ENSC 254 Lab 6

Summer 2018
Group 81
Steven Luu
Zhe Chen

Table of Contents

Objective **1**

Data Transfer Methods **1**

Table 1: Pros and Cons of Data Transfer Method 1

Conclusion **2**

Appendix **3**

References 3

Code 4

Objective

The objective of this lab is to research various data transfer method available with our FPGA board and to become familiar with using the UART with the board. The data transfer methods available are: UART, using the LED light intensity, displaying data on the OLED screen, using the audio jack, attaching a PMOD to the board, attaching an extension card to the board, over the network with the Ethernet jack, and using the VGA connector.

Data Transfer methods

Eight data transfer methods are available with our Zedboard and we will over the pros and cons of each method to determine which one we want to use. The pros and cons of each method are described in Table 1 below.

Data Transfer Method	Pros	Cons
USB UART: Dedicated device for text transfer	-well documented and widely used -has a parity bit for error checking -data packet structure can be changed as long as the UARTs are set up properly -no extra hardware required	-max size of data frame is limited to 9 bits -baud rate of both UART must be within 10% of each other -UART buffer can overflow
Light Intensity: Using a constructed vactrol and microcontroller to extract data from changes in LED light intensity	-LEDs are capable of transmitting data quickly	-must define light intensity levels -requires extra hardware
OLED: Display data on the OLED screen and typing data into computer by hand	-easy to interpret data on screen	-not automated, must manually input data -speed depends on user typing speed
Audio Jack: Transferring data over the audio jack into an audio recording and decoding it into text	-robust connection -compatible with many devices	-roundabout method, data transferred into audio recording then decoded
PMOD: Attaching a PMOD to the board and creating a custom protocol to extract data	-open standard -many available PMODs -multiple ways to transfer data	-custom protocol may be difficult to create -compatibility issues -additional costs -extra hardware required
Extension Card: Attaching an extension cord to the board and connecting it to the computer to extract data	-improved compatibility compared to PMODs -many kinds of extension cards -multiple ways to transfer data	-additional costs -extra hardware required

Ethernet Jack: Writing a firmware to connect the board over the network and transfer the data over HTTP	-fast data transfer	-requires writing firmware -requires web server -difficulty
VGA Connector: Writing a display driver and learns how the VGA standard work to create an image	-fast data transfer -allows visual output of data	-requires writing display driver -difficulty -requires VGA display

Table 1: Pros and Cons of Data Transfer Method

For our chosen method we would like to pick one that is fairly quick, not have a large cost associated with it, and relatively simple to implement. We see that the built-in jacks or connectors have fast data transfer rates, the additional attachments have varying data transfer rate depending on the module, and manually typing would be the slowest method. Several of these data transfer methods requires additional hardware, which can be fairly expensive depending on the module. If we were to ignore hardware costs then we would choose to attach an extension card as it would give us freedom in deciding how data is transferred. If we wanted the fastest data transfer method we would want to choose using the VGA connector as it has fast data transfer speeds and would not require a webserver. The easiest method would most likely be to display the data on the OLED screen and manually input the data however this method would be very slow compared to every other method. For our choice we choose would want to use UART since it is does not have any additional costs, it is easy to implement, and it is not the slowest data transfer method at 115200 baud.

Conclusion

From this lab we were able to gain experience in working with UART using TeraTerm and learned about the various data transfer rate available with our FPGA board. We researched the pros and cons of each method and broke them down into cost, complexity, and speed. While comparing these factors for the various data transfer methods we decided that the method we want to use is UART.

Appendix

References

[1] "Universal asynchronous receiver-transmitter", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter. [Accessed: 06- Jul- 2018].

[2] C. Basics, "Basics of UART Communication", *Circuit Basics*, 2018. [Online]. Available: http://www.circuitbasics.com/basics-uart-communication/. [Accessed: 06- Jul- 2018].

[3] “Pmod Interface,” *Wikipedia*, 03-Jul-2018. [Online]. Available: https://en.wikipedia.org/wiki/Pmod_Interface. [Accessed: 06-Jul-2018].

[4] “Phone connector (audio),” *Wikipedia*, 03-Jul-2018. [Online]. Available: https://en.wikipedia.org/wiki/Phone_connector_(audio). [Accessed: 06-Jul-2018].

[5] “Video Graphics Array,” *Wikipedia*, 03-Jul-2018. [Online]. Available: https://en.wikipedia.org/wiki/Video_Graphics_Array. [Accessed: 06-Jul-2018].

[6] “Ethernet,” *Wikipedia*, 03-Jul-2018. [Online]. Available: https://en.wikipedia.org/wiki/Ethernet. [Accessed: 06-Jul-2018].

[7] “Accessories,” *ZedBoard | Zedboard*. [Online]. Available: http://www.zedboard.org/accessories. [Accessed: 06-Jul-2018].

[8] “ZedBoard,” *ZedBoard | Zedboard*. [Online]. Available: http://www.zedboard.org/content/zedboard-0. [Accessed: 06-Jul-2018].

Code

```
// ENSC Lab 6
// Steven Luu
// Zhe Chen
// Takes roughly 2 mins 13 secs to transfer "Hello, World" 100,000 times

.global asm_main

asm_main:
 LDR R0, =0x41210000  @* LED
 LDR R1, =0x41200000  @* Buttons
 LDR R2, =0x41220000  @* Switches

 LDR R3, =0xE0001004     @* Transmitter
 LDR R4, =0xE0001018     @* baud rate generator
 LDR R5, =0xE0001034     @* baud rate divider
 LDR R6, =0xE0001000     @* control register

 LDR R7, =0x20
 STR R7, [R3, #0]        @* Setting up transmitter
 LDR R7, =#62
 STR R7, [R4, #0]        @* Setting up baud rate to 62
 LDR R7, =#6
 STR R7, [R5, #0]        @* Setting up BDIV value to 6
 LDR R7, =0x117
 STR R7, [R6, #0]        @* Setting up control register

Begin:
 BL Button_Debounce
 LDR R8, [R1, #0]        @* Load R8 with Button Press
 BL Button_Debounce
 CMP R8, #0
 BLNE Many_Print
 B Begin

Many_Print:
 Push {R6, R14}
 LDR R6, =#100000
LoopMP:
 BL Delay
 BL Print
 CMP R6, #0
 SUB R6, R6, #1
 BNE LoopMP
 Pop {R6, R14}
 MOV R15, R14

Print:                          @* Prints "Hello, World"
 Push {R14}
 LDR R9, =0xE0001030
 LDR R10, =#72                  @* ASCII Value of H = 72 in decimal
 STR R10, [R9]                  @* Load 72 to =0xE0001030
```

```
 LDR R10, =#101      @*e
 STR R10, [R9]
 LDR R10, =#108      @*l
 STR R10, [R9]
 LDR R10, =#108      @*l
 STR R10, [R9]
 LDR R10, =#111      @*o
 STR R10, [R9]
 LDR R10, =#44       @*,
 STR R10, [R9]
 LDR R10, =#32       @*_
 STR R10, [R9]
 LDR R10, =#87       @*W
 STR R10, [R9]
 LDR R10, =#111       @*o
 STR R10, [R9]
 LDR R10, =#114       @*r
 STR R10, [R9]
 LDR R10, =#108       @*l
 STR R10, [R9]
 LDR R10, =#100       @*d
 STR R10, [R9]
 LDR R10, =#32       @_
 STR R10, [R9]
 LDR R10, =#10       @*Newline
 STR R10, [R9]
 LDR R10, =#13       @*Carriage Return
 STR R10, [R9]
 Pop {R14}
 MOV R15, R14

Button_Debounce:                 @* Software Delay to Improve Button Bounce
 Push {R6}
 LDR R6, =#2000000
LoopBD:
 CMP R6, #0
 SUB R6, R6, #1
 BNE LoopBD
 Pop {R6}
 MOV R15, R14

Delay:                           @* Software Delay for the UART Buffer
 Push {R7, R14}
 LDR R7, =#33000
LoopDL:
 CMP R7, #0
 SUB R7, R7, #1
 BNE LoopDL
 Pop {R7, R14}
 MOV R15, R14
```