Internet of Things Introducing FPGAs in heterogeneous systems

Gerd Alliu and Betim Cico

October 14, 2019

Internet of Things
Introducing FPGAs in heterogeneous systems

Gerd Alliu and Betim Cico

Epoka University, Tirana, Albania

Abstract

The purpose of this study is to identify a new design concept in Internet of Things by exploring new patterns in which different technologies, devices and protocols can integrate with one another with the aim of defining the due processes of data acquisition, aggregation and transmission from the perspective of Field Programmable Gate Arrays (FPGAs), without inherently presenting major constraints to the designer.
To serve this purpose, different topologies and infrastructures can be proposed, with the designs that utilize higher processing rates at the center of the system being deprecated as opposed to the ones that perform more processing at the edge of the network. Current standards implicitly require system designs to follow the 'edge processing' paradigm, leading to the proposal of having most of the processing being performed by FPGAs in a heterogenous context of protocols and devices. A thorough introduction of the technology that enables efficient processing in FPGAs is presented. In addition, the benefits of parallel processing, a feature that the device provides us with, are subsequently discussed. Different ways of making the paradigm work are also shown, with one possible design elaborated in detail.
The implications arising throughout the design should provide a clear view on the economic and technical feasibility of the design, highlighting power proficiency as one key benefit alongside communication challenges as a potential drawback. The findings of this study will hopefully provide insights on the possible ways of increasing processing proficiency while following current industrial standards and well-known best practices.

Keywords: Internet of Things, Field Programmable Gate Arrays, Edge Processing

1 Introduction and Background

The first time the concept of Internet of Things was presented to technical audiences was in 1999, with Kevin Ashton linking the idea of RFID (Radio Frequency Identification) to the newly emerging Internet earning him recognition as an IoT pioneer [1]. Another definition of IoT has been provided by the IoT Global Standards Initiative, which represented one of the many initiatives of the agency in the United Nations, known as the International Telecommunication Union. The definition is as follows [2]:

"IoT is a global infrastructure for the information society, enabling advanced services interconnecting both physical and virtual things based on existing and evolving interoperable information and communication technologies".

The definition is clear and precise for most purposes. Nevertheless, certain questions are not answered by the definition in [3], such as:

How expensive can it be to operate many devices continuously?

Will these devices operate in a centralized manner or a decentralized one?

Do we need special purpose devices and protocols to achieve this vision?

The main objective of this paper is to identify IoT as system that can be regarded as a collection of devices that obtain unstructured information about their physical surroundings, process that information and transmit it to different participants in the system with the sole purpose of achieving what is known as Context Awareness.
In that regard, a list of the main characteristics of IoT subsystems, suited to the IERC standards, are outlined below [4].

Interconnectivity. Anything in IoT can be interconnected with the global information and communication infrastructure

Things-related services. IoT systems are capable of providing thing-related services within the constraints of things, such as privacy protection and semantic consistency between physical things and virtual things.

Heterogeneity. Devices in IoT are heterogeneous based on differences in hardware platforms and networks

Dynamic changes. The state of devices is always alternating among sleeping, waking-up and disconnected states. Moreover, the number of devices can change dynamically and the context of devices can also change based on location and processing speed [5].

Enormous scale. The number of devices that will need to be managed will be at least an order of magnitude larger than the current number of internet-connected devices.
The amount of communication actions triggered by humans will decrease and that of those triggered by devices will increase, meaning automation will increase as a result [3]. Moreover, the amount of data will increase and will require more efficient data handling and semantics [6].

1.1 Communication

Devices must communicate in a continuous and timely manner, obeying principles such as mobility, fault-tolerance and device discovery.

The main challenges arise when defining communication patterns for IoT systems because in a network designed to suit different needs, devices are likely to be produced from different vendors, thus speaking different protocols, usually proprietary ones. The only good news is that tech companies are trying to follow international standards, building devices which rely on the Internet Protocol at the Network Layer [2]. This has led to devices being identified as `Things` whenever they are connected to the Internet or to a proxy with Internet Access, presuming that they shall work with some degree of autonomy

1.2 Things

Things can refer to a wide variety of devices which are uniquely identifiable and which exhibit characteristic of embedded computing devices, all within a network. Such devices are the sensors built-in in automobiles, smart thermostatic sensors, UV cameras etc. Such devices are required to have capabilities such as data processing, monitoring, transmission of data and human interaction

--

In order to provide a discussion well-aligned with the aforementioned standards, the remainder of this paper is organized in the following way. In Section 2, a technical overview of Field Programmable Gate Arrays is provided based on literature. The communication considerations of IoT systems suitable to FGPAs are discussed in Section 3. Section 4 then offers an overview of the organization of a system, introducing design topologies and the means of implementation. The feasibility of the proposed solution is discussed in Section 5, yielding several conclusions

2 Field Programmable Gate Arrays

2.1 History and Overview

The beginnings of FPGAs are attributed to the initiative of Steve Casselman, who was the first person to have the idea of building a device that can contain 600,000 reprogrammable logic-gates. Casselman’s vision was indeed successful, as he conducted an experiment and later on obtained a patent for his innovation. However, the main concepts of reconfigurable computing had already been introduced, mostly in the form of early PLDs (Programmable Logic Devices), which played an important role in the creation of commercially viable FPGAs by industry giants Altera and Xilinx.

The device is of special value to the purpose of this paper, given its flexible design consisting of three main elements [8]:

1. Programmable logic blocks
2. Programmable interconnects, known as routes
3. I/O blocks that are used to make off-chip connections

The programmable logic blocks are generally organized in the form of a two-dimensional array, which we usually refer to as the 'grid'.

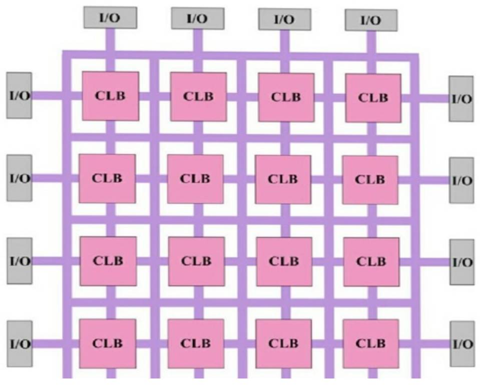

Figure 1. Inner organization of FPGA

Configurable logic blocks are the main units of interest for our considerations, as they are the containers of multiplexor units as well as gate arrays.

As seen in figure 1, the Logic Blocks are denoted as CLB (Configurable Logic Blocks). These blocks are shown as connected to I/O output blocks through several interconnection buses. Also, the blocks are organized in the form of a grid, as mentioned earlier.

Generally speaking, these blocks can consist of simple transistors in what we call fine-grained organization, or they can consist of entire processing units in what we call coarse-grained organization. Both organizations have their downsides, the first one being requiring many interconnects and using too much area, and the latter being unnecessarily complex. Between these two extremities, there exists a range of basic blocks, known as BLE (Basic logic elements) [8], which in turn consist of NAND-gates, one Look-Up table and several multiplexors, which is also the case of most general-purpose FPGAs.

This inner organization of FPGAs indicates that there is a certain level of hierarchy built into the device, and the approach to achieving re-programmability can be simply stated as a level by level interpretation of instructions, provided by the programmer, until the lower-level configurations are performed. Subsequently, the hierarchical organization of the logic elements yields several benefits for the pragmatic user of the device, which are discussed in following subsections.

2.2 Flexibility

Considering the case when the entire software has to be replaced, what happens first is that every logic block is cleared through specialized instructions on the FPGA firmware.

Also, certain procedures can be run to ensure that every logic block is unharmed. Afterwards the new instructions are loaded onto static memory units, awaiting the multiplexing process to take place. Each of these steps counts on the interconnections and routing logic which has to be correct and timely.

2.3 Hierarchical organization of logic elements

Making use of the spatial locality is very important to the power efficiency aspect of FGPAs. Having logically related blocks close to each other means that the most frequent signal transmissions take place in short wires, thus there is no frequent need for buffers and/or heavy routing to take place. Likewise, temporal locality reduces the need to re-drive signals in already occupied wires

2.4 Parallel computing

Parallel computing in FPGAs takes place whenever different blocks utilize their logic simultaneously. The main requirement is to have the blocks working independently, based on control signals, rather than having a central unit managing every aspect in the form of time-slicing etc. The need for this central control unit

2.5 Context of Internet of Things

FPGAs increase preprocessing. One of the main ideas discussed in our introduction to the IoT principles was the imminent need to have more data processing at the edge of a system [9], aiming at fog infrastructures. FPGAs thrive at this idea, since they appear to work at low abstraction levels, utilizing power efficiency and computational proficiency as soon as data is generated from sensor nodes.

FPGAs adjust data payloads through compression. IoT systems are heterogeneous by demand or by design. Either way, having devices working at different levels of data processing becomes a challenge in terms of data transmission. The transmission has to be timely and continuous, thus FPGAs are very useful since they can manage transmission bandwidth by altering the payload through data compression, at a reasonable amount of time.

FPGAs enable real-time communication. In order to achieve real-time acquisition and transmission of information, sensor networks must achieve very low routing times in which the data finds the path to the data gateways. Also, the aggregation of data must be timely and performed upon demand from the network with no latency. FPGAs' event-driven working principles apply here at best, providing very high responsiveness and fast routing.

FPGA's deterministic nature enables easy interfacing. The author in [10] claims that FPGAs' deterministic working principles make the device very suitable to interface with other devices. This implication is further enhanced by the author, who states that FPGAs can also be programmed to communicate through interfaces such as Bluetooth, allowing for mobility. Furthermore, in [10] the author expresses the ideas that FPGAs can alternate between on and off times and that FPGAs can work at different frequencies compared to other connected components.

3 Network considerations

As in most IoT systems, communication patterns pose strict requirements to the entire design. The communication must be performed in the most reliable way possible and must guarantee timely acquisition of data for all devices that participate in the same process. As mentioned earlier, the main challenge when trying to introduce a powerful processing device in a network comprising of different devices is to make these devices work together. Thus, in order to facilitate the integration of FPGAs in IoT the architecture of the system has to be considered from the perspective of the so-called Separation of Concerns principle. As a general thumb rule determined by IERC standards, every system which consists of heterogeneous devices must follow the principle [11] which itself is defined as follows:

` Separation of concerns is a design principle for separating parts of a system, such that each part addresses a specific issue with a high degree of autonomy in decision making`.

Technically wise, problems such as configurability, interfacing, down times and throughput rates need all be considered in order to have strict Separation of Concerns. Before doing so however, it is important to introduce the currently used IoT models of communication.

3.1 IoT model

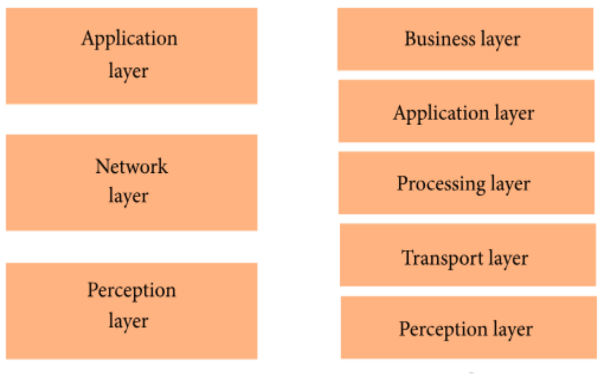

Figure 2. IoT network layer models

Figure 2 presents the most used models in IoT [2] infrastructures with regard to communication protocols.

The first type is a three-layer architecture, in which the Application Layer is concerned with delivering services to the user. Such an example would be a mobile application to control the blinds in a smart home environment. The network layer is concerned with the communication among devices which gather and process the information. The Perception layer is the layer in which data is gathered by physical devices. It can be observed that in such a separation of layers, it is unclear where the processing happens and where it should happen.

Considering the second architectural model with 5 layers, we can observe that the Application Layer is extended with more functionality, being the layer in which different types of Users can manage entire IoT systems as well as be at the receiving end of the processed information. Furthermore, the network layer is divided into three sub-layers.

The application layer is responsible for delivering services to actors in the business layer, based on the state of data processed by the lower levels in the architecture. This means that the application layer is not directly related to the user, instead it is related to management platforms in the Business layer.

The transport layer is concerned only with the communication patterns among devices residing in the perception layer. These patterns are similar, if not equivalent, to the well-known LAN, Bluetooth, RFID and Wi-Fi networks.

Lastly, the processing layer is the middleware of the architecture, in which the processing of data happens and also the storage capabilities are enhanced. Storage is usually achieved by means of cloud storages and processing is achieved by the utilization of general-purpose computers.

It is observable that in this architecture, the processing layer is the layer in which both storage and processing takes place. The ways how in particular are obscure and are usually left to the system designers to determine.

If we were to make use of FPGAs in this architecture, we would put them at the center of the processing layer, with limited communication functionality but with unlimited control over the physical layer. The reasons for this are highlighted below.

FPGAs are very efficient in the consumption of heavy processing tasks. As discussed in Section 2, FPGAs are built in a robust way, which makes them very power and time efficient in processing of information. This leads us to think of placing FPGAs in place of a general-purpose computer, to simply receive loads of sensed data and to apply some analytics and classification tasks on it.

FPGAs are not flexible enough to be used as simple sensing devices in the physical layers. Utilizing FPGAs to perform information sensing in a particular communication would be an inefficient way of using them, since we would be making use of only a portion of the device's capabilities in processing. Furthermore, FGPAs are programmed using low-level logic, making it difficult to achieve communication with non-proprietary networks such as the Internet.

FPGAs are flexible and efficient in controlling electronic and electro mechanic devices in the physical layer, to perform actuation. Using low-level logic, we would be able to customize the functionalities of a set of electronic equipment such as servomotors, thermal cameras etc. programming only the FPGA instead of configuring every device manually. The range of devices that FPGAs can support is about 15 devices per unit.

--

What can be gained from the discussion of the two models in tandem with the requirements of IoT systems is the fact that FPGAs are essentially a good fit for performing processing of data, possibly avoiding communication duties with heavy processing requirements. This conclusion might lead us to think of placing the FPGA at the center of the physical infrastructure, similar to how the CPU is placed at the center of a general-purpose computer. Doing so is actually conversed to the previously mentioned principle of Separation of Concerns. It is for this reason that this paper follows the principles of Fog Computing when introducing a potential design for the system.

3.2 Fog Computing

Fog Computing, also widely referred to as Edge Computing, is essentially the idea of having most of data processing functionalities placed closer to the physical layer of the system, meaning that devices autonomously gather and process the data before transmitting them to Cloud Storages or Gateways [12]. This means that the system structure becomes inherently more decentralized, as compared to other architectures, including the ones based on the two models discussed before.

The paradigm inherently envisions the usage of smarter electronics with enhanced processing capabilities, in order to ease the burden at the central parts of the network. In IoT, central parts of the network are usually devices acting as servers or gateways to servers which gather the information and bring it closer to the end user. This gives us some allowance of having FPGA central to a group of devices which gather information and in the same time having it not central to the overall process.

To make things clearer, if one of the two models discussed previously was to be used, the FGPA would take much more responsibilities with regard to communication as it would be the single point of interest in the network layer. If it were to fail, the entire network would fail and the end-user would not be able to access data. On the contrary, fog processing means that we can organize groups of devices in smaller units, each with some degree of autonomy. What makes this possible is that each device can now rely directly on the FGPA and its processing, without necessarily being interested in the larger network. This effectively shortens the communication distances and allows the end-user to access parts of the data in case of failure.

4 The proposed solution

The solution proposed in this paper for the design and implementation of the system follows the bottom-up approach. In previous sections, technical requirements were mentioned and two basic principles were outlined, namely *Fog Processing* and *Separation of Concerns.*

Following a bottom-up approach means that the first aspects to be considered are the ones primarily linked to the physical infrastructure of the system. Thus, the topology depicted in Figure 3 is proposed for the edge of the system.

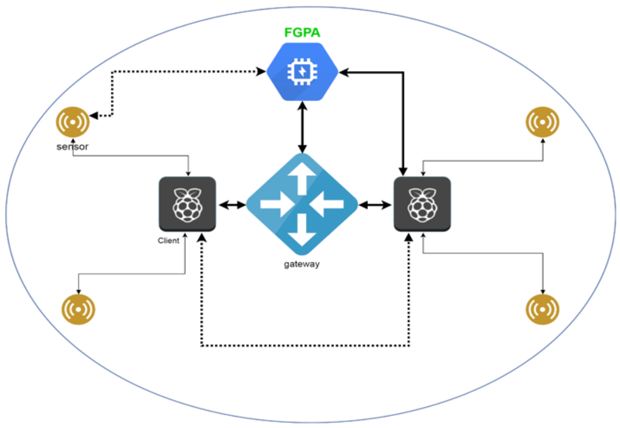

Figure 3. Cluster Topology

From the figure above, certain distinguishable elements such as sensors, gateway, clients and FGPA can be distinguished. These are the primary participants in the system but the cluster should not be restricted on the type of devices it can include.

The topology of this cluster is Star, with the gateway device serving as a central node. However, FGPAs can also be seen to communicate with the same number of device types as the gateway, implying that FPGAs are central to the topology as well. The difference is that from the viewpoint of several clusters being integrated into a mesh topology, FGPAs in one cluster are invisible to the ones in other clusters, whereas gateways are the only devices in a cluster that are made visible to other clusters. This separation of duties is a first step towards achieving a modular design in which the processing in each cluster is abstracted and made proprietary to the cluster, strictly following the fog processing paradigm.

Secondly, FPGAs should not present a single point of failure to the system. It is better to have sensors communicate with clients since the number of clients is usually higher than that of FPGAs in a cluster.

Thirdly, it becomes easier to disintegrate a cluster in case some normalization is required. That is, a cluster might need to be split into two different cluster for each cluster to target specific requirements. Doing so would require to have some of the sensors detached from the FGPA, instead of detaching only one of the clients responsible for a group of sensors.

Similar to the disintegration, whenever two clusters have to be merged, we can simply connect clients of one cluster to the FGPA of the other cluster.

The figure also depicts the gateway device, central to the topology, as being different from clients. This is not necessarily the case, since clients with transmitting capabilities can undertake gateway procedures. Actually, in this design, the gateway can be thought of as a client which is responsible for intra-cluster communication but which is not connected to any sensor whatsoever. As a matter of fact, gateways need to remain as lightweight as possible in terms of processing.

4.1 The Gateway

The cluster is the basic unit of organization in our design and as such, it must be interfaceable with other clusters in order to achieve a functional communication pattern overall. In order to maintain a way of communicating even as devices in the cluster are added or replaced, the interface needs to be general purpose and it is the specific duty of other client devices to adjust to the interface. In terms of physical devices, the gateway can represent an interface for all other clients to the outside world. The question remains, how to achieve the design of such a gateway?

First off, the cluster-based design would require 'client nodes' to perform the first aggregation of data as presented by the sensors. To achieve this, it would suffice to have each wired sensor directly connected to one of the GPIO pins of a device with Internet Access. Also, the device at hand would have to be programmed to read the incoming signals and to store the data in its local storage. This would be the case of devices such as RaspberryPi.

Secondly, the client node would have to perform some lightweight processing. The processing at this level basically consists of some manual filtering of the input to meet timing requirements and some calculations to make the data more suitable for the next level of processing. Provided that the processing is basic and supposedly does not add overhead, the readings have to be performed as frequently as possible to maintain a high input. On the contrary, estimations imply that most wired sensors require readings to be performed in fixed periods of 0.7-2.4 seconds, thus the design has to ensure that the duty cycles of each sensor are respected. Having to settle on a tradeoff, two solutions seem feasible; either each raspberry needs its sensor network to be as homogenous as possible or the raspberry needs to perform some initialization procedure just like the drivers that one needs to install to their device to use some peripheral.

Thirdly, the connectivity of the raspberry presents a challenging issue by default. What makes it challenging is the fact that the device can lose its connection to the WAN in case some changes occur to the LAN. As an example, let's assume that the client node serving as gateway is connected to the Internet through Wi-Fi offered by a nearby access point. In case the administrator of that LAN changes the password, the client node would not be able to reconnect as it does not know the new password.

A feasible but expensive solution would be to equip the device with LTE connectivity. Another time-expensive solution would be to reconfigure the client node each time such problems happen. In this design, the problem is again solved by deciding on a tradeoff where the gateway device is configured to functionalities such as encoding, decoding, generating tokenized keys etc.

4.2 The FPGA node

The FPGA node in our design is pivotal to the edge processing duties. As such, it can be considered all on its own as a system with the input, output and performance considerations akin to the considerations made on the entire system.

As a first step, we provide an abstraction on the way the processing tasks can be handled by virtual entities that we call *workers* [13] . A worker can be thought of as an entity which is assigned with only one task and which has no knowledge of the existence of other workers except for one. This one worker is a master worker that is assigned with the task of managing all other workers, called slave workers. It is clear that we have a Master-Slave relationship in which a master commands many slaves and the latter are not aware of any of their peers and the former synchronizes the work of each slave.

The reason why we isolate slaves from one another is because they should not be able to interfere with the duties of one another. What enables to do so is the fact that the slaves do not share resources and are guaranteed to have all the resources they need from the master. Thus, in our design we have to make sure that:

- Slaves are totally isolated
- Master worker can access all resources
- Master works at a predefined rate whereas slaves work at a rate defined by the Master

As a second step, we map the Master-Slave abstraction to a possible implementation in which FPGA processes work through components. The components can be organized in hierarchy with one component synchronizing other components. Also, by convention the components can be restrained from accessing other components. To make things clearer, a component is simply a module which implements several functionalities closely related to one another. For instance, we can have a component which performs the encryption of the data, performing the encryption of any incoming bitstream independently.

As a third and final step, the bottom-level components can be synchronized by the master, to work in parallel. In an event-based manner, the top-level component simply detects the events and spawns the right bottom-level components. From this point on, each component works independently and is responsible for other events and for the resources it can access. Considering the encryption component once again, the data that has to be encrypted can be found as a resource in a shared memory between the parent/master component and this component. Once the encryption is done, the results is stored in the shared memory together with a flag that signals the end of the task by the slave. The master can then communicate the results to the gateway client serially.

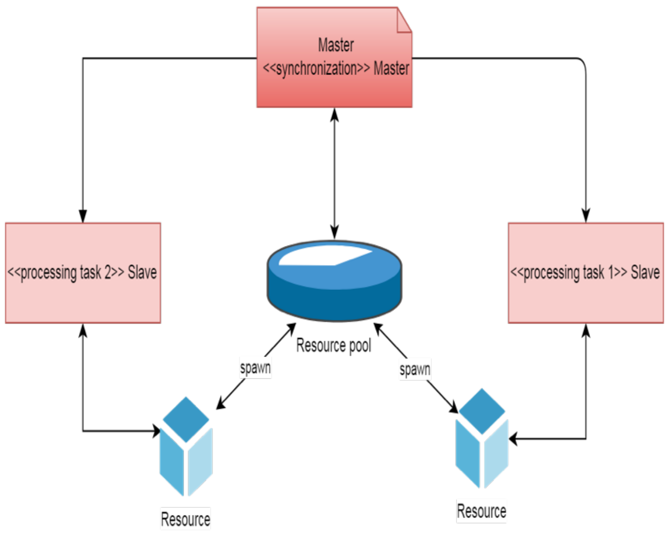
Master
<<synchronization>> Master
<<processing task 2>> Slave
<<processing task 1>> Slave
Resource pool
spawn
spawn
Resource
Resource

Figure 4.Master-slave workers

In our considerations thus far, three types of devices have been introduced, sensors, client nodes and FPGA nodes. Whereas sensors are practically interfaceable with both client and FPGA nodes, the communication between the client and the FPGA is the very heterogenous group in the system. In this design we offer a solution for the communication among the two. Solutions for more than two different types of devices should follow the same pattern.

The two devices can be physically connected through serial interfaces. The most commonly used interface is the UART (Universal Asynchronous Receiver Transmitter), for which USB to USB or USB to RS232 cables are required. The UART interface in FGPA follows a master-slave design similar to the one discussed previously.

In order to facilitate instant connectivity between a newly introduced client node to the FGPA unit of the cluster, the client node should be programmed such that it introduces itself to the FGPA upon first connection by providing its MAC address and the type of processing that must be performed. From the FGPAs perspective, it is important to automatically instantiate worker components from the event of connectivity.

This means that the master component of the FGPA should be able to bootstrap a slave component which performs the necessary tasks such as creating the resources needed and identifying the processing tasks that should be performed on the data provided by the connected Raspberry.

In case of more than two types of devices, the interfacing to the FPGA should follow the same pattern, since virtually all IoT elements have Serial interfaces.

4.3 Mesh design

Following suit on the bottom-up approach taken in this design, we now consider the design of the system as a set of an undefined number of clusters which are interoperable with each other. The main challenges faced are the dynamic nature of the system in which cluster can be dynamically allocated to the mesh, the inter-cluster communication and the mesh-cloud communication.

Based on the discussion of Cluster design, principles of discoverability and recovery for entire clusters follow similar patterns to those of simple devices in the role of client nodes. The only difference is the fact that among clusters we can achieve a more organized discovery and recovery.

Inter-cluster communication is performed through the Publisher – Subscriber pattern, but only logically so. Physically the communication is performed through Internet Protocols such as HTTP in the form of remote procedure calls (RPCs). The need for RPC is not evident in this work and remains a personal choice.

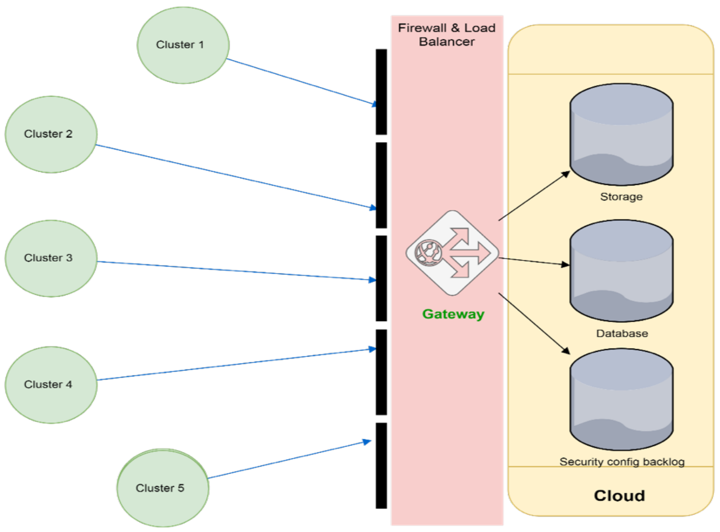

Figure 5.Mesh topology

Several cluster nodes can be seen in figure 5. They are directly linked to a long vertical rectangle, indicating the requests sent by each cluster to a unit which performs load balancing but also guards the gateway from malicious requests. It can also be seen that the gateway stands in the center of this unit.

Furthermore, the gateway sends its own requests to the Cloud unit on the far right, organized into Storage, Database and Security Configurations Backlog which is typically the organization of a PaaS. The gateway unit in this design is meant to be a simple server which filters requests and provides an API for any cluster. It should be noted that at this level, the cluster-gateway communication is performed through REST API calls that reflect the states of the devices in any of the clusters.

4.4 Enhanced Inter-cluster communication

In order to perform cluster-to-cluster communication, the topology must be enhanced such that it allows clusters to send and receive message to one another. The publisher subscriber schema was discussed briefly, introducing the idea of an event store in which a set of predefined event types are maintained and devices can subscribe or publish events of those predefined types. In this section, the publisher subscriber schema is included in the design in the form of the previously discussed RPC [14]. It should be noted that the participants in this communication pattern are exclusively the gateways of clusters and not devices of other types.

4.5 Failure recovery

In order to understand possible ways of performing recovery of a lost connection for an entire cluster, we should first discuss the reasons why such failures can occur.

The first and most probable reason is the physical failure of the gateway device in the cluster. Such an event would be recognized by participants in the cluster as explained and another device in the cluster would undertake the duties of the Gateway.

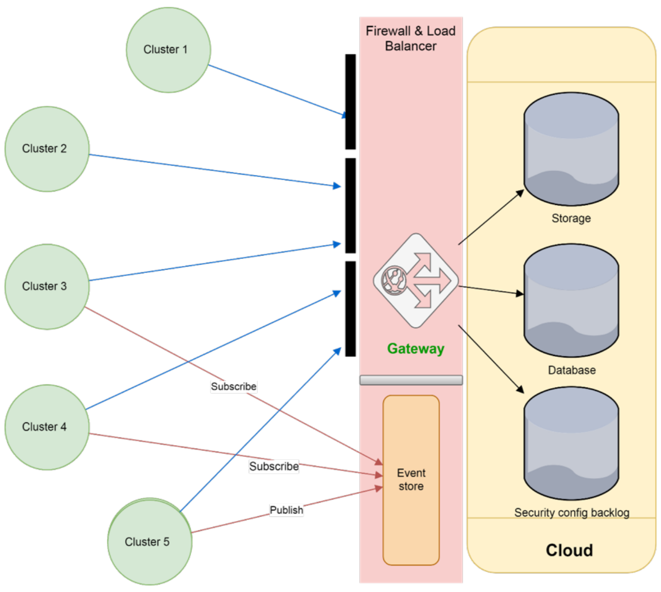

Figure 6.Enhanced mesh topology

The second reason is the logical failure of the gateway device because of some connectivity issue. This would possibly not be detected by participants in the cluster. Unfortunately, in this design no exact solution is provided for this problem except for the manual detection based on the lack of frequent responses. Usually a monitoring of the clusters is performed in similar systems, something out of the scope of this design. Assuming recovery from a logical failure, the cluster gateway should invoke some correction procedure in the gateway immediately to perform authentication for the cluster and to maintain consistency. In addition, it should update its state of data from all subscribed-to events.

5 Conclusions

In this paper, we have introduced the means of designing an IoT system based on a large-scale integration of both hardware and software technologies. Our main focus has been the designing and implementing of the system compliant to the edge processing paradigm by utilizing FPGAs.

Prior to discussing the technical details of the system, a survey of technological approaches used was provided, highlighting requirements and practices that are prominent in the world of IoT. The potential benefits presented by FPGA devices such as parallelism, lowered power consumption and computational proficiency have been put in the context of IoT. Provided that context, the presence of different types of devices has been included in the design and implementation, yielding different outcomes and requiring solutions.

Furthermore, the system design followed a bottom-up approach. The bottom-level of the proposed infrastructure was based on a star topology in which different types of devices integrate with one another on a physical level. The top-level of the infrastructure was based on a mesh-based model in which clusters communicate with each other through the internet. The upper-most layer was also introduced in the first section of this work as Cloud PaaS.

Implicitly relying on a cloud system eased both the design and implementation phases as it allowed for more work to be done at the edge of the system. Based on the implications derived throughout the survey, design and implementation, several conclusions have been achieved.

The first conclusion is that FPGAs offer flexibility in designing a hardware-based system in which processing power is required.

The second conclusion is that FPGA based designs are constrained by the difficulty in interfacing other devices to it.

The third conclusion is that on a comparative scale, FGPA based designs in IoT present several benefits such as reduction in power consumption, reduced number of devices, support for heterogeneity, increased efficiency and computationally proficiency.

The fourth conclusion is that the implementation of a system as proposed is difficult given the complex nature of FPGAs.

The fifth and last conclusion is that following the edge processing paradigm is necessary for ensuring scalability, and FPGAs provide the means of achieving full edge-processing through parallelism and automation.

References

1. K. Ashton, "That 'Internet of Things' Thing," *RFID Journal,* Jun 22,2009.
2. T. S. S. O. ITU, *GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS AND NEXT-GENERATION NETWORKS Overview of the Internet of things,* ITU-T, 06/2012.
3. F. F. G.-N. ,. B. L. F.-R. ,. L. B. a. M. A. A.-V. Jorge E. Ibarra-Esquer, "Tracking the Evolution of the Internet of Things Concept Across Different Application Domains," MDPI, Basel, Switzerland, 2017.
4. O. Friess, *Internet of Things From Research and Innovation to Market Deployment,* River Publisher Series in Communication, 2014.

5 T.Brand, *Connected Living- The next wave of mobile devices,* GSMA, 2012.

6 R. G. S. Tiwari, *A Review Paper on Home Automation System Based on Internet Of Things Technology,* International Research Journal of Engineering and Technology(IRJET), 2016.

7 K. Ashton, "The Internet Of Things," Korea Economic Daily's "Strong Korea", Seoul, June 19,2014.

8 G. &. R. S. Krishna, "Fundamentals of FPGA Architecture," 2017.

9 V. R. V. V. Dr. K.K.Sharma, "A Review of FGPA implementation of Internet Of Things," *ISSN 2229-5518,* vol. Volume 8, no. Issue 4, 4 April 2017.

10 P. Sarangi, *Internet of Things Architecture,Protocols and Applications,* Journal of Electrical and Computer Engineering, 2017.

11 F. P. W. J. T. V. Bart De Win, *On the importance of the separation-of-concerns principle in secure software engineering,* Leuven: Katholieke Universteit Leuven, Dept. of Computer Science, September 30, 2002.

12 Z. C. A. R. A. R. K. Firas Al-Doghman, "A review on Fog Computing technology," in *2016 IEEE International Conference on Systems, Man and Cybernetics*, Budapest, Hungary, 2016.

13 G. V. Sartaj Sahni, *The Master-Slave Paradigm in Parallel Computer and Industrial Settings,* Gainesville,Florida: Department of Computer and Information Sciences, University of Florida.

14 P.-M. P.Th Eugster, *The many faces of Publish/Subscribe,* Lausanne,Switzerland: Swiss Federal Institute of Technology,.