International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 2, April 2024, pp. 2344~2360
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i2.pp2344-2360

Remote field-programmable gate array laboratory for signal acquisition and design verification

Rithea Sum, Watcharapan Suwansantisuk, Pinit Kumhom
Department of Electronic and Telecommunication Engineering, Faculty of Engineering, King Mongkut's University of Technology Thonburi, Bangkok, Thailand

Article Info

Article history:

Received Aug 27, 2023
Revised Dec 9, 2023
Accepted Jan 5, 2024

Keywords:

Data compression
Effective debugging
Embedded systems
Internal signal acquisition
Remote field-programmable gate array laboratory

ABSTRACT

A remote laboratory utilizing field-programmable gate array (FPGA) technologies enhances students' learning experience anywhere and anytime in embedded system design. Existing remote laboratories prioritize hardware access and visual feedback for observing board behavior after programming, neglecting comprehensive debugging tools to resolve errors that require internal signal acquisition. This paper proposes a novel remote embedded-system design approach targeting FPGA technologies that are fully interactive via a web-based platform. Our solution provides FPGA board access and debugging capabilities beyond the visual feedback provided by existing remote laboratories. We implemented a lab module that allows users to seamlessly incorporate into their FPGA design. The module minimizes hardware resource utilization while enabling the acquisition of a large number of data samples from the signal during the experiments by adaptively compressing the signal prior to data transmission. The results demonstrate an average compression ratio of 2.90 across three benchmark signals, indicating efficient signal acquisition and effective debugging and analysis. This method allows users to acquire more data samples than conventional methods. The proposed lab allows students to remotely test and debug their designs, bridging the gap between theory and practice in embedded system design.

Corresponding Author:

Watcharapan Suwansantisuk
Department of Electronic and Telecommunication Engineering, King Mongkut's University of Technology Thonburi
126 Pracha Uthit Road, Thung Khru, Bangkok 10140, Thailand
Email: watcharapan.suw@kmutt.ac.th

1. INTRODUCTION

Embedded systems have become an integral part of our daily lives and profoundly affect our interactions with technology [1]. A critical component of embedded systems is the field-programmable gate array (FPGA), which allows complex digital designs to be implemented in hardware. FPGAs are used in consumer electronics [2], automobiles [3], medical devices [4], and the internet of things (IoT) [5], among others. Understanding how to use FPGA technology is essential for students and engineers in the field of embedded system design [6], [7] and requires users to interact physically with targeted FPGA boards. FPGA laboratories are indispensable learning assets that provide students and engineers with hands-on experience with FPGA technology.

Remote FPGA laboratories have emerged as elegant solutions for students and engineers seeking to develop their skills in embedded system design [8], [9]. While traditional FPGA laboratories are confined to classrooms during class periods, remote FPGA laboratories allow users to program, debug, and test FPGA

boards remotely anytime and anywhere with internet access. Remote FPGA laboratories alleviate the drawbacks of traditional laboratories, such as limited accessibility, high operating costs, stringent time constraints [10], and forced closures during pandemics [11]–[13]. Given their advantages, remote FPGA laboratories have gained popularity, and can complement or replace traditional laboratories [14].

Previous studies have proposed and implemented remote FPGA laboratories [10]–[13], [15]–[23] with mixed success. The desirability and acceptability criteria of the remote laboratory design are placed on low resource utilization on the FPGA boards, the ability of remote users to debug the program conveniently, the ability of users to acquire a large amount of FPGA board's data, and a simple user interface. The user interface designs of existing remote laboratories follow two common approaches to provide the FPGA board's output to remote users: live stream and direct capture of the internal signals. In the first approach, the remote user views the results and evaluates their design through a live stream of the FPGA board by using an internet protocol camera (IP camera) or webcam. For example, existing remote FPGA laboratories [21]–[23] offer live streams and basic FPGA experiments that include controlling the status of light-emitting diodes (LEDs), seven-segment displays, and switches on the target FPGA board. These experiments were evaluated through a user interface that incorporated virtual inputs (switches) and outputs (LEDs, seven-segment) directly into the web application. In the second approach, the initial signals of an FPGA are captured and transmitted to remote users by either i) probing the FPGA's output pin connected to an external logic analyzer (LA) or ii) using integrated logic analyzer (ILA) debug cores [24] within the FPGA board itself. For example, existing remote FPGA laboratories [12], [18] employ external logic analyzers to virtualize internal signals. However, the device configuration is performed locally and the display of the signal is constrained by the size of the logic analyzer window. Mohsen *et al.* [17] proposed a method that allows users to configure a sample size based on the ILA configuration and hence, more data to be acquired. In a remote lab, where the users cannot directly access the lab tools, acquiring more samples of data enriches the students' learning experience and simplifies the debugging process at the expense of a large amount of data being transmitted over a network.

To address the large amount of data, some existing studies chose to transmit uncompressed data, whereas others compressed the data before transmission. The benefits of not compressing the acquired data are a simple FPGA and low utilization of the FPGA board. The drawback is the limited amount of data that can be acquired by the end users at a given time. On the other hand, the benefit of data compression includes the acquisition of large samples, which allows FPGA experiments to become easier and faster for debugging and analysis. Common compression techniques that have been implemented in FPGA include the Huffman [25], [26], dictionary-based [27]–[31], and deflate-based [32], [33] encoders. However, these techniques require large amounts of hardware resources. Given the drawbacks and advantages, as well as the desirability and acceptability criteria, there is no clear distinction between no compression and compression. Compression is beneficial when the acquired data contains repetitions. However, when the acquired data are not repetitive, transmitting raw data without compression is preferred owing to the low FPGA utilization.

Although much work has been done toward the implementation of remote FPGA laboratories for signal acquisition, state-of-the-art methods are fundamentally limited. The entire process of acquiring the internal signal from the FPGA is not entirely web-based and requires the installation of additional software tools on the client side to perform the experiments. However, this requirement may be inconvenient for certain users. Furthermore, existing studies either compressed FPGA-acquired data or did not. A better approach is for a remote laboratory component to automatically and adaptively decide whether the acquired data should be compressed before transmission or not. A remote laboratory that is entirely web-based and adaptively compresses the acquired data will enhance users' learning experiences and extend a traditional laboratory beyond a physical classroom.

In this paper, we present the architecture of a remote FPGA laboratory platform that allows the users to remotely program their FPGA boards, and offers various features to test their design entirely on a web-based interface. To acquire a large internal signal, we adopt the concept of an add-on module [10], [21]–[23], [34] and implement a lab module with adaptive run-length-encoding data compression for users to incorporate into their design. Additionally, the concept of using a pre-built module allows users to build their FPGA design and connect the signal of their design that they want to capture to the module. The output of this module can be sent to the server using a standard communication protocol such as a universal asynchronous receiver transmitter (UART), inter-integrated circuit (I2C), or serial peripheral interface (SPI). The main contributions of this study are as follows: i) a remote laboratory architecture that enables users to remotely program an FPGA board and verify its design, ii) a method that is efficient for internal signal acquisition for FPGA remote laboratory, and iii) performance evaluation of the proposed laboratory in terms of available features, compression ratio, and hardware resource utilization. The proposed remote laboratory meets the desirability and acceptability design criteria that improve the distance-learning environment.

The remainder of this paper is organized as follows. Section 2 presents the overall architectural design and the implementation of the proposed FPGA remote laboratory. Section 3 focuses on the hardware

implementation of the two proposed modules and the proposed data compression algorithm for efficient signal acquisition. The performance evaluation and discussion of the proposed work are presented in section 4, followed by conclusions and future research directions in section 5.

2. ARCHITECTURE OF THE PROPOSED LAB

The objective of this study was to provide a method for students and engineers to remotely program their own FPGA boards and explore various interactive features to test their designs. These features include receiving data from and transmitting data to the board as well as live streaming of the board itself. The management of the lab on load balancing, task queuing, and lab session management was previously discussed in [35]–[37] and falls outside the scope of this study.

The proposed remote FPGA lab comprises a Linux-based PC server with medium specifications, several FPGA boards, and one webcam per board, as shown in Figure 1. A web-based interface is hosted on the server for users to conduct experiments without the need for additional software on client sites. Two modes of communication are available for users to connect with the proposed platform: those on the university campus can connect via the campus network, whereas off-campus users use the university's virtual private network (VPN) to access the platform. Furthermore, communication between the laboratory server and FPGA board was accomplished through a universal serial bus (USB) to program the FPGA board and serial communication to transmit data to and receive data from the experimental board during the experiments. The proposed remote FPGA lab is scalable and can serve any user via an internet connection.

The implementation of the laboratory server is the most crucial part of the proposed remote FPGA laboratory. The laboratory server comprises three sub-servers that include a web-based server, a core server, and a live-streaming server. The web-based server is designed to provide web-based applications to users, whereas the core server is responsible for executing every experiment-related task, such as moving the generated bitstream (.bit) file from the user to program the FPGA, real-time interaction with the programmed experimental board, and saving the data from the FPGA. The live-streaming server is responsible for capturing a live view of the experimental instance and displaying it on a web-based application. Next, we describe the design of the three sub-servers in detail.

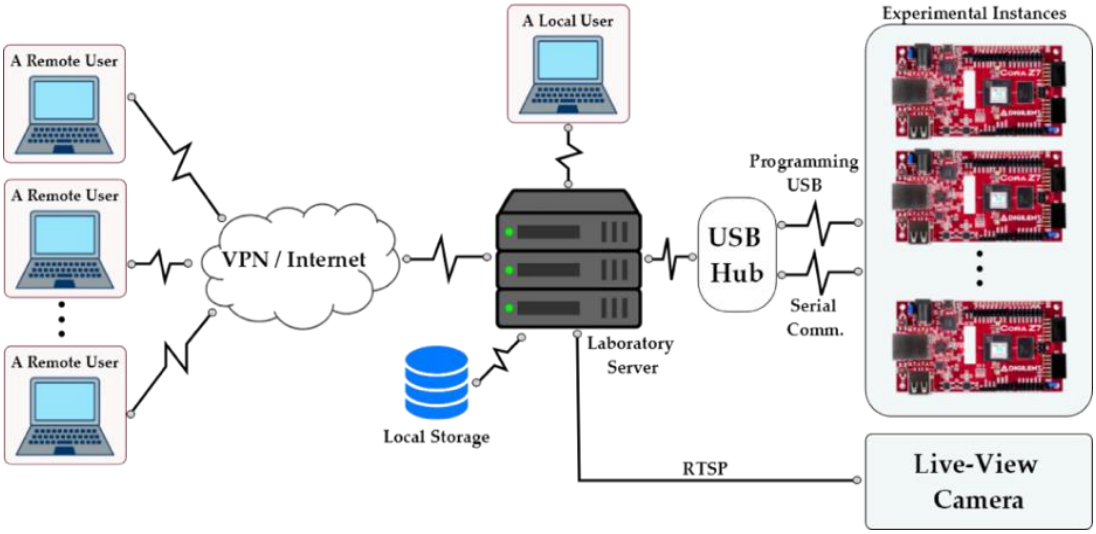


Figure 1. The architecture of the proposed FPGA remote laboratory consists of the laboratory server, experimental instances, and live-view camera system

2.1. Core server

The server in this layer is built with ExpressJS, a popular web framework for building server-side applications with Node.js, and is capable of building a representational state transfer (RESTful) application programming interface (API). This method allows for standardized communication and data exchange over the internet in an efficient and structured manner. After building the APIs, users can make requests to the server through hypertext transfer protocol (HTTP) methods. These requests are then processed by the core server, which communicates with the FPGA board to perform the requested actions.

The connection between the experimental instance and the laboratory server consists of USB cables for programming the FPGA, and a serial communication interface for interacting with the board after it is programmed. To program the FPGA board, we utilized the tool command language (TCL) console of the Xilinx Vivado tool, which can load the design file in *.bit* format in the FPGA. After receiving the design file from the user, the core server generates a TCL script to program the FPGA.

Regarding the serial communication interface for interacting with the FPGA board during the experiments, multiple communication protocols were available, but we chose UART for ease of development. Python code with various packages was used for the software aspect of the interaction. When a command is received by the web-based server on the server side, the script transmits input signals to the FPGA board if the command is to send input. Conversely, it will receive data from the FPGA if the command is to acquire a signal.

2.2. Web-based server

A web-based application built upon the ReactJS framework was used to serve users on tasks such as uploading a bitstream file, streaming a real-time video of the experimental instance, and providing a web console to show the status of the experiment. Currently, the system supports two types of remote FPGA laboratories, namely, basic FPGA laboratory and advanced FPGA laboratory.

a. Basic FPGA laboratory refers to a remote laboratory that requires real-time visual feedback. These labs typically consist of experiments that use output hardware peripherals to display results. As shown in Figure 2, on the developed basic FPGA laboratory page of the web-based application, the user can upload the generated bitstream file after designing it in Vivado and programming the embedded device remotely. Subsequently, a live-stream view appears on the top-right side of the interface, and the status of the entire experimental process can be observed in the *console.log* terminal at the bottom of the interface. Additionally, an option is provided that allows the user to send an input signal to the embedded device.

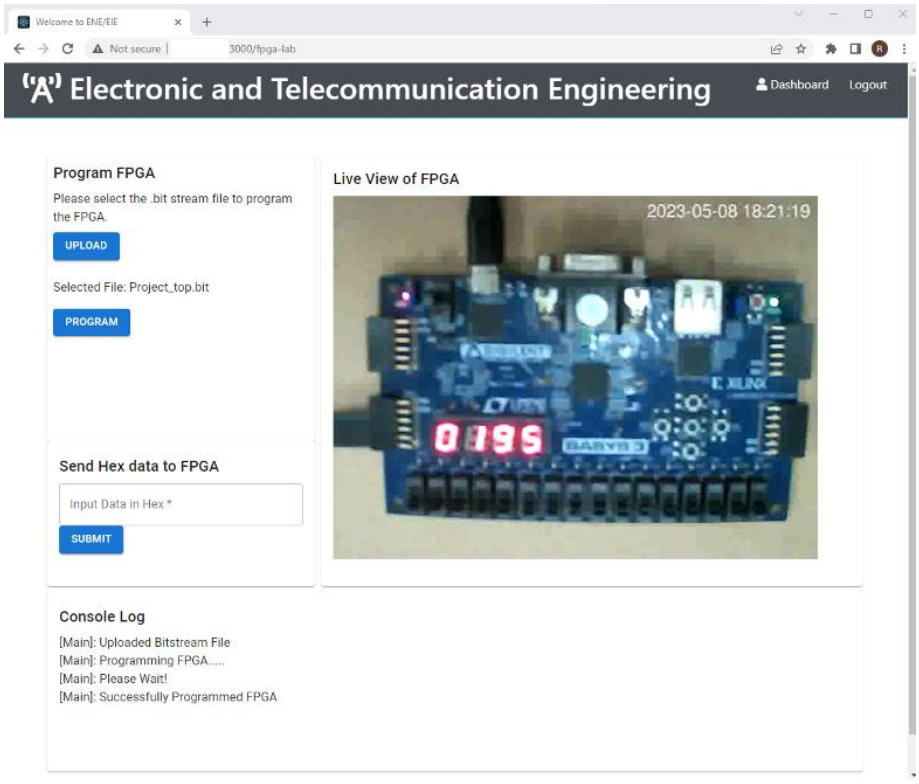


Figure 2. The interactive user interface of the proposed basic remote FPGA laboratory is web-based, enabling users to program the FPGA board with their bitstream file and observe the results through a live-view camera

b. Advanced FPGA laboratory is a remote laboratory that involves experimenting with complex FPGA designs. In addition to the features provided in the basic FPGA laboratory, the advanced FPGA laboratory allows users to capture internal signals within the FPGA during the experiment. This feature provides

users with deep insight into the functioning of their designs. Additionally, users can interact with the programmed FPGA and modify the design behavior in real-time. The laboratory user interface is shown in Figure 3.

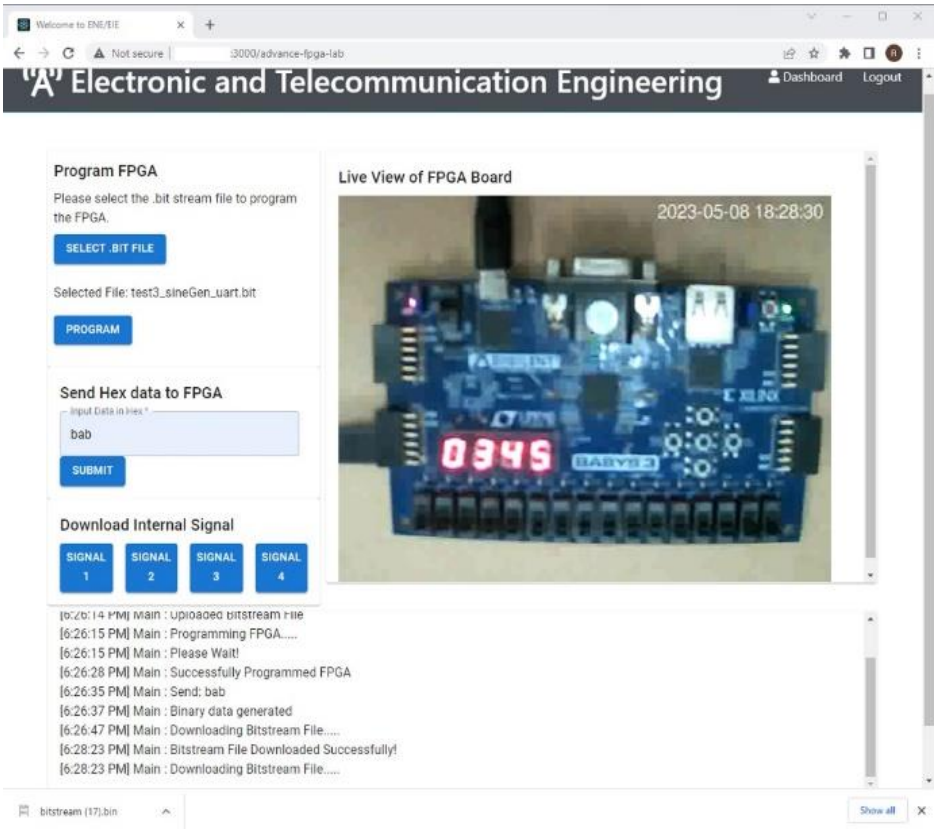


Figure 3. The interactive user interface of the proposed advanced remote FPGA laboratory enables users to capture the internal signal during experiments

2.3. Live-streaming server

A live-streaming server was deployed in parallel with the web-based server and laboratory server to capture the live view of the experimental instance during the experiment. The goal of streaming is to provide a minimum delay in the captured image to the web-based application. To implement the live-streaming server, several technologies were utilized, including a steaming protocol, a specific transmission control protocol/internet protocol (TCP/IP) port, HTTP live streaming (HLS) conversion, and an HTTP server for streaming the file. With these components, it is possible to offer a high-quality live-streaming experience to users.

Each experimental instance utilized a webcam to capture the visual feedback. The initial step involves streaming the webcam video to a TCP/IP port using a real-time streaming protocol (RTSP). Subsequently, the data stream was converted to a suitable format for HLS to enable playback on various web browsers. Once the stream is converted to the HLS format, an HTTP server is employed to host the generated file, allowing the web-based server to play the stream for users on the web-based application.

3. THE PROPOSED METHOD FOR A REMOTE FPGA LABORATORY

In this section, we present the hardware implementation of the two proposed modules that users can incorporate into their design to perform the basic FPGA laboratory and the advanced FPGA laboratory. In the proposed laboratory, users do not need to incorporate the lab module into their design if the design verification involves visual feedback but does not involve transmitting data to and receiving data from the FPGA. We then present an efficient data compression method for signal acquisition in a remote FPGA laboratory.

3.1. Hardware implementation of the lab modules

Users who wish to work on a project involving interaction with the FPGA board after programming are advised to incorporate their designs with our pre-made modules, namely, the *Basic_Remote_Lab* and *Advanced_Remote_Lab* modules. These modules were developed in very high-speed integrated circuit hardware description language (VHDL) and serve slightly different purposes. The *Basic_Remote_Lab* module enables the FPGA board to receive data transmitted from the server computer via UART, whereas the *Advanced_Remote_Lab* module allows both data transmission and reception via UART.

3.1.1. The *Basic_Remote_Lab* module

Figure 4 displays the *Basic_Remote_Lab* module designed at the register-transfer level (RTL). This block receives the input signal from the web server, and its output is an 8-bit hexadecimal representation of the American standard code for information interchange (ASCII) code corresponding to the input data provided by the user. Then, by incorporating this module into the user design, the output signal can be utilized to perform various configurations, such as displaying LED patterns and seven-segment numbers connected to the FPGA board.

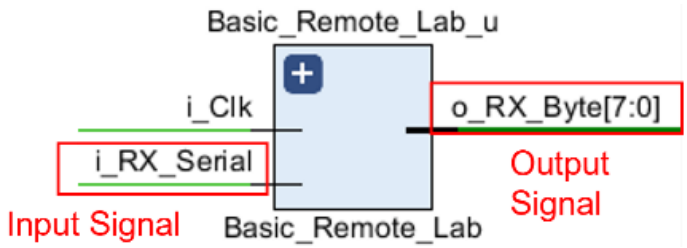


Figure 4. The RTL design schematic of the *Basic_Remote_Lab* module enables user-design integration for signal transmission to the FPGA during experiments

A use case for this module is proposed to demonstrate its practical application. Figure 5(a) shows a scenario in which a student designs their own modules, namely *DispScan* and *timer1us*, which are incorporated with the *Basic_Remote_Lab* module to display the received input signal from the user on a seven-segment display. Figure 5(b) illustrates the entire process of conducting this experiment using a web-based interface. The process begins by uploading the bitstream file, which is then loaded into the FPGA. The user can continuously send input data to the FPGA board and observe the corresponding display on the seven-segment display located on the right-hand side of the interface.

3.1.2. The *Advanced_Remote_Lab* module

The *Advanced_Remote_Lab* module, which is also designed at the RTL, is proposed and illustrated in Figure 6. The designed module allows users to capture signals from their designs during experiments with high data throughput and allows their design to take the input data from the UART. Our current implementation allowed the user to capture four signals with a default data size of 16 bits. This module is provided for users who use the system to perform complex RTL designs and wish to acquire signals for further analysis or debugging purposes.

The proposed hardware implementation of the *Advanced_Remote_Lab* can be implemented on various FPGA boards. The module consists of three sub-module designs, which include:

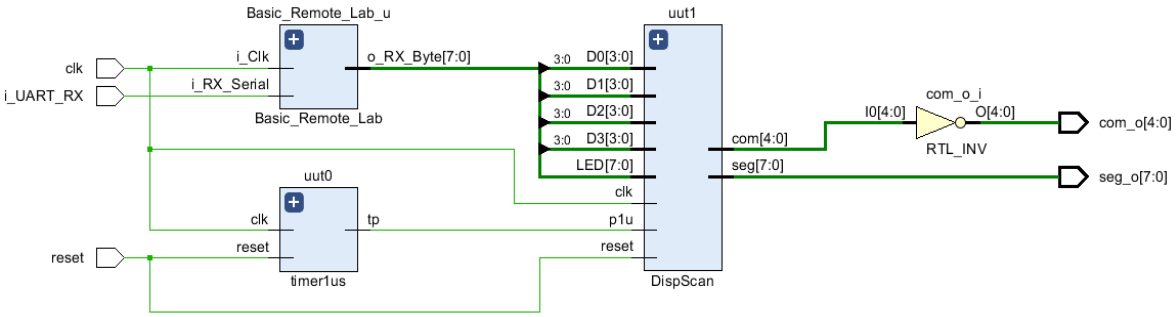
- Signal capturing module (*Signal_Capturing_x*) allows users to input a signal that they want to capture during the remote experiment. The size of the desired signal can be configured in this block via the configuration of the first-in, first-out (FIFO) generator intellectual property (IP) [38].
- Control unit (CU) controls the data moments in this module. The decision-making process of the module is based on the configuration command provided by the user through the web-based application. It reads the configuration command sent from the user and performs a function based on the provided command.
- UART module handles communication between the FPGA and the laboratory server. It receives configuration commands from the laboratory server for the CU to process and respond to by providing an internal signal when requested.

To demonstrate the use of this module, a project involving the generation of a sinusoidal waveform was assigned to students during the course of the embedded system. The designed code required 1048 clock cycles to generate each data sample. The complete RTL design depicted in Figure 7(a) illustrates how the output of the signal generator module is connected to one of the inputs of the *Advanced_Remote_Lab*

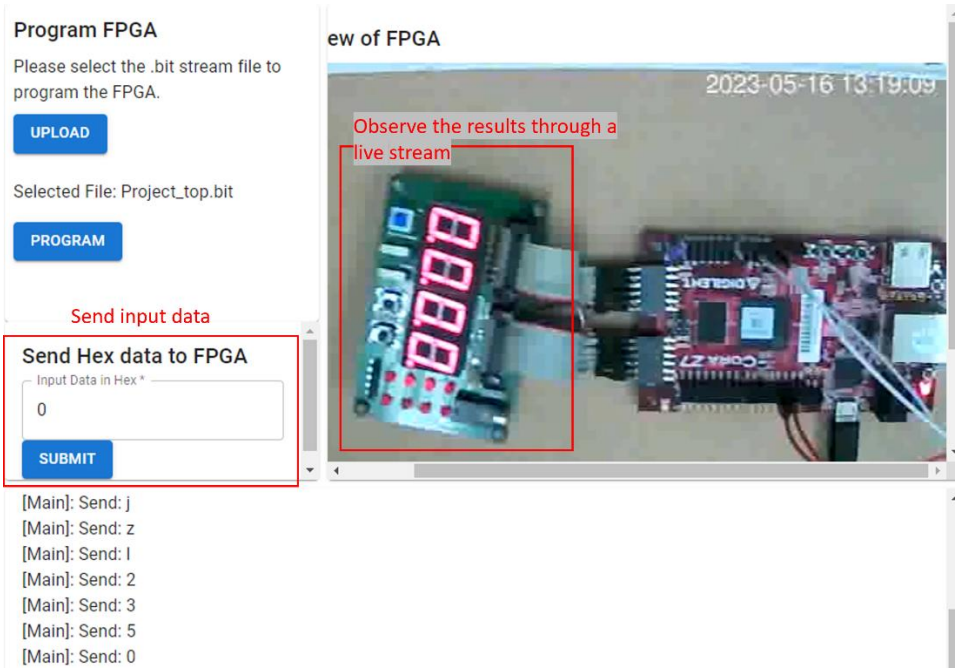
module. The testing process for this design can be initiated once the bitstream file generation is completed. The user can then load the bitstream file into the web application and program the FPGA board. Once the FPGA board is successfully programmed, the user must send a configuration signal to the board to enable the *Advanced_Remote_Lab* module to capture the signal and transmit it to the core server. When the core server receives the data, the user can click on the download option to retrieve the signal connected to the *Advanced_Remote_Lab* module that can be seen in Figure 7(b).

In this experiment, a sinusoidal waveform was captured using the *Advanced_Remote_Lab* module. Without pre-compressing the internal signal before storing it in a buffer, the total acquired data amount to 16,384 samples, which is equivalent to the number of acquired samples using an ILA to capture the signal. Figure 8 shows a plot of the acquired data. The analysis of the acquired data indicated the presence of only 16 unique signal levels. Consequently, the required buffer size was 32,768 bytes, with each data point represented in a 16-bit unsigned integer (uint16) format. Therefore, sending the entire data packet to the laboratory server is inefficient.

To address this issue, we checked whether signal duplication had occurred in the acquired signal. If duplication occurred, run-length encoding was executed within the FPGA before transmitting data to the core server. This adaptive compression method significantly reduces the amount of duplicated data when performing FPGA design, thereby improving the overall transmission performance of the proposed remote laboratory.



(a)



(b)

Figure 5. Users can (a) incorporate their own modules into the existing one, for example, an RTL design that was incorporated into the *Basic_Remote_Lab* module, and (b) verify their designs experimentally using a web-based interface that displays the design verification process in the basic FPGA laboratory

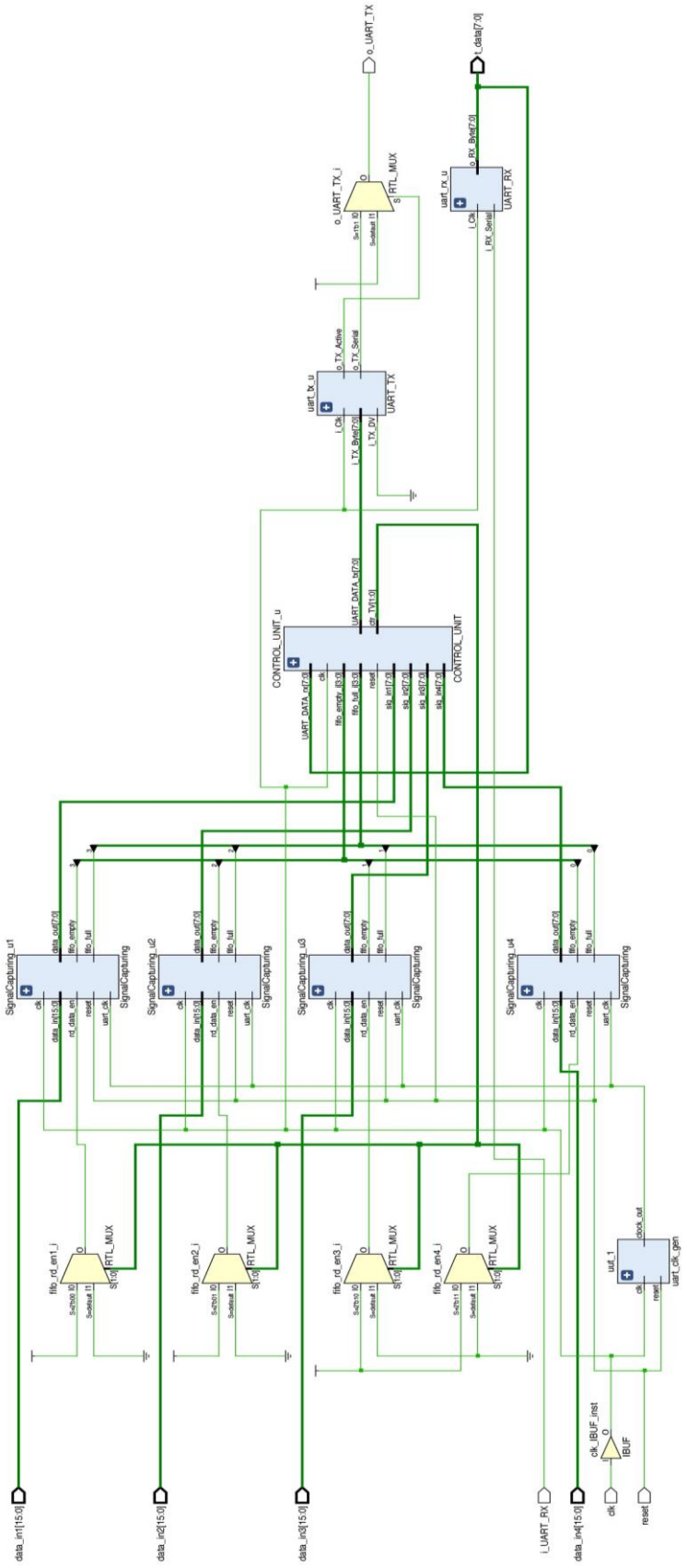
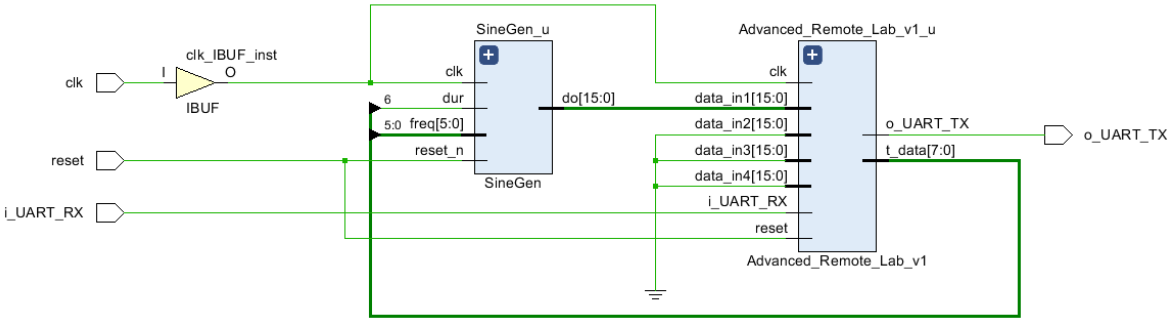
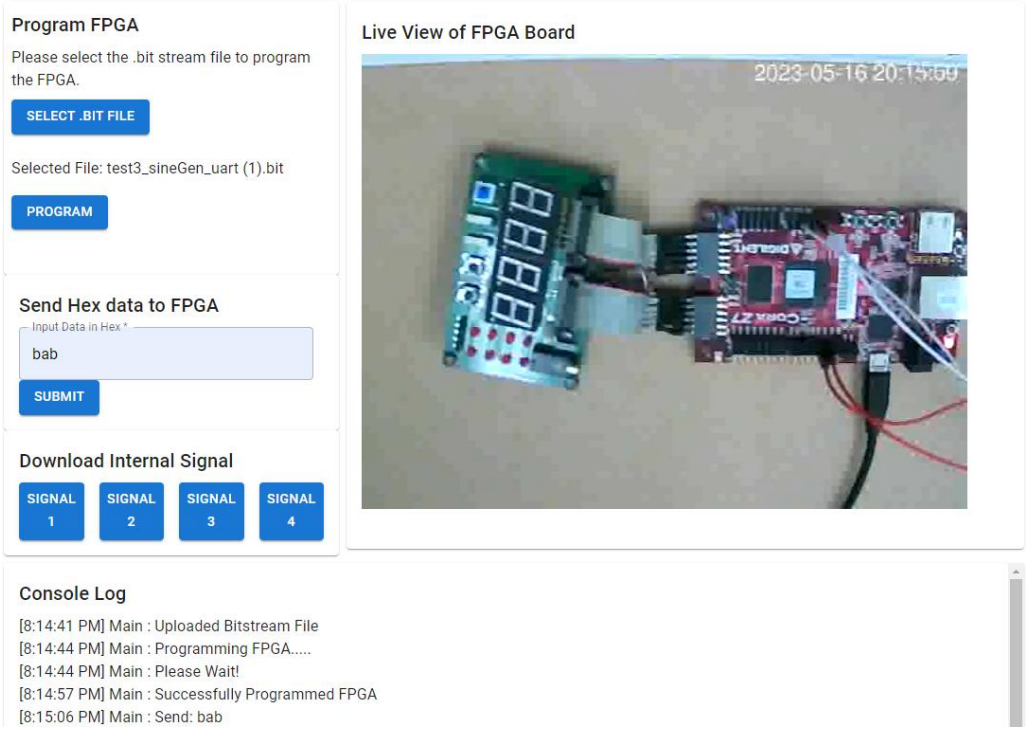


Figure 6. The RTL design schematic of *Advanced_Remote_Lab* can capture up to four different internal signals and be integrated with user-designed modules. The module facilitates both data transmission and reception with the laboratory server



(a)



(b)

Figure 7. The RTL design was (a) incorporated into the *Advanced_Remote_Lab* to capture the internal signal of a sinusoidal waveform generator, and (b) validated using the proposed advanced FPGA laboratory

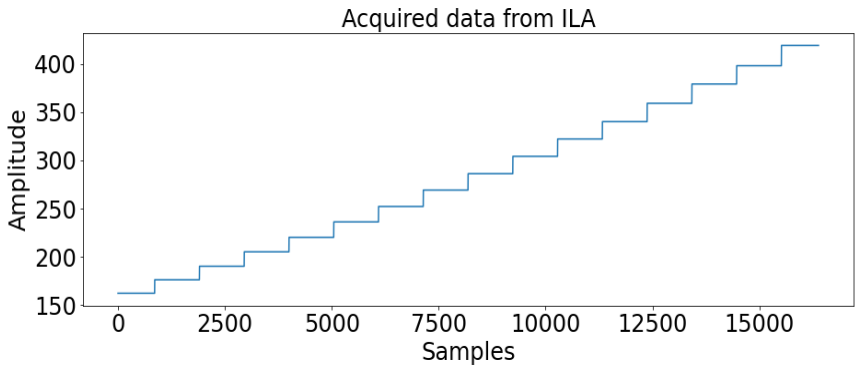


Figure 8. The samples were correctly acquired from the ILA using *Advanced_Remote_Lab* module when the represented signal is a sine wave. The samples can be compressed significantly because of repetitions in the sample values

3.2. Proposed data compression technique for remote FPGA laboratory

The data compression technique required for the proposed remote lab should use few parameters and operations, thereby achieving low hardware resource utilization when deployed in hardware-embedded devices (in our case, FPGAs). As mentioned in the introduction, several lossless compression techniques can be implemented in FPGA devices, but run-length encoding (RLE) is the most suitable for our remote laboratory because of its efficiency when processing input data with long sequences of repeated symbols. Additionally, RLE meets the requirements for utilizing low hardware resources. Therefore, in this study, we propose an adaptive RLE data compression technique suitable for acquiring a large number of signal samples within a specific time frame or buffer size, thereby maximizing the data throughput.

Algorithm 1 is a decision-making algorithm that determines whether the input data stream must be compressed using the adaptive run-length encoding data compression algorithm based on a threshold value. The function takes two parameters: the input data stream *D* and threshold *th*. It initializes the variable *count* to 0, *isEncode* to 0, and *prev* to null. The variable *isEncode* indicates the detection of consecutive runs of the input signal occurring for a defined threshold value. If *isEncode* is one, the data are encoded with the RLE before being sent to the laboratory server. Otherwise, the data are stored directly in a buffer and sent to the laboratory server. The algorithm then loops over the first *th* element of the input data stream and checks whether the current element is the same as the previous element. If it is, the algorithm increments the count variable; otherwise, it sets *count* to 1. Finally, the algorithm checks whether *count* is greater than or equal to the threshold *th* and sets *isEncode* to 1 if it is, and 0 otherwise. This function returns the value of *isEncode*.

Algorithm 1. Decision maker for determining whether the input data stream needs to be encoded using the adaptive RLE data compression

```
Require: input data stream D, threshold th
  1:  function ENCODEDECISION(D, th)
  2:           count ← 0
  3:           isEncode ← 0
  4:           prev ← null
  5:           for i ← 0 to (length of D)-1 do
  6:              s ← D[i]
  7:              if s = prev then
  8:                 count ← count + 1
  9:              else
 10:                 count ← 1
 11:              end if
 12:              if count ≥ th then
 13:                 isEncode ← 1
 14:                 return isEncode
 15:              end if
 16:              prev ← s
 17:           end for
 18:           isEncode ← 0
 19:           return isEncode
 20:  end function
```

After determining whether the input data stream should be encoded using RLE, Algorithm 2 applies adaptive RLE data compression to compress input data stream *D* based on the decision made by Algorithm 1. The function uses two parameters: the input data stream *D* and decision variable *isEncode*. It initializes the compressed data stream *C* to an empty set, the *count* variable to 0, and the *prev* variable to the first element of input data stream *D*. If the decision variable *isEncode* is 0, it appends a header to the compressed data stream, indicating that the data stream does not need to be encoded using RLE, and then appends the entire input data stream *D* to the compressed data stream *C*. If the decision variable *isEncode* is 1, it loops over each element of the input data stream *D*. For each element, the algorithm checks whether the current element is the same as the previous element. If it is, the algorithm increments the *count* variable; otherwise, the algorithm appends the previous element and its count to the compressed data stream *C* and sets *count* to 0. The algorithm returns data stream *C*.

The performance of the proposed compression method is evaluated based on its compression ratio (CR) and throughput. CR is a measure of the reduction in the size of the data achieved by a compression algorithm. It is calculated by dividing the size in bytes S_{uc} of the uncompressed data by the size in bytes S_c of the compressed data as (1):

$$CR = \frac{S_{uc}}{S_c} \quad (1)$$

A higher CR indicates a better compression algorithm. The throughput is the maximum number $Data_{\max}$ of bytes transmitted in a given duration, where

$$Data_{\max} = CR \times S_{\text{uc}} \quad (2)$$

Since $CR \leq 1$, the maximum number $Data_{\max}$ of bytes is at most S_{uc}. The CR and throughput are appropriate performance measures.

The adaptive RLE scheme ensures that the data are only compressed when it is beneficial to do so, thereby reducing the overhead associated with unnecessary compression and decompression of data. Simultaneously, the hardware implementation of the proposed algorithm was embedded in the *Signal_Capturing_x* block of the *Advanced_Remote_Lab module*. Additionally, the decompression process that occurs at the server computer involves checking the header of the data to determine whether the data are compressed using the RLE.

Algorithm 2. Adaptive RLE for advance remote lab module

```
Require: input data stream D, decision variable isEncode
  1:  function ADAPTIVERLE(D, isEncode)
  2:           C ← ∅
  3:           count ← 0
  4:           prev ← D[0]
  5:           if isEncode = 0 then
  6:              append isEncode to C                                  ▷ add header
  7:              append D to C      ▷ append the data stream to the compressed data stream
  8:           else
  9:              append isEncode to C                                  ▷ add header
 10:              for i ← 0 to (length of D)-1 do
 11:                 s ← D[i]
 12:                 if s = prev then
 13:                    count ← count + 1
 14:                 else
 15:                    append prev to C
 16:                    append count to C
 17:                    count ← 1
 18:                 end if
 19:                 prev ← s
 20:              end for
 21:              append prev to C
 22:              append count to C
 23:           end if
 24:           return C
 25:  end function
```

4. RESULT AND DISCUSSION

The proposed approach is evaluated in three aspects: i) Flexibility of designing an embedded system remotely using the proposed platform; ii) Throughput of the proposed adaptive RLE algorithm; and iii) Utilization of hardware resources of the proposed modules.

4.1. Flexibility of the proposed remote lab

Table 1 presents a comparative evaluation of the existing features in currently operated remote FPGA labs. The table presents a comparison of the features of the nine remote laboratory solutions with those of the solutions presented in this study. The desired features include: i) remote FPGA configuration, ii) real-time interaction, iii) ability to capture internal signals from the FPGA during the experiment, iv) timing diagram display, v) local view, vi) fully integrated web-based application, and vii) ability to freely program the FPGA with the users' own design instead of using the default experiment provided by the lab host. The more features a remote lab has, the better the user experience.

The results show that several related studies have proposed similar remote FPGA labs that utilize web-based interfaces to interact remotely with the FPGA. However, some of these proposed laboratories have drawbacks such as limited functionality, lack of flexibility, and dependency on the laboratory host's designed experiments. The proposed remote FPGA lab overcomes these limitations by providing users with the ability to freely program an FPGA board remotely using a standard web browser, making it easily accessible to a wide range of users, and offering various features to their design.

Additionally, the primary focus of most developers of remote FPGA lab platforms is to provide users with access to the FPGA board through a laboratory infrastructure. A less important feature for

developers is providing users with debugging capabilities to troubleshoot their programs. The existing methods for debugging a program capture the internal signals from the FPGA board and timing diagram display during experiments. The existing remote FPGA that provides these features does not allow users to perform the experiment in a single web-based application; hence, additional software must run in parallel during the experimentation process. Running additional software in parallel may introduce complexity for users on a remote lab platform, potentially causing difficulties in fully accessing the lab and leading to increased problems. Furthermore, the acquired data throughput was lower than that of the proposed remote FPGA lab because the proposed remote FPGA lab performs adaptive data compression before sending the data to the user. Our proposed FPGA remote lab fully integrates the entire testing process of the users' design into a web-based application, ensuring a seamless experience.

Table 1. The proposed remote FPGA laboratory supports more features than the existing remote FPGA laboratories support (✓=supports, ×=does not support)

Reference	Desired features						
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Winzker and Schwandt [16]	✓	✓	×	×	×	✓	✓
Valencia *et al.* [11]	×	✓	×	×	✓	×	✓
Monzo *et al.* [20]	×	✓	×	×	✓	×	×
Blochwitz *et al.* [21]	×	✓	×	×	✓	×	×
Oballe-Peinado *et al.* [22]	×	✓	×	×	✓	✓	×
Mohsen *et al.* [17]	✓	✓	✓	✓	✓	×	×
Melosik *et al.* [12]	✓	×	×	✓	✓	×	✓
Magyari and Chen [23]	×	✓	×	×	×	✓	✓
Touhafi *et al.* [18]	✓	×	✓	✓	×	✓	✓
Proposed Remote FPGA Lab	✓	✓	✓	×	✓	✓	✓

According to Table 1, a feature was missing from the proposed laboratory. Feature 4 is related to the timing diagram. The proposed laboratory did not provide a timing-diagram display in the web-based interface. However, user can still plot the timing diagram by downloading the data acquired from the FPGA and using an external tool to plot the timing diagram. Including a timing diagram in the web-based interface is a future research direction.

4.2. Data throughput

The throughput of the internal signals sent during the experiment was evaluated in two steps. First, we measured the average data size of the signal. Second, we measured the average compressed size achieved by utilizing the existing and proposed data compression techniques. Three types of benchmark signals with distinct characteristics were used for the evaluation. Signal 1 represents the data acquired from a micro-electrical-mechanical system (MEMS) microphone sensor with an input pulse density modulation (PDM) clock of 3.072 MHz and is deserialized to 16 bits. Signal 2 is 16-bit counter data generated at every clock cycle. Signal 3 is a sinusoidal waveform that requires 1,048 clock cycles to produce one sample. The timing diagram of these signals is shown in Figure 9. The three benchmark signals represent the typical signals found in the embedded system design and form a fair test dataset.

Table 2 compares the compression performance when different compression algorithms were applied to the test signals. Although the proposed method may occasionally result in larger signal sizes compared with existing methods, it achieves an average reduction of approximately 65.63% compared with the total average signal size, with a calculated CR of 2.90. The proposed adaptive RLE is significantly better than the next best compression method, Huffman, which has an average compressed size of 14,491 bytes equivalent to a 51.09% reduction from the total average signal size. Therefore, the proposed method is effective in compressing the data while the user wishes to acquire the internal signal during the experiment. Furthermore, the proposed method is particularly effective when the signal contains duplicate data for a certain period, as shown by the results obtained from signals 1 and 3.

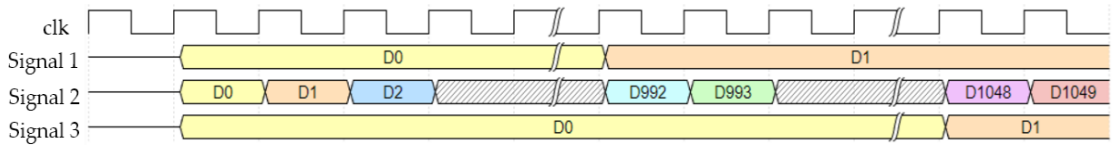


Figure 9. Timing diagrams of the three benchmark signals were acquired from the FPGA

Table 2. On average, the proposed adaptive RLE can compress more data than the existing methods can

Type of signal	Signal size (bytes)	Compressed size (bytes)			
		RLE	Huffman	LZW	Proposed adaptive RLE
Signal 1	32,768	504	8,229	2,872	505
Signal 2	32,768	65,536	27,109	42,668	32,769
Signal 3	32,768	512	8,225	2,878	513
Average size	32,768	22,184	14,491	16,139	11,262

4.3. Hardware utilization

In this section, we evaluate the hardware resource utilization of the proposed advanced remote lab module. The FPGA was implemented on a Zynq 7000 FPGA fabric using the Vivado version 2018.3. For resource utilization analysis, the designs were coded in VHDL and synthesized using the Xilinx Synthesis Tool, which is included in the Vivado version 2018.3. First, we compare the utilization of the proposed compression method with that of the existing method. Second, we determined the total resources required for the operation of the advanced remote lab module. The evaluation provides insight into the effectiveness of the proposed compression method, which aims to use the least amount of hardware resources on the experimental board while preserving more resources for students to perform their own design.

For comparison, we evaluated the resource utilization of existing compression methods, namely, entropy-based, dictionary-based, and deflate-based compression methods, which have been previously proposed and implemented on FPGA devices. The comparison presented in Table 3 shows that the proposed algorithm does not require any block random access memory (BRAM) because the implementation of RLE discussed in section 4 only uses look-up tables (LUTs) and flip-flops (FFs) when implemented in an FPGA. Additionally, the proposed method has a lower count of LUTs, FFs, and Slices compared with existing compression methods that have been implemented on an FPGA. Therefore, the proposed adaptive RLE is the most suitable data compression method for implementation in the proposed *Advanced_Remote_Lab* module and offers a high CR in certain cases.

Table 3. The proposed method stands out for its low hardware utilization

Method	Resource Utilization				Device
	BRAM 36k	LUT	FF	Slice	
Entropy-based [25]	-	624	-	-	Zynq 7035
Entropy-based [26]	62	-	-	1,836	Zynq 7020
Dictionary-based [27]	8	-	-	3,216	FPGA XC4VLX15-10
Dictionary-based [28]	3	1,323	1,379	-	Altera Stratix IV
Deflate-based [32]	131	69,114	49,779	-	Xilinx XCVU3P-FFVC1517
Deflate-based [33]	2	7,965	2,342	-	Altera DE2 board
Proposed algorithm	0	95	164	47	Zynq 7000

The *Advanced_Remote_Lab* module has been implemented on various FPGA devices, such as Cora-Z7, PYNQ-Z1, Basys3, and Zybo Z7-10. Figure 10 illustrates resource utilization, which varies across different devices. Despite these minor variations, the design process of the student model is unaffected. As a result, the proposed module is well-suited for implementation on different types of FPGA devices.

Table 4 lists the number of BRAMs required to acquire signals of different lengths. Based on this table, the number of BRAMs utilized in the FIFO generator varies depending on the length of the signal that the user wishes to acquire. Hence, users must be cautious in selecting the signal length they want to acquire because of the limitations of the BRAM unit in the targeted FPGA board. For instance, capturing a signal with a length of 131,072 samples requires 128 BRAMs of 36K. However, the Cora-Z7 board has only 50 available 36K BRAM. Therefore, users are advised to choose the appropriate signal length they want to acquire in their design to match the available BRAM of the targeted FPGA device.

Using the proposed adaptive RLE method to compress the signal before storing it in BRAM, more data can be acquired while utilizing a small amount of BRAM. As shown in Figure 8, without the use of adaptive RLE, 16 BRAMs were used to capture 16,384 samples. By contrast, utilizing the adaptive RLE allows the same usage of BRAMs to capture a maximum data sample ($Data_{\max}$) of up to 8,388,608 samples, as depicted in Figure 11 (with only 1,000,000 samples plotted for simplicity). Given the large amount of data acquired, the process of debugging and analysis becomes more convenient. For instance, the data illustrated in Figure 8 alone does not allow the user to determine whether it represents a sinusoidal waveform. However, Figure 11 demonstrates that when using the proposed compression method, the

acquired data clearly exhibit a sinusoidal waveform, verifying the correctness of the user's design. Moreover, if the user wishes to capture only 16,384 samples, the utilized BRAM will cost only 0.5 of the 36K BRAM or 1 of the 18K BRAM, which demonstrates the efficiency and potential resource utilization savings of the proposed method.

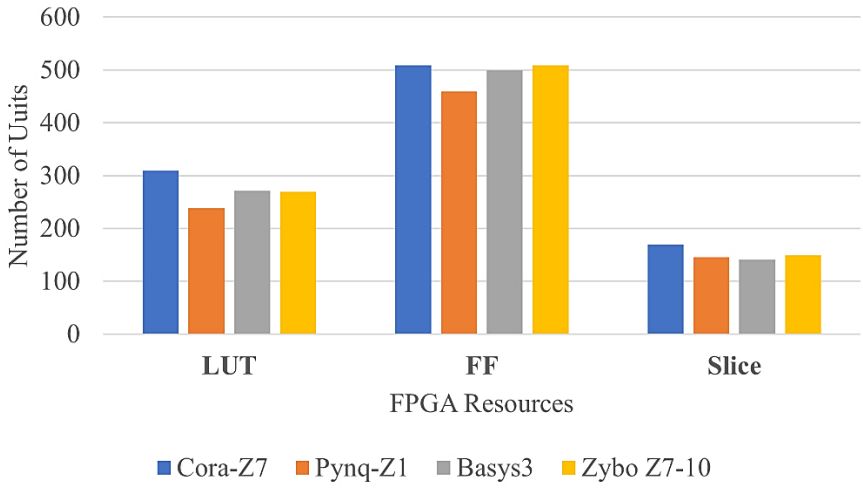


Figure 10. The total hardware resource utilization of *Advanced_Remote_Lab* varies across FPGA devices

Table 4. The number of available BRAMs in an FPGA board constrains the length of signal that users can acquire

Signal length (sample)	36k BRAM usage	Signal length (sample)	36k BRAM usage
16	0.5	2048	2
32	0.5	4096	4
64	0.5	8192	8
128	0.5	16384	16
256	0.5	32769	32
512	1	65536	64
1024	1	131072	128

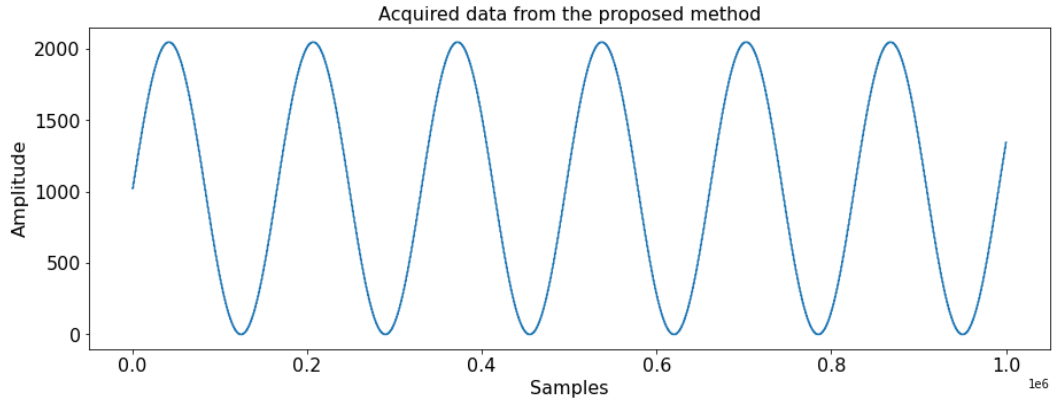


Figure 11. The proposed adaptive RLE is utilized with 16 BRAMs to compress the internal signal. The plot shows increased sample acquisition and representation of multiple cycles of a sine wave

5. CONCLUSION

In this study, we designed and evaluated a remote FPGA laboratory that aims to enhance the learning experience of students in the field of electronic engineering, particularly in FPGA design at RTL. The proposed remote laboratory offers flexibility to students by providing them with a fully integrated web-based application to conduct remote experiments. This platform enables students to program an FPGA board

from any location and offers a range of options for testing their design. To facilitate these features, we implemented two modules, *Basic_Remote_Lab* and *Advanced_Remote_Lab*, that students can choose and incorporate into their designs. One of the novel options is the ability to acquire high-throughput data from an FPGA board during experiments. This novel option has been proven to be an effective approach for conducting experiments on embedded systems. To optimize the data transmission, we incorporated a hardware-based adaptive data compression technique into the proposed module to compress the acquired data before sending them to the server. Furthermore, the total utilization of hardware resources remains below 5% of the targeted FPGA board, demonstrating that the use of additional lab modules does not significantly impact student designs.

There are several directions for extending the scope of this study. The primary direction is to implement the remaining feature, which enables users to access the graphical timing diagrams of the acquired signal. This feature is essential because it enhances the usability of our remote FPGA lab and makes it easier for users to comprehend and analyze the graphical timing diagrams of the acquired signal, thereby improving their overall experience and productivity. Other potential future research directions include enabling each user to program multiple FPGA boards. This capability is particularly valuable for experiments that involve device-to-device communication and parallel processing. Furthermore, we are seeking an architecture design for a remote FPGA lab that enables flexibility in moving experimental instances. This architecture is particularly advantageous for experiments involving signal acquisition from the sensor devices. This design should allow the seamless relocation of the experimental setup, allow users to gather data from different sensor locations, and facilitate experiments that require dynamic sensor placement. Future research directions will extend the current remote laboratory to include more use cases.

ACKNOWLEDGEMENTS

This work was supported by the Petch Pra Jom Klao Masters Degree Scholarship from King Mongkut's University of Technology Thonburi. This work was supported in part by the Research Strengthening Project of the Faculty of Engineering, King Mongkut's University of Technology Thonburi.

REFERENCES

[1] A. Shoufan, "Active distance learning of embedded systems," *IEEE Access*, vol. 9, pp. 41104–41122, 2021, doi: 10.1109/ACCESS.2021.3065248.

[2] B. C. Choi, S. H. Lee, J. C. Na, and J. H. Lee, "Secure firmware validation and update for consumer devices in home networking," *IEEE Transactions on Consumer Electronics*, vol. 62, no. 1, pp. 39–44, Feb. 2016, doi: 10.1109/TCE.2016.7448561.

[3] R. Tong, Q. Jiang, Z. Zou, T. Hu, and T. Li, "Embedded system vehicle based on multi-sensor fusion," *IEEE Access*, vol. 11, pp. 50334–50349, 2023, doi: 10.1109/ACCESS.2023.3277547.

[4] N. Arandia, J. I. Garate, and J. Mabe, "Embedded sensor systems in medical devices: requisites and challenges ahead," *Sensors*, vol. 22, no. 24, Dec. 2022, doi: 10.3390/s22249917.

[5] M. Mahbub, M. M. Hossain, and M. S. A. Gazi, "IoT-cognizant cloud-assisted energy efficient embedded system for indoor intelligent lighting, air quality monitoring, and ventilation," *Internet of Things*, vol. 11, Sep. 2020, doi: 10.1016/j.iot.2020.100266.

[6] A. Kumar, S. Fernando, and R. C. Panicker, "Project-based learning in embedded systems education using an FPGA platform," *IEEE Transactions on Education*, vol. 56, no. 4, pp. 407–415, Nov. 2013, doi: 10.1109/TE.2013.2246568.

[7] S. Pasricha, "Embedded systems education: experiences with application-driven pedagogy," *IEEE Embedded Systems Letters*, vol. 14, no. 4, pp. 167–170, Dec. 2022, doi: 10.1109/LES.2022.3175686.

[8] M. Ibro and G. Marinova, "Literature review on FPGA-based e-learning: power consumption design methodologies perspective," in *2022 29th International Conference on Systems, Signals and Image Processing (IWSSIP)*, Jun. 2022, pp. 1–4, doi: 10.1109/IWSSIP55020.2022.9854449.

[9] P. Minev, V. Kukenska, I. Varbov, and M. Dinev, "Systems for remote access to FPGA development boards: review," in *2022 XXXI International Scientific Conference Electronics (ET)*, Sep. 2022, pp. 1–6, doi: 10.1109/ET55967.2022.9920279.

[10] 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. 1–13, Oct. 2012, doi: 10.1145/2362374.2362382.

[11] F. Valencia de Almeida *et al.*, "Teaching digital electronics during the COVID-19 pandemic via a remote lab," *Sensors*, vol. 22, no. 18, Sep. 2022, doi: 10.3390/s22186944.

[12] M. Melosik Michałand Naumowicz, M. Kropidłowski, and W. Marszalek, "Remote prototyping of FPGA-based devices in the IoT concept during the COVID-19 pandemic," *Electronics*, vol. 11, no. 9, May 2022, doi: 10.3390/electronics11091497.

[13] A. Bekasiewicz, B. Pankiewicz, M. Wojcikowski, M. Klosowski, and S. Koziel, "Application of open-hardware-based solutions for rapid transition from stationary to the remote teaching model during pandemic," *IEEE Transactions on Education*, vol. 64, no. 3, pp. 299–307, Aug. 2021, doi: 10.1109/TE.2020.3043479.

[14] V. M. Aitor, J. Garcia-Zubia, I. Angulo, and L. Rodriguez-Gil, "Toward widespread remote laboratories: evaluating the effectiveness of a replication-based architecture for real-world multiinstitutional usage," *IEEE Access*, vol. 10, pp. 86298–86317, 2022, doi: 10.1109/ACCESS.2022.3198961.

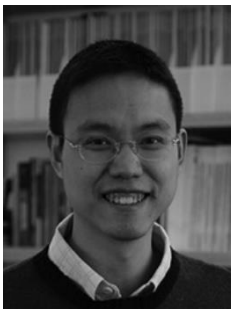
[15] M. L. Crespo *et al.*, "Remote laboratory for E-learning of systems on chip and their applications to nuclear and scientific instrumentation," *Electronics*, vol. 10, no. 18, p. 2191, Sep. 2021, doi: 10.3390/electronics10182191.

[16] M. Winzker and A. Schwandt, "Open education teaching unit for low-power design and FPGA image processing," *2019 IEEE Frontiers in Education Conference (FIE)*, Covington, KY, USA, 2019, pp. 1-9, doi: 10.1109/FIE43999.2019.9028694.

[17] A. E. R. Mohsen, M. Youssef Gadalrab, Z. E. Mahmoud, G. Alshaer, M. Asy, and H. Mostafa, "Remote FPGA Lab for ZYNQ and Virtex-7 Kits," in *Midwest Symposium on Circuits and Systems*, Aug. 2019, pp. 185–188, doi: 10.1109/MWSCAS.2019.8885064.
[18] A. Touhafi, A. Braeken, A. Tahiri, and M. Zbakh, "CoderLabs: a cloud based platform for real time online labs with user collaboration," in *Proceedings Of 2016 Application of Open-hardware-based Solutions for Rapid Transition from Stationary to The Remote Teaching Model During Pandemic International Conference on Cloud Computing Technologies and Applications*, May 2017, pp. 317–324, doi: 10.1109/CloudTech.2016.7847716.
[19] K. P. Ayodele, I. A. Inyang, and L. O. Kehinde, "An iLab for teaching advanced logic concepts with hardware descriptive languages," *IEEE Transactions on Education*, vol. 58, no. 4, pp. 262–268, Nov. 2015, doi: 10.1109/TE.2015.2395996.
[20] C. Monzo, G. Cobo, J. A. Morán, E. Santamaría, and D. García-Solórzano, "Remote laboratory for online engineering education: the RLAB-UOC-FPGA case study," *Electronics*, vol. 10, no. 9, May 2021, doi: 10.3390/electronics10091072.
[21] C. Blochwitz, P. Grothe, S. Dreier, W. Aljnabi, R. Buchty, and M. Berekovic, "RemEduLa - remote education laboratory for FPGA design technology," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2022, pp. 1773–1777, doi: 10.1109/ISCAS48785.2022.9937666.
[22] O. Oballe-Peinado, J. Castellanos-Ramos, J. A. Sanchez-Durqan, R. Navas-Gonzalez, A. Daza-Marquez, and J. A. Botin-Cordoba, "FPGA-based remote laboratory for digital electronics," in *2020 XIV Technologies Applied to Electronics Teaching Conference (TAEE)*, Jul. 2020, pp. 1–5, doi: 10.1109/TAEE46915.2020.9163676.
[23] A. Magyari and Y. Chen, "FPGA remote laboratory using IoT approaches," *Electronics*, vol. 10, no. 18, Sep. 2021, doi: 10.3390/electronics10182229.
[24] Xilinx, "Integrated logic analyzer v6.2 – LogiCORE IP product guide," *Vivado Design Suite*, *Xilinx*, 2016. Accessed: Dec. 25, 2023. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg172-ila
[25] Y. Shan, X. Chen, C. Qiu, and Y. Zhang, "Implementation of fast Huffman encoding based on FPGA," *Journal of Physics: Conference Series*, vol. 2189, no. 1, Feb. 2022, doi: 10.1088/1742-6596/2189/1/012021.
[26] J. Matai, J.-Y. Kim, and R. Kastner, "Energy efficient canonical Huffman encoding," in *2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors*, Jun. 2014, pp. 202–209, doi: 10.1109/ASAP.2014.6868663.
[27] W. Cui, "New LZW data compression algorithm and its FPGA implementation," *PCS 2007 - 26th Picture Coding Symposium*. Lisbon, Portugal, pp. 1145–1148, 2007.
[28] B. Sukhwani, B. Abali, B. Brezzo, and S. Asaad, "High-throughput, lossless data compression on FPGAs," in *2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*, May 2011, pp. 113–116, doi: 10.1109/FCCM.2011.56.
[29] W. Liu, F. Mei, C. Wang, M. O'Neill, and E. E. Swartzlander, "Data compression device based on modified LZ4 algorithm," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 1, pp. 110–117, Feb. 2018, doi: 10.1109/TCE.2018.2810480.
[30] J. L. Nunez and S. Jones, "Gbit/s lossless data compression hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 3, pp. 499–510, Jun. 2003, doi: 10.1109/TVLSI.2003.812288.
[31] S. Naqvi, R. Naqvi, R. A. Riaz, and F. Siddiqui, "Optimized RTL design and implementation of LZW algorithm for high bandwidth applications," *Przeglad Elektrotechniczny*, vol. 87, no. 4, 2011.
[32] M. Ledwon, B. F. Cockburn, and J. Han, "High-throughput FPGA-based hardware accelerators for deflate compression and decompression using high-level synthesis," *IEEE Access*, vol. 8, pp. 62207–62217, 2020, doi: 10.1109/ACCESS.2020.2984191.
[33] S. Rigler, W. Bishop, and A. Kennings, "FPGA-based lossless data compression using Huffman and LZ77 algorithms," in *2007 Canadian Conference on Electrical and Computer Engineering*, 2007, pp. 1235–1238, doi: 10.1109/CCECE.2007.315.
[34] R. De Jesus Navas-Gonzalez, O. Oballe-Peinado, J. Castellanos-Ramos, D. Rosas-Cervantes, and J. A. Sanchez-Duran, "Digital electronics practice projects for an FPGA-based remote laboratory," in *2022 Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica (XV Technologies Applied to Electronics Teaching Conference)*, Jun. 2022, pp. 1–6, doi: 10.1109/TAEE54169.2022.9840627.
[35] A. Villar-Martinez, L. Rodriguez-Gil, I. Angulo, P. Orduna, J. Garcia-Zubia, and D. Lopez-De-Ipina, "Improving the scalability and replicability of embedded systems remote laboratories through a cost-effective architecture," *IEEE Access*, vol. 7, pp. 164164–164185, 2019, doi: 10.1109/ACCESS.2019.2952321.
[36] C. Salzmann, S. Govaerts, W. Halimi, and D. Gillet, "The smart device specification for remote labs," in *Proceedings of 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, Feb. 2015, pp. 199–208, doi: 10.1109/REV.2015.7087292.
[37] F. Schauer, M. Krbecek, P. Beno, M. Gerza, L. Palka, and P. Spilakova, "REMLABNET - open remote laboratory management system for e-experiments," in *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, Feb. 2014, pp. 268–273, doi: 10.1109/REV.2014.6784273.
[38] Xilinx, "FIFO generator v13.1 – LogiCORE IP product guide," *Vivado Design Suite*, *Xilinx*, 2017. Accessed: Dec. 25, 2023. [Online]. Available: https://docs.xilinx.com/v/u/13.1-English/pg057-fifo-generator

BIOGRAPHIES OF AUTHORS

Rithea Sum received a B.Eng. degree in electrical communication and electronics engineering from King Mongkut's University of Technology Thonburi (KMUTT), Thailand (2021). He is currently pursuing a master's degree in electrical and information engineering at the Department of Electronic and Telecommunication Engineering, KMUTT, Thailand. His main research interests include digital signal processing, internet of things and their applications, and digital systems design and implementation. He can be contacted at email: rithea.s@mail.kmutt.ac.th.



Watcharapan Suwansantisuk received B.S. degrees in electrical and computer engineering and in computer science from Carnegie Mellon University, Pennsylvania, in 2002, and M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology in 2004 and 2012, respectively. He is currently an assistant professor at King Mongkut's University of Technology Thonburi (KMUTT), Thailand. Before joining KMUTT, he spent summers at the University of Bologna, Italy, as a visiting research scholar, and at the Alcatel-Lucent Bells Laboratory, NJ, as a research intern. His main research interests include wireless communications, synchronization, and statistical signal processing. He serves on the technical program committees for various international conferences and served as the symposium Co-Chair for the IEEE Global Communications Conference in 2015. He received the Leonard G. Abraham Prize in the field of communications systems from the IEEE Communications Society in 2011, jointly with Prof. M. Chiani and Prof. M. Win, and the Best Paper Award from the IEEE RIVF International Conference on Computing and Communication Technologies in 2016, jointly with N. Chedoloh. He can be contacted at email: watcharapan.suw@kmutt.ac.th.

Pinit Kumhom received the B.Eng. degree in electrical engineering from the King Mongkut's Institute of Technology Thonburi, Thailand, in 1988, and the Ph.D. degree in electrical and computer engineering from Drexel University, Pennsylvania, in 2000. He is currently an assistant professor with the King Mongkut's University of Technology Thonburi. His research interests include the internet of things and their applications, digital system design and implementation, and signal and image processing. He can be contacted at email: pinit.kumhom@mail.kmutt.ac.th.