ECE 484W: Final Project

Author:
Alicia Mand
UIN: 01116191
Lab Due: December 7, 2022

Honor Code:

"I pledge to support the honor system of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagiarism. I am aware that as a member of the academic community, it is my responsibility to turn in all suspected violators of the honor system. I will report to Honor Council hearings if I am summoned."

Digital Signature: *Alicia Mand*

Abstract

The purpose of this project is to build off of all of the preceding projects (projects 1, 2, 3, and 4) in order to display video image data to the onboard LCD screen of the VEEK-MT2S board. Specifically, this project required that the student interface with the board wirelessly using a router and overlay images on a live video feed, and manipulate both the brightness and contrast of that video feed. This was to be implemented using C++ to create the main program and UI, python for the server, and Quartus for setting up the FPGA board. The project was split up into two parts: the first of which was to transfer and overlay an image from the QT UI to the live webcam footage and the second was to adjust the brightness and contrast of the webcam footage using values transmitted from the GUI. The results of this project are presented in this paper.

Contents

List of Figures

1 Introduction

The purpose of this project is to implement all of the knowledge and skills obtained from projects 1, 2, 3, and 4 in order to display webcam footage on the LCD screen of the VEEK-MT2S FPGA board. This is similar to project 4, which required that images be transferred to the board's screen from the QT UI. This project extends beyond this by requiring that the board's web camera be open and running with images being displayed over top of the screen. The second part of this project also required that brightness and contrast values be transferred from the QT UI to the board and implemented in real-time on the web cam.

Communication is to be established via a virtual mini-B USB port (RS-232 port). This was implemented using a LINKSYS AC1000 model dual-band wifi router. In order to ensure a successful transmission of data a python server was used to receive signals from the QT code and transmit them in the appropriate form to the VEEK-MT2S FPGA board. This project required that the student have a good understanding of C, C++, Linux, python, version control, and image processing to ensure a successful experiment.

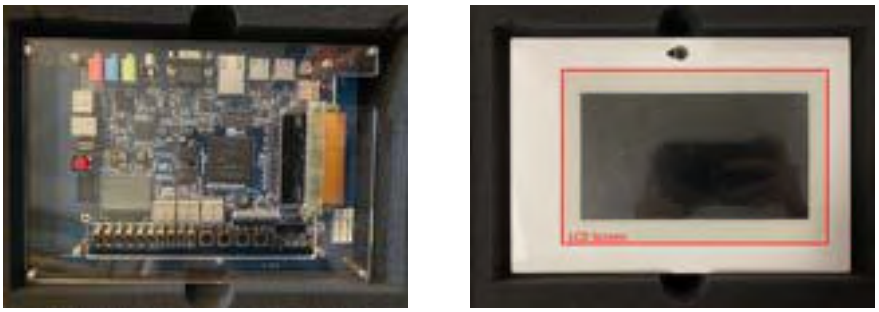
1.1 Objective

The objective of this assignment was to interface wirelessly with the VEEK-MT2S FPGA board from the student's personal laptop computer in order to transfer images for overlay and contrast and brightness values to update the webcam footage. The criteria for completing the project are as follows:

Project Objectives

1. Add an overlay image to every frame of the video from the onboard camera transmitted from the GUI on the user's computer.
2. Adjust the brightness and contrast of the onboard video stream from the GUI.
3. Show results on the onboard LCD screen.

This assignment is to be completed on the VEEK-MT2S FPGA board, specifically, the LCD screen. A figure of this is provided below in figure 1.



((a)) Back of Board ((b)) LCD Screen

Figure 1: Labeled FPGA Board

1.2 Current Technologies

Currently, there are multiple technologies that mimic the goals of this project. For example, video game controllers such as the XBOX controller exactly replicate the underlying behavior of this project. The controller sends inputs to the XBOX console wireless and directly impacts the output of the XBOX console itself by reflecting those inputs as changes to the user's motion or status in-game [1].

Figure 2: Xbox Controller for Interacting Wirelessly with Xbox Console

Controllers such as the one seen in figure 2 allow users to generate inputs for the console and send them to the console system remotely using a wifi connection established between the controller and console. When compared to this project, our computer acts as the controller, and the FPGA board is acting as the console. This behavior will be used as a baseline to determine the success of this project and its implementation.

Other technologies, such as the filters seen in apps such as Instagram, Snapchat, and TikTok utilize image-processing software, similar to this project. The primary difference is that these apps work in real-time and make use of the user's These apps take a live image of the user and edit it in real-time similar to how adjustments made on the student's PC are reflected on the FPGA board's image for this project. However, these technologies are more advanced than the scope of this project and instead also implement face-tracking features to adjust the image processing to be unique and optimized for each individual user. Additionally, this software is in real-time and does not require that the user update the image by selecting a "Transfer Image" button [2]. An example of one of these real-time image-processing software, specifically Snapchat, is provided below in figure 3.

Figure 3: Example of Real-Time Image Processing Software (SnapChat)

2 Design Methodology

The interface was designed to establish communication between the contrast slider program and the FPGA board. When the contrast slider had its values updated, the 7-segment LED on the board updated to reflect this value. For reference, a labeled image of the communication between the contrast program and the FPGA board is provided in figure 4 below.

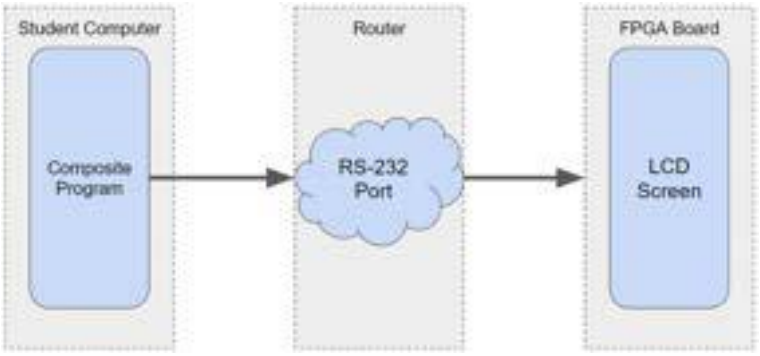


Figure 4: Communication between Computer and FPGA Board

In order to implement this, the student had to first design a program that allowed the upload of an image, adjusted the contrast of the image, and saved the values locally to the program. In this program, the system first tracks the initial values of the image and then the values of the slider once they have been updated. Once the slider has been updated, the contrast values are updated using histogram equalization, and a factor is taken from the contrast values. The factor is multiplied against the relative weight of each contrast value of the image's pixels in order to obtain a new contrast value, this is shown in equation 1 below. In order to adjust the brightness, a factor was calculated according to equation 2 below.

$$B' = (1 + \frac{1}{val}) \cdot w \cdot B_0 \tag{1}$$

$$B' = (1 + \frac{1}{val}) \cdot B_0 \tag{2}$$

where B' is the new brightness of the pixel, val is the value in QT, w is the relative weight of the pixel, and B_0 is the original brightness of the pixel. This new brightness was calculated for each pixel iteratively in order to obtain the new image. A diagram of this is provided below in figure 5.

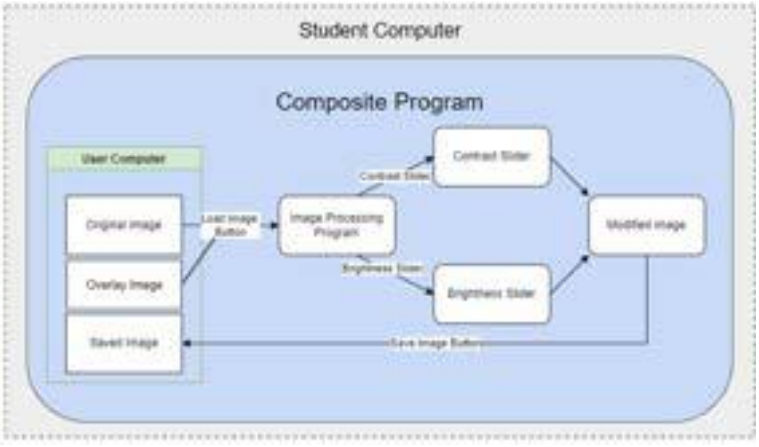


Figure 5: Contrast Program Diagram

In order to allow the upload of an overlay image, the program made use of QT QTcpSocket Library. Pseudocode of this is provided below in listing 1:

```
#include <QTcpSocket>

on_uploadOriginalButton_click()
{
    //Choose image from iamge explorer
    getFile();
    //Set the image in the UI
    ui->OriginalImageLabel->setPixMap(QPixmap::fromImage(originalImage)
        );
}

TransmitImage(QImage imageToSend){
    //Create a socket to transmit the data
     QTcpSocket *socket = new QTcpSocket(this);
    //socket->connectToHost(QHostAddress("192.168.0.110"), 1025);
    socket->connectToHost(QHostAddress([THE IP]), 1025);
    //Create a QByte Array to store the image
    QByteArray byteArray;
    QBuffer buffer(&byteArray);
    imageToSend.save(&buffer, "PNG");
    //Transmit the actual data
    socket->write(byteArray);
    socket->close();
}
```

Listing 1: Pseudocode for Overlay Image

In order to update the webcam footage of the FPGA to display the image overlayed with the image the board first had to interact with the computer hosting processing program as seen in figure 4. The actual implementation of this required the use of a transfer image button and the use of a python server to transmit data to the LCD screen. This is featured below in figure 6.

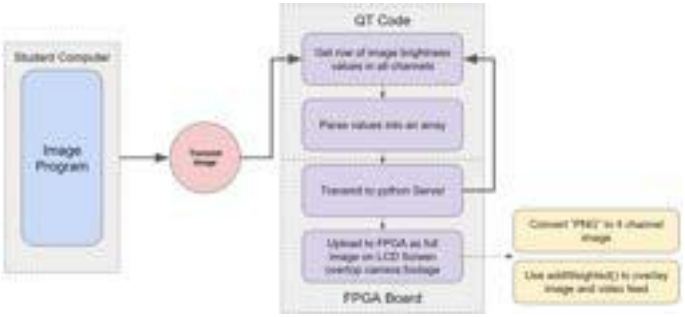


Figure 6: Image Transfer Diagram

As it can be seen in this figure, the data containing the brightness value of each pixel is transferred to the python server row-by-row. This is chosen to decrease the distortion of the image as much as possible. This is transmitted using a simple array. Once each row has been successfully transmitted to the FPGA it is then pushed to the LCD screen of the VEEK-MT2S FPGA board to be displayed using addWeighted() function contained within the camera program.

In order to display the footage on the LCD screen of the FPGA the camera_in.cpp program first checked to see if there was an image to display. If there was not it would simply open the camera of the FPGA and stream it to the LCD screen. In the case that there is an image found

the program uses the implemented addWeighted function to overlay the image on top of the camera feed using 30% opacity. This way, the camera feed is still easily seen below the image. A diagram of this is displayed below in figure 7.

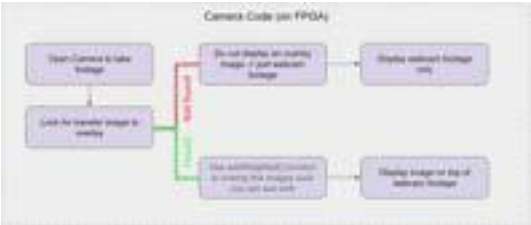


Figure 7: Display Camera Footage Diagram

2.1 Transmitting Data

In order to communicate with the board itself, the computer needed to communicate via a python server loaded onto the FPGA board. This was done using a modified version of Lab 4's python server and a LINKSYS AC1000 model router. The FPGA board was connected to the router via an ethernet cable and the computer wirelessly connected to the FPGA board using a virtual mini-b port through the python server. In order to send updates to the server, QT interfaced with the mini-b port using a transmitData(data) function, this is provided in the appendix for the reader's reference. This function was activated any time the user pressed the transmitImage button. This function communicated with the listening python server and sent updates to the webcam footage each time the contrast or brightness was adjusted on the computer. A visual of this is displayed in figure 6 above (previously mentioned).

2.2 Program UI, Design, and Bug-Fixes

Prior to the start of this project, the main image editing software had been created as part of assignment one. This program featured two sliders for brightness and contrast that adjusted the appropriate values of each pixel in the image. The program also allowed the user to upload an image of their choice, make changes to the image (contrast, brightness, or both) and then save the final image. However, for this particular project the ui was modified in order to accommodate more features. For example, previously, the overlay and contrast/brightness adjusters were implemented as two separate programs. Now, they are combined into one program. A figure of this ui is provided in figure 8 below.

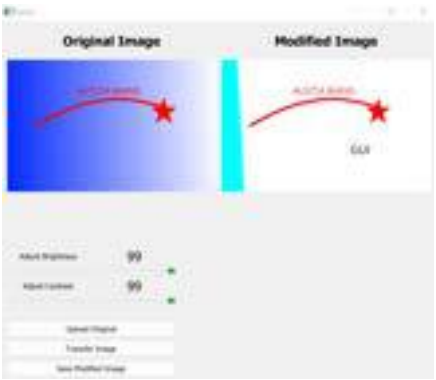


Figure 8: Program UI

This window features the upload and saves image buttons as well as a region for the images to appear. Additionally, the window features two sliders, one for adjusting the brightness and the other for adjusting the contrast of the image. Finally, the program features a "transfer image" button in order to transmit the modified image to the FPGA. Once an image has been uploaded and modified the right-hand side of the window updates to show the modified image.

2.3 Setup

Since this project required the wireless communication between the FPGA board and the computer, it was necessary to setup a python server between the two. This python server ran on the FPGA and waited for data to be sent via a virtual mini-USB port on the computer. A figure depicting the setup is shown below in figure 9.

Figure 9: Setup of FPGA And Computer (Grey Cable is Keyboard)

The figure above depicts the computer and FPGA with wireless communication between the two. Note: the grey cable is the keyboard connected to the FPGA in order to write input to the LXTerminal on the FPGA itself. The Black cable is the ethernet cable connecting the FPGA to the LINKSYS router. As it can be seen here the python server is waiting for images to be uploaded with the "READY" prompt at the top of the screen, the "data received" and "data transferred" prompts indicate that data has been sent to the FPGA and saved to it in that order using TCP Protocol.

3 Results and Analysis

After finishing the programming for this project, it was tested using the onboard LCD screen and QT program on the VEEK-MT2S FPGA board. The input image used for this was one picturing a blue-white color gradient and the students name in red, this was the student's own creation. It is important to note that this image is of the same size as the output of the webcam footage in order to avoid size errors in the camera_in.cpp's addWeighted() function. This image is featured in figure 7 (previously mentioned).

Upon clicking one of the "transfer image" buttons on the program's UI on the user's computer, the QT Program would interface with the FPGA board wireless via the python server and update the onboard LCD screen already featuring the webcam footage to reflect this. If no image was detected the camera_in footage would feature just the live webcam feed. An image of this is provided below in figure 10.

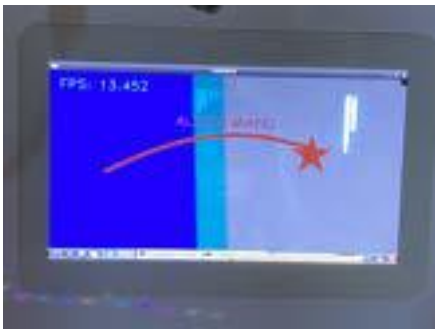
In this example, the frames per second can be observed to be 20.126 FPS.

Figure 10: Camera Output with No Image Detected

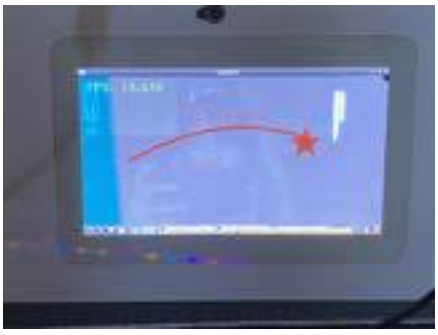
Once an image was detected by the FPGA after transferring, it would be sent to the camera_in.cpp program as an image to be overlayed on top of the video feed. This is seen in figure 11 below.

Figure 11: Image Overlayed on Camera Footage

As it can be seen above, the frames per second drop significantly when an additional image is overlayed on top of the initial footage. This is because the addWeighted() function is called to for every frame of the footage during display. In order to display the functionality of the sliders, the contrast and brightness were adjusted accordingly. This is seen below in figure 12.



((a)) Brightness = 0 Contrast = 99



((b)) Brightness = 99 Contrast = 99

Figure 12: Webcam footage with Brightness and Contrast Adjusted

When transferring and overlaying images, the programs overall were extremely successful.

This means that when sending an image from the UI the python server was able to successfully save this image and send it to the camera_in.cpp program to be overlayed on top of the camera footage. Additionally, when changes were made to the contrast or brightness, these values were reflected in the camera footage on the LCD screen of the FPGA. Overall it can be said that this lab was very successful in accomplishing its goals.

4 Alternative Design

The constraints of the problem require that the framerate of the video footage be displayed overtop of the camera feed. While this is useful information, it would also be optimal if the brightness and contrast were displayed overtop of the video feed. In order to implement this, the image send to the FPGA was modified to display the brightness and contrast values. Another aspect of alternative design was to implement a saveImage button that allowed the user to save the image after modifying its brightness and contrast values. A figure depicting this is displayed in figure 13 below.

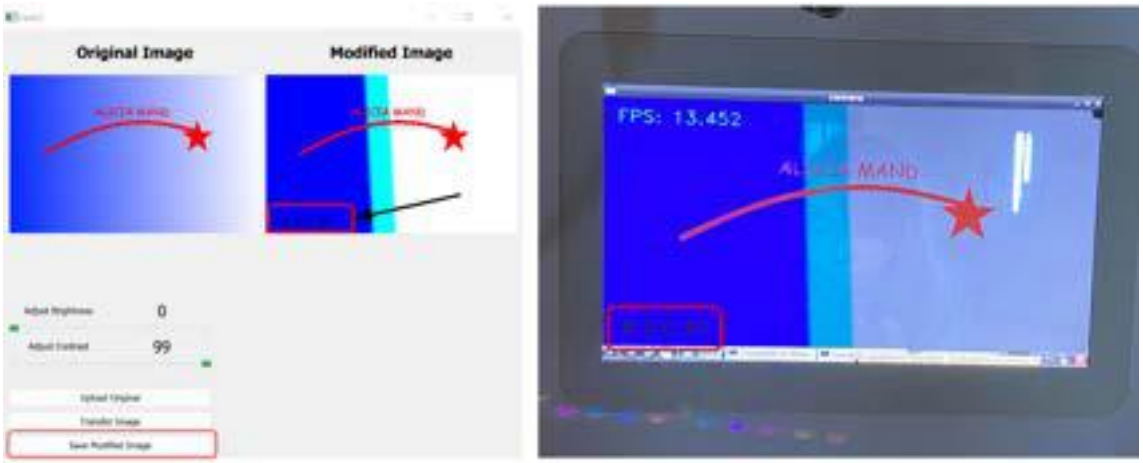


Figure 13: Alternative Design

From these results it is very easy to see that implementing the brightness and contrast onto the FPGA was easy and sucessfuly done, allowing for the resulting webcam footage to display the contrast and brightness. For this particular example, brightness was set to '0' and contrast was set to '99.'

5 Additional Considerations in Design Process

5.1 Economic Considerations

This project was designed with the end user's wallet in mind. All the software provided for this project was free-to-use and open-source. For example, the Win32 Disk Imager software is completely free and allows users to flash the SD card without worrying for paying for software. Additionally, the development software such as QTCreator were all free to use.

5.2 Cultural Consideration

The engineering code of ethics states that discrimination of any kind is not to be tolerated [7]. With this in mind, the program was designed to be user-friendly for anyone to use it. Because of this the program is designed to be used across any culture, regardless of country of origin.

5.3 Environmental Consideration

Similar to the Economic consideration the amount of power used to run the program was considered in order to mitigate the environmental impact of the project. Because of this environmental impact was considered along with public health to limit the use of excessive power and increase the environmental friendliness of the program, particularly by using LEDs over incandescent bulbs for the light output [6].

5.4 Societal Consideration

It is well known within the scientific community that the color red is the easiest color to see on the light spectrum [8]. Because of this, in order to increase accessibility, the color red was chosen for all light outputs when considering the societal consideration of this project. Additionally, the code was commented thoroughly such that the end user would be able to edit it as necessary/needed.

6 Broader Impact

6.1 Economic Impact

According to the Terasic shop website, the particular FPGA board used for this project is just under $900 [5]. Meaning that the accessibility of this particular project, if implemented on this board is quite low. According to the WorldBank database, the global median income is approximately $10,000/year. Meaning that on average, a person's take-home income is approximately $833 per month [9]. This means that it would take at the very **least** an average person two months to afford the FPGA board required by this project. Meaning that economically, this project is not very accessible nor is it prosperous due to its high cost and investment requirements.

6.2 Environmental Impact

The VEET-MT2S FPGA board utilizes silicon chips across the entirety of the board. It is well-known that the mining of silicon is terrible for the environment. Beyond this, however, in a paper published by Ashutosh Mishra in the Journal of Geography and Regional Planning, it is noted that the mining of silicon specifically has had negative impacts on the environment surrounding it. Beyond the obvious deterioration of the environment, the mining of silicon has resulted in land deterioration, loss of biodiversity, and pollution in the regions surrounding the mines. Mishra also notes that the mining of silicon has harmed the local environment in the surrounding villages and caused an overall lack of standard of living for the inhabitants [10]. Because of this, due to the materials used to produce the FPGA board, it goes without question that the production and use of the VEET-MT2S has a negative impact on the environment.

6.3 Global Impact

FPGAs are among the leading topics for machine learning and artificial intelligence (AI). According to Jeremy Fowers of Microsoft, FPGAs are currently being used by large corporations such as the leading technology company: Microsoft in order to design and implement AIs capable of learning. These "Real-time" AIs are able to exploit the programmability of FPGAs by reprogramming themselves in order to "learn" and become more intelligent [11]. Because of this, the global impact of FPGAs goes unquestioned. Since they are a leader in the development of AI, which is the next step in technological evolution, they play a key role globally in computer engineering research.

6.4 Societal Impact

Outside of the scope of this project, logic gates and basic FPGA programming are used across all industries. For example, FPGAs are used in education across the world to improve access and understanding in the field of computer engineering. More specifically, in Taiwan, the number of students participating in FPGA training courses has increased over the last 7 years with the increased availability of FPGAs [12]. Just from an increase in course attendance alone, it should be clearly seen that the societal impact of FPGAs is one in which education is unarguably positively impacted.

References

[1] “Xbox Wireless Controller: Xbox,” Xbox.com. [Online]. Available: https://www.xbox.com/en-US/accessories/controllers/xbox-wireless-controller. [Accessed: 20-Oct-2022].

[2] E. Moreau, “5 apps like Snapchat with face-tracking filters,” Lifewire, 25-Dec-2020. [Online]. Available: https://www.lifewire.com/snapchat-alternatives-with-face-tracking-filters-4075998. [Accessed: 03-Nov-2022].

[3] “Where the world builds software,” GitHub. [Online]. Available: https://github.com/. [Accessed: 21-Oct-2022].

[4] Jagdale, “Top 5 microcontroller development boards of 2021,” Embedded Computing Design. [Online]. Available: https://embeddedcomputing.com/technology/open-source/development-kits/top-5-microcontroller-development-boards-of-2021.

[5] “Terasic,” TERASIC Inc. - expertise in FPGA/ASIC design. [Online]. Available: https://www.terasic.com.tw/en/. [Accessed: 03-Oct-2022].

[6] “LED lighting,” Energy.gov. [Online]. Available: https://www.energy.gov/energysaver/led-lighting#:~:text=LED%20is%20a%20highly%20energy,times%20longer%2C%20than %20incandescent%20lighting. [Accessed: 03-Oct-2022].

[7] NSPE Code of Ethics for Engineers, Code of Ethics

[8] “A guide to color: New Mexico State University - be bold. shape the future.,” A Guide to Color | New Mexico State University - BE BOLD. Shape the Future. [Online]. Available: https://pubs.nmsu.edu/_c/C316/. [Accessed: 03-Oct-2022].

[9] “Adjusted net national income per capita (current US$),” Data. [Online]. Available: https://data.worldbank.org/indicator/NY.ADJ.NNTY.PC.CD. [Accessed: 03-Oct-2022].

[10] M. Ashutosh, “Impact of silica mining on the environment,” Journal of Geography and Regional Planning, vol. 8, no. 6, pp. 150–156, 2015.

[11] J. Fowers et al., "A Configurable Cloud-Scale DNN Processor for Real-Time AI," 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), 2018, pp. 1-14, doi: 10.1109/ISCA.2018.00012.

[12] Yu-Tsang Chang, Yu-Te Chou, Wei-Chang Tsai, Jiann-Jenn Wang and Chen-Yi Lee, "FPGA education and research activities in Taiwan," 2002 IEEE International Conference on Field-Programmable Technology, 2002. (FPT). Proceedings., 2002, pp. 445-448, doi: 10.1109/FPT.2002.1188732.