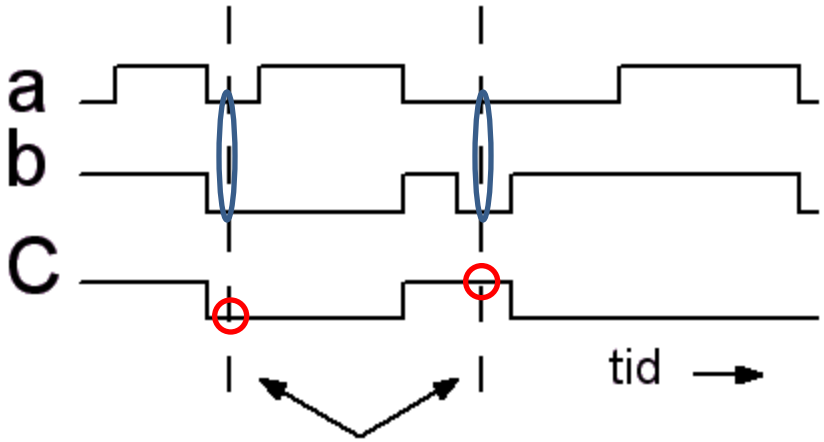
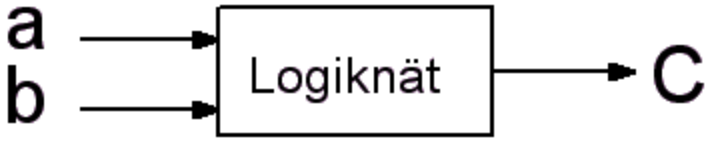
IE1205 Digital Design:

F10: Synkrona tillståndsautomater del 2

Sekvensnät





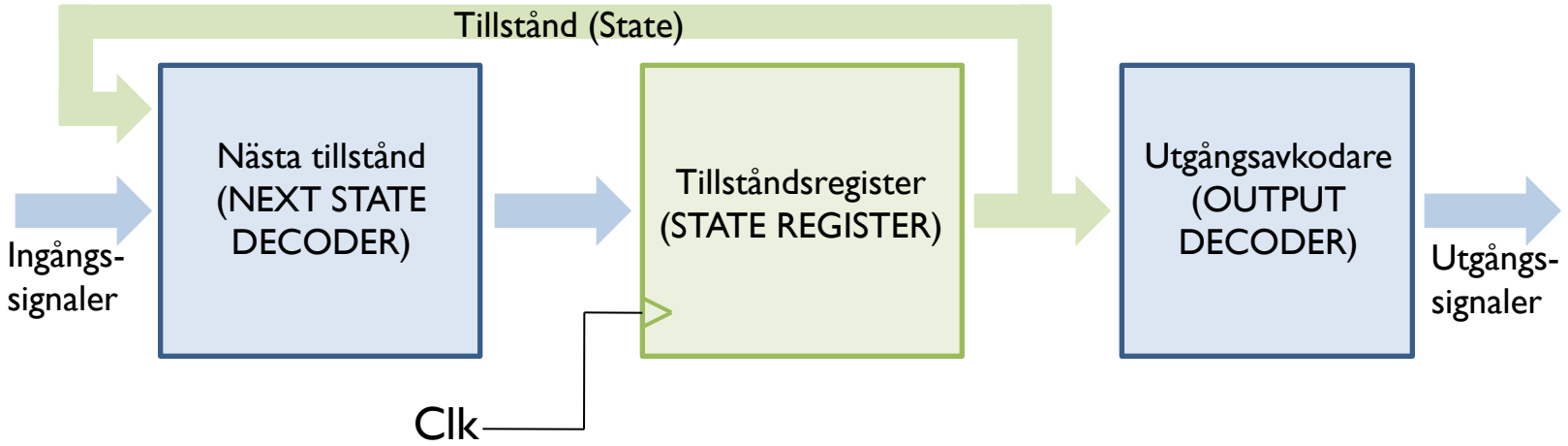
Om en och samma insignal kan ge upphov till olika utsignal, är logiknätet ett sekvensnät.
Det måste då ha ett ***inre minne*** som gör att utsignalen påverkas av både nuvarande och föregående insignaler!

Grundläggande metod för konstruktion av statemaskiner

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

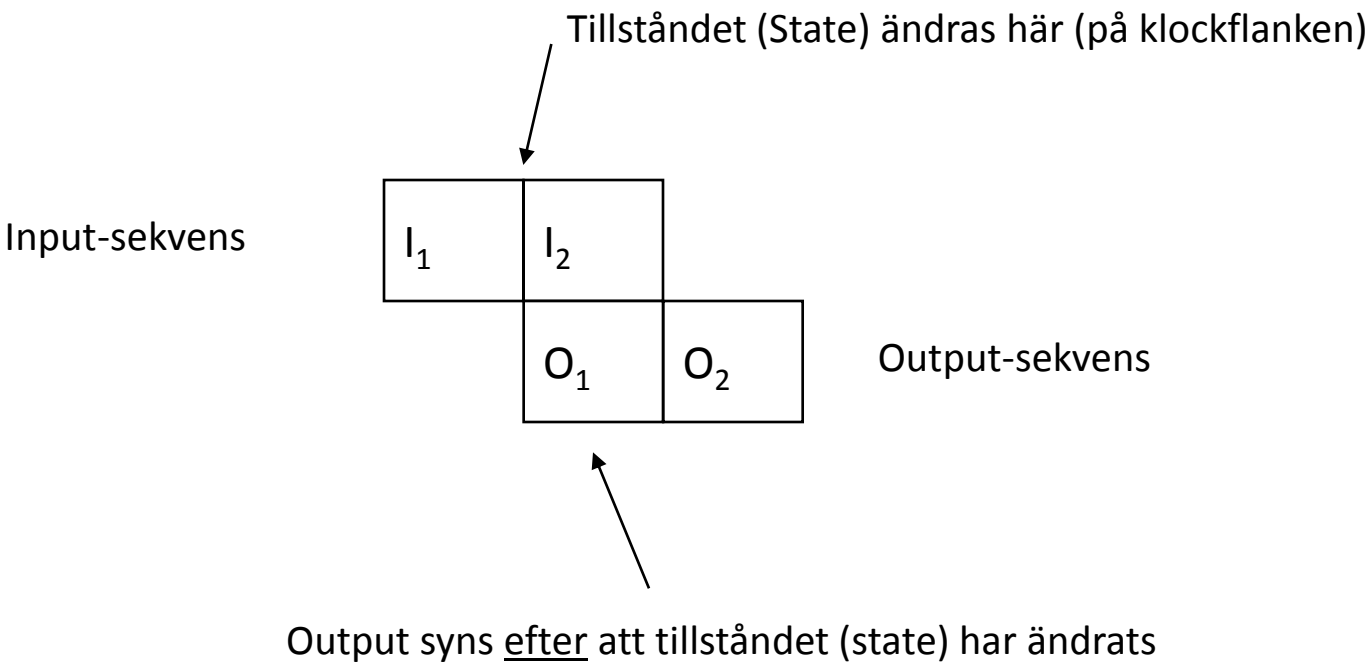
Exempel: Flaskautomat

Moore-Automat

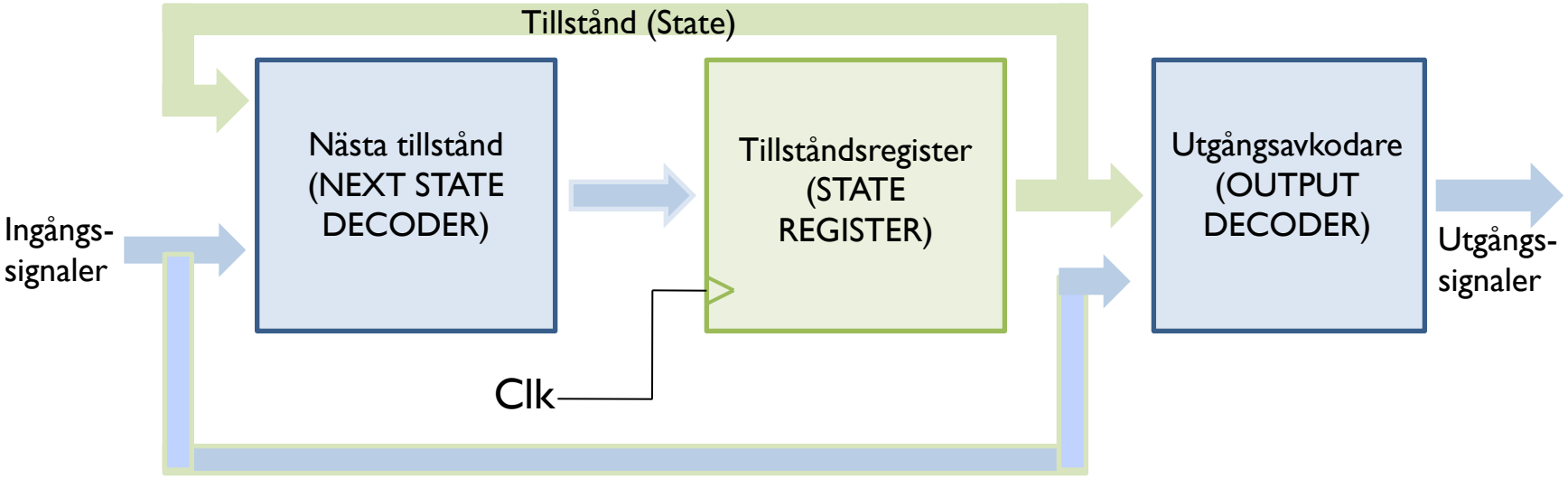


- I en Moore-automat beror utgångssignalerna bara på nuvarande tillstånd

Input vs Output - Moore

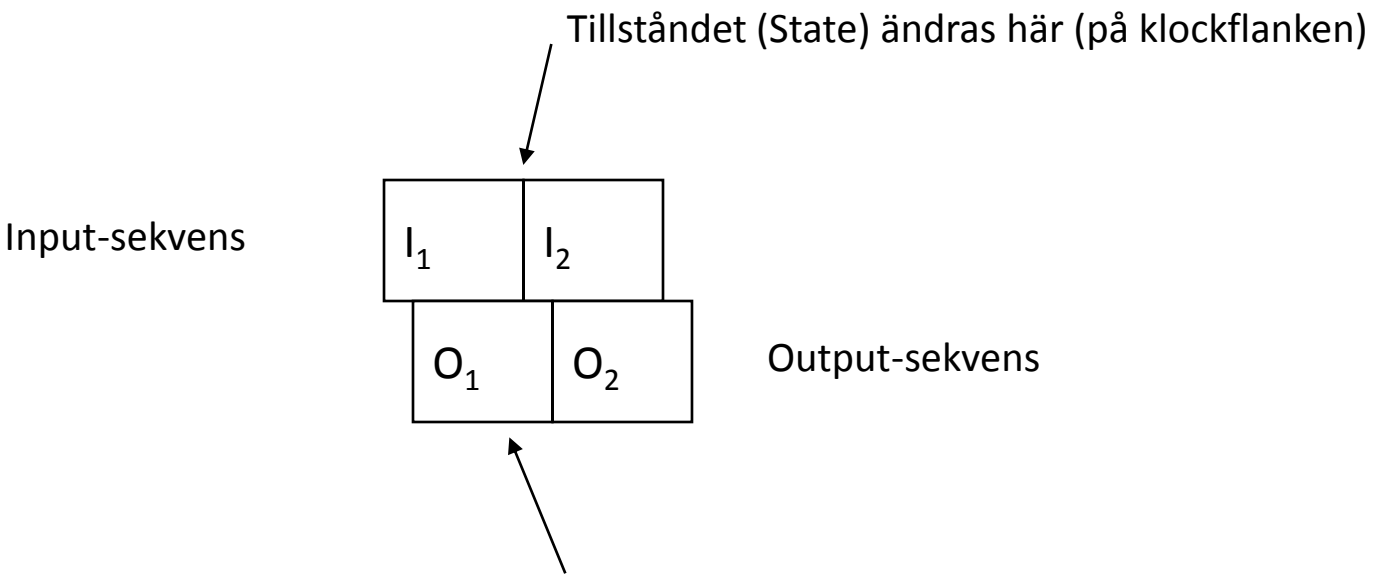


Mealy-Automaten (forts.)



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

Input vs Output - Mealy



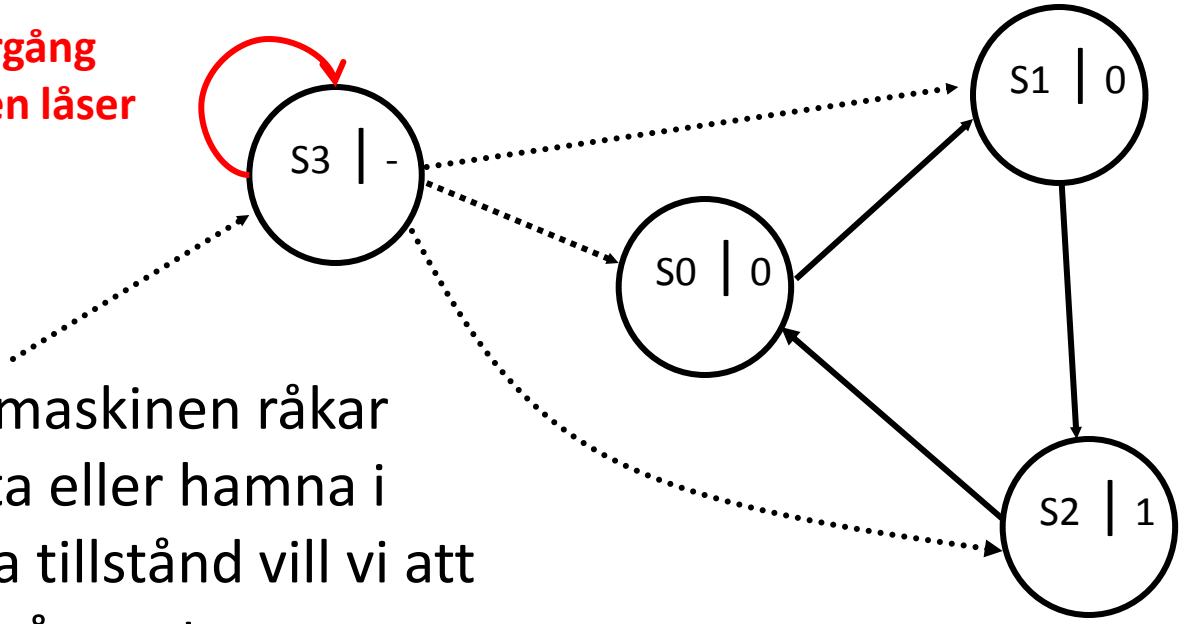
Överblivna tillstånd

- Ibland får man några states över när man väljer kod.
- Överblivna states måste tas om hand så att inte statemaskinen låser sig vid uppstart (om man inte använder sig av reset vill säga)

Exempel: Sekvensräknare (0,0,1)

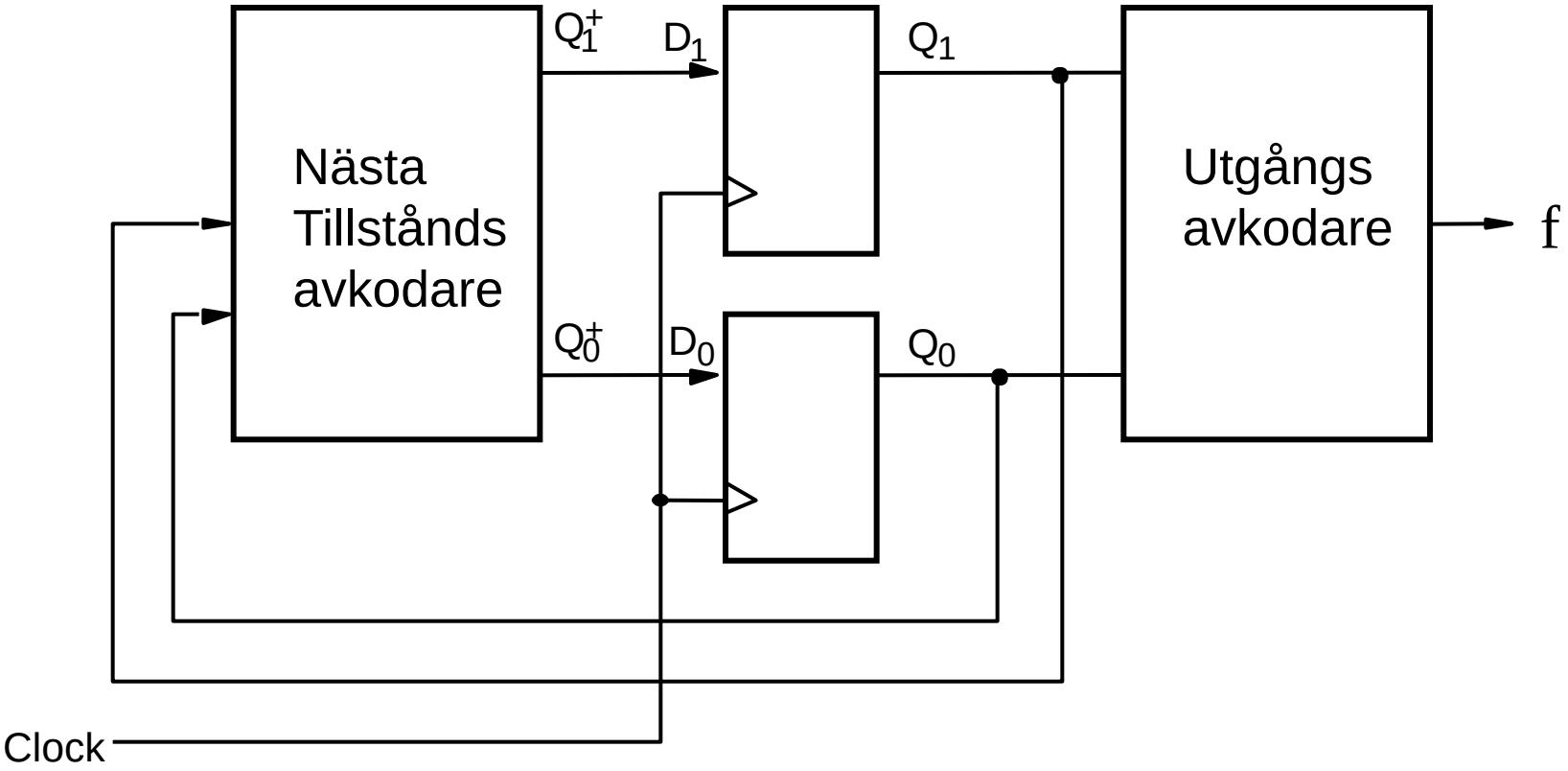
3 tillstånd => 2 vippor. Ett tillstånd över...

Farlig övergång (automaten låser sig)



Om maskinen råkar starta eller hamna i detta tillstånd vill vi att den så snart som möjligt hittar in i sekvensen.

Räknaren som automat

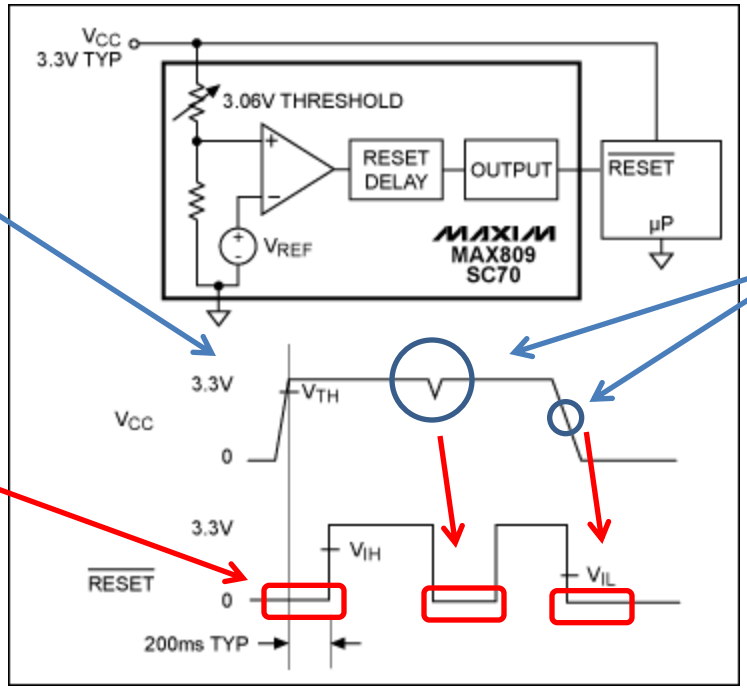


Nextstate-funktionen

	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)	- -

(RESET-generator chip)

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

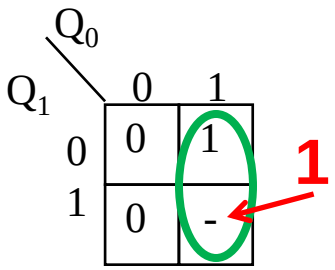


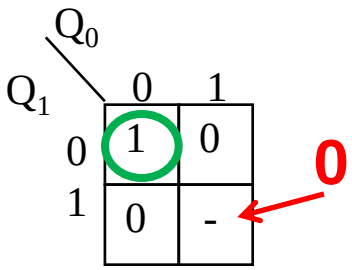
Om matnings-spänningen får problem, eller sjunker under viss nivå, så blir det **RESET**

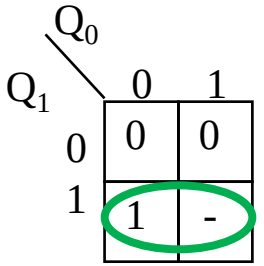
Bättre än att behöva skaffa extra skydd, är att designa förebyggande och från början ”ta hand om” alla tillstånd …

Karnaughdiagram

	Nuv. värde Q_1Q_0	Utsignal f	Nästa värde $Q_1^+Q_0^+$	D-vippa 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)	- -







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

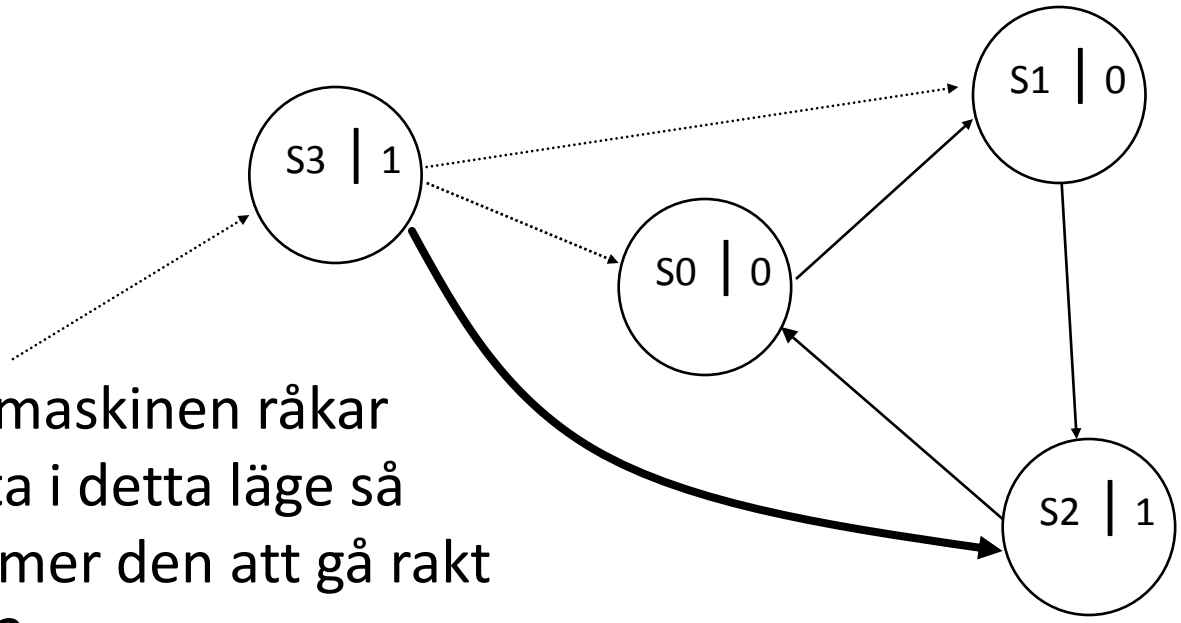
- **OK, 10 inte 11!**

State-tabell efter Karnaugh-minimering

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 i S2 i huvudsekvensen...

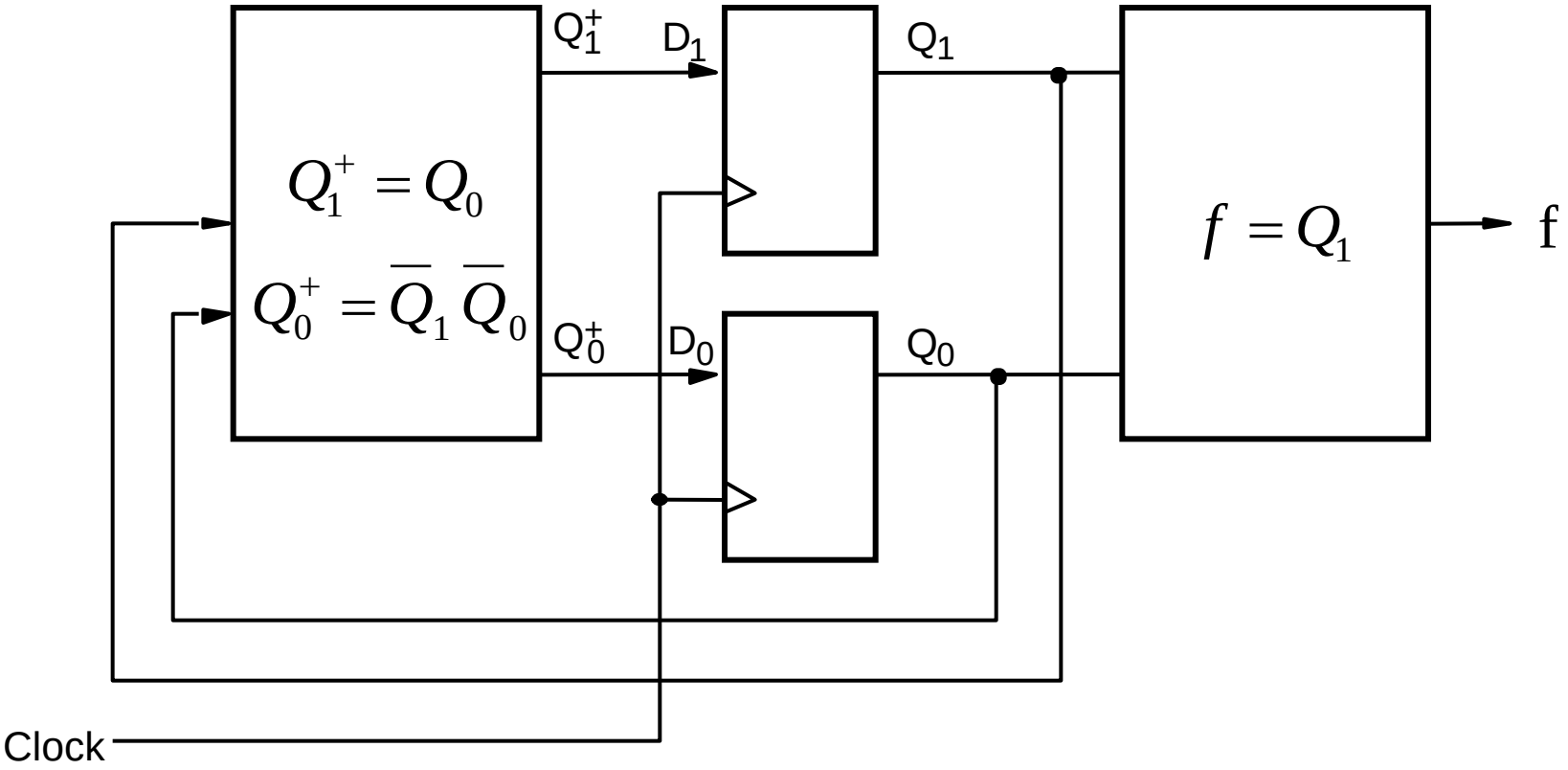
Exempel: Sekvensräknare (0,0,1)



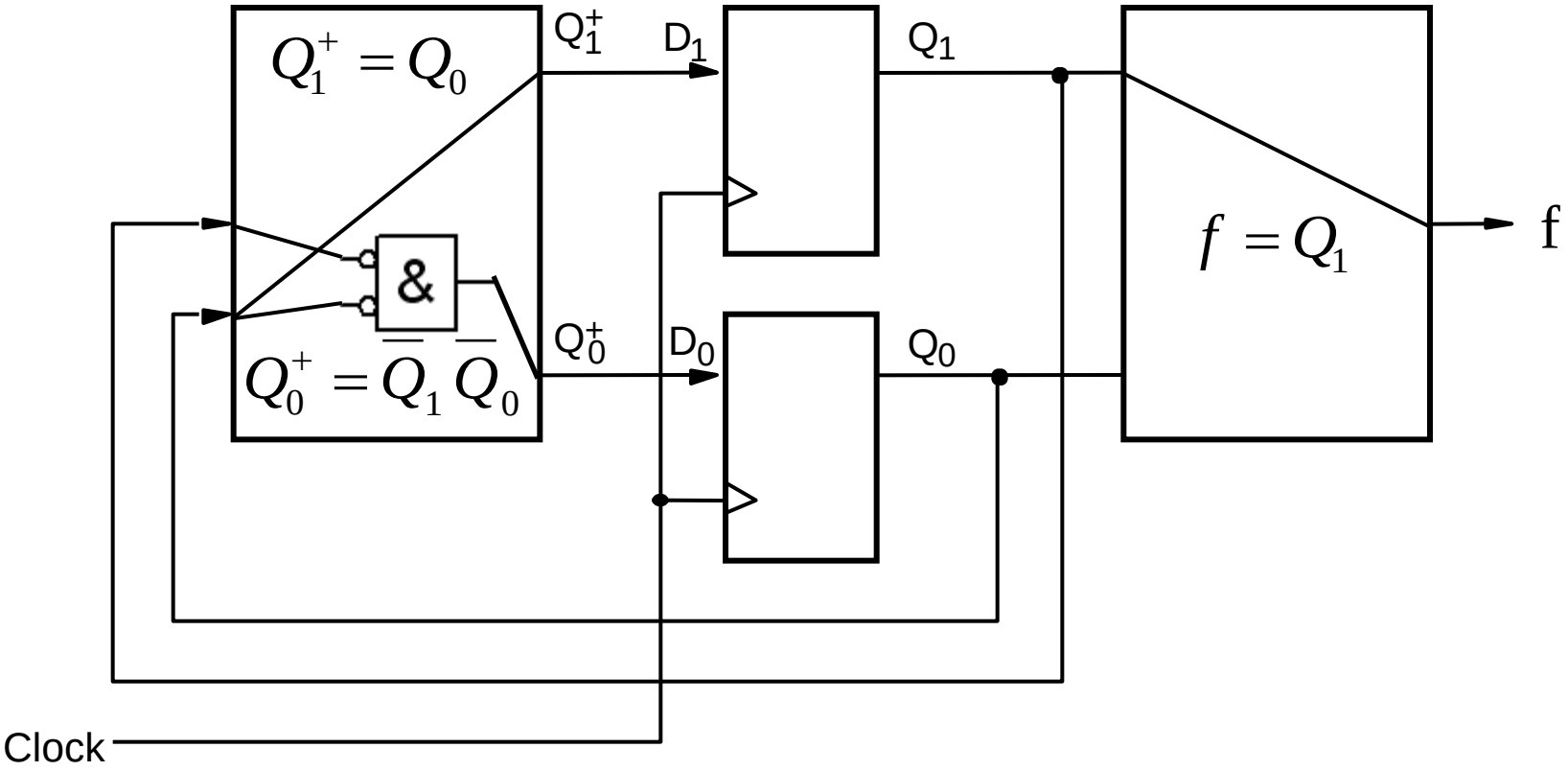
Om maskinen råkar starta i detta läge så kommer den att gå rakt till S2.

Bättre: Meddela att ett fel har inträffat!

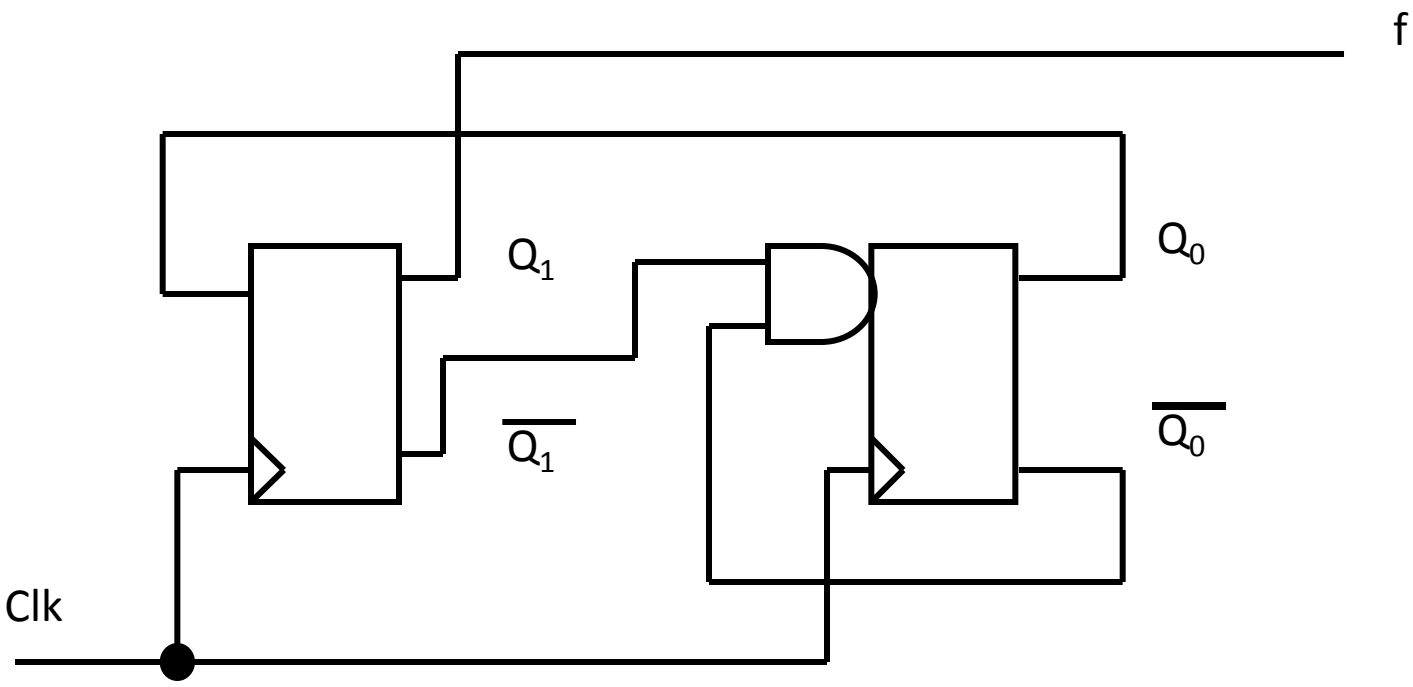
Räknaren



Grindnät för sekvensen



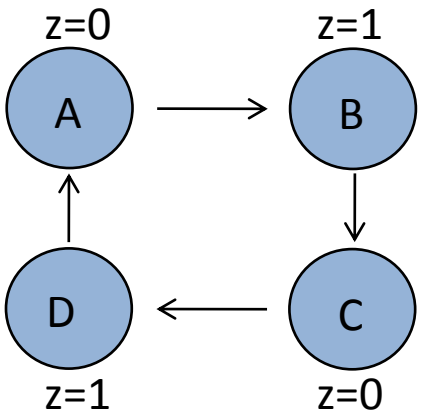
Grindnät för sekvensen

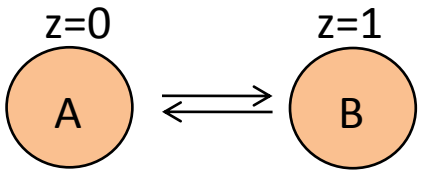


Alternativt sätt att rita

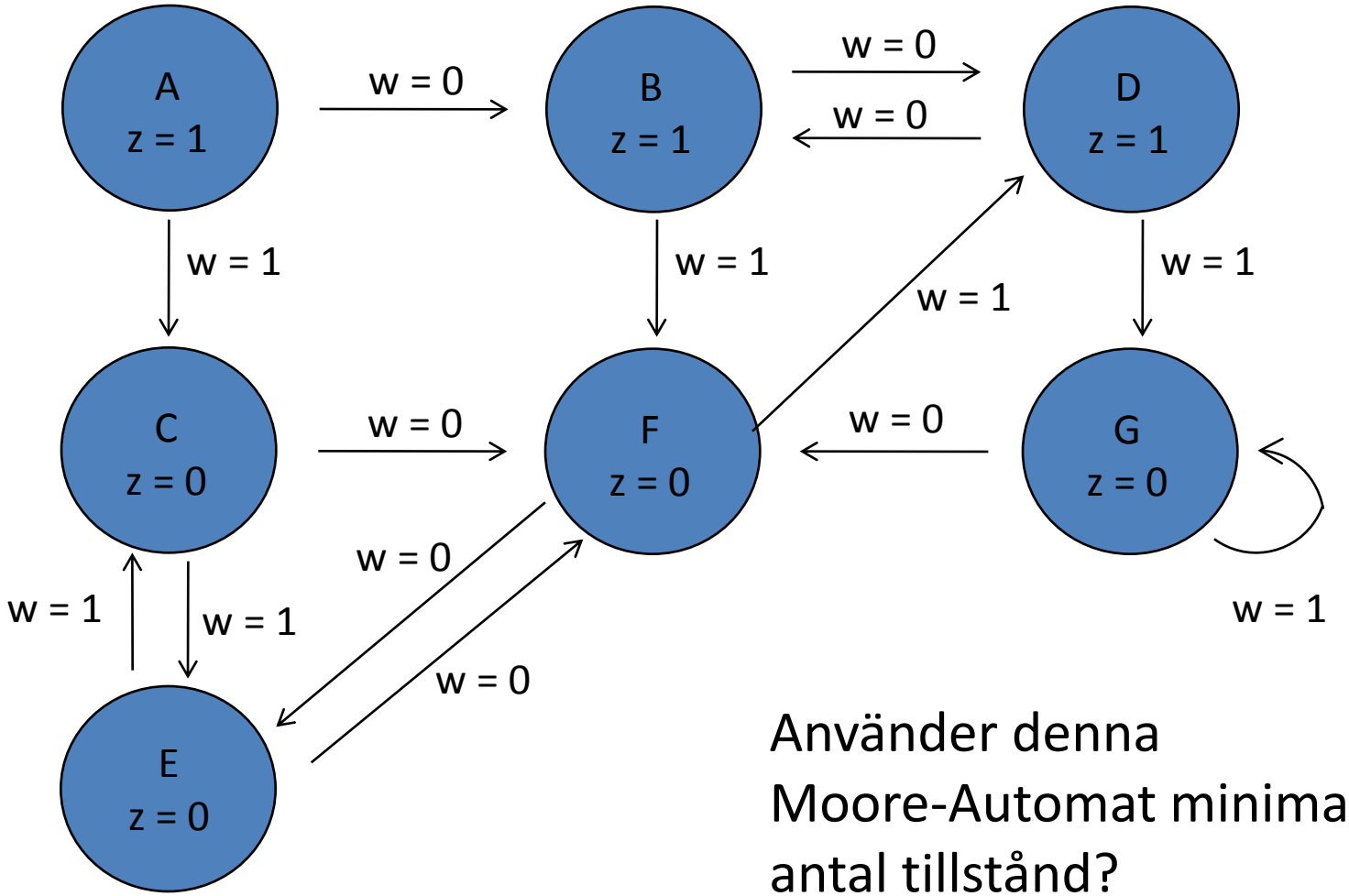
Tillståndsminimering

- När man konstruera komplexa tillståndsmaskiner så kan det lätt hända att det finns ekvivalenta och därmed redundanta tillstånd som kan tas bort för att få en effektivare implementering





Exempel Tillståndsminimering



Använder denna Moore-Automat minimalt antal tillstånd?

Tillståndsminimering

Hur kan vi visa att två tillstånd är ekvivalenta?

S_i och S_j är ekvivalenta om och endast om de, för varje möjlig insekvens, genererar identisk utsekvens oberoende om S_i eller S_j är ursprungligt tillstånd

Blockminimering

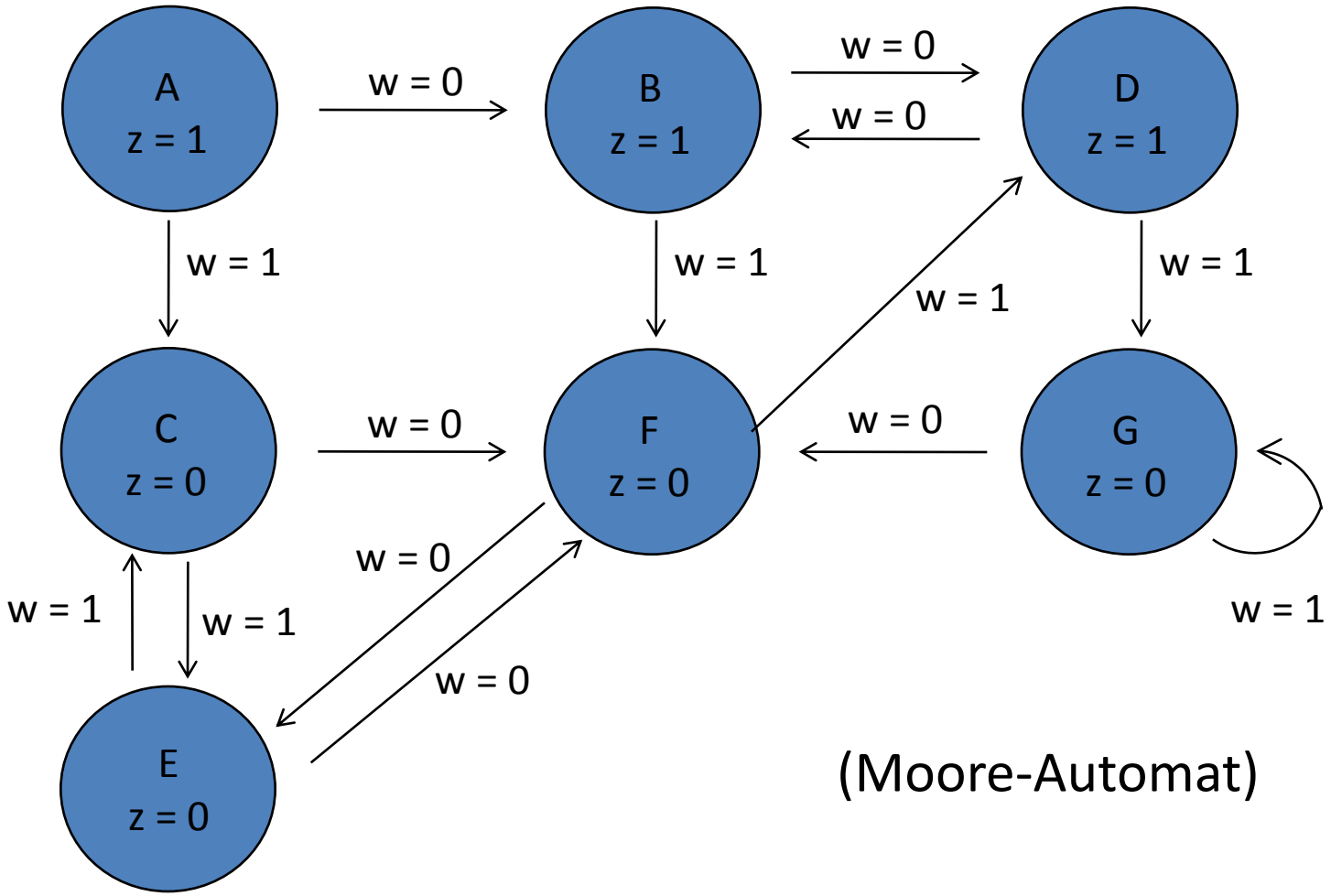
Metod för att minimera antalet tillstånd

1. Börja med ett block
2. Gruppera tillstånd som har olika utsignaler
3. För varje tillstånd i respektive grupp, dela upp i nya grupper beroende på om dess efterföljande tillstånd är i olika grupper
4. Repetera tills föregående uppdelning ekvivalent med nuvarande

Tillståndsminimering

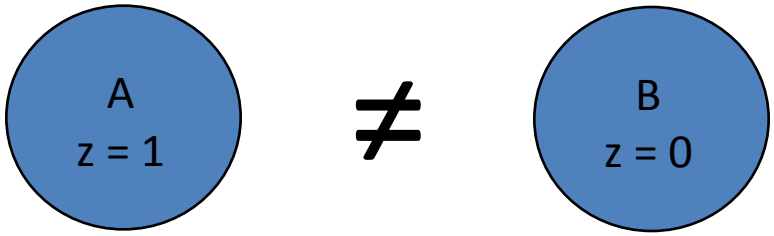
- Följande exempel illustrerar en minimeringsmetod för att förstå konceptet för tillståndsminimeringen
- Syntesverktyg använder andra algoritmer

Exempel Tillståndsminimering



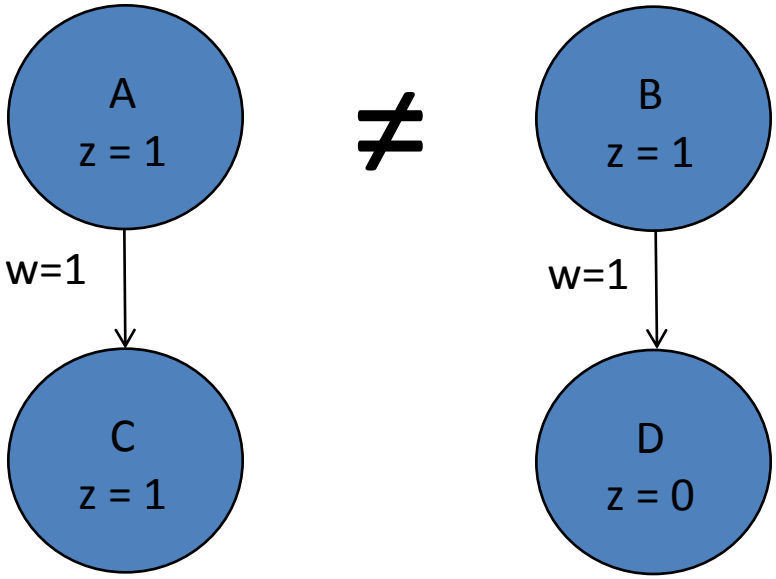
Tillståndsminimering Grundidé

- Två tillstånd är inte ekvivalenta om de har olika utgångsvärden, om
  1. de har olika utgångsvärden

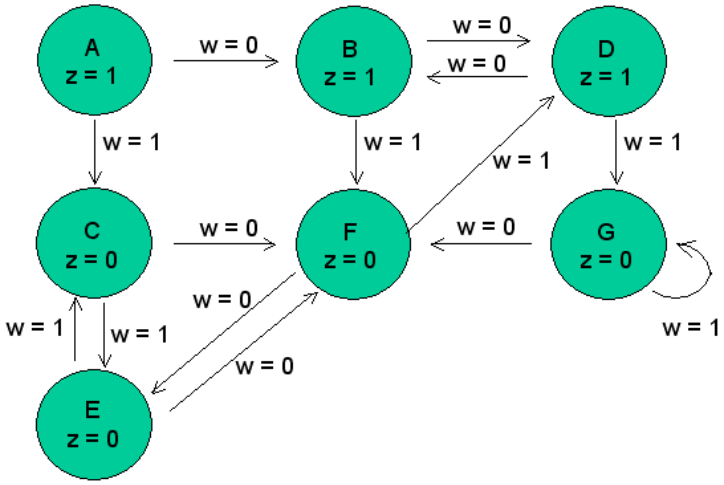


Tillståndsminimering Grundidé

- Två tillstånd är inte ekvivalenta om de har olika utgångsvärden,
  2. om åtminstone en av tillsåndsövergångarna går till olika efterföljande tillstånd



Tillståndstabell



Ursprungligt tillståndsdiagram

Present state	Next state		Output z
	w = 0	w = 1	
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

Exempel Tillståndsminimering

- Start
  - Bara en block med alla tillstånd
    - P_1=(ABCDEFG)

Exempel Tillståndsminimering

- Steg 1
  - Vilka tillstånd har olika utsignaler?
    - ABD har utsignal z = 1
    - CEFG har utsignal z = 0
    - => P_2=(ABD)(CEFG)

Exempel Tillståndsminimering

- Steg 2
  - Vilka tillstånd har olika följdtillstånd?
    - Block ABD
      - O-successor: A > B, B > D, D > B (alla övergångar till samma block)
      - 1-successor: A > C, B > F, C > G (alla övergångar till samma block)
    - Block CEFG
      - O-successor: C > F, E > F, F > E, G > F (alla övergångar till samma block)
      - 1-successor: C > E, E > C, F > D, G > G (F > D går till ett annat block)
    - => P_3=(ABD)(CEG)(F)

Exempel Tillståndsminimering

- Steg 3
  - Vilka tillstånd har olika följdtillstånd?
    I föregående iteration var alla övergångar från ABD till samma block (CEFG), men eftersom detta block ändrats (till (CEG)(F)) måste vi analysera ABD igen:
    - Block ABD
      - O-successor: A > B, B > D, D > B (alla övergångar till samma block)
      - 1-successor: A > C, B > F, C > G (B > F går till ett annat block)
    - => P_4=(AD)(B)(CEG)(F)

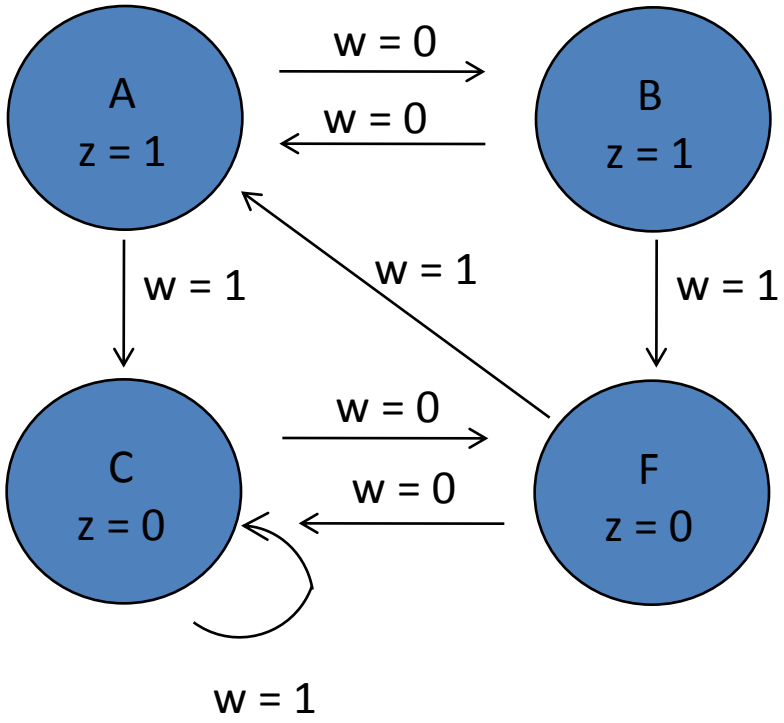
Exempel Tillståndsminimering

- Steg 4
  - Vilka tillstånd har olika följdtillstånd? (Vi behöver bara analysera (CEG)
    - Block CEFG
      - O-successor: C > F, E > F, G > F (alla övergångar till samma block)
      - 1-successor: C > E, E > C, G > G (alla övergångar till samma block)
  - Iterationen är avslutat
    - Fyra tillstånd behövs: P_4=(AD)(B)(CEG)(F)

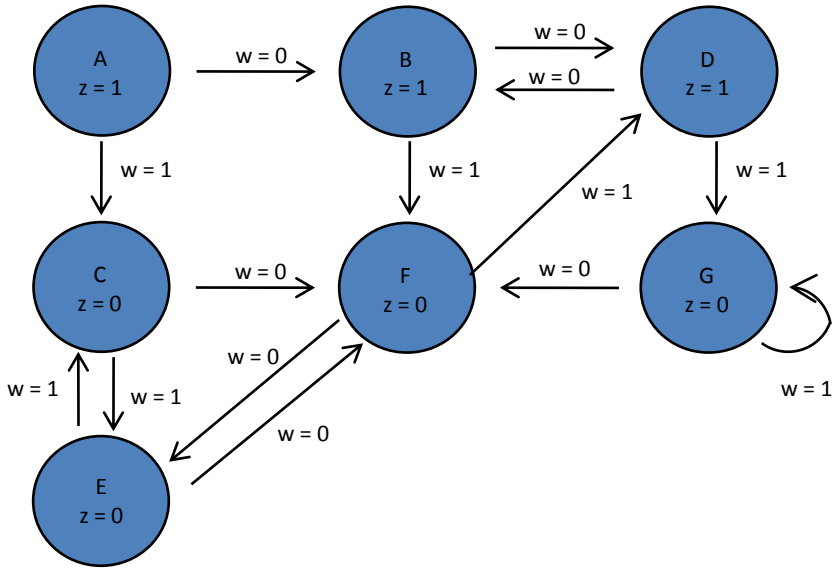
Slutgiltig Tillståndstabell

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

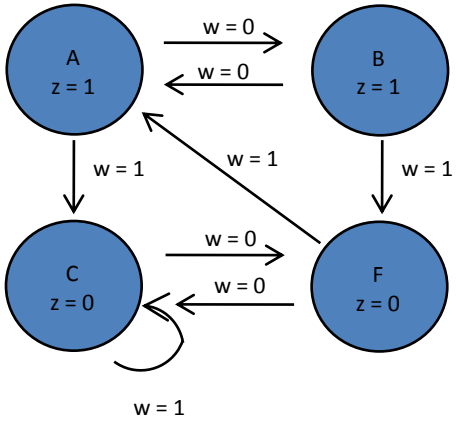
Slutgiltig Tillståndsdiagram



Tillståndsminimering



Före minimering

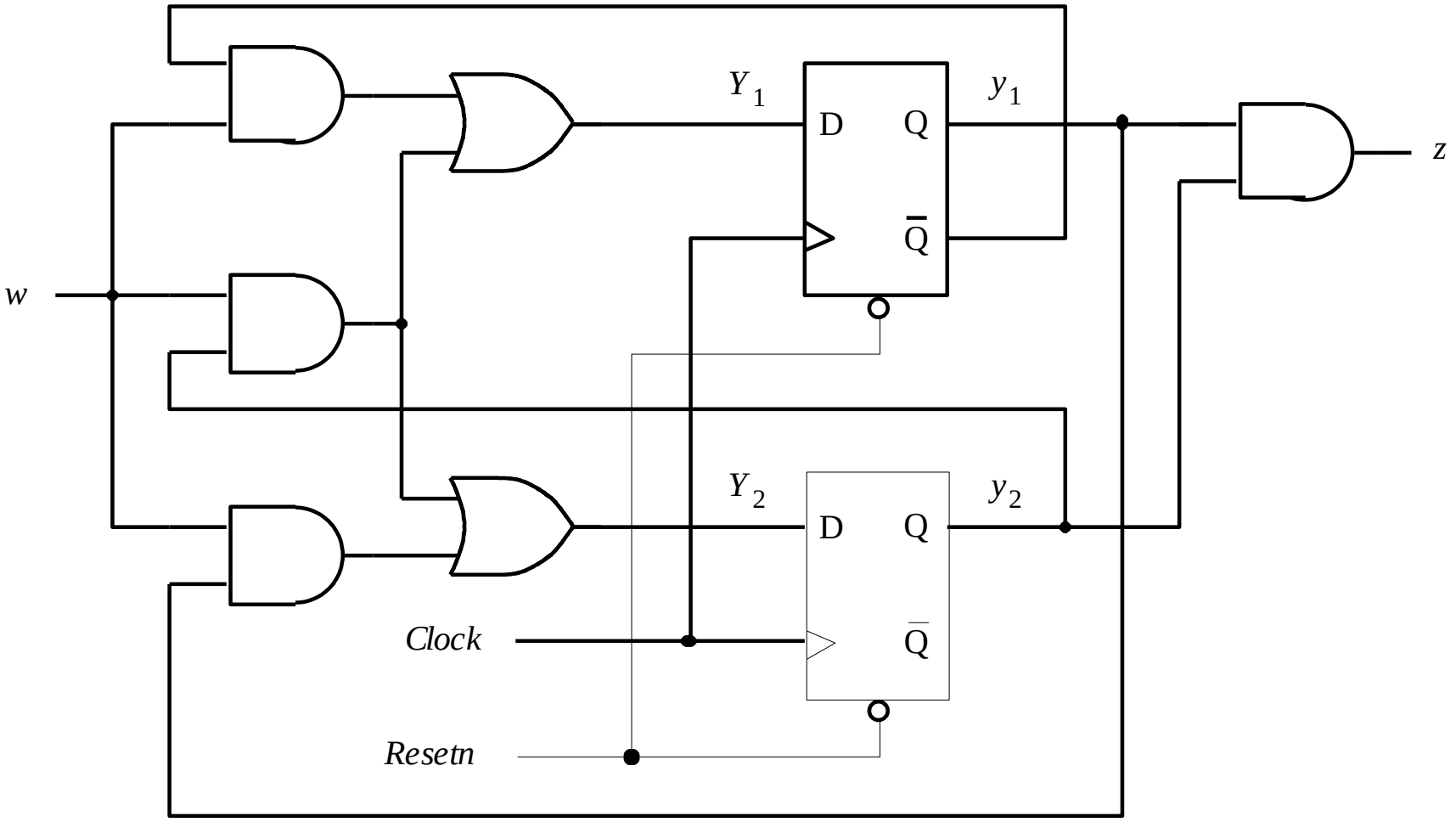


Efter minimering

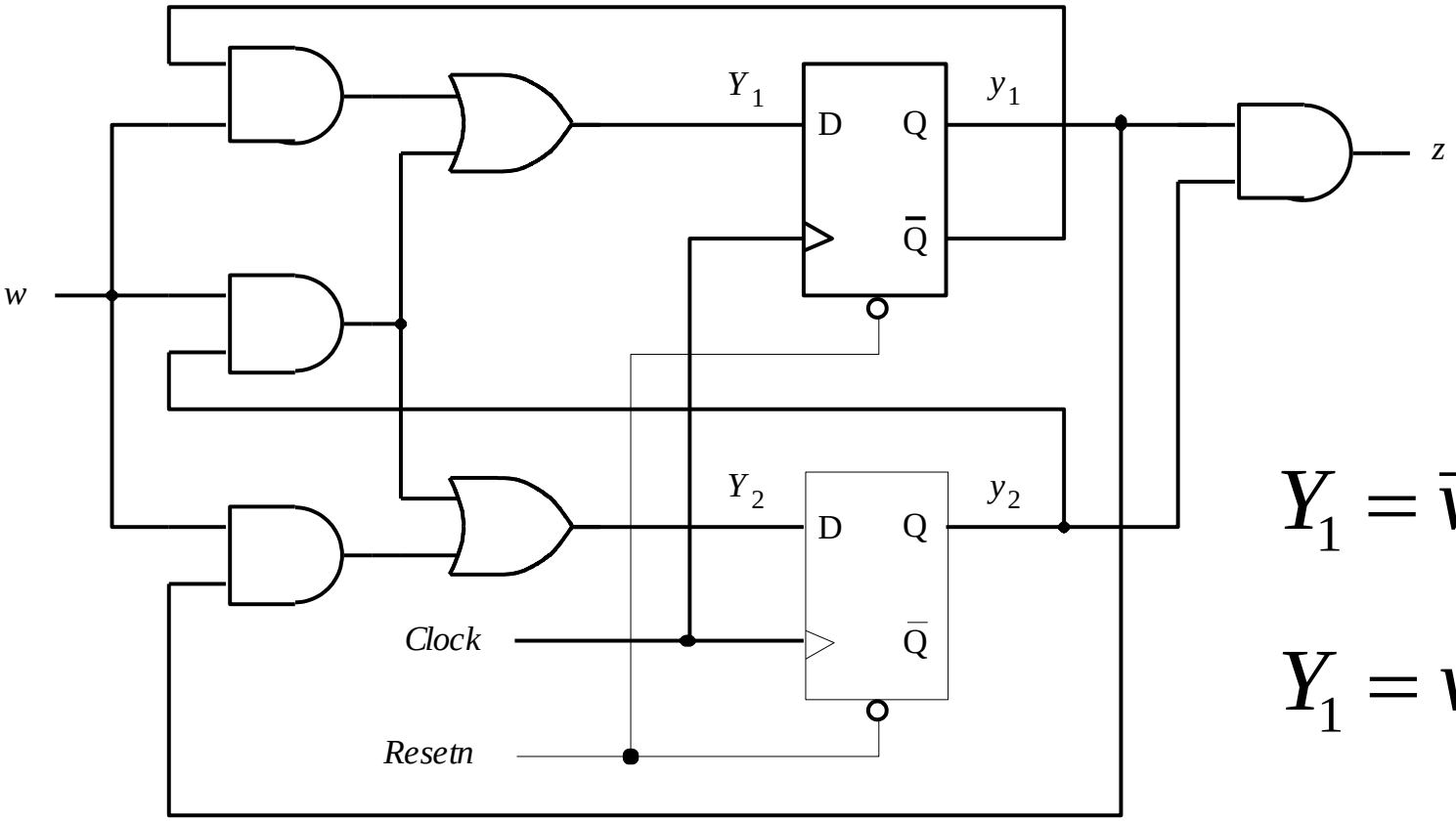
Analys av sykrona sekvensnät

- Given en implementering av en synkron krets, vi kan ta fram funktionen genom att göra syntesstegen I omvänd ordning!
  1. Ta fram uttrycken för
     - nästa-tillståndsavkodare
     - utgångsavkodare
  2. Ta fram tillståndstabellen
  3. Rita tillståndsdiagrammet

Exempel
Analys av sykrona sekvensnät



Exempel
Analys av sykrona sekvensnät



Alt: 1

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

Alt: 2

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

Alt: 3

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

Exempel
Analys av sykrona sekvensnät

1. Ta fram uttrycken för
   - nästa-tillståndsavkodare
   - utgångsavkodare

$$
\begin{aligned}
Y_1 &= w\bar{y}_1 + wy_2 \\
Y_2 &= wy_1 + wy_2 \\
z &= y_1 y_2
\end{aligned}
$$

Exempel
Analys av sykrona sekvensnät

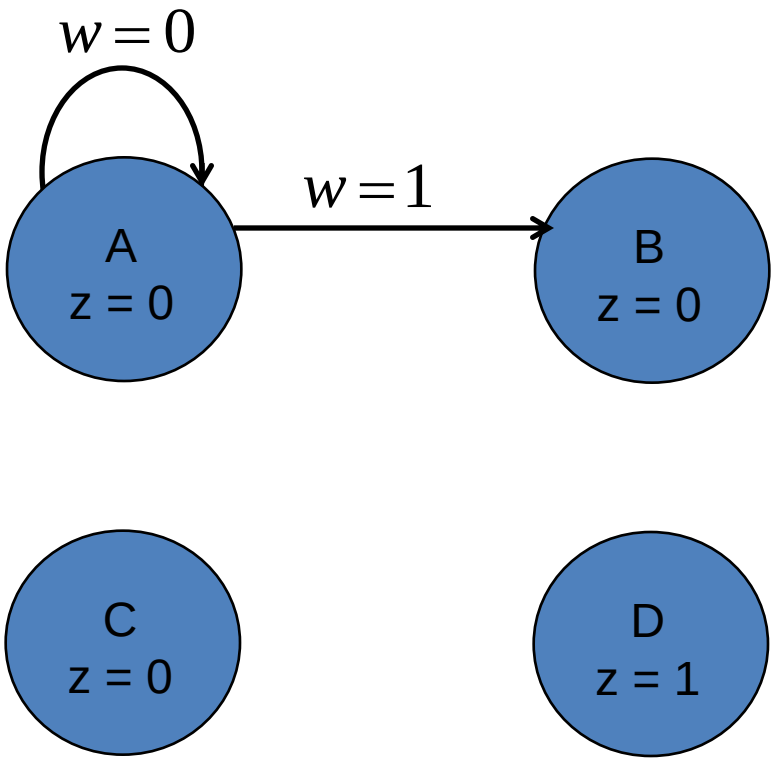
2. Ta fram tillståndstabellen

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

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

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

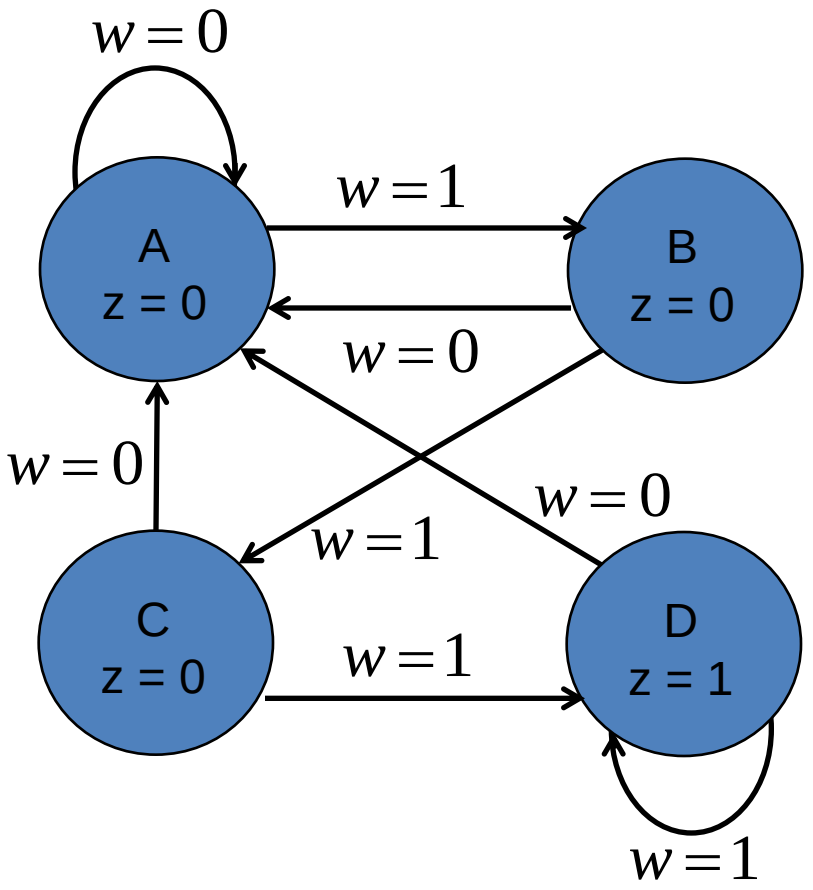


Rita färdigt tillståndsdiagrammet själv.
(På övning 6 löser vi ett liknande problem – kretsen är en ”tre i rad” -krets).

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

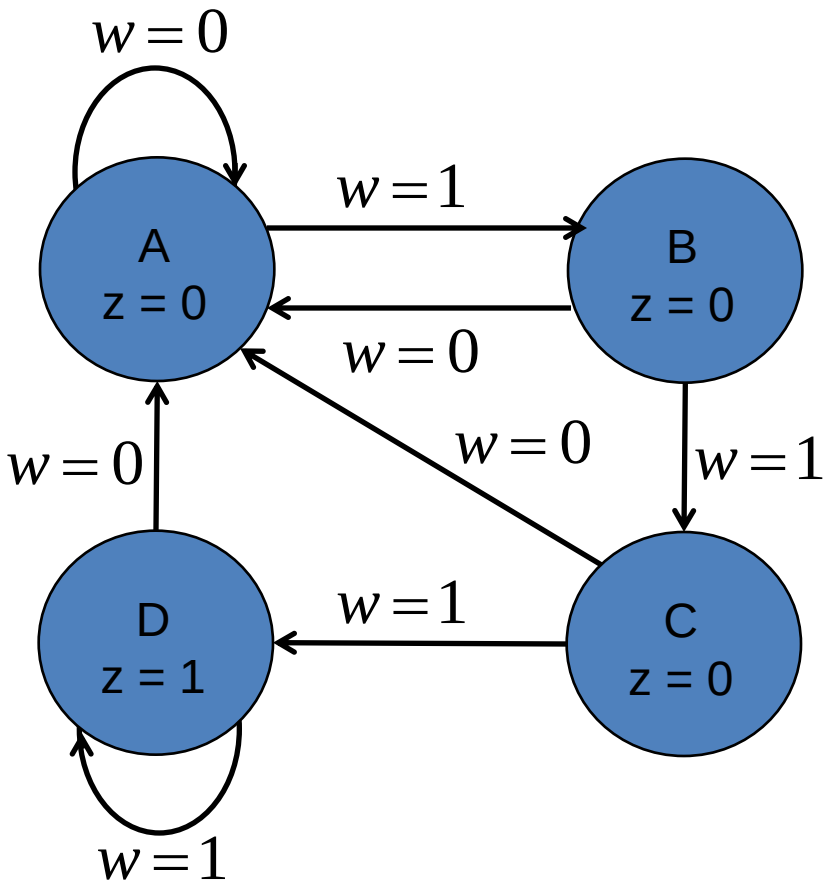
Ibland kan man behöva ändra ordningen på tillstånden för att få ett tydligare diagram



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

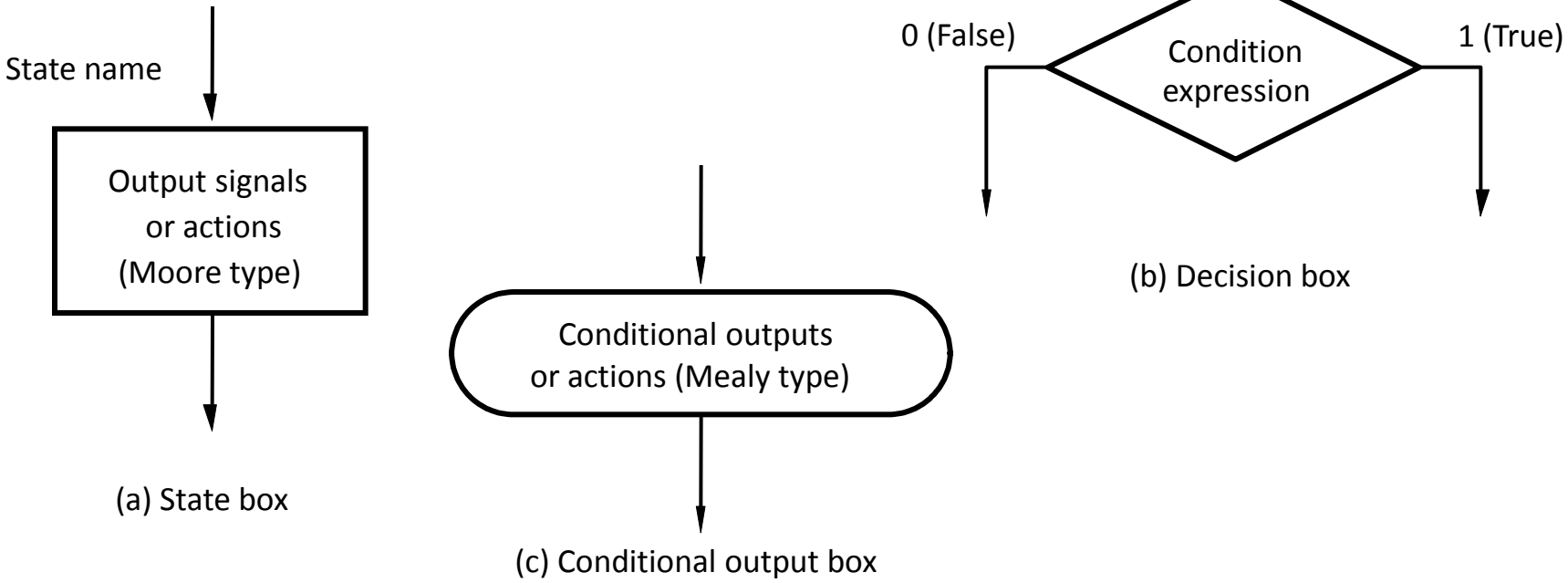


ASM-Charts

- För att beskriva större tillståndsmaskiner används ofta ett annat diagram: Algorithmic State Machine (ASM) Charts
- Det är ett flödesdiagram med tre element
  - Tillståndslåda (State Box)
  - Beslutslåda (Decision Box)
  - Villkorlig utgångslåda (Conditional Output Box)

ASM-Charts

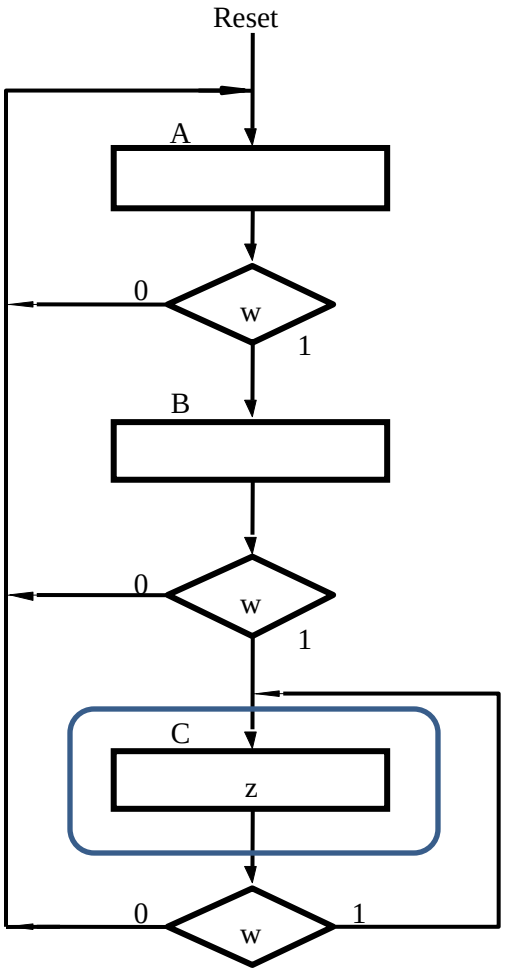
- En ASM-Chart är ett flödesdiagram med tre element

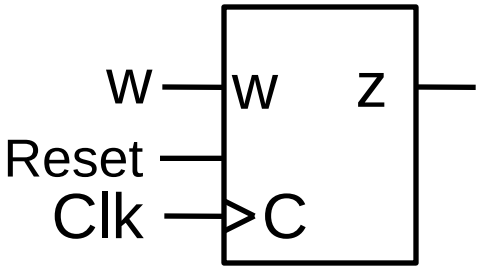


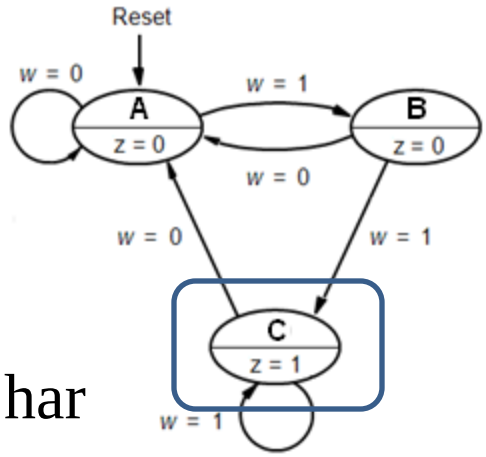
ASM-Charts

- 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

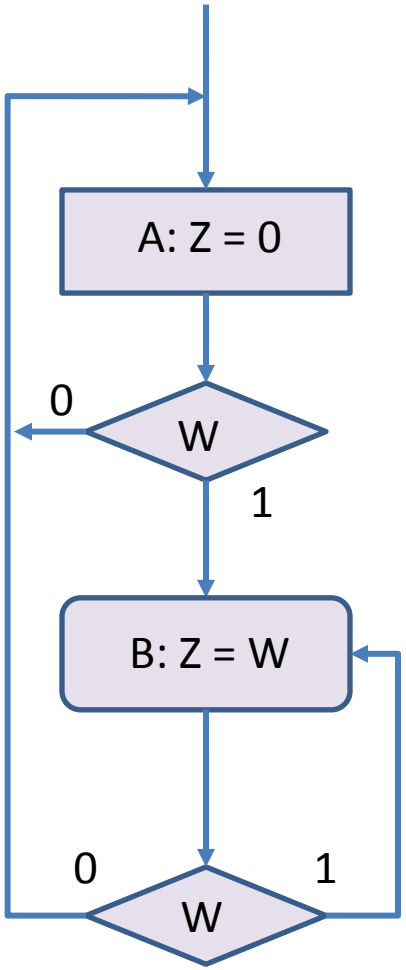


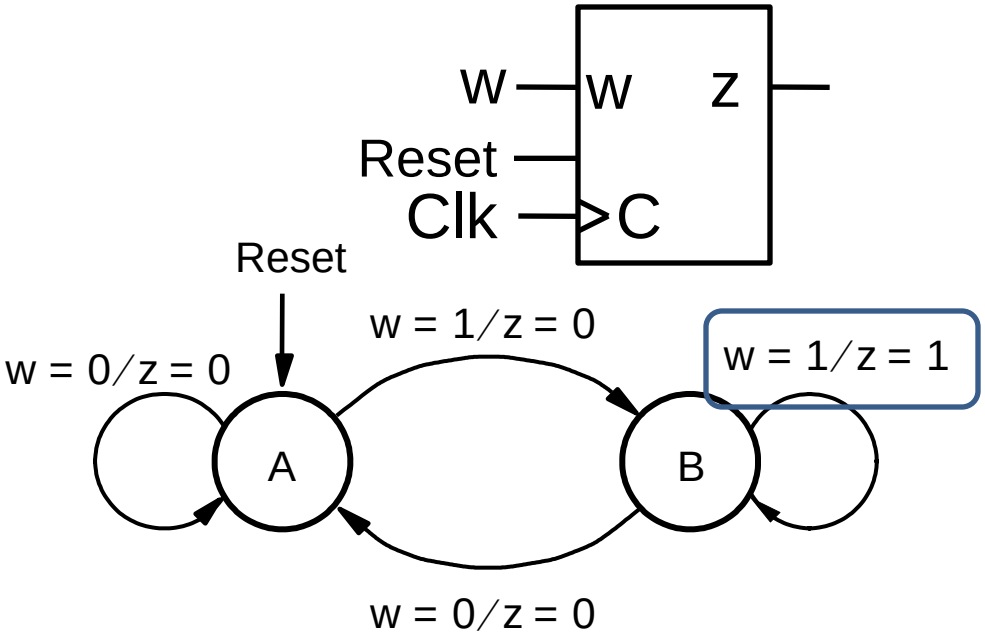




Bara i tillstånd C har z värdet 1

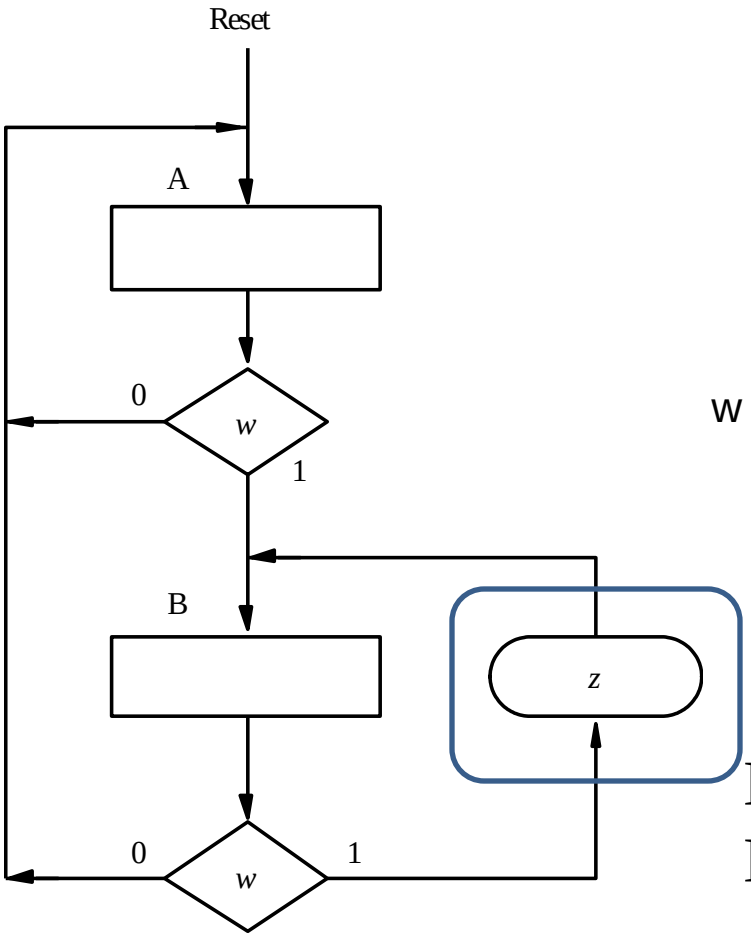
”två i rad” Mealy

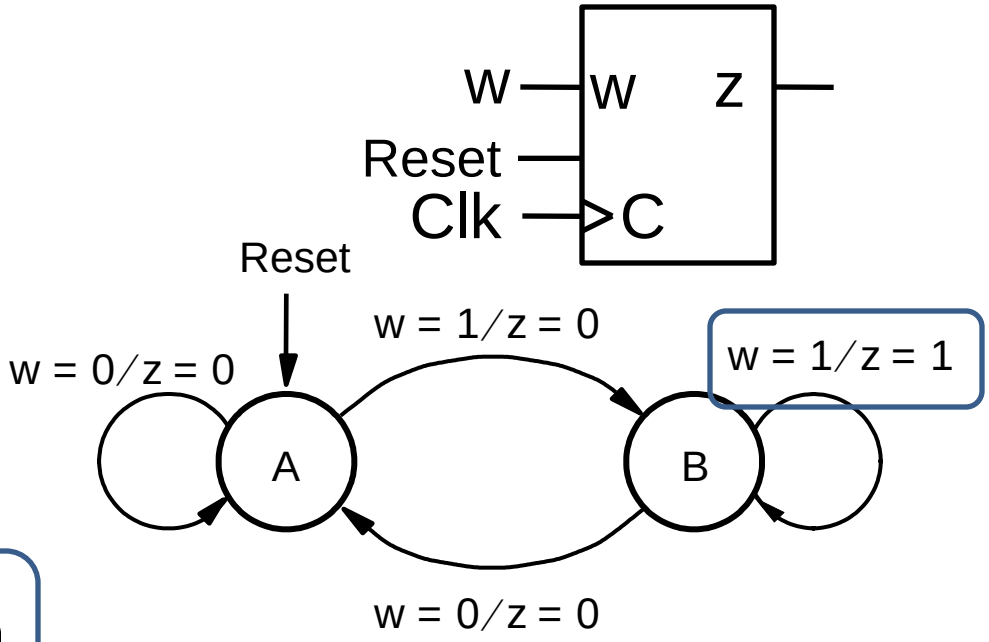




Bara vid tillståndsövergången
B-till-B med w = 1 har z värdet 1

”två i rad” Mealy



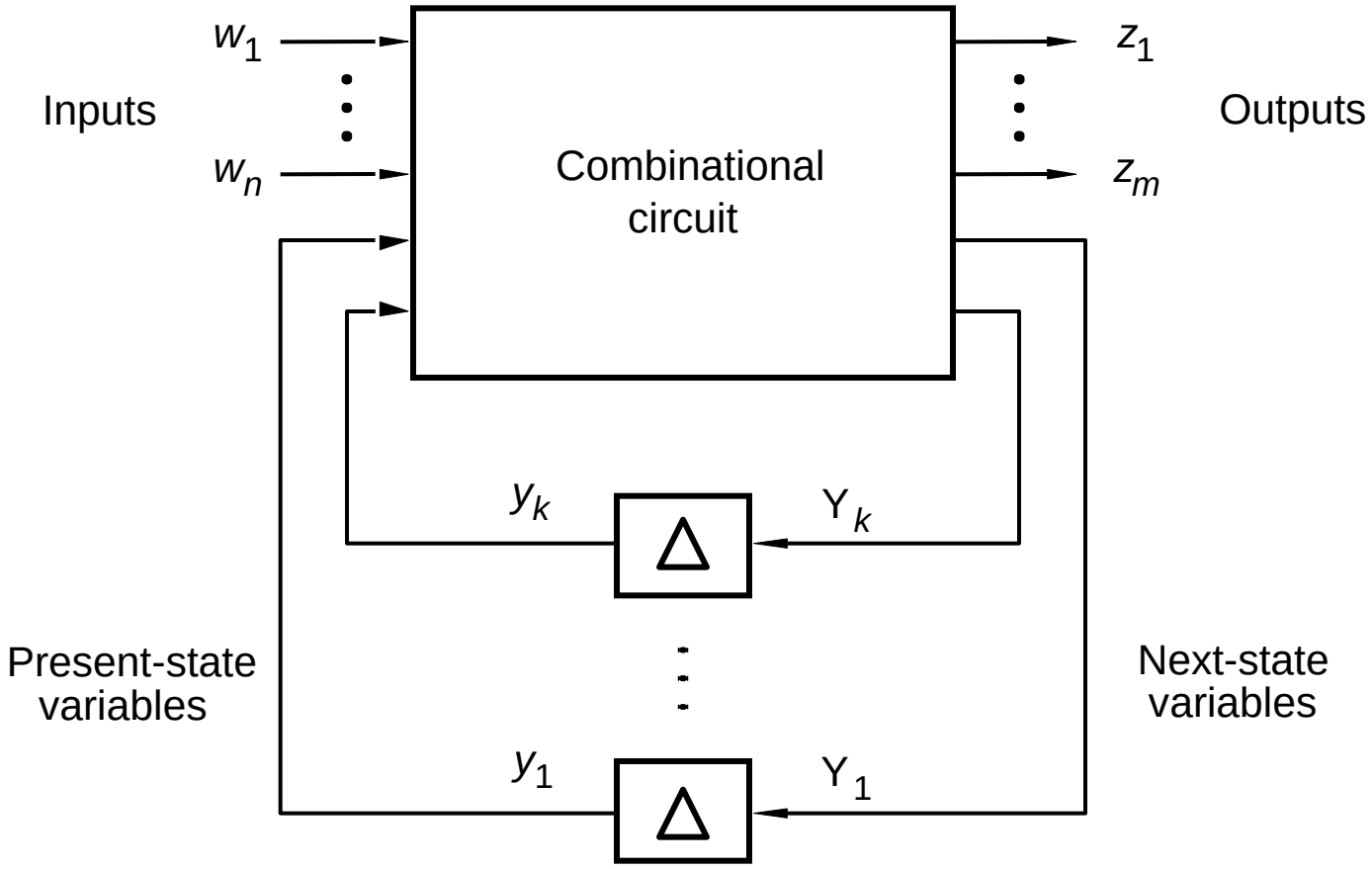


Bara vid tillståndsövergången
B-till-B med w = 1 har z värdet 1

Tillståndsautomater
Formell Model

- För att behandla tillståndsmaskiner matematisk behöver man en formell model
- Följande model kan beskriva både Moore- och Mealy-automaten

Tillståndsautomater
Formell Model



Tillståndsautomater
Formell Model

- En tillståndsmaskin kan formell definieras med

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

- W, Z, och Y beskriver ingångarna, utgångarna och tillsånd
- φ beskriver tillståndsövergångsfunktionen

$$S(t+1) = \varphi[W(t), S(t)]$$

- λ beskriver utgångsfunktion

$$Z = \lambda[W(t), S(t)] \quad \text{Mealy maskin}$$

$$Z = \lambda[S(t)] \quad \text{Moore maskin}$$

Formell modell för tillståndsautomat

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

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

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

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

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

Tillståndsmaskiner

- Överblivna tillstånd
- Tillståndsminimering
- ASM-chart
- Formell modell