USING CLOUDS FOR FPGA DEVELOPMENT – A COMMERCIAL PERSPECTIVE

Laurențiu A. Dumitru [1*]
Sergiu Eftimie [2]
Ciprian Răcuciu [3]

ABSTRACT

Field Programmable Gate Arrays (FPGA) are electronic devices that can be reconfigured at runtime. Due to the fact that they implement a small number of dedicated functions, FPGAs are used for hardware acceleration, alongside with general purpose processors. Several vendors provide different Integrated Development Environments, but all of them support the standard VHDL and Verilog hardware description languages. After the development phase, implementing an FPGA design can be a time-consuming and cpu-intensive task. The current paper examines existing technical solutions that provide build parallelism at high speeds, as opposed to workstation-local building, and tries to estimate at what point migrating towards a third party justifies the costs.

KEYWORDS: *FPGA development, FPGA synthesis in clouds*

1. INTRODUCTION

Field Programmable Gate Arrays are now used in more and more environments, due to their versatility and performance. From real-time tasks such as feedback and control in automotive and aviation applications to more general-purpose such as enabling IOT connectivity, FPGAs bridge the gap between flexibility and hardware-implemented algorithms. Traditional software, executed on a normal microprocessor, runs sequentially, while algorithms executed on FPGA hardware run in parallel. Furthermore, FPGAs interact directly with the external environment through input/output pins of various physical characteristic. This capability makes them perfect for multiple sensor acquisition at very high speeds.

In normal computing systems, FPGAs can be found in many places such as North and South Bridges, network cards or dedicated accelerators. Since FPGA chips can be reprogrammed as the need occurs, many vendors have chosen to use FPGA over traditional ASICs (Application Specific Integrated Circuit) due to easier firmware

[1]* corresponding author, PhD. Cand., Military Technical Academy, 39-49 George Coșbuc Bvd., Bucharest, Romania, dlaur@nipne.ro

[2] PhD. Cand., Military Technical Academy, 39-49 George Coșbuc Bvd., Bucharest, Romania, sergiu.eftimie@gmail.com

[3] Univ. Prof. PhD., Military Technical Academy, 39-49 George Coșbuc Bvd., Bucharest, Romania, ciprian.racuciu@gmail.com

upgrade and lower time-to-market. Evolution of fabrication techniques has permitted higher operating frequencies that were not available in the past. The 16nm fabrication process and technologies such as 3D IC offer higher interconnection speeds, larger reprogrammable areas and lower power consumption.

System-On-A-Chip architectures usually combine a normal microprocessor, such as ARM Cortex, with reprogrammable logic, such as an FPGA chip. The hardware microprocessor runs at a higher frequency than the FPGA, thus is capable of running modern operating systems without sacrificing performance, while the FPGA has the flexibility to implement various interfaces for outside communication. Such an example is the Xilinx Zynq product family. SoC designs are found in many consumer electronics that are IOT-capable.

Modern FPGA families have multiple capabilities such as radiation tolerance, integrity checking, error correction and partial reconfiguration. Such abilities make them suitable for mission-critical environments the error toleration rate is extremely low. In the past, FPGAs did not provide any technical solutions for upgrading a device without rebooting it but now, partial reconfiguration can be used for in-field software upgrades without the need of restarting the device. Such a feature can prove to be very useful in set-ups where high availability is needed. Multi-gigabit transceivers that can connect to many mediums have boosted the use of FPGAs in consumer-grade devices. The development for FPGA is usually done in a Hardware Definition Language (HDL), such as VHDL or Verilog, with the help of an Integrated Development Environment provided by the chip's vendor.

Most of these IDE's provide Software Development Kits that ease the deployment of software stacks on top of hardware or software-implemented processors. The development process of an embedded system can be split in hardware development and software development. Hardware development refers to the architecture implemented in the FPGA chip, not including the electronic part, but including any HDL-related code, while software development refers to applications written in high-level languages that are to be executed on the microprocessors available on the hardware design.

The software development of an embedded system usually refers to the programming the soft-core or physical microprocessor units installed. It can be done in ultra-low level languages, such as assembly, or with more developer-friendly alternatives such as C/C++. There are not so many IDEs that support just-in-time languages due to the necessity of an underlying translation unit, which, on embedded systems, can occupy resources and consume more power without giving back any benefits. For solutions that support multiple microprocessor architectures a programming language that can easily be ported to many targets is wanted. During the development phase, virtual environments that simulate the target architecture can be used to validate a specific code, before trying it on the development boards.

The development and testing of the hardware architecture is a process that can be time-consuming. The time interval of generating the final product of the design, the bit stream that is to be programmed in hardware, varies with the size of the FPGA, the complexity of the HDL code and the speed of the system on which the process is taking place. Functional validation is done by simulating the design, but, in many cases, actual validation can take place only when the hardware is programmed, probes are placed in specific parts and physical test benches evaluate the overall behaviour. Such an example is the development of a system that has PCIe connectivity. It can be fully tested only when the card is placed in an actual system, the drivers are active and data flow can occur. If the desired behaviour is not achieved, an error occurs or it just does not work, the system engineer must start the whole process all over again. Repeating this process can be painful given the fact that it can take from a few tens of minutes to tens of hours. If a company is only occasionally building such systems, it can be assumed that at best only a few workstations are available for the development. It is obvious how these high implementation times can affect a specific time frame inside a project's workflow.

While there are methods of speeding up the implementation, most of them come at a cost. Companies that have FPGA as a primary market have their own dedicated computing farms for such situation. Other companies, that do from time to time FPGA development, cannot justify the implementation costs of high performance systems. Alternatives exist for these companies. This paper analyses how a FPGA implementation processes can benefit from computing clouds, what are the technical requirements and what costs can be expected. The proposed technical approach is not the only one that could be implemented.

2. RELATED WORK

Since FPGA development is still considered inaccessible to the average consumer-oriented company due to high man-power costs, high technical costs and increased technical complexity there are not many viable remote building/compiling solutions. However, large companies such as Intel and Microsoft have already massively deployed FPGAs in their cloud products. When FPGA will be endorsing more and more technologies, more alternatives will be added to the few build options available to the current design flow.

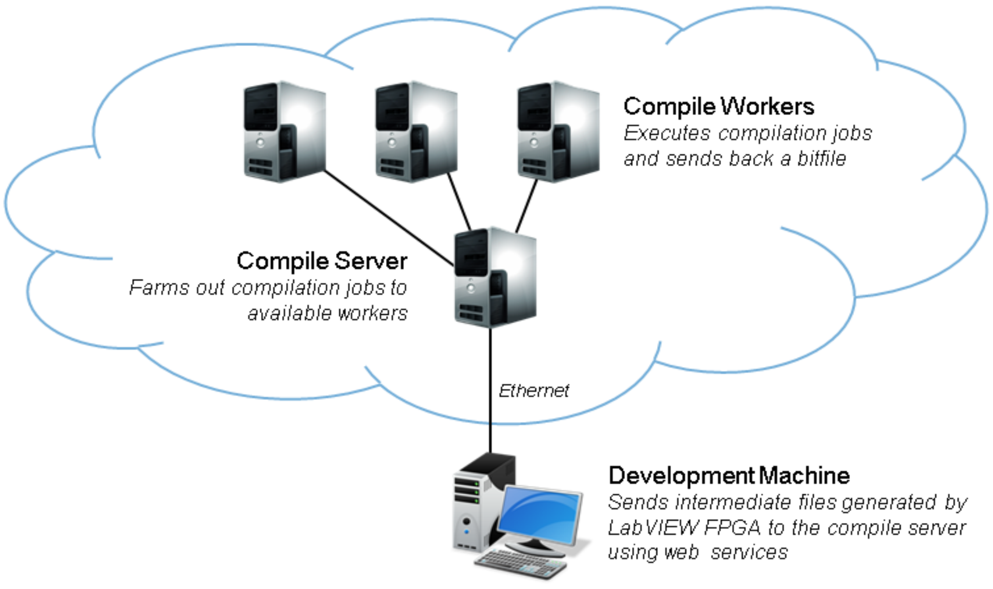
Compile Workers
Executes compilation jobs and sends back a bitfile
Compile Server
Farms out compilation jobs to available workers
Ethernet
Development Machine
Sends intermediate files generated by LabVIEW FPGA to the compile server using web services

Figure 1. NI Compile Cloud Architecture

LabVIEW offers the “NI LabVIEW FPGA Compile Cloud Service” for its clients. It is described on their whitepaper as providing shorter compile times enabled by high-performance Linux-based servers in the cloud, improved productivity by performing multiple compiles at a time, in parallel, and the convenience of being able to power-down your PC at any time during a compile. In [Fig. 1] we can observe the architecture, based on a client-service model, as described by the vendor.

Such a solution is, unfortunately, only available for customers that use their technologies. However, based on the architectural model, one can envision a general approach of the same workflow.

Trying to address the same gap, “Plunify”, a Singapore-based company founded in 2009, develops a cloud platform that enables semiconductor chip designers to shorten product time to market and reduce development costs. Plunify offers its products as add-ons to existing IDEs provided by various FPGA vendors. InTime, the product that addresses optimization products and timing closures by means of machine learning and raw computing power, supports Xilinx's ISE and Vivado and Altera's Quartus. It has various run targets: local systems, private clouds and public clouds. Since its launch, it has developed several interfaces that allow a developer to run multiple scenarios in parallel, manage computing resource, do scheduling and various other tasks. Regression testing, design optimization and resource management are also available.

During the process of developing an FPGA design, a series of compilation cycles are needed. Plunify also offers tools for regression testing and benchmarking that are highly important in these cases and help to analyse if a feature has been affected or broken due to design changes. Automation also plays an important role in the whole process because it can identify flaws with minimal effort in case of last minute changes, for example.

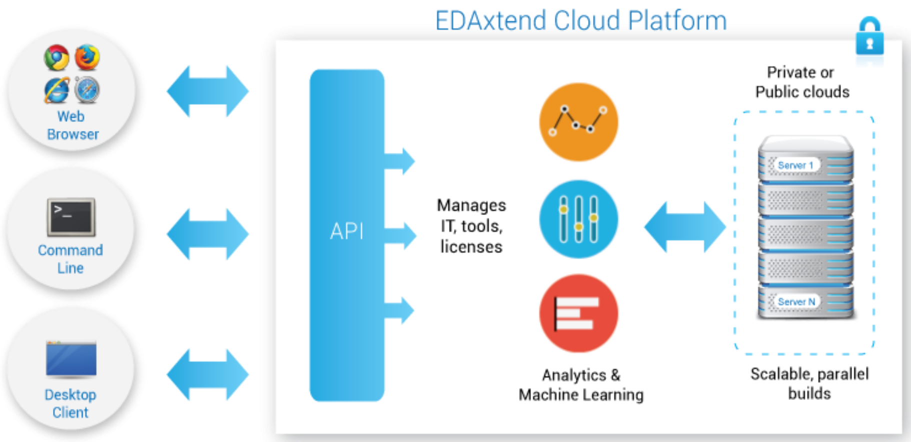

Figure 2. PLUNIFY EDAxtend Cloud Platform

EDAxtend is Plunify's cloud platform that can run in public and private clouds and uses the existing design tools so that engineering teams can, without having to learn new methodologies, harvest the power or large computing farms. API and script-based access is available so that both interactive and automated build scenarios are supported. Communication to and from the cloud platform is secured with VPNs, SSL/TLS and other techniques.

For large development teams and multiple projects, PLUNIFY's platform and tools are a major improvement from having to manage such in-house resources. For others, the TCO might still not justify the use of such a platform.

Standard cloud services, where one can run a Virtual Private Server, can be used to overcome the limitations of a small number of workstations. The following section proposes such an approach.

Amazon has also launched its FPGA service in 2016 called F1 [Fig. 3] that uses field-programmable gate arrays. The new instances are planned to become generally available during 2017. The company motivates their service offering in the increasing affordability of the FPGAs and the fact that they have become easier to program, opening the way to their use into a wide area of services. The increase availability in the cloud is believed to motivate the developers to start experimenting with them.

4K video processing and imaging, as well as machine learning are considered suitable candidates for FPGA development.

NGCodec is a company that worked with Amazon in order to test the new F1 instances. NGCodec implemented its product called RealityCodec for VR/AR processing using F1 instances within a month of development. NGCodec estimates that such an implementation could allow the run of a complex video processing needed to run a virtual reality device using a head-mounted display in the cloud. FPGAs have an important advantage over GPUs because the encoding involves processing that GPUs normally transfer it to the CPU. FPGAs are also more power efficient in this of scenario.

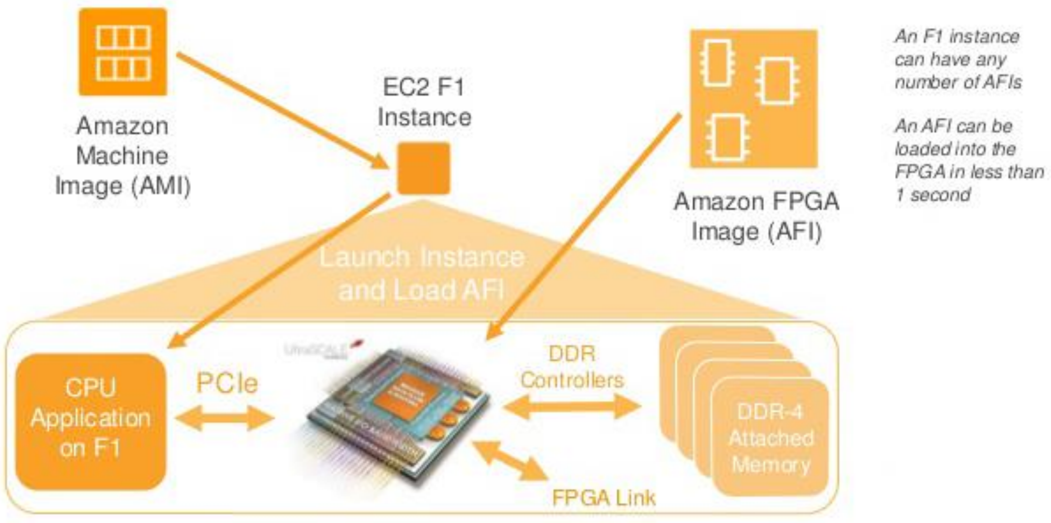
Amazon Machine Image (AMI)
EC2 F1 Instance
Amazon FPGA Image (AFI)
An F1 instance can have any number of AFIs
An AFI can be loaded into the FPGA in less than 1 second
Launch Instance and Load AFI
CPU Application on F1
PCIe
DDR Controllers
DDR-4 Attached Memory
FPGA Link

Figure 3. FPGA acceleration using an F1 instance

Amazon has a partnership with Xilinx, one of the major FPGA manufacturers. This is a list of specifications for F1 instances:

- Xilinx UltraScale+ VU9P fabricated using a 16 nm process.
- 64 GiB of ECC-protected memory on a 288-bit wide bus (four DDR4 channels).
- Dedicated PCIe x16 interface to the CPU.
- 2.5 million logic elements.
- 6,800 Digital Signal Processing (DSP) engines.
- Virtual JTAG interface for debugging.

Despite these advantages, FPGA programming remains a hard discipline. Amazon has announced that it won’t release tools for FPGA development (Xilinx will cover this aspect) but instead it will focus on the cloud side where it will release development kits and a machine image that the developers can use to get started with the F1 instances.

3. THE PROPOSED APPROACH

Infrastructure as a Service (IaaS) is a form of cloud computing that provides virtualized computing resources over the Internet. IaaS is one of three main categories of cloud computing services, alongside Software as a Service (SaaS) and Platform as a Service (PaaS). Amazon Web Services, Microsoft, Google or Rackspace can be found amongst the main companies that provide Infrastructure as a Service business plans. IaaS is suitable for a number of situations where demand on the infrastructure is volatile or where new companies do not possess the capital to invest in hardware. Both scaling and temporary needs for hardware are covered by IaaS. Cloud providers supply the resources in an on-demand manner from their pools of resources located in data centres. In our proposed approach the cloud provider should also invest in a pool of FPGA chips linked to an existing computing infrastructure [Fig. 4] in order to be able to provide a testing infrastructure to their clients. Among the advantages of using a IaaS are the rapid innovation due to the readiness of the infrastructure when needed and the focus on the core business, in our case, the FPGA development. The payment model eliminates the expenses involved in deploying on-site software and hardware. Despite this, users should monitor their IaaS console in order to avoid being charged for unauthorized malicious access.

Due to the fact that cloud providers own the IaaS infrastructure, the monitoring and the management of the systems may become difficult for users. Also, if an IaaS provider experiences downtime, users' workloads may be affected.

Cost-effectiveness may arise in specific scenarios. For example, once a certain software is tested, it can be moved from the IaaS environment to a proprietary infrastructure in order to free the resources for other development projects. Cloud computing uses automation in order to meet the unpredictable requirements of the users. Cloud software automates the provisioning and the scaling of the computing resources, storage and network. In our approach, we envision the development of an interface that would simplify the entire provisioning system through process templates used in the workflow, in order to simplify the provisioning activities.

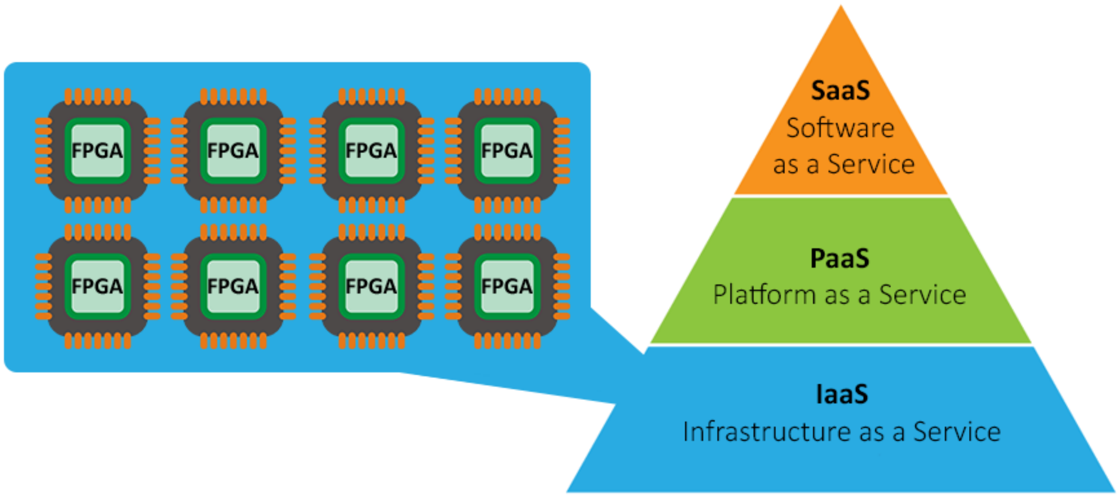

Figure 4. Hybrid IaaS-FPGA

In the following section, the “Consumer” is any company that will, at some point in a certain project, need to implement FPGA functionality. The “Provider” is a company that exposes cloud-based FPGA build solutions, with a different approach that those presented in chapter 2, “Related work”.

For exemplification purposes, the Consumer develops an FPGA-based PCIe communication card that implements various encryption algorithms for secured point-to-point communication. Apart from the technical design, the FPGA must implement the following components: a soft-core processor, Ethernet over Fibre Optics (SFP), PCIe communication core, Direct Memory Access, Reconfigurable encryption modules, Timers and other auxiliary components. Implementation times, as evaluated on a standard workstation with 8GB RAM and an Intel Core i7 @ 2.6 GHz, 3720, 4 cores and 8 Threads:

Table 1. Time comparison

Step	**Synthesis**	**Place and route**	**Bit stream generation**
Minimum time	20 min	30 min	2 min
Maximum time	~230 min	~300 min	~3 min

The times were recorded using various configurations, versions of the IP cores and the presence/absence of custom IP cores that were implemented in the project. Project design and implementation was done on a Xilinx Kintex 7 FPGA chip and Vivado IDE as development environment. Xilinx Kintex 7 has the best price-performance ratio on the market with 478k logic cells, VCXO component, AXI IP and AMS integration. The FPGA chip has also 32 × 12.5G GTs, 2,845 GMACs, 34Mb BRAM and DDR3-1866. It can be purchased at half the price of similar 40nm devices and utilises half the power used by the previous generation.

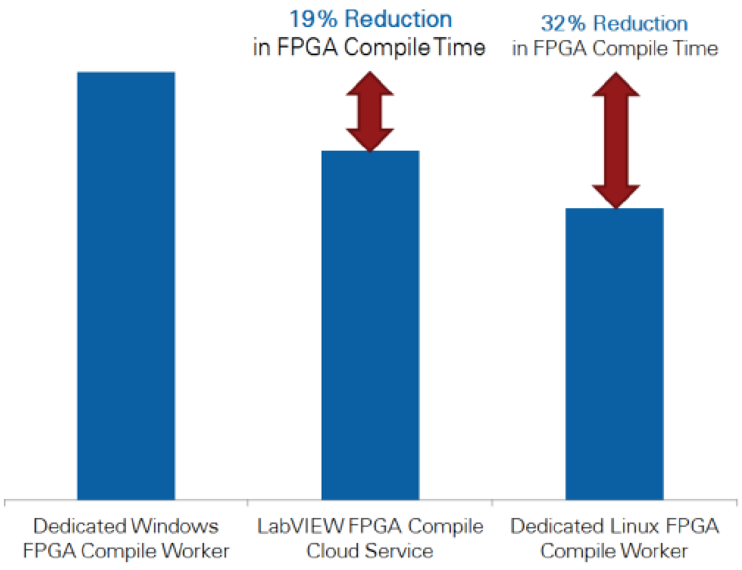

Figure 5. Compile time comparison

Such a FPGA design can have a lot of trial-and-error steps due to the fact that there are many components, apart from the actual FPGA architecture, that need to be interconnected – kernel module, electronics, user space applications. If, on such a small/mid-sized system, every small modification on the FPGA can last up to 9 hours of implementation, on larger chips, such as Xilinx's UltraScale Kintex and Virtex, the required completion time can be a lot larger than 9 hours. When doing compatibility and regression testing, multiple configurations of the same architecture are a requirement. All tests were done on Linux. Different kernel versions did not impact the overall implementation time. According to [2], Linux workstation perform better than their Windows counterparts. The reduction time is exemplified in Fig.5.

At this point, the management team is faced with multiple options: accepting the large implementation times, if they can meet the propose time-to-market criteria, invest in hardware in order to decrease the overall testing and validation time or use resources from a third party. If there are a small number of projects which will benefit from hardware investments, chances are that such an approach would be a poorer option over the third one – renting from a dedicated provider. An evaluation of a common investment for a 10x speedup, based on the above tests:

Table 2. License cost

Component	**Estimated cost**
10 × Workstations	10 × 1000 USD = 10k USD
10 × IDE Licenses	10 × 1800 USD = 18k USD
Administration and human resource costs (6-month project span)	6 × 800 USD = 4.8k USD

All costs were estimated at official “store” price list available online for an average configuration. No particular vendor or technology was targeted. It is clear that for an average 6-month FPGA step, the TCO can be very high. If the resource demand is higher, some companies may take into account private clouds, but with more costs [6]. If a company has a strict timeline and a tight-budget it is obvious that FPGA development is not an option.

A “Provider” would be any company that is willing to invest in hardware resources and software licences in order to provide a pay-per-use service. The service model is implemented in various Software-as-a-service and Infrastructure-as-a-Service setup [4]. The initial investment is larger than in the case of a single company, but the Provider would pursue a larger time frame for ROI, as opposed to the TCO of a single company. It is the provider's goal to approach companies that would like to develop FPGA architectures in-house for their project but do not have a constant flow of such projects.

From a technical point of view, the Provider would use a cloud solution such as CloudStack or OpenStack, fully automated, with resource management and dynamic control as in [8]. For each client that starts a project, a number of Virtual Private Servers would be started, with reserved resources according to the payment plan. If a 60 month TCO is planned, an expected 50 clients / year and a maximum number of 10 simultaneous clients, the investment plan for processing power would be:

- 400 cpu cores (4 cores / client × 10 clients × 10x speedup (parallel), as above)

This can be summed up as ~18 servers (dual processor 12 core = 6 clients, 64GB RAM) that price at around 2300USD. Monthly datacentre costs and administration can be around 1500USD, with a total of 90k USD for 60 months. The estimated TCO for 60 months, not accounting for unexpected situations, would be:

- 18 × 2300 + 1500 × 60 = 131k USD

The average market price per hour for 4 vCPUs with 16GB RAM is variable [5], but in the current year is, is around $0.23. For a project with ~500 runs of 16 hours/run of compile times, this would be around 1800$. It is clear that this pay per use model is much more efficient for any small company than having to invest in its own infrastructure. As for the Provider's TCO, if the targeted 50 clients per year are achieved, it can change from

ROI to Profit as early as the second or third year of the project. However, there are cases in which one may not want to expose private code to a third party, but, in such a situation the overall cost of the project should account for this situation.

There are other side-costs such as custom application development which must be accounted for. Nevertheless, these are one-time only and do not have such a high price than the infrastructure and running costs.

From a client's point of view, the whole process can be summarized as in [Fig. 6]:

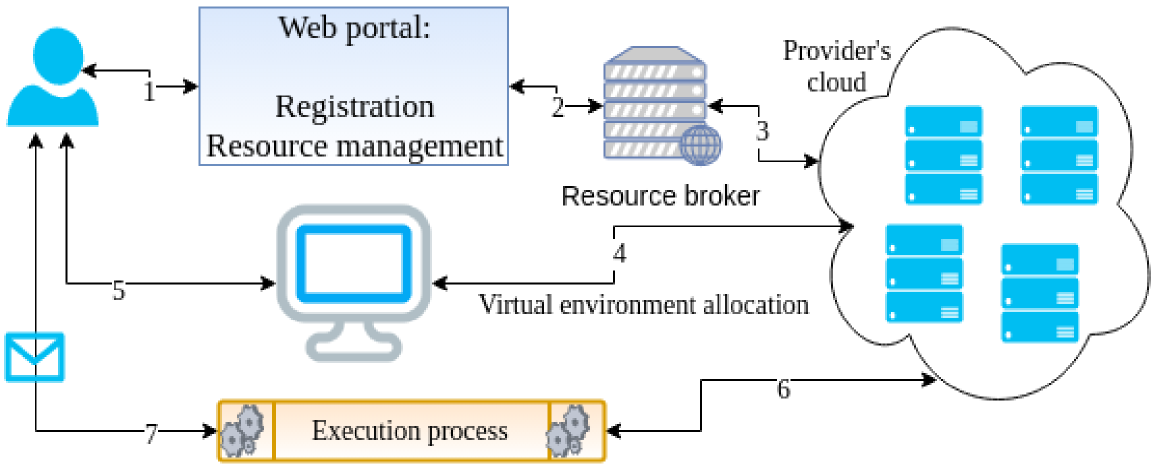

Figure 6. Process overview

The client manages its project through a web application. From here he can control assigned resources and keep costs under observation. After a project is defined, he will submit it for execution (2). The cloud resource broker would have already reserved the required servers (3) and starts up the execution. There are two different stages in the actual flow: an interactive one (4) and a batch/compile one (6). The interactive step forwards a virtual desktop (5) to the client in which he can do FPGA development in a desired, IDE, as chosen from the project's settings. Such interactive approaches of interface forwarding are already in use – example [7]. At this point, it is clear that the client does not have any software license cost. After he finishes the design, he will launch the project for synthesis, place and route and bit stream generation inside the cloud. This step is done in background (6). After the completion of the process, successful or not, the client will be notified. He then will log back into its account, access the allocated virtual environment and continue as required. All communication between the client and the provider are secured. The generated bit stream can be seamlessly integrated into the client's workspace by means of a VPN or other cloud transport method, such as the one proposed in [3]. Several authentication schemes can be implemented, based on the client’s needs.

From an economic point of view, the proposed workflow would require custom software components that need to be developed exclusively for this kind of project. The communication part (VPN, dedicated tunnels, a.s.o.) can be achieved by open source software, such as OpenVPN, or by using dedicated network hardware - such as Cisco’s

ASA platform. The virtual machine underlying infrastructure, with the necessary tools to managed hardware and software resources can also be implemented by the use of open source projects as CloudStack or OpenStack. The web portal, however, would have to be custom build for such a setup. A basic starting point for the development costs could be summarized as:

Table 3. Development costs

Project step / team	**Necessary team**	**Minimum time**
Requirement analysis	Project architect, Lead programmers, Team leaders	3 months
Frontend development	Graphics designer, User experience designer, 2 Front-end developers, Lead programmer	4 month
Backend development	Team leader, 3 to 5 software engineers, Lead programmer	5 months
Database design	Database design architect Team leader, Lead programmers	2 months
Platform integration	Linux system administrator, Lead programmer	2 months
Validation and testing	3 to 5 Quality assurance operators	4 months
Reporting	Lead programmer, reporting team (2 to 4)	3 months

4. RUNNING TESTS AT THE REMOTE SITE

In case of parallel building of multiple hardware modules, testing and validating them at the provider’s site could prove to be more efficient than retrieving bit-streams from each generated module and testing them one at a time. This could be the case when the client has only a few development FPGA boards but has many module versions. In such situations, the provider might offer an automated testing and validation service [Fig.7]

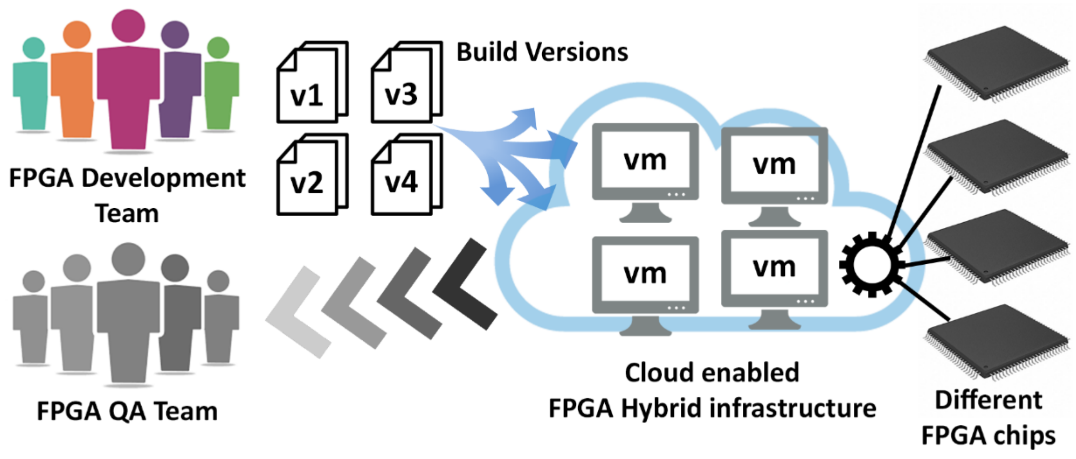

Figure 7. Testing infrastructure for FPGA cloud

If the provider chooses to implement hardware validation at a customer's request, it is obvious that its infrastructure must be equipped with multiple FPGA chip types. Furthermore, additional configuration for bridging PCIe cards to virtual machines in which the customer's code is being developed. As proposed before, the most cost-effective deployment would be that which is based on a cloud computing platform and every client receives a number of virtual machines. These virtual machines can be of two types: development and validation. The development ones are used to build the project whereas the validation ones must be connected in some way with the FPGA chips that are targeted for testing. A standard method of connecting a hardware devices directly to a virtual machine is by using a Input/Output MMU virtualization (Intel's VT-d and AMD's Vi), sometimes referred as pass-through. By using an IO memory management unit virtual guests can directly access hardware resources that are present on the hypervisor. The motherboard and the BIOS firmware must also support this feature, apart from the CPU. There is a difference between PCI and PCIe devices in the sense that all PCI resources at once can be passed through while PCIe devices can be configured individually. This situation arises from the protocol's designs. There are, however, certain restrictions amongst different hypervisors with regard to this technology. The following table summarizes compatibility:

Table 4. Hypervisor-PCIe support

Hypervisor	**Supported**
Linux KVM	Yes
VirtualBox	Only on Linux
Hyper-V 2005,2008,2012	No
VMware	Version/Product dependent

Given this restriction, the provider which would operate the infrastructure will have to pay attention to such features. Since KVM is the most used hypervisor on Linux, one choice would be the use of OpenStack, which can also integrate with VMware ESXi hosts. When using an x86-based processor, the hypervisor makes use of the native CPU instruction to achieve maximum performance. Intel VT's features enable faithful abstraction of the full prowess of Intel CPU to a virtual machine. All software in the VM can run without any performance or compatibility hit, as if it was running natively on a dedicated CPU. Live migration from one Intel CPU generation to another, as well as nested virtualization, is possible [9].

On OpenStack, the compute service is responsible for interacting with the underlying hypervisor on a particular host. It controls the hypervisor through an API server. Linux KVM is the default hypervisor for Compute. The PCI pass-through feature in OpenStack allows full access and direct control of a physical PCI device in guests. This mechanism is generic for any kind of PCI device. Thus, an FPGA card which is installed on a PCI/PCIe bus would be visible to the virtualized guest as if it was directly connected. Before the existence of this technology, any device exposed to the guest machine would have been emulated. The data exchange between the emulated device and the physical one would have been mediated by the hypervisor, thus leading to lower performance and often the lack of full capabilities of the exposed device inside the virtual machine. One of the problems introduced with device pass-through is when live migration is required. Live migration is the suspension and subsequent migration of a VM to a new physical host, at which point the VM is restarted. This lack could be resolved by the use of PCI hot plugging, a technology which permits the insertion and removal of PCI devices at runtime. Even if guest and hypervisor support is present, the PCI card must also be capable of supporting such a feature. In the case of many FPGA-based PCI devices, this is not to be expected since the actual physical removal and insertion when a system is powered in is unlikely to happen.

Some PCI devices provide Single Root I/O Virtualization and Sharing (SR-IOV) capabilities. When SR-IOV is used, a physical device is virtualized and appears as multiple PCI devices. Virtual PCI devices are assigned to the same or different guests. In the case of PCI pass-through, the full physical device is assigned to only one guest and cannot be shared [9]. If the device under contains at least one PCIe core which provides virtual functions, the underlying test infrastructure must be able to assign to multiple virtual test machines a corresponding virtual function, in order to fully test the FPGA configuration.

If using the pass-through method on a hypervisor which runs multiple test systems, then the system must be configured in such a way that it will never forward the same FPGA card to multiple virtual machines. This may lead to system instability and potential data-loss.

Another solution for a testing farm could be based on a private cluster architecture. This is fundamentally different from a cloud approach due to the fact that is non-interactive. A typical workflow is presented in Fig. 8:

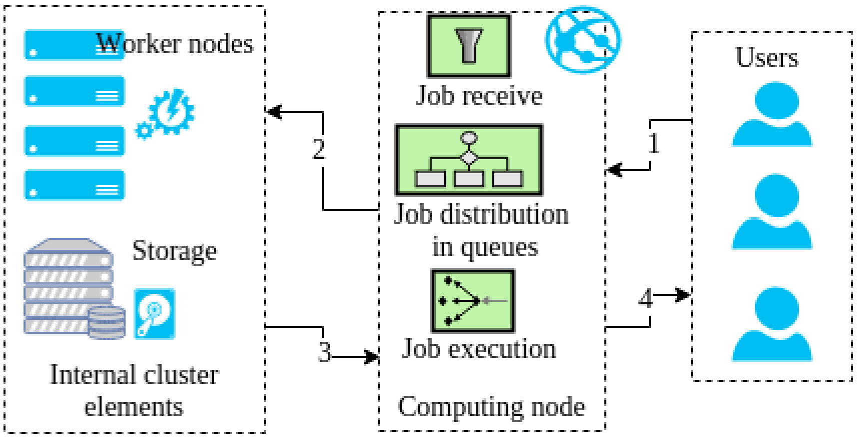

Figure 8. Private cluster architecture

Users submit jobs through a dedicated cluster interface which resides on the Computing Node. After the cluster manager receives the jobs, it evaluates their requirement, and queues them up for submission. When a node is free, the cluster manager submits a job to that node. When the job starts executing on the assigned node, it will first preconfigure the environment and fetch, if necessary, any input data. After the execution is done, the result is usually stored in a common location or on a Storage Element. At this point, the user that submitted the job can view its result. This setup, however, implied additional complexity which might not be visible in the beginning. The main challenge is to have a framework integrated with the cluster's batch system that can manage the FPGA card's resources, communicate with the host, exchange data with the software processes that interact with the FPGA and provide an easy to use Application Programming Interface (API) for high-level software programmers. At the same time, any hardware programmer must be able to design a specific acceleration module without the need of interacting with other components such as AXI buses or DMA cores. In order to be generic, the architecture must be vendor-independent and must be able to accommodate any acceleration module that implements the required interfaces and signals, without any particular hardware requirements. Since a large computing cluster is usually heterogeneous in terms of host system CPU architecture, data buses, operating systems and referenced libraries/functions by user jobs, the framework must be able to allow runtime environment rebuilding and it must automatically manage the underlying changes before a job is launched.

A cluster farm that is used exclusively for testing would be more cost efficient than a cloud infrastructure. In the testing scenario, the host on which a FPGA configuration is validated must have at least one FPGA connected. The difference is that of computing power and hardware resources required. While the cloud architecture would need more memory and CPU in order to support virtual machines, a worker node inside the testing cluster would need moderate resources since its only job is to test and communicate with the FPGA and not to run virtual machines.

There are also software considerations that need to be taken into account when deploying a cluster-based FPGA testing farm. Neither the cloud approach nor the cluster one provide out-of-the-box solutions for the scenarios discussed in the present paper. In both cases FPGAs need to be programmed with the firmware under test. Usually a complete device program involves a system reboot, especially true for PCIe setups, since the motherboard’s PCIe root may need to acknowledge and configure the newly instantiated cores.

5. CONCLUSIONS

FPGA-based solutions are becoming more and more visible as they are being incorporated into various electronic devices, in most cases as an extension to System on a Chip architectures. Companies that don't have FPGA development as their primary market, or small and start-up companies without a lot of investment funds can benefit from cloud services in order to decrease their FPGA development time and implementation costs. The alternative would be investing into in-house resources and managing a private grid or computing cluster. This approach can have a big financial impact on the whole project. Apart from the technical knowledge, the resources can be provided by an external party, such as the hypothetical company presented in the previous section. Thus, a company can exactly evaluate the monthly costs of such a service, for a limited period of time, which can lead to an overall lower development cost.

Current design flow can prove to be time consuming and the required resources might not fit into a project's financial flow, if the FPGA component performs an auxiliary, but mandatory, function, with regards to the overall project.

A cloud based approach, with interactive application forwarding and a solid back-end for batch building, can be a viable alternative for such situations. Cloud has changed the industry both in terms of financial returns and in the visible support that it offers to small businesses. By reducing the total cost of ownership, small companies can now access the power and versatility of FPGA chips to develop new and innovative solutions. We can envision a near future where all these technologies can help concepts such as smart cities to become reality. The smart city concept promotes the use of Information Technology to enhance the performance and quality of the services offered to citizens. FPGA technologies can enable applications such as urban traffic management where real-time response is crucial. Interconnection between different systems can be done in a secure manner by using dedicated FPGA’s for secure communication. In addition, companies such as Xilinx have been releasing tools that simplify the use of more common languages such as C and C++ to program FPGAs. This is an important factor in popularizing the FPGA development among start-ups.

A service offer represents a quantified set of services and a range of applications that end users can use through the provider. Service offerings should include resource guarantees, metering, resource management and billing cycles. Service management functionality should be developed in a such way that the defined services can be quickly and easily implemented and managed by the end user. For a cloud service to be truly on demand and at the same time to able to meet service level agreements, it must be able to manage at any time an increase in workload. Management solutions must possess the ability to create policies around workload and data management to ensure the efficiency and performance delivered by the system running in the cloud.

The paper has outlined existing solutions, with their advantages and disadvantages, and has proposed a new workflow that uses current cloud technology. Such a public service would be endorsed by a dedicated company which has focus on providing cloud FPGA compile services. The backend would be a cloud stack such as OpenStack or CloudStack, with a web frontend through which a client can manage his projects and resources. All communication would be encrypted and data would be stored on the provider's disks only during the project. Confidentiality and data integrity would be assured through normal means such as Service Level Agreements and Non-Disclosure agreements.

6. REFERENCES

[1] LabVIEW FPGA Compile Cloud Service, http://www.ni.com/white-paper/52328/en/

[2] LabVIEW FPGA Compile Worker Compile Time Benchmarks, http://www.ni.com/white-paper/14040/en/

[3] Laurențiu A. Dumitru, Sergiu Eftimie, Dan Fostea, *An FPGA-Based cloud storage gateway*, 2nd International Conference SEA-CONF, Academia Navală Mircea Cel Bătrân, Constanța, 2016

[4] Gorelik, Eugene. *Cloud computing models*. Diss. Massachusetts Institute of Technology, 2013.

[5] Yi, Sangho, Artur Andrzejak, and Derrick Kondo. "*Monetary cost-aware checkpointing and migration on Amazon cloud spot instances*." IEEE Transactions on Services Computing 5.4 (2012): 512-524.

[6] Greenberg, Albert, et al. "*The cost of a cloud: research problems in data center networks*." ACM SIGCOMM computer communication review 39.1 (2008): 68-73.

[7] Banik, Thomas, et al. "*System for virtual process interfacing via a remote desktop protocol (rdp).*" U.S. Patent Application No. 10/527,913.

[8] Dumitru Laurențiu A., Sergiu Eftimie, et al. "*A novel architecture for authenticating scalable resources in hybrid cloud.*" Communications (COMM), 2016 International Conference on. IEEE, 2016.

[9] OpenStack documentation https://docs.openstack.org

[10] Intel Virtualization Technology, http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html