BTP Report:
Communication System with Hardware Encryption

Mohil Patel - 160070002

Guide:
Prof. Madhav Desai

Autumn 2019

Contents

1 Introduction

Aim: The project aimed at building an end-to-end communication link with hardware encryption included in the flow to build a secure communication link.

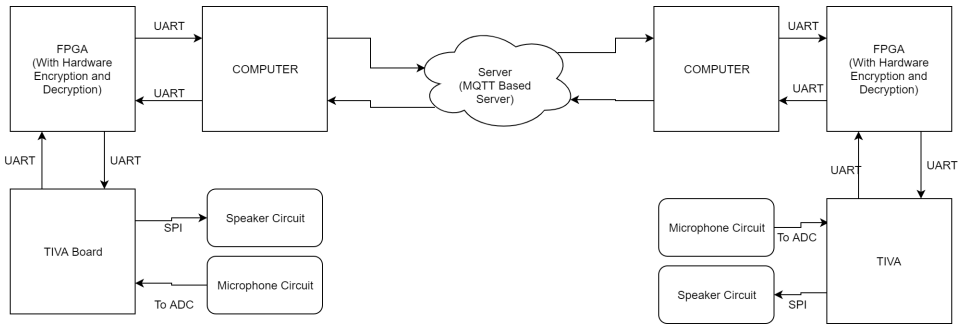

Figure 1: Complete Block Diagram of the System

The figure 1 shows the complete block diagram of the final design system. The audio in the system is captured using a microphone circuit which is digitized using TIVA on board ADC into a 12 bit number. The data is then send to FPGA Board through UART, where it is encrypted using an AES module design for communication link. The data is then flowed to the PC via UART from where it is send via a MQTT based server to the other side. On the receiver side the data is flowed through decryption unit, then TIVA and finally into Speaker Circuit. The connection is full duplex and both the clients can communicate with each other in real-time.

The following sections are as, section 2 will cover the analog circuit details of speaker and analog circuits, section 3 will cover details of TIVA board code & important configurations, section 4 will cover details of regarding FPGA & Encryption Engine and section 5 will cover details regarding the Server.

2 Analog Circuits

2.1 Microphone Circuit

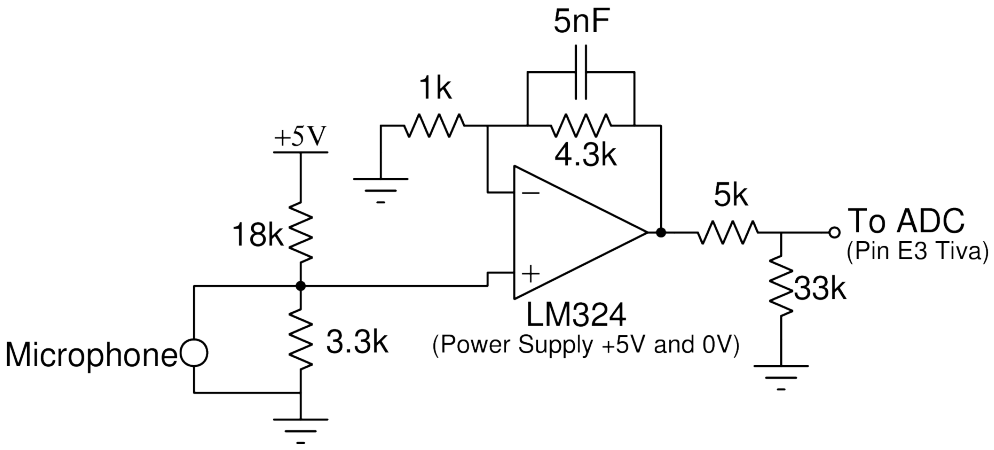

Figure 2: Microphone Circuit Diagram

The figure 2 describes the circuit for the microphone which was used in the system. The circuit uses a single supply OPAMP to ampilfy the signal from the microphone to appropriate level. Additional as the output of OPAMP is converted to digital data using on-board TIVA ADC, we have to ensure that the signal does not exceed the 3.3V, which is used as reference by TIVA ADC. To ensure that we added additional 5k and 33k to reduce the voltage to appropriate level. All the power is supplied to the circuit is provided by the TIVA Board itself.

2.2 Speaker Circuit

The figure 3 details the speaker circuit along with the DAC used in the project. In the project we have used the IC MCP4921 as the DAC. The data is transmitted to the IC using SPI communication, via the pins CSbar, SCK and SDI. The DAC requires 3.3V as Vcc which is supplied through TIVA Board. The Output of the DAC is passed through a speaker with controllable volume control, adjusted using the 10k resistor pot. The speakers generally have low-impedance and thus requires higher current outputs from the OPAMP, thus the circuit uses LM386 as it can provide high current along with low impedance.

The complete analog circuitry is designed ensuring that all the power supplies are single ended as it will make the transition to a battery based circuit much easier.

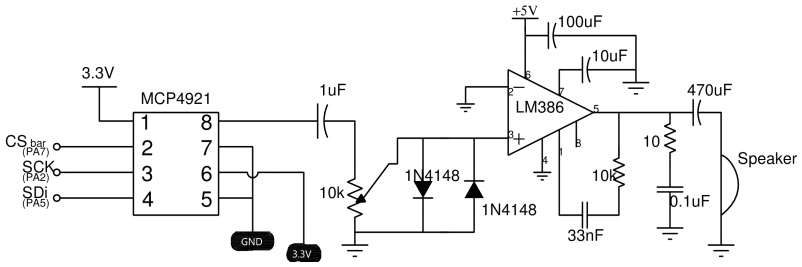

Figure 3: Speaker Circuit Diagram

3 Microcontroller Details

In the project we have used the TIVA TM4C123G LaunchPad Board as the central control unit from data handling. These are the following tasks that the TIVA Board handles:

1. Digitizing the analog data using ADC
2. Ensuring 8kHz sampling of the data
3. Communicating data to FPGA
4. Transmitting SPI data to the DAC
5. Proper Initialization of Encryption Engine

3.1 ADC: Microphone to TIVA

The ADC is setup in the Trigger Mode. In this mode whenever the microcontroller requests for the ADC, only then the data is sample. In the microcontroller the algorithm was build using Timer Interrupt to sample the data every $125\mu s$, ensuring a **sampling rate of 8kHz**.

The ADC outputs a 12 bit number which is then transmitted to the FPGA using UART communication on the FPGA.

3.2 UART: TIVA to FPGA

The ADC sampled data is 12 bits long, which is transmitted to FPGA using UART. UART is setup to transmit 8 bits at a time, thus every sample is broken down in two 8 bits samples before transmission. As the sampling rate is 8kHz this enforced the UART link to operate at **230400 baudrate**. Thus the UART is configured to transmit 8 bits at baudrate of 230400.

3.3 SPI: TIVA to DAC

To ensure proper reconstruction of the signal the data is transmitted to the DAC every $125\mu s$ only (i.e. 8kHz). The data received by the UART is stored is an array and every $125\mu s$ the data is transmitted using SPI to the DAC.

3.4 Encryption Module Initialization

The encryption module requires SBOX and RCON values for the computation, these values are transmitted to the module via the microcontroller in the initial phase before the real data is transmitted.

4 FPGA & Encryption Module

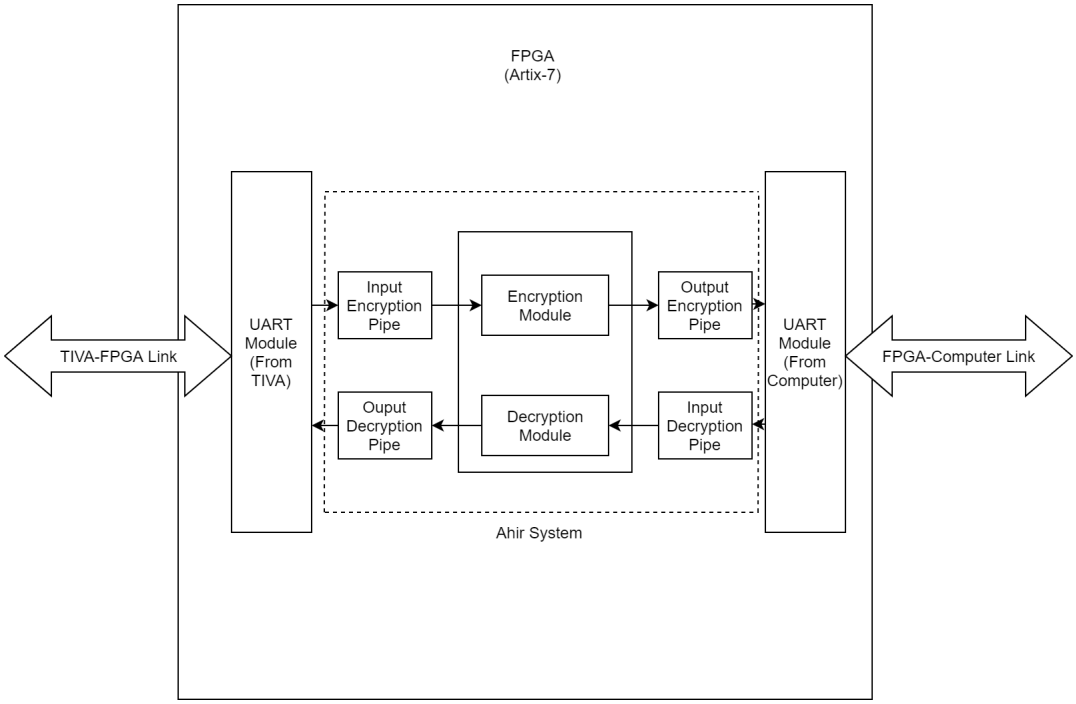

Figure 4: Encryption Block Diagram

The encryption module is designed using AHIR system. It requires a total of **5851 LUTs and 12767 Flip-Flops** for the complete design. The module works in counter mode.

The Encryption Module consists of multiple components, which are as follows:

1. Initialization Phase
2. Encryption Module
3. Decryption Module

Following sections cover these parts in more details.

4.1 Initialization Phase

There are multiple things that happen in this phase.
Firstly, the data transmitted by the Microcontroller to initialize ths SBOX and RCON is received and stored into the module for later use.

After, receiving the data the module must do handshaking with the module on the other side before it can begin data transmission. This is handled as follows, first the module will send a signal 0x00 to the other module indicating that it is alive. It will then wait for 0x01 from the other module to start its process. Suppose it receives a 0x00 from the other side then it will transmit a 0x00 followed by 0x01, and will wait for 0x01, if in any case it receives more than 1 0x00 only first one will send any data, later ones are neglected. Once 0x01 is received by the module, then it will begin with its encryption/decryption modules.

4.2 Encryption Module

The implementation of AES in the module uses the counter mode. In the counter mode the data is encrypted using a counter as well as the key. After every 128 bit encryption counter is increased by one, this increase the randomness in the data and ensures that the attacker is not able to extract any data by analysis the data patterns.

Once the initialization completes, this module starts. It will wait till it receive 14 Bytes in the input encryption pipe (refer to figure 4). Once it receives 14 Bytes, it will then calculate the checksum of all the Bytes, these 14 checksum are stored in the Byte number 15 and 16. After checksum calculation completes we will have a total of 16 Bytes, or 128 bits, which are then encrypted using the counter and the key. This encrypted data is then stored in Output encryption pipe. From Output pipe the data is transmitted via the UART to Computer for further transmission.

4.3 Decryption Module

The decryption module starts in parallel to encryption module after initialization finishes. The decryption modules receives 16 Bytes in the Input Decryption Pipe (refer figure 4). Once it receives all 128 bits (16 Bytes), then it decrypts

the data using the key and the counter. Once completed the checksum is calculated for Bytes 1 to 14 and compared with the values in Bytes 15 and 16. If there is mismatch the system informs the Encryption module about the error and the whole system stops. If there is no error then the 14 Bytes are put into the Output Decryption Pipe, and the module waits for next 16 Bytes.

5 Server Details

Once the data is encrypted by the FPGA, it is transmitted to the PC via a UART. On the PC side the data is received using a multi-threaded python program. This program in one of its thread receive the data and store it in a queue, the data from this queue is accessed by another queue which send the data to server for further transmission. Once the data is received by the server it transmitted it to its corresponding client, where a thread is waiting for the server message, once the data is received it is store in another queue. Another thread then accesses this thread and sends the data to the FPGA for decryption and further transmission.

The server is based on **MQTT protocol** which is a light weight messaging protocol, especially designed for "Machine-to-Machine" or "Internet of Things" applications which generally have low-bandwidth and requires low power consumption as they are battery operated.

6 Conclusion & Acknowledgement

The complete communication system can work as full duplex communication link. The data is sampled at **8kHz**, the UART communication requires a baud rate of **230400** and the encryption module requires a total logical components of **5851 LUTs** and **12767 Flip-Flops**. The communication system shows a delay of around 1 second in speaking and hearing the signal, but it is not significant and will not hamper the natural communication flow.

I would like to thank Akshat Singhal (170070040) and Vidur Shah (183079035) for their help in the project, and Prof. Madhav Desai for his continuous guidance.