ECE380 Digital Logic

Design of Finite State Machines
Using CAD Tools

FSM design using CAD tools

- VHDL provides a number of constructs for designing finite state machines
- There is not a standard way for defining an FSM
- Basic approach
  - Create a user-defined data type to represent the possible states of an FSM
  - This signal represents the outputs (state variables) of the flip-flops that implement the states in the FSM
  - VHDL compiler chooses the appropriate number of flip-flops during the synthesis process
  - The state assignment can be done by the compiler or can be user specified

User defined data types

- The **TYPE** keyword will be used to define a new data type used to represent states in the FSM

$$\underbrace{\textbf{TYPE}}_{\text{A user-defined data type definition}}\ \underbrace{\textbf{State\_type}}_{\text{Data type name}}\ \textbf{IS}\ (\underbrace{\textbf{A, B, C}}_{\text{Valid values for the data type}});$$

Defines a data type (called State_type) that can take on three distinct values: A, B, or C).

Representing states

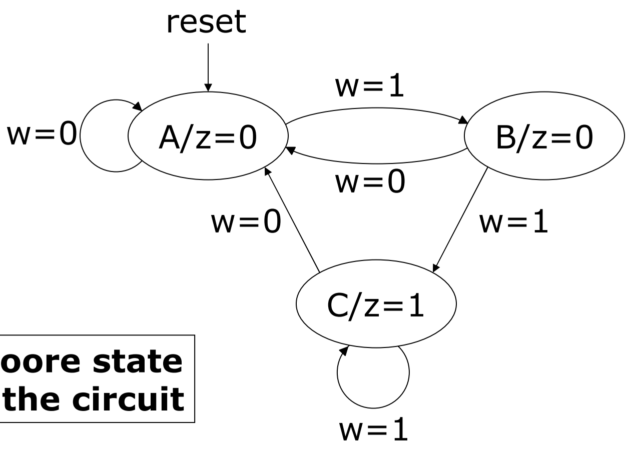
- A **SIGNAL** is defined, of the user-defined **State_type**, to represent the flip-flop outputs

```
TYPE State_type IS (A, B, C);
SIGNAL y: State_type;
```

The signal, y, can be used to represent the flip-flop outputs for an FSM that has three states

Design example

- Create a VHDL description for a circuit that detects a ’11’ input sequence on an input, **w**



Recall the Moore state diagram for the circuit

VHDL design example

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY detect IS
   PORT(      clk, resetn, w : IN    STD_LOGIC ;
              z              : OUT  STD_LOGIC) ;
END detect ;

ARCHITECTURE Behavior OF detect IS
   TYPE State_type IS (A,B,C) ;
   SIGNAL y: State_type ;
BEGIN
```

VHDL design example continued

```
PROCESS ( resetn, clk )
BEGIN
  IF resetn = ‘0’ THEN
    y <= A;
  ELSIF (clk’EVENT AND clk=‘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 <= C;
        END IF;
      WHEN C =>
        IF w=‘0’ THEN
          y <= A;
        ELSE
          y <= C;
        END IF;
    END CASE;
  END IF;
END PROCESS
z <= ‘1’ WHEN y=C ELSE ‘0’;
END Behavior;
```

Alternative style of VHDL code

- Another form for describing the circuit in VHDL is to define two signals to represent the state of the FSM
  - One signal, **y_present**, defines the present state of the FSM
  - The second, **y_next**, defines the next state of the machine
- This notation follows the **y** (present state) and **Y** (next state) notation used previously
- Two PROCESS statements will be used to describe the machine
  - The first describes the STATE TABLE as a combinational circuit
  - The second describes the flip-flops, stating that **y_present** should take on the value of **y_next** after each positive clock edge

Alternate VHDL description

```
ARCHITECTURE Behavior OF detect 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;
     WHEN B =>
          IF w='0' THEN
            y_next <= A;
          ELSE
            y_next <= C;
     WHEN C =>
          IF w='0' THEN
            y_next <= A;
          ELSE
            y_next <= C;
   END CASE;
END PROCESS;
PROCESS(clk,resetn)
BEGIN
   IF resetn='0' THEN
     y_present <= A;
   ELSIF(clk'EVENT AND clk='1') THEN
     y_present <= y_next;
   END IF;
END PROCESS
z <='1' WHEN y_present=C ELSE '0';
END Behavior;
```

Specifying a state assignment

- With the previous designs, state assignment is done by the VHDL compiler
- State assignments can be user specified using the following constructs

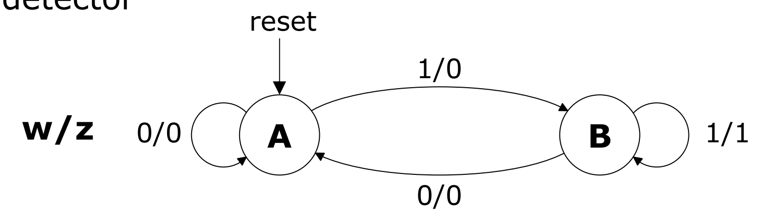
```
ARCHITECTURE Behavior OF simple IS
   TYPE State_type IS (A, B, C) ;
   ATTRIBUTE ENUM_ENCODING                    : STRING ;
   ATTRIBUTE ENUM_ENCODING OF State_type : TYPE IS "00 01 11" ;
   SIGNAL y_present, y_next : State_type ;
BEGIN
```

Or

```
ARCHITECTURE Behavior OF simple IS
   SIGNAL y_present, y_next : STD_LOGIC_VECTOR(1 DOWNTO 0);
   CONSTANT A         : STD_LOGIC_VECTOR(1 DOWNTO 0) := "00" ;
   CONSTANT B         : STD_LOGIC_VECTOR(1 DOWNTO 0) := "01" ;
   CONSTANT C         : STD_LOGIC_VECTOR(1 DOWNTO 0) := "11" ;
```

VHDL code of a Mealy FSM

- A Mealy FSM can be described in a similar manner as a Moore FSM
- The state transitions are described in the same way as the original VHDL example
- The major difference in the case of a Mealy FSM is the way in which the code for the output is written
- Recall the Mealy state diagram for the ’11’ sequence detector



Mealy ’11’ detector VHDL code

```
ARCHITECTURE Behavior OF detect IS
    TYPE State_type IS (A,B) ;
    SIGNAL y: State_type ;
BEGIN
PROCESS(resetn,clk)
BEGIN
    IF resetn=‘0’ THEN
           y <= A;
    ELSEIF(clk’EVENT AND clk=‘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;
```