جامعة نينوى
كلية هندسة الإلكترونيات

Circuit Design with VHDL 11

Textbook: Volnei A. Pedroni

Submitted By: Hussein Aideen

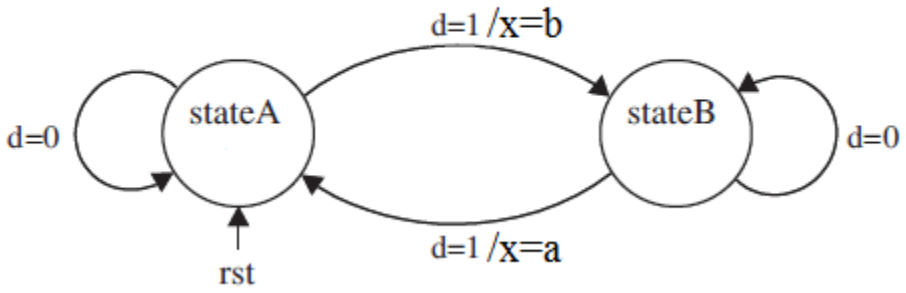
VHDL> State Machines

- Finite state machines (FSM) constitute a special modeling technique for sequential logic circuits.

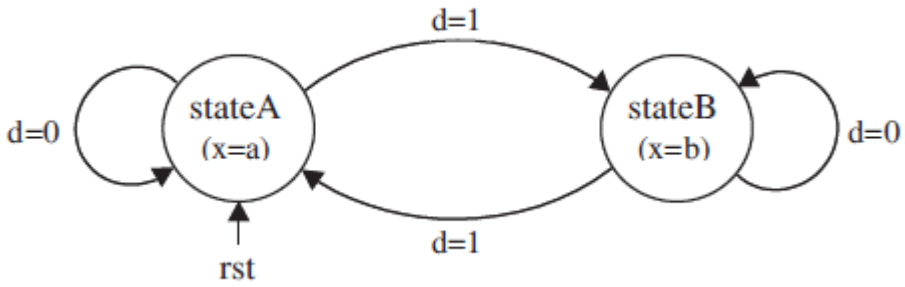
- helpful in the design of certain types of systems, (digital controllers, counters, for example).

VHDL> State Machines

- Mealy machine: the output of the machine depends not only on the present state but also on the current input.



- Moore machine: the output depends only on the current state.



VHDL> State Machines

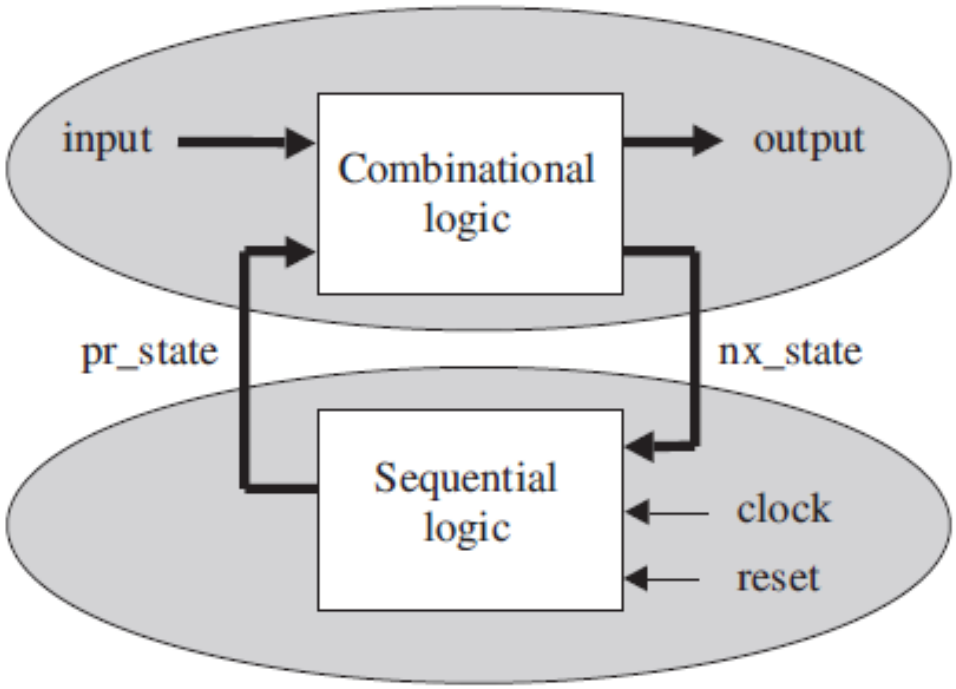


Figure 8.1
Mealy (Moore) state machine diagram.

VHDL> State Machines

Design Style #1:

- the design of the lower section is completely separated from that of the upper section.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
---------------------------------------------
ENTITY <entity_name> IS
   PORT ( input: IN <data_type>;
          reset, clock: IN STD_LOGIC;
          output: OUT <data_type>);
END <entity_name>;
---------------------------------------------
ARCHITECTURE <arch_name> OF <entity_name> IS
   TYPE state IS (state0, state1, state2, state3, ...);
   SIGNAL pr_state, nx_state: state;
BEGIN
```

VHDL> State Machines

Design of the Lower (Sequential) Section:

```
PROCESS (reset, clock)
BEGIN
    IF (reset='1') THEN
       pr_state <= state0;
    ELSIF (clock'EVENT AND clock='1') THEN
       pr_state <= nx_state;
    END IF;
END PROCESS;
```

VHDL> State Machines

Design of the Upper (Combinational) Section:

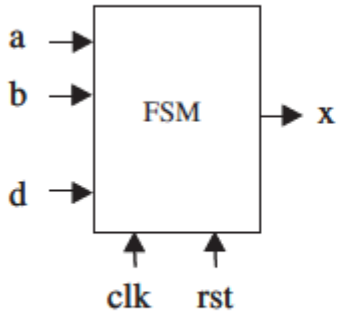
```
PROCESS (input, pr_state)
BEGIN
   CASE pr_state IS
      WHEN state0 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state1;
         ELSE ...
         END IF;
      WHEN state1 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state2;
         ELSE ...
         END IF;
      WHEN state2 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state3;
         ELSE ...
         END IF;
      ...
   END CASE;
END PROCESS;
END <arch_name>;
```

VHDL> State Machines

Example

Example 8.2: Simple FSM #1

Figure 8.4 shows the states diagram of a very simple FSM. The system has two states (stateA and stateB), and must change from one to the other every time d = '1' is received. The desired output is x = a when the machine is in stateA, or x = b when in stateB. The initial (reset) state is stateA.



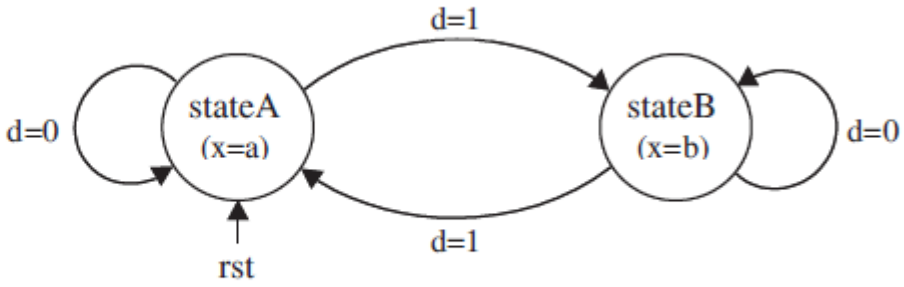


Figure 8.4
State machine of example 8.1.

VHDL> State Machines

```
1   ---------------------
2   ENTITY simple_fsm IS

3      PORT ( a, b, d, clk, rst: IN BIT;
4             x: OUT BIT);
5   END simple_fsm;
6   ----------------------------------------
7   ARCHITECTURE simple_fsm OF simple_fsm IS
8      TYPE state IS (stateA, stateB);
9      SIGNAL pr_state, nx_state: state;
10  BEGIN
11     ----- Lower section: -----------------
12     PROCESS (rst, clk)
13     BEGIN
14        IF (rst='1') THEN
15           pr_state <= stateA;
16        ELSIF (clk'EVENT AND clk='1') THEN
17           pr_state <= nx_state;
18        END IF;
19     END PROCESS;
```

VHDL> State Machines

```
20     ---------- Upper section: -----------------
21     PROCESS (a, b, d, pr_state)
22     BEGIN
23        CASE pr_state IS
24           WHEN stateA =>
25              x <= a;
26              IF (d='1') THEN nx_state <= stateB;
27              ELSE nx_state <= stateA;
28              END IF;
29           WHEN stateB =>
30              x <= b;
31              IF (d='1') THEN nx_state <= stateA;
32              ELSE nx_state <= stateB;
33              END IF;
34        END CASE;
35     END PROCESS;
36 END simple_fsm;
```

VHDL> State Machines

Design Style #2 (Stored Output):

- In #1: Notice that in this case, if it is a Mealy machine (one whose output is dependent on the current input), the output might change when the input changes (asynchronous output).

- To make Mealy machines synchronous.

VHDL> State Machines

Design Style #2 (Stored Output):

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------------
ENTITY <ent_name> IS
   PORT (input: IN <data_type>;
         reset, clock: IN STD_LOGIC;
         output: OUT <data_type>);
END <ent_name>;
-------------------------------------------------------
ARCHITECTURE <arch_name> OF <ent_name> IS
   TYPE states IS (state0, state1, state2, state3, ...);
   SIGNAL pr_state, nx_state: states;
   SIGNAL temp: <data_type>;
BEGIN
   ---------- Lower section: ------------------------
   PROCESS (reset, clock)
   BEGIN
      IF (reset='1') THEN
         pr_state <= state0;
      ELSIF (clock'EVENT AND clock='1') THEN
         output <= temp;
         pr_state <= nx_state;
      END IF;
   END PROCESS;
```

VHDL> State Machines

Design Style #2 (Stored Output):

```
---------- Upper section: -------------------------
PROCESS (pr_state)
BEGIN
   CASE pr_state IS
      WHEN state0 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state1;
         ...
         END IF;
      WHEN state1 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state2;
         ...
         END IF;
      WHEN state2 =>
         temp <= <value>;
         IF (condition) THEN nx_state <= state3;
         ...
         END IF;
      ...
   END CASE;
END PROCESS;
END <arch_name>;
```