EDA to the cloud – a case study to increase both effectiveness and user experience of EDA tools.

Pedro Biscaia
Universidade de Lisboa
Instituto Superior Técnico
Lisbon, Portugal
pedro.biscaia@tecnico.ulisboa.pt

***Abstract* — This work presents an evaluation of the different types of clouds and its providers and proposes an implemented solution of Electronic Design Automation (EDA) integrated with Amazon Web Services, focused on transposing all the computation into the cloud. Over the past years the cloud has evolved into a more secure and accessible range of services. Yet, despite the advantages of cloud services, EDA industry resists to adopt those services although EDA applications require processing and storage capacity, which are available in the cloud. Moreover, the approach of this work involves EC2 computing, S3 storage and RDS database services to be integrated with AIDA, an analog integrated circuit design automation environment. The end result is a scalable version of the AIDA application using Cloud Computing for all the processing.**

Keywords — Cloud Computing, Amazon Web Services, Electronic Design Automation, Analog Integrated Circuits, AIDA, EC2.

I. INTRODUCTION

Electronic Design Automation (EDA) is becoming ever more important with the continuous scaling of semiconductor devices and the growing complexities of their use in circuits and systems. New challenges to design automation systems are raised with the demand for lower-power, higher-reliability and more agile electronic systems. Historically, Central Processing Unit (CPU) performance and the amount of cores was the primary bottleneck to EDA tools. Semiconductor companies typically reacted by adding cores, when higher performance was needed.

The purpose of this thesis is to study the advantages of Cloud Computing services in a specific solution of EDA, AIDA, which requires high computational power and scalability in order to suit the application's complexity together with the demands for short execution time. Aiming to compare the runtimes of various circuits, different amounts of cores were used.

In this context, the first part of this study focuses on a preliminary evaluation of the cloud and all providers, according to the type of service they sell, in order to determine what cloud provider better suits the requirements of AIDA. Secondly, the functionalities of the application are identified with the correspondent cloud services, in particular (i) the computational service EC2, (ii) a database service Amazon RDS and (iii) a storage service Amazon S3. During this process the architectures of each of the required cloud services are analyzed and a global architecture for the application with the integrated cloud services is defined. Thirdly, the architecture is used as a blueprint to develop and implement the application. The subsequent steps include a guide to start, setup and launch the cloud services needed. Finally, the implemented solution is tested with two distinct circuit examples, a Low Noise Amplifier (LNA) and an Operational Transconductance Amplifier loaded with Folded Voltage Combiners (FVCOTA), using different computational resources to estimate some performance results.

The paper is organized as follows. Section II provides a detailed review of Cloud Computing concepts and structure. EDA industry and existing related cloud solutions are also presented. At the end of Section II, one service provider is selected to be used in this study. In Section III, cloud services and AIDA architecture are addressed. Section IV describes the implementation of the proposed solution, while Section V describes the implemented solution performance results with two case study examples. Finally, in Section VI the conclusions of this study are drawn.

II. CLOUD COMPUTING AND EDA

A. Cloud Computing Review

Cloud Computing presents a better alternative to run a business when compared to traditional computing. The success of cloud systems can be explained by the fact that companies do not run their applications themselves. Instead, applications run on a shared datacenter, to which companies simply need to plug in like a utility, this being a much faster and cheaper way to get started. As defined by the National Institute of Standards and Technology, "Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (*e.g.*, networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [1].

The five essential characteristics of Cloud Computing are:

- **On demand self-service** is the first characteristic, which refers to the service provided by Cloud Computing vendors that enables the provision of cloud resources on demand, such as computing power, storage, networks and software, by the consumer, whenever they are required;

- **Broad network access** establishes that resources and capabilities of Cloud Computing are available for access from a variety of devices in any given location that offers online access;
- **Resource Pooling** represents the idea that the cloud providers serve multiple clients, customers or tenants with scalable and provisional services, such as processing, bandwidth, memory, storage, network and virtual machines;
- **Rapid Elasticity** represents the scalable provisioning or the capability to provide scalable services;
- **Measured Service** refers to services where the cloud provider is able to measure or monitor the provision of services for different reasons, as well as billing, effective use of resources, or overall predictive planning.

The three main categories of cloud service models are: "Infrastructure as a Service", "Platform as a Service" and "Software as a Service". This classification is relevant to establish the role that a particular cloud service satisfies, and how that service accomplishes its role

Infrastructure as a Service (IaaS) is the lowest level cloud service and works as a self-service model for managing remote data center infrastructure. Computing resources such as storage, networks or servers are provided by IaaS over the Internet hosted by a third party. Instead of an organization acquiring hardware, companies acquire IaaS based on a consumption model, where the company only pays for what it uses. This model enables companies to add, delete or reconfigure IT infrastructure on-demand.

The **Platform as a Service (PaaS)** layer enables organizations to build, run and manage applications without the IT infrastructure. This accelerates and simplifies the process of developing, testing and deploying applications. In this context, developers can focus on writing code and create applications, instead of working on IT infrastructure activities such as provisioning servers, storage and backup. PaaS helps reducing the costs and management overhead, while at the same time makes it easier to innovate and scale the services on demand. PaaS offerings usually consist of a base operating system and a suite of applications and development tools. It is also often called "middleware", due to how it conceptually makes the connection between SaaS and IaaS.

The **Software as a Service (SaaS)**, known as "on-demand software", replaces the traditional on-device software with one that is licensed on a subscription basis. It is centrally hosted in the cloud and, in general, end users access its services through a web browser, the user's operating system being irrelevant. The billing system is based on consumption or on a flat monthly charge.

Figure 1 provides a visual representation of the cloud service models and its varying levels of control between customer and provider, over assets and services.

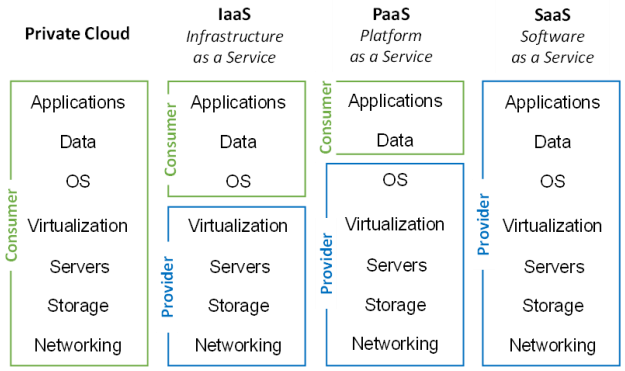


Figure 1 - Architecture and levels of control over assets and services in Cloud Computing.

There are three main factors to address what is the right cloud service model for a specific application, whether in business or personal context. A specific tool should be used for its particular purpose and there are three main factors that determine which cloud service model to use. The first factor is related to deciding whether to build its own code and to pay for a SaaS solution to provide the solution on demand. Secondly, performance and scalability requirements of the application differentiate between PaaS and IaaS. Specifically, PaaS solutions must provide auto scaling and failover capabilities for all tenants of the platform, thus being difficult to achieve a very high scale. On the other hand, regarding IaaS, it is up to the consumer to decide how to scale and prevent failover. Finally, the third factor is security. It is important to understand the backup plan of the cloud vendor in case of an outage or a crash.

Typically, Could Computing systems involve two main layers that make up a scalable cloud architecture, front-end and back-end, being interconnected via a virtual network or Internet. **Front-end** is the side that is visible to the client, customer or end user. It includes the client's computer system or network that is used for accessing the cloud system. The **back-end** area is where all the backbone needed for cloud architecture is. This is composed of server farms in a datacenter that house virtualized servers, virtual networks, shared storage security mechanism and services, all built in conformance with a deployment model.

B. Cloud providers comparison

In order to select the cloud provider that suits this work the best, it is necessary to analyze the available options on the cloud market, as well as to define the type of service model that better serves the purpose of this work. According to the Clutch, the best cloud service providers in 2018 are, Amazon Web Services, Microsoft Azure, Google Cloud Platform, IBM Cloud, Rackspace, GoDaddy, VMware, Oracle Cloud, 1&1, DigitalOcean, CloudSigma [2].

It is possible to filter some of the cloud providers' options. IBM cloud is directed towards business solutions and not so oriented to simple computing and storage purposes; Rackspace is dedicated to hosting services; GoDaddy and 1&1 are better suited for small business seeking website tools; VMware main efforts are on testing and developing applications and it is not suited for simple, private computing platforms.

As the purpose of this work is to move an application to the cloud, more specifically to the computational and storage part, the SaaS model which utilizes the internet to deliver applications to its users can be excluded from the service models options, because this work requires control over the platform and infrastructure. This thesis intends not only to move an application to the cloud, but also to quickly and regularly scale resources, if required. In addition, this work aims to accommodate heavier workloads almost instantly, while scaling back during periods of low usage. Not only IaaS is the service model that presents these characteristics, but it also has the advantage of lower costs, because there is no need for buying hardware, and it is easier to save money on maintenance and controlling costs.

The Cloud Computing market is composed by three main players: AWS, Microsoft Azure and Google Cloud Platform (GCP). In order to compare the three providers, computation, storage, networking, pricing, reliability and ease of use are used as main criteria. Some conclusions can be taken concerning the three main cloud service providers:

- If an enterprise requires Windows integration or if it is seeking a strong PaaS provider, Microsoft Azure would be the perfect choice.
- If there is a strong emphasis on analytics or if the budget is limited, GCP can be the solution.
- If a long-lasting reputation is important, and there is the need for IaaS or a wide range of other tools, AWS may be the best option.

In order to do some early tests without the concern of costs all the three vendors could be an option, because they have a free tier model. However, having selected IaaS as a service model and due to the considerable amount of detailed information and examples over the Internet, AWS was the cloud service provider chosen for this work.

C. EDA in the Cloud

EDA can be defined as a collection of design automation and test automation tools, where quality aspects of the electronic system, such as defect level, test cost or ease of self-test and diagnosis, are managed [3]. Moreover, EDA contains a set of tools essential for designing and analyzing a modern semiconductor chip, which can have billions of components [4].

Physical design and physical verification software are examples of compute-intensive EDA applications that require such techniques. The development of multi-threading and multi-processing capabilities, along with cheaper resources such as CPUs usage, leads to a cost-effective solution way to reduce EDA software run times. However, delivering the right solution requires new software architectures with flexibility for linear scalability across a large number of CPU cores, along with a powerful data model that allows concurrency between design and verification [5].

In order to define the desired solution for EDA industry, two sections have to be discussed: hardware and software.

Faced with the fact that increasing clock frequency is not sustainable due to power constraints, **hardware vendors** have incorporated parallelism to drive improved performance. Some powerful solutions for general computing contemplate multicore CPUs, Graphics Processor Units (GPUs) that are highly parallelized or single instruction multiple data (SIMD) machines for a better performance of multimedia use.

Software vendors' desired solution requires them to effectively use the hardware resources to affordably accelerate computation. This implies that a costly custom hardware solution cannot be implemented because cost increases when there is the need for a large amount of memory and the software is restricted to run only a single machine.

Private cloud solutions have been available for some time in the world of semiconductor design, and some EDA companies are offering those solutions in various forms. For example, Cadence Design Systems offers both Hosted Design Solutions (HDS) and QuickCycles Service. HDS includes hardware, software and IT infrastructure, whereas QuickCycles Service offers on-site or remote-access to Palladium emulation and simulation acceleration resources. One alternative cloud solution, hybrid cloud, is also emerging as an interesting option, where non critical data that is not frequently accessed can be stored with very low transport costs in the public cloud [6] [7].

Market of public clouds are changing constantly and are becoming more appealing to EDA as new improvements may arise. Meeting the growing infrastructure needs of today, while avoiding investments that are incompatible with the future cloud migrations, is a major challenge of IT administrators. Overcoming this challenge includes selecting a platform that delivers the performance and flexibility required from EDA companies and, at the same time, enabling easy migration from private to hybrid or even public cloud. Dell EMC Isilon is an EDA proven high performance network attached storage platform that provides native connectivity to the most popular public cloud providers, including AWS, Microsoft Azure and EMC Virtustream [8].

III. PROPOSED SOLUTION ARCHITECTURE

The final result of this thesis is an application able to run the layout generation part of AIDA computing in the cloud and subsequently run the proprietary applications (circuit simulator and extraction tools) locally. Different cloud services such as computing, database and storage are involved to accomplish and assembly the final solution. The proposed architecture works around legal and licensing limitations due to some external tools used in AIDA, namely Mentor Graphics' reference circuit simulator ELDO [9] and layout extraction tool Calibre [10], and only AIDA's layout generation tool AIDA-L is executed in the cloud.

A. AIDA SOFT, an EDA Case Study

An analog integrated circuit design automation environment named AIDA is the case study of this work, and it implements a design flow from a circuit-level specification to a physical layout description and is based on AIDA-C [11] and AIDA-L [12], the two in-house tools.

AIDA-C can be defined as an innovative layout-aware optimization-based methodology for automatic sizing of analog Integrated Circuits (ICs). AIDA-L is the layout generator, and implements a fully automated layout generation methodology. A layout-aware synthesis methodology for automatic optimization-based sizing of analog ICs is achieved with the integration of AIDA-C and AIDA-L [13].

As already mentioned, AIDA is composed by AIDA-C and AIDA-L and its architecture is illustrated in Figure 2. The whole process starts with AIDA-C carrying out a sizing optimization, based on given specifications. Then, from the sized circuit level description (output from AIDA-C), the transformation to the physical level is made by AIDA-L. After this stage is completed, an optimized physical design is generated (in GDSII format). Nevertheless, a final validation must be carried out to confirm if the initial specifications are met [14]. Finally, results are validated using industrial simulators and analysis tools, namely ELDO or CALIBRE.

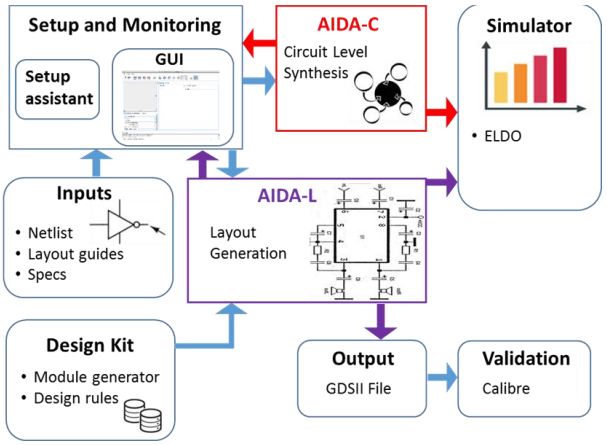


Figure 2 - AIDA Architecture.

B. Cloud Computing service model infrastructure

Taking into consideration the evaluation and conclusions made in Section II, the service model selected for supporting the application is IaaS and the cloud provider is AWS. Figure 3 illustrates the IaaS architecture, as well as the levels of control over the multiple cloud assets. With this selected architecture the user will have access to applications, data, runtime, middleware and operating system, while AWS is responsible for virtualization, servers, storage and networking.

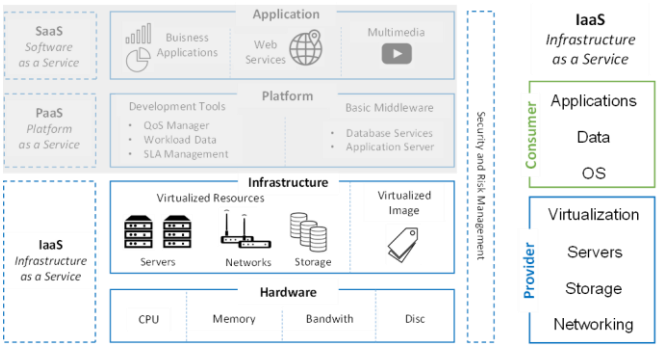


Figure 3 - IaaS service model description.

C. AWS Services

1) EC2 Elastic Compute Cloud

Amazon Elastic Block Store (Amazon EBS) is a cloud service that provides block level storage volumes for use with EC2 instances. In order to secure an instance to be launched, the user has to specify a key pair and a security group. Afterwards, the client must determine the private key of the key pair every time he/she wants to connect to the instance. The instance overview architecture is represented in Figure 4.

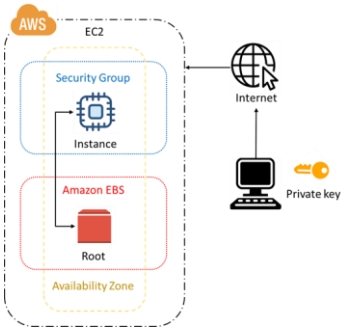


Figure 4 - EC2 instance overview architecture.

There are different types of instances varying the number of CPUs and memory. However, the only instance included in the free tier plan is the t2.micro, which has 1 CPU and 1 GB RAM and was used for early testing.

2) Amazon RDS Database

One of the modules of this application consists in regulating the user access with a login and registration mechanism, which requires the support of a database system. The data to be stored in the database is structured data, because the fields and type of data expected are known and predictable, thus, the most suitable database is a relational database.

As the AIDA application is built based on Java, the database engine MySQL was the one selected due to its compatibility with Java. This engine is included in the free-tier plan. It was required only one table named “user_info” with four fields: “user_id” as a primary key, “user_name”, “password” and “user_email”. The schematic view and information about the columns is presented in Figure 5.

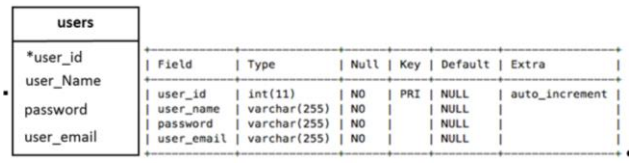


Figure 5 - Schematic view and information about the columns in the table user_info.

3) Amazon S3 Storage

In order to run this computational segment it is required a setup file. In addition, at the end of execution one file containing the results for the simulation is generated in the user local machine. Regarding the need for uploading and downloading files into and from the cloud, Amazon S3 is the solution for storage in AWS. It is organized in buckets and can be integrated with java applications and be used to store and retrieve any quantity of data, at any time and from anywhere on the web.

4) Application with Cloud Services Integrated

It is necessary to understand how the previous cloud services are integrated into the final solution, as well as the road map for the user to execute the program.

When the java application is launched, the user has to login or register and then login into the system in order to verify its identity, this step using the Amazon RDS services. If successfully logged in, the user can select the number of CPUs to run the computational segment of AIDA in the cloud, having previously uploaded the required setup file into the cloud for the execution. The previous actions regarding the computing in the cloud are achieved using the EC2 computing services and the S3 storage service. After the computation, the result file has to be downloaded from the cloud into the local machine. Subsequently, the user can do the extractions and simulations in the AIDA environment locally, as required due to licensing and non-disclosure agreements.

Moreover, AIDA standalone application, which is described in Chapter 3.1, is located in the Instituto Telecomunicações (IT) server and is also able to do the required computation and simulation. Specifically, the IT server can be defined as a private cloud where the AIDA tools work under a private environment with physical hardware maintained by the IT investigators.

An overview of the architecture of the AIDA standalone and the global application and the connections within its components is illustrated in Figure 6.

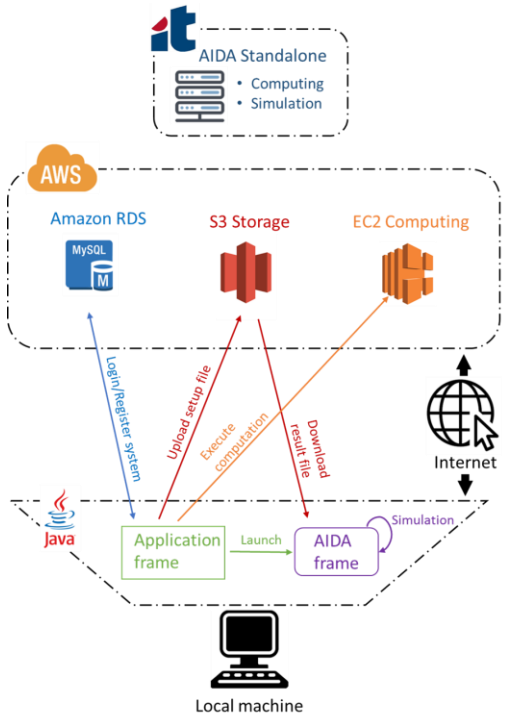


Figure 6 - Architecture overview.

IV. Implementation of a Working Solution

The presented architecture and concept were validated by developing a working solution. From a technical point of view, the solution consist of two modules: (i) configure and launching the AWS services, database, computation and storage, and (ii) the integration of those services with the java interface.

A. Interface and Application Structure

As the AIDA application is written in Java, the modules and interface of this work were also built in Java. The interface was built based on Swing, which purpose is to create window-based applications and, unlike Abstract Windowing Toolkit (AWT) Application Programming Interface (API), it is entirely written in Java and, therefore, Swing components are platform-independent or also known as lightweight components.

Aiming to separate the different modules and functions, a tabbed windowed structure was implemented, so that different components can share the same space. Four tabs were created: “Instructions”, “Login”, “Register” and “Application” as shown in Figure 7. Each tab encompasses the actions related with its name.

- **Instructions**: this tab contains the instructions for the correct use of the all application.
- **Login**: is the tab where the user can fill the user name and password to be validated.
- **Register**: similarly to the Login tab, the customer can do the registration, by filling in the email address, user name and password.
- **Application**: this tab is where the user can transfer files to/from the cloud, execute the computation in a selected number of CPUs and launches the AIDA frame to run the simulation.

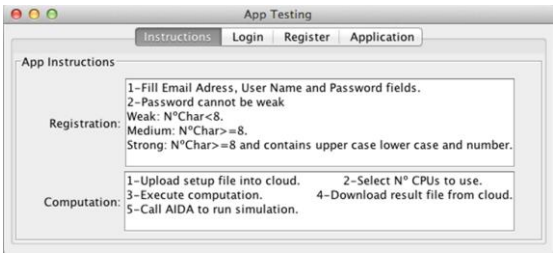


Figure 7 - Tabbed Windowed structure.

B. Implementation of Login/Registration Database System

In order to provide a basic level of security to the AIDA application, an authentication is required so that the access is restricted to the registered users. Moreover, each user should have a personal account, specifically, having dedicated storage space for files or documents. The starting point of both previous features is a login and register system, backed up with the database Amazon RDS.

Launching a database instance is a four step process after the user go to the RDS page and select “Launch DB Instance” option. Firstly, the customer has to choose an engine. In this work the engine selected was MySQL database. The following steps are left as default or filtered by being under the free usage tier.

To create a table inside the database to store the users’ information is required employing MySQL utility, establishing

a connection with the DB instance using its endpoint ("*sample-instance.curdstcko9sl.eu-west-2.rds.amazonaws.com*") and running the command on terminal, "*/usr/loca/mysql –h sample-instance.curdstcko9sl.eu-west-2.rds.amazonaws.com –P 3306 –u –awsPedro –p".*

1) Register Tab

The register tab contains three fields: email address, user name and password for the user to fill in and a submit button to execute the registration process. Provided that the email address and username are still not taken and that the password is valid, the user can register with those credentials.

A set of verifications take place within the registration action, Blank fields, repeated Email Address or User Name and Weak Passwords trigger alerts to correct those fields as it is shown in Figure 8. Three levels of password strength were defined and are displayed in the instructions tab:

- **Weak**: whenever the number of characters is less than eight.
- **Medium**: whenever the number of characters is equal or greater than eight.
- **Strong**: whenever the number of characters is equal or greater than eight and exists at least one upper and lower case letter and one number.

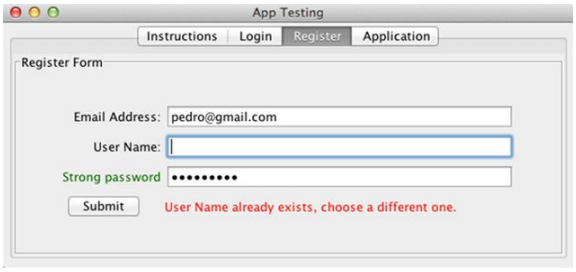


Figure 8 - Verification of repeated fields with the database.

Upon a successful registration, an action related with storage occurs. In order to provide a dedicated storage folder to each user, upon the registration, one folder is automatically created with the user name. However, Amazon S3 data model is a flat structure, meaning that the user creates a bucket and the bucket stores objects, with no hierarchy of sub-folders or sub-buckets. In order to overcome this flat structure issue, enabling the creation of a folder per user, the customer can infer logical hierarchy by using key name prefixes and delimiters as demonstrated in Figure 9.

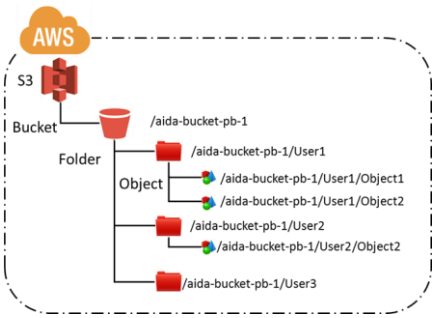


Figure 9 - Amazon S3 hierarchy.

1) Login Tab

The login tab is similar to the register tab, and the same interface verifications are applied in both. The login tab has the purpose of validating the user name and password with the database and, if the information matches, the user is successfully logged in. In this tab, the user is asked if he/she is already registered, and there is one button named "Register" that takes the user to the register tab, in case the user is not registered yet. Figure 10 represents the view of the login tab. There is another button named "Go To App" that takes the user to the application tab, only enabled after a successful.

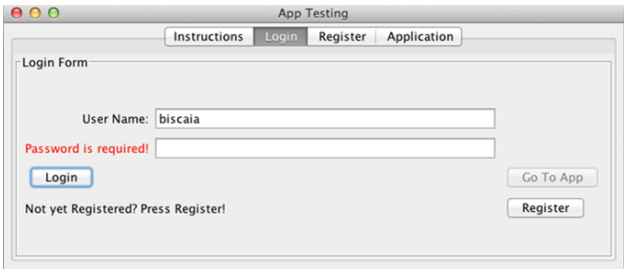


Figure 10 -- Login tab with verification of blank field alert.

C. Implementation of a Computational and Storage Module

Launching an Amazon EC2 instance is a simple seven step process, after the user accesses to EC2 services and selects the "Launch instance" option in the instance area. The selected Amazon Machine Image (AMI) was "Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type - ami-dff017b8" because its operative system is Linux, compatible with Java and MySQL and because it is eligible for the free tier. The instance types selected for the testing are specified on Table 1. To finish the launching process some security configurations that define what type of traffic can access the instance, have to be established.

Table 1 - Number of CPUs and corresponding instances.

CPUs¤	Instance·Name¤	Type·of·Instance¤	Memory·RAM·(GiB)¤	Price·($/hour)¤
1¤	t2.micro¤	General·purpose¤	1¤	Free·Tier·eligible¤
2¤	c5.large¤	Compute·optimized¤	4¤	0,085¤
4¤	c5.xlarge¤	Compute·optimized¤	8¤	0,17¤
8¤	c5.2xlarge¤	Compute·optimized¤	16¤	0,34¤
16¤	c5.4xlarge¤	Compute·optimized¤	32¤	0,68¤

In order to get started with Amazon S3, the user has to be logged in the AWS account, and a bucket can be created in the S3 service. A bucket is where every objects in Amazon S3 are stored. The free tier usage allows a customer to receive 5 GB of Amazon S3 storage in the standard storage class, 20000 Get Requests, 2000 Put Requests, and 15 GB of data transfer out each month for one year.

1) Application Tab

The "Application" tab is divided into three sections: "Transfer files", "Cloud Computing" and "Local AIDA GUI", as illustrated in Figure 11.

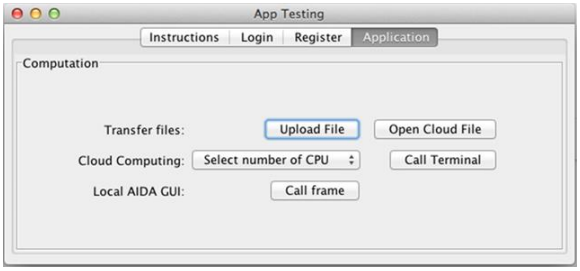


Figure 11 - Application tab.

The transfer files section contains two buttons: one button is used to upload files from the local machine into the S3 bucket inside a folder with the user name logged in; the other button allows to download an object from de S3 environment into the local machine. Both actions, download and upload, were implemented using the Java packages "com.amazonaws", which provide the classes that allow a Java application to connect with AWS services, and holds the client classes that are used for communicating with Amazon S3. However, they only provide connection between the local machine and the S3 bucket, being necessary that the uploaded setup file is inside the EC2 instance, and that the result file, which is a by-product of the computing in EC2, is in S3 to be downloaded.

To solve the connection problem between S3 and EC2, a copy command within the EC2 command line system is used. One example of this command is represented in Figure 12, where the file scriptEC.sh is copied from EC2 instance to S3, to the "bucket aida-bucket-pb-1" and under the folder "biscaia".

```
[ec2-user@ip-172-31-3-114 ~]$ aws s3 cp scriptEC.sh s3://aida-bucket-pb-1/biscaia/scriptEC.sh
upload: ./scriptEC.sh to s3://aida-bucket-pb-1/biscaia/scriptEC.sh
```

Figure 12 - Command to copy file from EC2 into S3 bucket and folder.

When the instances are launched they are configured to run the computation, which includes setting the environment with the relevant information and folders. Among these folders and files, is the setup file. Therefore, unless the user wants to use a different setup file, the upload file button will not be used. These required files and folders were transferred into the instance with the help of a SSH File Transfer Protocol (SFTP) named CyberDuck.

With the setup files and environment ready to compute in EC2, the user has to select the number of CPUs to use from a dropdown box, as shown in Figure 13. The number of CPUs available are 1, 2, 4, 8 and 16, each option corresponding to a different instance with the respective number of CPUs.

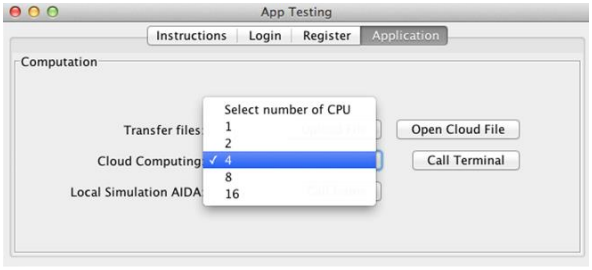


Figure 13 - Dropdown box for selecting the number of CPUs.

Having the number of CPUs selected the customer next step is to press button "Call Terminal" which will execute a local script in a new terminal window. This action is demonstrated in Figure 14. The script in a new terminal window gives some instructions for the user to execute after establishing an EC2 SSH connection. Once the user connects with the EC2 instance, the user can run the computing segment using the instructions in the script.

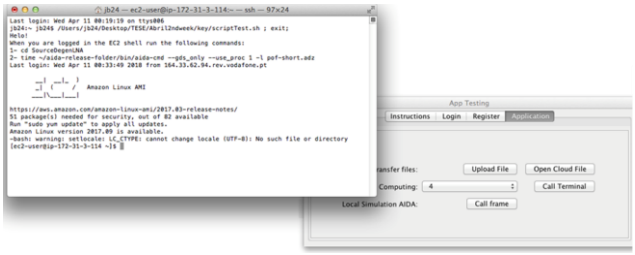


Figure 14 - Call Terminal action, running a script to connect with EC2 with instructions to run computation.

In the second command, "proc 1" represents de number of CPUs. So, there is one script for each instance, and the script to run is nominated accordingly with the selected number of CPUs in the dropdown box.

The successful outcome of this computing is the time it took to run and a result file in the EC2 environment. Regarding the time it took to run there are three times presented to the user: "real", "user" and "sys" (Figure 15). Nevertheless, the "real" is the time field took into consideration.

```
copt.LayoutAwareCircuitEvaluator.runLayoutAwareSimulation(LayoutAwareCircuitEvaluator.java:445): Thread.main: Complete Layout Generation
application.AIDACCmdLine$2.run(AIDACCmdLine.java:98): Thread.Thread-0: ShutdownHook terminated

real    10m59.853s
user    41m49.856s
sys     0m2.096s
```

Figure 15 - Terminal output from computation with time.

Once the result file is inside the EC2 environment, it is necessary to transfer it into the S3 storage, after which it can be downloaded by the user into the local machine using the "Open Cloud File" button. The command used to move the result file from EC2 into S3 is the same used in Figure 12, and the "Open Cloud File" button opens a file chooser where the user selects the local directory and name to download the result file, as illustrated in Figure 16.

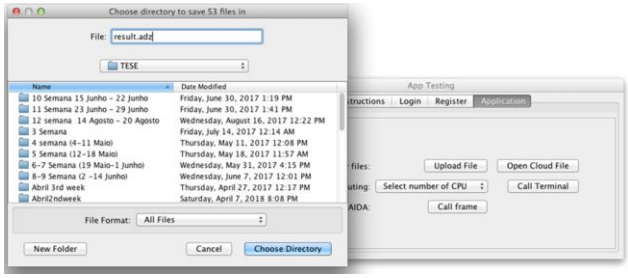


Figure 16 - Download action and file chooser.

The final execution is related with the local evaluation of the generated layouts using the AIDA frame, which is launched in the section "Local AIDA" by using the button "Call frame". The action result of launching the AIDA frame is displayed in Figure 17.

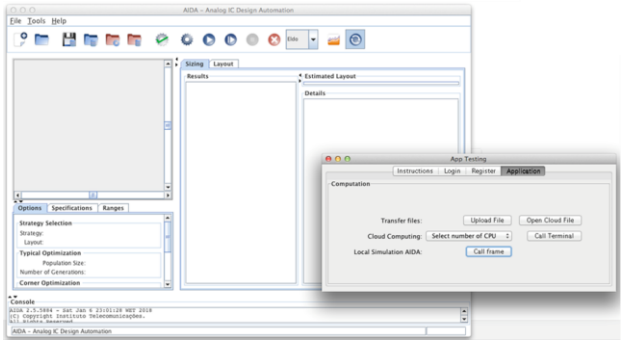


Figure 17 - AIDA frame launched from application.

The implemented solution requires the user to employ different mechanisms to transfer data, such as (i) Cyberduck a client for SSH File Transfer Protocol and (ii) terminal commands. Specifically, these mechanisms are used in the following situations:

(i) To upload files directly from the local machine to EC2, namely the setup files to run the computation process.

(ii) To transfer data from EC2 computation service and S3 storage service when the user needs to store the result file from computation in S3, and when the EC2 environment requires any file from S3.

To make the application more user friendly, the tools required for transferring data should be integrated within the API. However, this thesis is a work of concept proof, a more detailed and functional API being a major issue to further explore in future work.

V. Case Study

After having implemented the architecture solution to integrate AIDA and the cloud, several tests were executed varying the number of CPUs. Runtimes of the AIDA standalone solution and the AIDA integrated with the cloud were tested with examples of different complexity level.

Two examples were used to analyze the performance of the solution: (i) Low Noise Amplifier (LNA) [15] and (ii) operational transconductance amplifier biased by two folded voltage-combiners (FVCOTA) [16].

A. *Low Noise Amplifier (LNA)*

Figure 18 represents the LNA topology, which is a source degenerated topology.

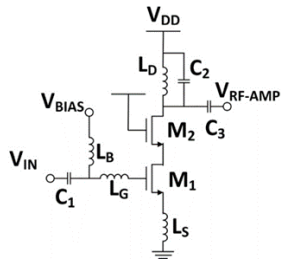


Figure 18 - LNA source degenerated topology.

A layout-corner-aware optimization methodology was used in the original paper to demonstrate the achievable results of the topology. This strategy implies that AIDA_L [12] performs the circuit layout and an automatic detailed routing is executed in order to obtain the layout for all solutions in the Pareto optimal front (POF) showed in Figure 19. The LNA POF represents the best LNA trade-offs between the circuit specifications for the considered technology, topology and frequency of operation, and one of the layouts obtained is illustrated in Figure 20.

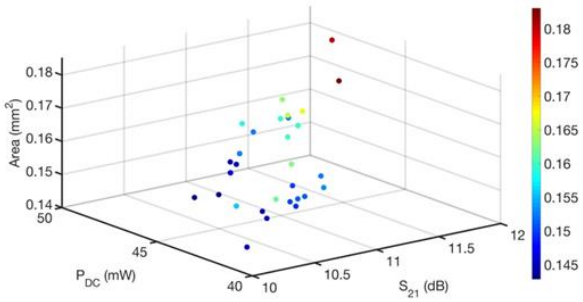


Figure 19 - LNA POF obtained with layout-corner-aware optimization.

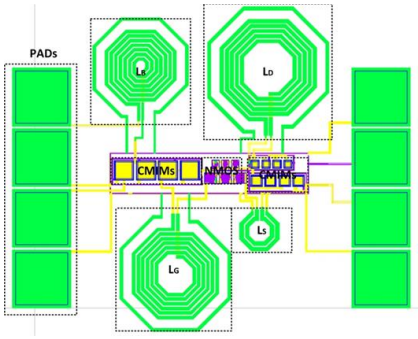


Figure 20 - Layout design of the LNA obtained with layout-variability-aware optimization.

In this work, layout generation, done by AIDA-L, was executed in the cloud, instead of locally. As shown in Table 2, this is a basic example with short runtimes.

Table 2 - Runtimes with example LNA.

CPUs¤	Instance·Name¤	Runtime·(minutes)¤
2¤	standalone¤	14'·16''¤
2¤	c5.large¤	21'·47''¤
4¤	c5.xlarge¤	10'·59''¤
8¤	c5.2xlarge¤	5'·58''¤
16¤	c5.4xlarge¤	3'·26''¤

When the CPUs were doubled, the runtime was almost reduced to half. This proportional tendency is illustrated in Figure 21.

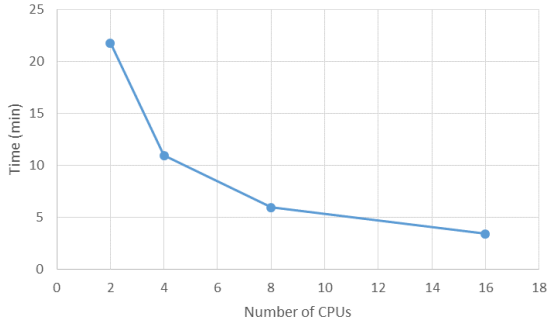


Figure 21 - Runtimes depending on the number of CPUs used in LNA example.

B. *Operational transconductance amplifier biased by two folded voltage-combiners*

The second circuit example, is represented in Figure 22 and consists of the operational transconductance amplifier biased by two folded voltage-combiners (FVCOTA) presented in [16].

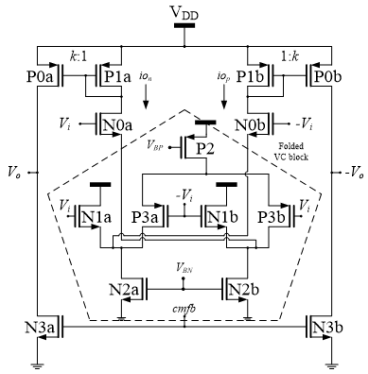
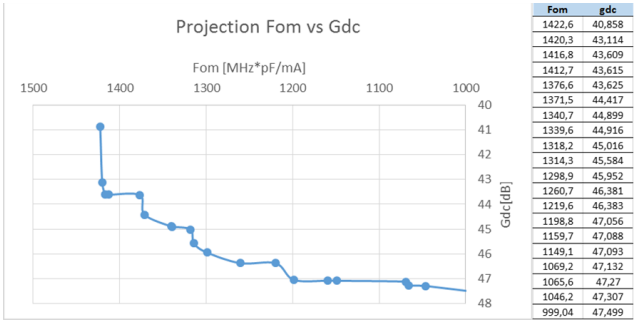


Figure 22 - Electrical scheme of the proposed folded VC biased OTA.

Running AIDA allow the automatic optimization of the circuit at sizing and layout level [13] and therefore the most competitive design solutions can be achieved and the tradeoffs between the energy-efficiency and low-frequency gain in the design of the OTA can be explored. Again output of AIDA sizing tool is a Pareto optimal front of feasible circuit solutions, representing a set of sized and laid out circuits all meeting the specifications but representing different compromises between the optimization objectives, providing the largest insight of the circuit under optimization. Figure 23 presents POF after optimization with corner case considerations.



Fom	gdc
1422,6	40,858
1420,3	43,114
1416,8	43,609
1412,7	43,615
1376,6	43,625
1371,5	44,417
1340,7	44,899
1339,6	44,916
1318,2	45,016
1314,3	45,584
1298,9	45,952
1260,7	46,381
1219,6	46,383
1198,8	47,056
1159,7	47,088
1149,1	47,093
1069,2	47,132
1065,6	47,27
1046,2	47,307
999,04	47,499

Figure 23 - AIDA POF after 4000 generations in Typical and 1500 generations in Corners.

Again the, the circuit layout generation was executed in the cloud, and, afterwards, the generated layouts can be visualized within AIDA locally as the Figure 24 shows.

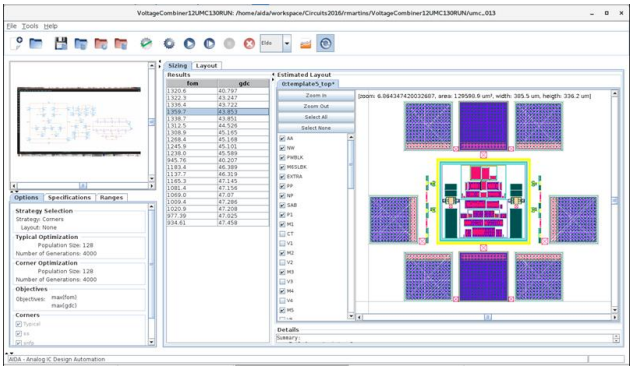


Figure 24 - Generated layouts within AIDA.

Listed in Table 3, are the runtimes of the FVCOTA example, which is more complex example than the LNA.

Table 3 - Runtimes with example FVCOTA.

CPUs¤	Instance·Name¤	Runtime·(minutes)¤
2¤	STANDALONE¤	1106'·47"¤
2¤	c5.large¤	1723'·17"¤
4¤	c5.xlarge¤	855'·24"¤
8¤	c5.2xlarge¤	486'·49"¤
16¤	c5.4xlarge¤	234'·5"¤

For this second test, the relation between number of CPUs used and the runtime is represented in Figure 25. It shows similar scaling as with the previous example. While for a single user the improvement of using the large amount of CPUs provided in the AWS seems to stagnate, in a multi-user scenario the advantages of having the scalable infra structure are immense, as it prevent over providing of local infrastructure while still providing good performance for each user.

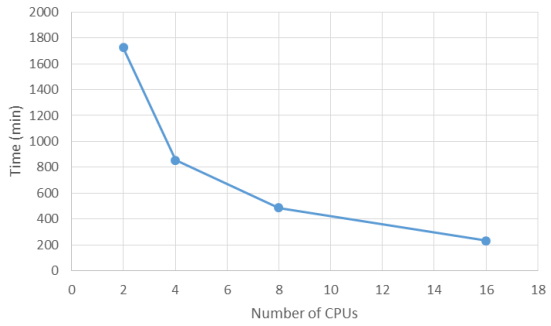


Figure 25 - Runtimes in example FCVOTA depending on the number of CPUs used.

VI. Conclusions

Over the past years great advancements occurred in computing technology, as the cloud has matured and companies became more confident in the technology's security and more aware of what cloud can offer in terms of capabilities, convenience and cost savings. Cloud services can be regarded as a solution for the EDA industry. In fact, this work validated that AIDA, an EDA application, can be integrated with AWS, having the entire computational segment running on EC2 machines, with various combinations of CPUs and memory, depending on the complexity of the task or the time constraint.

The conceptualized architecture was focused both on authenticating a user and on computing performance. Due to the low processing capacity of the instance included in the free tier usage, instances with more CPUs and memory were purchased to obtain valuable testing results. This work also proved how easy and more advantageous is to scale up from a machine with two CPUs to another machine with sixteen CPUs by a fraction of the price, rather than acquiring the whole servers and machines. The architecture and implemented solution are modular and atomic, meaning that another cloud services and features can be transposed and included.

Overall, the results of using different instances clearly showed a proportional relation between the time of computing and the number of CPUs used. Specifically, the time of execution to perform the computation is reduced to half if the number of CPUs doubles.

The existence of an implemented solution contributes to a better integration between AIDA application and cloud services. It also provides an interface for executing tests on different instances with a better user experience. Overall, the model can help reducing AIDA development and maintenance costs.

References

[1] P. Mell, L. Cheng, T. Grance, "The NIST Definition of Cloud Computing", 2011.

[2] Clutch, "Best Cloud Service Providers", [Online]. Available: https://www.sitepoint.com/a-side-by-side-comparison-of-aws-google-cloud-and-azure/ [Accessed 25 February 2018].

[3] L.-T. Wang, Y.-W. Chang, K.-T. Cheng, "Introduction", Electronic Design Automation, pp. 1-38, 2015.

[4] R. Brayton, L.P.Carloni, A. L. Sangiovanni-Vincentelli, T. Villa, "Design Automation of Electronic Systems: Past Accomplishments and Challenges Ahead", 2015.

[5] J. Lee, "The advantages of using massive software parallelism in EDA", [Online]. Available: https://www.design-reuse.com/articles/21680/massive-software-parallelism-eda.html. [Accessed 10 April 2018].

[6] Cadence, "Cadence Design Systems Hosted Design Services", [Online]. Available: http://www.cadence.com/services/hds/Pages/Default.aspx. [Accessed 10 April 2018].

[7] Cadence, "Cadence Design System QuickCycles Service", [Online]. Available: http://www.cadence.com/products/sd/quickcycles/pages/default.aspx. [Accessed 10 April 2018].

[8] DellEMC, "Dell EMC Isilon Scale-Out Network Attached Storage (NAS) ", [Online]. Available: https://www.dellemc.com/en-us/storage/isilon/index.htm#collapse/. [Accessed 7 May 2018].

[9] Mentor, "Eldo Platform", [Online]. Available: https://www.mentor.com/products/ic_nanometer_design/analog-mixed-signal-verification/eldo-platform/. [Accessed 7 May 2018].

[10] Mentor, "Calibre", [Online]. Available: https://www.mentor.com/training/course_categories/calibre/. [Accessed 7 May 2018].

[11] N. Lourenço, A. Canelas, R. Póvoa, R. Martins, N. Horta, "Floorplan-aware analog IC sizing and optimization based on topological constraints", 2015.

[12] R. Martins, N. Lourenço, N. Horta, "LAYGEN II—Automatic Layout Generation of Analog Integrated Circuits", 2013.

[13] N. Lourenço, R. Martins, A Canelas, R. Póvoa, N. Horta, "Layout-aware analog circuit-level sizing with in-loop layout generation", 2016.

[14] R. Martins, N. Lourenço, S. Rodrigues, J. Guilherme, N. Horta, "AIDA: Automated Analog IC Design Flow from Circuit Level to Layout", 2012.

[15] F. Passos, et al. "Handling the Effects of Variability and Layout Parasitics in the Automatic Synthesis of LNAs " in 15th international conference on synthesis, modeling, analysis and simulation methods and applications to circuit design (SMACD 2018), 2018.

[16] R. Póvoa, N. Lourenço, N. Horta, R. Santos-Tavares. J. Goes, "A cascode-free single-stage amplifier using a fully-differential folded voltage-combiner", Electronics, Circuits and Systems (ICECS) 2014.