1 FSM Code

Below is the code for my FSM.

```
/* ELEC 402 Project 1 - System Verilog FSM Project
 * Name: Arthur Hsueh
 * ID: 21582168
 * Description: This fsm is a generalized credit card
 *              payment controller, capable of taking payments
 *              in visa, mastercard and amex.
*/

module credit_card_payment_fsm
(
    input clk, reset,
    input process_init,
    input visa_choice_in,
    input mastercard_choice_in,
    input amex_choice_in,
    input pymt_amt_conf,
    input pymt_amt_denied,
    input pin_fail,
    input pin_success,
    input transaction_fail,
    input transaction_success,
    output reg light_bit,
    output reg [1:0] card_choice,
    output reg pymt_amt_print,
    output reg pin_process_init,
    output reg pymt_process_init,
    output reg process_abort
);

// wire delclarations
logic [1:0] fail_counter; // fail counter, used for pin fail detection
reg [11:0] state;        // state wire, will hold the value of all localparams

// state parameter definition
// This glitch-free method was taught by Dr. Yair Linn, in
// CPEN 311.
                            //   12'b1098_7_6543210
localparam idle                 = 12'b0000_0_0000000; // idle state wait for process_init
localparam init_process         = 12'b0001_0_1000000; // starts the process
localparam wait_credit_choice   = 12'b0010_0_1000000; // waits for the user input of their credit card
    payment choice
localparam choice_visa          = 12'b0011_0_1010000; // user has chosen visa, output bits will be driven
localparam choice_mastercard    = 12'b0100_0_1100000; // user has chosen mastercard, output bits will be
    driven
localparam choice_amex          = 12'b0101_0_1110000; // use has chosen amex, output bits will be driven
localparam init_amount_confirm  = 12'b0110_0_1001000; // initates the confirmation of the amount to be
    paid
localparam wait_amount_confirm  = 12'b0111_0_1000000; // wait fro the use to confirm the payment amount
localparam init_pin_process     = 12'b1000_0_1000100; // initates the input of the user's pin
localparam wait_pin_process     = 12'b1001_0_1000000; // waits for user to input their pin
localparam init_payment_handle  = 12'b1010_0_1000010; // initates the credit card payment handling
localparam wait_payment_handle  = 12'b1011_0_1000000; // waits for credit card to be confirmed
localparam payment_success      = 12'b1100_0_1000000; // payment has been success fully processed
localparam payment_fail         = 12'b1101_0_1000001; // payment has failed and the process will be
    aborted.
```

```
// glitch free method of outputs being directly driven by state bits
assign light_bit                = state[6];   // A simple bit for a light, 0 when idle, 1 when process
    running
assign card_choice              = state[5:4]; // 2 bits for 3 possible card choices, 00 otherwise
assign pymt_amt_print           = state[3];   // bit to trigger external printing of payment amount
assign pin_process_init         = state[2];   // bit to trigger external processing of pin
    entry/confirmation
assign pymt_process_init        = state[1];   // bit to trigger external credit card payment handling
assign process_abort            = state[0];   // bit to trigger an abort to all external processes of
    the paymet handling.

// We do not want an asynchronous reset, because it may cause
// errors if payment is abruptly reset
always_ff @(posedge clk)
begin
    if (reset)                  begin
                                    state <= idle;
                                    fail_counter <= 2'b00;
                                end
    else
        case (state)
            idle:               if (process_init) state <= init_process;
                                else state <= idle;
            init_process:       state <= wait_credit_choice;
            wait_credit_choice: if (visa_choice_in) state <= choice_visa;
                                else if (mastercard_choice_in) state <= choice_mastercard;
                                else if (amex_choice_in) state <= choice_amex;
                                else state <= wait_credit_choice;
            choice_visa:        state <= init_amount_confirm;
            choice_mastercard:  state <= init_amount_confirm;
            choice_amex:        state <= init_amount_confirm;
            init_amount_confirm: state <= wait_amount_confirm;
            wait_amount_confirm: if (pymt_amt_conf) state <= init_pin_process;
                                else if (pymt_amt_denied) state <= payment_fail;
                                else state <= wait_amount_confirm;
            init_pin_process:   state <= wait_pin_process;
            wait_pin_process:   if (fail_counter == 2'b11) state <= payment_fail;
                                else if (pin_success) state <= init_payment_handle;
                                else if (pin_fail) begin
                                    state <= init_pin_process;
                                    fail_counter <= fail_counter + 1'b1;
                                    end
                                else state <= wait_pin_process;
            init_payment_handle: state <= wait_payment_handle;
            wait_payment_handle: if (transaction_success) state <= payment_success;
                                else if (transaction_fail) state <= payment_fail;
                                else state <= wait_payment_handle;
            payment_success:    begin
                                    state <= idle;
                                    fail_counter <= 2'b00;
                                end
            payment_fail:       begin
                                    state <= idle;
                                    fail_counter <= 2'b00;
                                end
            default:            state <= idle;
        endcase
end

endmodule
```

2 FSM Testbench Code

Below is the code for my FSM's testbench.

```
/* ELEC 402 Project 1 - System Verilog FSM Project
 * Name: Arthur Hsueh
 * ID: 21582168
 * Description: This testbench is for the FSM
*/

module credit_card_payment_fsm_tb ();
    logic clk_tb, reset_tb;
    logic process_init_tb,
        visa_choice_in_tb,
        mastercard_choice_in_tb,
        amex_choice_in_tb,
        pymt_amt_conf_tb,
        pymt_amt_denied_tb,
        pin_fail_tb,
        pin_success_tb,
        transaction_fail_tb,
        transaction_success_tb,
        light_bit_tb,
        pymt_amt_print_tb,
        pin_process_init_tb,
        pymt_process_init_tb,
        process_abort_tb;
    logic [1:0] card_choice_tb;

    // module dut instantiation
    credit_card_payment_fsm dut(
        .clk                 (clk_tb),
        .reset               (reset_tb),
        .process_init        (process_init_tb),
        .visa_choice_in      (visa_choice_in_tb),
        .mastercard_choice_in (mastercard_choice_in_tb),
        .amex_choice_in      (amex_choice_in_tb),
        .pymt_amt_conf       (pymt_amt_conf_tb),
        .pymt_amt_denied     (pymt_amt_denied_tb),
        .pin_fail            (pin_fail_tb),
        .pin_success         (pin_success_tb),
        .transaction_fail    (transaction_fail_tb),
        .transaction_success (transaction_success_tb),
        .light_bit           (light_bit_tb),
        .card_choice         (card_choice_tb),
        .pymt_amt_print      (pymt_amt_print_tb),
        .pin_process_init    (pin_process_init_tb),
        .pymt_process_init   (pymt_process_init_tb),
        .process_abort       (process_abort_tb)
    );

    // clock generation
    always begin
        clk_tb = 1; #5;
        clk_tb = 0; #5;
    end

    initial begin
        reset_tb = 1; #10;       // initiate first 'idle' state and check for state wait
        reset_tb = 0; #10;
```

```
/*
FIRST ITERATION OF FSM
*/
process_init_tb = 1; #10; // state <= init_process
process_init_tb = 0; #10; // state <= wait_credit_choice state
assert (light_bit_tb === 1'b1) else $error("light_bit assert fail");
mastercard_choice_in_tb = 1; #10; // state <= choice_mastercard
assert (card_choice_tb === 2'b10) else $error ("Mastercard_choice value failed");// check output
#10; // state <= init_amount_confirm
assert (pymt_amt_print_tb === 1'b1) else $error ("payment_amt_print value fail should be 1'b1");
#10; // state <= wait_amount confirm
pymt_amt_conf_tb = 1; #10; // state <= init_pin_process
assert (pin_process_init_tb === 1'b1) else $error ("pin_process_init should be 1'b1");
pymt_amt_conf_tb = 0;
#10; // state <= wait_pin_process
pin_fail_tb = 1'b1;
#10; // state <= wait_pin_process
assert (dut.fail_counter === 2'b01) else $error("fail_counter should be 2'b01");
pin_fail_tb = 1'b0;
#10;
pin_fail_tb = 1'b1;
#10;
assert (dut.fail_counter === 2'b10) else $error("fail_counter should be 2'b10");
pin_fail_tb = 1'b0;
#10; // state <= wait_pin_process
pin_success_tb = 1'b1;
#10; // state <= init_payment_handle
assert (pymt_process_init_tb === 1'b1) else $error ("pymt_process_init should be 1'b1");
pin_success_tb = 1'b0;
#10; // state <= wait_payment_handle
transaction_success_tb = 1'b1;
#10; // state <= payment_success
transaction_success_tb = 1'b0;
#10; // state <= idle;
assert (dut.fail_counter === 2'b00) else $error("fail_counter should be 2'b00");

/*
SECOND ITERATION OF FSM
*/
process_init_tb = 1; #10; // state <= init_process
process_init_tb = 0; #10; // state <= wait_credit_choice state
assert (light_bit_tb === 1'b1) else $error("light_bit assert fail");
mastercard_choice_in_tb = 1; #10; // state <= choice_mastercard
assert (card_choice_tb === 2'b10) else $error ("Mastercard_choice value failed");// check output
#10; // state <= init_amount_confirm
assert (pymt_amt_print_tb === 1'b1) else $error ("payment_amt_print value fail should be 1'b1");
#10; // state <= wait_amount confirm
pymt_amt_denied_tb = 1; #10; // state <= payment_fail
assert (pin_process_init_tb === 1'b0) else $error ("pin_process_init should be 1'b0");
#10; // state <= idle
assert (dut.state === 12'b000000000000) else $error ("state should be idle, beit encoded 12'b0");

/*
THIRD ITERATION OF FSM
*/
process_init_tb = 1; #10; // state <= init_process
process_init_tb = 0; #10; // state <= wait_credit_choice state
assert (light_bit_tb === 1'b1) else $error("light_bit assert fail");
mastercard_choice_in_tb = 1; #10; // state <= choice_mastercard
assert (card_choice_tb === 2'b10) else $error ("Mastercard_choice value failed");// check output
#10; // state <= init_amount_confirm
```

```
        assert (pymt_amt_print_tb === 1'b1) else $error ("payment_amt_print value fail should be 1'b1");
        #10; // state <= wait_amount confirm
        pymt_amt_conf_tb = 1; #10; // state <= init_pin_process
        assert (pin_process_init_tb === 1'b1) else $error ("pin_process_init should be 1'b1");
        pymt_amt_conf_tb = 0;
        #10; // state <= wait_pin_process
        pin_success_tb = 1'b1;
        #10; // state <= init_payment_handle
        assert (pymt_process_init_tb === 1'b1) else $error ("pymt_process_init should be 1'b1");
        pin_success_tb = 1'b0;
        #10 // state <= wait_payment_handle;
        transaction_fail_tb = 1'b1;
        #10; // state <= payment_fail
        assert (process_abort_tb === 1'b1) else $error("process_abort should be 1'b1");
        #10; // state <= idle;
        assert (dut.state === 12'b000000000000) else $error ("state should be idle, beit encoded 12'b0");
        $stop;

    end

endmodule
```