```
// ELEC402_PRJ1_SystemVerilog_FSM_Project
// Project Name: Food Food_Dispenser
// Name: Xingwei Su
// File: Food_Dispenser_FSM
// Description: Design's main logical FSM

module food_dispenser_fsm (
    input reset,                    //High when reset
    input clk,                      //Base clk input (fast)
    input timesup,                  //High when count time is up
    input [6:0] food_weight,        //Foot weight measure under plate
    input [6:0] set_food_weight,    //Initialized target food weight (set on UI interface
module)
    input refill_detector,          //High when need refill
    input cap_detector,             //High when cap open
    input play_function_pedal,      //High when one full press cycle (press+release) is done
    input initialize_flag,          //High when UI interface module finish initialize
    input newday,                   //Posedge when new day

    output logic food_gate,         //High when gate open
    output logic warning,           //High when warning
    output logic play_function_flag,            //High when doing play function
    output logic play_function_fail_flag        //Posedge when fail
);

//Logic parameter setting
logic [3:0]
state;                                                                  //15 states
Max
logic [3:0] play_function_counter;                              //Counter counts to 15, daily
play function limit 15 times

//State Parameters
parameter [3:0]     initialize_state                = 4'd0;
parameter [3:0]     initialize_wait_for_finish      = 4'd1;
parameter [3:0]     idle                            = 4'd2;

parameter [3:0]     before_open_gate                = 4'd3;
parameter [3:0]     add_food                        = 4'd4;
parameter [3:0]     weight_detect                   = 4'd5;
parameter [3:0]     add_food_complete               = 4'd6;

parameter [3:0]     wait_for_refill                 = 4'd7;

parameter [3:0]     play_function_detected          = 4'd8;
parameter [3:0]     play_function_fail              = 4'd9;
parameter [3:0]     play_function_complete          = 4'd10;

parameter [3:0]     newday_reset                    = 4'd11;
```

```
//Value parameter
parameter [6:0]     play_function_1g_food           = 7'd1;

//FSM flip-flop
// always_ff @(posedge clk) begin
//     if (reset)  state <= 3'd0;
//     else begin
//         state <= nextstate;
//         play_function_counter <= next_play_function_counter;
//     end
// end

//FSM
always_ff @(posedge clk or posedge reset) begin

    if (reset)
        state <= initialize_state;
    else begin
        case (state)

            //Output values preset
            initialize_state: begin
                food_gate <= 1'b0;
                warning <= 1'b0;
                play_function_flag <= 1'b0;
                play_function_fail_flag <= 1'b0;
                play_function_counter <= 1'b0;
                state <= initialize_wait_for_finish;
            end

            //Wait for UI initialization
            initialize_wait_for_finish:
                state <= initialize_flag ? idle : initialize_wait_for_finish;

            //Idle stage (detect flags)
            idle:
                state <= newday ? newday_reset :
                            (refill_detector ? wait_for_refill :
                            (timesup ? before_open_gate :
                            (play_function_pedal ? play_function_detected : idle)));

            //Detect if there is still food left on plate
            //Warning if more than 1/3 target food weight is left
            before_open_gate: begin
                warning <= (food_weight > (set_food_weight / 2'd3)) ? 1'b1 : 1'b0;
                state <= add_food;
            end
```

```
            //Gate open
            add_food: begin
                food_gate <= 1'b1;
                state <= weight_detect;
            end

            //Detect the weight of the food
            //If on Play mode, only 1g of food will be send to plate
            //If equal close gate
            weight_detect:
                if (play_function_flag)
                    state <= (play_function_1g_food == food_weight) ? add_food_complete :
weight_detect;
                else
                    state <= (set_food_weight == food_weight) ? add_food_complete :
weight_detect;

            //Gate close
            add_food_complete: begin
                food_gate <= 1'b0;
                state <= play_function_flag ? play_function_complete : idle;
            end

            //Refill food in bank
            //Wait for cap to be closed to continue
            wait_for_refill:        state <= refill_detector ? wait_for_refill :
                                             (cap_detector ? wait_for_refill : idle);

            //Pedal trigered
            //Open function
            play_function_detected: begin
                play_function_fail_flag <= 1'b0;
                if (play_function_counter == 4'd15)
                    state <= play_function_fail;
                else begin
                    play_function_flag <= 1'b1;
                    state <= add_food;
                end
            end

            //More than 15 times daily
            play_function_fail:     {state, play_function_fail_flag} <= {idle, 1'b1};
                //blink LED (in module LED) <- test for posedge fail flag

            //Count up daily play times counter
            // Close function
            play_function_complete: begin
                play_function_counter <= play_function_counter + 1'b1;
                play_function_flag <= 1'b0;
```

```
                state <= idle;
            end

            newday_reset: begin
                play_function_counter <= 1'b0;
                state <= idle;
            end

            //Default to initialization
            default:    state <= initialize_state;
        endcase
    end
end

endmodule
```