Remote FPGA Lab For ZYNQ and Virtex-7 Kits

Abd El-Rahman Mohsen[x], Mohamed Youssef GadAlrab[xi], Zeina elhaya Mahmoud[xii], Gameel Alshaer[x], Mahmoud Asy[x] and Hassan Mostafa[x xi]
[x]Electronics and Communications Engineering Department, Cairo University, Giza 12613, Egypt.
[xi]Nanotechnology Department at Zewail City for Science and Technology, Cairo, Egypt.
[xii]Department of Communication and information technology engineering , Zewail City of Science and Technology, Egypt
[xiii]Faculty of Engineering, Alexandria University, Alexandria, Egypt

***Abstract*—This paper proposes a remotely programmable and interactive ZYNQ and Virtex-7 FPGA (Field Programmable Gate Array) Lab for testing and implementing arbitrary hardware circuit designs on real hardware. The online virtual lab facilitates the use of FPGA Boards in simple steps and provides graphical and command line interface to control and monitor FPGA signals in real time. The remote lab provides a scheduling system and allows multiple concurrent remote users. The remote interaction method doesn't depend on the type of the device; so it can be scaled to include different devices. The required hardware and software of the remote laboratory is developed, implemented and tested by the undergraduate and graduate students at Cairo University in Egypt.**

***Index Terms*—FPGA, ZYNQ, Virtex-7, Interactive Learning, Virtual Lab.**

I. Introduction

In the recent decades a new revolution known as Industry 4.0 took place. It takes benefit of the internet of things (IoT) and Cyber-physical systems to connect physical devices with digital world. It is a manufacturing revolution that requires smart means of device prototyping which can be accomplished using FPGAs. FPGAs, also, play a significant role in image processing; which is an essential part of any machine vision system. Most industrial processes require robot arms directed by machine vision systems. So, equipping engineers with FPGA design flow experience becomes a necessity. Universities often provide their students access to hardware laboratories related to their courses; to enhance the learning process, but this is not always applicable, due to the high price of FPGA boards, the capacity of the laboratory, and the time limit of the lab working hours. One of the proposed solutions for this problem is computer simulation. Although, this solution is easy, simulation has some limitations and gets slower as the designs get bigger. Also, simulations do not offer the same experience as the real hardware lab.

FPGA is a type of re-configurable device that is used to test and implement arbitrary hardware circuits. It is, also, used in digital signal processing and in prototyping hardware designs. This paper proposes a remote laboratory; which offers full interaction with real FPGA boards for teaching and testing FPGA system designs. Correspondingly, hardware such as Zynq-7000 SoC ZC702 Evaluation Kit [1] and Virtex-7 FPGA VC709[2] Connectivity Kit are efficiently utilized, moreover, lab users such as undergraduate and graduate students interact with FPGAs in a more efficient and interactive way than ordinary laboratories. The proposed lab allows multiple concurrent users at the same time as long as they use different devices. Each user selects a certain device at a certain time slot from a reservation system and a webserver grants permission over the device to the user at the reserved time slot as seen in Fig.1.

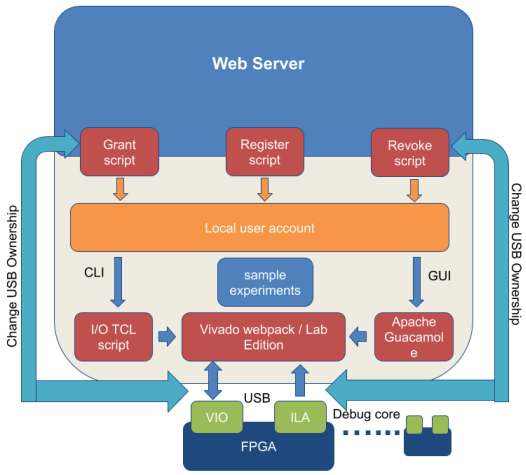

Fig. 1. Architecture of the webserver.

II. Related work

There are many remote FPGA labs that work with the same concept, but with lower FPGA specifications and different implementations[3]. The Online Lab Environment of CoderLabs[4] use physical logic analyzer and Labview[5] to monitor the behavior of the FPGA. Alternatively, the proposed remote Lab takes benefit of VIO (Virtual Input/Output) and ILA (Integrated Logic Analyzer) debug cores[6] to drive and monitor the internal signals in the FPGA design. One advantage of using the ILA Debug core is that it is implemented internally on the FPGA; so, it can monitor very high speed signals without getting delays from external connections or wires. There are other labs that use WINDOWS platform as the server operating system and use remote desktop protocol to forward the display[7][8][9] or build the GUI on the website. Installation of a certain package, or software client is often

required in order to perform an experiment. For example, the work in [10] introduces a remote FPGA lab that is built with a cloud-based Xilinx ISE tool chain and allows uploading Verilog/VHDL codes. Then, a Linux based server connected to the FPGA compiles the code, runs the complete Xilinx design flow, and programs the FPGA. Afterwards, it provides a summary report of the design process. Another paper has proposed a different system that allows users to interact with the FPGAs switches and LEDs (Light Emitting Diodes) through a GUI (graphical user interface), so they can manipulate inputs and test their designs [11].

The proposed remote lab in this paper has used linux as the server operating system, offers full interaction with multiple FPGAs at the same time and implements three different methods of remote interaction. This is done by creating an account for each user locally on the system after registration on a web server, giving permission to the user over the reserved USB (Universal Serial Port) device connected with the FPGA board whether ZYNQ or Virtex-7, running hardware instance of the reserved FPGA device and forwarding the graphical interface of the remote machine. Different GUI forwarding protocols were tested such as VNC (Virtual Network Computing), RDP (Remote Desktop Protocol) and X11. Forwarding the GUI directly to the browser without installing a certain software on the user's side; results in a better experience. Therefore, the user can open and interact with the program GUI on the web browser without installing any clients while the program is running on the server that is connected to the device. At first, XPRA[12] client was used based on X11 protocol for forwarding the display, but it consumed most of the internet bandwidth and resulted in lots of lags and delays. Secondly, Apache Guacamole[13] was used and it showed a better performance as it supports the three protocols mentioned earlier and consumes less internet bandwidth.

III. Lab Setup

The proposed remote virtual lab is composed of one server with Intel(R) Xeon(R) E5-2620 v2 @2.10GHz (24 CPUs) and 32 GB RAM, one Zynq FPGA board, one Virtex-7 FPGA board, and a webcam for each board. The server runs a webpage that allows the user to reserve a time slot to use the lab and interact with the FPGA through it. The server is connected to the FPGA board and runs Vivado software to interface and program the FPGA. Then, the server forwards Vivado GUI session through a webpage or allows the user to SSH (Secure Shell) and run a script that programs and debugs the FPGA. The server, also, runs hardware server provided by Vivado on a certain port and allows remote usage of the FPGA, provided that, the user has the same Vivado version as the one installed on the server.

A. Remote Lab Server Setup

The server runs Django Web framework[14] on Debian jessie[15] operating system and controls the user's access by creating a Linux user credentials for every account on the portal. Then, it locks all accounts to prevent any unauthorized access through SSH. When a time slot starts, the backend server calls a script that assigns the USB (Universal Serial Port) related to the reserved FPGA and unlocks the user account to allow SSH to the server. Another script, also, runs to create a connection for the user in Apache Guacamole[13] that forwards Vivado GUI and passes a certain address to a web-page. Thus, the user interacts with the GUI on that web-page. At the end of the time slot, a custom script runs to terminate all user's opened processes and the account gets locked again.

B. Remote Lab Webpage Setup

The webpage is developed using Django which is a high-level python framework for web development. It is characterized by its security and scalability[14]. The authorized users are only allowed to access the site after validating the date and time entered during reservation. This is performed in multiple steps. Authentication is performed using a bash script that schedules two other scripts to be run on behalf of the user. The first one is issued at the beginning of the session to grant access to the user and the other one is issued at the end of the session to terminate it on the server. At the time the user have access to the server, she/he is able to monitor the FPGA through web streaming camera. The user can also interact with the FPGA through an online terminal.

C. Remote Lab Hardware Setup

Both Zynq-7000 and Virtex-7 FPGAs are connected to the server with the onboard USB JTAG (Joint Test Action Group) interface in boundary scan mode and UART (Universal Asynchronous Receiver Transmitter). This connection not only allows the programming of the FPGA, but also, allows hardware debugging over JTAG. All users are given access to any USB connected FPGA device using Vivado. Limiting user access to the reserved FPGA - not all FPGAs - is done by changing the ownership permission of that USB device at the reserved time slot. In this case, UDEV (Linux dynamic device management)[16] has been used to link each FPGA with a fixed USB device. Input manipulation and output monitoring are facilitated for the user using Hardware debugging with VIO or ILA debug cores provided by Vivado IP catalogue[6]. This makes the remote lab more realistic and closer to real labs. Fig.2 displays the architecture of the proposed remote lab.

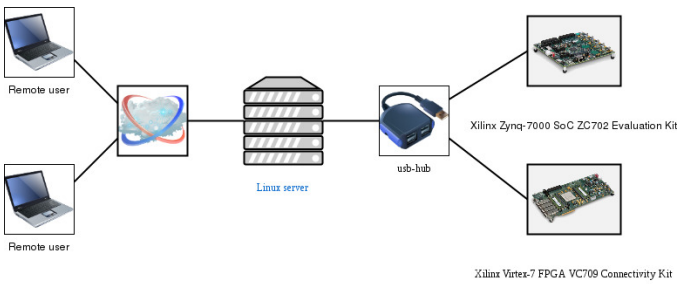

Fig. 2. Client server architecture of the proposed remote lab.

D. Multiuser Experience

The system allows multiple FPGA's to be programmed and used at the same time in three different ways. This is performed in the following manner. Each FPGA has a unique USB device name assigned by Udev rules. At the time of the experiment, the server unlocks the user account and changes the ownership permission of the USB device related to the reserved device at the reserved time slot. Therefore, the user connects to a specific USB device. The user, then, can connect to the FPGA using Vivado hardware server. Vivado hardware server allows connection to the FPGA remotely or from the same machine. Vivado by default runs a single hardware server that connects all FPGAs; allowing any user to connect to that hardware server and all FPGAs. In order to allow users to connect to a specific USB device, the server runs an instance of Vivado hardware server for each user on a different port after reservation; as seen in Fig3.

As for the Graphical interface; the server automatically creates a connection for each user in Apache Guacamole configuration file after signing up. The server, also, creates a display manager for each user on the system along with the local account. One of the advantages of the Linux server is that different users can view different displays at the same time. The proposed architecture uses xfce4 display manager and Xrdp as remote desktop forwarding protocol. Then, the user chooses the method of interaction whether using ssh, Apache Guacamole or Vivado hardware server.

The backbone of the website is the backend service developed which takes advantage of a certain scripts that automates the entire process of accounts creation, reservation, FPGA programming and debugging.

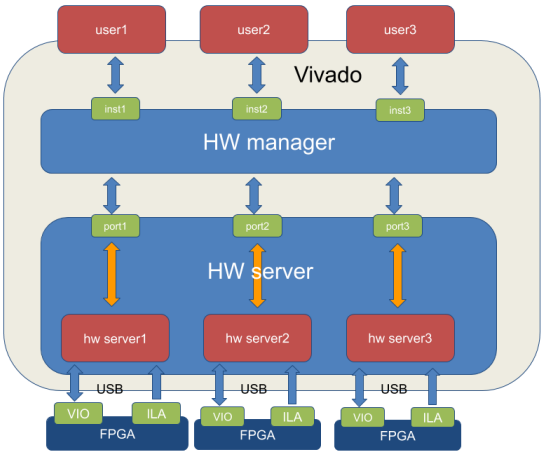

Fig. 3. Multiple Vivado hardware servers for different users.

IV. User Work flow

The FPGA usage is summarized in the following steps: bit-stream generation, FPGA programming and hardware debugging. The first step is assumed to be performed locally on the user local computer. The importance of the remote laboratory comes in the second and third steps. As the user does not have to purchase the expensive FPGA boards, the hardware can be programmed remotely at any time. The first step could also be performed remotely on the server, but it has some hardware and internet bandwidth limitations. For instance, the user cannot perform the first step on the Virtex-7 FPGA board[2] but can perform it on the ZYNQ board[1]. This is because Vivado webpack edition does not support Virtex-7 FPGAs[2] but supports ZYNQ FPGA[1]. There are some considerations to be taken by the user before programming the FPGA remotely. The proper addition of the debug core should be checked by The user. The device constraints such as the clock source and LED pins should also be checked. Fig. 5 shows the command line interface with the camera. This method offers more flexibility as the user can write the code, do the design flow, program the FPGA, debug and manipulate inputs using Virtual Input/Output core (VIO). The typical usage includes:(1) Bit-stream generation; which is done locally on the user computer targeting a certain FPGA. In this step, users must specify the monitored outputs and driven inputs and map these signal to a preferred debug core such as VIO and ILA, (2) Timeslot reservation; at which the server schedules a script that runs the necessary services for the user to program the FPGA at the start time, (3) User interaction; it is done with the server using the online terminal, SSH, Vivado hardware server or XPRA. According to the selected method of interaction, the user uploads FPGA programming files or uses the remote hardware server provided by Vivado, and programs the FPGA, (4) Time slot termination at which the server schedules another script to revoke all user grants and privileges, lock user account and terminate user processes at the end time specified during reservation. The user can emulate behavior of the board's push buttons by adding a VIO debug core then assigning the inputs of his design to this VIO core.

V. Experimental Results

A set of sample experiments and tutorials are provided to a sample of 50 undergraduate and graduate students at Cairo University in video and PDF (Portable Document Format) to facilitate the interaction with the developed remote lab and to get better understanding of its functionality and usage. For example, the first experiment introduced to the user, is a simple AND gate where the user sends the inputs using command line and views the output of the AND gate on the FPGA board LEDs through the mounted camera. In the second experiment, the user remotely controls four LEDs with 2 switches. The third experiment is a 128-bit implementation of the AES (Advanced Encryption System) encryption algorithm. Each of the previous experiments is provided by a user friendly TCL script that programs the FPGA with the experiment corresponding bit-stream and probes file, scans the implementation to get any assigned signals for debugging, checks on the signal type. Based on the signal type, the script whether directs the user to enter the value of the signal if it is an input signal or prints the value of the signal if it is an output signal. The script

Fig. 4. ZC702 Evaluation board connected to a server with webcam mounted on top of it.

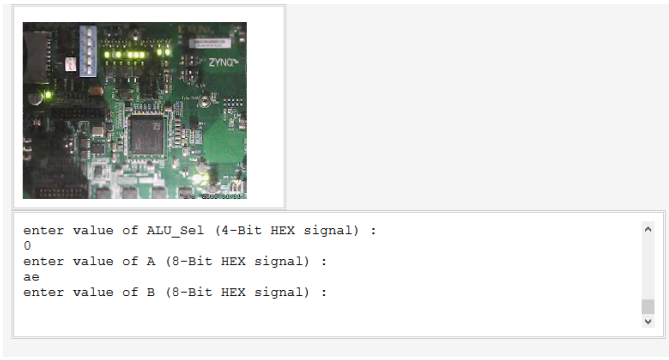

Fig. 5. ALU testing on remote FPGA Lab with results signals being altered on leds.

also checks on the existence of an ILA debug core and if the design contains this debug core then the script asks the user to select the results format (i.e., either VCD (Virtual Circuit Descriptor) or CSV (Comma Separated Value)).

VI. Conclusion and Future work

In this paper, the architecture and implementation of a remote Virtex-7 and ZYNQ FPGA based Lab have been presented. The proposed architecture of the remote FPGA Lab mainly focuses on organizing the FPGA usage and provides the user with complete control over the FPGA. The proposed future work is to add different hardware types such as GPUs (Graphical Processing Units) TPUs (Tensor Processing Units) and generalize the way of remote interaction.

Acknowledgment

This work was partially funded by ONE Lab at Zewail City of Science and Technology and Cairo University and NTRA.

References

1 "Xilinx zynq-7000 soc zc702 evaluation kit." [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html
2 "Xilinx virtex-7 fpga vc709 connectivity kit." [Online]. Available: https://www.xilinx.com/products/boards-and-kits/dk-v7-vc709-g.html
3 Morgan, F., Cawley, S., Callaly, F., Agnew, S., Rocke, P., O'Halloran, M., Drozd, N., Kepa, K., and Mc Ginley, B., "Remote fpga lab with interactive control and visualisation interface," 09 2011, pp. 496–499.
4 Touhafi, A., Braeken, A., Tahiri, A., and Zbakh, M., "Coderlabs: A cloud based platform for real time online labs with user collaboration," in *2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)*, May 2016, pp. 317–324.
5 "Labview national instruments." [Online]. Available: http://www.ni.com/
6 "Ug908 vivado design suite user guide programming and debugging," May 2014. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug908-vivado-programming-debugging.pdf
7 Soares, J. and Lobo, J., "A remote fpga laboratory for digital design students," 01 2011.
8 Rajasekhar, Y., Kritikos, W. V., Schmidt, A. G., and Sass, R., "Teaching fpga system design via a remote laboratory facility," in *2008 International Conference on Field Programmable Logic and Applications*, Sep. 2008, pp. 687–690.
9 Ravanasa, K. and Hashemian, R., "vlab, a high speed multi-accesses parallel processing remote laboratory access for fpga design technology," in *IEEE International Conference on Electro/Information Technology*, June 2014, pp. 377–381.
10 Doshi, J., Patil, P., Dave, Z., Gore, G., Joshi, J., Sonkusare, R., and Rathod, S., "Implementing a cloud based xilinx ise fpga design platform for integrated remote labs," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Aug 2015, pp. 533–537.
11 Soares, J. and Lobo, J., "A remote fpga laboratory for digital design students," 2011.
12 "multi-platform screen and application forwarding system screen for x11." [Online]. Available: http://xpra.org/
13 "clientless remote desktop gateway." [Online]. Available: https://guacamole.apache.org/
14 "Django the web framework for perfectionists with deadlines." [Online]. Available: https://www.djangoproject.com/
15 "Debian 8." [Online]. Available: https://www.debian.org/releases/jessie/
16 "udev dynamic device management." [Online]. Available: https://linux.die.net/man/7/udev