به نام خدا

امید حیدری 401104859

پروژه جبرانی(آسانسور)

در این سوال ما باید عملکرد آسانسوری را شبیه سازی کنیم.که شامل طبقه همکف و 4 طبقه دیگر است.در نظر داریم که هر طبقه یک کلید در خارج طبقه و یک کلید در داخل است.دو فایل ساخته که یکی همان بالابر و دیگری میشود کترلر آن.

ابندا ماژول بالابر را پیاده سازی میکنیم.

ماژول بالابر:

```
1   module BalaBar (
2       input clk,                              // Clock
3       input rst,                             // signal for reset
4       input [1:0] state,                      // state of BalaBar
5       output reg [4:0] floor_cur,            // Current floor
6       output reg [1:0] floor_state [4:0]     //  floor states
7   );
```

سه سیگنال ورودی دارد که یکی کلاک است و دیگری سیگنال ریست و دیگری سیگنال وضعیت که در هر لحظه وضعیت بالابر را نشان میدهد. همچنین در خروجی نیز یک سیگنال که طبقه فعلی آسانسور و همچنین یک آرایه برای تعیین وضعیت طبقات است. مثلا 3 یعنی توقف آسانسور در آن طبقه و یا 2 یعنی خانه فعلی آسانسور اما متحرک و 1 یعنی خانه قبلی آسانسور اما متحرک.

```
8
9       // local variable
10      integer i;
11      localparam STOP = 2'b00;
12      localparam UP_going = 2'b11;
13      localparam DOWN_going = 2'b10;
```

متغیر های داخلی را داریم که شامل i برای حلقه ها و همچنین سه متغیر تعریف شده برای وضعیت آسانسور که 00 به معنی توقف و 10 به معنی حرکت به سمت پایین و 11 به معنی حرکت به سمت بالاست.

```
15      // initial
16      initial begin
17          floor_cur = {5{1'b0}};
18          for( i = 0; i < 5; i=i+1)
19              floor_state[i] = 2'b00;
20      end
```

در بخش اولیه ابتدا وضعیت تمامی طبقات برابر 00 میگذاریم که یعنی عدم حضور آسانسور.همچنین طبقه فعلی را هیچکدام نمیگذارد.

```
22      // update states
23      always @(posedge clk or posedge rst) begin
24          for( i = 0; i < 5; i=i+1)
25              floor_state[i] = 2'b00;
26
27          if (rst) begin
28              floor_cur = {5{1'b0}};
29          end else begin
30              case (state)
31                  STOP: begin
32                      floor_state[floor_cur] = 2'b11;
33                      #100;
34                  end
35
36                  DOWN_going: begin
37                      if (floor_cur > 0) begin
38                          floor_state[floor_cur] = 2'b01;
39                          floor_cur = floor_cur - 1;
40                          floor_state[floor_cur] = 2'b10;
41                      end
42                      #10;
43                  end
44
45                  UP_going: begin
46                      if (floor_cur < 5) begin
47                          floor_state[floor_cur] = 2'b01;
48                          floor_cur = floor_cur + 1;
49                          floor_state[floor_cur] = 2'b10;
50                      end
51                      #10;
52                  end
53
54              endcase
55          end
56      end
```

بخش اصلی این ماژول که با 1 شدن کلاک و یا 1 شدن ریست فعال میشد.اگر ریست فعال بود که طبقه فعلی را همان همکف میگذارد و همچنین همیشه ابتدا وضعیت تمامی طبقات را برابر 00 میکند.
سپس با توجه به نحوه حرکت وضعیت طبقات را بروزرسانی میکند.برای مثال اگر در وضعیت ایستادن بود طبقه فعلی را برابر 11 میکند و یا اگر در حرکت بود با توجه به جهت حرکتش طبقه قبلی را 01 و طبقه جدید را 10 میکند.

این ماژول در واقع حرکت بالابر را شبیه سازی میکند و برای نمایش خروجی به کار می رود.

ماژول کنترلر :

```
module BalaBArController (
    input clk,                          // Clock
    input rst,                         // signal for reset
    input [4:0] buttons_ex,       // request from out
    input [4:0] buttons_in,       // request from in
    input [4:0] floor_cur,          // Current floor
    output reg direction,              // Move Direction signal : 1 ->  up, 0 -> down
    output reg motor                   // Motor movement : 1 -> motor, 0 -> stop
);

```

پورت های ورودی شامل کلاک و ریست و همچنین طبقه فعلی آسانسور و همچنین کلید های داخلی و خارجی آسانسور میشود.که این ها هر بیتشان که 1 شد یعنی برای آن طبقه درخواست وجود دارد. همچنین دو تا بیت خروجی که یکی نشان دهنده آن که آیا اصلا حرکت میکند یا ساکن است که این همان motor است و دیگری direction میباشد که اگر 1 بود یعنی حرکت به بالا و 0 یعنی حرکت به پایین.

```

    // local variable
    reg [4:0] requests;
    integer i;
    integer target;
    bit [2:0] queue [$:4];
    bit new_target;
    localparam STOP = 2'b00;
    localparam UP_going = 2'b11;
    localparam DOWN_going = 2'b10;
```

متغیر های داخلی که مانند ماژول قبلی شامل i و متغیر های داخلی STOP و ... میباشد. همچنین شامل صف برای رعایت ترتیب و متغیری برای درخواست مقصد جدید و یک آرایه که بیت مد نظر آن اگر 1 بود یعنی برای آن طبقه در خواستی وجود دارد.

```
21      // initial
22      initial begin
23          {motor, direction} = STOP;
24          requests = {5{1'b0}};
25      end
26
27      function automatic void clear();
28          int val;
29          while (queue.size() != 0)
30              val = queue.pop_front();
31      endfunction
```

تعریف اولیه که آمده و تمامی درخواست هارا برابر 0 گذاشته و وضعیت توقف را برای بالابر در نظر دارد.
همچنین در بخش پایین شاهد یک تابع هستیم که مسئول خالی کردن صف است.

```
33      // update requests
34      always @(negedge clk or posedge rst) begin
35          if (rst) begin
36              new_target <= 1'b1;
37              requests <= {5{1'b0}};
38          end
39          else
40              requests <= buttons_in | requests | buttons_ex;
41      end
```

در این بخش اگر ریست فعال بوده ریست میکنیم و تمامی درخواست ها برابر 0 میشوند و همچنین مفدار درخواست هارا با توجه به وضعیت کلید های درونی وبیرونی معلوم میکند.

```
43      // updating destination
44      always @(posedge rst) begin
45          if (rst)
46              target = floor_cur;
47      end
```

در این بخش نیز در صورت ریست کردن هدف را برابر طبقه فعلی میگذارد.

```
50      // always for the control logic
51      always @(negedge clk or posedge rst) begin
52          if (rst)
53              {motor, direction} <= STOP;
54          else begin
55              case ({motor, direction})
56                  STOP: begin
57                      if (requests != {5{1'b0}}) begin
58                          if (queue.size() > 0 && new_target)begin
59                              target = queue.pop_front();
60                              new_target = 0;
61                          end
62                          if (target > floor_cur)
63                              {motor, direction} <= UP_going;
64                          else if (target < floor_cur)
65                              {motor, direction} <= DOWN_going;
66                      end
67                      if (target == floor_cur) begin
68                              requests[floor_cur] <= 1'b0;
69                              new_target = 1;
70                          end
71                  end
72
73                  DOWN_going: begin
74                      if (0 == floor_cur) begin
75                          {motor, direction} <= STOP;
76                      end
77                      else begin
78                          if (requests[floor_cur] == 1'b1) begin
79                              {motor, direction} <= STOP;
80                              requests[floor_cur] <= 1'b0;
81                          end
82                          if (target == floor_cur) begin
83                              new_target = 1;
84                              {motor, direction} = STOP;
```

```
84                                  {motor, direction} = STOP;
85                              end
86                          end
87                      end
88
89                      UP_going: begin
90                          if (5 == floor_cur) begin
91                              {motor, direction} <= STOP;
92                          end
93                          else begin
94                              if (requests[floor_cur] == 1'b1) begin
95                                  {motor, direction} <= STOP;
96                                  requests[floor_cur] <= 1'b0;
97                              end
98                              if (target == floor_cur) begin
99                                  new_target = 1;
100                                 {motor, direction} = STOP;
101                             end
102                         end
103                     end
104                 endcase
105             end
106     end
107
```

در این بخش به درخواست ها رسیدگی میشود با توجه به وضعیت موتور و وضع درخواست ها.اگر نیاز است که به صف و و الا در مسیر به آن طبقه میپردازد.

```
108
109     // always for the request queue
110     always @(negedge clk or posedge rst) begin
111         if (rst)
112             clear();
113
114         else begin
115             for (i = 0; i < 5 ; i= i+1) begin
116                 if (buttons_in[i] | buttons_ex[i] == 1'b1) begin
117                     case ({motor, direction})
118                         STOP: begin
119                             if (target > floor_cur) begin
120                                 if (i < floor_cur)
121                                     queue.push_back(i);
122                                 else if (i > target)
123                                     queue.push_back(i);
124                             end
125                             else if (target < floor_cur) begin
126                                 if (i > floor_cur)
127                                     queue.push_back(i);
128                                 else if (i < target)
129                                     queue.push_back(i);
130                             end
131                             else
132                                 queue.push_back(i);
133                         end
134
135                         DOWN_going: begin
136                             if(i < target || i > floor_cur)
137                                 queue.push_back(i);
138                         end
139
140                         UP_going: begin
141                             if(i > target || floor_cur > i)
142                                 queue.push_back(i);
136                             if(i < target || i > floor_cur)
137                                 queue.push_back(i);
138                         end
139
140                         UP_going: begin
141                             if(i > target || floor_cur > i)
142                                 queue.push_back(i);
143                         end
144
145                     endcase
146                 end
147             end
148         end
149     end
150
151 endmodule
152
```

در بخش پایانی نیز با توجه به حرکت موتور و طبقه آن دستور میدهد که بالابر چگونه حرکت کند.

تست

```
module TESTBENCH;

    // variable
    reg clk;
    reg rst;
    reg [4:0] buttons_in;
    reg [4:0] buttons_ex;
    wire motor;
    wire direction;
    wire [4:0] floor_cur;
    wire [1:0] floor_state [0:4];

    // instance
    BalaBar balaBar (
        .clk(clk),
        .rst(rst),
        .state({motor, direction}),
        .floor_state(floor_state),
        .floor_cur(floor_cur)
    );

    BalaBArController controller (
        .clk(clk),
        .rst(rst),
        .floor_cur(floor_cur),
        .buttons_in(buttons_in),
        .buttons_ex(buttons_ex),
        .direction(direction),
        .motor(motor)
    );
```

در این بخش ما چند متغیر داخلی که همان ورودی های دو ما/زول قبلی اند را ساخته و یک instance از هر کدام آن ها میگیریم.

```
    always #5 clk = ~clk; // clock

    initial begin
        buttons_in = 0;
        buttons_ex = 0;
        clk = 0;
        rst = 1;


        #10 rst = 0;
```

در اینجا نیز ابتدا کلاک را هر 5 ثانیه یکبار تاگل میکنیم و در 10 ثانیه اول ریست میکنیم.

```
74
75      initial begin
76      $dumpfile("BalaBar.vcd");
77      $dumpvars;
78      end
79
80      reg [1:0] pre [4:0];
81
82      always @(posedge clk) begin
83          if (floor_state !== pre) begin
84              $display("at %4d moment", $time, " Floor state have change:\n this moment is :  %p", floor_state);
85          end
86          pre <= floor_state;
87      end
88
89  endmodule
```

این بخش هم ساز و کاری است برای نمایش خروجی و به محض تغییر وضعیت طبقات وضعیت هر طبقه را پرینت میکند.

حال به سراغ سناریوی جالبمان میرویم.

```
43          #10 buttons_in[4] = 1'b1;
44          #10 buttons_in[4] = 1'b0;
45          #100
46
47
48          #10 buttons_ex[0] = 1'b1;
49          #10 buttons_ex[0] = 1'b0;
50
51
52          #10 buttons_ex[1] = 1'b1;
53          #10 buttons_ex[1] = 1'b0;
54
55          #60 buttons_in[3] = 1'b1;
56          #10 buttons_in[3] = 1'b0;
57
58          #310 buttons_in[2] = 1'b1;
59          #10 buttons_in[2] = 1'b0;
60
61          #130 buttons_in[3] = 1'b1;
62          #10 buttons_in[3] = 1'b0;
63
64          #20 buttons_ex[1] = 1'b1;
65          #10 buttons_ex[1] = 1'b0;
66
67          #20 buttons_ex[4] = 1'b1;
68          #10 buttons_ex[4] = 1'b0;
69
70          #700;
71          $stop();
72      end
73
```

در خط 43 کلید طبقه 4 میخورد.

مشاهده میشود که بالابر پس ا زتوقف در طبقه 1 (به دلیل ریست) مستقیم به طبقه 4 میرود و در آنجا توقف میکند:

```
# at    5 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 0, 0}
# at   25 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 0, 3}
# at  125 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 2, 1}
# at  135 moment Floor state have change:
#  this moment is :  '{0, 0, 2, 1, 0}
# at  145 moment Floor state have change:
#  this moment is :  '{0, 2, 1, 0, 0}
# at  155 moment Floor state have change:
#  this moment is :  '{2, 1, 0, 0, 0}
# at  165 moment Floor state have change:
#  this moment is :  '{3, 0, 0, 0, 0}
```

سپس در حین همین حرکت در خطوط 48 الی 56 مشاهده میشود که کلبد طبقات 3 و 1 و همکف میخورد. با اینکه اول کلید همکف میخورد اما چون طبقات دیگر در مسیر هستند پس در آن جا نیز توقف میکند.

```
# at  165 moment Floor state have change:
#  this moment is :  '{3, 0, 0, 0, 0}
# at  265 moment Floor state have change:
#  this moment is :  '{1, 2, 0, 0, 0}
# at  275 moment Floor state have change:
#  this moment is :  '{0, 3, 0, 0, 0}
# at  375 moment Floor state have change:
#  this moment is :  '{0, 1, 2, 0, 0}
# at  385 moment Floor state have change:
#  this moment is :  '{0, 0, 1, 2, 0}
# at  395 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 3, 0}
# at  495 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 1, 2}
# at  505 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 0, 3}
```

در خط 58 درخواست طبقه 2 می آید که در حینش در خواست طبقه 3.

بهد از آن هم درخواست طبقه همکف و بعد از آن درخواست طبقه 4. اما مشاهده میشود که بالابر بعد از رفن به طبقه دو و بعد ار آن به سه، مستقیم سراغ نمیرود و اولویت دارد که کدام زود تر زده و به یک میرود و سپس بعد از آن به طبقه 4 میرود.

```
# at  505 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 0, 3}
# at  605 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 2, 1}
# at  615 moment Floor state have change:
#  this moment is :  '{0, 0, 2, 1, 0}
# at  625 moment Floor state have change:
#  this moment is :  '{0, 0, 3, 0, 0}
# at  725 moment Floor state have change:
#  this moment is :  '{0, 2, 1, 0, 0}
# at  735 moment Floor state have change:
#  this moment is :  '{0, 3, 0, 0, 0}
# at  835 moment Floor state have change:
#  this moment is :  '{0, 1, 2, 0, 0}
# at  845 moment Floor state have change:
#  this moment is :  '{0, 0, 1, 2, 0}
# at  855 moment Floor state have change:
#  this moment is :  '{0, 0, 0, 3, 0}
# at  955 moment Floor state have change:
#  this moment is :  '{0, 0, 2, 1, 0}
# at  965 moment Floor state have change:
#  this moment is :  '{0, 2, 1, 0, 0}
# at  975 moment Floor state have change:
#  this moment is :  '{2, 1, 0, 0, 0}
# at  985 moment Floor state have change:
#  this moment is :  '{3, 0, 0, 0, 0}
```

مشاهده شد که در تست تمامی حالات اعم از وسط راه وایسادن، اولویت در نظر داشتن، منطق درست، عدم تخطی از 5 طبقه مجاز و... به درستی عمل میکردن.

همچنین فایل vcd نیز موجود است و عکس از موج ها و تمامی عکس ها در فایل image مستند شده اند.