

جامعة نيويورك أبوظبي



NYU ABU DHABI

**ADVANCED DIGITAL LOGIC
ENGR – UH 2310**

Lab 3 Part 2

**GROUP 1
SPRING, 2025**

This report is entirely our own work, and we have kept a copy for our own records. We are aware of the University's policies on cheating, plagiarism, and the resulting consequences of their breach.

Submitted by:

Name	Participation	Net ID
Damiane Kapanadze	Test Bench, Lab Report	dk4770
Seoyoon Jung	Code, Lab Report	sj4260
Sipan Hovsepian	Code, Lab Report	sh7437

Contents

1	Timing sequences	2
1.1	Normal mode	2
1.2	Normal mode with walk lights	2
1.3	Extended green phase	2
1.4	Extended green phase with walk lights	3
2	Devising the FSM	3
2.1	State diagram	3
2.2	Board setup	4
3	VHDL code of all modules	4
3.1	FSM module	4
3.2	TestBench Module	10
4	Testbench simulation results	13
4.1	All inputs off	13
4.2	Main sensor on (Main green extended)	13
4.3	Both walk buttons on	14
4.4	Both walk buttons and main sensor on	14
4.5	"X" error for main sensor	14
4.6	"X" error for both walk buttons	15
5	Constraints file	15
6	FPGA implementation	15

1 Timing sequences

For detailed timing data, refer to the spreadsheet: Google Sheets Link.

1.1 Normal mode

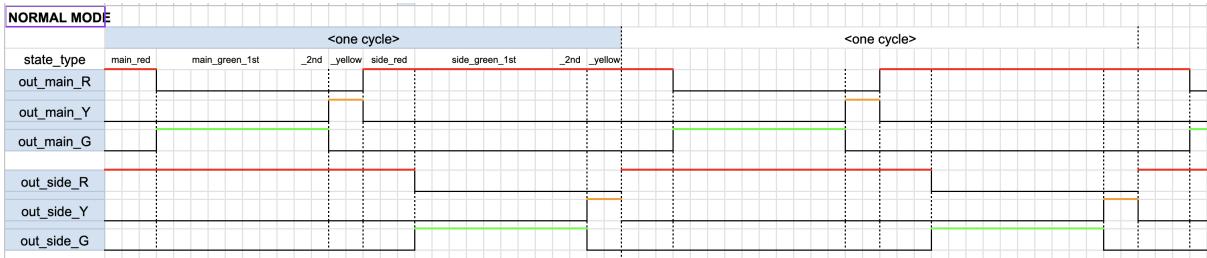


Figure 1: Normal mode

1.2 Normal mode with walk lights

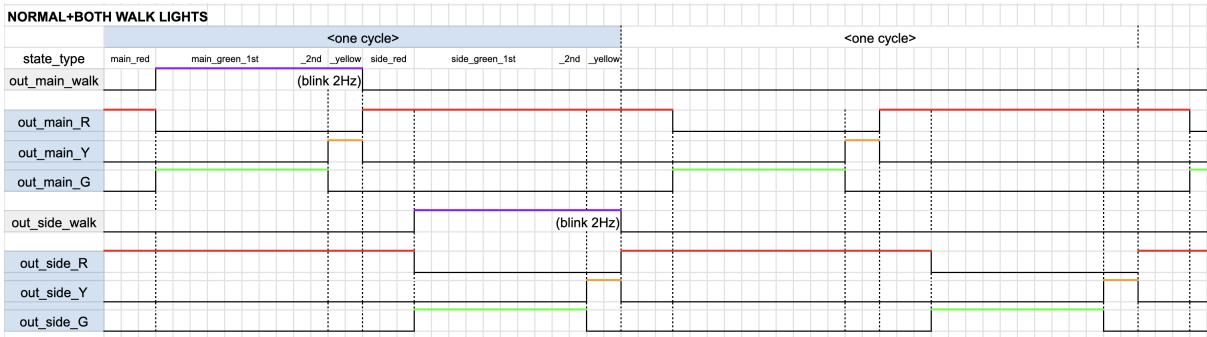


Figure 2: Normal mode with both walk lights triggered

1.3 Extended green phase



Figure 3: Extended green phase for main street

1.4 Extended green phase with walk lights

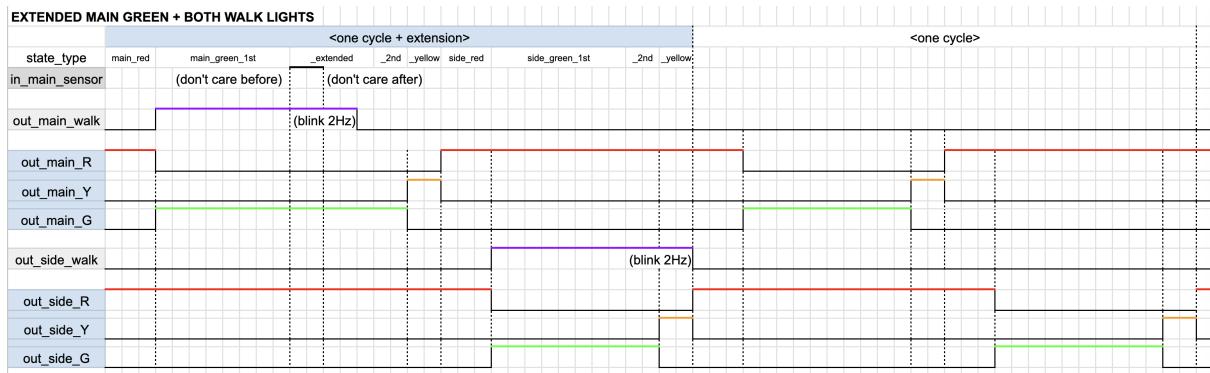
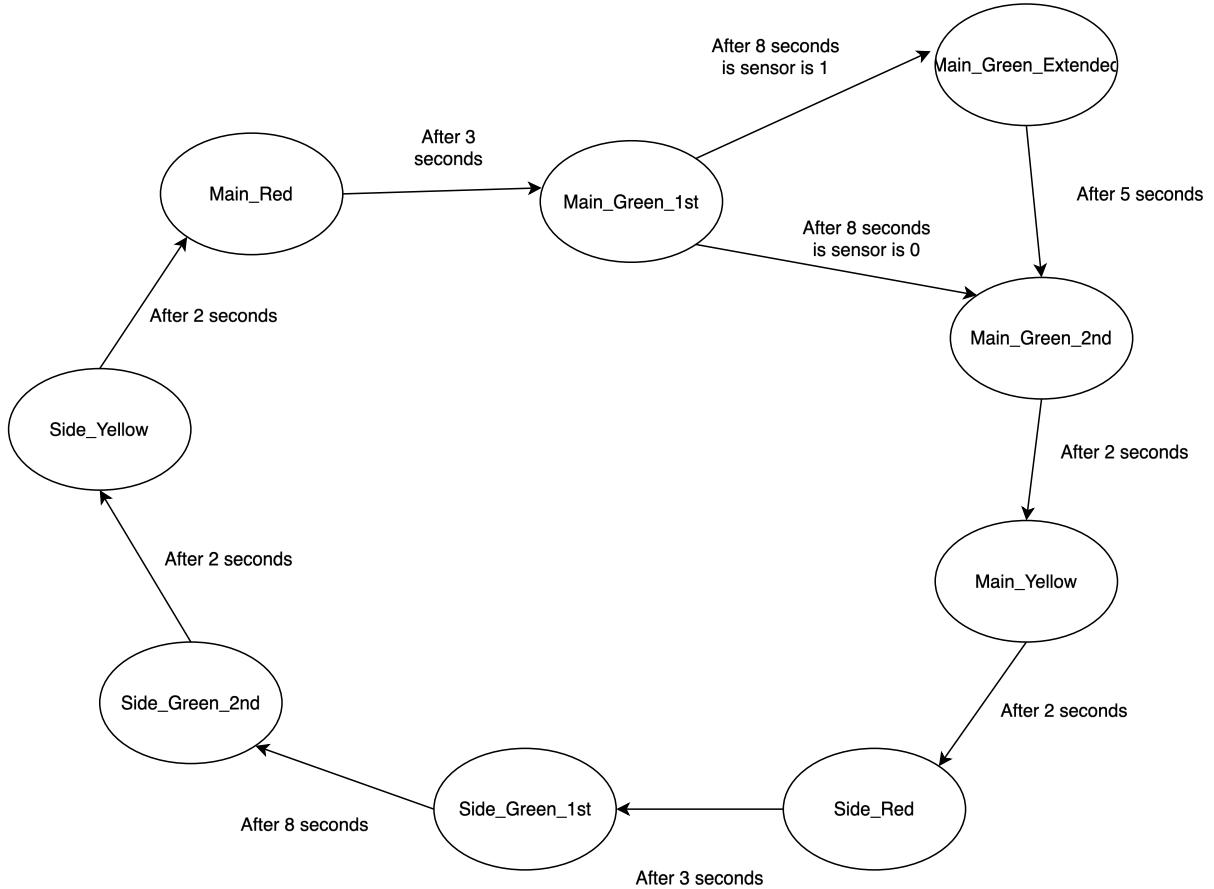


Figure 4: Extended green phase with for main street with both walk lights triggered

2 Devising the FSM

2.1 State diagram



2.2 Board setup

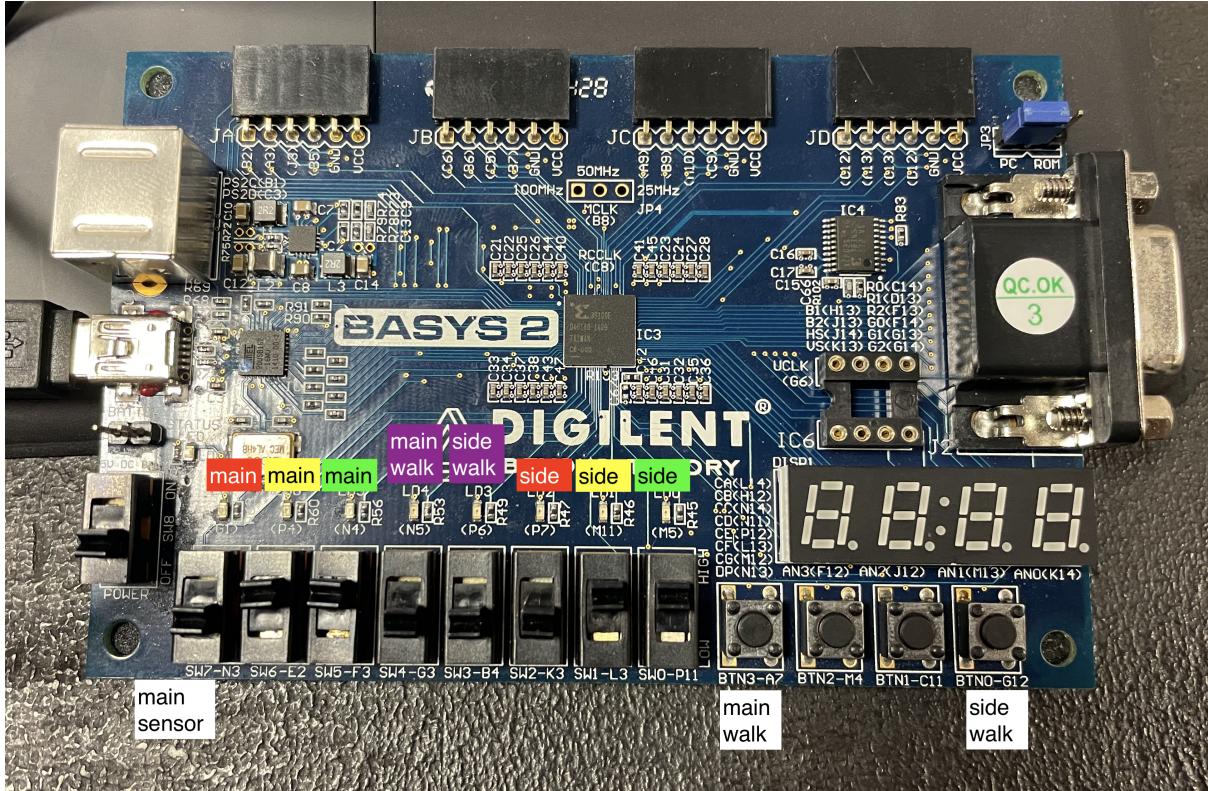


Figure 5: Board setup based on UCF

3 VHDL code of all modules

3.1 FSM module

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 -- Uncomment the following library declaration if using
5 -- arithmetic functions with Signed or Unsigned values
6 use IEEE.NUMERIC_STD.ALL;
7
8 -- Uncomment the following library declaration if instantiating
9 -- any Xilinx primitives in this code.
10--library UNISIM;
11--use UNISIM.VComponents.all;
12
13 entity FSM is PORT (
14     in_clk : IN std_logic;
15     in_sensor_main : IN std_logic;
16     in_main_walk : IN std_logic;
17     in_side_walk : IN std_logic;
18     out_main_red : OUT std_logic;
19     out_main_yellow : OUT std_logic;
```

```

20     out_main_green : OUT std_logic;
21     out_side_red : OUT std_logic;
22     out_side_yellow : OUT std_logic;
23     out_side_green : OUT std_logic;
24     out_main_walk : OUT std_logic;
25     out_side_walk : OUT std_logic
26 );
27 end FSM;
28
29 architecture Behavioral of FSM is
30
31 --*** FSM states
32 TYPE state_type IS (
33     main_red,
34     main_green_1st, -- first 8s (available for sensor input)
35     main_green_2nd, -- 2s
36     main_green_1st_extended,
37     main_yellow,
38     side_red,
39     side_green_1st,
40     side_green_2nd,
41     side_yellow
42 );
43 SIGNAL state, next_state : state_type := main_red;
44
45 --** constants, signals for clock dividers
46 constant sec_05 : integer := 25e6; -- counts 0.5sec, since board
        operates on 50MHz
47 constant sec_1 : integer := 2 * sec_05;
48 constant sec_2 : integer := 2 * sec_1;
49 constant sec_3 : integer := 3 * sec_1;
50 constant sec_5 : integer := 5 * sec_1;
51 constant sec_8 : integer := 8 * sec_1;
52 signal clk_count_state : integer range 0 to sec_8;
53 signal clk_count_blink : integer range 0 to sec_05;
54
55 --** signals for registers
56     -- store whether pedestrian has pressed walk button
57     -- a signal becomes a register when assigned inside clocked
        process
58 signal reg_main_walk, reg_side_walk, reg_blink : std_logic :=
59     '0';
60
61 begin
62
63 --*** clock divider process for state transitions; sequential
64 -- WHEN to move to next
65 PROCESS (in_clk)
66 BEGIN
67     if (rising_edge(in_clk)) then

```

```

68
69 CASE state IS
70
71     -- states with 3 s period: both lights red
72 WHEN main_red | side_red =>
73     if (clk_count_state = sec_3) then
74         state <= next_state;
75         clk_count_state <= 0;
76     else
77         clk_count_state <= clk_count_state + 1;
78     end if;
79
80     -- state with 8 s period: first green phase
81 WHEN main_green_1st =>
82     if (clk_count_state = sec_8) then
83         if (in_sensor_main = '1') then -- check
84             sensor
85             state <= main_green_1st_extended;
86         else
87             state <= main_green_2nd;
88         end if;
89
90         clk_count_state <= 0;
91     else
92         clk_count_state <= clk_count_state + 1;
93     end if;
94
95     -- state with 5 s period: extended green phase (main)
96 WHEN main_green_1st_extended =>
97     if (clk_count_state = sec_5) then
98         state <= main_green_2nd;
99         clk_count_state <= 0;
100    else
101        clk_count_state <= clk_count_state + 1;
102    end if;
103
104    -- states with 2 s period
105 WHEN main_green_2nd | main_yellow | side_green_2nd |
106     side_yellow =>
107     if (clk_count_state = sec_2) then
108         state <= next_state;
109         clk_count_state <= 0;
110     else
111         clk_count_state <= clk_count_state + 1;
112     end if;
113
114     -- state with 8 s period: second green phase (side)
115 WHEN side_green_1st =>
116     if (clk_count_state = sec_8) then

```

```

117         else
118             clk_count_state <= clk_count_state + 1;
119         end if;
120
121     WHEN OTHERS =>
122         clk_count_state <= 0;
123
124     END CASE;
125 end if;
126 END PROCESS;
127
128 *** clock divider process for blinking operation; sequential
129 -- BONUS revise this process (and possibly other parts) such that
130     the blinking starts properly, with light off and counter at
131     0, right after relevant state transitions
132 PROCESS (in_clk)
133 BEGIN
134     if (rising_edge(in_clk)) then
135         if (clk_count_blink = sec_05) then
136             reg_blink <= not reg_blink;
137             clk_count_blink <= 0;
138         else
139             clk_count_blink <= clk_count_blink + 1;
140         end if;
141     end if;
142 END PROCESS;
143
144 *** register; reg_side_walk
145 -- BONUS revise such that any further request that comes in just
146     after the cycle has finished is not ignored
147 PROCESS (in_clk)
148 BEGIN
149
150     if (rising_edge(in_clk)) then
151
152         -- condition of clearing the request
153         if (state = main_red and in_side_walk = '0') then
154             reg_side_walk <= '0';
155             -- for all other states, set the register in case the
156             -- button is pushed
157             elsif (in_side_walk = '1') then
158                 reg_side_walk <= '1';
159                 -- note: no final else, to hold request once set (and
160                 -- until reset)
161                 end if;
162             end if;
163
164     END PROCESS;
165
166 *** register; reg_main_walk (remembers walk request)

```

```

162 -- BONUS revise such that any further request that comes in just
163     after the cycle has finished is not ignored
164 PROCESS (in_clk)
165 BEGIN
166
167     if (rising_edge(in_clk)) then
168
169         -- condition of clearing the request
170         if (state = side_red and in_main_walk = '0') then
171             reg_main_walk <= '0';
172             -- for all other scenarios, set the register in case the
173             -- button is pushed
174             elsif (in_main_walk = '1') then -- if button is pushed
175                 reg_main_walk <= '1'; -- hold a pending request
176             -- note: no final else, to hold request once set (and
177                 -- until reset)
178             end if;
179         end if;
180
181     END PROCESS;
182
183 --** next state; sequential
184 -- WHERE to move next
185 PROCESS (state, in_sensor_main)
186 BEGIN
187
188     CASE state IS
189
190         WHEN main_red =>
191             next_state <= main_green_1st;
192
193         WHEN main_green_1st =>
194             if (in_sensor_main = '1') then
195                 next_state <= main_green_1st_extended;
196             else
197                 next_state <= main_green_2nd;
198             end if;
199
200         WHEN main_green_1st_extended =>
201             next_state <= main_green_2nd;
202
203         WHEN main_green_2nd =>
204             next_state <= main_yellow;
205
206         WHEN main_yellow =>
207             next_state <= side_red;
208
209         WHEN side_red =>
210             next_state <= side_green_1st;
211
212         WHEN side_green_1st =>
213             next_state <= main_red;
214
215     END CASE;
216
217     if (in_main_walk = '1') then
218         reg_main_walk <= '1';
219     end if;
220
221     if (in_sensor_main = '1') then
222         if (next_state = main_green_1st) then
223             reg_main_walk <= '1';
224         elsif (next_state = main_green_1st_extended) then
225             reg_main_walk <= '1';
226         elsif (next_state = main_green_2nd) then
227             reg_main_walk <= '1';
228         end if;
229     end if;
230
231     if (reg_main_walk = '1') then
232         if (next_state = main_red) then
233             reg_main_walk <= '0';
234         end if;
235     end if;
236
237     if (reg_main_walk = '1') then
238         if (next_state = side_red) then
239             reg_main_walk <= '0';
240         end if;
241     end if;
242
243     if (reg_main_walk = '1') then
244         if (next_state = side_green_1st) then
245             reg_main_walk <= '0';
246         end if;
247     end if;
248
249     if (reg_main_walk = '1') then
250         if (next_state = main_yellow) then
251             reg_main_walk <= '0';
252         end if;
253     end if;
254
255     if (reg_main_walk = '1') then
256         if (next_state = main_green_2nd) then
257             reg_main_walk <= '0';
258         end if;
259     end if;
260
261     if (reg_main_walk = '1') then
262         if (next_state = main_green_1st_extended) then
263             reg_main_walk <= '0';
264         end if;
265     end if;
266
267     if (reg_main_walk = '1') then
268         if (next_state = main_green_1st) then
269             reg_main_walk <= '0';
270         end if;
271     end if;
272
273     if (reg_main_walk = '1') then
274         if (next_state = side_green_1st) then
275             reg_main_walk <= '0';
276         end if;
277     end if;
278
279     if (reg_main_walk = '1') then
280         if (next_state = side_red) then
281             reg_main_walk <= '0';
282         end if;
283     end if;
284
285     if (reg_main_walk = '1') then
286         if (next_state = main_red) then
287             reg_main_walk <= '0';
288         end if;
289     end if;
290
291     if (reg_main_walk = '1') then
292         if (next_state = main_yellow) then
293             reg_main_walk <= '0';
294         end if;
295     end if;
296
297     if (reg_main_walk = '1') then
298         if (next_state = main_green_2nd) then
299             reg_main_walk <= '0';
300         end if;
301     end if;
302
303     if (reg_main_walk = '1') then
304         if (next_state = main_green_1st_extended) then
305             reg_main_walk <= '0';
306         end if;
307     end if;
308
309     if (reg_main_walk = '1') then
310         if (next_state = main_green_1st) then
311             reg_main_walk <= '0';
312         end if;
313     end if;
314
315     if (reg_main_walk = '1') then
316         if (next_state = side_green_1st) then
317             reg_main_walk <= '0';
318         end if;
319     end if;
320
321     if (reg_main_walk = '1') then
322         if (next_state = side_red) then
323             reg_main_walk <= '0';
324         end if;
325     end if;
326
327     if (reg_main_walk = '1') then
328         if (next_state = main_red) then
329             reg_main_walk <= '0';
330         end if;
331     end if;
332
333     if (reg_main_walk = '1') then
334         if (next_state = main_yellow) then
335             reg_main_walk <= '0';
336         end if;
337     end if;
338
339     if (reg_main_walk = '1') then
340         if (next_state = main_green_2nd) then
341             reg_main_walk <= '0';
342         end if;
343     end if;
344
345     if (reg_main_walk = '1') then
346         if (next_state = main_green_1st_extended) then
347             reg_main_walk <= '0';
348         end if;
349     end if;
350
351     if (reg_main_walk = '1') then
352         if (next_state = main_green_1st) then
353             reg_main_walk <= '0';
354         end if;
355     end if;
356
357     if (reg_main_walk = '1') then
358         if (next_state = side_green_1st) then
359             reg_main_walk <= '0';
360         end if;
361     end if;
362
363     if (reg_main_walk = '1') then
364         if (next_state = side_red) then
365             reg_main_walk <= '0';
366         end if;
367     end if;
368
369     if (reg_main_walk = '1') then
370         if (next_state = main_red) then
371             reg_main_walk <= '0';
372         end if;
373     end if;
374
375     if (reg_main_walk = '1') then
376         if (next_state = main_yellow) then
377             reg_main_walk <= '0';
378         end if;
379     end if;
380
381     if (reg_main_walk = '1') then
382         if (next_state = main_green_2nd) then
383             reg_main_walk <= '0';
384         end if;
385     end if;
386
387     if (reg_main_walk = '1') then
388         if (next_state = main_green_1st_extended) then
389             reg_main_walk <= '0';
390         end if;
391     end if;
392
393     if (reg_main_walk = '1') then
394         if (next_state = main_green_1st) then
395             reg_main_walk <= '0';
396         end if;
397     end if;
398
399     if (reg_main_walk = '1') then
400         if (next_state = side_green_1st) then
401             reg_main_walk <= '0';
402         end if;
403     end if;
404
405     if (reg_main_walk = '1') then
406         if (next_state = side_red) then
407             reg_main_walk <= '0';
408         end if;
409     end if;
410
411     if (reg_main_walk = '1') then
412         if (next_state = main_red) then
413             reg_main_walk <= '0';
414         end if;
415     end if;
416
417     if (reg_main_walk = '1') then
418         if (next_state = main_yellow) then
419             reg_main_walk <= '0';
420         end if;
421     end if;
422
423     if (reg_main_walk = '1') then
424         if (next_state = main_green_2nd) then
425             reg_main_walk <= '0';
426         end if;
427     end if;
428
429     if (reg_main_walk = '1') then
430         if (next_state = main_green_1st_extended) then
431             reg_main_walk <= '0';
432         end if;
433     end if;
434
435     if (reg_main_walk = '1') then
436         if (next_state = main_green_1st) then
437             reg_main_walk <= '0';
438         end if;
439     end if;
440
441     if (reg_main_walk = '1') then
442         if (next_state = side_green_1st) then
443             reg_main_walk <= '0';
444         end if;
445     end if;
446
447     if (reg_main_walk = '1') then
448         if (next_state = side_red) then
449             reg_main_walk <= '0';
450         end if;
451     end if;
452
453     if (reg_main_walk = '1') then
454         if (next_state = main_red) then
455             reg_main_walk <= '0';
456         end if;
457     end if;
458
459     if (reg_main_walk = '1') then
460         if (next_state = main_yellow) then
461             reg_main_walk <= '0';
462         end if;
463     end if;
464
465     if (reg_main_walk = '1') then
466         if (next_state = main_green_2nd) then
467             reg_main_walk <= '0';
468         end if;
469     end if;
470
471     if (reg_main_walk = '1') then
472         if (next_state = main_green_1st_extended) then
473             reg_main_walk <= '0';
474         end if;
475     end if;
476
477     if (reg_main_walk = '1') then
478         if (next_state = main_green_1st) then
479             reg_main_walk <= '0';
480         end if;
481     end if;
482
483     if (reg_main_walk = '1') then
484         if (next_state = side_green_1st) then
485             reg_main_walk <= '0';
486         end if;
487     end if;
488
489     if (reg_main_walk = '1') then
490         if (next_state = side_red) then
491             reg_main_walk <= '0';
492         end if;
493     end if;
494
495     if (reg_main_walk = '1') then
496         if (next_state = main_red) then
497             reg_main_walk <= '0';
498         end if;
499     end if;
499
500     if (reg_main_walk = '1') then
501         if (next_state = main_yellow) then
502             reg_main_walk <= '0';
503         end if;
504     end if;
505
506     if (reg_main_walk = '1') then
507         if (next_state = main_green_2nd) then
508             reg_main_walk <= '0';
509         end if;
510     end if;
511
512     if (reg_main_walk = '1') then
513         if (next_state = main_green_1st_extended) then
514             reg_main_walk <= '0';
515         end if;
516     end if;
517
518     if (reg_main_walk = '1') then
519         if (next_state = main_green_1st) then
520             reg_main_walk <= '0';
521         end if;
522     end if;
523
524     if (reg_main_walk = '1') then
525         if (next_state = side_green_1st) then
526             reg_main_walk <= '0';
527         end if;
528     end if;
529
530     if (reg_main_walk = '1') then
531         if (next_state = side_red) then
532             reg_main_walk <= '0';
533         end if;
534     end if;
535
536     if (reg_main_walk = '1') then
537         if (next_state = main_red) then
538             reg_main_walk <= '0';
539         end if;
540     end if;
541
542     if (reg_main_walk = '1') then
543         if (next_state = main_yellow) then
544             reg_main_walk <= '0';
545         end if;
546     end if;
547
548     if (reg_main_walk = '1') then
549         if (next_state = main_green_2nd) then
550             reg_main_walk <= '0';
551         end if;
552     end if;
553
554     if (reg_main_walk = '1') then
555         if (next_state = main_green_1st_extended) then
556             reg_main_walk <= '0';
557         end if;
558     end if;
559
560     if (reg_main_walk = '1') then
561         if (next_state = main_green_1st) then
562             reg_main_walk <= '0';
563         end if;
564     end if;
565
566     if (reg_main_walk = '1') then
567         if (next_state = side_green_1st) then
568             reg_main_walk <= '0';
569         end if;
570     end if;
571
572     if (reg_main_walk = '1') then
573         if (next_state = side_red) then
574             reg_main_walk <= '0';
575         end if;
576     end if;
577
578     if (reg_main_walk = '1') then
579         if (next_state = main_red) then
580             reg_main_walk <= '0';
581         end if;
582     end if;
583
584     if (reg_main_walk = '1') then
585         if (next_state = main_yellow) then
586             reg_main_walk <= '0';
587         end if;
588     end if;
589
590     if (reg_main_walk = '1') then
591         if (next_state = main_green_2nd) then
592             reg_main_walk <= '0';
593         end if;
594     end if;
595
596     if (reg_main_walk = '1') then
597         if (next_state = main_green_1st_extended) then
598             reg_main_walk <= '0';
599         end if;
600     end if;
601
602     if (reg_main_walk = '1') then
603         if (next_state = main_green_1st) then
604             reg_main_walk <= '0';
605         end if;
606     end if;
607
608     if (reg_main_walk = '1') then
609         if (next_state = side_green_1st) then
610             reg_main_walk <= '0';
611         end if;
612     end if;
613
614     if (reg_main_walk = '1') then
615         if (next_state = side_red) then
616             reg_main_walk <= '0';
617         end if;
618     end if;
619
620     if (reg_main_walk = '1') then
621         if (next_state = main_red) then
622             reg_main_walk <= '0';
623         end if;
624     end if;
625
626     if (reg_main_walk = '1') then
627         if (next_state = main_yellow) then
628             reg_main_walk <= '0';
629         end if;
630     end if;
631
632     if (reg_main_walk = '1') then
633         if (next_state = main_green_2nd) then
634             reg_main_walk <= '0';
635         end if;
636     end if;
637
638     if (reg_main_walk = '1') then
639         if (next_state = main_green_1st_extended) then
640             reg_main_walk <= '0';
641         end if;
642     end if;
643
644     if (reg_main_walk = '1') then
645         if (next_state = main_green_1st) then
646             reg_main_walk <= '0';
647         end if;
648     end if;
649
650     if (reg_main_walk = '1') then
651         if (next_state = side_green_1st) then
652             reg_main_walk <= '0';
653         end if;
654     end if;
655
656     if (reg_main_walk = '1') then
657         if (next_state = side_red) then
658             reg_main_walk <= '0';
659         end if;
660     end if;
661
662     if (reg_main_walk = '1') then
663         if (next_state = main_red) then
664             reg_main_walk <= '0';
665         end if;
666     end if;
667
668     if (reg_main_walk = '1') then
669         if (next_state = main_yellow) then
670             reg_main_walk <= '0';
671         end if;
672     end if;
673
674     if (reg_main_walk = '1') then
675         if (next_state = main_green_2nd) then
676             reg_main_walk <= '0';
677         end if;
678     end if;
679
680     if (reg_main_walk = '1') then
681         if (next_state = main_green_1st_extended) then
682             reg_main_walk <= '0';
683         end if;
684     end if;
685
686     if (reg_main_walk = '1') then
687         if (next_state = main_green_1st) then
688             reg_main_walk <= '0';
689         end if;
690     end if;
691
692     if (reg_main_walk = '1') then
693         if (next_state = side_green_1st) then
694             reg_main_walk <= '0';
695         end if;
696     end if;
697
698     if (reg_main_walk = '1') then
699         if (next_state = side_red) then
700             reg_main_walk <= '0';
701         end if;
702     end if;
703
704     if (reg_main_walk = '1') then
705         if (next_state = main_red) then
706             reg_main_walk <= '0';
707         end if;
708     end if;
709
710     if (reg_main_walk = '1') then
711         if (next_state = main_yellow) then
712             reg_main_walk <= '0';
713         end if;
714     end if;
715
716     if (reg_main_walk = '1') then
717         if (next_state = main_green_2nd) then
718             reg_main_walk <= '0';
719         end if;
720     end if;
721
722     if (reg_main_walk = '1') then
723         if (next_state = main_green_1st_extended) then
724             reg_main_walk <= '0';
725         end if;
726     end if;
727
728     if (reg_main_walk = '1') then
729         if (next_state = main_green_1st) then
730             reg_main_walk <= '0';
731         end if;
732     end if;
733
734     if (reg_main_walk = '1') then
735         if (next_state = side_green_1st) then
736             reg_main_walk <= '0';
737         end if;
738     end if;
739
740     if (reg_main_walk = '1') then
741         if (next_state = side_red) then
742             reg_main_walk <= '0';
743         end if;
744     end if;
745
746     if (reg_main_walk = '1') then
747         if (next_state = main_red) then
748             reg_main_walk <= '0';
749         end if;
750     end if;
751
752     if (reg_main_walk = '1') then
753         if (next_state = main_yellow) then
754             reg_main_walk <= '0';
755         end if;
756     end if;
757
758     if (reg_main_walk = '1') then
759         if (next_state = main_green_2nd) then
760             reg_main_walk <= '0';
761         end if;
762     end if;
763
764     if (reg_main_walk = '1') then
765         if (next_state = main_green_1st_extended) then
766             reg_main_walk <= '0';
767         end if;
768     end if;
769
770     if (reg_main_walk = '1') then
771         if (next_state = main_green_1st) then
772             reg_main_walk <= '0';
773         end if;
774     end if;
775
776     if (reg_main_walk = '1') then
777         if (next_state = side_green_1st) then
778             reg_main_walk <= '0';
779         end if;
780     end if;
781
782     if (reg_main_walk = '1') then
783         if (next_state = side_red) then
784             reg_main_walk <= '0';
785         end if;
786     end if;
787
788     if (reg_main_walk = '1') then
789         if (next_state = main_red) then
790             reg_main_walk <= '0';
791         end if;
792     end if;
793
794     if (reg_main_walk = '1') then
795         if (next_state = main_yellow) then
796             reg_main_walk <= '0';
797         end if;
798     end if;
799
800     if (reg_main_walk = '1') then
801         if (next_state = main_green_2nd) then
802             reg_main_walk <= '0';
803         end if;
804     end if;
805
806     if (reg_main_walk = '1') then
807         if (next_state = main_green_1st_extended) then
808             reg_main_walk <= '0';
809         end if;
810     end if;
811
812     if (reg_main_walk = '1') then
813         if (next_state = main_green_1st) then
814             reg_main_walk <= '0';
815         end if;
816     end if;
817
818     if (reg_main_walk = '1') then
819         if (next_state = side_green_1st) then
820             reg_main_walk <= '0';
821         end if;
822     end if;
823
824     if (reg_main_walk = '1') then
825         if (next_state = side_red) then
826             reg_main_walk <= '0';
827         end if;
828     end if;
829
830     if (reg_main_walk = '1') then
831         if (next_state = main_red) then
832             reg_main_walk <= '0';
833         end if;
834     end if;
835
836     if (reg_main_walk = '1') then
837         if (next_state = main_yellow) then
838             reg_main_walk <= '0';
839         end if;
840     end if;
841
842     if (reg_main_walk = '1') then
843         if (next_state = main_green_2nd) then
844             reg_main_walk <= '0';
845         end if;
846     end if;
847
848     if (reg_main_walk = '1') then
849         if (next_state = main_green_1st_extended) then
850             reg_main_walk <= '0';
851         end if;
852     end if;
853
854     if (reg_main_walk = '1') then
855         if (next_state = main_green_1st) then
856             reg_main_walk <= '0';
857         end if;
858     end if;
859
860     if (reg_main_walk = '1') then
861         if (next_state = side_green_1st) then
862             reg_main_walk <= '0';
863         end if;
864     end if;
865
866     if (reg_main_walk = '1') then
867         if (next_state = side_red) then
868             reg_main_walk <= '0';
869         end if;
870     end if;
871
872     if (reg_main_walk = '1') then
873         if (next_state = main_red) then
874             reg_main_walk <= '0';
875         end if;
876     end if;
877
878     if (reg_main_walk = '1') then
879         if (next_state = main_yellow) then
880             reg_main_walk <= '0';
881         end if;
882     end if;
883
884     if (reg_main_walk = '1') then
885         if (next_state = main_green_2nd) then
886             reg_main_walk <= '0';
887         end if;
888     end if;
889
890     if (reg_main_walk = '1') then
891         if (next_state = main_green_1st_extended) then
892             reg_main_walk <= '0';
893         end if;
894     end if;
895
896     if (reg_main_walk = '1') then
897         if (next_state = main_green_1st) then
898             reg_main_walk <= '0';
899         end if;
900     end if;
901
902     if (reg_main_walk = '1') then
903         if (next_state = side_green_1st) then
904             reg_main_walk <= '0';
905         end if;
906     end if;
907
908     if (reg_main_walk = '1') then
909         if (next_state = side_red) then
910             reg_main_walk <= '0';
911         end if;
912     end if;
913
914     if (reg_main_walk = '1') then
915         if (next_state = main_red) then
916             reg_main_walk <= '0';
917         end if;
918     end if;
919
920     if (reg_main_walk = '1') then
921         if (next_state = main_yellow) then
922             reg_main_walk <= '0';
923         end if;
924     end if;
925
926     if (reg_main_walk = '1') then
927         if (next_state = main_green_2nd) then
928             reg_main_walk <= '0';
929         end if;
930     end if;
931
932     if (reg_main_walk = '1') then
933         if (next_state = main_green_1st_extended) then
934             reg_main_walk <= '0';
935         end if;
936     end if;
937
938     if (reg_main_walk = '1') then
939         if (next_state = main_green_1st) then
940             reg_main_walk <= '0';
941         end if;
942     end if;
943
944     if (reg_main_walk = '1') then
945         if (next_state = side_green_1st) then
946             reg_main_walk <= '0';
947         end if;
948     end if;
949
950     if (reg_main_walk = '1') then
951         if (next_state = side_red) then
952             reg_main_walk <= '0';
953         end if;
954     end if;
955
956     if (reg_main_walk = '1') then
957         if (next_state = main_red) then
958             reg_main_walk <= '0';
959         end if;
960     end if;
961
962     if (reg_main_walk = '1') then
963         if (next_state = main_yellow) then
964             reg_main_walk <= '0';
965         end if;
966     end if;
967
968     if (reg_main_walk = '1') then
969         if (next_state = main_green_2nd) then
970             reg_main_walk <= '0';
971         end if;
972     end if;
973
974     if (reg_main_walk = '1') then
975         if (next_state = main_green_1st_extended) then
976             reg_main_walk <= '0';
977         end if;
978     end if;
979
980     if (reg_main_walk = '1') then
981         if (next_state = main_green_1st) then
982             reg_main_walk <= '0';
983         end if;
984     end if;
985
986     if (reg_main_walk = '1') then
987         if (next_state = side_green_1st) then
988             reg_main_walk <= '0';
989         end if;
990     end if;
991
992     if (reg_main_walk = '1') then
993         if (next_state = side_red) then
994             reg_main_walk <= '0';
995         end if;
996     end if;
997
998     if (reg_main_walk = '1') then
999         if (next_state = main_red) then
1000            reg_main_walk <= '0';
1001        end if;
1002    end if;
1003
1004    if (reg_main_walk = '1') then
1005        if (next_state = main_yellow) then
1006            reg_main_walk <= '0';
1007        end if;
1008    end if;
1009
1010    if (reg_main_walk = '1') then
1011        if (next_state = main_green_2nd) then
1012            reg_main_walk <= '0';
1013        end if;
1014    end if;
1015
1016    if (reg_main_walk = '1') then
1017        if (next_state = main_green_1st_extended) then
1018            reg_main_walk <= '0';
1019        end if;
1020    end if;
1021
1022    if (reg_main_walk = '1') then
1023        if (next_state = main_green_1st) then
1024            reg_main_walk <= '0';
1025        end if;
1026    end if;
1027
1028    if (reg_main_walk = '1') then
1029        if (next_state = side_green_1st) then
1030            reg_main_walk <= '0';
1031        end if;
1032    end if;
1033
1034    if (reg_main_walk = '1') then
1035        if (next_state = side_red) then
1036            reg_main_walk <= '0';
1037        end if;
1038    end if;
1039
1040    if (reg_main_walk = '1') then
1041        if (next_state = main_red) then
1042            reg_main_walk <= '0';
1043        end if;
1044    end if;
1045
1046    if (reg_main_walk = '1') then
1047        if (next_state = main_yellow) then
1048            reg_main_walk <= '0';
1049        end if;
1050    end if;
1051
1052    if (reg_main_walk = '1') then
1053        if (next_state = main_green_2nd) then
1054            reg_main_walk <= '0';
1055        end if;
1056    end if;
1057
1058    if (reg_main_walk = '1') then
1059        if (next_state = main_green_1st_extended) then
1060            reg_main_walk <= '0';
1061        end if;
1062    end if;
1063
1064    if (reg_main_walk = '1') then
1065        if (next_state = main_green_1st) then
1066            reg_main_walk <= '0';
1067        end if;
1068    end if;
1069
1070    if (reg_main_walk = '1') then
1071        if (next_state = side_green_1st) then
1072            reg_main_walk <= '0';
1073        end if;
1074    end if;
1075
1076    if (reg_main_walk = '1') then
1077        if (next_state = side_red) then
1078            reg_main_walk <= '0';
1079        end if;
1080    end if;
1081
1082    if (reg_main_walk = '1') then
1083        if (next_state = main_red) then
1084            reg_main_walk <= '0';
1085        end if;
1086    end if;
1087
1088    if (reg_main_walk = '1') then
1089        if (next_state = main_yellow) then
1090            reg_main_walk <= '0';
1091        end if;
1092    end if;
1093
1094    if (reg_main_walk = '1') then
1095        if (next_state = main_green_2nd) then
1096            reg_main_walk <= '0';
1097        end if;
1098    end if;
1099
1100    if (reg_main_walk = '1') then
1101        if (next_state = main_green_1st_extended) then
1102            reg_main_walk <= '0';
1103        end if;
1104    end if;
1105
1106    if (reg_main_walk = '1') then
1107        if (next_state = main_green_1st) then
1108            reg_main_walk <= '0';
1109        end if;
1110    end if;
1111
1112    if (reg_main_walk = '1') then
1113        if (next_state = side_green_1st) then
1114            reg_main_walk <= '0';
1115        end if;
1116    end if;
1117
1118    if (reg_main_walk = '1') then
1119        if (next_state = side_red) then
1120            reg_main_walk <= '0';
1121        end if;
1122    end if;
1123
1124    if (reg_main_walk = '1') then
1125        if (next_state = main_red) then
1126            reg_main_walk <= '0';
1127        end if;
1128    end if;
1129
1130    if (reg_main_walk = '1') then
1131        if (next_state = main_yellow) then
1132            reg_main_walk <= '0';
1133        end if;
1134    end if;
1135
1136    if (reg_main_walk = '1') then
1137        if (next_state = main_green_2nd) then
1138            reg_main_walk <= '0';
1139        end if;
1140    end if;
1141
1142    if (reg_main_walk = '1') then
1143        if (next_state = main_green_1st_extended) then
1144            reg_main_walk <= '0';
1145        end if;
1146    end if;
1147
1148    if (reg_main_walk = '1') then
1149        if (next_state = main_green_1st) then
1150            reg_main_walk <= '0';
1151        end if;
1152    end if;
1153
1154    if (reg_main_walk = '1') then
1155        if (next_state = side_green_1st) then
1156            reg_main_walk <= '0';
1157        end if;
1158    end if;
1159
1160    if (reg_main_walk = '1') then
1161        if (next_state = side_red) then
1162            reg_main_walk <= '0';
1163        end if;
1164    end if;
1165
1166    if (reg_main_walk = '1') then
1167        if (next_state = main_red) then
1168            reg_main_walk <= '0';
1169        end if;
1170    end if;
1171
1172    if (reg_main_walk = '1') then
1173        if (next_state = main_yellow) then
1174            reg_main_walk <= '0';
1175        end if;
1176    end if;
1177
1178    if (reg_main_walk = '1') then
1179        if (next_state = main_green_2nd) then
1180            reg_main_walk <= '0';
1181        end if;
1182    end if;
1183
1184    if (reg_main_walk = '1') then
1185        if (next_state = main_green_1st_extended) then
1186            reg_main_walk <= '0';
1187        end if;
1188    end if;
1189
1190    if (reg_main_walk = '1') then
1191        if (next_state = main_green_1st) then
1192            reg_main_walk <= '0';
1193        end if;
1194    end if;
1195
1196    if (reg_main_walk = '1') then
1197        if (next_state = side_green_1st) then
1198            reg_main_walk <= '0';
1199        end if;
1200    end if;
1201
1202    if (reg_main_walk = '1') then
1203        if (next_state = side_red) then
1204            reg_main_walk <= '0';
1205       
```

```

210     next_state <= side_green_2nd;
211
212 WHEN side_green_2nd =>
213     next_state <= side_yellow;
214
215 WHEN side_yellow =>
216     next_state <= main_red;
217
218 WHEN OTHERS => -- in case of invalid state,
219     next_state <= state; -- restart the cycle
220
221 END CASE;
222
223 END PROCESS;
224
225 --*** outputs; combinational
226 PROCESS (state, reg_side_walk, reg_main_walk, reg_blink)
227 BEGIN
228
229 -- default assignments
230 out_main_red <= '0';
231 out_main_yellow <= '0';
232 out_main_green <= '0';
233 out_side_red <= '0';
234 out_side_yellow <= '0';
235 out_side_green <= '0';
236 out_main_walk <= '0';
237 out_side_walk <= '0';
238
239 CASE state IS
240
241     WHEN main_red =>
242         out_main_red <= '1';
243         out_side_red <= '1';
244
245     WHEN main_green_1st =>
246         out_main_green <= '1';
247         out_side_red <= '1';
248         out_main_walk <= reg_main_walk;
249
250     WHEN main_green_1st_extended =>
251         out_main_green <= '1';
252         out_side_red <= '1';
253         out_main_walk <= reg_main_walk and reg_blink; -- can't
254             only be reg_blink since it toggles
255
256     WHEN main_green_2nd =>
257         out_main_yellow <= '1';
258         out_side_red <= '1';
259         out_main_walk <= reg_main_walk and reg_blink;

```

```

260 WHEN main_yellow =>
261     out_main_yellow <= '1';
262     out_side_red <= '1';
263     out_main_walk <= reg_main_walk and reg_blink;
264
265 WHEN side_red =>
266     out_main_red <= '1';
267     out_side_red <= '1';
268
269 WHEN side_green_1st =>
270     out_main_red <= '1';
271     out_side_green <= '1';
272     out_side_walk <= reg_side_walk;
273
274
275 WHEN side_green_2nd =>
276     out_main_red <= '1';
277     out_side_green <= '1';
278     out_side_walk <= reg_side_walk and reg_blink;
279
280
281 WHEN side_yellow =>
282     out_main_red <= '1';
283     out_side_yellow <= '1';
284     out_side_walk <= reg_side_walk and reg_blink;
285
286 WHEN OTHERS =>
287     NULL;
288
289
290
291
292 END CASE;
293
294 END PROCESS;
295
296 end Behavioral;

```

Listing 1: VHDL code for "FSM" module

3.2 TestBench Module

```

1 LIBRARY ieee;
2 USE ieee.std_logic_1164.ALL;
3
4 -- Uncomment the following library declaration if using
5 -- arithmetic functions with Signed or Unsigned values
6 --USE ieee.numeric_std.ALL;
7
8 ENTITY FSM_tb_c IS
9 END FSM_tb_c;

```

```

10
11 ARCHITECTURE behavior OF FSM_tb_c IS
12
13     -- Component Declaration for the Unit Under Test (UUT)
14
15     COMPONENT FSM
16     PORT(
17         in_clk : IN std_logic;
18         in_sensor_main : IN std_logic;
19         in_main_walk : IN std_logic;
20         in_side_walk : IN std_logic;
21         out_main_red : OUT std_logic;
22         out_main_yellow : OUT std_logic;
23         out_main_green : OUT std_logic;
24         out_side_red : OUT std_logic;
25         out_side_yellow : OUT std_logic;
26         out_side_green : OUT std_logic;
27         out_main_walk : OUT std_logic;
28         out_side_walk : OUT std_logic
29     );
30 END COMPONENT;
31
32
33     --Inputs
34     signal in_clk : std_logic := '0';
35     signal in_sensor_main_tb : std_logic := '0';
36     signal in_main_walk_tb : std_logic := '0';
37     signal in_side_walk_tb : std_logic := '0';
38
39     --Outputs
40     signal out_main_red_tb : std_logic;
41     signal out_main_yellow_tb : std_logic;
42     signal out_main_green_tb : std_logic;
43     signal out_side_red_tb : std_logic;
44     signal out_side_yellow_tb : std_logic;
45     signal out_side_green_tb : std_logic;
46     signal out_main_walk_tb : std_logic;
47     signal out_side_walk_tb : std_logic;
48
49     -- Clock period definitions
50     constant in_clk_period : time := 5 ns;
51
52 BEGIN
53
54     -- Instantiate the Unit Under Test (UUT)
55     uut: FSM PORT MAP (
56         in_clk => in_clk,
57         in_sensor_main => in_sensor_main_tb,
58         in_main_walk => in_main_walk_tb,
59         in_side_walk => in_side_walk_tb,
60         out_main_red => out_main_red_tb,

```

```

61      out_main_yellow => out_main_yellow_tb,
62      out_main_green  => out_main_green_tb,
63      out_side_red   => out_side_red_tb,
64      out_side_yellow => out_side_yellow_tb,
65      out_side_green  => out_side_green_tb,
66      out_main_walk   => out_main_walk_tb,
67      out_side_walk   => out_side_walk_tb
68  );
69
70  -- Clock process definitions
71  in_clk_process :process
72  begin
73      in_clk <= '0';
74      wait for in_clk_period/2;
75      in_clk <= '1';
76      wait for in_clk_period/2;
77  end process;
78
79
80  -- Stimulus process
81  stim_proc: process
82  begin
83  -- Before this, change: constant sec_05 : integer := 25e6; to 25
84  -- e1 in FSM
85  --     TB1 Everything is Zero
86  --
87  --     TB 2 - sensor on
88  --     in_sensor_main_tb <= '1';
89
90
91  --     TB_3 - press both walk buttons with sensor off
92  --     wait for 10 us;
93  --     -- TB_4 - change sensor to 1
94  --     in_sensor_main_tb <= '1';
95  --     in_side_walk_tb <= '1';
96  --     in_main_walk_tb <= '1';
97  --
98  --     wait for 5 us;
99  --     in_side_walk_tb <= '0';
100 --     in_main_walk_tb <= '0';
101
102
103 --     TB_4 - An x error is assigned for the sensor
104 --     in_sensor_main_tb <= 'X';
105
106 --     TB_5 - An x error is assigned to both walk buttons
107 --     in_side_walk_tb <= 'X';
108 --     in_main_walk_tb <= 'X';
109
110    wait;

```

```
111      end process;  
112  
113 END;
```

Listing 2: VHDL code for "FSM" module

4 Testbench simulation results

4.1 All inputs off

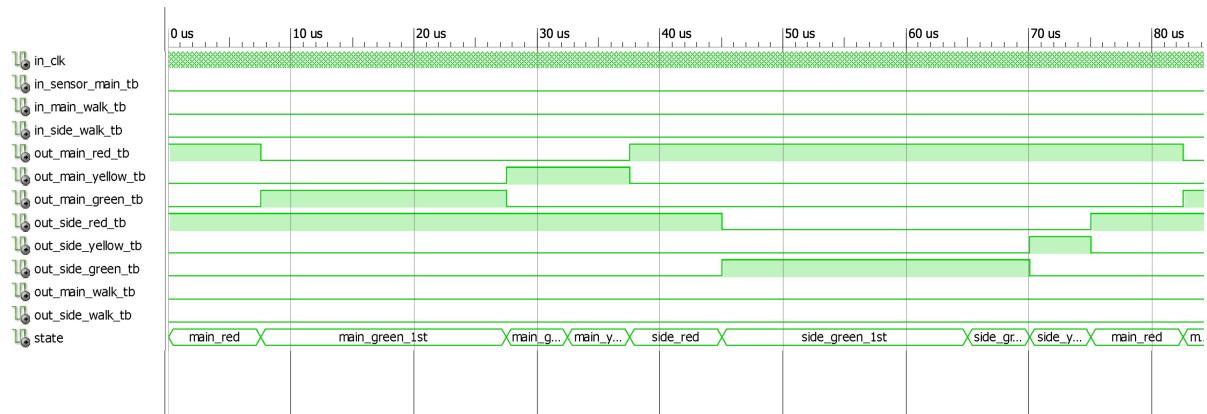


Figure 6: Board setup based on UCF

4.2 Main sensor on (Main green extended)

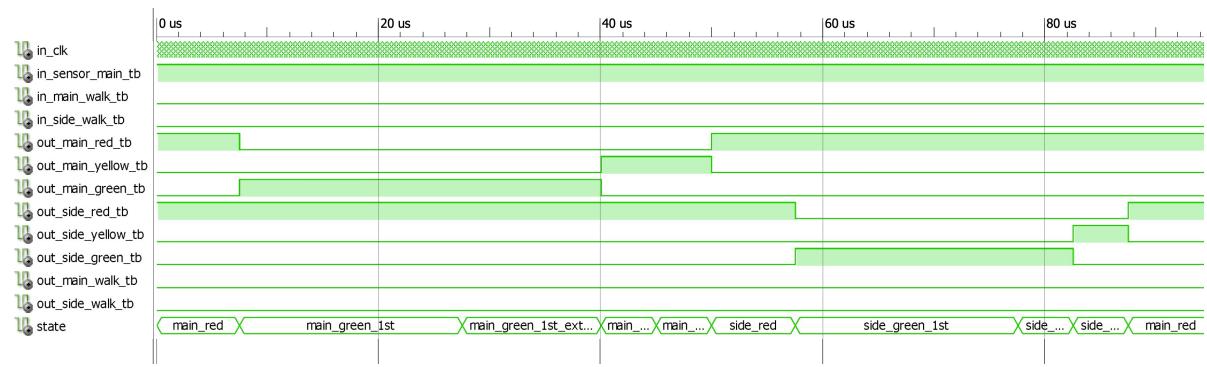


Figure 7: Board setup based on UCF

4.3 Both walk buttons on

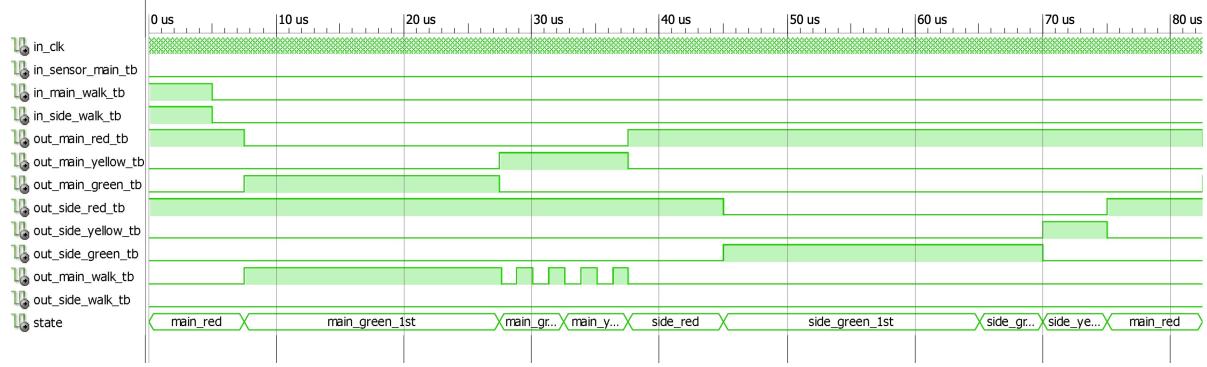


Figure 8: Board setup based on UCF

4.4 Both walk buttons and main sensor on

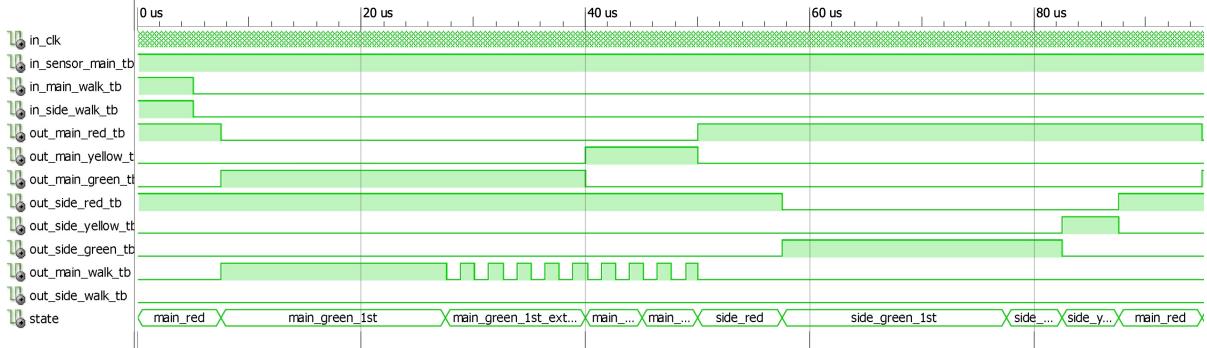


Figure 9: Board setup based on UCF

4.5 "X" error for main sensor

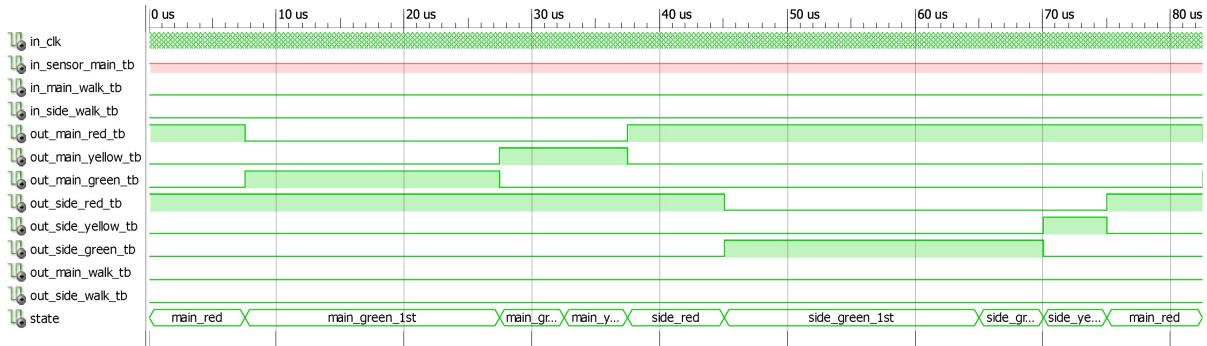


Figure 10: Board setup based on UCF

4.6 "X" error for both walk buttons

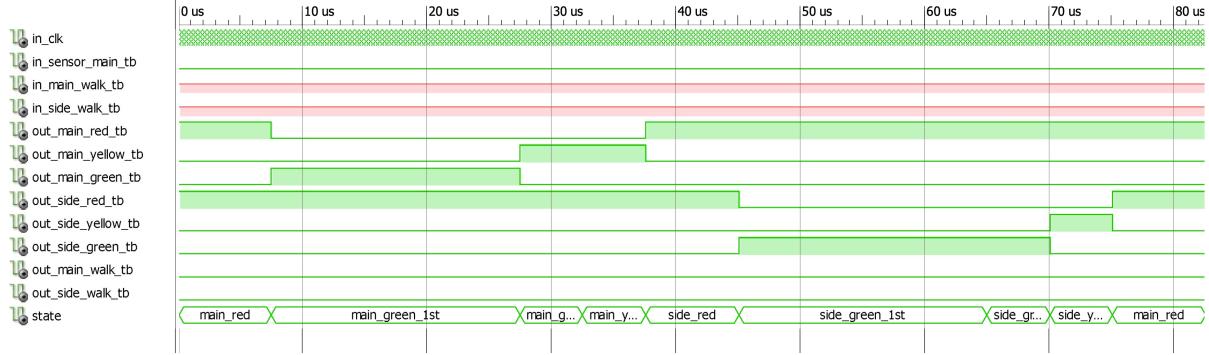


Figure 11: Board setup based on UCF

5 Constraints file

```

1 # clock pin for Basys2 Board
2 NET "in_clk" LOC = "B8"; # Bank = 0, Signal name = MCLK
3
4 # Pin assignment for LEDs
5 NET "out_main_red" LOC = "G1" ; # Bank = 3, Signal name = LD7
6 NET "out_main_yellow" LOC = "P4" ; # Bank = 2, Signal name = LD6
7 NET "out_main_green" LOC = "N4" ; # Bank = 2, Signal name = LD5
8
9 NET "out_main_walk" LOC = "N5" ;
10 NET "out_side_walk" LOC = "P6" ;
11
12 NET "out_side_red" LOC = "P7" ;
13 NET "out_side_yellow" LOC = "M11" ;
14 NET "out_side_green" LOC = "M5" ;
15
16 # Pin assignment for SWs
17 NET "in_sensor_main" LOC = "N3"; # Bank = 2, Signal name = SW7
18
19 # Pin assignment for BTNs
20 NET "in_main_walk" LOC = "A7";
21 NET "in_side_walk" LOC = "G12";

```

Listing 3: Constraints file for FSM module

6 FPGA implementation

[Click to watch video demonstration](#)