In this lecture, we introduce the general structure of a digital system and state the role of finite state machine (FSM) in its operation.
There are many ways to design finite state machines as a synchronous sequential circuit. Each results in a different cost and timing.

Mealy machines offer another set of possibilities. Output generated based on both the state of the circuit and the present values of its inputs.
First Example on Mealy-type FSM: Sequence Detector

- Circuits that detect the occurrence of a particular pattern on its input(s) are referred to as sequence detectors.
- Design a sequence detector circuit that detects if two or more consecutive 1s occur on its input $w$.

<table>
<thead>
<tr>
<th>Clock cycle:</th>
<th>t₀</th>
<th>t₁</th>
<th>t₂</th>
<th>t₃</th>
<th>t₄</th>
<th>t₅</th>
<th>t₆</th>
<th>t₇</th>
<th>t₈</th>
<th>t₉</th>
<th>t₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w$:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$z$:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Step 1. Specification

- Slightly different specifications

Difference:
Original: \( z = 1 \) in the clock cycle that follows the second occurrence of \( w = 1 \)

**New:** \( z = 1 \) in the same clock cycle when the second occurrence of \( w = 1 \) is detected

<table>
<thead>
<tr>
<th>Clock cycle:</th>
<th>( t_0 )</th>
<th>( t_1 )</th>
<th>( t_2 )</th>
<th>( t_3 )</th>
<th>( t_4 )</th>
<th>( t_5 )</th>
<th>( t_6 )</th>
<th>( t_7 )</th>
<th>( t_8 )</th>
<th>( t_9 )</th>
<th>( t_{10} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w: )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( z: )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Step 2. State Diagram

State diagram of an FSM that realizes the task
### Step 3. State Table

State table for Mealy FSM

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state $w = 0$</th>
<th>Next state $w = 1$</th>
<th>Output $z w = 0$</th>
<th>Output $z w = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### Step 4. State Assignment

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w = 0$</td>
<td>$w = 1$</td>
</tr>
<tr>
<td>$y$</td>
<td>$Y$</td>
<td>$z$</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$B$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

State-assigned table for the FSM
Assuming that $y$ is realized as a D-type flip-flop, the required next-state and output expressions are

\begin{align*}
Y &= D = w \\
z &= wy
\end{align*}
Step 6. Implementation

Implementation of Mealy FSM
Timing Diagram for Mealy-type FSM
Circuit and Timing Diagram for Moore-type FSM

(a) Circuit

(b) Timing diagram

Circuit that implements the Moore-type machine specification
Second Example on Mealy-type FSM: Bus Control Circuit

- Requires 3 instead of 4 states
- Still requires 2 flip-flops
- Timing is different: Generates output control signals one clock cycle sooner
- Entire process takes 3 instead of 4 cycles for the Moore machine
Design of FSM Using VHDL

Rudimentary method:

• Use the design technique previously described to design the next state logic, output logic and flip-flop circuits

• Use schematic diagrams or structural VHDL to describe these circuits. E.g. use components for the flip flops, etc…

• Use CAD tools to simulate the behavior

• Use CAD tools to automatically implement the circuit in a chip, such as a PLD
Behavioral Description

- Design the state diagram and let the CAD tools synthesize a FSM from it.

- CAD tools take care of synthesis and optimizations (state assignment, logic/state minimization).

- Enter the state diagram into CAD tools

- Draw using a graphical editor

- Use *behavioral* VHDL (more common) *behavioral* VHDL (more common)
Sequential Circuits for FSM Implementation

Remarks:

- The process block is required to synthesize a behavioral VHDL description of a synchronous sequential circuit

- You can alternatively use structural VHDL (FF components) and purely concurrent statements
Recall: VHDL Code for a D-type Flip-Flop

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY flipflop IS
  PORT (D, Clock : IN STD_LOGIC;
        Q : OUT STD_LOGIC);
END flipflop;

ARCHITECTURE behavioral OF flipflop IS
BEGIN
  PROCESS (Clock)
  BEGIN
    IF Clock'EVENT AND Clock = '1' THEN
      Q <= D;
    END IF;
  END PROCESS;
END behavioral;
There are many ways to describe a FSM in VHDL.

We will describe 2 ways.

Example: Moore-type sequence detector.
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY simple IS
PORT (Clock, Resetn, w
    : IN    STD_LOGIC ;
    Z: OUT   STD_LOGIC ) ;
END simple ;
ARCHITECTURE Behavior OF simple IS
TYPE      State_type IS (A, B, C) ;
SIGNAL y : State_type ;
BEGIN
PROCCESS ( 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 ;
        END CASE ;
    END IF ;
END PROCESS ;
END Behavior ;
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 ;
Remarks on first possible VHDL code for Moore-type sequence detector:

- The VHDL synthesizer will automatically creates the states i.e. it chooses the number of flip-flops to use and the assignment of state variable values to the state symbolic labels A, B, and C.

- CAD assumes the first state listed is the reset state and assigns it so that all FF outputs are equal to 0.
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;
END CASE;
END PROCESS;
END;

• In this code there are two process within the structure
• One for describing the state process as a combination logic
• The other process introduces flip-flops into the circuit
WHEN C =>
  IF w = '0' THEN
    y_next <= A ;
  ELSE
    y_next <= C ;
  END IF ;
END CASE ;
END PROCESS ;

PROCESS ( Resetn, Clock )
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 = C ELSE '0' ;

END Behavoir;
The previous code run in Quartus II software fine.
Some CAD tools may not recognize the type statement.

In the following code, we manually assign the states.
The idea is to use Constant to represent the states.
This code is compatible with any CAD tools

```vhdl
TYPE State_type IS (A, B, C);
SIGNAL y_present, y_next : State_type;
SIGNAL y_present, y_next : STD_LOGIC);
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”;
```
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 ;
The difference with Moore-type FSM is on the output part of the system.

In Mealy-type FSM, the output is inside of the process for which the sensitivity list depends on the input $w$.

This means the output changes whenever there is a change in the input of the system.

This is not the case in Moore-type FSM. In Moore-type FSM, the changes in the input does not affect the output until the positive edge of the clock is seen.
Simulation Results: Sequence Detector

Moore-type

Mealy-type