Pre-Report

FPGA Lab Eperiment 7

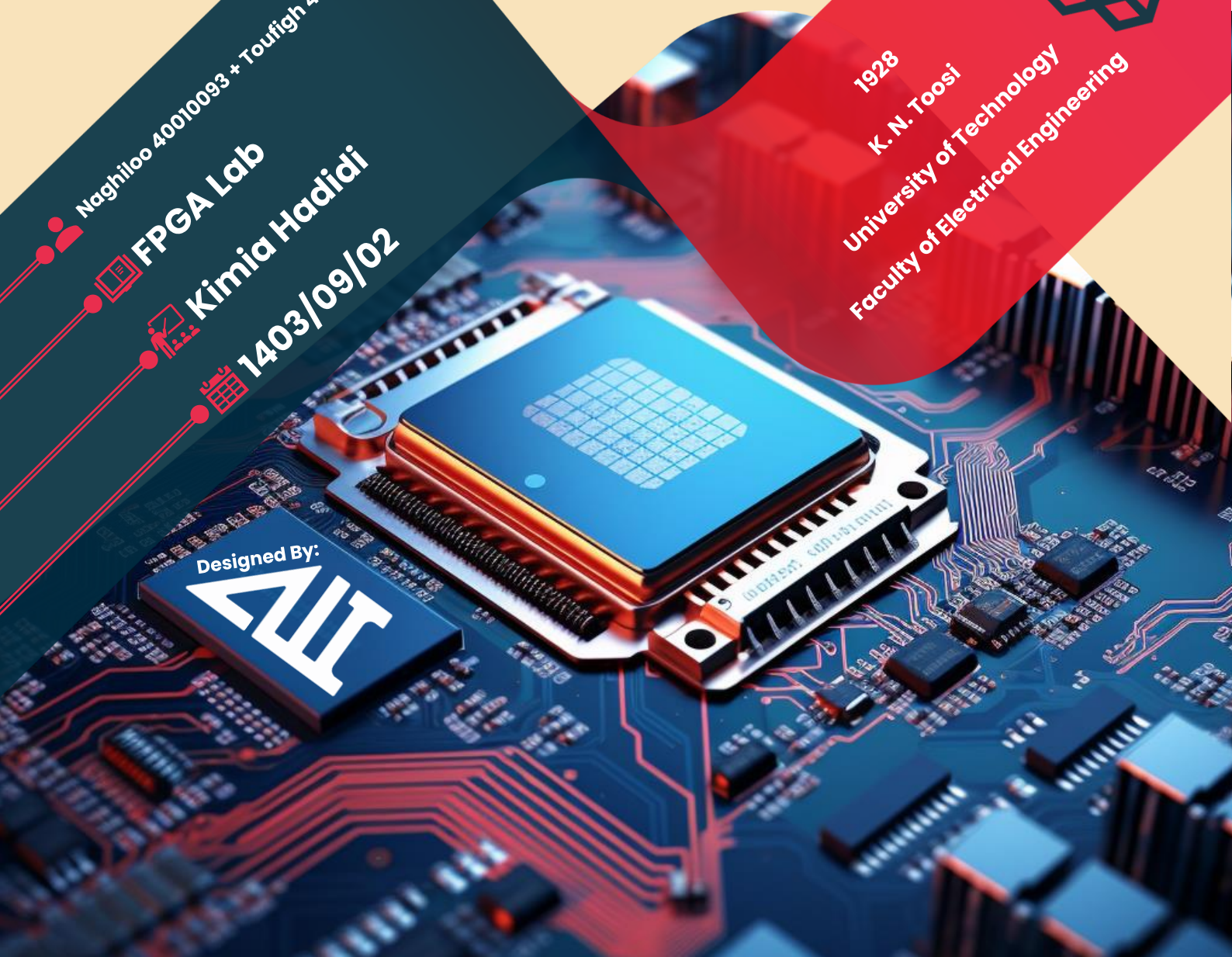
Naghiloo 40010093 + Toufigh 40003913

FPGA Lab

Kimia Hadidi

1403/09/02

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



۱- بله، هر جا کلیدی است که ممکن است بانس بوجود بیاورد، باید دیبانسر استفاده شود.

۲-

```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6
7  entity MatrixKey is
8      port ( Column  : in  std_logic_vector(1 to 4) := (others => '1');
9             CLK     : in  std_logic;
10            Row     : out std_logic_vector(1 to 4)     := (others =>
   '0');
11            Hit     : out std_logic                    := ('0');
12            Output  : out std_logic_vector(3 downto 0) := (others =>
   '0'));
13 end MatrixKey;
14
15 architecture BHV of MatrixKey is
16     type state_t is (Check, Row1, Row2, Row3, Row4);
17     signal State : state_t := Check;
18 begin
19     STATE_MEMORY: process(CLK)
20     begin
21         if (CLK'event and CLK='1') then
22             case State is
23             when Check =>
24                 if (Column="1111") then
25                     Hit <= '0'; State <= Check;
26                 else
27                     Row <= (1 => '0', others => 'Z'); State <= Row1;

28                 end if;
29             when Row1  =>
30                 if (Column="1111") then
31                     Row <= (2 => '0', others => 'Z'); State <= Row2;
32                 else
33                     Hit <= '1'; Row <= (others => '0'); State <= Check;
34                     if    (Column(1)='0') then Output <= x"0";
35                     elsif (Column(2)='0') then Output <= x"1";
36                     elsif (Column(3)='0') then Output <= x"2";
37                     else                       Output <= x"3";
38                     end if;
39                 end if;
40             when Row2  =>
41                 if (Column="1111") then
42                     Row <= (3 => '0', others => 'Z'); State <= Row3;
43                 else
44                     Hit <= '1'; Row <= (others => '0'); State <= Check;
45                     if    (Column(1)='0') then Output <= x"4";
46                     elsif (Column(2)='0') then Output <= x"5";
47                     elsif (Column(3)='0') then Output <= x"6";
48                     else                       Output <= x"7";
49                     end if;
50                 end if;
51             when Row3  =>
52                 if (Column="1111") then
53                     Row <= (4 => '0', others => 'Z'); State <= Row4;
54                 else
55                     Hit <= '1'; Row <= (others => '0'); State <= Check;
56                     if    (Column(1)='0') then Output <= x"8";
57                     elsif (Column(2)='0') then Output <= x"9";
58                     elsif (Column(3)='0') then Output <= x"A";
59                     else                       Output <= x"B";
60                     end if;
61                 end if;
62             when Row4  =>
63                     Hit <= '1'; Row <= (others => '0'); State <= Check;
64                     if    (Column(1)='0') then Output <= x"C";
65                     elsif (Column(2)='0') then Output <= x"D";
66                     elsif (Column(3)='0') then Output <= x"E";
67                     else                       Output <= x"F";
68                     end if;
69             when others =>  State <= Check;
70         end case;
71         end if;
72     end process STATE_MEMORY;
73 end architecture BHV;
```

این کد VHDL برای یک سیستم ماتریس کلیدی (Matrix Keypad) نوشته شده است که برای خواندن ورودی‌های یک صفحه کلید ماتریسی طراحی شده است. در اینجا، عملکرد اصلی به این صورت است که با استفاده از یک ماشین حالت(state machine) ، خطوط سطر (Row) به ترتیب فعال می‌شوند و وضعیت خطوط ستون (Column) بررسی می‌شود تا کلید فشرده‌شده شناسایی شود.

توضیح عملکرد:

۱. تعاریف اولیه:

- ورودی‌ها شامل Column(برای خواندن وضعیت ستون‌ها) و CLK(سیگنال کلاک) هستند.
- خروجی‌ها شامل) Rowبرای فعال‌سازی سطرها(، Hit(برای نشان دادن فشردن یک کلید)، و Output(برای نمایش مقدار کلید فشرده‌شده به صورت عدد هگزادسیمال) هستند.

2. ماشین حالت (State Machine):

- پنج حالت برای عملکرد سیستم تعریف شده است :
  - Check:بررسی می‌کند که آیا هیچ کلیدی فشرده نشده است .(Column = "1111") در صورت عدم فشردن کلید، سیستم در همین حالت باقی می‌ماند.
  - Row1، Row2، Row3، Row4:به ترتیب خطوط سطرها را فعال می‌کنند و ستون‌ها را بررسی می‌کنند. اگر کلیدی فشرده شود، مقدار هگزادسیمال مربوطه در خروجی Outputقرار می‌گیرد و سیستم به حالت Checkبازمی‌گردد.

۳. فرآیند اصلی:

- در هر لبه بالا رونده کلاک، حالت فعلی سیستم بررسی می‌شود.
- در هر سطر فعال، اگر مقدار ستون‌ها نشان‌دهنده فشردن یک کلید باشد (یکی از بیت‌های ستون برابر '۰' باشد)، مقدار کلید شناسایی می‌شود :

- مقادیر هگزادسیمال از 0تا Fبسته به موقعیت سطر و ستون تعیین می‌شوند.
  - اگر هیچ کلیدی شناسایی نشود، سطر بعدی فعال می‌شود.

۴. مدیریت خروجی‌ها:

- خروجی Hitزمانی که یک کلید فشرده شود، به '1' تغییر می‌کند.
- خروجی Outputمقدار کلید را به صورت هگزادسیمال ذخیره می‌کند.
- خروجی Rowنیز بسته به سطر فعال تغییر می‌کند.

این کد به طور کارآمد کلید فشرده‌شده در یک ماتریس کلیدی را شناسایی کرده و مقدار آن را در خروجی ارائه می‌دهد. از ماشین حالت برای ساده‌سازی منطق و کنترل دقیق زمان‌بندی استفاده شده است.

(بقیه موارد صفحه بعد)

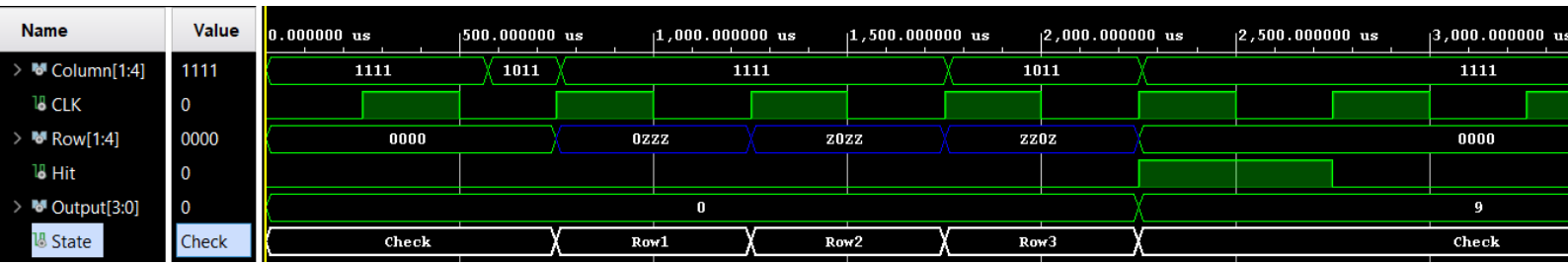
۳- نیاز به debouncer و Seven_Segment و Freq_Div داریم که همگی مشابه قبل هستند ولی طراحی Seven_Segment کمی تغییر کرده که کد آن را در زیر مشاهده میکنید:

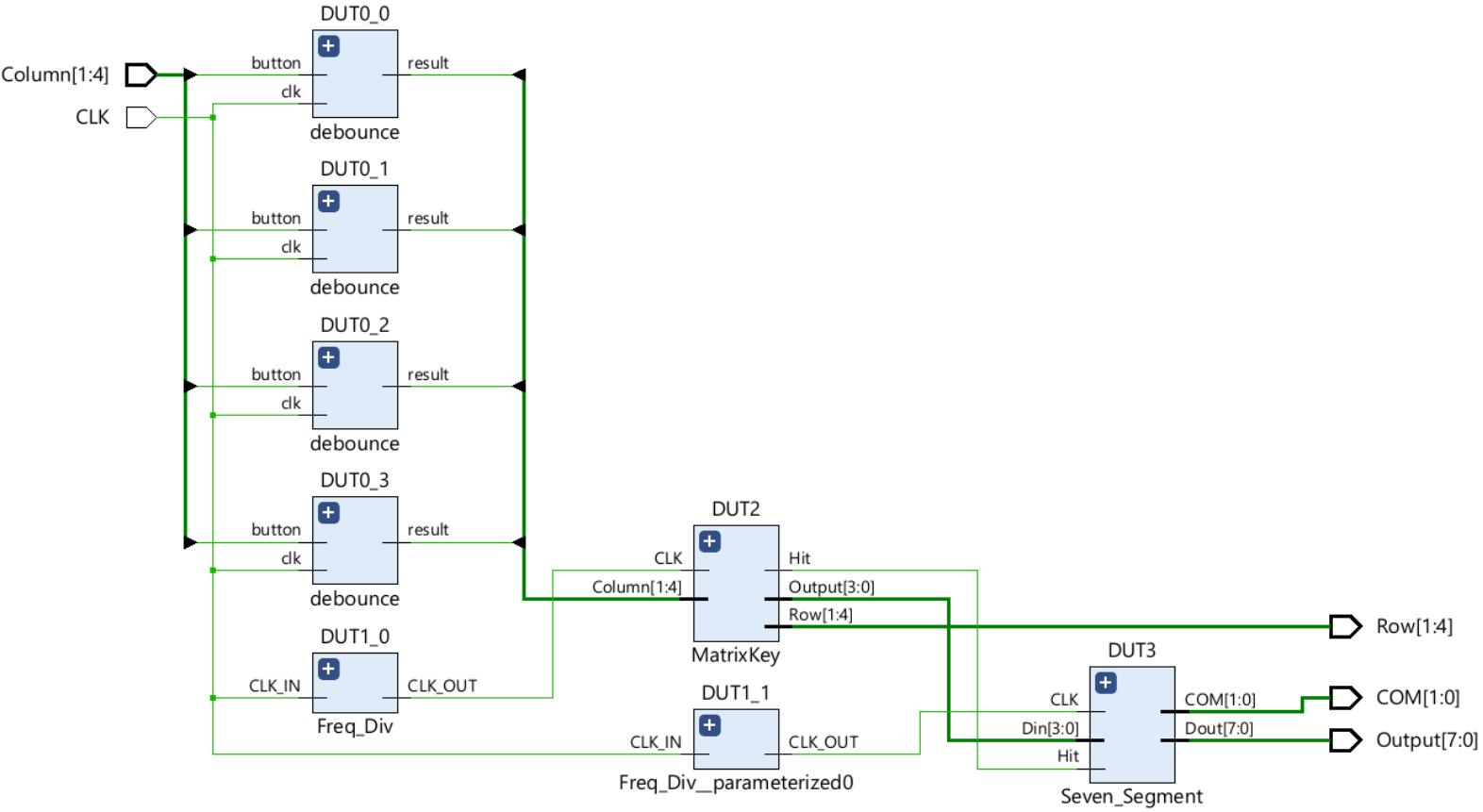
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;

entity Seven_Segment is

   Port(Din  : in  STD_LOGIC_VECTOR (3 downto 0);
        CLK, Hit  : in  STD_LOGIC;
        COM  : buffer STD_LOGIC_VECTOR(1 downto 0) := "01";
        Dout : out STD_LOGIC_VECTOR (7 downto 0));

end Seven_Segment;

architecture BHV of Seven_Segment is
    signal temp_Byte   : STD_LOGIC_VECTOR(7 downto 0);
    signal temp_Nibble : STD_LOGIC_VECTOR(3 downto 0);
    signal temp_Dout   : STD_LOGIC_VECTOR (7 downto 0);

    type memory1_t is array (0 to 15) of STD_LOGIC_VECTOR(7 downto 0);
    signal BIN2BCD : memory1_t := (
    b"0000_0000", -- BCD for 0
    b"0000_0001", -- BCD for 1
    b"0000_0010", -- BCD for 2
    b"0000_0011", -- BCD for 3
    b"0000_0100", -- BCD for 4
    b"0000_0101", -- BCD for 5
    b"0000_0110", -- BCD for 6
    b"0000_0111", -- BCD for 7
    b"0000_1000", -- BCD for 8
    b"0000_1001", -- BCD for 9
    b"0001_0000", -- BCD for 10
    b"0001_0001", -- BCD for 11
    b"0001_0010", -- BCD for 12
    b"0001_0011", -- BCD for 13
    b"0001_0100", -- BCD for 14
    b"0001_0101"  -- BCD for 15
    );

    type memory2_t is array (0 to 9) of STD_LOGIC_VECTOR(7 downto 0);
    signal BCD2Segments : memory2_t := (
    "11000000", -- BCD for 0
    "11111001", -- BCD for 1
    "10100100", -- BCD for 2
    "10110000", -- BCD for 3
    "10011001", -- BCD for 4
    "10010010", -- BCD for 5
    "10000010", -- BCD for 6
    "11111000", -- BCD for 7
    "10000000", -- BCD for 8
    "10010000"  -- BCD for 9
);

begin

    temp_Byte <= BIN2BCD(conv_integer(Din));

    process(CLK)
    begin
        if (CLK'event and CLK = '1') then
            if (COM = "10" or COM = "00") then
                COM  <= "01";
                temp_Nibble <= temp_Byte(3 downto 0);
            elsif (COM = "01") then
                if (Din > 9) then
                    COM <= "10";
                else
                    COM <= "00";
                end if;
                temp_Nibble <= temp_Byte(7 downto 4);
            end if;
        end if;
    end process;

    temp_Dout <= BCD2Segments(conv_integer(temp_Nibble));

    Dout <= "10111111" when Hit='0' else temp_Dout;

end BHV;
```

۴-

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

entity main is
    Port ( CLK : in STD_LOGIC;
           Column : in STD_LOGIC_VECTOR (1 to 4);
           Row : out STD_LOGIC_VECTOR (1 to 4);
           Output : out STD_LOGIC_VECTOR (7 downto 0);
           COM : out STD_LOGIC_VECTOR (1 downto 0));
end main;

architecture Structural of main is
    signal C : STD_LOGIC_VECTOR (1 to 4);
    signal CLK_MK : STD_LOGIC;
    signal CLK_SS : STD_LOGIC;
    signal Hit : STD_LOGIC;
    signal O : STD_LOGIC_VECTOR (3 downto 0);

begin

    DUT0_0: entity work.debounce port map(CLK, Column(1), C(1));
    DUT0_1: entity work.debounce port map(CLK, Column(2), C(2));
    DUT0_2: entity work.debounce port map(CLK, Column(3), C(3));
    DUT0_3: entity work.debounce port map(CLK, Column(4), C(4));

    DUT1_0: entity work.Freq_Div generic map(12500) port map(CLK, CLK_MK);
    DUT1_1: entity work.Freq_Div generic map(25*(10**3)) port map(CLK, CLK_SS);

    DUT2: entity work.MatrixKey port map(C, CLK_MK, Row, Hit, O);
    DUT3: entity work.Seven_Segment port map(O, CLK_SS, Hit, COM, Output);

end Structural;
```

مرحله بعد (تست بنچ نویسی):






```
NET "CLK" LOC = P80;

NET "Column[1]" LOC = P19;
NET "Column[2]" LOC = P20;
NET "Column[3]" LOC = P21;
NET "Column[4]" LOC = P22;

NET "Row[1]" LOC = P18;
NET "Row[2]" LOC = P16;
NET "Row[3]" LOC = P15;
NET "Row[4]" LOC = P13;

NET "COM[0]" LOC = P125;
NET "COM[0]" LOC = P126;

NET "Output[7]" LOC = P131;
NET "Output[6]" LOC = P132;
NET "Output[5]" LOC = P133;
NET "Output[4]" LOC = P135;
NET "Output[3]" LOC = P137;
NET "Output[2]" LOC = P138;
NET "Output[1]" LOC = P139;
NET "Output[0]" LOC = P140;
```