محمدجواد خوشدونی فراهانی ۴۰۱۱۰۵۸۹۴

۵- در این سوال خواسته شده تا مدار آسانسوری را پیاده سازی کنیم. برای اینکار یک ماژول می سازیم و ورودی های مورد نیاز را به ان می دهیم. در این سوال ورودی های ما سیگنال های درخواست اسانسور در طبقات و اسانسور هستند. از انجایی که فرقی بین ان ها وجود ندارد فقط یک سیگنال برای هر طبقه در نظر می گیریم(می توانستیم ۲ سیگنال بگیریم و آن دو را or کنیم و طبق ان عمل کنیم) و یک سری رجیستر درون ماژول نیاز داریم که در ادامه بررسی می کنیم.

```
1   module elevator(input wire [4:0] buttons , reg clk);
2
3       reg [2:0] dst, current_floor = 3'b000;
4       reg [2:0] pending1 , pending2, pending3;
5       reg [2:0] i = 0;
6       reg direction; //up = 1, down = 0
7       reg movement; //move = 1 , stop = 0
8       reg [4:0] in_sensor = 5'b00000;
9       reg [4:0] out_sensor = 5'b00000;
10
```

Buttons همان سیگنال های درخواست یا دکمه های طبقات هستند که برای راحتی کار ان ها را یک ارایه در نظر می گیریم هر بیت معادل طبقه نظیرش هست. یعنی بیت ۰ ام همکف، بیت اول طبقه اول و ...

clk کلاک می باشد که آسانسور با هر posedge آن در طبقات جابه جا می شود.

dst این رجیستر به طبقه مقصد اشاره دارد که طبق الگوریتمی که آن را بررسی خواهیم کرد تعیین می شود و اسانسور به سمت آن حرکت می کند.

current_floor به طبقه ای که الان هستیم اشاره می کند. از آنجایی که در صورت سوال اشاره ای نکرده ابتدا طبقه همکف در نظر گرفتم

Pending1..3 این رجیستر ها برای زمانی هستند که اسانسور مقصد تعیین شده ای داره ولی درخواست های جدیدی از طبقات دیگر می اید و این رجیستر ها این طبقات را به ترتیب زمان نگه می دارند.

i شمارنده لوپ است.

direction جهت حرکت اسانسور.

movement اگر ۱ باشد اسانسور در حال حرکت و اگر ۰ باشد اسانسور ایستاده است.

in_sensor, out_sensor حسگر های هر طبقه هستند که ورود و خروج به طبقات را نشان می دهد.

```
always @(buttons)begin
    for(i = 0;i < 5; i = i +1)begin
        if(buttons[i] == 1 && (dst !== i) && (current_floor != i)
            && (pending1 !== i) && (pending2 !== i))begin
            if(dst === 3'bx) begin
                dst = i;
                if($signed(dst - current_floor) > 0)
                    direction = 1;
                else
                    direction = 0;
            end
            else begin
                if(direction == 1 && ((current_floor > i ) || (i > dst)))begin
                    if(pending1 === 3'bx) begin
                        pending1 = i;
                    end
                    else begin
                        if(pending2 === 3'bx)
                            pending2 = i;
                        else
                            pending3 = i;
                    end
                end
                else if(direction == 0 && ((dst > i ) || (i > current_floor)))begin
                    if(pending1 === 3'bx)
                        pending1 = i;
                    else begin
                        if(pending2 === 3'bx)
                            pending2 = i;
                        else
                            pending3 = i;
                    end
                end
            end
        end
    end
end
```

این ماژول دارای always۲ است یکی بر اساس سیگنال های درخواست و دیگری بر اساس کلاک برای حرکت در طبقات. اولی را بررسی می کنیم:

با تغییر ورودی میخواهیم به بیتی که تغییر کرده است اگر ۱ شده نوبت مناسبی بدهیم. یعنی طبق اولویت ها و حرکت اسانسور ان را مقصدی قرار بدهیم که باید توقف کند.if خط ۶۴ به این دلیل است که بیت هایی که قبلا بررسی شده اند دیگر بررسی نشوند. اگر ما در حال حاضر مقصدی نداشته باشیم آن طبقه مقصد جدید می شود و طبق اختلاف آن و طبقه الانمان مشخص می شود که اسانسور باید بالا برود یا پایین. حالا اگر مقصدی داشتیم از قبل این بیت را بررسی می کنیم که اگر در مسیر الان اسانسور قرار داشت کاری نکنیم چرا که طبق سوال (مثال طبقه ۴ بین ۲ و ۵) باید بایستد و لازم نیست اولویت زمانی آن را نسبت به بقیه بسنجیم. در غیر این صورت باید آن را به رجیستر های pending ببریم و اولین pending که خالی بود ان را انجا قرار دهیم.

حالا سراغ always دیگر می رویم:

```
always @(posedge clk) begin
    if(dst === 3'bxxx)
        movement = 0;

    if(direction == 1 && movement == 1)begin
        in_sensor[current_floor] = 0;
        out_sensor[current_floor] = 1;
        out_sensor[current_floor - 1] = 0;
        current_floor = current_floor + 1;
        in_sensor[current_floor] = 1;
        out_sensor[current_floor] = 0;
    end
    else if(direction == 0 && movement == 1) begin
        in_sensor[current_floor] = 0;
        out_sensor[current_floor] = 1;
        out_sensor[current_floor + 1] = 0;
        current_floor = current_floor - 1;
        in_sensor[current_floor] = 1;
        out_sensor[current_floor] = 0;
    end

    if(buttons[current_floor] == 1 && movement == 1)begin
        movement = 0;
        out_sensor[current_floor] = 1;
        in_sensor[current_floor] = 1;
        #100;
        if(current_floor == dst)begin
            dst = pending1;
            pending1 = pending2;
            pending2 = pending3;
            pending3 = 3'bx;    //set new destination and change the waiting list
            if($signed(dst - current_floor) > 0)
                direction = 1;
            else
                direction = 0;  //set new direction base on new dest
        end
        else if(current_floor == pending1)begin //delete pending1
            pending1 = pending2;
            pending2 = pending3;
            pending3 = 3'bx;
        end
        else if(current_floor == pending2) begin //delete pending2
            pending2 = pending3;
            pending3 = 3'bx;
        end
    end
    else if(dst !==  3'bxxx) begin
        movement = 1;
    end
end
```

(تغییر های جزیی در کد اپلود شده نسبت به عکس های بالا داده شده است)

ابتدا مشخص شده است تا زمانی که مقصدی نداریم(مقصد اولیه در always قبلی مشخص می شد) اسانسور حرکت نکند. سپس می خواهیم در هر کلاک به طبقه جدید بریم و current_floor خود را اپدیت کنیم. این کار را با رجیستر direction انجام می دهیم اگر به بالا می رفت یکی افزایش می دهیم و اگر به پایین می رفت یکی کاهش می دهیم. در همین حین حسگر ها را هم اصلاح می کنیم. اگر از طبقه ای در حال خارج شدن هستیم حسگر out ان را ۱ و حسگر in ان را ۰ می کنیم و از انجایی که در کلاک قبل حسگر out ۲ طبقه قبل ۱ بوده و دیگر کاری به ان طبقه نداریم ان را ۰ می کنیم و طبقه ای که به ان وارد می شویم را نیز حسگر in مورد نظر را ۱ می کنیم. در مرحله بعد بررسی می کنیم که این طبقه درخواستی داشته است یا خیر. اگر درخواستی داشته است به مدت ۱۰۰ واحد زمان اسانسور را متوقف می کنیم و هردو حسگر ان طبقه ۱ می شوند. بعد از آن ۱۰۰ واحد تاخیر اگر نیاز به مقصد جدیدی داریم باید از لیست انتظار خود یک مقصد جدید با

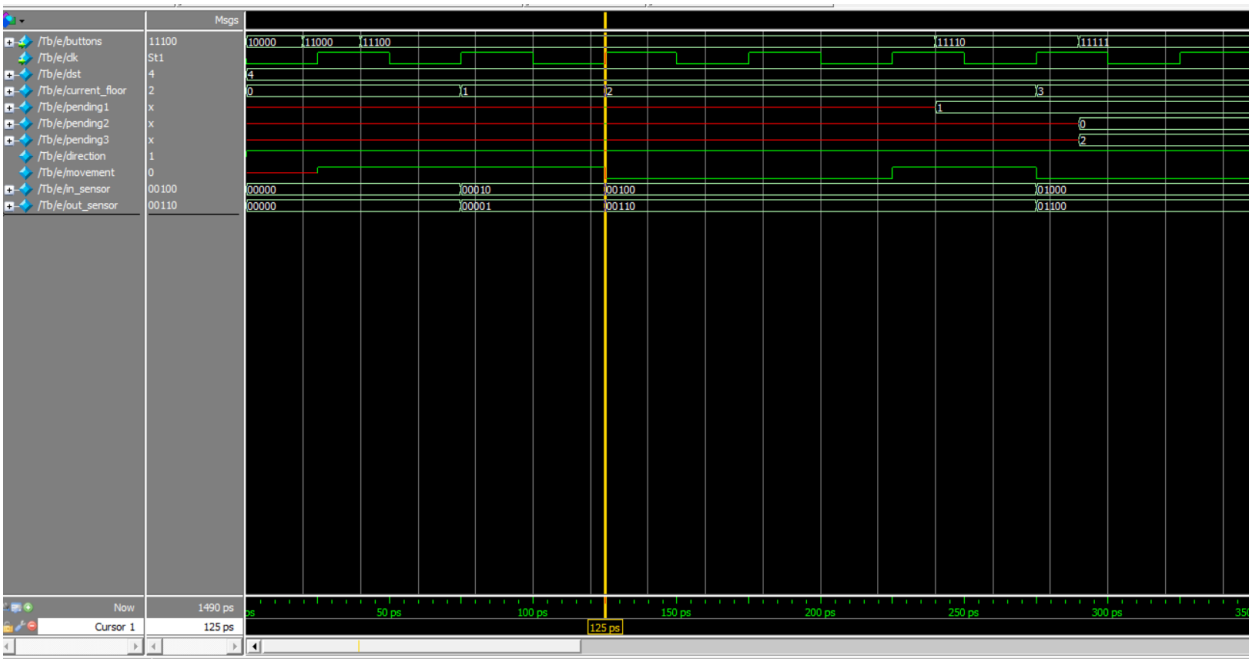
اولویت زمانی بالاتر(pending1) برداریم و به dst بدهیم و لیست انتظار خود را نیز اصلاح کنیم در ادامه جهت جدید جرکت اسانسور را بررسی می کنیم . حالا شروع به تست کردن برنامه می کنیم:

```
module Tb();

reg [4:0] buttons;
reg clk;
elevator e (buttons , clk);

initial clk = 0;
always #25 clk = ~clk;

initial begin
    //****test1****
    buttons[0] = 0;
    buttons[1] = 0;
    buttons[2] = 0;
    buttons[3] = 0;
    buttons[4] = 1;
    #20
    buttons[0] = 0;
    buttons[1] = 0;
    buttons[2] = 0;
    buttons[3] = 1;
    buttons[4] = 1;
    #20
    buttons[0] = 0;
    buttons[1] = 0;
    buttons[2] = 1;
    buttons[3] = 1;
    buttons[4] = 1;
    #200
    buttons[1] = 1;
    #50
    buttons[0] = 1;


    #1200
    $stop;
end
```

همانطور که مشاهده می شود ابتدا طبقه ۴ درخواست می شود و سپس بعد ۲۰ واحد طبقه ۳ هم درخواست می کند و به همین ترتیب واحد ۲. اسانسور باید به گونه ای عمل کند که وقتی به سمت طبقه ۴ می رود در این مسیر در این ۲ طبقه هم بایستد. سپس بعد از ۲۴۰ واحد زمانی دکمه طبقه۱ نیز زده می شود و بعد از همکف نیز زده می شود. طبق اولویت زمانی ابتدا باید به طبقه ۱ برود و سپس به همکف و ازانجایی که درخواست طبقات ۲و ۳ از قبل روشن بوده در این مسیر باید در این طبقات نیز توقف کند حالا خروجی مانیتور current_floor را ببینیم:

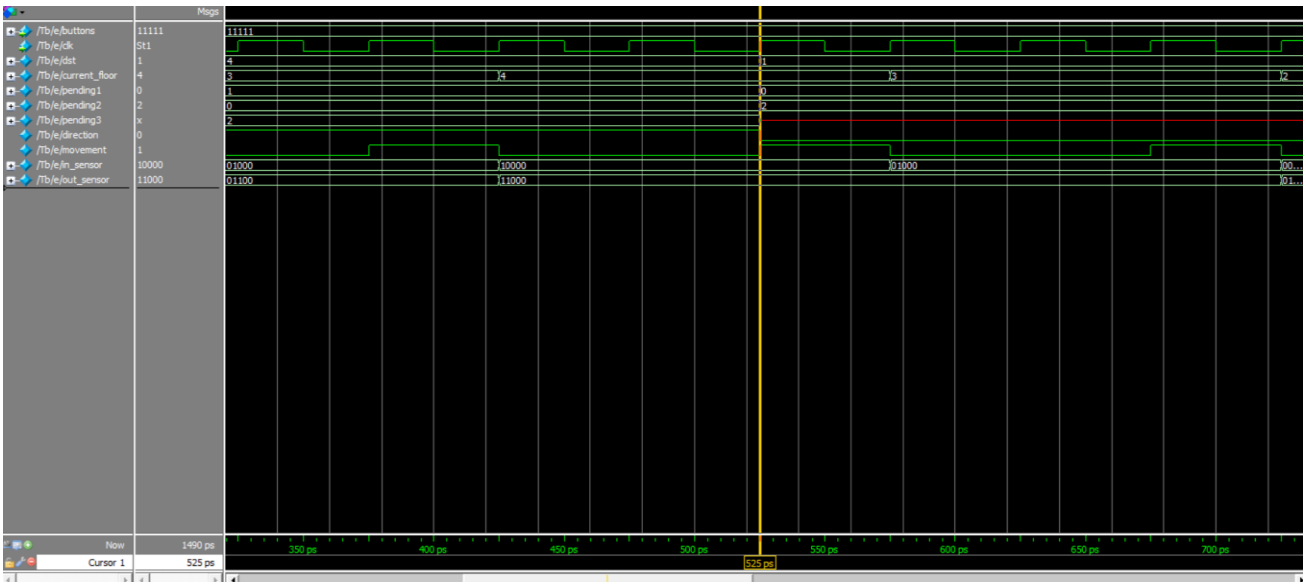
```
 25 current floor is: 0
 75 current floor is: 1
125 current floor is: 2
225 current floor is: 2
275 current floor is: 3
375 current floor is: 3
425 current floor is: 4
525 current floor is: 4
575 current floor is: 3
675 current floor is: 3
725 current floor is: 2
825 current floor is: 2
875 current floor is: 1
975 current floor is: 1
1025 current floor is: 0
1125 current floor is: 0
1175 current floor is: 0
1225 current floor is: 0
1275 current floor is: 0
1325 current floor is: 0
1375 current floor is: 0
1425 current floor is: 0
1475 current floor is: 0
```

ابتدا در همکف هستیم سپس به طبقه ۱ می رود. سپس به طبقه ۲ و طبق انتظار در انجا می ایستد در طبقه ۳ و ۴ نیز همینطور است .پس از طبقه ۴ مقصد جدیدی برای ان تعریف می شود و ان طبقه ۱ است. در مسیر ۴ به ۱ در طبقات ۲ و ۳ نیز توقف می کندو پس از ۱ مقصد جدید ما طبقه همکف است و بعد از ان دیگر درخواستی نداریم و در انجا می ماند. حالا waveان را بررسی کنیم تا سیگنال و حسگر ها را هم بررسی کنیم:

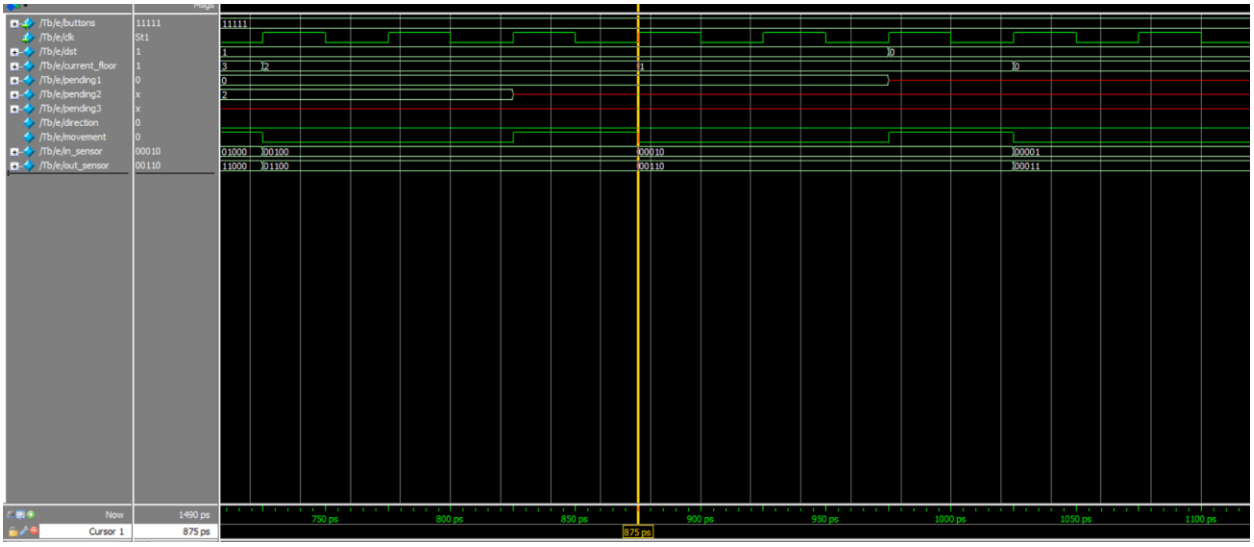


همانطور که مشاهده می شود اسانسور ابتدا dst طبقه ۴ قرار می دهد و در اولین لبه بالارونده کلاک شروع به حرکت در جهت بالا می کند(movement = 1 , direction= 1). در کلاک بعد اسانسور یک طبقه بالا می رود و به طبقه ۱ می رسد در نتیجه حسگر ورود طبقه۱ و حسگر خروج همکف ۱ می شود.در کلاک بعد به طبقه ۲ می رسیم. طبق انتظار باید در این طبقه توقف کنیم. به همین دلیل movement به مدت ۱۰۰ واحد ۰ می شود (از ۱۲۵ تا ۲۲۵) و همچنین هردو حسگر ورود و خروج این طبقه ۱ می شوند. بعد از ان مدت دوباره شروع به

حرکت می کند movement =1 می شود تا به طبقه ۳ برسیم و مراحل قبل تکرار می شود. اتفاق جدیدی که در این بین میافتد ، درخواست طبقه ۱ روشن می شود و از انجایی که ما یک dstداریم این درخواست به لیست انتظار می رود تا زمانی به مقصد خود برسیم . بعد از ان دکمه درخواست همکف هم زده می شود که به لیست انتظار می رود تا بعد از طبقه ۱ بررسی شود.(از انجایی که طبقه ۲ هنوز روشن است ان هم به لیست انتظار می رود).



در لحظه ۴۲۵ به طبقه ۴ می رسد و پس از ۱۰۰ واحد توقف dst جدید را از pending1 می گیرد و این لیست را اپدیت می کند. حالا اسانسور direction را عوض می کند و به پایین حرکت می کند همانطور که مشاهده می شود در طبقه ۳ توقف می کند چرا که درخواست ان هنوز روشن است و در طبقه ۲ نیز به همین شکل است.



پس از اینکه به ۲ رسید ان از لیست انتظار حذف می شود و به طبقه ۱ می رویم و بعد از توقف در انجا dst جدید همکف می شود و پس از یه کلاک به همکف می رسیم و در انجا نیز توقف می کنیم از انجایی که درخواست جدیدی صورت نگرفته movement۰ می ماند و همانجا ثابت میمانیم.

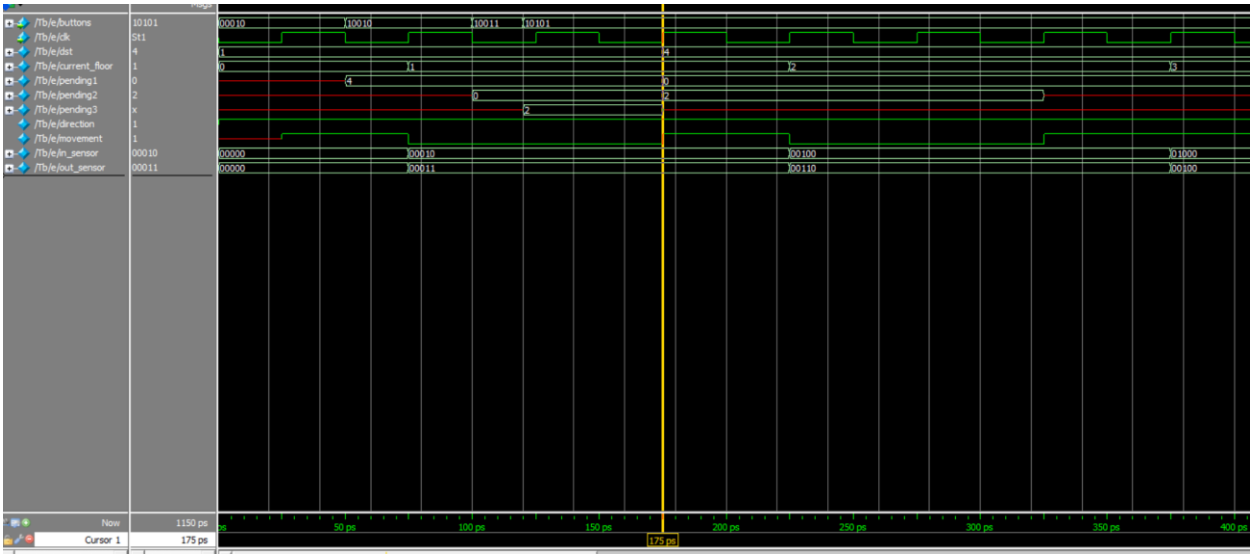
یک تست دیگر را بررسی می کنیم:

```
37      //****test2****
38      buttons[0] = 0;
39      buttons[1] = 1;
40      buttons[2] = 0;
41      buttons[3] = 0;
42      buttons[4] = 0;
43      #50
44      buttons[4] = 1;
45      #50
46      buttons[0] = 1;
47      #20
48      buttons[1] = 0;
49      buttons[2] = 1;
50      #830
51      buttons[1] = 1;
52      buttons[2] = 0;
53      buttons[4] = 0;
54      #200
55      $stop;
56
57  end
58
59  endmodule
60
```

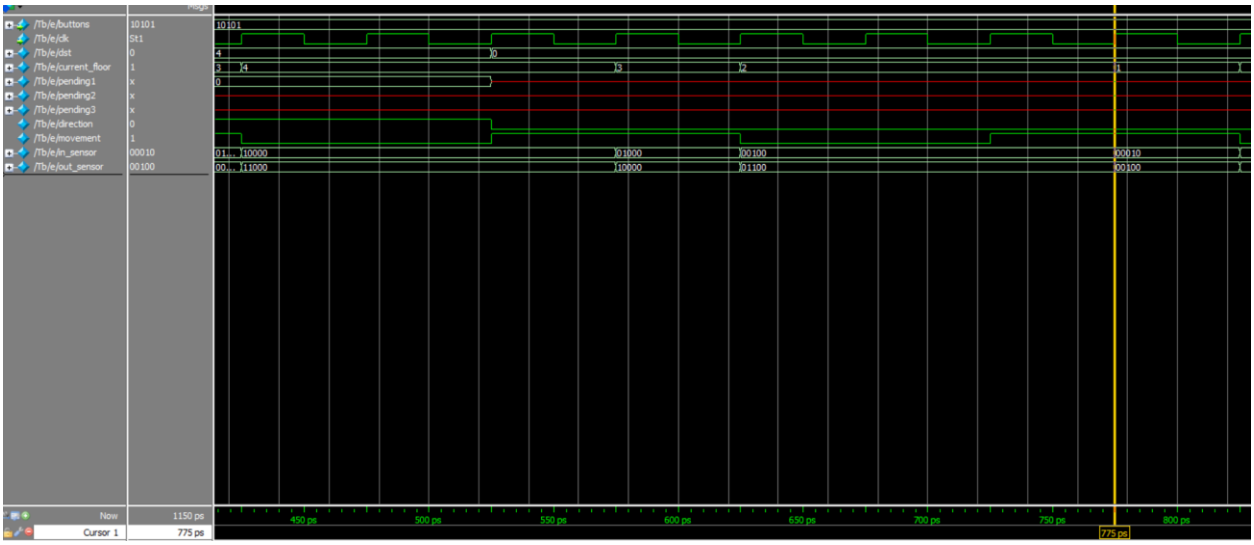
ابتدا درخواست طبقه ۱ روشن می شود و پس از ان طبقه ۴ و سپس همکف. از انجایی که اولویت ما زمان است به ترتیب به ۱ ۴ و همکف باید برویم . در این بین طبقه ۲ روشن می شود و درخواست طبقه ۱ قطع می شود(ازانجایی که دکمه کنسل در اسانسور نداریم این اتفاق بعد از رسیدن به طبقه ۱ رخ می دهد) تا دفعات بعدی که از ان عبور می کنیم توقف نکنیم. سپس یک حالتی در نظر گرفتیم برای زمانی که اسانسور برای یه مدتی درخواستی نداشته است تا ببینیم ایا دوباره کار می کند یا نه. پس از مدت زیاد (که مطمئنیم درخواست های قبل انجام شده است) درخواست طبقه ۱ روشن می شود تا اسانسور به ان برود.

```
  25 current floor is: 0
  75 current floor is: 1
 175 current floor is: 1
 225 current floor is: 2
 325 current floor is: 2
 375 current floor is: 3
 425 current floor is: 4
 525 current floor is: 4
 575 current floor is: 3
 625 current floor is: 2
 725 current floor is: 2
 775 current floor is: 1
 825 current floor is: 0
 925 current floor is: 0
 975 current floor is: 0
1025 current floor is: 1
1125 current floor is: 1
```

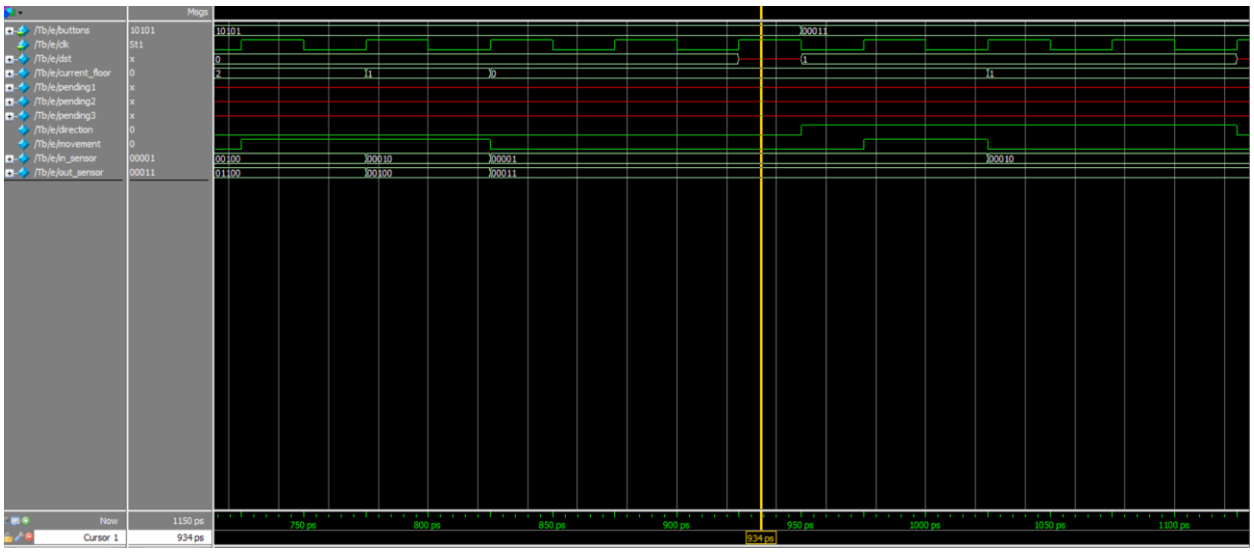
در طبقه ۱ توقف می کند و سپس به سمت ۴ حرکت می کند در این مسیر در ۲ نیز توقف می کند . بعد از این به ۴ رسید مقصد جدید همکف است و در این مسیر در ۲ توقف می کند ولی در ۱ دیگر توقف نمی کند و به همکف می رسد و انقد توقف می کند تا درخواست جدید صورت بگیرد. پس از اینکه طبقه اول ۱ می شود به ان می رود و در ان متوقف می شود . حالا wave ان را بررسی می کنیم:



به ۱ می رود و در ان متوقف می شود .pending1 برابر ۴ می شود یعنی مقصد بعدی باید ۴ باشد در همین حین ۲ درخواست جدید نیز می اید و ان ها را به ترتیب زمان به pending2و pending3 می دهیم. حالا پس از توقف ۱۰۰ واحد زمانی دوباره شروع به حرکت می کند و ۴ dst جدید می شود و لیست انتظار را نیز اصلاح می کنیم. در این مسیر در ۲ توقف می کنیم و باید ان را از لیست انتظار حذف کنیم.



پس از اینکه به ۴ رسیدیم direction به ۰ تبدیل می شود و اسانسور رو به پایین حرکت می کند و مقصد جدید همکف می شود در این مسیر در ۲ توقف می کنیم ولی دیگر در ۱ توقف نمی کنیم.



حالا که به ۰ رسیدیم در انجا متوقف می شویم تا درخواست جدید بیاید.(dst در زمان ۹۲۵ تا ۹۵۰ مقدار x دارد) و به محض درخواست جدید مقصد جدید تعیین می شود و شروع به حرکت می کنیم تا به ان برسیم و متوقف شویم.