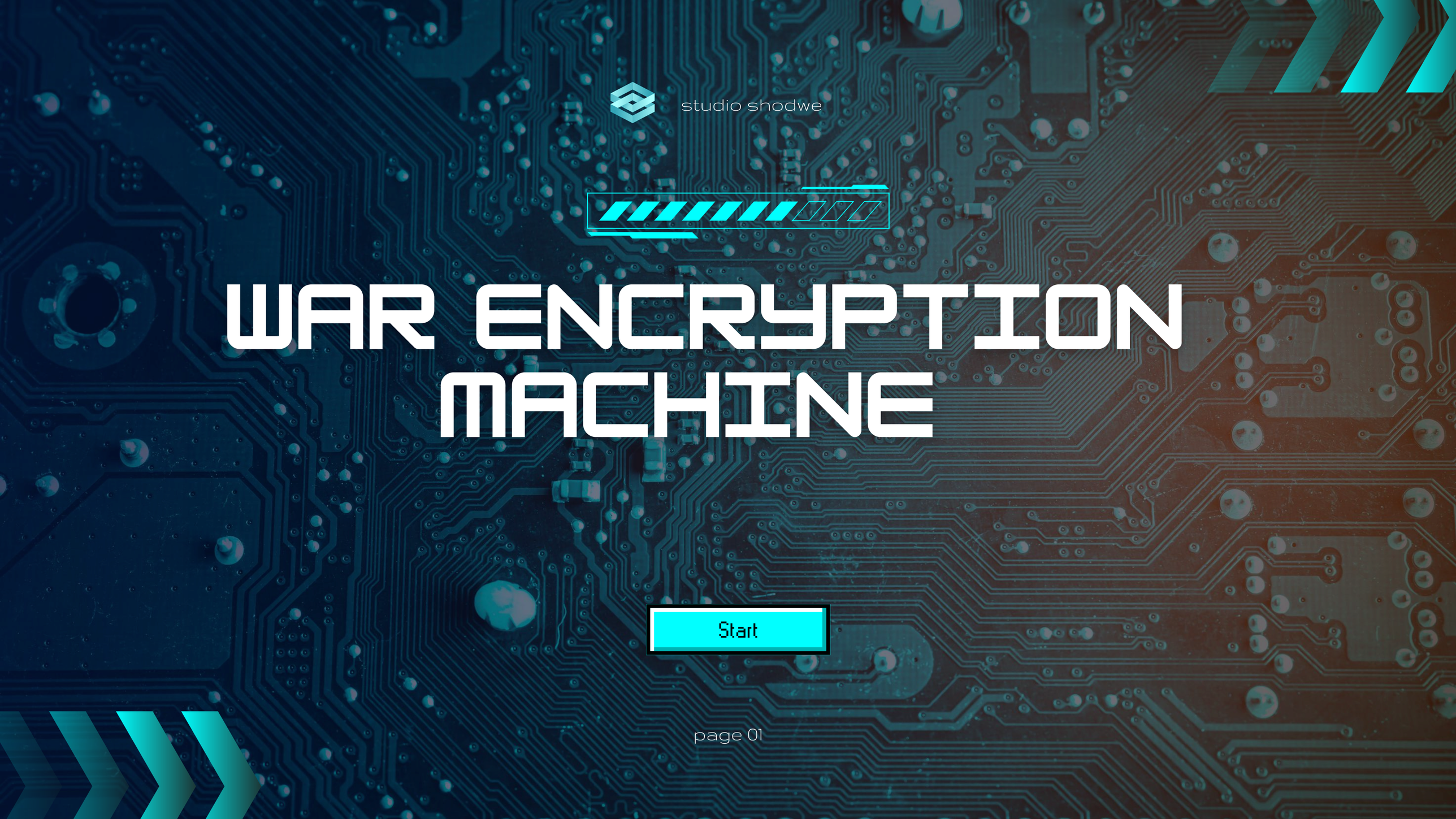
studio shodwe
WAR ENCRYPTION MACHINE
Start
page 01

ANGGOTA KELOMPOK

Reinathan Ezkhiel Kurniawan 2406397675

Haikal Gifari Inzaghi 2406432261

Fahmi Milan Amyar - 2406360060

Muhammad Rasya Syahputra - 2406357803

DEKRIPSI SINGKAT

War Encryption Machine di*desain* secara khusus untuk para intel menuliskan / meneruskan / membaca sebuah **pesan** rahasia yang ditunjukkan melalui **waveform** pada **Modelsim / Vivado**. Untuk mengakses pesan tersebut, pengguna harus melalui **proses autentikasi** yang bertahap dan terdapat konsekuensi yang akan didapat apabila proses tersebut gagal.

DEFINISI ENTITY

```
8  entity war_encryption is
9      port (
10         CLK        : in  std_logic;
11         RST        : in  std_logic;
12         enable     : in  std_logic;
13         read_flag  : in  std_logic;
14         write_flag : in  std_logic;
15
16         Pswd       : in std_logic_vector(6 downto 0);
17         MachID     : in std_logic_vector(6 downto 0);
18         UserBits   : in std_logic_vector(15 downto 0);
19
20         BBit0  : out std_logic; BBit1  : out std_logic;
21         BBit2  : out std_logic; BBit3  : out std_logic;
22         BBit4  : out std_logic; BBit5  : out std_logic;
23         BBit6  : out std_logic; BBit7  : out std_logic;
24         BBit8  : out std_logic; BBit9  : out std_logic;
25         BBit10 : out std_logic; BBit11 : out std_logic;
26         BBit12 : out std_logic; BBit13 : out std_logic;
27         BBit14 : out std_logic; BBit15 : out std_logic
28     );
29 end entity;
```

CLK sebagai variabel pengatur waktu, **RST** untuk mereset mesin war_encryption ke keadaan awal, **enable** untuk memajukan state ke ACTIVE dari keadaan tertentu, misalnya dari OFF atau HOLD. **read_flag** dan **write_flag** digunakan untuk menentukan kegiatan apa yang ingin dilakukan pengguna.

Pswd dan **MachID** merupakan inputan dari user yang merupakan bagian dari sistem keamanan, dimana **Pswd** harus sesuai dan **MachID** harus bernilai valid dan sama dengan pasangan kata yang dihasilkan **UserBits**

UserBits sendiri merupakan input user yang bersifat serbaguna berdasarkan keadaan dari FSM dalam saat tertentu.

BBIT

```
BBit0  : out std_logic; BBit1  : out std_logic;
BBit2  : out std_logic; BBit3  : out std_logic;
BBit4  : out std_logic; BBit5  : out std_logic;
BBit6  : out std_logic; BBit7  : out std_logic;
BBit8  : out std_logic; BBit9  : out std_logic;
BBit10 : out std_logic; BBit11 : out std_logic;
BBit12 : out std_logic; BBit13 : out std_logic;
BBit14 : out std_logic; BBit15 : out std_logic
```

BBit 1 - 15 merupakan bit yang akan menjadikan *output* pada *waveform*. Bit ini akan merepresentasikan **array pattern masing - masing huruf** pada program.

```
signal BBits_internal : std_logic_vector(15 downto 0);
```

```
TempBit0  : out std_logic;
TempBit1  : out std_logic;
TempBit2  : out std_logic;
TempBit3  : out std_logic;
TempBit4  : out std_logic;
TempBit5  : out std_logic;
TempBit6  : out std_logic;
TempBit7  : out std_logic;
TempBit8  : out std_logic;
TempBit9  : out std_logic;
TempBit10 : out std_logic;
```

TEMPBIT

TempBit 0 - 15 menjadi keluaran ketika pengguna menuliskan satu persatu huruf maka progres tersebut akan disimpan nilainya dalam **tempbit**.

STATE FSM

State yang ada adalah : **OFF** ketika pengguna pertama kali masuk ke dalam *machine.* **HOLD** adalah ketika pengguna sudah masuk tetapi salah mengetik *password* atau ketika waktu habis. **ACTIVE** adalah ketika pengguna berhasil login dan dapat memilih ingin masuk ke **write** (**UP**) atau hanya men-**read** (**PASSIVE**), **LOCK** terjadi ketika *password* yang dimasukkan gagal, maka *user* akan dibekukan sementara dan diberikan satu kesempatan kali lagi untuk mengetikan *password.* **SelfDes** merupakan kondisi setelah **LOCK** dimana mesin tidak dapat diproses untuk memasukkan apa - apa lagi setelah itu. **RESET** adalah ketika pengguna berada dalam *state* selain **LOCK** dan **SelfDes** untuk mengembalikan mesin dalam keadaan semula.

```
-------------------------------------------------------------------
--   STATE MACHINE
-------------------------------------------------------------------
type t_State is (OFF, HOLD, ACTIVE, UP, PASSIVE, LOCK, SelfDes);
signal current_state : t_State := OFF;
signal NextState     : t_State;

signal en : std_logic;
signal timer_cnt : integer := 0;

constant CLK_FREQ : integer := 50_000_000;
constant TIME_30S : integer := CLK_FREQ * 30;
constant TIME_60S : integer := CLK_FREQ * 60;
```

STATE FSM : OFF

```
when OFF =>
    if enable='1' then
        if RecordMatching(MachID,Wordz) and Match_UP_Pass(Pswd) then
            NextState <= ACTIVE;
        else
            NextState <= HOLD;
        end if;
    end if;
    if timer_cnt>=TIME_60S then
        NextState <= HOLD;
    end if;
```

Pada **OFF** maka mencocokan **ID** dan **UniqueWords**. Apabila *password* benar maka lanjut ke **ACTIVE**, jika tidak **HOLD**. Apabila tidak memasukkan *password* setelah 60 detik maka **HOLD**.

STATE FSM : RESET

```
-------------------------------------------------------------------
-- STATE MACHINE TIMER + UPDATE
-------------------------------------------------------------------
process(CLK,RST)
begin
    if RST='1' then
        current_state <= OFF;
        timer_cnt <= 0;
    elsif rising_edge(CLK) then
        if current_state=OFF or current_state=HOLD or current_state=LOCK then
            timer_cnt <= timer_cnt + 1;
        else
            timer_cnt <= 0;
        end if;

        current_state <= NextState;
    end if;
end process;
```

CurrentState menjadi **OFF** dan **timer count** akan menjadi 0 kembali seperti awal.

STATE FSM : ACTIVE

```
when ACTIVE =>
    if write_flag='1' then
        if Match_UP_Pass(Pswd) then
            NextState <= UP;
        end if;
    elsif read_flag='1' then
        NextState <= PASSIVE;
    end if;

when UP =>
    if Match_Exit_Pass(Pswd) then
        NextState <= ACTIVE;
    end if;

when PASSIVE =>
    if Match_Exit_Pass(Pswd) then
        NextState <= ACTIVE;
    end if;
    if read_flag='0' then
        NextState <= ACTIVE;
    end if;
```

Pada *state* ini pengguna dapat memasukkan **write Pwd** dan masuk ke **UP.** Saat di **UP** , pengguna bisa memasukkan **Exit Pwd** untuk ke mode *read* (**PASSIVE**).

STATE FSM : PASSIVE & SELFDES

```
when PASSIVE =>
    if Match_Exit_Pass(Pswd) then
        NextState <= ACTIVE;
    end if;
    if read_flag='0' then
        NextState <= ACTIVE;
    end if;
```

```
when SelfDes =>
    NextState <= SelfDes;
```

Dalam keadaan **PASSIVE** , FSM akan tetap berada pada state tersebut sampai user berhasil memasukkan password keluar, atau User pindah keluar dari mode membaca.

Dalam keadaan **SelfDes**, FSM dianggap sudah hancur sehingga tidak dapat pindah ke state lain sama sekali.

STATE FSM : HOLD

```
when HOLD =>
    if enable='1' then
        if RecordMatching(MachID,Wordz) and Match_UP_Pass(Pswd) then
            NextState <= ACTIVE;
        else
            NextState <= HOLD;
        end if;
    end if;
    if timer_cnt>=TIME_30S then
        NextState <= LOCK;
    end if;
```

Mesin yang berada di *state* **HOLD** maka akan diberikan kesempatan untuk memasukkan *password* . Namun apabila masih saja gagal, dan timer sudah mencapai 30 detik, maka FSM akan lanjut ke state **LOCK**.

STATE FSM : LOCK

```
when LOCK =>
    if timer_cnt>=TIME_60S then
        if RecordMatching(MachID,Wordz) and Match_UP_Pass(Pswd) then
            NextState <= ACTIVE;
        else
            NextState <= SelfDes;
        end if;
    end if;
```

Saat di **LOCK , pengguna akan dibekukan sementara selama 60 Seconds**. Sehabis itu, akan diberikan kesempatan satu kali lagi untuk memasukkan *password.* Apabila benar, maka akan masuk ke ACTIVE state, namun apabila gagal maka FSM akan self-destruct.

PASSWORD

Saat kita ingin masuk ke **UP** maka ada *password*nya yaitu **Write_Pwd.** Ketika kita ingin keluar dari **UP** maka ada **exit** *password*.

```
------------------------------------------------------------------
-- PASSWORD CHECK
------------------------------------------------------------------
constant Write_Pwd : std_logic_vector(6 downto 0) := "0001111";
constant Exit_Pwd  : std_logic_vector(6 downto 0) := "0000000";

function Match_UP_Pass(P : std_logic_vector(6 downto 0)) return boolean is
begin return P = Write_Pwd; end;
function Match_Exit_Pass(P : std_logic_vector(6 downto 0)) return boolean is
begin return P = Exit_Pwd; end;
```

ID AND UNIQUE WORDS

Setiap mesin memiliki *string* **unique word** masing - masing. **ID** dan **Word** harus sesuai sehingga permintaan pengguna bisa diproses, apabila beda maka ditolak dan mesin masuk ke state **HOLD**

```
-----------------------------------------------------------------------
--  RECORD MATCHING
-----------------------------------------------------------------------
function RecordMatching(ID : std_logic_vector(6 downto 0); W : string) return boolean is
begin
    if    (ID="0011001" and W="tango     ") then return true;
    elsif (ID="0011010" and W="alpha     ") then return true;
    elsif (ID="0011011" and W="delta     ") then return true;
    elsif (ID="1111000" and W="omega     ") then return true;
    elsif (ID="0101010" and W="gorga     ") then return true;
    elsif (ID="1010100" and W="fahmi     ") then return true;
    else return false;
    end if;
end function;
```

ID AND UNIQUE WORDS : INPUT

```
-----------------------------------------------------------------------
-- MAIN DECODE WORDZ
-----------------------------------------------------------------------


process(CLK)
begin
    if rising_edge(CLK) then
        Wordz <= words_input_decode(
            UserBits(0),UserBits(1),UserBits(2),UserBits(3),
            UserBits(4),UserBits(5),UserBits(6),UserBits(7),
            UserBits(8),UserBits(9),UserBits(10),UserBits(11),
            UserBits(12),UserBits(13),UserBits(14),UserBits(15)
        );
    end if;
end process;
```

Pengguna akan memasukkan **unique words** dalam bentuk bit (**UserBits**) dengan panjang 16 yang akan disesuaikan dengan fungsi **words_input_decode**.

ID AND UNIQUE WORDS : INPUT(2)

```
-------------------------------------------------------------------
--  WORD DECODER
-------------------------------------------------------------------
function words_input_decode(
    b0,b1,b2,b3,b4,b5,b6,b7 : std_logic;
    b8,b9,b10,b11,b12,b13,b14,b15 : std_logic
) return string is
    variable concat : std_logic_vector(15 downto 0);
begin
    concat := b0&b1&b2&b3&b4&b5&b6&b7&b8&b9&b10&b11&b12&b13&b14&b15;

    case concat is
        when "0000000000000100" => return "tango     ";
        when "0000000000000110" => return "delta     ";
        when "0000000000000111" => return "alpha     ";
        when "0000000000000101" => return "omega     ";
        when "0000000000001111" => return "gorga     ";
        when "0000000000001101" => return "fahmi     ";
        when others             => return "tidaktau  ";
    end case;
end function;
```

Pengguna yang sudah memasukkan **unique words** maka Bit tersebut akan di **concatenated** lalu di **test case** dengan *unique words* yang tertera. Selanjutnya akan diproses dengan fungsi **RecordMatching.**

MEKANISME PESAN : PENULISAN

```
-------------------------------------------------------------------
-- WRITE MODE: ENCRYPT + STORE + PRINT
-------------------------------------------------------------------
begin
process(CLK)
    variable L : line;
    variable op_str  : string(1 to 5);
    variable key_str : string(1 to 16);
    variable next_seed : std_logic_vector(15 downto 0);
begin
    if rising_edge(CLK) then
        if RST = '1' then
            MsgPtr <= 0;
        elsif write_flag = '1' and current_state = UP then
            if MsgPtr < MAX_MSG then
                -------------------------------------------------------------------
                -- UPDATE LFSR
                -------------------------------------------------------------------
                -- compute next seed into a variable so we can store the updated key
                next_seed := lfsr_next(lfsr_seed);
                lfsr_seed <= next_seed;
```

Setelah login dan masuk state UP dengan password admin yang benar, jika **write_flag** aktif dan memori belum penuh, mesin mengambil 5 bit terbawah **UserBits** sebagai Opcode. Bersamaan dengan itu, LFSR menghasilkan kunci acak 16-bit baru.

studio shodwe

MEKANISME PESAN : PENULISAN (2)

```
-------------------------------------------------------------------
-- ENCRYPT
-------------------------------------------------------------------
MsgStore(MsgPtr) <= UserBits(4 downto 0) xor next_seed(4 downto 0);
KeyStore(MsgPtr) <= next_seed;

-------------------------------------------------------------------
-- PREP STRINGS FOR TXT
-------------------------------------------------------------------
for i in 1 to 5 loop
    if UserBits(4-(i-1))='1' then op_str(i):='1';
    else                          op_str(i):='0';
    end if;
end loop;

for i in 1 to 16 loop
    if next_seed(16-i)='1' then key_str(i):='1';
    else                        key_str(i):='0';
    end if;
end loop;
```

Mesin kemudian mengenkripsi Opcode tersebut melalui operasi XOR dengan 5 bit kunci LFSR. Hasilnya disimpan ke **MsgStore**, sedangkan kunci utuh 16-bit disimpan ke **KeyStore**. Representasi biner keduanya dicatat ke file **write_log.txt**. Terakhir, benih LFSR diperbarui dan pointer memori digeser maju untuk pesan berikutnya.

```
file LogFile : text open write_mode is "write_log.txt";
```

```
-----------------------------------
-- PRINT TO TXT
-----------------------------------
write(L, string'("WRITE_EVENT "));
write(L, MsgPtr);
write(L, string'(" | OPCODE="));
write(L, op_str);
write(L, string'(" | KEY="));
write(L, key_str);
writeline(LogFile, L);
```

MEKANISME PESAN : PENYIMPANAN

```
-----------------------------------------------------------------
--  MESSAGE STORAGE
-----------------------------------------------------------------
constant MAX_MSG : integer := 16;

type MsgArray is array(0 to MAX_MSG-1) of std_logic_vector(4 downto 0);
type KeyArray is array(0 to MAX_MSG-1) of std_logic_vector(15 downto 0);

signal MsgStore : MsgArray := (others => (others=>'0'));
signal KeyStore : KeyArray := (others => (others=>'0'));
signal MsgPtr : integer := 0;
```

Mesin ini hanya dapat menyimpan sebanyak 16 pesan saja, dimana pesan ini disimpan di dalam MsgStore yang bertipe data MsgArray, yang merupakan list 16 slot berukuran 5 bit yang berisi text yang sudah terenkripsi. Sementara KeyStore yang bertipe data KeyArray menyimpan kunci buatan LFSR yang dapat digunakan untuk mendekripsi pesan pada MsgStore. MsgPtr berfungsi menunjukkan slot mana yang sedang kita isi dengan pesan .

MEKANISME PESAN : PEMBACAAN

```
signal ReadingIndex : integer := 0;
signal read_auth_ok : std_logic := '0';
signal read_wait_cycles : integer := 0;
signal pattern_step : integer := 0;
subtype PatternStep is std_logic_vector(15 downto 0);
type PatternArray is array (natural range <>) of PatternStep;
```

Sinyal - sinyal ini bertugas mengatur logika mesin ketika berada pada reading mode (**PASSIVE** state). Pertama, sinyal **read_auth_ok** dicek nilainya, apabila 1, maka logika mesin ditujukan untuk membaca pesan. **ReadingIndex** menunjukan pesan mana pada MsgStore yang sedang kita baca sekarang. **read_wait_cycles** digunakan sebagai penunda waktu, agar animasi tampilan huruf dapat dilihat mata manusia. Terakhir, ada **pattern_step** yang menandai baris ke berapa dari pola huruf A-Z yang sedang ditampilkan di LED saat ini.

MEKANISME PESAN : VISUALISASI

```
subtype PatternStep is std_logic_vector(15 downto 0);
type PatternArray is array (natural range <>) of PatternStep;
```

Kedua variabel ini digunakan untuk mencetak huruf ke LED pada proses pembacaan. Terdapat dua variabel, yang pertama **PatternStep**, yang merupakan subtype yang mendefinisikan bahwa satu baris dari pattern huruf memiliki 16 bit. yang kedua ada **PatternArray**, yang mendefinisikan bahwa sebuah pattern dari huruf adalah kumpulan dari beberapa baris bertipe **PatternStep,** dimana tinggi hurufnya bisa bebas tergantung masing-masing huruf.

MEKANISME CLOCK

```
signal timer_cnt : integer := 0;

constant CLK_FREQ : integer := 50_000_000;
constant TIME_30S : integer := CLK_FREQ * 30;
constant TIME_60S : integer := CLK_FREQ * 60;
```

Frekuensi Clock diatur ke **50 Mhz**, Maka perhitungan **Second** pada mesin ini mengacu ke **CLK_FREQ** (kelipatannya). **timer_cnt** adalah semacam stopwatch yang alan bertambah 1 setiap kali sinyal **CLK** berdetak. **TIME_30S** adalah sebuah variabel yang digunakan untuk membatasi waktu sampai 30 detik, dan digunakan untuk membatasi waktu sebuah mesin berada dalam state **HOLD** tanpa ada aktivitas sama sekali. Terakhir, ada **TIME_60S**, yang membatasi waktu sampai 60 detik, dan digunakan untuk membekukan user saat pertama kali masuk **LOCK** state.

SMART CITIES

1.

Energy Efficiency

Optimizing energy use through smart grids and sustainable infrastructure.

2.

Public Transportation

Enhancing transit systems with real-time data and autonomous vehicles.

3.

Waste Management

Implementing smart waste collection and recycling systems for cleaner cities.

RENEWABLE ENERGY

1. **Solar Power**

Utilizing photovoltaic panels to convert sunlight into electricity, providing a clean and renewable energy source that reduces carbon emissions.

2. **Wind Turbines**

Capturing wind energy through large turbines to generate electricity, contributing to a sustainable energy mix and decreasing reliance on fossil fuels.

VIRTUAL REALITY

Virtual Reality (VR) is extending its reach beyond gaming into areas like education, healthcare, and entertainment. By creating immersive, interactive experiences, VR has the potential to revolutionize how we learn, train, and even treat medical conditions. As the technology continues to improve, VR is expected to become an integral part of various industries, offering new ways to engage and experience the world.

SPACE EXPLORATION

Space exploration is entering a new era, driven by advances in technology and a renewed interest in reaching beyond our planet. From ambitious missions to Mars to the development of space tourism, these efforts are expanding our understanding of the universe and pushing the boundaries of what humanity can achieve. The future of space exploration holds the promise of new discoveries and the potential for human settlement on other planets.

THANK YOU!

As we look to the future, these technologies will play a crucial role in shaping a better world for generations to come.