1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1

</ Teams

Nikolas Samosir	2206059396
Rifqi Ramadhan	2206062964
Daniel Niko Mardjaja	2206026183
Fairuz Muhammad	2206814324

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Table of contents

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

Penjelasan Umum

01

} /> [

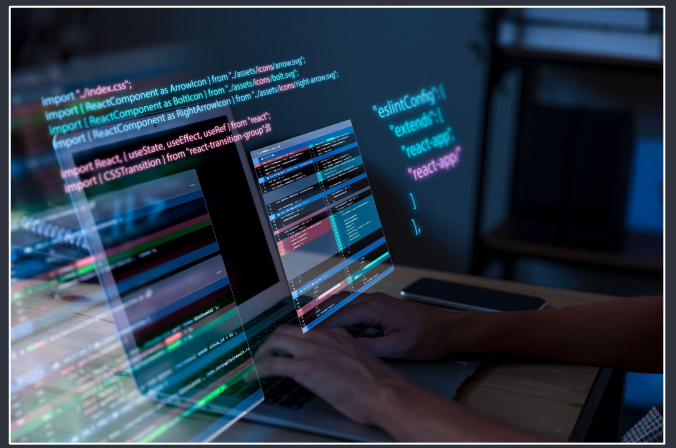
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Penjelasan Umum

Pada zaman modern ini, mayoritas sudah menggunakan password untuk keamanan pertama dari data pengguna, namun jika password tersebut tidak dilakukan enkripsi, maka akan mudah untuk diretas dan di ambil datanya.

Kelompok kami membuat program VHDL yang dapat melakukan enkripsi password agar tidak mudah dibaca oleh manusia pada database dan data password mereka akan aman pada database.

VHDL>



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

Tujuan proyek

02

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Tujuan Proyek

Membuat Program VHDL

Membuat program VHDL yang dapat mengimplementasikan enkripsi password.

Meningkatkan Keamanan User

Dengan melakukan enkripsi, data dari user akan lebih aman karena password yang mereka input akan disimpan dan tidak dapat dibaca oleh manusia

</>

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

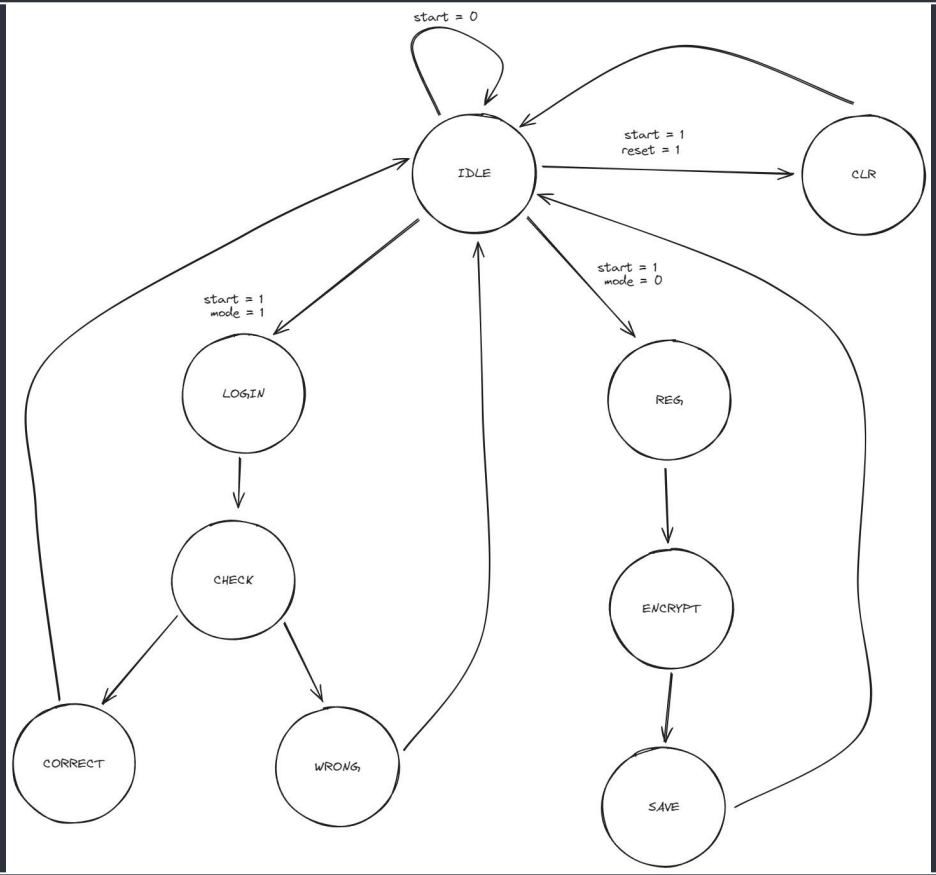
Implementasi

03

} /> [

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1

</ State Diagram



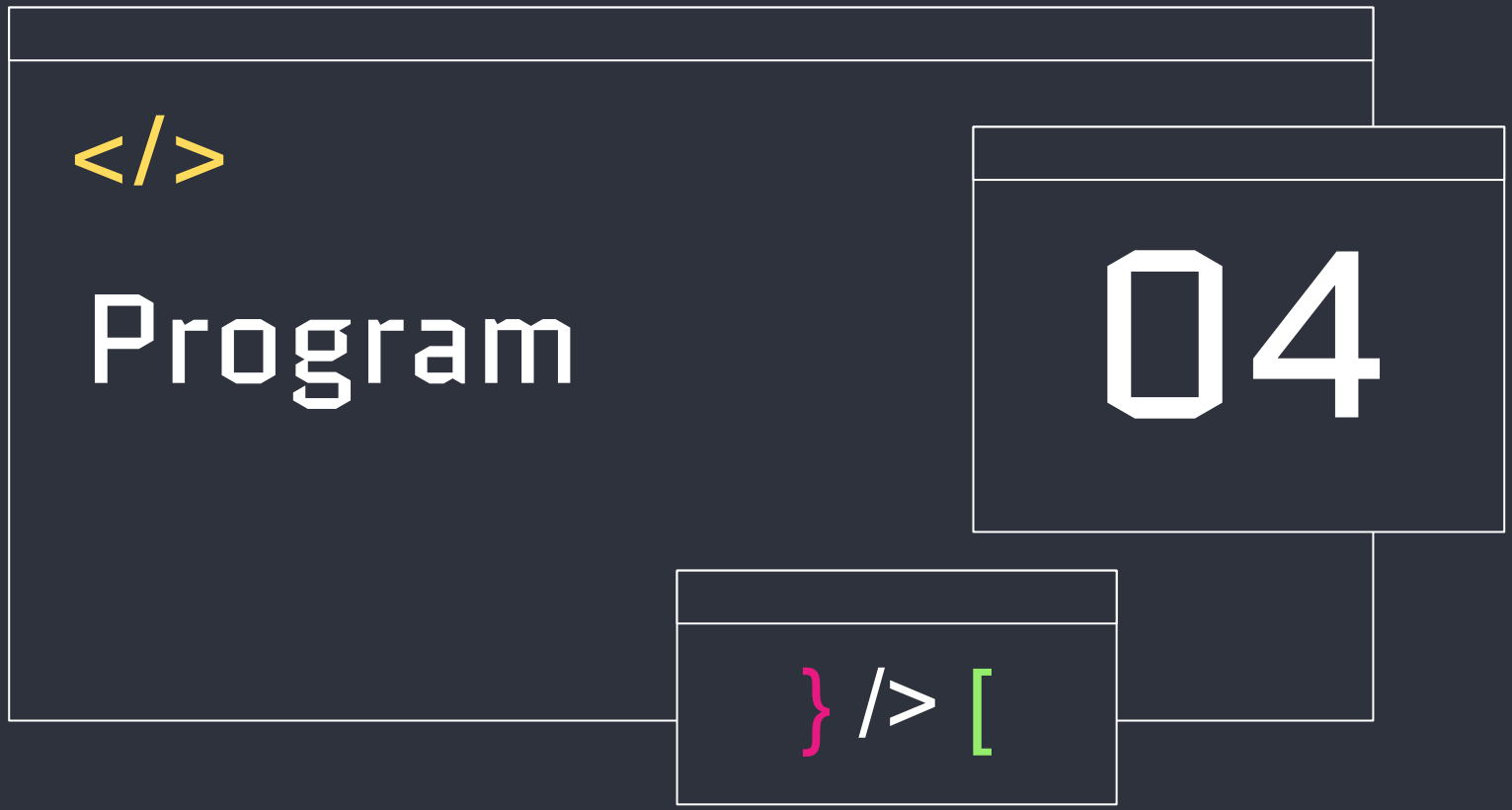
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Implementasi

Kelompok kami membuat program ini dengan mengimplementasikan Finite State Machine yang memiliki 9 state yang terdiri atas IDLE, CLR, REG, ENCRYPT, SAVE, LOGIN, CHECK, CORRECT, dan WRONG. Program ini memiliki 3 fitur utama yaitu register, login dan juga reset. Register digunakan untuk menambahkan password, login digunakan untuk user agar mendapatkan akses, dan reset yang digunakan untuk menghapus semua password yang telah di register.

Program ini juga mengimplementasikan portmap yang digunakan untuk menghubungkan beberapa program dengan file utama yaitu Main.vhd, dan file yang lainnya adalah FileHandling.vhd, dan RandomKey.vhd. FileHandling.vhd digunakan untuk melakukan save untuk password dan key yang didapat dari RandomKey.vhd. RandomKey.vhd digunakan untuk membuat key string secara acak sehingga key yang didapat untuk setiap password tidak akan sama dan dapat meningkatkan keamanan lebih lagi untuk user.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1



1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Main.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use std.textio.all;
use ieee.std_logic_textio.all;

entity Main is
    Port (
        clk : in STD_LOGIC;
        start : in STD_LOGIC;
        mode : in STD_LOGIC;
        reset : in STD_LOGIC;
        input_password : in string(1 to 8);

        success : out STD_LOGIC;
        fail : out STD_LOGIC
    );
end Main;

architecture Behavioral of Main is

    component RandomKey is
        port (
            clk : in std_logic;
            RandomVector : out string(1 to 8)
        );
    end component;

    component FileHandling is
        port (
            EN : in STD_LOGIC;
            saved_key : in string(1 to 8);
            saved_password : in string(1 to 8)
        );
    end component;

    type state_type is (IDLE, CLR, REG, ENCRYPT, SAVE, LOGIN, CHECK, WRONG, CORRECT);
    signal curr_state : state_type := IDLE;
    signal next_state : state_type;
    signal RandomVector : string(1 to 8);

    signal EN : std_logic;
    signal saved_password : string(1 to 8);
    signal saved_key : string(1 to 8);
```

```
    function check_valid_password(to_check : string(1 to 8))
        return STD_LOGIC is

        file pass_text : text ;
        variable row_read : line;

        variable encrypted_check : string (1 to 8);
        variable temp_pass_string : string(1 to 8);
        variable temp_key_string : string(1 to 8);
        variable temp_string : string(1 to 17);

        variable char_key_int : Integer;
        variable char_pass_int : Integer;
        variable char_encrypt_int : Integer;
    begin
        file_open(pass_text, "Pass.txt", read_mode);
        while not endfile(pass_text) loop
            readline(pass_text, row_read);
            read(row_read, temp_string);

            temp_pass_string := temp_string(1 to 8);
            temp_key_string := temp_string(10 to 17);
            for i in 1 to 8 loop
                char_key_int := character'pos(temp_key_string(i));
                char_pass_int := character'pos(to_check(i));
                char_encrypt_int := (char_pass_int + char_key_int - 5);
                char_encrypt_int := char_encrypt_int * 2;
                char_encrypt_int := (char_encrypt_int mod 93) + 33;

                encrypted_check(i) := character'val(char_encrypt_int);
            end loop;

            report temp_pass_string & "|||" & temp_key_string;
            if(encrypted_check = temp_pass_string) then
                return '1';
            end if;

        end loop;

        return '0';
    end function;
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Main.vhd

```
89      procedure encrypt_pass(signal pass : inout string (1 to 8);
90                             signal key : in string(1 to 8)) is
91          variable char_key_int : Integer;
92          variable char_pass_int : Integer;
93          variable char_encrypt_int : Integer;
94          variable pass_temp : string(1 to 8);
95      begin
96          for i in 1 to 8 loop
97              char_key_int := character'pos(key(i));
98              char_pass_int := character'pos(pass(i));
99              char_encrypt_int := (char_pass_int + char_key_int - 5);
100             char_encrypt_int := char_encrypt_int * 2;
101             char_encrypt_int := (char_encrypt_int mod 93) + 33;
102
103             pass_temp(i) := character'val(char_encrypt_int);
104         end loop;
105
106         pass <= pass_temp;
107     end procedure;
108
109     procedure clear_file(constant yes : std_logic) is
110         file pass_text : text;
111     begin
112         if(yes = '1') then
113             file_open(pass_text, "Pass.txt", write_mode);
114             write(pass_text, "");
115
116             report "All passcode has been deleted";
117
118             file_close(pass_text);
119         end if;
120     end procedure;
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Main.vhd

```
121  begin
122      RandomKey_instance : RandomKey port map (clk => clk, RandomVector => RandomVector);
123      FileHandling_instance : FileHandling port map (EN => EN, saved_key => saved_key, saved_password => saved_password);
124
125      process(CLK)
126      begin
127          if(rising_edge(CLK)) then
128              case curr_state is
129                  when IDLE =>
130                      EN <= '0';
131                      success <= '0';
132                      fail <= '0';
133                      if(start = '1') then
134                          if(reset = '1') then
135                              curr_state <= CLR;
136                          elsif (mode = '1') then
137                              curr_state <= LOGIN;
138                          elsif(mode = '0') then
139                              curr_state <= REG;
140                          else
141                              curr_state <= IDLE;
142                          end if;
143                      else
144                          curr_state <= IDLE;
145                      end if;
146
147                  when CLR =>
148                      clear_file('1');
149                      success <= '1';
150                      curr_state <= IDLE;
151
152                  when REG =>
153                      saved_password <= input_password;
154                      saved_key <= RandomVector;
155                      curr_state <= ENCRYPT;
156
157                  when ENCRYPT =>
158                      encrypt_pass(saved_password, saved_key);
159                      curr_state <= SAVE;
160
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Main.vhd

```
161                 when SAVE =>
162                     EN <= '1';
163                     success <= '1';
164                     curr_state <= IDLE;
165
166                 when LOGIN =>
167                     saved_password <= input_password;
168                     curr_state <= CHECK;
169
170                 when CHECK =>
171                     if(check_valid_password(saved_password) = '1') then
172                         curr_state <= CORRECT;
173                     else
174                         curr_state <= WRONG;
175                     end if;
176
177                 when CORRECT =>
178                     report "ANJAY MASUK";
179                     success <= '1';
180                     curr_state <= IDLE;
181
182                 when WRONG =>
183                     report "BJIR SALAH";
184                     fail <= '1';
185                     curr_state <= IDLE;
186             end case;
187         end if;
188     end process;
189
190
191 end architecture;
192
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Main.vhd

Program Main ini digunakan sebagai program utama untuk menjalankan program lainnya. Pada program ini terdapat Finite State Machine yang akan membuat program berjalan dengan teratur.

Untuk register program akan dimulai dari :
IDLE -> REG -> ENCRYPT -> SAVE -> IDLE

Untuk login program akan dimulai dari :
IDLE -> LOGIN -> CHECK -> CORRECT/WRONG -> IDLE

Untuk reset program akan dimulai dari :
IDLE -> CLR -> IDLE

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ FileHandling.vhd

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
use std.textio.all;
use IEEE.std_logic_textio.all;

entity FileHandling is
    port (
        EN : in STD_LOGIC;
        saved_key : in string(1 to 8);
        saved_password : in string(1 to 8)
    );
end entity FileHandling;

architecture rtl of FileHandling is
begin
    process (EN)
        file dest_text : text;
        file temp_text : text;

        variable row : line;
    begin
        if (EN = '1') then
            file_open(dest_text, "Pass.txt", read_mode);
            file_open(temp_text, "TEMP.txt", write_mode);

            while not endfile(dest_text) loop
                readline(dest_text, row);
                writeline(temp_text, row);
            end loop;

            file_close(dest_text);
            file_close(temp_text);

            file_open(dest_text, "Pass.txt", write_mode);
            file_open(temp_text, "TEMP.txt", read_mode);

            while not endfile(temp_text) loop
                readline(temp_text, row);
                writeline(dest_text, row);
            end loop;

            write(row, saved_password & " " & saved_key);
            writeline(dest_text, row);

            file_close(dest_text);
            file_close(temp_text);

            file_open(temp_text, "TEMP.txt", write_mode);
            write(temp_text, "");
            file_close(temp_text);
        end if;
    end process;

end architecture rtl;
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ FileHandling.vhd

Program FileHandling ini digunakan untuk save password ke dalam file dengan format .txt yang diberi nama Pass.txt.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ RandomKey.vhd

```vhdl
library ieee;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use std.textio.all;
use ieee.math_real.all;

entity RandomKey is
    port (
        clk : in std_logic;
        RandomVector : out string(1 to 8)
    );
end RandomKey;

architecture behavior of RandomKey is
begin
    process(CLK)
        variable seed1 : integer := 2;
        variable seed2 : integer:= 24;

        impure function random_alphabet_number_gen
            return Integer is
            variable r : Real;
            variable temp : Integer;
        begin
            uniform(seed1, seed2, r);
            temp := integer(r * 25.0);
            return temp;
        end function;

        impure function random_upper_down_gen
            return Integer is
            variable r : Real;
            variable temp : Integer;
            variable alpha_type : Integer;
        begin
            uniform(seed1, seed2, r);
            temp := integer(r * 2.0);
            if(temp = 1) then
                alpha_type := 65;
            else
                alpha_type := 97;
            end if;
            return alpha_type;
        end function;

        impure function GenerateRandomString return string is
            variable string_key : string(1 to 8);
            variable number_key : Integer;
        begin
            for i in 1 to 8 loop
                number_key := random_alphabet_number_gen + random_upper_down_gen;
                string_key(i) := character'val(number_key);
            end loop;
            return string_key;
        end function;

        variable vectorRandom : string(1 to 8);
    begin
        if rising_edge(clk) then
            vectorRandom := GenerateRandomString;
            RandomVector <= vectorRandom;
        end if;
    end process;
end architecture;
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ RandomKey.vhd

Program RandomKey ini digunakan untuk membuat key secara random yang digunakan untuk menambah keamanan dari user karena setiap key dari setiap password akan berbeda-beda sehingga sulit untuk meretas sistem.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

Percobaan 05

} /> [

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1

</ TestBench.vhd

Passcode_tb.vhd

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.std_logic_textio.all;
use std.textio.all;


entity Passcode_tb is
end entity;

architecture rtl of Passcode_tb is
    component Main
        Port (
            clk : in STD_LOGIC;
            start : in STD_LOGIC;
            mode : in STD_LOGIC;
            reset : in STD_LOGIC;
            input_password : in string(1 to 8);

            success : out STD_LOGIC;
            fail : out STD_LOGIC
        );
    end component;

    signal clk : std_logic := '1';
    signal start, mode, reset : std_logic := '0';
    signal input_password : string(1 to 8);

    signal success : std_logic;
    signal fail : std_logic;

    constant CLK_PERIOD : time := 100 ps;
begin
    MAIN_UUT : Main
        port map (
            clk => clk,
            start => start,
            mode => mode,
            reset => reset,
            input_password => input_password,
            success => success,
            fail => fail
        );

    CLOCK_TB : process
    begin
        for i in 0 to 32 loop
            wait for CLK_PERIOD / 2;
            CLK <= '0';
            wait for CLK_PERIOD / 2;
            CLK <= '1';
        end loop;
        wait;
    end process;

    tb : process
        type test_array is array (0 to 3) of string(1 to 8);
        variable register_password : test_array :=
            (0 => "abcdefgh",
             1 => "zxcvbnml",
             2 => "terqwyui",
             3 => "Daniel12");

        variable login_password : test_array :=
            (0 => "zxcvbnml",
             1 => "hgfasdjk",
             2 => "9283iuad",
             3 => "Daniel12");
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ TestBench.vhd

```
70      begin
71          --Register All Passcode
72          mode <= '0';
73
74          for i in 0 to 3 loop
75              input_password <= register_password(i);
76              wait for CLK_PERIOD * 4;
77          end loop;
78
79          mode <= '1';
80
81          for i in 0 to 3 loop
82              input_password <= login_password(i);
83              wait for CLK_PERIOD * 4;
84          end loop;
85          wait;
86      end process;
87
88      start_TB : process
89      begin
90          wait for CLK_PERIOD / 2;
91          start <= '1';
92          wait;
93      end process;
94  end architecture rtl;
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ TestBench.vhd

Program Testbench ini digunakan untuk melakukan percobaan pada program main.
Awalnya akan diberi input

```
start <= '0';
mode <= '0';
reset <= '0';
input_password <= (others => '0');
```

Ini digunakan untuk membuat semuanya menjadi 0 terlebih dahulu kemudian

```
start <= '1';
mode <= '0';
reset <= '0';
input_password <= "abcdefgh";
```

Ini digunakan untuk test register untuk program setelah itu

```
start <= '1';
mode <= '1';
reset <= '0';
input_password <= "abcdefgh";
```

Ini digunakan untuk test login pada program dengan password yang benar

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ TestBench.vhd

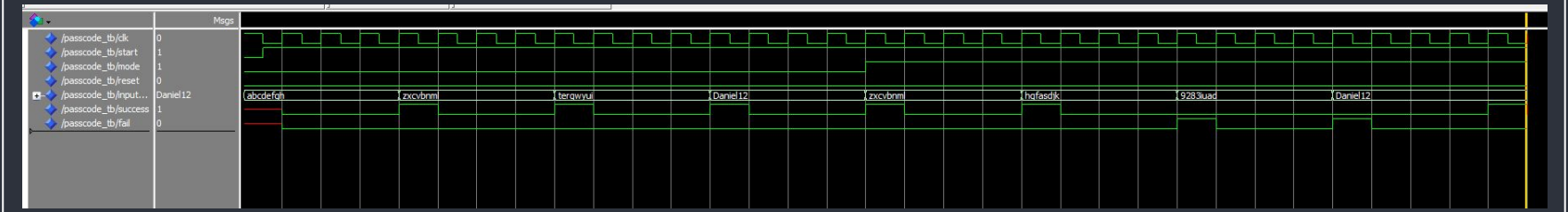
Yang terakhir adalah test login dengan password yang salah

```
start <= '1';
mode <= '1';
reset <= '0';
input_password <= "wrong123";
```

Jika login benar, hasil output success akan menjadi 1 namun jika login salah, hasil output fail akan menjadi 1.

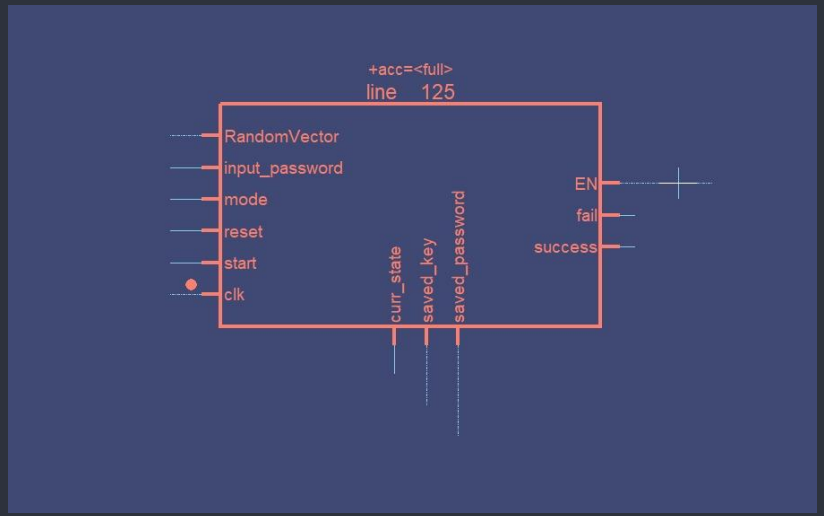
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Output TestBench.vhd

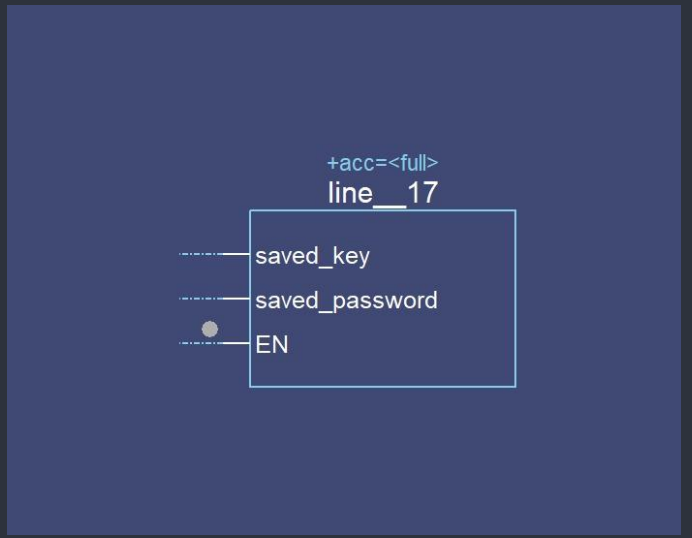


1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Hasil IC

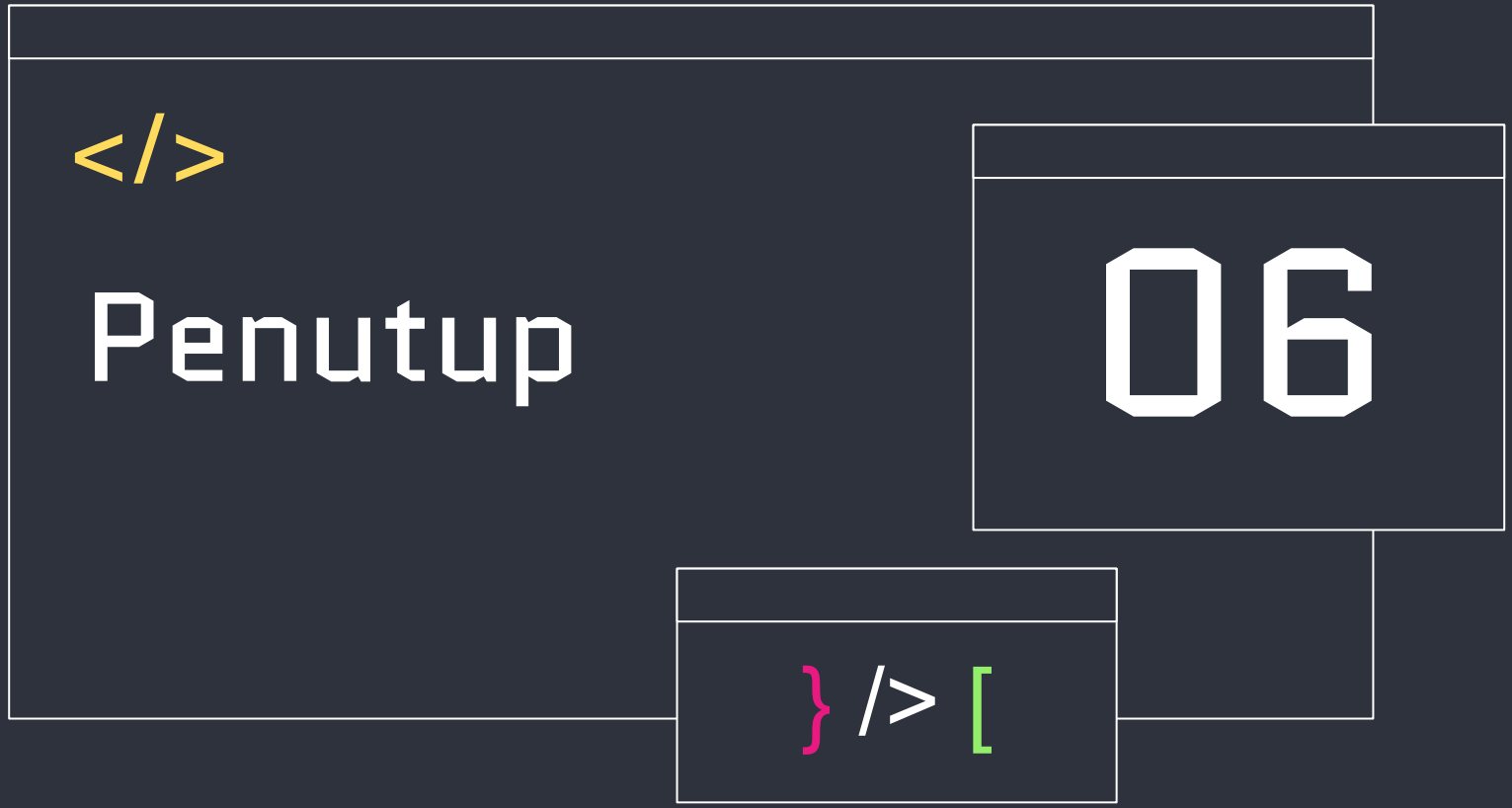


Main IC



File Handling

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

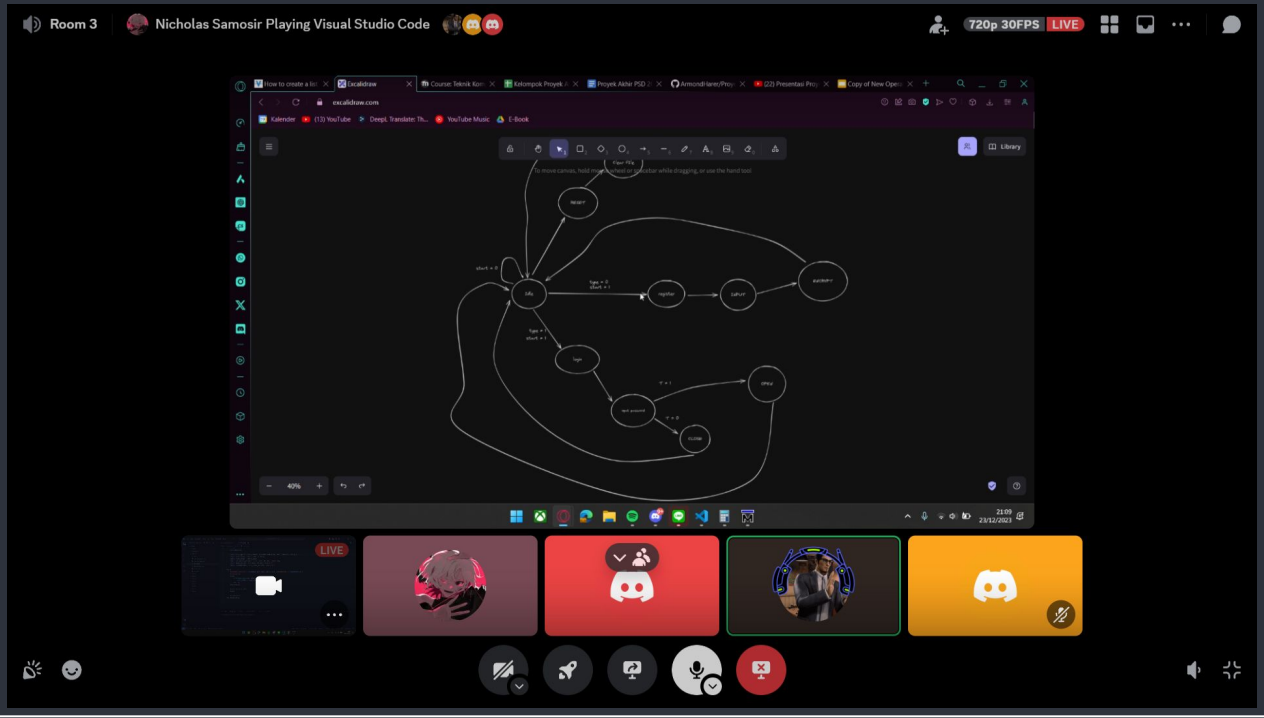

</>
Penutup
06
} /> [
1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Penutup

Sebagai Kesimpulan, Kelompok kami membuat program yang mengimplementasikan enkripsi dan dekripsi dengan menggunakan finite state. Terdapat beberapa state yang digunakan untuk melakukan enkripsi, dekripsi, login, dan reset. Program ini dibuat agar data dari para user dapat aman dengan membuat password yang telah mereka masukan menjadi tidak dapat dibaca oleh manusia.

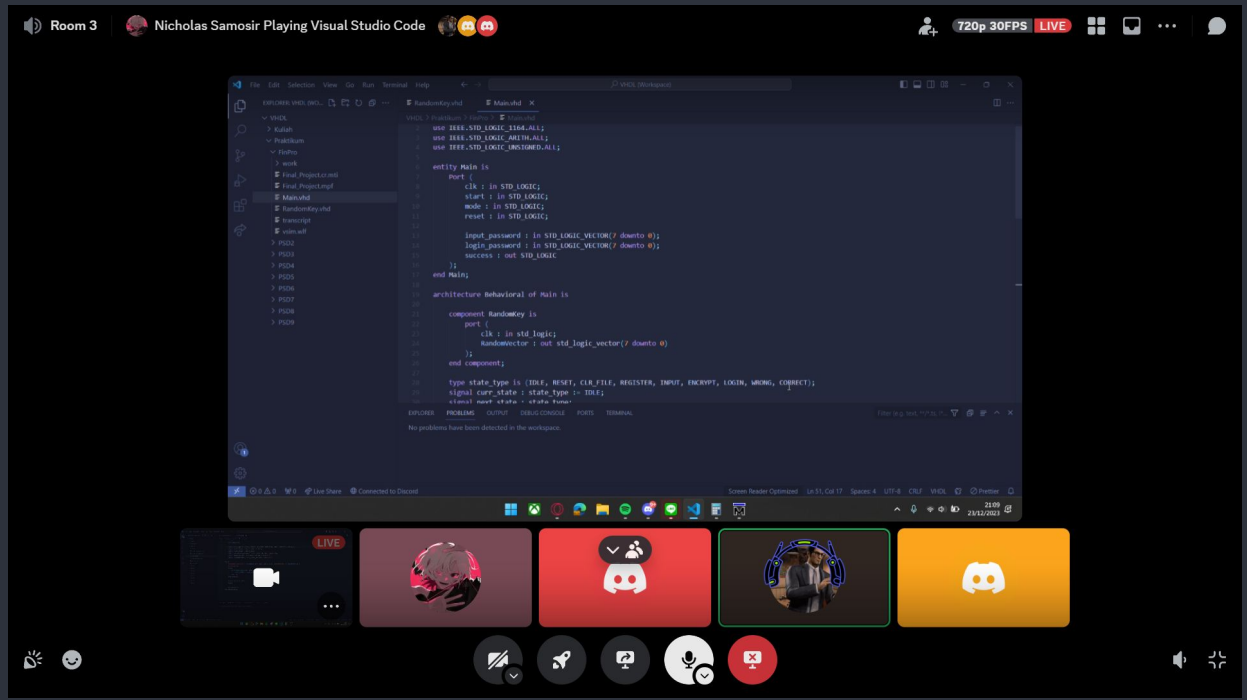
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Dokumentasi



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Dokumentasi

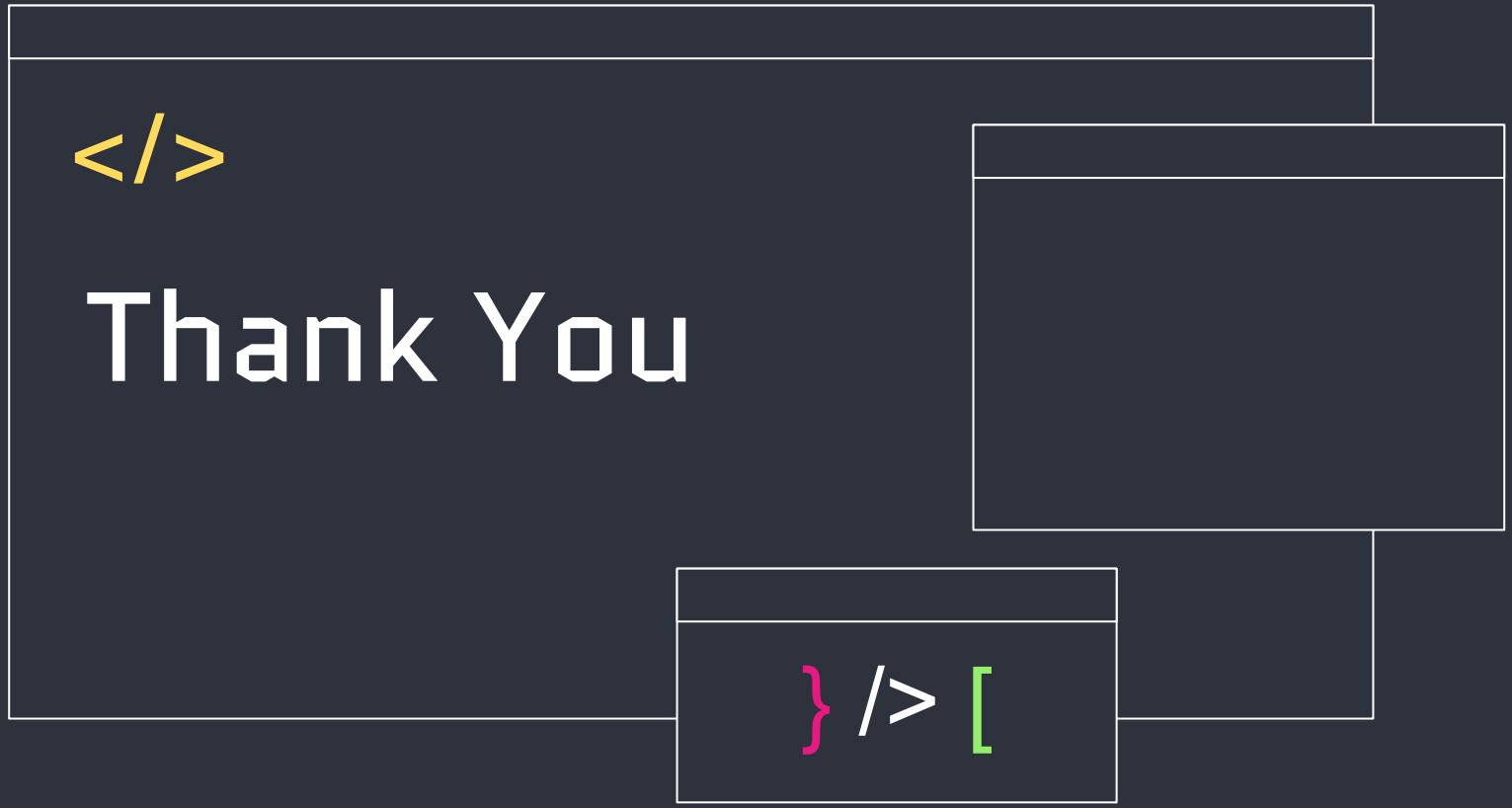


1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1

</ Reference

- S. Team et al., “VHDL component and Port Map tutorial,” Invent Logics, https://allaboutfpga.com/vhdl-component-port-map-tutorial/ (accessed Dec. 18, 2023).
- J. J. Jensen, “How to create a finite-state machine in VHDL,” VHDLwhiz, https://vhdlwhiz.com/finite-state-machine/ (accessed Dec. 18, 2023).
- “String,” VHDL - String, https://peterfab.com/ref/vhdl/vhdl_renerta/mobile/source/vhd00070.htm (accessed Dec. 18, 2023).
- J. J. Jensen, “How to use a for loop in VHDL,” VHDLwhiz, https://vhdlwhiz.com/for-loop/ (accessed Dec. 18, 2023).
- Russell, “VHDL Example Code of file Io,” Nandland, https://nandland.com/file-input-output/ (accessed Dec. 18, 2023).
- P. Loshin and M. Cobb, “What is encryption and how does it work? - techtarget,” Security, https://www.techtarget.com/searchsecurity/definition/encryption (accessed Dec. 17, 2023).
- J. J. Jensen, “How to generate random numbers in VHDL,” VHDLwhiz, https://vhdlwhiz.com/random-numbers/ (accessed Dec. 21, 2023).

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1


</>
Thank You
} /> [
1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1