RHLab: Towards Implementing a Partial Reconfigurable SDR Remote Lab

Zhiyun Zhang[1], Marcos Inoñan[1], Pablo Orduña[2], and Rania Hussein[1]

[1] University of Washington, Seattle WA 98195, USA,
{zzyzzy42, minonan, rhussein}@uw.edu,
[2] LabsLand, San Francisco, CA 94114, USA
pablo@labsland.com

Abstract. Software-Defined Radio (SDR) remote labs permit students to experiment with real wireless communication, designing Radio Frequency (RF) systems systems with minimal code adjustments. This feature allows them to create RF prototypes remotely in a fast way allowing them to complement their theory of communication classes. While SDR hardware suffices for most basic applications, some demand extensive Signal Processing stages that surpass the capabilities of standard SDR equipment. SDR devices are controlled by reprogrammable digital logic devices like FPGA which have some limitations in terms of capabilities/price factor. For this case Partial Reconfiguration (PR) emerges as a solution, leveraging to use the resources of these devices more efficiently. In the conventional approach, modifying FPGA designs required users to undertake the laborious process of resynthesizing, implementing, and programming the entire FPGA. Consequently, this procedure is time-consuming and impedes users' progress. However, with partial reconfiguration, users only need to resynthesize and program the specific portions or slices of the FPGA that necessitate modification. However, it necessitates a specialized understanding of FPGA design, involving the creation of modifiable regions. This paper takes initial strides towards establishing a remote laboratory for students to explore wireless communication concepts, harnessing PR for SDR devices.

Keywords: Software-Defined Radio, Remote Laboratory, Partial Reconfiguration, Embedded Systems

1 Introduction

In traditional Science, Technology, Engineering, and Mathematics (STEM) courses, students typically engage in hands-on laboratory experiments using physical lab kits provided by the instructional team in physical laboratories [1]. However, this conventional approach has several drawbacks. First, it is not inclusive for students with disabilities, as it requires them to physically attend school, which may pose risks to their physical and mental well-being [2]. Secondly, this method can lead to equipment damage, such as short circuits or the dropping of devices [3]. Lastly, it places a greater financial burden on educational institutions, as they

need to purchase and maintain a large number of lab kits [4]. Unfortunately, these lab kits are often underutilized, as not all students use the devices simultaneously, which fails to capitalize on the potential for greater flexibility.

These disadvantages were largely addressed through the advent of remote laboratories [5, 6]. Following the emergence of the internet, numerous educational institutions have shown an interest in creating laboratories that grant students access to physical devices remotely [7]. Despite initial skepticism, remote laboratories have demonstrated their effectiveness. Numerous engineering educators and students have reaped the benefits of these remote facilities [8, 9]. In a study conducted during a college-level FPGA course at the University of Washington, it was found that students exhibited improved analytical skills and achieved higher overall scores when utilizing a remote hardware laboratory, as compared to a hands-on laboratory. The flexible access to remote laboratories also facilitated a greater number of students in completing their work [10].

Software-Defined Radios, commonly known as SDRs, are radio systems that can be configured and controlled via software programming [11,12]. In contrast to traditional hardware-based radios, which are constrained by fixed internal components determining specifications like frequency range, bandwidth, and sampling rate, SDR devices provide substantial advantages in terms of flexibility, reconfigurability, and cost-effectiveness. They can be readily updated by reprogramming their software, and by replacing many hardware components with software-driven solutions, SDR devices often exhibit improved power efficiency and affordability [13, 14]. Over the past two decades, we have observed remarkable progress in SDR devices. They have grown more robust and capable while simultaneously reducing in size and cost [15].

As a result, a growing number of educational institutions have incorporated telecommunication courses into their curriculum, aiming to provide students with expertise in areas such as radio frequency (RF), wireless communication, and software-defined radio (SDR) [16]. Additionally, many of these courses mandate a laboratory section to give students a full learning experience. Addressing the limitations associated with conventional in-person labs, the Remote Hub Lab (RHL/RHLab)[3] group [17] and LabsLand[4] have developed the RHL-RELIA project [18]. This project, based on the MELODY model [19], seeks to establish a remote laboratory facilitating student access to popular SDR devices, enabling convenient and flexible usage from any location and at any time [20–22].

Building upon the achievements of the existing remote laboratory, where users were empowered to program ADALM-PLUTO (PlutoSDR) through a web interface, we are advancing the platform by introducing additional features. Specialized courses are being developed to harness the capabilities of this remote lab, including the creation of practical applications such as a radar remote laboratory. Moreover, we are in the final stages of integrating another cost-effective SDR device, the Red Pitaya, into the project. This paper covers our effort to incorporate partial reconfiguration into the existing RHL-RELIA system. This

[3] https://rhlab.ece.uw.edu
[4] https://labsland.com

step is aimed at optimizing the use of these more economical FPGAs and enabling students to explore the potentials of partial reconfiguration [23].

The paper is structured into the following sections: Section 2 provides an overview of partial reconfiguration using Zynq SoCs and explores its feasibility within the SDR domain. Section 3 summarizes our approach to implementing PR within an established hardware platform. Section 4 showcases our current progress in this endeavor. The final section, Section 5, offers a conclusion of the paper and outlines the forthcoming steps.

2 Background

Partial reconfiguration (PR) refers to the process of updating specific portions or slices of the FPGA while leaving other sections unchanged[5]. This allows for flexibility and adaptability in SDR programs, enabling improved performance and functionality [24–26].

The majority of inexpensive SDRs are equipped with Zynq 7000 System-on-Chips (SoCs), which belong to the Xilinx family of SoCs comprising a Processing System (PS) and a Programmable Logic (PL). The PL component is built around an FPGA, offering high-performance real-time computing capabilities. However, SDRs in the price range of $200 to $400 commonly utilize Zynq Z-7010 or Z-7020 SoCs, both of which are cost-optimized members within the Zynq-7000 SoC family. They come with a limited number of logic cells and Look-Up Tables (LUTs). For instance, the Z-7010 SoC features 28,000 logic cells and 17,600 LUTs, which are only about 1/10 of the logic cells and 1/10 of the LUTs found in the mid-range Z-7035, which boasts 275,000 logic cells and 171,900 LUTs[6].

Inexpensive SDRs often deplete the FPGAs rapidly if not used efficiently, given that most signal processing applications are resource-intensive and consume a significant number of processing logic cells and LUTs. Partial reconfiguration provides a solution to this issue by allowing specific sections of an FPGA to be updated and modified, adapting to the current operational mode as needed[7]. The effectiveness of partial reconfiguration has been demonstrated in the Software-Defined Radio domain. A study using the USRP E310 SDR revealed that performing partial reconfiguration is over four times faster than a full configuration of the entire FPGA. In their signal processing design, partial reconfiguration takes 33 ms, while a full reconfiguration lasts 143 ms [27].

Dynamic Partial Reconfiguration (DPR) stands out as a specialized form of Partial Reconfiguration (PR) that empowers engineers to modify specific portions of an FPGA while it is actively processing data [28–30]. In a study conducted by a research team at Cairo University, an assessment of DPR in wireless communication technologies like WiFi, Bluetooth, and LTE revealed that DPR

[5] https://www.xilinx.com/products/design-tools/partial-reconfiguration.html
[6] https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html#productTable
[7] https://www.xilinx.com/products/design-tools/partial-reconfiguration.html

can significantly reduce FPGA slice utilization and minimize power consumption. The findings indicate that DPR contributed to a 10.19% reduction in total area and a substantial 76.71% decrease in average power consumption when compared to system designs lacking DPR [31].

PR has found applications in remote laboratories [32]. In a particular project, researchers engineered a scalable remote System-on-Chip (SoC) virtual laboratory designed to offer training support and exercises for students delving into digital and embedded system design. Their utilization of partial reconfiguration aimed to enable "hardware multitasking", a feature that empowered their remote lab to concurrently execute up to four distinct FPGA designs from students [33].

Despite extensive research, we discovered a gap in remote laboratories utilizing the unique capability of partial reconfiguration for SDR projects. One of our objectives is to facilitate advanced FPGA-focused students in learning the intricacies of designing PR modules, while simultaneously offering radio frequency students the opportunity to delve into the creation and synergy of various components within a radio system.

3 System Design

This section provides an overview of the current RHL-RELIA system setup, elucidating its structure and components, in Section 3.1 and our plan to integrate partial reconfiguration into the existing architecture in Section 3.2.

3.1 RHL-RELIA System

The RHL-RELIA system comprises a user-friendly web interface, a server overseeing the allocation of SDR combinations per session, and a physical lab housing all hardware components such as SDRs, Raspberry Pis, and servers. Through this setup, students can upload their SDR designs from GNU Radio. Subsequently, the server uploads these programs onto Raspberry Pis, which then program the connected SDRs via USB cables. Once configured, users can observe and analyze real-time outputs and actively engage with the SDRs [20].

3.2 Integrating Partial Reconfiguration into the Architecture

To integrate partial reconfiguration into the RHL-RELIA network, there are two distinct architectural models that facilitate this feature: one based on a user interface (UI) and another using a command-line approach.

User Interface (UI)-Based Architecture The first approach centers on integrating the Vivado Design Suite, an intuitive user interface application developed by Xilinx[8]. This tool empowers users to craft FPGA programs tailored for their System-on-Chips (SoCs), encompassing the Z7000 series among others.

[8] https://www.xilinx.com/products/design-tools/vivado.html

Students are required to craft their partial reconfigurable FPGA programs using Vivado on their personal computing devices. This process encompasses project creation, design implementation, and synthesis. Upon error-free completion, students can generate bitstreams primed for uploading onto FPGA boards across different SDRs. Subsequently, these bitstreams can be uploaded onto RHL-RELIA's web user interface. Within Vivado's hardware manager window as depicted in Figure 1, students can select and upload their preferred bitstreams, either complete programs or partial programs, onto remote SDRs. The resultant output is promptly accessible for viewing on the website.

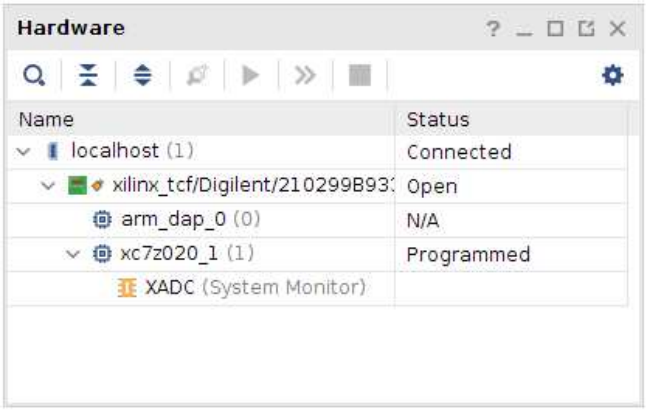

Fig. 1: Hardware manager window in Vivado Design Suite 2020.1.

UI-Based Hardware Setup The hardware setup for UI-based partial reconfiguration involves an X86 computer, a JTAG programmer compatible with Zynq FPGAs, an SDR device, and a Raspberry Pi. The X86 architecture-based computer is essential since Vivado exclusively operates on this architecture[9]. The JTAG programmer receives bitstreams from the X86 computer, performing the task of uploading and programming the Zynq FPGA [27, 34]. Its reliability is crucial; any incorrect FPGA programming often requires a hard reset, involving a complete power cycle to restore the device to its default mode. Once the successful upload of bitstreams is accomplished, the SDR device transmits data to a Raspberry Pi, which then displays the output on the RHL-RELIA website. Figure 2 provides a visual illustration of this interconnected configuration.

Command Line-Based Architecture The second approach focuses on integrating the command prompt within the existing RHL-RELIA system to facilitate partial configuration through Linux commands. Students gain the ability to upload both full and partial bitstreams onto the RHL-RELIA website and subsequently onto the FPGAs using specific commands. Although less intuitive than the UI-based interface, this method offers cost reductions in hardware setup.

[9] https://support.xilinx.com/s/question/0D52E00006hprrpSAA/vivado-on-arm-linux?language=en_US

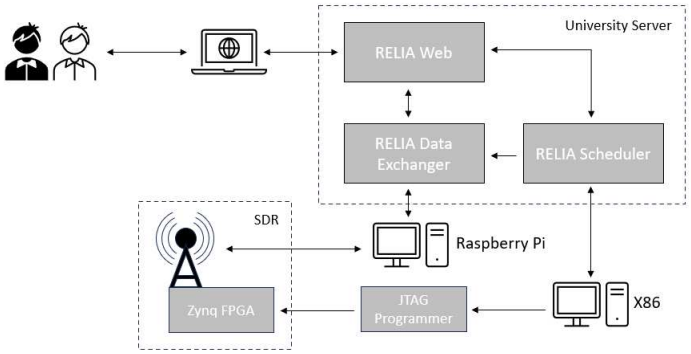

Fig. 2: RHL-RELIA architecture for UI-based FPGA PR.

Command-Based Hardware Setup The setup cost for a command-based architecture is lower compared to a UI-based architecture due to the exclusion of an X86 computer and JTAG programmer. While both devices serve as vital bridges between the Vivado Design Suite and the FPGA, their necessity diminishes when a UI is not utilized. A command-based architecture solely necessitates a Raspberry Pi and an SDR device. The Raspberry Pi serves dual functions as both a programmer and data collector: it programs the SDR's FPGA via Secure Shell Protocol (SSH), a widely-used network protocol, and retrieves data from the SDR, displaying it on the RHL-RELIA website. The command-based architecture is shown in Figure 3.

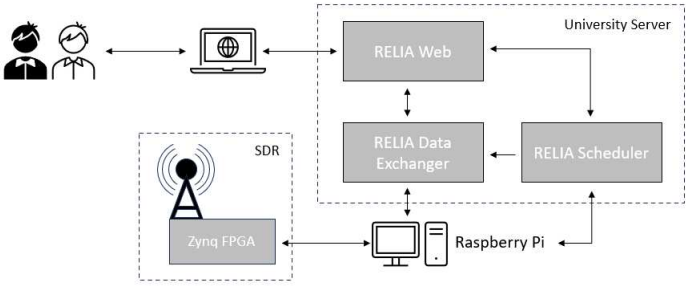

Fig. 3: RHL-RELIA architecture for command line-based FPGA PR.

4 Results

This section outlines the strides made by us in advancing the RHL-RELIA system to facilitate students in conducting real-time partial reconfiguration on SDR devices.

4.1 Testing Partial Reconfiguration: Evaluating Blackboard for UI-Based Architecture

Before experimenting with PR on an SDR, we initially assessed the feasibility of executing PR using the UI-based architecture on an economical evaluation board called Blackboard. This board integrates a Zynq 7007 SoC and incorporates a USB 2.0 transceiver capable of programming the FPGA[10].

The assessment starts with two basic calculator programs: a 4-bit calculator and an 8-bit calculator. Both programs allocate a region for PR. This segment can perform either addition or subtraction. Both full and partial bitstreams are uploaded onto the Zynq FPGA utilizing the hardware manager within Vivado. Initial findings are depicted in Table 1.

Both programs yield a complete bitstream of approximately 2 MB and partial bitstreams averaging around 100 KB. This leads to an average of 5.1% reduction in file size compared to a complete program with embedded child (partial) bitstreams. Consequently, the upload time for a partial bitstream is only 50.2% to 57.7% of that required for a full bitstream.

Table 1: Blackboard PR Program Size & Upload Time Comparison

	4-Bit Calculator			8-Bit Calculator		
	Full Bitstream	Child 1	Child 2	Full Bitstream	Child 1	Child 2
File Size (KB)	2036	104	104	2083	106	106
Upload Time (s)	4.09	2.12	2.36	4.40	2.34	2.21

4.2 SDR Selection

After confirming the feasibility of PR using Vivado on an economical evaluation board, we proceeded to choose the initial SDR platform. Our selection criteria favored an SDR with a programmable Zynq FPGA and an active community. After thorough research, we settled on the Red Pitaya.

Red Pitaya, often abbreviated as RP (to avoid confusion with PR for partial reconfiguration), stands as an open-source SDR renowned for its compact size, affordability, and robust capabilities. The engineers at Red Pitaya are not only open to exploring new concepts but also provide valuable assistance. Two RP models gained our attention: STEMlab 125-14[11] and SDRlab 122-16[12]. In comparison to the entry-level model, the STEMlab 125-14, the slightly upgraded SDRlab 122-16 boasts a more powerful Zynq 7020 SoC. This advanced model offers a substantial increase in specifications, including 204% more logic cells, 133% more block RAMs, and 175% more DSP slices. Consequently, the SDRlab 122-16 became our primary choice. For a detailed comparison between the STEMlab 125-14 and SDRlab 122-16, refer to Table 2.

[10] https://www.realdigital.org/hardware/blackboard
[11] https://redpitaya.com/stemlab-125-14/
[12] https://redpitaya.com/sdrlab-122-16/

Table 2: Red Pitaya STEMlab 125-14 & SDRlab 122-16 Specifications

	STEMLab 125-14	SDRlab 122-16
SoC	Zynq 7010	Zynq 7020
Connectivity	USB 2.0, 1 Gb Ethernet	
Number of RF Input Channels	2	
Number of RF Output Channels	2	
RF Input Bandwidth	DC - 60 MHz	300 kHz - 550 MHz
RF Output Bandwith	DC - 60 MHz	300 kHz - 60 MHz
Input Sampling Rate (MS/s)	125	122.88
Output Sampling Rate (MS/s)	125	122.8
Input Resolution (bit)	14	16
Output Resolution (bit)	14	

4.3 Analysis: SDR PR via Vivado vs. Command-Line Approach

Next, we conducted PR experiments on the Red Pitaya, employing Vivado and command-line approaches to assess the feasibility of the previously proposed architectures. The first subsection provides an overview of the PR outcomes on a Red Pitaya SDRlab 122-16 utilizing Vivado, while the second subsection outlines the results achieved through command-line PR execution.

SDR PR with Vivado Hardware Manager The process involved initiating an SDR project using a manufacturer-provided program and modifying it[13]. Subsequently, Vivado's hardware manager facilitated the upload of generated bitstreams onto the Red Pitaya. The hardware setup comprised an X86 Windows computer, a JTAG-HS3 programmer cable, and a Red Pitaya SDRlab 122-16. The JTAG-HS3 is an inexpensive programmer cable compatible with most Zynq FPGAs, including the Z7020 on the Red Pitaya[14]. It is available at approximately 60 USD. In contrast, the official Platform Cable USB II programmer costs nearly 300 USD (the prices are of January 2024).

The primary limitation of the JTAG-HS3 is its pinout disparity. While it has a total of 14 pins, there are only six JTAG pins on the Red Pitaya. Consequently, a 14-to-6-pin converter becomes necessary. However, these converters are pricey compared to non-specialized alternatives. To economize, we devised a converter by assembling a breadboard and jumper wires, detailed in Figure 4.

Upon programming the Red Pitaya using a PR program, data on file sizes and resource usage were gathered and summarized in Table 3. Opting for PR rather than generating complete programs containing both child programs resulted in a file size reduction exceeding 5%.

[13] https://redpitaya-knowledge-base.readthedocs.io/en/latest/learn_fpga/4_lessons/top.html#lessons

[14] https://digilent.com/shop/jtag-hs3-programming-cable/

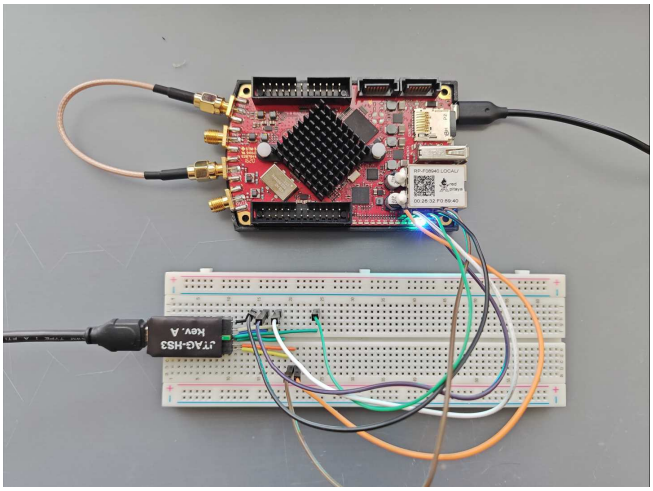

Fig. 4: JTAG-HS3 and Red Pitaya connection.

Table 3: Red Pitaya PR Program Size Comparison

	Program 1			Program 2		
	Fulll Bitstream	Child 1	Child 2	Full Bitstream	Child 1	Child 2
File Size (kB)	2096	106.3	106.3	4046	209.8	209.8

Additional specifics regarding resource usage for a specific SDR project are outlined in Table 4. The child program's LUT usage accounts for approximately 8.2% of the static program and utilizes about 16.9% of FFs. These figures closely align with the LUT and FF savings observed when transitioning from a non-PR program to a PR program with the same functionality.

Table 4: Red Pitaya PR Program Resource Usage

	Full Program with Empty PR Slice	Child 1	Child 2
LUT	4975	409	411
FF	5359	903	903
BRAM	32.0	0	0

SDR PR with Command Lines We also conducted tests on the architecture using command lines, which presented a less intricate hardware setup compared to using Vivado. We connected a Red Pitaya to a Raspberry Pi 4B using an Ethernet cable.

Upon receiving the bitstreams, the Raspberry Pi initiates an SSH connection into the Red Pitaya using its IP address and transfers all necessary bitstreams onto it. Uploading partial bitstreams onto the FPGA involves asserting a PR flag. We faced challenges when the FPGA froze after uploading partial bitstreams

using this approach. The FPGA becomes non-responsive after modifying the PR regions until it is reprogrammed using a full bitstream. Upon consultation with a Red Pitaya engineer, it was suggested that the issue might stem from using an outdated image.

5 Conclusion and Future Work

This work documents our investigation into integrating PR within the RHL-RELIA remote laboratory. Such a feature represents a high level of specialization in FPGA expertise, not common at the undergraduate level but valuable in graduate classes and professional projects. It underscores the inherent multidisciplinary nature of electrical engineering, necessitating collaborative efforts between FPGA and RF engineers.

Two distinct architectural models were introduced and assessed: a UI-based system and a command line-based approach. The detailed hardware setups and experimentation revealed the potential of PR in reducing upload times onto Zynq FPGAs while significantly optimizing resource utilization.

The UI-based approach configures the FPGA by an X86 computer separate from Raspberry Pis. This provides a high degree of flexibility, allowing changes during SDR execution and fostering a more professional project environment.

Conversely, the command-line method configures remote SDRs entirely by a Raspberry Pi, offering cost savings but lacking the flexibility of the UI-based approach. Users can only upload programs to FPGAs using commands.

As future work, while the Red Pitaya satisfies most requirements for an educational SDR device, exploring options for interoperability among different types of SDRs could enhance the system's capabilities. Due to challenges encountered in the command line-based approach, we will collaborate closely with Red Pitaya engineers to resolve these issues and ensure its functionality.

Acknowledgements

This work is supported by the National Science Foundation's Division Of Undergraduate Education under Grant No. 2141798.

References

1. Wang, L., Wang, J.: Design of laboratories for teaching mechatronics/electrical engineering in the context of manufacturing upgrades. International Journal of Electrical Engineering & Education 59(3), 251–265 (2022), https://doi.org/10.1177/0020720919837856
2. Grout, I.: Supporting access to stem subjects in higher education for students with disabilities using remote laboratories. In: Proceedings of 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV). pp. 7–13 (2015)
3. Love, T.: Addressing safety and liability in stem education: A review of important legal issues and case law 1. Technology Studies 39, 28–41 (09 2013)

4. Wei, C.: Research on university laboratory management and maintenance framework based on computer aided technology. Microprocessors and Microsystems p. 103617 (2020), https://www.sciencedirect.com/science/article/pii/S014193312030764X
5. Hussein, R., Maloney, R.C., Rodriguez-Gil, L., Beroz, J.A., Orduna, P.: Rhl-beadle: Bringing equitable access to digital logic design in engineering education. In: 2023 ASEE Annual Conference & Exposition (2023)
6. Dominik May, Beshoy Morkos, A.J.N.J.H.A.I., Beyette, F.: Rapid transition of traditionally hands-on labs to online instruction in engineering courses. European Journal of Engineering Education 48(5), 842–860 (2023), https://doi.org/10.1080/03043797.2022.2046707
7. Xu, Z., Chen, W., Qu, D., Hei, X., Li, W.: Developing a massive open online lab course for learning principles of communications. In: TALE. pp. 586–590. IEEE (2020)
8. Schnieder, M., Williams, S., Ghosh, S.: Comparison of in-person and virtual labs/tutorials for engineering students using blended learning principles. Education Sciences 12(3), 153 (Feb 2022), http://dx.doi.org/10.3390/educsci12030153
9. Schnieder, M., Ghosh, S., Williams, S.: Using gamification and flipped classroom for remote/virtual labs for engineering students (2 2022), https://repository.lboro.ac.uk/articles/conference_contribution/Using_gamification_and_flipped_classroom_for_remote_virtual_labs_for_engineering_students/19188251
10. Hussein, R., Wilson, D.: Remote versus in-hand hardware laboratory in digital circuits courses. In: 2021 ASEE Virtual Annual Conference Content Access. ASEE Conferences, Virtual Conference (July 2021), https://peer.asee.org/37662
11. Blossom, E.: Gnu radio: Tools for exploring the radio frequency spectrum. In: Linux Journal (2004)
12. Tato, A.: Software defined radio: A brief introduction. In: XoveTIC Congress 2018. XoveTIC 2018, MDPI (Sep 2018), http://dx.doi.org/10.3390/proceedings2181196
13. Șorecău, M., Șorecău, E., Sârbu, A., Bechet, P.: Real-time statistical measurement of wideband signals based on software defined radio technology. Electronics 12(13), 2920 (Jul 2023), http://dx.doi.org/10.3390/electronics12132920
14. Perotoni, M.B., Ferreira, L., Maniçoba, A.: Low-cost measurement of electromagnetic leakeage in domestic appliances using software-defined radios. Revista Brasileira de Ensino de Física 44, e20220009 (2022), https://doi.org/10.1590/1806-9126-RBEF-2022-0009
15. Collins, T., Getz, R., Wyglinski, A., Pu, D.: (2018)
16. Hussein, R., Guo, M., Amarante, P., RodriguezGil, L., Orduña, P.: Digital twinning and remote engineering for immersive embedded systems education. In: Frontiers in Education (FIE) Conference. IEEE, USA (2023)
17. Hussein, R., Chap, B., Inonan, M., Guo, M., Monroy, F., Maloney, R., Alves, S., Kalisi, S.: Remote hub lab – rhl: Broadly accessible technologies for education and telehealth. 20th annual International conference on Remote Engineering and Virtual Instrumentation REV 2023 (2023)
18. Inonan, M., Paul, A., May, D., Hussein, R.: Rhlab: Digital inequalities and equitable access in remote laboratories. In: 2023 ASEE Annual Conference & Exposition (2023)
19. Inonan, M., Hussein, R.: Melody: A platform-agnostic model for building and evaluating remote labs of software-defined radio technology. IEEE Access 11, 127550–127566 (2023), doi: 10.1109/ACCESS.2023.3331399

20. Inonan, M., Chap, B., Orduña, P., Hussein, R., Arabshahi, P.: Rhlab scalable software defined radio (sdr) remote laboratory. 20th annual International conference on Remote Engineering and Virtual Instrumentation REV 2023 (2023)
21. Hussein, R., Chap, B., Inonan, M., Guo, M., Monroy, F.L., Maloney, R., Alves, S., Kalisi, S.J.: Remote hub lab – rhl: Broadly accessible technologies for education and telehealth. In: 20th Annual International Conference on Remote Engineering and Virtual Instrumentation (REV) (2023)
22. Inonan, M., Orduña, P., Hussein, R.: Adapting a remote sdr lab to analyze digital inequalities in radiofrequency education in latin america. Revista Innovaciones Educativas (2023), in press
23. Vipin, K., Fahmy, S.A.: Zycap: Efficient partial reconfiguration management on the xilinx zynq. IEEE Embedded Systems Letters 6(3), 41–44 (2014)
24. Bucknall, A.R., Fahmy, S.A.: Runtime abstraction for autonomous adaptive systems on reconfigurable hardware. In: 2021 Design, Automation Test in Europe Conference Exhibition (DATE). pp. 1616–1621 (2021)
25. Bucknall, A.R., Shreejith, S., Fahmy, S.A.: Network enabled partial reconfiguration for distributed fpga edge acceleration. In: 2019 International Conference on Field-Programmable Technology (ICFPT). pp. 259–262 (2019)
26. Bucknall, A.R., Shreejith, S., Fahmy, S.A.: Build automation and runtime abstraction for partial reconfiguration on xilinx zynq ultrascale+. In: 2020 International Conference on Field-Programmable Technology (ICFPT). pp. 215–220 (2020)
27. Grassi, S., Convers, A., Dassatti, A.: Fpga partial reconfiguration in software defined radio devices. Proceedings of the GNU Radio Conference 5(1) (2020), https://pubs.gnuradio.org/index.php/grcon/article/view/68
28. Bucknall, A.R., Fahmy, S.A.: Zypr: End-to-end build tool and runtime manager for partial reconfiguration of fpga socs at the edge. ACM Trans. Reconfigurable Technol. Syst. 16(3) (jun 2023), https://doi.org/10.1145/3585521
29. Vipin, K., Fahmy, S.A.: Fpga dynamic and partial reconfiguration: A survey of architectures, methods, and applications. ACM Comput. Surv. 51(4) (jul 2018), https://doi.org/10.1145/3193827
30. Pham, K., Koch, D., Vaishnav, A., Georgopoulos, K., Malakonakis, P., Ioannou, A., Mavroidis, I.: Moving compute towards data in heterogeneous multi-fpga clusters using partial reconfiguration and i/o virtualisation. In: 2020 International Conference on Field-Programmable Technology (ICFPT). pp. 221–226 (2020)
31. Hosny, S., Elnader, E., Gamal, M., Hussien, A., Khalil, A.H., Mostafa, H.: A software defined radio transceiver based on dynamic partial reconfiguration. In: 2018 New Generation of CAS (NGCAS). pp. 158–161 (2018)
32. Somanaidu, U., Telagam, N., Kandasamy, N., Nanjundan, M.: Usrp 2901 based fm transceiver with large file capabilities in virtual and remote laboratory. In: International Journal of Online Engineering. pp. 193–200. iJOE (2018)
33. Machidon, O., Machidon, A., Cotfas, P., Cotfas, D.: Leveraging web services and fpga dynamic partial reconfiguration in a virtual hardware design lab. International Journal of Engineering Education 33 (01 2017)
34. Hassan, A., Ahmed, R., Mostafa, H., Fahmy, H.A.H., Hussien, A.: Performance evaluation of dynamic partial reconfiguration techniques for software defined radio implementation on fpga. In: 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS). pp. 183–186 (2015)