**Encrypted Radio Communication Using FPGAs**

**Objective and Motivation**

This project aims to implement encrypted radio communication between two FPGA SoCs. By leveraging software-defined radio (SDR) modules, each SoC will transmit and receive encrypted messages. The encryption and decryption will be performed using an AES encryption module implemented as a custom IP block on the FPGA, allowing for hardware acceleration to minimize latency. Unlike software-based encryption, FPGA acceleration ensures real-time secure communication. The primary goal is to establish a secure text-based communication system, with potential future expansion to audio transmission. This project provides valuable experience in cryptographic FPGA design and secure wireless data transfer.

Required Tools and Equipment:

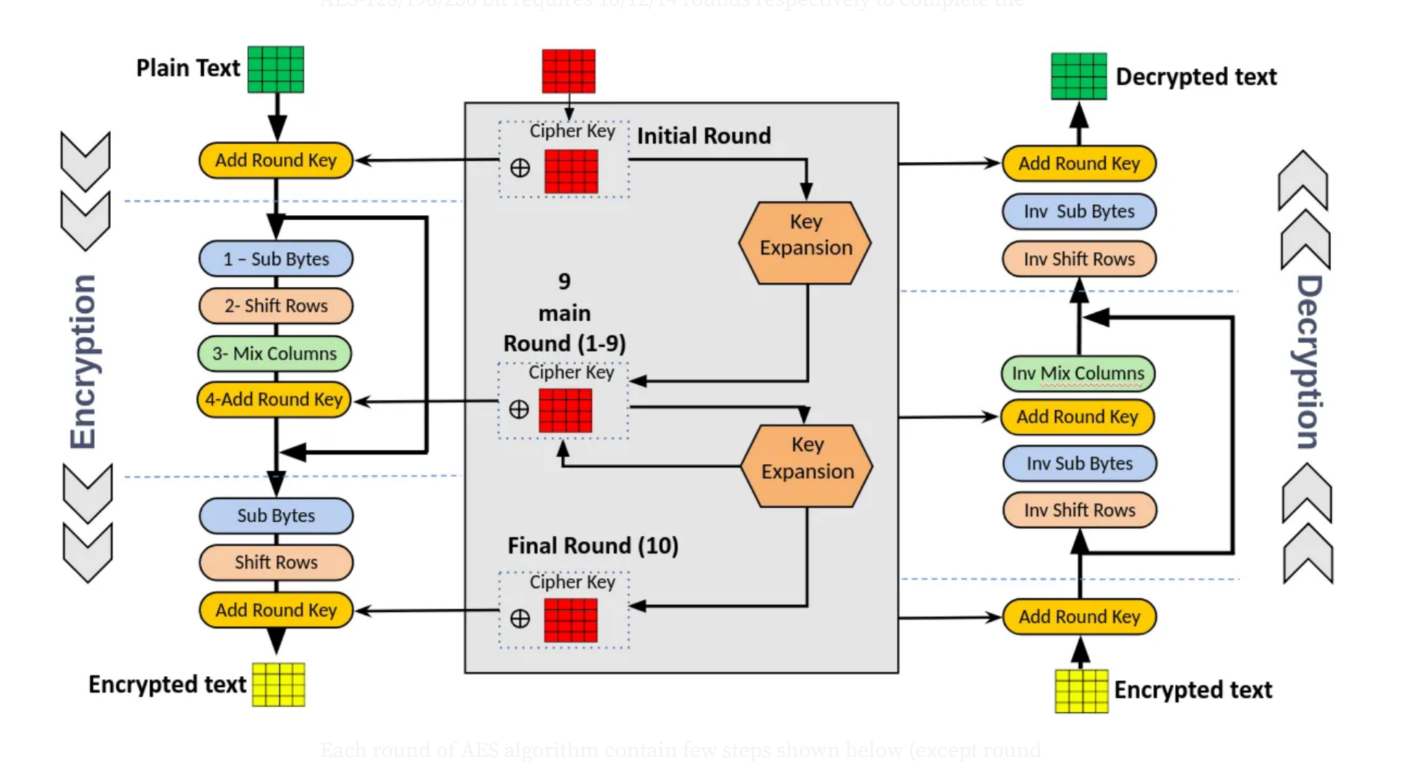
* PYNQ Z-1 FPGA SoCs (2)
* HC-12 Radio Modules (2)
* Laptop or PC with Xilinx and AES Module (2)

**Introduction/Context of Project**

Secure wireless communication is a critical requirement in modern technology, ensuring data integrity and confidentiality. Traditional encryption methods rely on software-based solutions, which can introduce latency. By implementing AES encryption on an FPGA, we can accelerate the encryption/decryption process, making real-time communication feasible. This project integrates cryptographic techniques with wireless transmission using SDR, demonstrating how hardware-based encryption can enhance security and performance.

**AES Encryption Overview**: The Advanced Encryption Standard (AES) is a symmetric encryption algorithm widely used for securing digital communications. AES operates on fixed-size blocks of data (128-bit) and supports key sizes of 128, 192, or 256 bits. For our implementation, we will be using 128 bit keys The encryption process consists of multiple rounds of substitutions, permutations, and XOR operations, making it highly secure. The number of rounds depends on the key size:

* **AES-128:** 10 rounds
* **AES-192:** 12 rounds
* **AES-256:** 14 rounds



*Figure 1. AES Encryption Process*

Figure 1 illustrates the Advanced Encryption Standard (AES) encryption and decryption process. On the left side, the encryption process begins with the plaintext, which undergoes an initial round where the "Add Round Key" operation is applied. This is followed by nine main rounds, each consisting of four steps: Sub Bytes (a non-linear byte substitution using an S-box), Shift Rows (cyclically shifting rows in the state matrix), Mix Columns (mixing data within columns using matrix multiplication), and Add Round Key (combining the state with a round key derived from the cipher key). The final round omits the Mix Columns step but retains the others. The resulting output is the encrypted text. On the right side, the decryption process reverses these transformations, starting with the encrypted text and applying inverse operations: Add Round Key, Inverse Shift Rows, Inverse Sub Bytes, and Inverse Mix Columns, ultimately restoring the original plaintext. The key expansion process generates round keys for both encryption and decryption, ensuring secure transformation between plaintext and ciphertext.

**SPI Communication with HC-12:** The HC-12 is a wireless transceiver module that supports UART communication for transmitting and receiving data over radio frequencies. The module operates in the 433 MHz band and provides a reliable wireless communication link. In this project, the HC-12 will interface with the PYNQ FPGA board through a UART connection. The SoC will configure the HC-12 to establish a wireless link, transmit encrypted messages, and receive incoming data for decryption. This setup allows secure real-time data exchange between the two FPGA SoCs.

**Implementation Details**

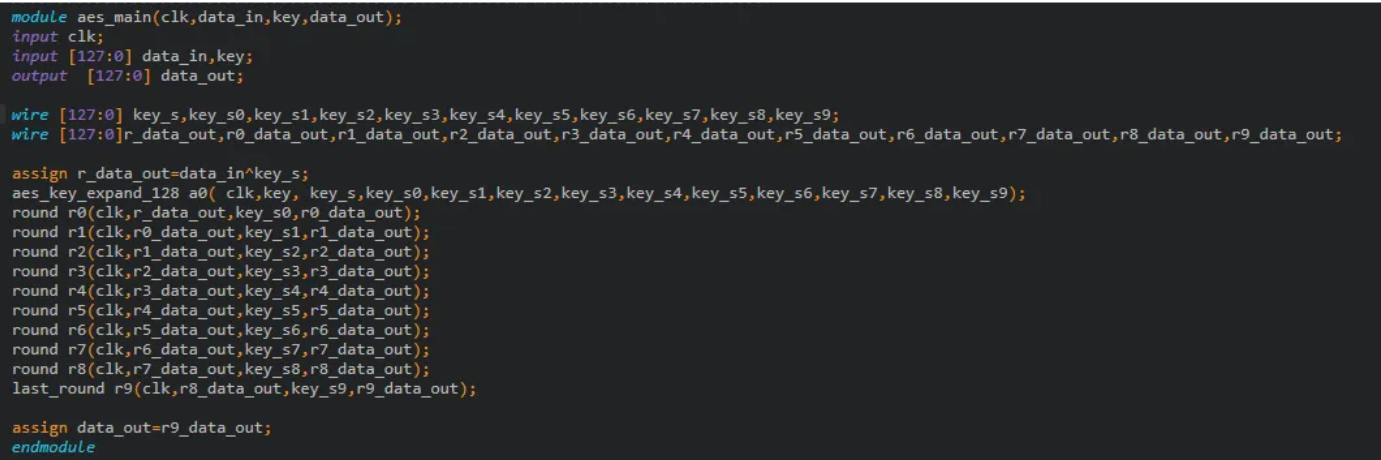
The project consists of several core components:

* AES Encryption Module: A custom IP block designed to handle AES encryption and decryption in hardware, reducing processing time compared to software solutions
* SDR Communication Modules: Python-based radio modules for handling the transmission and reception of encrypted messages
* FPGA-SoC to SDR Interfacing: Using the HC-12 module for wireless communication, with a UART connection facilitating data transfer

Process Flow:

1. A message is input to the FPGA SoC
2. The message is encrypted using the AES IP block of the FPGA
3. The encrypted data is sent via UART to the HC-12 transceiver
4. The HC-12 transceiver sends the encrypted data as a radio signal
5. The receiving HC-12 module captures the encrypted data and transmits it via UART to the second FPGA SoC
6. The second FPGA decrypts the received data using the AES IP block
7. The decrypted message is displayed in a Python terminal

In this setup, the encryption key is pre-shared between the two FPGA SoCs, ensuring secure data transmission. Figure 2 shows the top module where key expands and rounds are instantiated to produce the final AES encryption output.



*Figure 2. AES Main Verilog Code*

**Actions That Need to Be Completed**

* Set Up Development Environment
  + Install and configure Xilinx Vivado for FPGA development on both devices
  + Set up the PYNQ-Z1 board with the required drivers and software
  + Install Python libraries and other software packages for SDR communication and AES encryption testing (i.e. Overlay, pySerial, RemoteSSH)
* Design and Implement AES Encryption Module
  + Develop a custom AES IP core for FPGA acceleration using reference code
  + Verify AES functionality using simulation
  + Integrate AES IP module with the PYNQ SoC
* Integrate Uart Communication with HC-12
  + Develop UART communication firmware in Python for sending and receiving encrypted messages
  + Test data transmission and reception between the two FPGA boards
* Develop and Test Secure Data Flow
  + Handle message input and output in Python with pySerial
  + Encrypt messages in FPGA, transmit via one HC-12, and decrypt after reception from other HC-12 radio
* Expand to Audio Transmission (if time permits)
  + Implement ADC for encryption of audio messages

Expected Challenges:

* Integrating the AES IP block with the PYNQ board
* Configuring and utilizing SDR modules for wireless transmission
* Handling potential data loss due to wireless noise and implementing error correction mechanisms
* Managing latency introduced by remote development with Visual Studio RemoteSSH

**References/Source Files**

* https://github.com/hplp/2025-fpga-design-projects-EncryptedRadio
* https://medium.com/@imgouravsaini/aes-algorithm-and-its-hardware-implementation-on-fpga-a-step-by-step-guide-2bef178db736
* https://pynq.tue.nl/general/visualstudiocode-windows/
* https://www.allaboutcircuits.com/projects/understanding-and-implementing-the-hc-12-wireless-transceiver-module/