گزارشکار جلسه سوم آزمایشگاه مدار منطقی

آرمین افضلی 400521054

محمدصالح پژند 400521171

در این آزمایش قصد داریم با استفاده از 4 عدد 7seg یک شمارنده درست کنیم. برای این کار به طور کلی به چند process نیاز داریم که رد ادامه هرکدام توضیح داده میشوند.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee. numeric_std;
```

کتابخانه های استفاده شده: (برای جمع std_vector ها) از آنجایی که نیاز است اعداد باینری را با هم جمع کنیم از این کتابخانه ها استفاده میکنیم که کار را ساده تر میکند

```
entity bcd_7seg is
    Port ( gclock : in  STD_LOGIC;
           selseg : out std_logic_vector (3 downto 0);
           output : out  STD_LOGIC_VECTOR (6 downto 0));
end bcd_7seg;
```

موجودیت:

ورودی ها:

gclock که clock موجود بر روی برد است. و مبنای تمامی کلاک های مختلف ما در دستگاه است که با تغییر این فرکانس به انها را ایجاد میکنیم.

خروجی ها:

selseg که مشخص کننده active 7seg برد است. این خروجی مشخص میکند که در هر لحظه کدام سلکتورها باید فعال باشند. در این ازمایش ما در هر لحظه فقط یک 7seg را روش میکنیم اما چون فرکانس روشن شدنشان زیاد است به نظر میرسد که همگی همزمان روشن هستند.

output که مشخص کننده مقدار 7seg است.

```
signal clock10ms : std_logic := '0';
signal clock1s : std_logic := '0';

signal counter1, counter2, counter3, counter4 : std_logic_vector(3 downto 0) := "0000";
signal data1, data2, data3, data4 : std_logic_vector (6 downto 0) := "0000000";
```

سیگنال ها:

```
with counter1 select
   data1 <= "0111111" when "0000",
   "0000110" when "0001",
   "1011011" when "0010",
   "1001111" when "0011",
   "1100110" when "0100",
   "1101101" when "0101",
   "1111101" when "0110",
   "0000111" when "0111",
   "1111111" when "1000",
   "1101111" when "1001",
   "0000000" when others;
```

مقدار data که توسط counter تعیین می شود؛ این ساختار برای تمامی سیگنال های data و counter تکرار می شود.

```
process (gclock)
variable count_div : integer range 0 to 50000 := 0;
begin
   if (rising_edge(gclock)) then
      if (count_div < 50000) then
         count_div := count_div + 1;
      else
         count_div := 0;
         clock10ms <= not clock10ms;
      end if;
   end if;
end process;
```

ساختار clock 10ms: این کلاک به این منظور ساخته شده است که میزان زمان اکتیو بودن هر 7seg را مشخص کند.در ادامه خواهیم دید که بعد از اتمام هر سیکل این کلاک 7segفعلی خاموش شده و بعدی روشن میشود . اگر فرکانس این کلاک خیلی بیشتر باشد 7seg ها قبل از کامل روشن شدن سیکلشان به پایان میرسد و خاموش میشوند. اگر هم خیلی کمتر باشد بنظر میرسد که همزمان روشن نیستند.

```
process (clock10ms)
variable count_div : integer range 0 to 50 := 0;
begin
   if (rising_edge(clock10ms)) then
      if (count_div < 50) then
         count_div := count_div + 1;
      else
         count_div := 0;
         clock1s <= not clock1s;
      end if;
   end if;
end process;
```

ساختار clock 1s: نقش این کلاک شمارش اصلی عدد داخل 7seg ها است . در هر کلاک به عدد bcd کم ارزشترین 7seg اضافه میشود و طبق کدی که در ادامه میبینیم هر زمان عدد bcd معادل 9 شد به مقدار7seg بعدی اضافه میشود و مقدار خودش صفر میشود. این روند در تمامی 7seg ها جریان دارد.

```
process (clock10ms)
variable refresh : integer range 0 to 3 := 0;
begin
if (rising_edge(clock10ms)) then
   if (refresh < 4) then
      refresh := refresh + 1;
   else
      refresh := 0;
   end if;

   case refresh is
      when 0 =>
         selseg(3) <= '0';
         selseg(0) <= '1';
         output <= data1;
      when 1 =>
         selseg(0) <= '0';
         selseg(1) <= '1';
         output <= data2;
      when 2 =>
         selseg(1) <= '0';
         selseg(2) <= '1';
         output <= data3;
      when 3 =>
         selseg(2) <= '0';
         selseg(3) <= '1';
         output <= data4;
      when others => null;
   end case;
end if;
end process;
```

Refresh شدن 7seg ها که به کمک clock 10ms سریع اتفاق می افتد و توسط چشم غیر قابل تشخیص است.

نکته: همانطور که گفته شد در آن واحد تنها یک 7seg روشن است.

این process با استفاده از کلاکی که گفته شد در هر سیکل 7seg فعلی را خاموش و بعدی را روشن میکند و مقدار 7seg جدید را هم بر روی خروجی لود میکند

```
process(clock1s)

variable value1, value2, value3, value4 : std_logic_vector(3 downto 0) := "0000";

begin
if (rising_edge(clock1s)) then
   if (value1 < "1001") then
      value1 := value1 + 1;
   else
      value1 := "0000";
      if (value2 < "1001") then
         value2 := value2 + 1;
      else
         value2 := "0000";
         if (value3 < "1001") then
            value3:= value3 + 1;
         else
            value3 := "0000";
            if (value4 < "1001") then
               value4 := value4 + 1;
            else
               value4 := "0000";
            end if;
         end if;
      end if;
   end if;
   counter4 <= value4;
   counter3 <= value3;
   counter2 <= value2;
   counter1 <= value1;
   end if;
end process;
```

افزایش مقدار counter ها که به ترتیب از یک تا چهار نقش یکان، دهگان، صدگان و هزارگان را دارند.

در این قسمت که با کلاک یک ثانیه کار میکند در هر سیکل چک میکند اگر مقدارش مساوی 9 نباشد به یکان یکی اضافه میکند و اگر مقدار به 9 رسید وارد شرط بعدی میشود خودش را صفر میکند و به دهگان یکی اضافه میکند. این روند در برای همه ارزش های مکانی تکرار میشود. در انتهای هر سیکل مقادیر بر روی شمارنده ها لود میشود تا نشان داده شوند.

: Constrains

```
1  NET "gclock" CLOCK_DEDICATED_ROUTE = FALSE;
2  NET "gclock" LOC = P184;
3  NET "output[0]" LOC = P10;
4  NET "output[1]" LOC = P7;
5  NET "output[2]" LOC = P11;
6  NET "output[3]" LOC = P5;
7  NET "output[4]" LOC = P4;
8  NET "output[5]" LOC = P12;
9  NET "output[6]" LOC = P9;
10 NET "selseg[0]" LOC = P15;
11 NET "selseg[1]" LOC = P20;
12 NET "selseg[2]" LOC = P19;
13 NET "selseg[3]" LOC = P18;
```