

```

1
2
3 //***** LEVEL 1 FSM *****/
4 module lvl1FSM (
5     clk, reset, go, goJump, right, left, up, down,
6     resetAddress, drM, drML, erM, jump, jumpL, jumping, falling, drStage1,
7     moveRight,
8     done, jumpCounter, fall,
9     writeEn, rightColour, leftColour, ground, outofBounds, pipe, lvl1,
10    start, dead);
11
12    input [4:0] jumpCounter;
13    input clk, reset, left, up, down, go, goJump, right, ground, outofBounds, pipe, lvl1,
14    fall, start, dead;
15    input [11:0] rightColour, leftColour;
16
17    reg Facingright, Facingleft;
18    initial Facingright = 0;
19    initial Facingleft = 0;
20
21    input done;
22    output reg resetAddress, drM, drML, erM, jump, jumpL, jumping, falling, drStage1, writeEn
23    , moveRight;
24
25    reg [10:0] current_state, next_state;
26
27    localparam nothing      = 10'd0,
28    drawstage1             = 10'd1,
29    waitmario              = 10'd2,
30    drawmarioRight         = 10'd3,
31    drawmarioLeft          = 10'd4,
32    jumpmarioRight         = 10'd5,
33    jumpmarioLeft          = 10'd6,
34    erasemario             = 10'd7,
35    jumperase              = 10'd8,
36    fallingerase           = 10'd9,
37    waitState              = 10'd10;
38
39    always @(*)
40    begin: state_table
41        case(current_state)
42            waitState: next_state = lvl1 ? drawstage1 : waitState;
43
44            nothing: begin //wait state that controls direction control path should take
45                if (go && ((right && !outofBounds) || (left && !outofBounds)))
46                    next_state = erasemario;
47                else if (!ground && jumpCounter == 0 && !dead)
48                    next_state = fallingerase;
49                else if (up && goJump)
50                    next_state = jumperase;
51                else if (down && pipe)
52                    next_state = nothing;
53                else
54                    next_state = nothing;
55            end
56            drawstage1: //draws level one - initial state
57            begin
58                if (done)
59                    next_state = waitmario;
60                else
61                    next_state = drawstage1;
62            end
63            waitmario: next_state = drawmarioRight; //resets address
64
65            drawmarioRight: //draws mario in right orientation
66            begin
67                if (done)
68                    next_state = nothing;
69                else
70                    next_state = drawmarioRight;
71            end
72        end
73    end

```

```
70 drawmarioLeft: begin //draws mario left orientation
71   if (done)
72     next_state = nothing;
73   else
74     next_state = drawmarioLeft;
75 end
76
77 jumpmarioRight: begin //draws mario jumping to the right
78   if (done && !ground && goJump && fall)
79     next_state = fallingerase;
80   else if (done && ground && goJump && fall)
81     next_state = drawmarioRight;
82   else if (done && goJump && jumpCounter < 5'd22)
83     next_state = jumpingerase;
84   else if (done && goJump && jumpCounter == 5'd22)
85     next_state = erasemario;
86   else if (done && ground && goJump)
87     next_state = drawmarioRight;
88   else
89     next_state = jumpmarioRight;
90 end
91
92 jumpmarioLeft: begin //draws mario jumping to the left
93   if (done && !ground && goJump && fall)
94     next_state = fallingerase;
95   else if (done && ground && goJump && fall)
96     next_state = drawmarioLeft;
97   else if (done && goJump && jumpCounter < 5'd22)
98     next_state = jumpingerase;
99   else if (done && goJump && jumpCounter == 5'd22)
100     next_state = erasemario;
101   else if (done && ground && goJump)
102     next_state = drawmarioLeft;
103   else
104     next_state = jumpmarioLeft;
105 end
106
107 jumpingerase: begin
108   if (done && Facingright)
109     next_state = jumpmarioRight;
110   else if (done && Facingleft)
111     next_state = jumpmarioLeft;
112   else
113     next_state = jumpingerase;
114 end
115
116 fallingerase: begin
117   if (!ground && done && Facingright)
118     next_state = jumpmarioRight;
119   else if (ground && done && Facingright)
120     next_state = drawmarioRight;
121   else if (!ground && done && Facingleft)
122     next_state = jumpmarioLeft;
123   else if (ground && done && Facingleft)
124     next_state = drawmarioLeft;
125   else
126     next_state = fallingerase;
127 end
128
129 erasemario: //erases mario by drawing background
130 begin
131   if(done && right)
132     next_state = drawmarioRight;
133   else if (done && left)
134     next_state = drawmarioLeft;
135   else if (done && Facingright)
136     next_state = drawmarioRight;
137   else if (done && Facingleft)
138     next_state = drawmarioLeft;
139   else
140     next_state = erasemario;
141 end
142
```

```
143         default: next_state = waitState; //initial state draw background
144     endcase
145 end
146
147 always@(posedge clk)
148 begin: logic
149
150     drM = 0;
151     drML = 0;
152     writeEn = 0;
153     drStage1 = 0;
154     resetAddress = 0;
155     moveRight = 0;
156     erM = 0;
157     jump = 0;
158     jumpL = 0;
159     jumping = 0;
160     falling = 0;
161
162     case(current_state)
163
164     nothing: begin
165         resetAddress = 1;
166         writeEn = 0;
167     end
168     drawstage1: begin
169         drStage1 = 1;
170         writeEn = 1;
171     end
172     waitmario: begin
173         writeEn = 0;
174         resetAddress = 1;
175     end
176     drawmarioRight: begin
177         drM = 1;
178         writeEn = 1;
179         Facingright = 1;
180         Facingleft = 0;
181     end
182     drawmarioLeft: begin
183         drML = 1;
184         writeEn = 1;
185         Facingleft = 1;
186         Facingright = 0;
187     end
188     jumpmarioRight: begin
189         jump = 1;
190         writeEn = 1;
191     end
192     jumpmarioLeft: begin
193         jumpL = 1;
194         writeEn = 1;
195     end
196     jumperase: begin
197         writeEn = 1;
198         jumping = 1;
199     end
200     fallingerase: begin
201         writeEn = 1;
202         falling = 1;
203     end
204     erasemario: begin
205         erM = 1;
206         writeEn = 1;
207     end
208
209     endcase
210 end
211
212 always@(posedge clk)
213 begin: state_FFs
214     if (start || dead)
215         current_state <= waitState;
```

```

216         else
217             current_state <= next_state;
218         end
219     endmodule
220
221
222
223
224 //***** LEVEL 2 FSM *****/
225 module lvl2FSM (
226     clk, reset, go, goJump, right, left, up, down,
227     resetAddress, drM, drML, erM, jump, jumpL, jumping, falling, drStage2,
228     moveRight,
229     done, jumpCounter, fall,
230     writeEn, rightColour, leftColour, ground, outofBounds, next, lvl2,
231     start);
232
233     input [4:0] jumpCounter;
234     input clk, reset, left, up, down, go, goJump, right, ground, outofBounds, next, lvl2,
235     fall, start;
236     input [11:0] rightColour, leftColour;
237
238     reg Facingright, Facingleft;
239     initial Facingright = 0;
240     initial Facingleft = 0;
241
242     input done;
243     output reg resetAddress, drM, drML, erM, jump, jumpL, jumping, falling, drStage2, writeEn
244     , moveRight;
245
246     reg [10:0] current_state, next_state;
247
248     localparam nothing      = 10'd0,
249     drawstage2             = 10'd1,
250     waitmario              = 10'd2,
251     drawmarioRight         = 10'd3,
252     drawmarioLeft          = 10'd4,
253     jumpmarioRight          = 10'd5,
254     jumpmarioLeft           = 10'd6,
255     erasemario              = 10'd7,
256     jumpingerase            = 10'd8,
257     fallingerase            = 10'd9,
258     waitState               = 10'd10;
259
260     always @(*)
261     begin: state_table
262         case(current_state)
263             waitState: next_state = lvl2 ? drawstage2 : waitState;
264             nothing: begin //wait state that controls direction control path should take
265                 if (go && ((right && !outofBounds) || (left && !outofBounds)))
266                     next_state = erasemario;
267                 else if (!ground && jumpCounter == 0)
268                     next_state = fallingerase;
269                 else if (up && goJump && !outofBounds)
270                     next_state = jumpingerase;
271                 else
272                     next_state = nothing;
273             end
274             drawstage2: //draws level one - initial state
275             begin
276                 if (done)
277                     next_state = waitmario;
278                 else
279                     next_state = drawstage2;
280             end
281             waitmario: next_state = drawmarioRight; //resets address
282             drawmarioRight: //draws mario in right orientation
283             begin
284                 if (done)
285                     next_state = nothing;

```

```
285         else
286             next_state = drawmarioRight;
287     end
288
289     drawmarioLeft: begin //draws mario left orientation
290         if (done)
291             next_state = nothing;
292         else
293             next_state = drawmarioLeft;
294     end
295
296     jumpmarioRight: begin //draws mario jumping to the right
297         if (done && !ground && goJump && fall)
298             next_state = fallingerase;
299         else if (done && ground && goJump && fall)
300             next_state = drawmarioRight;
301         else if (done && goJump && jumpCounter < 5'd22)
302             next_state = jumpingerase;
303         else if (done && goJump && jumpCounter == 5'd22)
304             next_state = erasemario;
305         else if (done && ground && goJump)
306             next_state = drawmarioRight;
307         else
308             next_state = jumpmarioRight;
309     end
310
311     jumpmarioLeft: begin //draws mario jumping to the left
312         if (done && !ground && goJump && fall)
313             next_state = fallingerase;
314         else if (done && ground && goJump && fall)
315             next_state = drawmarioLeft;
316         else if (done && goJump && jumpCounter < 5'd22)
317             next_state = jumpingerase;
318         else if (done && goJump && jumpCounter == 5'd22)
319             next_state = erasemario;
320         else if (done && ground && goJump)
321             next_state = drawmarioLeft;
322         else
323             next_state = jumpmarioLeft;
324     end
325
326     jumpingerase: begin
327         if (done && Facingright)
328             next_state = jumpmarioRight;
329         else if (done && Facingleft)
330             next_state = jumpmarioLeft;
331         else
332             next_state = jumpingerase;
333     end
334
335     fallingerase: begin
336         if (!ground && done && Facingright)
337             next_state = jumpmarioRight;
338         else if (ground && done && Facingright)
339             next_state = drawmarioRight;
340         else if (!ground && done && Facingleft)
341             next_state = jumpmarioLeft;
342         else if (ground && done && Facingleft)
343             next_state = drawmarioLeft;
344         else
345             next_state = fallingerase;
346     end
347
348     erasemario: //erases mario by drawing background
349     begin
350         if(done && right)
351             next_state = drawmarioRight;
352         else if (done && left)
353             next_state = drawmarioLeft;
354         else if (done && Facingright)
355             next_state = drawmarioRight;
356         else if (done && Facingleft)
357             next_state = drawmarioLeft;
```

```
358         else
359             next_state = erasemario;
360         end
361
362         default: next_state = waitState; //initial state draw background
363     endcase
364 end
365
366 always@(posedge clk)
367 begin: logic
368
369     drM = 0;
370     drML = 0;
371     writeEn = 0;
372     drStage2 = 0;
373     resetAddress = 0;
374     moveRight = 0;
375     erM = 0;
376     jump = 0;
377     jumpL = 0;
378     jumping = 0;
379     falling = 0;
380
381     case(current_state)
382
383     nothing: begin
384         resetAddress = 1;
385         writeEn = 0;
386     end
387     drawstage2: begin
388         drStage2 = 1;
389         writeEn = 1;
390     end
391     waitmario: begin
392         writeEn = 0;
393         resetAddress = 1;
394     end
395     drawmarioRight: begin
396         drM = 1;
397         writeEn = 1;
398         Facingright = 1;
399         Facingleft = 0;
400     end
401     drawmarioLeft: begin
402         drML = 1;
403         writeEn = 1;
404         Facingleft = 1;
405         Facingright = 0;
406     end
407     jumpmarioRight: begin
408         jump = 1;
409         writeEn = 1;
410     end
411     jumpmarioLeft: begin
412         jumpL = 1;
413         writeEn = 1;
414     end
415     jumpingerase: begin
416         writeEn = 1;
417         jumping = 1;
418     end
419     fallingerase: begin
420         writeEn = 1;
421         falling = 1;
422     end
423     erasemario: begin
424         erM = 1;
425         writeEn = 1;
426     end
427
428     endcase
429 end
430
```

```

431     always@(posedge clk)
432     begin: state_FFs
433         if (start)
434             current_state <= waitState;
435         else
436             current_state <= next_state;
437     end
438
439 endmodule
440
441 //***** LEVEL 3 FSM *****
442 module lvl3FSM (
443     clk, reset, go, goJump, right, left, up, down,
444     resetAddress, drM, drML, erM, jump, jumpL, jumping, falling, drStage3,
445     moveRight,
446     done, jumpCounter, fall,
447     writeEn, rightColour, leftColour, ground, outofBounds, flag, lvl3,
448     start);
449     input [4:0] jumpCounter;
450     input clk, reset, left, up, down, go, goJump, right, ground, outofBounds, flag, lvl3,
451     fall, start;
452     input [11:0] rightColour, leftColour;
453
454     reg Facingright, Facingleft;
455     initial Facingright = 0;
456     initial Facingleft = 0;
457
458     input done;
459     output reg resetAddress, drM, drML, erM, jump, jumpL, jumping, falling, drStage3, writeEn
460     , moveRight;
461
462     reg [10:0] current_state, next_state;
463
464     localparam nothing          = 10'd0,
465     drawstage3                 = 10'd1,
466     waitmario                  = 10'd2,
467     drawmarioRight             = 10'd3,
468     drawmarioLeft              = 10'd4,
469     jumpmarioRight             = 10'd5,
470     jumpmarioLeft              = 10'd6,
471     erasemario                 = 10'd7,
472     jumperase                   = 10'd8,
473     fallingerase               = 10'd9,
474     waitState                   = 10'd10;
475
476     always @(*)
477     begin: state_table
478         case(current_state)
479             waitState: next_state = lvl3 ? drawstage3 : waitState;
480             nothing: begin //wait state that controls direction control path should take
481                 if (go && ((right && !outofBounds) || (left && !outofBounds)))
482                     next_state = erasemario;
483                 else if (!ground && jumpCounter == 0)
484                     next_state = fallingerase;
485                 else if (up && goJump)
486                     next_state = jumperase;
487                 else
488                     next_state = nothing;
489             end
490             drawstage3: //draws level one - initial state
491             begin
492                 if (done)
493                     next_state = waitmario;
494                 else
495                     next_state = drawstage3;
496             end
497             waitmario: next_state = drawmarioRight; //resets address
498             drawmarioRight: //draws mario in right orientation
499             begin

```

```
500         if (done)
501             next_state = nothing;
502         else
503             next_state = drawmarioRight;
504     end
505
506     drawmarioLeft: begin //draws mario left orientation
507         if (done)
508             next_state = nothing;
509         else
510             next_state = drawmarioLeft;
511         end
512
513     jumpmarioRight: begin //draws mario jumping to the right
514         if (done && !ground && goJump && fall)
515             next_state = fallingerase;
516         else if (done && ground && goJump && fall)
517             next_state = drawmarioRight;
518         else if (done && goJump && jumpCounter < 5'd22)
519             next_state = jumpingerase;
520         else if (done && goJump && jumpCounter == 5'd22)
521             next_state = erasemario;
522         else if (done && ground && goJump)
523             next_state = drawmarioRight;
524         else
525             next_state = jumpmarioRight;
526         end
527
528     jumpmarioLeft: begin //draws mario jumping to the left
529         if (done && !ground && goJump && fall)
530             next_state = fallingerase;
531         else if (done && ground && goJump && fall)
532             next_state = drawmarioLeft;
533         else if (done && goJump && jumpCounter < 5'd22)
534             next_state = jumpingerase;
535         else if (done && goJump && jumpCounter == 5'd22)
536             next_state = erasemario;
537         else if (done && ground && goJump)
538             next_state = drawmarioLeft;
539         else
540             next_state = jumpmarioLeft;
541         end
542
543     jumpingerase: begin
544         if (done && Facingright)
545             next_state = jumpmarioRight;
546         else if (done && Facingleft)
547             next_state = jumpmarioLeft;
548         else
549             next_state = jumpingerase;
550         end
551
552     fallingerase: begin
553         if (!ground && done && Facingright)
554             next_state = jumpmarioRight;
555         else if (ground && done && Facingright)
556             next_state = drawmarioRight;
557         else if (!ground && done && Facingleft)
558             next_state = jumpmarioLeft;
559         else if (ground && done && Facingleft)
560             next_state = drawmarioLeft;
561         else
562             next_state = fallingerase;
563         end
564
565     erasemario: //erases mario by drawing background
566     begin
567         if(done && right)
568             next_state = drawmarioRight;
569         else if (done && left)
570             next_state = drawmarioLeft;
571         else if (done && Facingright)
572             next_state = drawmarioRight;
```



```
573         else if (done && Facingleft)
574             next_state = drawmarioLeft;
575         else
576             next_state = erasemario;
577     end
578
579     default: next_state = waitState; //initial state draw background
580 endcase
581 end
582
583 always@(posedge clk)
584 begin: logic
585
586     drM = 0;
587     drML = 0;
588     writeEn = 0;
589     drStage3 = 0;
590     resetAddress = 0;
591     moveRight = 0;
592     erM = 0;
593     jump = 0;
594     jumpL = 0;
595     jumping = 0;
596     falling = 0;
597
598     case(current_state)
599
600     nothing: begin
601         resetAddress = 1;
602         writeEn = 0;
603     end
604     drawstage3: begin
605         drStage3 = 1;
606         writeEn = 1;
607     end
608     waitmario: begin
609         writeEn = 0;
610         resetAddress = 1;
611     end
612     drawmarioRight: begin
613         drM = 1;
614         writeEn = 1;
615         Facingright = 1;
616         Facingleft = 0;
617     end
618     drawmarioLeft: begin
619         drML = 1;
620         writeEn = 1;
621         Facingleft = 1;
622         Facingright = 0;
623     end
624     jumpmarioRight: begin
625         jump = 1;
626         writeEn = 1;
627     end
628     jumpmarioLeft: begin
629         jumpL = 1;
630         writeEn = 1;
631     end
632     jumperase: begin
633         writeEn = 1;
634         jumping = 1;
635     end
636     fallingerase: begin
637         writeEn = 1;
638         falling = 1;
639     end
640     erasemario: begin
641         erM = 1;
642         writeEn = 1;
643     end
644 endcase
645
```

```
646     end
647
648     always@(posedge clk)
649     begin: state_FF0
650         if (start)
651             current_state <= waitState;
652         else
653             current_state <= next_state;
654     end
655
656 endmodule
657
```