A red and white sign with yellow letters

AI-generated content may be incorrect.

Department of Electrical & Computer Engineering Technology (ECET)

School of Engineering, Technology, and Advanced Manufacturing (ETAM)

CET3136C - 22840

Logic Devices Programming

Project Notes

UART/FPGA-Based Security System

**Submitted by**

**Anthony Sevarino**

**Supervised by**

**Professor Ashley Evans**

**Submitted:**

**N/A**

**Physical Connections**

* UART requires a transmitter and receiver (**TX**, **RX**) connection between two boards. Since this is a bidirectional system, each board’s **TX** and **RX** pins must be used.
  + Those pins for now will be PIN\_V10 (GPIO 0) for o\_**TX**\_Serial and PIN\_W10 (GPIO 1) for i\_**RX**\_Serial.
  + I have to remember that most pints on the board operate based on **3.3V LVTTL** logic, must use board gnd.
  + I found a good intel community discussion board regarding a similar topic for quartus.
    - <https://community.intel.com/t5/Intel-Quartus-Prime-Software/UART/td-p/1427182>

**Clock and Baud Rate**

* In any clock-based system, the signal is changing, often several times per second. Thus, it is necessary to determine how many times the signal might change per second, as the data in this communication standard (UART) will be transmitted a single bit at a time over a single connection.
  + Not to be confused with bitrate, as that refers to the number of bits transmitted per second, while **Baud** is the number of times the signal can change per second
* I have to determine the **Baud** **rate** to use for the system, with the information that the clock operates on a 50MHz frequency in mind.
  + Common Baud rates are: 9600, 19200, 38400, and 115200.
  + To determine the Baud rate, I’m going to arbitrarily select 115200 Baud to derive the following value:
    - The above derived value represents the clock cycle count.
* <https://support.sbg-systems.com/sc/kb/latest/technology-insights/uart-baud-rate-and-output-rate#:~:text=The%20Baud%20rate%2C%20expressed%20in,each%20side%20of%20the%20communication>.
* <https://solace.com/blog/what-is-baud-rate-bit-rate/#:~:text=Summary,push%20through%20the%20transmission%20system>.

**Universal Asynchronous Receiver/Transmitter**

* I have to determine how many bits I would like to send over this system. It seems the standard is 10 bits, however only 8 of those are data bits. There needs to be at least two control bits, one for a start bit and one for an end bit, and this doesn’t even cover if I’d like any additional control bits sent for flags.
  + There are 10 switches available for use on each board, I’ll start with a system which uses this conventional 10-bits, but may change this later to accommodate all ten switches.
* This must operate based on a “tick” or “step” in the clock, which is represented by a cycle in the clock determined by its operating frequency and **baud rate**.
* Before I design the final system, I should create a **Moore FSM** to ensure I can transmit data from one board to another. This initial step of board communication is complex enough for a student of my skill level, so I don’t want to over complicate it early on.
  + The benefit of first creating a **Moore FSM** is that I won’t have to worry about inputs from either of the boards at first, and could just focus on changing states over UART communication. At least for the Transmitter
* I will use Lab 10 Part B for help in programming this component.
* UART transmission will operate on logic-low, or 0 start bit. Waiting state should be logic high or 1.
  + Need program flags to determine whether transmission is occurring and index for data bits
* For the Receiver, at I’ll primarily use a Moore machine for the aforementioned reasons, however, since the idle end state may depend on inputs, it will be a hybrid Moore and Mealy FSM.
* <https://forum.digikey.com/t/uart-vhdl/12670>

**First Test of Communication Protocol**

* I’m going to create two separate projects, one for each board, and run them concurrently in different instances of Quartus. One will send data to the other board, and that receiving board will display LEDs in sequence with the data it receives.
* Content from chapter 9 will aid me here, specifically with bitwise rotations for bit shift logic.
* Lab 3C helped in remember proper syntax for structural based systems, specifically with port mapping