**Department of Electrical & Computer Engineering Technology (ECET)**
**School of Engineering, Technology, and Advanced Manufacturing (ETAM)**

# CET3136C - 22840

## Logic Devices Programming
## Project Proposal

# UART/FPGA-Based Security System

**Submitted by**

**Anthony Sevarino**

**Supervised by**

**Professor Ashley Evans**

**Submitted:**

**March 12th, 2025**

# Project Description

This project aims to create a lock-and-key system which is programmed in VHDL, utilizing two MAX 10 DE-Lite Intel FPGA development boards in tandem using a Universal Asynchronous Receiver-Transmitter (UART) communication design. UART is a hardware communication protocol which allows for the asynchronous communication between two hardware devices, allowing for a modular, custom interface between the two (i.e. separate clock signal). A UART system will convert data from one device into data to be transmitted to another.

As this is a hardware based communication protocol, a fundamental understanding of the pinouts for the development board being used it vital to ensuring successful completion of the project. A successful end project will demonstrate the communication between a “key” board with a “lock” board, with the added option to convert a “key” board into a “key decrypting” board. Applications of this device include any scenario wherein a lock and key security system would prove beneficial, such as a safe or a door.

# Requirements Specification

***Project Function***

This project incorporates two microcontroller development boards into a single security system, wherein one board functions as a “key” board, and the other as the “lock” board. The “lock” board holds a 10-bit code which is determined by the state of each of the 10 switches found on the MAX 10DE-lite development board. In order for the “lock” board’s hex displays to print the result “unlocked”, the “lock” board must match these switch inputs exactly, acting as the teeth of a key. If the “key” board is unable to match the required code, the user has the option of changing the board into a decrypting state. In this state, the “key” board will extract the lock code from the “lock” board, and indicate to the user which switches must be flipped on the “key” board by lighting the respective LEDs above each switch.

In order to accomplish this, the following parameters will be followed by the circuit:

- Lock board stores 10-bit code based on switch positions.
- Key board (if in key mode) sends current switch position code to Lock board via Universal Asynchronous Receiver-Transmitter protocol
- If codes match, Lock board displays “unlocked” on hex displays.
- Key board displays key or decrypt mode based on button press
- Key board will request code from lock board if in decrypt mode, transfer data to LEDs above each switch

***Inputs and Outputs***

- Inputs
    - Key Board
        - SW0 – SW9(Switches 0 through 9) (key code to attempt to unlock)
        - KEY0 (Push-button0) (toggles between key and decrypt mode)
        - UART RX (receiver pin) (receive lock code from lock)
    - Lock Board
        - SW0 – SW9(Switches 0 through 9)
        - UART RX (receiver pin) (receive offered code from key)
- Outputs
    - Key Board
        - LED0 – LED9 (Leds 0 through 9) (used to denote correct code in decrypt mode)
        - HEX0 – HEX5 (Hex displays 0 through 5) (displaying key or decrypt mode)
        - UART TX (transmitter pin) (transmit offered code to lock)
    - Lock Board
        - HEX0 – HEX5 (Hex displays 0 through 5) (displaying locked or unlocked state)
        - UART TX (transmitter pin) (transmit code for decryption to key)

***Required Equipment***

- MAX 10 DE-Lite Intel FPGA Development Board (x2)
- Quartus II Software
- USB Blaster
- Pin Connector Wires
- USB Type A Cable (x2)

***Design Type***

This project will likely offer all three styles of VHDL coding, those being structural, behavioral, and dataflow coding, and a Universal Asynchronous Receiver-Transmitter communication protocol will be used to transfer data bidirectionally between the two development boards.

The following is a state diagram form the UART communication protocol:

Lock Board

- Idle: waiting for the clock
- Loading data: Reading switch inputs
- Transmitting data: sending data from one board to another
- Wait: Holding before receiving next data

Key Board

- Idle: waiting for the clock
- Receiving data: Reading 10-bit code

- Comparing data: Check received input compared to local input
- Helper: showing LEDs for correct code

# Project Schedule

| Task | Expected Completion Date | Solution | Potential Issues |
|---|---|---|---|
| Design UART Communication Protocol | March 20th | 12-bit message format (start, 10-bit code, end), write transmitter and receiver modules | Difficulty understanding UART protocol, transmission issues |
| Test board communication/fixes for UART finished | March 22nd | Either simulate or use testbenching | Issues would mean a redesign of UART is required, if so, also fix by this date |
| Signal sending and receiving | March 28th | Implement UART RX and TX modules and test with pattern signals. | If previous step completed successfully, only issues would be logic regarding input and output signals |
| "Lock" board holding code | March 30th | Storing result in data transferrable packet | Ensuring input from switches matches output |
| "Key" board input comparison | March 31st | Not necessarily against "lock" code, just comparison logic in general | Ensuring proper Boolean logic comparison with any value |
| Unlocking and locking between boards | April 2nd | Using previously completed UART communication protocol in tandem with code generation and comparison logic | Transferring the correct data to be used for comparison, and using the correct comparison logic |
| **PROJECT CHECK IN** | April 3rd | - | - |
| Design Decryption protocol | April 6th | Request current code from "lock" board, send to respective LED signals | Assigning correct data from different inputs and ensuring correct LED placement relative to switches |
| Final Debugging and Any Solutions | April 15th | Depending on issues throughout project, solve | Depending on issues throughout project |
| **PROJECT DEMO** | April 17th | - | - |
| Write Report | April 20th | - | - |
| Finish Presentation | April 22nd | - | - |
| **Final Presentation** | April 23rd | - | - |