



Table of Contents

[Abstract 2](#_30j0zll)

[Introduction 3](#_3znysh7)

[Design 3](#_2et92p0)

[Physical hardware design 3](#_tyjcwt)

[Synthesized hardware design 3](#_1t3h5sf)

[Design Structure 4](#_4d34og8)

[Modules 5](#_2s8eyo1)

[Clock Divider–clk\_dv.v 5](#)

[Button Lockout–button\_lockout.v 5](#)

[Mux UART Control–mux.v 6](#_wddy84crczvr)

[UART transmitter–UART\_tx.v 6](#_g1r18omc74oa)

[Top Level–Lab5.v 9](#_pea0xwh8q4os)

[Simulation and Testing 9](#_17dp8vu)

[Problems 10](#_3rdcrjn)

[Results and Conclusion 10](#_26in1rg)

[Appendix A – RTL Diagrams 11](#_lnxbz9)

[Appendix B – Waveforms 13](#_o1liip29khcm)

[Appendix C – State Machine Diagrams 14](#_k4qyftxwhzoe)

[Appendix D – Pin Tables 15](#_z9rzchshg9wc)

[Table of Figures 17](#_35nkun2)

[References 18](#_1ksv4uv)

# Abstract

This lab creates the transmit side of a UART. This transmitter adheres to the following specifications:

1. 9600 baud

2. Eight data bits

3. One start bit

4. One stop bit

5. Even parity bit

First, the data to be transmitted is hooked to the switches on the DE1-SoC board. When push button one is pressed, the system will transmit a single character on the UART. This is received using a USB serial dongle plugged into the computer. This requires creating a lockout for the button, so it only transmits a single character per button press. Secondly, a different button press sends a message to the UART serial dongle–“Hello Cari!”.

# Introduction

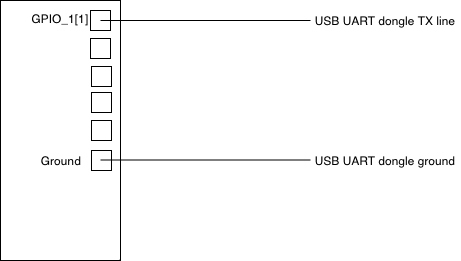
The purpose of this lab is to build the transmitting end of a UART. The goal is to effectively transmit data from two different sources in a serial UART fashion.

# Design

Here, the physical hardware and synthesized hardware designs are discussed.

## Physical hardware design

The UART USB dongle is used for reception of the serial UART data from the DE1-SoC board through a GPIO pin. A ground is connected to sync the dongle’s ground with the DE1-SoC’s as well.



*Figure 1: Hardware Connection Diagram*

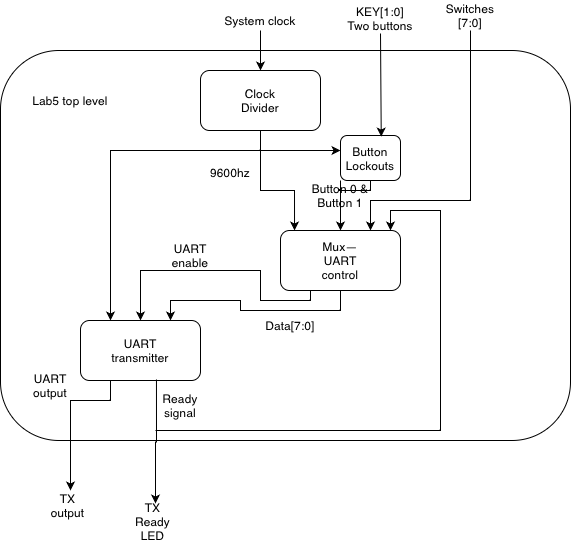
The “ready” output from the top level module is assigned to an LED on the board. KEY[0] and KEY[1] are connected to the [1:0]cntrl input, and used to transmit the data from the switches or signal the UART transmitter to transmit the pre-specified message, respectively. Switches 0-7 are connected to the SW[7:0] inputs of the top level, and SW[9] is assigned to the reset. The system 50 MHz clock is assigned as “clk.”

## Synthesized hardware design

In this section the major functional modules of the Design hierarchy are discussed.

### Design Structure

Here is the block diagram for the completed project:



*Figure 2: Hierarchical Block Diagram*

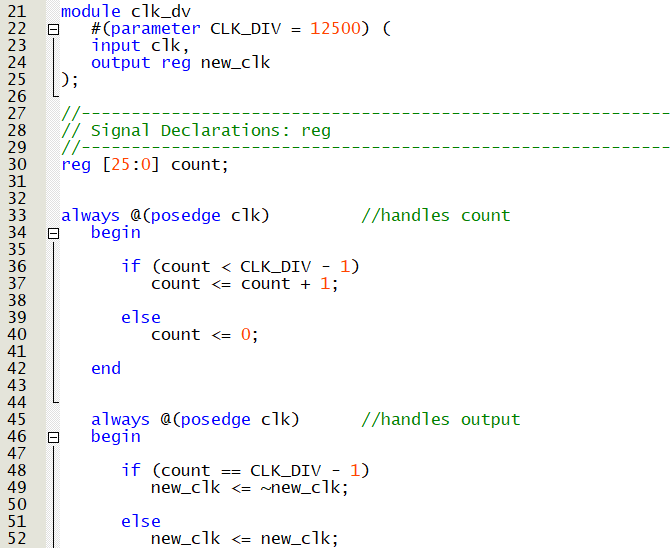
A clock divider takes the system clock and generates a 9600 Hz clock used by all other low level modules. Two button lockout modules each take the signal from one button and output a one clock cycle pulse for each button press. The mux module takes both of those outputs, the divided clock, the input from the switches on the DE1-SoC board, and a ready signal (output by the UART transmitter module). The mux module outputs an enable signal and eight bits of data, which is received by the UART transmitter module. The transmitter module outputs the serial UART output, and a ready signal, which is both sent back to the mux and to an LED on the board.

### Modules

Each module is further described below.

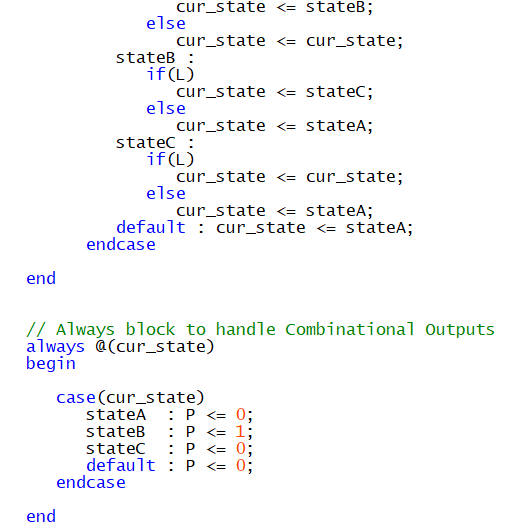
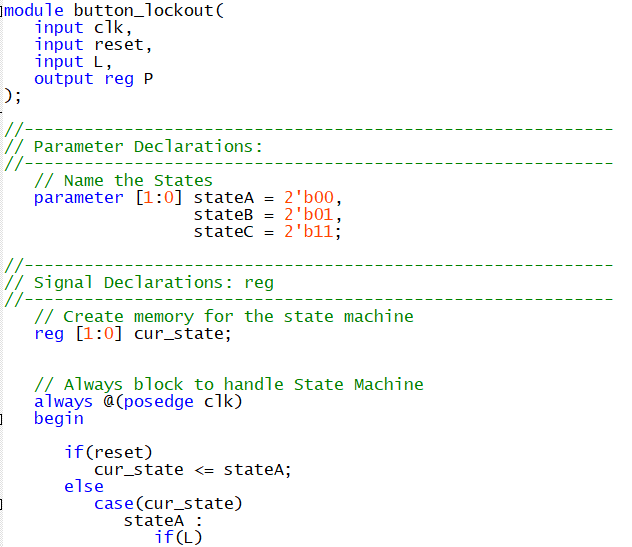
#### Clock Divider–clk\_dv.v

The clock divider takes the system clock as input and modifies its clock output based on a given parameter to produce a new clock of the desired frequency. The RTL for this module can be found in Appendix A. The module code follows.



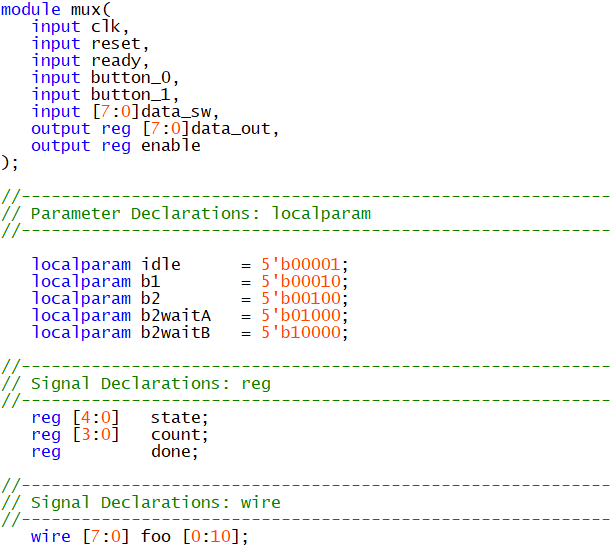
#### Button Lockout–button\_lockout.v

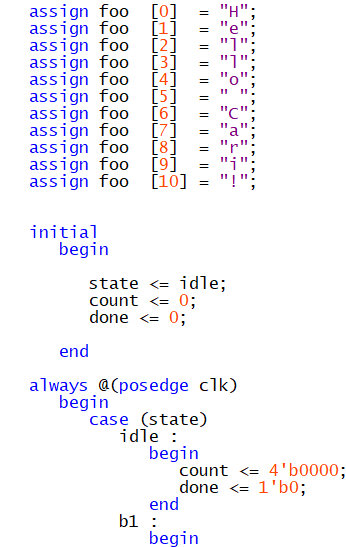
The button lockout module takes in a clock, a reset, and an “L” signal. This module is a state machine which kicks off when “L” is high. The state machine defaults to state A. When “L” is high, it transitions to state B on the next positive clock edge. If “L” is high in state B, it transitions to state C, otherwise back to state A. If “L” is high in state C, the machine remains in state C until it goes low, where it transitions back to state A again to wait for the next input change. The module’s sole output register, “P,” is only high in state B, and is only ever in state B for one clock cycle, transforming a level signal to a pulse. The module code is below. The state diagram can be found in Appendix C, and the RTL diagram can be found in Appendix A.

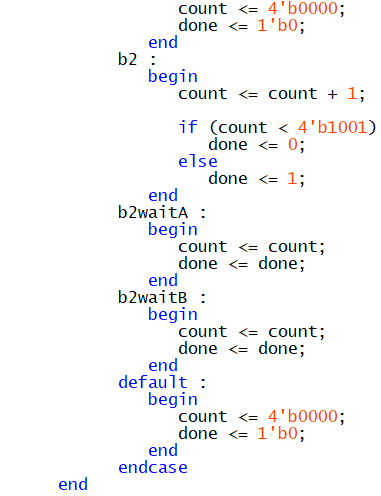


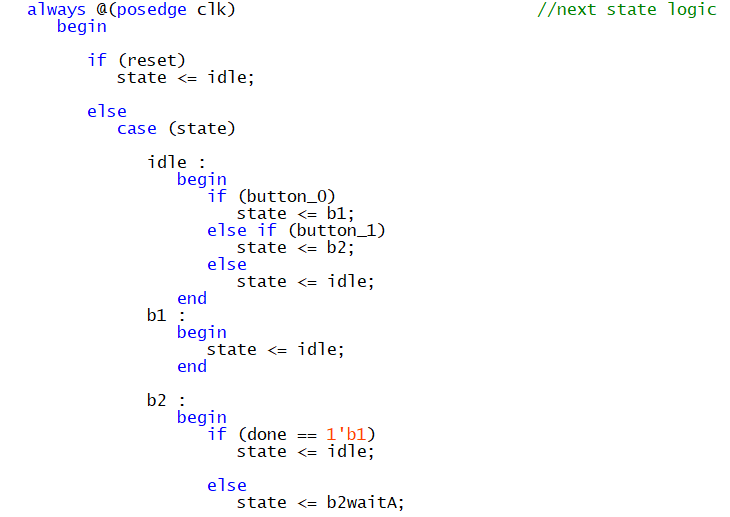
#### Mux UART Control–mux.v

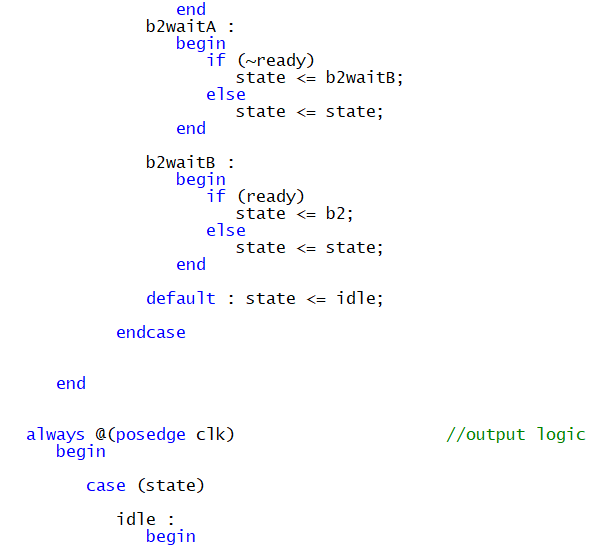
The Mux UART Control module’s inputs are a clock, reset signal, ready signal, button 0, button 1, and an 8 bit data source (to come from switches). The outputs are an eight bit data out register, and an enable signal. The mux contains internal registers to hold the ASCII characters for the message “Hello Cari!” The module is a state machine with five states. The default state is idle. From idle, if button 0’s input goes high, then the state machine transitions to the b1 state. In the b1 state, the output register is loaded with the data from the input switches, and the enable signal is made high. The state machine stays in the b1 state for one clock pulse, and then transitions back to idle. From idle, if button 1’s signal goes high, the state machine transitions to state b2. In state b2, an internal counting register begins to count, to be used as the index for the array of characters of the predetermined message. In state b2, the current counting index is used to load the output register with the data at that index of the array of characters. The state machine transitions to b2waitA at the next clock pulse, where the “ready” signal must drop before transitioning to b2waitB, where the “ready” signal must rise again to transition back to state b2, to continue loading the output register with the characters of the array, until the final count is reached, where the state machine transitions to idle again. The module’s code is found below. The state machine diagram can be found in Appendix C, and the RTL diagram can be found in Appendix A.

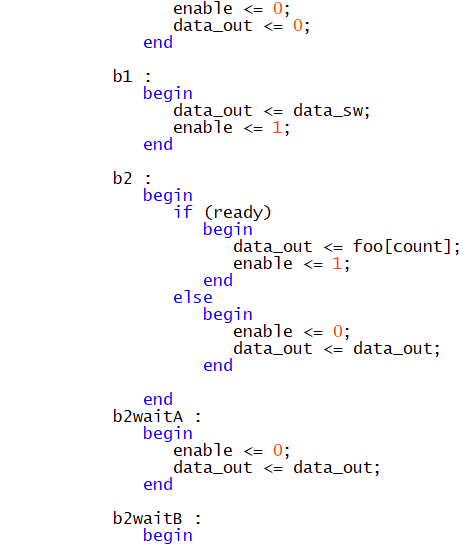


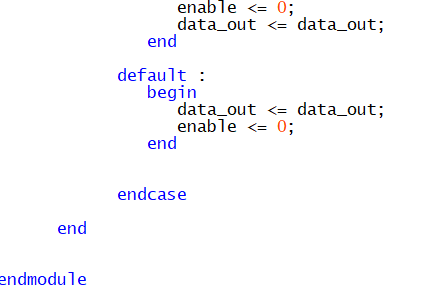






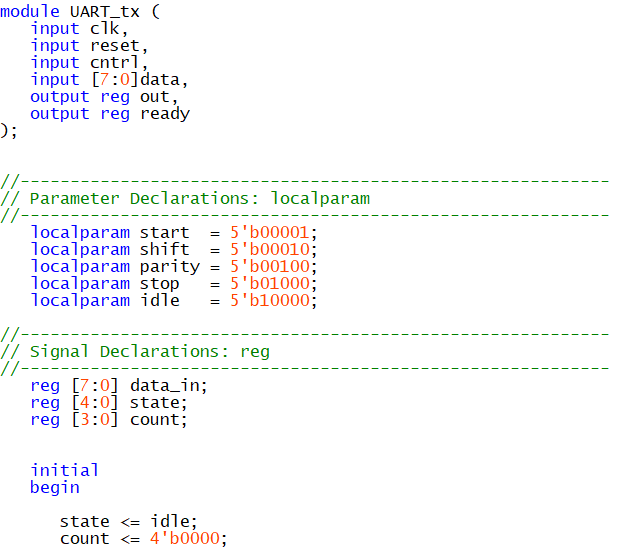


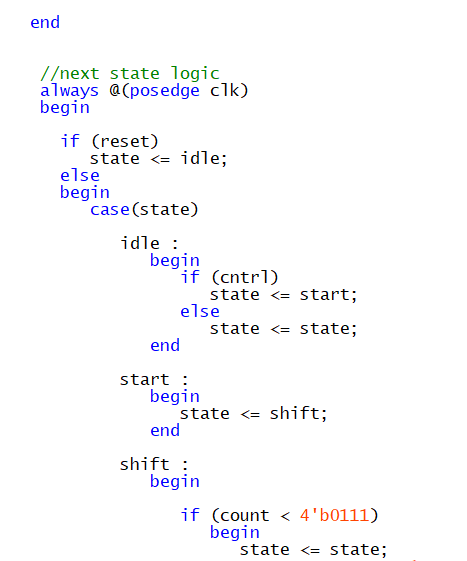


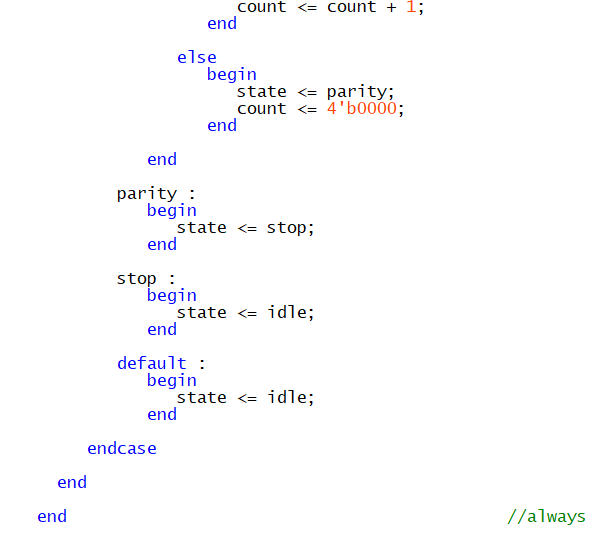


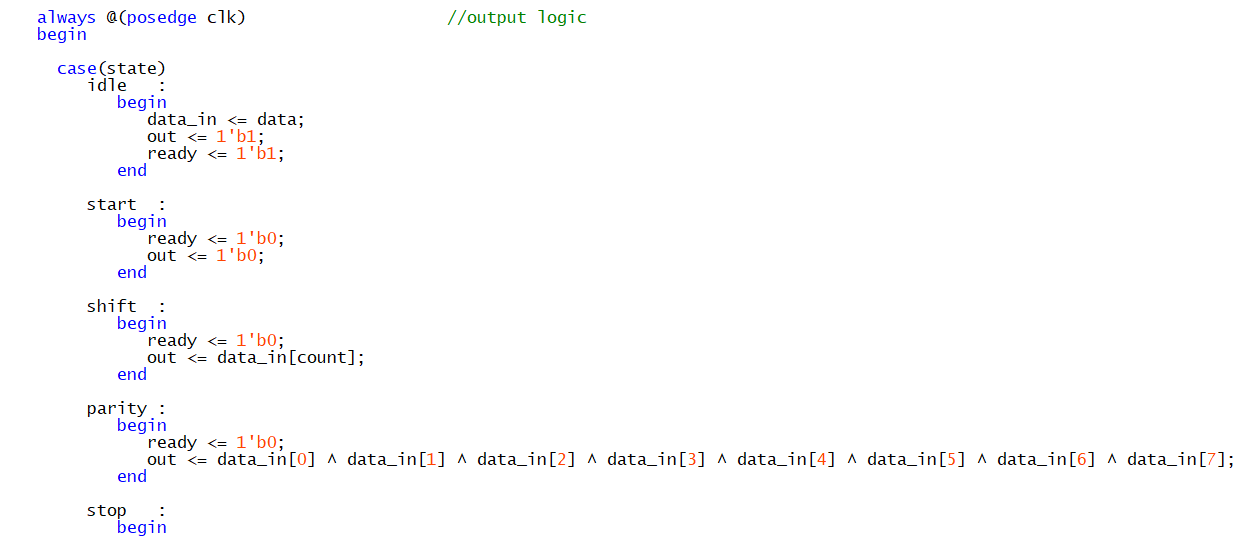
#### UART transmitter–UART\_tx.v

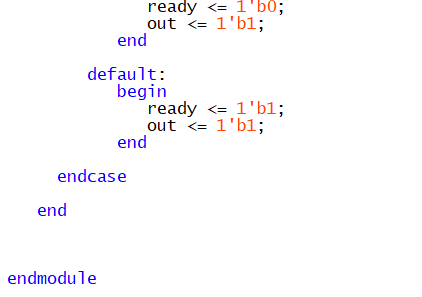
The UART transmitter takes in an 8 bit register for transmission data, a clock, a control signal, and a reset. When control is high, a state machine (diagram in Appendix C) kicks off to output the serial UART output of the data. The ready LED output signal is high when the state machine is in idle, and each bit of UART is output during each transition of the state machine. During the data stage, the register is checked for lowercase letters and converted to uppercase if necessary. The module’s code is given below. Appendix A contains the RTL for this diagram, Appendix B contains the waveforms from simulation, and Appendix C contains the state machine diagram.





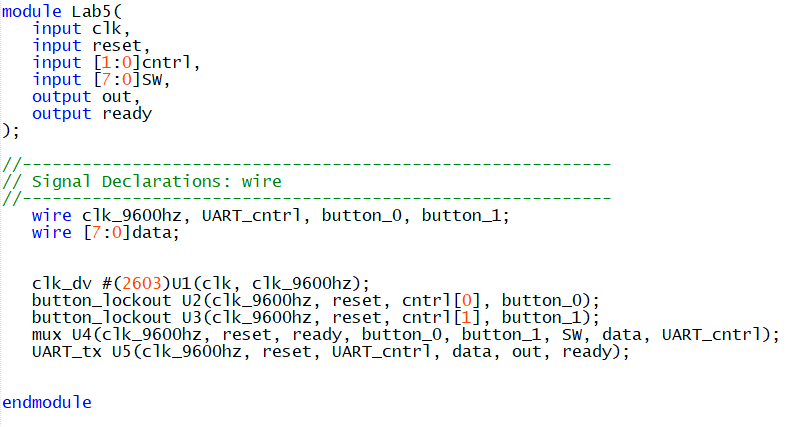






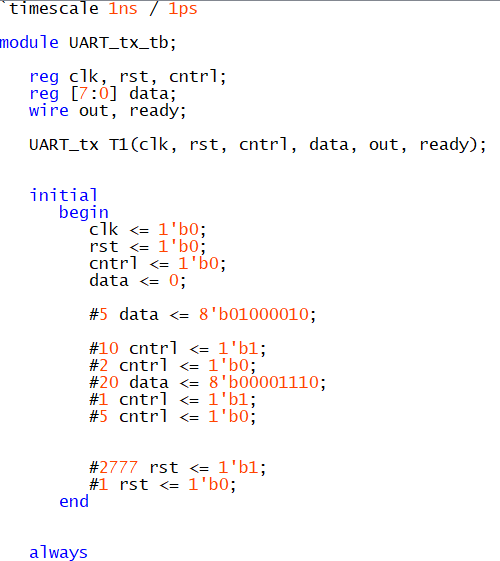
Top Level–Lab5.v

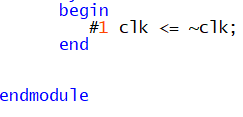
The top level module for this lab takes in the system clock, a reset, two bits of control, and 8 bits for the switches. Its outputs are the serial “out” output and a ready signal. This module instantiates the other modules in the lab, hooking them all up together as shown in the hierarchical block diagram. The module’s code is given below. The module’s RTL diagram can be found in Appendix A.



# Simulation and Testing

The testbench for the UART transmitter is included below. This emulates the transmission of a “B” and then “00001110.” It then resets to ensure a working reset. The waveform outputs of the testbench are included in Appendix B.





# Problems

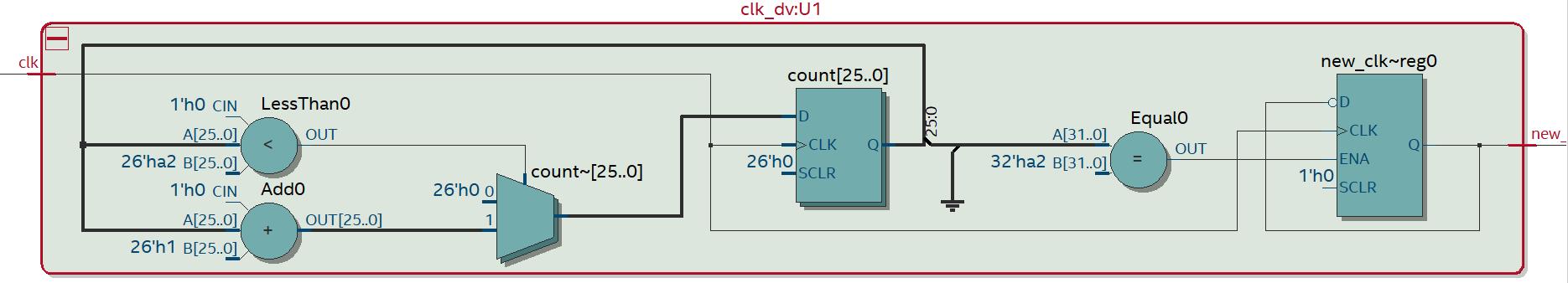
The hardest part of this lab was tying in both switch input and predetermined ASCII character inputs to be sent by different buttons of the board. At one point, the wrong buttons were being pressed for testing. Definitely will check in the future to be absolutely sure the correct buttons are used so as to avoid “fixing” code that is not broken. Another problem during the design process was that the predetermined message was transmitting–but characters were missing. Professor Scevers recommended adding extra states to the “mux” module to wait for the transmitter to be ready before transitioning to the next character, and that fixed the issue.

# 

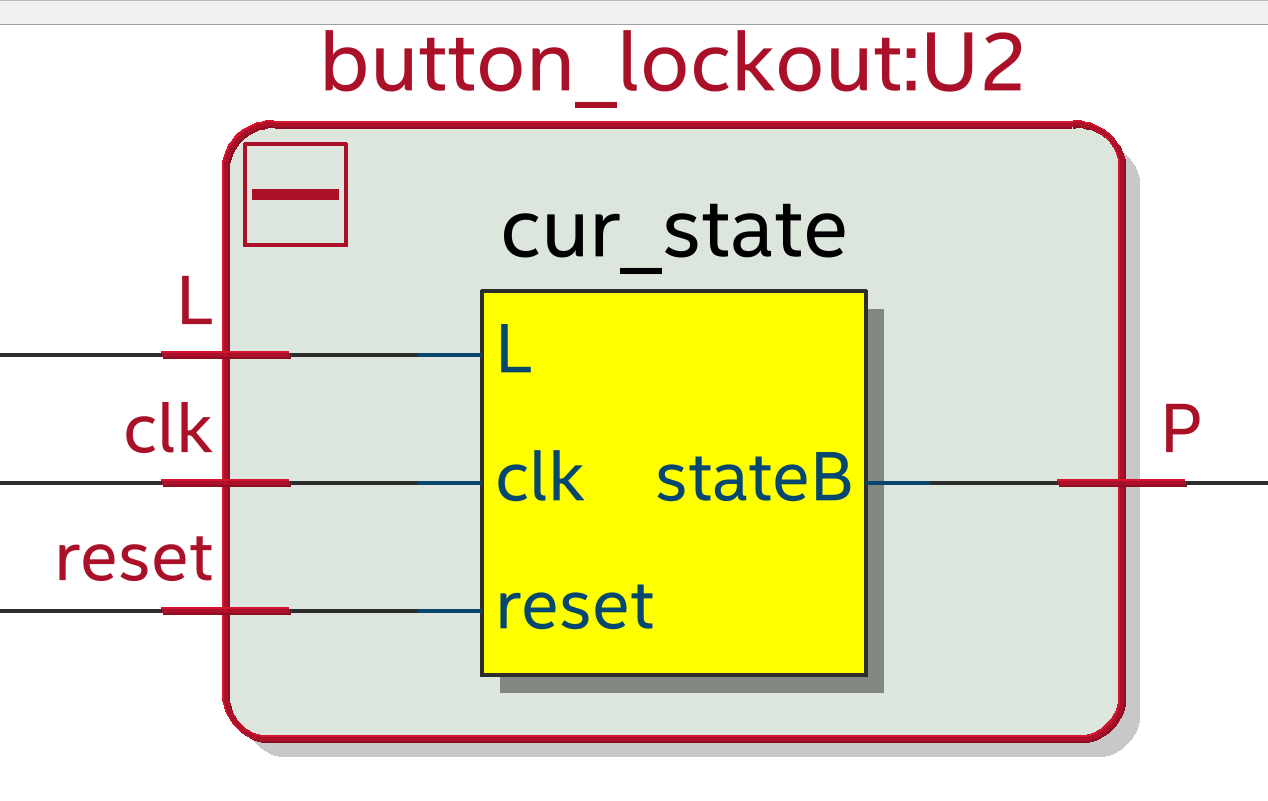
# Results and Conclusion

The result of this lab is a fully operational UART transmitter. It transmits according to UART standards, and can either receive data from an external source or from an internally determined source. Next time, being more careful with hardware to be sure it is being used as established in code will provide benefit. Careful consideration of timing when connecting modules together that are dependent on each other will also be a boon.

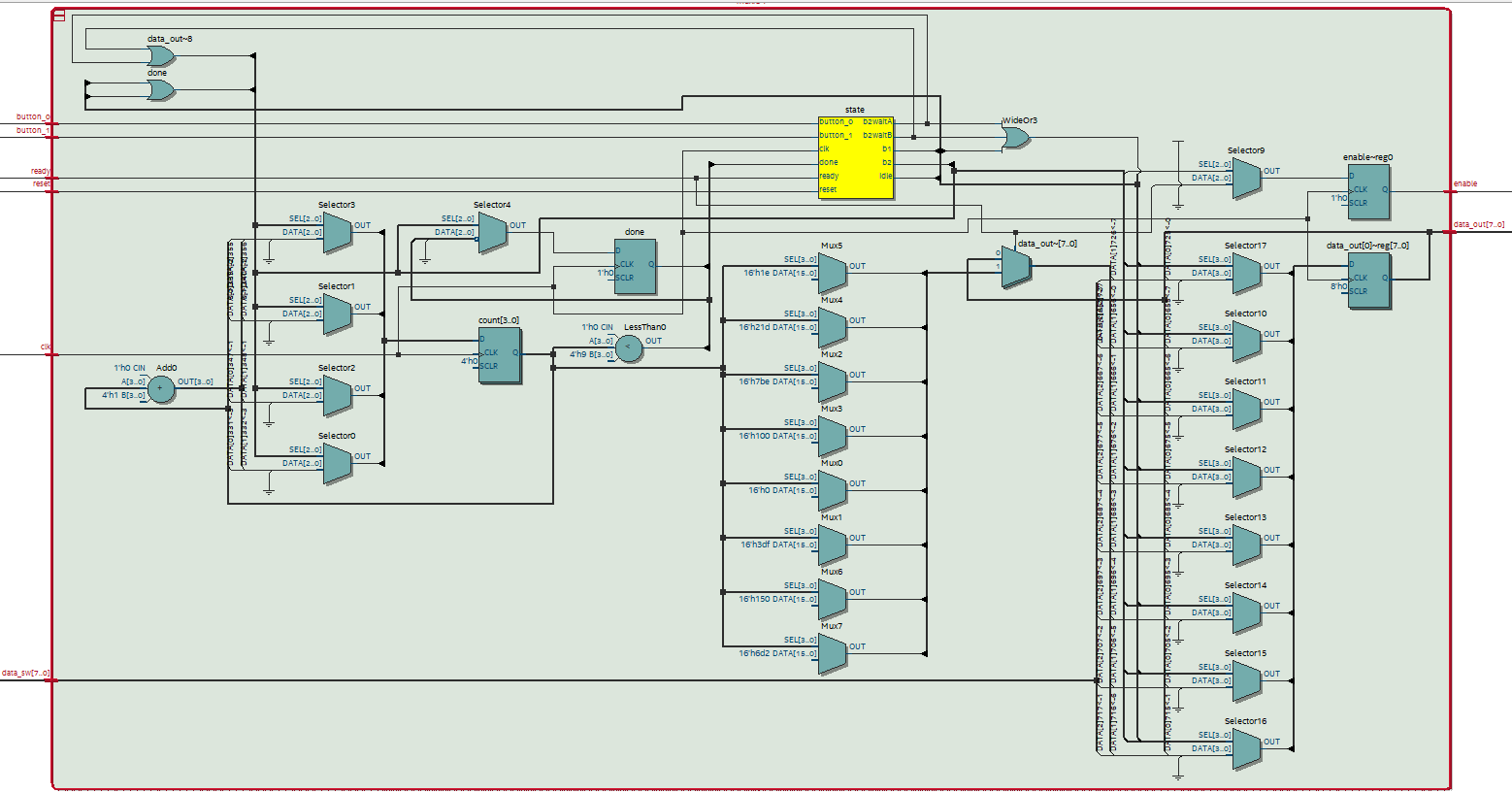
# Appendix A – RTL Diagrams



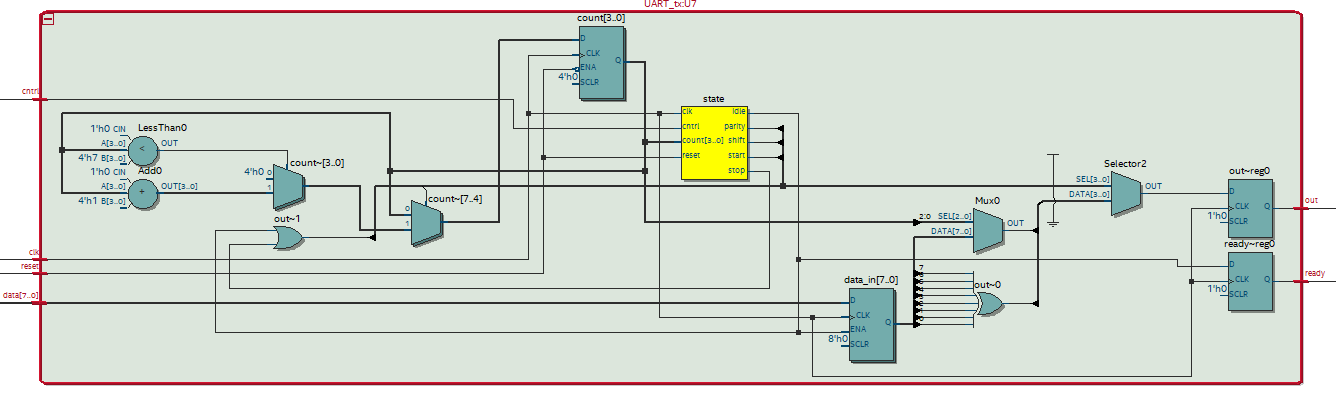
*Figure 3: Clock Divider RTL Diagram*



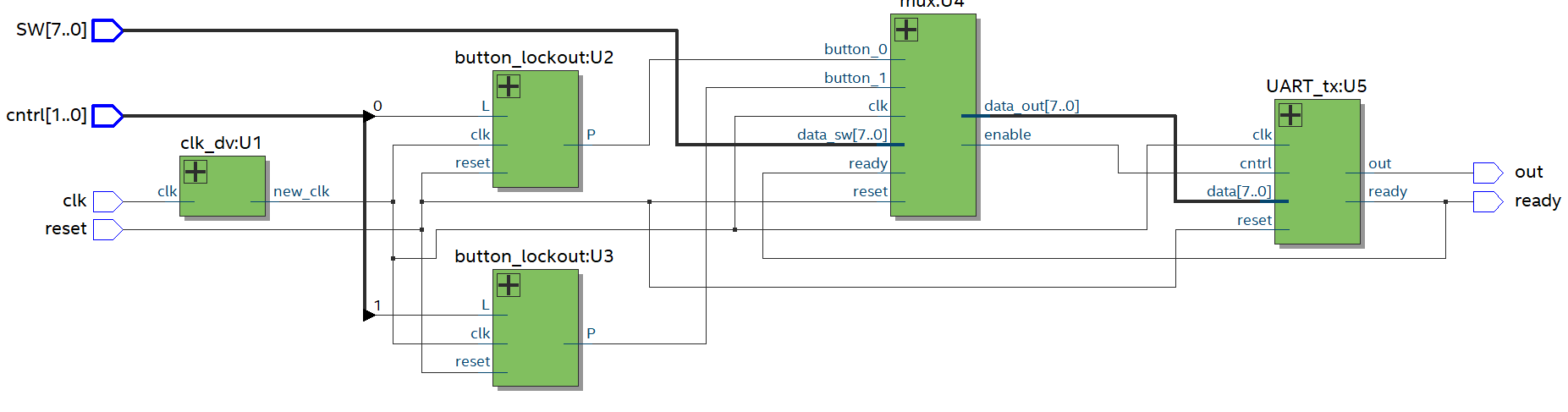
*Figure 4: Button Lockout RTL Diagram*

**

*Figure 5: Mux UART Control RTL Diagram*

**

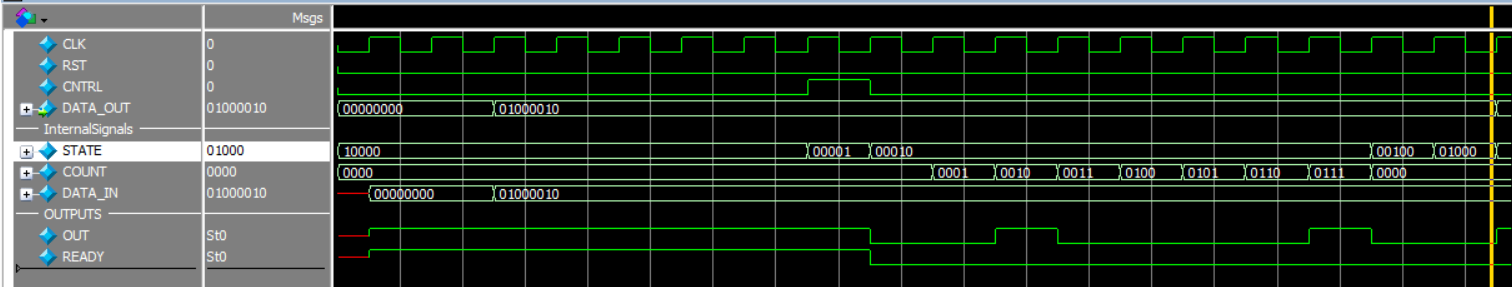
*Figure 6: UART Transmitter RTL Diagram*

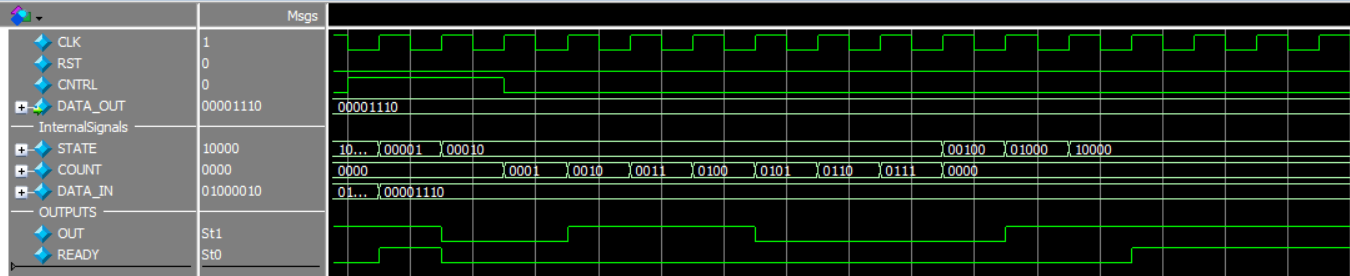


*Figure 7: Top Level RTL Diagram*

# 

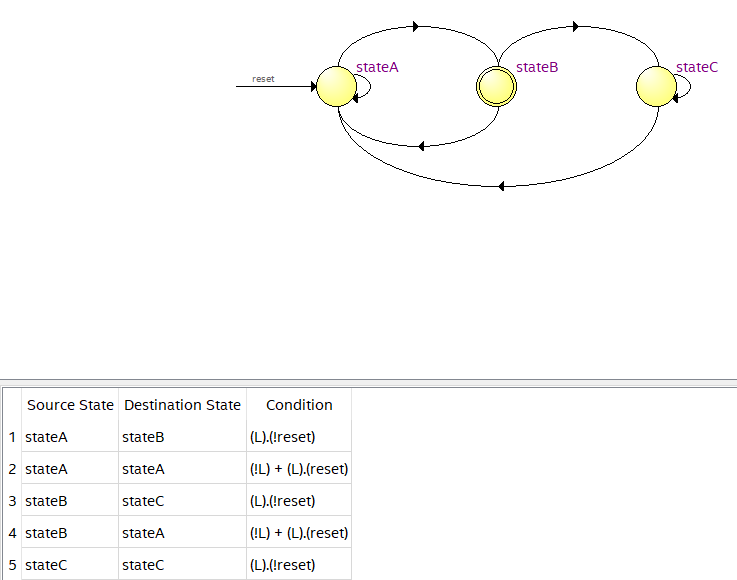
# Appendix B – Waveforms

*Figure 8: UART tx testbench waveforms part one*

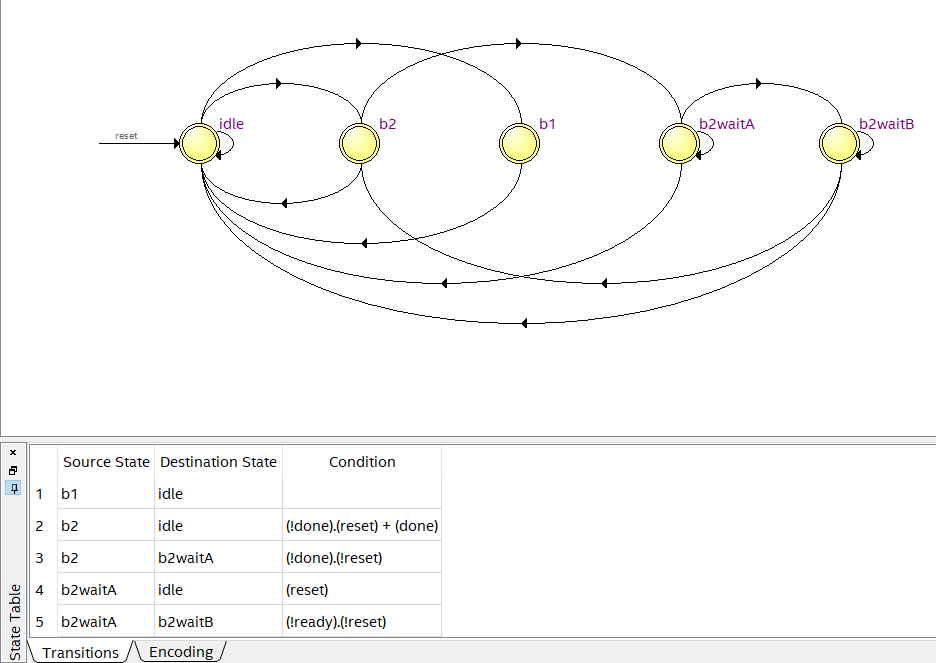
**

*Figure 9: UART tx testbench waveforms part two*

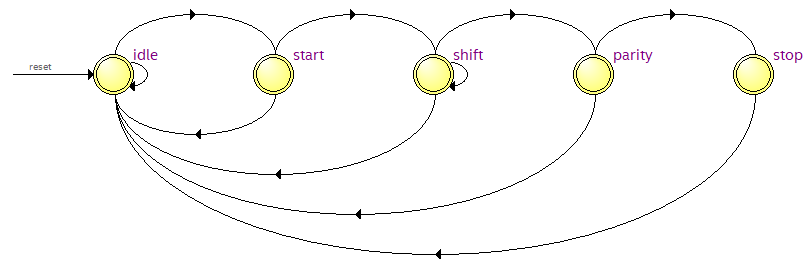
# Appendix C – State Machine Diagrams



*Figure 10: Button Lockout State Diagram*

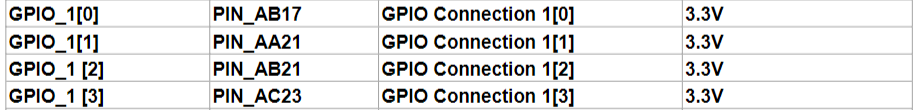
**

*Figure 11: Mux UART Controller State Diagram*

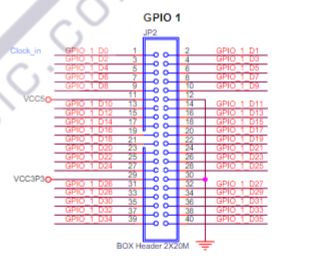


*Figure 12: UART tx State Diagram*

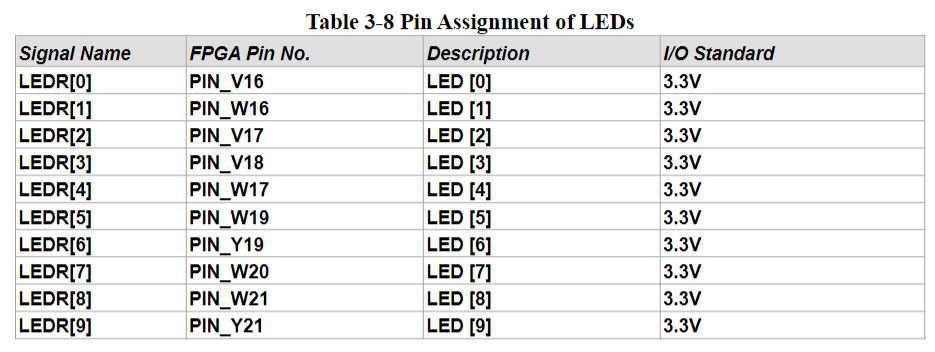
# Appendix D – Pin Tables



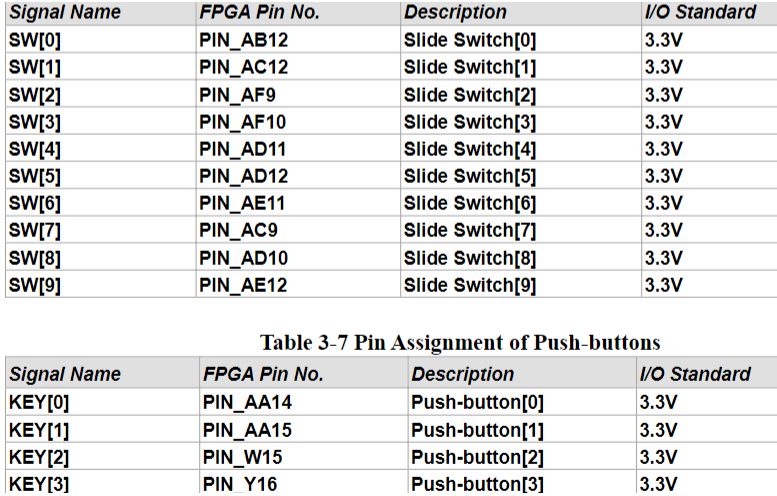
*Figure 13: Relevant GPIO\_1 Pin Assignment table from DE1-SoC User Manual*



*Figure 14: Pinout Diagram from DE1-SoC User Manual*



*Figure 15: LED Pin Assignment table from DE1-SoC User Manual*

**

*Figure 16: Switch and Button Pin Assignment table from DE1-SoC User Manual*

# Table of Figures

[Figure 1: Hardware Connection Diagram](#_3dy6vkm) 3

Figure 2: Hierarchical Block Diagram 4

[Figure 3:](#_3dy6vkm) Clock Divider RTL Diagram16

Figure 4: Button Lockout RTL Diagram 16

[Figure](#_3dy6vkm) 5: Mux UART Control RTL Diagram16

Figure 6: UART Transmitter RTL Diagram 17

[Figure 7:](#_3dy6vkm) Top Level RTL Diagram 17

Figure 8: UART tx testbench waveforms part one 18

[Figure 9:](#_3dy6vkm) UART tx testbench waveforms part two 18

Figure 10: Button Lockout State Diagram 19

[Figure](#_3dy6vkm) 11: Mux UART Controller State Diagram 19

Figure 12: UART tx State Diagram 20

[Figure 13:](#_3dy6vkm) Relevant GPIO\_1 Pin Assignment table from DE1-SoC User Manual 21

Figure 14: Pinout Diagram from DE1-SoC User Manual 21

[Figure 15:](#_3dy6vkm) LED Pin Assignment table from DE1-SoC User Manual 21

Figure 16: Switch and Button Pin Assignment table from DE1-SoC User Manual 22

# References

[1] *DE1-SoC User Manual* (2014). Accessed: January 28, 2023. [Online]. Available: https://inst-fs-iad-prod.inscloudgate.net/files/3607afc4-522a-4f44-a5c1-1580f22dd310/DE1-SoC\_User\_manual.pdf?download=1&token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE2NzQxOTc2NTcsInVzZXJfaWQiOiIxMjgzOTAwMDAwMDAwMzc5MjYiLCJyZXNvdXJjZSI6Ii9maWxlcy8zNjA3YWZjNC01MjJhLTRmNDQtYTVjMS0xNTgwZjIyZGQzMTAvREUxLVNvQ19Vc2VyX21hbnVhbC5wZGYiLCJqdGkiOiI4YzdhYjE1ZC03NTY5LTQ1YmUtYTQ4NC0xNzc4YjZmZjI1N2MiLCJob3N0Ijoib2l0Lmluc3RydWN0dXJlLmNvbSIsIm9yaWdpbmFsX3VybCI6Imh0dHBzOi8vYTEyODM5LTMwMjY0ODguY2x1c3RlcjkxLmNhbnZhcy11c2VyLWNvbnRlbnQuY29tL2NvdXJzZXMvMTI4Mzl-MTc2OTUvZmlsZXMvMTI4Mzl-.rAUBUNddvg1bZIEEdqQl46VLR4ma64u9MRfpNPFygfuL28WGrF8huwyLI-6aA0d8GSLMLjCmXed53\_VN1RCaog

[2] *ALTERA Cyclone V SoC Development & Education Board (DE1-SoC)* (2014). Accessed: January 28, 2023. [Online]. Available: https://oit.instructure.com/courses/17695/files/3026486/download?download\_frd=1