ShanghaiTech University

EE 115B: Digital Circuits

Fall 2022

Lecture 20

Hengzhao Yang
December 15, 2022

Sequence detector

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

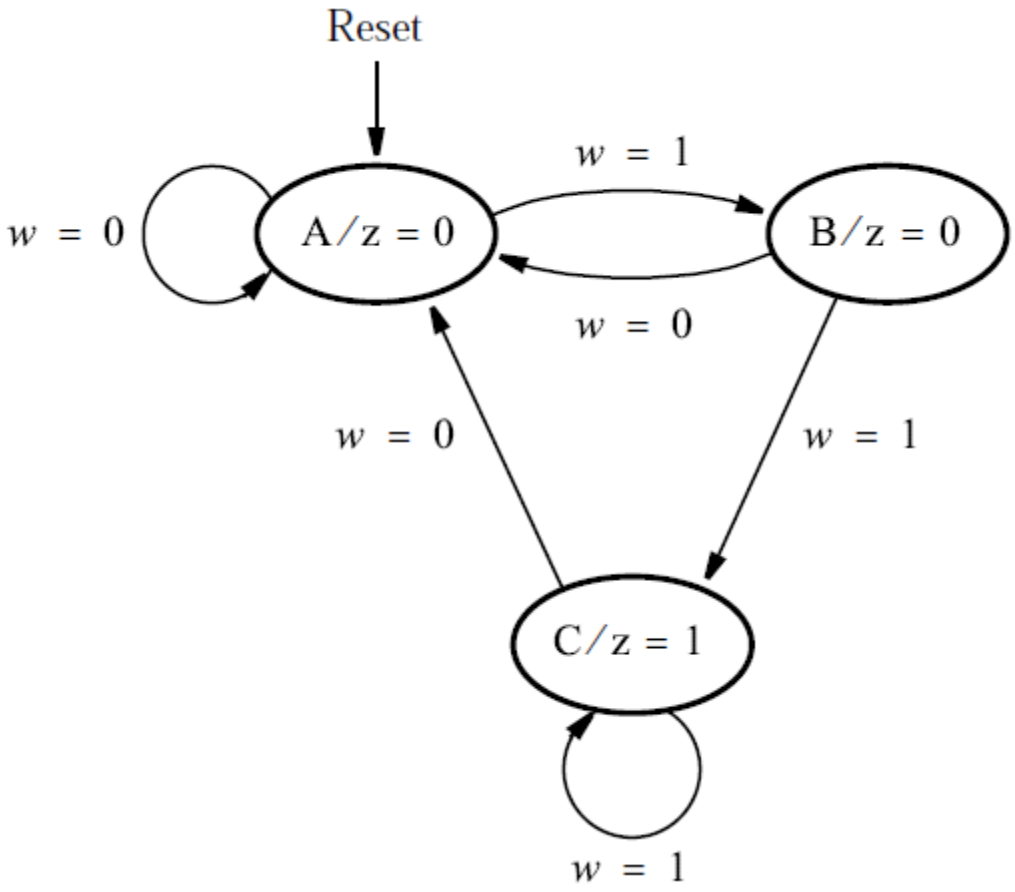
Clock cycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0

State diagram

- Identify states to describe circuit behavior
  - State A (starting state): when a reset signal is applied or w=0, circuit enters this state, z=0
  - State B: w=1 for one clock cycle, z=0
  - State C: w=1 for two clock cycles, z=1

State diagram

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



State table

- 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

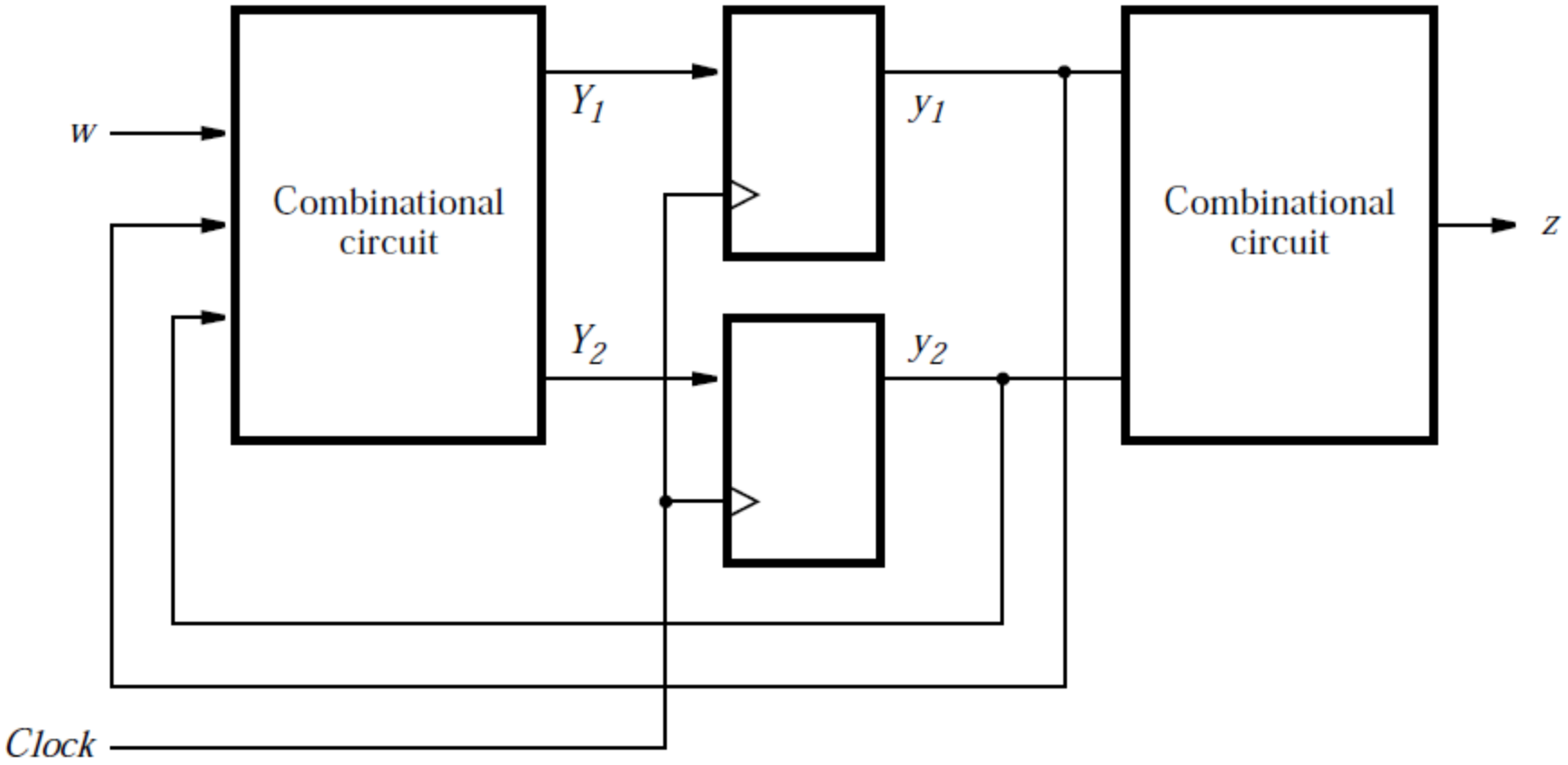
Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	C	1

State assignment

- Each state is represented by a combination of state variables
- Each state variable is implemented by a flip-flop
- Three states (A, B, C) need two state variables (y_1, y_2)
- Moore type: output only depends on state
- Present-state variables: y_1, y_2
- Next-state variables: Y_1, Y_2

State assignment

- Circuit block diagram



State-assigned table

- Assign binary values to states
  - A: 00, B: 01, C: 10
  - "11" is not used

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

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	dd	dd	d

Choice of flip-flops

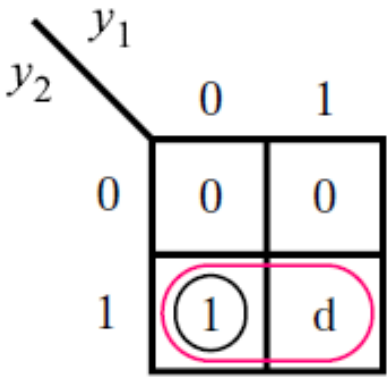
- D flip-flops: transfer Y_1 and Y_2 to y_1 and y_2

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	dd	dd	d

Output expression

- Output expression: z
  - Ignoring don't cares: $z = \bar{y}_1 y_2$
  - Using don't cares: $z = y_2$

y2	y1	z
0	0	0
0	1	0
1	0	1
1	1	d

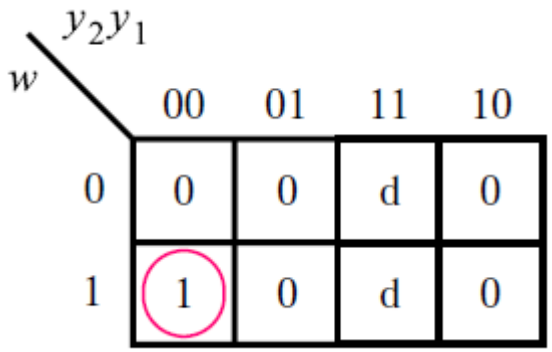


	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	dd	dd	d

Next-state expressions

w	y2	y1	Y_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	d
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	d

- Next-state expression: Y_1
  - Ignoring don't cares: $Y_1 = w\bar{y}_1\bar{y}_2$
  - Using don't cares: $Y_1 = w\bar{y}_1\bar{y}_2$

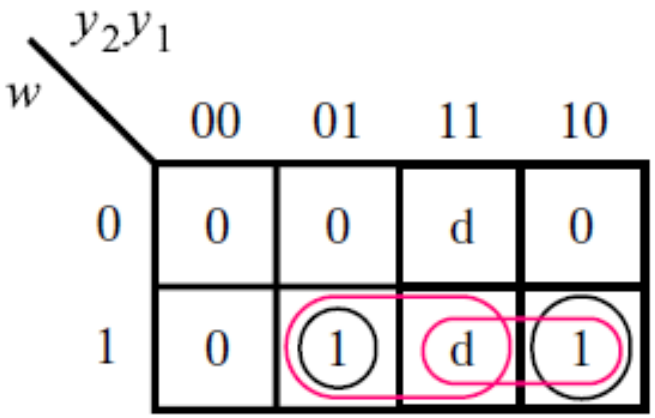


	Present state	Next state $w=0$	Next state $w=1$	Output
	y_2y_1	Y_2Y_1	Y_2Y_1	z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	dd	dd	d

Next-state expressions

- Next-state expression: Y_2
  - Ignoring don't cares: $Y_2 = wy_1\bar{y}_2 + w\bar{y}_1y_2$
  - Using don't cares: $Y_2 = wy_1 + wy_2$
    $= w(y_1 + y_2)$

w	y2	y1	Y_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	d
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	d



	Present state	Next state $w=0$	Next state $w=1$	Output z
	y_2y_1	Y_2Y_1	Y_2Y_1	
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Circuit (using don't cares)

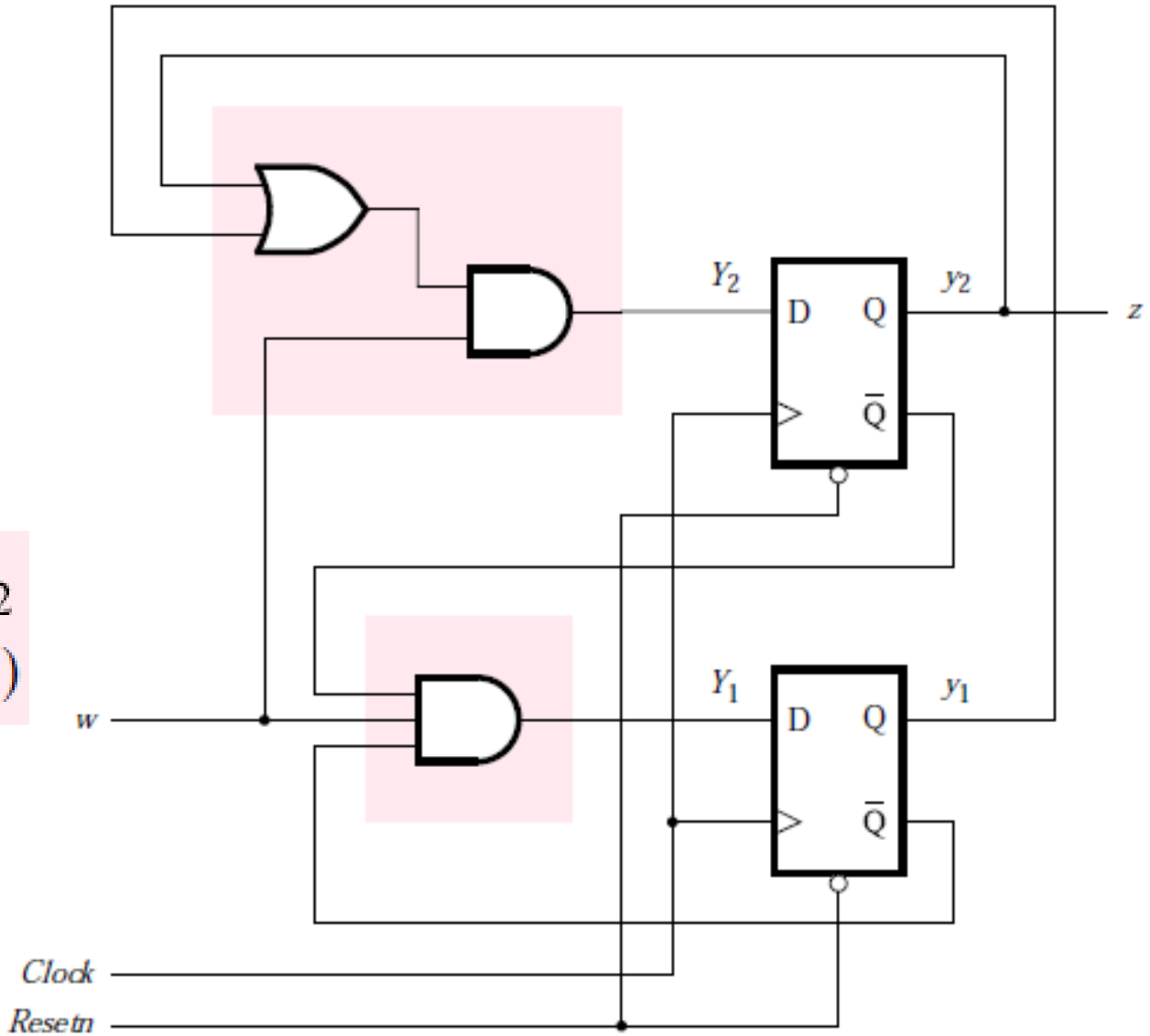
- Output

$$z = y_2$$

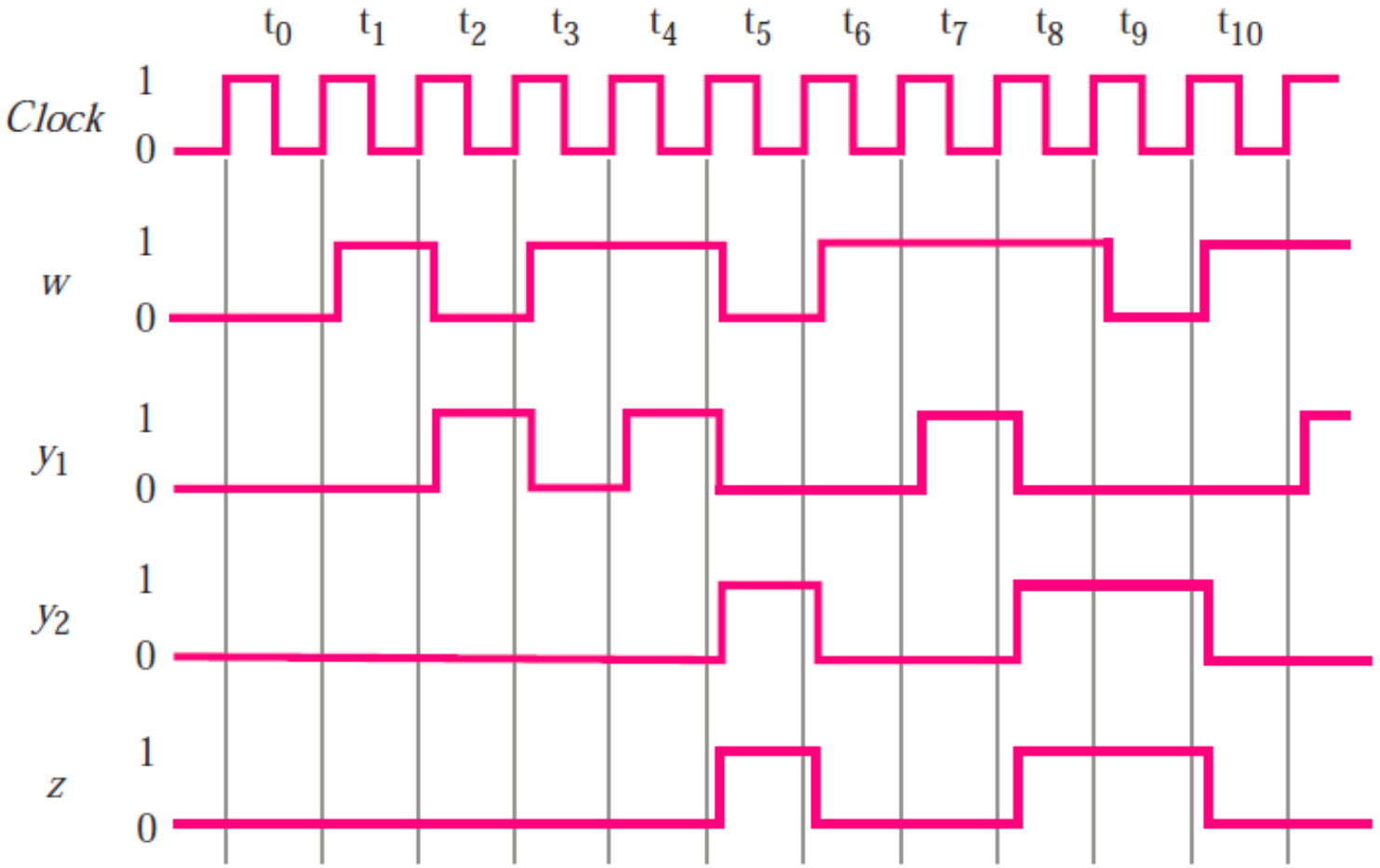
- Next-state

$$Y_1 = w\bar{y}_1\bar{y}_2$$

$$\begin{aligned} Y_2 &= wy_1 + wy_2 \\ &= w(y_1 + y_2) \end{aligned}$$



Timing diagram



Summary

- Design steps
  - Understand design specifications
  - Identify states to describe circuit behavior and draw state diagram for states and transitions
  - Convert state diagram to state table
  - Assign state variables
  - Choose flip-flops and determine output and next-state expressions
  - Implement circuit

State-assignment issue

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

- Original state assignment
  - A: 00, B: 01, C: 10
  - "11" is not used

	Present state $y_2 y_1$	Next state $w=0$ $Y_2 Y_1$	Next state $w=1$ $Y_2 Y_1$	Output z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

- Alternative state assignment
  - A: 00, B: 01, C: 11
  - "10" is not used

	Present state $y_2 y_1$	Next state $w=0$ $Y_2 Y_1$	Next state $w=1$ $Y_2 Y_1$	Output z
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

Output and next-state expressions

- Expressions

$$Y_1 = D_1 = w$$

$$Y_2 = D_2 = wy_1$$

$$z = y_2$$

	Present state	Next state		Output
		$w = 0$	$w = 1$	z
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

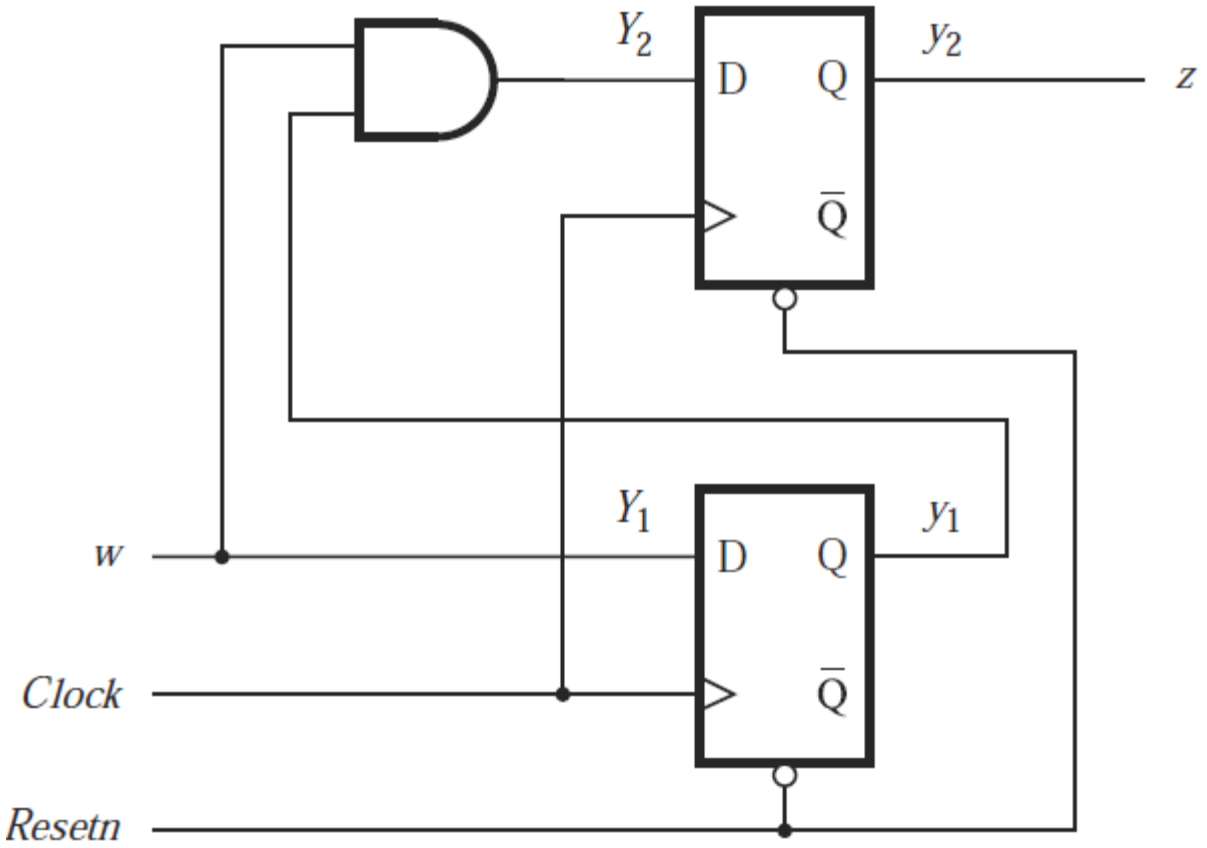
Circuit (with a lower cost)

- Expressions

$$Y_1 = D_1 = w$$

$$Y_2 = D_2 = wy_1$$

$$z = y_2$$



One-hot encoding

- One-hot encoding
  - Another approach to assign states
  - Number of states equals number of state variables
  - For each state, only ONE state variable is “hot”: its value is “1”. The values of all the other state variables are “0”.

Example

- State table
  - Three states: A, B, C

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

- One-hot state assignment
  - Three state variables: y_3, y_2, y_1
  - A: 001, B: 010, C: 100
  - Other combinations of state variables are don't cares

	Present state	Next state		Output
		$w = 0$	$w = 1$	z
	$y_3 y_2 y_1$	$Y_3 Y_2 Y_1$	$Y_3 Y_2 Y_1$	
A	0 0 1	0 0 1	0 1 0	0
B	0 1 0	0 0 1	1 0 0	0
C	1 0 0	0 0 1	1 0 0	1

Example

- Output and next-state expressions

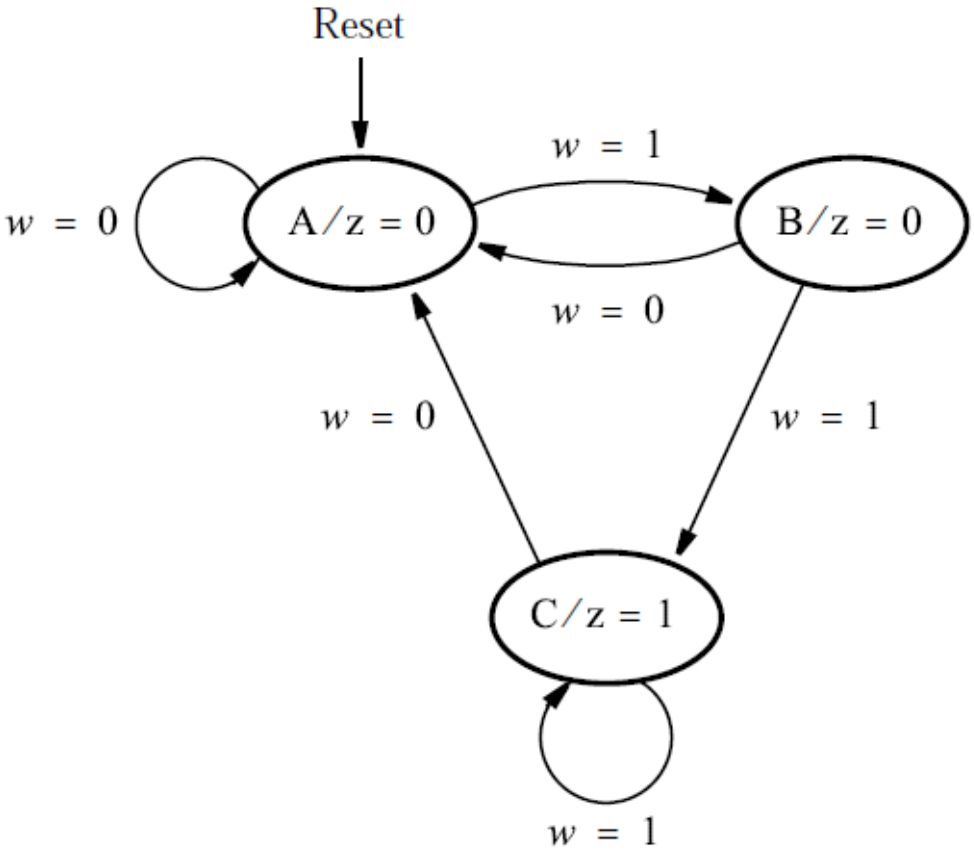
$$Y_1 = \overline{w}$$
$$Y_2 = wy_1$$
$$Y_3 = w\overline{y}_1$$
$$z = y_3$$

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_3 y_2 y_1$	$Y_3 Y_2 Y_1$	$Y_3 Y_2 Y_1$	z
A	0 0 1	0 0 1	0 1 0	0
B	0 1 0	0 0 1	1 0 0	0
C	1 0 0	0 0 1	1 0 0	1

- Circuit is not simpler for this example
- One-hot encoding is useful in many cases

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

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

3   ENTITY simple IS
4       PORT ( Clock, Resetn, w     : IN    STD_LOGIC ;
5              z                    : OUT   STD_LOGIC ) ;
6   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 ;
```

Moore type: code 1 (continued)

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

```
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 ;
```