پیش گزارش طراحی چراغ راهنمایی برای چهارراه (قسمت دوم)

نام: آناهیتا　　نام خانوادگی: اسدی　　شماره دانشجویی: 9724453

نام مدرس: سرکار خانم فرشته کلانتری　　روز و ساعت کلاس: یکشنبه ها، ساعت 13:30 الی 15:30

پاسخ مرحله 1-1

.1

```
library IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE IEEE.std_logic_unsigned.ALL;
```

اضافه کردن کتابخانه های موردنیاز؛ کتابخانه اول برای گیت های منطقی (در این آزمایش not کردن یک سیگنال)، کتابخانه دوم برای عملیات ریاضی (مانند جمع کردن i و count با 1) و کتابخانه سوم برای اعداد غیرعلامت دار (برای مثال 1000 را 8 (و نه -7) درنظر میگیرد) است. کتابخانه آخر، برای خواندن فایل است که عملیات readline و... و تایپ های file و line در آن تعریف شده.

```
entity blink_control is port( seg1, seg2: out std_logic_vector(6 downto 0);
                              mode: out std_logic_vector(1 downto 0);
                              clk: in std_logic;
                              LED1, LED2: out std_logic_vector(2 downto 0));
end blink_control;
```

تعریف پورت های ورودی (clk) و خروجی (seg1 و seg2 برای سون سگمنت ها، mode برای مشخص کردن این که کدام چراغ باید روشن باشد، LED1 و LED2 برای چراغ های سبز، زرد یا قرمز)

```
architecture behavioral of blink_control is

   type state is (led_off, led_on);
   signal pr_state: state := led_off;

   begin

process(clk)
   begin
      if rising_edge(clk) then
         case pr_state  is when led_off =>LED1 <= "000";
                                          LED2 <= "000";
                                          mode <= "00";
                                          pr_state <= led_on;
                           when led_on => LED1 <= "100";
                                          LED2 <= "010";
                                          mode <= "00";
                                          pr_state <= led_off;
                           when others => pr_state <= led_off;
         end case;
      end if;
end process;
end behavioral;
```

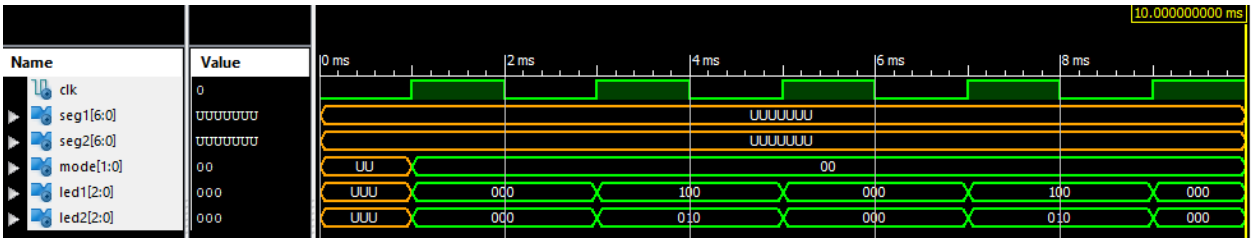
از ماشین حالتی استفاده شده که دو حالت دارد؛ روشن بودن LEDها (قرمز بودن اولی (بیت اول مختص قرمز، بیت دوم زرد و بیت سوم سبز است) و زرد بودن دومی) و خاموش بودن LEDها. و در لبه های بالارونده کلاک ها بین این دو حالت سوییچ میشود.

پاسخ مرحله 1-2

کد تست بنچ:

```
clk <= not(clk) after 1 ms;
```

نتیجه تست بنچ:



پاسخ مرحله 1-3

کد UCF:

```
net "mode[0]" loc = p24;
net "mode[1]" loc = p3;

net "clk" loc = p80;

net "LED1[0]" loc = p93;
net "LED1[1]" loc = p94;
net "LED1[2]" loc = p95;

net "LED2[0]" loc = p96;
net "LED2[1]" loc = p97;
net "LED2[2]" loc = p100;
```

در نهایت ماژول ها متصل شده اند.

پاسخ مرحله 2-1

```
library IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE IEEE.std_logic_unsigned.ALL;
```

اضافه کردن کتابخانه های موردنیاز

```
entity normal_control is port(seg1, seg2: out std_logic_vector(6 downto 0);
                              mode: out std_logic_vector(1 downto 0);
                              clk: in std_logic;
                              LED1, LED2: out std_logic_vector(2 downto 0);
                              Red, Green, Yellow: in std_logic_vector(6 downto 0));
end normal_control;
```

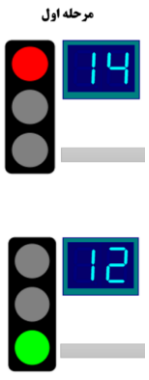
تعریف پورت های ورودی و خروجی

```
architecture behavioral of normal_control is

   type state is (S1, S2, S3, S4);
   signal pr_state: state := S1;

   begin

process(clk)

   variable count, seg2_temp, seg1_temp: std_logic_vector(6 downto 0) := "0000000";
```

تعریف ماشین حالت با 4 حالت

```
   begin
      if rising_edge(clk) then
         if(   Red < "1100011" and Green < "1100011" and Yellow < "1100011" and
               Red > "0000000" and Green > "0000000" and Yellow > "0000000") then
            case pr_state  is when S1 =>  LED1 <= "100";
                                          LED2 <= "001";
                                          mode <= "11";
                                          seg1_temp := Red - count;
                                          seg2_temp := Green - count;
                                          if seg2_temp = "0000000" then
                                             pr_state <= S2;
                                             count := "0000000";
                                          else
                                             count := count + 1;
                                             pr_state <= S1;
                                          end if;
```

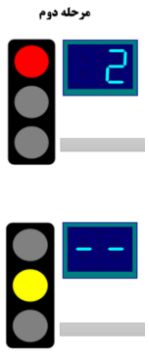
حالت اول برای شکل زیر است:



هر دو چراغ روشن اند (mode 11) و چراغ اول قرمز (100) و چراغ دوم سبز (001) است. ورودی Red وارد چراغ اول، و Green وارد چراغ دوم میشود. شمارش به پایین آغاز میشود و وقتی چراغ دوم 0 شد، مرحله دوم آغاز میشود:

```
when S2 =>  LED1 <= "100";
            LED2 <= "010";
            mode <= "10";
            seg1_temp := Yellow - count;
            if seg1_temp = "000000" then
               pr_state <= S3;
               count := "0000000";
            else
               count := count + 1;
               pr_state <= S2;
            end if;
```

چراغ اول قرمز (100) و چراغ دوم زرد (010) است. مقدار Yellow که همان |Red-Green| است وارد چراغ اول میشود و تا وقتی 0 شود، خروجی دومی – خواهد بود (mode 10).




```
when S3 =>  LED1 <= "001";
            LED2 <= "100";
            mode <= "11";
            seg1_temp := Green - count;
            seg2_temp := Red - count;
            if seg1_temp = "0000000" then
               pr_state <= S4;
               count := "0000000";
            else
               count := count + 1;
               pr_state <= S3;
            end if;
```

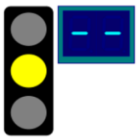
چراغ اول سبز (001) و دوم قرمز (100) است. Green وارد چراغ اول و Red وارد چراغ دوم میشود (هردو چراغ روشن اند پس مد کاری 11 است) و تا وقتی چراغ اول (سبز) 0 نشده شمارش پایین انجام میشود:

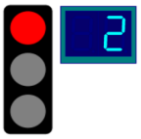
مرحله سوم


```
                when S4 =>  LED1 <= "001";
                            LED2 <= "100";
                            mode <= "01";
                            seg2_temp := Yellow - count;
                            if seg2_temp = "0000000" then
                               pr_state <= S1;
                               count := "0000000";
                            else
                               count := count + 1;
                               pr_state <= S4;
                            end if;
                when others => pr_state <= S1;
        end case;
```

Yellow وارد چراغ دوم میشود و تا وقتی 0 نشده، مد کاری 01 باقی میماند (چراغ اول: --) و سپس، چرخه به اول بازمیگردد.

مرحله چهارم






```
            seg1 <= seg1_temp;
            seg2 <= seg2_temp;
         end if;
      end if;
end process;
end behavioral;
```

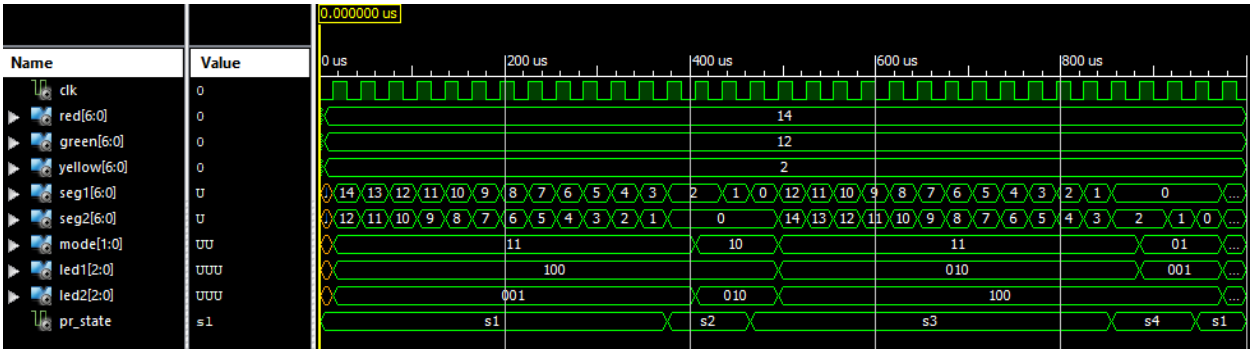
قرار دادن سیگنال های خروجی در پورت خروجی اصلی.

پاسخ مرحله 2-2

کد تست بنچ:

```
clk <= not(clk) after 15 us;
Red <= "0001110" after 5 us;
Yellow <= "0000010" after 5 us;
Green <= "0001100" after 5 us;
```

نتیجه تست بنچ:



پاسخ مرحله 2-3

کد UCF:

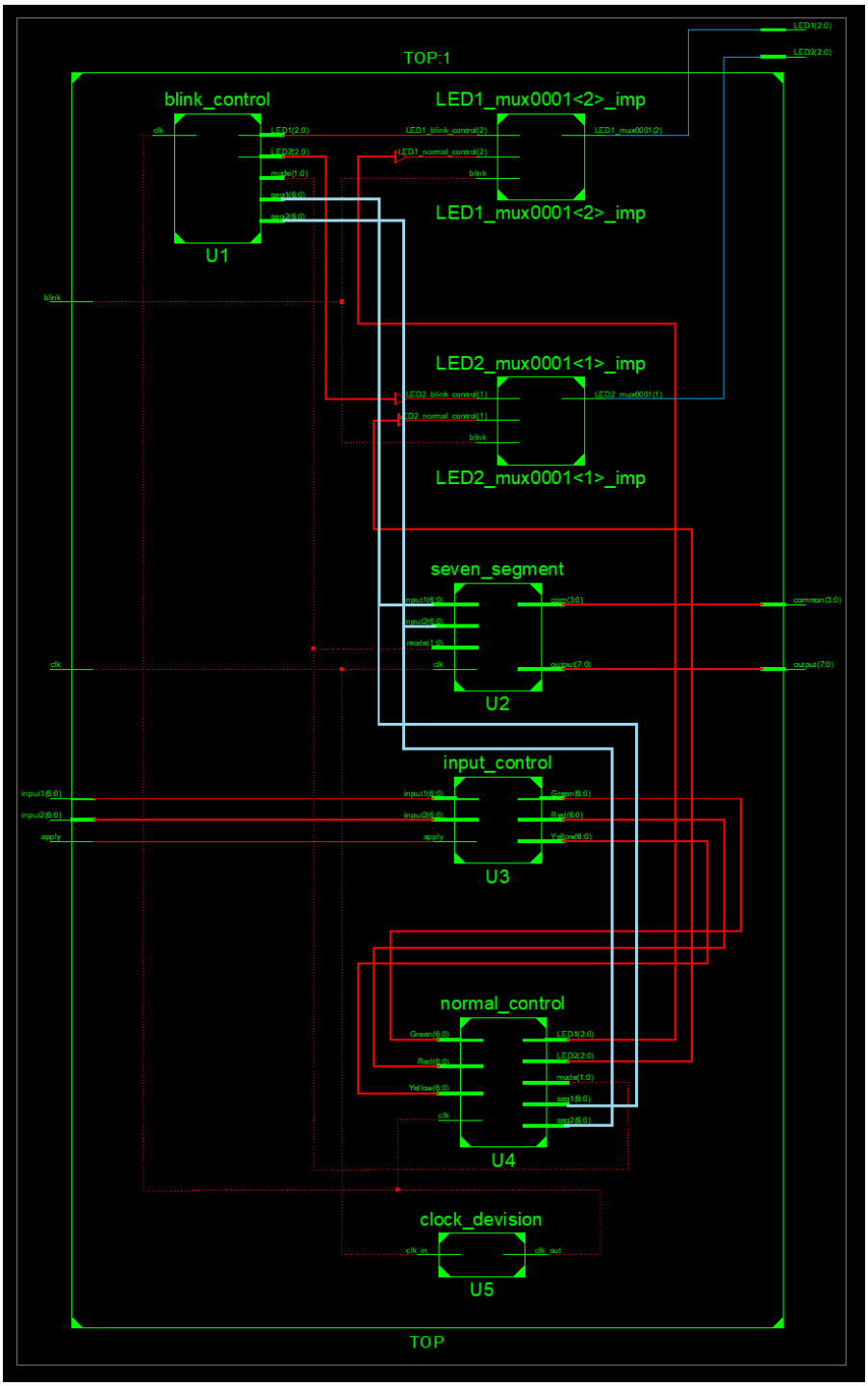
```
net "mode[0]" loc = p24;
net "mode[1]" loc = p3;

net "clk" loc = p80;

net "LED1[0]" loc = p93;
net "LED1[1]" loc = p94;
net "LED1[2]" loc = p95;

net "LED2[0]" loc = p96;
net "LED2[1]" loc = p97;
net "LED2[2]" loc = p100;
```

در نهایت ماژول ها متصل شده اند.

پاسخ مرحله 3-1

1. از آنجا که بجای استفاده از کامپوننت جدا، کد انتخاب بین blink_control و normal_control در پروسه نوشته شد، خود برنامه از آن را بعنوان مالتی پلکسر (با select دو حالته برای blink = ‘0’ و blink = ‘1’) سنتز کرده. بقیه کامپوننت ها، بصورت خواسته شده هستند. اعداد وارده (input1 و input2 وارد input_control میشوند، و درصورت دو رقمی بودن و یکسان نبودن، تفاضلشان در Yellow، عدد بزرگتر در Red و عدد کوچکتر در Green ریخته میشود. و سپس به normal_control داده میشوند. Multiplexerها متناسب با blink تعیین میکنند کلاک کدام کامپوننت (blink_control و normal_control) 0 و کدام یک برابر clk_dev (که فرکانس تقسیم شده کلاک داخلی FPGA

با DEV = 250 است) باشد. خروجی ها در LEDها و modeهایی ریخته میشوند که هر یک آندرلاین blink_control یا normal_control دارند. و دوباره MUX تعیین میکند کدام یک از آنها نهایی باشد و در LED پورت خروجی ریخته شود و بعنوان mode در بقیه کامپوننت ها مبادله شود. همچنین، خروجی عددی این کامپوننت ها تحت عنوان input_temp به سون سگمنت داده میشود، و خروجی نهایی آن از فایل values.txt به سون سگمنت داده میشود.



.2

```
library IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE IEEE.std_logic_unsigned.ALL;
USE STD.TEXTIO.ALL;
```

اضافه کردن کتابخانه ها

```
entity TOP is port(  apply: in std_logic;
                     blink: in std_logic;
                     clk: in std_logic;
                     input1, input2: in std_logic_vector (6 downto 0);
                     output: out bit_vector(7 downto 0);
                     common: out std_logic_vector(3 downto 0);
                     LED1, LED2: out std_logic_vector(2 downto 0));
end TOP;
```

تعریف پورت های ورودی و خروجی

```
architecture Behavioral of TOP is

   component blink_control port( seg1, seg2: out std_logic_vector(6 downto 0);
                                 mode: out std_logic_vector(1 downto 0);
                                 clk: in std_logic;
                                 LED1, LED2: out std_logic_vector(2 downto 0));
   END component;

   component seven_segment port( input1, input2: in std_logic_vector(6 downto 0);
                                 mode: in std_logic_vector(1 downto 0);
                                 clk: in std_logic;
                                 output: out bit_vector(7 downto 0);
                                 com: out std_logic_vector(3 downto 0));
   END component;

 component input_control port( input1, input2: in std_logic_vector(6 downto 0);
                               apply: in std_logic;
                               Green, Yellow, Red: out std_logic_vector(6 downto 0));
 END component;

 component normal_control port(seg1, seg2: out std_logic_vector(6 downto 0);
                               mode: out std_logic_vector(1 downto 0);
                               clk: in std_logic;
                               LED1, LED2: out std_logic_vector(2 downto 0);
                               Red, Green, Yellow: in std_logic_vector(6 downto 0));
 END component;

 component clock_devision generic(DIV  :  integer := 10);
                          port(   clk_in:  in std_logic;
                                  clk_out: out std_logic);
 END component;
```

اضافه کردن کامپوننت ها و تعریف پورت هایشان

```
 signal mode: std_logic_vector(1 downto 0) := "00";
 signal Green, Yellow, Red: std_logic_vector(6 downto 0) := "0000000";
 signal input1_temp, input2_temp: std_logic_vector(6 downto 0) := "0000000";
 signal clk_dev: std_logic := '0';
 ------------------------------------------------------------------------------------------
 signal seg1_blink_control, seg2_blink_control, seg1_normal_control, seg2_normal_control: std_logic_vector(6 downto 0) :
 signal mode_blink_control, mode_normal_control: std_logic_vector(1 downto 0) := "00";
 signal blink_select, normal_select: std_logic := '0';
 signal LED1_blink_control, LED2_blink_control, LED1_normal_control, LED2_normal_control: std_logic_vector(2 downto 0) :
```

تعریف سیگنال های کمکی (که در پورت های ورودی و خروجی نیستند ولی در کامپوننت ها بکار رفته اند)

```
begin

   U1: blink_control port map(seg1_blink_control, seg2_blink_control, mode_blink_control, blink_select,
                              LED1_blink_control, LED2_blink_control);
   U2: seven_segment port map(input1_temp, input2_temp, mode, clk, output, common);
   U3: input_control port map(input1, input2, apply, Green, Yellow, Red);
   U4: normal_control port map(  seg1_normal_control, seg2_normal_control, mode_normal_control, normal_select,
                                 LED1_normal_control, LED2_normal_control, Red, Green, Yellow);
   U5: clock_devision generic map (250) port map(clk, clk_dev);
```

پورت مپ و جنریک مپ کردن کامپوننت ها

```
   process(clk)
   begin
      if rising_edge(clk) then
         if blink = '1' then
            normal_select <= '0';
            blink_select <= clk_dev;
            mode <= mode_blink_control;
            LED1 <= LED1_blink_control;
            LED2 <= LED2_blink_control;
            input1_temp <= seg1_blink_control;
            input2_temp <= seg2_blink_control;
         else
            blink_select <= '0';
            normal_select <= clk_dev;
            mode <= mode_normal_control;
            LED1 <= LED1_normal_control;
            LED2 <= LED2_normal_control;
            input1_temp <= seg1_normal_control;
            input2_temp <= seg2_normal_control;
         end if;
      end if;
   end process;

end Behavioral;
```

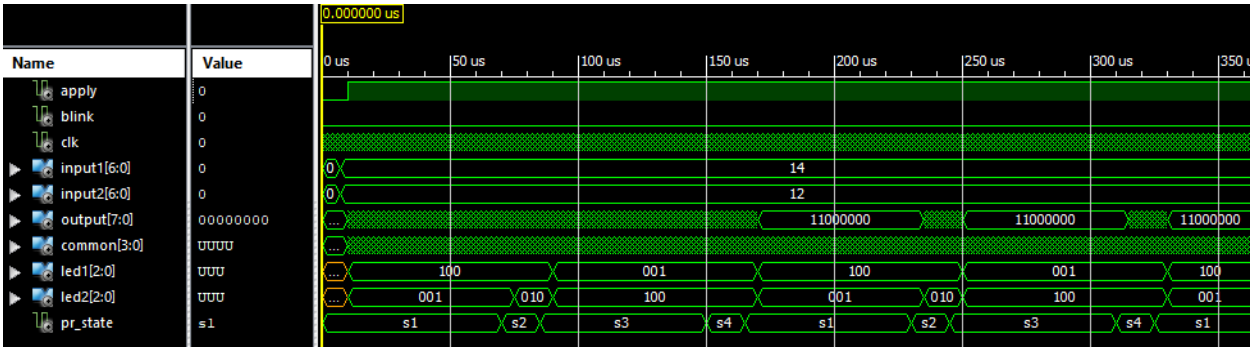
کد انتخاب بین سیگنال های خروجی blink_control و normal_control و همچنین خاموش کردن کامپوننتی بکار نرفته

پاسخ مرحله 3-2

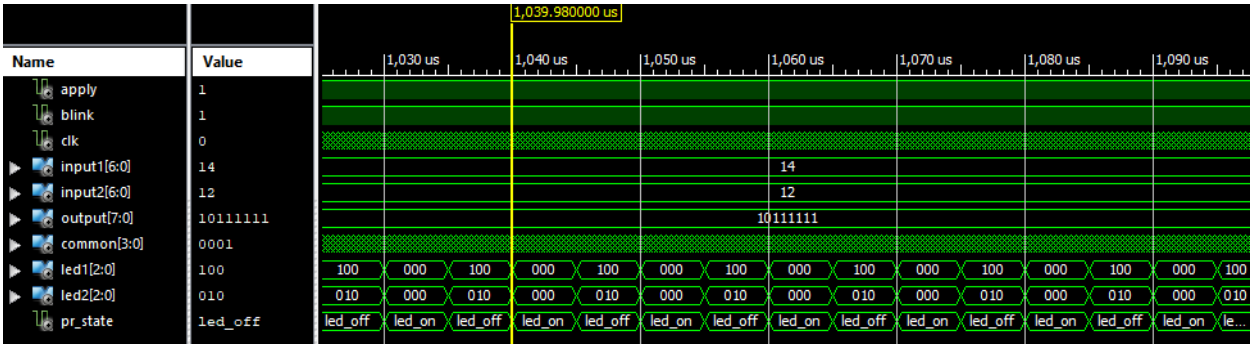
کد تست بنچ:

```
apply <= '1' after 10 us;
blink <= '1' after 1 ms;
clk <= not(clk) after 5 ns;
input1 <= "0001110" after 7 us;
input2 <= "0001100" after 7 us;
```

نتیجه تست بنچ برای حالت نرمال:



حالت چشمک زن:



پاسخ مرحله 3-3

کد UCF:

```
net "apply" loc=p34;

net "blink" loc=p12;

net "clk" loc=p80;

net "input1[0]" loc=p33;
net "input1[1]" loc=p31;
net "input1[2]" loc=p29;
net "input1[3]" loc=p28;
net "input1[4]" loc=p27;
net "input1[5]" loc=p26;
net "input1[6]" loc=p24;

net "input2[0]" loc=p11;
net "input2[1]" loc=p10;
net "input2[2]" loc=p9;
net "input2[3]" loc=p7;
net "input2[4]" loc=p5;
net "input2[5]" loc=p4;
net "input2[6]" loc=p3;
```

```
net "output[0]" loc=p131;
net "output[1]" loc=p140;
net "output[2]" loc=p139;
net "output[3]" loc=p138;
net "output[4]" loc=p137;
net "output[5]" loc=p135;
net "output[6]" loc=p133;
net "output[7]" loc=p132;

net "common[0]" loc=p125;
net "common[1]" loc=p126;
net "common[2]" loc=p128;
net "common[3]" loc=p130;

net "LED1[0]" loc=p93;
net "LED1[1]" loc=p94;
net "LED1[2]" loc=p95;

net "LED2[0]" loc=p161;
net "LED2[1]" loc=p162;
net "LED2[2]" loc=p166;
```

سیگنال های ورودی همگی به دیپ سوییچ (بجز clk که به کلاک داخلی وصل شده)، LEDها به LED، output به پایه های سون سگمنت و commonها به COMهای سون سگمنت وصل شده اند.