```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   Entity MEALY_SM IS Port
6   (
7            clk_input,X_MOTION, Y_MOTION, Extender_Out,X_EQ,X_GT,X_LT,Y_EQ,Y_GT,Y_LT    : in
    std_logic;
8            rst_n                                                                      : in
    std_logic  := '0';
9            clk_en_X,clk_en_Y, Xcount,Ycount,ERROR_led,Extender_en                     : out
    std_logic
10   );
11  END ENTITY;
12
13   Architecture SM of MEALY_SM is
14
15   TYPE STATE_NAME IS (INIT,MOVE,STOP,ERROR);   -- All of the STATE_NAME values which are
    initial, moving, stopped and error
16
17   SIGNAL current_state, next_state   :  STATE_NAME;       -- signal of the type STATE_NAME
18
19   BEGIN
20   --------------------------------------------------------------------------------
21   --State Machine:
22   --------------------------------------------------------------------------------
23
24   -- REGISTER_LOGIC PROCESS:
25
26  Register_Section: PROCESS (clk_input, rst_n, next_state)  -- this process synchronizes the
    activity to a clock
27  BEGIN
28     IF (rst_n = '0') THEN
29        current_state <= INIT;  -- current_state will be inital state when we reset the clock
30     ELSIF(rising_edge(clk_input)) THEN
31        current_state <= next_state;  -- current_state will become next_state on rising edge
    of the clock
32     END IF;
33  END PROCESS;
34
35  -- Here X_MOTION is pb(3) and Y_MOTION is pb(2)
36
37  -- TRANSITION LOGIC PROCESS
38
39  Transition_Section: PROCESS (current_state, X_MOTION, Y_MOTION, X_EQ, X_GT,X_LT, Y_EQ, Y_GT,
     Y_LT, Extender_Out) -- Sensitivity list
40
41  BEGIN
42
43       CASE current_state IS
44
45           WHEN INIT =>
46              next_state <= INIT;
47              IF((X_MOTION = '0' OR Y_MOTION = '0') AND X_EQ = '0' AND Y_EQ = '0' AND
    Extender_Out = '0')  THEN
48                 next_state <= MOVE;     -- If either pb(3) or pb(2) is pressed, along with
    both Y and X equal to 0 and the extender is not out then we should move the digits to reach
    targeted value
49              ELSIF (X_EQ = '1' AND Y_EQ = '1') THEN
50                 next_state <= STOP;     -- If the X and Y equal to 1, then we must stop the
    changing values on the digits
51              END IF;
52
53           WHEN MOVE =>
54              IF(Extender_Out = '1') THEN
55                 next_state <= ERROR;    -- If the extender is out, then we must get an error
    while trying to change the coordinates
56              ELSIF ((X_MOTION='0' OR Y_MOTION = '0') AND (X_EQ = '0' AND Y_EQ = '0')) THEN
57                 next_state <= MOVE;     -- If either of the push buttons are pressed and X
    and Y are not equal, then we must keep on going till we reach the target
58              ELSIF(X_MOTION='1' AND Y_MOTION = '1' AND X_EQ = '1' AND Y_EQ = '1') THEN
59                 next_state <= STOP;     -- If not pb is pressed and X and Y are equal to 1,
    then we must stop
60              END IF;
61
62           WHEN STOP =>
```

```
63                next_state <= STOP;
64                IF(X_MOTION='0' OR Y_MOTION = '0') THEN
65                   next_state <= MOVE;    -- If we press the pb again, then we must start moving
66                END IF;
67
68             WHEN ERROR =>
69                next_state <= ERROR;
70                IF(Extender_Out = '0') THEN
71                   next_state <= STOP;    -- If extender is not out, then we must stop
72                END IF;
73
74          END CASE;
75     END PROCESS;
76
77     -- DECODER SECTION PROCESS
78
79   Decoder_Section: PROCESS (current_state, X_MOTION, Y_MOTION, X_EQ, X_GT,X_LT, Y_EQ, Y_GT,
     Y_LT, Extender_Out)
80
81   BEGIN
82        CASE current_state IS
83
84             WHEN INIT =>
85                Extender_en <= '0';   -- In the initial stage, giving the extender enable a
     value of 0
86
87             WHEN MOVE =>
88                Extender_en <= '0';
89                clk_en_X <= '1';     -- enabling X for the Up/Down binary counter
90                clk_en_Y <= '1';     -- enabling Y for the Up/Down binary counter
91                IF(X_MOTION='0' AND X_GT = '1') THEN
92                   Xcount <= '0';   -- If pb(3) is pressed and current coordinates are greater
     than targeted, so we must go down to get to the targeted one, that is why Xcount = 0
93                ELSIF(X_MOTION='1' OR X_EQ = '1') THEN
94                   clk_en_X <= '0'; -- If it becomes equal, then we disable the enabler
95                ELSIF(X_MOTION='0' AND X_LT = '1') THEN
96                   Xcount <= '1';   -- If pb(3) is pressed and current coordinates are less
     than targeted, so we must go up to get to the targeted one, that is why Xcount = 1
97                END IF;
98                -- Similarly for Y coordinates
99                IF(Y_MOTION='0' AND Y_GT = '1') THEN
100                  Ycount <= '0';
101               ELSIF(Y_MOTION='1' OR Y_EQ = '1') THEN
102                  clk_en_Y <= '0';
103               ELSIF (Y_MOTION='0' AND Y_LT = '1') THEN
104                  Ycount <= '1';
105               END IF;
106
107            WHEN STOP =>
108               Extender_en <= '0';  -- Keeping its value 0
109               IF (X_EQ = '1' AND Y_EQ = '1') THEN
110                  Extender_en <= '1';   -- Only when the X and Y coordinates are in their
     targeted values, we can open the extender
111               END IF;
112
113            WHEN ERROR =>
114               ERROR_led <= '1';   -- When in error state, the value of led should be 1
115               IF(Extender_Out = '0') THEN
116                  ERROR_led <= '0';  -- If extender is not out, then led should not light up
117               END IF;
118
119         END CASE;
120   END PROCESS;
121
122    END ARCHITECTURE SM;
```