



**FUNDAMENTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT  
DEPARTMENT OF ELECTRICAL ENGINEERING  
UNIVERSITAS INDONESIA**

**VHDL Powered Cryptography with Caesar and Hill Cipher on FPGA**

**GROUP PA02**

<b>DEANDRO NAJWAN AHMAD S</b>	<b>(2306213174)</b>
<b>KHARISMA APRILIA</b>	<b>(2306223244)</b>
<b>REYHAN AHNAF DEANNOVA</b>	<b>(2306267100)</b>
<b>SHAFWAN HASYIM</b>	<b>(2306209113)</b>

## PREFACE

Puji dan syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga laporan proyek akhir Perancangan Sistem Digital yang berjudul **“VHDL Powered Cryptography with Caesar and Hill Cipher on FPGA”** dapat diselesaikan dengan baik. Ucapan terima kasih juga kami sampaikan kepada para asisten laboratorium, serta teman-teman yang telah berkontribusi dalam proses pengerjaan laporan proyek akhir ini.

Laporan ini disusun untuk melengkapi proyek akhir yang merupakan pemenuhan dari modul 10: Proyek Akhir Praktikum Perancangan Sistem Digital Tahun Ajaran 2022/2023. Laporan ini membahas tentang detail dari proyek yang telah kami buat. Laporan ini meliputi latar belakang, dekripsi, hasil dan analisis dari proyek yang telah kami buat.

Adapun karena keterbatasan pengetahuan maupun pengalaman kami, kami menyadari masih terdapat kekurangan dalam pengerjaan dan penyusunan laporan proyek akhir ini yang perlu diperbaiki. Oleh karena itu, kritik dan saran sangat kami harapkan sehingga dapat dijadikan bahan evaluasi bagi kami kedepannya. Kami juga memohon maaf apabila ada kesalahan dan kekurangan dalam penyusunan laporan ini

Depok, December 08, 2024

Group PA02

## TABLE OF CONTENTS

<b>CHAPTER 1: INTRODUCTION.....</b>	<b>3</b>
1.1 Background.....	3
1.2 Project Description.....	3
1.3 Objectives.....	4
1.4 Roles and Responsibilities.....	5
<b>CHAPTER 2: IMPLEMENTATION.....</b>	<b>6</b>
2.1 Equipment.....	6
2.2 Implementation.....	6
<b>CHAPTER 3: TESTING AND ANALYSIS.....</b>	<b>10</b>
3.1 Testing.....	10
3.2 Result.....	10
3.3 Analysis.....	12
<b>CHAPTER 4: CONCLUSION.....</b>	<b>14</b>
<b>REFERENCES.....</b>	<b>15</b>
<b>APPENDICES.....</b>	<b>16</b>
Appendix A: Project Schematic.....	16
Appendix B: Documentation.....	19

# CHAPTER 1

## INTRODUCTION

### 1.1 BACKGROUND

Keamanan data digital merupakan aspek yang sangat penting di era informasi saat ini. Dengan semakin pesatnya perkembangan teknologi, kebutuhan akan sistem enkripsi yang efektif dan efisien semakin mendesak. Enkripsi berfungsi untuk melindungi informasi penting dari akses yang tidak sah dan memastikan bahwa hanya pihak yang berwenang yang dapat mengakses data tersebut.

Namun, dalam implementasinya, proses enkripsi dan dekripsi seringkali memerlukan waktu yang lama dan beban komputasi yang tinggi, terutama pada sistem yang menggunakan perangkat keras terbatas. Oleh karena itu, diperlukan sistem yang mampu mengoptimalkan proses enkripsi dengan menggunakan metode yang lebih efisien, namun tetap menjaga tingkat keamanannya.

Metode enkripsi klasik, seperti Caesar Cipher dan Hill Cipher, adalah dua contoh algoritma yang banyak digunakan dalam pengamanan data. Caesar Cipher merupakan algoritma yang mudah dipahami dan diterapkan, sementara Hill Cipher, meskipun lebih kompleks, menawarkan tingkat keamanan yang lebih tinggi dengan menggunakan matriks kunci untuk enkripsi.

Untuk meningkatkan kecepatan dan efisiensi enkripsi ini, implementasi menggunakan perangkat keras (hardware) menjadi pilihan yang tepat. Salah satu perangkat keras yang dapat digunakan adalah FPGA (Field-Programmable Gate Array), yang memungkinkan desain dan implementasi algoritma enkripsi dalam bentuk yang lebih cepat dan efisien.

### 1.2 PROJECT DESCRIPTION

Proyek ini bertujuan untuk mengembangkan sistem enkripsi dan dekripsi menggunakan dua metode klasik populer, yaitu **Caesar Cipher dan Hill Cipher**. Sistem ini memproses teks yang diberikan melalui dua tahap enkripsi atau dekripsi, tergantung pada mode yang dipilih. Pada mode pertama, sistem akan menggunakan Caesar Cipher, yang

mengenkripsi teks dengan cara menggeser setiap karakter berdasarkan jumlah yang ditentukan. Hasil enkripsi dari Caesar Cipher kemudian akan digunakan sebagai input untuk tahap kedua menggunakan Hill Cipher. Hill Cipher adalah metode enkripsi yang lebih kuat dengan menggunakan matriks kunci untuk mengenkripsi data yang telah ditentukan dalam code. Proses enkripsi ini memberikan dua lapisan pengamanan yang lebih kuat terhadap data yang ingin dilindungi.

Sistem ini bekerja dengan cara membaca teks dari file input.txt lalu melakukan enkripsi atau dekripsi sesuai mode yang ditentukan, dan menyimpan hasilnya pada dua file output yang berbeda: 'encryptOutput.txt' untuk hasil enkripsi dan 'decryptOutput.txt' untuk hasil dekripsi. Setelah tahap pertama menggunakan Caesar Cipher selesai, hasilnya disimpan di file phase.txt untuk digunakan sebagai input pada tahap kedua (Hill Cipher). Pada proses dekripsi, sistem akan menjalankan Hill Cipher untuk mendekripsi data yang dienkripsi sebelumnya, kemudian menggunakan Caesar Cipher untuk proses dekripsi akhir, sehingga teks asli dapat dipulihkan.

Dengan menggunakan file I/O, proyek ini memudahkan pengguna untuk memberikan data dan mendapatkan hasil tanpa harus berinteraksi langsung dengan kode program. Selain itu, sistem ini dirancang untuk mengelola setiap karakter dalam teks input secara terstruktur, memastikan bahwa setiap proses enkripsi atau dekripsi dijalankan dengan benar dan hasilnya tersimpan dengan baik di file yang sesuai.

### **1.3 OBJECTIVES**

Tujuan dari proyek ini:

1. Sebagai pemenuhan nilai dalam Praktikum Perancangan Sistem Digital
2. Mengimplementasikan Pemrograman VHDL
3. Merancang dan Mengimplementasikan Sistem Enkripsi dan Dekripsi Ganda Menggunakan Caesar Cipher dan Hill Cipher
4. Meningkatkan Keamanan Data dengan Implementasi Perangkat Keras Berbasis FPGA

#### 1.4 ROLES AND RESPONSIBILITIES

Roles dan tanggung jawab yang diberikan kepada anggota kelompok adalah sebagai berikut:

Roles	Person
Membuat kode dan menyusun laporan	Shafwan Hasyim
Membuat kode dan menyusun laporan	Kharisma Aprilia
Membuat kode, Menyusun Github dan menyusun laporan	Deandro Najwan Ahmad S
Menyusun laporan dan membuat PPT serta dokumentasi	Reyhan Ahnaf D

Table 1. Roles and Responsibilities

## **CHAPTER 2**

### **IMPLEMENTATION**

#### **2.1 EQUIPMENT**

tools yang kami gunakan dalam mengerjakan proyek ini, yaitu:

- Visual Studio Code
- ModelSim
- Quartus
- Github

#### **2.2 IMPLEMENTATION**

##### **1. Dataflow**

Pada bagian Dataflow, sistem diorganisasikan dengan cara yang memungkinkan data mengalir tanpa gangguan dari satu komponen ke komponen lainnya. Modul ini berfungsi untuk menangani aliran data dalam sistem enkripsi menggunakan algoritma Caesar Cipher. Setelah teks input diterima, data langsung diproses dalam rangkaian secara paralel, di mana setiap karakter teks yang dienkripsi langsung dihitung hasil pergeserannya tanpa menunggu input lainnya. Pendekatan ini memungkinkan penggunaan paralelisme dalam perangkat keras FPGA, yang mempercepat waktu proses enkripsi, terutama untuk teks yang lebih panjang. Dengan memanfaatkan aliran data langsung, modul ini meningkatkan efisiensi sistem secara keseluruhan, meminimalkan keterlambatan dan meningkatkan throughput proses enkripsi.

##### **2. Behavioral**

Modul Behavioral digunakan untuk mendeskripsikan fungsi atau perilaku dari sistem pada tingkat yang lebih abstrak dibandingkan dengan merinci struktur perangkat keras secara detail. Dalam proyek ini, modul ini digunakan untuk menggambarkan proses enkripsi menggunakan Caesar Cipher dan Hill Cipher, dengan fokus pada alur proses fungsionalnya. Sebagai contoh, pada Caesar Cipher, enkripsi

dilakukan berdasarkan pergeseran karakter teks sesuai dengan parameter yang ditentukan. Keuntungan dari pendekatan ini adalah memberikan fleksibilitas dalam merancang logika sistem tanpa terikat oleh detail implementasi perangkat keras. Modul ini memungkinkan pengujian dan perancangan proses enkripsi secara teoritis sebelum implementasi fisiknya.

### **3. Testbench**

Testbench adalah kode yang digunakan untuk menguji dan memverifikasi fungsionalitas entitas yang telah dirancang dalam sistem. Pada proyek ini, testbench digunakan untuk mensimulasikan berbagai input yang akan dienkripsi oleh Caesar Cipher dan Hill Cipher, serta memeriksa apakah output yang dihasilkan sesuai dengan yang diharapkan. Fungsi utama testbench adalah untuk mensimulasikan input berupa teks, memverifikasi hasil output setelah enkripsi, dan memastikan bahwa hasil enkripsi tersebut sesuai dengan ekspektasi. Dengan menggunakan testbench, pengujian dapat dilakukan lebih cepat, memungkinkan verifikasi yang lebih efisien sebelum implementasi sistem pada perangkat keras FPGA.

### **4. Structural**

Desain Structural dalam VHDL memungkinkan kita untuk menggambarkan sistem menggunakan pendekatan modular. Setiap komponen, seperti Caesar Cipher dan Hill Cipher, dijelaskan dalam entitas terpisah dan kemudian digabungkan dalam entitas top-level untuk membentuk sistem yang lebih besar. Pendekatan ini mengorganisir sistem menjadi bagian-bagian fungsional yang terpisah, membuat pengujian dan pemeliharaan lebih mudah. Dengan struktur modular ini, setiap komponen dapat diuji secara terpisah sebelum digabungkan dalam tahap akhir, membantu meminimalkan kesalahan dan mempercepat proses debugging. Desain ini memberikan fleksibilitas dalam pengembangan dan memungkinkan perubahan atau peningkatan pada satu komponen tanpa mempengaruhi sistem secara keseluruhan.

### **5. Looping**

Konstruksi Looping digunakan untuk melakukan perulangan pada karakter atau blok data yang akan diproses dalam enkripsi dan dekripsi. Dalam proyek ini, perulangan digunakan pada beberapa bagian proses, misalnya dengan for-loop untuk mengenkripsi setiap karakter dalam teks menggunakan Caesar Cipher, sementara



while-loop digunakan untuk mengolah matriks kunci dalam Hill Cipher. Looping memungkinkan pemrosesan data yang memerlukan pengulangan tanpa menulis kode yang redundan, meningkatkan efisiensi sistem. Dengan penggunaan perulangan ini, proses enkripsi atau dekripsi pada teks panjang atau matriks besar dapat dilakukan secara otomatis dan efisien, mengurangi kebutuhan untuk kode yang berulang.

## 6. Function

Modul Function digunakan untuk memisahkan bagian-bagian kode yang berulang dan mengisolasi operasi tertentu ke dalam fungsi terpisah. Pada proyek ini, fungsi digunakan untuk operasi seperti pergeseran karakter dalam Caesar Cipher dan perhitungan matriks dalam Hill Cipher. Penggunaan fungsi memungkinkan operasi yang sering digunakan untuk dipanggil kembali tanpa menulis kode yang sama berulang kali, meningkatkan modularitas kode, serta membuat pemeliharaan dan pengembangan sistem lebih mudah dan efisien. Pendekatan ini juga mengurangi duplikasi kode, yang tidak hanya membuat sistem lebih bersih tetapi juga mempermudah debugging dan pengembangan lebih lanjut.

## 7. Finite State Machine (FSM)

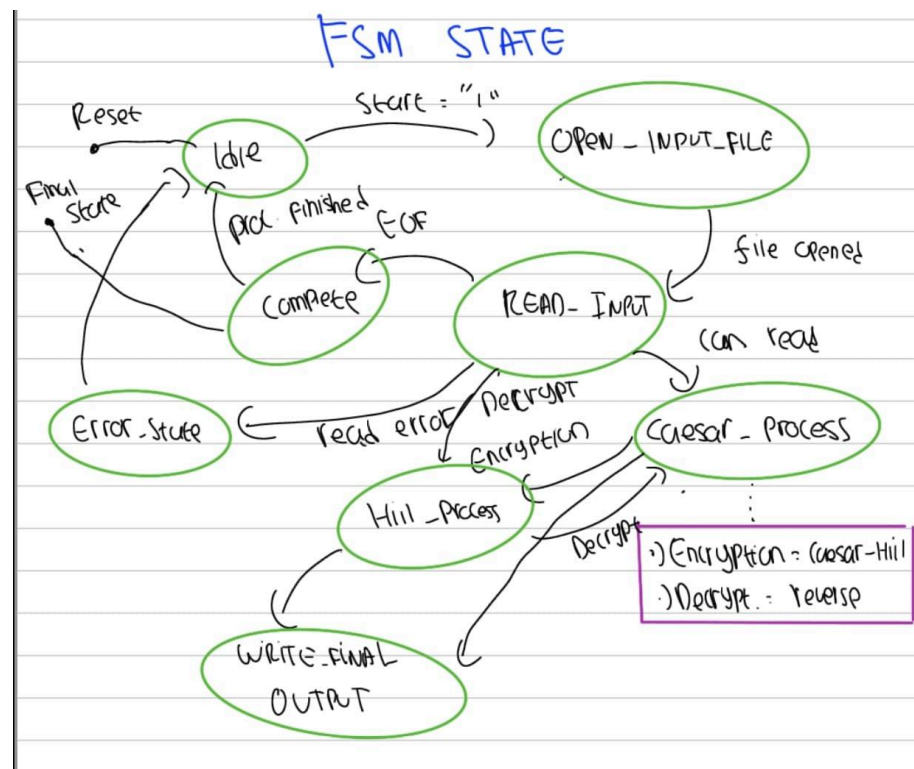


Fig 1. FSM Dual Cipher

Finite State Machine (FSM) adalah model matematika yang digunakan untuk merepresentasikan sistem berbasis status. FSM terdiri dari sejumlah status yang telah terdefinisi, transisi antara status tersebut, dan kondisi yang mengatur perpindahan antar status. Pada sistem ini, FSM digunakan untuk mengontrol alur proses enkripsi dan dekripsi dengan dua algoritma cipher, yaitu Caesar Cipher dan Hill Cipher, yang bekerja pada file input dan menghasilkan file output sesuai dengan mode operasi yang telah ditentukan.

Sistem dimulai dari status IDLE, di mana ia menunggu sinyal start untuk memulai. Berdasarkan nilai sinyal mode, sistem menentukan apakah operasi yang akan dilakukan adalah enkripsi (mode = '0') atau dekripsi (mode = '1'). Setelah mode operasi ditentukan, sistem beralih ke status OPEN\_INPUT\_FILE, di mana file input dibuka untuk dibaca, dan file output disiapkan untuk menulis hasilnya. Jika file berhasil dibuka, sistem melanjutkan ke status READ\_INPUT, di mana data dibaca satu per satu karakter dari file input.

Data yang telah dibaca diproses melalui beberapa status yang mengimplementasikan algoritma cipher. Dalam proses enkripsi, karakter input diproses terlebih dahulu oleh Caesar Cipher di status CAESAR\_PROCESS, menghasilkan output sementara yang kemudian diproses lebih lanjut oleh Hill Cipher di status HILL\_PROCESS. Hasil akhir dari Hill Cipher ditulis ke file output sementara di status WRITE\_FINAL\_OUTPUT sebelum sistem kembali membaca karakter berikutnya. Sebaliknya, dalam proses dekripsi, alur proses ini dimulai dari Hill Cipher, dilanjutkan oleh Caesar Cipher, dengan hasil akhirnya langsung ditulis ke file output.

Setelah semua karakter dalam file selesai diproses, sistem memasuki status COMPLETE, di mana file input dan output ditutup dengan aman. Pada titik ini, sinyal done diatur aktif ('1') untuk menunjukkan bahwa operasi telah selesai. Jika terjadi kesalahan selama proses, sistem akan beralih ke status ERROR\_STATE, memberikan indikasi melalui sinyal error dan kembali ke status IDLE untuk siap menerima operasi berikutnya. Dengan FSM, sistem ini mampu mengelola alur kerja yang kompleks dengan struktur yang sistematis dan modular, memastikan setiap langkah dilakukan secara tertib dan efisien.

## CHAPTER 3

### TESTING AND ANALYSIS

#### 3.1 TESTING

Untuk memastikan bahwa sistem enkripsi dan dekripsi berjalan dengan baik, dilakukan pengujian menggunakan menggunakan Testbench dan file I/O yang telah dibuat untuk memastikan seluruh proses berjalan dengan baik dan sesuai dengan yang diharapkan. Proses testing ini dimulai dengan membaca file input.txt, yang berisi teks “hello”, sebagai data masukan untuk sistem. Selanjutnya, sistem ini akan menjalankan proses enkripsi menggunakan Caesar Cipher dan Hill Cipher, dan hasil dari enkripsi pertama (Caesar Cipher) akan disimpan di file sementara phase1.txt. Setelah itu, proses enkripsi kedua menggunakan Hill Cipher dilakukan, dan hasil akhirnya disimpan di file 'encryptOutput.txt'.

Setelah itu, untuk memverifikasi proses dekripsi, file 'encryptOutput.txt', yang berisi data terenkripsi, akan dibaca, dan proses dekripsi dilakukan dengan menggunakan Hill Cipher dan Caesar Cipher dalam urutan terbalik. Hasil dekripsi ini kemudian disimpan di 'decryptOutput.txt'. Dalam testing ini, kami memeriksa apakah hasil di file 'decryptOutput.txt' sesuai dengan input awal yang terdapat pada 'input.txt', yaitu teks “hello”.

Tujuan dari testing ini adalah untuk memastikan bahwa kedua proses enkripsi dan dekripsi berjalan dengan benar, dan data yang terenkripsi dapat dikembalikan menjadi bentuk aslinya setelah melalui kedua tahap proses tersebut. Harapannya, hasil yang ditemukan di 'decryptOutput.txt' setelah menjalankan dekripsi akan sama dengan input yang diberikan di 'input.txt' yang menunjukkan bahwa sistem enkripsi dan dekripsi berfungsi sebagaimana mestinya dan tanpa adanya kesalahan dalam pengolahan data.

#### 3.2 RESULT

Secara keseluruhan, sistem enkripsi dan dekripsi yang kami implementasikan telah berjalan sesuai rencana. Proses enkripsi menghasilkan output yang sesuai dengan harapan, yaitu "mlwx" ketika teks input "hello" diproses menggunakan kombinasi Caesar Cipher dan Hill Cipher. Hasil ini menunjukkan bahwa tahap enkripsi telah berhasil dilakukan dengan baik, dan data berhasil disalin ke file 'encryptOutput.txt'.

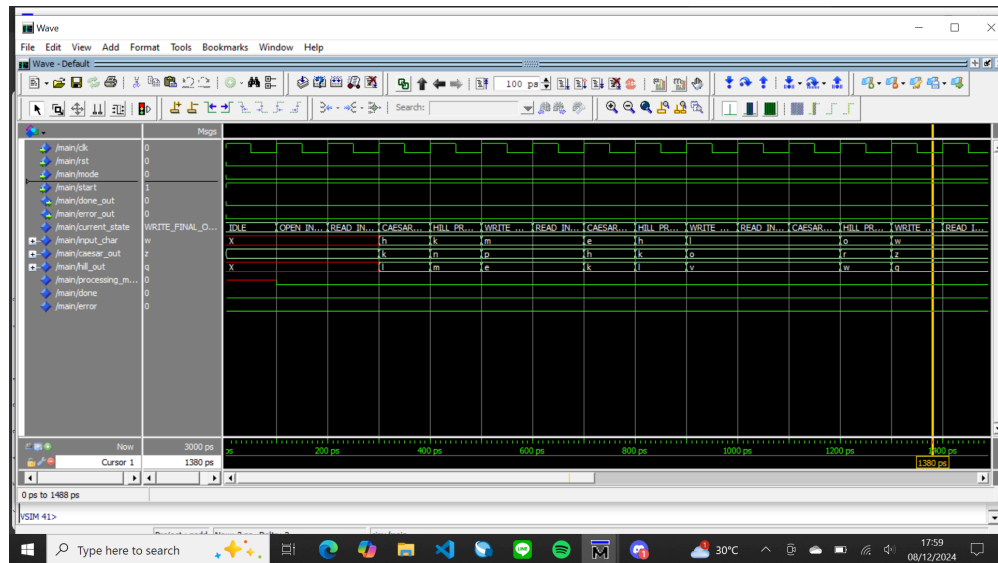


Fig.2 Testing Result Modelsim

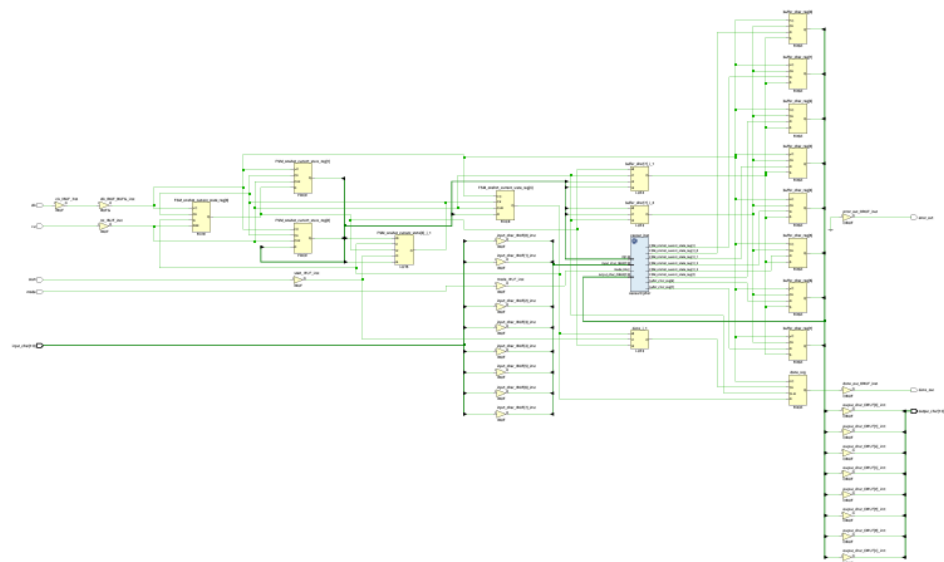


Fig 3. Synthesized Result

```

# ** Note: Reading Encrypted Output
#   Time: 255 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: m
#   Time: 255 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: l
#   Time: 255 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: w
#   Time: 255 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: w
#   Time: 255 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: x
#   Time: 255 ns   Iteration: 0   Instance: /maintb
# ** Note: Output matches expected data
#   Time: 255 ns   Iteration: 0   Instance: /maintb
# ** Note: SKENARIO DEKRIPSI
#   Time: 265 ns   Iteration: 0   Instance: /maintb
# ** Note: Reading Decrypted Output
#   Time: 455 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: e
#   Time: 455 ns   Iteration: 0   Instance: /maintb

# ** Note: Output Char: b
#   Time: 455 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: i
#   Time: 455 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: i
#   Time: 455 ns   Iteration: 0   Instance: /maintb
# ** Note: Output Char: l
#   Time: 455 ns   Iteration: 0   Instance: /maintb
# ** Note: Got:      ebiil
#   Time: 455 ns   Iteration: 0   Instance: /maintb

```

Fig 4. Transcript Result

Namun, pada tahap dekripsi, meskipun mekanisme sistem telah mengikuti alur yang benar, output yang dihasilkan adalah "ebiil" tidak seperti yang diharapkan. Kami telah memastikan bahwa semua langkah dalam Finite State Machine (FSM) dan proses enkripsi-dekripsi dua tahap telah diimplementasikan dengan benar. Meskipun demikian, berdasarkan hasil pengujian, alur kode dan fungsionalitas dasar sistem sudah sesuai. Kami berencana untuk melakukan pengecekan lebih lanjut untuk mengidentifikasi penyebab ketidaksesuaian ini dan melakukan perbaikan.

### 3.3 ANALYSIS

Proses enkripsi dimulai dengan membaca teks "hello" dari file `input.txt`. Pada tahap pertama, Caesar Cipher mengenkripsi teks dengan cara menggeser setiap karakter sebanyak tiga posisi dalam alfabet, menghasilkan teks "khood". Hasil dari enkripsi Caesar Cipher ini kemudian diproses menggunakan Hill Cipher untuk tahap enkripsi kedua. Hill Cipher mengenkripsi teks "khood" menggunakan matriks kunci, menghasilkan teks "mlwx" yang disimpan dalam file 'encryptOutput.txt'.

Selanjutnya, proses dekripsi dimulai dengan membaca file 'encryptOutput.txt' yang berisi "mlwx". Teks ini pertama-tama didekripsi menggunakan Hill Cipher, kemudian hasil dekripsi Hill Cipher diproses kembali dengan Caesar Cipher untuk mendapatkan kembali teks asli. Harapannya, setelah kedua proses dekripsi, teks yang dihasilkan adalah "hello". Namun, hasil dekripsi yang kami dapatkan adalah "ebiil" bukan "hello". Meskipun mekanisme dan alur proses sudah benar, ketidaksesuaian hasil ini menunjukkan adanya kemungkinan kesalahan dalam implementasi pada tahap dekripsi.

## **CHAPTER 4**

### **CONCLUSION**

Proyek ini berhasil mengimplementasikan sistem enkripsi dan dekripsi ganda menggunakan Caesar Cipher dan Hill Cipher pada FPGA. Proses enkripsi dilakukan dalam dua tahap: pertama menggunakan Caesar Cipher untuk mengenkripsi teks berdasarkan pergeseran karakter, dan kedua menggunakan Hill Cipher yang mengenkripsi hasil dari Caesar Cipher dengan matriks kunci. Sistem ini bekerja dengan baik pada proses enkripsi, menghasilkan output yang sesuai dengan yang diharapkan.

Namun, meskipun mekanisme dan alur sistem sudah diimplementasikan dengan benar, hasil dekripsi tidak sepenuhnya sesuai dengan yang diinginkan. Setelah proses dekripsi menggunakan Hill Cipher dan Caesar Cipher, hasil yang diperoleh adalah "ebiil" alih-alih "hello" yang seharusnya. Meskipun demikian, alur Finite State Machine (FSM) dan proses enkripsi-dekripsi dua tahap telah diterapkan dengan benar. Oleh karena itu, meskipun ada ketidaksesuaian dalam hasil dekripsi, kami yakin bahwa sebagian besar kode dan sistem bekerja dengan baik, dan akan melakukan analisis lebih lanjut untuk mencari tahu penyebab kesalahan ini. Meskipun begitu, sistem ini menunjukkan potensi besar dalam menerapkan enkripsi yang lebih kuat melalui dua tahap pengolahan data.

## REFERENCES

- [1] “Final Project Digital ... | Digilab UI,” Digilabdte.com, 2024.  
<https://learn.digilabdte.com/books/digital-system-design/page/final-project-digital-system-design> (accessed Dec. 08, 2024).
- [2] GeeksforGeeks, “Caesar Cipher in Cryptography,” GeeksforGeeks, Jun. 02, 2016.  
<https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/>
- [3] D. Anand, “Hill Cipher - GeeksforGeeks,” GeeksforGeeks, Jul. 21, 2021.  
<https://www.geeksforgeeks.org/hill-cipher/>
- [4] A. Hidayat and Tuty Alawiyah, “Enkripsi dan Dekripsi Teks menggunakan Algoritma Hill Cipher dengan Kunci Matriks Persegi Panjang,” *Jurnal Matematika Integratif*, vol. 9, no. 1, pp. 39–39, Apr. 2013, doi: <https://doi.org/10.24198/jmi.v9i1.10196>.



## APPENDICES

### Appendix A: Project Schematic

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use STD.TEXTIO.ALL;

entity Main is
    Port (
        clk      : in STD_LOGIC;
        rst      : in STD_LOGIC;
        mode      : in STD_LOGIC; -- '0' for encrypt, '1' for decrypt
        start     : in STD_LOGIC;
        done_out  : out STD_LOGIC;
        error_out : out STD_LOGIC
    );
end Main;

architecture Behavioral of Main is
    -- State definitions
    type state_type is (
        IDLE,
        OPEN_INPUT_FILE,
        READ_INPUT,
        CAESAR_PROCESS,
        HILL_PROCESS,
        WRITE_PHASE1,
        WRITE_FINAL_OUTPUT,
        COMPLETE,
        ERROR_STATE
    );

    signal current_state : state_type := IDLE;

    -- Component declarations
    component caesarCipher
        port (
            input      : in  std_logic_vector(7 downto 0);
            mode       : in  std_logic;
            shift_char : in  integer range 0 to 25;
            cipher     : out std_logic_vector(7 downto 0)
        );
    end component;

    component hillCipher
        port (
            input  : in STD_LOGIC_VECTOR(7 downto 0);
            mode   : in STD_LOGIC;
            output : out STD_LOGIC_VECTOR(7 downto 0)
        );
    end component;

    -- Signals
    signal input_char : std_logic_vector(7 downto 0);
    signal caesar_out, hill_out : std_logic_vector(7 downto 0);
    signal processing_mode : std_logic;
    signal done : std_logic := '0';
    signal error : std_logic := '0';
    constant CAESAR_SHIFT : integer := 3;

begin
    -- Component instantiations
    caesar_inst : caesarCipher port map (
```

```

        input => input_char,
        mode => processing_mode,
        shift_char => CAESAR_SHIFT,
        cipher => caesar_out
    );

hill_inst : hillCipher port map (
    input => input_char,
    mode => processing_mode,
    output => hill_out
);

-- Output assignments
done_out <= done;
error_out <= error;

-- Main state machine
state_machine: process(clk, rst)
    file input_file : text;
    file phase1_file : text;
    file final_output_file : text;

    variable line_in : line;
    variable line_out : line;
    variable input_char_var : character;
    variable is_file_open : boolean := false;
begin
    if rst = '1' then
        current_state <= IDLE;
        done <= '0';
        error <= '0';
        processing_mode <= '0';
        is_file_open := false;
    elsif rising_edge(clk) then
        case current_state is
            when IDLE =>
                if start = '1' then
                    processing_mode <= mode;
                    current_state <= OPEN_INPUT_FILE;
                    done <= '0';
                    error <= '0';
                end if;

                when OPEN_INPUT_FILE =>
                    file_open(input_file, "input.txt", read_mode);

                    if processing_mode = '0' then
                        file_open(phase1_file, "phase1.txt", write_mode);
                        file_open(final_output_file, "encryptOutput.txt", write_mode);
                    else
                        file_open(phase1_file, "phase1.txt", write_mode);
                        file_open(final_output_file, "decryptOutput.txt", write_mode);
                    end if;

                    is_file_open := true;
                    current_state <= READ_INPUT;

                when READ_INPUT =>
                    if not endfile(input_file) then
                        readline(input_file, line_in);
                        read(line_in, input_char_var);
                        input_char <= std_logic_vector(to_unsigned(character'pos(input_char_var), 8));

                        IF processing_mode = '0' THEN
                            current_state <= CAESAR_PROCESS;
                        ELSE
                            current_state <= HILL_PROCESS;
                        END IF;
                    else
                        current_state <= COMPLETE;
                    end if;
                end case;
            end if;
        end case;
    end if;
end process;

```

```

        end if;

        when HILL_PROCESS =>
            if processing_mode = '1' then -- Dekripsi
                input_char <= hill_out; -- Dekripsi Hill Cipher dulu
                write(line_out, character'val(to_integer(unsigned(hill_out))));
                writeline(phase1_file, line_out);
                current_state <= CAESAR_PROCESS;
            else -- Enkripsi (tetap sama)
                input_char <= hill_out;
                write(line_out, character'val(to_integer(unsigned(hill_out))));
                writeline(final_output_file, line_out);
                current_state <= WRITE_FINAL_OUTPUT;
            end if;

        when CAESAR_PROCESS =>
            if processing_mode = '1' then -- Dekripsi
                input_char <= caesar_out; -- Kemudian dekripsi Caesar Cipher
                write(line_out, character'val(to_integer(unsigned(caesar_out))));
                writeline(final_output_file, line_out);
                current_state <= WRITE_FINAL_OUTPUT;
            else -- Enkripsi (tetap sama)
                input_char <= caesar_out;
                write(line_out, character'val(to_integer(unsigned(caesar_out))));
                writeline(phase1_file, line_out);
                current_state <= HILL_PROCESS;
            end if;

        when WRITE_FINAL_OUTPUT =>
            current_state <= READ_INPUT;

        when COMPLETE =>
            if is_file_open then
                file_close(input_file);
                file_close(phase1_file);
                file_close(final_output_file);
                is_file_open := false;
            end if;

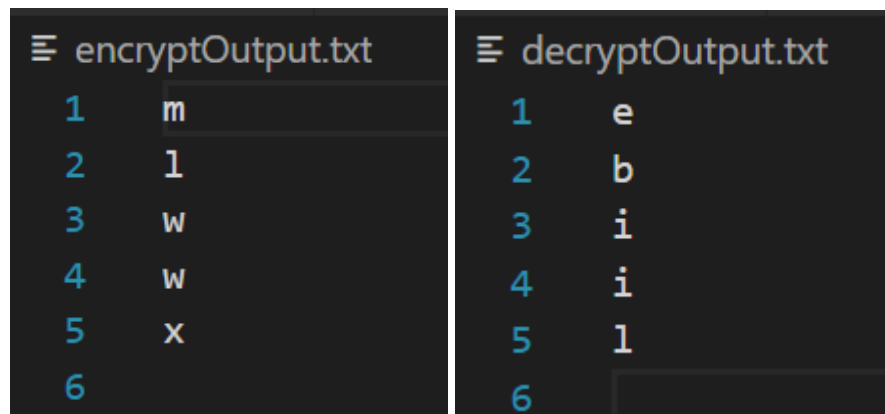
            done <= '1';
            current_state <= IDLE;

        when ERROR_STATE =>
            error <= '1';
            current_state <= IDLE;

        when others =>
            current_state <= ERROR_STATE;
        end case;
    end if;
end process state_machine;
end Behavioral;

```

## Appendix B: Documentation



The image shows two terminal windows side-by-side. The left window is titled 'encryptOutput.txt' and contains a list of six items: 1 m, 2 l, 3 w, 4 w, 5 x, and 6. The right window is titled 'decryptOutput.txt' and contains a list of six items: 1 e, 2 b, 3 i, 4 i, 5 l, and 6. The numbers 1 through 6 are in blue, and the letters are in white. The background of the terminals is dark.

encryptOutput.txt	decryptOutput.txt
1 m	1 e
2 l	2 b
3 w	3 i
4 w	4 i
5 x	5 l
6	6

Fig 5. Text I/O Result