FUNDAMENTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITAS INDONESIA

ELECTRONIC SLOT MACHINE with CREDIT MANAGER

GROUP IP-16

Ayesha Zelene Faeyza	**2406359166**
Diandra Pramesti Wicaksono	**2406342360**
Muhammad Rafif Batubara	**2406436253**
Radya Gardian Pranoto	**2406404592**

PREFACE

Puji syukur kami panjatkan kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan ini dengan baik. Laporan ini disusun dengan tujuan untuk memenuhi tugas akhir pada mata kuliah Perancangan Sistem Digital. Proyek ini berfokus pada pengembangan sistem “Electronic Slot Machine” dengan pengelolaan kredit. Dalam proyek ini, kami menggunakan bahasa VHDL untuk merancang dan mengimplementasikan sistem yang diperlukan.

Dalam laporan ini, kami akan membahas secara rinci mengenai desain, implementasi, dan pengujian slot machine yang kami kembangkan menggunakan VHDL. Kami harap laporan ini dapat memberikan wawasan yang bermanfaat bagi pembaca, serta menjadi referensi bagi penelitian dan pengembangan lebih lanjut di bidang teknologi dan sistem digital.

Kami menyadari bahwa penyelesaian laporan ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, kami ingin mengucapkan terima kasih kepada asisten laboratorium dari Digital Laboratorium yang telah memberikan bimbingan dan arahan, serta kepada teman-teman yang selalu mendukung dan memberikan motivasi selama proses pengerjaan proyek ini. Selain itu, kami juga berterima kasih kepada semua pihak yang telah berkontribusi dalam pengumpulan data dan informasi yang diperlukan.

Depok, December 7, 2025

Group IP-16

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Di zaman ini, banyak produk dalam industri *gaming* yang bergantung pada *randomness* untuk menciptakan sebuah suasana yang adil dan tidak terprediksi, contohnya dalam permainan *gacha*, *loot* yang muncul di tanah, dan sebagainya. Dalam sebuah sistem digital seperti FPGA, menghasilkan angka yang benar-benar acak merupakan tantangan tersendiri karena sifat deterministik dari logika digital. Salah satu solusi yang efisien untuk diimplementasikan adalah dengan Pseudo-Random Number Generator (PRNG).

Mesin slot adalah salah satu contoh sederhana sistem yang membutuhkan *randomness* dan *state management* yang konsisten untuk menangani koin, putaran, dan pembayaran. Dengan menggunakan VHDL, kita dapat merancang sistem kontrol mesin slot yang mencakup logika permainan, manajemen kredit, dan fitur administrasi untuk tujuan pengujian. Proyek ini mengeksplorasi penggunaan FSM untuk logika permainan dan LFSR untuk pembangkitan angka acak.

1.2 PROJECT DESCRIPTION

Proyek ini berjudul "Electronic Slot Machine with Credit Manager". Sistem ini mensimulasikan sebuah 3-reel slot machine, dimana mesin slot tersebut memiliki tiga item atau gulungan. Pengguna dapat memasukkan sebuah koin virtual untuk menambah kredit. Permainan dimulai ketika pengguna menekan pemicu *spin*, yang akan memutar angka pada reel.

Kami mengimplementasikan modul LFSR (*Linear Feedback Shift Register*) 16-bit pada proyek ini untuk menentukan hasil dari setiap reel setelah diacak. Fitur unik dari proyek yang kami buat adalah adanya input admin_cheat, yang memungkinkan pengguna untuk mendapatkan hasil "Jackpot" (7-7-7) yang dapat digunakan untuk demonstrasi atau *debugging*. Sistem juga memiliki mekanisme penghitungan hadiah otomatis berdasarkan kombinasi angka yang muncul.

1.3 OBJECTIVES

Tujuan dari proyek ini adalah sebagai berikut:

1. Mengimplementasikan Finite State Machine (FSM) untuk mengatur alur permainan (Idle, Spinning, Stopping, Evaluate, Payout).
2. Menerapkan algoritma LFSR untuk menghasilkan angka random pada FPGA.
3. Menguji coba fitur Admin yang digunakan untuk mendapatkan jackpot untuk fungsi demonstrasi dan/atau debugging

1.4 ROLES AND RESPONSIBILITIES

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

Roles	Responsibilities	Person
Ketua kelompok, lead programmer, readme file	- Membuat kode - Melakukan testing dan sintesis pada program - Membuat readme file	Muhammad Rafif Batubara
Pembuat Laporan dan membuat testbench	- Membuat dan menyusun laporan - Membuat slides presentasi - Membuat dan melakukan testing dengan testbench	Ayesha Zelene Faeyza
Pembuat Laporan dan membuat testbench	- Membuat dan menyusun laporan - Membuat slides presentasi	Diandra Pramesti Wicaksono

	- Membuat dan melakukan testing dengan testbench	
Pembuat Laporan, Source code programmer, readme file	- Membuat dan menyusun laporan - Membantu penulisan kode VHDL - Membuat readme file	Radya Gardian Pranoto

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Alat yang kami gunakan pada project ini sebagai berikut:

- VHDL (VHSIC Hardware Description Language) : Bahasa utama untuk mengimplementasikan logika slot machine, FSM, counter reels, dan credit manager.
- ModelSim : Simulator utama yang digunakan untuk menjalankan testbench, membaca file input, dan menghasilkan output log dari mesin slot.
- Quartus Prime 21.1 : Kompilasi, analisis, dan menghasilkan RTL viewer.
- GitHub : Tempat penyimpanan proyek.

2.2 IMPLEMENTATION

Komponen utama :

a. SlotMachine_Advanced, merupakan komponen utama yang mengelola seluruh mekanisme slot. Modul ini mengatur input coin, menjalankan proses spin, menghentikan reel, mengevaluasi hasil, sampai memberikan payout. SlotMachine_Advanced juga menangani tiga reel sekaligus dengan melakukan pembentukan angka acak menggunakan LFSR, serta menentukan status permainan (Idle, Spin, Win, Lose).
b. Reel Logic, merupakan bagian internal yang mengendalikan pergerakan masing-masing reel menggunakan counter cepat dan nilai pseudo-random dari LFSR. Reel 1, 2, dan 3 memiliki pola putaran berbeda sehingga menghasilkan efek spinning. Modul ini mengatur proses stopping reel dan menentukan angka akhir setelah spin selesai.
c. Credit Manager, merupakan sub-modul dalam SlotMachine_Advanced yang bertanggung jawab mengelola saldo pemain. Kredit akan bertambah ketika coin dimasukkan, berkurang otomatis saat spin dilakukan, dan bertambah

kembali ketika pemain menang. Credit manager juga memastikan bahwa saldo tidak bernilai negatif serta menangani pembacaan status payout.

d. Random Generator, merupakan komponen yang menghasilkan angka acak berbasis Linear Feedback Shift Register. LFSR di update pada setiap rising edge clock dan menghasilkan angka 0-9 yang digunakan sebagai input reel. LFSR ini berperan dalam modul generic untuk mengatur probabilitas kemenangan player.

Implementasi Tiap Modul

a. Modul 2 : Concurrent Statement dan Dataflow

Sistem menerapkan modul Concurrent dan Dataflow Style untuk beberapa bagian utama, terutama di mekanisme keluaran dan pembaruan sinyal yang berjalan paralel. Penggunaan concurrent terdapat pada:

- Assignment output yang mengikuti sinyal internal, berjalan concurrently dan memperbarui output tanpa menunggu urutan eksekusi dalam process

```
credit_saldo <= internal_credit;

   reel1_val <= reels(1);

   reel2_val <= reels(2);

   reel3_val <= reels(3);
```

- Pemodelan dataflow pada LFSR, yakni operasi shifting dan XOR

```
lfsr_reg(15 downto 1) <= lfsr_reg(14 downto 0);

                lfsr_reg(0) <= lfsr_reg(15) XOR
lfsr_reg(13) XOR lfsr_reg(12) XOR lfsr_reg(10);
```

- Instansiasi komponen di testbench yang berjalan secara concurrent. Unit Under Test (UUT), clock process, dan stimulus process bekerja bersamaan.

b. Modul 3 : Behavioral Style Modelling

Sistem slot machine dirancang menggunakan Behavioral Modelling, di mana seluruh proses permainan, pengelolaan kredit, logika reel, dan evaluasi kemenangan dideskripsikan melalui proses berbasis clock, state transition, dan signal assignment. Penerapan behavioral style pada sistem mencakup :

- Pembacaan nilai koin secara dinamis melalui port input, di mana setiap perubahan coin_in langsung menambah saldo internal jika pada state IDLE,
- Menjalankan proses reel spinning menggunakan counter spin_timer dan nilai pseudo-random dari LFSR untuk mengubah nilai masing-masing reel setiap clock,
- Mengatur perpindahan state FSM ketika tombol spin ditekan,
- Menghasilkan nilai reel akhir pada state “STOPPING”,
- Menangani mode cheat untuk memaksa keluaran reel menjadi kombinasi jackpot 7-7-7,
- Melakukan evaluasi pola kemenangan menggunakan function calculate_reward, yang menentukan nilai hadiah berdasarkan kombinasi reel,
- Memperbarui saldo kredit pengguna pada state “EVALUATE” dan “PAYOUT” sesuai hasil spin.

c. Modul 4 : Testbench & File I/O

Project ini menerapkan testbench untuk menguji perilaku slot machine menggunakan skenario nyata dari pengguna. Terdapat pada file tb_SlotMachine, serta implementasi I/O meliputi :

- File Input (input_cmds.txt)

  Testbench membaca perintah pemain secara berurutan menggunakan :

```
readline(input_file, line_in);
          read(line_in, cmd_code);
          read(line_in, val_in);
```

  Perintah ini mencakup insert coin, spin, enable cheat, dan disable cheat.

- File Output (output_log.txt)

Terdapat dalam kode :

```
write(line_out, string'("..."));
                    write(line_out, credit);
                                writeline(output_file,
line_out);
```

Setiap perilaku dan hasil spin ditulis ke file melalui kode di atas.

d. Modul 5 : Type dan Array

Type dan array digunakan untuk menyusun data internal agar lebih mudah dikelola dan modular. Arrau Reel digunakan untuk menyimpan semua nilai reel sekaligus sehingga mengurangi jumlah sinyal yang harus dituliskan dan memudahkan passing ke function reward evaluator. Sistem juga menggunakan Enumeration Type untuk FSM agar lebih terbaca, lebih mudah dipelihara, dan lebih aman dibandingkan angka integer raw.

Penggunaan dalam kode :

```
type Reel_Array is array (1 to 3) of INTEGER range 0
to 9;
type State_Type is (IDLE, SPINNING, STOPPING,
EVALUATE, PAYOUT);
```

e. Modul 6 : Looping

Looping digunakan dalam kode testbench. Looping pada testbench untuk memproses seluruh instruksi dari file input. Pada sistem, digunakan while untuk membaca tiap baris peintah pengguna hingga file selesai. Setiap iterasinya menyimulasikan aksi yang berbeda, menulis hasil ke file output, dan mensimulasikan alur permainan.

Implementasi pada kode :

```
while not endfile(input_file) loop
..
end loop
```

f. Modul 7 : Function

Modul function digunakan untuk memisahkan logika evaluasi kemenangan sehingga desain lebih modular dan mudah di-*maintenance*. Function ini dipanggil dalam FSM pada state EVALUATE dan PAYOUT, kode menggunakan function karena perhitungan bergantung hanya pada input formal function dan tidak mengakses sinyal global apapun.

Function calculate_reward berfungsi untuk menentukan *reward* berdasarkan pola tiga reel yang muncul setelah proses STOPPING. Logika yang dicek antara lain:

- 777 untuk jackpot dan mendapatkan 500 kredit,
- Triple (selain 7) mendapatkan 100 kredit,
- Pair (dua angka yang sama) mendapatkan 20 kredit,
- Kombinasi lainnya tidak mendapatkan kredit apapun (0).

Penggunaan dalam kode :

```
function calculate_reward(current_reels : Reel_Array; bet_cost : integer) return integer is
   begin
                if (current_reels(1) = 7) and (current_reels(2) = 7) and (current_reels(3) = 7) then
           return 500;
        elsif (current_reels(1) = current_reels(2)) and (current_reels(2) = current_reels(3)) then
           return 100;
        elsif (current_reels(1) = current_reels(2)) or (current_reels(2) = current_reels(3)) or (current_reels(1) = current_reels(3)) then
           return 20;
       else
```

```
            return 0;
        end if;
    end function;
```

g. Modul 9 : Hardwired Control Unit (FSM)

Sistem slot machine menggunakan Hardwired Control Unit dengan Finite State Machine untuk mengatur seluruh alur permainan secara sinkron berdasarkan sinyal clock. FSM akan memastikan setiap tahapan dilakukan berurutan dan konsisten, mulai dari memasukkan koin sampai menentukan hasil akhir. FSM terdapat 5 state utama, yakni :

- IDLE : state awal ketika mesin menunggu perintah dari pengguna. Fungsinya untuk menerima nilai koin melalui coin_in, menambahkan kredit ke saldo internal, menunggu tombol spin_trig, dan memastikan saldo cukup sebelum memulai permainan. Jika spin_trig bernilai 1, dan kreditnya minimal 10, FSM akan berpindah ke SPINNING.
- SPINNING : Pada state ini, ketiga reel akan diputar untuk dengan menggunakan counter dan LFSR. Ketika spin_timer lebih dari 10, FSM akan berpindah ke STOPPING.
  - reels(1), increment 1 tiap clock
  - reels(2), bertambah pseudo-random dari nilai LFSR
  - reels(3), increment 2 tiap clock
- STOPPING : mesin berhenti memutar reel dan menentukan angka final untuk ketiga reel. Jika admin_cheat bernilai 1, reel dipaksa mengeluarkan kombinasi 7-7-7.
- EVALUATE : Setelah nilai final tersimpan FSM memasuki evaluate. Hadiah akan dihitung menggunakan function calculate_reward(). State ini untuk menentukan apa kombinasi reel (jackpot, triple, pair, atau lose), serta menambahkan reward ke internal_credit.
- PAYOUT : State terakhir sebelum kembali ke IDLE. Fungsi state ini untuk memperbarui sinyal game_status (“111” untuk WIN, “100” untuk LOSE), menampilkan status permainan, dan mengembalikan kontrol ke IDLE.

Implementasi Kode

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

entity SlotMachine_Advanced is
    Port (
        clk           : in  STD_LOGIC;
        reset         : in  STD_LOGIC;
        coin_in       : in  INTEGER range 0 to 1000;
        spin_trig     : in  STD_LOGIC;
        admin_cheat   : in  STD_LOGIC;
        reel1_val     : out INTEGER range 0 to 9;
        reel2_val     : out INTEGER range 0 to 9;
        reel3_val     : out INTEGER range 0 to 9;
        credit_saldo  : out INTEGER;
        game_status   : out STD_LOGIC_VECTOR(2 downto 0)
    );
end SlotMachine_Advanced;

architecture Behavioral of SlotMachine_Advanced is

    type Reel_Array is array (1 to 3) of INTEGER range 0 to 9;
    signal reels : Reel_Array := (0, 0, 0);
```

```
    type State_Type is (IDLE, SPINNING, STOPPING, EVALUATE,
PAYOUT);
   signal current_state : State_Type := IDLE;


   signal internal_credit : INTEGER := 0;
   signal spin_timer      : INTEGER := 0;
    signal lfsr_reg         : STD_LOGIC_VECTOR(15 downto 0) :=
x"ACE1";
   signal rand_val        : INTEGER range 0 to 9;


      function  calculate_reward(current_reels  :  Reel_Array;
bet_cost : integer) return integer is
   begin
         if (current_reels(1) = 7) and (current_reels(2) = 7)
and (current_reels(3) = 7) then
           return 500;
            elsif  (current_reels(1)  =  current_reels(2))  and
(current_reels(2) = current_reels(3)) then
           return 100;
             elsif  (current_reels(1)  =  current_reels(2))  or
(current_reels(2)  =  current_reels(3))  or  (current_reels(1)  =
current_reels(3)) then
           return 20;
       else
           return 0;
       end if;
   end function;
```

```
begin


   process(clk, reset)
   begin
       if reset = '1' then
           current_state <= IDLE;
           internal_credit <= 0;
           lfsr_reg <= x"ACE1";
           reels <= (0, 0, 0);
           game_status <= "000";

       elsif rising_edge(clk) then

           lfsr_reg(15 downto 1) <= lfsr_reg(14 downto 0);
              lfsr_reg(0) <= lfsr_reg(15) XOR lfsr_reg(13) XOR
lfsr_reg(12) XOR lfsr_reg(10);
              rand_val <= to_integer(unsigned(lfsr_reg(3 downto
0))) mod 10;

           case current_state is

               when IDLE =>
                   game_status <= "000";
                   if coin_in > 0 then
```

```
                        internal_credit <= internal_credit +
coin_in;

                        report "[DESIGN] Coin Accepted. Current
Credit: " & integer'image(internal_credit + coin_in);
                    end if;

                    if spin_trig = '1' and internal_credit >=
10 then
                        internal_credit <= internal_credit -
10;
                        spin_timer <= 0;
                        current_state <= SPINNING;

                            report "[DESIGN] Spin Triggered!
Deducting 10 credits.";
                    end if;

                when SPINNING =>
                    game_status <= "001";
                    spin_timer <= spin_timer + 1;

                    reels(1) <= (reels(1) + 1) mod 10;
                    reels(2) <= (reels(2) + rand_val) mod 10;
                    reels(3) <= (reels(3) + 2) mod 10;
```

```
                    if spin_timer > 10 then
                        current_state <= STOPPING;
                    end if;

                when STOPPING =>
                    if admin_cheat = '1' then
                        reels <= (7, 7, 7);

                         report "[DESIGN] ADMIN CHEAT DETECTED!
Forcing Jackpot.";
                    else
                                                    reels(1)  <=
to_integer(unsigned(lfsr_reg(3 downto 0))) mod 10;
                                                    reels(2)  <=
to_integer(unsigned(lfsr_reg(7 downto 4))) mod 10;
                                                    reels(3)  <=
to_integer(unsigned(lfsr_reg(11 downto 8))) mod 10;
                    end if;
                    current_state <= EVALUATE;

                when EVALUATE =>
                          internal_credit <= internal_credit +
calculate_reward(reels, 10);
                    current_state <= PAYOUT;

                when PAYOUT =>
```

```
                    if calculate_reward(reels, 10) > 0 then
                        game_status <= "111";

                           report "[DESIGN] WINNER! Reward: " &
integer'image(calculate_reward(reels, 10));
                    else
                        game_status <= "100";

                        report "[DESIGN] No Win.";
                    end if;
                    current_state <= IDLE;

            end case;
        end if;
    end process;

    credit_saldo <= internal_credit;
    reel1_val <= reels(1);
    reel2_val <= reels(2);
    reel3_val <= reels(3);

end Behavioral;
```

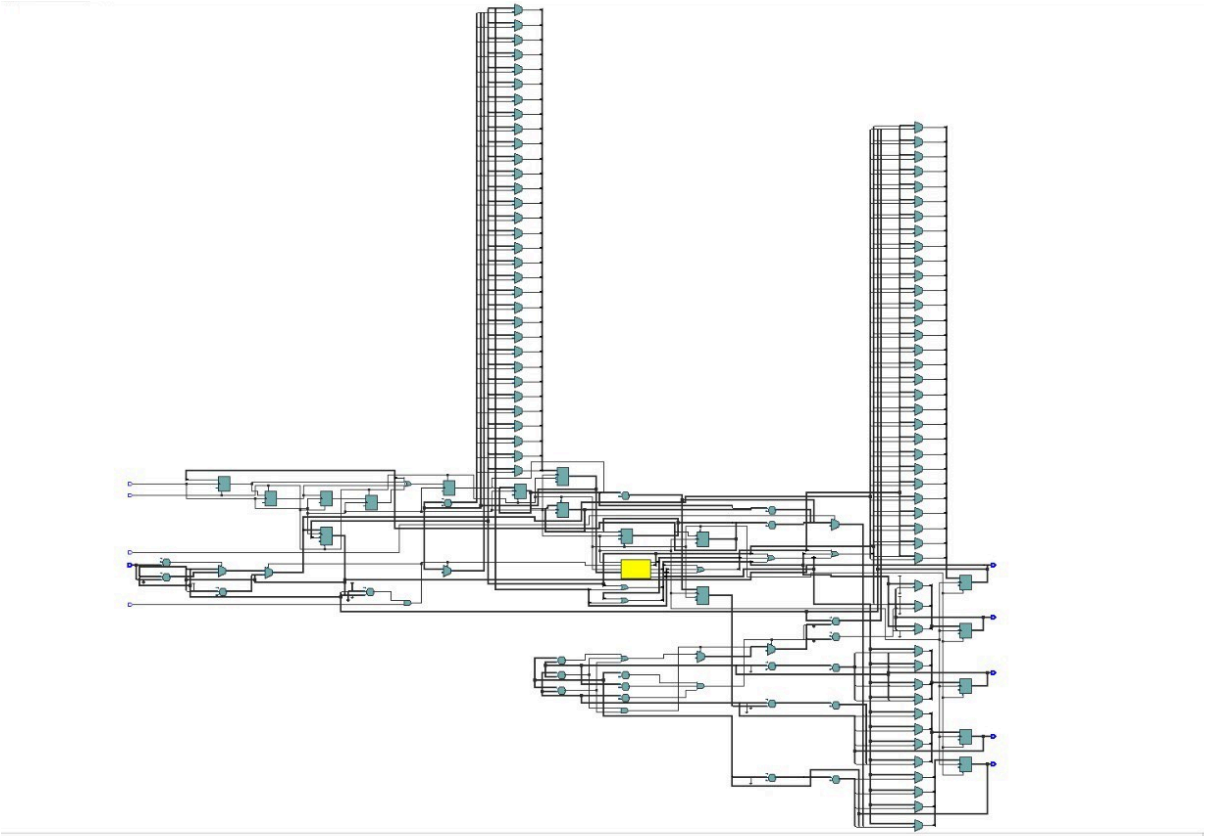


Fig 1. RTL Design

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Proses verifikasi dan pengujian modul utama SlotMachine_Advanced.vhdl dilakukan menggunakan kode *testbench* VHDL. Kami memanfaatkan file .txt eksternal yang disimpan dalam directory yang sama dengan kode VHDL agar pengujian lebih fleksibel dan efisien, alih-alih menulis kode statis yang panjang pada *testbench* untuk setiap kasus uji seperti biasa. Dengan ini, kami dapat mengeksplorasi dan memodifikasi berbagai konfigurasi hanya dengan mengedit file teks tersebut. Jadi tidak ada kebutuhan untuk mengkompilasi kode berulang kali. Kode *testbench* yang dimaksud dapat dilihat dibawah:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use std.textio.all;

entity tb_SlotMachine is
end tb_SlotMachine;

architecture Behavioral of tb_SlotMachine is

    component SlotMachine_Advanced is
        Port (
            clk           : in  STD_LOGIC;
            reset         : in  STD_LOGIC;
            coin_in       : in  INTEGER;
            spin_trig     : in  STD_LOGIC;
            admin_cheat   : in  STD_LOGIC;
            reel1_val     : out INTEGER;
            reel2_val     : out INTEGER;
            reel3_val     : out INTEGER;
            credit_saldo  : out INTEGER;
```

```
            game_status   : out STD_LOGIC_VECTOR(2 downto 0)
        );
    end component;

     signal clk, reset, spin_trig, admin_cheat : STD_LOGIC :=
'0';
    signal coin_in : INTEGER := 0;
    signal r1, r2, r3, credit : INTEGER;
    signal status : STD_LOGIC_VECTOR(2 downto 0);

    constant clk_period : time := 10 ns;

begin

    UUT: SlotMachine_Advanced
    port map (
        clk => clk, reset => reset, coin_in => coin_in,
        spin_trig => spin_trig, admin_cheat => admin_cheat,
        reel1_val => r1, reel2_val => r2, reel3_val => r3,
        credit_saldo => credit, game_status => status
    );

    clk_process : process
    begin
        clk <= '0'; wait for clk_period/2;
        clk <= '1'; wait for clk_period/2;
    end process;

    stim_proc: process
               file  input_file    :  text  open  read_mode  is
"input_cmds.txt";
               file  output_file   :  text  open  write_mode  is
"output_log.txt";
        variable line_in, line_out : line;
```

```
        variable cmd_code : integer;
        variable val_in   : integer;

    begin
        -- Initial Reset
        reset <= '1'; wait for 20 ns; reset <= '0';
        wait for 20 ns;

        report "=======================================";
        report "   STARTING SLOT MACHINE SIMULATION    ";
        report "=======================================";

        -- Header for file output
            write(line_out, string'("--- SIMULATION LOG START
---"));
        writeline(output_file, line_out);

        while not endfile(input_file) loop
            readline(input_file, line_in);
            read(line_in, cmd_code);
            read(line_in, val_in);

            case cmd_code is
                when 1 =>
                    coin_in <= val_in;
                    wait for clk_period;
                    coin_in <= 0;


                      report "[TESTBENCH] Action: Inserting " &
integer'image(val_in) & " coins.";
```

```
                    write(line_out, string'("ACTION: Insert
Coin -> Total Credit: "));
                write(line_out, credit);
                writeline(output_file, line_out);

            when 2 =>

                report "[TESTBENCH] Action: Pressing SPIN
Button.";

                spin_trig <= '1';
                wait for clk_period * 2;
                spin_trig <= '0';

                wait for clk_period * 20;


                        report "[TESTBENCH] Result: " &
integer'image(r1) & "-" & integer'image(r2) & "-" &
integer'image(r3);


                    write(line_out, string'("RESULT: Reels
["));
                      write(line_out, r1); write(line_out,
string'("-"));
                      write(line_out, r2); write(line_out,
string'("-"));
                      write(line_out, r3); write(line_out,
string'("] "));

                if status = "111" then
                    write(line_out, string'("WINNER! "));
```

```
                                report "[TESTBENCH] >>> JACKPOT
WINNER! <<<";
                    else
                        write(line_out, string'("LOST. "));
                        report "[TESTBENCH] You Lost.";
                    end if;

                      write(line_out, string'("Current Credit:
")); write(line_out, credit);
                    writeline(output_file, line_out);

                when 3 =>
                    admin_cheat <= '1';
                          report "[TESTBENCH] *** ADMIN CHEAT
ENABLED ***";
                        write(line_out, string'("ACTION: ADMIN
CHEAT ENABLED"));
                    writeline(output_file, line_out);

                when 4 =>
                    admin_cheat <= '0';
                          report "[TESTBENCH] *** ADMIN CHEAT
DISABLED ***";
                        write(line_out, string'("ACTION: ADMIN
CHEAT DISABLED"));
                    writeline(output_file, line_out);

                when others => null;
            end case;

            wait for 20 ns;
        end loop;

        report "====================================";
```

```
        report "    SIMULATION COMPLETED SUCCESSFULLY  ";
        report "=======================================";

        write(line_out, string'("--- SIMULATION END ---"));
        writeline(output_file, line_out);
        wait;
    end process;
end Behavioral;
```

3.2 RESULT

Dalam implementasi pengujian berbasis file, data input untuk *testbench* diatur sebagai sekelompok nilai diskrit dalam file eksternal tersebut. Kami menggunakan urutan 1-2-2-3-2-4-2 untuk merepresentasikan rangkaian instruksi pada modul utama. Command '1' mengindikasikan instruksi yang memerlukan input data tambahan untuk menentukan nilai koin ('val_in'). Sementara, instruksi lainnya dalam urutan tersebut seperti 2, 3, dan 4 merepresentasikan instruksi operasional modul yang tidak memerlukan nilai input dari pengguna, jadi akan dieksekusi berdasarkan status internal modul.

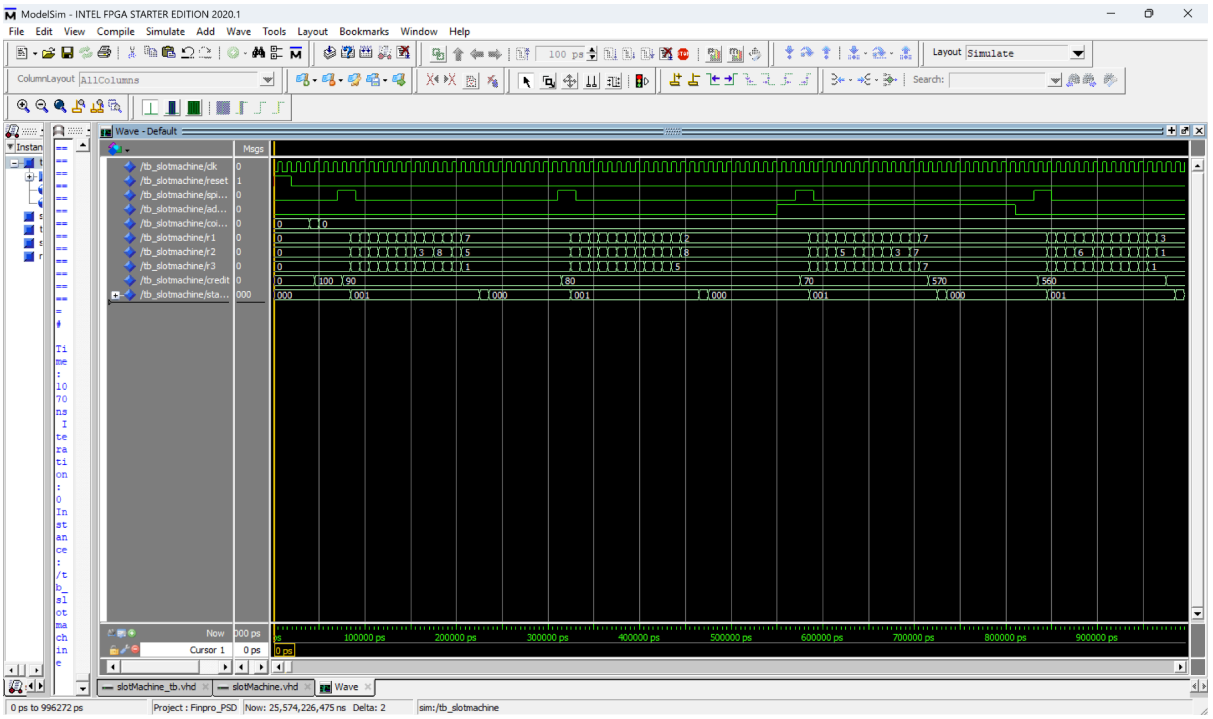


Fig 2. Testing Result

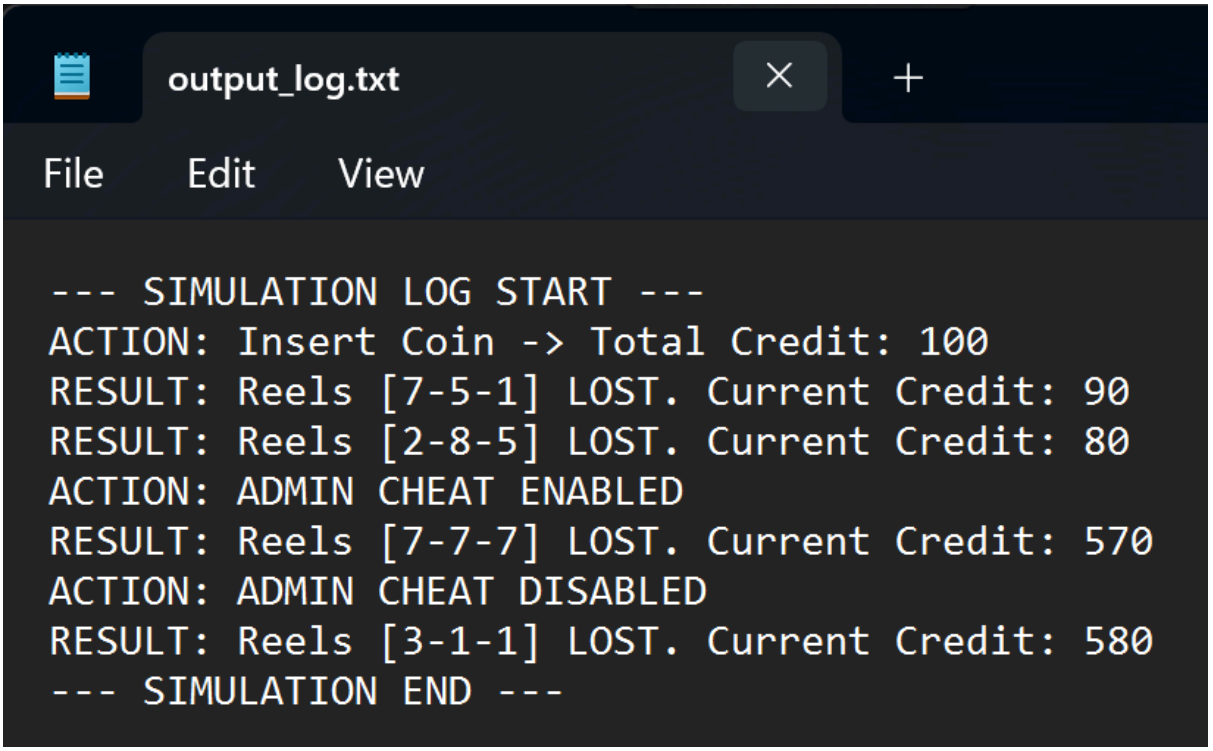


Fig 3. Testing Result Text Output

3.3 ANALYSIS

Testbench menjadi penghubung antara modul utama (‘*hardware*’) dan program internal (‘*software*’) dimana sinyal pada *testbench* akan disambung dengan port pada modul utama. Testbench akan mengirim sinyal seperti clk, reset, spin_trig, admin_cheat, dan coin_in untuk mengendalikan mesin ketika menerima sinyal output seperti nilai pada reel (r1, r2, r3), saldo kredit (credit), dan status game (status). Hubungan ini memungkinkan testbench untuk mengeksekusi berbagai skenario secara otomatis dengan membaca command pada file teks dan mencatat hasil outputnya.

Ketika testbench membaca command pemasukan koin (‘1’ diikuti dengan jumlah koin), ia akan menentukan sinyal coin_in berdasarkan jumlah yang dimasukkan untuk 1 siklus clock (10 ns). Lalu, testbench akan mencatat kegiatan tersebut dan menunjukkan total kredit terbaru. Untuk command spin (‘2’), testbench akan “menyalakan” sinyal spin_trig (‘1’) untuk 2 siklus clock lalu menunggu 20 siklus clock untuk menyelesaikan operasi spin. Disini, testbench akan melakukan operasi yang lebih kompleks berdasarkan kode pada modul utama. Command ‘3’ dan ‘4’ berfokus pada instruksi untuk memaksa sistem dimana kode mengatur sinyal admin_cheat menjadi ‘1’ atau ‘0’. Saat sinyal bernilai 1 (*enabled*), spin selanjutnya akan otomatis menunjukan (7-7-7) saat dalam status stopping.

Pada kasus uji yang kami lakukan, 100 kredit dimasukkan, slot berputar sebanyak 2 kali, *enable* mode cheat, slot akan berputar sekali, lalu mode cheat di-*disable* sebelum slot berputar untuk terakhir kalinya. Output akan menunjukan kredit awal bernilai 100, kalah dalam putaran pertama (7-5-1) dan kredit menjadi 90, putaran kedua juga kalah (2-8-5) menyisakan 80 kredit. Ketika mode cheat di-*enable*, putaran selanjutnya menang (7-7-7) dan kredit naik menjadi 570. Namun, ketika mode cheat kembali di-*disable*, putaran selanjutnya kalah lagi (3-1-1) dan kredit akhir menjadi 560. Ini mendemonstrasikan bagaimana kredit bergerak di sistem dan mode cheat menjamin kemenangan.

CHAPTER 4

CONCLUSION

Pada proyek akhir ini kami mengimplementasikan *Finite State Machine* (FSM) dan *Linear Feedback Shift Register* (LFSR) 16-bit sebagai pengatur alur sistem permainan dan generator pseudo-random pada “Electronic Slot Machine” berbasis VHDL. Modul utama didesain dan diintegrasikan secara modular agar sistem dapat bekerja dengan stabil, efisien, dan realistis dalam mensimulasikan mekanisme sebuah *slot machine*. Penggunaan concurrent, dataflow, dan behavioral modelling memastikan bahwa setiap proses berjalan paralel dan mengikuti pola kerja hardware digital.

Dengan testbench yang memanfaatkan file I/O eksternal, proses pengujian dilakukan secara fleksibel, terstruktur, dan mudah digunakan kembali. Testbench membaca instruksi dari file input dan mencatat output ke file log, jadi dapat dianalisis dengan jelas terhadap setiap kegiatan seperti memasukkan koin, spin, dan *enable/disable* mode cheat. Hasil pengujian membuktikan bahwa sistem dapat mengelola kredit dengan benar, menghasilkan nilai reel yang acak, menjalankan status game dengan konsisten, serta memaksa kondisi jackpot (7-7-7) ketika admit_cheat diaktifkan.

Secara keseluruhan, proyek ini menunjukkan integrasi yang efektif antara logika digital, pengelolaan state, dan pseudo-random number generator pada FPGA. Sistem *slot machine* yang dibuat tidak hanya memenuhi seluruh objektif, tetapi juga memberikan wawasan praktis terkait desain hardware terprogram, modularitas kode VHDL, serta strategi pengujian menggunakan testbench berbasis file eksternal.

REFERENCES

[1] "100+ VHDL Projects with Source Code for ME MTech Students," Mtechprojects.com, 2022. https://www.mtechprojects.com/ieee-vhdl-projects.html (accessed Nov. 28, 2025).

[2] Digilab, "Digital System Design," Digilabdte.com, 2025. https://learn.digilabdte.com/books/digital-system-design (accessed Nov. 30, 2025).

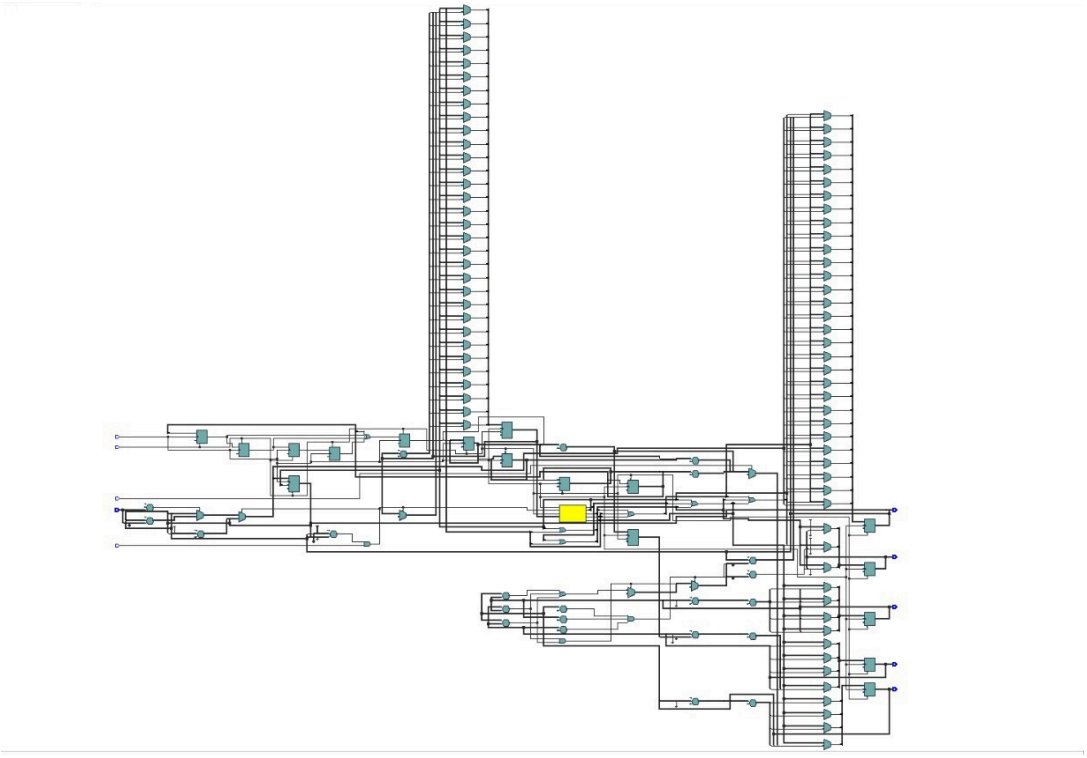
[3] "VHDL Behavioral Modeling Style," Surf-VHDL. https://surf-vhdl.com/vhdl-syntax-web-course-surf-vhdl/vhdl-behavioral-modeling-style/ (accessed Nov. 30, 2025).

[4] "Mastering loops in VHDL: Enhancing flexibility and performance in hardware design," FPGA Insights, https://fpgainsights.com/fpga/mastering-loops-in-vhdl-enhancing-flexibility-and-performance-in-hardware-design (accessed Dec. 1, 2025).

[5] "Implementing a finite state machine in VHDL" All About Circuits, https://www.allaboutcircuits.com/technical-articles/implementing-a-finite-state-machine-in-vhdl/ (accessed Dec. 1, 2025).

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation