M1 Info – ARC - LAB2

Olivier HUREAU

18/03/2020

1 Dessin de l'automate

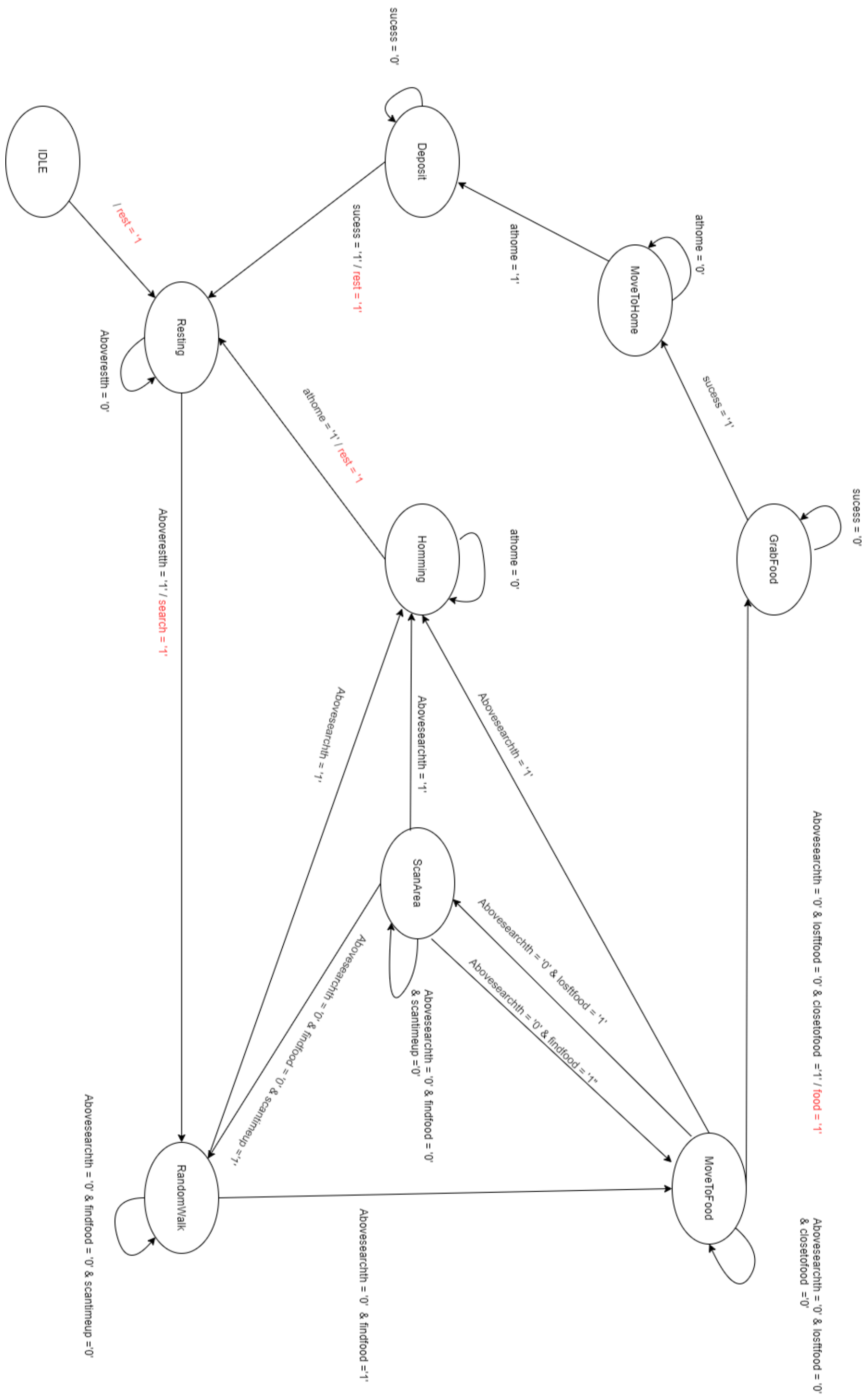

IDLE
Deposit
Resting
MoveToHome
Homming
GrabFood
ScanArea
RandomWalk
MoveToFood
sucess = '0'
/ rest = '1'
sucess = '1' / rest = '1'
athome = '1'
athome = '0'
Aboverestth = '0'
sucess = '1'
athome = '1' / rest = '1'
sucess = '0'
athome = '0'
Aboverestth = '1' / search = '1'
Abovesearchth = '1'
Abovesearchth = '1'
Abovesearchth = '1'
Abovesearchth = '0' & lostfood = '0' & closetofood ='1' / food = '1'
Abovesearchth = '0' & lostfood = '1'
Abovesearchth = '0' & findfood = '1'
Abovesearchth = '0' & findfood = '0' & scantimeup = '1'
Abovesearchth = '0' & findfood = '0' & scantimeup ='0'
Abovesearchth = '0' & findfood = '0' & scantimeup ='0'
Abovesearchth = '0' & findfood ='1'
Abovesearchth = '0' & lostfood = '0' & closetofood ='0'

Figure 1: Automate du robot

2 Description VHDL comportementale

Le code source est donné en annexe du document (robot.vhd et testRobot.vhd)

3 TestRobot

3.1 Réalisation du test

Pour tester mon implémentation, j'ai tout d'abord rédigé un tableau des différentes entrées et sorties et des comportements que ceux-ci devais avoir pour toutes les transitions ainsi que les différentes pour y accéder depuis un reset de l'automate.

Dans ce tableau se trouve différentes informations tel que : En rouge : les inputs que je doit mettre à 1, En bleu : les données que doivent prendre les outputs.

Ce tableau est fournis en annexe.

3.2 Interpretation du test

Après simulation, on obtiens les différents graphes fournis en annexe. La figure 2 représente la totalité de la simulation. Les figures 3, 4 et 5 sont cette même simulation partitionné.

Ce que l'on peux observer avec ces différents graphes c'est que notre simulation implémente bien le problème. Les sorties et la suite d'état correspondent bien à ce qui a était prédit dans notre tableau.

4 Counter

Le code source est donné en annexe du document (count.vhd et testCount.vhd)

Après simulation, sur la figure 6 on observe bien que le compteur ne se met en marche uniquement lorsque l'état start est à 1 et le reset fonctionne bien. Le nombre de front montant compté est le bon.

```vhdl
-- ROBOT.VHD --

library ieee;
use ieee.std_logic_1164.all;

entity Robot is
    port(reset, clk, athome, findfood, lostfood, closetofood,
    success, aboverestth, abovesearchth, scantimeup: in std_logic;
    rest, search, food: out std_logic);
end Robot;

architecture automate_robot of Robot is

    type States is (IDLE, RESTING, RANDOMWALK, SCANAREA, HOMING, MOVETOFOOD,
    MOVETOHOME, DEPOSIT, GRABFOOD) ;
    Signal state, nextstate : States := IDLE;

begin
    -- Calcul de l'état suivant
    -- Comme on est en std_logic,"elsif ='0'" et non "else", car le signal peux
    avoir d'autre valeur
    process (state, athome, findfood, lostfood, closetofood, success, aboverestth,
    abovesearchth, scantimeup)
    begin
        case state is
            when IDLE => nextstate <= RESTING;
            when RESTING =>
                if aboverestth = '1' then nextstate <= RANDOMWALK;
                elsif aboverestth = '0' then nextstate <= RESTING;
                end if;

            when RANDOMWALK =>
                if abovesearchth = '1' then nextstate <= HOMING;
                elsif  abovesearchth = '0' then
                    if findfood = '1' then nextstate <= MOVETOFOOD;
                    elsif findfood = '0' then
                        nextstate <= RANDOMWALK;
                    end if;
                end if;

            when SCANAREA =>
                if abovesearchth = '1' then nextstate <= HOMING;
                elsif  abovesearchth = '0' then
                    if findfood = '1' then nextstate <= MOVETOFOOD;
                    elsif findfood = '0' then
                        if scantimeup = '1' then nextstate <= RANDOMWALK;
                        elsif scantimeup = '0' then nextstate <= SCANAREA;
                        end if;
                    end if;
                end if;
            when HOMING => nextstate <= RESTING;
            when MOVETOFOOD =>
                if abovesearchth = '1' then nextstate <= HOMING;
                elsif  abovesearchth = '0' then
                    if lostfood = '1' then nextstate <= SCANAREA;
                    elsif lostfood = '0' then
                        if closetofood = '1' then nextstate <= GRABFOOD;
                        elsif closetofood = '0' then
                            nextstate <= MOVETOFOOD;
                        end if;
                    end if;
                end if;
            when GRABFOOD =>
                if success = '1' then nextstate <= MOVETOHOME;
                elsif success = '0' then nextstate <= GRABFOOD;
                end if;
            when MOVETOHOME =>
                if athome = '1' then nextstate <= DEPOSIT;
                elsif athome = '0' then nextstate <= MOVETOFOOD;
                end if;
            when DEPOSIT =>
                if success = '1' then nextstate <= RESTING;
                elsif success = '0' then nextstate <= DEPOSIT;
                end if;
        end case;
    end process;

    -- MISE A JOUR DU REGISTRE D'ETAT

    process(reset, clk)
    begin
        -- RESET : asynchrone haut
        if reset = '1' then state <= IDLE;
        -- HORLOGE : front montant
        elsif (clk'event and clk = '1') then
            state <= nextstate;
        end if;
    end process;


    -- MISE A JOUR DES OUTPUTS
    rest <= '1' when (( state = DEPOSIT and success = '1' ) OR (state = IDLE) OR
    (state = HOMING and athome = '1') ) else '0';
    search <= '1' when (state = RESTING and aboverestth = '1' ) else '0';
    food <= '1' when  (state = MOVETOFOOD and abovesearchth = '0' and lostfood = '0'
    and closetofood ='1') else '0';



end automate_robot;
```

```vhdl
-- TESTROBOT.VHD --

library ieee;
use ieee.std_logic_1164.all;

entity testRobot is
end testRobot;

architecture test1 of testRobot is
  component Robot is
     port(reset, clk, athome, findfood, lostfood, closetofood,
    success, aboverestth, abovesearchth, scantimeup: in std_logic;
    rest, search, food: out std_logic);
  end component;
  signal r, clk, ah, f, l, c, s, arest, asearch,scan, re, se, fo : std_logic := '0';
begin
     A: Robot port map(r, clk, ah, f, l, c, s, arest, asearch,scan, re, se, fo);
    -- manage reset
    r <= '0', '1' after 44 ns, '0' after 45 ns, '1' after 84 ns, '0' after 85 ns,
    '1' after  144 ns, '0' after 145  ns,
    '1' after 224 ns, '0' after  225 ns, '1' after  324 ns, '0' after 325  ns, '1'
    after  444 ns, '0' after 445  ns, '1' after  584 ns, '0' after  585 ns,
    '1' after  684 ns, '0' after 685  ns, '1' after  804 ns, '0' after 805  ns, '1'
    after 904  ns, '0' after 905  ns, '1' after  984 ns, '0' after  985 ns,
    '1' after 1064  ns, '0' after 1065  ns, '1' after 1204  ns, '0' after 1205  ns,
    '1' after 1324  ns, '0' after 1325  ns;
    -- manage clock
    process
    begin
        clk <= '0';
        wait for 10 ns;
        clk <= '1';
        wait for 10 ns;
    end process;
    -- manage athome
    ah <= '0', '1' after 545 ns, '0' after 555 ns, '1' after 1045 ns, '0' after 1055
    ns, '1' after 1165 ns, '0' after 1175 ns;
    -- manage findfood
    f <= '0', '1' after 185 ns, '0' after 195 ns, '1' after 265 ns, '0' after 275
    ns, '1' after 365 ns, '0' after 375 ns,
    '1' after 485 ns, '0' after 495 ns, '1' after 625 ns, '0' after 635 ns, '1'
    after 725 ns, '0' after 735 ns,
    '1' after 845 ns, '0' after 855 ns, '1' after 1105 ns, '0' after 1115 ns, '1'
    after 1245 ns, '0' after 1255 ns,
    '1' after 1265 ns, '0' after 1275 ns;
    -- manage lostfood
    l <= '0', '1' after 645 ns, '0' after 655 ns, '1' after 745 ns, '0' after 755
    ns, '1' after 1265 ns, '0' after 1275 ns;
    --manage closetofood
    c <= '0' , '1' after 285 ns, '0' after 295 ns, '1' after 385 ns, '0' after 395
    ns, '1' after 505 ns, '0' after 515 ns,
    '1' after 1125 ns, '0' after 1135 ns;
    --manage sucess
    s <= '0', '1' after 405 ns, '0' after 415 ns, '1' after 525 ns, '0' after 535
    ns, '1' after 1145 ns, '0' after 1155 ns,
    '1' after 1185 ns, '0' after 1195 ns;

    -- manage aboverestth

    arest <= '0', '1' after 105 ns, '0' after 115 ns, '1' after 165 ns, '0' after
    175 ns,
    '1' after 245 ns, '0' after 255 ns , '1' after 345 ns, '0' after 355 ns, '1'
    after 465 ns, '0' after 475 ns,
    '1' after 605 ns, '0' after 615 ns, '1' after 705  ns, '0' after 715 ns, '1'
    after 825 ns, '0' after 835 ns,
    '1' after 925 ns, '0' after 935 ns, '1' after 1005 ns, '0' after 1015 ns, '1'
    after 1085 ns, '0' after 1095 ns,
    '1' after 1225 ns, '0' after 1235 ns;

    -- manage abovesearchth
    asearch <= '0', '1' after 765 ns, '0' after 775 ns, '1' after 865 ns, '0' after
    875 ns, '1' after 945 ns, '0' after 955 ns,
    '1' after 1025 ns, '0' after 1035 ns;


    -- manage scantimeup
    scan <= '0', '1' after 1405 ns, '0' after 1415 ns;

end test1;

library work;
configuration config1 of work.testRobot is
    for test1
       for A:Robot use entity work.Robot(automate_robot);
       end for;
    end for;
end config1;
```

STATE	NEXTSTATE	INPUT								OUTPUT			CYCLE	TEMPS MIS A '1'	TEMPS MIS A '0'	MARQUE
		athome	findfood	lostfood	closetofood	success	aboveresttth	abovesearchti	scantimeup	rest	food	search				
FROM IDLE TO IDLE																
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	1	25	35	1
FROM IDLE TO RESTING													RESET	44	45	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	2	45	55	2
RESTING	RESTING	0	0	0	0	0	0	0	0	0	0	0	3	65	75	
FROM IDLE TO RANDOWALK													RESET	84	85	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	4	85	95	3
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	5	105	115	
RANDOMWAL	RANDOMWAL	0	0	0	0	0	0	0	0	0	0	0	6	125	135	
FROM IDLE TO MOVETOFOOD													RESET	144	145	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	7	145	155	4
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	8	165	175	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	9	185	195	
MOVETOFOOD	MOVETOFOOD	0	0	0	0	0	0	0	0	0	0	0	10	205	215	
FROM IDLE TO GRABFOOD													RESET	224	225	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	11	225	235	5
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	12	245	255	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	13	265	275	
MOVETOFOOD	GRABFOOD	0	0	0	1	0	0	0	0	0	1	0	14	285	295	
GRABFOOD	GRABFOOD	0	0	0	0	0	0	0	0	0	0	0	15	305	315	
FROM IDLE T MOVETOHOME													RESET	324	325	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	16	325	335	6
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	17	345	355	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	18	365	375	
MOVETOFOOD	GRABFOOD	0	0	0	1	0	0	0	0	0	1	0	19	385	395	
GRABFOOD	MOVETOHOM	0	0	0	0	1	0	0	0	0	0	0	20	405	415	
MOVETOHOM	MOVETOHOM	0	0	0	0	0	0	0	0	0	0	0	21	425	435	
FROM IDLE TO DEPOSIT													RESET	444	445	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	22	445	455	7
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	23	465	475	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	24	485	495	
MOVETOFOOD	GRABFOOD	0	0	0	1	0	0	0	0	0	1	0	25	505	515	
GRABFOOD	MOVETOHOM	0	0	0	0	1	0	0	0	0	0	0	26	525	535	
MOVETOHOM	DEPOSIT	1	0	0	0	0	0	0	0	0	0	0	27	545	555	
DEPOSIT	DEPOSIT	0	0	0	0	0	0	0	0	0	0	0	28	565	575	
FROM IDLE TO SCAN AREA													RESET	584	585	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	29	585	595	8
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	30	605	615	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	31	625	635	
MOVETOFOOD	SCANAREA	0	0	1	0	0	0	0	0	0	0	0	32	645	655	
SCANAREA	SCANAREA	0	0	0	0	0	0	0	0	0	0	0	33	665	675	
FROM IDLE TO HOMING VIA SCANAREA													RESET	684	685	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	34	685	695	9
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	35	705	715	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	36	725	735	
MOVETOFOOD	SCANAREA	0	0	1	0	0	0	0	0	0	0	0	37	745	755	
SCANAREA	HOMING	0	0	0	0	0	0	1	0	0	0	0	38	765	775	
HOMING	HOMING	0	0	0	0	0	0	0	0	0	0	0	39	785	795	
FROM IDLE TO HOMING VIA MOVETOFOOD													RESET	804	805	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	40	805	815	10
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	41	825	835	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	42	845	855	
MOVETOFOOD	HOMING	0	0	0	0	0	0	1	0	0	0	0	43	865	875	
HOMING	HOMING	0	0	0	0	0	0	0	0	0	0	0	44	885	895	
FROM IDLE TO HOMING VIA RANDOMWALK													RESET	904	905	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	45	905	915	11
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	46	925	935	
RANDOMWAL	HOMING	0	0	0	0	0	0	1	0	0	0	0	47	945	955	
HOMING	HOMING	0	0	0	0	0	0	0	0	0	0	0	48	965	975	
FROM IDLE TO RESTING VIA HOMMING													RESET	984	985	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	49	985	995	12
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	50	1005	1015	
RANDOMWAL	HOMING	0	0	0	0	0	0	1	0	0	0	0	51	1025	1035	
HOMING	RESTING	1	0	0	0	0	0	0	0	1	0	0	52	1045	1055	
FROM IDLE TO RESTING VIA DEPOSIT													RESET	1064	1065	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	53	1065	1075	13
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	54	1085	1095	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	55	1105	1115	
MOVETOFOOD	GRABFOOD	0	0	0	1	0	0	0	0	0	1	0	56	1125	1135	
GRABFOOD	MOVETOHOM	0	0	0	0	1	0	0	0	0	0	0	57	1145	1155	
MOVETOHOM	DEPOSIT	1	0	0	0	0	0	0	0	0	0	0	58	1165	1175	
DEPOSIT	RESTING	0	0	0	0	1	0	0	0	1	0	0	59	1185	1195	
FROM IDLE TO MOVETOFOOD VIA SCANAREA													RESET	1204	1205	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	60	1205	1215	14
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	61	1225	1235	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	62	1245	1255	
MOVETOFOOD	SCANAREA	0	0	1	0	0	0	0	0	0	0	0	64	1285	1295	
SCANAREA	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	65	1305	1315	
FROM IDLE TO RANDOMWALK VIA SCANAREA													RESET	1324	1325	
IDLE	RESTING	0	0	0	0	0	0	0	0	1	0	0	66	1325	1335	15
RESTING	RANDOMWAL	0	0	0	0	0	1	0	0	0	0	1	67	1345	1355	
RANDOMWAL	MOVETOFOOD	0	1	0	0	0	0	0	0	0	0	0	68	1365	1375	
MOVETOFOOD	SCANAREA	0	0	1	0	0	0	0	0	0	0	0	69	1385	1395	
SCANAREA	RANDOMWAL	0	0	0	0	0	0	0	1	0	0	0	70	1405	1415	

```vhdl
-- COUNT.VHD --
library ieee;
use ieee.std_logic_1164.all;


entity Count is generic(threshold: natural := 10);
 port(reset, clk, start: in std_logic; aboveth: out std_logic);
end Count;

architecture Behav of Count is
    type States is (IDLE, COUNTING);
    Signal state, nextstate : States := IDLE;
    Signal c : natural := 0;
begin
    -- Calcul de l'état suivant
    -- Comme on est en std_logic,"elsif ='0'" et non "else", car le signal peux
    avoir d'autre valeur
    process (state, reset, clk, start)
    begin
        case state is
        when IDLE =>
            if start = '1' then
                nextstate <= COUNTING;
            elsif start = '0' then
                nextstate <= IDLE;
            end if;
        when COUNTING =>
            if c < threshold then
                nextstate <= COUNTING;
            else
                nextstate <= IDLE;
            end if;
        end case;
    end process;

    -- MISE A JOUR DU REGISTRE D'ETAT
    process(reset, clk)
    begin
        -- RESET : asynchrone haut
        if reset = '1' then
            state <= IDLE;
        -- HORLOGE : front montant
        elsif (clk'event and clk = '1') then
            state <= nextstate;
        end if;
    end process;

    -- MISE A JOUR A CHAQUE FRONT MONTANT DE LA CLOCK POUR C ou sur un reset
    process(start, clk, c, reset)
    begin
        if(reset = '1') then
            c <= 0;

        else
            if (clk'event and clk = '1') then
                if (state = IDLE and start = '0') then

                elsif ( state = IDLE and start = '1') then
                    c <= c + 1;
                elsif (state = COUNTING and c < threshold) then
                    c <= c + 1;
                elsif(state = COUNTING and c >= threshold) then
                    c <= 0;
                end if;
            end if;
        end if;
    end process;

    -- Mise a jour de aboveth
    aboveth <= '0' when c < threshold else '1';

end Behav;
```

```vhdl
-- TESTCOUNT.VHD --

library ieee;
use ieee.std_logic_1164.all;

entity testCount is
end testCount;

architecture test2 of testCount is
  component Count is
    generic (threshold : natural);
     port(reset, clk, start: in std_logic; aboveth: out std_logic);
  end component;
  signal r, c, s,a : std_logic := '0';
begin
     B: Count
     generic map(3)
     port map(r,c,s,a);

    process
    begin
        c <= '0';
        wait for 10 ns;
        c <= '1';
        wait for 10 ns;
    end process;

    s <= '0', '1' after 20 ns , '0' after 40 ns, '1' after 170 ns, '0' after 190 ns,
    '1' after 210 ns;
    r <= '0', '1' after 200 ns, '0' after 201 ns, '1' after 290 ns, '0' after 300 ns;
end test2;

library work;
configuration config2 of work.testCount is
    for test2
       for B:Count use entity work.testCount(Behav);
       end for;
    end for;
end config2;
```

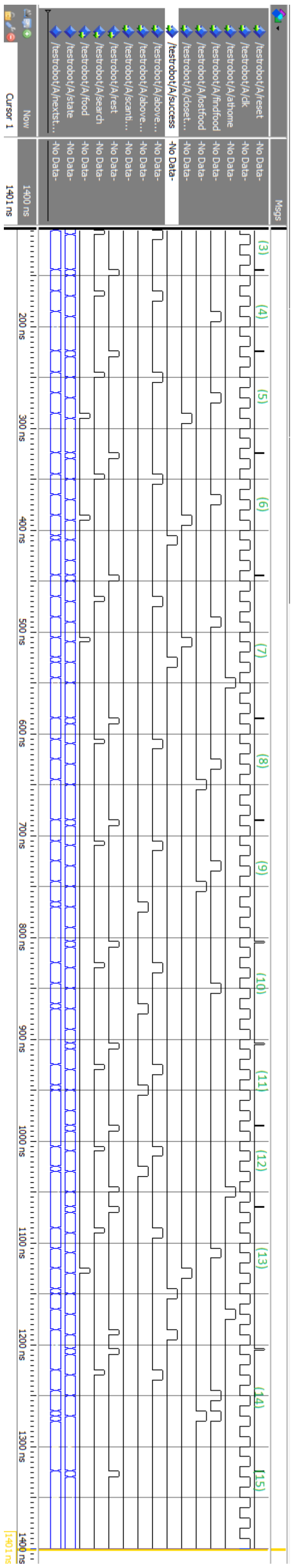


Figure 2: Simulation du robot

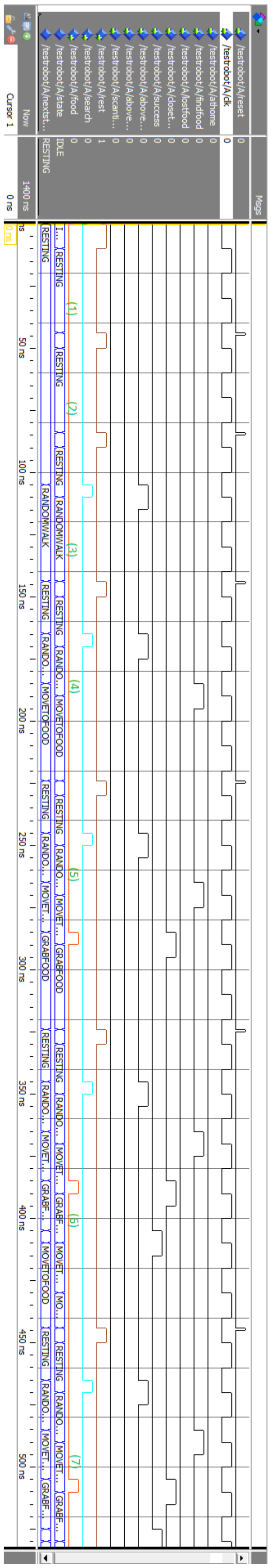


Figure 3: Simulation du robot : 0 ns à 500 ns

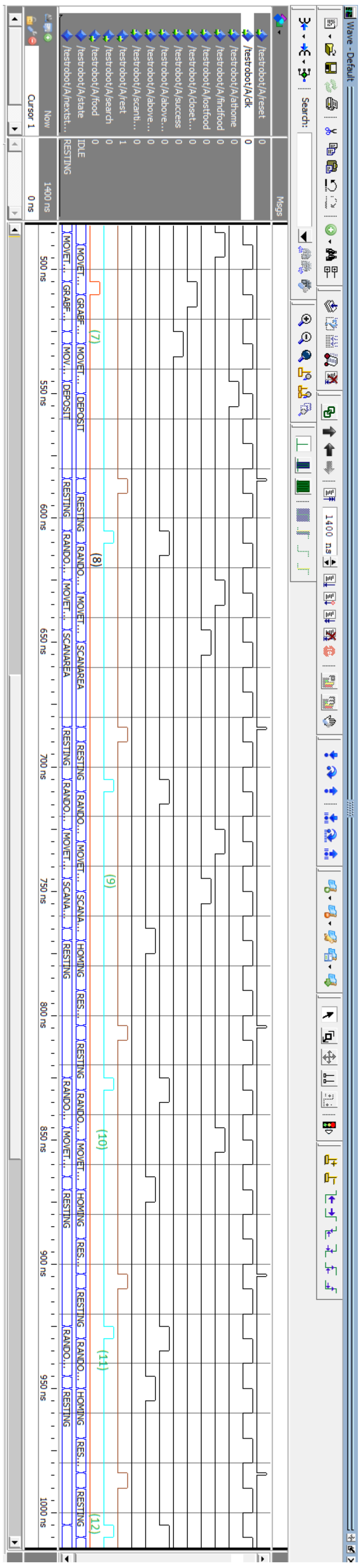


Figure 4: Simulation du robot : 500 ns à 1000 ns

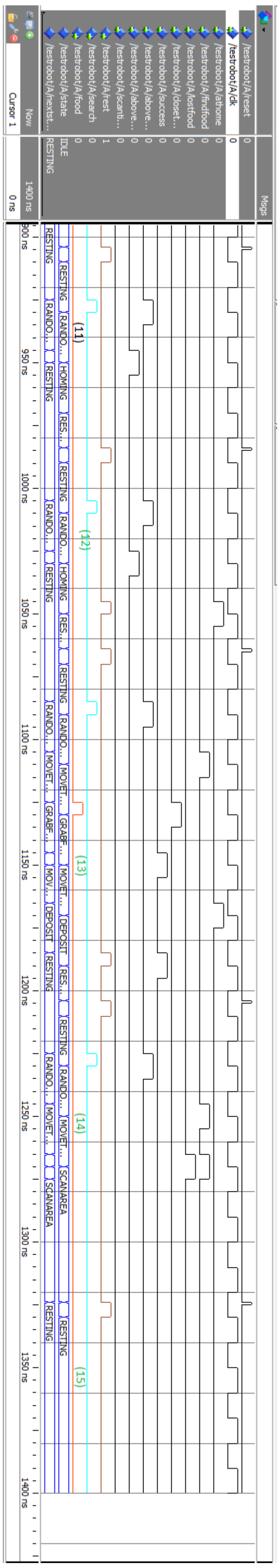
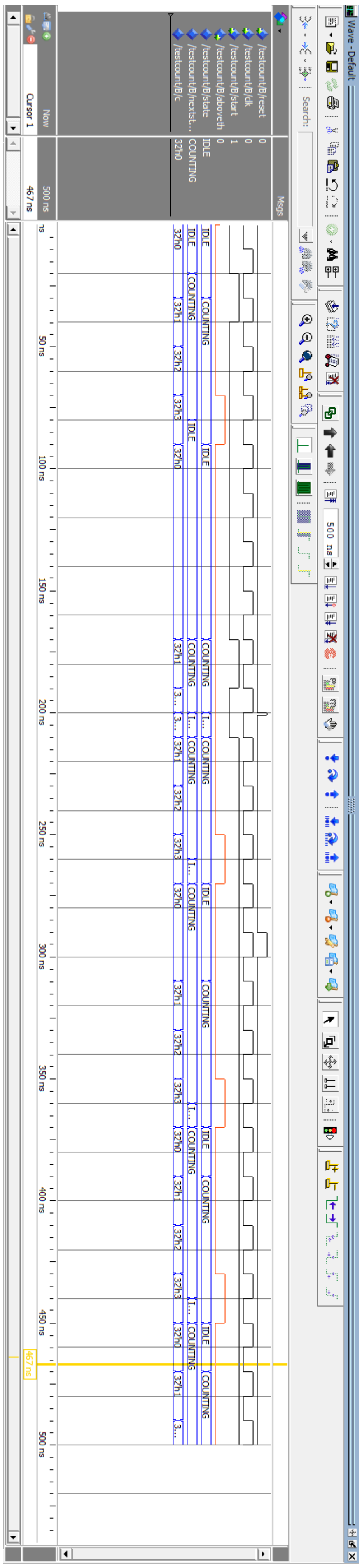


Figure 5: Simulation du robot : 1000 ns à 1500 ns



```
/testcount/B/reset      0
/testcount/B/clk        0
/testcount/B/start      1
/testcount/B/aboveth    0
/testcount/B/state      IDLE
/testcount/B/nextst...  COUNTING
/testcount/B/c          32'h0
Now                     500 ns
Cursor 1                467 ns
```

Figure 6: Simulation du Count