پیش گزارش راه اندازی کیبورد توسط پروتکل PS2

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_unsigned.all;
```

تعریف کتابخانه ها

```
ENTITY debouncer IS
  GENERIC(
    counter_size  :  INTEGER := 10); --counter size (10 bits gives 40.9us with 50MHz clock)
  PORT(
    clk     : IN  STD_LOGIC;  --input clock 50MHz
    button  : IN  STD_LOGIC;  --input signal to be debounced
    result  : OUT STD_LOGIC); --debounced signal
END debounce;
```

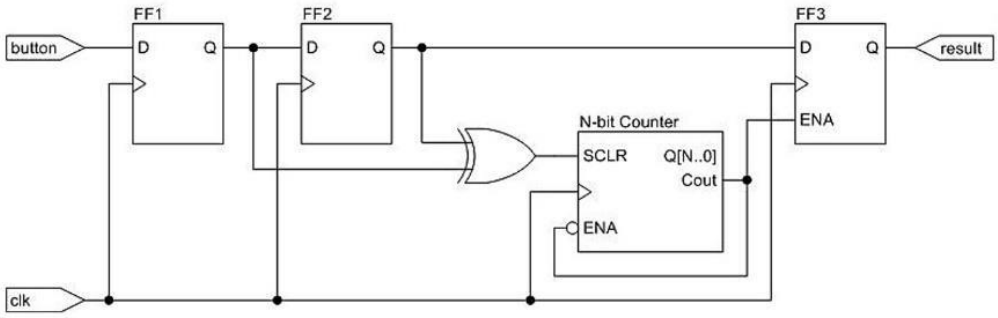
ورودی های این کد، کلاک 50MHz داخلی FPGA و داده یک بیتی هستند. خروجی قرار است همان داده یک بیتی باشد.

```
ARCHITECTURE logic OF debouncer IS
  SIGNAL flipflops   : STD_LOGIC_VECTOR(1 DOWNTO 0); --input flip flops
  SIGNAL counter_set : STD_LOGIC;                    --sync reset to zero
  SIGNAL counter_out : STD_LOGIC_VECTOR(counter_size DOWNTO 0) := (OTHERS => '0'); --counter output
BEGIN

  counter_set <= flipflops(0) xor flipflops(1);   --determine when to start/reset counter

  PROCESS(clk)
  BEGIN
    IF(clk'EVENT and clk = '1') THEN
      flipflops(0) <= button;
      flipflops(1) <= flipflops(0);
      If(counter_set = '1') THEN                  --reset counter because input is changing
        counter_out <= (OTHERS => '0');
      ELSIF(counter_out(counter_size) = '0') THEN --stable input time is not yet met
        counter_out <= counter_out + 1;
      ELSE                                        --stable input time is met
        result <= flipflops(1);
      END IF;
    END IF;
  END PROCESS;
END logic;
```

مطابق شکل زیر:



ورودی فلیپ فلاپ اول، همان ورودی ما (botton) است. باید از دو فلیپ فلاپ استفاده (برای مقایسه داده در دو زمان متفاوت –که به فرکانس کلاک بستگی دارد-) شود و خروجی هایشان مقایسه شود. این سیگنال چون در process(clk) قرار میگیرند خود به خود بصورت فلیپ فلاپ سنتز میشوند.

اگر ورودی هنوز bounce داشته باشد، خروجی های این دو فلیپ فلاپ متفاوت خواهند بود، پس خروجی گیت XOR (ورودی counter) برابر 1 میشود. طبق کد، اگر ورودی counter برابر 1 باشد، خروجی آن 0 است (reset شده) و چون خروجی counter به enable فلیپ فلاپ سوم وصل است، این فلیپ فلاپ خاموش میماند.

اگر bounce ورودی تمام شده باشد، خروجی فلیپ فلاپ ها یکسان خواهد بود. پس خروجی XOR برابر 0 میشود. طبق کد، در این حالت counter از 0 تا 01111111111 میشمارد (چون به محض 1 شدن بیت یازدهم، ورودی در result قرار میگیرد) پس یعنی 2^{11} تا میشمارد (یعنی تاخیر تقریبا برابر $40.6\mu s \cong \frac{2^{11}}{50\times10^{6}}$) و سپس خروجی فلیپ فلاپ دوم را بعنوان نتیجه نهایی میدهد.

یعنی ورودی پس از 40.6us نهایی میشود.

.2

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

اضافه کردن کتابخانه ها

```
entity PS2 is port(  ps2_clk: in std_logic;
                     ps2_data: in std_logic;
                     ps2_code: out std_logic_vector(7 downto 0);
                     ps2_code_new: out std_logic);
end PS2;
```

تعریف پورت های ورودی (کلاک و داده یک بیتی سری) و خروجی (داده 8 بیتی موازی و داده یک بیتی برای نشان دادن آماده بودن PS2 برای دریافت سری 11 تایی از داده های بعدی.

```
architecture behavioral of PS2 is
   signal dout: std_logic_vector(7 downto 0) := (others => '0');
   signal parity_check: std_logic;
   type state is (start, B0, B1, B2, B3, B4, B5, B6, B7, parity, stop);
   signal pr_state: state;
```

تعریف انواع حالات ماشین حالت (دریافت 8 بیت، شروع، پایان، و چک کردن parity) و همچنین تعریف حالت حاضر (pr_state). همچنین داده های سری (داده دوم دریافت شده تا نهم، چون داده اول متعلق به frame شروع است و باید برابر 0 باشد تا کار کند) در dout ذخیره میشوند که در حالت parity، parity آنها حساب شود. جزییات در کد زیر مشخص است:

```
begin
   process(ps2_clk)
   begin
      if falling_edge(ps2_clk) then
         case pr_state is  when start =>  if (ps2_data = '0') then
                                             pr_state <= B0;
                                             parity_check <= '0';
                                             dout <= X"00";
                                          end if;
                           when B0 =>  pr_state <= B1;
                                       dout(0) <= ps2_data;
                           when B1 =>  pr_state <= B2;
                                       dout(1) <= ps2_data;
                           when B2 =>  pr_state <= B3;
                                       dout(2) <= ps2_data;
                           when B3 =>  pr_state <= B4;
                                       dout(3) <= ps2_data;
                           when B4 =>  pr_state <= B5;
                                       dout(4) <= ps2_data;
                           when B5 =>  pr_state <= B6;
                                       dout(5) <= ps2_data;
```

```
                when B6 =>  pr_state <= B7;
                            dout(6) <= ps2_data;
                when B7 =>  pr_state <= parity;
                            dout(7) <= ps2_data;
                when parity => pr_state <= stop;
                               parity_check <=   dout(0) xor dout(1) xor dout(2) xor
                                                 dout(3) xor dout(4) xor dout(5) xor
                                                 dout(6) xor dout(7);
                               if (ps2_data = parity_check) then
                                  pr_state <= stop;
                               end if;
                when stop =>   if (ps2_data = '1') then
                                  ps2_code_new <= '1';
                                  pr_state <= start;
                                  ps2_code <= dout;
                               end if;
                when others => pr_state <= start;
        end case;
      end if;
   end process;
end behavioral;
```

با هر لبه پایین رونده کلاک، داده های ورودی خوانده میشوند. اگر اولین بیت برابر 0 بود، ماشین حالت وارد حالات B میشود و هر 8 بیت را در dout ذخیره میکند. سپس XOR آنها را محاسبه میکند و با بیت دهم دریافت شده چک میکند. اگر این دو برابر بودند، یعنی داده ها بدون خطا دریافت شده اند پس ماشین حالت به حالت نهایی (stop) میرود. در این حالت، داده نهایی نمایش داده میشود خروجی ps2_code_new برابر 1 میشود؛ به این معنی که آماده دریافت سری داده بعدی است.

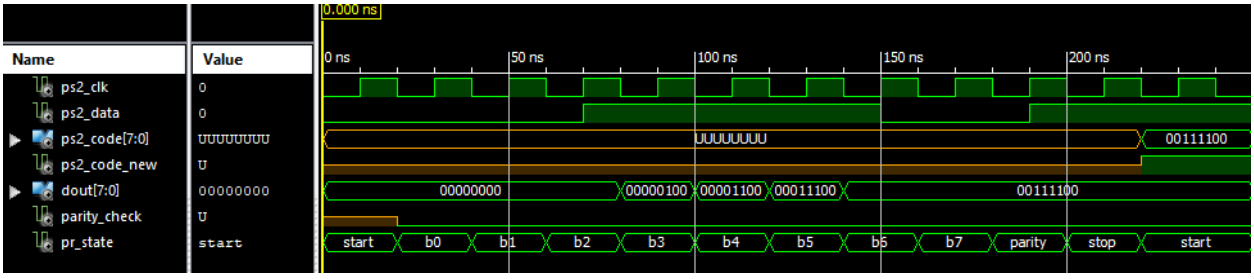
پاسخ مرحله 1-2

کد تست بنچ:

```
ps2_clk  <= not(ps2_clk) after 10 ns;

ps2_data <= '1' after 70 ns, '0' after 150 ns, '1' after 190 ns;
```

نتیجه تست بنچ:



پاسخ مرحله 2-1

```
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. کتابخانه آخر، برای خواندن فایل است که عملیات readline و... و تایپ های file و line در آن تعریف شده.

```
entity sevensegment is port(  clk: in std_logic;
                              din_ones: in std_logic_vector(3 downto 0);
                              din_tens: in std_logic_vector(3 downto 0);
                              dout: out bit_vector(7 downto 0);
                              com: out std_logic_vector(3 downto 0));
end sevensegment;
```

تعریف پورت ورودی و خروجی. 4 بیت اول dout برای یکان است، که تبدیل شده din_ones در آن قرار میگیرد و 4 بیت دوم dout برای دهگان که تبدیل شده din_tens در آن قرار میگیرد

```
architecture behavioral of sevensegment is
   type RAM_type is array (0 to 15) of bit_vector(7 downto 0);

   function ram_initialize(filename: in string) return RAM_type is
      file filel: text is in filename;
      variable linel: line;
      variable RAM1: RAM_type;
      begin for i in RAM_type'range loop  readline(filel, linel);
                                          read(linel, RAM1(i));
                                    end loop;
      return RAM1;
   end function;
```

تعریف فانکشن (زیربرنامه) خواندن فایل. این فانکشن، نام فایل را میگیرد و در حافظه RAM (که پیشتر تعریف شده 16 بایت است) میریزد.

به این صورت که فایل (که از نوع text است) را طبق نام داده شده باز میکند، و هر خط را بواسطه متغیر داخل RAM1 (خروجی) میریزد. و این کار را برای هر 16 بایت (که هر بایت در خط جدا ذخیره شده، پس برای 16 خط) تکرار میکند.

```
   signal RAM: RAM_type := ram_initialize("values.txt");
```

فانکشن را صدا میزنیم، نام فایل را بهش میدهیم، و خروجی فانکشن را در RAM ذخیره میکنیم.

```
    begin
    process(clk)
       variable flag: std_logic := '0';
    begin
    if rising_edge(clk) then
       if (flag = '0') then
          com <= "0001";
          dout <= RAM(conv_integer(din_ones));
          flag := '1';
       else
          com <= "0010";
          dout <= RAM(conv_integer(din_tens));
          flag := '0';
       end if;
    end if;
    end process;
end behavioral;
```

ورودی یکان، ابتدا به integer تبدیل میشود، که معادل 0 تا 15 خانه ی آرایه RAM شود. سپس مقدار خانه معادل، در خروجی ریخته میشود. این عمل برای دهگان نیز تکرار میشود.

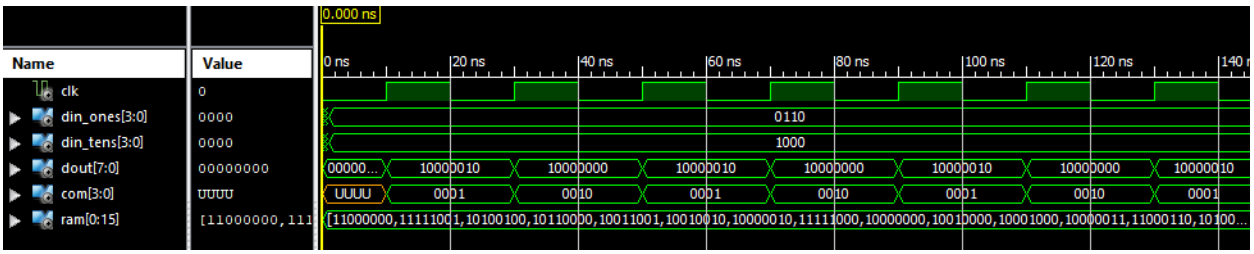
با هر کلاک، خروجی بین یکان و دهگان سوییچ میشود (با تغییر پایه common). چشم متوجه این سوییچ در فرکانس بالای کلاک (50MHz) نمیشود.

پاسخ مرحله 2-2

کد تست بنچ:

```
clk <= not(clk) after 10 ns;
din_ones <= "0110" after 1 ns;
din_tens <= "1000" after 1 ns;
```

نتیجه:



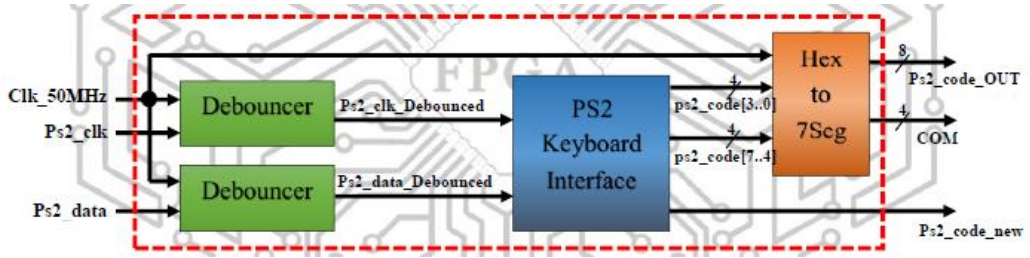
پاسخ مرحله 3-1

```
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 هایی که در componentها بکار رفته اند.

```
entity TOP is port(  ps2_clk: in std_logic;
                     clk_50MHz: in std_logic;
                     ps2_data: in std_logic;
                     ps2_code_out: out bit_vector(7 downto 0);
                     ps2_code_new: out std_logic;
                     com: out std_logic_vector(3 downto 0));
end TOP;
```

تعریف پورت های ورودی و خروجی مطابق شکل: (ps2_code_out چون داده های خوانده شده از فایل بود، و خروجی فایل هم bit_vector بود، این گونه تعریف شد)




```
architecture structral of TOP is
   component debouncer  GENERIC( counter_size  :  INTEGER := 10);
                        PORT( clk     : IN  STD_LOGIC;
                              button  : IN  STD_LOGIC;
                              result  : OUT STD_LOGIC);
   END component;
   component sevensegment port(  clk: in std_logic;
                                 din_ones: in std_logic_vector(3 downto 0);
                                 din_tens: in std_logic_vector(3 downto 0);
                                 dout: out bit_vector(7 downto 0);
                                 com: out std_logic_vector(3 downto 0));
   end component;
   component PS2 port(  ps2_clk: in std_logic;
                        ps2_data: in std_logic;
                        ps2_code: out std_logic_vector(7 downto 0);
                        ps2_code_new: out std_logic);
   end component;
   signal ps2_clk_debounced: std_logic := '0';
   signal ps2_data_debounced: std_logic := '0';
   signal ps2_code: std_logic_vector(7 downto 0):= X"00";
begin
   U1: debouncer generic map (10) port map(clk_50MHz, ps2_clk, ps2_clk_debounced);
   U2: debouncer generic map (10) port map(clk_50MHz, ps2_data, ps2_data_debounced);
   U3: PS2 port map(ps2_clk_debounced, ps2_data_debounced, ps2_code, ps2_code_new);
   U4: sevensegment port map(clk_50MHz, ps2_code(3 downto 0), ps2_code(7 downto 4), ps2_code_out, com);
end structral;
```

تعریف componentها مطابق فایل های نوشته شده. داده های سری تک بیتی ps2_clk و ps2_data، با فرکانس کلاک داخلی، debounce میشوند و با پروتکل PS2، به داده های موازی تبدیل میشوند. خروجی مبدل سون سگمنت، چون داده های خوانده شده از فایل بصورت bit_vector است، باید با این نوع تعریف شود.

پاسخ مرحله 3-2

کد UCF:

```
net "ps2_clk" loc = p64;
net "clk_50MHz" loc = p80;
net "ps2_data" loc = p63;

net "ps2_code_out[0]" = p140;
net "ps2_code_out[1]" = p139;
net "ps2_code_out[2]" = p138;
net "ps2_code_out[3]" = p137;
net "ps2_code_out[4]" = p135;
net "ps2_code_out[5]" = p133;
net "ps2_code_out[6]" = p132;
net "ps2_code_out[7]" = p131;

net "ps2_code_new" = p93;

net "std_logic_vector[0]" = p125;
net "std_logic_vector[1]" = p126;
net "std_logic_vector[2]" = p128;
net "std_logic_vector[3]" = p130;
```

Connector Clk

50MHz

Connector Data

DP-g ports of 7SEG

LED0

COM1-4 ports of 7SEG