

Implementation of Tic-Tac-Toe Game via VGA Interface and VHDL

Appendix

Digital Electronics (ENGS 31/ COSC 56)

Dartmouth College, Summer 2022

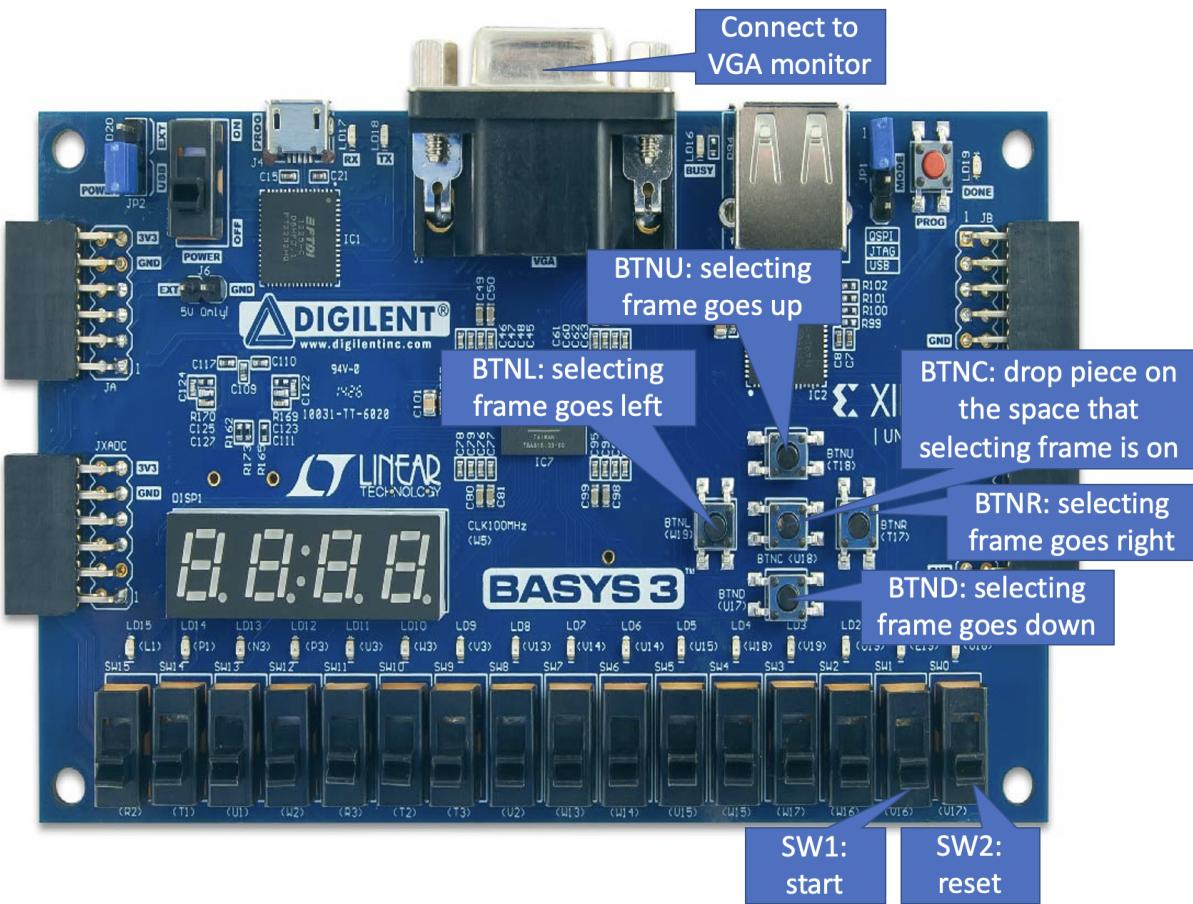
Students: Di Luo, Itish Goel

Instructors: Benjamin Livingston Dobbins, Tad Truex

Catalog

Appendix A: User interface on Basys 3 -----	3
Appendix B: VGA_driver.vhd -----	4
Appendix C: game_logic.vhd -----	8
Appendix D: pixel_generation.vhd -----	22
Appendix E: system_clock_generator.vhd -----	37
Appendix F: input_conditioning.vhd -----	40
Appendix G: TicTacToe_shell.vhd -----	45
Appendix H: game_logic_tb.vhd -----	51
Appendix I: Test results of game_logic_tb.vhd -----	61
Appendix J: VGA driver testing with patterns in vga_test_pattern_12.vhd and the one based on Tad's suggestions -----	63

Appendix A: User interface on Basys 3



Appendix B: VGA_driver.vhd

```
VGA_driver.vhd                                         8/20/22, 1:33 PM

1 ---
2 =====
3 --ENGS 31/ CoSc 56
4 --Final Project: Tic-tac-toe in VGA
5 --VGA driver
6 --Implemented based on template: https://www.edaplayground.com/x/6Lwd
7 --Instructors: Ben Dobbins, Tad Truex
8 --Student name: Di Luo
9 --
10 --
11 --Library Declarations:
12 --
13 library IEEE;
14 use IEEE.STD_LOGIC_1164.ALL;
15 use IEEE.NUMERIC_STD.ALL;
16 use ieee.math_real.all;
17 library UNISIM;
18 use UNISIM.VComponents.all;
19 --
20 --
21 --Entity Declaration:
22 --
23 ENTITY VGA_driver IS
24 PORT (
25     vclk_port      : in STD_LOGIC;    -- 25 MHz clock
26     Vsync_port      : out STD_LOGIC;   -- vertical sync signal to VGA monitor
27     Hsync_port      : out STD_LOGIC;   -- horizontal sync signal to VGA
28     monitor
29     video_on_port  : out STD_LOGIC;   -- accomplish blanking. 1 when
30     inside display area. 0 when outside display area
31     pixel_x_port   : out STD_LOGIC_VECTOR(9 downto 0);    -- current pixel's x
32     coordinate
33     pixel_y_port   : out STD_LOGIC_VECTOR(9 downto 0));   -- current
34     pixel's y coordinate
35 end VGA_driver;
36 --
37 --
38 --Architecture Type:
39 --
40 --
41 architecture behavior of VGA_driver is
42 --
43 
```

```

37 --Signal Declarations:
38 --
39 signal H_video_on    : STD_LOGIC := '0';
40 signal V_video_on    : STD_LOGIC := '0';
41 --Add your signals here
42 signal h_counter     : unsigned(9 downto 0) := (others => '0'); -- horizontal
43 signal v_counter     : unsigned(9 downto 0) := (others => '0'); -- vertical
44
45 --VGA Constants (taken directly from VGA Class Notes)
46 constant left_border : integer := 48; -- back porch
47 constant h_display   : integer := 640; -- active image
48 constant right_border: integer := 16; -- front porch
49 constant h_retrace   : integer := 96; -- Hsync pulse width
50 constant HSCAN      : integer := left_border + h_display
51             + right_border + h_retrace; -- =800
52
53 constant top_border  : integer := 29; -- 29
54 constant v_display   : integer := 480; -- active image
55 constant bottom_border: integer := 10; -- 10
56 constant v_retrace   : integer := 2; -- Vsync pulse width
57 constant VSCAN       : integer := top_border + v_display
58             + bottom_border + v_retrace; -- =521
59
60 BEGIN
61
62 --
63 --Processes:
64 --
65 -- Horizontal counter generating process
66 h_counter_proc : process(vclk_port)
67 begin
68     if rising_edge(vclk_port) then
69         -- counter increments every clk cycle
70         if (h_counter < HSCAN - 1) then
71             h_counter <= h_counter + 1;
72         else
73             h_counter <= (others => '0');
74         end if;
75     end if;
76 end process h_counter_proc;
77
78 -- H_video_on generating process
79 H_video_on_proc : process(h_counter)
80 begin

```

```

81 -- H_video_on is asserted only in the active display area (left-to-
right)
82     if (h_counter < h_display) then
83         H_video_on <= '1';
84     else
85         H_video_on <= '0';
86     end if;
87 end process H_video_on_proc;
88
89 -- Hsync generating process
90 Hsync_proc : process(h_counter)
91 begin
92     -- Hsync is high except for retracing
93     if (h_counter >= h_display + right_border) and (h_counter < h_display
+ right_border + h_retrace) then
94         Hsync_port <= '0';
95     else
96         Hsync_port <= '1';
97     end if;
98 end process Hsync_proc;
99
100 -- Vertical counter generating process
101 v_counter_proc : process(vclk_port)
102 begin
103     if rising_edge(vclk_port) then
104         -- counter increments every time Hsync is asserted
105         if (h_counter = h_display + right_border - 1) then
106             if (v_counter < VSCAN - 1) then
107                 v_counter <= v_counter + 1;
108             else
109                 v_counter <= (others => '0');
110             end if;
111         end if;
112     end if;
113 end process v_counter_proc;
114
115 -- V_video_on generating process
116 V_video_on_proc : process(v_counter)
117 begin
118     -- V_video_on is asserted only in the active display area (top-to-bottom)
119     if (v_counter < v_display) then
120         V_video_on <= '1';
121     else
122         V_video_on <= '0';
123     end if;
124 end process V_video_on_proc;
125
126 -- Vsync generating process
127 Vsync_proc : process(v_counter)
128 begin

```

```
129  -- Vsync is high except for retracing
130  if (v_counter >= v_display + bottom_border) and (v_counter < v_display +
bottom_border + v_retrace) then
131      Vsync_port <= '0';
132  else
133      Vsync_port <= '1';
134  end if;
135 end process Vsync_proc;
136
137 --
138 --Output Mapping:
139 --
140 -- Only enable video out , when H_video_out and V_video_out are high.
141 -- It's important to set the output to zero when you aren't actively
displaying video.
142 -- That's how the monitor determines the black level.
143 video_on_port <= H_video_on and V_video_on;
144 pixel_x_port <= std_logic_vector(h_counter);
145 pixel_y_port <= std_logic_vector(v_counter);
146
147 end behavior;
148
```

Appendix C: game_logic.vhd

```

game_logic.vhd                                         8/20/22, 1:38 PM

1 ---
2 =====
3 --ENGS 31/ CoSc 56
4 --Final Project: Tic-tac-toe in VGA
5 --Game Logic
6 --Instructors: Ben Dobbins, Tad Truex
7 --Student name: Di Luo
8 --
9 --
10 --Library Declarations:
11 --
12 library IEEE;
13 use IEEE.STD_LOGIC_1164.ALL;
14 use IEEE.NUMERIC_STD.ALL;
15 use ieee.math_real.all;
16 library UNISIM;
17 use UNISIM.VComponents.all;
18 --
19 --
20 --Entity Declaration:
21 --
22 entity game_logic is
23     port (
24         -- system clock
25         clk_port      : in std_logic;    -- 25 MHz clock
26         -- inputs from external ports/input conditioning circuits
27         start_port   : in std_logic; -- start a new game
28         reset_port   : in std_logic; -- clear the chessboard
29         drop_port    : in std_logic;   -- put a piece on the selected position
30             up_port      : in std_logic; -- move selecting frame up
31             down_port    : in std_logic; -- move selecting frame down
32             left_port    : in std_logic; -- move selecting frame left
33             right_port   : in std_logic; -- move selecting frame right
34         -- output to pixel generation circuit
35         p1_port      : out std_logic_vector(8 downto 0); -- indicates the
36         blocks that player 1 has dropped a chess piece
37         p2_port      : out std_logic_vector(8 downto 0); -- indicates the
38         blocks that player 2 has dropped a chess piece
39         sf_port      : out std_logic_vector(3 downto 0); -- indicates the
40         location of the selecting frame
41         sf_color_port: out std_logic;                      -- indicates
42         which player is dropping next
43         p1_win_port  : out std_logic_vector(8 downto 0); -- indicates the
44         blocks that make player 1 win

```

```

40    blocks that make player 1 win
40      p2_win_port : out std_logic_vector(8 downto 0)); -- indicates the
blocks that make player 2 win
41  end game_logic;
42
43 --
44 =====
44 --Architecture Type:
45 --
46 =====
46 architecture behavior of game_logic is
47 --
48 =====
48 --Signal & Constant Declarations:
49 --
50 =====
50 -- finite state machine
51 type state_type is (sStart, sPick, sTopLeft1, sTopMid1, sTopRight1,
52                      sCtrLeft1, sCtrMid1, sCtrRight1,
53                      sBtmLeft1, sBtmMid1, sBtmRight1,
54                      sTopLeft2, sTopMid2, sTopRight2,
55                      sCtrLeft2, sCtrMid2, sCtrRight2,
56                      sBtmLeft2, sBtmMid2, sBtmRight2,
57                      sDrop1, sDrop2, sCheck, sTie, sP1_win_0, sP1_win_1,
58                      sP1_win_2, sP1_win_3, sP1_win_4, sP1_win_5, sP1_win_6, sP1_win_7,
58                      sP2_win_0, sP2_win_1,
59                      sP2_win_2, sP2_win_3, sP2_win_4, sP2_win_5, sP2_win_6, sP2_win_7);
59 signal curr_state : state_type := sStart;
60 signal next_state : state_type;
61 -- indicates the blocks that the player has dropped a chess piece
62 signal p1_position, p2_position : unsigned(8 downto 0) := (others => '0');
63 -- indicates the location of selecting frame
64 signal sf_position: unsigned(8 downto 0);
65 -- counter that counts how many chess pieces have been dropped
66 signal counter: unsigned(3 downto 0);
67
68 signal p1_position_update_en: std_logic; -- enable the update of position
of player 1's chess pieces
69 signal p2_position_update_en: std_logic; -- enable the update of position
of player 2's chess pieces
70 signal position_update_reset: std_logic; -- reset position of chess pieces
71
72 signal counter_en: std_logic; -- enable counter to increment
73 signal counter_reset: std_logic; -- reset counter
74 signal counter_pause: std_logic; -- if player dropped on an occupied block
counter would pause
75
76 signal sf_position_idx: std_logic_vector(3 downto 0); -- indexing selecting
frames, 9 blocks so 4 bits
77 .

```

game_logic.vhd

8/20/22, 1:38 PM

```

78 begin
79 --
80 =====
81 --Processes:
82 --
83 -- State update:
84 StateUpdate: process(clk_port)
85 begin
86   if rising_edge(clk_port) then
87     curr_state <= next_state;
88   end if;
89 end process StateUpdate;
90 
91 -- Next state logic:
92 nextStateLogic: process(curr_state, start_port, reset_port, drop_port,
93 up_port, down_port, left_port, right_port, p1_position, p2_position, counter,
94 counter_pause)
95 begin
96   next_state <= curr_state;
97   case curr_state is
98     when sStart =>
99       if reset_port = '0' then
100         if start_port = '1' then
101           next_state <= sPick;
102         end if;
103       end if;
104     when sPick =>
105       if reset_port = '1' then
106         next_state <= sStart;
107       elsif counter = "0000" or counter = "0010" or counter = "0100" or
108       counter = "0110" or counter = "1000" then --player 1's round
109         next_state <= sCtrMid1;
110       elsif counter = "0001" or counter = "0011" or counter = "0101" or
111       counter = "0111" then --player 2's round
112         next_state <= sCtrMid2;
113       else
114         next_state <= sStart; -- counter only goes from 0 to 9 so
115       should never happen
116     end if;
117     when sCtrMid1 =>
118       if reset_port = '1' then
119         next_state <= sStart;
120       else
121         if drop_port = '1' then
122           next_state <= sDrop1;
123         else
124           if up_port = '1' then
125             next_state <= sTopMid1;
126           elsif down_port = '1' then
127             next_state <= sBottomMid1;
128           end if;
129         end if;
130       end if;
131     when sDrop1 =>
132       if reset_port = '1' then
133         next_state <= sStart;
134       else
135         if drop_port = '1' then
136           next_state <= sDrop2;
137         else
138           if up_port = '1' then
139             next_state <= sTopMid2;
140           elsif down_port = '1' then
141             next_state <= sBottomMid2;
142           end if;
143         end if;
144       end if;
145     when sTopMid1 =>
146       if reset_port = '1' then
147         next_state <= sStart;
148       else
149         if drop_port = '1' then
150           next_state <= sDrop1;
151         else
152           if up_port = '1' then
153             next_state <= sTopMid2;
154           elsif down_port = '1' then
155             next_state <= sBottomMid1;
156           end if;
157         end if;
158       end if;
159     when sBottomMid1 =>
160       if reset_port = '1' then
161         next_state <= sStart;
162       else
163         if drop_port = '1' then
164           next_state <= sDrop1;
165         else
166           if up_port = '1' then
167             next_state <= sTopMid1;
168           elsif down_port = '1' then
169             next_state <= sBottomMid2;
170           end if;
171         end if;
172       end if;
173     when sTopMid2 =>
174       if reset_port = '1' then
175         next_state <= sStart;
176       else
177         if drop_port = '1' then
178           next_state <= sDrop2;
179         else
180           if up_port = '1' then
181             next_state <= sTopMid1;
182           elsif down_port = '1' then
183             next_state <= sBottomMid2;
184           end if;
185         end if;
186       end if;
187     when sBottomMid2 =>
188       if reset_port = '1' then
189         next_state <= sStart;
190       else
191         if drop_port = '1' then
192           next_state <= sDrop2;
193         else
194           if up_port = '1' then
195             next_state <= sTopMid2;
196           elsif down_port = '1' then
197             next_state <= sBottomMid1;
198           end if;
199         end if;
200       end if;
201     when sDrop2 =>
202       if reset_port = '1' then
203         next_state <= sStart;
204       else
205         if drop_port = '1' then
206           next_state <= sDrop3;
207         else
208           if up_port = '1' then
209             next_state <= sTopMid3;
210           elsif down_port = '1' then
211             next_state <= sBottomMid3;
212           end if;
213         end if;
214       end if;
215     when sDrop3 =>
216       if reset_port = '1' then
217         next_state <= sStart;
218       else
219         if drop_port = '1' then
220           next_state <= sDrop4;
221         else
222           if up_port = '1' then
223             next_state <= sTopMid4;
224           elsif down_port = '1' then
225             next_state <= sBottomMid4;
226           end if;
227         end if;
228       end if;
229     when sDrop4 =>
230       if reset_port = '1' then
231         next_state <= sStart;
232       else
233         if drop_port = '1' then
234           next_state <= sDrop5;
235         else
236           if up_port = '1' then
237             next_state <= sTopMid5;
238           elsif down_port = '1' then
239             next_state <= sBottomMid5;
240           end if;
241         end if;
242       end if;
243     when sDrop5 =>
244       if reset_port = '1' then
245         next_state <= sStart;
246       else
247         if drop_port = '1' then
248           next_state <= sDrop6;
249         else
250           if up_port = '1' then
251             next_state <= sTopMid6;
252           elsif down_port = '1' then
253             next_state <= sBottomMid6;
254           end if;
255         end if;
256       end if;
257     when sDrop6 =>
258       if reset_port = '1' then
259         next_state <= sStart;
260       else
261         if drop_port = '1' then
262           next_state <= sDrop7;
263         else
264           if up_port = '1' then
265             next_state <= sTopMid7;
266           elsif down_port = '1' then
267             next_state <= sBottomMid7;
268           end if;
269         end if;
270       end if;
271     when sDrop7 =>
272       if reset_port = '1' then
273         next_state <= sStart;
274       else
275         if drop_port = '1' then
276           next_state <= sDrop8;
277         else
278           if up_port = '1' then
279             next_state <= sTopMid8;
280           elsif down_port = '1' then
281             next_state <= sBottomMid8;
282           end if;
283         end if;
284       end if;
285     when sDrop8 =>
286       if reset_port = '1' then
287         next_state <= sStart;
288       else
289         if drop_port = '1' then
290           next_state <= sDrop9;
291         else
292           if up_port = '1' then
293             next_state <= sTopMid9;
294           elsif down_port = '1' then
295             next_state <= sBottomMid9;
296           end if;
297         end if;
298       end if;
299     when sDrop9 =>
300       if reset_port = '1' then
301         next_state <= sStart;
302       else
303         if drop_port = '1' then
304           next_state <= sDrop10;
305         else
306           if up_port = '1' then
307             next_state <= sTopMid10;
308           elsif down_port = '1' then
309             next_state <= sBottomMid10;
310           end if;
311         end if;
312       end if;
313     when sDrop10 =>
314       if reset_port = '1' then
315         next_state <= sStart;
316       else
317         if drop_port = '1' then
318           next_state <= sDrop11;
319         else
320           if up_port = '1' then
321             next_state <= sTopMid11;
322           elsif down_port = '1' then
323             next_state <= sBottomMid11;
324           end if;
325         end if;
326       end if;
327     when sDrop11 =>
328       if reset_port = '1' then
329         next_state <= sStart;
330       else
331         if drop_port = '1' then
332           next_state <= sDrop12;
333         else
334           if up_port = '1' then
335             next_state <= sTopMid12;
336           elsif down_port = '1' then
337             next_state <= sBottomMid12;
338           end if;
339         end if;
340       end if;
341     when sDrop12 =>
342       if reset_port = '1' then
343         next_state <= sStart;
344       else
345         if drop_port = '1' then
346           next_state <= sDrop13;
347         else
348           if up_port = '1' then
349             next_state <= sTopMid13;
350           elsif down_port = '1' then
351             next_state <= sBottomMid13;
352           end if;
353         end if;
354       end if;
355     when sDrop13 =>
356       if reset_port = '1' then
357         next_state <= sStart;
358       else
359         if drop_port = '1' then
360           next_state <= sDrop14;
361         else
362           if up_port = '1' then
363             next_state <= sTopMid14;
364           elsif down_port = '1' then
365             next_state <= sBottomMid14;
366           end if;
367         end if;
368       end if;
369     when sDrop14 =>
370       if reset_port = '1' then
371         next_state <= sStart;
372       else
373         if drop_port = '1' then
374           next_state <= sDrop15;
375         else
376           if up_port = '1' then
377             next_state <= sTopMid15;
378           elsif down_port = '1' then
379             next_state <= sBottomMid15;
380           end if;
381         end if;
382       end if;
383     when sDrop15 =>
384       if reset_port = '1' then
385         next_state <= sStart;
386       else
387         if drop_port = '1' then
388           next_state <= sDrop16;
389         else
390           if up_port = '1' then
391             next_state <= sTopMid16;
392           elsif down_port = '1' then
393             next_state <= sBottomMid16;
394           end if;
395         end if;
396       end if;
397     when sDrop16 =>
398       if reset_port = '1' then
399         next_state <= sStart;
400       else
401         if drop_port = '1' then
402           next_state <= sDrop17;
403         else
404           if up_port = '1' then
405             next_state <= sTopMid17;
406           elsif down_port = '1' then
407             next_state <= sBottomMid17;
408           end if;
409         end if;
410       end if;
411     when sDrop17 =>
412       if reset_port = '1' then
413         next_state <= sStart;
414       else
415         if drop_port = '1' then
416           next_state <= sDrop18;
417         else
418           if up_port = '1' then
419             next_state <= sTopMid18;
420           elsif down_port = '1' then
421             next_state <= sBottomMid18;
422           end if;
423         end if;
424       end if;
425     when sDrop18 =>
426       if reset_port = '1' then
427         next_state <= sStart;
428       else
429         if drop_port = '1' then
430           next_state <= sDrop19;
431         else
432           if up_port = '1' then
433             next_state <= sTopMid19;
434           elsif down_port = '1' then
435             next_state <= sBottomMid19;
436           end if;
437         end if;
438       end if;
439     when sDrop19 =>
440       if reset_port = '1' then
441         next_state <= sStart;
442       else
443         if drop_port = '1' then
444           next_state <= sDrop20;
445         else
446           if up_port = '1' then
447             next_state <= sTopMid20;
448           elsif down_port = '1' then
449             next_state <= sBottomMid20;
450           end if;
451         end if;
452       end if;
453     when sDrop20 =>
454       if reset_port = '1' then
455         next_state <= sStart;
456       else
457         if drop_port = '1' then
458           next_state <= sDrop21;
459         else
460           if up_port = '1' then
461             next_state <= sTopMid21;
462           elsif down_port = '1' then
463             next_state <= sBottomMid21;
464           end if;
465         end if;
466       end if;
467     when sDrop21 =>
468       if reset_port = '1' then
469         next_state <= sStart;
470       else
471         if drop_port = '1' then
472           next_state <= sDrop22;
473         else
474           if up_port = '1' then
475             next_state <= sTopMid22;
476           elsif down_port = '1' then
477             next_state <= sBottomMid22;
478           end if;
479         end if;
480       end if;
481     when sDrop22 =>
482       if reset_port = '1' then
483         next_state <= sStart;
484       else
485         if drop_port = '1' then
486           next_state <= sDrop23;
487         else
488           if up_port = '1' then
489             next_state <= sTopMid23;
490           elsif down_port = '1' then
491             next_state <= sBottomMid23;
492           end if;
493         end if;
494       end if;
495     when sDrop23 =>
496       if reset_port = '1' then
497         next_state <= sStart;
498       else
499         if drop_port = '1' then
500           next_state <= sDrop24;
501         else
502           if up_port = '1' then
503             next_state <= sTopMid24;
504           elsif down_port = '1' then
505             next_state <= sBottomMid24;
506           end if;
507         end if;
508       end if;
509     when sDrop24 =>
510       if reset_port = '1' then
511         next_state <= sStart;
512       else
513         if drop_port = '1' then
514           next_state <= sDrop25;
515         else
516           if up_port = '1' then
517             next_state <= sTopMid25;
518           elsif down_port = '1' then
519             next_state <= sBottomMid25;
520           end if;
521         end if;
522       end if;
523     when sDrop25 =>
524       if reset_port = '1' then
525         next_state <= sStart;
526       else
527         if drop_port = '1' then
528           next_state <= sDrop26;
529         else
530           if up_port = '1' then
531             next_state <= sTopMid26;
532           elsif down_port = '1' then
533             next_state <= sBottomMid26;
534           end if;
535         end if;
536       end if;
537     when sDrop26 =>
538       if reset_port = '1' then
539         next_state <= sStart;
540       else
541         if drop_port = '1' then
542           next_state <= sDrop27;
543         else
544           if up_port = '1' then
545             next_state <= sTopMid27;
546           elsif down_port = '1' then
547             next_state <= sBottomMid27;
548           end if;
549         end if;
550       end if;
551     when sDrop27 =>
552       if reset_port = '1' then
553         next_state <= sStart;
554       else
555         if drop_port = '1' then
556           next_state <= sDrop28;
557         else
558           if up_port = '1' then
559             next_state <= sTopMid28;
560           elsif down_port = '1' then
561             next_state <= sBottomMid28;
562           end if;
563         end if;
564       end if;
565     when sDrop28 =>
566       if reset_port = '1' then
567         next_state <= sStart;
568       else
569         if drop_port = '1' then
570           next_state <= sDrop29;
571         else
572           if up_port = '1' then
573             next_state <= sTopMid29;
574           elsif down_port = '1' then
575             next_state <= sBottomMid29;
576           end if;
577         end if;
578       end if;
579     when sDrop29 =>
580       if reset_port = '1' then
581         next_state <= sStart;
582       else
583         if drop_port = '1' then
584           next_state <= sDrop30;
585         else
586           if up_port = '1' then
587             next_state <= sTopMid30;
588           elsif down_port = '1' then
589             next_state <= sBottomMid30;
590           end if;
591         end if;
592       end if;
593     when sDrop30 =>
594       if reset_port = '1' then
595         next_state <= sStart;
596       else
597         if drop_port = '1' then
598           next_state <= sDrop31;
599         else
600           if up_port = '1' then
601             next_state <= sTopMid31;
602           elsif down_port = '1' then
603             next_state <= sBottomMid31;
604           end if;
605         end if;
606       end if;
607     when sDrop31 =>
608       if reset_port = '1' then
609         next_state <= sStart;
610       else
611         if drop_port = '1' then
612           next_state <= sDrop32;
613         else
614           if up_port = '1' then
615             next_state <= sTopMid32;
616           elsif down_port = '1' then
617             next_state <= sBottomMid32;
618           end if;
619         end if;
620       end if;
621     when sDrop32 =>
622       if reset_port = '1' then
623         next_state <= sStart;
624       else
625         if drop_port = '1' then
626           next_state <= sDrop33;
627         else
628           if up_port = '1' then
629             next_state <= sTopMid33;
630           elsif down_port = '1' then
631             next_state <= sBottomMid33;
632           end if;
633         end if;
634       end if;
635     when sDrop33 =>
636       if reset_port = '1' then
637         next_state <= sStart;
638       else
639         if drop_port = '1' then
640           next_state <= sDrop34;
641         else
642           if up_port = '1' then
643             next_state <= sTopMid34;
644           elsif down_port = '1' then
645             next_state <= sBottomMid34;
646           end if;
647         end if;
648       end if;
649     when sDrop34 =>
650       if reset_port = '1' then
651         next_state <= sStart;
652       else
653         if drop_port = '1' then
654           next_state <= sDrop35;
655         else
656           if up_port = '1' then
657             next_state <= sTopMid35;
658           elsif down_port = '1' then
659             next_state <= sBottomMid35;
660           end if;
661         end if;
662       end if;
663     when sDrop35 =>
664       if reset_port = '1' then
665         next_state <= sStart;
666       else
667         if drop_port = '1' then
668           next_state <= sDrop36;
669         else
670           if up_port = '1' then
671             next_state <= sTopMid36;
672           elsif down_port = '1' then
673             next_state <= sBottomMid36;
674           end if;
675         end if;
676       end if;
677     when sDrop36 =>
678       if reset_port = '1' then
679         next_state <= sStart;
680       else
681         if drop_port = '1' then
682           next_state <= sDrop37;
683         else
684           if up_port = '1' then
685             next_state <= sTopMid37;
686           elsif down_port = '1' then
687             next_state <= sBottomMid37;
688           end if;
689         end if;
690       end if;
691     when sDrop37 =>
692       if reset_port = '1' then
693         next_state <= sStart;
694       else
695         if drop_port = '1' then
696           next_state <= sDrop38;
697         else
698           if up_port = '1' then
699             next_state <= sTopMid38;
700           elsif down_port = '1' then
701             next_state <= sBottomMid38;
702           end if;
703         end if;
704       end if;
705     when sDrop38 =>
706       if reset_port = '1' then
707         next_state <= sStart;
708       else
709         if drop_port = '1' then
710           next_state <= sDrop39;
711         else
712           if up_port = '1' then
713             next_state <= sTopMid39;
714           elsif down_port = '1' then
715             next_state <= sBottomMid39;
716           end if;
717         end if;
718       end if;
719     when sDrop39 =>
720       if reset_port = '1' then
721         next_state <= sStart;
722       else
723         if drop_port = '1' then
724           next_state <= sDrop40;
725         else
726           if up_port = '1' then
727             next_state <= sTopMid40;
728           elsif down_port = '1' then
729             next_state <= sBottomMid40;
730           end if;
731         end if;
732       end if;
733     when sDrop40 =>
734       if reset_port = '1' then
735         next_state <= sStart;
736       else
737         if drop_port = '1' then
738           next_state <= sDrop41;
739         else
740           if up_port = '1' then
741             next_state <= sTopMid41;
742           elsif down_port = '1' then
743             next_state <= sBottomMid41;
744           end if;
745         end if;
746       end if;
747     when sDrop41 =>
748       if reset_port = '1' then
749         next_state <= sStart;
750       else
751         if drop_port = '1' then
752           next_state <= sDrop42;
753         else
754           if up_port = '1' then
755             next_state <= sTopMid42;
756           elsif down_port = '1' then
757             next_state <= sBottomMid42;
758           end if;
759         end if;
760       end if;
761     when sDrop42 =>
762       if reset_port = '1' then
763         next_state <= sStart;
764       else
765         if drop_port = '1' then
766           next_state <= sDrop43;
767         else
768           if up_port = '1' then
769             next_state <= sTopMid43;
770           elsif down_port = '1' then
771             next_state <= sBottomMid43;
772           end if;
773         end if;
774       end if;
775     when sDrop43 =>
776       if reset_port = '1' then
777         next_state <= sStart;
778       else
779         if drop_port = '1' then
780           next_state <= sDrop44;
781         else
782           if up_port = '1' then
783             next_state <= sTopMid44;
784           elsif down_port = '1' then
785             next_state <= sBottomMid44;
786           end if;
787         end if;
788       end if;
789     when sDrop44 =>
790       if reset_port = '1' then
791         next_state <= sStart;
792       else
793         if drop_port = '1' then
794           next_state <= sDrop45;
795         else
796           if up_port = '1' then
797             next_state <= sTopMid45;
798           elsif down_port = '1' then
799             next_state <= sBottomMid45;
800           end if;
801         end if;
802       end if;
803     when sDrop45 =>
804       if reset_port = '1' then
805         next_state <= sStart;
806       else
807         if drop_port = '1' then
808           next_state <= sDrop46;
809         else
810           if up_port = '1' then
811             next_state <= sTopMid46;
812           elsif down_port = '1' then
813             next_state <= sBottomMid46;
814           end if;
815         end if;
816       end if;
817     when sDrop46 =>
818       if reset_port = '1' then
819         next_state <= sStart;
820       else
821         if drop_port = '1' then
822           next_state <= sDrop47;
823         else
824           if up_port = '1' then
825             next_state <= sTopMid47;
826           elsif down_port = '1' then
827             next_state <= sBottomMid47;
828           end if;
829         end if;
830       end if;
831     when sDrop47 =>
832       if reset_port = '1' then
833         next_state <= sStart;
834       else
835         if drop_port = '1' then
836           next_state <= sDrop48;
837         else
838           if up_port = '1' then
839             next_state <= sTopMid48;
840           elsif down_port = '1' then
841             next_state <= sBottomMid48;
842           end if;
843         end if;
844       end if;
845     when sDrop48 =>
846       if reset_port = '1' then
847         next_state <= sStart;
848       else
849         if drop_port = '1' then
850           next_state <= sDrop49;
851         else
852           if up_port = '1' then
853             next_state <= sTopMid49;
854           elsif down_port = '1' then
855             next_state <= sBottomMid49;
856           end if;
857         end if;
858       end if;
859     when sDrop49 =>
860       if reset_port = '1' then
861         next_state <= sStart;
862       else
863         if drop_port = '1' then
864           next_state <= sDrop50;
865         else
866           if up_port = '1' then
867             next_state <= sTopMid50;
868           elsif down_port = '1' then
869             next_state <= sBottomMid50;
870           end if;
871         end if;
872       end if;
873     when sDrop50 =>
874       if reset_port = '1' then
875         next_state <= sStart;
876       else
877         if drop_port = '1' then
878           next_state <= sDrop51;
879         else
880           if up_port = '1' then
881             next_state <= sTopMid51;
882           elsif down_port = '1' then
883             next_state <= sBottomMid51;
884           end if;
885         end if;
886       end if;
887     when sDrop51 =>
888       if reset_port = '1' then
889         next_state <= sStart;
890       else
891         if drop_port = '1' then
892           next_state <= sDrop52;
893         else
894           if up_port = '1' then
895             next_state <= sTopMid52;
896           elsif down_port = '1' then
897             next_state <= sBottomMid52;
898           end if;
899         end if;
900       end if;
901     when sDrop52 =>
902       if reset_port = '1' then
903         next_state <= sStart;
904       else
905         if drop_port = '1' then
906           next_state <= sDrop53;
907         else
908           if up_port = '1' then
909             next_state <= sTopMid53;
910           elsif down_port = '1' then
911             next_state <= sBottomMid53;
912           end if;
913         end if;
914       end if;
915     when sDrop53 =>
916       if reset_port = '1' then
917         next_state <= sStart;
918       else
919         if drop_port = '1' then
920           next_state <= sDrop54;
921         else
922           if up_port = '1' then
923             next_state <= sTopMid54;
924           elsif down_port = '1' then
925             next_state <= sBottomMid54;
926           end if;
927         end if;
928       end if;
929     when sDrop54 =>
930       if reset_port = '1' then
931         next_state <= sStart;
932       else
933         if drop_port = '1' then
934           next_state <= sDrop55;
935         else
936           if up_port = '1' then
937             next_state <= sTopMid55;
938           elsif down_port = '1' then
939             next_state <= sBottomMid55;
940           end if;
941         end if;
942       end if;
943     when sDrop55 =>
944       if reset_port = '1' then
945         next_state <= sStart;
946       else
947         if drop_port = '1' then
948           next_state <= sDrop56;
949         else
950           if up_port = '1' then
951             next_state <= sTopMid56;
952           elsif down_port = '1' then
953             next_state <= sBottomMid56;
954           end if;
955         end if;
956       end if;
957     when sDrop56 =>
958       if reset_port = '1' then
959         next_state <= sStart;
960       else
961         if drop_port = '1' then
962           next_state <= sDrop57;
963         else
964           if up_port = '1' then
965             next_state <= sTopMid57;
966           elsif down_port = '1' then
967             next_state <= sBottomMid57;
968           end if;
969         end if;
970       end if;
971     when sDrop57 =>
972       if reset_port = '1' then
973         next_state <= sStart;
974       else
975         if drop_port = '1' then
976           next_state <= sDrop58;
977         else
978           if up_port = '1' then
979             next_state <= sTopMid58;
980           elsif down_port = '1' then
981             next_state <= sBottomMid58;
982           end if;
983         end if;
984       end if;
985     when sDrop58 =>
986       if reset_port = '1' then
987         next_state <= sStart;
988       else
989         if drop_port = '1' then
990           next_state <= sDrop59;
991         else
992           if up_port = '1' then
993             next_state <= sTopMid59;
994           elsif down_port = '1' then
995             next_state <= sBottomMid59;
996           end if;
997         end if;
998       end if;
999     when sDrop59 =>
1000       if reset_port = '1' then
1001         next_state <= sStart;
1002       else
1003         if drop_port = '1' then
1004           next_state <= sDrop60;
1005         else
1006           if up_port = '1' then
1007             next_state <= sTopMid60;
1008           elsif down_port = '1' then
1009             next_state <= sBottomMid60;
1010           end if;
1011         end if;
1012       end if;
1013     when sDrop60 =>
1014       if reset_port = '1' then
1015         next_state <= sStart;
1016       else
1017         if drop_port = '1' then
1018           next_state <= sDrop61;
1019         else
1020           if up_port = '1' then
1021             next_state <= sTopMid61;
1022           elsif down_port = '1' then
1023             next_state <= sBottomMid61;
1024           end if;
1025         end if;
1026       end if;
1027     when sDrop61 =>
1028       if reset_port = '1' then
1029         next_state <= sStart;
1030       else
1031         if drop_port = '1' then
1032           next_state <= sDrop62;
1033         else
1034           if up_port = '1' then
1035             next_state <= sTopMid62;
1036           elsif down_port = '1' then
1037             next_state <= sBottomMid62;
1038           end if;
1039         end if;
1040       end if;
1041     when sDrop62 =>
1042       if reset_port = '1' then
1043         next_state <= sStart;
1044       else
1045         if drop_port = '1' then
1046           next_state <= sDrop63;
1047         else
1048           if up_port = '1' then
1049             next_state <= sTopMid63;
1050           elsif down_port = '1' then
1051             next_state <= sBottomMid63;
1052           end if;
1053         end if;
1054       end if;
1055     when sDrop63 =>
1056       if reset_port = '1' then
1057         next_state <= sStart;
1058       else
1059         if drop_port = '1' then
1060           next_state <= sDrop64;
1061         else
1062           if up_port = '1' then
1063             next_state <= sTopMid64;
1064           elsif down_port = '1' then
1065             next_state <= sBottomMid64;
1066           end if;
1067         end if;
1068       end if;
1069     when sDrop64 =>
1070       if reset_port = '1' then
1071         next_state <= sStart;
1072       else
1073         if drop_port = '1' then
1074           next_state <= sDrop65;
1075         else
1076           if up_port = '1' then
1077             next_state <
```

```

121      next_state <= sBtmMid1;
122      elsif left_port = '1' then
123          next_state <= sCtrLeft1;
124      elsif right_port = '1' then
125          next_state <= sCtrRight1;
126      end if;
127          end if;
128      end if;
129  when sCtrMid2 =>
130      if reset_port = '1' then
131          next_state <= sStart;
132      else
133          if drop_port = '1' then
134              next_state <= sDrop2;
135          else
136              if up_port = '1' then
137                  next_state <= sTopMid2;
138              elsif down_port = '1' then
139                  next_state <= sBtmMid2;
140              elsif left_port = '1' then
141                  next_state <= sCtrLeft2;
142              elsif right_port = '1' then
143                  next_state <= sCtrRight2;
144              end if;
145          end if;
146      end if;
147  when sCtrLeft1 =>
148      if reset_port = '1' then
149          next_state <= sStart;
150      else
151          if drop_port = '1' then
152              next_state <= sDrop1;
153          else
154              if up_port = '1' then
155                  next_state <= sTopLeft1;
156              elsif down_port = '1' then
157                  next_state <= sBtmLeft1;
158              elsif right_port = '1' then
159                  next_state <= sCtrMid1;
160              end if;
161          end if;
162      end if;
163  when sCtrLeft2 =>
164      if reset_port = '1' then
165          next_state <= sStart;
166      else
167          if drop_port = '1' then
168              next_state <= sDrop2;
169          else
170              if up_port = '1' then

```

```

-- -- -- -- --
171      next_state <= sTopLeft2;
172      elsif down_port = '1' then
173          next_state <= sBtmLeft2;
174      elsif right_port = '1' then
175          next_state <= sCtrMid2;
176      end if;
177      end if;
178  end if;
179  when sCtrRight1 =>
180      if reset_port = '1' then
181          next_state <= sStart;
182      else
183          if drop_port = '1' then
184              next_state <= sDrop1;
185          else
186              if up_port = '1' then
187                  next_state <= sTopRight1;
188              elsif down_port = '1' then
189                  next_state <= sBtmRight1;
190              elsif left_port = '1' then
191                  next_state <= sCtrMid1;
192              end if;
193          end if;
194      end if;
195  when sCtrRight2 =>
196      if reset_port = '1' then
197          next_state <= sStart;
198      else
199          if drop_port = '1' then
200              next_state <= sDrop2;
201          else
202              if up_port = '1' then
203                  next_state <= sTopRight2;
204              elsif down_port = '1' then
205                  next_state <= sBtmRight2;
206              elsif left_port = '1' then
207                  next_state <= sCtrMid2;
208              end if;
209          end if;
210      end if;
211  when sTopMid1 =>
212      if reset_port = '1' then
213          next_state <= sStart;
214      else
215          if drop_port = '1' then
216              next_state <= sDrop1;
217          else
218              if down_port = '1' then
219                  next_state <= sCtrMid1;
220              elsif left_port = '1' then

```

```

220      elsif left_port = '1' then
221          next_state <= sTopLeft1;
222      elsif right_port = '1' then
223          next_state <= sTopRight1;
224      end if;
225      end if;
226  end if;
227 when sTopMid2 =>
228     if reset_port = '1' then
229         next_state <= sStart;
230     else
231         if drop_port = '1' then
232             next_state <= sDrop2;
233         else
234             if down_port = '1' then
235                 next_state <= sCtrMid2;
236             elsif left_port = '1' then
237                 next_state <= sTopLeft2;
238             elsif right_port = '1' then
239                 next_state <= sTopRight2;
240             end if;
241         end if;
242     end if;
243 when sTopLeft1 =>
244     if reset_port = '1' then
245         next_state <= sStart;
246     else
247         if drop_port = '1' then
248             next_state <= sDrop1;
249         else
250             if down_port = '1' then
251                 next_state <= sCtrLeft1;
252             elsif right_port = '1' then
253                 next_state <= sTopMid1;
254             end if;
255         end if;
256     end if;
257 when sTopLeft2 =>
258     if reset_port = '1' then
259         next_state <= sStart;
260     else
261         if drop_port = '1' then
262             next_state <= sDrop2;
263         else
264             if down_port = '1' then
265                 next_state <= sCtrLeft2;
266             elsif right_port = '1' then
267                 next_state <= sTopMid2;
268             end if;
269         end if;

```

```

270      end if;
271  when sTopRight1 =>
272    if reset_port = '1' then
273      next_state <= sStart;
274    else
275      if drop_port = '1' then
276        next_state <= sDrop1;
277      else
278        if down_port = '1' then
279          next_state <= sCtrRight1;
280        elsif left_port = '1' then
281          next_state <= sTopMid1;
282        end if;
283      end if;
284    end if;
285  when sTopRight2 =>
286    if reset_port = '1' then
287      next_state <= sStart;
288    else
289      if drop_port = '1' then
290        next_state <= sDrop2;
291      else
292        if down_port = '1' then
293          next_state <= sCtrRight2;
294        elsif left_port = '1' then
295          next_state <= sTopMid2;
296        end if;
297      end if;
298    end if;
299  when sBtmMid1 =>
300    if reset_port = '1' then
301      next_state <= sStart;
302    else
303      if drop_port = '1' then
304        next_state <= sDrop1;
305      else
306        if up_port = '1' then
307          next_state <= sCtrMid1;
308        elsif left_port = '1' then
309          next_state <= sBtmLeft1;
310        elsif right_port = '1' then
311          next_state <= sBtmRight1;
312        end if;
313      end if;
314    end if;
315  when sBtmMid2 =>
316    if reset_port = '1' then
317      next_state <= sStart;
318    else
319      if drop_port = '1' then

```

```

320      next_state <= sDrop2;
321  else
322      if up_port = '1' then
323          next_state <= sCtrMid2;
324      elsif left_port = '1' then
325          next_state <= sBtmLeft2;
326      elsif right_port = '1' then
327          next_state <= sBtmRight2;
328      end if;
329  end if;
330 end if;
331 when sBtmLeft1 =>
332     if reset_port = '1' then
333         next_state <= sStart;
334     else
335         if drop_port = '1' then
336             next_state <= sDrop1;
337         else
338             if up_port = '1' then
339                 next_state <= sCtrLeft1;
340             elsif right_port = '1' then
341                 next_state <= sBtmMid1;
342             end if;
343         end if;
344     end if;
345 when sBtmLeft2 =>
346     if reset_port = '1' then
347         next_state <= sStart;
348     else
349         if drop_port = '1' then
350             next_state <= sDrop2;
351         else
352             if up_port = '1' then
353                 next_state <= sCtrLeft2;
354             elsif right_port = '1' then
355                 next_state <= sBtmMid2;
356             end if;
357         end if;
358     end if;
359 when sBtmRight1 =>
360     if reset_port = '1' then
361         next_state <= sStart;
362     else
363         if drop_port = '1' then
364             next_state <= sDrop1;
365         else
366             if up_port = '1' then
367                 next_state <= sCtrRight1;
368             elsif left_port = '1' then
369                 next_state <= sRtmMid1;

```

```

370      end if;
371      end if;
372    end if;
373  when sBtmRight2 =>
374    if reset_port = '1' then
375      next_state <= sStart;
376    else
377      if drop_port = '1' then
378        next_state <= sDrop2;
379      else
380        if up_port = '1' then
381          next_state <= sCtrRight2;
382        elsif left_port = '1' then
383          next_state <= sBtmMid2;
384        end if;
385      end if;
386    end if;
387  when sDrop1 =>
388    if reset_port = '1' then
389      next_state <= sStart;
390    else
391      next_state <= sCheck;
392    end if;
393  when sDrop2 =>
394    if reset_port = '1' then
395      next_state <= sStart;
396    else
397      next_state <= sCheck;
398    end if;
399  when sCheck =>
400    if (p1_position(0) = '1' and p1_position(1) = '1' and
401        p1_position(2) = '1') then
402      next_state <= sP1_win_0;
403    elsif (p1_position(3) = '1' and p1_position(4) = '1' and
404           p1_position(5) = '1') then
405      next_state <= sP1_win_1;
406    elsif (p1_position(6) = '1' and p1_position(7) = '1' and
407           p1_position(8) = '1') then
408      next_state <= sP1_win_2;
409    elsif (p1_position(0) = '1' and p1_position(3) = '1' and
410           p1_position(6) = '1') then
411      next_state <= sP1_win_3;
412    elsif (p1_position(1) = '1' and p1_position(4) = '1' and
413           p1_position(7) = '1') then
414      next_state <= sP1_win_4;
415    elsif (p1_position(2) = '1' and p1_position(5) = '1' and
416           p1_position(8) = '1') then
417      next_state <= sP1_win_5;
418    elsif (p1_position(0) = '1' and p1_position(4) = '1' and
419           p1_position(8) = '1') then
420      next_state <= sP1_win_6;
421    end if;

```

```

413    p1_position(8) = '1' ) then
414        next_state <= sP1_win_6;
415    elsif (p1_position(2) = '1' and p1_position(4) = '1' and
416    p1_position(6) = '1') then
417        next_state <= sP1_win_7;
418    elsif (p2_position(0) = '1' and p2_position(1) = '1' and
419    p2_position(2) = '1') then
420        next_state <= sP2_win_0;
421    elsif (p2_position(3) = '1' and p2_position(4) = '1' and
422    p2_position(5) = '1') then
423        next_state <= sP2_win_1;
424    elsif (p2_position(6) = '1' and p2_position(7) = '1' and
425    p2_position(8) = '1') then
426        next_state <= sP2_win_2;
427    elsif (p2_position(0) = '1' and p2_position(3) = '1' and
428    p2_position(6) = '1') then
429        next_state <= sP2_win_3;
430    elsif (p2_position(1) = '1' and p2_position(4) = '1' and
431    p2_position(7) = '1') then
432        next_state <= sP2_win_4;
433    elsif (p2_position(2) = '1' and p2_position(5) = '1' and
434    p2_position(8) = '1') then
435        next_state <= sP2_win_5;
436    elsif (p2_position(0) = '1' and p2_position(4) = '1' and
437    p2_position(8) = '1') then
438        next_state <= sP2_win_6;
439    elsif (p2_position(2) = '1' and p2_position(4) = '1' and
440    p2_position(6) = '1') then
441        next_state <= sP2_win_7;
442    elsif counter = "1001" then -- if counter goes to 9, meaning
443        there is no free space on the chessboard
444        next_state <= sTie;
445    else
446        next_state <= sPick;
447    end if;
448    when others =>
449        if reset_port = '1' then
450            next_state <= sStart;
451        end if;
452    end case;
453 end process NextStateLogic;
454
455 -- Output logic:
456 OutputLogic: process(curr_state)
457 begin
458     -- default values
459     p1_win_port <= (others => '0');
460     p2_win_port <= (others => '0');
461     p1_position_update_en <= '0';
462     p2_position_update_en <= '0';

```

```

452 position_update_reset <= '0';
453 counter_en <= '0';
454 counter_reset <= '0';
455 sf_position_idx <= "1001";
456 sf_color_port <= '0';
457 -- output in each state
458 case curr_state is
459 when sStart =>
460     position_update_reset <= '1';
461     counter_reset <= '1';
462     When sPick =>
463     When sTopLeft1 =>
464         sf_position_idx <= "0000";
465     When sTopMid1 =>
466         sf_position_idx <= "0001";
467     When sTopRight1 =>
468         sf_position_idx <= "0010";
469     When sCtrLeft1 =>
470         sf_position_idx <= "0011";
471     When sCtrMid1 =>
472         sf_position_idx <= "0100";
473     When sCtrRight1 =>
474         sf_position_idx <= "0101";
475     When sBtmLeft1 =>
476         sf_position_idx <= "0110";
477     When sBtmMid1 =>
478         sf_position_idx <= "0111";
479     When sBtmRight1 =>
480         sf_position_idx <= "1000";
481     When sTopLeft2 =>
482         sf_position_idx <= "0000";
483         sf_color_port <= '1';
484     When sTopMid2 =>
485         sf_position_idx <= "0001";
486         sf_color_port <= '1';
487     When sTopRight2 =>
488         sf_position_idx <= "0010";
489         sf_color_port <= '1';
490     When sCtrLeft2 =>
491         sf_position_idx <= "0011";
492         sf_color_port <= '1';
493     When sCtrMid2 =>
494         sf_position_idx <= "0100";
495         sf_color_port <= '1';
496     When sCtrRight2 =>
497         sf_position_idx <= "0101";
498         sf_color_port <= '1';
499     When sBtmLeft2 =>
500         sf_position_idx <= "0110";
501         sf_color_port <= '1';

```

```

502      When sBtmMid2 =>
503          sf_position_idx <= "0111";
504          sf_color_port <= '1';
505      When sBtmRight2 =>
506          sf_position_idx <= "1000";
507          sf_color_port <= '1';
508      When sDrop1 =>
509          p1_position_update_en <= '1';
510          counter_en <= '1';
511      When sDrop2 =>
512          p2_position_update_en <= '1';
513          counter_en <= '1';
514      When sCheck =>
515      When sP1_win_0 =>
516          p1_win_port <= "000000111";
517      When sP1_win_1 =>
518          p1_win_port <= "000111000";
519      When sP1_win_2 =>
520          p1_win_port <= "111000000";
521      When sP1_win_3 =>
522          p1_win_port <= "001001001";
523      When sP1_win_4 =>
524          p1_win_port <= "010010010";
525      When sP1_win_5 =>
526          p1_win_port <= "100100100";
527      When sP1_win_6 =>
528          p1_win_port <= "100010001";
529      When sP1_win_7 =>
530          p1_win_port <= "001010100";
531      When sP2_win_0 =>
532          p2_win_port <= "000000111";
533      When sP2_win_1 =>
534          p2_win_port <= "000111000";
535      When sP2_win_2 =>
536          p2_win_port <= "111000000";
537      When sP2_win_3 =>
538          p2_win_port <= "001001001";
539      When sP2_win_4 =>
540          p2_win_port <= "010010010";
541      When sP2_win_5 =>
542          p2_win_port <= "100100100";
543      When sP2_win_6 =>
544          p2_win_port <= "100010001";
545      When sP2_win_7 =>
546          p2_win_port <= "001010100";
547      When others => null;
548  end case;
549 end process OutputLogic;
550
551 -- Chess piece positions register file.

```

```

-- Chess piece positions register file:
552 PositionUpdate: process(clk_port, position_update_reset,
553 p1_position_update_en, p2_position_update_en, sf_position)
554 begin
555   if rising_edge(clk_port) then
556     if position_update_reset = '1' then
557       p1_position <= (others => '0');
558       p2_position <= (others => '0');
559     else
560       if (p1_position and sf_position) = "000000000" and (p2_position
561 and sf_position) = "000000000" then -- player 1/2 is not dropping on the spot
562 that has already been dropped
563       counter_pause <= '0';
564       if p1_position_update_en = '1' then
565         p1_position <= p1_position + sf_position;
566       elsif p2_position_update_en = '1' then
567         p2_position <= p2_position + sf_position;
568       end if;
569     else
570       counter_pause <= '1';
571     end if;
572   end if;
573 end process PositionUpdate;
574
-- Counter register file:
574 CounterUpdate: process(clk_port, counter_reset, counter_en, counter_pause)
575 begin
576   if rising_edge(clk_port) then
577     if counter_reset = '1' then
578       counter <= "0000";
579     elsif counter_pause = '0' then
580       if counter_en = '1' then
581         counter <= counter + 1;
582       end if;
583     end if;
584   end if;
585 end process CounterUpdate;
586
-- Selecting frame register file:
587 SelectingFrameUpdate: process(clk_port, sf_position_idx)
588 begin
589   if rising_edge(clk_port) then
590     case sf_position_idx is
591       when "0000" =>
592         sf_position <= "000000001";
593       when "0001" =>
594         sf_position <= "000000010";
595       when "0010" =>
596         sf_position <= "000000100";
597       ...
598     end case;
599   end if;
600 end process SelectingFrameUpdate;

```

game_logic.vhd

8/20/22, 1:38 PM

```

598      when "0011" =>
599          sf_position <= "000001000";
600      when "0100" =>
601          sf_position <= "000010000";
602      when "0101" =>
603          sf_position <= "000100000";
604      when "0110" =>
605          sf_position <= "001000000";
606      when "0111" =>
607          sf_position <= "010000000";
608      when "1000" =>
609          sf_position <= "100000000";
610      when others =>
611          null;
612      end case;
613  end if;
614 end process SelectingFrameUpdate;
615
616 --
617 --Output Mapping:
618 --
619 p1_port <= std_logic_vector(p1_position);
620 p2_port <= std_logic_vector(p2_position);
621 sf_port <= sf_position_idx;
622
623 end behavior;
624

```

Appendix D: pixel_generation.vhd

```

pixel_generation.vhd                                         8/20/22, 1:38 PM

1 ---
2 =====
3 --ENGS 31/ CoSc 56
4 --Final Project: Tic-tac-toe in VGA
5 --Pixel Generation Circuit
6 --Instructors: Ben Dobbins, Tad Truex
7 --Student name: Di Luo
8 --
9 --
10 --Library Declarations:
11 --
12 library IEEE;
13 use IEEE.STD_LOGIC_1164.ALL;
14 use IEEE.NUMERIC_STD.ALL;
15 use ieee.math_real.all;
16 --
17 --
18 --Entity Declaration:
19 --
20 entity pixel_generation is
21     port (
22         -- inputs from VGA_driver
23         pixel_x_port      : in std_logic_vector(9 downto 0);      -- current
24         pixel's x coordinate
25         pixel_y_port      : in std_logic_vector(9 downto 0);      -- current
26         pixel's y coordinate
27         video_on_port    : in std_logic;    -- accomplish blanking. 1 when
28         inside display area. 0 when outside display area
29         -- inputs from game_logic
30         p1_port          : in std_logic_vector(8 downto 0);      -- indicates
31         the blocks that player 1 has dropped a chess piece
32         p2_port          : in std_logic_vector(8 downto 0);      -- indicates
33         the blocks that player 2 has dropped a chess piece
34         sf_port          : in std_logic_vector(3 downto 0);      -- indicates
35         the location of the selecting frame
36         sf_color_port    : in std_logic;                      -- indicates
37         which player is dropping next
38         p1_win_port      : in std_logic_vector(8 downto 0);      -- indicates
39         the blocks that make player 1 win
40         p2_win_port      : in std_logic_vector(8 downto 0);      -- indicates
41         the blocks that make player 2 win
42         -- output to VGA monitor
43         color_port       : out std_logic_vector(11 downto 0)); -- 12 bits
44         for color coding. 4 for each of R,G,B

```

pixel_generation.vhd

8/20/22, 1:38 PM

```

35    for color_coding, 4 for each of RGB
36    end pixel_generation;
37
38    --
39    =====
40    --Architecture Type:
41    =====
42    --Signal & Constant Declarations:
43
44    -- Predefined 12-bit colors
45    constant BLACK : std_logic_vector(11 downto 0) := "000000000000";    --
46    100% black
47    constant RED75 : std_logic_vector(11 downto 0) := "100000000000";    --
48    75% red
49    constant BLUE75 : std_logic_vector(11 downto 0) := "000000001000";    --
50    75% blue
51    constant GREEN75: std_logic_vector(11 downto 0) := "000010000000";    --
52    75% green
53    constant WHITE75: std_logic_vector(11 downto 0) := "110011001100";    --
54    50% white
55
56    -- coordinates of centers in each block
57    constant x_center_0 : integer := 106;
58    constant y_center_0 : integer := 80;
59    constant x_center_1 : integer := 319;
60    constant y_center_1 : integer := 240;
61    constant x_center_2 : integer := 533;
62    constant y_center_2 : integer := 400;
63
64    signal row, column : unsigned(9 downto 0);
65
66 begin
67    row <= unsigned(pixel_y_port);
68    column <= unsigned(pixel_x_port);
69
70    --
71    =====
72    --Processes:
73    --
74    =====
75    pixel_rendering: process(row, column, p1_port, p2_port, sf_port,
76    sf_color_port, video_on_port, p1_win_port, p2_win_port)
77    begin
78        if video_on_port = '1' then
79            if row > 0 and row < 160 and column > 0 and column < 213 then

```

```

72      if p1_win_port(0) = '1' then
73          color_port <= RED75;
74      elsif p2_win_port(0) = '1' then
75          color_port <= BLUE75;
76      else
77          if sf_port = "0000" then
78              if sf_color_port = '0' then
79                  color_port <= RED75;
80              else
81                  color_port <= BLUE75;
82              end if;
83          if row > 10 and row < 150 and column > 10 and column <
203 then
84              case p1_port(0) is
85                  when '1' =>
86                      if ((column + y_center_0 = row + x_center_0)
or (column + row = x_center_0 + y_center_0)) and column > x_center_0 - 50 and
column < x_center_0 + 50 and row > y_center_0 - 50 and row < y_center_0 + 50
then
87                          color_port <= RED75;
88                      else
89                          color_port <= WHITE75;
90                      end if;
91                  when others =>
92                      case p2_port(0) is
93                          when '1' =>
94                              if (to_integer(column) - x_center_0)*
(to_integer(column) - x_center_0) + (to_integer(row) - y_center_0)*
(to_integer(row) - y_center_0) > 49*49 and (to_integer(column) - x_center_0)*
(to_integer(column) - x_center_0) + (to_integer(row) - y_center_0)*
(to_integer(row) - y_center_0) < 50*50 then
95                                  color_port <= BLUE75;
96                              else
97                                  color_port <= WHITE75;
98                              end if;
99                          when others =>
100                              color_port <= WHITE75;
101                      end case;
102                  end case;
103              end if;
104          else
105              case p1_port(0) is
106                  when '1' =>
107                      if ((column + y_center_0 = row + x_center_0) or
(column + row = x_center_0 + y_center_0)) and column > x_center_0 - 50 and
column < x_center_0 + 50 and row > y_center_0 - 50 and row < y_center_0 + 50
then
108                          color_port <= RED75;
109                      else
110                          color_port <= WHITE75;

```

```

111         end if;
112     when others =>
113         case p2_port(0) is
114             when '1' =>
115                 if (to_integer(column) - x_center_0)*
116                     (to_integer(column) - x_center_0) + (to_integer(row) - y_center_0)*
117                     (to_integer(row) - y_center_0) > 49*49 and (to_integer(column) - x_center_0)*
118                     (to_integer(column) - x_center_0) + (to_integer(row) - y_center_0)*
119                     (to_integer(row) - y_center_0) < 50*50 then
120                         color_port <= BLUE75;
121                     else
122                         color_port <= WHITE75;
123                     end if;
124             when others =>
125                 color_port <= WHITE75;
126         end case;
127     end if;
128 end if;
129 elsif row > 0 and row < 160 and column > 213 and column < 426 then
130     if p1_win_port(1) = '1' then
131         color_port <= RED75;
132     elsif p2_win_port(1) = '1' then
133         color_port <= BLUE75;
134     else
135         if sf_port = "0001" then
136             if sf_color_port = '0' then
137                 color_port <= RED75;
138             else
139                 color_port <= BLUE75;
140             end if;
141         if row > 10 and row < 150 and column > 223 and column <
142             416 then
143             case p1_port(1) is
144                 when '1' =>
145                     if ((column + y_center_0 = row + x_center_1) or (column + row = x_center_1 + y_center_0)) and column > x_center_1 - 50 and
146                     column < x_center_1 + 50 and row > y_center_0 - 50 and row < y_center_0 + 50
147                     then
148                         color_port <= RED75;
149                     else
150                         color_port <= WHITE75;
151                     end if;
152             when others =>
153                 case p2_port(1) is
154                     when '1' =>
155                         if (to_integer(column) - x_center_1)*
156                             (to_integer(column) - x_center_1) + (to_integer(row) - y_center_0)*
157                             (to_integer(row) - y_center_0) > 49*49 and (to_integer(column) - x_center_1)*
158                             (to_integer(column) - x_center_1) + (to_integer(row) - y_center_0)*
159

```

pixel_generation.vhd

8/20/22, 1:38 PM

```

150      (to_integer(row) - y_center_0) < 50*50 then
151          color_port <= BLUE75;
152      else
153          color_port <= WHITE75;
154      end if;
155      when others =>
156          color_port <= WHITE75;
157      end case;
158  end if;
159  else
160      case p1_port(1) is
161          when '1' =>
162              if ((column + y_center_0 = row + x_center_1) or
163                  (column + row = x_center_1 + y_center_0)) and column > x_center_1 - 50 and
164                  column < x_center_1 + 50 and row > y_center_0 - 50 and row < y_center_0 + 50
165              then
166                  color_port <= RED75;
167              else
168                  color_port <= WHITE75;
169              end if;
170          when others =>
171              case p2_port(1) is
172                  when '1' =>
173                      if (to_integer(column) - x_center_1)*
174                          (to_integer(column) - x_center_1) + (to_integer(row) - y_center_0)*
175                          (to_integer(row) - y_center_0) > 49*49 and (to_integer(column) - x_center_1)*
176                          (to_integer(column) - x_center_1) + (to_integer(row) - y_center_0)*
177                          (to_integer(row) - y_center_0) < 50*50 then
178                              color_port <= BLUE75;
179                          else
180                              color_port <= WHITE75;
181                          end if;
182                      when others =>
183                          color_port <= WHITE75;
184                      end case;
185                  end if;
186              end if;
187          elsif row > 0 and row < 160 and column > 426 and column < 640 then
188              if p1_win_port(2) = '1' then
189                  color_port <= RED75;
190              elsif p2_win_port(2) = '1' then
191                  color_port <= BLUE75;
192              else
193                  if sf_port = "0010" then
194                      if sf_color_port = '0' then
195                          color_port <= RED75;
196                      else
197                          color_port <= BLUE75;
198                      end if;
199                  end if;
200              end if;
201          end if;

```

```

191           color_port <= BLUE75;
192       end if;
193       if row > 10 and row < 150 and column > 436 and column <
194       630 then
195           case p1_port(2) is
196               when '1' =>
197                   if ((column + y_center_0 = row + x_center_2)
198                       or (column + row = x_center_2 + y_center_0)) and column > x_center_2 - 50 and
199                       column < x_center_2 + 50 and row > y_center_0 - 50 and row < y_center_0 + 50
200                       then
201                           color_port <= RED75;
202                       else
203                           color_port <= WHITE75;
204                       end if;
205               when others =>
206                   case p2_port(2) is
207                       when '1' =>
208                           if (to_integer(column) - x_center_2)*
209                               (to_integer(column) - x_center_2) + (to_integer(row) - y_center_0)*
210                               (to_integer(row) - y_center_0) > 49*49 and (to_integer(column) - x_center_2)*
211                               (to_integer(column) - x_center_2) + (to_integer(row) - y_center_0)*
212                               (to_integer(row) - y_center_0) < 50*50 then
213                                   color_port <= BLUE75;
214                               else
215                                   color_port <= WHITE75;
216                               end if;
217                           when others =>
218                               color_port <= WHITE75;
219                           end case;
220                       end if;
221                   else
222                       case p1_port(2) is
223                           when '1' =>
224                               if ((column + y_center_0 = row + x_center_2) or
225                                   (column + row = x_center_2 + y_center_0)) and column > x_center_2 - 50 and
226                                   column < x_center_2 + 50 and row > y_center_0 - 50 and row < y_center_0 + 50
227                                   then
228                                       color_port <= RED75;
229                                   else
230                                       color_port <= WHITE75;
231                                   end if;
232                           when others =>
233                               case p2_port(2) is
234                                   when '1' =>
235                                       if (to_integer(column) - x_center_2)*
236                               (to_integer(column) - x_center_2) + (to_integer(row) - y_center_0)*
237                               (to_integer(row) - y_center_0) > 49*49 and (to_integer(column) - x_center_2)*
238                               (to_integer(column) - x_center_2) + (to_integer(row) - y_center_0)*
239                               (to_integer(row) - y_center_0) < 50*50 then

```

```

226          color_port <= BLUE75;
227      else
228          color_port <= WHITE75;
229      end if;
230      when others =>
231          color_port <= WHITE75;
232      end case;
233  end case;
234  end if;
235 end if;
236 elsif row > 160 and row < 320 and column > 0 and column < 213 then
237     if p1_win_port(3) = '1' then
238         color_port <= RED75;
239     elsif p2_win_port(3) = '1' then
240         color_port <= BLUE75;
241     else
242         if sf_port = "0011" then
243             if sf_color_port = '0' then
244                 color_port <= RED75;
245             else
246                 color_port <= BLUE75;
247             end if;
248         if row > 170 and row < 310 and column > 10 and column <
249 203 then
250             case p1_port(3) is
251                 when '1' =>
252                     if ((column + y_center_1 = row + x_center_0)
253 or (column + row = x_center_0 + y_center_1)) and column > x_center_0 - 50 and
254 column < x_center_0 + 50 and row > y_center_1 - 50 and row < y_center_1 + 50
255 then
256                     color_port <= RED75;
257                 else
258                     color_port <= WHITE75;
259                 end if;
260             when others =>
261                 case p2_port(3) is
262                     when '1' =>
263                         if (to_integer(column) - x_center_0)*
264 (to_integer(column) - x_center_0) + (to_integer(row) - y_center_1)*
265 (to_integer(row) - y_center_1) > 49*49 and (to_integer(column) - x_center_0)*
266 (to_integer(column) - x_center_0) + (to_integer(row) - y_center_1)*
267 (to_integer(row) - y_center_1) < 50*50 then
268                     color_port <= BLUE75;
269                 else
270                     color_port <= WHITE75;
271                 end if;
272             when others =>
273                 color_port <= WHITE75;
274             end case;
275         end case;
276     end case;

```

```

268      end if;
269  else
270      case p1_port(3) is
271          when '1' =>
272              if ((column + y_center_1 = row + x_center_0) or
273 (column + row = x_center_0 + y_center_1)) and column > x_center_0 - 50 and
274 column < x_center_0 + 50 and row > y_center_1 - 50 and row < y_center_1 + 50
275      then
276          color_port <= RED75;
277      else
278          color_port <= WHITE75;
279      end if;
280      when others =>
281          case p2_port(3) is
282              when '1' =>
283                  if (to_integer(column) - x_center_0)*
284 (to_integer(column) - x_center_0) + (to_integer(row) - y_center_1)*
285 (to_integer(row) - y_center_1) > 49*49 and (to_integer(column) - x_center_0)*
286 (to_integer(column) - x_center_0) + (to_integer(row) - y_center_1)*
287 (to_integer(row) - y_center_1) < 50*50 then
288                  color_port <= BLUE75;
289              else
290                  color_port <= WHITE75;
291              end if;
292      when others =>
293          color_port <= WHITE75;
294      end case;
295  end if;
296  elsif row > 160 and row < 320 and column > 213 and column < 426 then
297      if p1_win_port(4) = '1' then
298          color_port <= RED75;
299      elsif p2_win_port(4) = '1' then
300          color_port <= BLUE75;
301      else
302          if sf_port = "0100" then
303              if sf_color_port = '0' then
304                  color_port <= RED75;
305              else
306                  color_port <= BLUE75;
307              end if;
308          if row > 170 and row < 310 and column > 223 and column <
416 then
309              case p1_port(4) is
310                  when '1' =>
311                      if ((column + y_center_1 = row + x_center_1)
312 or (column + row = x_center_1 + y_center_1)) and column > x_center_1 - 50 and
313 column < x_center_1 + 50 and row > y_center_1 - 50 and row < y_center_1 + 50
314      then

```

```

      ...
      color_port <= RED75;
    else
      color_port <= WHITE75;
    end if;
  when others =>
    case p2_port(4) is
      when '1' =>
        if (to_integer(column) - x_center_1)*
          (to_integer(column) - x_center_1) + (to_integer(row) - y_center_1)*
          (to_integer(row) - y_center_1) > 49*49 and (to_integer(column) - x_center_1)*
          (to_integer(column) - x_center_1) + (to_integer(row) - y_center_1)*
          (to_integer(row) - y_center_1) < 50*50 then
          color_port <= BLUE75;
        else
          color_port <= WHITE75;
        end if;
      when others =>
        color_port <= WHITE75;
      end case;
    end case;
  end if;
else
  case p1_port(4) is
    when '1' =>
      if ((column + y_center_1 = row + x_center_1) or
        (column + row = x_center_1 + y_center_1)) and column > x_center_1 - 50 and
        column < x_center_1 + 50 and row > y_center_1 - 50 and row < y_center_1 + 50
      then
        color_port <= RED75;
      else
        color_port <= WHITE75;
      end if;
    when others =>
      case p2_port(4) is
        when '1' =>
          if (to_integer(column) - x_center_1)*
            (to_integer(column) - x_center_1) + (to_integer(row) - y_center_1)*
            (to_integer(row) - y_center_1) > 49*49 and (to_integer(column) - x_center_1)*
            (to_integer(column) - x_center_1) + (to_integer(row) - y_center_1)*
            (to_integer(row) - y_center_1) < 50*50 then
              color_port <= BLUE75;
            else
              color_port <= WHITE75;
            end if;
        when others =>
          color_port <= WHITE75;
        end case;
      end case;
    end if;
  end if;

```

```

340      end if;
341      elsif row > 160 and row < 320 and column > 426 and column < 640 then
342          if p1_win_port(5) = '1' then
343              color_port <= RED75;
344          elsif p2_win_port(5) = '1' then
345              color_port <= BLUE75;
346          else
347              if sf_port = "0101" then
348                  if sf_color_port = '0' then
349                      color_port <= RED75;
350                  else
351                      color_port <= BLUE75;
352                  end if;
353                  if row > 170 and row < 310 and column > 436 and column <
354                      630 then
355                          case p1_port(5) is
356                              when '1' =>
357                                  if ((column + y_center_1 = row + x_center_2)
358                                      or (column + row = x_center_2 + y_center_1)) and column > x_center_2 - 50 and
359                                      column < x_center_2 + 50 and row > y_center_1 - 50 and row < y_center_1 + 50
360                                      then
361                                          color_port <= RED75;
362                                      else
363                                          color_port <= WHITE75;
364                                      end if;
365                                      when others =>
366                                          case p2_port(5) is
367                                              when '1' =>
368                                                  if (to_integer(column) - x_center_2)*
369                                                      (to_integer(column) - x_center_2) + (to_integer(row) - y_center_1)*
370                                                      (to_integer(row) - y_center_1) > 49*49 and (to_integer(column) - x_center_2)*
371                                                      (to_integer(column) - x_center_2) + (to_integer(row) - y_center_1)*
372                                                      (to_integer(row) - y_center_1) < 50*50 then
373                                          color_port <= BLUE75;
374                                          else
375                                              color_port <= WHITE75;
376                                          end if;
377                                          when others =>
378                                              color_port <= WHITE75;
379                                          end case;
380                                      end case;
381                                      end if;
382                                  else
383                                      case p1_port(5) is
384                                          when '1' =>
385                                              if ((column + y_center_1 = row + x_center_2) or
386                                              (column + row = x_center_2 + y_center_1)) and column > x_center_2 - 50 and
387                                              column < x_center_2 + 50 and row > y_center_1 - 50 and row < y_center_1 + 50
388                                              then
389                                              color_port <= RED75;

```

```

384         else
385             color_port <= WHITE75;
386         end if;
387     when others =>
388         case p2_port(5) is
389             when '1' =>
390                 if (to_integer(column) - x_center_2)*
391                     (to_integer(column) - x_center_2) + (to_integer(row) - y_center_1)*
392                     (to_integer(row) - y_center_1) > 49*49 and (to_integer(column) - x_center_2)*
393                     (to_integer(column) - x_center_2) + (to_integer(row) - y_center_1)*
394                     (to_integer(row) - y_center_1) < 50*50 then
395                         color_port <= BLUE75;
396                     else
397                         color_port <= WHITE75;
398                     end if;
399                 when others =>
400                     color_port <= WHITE75;
401             end case;
402         end if;
403     end if;
404     elsif row > 320 and row < 480 and column > 0 and column < 213 then
405         if p1_win_port(6) = '1' then
406             color_port <= RED75;
407         elsif p2_win_port(6) = '1' then
408             color_port <= BLUE75;
409         else
410             if sf_port = "0110" then
411                 if sf_color_port = '0' then
412                     color_port <= RED75;
413                 else
414                     color_port <= BLUE75;
415                 end if;
416             if row > 330 and row < 470 and column > 10 and column <
417             203 then
418                 case p1_port(6) is
419                     when '1' =>
420                         if ((column + y_center_2 = row + x_center_0)
421                             or (column + row = x_center_0 + y_center_2)) and column > x_center_0 - 50 and
422                             column < x_center_0 + 50 and row > y_center_2 - 50 and row < y_center_2 + 50
423                             then
424                                 color_port <= RED75;
425                             else
426                                 color_port <= WHITE75;
427                             end if;
428                         when others =>
429                             case p2_port(6) is
430                                 when '1' =>
431                                     if (to_integer(column) - x_center_0)*
432                                         (to integer(column) - x center 0) + (to integer(row) - v center 2)*

```

pixel_generation.vhd

8/20/22, 1:38 PM

```

(to_integer(row) - y_center_2) > 49*49 and (to_integer(column) - x_center_0)*
(to_integer(column) - 0) + (to_integer(row) - y_center_2)*(to_integer(row) -
y_center_2) < 50*50 then
425                      color_port <= BLUE75;
426                      else
427                          color_port <= WHITE75;
428                      end if;
429                      when others =>
430                          color_port <= WHITE75;
431                      end case;
432                  end case;
433              end if;
434          else
435              case p1_port(6) is
436                  when '1' =>
437                      if ((column + y_center_2 = row + x_center_0) or
(column + row = x_center_0 + y_center_2)) and column > x_center_0 - 50 and
column < x_center_0 + 50 and row > y_center_2 - 50 and row < y_center_2 + 50
then
438                      color_port <= RED75;
439                      else
440                          color_port <= WHITE75;
441                      end if;
442                      when others =>
443                          case p2_port(6) is
444                              when '1' =>
445                                  if (to_integer(column) - x_center_0)*
(to_integer(column) - x_center_0) + (to_integer(row) - y_center_2)*
(to_integer(row) - y_center_2) > 49*49 and (to_integer(column) - x_center_0)*
(to_integer(column) - x_center_0) + (to_integer(row) - y_center_2)*
(to_integer(row) - y_center_2) < 50*50 then
446                                      color_port <= BLUE75;
447                                      else
448                                          color_port <= WHITE75;
449                                      end if;
450                                      when others =>
451                                          color_port <= WHITE75;
452                                      end case;
453                                  end case;
454                              end if;
455                          end if;
456                      elsif row > 320 and row < 480 and column > 213 and column < 426 then
457                          if p1_win_port(7) = '1' then
458                              color_port <= RED75;
459                          elsif p2_win_port(7) = '1' then
460                              color_port <= BLUE75;
461                          else
462                              if sf_port = "0111" then
463                                  if sf_color_port = '0' then
464                                      color_port <= RED75;

```

<http://localhost:4649/?mode=vhdl>

Page 12 of 15

```

465      color_port := RED75;
466
467      else
468          color_port <= BLUE75;
469      end if;
470      if row > 330 and row < 470 and column > 223 and column <
416 then
471          case p1_port(7) is
472              when '1' =>
473                  if ((column + y_center_2 = row + x_center_1)
474                      or (column + row = x_center_1 + y_center_2)) and column > x_center_1 - 50 and
475                      column < x_center_1 + 50 and row > y_center_2 - 50 and row < y_center_2 + 50
476                  then
477                      color_port <= RED75;
478                  else
479                      color_port <= WHITE75;
480                  end if;
481              when others =>
482                  case p2_port(7) is
483                      when '1' =>
484                          if (to_integer(column) - x_center_1)*
485                              (to_integer(column) - x_center_1) + (to_integer(row) - y_center_2)*
486                              (to_integer(row) - y_center_2) > 49*49 and (to_integer(column) - x_center_1)*
487                              (to_integer(column) - x_center_1) + (to_integer(row) - y_center_2)*
488                              (to_integer(row) - y_center_2) < 50*50 then
489                                  color_port <= BLUE75;
490                                  else
491                                      color_port <= WHITE75;
492                                  end if;
493                                  when others =>
494                                      color_port <= WHITE75;
495                                  end case;
496
497                  end case;
498              end if;
499          else
500              case p1_port(7) is
501                  when '1' =>
502                      if ((column + y_center_2 = row + x_center_1) or
503                          (column + row = x_center_1 + y_center_2)) and column > x_center_1 - 50 and
504                          column < x_center_1 + 50 and row > y_center_2 - 50 and row < y_center_2 + 50
505                      then
506                          color_port <= RED75;
507                          else
508                              color_port <= WHITE75;
509                          end if;
510                      when others =>
511                          case p2_port(7) is
512                              when '1' =>
513                                  if (to_integer(column) - x_center_1)*
514                                      (to_integer(column) - x_center_1) + (to_integer(row) - y_center_2)*
515                                      (to_integer(row) - y_center_2) > 49*49 and (to_integer(column) - x_center_1)*
516                                      (to_integer(column) - x_center_1) + (to_integer(row) - y_center_2)*
517                                      (to_integer(row) - y_center_2) < 50*50 then
518                                          color_port <= BLUE75;
519                                          else
520                                              color_port <= WHITE75;
521                                          end if;
522                                      when others =>
523                                          color_port <= WHITE75;
524                                      end case;
525
526                  end case;
527              end if;
528          end if;
529      end if;
530  end process;

```

pixel_generation.vhd

8/20/22, 1:38 PM

```

501      (to_integer(column) - x_center_1) + (to_integer(row) - y_center_2)*
502      (to_integer(row) - y_center_2) < 50*50 then
503          color_port <= BLUE75;
504      else
505          color_port <= WHITE75;
506      end if;
507      when others =>
508          color_port <= WHITE75;
509      end case;
510      end if;
511  elsif row > 320 and row < 480 and column > 426 and column < 640 then
512      if p1_win_port(8) = '1' then
513          color_port <= RED75;
514      elsif p2_win_port(8) = '1' then
515          color_port <= BLUE75;
516      else
517          if sf_port = "1000" then
518              if sf_color_port = '0' then
519                  color_port <= RED75;
520              else
521                  color_port <= BLUE75;
522              end if;
523          if row > 330 and row < 470 and column > 436 and column <
630 then
524              case p1_port(8) is
525                  when '1' =>
526                      if ((column + y_center_2 = row + x_center_2)
527                          or (column + row = x_center_2 + y_center_2)) and column > x_center_2 - 50 and
528                          column < x_center_2 + 50 and row > y_center_2 - 50 and row < y_center_2 + 50
529                          then
530                              color_port <= RED75;
531                          else
532                              color_port <= WHITE75;
533                          end if;
534                      when others =>
535                          case p2_port(8) is
536                              when '1' =>
537                                  if (to_integer(column) - x_center_2)*
538                                  (to_integer(column) - x_center_2) + (to_integer(row) - y_center_2)*
539                                  (to_integer(row) - y_center_2) > 49*49 and (to_integer(column) - x_center_2)*
540                                  (to_integer(column) - x_center_2) + (to_integer(row) - y_center_2)*
541                                  (to_integer(row) - y_center_2) < 50*50 then
542                                      color_port <= BLUE75;
543                                      else
544                                          color_port <= WHITE75;
545                                      end if;
546                      when others =>
547                          color_port <= WHITE75;

```

```

541           end case;
542       end case;
543   end if;
544 else
545     case p1_port(8) is
546       when '1' =>
547         if ((column + y_center_2 = row + x_center_2) or
548             (column + row = x_center_2 + y_center_2)) and column > x_center_2 - 50 and
549             column < x_center_2 + 50 and row > y_center_2 - 50 and row < y_center_2 + 50
550             then
551               color_port <= RED75;
552             else
553               color_port <= WHITE75;
554             end if;
555             when others =>
556               case p2_port(8) is
557                 when '1' =>
558                   if (to_integer(column) - x_center_2)*
559                     (to_integer(column) - x_center_2) + (to_integer(row) - y_center_2)*
560                     (to_integer(row) - y_center_2) > 49*49 and (to_integer(column) - x_center_2)*
561                     (to_integer(column) - x_center_2) + (to_integer(row) - y_center_2)*
562                     (to_integer(row) - y_center_2) < 50*50 then
563                       color_port <= BLUE75;
564                     else
565                       color_port <= WHITE75;
566                     end if;
567                     when others =>
568                       color_port <= WHITE75;
569                     end case;
570                   end if;
571                 end if;
572               end process pixel_rendering;
573             end behavior;
574

```

Appendix E: system_clock_generator.vhd

```

system_clock_generator.vhd                                         8/20/22, 1:39 PM

1 ---
2 =====
3 --ENGS 31/ CoSc 56
4 --Final Project: Tic-tac-toe in VGA
5 --System Clock Generator
6 --Source: Lab 6
7 --Instructors: Ben Dobbins, Tad Truex
8 --Student name: Di Luo
9 ---
10 ---
11 --Library Declarations:
12 --
13 library IEEE;
14 use IEEE.STD_LOGIC_1164.ALL;
15 use IEEE.NUMERIC_STD.ALL;
16 use ieee.math_real.all;
17 library UNISIM;
18 use UNISIM.VComponents.all;
19 --
20 --
21 --Entity Declaration:
22 --
23 entity system_clock_generator is
24     generic (CLOCK_DIVIDER_RATIO : integer);
25     port (
26         input_clk_port    : in std_logic;
27         system_clk_port   : out std_logic;
28         fwd_clk_port     : out std_logic);
29 end system_clock_generator;
30 --
31 --
32 --Architecture Type:
33 --
34 architecture behavioral_architecture of system_clock_generator is
35 --
36 --Signal Declarations:
37 --
38 --CONSTANT FOR SYNTHESIS:
39 constant CLOCK_DIVIDER_TC: integer := CLOCK_DIVIDER_RATIO/2;
40     constant END_CTMU11_ATTOMI;

```

system_clock_generator.vhd

8/20/22, 1:39 PM

```

--CONSTANT FOR SIMULATION.
41 constant CLOCK_DIVIDER_TC: integer := 5;
42
43 --Automatic register sizing:
44 constant COUNT_LEN      : integer := integer(ceil(log2(
real(CLOCK_DIVIDER_TC) ) ));
45 signal system_clk_divider_counter : unsigned(COUNT_LEN-1 downto 0) := (others
=> '0');
46 signal system_clk_tog       : std_logic := '0';
47 signal system_clk           : std_logic := '0';

48 begin
49
50
51 --
52 =====
53 --Processes:
54 --
55 =====
56 --
57 --Clock (frequency) Divider:
58 --
59 Clock_divider: process(input_clk_port)
60 begin
61   if rising_edge(input_clk_port) then
62     if system_clk_divider_counter = CLOCK_DIVIDER_TC-1 then      -- Counts to
1/2 clk period
63       system_clk_tog <= NOT(system_clk_tog);                      -- T flip flop
64       system_clk_divider_counter <= (others => '0');            -- Reset
65     else
66       system_clk_divider_counter <= system_clk_divider_counter + 1; -- Count
67     up
68     end if;
69   end if;
70 end process Clock_divider;
71 --
72 =====
73 -- Clock buffer for the system clock
74 --
75 =====
76 -- The BUFG component puts the system clock onto the FPGA clocking network
77 system_clock_buffer: BUFG
78   port map (
    I => system_clk_tog,
    O => system_clk);
79   system_clk_port <= system_clk;
80
81

```

<http://localhost:4649/?mode=vhdl>

Page 2 of 3

system_clock_generator.vhd

8/20/22, 1:39 PM

```
79 --
80 -- Clock Forwarding
81 --
82 ++++++
83 clock_forwarding_ODDR : ODDR
84 generic map(
85   DDR_CLK_EDGE => "SAME_EDGE", -- "OPPOSITE_EDGE" or "SAME_EDGE"
86   INIT => '0', -- Initial value for Q port ('1' or '0')
87   SRTYPE => "SYNC") -- Reset Type ("ASYNC" or "SYNC")
88 port map (
89   Q => fwd_clk_port, -- 1-bit DDR output
90   C => system_clk, -- 1-bit clock input
91   CE => '1', -- 1-bit clock enable input
92   D1 => '1', -- 1-bit data input (positive edge)
93   D2 => '0', -- 1-bit data input (negative edge)
94   R => '0', -- 1-bit reset input
95   S => '0' -- 1-bit set input
96 );
97 end behavioral_architecture;
```

Appendix F: input_conditioning.vhd

input_conditioning.vhd

8/20/22, 1:45 PM

```

1 ---
2 =====
3 --ENGS 31/ CoSc 56
4 --Final Project: Tic-tac-toe in VGA
5 --Input Conditioning
6 --Source: Lab 5
7 --Instructors: Ben Dobbins, Tad Truex
8 --Student name: Di Luo
9 --
10 --
11 --Library Declarations:
12 --
13 library IEEE;
14 use IEEE.STD_LOGIC_1164.ALL;
15 use IEEE.NUMERIC_STD.ALL;
16 use ieee.math_real.all;
17 library UNISIM;
18 use UNISIM.VComponents.all;
19 --
20 --
21 --Entity Declaration:
22 --
23 entity button_interface is
24     Port( clk_port      : in  std_logic;
25           button_port    : in  std_logic;
26           button_db_port : out std_logic;
27           button_mp_port : out std_logic);
28 end button_interface;
29 --
30 --
31 --Architecture
32 --
33 architecture Behavioral of button_interface is
34 --
35 --Signal Declarations
36 --
37 --
38 ++++++
39 --Synchronizer
40 
```

<http://localhost:4649/?mode=vhdl>

Page 1 of 5

```

39 --
40 signal double_flop_synchronizer : std_logic_vector(1 downto 0) := "00";
41 signal synchronized_button_press : std_logic := '0';
42
43 --
44 --Debouncer
45 --
46 type state_type is (BUTTON_OUTPUT_LOW, LOW_TO_HIGH_TRANSITION,
47                      BUTTON_OUTPUT_HIGH, HIGH_TO_LOW_TRANSITION);
48 signal current_state      : state_type := BUTTON_OUTPUT_LOW;
49 signal next_state         : state_type;
50 signal timeout_counter   : unsigned(8 downto 0) := (others => '0');
51 signal reset              : std_logic := '0';
52 signal timeout            : std_logic := '0';
53 constant MAXCOUNT        : integer := 500;
54 signal debounced_input    : std_logic := '0';
55
56 --
57 --Monopulser
58 --
59 signal mp_delay_register : std_logic := '0';
60
61 --
62 =====
63 --Processes
64 --
65 =====
66 begin
67 --
68 --Double-Flop Synchronizer:
69 --
70 --Prevents metastability on the input by passing the raw button press through
71 --a
72 --double flop synchronizer.
73 synchronize: process(clk_port, double_flop_synchronizer)
74 begin
75     if rising_edge(clk_port) then
76         double_flop_synchronizer(1) <= button_port;
77         double_flop_synchronizer(0) <= double_flop_synchronizer(1);
78     end if;
79     synchronized_button_press <= double_flop_synchronizer(0);
80 end process;
81
82

```

input_conditioning.vhd

8/20/22, 1:45 PM

```

79 --
80 --Debouncer:
81 --
82 --State Update Logic
83 state_update: process(clk_port)
84 begin
85     if rising_edge(clk_port) then
86         current_state <= next_state;
87     end if;
88 end process;
89
90 --Next State Logic
91 next_state_logic: process(current_state, synchronized_button_press, timeout)
92 begin
93     next_state <= current_state;
94     case current_state is
95         when BUTTON_OUTPUT_LOW =>
96             if synchronized_button_press = '1' then
97                 next_state <= LOW_TO_HIGH_TRANSITION;
98             end if;
99
100        when LOW_TO_HIGH_TRANSITION =>
101            if synchronized_button_press = '0' then
102                next_state <= BUTTON_OUTPUT_LOW;
103            elsif timeout = '1' then
104                next_state <= BUTTON_OUTPUT_HIGH;
105            end if;
106
107        when BUTTON_OUTPUT_HIGH =>
108            if synchronized_button_press = '0' then
109                next_state <= HIGH_TO_LOW_TRANSITION;
110            end if;
111
112        when HIGH_TO_LOW_TRANSITION =>
113            if synchronized_button_press = '1' then
114                next_state <= BUTTON_OUTPUT_HIGH;
115            elsif timeout = '1' then
116                next_state <= BUTTON_OUTPUT_LOW;
117            end if;
118     end case;
119 end process;
120
121 --Output Logic
122 output_logic: process(current_state)
123 begin
124     case current_state is
125         when BUTTON_OUTPUT_LOW =>
126             reset      <= '1';

```

<http://localhost:4649/?mode=vhdl>

Page 3 of 5

input_conditioning.vhd

8/20/22, 1:45 PM

```

127      debounced_input    <= '0';
128
129      when LOW_TO_HIGH_TRANSITION =>
130          reset            <= '0';
131          debounced_input  <= '0';
132
133      when BUTTON_OUTPUT_HIGH =>
134          reset            <= '1';
135          debounced_input  <= '1';
136
137      when HIGH_TO_LOW_TRANSITION =>
138          reset            <= '0';
139          debounced_input  <= '1';
140
141  end case;
142 end process;
143 button_db_port <= debounced_input;
144
--Timer Subroutine
145 timer: process(clk_port)
146 begin
147     if rising_edge(clk_port) then
148         if reset = '1' then
149             timeout_counter <= (others => '0');
150         else
151             timeout_counter <= timeout_counter + 1;
152         end if;
153     end if;
154 end process;
155
--Timeout Flag
156 timeout_flag: process(timeout_counter)
157 begin
158     if timeout_counter = MAXCOUNT - 1 then
159         timeout <= '1';
160     else
161         timeout <= '0';
162     end if;
163 end process;
164
165 --
166 ++++++
167 --Monopulser:
168 --
169 ++++++
170 --A monopulsed output is an output that is high for one clock cycle.
171 monopulser: process(clk_port, debounced_input, mp_delay_register)
172 begin
173     if rising_edge(clk_port) then
174         mp_delay_register <= debounced_input;
175     end if;

```

<http://localhost:4649/?mode=vhdl>

Page 4 of 5

ENGS 31 - Final Project

input_conditioning.vhd

8/20/22, 1:45 PM

```
175 |     button_mp_port <= debounced_input and not(mp_delay_register);  
176 | end process monopulser;  
177 | end Behavioral;  
178 |
```

Appendix G: TicTacToe_shell.vhd

```

TicTacToe_shell.vhd                                         8/20/22, 1:39 PM

1 ---
2 =====
3 --ENGS 31/ CoSc 56
4 --Final Project: Tic-tac-toe in VGA
5 --Tic-tac-toe Top Level
6 --Instructors: Ben Dobbins, Tad Truex
7 --Student name: Di Luo
8 --
9 --
10 --Library Declarations:
11 --
12 library IEEE;
13 use IEEE.std_logic_1164.all;
14 use IEEE.numeric_std.all;      -- needed for arithmetic
15 use ieee.math_real.all;        -- needed for automatic register sizing
16 library UNISIM;              -- needed for the BUFG component
17 use UNISIM.Vcomponents.ALL;
18 --
19 --
20 --Entity Declaration:
21 --
22 entity TicTacToe_shell is
23     port (
24         clk_ext_port      : in std_logic;                      -- ext 100 MHz
25         clock            : in std_logic;                      -- switch[1] that
26         starts           : in std_logic;                      -- starts the game
27         reset_ext_port   : in std_logic;                      -- switch[0] that
28         resets           : in std_logic;                      -- resets the game
29         drop_ext_port    : in std_logic;                      -- central button
30         that drops the chess piece
31         btnU_ext_port   : in std_logic;                      -- up button that
32         moves selecting frame up
33         btnD_ext_port   : in std_logic;                      -- down button
34         that moves selecting frame down
35         btnL_ext_port   : in std_logic;                      -- left button
36         that moves selecting frame left
37         btnR_ext_port   : in std_logic;                      -- right button
38         that moves selecting frame right
39         color_ext_port  : out std_logic_vector(11 downto 0);-- 12 bits for
40         color coding, 4 for each of RGB
41         Hsync_ext_port  : out std_logic;                      -- horizontal
42         sync signal to VGA monitor
43         Vsync_ext_port  : out std_logic;                      -- vertical sync
44

```

TicTacToe_shell.vhd

8/20/22, 1:39 PM

```

34      vsync_ext_port : out std_logic;           -- vertical sync
35  signal to VGA monitor
36 );
37 end TicTacToe_shell;
38 -----
39 --Architecture + Component Declarations
40 -----
41 architecture behavior of TicTacToe_shell is
42 --System Clock Generation:
43 component system_clock_generator is
44   generic (
45     CLOCK_DIVIDER_RATIO : integer);
46   port (
47     input_clk_port : in std_logic;    -- 100 MHz clk of FPGA
48     system_clk_port : out std_logic;  -- 25 MHz clk
49     fwd_clk_port   : out std_logic); -- unused, open
50 end component;
51
52 --Input Conditioning:
53 component button_interface is
54   Port( clk_port       : in std_logic;  -- system clk
55         button_port    : in std_logic;  -- external buttons
56         button_db_port : out std_logic; -- debounced button output
57         button_mp_port : out std_logic); -- monopulse button output
58 end component;
59
60 --VGA driver:
61 component VGA_driver is
62   port ( vclk_port      : in STD_LOGIC; -- 25 MHz clock
63          Vsync_port     : out STD_LOGIC; -- vertical sync signal to VGA
64          monitor        : VGA monitor;  -- horizontal sync signal to
65          Hsync_port     : out STD_LOGIC; -- VGA monitor
66          video_on_port  : out STD_LOGIC; -- accomplish blanking. 1 when
67          inside display area. 0 when outside display area
68          pixel_x_port   : out STD_LOGIC_VECTOR(9 downto 0); -- current
69          pixel_y_port   : out STD_LOGIC_VECTOR(9 downto 0)); -- current
70          pixel's x coordinate
71          pixel's y coordinate
72 end component;
73
74 -- Game logic circuit
75 component game_logic is
76   port ( clk_port      : in std_logic;  -- 25 MHz clock
77         start_port    : in std_logic; -- start a new game
78         reset_port    : in std_logic; -- clear the chessboard
79         drop_port     : in std_logic; -- put a piece on the chessboard
80         up_port       : in std_logic; -- move selecting frame up
81         down_port    : in std_logic; -- move selecting frame down
82         );
83 end component;

```

<http://localhost:4649/?mode=vhdl>

Page 2 of 6

```

78      left_port    :  in std_logic;   -- move selecting frame left
79      right_port   :  in std_logic;   -- move selecting frame right
80      p1_port      :  out std_logic_vector(8 downto 0);   -- indicates
the blocks that player 1 has dropped a chess piece
81      p2_port      :  out std_logic_vector(8 downto 0);   -- indicates
the blocks that player 2 has dropped a chess piece
82      sf_port      :  out std_logic_vector(3 downto 0);   -- indicates
the location of the selecting frame
83      sf_color_port:  out std_logic;                      -- indicates
which player is dropping next
84      p1_win_port   :  out std_logic_vector(8 downto 0);   -- indicates
the blocks that make player 1 win
85      p2_win_port   :  out std_logic_vector(8 downto 0));  -- indicates
the blocks that make player 2 win
86 end component;
87
88 --Pixel generation circuit:
89 component pixel_generation is
90     port ( pixel_x_port   :  in std_logic_vector(9 downto 0);   --
current pixel's x coordinate
91            pixel_y_port   :  in std_logic_vector(9 downto 0);   --
current pixel's y coordinate
92            video_on_port  :  in std_logic;   -- accomplish blanking. 1
when inside display area. 0 when outside display area
93            p1_port        :  in std_logic_vector(8 downto 0);
94            p2_port        :  in std_logic_vector(8 downto 0);
95            sf_port        :  in std_logic_vector(3 downto 0);
96            sf_color_port :  in std_logic;                      --
indicates which player is dropping next
97            p1_win_port   :  in std_logic_vector(8 downto 0);   --
indicates the blocks that make player 1 win
98            p2_win_port   :  in std_logic_vector(8 downto 0);   --
indicates the blocks that make player 2 win
99            color_port    :  out std_logic_vector(11 downto 0)); -- 12
bits for color coding, 4 for each of RGB
100 end component;
101
102 =====
103 --Local Signal Declaration
104 =====
105 -- clocking
106 signal system_clk : std_logic;
107 -- VGA to pixel_gene
108 signal x: std_logic_vector(9 downto 0);
109 signal y: std_logic_vector(9 downto 0);
110 signal video_on: std_logic;
111 -- input conditioning to game logic
112 signal drop: std_logic;
113 signal up: std_logic;
114 signal down: std_logic;

```

TicTacToe_shell.vhd

8/20/22, 1:39 PM

```

115 signal left: std_logic;
116 signal right: std_logic;
117 -- game logic to pixel generation
118 signal p1: std_logic_vector(8 downto 0);
119 signal p2: std_logic_vector(8 downto 0);
120 signal sf: std_logic_vector(3 downto 0);
121 signal sf_color: std_logic;
122 signal p1_win: std_logic_vector(8 downto 0);
123 signal p2_win: std_logic_vector(8 downto 0);
124
125 begin
126
127 =====
128 --Port Mapping + Processes:
129 =====
130 --System Clock Generation:
131 --Wire the system clock generator into the shell with a port map
132 clocking: system_clock_generator
133 generic map(
134   CLOCK_DIVIDER_RATIO => 4)
135 port map(
136   input_clk_port      => clk_ext_port,
137   system_clk_port    => system_clk,
138   fwd_clk_port       => open);
139
140 --Wire the input conditioning block into the shell with a port map:
141 drop_monopulse: button_interface
142 port map(
143   clk_port           => system_clk,
144   button_port        => drop_ext_port,
145   button_db_port    => open,
146   button_mp_port    => drop);
147
148 up_monopulse: button_interface
149 port map(
150   clk_port           => system_clk,
151   button_port        => btnU_ext_port,
152   button_db_port    => open,
153   button_mp_port    => up);
154
155 down_monopulse: button_interface
156 port map(
157   clk_port           => system_clk,
158   button_port        => btnD_ext_port,
159   button_db_port    => open,
160   button_mp_port    => down);
161
162 left_monopulse: button_interface
163 port map(
164   clk_port           => system_clk.

```

<http://localhost:4649/?mode=vhdl>

Page 4 of 6

TicTacToe_shell.vhd

8/20/22, 1:39 PM

```

165      button_port      => btnL_ext_port,
166      button_db_port   => open,
167      button_mp_port   => left);
168
169 right_monopulse: button_interface
170 port map(
171     clk_port          => system_clk,
172     button_port       => btnR_ext_port,
173     button_db_port   => open,
174     button_mp_port   => right);
175
176 game_logic_control: game_logic
177 port map(
178     clk_port          => system_clk,
179     start_port        => start_ext_port,
180     reset_port        => reset_ext_port,
181     drop_port         => drop,
182     up_port           => up,
183     down_port         => down,
184     left_port         => left,
185     right_port        => right,
186     p1_port           => p1,
187     p2_port           => p2,
188     sf_port           => sf,
189     sf_color_port    => sf_color,
190     p1_win_port       => p1_win,
191     p2_win_port       => p2_win);
192
193 vga_control: VGA_driver
194 port map(
195     vclk_port         => system_clk,
196     Vsync_port        => Vsync_ext_port,
197     Hsync_port        => Hsync_ext_port,
198     video_on_port    => video_on,
199     pixel_x_port     => x,
200     pixel_y_port     => y);
201
202 pixel_generation_datapath: pixel_generation
203 port map(
204     pixel_x_port     => x,
205     pixel_y_port     => y,
206     video_on_port   => video_on,
207     p1_port          => p1,
208     p2_port          => p2,
209     sf_port          => sf,
210     sf_color_port   => sf_color,
211     p1_win_port     => p1_win,
212     p2_win_port     => p2_win,
213     color_port       => color_ext_port);

```

<http://localhost:4649/?mode=vhdl>

Page 5 of 6

ENGS 31 - Final Project

TicTacToe_shell.vhd

8/20/22, 1:39 PM

```
214  
215 | end behavior;  
216 |
```

Appendix H: game_logic_tb.vhd

```

game_logic_tb.vhd                                         8/23/22, 4:10 PM

1 ---
2 =====
3 --ENGS 31/ CoSc 56
4 --Final Project: Tic-tac-toe in VGA
5 --Game Logic Circuit Testbench
6 --Instructors: Ben Dobbins, Tad Truex
7 --Student name: Di Luo
8 ---
9 ---
10 --Library Declarations:
11 --
12 library IEEE;
13 use IEEE.STD_LOGIC_1164.ALL;
14 use IEEE.NUMERIC_STD.ALL;
15 --
16 --Testbench Entity Declaration
17 --
18 entity game_logic_tb is
19 end game_logic_tb;
20 --
21 --Testbench declarations
22 --
23 architecture testbench of game_logic_tb is
24 --
25 --
26 --Component Declaration:
27 --
28 component game_logic is
29   Port (
30     -- system clock
31     clk_port      : in std_logic;    -- 25 MHz clock
32     -- inputs from external ports/input conditioning circuits
33     start_port    : in std_logic;    -- start a new game
34     reset_port    : in std_logic;    -- clear the chessboard
35     drop_port     : in std_logic;    -- put a piece on the selected position
36     up_port       : in std_logic;    -- move selecting frame up
37     down_port     : in std_logic;    -- move selecting frame down
38     left_port     : in std_logic;    -- move selecting frame left
39     right_port    : in std_logic;    -- move selecting frame right
40     -- output to pixel generation circuit
41     p1_port       : out std_logic_vector(8 downto 0);    -- indicates the
42     -- blocks that player 1 has dropped a chess piece
43   );

```

game_logic_tb.vhd

8/23/22, 4:10 PM

```

44    blocks that player 1 has dropped a chess piece
45        p2_port      :  out std_logic_vector(8 downto 0);  -- indicates the
blocks that player 2 has dropped a chess piece
46        sf_port      :  out std_logic_vector(3 downto 0);  -- indicates the
location of the selecting frame
47        sf_color_port:  out std_logic;                      -- indicates
which player is dropping next
48        p1_win_port   :  out std_logic_vector(8 downto 0);  -- indicates the
blocks that make player 1 win
49        p2_win_port   :  out std_logic_vector(8 downto 0)); -- indicates the
blocks that make player 2 win
50    end component;
51
52    =====
53    --Local Signal Declaration
54    =====
55    --Simulated input signals
56    signal clk: std_logic := '0';
57    signal start: std_logic := '0';
58    signal reset: std_logic := '0';
59    signal drop: std_logic := '0';
60    signal up: std_logic := '0';
61    signal down: std_logic := '0';
62    signal left: std_logic := '0';
63    signal right: std_logic := '0';
64    --Simulated output signals
65    signal p1: std_logic_vector(8 downto 0) := (others => '0');
66    signal p2: std_logic_vector(8 downto 0) := (others => '0');
67    signal sf: std_logic_vector(3 downto 0) := "0000";
68    signal sf_color: std_logic := '0';
69    signal p1_win: std_logic_vector(8 downto 0) := (others => '0');
70    signal p2_win: std_logic_vector(8 downto 0) := (others => '0');
71    -- Clock period definitions
72    constant clk_period: time := 40ns;  -- 25 MHz clock
73
74    begin
75
76    =====
77    --Port Mapping + Processes:
78    =====
79    -- wire the game logic circuit with local signals
80    dut: game_logic port map (
81        clk_port      => clk,
82        start_port   => start,
83        reset_port   => reset,
84        drop_port    => drop,
85        up_port       => up,
86        down_port    => down,
87        left_port    => left,
88        right_port   => right,

```

<http://localhost:4649/?mode=vhdl>

Page 2 of 10

game_logic_tb.vhd

8/23/22, 4:10 PM

```

88      p1_port      => p1,
89      p2_port      => p2,
90      sf_port      => sf,
91      sf_color_port => sf_color,
92      p1_win_port  => p1_win,
93      p2_win_port  => p2_win);
94
95  --Timing:
96  clk_process: process
97  begin
98    clk <= '0';
99    wait for clk_period/2;
100   clk <= '1';
101   wait for clk_period/2;
102 end process;
103
104 --Stimulus Process:
105 stim_proc: process
106 begin
107   wait for 2*clk_period;  -- hold in sStart state for 2 clk cycles
108
109   --
110   --Test a normal game:
111   --
112   -- press start button
113   start <= '1';
114   wait for clk_period;
115   start <= '0';
116   wait for clk_period;
117
118   -- Step 1 (player 1)
119   -- move left from Central Mid to Central Left
120   left <= '1';
121   wait for clk_period;
122   left <= '0';
123   wait for 4*clk_period;
124
125   -- move up to Top Left
126   up <= '1';
127   wait for clk_period;
128   up <= '0';
129   wait for 4*clk_period;
130
131   -- drop at Top Left
132   drop <= '1';
133   wait for clk_period;
134   drop <= '0';
135   wait for 4*clk_period;

```

<http://localhost:4649/?mode=vhdl>

Page 3 of 10

game_logic_tb.vhd

8/23/22, 4:10 PM

```

136  -- X_
137  --
138  --
139
140  -- Step 2 (player 2)
141  -- move right from Central Mid to Central Right
142  right <= '1';
143  wait for clk_period;
144  right <= '0';
145  wait for 4*clk_period;
146
147  -- move up to Top Right
148  up <= '1';
149  wait for clk_period;
150  up <= '0';
151  wait for 4*clk_period;
152
153  -- drop at Top Right
154  drop <= '1';
155  wait for clk_period;
156  drop <= '0';
157  wait for 4*clk_period;
158  -- X_0
159  --
160  --
161
162  -- Step 3 (player 1)
163  -- move left from Central Mid to Central Left
164  left <= '1';
165  wait for clk_period;
166  left <= '0';
167  wait for 4*clk_period;
168
169  -- drop at Central Left
170  drop <= '1';
171  wait for clk_period;
172  drop <= '0';
173  wait for 4*clk_period;
174  -- X_0
175  -- X_
176  --
177
178  -- Step 4 (player 2)
179  -- move right from Central Mid to Central Right
180  right <= '1';
181  wait for clk_period;
182  right <= '0';
183  wait for 4*clk_period;
184
185  -- drop at Central Right

```

<http://localhost:4649/?mode=vhdl>

Page 4 of 10

game_logic_tb.vhd

8/23/22, 4:10 PM

```

186      drop <= '1';
187      wait for clk_period;
188      drop <= '0';
189      wait for 4*clk_period;
190      -- X_0
191      -- X_0
192      --
193
194      -- Step 5 (player 1)
195      -- move left from Central Mid to Central Left
196      left <= '1';
197      wait for clk_period;
198      left <= '0';
199      wait for 4*clk_period;
200
201      -- move down to Bottom Left
202      down <= '1';
203      wait for clk_period;
204      down <= '0';
205      wait for 4*clk_period;
206
207      -- drop at Bottom Left, 1st column all X, player 1 wins
208      drop <= '1';
209      wait for clk_period;
210      drop <= '0';
211      wait for 8*clk_period;
212      -- X_0
213      -- X_0
214      -- X_
215
216      reset <= '1';
217      wait for clk_period;
218      reset <= '0';
219      wait for 8*clk_period;
220
221      --
222      ++++++
223      --Test a tie game:
224      --
225      ++++++
226      -- press start button
227      start <= '1';
228      wait for clk_period;
229      start <= '0';
230      wait for clk_period;
231
232      -- Step 1 (player 1)
233      -- move left from Central Mid to Central Left
234      left <= '1';
235      wait for clk_period;

```

<http://localhost:4649/?mode=vhdl>

Page 5 of 10

game_logic_tb.vhd

8/23/22, 4:10 PM

```

233      wait for clk_period;
234      left <= '0';
235      wait for 4*clk_period;
236
237      -- move up to Top Left
238      up <= '1';
239      wait for clk_period;
240      up <= '0';
241      wait for 4*clk_period;
242
243      -- drop at Top Left
244      drop <= '1';
245      wait for clk_period;
246      drop <= '0';
247      wait for 4*clk_period;
248      -- X_
249      --
250      --
251
252      -- Step 2 (player 2)
253      -- move up to Top Mid
254      up <= '1';
255      wait for clk_period;
256      up <= '0';
257      wait for 4*clk_period;
258
259      -- drop at Top Mid
260      drop <= '1';
261      wait for clk_period;
262      drop <= '0';
263      wait for 4*clk_period;
264      -- X0_
265      --
266      --
267
268      -- Step 3 (player 1)
269      -- move left from Central Mid to Central Left
270      left <= '1';
271      wait for clk_period;
272      left <= '0';
273      wait for 4*clk_period;
274
275      -- drop at Central Left
276      drop <= '1';
277      wait for clk_period;
278      drop <= '0';
279      wait for 4*clk_period;
280      -- X0_
281      -- X_
282      --

```

<http://localhost:4649/?mode=vhdl>

Page 6 of 10

game_logic_tb.vhd

8/23/22, 4:10 PM

```

283
284      -- Step 4 (player 2)
285      -- drop at Central Mid
286      drop <= '1';
287      wait for clk_period;
288      drop <= '0';
289      wait for 4*clk_period;
290      -- X0_
291      -- X0_
292      -- __
293
294      -- Step 5 (player 1)
295      -- move down to Bottom Mid
296      down <= '1';
297      wait for clk_period;
298      down <= '0';
299      wait for 4*clk_period;
300
301      -- drop at Bottom Mid
302      drop <= '1';
303      wait for clk_period;
304      drop <= '0';
305      wait for 4*clk_period;
306      -- X0_
307      -- X0_
308      -- _X_
309
310      -- step 6 (player 2)
311      -- move down to Bottom Mid
312      down <= '1';
313      wait for clk_period;
314      down <= '0';
315      wait for 4*clk_period;
316
317      -- move left to Bottom Left
318      left <= '1';
319      wait for clk_period;
320      left <= '0';
321      wait for 4*clk_period;
322
323      -- drop at Bottom Left
324      drop <= '1';
325      wait for clk_period;
326      drop <= '0';
327      wait for 4*clk_period;
328      -- X0_
329      -- X0_
330      -- OX_
331
332      -- step 7 (player 1)

```

<http://localhost:4649/?mode=vhdl>

Page 7 of 10

```

333      -- move up to Top Mid
334      up <= '1';
335      wait for clk_period;
336      up <= '0';
337      wait for 4*clk_period;
338
339      -- move right to Top Right
340      right <= '1';
341      wait for clk_period;
342      right <= '0';
343      wait for 4*clk_period;
344
345      -- drop at Top Right
346      drop <= '1';
347      wait for clk_period;
348      drop <= '0';
349      wait for 4*clk_period;
350      -- XOX
351      -- XO_
352      -- OX_
353
354      -- step 8 (player 2)
355      -- move down to Bottom Mid
356      down <= '1';
357      wait for clk_period;
358      down <= '0';
359      wait for 4*clk_period;
360
361      -- move right to Bottom Right
362      right <= '1';
363      wait for clk_period;
364      right <= '0';
365      wait for 4*clk_period;
366
367      -- drop at Bottom Right
368      drop <= '1';
369      wait for clk_period;
370      drop <= '0';
371      wait for 4*clk_period;
372      -- XOX
373      -- XO_
374      -- OXO
375
376      -- step 9 (player 1)
377      -- move right to Central Right
378      right <= '1';
379      wait for clk_period;
380      right <= '0';
381      wait for 4*clk_period;
382

```

game_logic_tb.vhd

8/23/22, 4:10 PM

```

383    -- drop at Central Right
384    drop <= '1';
385    wait for clk_period;
386    drop <= '0';
387    wait for 8*clk_period;
388    -- XOX
389    -- XOX
390    -- OXO
391
392    -- Nobody wins. Tie!
393
394    reset <= '1';
395    wait for clk_period;
396    reset <= '0';
397    wait for 8*clk_period;
398
399    --
400    ++++++
401    --Test dropping at pre-occupied space:
402    --
403    ++++++
404    -- press start button
405    start <= '1';
406    wait for clk_period;
407    start <= '0';
408    wait for clk_period;
409
410    -- Step 1 (player 1)
411    -- drop at Central Mid
412    drop <= '1';
413    wait for clk_period;
414    drop <= '0';
415    wait for 4*clk_period;
416
417    --
418    -- _X_
419    -- __
420
421    -- Step 2 (player 2)
422    -- attempt to drop at Central Mid
423    drop <= '1';
424    wait for clk_period;
425    drop <= '0';
426    wait for 4*clk_period;
427
428    --
429    -- _X_
430    -- __
431
432    -- Step 2 (player 2)
433    -- attempt again to drop at Central Mid
434    drop <= '1';

```

<http://localhost:4649/?mode=vhdl>

Page 9 of 10

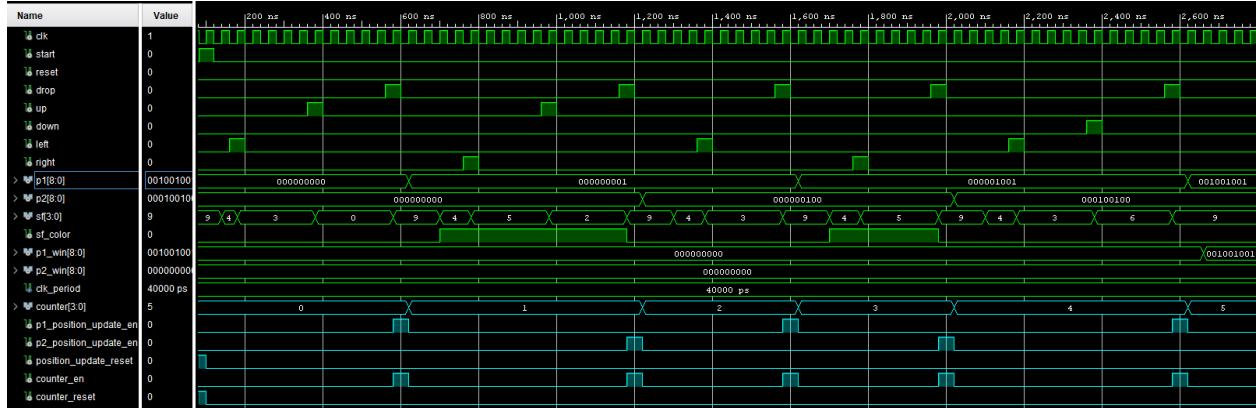
game_logic_tb.vhd

8/23/22, 4:10 PM

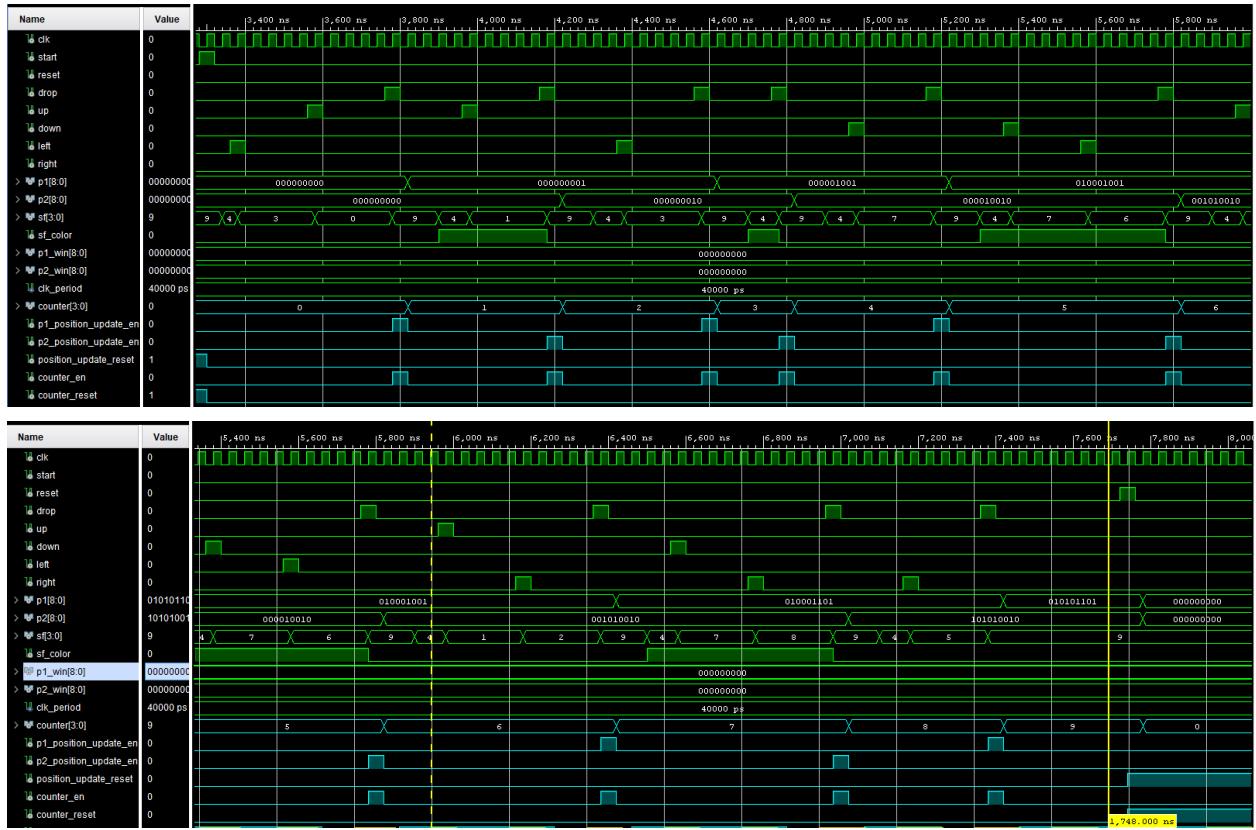
```
430     drop <= '1';
431     wait for clk_period;
432     drop <= '0';
433     wait for 4*clk_period;
434     --
435     -- _X_
436     --
437
438     reset <= '1';
439     wait for clk_period;
440     reset <= '0';
441     wait;
442 end process stim_proc;
443 end testbench;
444
```

Appendix I: Test results of game_logic_tb.vhd (Please refer to the comments in Appendix H that graphically presents the chessboard)

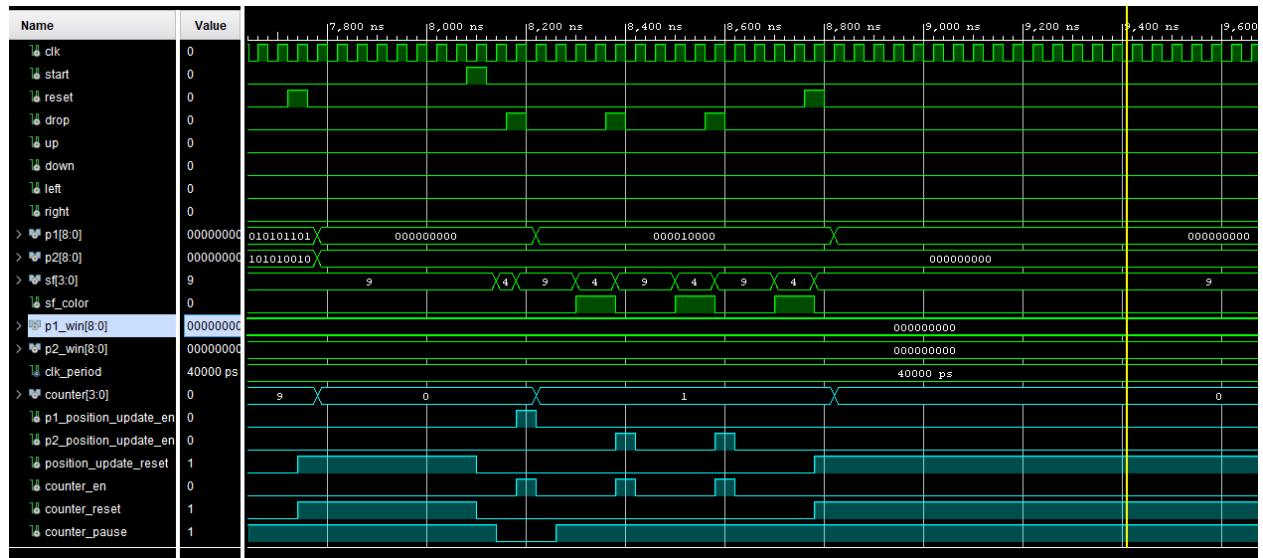
Test a normal game:



Test a tie game:



Test dropping at a pre-occupied space: (counter_pause stays high so that the pre-occupied space cannot be dropped again, and the counter doesn't increment.)



Appendix J: VGA driver testing with patterns in vga_test_pattern_12.vhd and the one based on Tad's suggestions

