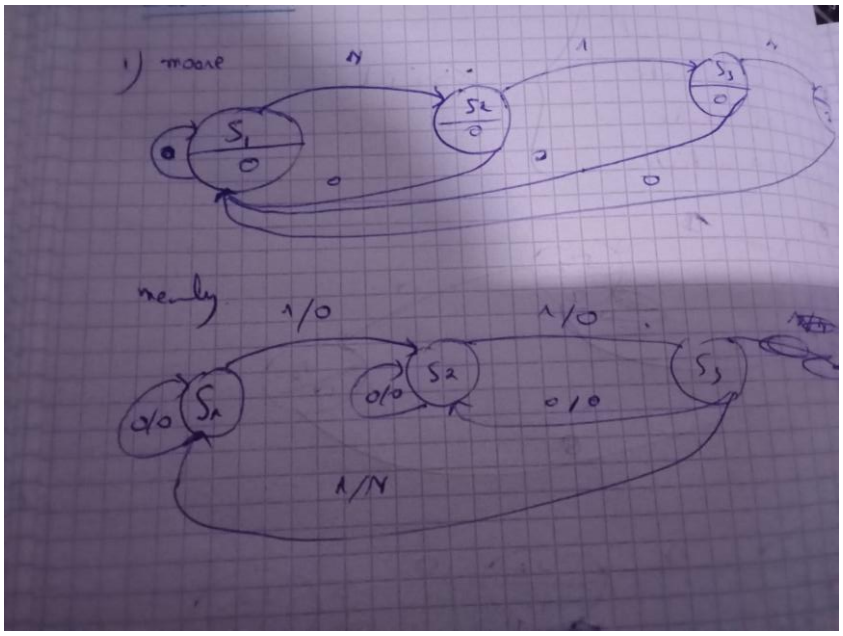
Mohamed Manessouri
SSE

Tp6 : Modélisation de machines à états finis

Exercice 1 :

1)



2)

Moore.vhdl

```
32  entity moore is
33  Port (
34          clk    : in  STD_LOGIC;
35          reset  : in  STD_LOGIC;
36          E      : in  STD_LOGIC;
37          S      : out STD_LOGIC
38      );
39  end moore;
40
41  architecture Behavioral of moore is
42      type Etat_Type is (Etat0, Etat1, Etat2, Etat3);
43      signal Etat_present, Etat_futur : Etat_Type := Etat0;
44  begin
45  process(clk, reset)
46      begin
47          if reset = '0' then
48              Etat_present <= Etat0;
49          elsif rising_edge(clk) then
50              Etat_present <= Etat_futur;
51          end if;
52      end process;
53
54
55  process(E, Etat_present)
56      begin
57          case Etat_present is
58              when Etat0 =>
```

```
58              when Etat0 =>
59                  if E = '1' then
60                      Etat_futur <= Etat1;
61                  else
62                      Etat_futur <= Etat0;
63                  end if;
64              when Etat1 =>
65                  if E = '1' then
66                      Etat_futur <= Etat2;
67                  else
68                    Etat_futur <= Etat0;
69                  end if;
70              when Etat2 =>
71                  if E = '1' then
72                      Etat_futur <= Etat3;
73                  else
74                      Etat_futur <= Etat0;
75                  end if;
76                  when Etat3 =>
77                  if E = '1' then
78                      Etat_futur <= Etat3;
79                  else
80                      Etat_futur <= Etat0;
81                  end if;
82          end case;
83      end process;
84
85  process(Etat_present)
86      begin
87          case Etat_present is
88              when Etat3 =>
89                  S <= '1';
90              when others =>
91                  S <= '0';
92          end case;
93      end process;
94  end Behavioral;
95
96
```

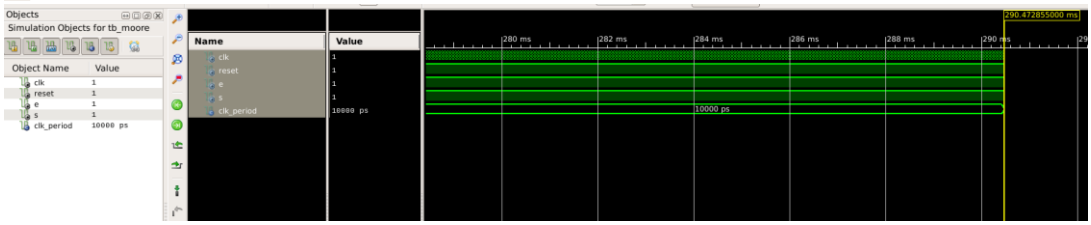
Mealy.vhdl

```
41  architecture Behavioral of mealy is
42      type Etat_Type is (Etat0, Etat1, Etat2);
43      signal Etat_present, Etat_futur : Etat_Type := Etat0;
44  begin
45  process(clk, reset)
46      begin
47          if reset = '0' then
48              Etat_present <= Etat0;
49          elsif rising_edge(clk) then
50              Etat_present <= Etat_futur;
51          end if;
52      end process;
53
54  process(E, Etat_present)
55      begin
56          S <= '0';
57          case Etat_present is
58              when Etat0 =>
59                  if E = '1' then
60                      Etat_futur <= Etat1;
61                  else
62                      Etat_futur <= Etat0;
63                  end if;
64              when Etat1 =>
65                  if E = '1' then
66                      Etat_futur <= Etat2;
67                  else
68                      Etat_futur <= Etat0;
69                  end if;
70              when Etat2 =>
71                  if E = '1' then
72                      Etat_futur <= Etat2;
73                      S <= '1';
74                  else
75                      Etat_futur <= Etat0;
76                  end if;
77          end case;
78      end process;
79
80  end Behavioral;
81
```

3) Test Bench :

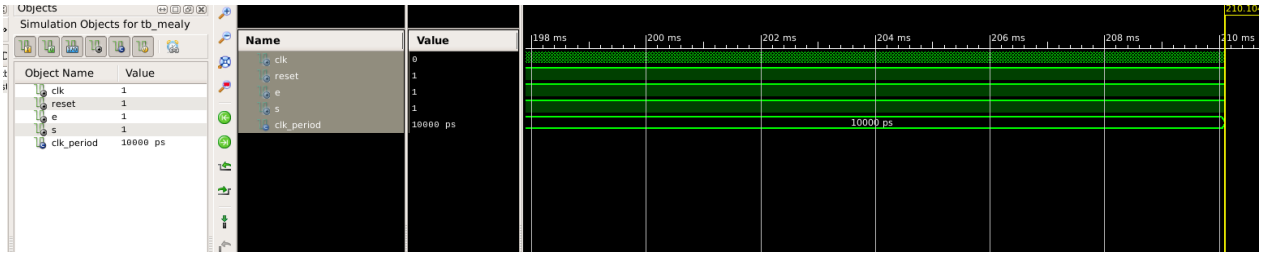
Tb_moore.vhdl

```
    process
     begin
         -- Réinitialisation
         reset <= '0';
         wait for 20 ns;
         reset <= '1';
         wait for 20 ns;

         E <= '0'; wait for CLK_PERIOD;
         E <= '1'; wait for CLK_PERIOD;
         E <= '1'; wait for CLK_PERIOD;
         E <= '1'; wait for CLK_PERIOD;
         E <= '0'; wait for CLK_PERIOD;

         E <= '1'; wait for CLK_PERIOD;
         E <= '0'; wait for CLK_PERIOD;
         E <= '1'; wait for CLK_PERIOD;
         E <= '1'; wait for CLK_PERIOD;

         wait;
     end process;
end behavior;
```



Tb_mealy.vhdl

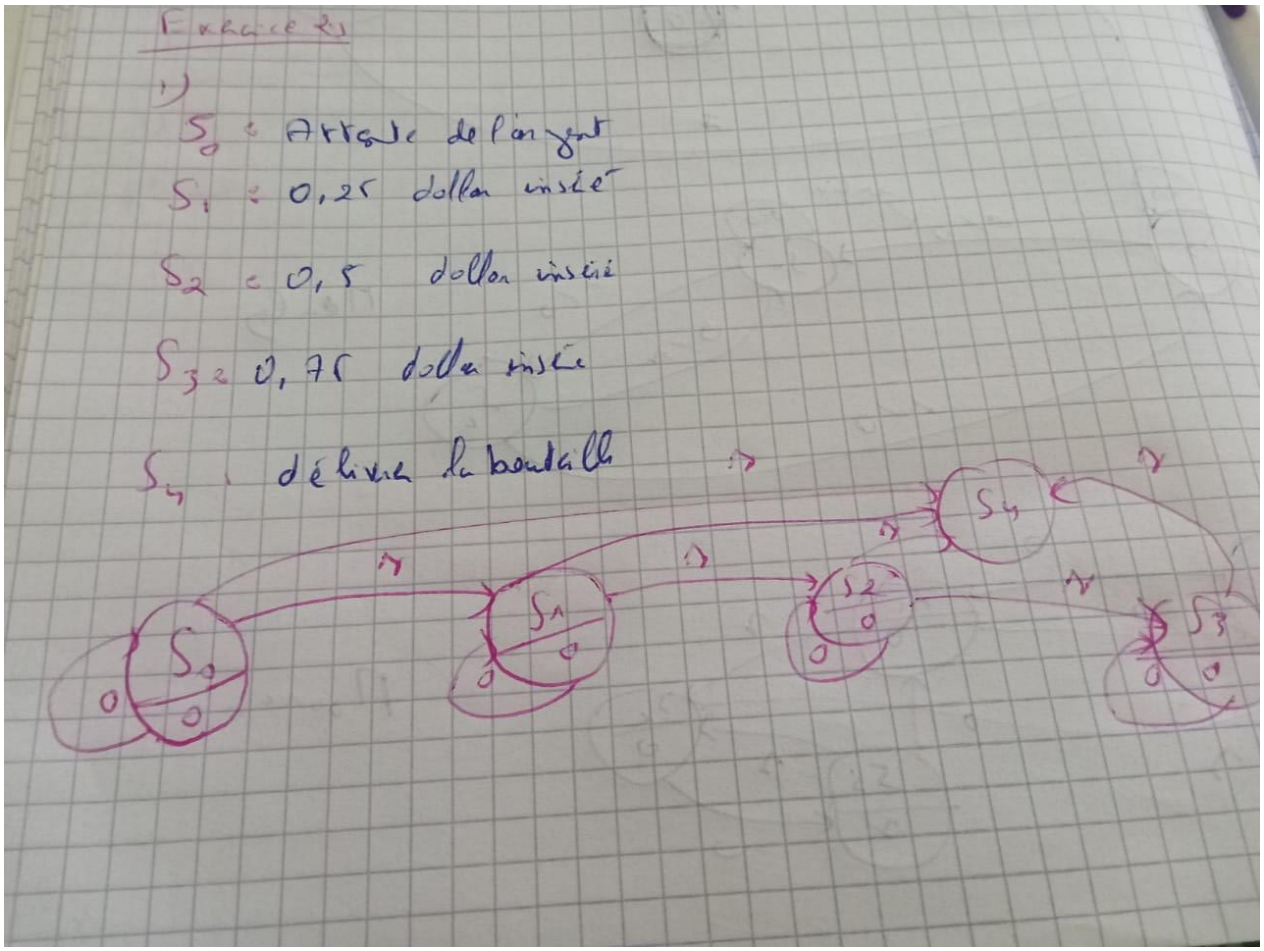
```
82
83      process
84       begin
85           -- Réinitialisation
86           reset <= '0';
87           wait for 20 ns;
88           reset <= '1';
89           wait for 20 ns;
90
91           E <= '0'; wait for CLK_PERIOD;
92           E <= '1'; wait for CLK_PERIOD;
93           E <= '1'; wait for CLK_PERIOD;
94           E <= '1'; wait for CLK_PERIOD;
95           E <= '0'; wait for CLK_PERIOD;
96
97           E <= '1'; wait for CLK_PERIOD;
98           E <= '0'; wait for CLK_PERIOD;
99           E <= '1'; wait for CLK_PERIOD;
100          E <= '1'; wait for CLK_PERIOD;
101
102          wait;
103      end process;
104  END;
```



4) Les sorties d'une machine de Mealy dépendent des entrées et changent immédiatement (asynchrone), tandis que celles d'une machine de Moore dépendent uniquement de l'état présent et changent sur un front d'horloge (synchrone).

Exercice 2 :

1)



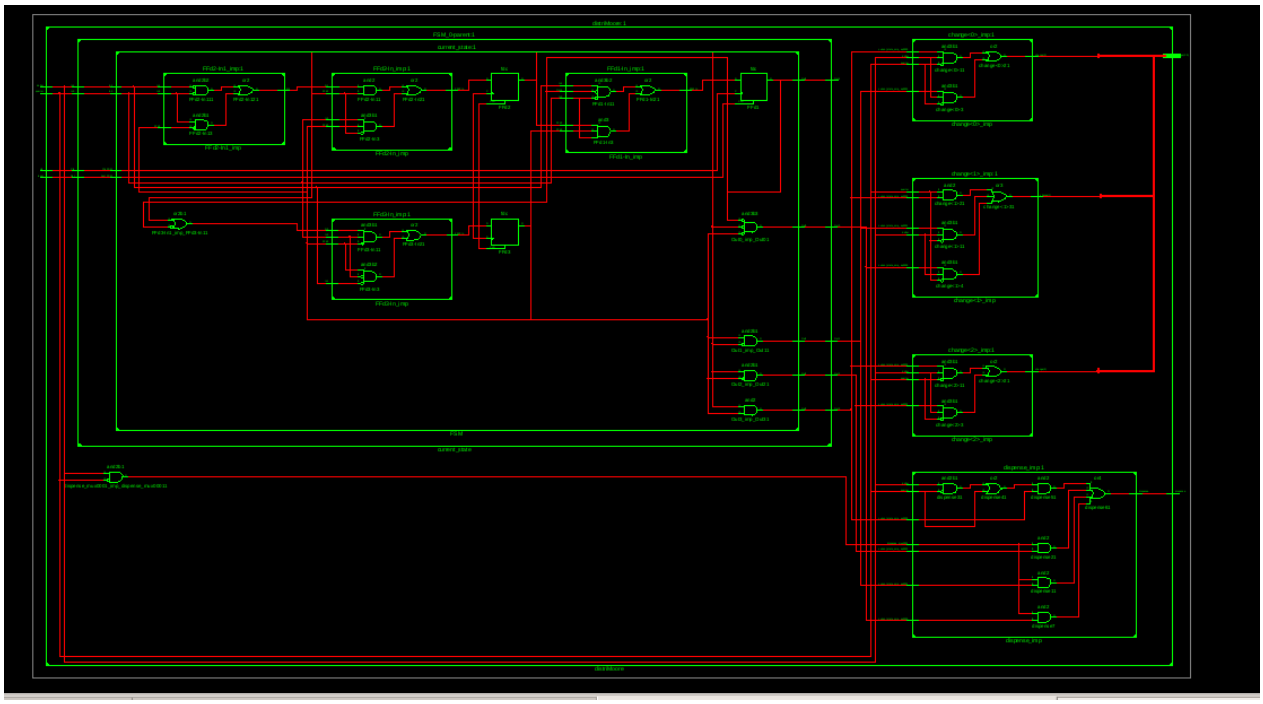
2)

Moore.vhdl

```
32  entity moore is
33   Port (
34          clk       : in  STD_LOGIC;
35          reset     : in  STD_LOGIC;
36          D_in      : in  STD_LOGIC;
37          Q_in      : in  STD_LOGIC;
38          dispense  : out STD_LOGIC;
39          change    : out STD_LOGIC
40      );
41  end moore;
42
43  architecture Behavioral of moore is
44      type state_type is (IDLE, Q25, Q50, Q75, dis);
45      signal current_state, next_state : state_type := IDLE;
46
47      signal internal_dispense : STD_LOGIC := '0';
48      signal internal_change   : STD_LOGIC := '0';
49
50  begin
51  internal_dispense <= '1';
52  dispense <= internal_dispense;
53
```

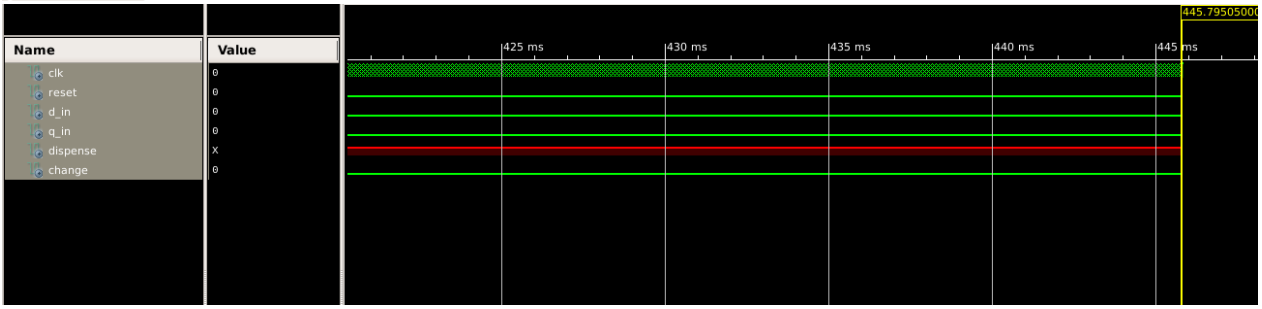
```
50  begin
51  internal_dispense <= '1';
52  dispense <= internal_dispense;
53
54
55  change   <= internal_change;
56  process(clk, reset)
57      begin
58          if reset = '1' then
59              current_state <= IDLE;
60          elsif rising_edge(clk) then
61              current_state <= next_state;
62          end if;
63      end process;
64
65   process(current_state, D_in, Q_in)
66      begin
67          -- Valeurs par défaut
68          next_state          <= current_state;
69          internal_dispense   <= '0';
70          internal_change     <= '0';
71
```

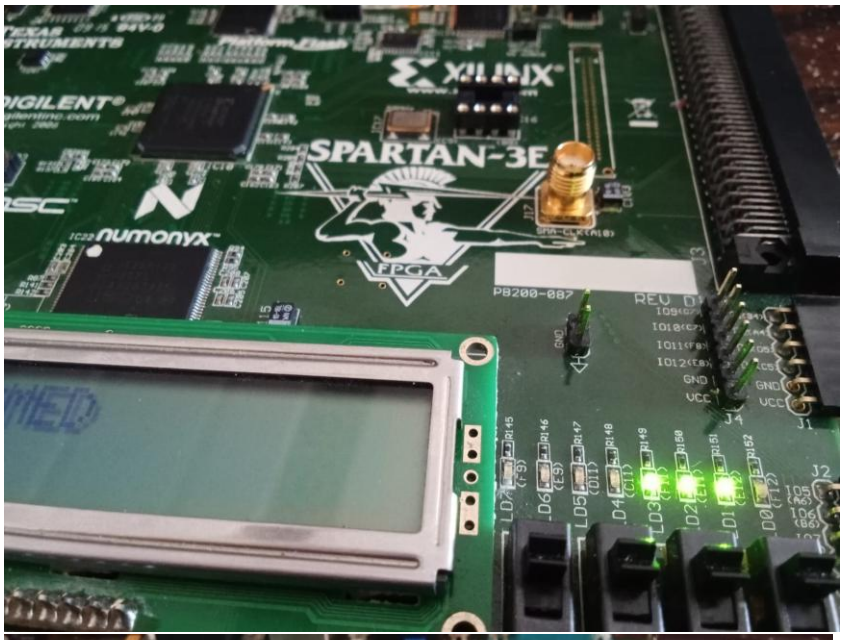
```
72          case current_state is
73              when IDLE =>
74                  if D_in = '1' then
75                      next_state <= dis;
76                  elsif Q_in = '1' then
77                      next_state <= Q25;
78                  end if;
79
80                  when Q25 =>
81                  if D_in = '1' then
82                      next_state <= dis;
83                  elsif Q_in = '1' then
84                      next_state <= Q50;
85                  end if;
86
87              when Q50 =>
88                  if D_in = '1' then
89                      next_state <= dis;
90                  elsif Q_in = '1' then
91                      next_state <= Q75;
92                  end if;
93
94              when Q75 =>
95                  if D_in = '1' then
96                      next_state <= dis;
97                  end if;
98
99              when dis =>
100                 internal_dispense <= '1';
101                 internal_change   <= '1';
102                 next_state        <= IDLE;
103
104             when others =>
105                 next_state <= IDLE;
106         end case;
107     end process;
108
109
110 end Behavioral;
111
112
```

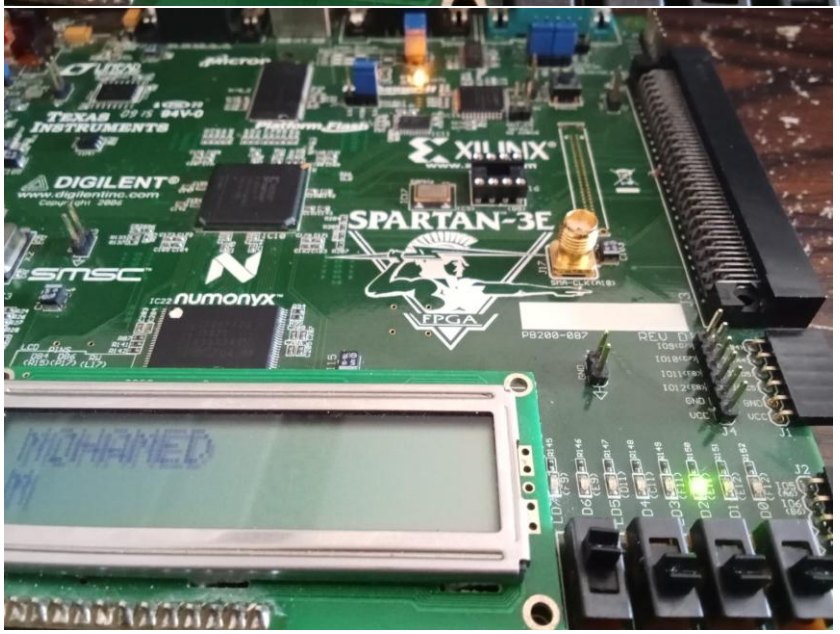


3) Test Bench :

```
86
87      stimulus: process
88      begin
89          reset <= '1';
90          wait for 20 ns;
91          reset <= '0';
92
93          Q_in <= '1';
94          wait for 20 ns;
95          Q_in <= '0';
96          wait for 20 ns;
97
98          Q_in <= '1';
99          wait for 20 ns;
100         Q_in <= '0';
101         wait for 20 ns;
102
103         D_in <= '1';
104         wait for 20 ns;
105         D_in <= '0';
106
107         wait;
108     end process;
109
110
111
112
113 END;
```

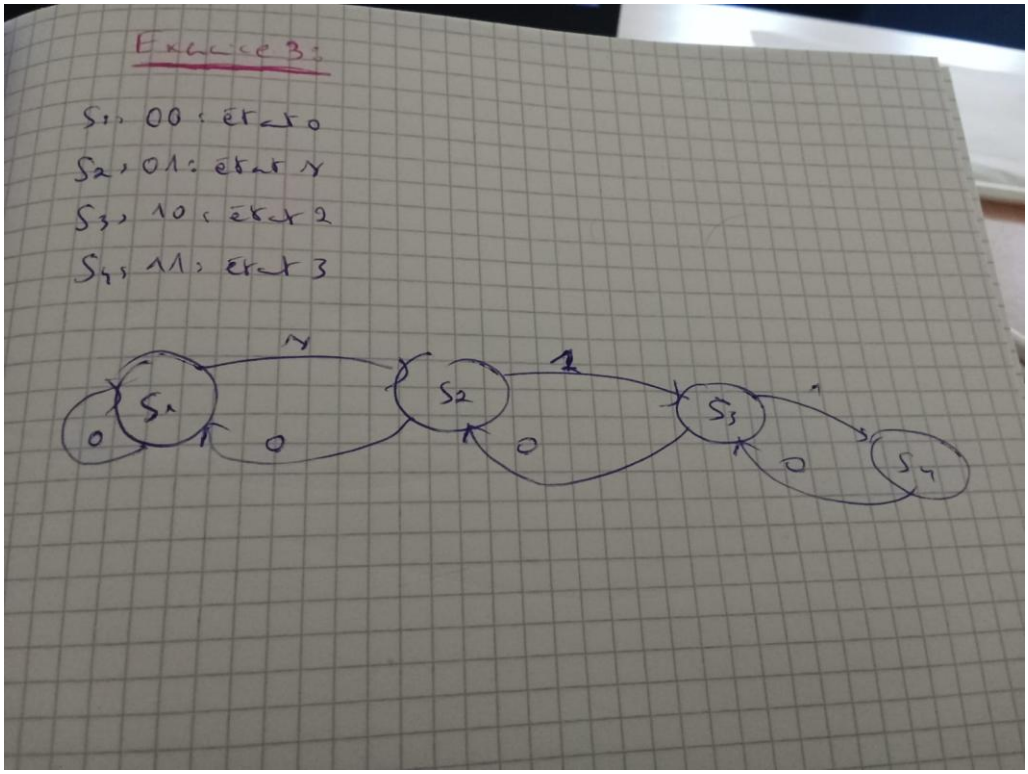



XILINX
SPARTAN-3E
FPGA
numonyx
DIGILENT
PB200-087


XILINX
SPARTAN-3E
FPGA
numonyx
DIGILENT
smsc
PB200-087
MOHAMED

Exercice 3 :

1)



2)

```
 6  entity moore is
 7  Port (
 8          clk    : in  STD_LOGIC;
 9          reset  : in  STD_LOGIC;
10          up     : in  STD_LOGIC;
11          cnt    : out STD_LOGIC_VECTOR(1 downto 0)
12      );
13  end moore;
14
15
16  architecture Behavioral of moore is
17  signal count : STD_LOGIC_VECTOR(1 downto 0) := "00";
18  signal count_int : unsigned(1 downto 0);
19  begin
20      count_int <= unsigned(count);
21   process(clk, reset)
22      begin
23          if reset = '1' then
24              count_int <= (others => '0');
25          elsif rising_edge(clk) then
26              if up = '1' then
27                 count_int <= count_int + 1;
28              else
29                  count_int <= count_int - 1;
30              end if;
31          end if;
32      end process;
33      cnt <= std_logic_vector(count_int);
34  end Behavioral;
```

3) Test Bench :

```
 79   stimulus: process
 80     begin
 81         reset <= '1';
 82         wait for 20 ns;
 83         reset <= '0';
 84         wait for 20 ns;
 85
 86         up <= '1';
 87         wait for 40 ns;
 88         up <= '0';
 89         wait for 40 ns;
 90
 91         up <= '0';
 92         wait for 40 ns;
 93         up <= '1';
 94         wait for 40 ns;
 95
 96         up <= '1';
 97         wait for 40 ns;
 98         up <= '0';
 99         wait for 40 ns;
100
101         wait;
102     end process;
103
104 END;
```