3/11/2022

Christopher Thomas

CST 231 – Digital System Design I

Lab 5

Tx

Christopher Thomas

Table of Contents

[Abstract 2](#_Toc97922159)

[Introduction 3](#_Toc97922160)

[Design 3](#_Toc97922161)

[Physical hardware design 3](#_Toc97922162)

[Synthesized hardware design 3](#_Toc97922163)

[Modules 3](#_Toc97922164)

[Simulation and Testing 4](#_Toc97922165)

[Problems 4](#_Toc97922166)

[Results and Conclusion 4](#_Toc97922167)

[Appendix – A 5](#_Toc97922168)

[A-1 Pin Mappings Table 5](#_Toc97922169)

[A-2 RTL Diagrams 5](#_Toc97922170)

[A-2.a Lab05 5](#_Toc97922171)

[A-2.b clk\_divider 5](#_Toc97922172)

[A-2.c transmit\_control 6](#_Toc97922173)

[A-2.d transmit 6](#_Toc97922174)

[A-3 State Machine Diagrams 7](#_Toc97922175)

[A-3.a transmit\_control 7](#_Toc97922176)

[A-3.b transmit 7](#_Toc97922177)

[A-4 Module Code 8](#_Toc97922178)

[A-4.a Lab05 8](#_Toc97922179)

[A-4.b clk\_divider 8](#_Toc97922180)

[A-4.c transmit\_control 8](#_Toc97922181)

[A-4.d transmit 10](#_Toc97922182)

[Table of Figures 15](#_Toc97922183)

[References 16](#_Toc97922184)

# Abstract

There are two parts to this lab. In the first part you will hook the data to be transmitted to the switches on the DE1-SoC board. When push button one is pressed, the system will transmit a single character on the UART. You will receive this using your USB to serial dongle that came with the lab supplies. This will require creating a lockout for the button, so it only transmits a single character per button press.

For the second part of the lab, instead of getting the data from the switches. It will send a message on the UART to the PC. That message will be “Hello You Name Here”.

# Introduction

The purpose of this lab is to become familiar with the UART transmission. It will require sending a single character to the computer through serial communication. The character will be set by the switches on the board and then transmitted to PuTTy. The second part of the lab will have a pre-set message of “Hello Your Name” to be sent via serial communication.

Diagram

Description automatically generated

Figure : Top Level Diagram

# Design

## Physical hardware design

The only physical hardware was connecting the serial device Tx to the GPIO.

## Synthesized hardware design

### Modules

#### Module: clk\_divider

The clk\_divider module divides the 50MHz default signal from the device to the 9600Hz we need it to be for this application. This is achived by making a clk\_out output that only inverts its value after the onboard clock signal has inverted a certain number of times to slow the clock to the desired speed.

#### Module: transmit

The transmit module handles transmitting serial data from the device. It is a state machine that sends out a start bit, 8 data bits, a parity bit, and a stop bit. This module is running on a clock that is 9600Hz

#### Module: transmit\_control

The transmit\_control module worked as a lockout for the button press so you didn’t get multiple presses. You had to release the button to transmit again.

# Simulation and Testing

Simulating the transmitter required giving it a low for the but input and then monitoring that it traveled correctly through the states and sent out each data bit correctly.

# Problems

Some problems to keep in mind when trying to complete this lab are that the buttons are pull-up, so they are at a logic 1 by default. And that a case statement works well for indexing a register. This was specifically an issue in part 2 of the lab when we had a register that contained a full string and wanted to access a single character at a time. It was necessary to access 8 bits at a time but from a specific range, this is where the case statement was incredibly useful. Another issue that arose was my calculated parameter for my clk\_divider was too fast because of slight differences in the crystal.

# Results and Conclusion

Overall, the lab is self-explanatory with use of the data sheet for the UART device. The biggest issues faced were inconsistencies with the clock crystal and dealing with a string. If the clock is slightly too fast or too slow, you may see the correct character being displayed but being followed by some garbage. This will require some adjustments to the clock speed.

# Appendix – A

## A-1 Pin Mappings Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **FPGA Pin No.** | **Description** | **I/O Standard** |
| CLOCK\_50 | PIN\_AF14 | 50MHz clock | 3.3-V LVTTL |
| GPIO\_0 | PIN\_AC18 | GPIO\_0[0] | 3.3-V LVTTL |
| KEY | PIN\_AA14 | Button 0 | 3.3-V LVTTL |

## A-2 RTL Diagrams

### A-2.a Lab05

Diagram

Description automatically generated

### A-2.b clk\_divider

Diagram, schematic

Description automatically generated

### A-2.c transmit\_control

Diagram

Description automatically generated

### A-2.d transmit

Diagram, schematic

Description automatically generated

## A-3 State Machine Diagrams

### A-3.a transmit\_control

Diagram

Description automatically generated

### A-3.b transmit

A picture containing chain, necklet

Description automatically generated

## A-4 Module Code

### A-4.a Lab05

Text, letter

Description automatically generated

### A-4.b clk\_divider

Graphical user interface, text, application

Description automatically generated

### A-4.c transmit\_control

module transmit\_control(

input clk,

input transmit,

output reg but

);

reg [1:0] state;

parameter Idle = 2'b00,

Push = 2'b01,

Wait = 2'b10;

initial

begin

but <= 0;

state <= Idle;

end

always @(posedge clk)

begin

case(state)

Idle:

begin

but <= 0; //if the button is pushed moved to Push state

if(!transmit)

begin

state <= Push;

end

else

begin

state <= Idle; //else stay in Idle state

end

end

Push:

begin

but <= 1; //set the output to high

state <= Wait; //move to Wait state

end

Wait:

begin

but <= 0;

if(transmit) //wait for button to be released so we dont get multiple inputs

begin

state <= Idle;

end

else

begin

state <= Wait;

end

end

endcase

end

endmodule

### A-4.d transmit

module transmit(

input clk,

input but,

output reg data\_out = 1

);

reg [2:0] state;

reg [3:0] count;

reg [18\*8-1:0] message = "Hello Chris Thomas";

reg [4:0] message\_count;

reg [7:0] data;

parameter Length = 18,

Idle = 3'b000,

Start = 3'b001,

Transmit = 3'b010,

Parity = 3'b011,

Stop = 3'b100;

initial

begin

data\_out <= 1;

state <= Idle;

count <= 0;

end

always @(posedge clk)

begin

case(state)

Idle: //idle state to reset outputs and wait for button push

begin

data\_out <= 1;

count <= 0;

message\_count <= 0;

if(but)

begin

state <= Start;

end

else

begin

data\_out <= 1;

state <= Idle; //stay in idle state until button is pushed

end

end

Start:

begin

count <= 0;

case(message\_count)

5'b00000 : data <= message[143:136];

5'b00001 : data <= message[135:128];

5'b00010 : data <= message[127:120];

5'b00011 : data <= message[119:112];

5'b00100 : data <= message[111:104];

5'b00101 : data <= message[103:96];

5'b00110 : data <= message[95:88];

5'b00111 : data <= message[87:80];

5'b01000 : data <= message[79:72];

5'b01001 : data <= message[71:64];

5'b01010 : data <= message[63:56];

5'b01011 : data <= message[55:48];

5'b01100 : data <= message[47:40];

5'b01101 : data <= message[39:32];

5'b01110 : data <= message[31:24];

5'b01111 : data <= message[23:16];

5'b10000 : data <= message[15:8];

5'b10001 : data <= message[7:0];

endcase

data\_out <= 0; //drive the start bit low to start transmit

state <= Transmit;

end

Transmit:

begin

data\_out <= data[count]; //output the data bits

if(count < 4'b1000)

begin

count <= count + 1; //if count is less then the number of data bits

state <= Transmit; //increment count and stay in push state

end

else

begin

message\_count <= message\_count + 1;

state <= Parity; //Move to wait state

end

end

Parity:

begin

data\_out <= (data[0] ^ data[1] ^ data[2] ^ data[3] ^ data[4] ^ data[5] ^ data[6] ^ data[7]);

state <= Stop; //return to idle state

end

Stop:

begin

data\_out <= 1;

if(message\_count < 5'b10010)

begin

state <= Start;

end

else

begin

state <= Idle;

end

end

endcase

end

endmodule

# Table of Figures

[Figure 1: Top Level Diagram 3](#_Toc96764961)

# References

**There are no sources in the current document.**