Rensselaer Polytechnic Institute

Computer Hardware Design
ECSE 4770

Report

Lab Five
FPGA UART Laboratory

Group:
Gideon Bartlett
Joseph Catalano
Kristian Gutenmann
Nikoleta Koleva
TA: Joe Ebel
Date: 10 December, 2019
Instructor: John F. McDonald

Table of Contents

Abstract

This lab revolves around the implementation on our lab 4 design in Quartus. The interface from lab 4 was meant to take an input from a computer and feed it through an optoisolator into a UART. From there that data could be stored into a RAM until it had to be retrieved from the RAM and go through the UART and optoisolator to get back to the computer display. We struggled greatly with being able to implement this lab as were unable to get our desired functionality in lab 4 in the first place. This meant that we did not spend a great deal of time on this lab and we were unable to get it to function properly as well.

Parts List

Part	Name	Amount
7404	Hex Inverter	2
7408	Quad 2-Input AND Gate	1
7421	Dual 4-Input AND Gate	1
7432	Quad 2-Input OR Gate	1
7473	JK Flip-Flop	1
7474	Dual D Flip-Flop	1
7489	RAM	2
74163	4-Bit Binary Counter, Synchronous Reset	1
6402	UART	1
4N30	Optoisolator	1
4N37	Optoisolator	1
220 Ω	R1	1
560 Ω	R2	1
220 Ω	R3	1
470 Ω	R4	1

Description of Internal Controller Operation

Introduction

The URAT, RAM, and Optoisolator circuit will be used as a demonstration of I/O interfacing. A circuit will be designed to interface and allow for effective communication between the URAT and the RAM that stores 16 characters. The circuitry for the connection of the UART and RAM will be designed as well as a state machine for stepping through states. The optoisolator circuit was supplied values for resistors were chosen based on the components.

Current Loop

The first step of this lab was designing and implementing the communication between the I/O and the UART chip. This current loop in its most basic form was provided in the lab document. The main task for the group was calculating the resistor values to be used in order for the interface to function as expected. The restraint for the loop was that it had to be a 20mA current loop, meaning that when driven high, there would be 20mA of current going through the transmitter and subsequent receiver loops. The provided circuit contains two optocouplers, which are tasked with isolating the UART from the terminals. Isolating the UART is accomplished by a diode that transmits to the phototransistor.

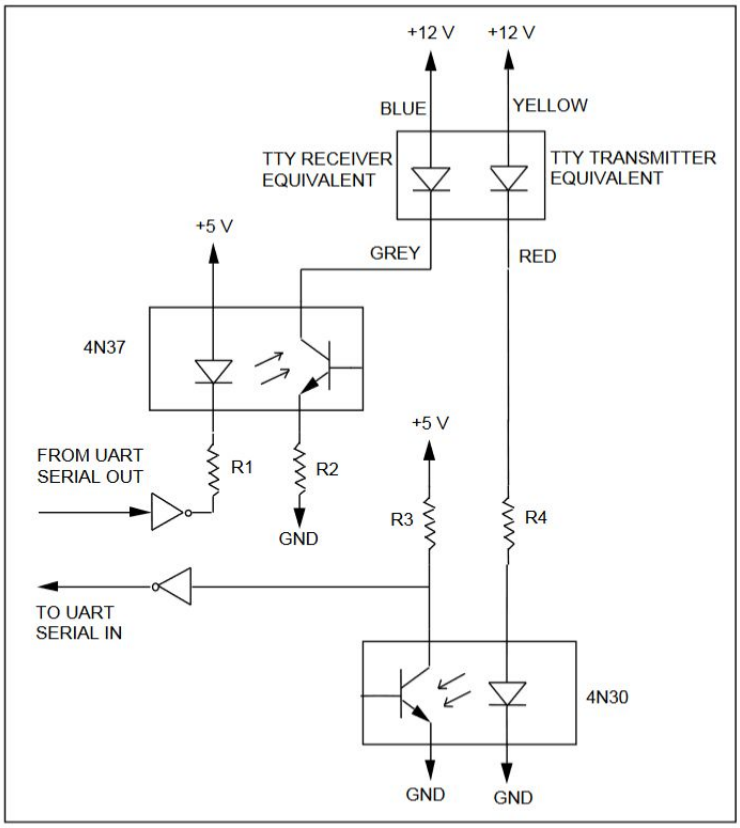


Figure _: Basic 20mA current loop provided

Provided below are the calculations for the four transistors needed to fully implement the current loop circuit.

$V_{CE\text{-}SAT} = 0.3$ V
$V_{TTY\text{-}DROP} = 1.5$ V
$V_{F\text{-}AVG} = ((0.7+1.7)/2)$ V $= 1.2$ V

R1:
$I_F = 20$mA
$V_{R1} = 5\text{-}1.2 = 3.8$V
$R1 = V_{R1}/I_F$
$R1 = 190\Omega$
Note: Because there were no 190Ω resistors available, we used the closest (higher) resistor value, which was 220Ω

R2:
$I_F = 20$mA
$V_{R2} = 12 - 1.5 - 0.3 = 10.2$ V
$R2 = V_{R2}/I_F$
$R2 = 510\Omega$
Note: Because there were no 510Ω resistors available, we used the closest (higher) resistor value, which was 560Ω

R3:
$I_F = 20$mA
$V_{R3} = 5 - V_{CE\text{-}SAT} = 4$V ($V_{CE\text{-}SAT}$ is 1V in this instance, as per datasheet)
$R3 = V_{R3}/I_F$
$R3 = 200\Omega$
Note: Because there were no 200Ω resistors available, we used the closest (higher) resistor value, which was 22Ω

R4:
$I_F = 20$mA
$V_{R4} = 12 - 1.5 - 1.2$ V $= 9.3$ V
$R4 = V_{R4}/I_F$
$R4 = 465\Omega$
Note: Because there were no 465Ω resistors available, we used the closest (higher) resistor value, which was 470Ω

Amplitude Modulation

The base lead of the output transistor could be used to amplify the signal. The optoisolator could be used as an amplitude modulator by placing the BJT portion of the optoisolator in an H-circuit.

For the machine design, the optoisolator circuit will connect the interface input to the UART. The UART will be connected to two RAM chips. One RAM chip will hold the low order bits while the other RAM chip holds the high order bits. A T-Flip-Flip will keep track of the state. When the output of the flip-flop is low then data is being written to the RAMs but when it is high then the data is being read from the RAMs to the UART. A 74163 counter is used to keep track of where the data is being stored to and extracted from the RAMs. As data goes to the RAMs the counter counts up so that the next piece of data goes to the next available slot on the RAM. Once the counter reaches its maximum value the flip flop changes and the state changes as well. The machine enters read mode and the counter now counts through all of the data in the RAM and sends it to the UART. This repeats back and forth until the machine is turned off.

UART

The UART chip used was a UART 6402 chip. There were twenty input connections that we had to understand and properly wire-up. In the initial circuit design, not all input connections were connected and some had wrong logic, which led to initial tests failing. Due to this, the whole design was scrapped and new input connections were brainstormed. The following are the redacted UART input connections (the missing numbers are all output connections):

Pin 4 - LO (S_o), HI (S_1)
Pin 16 - LO
Pin 17 - 153.6kHz CLK
Pin 18 - LO (S_o), only after receiving HI from Pin 19, HI (S_1)
Pin 20 - Serial IN
Pin 21 - Power on/Reset
Pin 23 - HI (S_o), LO (S_1), only when all data is loaded-in per word
Pins 26-33 - Transmitter buffer register
Pin 34 - HI
Pin 35 - HI
Pin 36 - LO
Pin 37 - HI
Pin 38 - HI
Pin 39 - LO (Don't Care)
Pin 40 - 153.6kHz CLK

Memory

For memory, the RAM chips used were two 7489s. Here, memory enable (ME) was driven to ground so that it would always be enabled since ME is active low. The write read (WR), also active low, would always be in its read state, except when data received (DR, active low) is HI. Write read would read either DR or Q which would be in its write state.

Counter

We used a 74163 counter to keep track of our address locations for storing data from the UART to the RAM and for keeping track of data being transferred from the RAM to the UART. Whenever data is being transferred between the RAM and the UART the Counter begins counting up from 0. As it does so it counts up through all available RAM locations. Once all locations have been counted through then the counter knows that it has used all of the RAM data locations. This then causes the counter to trigger a change of state in the state machine. If the system was in Read mode then it moves to Write mode and vice versa. This way the counter makes sure that the RAM is totally used.

Reset I/O

We installed a Reset Button on our machine to allow us to set everything in our design back to knowable values on startup. This was done by setting up a button that performed three reset tasks. The first task was to reset the Counter. This was done using the clear pin on the Counter. A similar method was used on the reset pin of the State Machine Flip Flop. This was the second reset task. The third, and final, reset task was performed on the UART. Pin 21, the Master Reset Pin is driven high on "startup" as the button is pressed so that the UART can be expected to function normally.

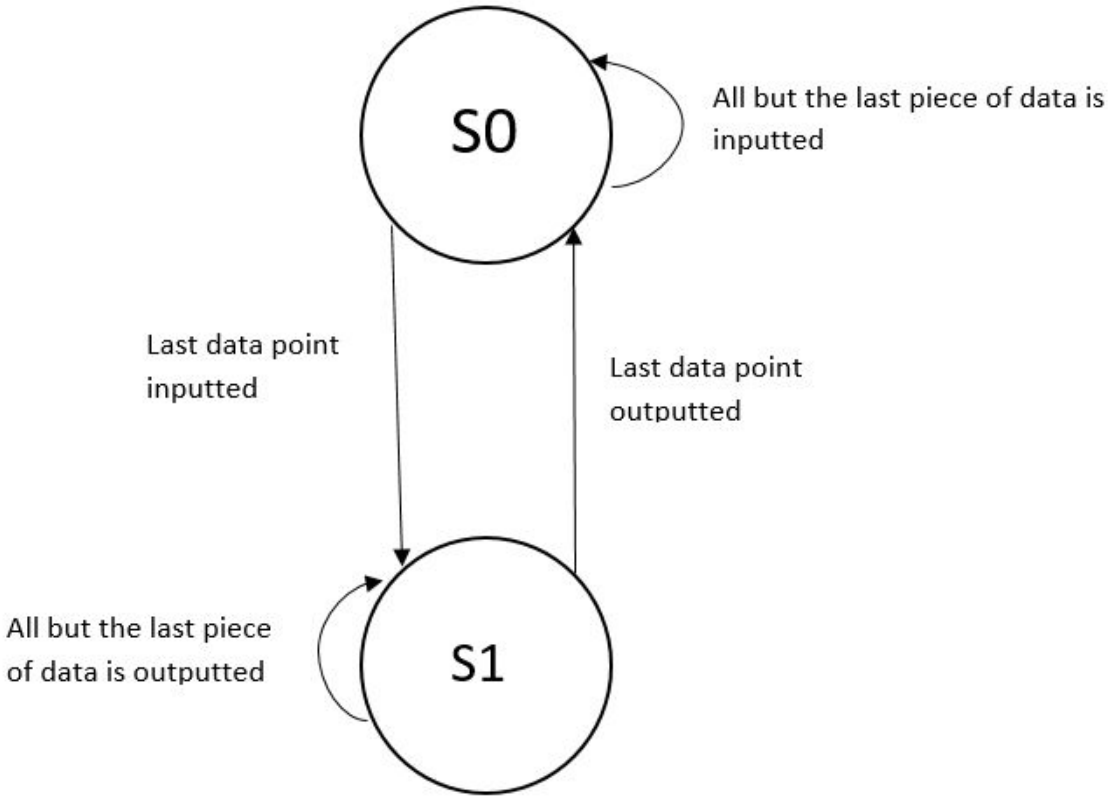
State Machine

Our state machine relies only on two states that are flipped through as proper conditions are met in each state. This setup if perfect for a T-Flip Flop or Toggle Flip-Flop based design. This flip flop changes outputs every time it receives a pulse. By having the T Flip-Flop receive a pulse once the RAM had either read from or written to each possible location we could make sure that the state would change independently of us needing to manually monitor the state. The outputs of the state machine would then let us know if we were in read mode or write mode.

Concise Description of Debugging Process

The debugging process for this lab was heavily tied to the debugging process for lab 4. Since we were unable to finish debugging lab 4 we did not have time to double check debugging once the lab 4 design was put into Quartus.

State Diagram



Logic Equations

Chip Name	Pin Name	Pin #	Logic Equation
RAM 7489	WR	3	DR OR Q
74163	LOAD	9	State 0 State 1 Q'*ABCD*(DR) + Q*ABCD*(TBRE)
74163	ENP	7	State 0 State 1 Q'*DR + Q*TBRE
UART	RRD	4	Q
UART	DRR	18	Q' NAND DR
UART	TBRL	23	Q NAND TBRE

Full Logic Diagram

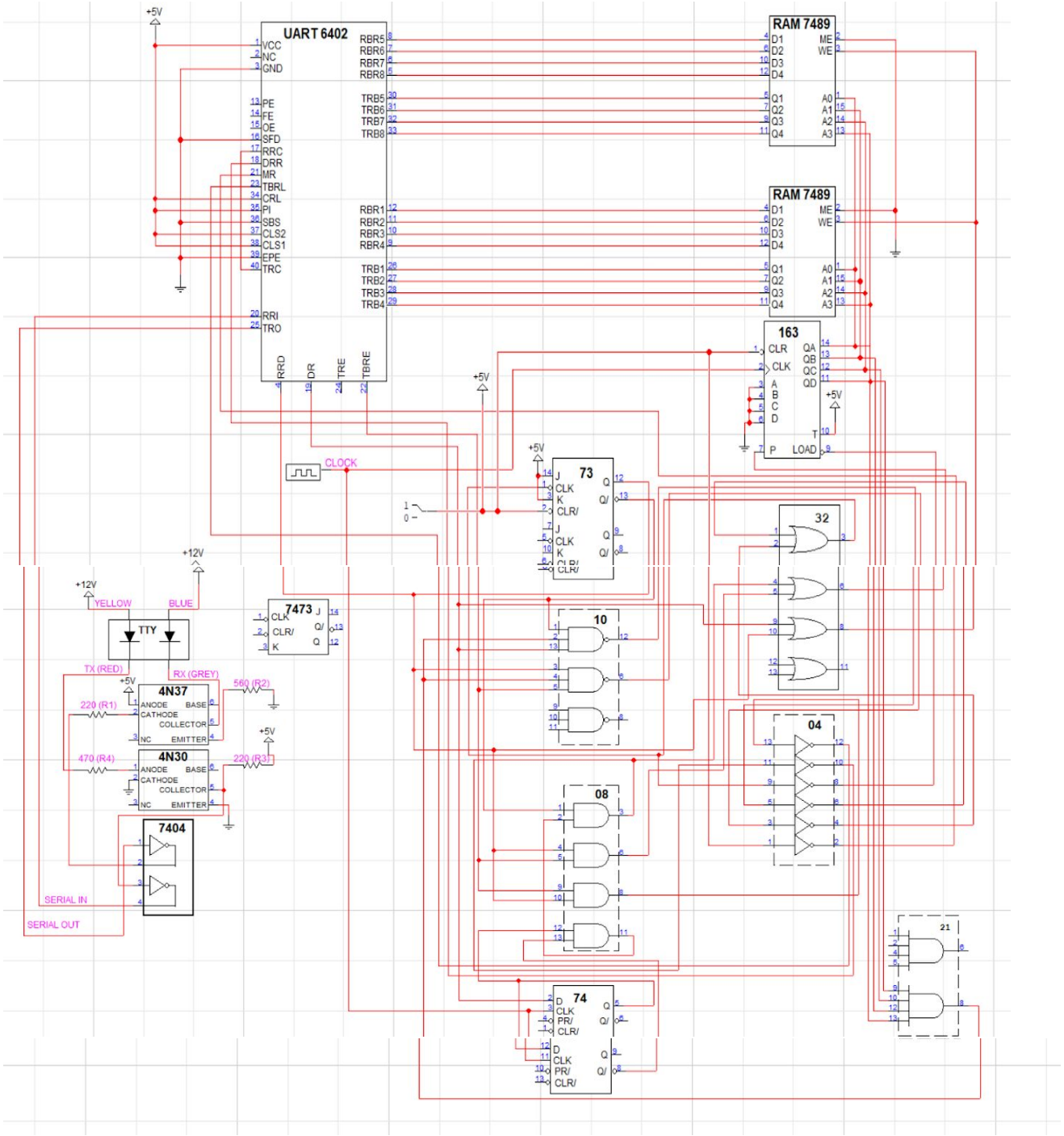
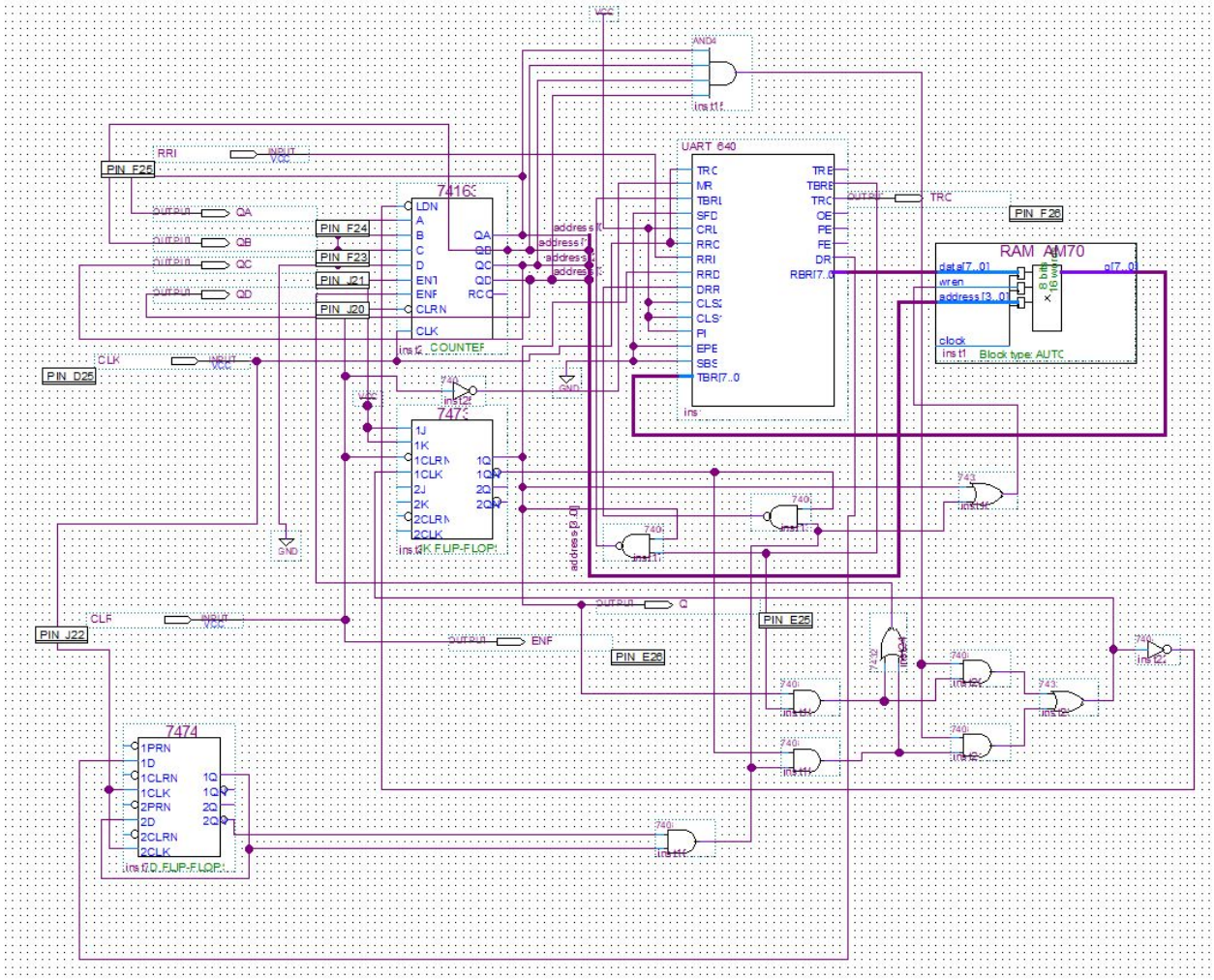
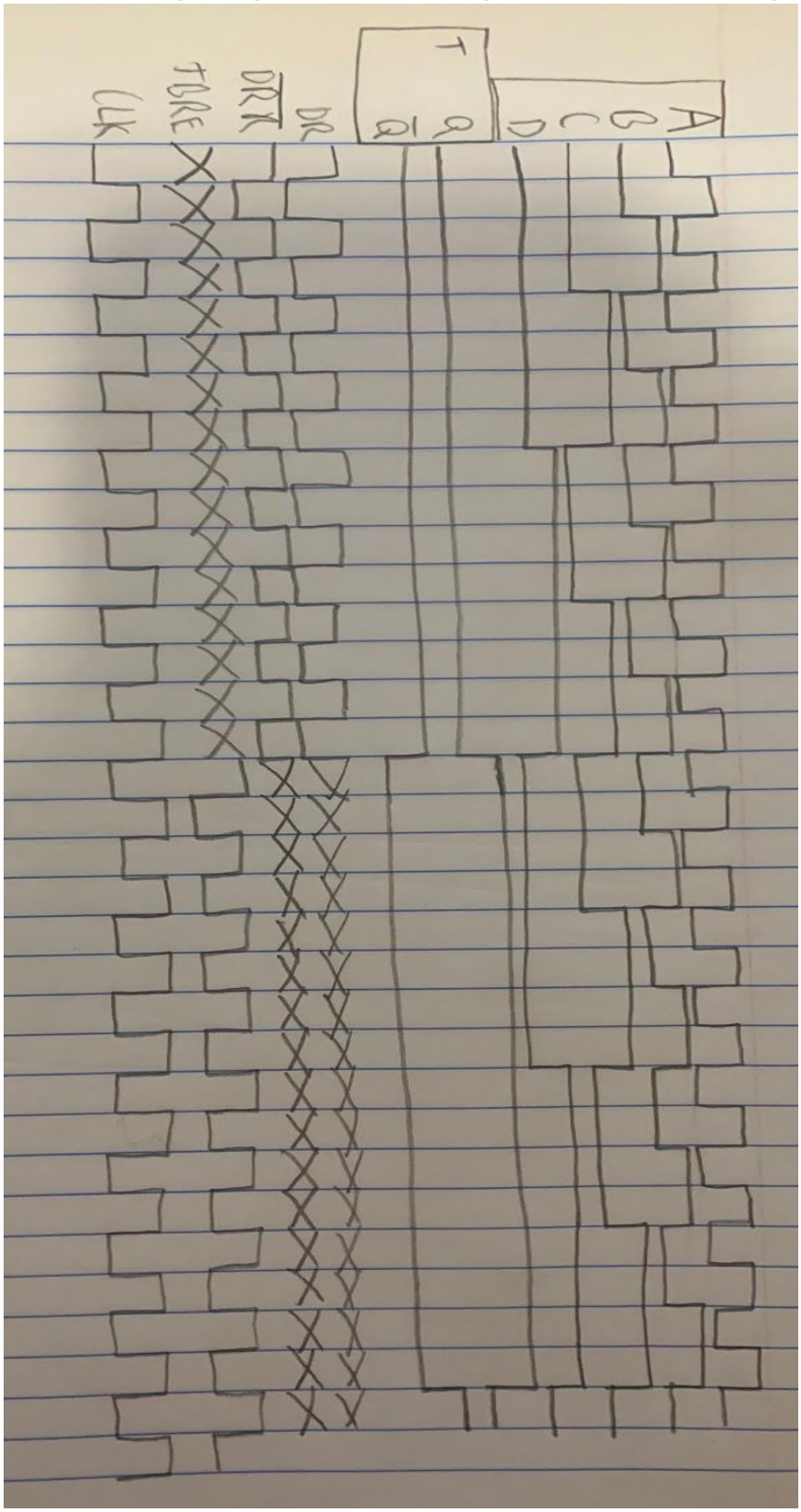


Photo of Final Design



Final Timing Diagram (Ideal Diagram Based on Design)



Conclusion

This lab centered on implementing a UART circuit design in quartus. It posed a great challenge to us as timing and logic were big hurdles in this project. The process for design in this lab took a great deal of time as we struggled greatly with the implementation of lab 4. As a result we didn't have time to properly test and debug this design.