% of the execution time. Using a Cyclone V SoC device, the evaluation of the fit function is moved to hardware. In a first approach, the processor waits in idle state for the FPGA to finish this evaluation. Even though, in this particular case, the fit function is neither internally parallelizable nor pipelineable, it can be concurrently computed for 12 particles, resulting in 3.4 times acceleration with regard to the pure software implementation.

An improved solution takes advantage of idle processor time for it to generate the random numbers to be used in subsequent iterations of the algorithm, resulting in 4.8 times acceleration. The achieved performance is comparable to that obtained with a desktop computer but with much lower size, cost, and power consumption, as shown in Figure 9(a). The whole monitoring system fits in a small electric box [Figure 9(b)] and can be located under each panel.

Closing Discussion

The ubiquitous deployment of machine learning and artificial intelligence across IoT devices has introduced various intelligence and cognitive capabilities. One may conclude that these capabilities



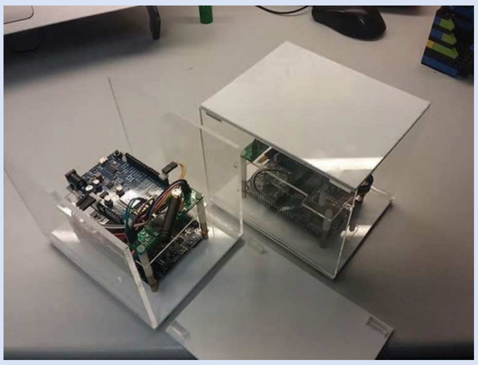


FIGURE 9 – (a) A performance comparison of particle swarm optimization algorithm for different Cyclone V SoC implementations and a desktop computer. (b) The system based on a Cyclone V SoC board.

have led to the success of a wide and ever-growing number of applications, such as object/face/speech recognition, wearable devices and biochips, diagnosis software, or intelligent security and preventive maintenance.

Developments in other areas, such as humanoid robots, self-driving cars, or smart buildings and cities, will likely revolutionize the way we live in the very near future. This new reality comes with significant advantages but also with many challenges related to the acquisition, processing, storage, exchange, sharing, and interpretation of the continuously growing, overwhelming amount of data generated by the IoT.

Up to now, complex applications involving deep neural networks have mainly used the brute force of GPUs for both training and inference. In the last two years, some companies have produced ASICs with better performance and lower power consumption than GPUs. These solutions are suitable for high-performance computing applications, but neither the low flexibility of ASICs nor the high-power consumption of GPUs is suitable for many IoT applications, which demand energy-efficient, flexible embedded systems capable of coping with the increasing diversification of the IoT.

In contrast, FPSoC architectures, which include processors and FPGA fabric in the same chip, are a balanced solution to implement machine-learning applications for IoT devices. The latest advancements in FPGA hardware allow a wide range of machine-learning algorithms to be efficiently implemented. FPGAs are very well suited to perform deep neural network inference because of the parallel arrangement of neurons in layers and the type of mathematical functions they have to compute. This will be even more so in the future because of the trend to use simpler neuron activation functions (like ReLu) that, in addition to improving training, fit better in FPGA resources. Moreover, the use of quantization techniques and custom data types (which is difficult to achieve, if possible at all, in devices with fixed architectures like ASICs and GPUs) can significantly reduce complexity and improve performance. In our opinion, the trends for neural network implementation in IoT devices in the following years can be summarized as follows.

The ubiquitous deployment of machine learning and artificial intelligence across IoT devices has introduced various intelligence and cognitive capabilities.

- Training will rely on heavy-duty cloud-based GPUs. ASICs like the new Google TPU (optimized for both inference and training, with impressive performance) will have a piece of the pie here, but with the limitation posed by their lack of flexibility.
- The simplest IoT devices will use CPUs and ASICs for inference to reduce cost and power consumption, respectively. Larger devices will use FPGAs/FPSoCs for inference because of their balanced flexibility and computer power. For heavy-duty inference, the same considerations as for training apply.

FPSoCs are an excellent alternative for evolutionary computing, because they allow the algorithm itself to be executed in software while the objective function can be computed in parallel in hardware for all individuals. However, their efficiency in this context greatly depends on whether or not the specific operations involved in the computation of the objective function fit available hardware resources. It can be concluded that, thanks to the availability of hard processors with floating-point units, FPSoCs are very suitable for implementing evolutionary computing algorithms. In the case of particle swarm, it has been discussed how the same performance as a desktop computer can be achieved with FPSoCs with a fraction of the size, cost, and power consumption.

In our opinion, the implementation in FPSoCs of IoT devices with machine-learning capabilities will be boosted by the availability of increasingly efficient high-level synthesis tools based on widely known and used languages, such as OpenCL, C/C++, or MATLAB, enabling software designers to take advantage of the excellent characteristics of FPSoC devices.

Acknowledgments

Roberto Fernandez Molanes's and Juan J. Rodriguez-Andina's work in this article has been supported by the Spanish Ministerio de Economia y Competitividad under grant TEC2014-56613-C2-1-P.

Biographies

Roberto Fernandez Molanes (robertofem@uvigo.es) received his M.Sc. degree in electrical engineering and M.Sc. degree in advanced technologies and processes in industry from the University of Vigo, Spain, in 2012 and 2013, respectively, where he is currently working toward his Ph.D. degree in the Department of Electronic Technology. His research interests include the design of hardware/software coprocessing systems and high-performance instrumentation using field-programmable system-on-chip platforms. He is a Student Member of the IEEE and a member of the IEEE Industrial Electronics Society.

Kasun Amarasinghe (amarasinghek@vcu.edu) received his B.Sc. degree in computer science from the University of Peradeniya, Sri Lanka, in 2011. He is currently reading for his doctoral degree in computer science at Virginia Commonwealth University, Richmond. His research interests include interpretable machine learning, fuzzy systems, deep learning, data mining, and natural language processing. His interests also extend to applications of such algorithms to a multitude of domains, including cyberphysical systems and energy systems. He has gained experience on Internet of Things systems from multiple research projects funded by the U.S. Department of Energy and industry leaders. He is a Student Member of the IEEE and a member of the IEEE Industrial Electronics Society.

Juan J. Rodriguez-Andina (jjrdguez@uvigo.es) received his M.Sc. degree from the Technical University of Madrid, Spain, in 1990 and his Ph.D. degree from the University of Vigo, Spain, in 1996, both in electrical engineering. He is an associate professor in the Department of Electronic Technology, University of Vigo. In 2010–2011, he was on sabbatical leave as a visiting professor at the Advanced Diagnosis, Automation, and Control Laboratory, Electrical and Computer Engineering Department, North Carolina State University, Raleigh. His research interests include the implementation of complex control and processing algorithms and intelligent sensors in embedded platforms. He has authored more than 160 journal and conference articles and holds several Spanish, European, and U.S. patents. He is a Senior Member of the IEEE and a member of the IEEE Industrial Electronics Society.

Milos Manic (misko@ieee.org) received his M.S. degree in computer science from the University of Nis, Serbia, in 1996 and his Ph.D. degree in computer science from the University of Idaho in 2003. He is a professor in the Computer Science Department and director of the Modern Heuristics Research Group at Virginia Commonwealth University, Richmond. He has more than 20 years of academic and industrial experience leading more than 30 research grants focusing on machine and deep learning in energy, resilience, cybersecurity, and human–system interaction in mission-critical infrastructures. He is a founder of the IEEE Industrial Electronics Society Technical Committee on Resilience and Security in Industry. He has published more than 180 refereed articles in international journals, books, and conferences and holds several U.S. patents. He built his expertise through research on a number of U.S. Department of Energy and industry-funded projects. He is a Senior Member of the IEEE and an IEEE Industrial Electronics Society Senior AdCom member.

References

[1] M. Jaffe. (2014). IoT won't work without artificial intelligence. *WIRED.* [Online]. Available: https://www.wired.com/insights/2014/11/iot-wont-work-without-artificial-intelligence/

[2] E. Sappin. (2017, June 28). How AI and IoT must work together. *VentureBeat.* [Online]. Available: https://venturebeat.com/2017/06/28/how-ai-and-iot-must-work-together/

[3] E. Ahmed, I. Yaqoob, I. Abaker Targio Hashem, I. Khan, A. Ibrahim Abdalla Ahmed, M. Imran, and A. V. Vasilakos, "The role of big data analytics in Internet of Things," *Comput. Netw.*, vol. 129, pp. 459–471, Dec. 2017.

[4] B. Del Monte and R. Prodan, "A scalable GPU-enabled framework for training deep neural networks," in *Proc. 2016 2nd Int. Conf. Green High Performance Computing (ICGHPC 2016).*

[5] M. D. Valdés Peña, J. J. Rodriguez-Andina, and M. Manic, "The Internet of Things: The role of reconfigurable platforms," *IEEE Ind. Electron. Mag.*, vol. 11, no. 3, pp. 6–19, Sept. 2017.

[6] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani. (2017). Deep learning for IoT big data and streaming analytics: A survey. arXiv. [Online]. Available: https://arxiv.org/abs/1712.04301

[7] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sept. 2013.

[9] F. Chen, P. Deng, J. Wan, D. Zhang, A. V. Vasilakos, and X. Rong, "Data mining for the Internet of Things: Literature review and challenges," *Int. J. Distrib. Sens. Netw.*, vol. 11, no. 8, pp. 1–14, Aug. 2015.

[10] C. Perera, S. Member, A. Zaslavsky, and P. Christen, "Context aware computing for the Internet of Things: A survey," *Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 1–41, 2014.

[11] J. Tang, D. Sun, S. Liu, and J. L. Gaudiot, "Enabling deep learning on IoT devices," *Comput.*, vol. 50, no. 10, pp. 92–96, 2017.

[12] K. Panetta. (2016, Oct. 18). Gartner's top 10 strategic technology trends for 2017. *Smarter with Gartner.* [Online]. Available: https://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017/

[13] X.-W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, May 2014.

[14] M. Hilbert, "Big data for development: A review of promises and challenges," *Dev. Policy Rev.*, vol. 34, no. 1, pp. 135–174, Jan. 2016.

[15] H. Hu, Y. Wen, T. S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, May 2014.

[16] A. Akbar, A. Khan, F. Carrez, and K. Moessner, "Predictive analytics for complex IoT data streams," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1571–1582, Oct. 2017.

[17] D. Nallaperuma, D. De Silva, D. Alahakoon, and X. Yu, "A cognitive data stream mining technique for context-aware IoT systems," in *Proc. IECON 2017—43rd Annu. Conf. IEEE Industrial Electronics Society*, pp. 4777–4782.

[18] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, *Towards a Better Understanding of Context and Context-Awareness.* New York: Springer-Verlag, 1999, pp. 304–307.

[19] I. Goodfellow, B. Yoshua, and A. Courville, *Deep Learning.* Cambridge, MA: MIT Press, 2016.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[21] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. on Machine Learning*, 2015, pp. 2048–2057.

[22] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. (2014). Translating videos to natural language using deep recurrent neural networks. arXiv. [Online]. Available: https://arxiv.org/abs/1412.4729

[23] S. Wang and J. Jiang. (2015). Learning natural language inference with LSTM. arXiv. [Online]. Available: https://arxiv.org/abs/1512.08849

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[25] K. Simonyan and A. Zisserman. (2014). Very deep convolutional networks for large-scale image recognition. arXiv. [Online]. Available: https://arxiv.org/abs/1409.1556

[26] J. Walker. (2017, Sept. 14). Smart city artificial intelligence applications and trends. *TechEmergence.* [Online]. Available: https://www.techemergence.com/smart-city-artificial-intelligence-applications-trends/

[27] J. Chin, V. Callaghan, and I. Lam, "Understanding and personalising smart city services using machine learning, the Internet-of-Things and big data," in *Proc. 2017 IEEE 26th Int. Symp. on Industrial Electronics (ISIE)*, pp. 2050–2055.

[28] E. Woyke. (2018, Feb. 21). A smarter smart city. *MIT Technology Review.* [Online]. Available: https://www.technologyreview.com/s/610249/a-smarter-smart-city/

[29] D. Marino, K. Amarasinghe, and M. Manic, "Building energy load forecasting using deep neural networks," in *Proc. IECON 2016 42nd Annu. Conf. IEEE Industrial Electronics Society*, pp. 7046–7051.

[30] K. Amarasinghe, D. L. Marino, and M. Manic, "Deep neural networks for energy load forecasting," in *Proc. IEEE Int. Symp. Industrial Electronics*, 2017, pp. 1483–1488.

[31] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustain. Energy Grids Netw.*, vol. 6, pp. 91–99, June 2016.

[32] M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, and C. Rieger, "Intelligent buildings of the future: Cyberaware, deep learning powered, and human interacting," *IEEE Ind. Electron. Mag.*, vol. 10, no. 4, Dec. 2016.

[33] R. J. F. Rossetti, "Traffic control & management systems in smart cities," *Readings Smart Cities*, vol. 2, no. 3, 2016.

[34] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, (2017). Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv. [Online]. Available: https://arxiv.org/abs/1704.07911

[35] C.-R. Yu, C.-L. Wu, C.-H. Lu, and L.-C. Fu, "Human localization via multi-cameras and floor sensors in smart home," in *Proc. 2006 IEEE Int. Conf. Systems, Man and Cybernetics*, pp. 3822–3827.

[36] A. H. M. Amin, N. M. Ahmad, and A. M. M. Ali, "Decentralized face recognition scheme for distributed video surveillance in IoT-cloud infrastructure," in *Proc. 2016 IEEE Region 10 Symp. (TENSYMP 2016)*, pp. 119–124.

[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, Nov. 1998.

[38] J. H. Ko, Y. Long, M. Faisal Amir, D. Kim, J. Kung, T. Na, A. Ranjan Trivedi, and S. Mukhopadhyay, "Energy-efficient neural image processing for Internet-of-Things edge devices," in *Proc. Midwest Symp. Circuits Systems*, 2017, pp. 1069–1072.

[39] H. Fang and C. Hu, "Recognizing human activity in smart home using deep learning algorithm," in *Proc. 33rd Chinese Control Conf.*, 2014, pp. 4716–4720.

[40] Nest Labs. (2018). What makes a Nest thermostat a Nest thermostat? *Nest Labs.* [Online]. Available: https://nest.com/thermostats/

[41] G. Anders. (2017, Aug. 9). “Alexa, understand me.” *MIT Technology Review*. [Online]. Available: https://www.technologyreview.com/s/608571/alexa-understand-me/

[42] B. Marsh. (2018). The intelligent pacemaker that can talk to your doctor. *Daily Mail*. [Online]. Available: http://www.dailymail.co.uk/health/article-122444/The-intelligent-pacemaker-talk-doctor.html

[43] W. R. Dassen, K. Dulk, and H. J. Wellens, “Modern pacemakers: Implantable artificial intelligence?” *Pacing Clin. Electrophysiol.*, vol. 11, no. 11, pp. 2114–2120, Nov. 1988.

[44] J. Pacheco and S. Hariri, “IoT security framework for smart cyber infrastructures,” in *Proc. 2016 IEEE 1st Int. Workshops Foundations and Applications of Self* Systems (FAS*W)*, pp. 242–247.

[45] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and privacy challenges in industrial Internet of Things,” in *Proc. 52nd Annu. Design Automation Conf. (DAC ’15)*, 2015, pp. 1–6.

[46] A. A. Diro and N. Chilamkurti, “Distributed attack detection scheme using deep learning approach for Internet of Things,” *Future Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.

[47] H. Haddad Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “A deep recurrent neural network based approach for Internet of Things malware threat hunting,” *Future Gener. Comput. Syst.*, vol. 85, pp. 88–96, Aug. 2018.

[48] J. Canedo and A. Skjellum, “Using machine learning to secure IoT systems,” in *Proc. 2016 14th Annu. Conf. Privacy Security Trust (PST 2016)*, pp. 219–222.

[49] NVIDIA. (2018). NVIDIA Volta. *NVIDIA*. [Online]. Available: https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/

[50] Y. Pu, C. Shi, G. Samson, D. Park, K. Easton, R. Beraha, A. Newham, M. Lin, V. Rangan, K. Chatha, D. Butterfield, and R. Attar, “A 9-mm2 ultra-low-power highly integrated 28-nm CMOS SoC for Internet of Things,” *IEEE J. Solid-State Circuits*, vol. 53, no. 3, pp. 1–13, 2018.

[51] C. Trigas, “Design challenges for system-in-package vs system-on-chip,” in *Proc. IEEE 2003 Custom Integrated Circuits Conf.*, 2003, pp. 663–666.

[52] J. J. Rodríguez Andina, E. de la Torre-Arnanz, and M. D. Valdes Peña, *FPGAs : Fundamentals, Advanced Features, and Applications in Industrial Electronics*. Boca Raton, FL: CRC, 2017.

[53] S. S. Iyer, “Heterogeneous integration for performance and scaling,” *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 6, no. 7, pp. 973–982, July 2016.

[54] M. Sadri, C. Weis, N. Wehn, and L. Benini, “Energy and performance exploration of accelerator coherency port using Xilinx ZYNQ,” in *Proc. 10th FPGA World Conf.* 2013, pp. 1–8.

[55] L. Costas, R. Fernandez-Molanes, J. J. Rodriguez-Andina, and J. Farina, “Characterization of FPGA-master ARM communication delays in zynq devices,” in *Proc. 2017 IEEE Int. Conf. Industrial Technology (ICIT)*, pp. 942–947.

[56] R. Fernandez-Molanes, J. J. Rodriguez-Andina, and J. Farina, “Performance characterization and design guidelines for efficient processor–FPGA communication in Cyclone V FPSoCs,” *IEEE Trans. Ind. Electron.*, vol. 65, no. 5, pp. 4368–4377, May 2018.

[57] B. Fort, A. Canis, J. Choi, N. Calagar, R. Lian, S. Hadjis, Y. T. Chen, M. Hall, B. Syrowik, T. Czajkowski, S. Brown, and J. Anderson, “Automating the design of processor/accelerator embedded systems with LegUp high-level synthesis,” in *Proc. 2014 12th IEEE Int. Conf. Embedded and Ubiquitous Computing*, pp. 120–129.

[58] Altera Corporation, “Implementing FPGA design with the OpenCL standard,” Altera, San Jose, CA, WP-01173-3.0, 2013.

[59] N. Cardoso, P. Garcia, T. Gomes, F. Salgado, P. Rodrigues, J. Cabral, J. Mendes, and A. Tavares, “Multi-camera home appliance network: Handling device interoperability,” in *Proc. IEEE 10th Int. Conf. Industrial Informatics*, 2012, pp. 69–74.

[60] ARM Security Technology, “Building a secure system using TrustZone technology,” ARM, San Jose, CA, PRD29-GENC-009492, 2009.

[61] Y. Liu, J. Briones, R. Zhou, and N. Magotra, “Study of secure boot with a FPGA-based IoT device,” in *Proc. Midwest Symp. Circuits Systems*, 2017, pp. 1053–1056.

[62] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, “Implementation and characterization of a physical unclonable function for IoT: A case study with the TERO-PUF,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 97–109, Jan. 2018.

[63] ARM. (2018). Security on ARM TrustZone. *ARM*. [Online]. Available: https://www.arm.com/products/security-on-arm/trustzone

[64] R. Fernandez-Molanes, M. Garaj, W. Tang, J. J. Rodriguez-Andina, J. Farina, K. F. Tsang, and K. F. Man, “Implementation of particle swarm optimization in FPSoC devices,” in *Proc. 2017 IEEE 26th Int. Symp. Industrial Electronics (ISIE)*, pp. 1274–1279.

[65] S. Sridharan, P. Durante, C. Faerber, and N. Neufeld, “Accelerating particle identification for high-speed data-filtering using OpenCL on FPGAs and other architectures,” in *Proc. 2016 26th Int. Conf. Field Programmable Logic and Applications (FPL)*, pp. 1–7.

[66] D. Mahajan, J. Park, E. Amaro, H. Sharma, A. Yazdanbakhsh, J. K. Kim, and H. Esmaeilzadeh, “TABLA: A unified template-based framework for accelerating statistical machine learning,” in *Proc. 2016 IEEE Int. Symp. High Performance Computer Architecture (HPCA)*, pp. 14–26.

[67] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, “Angel-Eye: A complete design flow for mapping CNN onto embedded FPGA,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 35–47, Jan. 2018.

[68] A. X. M. Chang and E. Culurciello, “Hardware accelerators for recurrent neural networks on FPGA,” in *Proc. 2017 IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 1–4.

[69] J. Lachmair, T. Mieth, R. Griessl, J. Hagemeyer, and M. Porrmann, “From CPU to FPGA—Acceleration of self-organizing maps for data mining,” in *Proc. Int. Joint Conf. Neural Networks*, 2017, pp. 4299–4308.

[70] W. Fang, Y. Zhang, B. Yu, and S. Liu. (2017). FPGA-based ORB feature extraction for real-time visual SLAM. arXiv. [Online]. Available: https://arxiv.org/abs/1710.07312

[71] T. Mekonnen, M. Komu, R. Morabito, T. Kauppinen, E. Harjula, T. Koskela, and M. Ylianttila, “Energy consumption analysis of edge orchestrated virtualized wireless multimedia sensor networks,” *IEEE Access*, vol. 6, pp. 5090–5100, Dec. 2017.

[72] X. Ma, W. A. Najjar, and A. K. Roy-Chowdhury, “Evaluation and acceleration of high-throughput fixed-point object detection on FPGAs,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 6, pp. 1051–1062, June 2015.

[73] M. Sit, R. Kazami, and H. Amano, “FPGA-based accelerator for losslessly quantized convolutional neural networks,” in *Proc. 2017 Int. Conf. Field Programmable Technology (ICFPT)*, pp. 295–298.

[74] H. Nakahara, A. Jinguji, T. Fujii, and S. Sato, “An acceleration of a random forest classification using Altera SDK for OpenCL,” in *Proc. 2016 Int. Conf. Field-Programmable Technology (FPT)*, pp. 289–292.

[75] Altera Corporation, “Floating-point IP cores user guide,” Altera, San Jose, CA, UG-01058, 2016.

[76] R. Domingo, R. Salvador, H. Fabelo, D. Madroñal, S. Ortega, R. Lazcano, E. Juarez, G. Callico, and C. Sanz, “High-level design using Intel FPGA OpenCL: A hyperspectral imaging spatial-spectral classifier,” in *Proc. 2017 12th Int. Symp. Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, pp. 1–8.

[77] R. Finker, J. Echanobe, I. del Campo, and K. Basterretxea, “Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementation of artificial neurons,” *Electron. Lett.*, vol. 49, no. 25, pp. 1598–1600, Dec. 2013.

[78] S. Gomar, M. Mirhassani, and M. Ahmadi, “Precise digital implementations of hyperbolic tanh and sigmoid function,” in *Proc. 2016 50th Asilomar Conf. Signals, Systems and Computers*, pp. 1586–1589.

[79] F. Ertam and G. Aydin, “Data classification with deep learning using Tensorflow,” in *Proc. 2017 Int. Conf. Computer Science and Engineering (UBMK)*, pp. 755–758.

[80] S. Shin, K. Hwang, and W. Sung, “Fixed-point performance analysis of recurrent neural networks,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing 2016*, pp. 976–980.

[81] M. Shah, J. Wang, D. Blaauw, D. Sylvester, H.-S. Kim, and C. Chakrabarti, “A fixed-point neural network for keyword detection on resource constrained hardware,” in *Proc. 2015 IEEE Workshop Signal Processing Systems (SiPS)*, pp. 1–6.

[82] X. Zhang, A. Ramachandran, C. Zhuge, D. He, W. Zuo, Z. Cheng, K. Rupnow, and D. Chen, “Machine learning on FPGAs to face the IoT revolution,” in *Proc. 2017 IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, pp. 894–901.

[83] R. Doshi, K.-W. Hung, L. Liang, and K.-H. Chiu, “Deep learning neural networks optimization using hardware cost penalty,” in *Proc. 2016 IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 1954–1957.

[84] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, “Going deeper with embedded FPGA platform for convolutional neural network,” in *Proc. 2016 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA ’16)*, pp. 26–35.

[85] N. Sugimoto, T. Mitsuishi, T. Kaneda, C. Tsuruta, R. Sakai, H. Shimura, and H. Amano, “Trax solver on Zynq with deep Q-network,” in *Proc. 2015 Int. Conf. Field Programmable Technology (FPT)*, pp. 272–275.

[86] Y. Wang, J. Xu, Y. Han, H. Li, and X. Li, “DeepBurning: Automatic generation of FPGA-based learning accelerators for the neural network family,” in *Proc. Design Automation Conf. (DAC)*, 2016.

[87] C.-F. Juang, C.-M. Lu, C. Lo, and C.-Y. Wang, “Ant colony optimization algorithm for fuzzy controller design and its FPGA implementation,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 3, pp. 1453–1462, Mar. 2008.

[88] W. Wang, A. C.-F. Liu, H. S.-H. Chung, R. W.-H. Lau, J. Zhang, and A. W.-L. Lo, “Fault diagnosis of photovoltaic panels using dynamic current–voltage characteristics,” *IEEE Trans. Power Electron.*, vol. 31, no. 2, pp. 1588–1599, Feb. 2016.

[89] T. M. Chan, K. F. Man, K. S. Tang, and S. Kwong, “A jumping gene paradigm for evolutionary multiobjective optimization,” *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 143–159, Apr. 2008.

[90] N. N. Morsi, M. B. Abdelhalim, and K. A. Shehata, “Efficient hardware implementation of PSO-based object tracking system,” in *Proc. 2013 Int. Conf. Electronics, Computer and Computation (ICECCO)*, pp. 155–158.

[91] S.-A. Li, C.-C. Wong, C.-J. Yu, and C.-C. Hsu, “Hardware/software co-design for particle swarm optimization algorithm,” in *Proc. 2010 IEEE Int. Conf. Systems, Man and Cybernetics*, pp. 3762–3767.

[92] R. Courtland. (2015, Mar. 30). Gordon Moore: The man whose name means progress. *IEEE Spectrum*. [Online]. Available: https://spectrum.ieee.org/computing/hardware/gordon-moore-the-man-whose-name-means-progress