# FPGA Hardware Design Report

## Introduction

The FPGA hardware design process underwent multiple iterations as we sought to establish secure wireless communication between the FPGA and an external device. The initial plan was to implement an AES-128 encryption module for data security, but various issues forced a shift to alternative solutions. The final iteration aimed to demonstrate basic Bluetooth functionality, even if the full implementation of data processing was incomplete.

This section details the major hardware design changes, the challenges encountered at each stage, and the final implementation that will be demonstrated.

## Design Evolution and Challenges

### Plan A: AES-128 Module Implementation (Failed)

The initial approach was to implement an **AES-128 encryption module** on the FPGA to securely process authentication data. However, the module could not be validated successfully in **Vivado 2022.2**, leading to persistent issues with synthesis and hardware validation. Additionally, integrating the AES module with the FPGA’s I/O introduced unexpected pin assignment errors, preventing successful bitstream generation. Due to these roadblocks, the AES-128 approach was **abandoned** in favor of pre-built alternatives.

### Plan B: Pre-Made IP Core for Encryption (Failed)

To circumvent the validation issues in Plan A, we explored **pre-made IP cores** for encryption. The first IP core considered (**TinyAES** or a similar module) was built for a different mode of operation, where signals were left **floating** instead of being packaged into a **single RX/TX communication line**, making integration difficult.

Further attempts to use **other IP cores** were unsuccessful due to:

* Compatibility issues with **Vivado 2022.2** and **Vitis**.
* Incorrect signal mapping in pre-made IPs.
* Some IP cores being built for **older versions** of Vivado, making them incompatible with our hardware.

After multiple failed integration attempts, we **discarded** Plan B and moved to direct Bluetooth communication.

### Plan C: HC-05 Bluetooth Module Direct Connection (Failed)

With encryption no longer viable, the next approach was to establish **direct Bluetooth communication** between the FPGA and a Raspberry Pi 5 (RPi5) using an **HC-05 Bluetooth module**. The **HC-05 was physically connected to the FPGA’s PMOD ports (JB header), and initialized properly via Vitis**.

**Challenges encountered:**

* The **HC-05 module was recognized and initialized**, but it **did not show up in Bluetooth device scans** on **any phone**.
* The **MAC address was detected**, but attempting to manually connect via **bluetoothctl** on the RPi5 resulted in **persistent connection failures**.
* The **HC-05 would enter pairing mode, allow pairing, but fail to establish an actual connection**.
* Every attempted **fix suggested by ChatGPT** was tested, including manually binding RFCOMM, modifying Bluetooth profiles, and resetting the module via AT commands—**none worked**.

After **several days of debugging without success**, we shifted to Plan D.

### Plan D: FPGA as I²C Slave, RPi5 as Bluetooth Relay (Failed)

The next attempt involved **replacing the HC-05 module entirely** and using the **Raspberry Pi 5 as a Bluetooth relay**. The idea was:

1. The **RPi5 receives Bluetooth messages from a phone**.
2. The **RPi5 forwards messages over I²C to the FPGA**.
3. The **FPGA processes the received data**.

**Issues with Plan D:**

* The FPGA was **configured as an I²C slave**, but the design **failed to compile** due to hardware constraints. Project file was reverted to an older version to attempt Plan F.
* **I²C communication validation failed**, preventing message exchange between the RPi5 and FPGA.
* Due to the **time remaining before the project demo**, debugging the I²C issue became infeasible.

At this point, we **abandoned** Plan D and moved to a **simplified demonstration approach**.

### Plan F: Demonstration of Bluetooth Initialization & Message Handling

Since neither **HC-05 nor I²C communication** could be successfully implemented within the project timeline, we chose a **demonstration-based approach** (Plan F).

**Implementation:**

* The **HC-05 module is initialized and verified via the Vitis application**.
* The **RPi5 attempts to connect to the HC-05 module**—this connection will fail, as previously encountered.
* **Dummy code is implemented to handle received Bluetooth messages**, even though no actual messages will be received.

**Code Implementation for Received Messages**

To simulate the intended functionality, we implemented a message handling function that **waits for a PIN input**, then **controls a servo and LED** based on the received message.

// Wait for PIN

if (ReceivedBytes == PIN) {

xil\_printf("Pin Match Found! Opening Lock...\n\r");

usleep(1000);

// Send PWM Control Signal to Servo on JB3 (Channel 1)

XGpio\_DiscreteWrite(&GpioPWM, 1, ON);

// Send PWM Control Signal to LED on JB4 (Channel 2)

XGpio\_DiscreteWrite(&GpioPWM, 2, ON);

usleep(3000); // Keep on for 3 seconds, estimated lock turn time

xil\_printf("Lock Open!\n\r");

} else {

xil\_printf("Pin Mismatch! Try again..\n\r");

}

**Notes:**

* XGpio\_DiscreteWrite() is currently a **placeholder function**.
* The **servo (JB3) and LED (JB4) were planned to be controlled via GPIO**, but due to time constraints, **this part remains untested**.
* The demonstration will focus on **showing the Bluetooth module initializing** and the expected behavior if messages were successfully received.

## Conclusion

The FPGA hardware implementation went through **six different design plans** due to **unforeseen hardware constraints, software compatibility issues, and debugging difficulties**. Despite setbacks, the final demo will demonstrate the Bluetooth module’s **initialization and connection attempts**, along with a **simulated message handling system**.

While full communication was not established, significant progress was made in:

* **Hardware interfacing with the HC-05 Bluetooth module.**
* **Testing and debugging multiple pre-built IP cores for communication.**
* **Setting up the FPGA as an I²C slave (even though the final implementation failed).**
* **Writing message-handling logic for PIN authentication and device control.**

Given additional time, further improvements would involve:

* **Revisiting I²C communication between the RPi5 and FPGA.**
* **Replacing the HC-05 with a Bluetooth module known to work reliably with modern Linux systems.**
* **Finalizing servo and LED control logic on the FPGA.**

The final demonstration serves as a **proof of concept**, showcasing the work completed and highlighting areas for further refinement.