A Novel Remote FPGA Lab Platform Using MCU-based Controller Board

Naoki Fujieda and Atsuki Okuchi
Department of Electrical and Electronics Engineering,
Faculty of Engineering,
Aichi Institute of Technology,
Toyota, Aichi, Japan
nfujieda@aitech.ac.jp (Naoki Fujieda)

***Abstract*—This paper introduces a new remote FPGA lab platform that aims to overcome a problem of existing solutions — difficulty of giving students a feeling of touching hardware. The key component of the proposed platform is an inexpensive controller board that communicates changes of I/O ports with an FPGA board on a remote server. By distributing the controller boards to students, they will virtually *touch* hardware even in the remote lab environment. We describe an overall design and implementations of hardware/software components in this paper. According to our evaluation, the latency of the proposed platform was 72% smaller than a WebCam-based solution.**

***Index Terms*—Remote laboratory, Field programmable gate arrays, Dynamic partial reconfiguration**

I. Introduction

In 2020, the COVID-19 pandemic forced professors and instructors all over the world to switch their classes and labs from in-person to remote [22]. Thanks to advances in information and communication technologies such as an LMS (Leaning Management System), a Web conference system, and video streaming, remote classes have become easy to set up. However, remote labs are still much more difficult, especially when they require hardware equipments.

A major challenge for a remote lab is how to achieve lab experiences equivalent or similar to those in the physical lab. A remote lab platform that overcomes this challenge will be useful even though the physical lab is avaliable, because it enables self-directed learning of students without temporal or spatial limitation. Regarding the learning of digital design and embedded systems, various platforms based on FPGA (Field-Programmable Gate Array) have been proposed to this end [1], [2], [9], [11], [13], [18], [21], [24].

A problem of existing remote FPGA lab platforms is difficulty of giving students a feeling of touching hardware through the handling of remote system. Most of the existing platforms provide virtual access to input ports of the FPGA board through a Web interface. However, this is not enough for some students to understand the difference of digital design from software programming, especially when they are using HDL (Hardware Description Language). Of course, it is not desirable to lend an FPGA board to each student. We could purchase an FPGA module, designed for edge computing, for several tens of USD [17]; however, it has no or very few I/O components for students to handle, such as switches, LEDs, and seven-segment LEDs. An FPGA board suitable for labs costs at least 80 USD [3], [19]. A student might handle their board improperly, which results in an accidental breakdown of their board. If multiple classes share the same physical lab by different lab hours, extra sets of boards have to be prepared. These can be financial burdens.

To overcome this problem, we propose a novel remote FPGA lab platform using an inexpensive controller board based on an MCU (Micro-Controller Unit). The MCU and the remote FPGA communicate command strings with each other. The controller board translates changes of input ports into command strings and sends to the remote FPGA. It receives changes of output ports of the FPGA board as command strings and translates them back. To achieve this, a user circuit has to be combined with a prepared I/O circuit in the remote FPGA, using dynamic partial reconfiguration. We also developed a GUI front-end application to automate this process. In this paper, the proposed platform is described in detail and the latency is evaluated.

II. Related Work

There are two typical approaches to achieve remote labs that require hardware equipments: to simulate the hardware or to provide virtual access to the remote hardware [22]. We briefly review existing environments in the separated subsections.

A. Circuit Simulation

Deeds [5] is a simulation-based learning environment for digital design and MCU. Deeds is composed of a digital circuit simulator Deeds-DcS, a state machine simulator Deeds-FsM, and an MCU emulator Deeds-McE. Using the built-in schematics editor of Deeds-DcS, students can easily construct circuits and see the values of signals. It also provide an integration with the Intel Quartus II EDA (Electronic Design Automation) tool. However, this feature assumes that the EDA tool is installed on the local machine and the student have a specific FPGA board on hand.

To understand how a processor or an MCU works, various educational processor simulators are available. Deeds-McE, introduced in the previous paragraph, is one of them. They usually includes an assembly code editor. Students can write their program and trace the changes of registers and memories instruction by instruction. Some simulators have more complicated visualization features for a pipelined datapath and/or caches [7], [15]. The intended users of these tools are students who are already familiar with digital design and can follow the flow of signals from a block diagram. The proposed platform targets students on a more elementary level.

If an instructor put a higher value on the acquirement of HDL skills than using FPGA boards, an online HDL learning environment would be an attractive option. An IDE (Integrated Development Environment) developed by Kumar et al. [10] has a feature of submitting a circuit, written in VHDL, to a server. The student then receives whether the waveforms of its output signals are correct. VELS [14] is an e-mail-based automated grading system for VHDL, which can give a parameterized assignment for each student. Trenas et al. [20] presented a grading system working on the Moodle LMS, which combined formal checking and functional test. Jelemenska et al. [8] developed another Moodle-based system that assessed the similarity of code blocks. It can detect a circuit that is functionally equivalent but not written in the way specified in the assignment. EDA Playground [6] is a browser-based HDL simulation and synthesis environment. It supports various tools, both commercial and open source. It also includes a waveform viewer called EPWave in order to display the result of circuit simulation on the user's browser. However, since it is not an open source project, a modification tailored to each lab environment is difficult.

B. Remote FPGA Lab

RBoot [2] is the first remote FPGA lab platform in the literature. It provides a pool of sixty-four Xilinx ML-310 boards and a software infrastructure, which reserves and controls the boards for students. It is not described how a student see the behavior of a circuit on the reserved FPGA board. Subsequent platforms offer either of three kinds of measures to observe the output of the boards: using a WebCam [9], [18], via a Web interface that mimics the board [21], [24], and both [1], [11], [13]. Most of them provide access to input ports through a Web interface.

Wan et al. [21] presented a Web-based remote FPGA lab for their computer organization course. Their FPGA board includes two FPGA chips: an experiment FPGA is programmed by a student, while a monitor FPGA controls and observes the I/O ports of the experiment FPGA. A management server is on the public cloud. It authenticates students, allocates the boards to the students, and provides a Web front-end based on Vue.js. A shortcoming of their platform is that custom boards with multiple FPGA chips are required. If a workflow using dynamic partial reconfiguration is available to students, the functionality of these FPGAs can be integrated into a single FPGA.

RemEduLa [1] is a remote FPGA lab with three kinds of commercially available board (ZedBoard, Basys 3, and PYNQ-Z1), fourty-eight boards in total. It adopts an overlay design consists of a prepared base circuit and a user-space circuit. The base circuit includes VIO (virtual I/O) IP cores and a JTAG interface. The VIO can be controlled and observed through a Web interface. The working board can also be seen from a WebCam.

The most important difference of our proposal from these existing platforms is that I/O ports of a remote FPGA board is controlled and monitored by an inexpensive controller board. This will help students have a stronger feeling of touching hardware in a remote lab.

VirtLAB [16] has a similar idea to ours: distributing relatively inexpensive boards to students. The VirtLAB board includes two MCUs and two FPGA chips. One pair of MCU and FPGA is open to students, while the other is for management, measurement, signal generation, and so on. The goal of VirtLAB is to minimize the unit cost of electronics lab equipment by integrating expensive instruments, such as a digital oscilloscope and a waveform generator, into a single board. In the proposed platform, a remote board costing hundreds of USD is controlled by a local board costing tens of USD. Briefly speaking, the assumed cost ranges differ by about one order of magnitude.

III. System Design

A. Overview

Figure 1 depicts the overview of our remote FPGA lab. There, a student anywhere in our campus has a controller board and his own or shared PC, while each of the FPGA boards is connected to a development VM (Virtual Machine). The both boards are recognized as USB devices and communicate via the UART (Universal Asynchronous Receiver Transmitter) protocol. The baud rate is set to 115,200 bps. Since we use Digilent Nexys A7 [4] boards that include Xilinx Artix-7 FPGA for our lab, Vivado is installed to the development

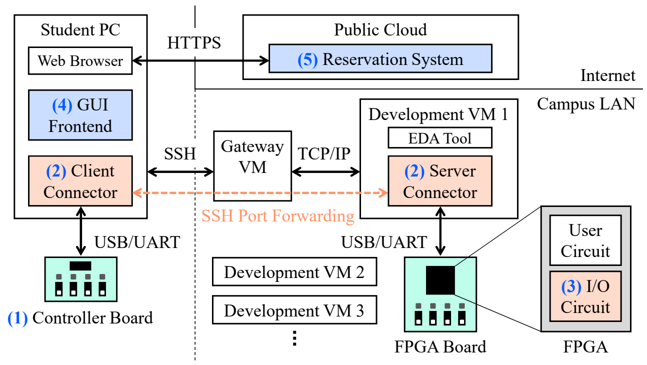


Fig. 1. Overview of the proposed remote FPGA lab.

TABLE I
DEFINITION OF CHARACTERS OF COMMAND STRINGS.

Character	Definition
0–3	Select a digit of the seven-segment LED.
4	Select the array of LEDs.
A–H/a–h	Turn on/off the corresponding segment of the selected LEDs.
I–P	Select the corresponding slide switch.
Q–S	Select the corresponding tactile switch.
U/u	Turn on/off the selected switch.
V followed by X	Request for board-specific response.
V followed by Z	Request to resend all of the values of LEDs or switches.

VMs. The student connects to the gateway VM via SSH and accesses to one of the development VMs using port forwarding. Currently we set up five development VMs. All of the VMs, including the gateway, are built on a single physical server of Ubuntu 20.04 LTS. Although the security policy of our institute does not allow SSH access from the Internet, technically the proposed platform can be open to the Internet.

Table I summarizes the definition of the commands. The controller board has eight LEDs, a 4-digit seven-segment LED, eight slide switches, and three tactile switches. Each of the changes of these I/O components is translated into a command string, which consists of two ASCII characters. The changes of switches are sent from the controller board, while the changes of LEDs are sent from the FPGA board. Two special commands are also defined to let the student PC or the development VM recognize the board and refresh its state.

We developed the following five components, which will be described later in the separated subsections of Section IV:

1) a controller board connected to the student PC,
2) client and server editions of connector apps that recognize the boards and forward command strings to the other side,
3) an I/O circuit on the FPGA side for translation between changes of I/O ports and command strings,
4) a GUI front-end on the student PC to automate the development process, and
5) a cloud-based reservation system.

```
open_checkpoint $checkpoint_base
read_checkpoint -cell [get_cells DR]
                $checkpoint_proj
opt_design
place_design
route_design
write_bitstream -force ${project_name}.bit
close_project
```

Fig. 2. The main part of the script of bitstream generation.

B. Students' workflow

In the proposed remote lab, a student first prepares files required by the development VM in the local PC. They writes their own circuit in VHDL, either by hand or using a schematics editor like Deeds-DcS [5]. They then start up the GUI front-end and choose a folder of the VHDL codes. The front-end recognizes the top entity of the student's circuit and lists its I/O ports. The student assigns the ports to I/O components of the FPGA board using GUI interface. After that, they make the front-end generate required files, including a wrapper of the user circuit and three Tcl scripts. All of the VHDL files, generated scripts, and the prepared checkpoint file of the I/O circuit are uploaded using a SCP/SFTP client before using the development VM. In parallel, a student makes a reservation of a desired time slot of a development VM by accessing the reservation system.

When the beginning of the reserved time slot comes, the student runs the `ssh` command to log in the gateway VM, adding two `-L` options for port forwarding. One port is used by RDP (Remote Desktop Protocol) to launch Vivado in the remote desktop. The other port is used by the connector apps to control the FPGA board.

After launching Vivado in the remote desktop, the student synthesizes their circuit, generates a bitstream file, and programs the FPGA using the Tcl scripts obtained from the front-end. The task of each of the scripts is as follows.

- The first script opens or creates a Vivado project that contains all of the VHDL files. The student then synthesizes the user circuit and saves the synthesis result as a checkpoint file. The student can also simulate the circuits if they have a testbench.
- The second script reads the checkpoint files of both the user circuit and the I/O citcuit and implements them. Figure 2 extracts the main part of the second script. This follows a tool flow of dynamic partial reconfiguration, called DFX (Dynamic Function eXchange) by Xilinx [23].
- The last script opens a hardware manager screen of Vivado. The student then specifies the generated bitstream file and programs the FPGA.

Finally, the student starts up the client edition of the connector app. They specify the COM port number of the controller board and the TCP port number to connect the server edition of the app. When the both ports are connected, the student will be able to turn on and off the switches of the

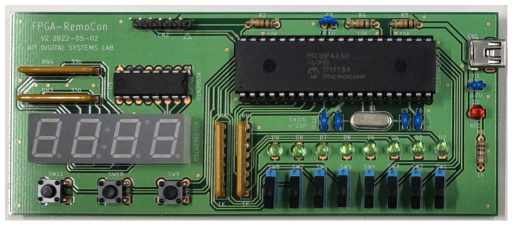


Fig. 3. Appearance of the controller board.

TABLE II
MAIN COMPONENTS OF THE CONTROLLER BOARD. PRICES ARE IN USD.

Part	Unit price	Qty	Subtotal
PIC18F4450 MCU	5.4500	1	5.45
Slide switch	0.6023	8	4.82
Seven-segment LED	2.2704	1	2.27
Mini USB connector	0.9019	1	0.90
Green LED	0.0744	8	0.60
ULN2003A Transistor Array	0.5213	1	0.52
Others			2.20
Total			16.76

FPGA board and see the changes of the LEDs virtually, using the controller board.

IV. Implementation of Components

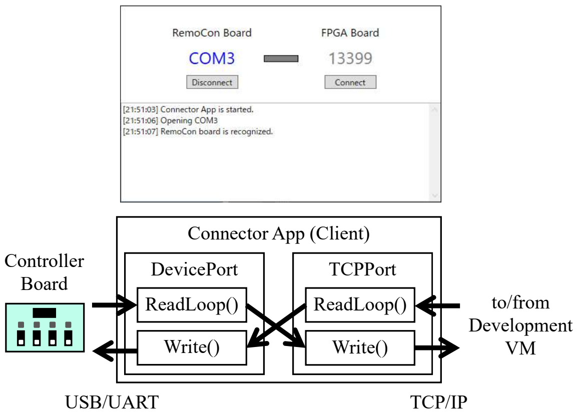
A. Controller Board

Figure 3 depicts the appearance of the controller board. The dimension of the board is 130 $\times$ 55 mm. The board includes a Microchip PIC18F4450 MCU [12], which incorporates a USB 2.0 module. Using the USB library from the Microchip Libraries for Applications, the MCU operates as a USB CDC (Communications Device Class) device, which means an external USB-UART converter such as FTDI FT232 is not required. The MCU has thirty-three user I/O ports. We assigned two for USB (D+ and D-), eleven for the switches, eight for the LEDs, and twelve for the seven-segment LED. We preferred through-hole components for ease of implementation of prototypes by hand.

Table II summarizes the price of the main components of the controller board. We assumed that we would purchase components required for 100 boards from Digi-Key Electronics in June, 2023. Note that board manufacturing and mounting costs are not included in the table. The cost of components for each board in this assumption is less than 20 USD, excluding tax. However, the total unit cost in our first test production became around 65 USD, including tax. This was mainly because the mounting cost of through-hole components was high. Our next prototype would have as many surface-mount components as possible to reduce the cost, though it is left for future work.

B. Connector Apps

Figure 4 abstracts the client edition of the connector app, written in C# and based on WPF (Windows Presentation Foundation). A student can select the COM and TCP port numbers by clicking the corresponding labels. If the board is successfully recognized after clicking the connect button, the



```
[21:51:03] Connector App is started.
[21:51:06] Opening COM3
[21:51:07] RemoCon board is recognized.
```

Fig. 4. Appearance and internal of the client edition of the connector app.

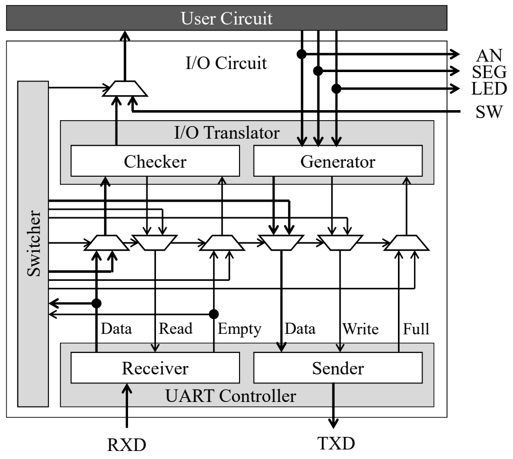


Fig. 5. Block diagram of the I/O circuit.

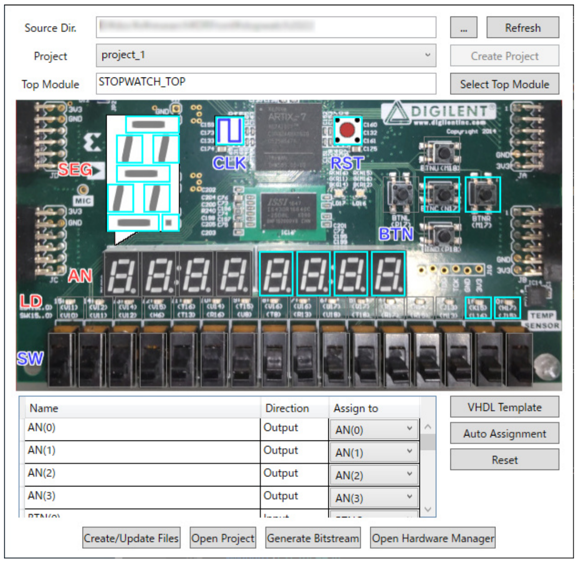
color of the corresponding label becomes blue. If both of the boards are recognized (i.e., the controller board is working), the color of the bar in the middle also becomes blue.

Management of the serial port to the controller board and the TCP connection to the development VM is done by the respective classes, DevicePort and TCPPort. The both classes have methods of the same names: `Open()`, `Close()`, `ReadLoop()`, and `Write()`. The `ReadLoop()` methods are executed in dedicated threads. The read characters are passed to the `Write()` method of the other class, only when the both boards are recognized.

The server edition of the connector app does similar tasks to the client edition. The differences between the editions are twofold. First, the server edition is written in Python and executed as a daemon process. Second, the TCPPort class behaves as a TCP server, instead of a client.

C. I/O Circuit

Figure 5 describes the block diagram of the I/O circuit. The circuit has three major components: a UART controller, a switcher, an I/O translator. The UART controller provides a simple FIFO-based interface. The output signals of the user circuit are also connected to output components of the board.


Source Dir.

Project: project_1

Top Module: STOPWATCH_TOP

Name	Direction	Assign to
AN(0)	Output	AN(0)
AN(1)	Output	AN(1)
AN(2)	Output	AN(2)
AN(3)	Output	AN(3)

Fig. 6. Appearance of the front-end (the source directory location is blurred).

The switcher monitors the received characters and see if a request command ("VX") is sent from the server. Before receiving the command, the switcher simply connects the switches of the board to the user circuit. This functionality is required to let the user circuit work as if the I/O circuit is not present, when the board is not connected to the server. If the request command comes, the I/O translator starts working.

The checker part stores the current state of the switches and monitors the command strings. If it receives a character to turn a switch on or off ('U' or 'u'), it sets the state of the selected switch to '1' or '0', respectively.

The generator part stores the state of the LEDs that the controller board is supposed to know. If it differs from the current state of the LEDs of the user circuit, the generator part picks one of the different segment, generates the corresponding command string, and updates the state of that segment.

The generator also includes sampler circuits. The goals of the samplers are threefold. The first goal is to reduce the frequency of the commands being sent, considering the very limited bandwidth of UART and processing speed of the MCU. The frequency of the sampling signal is set to 100 Hz and 500 Hz for the seven-segment LED and the LED array, respectively. The second goal is to make the output of the seven-segment LED stable, which is dynamically driven. The sampler for the seven-segment LED tracks its appearance. No additional commands are sent as long as the appearance remains unchanged. The last goal is to roughly reproduce the PWM (pulse width modulation) signals of LEDs. Depending on the sampling period, simple sampling might turn PWM signals into unintended ones. To avoid this, the sampler for the LEDs has a thresholding circuit, which determines the final output based on the time ratio of outputting '1' in a period.

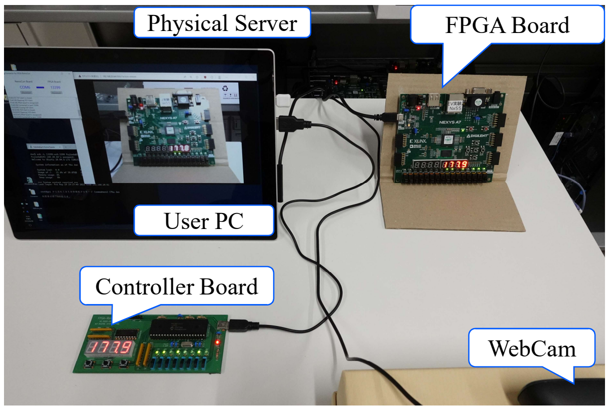


Fig. 7. Equipments used in our evaluation.

D. GUI Front-end

Figure 6 depicts the appearance of the GUI front-end, after the I/O ports are assigned and the bitstream file is generated. The front-end is also written in C# and based on WPF. At the center of the window, the picture of the bottom half of the board is shown and each of the components that can be assigned a signal is shown with a gray frame. When a student assign a signal by selecting a component with a drop-down list or performing drag and drop, the color of the corresponding frame becomes cyan.

Generating files for Vivado and launching Vivado can be done with the buttons in the bottom row. Since Vivado is installed in the students' PCs of our physical lab, they can continue using the front-end there, until programming the FPGA.

E. Reservation System

In addition, we developed a simple reservation system with a combination of Forms, Excel, and Power Automate of the Microsoft 365 cloud service. A student can make a reservation by posting an answer at the reservation form. The posted answers are automatically added to a table in an Excel workbook by a cloud flow of Power Automate. The latest reservations for each student are extracted, and the first student who reserved each time slot is displayed in a status sheet of the workbook.

V. Evaluation

At the time of writing this paper, we have not yet received feedbacks from lending our controller board in our labs. Meanwhile, generally one of the most important concerns for a network-based system is latency: in our case, students may find difficulties on using the system if the latency is too high. We made an experiment, presented in this section, to compare the latency of the proposed system with a WebCam-based solution,

A. Methodology

Figure 7 shows a view of our evaluation experiment. There are five kinds of equipments: a physical server, FPGA boards,

TABLE III
EVALUATION RESULTS OF LATENCY (IN SECONDS).

	FPGA	Controller		WebCam	
#	Time	Time	Latency	Time	Latency
1	122.24	122.22	0.02	122.13	0.11
2	128.92	128.90	0.02	128.80	0.12
3	135.89	135.87	0.02	135.80	0.09
4	141.76	141.74	0.02	141.66	0.10
5	148.59	148.57	0.02	148.46	0.13
6	154.91	154.80	0.11	154.80	0.11
7	160.30	160.29	0.01	160.20	0.10
8	165.67	165.66	0.01	165.56	0.11
9	170.91	170.80	0.11	170.80	0.11
10	177.95	177.93	0.02	177.80	0.15
Avg.			**0.036**		**0.113**

a user PC, a controller board, and a WebCam. The FPGA boards and the WebCam are connected to the server, while the controller board is plugged into the user PC. The controller board reflects the state of the LED array and the seven-segment LED using the proposed system, depicted in Fig. 1. For comparison, the server streams a video from the WebCam using the *mjpg-streamer* software. The video is sent to the user PC and displayed in a Web browser. The server is connected via Ethernet while the user PC is connected via Wi-Fi. The ping value between them was 1.6 millseconds on average.

The FPGA board is programmed to work as a stopwatch that measures elapsed time in 1/100 seconds. Digits in the 100 seconds through 1/10 seconds places are displayed on the seven-segment LED. Digits in the 1/10 and 1/100 seconds places are shown in the left and right halves of the LED array, respectively. These digits are described in gray code to reduce read error. The FPGA board, the controller board, and the monitor of the user PC (i.e., the video sent from the WebCam) are photographed at a time to compare the latency. We took ten pictures and read the elapsed times from them.

B. Result

Table III tabulates the times read from the pictures. The columns with *Latency* are calculated from the time difference from the FPGA board. The proposed platform achieved 36 milliseconds of latency on average, which was 72% smaller than a WebCam-based solution. Students will have a more timely experience with the proposed platform.

In the most cases, the latency of the proposed platform was 20 milliseconds. The possible causes of the latency are the sampler circuits, the network, UART, and the processing of the MCU. The latency from each of the first three causes are estimated as up to several milliseconds. This leads to an inference that most of the latency comes from the MCU.

However, in two cases (i.e, #6 and #9), large latency of 110 milliseconds was observed. We think the actual latency was 10 milliseconds but we failed to read the digit of 1/10 seconds places as 9. It is probably due to the thresholding circuit in the samplers. When an LED turns off and another turns on at the same time, it is possible that the both signals do not reach the threshold. This might lead both LEDs on the controller board to turning off for a short period. We leave it for future work to conduct another evaluation to measure the latency of the proposed platform in a more precise, reliable way.

VI. CONCLUSION

In this paper, we presented a new remote FPGA lab platform where the I/O ports of the FPGA board were controlled and observed by an inexpensive controller board. Starting from the spring semester in 2023, we are lending our controller boards to some of the students and they are enjoying their remote FPGA lab. Making evaluations in educational context through these experiences, including feedbacks from the students, is left for future work.

Based on the developed platform, many ways of extension will be possible. For example, if we modified the I/O circuit to use VIO cores and a JTAG interface (as RemEduLa did), we might release the USB/UART port for the user circuit. It might be interesting if the controller board had an I/O component that was not present in the target FPGA board. Of course, incremental improvements to resolve dissatisfactions from students are also important. We will incorporate such improvements into future versions of our platform.

ACKNOWLEGMENT

The first author personally thanks Prof. Kenji Kise and Dr. Takefumi Miyoshi for giving us know-hows of operating remote development servers.

REFERENCES

[1] C. Blochwitz *et al.*, “RemEduLa – Remote Education Laboratory for FPGA Design Technology,” in *55th IEEE International Symposium on Circuits and Systems*, 2022, pp. 1773–1777.
[2] K. Datta and R. Sass, “RBoot: Software Infrastructure for a Remote FPGA Laboratory,” in *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2007, pp. 343–344.
[3] Digilent. Academic Price list. [Online]. Available: https://digilent.com/shop/academic/academic-price-list/
[4] ——. Nexys A7 Reference Manual. [Online]. Available: https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual
[5] G. Donzellini and D. Ponta, “From gates to FPGA: Learning digital design with Deeds,” in *3rd Interdisciplinary Engineering Design Education Conference*, 2013, pp. 41–48.
[6] Doulos. EDA Playground. [Online]. Available: https://edaplayground.com/
[7] R. Giorgi and G. Mariotti, “WebRISC-V: a Web-Based Education-Oriented RISC-V Pipeline Simulation Environment,” in *Workshop on Computer Architecure Education held in conjunction with ISCA ’19*, 2019, pp. 3:1–3:6.
[8] K. Jelemenska, P. Cicak, and M. Gazik, “VHDL models e-assessment in Moodle environment,” in *14th International Conference on Emerging eLearning Technologies and Applications*, 2016, pp. 141–146.
[9] T. Kodama, Y. Suzuki, and S. Chiba, “Development of a remote practice system for embedded system education,” in *6th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, 2010, pp. 53–58.
[10] A. Kumar, R. C. Panicker, and A. Kassim, “Enhancing VHDL Learning through a Light-weight Integrated Environment for Development and Automated Checking,” in *2nd IEEE International Conference on Teaching, Assessment and Learning for Engineering*, 2013, pp. 570–575.
[11] C. A. Mayoz *et al.*, “FPGA remote laboratory: experience of a shared laboratory between UPNA and UNIFESP,” in *14th Technologies Applied to Electronics Teaching Conference*, 2020, pp. 1–8.
[12] Microchip Technology Inc., *PIC18F2450/4450 Data Sheet*, DS39760D, 2008.
[13] F. Morgan, S. Cawley, and D. Newell, “Remote FPGA Lab for Enhancing Learning of Digital Systems,” *ACM Transactions on Reconfigurable Technology and Systems*, vol. 5, no. 3, pp. 18:1–18:13, 2012.

[14] M. Mosbeck, D. Hauer, and A. Jantsch, “VELS: VHDL E-Learning System for Automatic Generation and Evaluation of Per-Student Randomized Assignments,” in *4th IEEE Nordic Circuits and Systems Conference*, 2018, pp. 1–7.

[15] M. B. Petersen, “Ripes: A Visual Computer Architecture Simulator,” in *Workshop on Computer Architecure Education held in conjunction with ISCA ’21*, 2021, pp. 5:1–5:8.

[16] M. R. Roch and M. Martina, “VirtLAB: A Low-Cost Platform for Electronics Lab Experiments,” *Sensors*, vol. 22, no. 13, pp. 4840:1–4840:18, 2022.

[17] Seeed Technology. Spartan Edge Accelerator Board. [Online]. Available: https://www.seeedstudio.com/Spartan-Edge-Accelerator-Board-p-4261.html

[18] J. Soares and J. Lobo, “A Remote FPGA Laboratory for Digital Design Students,” in *7th Portuguese Meeting on Reconfigurable Systems*, 2011, pp. 95–98.

[19] Terasic Inc. DE10-Lite Board. [Online]. Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1021

[20] M. A. Trenas, J. Ramos, E. D. Guitérrez, S. Romero, and F. Corbera, “Use of a New Moodle Module for Improving the Teaching of a Basic Course on Computer Architecture,” *IEEE Transactions on Education*, vol. 54, no. 2, pp. 222–228, 2011.

[21] H. Wan, K. Liu, J. Lin, and X. Gao, “A Web-based Remote FPGA Laboratory for Computer Organization Course,” in *29th ACM Great Lakes Symposium on VLSI*, 2019, pp. 243–248.

[22] X. Wang *et al.*, “A Survey on the E-learning platforms used during COVID-19,” in *11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference*, 2020, pp. 808–814.

[23] Xilinx Inc., *Vivado Design Suite User Guide: Dynamic Function eXchange*, User Guide UG909 v2020.2, 2022.

[24] Y. Zhang *et al.*, “Remote FPGA lab platform for computer system curriculum,” in *ACM Turing 50th Celebration Conference*, 2017, pp. 3:1–3:6.