



Ain shams university
Faculty of engineering
Computer engineering and software systems programs

Course: EDA (CSE215)

Project 2

Submitted by: Shehab Ahmed Hassan Kotb

Id: 16p6014

Email : shehabktob@gmail.com

Submitted to: Dr. Mohamed Dessouky
Eng. Michael Hany

Table of Contents

1.0 Introduction	2
2.0 boom comparison	3
3.0 loon comparison	3
4.0 XSCH visualization	4
5.0 Netlist checking.....	4
6.0 delay simulation (behavioral & structural)	5
7.0 DFT	5
8.0 DFT simulation	6
Appendix	7
Flatbeh logs	7
Proof logs	8
.path file	9

1.0 Introduction

This document covers the low-level synthesis of a Digital access control system, it's high level design was already covered in the previous document.

The low-level synthesis is achieved by using alliance tool such as syf, boom, boog, loon; the output of these tool is detailed bellow.

2.0 boom comparison

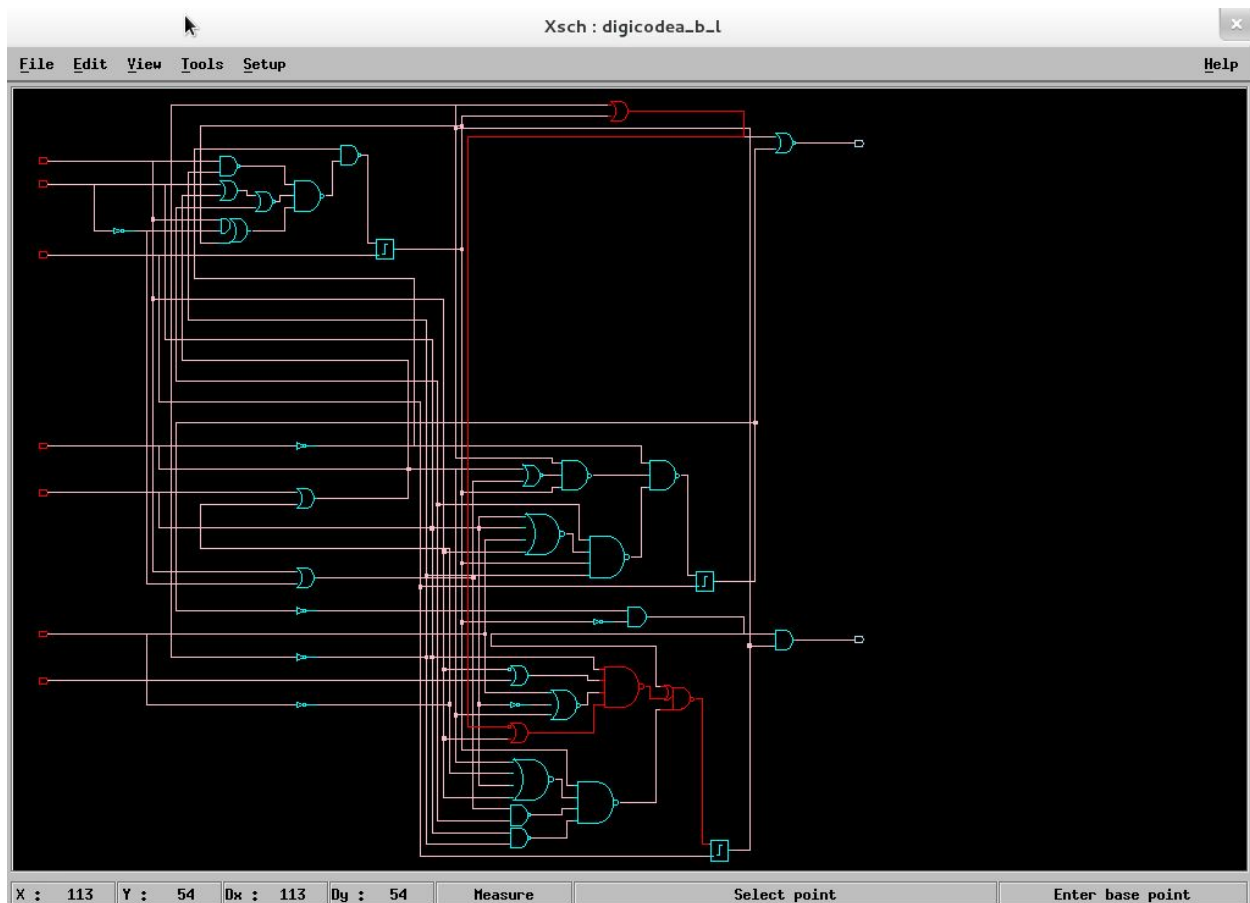
encoding	before boom (literals)	after boom (literals)
a	137	54
j	138	56
m	137	56
o	122	70
r	138	70

3.0 loon comparison

encoding	before loon (delay ps)	after loon (delay ps)	before loon (area λ^2)	after loon (area λ^2)
a	1808	1783	51250	51250
j	1858	1836	52500	52750
m	2229	2119	50750	53000
o	2919	2805	81000	85000
r	2651	2536	68000	70000

We will choose to continue with the a encoding as we can see it has the lowest area and the lowest delay making it the best among all the other encodings.

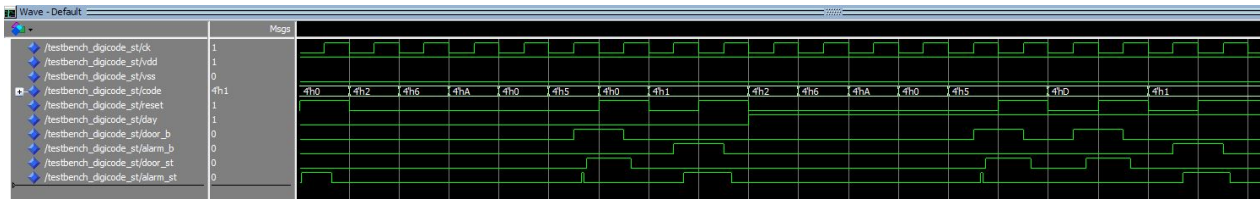
4.0 XSCH visualization



5.0 Netlist checking

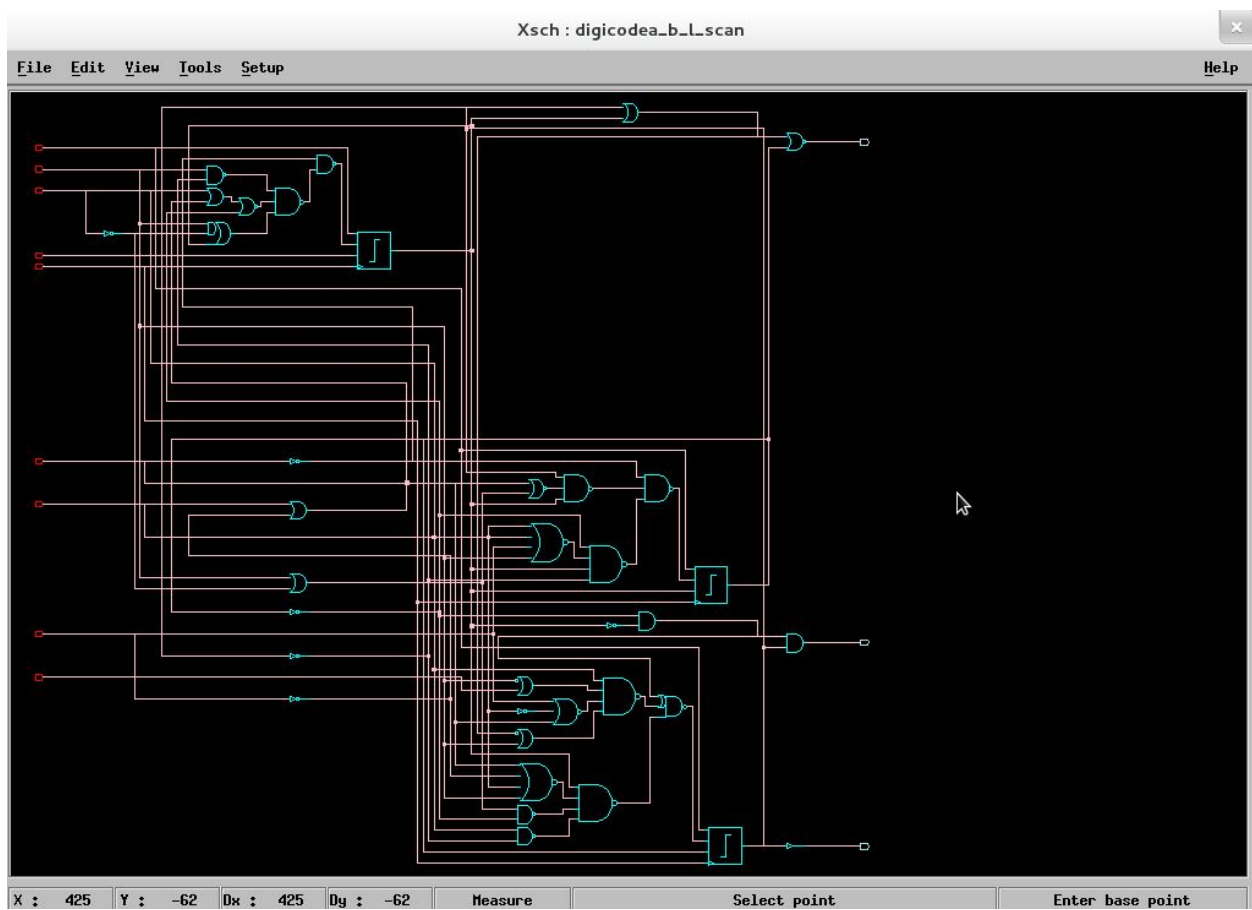
The log files of flatbeh & proof are in the appendix.

6.0 delay simulation (behavioral & structural)



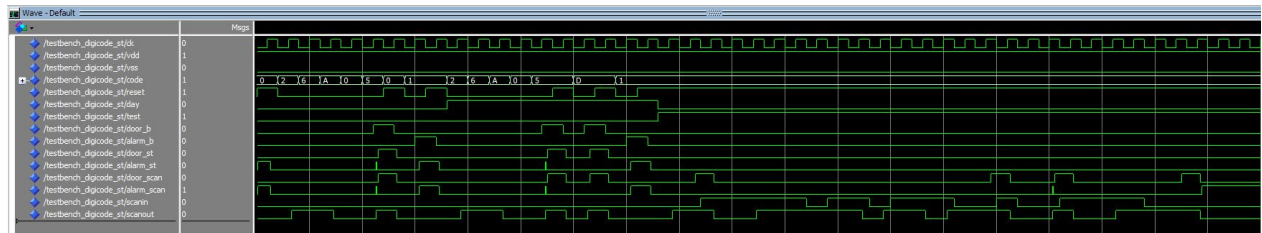
The test are the same as last project, except a reset case added at the begging to stabilize initial outputs; from the graph we can see that the structural output follow the behavioral outputs by a slight delay.

7.0 DFT



The .path file is attached in appendix.

8.0 DFT simulation



The code is in hexadecimal, D in hexadecimal is (1101) in binary which is the encoding given for the O button; after the normal testing is done test becomes 1 and scanout follows scanin with delay of 3 clock cycles because we have three flipflops the result takes 3 cycles to propagate from scanin to scanout.

Appendix

Flatbeh logs

```
1
2      00000000 0000      00000000      000
3      00 0 00      0 00 00      00
4      00 0 00      00 00 00      00
5      00      00 0000 00 00 00 0000 00 000
6      00 0 00 00 0 0000000 00 00 0 0 000 00
7      000000 00 00 00 00 00 000000 00 00 00
8      00 0 00 00000 00 00 00 00000000 00 00
9      00      00 00 00 00 00 00 00 00 00 00
10     00      00 00 00 00 00 00 00 00 00 00
11     00      00 00 00 00 00 0 00 00 00 00 00
12     000000 000000 0000 00 0000 00000000 0000 0000 0000
13
14      a netlist abstractor
15
16      Alliance CAD System 5.0 20090901, flatbeh 5.0 [2000/11/01]
17      Copyright (c) 1993-2019, ASIM/LIP6/UPMC
18      Author(s): François DONNET, Huu Nghia VUONG
19      E-mail : alliance-users@asim.lip6.fr
20
21     ===== Environnement =====
22     MBK WORK LIB = .
23     MBK CATA LIB =
24     ./usr/lib64/alliance/cells/sxlib:/usr/lib64/alliance/cells/dp sxlib:/usr/lib64/alli
25     ance/cells/rflib:/usr/lib64/alliance/cells/rf2lib:/usr/lib64/alliance/cells/ramlib:/
26     usr/lib64/alliance/cells/romlib:/usr/lib64/alliance/cells/pwlib:/usr/lib64/alliance/
27     cells/padlib
28     MBK CATAL NAME = CATAL
29
30     ===== Files =====
31     Netlist file = digicodea b 1.vst
32     Output file = digicodea_b_1_net.vbe
33
34     Loading './digicodea b 1.vst'
35     flattening figure digicodea b 1
36     loading oa22_x2
37     loading o2_x2
38     loading na3_x1
39     loading na2_x1
40     loading no4_x1
41     loading inv_x2
42     loading no3_x1
43     loading on12_x1
44     loading na4_x1
45     loading nao22_x1
46     loading sff1_x4
47     loading a2_x2
48     loading no2_x1
49     Restoring array's orders
50     BEH : Saving 'digicodea b 1 net' in a vhdl file (vbe)
```


Proof logs

```

1
2      0000000          000
3      00   00        00  00
4      00   00        00  00
5      00   00 000 000    000    000    00
6      00  00   000 00 00  00  00  00  00000000
7      000000    00  00  00  00  00  00  00
8      00         00     00     00  00     00  00
9      00         00     00     00  00     00  00
10     00         00     00     00  00     00  00
11     00         00     00     00  00     00  00
12     000000    0000    000    000    000000
13
14              Formal Proof
15
16              Alliance CAD System 5.0 20090901, proof 5.0
17              Copyright (c) 1990-2019, ASIM/LIP6/UPMC
18              E-mail       : alliance-users@asim.lip6.fr
19
20  ===== Environment =====
21  MBK WORK LIB      = .
22  MBK CATA LIB      =
23  ./usr/lib64/alliance/cells/sxlib:/usr/lib64/alliance/cells/dp_sxlib:/usr/lib64/alliance/
24  cells/rflib:/usr/lib64/alliance/cells/rf2lib:/usr/lib64/alliance/cells/ramlib:/usr/lib64/
25  alliance/cells/romlib:/usr/lib64/alliance/cells/pwlib:/usr/lib64/alliance/cells/padlib
26  ===== Files, Options and Parameters =====
27  First VHDL file   = digicodea.vbe
28  Second VHDL file  = digicodea b 1 net.vbe
29  The auxiliary signals are erased
30  Errors are displayed
31  =====
32
33  Compiling 'digicodea' ...
34  Compiling 'digicodea b 1 net' ...
35  ---> final number of nodes = 287(139)
36
37  Running Abl2Bdd on `digicodea b 1 net`
38  -----
39
40              Formal proof with Ordered Binary Decision Diagrams between
41
42              './digicodea' and './digicodea b 1 net'
43  -----
44
45              PRIMARY OUTPUT =====
46              AUXILIARY SIGNAL =====
47              REGISTER SIGNAL =====
48              EXTERNAL BUS =====
49              INTERNAL BUS =====
50
51              Formal Proof : OK
52
53  PFFFFFFFFFFFFFFFFFFFFFFFFFFFFXXXXXXXXXXXXXCCCCCCCCCCCCCCCCCCCCCCCCCCOFFFFFFFFFFFFFFF

```

.path file

```
1 BEGIN_PATH_REG
2 digicode cs 0 ins
3 digicode_cs_1_ins
4 digicode cs 2 ins
5 END_PATH_REG
6
7 BEGIN_CONNECTOR
8 SCAN_IN scanin
9 SCAN_OUT scanout
10 SCAN_TEST test
11 END_CONNECTOR
12
```

```

1  -----original behavioral fsm-----
2
3  library IEEE;
4  USE ieee.std_logic_1164.ALL;
5
6  entity DigiCode is
7  port (
8      ck      : in  bit;
9      vdd     : in  bit;
10     vss     : in  bit;
11     code    : in  bit_vector (3 downto 0);
12     reset   : in  bit;
13     day     : in  bit;
14     alarm   : out bit;
15     door    : out bit
16 );
17 end DigiCode;
18
19 architecture MOORE of DigiCode is
20     type STATE_TYPE is (S0, S1, s2, s3, s4, s5, s6);
21     signal NS, CS : STATE_TYPE;
22
23 begin
24     process (CS, reset, code, day)
25     begin
26
27         if (reset='1') then
28             NS<=S0;
29         else
30             case CS is
31                 when S0 =>
32                     if (day='1' and code = "1101") then --code = 0
33                         NS <= S5;
34                     elsif (code = x"2") then
35                         NS <= s1;
36                     else
37                         Ns <= s6;
38                     end if;
39
40                 when S1 =>
41                     if (day='1' and code = "1101") then --code = 0
42                         NS <= S5;
43                     elsif (code = x"6") then
44                         NS <= s2;
45                     else
46                         Ns <= s6;
47                     end if;
48
49
50                 when S2 =>
51                     if (day='1' and code = "1101") then --code = 0
52                         NS <= S5;
53                     elsif (code = "1010") then
54                         NS <= s3;
55                     else
56                         Ns <= s6;
57                     end if;
58
59
60                 when S3 =>
61                     if (day='1' and code = "1101") then --code = 0
62                         NS <= S5;
63                     elsif (code = x"0") then
64                         NS <= s4;
65                     else
66                         Ns <= s6;

```

```

67         end if;
68
69
70     when S4 =>
71         if (day='1' and code = "1101") then --code = o
72             NS <= S5;
73         elsif (code = x"5") then
74             NS <= s5;
75         else
76             Ns <= s6;
77         end if;
78
79
80     when S5 =>
81         --door <= '1';
82         null;
83         --alarm <= '0';
84         -- if (day='1' and code == '1100') then --code = o
85             -- NS <= S5;
86         -- elsif (code == '0')
87             -- NS <= s4;
88         -- else
89             -- Ns <= s6;
90         -- end if;
91
92
93     when S6 =>
94         --door <= '0';
95         --alarm <= '1';
96         null;
97         -- if (day='1' and code == '1100') then --code = o
98             -- NS <= S6;
99         -- elsif (code == '0')
100             -- NS <= s5;
101         -- else
102             -- Ns <= s6;
103         -- end if;
104
105     when others => assert(false)
106     report "illegal state" severity error;
107
108
109     end case;
110 end if;
111
112 case cs is
113     when s0 =>
114         door <= '0';
115         alarm <= '0';
116     when s1 =>
117         door <= '0';
118         alarm <= '0';
119     when s2 =>
120         door <= '0';
121         alarm <= '0';
122     when s3 =>
123         door <= '0';
124         alarm <= '0';
125     when s4 =>
126         door <= '0';
127         alarm <= '0';
128     when s5 =>
129         door <= '1';
130         alarm <= '0';
131     when s6 =>
132         door <= '0';

```

```
133     alarm <= '1';
134     end case;
135 end process;
136
137 -- Process (2): State update (sequential)
138 process(ck)
139 begin
140     if(ck = '1' and not ck'stable)then
141         CS <= NS;
142     end if;
143 end process;
144
145 end MOORE;
146
```

```

1  -----structural output from loon-----
2
3  LIBRARY sxlib_ModelSim;
4
5  entity digicodea_b_1 is
6      port (
7          ck      : in      bit;
8          vdd     : in      bit;
9          vss     : in      bit;
10         code    : in      bit_vector(3 downto 0);
11         reset   : in      bit;
12         day     : in      bit;
13         alarm   : out     bit;
14         door    : out     bit
15     );
16 end digicodea_b_1;
17
18 architecture structural of digicodea_b_1 is
19     Component oa22_x2
20     port (
21         i0  : in      bit;
22         i1  : in      bit;
23         i2  : in      bit;
24         q   : out     bit;
25         vdd : in      bit;
26         vss : in      bit
27     );
28 end component;
29
30     Component o2_x2
31     port (
32         i0  : in      bit;
33         i1  : in      bit;
34         q   : out     bit;
35         vdd : in      bit;
36         vss : in      bit
37     );
38 end component;
39
40     Component na3_x1
41     port (
42         i0  : in      bit;
43         i1  : in      bit;
44         i2  : in      bit;
45         nq  : out     bit;
46         vdd : in      bit;
47         vss : in      bit
48     );
49 end component;
50
51     Component na2_x1
52     port (
53         i0  : in      bit;
54         i1  : in      bit;
55         nq  : out     bit;
56         vdd : in      bit;
57         vss : in      bit
58     );
59 end component;
60
61     Component no4_x1
62     port (
63         i0  : in      bit;
64         i1  : in      bit;
65         i2  : in      bit;
66         i3  : in      bit;

```

```

67         nq  : out    bit;
68         vdd : in     bit;
69         vss : in     bit
70     );
71 end component;
72
73 Component inv_x2
74     port (
75         i    : in     bit;
76         nq   : out    bit;
77         vdd  : in     bit;
78         vss  : in     bit
79     );
80 end component;
81
82 Component no3_x1
83     port (
84         i0   : in     bit;
85         i1   : in     bit;
86         i2   : in     bit;
87         nq   : out    bit;
88         vdd  : in     bit;
89         vss  : in     bit
90     );
91 end component;
92
93 Component on12_x1
94     port (
95         i0   : in     bit;
96         i1   : in     bit;
97         q    : out    bit;
98         vdd  : in     bit;
99         vss  : in     bit
100    );
101 end component;
102
103 Component na4_x1
104     port (
105         i1   : in     bit;
106         i0   : in     bit;
107         i3   : in     bit;
108         i2   : in     bit;
109         nq   : out    bit;
110         vdd  : in     bit;
111         vss  : in     bit
112    );
113 end component;
114
115 Component nao22_x1
116     port (
117         i1   : in     bit;
118         i0   : in     bit;
119         i2   : in     bit;
120         nq   : out    bit;
121         vdd  : in     bit;
122         vss  : in     bit
123    );
124 end component;
125
126 Component sff1_x4
127     port (
128         ck   : in     bit;
129         i    : in     bit;
130         q    : out    bit;
131         vdd  : in     bit;
132         vss  : in     bit

```

```

133     );
134 end component;
135
136 Component a2_x2
137     port (
138         i0  : in      bit;
139         i1  : in      bit;
140         q   : out     bit;
141         vdd : in      bit;
142         vss : in      bit
143     );
144 end component;
145
146 Component no2_x1
147     port (
148         i0  : in      bit;
149         i1  : in      bit;
150         nq  : out     bit;
151         vdd : in      bit;
152         vss : in      bit
153     );
154 end component;
155
156 signal digicode_cs      : bit_vector( 2 downto 0);
157 signal not_code         : bit_vector( 2 downto 1);
158 signal not_digicode_cs  : bit_vector( 2 downto 0);
159 signal on12_x1_sig      : bit;
160 signal on12_x1_2_sig    : bit;
161 signal oa22_x2_sig      : bit;
162 signal o2_x2_sig        : bit;
163 signal not_reset        : bit;
164 signal not_aux3         : bit;
165 signal not_aux2         : bit;
166 signal not_aux1         : bit;
167 signal not_aux0         : bit;
168 signal no4_x1_sig       : bit;
169 signal no4_x1_2_sig     : bit;
170 signal no3_x1_sig       : bit;
171 signal no2_x1_sig       : bit;
172 signal no2_x1_2_sig     : bit;
173 signal nao22_x1_sig     : bit;
174 signal na4_x1_sig       : bit;
175 signal na4_x1_3_sig     : bit;
176 signal na4_x1_2_sig     : bit;
177 signal na3_x1_sig       : bit;
178 signal na3_x1_3_sig     : bit;
179 signal na3_x1_2_sig     : bit;
180 signal na2_x1_sig       : bit;
181 signal na2_x1_4_sig     : bit;
182 signal na2_x1_3_sig     : bit;
183 signal na2_x1_2_sig     : bit;
184 signal inv_x2_sig       : bit;
185
186 begin
187
188 not_aux0_ins : o2_x2
189     port map (
190         i0  => digicode_cs(2),
191         i1  => digicode_cs(0),
192         q   => not_aux0,
193         vdd => vdd,
194         vss => vss
195     );
196
197 not_aux2_ins : a2_x2
198     port map (

```



```

199         i0 => not_digicode_cs(1),
200         i1 => not_digicode_cs(0),
201         q  => not_aux2,
202         vdd => vdd,
203         vss => vss
204     );
205
206 not_aux1_ins : o2_x2
207     port map (
208         i0 => code(3),
209         i1 => not_code(2),
210         q  => not_aux1,
211         vdd => vdd,
212         vss => vss
213     );
214
215 not_aux3_ins : o2_x2
216     port map (
217         i0 => code(0),
218         i1 => not_code(1),
219         q  => not_aux3,
220         vdd => vdd,
221         vss => vss
222     );
223
224 not_digicode_cs_1_ins : inv_x2
225     port map (
226         i  => digicode_cs(1),
227         nq => not_digicode_cs(1),
228         vdd => vdd,
229         vss => vss
230     );
231
232 not_digicode_cs_2_ins : inv_x2
233     port map (
234         i  => digicode_cs(2),
235         nq => not_digicode_cs(2),
236         vdd => vdd,
237         vss => vss
238     );
239
240 not_digicode_cs_0_ins : inv_x2
241     port map (
242         i  => digicode_cs(0),
243         nq => not_digicode_cs(0),
244         vdd => vdd,
245         vss => vss
246     );
247
248 not_code_2_ins : inv_x2
249     port map (
250         i  => code(2),
251         nq => not_code(2),
252         vdd => vdd,
253         vss => vss
254     );
255
256 not_code_1_ins : inv_x2
257     port map (
258         i  => code(1),
259         nq => not_code(1),
260         vdd => vdd,
261         vss => vss
262     );
263
264 not_reset_ins : inv_x2

```

```

265     port map (
266         i    => reset,
267         nq   => not_reset,
268         vdd  => vdd,
269         vss  => vss
270     );
271
272     oa22_x2_ins : oa22_x2
273     port map (
274         i0    => code(3),
275         i1    => not_code(2),
276         i2    => digicode_cs(0),
277         q     => oa22_x2_sig,
278         vdd   => vdd,
279         vss   => vss
280     );
281
282     o2_x2_ins : o2_x2
283     port map (
284         i0    => code(2),
285         i1    => not_aux3,
286         q     => o2_x2_sig,
287         vdd   => vdd,
288         vss   => vss
289     );
290
291     no2_x1_ins : no2_x1
292     port map (
293         i0    => o2_x2_sig,
294         i1    => not_digicode_cs(1),
295         nq    => no2_x1_sig,
296         vdd   => vdd,
297         vss   => vss
298     );
299
300     na2_x1_2_ins : na2_x1
301     port map (
302         i0    => code(3),
303         i1    => not_digicode_cs(2),
304         nq    => na2_x1_2_sig,
305         vdd   => vdd,
306         vss   => vss
307     );
308
309     na3_x1_ins : na3_x1
310     port map (
311         i0    => na2_x1_2_sig,
312         i1    => no2_x1_sig,
313         i2    => oa22_x2_sig,
314         nq    => na3_x1_sig,
315         vdd   => vdd,
316         vss   => vss
317     );
318
319     na2_x1_ins : na2_x1
320     port map (
321         i0    => not_reset,
322         i1    => na3_x1_sig,
323         nq    => na2_x1_sig,
324         vdd   => vdd,
325         vss   => vss
326     );
327
328     digicode_cs_0_ins : sffl_x4
329     port map (
330         ck    => ck,

```

```

331         i    => na2_x1_sig,
332         q    => digicode_cs(0),
333         vdd  => vdd,
334         vss  => vss
335     );
336
337     no4_x1_ins : no4_x1
338     port map (
339         i0  => code(0),
340         i1  => code(2),
341         i2  => code(1),
342         i3  => code(3),
343         nq  => no4_x1_sig,
344         vdd  => vdd,
345         vss  => vss
346     );
347
348     na4_x1_ins : na4_x1
349     port map (
350         i0  => not_digicode_cs(1),
351         i1  => no4_x1_sig,
352         i2  => digicode_cs(0),
353         i3  => not_digicode_cs(2),
354         nq  => na4_x1_sig,
355         vdd  => vdd,
356         vss  => vss
357     );
358
359     no2_x1_2_ins : no2_x1
360     port map (
361         i0  => not_aux3,
362         i1  => not_aux1,
363         nq  => no2_x1_2_sig,
364         vdd  => vdd,
365         vss  => vss
366     );
367
368     na3_x1_3_ins : na3_x1
369     port map (
370         i0  => digicode_cs(2),
371         i1  => no2_x1_2_sig,
372         i2  => digicode_cs(0),
373         nq  => na3_x1_3_sig,
374         vdd  => vdd,
375         vss  => vss
376     );
377
378     na3_x1_2_ins : na3_x1
379     port map (
380         i0  => not_reset,
381         i1  => na3_x1_3_sig,
382         i2  => na4_x1_sig,
383         nq  => na3_x1_2_sig,
384         vdd  => vdd,
385         vss  => vss
386     );
387
388     digicode_cs_1_ins : sff1_x4
389     port map (
390         ck  => ck,
391         i   => na3_x1_2_sig,
392         q   => digicode_cs(1),
393         vdd  => vdd,
394         vss  => vss
395     );
396

```

```

397 na2_x1_3_ins : na2_x1
398     port map (
399         i0  => code(2),
400         i1  => not_digicode_cs(2),
401         nq  => na2_x1_3_sig,
402         vdd => vdd,
403         vss => vss
404     );
405
406 na2_x1_4_ins : na2_x1
407     port map (
408         i0  => not_aux1,
409         i1  => not_digicode_cs(1),
410         nq  => na2_x1_4_sig,
411         vdd => vdd,
412         vss => vss
413     );
414
415 no4_x1_2_ins : no4_x1
416     port map (
417         i0  => reset,
418         i1  => not_code(1),
419         i2  => code(0),
420         i3  => code(3),
421         nq  => no4_x1_2_sig,
422         vdd => vdd,
423         vss => vss
424     );
425
426 na4_x1_2_ins : na4_x1
427     port map (
428         i0  => digicode_cs(0),
429         i1  => no4_x1_2_sig,
430         i2  => na2_x1_4_sig,
431         i3  => na2_x1_3_sig,
432         nq  => na4_x1_2_sig,
433         vdd => vdd,
434         vss => vss
435     );
436
437 inv_x2_ins : inv_x2
438     port map (
439         i    => code(0),
440         nq   => inv_x2_sig,
441         vdd  => vdd,
442         vss  => vss
443     );
444
445 no3_x1_ins : no3_x1
446     port map (
447         i0  => code(1),
448         i1  => inv_x2_sig,
449         i2  => reset,
450         nq  => no3_x1_sig,
451         vdd => vdd,
452         vss => vss
453     );
454
455 on12_x1_ins : on12_x1
456     port map (
457         i0  => not_aux0,
458         i1  => code(3),
459         q   => on12_x1_sig,
460         vdd => vdd,
461         vss => vss
462     );

```

```

463
464 onl2_x1_2_ins : onl2_x1
465     port map (
466         i0  => code(3),
467         i1  => day,
468         q   => onl2_x1_2_sig,
469         vdd => vdd,
470         vss => vss
471     );
472
473 na4_x1_3_ins : na4_x1
474     port map (
475         i1  => onl2_x1_2_sig,
476         i0  => code(2),
477         i3  => onl2_x1_sig,
478         i2  => no3_x1_sig,
479         nq  => na4_x1_3_sig,
480         vdd => vdd,
481         vss => vss
482     );
483
484 nao22_x1_ins : nao22_x1
485     port map (
486         i1  => na4_x1_3_sig,
487         i0  => not_aux2,
488         i2  => na4_x1_2_sig,
489         nq  => nao22_x1_sig,
490         vdd => vdd,
491         vss => vss
492     );
493
494 digicode_cs_2_ins : sffl_x4
495     port map (
496         ck  => ck,
497         i   => nao22_x1_sig,
498         q   => digicode_cs(2),
499         vdd => vdd,
500         vss => vss
501     );
502
503 door_ins : a2_x2
504     port map (
505         i0  => not_aux2,
506         i1  => digicode_cs(2),
507         q   => door,
508         vdd => vdd,
509         vss => vss
510     );
511
512 alarm_ins : no2_x1
513     port map (
514         i0  => not_aux0,
515         i1  => digicode_cs(1),
516         nq  => alarm,
517         vdd => vdd,
518         vss => vss
519     );
520
521
522 end structural;
523

```

```

1  -----structural output from scapin-----
2
3  LIBRARY sxlib_ModelSim;
4
5  entity digicodea_b_l_scan is
6      port (
7          ck      : in      bit;
8          vdd      : in      bit;
9          vss      : in      bit;
10         code     : in      bit_vector(3 downto 0);
11         reset    : in      bit;
12         day      : in      bit;
13         alarm    : out     bit;
14         door     : out     bit;
15         scanin   : in      bit;
16         test     : in      bit;
17         scanout  : out     bit
18     );
19 end digicodea_b_l_scan;
20
21 architecture structural of digicodea_b_l_scan is
22 Component oa22_x2
23     port (
24         i0 : in      bit;
25         i1 : in      bit;
26         i2 : in      bit;
27         q  : out     bit;
28         vdd : in     bit;
29         vss : in     bit
30     );
31 end component;
32
33 Component o2_x2
34     port (
35         i0 : in      bit;
36         i1 : in      bit;
37         q  : out     bit;
38         vdd : in     bit;
39         vss : in     bit
40     );
41 end component;
42
43 Component na3_x1
44     port (
45         i0 : in      bit;
46         i1 : in      bit;
47         i2 : in      bit;
48         nq : out     bit;
49         vdd : in     bit;
50         vss : in     bit
51     );
52 end component;
53
54 Component na2_x1
55     port (
56         i0 : in      bit;
57         i1 : in      bit;
58         nq : out     bit;
59         vdd : in     bit;
60         vss : in     bit
61     );
62 end component;
63
64 Component no4_x1
65     port (
66         i0 : in      bit;

```

```

67         i1  : in      bit;
68         i2  : in      bit;
69         i3  : in      bit;
70         nq  : out     bit;
71         vdd  : in      bit;
72         vss  : in      bit
73     );
74 end component;
75
76 Component inv_x2
77     port (
78         i    : in      bit;
79         nq   : out     bit;
80         vdd  : in      bit;
81         vss  : in      bit
82     );
83 end component;
84
85 Component no3_x1
86     port (
87         i0   : in      bit;
88         i1   : in      bit;
89         i2   : in      bit;
90         nq   : out     bit;
91         vdd  : in      bit;
92         vss  : in      bit
93     );
94 end component;
95
96 Component on12_x1
97     port (
98         i0   : in      bit;
99         i1   : in      bit;
100        q    : out     bit;
101        vdd  : in      bit;
102        vss  : in      bit
103    );
104 end component;
105
106 Component na4_x1
107     port (
108         i1   : in      bit;
109         i0   : in      bit;
110         i3   : in      bit;
111         i2   : in      bit;
112         nq   : out     bit;
113         vdd  : in      bit;
114         vss  : in      bit
115    );
116 end component;
117
118 Component nao22_x1
119     port (
120         i1   : in      bit;
121         i0   : in      bit;
122         i2   : in      bit;
123         nq   : out     bit;
124         vdd  : in      bit;
125         vss  : in      bit
126    );
127 end component;
128
129 Component a2_x2
130     port (
131         i0   : in      bit;
132         i1   : in      bit;

```

```

133         q    : out    bit;
134         vdd  : in     bit;
135         vss  : in     bit
136     );
137 end component;
138
139 Component no2_x1
140     port (
141         i0   : in     bit;
142         i1   : in     bit;
143         nq   : out    bit;
144         vdd  : in     bit;
145         vss  : in     bit
146     );
147 end component;
148
149 Component sff2_x4
150     port (
151         ck   : in     bit;
152         cmd  : in     bit;
153         i0   : in     bit;
154         i1   : in     bit;
155         q    : out    bit;
156         vdd  : in     bit;
157         vss  : in     bit
158     );
159 end component;
160
161 Component buf_x2
162     port (
163         i    : in     bit;
164         q    : out    bit;
165         vdd  : in     bit;
166         vss  : in     bit
167     );
168 end component;
169
170 signal digicode_cs      : bit_vector( 2 downto 0);
171 signal not_code         : bit_vector( 2 downto 1);
172 signal not_digicode_cs : bit_vector( 2 downto 0);
173 signal on12_x1_sig      : bit;
174 signal on12_x1_2_sig    : bit;
175 signal oa22_x2_sig      : bit;
176 signal o2_x2_sig        : bit;
177 signal not_reset        : bit;
178 signal not_aux3         : bit;
179 signal not_aux2         : bit;
180 signal not_aux1         : bit;
181 signal not_aux0         : bit;
182 signal no4_x1_sig       : bit;
183 signal no4_x1_2_sig     : bit;
184 signal no3_x1_sig       : bit;
185 signal no2_x1_sig       : bit;
186 signal no2_x1_2_sig     : bit;
187 signal nao22_x1_sig     : bit;
188 signal na4_x1_sig       : bit;
189 signal na4_x1_3_sig     : bit;
190 signal na4_x1_2_sig     : bit;
191 signal na3_x1_sig       : bit;
192 signal na3_x1_3_sig     : bit;
193 signal na3_x1_2_sig     : bit;
194 signal na2_x1_sig       : bit;
195 signal na2_x1_4_sig     : bit;
196 signal na2_x1_3_sig     : bit;
197 signal na2_x1_2_sig     : bit;
198 signal inv_x2_sig       : bit;

```



```

199
200 begin
201
202 not_aux0_ins : o2_x2
203     port map (
204         i0 => digicode_cs(2),
205         i1 => digicode_cs(0),
206         q  => not_aux0,
207         vdd => vdd,
208         vss => vss
209     );
210
211 not_aux2_ins : a2_x2
212     port map (
213         i0 => not_digicode_cs(1),
214         i1 => not_digicode_cs(0),
215         q  => not_aux2,
216         vdd => vdd,
217         vss => vss
218     );
219
220 not_aux1_ins : o2_x2
221     port map (
222         i0 => code(3),
223         i1 => not_code(2),
224         q  => not_aux1,
225         vdd => vdd,
226         vss => vss
227     );
228
229 not_aux3_ins : o2_x2
230     port map (
231         i0 => code(0),
232         i1 => not_code(1),
233         q  => not_aux3,
234         vdd => vdd,
235         vss => vss
236     );
237
238 not_digicode_cs_1_ins : inv_x2
239     port map (
240         i  => digicode_cs(1),
241         nq => not_digicode_cs(1),
242         vdd => vdd,
243         vss => vss
244     );
245
246 not_digicode_cs_2_ins : inv_x2
247     port map (
248         i  => digicode_cs(2),
249         nq => not_digicode_cs(2),
250         vdd => vdd,
251         vss => vss
252     );
253
254 not_digicode_cs_0_ins : inv_x2
255     port map (
256         i  => digicode_cs(0),
257         nq => not_digicode_cs(0),
258         vdd => vdd,
259         vss => vss
260     );
261
262 not_code_2_ins : inv_x2
263     port map (
264         i  => code(2),

```

```

265         nq => not_code(2),
266         vdd => vdd,
267         vss => vss
268     );
269
270     not_code_1_ins : inv_x2
271     port map (
272         i    => code(1),
273         nq   => not_code(1),
274         vdd  => vdd,
275         vss  => vss
276     );
277
278     not_reset_ins : inv_x2
279     port map (
280         i    => reset,
281         nq   => not_reset,
282         vdd  => vdd,
283         vss  => vss
284     );
285
286     oa22_x2_ins : oa22_x2
287     port map (
288         i0   => code(3),
289         i1   => not_code(2),
290         i2   => digicode_cs(0),
291         q    => oa22_x2_sig,
292         vdd  => vdd,
293         vss  => vss
294     );
295
296     o2_x2_ins : o2_x2
297     port map (
298         i0   => code(2),
299         i1   => not_aux3,
300         q    => o2_x2_sig,
301         vdd  => vdd,
302         vss  => vss
303     );
304
305     no2_x1_ins : no2_x1
306     port map (
307         i0   => o2_x2_sig,
308         i1   => not_digicode_cs(1),
309         nq   => no2_x1_sig,
310         vdd  => vdd,
311         vss  => vss
312     );
313
314     na2_x1_2_ins : na2_x1
315     port map (
316         i0   => code(3),
317         i1   => not_digicode_cs(2),
318         nq   => na2_x1_2_sig,
319         vdd  => vdd,
320         vss  => vss
321     );
322
323     na3_x1_ins : na3_x1
324     port map (
325         i0   => na2_x1_2_sig,
326         i1   => no2_x1_sig,
327         i2   => oa22_x2_sig,
328         nq   => na3_x1_sig,
329         vdd  => vdd,
330         vss  => vss

```

```

331     );
332
333     na2_x1_ins : na2_x1
334     port map (
335         i0  => not_reset,
336         i1  => na3_x1_sig,
337         nq  => na2_x1_sig,
338         vdd => vdd,
339         vss => vss
340     );
341
342     no4_x1_ins : no4_x1
343     port map (
344         i0  => code(0),
345         i1  => code(2),
346         i2  => code(1),
347         i3  => code(3),
348         nq  => no4_x1_sig,
349         vdd => vdd,
350         vss => vss
351     );
352
353     na4_x1_ins : na4_x1
354     port map (
355         i1  => no4_x1_sig,
356         i0  => not_digicode_cs(1),
357         i3  => not_digicode_cs(2),
358         i2  => digicode_cs(0),
359         nq  => na4_x1_sig,
360         vdd => vdd,
361         vss => vss
362     );
363
364     no2_x1_2_ins : no2_x1
365     port map (
366         i0  => not_aux3,
367         i1  => not_aux1,
368         nq  => no2_x1_2_sig,
369         vdd => vdd,
370         vss => vss
371     );
372
373     na3_x1_3_ins : na3_x1
374     port map (
375         i0  => digicode_cs(2),
376         i1  => no2_x1_2_sig,
377         i2  => digicode_cs(0),
378         nq  => na3_x1_3_sig,
379         vdd => vdd,
380         vss => vss
381     );
382
383     na3_x1_2_ins : na3_x1
384     port map (
385         i0  => not_reset,
386         i1  => na3_x1_3_sig,
387         i2  => na4_x1_sig,
388         nq  => na3_x1_2_sig,
389         vdd => vdd,
390         vss => vss
391     );
392
393     na2_x1_3_ins : na2_x1
394     port map (
395         i0  => code(2),
396         i1  => not_digicode_cs(2),

```

```

397         nq => na2_x1_3_sig,
398         vdd => vdd,
399         vss => vss
400     );
401
402     na2_x1_4_ins : na2_x1
403     port map (
404         i0 => not_aux1,
405         i1 => not_digicode_cs(1),
406         nq => na2_x1_4_sig,
407         vdd => vdd,
408         vss => vss
409     );
410
411     no4_x1_2_ins : no4_x1
412     port map (
413         i0 => reset,
414         i1 => not_code(1),
415         i2 => code(0),
416         i3 => code(3),
417         nq => no4_x1_2_sig,
418         vdd => vdd,
419         vss => vss
420     );
421
422     na4_x1_2_ins : na4_x1
423     port map (
424         i1 => no4_x1_2_sig,
425         i0 => digicode_cs(0),
426         i3 => na2_x1_3_sig,
427         i2 => na2_x1_4_sig,
428         nq => na4_x1_2_sig,
429         vdd => vdd,
430         vss => vss
431     );
432
433     inv_x2_ins : inv_x2
434     port map (
435         i    => code(0),
436         nq   => inv_x2_sig,
437         vdd  => vdd,
438         vss  => vss
439     );
440
441     no3_x1_ins : no3_x1
442     port map (
443         i0 => code(1),
444         i1 => inv_x2_sig,
445         i2 => reset,
446         nq => no3_x1_sig,
447         vdd => vdd,
448         vss => vss
449     );
450
451     on12_x1_ins : on12_x1
452     port map (
453         i0 => not_aux0,
454         i1 => code(3),
455         q  => on12_x1_sig,
456         vdd => vdd,
457         vss => vss
458     );
459
460     on12_x1_2_ins : on12_x1
461     port map (
462         i0 => code(3),

```

```

463         i1 => day,
464         q  => onl2_x1_2_sig,
465         vdd => vdd,
466         vss => vss
467     );
468
469     na4_x1_3_ins : na4_x1
470     port map (
471         i1 => onl2_x1_2_sig,
472         i0 => code(2),
473         i3 => onl2_x1_sig,
474         i2 => no3_x1_sig,
475         nq => na4_x1_3_sig,
476         vdd => vdd,
477         vss => vss
478     );
479
480     nao22_x1_ins : nao22_x1
481     port map (
482         i1 => na4_x1_3_sig,
483         i0 => not_aux2,
484         i2 => na4_x1_2_sig,
485         nq => nao22_x1_sig,
486         vdd => vdd,
487         vss => vss
488     );
489
490     door_ins : a2_x2
491     port map (
492         i0 => not_aux2,
493         i1 => digicode_cs(2),
494         q  => door,
495         vdd => vdd,
496         vss => vss
497     );
498
499     alarm_ins : no2_x1
500     port map (
501         i0 => not_aux0,
502         i1 => digicode_cs(1),
503         nq => alarm,
504         vdd => vdd,
505         vss => vss
506     );
507
508     digicode_cs_0_ins_scan_0 : sff2_x4
509     port map (
510         ck => ck,
511         cmd => test,
512         i0 => na2_x1_sig,
513         i1 => scanin,
514         q  => digicode_cs(0),
515         vdd => vdd,
516         vss => vss
517     );
518
519     digicode_cs_1_ins_scan_1 : sff2_x4
520     port map (
521         ck => ck,
522         cmd => test,
523         i0 => na3_x1_2_sig,
524         i1 => digicode_cs(0),
525         q  => digicode_cs(1),
526         vdd => vdd,
527         vss => vss
528     );

```

```
529
530 digicode_cs_2_ins_scan_2 : sff2_x4
531     port map (
532         ck  => ck,
533         cmd => test,
534         i0  => nao22_x1_sig,
535         i1  => digicode_cs(1),
536         q   => digicode_cs(2),
537         vdd => vdd,
538         vss => vss
539     );
540
541 buf_scan_3 : buf_x2
542     port map (
543         i   => digicode_cs(2),
544         q   => scanout,
545         vdd => vdd,
546         vss => vss
547     );
548
549
550 end structural;
551
```

```

1  -----test bench of behavioral, loon output, scapin output-----
2
3  library IEEE;
4  USE ieee.std_logic_1164.ALL;
5
6  -- Entity declaration for your testbench. Don't declare any ports here
7  ENTITY testbench_DigiCode_st IS
8  END ENTITY testbench_DigiCode_st;
9
10 ARCHITECTURE testbench_DigiCode_st OF testbench_DigiCode_st IS
11
12 -- Component Declaration for the Device Under Test (DUT)
13 COMPONENT DigiCode IS
14 port (
15     ck      : in  bit;
16     vdd     : in  bit;
17     vss     : in  bit;
18     code    : in  bit_vector (3 downto 0);
19     reset   : in  bit;
20     day     : in  bit;
21     alarm   : out bit;
22     door    : out bit
23 );
24 END COMPONENT DigiCode;
25
26 FOR dut_beh: DigiCode USE ENTITY WORK.DigiCode (MOORE);
27
28 COMPONENT digicodea_b_l IS
29 port (
30     ck      : in  bit;
31     vdd     : in  bit;
32     vss     : in  bit;
33     code    : in  bit_vector(3 downto 0);
34     reset   : in  bit;
35     day     : in  bit;
36     alarm   : out bit;
37     door    : out bit
38 );
39 END COMPONENT digicodea_b_l;
40
41 FOR dut_st: digicodea_b_l USE ENTITY WORK.digicodea_b_l (structural);
42
43 COMPONENT digicodea_b_l_scan IS
44 port (
45     ck      : in      bit;
46     vdd     : in      bit;
47     vss     : in      bit;
48     code    : in      bit_vector(3 downto 0);
49     reset   : in      bit;
50     day     : in      bit;
51     alarm   : out     bit;
52     door    : out     bit;
53     scanin  : in      bit;
54     test    : in      bit;
55     scanout : out     bit
56 );
57 END COMPONENT digicodea_b_l_scan;
58
59 FOR dut_scan: digicodea_b_l_scan USE ENTITY WORK.digicodea_b_l_scan (structural);
60
61
62 -- Declare input signals and initialize them
63 SIGNAL ck      : bit := '0';
64 SIGNAL vdd     : bit := '1';
65 SIGNAL vss     : bit := '0';

```

```

66 SIGNAL code : bit_vector(3 downto 0) := x"0";
67 SIGNAL reset : bit := '0';
68 SIGNAL day : bit := '0';
69
70 SIGNAL test : bit := '0';
71
72 SIGNAL door_b : bit := '0';
73 SIGNAL alarm_b : bit := '0';
74
75 SIGNAL door_st : bit := '0';
76 SIGNAL alarm_st : bit := '0';
77
78 SIGNAL door_scan : bit := '0';
79 SIGNAL alarm_scan : bit := '0';
80
81 SIGNAL scanin : bit := '0';
82 SIGNAL scanout : bit := '0';
83
84 -- Constants and Clock period definitions
85 constant clk_period : time := 20 ns;
86 constant sequence : bit_vector := "0011111011011100100111100";
87 BEGIN
88
89 -- Instantiate the Device Under Test (DUT)
90 dut_beh: DigiCode PORT MAP (ck, vdd, vss, code, reset, day, alarm_b, door_b);
91 dut_st: digicodea_b_l PORT MAP (ck, vdd, vss, code, reset, day, alarm_st, door_st);
92 dut_scan: digicodea_b_l_scan PORT MAP (ck, vdd, vss, code, reset, day, alarm_scan,
93 door_scan, scanin, test, scanout);
94
95 -- Clock process definitions( clock with 50% duty cycle )
96 clk_process :process
97 begin
98 ck <= '0';
99 wait for clk_period/2;
100 ck <= '1';
101 wait for clk_period/2;
102 end process;
103
104 -- Stimulus process, refer to clock signal
105 stim_proc: PROCESS IS
106 BEGIN
107 --reset case added to stabllise output at the beginnning
108 code <= x"0";
109 reset <= '1';
110 test <= '0';
111 WAIT FOR clk_period;
112 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
113 alarm_scan = '0' and door_scan = '0'
114 REPORT "output error"
115 SEVERITY error;
116
117 -- case 1 correct code entered at night "day = 0"
118 code <= x"2";
119 reset <= '0';
120 WAIT FOR clk_period;
121 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
122 alarm_scan = '0' and door_scan = '0'
123 REPORT "output error"
124 SEVERITY error;
125
126 code <= x"6";
127 WAIT FOR clk_period;
128 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
129 alarm_scan = '0' and door_scan = '0'
130 REPORT "output error"
131 SEVERITY error;

```



```

128
129 code <= x"a";
130 WAIT FOR clk_period;
131 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
132 REPORT "output error"
133 SEVERITY error;
134
135 code <= x"0";
136 WAIT FOR clk_period;
137 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
138 REPORT "output error"
139 SEVERITY error;
140
141 code <= x"5";
142 WAIT FOR clk_period;
143 ASSERT alarm_b = '0' and door_b = '1' and alarm_st = '0' and door_st = '1' and
alarm_scan = '0' and door_scan = '1'
144 REPORT "output error"
145 SEVERITY error;
146
147 --case 2 reset test
148 code <= x"0";
149 reset <= '1';
150 WAIT FOR clk_period;
151 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
152 REPORT "output error"
153 SEVERITY error;
154
155 --case 3 testing wrong code at night "day = 0"
156 code <= x"1";
157 reset <= '0';
158 WAIT FOR clk_period;
159 ASSERT alarm_b = '1' and door_b = '0' and alarm_st = '1' and door_st = '0' and
alarm_scan = '1' and door_scan = '0'
160 REPORT "output error"
161 SEVERITY error;
162
163 --returning to s0
164 reset <= '1';
165 WAIT FOR clk_period;
166 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
167 REPORT "output error"
168 SEVERITY error;
169
170 --case 4 testing correct code with day = 1
171 code <= x"2";
172 day <= '1';
173 reset <= '0';
174 WAIT FOR clk_period;
175 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
176 REPORT "output error"
177 SEVERITY error;
178
179 code <= x"6";
180 day <= '1';
181 reset <= '0';
182 WAIT FOR clk_period;
183 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
184 REPORT "output error"
185 SEVERITY error;

```

```

186
187 code <= x"a";
188 day <= '1';
189 reset <= '0';
190 WAIT FOR clk_period;
191 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
192 REPORT "output error"
193 SEVERITY error;
194
195 code <= x"0";
196 day <= '1';
197 reset <= '0';
198 WAIT FOR clk_period;
199 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
200 REPORT "output error"
201 SEVERITY error;
202
203 code <= x"5";
204 day <= '1';
205 reset <= '0';
206 WAIT FOR clk_period;
207 ASSERT alarm_b = '0' and door_b = '1' and alarm_st = '0' and door_st = '1' and
alarm_scan = '0' and door_scan = '1'
208 REPORT "output error"
209 SEVERITY error;
210
211 --case 5 testing rest when day = 1
212 reset <= '1';
213 WAIT FOR clk_period;
214 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
215 REPORT "output error"
216 SEVERITY error;
217
218 --case 6 testing "o" button when day = 1
219 code <= "1101";
220 day <= '1';
221 reset <= '0';
222 WAIT FOR clk_period;
223 ASSERT alarm_b = '0' and door_b = '1' and alarm_st = '0' and door_st = '1' and
alarm_scan = '0' and door_scan = '1'
224 REPORT "output error"
225 SEVERITY error;
226
227 --returning to s0
228 reset <= '1';
229 WAIT FOR clk_period;
230 ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
231 REPORT "output error"
232 SEVERITY error;
233
234 --case 6 wrong code when day = 1
235 code <= x"1";
236 day <= '1';
237 reset <= '0';
238 WAIT FOR clk_period;
239 ASSERT alarm_b = '1' and door_b = '0' and alarm_st = '1' and door_st = '0' and
alarm_scan = '1' and door_scan = '0'
240 REPORT "output error"
241 SEVERITY error;
242
243 --returning to s0
244 reset <= '1';

```

```
245     WAIT FOR clk_period;
246     ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
alarm_scan = '0' and door_scan = '0'
247     REPORT "output error"
248     SEVERITY error;
249
250     --case 7 test = 1
251     day <= '0';
252     test <= '1';
253     for i in 0 to sequence'length-1 loop
254         scanin <= sequence(i);
255         wait for clk_period;
256         if i>=2 then
257             ASSERT scanout=sequence(i-2)
258             REPORT "scanout does not follow scan in"
259             SEVERITY error;
260         end if;
261     end loop;
262
263
264
265     WAIT; -- stop process simulation run
266
267 END PROCESS stim_proc;
268 END ARCHITECTURE testbench_DigiCode_st;
269
270
271
```