به نام خدا

گزارش پروژه درس مدارهای منطقی و آزمایشگاه

استاد:دکتر محمدزاده

سید محمد امین منصوری طهرانی
شماره دانشجویی:۹۴۱۰۵۱۷۴

موضوع: بازی سرعت عمل

تاریخ تحویل پروژه: ۱۰ بهمن

سوال:

ابتدا بازیکن عددی را بین 0 و 9 وارد میکند.

یک دکمه داریم . به محض فشار دادن آن توسط کاربر ،اعدادی با سرعت پایین روی 7segment شروع به چرخش بین 0 تا 9 می کنند و وقتی کاربر دستش را از روی دکمه برداشت می ایستند . اگر عددی که روی آن ایستاده با عدد وارد شده برابر بود کاربر 1 امتیاز میگیرد درغیر اینصورت هیچ.

سپس اگر کاربر درست وارد کرده بود وارد مرحله بعدی می شود به این صورت که سرعت چرخش اعداد روی 7segmentدو برابر می شود و این روند ادامه می‌یابد تا جایی کاربر بسوزد (یعنی در مرحله ای عدد روی عدد وارد شده نایستد).

در ضمن در هر مرحله امتیاز درست وارد کردن 2 برابر مرحله قبل است.

فرکانس ورودی در اختیار خود شماست.

در آخر وقتی کاربر سوخت باید امتیاز روی 7segment نمایش داده شود.

کد پاسخ:

```
module finalproject(in,clk,push,reset,seg,d1,d2,d3,d4);
input clk,push,reset;
input [3:0]in;
reg [3:0]num;
reg [6:0] point;
output reg [7:1]seg;
output reg d1,d2,d3,d4;
reg [3:0]level,q;
reg ud;
reg start=1'b0;
reg[15:0]cnt;
integer k;
reg flag;
reg flag1;
reg [3:0]second1;
reg [3:0]second2;
```

```
always@ (posedge clk or posedge reset)
          begin
               if(reset)
                         begin
                              point<=4'b0; num<=4'b1111;
                              ud<=1; k<=0; start<=1;
                              flag<=0; level<=1; flag1<=0;
                         end
               else
                         begin
               if(start)
               begin
               if(push)
               begin
                    flag1<=1;
                    if(flag1==0)
                         begin
               num<=4'b0000;//avoiding 0 selection for
               input number to be distinguished by the
               startng number

                         end

                    if(k<((10_000_000)/(2**(level-1))))
                    k<=k+1;

               else

               begin

               k<=0; flag<=0;
               if(ud)
               begin
               if(num<8)num<=num+1;
```

```
          else
          begin num<=num+1; ud<=0; end
          end

          else
          begin if(num>1)
          num<=num-1;
          else
          begin num<=num-1; ud<=1;
          end
          end


          end

end

else//if ~push

begin

                              if(in==num)

                                   begin


     if(flag==1)

begin

point<=point; level<=level;

end


     else


          begin
```

```
if(level==1)

begin


point<=1; level<=level+1; flag<=1;


          end


     else


begin


point<=((2**(level-1)+point));
level<=(level+1); flag<=1;


          end


end

end

else

     begin
          level <=level;
          1point<=point;
          num<=4'b1111;
          end
```

```
     end
end
else
begin
     level<=4'b0001;
end
                    end
               end

always@(posedge clk )
          begin
                    cnt<=cnt+1;
                    if(point<10)
                         begin
                                        second2<=4'b1111;
                                        second1<=point;
                         end
                    if(point>9)
                         begin
                                        second2<=point/10;
                                        second1<=point%10;
                         end
                    case(cnt[15:14])
0 : begin d1<=0; d2<=1; d3<=1; d4<=1; q<=level; end
1 : begin d1<=1; d2<=0; d3<=1; d4<=1; q<=second2; end
2 : begin d1<=1; d2<=1; d3<=0; d4<=1; q<=second1; end
3 : begin d1<=1; d2<=1; d3<=1; d4<=0; q<=num; end
default : begin d1<=1; d2<=1; d3<=1; d4<=1; end
               endcase
```

```
            case(q)
                4'b0000 : seg<=7'b1111110;
                4'b0001 : seg<=7'b0110000;
                4'b0010 : seg<=7'b1101101;
                4'b0011 : seg<=7'b1111001;
                4'b0100 : seg<=7'b0110011;
                4'b0101 : seg<=7'b1011011;
                4'b0110 : seg<=7'b0011111;
                4'b0111 : seg<=7'b1110000;
                4'b1000 : seg<=7'b1111111;
                4'b1001 : seg<=7'b1110011;
                4'b1111 : seg<=7'b0000001;
                default : seg<=7'b0;
            endcase

        end
endmodule
```

توضیحات مشکلات پیش آمده همراه با راه حل و توضیح کد:

مشکلات ایجاد شده به نوشتن کدهایی منجر شده که در ادامه آورده شده است. بنابراین این توضیحات همان توضیحات اشکالات به وجود آمده و راه حل رفع آن است.

ابتدا ورودی ها و خروجی ها مشخص می‌شوند.in همان عدد وارد شده توسط کاربر می‌باشد.این عدد را ۴ بیتی انتخاب می‌کنیم.clk همان کلاک خود بورد FPGA است که فرکانسی برابر با $10MHz$ دارد.push کلیدی است که در صورت شروع شدن بازی با نگه داشتن آن توسط کاربر اعداد روی بورد از ۰ تا ۹ تغییر می‌کنند و با رهاکردن کلید تغییر متوقف می‌شود.
reset همان ریست آسنکرون است. در مدار ما دو کاربرد دارد. کاربرد اول به منظور پایان دادن به بازی در هر مرحله ای و هر زمان از بازی است. کاربرد دوم زمانی است که کاربر باخته است و قصد شروع مجدد بازی را دارد. برای این منظور می بایست ابتدا کلید reset را فشار دهد تا همه امتیازات و مراحل به مقدار اولیه برگردند. سپس مجددا با استفاده از کلید push می تواند بازی را ادامه دهد.
seg خروجی ۷ بیتی است که به پایانه های 7-segment مربوط است.
d1,d2,d3,d4 نیز هرکدام به یکی از 7-segment های بورد مربوط اند و با آنها مشخص می‌کنیم هر کدام از 7-segment ها کدام متغیر را نشان دهند.

سایر متغیرها و رجیسترها:

num: این متغیر ۳ بیتی برای نمایش عددی است که بین ۰ تا ۹ می‌چرخد.
point: امتیاز کاربر می‌باشد که در هر مرحله و همه زمان ها نشان داده می‌شود. با توجه به اینکه می‌خواستیم اعداد دورقمی را نیز نشان دهد و ماکزیمم عدد دورقمی که شخص می‌تواند بگیرد ۶۳ است این رجیستر را ۶ بیتی تعریف کردیم. لازم به ذکر است که چون در هر مرحله هم امتیاز و مرحله و هم عدد متغیر روی بورد نمایش داده می‌شود نهایتا قادر بودیم اعداد دورقمی را نشان دهم. هم چنین خارج از خواسته مساله که امتیاز و مرحله در زمان باخت خواسته شده است این کد امتیاز و مرحله را در همه زمان ها نشان می‌دهد.

level: این متغیر با توجه به تعداد مراحل ۴ بیتی تعریف شد.
q: متغیری است که برای تخصیص متغیرها به 7-segment ها استفاده می‌شود.
ud: این متغیر به شمارش بالا-پایین یا بالعکس مربوط است. اضافه بر خواسته سوال به جای رفتن به ۹ و شروع از صفر این کد از صفر به ۹ رفته و سپس به جای رفتن مستقیم به صفر از ۹ به صفر بازمی‌گردد.
cnt: این رجیستر ۱۶ بیتی برای نشان دادن مناسب اعداد خواسته شده بر روی 7-segment ها است. در واقع برای نمایش با نور کافی فرکانس ۱۰۰ هرتز مناسب بوده و در ادامه روند توضیح داده خواهد شد. به این ترتیب اعداد با فرکانس ۱۰۰ هرتز نمایش داده خواهند شد.

K: متغیری است از نوع integer که نقش شمارنده را دارد. چون کلاک از خود FPGA گرفته شده و ما هر مرحله کلاک متفاوتی داریم با این عدد فرکانس مورد نظر خود را تنظیم می‌کنیم.

flag,flag1: این دو متغیر تک بیتی در ادامه توضیح داده خواهند شد.
second1,second2:این دو متغیر برای نشان دادن اعداد دو رقمی استفاده شده اند.

بلاک always اول:
در لیست حساسیت این بلاک لبه بالارونده کلاک FPGA و لبه بالارونده reset استفاده می‌شود.قسمت دوم برای ریست آسنکرون است که گفته شد.
پس در هرکلاک $10MHz$ وارد این بلاک می‌شود.اگر کلید reset زده شده باشد در شرط اول می‌رود و همه شرایط اولیه را برای بازی تنظیم می‌کند. یعنی امتیاز صفر شده و مرحله ۱ می‌شود. متغیر num به 4'b1111 assign می‌شود تا خط تیره به جای آن نمایش داده شود. یعنی بازی شروع نشده است.ud ۱ شده و از صفر شروع خواهیم کرد به سمت k.۹ متغیر شمارنده ۰ می‌شوند. متغیرهای نشان(flag) نیز صفر می‌شوند.start نیز ۱ می‌شود. یعنی کاربر کلید ریست را زده و اجازه شروع بازی را دارد.
اگر ریست زده نشده باشد چون استارت صفر است وارد بلاک شرطی نمی‌شود.

پس از اینکه ریست زده شد در هرکلاک وارد بلاک شرط استارت می‌شود. اگر کلید push فشرده نشده باشد وارد اولین else مربوط به شرط push شده و چون در ابتدا عدد ورودی با num که 4'b1111 است برابر نیست وارد else مربوط به شرط برابری ورودی و num شده و مرحله همان ۱ و امتیاز صفر می‌ماند.این اتفاق آنقدر ادامه می‌یابد تا کلید push فشرده شود. در این حالت وارد بلاک شرطی push می‌شویم. سپس flag1 یک می‌شود. چون قبل از آن صفر بوده در کلاک قبل وارد اولین if شده و به num مقدار صفر را می‌دهد. از این به بعد در کلاک های بعدی دیگر وارد این قسمت نخواهد شد چون بلاک if با یک شدن flag1 قابل دسترسی نخواهد بود.

دومین بلاک if در بلاک شرطی کلید push:
در این قسمت عملیات دو برابر کردن فرکانس اتفاق می‌افتد. به این صورت که وقتی مرحله اول هستیم پس از هر ۱۰ میلیون کلاک FPGA یا ۱۰ میلیون بار ۱۰۰ نانوثانیه که همان ۱ ثانیه است عدد روی 7-segment عوض می‌شود. سپس در مراحل بعدی با توان های ۲ و مرحله فرکانس ۲ برابر می‌شود. وقتی شمارنده به مقدار مورد نظر برسد در بلاک else عدد نمایشگر عوض می‌شود. با الگوریتم نوشته شده از صفر تا ۹ رفته و در عدد ۸ متغیر ud صفر می‌شود و عدد ۹. سپس در مرحله بعدی عدد کم شده تا به صفر برسد و این روند مرتبا ادامه می یابد تا کاربر کلید push را رها نماید.

بلاک else مربوط به push:

اگر عدد ورودی با عددی که در این لحظه متغیر num دارد برابر نباشد وارد بلاک else شده و متغیر نشان دهنده عدد را به خط تیره تبدیل می‌کند(اتمام بازی) و شماره مرحله باخت و امتیاز نهایی را نشان می‌دهد.
در صورتیکه عدد ورودی با num برابر باشد چون flag صفر است وارد اولین else می‌شود.اگر در مرحله ۱ باشیم امتیاز ۱ شده و مرحله ۲ می‌شود.flag نیز ۱ می‌شود و در کلاک های بعدی امتیاز و مرحله عوض نمی‌شود.

اگر در مرحله ای غیر از ۱ باشیم هم باز امتیاز با فرمول گفته شده در صورت سوال زیاد شده و مرحله یکی بالا می‌رود.متغیر flag برای جلوگیری از اضافه شدن امتیاز در هر مرحله است.

بلاک always دوم:

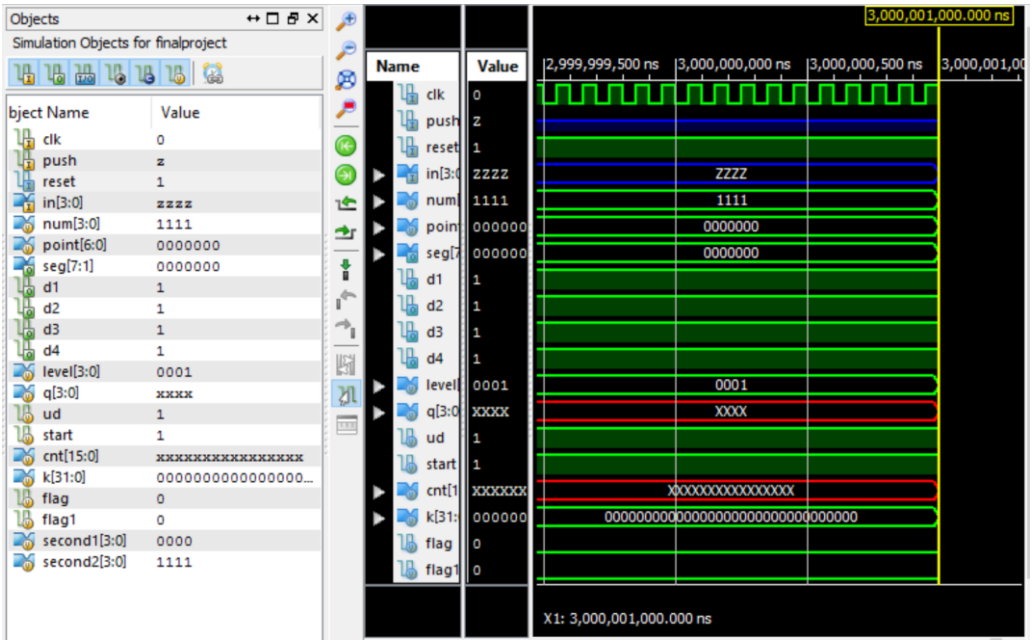
در این جا نمایش اعداد روی بورد تنظیم می‌شود. هر کلاک مقدار cnt یکی افزایش می‌یابد. Case روی دو بیت آخر cnt است. اگر صفر باشند رقم اول روشن شده و مرحله را نمایش می‌دهد.اگر این دو بیت برابر عدد باینری ۱ باشند رقم دوم روشن شده و دهگان امتیاز را نشان می‌دهد. اگر عدد باینری ۲ باشند رقم سوم روشن شده و یکان عدد را نشان خواهد داد. نهایتا اگر باینری ۳ باشد رقم ۴ ام روشن شده و عدد num که بین ۰ تا ۹ می‌چرخد نمایش داده خواهد شد. در واقع به این صورت چون کلاک FPGA خیلی سریع است با این روش این اعداد مرتبا روشن و خاموش شده ولی چشم ما به علت سرعت بالا قادر به تشخیص آن نمی باشد و تقریبا با فرکانس ۱۰۰ هرتز هر رقم خاموش و روشن می‌شود.

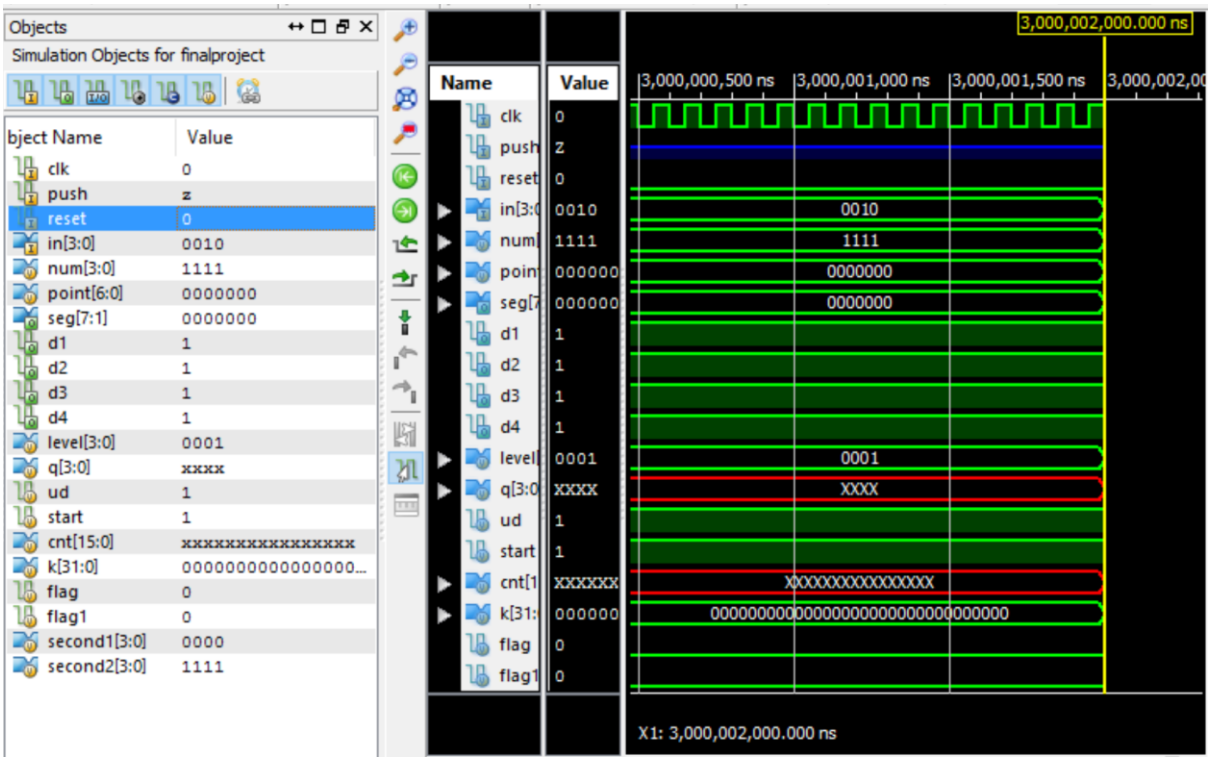
دو بلاک شرطی نیز برای نمایش اعداد دو رقمی اند. به این صورت که تا زمانیکه عدد کمتر از ۱۰ است رقم دهگان خط تیره بوده و رقم یکان نمایش داده می شود و اگر عدد امتیاز از ۱۰ بیشتر باشد رقم یکان باقی مانده تقسیم امتیاز بر ۱۰ و دهگان آن حاصل تقسیم امتیاز بر ۱۰ است.

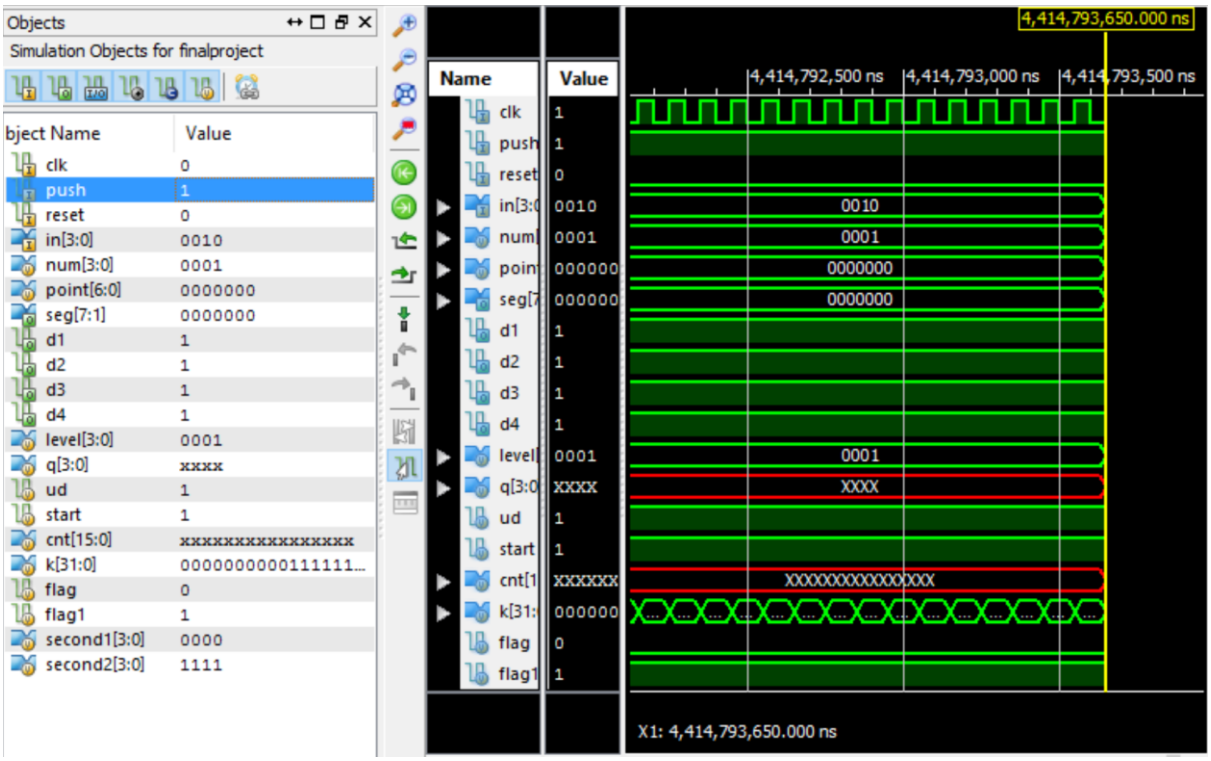
بلاک case: این بلاک به 7-segment مربوط است و مشخص می‌کند برای هر عدد کدام خط های هر 7-segment روشن شود.

ضمنا indentation ها با کپی از وریلاگ مقداری جابجا شده اند.

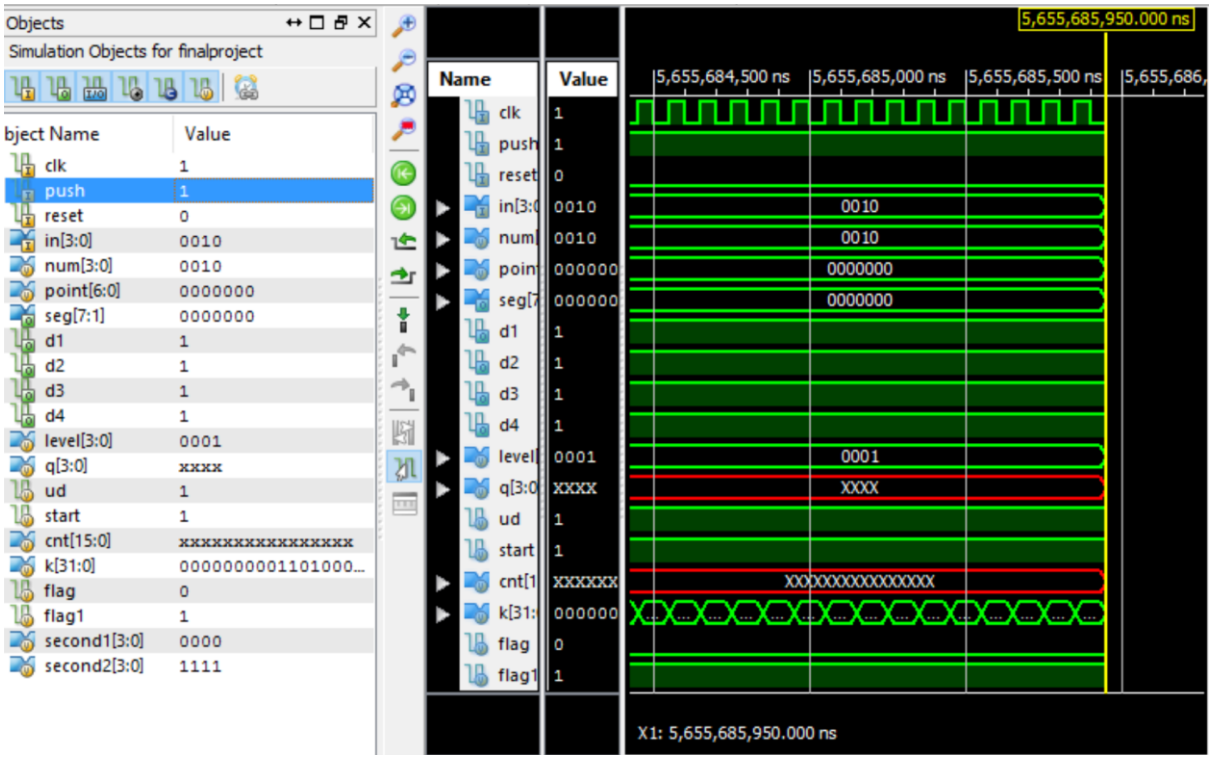
در شکل های زیر شبیه سازی کد را به ازای ورودی ۲ مشاهده می‌کنیم.باید به مقادیر num و in که به ترتیب عدد نمایشگر و عدد ورودی هستند و همینطور point که امتیاز است توجه کنیم.level نیز مرحله را نشان می‌دهد.



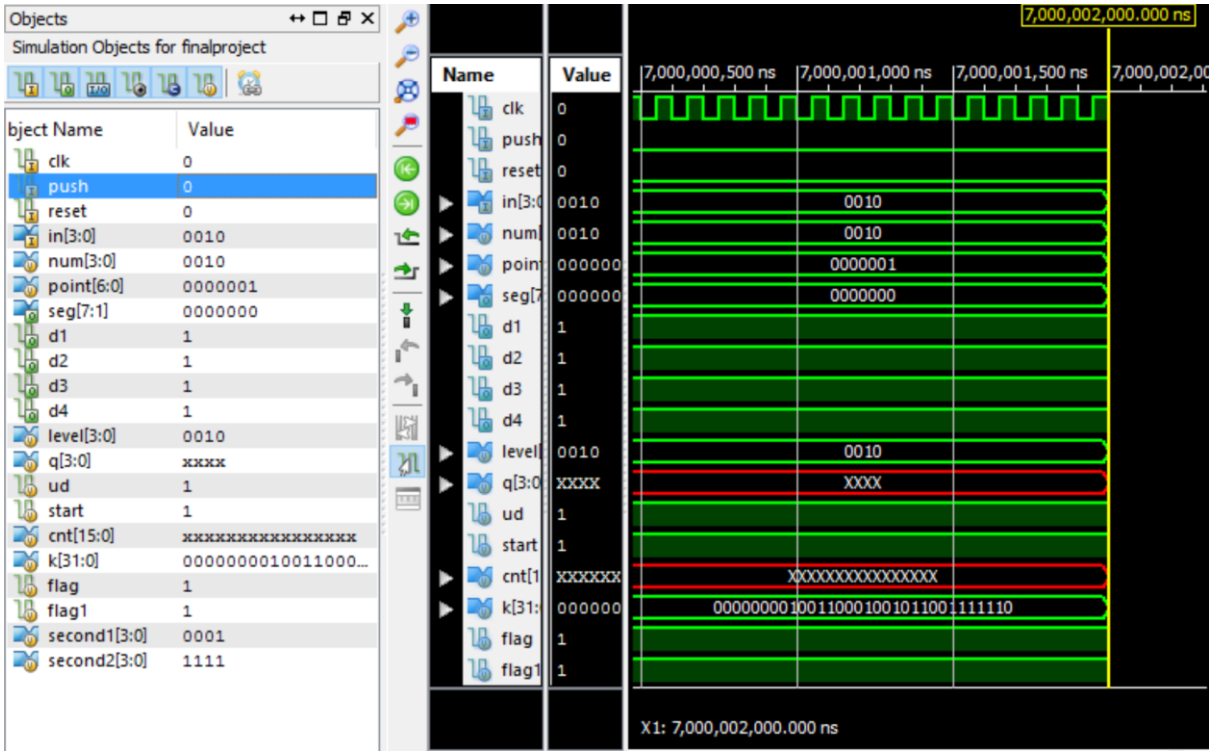

Objects
Simulation Objects for finalproject
bject Name
Value
clk 0
push z
reset 0
in[3:0] 0010
num[3:0] 1111
point[6:0] 0000000
seg[7:1] 0000000
d1 1
d2 1
d3 1
d4 1
level[3:0] 0001
q[3:0] xxxx
ud 1
start 1
cnt[15:0] xxxxxxxxxxxxxxxx
k[31:0] 0000000000000000...
flag 0
flag1 0
second1[3:0] 0000
second2[3:0] 1111
Name
Value
3,000,002,000.000 ns
3,000,000,500 ns
3,000,001,000 ns
3,000,001,500 ns
0010
1111
0000000
0001
XXXX
XXXXXXXXXXXXXXX
X1: 3,000,002,000.000 ns


Objects
Simulation Objects for finalproject
bject Name
Value
clk 0
push 1
reset 0
in[3:0] 0010
num[3:0] 0001
point[6:0] 0000000
seg[7:1] 0000000
d1 1
d2 1
d3 1
d4 1
level[3:0] 0001
q[3:0] xxxx
ud 1
start 1
cnt[15:0] xxxxxxxxxxxxxxxx
k[31:0] 0000000000111111...
flag 0
flag1 1
second1[3:0] 0000
second2[3:0] 1111
Name
Value
4,414,793,650.000 ns
4,414,792,500 ns
4,414,793,000 ns
4,414,793,500 ns
0010
0001
0000000
XXXX
XXXXXXXXXXXXXXX
X1: 4,414,793,650.000 ns

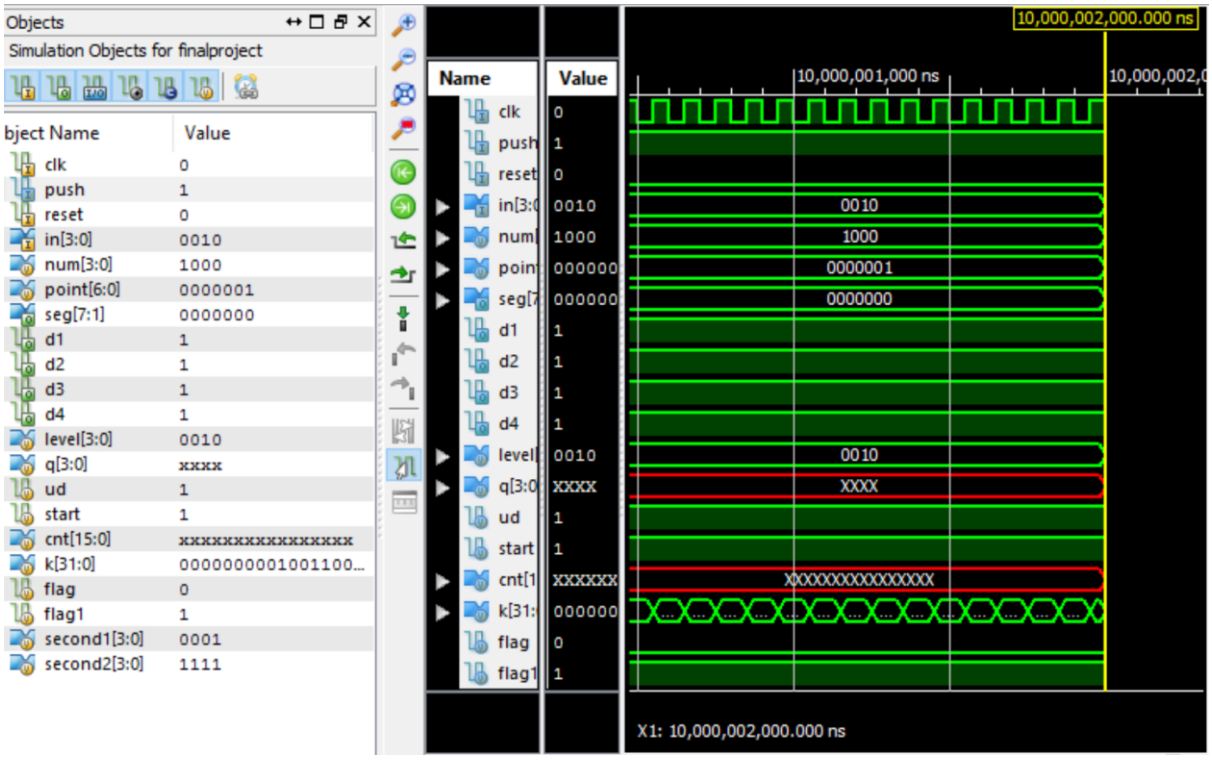
در لحظه عکس زیر باید push را force constant کنیم به مقدار صفر که معنی رها شدن است. سپس امتیاز مشاهده می‌شود.



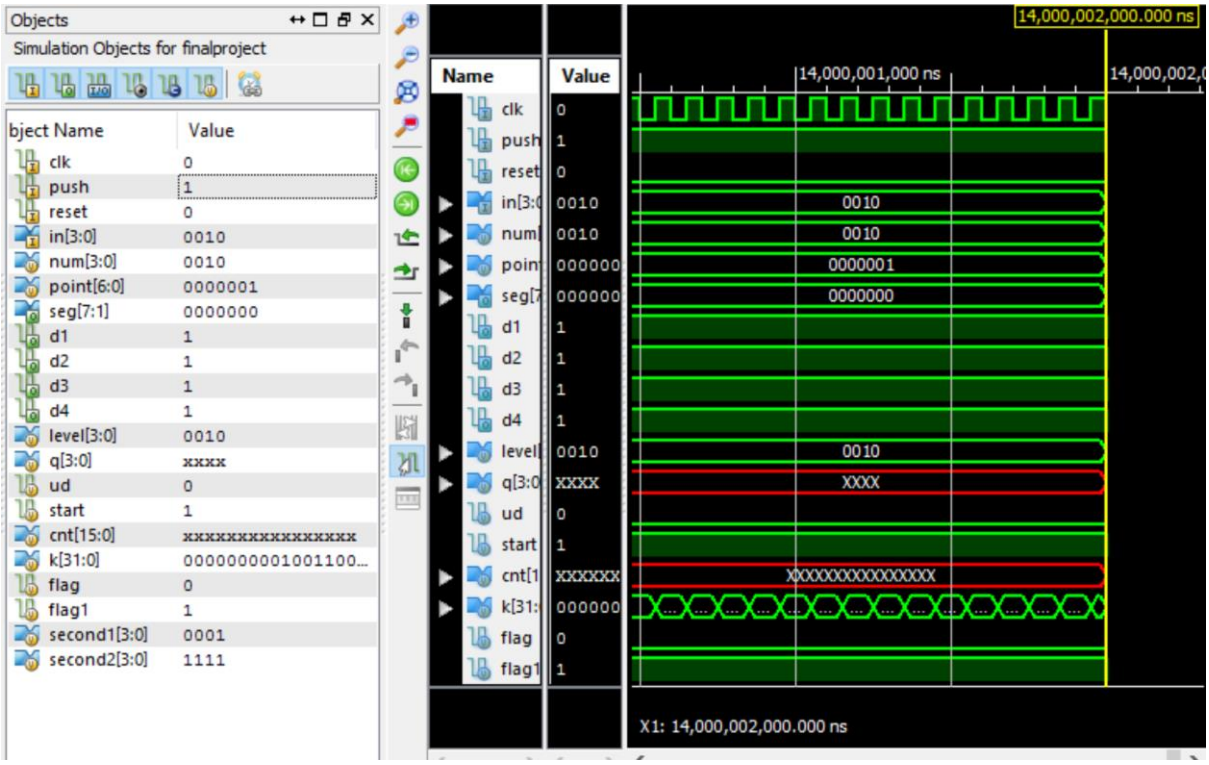
در عکس زیر Level و point هر دو اضافه شده و در مقدار صحیح قرار دارند.



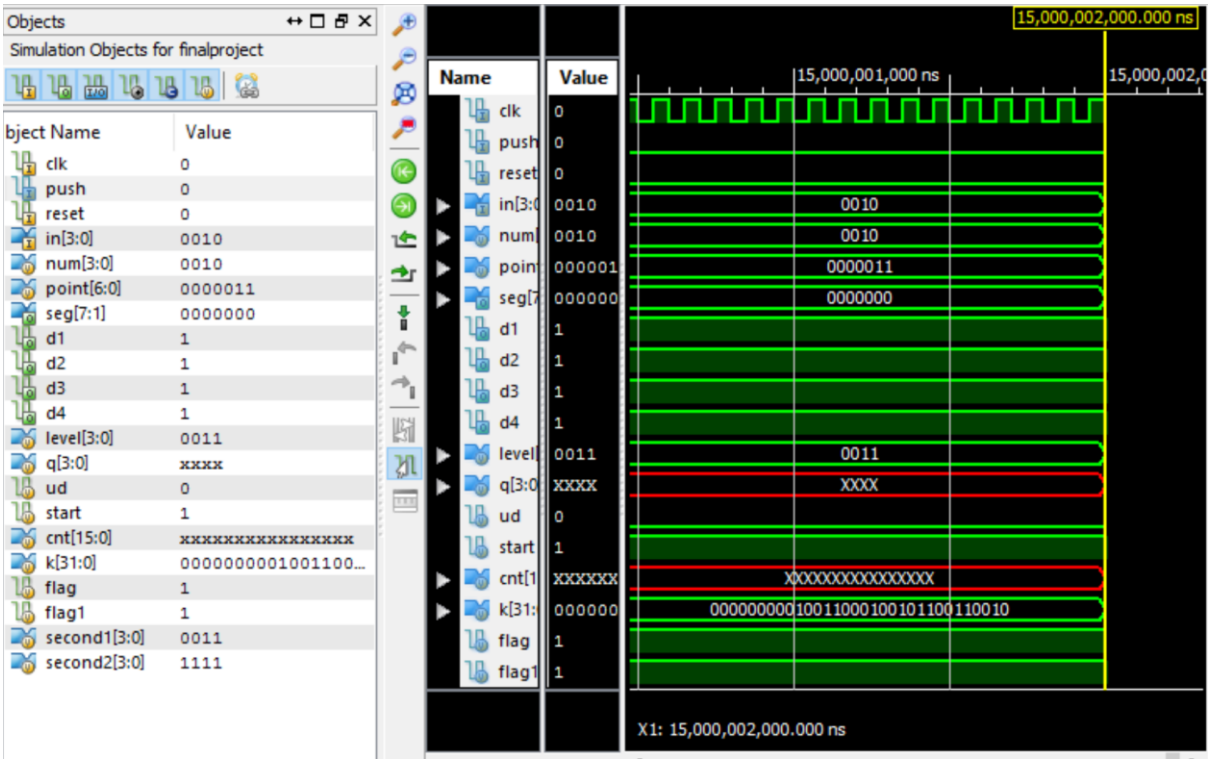
در تصویر زیر یکی از مراحل میانی که عدد ۸ است نمایش داده شده است.



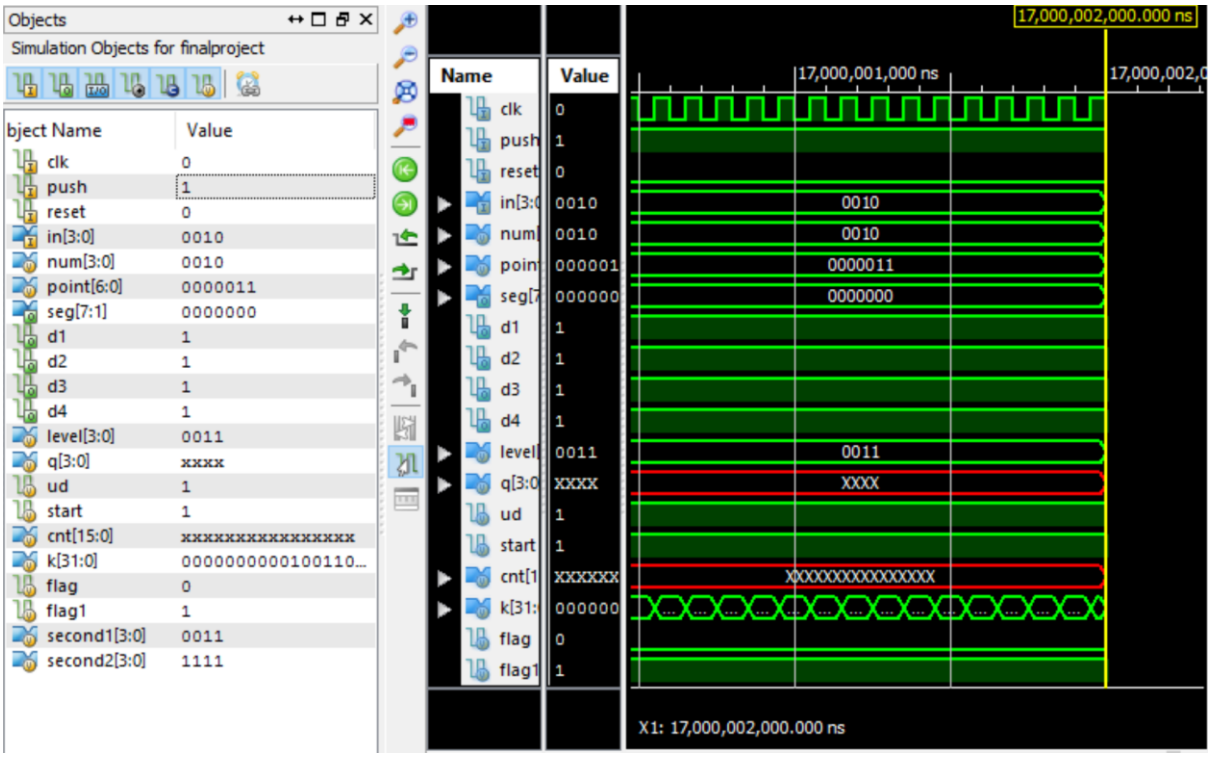
در تصویر زیر باز باید push را force constant کنیم به صفر که رها شود.



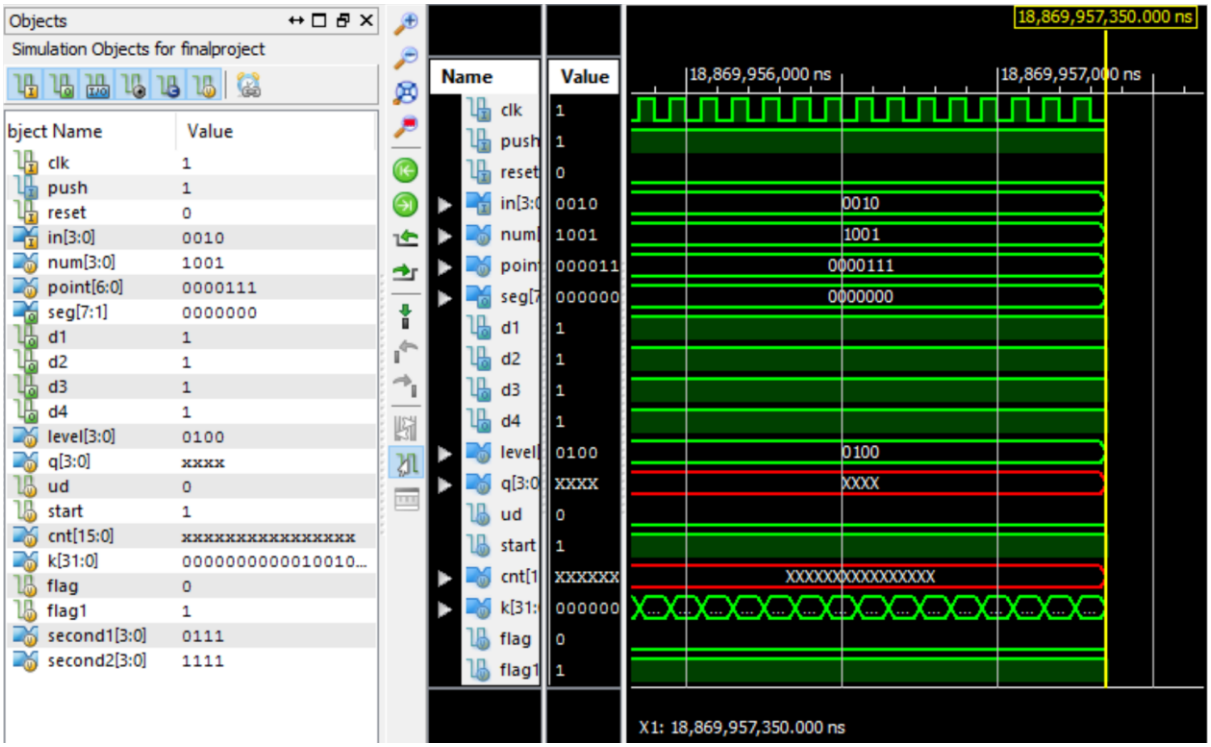
در تصویر زیر مشاهده می‌شود که مرحله و امتیاز به درستی نشان داده شده اند.



در این مرحله نیز باید کلید رها شود تا به مرحله بعد برویم.



مشاهده می‌شود که در مراحل میانی امتیاز و مرحله به درستی نشان داده شده است.



به همین ترتیب می‌توان مشاهده کرد که تا امتیاز ۶۳ قابل نمایش است که در تحویل بررسی شده است.