M2 Master E3A Integration Circuits Systems

TD4: Realization of Finite State Machine

1. IFTAKHER Mohammed Akib – No. 22211204

2. KIBIWOTT Albert Kiplagat - No. 22211612

Instructor: Professor Hervé MATHIAS

Realization of Finite State Machine

The aim of this practical session is to describe the sequencing of a traffic light for pedestrians. Here, a single traffic light is considered and not crossroads. Below are the codes:

Part 1: State Machine with 5 states

Clock Divider Code:

```
1
2   -- open the standard libraries
3   library ieee;
4   use ieee.std_logic_1164.all ; -- définit le types std_logic std_logic_vector
5
6   -- the TOP level entity
7   entity clk_div is
8
9   -- defines the generic parameters (project constants)
10  generic (factor: integer := 25000000);  -- facteur de division d'horloge. Ici on
    obtiendra 1 hz à partir de 50 Mhz
11
12  -- Input/Outputs definition
13  port (  RESET   :   in std_logic;      -- 50 Mhz Clock
14          CLK_IN  :   in std_logic;      -- 50 Mhz Clock
15          CLK_OUT :  out std_logic    -- 1 Hz clock
16          ) ;
17
18  end entity;
19
20
21  architecture a1 of clk_div is
22  signal clkcnt : integer := factor;
23  signal clkout : std_logic := '0';
24
25  begin
26     process(RESET, CLK_IN)
27        begin
28           if (RESET = '0') then
29              clkcnt <= factor;
30              clkout <= '0';
31
32           elsif (rising_edge(CLK_IN)) then
33              if (clkcnt = 0) then
34                 clkcnt <=  factor - 1;
35                 clkout <= not clkout;
36              else
37              clkcnt <= clkcnt - 1;
38              end if;
39           end if;
40     CLK_OUT <= clkout;
41     end process;
42
```

State Machine Code:

```
1   Library ieee ;
2   Use ieee.std_logic_1164.all ;
3
4   entity feux is
5   port (clk, raz : in  std_logic ;  -- horloge, reset, et bouton poussoir
6         r,o,v    : out std_logic ) ;       -- commande des lumières rouge, orange et vert
7   end entity feux ;
8
9   architecture a1 of feux is
10
11  type statetype is (S0, S1, S2, S3, S4, S5);     -- états de la machine d'état
12  signal state : statetype;                    -- registre d'état
13
14
15  begin
16
17    process(clk,raz)
18    begin
19      if (raz='0') then                      -- reset asynchrone
20          state <= S0 ;
21      elsif (clk'event and clk='1') then       -- au top d'horloge
22        case state is                        -- calcul du nouvel etat
23          when S0 => state <= S1 ;
24          when S1 => state <= S2 ;
25          when S2 => state <= S3 ;
26          when S3 => state <= S4 ;
27          when S4 => state <= S5 ;
28          when S5 => state <= S0 ;
29        end case ;
30      end if ;
31    end process;
32
33    -- calcul des sorties en fonction de l'état
34    r <= '1' when state=S0 or state=S1 or state=S2 else '0' ;
35    v <= '1' when state=S3 or state=S4 else '0' ;
36    o <= '1' when state=S5 else '0' ;
37  end a1;
```

Part 2: State Machine with 3 states

```
1   Library ieee ;
2   Use ieee.std_logic_1164.all ;
3
4   entity feux is
5   port (clk, raz : in  std_logic ;  -- horloge, reset, et bouton poussoir
6         r,o,v    : out std_logic ) ;       -- commande des lumières rouge, orange et vert
7   end entity feux ;
8
9   architecture a1 of feux is
10
11  type statetype is (State0, State1, State2);     -- états de la machine d'état
12  signal next_state, present_state : statetype;                  -- registre d'état
13
14
15  begin
16
17    process(clk,raz, present_state)
18    begin
19      if (raz='0') then                       -- reset asynchrone
20          present_state <= State0 ;
21      elsif (clk'event and clk='1') then       -- au top d'horloge
22        case present_state is                        -- calcul du nouvel etat
23          when State0 =>
24              next_state <= State1 ;
25              r <= '1';
26              v <= '0';
27              o <= '0';
28          when State1 =>
29              next_state <= State2 ;
30              r <= '0';
31              v <= '1';
32              o <= '0';
33          when State2 =>
34              next_state <= State0 ;
35              r <= '0';
36              v <= '0';
37              o <= '1';
38
39        end case ;
40        present_state <= next_state;
41      end if ;
42    end process;
43
44
45  end a1;
```

Part 3: State Machine with 3 states with duration configure

```
1   Library ieee ;
2   Use ieee.std_logic_1164.all ;
3
4   entity feux is
5   generic ( rcount : integer := 5;
6             ocount : integer := 3;
7             vcount : integer := 1);
8
9   port (clk, raz : in  std_logic ;  -- horloge, reset, et bouton poussoir
10        r,o,v    : out std_logic ) ;       -- commande des lumières rouge, orange et vert
11  end entity feux ;
12
13  architecture a1 of feux is
14
15  type statetype is (SR, SV, SO);     -- états de la machine d'état
16  signal current_state : statetype;
17  signal counter : integer := rcount;                    -- registre d'état
18
19
20  begin
21
22    process(clk,raz)
23    begin
24      if (raz='0') then                     -- reset asynchrone
25          current_state <= SR ;
26
27      elsif (clk'event and clk='1') then       -- au top d'horloge
28
29           case current_state is
30              when SR =>
31                 if (counter = 0) then
32                    current_state <= SV;
33                    counter <= vcount;
34
35                 else
36                    counter <= counter - 1;
37                 end if;
38
39              when SV =>
40                 if (counter = 0) then
41                    current_state <= SO;
42                    counter <= ocount;
43
44                 else
45                    counter <= counter - 1;
46                 end if;
47
48              when SO =>
49                 if (counter = 0) then
50                    current_state <= SR;
51                    counter <= rcount;
52
53                 else
54                    counter <= counter - 1;
55                 end if;
56              end case;
57
58      end if ;
59    end process;
60
61    -- calcul des sorties en fonction de l'état
62    r <= '1' when current_state=SR else '0' ;
63    v <= '1' when current_state=SV else '0' ;
64    o <= '1' when current_state=SO else '0' ;
65  end a1;
```

Part 4: With a pedestrian call button

```
1   Library ieee ;
2   Use ieee.std_logic_1164.all ;
3
4   entity feux is
5   generic ( rcount : integer := 5;
6             ocount : integer := 3;
7             vcount : integer := 1);
8
9   port (clk, raz, btn : in  std_logic ;  -- horloge, reset, et bouton poussoir
10        r,o,v    : out std_logic ) ;       -- commande des lumières rouge, orange et vert
11  end entity feux ;
12
13  architecture a1 of feux is
14
15  type statetype is (SR, SV, SO);     -- états de la machine d'état
16  signal current_state : statetype;
17  signal counter : integer := rcount;                    -- registre d'état
18
19  begin
20
21    process(clk,raz, btn)
22    begin
23      if (raz='0') then                      -- reset asynchrone
24          current_state <= SR ;
25
26      elsif (clk'event and clk='1') then       -- au top d'horloge
27
28           case current_state is
29              when SR =>
30                 if (counter = 0) then
31                    current_state <= SV;
32                    counter <= vcount;
33
34                 else
35                    counter <= counter - 1;
36                 end if;
37
38              when SV =>
39
40                 if (btn = '0') then
41                    current_state <= SO;
42                    counter <= ocount;
43
44                 else
45
46                    if (counter = 0) then
47                       current_state <= SO;
48                       counter <= ocount;
49
50                    else
51                       counter <= counter - 1;
52                    end if;
53                 end if;
54
55              when SO =>
56                 if (counter = 0) then
57                    current_state <= SR;
58                    counter <= rcount;
59
60                 else
61                    counter <= counter - 1;
62                 end if;
63              end case;
64      end if ;
65    end process;
66
67    -- calcul des sorties en fonction de l'état
68    r <= '1' when current_state=SR else '0' ;
69    v <= '1' when current_state=SV else '0' ;
70    o <= '1' when current_state=SO else '0' ;
71  end a1;
```

Part 5: With the change to orange occurring only after 5 seconds and memorization of the command if the user presses for the first 5 seconds

Code:

```
Library ieee ;
Use ieee.std_logic_1164.all ;

entity feux is
generic ( rcount : integer := 5;
          ocount : integer := 3;
          vcount : integer := 5);

port (clk, raz, btn : in  std_logic ;  -- horloge, reset, et bouton poussoir
      r,o,v    : out std_logic ) ;       -- commande des lumières rouge, orange et vert
end entity feux ;

architecture a1 of feux is

type statetype is (SR, SV, SO);     -- états de la machine d'état
signal current_state : statetype;
signal counter : integer := rcount;                  -- registre d'état
signal wait_counter : integer := 5;                  -- registre d'état


begin



process(clk,raz, btn)
variable lock : std_logic;                    -- registre d'état
variable tm1, tmp2, tmp3, tmp4, tmp5 : std_logic;

  begin
    if (raz='0') then                      -- reset asynchrone
        current_state <= SR ;

    elsif (clk'event and clk='1') then       -- au top d'horloge

         case current_state is
            when SR =>
               if (counter = 0) then
                  current_state <= SV;
                  counter <= vcount;
                  lock <= '1';

               else
                  counter <= counter - 1;
               end if;

            when SV =>
                  if (btn = '0' and lock ='1') then
                        wait_counter <= wait_counter - 1;
                        current_state <= SV;
                        lock <='0';
                        tmp1 :='1';


                  elsif (lock ='0' and wait_counter = 4) then
                        wait_counter <= wait_counter - 1;
                        current_state <= SV;
                        lock <='0';

                        if (btn = '0') then
                        tmp2 :='1';

                        end if;


                  elsif (lock ='0' and wait_counter = 3) then
                        wait_counter <= wait_counter - 1;
                        current_state <= SV;
                        lock <='0';

                        if (btn = '0') then
                        tmp3 :='1';
```

```
72                          end if;
73
74
75                      elsif (lock ='0' and wait_counter = 2) then
76                            wait_counter <= wait_counter - 1;
77                            current_state <= SV;
78                            lock <='0';
79
80                            if (btn = '0') then
81                            tmp4 :='1';
82                            end if;
83
84                      elsif (lock ='0' and wait_counter = 1) then
85                            wait_counter <= 5;
86                            current_state <= SO;
87                            counter <= ocount;
88
89                      if (btn = '0') then
90                            tmp4 :='1';
91                            end if;
92
93
94                   else
95
96                      if (counter = 0) then
97                         current_state <= SO;
98                         counter <= ocount;
99
100                     else
101                        counter <= counter - 1;
102                     end if;
103                  end if;
104
105               when SO =>
106                  if (counter = 0) then
107                     current_state <= SR;
108                     counter <= rcount;
109
110                  else
111                     counter <= counter - 1;
112                  end if;
113               end case;
114
115      end if ;
116    end process;
117
118    -- calcul des sorties en fonction de l'état
119    r <= '1' when current_state=SR else '0' ;
120    v <= '1' when current_state=SV else '0' ;
121    o <= '1' when current_state=SO else '0' ;
122 end a1;
```

Conclusion

In this practical session, we implemented the operation of the traffic light with different scenarios, including when the duration is configured and when the pedestrian presses the call button. We experienced a challenge during testing of the last two codes as our device was not working as expected.