ShanghaiTech University

EE 115B: Digital Circuits

Fall 2022

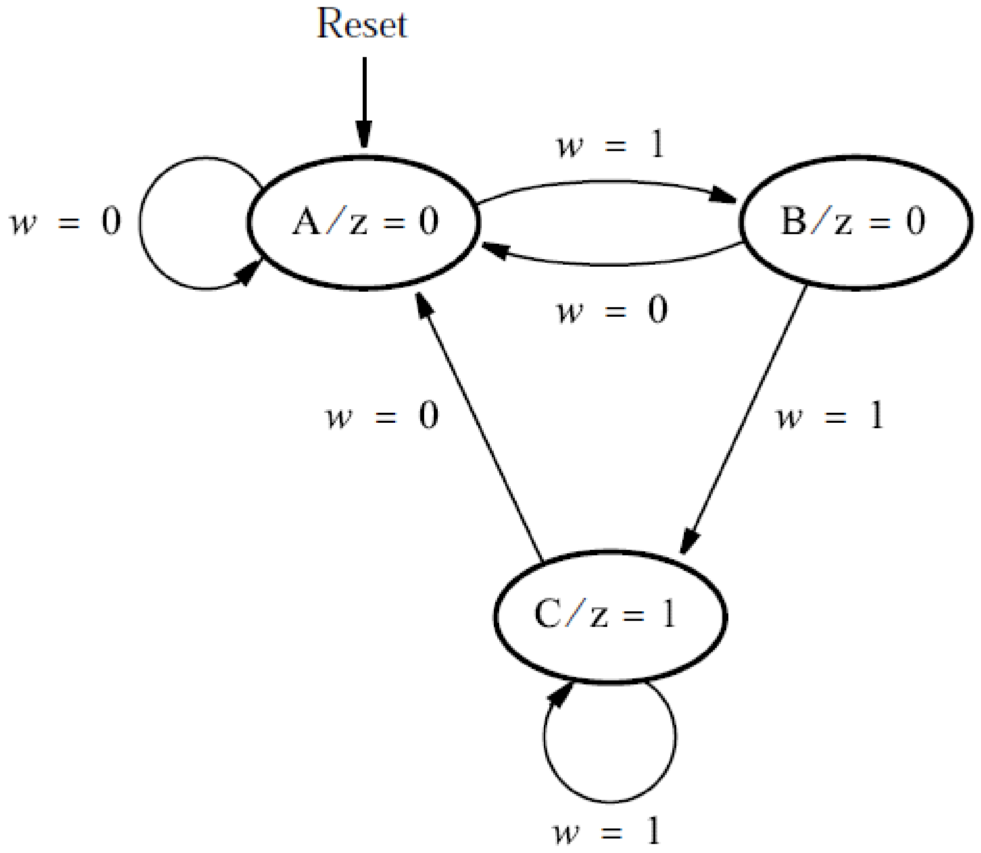
Lecture 21

Hengzhao Yang

December 20, 2022

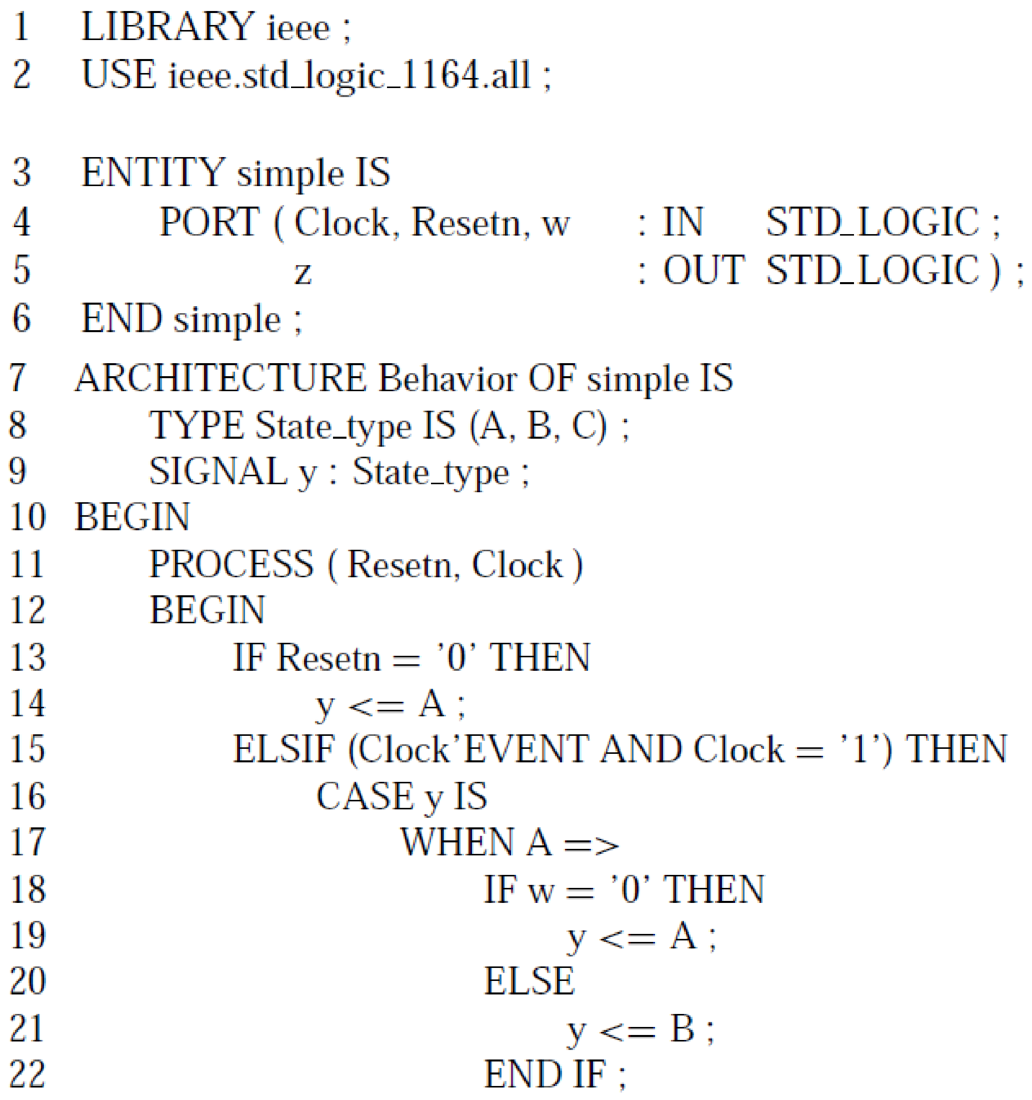
VHDL code for Moore FSMs

- Design input: state diagram
- Example: “11” sequence detector



Moore type: code 1

- TYPE keyword
  - User-defined signal type
  - Name: State_type
  - Values: A, B, C
- Signal y
  - Type: State_type
  - Describes FSM state
  - Takes values (A, B, C)
- Initial state A by Resetn
- CASE statement
  - State transitions



```
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
      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 ;
```

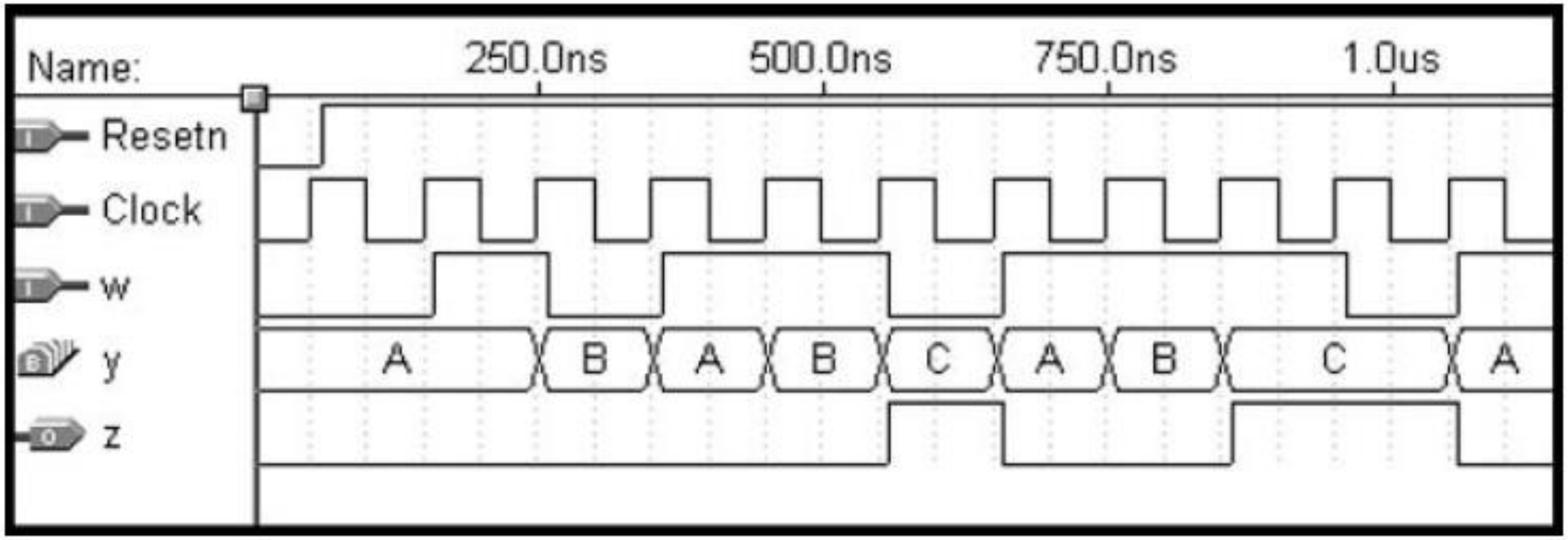
Moore type: code 1 (continued)

- WHEN statement
  - Output expression
- Summary
  - One signal (y) for state
  - One PROCESS block

```
                  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 ;
```

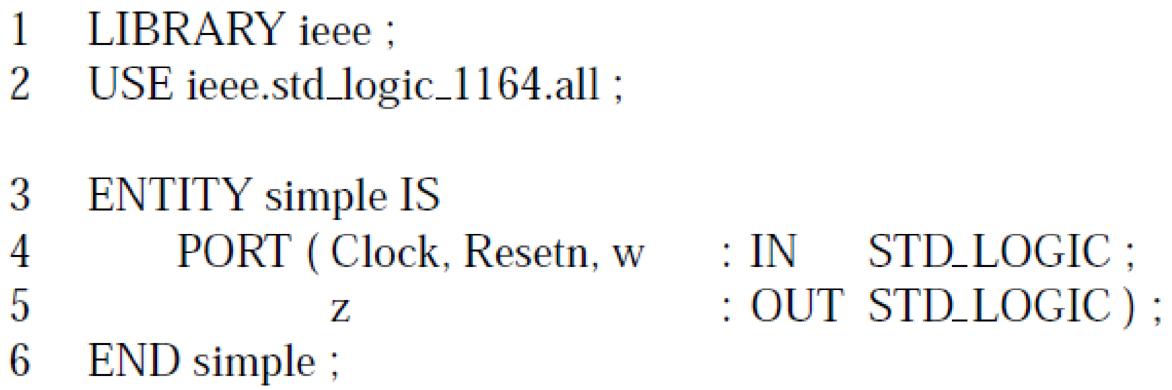
Moore type: code 1, simulation results

- Signal y: state



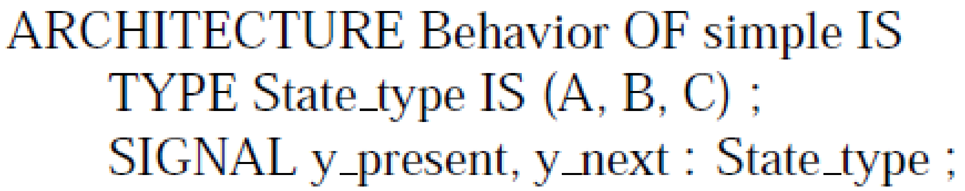
Moore type: code 2

- Same ENTITY



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

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

- Two signals for state
  - Present state: y_present
  - Next state: y_next
  - Same type: State_type
  - Same values: A, B, C

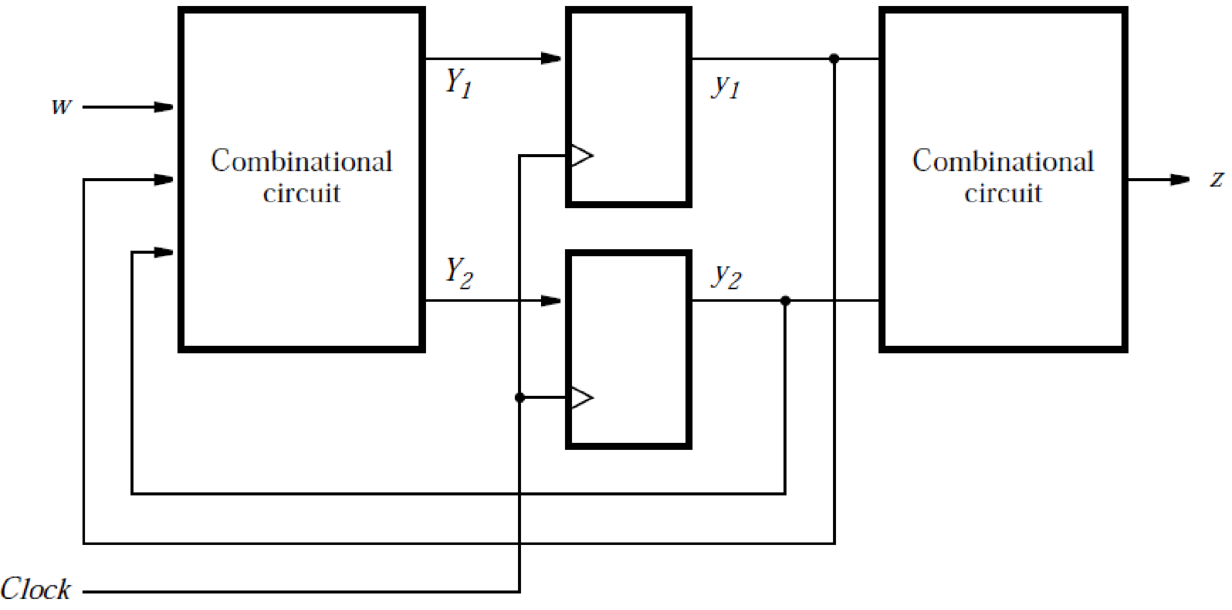


```
ARCHITECTURE Behavior OF simple IS
    TYPE State_type IS (A, B, C) ;
    SIGNAL y_present, y_next : State_type ;
```

- Two PROCESS blocks

Moore type: code 2 (continued)

- PROCESS 1
  - Next-state expressions
  - Left combinational circuit



```
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 ;
```

Moore type: code 2 (continued)

- PROCESS 2
  - State update
  - Sequential 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 ;
```

- WHEN statement
  - Output expression
  - Right combinational circuit

Moore type: code 3

- Code 1 and 2
  - Automatic state assignment
  - Type of state signals (y, y_present, y_next): user-defined (State_type)
- Code 3
  - Manual (explicit) state assignment
  - Type of state signals: standard (std_logic_vector)

Moore type: code 3 (continued)

- State assignment
  - A: 00, B: 01, C: 11

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

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

- State signals
  - y_present, y_next: 2-bit vector
  - States and values: A=“00”, B=“01”, C=“11”
  - State is defined as CONSTANT
  - Operator for CONSTANT value assignment: “:=”

```
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” ;
```

Moore type: code 3 (continued)

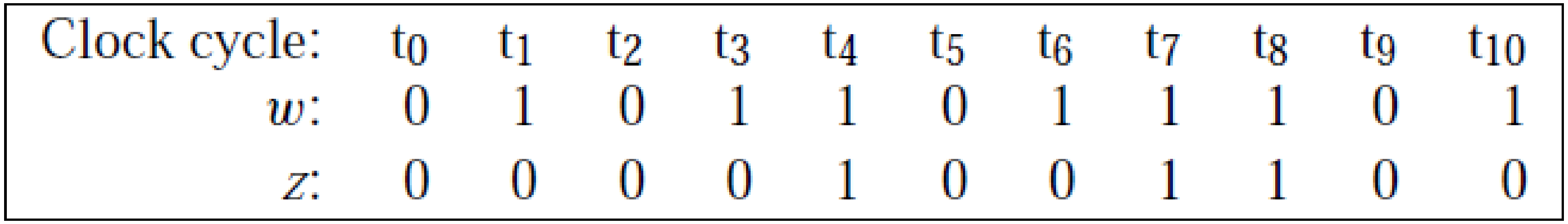
- PROCESS 1 (right)
  - WHEN OTHERS
    - Covers unused values
- PROCESS 2 (left)
- Output (left)

```
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 ;
            WHEN OTHERS =>
                y_next <= A ;
        END CASE ;
    END PROCESS ;
```

```
    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 ;
```

Mealy FSM

- Sequence detector
  - Circuit has one input (w) and one output (z)
  - All changes occur at positive clock edges
  - Operation: z=1 in the same clock cycle when the second occurrence of w=1 is detected; z=0, otherwise



Clock cycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	1	0	0	1	1	0	0

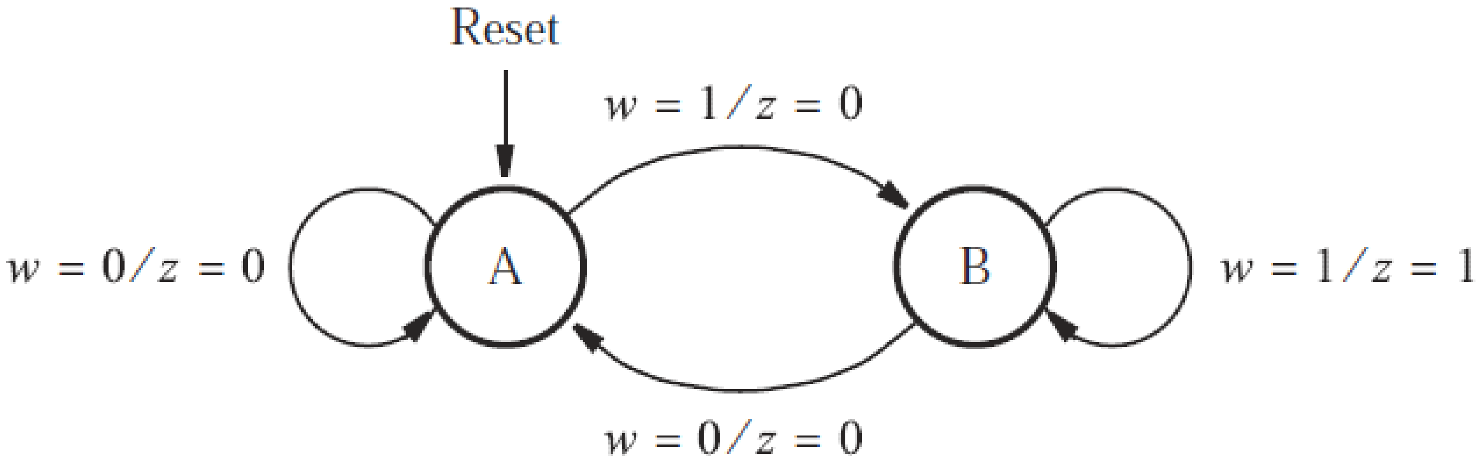
Comparison: Moore FSM

- Design specifications (problem statement)
  - Circuit has one input (w) and one output (z)
  - All changes occur on positive clock edges
  - Operation: z=1 if w=1 during two immediately preceding clock cycles; z=0, otherwise

Clock cycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0

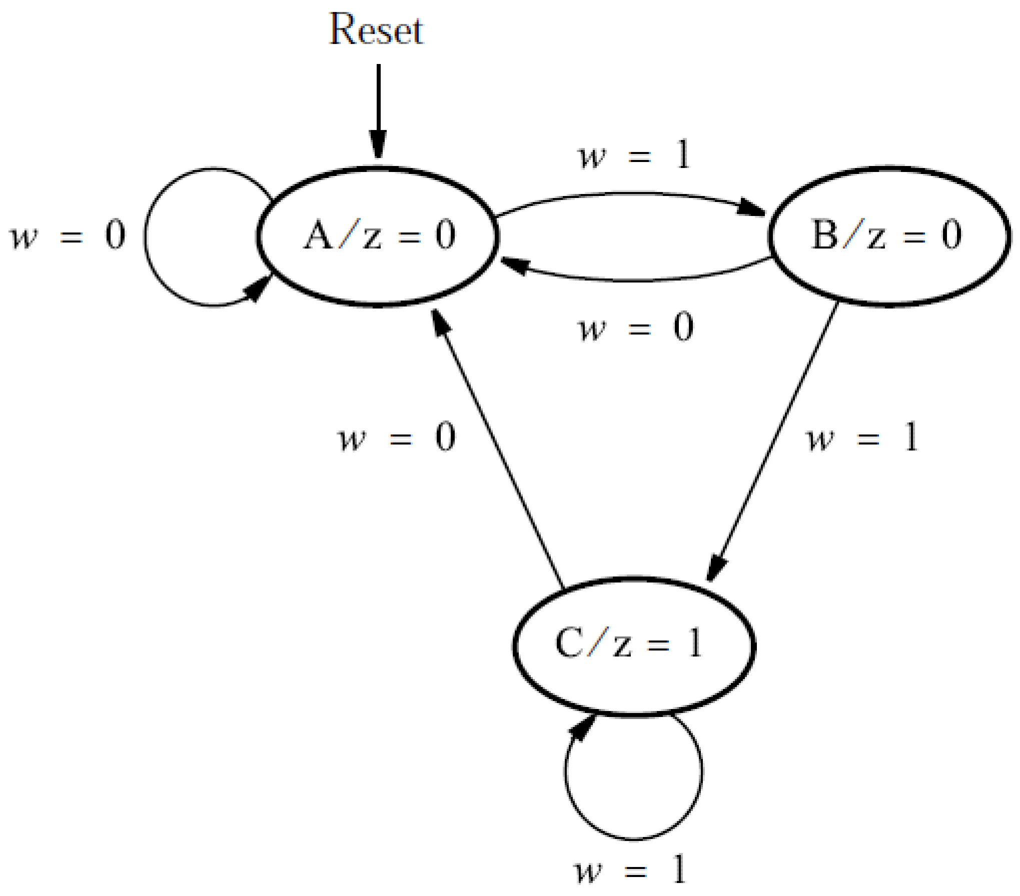
State diagram

- States
  - State A (starting state): when a reset signal is applied or w=0, circuit enters this state
  - State B: w=1 for one clock cycle



Comparison: Moore FSM

- State diagram
  - Nodes: states
  - Directed arcs: state transitions



State table and state-assigned table

- Two states: A and B

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	A	B	0	1

- One state variable: y
  - A: 0, B: 1

	Present state	Next state		Output	
		$w = 0$	$w = 1$	$w = 0$	$w = 1$
	y	Y	Y	z	z
A	0	0	1	0	0
B	1	0	1	0	1

Comparison: Moore FSM

- Convert state diagram to state table
  - First state is starting state (state A)
  - Present state, next state, and state transitions
  - Output is specified with respect to present state

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	A	B	0
B	A	C	0
C	A	C	1

Output and next-state expressions

- Expressions

$$Y = D = w$$

$$z = wy$$

	Present state	Next state		Output	
		$w = 0$	$w = 1$	$w = 0$	$w = 1$
	y	Y	Y	z	z
A	0	0	1	0	0
B	1	0	1	0	1

- Truth table: output and next-state

y	w	z
0	0	0
0	1	0
1	0	0
1	1	1

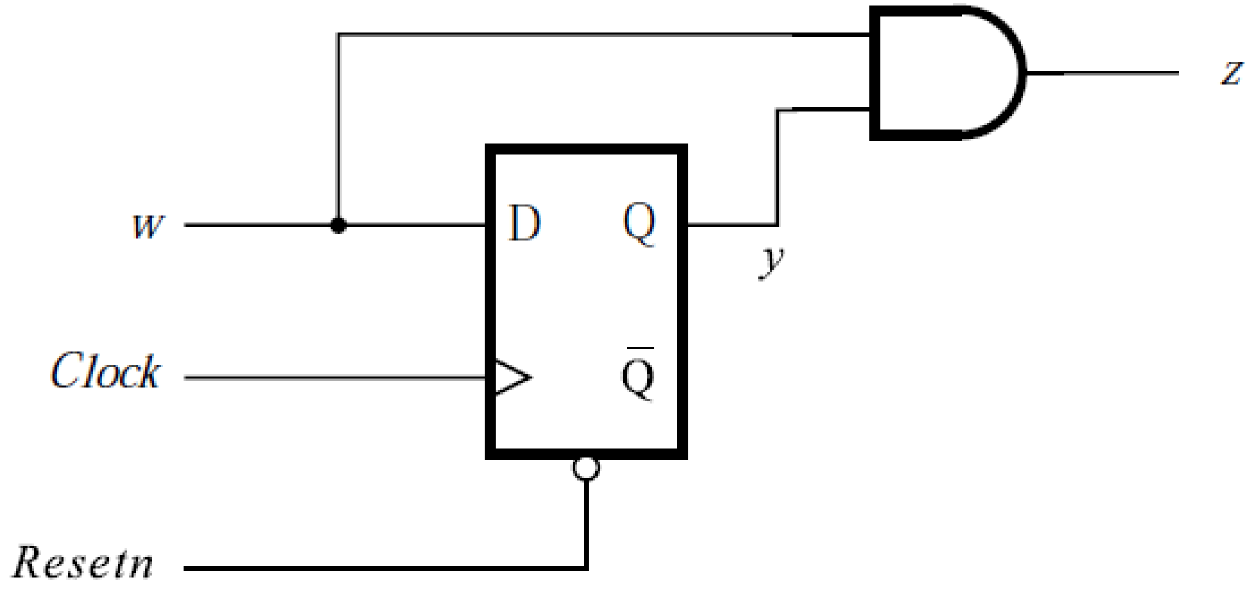
y	w	Y
0	0	0
0	1	1
1	0	0
1	1	1

Circuit

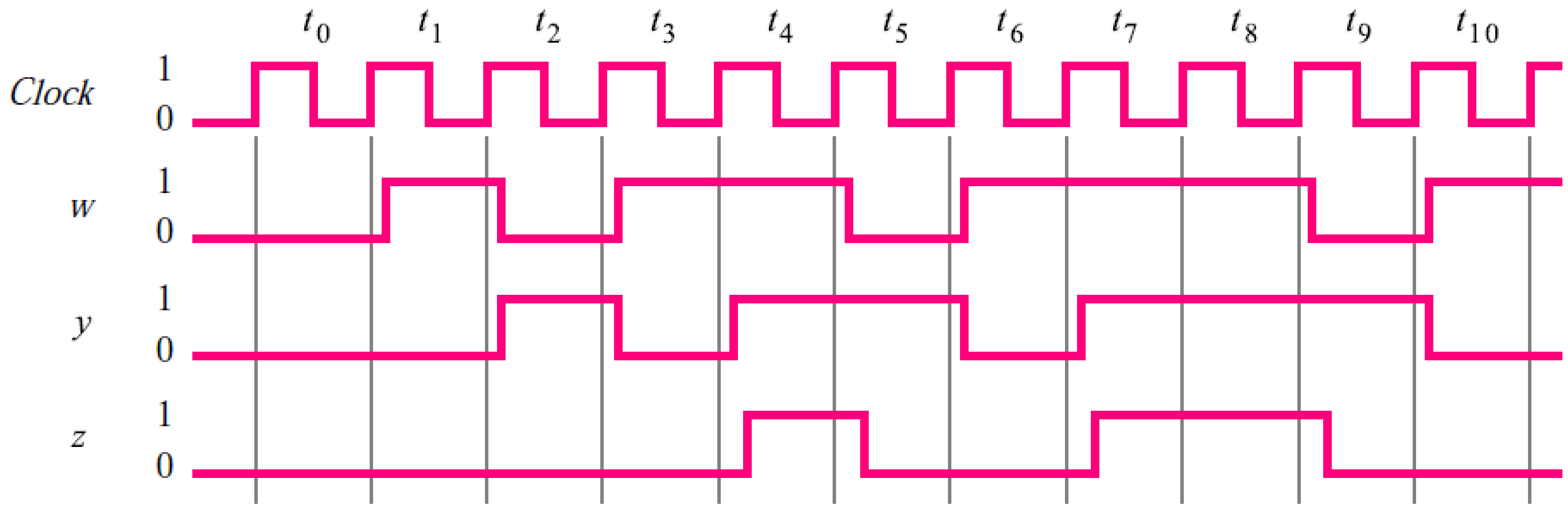
- Expressions

$$Y = D = w$$

$$z = wy$$

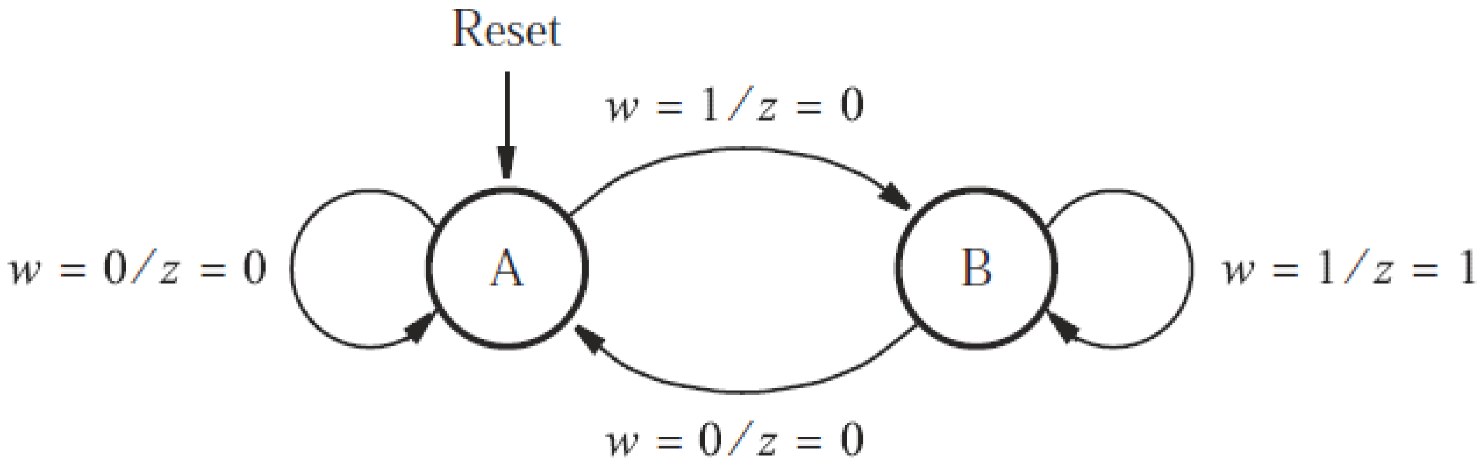


Timing diagram



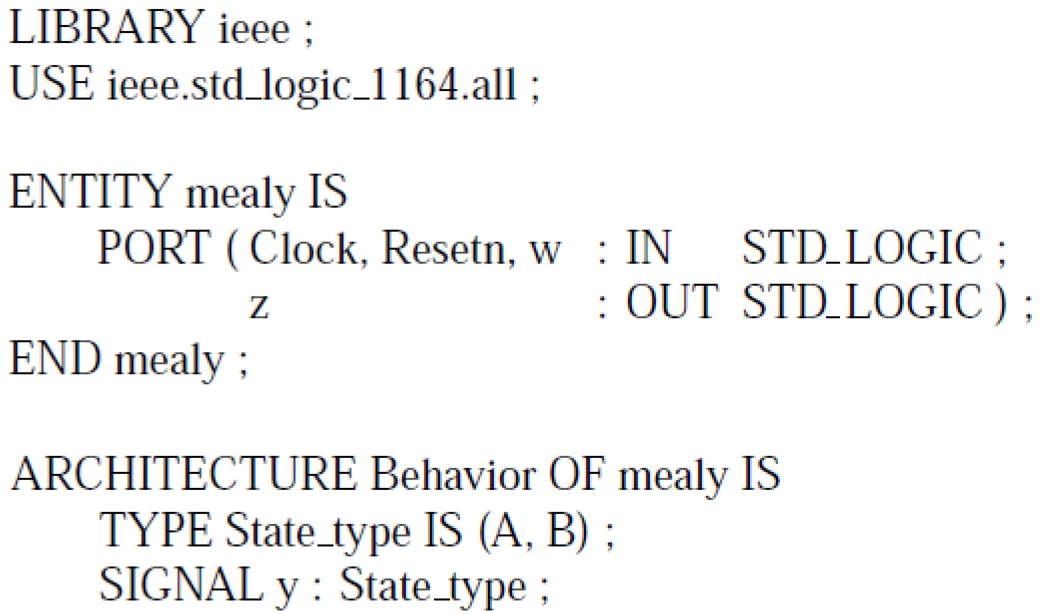
VHDL code for Mealy FSMs

- Design input: state diagram
- Example: “11” sequence detector



Mealy type code

- Similar to Moore type code 1
- Two states: A, B
- User-defined state signal type: State_type

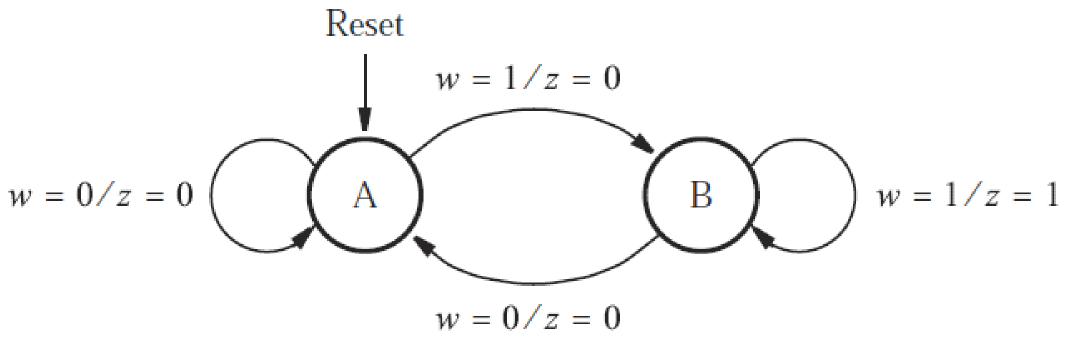


```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

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 ;
```

Mealy type code (continued)

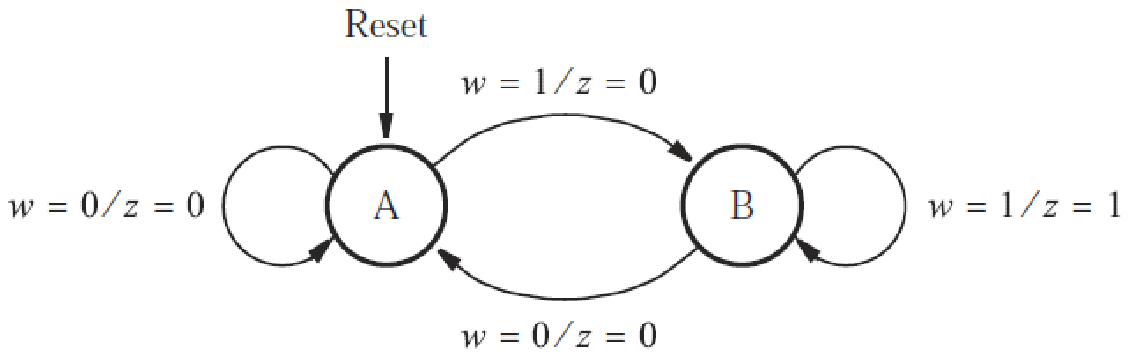
- PROCESS 1
  - State transitions

```
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 ;
```



Mealy type code (continued)

- PROCESS 2
  - Output depends on y and w



```
    PROCESS ( y, w )
    BEGIN
        CASE y IS
            WHEN A =>
                z <= '0' ;
            WHEN B =>
                z <= w ;
        END CASE ;
    END PROCESS ;
END Behavior ;
```

Review

- Chapter 1: Introduction
  - Basic concepts

Chapter 2: Numbers and codes

- Numbers
  - Decimal
  - Binary
  - Hexadecimal
  - Octal
- Codes
  - BCD, Gray, and ASCII
  - Parity check

Chapter 3: Logic gates

- Circuit symbol, operation, truth table, and logic expression
  - NOT
  - AND
  - OR
  - NAND
  - NOR
  - XOR
  - XNOR

Chapter 4: Boolean algebra and logic simplification

- Basic Boolean algebra relationships
- DeMorgan’s theorem
- SOP and POS expressions: standard and minimum, explicit and concise
- Karnaugh map: minimum SOP and POS expressions, with and without don’t cares
- Quine-McCluskey method

Chapter 5: Combinational logic analysis

- Logic forms: AND-OR, AND-OR-Invert (AOI), NAND, and NOR
- Combinational logic design flow

Chapter 6: Combinational logic blocks

- Adder
- Encoder
- Decoder
- Comparator
- MUX
- DEMUX

VHDL

- Basic structures: entity and architecture
- Behavioral and structural coding
- Combinational and sequential logic

Chapter 7: Basic sequential circuits

- Blocks: circuit symbol, operation, and characteristic table
  - Latches
  - Flip-flops
- Modules
  - Registers
  - Counters

Chapter 8: Synchronous sequential circuits

- Design flow
  - Moore FSMs
  - Mealy FSMs