# Synthesized Verilog Project

Eric Wu 75286914

University of British Columbia

# Modified FSM modules

Due to delay in actual gates, combination logic in my original verilog design failed to operate normally. To fix that, both “fsm” and “num_capture” we’re modified.

num_capture:

```
/*
Number button capturer
(c) Copyright Eric Wu
All rights reserved.

Author: Eric Wu
Email: ew820120@gmail.com
Student ID: 75286914

*/

/*
Description:
A number button capturer that detects button press and saves pressed value.

Inputs:
   zero        - button signal for zero (pressed = 1, unpressed = 0)
   one         - button signal for one (pressed = 1, unpressed = 0)
   two         - button signal for two (pressed = 1, unpressed = 0)
   three       - button signal for three (pressed = 1, unpressed = 0)
   four        - button signal for four (pressed = 1, unpressed = 0)
   five        - button signal for five (pressed = 1, unpressed = 0)
   six         - button signal for six (pressed = 1, unpressed = 0)
   seven       - button signal for seven (pressed = 1, unpressed = 0)
   eight       - button signal for eight (pressed = 1, unpressed = 0)
   nine        - button signal for nine (pressed = 1, unpressed = 0)
   clk         - clock

Outputs:
   press_val         - value of number pressed
   num_press         - edge captured signal of button press
*/
```

```
36  module num_capture(input logic zero, one, two, three, four, five, six, seven, eight, nine,
37                     input logic clk,
38                     output logic [3:0] press_val,
39                     output logic num_press);
40
41  /*
42  edge detector: detects key press and provides output edge captured signal num_press.
43  two inverters were used as delay for the output num_press, amount of delay will be configured after synthesis.
44  */
45  logic key_press;
46  logic delay_reg;
47  logic np;
48
49
50  assign key_press = zero | one | two | three | four | five | six | seven | eight | nine;
51
52  always_ff @(posedge clk)
53  delay_reg <= key_press;
54
55  assign num_press = ~delay_reg & key_press & (press_val != 4'b1111);
56
57
58
59  //gets number press value each key press
60  always_comb
61  case({zero, one, two, three, four, five, six ,seven ,eight, nine})
62  10'b0000000001: press_val = 4'd9;
63  10'b0000000010: press_val = 4'd8;
64  10'b0000000100: press_val = 4'd7;
65  10'b0000001000: press_val = 4'd6;
66  10'b0000010000: press_val = 4'd5;
67  10'b0000100000: press_val = 4'd4;
68  10'b0001000000: press_val = 4'd3;
69  10'b0010000000: press_val = 4'd2;
70  10'b0100000000: press_val = 4'd1;
71  10'b1000000000: press_val = 4'd0;
72  default: press_val = 4'b1111; //idle
73  endcase
74
75
76
77  endmodule
```

For my fsm to operate normally, press_val has to be ready before num_press gets logic 1. (For if statements in S11- S20)

I changed “num_press = ~delay_reg & key_press” to “num_press = ~delay_reg & key_press & (press_val != 4’b1111)” (line 55) to make sure press_val has the pressed number value before num_press is declared.

For press_val assignment, I changed “always_ff @(posedge num_press)” to “always_comb” (line 60) for the same reason mentioned above. My FSM design was also changed because of this which will be discussed below.

fsm:

```
/*
Hotel safe box lock FSM
(c) Copyright Eric Wu
All rights reserved.

Author: Eric Wu
Email: ew820120@gmail.com
Student ID: 75286914
*/

/*
Description:
A FSM that goes through three main stages.
- Setting up security code S0~S4
- Setting up customer code S5~S9
- Unlocking and Resetting S10~S20

Security code saved in sc[15:0]
Customer code saved in cc[15:0]

Inputs:
   clk         - clock
   zero        - button signal for zero (pressed = 1, unpressed = 0)
   one         - button signal for one (pressed = 1, unpressed = 0)
   two         - button signal for two (pressed = 1, unpressed = 0)
   three       - button signal for three (pressed = 1, unpressed = 0)
   four        - button signal for four (pressed = 1, unpressed = 0)
   five        - button signal for five (pressed = 1, unpressed = 0)
   six         - button signal for six (pressed = 1, unpressed = 0)
   seven       - button signal for seven (pressed = 1, unpressed = 0)
   eight       - button signal for eight (pressed = 1, unpressed = 0)
   nine        - button signal for nine (pressed = 1, unpressed = 0)
   enter       - button signal for enter (pressed = 1, unpressed = 0)
   init        - signal for force initialization (state goes to S0)
   reset       - button signal for resetting cc[15:0] (pressed = 1, unpressed = 0)
   clear       - button signal for clearing previous entered numbers (pressed = 1, unpressed = 0)

Outputs:
   press_val         - value of number pressed (for HEX display purposes, which is not included in this project)
   lock              - lock signal, locked = 1, unlocked = 0 (controls the actual lock)

*/
```

```
module fsm(input logic clk,
           input logic zero, one, two, three, four, five, six, seven, eight, nine, enter,
           input logic init, reset, clear,
           output logic [3:0] press_val,
           output logic lock,

         //Just for debugging
           output logic [5:0] state,
           output logic [15:0] sc, cc);

//instantiation of num_capture
logic num_press;
num_capture asdf(.zero(zero), .one(one), .two(two), .three(three), .four(four), .five(five), .six(six), .seven(seven), .eight(eight), .nine(nine),
                 .clk(clk), .press_val(press_val), .num_press(num_press));

//internal signals
logic [7:0] ctrl;
logic [14:0] superstate, next_superstate;


//state register
always_ff @(posedge clk)
if (init) superstate <= {9'b0_00000011, 6'd0};
else superstate <= next_superstate;

//outputs(state_bits and further comb logic)
assign lock = superstate[14];
assign ctrl = superstate[13:6];
assign state = superstate[5:0];
```

```
76    //sc, cc assignment using control bit
77    always_comb
78    case(ctrl)
79    8'b00000011: sc = 16'd0;
80    8'b00000001: sc[3:0] = press_val;
81    8'b00000010: sc[7:4] = press_val;
82    8'b00000100: sc[11:8] = press_val;
83    8'b00001000: sc[15:12] = press_val;
84    8'b00110000: cc = 16'd0;
85    8'b00010000: cc[3:0] = press_val;
86    8'b00100000: cc[7:4] = press_val;
87    8'b01000000: cc[11:8] = press_val;
88    8'b10000000: cc[15:12] = press_val;
89    default: begin
90    sc = sc;
91    cc = cc;
92    end
93
94    endcase
```

```
 97    //next state logic
 98    always_comb
 99    case(superstate)
100
101    //Project2 Added states(for writing)
102
103    //S21
104    //saves first number for sc
105    {9'b0_00000001, 6'd21}: next_superstate = {9'b0_00000000, 6'd1};
106
107    //S22
108    //saves second number for sc
109    {9'b0_00000010, 6'd22}: next_superstate = {9'b0_00000000, 6'd2};
110
111    //S23
112    //saves third number for sc
113    {9'b0_00000100, 6'd23}: next_superstate = {9'b0_00000000, 6'd3};
114
115    //S24
116    //saves fourth number for sc
117    {9'b0_00001000, 6'd24}: next_superstate = {9'b0_00000000, 6'd4};
118
119    //S25
120    //saves first number for cc
121    {9'b0_00010000, 6'd25}: next_superstate = {9'b0_00000000, 6'd6};
122
123    //S26
124    //saves second number for cc
125    {9'b0_00100000, 6'd26}: next_superstate = {9'b0_00000000, 6'd7};
126
127    //S27
128    //saves third number for cc
129    {9'b0_01000000, 6'd27}: next_superstate = {9'b0_00000000, 6'd8};
130
131    //S28
132    //saves fourth number for cc
133    {9'b0_10000000, 6'd28}: next_superstate = {9'b0_00000000, 6'd9};
134
135
```

```
135
136     /*Security code setup*/
137
138     //S0
139     //wait for number press
140   {9'b0_00000011, 6'd0}: begin
141     if (num_press) next_superstate = {9'b0_00000001, 6'd21};
142     else next_superstate = {9'b0_00000011, 6'd0};
143     end
144
145     //S1
146     //wait for number press
147   {9'b0_00000000, 6'd1}: begin
148     if (clear) next_superstate = {9'b0_00000011, 6'd0};
149     else if (num_press) next_superstate = {9'b0_00000010, 6'd22};
150     else next_superstate = {9'b0_00000000, 6'd1};
151     end
152
153     //S2
154     //wait for number press
155   {9'b0_00000000, 6'd2}: begin
156     if (clear) next_superstate = {9'b0_00000011, 6'd0};
157     else if (num_press) next_superstate = {9'b0_00000100, 6'd23};
158     else next_superstate = {9'b0_00000000, 6'd2};
159     end
160
161     //S3
162     //wait for number press
163   {9'b0_00000000, 6'd3}: begin
164     if (clear) next_superstate = {9'b0_00000011, 6'd0};
165     else if (num_press) next_superstate = {9'b0_00001000, 6'd24};
166     else next_superstate = {9'b0_00000000, 6'd3};
167     end
168
169     //S4
170     //wait for enter press
171   {9'b0_00000000, 6'd4}: begin
172     if (clear) next_superstate = {9'b0_00000011, 6'd0};
173     else if (enter) next_superstate = {9'b0_00110000, 6'd5};
174     else next_superstate = {9'b0_00000000, 6'd4};
175     end
176
```

```
179     /*Customer code setup*/
180
181     //S5
182     //wait for number press
183   {9'b0_00110000, 6'd5}: begin
184     if (num_press) next_superstate = {9'b0_00010000, 6'd25};
185     else next_superstate = {9'b0_00110000, 6'd5};
186     end
187
188     //S6
189     //wait for number press
190   {9'b0_00000000, 6'd6}: begin
191     if (clear) next_superstate = {9'b0_00110000, 6'd5};
192     else if (num_press) next_superstate = {9'b0_00100000, 6'd26};
193     else next_superstate = {9'b0_00000000, 6'd6};
194     end
195
196     //S7
197     //wait for number press
198   {9'b0_00000000, 6'd7}: begin
199     if (clear) next_superstate = {9'b0_00110000, 6'd5};
200     else if (num_press) next_superstate = {9'b0_01000000, 6'd27};
201     else next_superstate = {9'b0_00000000, 6'd7};
202     end
203
204     //S8
205     //wait for number press
206   {9'b0_00000000, 6'd8}: begin
207     if (clear) next_superstate = {9'b0_00110000, 6'd5};
208     else if (num_press) next_superstate = {9'b0_10000000, 6'd28};
209     else next_superstate = {9'b0_00000000, 6'd8};
210     end
211
212     //S9
213     //wait for enter press
214   {9'b0_00000000, 6'd9}: begin
215     if (clear) next_superstate = {9'b0_00110000, 6'd5};
216     else if (enter) next_superstate = {9'b1_00000000, 6'd10};
217     else next_superstate = {9'b0_00000000, 6'd9};
218     end
219
```

```
220
221   //Unlock and Reset states
222
223
224   //S10
225   //Wait for reset or num_press, compare press_val with cc[3:0] if num_press
226  {9'b1_00000000, 6'd10}: begin
227   if (reset) next_superstate = {9'b1_00000000, 6'd11};
228  else if (num_press) begin
229      if (press_val == cc[3:0]) next_superstate = {9'b1_00000000, 6'd18};
230      else next_superstate = {9'b1_00000000, 6'd15};
231     end
232   else next_superstate = {9'b1_00000000, 6'd10};
233   end
234
235
236   //S11
237   //compare sc[3:0] with press_val
238  {9'b1_00000000, 6'd11}: begin
239   if (clear) next_superstate = {9'b1_00000000, 6'd10};
240  else if (num_press) begin
241      if (press_val == sc[3:0]) next_superstate = {9'b1_00000000, 6'd12};
242      else next_superstate = {9'b1_00000000, 6'd15};
243     end
244   else next_superstate = {9'b1_00000000, 6'd11};
245   end
246
247   //S12
248   //compare sc[7:4] with press_val
249  {9'b1_00000000, 6'd12}: begin
250   if (clear) next_superstate = {9'b1_00000000, 6'd10};
251  else if (num_press) begin
252      if (press_val == sc[7:4]) next_superstate = {9'b1_00000000, 6'd13};
253      else next_superstate = {9'b1_00000000, 6'd16};
254     end
255   else next_superstate = {9'b1_00000000, 6'd12};
256   end
257
258   //S13
259   //compare sc[11:8] with press_val
260  {9'b1_00000000, 6'd13}: begin
261   if (clear) next_superstate = {9'b1_00000000, 6'd10};
262  else if (num_press) begin
263      if (press_val == sc[11:8]) next_superstate = {9'b1_00000000, 6'd14};
264      else next_superstate = {9'b1_00000000, 6'd17};
265     end
266   else next_superstate = {9'b1_00000000, 6'd13};
267   end
```

```
269     //S14
270     //compare sc[15:12] with press_val
271   {9'b1_00000000, 6'd14}: begin
272     if (clear) next_superstate = {9'b1_00000000, 6'd10};
273   else if (num_press) begin
274       if (press_val == sc[15:12]) next_superstate = {9'b0_00110000, 6'd5};
275       else next_superstate = {9'b1_00000000, 6'd10};
276       end
277    else next_superstate = {9'b1_00000000, 6'd14};
278    end
279
280
281    //S15
282    //counterfeit state
283   {9'b1_00000000, 6'd15}: begin
284    if (clear) next_superstate = {9'b1_00000000, 6'd10};
285    else if (num_press) next_superstate = {9'b1_00000000, 6'd16};
286    else next_superstate = {9'b1_00000000, 6'd15};
287    end
288
289    //S16
290    //counterfeit state
291   {9'b1_00000000, 6'd16}: begin
292    if (clear) next_superstate = {9'b1_00000000, 6'd10};
293    else if (num_press) next_superstate = {9'b1_00000000, 6'd17};
294    else next_superstate = {9'b1_00000000, 6'd16};
295    end
296
297    //S17
298    //counterfeit state
299   {9'b1_00000000, 6'd17}: begin
300    if (clear) next_superstate = {9'b1_00000000, 6'd10};
301    else if (num_press) next_superstate = {9'b1_00000000, 6'd10};
302    else next_superstate = {9'b1_00000000, 6'd17};
303    end
304
305
306    //S18
307    //compare cc[7:4] with press_val
308   {9'b1_00000000, 6'd18}: begin
309    if (clear) next_superstate = {9'b1_00000000, 6'd10};
310   else if (num_press) begin
311       if (press_val == cc[7:4]) next_superstate = {9'b1_00000000, 6'd19};
312       else next_superstate = {9'b1_00000000, 6'd16};
313       end
314    else next_superstate = {9'b1_00000000, 6'd18};
315    end
316
```

```
328     //S20
329     //compare cc[15:12] with press_val
330   {9'b1_00000000, 6'd20}: begin
331    if (clear) next_superstate = {9'b1_00000000, 6'd10};
332   else if (num_press) begin
333       if (press_val == cc[15:12]) next_superstate = {9'b0_00110000, 6'd5};
334       else next_superstate = {9'b1_00000000, 6'd10};
335       end
336    else next_superstate = {9'b1_00000000, 6'd20};
337    end
338
339
340    default: next_superstate = {9'b0_00000011, 6'd0};
341
342    endcase
343
344    endmodule
345
```

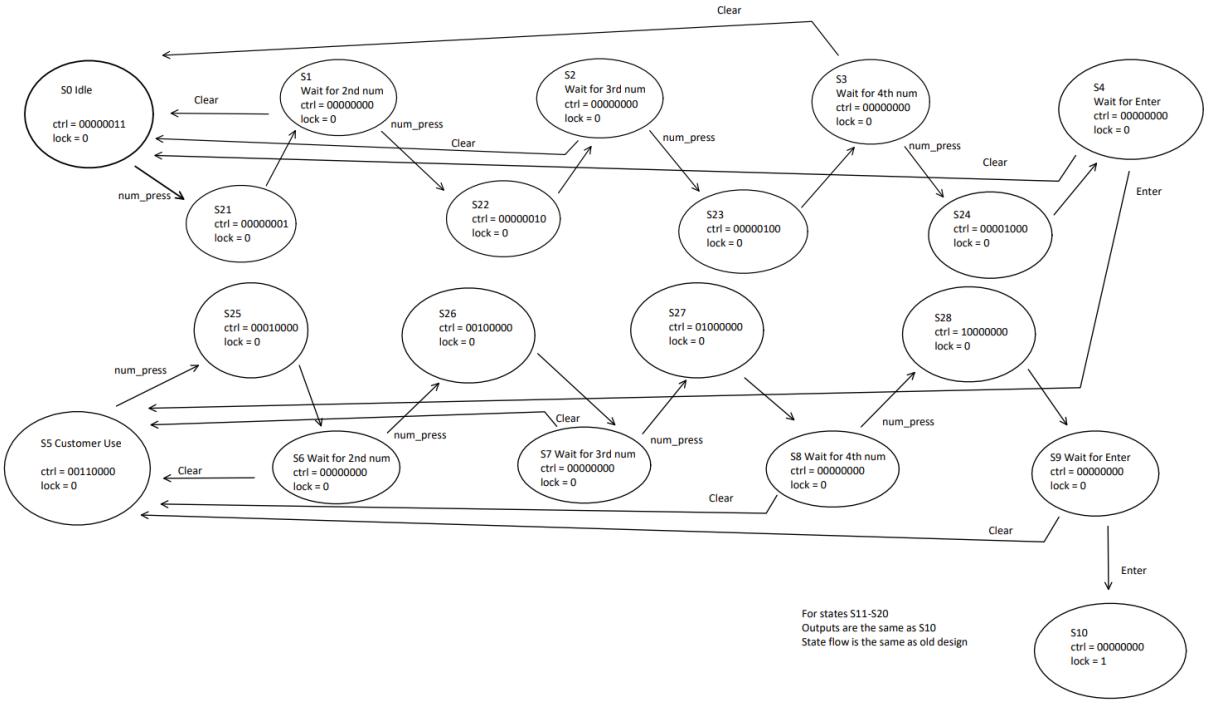
For my previous fsm design, press_value was being constantly written into sc or cc when waiting for next number press. This is problematic when actual delay is added. To fix that, instead of giving sc and cc value during a state, I created a ctrl signal that controls the assigning of sc and cc and declared it as state bits. To work with the new design, 8 more states were added for writing (line 100 -> 135). Now S0 – S9 waits for the next signal without constantly writing press_val to cc or sc making the fsm work even when press_val gets the new value before num_press (which is what we're getting now with new num_capture design). The output lock signal was also change into a state bit as I was experiencing some bugs when I assigned it in states. S10 – S20 works exactly the same as before.

I used the same testbench for simulation because the new design has same behavior as the old design, just a bonus state when writing.

# Updated State Diagram

States after S10 are the same so it is not included.

Block diagrams are not included because it is the same as the old design. (Outputs were not changed)



Mapped Verilog generated by RTL Compiler

```
1
2   // Generated by Cadence Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
3
4   // Verification Directory fv/fsm
5
6   module num_capture(zero, one, two, three, four, five, six, seven,
7        eight, nine, clk, press_val, num_press);
8     input zero, one, two, three, four, five, six, seven, eight, nine, clk;
9     output [3:0] press_val;
10    output num_press;
11    wire zero, one, two, three, four, five, six, seven, eight, nine, clk;
12    wire [3:0] press_val;
13    wire num_press;
14    wire delay_reg, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
15    wire n_8, n_9, n_10, n_11, n_12, n_13, n_14, n_15;
16    wire n_16, n_17, n_18, n_19, n_20, n_21, n_22, n_23;
17    wire n_24, n_25, n_26, n_27, n_28, n_29, n_30, n_31;
18    wire n_32, n_33, n_34, n_35, n_36, n_37, n_38, n_39;
19    wire n_40, n_41, n_43, n_44, n_45;
20    NOR2_X1 g638(.A1 (n_44), .A2 (delay_reg), .ZN (num_press));
21    NAND2_X1 g639(.A1 (n_45), .A2 (n_28), .ZN (press_val[3]));
22    NAND2_X1 g640(.A1 (n_45), .A2 (n_41), .ZN (press_val[2]));
23    NAND4_X1 g641(.A1 (n_45), .A2 (n_39), .A3 (n_31), .A4 (n_27), .ZN
24         (press_val[1]));
25    INV_X1 g642(.I (n_45), .ZN (n_44));
26    NAND4_X1 g643(.A1 (n_43), .A2 (n_31), .A3 (n_28), .A4 (n_27), .ZN
27         (n_45));
28    AND3_X2 g644(.A1 (n_41), .A2 (n_37), .A3 (n_36), .Z (n_43));
29    AND4_X1 g645(.A1 (n_38), .A2 (n_37), .A3 (n_32), .A4 (n_31), .Z
30         (press_val[0]));
31    NOR2_X1 g646(.A1 (n_40), .A2 (n_34), .ZN (n_41));
32    NAND2_X1 g647(.A1 (n_39), .A2 (n_25), .ZN (n_40));
33    NOR2_X1 g648(.A1 (n_33), .A2 (n_35), .ZN (n_39));
34    NOR2_X1 g649(.A1 (n_34), .A2 (n_20), .ZN (n_38));
35    NAND3_X1 g650(.A1 (n_26), .A2 (n_4), .A3 (zero), .ZN (n_37));
36    NAND3_X1 g651(.A1 (n_26), .A2 (n_5), .A3 (one), .ZN (n_36));
37    NOR4_X1 g652(.A1 (n_24), .A2 (six), .A3 (five), .A4 (four), .ZN
38         (n_35));
39    NOR2_X1 g653(.A1 (n_29), .A2 (six), .ZN (n_34));
40    INV_X1 g654(.I (n_32), .ZN (n_33));
41    NAND2_X1 g655(.A1 (n_30), .A2 (six), .ZN (n_32));
42    NAND4_X1 g656(.A1 (n_16), .A2 (n_7), .A3 (n_1), .A4 (two), .ZN
43         (n_31));
44    DFFRNQ_X1 delay_reg_reg(.RN (1'b1), .CLK (clk), .D (n_23), .Q
45         (delay_reg));
46    NOR2_X1 g658(.A1 (n_22), .A2 (four), .ZN (n_30));
47    NAND2_X1 g659(.A1 (n_21), .A2 (four), .ZN (n_29));
48    AOI21_X1 g660(.A1 (n_19), .A2 (nine), .B (n_20), .ZN (n_28));
49    NAND3_X1 g661(.A1 (n_16), .A2 (n_10), .A3 (three), .ZN (n_27));
```

```
50      NOR3_X1 g662(.A1 (n_15), .A2 (three), .A3 (two), .ZN (n_26));
51      NAND3_X1 g663(.A1 (n_18), .A2 (n_8), .A3 (five), .ZN (n_25));
52      NAND2_X1 g664(.A1 (n_18), .A2 (seven), .ZN (n_24));
53      NAND2_X1 g665(.A1 (n_18), .A2 (n_11), .ZN (n_23));
54      INV_X1 g666(.I (n_21), .ZN (n_22));
55      NOR3_X1 g667(.A1 (n_17), .A2 (five), .A3 (seven), .ZN (n_21));
56      NOR3_X1 g668(.A1 (n_14), .A2 (n_3), .A3 (nine), .ZN (n_20));
57      NOR2_X1 g669(.A1 (n_14), .A2 (eight), .ZN (n_19));
58      INV_X1 g670(.I (n_18), .ZN (n_17));
59      NOR3_X1 g671(.A1 (n_12), .A2 (eight), .A3 (nine), .ZN (n_18));
60      INV_X1 g672(.I (n_15), .ZN (n_16));
61      NAND3_X1 g673(.A1 (n_11), .A2 (n_3), .A3 (n_2), .ZN (n_15));
62      NAND2_X1 g674(.A1 (n_13), .A2 (n_11), .ZN (n_14));
63      INV_X1 g675(.I (n_12), .ZN (n_13));
64      NAND2_X1 g676(.A1 (n_10), .A2 (n_1), .ZN (n_12));
65      NOR2_X1 g677(.A1 (n_9), .A2 (five), .ZN (n_11));
66      NOR2_X1 g678(.A1 (n_6), .A2 (two), .ZN (n_10));
67      INV_X1 g679(.I (n_8), .ZN (n_9));
68      NOR3_X1 g680(.A1 (six), .A2 (four), .A3 (seven), .ZN (n_8));
69      INV_X1 g681(.I (n_6), .ZN (n_7));
70      NAND2_X1 g682(.A1 (n_5), .A2 (n_4), .ZN (n_6));
71      INV_X1 g683(.I (zero), .ZN (n_5));
72      INV_X1 g684(.I (one), .ZN (n_4));
73      INV_X1 g685(.I (eight), .ZN (n_3));
74      INV_X1 g686(.I (nine), .ZN (n_2));
75      INV_X1 g687(.I (three), .ZN (n_1));
76    endmodule
77
78  module fsm_map(clk, zero, one, two, three, four, five, six, seven, eight,
79        nine, enter, init, reset, clear, press_val, lock, state, sc, cc);
80      input clk, zero, one, two, three, four, five, six, seven, eight,
81          nine, enter, init, reset, clear;
82      output [3:0] press_val;
83      output lock;
84      output [5:0] state;
85      output [15:0] sc, cc;
86      wire clk, zero, one, two, three, four, five, six, seven, eight, nine,
87          enter, init, reset, clear;
88      wire [3:0] press_val;
89      wire lock;
90      wire [5:0] state;
91      wire [15:0] sc, cc;
92      wire [14:0] superstate;
93      wire n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8;
94      wire n_9, n_10, n_11, n_12, n_13, n_14, n_15, n_16;
```

```
 94      wire n_9, n_10, n_11, n_12, n_13, n_14, n_15, n_16;
 95      wire n_17, n_18, n_19, n_20, n_21, n_22, n_23, n_25;
 96      wire n_26, n_27, n_28, n_29, n_30, n_31, n_32, n_33;
 97      wire n_34, n_35, n_36, n_37, n_38, n_39, n_40, n_41;
 98      wire n_42, n_43, n_44, n_45, n_46, n_47, n_48, n_49;
 99      wire n_50, n_51, n_52, n_53, n_54, n_55, n_56, n_57;
100      wire n_58, n_59, n_60, n_61, n_62, n_63, n_64, n_65;
101      wire n_66, n_67, n_68, n_69, n_70, n_71, n_72, n_73;
102      wire n_74, n_75, n_76, n_77, n_78, n_79, n_80, n_81;
103      wire n_82, n_83, n_84, n_85, n_86, n_87, n_88, n_89;
104      wire n_90, n_91, n_92, n_93, n_94, n_95, n_96, n_97;
105      wire n_98, n_99, n_100, n_101, n_102, n_103, n_104, n_105;
106      wire n_106, n_107, n_108, n_109, n_110, n_111, n_112, n_113;
107      wire n_114, n_115, n_116, n_117, n_118, n_119, n_120, n_121;
108      wire n_122, n_123, n_124, n_125, n_126, n_127, n_128, n_129;
109      wire n_130, n_131, n_132, n_133, n_134, n_135, n_136, n_137;
110      wire n_138, n_139, n_140, n_141, n_142, n_143, n_144, n_145;
111      wire n_146, n_147, n_148, n_149, n_150, n_151, n_152, n_154;
112      wire n_155, n_156, n_157, n_158, n_159, n_160, n_161, n_162;
113      wire n_163, n_164, n_165, n_166, n_167, n_168, n_169, n_170;
114      wire n_171, n_172, n_173, n_175, n_176, n_177, n_178, n_179;
115      wire n_180, n_181, n_182, n_183, n_184, n_185, n_186, n_187;
116      wire n_188, n_189, n_190, n_191, n_192, n_193, n_194, n_195;
117      wire n_196, n_197, n_198, n_199, n_200, n_201, n_202, n_203;
118      wire n_204, n_206, n_207, n_208, n_209, n_210, n_211, n_212;
119      wire n_213, n_214, n_215, n_216, n_217, n_218, n_219, n_220;
120      wire n_221, n_222, n_223, n_224, n_225, n_226, n_227, n_228;
121      wire n_229, n_230, n_231, n_232, n_233, n_234, n_235, n_236;
122      wire n_237, n_238, n_239, n_240, n_241, n_242, n_243, n_244;
123      wire n_245, n_246, n_247, n_248, n_249, n_250, n_251, n_252;
124      wire n_253, n_254, n_255, n_256, n_257, n_258, n_259, n_260;
125      wire n_261, n_262, n_263, n_264, n_266, n_267, n_268, n_269;
126      wire n_270, n_271, n_272, n_273, n_274, n_275, n_277, n_279;
127      wire n_280, n_281, n_282, n_283, n_284, n_285, n_286, n_287;
128      wire n_288, n_289, n_290, n_291, n_292, n_293, n_294, n_295;
129      wire n_296, n_297, n_298, n_299, n_300, n_301, n_302, n_303;
130      wire n_304, n_305, n_306, n_307, n_308, n_309, n_310, n_311;
131      wire n_312, n_313, n_314, n_315, n_316, n_317, n_318, n_319;
132      wire n_320, n_321, n_322, n_323, n_324, n_325, n_327, n_328;
133      wire n_329, n_330, n_331, n_333, n_334, n_335, n_336, n_337;
134      wire n_338, n_339, n_340, n_341, n_342, n_343, n_344, n_345;
135      wire n_346, n_347, n_348, n_349, n_350, n_351, n_352, n_353;
136      wire n_354, n_355, n_356, n_357, n_358, n_359, n_360, n_361;
137      wire n_362, n_363, n_364, n_365, n_366, n_367, n_368, n_369;
138      wire n_370, n_371, n_372, n_373, n_374, n_375, n_376, n_377;
139      wire n_378, n_379, n_380, n_381, n_382, n_383, n_384, n_437;
```

```
140     wire n_438, n_439, n_440, n_441, n_442, num_press;
141     assign state[5] = 1'b0;
142     num_capture asdf(.zero (zero), .one (one), .two (two), .three
143          (three), .four (four), .five (five), .six (six), .seven (seven),
144          .eight (eight), .nine (nine), .clk (clk), .press_val
145          (press_val), .num_press (num_press));
146     LHQ_X1 \cc_reg[0] (.E (n_338), .D (n_290), .Q (cc[0]));
147     LHQ_X1 \cc_reg[10] (.E (n_303), .D (n_230), .Q (cc[10]));
148     LHQ_X1 \cc_reg[11] (.E (n_303), .D (n_234), .Q (cc[11]));
149     LHQ_X1 \cc_reg[12] (.E (n_302), .D (n_213), .Q (cc[12]));
150     LHQ_X1 \cc_reg[13] (.E (n_302), .D (n_212), .Q (cc[13]));
151     LHQ_X1 \cc_reg[14] (.E (n_302), .D (n_211), .Q (cc[14]));
152     LHQ_X1 \cc_reg[15] (.E (n_302), .D (n_210), .Q (cc[15]));
153     LHQ_X1 \cc_reg[1] (.E (n_338), .D (n_289), .Q (cc[1]));
154     LHQ_X1 \cc_reg[2] (.E (n_338), .D (n_288), .Q (cc[2]));
155     LHQ_X1 \cc_reg[3] (.E (n_338), .D (n_287), .Q (cc[3]));
156     LHQ_X1 \cc_reg[4] (.E (n_307), .D (n_259), .Q (cc[4]));
157     LHQ_X1 \cc_reg[5] (.E (n_307), .D (n_256), .Q (cc[5]));
158     LHQ_X1 \cc_reg[6] (.E (n_307), .D (n_254), .Q (cc[6]));
159     LHQ_X1 \cc_reg[7] (.E (n_307), .D (n_253), .Q (cc[7]));
160     LHQ_X1 \cc_reg[8] (.E (n_303), .D (n_232), .Q (cc[8]));
161     LHQ_X1 \cc_reg[9] (.E (n_303), .D (n_231), .Q (cc[9]));
162     LHQ_X1 \sc_reg[0] (.E (n_304), .D (n_215), .Q (sc[0]));
163     LHQ_X1 \sc_reg[10] (.E (n_306), .D (n_224), .Q (sc[10]));
164     LHQ_X1 \sc_reg[11] (.E (n_306), .D (n_227), .Q (sc[11]));
165     LHQ_X1 \sc_reg[12] (.E (n_305), .D (n_222), .Q (sc[12]));
166     LHQ_X1 \sc_reg[13] (.E (n_305), .D (n_217), .Q (sc[13]));
167     LHQ_X1 \sc_reg[14] (.E (n_305), .D (n_221), .Q (sc[14]));
168     LHQ_X1 \sc_reg[15] (.E (n_305), .D (n_220), .Q (sc[15]));
169     LHQ_X1 \sc_reg[1] (.E (n_304), .D (n_216), .Q (sc[1]));
170     LHQ_X1 \sc_reg[2] (.E (n_304), .D (n_219), .Q (sc[2]));
171     LHQ_X1 \sc_reg[3] (.E (n_304), .D (n_218), .Q (sc[3]));
172     LHQ_X1 \sc_reg[4] (.E (n_308), .D (n_260), .Q (sc[4]));
173     LHQ_X1 \sc_reg[5] (.E (n_308), .D (n_261), .Q (sc[5]));
174     LHQ_X1 \sc_reg[6] (.E (n_308), .D (n_258), .Q (sc[6]));
175     LHQ_X1 \sc_reg[7] (.E (n_308), .D (n_257), .Q (sc[7]));
176     LHQ_X1 \sc_reg[8] (.E (n_306), .D (n_226), .Q (sc[8]));
177     LHQ_X1 \sc_reg[9] (.E (n_306), .D (n_225), .Q (sc[9]));
178     DFFSNQ_X1 \superstate_reg[0] (.SN (1'b1), .CLK (clk), .D (n_383), .Q
179          (state[0]));
180     DFFSNQ_X1 \superstate_reg[10] (.SN (1'b1), .CLK (clk), .D (n_372), .Q
181          (superstate[10]));
182     DFFSNQ_X1 \superstate_reg[11] (.SN (1'b1), .CLK (clk), .D (n_371), .Q
183          (superstate[11]));
```

```
DFFSNQ_X1 \superstate_reg[12] (.SN (1'b1), .CLK (clk), .D (n_214), .Q
      (superstate[12]));
DFFSNQ_X1 \superstate_reg[13] (.SN (1'b1), .CLK (clk), .D (n_175), .Q
      (superstate[13]));
DFFSNQ_X1 \superstate_reg[14] (.SN (1'b1), .CLK (clk), .D (n_374), .Q
      (lock));
DFFSNQ_X1 \superstate_reg[1] (.SN (1'b1), .CLK (clk), .D (n_384), .Q
      (state[1]));
DFFSNQ_X1 \superstate_reg[2] (.SN (1'b1), .CLK (clk), .D (n_376), .Q
      (state[2]));
DFFSNQ_X1 \superstate_reg[3] (.SN (1'b1), .CLK (clk), .D (n_382), .Q
      (state[3]));
DFFSNQ_X1 \superstate_reg[4] (.SN (1'b1), .CLK (clk), .D (n_379), .Q
      (state[4]));
DFFSNQ_X1 \superstate_reg[6] (.SN (1'b1), .CLK (clk), .D (n_373), .Q
      (superstate[6]));
DFFSNQ_X1 \superstate_reg[7] (.SN (1'b1), .CLK (clk), .D (n_378), .Q
      (superstate[7]));
DFFSNQ_X1 \superstate_reg[8] (.SN (1'b1), .CLK (clk), .D (n_245), .Q
      (superstate[8]));
DFFSNQ_X1 \superstate_reg[9] (.SN (1'b1), .CLK (clk), .D (n_178), .Q
      (superstate[9]));
NAND4_X1 g47555(.A1 (n_381), .A2 (n_360), .A3 (n_366), .A4 (n_355),
      .ZN (n_384));
NAND4_X1 g47556(.A1 (n_375), .A2 (n_349), .A3 (n_380), .A4 (n_350),
      .ZN (n_383));
NOR4_X1 g47563(.A1 (n_356), .A2 (n_361), .A3 (n_357), .A4 (n_370),
      .ZN (n_382));
NOR4_X1 g47564(.A1 (n_342), .A2 (n_362), .A3 (n_329), .A4 (n_246),
      .ZN (n_381));
NOR4_X1 g47565(.A1 (n_377), .A2 (n_319), .A3 (n_245), .A4 (n_325),
      .ZN (n_380));
NAND4_X1 g47568(.A1 (n_363), .A2 (n_367), .A3 (n_320), .A4 (n_273),
      .ZN (n_379));
OR4_X1 g47569(.A1 (n_246), .A2 (n_321), .A3 (n_370), .A4 (n_340), .Z
      (n_378));
NAND4_X1 g47570(.A1 (n_369), .A2 (n_298), .A3 (n_297), .A4 (n_161),
      .ZN (n_377));
NAND4_X1 g47571(.A1 (n_368), .A2 (n_346), .A3 (n_262), .A4 (n_330),
      .ZN (n_376));
AND4_X1 g47572(.A1 (n_328), .A2 (n_334), .A3 (n_364), .A4 (n_282), .Z
      (n_375));
OAI21_X1 g47573(.A1 (n_310), .A2 (init), .B (n_365), .ZN (n_374));
OR4_X1 g47574(.A1 (n_370), .A2 (n_340), .A3 (n_243), .A4 (init), .Z
      (n_373));
NAND3_X1 g47575(.A1 (n_349), .A2 (n_352), .A3 (n_250), .ZN (n_372));
```

```
230     NAND3_X1 g47576(.A1 (n_349), .A2 (n_315), .A3 (n_352), .ZN (n_371));
231     NOR4_X1 g47577(.A1 (n_348), .A2 (n_339), .A3 (n_237), .A4 (n_318),
232          .ZN (n_370));
233     AOI21_X1 g47578(.A1 (n_96), .A2 (n_237), .B (n_362), .ZN (n_369));
234     NOR4_X1 g47579(.A1 (n_345), .A2 (n_336), .A3 (n_319), .A4 (n_358),
235          .ZN (n_368));
236     NOR4_X1 g47580(.A1 (n_354), .A2 (n_245), .A3 (n_191), .A4 (n_285),
237          .ZN (n_367));
238     NOR2_X1 g47581(.A1 (n_359), .A2 (n_347), .ZN (n_366));
239     NOR4_X1 g47582(.A1 (n_353), .A2 (n_342), .A3 (n_323), .A4 (n_314),
240          .ZN (n_365));
241     NOR3_X1 g47583(.A1 (n_285), .A2 (n_331), .A3 (n_440), .ZN (n_364));
242     NOR4_X1 g47584(.A1 (n_335), .A2 (n_286), .A3 (n_251), .A4 (n_235),
243          .ZN (n_363));
244     NOR3_X1 g47585(.A1 (n_316), .A2 (n_1), .A3 (n_170), .ZN (n_362));
245     NAND3_X1 g47586(.A1 (n_337), .A2 (n_344), .A3 (n_262), .ZN (n_361));
246     NOR4_X1 g47587(.A1 (n_442), .A2 (n_270), .A3 (n_199), .A4 (n_247),
247          .ZN (n_360));
248     NAND4_X1 g47588(.A1 (n_333), .A2 (n_295), .A3 (n_299), .A4 (n_274),
249          .ZN (n_359));
250     NAND4_X1 g47589(.A1 (n_343), .A2 (n_292), .A3 (n_284), .A4 (n_317),
251          .ZN (n_358));
252     NAND4_X1 g47590(.A1 (n_266), .A2 (n_322), .A3 (n_300), .A4 (n_330),
253          .ZN (n_357));
254     NAND4_X1 g47591(.A1 (n_320), .A2 (n_327), .A3 (n_309), .A4 (n_311),
255          .ZN (n_356));
256     AOI21_X1 g47592(.A1 (n_314), .A2 (clear), .B (n_351), .ZN (n_355));
257     OR4_X1 g47593(.A1 (n_269), .A2 (n_312), .A3 (n_214), .A4 (n_246), .Z
258          (n_354));
259     OR2_X1 g47594(.A1 (n_442), .A2 (n_191), .Z (n_353));
260     NOR2_X1 g47595(.A1 (n_228), .A2 (n_331), .ZN (n_352));
261     NAND2_X1 g47596(.A1 (n_309), .A2 (n_324), .ZN (n_351));
262     AOI21_X1 g47597(.A1 (n_280), .A2 (reset), .B (n_329), .ZN (n_350));
263     AOI21_X1 g47598(.A1 (n_229), .A2 (enter), .B (n_336), .ZN (n_349));
264     NAND4_X1 g47599(.A1 (n_313), .A2 (n_283), .A3 (n_239), .A4 (n_194),
265          .ZN (n_348));
266     NOR2_X1 g47600(.A1 (n_310), .A2 (n_25), .ZN (n_347));
267     NOR4_X1 g47601(.A1 (n_301), .A2 (n_270), .A3 (n_228), .A4 (n_229),
268          .ZN (n_346));
269     NAND4_X1 g47602(.A1 (n_328), .A2 (n_176), .A3 (n_282), .A4 (n_223),
270          .ZN (n_345));
271     NOR4_X1 g47603(.A1 (n_191), .A2 (n_296), .A3 (n_167), .A4 (init), .ZN
272          (n_344));
273     NOR4_X1 g47604(.A1 (n_191), .A2 (n_162), .A3 (n_296), .A4 (n_294),
274          .ZN (n_343));
```

```
275   INV_X1 g47605(.I (n_341), .ZN (n_342));
276   AOI22_X1 g47606(.A1 (n_264), .A2 (n_134), .B1 (n_96), .B2 (n_169),
277       .ZN (n_341));
278   OAI22_X1 g47607(.A1 (n_271), .A2 (n_6), .B1 (n_187), .B2 (n_25), .ZN
279       (n_340));
280   NAND4_X1 g47608(.A1 (n_197), .A2 (n_275), .A3 (n_252), .A4 (n_133),
281       .ZN (n_339));
282   INV_X1 g47613(.I (n_336), .ZN (n_337));
283   NOR2_X1 g47614(.A1 (n_263), .A2 (n_95), .ZN (n_336));
284   INV_X1 g47615(.I (n_334), .ZN (n_335));
285   NOR2_X1 g47616(.A1 (n_267), .A2 (n_255), .ZN (n_334));
286   NOR2_X1 g47617(.A1 (n_291), .A2 (n_294), .ZN (n_333));
287   INV_X1 g47619(.I (n_331), .ZN (n_330));
288   AOI21_X1 g47620(.A1 (n_181), .A2 (n_248), .B (n_25), .ZN (n_331));
289   NOR2_X1 g47621(.A1 (n_126), .A2 (n_292), .ZN (n_329));
290   NOR2_X1 g47622(.A1 (n_268), .A2 (n_291), .ZN (n_328));
291   NOR2_X1 g47623(.A1 (n_269), .A2 (n_279), .ZN (n_327));
292   OAI21_X1 g47625(.A1 (n_95), .A2 (n_197), .B (n_299), .ZN (n_325));
293   INV_X1 g47626(.I (n_323), .ZN (n_324));
294   OAI21_X1 g47627(.A1 (n_157), .A2 (n_101), .B (n_281), .ZN (n_323));
295   NOR3_X1 g47628(.A1 (n_286), .A2 (n_255), .A3 (n_235), .ZN (n_322));
296   OAI21_X1 g47629(.A1 (num_press), .A2 (n_242), .B (n_5), .ZN (n_321));
297   OR4_X1 g47630(.A1 (n_147), .A2 (n_1), .A3 (n_281), .A4 (reset), .Z
298       (n_320));
299   NOR4_X1 g47631(.A1 (n_148), .A2 (n_1), .A3 (n_281), .A4 (reset), .ZN
300       (n_319));
301   NAND4_X1 g47632(.A1 (n_206), .A2 (n_168), .A3 (n_159), .A4 (n_166),
302       .ZN (n_318));
303   OAI21_X1 g47633(.A1 (n_238), .A2 (n_169), .B (n_96), .ZN (n_317));
304   OAI21_X1 g47634(.A1 (n_4), .A2 (cc[6]), .B (n_438), .ZN (n_316));
305   NOR2_X1 g47635(.A1 (n_207), .A2 (n_296), .ZN (n_315));
306   OAI21_X1 g47636(.A1 (n_239), .A2 (init), .B (n_203), .ZN (n_314));
307   NOR4_X1 g47637(.A1 (n_272), .A2 (n_240), .A3 (n_188), .A4 (n_179),
308       .ZN (n_313));
309   OAI21_X1 g47638(.A1 (n_1), .A2 (n_250), .B (n_176), .ZN (n_312));
310   AOI21_X1 g47639(.A1 (n_142), .A2 (clear), .B (n_293), .ZN (n_311));
311   NOR4_X1 g47640(.A1 (n_277), .A2 (n_240), .A3 (n_193), .A4 (n_140),
312       .ZN (n_310));
313   AOI22_X1 g47641(.A1 (n_244), .A2 (n_5), .B1 (n_96), .B2 (n_142), .ZN
314       (n_309));
315   NOR2_X1 g47672(.A1 (n_200), .A2 (n_196), .ZN (n_301));
316   NOR2_X1 g47673(.A1 (n_251), .A2 (n_228), .ZN (n_300));
317   NAND2_X1 g47674(.A1 (n_31), .A2 (n_202), .ZN (n_299));
318   NAND2_X1 g47675(.A1 (n_200), .A2 (n_195), .ZN (n_298));
319   NOR2_X1 g47676(.A1 (n_204), .A2 (n_249), .ZN (n_297));
320   NOR2_X1 g47677(.A1 (num_press), .A2 (n_250), .ZN (n_296));
321   NAND2_X1 g47678(.A1 (n_1), .A2 (n_251), .ZN (n_295));
322   NOR2_X1 g47679(.A1 (n_201), .A2 (init), .ZN (n_294));
```

```
323     NAND2_X1 g47680(.A1 (n_184), .A2 (n_198), .ZN (n_338));
324     NAND2_X1 g47681(.A1 (n_201), .A2 (n_128), .ZN (n_293));
325     NAND2_X1 g47682(.A1 (n_23), .A2 (n_202), .ZN (n_292));
326     NOR2_X1 g47683(.A1 (n_97), .A2 (n_233), .ZN (n_291));
327     NOR2_X1 g47684(.A1 (n_7), .A2 (n_198), .ZN (n_290));
328     NOR2_X1 g47685(.A1 (n_2), .A2 (n_198), .ZN (n_289));
329     NOR2_X1 g47686(.A1 (n_4), .A2 (n_198), .ZN (n_288));
330     NOR2_X1 g47687(.A1 (n_3), .A2 (n_198), .ZN (n_287));
331     NOR2_X1 g47688(.A1 (n_95), .A2 (n_233), .ZN (n_286));
332     INV_X1 g47689(.I (n_284), .ZN (n_285));
333     NAND2_X1 g47690(.A1 (num_press), .A2 (n_243), .ZN (n_284));
334     AND2_X1 g47691(.A1 (n_233), .A2 (n_209), .Z (n_283));
335     NAND2_X1 g47692(.A1 (n_96), .A2 (n_240), .ZN (n_282));
336     INV_X1 g47693(.I (n_280), .ZN (n_281));
337     NOR2_X1 g47694(.A1 (n_209), .A2 (init), .ZN (n_280));
338     OAI21_X1 g47695(.A1 (n_137), .A2 (n_114), .B (n_252), .ZN (n_279));
339     NAND3_X1 g47697(.A1 (n_197), .A2 (n_194), .A3 (n_233), .ZN (n_277));
340     NOR3_X1 g47699(.A1 (n_140), .A2 (n_244), .A3 (n_156), .ZN (n_275));
341     NAND2_X1 g47700(.A1 (n_236), .A2 (n_28), .ZN (n_274));
342     OAI21_X1 g47701(.A1 (n_180), .A2 (n_142), .B (n_94), .ZN (n_273));
343     NAND4_X1 g47702(.A1 (n_186), .A2 (n_192), .A3 (n_130), .A4 (n_128),
344          .ZN (n_272));
345     NOR3_X1 g47703(.A1 (n_185), .A2 (n_142), .A3 (n_167), .ZN (n_271));
346     NOR3_X1 g47704(.A1 (n_145), .A2 (n_95), .A3 (n_241), .ZN (n_270));
347     NOR3_X1 g47705(.A1 (n_146), .A2 (n_95), .A3 (n_239), .ZN (n_269));
348     NOR2_X1 g47706(.A1 (n_208), .A2 (n_239), .ZN (n_268));
349     INV_X1 g47707(.I (n_266), .ZN (n_267));
350     NAND3_X1 g47708(.A1 (n_145), .A2 (n_94), .A3 (n_240), .ZN (n_266));
351     OAI21_X1 g47710(.A1 (n_149), .A2 (n_95), .B (n_25), .ZN (n_264));
352     AOI22_X1 g47711(.A1 (n_149), .A2 (n_134), .B1 (n_151), .B2 (n_169),
353          .ZN (n_263));
354     AOI22_X1 g47712(.A1 (n_96), .A2 (n_180), .B1 (n_31), .B2 (n_160), .ZN
355          (n_262));
356     NOR2_X1 g47715(.A1 (n_2), .A2 (n_189), .ZN (n_261));
357     NOR2_X1 g47716(.A1 (n_7), .A2 (n_189), .ZN (n_260));
358     NOR2_X1 g47717(.A1 (n_7), .A2 (n_190), .ZN (n_259));
359     NOR2_X1 g47718(.A1 (n_4), .A2 (n_189), .ZN (n_258));
360     NOR2_X1 g47719(.A1 (n_3), .A2 (n_189), .ZN (n_257));
361     NOR2_X1 g47720(.A1 (n_2), .A2 (n_190), .ZN (n_256));
362     NOR2_X1 g47721(.A1 (n_97), .A2 (n_194), .ZN (n_255));
363     NOR2_X1 g47722(.A1 (n_4), .A2 (n_190), .ZN (n_254));
364     NOR2_X1 g47723(.A1 (n_3), .A2 (n_190), .ZN (n_253));
365     NOR2_X1 g47724(.A1 (n_154), .A2 (n_164), .ZN (n_252));
366     NAND2_X1 g47725(.A1 (n_170), .A2 (n_196), .ZN (n_251));
367     INV_X1 g47726(.I (n_250), .ZN (n_249));
368     NAND2_X1 g47727(.A1 (n_165), .A2 (n_5), .ZN (n_250));
369     AND2_X1 g47728(.A1 (n_130), .A2 (n_157), .Z (n_248));
370     NOR2_X1 g47729(.A1 (n_161), .A2 (clear), .ZN (n_247));
```

```
371     NOR2_X1 g47730(.A1 (n_95), .A2 (n_187), .ZN (n_246));
372     NOR2_X1 g47731(.A1 (n_95), .A2 (n_186), .ZN (n_245));
373     NAND2_X1 g47732(.A1 (n_152), .A2 (n_155), .ZN (n_244));
374     INV_X1 g47733(.I (n_242), .ZN (n_243));
375     NAND2_X1 g47734(.A1 (n_164), .A2 (n_5), .ZN (n_242));
376     INV_X1 g47735(.I (n_240), .ZN (n_241));
377     NOR2_X1 g47736(.A1 (n_182), .A2 (n_90), .ZN (n_240));
378     INV_X1 g47737(.I (n_238), .ZN (n_239));
379     NOR2_X1 g47738(.A1 (n_182), .A2 (n_103), .ZN (n_238));
380     NAND2_X1 g47739(.A1 (n_187), .A2 (n_141), .ZN (n_237));
381     NAND2_X1 g47740(.A1 (n_181), .A2 (n_186), .ZN (n_236));
382     NAND2_X1 g47741(.A1 (n_184), .A2 (n_171), .ZN (n_303));
383     NOR2_X1 g47742(.A1 (n_197), .A2 (n_27), .ZN (n_235));
384     NOR2_X1 g47743(.A1 (n_3), .A2 (n_171), .ZN (n_234));
385     NAND2_X1 g47744(.A1 (n_183), .A2 (n_104), .ZN (n_233));
386     NOR2_X1 g47745(.A1 (n_7), .A2 (n_171), .ZN (n_232));
387     NOR2_X1 g47746(.A1 (n_2), .A2 (n_171), .ZN (n_231));
388     NOR2_X1 g47747(.A1 (n_4), .A2 (n_171), .ZN (n_230));
389     AND2_X1 g47748(.A1 (n_167), .A2 (n_6), .Z (n_229));
390     NOR2_X1 g47749(.A1 (n_161), .A2 (n_6), .ZN (n_228));
391     NOR2_X1 g47750(.A1 (n_3), .A2 (n_158), .ZN (n_227));
392     NOR2_X1 g47751(.A1 (n_7), .A2 (n_158), .ZN (n_226));
393     NOR2_X1 g47752(.A1 (n_2), .A2 (n_158), .ZN (n_225));
394     NOR2_X1 g47753(.A1 (n_4), .A2 (n_158), .ZN (n_224));
395     NAND2_X1 g47754(.A1 (n_154), .A2 (n_5), .ZN (n_223));
396     NOR2_X1 g47755(.A1 (n_7), .A2 (n_173), .ZN (n_222));
397     NAND2_X1 g47756(.A1 (n_177), .A2 (n_173), .ZN (n_305));
398     NOR2_X1 g47757(.A1 (n_4), .A2 (n_173), .ZN (n_221));
399     NOR2_X1 g47758(.A1 (n_3), .A2 (n_173), .ZN (n_220));
400     NOR2_X1 g47759(.A1 (n_4), .A2 (n_163), .ZN (n_219));
401     NOR2_X1 g47760(.A1 (n_3), .A2 (n_163), .ZN (n_218));
402     NOR2_X1 g47761(.A1 (n_2), .A2 (n_173), .ZN (n_217));
403     NOR2_X1 g47762(.A1 (n_2), .A2 (n_163), .ZN (n_216));
404     NAND2_X1 g47763(.A1 (n_177), .A2 (n_163), .ZN (n_304));
405     NOR2_X1 g47764(.A1 (n_7), .A2 (n_163), .ZN (n_215));
406     NAND2_X1 g47765(.A1 (n_184), .A2 (n_172), .ZN (n_302));
407     AND2_X1 g47766(.A1 (n_23), .A2 (n_160), .Z (n_214));
408     NAND2_X1 g47767(.A1 (n_177), .A2 (n_158), .ZN (n_306));
409     NOR2_X1 g47768(.A1 (n_7), .A2 (n_172), .ZN (n_213));
410     NOR2_X1 g47769(.A1 (n_2), .A2 (n_172), .ZN (n_212));
411     NOR2_X1 g47770(.A1 (n_4), .A2 (n_172), .ZN (n_211));
412     NOR2_X1 g47771(.A1 (n_3), .A2 (n_172), .ZN (n_210));
413     NAND2_X1 g47772(.A1 (n_183), .A2 (n_93), .ZN (n_209));
414     NAND2_X1 g47773(.A1 (n_146), .A2 (n_94), .ZN (n_208));
415     NOR2_X1 g47774(.A1 (n_95), .A2 (n_181), .ZN (n_207));
416     AOI21_X1 g47775(.A1 (n_138), .A2 (n_131), .B (n_144), .ZN (n_206));
417     NOR3_X1 g47777(.A1 (n_157), .A2 (n_27), .A3 (enter), .ZN (n_204));
418     INV_X1 g47778(.I (n_202), .ZN (n_203));
```

```
419     NOR3_X1 g47779(.A1 (n_125), .A2 (n_143), .A3 (init), .ZN (n_202));
420     AOI21_X1 g47780(.A1 (n_135), .A2 (n_121), .B (n_144), .ZN (n_201));
421     NAND4_X1 g47781(.A1 (n_124), .A2 (n_43), .A3 (n_64), .A4 (num_press),
422          .ZN (n_200));
423     NOR3_X1 g47782(.A1 (n_1), .A2 (n_194), .A3 (init), .ZN (n_199));
424     NAND4_X1 g47783(.A1 (n_120), .A2 (n_35), .A3 (n_15), .A4
425          (superstate[10]), .ZN (n_198));
426     NAND2_X1 g47784(.A1 (n_139), .A2 (n_100), .ZN (n_197));
427     INV_X1 g47785(.I (n_196), .ZN (n_195));
428     NAND2_X1 g47786(.A1 (n_140), .A2 (n_28), .ZN (n_196));
429     NAND2_X1 g47787(.A1 (n_139), .A2 (n_87), .ZN (n_194));
430     INV_X1 g47788(.I (n_192), .ZN (n_193));
431     NAND2_X1 g47789(.A1 (n_139), .A2 (n_93), .ZN (n_192));
432     NOR2_X1 g47790(.A1 (n_97), .A2 (n_133), .ZN (n_191));
433     NAND2_X1 g47791(.A1 (n_307), .A2 (n_22), .ZN (n_190));
434     NAND2_X1 g47792(.A1 (n_308), .A2 (n_13), .ZN (n_189));
435     NOR2_X1 g47793(.A1 (n_125), .A2 (n_143), .ZN (n_188));
436     NAND2_X1 g47794(.A1 (n_138), .A2 (n_87), .ZN (n_187));
437     INV_X1 g47795(.I (n_186), .ZN (n_185));
438     NAND2_X1 g47796(.A1 (n_138), .A2 (n_93), .ZN (n_186));
439     NAND2_X1 g47797(.A1 (n_307), .A2 (superstate[10]), .ZN (n_184));
440     INV_X1 g47798(.I (n_183), .ZN (n_182));
441     NOR2_X1 g47799(.A1 (n_143), .A2 (n_108), .ZN (n_183));
442     INV_X1 g47800(.I (n_181), .ZN (n_180));
443     NAND2_X1 g47801(.A1 (n_138), .A2 (n_89), .ZN (n_181));
444     NOR3_X1 g47802(.A1 (n_132), .A2 (n_85), .A3 (n_112), .ZN (n_179));
445     NOR2_X1 g47803(.A1 (n_95), .A2 (n_141), .ZN (n_178));
446     NAND2_X1 g47804(.A1 (n_308), .A2 (superstate[6]), .ZN (n_177));
447     INV_X1 g47805(.I (n_175), .ZN (n_176));
448     NOR2_X1 g47806(.A1 (n_95), .A2 (n_130), .ZN (n_175));
449     NAND3_X1 g47808(.A1 (n_115), .A2 (n_82), .A3 (n_13), .ZN (n_173));
450     NAND3_X1 g47809(.A1 (n_120), .A2 (n_79), .A3 (n_22), .ZN (n_172));
451     NAND3_X1 g47810(.A1 (n_120), .A2 (n_91), .A3 (superstate[12]), .ZN
452          (n_171));
453     NAND3_X1 g47811(.A1 (n_139), .A2 (n_93), .A3 (n_28), .ZN (n_170));
454     INV_X1 g47812(.I (n_169), .ZN (n_168));
455     NOR3_X1 g47813(.A1 (n_143), .A2 (n_108), .A3 (n_88), .ZN (n_169));
456     NOR3_X1 g47814(.A1 (n_137), .A2 (n_103), .A3 (init), .ZN (n_167));
457     INV_X1 g47815(.I (n_165), .ZN (n_166));
458     NOR4_X1 g47816(.A1 (n_90), .A2 (n_113), .A3 (n_80), .A4 (n_105), .ZN
459          (n_165));
460     NOR4_X1 g47817(.A1 (n_129), .A2 (n_99), .A3 (n_37), .A4 (state[3]),
461          .ZN (n_164));
462     NAND3_X1 g47818(.A1 (n_115), .A2 (n_106), .A3 (superstate[6]), .ZN
463          (n_163));
464     NOR3_X1 g47819(.A1 (n_95), .A2 (n_137), .A3 (n_114), .ZN (n_162));
465     INV_X1 g47820(.I (n_161), .ZN (n_160));
466     NAND3_X1 g47821(.A1 (n_138), .A2 (n_104), .A3 (n_5), .ZN (n_161));
```

```
467   NAND4_X1 g47822(.A1 (n_135), .A2 (n_98), .A3 (n_35), .A4
468        (superstate[12]), .ZN (n_159));
469   NAND3_X1 g47823(.A1 (n_115), .A2 (n_109), .A3 (n_13), .ZN (n_158));
470   INV_X1 g47824(.I (n_157), .ZN (n_156));
471   NAND4_X1 g47825(.A1 (n_107), .A2 (n_87), .A3 (n_54), .A4 (n_78), .ZN
472        (n_157));
473   NAND4_X1 g47826(.A1 (n_116), .A2 (n_89), .A3 (n_86), .A4 (n_81), .ZN
474        (n_155));
475   AND4_X1 g47827(.A1 (n_116), .A2 (n_100), .A3 (n_84), .A4 (n_82), .Z
476        (n_154));
477   NAND4_X1 g47829(.A1 (n_116), .A2 (n_86), .A3 (n_104), .A4 (n_109),
478        .ZN (n_152));
479   INV_X1 g47830(.I (n_150), .ZN (n_151));
480   NAND4_X1 g47831(.A1 (n_74), .A2 (n_47), .A3 (n_75), .A4 (n_42), .ZN
481        (n_150));
482   NOR4_X1 g47832(.A1 (n_72), .A2 (n_70), .A3 (n_59), .A4 (n_46), .ZN
483        (n_149));
484   INV_X1 g47833(.I (n_147), .ZN (n_148));
485   NAND4_X1 g47834(.A1 (n_68), .A2 (n_58), .A3 (n_69), .A4 (n_48), .ZN
486        (n_147));
487   NOR4_X1 g47835(.A1 (n_62), .A2 (n_65), .A3 (n_45), .A4 (n_50), .ZN
488        (n_146));
489   NAND4_X1 g47836(.A1 (n_49), .A2 (n_77), .A3 (n_57), .A4 (n_66), .ZN
490        (n_145));
491   NOR4_X1 g47837(.A1 (n_122), .A2 (n_85), .A3 (n_105), .A4 (n_22), .ZN
492        (n_144));
493   NOR2_X1 g47838(.A1 (n_119), .A2 (n_80), .ZN (n_307));
494   NAND2_X1 g47839(.A1 (n_111), .A2 (n_35), .ZN (n_143));
495   INV_X1 g47840(.I (n_142), .ZN (n_141));
496   NOR2_X1 g47841(.A1 (n_125), .A2 (n_123), .ZN (n_142));
497   NOR2_X1 g47842(.A1 (n_125), .A2 (n_110), .ZN (n_140));
498   NOR2_X1 g47843(.A1 (n_110), .A2 (n_108), .ZN (n_139));
499   AND2_X1 g47844(.A1 (n_115), .A2 (n_81), .Z (n_308));
500   INV_X1 g47845(.I (n_138), .ZN (n_137));
501   NOR2_X1 g47846(.A1 (n_123), .A2 (n_108), .ZN (n_138));
502   NAND3_X1 g47847(.A1 (n_39), .A2 (n_41), .A3 (n_44), .ZN (n_136));
503   NOR3_X1 g47848(.A1 (n_108), .A2 (n_85), .A3 (lock), .ZN (n_135));
504   INV_X1 g47849(.I (n_134), .ZN (n_133));
505   NOR3_X1 g47850(.A1 (n_110), .A2 (n_103), .A3 (n_108), .ZN (n_134));
506   AOI21_X1 g47851(.A1 (n_102), .A2 (n_79), .B (n_121), .ZN (n_132));
507   OR3_X1 g47852(.A1 (n_102), .A2 (n_89), .A3 (n_104), .Z (n_131));
508   NAND3_X1 g47853(.A1 (n_118), .A2 (n_100), .A3 (n_78), .ZN (n_130));
509   NAND4_X1 g47854(.A1 (n_91), .A2 (n_81), .A3 (n_16), .A4
510        (superstate[6]), .ZN (n_129));
511   INV_X1 g47855(.I (n_127), .ZN (n_128));
512   NOR4_X1 g47856(.A1 (n_117), .A2 (n_90), .A3 (n_26), .A4 (n_13), .ZN
513        (n_127));
514   NOR4_X1 g47857(.A1 (n_56), .A2 (n_55), .A3 (n_52), .A4 (n_51), .ZN
```

```
Auto Hide
51     (n_126));
516    NAND2_X1 g47858(.A1 (n_107), .A2 (n_98), .ZN (n_125));
517    NOR2_X1 g47859(.A1 (n_63), .A2 (n_60), .ZN (n_124));
518    NAND2_X1 g47860(.A1 (n_54), .A2 (n_83), .ZN (n_123));
519    NAND2_X1 g47861(.A1 (n_87), .A2 (n_54), .ZN (n_122));
520    NOR2_X1 g47862(.A1 (n_92), .A2 (n_80), .ZN (n_121));
521    INV_X1 g47863(.I (n_119), .ZN (n_120));
522    NAND2_X1 g47864(.A1 (n_106), .A2 (n_13), .ZN (n_119));
523    INV_X1 g47865(.I (n_117), .ZN (n_118));
524    NAND2_X1 g47866(.A1 (n_107), .A2 (n_54), .ZN (n_117));
525    NOR2_X1 g47867(.A1 (n_53), .A2 (superstate[10]), .ZN (n_116));
526    AND2_X1 g47868(.A1 (n_91), .A2 (n_15), .Z (n_115));
527    NOR2_X1 g47869(.A1 (n_93), .A2 (n_87), .ZN (n_114));
528    NAND3_X1 g47870(.A1 (n_83), .A2 (n_16), .A3 (superstate[10]), .ZN
529        (n_113));
530    NAND3_X1 g47871(.A1 (n_36), .A2 (n_32), .A3 (n_16), .ZN (n_112));
531    NOR4_X1 g47872(.A1 (n_37), .A2 (superstate[6]), .A3 (n_16), .A4
532        (n_17), .ZN (n_111));
533    NAND4_X1 g47873(.A1 (n_86), .A2 (n_35), .A3 (n_15), .A4 (lock), .ZN
534        (n_110));
535    AND2_X1 g47874(.A1 (n_36), .A2 (superstate[8]), .Z (n_109));
536    INV_X1 g47875(.I (n_108), .ZN (n_107));
537    NAND2_X1 g47876(.A1 (n_32), .A2 (n_36), .ZN (n_108));
538    INV_X1 g47877(.I (n_105), .ZN (n_106));
539    NAND2_X1 g47878(.A1 (n_36), .A2 (n_14), .ZN (n_105));
540    NOR2_X1 g47879(.A1 (n_38), .A2 (n_12), .ZN (n_104));
541    INV_X1 g47880(.I (n_103), .ZN (n_102));
542    NAND2_X1 g47881(.A1 (n_33), .A2 (state[2]), .ZN (n_103));
543    NAND2_X1 g47882(.A1 (n_28), .A2 (enter), .ZN (n_101));
544    INV_X1 g47883(.I (n_99), .ZN (n_100));
545    NAND2_X1 g47884(.A1 (n_33), .A2 (n_12), .ZN (n_99));
546    NOR2_X1 g47885(.A1 (n_38), .A2 (state[2]), .ZN (n_98));
547    INV_X1 g47886(.I (n_97), .ZN (n_96));
548    NAND2_X1 g47887(.A1 (n_31), .A2 (n_5), .ZN (n_97));
549    INV_X1 g47888(.I (n_95), .ZN (n_94));
550    NAND2_X1 g47889(.A1 (num_press), .A2 (n_28), .ZN (n_95));
551    INV_X1 g47890(.I (n_93), .ZN (n_92));
552    NOR2_X1 g47891(.A1 (n_30), .A2 (state[2]), .ZN (n_93));
553    NOR2_X1 g47892(.A1 (n_34), .A2 (superstate[10]), .ZN (n_91));
554    NAND2_X1 g47893(.A1 (n_29), .A2 (state[2]), .ZN (n_90));
555    INV_X1 g47894(.I (n_89), .ZN (n_88));
556    NOR2_X1 g47895(.A1 (n_30), .A2 (n_12), .ZN (n_89));
557    AND2_X1 g47896(.A1 (n_29), .A2 (n_12), .Z (n_87));
558    NOR2_X1 g47897(.A1 (n_26), .A2 (superstate[6]), .ZN (n_86));
559    INV_X1 g47898(.I (n_85), .ZN (n_84));
560    NAND3_X1 g47899(.A1 (n_13), .A2 (state[4]), .A3 (state[3]), .ZN
561        (n_85));
562    NOR3_X1 g47900(.A1 (superstate[6]), .A2 (state[3]), .A3 (state[4]),
```

```
563       .ZN (n_83));
564   NOR3_X1 g47901(.A1 (n_19), .A2 (superstate[7]), .A3 (superstate[8]),
565       .ZN (n_82));
566   AND3_X2 g47902(.A1 (n_14), .A2 (n_19), .A3 (superstate[7]), .Z
567       (n_81));
568   NAND3_X1 g47903(.A1 (n_20), .A2 (n_15), .A3 (superstate[11]), .ZN
569       (n_80));
570   NOR3_X1 g47904(.A1 (n_20), .A2 (superstate[11]), .A3
571       (superstate[12]), .ZN (n_79));
572   NOR3_X1 g47905(.A1 (n_17), .A2 (superstate[6]), .A3 (state[4]), .ZN
573       (n_78));
574   INV_X1 g47906(.I (n_76), .ZN (n_77));
575   OAI22_X1 g47907(.A1 (n_2), .A2 (sc[9]), .B1 (n_3), .B2 (sc[11]), .ZN
576       (n_76));
577   AOI22_X1 g47908(.A1 (n_2), .A2 (sc[13]), .B1 (n_3), .B2 (sc[15]), .ZN
578       (n_75));
579   INV_X1 g47909(.I (n_73), .ZN (n_74));
580   OAI22_X1 g47910(.A1 (n_3), .A2 (sc[15]), .B1 (n_2), .B2 (sc[13]), .ZN
581       (n_73));
582   INV_X1 g47911(.I (n_71), .ZN (n_72));
583   AOI22_X1 g47912(.A1 (n_2), .A2 (cc[13]), .B1 (n_3), .B2 (cc[15]), .ZN
584       (n_71));
585   OAI22_X1 g47913(.A1 (n_3), .A2 (cc[15]), .B1 (n_4), .B2 (cc[14]), .ZN
586       (n_70));
587   AOI22_X1 g47914(.A1 (n_2), .A2 (cc[1]), .B1 (n_4), .B2 (cc[2]), .ZN
588       (n_69));
589   INV_X1 g47915(.I (n_67), .ZN (n_68));
590   OAI22_X1 g47916(.A1 (n_3), .A2 (cc[3]), .B1 (n_4), .B2 (cc[2]), .ZN
591       (n_67));
592   AOI22_X1 g47917(.A1 (n_7), .A2 (sc[8]), .B1 (n_3), .B2 (sc[11]), .ZN
593       (n_66));
594   OAI22_X1 g47918(.A1 (n_3), .A2 (sc[7]), .B1 (n_4), .B2 (sc[6]), .ZN
595       (n_65));
596   AOI22_X1 g47919(.A1 (n_7), .A2 (cc[8]), .B1 (n_3), .B2 (cc[11]), .ZN
597       (n_64));
598   OAI22_X1 g47920(.A1 (n_2), .A2 (cc[9]), .B1 (n_3), .B2 (cc[11]), .ZN
599       (n_63));
600   INV_X1 g47921(.I (n_61), .ZN (n_62));
601   AOI22_X1 g47922(.A1 (n_4), .A2 (sc[6]), .B1 (n_3), .B2 (sc[7]), .ZN
602       (n_61));
603   OAI22_X1 g47923(.A1 (press_val[1]), .A2 (n_10), .B1 (n_7), .B2
604       (cc[8]), .ZN (n_60));
605   OAI22_X1 g47924(.A1 (press_val[2]), .A2 (n_11), .B1 (n_2), .B2
606       (cc[13]), .ZN (n_59));
607   AOI22_X1 g47925(.A1 (n_3), .A2 (cc[3]), .B1 (press_val[1]), .B2
608       (n_8), .ZN (n_58));
609   AOI22_X1 g47926(.A1 (n_2), .A2 (sc[9]), .B1 (press_val[0]), .B2
610       (n_9), .ZN (n_57));
```

```
611     XNOR2_X1 g47927(.A1 (n_3), .A2 (sc[3]), .ZN (n_56));
612     XNOR2_X1 g47928(.A1 (n_4), .A2 (sc[2]), .ZN (n_55));
613     INV_X1 g47929(.I (n_54), .ZN (n_53));
614     NOR4_X1 g47930(.A1 (superstate[13]), .A2 (superstate[11]), .A3
615          (lock), .A4 (superstate[12]), .ZN (n_54));
616     XNOR2_X1 g47931(.A1 (n_2), .A2 (sc[1]), .ZN (n_52));
617     XNOR2_X1 g47932(.A1 (n_7), .A2 (sc[0]), .ZN (n_51));
618     XNOR2_X1 g47933(.A1 (n_7), .A2 (sc[4]), .ZN (n_50));
619     XNOR2_X1 g47934(.A1 (press_val[2]), .A2 (sc[10]), .ZN (n_49));
620     XNOR2_X1 g47935(.A1 (press_val[0]), .A2 (cc[0]), .ZN (n_48));
621     XNOR2_X1 g47936(.A1 (press_val[2]), .A2 (sc[14]), .ZN (n_47));
622     XNOR2_X1 g47937(.A1 (n_7), .A2 (cc[12]), .ZN (n_46));
623     XNOR2_X1 g47938(.A1 (n_2), .A2 (sc[5]), .ZN (n_45));
624     XNOR2_X1 g47939(.A1 (press_val[0]), .A2 (cc[4]), .ZN (n_44));
625     XNOR2_X1 g47940(.A1 (press_val[2]), .A2 (cc[10]), .ZN (n_43));
626     XNOR2_X1 g47941(.A1 (press_val[0]), .A2 (sc[12]), .ZN (n_42));
627     NAND2_X1 g47942(.A1 (n_3), .A2 (cc[7]), .ZN (n_41));
628     NOR2_X1 g47943(.A1 (n_3), .A2 (cc[7]), .ZN (n_40));
629     NAND2_X1 g47944(.A1 (n_4), .A2 (cc[6]), .ZN (n_39));
630     NAND2_X1 g47945(.A1 (state[0]), .A2 (state[1]), .ZN (n_38));
631     NAND2_X1 g47946(.A1 (n_21), .A2 (n_15), .ZN (n_37));
632     NOR2_X1 g47947(.A1 (superstate[9]), .A2 (superstate[7]), .ZN (n_36));
633     INV_X1 g47948(.I (n_35), .ZN (n_34));
634     NOR2_X1 g47949(.A1 (superstate[13]), .A2 (superstate[11]), .ZN
635          (n_35));
636     NOR2_X1 g47950(.A1 (state[1]), .A2 (state[0]), .ZN (n_33));
637     NOR2_X1 g47951(.A1 (superstate[8]), .A2 (superstate[10]), .ZN (n_32));
638     NOR2_X1 g47952(.A1 (num_press), .A2 (clear), .ZN (n_31));
639     NAND2_X1 g47953(.A1 (n_18), .A2 (state[1]), .ZN (n_30));
640     NOR2_X1 g47954(.A1 (n_18), .A2 (state[1]), .ZN (n_29));
641     INV_X1 g47955(.I (n_28), .ZN (n_27));
642     NOR2_X1 g47956(.A1 (init), .A2 (clear), .ZN (n_28));
643     NAND2_X1 g47957(.A1 (n_17), .A2 (state[4]), .ZN (n_26));
644     NAND2_X1 g47959(.A1 (n_5), .A2 (clear), .ZN (n_25));
645     NOR2_X1 g47960(.A1 (n_1), .A2 (clear), .ZN (n_23));
646     INV_X1 g47961(.I (superstate[10]), .ZN (n_22));
647     INV_X1 g47962(.I (state[4]), .ZN (n_21));
648     INV_X1 g47963(.I (superstate[13]), .ZN (n_20));
649     INV_X1 g47964(.I (superstate[9]), .ZN (n_19));
650     INV_X1 g47965(.I (state[0]), .ZN (n_18));
651     INV_X1 g47966(.I (state[3]), .ZN (n_17));
652     INV_X1 g47967(.I (lock), .ZN (n_16));
653     INV_X1 g47968(.I (superstate[12]), .ZN (n_15));
654     INV_X1 g47969(.I (superstate[8]), .ZN (n_14));
655     INV_X1 g47970(.I (superstate[6]), .ZN (n_13));
656     INV_X1 g47971(.I (state[2]), .ZN (n_12));
657     INV_X1 g47972(.I (cc[14]), .ZN (n_11));
658     INV_X1 g47973(.I (cc[9]), .ZN (n_10));
```

```
659       INV_X1 g47974(.I (sc[8]), .ZN (n_9));
660       INV_X1 g47975(.I (cc[1]), .ZN (n_8));
661       INV_X1 g47976(.I (press_val[0]), .ZN (n_7));
662       INV_X1 g47977(.I (clear), .ZN (n_6));
663       INV_X1 g47978(.I (init), .ZN (n_5));
664       INV_X1 g47979(.I (press_val[2]), .ZN (n_4));
665       INV_X1 g47980(.I (press_val[3]), .ZN (n_3));
666       INV_X1 g47981(.I (press_val[1]), .ZN (n_2));
667       INV_X1 g47982(.I (num_press), .ZN (n_1));
668       NOR3_X1 g2(.A1 (n_437), .A2 (n_136), .A3 (n_40), .ZN (n_438));
669       XNOR2_X1 g3(.A1 (n_2), .A2 (cc[5]), .ZN (n_437));
670       AOI21_X1 g47983(.A1 (n_439), .A2 (n_152), .B (init), .ZN (n_440));
671       NOR2_X1 g47984(.A1 (n_127), .A2 (n_179), .ZN (n_439));
672       AOI21_X1 g47985(.A1 (n_441), .A2 (n_25), .B (n_168), .ZN (n_442));
673       NAND2_X1 g47986(.A1 (n_150), .A2 (n_28), .ZN (n_441));
674     endmodule
```

In num_capture module, DFF was only used for the peak detector of press signals and the mapped verilog matched that design.

And as expected, the fsm was built with combinational logic except the state register, meaning that there are no inferred latches or other problematic design.

Visual Waveforms from Mapped Verilog

For each figure, it will contain waveforms for behavior verilog (up) and mapped verilog (down).

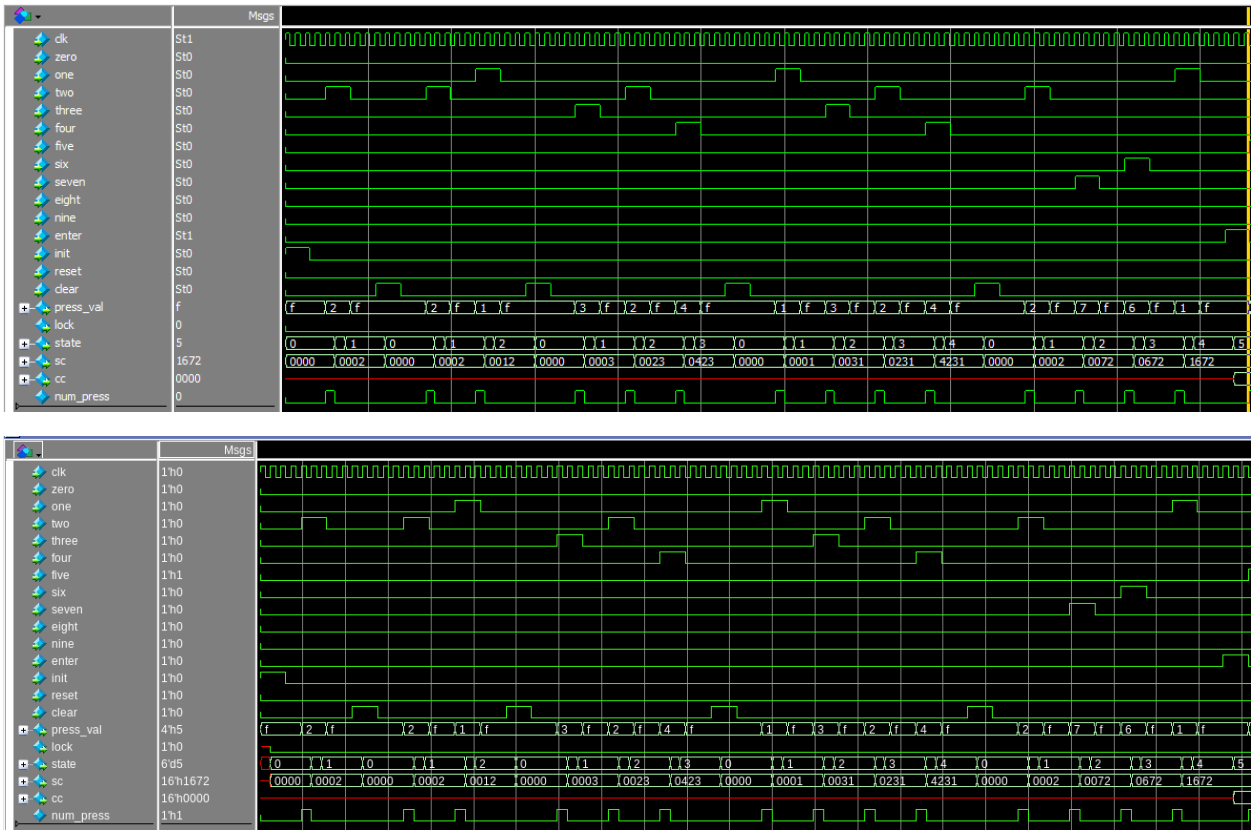



Figure 1. Waveform of “Testing security code states (S0-S4)” in TB

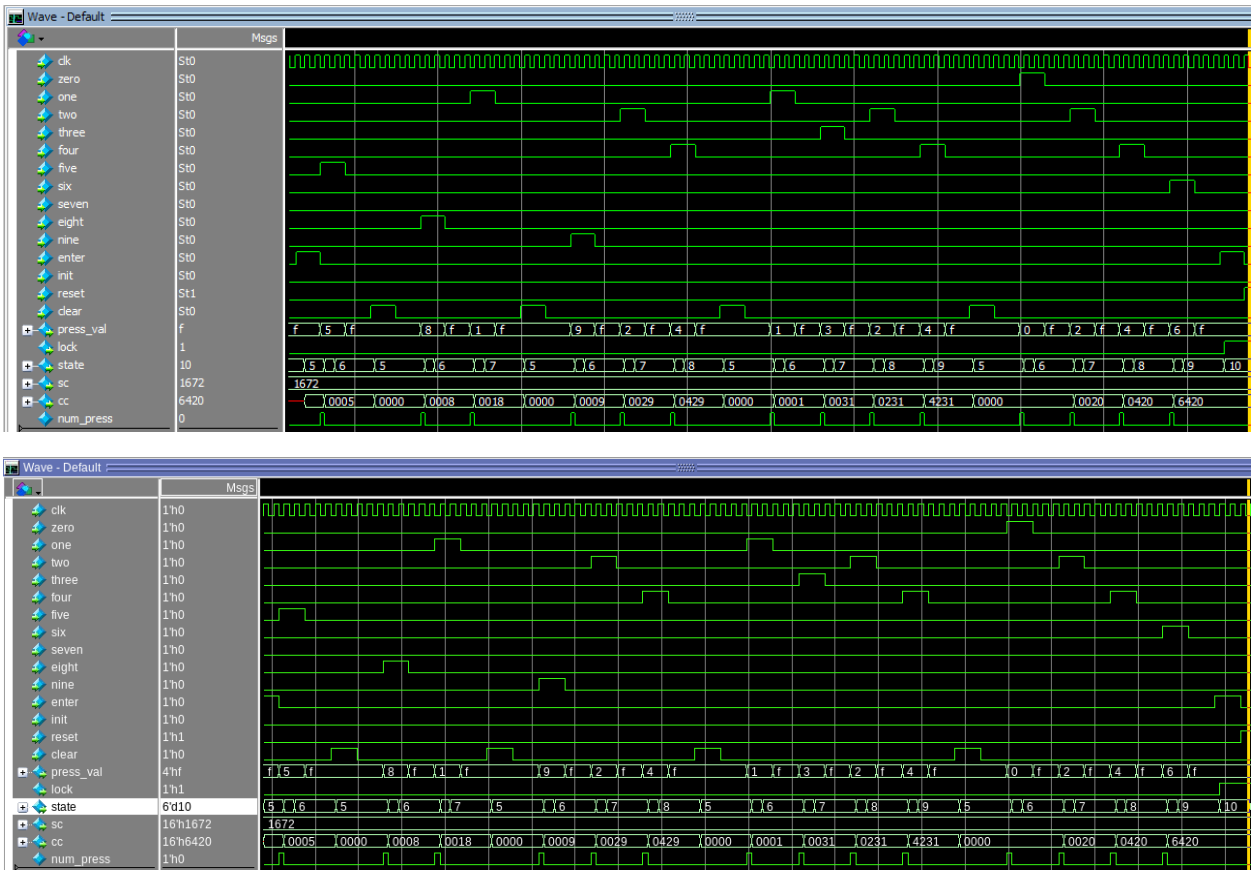



Figure 2. Waveform of “Testing customer code states (S5-S9)” in TB

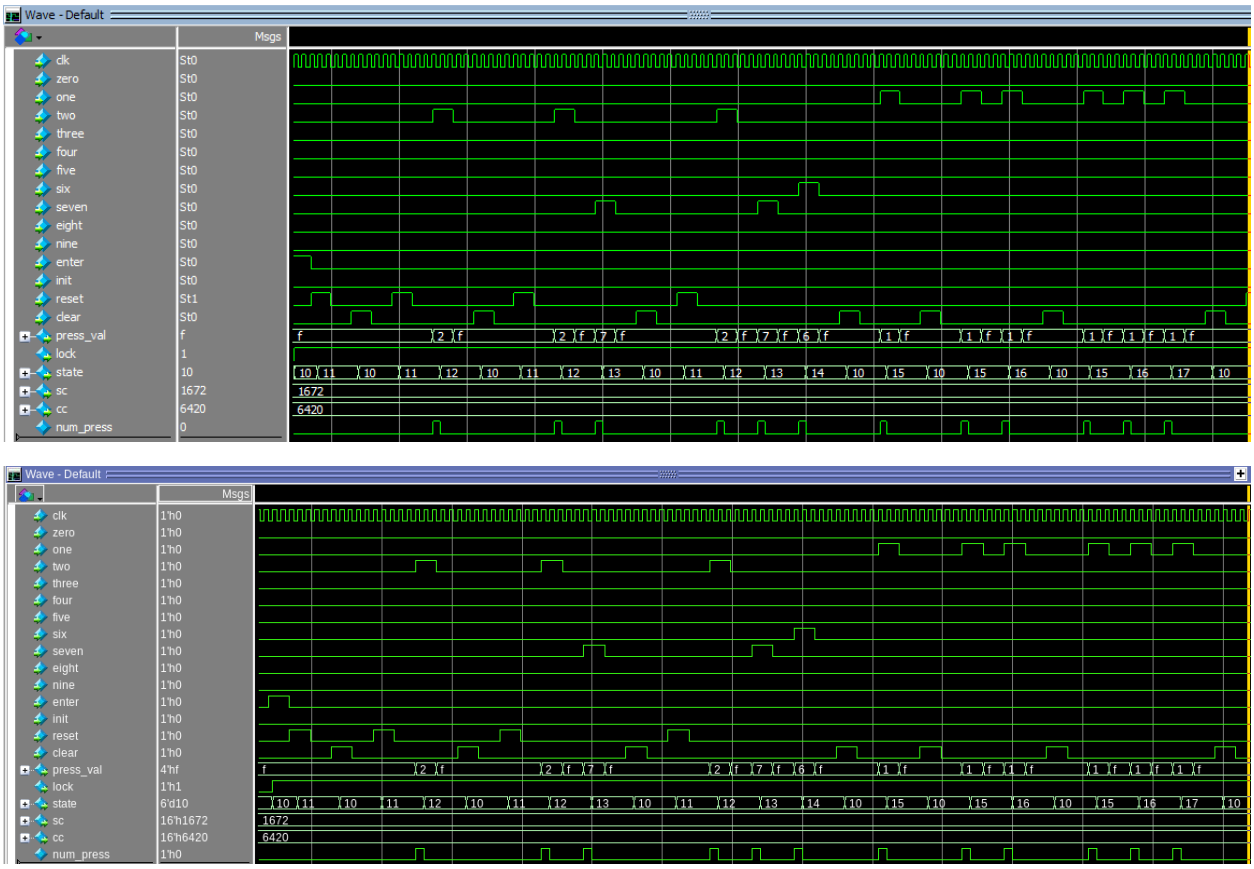



Figure 3. Waveform of “Testing resetting/counterfeit states (S10-S17)”, “Clear tests” in TB

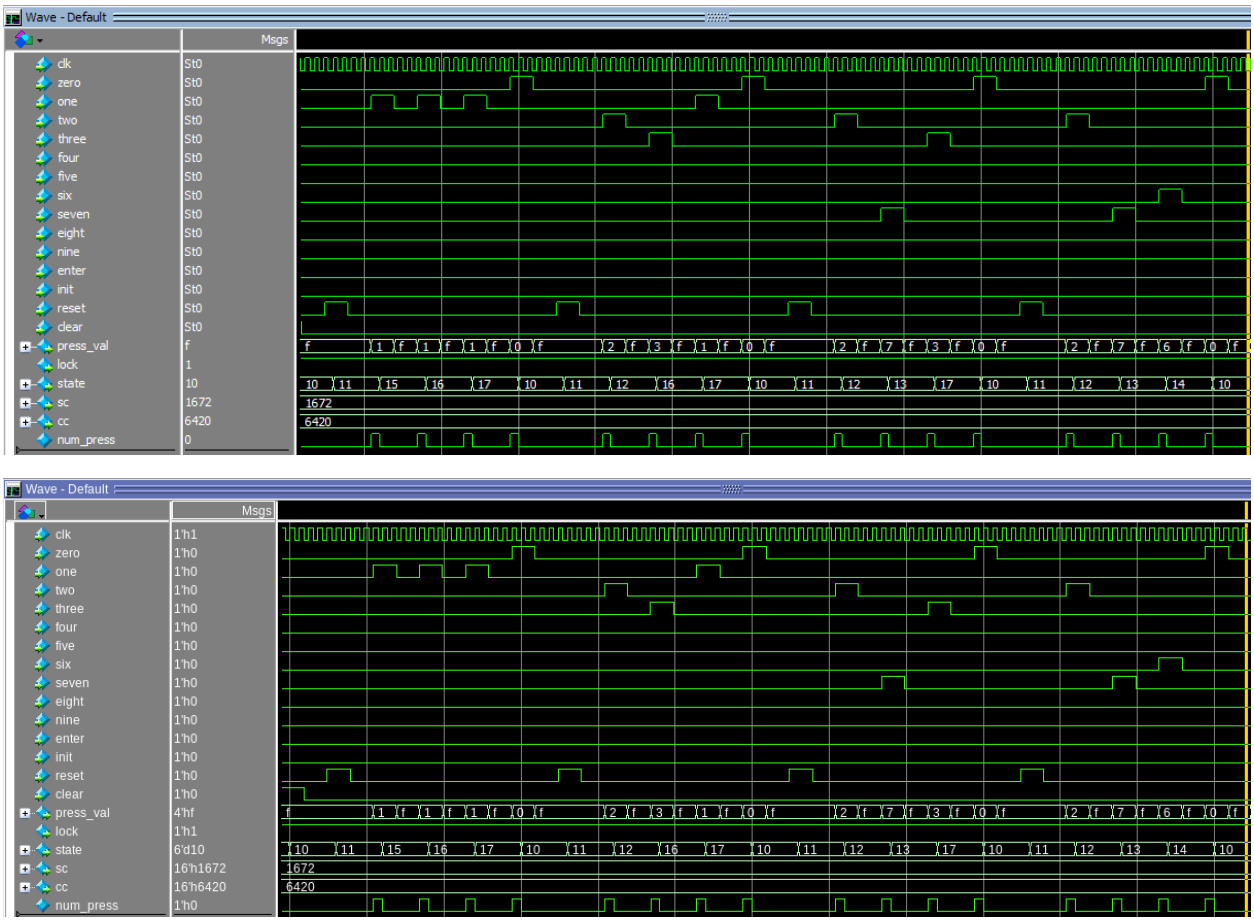



Figure 4. Waveform of “Testing resetting/counterfeit states (S10-S17)”, “Security code fail tests” in TB

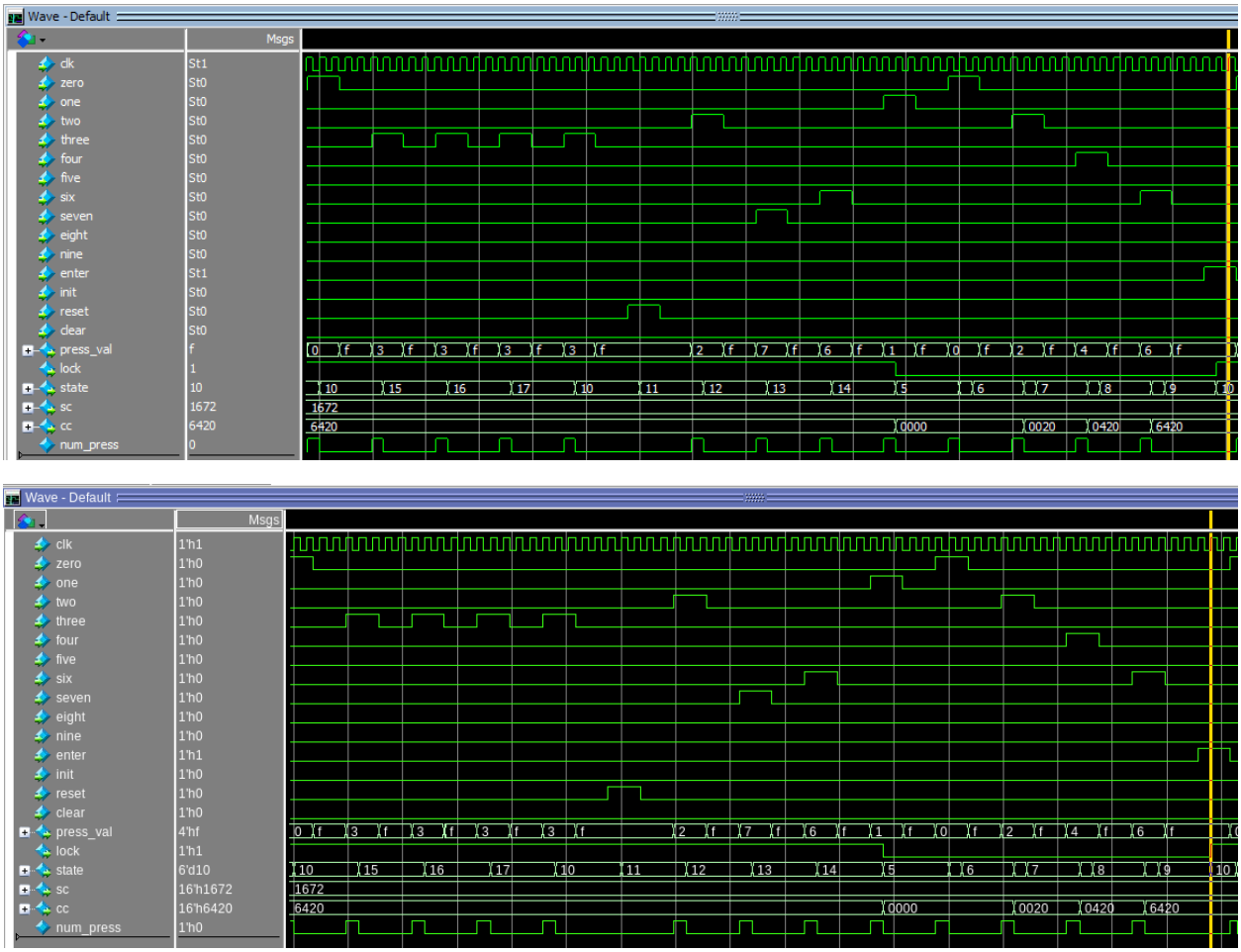



Figure 5. Waveform of “Testing resetting/counterfeit states (S10-S17)”, “Fully counterfeit test” + “Reset test” + “Back to S10” with same cc” in TB


Figure 6. Waveform of “Testing unlocking states”, “Clear tests” + “Customer code fail tests” in TB

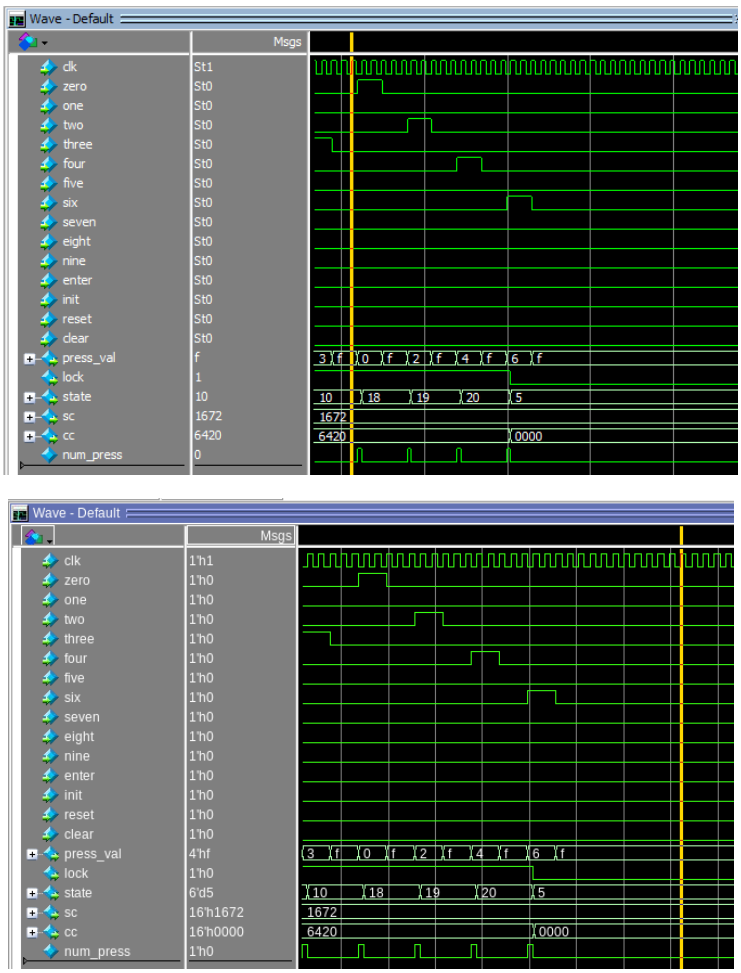



Figure 7. Waveform of “Testing unlocking states”, “Unlock test” in TB

The difference between the two waveforms isn’t very clear when we try to look at its state transitions.

I zoomed in a specific part of the waveform just to show the how the waveform of mapped verilog is different from behavior verilog. (Figure 8)

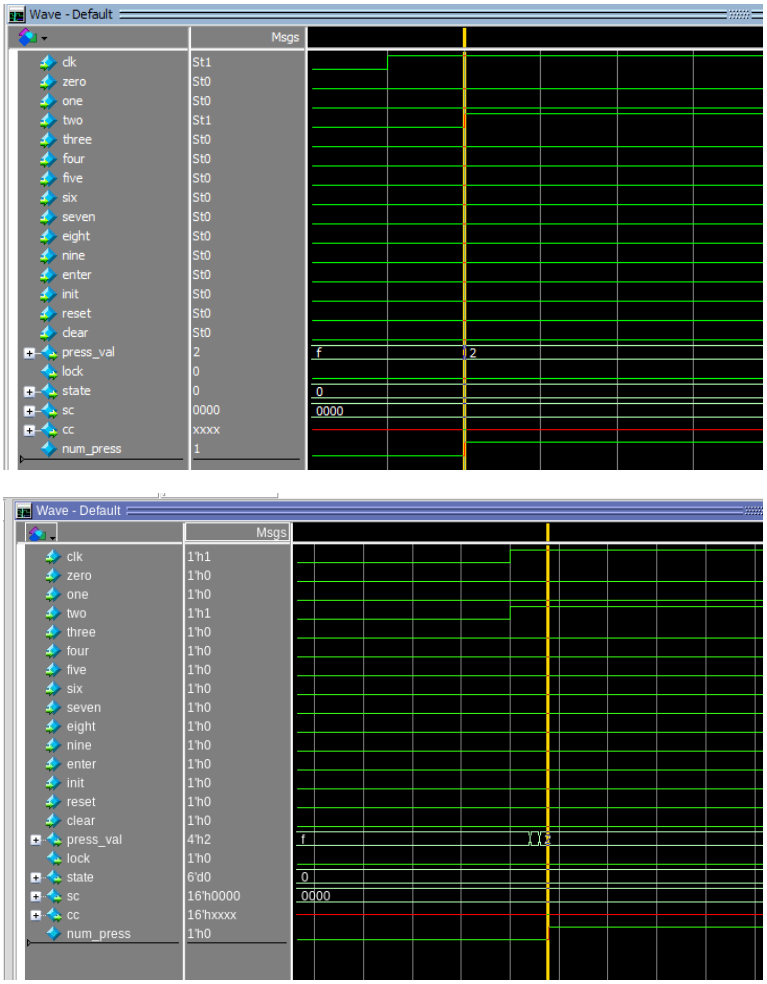



Figure 8. Showcase of difference between two waveforms

In the behavior verilog waveform, we see that “num_press” rises at the same time as “two”, but in the mapped verilog waveform it has a delay caused by combinational logic.

The other difference is press_val. Press_val is the result of combination logic from inputs zero ~ nine. We can see that when it transitions from f to 2, there is an incorrect value in between, this is probably because the combinational logic hasn’t fully processed yet, or in other words, the bus isn’t stable yet.

These are the two most significant difference I found between the waveforms. It also caused errors in my original design which is the reason why I modified my behavioral verilog code. This tells us that when we write HDL code, it is very important to think about its hardware, or else the design will fail after synthesis.

Compile Outputs

```
╟============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
  Generated on:           Oct 06 2021  11:02:03 pm
  Module:                 fsm
  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
--------------------------------------------------------------------
fsm         477        146         0         146    <none> (D)
  asdf       50         12         0          12    <none> (D)

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

Figure 9. Number of Cells

```
╟============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
  Generated on:           Oct 06 2021  11:02:03 pm
  Module:                 fsm
  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            8    2.359    NanGate_15nm_OCL
AND3_X2            2    0.786    NanGate_15nm_OCL
AND4_X1            3    1.327    NanGate_15nm_OCL
AOI21_X1          12    3.539    NanGate_15nm_OCL
AOI22_X1          12    4.129    NanGate_15nm_OCL
DFFRNQ_X1          1    1.278    NanGate_15nm_OCL
DFFSNQ_X1         14   17.891    NanGate_15nm_OCL
INV_X1            83   12.239    NanGate_15nm_OCL
LHQ_X1            32   22.020    NanGate_15nm_OCL
NAND2_X1          68   13.369    NanGate_15nm_OCL
NAND3_X1          23    6.783    NanGate_15nm_OCL
NAND4_X1          28    9.634    NanGate_15nm_OCL
NOR2_X1          100   19.661    NanGate_15nm_OCL
NOR3_X1           26    7.668    NanGate_15nm_OCL
NOR4_X1           25    8.602    NanGate_15nm_OCL
OAI21_X1          11    3.244    NanGate_15nm_OCL
OAI22_X1           9    3.097    NanGate_15nm_OCL
OR2_X1             1    0.295    NanGate_15nm_OCL
OR3_X1             1    0.393    NanGate_15nm_OCL
OR4_X1             4    1.769    NanGate_15nm_OCL
XNOR2_X1          14    6.193    NanGate_15nm_OCL
-------------------------------------------------
total            477  146.276


   Type    Instances   Area  Area %
-----------------------------------
sequential        47  41.189   28.2
inverter          83  12.239    8.4
logic            347  92.848   63.5
-----------------------------------
total            477 146.276  100.0
```

Figure 10. Area/Gates

```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
  Generated on:           Oct 06 2021  11:02:04 pm
  Module:                 fsm
  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)
------------------------------------------------
fsm        477   119.446 226302.584 226422.030
  asdf      50    10.004  12643.756  12653.760
```

Figure 11. Power

```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1
  Generated on:           Oct 06 2021  11:02:03 pm
  Module:                 fsm
  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
superstate_reg[12]/CLK                                 0       0 R
superstate_reg[12]/Q     DFFSNQ_X1       5  4.2    7   +16      16 F
g47930/A4                                             +0      16
g47930/ZN                NOR4_X1         5  5.0   28   +18      35 R
g47860/A1                                             +0      35
g47860/ZN                NAND2_X1        2  1.7   10    +8      43 F
g47846/A1                                             +0      43
g47846/ZN                NOR2_X1         6  6.0   18   +13      56 R
g47796/A1                                             +0      56
g47796/ZN                NAND2_X1        4  3.4   11   +10      65 F
g47702/A1                                             +0      65
g47702/ZN                NAND4_X1        1  1.1    5    +5      70 R
g47637/A1                                             +0      70
g47637/ZN                NOR4_X1         1  0.9    5    +3      73 F
g47599/A1                                             +0      73
g47599/ZN                NAND4_X1        1  1.1    6    +4      77 R
g47577/A1                                             +0      77
g47577/ZN                NOR4_X1         3  2.6    6    +5      82 F
g47569/A3                                             +0      82
g47569/Z                 OR4_X1          1  0.5    3   +10      91 F
superstate_reg[7]/D      DFFSNQ_X1                     +0      91
superstate_reg[7]/CLK    setup                         0    +9    100 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)              capture                              120 R
-------------------------------------------------------------------
Cost Group   : 'clk' (path_group 'clk')
Timing slack :      20ps
Start-point  : superstate_reg[12]/CLK
End-point    : superstate_reg[7]/D
```

Figure 12. Timing

Clock period was set to 120ns because with 100ns clock period my design was getting 0ps timing slack. Testbench was also using a 120ns period clock to match our synthesis.

The clock period we synthesize with basically tells us the highest clock frequency our circuit can run at. For my case it would be 10MHz because I’m getting 0ps timing slack at clock period 100ns.

If we use a library with gates smaller than 15nm, the power and area would decrease but the cell count will be the same since it is determined by our verilog logic.

# Notes

- Mapped Verilog file fsm_map.v , updated num_capture.sv and updated fsm.sv are submitted along with this report.
- This is re-submission, please ignore my first and second submission.