

```

1  -- ROBOT.VHD --
2
3  library ieee;
4  use ieee.std_logic_1164.all;
5
6  entity Robot is
7      port(reset, clk, athome, findfood, lostfood, closetofood,
8            success, aboverestth, abovesearchth, scantimeup: in std_logic;
9            rest, search, food: out std_logic);
10 end Robot;
11
12 architecture automate_robot of Robot is
13
14     type States is (IDLE, RESTING, RANDOMWALK, SCANAREA, HOMING, MOVETOFOOD,
15                     MOVETOHOME, DEPOSIT, GRABFOOD) ;
16     Signal state, nextstate : States;
17     -- psl default clock is rising_edge(clk);
18
19     -- psl property p1 is always (search = '1' -> (findfood = '1') before! (state =
20     GRABFOOD) );
21     -- psl assert p1;
22
23     -- psl property p2 is always (search = '1' -> (abovesearchth = '1') before!
24     (state = HOMING) );
25     -- psl assert p2;
26
27     -- psl property p3 is always (state = MOVETOHOME -> state=DEPOSIT before! rest =
28     '1');
29     -- psl assert p3;
30
31     -- psl property p4 is
32
33     -- always { state = RANDOMWALK and abovesearchth = '0';
34     -- (abovesearchth = '0' and findfood = '0' and not(state = IDLE) ) [*];
35     -- (abovesearchth = '0' and findfood = '1' and not(state = IDLE)) ;
36     -- (abovesearchth = '0' and lostfood = '0' and closetofood = '0' and not(state =
37     IDLE)) [*];
38     -- (abovesearchth = '0' and lostfood = '1' and not(state = IDLE)) ;
39     -- (abovesearchth = '0' and findfood = '0' and scantimeup = '0' and not(state =
40     IDLE)) [*];
41     -- (abovesearchth = '0' and findfood = '0' and scantimeup = '1' and not(state =
42     IDLE)) } | => {state = RANDOMWALK} ;
43     -- psl assert p4;
44
45     -- psl property p5 is always ( {state = RESTING } | => {[*] ; state = RANDOMWALK
46     });
47     -- psl assert p5;
48
49 begin
50
51     -- Calcul de l'état suivant
52     -- Comme on est en std_logic, "elsif = '0'" et non "else", car le signal peut
53     avoir d'autre valeur
54     process (state, athome, findfood, lostfood, closetofood, success, aboverestth,
55             abovesearchth, scantimeup)
56     begin
57         case state is
58             when IDLE => nextstate <= RESTING;
59             when RESTING =>
60                 if aboverestth = '1' then nextstate <= RANDOMWALK;
61                 else--elsif aboverestth = '0' then
62                     nextstate <= RESTING;
63                 end if;
64
65             when RANDOMWALK =>
66                 if abovesearchth = '1' then nextstate <= HOMING;
67                 else-- abovesearchth = '0' then
68                     if findfood = '1' then nextstate <= MOVETOFOOD;
69                     else--elsif findfood = '0' then
70                         nextstate <= RANDOMWALK;
71                     end if;
72
73             when SCANAREA =>
74                 if closetofood = '1' then nextstate <= MOVETOHOME;
75                 else-- closetofood = '0' then
76                     if search = '1' then nextstate <= GRABFOOD;
77                     else--elsif search = '0' then
78                         nextstate <= RESTING;
79                     end if;
80
81             when HOMING =>
82                 if scantimeup = '1' then nextstate <= MOVETOFOOD;
83                 else-- scantimeup = '0' then
84                     nextstate <= RESTING;
85                 end if;
86
87             when MOVETOHOME =>
88                 if rest = '1' then nextstate <= DEPOSIT;
89                 else-- rest = '0' then
90                     nextstate <= RESTING;
91                 end if;
92
93             when DEPOSIT =>
94                 if food = '1' then nextstate <= GRABFOOD;
95                 else-- food = '0' then
96                     nextstate <= RESTING;
97                 end if;
98
99             when GRABFOOD =>
100                 if success = '1' then nextstate <= RESTING;
101                 else-- success = '0' then
102                     nextstate <= RANDOMWALK;
103                 end if;
104
105         end case;
106     end process;
107
108     state <= nextstate;
109 end;

```

```

64         end if;
65
66     when SCANAREA =>
67         if abovesearchth = '1' then nextstate <= HOMING;
68         else--elsif abovesearchth = '0' then
69             if findfood = '1' then nextstate <= MOVETOFOOD;
70             else--elsif findfood = '0' then
71                 if scantimeup = '1' then nextstate <= RANDOMWALK;
72                 else--elsif scantimeup = '0' then
73                     nextstate <= SCANAREA;
74                 end if;
75             end if;
76         end if;
77     when HOMING => if(athome = '1') then nextstate <= RESTING; else
nextstate <= HOMING; end if;
78     when MOVETOFOOD =>
79         if abovesearchth = '1' then nextstate <= HOMING;
80         else--elsif abovesearchth = '0' then
81             if lostfood = '1' then nextstate <= SCANAREA;
82             else--elsif lostfood = '0' then
83                 if closetofood = '1' then nextstate <= GRABFOOD;
84                 else--elsif closetofood = '0' then
85                     nextstate <= MOVETOFOOD;
86                 end if;
87             end if;
88         end if;
89     when GRABFOOD =>
90         if success = '1' then nextstate <= MOVETOHOME;
91         else--elsif success = '0' then
92             nextstate <= GRABFOOD;
93         end if;
94     when MOVETOHOME =>
95         if athome = '1' then nextstate <= DEPOSIT;
96         else--elsif athome = '0' then
97             nextstate <= MOVETOHOME;
98         end if;
99     when DEPOSIT =>
100         if success = '1' then nextstate <= RESTING;
101         else--elsif success = '0' then
102             nextstate <= DEPOSIT;
103         end if;
104     end case;
105 end process;
106
107 -- MISE A JOUR DU REGISTRE D'ETAT
108
109 process(reset, clk)
110 begin
111     -- RESET : asynchrone haut
112     if reset = '1' then state <= IDLE;
113     -- HORLOGE : front montant
114     elsif (clk'event and clk = '1') then
115         state <= nextstate;
116     end if;
117 end process;
118
119
120 -- MISE A JOUR DES OUTPUTS
121 rest <= '1' when (( state = DEPOSIT and success = '1' ) OR (state = IDLE) OR
(state = HOMING and athome = '1') ) else '0';
122 search <= '1' when (state = RESTING and aboverestth = '1' ) else '0';
123 food <= '1' when (state = MOVETOFOOD and abovesearchth = '0' and lostfood = '0'
and closetofood = '1') else '0';
124
125
126
127 end automate_robot;
128
129

```