



# VHDL POWERED CRYPTOGRAPHY WITH CAESAR AND HILL CIPHER ON FPGA



# ANGGOTA KELOMPOK:



Reyhan Ahnaf Deannova  
2306267100



Deandro Najwan Ahmad S.  
2306213174



Kharisma Aprillia  
2306223244

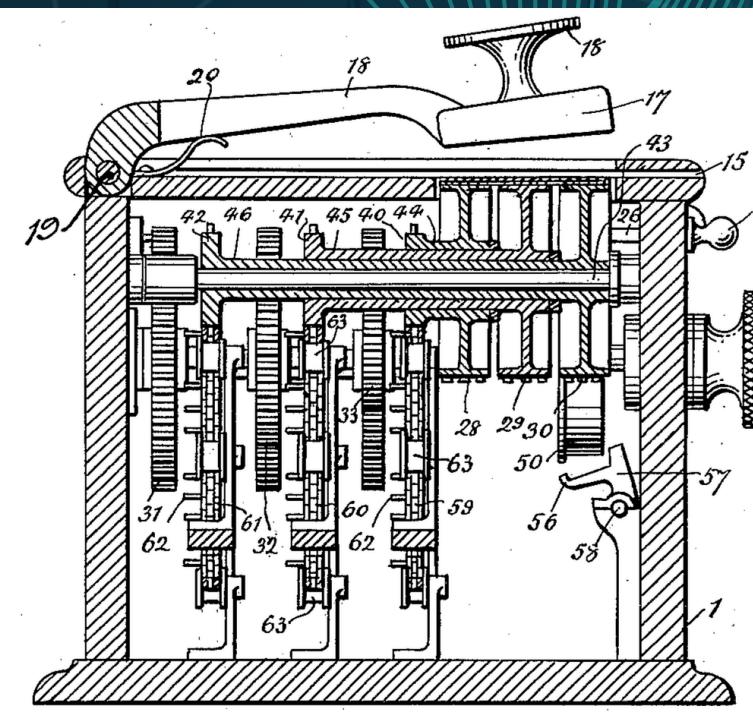


Shafwan Hasyim  
2306209113



# LATAR BELAKANG

Keamanan data digital menjadi aspek krusial di era informasi modern. Sistem enkripsi yang efektif dibutuhkan untuk melindungi informasi penting dari akses tidak sah dan memastikan hanya pihak berwenang yang dapat mengaksesnya. Namun, proses enkripsi dan dekripsi sering memerlukan waktu lama dan beban komputasi tinggi, terutama pada perangkat keras terbatas. Oleh karena itu, diperlukan metode enkripsi yang lebih efisien tanpa mengorbankan tingkat keamanannya.



Algoritma klasik seperti Caesar Cipher dan Hill Cipher sering digunakan untuk pengamanan data. Caesar Cipher sederhana dan mudah diterapkan, sedangkan Hill Cipher lebih kompleks namun menawarkan keamanan lebih tinggi dengan matriks kunci. Untuk meningkatkan kecepatan dan efisiensi enkripsi, FPGA (Field-Programmable Gate Array) dapat dimanfaatkan untuk mengimplementasikan algoritma ini secara lebih cepat dan efisien dalam perangkat keras.



# PROJECT DESCRIPTION



Proyek ini bertujuan untuk mengembangkan sebuah sistem enkripsi dan dekripsi menggunakan dua metode klasik: Caesar Cipher dan Hill Cipher. Dalam sistem ini, proses enkripsi dimulai dengan penggunaan Caesar Cipher yang menggeser setiap huruf dari plain text sesuai dengan jumlah shift yang diinputkan saat simulasi. Setelah plain text dienkripsi dengan Caesar Cipher, hasilnya akan menjadi input untuk proses enkripsi selanjutnya menggunakan Hill Cipher.

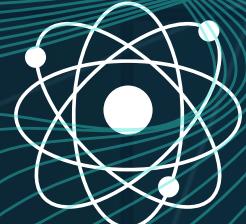
Hill Cipher, yang menggunakan matriks kunci untuk mengenkripsi data, akan memberikan lapisan keamanan tambahan pada hasil enkripsi pertama. Matriks kunci untuk Hill Cipher ditentukan dalam kode, dan hasil enkripsi kedua akan disimpan dalam file terpisah. Setelah proses enkripsi selesai, dekripsi akan dilakukan dengan urutan yang terbalik: pertama menggunakan Hill Cipher untuk mendekripsi hasil enkripsi Hill Cipher, diikuti dengan dekripsi menggunakan Caesar Cipher untuk mengembalikan hasil enkripsi ke bentuk plain text asli.



# OBJECTIVE



Sebagai pemenuhan nilai dalam Praktikum Perancangan Sistem Digital



Mengimplementasikan Pemrograman VHDL



Merancang dan Mengimplementasikan Sistem Enkripsi dan Dekripsi Ganda Menggunakan Caesar Cipher dan Hill Cipher



Meningkatkan Keamanan Data dengan Implementasi Perangkat Keras Berbasis FPGA



# TOOLS



VISUAL STUDIO CODE



MODEL SIM



GITHUB



QUARTUS



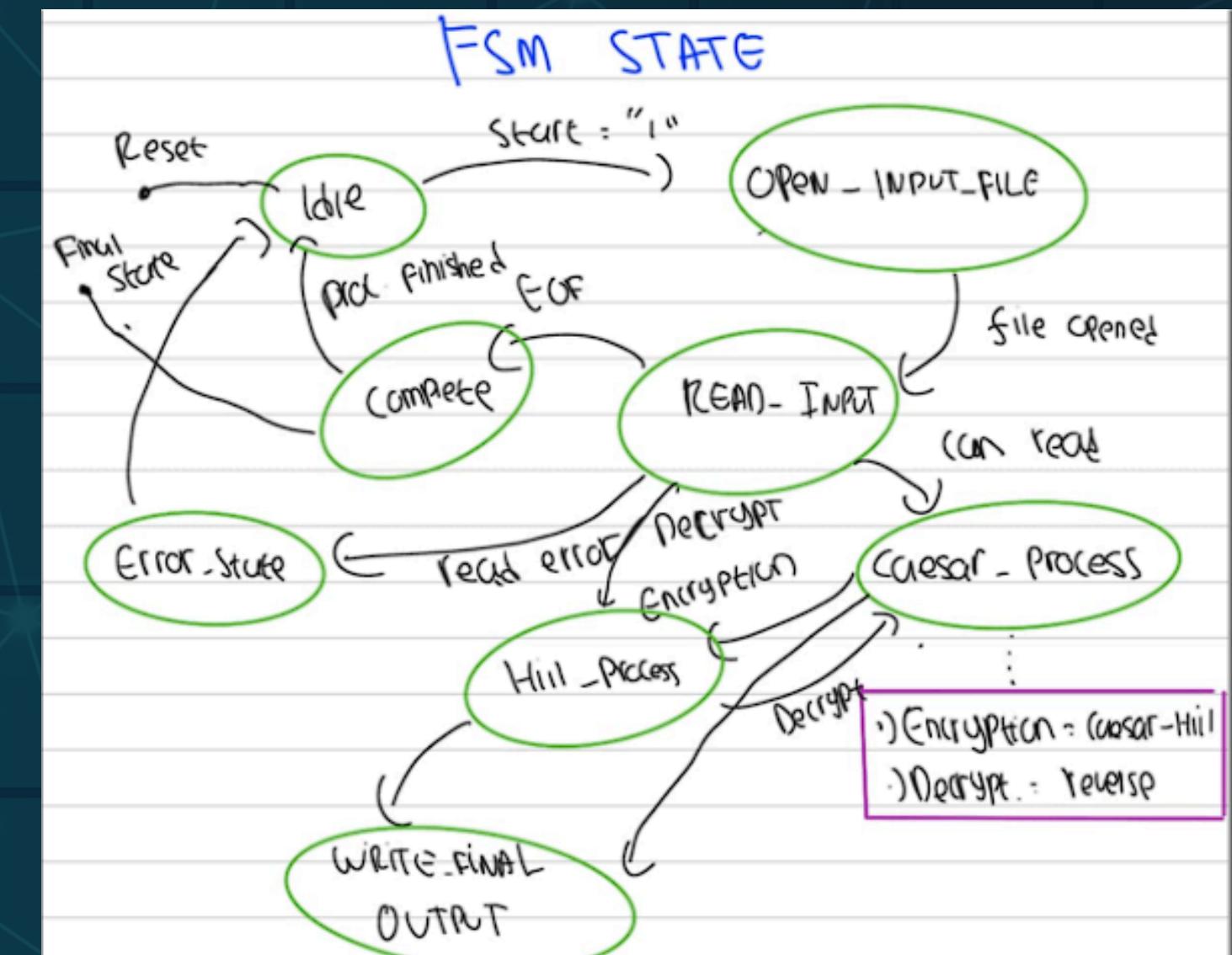
# FSM DIAGRAM

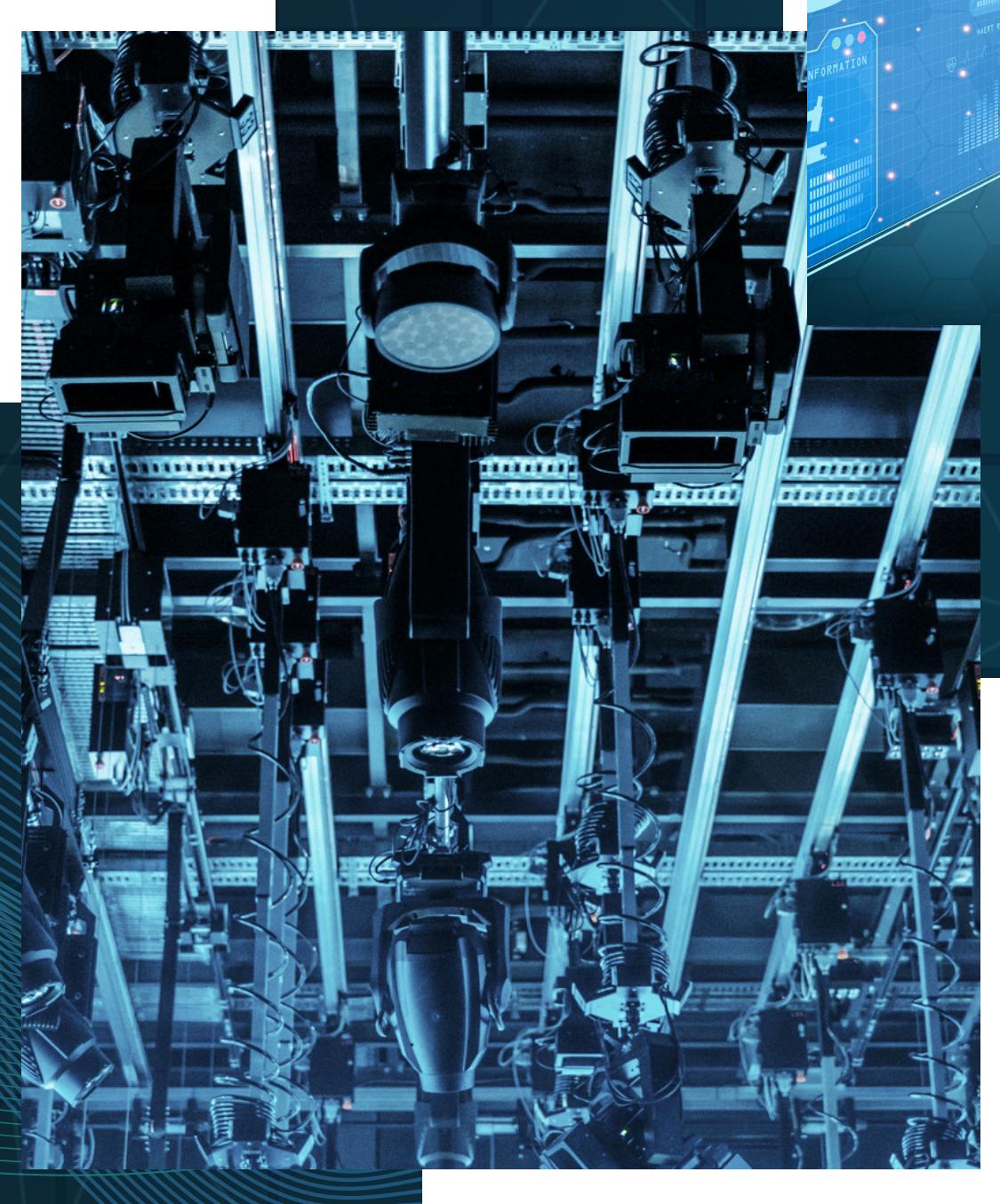
Sistem ini menggunakan FSM untuk mengontrol proses enkripsi dan dekripsi dengan Caesar Cipher dan Hill Cipher, bekerja pada file input dan menghasilkan file output berdasarkan mode operasi. Sistem dimulai dari status IDLE, menunggu sinyal start. Mode operasi menentukan enkripsi (mode = '0') atau dekripsi (mode = '1').

Setelah mode ditentukan, sistem beralih ke status OPEN\_INPUT\_FILE untuk membuka file input dan menyiapkan file output. Jika berhasil, sistem masuk ke status READ\_INPUT untuk membaca data karakter demi karakter.

Dalam enkripsi, data diproses melalui CAESAR\_PROCESS lalu HILL\_PROCESS, sebelum ditulis ke file output pada WRITE\_FINAL\_OUTPUT. Untuk dekripsi, proses dimulai dengan HILL\_PROCESS, diikuti CAESAR\_PROCESS, lalu hasilnya langsung ditulis ke output.

Setelah semua karakter diproses, sistem masuk status COMPLETE untuk menutup file dengan aman dan mengaktifkan sinyal done. Jika terjadi kesalahan, sistem beralih ke ERROR\_STATE, mengaktifkan sinyal error, dan kembali ke IDLE. FSM memastikan alur kerja tertib dan modular.





# IMPLEMENTATION



# MAIN CODE

Main code berisi sebuah entitas bernama Main yang mengimplementasikan mesin keadaan untuk proses kriptografi. Entitas ini memiliki port input seperti clk, rst, mode, dan start untuk mengontrol operasi, serta output done\_out dan error\_out untuk menunjukkan status proses. Mesin keadaan didefinisikan dalam tipe state\_type dengan beberapa keadaan seperti IDLE, READ\_INPUT, CAESAR\_PROCESS, HILL\_PROCESS, dan lainnya untuk mengelola alur kerja. Terdapat dua komponen yang digunakan, yaitu caesarCipher untuk enkripsi Caesar dan hillCipher untuk enkripsi Hill, masing-masing memproses data melalui sinyal seperti input\_char, caesar\_out, dan hill\_out. Sinyal current\_state dan next\_state digunakan untuk transisi antar-keadaan, sementara sinyal tambahan seperti done dan error melacak hasil proses. Mode operasi dikontrol oleh sinyal processing\_mode yang menentukan apakah algoritma Caesar atau Hill digunakan.

```
1 entity Main is
2   Port (
3     clk      : in STD_LOGIC;
4     rst      : in STD_LOGIC;
5     mode     : in STD_LOGIC;
6     start    : in STD_LOGIC;
7     done_out : out STD_LOGIC;
8     error_out: out STD_LOGIC
9   );
10 end Main;
11
12 architecture Behavioral of Main is
13   type state_type is (
14     IDLE,
15     READ_INPUT,
16     CAESAR_PROCESS,
17     HILL_PROCESS,
18     WRITE_OUTPUT,
19     COMPLETE,
20     ERROR_STATE
21   );
22
23   signal current_state : state_type := IDLE;
24   signal next_state : state_type := IDLE;
25
26   component caesarCipher is
27     port (
28       input   : in std_logic_vector(7 downto 0);
29       cipher  : out std_logic_vector(7 downto 0)
30     );
31   end component;
32
33   component hillCipher is
34     port (
35       input   : in STD_LOGIC_VECTOR(7 downto 0);
36       mode    : in STD_LOGIC;
37       output  : out STD_LOGIC_VECTOR(7 downto 0)
38     );
39   end component;
```



# CAESAR CIPHER CODE

Caesar cipher code berisi sebuah entitas bernama caesarCipher yang menerapkan enkripsi Caesar dengan konstanta SHIFT\_VALUE sebesar 3. Entitas ini memiliki input berupa std\_logic\_vector 8-bit yang merepresentasikan karakter ASCII dan output berupa hasil enkripsi dalam format yang sama. Dalam arsitektur Behavioral, proses dilakukan dengan membaca input, mengonversinya menjadi tipe integer menggunakan fungsi to\_integer dan unsigned. Logika pengolahan memeriksa apakah nilai ASCII input berada dalam rentang huruf besar A-Z atau huruf kecil a-z. Jika iya, pergeseran karakter dihitung berdasarkan operasi modular untuk memastikan tetap dalam rentang alfabet. Hasil akhirnya dikonversi kembali ke tipe std\_logic\_vector menggunakan to\_unsigned sebelum diberikan ke output cipher.

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

entity caesarCipher is
    port (
        input      : in  std_logic_vector(7 downto 0);
        cipher     : out std_logic_vector(7 downto 0)
    );
end entity caesarCipher;

architecture Behavioral of caesarCipher is
    constant SHIFT_VALUE : integer := 3;
begin
    process(input)
        variable temp : integer;
    begin
        temp := to_integer(unsigned(input));

        if (temp >= 65 and temp <= 90) then
            temp := ((temp - 65 + SHIFT_VALUE) mod 26) + 65;
        elsif (temp >= 97 and temp <= 122) then
            temp := ((temp - 97 + SHIFT_VALUE) mod 26) + 97;
        end if;

        cipher <= std_logic_vector(to_unsigned(temp, 8));
    end process;
end architecture Behavioral;
```



# HILL CIPHER CODE

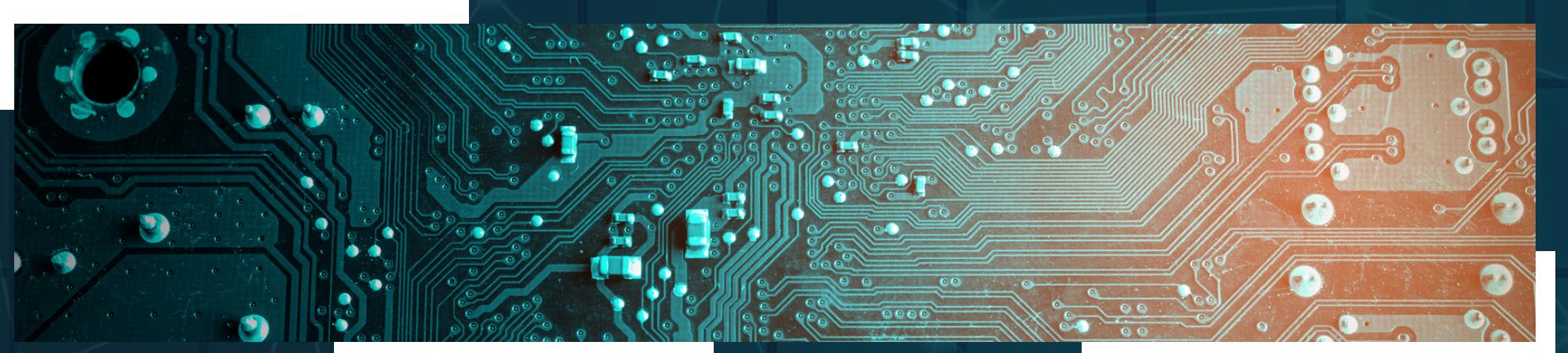
Hill cipher code berisi sebuah entitas bernama hillCipher yang menerapkan enkripsi dan dekripsi menggunakan algoritma Hill Cipher dengan matriks kunci 2x2. Entitas ini memiliki input berupa std\_logic\_vector 8-bit yang merepresentasikan karakter ASCII, mode untuk menentukan proses enkripsi atau dekripsi, dan output berupa hasil transformasi. Dalam arsitektur Behavioral, matriks kunci dan determinan invers didefinisikan sebagai konstanta, sementara fungsi mod26 digunakan untuk menjaga hasil tetap dalam rentang 0 hingga 25. Proses transformasi memetakan input ke integer, lalu menghitung nilai baru berdasarkan mode operasi menggunakan matriks kunci untuk enkripsi atau determinan invers untuk dekripsi. Karakter hasil dikembalikan ke tipe std\_logic\_vector setelah ditambahkan offset untuk menjaga format huruf besar atau kecil, sementara karakter non-alfabet diteruskan tanpa perubahan.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity hillCipher is
    port (
        input : in STD_LOGIC_VECTOR(7 downto 0);
        mode : in STD_LOGIC; -- 0 for forward process, 1 for reverse
        output : out STD_LOGIC_VECTOR(7 downto 0)
    );
end entity hillCipher;

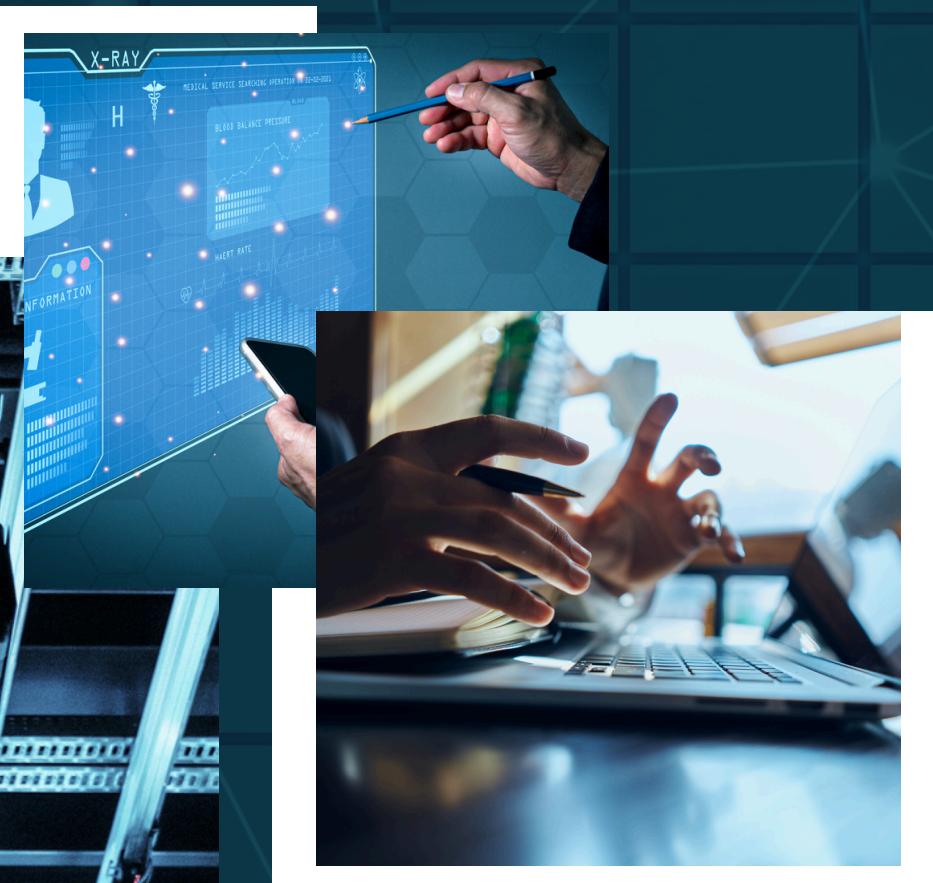
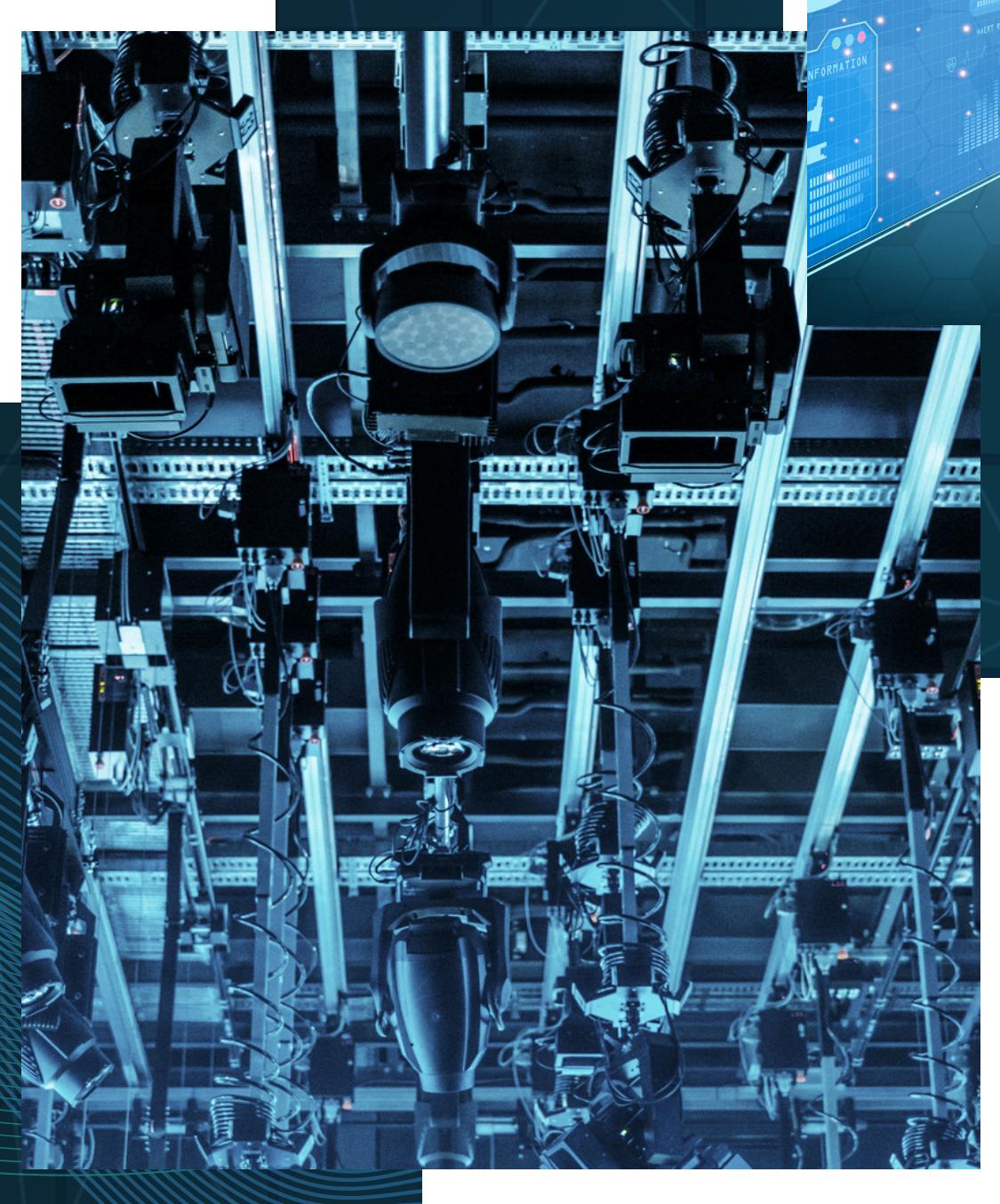
architecture Behavioral of hillCipher is
    -- Matriks kunci 2x2 untuk Hill Cipher
    type matrix_2x2 is array(0 to 1, 0 to 1) of integer;
    constant KEY_MATRIX : matrix_2x2 := ((3, 2), (2, 5));
    constant DET_KEY : integer := 3 * 5 - 2 * 2;
    constant DET_INV : integer := 17; -- Inverse dari determinan
    function mod26(x : integer) return integer is
    begin
        return ((x mod 26 + 26) mod 26);
    end function;
begin
    process(input, mode)
        variable char_value : integer range 0 to 255;
        variable transformed_1 : integer;
    begin
        char_value := to_integer(unsigned(input));
        if char_value >= 65 and char_value <= 90 then
            if mode = '0' then -- Forward process (Encrypt)
                transformed_1 := mod26(KEY_MATRIX(0,0) * (char_value - 65) + KEY_MATRIX(0,1) * (char_value - 65));
                output <= std_logic_vector(to_unsigned(transformed_1 + 65, 8));
            else -- Reverse process (Decrypt)
                transformed_1 := mod26(DET_INV * (KEY_MATRIX(1,1) * (char_value - 65) - KEY_MATRIX(0,1) * (char_value - 65)));
                output <= std_logic_vector(to_unsigned(transformed_1 + 65, 8));
            end if;
        elsif char_value >= 97 and char_value <= 122 then
            if mode = '0' then -- Forward process (Encrypt)
                transformed_1 := mod26(KEY_MATRIX(0,0) * (char_value - 97) + KEY_MATRIX(0,1) * (char_value - 97));
                output <= std_logic_vector(to_unsigned(transformed_1 + 97, 8));
            else -- Reverse process (Decrypt)
                transformed_1 := mod26(DET_INV * (KEY_MATRIX(1,1) * (char_value - 97) - KEY_MATRIX(0,1) * (char_value - 97)));
                output <= std_logic_vector(to_unsigned(transformed_1 + 97, 8));
            end if;
        else
            output <= input;
        end if;
    end process;
end architecture Behavioral;
```



# TESTING



Pengujian sistem dilakukan dengan Testbench dan file I/O untuk memastikan proses enkripsi dan dekripsi berjalan sesuai harapan. File input.txt berisi teks hello digunakan sebagai masukan. Proses enkripsi diawali dengan Caesar Cipher, menghasilkan file sementara phase1.txt, dilanjutkan Hill Cipher dengan hasil akhir disimpan di encryptOutput.txt. Untuk verifikasi, file encryptOutput.txt diproses ulang melalui dekripsi Hill Cipher dan Caesar Cipher secara terbalik, dengan hasil disimpan di decryptOutput.txt. Pengujian memastikan bahwa isi decryptOutput.txt sesuai dengan input.txt, membuktikan sistem bekerja dengan benar tanpa kesalahan.



# RESULT



# KOMBINASI CAESAR DAN HILL CYPHER DALAM INPUT HELLO

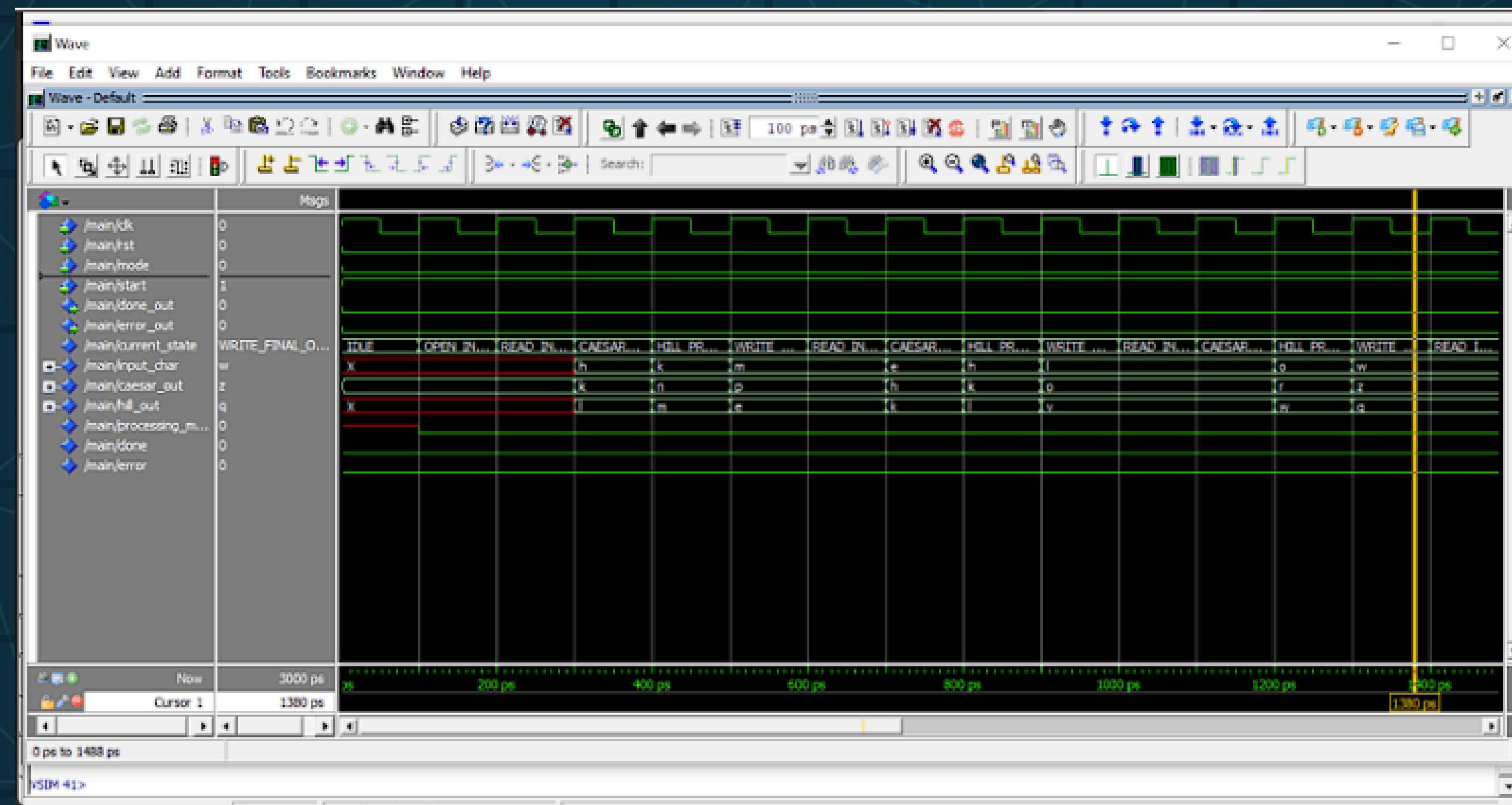
≡ encryptOutput.txt	
1	m
2	l
3	w
4	w
5	x
6	

≡ decryptOutput.txt	
1	e
2	b
3	i
4	i
5	l
6	

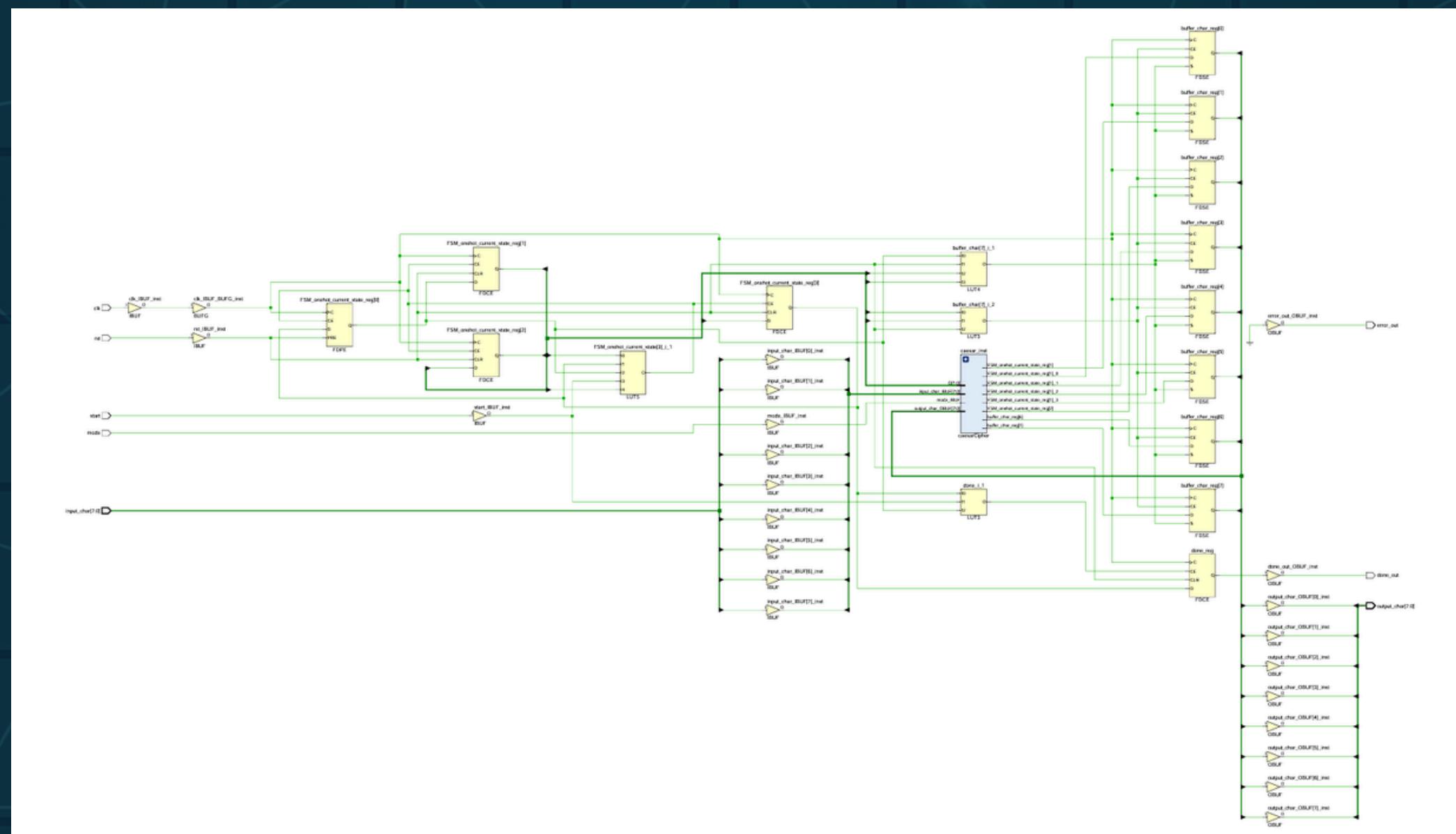


# KOMBINASI CAESAR DAN HILL CYPHER DALAM INPUT HELLO





# KOMBINASI CAESAR DAN HILL CYPHER DALAM INPUT HELLO





# KOMBINASI CAESAR DAN HILL CYPHER DALAM INPUT HELLO

```
VSIM 3> run -all
# ** Warning: NUMERIC_STD.IO_INTEGER: metavalue detected, returning 0
#   Time: 0 ps Iteration: 0 Instance: /maintb/uut/caesar_inst
# ** Warning: NUMERIC_STD.IO_INTEGER: metavalue detected, returning 0
#   Time: 0 ps Iteration: 2 Instance: /maintb/uut/caesar_inst
# ** Note: SKENARIO ENKRIPSI
#   Time: 10 ns Iteration: 0 Instance: /maintb
# ** Note: Input written to file: hello
#   Time: 10 ns Iteration: 0 Instance: /maintb
# ** 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
#   WORMA: ---- - ??
```



# KOMBINASI CAESAR DAN HILL CYPHER DALAM INPUT HELLO

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.



# KESIMPULAN

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.



# TERIMA KASIH