**Code**

* **Design:**

module password\_RAM\_ROM (

input wire clk,

input wire rst,

input wire [7:0] password\_input,

input wire [7:0] data\_input,

input wire write\_enable,

input wire read\_enable,

output reg [7:0] data\_output

);

// Parameters

parameter [7:0] RAM\_PASSWORD = 8'h1x111111;//ff or bf

parameter [7:0] ROM\_PASSWORD = 8'h0x111110;//7e or 3e

parameter Ram\_file = "ram\_file.txt";

parameter Rom\_file = "rom\_file.txt";

// Internal signals

reg [7:0] password\_buffer;

reg [2:0] state;

reg [4:0] file\_handle; // file handle variable that stores the name of file

reg file\_open;

// RAM and ROM storage

reg [7:0] ram\_data;

reg [7:0] rom\_data;

always @(posedge clk or posedge rst) begin

if (rst) begin

state <= 3'b000;

password\_buffer <= 8'b0;

file\_handle <= 5'b0; // Initialize file handle

file\_open <= 1'b0;

data\_output <= 8'b0; // Clear data\_output on reset

end else begin

case (state)

3'b000: begin // Idle state

if (password\_input[7] == 1'b1) begin

state <= 3'b001; // RAM mode

end else if (password\_input[7] == 1'b0) begin

state <= 3'b010; // ROM mode

end

end

3'b001: begin // RAM Editing/Reading state

if (password\_input[6] == 1'b1) begin

state <= 3'b011; // RAM Reading mode

end else begin

password\_buffer <= password\_input[7:0];// Store password for RAM writing

state <= 3'b011; // Move to RAM Writing mode

end

end

3'b010: begin // ROM Editing/Reading state

if (password\_input[6] == 1'b1) begin

state <= 3'b100; // ROM Reading mode

end else begin

password\_buffer <= password\_input[7:0]; // Store password for ROM writing

state <= 3'b100; // ROM Reading mode

end

end

3'b011, 3'b100: begin // Data editing/reading state

if (password\_input[6] == 1'b0) begin // Write mode

if (password\_input[7:0] == password\_buffer) begin

if (state == 3'b011) begin // RAM Writing mode

ram\_data <= data\_input; // Update RAM data

end else if (state == 3'b100) begin // ROM Writing mode

rom\_data <= data\_input; // Update ROM data

end

end

end

state <= 3'b000; // Return to idle state

end

default: state <= 3'b000; // Default to idle state

endcase

// Simulate file operations for RAM

if ((state == 3'b011 || state == 3'b100)) begin

file\_handle = $fopen((state == 3'b011) ? Ram\_file : Rom\_file, "r"); // Open file for reading

if (file\_handle != 5'b0) begin

// Read data from file

if (state == 3'b011) begin

$fscanf(file\_handle, "%h", ram\_data);

end

else begin

$fscanf(file\_handle, "%h", rom\_data);

end

$fclose(file\_handle); // Close file after reading

end

end

if (((state == 3'b011 && (password\_input[5:0]==RAM\_PASSWORD[5:0]))|| (state == 3'b100 && (password\_input[5:0]==ROM\_PASSWORD[5:0]))) && write\_enable) begin

file\_handle = $fopen((state == 3'b011) ? Ram\_file : Rom\_file, "w"); // Open file for writing

if (file\_handle != 5'b0) begin

// Write data to file

$fwrite(file\_handle, "%h\n", (state == 3'b011) ? ram\_data : rom\_data);

file\_open <= 1'b1;

end

end

if (state == 3'b000 && file\_open) begin

$fdisplay((state == 3'b011) ? Ram\_file : Rom\_file, "File closed");

file\_open <= 1'b0;

end

end

end

// Output ROM data in ROM mode only when read\_enable is 1

always @(posedge clk) begin

if (read\_enable && state == 3'b010) begin

data\_output <= rom\_data;

end else if (read\_enable && state == 3'b001)begin

data\_output <= ram\_data;

end else begin

data\_output <= 8'b0;

end

end

endmodule