

```
1  -----
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    11:55:58 03/22/2018
6  -- Design Name:
7  -- Module Name:    fsm_1 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 entity fsm_1 is
33     Port ( x : in  STD_LOGIC;
34           clk : in  STD_LOGIC;
35           z : out  STD_LOGIC);
36 end fsm_1;
37
38 architecture Behavioral of fsm_1 is
39
40     architecture Behavioral of testfsm is
41         type state_type is (state_a, state_b, state_c, state_d, state_e, state_f,
42                             state_g, state_h, state_i, state_j, state_k, state_l,
43                             state_m, state_n, state_o, state_p, state_q, state_r);
44         signal present_state : state_type := state_a;
45         signal next_state : state_type := state_a;
46
47
48     begin
49
50         State: Process (present_state, X)
51         begin
52
53             If x='0' THEN
54                 Case present_state is
55                     When state_a => next_state <=state_b; z <= '0';
56                     When state_b => next_state <=state_e; z <= '0';
57                     When state_c => next_state <=state_c; z <= '0';
```

```
58         When state_d => next_state <=state_e; z <= '0';
59         When state_e => next_state <=state_n; z <= '0';
60         When state_f => next_state <=state_g; z <= '0';
61         When state_g => next_state <=state_h; z <= '0';
62         When state_h => next_state <=state_i; z <= '0';
63         When state_i => next_state <=state_m; z <= '0';
64         When state_j => next_state <=state_m; z <= '0';
65         When state_k => next_state <=state_n; z <= '0';
66         When state_l => next_state <=state_l; z <= '0';
67         When state_m => next_state <=state_m; z <= '0';
68         When state_n => next_state <=state_o; z <= '0';
69         When state_o => next_state <=state_p; z <= '0';
70         When state_p => next_state <=state_q; z <= '0';
71         When state_q => next_state <=state_q; z <= '0';
72         When state_r => next_state <=state_r; z <= '1';
73         When OTHERS => NULL;
74     end case;
75
76     else If x='1' THEN
77         Case present_state is
78             When state_a => next_state <=state_a; z <= '0';
79             When state_b => next_state <=state_b; z <= '0';
80             When state_c => next_state <=state_d; z <= '0';
81             When state_d => next_state <=state_j; z <= '0';
82             When state_e => next_state <=state_f; z <= '0';
83             When state_f => next_state <=state_g; z <= '0';
84             When state_g => next_state <=state_k; z <= '0';
85             When state_h => next_state <=state_n; z <= '0';
86             When state_i => next_state <=state_j; z <= '0';
87             When state_j => next_state <=state_j; z <= '0';
88             When state_k => next_state <=state_p; z <= '0';
89             When state_l => next_state <=state_k; z <= '0';
90             When state_m => next_state <=state_l; z <= '0';
91             When state_n => next_state <=state_o; z <= '0';
92             When state_o => next_state <=state_o; z <= '0';
93             When state_p => next_state <=state_q; z <= '0';
94             When state_q => next_state <=state_r; z <= '0';
95             When state_r => next_state <=state_r; z <= '1';
96
97             When OTHERS => NULL;
98         end case;
99     end if; -- end else if
100 end if;
101
102 end Process State;
103
104
105 clocked: Process (Clk)
106     begin
107         If((Clk='1') and (Clk'last_value='0')) Then
108             present_state <= next_state;
109         end if;
110     end Process clocked;
111
112
113
114 end Behavioral;
```

115

116

```
1  -----
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    11:55:58 03/22/2018
6  -- Design Name:
7  -- Module Name:    fsm_1 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 entity fsm_1 is
33     Port ( x : in  STD_LOGIC;
34           clk : in  STD_LOGIC;
35           z : out  STD_LOGIC);
36 end fsm_1;
37
38 architecture Behavioral of fsm_1 is
39
40     architecture Behavioral of testfsm is
41         type state_type is (state_a, state_b, state_c, state_d, state_e, state_f,
42                             state_g, state_h, state_i, state_j, state_k, state_l,
43                             state_m, state_n, state_o, state_p, state_q, state_r);
44         signal present_state : state_type := state_a;
45         signal next_state : state_type := state_a;
46
47
48     begin
49
50         State: Process (present_state, X)
51         begin
52
53             If x='0' THEN
54                 Case present_state is
55                     When state_a => next_state <=state_b; z <= '0';
56                     When state_b => next_state <=state_e; z <= '0';
57                     When state_c => next_state <=state_c; z <= '0';
```

```
58         When state_d => next_state <=state_e; z <= '0';
59         When state_e => next_state <=state_n; z <= '0';
60         When state_f => next_state <=state_g; z <= '0';
61         When state_g => next_state <=state_h; z <= '0';
62         When state_h => next_state <=state_i; z <= '0';
63         When state_i => next_state <=state_m; z <= '0';
64         When state_j => next_state <=state_m; z <= '0';
65         When state_k => next_state <=state_n; z <= '0';
66         When state_l => next_state <=state_l; z <= '0';
67         When state_m => next_state <=state_m; z <= '0';
68         When state_n => next_state <=state_o; z <= '0';
69         When state_o => next_state <=state_p; z <= '0';
70         When state_p => next_state <=state_q; z <= '0';
71         When state_q => next_state <=state_q; z <= '0';
72         When state_r => next_state <=state_r; z <= '1';
73         When OTHERS => NULL;
74     end case;
75
76     else If x='1' THEN
77         Case present_state is
78             When state_a => next_state <=state_a; z <= '0';
79             When state_b => next_state <=state_b; z <= '0';
80             When state_c => next_state <=state_d; z <= '0';
81             When state_d => next_state <=state_j; z <= '0';
82             When state_e => next_state <=state_f; z <= '0';
83             When state_f => next_state <=state_g; z <= '0';
84             When state_g => next_state <=state_k; z <= '0';
85             When state_h => next_state <=state_n; z <= '0';
86             When state_i => next_state <=state_j; z <= '0';
87             When state_j => next_state <=state_j; z <= '0';
88             When state_k => next_state <=state_p; z <= '0';
89             When state_l => next_state <=state_k; z <= '0';
90             When state_m => next_state <=state_l; z <= '0';
91             When state_n => next_state <=state_o; z <= '0';
92             When state_o => next_state <=state_o; z <= '0';
93             When state_p => next_state <=state_q; z <= '0';
94             When state_q => next_state <=state_r; z <= '0';
95             When state_r => next_state <=state_r; z <= '1';
96
97             When OTHERS => NULL;
98         end case;
99     end if; -- end else if
100 end if;
101
102 end Process State;
103
104
105 clocked: Process (Clk)
106     begin
107         If((Clk='1') and (Clk'last_value='0')) Then
108             present_state <= next_state;
109         end if;
110     end Process clocked;
111
112
113
114 end Behavioral;
```

115

116

```
1  -----
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    11:55:58 03/22/2018
6  -- Design Name:
7  -- Module Name:    fsm_1 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 entity fsm_1 is
33     Port ( x : in  STD_LOGIC;
34           clk : in  STD_LOGIC;
35           z : out  STD_LOGIC);
36 end fsm_1;
37
38 architecture Behavioral of fsm_1 is
39
40     architecture Behavioral of testfsm is
41         type state_type is (state_a, state_b, state_c, state_d, state_e, state_f,
42                             state_g, state_h, state_i, state_j, state_k, state_l,
43                             state_m, state_n, state_o, state_p, state_q, state_r);
44         signal present_state : state_type := state_a;
45         signal next_state : state_type := state_a;
46
47
48     begin
49
50         State: Process (present_state, X)
51         begin
52
53             If x='0' THEN
54                 Case present_state is
55                     When state_a => next_state <=state_b; z <= '0';
56                     When state_b => next_state <=state_e; z <= '0';
57                     When state_c => next_state <=state_c; z <= '0';
```

```
58         When state_d => next_state <=state_e; z <= '0';
59         When state_e => next_state <=state_n; z <= '0';
60         When state_f => next_state <=state_g; z <= '0';
61         When state_g => next_state <=state_h; z <= '0';
62         When state_h => next_state <=state_i; z <= '0';
63         When state_i => next_state <=state_m; z <= '0';
64         When state_j => next_state <=state_m; z <= '0';
65         When state_k => next_state <=state_n; z <= '0';
66         When state_l => next_state <=state_l; z <= '0';
67         When state_m => next_state <=state_m; z <= '0';
68         When state_n => next_state <=state_o; z <= '0';
69         When state_o => next_state <=state_p; z <= '0';
70         When state_p => next_state <=state_q; z <= '0';
71         When state_q => next_state <=state_q; z <= '0';
72         When state_r => next_state <=state_r; z <= '1';
73         When OTHERS => NULL;
74     end case;
75
76     else If x='1' THEN
77         Case present_state is
78             When state_a => next_state <=state_a; z <= '0';
79             When state_b => next_state <=state_b; z <= '0';
80             When state_c => next_state <=state_d; z <= '0';
81             When state_d => next_state <=state_j; z <= '0';
82             When state_e => next_state <=state_f; z <= '0';
83             When state_f => next_state <=state_g; z <= '0';
84             When state_g => next_state <=state_k; z <= '0';
85             When state_h => next_state <=state_n; z <= '0';
86             When state_i => next_state <=state_j; z <= '0';
87             When state_j => next_state <=state_j; z <= '0';
88             When state_k => next_state <=state_p; z <= '0';
89             When state_l => next_state <=state_k; z <= '0';
90             When state_m => next_state <=state_l; z <= '0';
91             When state_n => next_state <=state_o; z <= '0';
92             When state_o => next_state <=state_o; z <= '0';
93             When state_p => next_state <=state_q; z <= '0';
94             When state_q => next_state <=state_r; z <= '0';
95             When state_r => next_state <=state_r; z <= '1';
96
97             When OTHERS => NULL;
98         end case;
99     end if; -- end else if
100 end if;
101
102 end Process State;
103
104
105 clocked: Process (Clk)
106     begin
107         If((Clk='1') and (Clk'last_value='0')) Then
108             present_state <= next_state;
109         end if;
110     end Process clocked;
111
112
113
114 end Behavioral;
```

115

116

```
1  -----
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    11:55:58 03/22/2018
6  -- Design Name:
7  -- Module Name:    fsm_1 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 entity fsm_1 is
33     Port ( x : in  STD_LOGIC;
34           clk : in  STD_LOGIC;
35           z : out  STD_LOGIC);
36 end fsm_1;
37
38 architecture Behavioral of fsm_1 is
39
40     architecture Behavioral of testfsm is
41         type state_type is (state_a, state_b, state_c, state_d, state_e, state_f,
42                             state_g, state_h, state_i, state_j, state_k, state_l,
43                             state_m, state_n, state_o, state_p, state_q, state_r);
44         signal present_state : state_type := state_a;
45         signal next_state : state_type := state_a;
46
47
48     begin
49
50         State: Process (present_state, X)
51         begin
52
53             If x='0' THEN
54                 Case present_state is
55                     When state_a => next_state <=state_b; z <= '0';
56                     When state_b => next_state <=state_e; z <= '0';
57                     When state_c => next_state <=state_c; z <= '0';
```

```
58         When state_d => next_state <=state_e; z <= '0';
59         When state_e => next_state <=state_n; z <= '0';
60         When state_f => next_state <=state_g; z <= '0';
61         When state_g => next_state <=state_h; z <= '0';
62         When state_h => next_state <=state_i; z <= '0';
63         When state_i => next_state <=state_m; z <= '0';
64         When state_j => next_state <=state_m; z <= '0';
65         When state_k => next_state <=state_n; z <= '0';
66         When state_l => next_state <=state_l; z <= '0';
67         When state_m => next_state <=state_m; z <= '0';
68         When state_n => next_state <=state_o; z <= '0';
69         When state_o => next_state <=state_p; z <= '0';
70         When state_p => next_state <=state_q; z <= '0';
71         When state_q => next_state <=state_q; z <= '0';
72         When state_r => next_state <=state_r; z <= '1';
73         When OTHERS => NULL;
74     end case;
75
76     else If x='1' THEN
77         Case present_state is
78             When state_a => next_state <=state_a; z <= '0';
79             When state_b => next_state <=state_b; z <= '0';
80             When state_c => next_state <=state_d; z <= '0';
81             When state_d => next_state <=state_j; z <= '0';
82             When state_e => next_state <=state_f; z <= '0';
83             When state_f => next_state <=state_g; z <= '0';
84             When state_g => next_state <=state_k; z <= '0';
85             When state_h => next_state <=state_n; z <= '0';
86             When state_i => next_state <=state_j; z <= '0';
87             When state_j => next_state <=state_j; z <= '0';
88             When state_k => next_state <=state_p; z <= '0';
89             When state_l => next_state <=state_k; z <= '0';
90             When state_m => next_state <=state_l; z <= '0';
91             When state_n => next_state <=state_o; z <= '0';
92             When state_o => next_state <=state_o; z <= '0';
93             When state_p => next_state <=state_q; z <= '0';
94             When state_q => next_state <=state_r; z <= '0';
95             When state_r => next_state <=state_r; z <= '1';
96
97             When OTHERS => NULL;
98         end case;
99     end if; -- end else if
100 end if;
101
102 end Process State;
103
104
105 clocked: Process (Clk)
106     begin
107         If((Clk='1') and (Clk'last_value='0')) Then
108             present_state <= next_state;
109         end if;
110     end Process clocked;
111
112
113
114 end Behavioral;
```

115

116