|  |
| --- |
| 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**

**FSM General Description**

**Additional Module Description**

**Test Bench**

*Scenario 1*

*Scenario 2*

*Scenario 3*

*Scenario 4*

*Scenario 5*

*Scenario 6*

*Scenario 7*

*Scenario 8*

**Block Diagrams**

*SFM Block Diagram*

*FSM + Addition Modules BD*

*Test Bench Connection BD*

**Appendix**

“Coin\_casher.sv”

“coin\_casher\_tb.sv”

*Page 1*

*Page 1*

*Page 1*

*Page 2*

*Page 2*

*Page 2*

*Page 3*

*Page 3*

*Page 4*

*Page 4*

*Page 5*

*Page 5*

*Page 6*

*Page 6*

*Page 6*

*Page 6*

*Page 7*

*Page 7*

Page 10

**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.

Chart

Description automatically generated

*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.

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

Graphical user interface, chart

Description automatically generated

*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.

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

Graphical user interface

Description automatically generated with medium confidence

*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.

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

A picture containing graphical user interface

Description automatically generated

*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.

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

A picture containing graphical user interface

Description automatically generated

*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.

1. 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.

Graphical user interface

Description automatically generated

*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.

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

Chart

Description automatically generated

*Figure 1.7 test bench 7th scenario*

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

1. 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.

Graphical user interface

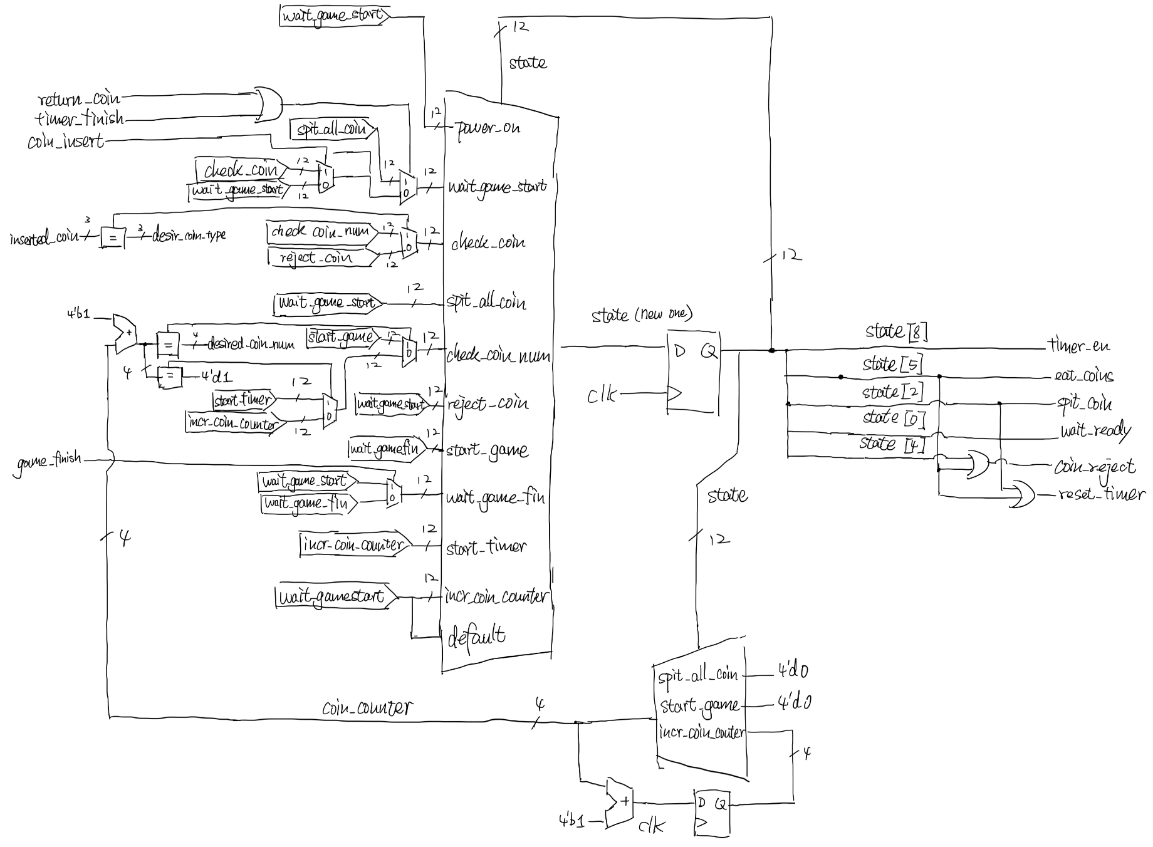
Description automatically generated

*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.

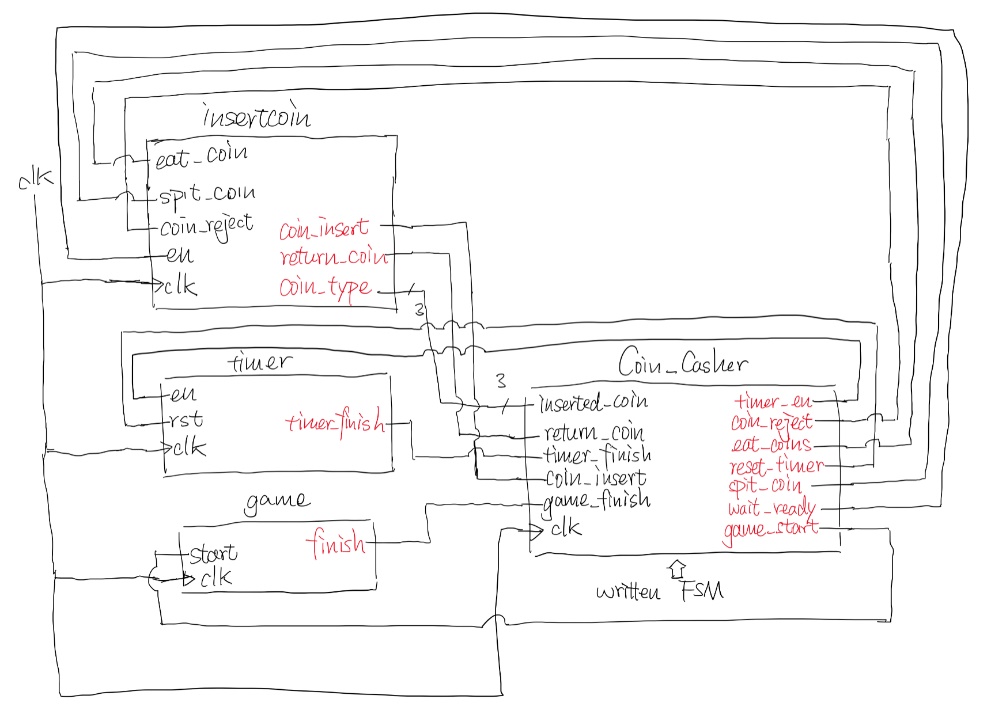
**Block Diagrams**

**SFM block diagram**



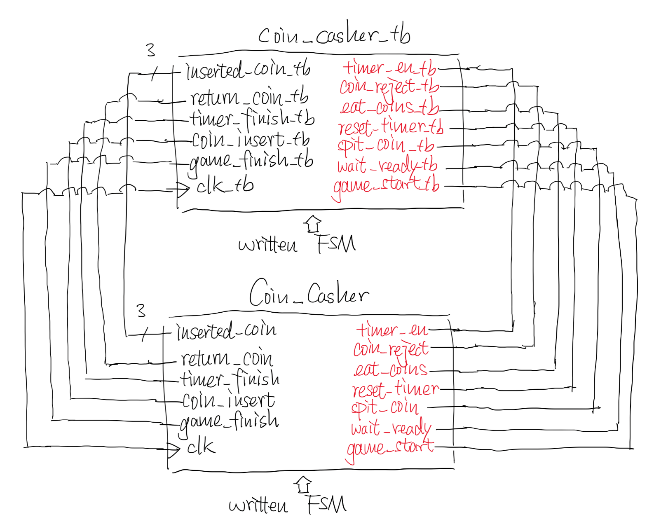
*Figure 2.1 module “Coin\_casher” block diagram*

**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 dollar

    output  timer\_en, coin\_reject, eat\_coins, reset\_timer, spit\_coin, wait\_ready, game\_start    // 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 inserted, theoretically the most common state the FSM stay at

    parameter check\_coin        = 12'b10;           // check if the inseted coin is the desired

    parameter spit\_all\_coin     = 12'b100;          // return all holding coins to the player

    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 inserted 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 coins, 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 only 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 of 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\_ready\_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 high

        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