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

AES ENCRYPTOR DECRYPTOR GENERATOR USING VHDL

GROUP 25

NAHL SYAREZA RAHIDRA	**2206830340**
XAVIER DANISWARA	**2206030230**
M. SESARAFLI ALJAGRA	**2206828071**
REICHAN ADHIGUNO	**2106703273**
SAMUEL TANAKA	**2206059710**

PREFACE

This report details the final project for the Fundamental of Digital System course, focusing on the design and implementation of an **AES Encryptor Decryptor Generator using VHDL**.

The project successfully implements the 128-bit Advanced Encryption Standard (AES) cryptographic algorithm using the VHDL hardware description language. The core objective of this system is to process text input and perform either encryption or decryption, governed by a specific operation code (opcode). The system is built as a single, integrated, and modular entity.

A key architectural feature of this project is the use of a **Finite State Machine (FSM)** to control the execution flow. The FSM ensures that every stage of the AES process, including the S-Box, ShiftRows, MixColumns, AddRoundKey, and Key Expansion modules, runs in a structured and sequential manner.

The comprehensive implementation of this project demonstrates the application of various core digital system design concepts, including:

- **Behavioral Style** in all process statements.
- **Structural Modeling** and **Structural Programming** via component instantiation in the main module.
- **Loop Constructs** (FOR loop) utilized for repetitive processes, such as in the S-Box and Inverse S-Box operations.
- **Functions** and **Procedures** for process encapsulation, including Galois Field operations and byte substitution.
- **Microprogramming** through the use of opcodes to control system functions.

The successful integration of these modules validates a deep understanding of the course material and the ability to apply it in a practical data security application

Depok, December 06, 2025

Group 25

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Proyek ini dirancang sebagai sebuah aplikasi berbasis bahasa pemrograman VHDL yang mampu memproses input berupa teks dan mengubahnya melalui mekanisme enkripsi ataupun mengembalikannya ke bentuk semula melalui proses dekripsi. Mode operasi aplikasi ini dikendalikan menggunakan sebuah opcode yang menentukan tindakan yang harus dilakukan oleh sistem. Program ini menggunakan metode enkripsi AES (Advanced Encryption Standard), yang merupakan salah satu algoritma kriptografi paling umum digunakan karena keamanannya yang tinggi dan implementasinya yang efisien.

Selain itu, untuk memastikan alur operasi berlangsung secara teratur dan terstruktur, sistem ini dibangun menggunakan pendekatan Finite State Machine (FSM). Dengan FSM, setiap tahapan proses, dari menerima input, menentukan mode operasi, hingga menghasilkan output, dapat dikelola secara tersusun dan mudah dianalisis.

1.2 PROJECT DESCRIPTION

Proyek ini berfokus pada perancangan dan implementasi sistem keamanan digital menggunakan algoritma kriptografi Advanced Encryption Standard (AES) 128-bit yang dibangun dengan bahasa pemrograman VHDL. Sistem ini dirancang sebagai aplikasi modular yang mampu melakukan dua fungsi utama, yaitu enkripsi dan dekripsi teks, dalam satu entitas terintegrasi. Fitur pembeda utama dari sistem ini adalah penggunaan mekanisme Operation Code (Opcode) sebagai pengendali mode operasi. Dengan memasukkan kode biner tertentu, pengguna dapat secara dinamis memerintahkan sistem untuk beralih antara proses pengacakan data (enkripsi) atau pemulihan data (dekripsi) tanpa perlu mengubah konfigurasi perangkat keras.

Untuk memastikan stabilitas dan keteraturan alur pemrosesan data, arsitektur sistem ini dikendalikan oleh Finite State Machine (FSM). FSM berfungsi sebagai unit kontrol pusat yang mengatur urutan eksekusi modul-modul kriptografi, meliputi S-Box, ShiftRows, MixColumns, dan Key Expansion, serta AddRoundKey. Selain itu, proyek ini juga mengintegrasikan berbagai konsep perancangan sistem digital yang komprehensif, seperti

penggunaan behavioral style, structural modeling, loop constructs, serta penerapan functions dan procedures. Pendekatan ini tidak hanya menghasilkan sistem kriptografi yang aman dan efisien, tetapi juga menunjukkan struktur kode yang terorganisir dan mudah dianalisis.

1.3 OBJECTIVES

The objectives of this project are as follows:

1. Merancang dan mengimplementasikan sistem enkripsi dan dekripsi menggunakan algoritma AES-128 dengan bahasa pemrograman VHDL.
2. Menggunakan opcode dalam pengendalian operasi untuk membedakan mode enkripsi dan dekripsi.
3. Menerapkan Finite State Machine (FSM) untuk memastikan setiap tahapan AES berjalan secara terstruktur dan berurutan
4. Membangun arsitektur modular yang meliputi S-Box, Inverse S-Box, AddRoundKey, ShiftRows, MixColumns, dan Key Expansion.
5. Mengintegrasikan modul perancangan sistem digital, seperti behavioral style, structural modeling, loop constructs, functions, procedures, impure functions, dan microprogramming.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Leader	Menulis laporan Chapter 3 Menulis kode main untuk encryption dan decryption Menyatukan berbagai modul yang ada menjadi satu kesatuan pada main Menulis kode Shift Rows Penggagas ide proyek	Nahl Syareza Rahidra

Anggota Kelompok	Menulis laporan Chapter 1 dan 2 Merancang Testbench untuk Enkripsi dan Dekripsi	Xavier Daniswara
Anggota Kelompok	Menulis laporan Chapter 1 dan 3 Membuat kode untuk Key Rounds	M. Sesarafli Aljagra
Anggota Kelompok	Menulis laporan Chapter 1 dan 2 Membuat kode untuk Inverse Mix Columns	Reichan Adhiguno
Anggota Kelompok	Menulis laporan Chapter 1 dan 4 Membuat kode bagian Mix Columns	Samuel Tanaka

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

The tools that are going to be used in this project are as follows:

- ModelSim, digunakan untuk simulasi kode VHDL untuk menganalisis dan verifikasi output dari kode.
- Visual Studio Code, digunakan sebagai code editor untuk menulis dan mengedit kode VHDL.
- Intel Quartus Prime, digunakan untuk proses sintesis, compile FPGA, dan analisis pada design VHDL.
- GitHub, digunakan sebagai platform untuk menyimpan kode hasil kolaborasi antar anggota kelompok.

2.2 IMPLEMENTATION

Komponen

SubBytes

- Direpresentasikan oleh `s_box`.
- Langkah substitusi non-linier yang menggunakan tabel S-Box untuk mengubah setiap byte state.
- Setiap byte diganti dengan nilai S-Box untuk enkripsi, dan dengan nilai inverse S-Box untuk dekripsi.
- Signal `s_box_inp` untuk original input, `s_box_outp` untuk hasil input yang sudah diubah, dan `encrypt_or_decrypt` untuk *selector*/pemilihan mode enkripsi atau dekripsi.

ShiftRows

- Direpresentasikan oleh `le_shift`.
- Menggeser baris pada matriks state.
- Pergeseran ke arah kiri untuk enkripsi, dan kanan untuk dekripsi.

- Signal `le_shift_inp` untuk original input, `le_shift_outp` untuk hasil shifting atau pergeseran, `le_shift_enable` untuk memulai proses pergeseran, `le_shift_done` untuk flag sebagai penanda bahwa proses shifting atau pergeseran sudah selesai, dan `encrypt_or_decrypt` untuk menentukan arah shifting (kiri untuk enkripsi dan kanan untuk dekripsi).
- Urutan pergeseran baris adalah pada baris pertama matriks tidak dilakukan shifting, baris kedua dilakukan shifting sekali, baris kedua dilakukan shifting dua kali, dan baris ketiga dilakukan shifting sebanyak tiga kali.

MixColumns

- Direpresentasikan dengan `el_mixer` untuk proses enkripsi dan `rexim_le` untuk proses dekripsi.
- Melakukan operasi matriks pada setiap kolom 4 byte dengan perkalian Galois Field (GF 2^8)
- Melakukan operasi matriks menggunakan matriks invers pada dekripsi.

KeyExpansion

- Direpresentasikan oleh key_round.
- KeyExpansion menghasilkan round keys dari key awal.
- Untuk AES-128, terdapat total 11 round keys (round 0 sampai round 10).
- Signal `key_round_sel` untuk memilih round yang mana, `key_round_inp` untuk input, `key_round_outp` untuk output, dan `key_round_kagi` untuk round yang sudah di-expand.
- Untuk urutan proses expansion pada mode enkripsi adalah sebagai berikut:
    - RotWord untuk melakukan rotasi satu byte ke kiri,
    - SubWord untuk melakukan substitusi dengan S-Box, dan
    - XOR dengan konstanta untuk round keys Rcon.

Proses enkripsi pada algoritma AES dilaksanakan melalui rangkaian tahapan SubBytes, ShiftRows, MixColumns, dan AddRoundKey pada setiap putaran (round) dari round ke-1 hingga round ke-9, dengan pengecualian pada round ke-10 yang tidak menyertakan tahap MixColumns. Sebaliknya, proses dekripsi menerapkan urutan Inverse ShiftRows, Inverse SubBytes, AddRoundKey, dan Inverse MixColumns, dengan ketentuan bahwa pada round ke-10 tahap MixColumns juga dihilangkan.

IMPLEMENTASI DALAM SETIAP MODUL

a. Behavioral Style

   Behavioral Style digunakan dalam setiap entity yang ada, dengan pengimplementasian process pada setiap architecture dari entity.

b. Testbench

   Testbench dibuat untuk memudahkan simulasi dan melakukan pengecekkan bila terdapat output yang tidak sesuai harapan.

c. Structural Programming

   Structural Programming digunakan untuk mempermudah dalam mendesain struktur kode. Masing-masing tahapan dari AES encryption memiliki entity masing-masing yang melakukan fungsinya sendiri dan kemudian dipanggil pada main untuk digunakan sebagai satu kesatuan.

d. Looping

   Looping Construct digunakan untuk proses yang memerlukan repetisi, contoh misalnya pada proses Sub Bytes di enkripsi maupun dekripsi.

e. Procedure, Function, and Impure Function

   Ketiga hal digunakan untuk mengenkapsulasi proses pada architecture. Seringkali ketika kita memprogram VHDL, kita memerlukan proses yang sama berulang kali. Procedure, Function, dan Impure Function bisa mengatasi permasalahan tersebut.

f. Finite State Machine

   FSM atau Finite State Machine digunakan untuk merincikan dan mengarahkan program akan alur yang state yang harus diambil selanjutnya dan proses yang harus dilakukan. Setiap state pada FSM dikontrol oleh opcode

g. Microprogramming

   Digunakan untuk bisa memberikan instruksi kepada kode VHDL akan tahapan yang harus dilakukan sekarang. Instruksi yang diberikan merupakan 4 bit

dengan MSB yang menandakan apakah instruksi yang diberikan berbentuk enkripsi atau dekripsi.

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Pada bagian testing, kita akan melihat hasil dari proses enkripsi dan dekripsi pada satu ronde. Hasil dari output kode akan dicocokkan pada nilai yang seharusnya terjadi setelah step pada ronde tersebut selesai

3.1.1 ENKRIPSI

Untuk bagian enkripsi, ia akan melewati beberapa-beberapa tahapan dulu pada round pertamanya sebagai permulaan dari proses enkripsi AES. Tahapan yang pertama adalah XOR, dilanjut dengan Sub Bytes, Shift Rows, Mix Columns, dan terakhir Round Key. Untuk contoh ini, akan digunakan nilai berikut sebagai input dan key.

input = 48656C6C6F20576F726C642121212121

key = 2B7E151628AED2A6ABF7158809CF4F3C

- XOR

  Proses ini sederhana, yaitu lakukan operasi XOR antara input dengan Key ke-0 (alias key awal asli). Dalam kode, hasil disimpan dulu pada oi_buffer. Hasil dair XOR ini adalah 631B797A478E85C9D99B71A928EE6E1D.

- Sub Bytes

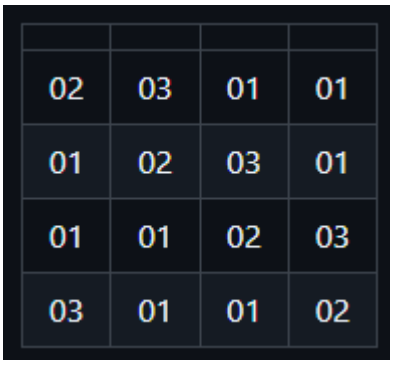
  Proses ini agak sedikit kompleks. Proses ini adalah mengubah masing-masing byte menjadi bentuk lain menggunakan lookup table S-Box. Dikarenakan ini adalah algoritma AES 128 bit, maka akan terdapat 16 byte, sehingga akan terjadi 16 kali pertukaran nilai byte. Input pada tahapan ini adalah output dari sebelumnya, yaitu XOR. Hasil pada tahapan ini adalah FBAFB6DAA01997DD3514A3D334289FA4.

- Shift Rows

  Pada proses ini akan dilakukan shifting ke kiri secara siklik kepada masing-masing baris. Pada tahap ini, kita menganggap input kita menjadi sebuah matriks 4x4. Baris pertama tidak melakukan shift, baris kedua dilakukan 1 kali, baris ketiga dilakukan 2 kali, dan baris keempat dilakukan 3 kali. Sama seperti sebelumnya, input pada tahapan ini adalah output dari tahapan sebelumnya. Hasil pada tahap ini adalah FB19A3A4A0149FDA3528B6DD34AF97D3.

- Mix Columns

  Proses ini paling kompleks dibandingkan proses-proses sebelumnya. Intinya, kita akan mengalikan input kita dengan sebuah matriks 4x4.



02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

  Terdengar sederhana secara konsep, namun kompleksitasnya berada pada prosedurnya. Byte harus melewati proses dimana ia dikali 2, namun jika MSB pada Byte adalah 1, maka ia harus di XOR dengan 0x1B. Byte juga harus melewati proses dikali 3, namun ini cukup mudah yaitu dengan melakukan XOR dari output kali dua sebelumnya dengan Byte itu sendiri. Input pada tahapan ini adalah output dari tahapan sebelumnya. Hasil pada tahap ini adalah C19348FF22E8E4DF79791660C600C0D9.

- Round Key

  Proses pada tahap ini sederhana, yaitu hanya melakukan XOR antara input sebelumnya dengan key, namun key pada tahapan ini bukanlah key awal, namun Key untuk round pertama, alias Key ke-1. Untuk mendapatkan Key Round tersebut, key asli harus di rotate ke kiri, di lookup dengan S-Box, dan di XOR dengan Round Constant. Hasil tahap ini adalah 6169B6E8AABCC86E5ADA2F59EC6CB6DC, yang merupakan hasil untuk Ronde Pertama. Dalam program, terdapat sebuah tahapan terakhir yaitu ENCRYPT_FINAL untuk menginisialisasikan variabel oi_buffer untuk bisa digunakan pada ronde selanjutnya.

3.1.2 DEKRIPSI

Secara sederhana, proses dekripsi adalah proses kebalikan dari enkripsi. Namun terdapat beberapa hal yang perlu diperhatikan. Pada akhir ronde, yaitu Rone Kesepuluh pada enkripsi, ia melongkap tahapan untuk Mix Columns. Maka dari itu, sebagai awalan dari proses dekripsi, tidak akan dilakukan proses Mix Columns (dalam kasus ini berarti Unmix harusnya, tapi yaudah). Maka dari itu, untuk contoh ini, akan dilakukan proses XOR, Shift Rows, Sub Bytes, dan Round Key. Untuk contoh ini, akan digunakan input dan key (key yang digunakan sama) yaitu:

input = 052BC3FB4A020CEDCBAB86FC0C06D404

key = 2B7E151628AED2A6ABF7158809CF4F3C

- XOR

  Mirip dengan proses XOR pertama pada saat enkripsi, namun bedanya key yang digunakan adalah key pada Ronde Kesepuluh. Ini akan memberikan hasil output dari proses Shift Rows pada proses enkripsi Ronde Kesepuluh. Hasil pada tahap ini adalah. D53F3A5383EC29642A948A34BA65D8A2.

- Shift Rows

  Melakukan hal yang sama seperti Shift Rows pada tahap enkripsi, namun pada dekripsi, lakukan shift ke kanan sebagai bentuk invers dari shift ke kiri pada tahap enkripsi. Hasil pada tahap ini adalah D5658A64833FD8342AEC3AA2BA942953.

- Sub Bytes

  Proses ini juga mirip seperti enkripsi, namun lookup table yang digunakan adalah lookup table untuk Inverse S-Box. Sederhananya, ia akan mengembalikan text awal dari text yang sudah di encode dengan S-Box. Hasil pada tahap ini adalah B5BCCF8C41252D289583A21AC0E74C50. Hasil tersebut juga merupakan hasil akhir dari Ronde Kesembilan.

- Round Key

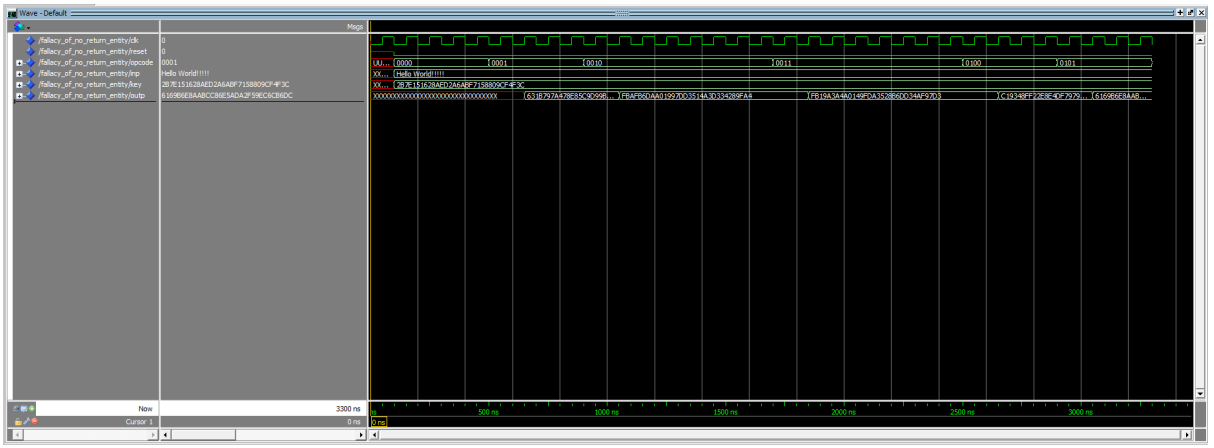
  Namun kita tidak bisa berhenti saja setelah mendapatkan hasil tersebut, karena itu bukan yang kita butuhkan sebagai input pada tahapan selanjutnya. Input yang dibutuhkan untuk tahapan selanjutnya adalah input yang di XOR dengan key Ronde Kesembilan yang menghasilkan hasil B5BCCF8C41252D289583A21AC0E74C50. Untuk mendapatkannya sederhana, yaitu lakukan XOR key Ronde Kesembilan dengan B5BCCF8C41252D289583A21AC0E74C50. Hasil pada tahap ini adalah 19CBA97F58DFF109BD528B5B97BB4C3E. Sama seperti enkripsi, terdapat DECRYPT_FINAL untuk mempersiapkan kode dan variable oi_buffer untuk tahapan selanjutnya pada dekripsi.

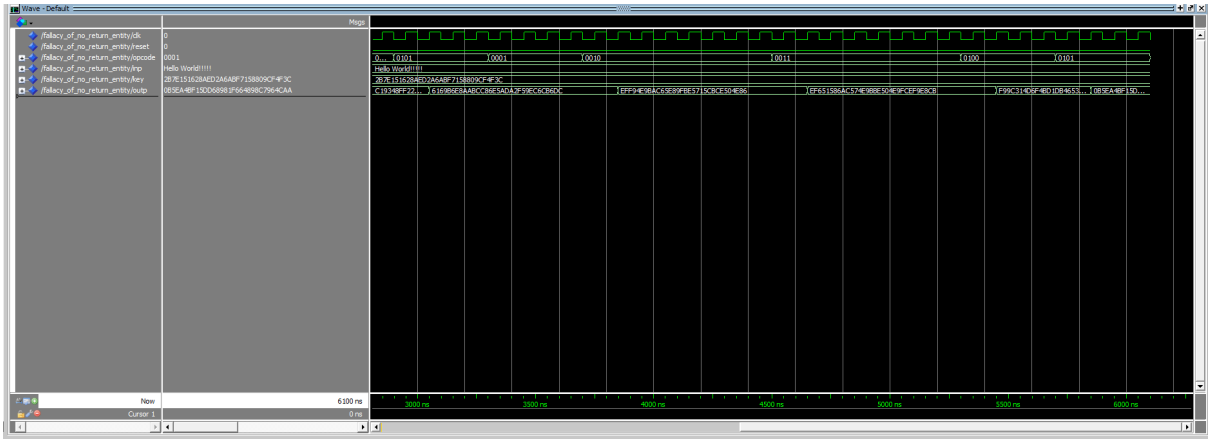
3.2 RESULT

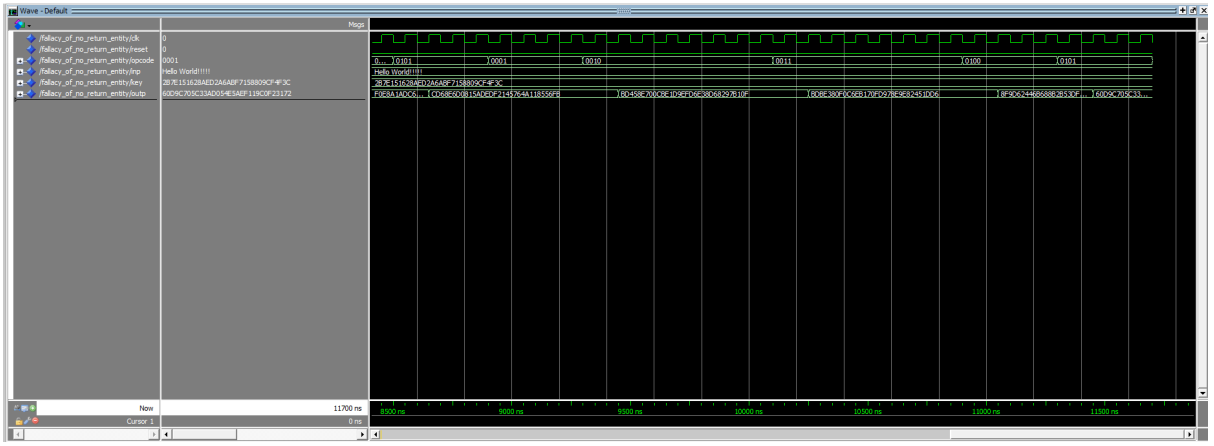
Berikut adalah hasil yang diberikan oleh program. Di sini akan diberikan hasil dari proses enkripsi dan dekripsi secara keseluruhan, yang berarti telah dilakukan proses untuk 10 ronde.

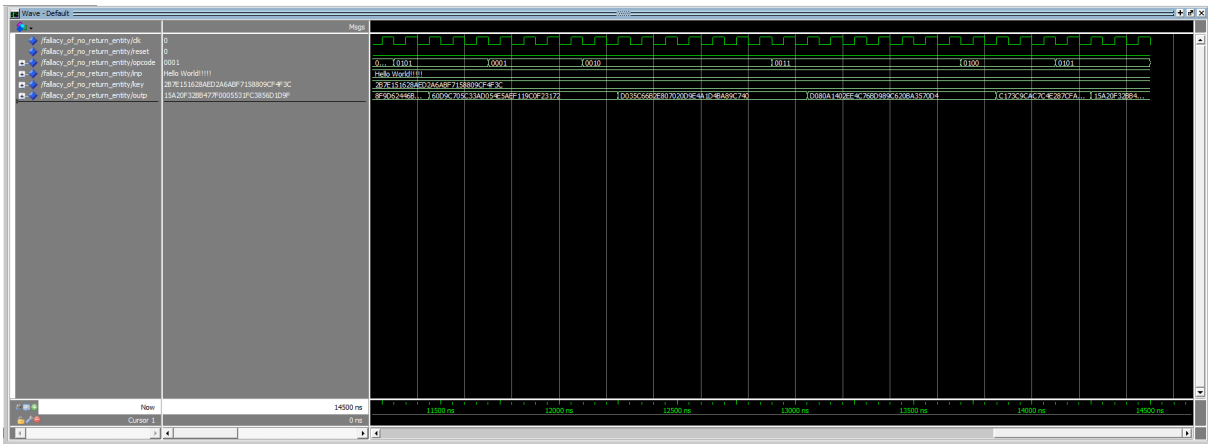
3.2.1 ENKRIPSI

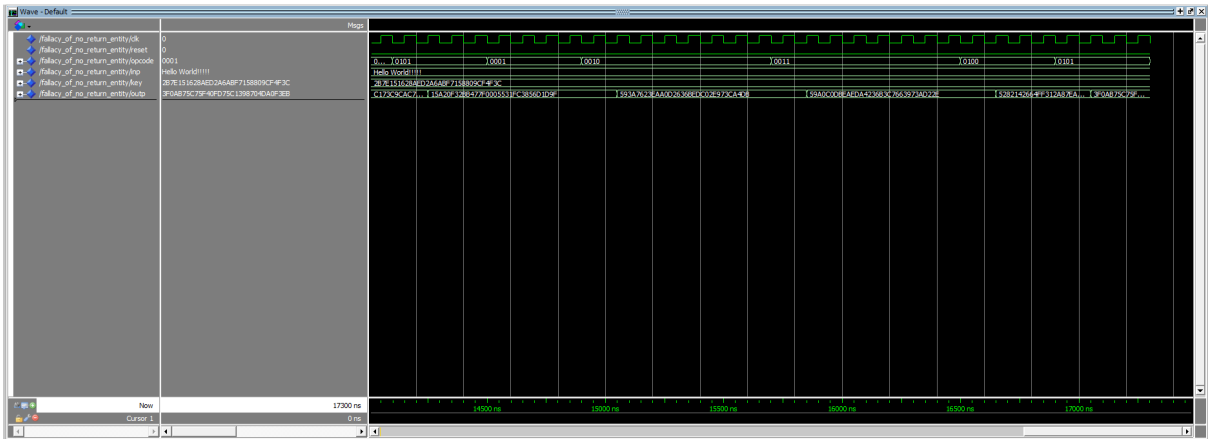
Berikut adalah hasil waveform dari enkripsi pada masing-masing ronde nya

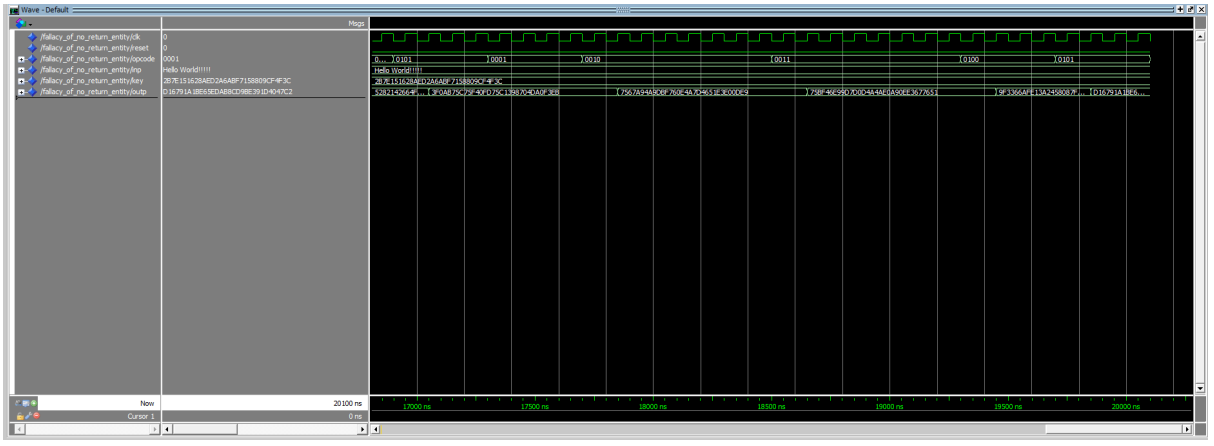


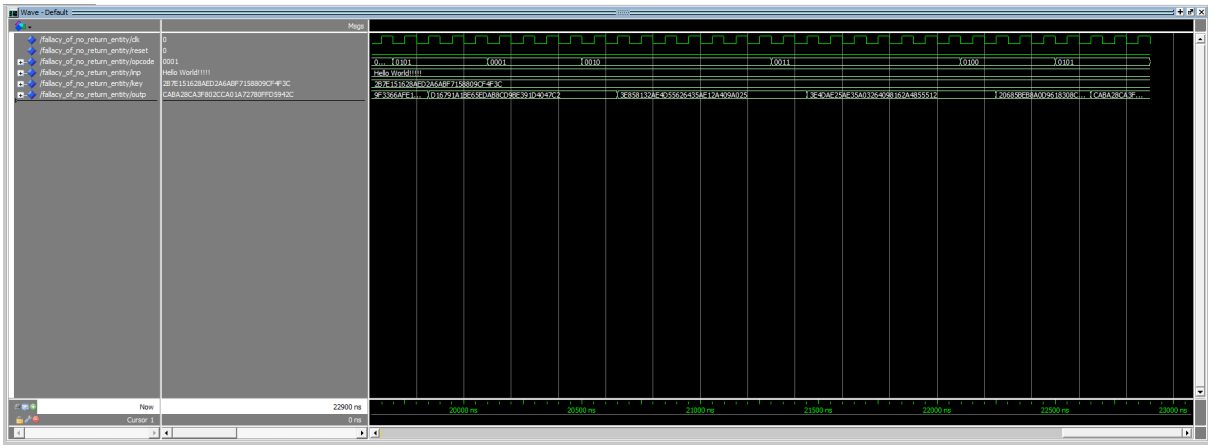


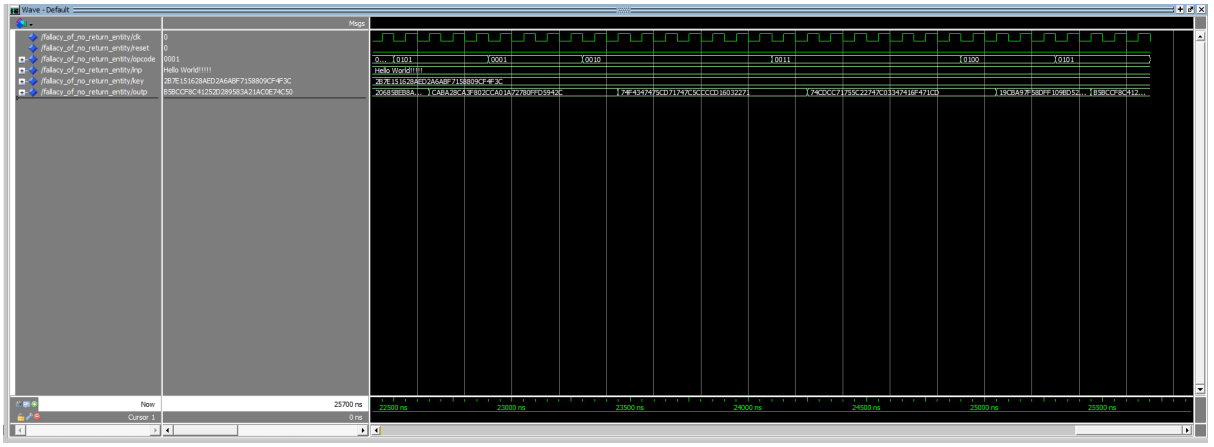

Wave - Default
Msgs
/fallacy_of_no_return_entity/clk
/fallacy_of_no_return_entity/reset
/fallacy_of_no_return_entity/opcode
/fallacy_of_no_return_entity/inp
/fallacy_of_no_return_entity/key
/fallacy_of_no_return_entity/outp
0
0
0001
Hello World!!!!!
2B7E151628AED2A6ABF7158809CF4F3C
60D9C705C33AD054E5AEF119C0F23172
0101
0001
0010
0011
0100
0101
Now
11700 ns
Cursor 1
0 ns
8500 ns
9000 ns
9500 ns
10000 ns
10500 ns
11000 ns
11500 ns

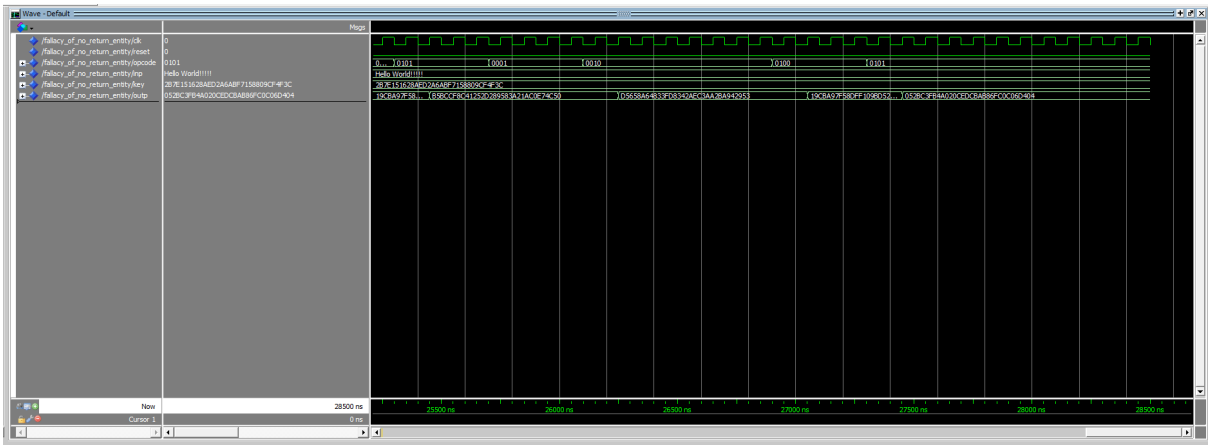

Wave - Default
Msgs
/fallacy_of_no_return_entity/clk
/fallacy_of_no_return_entity/reset
/fallacy_of_no_return_entity/opcode
/fallacy_of_no_return_entity/inp
/fallacy_of_no_return_entity/key
/fallacy_of_no_return_entity/outp
0
0
0001
Hello World!!!!!
2B7E151628AED2A6ABF7158809CF4F3C
15A20F32B847
0101
0001
0010
0011
0100
0101
Now
14500 ns
Cursor 1
0 ns
11500 ns
12000 ns
12500 ns
13000 ns
13500 ns
14000 ns
14500 ns


Wave - Default
Msgs
/fallacy_of_no_return_entity/clk
/fallacy_of_no_return_entity/reset
/fallacy_of_no_return_entity/opcode
/fallacy_of_no_return_entity/inp
/fallacy_of_no_return_entity/key
/fallacy_of_no_return_entity/outp
0
0
0001
Hello World!!!!!
2B7E151628AED2A6ABF7158809CF4F3C
0101
0001
0010
0011
0100
0101
Now
17300 ns
Cursor 1
0 ns
14500 ns
15000 ns
15500 ns
16000 ns
16500 ns
17000 ns


Wave - Default
Msgs
/fallacy_of_no_return_entity/clk
/fallacy_of_no_return_entity/reset
/fallacy_of_no_return_entity/opcode
/fallacy_of_no_return_entity/inp
/fallacy_of_no_return_entity/key
/fallacy_of_no_return_entity/outp
0
0
0001
Hello World!!!!!
2B7E151628AED2A6ABF7158809CF4F3C
0101
0001
0010
0011
0100
0101
Now
20100 ns
Cursor 1
0 ns
17000 ns
17500 ns
18000 ns
18500 ns
19000 ns
19500 ns
20000 ns


Wave - Default
Msgs
/fallacy_of_no_return_entity/clk
/fallacy_of_no_return_entity/reset
/fallacy_of_no_return_entity/opcode
/fallacy_of_no_return_entity/inp
/fallacy_of_no_return_entity/key
/fallacy_of_no_return_entity/outp
0
0
0001
Hello World!!!!!
2B7E151628AED2A6ABF7158809CF4F3C
CABA28CA3FB02CCA01A72780FFD5942C
0101
0001
0010
0011
0100
0101
Now
22900 ns
Cursor 1
0 ns
20000 ns
20500 ns
21000 ns
21500 ns
22000 ns
22500 ns
23000 ns


Wave - Default
Msgs
/fallacy_of_no_return_entity/clk
/fallacy_of_no_return_entity/reset
/fallacy_of_no_return_entity/opcode
/fallacy_of_no_return_entity/inp
/fallacy_of_no_return_entity/key
/fallacy_of_no_return_entity/outp
0
0
0001
Hello World!!!!!
2B7E151628AED2A6ABF7158809CF4F3C
B5BCCF8C41252D2895B3A21AC0E74C50
0101
0001
0010
0011
0100
0101
Now
25700 ns
Cursor 1
0 ns
22500 ns
23000 ns
23500 ns
24000 ns
24500 ns
25000 ns
25500 ns

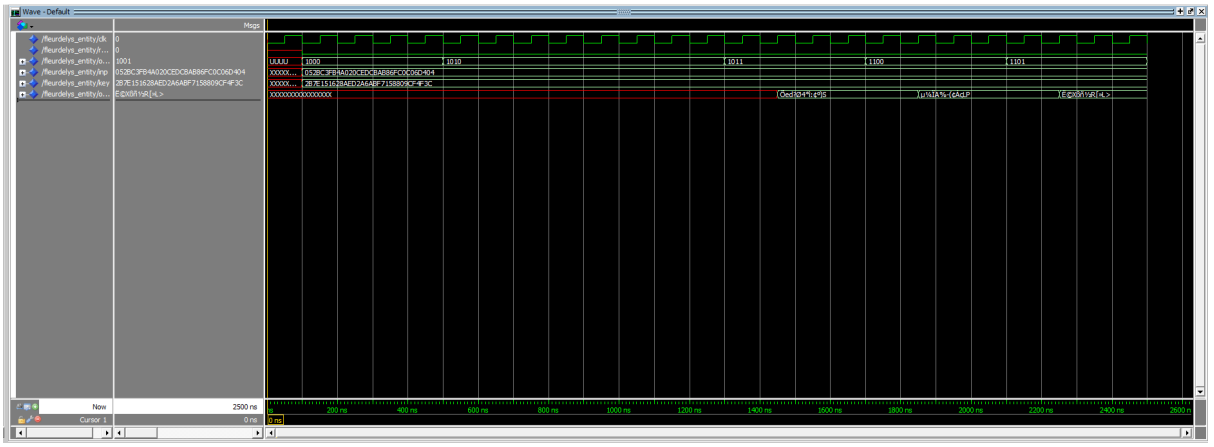

Wave - Default
Msgs
/fallacy_of_no_return_entity/clk
/fallacy_of_no_return_entity/reset
/fallacy_of_no_return_entity/opcode
/fallacy_of_no_return_entity/inp
/fallacy_of_no_return_entity/key
/fallacy_of_no_return_entity/outp
0
0
0101
Hello World!!!!!
2B7E151628AED2A6ABF7158809CF4F3C
052BCCF8C41252D2895B3A21AC0E74C50
0101
0010
0100
0101
Now
28500 ns
Cursor 1
0 ns
25500 ns
26000 ns
26500 ns
27000 ns
27500 ns
28000 ns
28500 ns

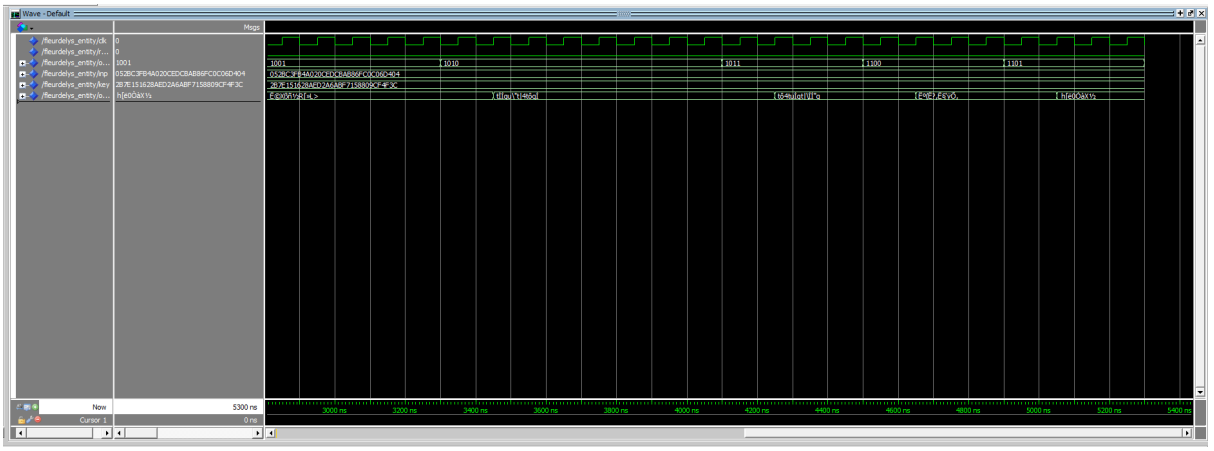
```
Input: 48656C6C6F20576F726C642121212121
Key: 2B7E151628AED2A6ABF7158809CF4F3C

Output: 052BC3FB4A020CEDCBAB86FC0C06D404
Key: 2B7E151628AED2A6ABF7158809CF4F3C

Output: 052BC3FB4A020CEDCBAB86FC0C06D404
Key: 2B7E151628AED2A6ABF7158809CF4F3C

Output: 052BC3FB4A020CEDCBAB86FC0C06D404
Key: 2B7E151628AED2A6ABF7158809CF4F3C
```

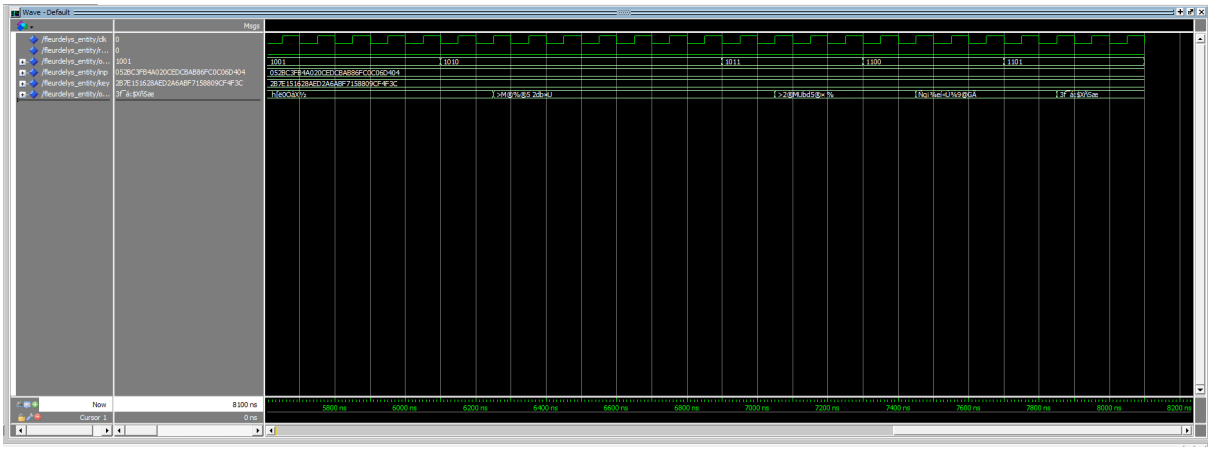
Hasil dari output itu sendiri merupakan hasil akhir dari masing-masing ronde.

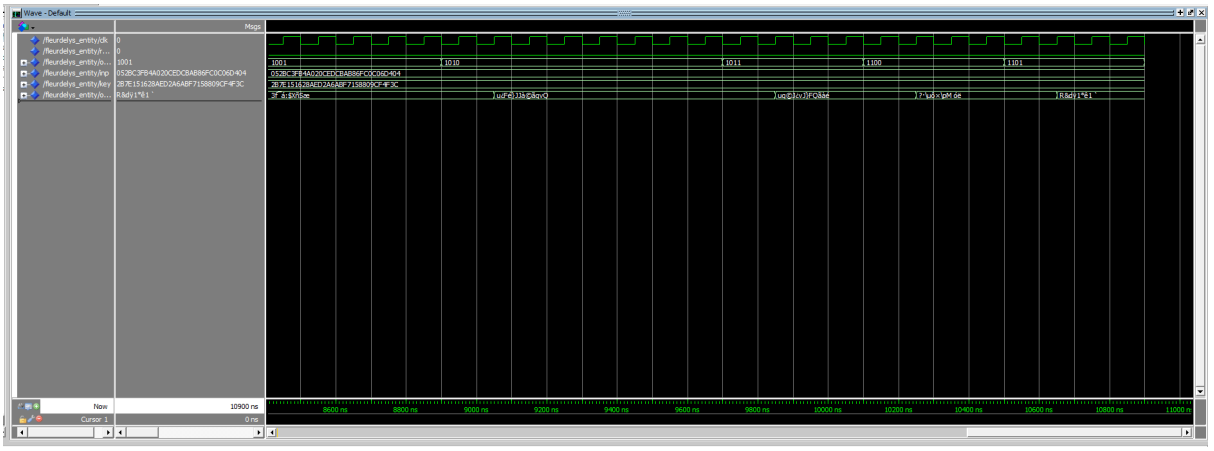
3.2.2 DEKRIPSI

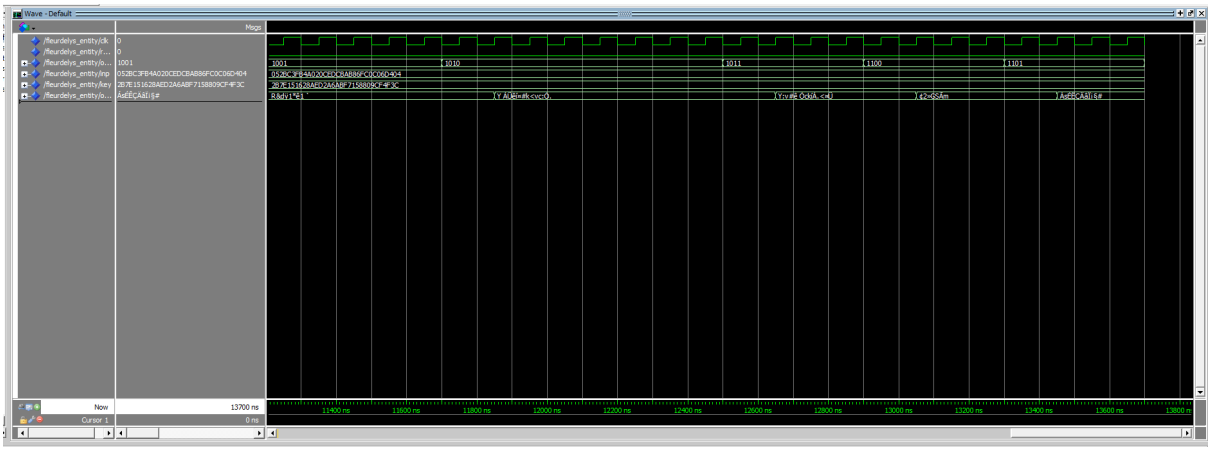
Berikut adalah waveform untuk dekripsi per ronde.

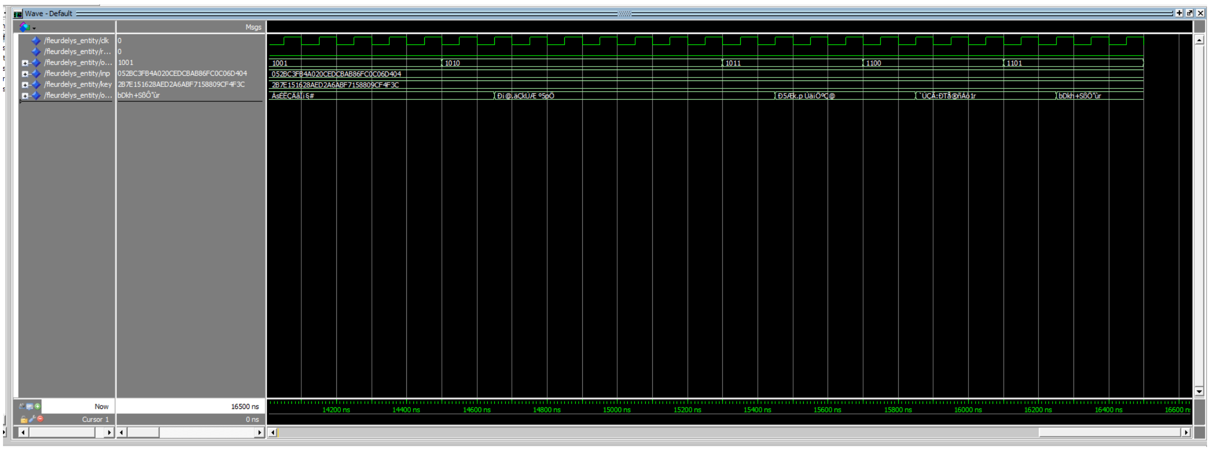




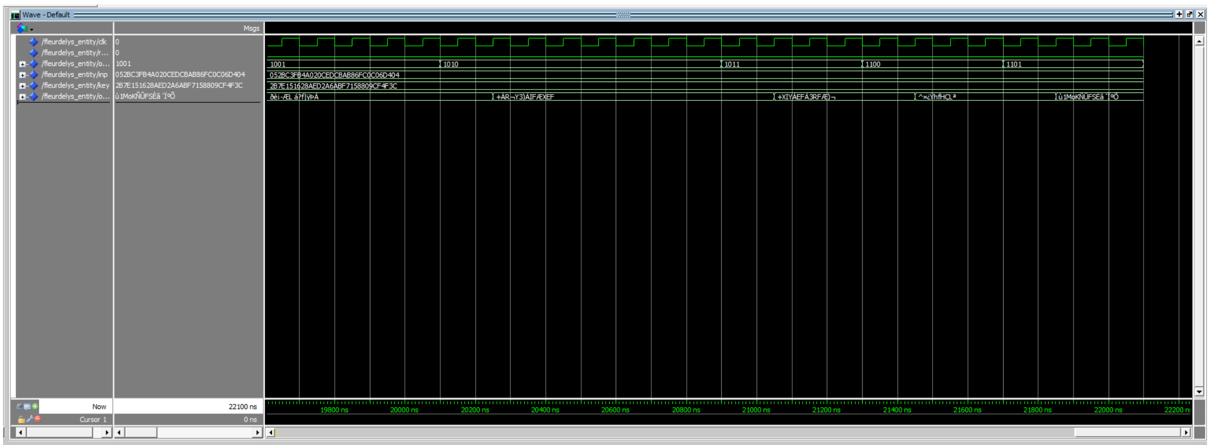

Wave - Default
Msgs
/fleurdelys_entity/clk
/fleurdelys_entity/r...
/fleurdelys_entity/o...
/fleurdelys_entity/inp
/fleurdelys_entity/key
/fleurdelys_entity/o...
0
0
1001
052BC3FB4A020CEDCBAB86FC0C06D404
2B7E151628AED2A6ABF7158809CF4F3C
Now
8100 ns
Cursor 1
0 ns
1001
1010
1011
1100
1101
9800 ns
9600 ns
9400 ns
9200 ns
9000 ns
8800 ns
8600 ns
8400 ns
8200 ns

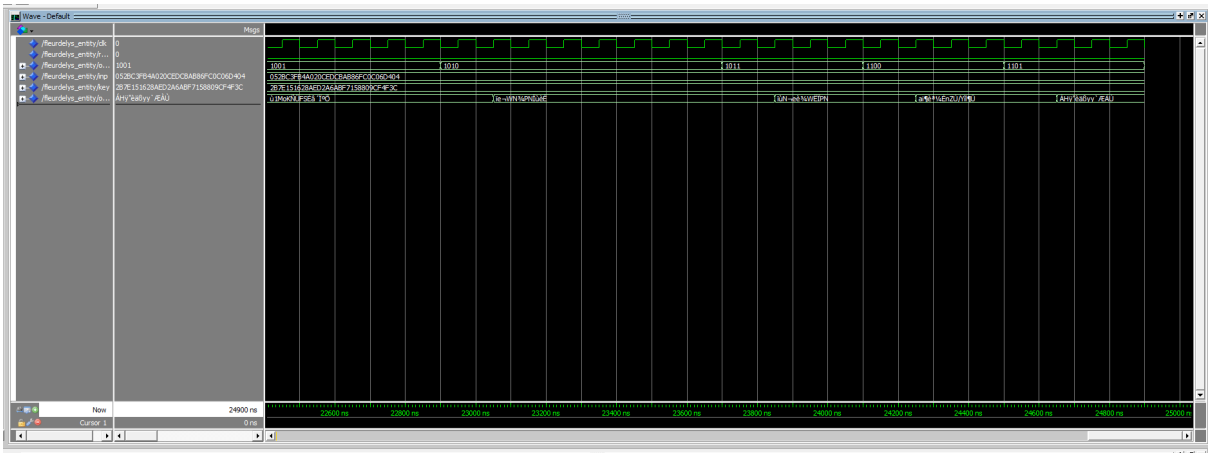

Wave - Default
Msgs
/fleurdelys_entity/clk
/fleurdelys_entity/r...
/fleurdelys_entity/o...
/fleurdelys_entity/inp
/fleurdelys_entity/key
/fleurdelys_entity/o...
0
0
1001
052BC3FB4A020CEDCBAB86FC0C06D404
2B7E151628AED2A6ABF7158809CF4F3C
Now
10900 ns
Cursor 1
0 ns
1001
1010
1011
1100
1101
10800 ns
10600 ns
10400 ns
10200 ns
10000 ns
9800 ns
9600 ns
9400 ns
9200 ns
9000 ns
8800 ns
8600 ns

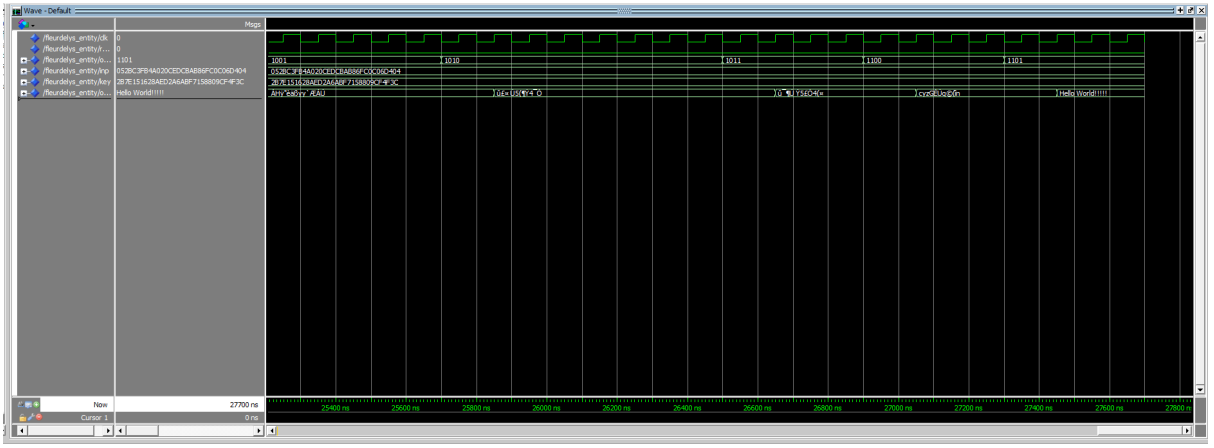

Wave - Default
Msgs
/fleurdelys_entity/clk
/fleurdelys_entity/r...
/fleurdelys_entity/o...
/fleurdelys_entity/inp
/fleurdelys_entity/key
/fleurdelys_entity/o...
0
0
1001
052BC3FB4A020CEDCBAB86FC0C06D404
2B7E151628AED2A6ABF7158809CF4F3C
Now
13700 ns
Cursor 1
0 ns
1001
1010
1011
1100
1101
13800 ns
13600 ns
13400 ns
13200 ns
13000 ns
12800 ns
12600 ns
12400 ns
12200 ns
12000 ns
11800 ns
11600 ns
11400 ns


Wave - Default
/fleurdelys_entity/clk
/fleurdelys_entity/inp
/fleurdelys_entity/key
052BC3FB4A020CEDCBAB86FC0C06D404
2B7E151628AED2A6ABF7158809CF4F3C
1001
1010
1011
1100
1101
Now
16500 ns
Cursor 1
0 ns


Wave - Default
/fleurdelys_entity/clk
/fleurdelys_entity/inp
/fleurdelys_entity/key
052BC3FB4A020CEDCBAB86FC0C06D404
2B7E151628AED2A6ABF7158809CF4F3C
1001
1010
1011
1100
1101
Now
19300 ns
Cursor 1
0 ns


Wave - Default
/fleurdelys_entity/clk
/fleurdelys_entity/inp
/fleurdelys_entity/key
052BC3FB4A020CEDCBAB86FC0C06D404
2B7E151628AED2A6ABF7158809CF4F3C
1001
1010
1011
1100
1101
Now
22100 ns
Cursor 1
0 ns





```
Input: 052BC3FB4A020CEDCBAB86FC0C06D404
Key: 2B7E151628AED2A6ABF7158809CF4F3C

Output: 48656C6C6F20576F726C642121212121
Key: 2B7E151628AED2A6ABF7158809CF4F3C
```

Untuk bagian dekripsi, output bukanlah hasil output dari ronde sebelumnya, namun merupakan data yang di XOR dengan key pada ronde tersebut yang akan menghasilkan output akhir per ronde nya.

3.3 ANALYSIS

Analisis yang bisa kita lakukan pada kode ini adalah bagaimana cara kerjanya. Selain mengimplementasikan tahap-tahap pada AES encryption/decryption, terdapat beberapa tahap tambahan yang perlu dilakukan untuk memastikan alur dapat berjalan dengan benar dalam VHDL sehingga menghasilkan ciphertext yang benar pula. Tahap yang ditambahkan adalah INIT dan FINAL. Tahapan INIT digunakan hanya pada awal-awal proses enkripsi atau dekripsi. Hal ini khusus karena bagi kedua proses, ronde pertama yang dilakukan cenderung berbeda dengan ronde-ronde selanjutnya. Misal pada ronde pertama enkripsi, sebelum input dilakukan Sub Bytes dengan S-Box, ia perlu di XOR terlebih dahulu dengan key asli. Sedangkan untuk dekripsi, ronde pertamanya juga melakukan hal yang mirip dengan enkripsi, namun dengan key yang sudah ditransformasi sebanyak 10 kali. Untuk yang kedua yaitu FINAL, tahapan tersebut digunakan untuk memberikan hasil kepada port output. Hal ini dikarenakan port output hanya bisa mendapatkan hasil dari tahap sebelumnya dan bukan tahap yang sekarang ini karena ia bergantung pada hasil yang sudah diproses dan bukan yang sedang diproses. Seperti misalnya, untuk menampilkan output dari tahapan Sub Bytes, kita harus masuk dulu ke dalam tahap Shift Rows. Maka dari itu, FINAL hadir untuk bisa memberikan output dari tahapan Key Round.

Selain penambahan tahapan-tahapan baru, module-module yang digunakan juga memiliki dua mode, yaitu encrypt or decrypt. Hal ini dapat mempermudah kontrol dan juga membuat kode sedikit lebih sederhana karena proses untuk tahapan enkripsi atau dekripsi terdapat pada satu file yang sama. Namun, tidak demikian untuk proses Mix Columns. Dikarenakan kesalahan berpikir dan penafsiran cara kerja dari Inverse Mix Columns, akhirnya kami memutuskan untuk menggunakan dua file berbeda untuk masing-masing tahap, tergantung dengan mode yang digunakan apakah enkripsi atau dekripsi.

Dalam kode pun, dapat diobservasi penggunaan Nested FSM, yaitu sebuah FSM di dalam FSM. Dalam kasus ini, entity main (yang merupakan FSM) memiliki module-module entity pembantu seperti le_shift (untuk Shift Rows), el_mixer (untuk Mix Columns), dan rexim_le (untuk Inverse Mix Columns). Ketiga entity pembantu tersebut memiliki FSM sendiri di dalamnya. Agar kode tetap tersinkronisasi, diperlukan sebuah Flag yang dapat memberitahukan apakah child FSM sudah selesai. Hal ini dapat dengan mudah diimplementasikan dengan menambahkan port Done di child FSM entity. Hal yang juga perlu diperhatikan ketika menggunakan sebuah Nested FSM adalah enable. Jika kita tidak

mendefinisikan enable, maka child FSM akan langsung berjalan walaupun opcode yang digunakan bukanlah opcode yang menggunakan child FSM tersebut. Masalah sederhana ini dapat di atas dengan menambahkan port Enable pada child FSM. Port Enable ini hanya diaktifkan ketika kita masuk ke dalam opcode yang menggunakan child FSM. Ketika selesai menggunakan child FSM atau ketika ingin melanjutkan ke tahap berikutnya, tentu kita perlu mematikan Enable. Ketika kami mencoba, Enable dimatikan ketika masuk ke tahap selanutnya. Namun kondisi tersebut membuat child FSM tidak dapat bekerja dengan baik ketika digunakan lagi pada ronde berikutnya. Maksud dari tidak dapat bekerja dengan baik adalah dibutuhkan clock cycle yang lebih banyak dibandingkan ketika pertama kali menggunakan child FSM tersebut di ronde pertama. Hal ini tentunya akan mengganggu pembuatan dari testbench. Namun kami menemukan solusi yaitu dengan mematikan Enable ketika state IDLE. Hal ini dikarenakan state IDLE adalah state selanjutnya ketika semua state yang ada pada opcode berakhir. Jadi, sewaktu child FSM selesai berproses, ia akan reset ke state awal dan Enable akan dimatikan, yang membuat child FSM berada di posisi IDLE nya sampai ia akan digunakan lagi pada ronde selanjutnya.

CHAPTER 4

CONCLUSION

Proyek Encryptor Decryptor Generator telah berhasil diimplementasikan menggunakan bahasa VHDL dengan menerapkan algoritma Advanced Encryption Standard (AES) 128-bit. Sistem ini mampu melakukan proses enkripsi dan dekripsi data berdasarkan opcode yang diberikan pengguna, di mana opcode "0000" menginisiasi proses enkripsi dan opcode "1000" menginisiasi proses dekripsi. Keseluruhan alur sistem dikendalikan oleh Finite State Machine (FSM) yang mengatur transisi antara tahapan-tahapan AES, yaitu AddRoundKey, SubBytes, ShiftRows, dan MixColumns untuk enkripsi, serta kebalikannya untuk dekripsi. Implementasi ini membuktikan bahwa konsep-konsep perancangan sistem digital dapat diterapkan secara efektif untuk membangun sistem kriptografi yang fungsional.

Tujuh modul praktikum Perancangan Sistem Digital telah berhasil diterapkan dalam proyek ini, meliputi Behavioral Style pada seluruh process statement, Structural Programming melalui instansiasi komponen pada modul utama, Looping Construct menggunakan FOR loop pada S-Box dan Inverse S-Box, Procedure untuk substitusi byte, Function untuk operasi Galois Field pada MixColumns dan key expansion, Impure Function pada Inverse S-Box yang mengakses shared variable, Finite State Machine sebagai pengendali utama sistem, Microprogramming melalui penggunaan opcode, serta Testbench untuk verifikasi fungsionalitas. Keberhasilan integrasi seluruh modul ini menunjukkan pemahaman yang komprehensif terhadap materi perkuliahan dan kemampuan menerapkannya dalam proyek nyata yang memiliki aplikasi praktis di bidang keamanan data.

REFERENCES

[1] Ashenden, P. J. "The Designer's Guide to VHDL," 3rd Edition, Morgan Kaufmann Publishers, 2008. Available: https://picture.iczhiku.com/resource/eetop/sYiEyoAUyiEkPBBb.pdf

[2] Rushton, A. "VHDL for Logic Synthesis," 3rd Edition, John Wiley & Sons, 2011. Available: https://www.r-5.org/files/books/computers/hw-layers/hardware/vhdl/Andrew_Rushton-VHDL_for_Logic_Synthesis-EN.pdf

[3] GeeksforGeeks, "Advanced Encryption Standard (AES)," *GeeksforGeeks*, Oct. 15, 2021. https://www.geeksforgeeks.org/computer-networks/advanced-encryption-standard-aes/

[4] "Cryptography - AES Key Expansion Algorithm," *Tutorialspoint.com*, 2024. https://www.tutorialspoint.com/cryptography/cryptography_aes_key_expansion_algorithm.htm

[5] tutorialspoint.com, "Advanced Encryption Standard," *www.tutorialspoint.com*, 2019. https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm

APPENDICES

Appendix A: Project Schematic

Itu merupakan hasil output sintesis jika dimuat dalam satu halaman.