

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

3/11/2022

Lab 5

UART Transmitter

Several thin, curved lines in dark blue and light grey originate from the bottom left and curve upwards and to the right.

Spence Johnston

CST 231 – DIGITAL SYSTEM DESIGN I

Spence Johnston

Table of Contents

Abstract.....	2
Introduction	3
Design.....	3
Physical hardware design.....	3
Synthesized hardware design	3
Design Structure.....	4
Modules	4
Simulation and Testing.....	5
Problems	6
Results and Conclusion	6
Appendix – Appendix Label goes Here (i.e. A)	7
Table of Figures	8
References	9

Abstract

The purpose of this lab is to create a UART transmitter. The project should output character to a computer. This is done with a USB to UART adapter. The objective of this goal is to output character via the UART protocol to a computer.

The UART communication should adhere to the following specifications: 9600 baud, eight-bit data, one start bit, one stop bit, and even parity bit.

The first thing needed is a clock divider. This is used to reduce the clock to 9600 Hz. Next, an input handler is needed to detect a button press. This controller sends an enable signal to a string controller, which sends characters one at a time. This output is fed to a transmitter module. This module handles the formatting and sending of the UART package. This module also has a ready signal, as feedback, to the string controller.

The biggest issue I had with this lab was timing. I was having issues connecting my modules together. This is because one required at least two cycles until the next enable could be processed. This stipulation messed with my transmitter module.

Overall, I found this lab very educational. Learning how to communicate via serial protocols seems very beneficial to know for the future.

Introduction

The purpose of this lab is to create a UART transmitter. The project should output character to a computer. This is done with a USB to UART adapter. The objective of this goal is to output character via the UART protocol to a computer.

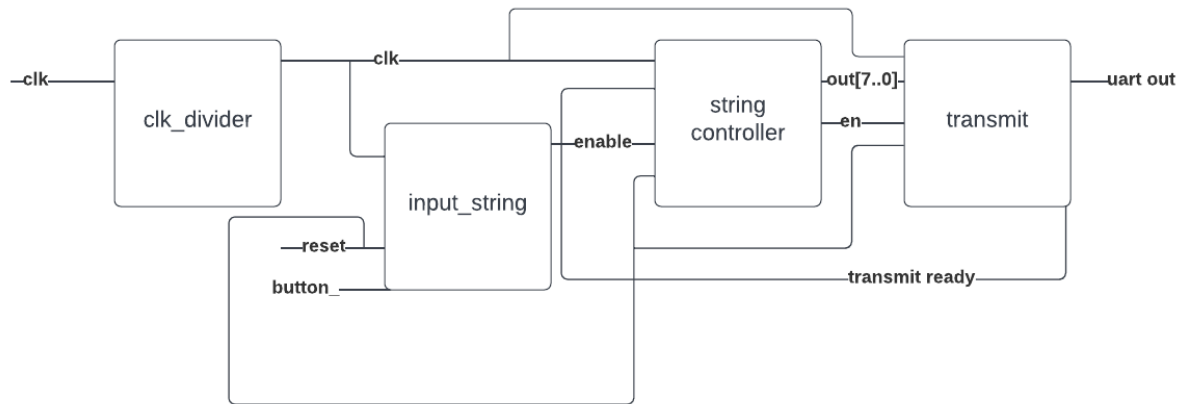


Figure 1: Block diagram

Design

The UART communication should adhere to the following specifications: 9600 baud, eight-bit data, one start bit, one stop bit, and even parity bit.

Physical hardware design

The development board used is the Cyclone 5. To interface with the computer, a USB to UART adapter was used. The GPIO was used to connect to this adapter via Rx. A common ground was necessary for reference voltages. No additional hardware was necessary.

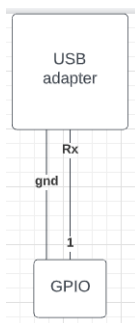


Figure 2: wiring schematic

Synthesized hardware design

The first thing needed is a clock divider. This is used to reduce the clock to 9600 Hz. Next, an input handler is needed to detect a button press. This controller sends an enable signal to a string controller, which sends characters one at a time. This output is fed to a transmitter module. This module handles

the formatting and sending of the UART package. This module also has a ready signal, as feedback, to the string controller.

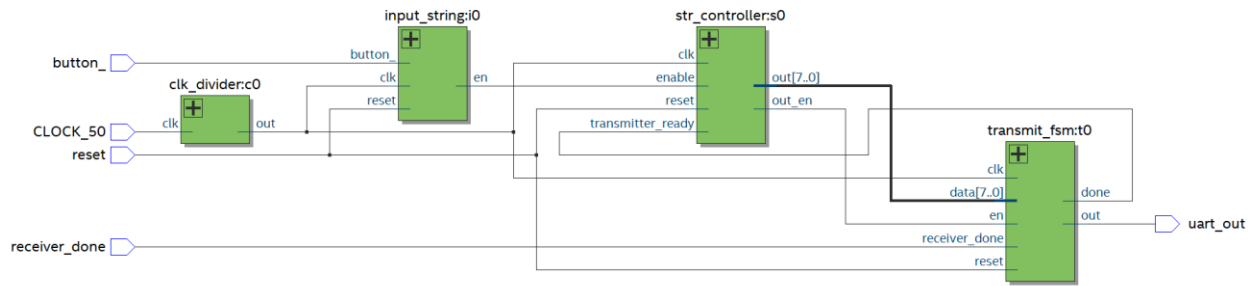


Figure 3: Top level RTL

Design Structure

Modules

`input_string`

This module accepts a button, clock, and reset signal. If reset is high, the state will go to idle. If the button is pressed, an enable signal is sent to the string controller. This module locks out the button until unpressed. As shown below, this module is a state machine.

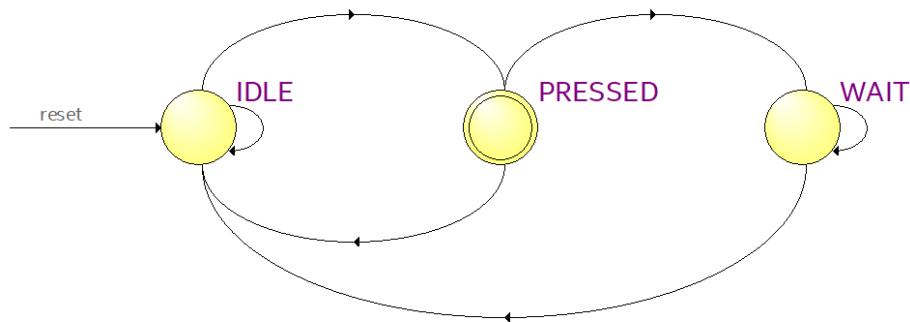


Figure 4: `input_string` state diagram

`str_controller`

This module outputs a string one character at a time. It holds when the transmitter module is not ready. A buffer state is needed to interact with other modules properly.

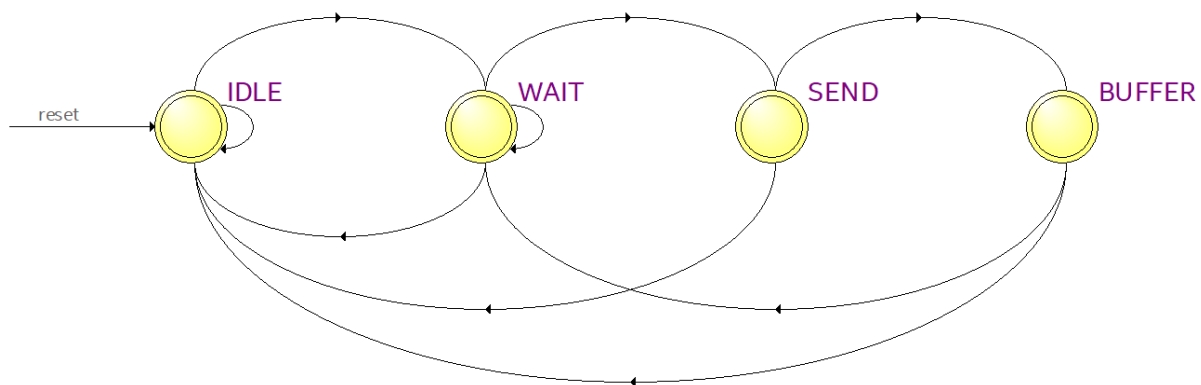


Figure 5: str_controller state diagram

transmit_fsm

This module handles the UART communication. This accepts an enable and a character. It wraps the data in the UART specifications, and outputs it serially.

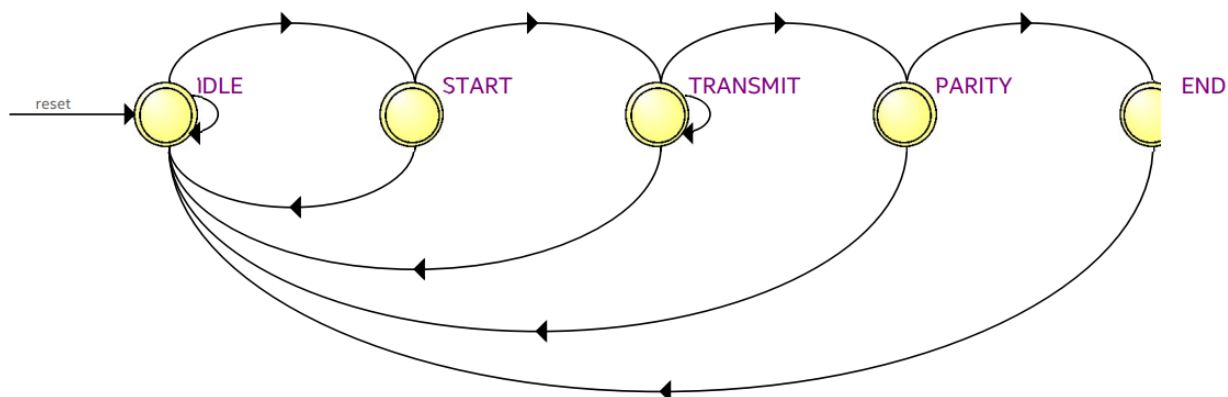


Figure 6: transmit_fsm state diagram

Simulation and Testing

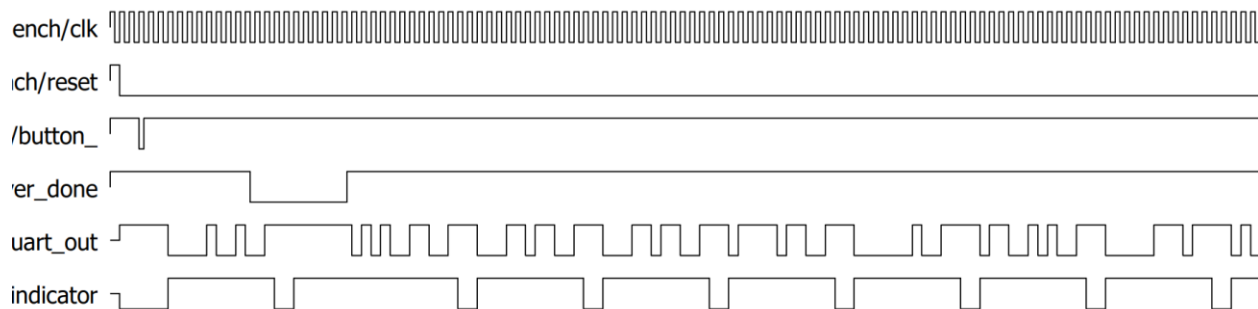


Figure 7: Top level simulation

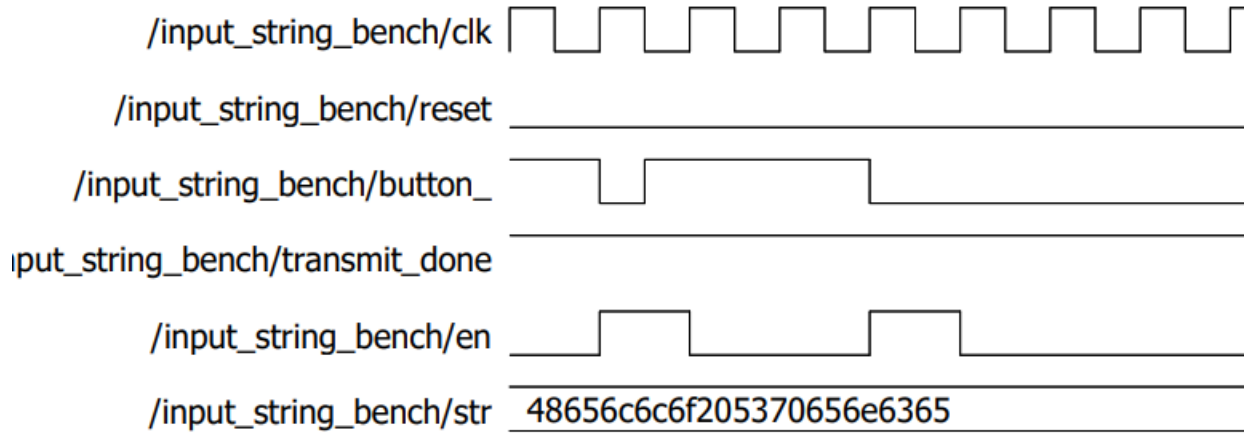


Figure 8: `input_string`

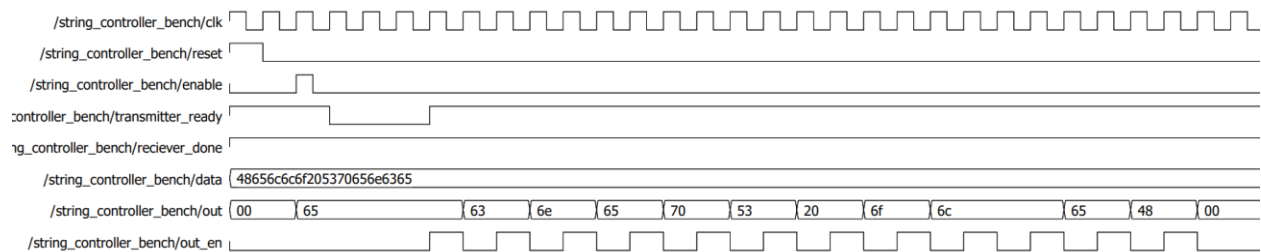


Figure 9: `str_controller`

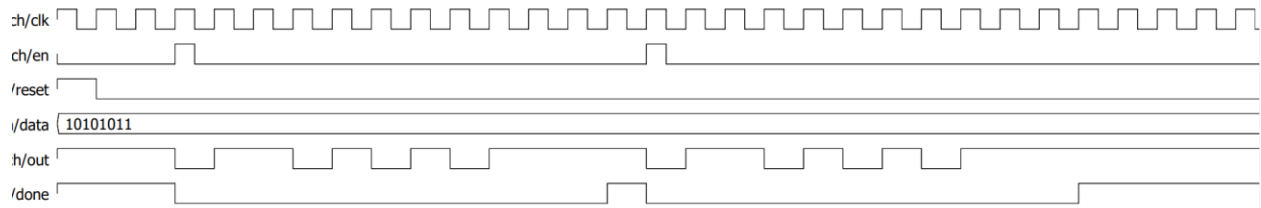


Figure 10: `transmit_fsm`

Problems

The biggest issue I had with this lab was timing. I was having issues connecting my modules together. This is because one required at least two cycles until the next enable could be processed. This stipulation messed with my transmitter module.

Results and Conclusion

Overall, I found this lab very educational. Learning how to communicate via serial protocols seems very beneficial to know for the future.

Appendix







Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate
 CLOCK_50	Input	PIN_AF14	3B	B3B_N0	3.3-V LVTTL		16mA (default)	
 button_	Input	PIN_Y16	3B	B3B_N0	3.3-V LVTTL		16mA (default)	
 receiver_done	Input	PIN_Y18	4A	B4A_N0	3.3-V LVTTL		16mA (default)	
 reset	Input	PIN_AB12	3A	B3A_N0	3.3-V LVTTL		16mA (default)	
 uart_out	Output	PIN_Y17	4A	B4A_N0	3.3-V LVTTL		16mA (default)	1 (default)
 led_sensor	Unknown	PIN_Y21	5A	B5A_N0	3.3-V LVTTL		16mA (default)	

Figure 11: pinout

Table of Figures

Figure 1: Block diagram.....	3
Figure 2: wiring schematic	3
Figure 3:Top level RTL	4
Figure 4: input_string state diagram.....	4
Figure 5: str_controller state diagram	5
Figure 6: transmit_fsm state diagram.....	5
Figure 7: Top level simulation	5
Figure 8: input_string.....	6
Figure 9: str_controller	6
Figure 10: transmit_fsm.....	6
Figure 11: pinout.....	7

References

- E. Pena, "UART: A hardware communication protocol understanding universal asynchronous receiver/transmitter," *UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter / Analog Devices*. [Online]. Available: <https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>. [Accessed: 11-Mar-2022].