An FPGA-based Remote Laboratory: Implementing Semi-Automatic Experiments in the Hybrid Cloud

Yuichi Toyoda, Nobuhiko Koike, Yamin Li
Graduate School of Computer and Information Sciences, Hosei University, Tokyo, Japan
10k1027@stu.hosei.ac.jp

***Abstract*—An FPGA-based remote laboratory implemented in a hybrid cloud for semi-automatic FPGA-run experiments (SAFRE) is presented. The design file and the FPGA test flow file including the FPGA-run parameters are sent to the hybrid cloud to trigger the sequence of the following tasks automatically: the logic-synthesis, the logic-simulation, and the FPGA-run, with the video recording. In this way, the amount of the users works can be reduced. The user only prepares the high-level Verilog HDL description and a few FPGA device setup parameters. The remote laboratory manager which provides semi-automatic experiment service handles the rest of the works, and the FPGA-run results together with the live recorded video are returned to the user for further inspection/debugging. This semi-automatic experiment service eliminates many cumbersome user works. Each user can have an access to the remote laboratory though a web browser and communicate with the experiment task manager via the Web Services and the WebSockets. As the manager acts like an agent for the user and realizes the semi-automatic experiment, the user can concentrate on the high-level designs. The user can get the feeling of the exclusive and interactive use of an FPGA-based experiment platform, while allowing efficient sharing of the platforms at the same time. In case of the device usage conflicts, the task manager resolves the contentions by reallocating the platforms in both space- and time-division fashions.**

***Keywords*-FPGA remote laboratory; semi-automatic FPGA experiment; Verilog HDL design; hybrid cloud**

I. Introduction

Recently, quite a few FPGA-based remote laboratories have been proposed. Most such systems provide functionality either to simulate experiment environments or to allow exclusive remote use of an actual device through the internet. Also, there are two different ways of the platform usages, namely, the non-real time and the real time.

In the case of the non-real time, it has been successful in efficient use of FPGA development and experiment environment without being conscious of resources such as the FPGA devices [1]. In the real time case, it has realized an interactive but exclusive use of FPGA and the live video recording of the FPGA boards using web camera and virtual remote controller [2]. However, it usually takes long time due to repeated failed design and FPGA-run cycles. In both cases, it results in the waste of time and inefficient platform usages, if the design contains errors. The test automation of FPGA by making use of the vector files [3] becomes an important solution to overcome such shortcomings. Former remote laboratory systems employ actual laboratory machines as remote laboratory platforms as well. However, the limited number of actual platforms has a scalability problem. The use of cloud computing becomes the solution to overcome the problem. However, the FPGA-based experiment environments cannot be migrated to a public cloud solution because the FPGA devices and the licensed synthesis tools are tightly coupled with laboratory machines. Therefore, a hybrid cloud organization which contains the laboratory machines in an on premise private cloud and experiment services in a public cloud, becomes the solution.

This paper proposes an FPGA remote laboratory prototype that can realize the semi-automatic FPGA-run experiments (SAFRE) in the hybrid cloud. This remote laboratory is developed based on Altera DE2-115 board [4], each of which is equipped with a private server. The licensed logic-synthesizing and SAFRE programs have been implemented as the Web Services in the private cloud. In order to ensure scalability, the tasks such as the design in Verilog HDL, the schematic capture, logic-simulation, and the post experiment analysis/debugging have been migrated to the public cloud. As Verilog HDL and FPGA test flow (FTF) files are only transferred among servers in the system, an effective remote laboratory system can be realized in this hybrid cloud. The new feature of the proposed remote laboratory is to automate the tedious test of FPGA in an environment where the resources and devices are shared by users through FTF files and the virtual controls. This semi-automatic mechanism allows the user to check the result at any design stage. Therefore, users can perform experiments of their designs by making use of the proposed FPGA remote laboratory concurrently while automatically resolving the platform usage conflicts.

II. System Considerations

Fig. 1 shows the general overview of the proposed remote laboratory. It realizes the semi-automatic test FPGA experiment environment, where each user can get the feeling of occupying an experiment environment exclusively.

Plural users can perform such experiments concurrently by sharing the laboratory platforms. In order to offload the experiment platforms workloads, the hybrid cloud scheme is employed. Several laboratory platforms have been organized as the on-premise private cloud, where device dependent services and licensed software services should be run in the form of the Web services. The remaining experiment services are offloaded from them and migrated to a public cloud. The SAFRE

978-1-4673-8246-5/16/$31.00 ©2016 IEEE

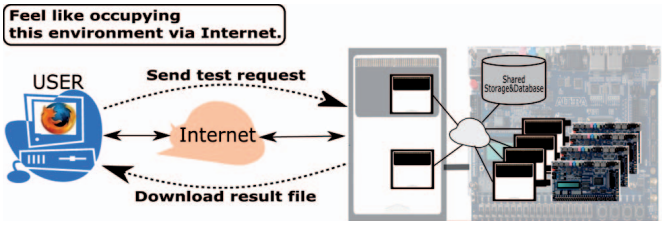

Fig. 1. Basis concept of the proposed FPGA remote laboratory

realizes the test automation by allocating one of the servers in the public cloud and one of laboratory platforms in the on-premise private cloud for each user in both space-division and time-division fashions. Thus a scalable and efficient remote laboratory can be realized.

From the users point of view, the first action is to login the system through the user authorization webpage. After the successful authorization, the user can access the user control webpage by making use of the web browser and the WebSocket protocol. This protocol has been employed to achieve faster two-way communications between the server and the user side computer/tablet. Once the connection has been established, the users requests can always be sent to the same server and obtain the results in return. Also, the server can send information, such as the current status Dash-board, to the user at any time, using the push mechanism.

The first user work is to prepare his/her own high-level designs written in Verilog HDL by making use of either his/her own computer or the server in the public cloud through the remote desk top service. As the designed files are automatically stored in the shared storage/database (SSDB), all servers in the hybrid cloud can share them and unnecessary file transfer can be avoided. After the completion of the design, the user is asked to provide the pin assignment information and the input FTF file information for configuring the FPGA board and experiment environment setup.

The automated services are then initiated according to the given pin assignment and FTF. The working sequence of the services is logic checking/compilation, high-level Verilog HDL logic synthesis, logic simulation, and FPGA-run. Furthermore, the user can initiate to perform differently configured experiments simultaneously, by making use of a number of available laboratory platforms. In case when enough platforms are not available, some experiments can be handled one by one in a platform in a time-division fashion. As the experiment results are automatically stored in the shared database, the user can inspect the experiment results afterwards. The attached live video recordings will give the user a real life experiment atmosphere. Even if the container is closed by the user during the execution, the user can see the results later because the results are in the database.

Fig. 2 shows an overview of this semi-automatic experiment, where a number of FPGA boards in the platform execute separate experiments at the same time in a real-time mode.

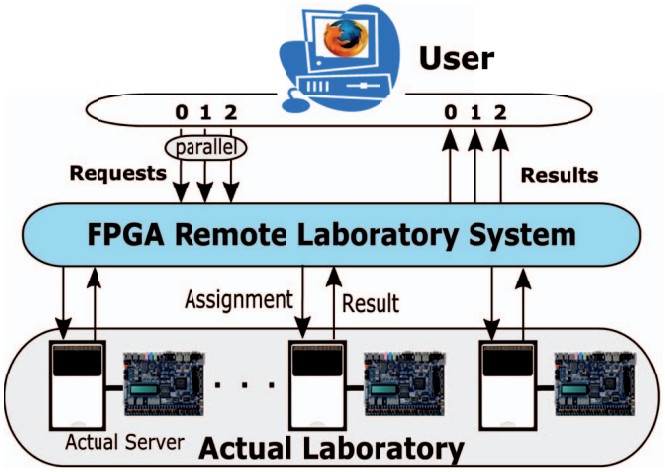

Fig. 2. Automatic concurrent experiments by making use of FPGA boards

As mentioned above, one platform can perform multiple experiments in a non-real-time mode if there are not enough platforms.

III. The FPGA Remote Laboratory Organization

The FPGA remote laboratory is combined with the actual laboratory to give almost the same operating environments to users. The hybrid cloud consists of the public cloud and the private cloud, connected by VPN (virtual private network). Fig. 3 shows the FPGA remote laboratory in a hybrid cloud. The on-premise private cloud is necessary to perform SAFRE works through the web services and SSDB which acts like a glue among servers in the hybrid cloud and user computers to minimize the design/results file transmissions. The remaining works can be handled by the public cloud and the number of servers can be increased or decreased, according to the user usages and seasonal load changes.

In addition, we use the container instead of the Virtual Machine (VM). Because the containers are lighter-weight and more portable than VMs; the bootstrap and shutdown operations are much quicker than VMs. Also, the scale-out time and shrinking time can be shortened. Containers share host machines IP addresses because containers cannot have their own IP addresses for connecting with the external computers across the internet. Fig. 4 shows how the multiple connections can be established with containers in a server through the internet. The containers use port forwarding and separate link ports in the host machine to connect to the external network.

At the logic-synthesis, logic-simulation, and FPGA-run phases, the corresponding web services are issued, and one of the laboratory platforms in the private cloud handles the services and returns the results to the SSDB. The user can download the result files from SSDB and check whether experiments are successfully completed or not, using a web application via a web browser.

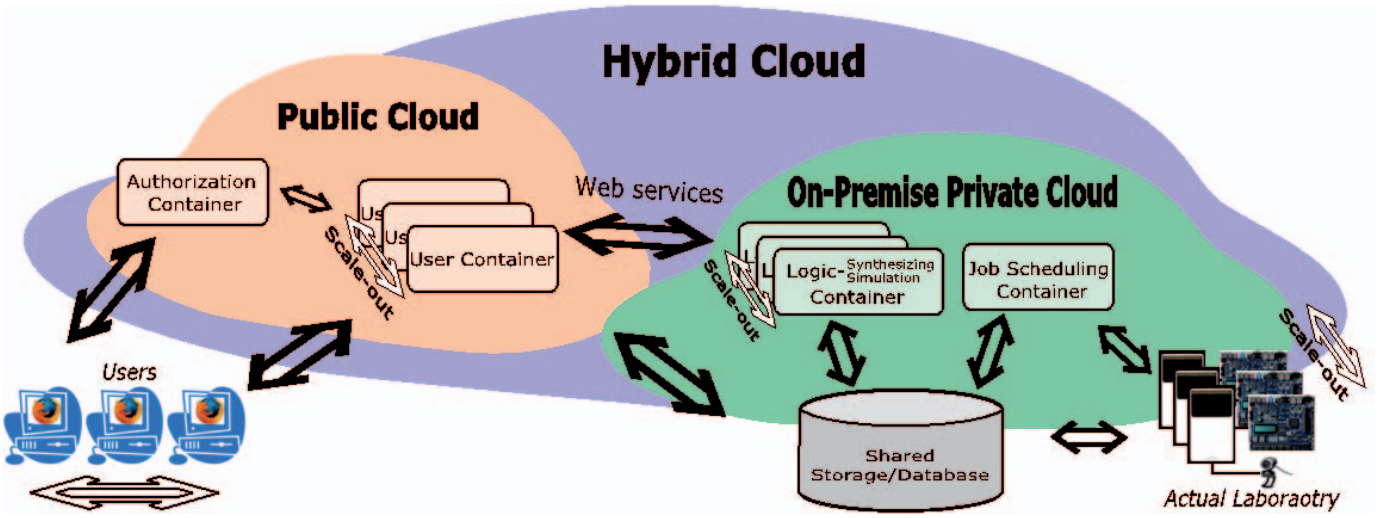

Fig. 3. The overall of the FPGA remote laboratory in a hybrid cloud

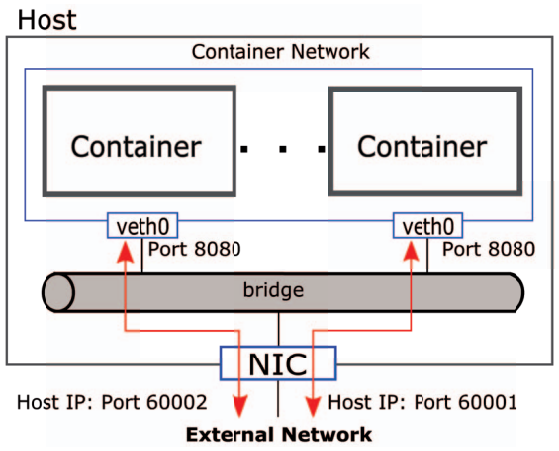

Fig. 4. Containers multiple connections

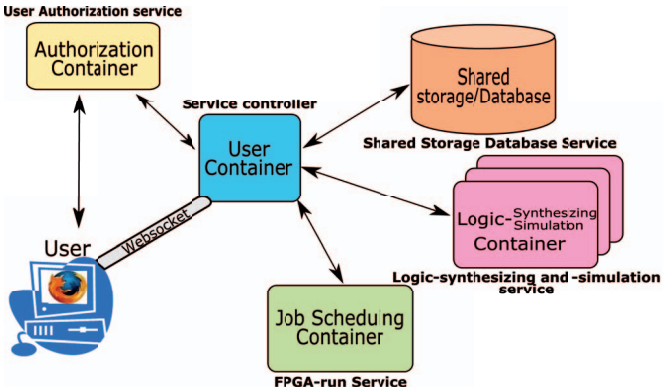

Fig. 5. Some services through User Container

A. Public Cloud for Providing Device Independent Services to Users

The user can upload/download the necessary files to/from the SSDB in the private cloud and utilize device dependent web services (logic-synthesis, logic-simulation, and semi-automatic FPGA-run). These services are built in the private cloud side laboratory servers, accessed via a web browser through the public cloud. First, the user authentication processing is carried out by the manager in the Authorized Container. If the authentication is successful, the User Containers can be started. After a port is opened, the user can be connected to the web services by making use of the WebSocket protocol via the web browser in User Containers. Also, the User Container utilizes the JSON (JavaScript Object Notation) file which contains various requests for those services to upload the user webpages dynamically and interact with other containers.

1) Authorization Container: This container acts as a web server for the user authorization. The user performs the user authorization which checks the user information from the SSDB. If the user authorization is successful, the Authorized Container boots one User Container for the user and opens a separate port for connecting the user to distinguish from other containers ports.

2) User Containers for web application service control: Referring to Fig. 5, this container contains a web server and associated web applications corresponding to these services of the FPGA remote laboratory such as uploading the file to the SSDB, getting error and success messages, checking the Verilog HDL syntax, and issuing the execution request to this laboratory server in the private cloud via web browser which supports the WebSocket protocol. The user can upload/download the necessary files to/from the SSDB and issue the execution request and FTF file to this laboratory through the web application in this container. Also, this container performs simple syntax checking of the user designed file written in Verilog HDL before uploading it to the SSDB. If there is any syntax error in the design file, the user container

does not upload it to the SSDB. Instead, it returns the error message in the form of the web service return value to the user. OSS Icarus Verilog [5] is convenient for Verilog HDL syntax error checking.

B. On-Premise Private Cloud for the FPGA Experiments

In the on-premise private cloud, device dependent services or licensed software services, such as logic-simulation, logic-synthesizing, and SAFRE services, are handled by laboratory servers. The private cloud combines individual actual laboratory, where the following containers handle the FPGA experiments.

1) Logic Synthesizing and Simulation Containers for HDL Compile: This licensed container creates the top module of Verilog HDL which contains the user design for FPGA-run and compiles the top module file with Altera Quartus II Software [6]. When this container receives the request, it downloads the required Verilog HDL files from the SSDB before performing logic-synthesis and logic-simulation. Generated files are then stored back to the SSDB. If the Verilog HDL description contains syntax errors, this container does not upload the design file to the SSDB; it returns the errors to let the user know of the syntax errors. This container converts the FTF file into the Verilog HDL file for the logic-simulation when the user issues the execution request to this laboratory. When the conversion is completed, the container downloads the required design file and performs the logic-simulation using Icarus Verilog. The FTF conversion is straightforward, as the FTF file mainly consists of a sequence of switches and keys signals and can be easily converted into Verilog HDL file for logic-simulation.

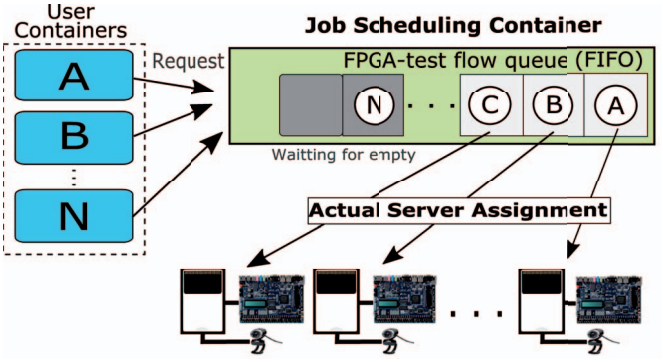

Fig. 6. Job scheduling

2) Job Scheduling Containers (JSC) for allocating actual servers: Referring to Fig. 6, this container manages the multiple actual servers for scheduling FPGA-run services. When a request for the FPGA-run service from User Containers arrived, JSC assigns a job to an idle actual server in the actual laboratory. In case when many users are requesting the same services at the same time, it resolves conflicts among users. If all actual servers are busy, JSC pushes new requests into the job queue and keeps them waiting for the completion of other jobs in actual servers. After one of these servers finished its service, JSC assigns the job at the top of the job queue to the server in a first come first service fashion. Since device dependent services or licensed tool services cannot be executed concurrently within the same server, the JSC allocates these services to available laboratory machines in both space-division and time-division fashions. When the actual server completed the job, the actual server notices JSC of the termination.

C. Shared Storage/Database for Shared Resources of Users

All resources such as the user information, the uploaded Verilog HDL design files, the FTF files, the FPGA-run files, the result videos, and the logic-simulation files (VCD files) are stored in the SSDB. All containers can access the SSDB and upload/download the necessary files. All Verilog HDL files can be syntax error free, as the Verilog HDL compiler resided in the public cloud can check the syntax of the source files before uploading to the SSDB. The SSDB provides the seamless environment, and the FPGA remote laboratory can work actively as the background job.

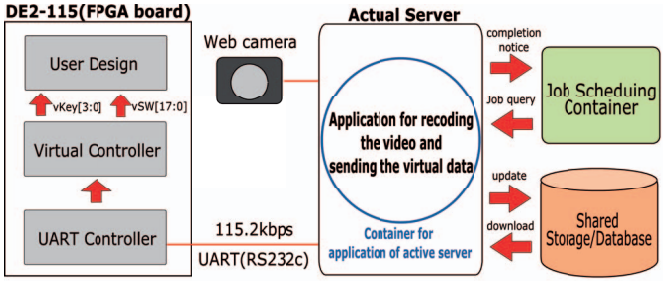

Fig. 7. Actual server organization

IV. Actual Laboratory Organization

The actual laboratory is implemented in the private cloud to realize semi-automatic FPGA-run services. The private cloud contains a number of actual servers. Each server which is managed by JSC to perform the semi-automatic FPGA-run service handles the services as a single node mode. An FPGA-run experiment is realized as the sequence of design automation services specified by the FTF file. The FPGA-run file and FTF file are obtained from the SSDB, and the experiment is carried out according to the FTF file. When the FPGA-run service has completed, the result video file is also written back to the SSDB. Fig. 7 shows the actual server organization.

A. Actual Server Attached to FPGA and Web Camera

As the user cannot control or monitor the remote FPGA device behavior directly, the virtual controller and the web camera have been equipped to realize the remote experiment environment. The virtual controller is useful for mimicking manual key or switch operation to control the FPGA-run. The virtual controller information is sent to the server beforehand and the server communicates with the FPGA via JTAG through UART. The actual server sends the key and switch signal

information during the FPGA-run execution. The web camera records the entire FPGA board behaviors during the FPGA-run service and the recorded video file is stored in the SSDB for further user inspection.

B. Semi-Automatic FPGA-Run Experiment (SAFRE) System

The SAFRE system performs FPGA-run along with the FTF file when the request comes from JSC. The file contains the information such as the operating sequences of keys and push/pull control switches. It controls the operations of FPGA via UART according to the user defined test sequences. Therefore, the user can use the test procedure to mimic manual key operations. In case of the device usage conflicts, the services should be handled sequentially or in parallel by making use of different servers if idle servers are available.

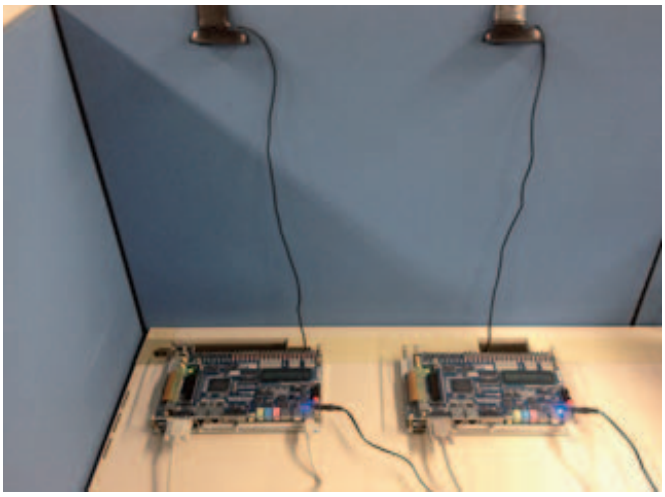

Fig. 8. Actual Laboratory Prototype

V. Preliminary Evaluations

In order to show the effectiveness of the proposed system, a prototype system has been constructed and the preliminary system evaluation has been carried out. As a container in a laboratory server is necessary to work for handling both video recording and FPGA control simultaneously, dual core CPUs are employed for the actual servers. Fig. 8 shows the prototype of the actual laboratory. Fig. 9 shows a page output of the web application for the task, file, and service manager. The user can specify his/her job and get the current status of the job. The user can upload the design file, the FPGA pin assignment specifications, and the test flow file. First, the user uploads the Verilog HDL file. Second, the user creates the FPGA pin setting as shown in Fig. 10 and the FTF file through web applications as shown in Fig. 11. Finally the user selects the task and sends the service requests. If the user has already performed those processes, the user can skip and reuse those results. After the completion of the experiment, the user can download the results through the web applications.

For the preliminary evaluations, series of semi-automatic experiment runs whose duration time for FPGA-runs are fixed

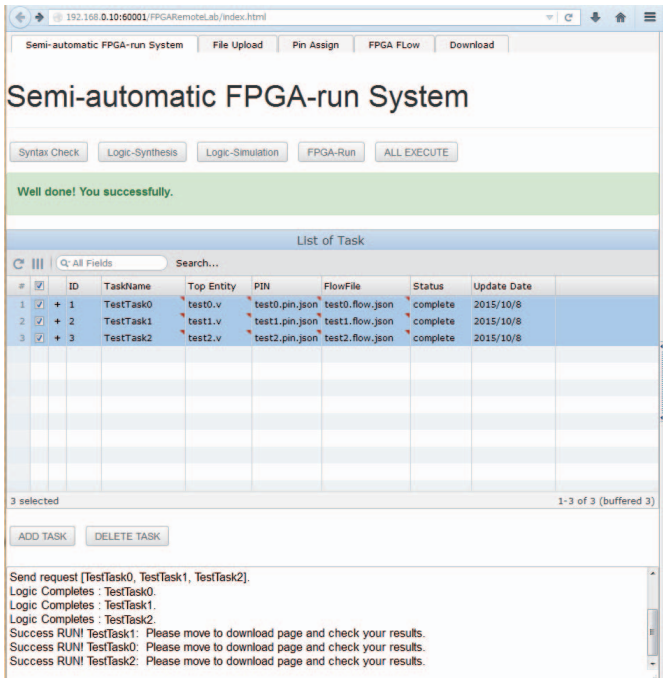

Fig. 9. Web application for controlling the provided services

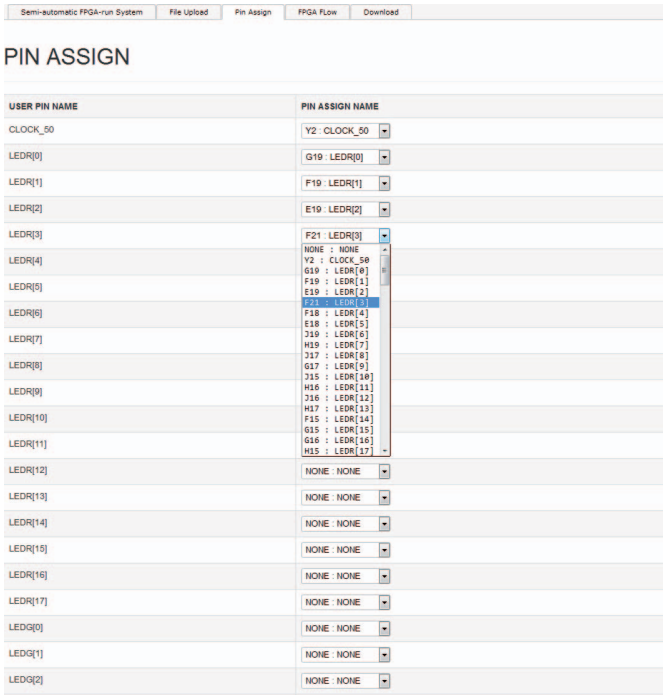

Fig. 10. Web application for pin assign

to five minutes, were carried out. It results in a few seconds excess beyond five minutes at an average, until all services have completed. So, there was no significant increase in the experiment elapsed time compared with the standalone FPGA platform elapsed time for a single user experiment. The experiment time differences between the real and the remote environments can be contained within an average of

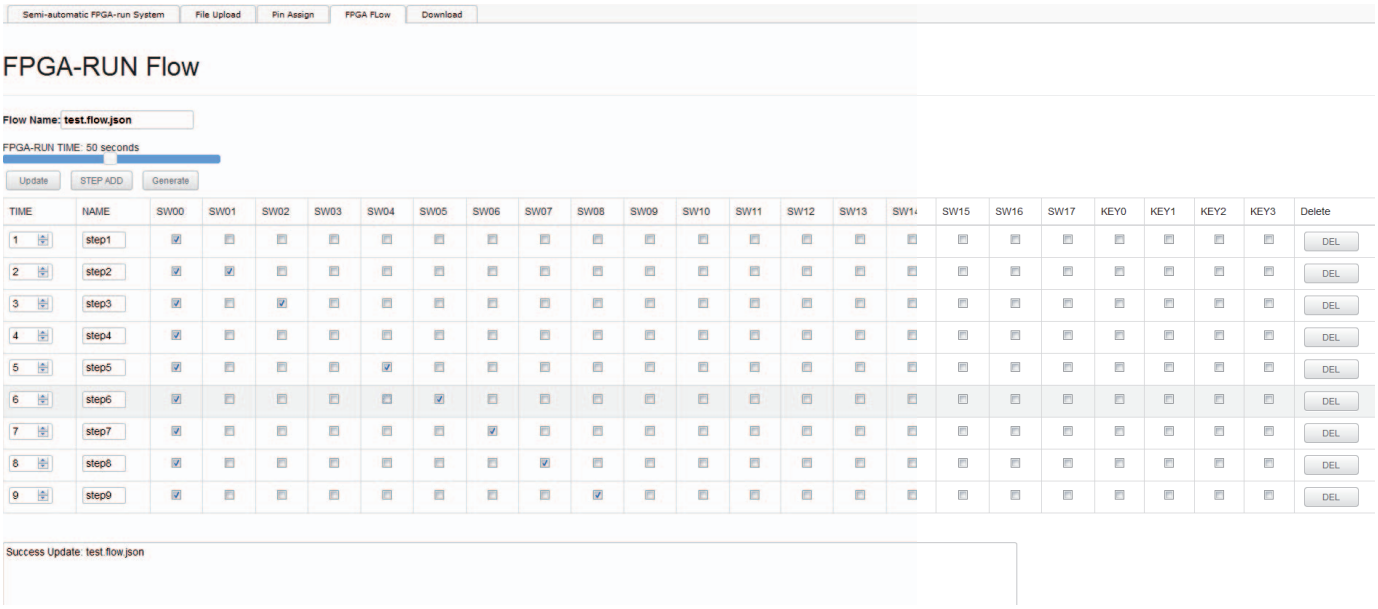

Fig. 11. Web application for creating FTF file

10 seconds for the case of a 5 minutes experiment. We have compared the time which we utilize synthesis tools and have 5 minutes experiment in real with the time which we utilize this remote lab for downloads the experiment result file (mp4) and sending the job. Most delays should come from the containers overhead in the network response and I/O handling. For the actual runs, the elapsed time varies according to the number of concurrent users versus the number of available laboratory platforms. The averaged elapsed time will increase rather mildly according to the number of the concurrent student users, as the actual FPGA-run times for student users are rather short and a number of laboratory platforms can be allocated in both space-division and time-division fashions.

VI. Conclusion and Future Work

The paper described the implementation of the FPGA-based remote laboratory for semi-automatic FPGA-run experiment in the hybrid cloud. The remote laboratory is easy to use because it avoids cumbersome tasks from the user by providing the semi-automatic experiment service.

The prototype system has finished its development, using two PC clusters realizing as the on-premise private cloud and a public cloud to organize as the hybrid could. The remote laboratory can easily be migrated to any other cloud as it is portable. For the actual run, the Amazon Web Service [7] will be employed as the public cloud. As for the private cloud, the OpenStack [8] will be employed. The current implementation assigns the logic-synthesis services to the on-premise laboratory servers. However, the laboratory servers should be overloaded if the number of concurrent users should be increased. It would result in prolonged elapsed time and cause the scalability problem. Migration of the logic synthesis services from the private cloud to the public cloud would be one solution if the security issues and proper licensed tool management issues can be resolved.

We will continue to strive for further test automation and add the real time functionality to actual servers. After the successful prototype evaluation, we have the plan to migrate the experiment public cloud cluster to an actual public cloud, namely the Amazon Web Service (AWS). By combining the existing private cloud (laboratory servers) and the AWS public cloud to be organized as a hybrid cloud, a scalable and efficient remote laboratory can be realized. They should be suitable for academic hardware experiment laboratory use.

References

[1] N. Koike, Cyber laboratory: Migration to the hybrid cloud solution for device dependent hardware experiments, in *Information Technology Based Higher Education and Training (ITHET), 2014*, pp.1-5, 11-13, Sept. 2014

[2] F. Morgan, S. Cawley, M. Kane, A. Coffey, and F. Callaly, Remote FPGA Lab applications, interactive timing diagrams and assessment, in *Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014). 25th IET*, pp.221-226, 26-27 June 2013

[3] J. S. T. Jethra, S. B. Patkar, and S. Datta, Remote Triggered FPGA based Automated System, in *Remote Engineering and Virtual Instrumentation (REV), 2014 11th International Conference on*, pp.309-314, 26-28 Feb. 2014

[4] Altera DE2-115 Board (2015, August) http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=502

[5] Icarus Verilog (accessed on July 30, 2015) http://iverilog.icarus.com/

[6] Altera Quartus II Software (2015, August) https://www.altera.com/products/design-software/fpga-design/quartus-ii/quartus-ii-web-edition.html

[7] Amazon web service (2015, August) https://aws.amazon.com/jp/

[8] OpenStack (2015, August) https://www.openstack.org/