|  |
| --- |
| 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**

**Changes in coin\_casher FSM**

**Added Module Descriptios**

**Waveforms**

Waveforms from the new FSM

Waveforms from the mapped Verilog *Scenario 4*

**Mapped Verilog**

**Appendix**

[1] Modified FSM & testbench

[2] Generated Verilog

[3] Reports

*Page 1*

*Page 1*

*Page 1*

*Page 2*

*Page 2*

*Page 3*

*Page 4*

*Page 5*

*Page 5*

*Page 12*

*Page 23*

**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”.

Chart, diagram

Description automatically generated

1. Overview waveform of the new FSM:

A picture containing graphical user interface

Description automatically generated

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.*

Graphical user interface

Description automatically generated with low confidence

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

A picture containing diagram

Description automatically generated

**Mapped Verilog**

The generated design contains 273 cells in total with a time slack of 10ps (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.

**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 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

**[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