Objective

The purpose of this lab is to further our knowledge of Verilog and circuit design by walking the student through the process of designing a combination lock. The lock is very similiar to the locks found on high school lockers. After building and designing a 3 digit lock, the student will then have to implement a 4-digit lock by modifying the existing design.

Design

Experiment 1

The first part of **Experiment 1** consisted of simulating the implemented design of the Finite State Machine (FSM) given below.

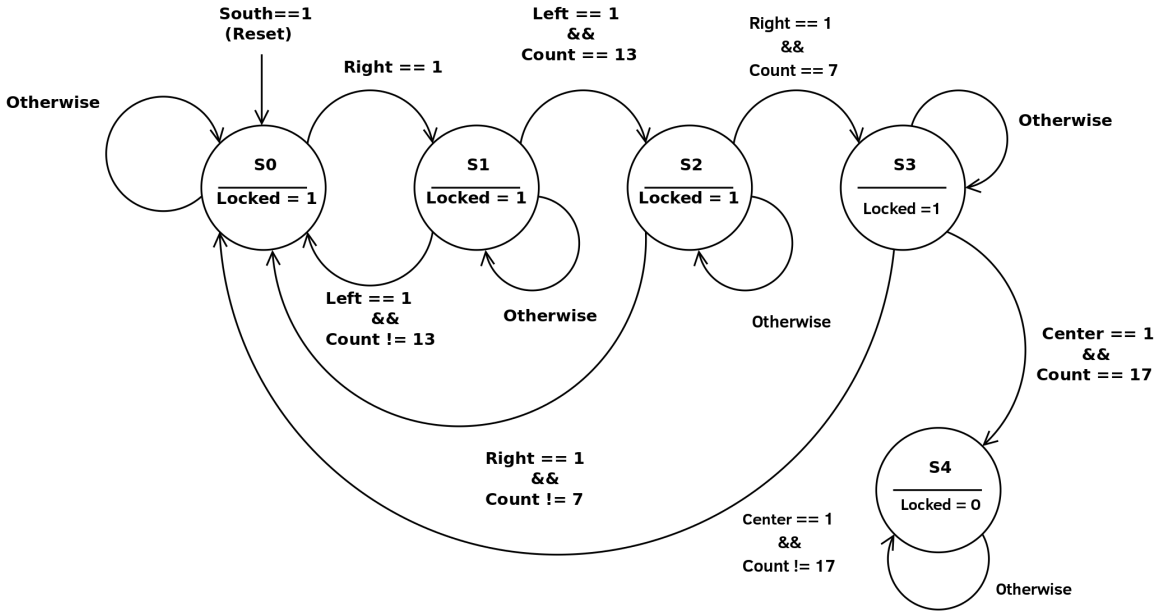

Figure 1: *Rotary Combination-Lock State Diagram*

Below is the Verilog code for the aforementioned Rotary Combination-Lock state diagram that was tested against the appropriate test bench.

Code Block 1: Combination Lock FSM

```
1 `timescale 1ns / 1ps
```

```
2
3  module combination_lock_fsm(output reg [2:0] state,
4    output wire Locked, // asserted when locked
5    input wire Right, Left, // indicate direction
6    input wire [4:0] Count, // indicate position
7    input wire Center, // the unlock button
8    input wire Clk, South // clock and reset
9    );
10
11   parameter S0 = 3'b000,
12             S1 = 3'b001,
13             S2 = 3'b010,
14             S3 = 3'b011,
15                          S4 = 3'b100,
16
17   reg [2:0] nextState;
18
19   always @ ( * ) begin
20     case (state)
21       S0: begin
22         if (Right)
23                                    nextState = S1;
24         else
25           nextState = S0;
26         end
27       S1: begin
28         if (Left)
29                                    if(Count == 5'b01101)
30                                            nextState = S2;
31                                    else
32                                            nextState = S0;
```

```
33                          else
34                                  nextState = S1;
35          end
36        S2: begin
37          if (Right)
38                                  if (Count == 5'b00111)
39                                          nextState = S3;
40                                  else
41                                          nextState = S0;
42                          else
43                                  nextState = S2;
44          end
45        S3: begin
46                          if (Left)
47                                  if (Count == 5'b10001)
48                                          nextState = S4;
49                                  else
50                                          nextState = S0;
51                          else
52                                  nextState = S3;
53          end
54                  S4: begin
55          nextState = S4;
56          end
57
58                  default: begin
59                          nextState = S0;
60                  end
61      endcase
62    end
63
```

```
64          assign Locked = (state == S4) ? 0:1;
65
66          always@ (posedge Clk)
67                  if (South)
68                          state <= S0;
69                  else
70                          state <= nextState;
71  endmodule // combination_lock_fsm
```

For the 2nd part of **Experiment 1**, the top level module was designed. The top level module is a 0-to-19 Up/Down Counter. This module keeps track of the position of the rotary knob located on the FBGA board. The module below was designed with behavioral Verilog and then tested against the appropriate test bench.

Code Block 2: Up/Down Counter

```
1  `timescale 1ns / 1ps
2
3  module up_down_counter(
4    output reg [4:0] Count,
5    input wire Up, Down,
6    input wire Clk, South
7    );
8
9    always @ (posedge Clk) begin
10     if (South)
11       Count <= 0;
12     else if (Up)
13       begin
14       if (Count == 19)
15         Count <= 0;
16       else
17         Count <= Count +1;
```

```
18          end
19      else if (Down)
20                  if (Count == 0)
21                          Count <= 19;
22                  else
23                          Count <= Count - 1;
24          end
25
26  endmodule // up_down_counter
```

Experiment 2

In the next part of the lab, the previously simulated modules were integrated with a given rotary encoder and LCD driver modules into a top-level module. The rotary combination lock module was set as the top level module. The other modules are as follows: "rotary_combination_lock.ucf" (the UCF for the top-level module), "synchronizer.v" (the synchronizer module for the asynchronous inputs), "lcd_driver.v" (the driver module for the character LCD screen), and "rotary_encoder_module.v" (the quadrature decoding module). These modules are below.

Code Block 3: Rotary Combination Lock Top Level Module

```
1  /*This is the top-level module for our digital *
2   *rotary combination-lock based on the diagram *
3   *provide in the lab manual                    */
4
5  module rotary_combination_lock(
6      /*LCD interface wires make up our output!*/
7      output wire LCD_E, LCD_RW, LCD_RS,
8      output wire [3:0] SF_D,
9      /*Let's output state for debugging!*/
10     output wire [2:0] J1,
11     input Clk,
12     /*the buttons and rotary encoder outputs*
```

```
13          *provide input to our top-level circuit*/
14      input Center,
15      input South,
16      input wire rotA, rotB
17  );
18
19      /*intermediate nets*/
20      wire CenterSync, SouthSync;
21      wire Right, Left;
22      wire Locked;
23      wire [4:0] Count;
24
25      /*synchronize button inputs*/
26      synchronizer syncA(CenterSync, Center, Clk);
27      synchronizer syncB(SouthSync, South, Clk);
28
29      /*wire up rotary encoder module*/
30      rotary_encoder_module U0(
31              .Left(Left),
32              .Right(Right),
33              .Clk(Clk),
34              .rotA(rotA),
35              .rotB(rotB)
36      );
37
38      /*wire up combination lock FSM*/
39      combination_lock_fsm U1(
40              .Locked(Locked),
41              .Right(Right),
42              .state(J1),
43              .Left(Left),
```

```
44              .Center(CenterSync),
45              .Clk(Clk),
46              .South(SouthSync),
47              .Count(Count)
48      );
49
50      /*instantiate up down counter*/
51      up_down_counter U2(
52              .Count(Count),
53              .Up (Left),
54              .Down(Right),
55              .Clk(Clk),
56              .South(South)
57      );
58
59      /*hook up LCD driver*/
60      lcd_driver U3(Clk, South, Count, Locked, SF_D, LCD_E, LCD_RS, LCD_RW);
61
62  endmodule
```

Code Block 4: Rotary Combination Lock UCF File

```
1  #push buttons
2  NET "South"  LOC = "K17"  | IOSTANDARD = LVTTL | PULLDOWN ;
3  NET "Center" LOC = "V16"  | IOSTANDARD = LVTTL | PULLDOWN ;
4
5  #rotary encoder
6  NET "rotA" LOC = "K18" | IOSTANDARD = LVTTL | PULLUP ;
7  NET "rotB" LOC = "G18" | IOSTANDARD = LVTTL | PULLUP ;
8
9  #J1 connector for debugging!
10 NET "J1<0>" LOC = "B4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
```

```
11 NET "J1<1>" LOC = "A4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
12 NET "J1<2>" LOC = "D5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
13
14 #Clock
15 NET "Clk" LOC = "C9" ;
16 NET "Clk" PERIOD = 20.0ns HIGH 40%;
17
18 #LCD interface signals
19 NET "LCD_E" LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
20 NET "LCD_RS" LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW
21 NET "LCD_RW" LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW
22 # The LCD four-bit data interface is shared with the StrataFlash.
23 NET "SF_D[0]" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW
24 NET "SF_D[1]" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW
25 NET "SF_D[2]" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW
26 NET "SF_D[3]" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW
```

Code Block 5: Synchronizer Module

```
1  /*This module provides the synchronization   *
2   *necessary to prevent metastability when    *
3   *transitioning from an asynchronous to a    *
4   *synchronous domain. In other words, when   *
5   *we bring an input signal in from the FPGA  *
6   *board into a clocked domain, we must do    *
7   *this buffering!                            */
8
9  module synchronizer(
10       output wire OutSignal,
11       input wire InSignal,
12       input wire Clk
13 );
```

```
14
15      /*intermediate nets*/
16      reg buff0, buff1, buff2;
17
18      always@(posedge Clk)
19              begin
20                  buff0 <= InSignal;
21                  buff1 <= buff0;
22                  buff2  <= buff1;
23              end
24
25      assign OutSignal = buff2;
26
27  endmodule
```

Code Block 6: LCD Driver Module

```
1  `timescale 1ns / 1ps
2  /*this module generates the interface signal necessary to*
3   *communicate with the character LCD on the Spartan 3e   *
4   *This code was modified from earlier semesters of 248   */
5
6  module lcd_driver(clk, reset, Position, Locked, SF_D, LCD_E, LCD_RS, LCD_RW
7  input [4:0] Position; //position of rotary knob
8  input Locked; //output of rotary combination FSM
9  input clk, reset;
10
11 /*needed at end of code to construct display output*/
12 reg [63:0] status_ascii; //8 characters for LOCKED or UNLOCKED
13 reg [15:0] count_ascii; //2 characters for decimal value of count
14
15 /*This is shitty code but I don't have time to fix it right now....*/
```

```
16 output [3:0] SF_D;
17 reg [3:0] SF_D=4'd0;
18 output LCD_E, LCD_RS, LCD_RW;
19 reg LCD_E=1'b0;
20 reg LCD_RS=1'b0;
21 reg LCD_RW=1'b0;
22 reg [20:0] count=21'd0;
23 reg [7:0] d=8'd0;
24 wire [255:0] display_data;
25
26 //Sequential Logic to generate Timing
27 always @ (posedge clk)
28 begin
29 case (reset)
30 1'b1:
31 begin
32         count <= count + 21'd1;
33         if ((count == 21'd750000) && (d==8'd0))
34         begin
35                 d <= d + 8'd1;
36                 count <= 21'd0;
37                 SF_D <= 4'b0011;
38                 LCD_E <= 1'b1;
39         end
40         if ((count == 21'd20) && (d==8'd1))
41         begin
42                 d <= d + 8'd1;
43                 count <= 21'd0;
44                 SF_D <= 4'b0000;
45                 LCD_E <= 1'b0;
46         end
```

```
47          if ((count == 21'd205000) && (d==8'd2))
48          begin
49                  d <= d + 8'd1;
50                  count <= 21'd0;
51                  SF_D <= 4'b0011;
52                  LCD_E <= 1'b1;
53          end
54          if ((count == 21'd20) && (d==8'd3))
55          begin
56                  d <= d + 8'd1;
57                  count <= 21'd0;
58                  SF_D <= 4'b0000;
59                  LCD_E <= 1'b0;
60          end
61          if ((count == 21'd5000) &&(d==8'd4))
62          begin
63                  d <= d + 8'd1;
64                  count <= 21'd0;
65                  SF_D <= 4'b0011;
66                  LCD_E <= 1'b1;
67          end
68          if ((count == 21'd20) && (d==8'd5))
69          begin
70                  d <= d + 8'd1;
71                  count <= 21'd0;
72                  SF_D <= 4'b0000;
73                  LCD_E <= 1'b0;
74          end
75          if ((count == 21'd2200) && (d==8'd6))
76          begin
77                  d <= d + 8'd1;
```

```
78                  count <= 21'd0;
79                  SF_D <= 4'b0010;
80                  LCD_E <= 1'b1;
81          end
82          if ((count == 21'd20) && (d==8'd7))
83          begin
84                  d <= d + 8'd1;
85                  count <= 21'd0;
86                  SF_D <= 4'b0000;
87                  LCD_E <= 1'b0;
88          end
89          //DISPLAY CONFIG
90          //Function Set Command
91          if ((count == 21'd2200) && (d==8'd8))
92          begin
93                  d <= d + 8'd1;
94                  count <= 21'd0;
95                  SF_D <= 4'b0010;
96                  LCD_E <= 1'b1;
97          end
98          if ((count == 21'd20) && (d==8'd9))
99          begin
100                 d <= d + 8'd1;
101                 count <= 21'd0;
102                 SF_D <= 4'b0000;
103                 LCD_E <= 1'b0;
104         end
105         if ((count == 21'd60) && (d==8'd10))
106         begin
107                 d <= d + 8'd1;
108                 count <= 21'd0;
```

```
109                 SF_D <= 4'b1000;
110                 LCD_E <= 1'b1;
111             end
112             if ((count == 21'd20) && (d==8'd11))
113             begin
114                 d <= d + 8'd1;
115                 count <= 21'd0;
116                 SF_D <= 4'b0000;
117                 LCD_E <= 1'b0;
118             end
119             //Entry Mode Set Command
120             if ((count == 21'd2200) && (d==8'd12))
121             begin
122                 d <= d + 8'd1;
123                 count <= 21'd0;
124                 SF_D <= 4'b0000;
125                 LCD_E <= 1'b1;
126             end
127             if ((count == 21'd20) && (d==8'd13))
128             begin
129                 d <= d + 8'd1;
130                 count <= 21'd0;
131                 SF_D <= 4'b0000;
132                 LCD_E <= 1'b0;
133             end
134             if ((count == 21'd60) && (d==8'd14))
135             begin
136                 d <= d + 8'd1;
137                 count <= 21'd0;
138                 SF_D <= 4'b0110;
139                 LCD_E <= 1'b1;
```

```
140             end
141             if ((count == 21'd20) && (d==8'd15))
142             begin
143                     d <= d + 8'd1;
144                     count <= 21'd0;
145                     SF_D <= 4'b0000;
146                     LCD_E <= 1'b0;
147             end
148             //Display ON/OFF command
149             if ((count == 21'd2200) && (d==8'd16))
150             begin
151                     d <= d + 8'd1;
152                     count <= 21'd0;
153                     SF_D <= 4'b0000;
154                     LCD_E <= 1'b1;
155             end
156             if ((count == 21'd20) && (d==8'd17))
157             begin
158                     d <= d + 8'd1;
159                     count <= 21'd0;
160                     SF_D <= 4'b0000;
161                     LCD_E <= 1'b0;
162             end
163             if ((count == 21'd60) && (d==8'd18))
164             begin
165                     d <= d + 8'd1;
166                     count <= 21'd0;
167                     SF_D <= 4'b1111;
168                     LCD_E <= 1'b1;
169             end
170             if ((count == 21'd20) && (d==8'd19))
```

```
171         begin
172                 d <= d + 8'd1;
173                 count <= 21'd0;
174                 SF_D <= 4'b0000;
175                 LCD_E <= 1'b0;
176         end
177         //Clear Display
178         if ((count == 21'd2200) && (d==8'd20))
179         begin
180                 d <= d + 8'd1;
181                 count <= 21'd0;
182                 SF_D <= 4'b0000;
183                 LCD_E <= 1'b1;
184         end
185         if ((count == 21'd20) && (d==8'd21))
186         begin
187                 d <= d + 8'd1;
188                 count <= 21'd0;
189                 SF_D <= 4'b0000;
190                 LCD_E <= 1'b0;
191         end
192         if ((count == 21'd60) && (d==8'd22))
193         begin
194                 d <= d + 8'd1;
195                 count <= 21'd0;
196                 SF_D <= 4'b0001;
197                 LCD_E <= 1'b1;
198         end
199         if ((count == 21'd20) && (d==8'd23))
200         begin
201                 d <= d + 8'd1;
```

```
202                     count <= 21'd0;
203                     SF_D <= 4'b0000;
204                     LCD_E <= 1'b0;
205             end
206             if ((count == 21'd80000) && (d==8'd24))
207             begin
208                     d <= d + 8'd1;
209                     count <= 21'd0;
210                     SF_D <= 4'b0000;
211                     LCD_E <= 1'b0;
212             end
213 end
214
215 1'b0:
216 begin
217             count <= count + 21'd1;
218             //Write Address and Data
219             //Write Initial Address
220             if ((count == 21'd2000) && (d==8'd25))
221             begin
222                     d <= d + 8'd1;
223                     count <= 21'd0;
224                     SF_D <= 4'b1000;
225                     LCD_E <= 1'b1;
226                     LCD_RS <= 1'b0;
227                     LCD_RW <=1'b0;
228             end
229             if ((count == 21'd20) && (d==8'd26))
230             begin
231                     d <= d + 8'd1;
232                     count <= 21'd0;
```

```
233                 SF_D <= 4'b0000;
234                 LCD_E <= 1'b0;
235                 LCD_RS <= 1'b0;
236                 LCD_RW <=1'b0;
237             end
238             if ((count == 21'd60) && (d==8'd27))
239             begin
240                 d <= d + 8'd1;
241                 count <= 21'd0;
242                 SF_D <= 4'b0000;
243                 LCD_E <= 1'b1;
244                 LCD_RS <= 1'b0;
245                 LCD_RW <=1'b0;
246             end
247             if ((count == 21'd20) && (d==8'd28))
248             begin
249                 d <= d + 8'd1;
250                 count <= 21'd0;
251                 SF_D <= 4'b0000;
252                 LCD_E <= 1'b0;
253             end
254             //Write Data=1 on each first space
255             if ((count == 21'd2200) && (d==8'd29))
256             begin
257                 d <= d + 8'd1;
258                 count <= 21'd0;
259                 SF_D <= display_data[255:252];
260                 LCD_E <= 1'b1;
261                 LCD_RS <= 1'b1;
262                 LCD_RW <=1'b0;
263
```

```
264             end
265             if ((count == 21'd20) && (d==8'd30))
266             begin
267                     d <= d + 8'd1;
268                     count <= 21'd0;
269                     SF_D <= 4'b0000;
270                     LCD_E <= 1'b0;
271             end
272             if ((count == 21'd60) && (d==8'd31))
273             begin
274                     d <= d + 8'd1;
275                     count <= 21'd0;
276                     SF_D <= display_data[251:248];
277                     LCD_E <= 1'b1;
278                     LCD_RS <= 1'b1;
279                     LCD_RW <=1'b0;
280             end
281             if ((count == 21'd20) && (d==8'd32))
282             begin
283                     d <= d + 8'd1;
284                     count <= 21'd0;
285                     SF_D <= 4'b0000;
286                     LCD_E <= 1'b0;
287             end
288             //Data=2
289             if ((count == 21'd2200) && (d==8'd33))
290             begin
291                     d <= d + 8'd1;
292                     count <= 21'd0;
293                     SF_D <= display_data[247:244];
294                     LCD_E <= 1'b1;
```

```
295                 LCD_RS <= 1'b1;
296                 LCD_RW <=1'b0;
297
298         end
299         if ((count == 21'd20) && (d==8'd34))
300         begin
301                 d <= d + 8'd1;
302                 count <= 21'd0;
303                 SF_D <= 4'b0000;
304                 LCD_E <= 1'b0;
305         end
306         if ((count == 21'd60) && (d==8'd35))
307         begin
308                 d <= d + 8'd1;
309                 count <= 21'd0;
310                 SF_D <= display_data[243:240];
311                 LCD_E <= 1'b1;
312                 LCD_RS <= 1'b1;
313                 LCD_RW <=1'b0;
314         end
315         if ((count == 21'd20) && (d==8'd36))
316         begin
317                 d <= d + 8'd1;
318                 count <= 21'd0;
319                 SF_D <= 4'b0000;
320                 LCD_E <= 1'b0;
321         end
322         //Data=3
323         if ((count == 21'd2200) && (d==8'd37))
324         begin
325                 d <= d + 8'd1;
```

```
326                 count <= 21'd0;
327                 SF_D <= display_data[239:236];
328                 LCD_E <= 1'b1;
329                 LCD_RS <= 1'b1;
330                 LCD_RW <=1'b0;
331
332         end
333         if ((count == 21'd20) && (d==8'd38))
334         begin
335                 d <= d + 8'd1;
336                 count <= 21'd0;
337                 SF_D <= 4'b0000;
338                 LCD_E <= 1'b0;
339         end
340         if ((count == 21'd60) && (d==8'd39))
341         begin
342                 d <= d + 8'd1;
343                 count <= 21'd0;
344                 SF_D <= display_data[235:232];
345                 LCD_E <= 1'b1;
346                 LCD_RS <= 1'b1;
347                 LCD_RW <=1'b0;
348         end
349         if ((count == 21'd20) && (d==8'd40))
350         begin
351                 d <= d + 8'd1;
352                 count <= 21'd0;
353                 SF_D <= 4'b0000;
354                 LCD_E <= 1'b0;
355         end
356         //Data=4
```

```
357             if ((count == 21'd2200) && (d==8'd41))
358             begin
359                     d <= d + 8'd1;
360                     count <= 21'd0;
361                     SF_D <= display_data[231:228];
362                     LCD_E <= 1'b1;
363                     LCD_RS <= 1'b1;
364                     LCD_RW <=1'b0;
365
366             end
367             if ((count == 21'd20) && (d==8'd42))
368             begin
369                     d <= d + 8'd1;
370                     count <= 21'd0;
371                     SF_D <= 4'b0000;
372                     LCD_E <= 1'b0;
373             end
374             if ((count == 21'd60) && (d==8'd43))
375             begin
376                     d <= d + 8'd1;
377                     count <= 21'd0;
378                     SF_D <= display_data[227:224];
379                     LCD_E <= 1'b1;
380                     LCD_RS <= 1'b1;
381                     LCD_RW <=1'b0;
382             end
383             if ((count == 21'd20) && (d==8'd44))
384             begin
385                     d <= d + 8'd1;
386                     count <= 21'd0;
387                     SF_D <= 4'b0000;
```

```
388                 LCD_E <= 1'b0;
389             end
390             //Data=5
391             if ((count == 21'd2200) && (d==8'd45))
392             begin
393                 d <= d + 8'd1;
394                 count <= 21'd0;
395                 SF_D <= display_data[223:220];
396                 LCD_E <= 1'b1;
397                 LCD_RS <= 1'b1;
398                 LCD_RW <=1'b0;
399
400             end
401             if ((count == 21'd20) && (d==8'd46))
402             begin
403                 d <= d + 8'd1;
404                 count <= 21'd0;
405                 SF_D <= 4'b0000;
406                 LCD_E <= 1'b0;
407             end
408             if ((count == 21'd60) && (d==8'd47))
409             begin
410                 d <= d + 8'd1;
411                 count <= 21'd0;
412                 SF_D <= display_data[219:216];
413                 LCD_E <= 1'b1;
414                 LCD_RS <= 1'b1;
415                 LCD_RW <=1'b0;
416             end
417             if ((count == 21'd20) && (d==8'd48))
418             begin
```

```
419                 d <= d + 8'd1;
420                 count <= 21'd0;
421                 SF_D <= 4'b0000;
422                 LCD_E <= 1'b0;
423         end
424         //Data=6
425         if ((count == 21'd2200) && (d==8'd49))
426         begin
427                 d <= d + 8'd1;
428                 count <= 21'd0;
429                 SF_D <= display_data[215:212];
430                 LCD_E <= 1'b1;
431                 LCD_RS <= 1'b1;
432                 LCD_RW <=1'b0;
433
434         end
435         if ((count == 21'd20) && (d==8'd50))
436         begin
437                 d <= d + 8'd1;
438                 count <= 21'd0;
439                 SF_D <= 4'b0000;
440                 LCD_E <= 1'b0;
441         end
442         if ((count == 21'd60) && (d==8'd51))
443         begin
444                 d <= d + 8'd1;
445                 count <= 21'd0;
446                 SF_D <= display_data[211:208];
447                 LCD_E <= 1'b1;
448                 LCD_RS <= 1'b1;
449                 LCD_RW <=1'b0;
```

```
450             end
451             if ((count == 21'd20) && (d==8'd52))
452             begin
453                     d <= d + 8'd1;
454                     count <= 21'd0;
455                     SF_D <= 4'b0000;
456                     LCD_E <= 1'b0;
457             end
458             //Data=7
459             if ((count == 21'd2200) && (d==8'd53))
460             begin
461                     d <= d + 8'd1;
462                     count <= 21'd0;
463                     SF_D <= display_data[207:204];
464                     LCD_E <= 1'b1;
465                     LCD_RS <= 1'b1;
466                     LCD_RW <=1'b0;
467
468             end
469             if ((count == 21'd20) && (d==8'd54))
470             begin
471                     d <= d + 8'd1;
472                     count <= 21'd0;
473                     SF_D <= 4'b0000;
474                     LCD_E <= 1'b0;
475             end
476             if ((count == 21'd60) && (d==8'd55))
477             begin
478                     d <= d + 8'd1;
479                     count <= 21'd0;
480                     SF_D <= display_data[203:200];
```

```
481                 LCD_E <= 1'b1;
482                 LCD_RS <= 1'b1;
483                 LCD_RW <=1'b0;
484         end
485         if ((count == 21'd20) && (d==8'd56))
486         begin
487                 d <= d + 8'd1;
488                 count <= 21'd0;
489                 SF_D <= 4'b0000;
490                 LCD_E <= 1'b0;
491         end
492         //Data=8
493         if ((count == 21'd2200) && (d==8'd57))
494         begin
495                 d <= d + 8'd1;
496                 count <= 21'd0;
497                 SF_D <= display_data[199:196];
498                 LCD_E <= 1'b1;
499                 LCD_RS <= 1'b1;
500                 LCD_RW <=1'b0;
501
502         end
503         if ((count == 21'd20) && (d==8'd58))
504         begin
505                 d <= d + 8'd1;
506                 count <= 21'd0;
507                 SF_D <= 4'b0000;
508                 LCD_E <= 1'b0;
509         end
510         if ((count == 21'd60) && (d==8'd59))
511         begin
```

```
512                 d <= d + 8'd1;
513                 count <= 21'd0;
514                 SF_D <= display_data[195:192];
515                 LCD_E <= 1'b1;
516                 LCD_RS <= 1'b1;
517                 LCD_RW <=1'b0;
518         end
519         if ((count == 21'd20) && (d==8'd60))
520         begin
521                 d <= d + 8'd1;
522                 count <= 21'd0;
523                 SF_D <= 4'b0000;
524                 LCD_E <= 1'b0;
525         end
526         //Data=9
527         if ((count == 21'd2200) && (d==8'd61))
528         begin
529                 d <= d + 8'd1;
530                 count <= 21'd0;
531                 SF_D <= display_data[191:188];
532                 LCD_E <= 1'b1;
533                 LCD_RS <= 1'b1;
534                 LCD_RW <=1'b0;
535
536         end
537         if ((count == 21'd20) && (d==8'd62))
538         begin
539                 d <= d + 8'd1;
540                 count <= 21'd0;
541                 SF_D <= 4'b0000;
542                 LCD_E <= 1'b0;
```

```
543             end
544             if ((count == 21'd60) && (d==8'd63))
545             begin
546                     d <= d + 8'd1;
547                     count <= 21'd0;
548                     SF_D <= display_data[187:184];
549                     LCD_E <= 1'b1;
550                     LCD_RS <= 1'b1;
551                     LCD_RW <=1'b0;
552             end
553             if ((count == 21'd20) && (d==8'd64))
554             begin
555                     d <= d + 8'd1;
556                     count <= 21'd0;
557                     SF_D <= 4'b0000;
558                     LCD_E <= 1'b0;
559             end
560             //Data=10
561             if ((count == 21'd2200) && (d==8'd65))
562             begin
563                     d <= d + 8'd1;
564                     count <= 21'd0;
565                     SF_D <= display_data[183:180];
566                     LCD_E <= 1'b1;
567                     LCD_RS <= 1'b1;
568                     LCD_RW <=1'b0;
569
570             end
571             if ((count == 21'd20) && (d==8'd66))
572             begin
573                     d <= d + 8'd1;
```

```
574                 count <= 21'd0;
575                 SF_D <= 4'b0000;
576                 LCD_E <= 1'b0;
577         end
578         if ((count == 21'd60) && (d==8'd67))
579         begin
580                 d <= d + 8'd1;
581                 count <= 21'd0;
582                 SF_D <= display_data[179:176];
583                 LCD_E <= 1'b1;
584                 LCD_RS <= 1'b1;
585                 LCD_RW <=1'b0;
586         end
587         if ((count == 21'd20) && (d==8'd68))
588         begin
589                 d <= d + 8'd1;
590                 count <= 21'd0;
591                 SF_D <= 4'b0000;
592                 LCD_E <= 1'b0;
593         end
594         //Data=11
595         if ((count == 21'd2200) && (d==8'd69))
596         begin
597                 d <= d + 8'd1;
598                 count <= 21'd0;
599                 SF_D <= display_data[175:172];
600                 LCD_E <= 1'b1;
601                 LCD_RS <= 1'b1;
602                 LCD_RW <=1'b0;
603
604         end
```

```
605         if ((count == 21'd20) && (d==8'd70))
606         begin
607                 d <= d + 8'd1;
608                 count <= 21'd0;
609                 SF_D <= 4'b0000;
610                 LCD_E <= 1'b0;
611         end
612         if ((count == 21'd60) && (d==8'd71))
613         begin
614                 d <= d + 8'd1;
615                 count <= 21'd0;
616                 SF_D <= display_data[171:168];
617                 LCD_E <= 1'b1;
618                 LCD_RS <= 1'b1;
619                 LCD_RW <=1'b0;
620         end
621         if ((count == 21'd20) && (d==8'd72))
622         begin
623                 d <= d + 8'd1;
624                 count <= 21'd0;
625                 SF_D <= 4'b0000;
626                 LCD_E <= 1'b0;
627         end
628         //Data=12
629         if ((count == 21'd2200) && (d==8'd73))
630         begin
631                 d <= d + 8'd1;
632                 count <= 21'd0;
633                 SF_D <= display_data[167:164];
634                 LCD_E <= 1'b1;
635                 LCD_RS <= 1'b1;
```

```
636                 LCD_RW <=1'b0;
637
638         end
639         if ((count == 21'd20) && (d==8'd74))
640         begin
641                 d <= d + 8'd1;
642                 count <= 21'd0;
643                 SF_D <= 4'b0000;
644                 LCD_E <= 1'b0;
645         end
646         if ((count == 21'd60) && (d==8'd75))
647         begin
648                 d <= d + 8'd1;
649                 count <= 21'd0;
650                 SF_D <= display_data[163:160];
651                 LCD_E <= 1'b1;
652                 LCD_RS <= 1'b1;
653                 LCD_RW <=1'b0;
654         end
655         if ((count == 21'd20) && (d==8'd76))
656         begin
657                 d <= d + 8'd1;
658                 count <= 21'd0;
659                 SF_D <= 4'b0000;
660                 LCD_E <= 1'b0;
661         end
662         //Data=13
663         if ((count == 21'd2200) && (d==8'd77))
664         begin
665                 d <= d + 8'd1;
666                 count <= 21'd0;
```

```
667                 SF_D <= display_data[159:156];
668                 LCD_E <= 1'b1;
669                 LCD_RS <= 1'b1;
670                 LCD_RW <=1'b0;
671
672         end
673         if ((count == 21'd20) && (d==8'd78))
674         begin
675                 d <= d + 8'd1;
676                 count <= 21'd0;
677                 SF_D <= 4'b0000;
678                 LCD_E <= 1'b0;
679         end
680         if ((count == 21'd60) && (d==8'd79))
681         begin
682                 d <= d + 8'd1;
683                 count <= 21'd0;
684                 SF_D <= display_data[155:152];
685                 LCD_E <= 1'b1;
686                 LCD_RS <= 1'b1;
687                 LCD_RW <=1'b0;
688         end
689         if ((count == 21'd20) && (d==8'd80))
690         begin
691                 d <= d + 8'd1;
692                 count <= 21'd0;
693                 SF_D <= 4'b0000;
694                 LCD_E <= 1'b0;
695         end
696         //Data=14
697         if ((count == 21'd2200) && (d==8'd81))
```

```
698         begin
699                 d <= d + 8'd1;
700                 count <= 21'd0;
701                 SF_D <= display_data[151:148];
702                 LCD_E <= 1'b1;
703                 LCD_RS <= 1'b1;
704                 LCD_RW <=1'b0;
705
706         end
707         if ((count == 21'd20) && (d==8'd82))
708         begin
709                 d <= d + 8'd1;
710                 count <= 21'd0;
711                 SF_D <= 4'b0000;
712                 LCD_E <= 1'b0;
713         end
714         if ((count == 21'd60) && (d==8'd83))
715         begin
716                 d <= d + 8'd1;
717                 count <= 21'd0;
718                 SF_D <= display_data[147:144];
719                 LCD_E <= 1'b1;
720                 LCD_RS <= 1'b1;
721                 LCD_RW <=1'b0;
722         end
723         if ((count == 21'd20) && (d==8'd84))
724         begin
725                 d <= d + 8'd1;
726                 count <= 21'd0;
727                 SF_D <= 4'b0000;
728                 LCD_E <= 1'b0;
```

```
729             end
730             //Data=15
731             if ((count == 21'd2200) && (d==8'd85))
732             begin
733                     d <= d + 8'd1;
734                     count <= 21'd0;
735                     SF_D <= display_data[143:140];
736                     LCD_E <= 1'b1;
737                     LCD_RS <= 1'b1;
738                     LCD_RW <=1'b0;
739
740             end
741             if ((count == 21'd20) && (d==8'd86))
742             begin
743                     d <= d + 8'd1;
744                     count <= 21'd0;
745                     SF_D <= 4'b0000;
746                     LCD_E <= 1'b0;
747             end
748             if ((count == 21'd60) && (d==8'd87))
749             begin
750                     d <= d + 8'd1;
751                     count <= 21'd0;
752                     SF_D <= display_data[139:136];
753                     LCD_E <= 1'b1;
754                     LCD_RS <= 1'b1;
755                     LCD_RW <=1'b0;
756             end
757             if ((count == 21'd20) && (d==8'd88))
758             begin
759                     d <= d + 8'd1;
```

```
760                 count <= 21'd0;
761                 SF_D <= 4'b0000;
762                 LCD_E <= 1'b0;
763         end
764         //Data=16
765         if ((count == 21'd2200) && (d==8'd89))
766         begin
767                 d <= d + 8'd1;
768                 count <= 21'd0;
769                 SF_D <= display_data[135:132];
770                 LCD_E <= 1'b1;
771                 LCD_RS <= 1'b1;
772                 LCD_RW <=1'b0;
773
774         end
775         if ((count == 21'd20) && (d==8'd90))
776         begin
777                 d <= d + 8'd1;
778                 count <= 21'd0;
779                 SF_D <= 4'b0000;
780                 LCD_E <= 1'b0;
781         end
782         if ((count == 21'd60) && (d==8'd91))
783         begin
784                 d <= d + 8'd1;
785                 count <= 21'd0;
786                 SF_D <= display_data[131:128];
787                 LCD_E <= 1'b1;
788                 LCD_RS <= 1'b1;
789                 LCD_RW <=1'b0;
790         end
```

```
791         if ((count == 21'd20) && (d==8'd92))
792         begin
793                 d <= d + 8'd1;
794                 count <= 21'd0;
795                 SF_D <= 4'b0000;
796                 LCD_E <= 1'b0;
797         end
798         //Write 2nd Line Initial Address
799         if ((count == 21'd2000) && (d==8'd93))
800         begin
801                 d <= d + 8'd1;
802                 count <= 21'd0;
803                 SF_D <= 4'b1100;
804                 LCD_E <= 1'b1;
805                 LCD_RS <= 1'b0;
806                 LCD_RW <=1'b0;
807         end
808         if ((count == 21'd20) && (d==8'd94))
809         begin
810                 d <= d + 8'd1;
811                 count <= 21'd0;
812                 SF_D <= 4'b0000;
813                 LCD_E <= 1'b0;
814         end
815         if ((count == 21'd60) && (d==8'd95))
816         begin
817                 d <= d + 8'd1;
818                 count <= 21'd0;
819                 SF_D <= 4'b0000;
820                 LCD_E <= 1'b1;
821                 LCD_RS <= 1'b0;
```

```
822                 LCD_RW <=1'b0;
823             end
824             if ((count == 21'd20) && (d==8'd96))
825             begin
826                 d <= d + 8'd1;
827                 count <= 21'd0;
828                 SF_D <= 4'b0000;
829                 LCD_E <= 1'b0;
830             end
831             ///Data=17
832             if ((count == 21'd2200) && (d==8'd97))
833             begin
834                 d <= d + 8'd1;
835                 count <= 21'd0;
836                 SF_D <= display_data[127:124];
837                 LCD_E <= 1'b1;
838                 LCD_RS <= 1'b1;
839                 LCD_RW <=1'b0;
840
841             end
842             if ((count == 21'd20) && (d==8'd98))
843             begin
844                 d <= d + 8'd1;
845                 count <= 21'd0;
846                 SF_D <= 4'b0000;
847                 LCD_E <= 1'b0;
848             end
849             if ((count == 21'd60) && (d==8'd99))
850             begin
851                 d <= d + 8'd1;
852                 count <= 21'd0;
```

```
853                 SF_D <= display_data[123:120];
854                 LCD_E <= 1'b1;
855                 LCD_RS <= 1'b1;
856                 LCD_RW <=1'b0;
857         end
858         if ((count == 21'd20) && (d==8'd100))
859         begin
860                 d <= d + 8'd1;
861                 count <= 21'd0;
862                 SF_D <= 4'b0000;
863                 LCD_E <= 1'b0;
864         end
865         //Data=18
866         if ((count == 21'd2200) && (d==8'd101))
867         begin
868                 d <= d + 8'd1;
869                 count <= 21'd0;
870                 SF_D <= display_data[119:116];
871                 LCD_E <= 1'b1;
872                 LCD_RS <= 1'b1;
873                 LCD_RW <=1'b0;
874
875         end
876         if ((count == 21'd20) && (d==8'd102))
877         begin
878                 d <= d + 8'd1;
879                 count <= 21'd0;
880                 SF_D <= 4'b0000;
881                 LCD_E <= 1'b0;
882         end
883         if ((count == 21'd60) && (d==8'd103))
```

```
884         begin
885                 d <= d + 8'd1;
886                 count <= 21'd0;
887                 SF_D <= display_data[115:112];
888                 LCD_E <= 1'b1;
889                 LCD_RS <= 1'b1;
890                 LCD_RW <=1'b0;
891         end
892         if ((count == 21'd20) && (d==8'd104))
893         begin
894                 d <= d + 8'd1;
895                 count <= 21'd0;
896                 SF_D <= 4'b0000;
897                 LCD_E <= 1'b0;
898         end
899         //Data=19
900         if ((count == 21'd2200) && (d==8'd105))
901         begin
902                 d <= d + 8'd1;
903                 count <= 21'd0;
904                 SF_D <= display_data[111:108];
905                 LCD_E <= 1'b1;
906                 LCD_RS <= 1'b1;
907                 LCD_RW <=1'b0;
908
909         end
910         if ((count == 21'd20) && (d==8'd106))
911         begin
912                 d <= d + 8'd1;
913                 count <= 21'd0;
914                 SF_D <= 4'b0000;
```

```
915             LCD_E <= 1'b0;
916         end
917         if ((count == 21'd60) && (d==8'd107))
918         begin
919             d <= d + 8'd1;
920             count <= 21'd0;
921             SF_D <= display_data[107:104];
922             LCD_E <= 1'b1;
923             LCD_RS <= 1'b1;
924             LCD_RW <=1'b0;
925         end
926         if ((count == 21'd20) && (d==8'd108))
927         begin
928             d <= d + 8'd1;
929             count <= 21'd0;
930             SF_D <= 4'b0000;
931             LCD_E <= 1'b0;
932         end
933         //Data=20
934         if ((count == 21'd2200) && (d==8'd109))
935         begin
936             d <= d + 8'd1;
937             count <= 21'd0;
938             SF_D <= display_data[103:100];
939             LCD_E <= 1'b1;
940             LCD_RS <= 1'b1;
941             LCD_RW <=1'b0;
942
943         end
944         if ((count == 21'd20) && (d==8'd110))
945         begin
```

```
946                 d <= d + 8'd1;
947                 count <= 21'd0;
948                 SF_D <= 4'b0000;
949                 LCD_E <= 1'b0;
950         end
951         if ((count == 21'd60) && (d==8'd111))
952         begin
953                 d <= d + 8'd1;
954                 count <= 21'd0;
955                 SF_D <= display_data[99:96];
956                 LCD_E <= 1'b1;
957                 LCD_RS <= 1'b1;
958                 LCD_RW <=1'b0;
959         end
960         if ((count == 21'd20) && (d==8'd112))
961         begin
962                 d <= d + 8'd1;
963                 count <= 21'd0;
964                 SF_D <= 4'b0000;
965                 LCD_E <= 1'b0;
966         end
967         //Data=21
968         if ((count == 21'd2200) && (d==8'd113))
969         begin
970                 d <= d + 8'd1;
971                 count <= 21'd0;
972                 SF_D <= display_data[95:92];
973                 LCD_E <= 1'b1;
974                 LCD_RS <= 1'b1;
975                 LCD_RW <=1'b0;
976
```

```
977         end
978         if ((count == 21'd20) && (d==8'd114))
979         begin
980                 d <= d + 8'd1;
981                 count <= 21'd0;
982                 SF_D <= 4'b0000;
983                 LCD_E <= 1'b0;
984         end
985         if ((count == 21'd60) && (d==8'd115))
986         begin
987                 d <= d + 8'd1;
988                 count <= 21'd0;
989                 SF_D <= display_data[91:88];
990                 LCD_E <= 1'b1;
991                 LCD_RS <= 1'b1;
992                 LCD_RW <=1'b0;
993         end
994         if ((count == 21'd20) && (d==8'd116))
995         begin
996                 d <= d + 8'd1;
997                 count <= 21'd0;
998                 SF_D <= 4'b0000;
999                 LCD_E <= 1'b0;
1000        end
1001        //Data=22
1002        if ((count == 21'd2200) && (d==8'd117))
1003        begin
1004                d <= d + 8'd1;
1005                count <= 21'd0;
1006                SF_D <= display_data[87:84];
1007                LCD_E <= 1'b1;
```

```
1008                LCD_RS <= 1'b1;
1009                LCD_RW <=1'b0;
1010
1011        end
1012        if ((count == 21'd20) && (d==8'd118))
1013        begin
1014                d <= d + 8'd1;
1015                count <= 21'd0;
1016                SF_D <= 4'b0000;
1017                LCD_E <= 1'b0;
1018        end
1019        if ((count == 21'd60) && (d==8'd119))
1020        begin
1021                d <= d + 8'd1;
1022                count <= 21'd0;
1023                SF_D <= display_data[83:80];
1024                LCD_E <= 1'b1;
1025                LCD_RS <= 1'b1;
1026                LCD_RW <=1'b0;
1027        end
1028        if ((count == 21'd20) && (d==8'd120))
1029        begin
1030                d <= d + 8'd1;
1031                count <= 21'd0;
1032                SF_D <= 4'b0000;
1033                LCD_E <= 1'b0;
1034        end
1035        //Data=23
1036        if ((count == 21'd2200) && (d==8'd121))
1037        begin
1038                d <= d + 8'd1;
```

```
1039                count <= 21'd0;
1040                SF_D <= display_data[79:76];
1041                LCD_E <= 1'b1;
1042                LCD_RS <= 1'b1;
1043                LCD_RW <=1'b0;
1044
1045        end
1046        if ((count == 21'd20) && (d==8'd122))
1047        begin
1048                d <= d + 8'd1;
1049                count <= 21'd0;
1050                SF_D <= 4'b0000;
1051                LCD_E <= 1'b0;
1052        end
1053        if ((count == 21'd60) && (d==8'd123))
1054        begin
1055                d <= d + 8'd1;
1056                count <= 21'd0;
1057                SF_D <= display_data[75:72];
1058                LCD_E <= 1'b1;
1059                LCD_RS <= 1'b1;
1060                LCD_RW <=1'b0;
1061        end
1062        if ((count == 21'd20) && (d==8'd124))
1063        begin
1064                d <= d + 8'd1;
1065                count <= 21'd0;
1066                SF_D <= 4'b0000;
1067                LCD_E <= 1'b0;
1068        end
1069        //Data=24
```

```
1070            if ((count == 21'd2200) && (d==8'd125))
1071            begin
1072                    d <= d + 8'd1;
1073                    count <= 21'd0;
1074                    SF_D <= display_data[71:68];
1075                    LCD_E <= 1'b1;
1076                    LCD_RS <= 1'b1;
1077                    LCD_RW <=1'b0;
1078
1079            end
1080            if ((count == 21'd20) && (d==8'd126))
1081            begin
1082                    d <= d + 8'd1;
1083                    count <= 21'd0;
1084                    SF_D <= 4'b0000;
1085                    LCD_E <= 1'b0;
1086            end
1087            if ((count == 21'd60) && (d==8'd127))
1088            begin
1089                    d <= d + 8'd1;
1090                    count <= 21'd0;
1091                    SF_D <= display_data[67:64];
1092                    LCD_E <= 1'b1;
1093                    LCD_RS <= 1'b1;
1094                    LCD_RW <=1'b0;
1095            end
1096            if ((count == 21'd20) && (d==8'd128))
1097            begin
1098                    d <= d + 8'd1;
1099                    count <= 21'd0;
1100                    SF_D <= 4'b0000;
```

```
1101                LCD_E <= 1'b0;
1102            end
1103            //Data=25
1104            if ((count == 21'd2200) && (d==8'd129))
1105            begin
1106                d <= d + 8'd1;
1107                count <= 21'd0;
1108                SF_D <= display_data[63:60];
1109                LCD_E <= 1'b1;
1110                LCD_RS <= 1'b1;
1111                LCD_RW <=1'b0;
1112
1113            end
1114            if ((count == 21'd20) && (d==8'd130))
1115            begin
1116                d <= d + 8'd1;
1117                count <= 21'd0;
1118                SF_D <= 4'b0000;
1119                LCD_E <= 1'b0;
1120            end
1121            if ((count == 21'd60) && (d==8'd131))
1122            begin
1123                d <= d + 8'd1;
1124                count <= 21'd0;
1125                SF_D <= display_data[59:56];
1126                LCD_E <= 1'b1;
1127                LCD_RS <= 1'b1;
1128                LCD_RW <=1'b0;
1129            end
1130            if ((count == 21'd20) && (d==8'd132))
1131            begin
```

```
1132                d <= d + 8'd1;
1133                count <= 21'd0;
1134                SF_D <= 4'b0000;
1135                LCD_E <= 1'b0;
1136        end
1137        //Data=26
1138        if ((count == 21'd2200) && (d==8'd133))
1139        begin
1140                d <= d + 8'd1;
1141                count <= 21'd0;
1142                SF_D <= display_data[55:52];
1143                LCD_E <= 1'b1;
1144                LCD_RS <= 1'b1;
1145                LCD_RW <=1'b0;
1146
1147        end
1148        if ((count == 21'd20) && (d==8'd134))
1149        begin
1150                d <= d + 8'd1;
1151                count <= 21'd0;
1152                SF_D <= 4'b0000;
1153                LCD_E <= 1'b0;
1154        end
1155        if ((count == 21'd60) && (d==8'd135))
1156        begin
1157                d <= d + 8'd1;
1158                count <= 21'd0;
1159                SF_D <= display_data[51:48];
1160                LCD_E <= 1'b1;
1161                LCD_RS <= 1'b1;
1162                LCD_RW <=1'b0;
```

```
1163            end
1164            if ((count == 21'd20) && (d==8'd136))
1165            begin
1166                    d <= d + 8'd1;
1167                    count <= 21'd0;
1168                    SF_D <= 4'b0000;
1169                    LCD_E <= 1'b0;
1170            end
1171            //Data=27
1172            if ((count == 21'd2200) && (d==8'd137))
1173            begin
1174                    d <= d + 8'd1;
1175                    count <= 21'd0;
1176                    SF_D <= display_data[47:44];
1177                    LCD_E <= 1'b1;
1178                    LCD_RS <= 1'b1;
1179                    LCD_RW <=1'b0;
1180
1181            end
1182            if ((count == 21'd20) && (d==8'd138))
1183            begin
1184                    d <= d + 8'd1;
1185                    count <= 21'd0;
1186                    SF_D <= 4'b0000;
1187                    LCD_E <= 1'b0;
1188            end
1189            if ((count == 21'd60) && (d==8'd139))
1190            begin
1191                    d <= d + 8'd1;
1192                    count <= 21'd0;
1193                    SF_D <= display_data[43:40];
```

```
1194                LCD_E <= 1'b1;
1195                LCD_RS <= 1'b1;
1196                LCD_RW <=1'b0;
1197            end
1198            if ((count == 21'd20) && (d==8'd140))
1199            begin
1200                d <= d + 8'd1;
1201                count <= 21'd0;
1202                SF_D <= 4'b0000;
1203                LCD_E <= 1'b0;
1204            end
1205            //Data=28
1206            if ((count == 21'd2200) && (d==8'd141))
1207            begin
1208                d <= d + 8'd1;
1209                count <= 21'd0;
1210                SF_D <= display_data[39:36];
1211                LCD_E <= 1'b1;
1212                LCD_RS <= 1'b1;
1213                LCD_RW <=1'b0;
1214
1215            end
1216            if ((count == 21'd20) && (d==8'd142))
1217            begin
1218                d <= d + 8'd1;
1219                count <= 21'd0;
1220                SF_D <= 4'b0000;
1221                LCD_E <= 1'b0;
1222            end
1223            if ((count == 21'd60) && (d==8'd143))
1224            begin
```

```
1225                    d <= d + 8'd1;
1226                    count <= 21'd0;
1227                    SF_D <= display_data[35:32];
1228                    LCD_E <= 1'b1;
1229                    LCD_RS <= 1'b1;
1230                    LCD_RW <=1'b0;
1231            end
1232            if ((count == 21'd20) && (d==8'd144))
1233            begin
1234                    d <= d + 8'd1;
1235                    count <= 21'd0;
1236                    SF_D <= 4'b0000;
1237                    LCD_E <= 1'b0;
1238            end
1239            //Data=29
1240            if ((count == 21'd2200) && (d==8'd145))
1241            begin
1242                    d <= d + 8'd1;
1243                    count <= 21'd0;
1244                    SF_D <= display_data[31:28];
1245                    LCD_E <= 1'b1;
1246                    LCD_RS <= 1'b1;
1247                    LCD_RW <=1'b0;
1248
1249            end
1250            if ((count == 21'd20) && (d==8'd146))
1251            begin
1252                    d <= d + 8'd1;
1253                    count <= 21'd0;
1254                    SF_D <= 4'b0000;
1255                    LCD_E <= 1'b0;
```

```
1256            end
1257            if ((count == 21'd60) && (d==8'd147))
1258            begin
1259                    d <= d + 8'd1;
1260                    count <= 21'd0;
1261                    SF_D <= display_data[27:24];
1262                    LCD_E <= 1'b1;
1263                    LCD_RS <= 1'b1;
1264                    LCD_RW <=1'b0;
1265            end
1266            if ((count == 21'd20) && (d==8'd148))
1267            begin
1268                    d <= d + 8'd1;
1269                    count <= 21'd0;
1270                    SF_D <= 4'b0000;
1271                    LCD_E <= 1'b0;
1272            end
1273            //Data=30
1274            if ((count == 21'd2200) && (d==8'd149))
1275            begin
1276                    d <= d + 8'd1;
1277                    count <= 21'd0;
1278                    SF_D <= display_data[23:20];
1279                    LCD_E <= 1'b1;
1280                    LCD_RS <= 1'b1;
1281                    LCD_RW <=1'b0;
1282
1283            end
1284            if ((count == 21'd20) && (d==8'd150))
1285            begin
1286                    d <= d + 8'd1;
```

```
1287                count <= 21'd0;
1288                SF_D <= 4'b0000;
1289                LCD_E <= 1'b0;
1290        end
1291        if ((count == 21'd60) && (d==8'd151))
1292        begin
1293                d <= d + 8'd1;
1294                count <= 21'd0;
1295                SF_D <= display_data[19:16];
1296                LCD_E <= 1'b1;
1297                LCD_RS <= 1'b1;
1298                LCD_RW <=1'b0;
1299        end
1300        if ((count == 21'd20) && (d==8'd152))
1301        begin
1302                d <= d + 8'd1;
1303                count <= 21'd0;
1304                SF_D <= 4'b0000;
1305                LCD_E <= 1'b0;
1306        end
1307        //Data=31
1308        if ((count == 21'd2200) && (d==8'd153))
1309        begin
1310                d <= d + 8'd1;
1311                count <= 21'd0;
1312                SF_D <= display_data[15:12];
1313                LCD_E <= 1'b1;
1314                LCD_RS <= 1'b1;
1315                LCD_RW <=1'b0;
1316
1317        end
```

```
1318            if ((count == 21'd20) && (d==8'd154))
1319            begin
1320                    d <= d + 8'd1;
1321                    count <= 21'd0;
1322                    SF_D <= 4'b0000;
1323                    LCD_E <= 1'b0;
1324            end
1325            if ((count == 21'd60) && (d==8'd155))
1326            begin
1327                    d <= d + 8'd1;
1328                    count <= 21'd0;
1329                    SF_D <= display_data[11:8];
1330                    LCD_E <= 1'b1;
1331                    LCD_RS <= 1'b1;
1332                    LCD_RW <=1'b0;
1333            end
1334            if ((count == 21'd20) && (d==8'd156))
1335            begin
1336                    d <= d + 8'd1;
1337                    count <= 21'd0;
1338                    SF_D <= 4'b0000;
1339                    LCD_E <= 1'b0;
1340            end
1341            //Data=32
1342            if ((count == 21'd2200) && (d==8'd157))
1343            begin
1344                    d <= d + 8'd1;
1345                    count <= 21'd0;
1346                    SF_D <= display_data[7:4];
1347                    LCD_E <= 1'b1;
1348                    LCD_RS <= 1'b1;
```

```
1349                    LCD_RW <=1'b0;
1350
1351            end
1352            if ((count == 21'd20) && (d==8'd158))
1353            begin
1354                    d <= d + 8'd1;
1355                    count <= 21'd0;
1356                    SF_D <= 4'b0000;
1357                    LCD_E <= 1'b0;
1358            end
1359            if ((count == 21'd60) && (d==8'd159))
1360            begin
1361                    d <= d + 8'd1;
1362                    count <= 21'd0;
1363                    SF_D <= display_data[3:0];
1364                    LCD_E <= 1'b1;
1365                    LCD_RS <= 1'b1;
1366                    LCD_RW <=1'b0;
1367            end
1368            if ((count == 21'd20) && (d==8'd160))
1369            begin
1370                    d <= 8'd25;
1371                    count <= 21'd0;
1372                    SF_D <= 4'b0000;
1373                    LCD_E <= 1'b0;
1374            end
1375            //Wait for Next Data & Go back to Address Cycle
1376            //Wait for more than half a second here
1377
1378
1379    end
```

```
1380 endcase
1381 end
1382
1383 /*wow this has to be the worst Verilog code I have ever seen...*/
1384 /*okay my modifications start here... let's see how it goes...*/
1385
1386 /*Let's just keep is simple okay  so the following is an example
1387  *of the LCD output when locked with 7 dialed in
1388    >07  LOCKED
1389
1390  *Now here is what it should look like with 17 dialed in after
1391  *being unlocked
1392
1393  >17  UNLOCKED
1394
1395  *ASCII cheat sheet:
1396  *  character  Hex Code
1397  *  >          3e
1398  *  0          30
1399  *  1          31
1400  *  2          32
1401  *  3          33
1402  *  4          34
1403  *  5          35
1404  *  6          36
1405  *  7          37
1406  *  8          38
1407  *  9          39
1408  *  U          55
1409  *  N          4e
1410  *  L          4c
```

```
1411  * O          4f
1412  * C          43
1413  * K          4b
1414  * E          45
1415  * D          44
1416  *SPACE       20
1417 */
1418
1419 /*status_ascii is the ascii representation of LOCKED or UNLOCKED
      */
1420 always@(*)
1421     if(Locked)//print LOCKED
1422         status_ascii = {8'h20, 8'h20, 8'h4c, 8'h4f, 8'h43, 8'h4b, 8'h45, 8
1423     else //print UNLOCKED
1424         status_ascii = {8'h55, 8'h4e, 8'h4c, 8'h4f, 8'h43, 8'h4b, 8'h45, 8
1425
1426 /*count_ascii is the ascii representation of the 2 decimal digits of count*
1427 always@(*)//giant encoder!
1428     case(Position)
1429         5'd0:  count_ascii = 16'h3030;
1430         5'd1:  count_ascii = 16'h3031;
1431         5'd2:  count_ascii = 16'h3032;
1432         5'd3:  count_ascii = 16'h3033;
1433         5'd4:  count_ascii = 16'h3034;
1434         5'd5:  count_ascii = 16'h3035;
1435         5'd6:  count_ascii = 16'h3036;
1436         5'd7:  count_ascii = 16'h3037;
1437         5'd8:  count_ascii = 16'h3038;
1438         5'd9:  count_ascii = 16'h3039;
1439         5'd10: count_ascii = 16'h3130;
1440         5'd11: count_ascii = 16'h3131;
```

```
1441            5'd12: count_ascii = 16'h3132;
1442            5'd13: count_ascii = 16'h3133;
1443            5'd14: count_ascii = 16'h3134;
1444            5'd15: count_ascii = 16'h3135;
1445            5'd16: count_ascii = 16'h3136;
1446            5'd17: count_ascii = 16'h3137;
1447            5'd18: count_ascii = 16'h3138;
1448            5'd19: count_ascii = 16'h3139;
1449            5'd20: count_ascii = 16'h3230;
1450            5'd21: count_ascii = 16'h3231;
1451            5'd22: count_ascii = 16'h3232;
1452            5'd23: count_ascii = 16'h3233;
1453            5'd24: count_ascii = 16'h3234;
1454            5'd25: count_ascii = 16'h3235;
1455            5'd26: count_ascii = 16'h3236;
1456            5'd27: count_ascii = 16'h3237;
1457            5'd28: count_ascii = 16'h3238;
1458            5'd29: count_ascii = 16'h3239;
1459            5'd30: count_ascii = 16'h3330;
1460            5'd31: count_ascii = 16'h3331;
1461        endcase
1462
1463    /*Display data is a 256 bit vector which hold 32 8-bit ascii *
1464     *characters which are to be displayed on the LCD screen     */
1465    assign display_data = {8'h3e, count_ascii, 8'h20, 8'h20, status_ascii, {19-
1466
1467    endmodule
```

Code Block 7: Rotary Encoder Module

```
1  `timescale 1 ns / 1 ps
2  `default_nettype none
```

```
3
4  /*This module takes as input the quadrature outputs *
5   *A and B from the rotary encoder on the Spartan 3e *
6   *board, filters out electrical chatter, and outputs*
7   *a Right and a Left signal. Left pulses every 18   *
8   *as the rotary shaft rotates to the left, while    *
9   *Right pulses every 18 degrees as the rotary shaft *
10  *rotates to the right.                             *
11  *The technique describe here is provided by an     *
12  *an Xilinx application engineer in a document      *
13  *entitled "Rotary Encoder Interface for            *
14  *Spartan-3E Starter Kit"                           */
15 module rotary_encoder_module(
16       output wire Left, Right,
17       input Clk,
18       input wire rotA, rotB
19 );
20
21       /*internal nets*/
22       wire buffA, buffB; //input buffers
23       reg rotary_q1, rotary_q2;
24       reg rotary_event, rotary_left;
25       reg rotary_q1_old;
26
27
28       /*buffer inputs*/
29       synchronizer syncA(buffA, rotA, Clk);
30       synchronizer syncB(buffB, rotB, Clk);
31
32
33       /*elimentate bounce! The result is q1 and q2*/
```

```
34        /*XNOR*/
35        always@(posedge Clk)
36              if(buffA & buffB)
37                    rotary_q1 <= 1'b1;
38              else if(~buffA & ~ buffB)
39                    rotary_q1 <= 1'b0;
40        /*XOR*/
41        always@(posedge Clk)
42              if(~buffA & buffB)
43                    rotary_q2 <= 1'b1;
44              else if(buffA & ~buffB)
45                    rotary_q2 <= 1'b0;
46
47      /*detect rising edge on q1 to signal an event*/
48      always@(posedge Clk)
49              rotary_q1_old <= rotary_q1;
50
51       always@(posedge Clk)
52              rotary_event <= ~rotary_q1_old & rotary_q1;
53
54      /*determine the direction of rotation based on q1 and q2*/
55       always@(posedge Clk)
56              if(~rotary_q1_old & rotary_q1)
57                    rotary_left <= rotary_q2;
58
59       /*generate Left and Right signals*/
60       assign Left  = rotary_event & rotary_left;
61       assign Right = rotary_event & ~rotary_left;
62
63  endmodule
```

The modules were then synthesized onto the FBGA board and tested.

In order to add a 4th digit (17), the following changes were made to the combination lock Verilog code.

Code Block 8: Combination Lock FSM

```
1  `timescale 1ns / 1ps
2
3  module combination_lock_fsm(output reg [2:0] state,
4    output wire Locked, // asserted when locked
5    input wire Right, Left, // indicate direction
6    input wire [4:0] Count, // indicate position
7    input wire Center, // the unlock button
8    input wire Clk, South // clock and reset
9    );
10
11   parameter S0 = 3'b000,
12             S1 = 3'b001,
13             S2 = 3'b010,
14             S3 = 3'b011,
15                          S4 = 3'b100,
16                          S5 = 3'b101;
17
18   reg [2:0] nextState;
19
20   always @ ( * ) begin
21     case (state)
22       S0: begin
23         if (Right)
24                                      nextState = S1;
25         else
26           nextState = S0;
```

```
27          end
28        S1: begin
29          if (Left)
30                                  if(Count == 5'b01101)
31                                          nextState = S2;
32                                  else
33                                          nextState = S0;
34                          else
35                                  nextState = S1;
36          end
37        S2: begin
38          if (Right)
39                                  if (Count == 5'b00111)
40                                          nextState = S3;
41                                  else
42                                          nextState = S0;
43                          else
44                                  nextState = S2;
45          end
46        S3: begin
47                          if (Left)
48                                  if (Count == 5'b10001)
49                                          nextState = S4;
50                                  else
51                                          nextState = S0;
52                          else
53                                  nextState = S3;
54          end
55        S4: begin
56                          if (Center)
57                                  if (Count == 5'b10001)
```

```
58                                          nextState = S5;
59                                  else
60                                          nextState = S0;
61                          else if (Right)
62                                  nextState = S0;
63                          else
64                                  nextState = S4;
65                          end
66                  S5: begin
67          nextState = S5;
68          end
69
70                  default: begin
71                          nextState = S0;
72                  end
73      endcase
74    end
75
76          assign Locked = (state == S5) ? 0:1;
77
78          always@ (posedge Clk)
79                  if (South)
80                          state <= S0;
81                  else
82                          state <= nextState;
83  endmodule // combination_lock_fsm
```

This module was then synthesized, along with all the rest, onto the FBGA board.

Results

Experiment 1

Below are the results of the the tests as well as the waveform for the combination lock FSM Verilog design.

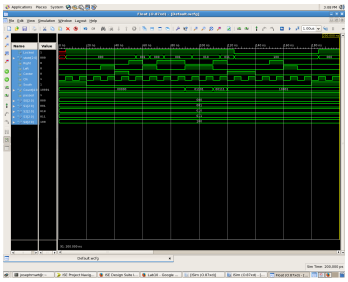

Figure 2: *Combination Lock FMS Waveform*

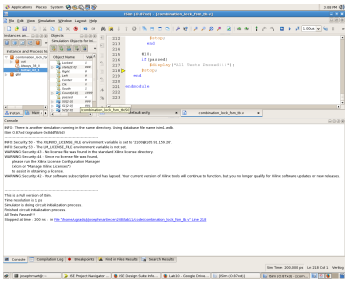

Figure 3: *Combination Lock FMS Test Results*

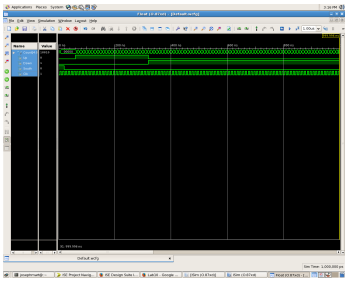

Figure 4: *Up/Down Counter Waveform*

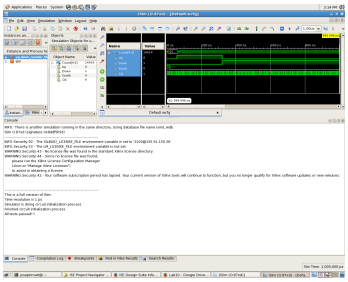

Figure 5: *Up/Down Counter Test Results*

Conclusion

Questions

1. **Include the source code with comments for all modules you simulated and/or implemented in lab. You do not have to include test bench code that was provided! Code without comments will not be accepted!**

2. **Include screenshots of all waveforms captured during simulation in addition to the test bench console output for each test bench simulation.**

3. **Answer all questions throughout the lab manual.**

4. **A possible attack on your combination-lock is a brute-force attack in which every possible input combination is tried. Given the original design with a combination of three numbers between 0 and 19, how many**

possible input combinations exist? How about for the modified design with a combination of four numbers?

Student Feedback

1. **What did you like most about the lab assignment and why? What did you like least about it and why?**

2. **Were there any section of the lab manual that were unclear? If so, what was unclear? Do you have any suggestions for improving the clarity?**

3. **What suggestions do you have to improve the overall lab assignment?**