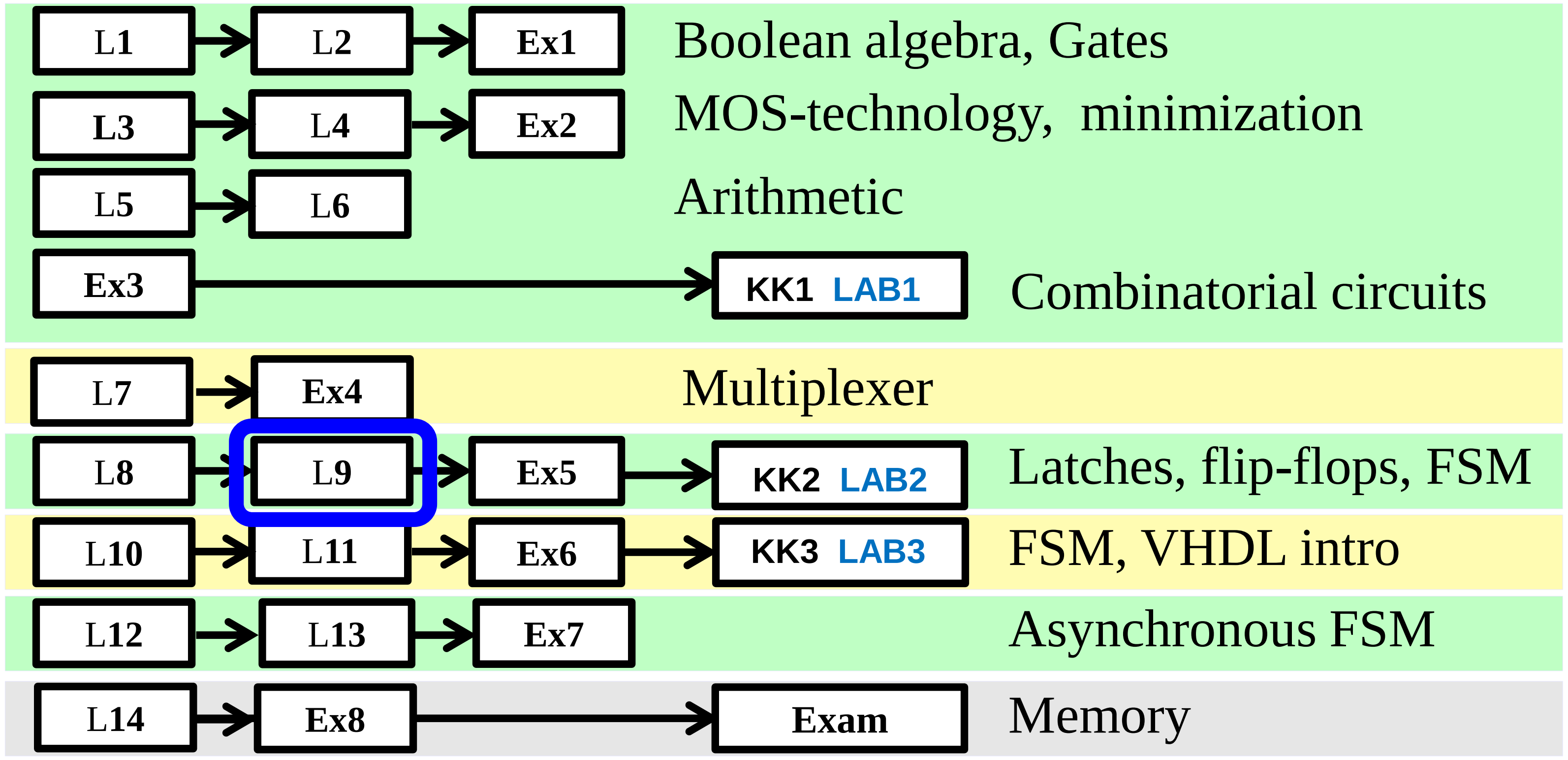
KTH Informations- och kommunikationsteknik

IE1204 Digital Design

L9: State Machines

Masoumeh (Azin) Ebrahimi
KTH/ICT
mebr@kth.se

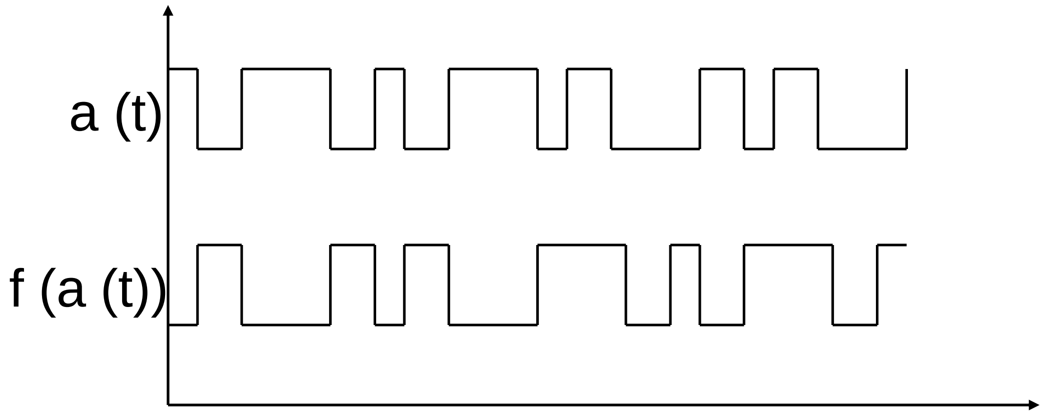
IE1204 Digital Design



This lecture

- BV pp. 485-507

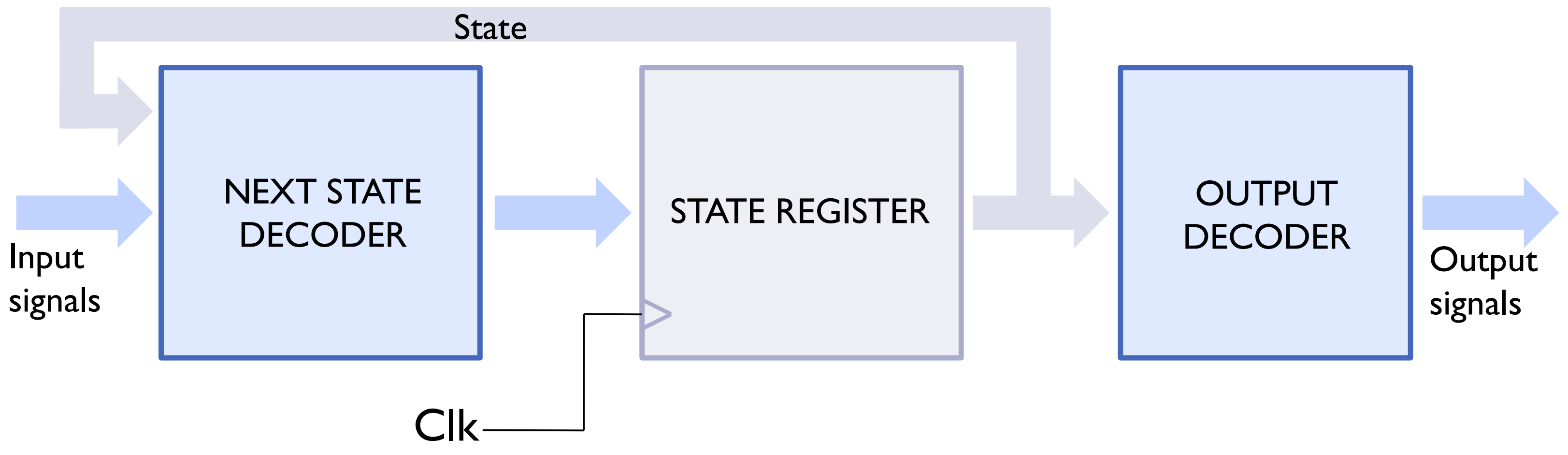
Sequential System



A sequential system has a built-in memory - the output depends therefore BOTH on the current and previous value(s) of the input signal

Lecture 8 - Lecture 13

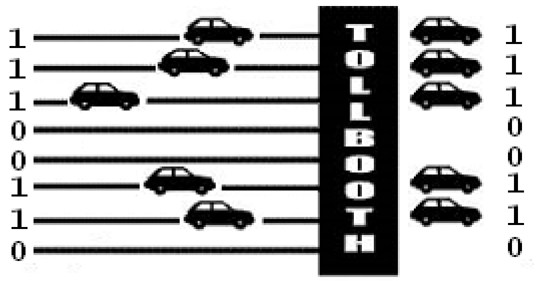
Moore-type machine

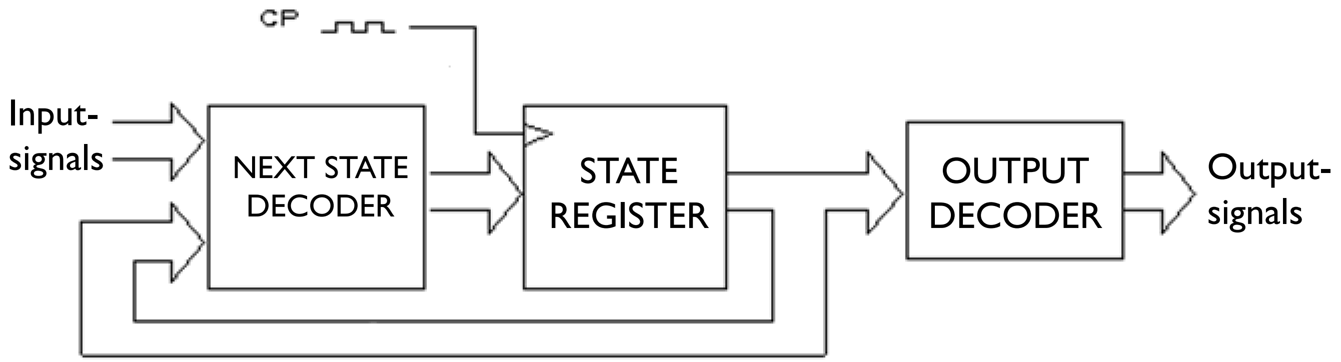


- In a Moore-type machine, output signals depend only on the current state

State register D-flip-flops

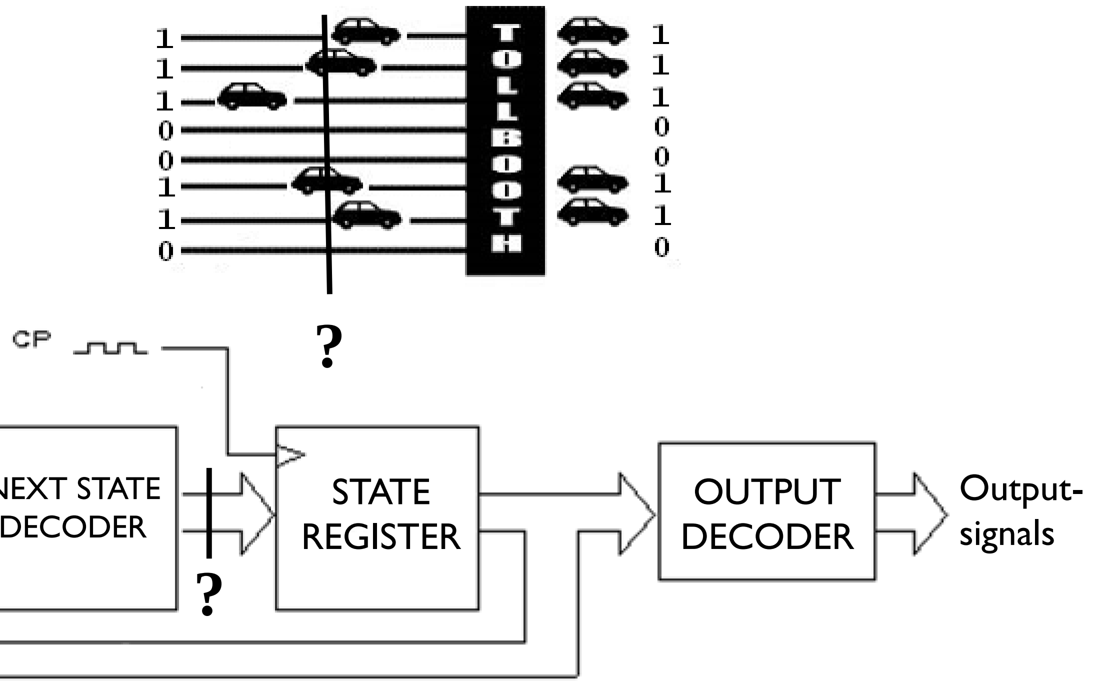
State register D flip-flops slows down the race between signals until the value is stable. (Compare with the tollbooth).





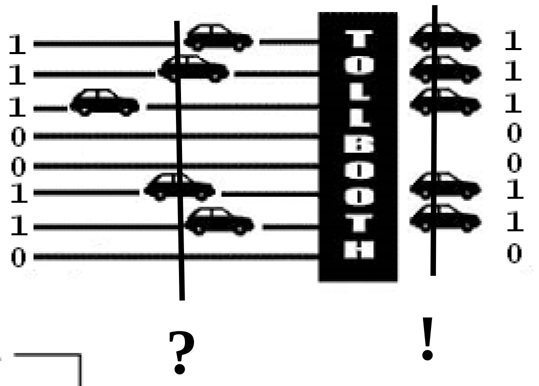
State register D-flip-flops

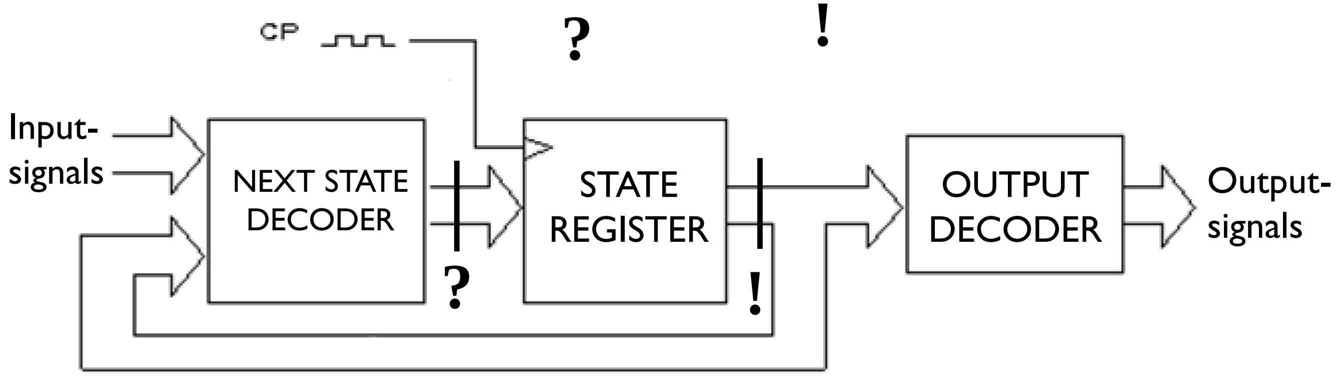
State register D flip-flops slows down the race between signals until the value is stable. (Compare with the tollbooth).



State register D-flip-flops

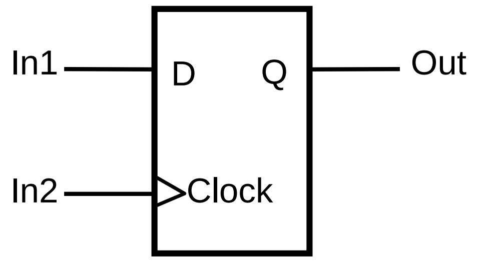
State register D flip-flops slows down the race between signals until the value is stable. (Compare with the tollbooth).

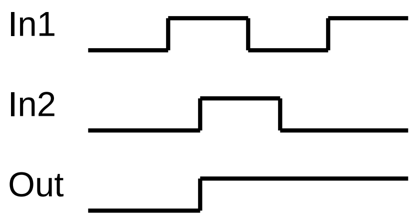




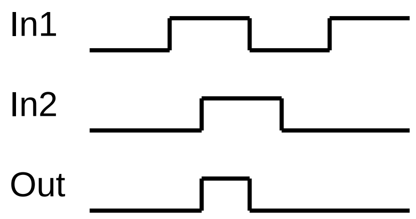
Quickie Question …

Which of the following timing diagram is valid for **a edge-triggered D flip-flop**?

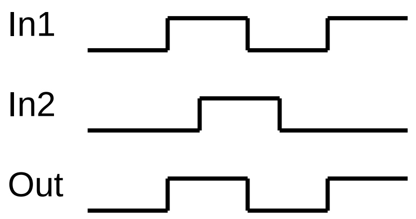




Alt: A



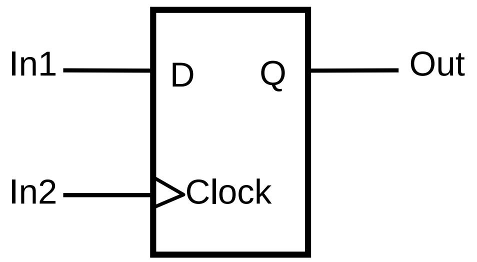
Alt: B

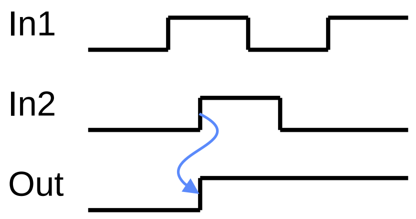


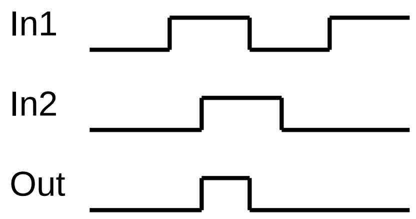
Alt: C

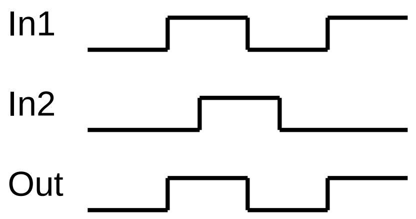
Quickie Question …

Which of the following timing diagram is valid for **a edge-triggered D flip-flop**?







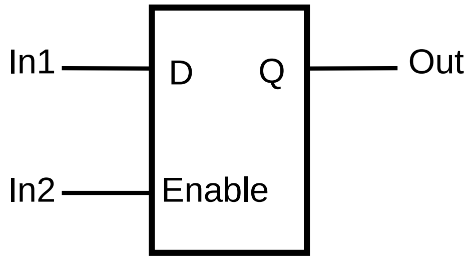


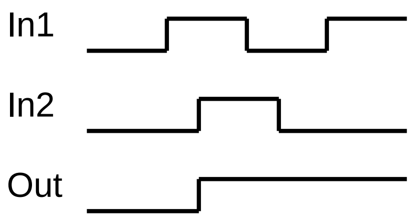
[✓ Alt: A] Alt: B Alt: C

D is copied to output at the edge, when Clock goes from 0 to 1

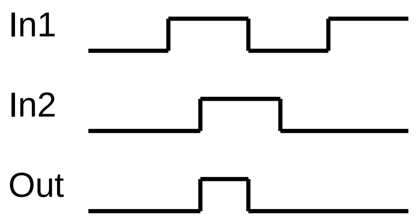
Quickie Question …

Which of the following timing diagram is valid for **a D Latch**?

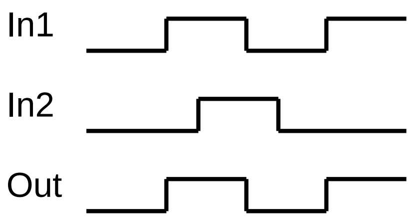




Alt: A



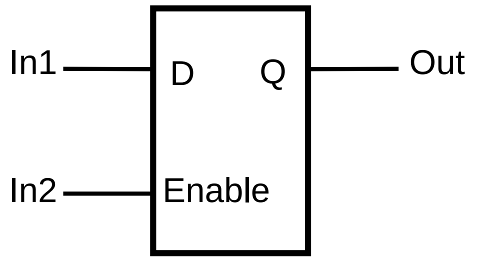
Alt: B

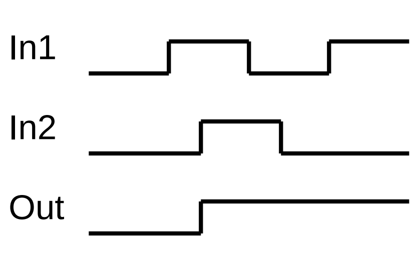


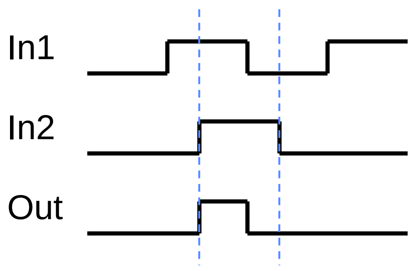
Alt: C

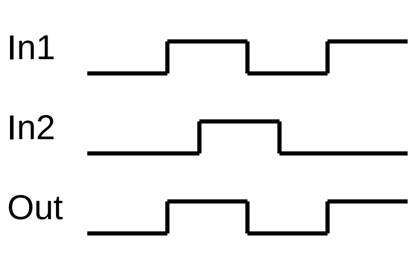
Quickie Question … *Latch*

Which of the following timing diagram is valid for **a D Latch**?









Alt: A [✓ Alt: B] Alt: C

D is connected to output when Enable is 1, and is locked when Enable is 0

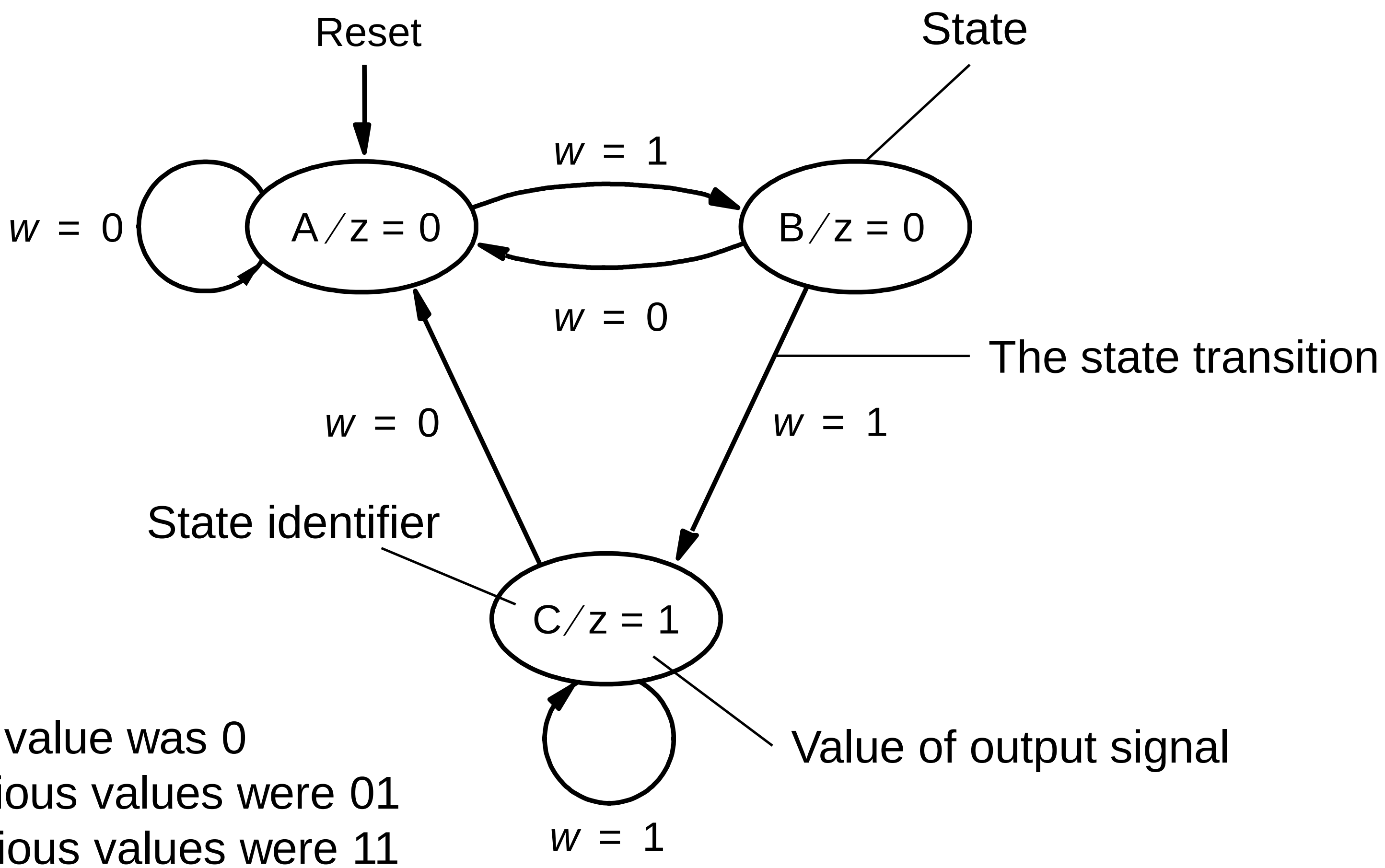
Design task: Bit sequence detector

- Specification
  - Circuit has an input w and an output z
  - If the input w is 1 in two consecutive clock pulses (or more), the output z should be set to 1 otherwise 0
  - Use a Moore machine with D flip-flops to implement the design

State Diagram

- There are many ways to draw a state diagram
- Here we follow the notation of the textbook (Brown/Vranesic)

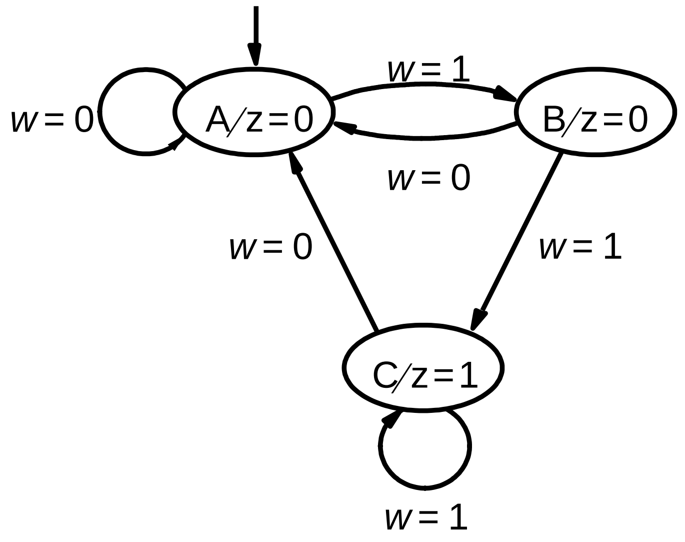
State Diagram of the bit sequence detector



A = previous value was 0
B = two previous values were 01
C = two previous values were 11

State Table

State Diagram

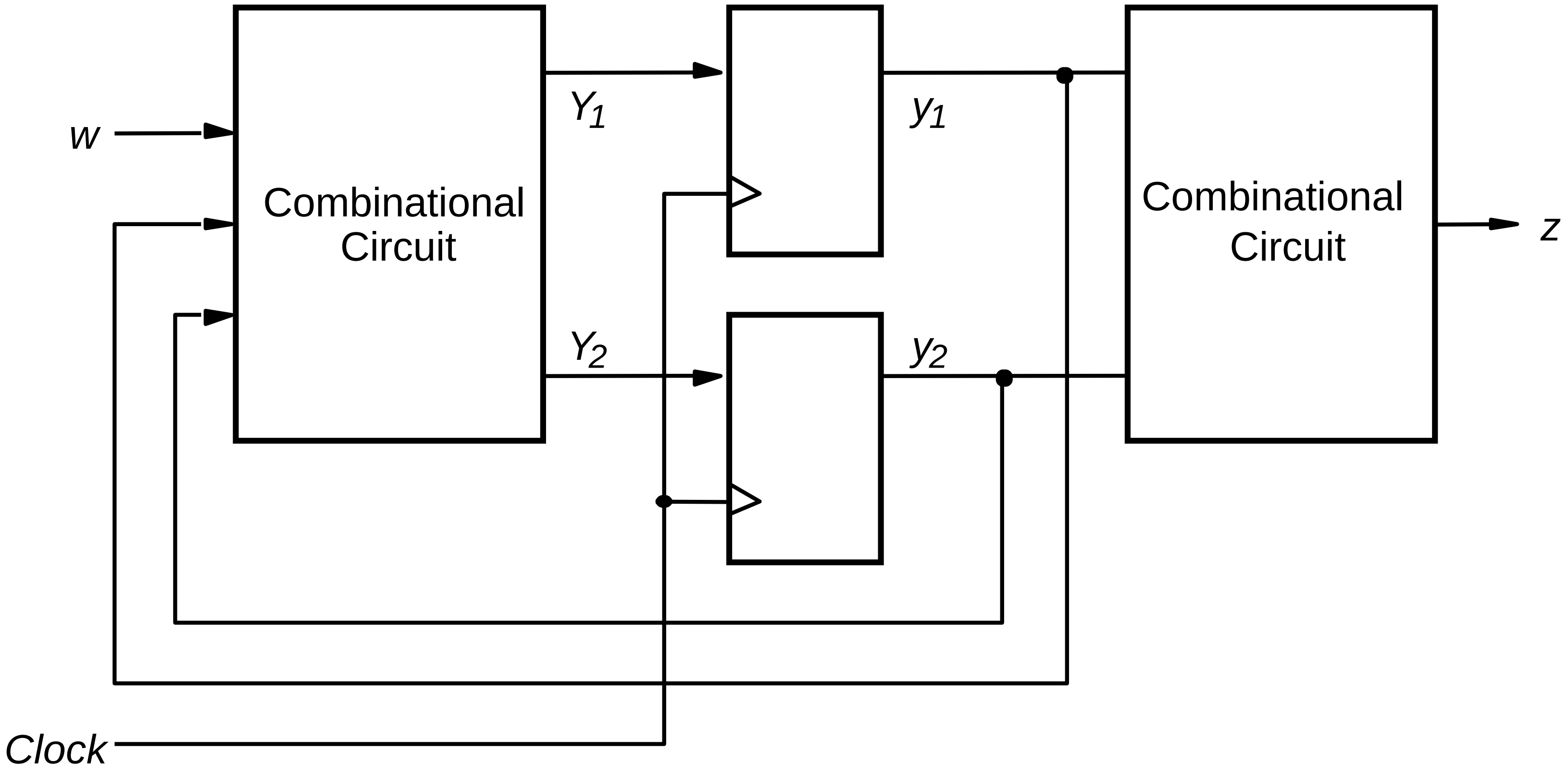


State Table

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

Three states - two bits are needed!

Schematic of the implementation of the bit sequence detector



Design decisions: Choice of flip-flops and state assignment

- The designer must determine which flip-flop to use
  - D, T, or JK flip-flop
- The designer must choose the assignment for each state

State Assignment for the bit sequence detector

- In this example, we use
  - D flip-flops
  - State assignmnets A = 00, B = 01, C = 10
  - The assignmnet 11 is not present. We make it ”don’t care”.

State Assignment bit sequence detector

State Table

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

A = 00
B = 01
C = 10

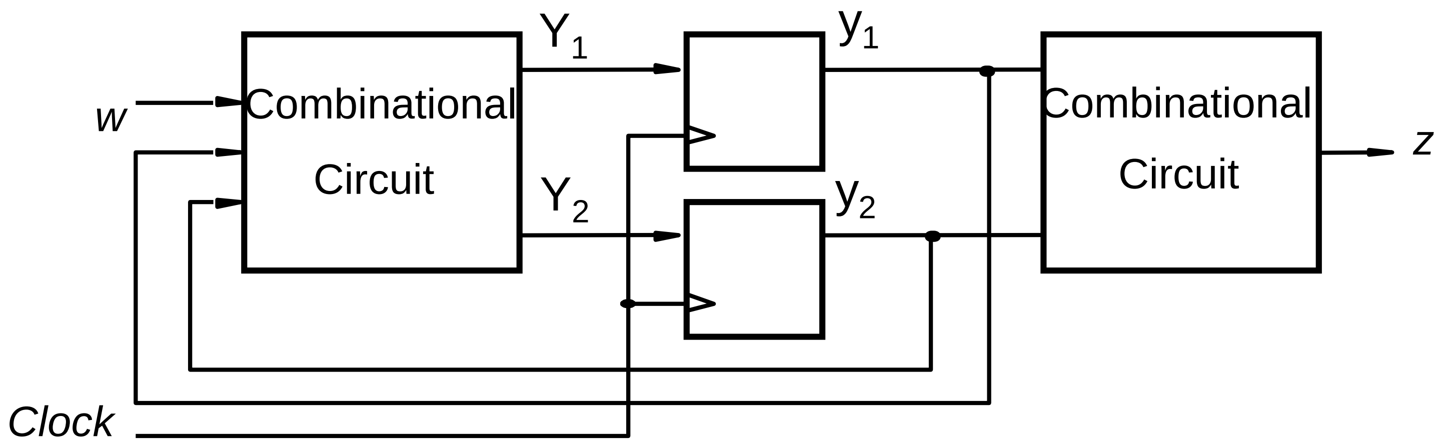
State-assigned Table

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

$$Y_2Y_1 = f(y_2y_1w) \quad z = f(y_2y_1)$$

Next-state expressions

- We must now derive next-state expressions
  - Inputs to both flip-flops, Y_1 and Y_2
- To get minimized logic functions, we use Karnaugh diagrams

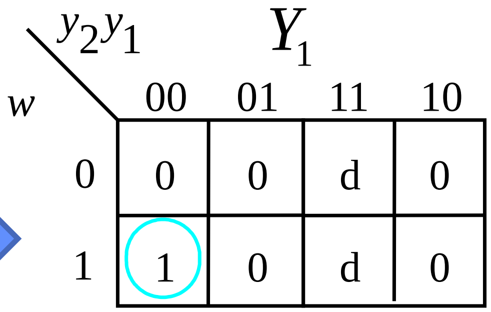


$$Y_2Y_1 = f(y_2y_1w) \quad Y_2 = f(y_2y_1w) \quad Y_1 = f(y_2y_1w) \quad z = f(y_2y_1)$$

From State-assigned Table to Karnaughmap

$$Y_2Y_1 = f(y_2y_1w) \quad Y_2 = f(y_2y_1w) \quad Y_1 = f(y_2y_1w)$$

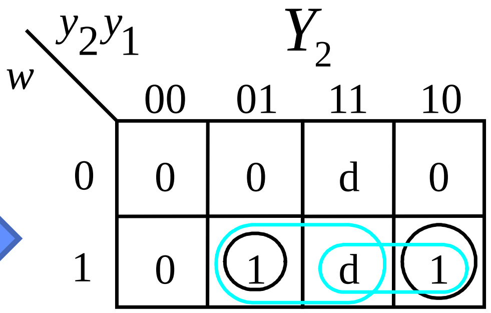
Present state	Next state		Output
	$w = 0$	$w = 1$	z
y_2y_1	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	10	1
11	*dd*	*dd*	*d*



w \ y_2y_1	00	01	11	10
0	0	0	d	0
1	1	0	d	0

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

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



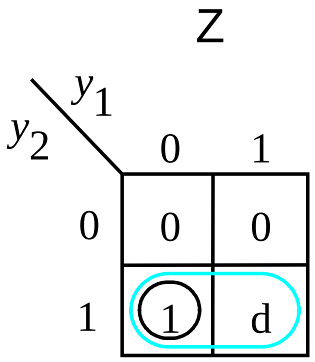
w \ y_2y_1	00	01	11	10
0	0	0	d	0
1	0	1	d	1

$$Y_2 = wy_1 + wy_2 =$$
$$= w(y_1 + y_2)$$

Output expression

- Now we derive the expression for output

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



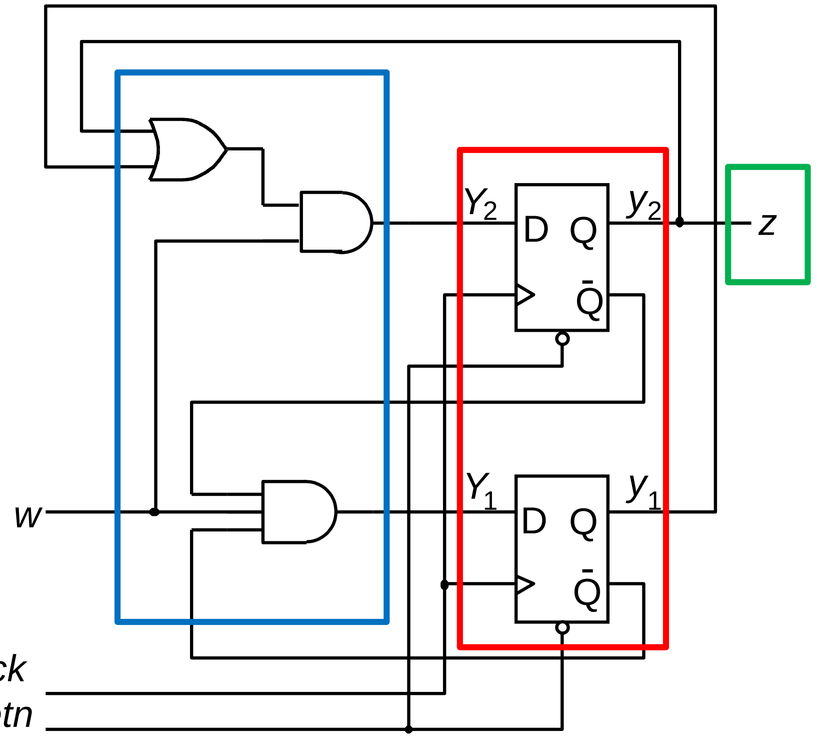
$$z = y_2$$

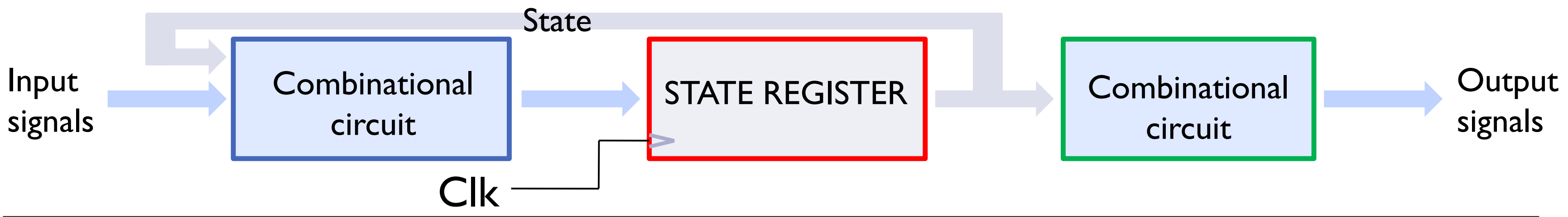
Implementation

$$Y_2 = wy_1 + wy_2 =$$
$$= w(y_1 + y_2)$$

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

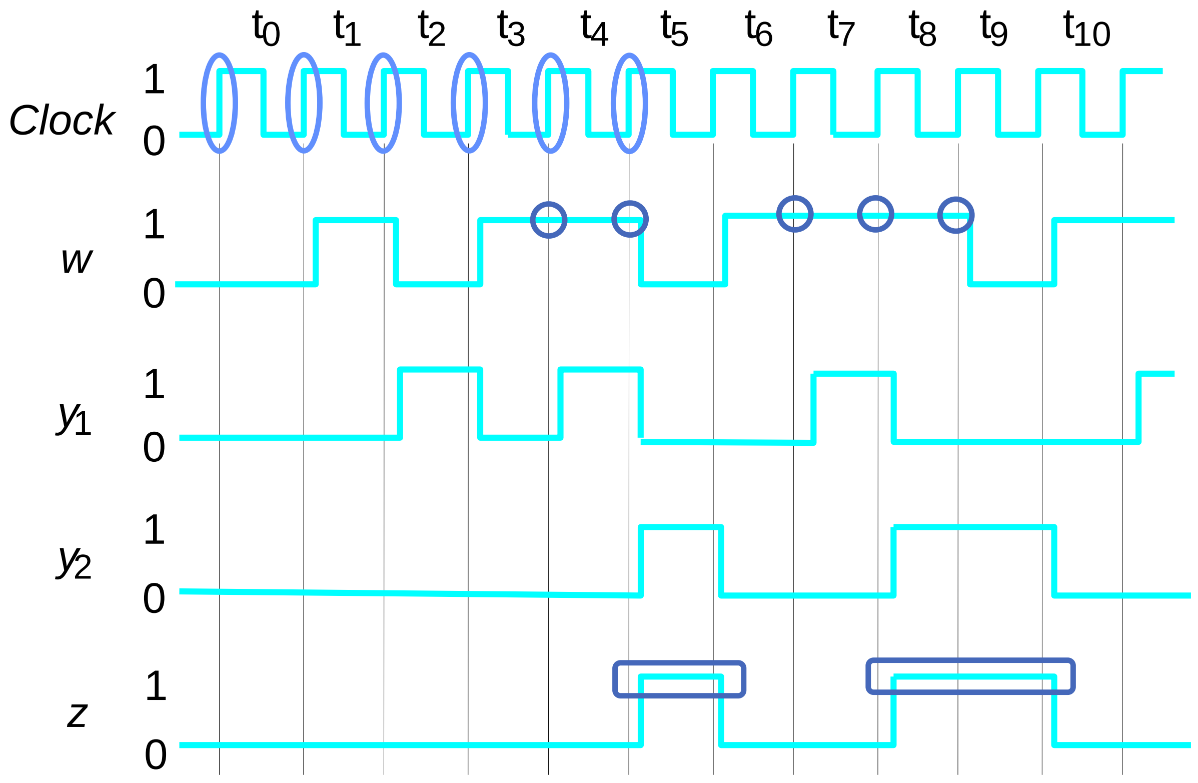
$$z = y_2$$





Timing diagram

State transitions occur only on the positive clock edge!

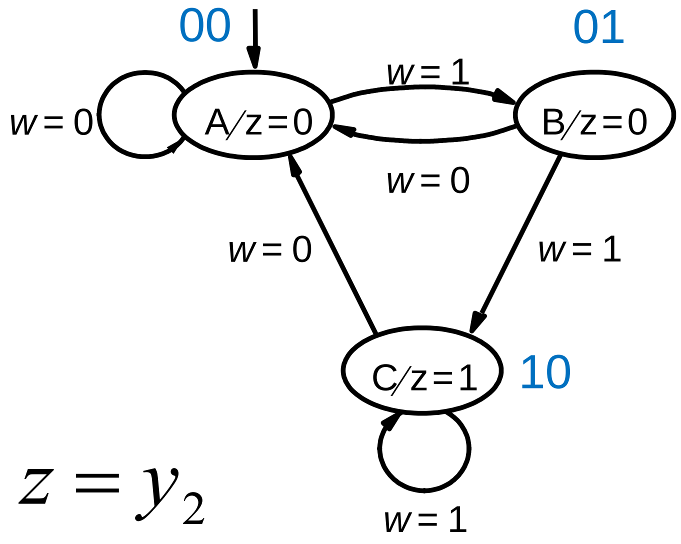

t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
Clock
w
y_1
y_2
z
1
0

y_2y_1

A = 00

B = 01

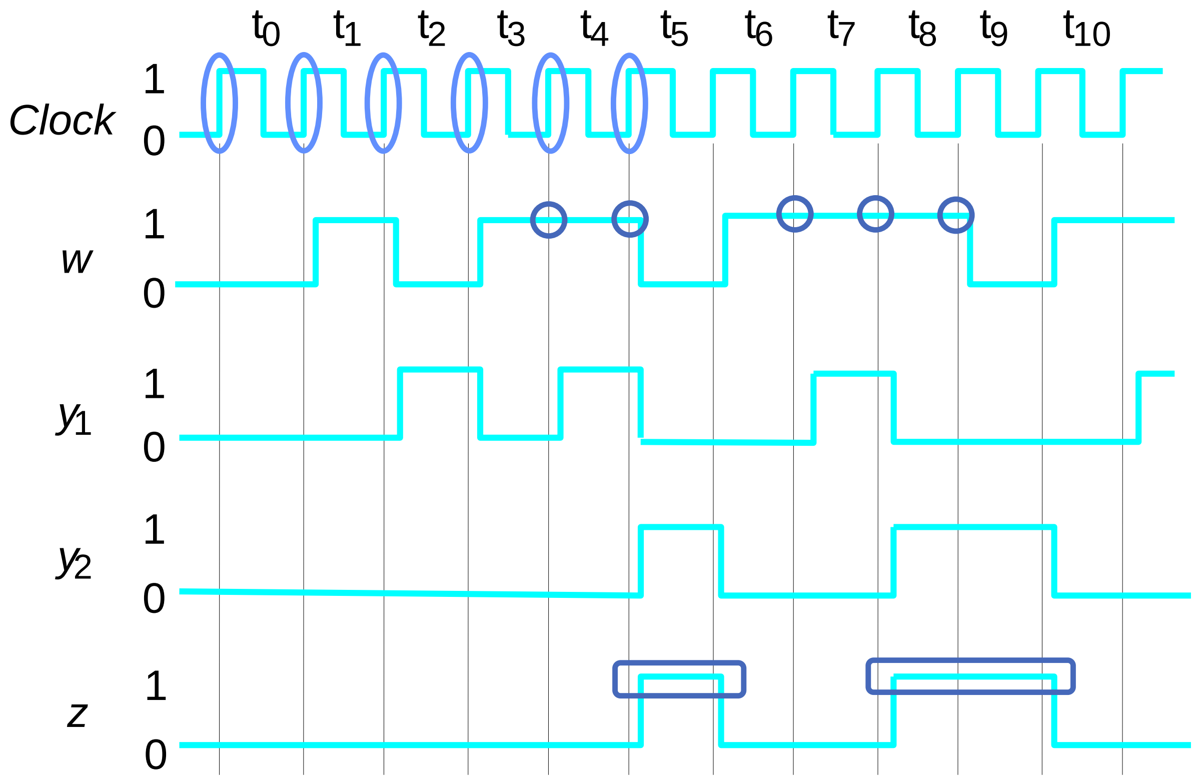
C = 10

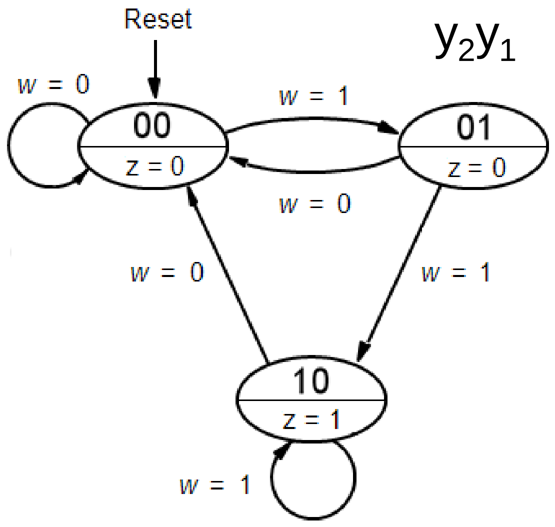


$$z = y_2$$

Timing diagram

State transitions occur only on the positive clock edge!





$$z = y_2$$

In other terms

Present state

$$y$$

Next state

$$y^+$$

Exercises and Hemert-book:

$$y_2^+ y_1^+ = f(y_2 y_1 w) \quad y_2^+ = f(y_2 y_1 w) \quad y_1^+ = f(y_2 y_1 w)$$

With another lineup

$$y_2^+ y_1^+ = f(y_2 y_1 w) \quad y_2^+ = f(y_2 y_1 w) \quad y_1^+ = f(y_2 y_1 w)$$

$y_2^+ y_1^+$

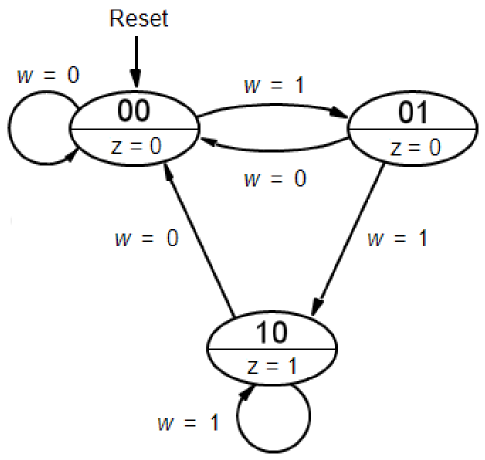
	$y_2 y_1$ \ w	0	1
A	00	00	01
B	01	00	10
	11	dd	dd
C	10	00	10

y_2^+

$y_2 y_1$ \ w	0	1
00	0	0
01	0	1
11	d	d
10	0	1

y_1^+

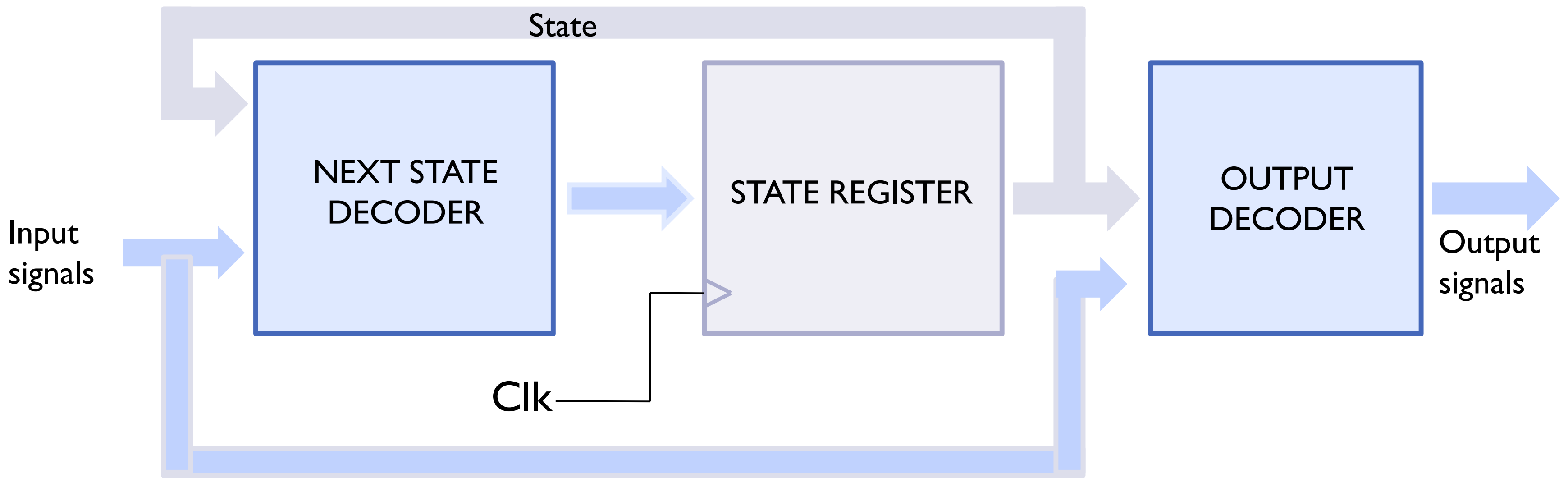
$y_2 y_1$ \ w	0	1
00	0	1
01	0	0
11	d	d
10	0	0



In exercises we use this method

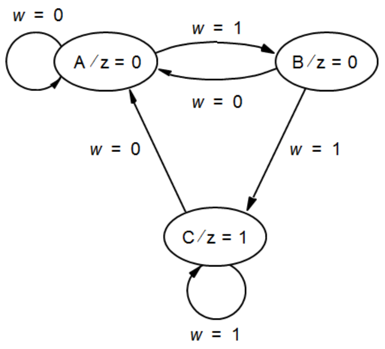
You could directly write down the codet state table as a ”Karnaugh map”.

Mealy-type machine

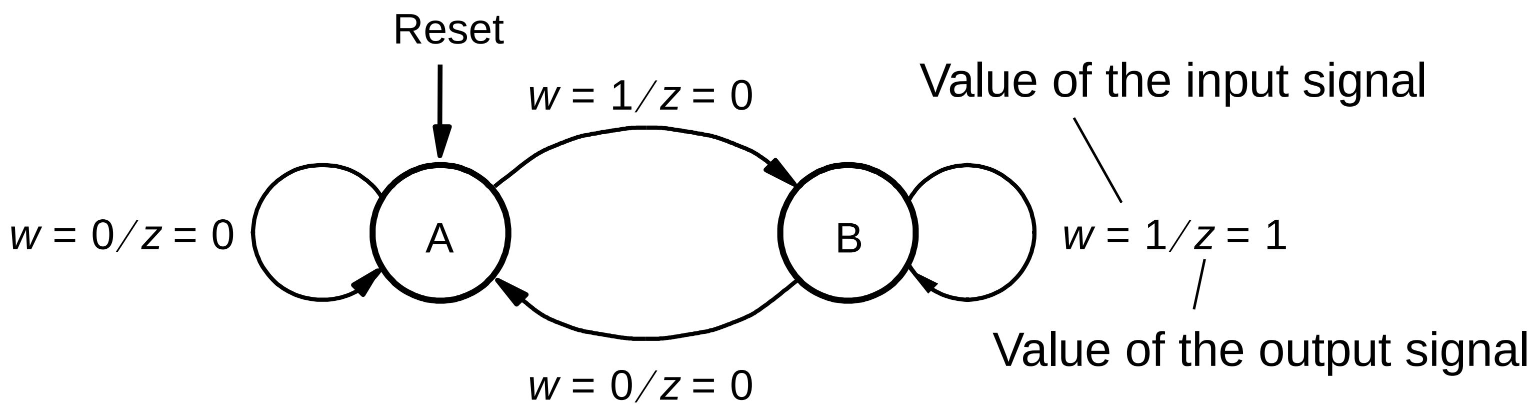


- In a Mealy machine, output signals depend on both the current state <u>and</u> inputs

Bit sequences detector Mealy machine



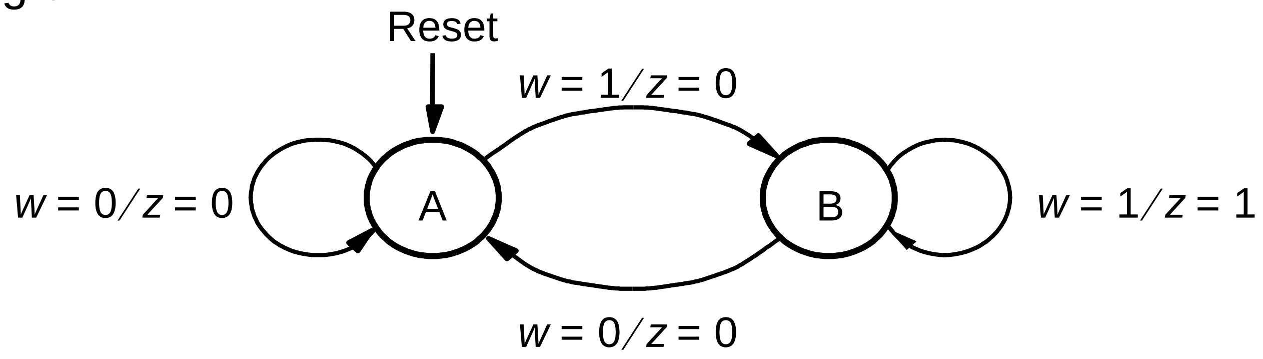
- The state diagram for the Mealy machine requires only two states
- The output depends on both the state and the input



A = previous value was 0
B = previous value was 1

State table

State Diagram



State Table

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

Two states - just one flip-flop is needed!

State Assignment

State Table

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

State-assigned Table

A = 0
B = 1

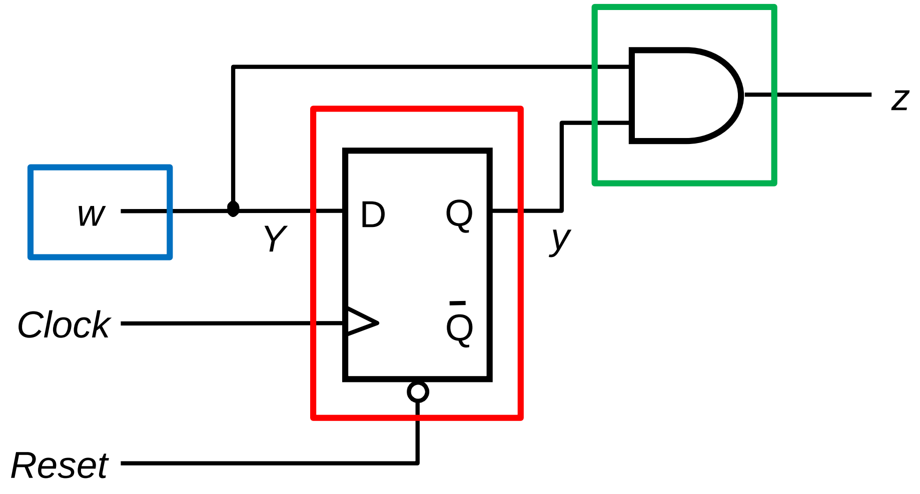
	Present state	Next state		Output	
		$w = 0$	$w = 1$	$w = 0$	$w = 1$
	y	Y	Y	z	z
A	0	0	$\boxed{1}$	0	0
B	1	0	$\boxed{1}$	0	1

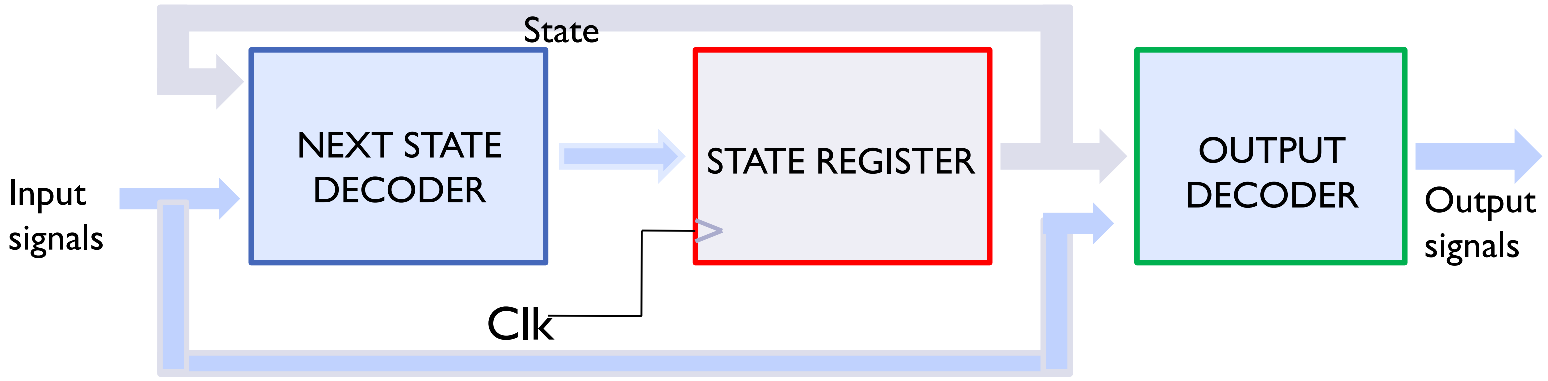
Y = w   z = yw

Implementation

$Y = w$

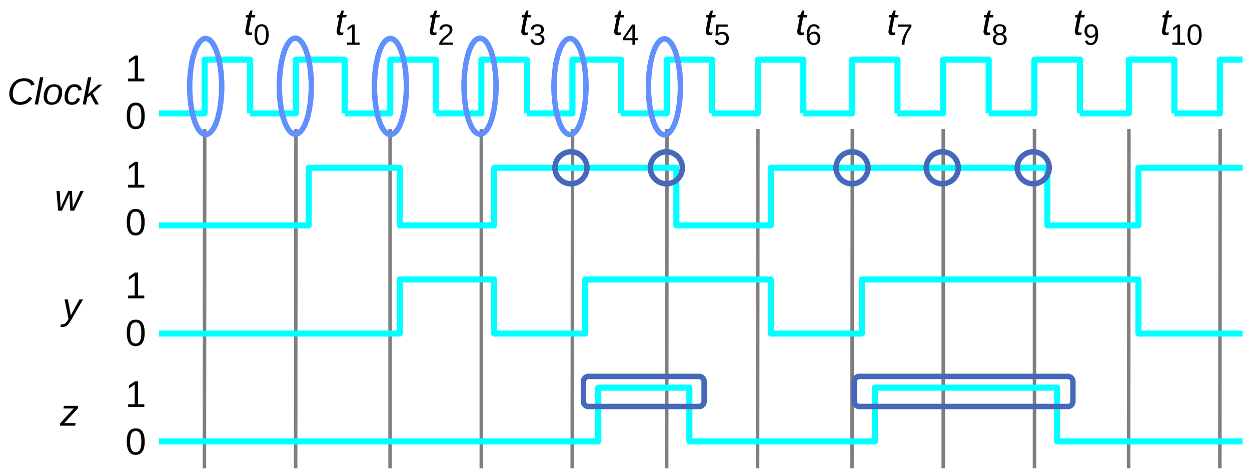
$z = yw$



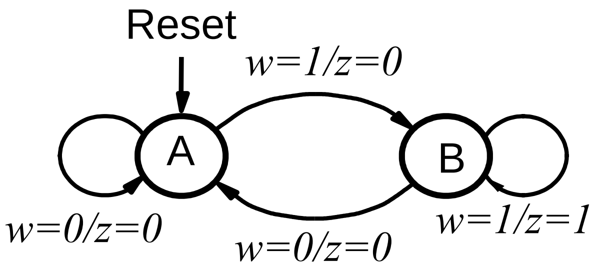


Timing diagram

- The output may change during the clock period because it is a function of the input signal
- Compared to Moore machine, Mealy machine “reacts” earlier (bit sequence is detected in t_4 compared to the t_5 in Moore-machine)



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

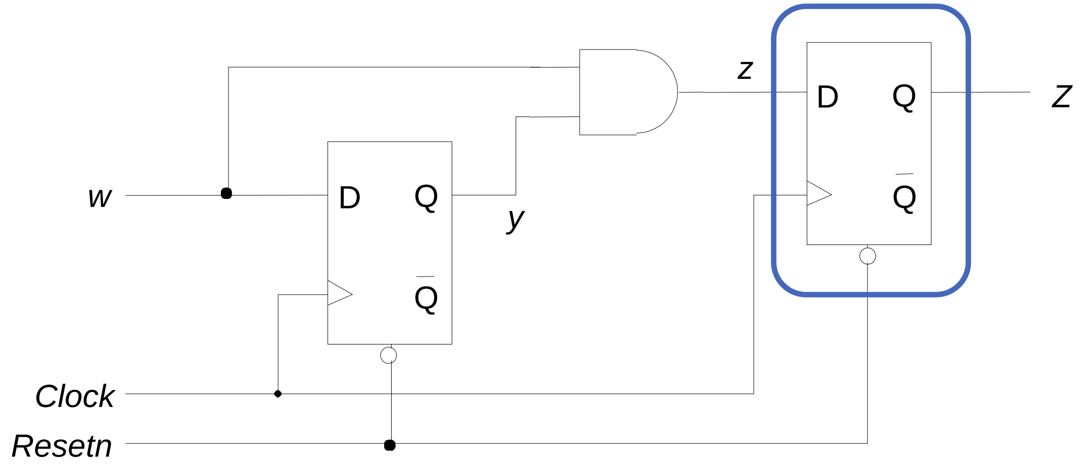


$z = yw$

	y
A	= 0
B	= 1

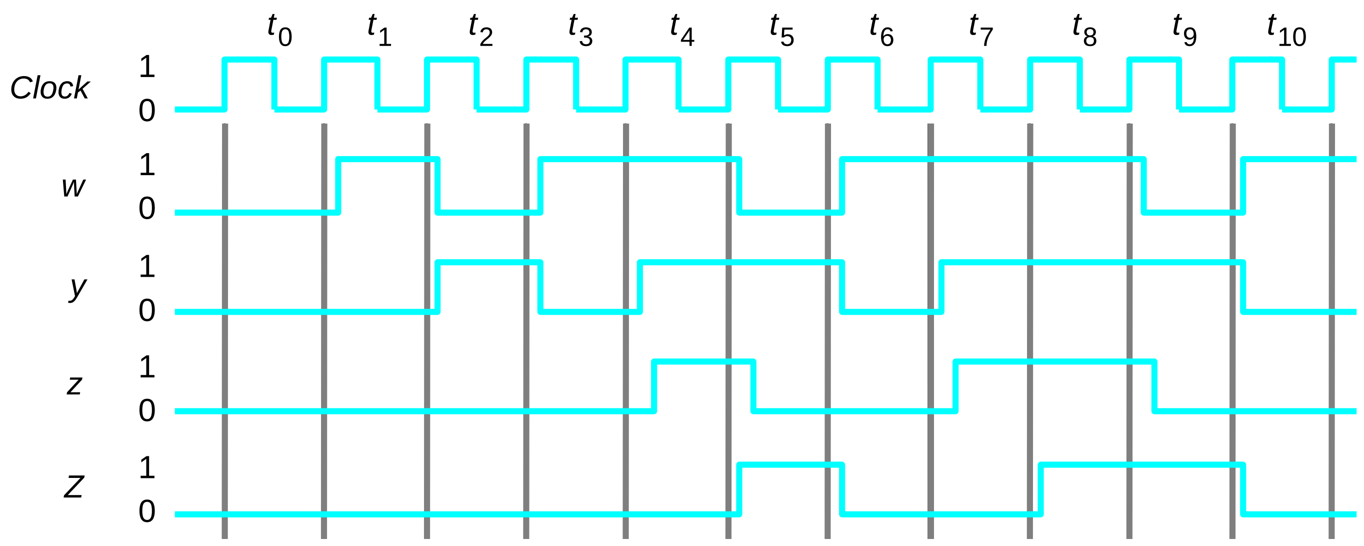
Mealy machine with output register

- In the Mealy machine the output can change during the entire clock period
- We can add a register (flip-flop) at the end in order to synchronize the output with the clock edge



Timing diagram

- Mealy machine with output registers has the same timing diagram as Moore machine



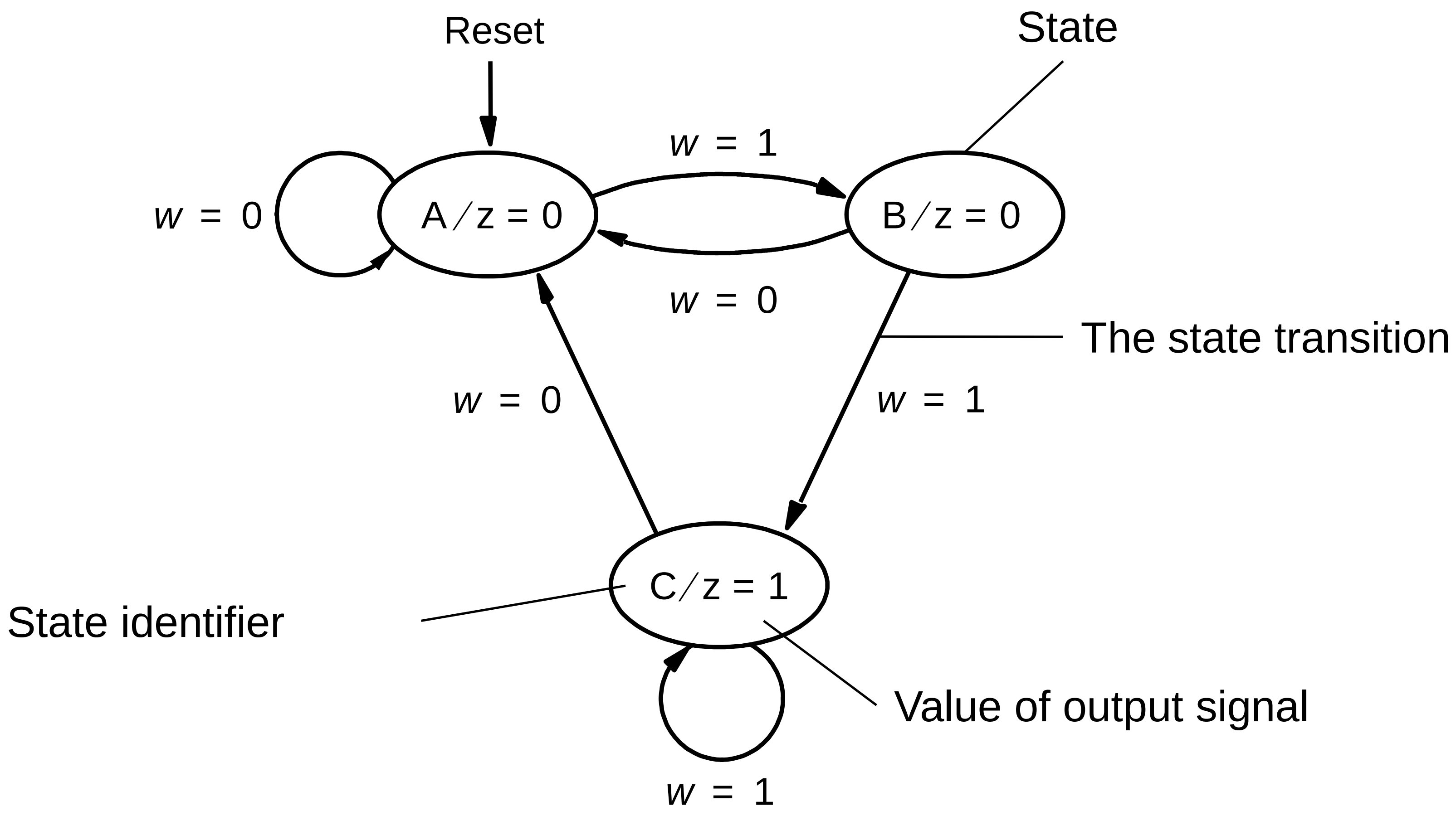
Discussion
Moore vs Mealy

- Moore machine output values depend only on the current state
- Mealy machine output values depend on the current state and the values of the input signals
- Mealy machine often needs fewer state
- Mealy machine outputs are not synchronized with the clock, which is why an output register is often added

Alternative state assignments

- The choice of state assignment may play a major role in implementation as it affects the logic for
  - next-state decoder
  - output decoder

State diagram
bit sequences detector



Binary encoding

- Binary code follows the representation of a binary number, that is, the order is 00, 01, 10, 11

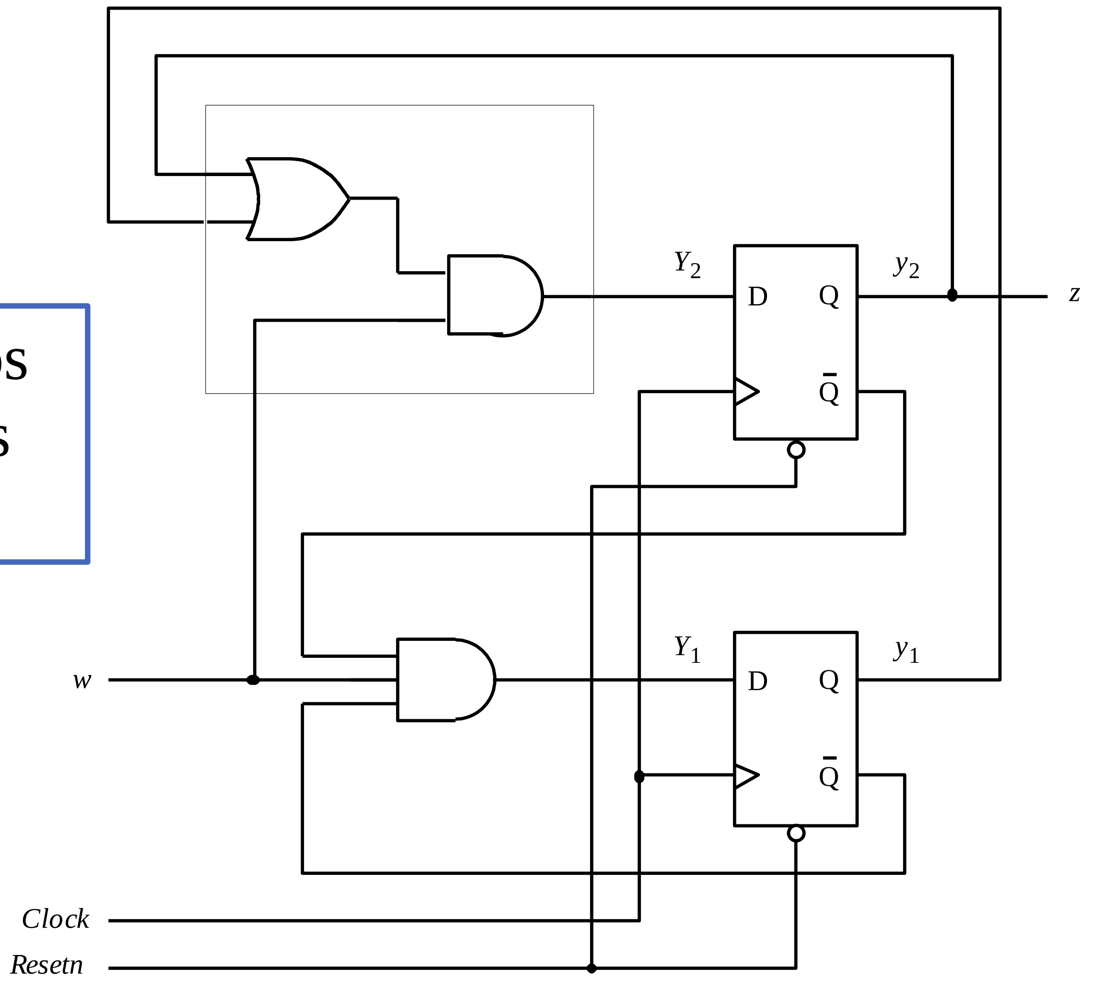
A = 00
B = 01
C = 10
11

➡

	Present state	Next state		Output
		$w = 0$	$w = 1$	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*

Implementation of bit sequence detector for binary code

2 D-flip-flops
2 AND-gates
1 OR-gate



Gray code

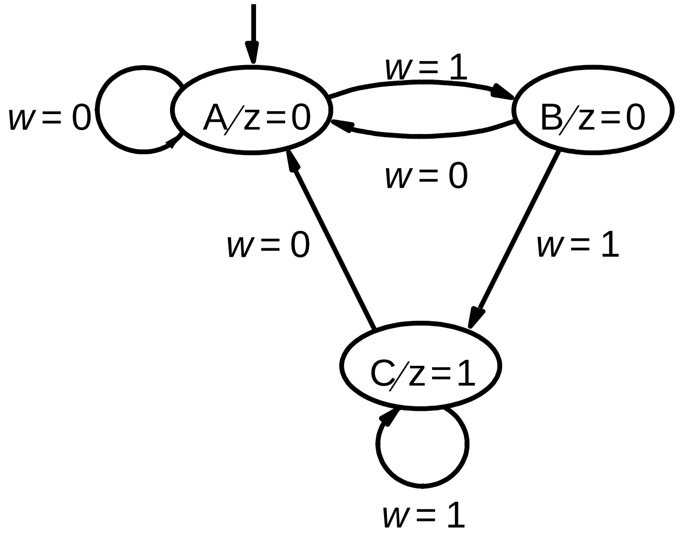
- In Gray code, one bit is changed at a time, i.e, 00, 01, 11, 10
- Gray code is good for counters

A = 00
B = 01
C = 11
10

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

State Table

State Diagram



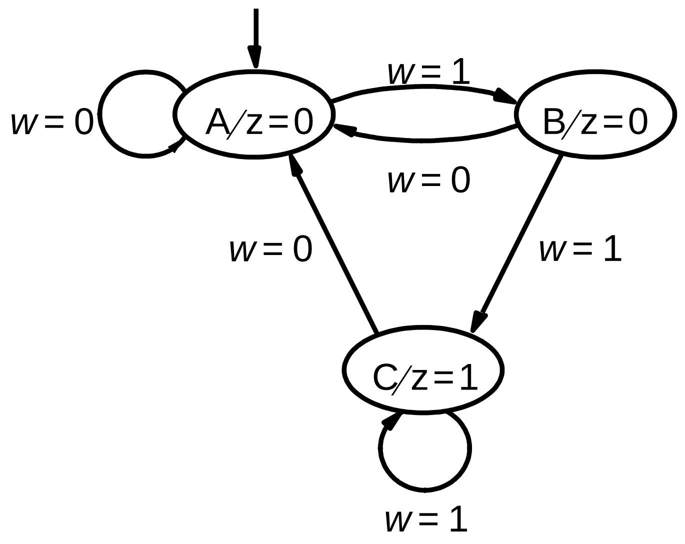
State Table

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

Three states - two bits are needed!

State Table

State Diagram



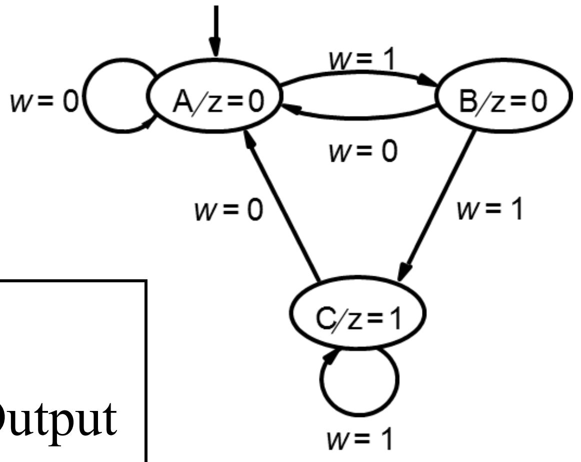
State Table

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

Three states - two bits are needed!

Gray code

- In Gray code, one bit is changed at a time, i.e, 00, 01, 11, 10
- Gray code is good for counters



A = 00
B = 01
C = 11
10

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

Gray code

- In Gray code, one bit is changed at a time, i.e, 00, 01, 11, 10
- Gray code is good for counters

A = 00
B = 01
C = 11
10

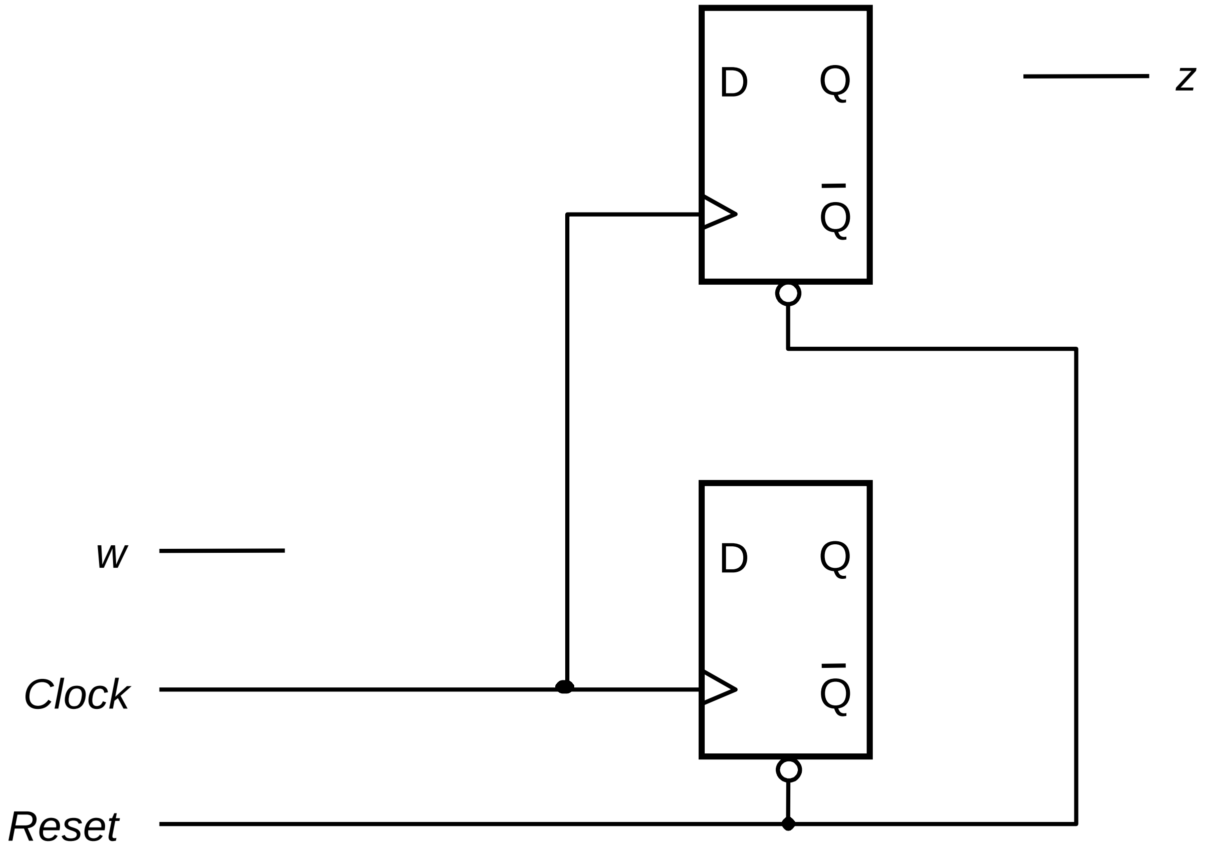
	Present state	Next state		Output
		$w = 0$	$w = 1$	
	y_2y_1	Y_2Y_1	Y_2Y_1	z
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

$$Y_1 = w$$

$$Y_2 = wy_1$$

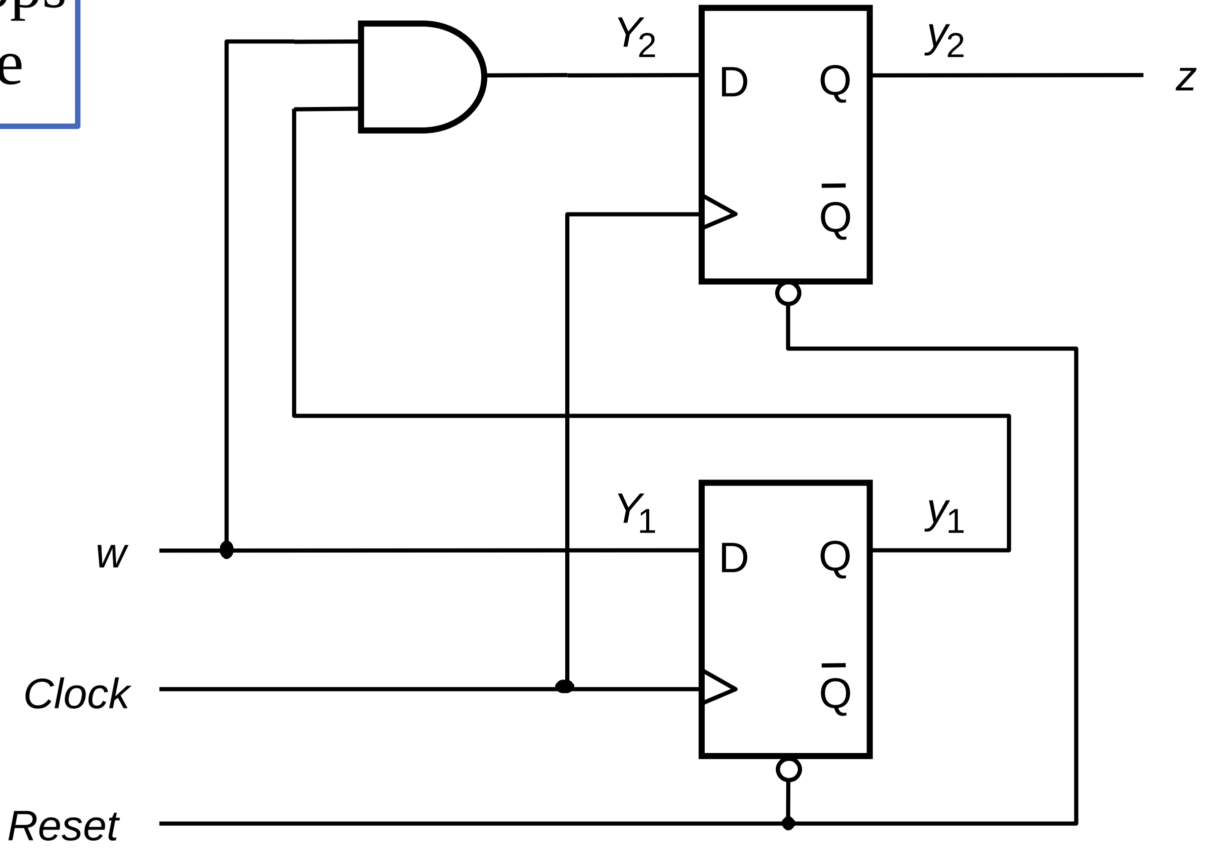
$$z = y_2$$

Implementation of bit sequence detector for Gray code



Implementation of bit sequence detector for Gray code

2 D-flip-flops
1 AND-gate



One-hot encoding

- One-hot encoding uses one flip-flop per state
- For each state, only one bit is 'hot' (1), all other bits are 0, i.e. 0001, 0010, 0100, 1000
- One-hot encoding typically minimizes the combinatorial logic, but increases the number of flip-flops

Which encoding is best?

- There is no code which is the best in every situation, but it all depends on the state diagram
- One can also use 'own codes' suitable for implementation, for example 00, 11, 10, 01

Summary

- There are two types of state machines
  - Moore machine
  - Mealy machine
- Different state assignments give us different circuit implementations
- Next lecture: BV pp. 528-566