PixyCAM UART Interface

List of Authors (Kristof von Czarnowski, Matthew Wener, Luke Pridemore, Randy Wittorp)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI

e-mails: kpvonczarnowski@oakland.edu, mwener@oakland.edu, lpridemore@oakland.edu, randywittorp@oakland.edu

***Abstract*—The purpose of this project is to interface an FPGA with a PIXY CMUCam using the internal serial UART protocol.**

I. Project Description

The goal of this project is to utilize the UART protocol to display information on the FPGA coming from the Pixy Cam. The UART protocol on the FPGA allows for data transfer between two devices using serial communication. The Pixy Cam comes with an onboard algorithm that can track objects based on their color signature. The Pixy Cam provides a x-y coordinate pixel mapping relative to a 320x480 pixel viewport, as well as the pixel height and width of the tracked objects mapping. The purpose of the VHDL program is to parse this information from every bit sent from the Pixy Cam's UART protocol. The positional and dimensional information for the object is to be stored in six 16-bit registers and displayed on the onboard 7-segment display array. A multiplexer controlled by the onboard switches is used to select which stored data is passed through to the 7-segment display. These registers store the x coordinate, the y-coordinate, the width of the object and the height of the pink ball relative to the Pixy Cam.

The motivation behind the project stems from the whole group being interested in data transfer between electronic devices. A member already had the Pixy Cam, so it was decided to investigate uses of that camera via an FPGA. This system has many practical applications, such as tracking the change in dimensions of an object to prevent collisions for an autonomous rover. The Pixy Cam could also recognize a specific color for filtering applications and forward that information for further processing. The interfacing of the Pixy Cam and the FPGA showed how finite state machine could be used to parse data and how to use an oversampling bit generator. The most difficult part was properly storing data coming from the UART parsed signal block. This was also the part of the project that involved the most research.

II. Methodology

The interfacing of the Pixy Cam and the FPGA uses a total of eight VHDL components:

1) UART Parsing Block [2]
2) Clock Divider (sets oversampling rate) [3]
3) Storage Controller Finite State Machine
4) 8-bit Registers [3]
5) 16-bit Registers [3]
6) 4-to-1 MUX w/ 2 Select Lines [3]
7) 7-Segment Display Serializer [3]
8) RGB Led Controller [3]

The top-level design of the project begins with the oversampling clock divider. The data from the Pixy Cam is sent to 8-bit registers to be stored as memory, each corresponding to the data it represents. Then the data that was separated by the UART stream into two-byte size pieces is taken from the two corresponding 8-bit registers and concatenated into a two byte sized piece of data that is stored in a 16-bit register. The four pieces of data (X, Y, H and W) are displayed on the serializer, using a multiplexer to switch between them. The multiplexer is controlled by switches on the FPGA so that all the data can be seen by the user and a corresponding color code is displayed on the RGB for the user to help identify the data being displayed. The checksum and the signature number bytes are also stored in two of the 16-bit registers, but this data is not used for the project.

Figure 1. The PixyCam

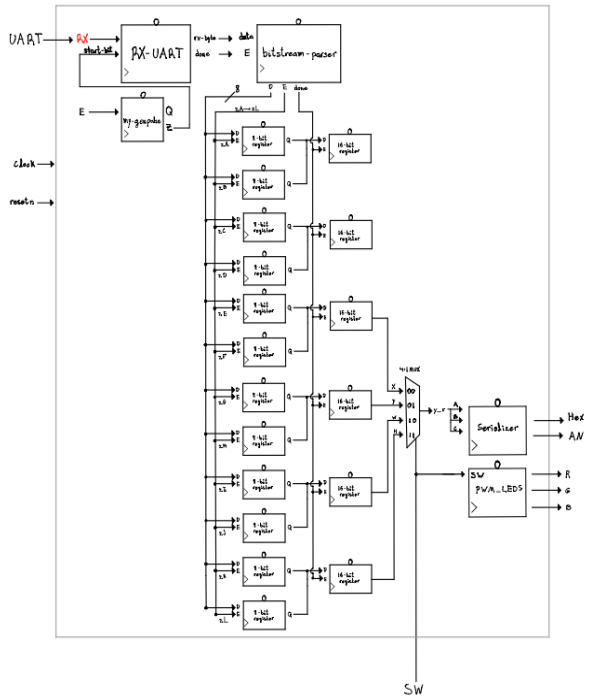

Figure 2. Top level design

The most difficult part of the project revolved around the signal processing. Initially, the group was unsure how to assign data properly to the register coming from the UART parsing block. We overcame this issue by identifying how VHDL can be used to store data at specific point in time relative to our data signals.

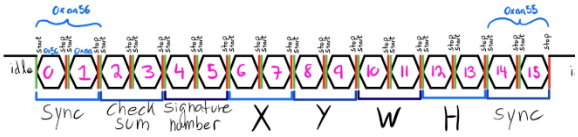

Figure 3. Pixy Cam Data

Figure 3 depicts the how the data is transferred from the Pixy Cam [1]. The signal is high (idle) until a start bit is detected and then the sync, checksum, signature number, X, Y, W, and H bytes are sent and the useful data is stored into several 8-bit registers.

Parsing a UART signal proved to be more difficult than expected. However, the answer to this issue was found online in the form of existing VHDL code that had been written to parse any UART data stream [2].

The baud rate is set on the PixyCAM (in this case, it is 19,200 bits per second). It was found that interpreting the UART signal can be done by oversampling the signal in a FSM. To oversample the UART signal, a frequency of 16 times the baud rate (307 kHz) must be used in a counter.

As the counter starts to increment, starting from zero, every time the start signal is high until it reaches 7 (the center of the start bit). The counter clears, and the state transitions into data collection mode. With the center of the start bit detected, the counter increments another 16 times until it reaches the center of the first data bit. The value is stored, and the process repeats for the number of data bits in the given signal (in this case 8 bits). Once data collection is complete, the stored data is assigned to an output and waits 16 ticks so that it is in the stop bit to wait for new data.

A finite state machine, bitstream_parser, had to process the stored data from the RX UART component which is used to assign data to the correct registers.

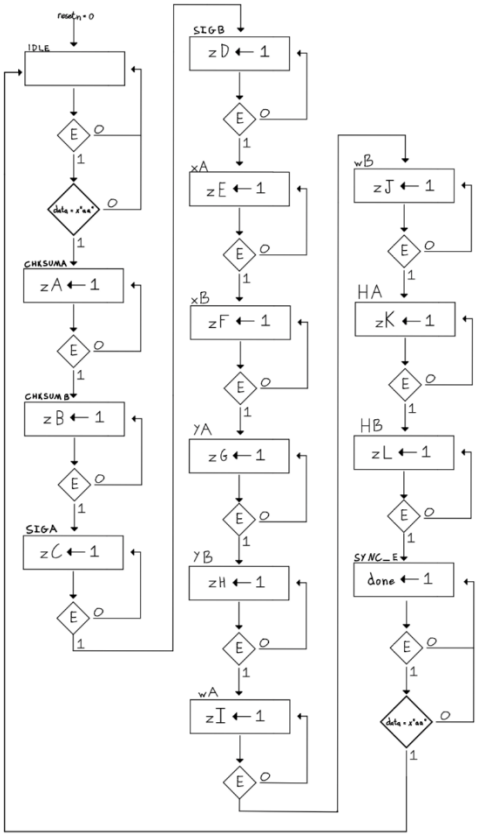

Figure 4. FSM Diagram

The FSM has 14 states for creating enable signals 8-bit registers so that the store the correct data as it is transferred. The done signal in the final state is used to activate the 16-bit registers that combine the byte sized sections into their final two-byte value.

III. EXPERIMENTAL SETUP

The experimental setup involved an Arduino board for a 5 V power supply, the Pixy Cam, an object to track, and the FPGA. Prior to testing, the Pixy Cam had to be calibrated to track an object. For this project, we calibrated it to detect a pink ball.

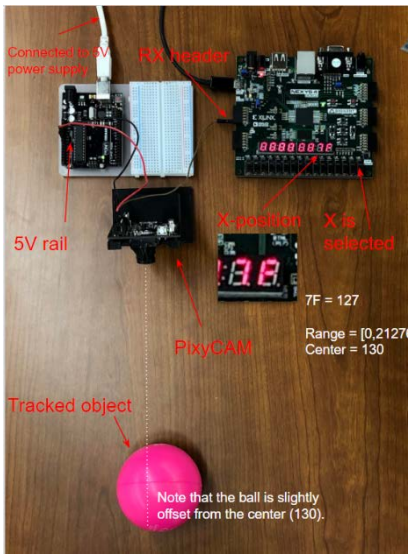

Figure 5. Experimental Setup

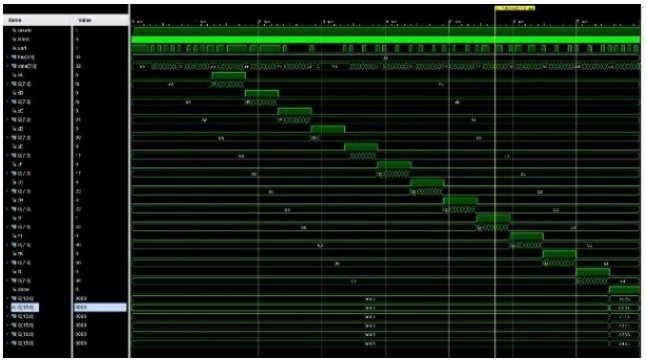

Figure 6. Testbench Simulation

A testbench was developed to verify if the digital blocks were working as intended. The signals that activate each register cascade, corresponding to the progression of states in the FSM. The signals correspond to which registers are being selected for storage. A done signal indicates when the individual data bytes should be combined into their final two-byte values.

IV. RESULTS

The interfacing between the FPGA and the Pixy Cam proved to be successful. The Pixy Cam actively sends data to the system and the FSM stores that data to registers, and once the Pixy Cam stops detaching the object, the memory will hold the previous value. The switches on the FPGA allow for the X, Y, width, or height values to be selected to be forwarded to the serializer and 7-segment display.

Figure 7. X value coming from the Pixy Cam.

The value of the X-coordinate in Figure 5 is 7F in hexadecimal. This corresponds to 127 in decimal notation. The maximum value X and Y can reach are the limits of the Pixy Cam's resolution (320x480). The width has a limit of 319 and the height limit is 199.

A video of the demonstration can be found at https://www.youtube.com/watch?v=lFH3Aim-BeY.

V. CONCLUSION

The project taught the group more applications for the UART communication protocol. We encountered many issues throughout the process, but we were able to solve them by researching more VHDL.

If we had more time to work on the project, we would implement distance detection between the Pixy Cam and the object being tracked. We could display the distance as a vector with a magnitude and direction on another 7-segment display.

REFERENCES

[1] CMUcam5 Pixy, "Pixy Serial Protocol," *cmucam.org*, July 5, 2014. [Online]. Available: http://www.cmucam.org/projects/cmucam5/wiki/Pixy_Serial_Protocol. [Accessed April 5, 2018]

[2] J. Plusquellic. "Hardware Design with VHDL Design Example: UART," Nov. 23, 2015. [Online]. Available: http://ece-research.unm.edu/jimp/vhdl_fpgas/slides/UART.pdf. [Accessed April 5, 2018].

[3] D. Llamocca, "Reconfigurable Computing Research Laboratory," April 2018. [Online]. Available: http://www.secs.oakland.edu/~llamocca/ [Accessed April 5, 2018].