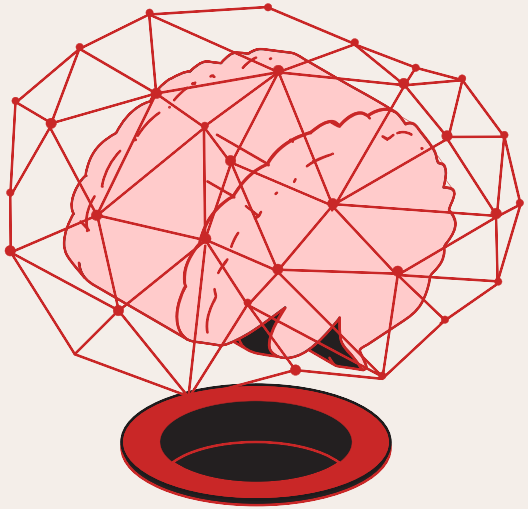
Capped Drinks

Mateo Balzola, Rahul Rangarajan,
Fairuz Abushgarh, Richard Kalich

Table of Contents



Motivation

Goals & Purposes of our Project

Motivations & Goals

What will our Project do?

- Our project utilizes the FPGA as a login management system. It stores user input and applies an *'r4c'* encryption algorithm to store the user's text as a hash.

What are the Motivations of our Project?

- Explore and implement different encryption schemes
- Late november Fidelity cybersecurity breach spiked internship curiosities
- Provide secure means of data transmission between host computer and FPGA

What is Encryption?

- Process of converting data into secure code to prevent unauthorized access
- Ensures confidentiality, integrity, and authenticity of information

02 Functionality

Summary of Our Design

Functionality

Encrypt & Decrypt

Store data & passkey with memory and encrypt and decrypt successfully

Host-to-FPGA

FPGA network module connection communicate user's computer to alert about potential unauthorized access.

Verification

Verification functionality includes the FPGA checking the user's passkey for correctness before decrypting data.

Perceiving Modules

Buzzer and SevenSeg display modules to provide real-world feedback on the correctness of the passkey input.

03 Specifications

Requirements & Constraints

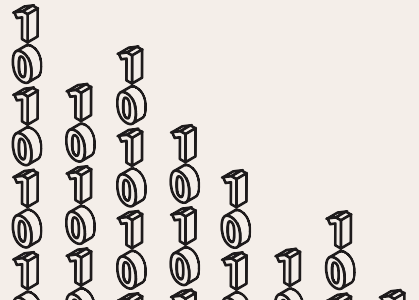
Design Specifications

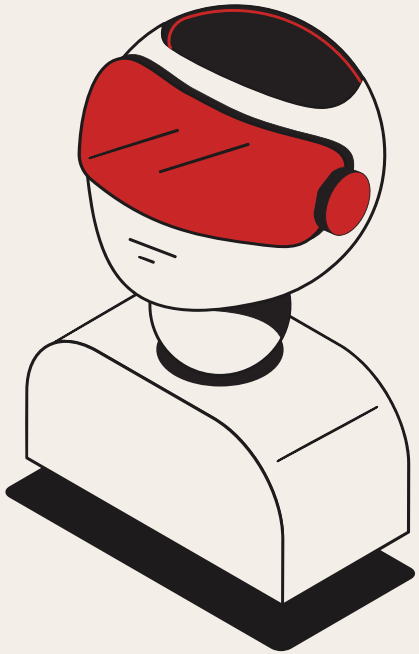
Requirements

- Send 8-bit data
- Choose an appropriate encryption and decryption method
- Utilize FPGA as a login management system
- Use of 7-segment display when inputting the passkey
- Implement push notification mechanism from FPGA to host

Constraints

- FPGA Processing Speed
- Resource Capabilities
- Complexity of Advanced Encryption Schemes
- Lab Conditions (underwater)





04
Block Diagrams

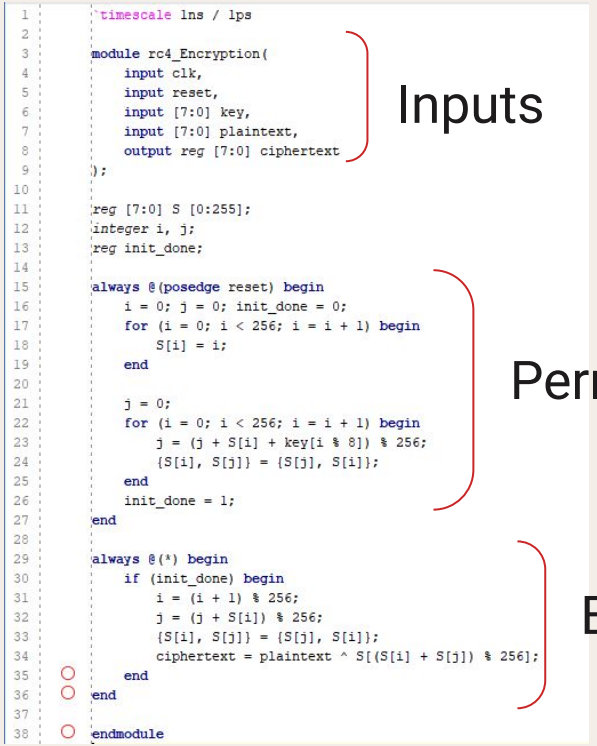
Visualize Our Design

Block Diagram

05
Code Snippets

Discussing Our Code

Code Snippet



```
`timescale 1ns / 1ps

module rc4_Encryption(
    input clk,
    input reset,
    input [7:0] key,
    input [7:0] plaintext,
    output reg [7:0] ciphertext
);

reg [7:0] S [0:255];
integer i, j;
reg init_done;

always @(posedge reset) begin
    i = 0; j = 0; init_done = 0;
    for (i = 0; i < 256; i = i + 1) begin
        S[i] = i;
    end

    j = 0;
    for (i = 0; i < 256; i = i + 1) begin
        j = (j + S[i] + key[i % 8]) % 256;
        {S[i], S[j]} = {S[j], S[i]};
    end
    init_done = 1;
end

always @(*) begin
    if (init_done) begin
        i = (i + 1) % 256;
        j = (j + S[i]) % 256;
        {S[i], S[j]} = {S[j], S[i]};
        ciphertext = plaintext ^ S[(S[i] + S[j]) % 256];
    end
end

endmodule
```

Inputs

Permutation array

Encryption

- Main Blocks
- Decryption module
- Symmetric vs Asymmetric Encryption Schemes
- Adding more bits

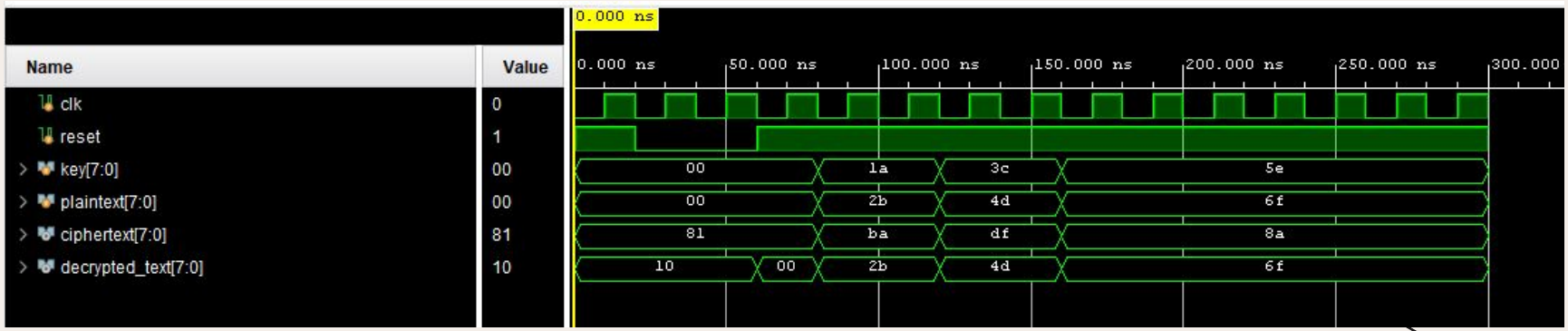
Conclusion

Successes, Failures, and Anticipations

Conclusions

Successes

We were able to send 4 different pieces of 8 bit data and decrypt it back again on FPGA, successfully.



Conclusions

Failure

Encountered issues with LED and SevenSeg display output while inputting passkey during tests on a personal FPGA board. Uncertain if failures stem from hardware differences or constraints file issues. Plan to test on school-provided FPGAs for clarity and resolution.

Anticipation

Future development includes adding a secure login password storage feature. We're also integrating an FPGA-based security alert system to send immediate push notifications to the host computer for potential data breaches, enhancing both security and user experience.

Thank you

Questions?