University of British Columbia

Coin Casher for Arcade machine

Assignment 1, ELEC 402 101, Tutorial section T1A,
Instructor: MOLAVI, REZA

Yuqian (Alan) Hu, 64133713
9-30-2021

Table of content

Introduction

Recently one of my friends brought me to a local game arcade place. The game machines inside the arcade are fun to play with, and relatively simple compared to the modern game consoles. Inspecting the arcade machines, I started noticing their coin casher systems are nearly identical, so I thought it would be interesting to replicate such system in a FSM which can be used as a building block to optimize the arcade machine to suit the modern need.

FSM General Description

Module “Coin_casher” is the main FSM for coin casher. This FSM constantly checks if the user inserts a coin, and keeps track of how many coin has been inserted.

If user insert a wrong coin type (1 dollar coin as default), the FSM will return the coin. Once the user inserts one correct coin into the casher, the FSM will start a timer. The casher will return the inserted coins if the user didn’t insert enough coins before the timer ends. When appropriated number of coins are inserted (3 coins as default), The FSM will tell the arcade machine to start the game. During the game, the FSM is halted. After the game finish the FSM will reset and wait for coin to be inserted. If the “return all coins” button is pressed the FSM will return all the inserted coins and reset the timer.

Additional Module Description

Module “**insertcoin**” set a flag to high for one clock cycle if it detects any coin inserted into the casher, and it also output the type of the inserted coin. This module also serve as the physical interface to user, where it handle all physical action related to coin (example: spit all coins, reject coin, eat coins). (Abstraction, not actually written)

Module “**timer**” is default halting until it receives a enable flag from the FSM, in which case it starts count down and set a flag to high for one clock cycle once the count down finish. It also receives a reset flag, which tells this module to reset the count down and go back to halting when the flag is set to high. (Abstraction, not actually written)

Module “**game**” is the game program in the Arcade machines. No game is programed in this project. This module starts the game when prompt by the FSM, and tell FSM when the game is finished. (Abstraction, not actually written)

Test Bench

The test bench verifies the state transitions/outputs in the following scenarios:

Note: Details about expected state transitions/outputs is listed and explained in the comments of the test bench, with a time stamp ("@20ns" in the first scenario) at the end of each test section.

1. Nothing is inserted and no button is pressed. The system should keep waiting.

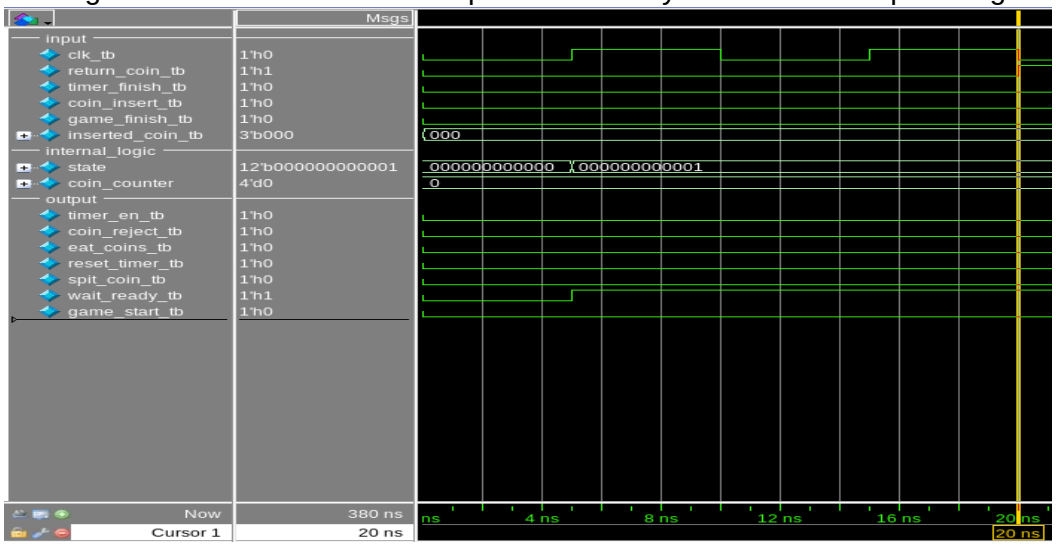

Figure 1.1 test bench 1st scenario

In this case, the flag "wait_ready_tb" stays on high indicating the FSM is waiting for coin to be inserted.

2. Press return all coin. The system should spit all coins, and reset timer.

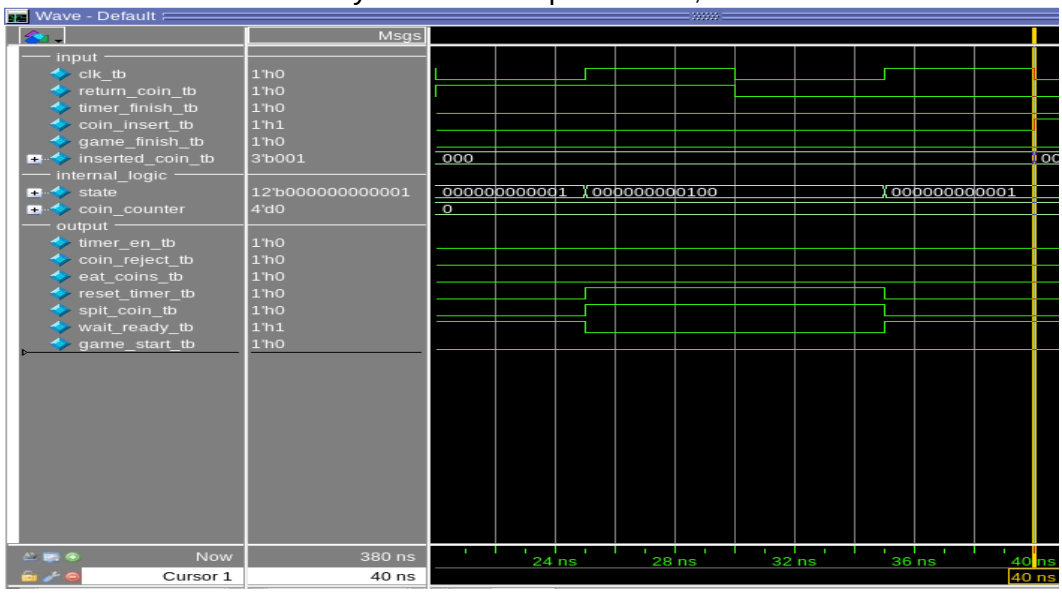

Figure 1.2 test bench 2nd scenario

In this case, the flag "reset_timer_tb" and "spit_coin_tb" is set to high, telling the timer module to restart and insertcoin module to return all coins.

3. One wrong coin is inserted. The system should reject the coin, and back to waiting.

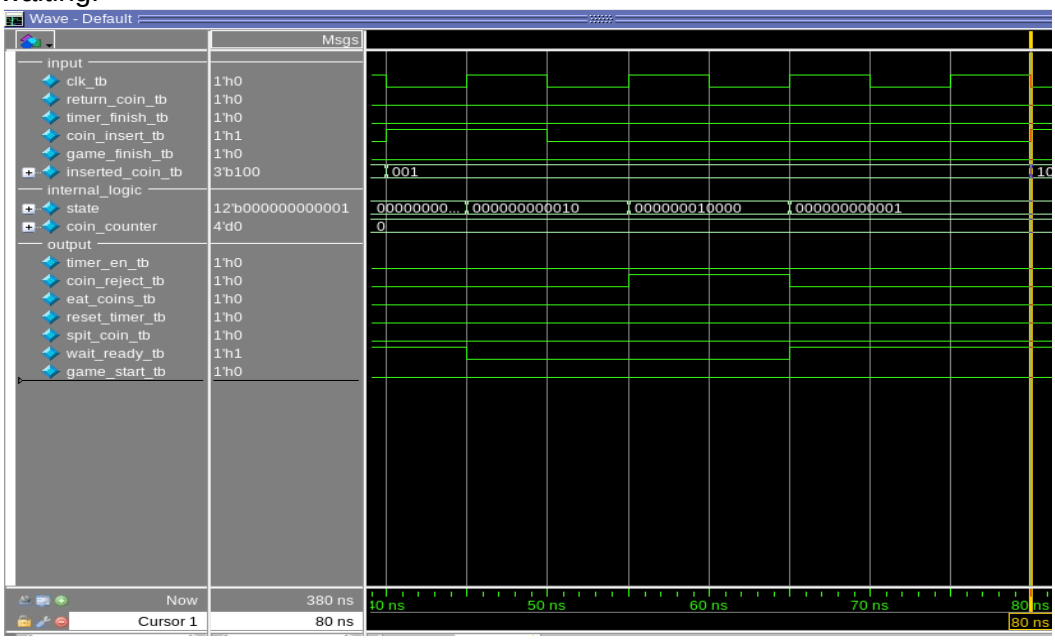

Figure 1.3 test bench 3rd scenario

In this case, “coin_reject” is set to high telling insertcoin module to reject the coin. Notice that the “coin_counter” is unchanged.

4. One appropriate coin is inserted. The system should start the timer, increment coin count, then back to waiting

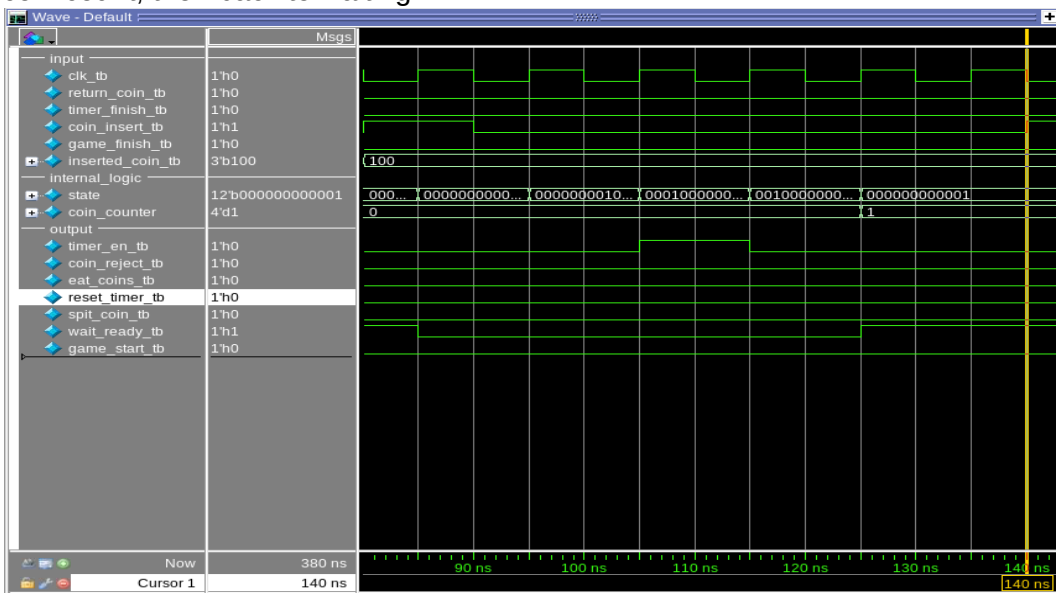

Figure 1.4 test bench 4th scenario

Since it’s the first correct, the FSM will start a timer (see “timer_en_tb” is set to high). After that, the “coin_counter” increments.

5. Another appropriate coin is inserted. The system should increment coin count, then back to waiting.

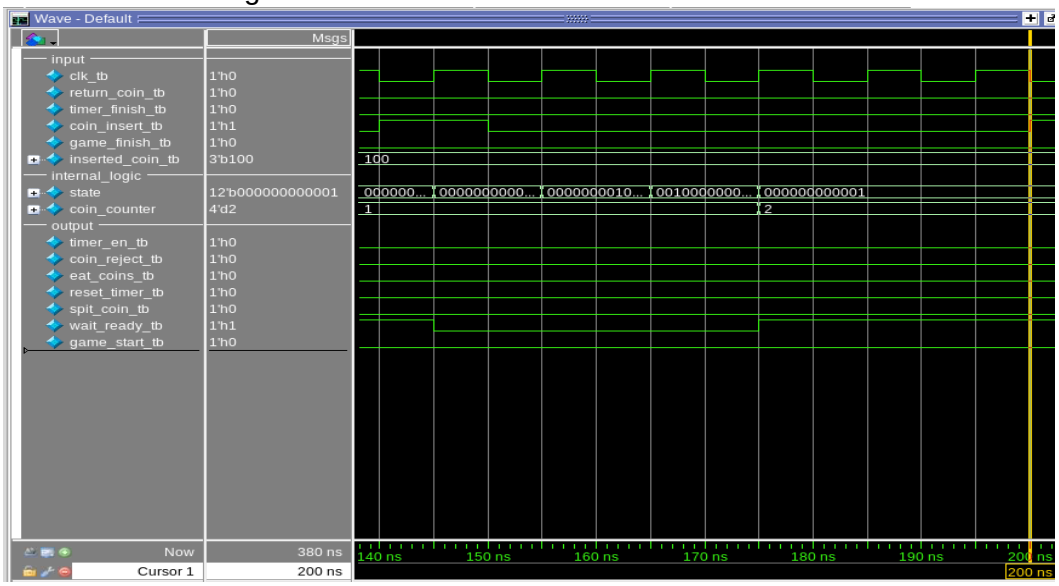

Figure 1.5 test bench 5th scenario

Since it isn't the first coin inserted, the system WOULD NOT enable the timer again (timer_en is always 0), but the "coin_couter" gets incremented.

6. Another appropriate coin is inserted. The system should start the game, clear the coin count, reset the timer, reject any coin during the game, and wait for the game to finish.

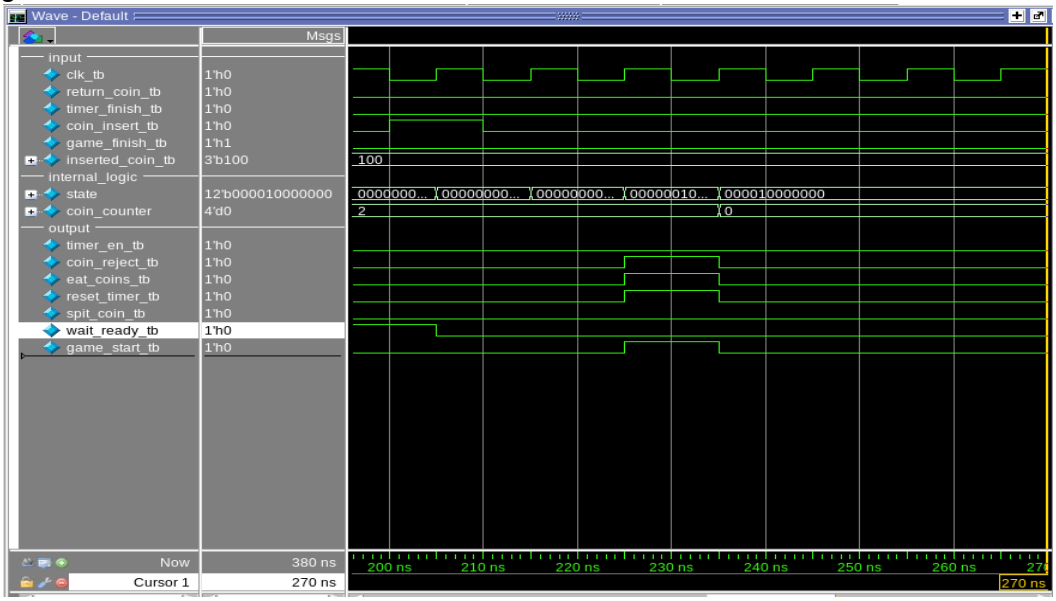

Figure 1.6 test bench 6th scenario

As shown in the figure, flags "coin_reject", "eat_coins", "reset_timer" and "start_game" is set to high, telling the casher to store all the inserted coins, reject any coin during the game and reset timer. Additionally, the "coin_counter" is set to 0.

7. Game is finished. The system should return to waiting (for coins).

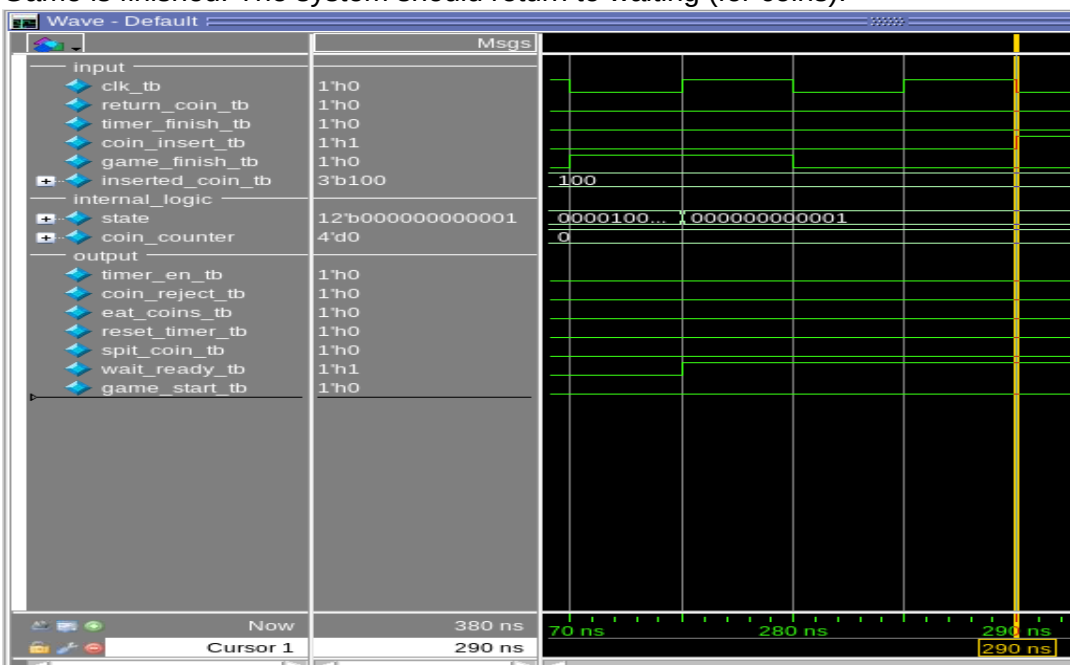

Figure 1.7 test bench 7th scenario

After the game finished (“game_finish” is high), the system goes to wait for coin insert.

8. Insert one coin, and wait for timeout. The system should System should start timer, increment coin counter. After timeout, the system should spit all coins, and reset timer.

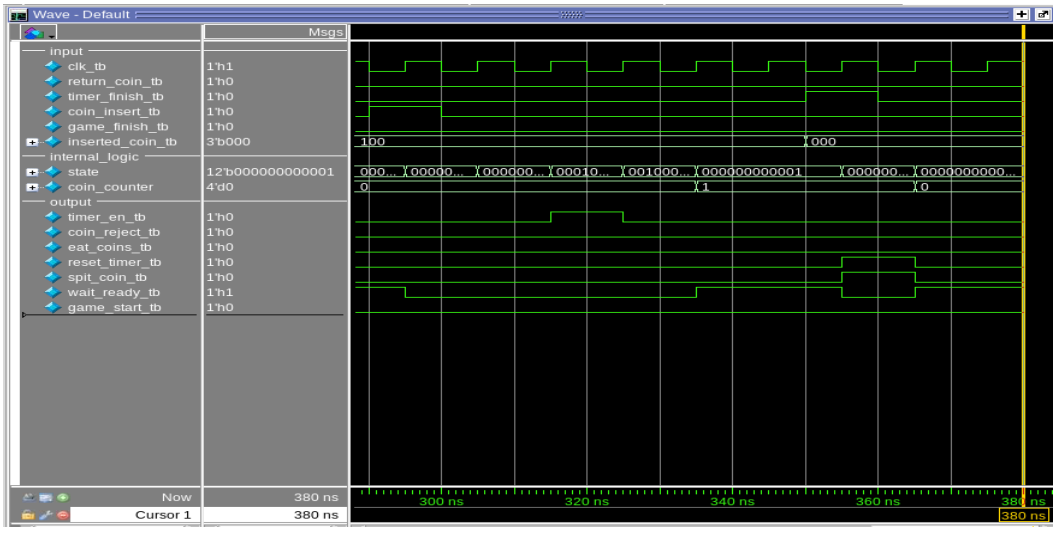

Figure 1.8 test bench 8th scenario

As timer finish count down (“timer_finish” set to high) the casher resets timer (“reset_timer” goes to high) and returns all the inserted coins (“spit_coin” set to high). After that, the casher returns to wait for coin insert.

Block Diagrams

SFM block diagram

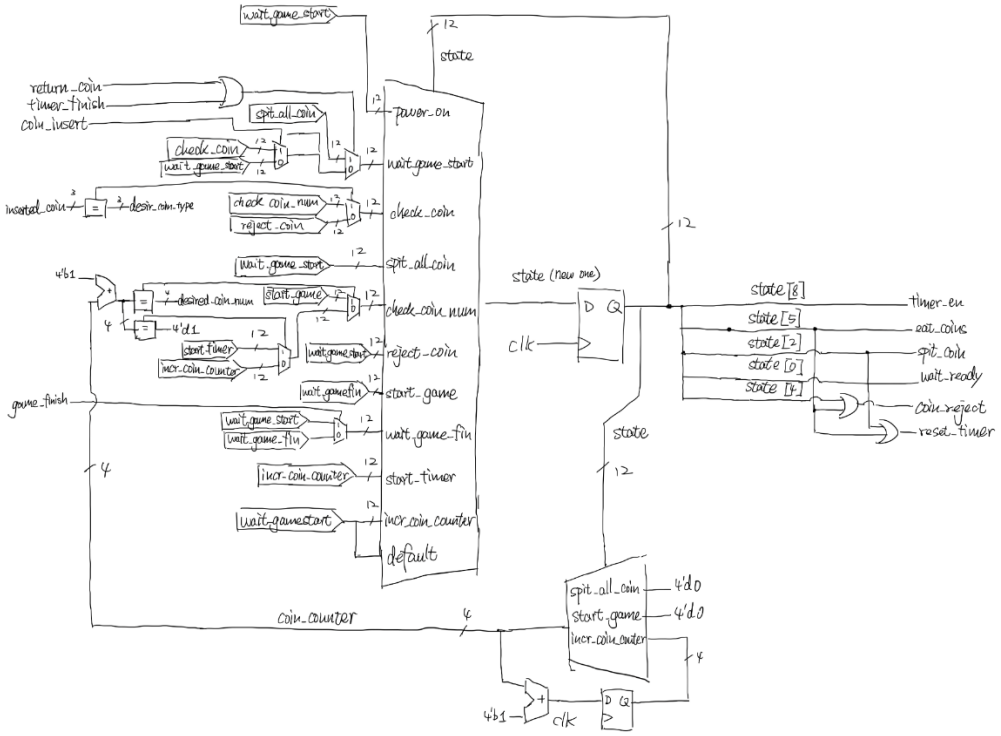

Figure 2.1 module "Coin_casher" block diagram

FSM + Addition Modules Block Diagram

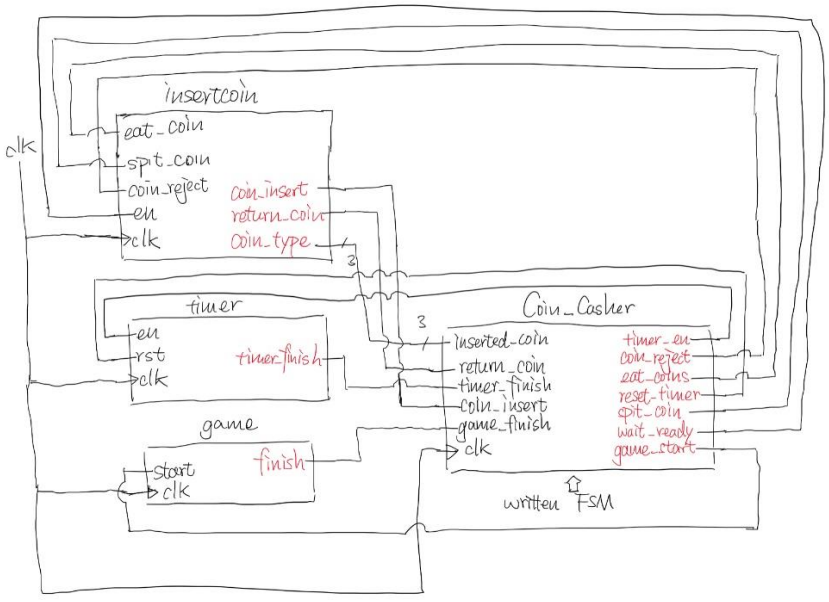

Figure 2.2 block diagram for all modules

Test Bench Connection Block diagram

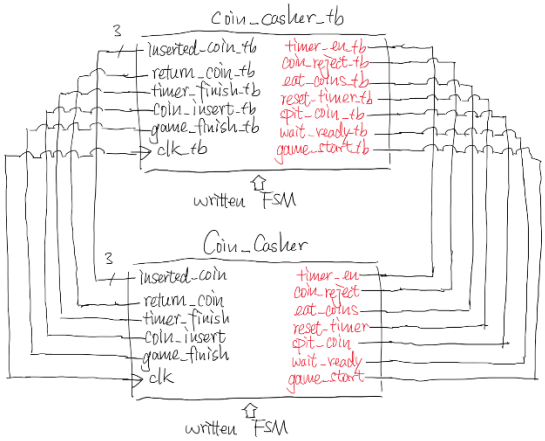

Figure 2.3 test bench block diagram

Appendix

Code:

"Coin_casher.sv":

```
module CoinCahser(
    input   clk, return_coin, timer_finish, coin_insert, game_finish, // "coin_insert" is
high when coin is inserted into the arcade machine
    input   [2:0] inserted_coin,          // "inserted_coin" tells the Denomination of the
 coin, 0001 for 5 cents, 010 for 10 cents, 011 for 25 cents, 100 for 1 dollar, 101 for 2 d
ollar
    output  timer_en, coin_reject, eat_coins, reset_timer, spit_coin, wait_ready, game_sta
rt    // output flags
);
    logic [11:0] state = 12'b0;
    parameter power_on          = 12'b0;            // the start-up/fefault state
    parameter wait_game_start   = 12'b1;            // where FSM wait for coin to be inser
ted, theoretically the most common state the FSM stay at
    parameter check_coin        = 12'b10;           // check if the inseted coin is the de
sired
    parameter spit_all_coin     = 12'b100;          // return all holding coins to the pla
yer
    parameter check_coin_num    = 12'b1000;         // check the number of inserted coin
    parameter reject_coin       = 12'b10000;        // output flag to reject the inserted
coin
    parameter start_game        = 12'b100000;       // tells the game module (dont care in
 this project) to start
    //parameter check_fir_coin    = 12'b1000000;
    parameter wait_game_fin     = 12'b10000000;     // wait for the game module to finish
```

```
    parameter start_timer       = 12'b100000000;    // tell timer to start when palyer ins
erted the first coin
    parameter incr_coin_count   = 12'b1000000000;   // the state that increament the count
 of the inserted coins

    logic [3:0] coin_counter = 4'd0;    // typical arcade machine accepts no more than 8 c
oins, default setting: accept 2 coins to start

    parameter desired_coin_type = 3'b100;
    parameter desired_coin_num  = 4'd3;

    // state transition
    always_ff @(posedge clk) begin
        case(state)
            power_on:       state <= wait_game_start;
            wait_game_start: begin
                if(return_coin || timer_finish)
                    state <= spit_all_coin;
                else if(coin_insert)
                    state <= check_coin;
                else
                    state <= wait_game_start;
            end
            check_coin: begin
                if(inserted_coin == desired_coin_type) // default setting: the machine onl
y accept 1 dollar coin
                    state <= check_coin_num;
                else
                    state <= reject_coin;
            end
            spit_all_coin: begin
                state <= wait_game_start;
                coin_counter <= 4'd0;   // reset coin counter
            end
            check_coin_num: begin
                if((coin_counter + 1'b1) == desired_coin_num)   // since non-
blocking assignment, here need to compare "coin_counter + 1"
                    state <= start_game;
                else if ((coin_counter + 1'b1) == 4'd1)
                    state <= start_timer;
                else
                    state <= incr_coin_count;
            end
            reject_coin:    state <= wait_game_start;
            start_game: begin
```

```
                state <= wait_game_fin;
                coin_counter <= 4'd0;   // reset coin counter
            end
            //check_fir_coin:
            wait_game_fin:  state <= game_finish ? wait_game_start : wait_game_fin;
            start_timer:    state <= incr_coin_count;
            incr_coin_count: begin
                state <= wait_game_start;
                coin_counter <= coin_counter + 4'b1;
            end
            default: state <= wait_game_start;
        endcase
    end

    // FSM outputs
    assign timer_en     = state[8];
    assign coin_reject  = state[4] || state[5];
    assign eat_coins    = state[5];
    assign reset_timer  = state[5] || state[2];
    assign spit_coin    = state[2];
    assign wait_ready   = state[0]; // tell the insert coin module to start detect coins
    assign game_start   = state[5];

endmodule

// module that detect coin and tell the main FSM coin type + flag
// all flags is sync with the main FSM
// module insertcoin (
//     input clk, en, coin_reject, eat_coins,
//     output coin_insert, return_coin,
//     output [2:0] coin_type
// );
//      ...
// endmodule

// this module helps the FSM to count down, who wait for the customer to insert the rest o
f the coin
// the parameter makes this FSM count 60s in default (depends on clk frquency)
// all falgs is sync with the main FSM
// module timer #(
//     parameter count = ...
// ) (
//     input clk, en, rst,
//     output timer_fin
// );
```

```
// endmodule

// the game module that start the game when prompt by the coin casher FSM
// also tell the FSM whenn the game is finished
// all flags is sync with the main FSM
// module game (
//     input clk, start,
//     output finish
// );

// endmodule
```

“coin_casher_tb.sv”:

```
module coin_casher_tb;
    logic    clk_tb, return_coin_tb, timer_finish_tb, coin_insert_tb, game_finish_tb;
    logic    [2:0] inserted_coin_tb;
    logic    timer_en_tb, coin_reject_tb, eat_coins_tb, reset_timer_tb, spit_coin_tb, wait
_ready_tb, game_start_tb;

    CoinCahser DUT1(
        clk_tb, return_coin_tb, timer_finish_tb, coin_insert_tb, game_finish_tb,
        inserted_coin_tb,
        timer_en_tb, coin_reject_tb, eat_coins_tb, reset_timer_tb, spit_coin_tb, wait_read
y_tb, game_start_tb
    );

    initial forever begin
        clk_tb = 1'b0;  #5;
        clk_tb = 1'b1;  #5;
    end

    initial begin
        // testing FSM with no input on high,
        // the state should transition to "wait_game_start" and stay in it (12'b1)
        // output flag "wait_ready" should be 1
        return_coin_tb  = 1'b0;
        timer_finish_tb = 1'b0;
        coin_insert_tb  = 1'b0;
        game_finish_tb  = 1'b0;
        inserted_coin_tb   = 3'b0;
        #20;    // @20ns

        // testing return coin function,
```

```
        // expected state trans:
        // wait_game_start(12'b1) --> spit_all_coin (12'b100) --> wait_game_start(12'b1),
        // output flags: "spit_coins", "reset_timer" becomes high @ "spit_all_coin(12'b100
)"
        return_coin_tb  = 1'b1;
        timer_finish_tb = 1'b0;
        coin_insert_tb  = 1'b0;
        game_finish_tb  = 1'b0;
        inserted_coin_tb   = 3'b0;
        #10;
        return_coin_tb  = 1'b0;
        #10;    // @40ns

        // testing insert coin function,
        // first, insert a wrong coin type
        // expected state trans:
        // wait_game_start(12'b1) --> check_coin(12'b10) --> reject_coin(12'b10000) -
-> wait_game_start(12'b1)
        // output flag: "coin_reject" becomes high @ "reject_coin(12'b10000)"
        return_coin_tb  = 1'b0;
        timer_finish_tb = 1'b0;
        coin_insert_tb  = 1'b1;
        game_finish_tb  = 1'b0;
        inserted_coin_tb   = 3'b1;  // 5 cents
        #10;
        coin_insert_tb = 1'b0;
        #30;    // @80ns
        // then, insert a correct coin
        // expected state trans:
        // wait_game_start(12'b1) --> check_coin(12'b10) --> check_coin_num(12'b1000) -->
        // start_timer(12'b100000000) --> incr_coin_count(12'b1000000000) -
-> wait_game_start(12'b1)
        // output flag: "timer_enable" becomes high @ "start_timer(12'b100000000)"
        // coin_counter should increament after "incr_coin_count(12'b1000000000)"
        return_coin_tb  = 1'b0;
        timer_finish_tb = 1'b0;
        coin_insert_tb  = 1'b1;
        game_finish_tb  = 1'b0;
        inserted_coin_tb   = 3'b100;    // 1 dollar
        #10;
        coin_insert_tb = 1'b0;
        #50;    // @140ns
        // then, insert another coin (we need three in total)
        // this time we will skip the "timer_start" statr
        // expected state trans:
```

```
        // wait_game_start(12'b1) --> check_coin(12'b10) --> check_coin_num(12'b1000) -->
        // incr_coin_count(12'b1000000000) --> wait_game_start(12'b1)
        // output flag: NO CHANGE
        // coin_counter should increament after "incr_coin_count(12'b1000000000)"
        return_coin_tb  = 1'b0;
        timer_finish_tb = 1'b0;
        coin_insert_tb  = 1'b1;
        game_finish_tb  = 1'b0;
        inserted_coin_tb   = 3'b100;    // 1 dollar
        #10;
        coin_insert_tb = 1'b0;
        #50;    // @200ns
        // then, insert another coin,
        // this time the game shall start
        // expected state trans:
        // wait_game_start(12'b1) --> check_coin(12'b10) --> check_coin_num(12'b1000) -->
        // start_game(12'b100000) --> wait_game_fin(12'b10000000) <--> loop back
        // output flag: "game_start", "eat_coins", "reset_timer", and "coin_reject" should
 be high @ "start_game(12'b100000)"
        // coin counter should be 0 @ "start_game(12'b100000)"
        // the state should loop in wait_game_fin(12'b10000000) until "game_finish" is hig
h
        return_coin_tb  = 1'b0;
        timer_finish_tb = 1'b0;
        coin_insert_tb  = 1'b1;
        game_finish_tb  = 1'b0;
        inserted_coin_tb   = 3'b100;    // 1 dollar
        #10;
        coin_insert_tb = 1'b0;
        #60;    // @270ns
        // tell the FSM play has finish the game, reset
        game_finish_tb  = 1'b1;
        #10;
        game_finish_tb  = 1'b0;
        #10;    // @290ns

        // testing time out function,
        // first insert a coin,
        // then wait till time out (external timeout flag)
        return_coin_tb  = 1'b0;
        timer_finish_tb = 1'b0;
        coin_insert_tb  = 1'b1;
        game_finish_tb  = 1'b0;
        inserted_coin_tb   = 3'b100;    // 1 dollar
        #10;
```

```
        coin_insert_tb = 1'b0;
        #50;    // @350ns
        // expected state trans:
        // wait_game_start(12'b1) --> spit_all_coin (12'b100) --> wait_game_start(12'b1),
        // output flags: "spit_coins", "reset_timer" becomes high @ "spit_all_coin(12'b100
)"
        // coin count reset to 0
        return_coin_tb  = 1'b0;
        timer_finish_tb = 1'b1;
        coin_insert_tb  = 1'b0;
        game_finish_tb  = 1'b0;
        inserted_coin_tb   = 3'b000;
        #10;
        timer_finish_tb = 1'b0;
        #20;    // @380ns
        $stop;
    end

endmodule
```