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

SMART OFFICE LOCKING DOOR

GROUP B3

Brian Yudha Sandi	**STUDENT ID 1**
Cecilia Inez Reva M.	**2106636994**
Fayza Nirwasita	**2106635700**
Zefanya Christira Deardo	**2106637214**

PREFACE

Puji syukur atas kehadirat Tuhan Yang Maha Esa karena berkat dan rahmat-Nya, kami dapat menyusun laporan praktikum. Laporan ini disusun dengan tujuan implementasi pemrograman VHDL dapat diterapkan dalam masalah dan kehidupan sehari-hari. Laporan ini membahas penerapan pintu otomatis, yang merupakan salah satu permasalahan kehidupan sehari-hari yang dapat diimplementasikan teknologi FSM.

Penulisan laporan bertujuan memenuhi objektif proyek akhir praktikum Perancangan Sistem Digital. Selain itu, laporan dibuat untuk menunjukkan implementasi teknologi yang sebelumnya dipelajari selama masa pembelajaran mata kuliah Perancangan Sistem Digital. Diharapkan dari penulisan laporan ini, tujuan penulis dapat tersampaikan dan dapat diterapkan dalam kehidupan nyata secara aplikatif, solutif, dan efisien.

Penulis mengucapkan terima kasih kepada asisten laboratorium selaku pembimbing praktikum perancangan sistem digital. Penulis juga mengucapkan terima kasih kepada teman-teman dan orang tua dalam membantu baik dukungan dan materi untuk membuat laporan ini. Penulis menyadari bahwa terdapat kekurangan dalam menyusun laporan. Maka dari itu, penulis memohon kritik dan saran yang membangun untuk menyempurnakan dalam pembuatan makalah berikutnya. Harapan penulis adalah dengan makalah ini semoga pembaca memahami dan menerapkan ilmu dasar sistem digital dengan baik

Depok, December 10, 2022

Group B3

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Setelah masa pandemi, perkembangan teknologi masih terus dan akan berkembang pesat. Bahkan dengan adanya keterbatasan-keterbatasan yang muncul selama masa pandemi, berbagai kreativitas melahirkan aplikasi teknologi yang dapat memudahkan dan meningkatkan efisiensi dalam kehidupan manusia. Terutama dalam kehidupan sehari-hari yang sebelumnya tidak terlalu dipertimbangkan oleh manusia sebelumnya.

Pintu otomatis adalah salah satu contoh aplikasi teknologi yang dibutuhkan dalam kehidupan sehari-hari. Umumnya pintu otomatis dibutuhkan dalam perkantoran atau ruangan publik yang sering digunakan oleh banyak pihak. Karena ruangan publik selalu dipakai oleh berbagai pihak, maka masalah keamanan selalu menjadi topik yang selalu dipertimbangkan dalam penggunaan ruangan.

Dalam beberapa kasus, beberapa ruangan menginginkan hanya orang-orang tertentu yang dapat mengakses suatu ruangan, atau yang terverifikasi datanya untuk menggunakan ruangan. Hal ini tidak hanya berlaku untuk ruangan publik saja, tetapi juga ruangan pribadi yang ingin menjaga privasi pemilik ruangan. Jika selama ini masalah keamanan ini diselesaikan dengan menyediakan petugas keamanan atau rekaman tamu, kenyataannya masih terjadi kasus-kasus penyusupan ruangan oleh pihak yang tidak diinginkan.

Maka untuk meningkatkan sistem keamanan tersebut, maka teknologi pintu otomatis dapat diaplikasikan dalam permasalahan ini. Di mana pintu bekerja saat menginput kode berupa angka atau kartu. Pintu akan memeriksa kode input untuk verifikasi, lalu akan terbuka otomatis jika input terverifikasi. Pengguna kemudian dapat mengakses ruangan tersebut, dan jika ingin keluar ruangan pengguna tidak perlu menginput kode untuk verifikasi. Selain itu untuk mencegah adanya penyusup saat pengguna mengakses ruangan, pintu ditambahkan *timer* di mana dalam waktu singkat jika pintu masih terbuka akan tertutup secara otomatis dan mengunci.

Dengan mengaplikasikan teknologi pintu otomatis, diharapkan dapat memaksimalkan fungsi pintu otomatis dan juga meningkatkan keamanan. Selain itu juga dengan aplikasi

teknologi tersebut, diharapkan *human error* yang mungkin terjadi dapat diminimalisir. Berdasarkan hal tersebut, kami memutuskan untuk melaksanakan proyek *Smart Locking Door* untuk mengimplementasikan sistem pintu otomatis dengan menggunakan FSM.

1.2 PROJECT DESCRIPTION

Proyek praktikum berjudul *Smart Office Locking Door. Smart Office Locking Door* adalah suatu sistem di mana pintu dapat terbuka secara otomatis jika memasukkan kode input/kartu yang sesuai dan terverifikasi. Pintu akan tertutup setelah beberapa waktu tertentu menggunakan *timer*, dan akan terkunci secara otomatis. Jika dimasukkan kode input yang salah maka pintu tetap tertutup dan tidak memberikan akses masuk. Pengguna juga dapat keluar dari dalam ruangan namun dalam kondisi ini tidak perlu menggunakan kode untuk *unlock* pintu.

Sistem terdiri atas input berupa kode dan sensor pintu. Sensor pintu bernilai 0 dan 1 untuk identifikasi kondisi tertutup dan terbuka. Untuk output sistem memiliki enam outputs, yaitu LED merah dan hijau untuk menunjukkan tanda *true* dan *false*, *Unlock Door* untuk menunjukkan kondisi pintu terbuka, *counter* untuk menghitung jumlah pengguna yang masuk dalam ruangan, dan *seven segment* untuk *display* hasil counter dan menunjukkan jumlah orang yang masuk ruangan.

Sistem memiliki penambahan fitur berupa *timer* dan implementasi *hash. Timer* digunakan untuk menerapkan waktu terbatas berapa lama pintu tetap terbuka setelah input terverifikasi. Setelah melewati waktu yang ditetapkan, maka pintu akan tertutup dan terkunci secara otomatis. Implementasi *hash* adalah untuk metode enkripsi kode input pintu otomatis untuk meningkatkan keamanan sistem.

1.3 OBJECTIVES

Tujuan dari pembuatan proyek akhir adalah sebagai berikut:

1. Mengimplementasikan materi dan modul-modul praktikum Perancangan Sistem Digital
2. Dapat membuat sebuah sistem pintu otomatis yang dapat menghitung jumlah pengguna dalam akses ruangan

3. Mengimplementasikan sistem FSM ke dalam kode VHDL
4. Membuat *testbench* untuk menguji kode agar sesuai input dan output yang diinginkan.

1.4 ROLES AND RESPONSIBILITIES

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

Roles	Responsibilities	Person
Anggota	Membuat Laporan Membuat FSM Chart	Fayza Nirwasita
Anggota	Membuat Kode VHDL Membuat Laporan	Cecilia Inez Reva M.
Anggota	Membuat Kode VHDL Membuat Laporan	Zefanya Christira Deardo
Anggota	Membuat Laporan Membuat PPT	Brian Yudha Sandi

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Peralatan yang akan digunakan dalam proyek akhir adalah sebagai berikut:

- Visual Studio Code (.vhd)
- Draw.io
- Modelsim

2.2 IMPLEMENTATION

Smart Office Locking Door (SOLD) menggunakan *Finite State Machine* dalam implementasi sistem pintu otomatis. Sistem memiliki tujuh buah state, yaitu: *IDLE, WAIT_OPEN, NO_ACCESS, OPEN_LOCK, TIMER, LIMIT,* dan *QUIT.*

Penjelasan dari 7 state :

- idle : terus looping jika pintu dalam keadaan tidak terbuka dan juga dalam kondisi tidak ada kode input dari luar
- wait_open : kondisi memeriksa kode akses kartu sesuai atau tidak sesuai
- no_access : untuk penyampaian informasi kepada pengguna bahwa kode akses kartu salah dengan led merah sebagai tanda untuk kode yang salah
- open_lock : menunjukkan bahwa akses ruangan office berhasil dengan led hijau sebagai tanda untuk kode yang benar dan kunci pintu akan terbuka.
- timer : kondisi penghitung waktu untuk durasi tertentu saat kunci pintu sudah berhasil terbuka namun pintu tidak segera dibuka
- limit : ruangan office hanya dapat diakses oleh beberapa orang sehingga jika ruangan sudah penuh, led merah akan menyala
- quit : kondisi ketika terdapat orang yang akan keluar dari ruangan dan dilakukan juga perhitungan menggunakan counter untuk menunjukkan jumlah orang di dalam ruangan yang akan ditampilkan melalui seven segment.

Untuk mendeskripsikan sistem dalam kode VHDL, program menggunakan arsitektur behavioral dan sebuah entity yang disebut juga dengan blackbox. Blackbox atau entity digunakan untuk merepresentasikan input pada sistem yaitu sensor yang terdapat pada gagang pintu otomatis dari kode akses kartu untuk membuka pintu dan juga akses untuk membuka pintu dari dalam ruangan tanpa diperlukan kode. Selain itu, blackbox juga merepresentasikan output pada sistem yaitu kunci depan dan kunci belakang yang ditandai dengan menggunakan LED hijau dan LED merah sebagai tanda keberhasilan dan kegagalan dalam mengakses pintu serta 2 buah seven-segment untuk menampilkan jumlah orang yang berada di ruangan dalam jumlah satuan dan puluhan. Untuk penjelasan dari blackbox dari sistem *Smart Office Locking Door* dapat dilihat pada Fig 1.

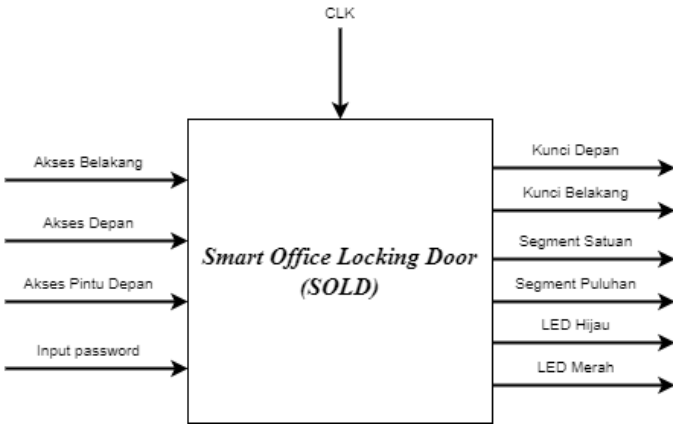


Fig 1. Skematik Blackbox dari Smart Office Locking Door (SOLD)

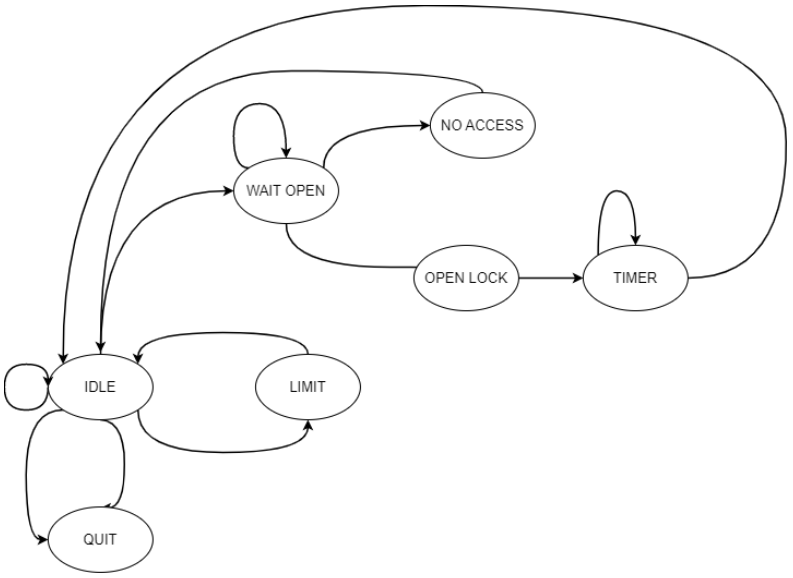


Fig 2. Skematik FSM

Untuk penjelasan dari Finite State Machine dari sistem *Smart Office Locking Door* dapat dilihat pada Fig 2. State awal dari sistem adalah IDLE. Pada IDLE terdapat 4 kondisi yang berbeda untuk menentukan state selanjutnya. Apabila terdapat orang yang ingin mengakses pintu depan sementara tidak ada yang ingin keluar dari ruangan maka kode akses kartu akan dimasukkan ke dalam data_in yang akan menggunakan metode hashing yaitu md5 hash pada kode akses kartu. Kemudian akan memulai hash. Jika counter adalah 00110010 atau merupakan limit untuk jumlah orang di dalam ruangan maka sistem akan lanjut ke state LIMIT. Jika tidak, maka akan ke state WAIT_OPEN. Kemudian. untuk kondisi ketika terdapat orang dari dalam ruangan office yang ingin keluar dari ruangan atau akses belakang = 1 maka akan memulai counter yaitu perhitungan untuk jumlah orang di dalam office.

State selanjutnya adalah WAIT_OPEN yaitu kondisi untuk mengecek kode akses kartu yang menggunakan metode hash. Jika sesuai maka counter juga akan menghitung orang yang masuk ke dalam ruangan dan menuju ke state OPEN_LOCK dimana pintu terbuka. Jika tidak maka akan ke state NO_ACCESS yaitu kondisi pintu akan dikunci dan LED merah untuk menginformasikan kepada pengguna bahwa pintu tidak dapat terbuka. Pada state OPEN_LOCK pintu akan terbuka, counter akan menghitung sebagai pengguna 1 setiap pintu terbuka dan akan menuju ke state TIMER dimana timer sebagai pengatur waktu untuk durasi tertentu ketika kunci pintu berhasil dibuka namun tidak terbuka maka akses membuka room akan dibatalkan dan balik ke state IDLE. State terakhir adalah state QUIT yaitu kondisi ketika orang yang sudah berada di dalam ruangan ingin keluar dari ruangan tanpa perlu akses kartu dapat segera keluar (kunci_belakang = 1) dan counter juga akan menghitung pengurangan jumlah orang yang keluar.

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Dalam percobaan, kami melakukan pengujian pada VHDL yang telah dibuat sebelumnya. Dalam hal ini kami menggunakan kode VHDL dengan mendefinisikan dalam satu file yaitu Smart Office Locking Door menggunakan Visual Code Studio. Dalam kode VHDL, kami membagi statenya menjadi beberapa state, yaitu *IDLE, WAIT_OPEN, NO_ACCESS, OPEN_LOCK, TIMER, LIMIT,* dan *QUIT.*

- Door Lock VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;



-- Entity
ENTITY doorLock IS
    PORT (
        -- Input
        CLK : IN STD_LOGIC;
        Pintu_depan, Akses_depan, Akses_belakang : IN STD_LOGIC;
        Input_Pass : IN STD_LOGIC_VECTOR(31 DOWNTO 0);

        -- Output
        Kunci_depan, Kunci_belakang : OUT STD_LOGIC;
        Segment1, Segment2 : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);
        LED_hijau, LED_merah : OUT STD_LOGIC
```

```
    );
END doorLock;


-- Architecture
ARCHITECTURE behavioral OF doorLock IS


    component timer_count is
        generic(
            Second : integer := 5
        );
        port(
            Clk     : in std_logic;
            START : in std_logic;
            DONE : out std_logic
        );
    end component;


    component md5_hash is
        Port (
            data_in:     in  STD_LOGIC_VECTOR (31 downto 0);
              data_out:     out STD_LOGIC_VECTOR (127 downto 0) :=
(others => '0');
            hash_done:   out STD_LOGIC;
            hash_start:  in  STD_LOGIC;
            clk:         in  STD_LOGIC;
            reset:       in  STD_LOGIC
        );
    end component;
```

```
    TYPE states IS (IDLE, WAIT_OPEN, OPEN_LOCK, NO_ACCESS, LIMIT,
QUIT, TIMER);

    SIGNAL CS, NS : states;

    SIGNAL X, Y : std_logic_vector(7 downto 0) := X"00";

    SIGNAL CNT : std_logic_vector(3 downto 0) := x"0";

    SIGNAL Counter : std_logic_vector(7 downto 0) := x"00";

    SIGNAL START, DONE : std_logic := '0';


     SIGNAL data_in: STD_LOGIC_VECTOR (31 downto 0) := (others =>
'0');

     SIGNAL data_out: STD_LOGIC_VECTOR (127 downto 0) := (others
=> '0');

    SIGNAL hash_start, reset: STD_LOGIC := '0';


         constant  Pass  :  std_logic_vector(127  downto  0)  :=
x"403AD5A2515657485E4C3AD825814D34";


BEGIN


    timer_proc: timer_count

        generic map (

            Second => 5

        )

        port map(

            CLK => CLK,

            START => START,

            DONE => DONE

        );


    hash_proc : md5_hash
```

```
        port map(
            data_in => data_in,
            data_out => data_out,
            hash_start => hash_start,
            CLK => CLK,
            reset => reset
        );


    sync_proc : PROCESS (CLK, NS)
    BEGIN
        IF (rising_edge(CLK)) THEN
            CS <= NS;
        END IF;
    END PROCESS;


      comb_proc : PROCESS (CS, Akses_depan, Akses_belakang, CNT,
Pintu_depan, DONE, START, data_out)
    BEGIN
        CASE CS IS
                WHEN IDLE => -- terus looping jika tidak ada yang
masuk/keluar office
                hash_start <= '0';
                START <= '0';


                Kunci_depan <= '0';
                Kunci_belakang <= '0';


                LED_hijau <= '0';
                LED_merah <= '0';
```

```
                IF (Akses_depan = '1' AND Akses_belakang = '0')
THEN

                    data_in <= Input_Pass;

                    hash_start <= '1';

                    IF (Counter = "00110010") THEN

                        NS <= LIMIT;

                    ELSE

                        NS <= WAIT_OPEN;

                    END IF;

                    ELSIF (Akses_belakang = '1' AND Akses_depan =
'0') THEN

                     Counter <= std_logic_vector(unsigned(Counter)
- to_unsigned(1, Counter'length));

                    NS <= QUIT;

                ELSE

                    NS <= IDLE;

                END IF;


                 WHEN WAIT_OPEN => -- mengecek kode akses kartu
benar/tidak

                                              if   (NOT(data_out   =
x"00000000000000000000000000000000")) then

                    IF (data_out = Pass) THEN

                                                        Counter  <=
std_logic_vector(unsigned(Counter)        +        to_unsigned(1,
Counter'length));

                        NS <= OPEN_LOCK;

                    ELSE

                        NS <= NO_ACCESS;

                    END if;

                else
```

```
                    NS <= WAIT_OPEN;
                end if;

             WHEN NO_ACCESS => -- kode salah, pintu tidak dibuka,
led merah
                Kunci_depan <= '0';
                LED_hijau <= '0';
                LED_merah <= '1';
                NS <= IDLE;

            WHEN OPEN_LOCK => -- pintu dibuka
                X <= std_logic_vector(unsigned(Counter) / 10);
                Y <= std_logic_vector(unsigned(Counter) mod 10);
                Kunci_depan <= '1';
                LED_hijau <= '1';
                START <= '1';
                reset <= '1';
                NS <= TIMER;

            WHEN TIMER =>
                reset <= '0';
                if (DONE = '1' OR Pintu_depan = '1') then
                    NS <= IDLE;
                else
                    NS <= TIMER;
                end if;

            WHEN LIMIT => -- limit pegawai kantor dalam 1 ruangan
                LED_merah <= '1';
```

```
                NS <= IDLE;


              WHEN QUIT => -- kondisi keluar ruangan tanpa akses
kartu (otomatis terbuka)
                X <= std_logic_vector(unsigned(Counter) / 10);
                Y <= std_logic_vector(unsigned(Counter) mod 10);
                Kunci_belakang <= '1';
                NS <= IDLE;


            WHEN OTHERS =>
                NS <= IDLE;
        END CASE;
    END PROCESS;


    -- menghitung jumlah pegawai kantor yang masuk ke office room
    WITH X SELECT
        Segment1 <= "0111111" WHEN "00000000",
        "0000110" WHEN "00000001",
        "1011011" WHEN "00000010",
        "1001111" WHEN "00000011",
        "1100110" WHEN "00000100",
        "1101101" WHEN "00000101",
        "1111101" WHEN "00000110",
        "0000111" WHEN "00000111",
        "1111111" WHEN "00001000",
        "1101111" WHEN "00001001",
        "0000000" WHEN OTHERS;
    WITH Y SELECT
        Segment2 <= "0111111" WHEN "00000000",
```

```
        "0000110" WHEN "00000001",
        "1011011" WHEN "00000010",
        "1001111" WHEN "00000011",
        "1100110" WHEN "00000100",
        "1101101" WHEN "00000101",
        "1111101" WHEN "00000110",
        "0000111" WHEN "00000111",
        "1111111" WHEN "00001000",
        "1101111" WHEN "00001001",
        "0000000" WHEN OTHERS;
END behavioral;
```

- Md5 Hash VHDL:

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

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity md5_hash is
    Port (
        data_in:     in  STD_LOGIC_VECTOR (31 downto 0);
            data_out:      out STD_LOGIC_VECTOR (127 downto 0) :=
(others => '0');
        hash_done:   out STD_LOGIC := '0';
        hash_start:  in  STD_LOGIC;
```

```
        clk:          in  STD_LOGIC;
        reset:        in  STD_LOGIC);
end md5_hash;

architecture Behavioral of md5_hash is
subtype uint512_t is unsigned(0 to 511);
subtype uint32_t is unsigned(31 downto 0);
subtype uint8_t is unsigned(7 downto 0);

type const_s is array (0 to 63) of uint8_t;
type const_k is array (0 to 63) of uint32_t;
type message is array (0 to 15) of uint32_t;

constant S: const_s := (
    X"07", X"0C", X"11", X"16", -- 7, 12, 17, 22,
    X"07", X"0C", X"11", X"16", -- 7, 12, 17, 22,
    X"07", X"0C", X"11", X"16", -- 7, 12, 17, 22,
    X"07", X"0C", X"11", X"16", -- 7, 12, 17, 22,

    X"05", X"09", X"0E", X"14", -- 5,  9, 14, 20,
    X"05", X"09", X"0E", X"14", -- 5,  9, 14, 20,
    X"05", X"09", X"0E", X"14", -- 5,  9, 14, 20,
    X"05", X"09", X"0E", X"14", -- 5,  9, 14, 20,

    X"04", X"0B", X"10", X"17", -- 4, 11, 16, 23,
    X"04", X"0B", X"10", X"17", -- 4, 11, 16, 23,
    X"04", X"0B", X"10", X"17", -- 4, 11, 16, 23,
    X"04", X"0B", X"10", X"17", -- 4, 11, 16, 23,
```

```
    X"06", X"0A", X"0F", X"15", -- 6, 10, 15, 21);
    X"06", X"0A", X"0F", X"15", -- 6, 10, 15, 21);
    X"06", X"0A", X"0F", X"15", -- 6, 10, 15, 21);
    X"06", X"0A", X"0F", X"15"
    ); -- 6, 10, 15, 21);

constant K: const_k := (
    X"d76aa478", X"e8c7b756", X"242070db", X"c1bdceee",
    X"f57c0faf", X"4787c62a", X"a8304613", X"fd469501",
    X"698098d8", X"8b44f7af", X"ffff5bb1", X"895cd7be",
    X"6b901122", X"fd987193", X"a679438e", X"49b40821",
    X"f61e2562", X"c040b340", X"265e5a51", X"e9b6c7aa",
    X"d62f105d", X"02441453", X"d8a1e681", X"e7d3fbc8",
    X"21e1cde6", X"c33707d6", X"f4d50d87", X"455a14ed",
    X"a9e3e905", X"fcefa3f8", X"676f02d9", X"8d2a4c8a",
    X"fffa3942", X"8771f681", X"6d9d6122", X"fde5380c",
    X"a4beea44", X"4bdecfa9", X"f6bb4b60", X"bebfbc70",
    X"289b7ec6", X"eaa127fa", X"d4ef3085", X"04881d05",
    X"d9d4d039", X"e6db99e5", X"1fa27cf8", X"c4ac5665",
    X"f4292244", X"432aff97", X"ab9423a7", X"fc93a039",
    X"655b59c3", X"8f0ccc92", X"ffeff47d", X"85845dd1",
    X"6fa87e4f", X"fe2ce6e0", X"a3014314", X"4e0811a1",
    X"f7537e82", X"bd3af235", X"2ad7d2bb", X"eb86d391"
    );

signal M : uint512_t := (others => '0');
signal message_length : uint32_t := (others => '0');
signal data_counter : natural := 0;
signal loop_counter, loop_counter_n : natural := 0;
```

```
constant a0 : uint32_t := X"67452301";
constant b0 : uint32_t := X"efcdab89";
constant c0 : uint32_t := X"98badcfe";
constant d0 : uint32_t := X"10325476";

signal A, A_n : uint32_t := a0;
signal B, B_n : uint32_t := b0;
signal C, C_n : uint32_t := c0;
signal D, D_n : uint32_t := d0;
signal F      : uint32_t := to_unsigned(0, A'length);
signal g      : integer := 0;

type state_t is (
    idle,
    load_length,
    load_data,
    pad,
    rotate,
    stage1_F, stage1_B,
    stage2_F, stage2_B,
    stage3_F, stage3_B,
    stage4_F, stage4_B,
    stage5, -- add a0 to A, b0 to B etc.
    stage6, -- swap endianness
    finished,
    store_data
    );
signal state, state_n : state_t;
```

```
function leftrotate(x: in uint32_t; c: in uint8_t) return uint32_t is
begin
        return SHIFT_LEFT(x, to_integer(c)) or SHIFT_RIGHT(x, to_integer(32-c));
end function leftrotate;

function swap_endianness(x: in uint32_t) return uint32_t is
begin
    return x(7 downto 0) &
        x(15 downto 8) &
        x(23 downto 16) &
        x(31 downto 24);
end function swap_endianness;

begin
    main: process(reset, clk)
    begin
        if (reset = '1') then
            state <= idle;
            loop_counter <= 0;
        elsif (rising_edge(clk)) then
            state <= state_n;
            loop_counter <= loop_counter_n;
            A <= A_n;
            B <= B_n;
            C <= C_n;
            D <= D_n;
        end if;
```

```
    end process main;


     fsm: process(state, hash_start, loop_counter, data_counter,
message_length)
    begin
        state_n <= state;


        case state is
            when idle =>
                if (hash_start = '1') then
                    state_n <= load_length;
                end if;


            when load_length =>
                state_n <= load_data;


            when load_data =>
                if (data_counter >= message_length) then
                    state_n <= pad;
                end if;


            when pad =>
                state_n <= rotate;


            when rotate =>
                state_n <= stage1_F;


            when stage1_F =>
                state_n <= stage1_B;
```

```
        when stage1_B =>
            if (loop_counter = 15) then
                state_n <= stage2_F;
            else
                state_n <= stage1_F;
            end if;

        when stage2_F =>
            state_n <= stage2_B;

        when stage2_B =>
            if (loop_counter = 31) then
                state_n <= stage3_F;
            else
                state_n <= stage2_F;
            end if;

        when stage3_F =>
            state_n <= stage3_B;

        when stage3_B =>
            if (loop_counter = 47) then
                state_n <= stage4_F;
            else
                state_n <= stage3_F;
            end if;

        when stage4_F =>
```

```
                state_n <= stage4_B;

            when stage4_B =>
                if (loop_counter = 63) then
                    state_n <= stage5;
                else
                    state_n <= stage4_F;
                end if;

            when stage5 =>
                state_n <= stage6;

            when stage6 =>
                state_n <= store_data;

            when store_data =>
                state_n <= idle;

            when others => null;
        end case;
    end process fsm;

    calc: process(reset, clk, state, data_counter, loop_counter)
    begin
        if (reset = '0' and rising_edge(clk)) then

            case state is
                when load_length =>
```

```
                    message_length <= to_unsigned(data_in'length,
message_length'length);


                when load_data =>

                            M(data_counter to data_counter+31) <=
unsigned(data_in);

                    if (data_counter < message_length) then

                        data_counter <= data_counter + 32;

                    end if;


                when pad =>

                    M(to_integer(message_length)) <= '1';

                        M(to_integer(message_length+1) to 447) <=
(others => '0');

                    M(448 to 511) <=

                                swap_endianness(message_length) &
"00000000000000000000000000000000";


                when rotate =>

                    for i in 0 to 15 loop

                                        M(32*i to 32*i+31) <=
swap_endianness(M(32*i to 32*i+31));

                    end loop;


                when stage1_B | stage2_B | stage3_B | stage4_B =>

                    A_n <= D;

                     B_n <= B + leftrotate(A + F + K(loop_counter)
+ M(g to g+31), s(loop_counter));

                    C_n <= B;

                    D_n <= C;

                    loop_counter_n <= loop_counter + 1;
```

```
            when stage1_F =>
                F <= (B_n and C_n) or (not B_n and D_n);
                g <= 32*loop_counter_n;

            when stage2_F =>
                F <= (D_n and B_n) or (not D_n and C_n);
                g <= 32*((5*loop_counter_n + 1) mod 16);

            when stage3_F =>
                F <= B_n xor C_n xor D_n;
                g <= 32*((3*loop_counter_n + 5) mod 16);

            when stage4_F =>
                F <= C_n xor (B_n or not D_n);
                g <= 32*((7*loop_counter_n) mod 16);

            when stage5 =>
                A_n <= A_n + a0;
                B_n <= B_n + b0;
                C_n <= C_n + c0;
                D_n <= D_n + d0;

            when stage6 =>
                A_n <= swap_endianness(A_n);
                B_n <= swap_endianness(B_n);
                C_n <= swap_endianness(C_n);
                D_n <= swap_endianness(D_n);
```

```
                when store_data =>
                    hash_done <= '1';
                                data_out  <=  std_logic_vector(A)  &
std_logic_vector(B) & std_logic_vector(C) & std_logic_vector(D);

                when others => null;
            end case;

        end if;
    end process calc;

end Behavioral;
```

- Timer Count VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity timer_count is
    generic(
        Second : integer := 5);
    port(
        Clk     : in std_logic;
        START : in std_logic;
        DONE : out std_logic := '0'
    );
end entity;

architecture rtl of timer_count is
```

```
    -- Signal for counting clock periods
    signal Ticks : integer := 0;
    signal RST : std_logic := '0';

begin

    process(Clk, Rst, Ticks, START) is
    begin
        if rising_edge(Clk) then
            if (START = '1') then
                if (Rst = '1') then
                    Ticks   <= 0;
                else
                    if Ticks = Second - 1 then
                        Ticks <= 0;
                        DONE <= '1';
                    else
                        Ticks <= Ticks + 1;
                    end if;

                end if;
            end if;
        end if;
    end process;

end architecture;
```

- Testbench:

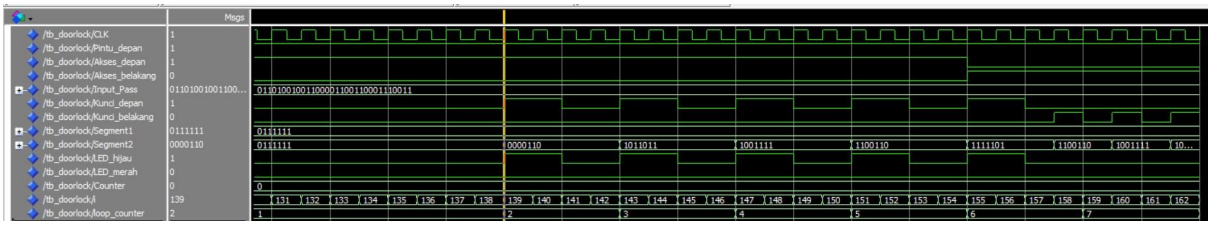
```
-- Libraries / Package

LIBRARY ieee;

USE ieee.std_logic_1164.ALL;


-- Entity

ENTITY tb_doorLock IS

END tb_doorLock;


-- Architecture

ARCHITECTURE behavioral OF tb_doorLock IS

    component doorLock IS

    PORT (

        -- Input

        CLK : IN STD_LOGIC;

        Pintu_depan, Akses_depan, Akses_belakang : IN STD_LOGIC;

        Input_Pass : IN STD_LOGIC_VECTOR(31 DOWNTO 0);


        -- Output

        Kunci_depan, Kunci_belakang : OUT STD_LOGIC;

        Segment1, Segment2 : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);

        LED_hijau, LED_merah : OUT STD_LOGIC

    );

    end component;


    SIGNAL CLK : STD_LOGIC;

    SIGNAL Pintu_depan, Akses_depan, Akses_belakang : STD_LOGIC;

       Signal  Input_Pass  :  STD_LOGIC_VECTOR  (31  DOWNTO  0)  :=
("01101001001100001100110001110011"); --X"6930CC73";
```

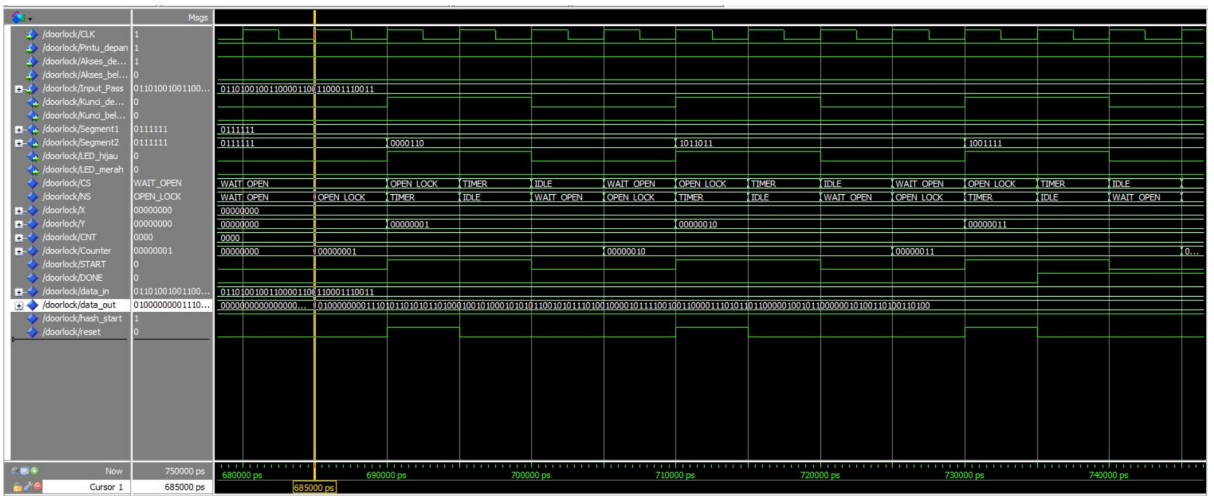
```
    Signal Kunci_depan, Kunci_belakang : STD_LOGIC;
    Signal Segment1, Segment2 : STD_LOGIC_VECTOR (6 DOWNTO 0);
    Signal LED_hijau, LED_merah :  STD_LOGIC;
    SIGNAL Counter : INTEGER RANGE 0 TO 50;
    CONSTANT T : TIME := 50 ns;
    CONSTANT T2 : TIME := 6900 ns;
    CONSTANT max_clk : INTEGER := 200;
    SIGNAL i : INTEGER := 1;
    SiGNAL loop_counter : INTEGER := 0;

BEGIN
    uut : doorLock PORT MAP(
        CLK => CLK,
        Pintu_depan => Pintu_depan,
        Akses_depan => Akses_depan,
        Akses_belakang => Akses_belakang,
        Input_Pass => Input_Pass,

        Kunci_depan => Kunci_depan,
        Kunci_belakang => Kunci_belakang,
        Segment1 => Segment1,
        Segment2 => Segment2,
        LED_hijau => LED_hijau,
        LED_merah => LED_merah
    );

    clock_generator : PROCESS
    BEGIN
```

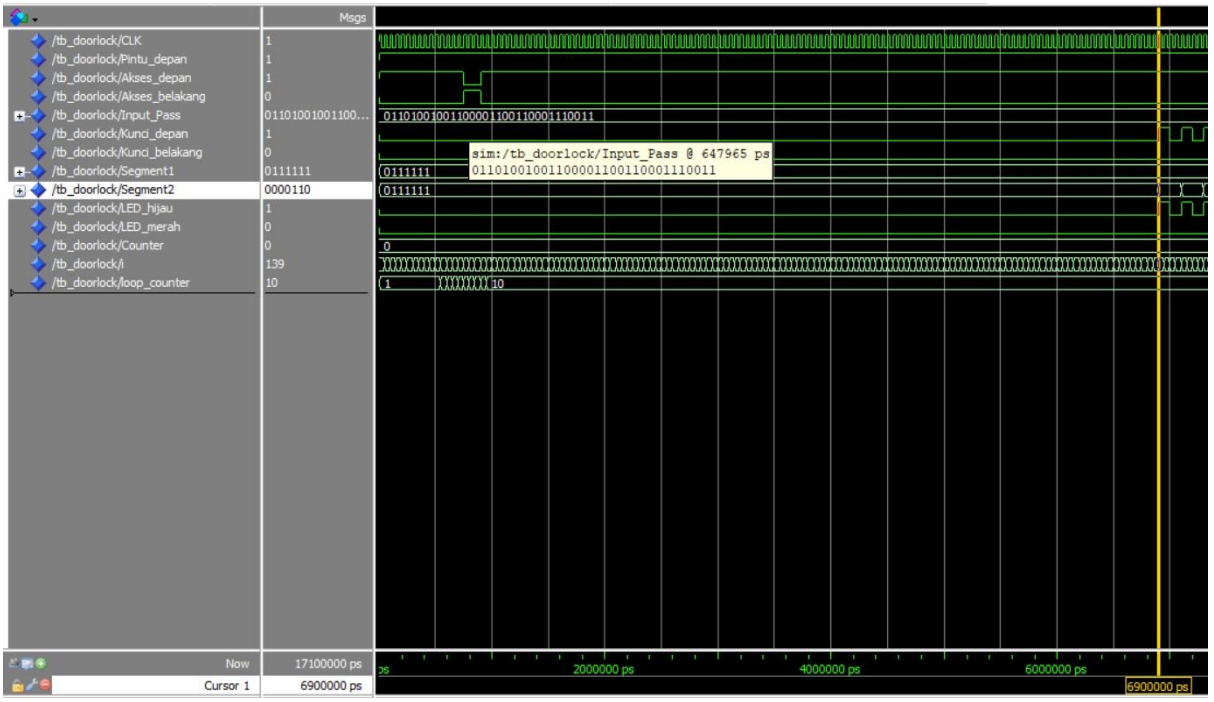
```
        CLK <= '1';
        WAIT FOR T/2;
        CLK <= '0';
        WAIT FOR T/2;
        IF (i < max_clk) THEN
            i <= i + 1;
        ELSE
            WAIT;
        END IF;
    END PROCESS;

    stim_proc : PROCESS
         constant Pintu_depan_stream : STD_LOGIC_VECTOR(0 TO 9) :=
("1111111111");
         CONSTANT Akses_depan_stream : STD_LOGIC_VECTOR(0 TO 9) :=
("1111100011");
         CONSTANT Akses_belakang_stream : STD_LOGIC_VECTOR(0 TO 9)
:= ("0000011100");
          CONSTANT LED_hijau_stream : STD_LOGIC_VECTOR(0 TO 9) :=
("1101100011");
          CONSTANT LED_merah_stream : STD_LOGIC_VECTOR(0 TO 9) :=
("0000001100");
    BEGIN
        FOR j IN 0 TO 9 LOOP

            loop_counter <= loop_counter + 1;
            Pintu_depan <= Pintu_depan_stream (j);
            Akses_depan <= Akses_depan_stream (j);
            Akses_belakang <= Akses_belakang_stream(j);
            if (j = 0) then
                WAIT for T2;
```

```
            else WAIT FOR 4*T;

            end if;


        END LOOP;

        WAIT;

    END PROCESS;

END behavioral;
```

3.2 RESULT







3.3 ANALYSIS

Dari hasil pengujian melalui testbench dan secara langsung

CHAPTER 4

CONCLUSION

SOLD atau *Smart Office Locking Door* adalah sebuah sistem keamanan otomatis dari sebuah pintu. Pada sistem ini, digunakan kode input, atau bisa juga menggunakan kartu sebagai metode verifikasi. Sebagai bagian dari mekanisme penguncian pintu, digunakan Finite State Machine, dengan tujuh buah state yaitu idle, wait_open, no_access, open_lock, timer, limit, dan quit. Untuk indikator output pada sistem ini, digunakan lampu LED yang akan berwarna hijau apabila kode input atau kartu berhasil terverifikasi, dan akan berwarna merah jika gagal. Dan juga, terdapat seven-segment display, untuk menampilkan jumlah orang yang terdapat dalam ruangan yang terintegrasi dengan sistem SOLD ini

Secara sederhana, sistem ini bekerja dengan mendeteksi dan memeriksa kode akses, lalu kemudian akan memulai hash. Jika di dalam ruangan tersebut sudah memenuhi batas,

maka sistem akan masuk ke dalam state limit, namun jika tidak akan masuk ke dalam wait_open dan kemudian akan masuk ke dalam state open_lock, dengan catatan, terdapat state timer, yang akan menghitung waktu apabila pintu tersebut tidak segera dibuka, maka akan kembali ke state idle.

REFERENCES

[1] B. Mealy, F. Tappero. 2018. FREE RANGE VHDL, diakses pada tanggal 28 November 2022.

[2] C. H. Roth, L. K. John. 2008. Digital Systems Design Using VHDL, 2nd edition, diakses pada tanggal 30 November 2022

[3] P. P. Chu. 2008. FPGA PROTOTYPING BY VERILOG EXAMPLES, 3rd edition., diakses pada tanggal 30 November 2022.