Pre-Report

FPGA Lab Eperiment 4

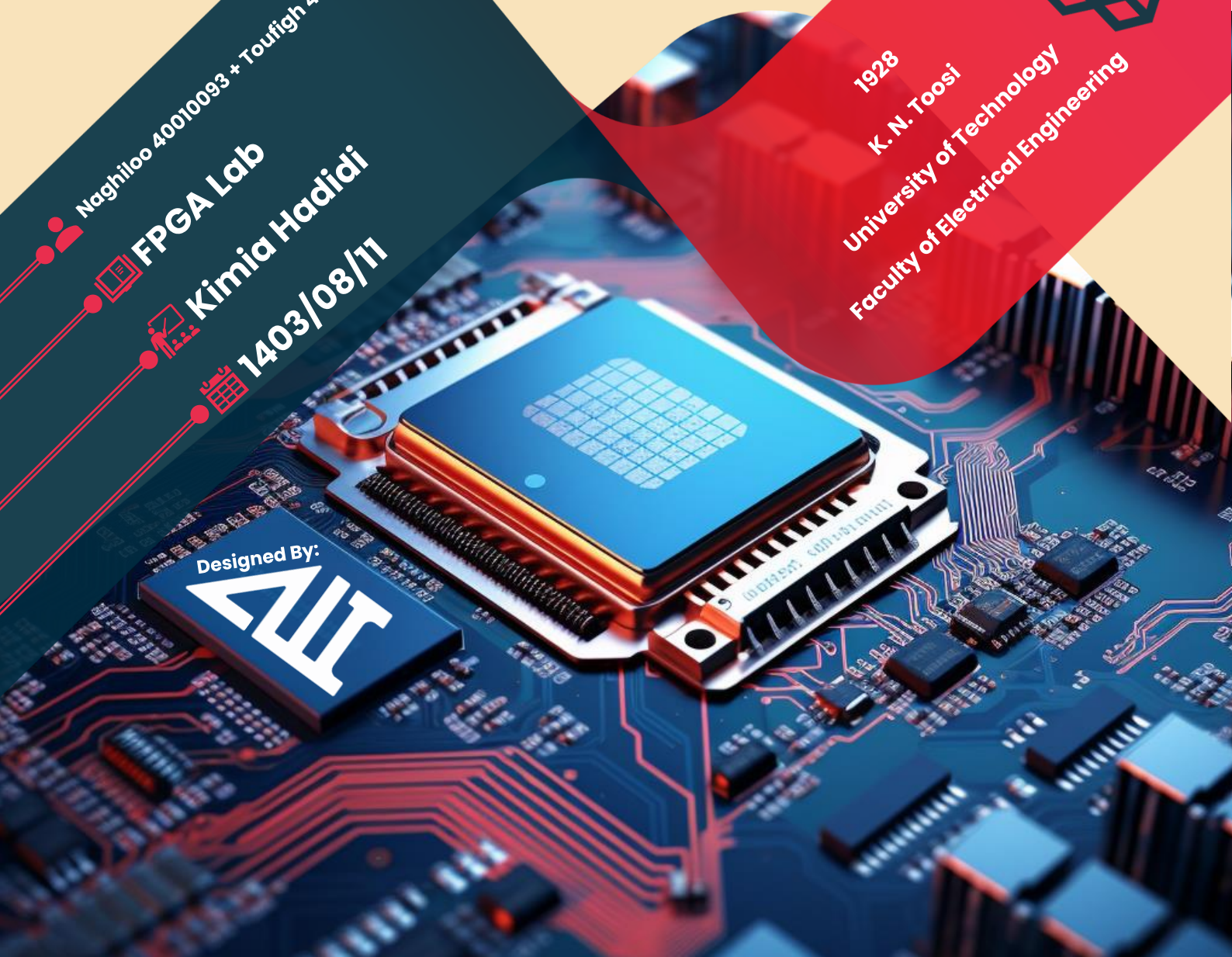
Naghiloo 40010093 + Toufigh 40003913

FPGA Lab

Kimia Hadidi

1403/08/11

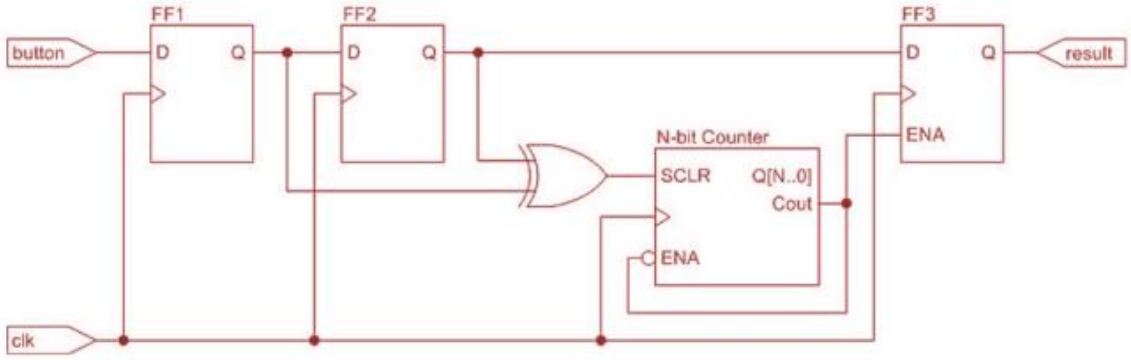
1928
K. N. Toosi
University of Technology
Faculty of Electrical Engineering



بخش اول: راه اندازی پروتکل PS2

مرحله ۱-۱: تشریح کد Debouncer

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY debounce 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;

ARCHITECTURE logic OF debounce 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;
```



این کد VHDL یک دیبانسر (Debouncer) را پیاده‌سازی می‌کند که دو ورودی دارد (کلاک و سیگنال ورودی دکمه) و یک خروجی (سیگنال رفع نویز شده). کلاک ورودی برای عملکرد فلیپ‌فلاپ‌ها و شمارنده به کار می‌رود. سیگنال‌های تعریف‌شده شامل flipflopsبرای فلیپ‌فلاپ‌ها، counter_setبه عنوان نتیجه عملگر XOR روی فلیپ‌فلاپ‌ها، و counter_outبرای خروجی شمارنده هستند. در لبه بالا رونده کلاک، مقدار ورودی به فلیپ‌فلاپ اول منتقل می‌شود و مقدار فلیپ‌فلاپ اول به فلیپ‌فلاپ دوم منتقل می‌گردد. با XOR شدن این دو مقدار، سیگنال counter_setمقداردهی می‌شود. اگر این مقدار ۱ باشد، نشان‌دهنده تغییر در سیگنال دکمه است، و کانتر ریست می‌شود. در غیر این صورت، کانتر به افزایش خود ادامه می‌دهد تا زمانی که به بیت پرارزش برسد و نشان دهد سیگنال دکمه پایدار شده است. در این حالت، عملیات رفع نویز انجام شده و نتیجه به خروجی منتقل می‌شود. پارامتر counter_sizeبه صورت genericتعریف شده و می‌تواند در طراحی اصلی با مقدار دلخواه تنظیم شود، اما مقدار پیش‌فرض آن 10 است.

مرحله ۱-۱: پروتکل PS2

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Ex_4_1 is

    port(PS2_CLK, PS2_data : in std_logic;
         PS2_code_new      : out std_logic;
         PS2_code          : out std_logic_vector(7 downto 0));

end entity;

architecture BHV of Ex_4_1 is

    type state_t is (start, D0, D1, D2, D3, D4, D5, D6, D7, parity, stop);
    signal state : state_t := start;
    signal P : std_logic :='0';

begin

    P1: process(PS2_CLK)
        begin
            if (PS2_CLK'event and PS2_CLK = '0') then
                case state is
                when start =>
                                if (PS2_data = '0') then PS2_code_new <= '0';
                                                         P            <= '0';
                                                         state        <= D0;

                                else state <= start; end if;

                when D0 =>
                                PS2_code(0) <= PS2_data;
                                P           <= P xor PS2_data;
                                state    <= D1;

                when D1 =>
                                PS2_code(1) <= PS2_data;
                                P           <= P xor PS2_data;
                                state       <= D2;

                when D2 =>
                                PS2_code(2) <= PS2_data;
                                P           <= P xor PS2_data;
                                state       <= D3;

                when D3 =>
                                PS2_code(3) <= PS2_data;
                                P           <= P xor PS2_data;
                                state       <= D4;

                when D4 =>
                                PS2_code(4) <= PS2_data;
                                P           <= P xor PS2_data;
                                state       <= D5;

                when D5 =>
                                PS2_code(5) <= PS2_data;
                                P           <= P xor PS2_data;
                                state       <= D6;

                when D6 =>
                                PS2_code(6) <= PS2_data;
                                P           <= P xor PS2_data;
                                state    <= D7;

                when D7 =>
                                PS2_code(7) <= PS2_data;
                                P           <= P xor PS2_data;
                                state       <= parity;

                when parity =>
                                P        <= P xor PS2_data;
                                state    <= stop;

                when stop =>
                                PS2_code_new <= (Ps2_data) and (not p);
```

- کتابخانه‌ها و موجودیت: این کد از کتابخانه‌های استاندارد IEEE برای انواع منطقی استفاده می‌کند و موجودیتی با دو ورودی PS2_CLK) و (PS2_data و دو خروجی PS2_code_new) و (PS2_code تعریف شده است.
- تعریف ماشین حالت: نوع داده state_t یک نوع شمارشی است که حالت‌های مختلف ماشین حالت را تعریف می‌کند، از جمله start) شروع(، D0 تا D7) برای خواندن ۸ بیت داده(، parity(برای بررسی بیت توازن)، و stop)پایان فریم.(

منطق اصلی

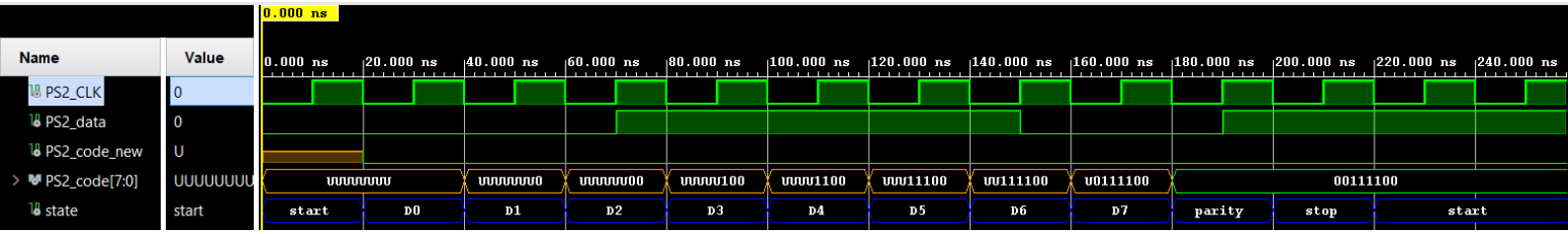
- سیگنال‌ها:
  - state حالت فعلی ماشین حالت را نگه می‌دارد و در ابتدا روی start تنظیم می‌شود.
  - P یک سیگنال XOR است که برای بررسی بیت توازن استفاده می‌شود و در ابتدا صفر است.
- بلوک پردازش:(Process Block)
  - این بلوک روی لبه نزولی سیگنال PS2_CLK اجرا می‌شود. وقتی یک رویداد (لبه نزولی) روی سیگنال کلاک تشخیص داده می‌شود، ماشین حالت فعال می‌شود.
  - در حالت start ، اگر PS2_data صفر باشد، فرآیند دریافت داده شروع می‌شود و ماشین حالت به D0 منتقل می‌شود. در غیر این صورت، در حالت start باقی می‌ماند.
  - حالت‌های D0 تا D7 هرکدام یک بیت داده از PS2_data می‌خوانند و در بردار PS2_code ذخیره می‌کنند. در هر مرحله، بیت خوانده شده با سیگنال P XOR می‌شود تا برای بررسی توازن استفاده شود.
  - در حالت parity ، بیت توازن پردازش و با P مقایسه می‌شود.
  - در حالت stop ، مقدار PS2_code_new بر اساس نتیجه XOR داده و توازن تنظیم می‌شود و ماشین حالت به start برمی‌گردد تا برای دریافت فریم جدید آماده شود.

توضیح عملکرد

این کد به طور مؤثر فریم‌های داده را که از دستگاه PS/2 دریافت می‌شوند، پردازش می‌کند و کد ۸ بیتی را پس از بررسی بیت توازن تولید می‌کند. بیت توازن به منظور اطمینان از صحت داده‌ها محاسبه و بررسی می‌شود.

```
                        state       <= start;

                when others =>
                        state <= start;

                end case;
            end if;
        end process;

end architecture;
```

مرحله ۱-۲: تست بنچ نویسی



بخش دوم: خواندن از فایل برای نمایش در سون سگمنت

مرحله ۱-۲:

این کد VHDL به منظور نمایش یک عدد دو رقمی روی دو سون سگمنت که یک باس داده مشترک دارند، طراحی شده است. برخلاف روش مستقیم وارد کردن مقادیر ۸ بیتی معادل سون سگمنت، در این آزمایش باید ابتدا مقادیر معادل ۸ بیتی برای ارقام ۰ تا F (۱۶ رقم) را در یک فایل متنی ذخیره کنید و سپس آن‌ها را با استفاده از یک زیر برنامه بخوانید و در یک حافظه ۱۶ خانه‌ای ذخیره کنید.

```
   library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_textio.ALL;
use std.textio.all;

entity Ex_4_2 is

    Port (CLK  : in     STD_LOGIC;
          Din  : in     STD_LOGIC_VECTOR(7 downto 0);
          COM  : buffer STD_LOGIC_VECTOR(3 downto 0) := "0001";
          Dout : out    STD_LOGIC_VECTOR(7 downto 0));

end entity;


architecture Behavioral of Ex_4_2 is

    signal temp     : STD_LOGIC_VECTOR(3 downto 0);
    type memory_t is array (0 to 15) of STD_LOGIC_VECTOR(7 downto 0);
    impure function init_RAM (input : string) return memory_t is

        file my_file          : text open read_mode is input;
        variable current_line : line;
        variable RAM          : memory_t;
        variable OK           : boolean;

    begin

        for i in memory_t'range loop
            readline(my_file, current_line);
            hread(current_line, RAM(i),OK);
            assert OK
            report "Read 'RAM' failed for line: " & current_line.all
            severity failure;
        end loop;
```

توضیح ساختار کد

موجودیت:(Entity)

CLK: ورودی کلاک که برای مدیریت زمان‌بندی تغییرات استفاده می‌شود.

Din: ورودی ۸ بیتی شامل دو عدد ٤ بیتی برای ارقام یکان و دهگان.

COM: خروجی ٤ بیتی که به صورت تناوبی مشخص می‌کند کدام سون سگمنت فعال باشد (دهگان یا یکان).

Dout: خروجی ۸ بیتی که داده سون سگمنت را ارسال می‌کند.

```
    file_close(my_file);
        return RAM;

end function;

    signal RAM : memory_t := init_RAM("C:\Users\Ali\Desktop\Ex4_2\7seg.txt");

begin

    process(CLK)

    begin

        if (CLK'event and CLK = '1') then
            if (COM = "0010") then
                COM  <= "0001";
                temp <= Din(3 downto 0);
            elsif (COM = "0001") then
                COM  <= "0010";
                temp <= Din(7 downto 4);
            end if;
        end if;

    end process;

    Dout <= RAM(conv_integer(temp));

end architecture;
```

تعریف حافظه:

یک نوع آرایه به نام memory_t تعریف شده که ۱۶ خانه دارد و هر خانه ۸ بیت است. این آرایه برای ذخیره مقادیر معادل سون سگمنت اعداد ۰ تا F استفاده می‌شود.

یک تابع با نام init_RAM برای خواندن مقادیر از فایل متنی seg.txt ۷نوشته شده است. این تابع به صورت impure تعریف شده است، زیرا ورودی از دنیای خارج (فایل متنی) می‌گیرد.

درون این تابع، مقادیر خط به خط از فایل متنی خوانده شده و در حافظه RAM ذخیره می‌شوند.

بلوک پردازش:(Process Block)

در هر لبه بالا رونده کلاک، مقدار COM به صورت تناوبی بین "۰۰۰۱" (برای فعال‌سازی یکان) و "۰۰۱۰" (برای فعال‌سازی دهگان) تغییر می‌کند.

بسته به مقدارCOM ، قسمت مربوط به یکان یا دهگان از Din انتخاب و در temp ذخیره می‌شود.

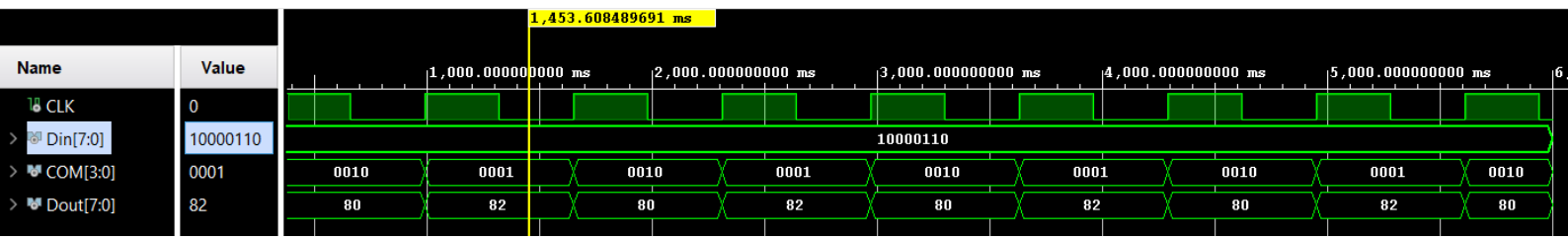
تخصیص خروجی:

خروجی Dout مقدار متناظر با temp را از حافظه RAM می‌خواند و روی سون سگمنت ارسال می‌کند. مقدار temp به عدد صحیح تبدیل می‌شود تا به عنوان شاخص برای دسترسی به آرایه RAM استفاده شود.

عملکرد کلی

این کد با استفاده از یک فایل متنی مقادیر معادل سون سگمنت را می‌خواند و در یک حافظه ذخیره می‌کند. در هر چرخه کلاک، یکی از دو سون سگمنت فعال شده و مقدار صحیح برای نمایش عدد مربوطه از حافظه خوانده می‌شود. به این ترتیب، یک عدد دو رقمی به طور صحیح و مرتب روی دو سون سگمنت نمایش داده می‌شود.

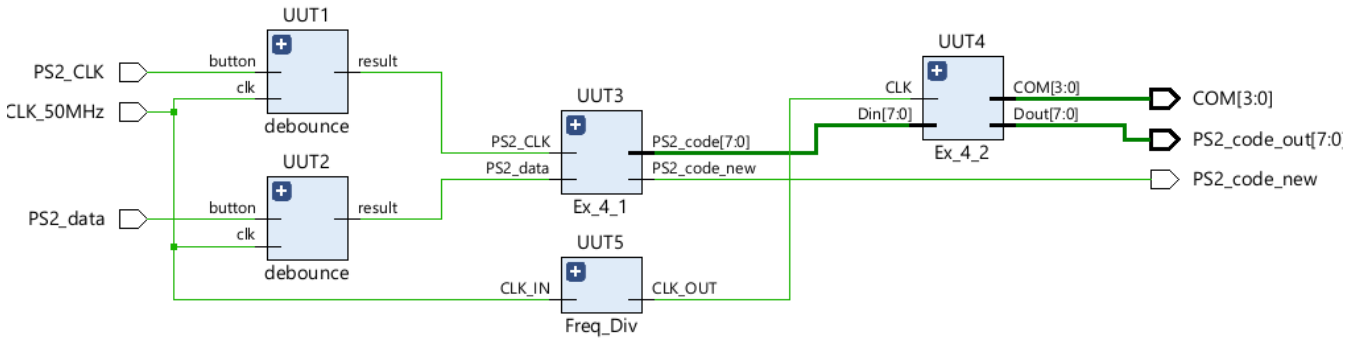
مرحله ۲-۲: تست بنچ نویسی



بخش سوم: ساختاری

مرحله ۱-۳: کد ساختاری

```
 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Ex_4_3 is
    port(CLK_50MHz, PS2_CLK, PS2_data : in  std_logic;
         PS2_code_out                 : out std_logic_vector(7 downto 0);
         COM                          : out std_logic_vector(3 downto 0);
         PS2_code_new                 : out std_logic);
end Ex_4_3;

architecture Structural of Ex_4_3 is

    signal PS2_CLK_Debounced, PS2_data_Debounced : std_logic;
    signal PS2_code : std_logic_vector(7 downto 0);
    signal div : std_logic;


begin

    UUT1: entity work.debounce port map(CLK_50MHz, PS2_CLK, PS2_CLK_Debounced);
    UUT2: entity work.debounce port map(CLK_50MHz, PS2_data, PS2_data_Debounced);
    UUT3: entity work.Ex_4_1 port map(PS2_CLK_Debounced, PS2_data_Debounced, PS2_code_new, PS2_code);
    UUT4: entity work.EX_4_2 port map(div, PS2_code, COM, PS2_code_out);
    UUT5: entity work.Freq_Div generic map (25 * (10**6)) port map(CLK_50MHz, div);

end Structural;
```



مرحله ۲-۳: کد UCF

```
NET "COM[3]" LOC = p130 ;
NET "COM[2]" LOC = p128 ;
NET "COM[1]" LOC = p126 ;
NET "COM[0]" LOC = p125 ;

NET "PS2_code_out[7]" LOC = p131 ;
NET "PS2_code_out[6]" LOC = p132 ;
NET "PS2_code_out[5]" LOC = p133 ;
NET "PS2_code_out[4]" LOC = p135 ;
NET "PS2_code_out[3]" LOC = p137 ;
NET "PS2_code_out[2]" LOC = p138 ;
NET "PS2_code_out[1]" LOC = p139 ;
NET "PS2_code_out[0]" LOC = p140 ;

NET "CLK_50MHz" LOC = p80;
NET "PS2_CLK" LOC = p64;
NET "PS2_data" LOC = p63;
NET "PS2_code_new" LOC = p93;
```