The Keyboard Keylogger

by

Michael Martin / KokHwa Khor

A Technical Report Submitted to the Faculty of
Computer Science and Engineering
University of Nebraska-Lincoln

Submitted in partial fulfillment for the requirements of
CSCE 436 - Advanced Embedded Systems
April 28, 2021

Chapter 1: Design Goals **3**
1.1 Need Statement 3
1.2 Marketing Requirements 3
1.3 Level-0 Graphic Diagram 3
Chapter 2: Detailed Design **5**
2.1 Level-1 Diagram 5
2.2 Datapath and Control 6
2.3 Calculations 7
2.4 Technical Requirements 7
2.5 Bill of Materials 8
Chapter 3: Implementation **9**
3.1 Milestone I 9
3.2 Milestone II 9
3.3 Final Implementation 10
References **12**
Appendix A: Running the Project **13**
Appendix B: Project Git Repository **14**

Chapter 1: Design Goals

1.1 Need Statement

There's been a need in surveillance technology to monitor and record keystrokes on a keyboard. This technology is often used in business to troubleshoot, improve user experience, or monitor employees. It is also used by hackers to monitor activity and obtain personal data, such as passwords or credit card information.

1.2 Marketing Requirements

We built an embedded system which acts as a keyboard keylogger device. This device can capture keyboard presses from a USB keyboard, decode it, and store it as memory in a Nexys Video FPGA board. A maximum of 4096 data could be stored in the BRAM. It is able to export all the data in 1024 blocks and print it to a terminal or Tera Term. The Nexys Video board can be booted from an SD card to act as a standalone device without having to plug it into a computer and program it everytime.

1.3 Level-0 Graphic Diagram

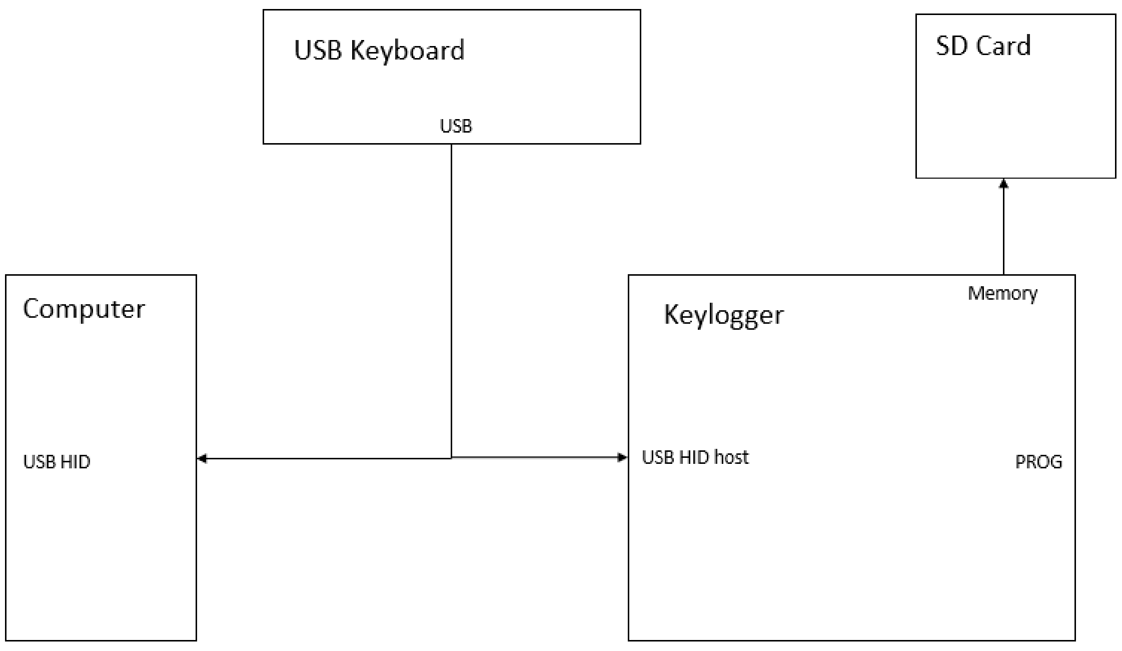


Figure 1: Shows a parallel usb device.

Inputs	USB keyboard through USB HID, program in SD Card
Outputs	Key presses from the keyboard
Behavior	Key presses on the keyboard will be logged and decoded. It will then be stored as memory in the BRAM, with a maximum of 4096 data. The board can be programmed from an SD card.

Table 1: Function Table

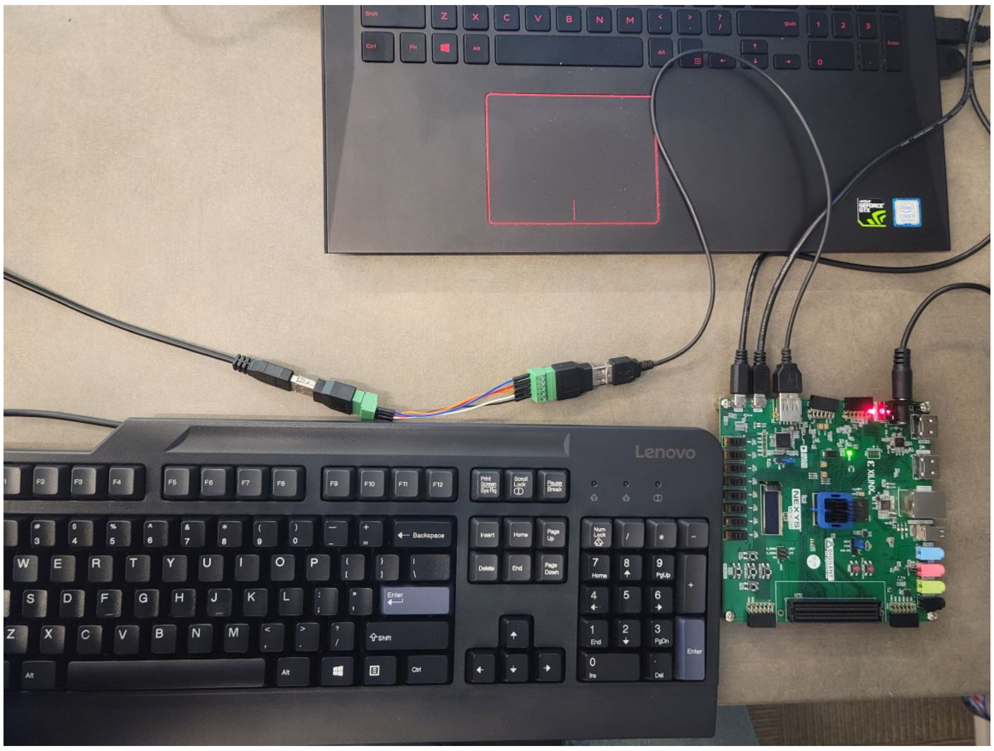


Figure 2: Hardware Connection

Chapter 2: Detailed Design

2.1 Level-1 Diagram

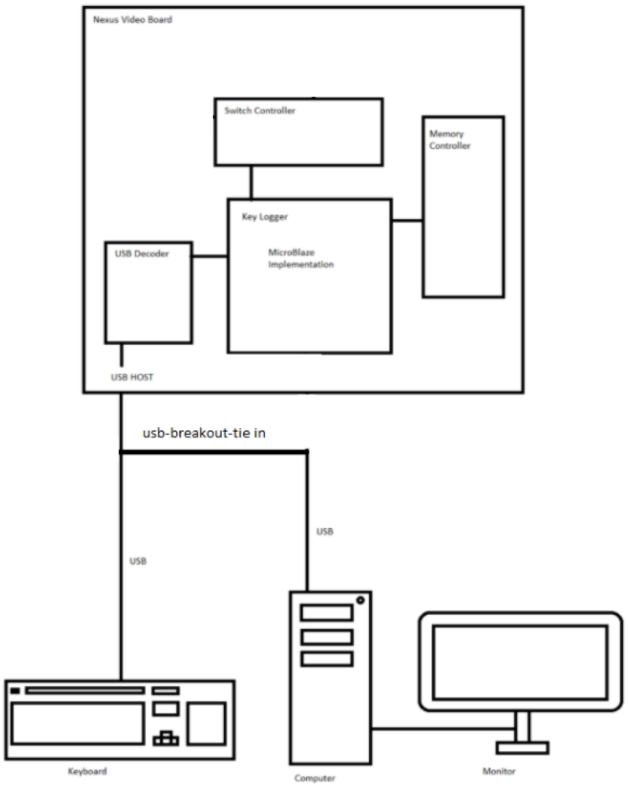


Figure 3a: Proposed Connection methods and basic layout of hardware design

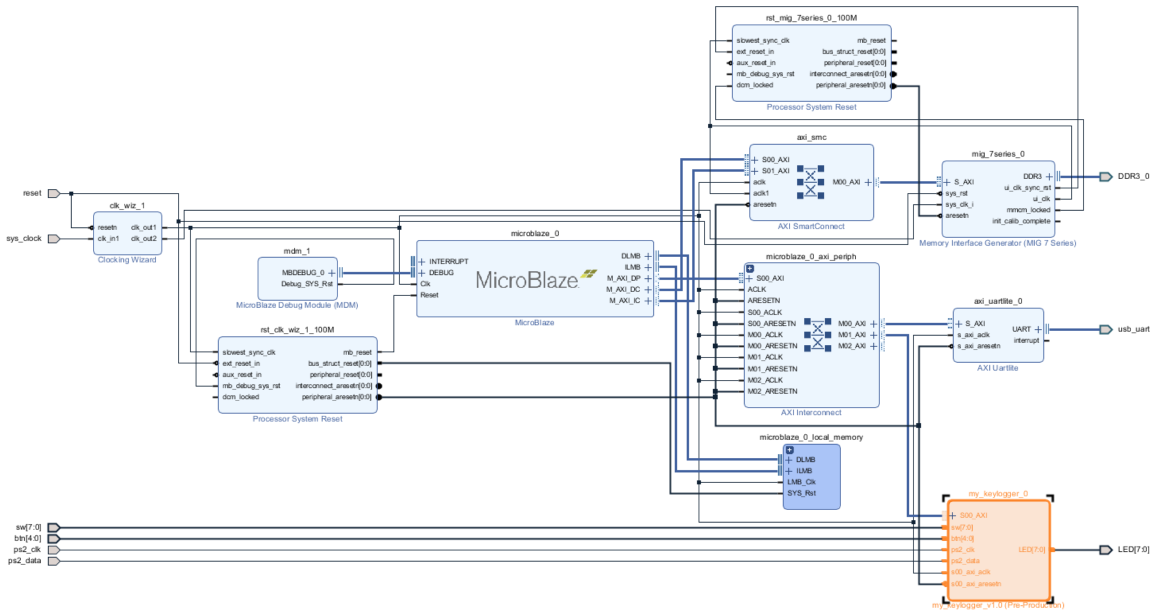


Figure 3b: Block Diagram of layout design.

2.2 Datapath and Control

```
entity keylogger_datapath is
  port(
    clk : in std_logic;
    reset_n : in std_logic;
    ps2_clk, ps2_data : in std_logic; --pins W17 and N13 from USB HID
    data_out : out std_logic_vector(7 downto 0); --out data
    ready : out std_logic;
    read_address : in std_logic_vector(11 downto 0);
    ctrl : in std_logic_vector(7 downto 0); --control for LED's
    LED : out std_logic_vector(7 downto 0);
    sw : in std_logic_vector(7 downto 0); --switches
    btn : in std_logic_vector(4 downto 0) --buttons
  );
end keylogger_datapath;
```

Figure 4: VHDL Code for Datapath Entity

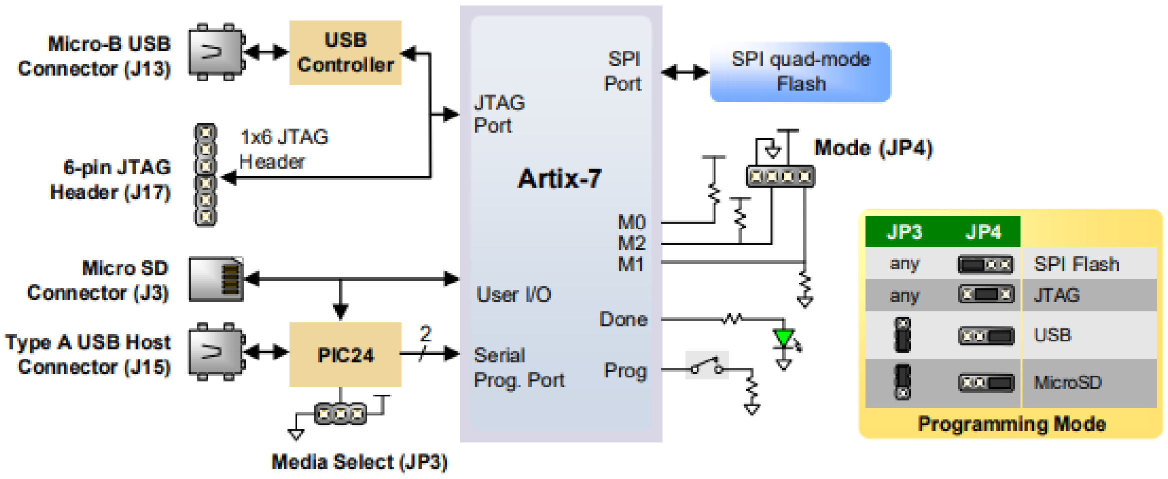


Figure 5: Configuration of Jumpers on the Nexys Board for using SD Card Programming

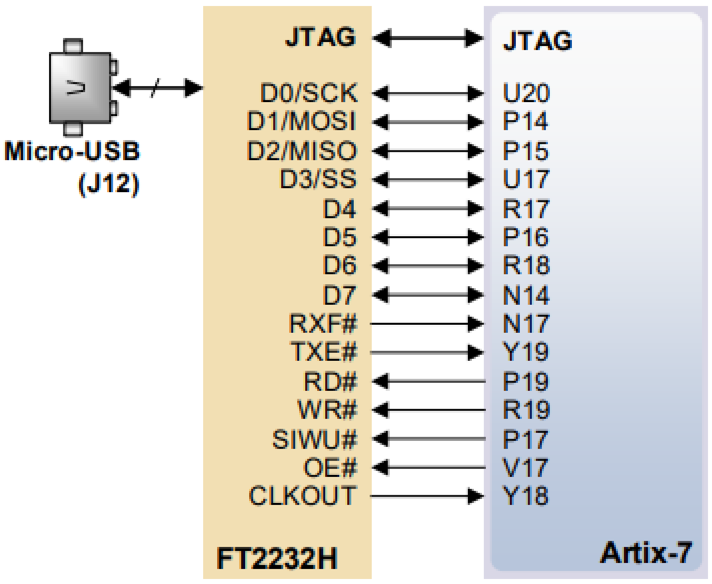


Figure 6: Micro-USB Pin-out

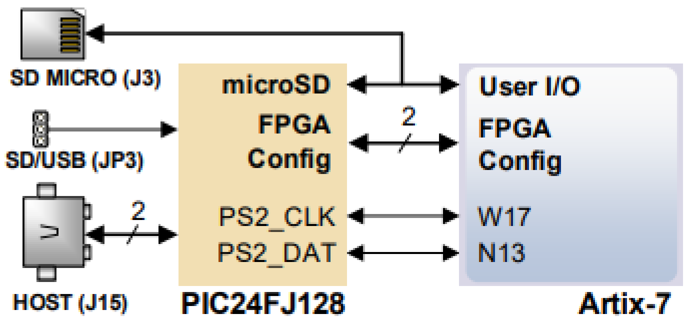


Figure 7: PIC24FJ128 is an on board chip that decodes USB to PS2. Image shows how wires connect to signals that can be tied into hardware design.

2.3 Calculations

User presses a key on the keyboard, this sends a keyboard scan code to the Nexys Video. This scan code is read and transmitted to a terminal application via the USB-UART bridge. A scan code pair of F0xx indicates that key xx has been released.

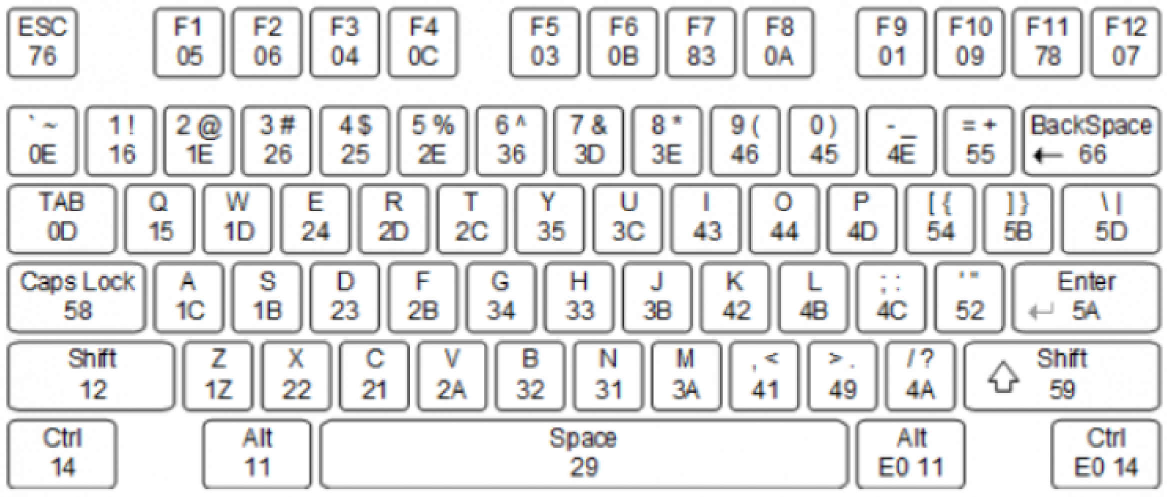


Figure 8: Keypress to ASCII conversion

2.4 Technical Requirements

1. Being able to interface with a USB keyboard from the Nexys Video board through MicroBlaze.
2. Decode the keypress from the keyboard using HID.
3. Store it as memory in MicroBlaze BRAM implementation.
4. Read the memory from Tera Term.
5. Program the Nexys video board via SD card on power up.

2.5 Bill of Materials

Parts already owned

- USB keyboard
- Nexys Video Board
- Laptop/Computer
- SD card

Parts needed to purchase/rent

- USB female breakout connectors ~$20
- USB to USB connector cable ~$10

Chapter 3: Implementation

3.1 Milestone I

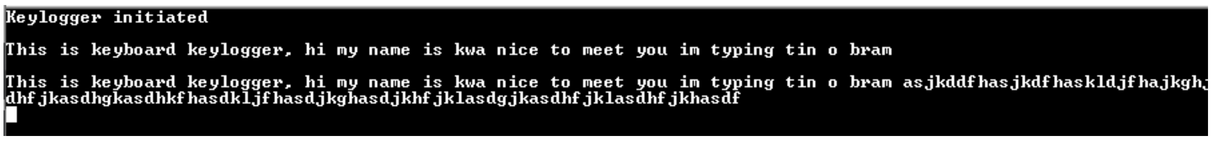
1. Wrote the vhdl code for keyboard decoder and built an SDK on Vivado to be able to see the output from the keyboard.
2. Hooked up the keyboard, Nexys Video, and computer as shown in Figure 1 above. Had our hardware designed and implemented. Were able to capture keypresses and display it on Tera Term.
3. Milestone 1 was achieved. We had to pivot our original plan of using RGB keyboard because Nexys Video can only interface with USB/PS2 keyboard via HID. Hence, we were unable to even connect the RGB keyboard to Nexys Video and have data going through it.

Video demo for Milestone I: https://youtu.be/RBONUyHvFG0

3.2 Milestone II

1. Since we got the Nexys Video to register keypresses from a normal USB keyboard and display it through Tera Term, we are aiming to save the data in BRAM as memory and be able to export all the data from another computer. We achieved this functionality and the main issue we had was in the constraint file. The issue we found with why our keypresses were not registering was that the ps2_data and ps2_clk signals needed to be set to pull-up mode in the constraints file, once this was done we were able to get 1 of our keyboards to register as expected.
2. We interfaced this on Tera Term, and it was able to store 4096 ASCII values in total. Values are split into 1024 blocks and can be displayed using the number keys '1' to '4'. Before outputting the data, we have to load it into the array by pressing the 'E' key.

Video demo for milestone II: https://youtu.be/M90FziXlrpE



```
Keylogger initiated

This is keyboard keylogger, hi my name is kwa nice to meet you im typing tin o bram

This is keyboard keylogger, hi my name is kwa nice to meet you im typing tin o bram asjkddfhasjkdfhaskldjfhajkgh
dhfjkasdhgkasdhkfhasdkljfhasdjkghasdjkhfjklasdgjkasdhfjklasdhfjkhasdf
█
```

Figure 9: Keypress output on Tera Term

3.3 Final Implementation

1. We were able to read keypresses from the keyboard, decode it and store it in BRAM in the Nexys Video FPGA board. All memory was stored in the hardware design.
2. The maximum array size that can store the data is 4096 ASCII values, but when it overflows, it will write back to the first data. So it will store the latest 4096 data.
3. The data can be exported over UART when connected via a terminal command. The entire BRAM data is read into an array which can be displayed in 1024 character chunks.
4. The bit file was loaded onto an SD card which could be used to program the Nexys Video FPGA board without plugging it into the computer. This allowed for the board to be powered on and only the keyboard would need to be attached for it to start its base functionality.
5. We weren't able to achieve the functionality to control LED's on the RGB keyboard due to the USB host on the Nexys Video only accepting signals from standard USB/PS2. Since the RGB keyboard we were using was USB type-C to type-A connector, the board was unable to register any keystrokes at all. We were only able to get 1 out of 4 of the keyboards we had available for this to send the proper signalling for the board to recognize.
6. We also couldn't achieve the passive interception of signals. Our plan was to connect the keyboard to the computer through two of the USB type-A to 5 terminal adapters, as shown in the figure below. And the third one will be connected to the Nexys Video FPGA board. This was unfeasible because when we found this complication it would have required a redesign of how our signals were interpreted and captured by the board. The idea to get around this would have been to have the video board act as a dummy USB device and connect to the PC via the UART port. The board would have to respond to device identification exactly how the keyboard would in order to fit with our intended goal. There just was not enough time left after the discovery of this to make it work.

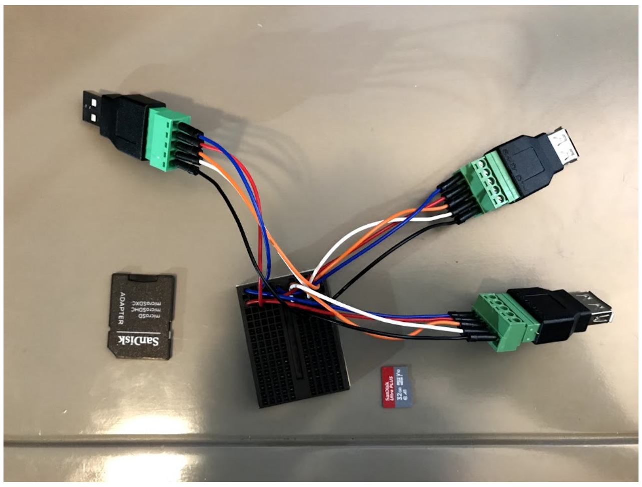


Figure 10: Micro-SD card and USB connectors wired for attempted passive interception.

7. Video Link to Functionality Demo https://youtu.be/q6_3IaAUkzQ

References

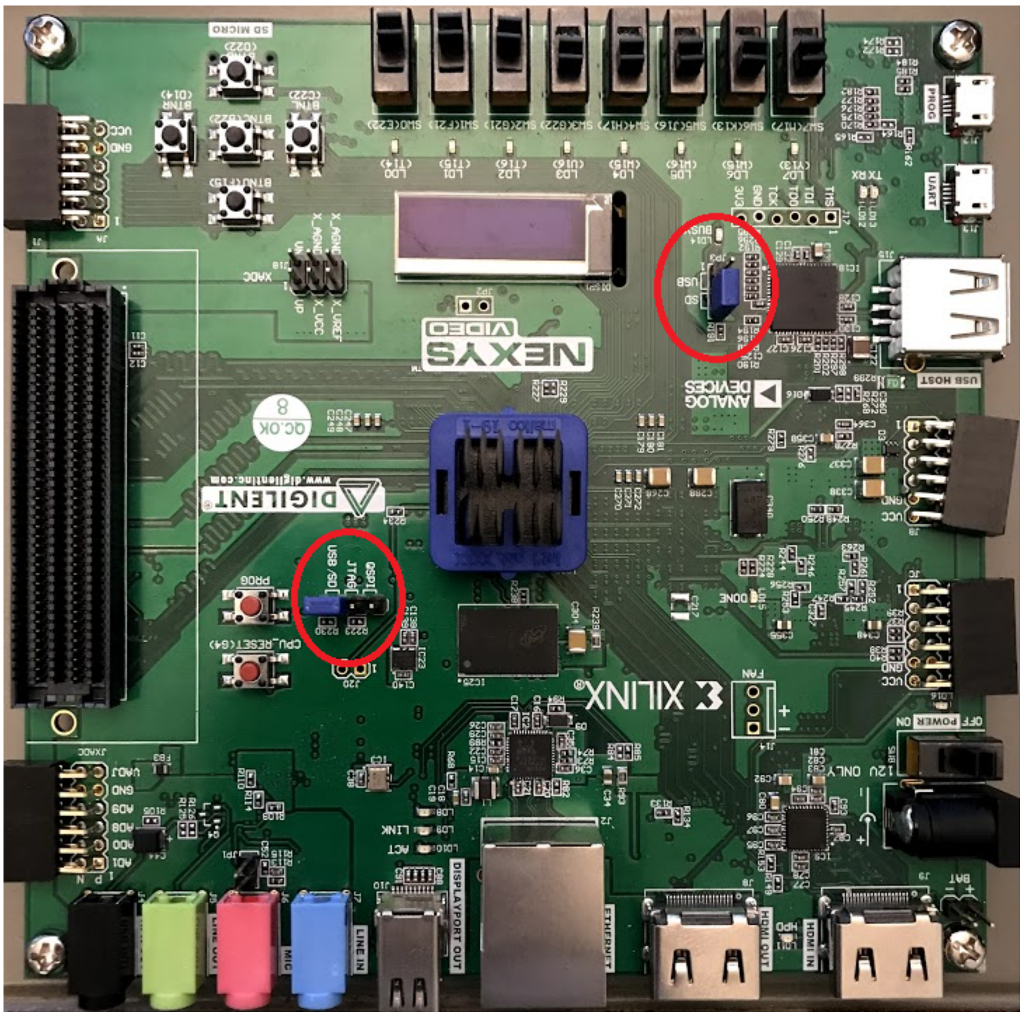
[1] Reference.digilentinc.com. 2020. *Nexys Video™ FPGA Board Reference Manual*. [online] Available at: <https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-video/nexys-video_rm.pdf> [Accessed 29 April 2021].

[2] Engineering and Component Solution Forum - TechForum │ Digi-Key. 2021. *PS/2 Keyboard to ASCII Converter (VHDL)*. [online] Available at: <https://forum.digikey.com/t/ps-2-keyboard-to-ascii-converter-vhdl/12616> [Accessed 29 April 2021].

[3] Brown, A., 2019. *Digilent/Nexys-Video-AXI-PS2-Keyboard*. [online] GitHub. Available at: <https://github.com/Digilent/Nexys-Video-AXI-PS2-Keyboard/blob/master/README.md> [Accessed 29 April 2021].

Appendix A: Running the Project

1. Load the BIT FILE into the root directory of a micro-SD card.
2. Connect the micro-SD card into the SD card slot on the FPGA.
3. Move the jumpers to the positions shown in



4. Plug in USB/PS/2 Keyboard into the USB HOST port.
5. Power the FPGA Board and wait for green ready light.
6. Keypresses will be captured at this point.
7. Connect a PC with Vivado SDK to the UART and PROG micro USB ports.
8. Program the board with the Keylogger SDK project code.
9. Follow commands displayed in the menu for desired functionality.
10. Copy Terminal output for data processing or long term storage.

Appendix B: Project Git Repository

Following is a link to the bitbucket repository for the code created and used in this project. This is a private repo but has been shared with the Professor of this course.
https://bitbucket.org/CloakedGhost07/keylogger/src/main/