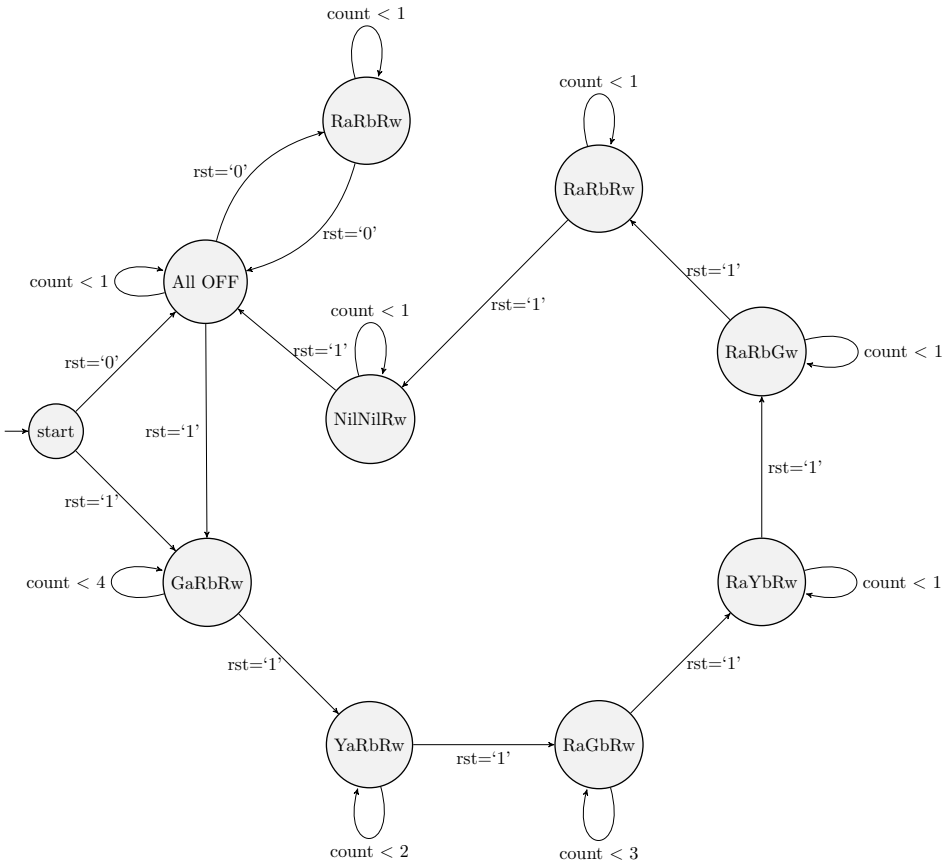
تمرین چراغ راهنمایی و رانندگی

فاطمه علی‌ملکی
مهدی حق‌وردی

۶ دی ۱۴۰۲

فهرست مطالب

۱ دیاگرام



در این دیاگرام، وضعیت‌های ماشین کشیده شده‌اند. یال‌های حلقه روی وضعیت‌ها میزان زمان ایستادن روی آن وضعیت هستند.

۲ توضیحات

در این قسمت توضیحاتی راجع به کدها می‌دهیم.

۱.۲ `clock_divide.vhdl`

که قبلا توضیح داده شده و تعداد کلاک 100 MHz را به عدد 1 MHz تبدیل می‌کند.

۲.۲ trfc.ucf

با مراجعه به توضیحات سایت تولید کننده‌ی برد این فایل را نوشتیم.

```
NET "clk_100MHz" LOC = P50;
NET "rst" LOC = P51;
NET "LEDS[0]" LOC = P132;
NET "LEDS[1]" LOC = P131;
NET "LEDS[2]" LOC = P127;
NET "LEDS[3]" LOC = P126;
NET "LEDS[4]" LOC = P124;
NET "LEDS[5]" LOC = P123;
NET "LEDS[6]" LOC = P121;
NET "LEDS[7]" LOC = P120;
```

۳.۲ traffic.vhdl

۱.۳.۲ موجودیت traffic

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_unsigned.all;

entity traffic is
    port (
        clk_100MHZ: in std_logic;
        rst: in std_logic;
        LEDS: out std_logic_vector(7 downto 0)
    );
end traffic;
```

- clk_100MHZ: کلاک ورودی از برد.
- rst: دکمه‌ی K1
- LEDS: چراغ‌های LED

۲.۳.۲ معماری

```
12 architecture traffic of traffic is
13     component clock_divide
14         port (
15             clk_in: in std_logic;
16             clk_out: out std_logic
17         );
18     end component;
19 
20     -- clock
21     signal clk_1Hz: std_logic;
22 
23     -- states
24     type state_type is (s0, s1, s2, s3, s4, s5, s6, s7);
25     signal state: state_type;
26 
27     -- state counter
28     signal count: std_logic_vector(2 downto 0);
29 
30     -- helper counter for s7
31     signal hcount: std_logic_vector(0 downto 0);
32 
33 begin
34 clock_divider: clock_divide port map (clk_100MHZ, clk_1Hz);
```

ابتدا از موجودیت `clock_divider` یک کامپوننت تعریف می‌کنیم تا بتوانیم از آن استفاده کنیم، سپس ۳ سیگنال تعریف می‌کنیم:

- `clk_1Hz`

  این سیگنال، مقدار خروجی کامپوننت `clock_divider` را دریافت می‌کند.

- `state`

  این سیگنال از نوع `state_type` است که بالای سیگنال تعریف شده و مقادیر stateهای موجود در دیاگرام (۱) را می‌تواند داشته باشد.

- `count`

  سیگنالی برای شماره ثانیه‌ها؛ به کمک این سیگنال بین stateها جابجا می‌شویم.

۳.۳.۲ اولین فرآیند

```
process(clk_1Hz, rst)
-- needed variables
variable one_sec : std_logic_vector(2 downto 0) := '001';
variable two_sec : std_logic_vector(2 downto 0) := '010';
variable three_sec : std_logic_vector(2 downto 0) := '011';
variable four_sec : std_logic_vector(2 downto 0) := '100';

begin
    if rst = '1' then
        if clk_1Hz'event and clk_1Hz = '1' then
            case state is
                when s0 =>
                    if count < four_sec then
                        state <= s0;
                        count <= count + 1;
                    else
                        state <= s1;
                        count <= "000";
                    end if;
                when s1 =>
                    if count < two_sec then
                        state <= s1;
                        count <= count + 1;
                    else
                        state <= s2;
                        count <= "000";
                    end if;
                when s2 =>
                    if count < three_sec then
                        state <= s2;
                        count <= count + 1;
                    else
                        state <= s3;
                        count <= "000";
                    end if;
                when s3 =>
                    if count < one_sec then
                        state <= s3;
                        count <= count + 1;
                    else
                        state <= s4;
```

```
70                      count <= "000";
71                  end if;
72              when s4 =>
73                  if count < two_sec then
74                      state <= s4;
75                      count <= count + 1;
76                  else
77                      state <= s5;
78                      count <= "000";
79                  end if;
80              when s5 =>
81                  if count < one_sec then
82                      state <= s5;
83                      count <= count + 1;
84                      hcount <= "0";
85                  else
86                      state <= s7;
87                      count <= "000";
88                  end if;
89              when s7 =>
90                  if count < one_sec then
91                      state <= s7;
92                      count <= count + 1;
93                  else
94                      state <= s6;
95                      count <= "000";
96                  end if;
97              when s6 =>
98                  if (count < one_sec) and (hcount > one_sec)
                        then
99                      state <= s6;
100                     count <= count + 1;
101                 elsif hcount < one_sec then
102                     state <= s7;
103                     hcount <= hcount + 1;
104                     count <= "000";
105                 else
106                     state <= s0;
107                     count <= "000";
108                 end if;
109             when others =>
110                 state <= s0;
111                 count <= "000";
112         end case;
```

```
113         end if;
```

این فرآیند، به کلاک ۱ هرتز و کلید rst حساس است. سپس به اندازه‌ی مقدار ثانیه‌هایی که در فایل پروژه نوشته شده است، متغیرهایی تعریف می‌کنیم تا از آنها استفاده کنیم.

این فرآیند یک بلاک if بزرگ دارد که از بین ۰ و ۱ (مقادیر rst) انتخاب می‌کند؛ اگر مقدار آن ۱ باشد، وارد حلقه‌ی بزرگ‌تر دیاگرام (۱) می‌شویم.

از بین stateهای s0 تا s6 با توجه به زمانی که برای هر کدام تعریف شده، چرخش انجام می‌دهیم.

این فرآیند، وقتی که مقدار state را تغییر می‌دهد، فرآیند دیگر را (۵.۳.۲) بیدار می‌کند که در ادامه به توضیح آن می‌پردازیم.

۴.۳.۲ قسمت else

```
101     else -- rst = '0'
102         if clk_1Hz'event and clk_1Hz = '1' then
103             if state /= s5 and state /= s6 then
104                 state <= s5;
105                 count <= "000";
106             end if;
107             case state is
108                 when s5 =>
109                     if count < one_sec then
110                         state <= s5;
111                         count <= count + 1;
112                     else
113                         state <= s6;
114                         count <= "000";
115                     end if;
116                 when s6 =>
117                     if count < one_sec then
118                         state <= s6;
119                         count <= count + 1;
120                     else
121                         state <= s5;
122                         count <= "000";
123                     end if;
124                 when others =>
125                     state <= s5;
126                     count <= "000";
127             end case;
128         end if;
129     end if;
130 end process;
```

در این قسمت، اگر مقدار rst صفر باشد، ابتدا چک می‌کنیم که آیا در state ۵ یا ۶ نیستیم به state ۵ برویم. سپس با توجه به دیاگرام، بین این دو state می‌چرخیم، تا رفتار خواسته شده نمایش داده شود.

RaRbRw[۱]
All OFF[۲]

۵.۳.۲ فرآیند دوم

```
131 process(state)
132 begin
133 case state is
134        -- Red Yellow Green, Red Yellow Green, Red Green
135        when s0 => LEDS <= "00110010"; -- Ga=1, Rb=1, Rw=1
136        when s1 => LEDS <= "01010010"; -- Ya=1, Rb=1, Rw =1
137        when s2 => LEDS <= "10000110"; -- Ra=1, Gb=1, Rw=1
138        when s3 => LEDS <= "10001010"; -- Ra=1, Yb=1, Rw=1
139        when s4 => LEDS <= "10010001"; -- Ra=1, Rb=1, Gw=1
140        when s5 => LEDS <= "10010010"; -- Ra=1, Rb=1, Rw=1
141        when s6 => LEDS <= "00000000"; -- All LEDs are off
142        when s7 => LEDS <= "00000010"; -- Ra=0, Rb=0, Rw=1
143        when others => LEDS <= "00110010"; -- Ga=1, Rb=1, Rw=1
144 end case;
145 end process;
146 end traffic;
```

این فرآیند، به مقدار state حساس است و با توجه به مقدار state چراغ‌های لازم را روشن می‌کند.