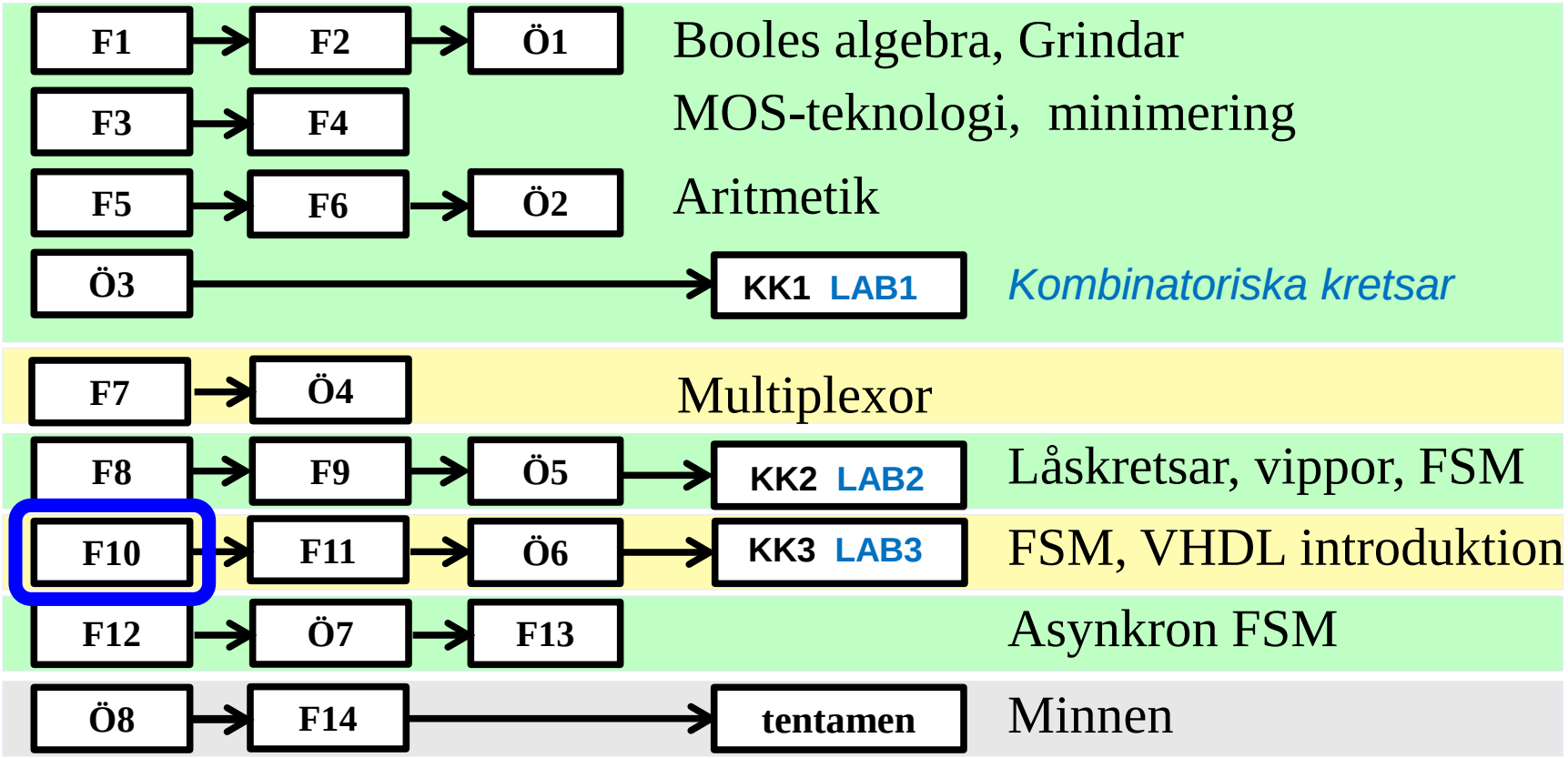
Digital Design IE1204

F10 Tillståndsautomater del II

william@kth.se

IE1204 Digital Design



Föreläsningar och övningar bygger på varandra! Ta alltid igen det Du missat!
Läs på i förväg – delta i undervisningen – arbeta igenom materialet efteråt!

William Sandqvist william@kth.se

Detta har hänt i kursen …

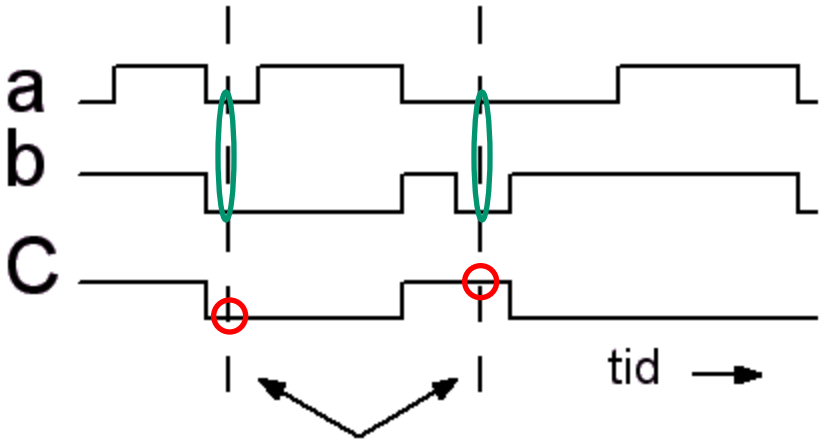
Decimala, hexadecimala, oktala och binära talsystemen
AND OR NOT EXOR EXNOR Sanningstabell, mintermer Maxtermer PS-form
Booles algebra SP-form deMorgans lag Bubbelgrindar Fullständig logik
NAND NOR CMOS grindar, standardkretsar Minimering med Karnaugh-diagram 2, 3, 4, 5, 6 variabler
Registeraritmetik tvåkomplementrepresentation av binära tal
Additionskretsar Multiplikationskrets Divisionskrets
Multiplexorer och Shannon dekomposition Dekoder/Demultiplexor Enkoder
Prioritetsenkoder Kodomvandlare
VHDL introduktion
Vippor och Låskretsar SR-latch D-latch D-vippa JK-vippa T-vippa Räknare
Skiftregister Vippor i VHDL

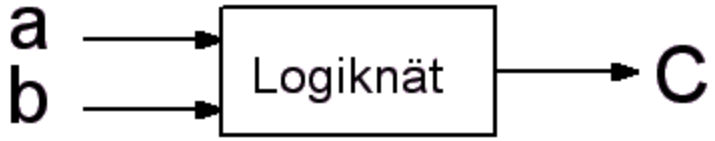
Moore-automat Mealy-automat Tillståndskod

William Sandqvist william@kth.se

Sekvensnät

Som Du kommer ihåg …





William Sandqvist william@kth.se

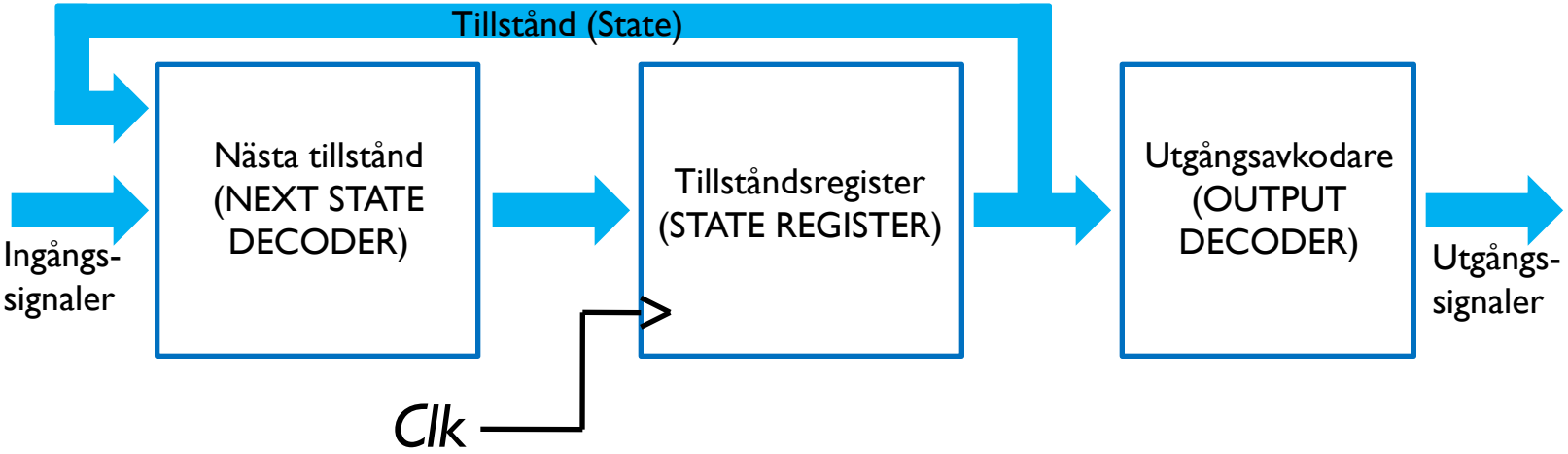
Designmetodik

Grundläggande designmetodik för tillståndsmaskiner.

1. Analysera specifikationen för kretsen
2. Skapa tillståndsdiagram
3. Ställ upp tillståndstabellen
4. Minimera tillståndstabellen
5. Tilldela koder för tillstånden
6. Välj typ av vippor
7. Realisera kretsen mha Karnaugh-diagram.

William Sandqvist william@kth.se

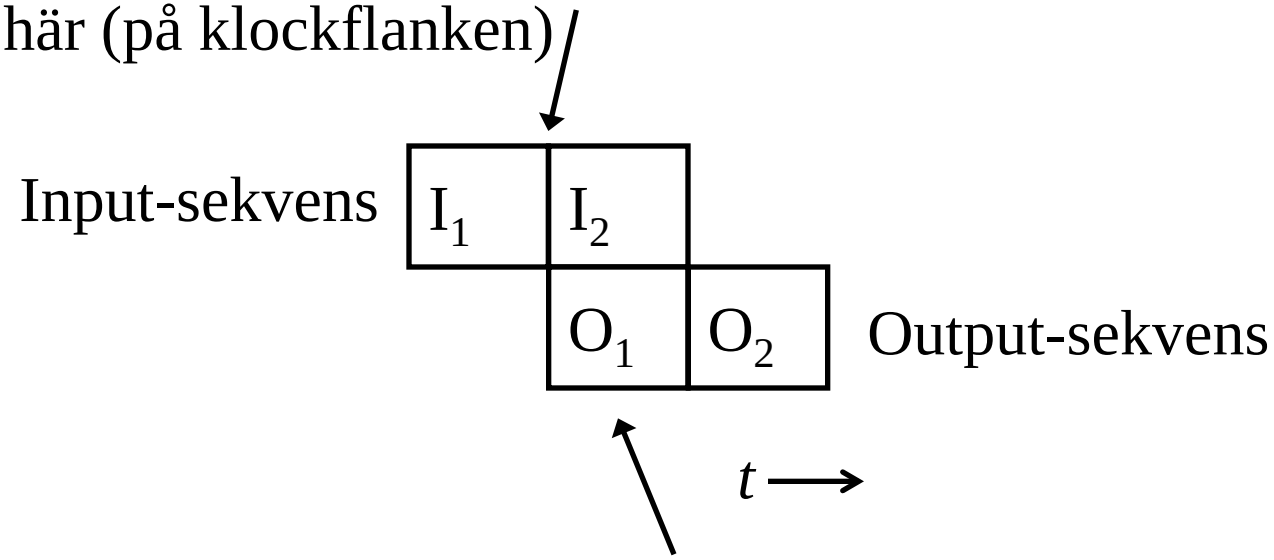
Moore-automat



För Moore-automaten beror utsignalerna på det inre tillståndet.

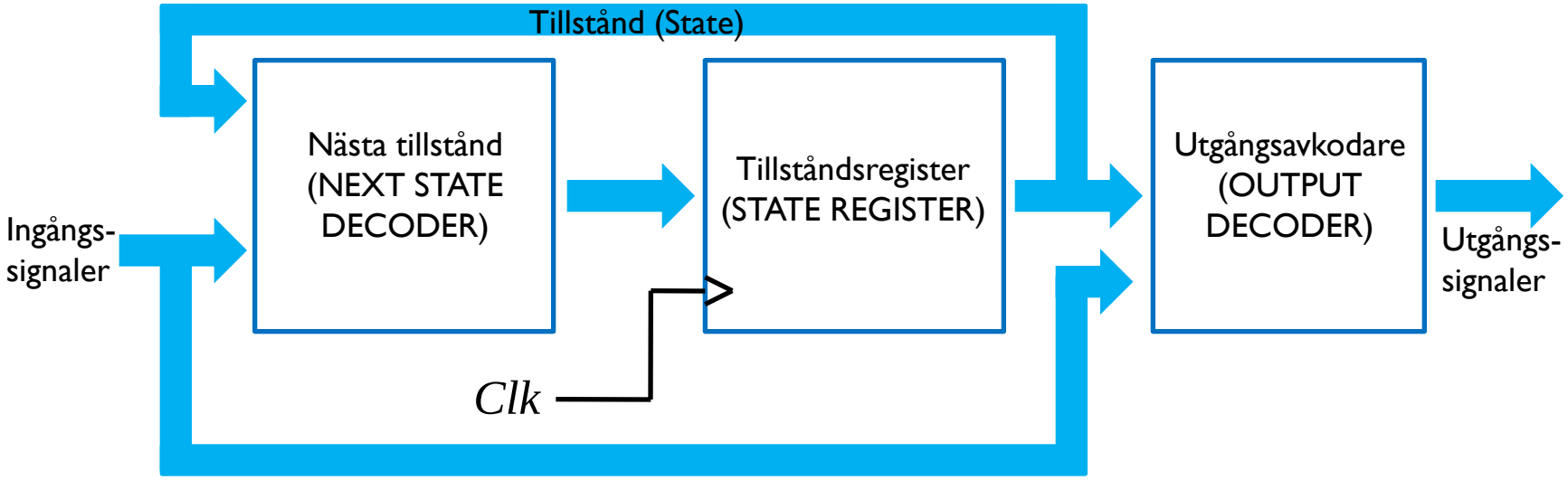
Moore: input och output

Tillståndet (State) ändras
här (på klockflanken)



Output syns ***efter*** att tillståndet (state) har ändrats

Mealy-automat

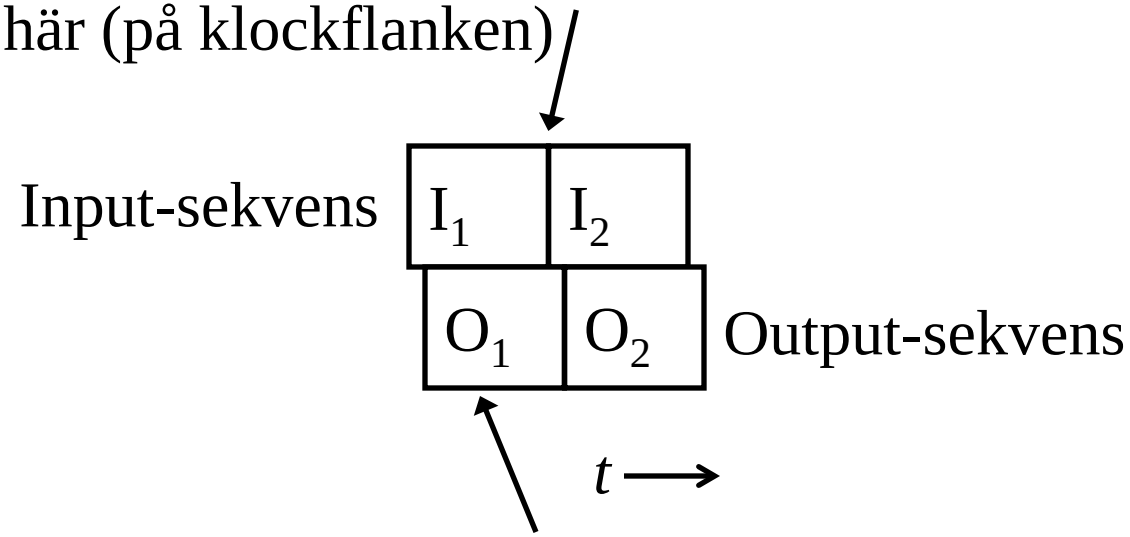


I en **Mealy**-Automat beror utgångssignalerna både på nuvarande tillstånd ***och*** ingångarna

William Sandqvist william@kth.se

Mealy: input och output

Tillståndet (State) ändras
här (på klockflanken)



Output syns ***direkt efter*** att input har ändrats

Överblivna tillstånd?

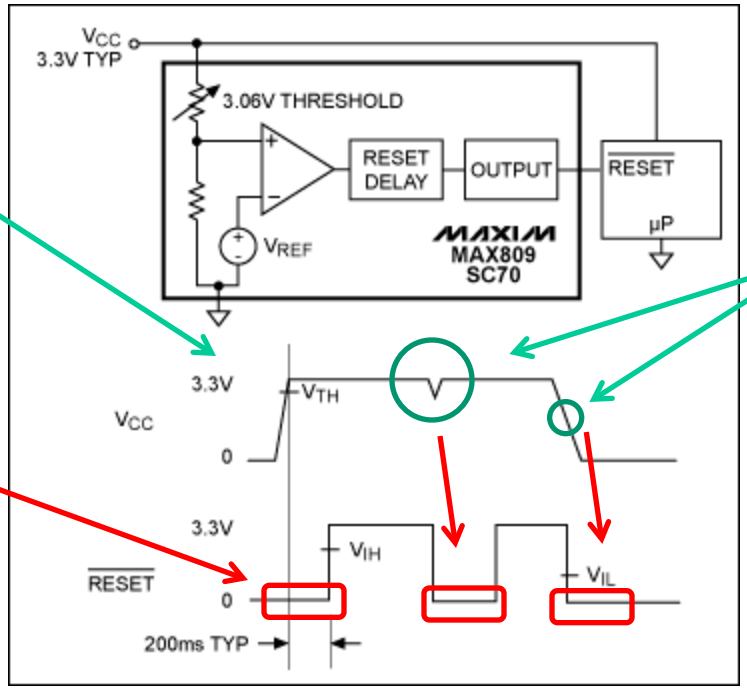
• Ibland får man några states ***över*** när man väljer kod.
(Totala antalet states är alltid potenser av 2)

• Överblivna states måste tas om hand så att inte statemaskinen låser sig vid uppstart

Ett annat sätt är att man alltid (tex. automatiskt) gör RESET vid uppstart.

(RESET-generator chip)

Matnings-spänning ”på” ger **RESET** i 200 ms



Om matnings-spänningen får problem, eller sjunker under viss nivå, så blir det **RESET**

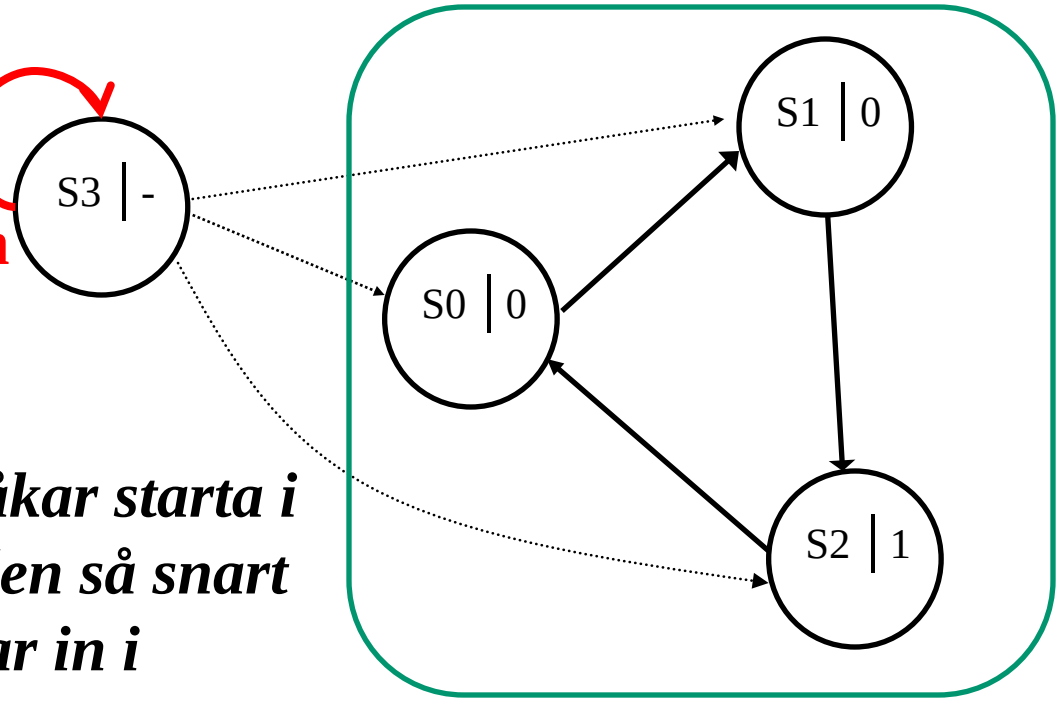
Bättre än att behöva skaffa extra skydd, är att designa förebyggande och från början ”ta hand om” alla tillstånd …

William Sandqvist william@kth.se

Ex. räknare {0,1,2}

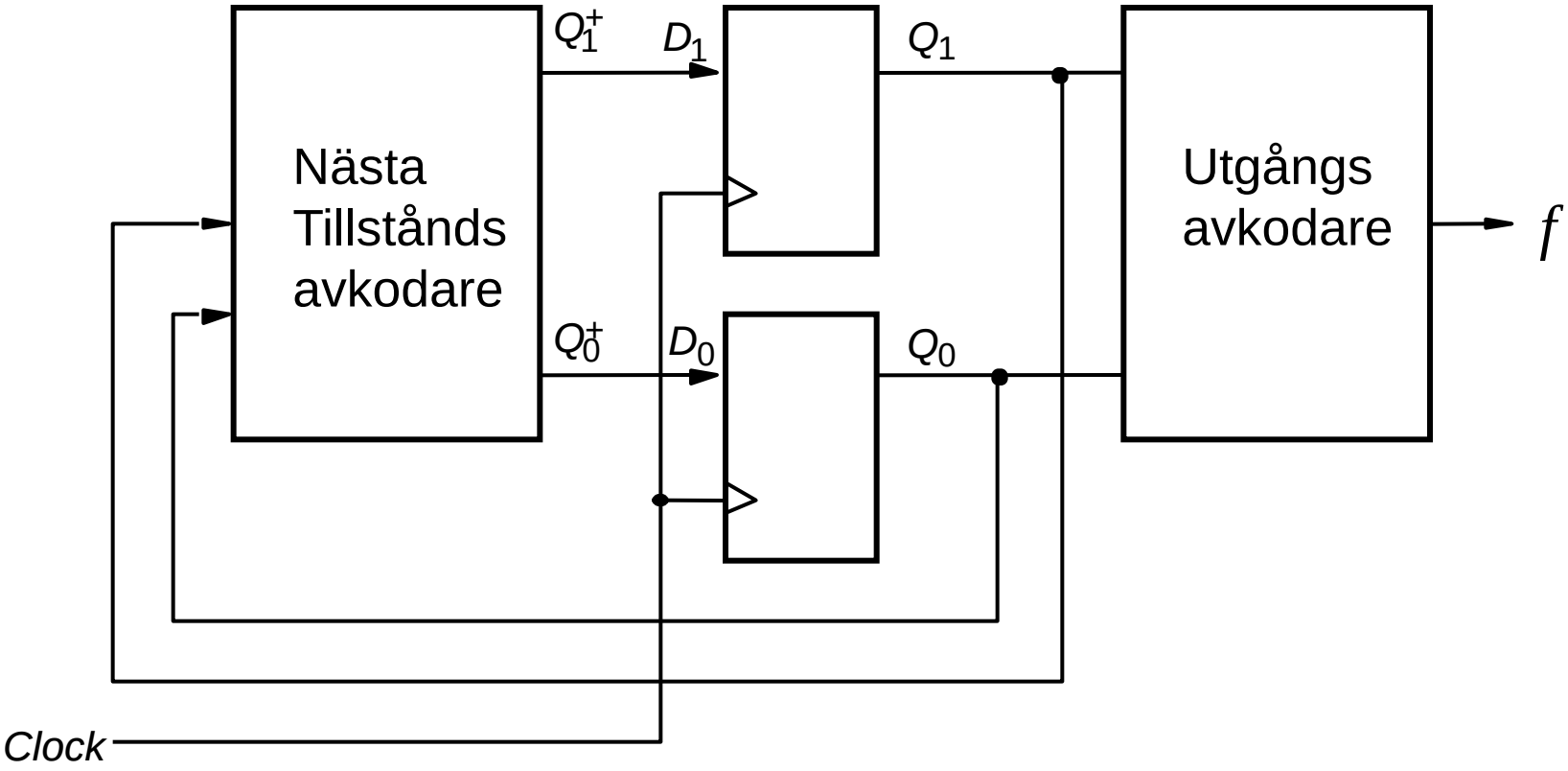
3 tillstånd $\rightarrow$ 2 vippor $\rightarrow$ 2^2 = 4 tillstånd. Ett tillstånd blir över ...

Farlig övergång (automaten låser sig)



Om maskinen råkar starta i S3 så vill vi att den så snart som möjligt hittar in i sekvensen!

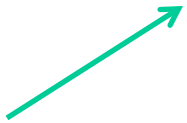
Räknaren som automat



William Sandqvist william@kth.se

Nextstate-funktion

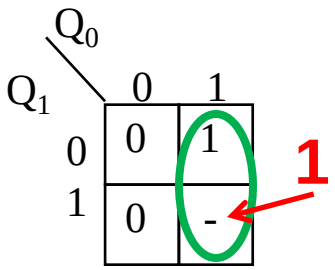
	Nuv. värde	Utsignal	Nästa värde	D-vippa
	Q_1Q_0	f	$Q_1^+Q_0^+$	D_1D_0
S0	0 0	0	0 1	0 1
S1	0 1	0	1 0	1 0
S2	1 0	1	0 0	0 0
	1 1	-	- - (ej 11)	- -

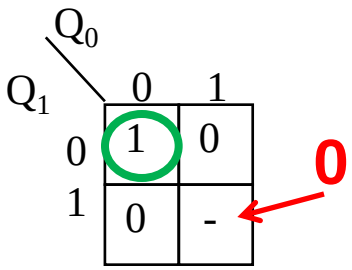


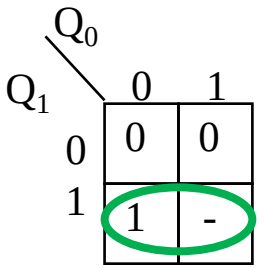
Vi kan specifiera vad som helst här – ***utom*** att ”stanna kvar (dvs. ej 11)!

Karnaughdiagram

	Nuv. värde	Utsignal	Nästa värde	D-vippa
	Q_1Q_0	f	$Q_1^+Q_0^+$	D_1D_0
S0	0 0	0	0 1	0 1
S1	0 1	0	1 0	1 0
S2	1 0	1	0 0	0 0
	1 1	-	- - (ej 11)	- -







$$Q_1^+ = D_1 = Q_0 \qquad Q_0^+ = D_0 = \overline{Q}_1\,\overline{Q}_0 \qquad f = Q_1$$

- **OK, 10 *inte* 11!**

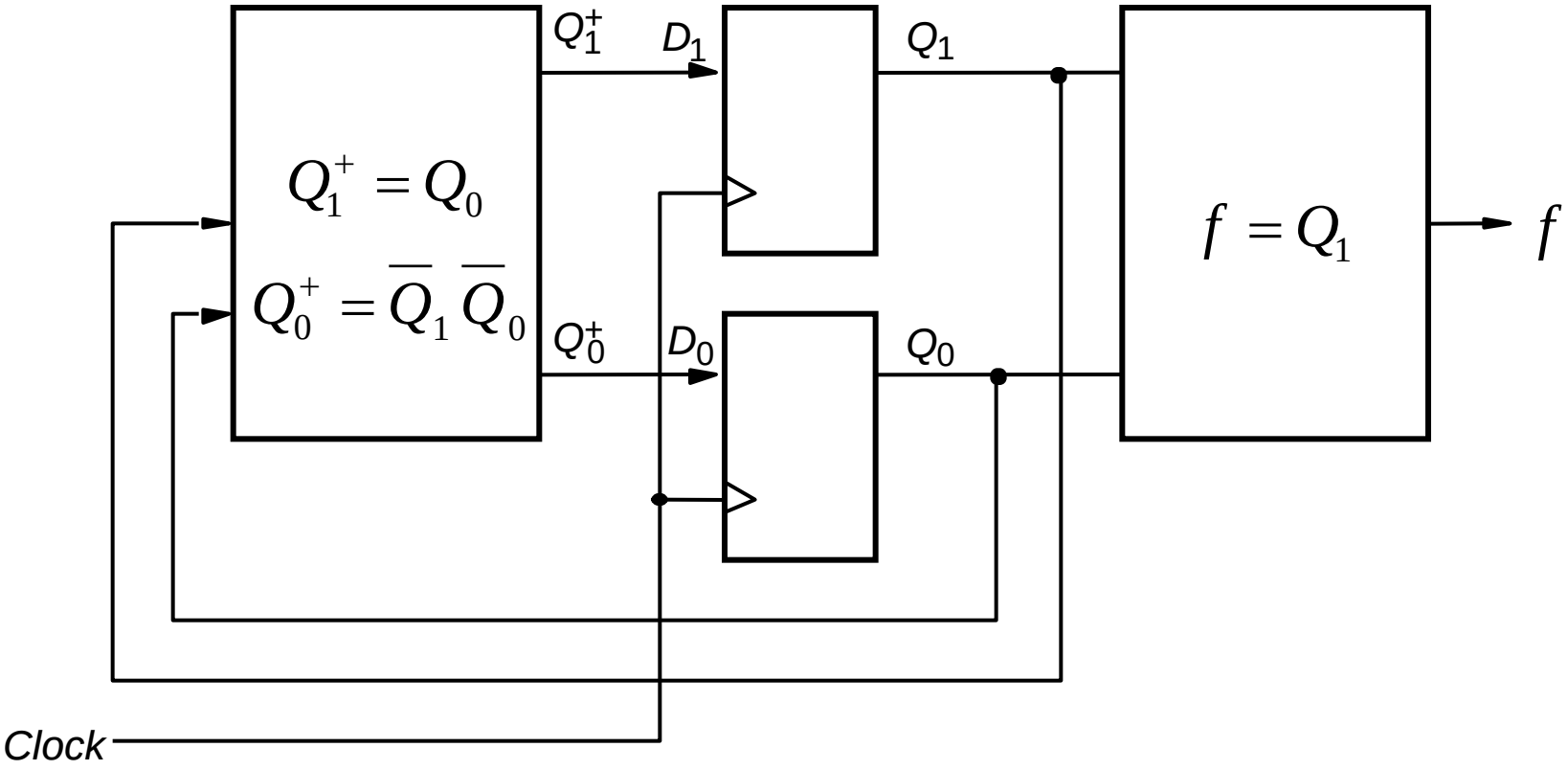
William Sandqvist william@kth.se

Minimerad kodad tillståndstabell

Nuv. värde	Utsignal	Nästa värde	D-vippa
Q_1Q_0	f	$Q_1^+Q_0^+$	D_1D_0
0 0	0	0 1	0 1
0 1	0	1 0	1 0
1 0	1	0 0	0 0
1 1	1	1 0 (ej 11)	1 0

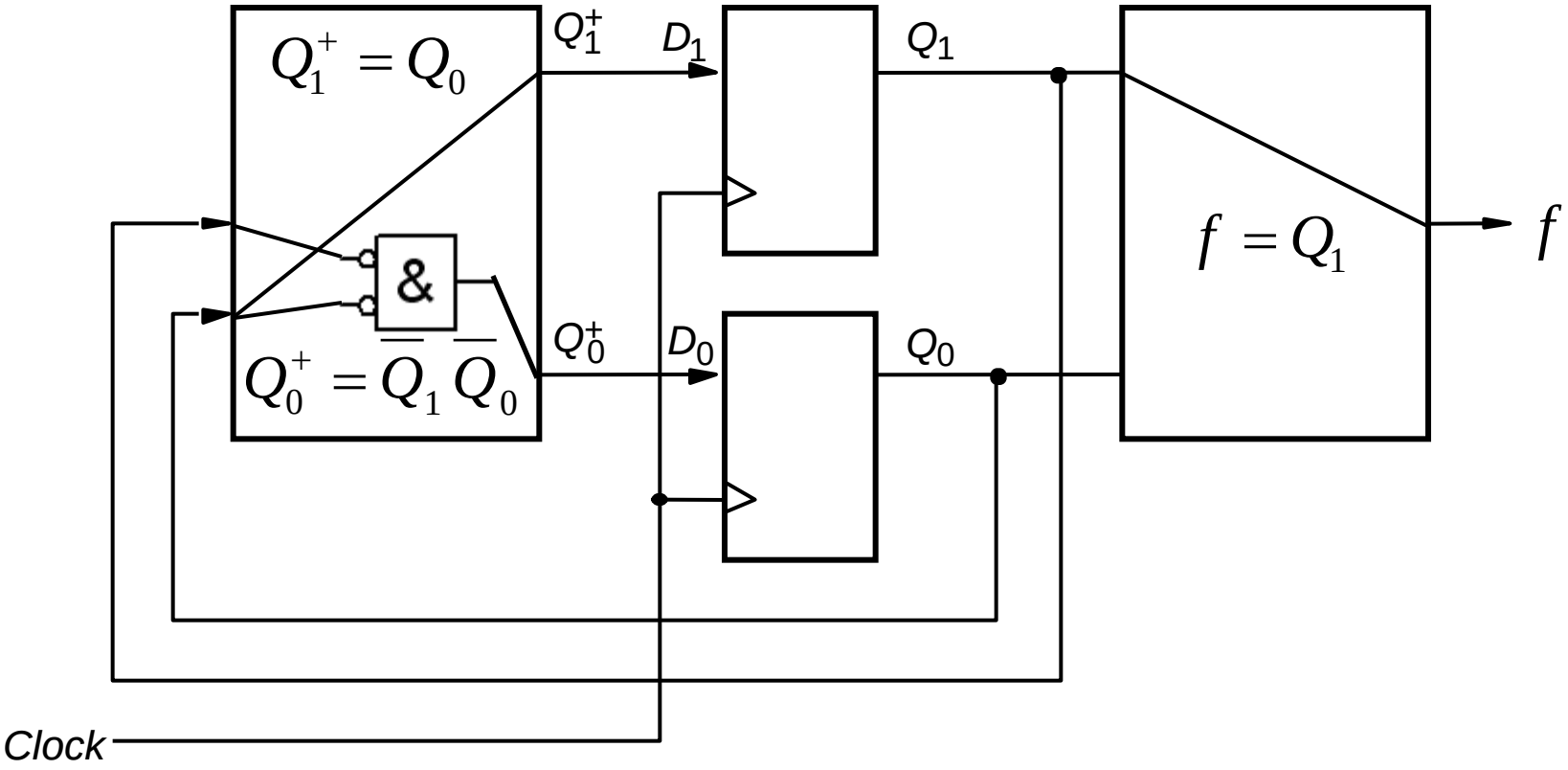
Dvs, det extra tillståndet går in till S2 i huvudsekvensen...

Räknaren



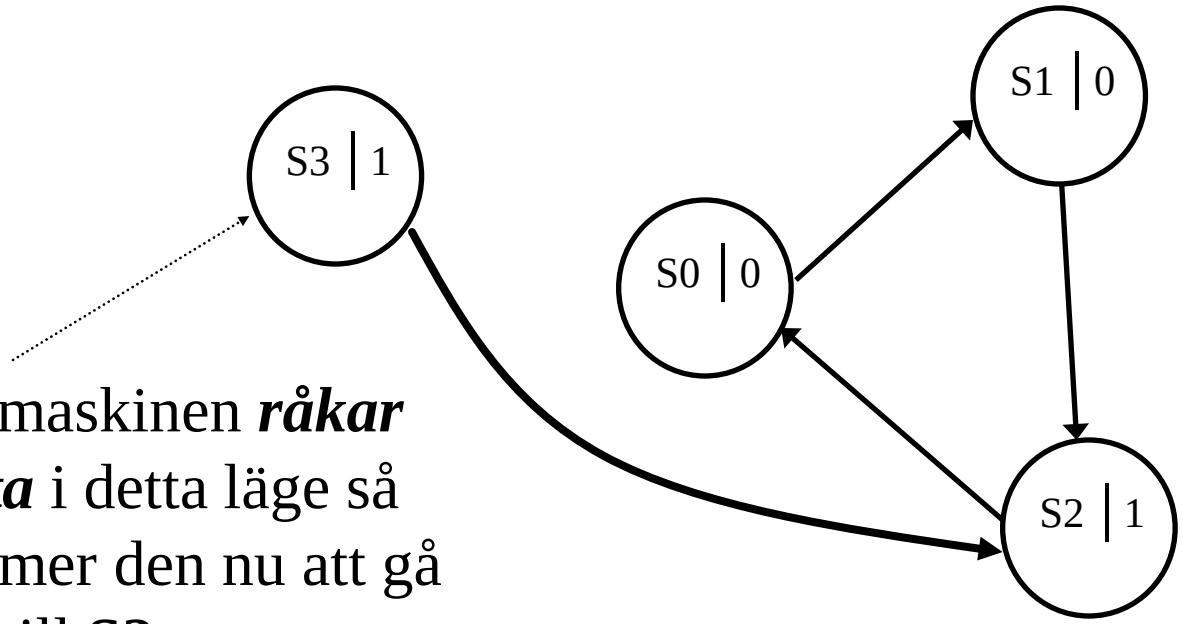
William Sandqvist william@kth.se

Räknaren



William Sandqvist william@kth.se

Slutgiltigt tillståndsdiagram

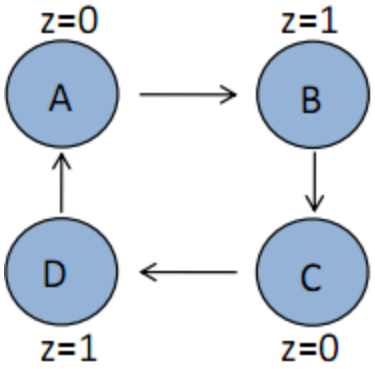


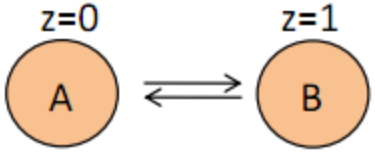
Om maskinen ***råkar starta*** i detta läge så kommer den nu att gå rakt till **S2**.

William Sandqvist william@kth.se

Tillståndsminimering

När man konstruerar komplexa tillstånds-maskiner så kan det lätt hända att det finns **ekvivalenta** och därmed redundanta tillstånd ***som kan tas bort*** för att få en effektivare implementering.



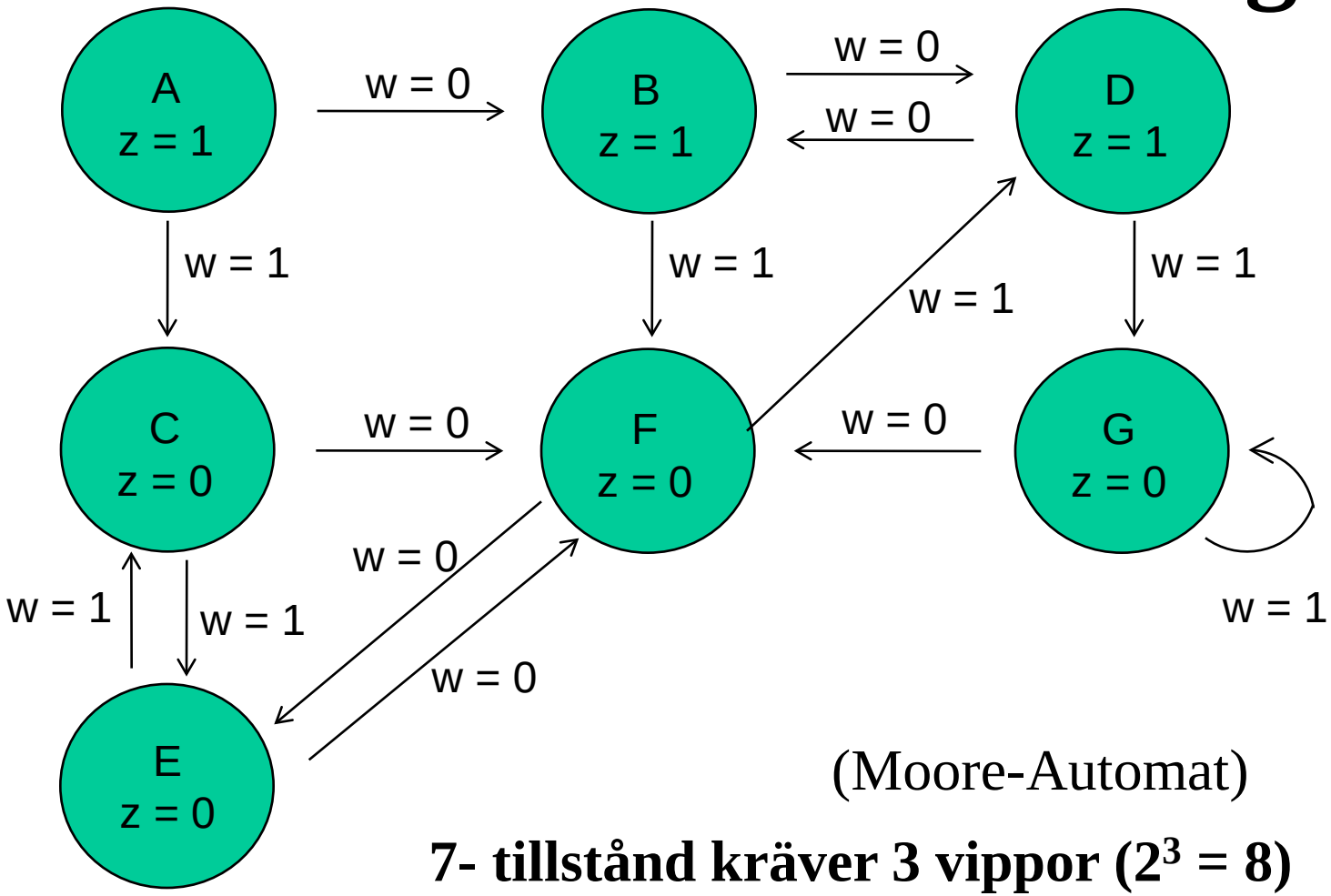


Minimeringsmetod

- Följande exempel illustrerar en manuell minimeringsmetod
  - syftet är att förklara begreppet tillståndsminimering

- Observera att CAD-syntesverktyg använder andra (effektivare) algoritmer

Ex. Tillståndsminimering



(Moore-Automat)

7- tillstånd kräver 3 vippor ($2^3 = 8$)

William Sandqvist william@kth.se

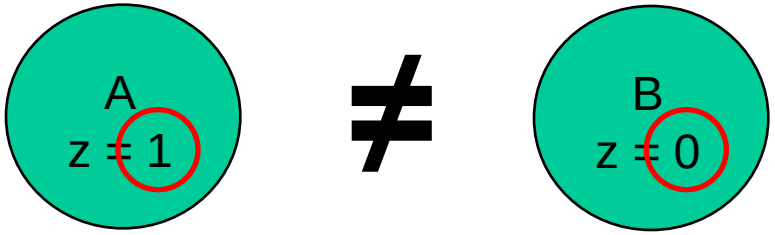
Inte ekvivalenta tillstånd

Det är mycket enklare att skilja ut tillstånd som absolut* inte *kan vara ekvivalenta än att direkt leta reda på ekvivalenta tillstånd …

Minimeringens grundidé

Två tillstånd är *inte* ekvivalenta om de har olika utgångsvärden, dvs. om:

1. **de har olika utgångsvärden**

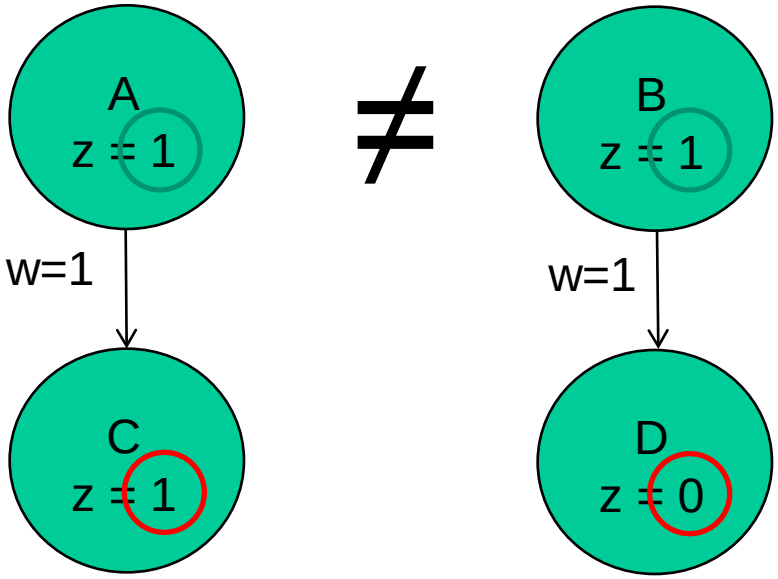


William Sandqvist william@kth.se

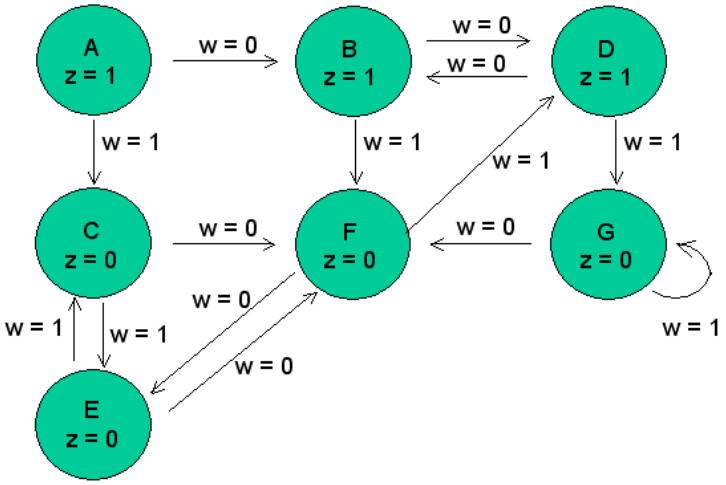
Minimeringens grundidé

Två tillstånd är *inte* ekvivalenta om de har lika utgångsvärden, men

2. **om *någon* av tillståndsövergångarna leder till *olika* efterföljande utgångsvärden**



Tillståndstabell

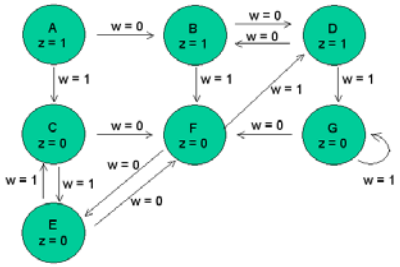


Ursprungligt tillståndsdiagram

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Ursprunglig tillståndstabell

William Sandqvist william@kth.se



Minimera tillstånd

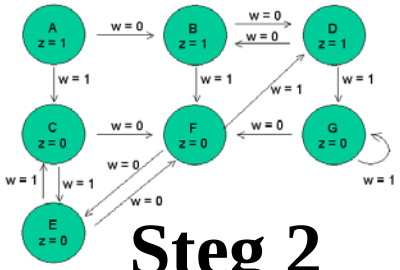
Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Partitioner. Grupper av tillstånd.

Start

$P_{1.}$ Från början utgör ***alla*** tillstånd ett enda block,

$\mathbf{P_1}$ = (ABCDEFG)



Minimera tillstånd

Present state	Next state $w = 0$	Next state $w = 1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

$\mathbf{P_1}$ = (ABDCDEFG)

Steg 2

Gruppera nu tillstånden i grupper efter samma utsignal

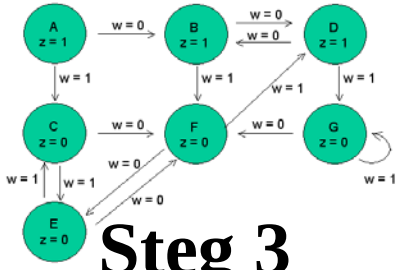
Vilka tillstånd har samma utsignal?

- ABD har utsignalen $z = 1$
- CEFG har utsignalen $z = 0$

$\mathbf{P_2}$ = (ABD)(CEFG)

*Tillstånden A, B, D kan därför **aldrig** vara ekvivalenta med något av tillstånden C, E, F, G eller tvärtom*

William Sandqvist william@kth.se



Minimera tillstånd

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Steg 3

$\mathbf{P_2} = (ABD)(CEFG)$

Vilka **följdtillstånd** har tillstånden?

Block (ABD)

$w = 0$: "0-successor": A → (A**B**D), B → (AB**D**), D → (A**B**D)

alla är övergångar till samma block (ABD)

$w = 1$: "1-successor": A → (**C**EFG), B → (CE**F**G), D → (CEF**G**)

alla är övergångar till samma block (CEFG)

P = (ABD)(CEFG) ingen ändring av blocken

Block (CEFG)

$w = 0$: 0-successor : C → (CE**F**G), E → (CE**F**G), F → (C**E**FG),

G → (CE**F**G) alla är övergångar till samma block (CEFG)

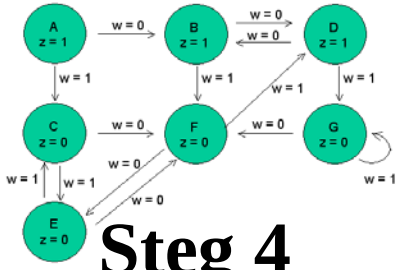
$w = 1$: 1-successor : C → (C**E**FG), E → (**C**EFG), F → (AB**D**),

G → (CEF**G**) C E G går till samma block

F avviker, går till ett annat block

$\mathbf{P_3} = (ABD)(CEG)(F)$

William Sandqvist william@kth.se



Minimera tillstånd

Present state	Next state $w = 0$	Next state $w = 1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Steg 4

$\mathbf{P_3} = (ABD)(CEG)(F)$

Vilka **följdtillstånd** har tillstånden?

Block (ABD)

$w = 0$: "0-successor": A $\rightarrow$ (A**B**D), B $\rightarrow$ (AB**D**), D $\rightarrow$ (A**B**D)

alla är övergångar till samma block ABD

$w = 1$: "1-successor": A $\rightarrow$ (**C**EG), B $\rightarrow$ (**F**), D $\rightarrow$ (CE**G**)

A $\rightarrow$ C, D $\rightarrow$ G är övergångar till samma block

B $\rightarrow$ F avviker, går till ett annat block

$\mathbf{P_4} = (AD)(B)(CEG)(F)$ ny indelning av blocken

Block (CEG)

$w = 0$: "0-successor" : C $\rightarrow$ (**F**), E $\rightarrow$ (**F**), G $\rightarrow$ (**F**)

alla är övergångar till samma block (F)

$w = 1$: "1-successor" : C $\rightarrow$ (C**E**G), E $\rightarrow$ (C**E**G), G $\rightarrow$ (CE**G**)

alla är övergångar till samma block (CEG)

$\mathbf{P_4} = (AD)(B)(CEG)(F)$ ingen ändring av blocken

William Sandqvist william@kth.se

Minimerat

Nästa partition $\mathbf{P_5}$ blir densamma som $\mathbf{P_4}$. Processen är därför klar. AD respektive CEG är **ekvivalenta.**

A’ blir en ny beteckning för AD, C’ blir ny beteckning för CEG.

$$\mathbf{P_4} = (AD)(B)(CEG)(F) = (A')(B)(C')(F)$$

William Sandqvist william@kth.se

Minimerad tillståndstabell

$\mathbf{P_4}$ = (AD)(B)(CEG)(F) = (A')(B)(C')(F)

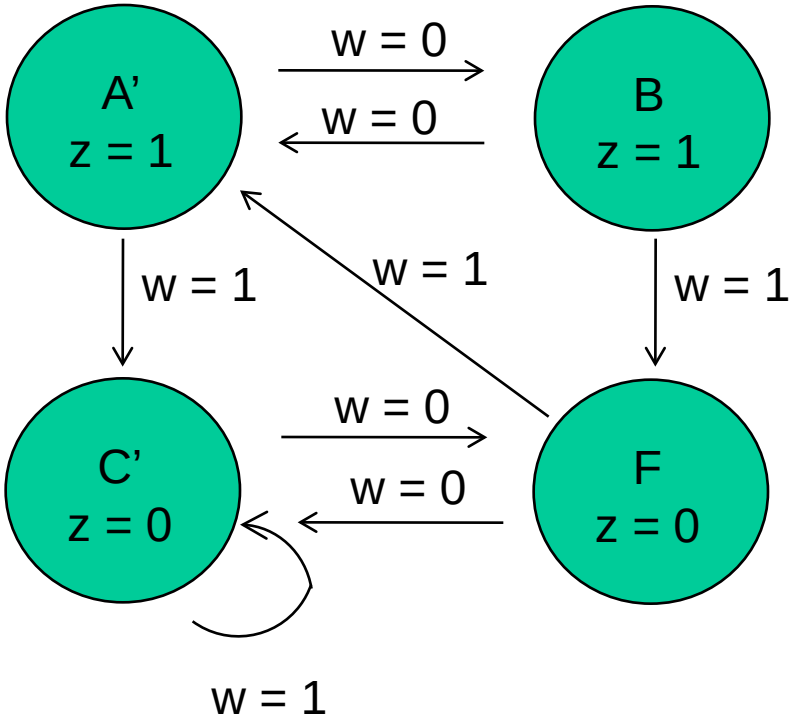
A' är ny beteckning för AD, C' är ny för CEG.

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Present state	Nextstate		Output
	$w = 0$	$w = 1$	z
A'	B	C'	1
B	A'	F	1
C'	F	C'	0
F	C'	A'	0

Minimerat tillståndsdiagram

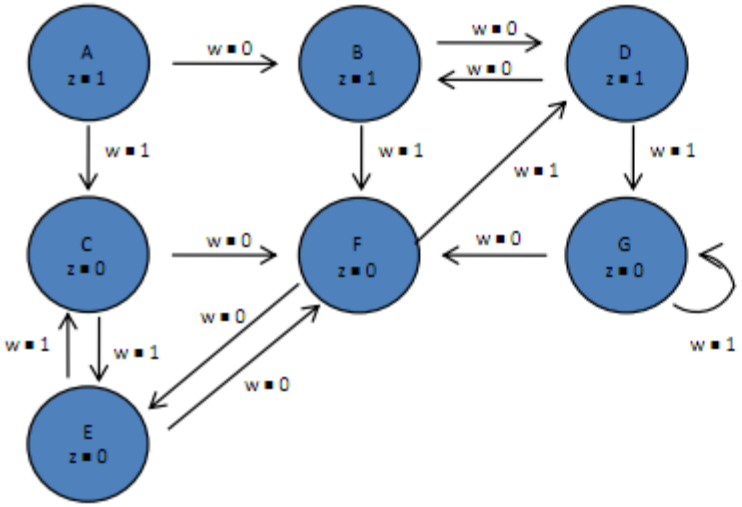
Present state	Nextstate		Output z
	w = 0	w = 1	
A’	B	C’	1
B	A’	F	1
C’	F	C’	0
F	C’	A’	0



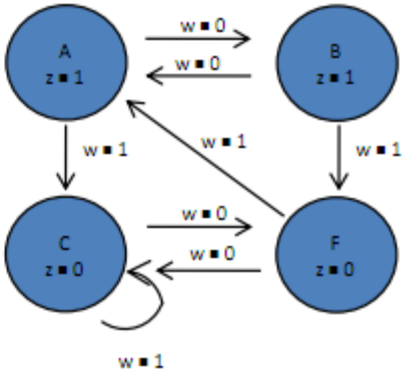
4 tillstånd kräver 2 vippor ($2^2 = 4$).

William Sandqvist william@kth.se

Jämförelse



Före minimering



Efter minimering

William Sandqvist william@kth.se

Värdet av tillståndsminimering?

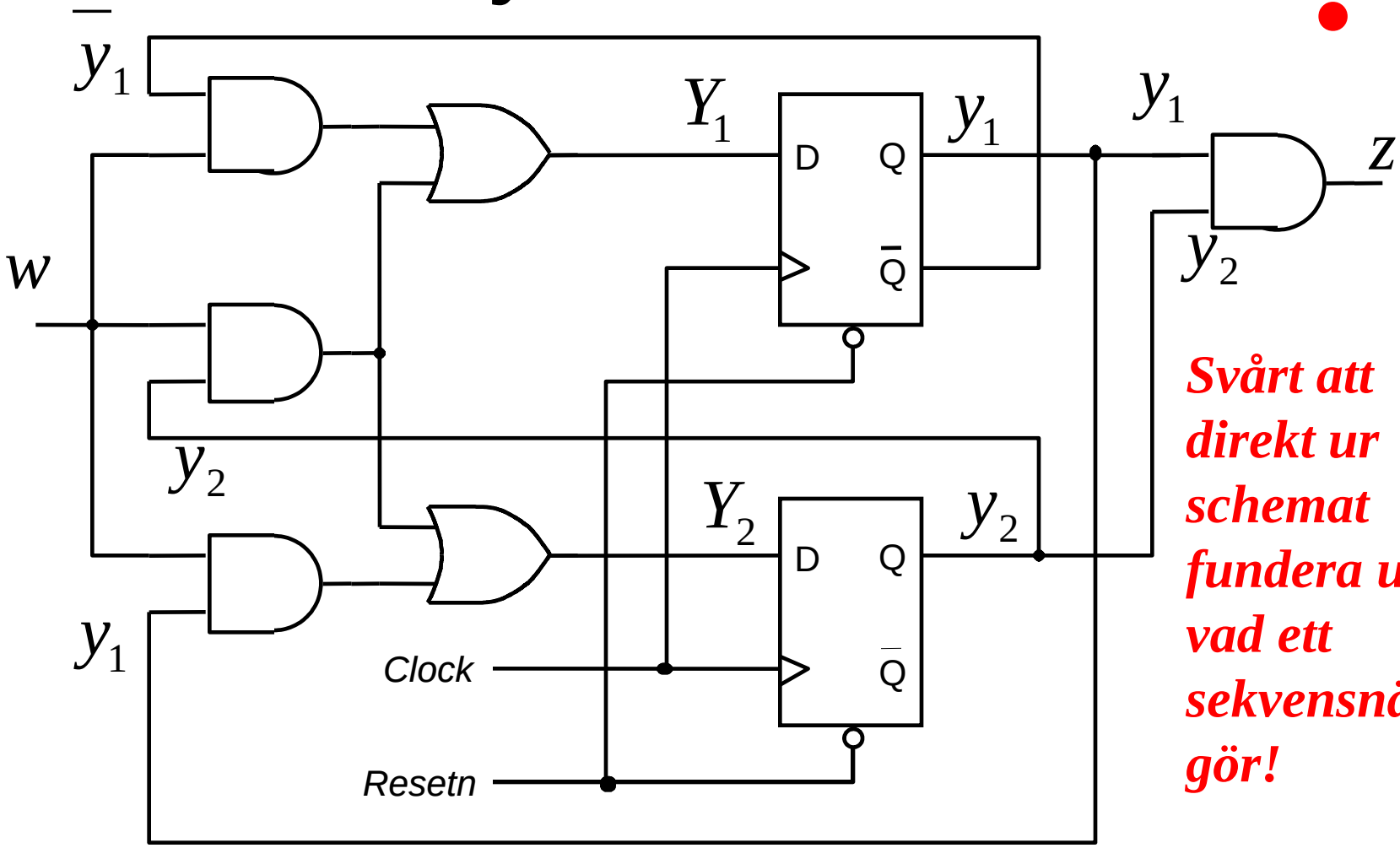
- **Det är *inte* säkert att färre tillstånd leder till ett *enklare* nät!**

Fördelen med tillståndsminimering ligger i stället i att det blir enklare att skapa det ursprungliga tillståndsdiagrammet när man inte behöver anstränga sig för att det dessutom ska bli minimalt från början!

CAD-verktygen minimerar sedan det ursprungliga tillståndsdiagrammet till ett slutgiltigt.

William Sandqvist william@kth.se

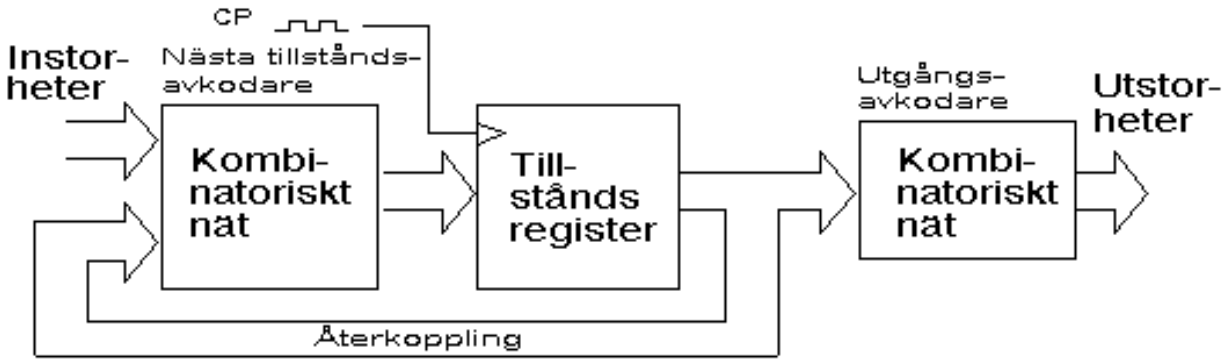
Analys av sekvensnät ?

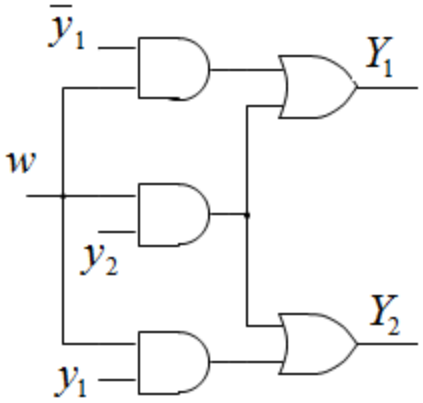


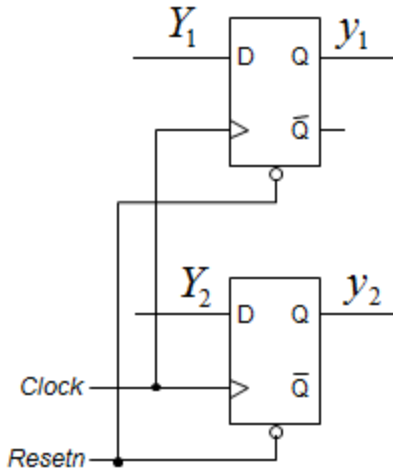
Svårt att direkt ur schemat fundera ut vad ett sekvensnät gör!

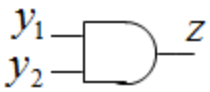
William Sandqvist william@kth.se

Tänk Moore-automat!



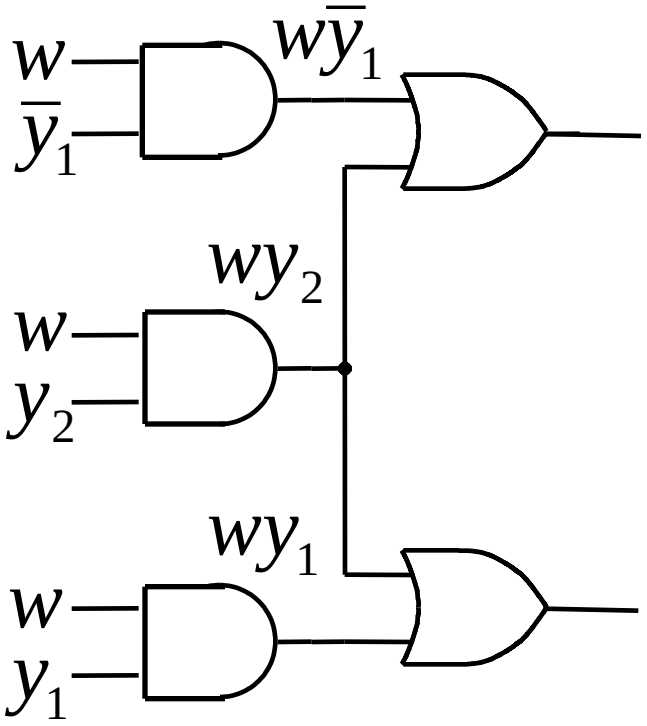






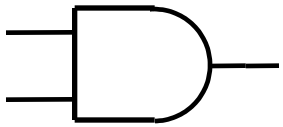
William Sandqvist william@kth.se

Analysera grindnäten



$$Y_1 = w\bar{y}_1 + wy_2$$

$$z = y_1 \cdot y_2$$

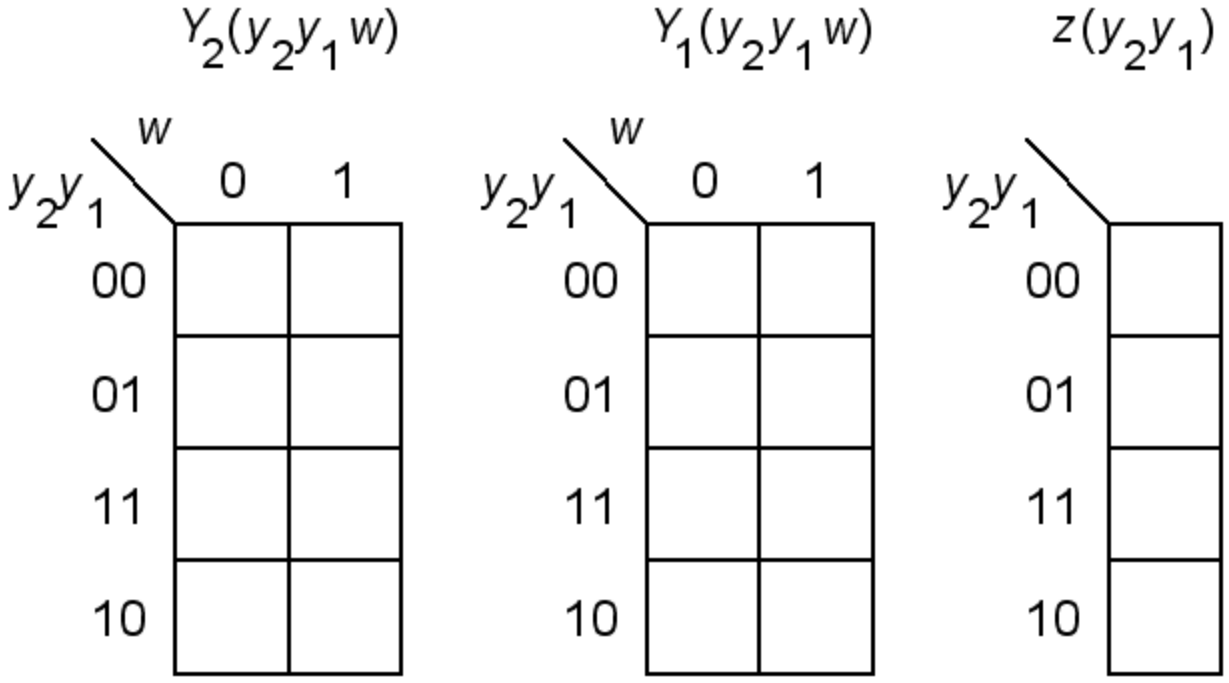


$$Y_2 = wy_1 + wy_2$$

William Sandqvist william@kth.se

Fyll i Karnaughdiagram

Kan Du fylla i Karnaughdiagrammen med funktionerna?

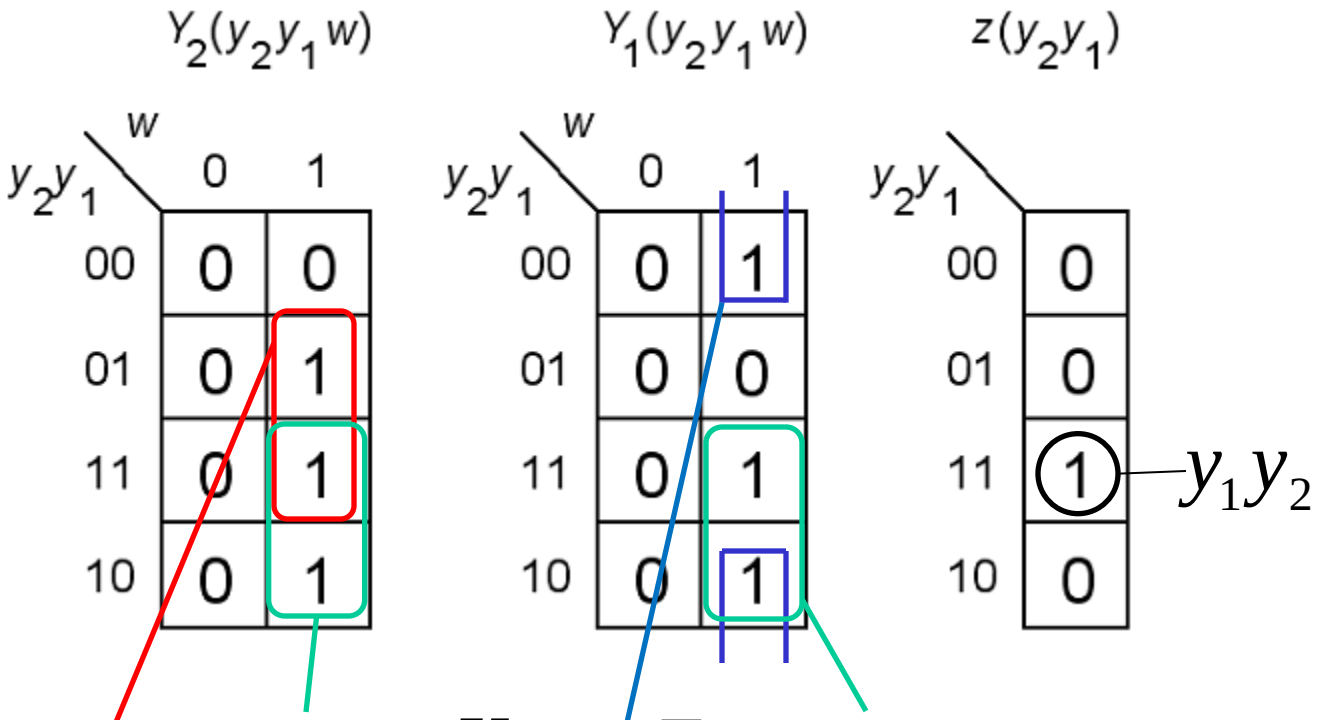


$$Y_2 = wy_1 + wy_2 \quad Y_1 = w\bar{y}_1 + wy_2 \quad z = y_1 \cdot y_2$$

William Sandqvist william@kth.se

Ifyllda Karnaughdiagram

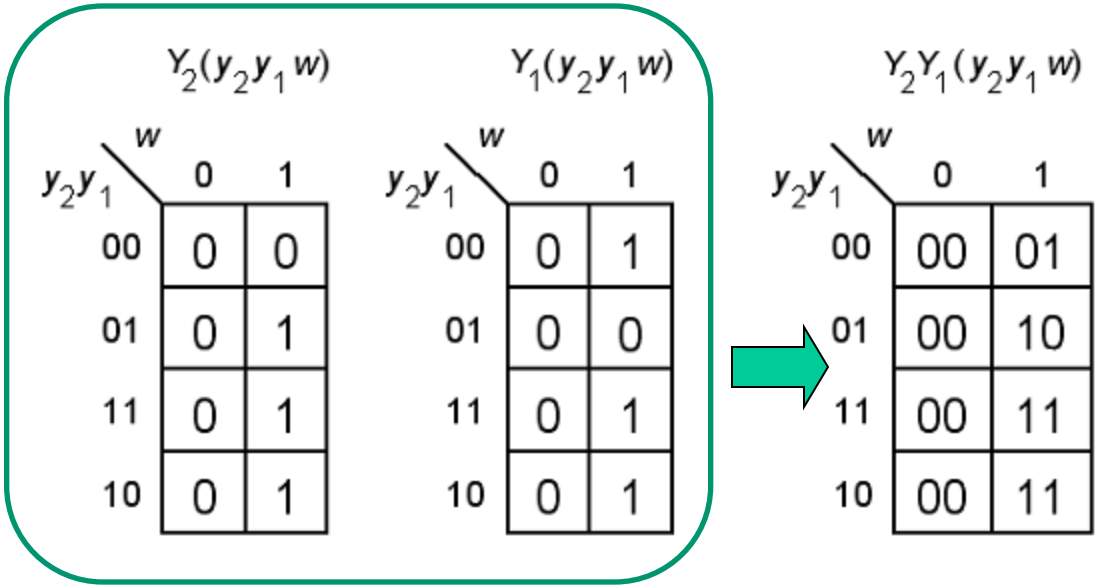
Ifyllda Karnaughdiagram



$$Y_2 = wy_1 + wy_2 \qquad Y_1 = w\bar{y}_1 + wy_2 \qquad z = y_1 \cdot y_2$$

William Sandqvist william@kth.se

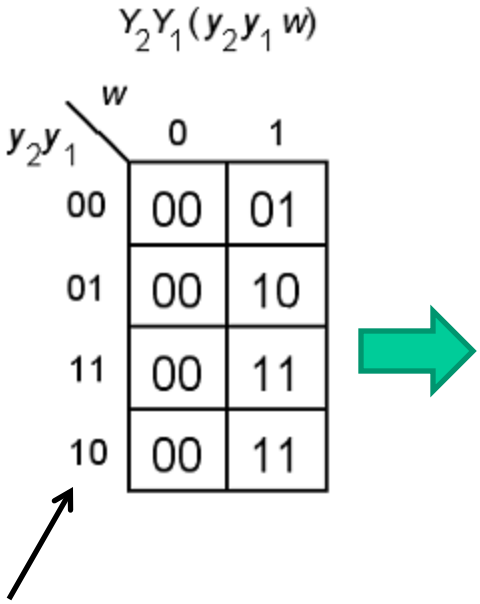
Kodad tillståndstabell



Slå ihop Karnaughdiagrammen till en kodad tillståndstabell

William Sandqvist william@kth.se

Kodad tillståndstabell



Present state y_2y_1	Next State w = 0 Y_2Y_1	Next State w = 1 Y_2Y_1	Output z
0 0	0 0	01	0
0 1	0 0	10	0
1 0	0 0	11	0
1 1	0 0	11	1

Graykod

Binärkod *(BV använder binärkod)*

William Sandqvist william@kth.se

Tillståndstabell

***Kodad** tillståndstabell*

Present state y_2y_1	Next State Y_2Y_1 w = 0	Next State Y_2Y_1 w = 1	Output z
0 0	0 0	01	0
0 1	0 0	10	0
1 0	0 0	11	0
1 1	0 0	11	1

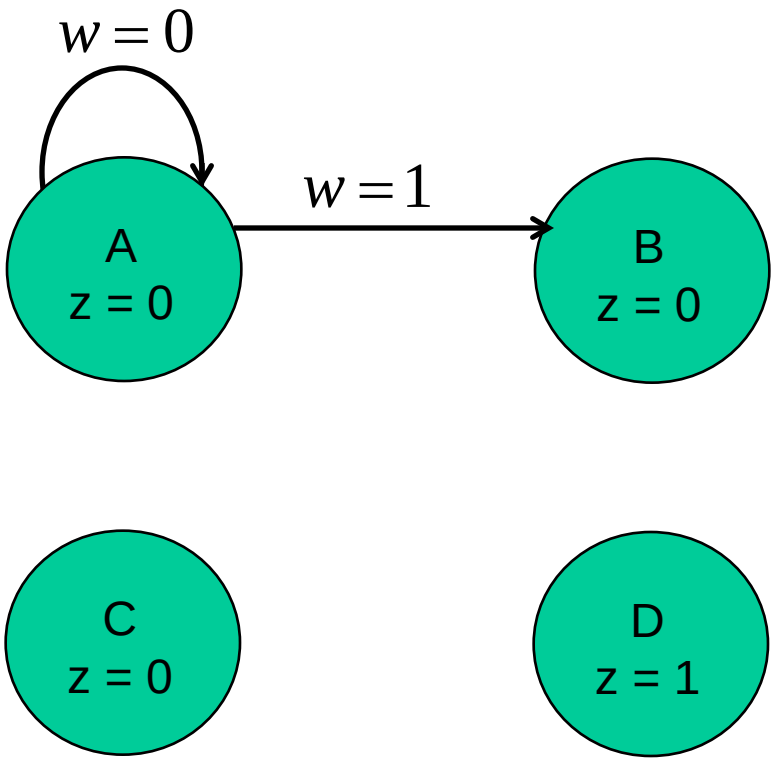
Tillståndstabell

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

Den okodade tillståndstabellen är utgångspunkt om man vill byta till en annan tillståndskodning.

Tillståndsdiagram

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

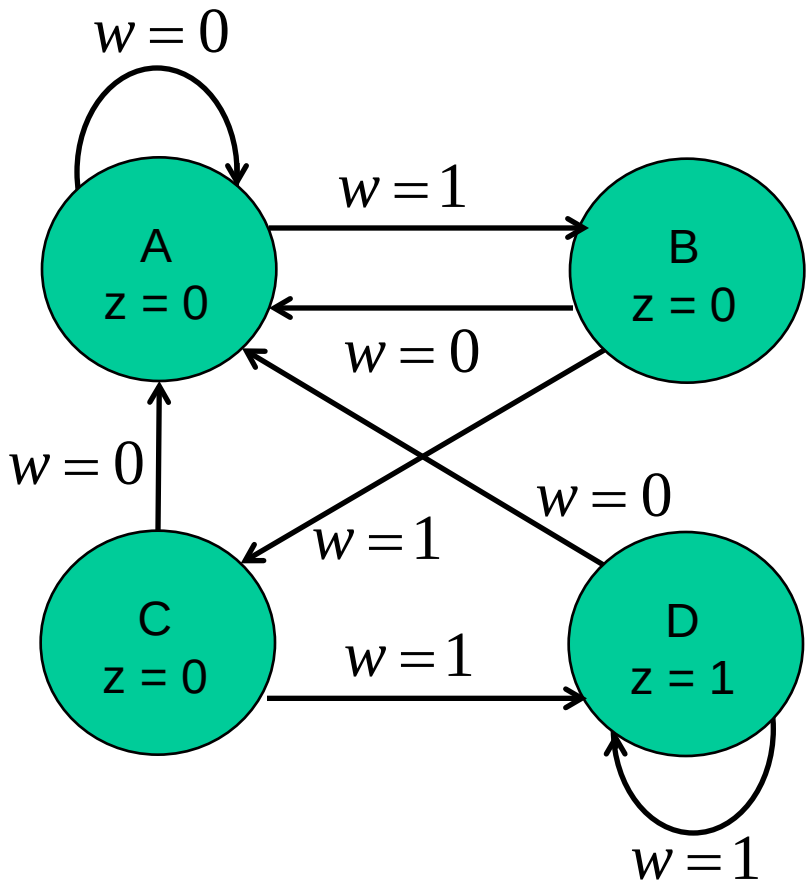


Rita färdigt tillståndsdia-grammet själv.
(På övning 6 löser vi ett liknande problem – kretsen är en ”tre i rad” - krets).

Tillståndsdiagram

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

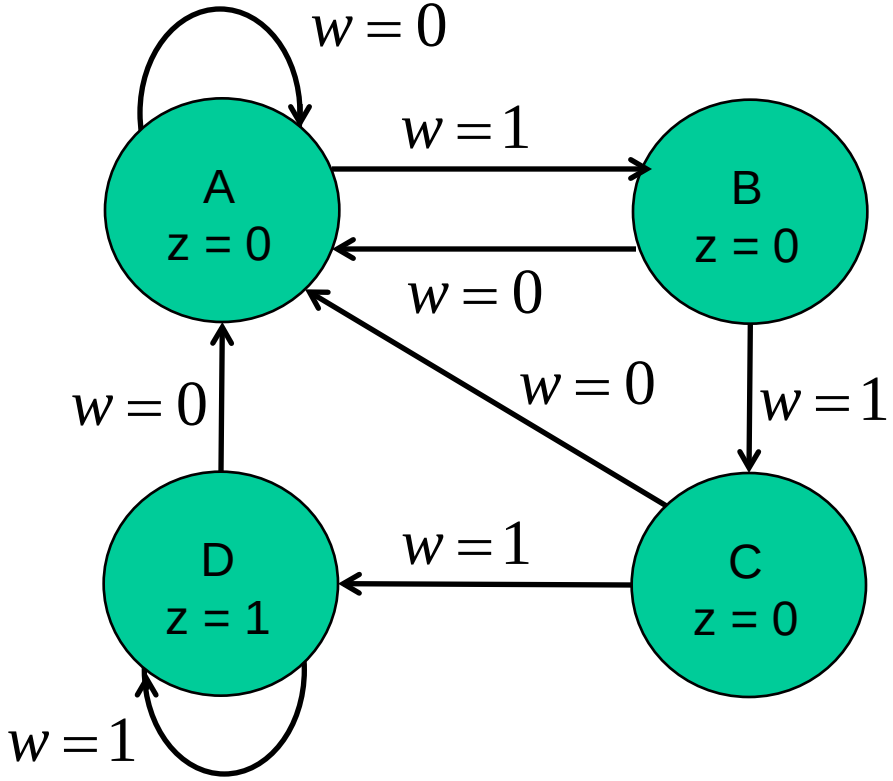
Ibland kan man behöva ändra ordningen på tillstånden för att få ett tydligare diagram



William Sandqvist william@kth.se

Tillståndsdiagram

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



C och D har bytt plats – snyggare, inga korsande tillståndspilar

- *Att kräva samma insignal ”tre gånger i rad” är en ofta använd säkerhetsåtgärd.*

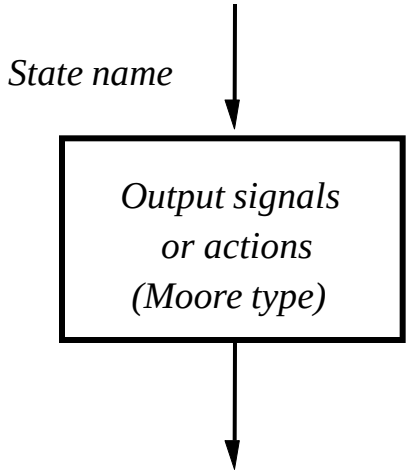
William Sandqvist william@kth.se

ASM-charts

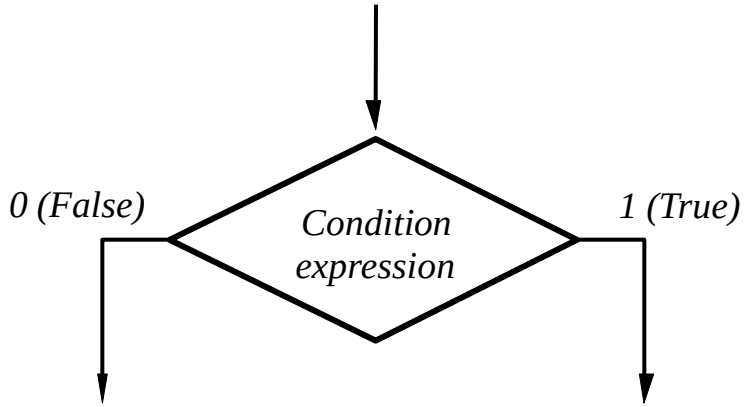
För att beskriva större tillståndsmaskiner används ofta ett annat diagram: Algorithmic State Machine (**ASM**) Charts

ASM-chart, tre byggstenar

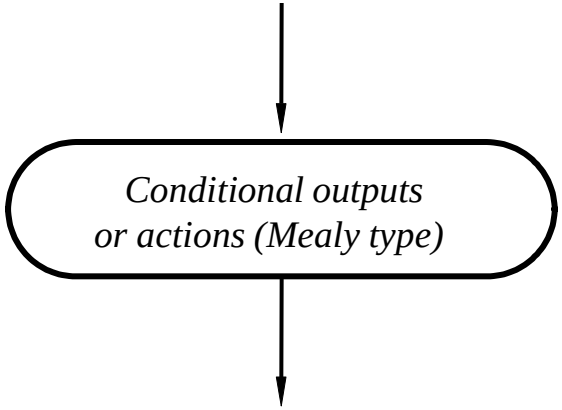
Ett ASM-chart är ett flödesdiagram som byggs upp av tre olika byggstenar.



(a) State box



(b) Decision box



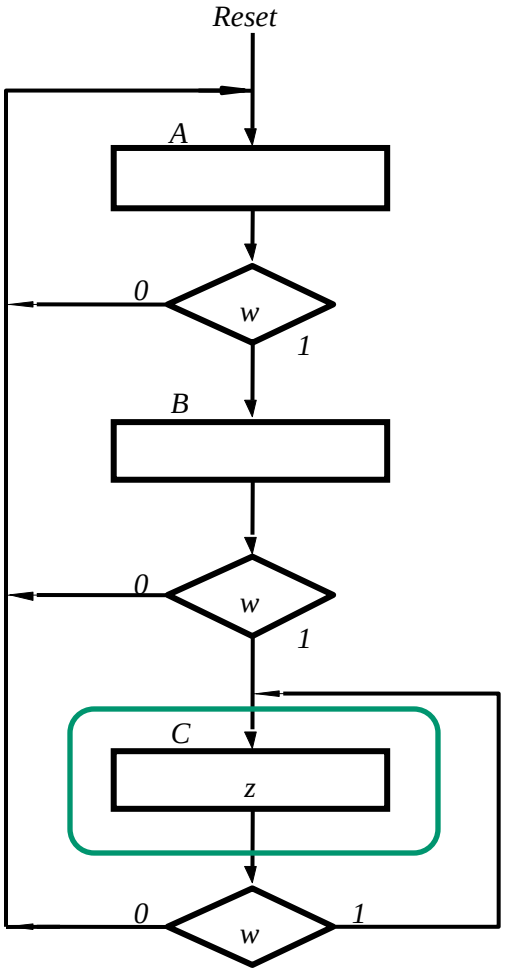
(c) Conditional output box

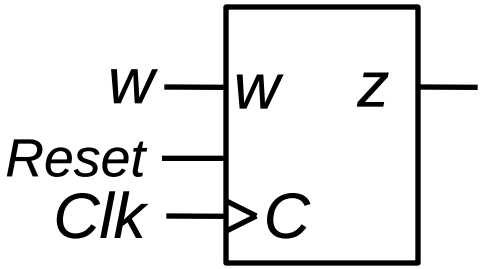
William Sandqvist william@kth.se

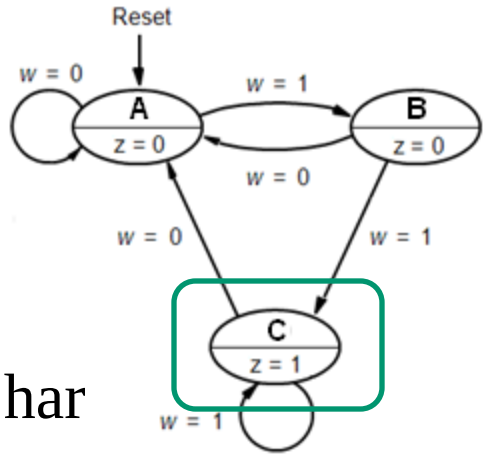
ASM-charts

- **Tillståndslåda (State Box)**
  - **Representerar ett tillstånd i ett FSM**
    - **utgångsvärden för tillståndet anges här (Moore-outputs)**
- **Beslutslåda (Decision Box)**
  - **Beroende på värden på insignaler bestäms övergången till nästa tillstånd**
- **Villkorlig utgångslåda (Conditional outputs)**
  - **Här anges värden av utgångarna vid en tillståndsövergång (Mealy-outputs)**

”två i rad” Moore



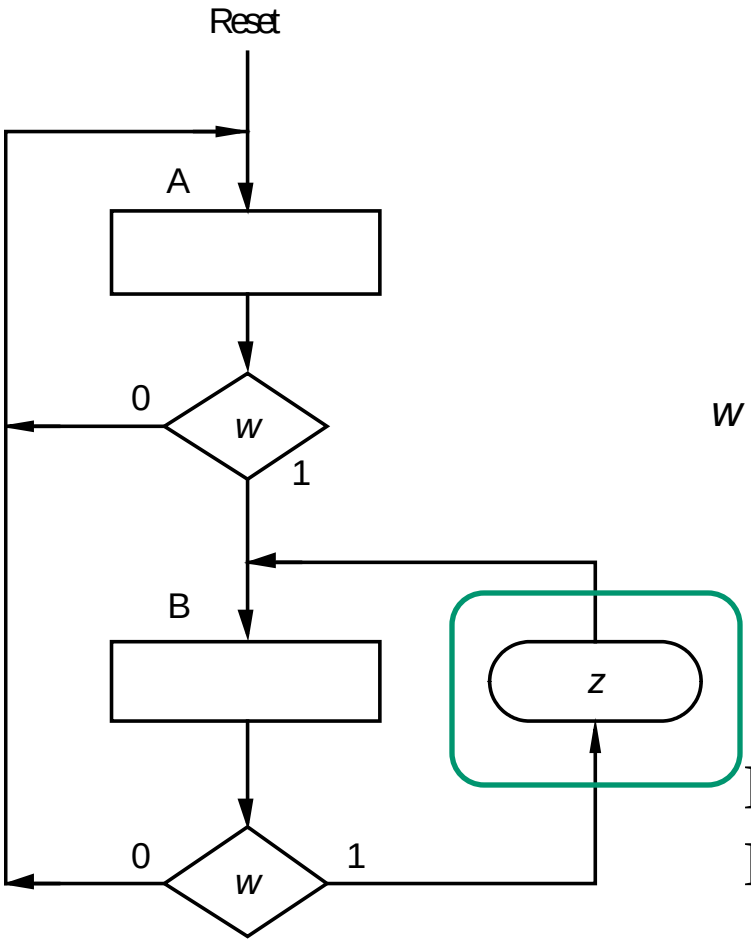


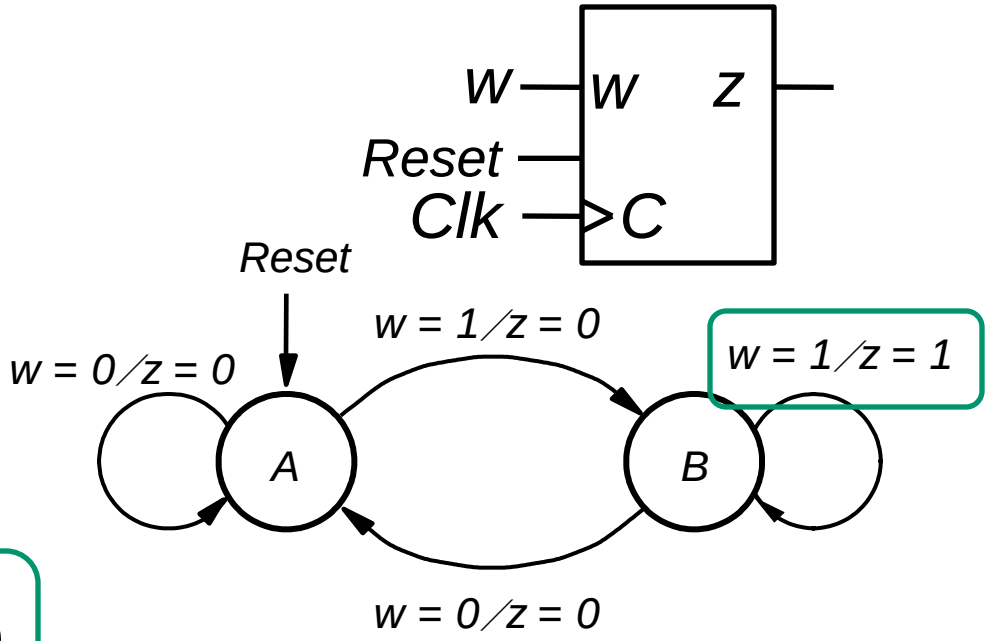


Bara i tillstånd C har z värdet 1

William Sandqvist william@kth.se

”två i rad” Mealy





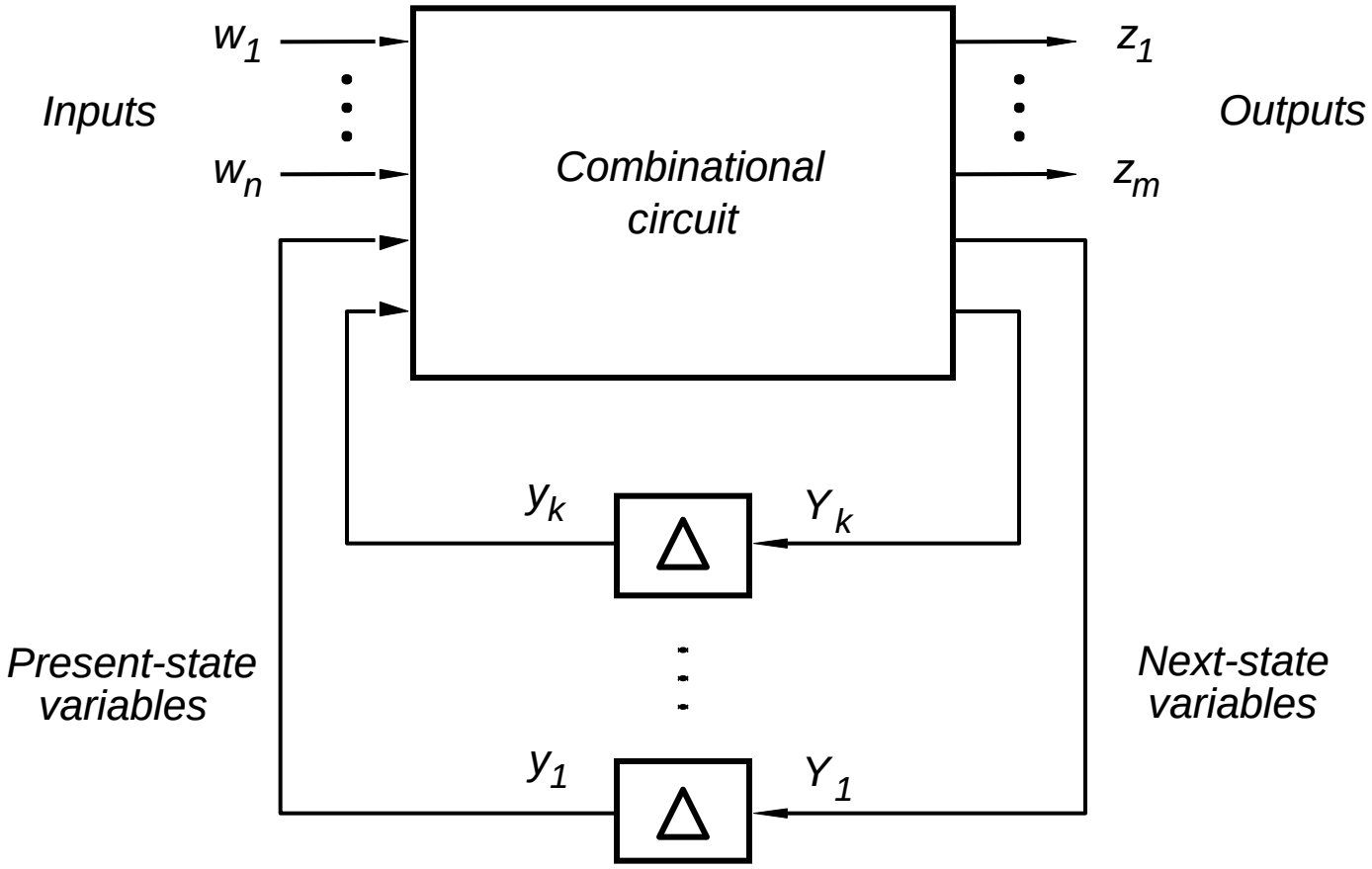
Bara vid tillståndsövergången
B-till-B med $w = 1$ har z värdet 1

Formell modell för tillståndsautomat

För att behandla tillståndsmaskiner matematiskt behöver man en formell modell

Följande modell kan beskriva både Moore- och Mealy-automaten

Formell modell för tillståndsautomat



Formell modell för tillståndsautomat

En tillståndsmaskin kan formellt definieras med

$$M = (W, Z, S, \varphi, \lambda)$$

W, *Z*, och *S* beskriver ingångarna (W), utgångarna (Z) och tillstånd (S)

φ beskriver tillståndsövergångsfunktionen

λ beskriver utgångsfunktion

Formell modell för tillståndsautomat

$$M = (W, Z, S, \varphi, \lambda)$$

$$S(t + \Delta t) = \varphi(W(t), S(t))$$

$$\lambda_{Moore}(t) = \lambda(S(t)) \qquad \lambda_{Mealy}(t) = \lambda(W(t), S(t))$$

$$S(t + \Delta t) = Y_k \ldots Y_1 = \varphi(w_n \ldots w_1, y_k \ldots y_1)$$

$$Z_{Moore} = z_m \ldots z_1 = \lambda(y_k \ldots y_1) \qquad Z_{Mealy} = z_m \ldots z_1 = \lambda(w_n \ldots w_1, y_k \ldots y_1)$$

Nu har Du sett hur man uttrycker detta i matematikämnet!

William Sandqvist william@kth.se

Exempel på automater

- Exemplet med dryckesautomaten
- Demo – ”tre i rad krets” är det tillräckligt säkert?

William Sandqvist william@kth.se