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

WAR ENCRYPTION

GROUP 12

Muhammad Rasya Syahputra	**2406357803**
Reinathan Ezkhiel Kurniawan	**2406397675**
Haikal Gifari Inzaghi	**2406432261**
Fahmi Milan Amyar	**2406360060**

PREFACE

Laporan ini disusun untuk mendokumentasikan proses kelompok kami mengenai perancangan dan implementasi war encryption machine, sebuah mesin yang dirancang untuk para intel menuliskan, meneruskan maupun membaca sebuah pesan rahasia yang ditujukkan melalui waveform pada aplikasi Modelsim ataupun vivado.

Proyek ini bertujuan untuk menyediakan mekanisme yaang mempunyai keamanan tingkat tinggi, dimana menggabungkan Finite State Machine (FSM), Linear Feedback Shift Register (LFSR) untuk pembuatan kunci acak, dan decoder pola visual

Laporan ini juga menyertakan alur kerja sistem, desain arsitektur RTL, penjelasan kode yang digunakan untuk mengimplementasikan setiap fungs,, serta kesimpulan mengenai keberhasilan, dan tantangan yang dihadapi dalam mengembagkan War Encryption Machine.

Depok, December 7, 2025

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Keamanan informasi adalah aspek yang sangat vital dalam sistem pertahanan dan komunikasi di era modern. Dalam situasi yang genting dan kritis, hardware harus mampu melakukan enkripsi data dengan cepat dan memiliki mekanisme perlindungan diri dari akses yang tidak sah. Penggunaan *software* saja seringkali tidak cukup cepat dan rentan terhadap *cyber attack*. Oleh karena itu, implementasi enkripsi berbasis *n* menjadi solusi yang relevan.

Proyek ini, War Machine Encryption, mempunyai latar belakang oleh kebutuhan akan sistem autentikasi ganda yang berhubungan dengan generator kunci acak untuk mengamankan transmisi data bit. Sistem ini dirancang untuk beroperasi dalam berbagai state keamanan yang ketat

1.2 PROJECT DESCRIPTION

Proyek ini bertujuan untuk merancang dan mengimplementasikan modul VHDL yang memiliki kemampuan *Machine Learning* sederhana melalui *Pattern Matching.*

Sistem ini memiliki beberapa fitur utama yaitu Finite State Machine, Encryption Engine, Authentication System, dan Visual Pattern Decoder/ Finite State Machine berfungsi untuk mengontrol mode operasi machine *(OFF, ACTIVE, LOCK*) untuk mencegah akses yang ilegal. *Encryption Engine* yang menggunakan LFSR 16 bit untuk menghasilkan seed acak yang di XOR dengan data pengguna untuk enkripsi yang efisien. *Autentication System* yang berfungsi untuk memverifikasi *Machine ID* dan *Password* sebelum memberikan akses. Visual *Pattern Decoder* berfungsi untuk mengubah input biner menjadi pola visual (Pattern A-Z) untuk *output display*.

1.3 OBJECTIVES

The objectives of this project are as follows:

1. Merancang Finite State Machine (FSM) yang cukup kompleks untuk manajemen keamanan sistem.
2. Mengimplementasikan algoritma enkripsi berbasis XOR dan LFST menggunakan VHDL.
3. Membuat sistem *word decoder* untuk mengenali input spesifik.

1.4 ROLES AND RESPONSIBILITIES

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

Roles	Responsibilities	Person
Idea Maker	Mengkonstruksi *framework*	Reinathan
Presentation	Buat PPT menjelaskan cara kerja Kode	Rasya
PPT & Github	Membantu membuat PPT serta Github	Milan
Laporan	Menyusun laporan	Haikal

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 dan verifikasi desain VHDL.
- VSCode sebagai teks editor untuk menulis dan mengedit kode VHDL
- Quartus Prime, digunakan untuk mengkompilasi kode VHDL dan mensintesisnya menjadi desain hardware

2.2 IMPLEMENTATION

State Machine Control menggunakan tipe data enumerasi **t_State** (OFF, HOLD, ACTIVE, UP, PASSIVE, LOCK, *Self destruction*). Transisi antar state diatur oleh sinyal clock dan input validasi. Contohnya, sistem akan masuk ke mode **LOCK** jika berada di **HOLD** selama lebih dari 30 detik (**TIME_30S**) tanpa aktivitas yang valid.

Cipher adalah metode atau seperangkat aturan untuk mengubah data yang bisa dibaca (plaintext) menjadi data yang dikodekan (ciphertext), dan sebaliknya. Proses pengubahan plaintext menjadi ciphertext disebut **enkripsi**, sedangkan mengembalikannya ke plaintext disebut **dekripsi**. Untuk melakukan transformasi tersebut, cipher menggunakan **kunci (key)** yang hanya diketahui oleh pihak yang berwenang agar data tetap rahasia.

LFSR (*Linear Feedback Shift Register*) adalah jenis **shift register** (register geser) di mana bit input berikutnya merupakan fungsi linear dari keadaan (state) sebelumnya. Fungsi linear yang paling umum digunakan adalah gerbang logika **XOR** (Exclusive-OR). mempunyai fungsi **lfsr_next** yang berfungsi mengimplementasikan polinomial umpan balik pada bit ke-15, 13, 12, dan 10. Ini menghasilkan urutan pseudo random yang digunakan sebagai **KeyStore**.

Message Storage & Encryption, data disimpan dalam array **MsgStore**. Proses enkripsi terjadi pada state **Up**, dimana **UserBits** di XOR dengan **next_seed**. Hasil enkripsi dan kuncinya kemudian dicatat ke dalam file eksternal **write_log.txt** menggunakan fitur *textio* di VHDL.

Pada Pattern **Generation**, **PatternOf** berfungsi untuk memetakan *opcode* 5 bit ke dalam *array* pola visual 16 bit seperti A_PATTERN, B_PATTERN, Dll. ini memungkinkan sistem menampilkan huruf atau simbol pada *output* 16 bit (*BBit0 - BBit15*) secara berurutan saat mode **READ** aktif.

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Sejauh ini kami melakukan uji coba dengan berfokus pada penjalanan *finite-state-machine* pada kode utama. Meskipun begitu, karena ide kami adalah menggunakan **key** pada **read_mode** membuat *testbench* harus bisa men-*lookup* apa saja yang dibutuhkannya untuk melakukan **read**.

Selain itu, untuk merangkum pemrograman yang lebih modular, kami juga menambahkan **package** yang berisi *pattern* huruf. Kami berencana untuk membuat *testbench* yang bisa mengetes semua, namun sayangnya hal itu belum terealisasikan. Oleh karena itu *testbench* yang dibuat hanya menguji satu dari beberapa hal yang memungkinkan pada kode.

Test yang kami buat pada dasarnya menguji kemampuan mesin dari uji keamanan terendah (**OFF**) hingga mencapai **write_mode** kemudian menuliskan kata **F I N P R O** dan *testbench* akan memanggil dirinya sendiri untuk melakukan proses baca.

3.2 RESULT

Awalnya kami mengkonstruksi kode yang akan mengecek *testcase* mesin ini dengan mengenkapsulasi beberapa skenario seperti **password salah**, **ID salah**, **menulis**, dsbg. , namun hal - hal tersebut tidak lengkap karena hasil tidak ditulis ke dalam **txt** dan **lsfr** tidak bekerja.

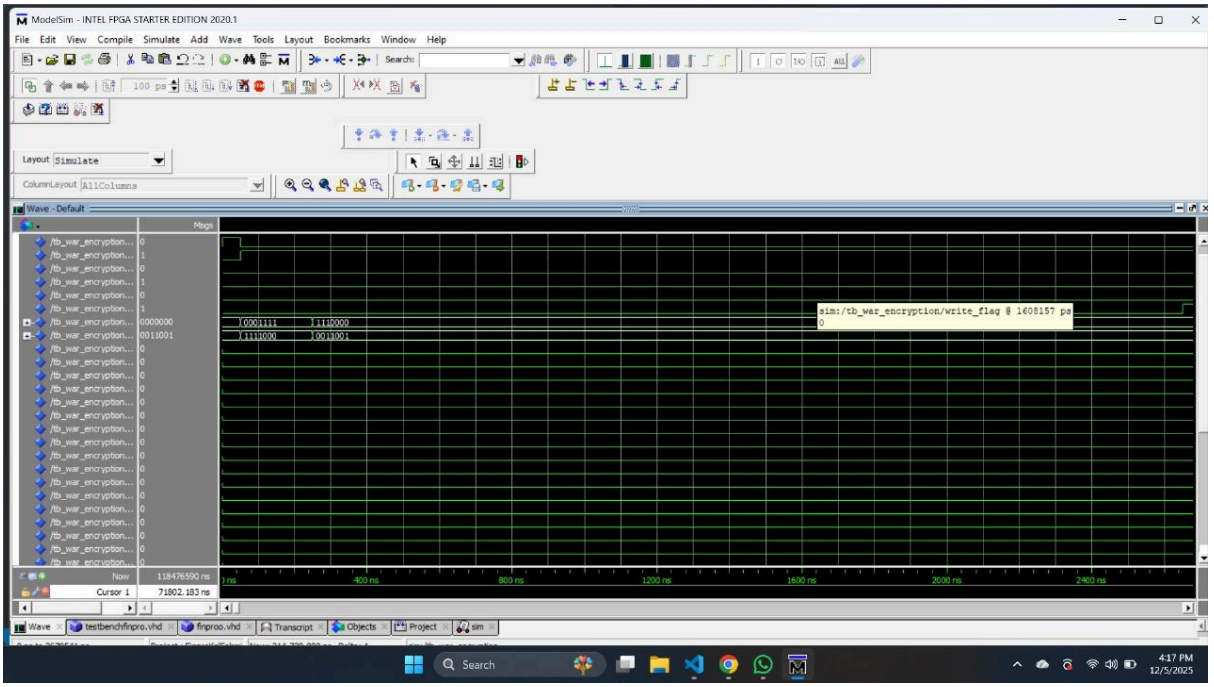


Fig 1. Hasil *test* pertama (*sederhana*, **mock up**)

Oleh karena itu, kami me**refactor** kode kami tapi sayangnya *testbench* yang utuh masih saja belum bisa terealisasikan karena kami terpaksa untuk membuat **mock-up keys** jadi *keystore* menggunakan sintesis manual pada *testbench* (**tidak dari DUT**).

```
#           File in use by: Reinathan ZK  Hostname: WINDOWS-5UL941G  ProcessID: 10108
#           Attempting to use alternate WLF file "./wlftnmtvwv".
# ** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
#           Using alternate file: ./wlftnmtvwv
VSIM 24> run -all
# ** Note: Feeding keystream index=0
#    Time: 1020 ns  Iteration: 0  Instance: /tb_war_encryption
# ** Note: Feeding keystream index=1
#    Time: 1520 ns  Iteration: 0  Instance: /tb_war_encryption
# ** Note: Feeding keystream index=2
#    Time: 2020 ns  Iteration: 0  Instance: /tb_war_encryption
# ** Note: Feeding keystream index=3
#    Time: 2520 ns  Iteration: 0  Instance: /tb_war_encryption
# ** Note: Feeding keystream index=4
#    Time: 3020 ns  Iteration: 0  Instance: /tb_war_encryption
# ** Note: Feeding keystream index=5
#    Time: 3520 ns  Iteration: 0  Instance: /tb_war_encryption
# ** Note: Feeding keystream index=6
#    Time: 4020 ns  Iteration: 0  Instance: /tb_war_encryption
# ** Note: Post-write (direct) populated, total keys=7
#    Time: 4620 ns  Iteration: 0  Instance: /tb_war_encryption
# ** Note: FINISHED TESTBENCH RUN
#    Time: 4620 ns  Iteration: 0  Instance: /tb_war_encryption
```

Fig 2. *Transcript* dari tes kedua

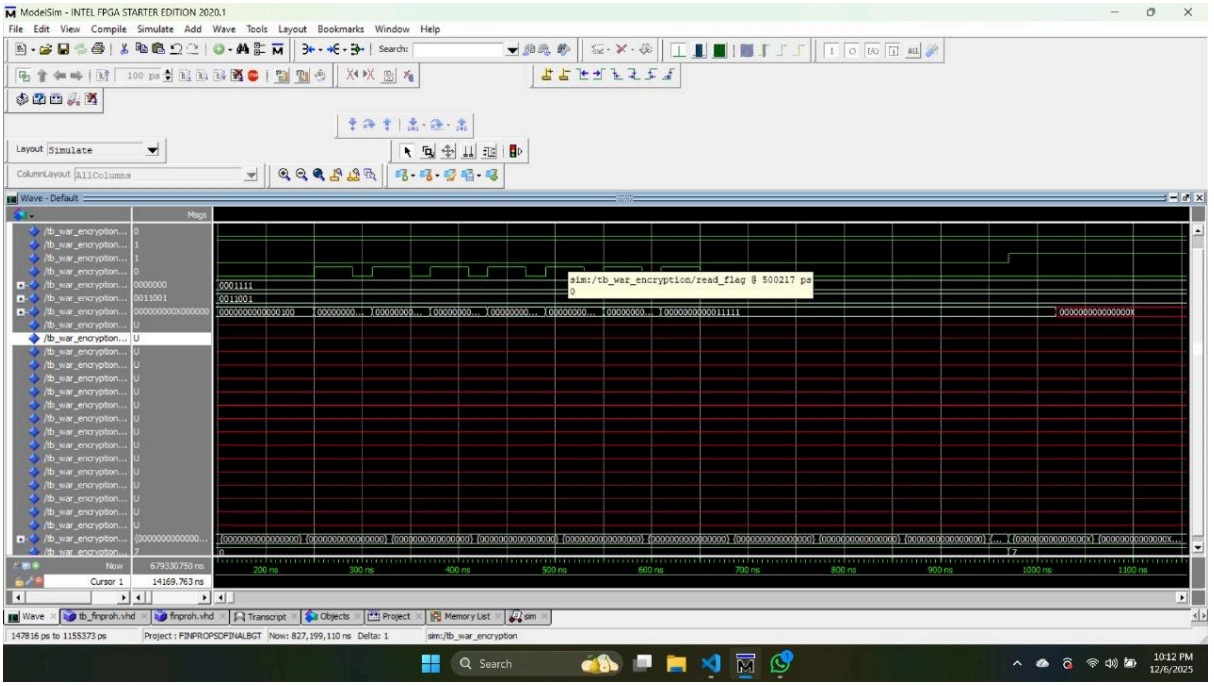


Fig 3. *Waveform* dari tes kedua, **BBits** tidak memakai *placeholder* dan *keystore* masih **null**

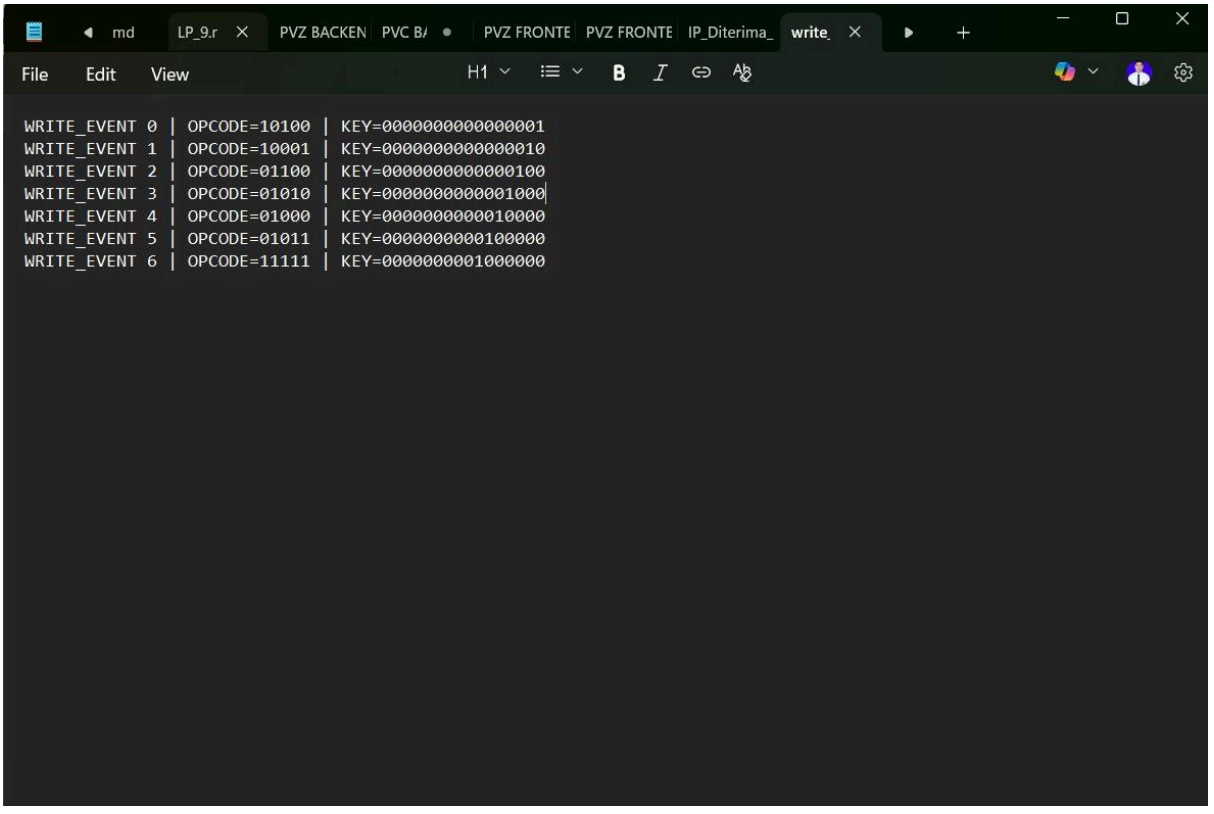


Fig 4. **write_log.txt** dari percobaan ke -2

Ini adalah hasil terbaru kami, **keystore masih tetap saja null.** Dalam *test* yang terbaru, kita sudah melihat jelas pada saat mesin melakukan tahap tulis, mesin berproses secara lancar.

3.3 ANALYSIS

Hal ini terjadi karena **read mode merupakan proses yang lebih kompleks**. Uji *read mode* mengharuskan *testbench* untuk *file read process* dan meninjau elemen **read_auth**. Kemampuan dari *testbench* untuk mengakses komponen pada DUT sangat diperlukan (terutama **keystore**). Oleh sebab itu, kami memutuskan untuk memisahkan **constant array** daripada pola huruf menjadi satu *file* terpisah yaitu **packagepattern.**

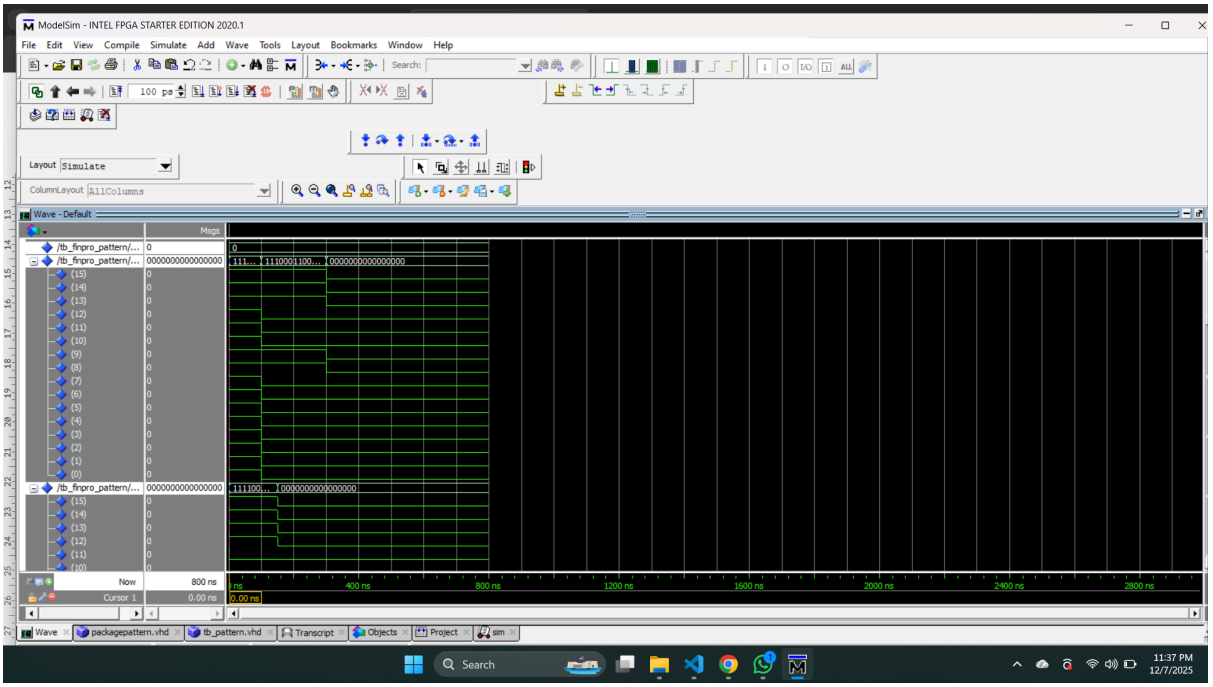


Fig 5. Tes *pattern* “ F I N P R O” -1

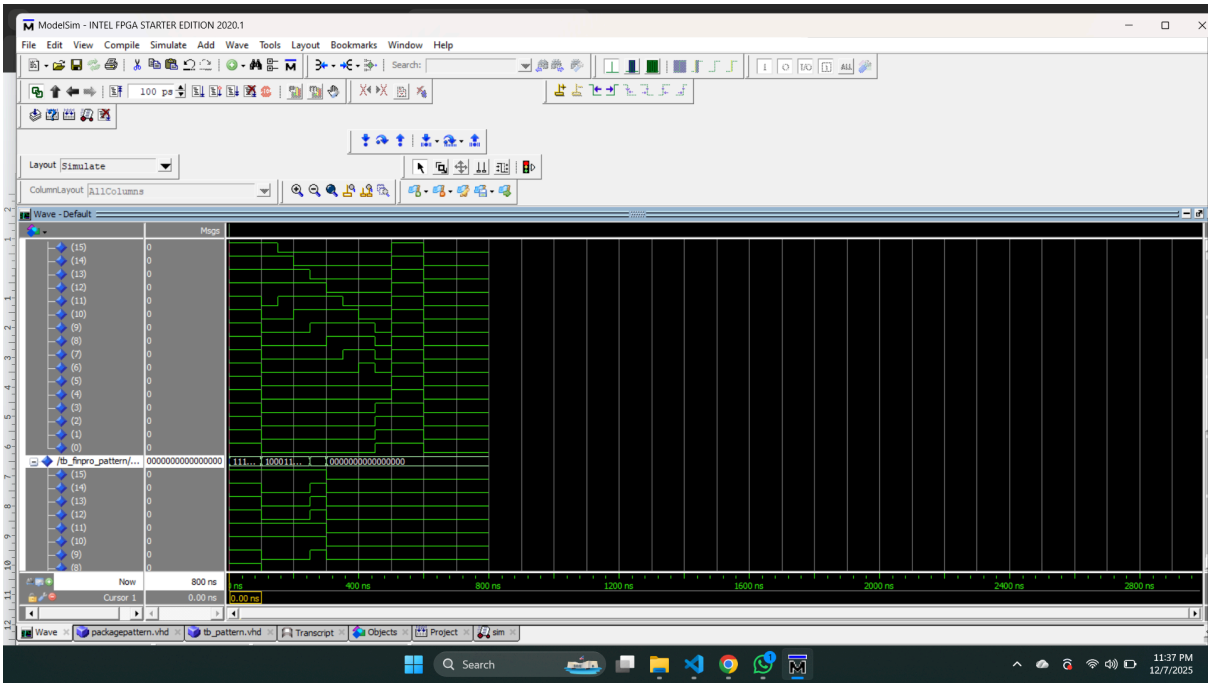


Fig 6. Tes *pattern* " F I N P R O" -2

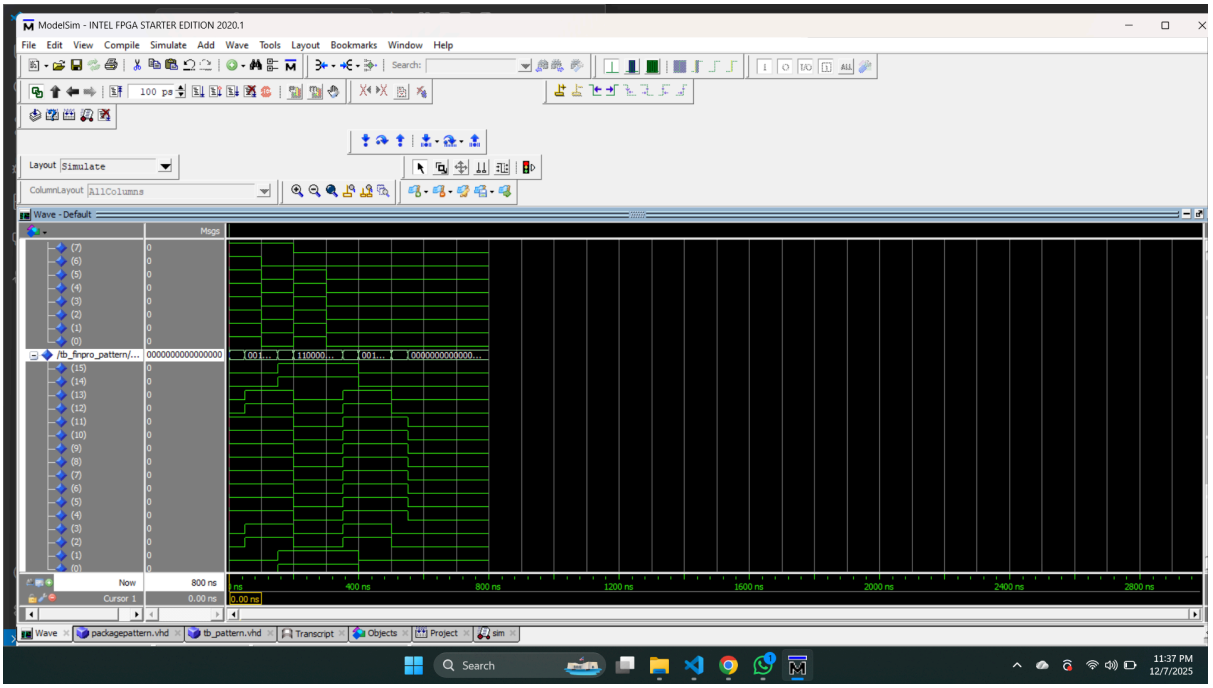


Fig 7. Tes *pattern* " F I N P R O" -3

Hal ini membuat *testbench* harus mengimpor dua **work** yang berbeda sehingga tes bisa dilakukan secara menyeluruh (mengetes alur *FSM* secara lengkap sembari menampilkan **Pattern** yang ditulis secara lengkap).

CHAPTER 4

CONCLUSION

Berdasarkan dari perancangan, implementasi, dan pengujian yang telah dilakukan pada proyek *War Machine Encryption*, program masih bersifat **dalam pengembangan**. Hal ini dikarenakan pengujian masih bersifat modular. Pada *github*, kami menginisiasikan 4 *file* yang mana merupakan tes individual **pattern** dan **FSM**. Oleh karena itu, sejauh ini belum terdapat hasil *testing* yang pasti yang dapat diakumulasikan sebagai hasil akhir proyek kami.

REFERENCES

[1] Russell, "Linear Feedback Shift Register for FPGA," *Nandland*, Jun. 09, 2022. https://nandland.com/lfsr-linear-feedback-shift-register/

[2] GeeksforGeeks, "Linear Feedback Shift Registers (LFSR)," *GeeksforGeeks*, Sep. 10, 2024. https://www.geeksforgeeks.org/digital-logic/linear-feedback-shift-registers-lfsr/

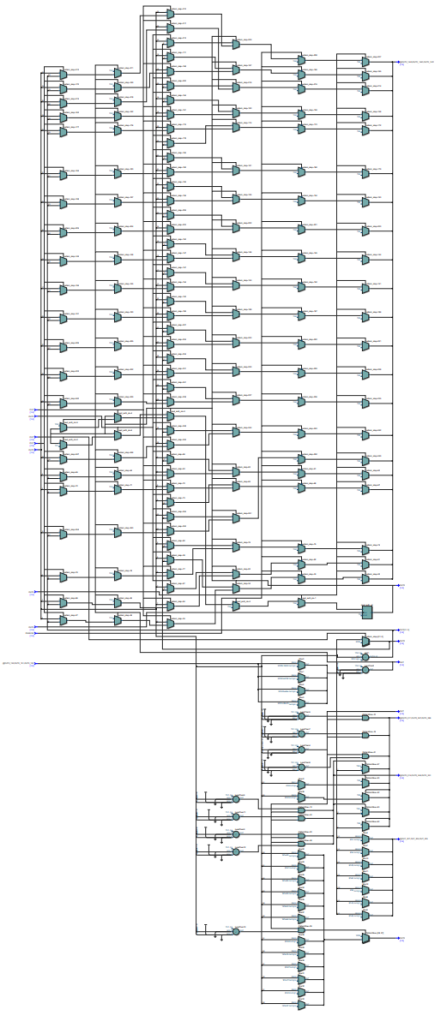
[3] Tutorialspoint, "Digital Electronics - Finite State Machines," *Tutorialspoint.com*, 2024. https://www.tutorialspoint.com/digital-electronics/digital-electronics-finite-state-machines.htm

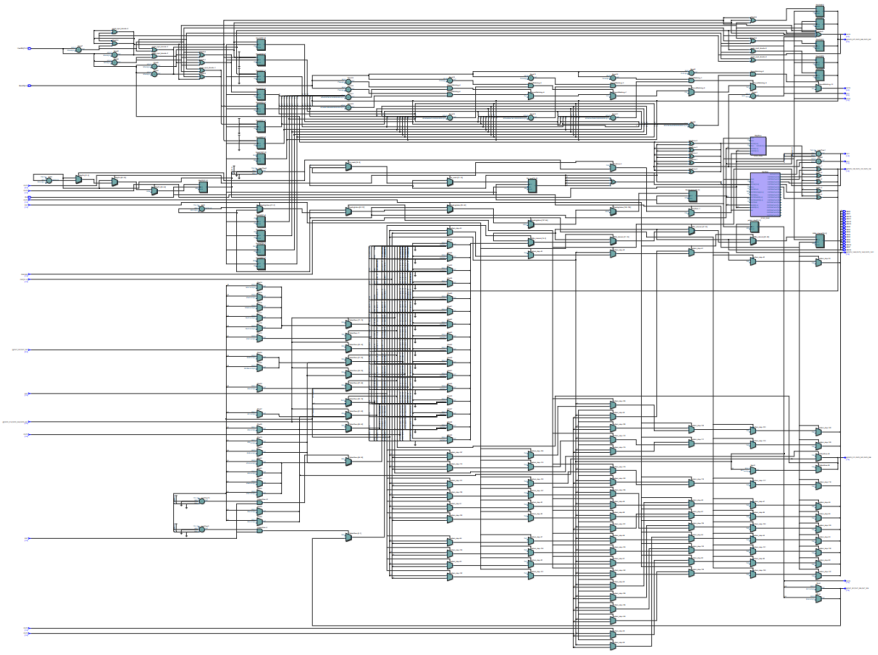
[4] GeeksforGeeks, "*What is Cipher?*," *GeeksforGeeks*, 2023. [Online]. Available: https://www.geeksforgeeks.org/computer-networks/what-is-cipher/ . [Accessed: 07-Dec-2025].

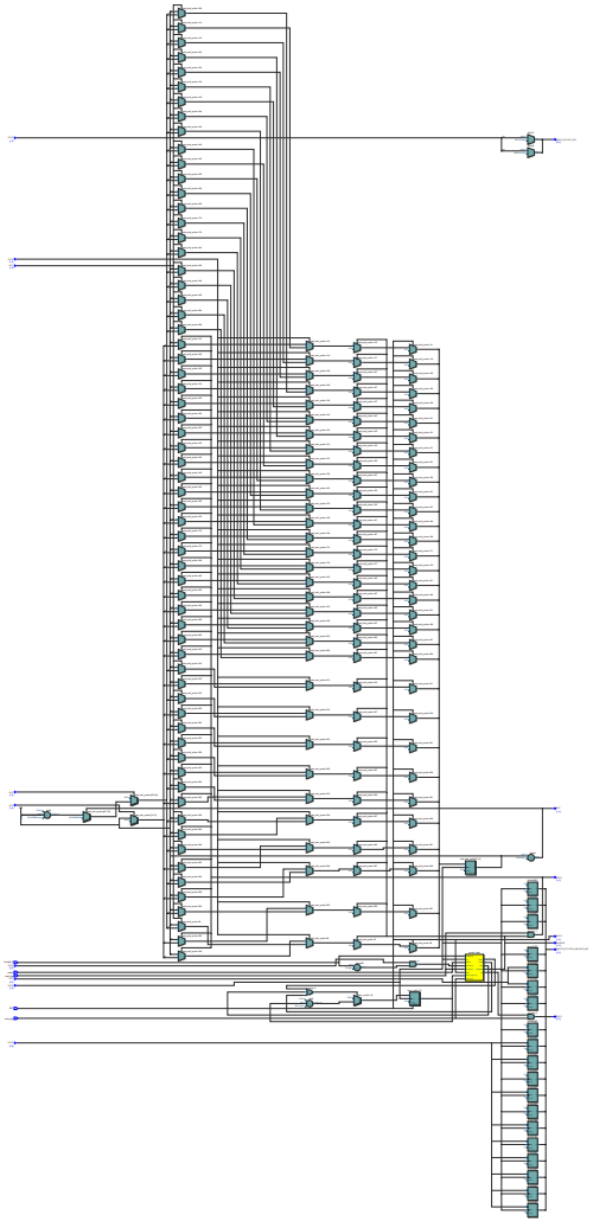
[5] GeeksforGeeks, "What is Bitmasking," *GeeksforGeeks*, Mar. 02, 2023. https://www.geeksforgeeks.org/dsa/what-is-bitmasking/

APPENDICES

Appendix A: Project Schematic







Appendix B: Documentation