Design A: Moore-type FSM

- Sequence detector: Detects input test vector that contains the sequence of 100.
- Mealy state diagram
- VHDL code (you need to modify the code for Lab 10 to detect “100”)

FSM: Moore-type

- User defined signal:
  - TYPE State_type IS (A, B, C) ;

    SIGNAL y : State_type;
  - Automatically indexing
- FSM: Lines 11-37
  - Asynchronous RESETN
  - First combination circuit and flip-flop (together)
- Output: Concurrent statement (second combination circuit)

```
1   LIBRARY ieee ;
2   USE ieee.std_logic_1164.all ;

3   ENTITY simple IS
4           PORT (      Clock, Resetn, w     : IN     STD_LOGIC ;
                        z                    : OUT    STD_LOGIC ) ;
5   END simple ;

7   ARCHITECTURE Behavior OF simple IS
8           TYPE State_type IS (A, B, C) ;
9           SIGNAL y : State_type ;
10  BEGIN
11          PROCESS ( Resetn, Clock )
12          BEGIN
13                  IF Resetn = '0' THEN
14                          y <= A ;
15                  ELSIF (Clock'EVENT AND Clock = '1') THEN
16                          CASE y IS
17                                  WHEN A =>
18                                          IF w = '0' THEN
19                                                  y <= A ;
20                                          ELSE
21                                                  y <= B ;
22                                          END IF ;
23                                  WHEN B =>
24                                          IF w = '0' THEN
25                                                  y <= A ;
26                                          ELSE
27                                                  y <= C ;
28                                          END IF ;
29                                  WHEN C =>
30                                          IF w = '0' THEN
31                                                  y <= A ;
32                                          ELSE
33                                                  y <= C ;
34                                          END IF ;
35                          END CASE ;
36                  END IF ;
37          END PROCESS ;
38          z <= '1' WHEN y = C ELSE '0' ;
39  END Behavior ;
```

FSM: Moore-type (another style)

```
ARCHITECTURE Behavior OF simple IS
    TYPE State_type IS (A, B, C) ;
    SIGNAL y_present, y_next : State_type ;
BEGIN
    PROCESS ( w, y_present )
    BEGIN
        CASE y_present IS
            WHEN A =>
                IF w = '0' THEN
                    y_next <= A ;
                ELSE
                    y_next <= B ;
                END IF ;
            WHEN B =>
                IF w = '0' THEN
                    y_next <= A ;
                ELSE
                    y_next <= C ;
                END IF ;
            WHEN C =>
                IF w = '0' THEN
                    y_next <= A ;
                ELSE
                    y_next <= C ;
                END IF ;
        END CASE ;
    END PROCESS ;
```

Continuing at the right column…

- Separate processes for first combination circuit and flip-flops
- Output: Concurrent statement (second combination circuit)

```
    PROCESS (Clock, Resetn)
    BEGIN
        IF Resetn = '0' THEN
            y_present <= A ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            y_present <= y_next ;
            END IF ;
        END PROCESS ;
        z <= '1' WHEN y_present = C ELSE '0' ;
END Behavior ;
```

Design B: Mealy-type FSM

- Sequence detector: Detects input test vector that contains the sequence of 100.
- Mealy state diagram
- VHDL code (you need to modify the code for Lab 10 to detect “100”)

FSM: Mealy-type

- First process
  - Asynchronous RESETN
  - First combination circuit and D flip-flops
- Second process:
  - Second combination circuit: output

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY mealy IS
      PORT ( Clock, Resetn, w   : IN   STD_LOGIC ;
                  z                      : OUT    STD_LOGIC ) ;
END mealy ;
ARCHITECTURE Behavior OF mealy IS
      TYPE State_type IS (A, B) ;
      SIGNAL y : State_type ;
BEGIN
      PROCESS ( Resetn, Clock )
      BEGIN
            IF Resetn = '0' THEN
                  y <= A ;
            ELSIF (Clock'EVENT AND Clock = '1') THEN
                  CASE y IS
                        WHEN A =>
                              IF w = '0' THEN y <= A ;
                              ELSE y <= B ;
                              END IF ;
                         WHEN B =>
                              IF w = '0' THEN y <= A ;
                              ELSE y <= B ;
                              END IF ;
                  END CASE ;
            END IF ;
      END PROCESS ;
      PROCESS ( y, w )
      BEGIN
            CASE y IS
               WHEN A =>
                  z <= '0' ;
               WHEN B =>
                  z <= w ;
            END CASE ;
      END PROCESS ;
END Behavior ;
```