

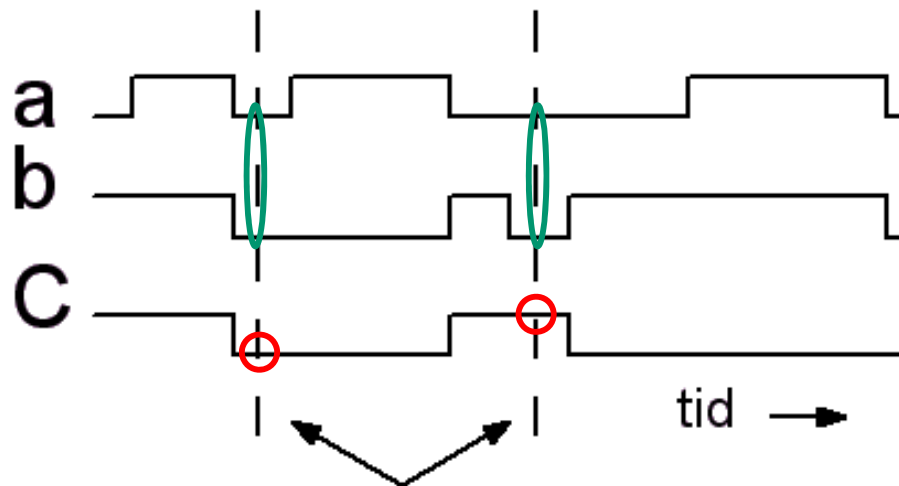
Digital Design IE1204

Föreläsningsbilder av William Sandqvist

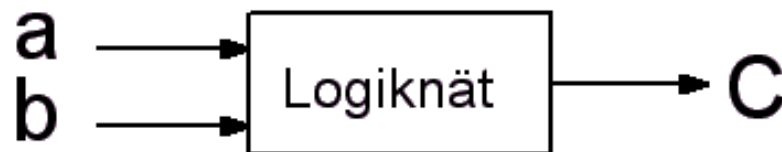
Tillståndsautomater

Carl-Mikael Zetterling
bellman@kth.se

Sekvensnät



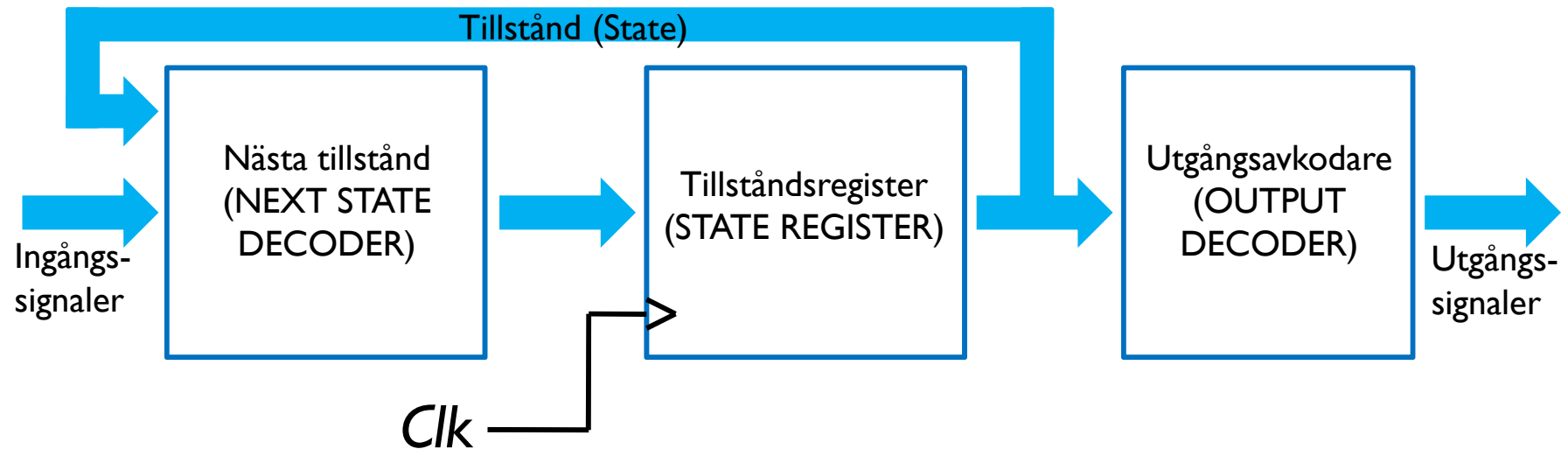
Samma insignal
kan ge olika utsignal



Om en och samma insignal
kan ge upphov till olika
utsignal, är logiknätet ett
sekvensnät.

Det måste då ha ett *inre minne* som gör att utsignalen
påverkas av både nuvarande
och föregående insignaler!

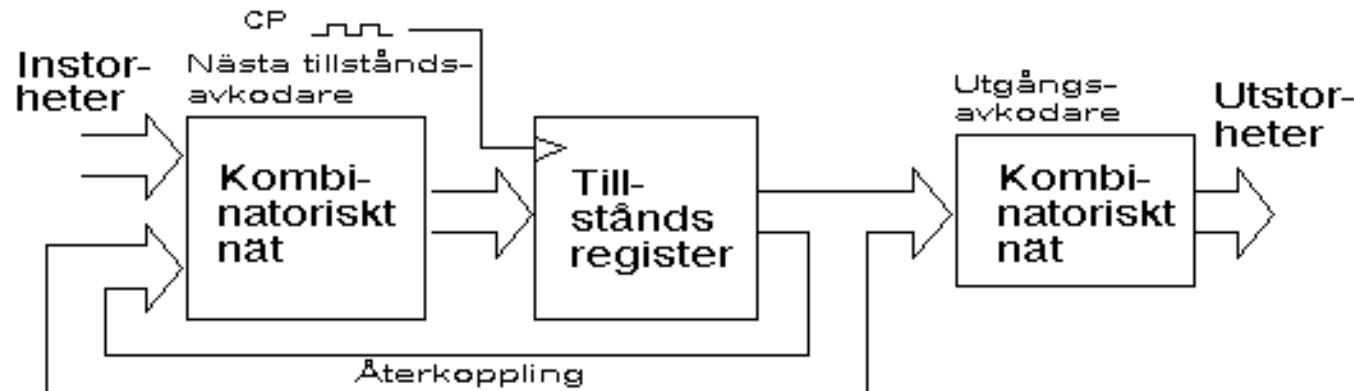
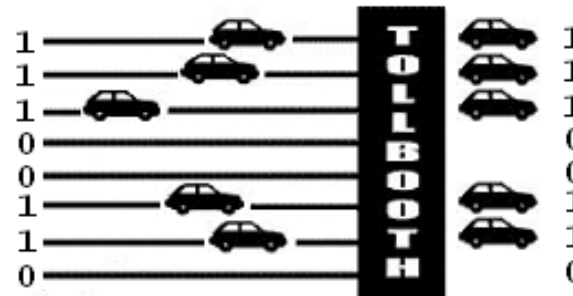
Moore-automat



För Moore-automaten beror utsignalerna på insignalerna och det inre tillståndet. Det inre minnet är tillståndsregistret som består av **D-vippor**.

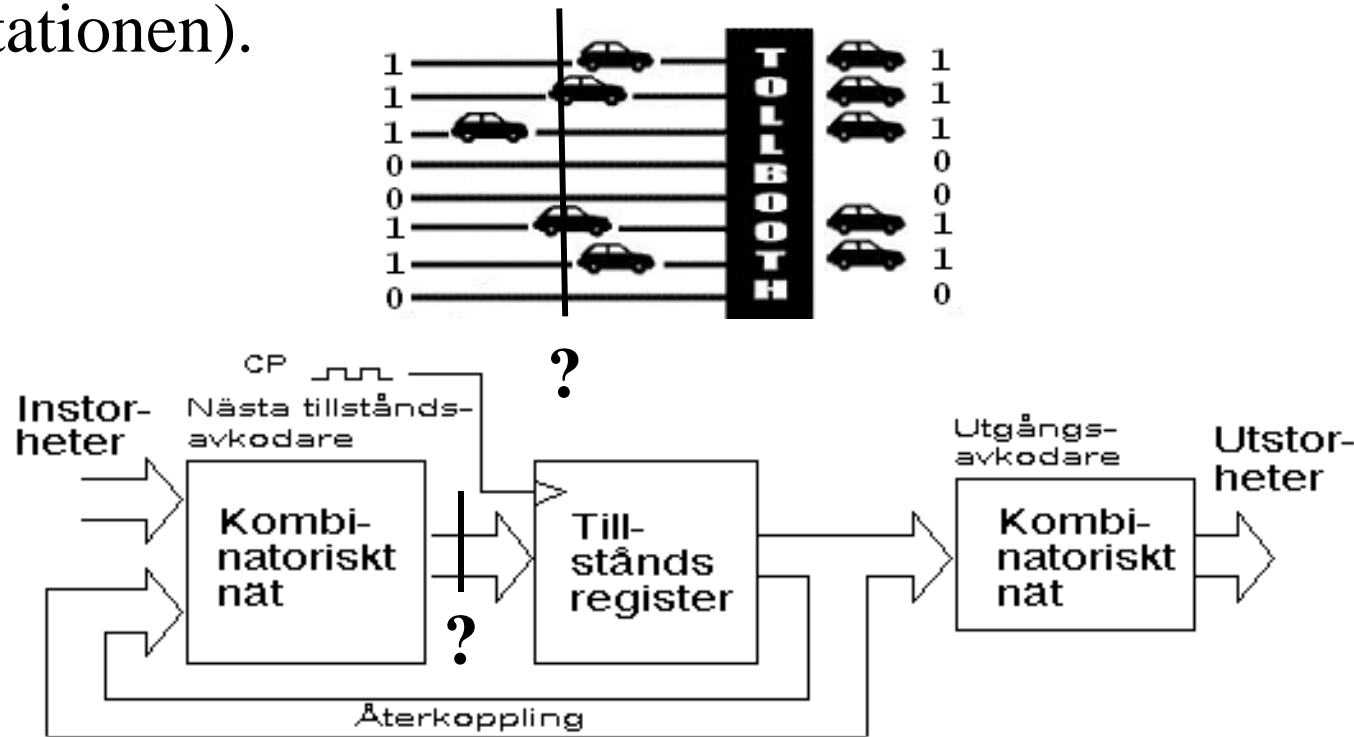
Tillståndsregistrets D-vippor

Tillståndsregistrets D-vippor bromsar upp *kapplöpningen* mellan signalerna tills värdet är stabilt. (Jämför med tullstationen).



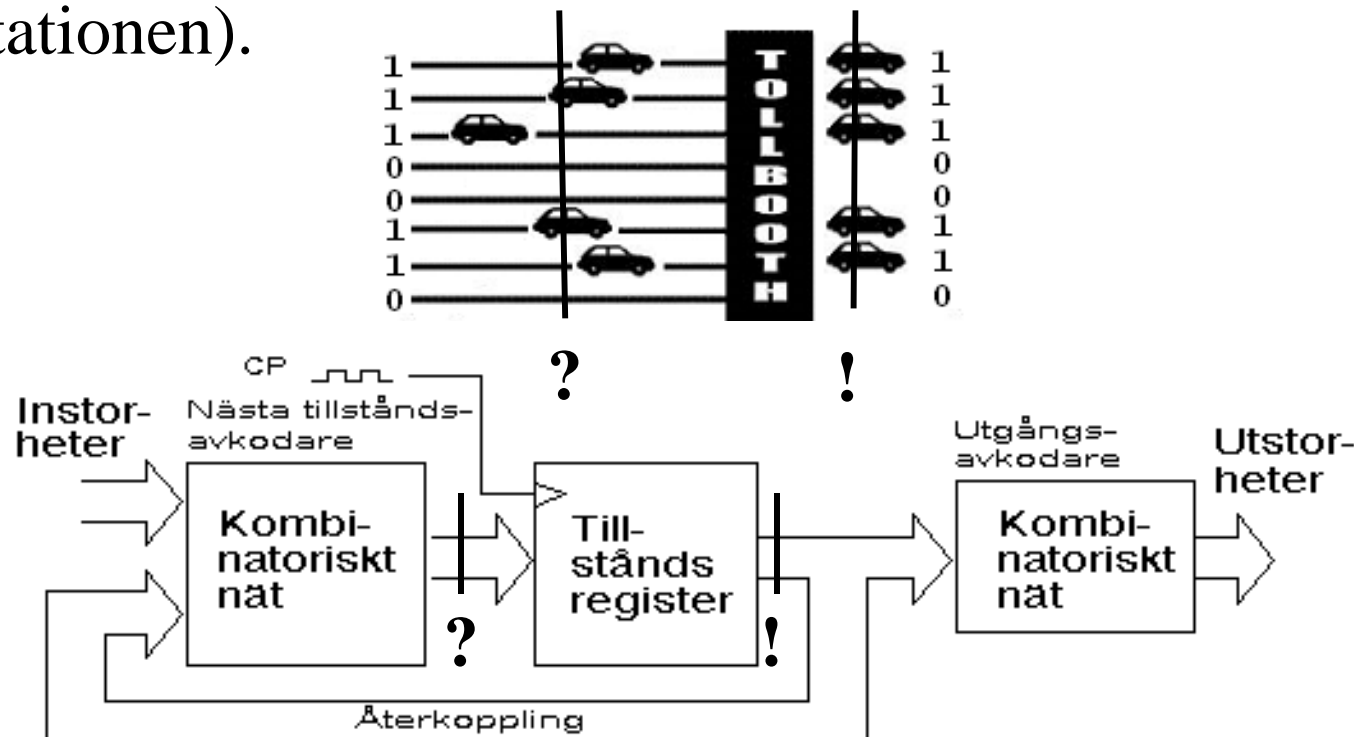
Tillståndsregistrets D-vippor

Tillståndsregistrets D-vippor bromsar upp *kapplöpningen* mellan signalerna tills värdet är stabilt. (Jämför med tullstationen).



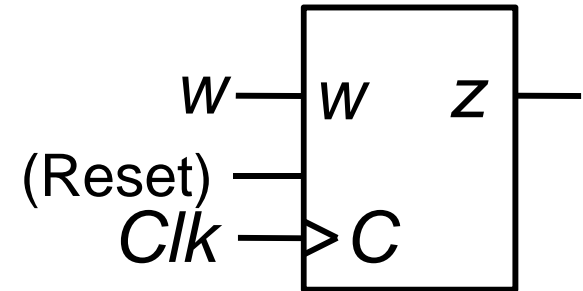
Tillståndsregistrets D-vippor

Tillståndsregistrets D-vippor bromsar upp *kapplöpningen* mellan signalerna tills värdet är stabilt. (Jämför med tullstationen).



Designexempel ”två i rad”

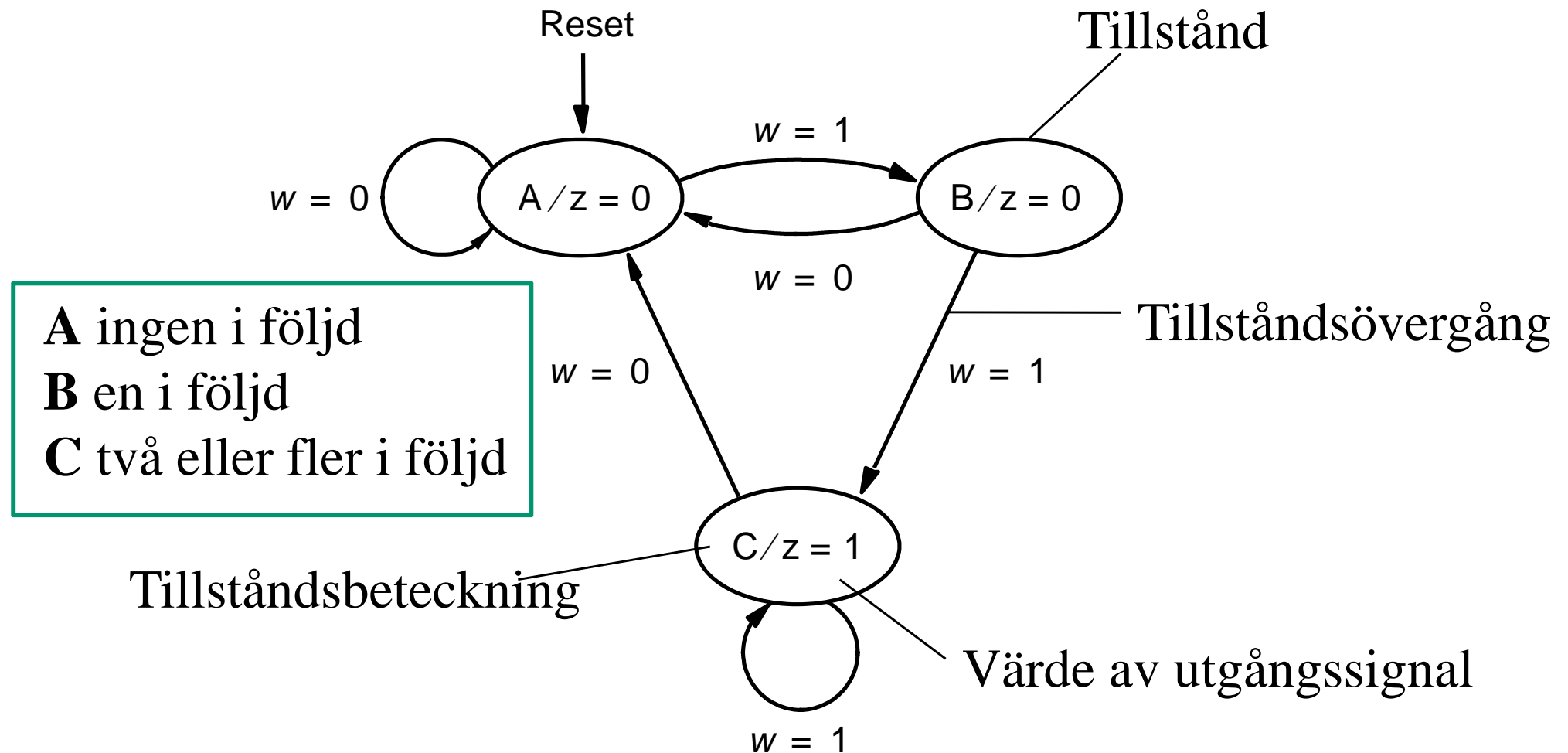
Sekvensdetektor. Om w har varit 1 under två (eller fler) klockcykler i rad skall $z = 1$.



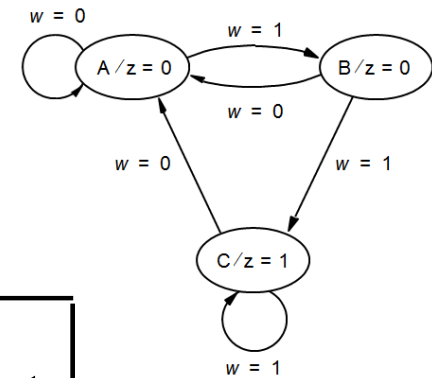
Specifikation

Sekvenskretsen har en ingång w och en utgång z
Om ingången w har varit 1 under nuvarande och föregående klockcykel så ska utgången z sättas till 1
Använd en Moore-automat med D-vippor för att realisera konstruktionen

Tillståndsdigram ”två i rad”



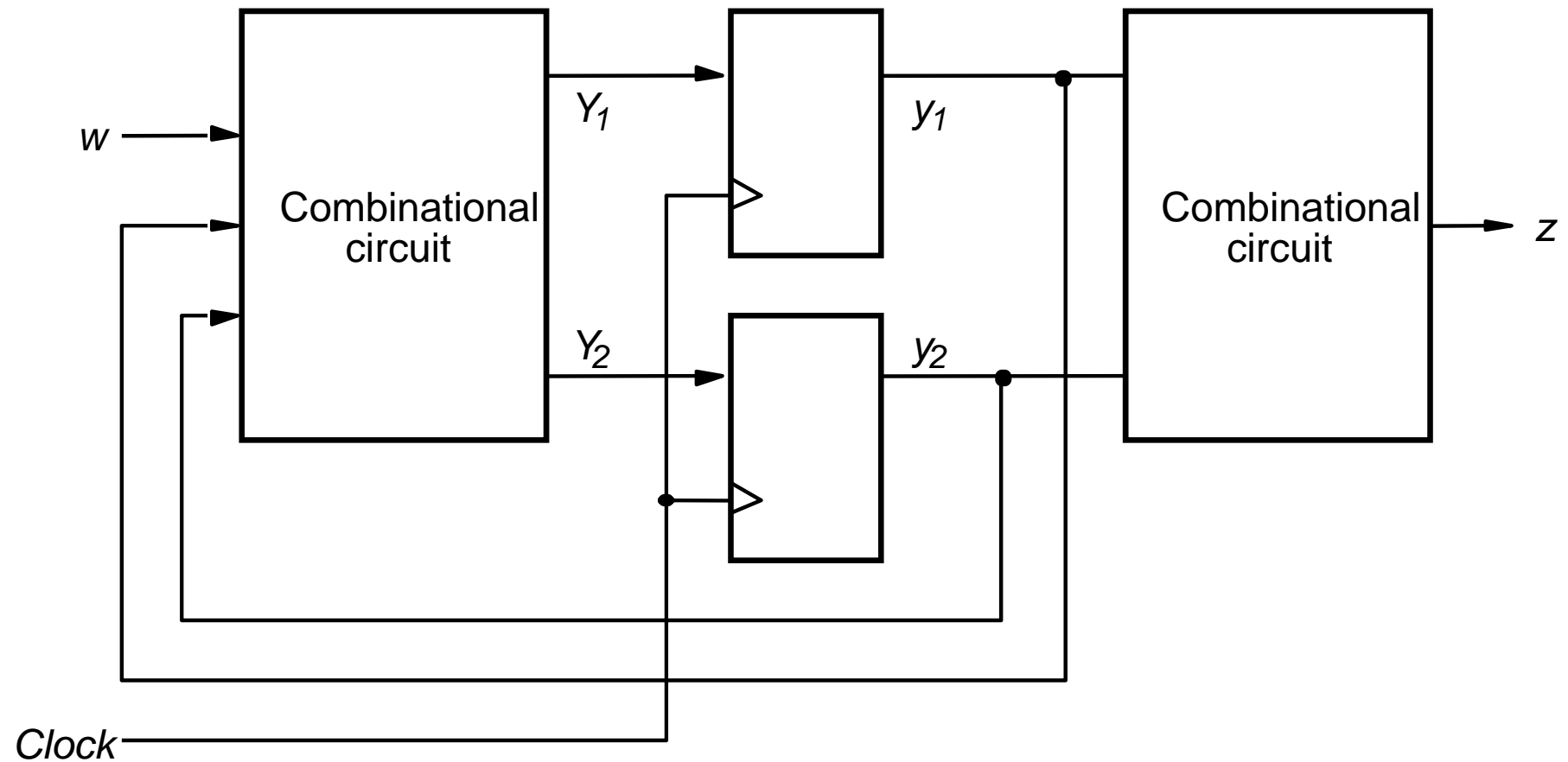
Tillståndstabell



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

Tre tillstånd – två vippor behövs för att minnas tillståndets nummer!

”två i rad” som Moore-automat



Designbeslut

- Designern måste bestämma vilka vippor som ska användas
D-, T-, eller JK-vippa
- Designern måste välja koden för varje tillstånd

Designbeslut

Denna gång givet:

- D-vippor
- Tillståndsavkodning $A = 00$, $B = 01$, $C = 10$
- Koden 11 ska *inte* förekommer.
Vi väljer don't care.

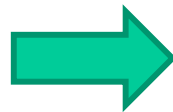
Kodad tillståndstabell

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

A = 00

B = 01

C = 10



A
B
C

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

$$Y_2 Y_1 = f(y_2 y_1 w) \quad z = f(y_2 y_1)$$

Nästa tillståndsavkodare

Nästa tillståndsavkodaren består av de två logiknäten som finns som ingångsnät till de två vipporna.

För att kunna minimera logiknäten skriver man in sanningstabellerna i Karnaughdiagram.

$$Y_2 Y_1 = f(y_2 y_1 w) \quad Y_2 = f(y_2 y_1 w) \quad Y_1 = f(y_2 y_1 w)$$

$$z = f(y_2 y_1)$$

Från kodad tillståndstabell till Karnaughdiagram

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

Present state y_2y_1	Next state		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	10	1
11	<i>dd</i>	<i>dd</i>	<i>d</i>



		Y_1			
		00	01	11	10
w	0	0	0	d	0
	1	1	0	d	0

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

Present state y_2y_1	Next state		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	10	0
10	00	10	1
11	<i>dd</i>	<i>dd</i>	<i>d</i>



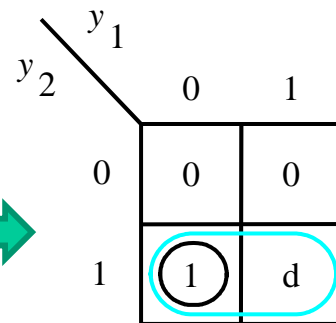
		Y_2			
		00	01	11	10
w	0	0	0	d	0
	1	0	1	d	1

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

Utgångsavkodaren

$$z = f(y_2 y_1)$$

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



$$z = y_2$$

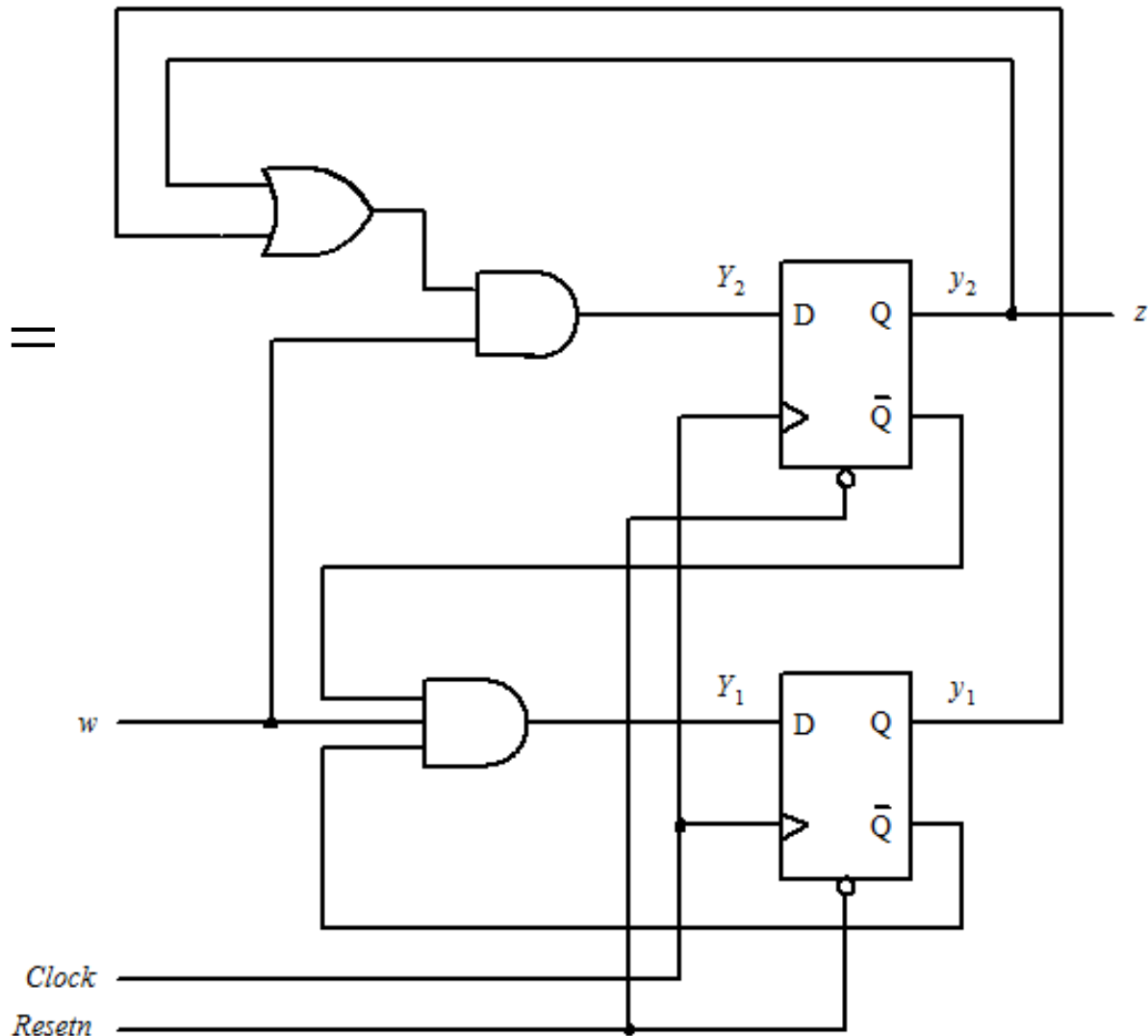
Implementeringen

$$Y_2 = wy_1 + wy_2 =$$

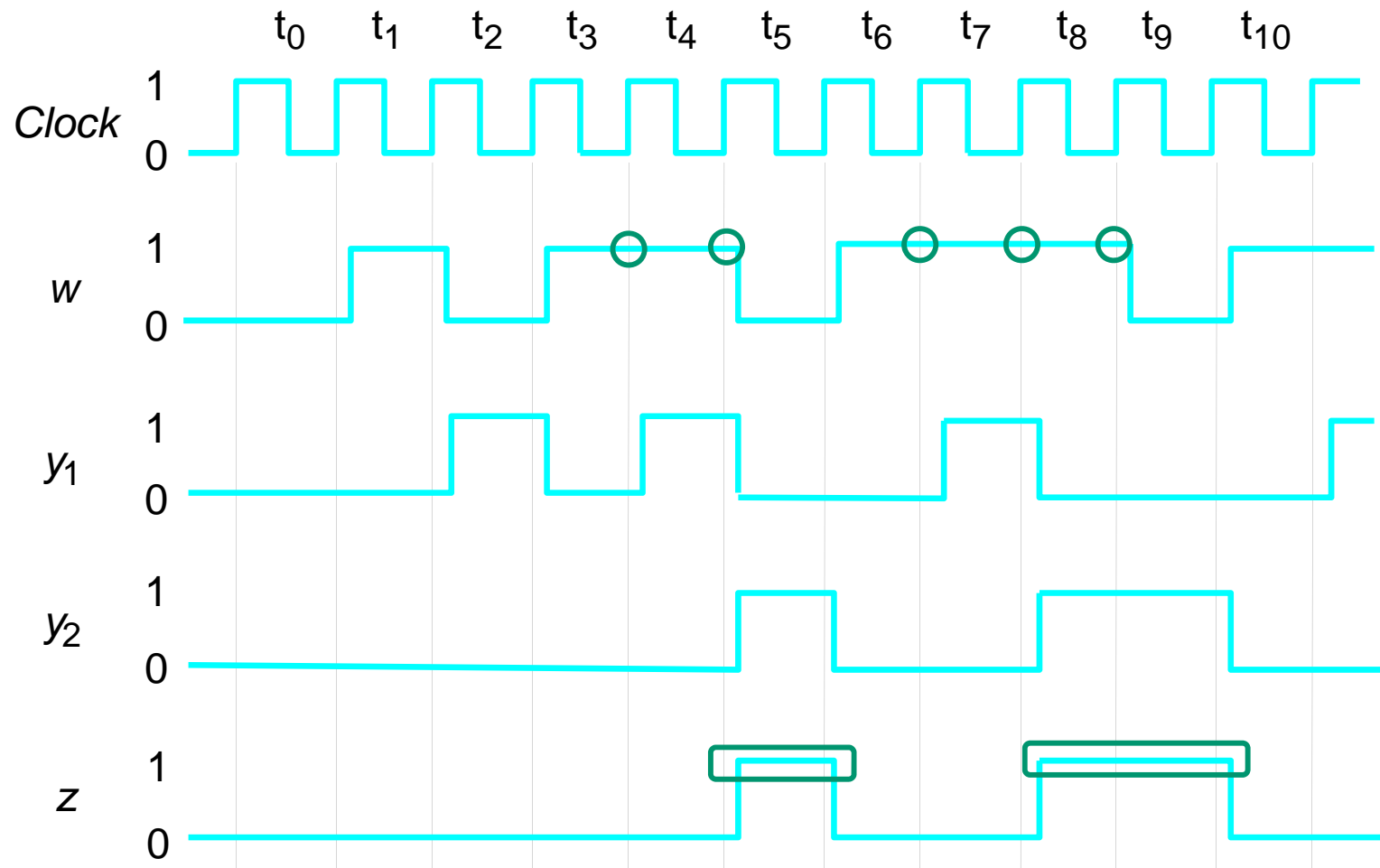
$$= w(y_1 + y_2)$$

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

$$z = y_2$$



Tidsdiagram ”två i rad”



Tillståndsover-
gångar sker
bara på den
positiva
klockflanken!

Med andra beteckningar

Nuvarande tillstånd

y

Nästa tillstånd

y^+

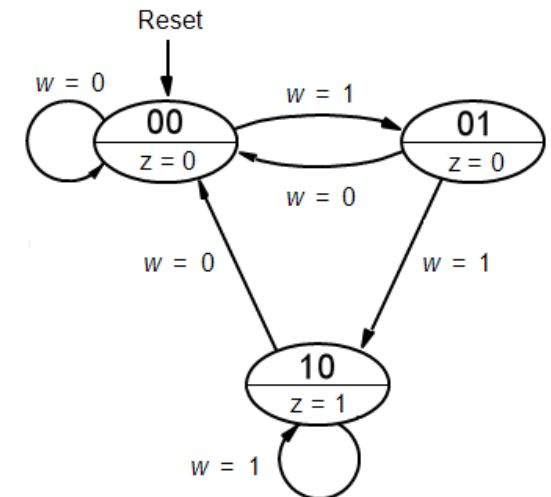
Med annan uppställning

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

$y_2 y_1$		w		$y_2^+ y_1^+$	
				0	1
A	00			00	01
B	01			00	10
	11			<i>dd</i>	<i>dd</i>
C	10			00	10

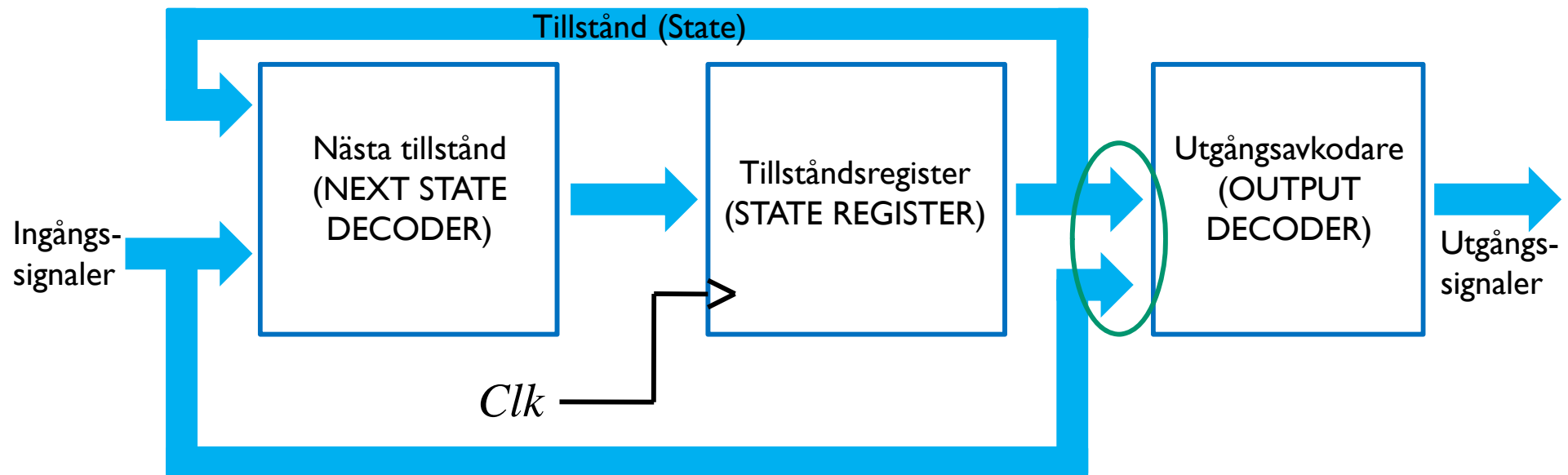
$y_2 y_1$		w		y_2^+	
				0	1
00				0	0
01				0	1
11				<i>d</i>	<i>d</i>
10				0	1

$y_2 y_1$		w		y_1^+	
				0	1
00				0	1
01				0	0
11				<i>d</i>	<i>d</i>
10				0	0



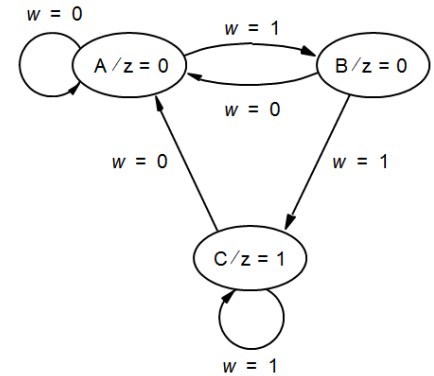
Man kan direkt ställa upp den kodade tillståndstabellen som ett "Karnaugh-diagram".

Mealy-automat



I en **Mealy**-Automat beror utgångssignalerna både på nuvarande tillstånd *och* ingångarna

Moore automaten



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

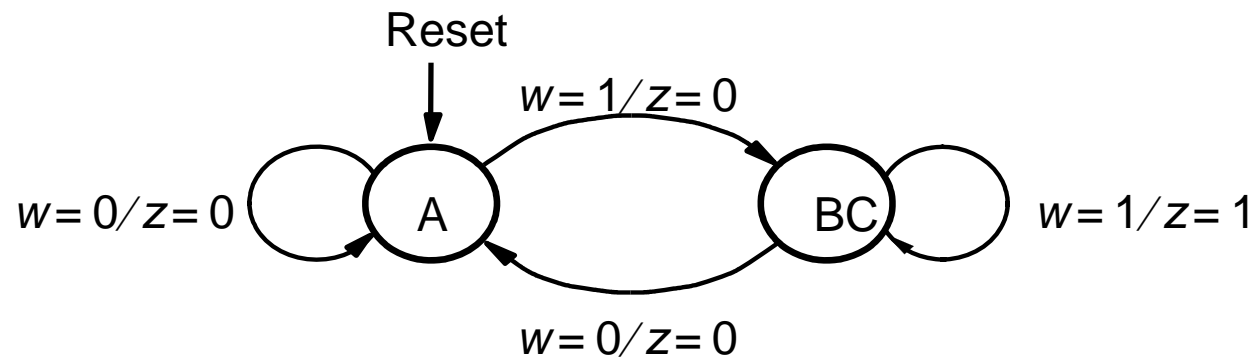
Mealy automaten

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

Två av tillstånden som bara skiljer i utsignal kan slås ihop – insignalen kan användas för att skilja utsignalerna!

Mealy automaten

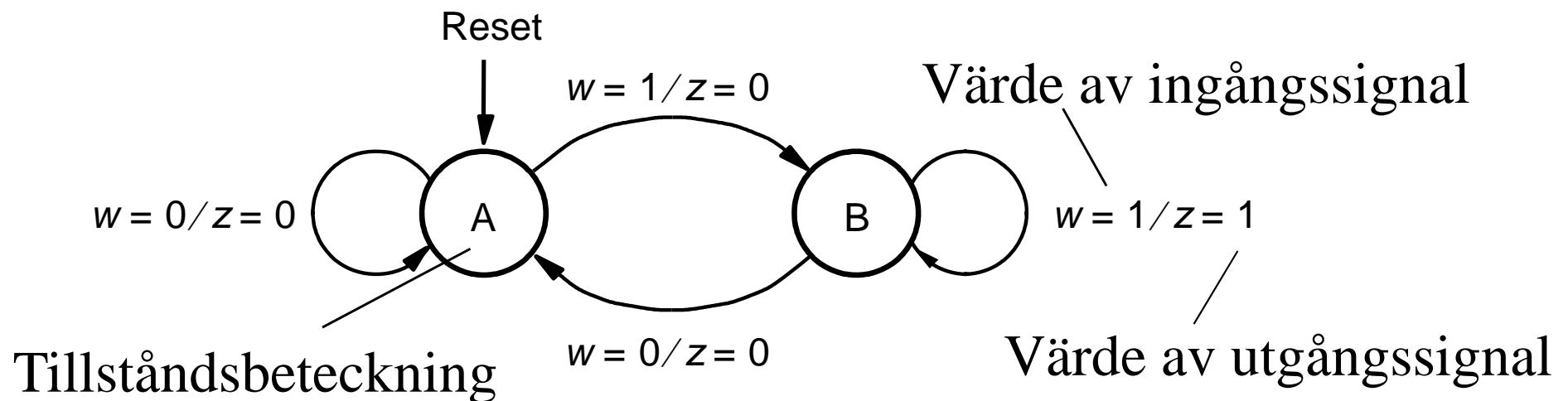
Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	BC	0	0
BC	A	BC	0	1



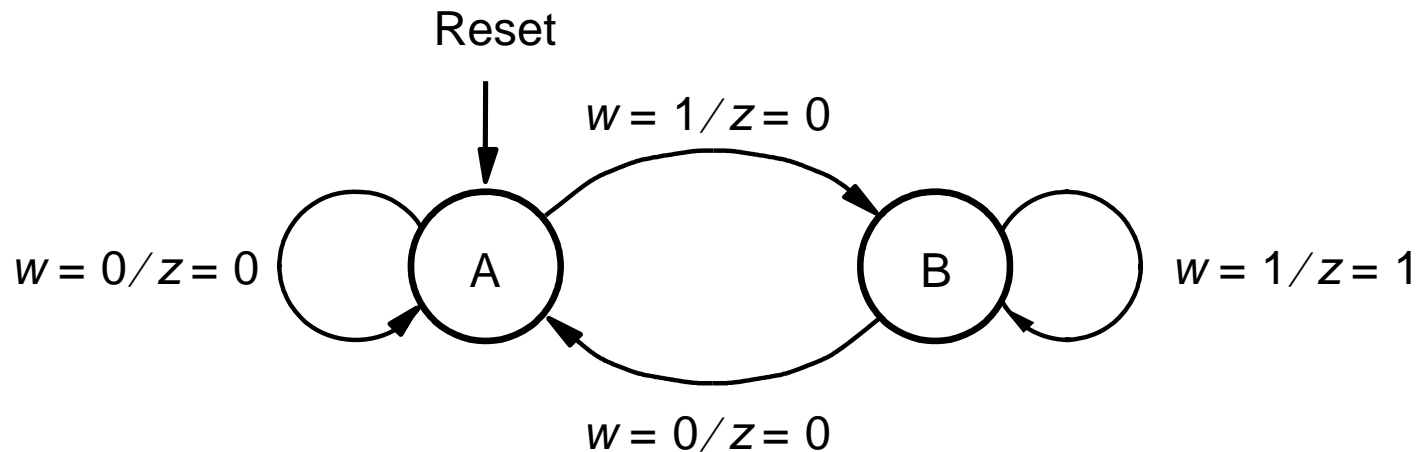
Tillståndsdigram Mealy

”Två i rad”

- Tillståndsdigrammet för Mealy-automaten behöver bara två tillstånd
- Utsignalen beror på *både* tillstånd och insignaler



Tillståndstabell



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

Två tillstånd – bara en vipa behövs!

Kodad tillståndstabell

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

$$Y = f(yw) \quad z = f(yw)$$

$$\mathbf{A} = 0$$

$$\mathbf{B} = 1$$

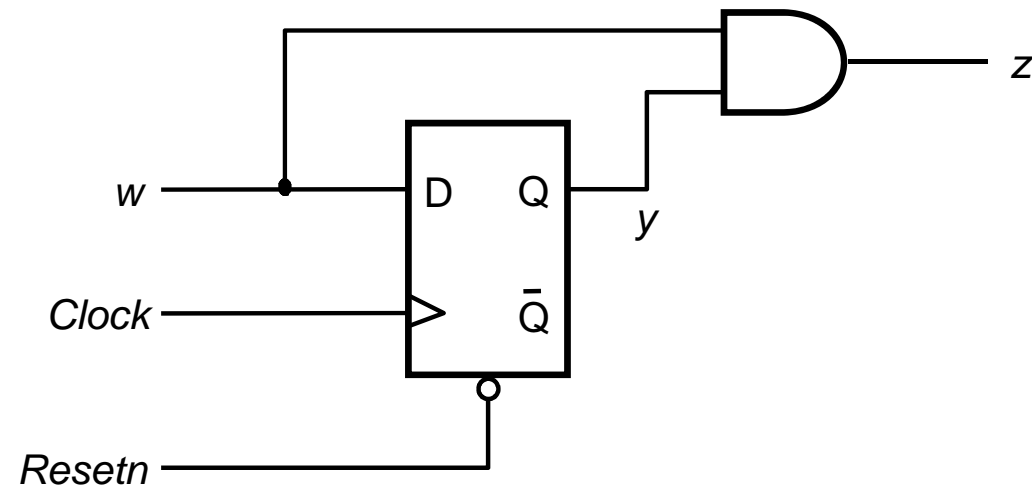


A
B

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

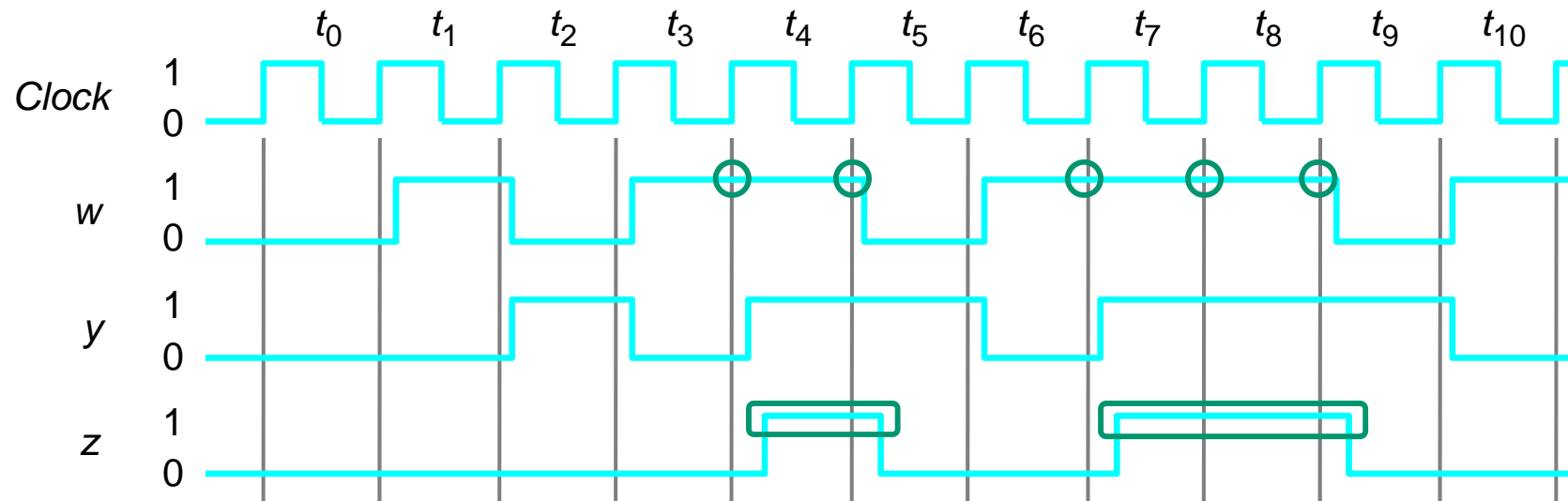
Direkt ur tabellen: $Y = w \quad z = yw$

Implementeringen



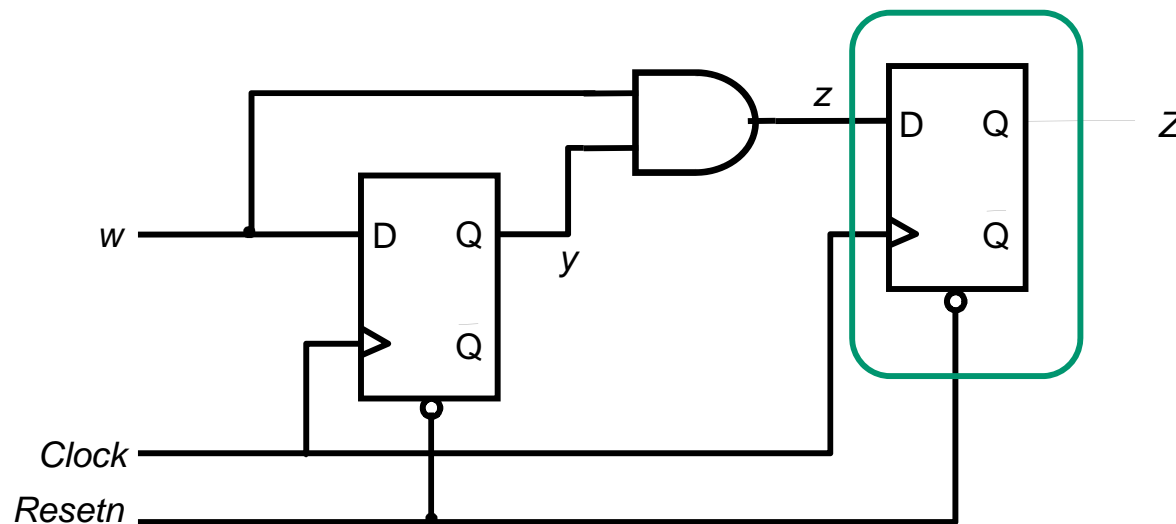
Tidsdiagram

- Utsignalen kan ändras under hela klockperioden eftersom den är en funktion av insignalen
- Jämfört med Moore-automaten så 'reagerar' Mealy-automaten *tidigare* (bitsekvensen detekteras i t_4 jämfört med t_5 i Moore-automaten)



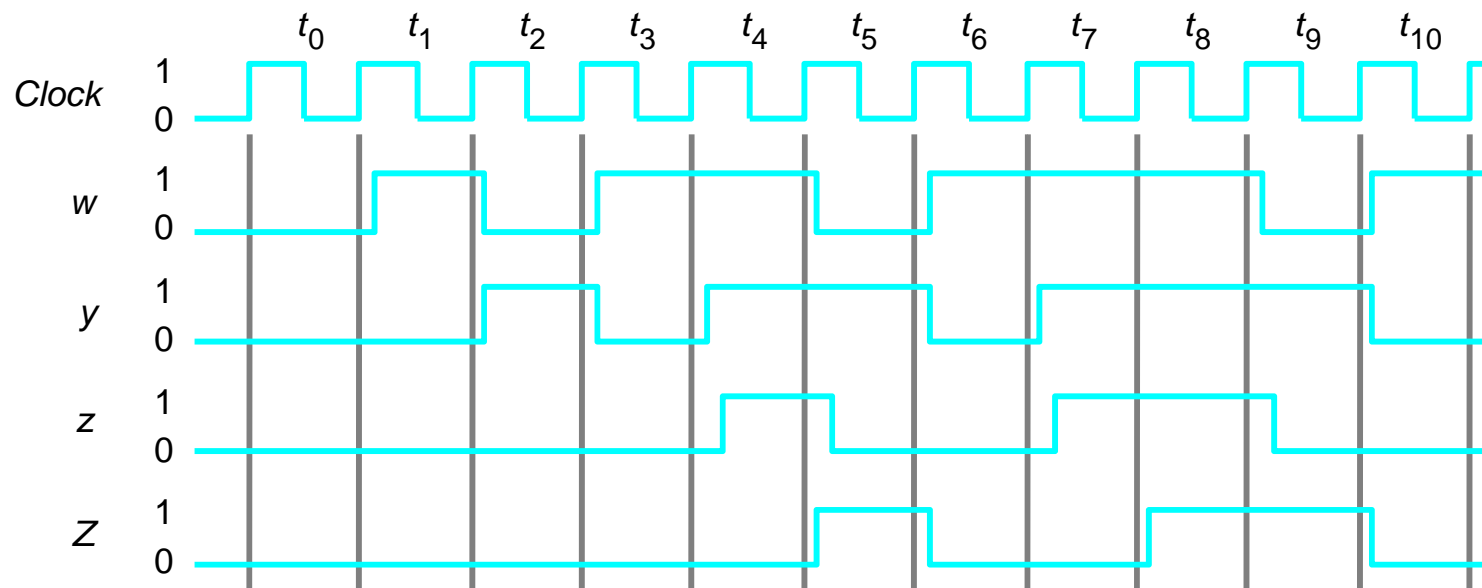
Mealy med utgångsregister

- Nackdelen med Mealy-automaten är att utsignalen kan ändras under hela klockperioden
- Man kan lägga till en register (vipa) på utgången så för att synkronisera utgången med klockflanken



Tidsdiagram med utgångsregister

Med ett utgångsregister så försvinner skillnaden mellan tidsdiagrammen!



Moore vs Mealy

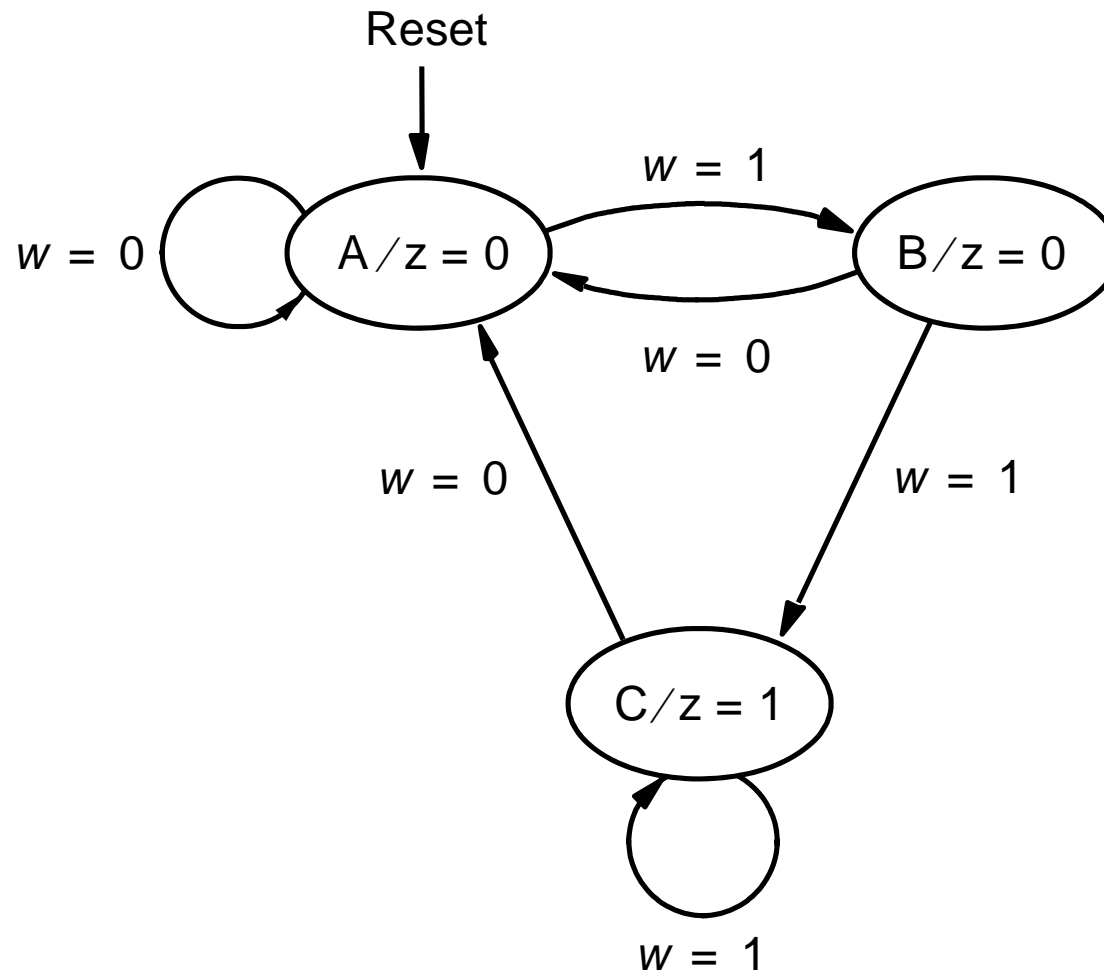
- Moore-automatens utgångsvärden beror bara på det nuvarande tillståndet
- Mealy-automatens utgångsvärden beror på det nuvarande tillståndet och värden på ingångssignalerna
- Mealy-automaten behöver ofta färre tillstånd
- Mealy-automatens utsignaler är *inte synkroniserade* med klockan, varför man ofta lägger till ett utgångsregister

Val av tillståndskodning

Valet av tillståndskodningen kan spela en stor roll för implementeringen eftersom den påverkar logiken för

- Next-state-decoder
- Utgångsavkodare


”två i rad” tillståndsdigram



Tillståndskod = Binärkod

A = 00
B = 01
C = 10
 11

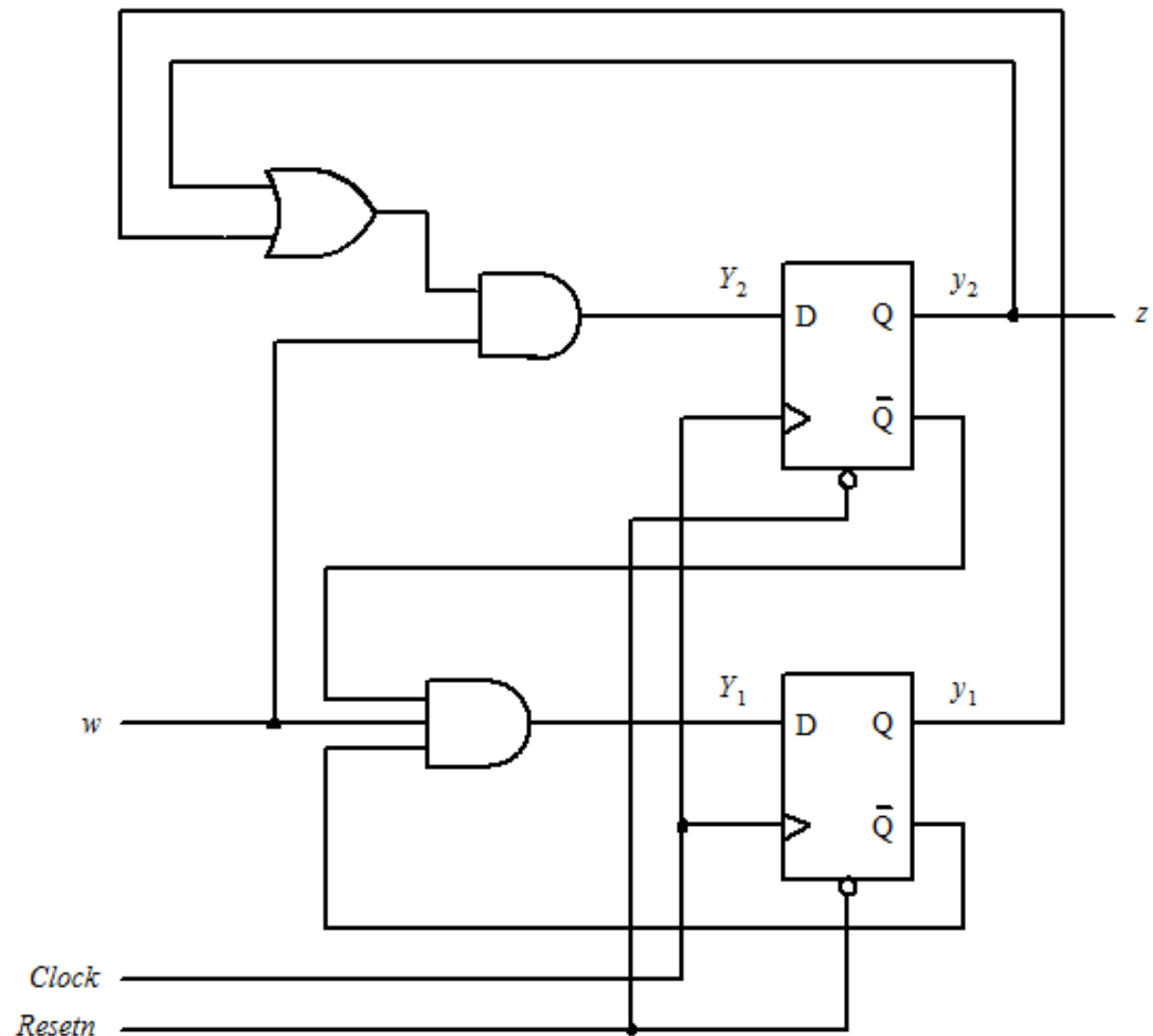
A
 B
 C



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

Realisering (Binärkod)

2 D-vippor
2 AND-grindar
1 OR-grind



Tillståndskod = Graykod

- I Gray-koden ändras bara en bit åt gången, dvs 00, 01, 11, 10
- Gray-koden är **bra för räknare**

A = 00
B = 01
C = 11
 10

A
 B
 C

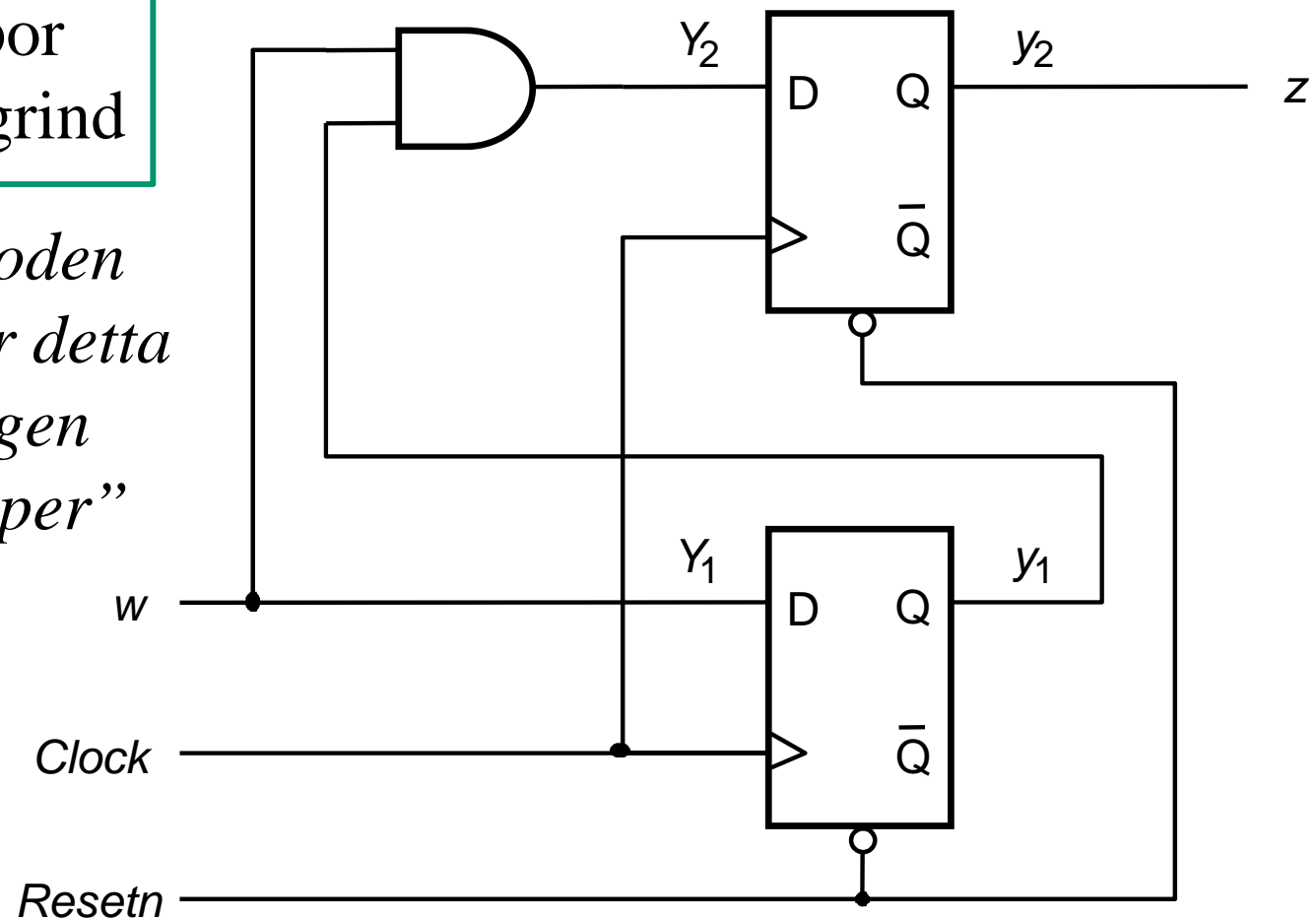


Present state y_2y_1	Next state		Output z
	$w = 0$	$w = 1$	
	Y_2Y_1	Y_2Y_1	
00	00	01	0
01	00	11	0
11	00	11	1
10	<i>dd</i>	<i>dd</i>	<i>d</i>

Realisering (Graykod)

2 D-vippor
1 AND-grind

*Eftersom Graykoden
lyckades bra har detta
sekvensnät tydligen
”räknaregenskaper”*



One-Hot-kodning

- One-hot-kodningen använder en vippa per tillstånd
- För varje tillstånd är en bit ‘hot’ (**1**), alla andra bitar är 0,
dvs 000**1**, 00**1**0, 0**1**00, **1**000
- One-hot kodningen minimerar den kombinatoriska logiken men *ökar* antalet vippor

Vilken kod ska man välja?

- Det finns inte en kod som är den bästa i alla lägen, utan det beror helt på tillståndsdigrammet
- Man kan även ha ‘egna koder’ som passar till konstruktionen, t ex 00, 11, 10, 01