University of British Columbia

Synthesized Coin Casher for Arcade machine

Assignment 2, ELEC 402 101, Tutorial section T1A,
Instructor: MOLAVI, REZA

Yuqian (Alan) Hu, 64133713
10-9-2021

Table of content

Introduction

The Coin casher FSM from the last assignment is synthesized in Cadence RTL compiler. Some minor changes are applied to the FSM to produce a synthesizable result (without compromising on the functionalities, more details in the later sections). Timer module is completed and added to the FSM.

Changes in coin_casher FSM

- Module “Coin_Casher” is renamed to “coin_casher” since design with upper case letter can trigger error in the RTL compiler.
- State encoding is changed to binary encoding (encoded in hot-code originally), since hot-code encoded state won’t trigger proper state transitions.
- More initializing assignments is added to the “power_on” state, since the state/output would likely be “x” or “z” if state/output haven’t been initialized properly.
- Instead of assigning output to a state bit (hot-code), output is assigned in a combinational logic always block.

Note: More details attached in the appendix [1], the mapped Verilog is also attached in the appendix [2].

Added Module Description

Module “**timer**” is a simple FSM with three states in total used to enable a 30 seconds count down. It 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.

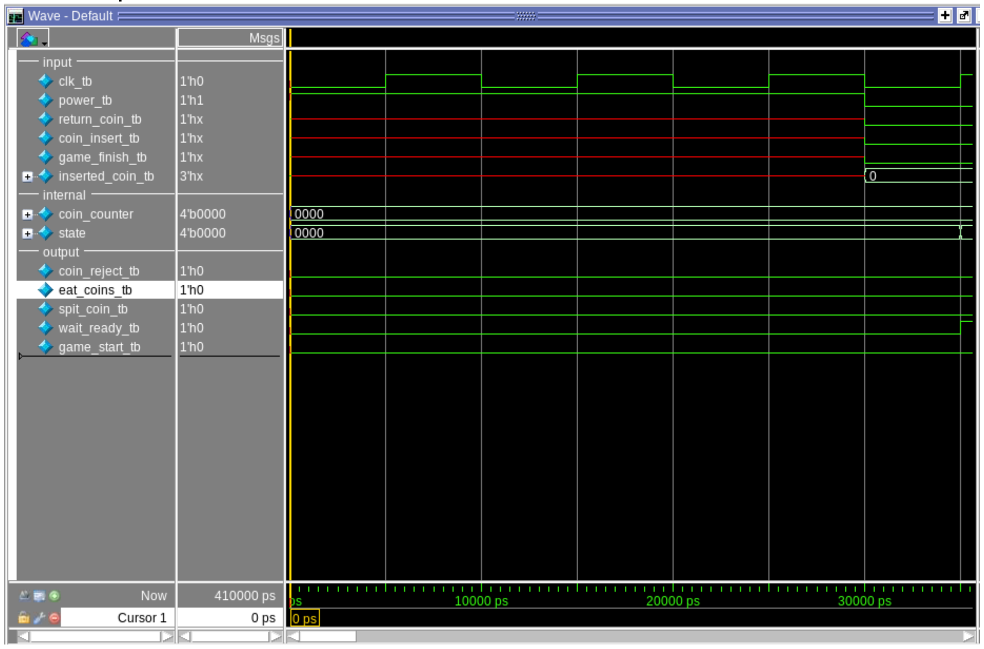
Waveforms

Waveforms from the **new FSM**:

Note 1: 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.

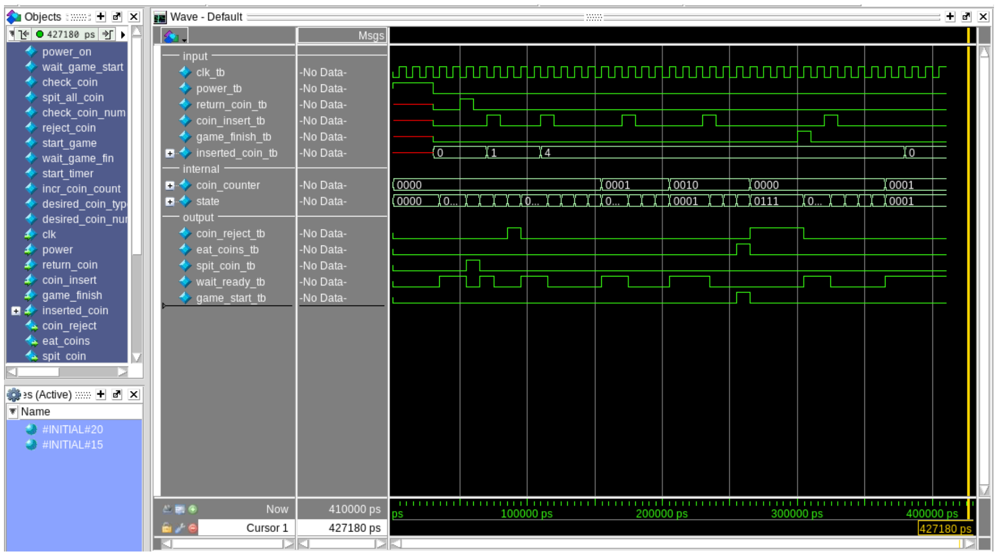
Note 2: Behavior of the new FSM is exactly identical to the old one except the modified "power_on" state, which it initialize state, "coin_counter", and outputs.

1. "Power" signal is set to high, so FSM should be in "power_on" state; state/outputs should be initialized to "4'b0"/"1'b0".



```
Wave - Default
                      Msgs
input
  clk_tb              1'h0
  power_tb            1'h1
  return_coin_tb      1'hx
  coin_insert_tb      1'hx
  game_finish_tb      1'hx
  inserted_coin_tb    3'hx
internal
  coin_counter        4'b0000
  state               4'b0000
output
  coin_reject_tb      1'h0
  eat_coins_tb        1'h0
  spit_coin_tb        1'h0
  wait_ready_tb       1'h0
  game_start_tb       1'h0

Now        410000 ps
Cursor 1   0 ps
```

2. Overview waveform of the new FSM:

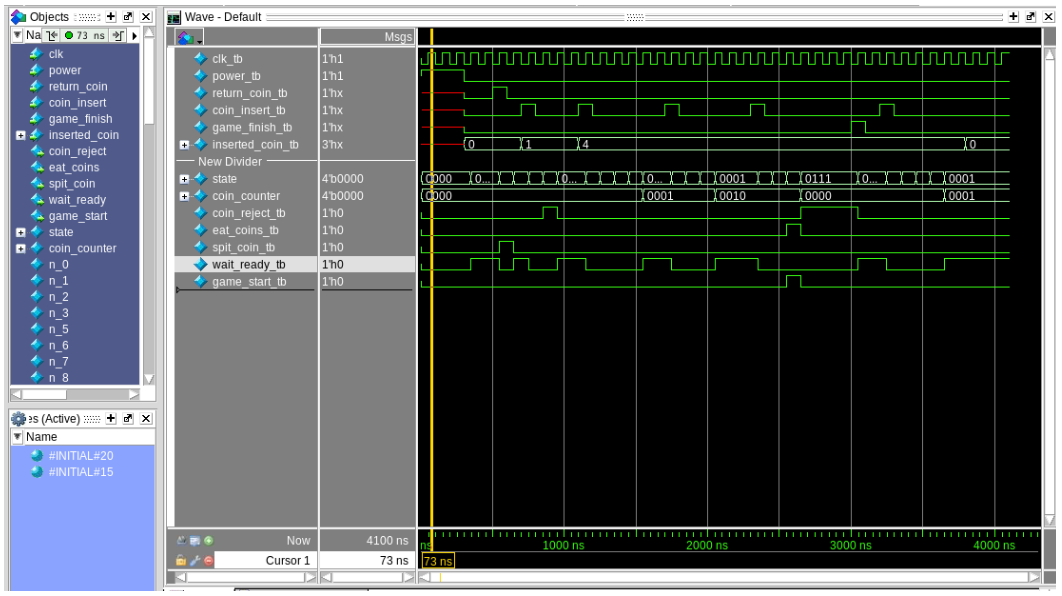


```
Objects
power_on
wait_game_start
check_coin
spit_all_coin
check_coin_num
reject_coin
start_game
wait_game_fin
start_timer
incr_coin_count
desired_coin_typ
desired_coin_nu
clk
power
return_coin
coin_insert
game_finish
inserted_coin
coin_reject
eat_coins
spit_coin

Processes (Active)
Name
#INITIAL#20
#INITIAL#15

Wave - Default
                      Msgs
input
  clk_tb              -No Data-
  power_tb            -No Data-
  return_coin_tb      -No Data-
  coin_insert_tb      -No Data-
  game_finish_tb      -No Data-
  inserted_coin_tb    -No Data-
internal
  coin_counter        -No Data-
  state               -No Data-
output
  coin_reject_tb      -No Data-
  eat_coins_tb        -No Data-
  spit_coin_tb        -No Data-
  wait_ready_tb       -No Data-
  game_start_tb       -No Data-

Now        410000 ps
Cursor 1   427180 ps
```

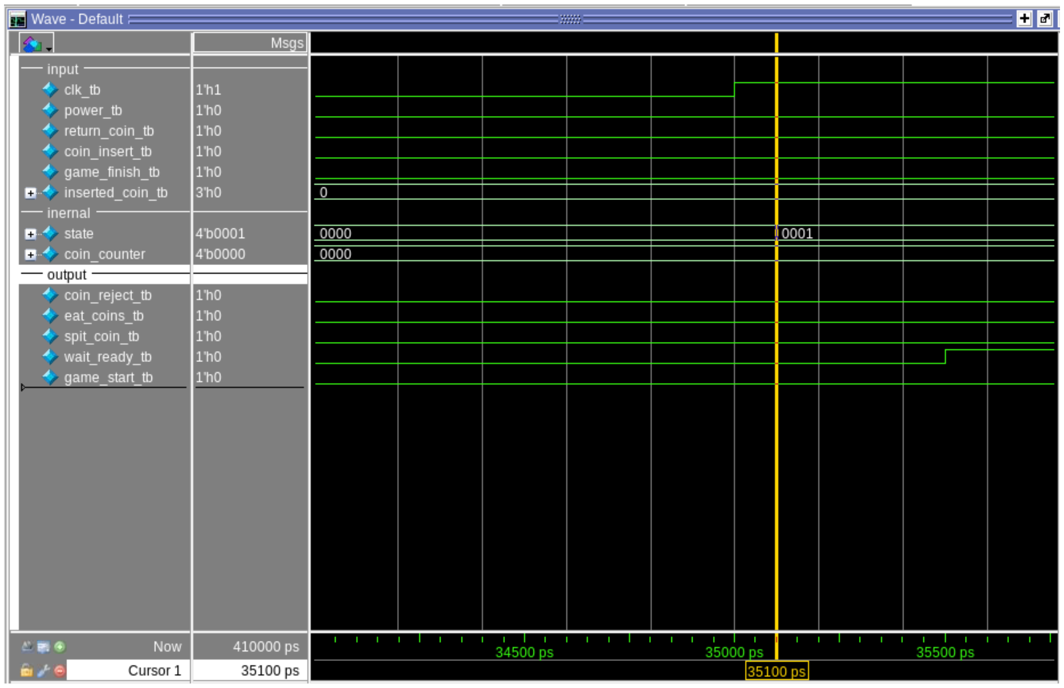
Waveforms from the **mapped Verilog**:

An overview of waveform from the **mapped Verilog** (100ns/10ns per period), which is exactly the same as the above overview waveform from the new FSM**:**

Note: As shown inside the "objects" window, "n_1", "n_2", etc. are the unique signals in the mapped Verilog, thus proving this waveform is generated from the mapped Verilog.



Simulated delays (about 100ps from clock posedge to state transition):



Mapped Verilog

The generated design contains <u>273 cells</u> in total with a <u>time slack of 10ps</u> (more detail shown in the appendix [3]). There are three separate modules in the generated design, namely the “increment_unsigned_16”, “timer”, and “coin_casher”. The module “increment_unsigned_16” is used to keep track of the incrementing counter inside the timer module.

There is also a section called “wait_for_user” containing 195 cells in the area report which supposedly handles the state transitions.

Summary

The design uses 273 cells and occupies 113-unit-area of space.

According to the report “coin_casher_gate.rpt” (Appendix [3]), the sequential and the combinational logic roughly occupies the equal amount of area on this design (51.8/51.2). The most area-consuming component is the “DFFRNQ_X1” (D flip-flop), which is instantiated 37 times and makes up the majority of the sequential logic blocks (47.284/51.806). The most used gate is “INV_X1” (invertor), which is instantiated 65 times and makes up 8.5% of the total area.

The power consumption of this design is 0.73 W with over 99.99% of the power allocates to dynamic power, and the Leakage power is only about 75nW (Appendix [3]).

The timing analysis shows that the system has a timing slack of 10ps @100ns clock period. The D flip-flop in sequential logic contributes the highest delay around 18ps, and each of the NAND/NOR gates in the combinational logic contribute additional delay of 10ps (Appendix [3]).

Appendix

[1] Modified FSM & testbench:

“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
```

[2] Mapped Verilog:

“coin_casher_map.v”

```
// Generated by Cadence Encounter(R) RTL Compiler RC14.13 - v14.10-
s027_1

// Verification Directory fv/coin_casher

module increment_unsigned_16(A, CI, Z);
  input [28:0] A;
  input CI;
  output [28:0] Z;
  wire [28:0] A;
  wire CI;
  wire [28:0] Z;
  wire n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8;
  wire n_9, n_10, n_11, n_12, n_13, n_14, n_15, n_16;
  wire n_17, n_18, n_19, n_20, n_22, n_23, n_24, n_25;
  wire n_26, n_27, n_28, n_29, n_30, n_31, n_32, n_33;
  wire n_34, n_36, n_37, n_38, n_39, n_40, n_41, n_42;
  wire n_43, n_45, n_48, n_49, n_50, n_51, n_52, n_53;
  wire n_54, n_55, n_56, n_57, n_58, n_60, n_73;
  assign Z[0] = 1'b0;
  XOR2_X1 g1850(.A1 (n_73), .A2 (A[17]), .Z (Z[17]));
  XOR2_X1 g1851(.A1 (n_60), .A2 (A[9]), .Z (Z[9]));
  XNOR2_X1 g1852(.A1 (n_55), .A2 (A[25]), .ZN (Z[25]));
  XNOR2_X1 g1853(.A1 (n_48), .A2 (A[23]), .ZN (Z[23]));
  XNOR2_X1 g1854(.A1 (n_54), .A2 (A[21]), .ZN (Z[21]));
  XNOR2_X1 g1855(.A1 (n_49), .A2 (A[27]), .ZN (Z[27]));
  XNOR2_X1 g1856(.A1 (n_50), .A2 (A[19]), .ZN (Z[19]));
  XOR2_X1 g1857(.A1 (n_45), .A2 (A[5]), .Z (Z[5]));
  XOR2_X1 g1858(.A1 (n_56), .A2 (A[24]), .Z (Z[24]));
  HA_X1 g1859(.A (A[16]), .B (n_41), .CO (n_73), .S (Z[16]));
  XOR2_X1 g1860(.A1 (n_57), .A2 (A[20]), .Z (Z[20]));
  XOR2_X1 g1861(.A1 (n_51), .A2 (A[26]), .Z (Z[26]));
  XOR2_X1 g1862(.A1 (n_52), .A2 (A[22]), .Z (Z[22]));
  XOR2_X1 g1863(.A1 (n_53), .A2 (A[28]), .Z (Z[28]));
  XOR2_X1 g1864(.A1 (n_58), .A2 (A[18]), .Z (Z[18]));
  XNOR2_X1 g1865(.A1 (n_39), .A2 (A[13]), .ZN (Z[13]));
  XNOR2_X1 g1866(.A1 (n_43), .A2 (A[12]), .ZN (Z[12]));
  XNOR2_X1 g1867(.A1 (n_37), .A2 (A[11]), .ZN (Z[11]));
  XNOR2_X1 g1868(.A1 (n_38), .A2 (A[15]), .ZN (Z[15]));
  XNOR2_X1 g1869(.A1 (n_36), .A2 (A[14]), .ZN (Z[14]));
  XOR2_X1 g1870(.A1 (n_34), .A2 (A[3]), .Z (Z[3]));
  HA_X1 g1871(.A (A[8]), .B (n_32), .CO (n_60), .S (Z[8]));
  XOR2_X1 g1872(.A1 (n_42), .A2 (A[10]), .Z (Z[10]));
  NOR2_X1 g1873(.A1 (n_40), .A2 (n_6), .ZN (n_58));
  NOR2_X1 g1874(.A1 (n_40), .A2 (n_19), .ZN (n_57));
  NOR2_X1 g1875(.A1 (n_40), .A2 (n_25), .ZN (n_56));
  NAND3_X1 g1876(.A1 (n_41), .A2 (n_24), .A3 (A[24]), .ZN (n_55));
  NAND3_X1 g1877(.A1 (n_41), .A2 (n_18), .A3 (A[20]), .ZN (n_54));
  NOR4_X1 g1878(.A1 (n_40), .A2 (n_25), .A3 (n_1), .A4 (n_3), .ZN
       (n_53));
  NOR3_X1 g1879(.A1 (n_40), .A2 (n_19), .A3 (n_13), .ZN (n_52));
  NOR3_X1 g1880(.A1 (n_40), .A2 (n_25), .A3 (n_1), .ZN (n_51));
  NAND3_X1 g1881(.A1 (n_41), .A2 (n_7), .A3 (A[18]), .ZN (n_50));
  NAND4_X1 g1882(.A1 (n_24), .A2 (n_41), .A3 (n_2), .A4 (A[26]), .ZN
       (n_49));
  NAND4_X1 g1883(.A1 (n_41), .A2 (n_18), .A3 (n_14), .A4
(A[22]), .ZN
       (n_48));
  XNOR2_X1 g1884(.A1 (n_30), .A2 (A[7]), .ZN (Z[7]));
  HA_X1 g1885(.A (A[4]), .B (n_27), .CO (n_45), .S (Z[4]));
  XOR2_X1 g1886(.A1 (n_33), .A2 (A[6]), .Z (Z[6]));
  NAND2_X1 g1887(.A1 (n_32), .A2 (n_20), .ZN (n_43));
  NOR2_X1 g1888(.A1 (n_31), .A2 (n_9), .ZN (n_42));
  INV_X1 g1889(.I (n_41), .ZN (n_40));
  AND2_X1 g1890(.A1 (n_29), .A2 (n_23), .Z (n_41));
  NAND3_X1 g1891(.A1 (n_32), .A2 (n_20), .A3 (A[12]), .ZN (n_39));
  NAND4_X1 g1892(.A1 (n_32), .A2 (n_20), .A3 (n_16), .A4
(A[14]), .ZN
       (n_38));
  NAND3_X1 g1893(.A1 (n_32), .A2 (n_8), .A3 (A[10]), .ZN (n_37));
  NAND3_X1 g1894(.A1 (n_32), .A2 (n_20), .A3 (n_16), .ZN (n_36));
  HA_X1 g1895(.A (A[2]), .B (n_22), .CO (n_34), .S (Z[2]));
  NOR2_X1 g1896(.A1 (n_28), .A2 (n_5), .ZN (n_33));
  INV_X1 g1897(.I (n_32), .ZN (n_31));
  NOR3_X1 g1898(.A1 (n_28), .A2 (n_17), .A3 (n_5), .ZN (n_32));
  NAND3_X1 g1899(.A1 (n_27), .A2 (n_4), .A3 (A[6]), .ZN (n_30));
  NOR3_X1 g1900(.A1 (n_26), .A2 (n_9), .A3 (n_5), .ZN (n_29));
  INV_X1 g1901(.I (n_28), .ZN (n_27));
  NAND2_X1 g1902(.A1 (n_22), .A2 (n_11), .ZN (n_28));
  NAND3_X1 g1903(.A1 (n_22), .A2 (A[14]), .A3 (A[15]), .ZN (n_26));
  INV_X1 g1904(.I (n_25), .ZN (n_24));
  NAND4_X1 g1905(.A1 (n_18), .A2 (n_14), .A3 (A[22]), .A4
(A[23]), .ZN
       (n_25));
  NOR4_X1 g1906(.A1 (n_17), .A2 (n_12), .A3 (n_10), .A4 (n_15), .ZN
       (n_23));
  HA_X1 g1907(.A (A[1]), .B (A[0]), .CO (n_22), .S (Z[1]));
  NOR2_X1 g1908(.A1 (n_9), .A2 (n_12), .ZN (n_20));
  INV_X1 g1909(.I (n_19), .ZN (n_18));
  NAND3_X1 g1910(.A1 (n_7), .A2 (A[18]), .A3 (A[19]), .ZN (n_19));
  NAND2_X1 g1911(.A1 (A[7]), .A2 (A[6]), .ZN (n_17));
  INV_X1 g1912(.I (n_15), .ZN (n_16));
  NAND2_X1 g1913(.A1 (A[13]), .A2 (A[12]), .ZN (n_15));
  INV_X1 g1914(.I (n_13), .ZN (n_14));
  NAND2_X1 g1915(.A1 (A[20]), .A2 (A[21]), .ZN (n_13));
  NAND2_X1 g1916(.A1 (A[11]), .A2 (A[10]), .ZN (n_12));
  INV_X1 g1917(.I (n_10), .ZN (n_11));
  NAND2_X1 g1918(.A1 (A[3]), .A2 (A[2]), .ZN (n_10));
  INV_X1 g1919(.I (n_9), .ZN (n_8));
  NAND2_X1 g1920(.A1 (A[9]), .A2 (A[8]), .ZN (n_9));
  INV_X1 g1921(.I (n_6), .ZN (n_7));
  NAND2_X1 g1922(.A1 (A[16]), .A2 (A[17]), .ZN (n_6));
  INV_X1 g1923(.I (n_5), .ZN (n_4));
  NAND2_X1 g1924(.A1 (A[5]), .A2 (A[4]), .ZN (n_5));
  NAND2_X1 g1925(.A1 (A[27]), .A2 (A[26]), .ZN (n_3));
  INV_X1 g1926(.I (n_1), .ZN (n_2));
  NAND2_X1 g1927(.A1 (A[25]), .A2 (A[24]), .ZN (n_1));
endmodule

module timer(clk, en, rst, timer_fin);
  input clk, en, rst;
  output timer_fin;
  wire clk, en, rst;
  wire timer_fin;
  wire [28:0] counter;
  wire [1:0] state;
  wire UNCONNECTED, n_0, n_1, n_2, n_3, n_4, n_5, n_6;
  wire n_7, n_8, n_9, n_10, n_11, n_12, n_13, n_14;
  wire n_15, n_16, n_17, n_18, n_19, n_20, n_21, n_22;
  wire n_23, n_24, n_25, n_26, n_27, n_28, n_29, n_30;
  wire n_31, n_32, n_33, n_34, n_35, n_36, n_37, n_38;
  wire n_39, n_40, n_41, n_42, n_43, n_44, n_45, n_46;
  wire n_47, n_48, n_49, n_50, n_51, n_52, n_53, n_54;
  wire n_55, n_56, n_57, n_58, n_59, n_60, n_61, n_62;
  wire n_63, n_64, n_65, n_66, n_67, n_68, n_69, n_70;
  wire n_71, n_72, n_73, n_74, n_75, n_76, n_77, n_78;
  wire n_79, n_80, n_81, n_82, n_83, n_85, n_88, n_89;
  wire n_90, n_91, n_92, n_93, n_94, n_95, n_96, n_97;
  wire n_98, n_99, n_100, n_101, n_102, n_103, n_104, n_105;
  wire n_106, n_107, n_108, n_109, n_110, n_111, n_112, n_113;
  wire n_114, n_115;
  increment_unsigned_16 inc_add_373_40_2(.A (counter), .CI
(1'b1), .Z
       ({n_88, n_89, n_90, n_91, n_92, n_93, n_94, n_95, n_96, n_97,
       n_98, n_99, n_100, n_101, n_102, n_103, n_104, n_105, n_106,
       n_107, n_108, n_109, n_110, n_111, n_112, n_113, n_114,
n_115,
       UNCONNECTED}));
  INV_X1 g483(.I (rst), .ZN (n_85));
  SDFFRNQ_X1 \counter_reg[0] (.RN (n_85), .CLK (clk), .D (n_25), .SI
       (n_26), .SE (counter[0]), .Q (counter[0]));
  DFFRNQ_X1 \counter_reg[10] (.RN (n_85), .CLK (clk), .D (n_49), .Q
       (counter[10]));
  DFFRNQ_X1 \counter_reg[11] (.RN (n_85), .CLK (clk), .D (n_41), .Q
       (counter[11]));
  DFFRNQ_X1 \counter_reg[12] (.RN (n_85), .CLK (clk), .D (n_37), .Q
       (counter[12]));
  DFFRNQ_X1 \counter_reg[13] (.RN (n_85), .CLK (clk), .D (n_35), .Q
       (counter[13]));
  DFFRNQ_X1 \counter_reg[14] (.RN (n_85), .CLK (clk), .D (n_33), .Q
       (counter[14]));
  DFFRNQ_X1 \counter_reg[15] (.RN (n_85), .CLK (clk), .D (n_31), .Q
       (counter[15]));
  DFFRNQ_X1 \counter_reg[16] (.RN (n_85), .CLK (clk), .D (n_29), .Q
       (counter[16]));
  DFFRNQ_X1 \counter_reg[17] (.RN (n_85), .CLK (clk), .D (n_79), .Q
       (counter[17]));
  DFFRNQ_X1 \counter_reg[18] (.RN (n_85), .CLK (clk), .D (n_83), .Q
       (counter[18]));
  DFFRNQ_X1 \counter_reg[19] (.RN (n_85), .CLK (clk), .D (n_81), .Q
       (counter[19]));
  DFFRNQ_X1 \counter_reg[1] (.RN (n_85), .CLK (clk), .D (n_77), .Q
       (counter[1]));
  DFFRNQ_X1 \counter_reg[20] (.RN (n_85), .CLK (clk), .D (n_75), .Q
       (counter[20]));
  DFFRNQ_X1 \counter_reg[21] (.RN (n_85), .CLK (clk), .D (n_73), .Q
       (counter[21]));
  DFFRNQ_X1 \counter_reg[22] (.RN (n_85), .CLK (clk), .D (n_71), .Q
       (counter[22]));
  DFFRNQ_X1 \counter_reg[23] (.RN (n_85), .CLK (clk), .D (n_69), .Q
       (counter[23]));
  DFFRNQ_X1 \counter_reg[24] (.RN (n_85), .CLK (clk), .D (n_67), .Q
       (counter[24]));
  DFFRNQ_X1 \counter_reg[25] (.RN (n_85), .CLK (clk), .D (n_65), .Q
       (counter[25]));
  DFFRNQ_X1 \counter_reg[26] (.RN (n_85), .CLK (clk), .D (n_63), .Q
       (counter[26]));
  DFFRNQ_X1 \counter_reg[27] (.RN (n_85), .CLK (clk), .D (n_61), .Q
       (counter[27]));
  DFFRNQ_X1 \counter_reg[28] (.RN (n_85), .CLK (clk), .D (n_59), .Q
       (counter[28]));
  DFFRNQ_X1 \counter_reg[2] (.RN (n_85), .CLK (clk), .D (n_57), .Q
       (counter[2]));
  DFFRNQ_X1 \counter_reg[3] (.RN (n_85), .CLK (clk), .D (n_55), .Q
       (counter[3]));
  DFFRNQ_X1 \counter_reg[4] (.RN (n_85), .CLK (clk), .D (n_53), .Q
       (counter[4]));
  DFFRNQ_X1 \counter_reg[5] (.RN (n_85), .CLK (clk), .D (n_51), .Q
       (counter[5]));
  DFFRNQ_X1 \counter_reg[6] (.RN (n_85), .CLK (clk), .D (n_47), .Q
       (counter[6]));
  DFFRNQ_X1 \counter_reg[7] (.RN (n_85), .CLK (clk), .D (n_45), .Q
       (counter[7]));
  DFFRNQ_X1 \counter_reg[8] (.RN (n_85), .CLK (clk), .D (n_43), .Q
       (counter[8]));
  DFFRNQ_X1 \counter_reg[9] (.RN (n_85), .CLK (clk), .D (n_39), .Q
       (counter[9]));
  DFFSNQ_X1 \state_reg[0] (.SN (n_85), .CLK (clk), .D (n_27), .Q
       (state[0]));
  DFFRNQ_X1 \state_reg[1] (.RN (n_85), .CLK (clk), .D (n_19), .Q
       (state[1]));
  SDFFRNQ_X1 timer_fin_reg(.RN (n_85), .CLK (clk), .D (n_10), .SI
       (state[1]), .SE (state[0]), .Q (timer_fin));
  INV_X1 g10635(.I (n_82), .ZN (n_83));
  AOI22_X1 g10636(.A1 (n_26), .A2 (counter[18]), .B1 (n_98), .B2
       (n_25), .ZN (n_82));
  INV_X1 g10637(.I (n_80), .ZN (n_81));
  AOI22_X1 g10638(.A1 (n_26), .A2 (counter[19]), .B1 (n_97), .B2
       (n_25), .ZN (n_80));
  INV_X1 g10639(.I (n_78), .ZN (n_79));
  AOI22_X1 g10640(.A1 (n_26), .A2 (counter[17]), .B1 (n_99), .B2
       (n_25), .ZN (n_78));
  INV_X1 g10641(.I (n_76), .ZN (n_77));
  AOI22_X1 g10642(.A1 (n_26), .A2 (counter[1]), .B1 (n_25), .B2
       (n_115), .ZN (n_76));
  INV_X1 g10643(.I (n_74), .ZN (n_75));
  AOI22_X1 g10644(.A1 (n_26), .A2 (counter[20]), .B1 (n_96), .B2
       (n_25), .ZN (n_74));
  INV_X1 g10645(.I (n_72), .ZN (n_73));
  AOI22_X1 g10646(.A1 (n_26), .A2 (counter[21]), .B1 (n_95), .B2
       (n_25), .ZN (n_72));
```

```
  INV_X1 g10647(.I (n_70), .ZN (n_71));
  AOI22_X1 g10648(.A1 (n_26), .A2 (counter[22]), .B1 (n_94), .B2
       (n_25), .ZN (n_70));
  INV_X1 g10649(.I (n_68), .ZN (n_69));
  AOI22_X1 g10650(.A1 (n_26), .A2 (counter[23]), .B1 (n_93), .B2
       (n_25), .ZN (n_68));
  INV_X1 g10651(.I (n_66), .ZN (n_67));
  AOI22_X1 g10652(.A1 (n_26), .A2 (counter[24]), .B1 (n_92), .B2
       (n_25), .ZN (n_66));
  INV_X1 g10653(.I (n_64), .ZN (n_65));
  AOI22_X1 g10654(.A1 (n_26), .A2 (counter[25]), .B1 (n_91), .B2
       (n_25), .ZN (n_64));
  INV_X1 g10655(.I (n_62), .ZN (n_63));
  AOI22_X1 g10656(.A1 (n_26), .A2 (counter[26]), .B1 (n_90), .B2
       (n_25), .ZN (n_62));
  INV_X1 g10657(.I (n_60), .ZN (n_61));
  AOI22_X1 g10658(.A1 (n_26), .A2 (counter[27]), .B1 (n_89), .B2
       (n_25), .ZN (n_60));
  INV_X1 g10659(.I (n_58), .ZN (n_59));
  AOI22_X1 g10660(.A1 (n_26), .A2 (counter[28]), .B1 (n_88), .B2
       (n_25), .ZN (n_58));
  INV_X1 g10661(.I (n_56), .ZN (n_57));
  AOI22_X1 g10662(.A1 (n_26), .A2 (counter[2]), .B1 (n_25), .B2
       (n_114), .ZN (n_56));
  INV_X1 g10663(.I (n_54), .ZN (n_55));
  AOI22_X1 g10664(.A1 (n_26), .A2 (counter[3]), .B1 (n_25), .B2
       (n_113), .ZN (n_54));
  INV_X1 g10665(.I (n_52), .ZN (n_53));
  AOI22_X1 g10666(.A1 (n_26), .A2 (counter[4]), .B1 (n_25), .B2
       (n_112), .ZN (n_52));
  INV_X1 g10667(.I (n_50), .ZN (n_51));
  AOI22_X1 g10668(.A1 (n_26), .A2 (counter[5]), .B1 (n_25), .B2
       (n_111), .ZN (n_50));
  INV_X1 g10669(.I (n_48), .ZN (n_49));
  AOI22_X1 g10670(.A1 (n_26), .A2 (counter[10]), .B1 (n_25), .B2
       (n_106), .ZN (n_48));
  INV_X1 g10671(.I (n_46), .ZN (n_47));
  AOI22_X1 g10672(.A1 (n_26), .A2 (counter[6]), .B1 (n_25), .B2
       (n_110), .ZN (n_46));
  INV_X1 g10673(.I (n_44), .ZN (n_45));
  AOI22_X1 g10674(.A1 (n_26), .A2 (counter[7]), .B1 (n_25), .B2
       (n_109), .ZN (n_44));
  INV_X1 g10675(.I (n_42), .ZN (n_43));
  AOI22_X1 g10676(.A1 (n_26), .A2 (counter[8]), .B1 (n_25), .B2
       (n_108), .ZN (n_42));
  INV_X1 g10677(.I (n_40), .ZN (n_41));
  AOI22_X1 g10678(.A1 (n_26), .A2 (counter[11]), .B1 (n_25), .B2
       (n_105), .ZN (n_40));
  INV_X1 g10679(.I (n_38), .ZN (n_39));
  AOI22_X1 g10680(.A1 (n_26), .A2 (counter[9]), .B1 (n_25), .B2
       (n_107), .ZN (n_38));
  INV_X1 g10681(.I (n_36), .ZN (n_37));
  AOI22_X1 g10682(.A1 (n_26), .A2 (counter[12]), .B1 (n_25), .B2
       (n_104), .ZN (n_36));
  INV_X1 g10683(.I (n_34), .ZN (n_35));
  AOI22_X1 g10684(.A1 (n_26), .A2 (counter[13]), .B1 (n_25), .B2
       (n_103), .ZN (n_34));
  INV_X1 g10685(.I (n_32), .ZN (n_33));
  AOI22_X1 g10686(.A1 (n_26), .A2 (counter[14]), .B1 (n_25), .B2
       (n_102), .ZN (n_32));
  INV_X1 g10687(.I (n_30), .ZN (n_31));
  AOI22_X1 g10688(.A1 (n_26), .A2 (counter[15]), .B1 (n_25), .B2
       (n_101), .ZN (n_30));
  INV_X1 g10689(.I (n_28), .ZN (n_29));
  AOI22_X1 g10690(.A1 (n_26), .A2 (counter[16]), .B1 (n_25), .B2
       (n_100), .ZN (n_28));
  NOR2_X1 g10691(.A1 (n_25), .A2 (n_12), .ZN (n_27));
  OAI21_X2 g10692(.A1 (n_22), .A2 (state[0]), .B (n_24), .ZN
(n_26));
  AND3_X2 g10693(.A1 (n_23), .A2 (n_22), .A3 (n_2), .Z (n_25));
  AOI21_X1 g10694(.A1 (n_20), .A2 (n_17), .B (n_1), .ZN (n_24));
  NAND2_X1 g10695(.A1 (n_20), .A2 (n_9), .ZN (n_23));
  AOI21_X1 g10696(.A1 (n_13), .A2 (counter[28]), .B (n_21), .ZN
(n_22));
  NOR3_X1 g10699(.A1 (n_11), .A2 (n_14), .A3 (n_4), .ZN (n_21));
  NOR4_X1 g10700(.A1 (n_18), .A2 (n_14), .A3 (n_5), .A4 (n_4), .ZN
       (n_20));
  NAND2_X1 g10701(.A1 (n_16), .A2 (n_3), .ZN (n_19));
  NOR3_X1 g10702(.A1 (n_15), .A2 (counter[14]), .A3
(counter[10]), .ZN
       (n_18));
  NOR2_X1 g10703(.A1 (n_8), .A2 (state[0]), .ZN (n_17));
  NAND3_X1 g10704(.A1 (en), .A2 (n_1), .A3 (state[0]), .ZN (n_16));
  NAND2_X1 g10705(.A1 (n_7), .A2 (n_6), .ZN (n_15));
  NAND3_X1 g10706(.A1 (counter[28]), .A2 (counter[22]), .A3
       (counter[23]), .ZN (n_14));
  OR3_X1 g10707(.A1 (counter[27]), .A2 (counter[25]), .A3
       (counter[26]), .Z (n_13));
  AOI21_X1 g10708(.A1 (n_0), .A2 (state[0]), .B (state[1]), .ZN
(n_12));
```

```
  NOR4_X1 g10709(.A1 (counter[20]), .A2 (counter[17]), .A3
       (counter[19]), .A4 (counter[18]), .ZN (n_11));
  AND2_X1 g10710(.A1 (n_1), .A2 (timer_fin), .Z (n_10));
  INV_X1 g10711(.I (n_8), .ZN (n_9));
  NAND2_X1 g10712(.A1 (counter[15]), .A2 (counter[16]), .ZN (n_8));
  NAND2_X1 g10713(.A1 (counter[8]), .A2 (counter[9]), .ZN (n_7));
  NOR2_X1 g10714(.A1 (counter[11]), .A2 (counter[12]), .ZN (n_6));
  NOR2_X1 g10715(.A1 (counter[13]), .A2 (counter[14]), .ZN (n_5));
  NAND2_X1 g10716(.A1 (counter[21]), .A2 (counter[24]), .ZN (n_4));
  INV_X1 g10717(.I (n_2), .ZN (n_3));
  NOR2_X1 g10718(.A1 (n_1), .A2 (state[0]), .ZN (n_2));
  INV_X1 g10720(.I (state[1]), .ZN (n_1));
  INV_X1 g10738(.I (en), .ZN (n_0));
endmodule

module coin_casher(clk, power, return_coin, coin_insert,
game_finish,
     inserted_coin, coin_reject, eat_coins, spit_coin, wait_ready,
     game_start);
  input clk, power, return_coin, coin_insert, game_finish;
  input [2:0] inserted_coin;
  output coin_reject, eat_coins, spit_coin, wait_ready, game_start;
  wire clk, power, return_coin, coin_insert, game_finish;
  wire [2:0] inserted_coin;
  wire coin_reject, eat_coins, spit_coin, wait_ready, game_start;
  wire [3:0] state;
  wire [3:0] coin_counter;
  wire n_0, n_1, n_2, n_3, n_5, n_6, n_7, n_8;
  wire n_9, n_10, n_11, n_12, n_13, n_14, n_15, n_16;
  wire n_18, n_19, n_20, n_21, n_22, n_23, n_24, n_25;
  wire n_27, n_28, n_30, n_31, n_32, n_33, n_34, n_38;
  wire n_40, n_41, n_42, n_43, n_44, n_46, n_48, n_49;
  wire n_50, n_51, n_52, n_53, n_54, n_55, n_56, n_57;
  wire n_58, n_59, n_61, n_62, n_63, n_64, n_67, n_68;
  wire n_69, n_70, n_72, n_73, n_74, n_95, n_96, n_97;
  wire n_98, n_99, timer_finish;
  assign game_start = eat_coins;
  timer wait_for_user(clk, n_68, n_72, timer_finish);
  INV_X1 g533(.I (game_finish), .ZN (n_62));
  NOR2_X1 g654(.A1 (n_61), .A2 (n_64), .ZN (n_67));
  OAI21_X1 g655(.A1 (n_63), .A2 (n_74), .B (n_38), .ZN (n_72));
  NOR2_X1 g656(.A1 (n_63), .A2 (n_59), .ZN (n_68));
  NOR2_X1 g657(.A1 (n_74), .A2 (n_58), .ZN (wait_ready));
  INV_X1 g658(.I (coin_reject), .ZN (n_61));
  NOR2_X1 g659(.A1 (n_73), .A2 (n_56), .ZN (coin_reject));
  NOR3_X1 g660(.A1 (n_74), .A2 (n_56), .A3 (n_64), .ZN (spit_coin));
  NOR3_X1 g662(.A1 (n_73), .A2 (n_64), .A3 (state[0]), .ZN
(eat_coins));
  INV_X1 g663(.I (n_70), .ZN (n_59));
  NOR2_X1 g664(.A1 (n_57), .A2 (state[2]), .ZN (n_70));
  NAND2_X1 g665(.A1 (n_57), .A2 (state[2]), .ZN (n_73));
  OR2_X1 g666(.A1 (state[2]), .A2 (state[3]), .Z (n_74));
  NAND2_X1 g667(.A1 (n_56), .A2 (n_64), .ZN (n_63));
  INV_X1 g668(.I (n_69), .ZN (n_58));
  NOR2_X1 g669(.A1 (n_56), .A2 (state[1]), .ZN (n_69));
  INV_X1 g670(.I (state[1]), .ZN (n_64));
  INV_X1 g671(.I (state[3]), .ZN (n_57));
  INV_X1 g672(.I (state[0]), .ZN (n_56));
  DFFRNQ_X1 \coin_counter_reg[3] (.RN (n_3), .CLK (clk), .D
(n_54), .Q
       (coin_counter[3]));
  DFFRNQ_X1 \state_reg[0] (.RN (n_3), .CLK (clk), .D (n_55), .Q
       (state[0]));
  DFFRNQ_X1 \coin_counter_reg[2] (.RN (n_3), .CLK (clk), .D
(n_49), .Q
       (coin_counter[2]));
  DFFRNQ_X1 \state_reg[1] (.RN (n_3), .CLK (clk), .D (n_52), .Q
       (state[1]));
  DFFRNQ_X1 \state_reg[2] (.RN (n_3), .CLK (clk), .D (n_53), .Q
       (state[2]));
  NAND2_X1 g1139(.A1 (n_98), .A2 (n_51), .ZN (n_55));
  OAI21_X1 g1140(.A1 (n_27), .A2 (n_18), .B (n_50), .ZN (n_54));
  DFFRNQ_X1 \state_reg[3] (.RN (n_3), .CLK (clk), .D (n_95), .Q
       (state[3]));
  DFFRNQ_X1 \coin_counter_reg[1] (.RN (n_3), .CLK (clk), .D
(n_46), .Q
       (coin_counter[1]));
  NAND2_X1 g1143(.A1 (n_97), .A2 (n_44), .ZN (n_53));
  NAND3_X1 g1144(.A1 (n_38), .A2 (n_28), .A3 (n_44), .ZN (n_52));
  AOI21_X1 g1145(.A1 (wait_ready), .A2 (n_99), .B (n_96), .ZN
(n_51));
  OAI21_X1 g1146(.A1 (n_40), .A2 (n_24), .B (coin_counter[3]), .ZN
       (n_50));
  OAI21_X1 g1147(.A1 (n_27), .A2 (n_8), .B (n_48), .ZN (n_49));
  DFFRNQ_X1 \coin_counter_reg[0] (.RN (n_3), .CLK (clk), .D
(n_42), .Q
       (coin_counter[0]));
  NAND2_X1 g1149(.A1 (n_40), .A2 (coin_counter[2]), .ZN (n_48));
  OAI22_X1 g1151(.A1 (n_32), .A2 (n_5), .B1 (n_27), .B2
       (coin_counter[1]), .ZN (n_46));
```

```
AOI22_X1 g1153(.A1 (n_31), .A2 (n_14), .B1 (n_67), .B2 (n_62), .ZN
     (n_44));
NAND2_X1 g1154(.A1 (n_30), .A2 (n_14), .ZN (n_43));
INV_X1 g1155(.I (n_41), .ZN (n_42));
AOI21_X1 g1156(.A1 (n_21), .A2 (coin_counter[0]), .B (n_23), .ZN
     (n_41));
OAI21_X1 g1157(.A1 (n_10), .A2 (coin_counter[1]), .B (n_32), .ZN
     (n_40));
NOR2_X1 g1163(.A1 (n_25), .A2 (n_20), .ZN (n_34));
OAI21_X1 g1164(.A1 (n_16), .A2 (inserted_coin[0]), .B (n_20), .ZN
     (n_33));
NOR2_X1 g1165(.A1 (n_21), .A2 (n_23), .ZN (n_32));
INV_X1 g1166(.I (n_30), .ZN (n_31));
NAND2_X1 g1167(.A1 (n_19), .A2 (coin_counter[1]), .ZN (n_30));
OAI21_X1 g1169(.A1 (n_12), .A2 (coin_insert), .B (wait_ready), .ZN
     (n_28));
NAND2_X1 g1170(.A1 (n_9), .A2 (coin_counter[0]), .ZN (n_27));
NAND2_X1 g1172(.A1 (n_1), .A2 (n_13), .ZN (n_25));
NOR2_X1 g1173(.A1 (n_10), .A2 (coin_counter[2]), .ZN (n_24));
NOR2_X1 g1174(.A1 (n_10), .A2 (coin_counter[0]), .ZN (n_23));
OAI21_X1 g1175(.A1 (state[1]), .A2 (state[2]), .B (state[3]), .ZN
     (n_22));
NOR3_X1 g1176(.A1 (n_72), .A2 (n_9), .A3 (spit_coin), .ZN (n_21));
NOR3_X1 g1177(.A1 (n_74), .A2 (n_64), .A3 (state[0]), .ZN (n_20));
NOR3_X1 g1178(.A1 (coin_counter[2]), .A2 (coin_counter[0]), .A3
     (coin_counter[3]), .ZN (n_19));
NAND3_X1 g1179(.A1 (coin_counter[1]), .A2 (coin_counter[2]), .A3
     (n_7), .ZN (n_18));
NAND2_X1 g1181(.A1 (n_0), .A2 (inserted_coin[2]), .ZN (n_16));
NAND2_X1 g1182(.A1 (n_2), .A2 (n_68), .ZN (n_15));
INV_X1 g1183(.I (n_14), .ZN (n_13));
NOR2_X1 g1184(.A1 (n_63), .A2 (n_73), .ZN (n_14));
INV_X1 g1185(.I (n_11), .ZN (n_12));
NOR2_X1 g1186(.A1 (timer_finish), .A2 (return_coin), .ZN (n_11));
INV_X1 g1187(.I (n_10), .ZN (n_9));
NAND2_X1 g1188(.A1 (n_69), .A2 (n_70), .ZN (n_10));
NAND2_X1 g1189(.A1 (n_6), .A2 (coin_counter[1]), .ZN (n_8));
INV_X1 g1190(.I (coin_counter[3]), .ZN (n_7));
INV_X1 g1191(.I (coin_counter[2]), .ZN (n_6));
INV_X1 g1192(.I (coin_counter[1]), .ZN (n_5));
INV_X1 g1194(.I (power), .ZN (n_3));
INV_X1 g1195(.I (n_67), .ZN (n_2));
INV_X1 g1196(.I (wait_ready), .ZN (n_1));
INV_X1 g1197(.I (inserted_coin[1]), .ZN (n_0));
INV_X1 g1200(.I (eat_coins), .ZN (n_38));
```

```
  NAND2_X1 g2(.A1 (n_15), .A2 (n_43), .ZN (n_95));
  OAI21_X1 g1201(.A1 (n_13), .A2 (n_19), .B (n_33), .ZN (n_96));
  NOR2_X1 g1202(.A1 (eat_coins), .A2 (n_20), .ZN (n_97));
  NAND2_X1 g1203(.A1 (n_22), .A2 (n_34), .ZN (n_98));
  NAND2_X1 g1204(.A1 (n_11), .A2 (coin_insert), .ZN (n_99));
endmodule
```

[3] Reports：

“coin_casher_area.rpt”:

```
============================================================
  Generated by:          Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
  Generated on:          Oct 07 2021  10:08:32 pm
  Module:                coin_casher
  Technology library:    NanGate_15nm_OCL revision 1.0
  Operating conditions:  worst_low (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

     Instance       Cells  Cell Area  Net Area  Total Area  Wireload
-------------------------------------------------------------------------
coin_casher           273       113         0        113    <none> (D)
  wait_for_user       195        87         0         87    <none> (D)
    inc_add_373_40_2   78        25         0         25    <none> (D)

 (D) = wireload is default in technology library
```

“coin_casher_gates.rpt”:

```
============================================================
  Generated by:          Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
  Generated on:          Oct 07 2021  10:08:32 pm
  Module:                coin_casher
  Technology library:    NanGate_15nm_OCL revision 1.0
  Operating conditions:  worst_low (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================


   Gate     Instances   Area        Library
---------------------------------------------------
```

```
AND2_X1            2    0.590    NanGate_15nm_OCL
AND3_X2            1    0.393    NanGate_15nm_OCL
AOI21_X1           5    1.475    NanGate_15nm_OCL
AOI22_X1          29    9.978    NanGate_15nm_OCL
DFFRNQ_X1         37   47.284    NanGate_15nm_OCL
DFFSNQ_X1          1    1.278    NanGate_15nm_OCL
HA_X1              5    3.195    NanGate_15nm_OCL
INV_X1            65    9.585    NanGate_15nm_OCL
NAND2_X1          34    6.685    NanGate_15nm_OCL
NAND3_X1          13    3.834    NanGate_15nm_OCL
NAND4_X1           4    1.376    NanGate_15nm_OCL
NOR2_X1           24    4.719    NanGate_15nm_OCL
NOR3_X1           11    3.244    NanGate_15nm_OCL
NOR4_X1            4    1.376    NanGate_15nm_OCL
OAI21_X1           9    2.654    NanGate_15nm_OCL
OAI21_X2           1    0.442    NanGate_15nm_OCL
OAI22_X1           1    0.344    NanGate_15nm_OCL
OR2_X1             1    0.295    NanGate_15nm_OCL
OR3_X1             1    0.393    NanGate_15nm_OCL
SDFFRNQ_X1         2    3.244    NanGate_15nm_OCL
XNOR2_X1          11    4.866    NanGate_15nm_OCL
XOR2_X1           12    5.308    NanGate_15nm_OCL
----------------------------------------------------
total            273  112.558


   Type    Instances   Area  Area %
------------------------------------
sequential        40  51.806   46.0
inverter          65   9.585    8.5
logic            168  51.167   45.5
------------------------------------
total            273 112.558  100.0
```

“coin_casher_power.rpt”:

```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
  Generated on:           Oct 07 2021  10:08:32 pm
  Module:                 coin_casher
  Technology library:     NanGate_15nm_OCL revision 1.0
  Operating conditions:   worst_low (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                       Leakage    Dynamic     Total
     Instance      Cells Power(nW)  Power(nW)  Power(nW)
-----------------------------------------------------------
coin_casher          273    75.590 733616.399 733691.989
  wait_for_user      195    56.807 473757.484 473814.292
    inc_add_373_40_2   78    18.103      0.000     18.103
```

“coin_casher_timing.rpt”:

```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
  Generated on:           Oct 07 2021  10:08:32 pm
  Module:                 coin_casher
  Technology library:     NanGate_15nm_OCL revision 1.0
  Operating conditions:   worst_low (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

       Pin            Type      Fanout Load Slew Delay Arrival
                                       (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------------
(clock clk)           launch                             0 R
wait_for_user
```

```
 counter_reg[17]/CLK                          0            0 R
 counter_reg[17]/Q     DFFRNQ_X1       4  4.8   10   +18      18 R
 inc_add_373_40_2/A[17]
   g1922/A2                                  +0      18
   g1922/ZN          NAND2_X1       2  1.7    6    +6      24 F
   g1921/I                                 +0      24
   g1921/ZN          INV_X1         2  1.8    4    +4      28 R
   g1910/A1                                  +0      28
   g1910/ZN          NAND3_X1       3  2.6   11    +7      34 F
   g1909/I                                 +0      34
   g1909/ZN          INV_X1         3  2.6    6    +6      40 R
   g1905/A1                                  +0      40
   g1905/ZN          NAND4_X1       4  3.5   18   +10      51 F
   g1904/I                                 +0      51
   g1904/ZN          INV_X1         2  1.7    7    +6      57 R
   g1882/A1                                  +0      57
   g1882/ZN          NAND4_X1       1  1.5   10    +6      64 F
   g1855/A1                                  +0      64
   g1855/ZN          XNOR2_X1       1  0.9    3   +10      74 R
 inc_add_373_40_2/Z[27]
 g10658/B1                                   +0      74
 g10658/ZN            AOI22_X1       1  0.8   11    +4      78 F
 g10657/I                                  +0      78
 g10657/ZN            INV_X1         1  0.6    4    +4      82 R
 counter_reg[27]/D     DFFRNQ_X1                    +0      82
 counter_reg[27]/CLK   setup                  0    +8      90 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)           capture                          100 R
--------------------------------------------------------------------
Cost Group   : 'clk' (path_group 'clk')
Timing slack :      10ps
Start-point  : wait_for_user/counter_reg[17]/CLK
End-point    : wait_for_user/counter_reg[27]/D
```