```
1   //VGA control module
2
3   module VGAcontrol(clk,
4                     resetn,
5                     start,
6                     score1,
7                     score2,
8                     score3,
9                     x_out,
10                    y_out,
11                    colour_out,
12                    draw,
13                    currentmain
14                    );
15
16      input resetn;
17      input clk;
18      input start;
19      input [3:0] score1, score2, score3;
20      input [3:0] currentmain;
21      output [7:0] x_out;
22      output [6:0] y_out;
23      output [23:0] colour_out;
24      output draw;
25
26      //declares wires for drawcontrol
27      wire countdone;
28      wire draw_done;
29      wire countstart;
30      wire [3:0] current;
31      wire pre;
32
33      //declares wires for drawdatapath
34      wire [23:0] imagedata;
35      wire [17:0] address_start;
36      wire [11:0] num1_start, num2_start, num3_start;
37      wire [23:0] numberdata1, numberdata2, numberdata3;
38      wire [7:0] x_counter;
39      wire [6:0] y_counter;
40      wire [7:0] x_out;
41      wire [6:0] y_out;
42      wire [17:0] address;
43
44
45      wire [11:0] num1address, num2address, num3address;
46      wire [8:0] pixel;
47
```

```
48      //pixel describes the position of the pixel being drawn in 20x20 blocks at the top of the screen
49      //this is used with num#_start to determine which part of the rom should be accessed
50      //used to draw the score numbers
51      assign pixel = (y_counter * 20) + x_counter + 2;
52
53      //look_up table module to find address start value using score values
54      numberSelect l1(.num_select(score1), .address_num(num1_start));
55      numberSelect l2(.num_select(score2), .address_num(num2_start));
56      numberSelect l3(.num_select(score3), .address_num(num3_start));
57
58      //assigns the address to access the rom using the pixel value and start value from lookup table
59      assign num1address = num1_start + pixel;
60      assign num2address = num2_start + pixel;
61      assign num3address = num3_start + pixel;
62
63      //pulls colour data from rom based on num#address
64      ROMnumbers ROM1(.address(num1address), .clock(clk), .q(numberdata1));
65      ROMnumbers ROM2(.address(num2address), .clock(clk), .q(numberdata2));
66      ROMnumbers ROM3(.address(num3address), .clock(clk), .q(numberdata3));
67
68      wire [17:0] imageaddress;
69
70      //gets rom address start position based on FSM output
71      backSelect back1(.back_select(current), .address_back(address_start));
72
73      //calculates current draw position based on y_out and x_out values
74      assign imageaddress = address_start + ((y_out - 20) * 160) + x_out + 3;
75
76      //pulls colour data from rom based on imageaddress
77      ROMbackground ROM0(.address(imageaddress), .clock(clk), .q(imagedata));
78
79      //declares fsm for the video output
```

```
 80      drawcontrol dc(.clk(clk),
 81                     .resetn(resetn),
 82                     .start(start),
 83                     .pre(pre),
 84                     .countdone(countdone),
 85                     .draw_done(draw_done),
 86                     .draw(draw),
 87                     .countstart(countstart),
 88                     .current(current),
 89                     .currentmain(currentmain));
 90
 91      //declares datapath for the video output
 92      drawdatapath(  .clk(clk),
 93                     .resetn(resetn),
 94                     .pre(pre),
 95                     .draw(draw),
 96                     .imagedata(imagedata),
 97                     .numberdata1(numberdata1),
 98                     .numberdata2(numberdata2),
 99                     .numberdata3(numberdata3),
100                     .colour_out(colour_out),
101                     .x_out(x_out),
102                     .y_out(y_out),
103                     .address(address),
104                     .draw_done(draw_done),
105                     .x_counter(x_counter),
106                     .y_counter(y_counter)
107                     );
108
109      //declares 2 second countdown module
110      countdown_2 cd3(.clk(clk), .loadEnable(countstart
     ), .countDone(countdone));
111
112  endmodule
113
114
115  //the drawcontrol module is the FSM for the Video
     Output
116  //it uses the state of the main FSM to determine
     which image should be shown
117  module drawcontrol(clk,
118                     resetn,
119                     start,
120                     pre,
121                     countdone,
122                     draw_done,
123                     draw,
```

```
124                     countstart,
125                     current,
126                     currentmain);
127
128     input clk;
129     input resetn;
130     input start;
131     input countdone;
132     input draw_done;
133     input [3:0] currentmain;
134     output reg pre;
135     output reg countstart;
136     output reg draw;
137     output reg [3:0] current;
138
139     reg [3:0] next;
140
141     //declares states
142     localparam  HOLD            = 'd0,
143                 DRAW_PRE        = 'd1,
144                 DRAW_START      = 'd2,
145                 DRAW_WIN        = 'd3,
146                 DRAW_LOSE       = 'd4,
147                 DRAW_GAMEOVER   = 'd5,
148                 DRAW_PRE_HAND   = 'd6,
149                 DRAW_PRE_WAIT   = 'd7,
150                 DRAW_START_WAIT = 'd8,
151                 DRAW_WIN_WAIT   = 'd9,
152                 DRAW_LOSE_WAIT  = 'd10,
153                 DRAW_GAMEOVER_WAIT  = 'd11,
154                 DRAW_PRE_HAND_WAIT  = 'd12;
155
156     //this section dictates the order of states and
    the
157     //requirements for switching between
    states
158     always@(*)
159     begin
160     case(current)
161         HOLD: begin
162             if (currentmain == 'd0 || currentmain == 'd7
    ) next = DRAW_PRE; //If the main state is PRESTART
    or READY goes to DRAW_PRE
163             else if (start) next = DRAW_START; //if
    start is true goes to DRAW_START
164             end
165         DRAW_PRE: begin
```

```
166             if (draw_done) next = DRAW_PRE_WAIT;
    //after drawing is done goes to DRAW_PRE_WAIT state
167             else if (currentmain == 'd8 || currentmain
    == 'd10) next = DRAW_LOSE; //If the main state is
    LOSE or AFTERLOSE goes to DRAW_LOSE
168         end
169         DRAW_PRE_WAIT:begin
170             if (countdone) next = DRAW_PRE_HAND;
    //after 2 seconds goes to DRAW_PRE_HAND
171             else if (start) next = DRAW_START;//if
    start is true goes to DRAW_START
172             else if (currentmain == 'd8 || currentmain
    == 'd10) next = DRAW_LOSE; //If the main state is
    LOSE or AFTERLOSE goes to DRAW_LOSE
173         end
174         DRAW_PRE_HAND: begin
175             if(draw_done) next = DRAW_PRE_HAND_WAIT;
    //after drawing is done goes to DRAW_PRE_HAND_WAIT
    state
176             else if (currentmain == 'd8 || currentmain
    == 'd10) next = DRAW_LOSE; //If the main state is
    LOSE or AFTERLOSE goes to DRAW_LOSE
177         end
178         DRAW_PRE_HAND_WAIT: begin
179             if (countdone) next = DRAW_PRE;  //after 2
    seconds goes to DRAW_PRE
180             else if (start) next = DRAW_START;//if
    start is true goes to DRAW_START
181             else if (currentmain == 'd8 || currentmain
    == 'd10) next = DRAW_LOSE; //If the main state is
    LOSE or AFTERLOSE goes to DRAW_LOSE
182         end
183         DRAW_START: begin
184             if(draw_done) next = DRAW_START_WAIT;
    //after drawing is done goes to DRAW_START_WAIT state
185             else if (currentmain == 'd8 || currentmain
    == 'd10) next = DRAW_LOSE; //If the main state is
    LOSE or AFTERLOSE goes to DRAW_LOSE
186         end
187         DRAW_START_WAIT: begin
188             if(currentmain == 'd4) next = DRAW_WIN;
    //after 2 seconds goes to DRAW_WIN
189             else if (currentmain == 'd8 || currentmain
    == 'd10) next = DRAW_LOSE; //If the main state is
    LOSE or AFTERLOSE goes to DRAW_LOSE
190         end
191         DRAW_WIN: begin
```

```
192             if(draw_done) next = DRAW_WIN_WAIT; //after
    drawing is done goes to DRAW_WIN_WAIT state
193           end
194           DRAW_WIN_WAIT: begin
195             if(countdone) next = DRAW_PRE;  //after 2
    seconds goes to DRAW_PRE
196             else if (currentmain == 'd0 || currentmain
    == 'd7) next = DRAW_PRE; //If the main state is
    PRESTART or READY goes to DRAW_PRE
197             else if (start) next = DRAW_START;//if
    start is true goes to DRAW_START
198           end
199           DRAW_LOSE: begin
200             if(draw_done) next = DRAW_LOSE_WAIT;
    //after drawing is done goes to DRAW_LOSE_WAIT state
201           end
202           DRAW_LOSE_WAIT: begin
203             if(countdone) next = DRAW_GAMEOVER;
    //after 2 seconds goes to DRAW_GAMEOVER
204             else if (currentmain == 'd0 || currentmain
    == 'd7) next = DRAW_PRE; //If the main state is
    PRESTART or READY goes to DRAW_PRE
205             else if (start) next = DRAW_START; //if
    start is true goes to DRAW_START
206           end
207           DRAW_GAMEOVER: begin
208             if(draw_done) next = DRAW_GAMEOVER_WAIT;
    //after drawing is done goes to DRAW_GAMEOVER_WAIT
    state
209           end
210           DRAW_GAMEOVER_WAIT: begin
211             if (currentmain == 'd0 || currentmain == 'd7
    ) next = DRAW_PRE; //If the main state is PRESTART
    or READY goes to DRAW_PRE
212           end
213           default: next = HOLD;
214       endcase
215       end
216
217       //sets the draw and countstart values for each
    state determining when drawing happens and when to
    wait
218       //pre value is set as well and used in
    drawdatapath to reset values
219       always@(*)
220       begin
221           pre             <= 0;
```

```
222             countstart      <= 0;
223             draw            <= 1;
224         case(current)
225             HOLD: begin
226                 pre             <= 1;
227                 countstart      <= 0;
228                 draw            <= 1;
229             end
230             DRAW_PRE: begin
231                 pre             <= 0;
232                 countstart      <= 1;
233                 draw            <= 1;
234             end
235             DRAW_PRE_WAIT: begin
236                 pre             <= 0;
237                 countstart      <= 0;
238                 draw            <= 0;
239             end
240             DRAW_PRE_HAND: begin
241                 pre             <= 0;
242                 countstart      <= 0;
243                 draw            <= 1;
244             end
245             DRAW_PRE_HAND_WAIT: begin
246                 pre             <= 0;
247                 countstart      <= 1;
248                 draw            <= 0;
249             end
250             DRAW_START: begin
251                 pre             <= 0;
252                 draw            <= 1;
253             end
254             DRAW_START_WAIT: begin
255                 pre             <= 0;
256                 countstart      <= 1;
257                 draw            <= 0;
258             end
259             DRAW_WIN: begin
260                 pre             <= 0;
261                 countstart      <= 0;
262                 draw            <= 1;
263             end
264             DRAW_WIN_WAIT: begin
265                 pre             <= 0;
266                 countstart      <= 1;
267                 draw            <= 0;
268             end
```

```
269             DRAW_LOSE : begin
270                 pre             <= 0;
271                 countstart      <= 0;
272                 draw            <= 1;
273             end
274             DRAW_LOSE_WAIT : begin
275                 pre             <= 0;
276                 countstart      <= 1;
277                 draw            <= 0;
278             end
279             DRAW_GAMEOVER : begin
280                 pre             <= 0;
281                 countstart      <= 0;
282                 draw            <= 1;
283             end
284             DRAW_GAMEOVER_WAIT : begin
285                 pre             <= 0;
286                 countstart      <= 1;
287                 draw            <= 0;
288             end
289         endcase
290         end
291
292         //sets state transition to clock edge
293         always@(posedge clk)
294         begin
295         if(!resetn)
296             current = HOLD;
297         else
298             current = next;
299         end
300
301     endmodule
302
303
304     //the drawdatapath is used to set the draw location
        and give it a colour
305     module drawdatapath( clk,
306                          resetn,
307                          pre,
308                          draw,
309                          imagedata,
310                          numberdata1,
311                          numberdata2,
312                          numberdata3,
313                          colour_out,
314                          x_out,
```

```
315                         y_out,
316                         address,
317                         draw_done,
318                         x_counter,
319                         y_counter
320                         );
321
322     input clk;
323     input resetn;
324     input pre;
325     input draw;
326     input [23:0] imagedata;
327     input [23:0] numberdata1;
328     input [23:0] numberdata2;
329     input [23:0] numberdata3;
330     output reg [23:0] colour_out;
331     output reg [7:0] x_out;
332     output reg [6:0] y_out;
333     output reg [16:0] address;
334     output reg draw_done;
335     output reg [7:0] x_counter;
336     output reg [6:0] y_counter;
337
338     //x and y denote the start position for each
    destinct drawing cell
339     reg [7:0] x;
340     reg [6:0] y;
341
342     always@(posedge clk)
343     begin
344
345     //combines start positions and their respective
    counters
346     x_out <= x + x_counter;
347     y_out <= y + y_counter;
348
349         //resets values to 0 when pre is true
350         if(pre) begin
351             x <= 0;
352             y <= 0;
353             x_counter <= 0;
354             y_counter <= 0;
355             x_out <= 0;
356             y_out <= 0;
357             draw_done <= 0;
358         end
359
```

```
360         if(draw) begin
361
362             //sets colour output for the first 20 rows
    with the last 3 20x20 square displaying the score
363             if(x == 0 && y == 0) colour_out <= 'b0;
364             else if(x == 20 && y == 0) colour_out <= 'b0;
365             else if(x == 40 && y == 0) colour_out <= 'b0;
366             else if(x == 60 && y == 0) colour_out <= 'b0;
367             else if(x == 80 && y == 0) colour_out <= 'b0;
368             else if(x == 100 && y == 0) colour_out <=
    numberdata3;
369             else if(x == 120 && y == 0) colour_out <=
    numberdata2;
370             else if(x == 140 && y == 0) colour_out <=
    numberdata1;
371             //sets colour output for the rest of the
    display
372             else if(y == 20) begin
373                 colour_out <= imagedata;
374             end
375
376             //increments draw location through the
    first 20 rows
377             //it increments x 20 pixels then increments
    y by 1
378             //when y and x get to the bottom right
    corner of the 20x20 square
379             //it moves to the top left of the next square
380             //after the row of squares is done y is set
    to 20 and the main image is drawn
381             if(y == 0) begin
382                 if(x_counter < 18) begin
383                     x_counter <= x_counter + 1;
384                 end
385                 else if(x_counter == 18) begin
386                     x_counter <= 0;
387                     if(y_counter < 19) y_counter <=
    y_counter + 1;
388                     else begin
389                         x_counter <= 0;
390                         y_counter <= 0;
391                         if(x < 160) x <= x + 20;
392                         else begin
393                             y<=20;
394                             x<=0;
395                             x_counter <= 0;
396                             y_counter <= 0;
```

```
397                     end
398                 end
399             end
400         end
401
402
403         //when y == 20 the drawn pixels are incremented across the display and at the end moved down by one
404         else if(y == 20) begin
405             x <= 0;
406             y <= 20;
407             if(x_counter < 159) begin
408                 x_counter <= x_counter + 1;
409             end
410             else if(x_counter == 159) begin
411                 x_counter <= 0;
412                 if(y_counter < 100) y_counter <= y_counter + 1;
413                 else begin
414                     draw_done <= 1;
415                     x <= 0;
416                     y <= 0;
417                 end
418             end
419
420         end
421
422
423
424     end
425     else begin
426         x <= 0;
427         y <= 0;
428         x_counter <= 0;
429         y_counter <= 0;
430         x_out <= 0;
431         y_out <= 0;
432         draw_done <= 0;
433     end
434 end
435
436
437 endmodule
438
439 //selects the starting address to draw the main image base on the FSM output
```

```
440   module backSelect(back_select, address_back);
441      input [3:0] back_select;
442      output reg [17:0] address_back;
443
444      always@(*)
445      begin
446         case(back_select)
447            4'd0     :  address_back = 'd0;
      //HOLD
448            4'd1     :  address_back = 'd0;
      //PRE
449            4'd2     :  address_back = 'd16000;
      //START
450            4'd3     :  address_back = 'd32000;
      //WIN
451            4'd4     :  address_back = 'd48000;
      //LOSE
452            4'd5     :  address_back = 'd64000;
      //GAMEOVER
453            4'd6     :  address_back = 'd16000;
      //DRAW_PRE_HAND
454            4'd7     :  address_back = 'd0;
      //PRE
455            4'd8     :  address_back = 'd16000;
      //START
456            4'd9     :  address_back = 'd32000;
      //WIN
457            4'd10    :  address_back = 'd48000;
      //LOSE
458            4'd11    :  address_back = 'd64000;
      //GAMEOVER
459            4'd12    :  address_back = 'd16000;
      //DRAW_PRE_HAND
460            default  :  address_back = 'd0;
      //0
461         endcase
462      end
463
464      endmodule
465
466   //selects the starting address to draw the number
      for each score digit based
467   //on score inputs from the main game datapath
468   module numberSelect(num_select, address_num);
469      input [4:0] num_select;
470      output reg [11:0] address_num;
471
```

```
472         always@(*)
473         begin
474            case(num_select)
475               4'b0000      :  address_num = 'd0;
                     //0
476               4'b0001      :  address_num = 'd400;
                   //1
477               4'b0010      :  address_num = 'd800;
                   //2
478               4'b0011      :  address_num = 'd1200;
                  //3
479               4'b0100      :  address_num = 'd1600;
                  //4
480               4'b0101      :  address_num = 'd2000;
                  //5
481               4'b0110      :  address_num = 'd2400;
                  //6
482               4'b0111      :  address_num = 'd2800;
                  //7
483               4'b1000      :  address_num = 'd3200;
                  //8
484               4'b1001      :  address_num = 'd3600;
                  //9
485               default   :  address_num = 'd0;
      //0
486            endcase
487         end
488
489         endmodule
490
491         //2 second countdown clock
492      module countdown_2(clk, loadEnable, countDone);
493          input clk, loadEnable;
494          output reg countDone;
495
496          reg [32:0]countVal;
497
498         always @(posedge clk) begin
499         if (!loadEnable) begin
500            countVal <= 'd100000000;
501            countDone <= 0;
502         end
503
504         else if (countVal == 'd0) begin
505               countDone <= 1;
506         end
507
```

```
508         else if(countVal != 'd0) begin
509             countVal <= countVal - 1;
510             countDone <= 0;
511         end
512         end
513 
514     endmodule
```