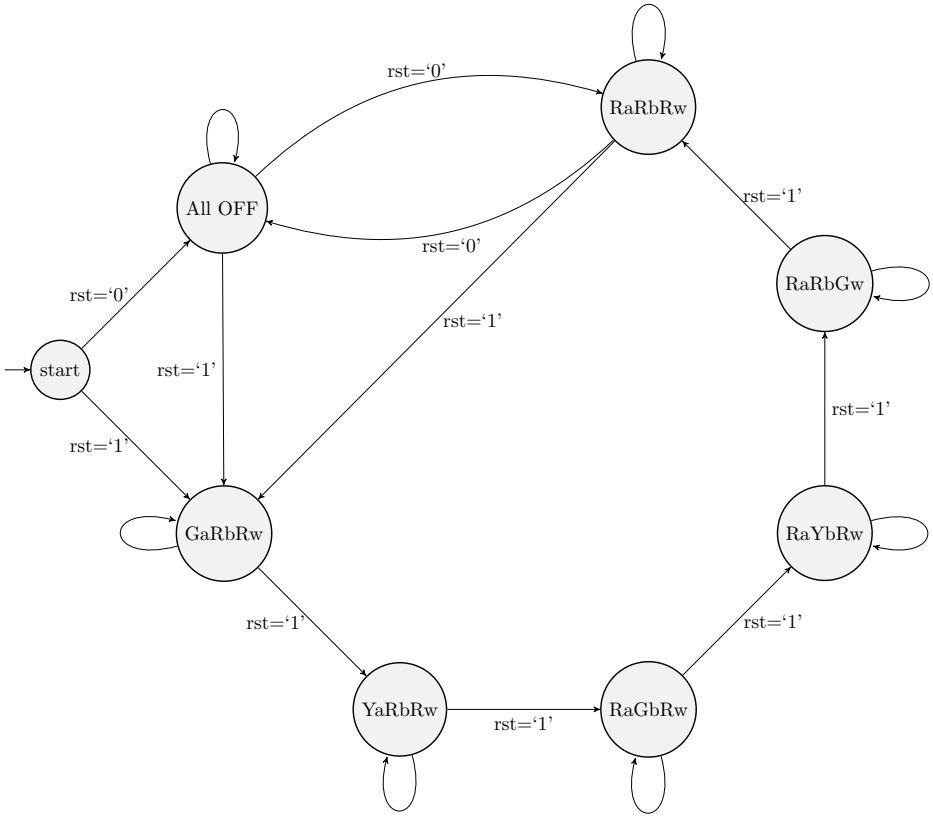
تمرین چراغ راهنمایی و رانندگی

فاطمه علی‌ملکی
مهدی حق‌وردی

۵ دی ۱۴۰۲

فهرست مطالب

۱ دیاگرام



در این دیاگرام، وضعیت‌های ماشین کشیده شده‌اند. یال‌های حلقه روی وضعیت‌ها میزان زمان ایستادن روی آن وضعیت هستند که وقتی در کلاس آزمایش انجام شد و اعداد آنها بدست آمد، نوشته می‌شوند.

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

۳.۲ traffic.vhdl

۱.۳.۲ موجودیت traffic

```vhdl
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

```
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);
25     signal state: state_type;
26 
27     -- state counter
28     signal count: std_logic_vector(2 downto 0);
29 
30 begin
31 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.

```
29 process(clk_1Hz, rst)
30 -- needed variables
31 variable one_sec : std_logic_vector(2 downto 0) := '001';
32 variable two_sec : std_logic_vector(2 downto 0) := '010';
33 variable three_sec : std_logic_vector(2 downto 0) := '011';
34 variable four_sec : std_logic_vector(2 downto 0) := '100';
35
36 begin
37     if rst = '1' then
38         if clk_1Hz'event and clk_1Hz = '1' then
39             case state is
40                 when s0 =>
41                     if count < four_sec then
42                         state <= s0;
43                         count <= count + 1;
44                     else
45                         state <= s1;
46                         count <= "000";
47                     end if;
48                 when s1 =>
49                     if count < two_sec then
50                         state <= s1;
51                         count <= count + 1;
52                     else
53                         state <= s2;
54                         count <= "000";
55                     end if;
56                 when s2 =>
57                     if count < three_sec then
58                         state <= s2;
59                         count <= count + 1;
60                     else
61                         state <= s3;
62                         count <= "000";
63                     end if;
64                 when s3 =>
65                     if count < one_sec then
66                         state <= s3;
67                         count <= count + 1;
68                     else
69                         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                  else
85                      state <= s6;
86                      count <= "000";
87                  end if;
88              when s6 =>
89                  if count < one_sec then
90                      state <= s6;
91                      count <= count + 1;
92                  else
93                      state <= s0;
94                      count <= "000";
95                  end if;
96              when others =>
97                  state <= s0;
98                  count <= "000";
99          end case;
100     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 بیدار می‌کند (فرآیند ؟؟) که در ادامه به توضیح آن می‌پردازیم.