IE1205 Digital Design:

F12: Asynkrona Sekvensnät (Del 1)

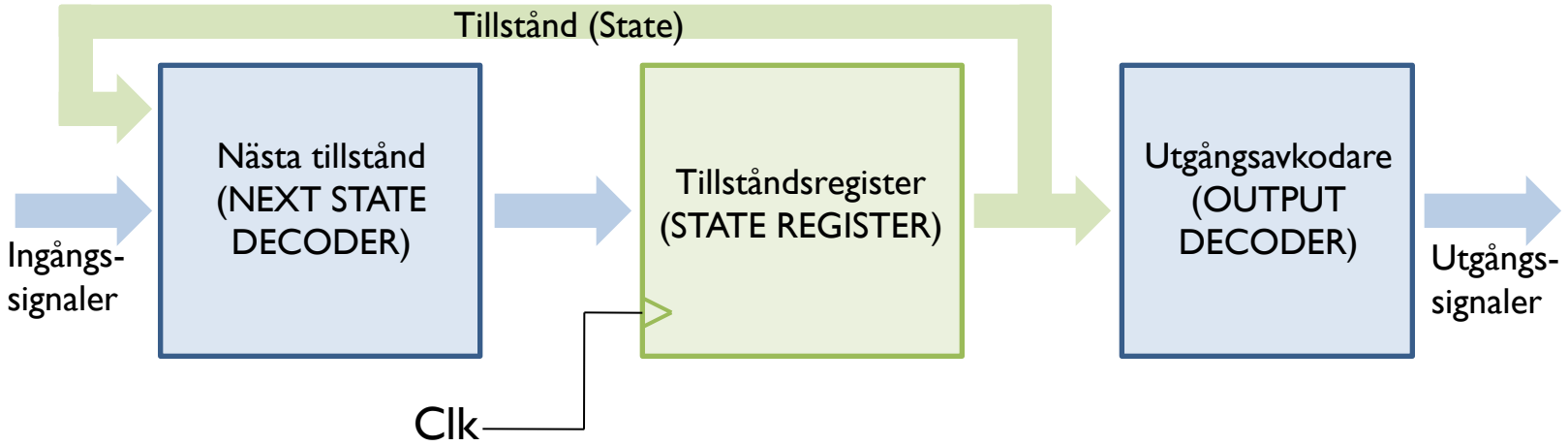
Asynkrona Sekvensmaskiner

- En asynkron sekvensmaskin är en sekvensmaskin utan vippor
- Asynkrona sekvensmaskiner bygger på en analys av återkopplade kombinatoriska grindnätverk
- **<u>Antagandet</u>: Endast EN signal i taget i grindnätet kan förändra sitt värde vid någon tidpunkt**

Asynkrona tillståndsmaskiner

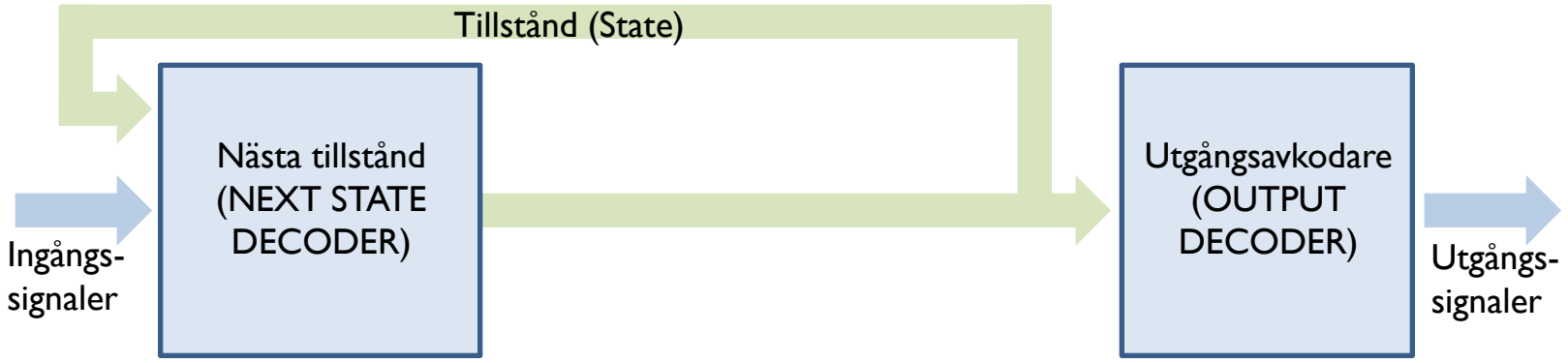
- Asynkrona tillståndsmaskiner används då det är nödvändigt att bibehålla ett tillstånd, men då det inte finns någon klocka tillgänglig.
  - Alla vippor och latchar är asynkrona tillståndsmaskiner
  - Användbara för att synkronisera händelser i situationer där metastabilitet är/kan vara ett problem

Synkron Moore-Automat



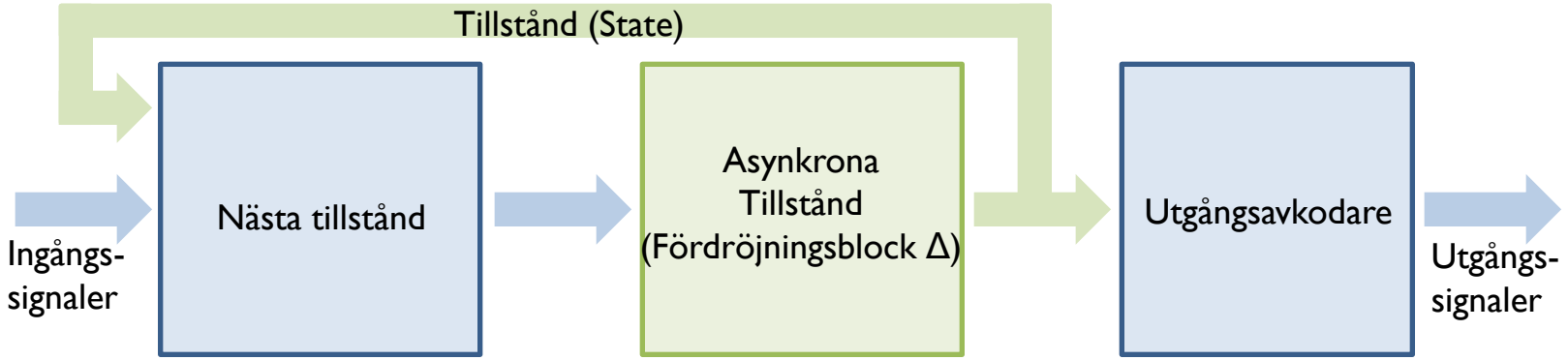
- I en synkron Moore-automat klockas tillståndsregistret av en klocka

Asynkron Moore-Automat



- Asynkrona sekvensnät har liknande uppbyggnad som synkrona sekvensnät, men inte några klockade vippor
- Kräver att vi har en fördröjning i nästa tillstånds kodaren (vilket i praktiken aldrig är något problem)

Asynkrona tillståndsmaskiner
Moore-kompatibel automat



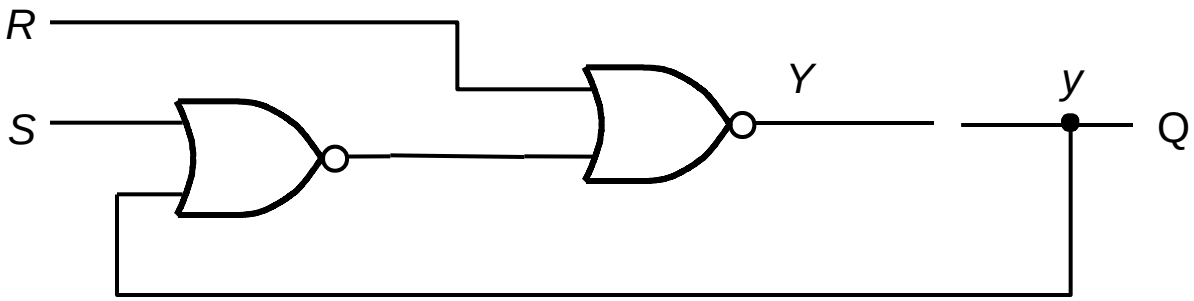
- För att förenkla analysen och kunna teckna ekvationer, gruppera alla grindfördröjningar i ett block
- I en verklig implementering utnyttjar vi grindarna i fördröjning

Gyllene regeln

Asynkron sekvenskrets
SR-latch med NOR-grindar

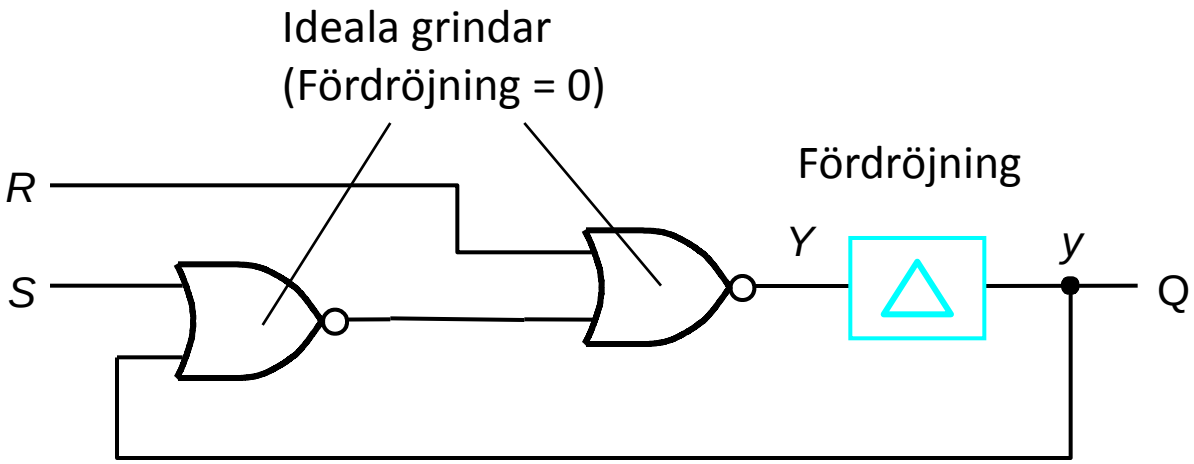
”klipp upp kretsen”

- y – Nuvarande tillstånd
- Y – Nästa tillstånd



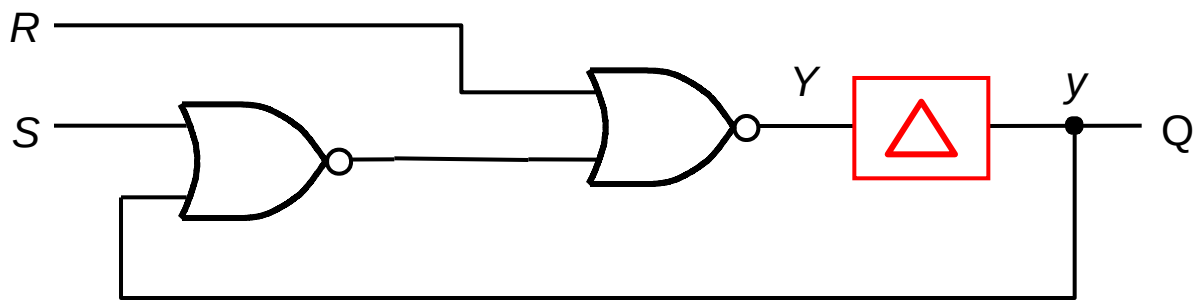
Asynkron sekvenskrets
SR-latch med NOR-grindar

- För att analyserar beteendet av en asynkron krets så anta man ideala grindar och sammanfatta delayn till en enda block med fördröjningen Δ.



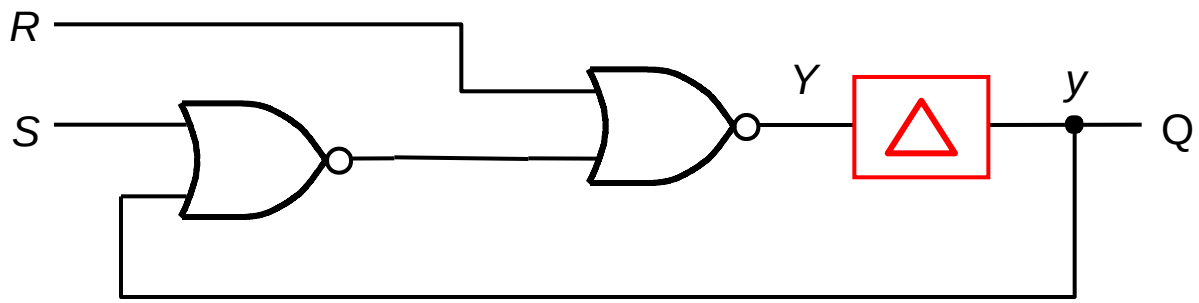
Analys av det asynkrona sekvensnätet

- Genom att vi har en fördröjningsblock kan vi betrakta
  - y som nuvarande tillstånd
  - Y som nästa tillstånd



Tillståndstabell

- Därmed kan vi ta fram tillståndstabellen där nästa tillstånd Y beror på insignalerna och nuvarande tillstånd y



$$Y = \overline{R + \overline{(S + y)}}$$

Tillståndstabell

Present state y	Next state			
	SR = 00	01	10	11
	Y	Y	Y	Y
0	0	0	1	0
1	1	0	1	0

$$Y = R + \overline{(S + y)}$$

Tillståndstabell

BV använder binärkodsordning

Från tillståndsfunktion till sanningstabell

$$Y = \overline{R + \overline{(S + y)}}$$

y	S	R	$Y = \overline{R + \overline{(S + y)}}$
0	0	0	$0 = \overline{0 + \overline{(0 + 0)}}$
0	0	1	$0 = \overline{1 + \overline{(0 + 0)}}$
0	1	0	$1 = \overline{1 + \overline{(1 + 0)}}$
0	1	1	$0 = \overline{1 + \overline{(1 + 0)}}$
1	0	0	$1 = \overline{0 + \overline{(0 + 1)}}$
1	0	1	$0 = \overline{1 + \overline{(0 + 1)}}$
1	1	0	$1 = \overline{0 + \overline{(1 + 1)}}$
1	1	1	$0 = \overline{1 + \overline{(1 + 1)}}$

Present state y	*Next state*			
	SR = 00	*01*	*10*	*11*
	Y	*Y*	*Y*	*Y*
0	*0*	*0*	*1*	*0*
1	*1*	*0*	*1*	*0*

Eller som på övningen – med hjälp av Karnaughdiagram …

Stabila tillstånd

Present state y	Next state			
SR =	00	01	10	11
	Y	Y	Y	Y
0	0	0	1	0
1	1	0	1	0

- Eftersom vi inte har vippor utan bara kombinatoriska kretsar kan en tillståndsändring medföra ytterligare tillståndsändringar
- En tillstånd är
  - stabil om Y(t) = y(t + Δ)
  - instabil om Y(t) ≠ y(t + Δ)

$$\boxed{Y = y}$$ stabilt

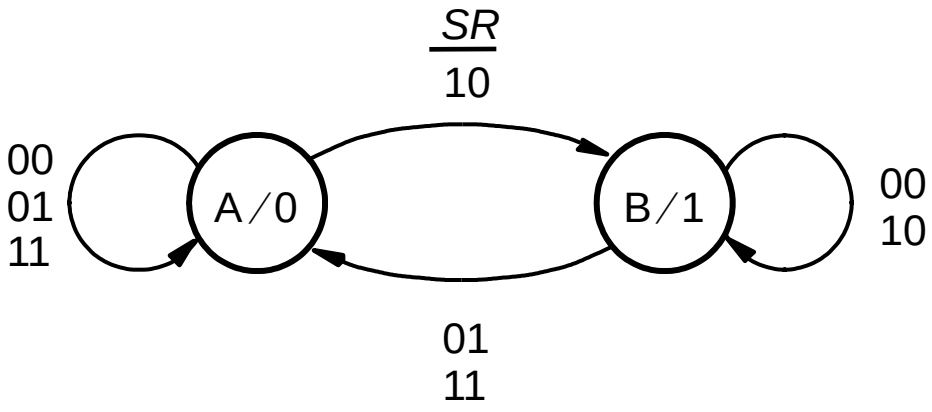
Den asynkrona tillståndstabellen Excitationstabellen

- Den asynkrona tillståndstabellen kallas för *Excitationstabell*
- Stabila tillstånd (next state = present state) ringas in

Present state y	Next state			
	SR = 00	01	10	11
	Y	Y	Y	Y
0	(0)	(0)	1	(0)
1	(1)	0	(1)	0

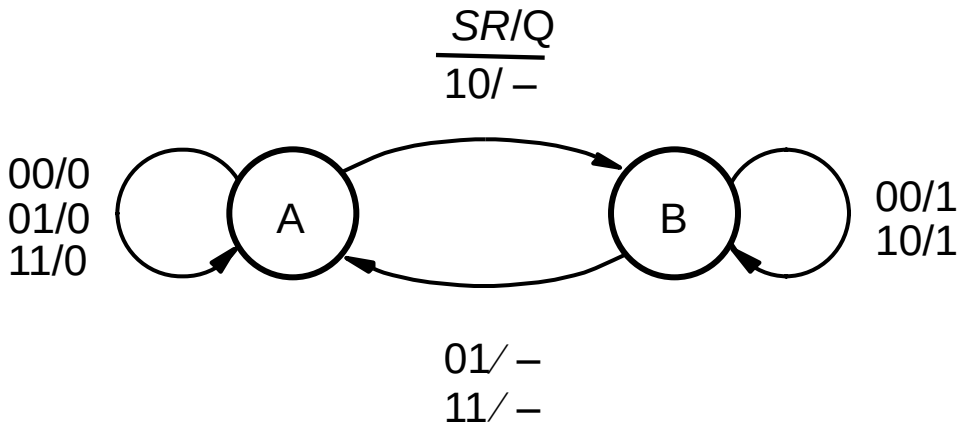
Flödestabell och Tillståndsdiagram (Moore)

Present state	Next state				Output Q
	SR = 00	01	10	11	
A	(A)	(A)	B	(A)	0
B	(B)	A	(B)	A	1



Flödestabell och Tillståndsdiagram (Mealy)

Present state	Next state				Output, Q			
	SR = 00	01	10	11	00	01	10	11
A	(A)	(A)	B	(A)	0	0	–	0
B	(B)	A	(B)	A	1	–	1	–

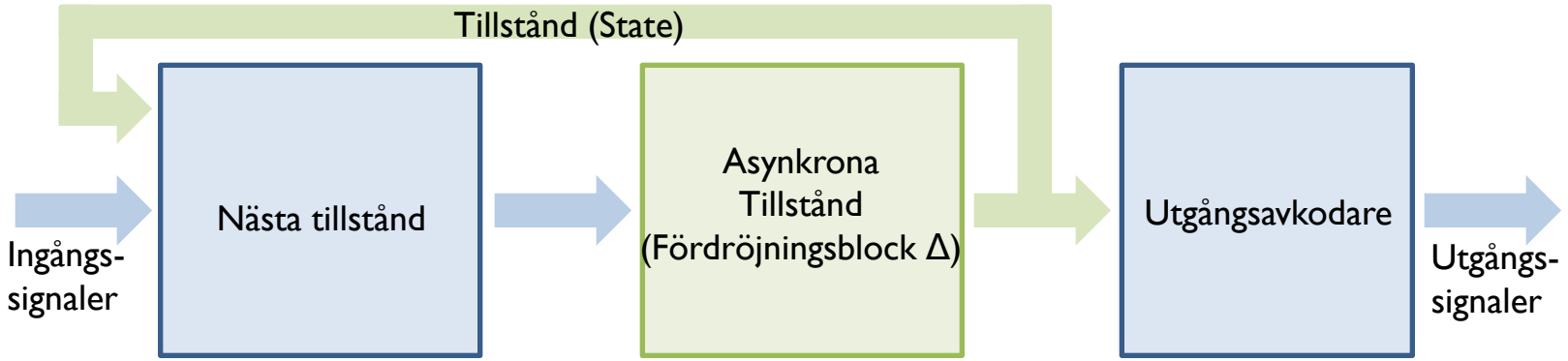


Don’t care (‘-’) har valts för utgångsavkodaren, eftersom utgången ändras direkt efter tillståndsövergången (enklare implementering)

Terminologi

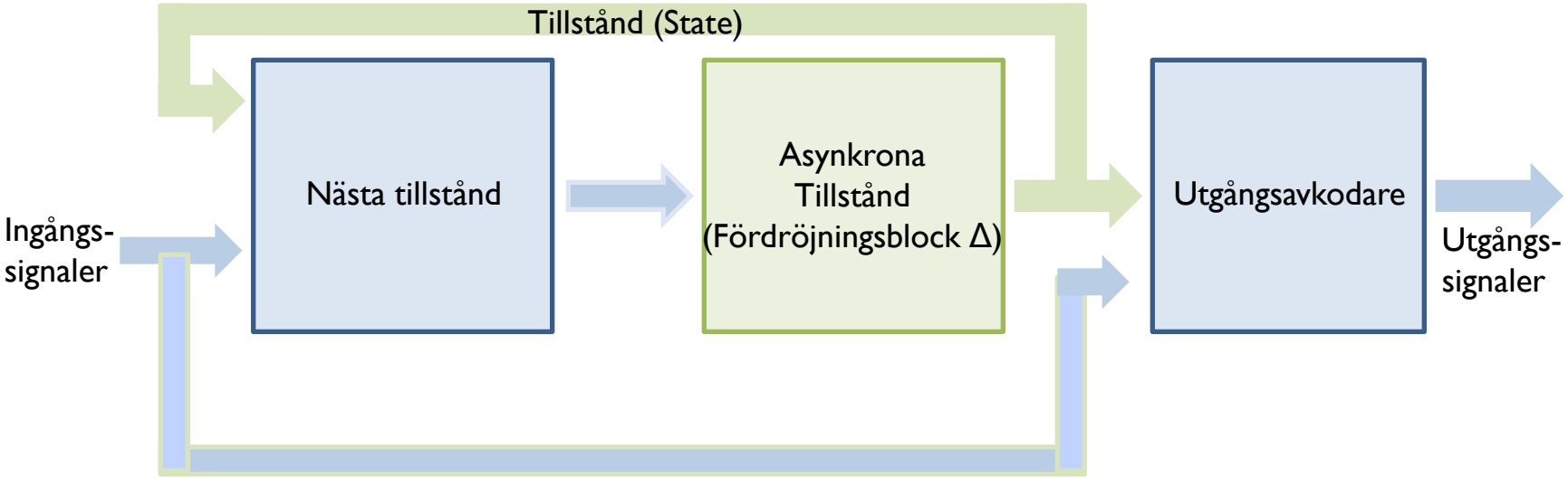
- När man arbetar med asynkrona sekvensnät så används det en annan terminologi
  - Den asynkrona tillståndstabellen kallas *flödestabell*

Asynkrona tillståndsmaskiner
Moore-kompatibel automat



- Asynkrona sekvensnät har liknande uppbyggnad som synkrona sekvensnät
- I stället för vipppor har man fördröjningsblock

Asynkrona tillståndsmaskiner
Mealy-kompatibel automat

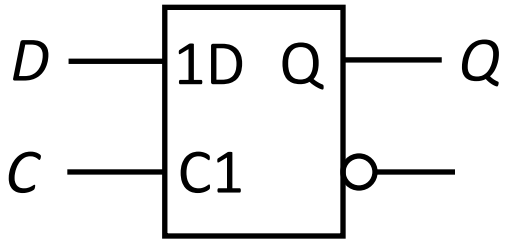


- Asynkrona sekvensnät har liknande uppbyggnad som synkrona sekvensnät
- I stället för vippor har man fördröjningsblock

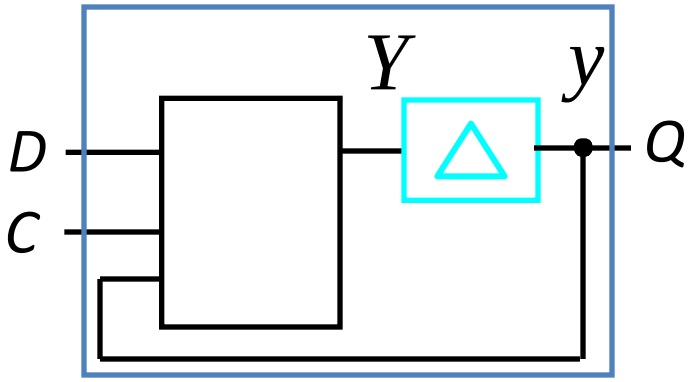
Analys av Asynkrona kretsar

- Analysen görs i följande steg:
  1) Ersätt återkopplingar i kretsen med ett delay-element Δ_i. Insignalen till delay-elementet bildar nästa tillstånd (next state) signalen Y_i, medan utsignalen y_i representerar nuvarande tillstånd (present state).
  2) Ta reda på next-state och output uttryck
  3) Ställ upp motsvarande excitationstabell
  4) Skapa en flödestabell och byt ut kodade tillstånd mot symboliska
  5) Rita ett tillståndsdiagram om så behövs

Först: D-latchens tillståndsfunktion



$$C = follow / \overline{latch}$$



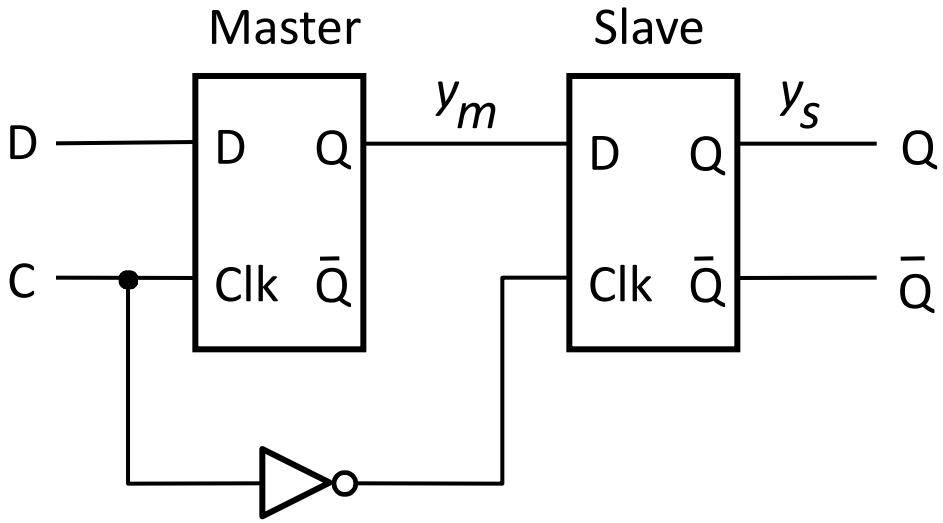
D-latchens tillståndsfunktion. Funktionssambandet mellan nuvarande tillstånd *y* och nästa tillstånd *Y*

$$Y = D \cdot C + y \cdot \overline{C}$$

$follow$ $\overline{latch}$

Exempel: Master-slave D-Vippa

- Master-slave D-vippan är konstruerad mha två asynkrona kretsar: D-latchar



Ekvationerna för nästa tillstånd:

$$Y_m = CD + \overline{C} y_m$$

$$Y_s = \overline{C} y_m + C y_s$$

Excitationstabell

- Ur ekvationerna kan man direkt härleda excitationstabellen

$$Y_m = CD + \overline{C} y_m$$
$$Y_s = \overline{C} y_m + C y_s$$

Present state $y_m\ y_s$	Next state $Y_m\ Y_s$				Output Q
	CD = 00	01	10	11	
00	(00)	(00)	(00)	10	0
01	00	00	(01)	11	1
10	11	11	00	(10)	0
11	(11)	(11)	01	(11)	1

Flödestabell

Vi definierar fyra tillstånd S1, S2, S3, S4 och erhåller då flödestabellen

Present state $y_m\ y_s$	*Next state* CD = 00	01	10	11	*Output* Q
	$Y_m\ Y_s$				
00	(00)	(00)	(00)	10	*0*
01	00	00	(01)	11	*1*
10	11	11	00	(10)	*0*
11	(11)	(11)	01	(11)	*1*

Present state	Nextstate CD = 00	01	10	11	Output Q
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	S1	(S2)	S4	1
S3	S4	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

Flödestabell

Present state	Next state				Output Q
	CD = 00	01	10	11	
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	S1	(S2)	S4	1
S3	S4	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

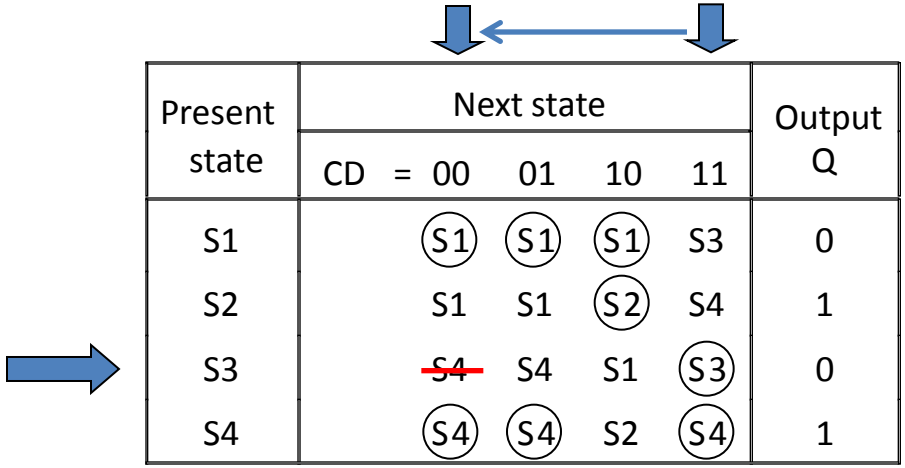
- Kom ihåg: Bara en ingång kan ändras samtidigt
- Därmed kan vissa övergångar aldrig inträffa!

Flödestabell (omöjliga övergångar)

Present state	Next state				Output Q
	CD = 00	01	10	11	
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	~~S1~~	(S2)	S4	1
S3	~~S4~~	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

- Tillstånd S3
  - Enda stabila tillstånd: S3 med ingångskombination 11
  - Bara en ingång kan ändras => möjliga ändringar (11 => 01, 11 => 10)
    - Dessa kombinationer lämnar S3!
    - Ingångskombinationen 00 i S2 är inte möjligt!
    - Ingångskombinationen 00 sätts till don't care!

Flödestabell – omöjliga övergångar



Present state	Next state CD = 00	01	10	11	Output Q
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	S1	(S2)	S4	1
S3	~~S4~~	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

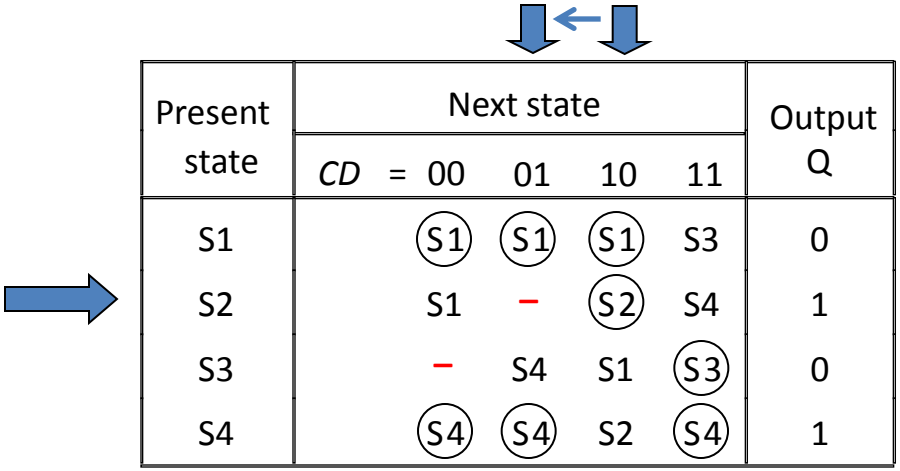
Tillstånd S3

Enda stabila tillståndet för **S3** är när ingångskombinationen är 11

Bara en ingång kan ändras → möjliga ändringar är 11 → 01, 11 → 10

- Dessa kombinationer lämnar S3!
- Ingångskombinationen 00 i S3 är *inte* möjligt!
- Ingångskombinationen 00 sätts därför till **don't care**!

Flödestabell – omöjliga övergångar



Present state	Next state				Output Q
	CD = 00	01	10	11	
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	–	(S2)	S4	1
S3	–	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

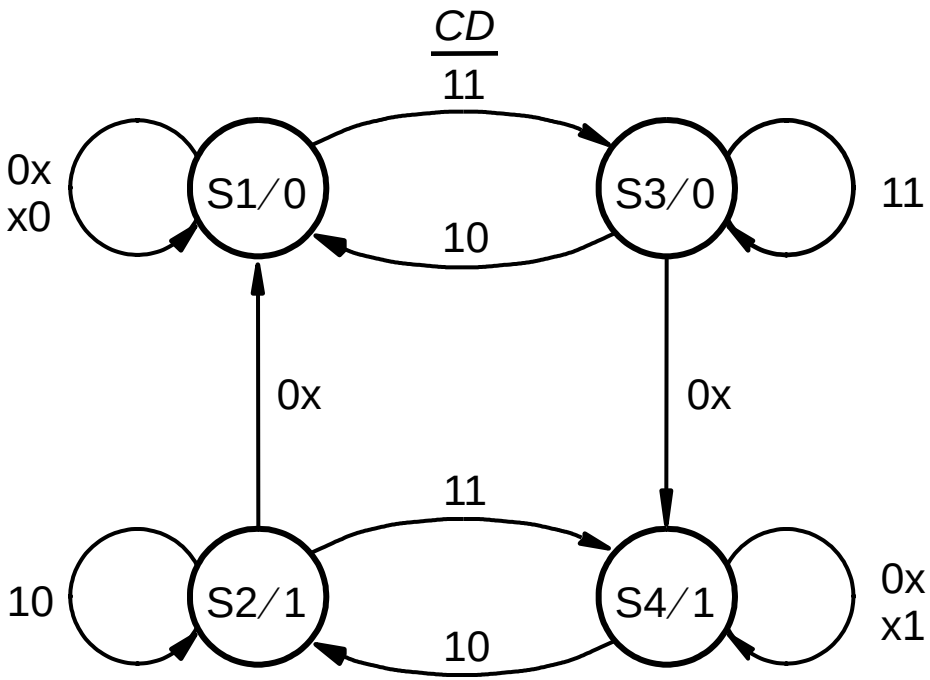
Tillstånd S2

Enda stabila tillståndet för **S2** är när ingångskombinationen är 10

Bara en ingång kan ändras → möjliga ändringar är 10 → 11, 10 → 00

- Dessa kombinationer lämnar S2!
- Ingångskombinationen 01 i S2 är *inte* möjligt!
- Ingångskombinationen 01 sätts därför till **don't care**!

Tillståndsdiagram D-vippan



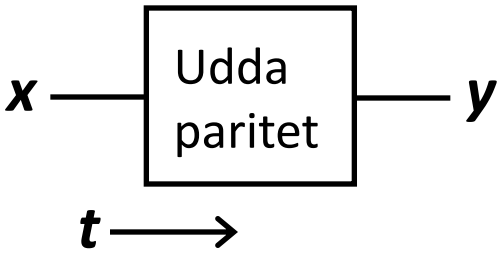
Syntes av asynkrona kretsar

- Syntesen genomförs i följande steg:
  1) Skapa ett tillståndsdiagram enligt funktionsbeskrivningen
  2) Skapa en flödestabell och reducera antalet tillstånd om möjligt
  3) Tilldela koder till tillstånden och skapa excitationstabellen
  4) Ta fram uttryck (överföringsfunktioner) för nästa tillstånd samt utgångar
  5) Konstruera en krets som implementerar ovanstående uttryck

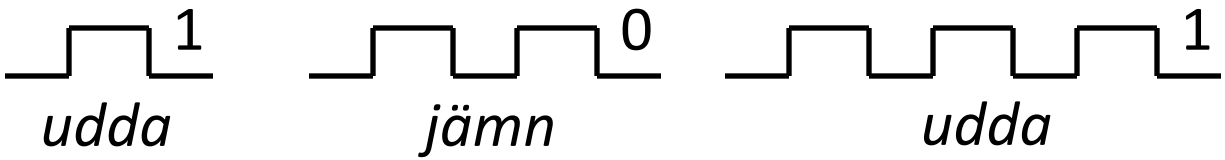
Exempel: seriell paritetskrets

Ingång ***x*** Utgång ***y***

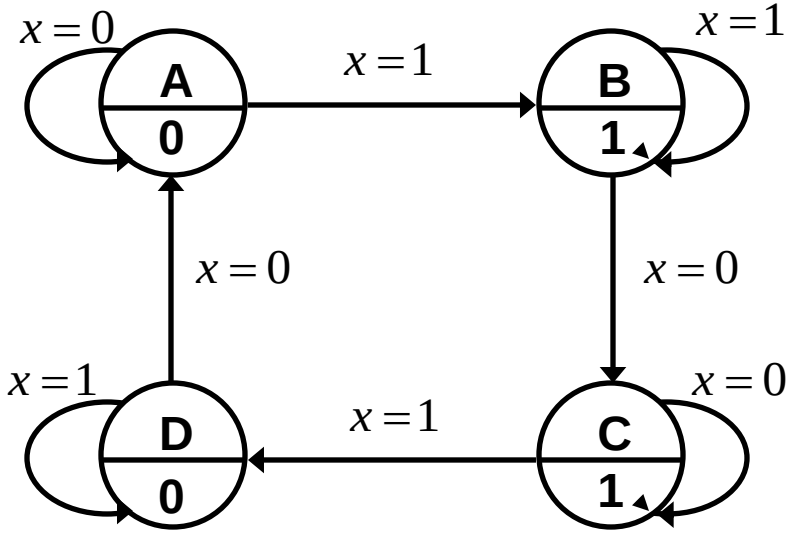
y = 1 om antalet pulser på ingången ***x*** har varit udda.

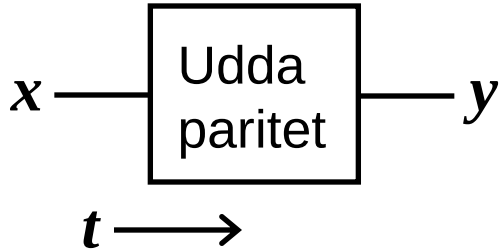


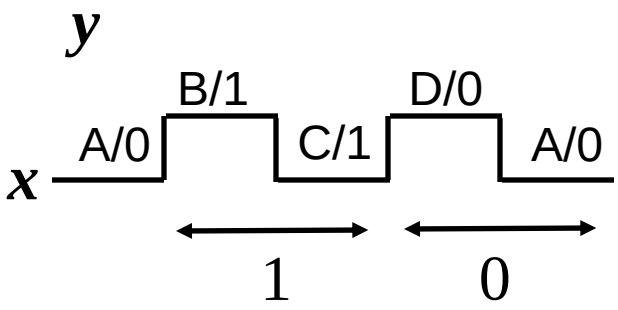
Med andra ord en ”varannangång” krets …



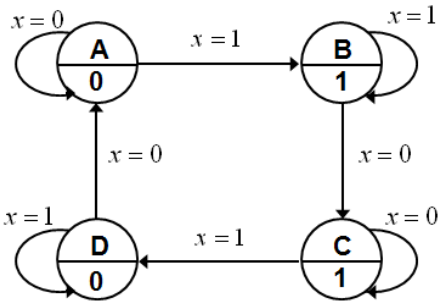
Skapa tillståndsdiagram

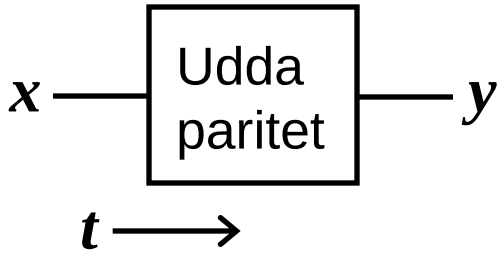






Skapa flödestabellen





Pres state	Next State		Q
	X=0	1	
A	(A)	B	0
B	C	(B)	1
C	(C)	D	1
D	A	(D)	0

Vad är bra tillståndskod?

00, 01, 10, 11 - binärkod

Pres state	Next State		Q
	X=0	1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	10	01	1
10	10	11	1
11	00	11	0

Dålig kodning (HD=2!)

- *Antag*

$X = 1 \quad Y_2Y_1 = 11$

- *därefter*

$X \rightarrow 0 \rightarrow Y_2Y_1 = 00?$

$11 \rightarrow 10!$

$11 \rightarrow 01 \rightarrow 10! \quad ? \rightarrow 00$

Vi når aldrig 00?

Vad är bra tillståndskod?

00, 01, 11, 10 - graykod

- *Antag*

$X = 1 \quad Y_2Y_1 = 10$

- *därefter*

$X \rightarrow 0 \rightarrow Y_2Y_1 = 00$

$10 \rightarrow 00$

Pres state	Next State		Q
	X=0	1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0

Bra kodning (HD=1)

Tillståndskodning

- I asynkrona sekvensnät är det omöjligt att garantera att två tillståndsvariabler ändrar värdet samtidigt
  - Därmed kan en övergång 00 $\rightarrow$ 11 resultera i
    - en övergång 00 $\rightarrow$ 01 $\rightarrow$???
    - en övergång 00 $\rightarrow$ 10 $\rightarrow$???
- För att säkerställa funktionen **MÅSTE** alla tillståndsövergångar ha ***Hamming distansen* 1**
  - Hamming distansen är antalet bitar som skiljer sig i två binära tal
    - Hamming distansen mellan 00 och 11 är 2
    - Hamming distansen mellan 00 och 01 är 1

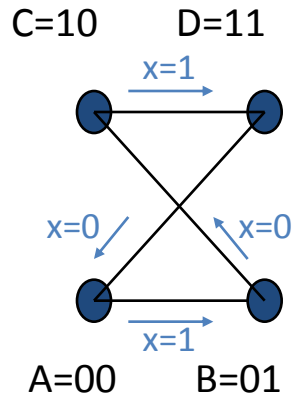
Richard Hamming

Tillståndskodning

- Procedur för att erhålla bra koder:
  1) Rita transitionsdiagram längs kanterna i hyperkuben som bildas av koderna
  2) Ta bort ev korsande linjer genom att
     a) byta plats på två närliggande noder
     b) utnyttja tillgängliga icke använda koder (utnyttja instabila tillstånd)
     c) introducera fler dimensioner i hyperkuben

Exempel: Serial Parity Generator

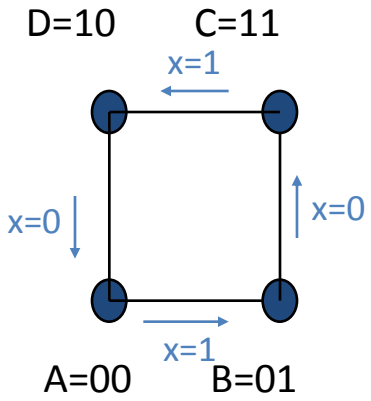
Pres state	Next State		Q
	X=0	1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	10	01	1
10	10	11	1
11	00	11	0



Dålig kodning –
Hamming Distance = 2
(korsande linjer)

Exempel: Serial Parity Generator

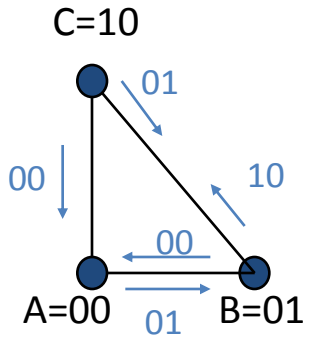
Pres state	Next State		Q
	X=0	1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0



Bra kodning
Hamming Distance = 1
(inga korsande linjer)

Icke stabila tillstånd

Present state	Next state				Output
	$r_2 r_1$ = 00	01	10	11	$g_2 g_1$
A	(A)	B	C	–	00
B	A	(B)	C	(B)	01
C	A	B	(C)	(C)	10

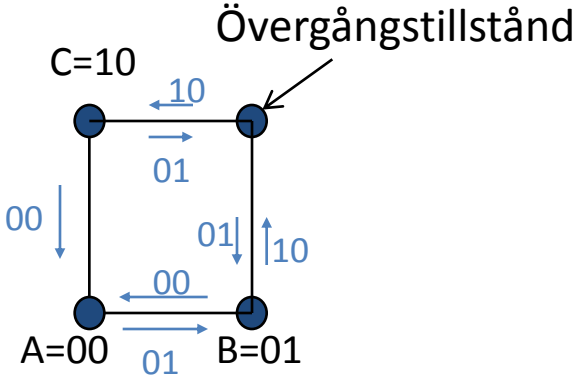


Dålig kodning

Vid övergången från B to C (eller C to B) är Hamming distansen 2!
Risk att man fastnar i ett ospecificerat tillstånd (med kod 11)!

Icke stabila tillstånd

- Lösning: Införandet av ett övergångstillstånd som säkerställa att man inte hamnar i ett odefinierat läge!

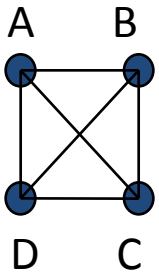


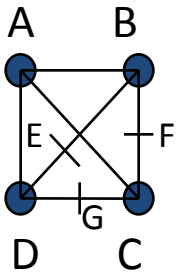
Bra kodning

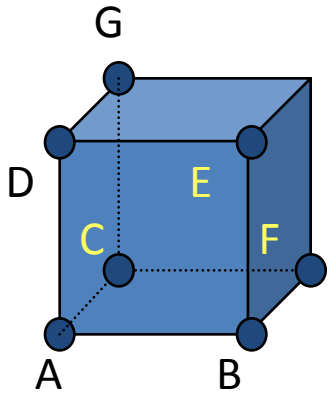
	Present state y_2y_1	Next state r_2r_1 = 00	01	11	10	Output g_2g_1
		Y_2Y_1				
A	00	(00)	01	–	10	00
B	01	00	(01)	(01)	11	01
D	11	–	01	–	10	dd
C	10	00	11	(10)	(10)	10

Extra tillstånd (fler dimensioner)

- Man kan öka antalet dimensioner för att kunna införa säkra tillståndsövergångar



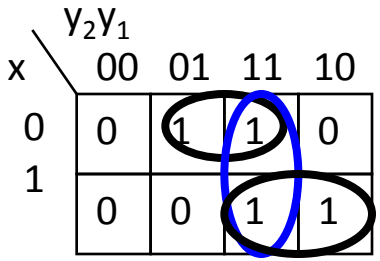




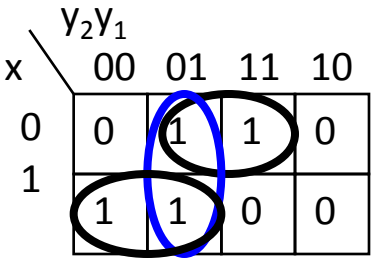
Om det inte på något sätt går att rita om diagrammet till HD=1 får man lägga till fler tillstånd genom att lägga till extra dimensioner. Man tar då närmsta större hyperkub och drar övergångarna genom tillgängliga icke stabila tillstånd.

Steg 4:
Skapa Karnaugh-diagram

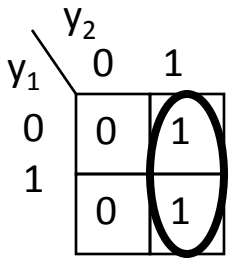
Pres state	Next State		Q
	X=0	1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0



$Y_2 = \bar{x}\, y_1 + y_2 y_1 + x\, y_2$



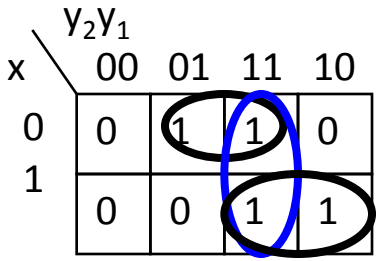
$Y_1 = x\, \bar{y_2} + \bar{y_2} y_1 + x\, \bar{y_1}$



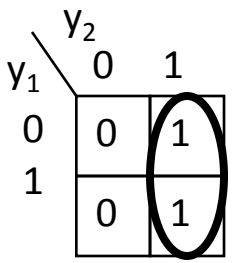
$Q = y_2$

De blå inringningarna är pga att undvika Hazard (se senare avsnitt)!

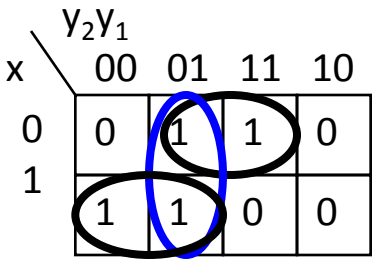
Färdig krets



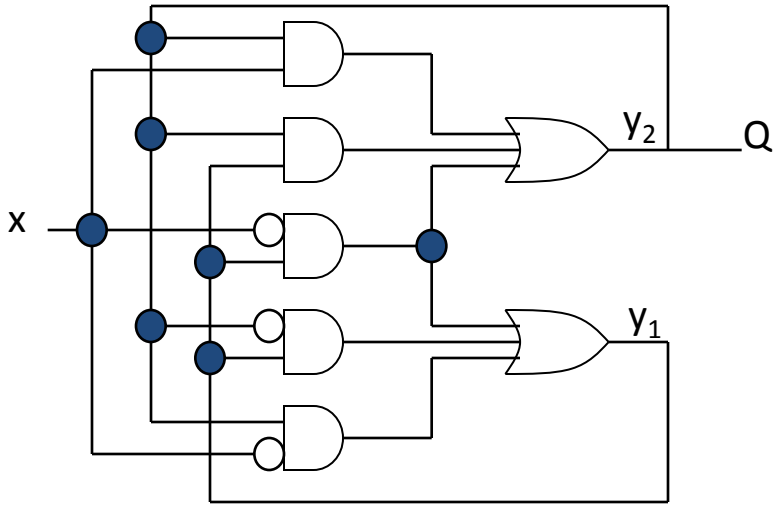
$Y_2 = \bar{x}\, y_1 + y_2 y_1 + x\, y_2$



$Q = y_2$



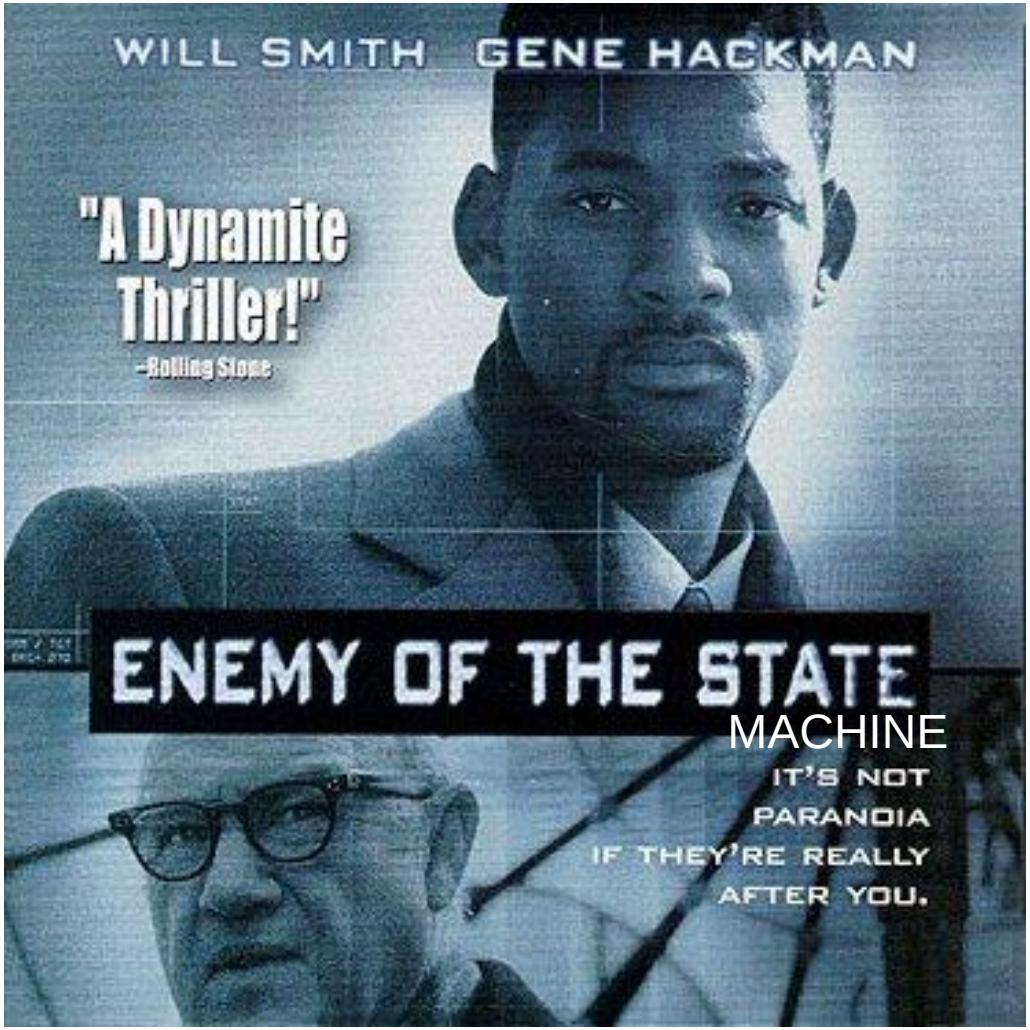
$Y_1 = x\, \bar{y}_2 + \bar{y}_2 y_1 + x\, \bar{y}_1$



Vad är Hazard?

- Hazard är ett begrepp som innebär att det finns en fara för att utgångsvärdet inte är stabilt, utan att det kan blinka till vid vissa ingångskombinationer.
- Hazard uppkommer om det är olika långt från olika ingångar till en utgång, s.k., signal-kapplöpning.
- För att motverka detta måste man lägga till prim-implikanter för att täcka upp den farliga övergången.

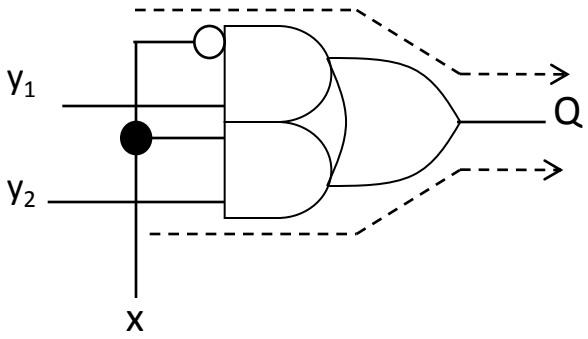
Hazard – Enemy of the state machine!

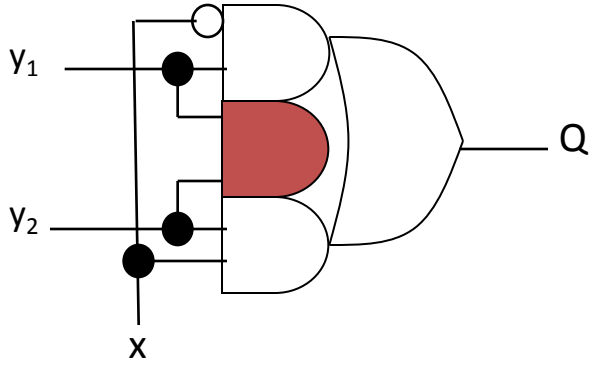


Exempel på Hazard: Muxen

x \ y_2y_1	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$Y_2 = \overline{x}\, y_1 + y_2 y_1 + x\, y_2$





Vid övergång från xy_2y_1=(111)->(011) kan utgången Q blinka till, eftersom vägen från x till Q är längre via den övre AND-grinden än den lägre (kapplöpning).

MER OM HAZARD I NÄSTA FÖRELÄSNING!

Steg 5. Färdig krets

x \ y_2y_1	00	01	11	10
0	0	1	1	0
1	0	0	1	1

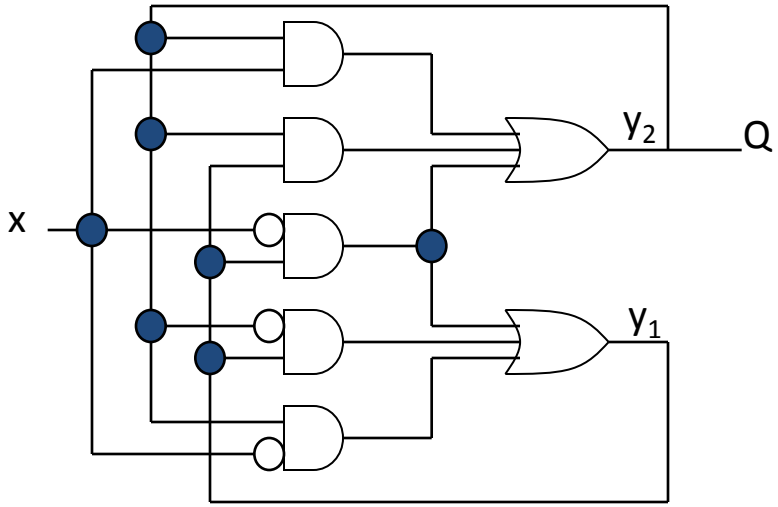
$Y_2 = \bar{x}\, y_1 + y_2 y_1 + x\, y_2$

y_1 \ y_2	0	1
0	0	1
1	0	1

$Q = y_2$

x \ y_2y_1	00	01	11	10
0	0	1	1	0
1	1	1	0	0

$Y_1 = x\, \bar{y}_2 + \bar{y}_2 y_1 + x\, \bar{y}_1$



Tillståndsminimering

Asynkrona statemaskiner har många ”ospecifierade” positioner i flödestabellen som man kan utnyttja för att minimera antalet tillstånd.

Sannolikheten för att färre tillstånd leder till en enklare realisering är* hög *när det gäller asynkrona nät!

Tillståndsminimering

Två steg:

Ekvivalens – ekvivalenta tillstånd. Samma steg som vid tillståndsminimering av synkrona sekvensnät, full flexibilitet finns kvar.

Kompatibilitet – kompatibla tillstånd blir olika för Moore-kompatibel eller Mealy-kompatibel realisering, de val man nu gör påverkar den fortsatta flexibiliteten.

Tillståndsminimering

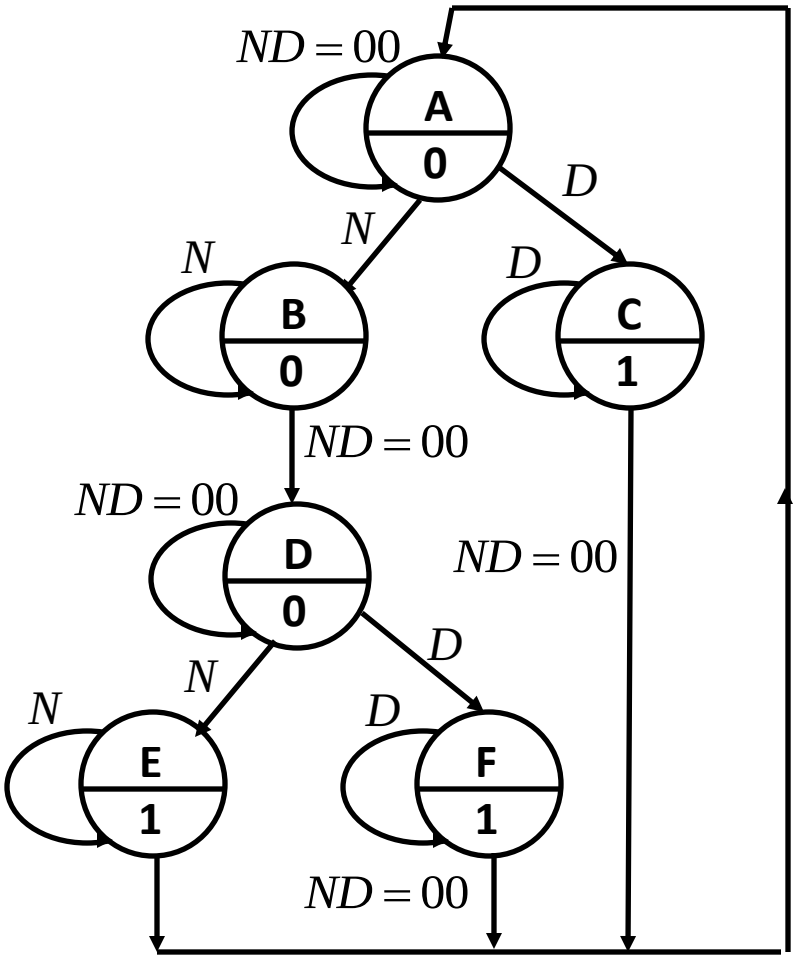
- **Procedur för minimering av antalet tillstånd**

1. Bilda **ekvivalensgrupper**.
   För att vara i samma grupp ska följande gälla:
   - Utgångar måste ha samma värde
   - Stabila tillstånd måste finnas på samma plats (kolumn)
   - Don’t cares för next state måste finnas på samma plats (kolumn)
2. Minimera ekvivalensgrupperna (state-reduktion)
3. Bilda **sammanslagningsdiagram**, olika för Mealy eller för Moore.
4. Slå ihop kompatibla tillstånd i grupper. Minimera samtidigt antalet grupper. Varje tillstånd får endast ingå i en grupp.
5. Konstruera den reducerade flödestabellen genom att slå samman raderna i de valda grupperna
6. Repetera steg 3-5 för att se om fler minimeringar kan göras

Godisautomat (BV sid 610)

- Godismaskinen har två ingångar:
  - ***N***: Nickel (5 cent)
  - ***D***: Dime (10 cent)
- En godisbit kostar 10 cent
- Maskinen returnerar inga pengar om det finns 15 cent i automaten (en godisbit returneras)
- Utgången ***z*** är aktiv när det finns tillräckligt med pengar för en godisbit

Tillståndsdiagram, Flödestabell



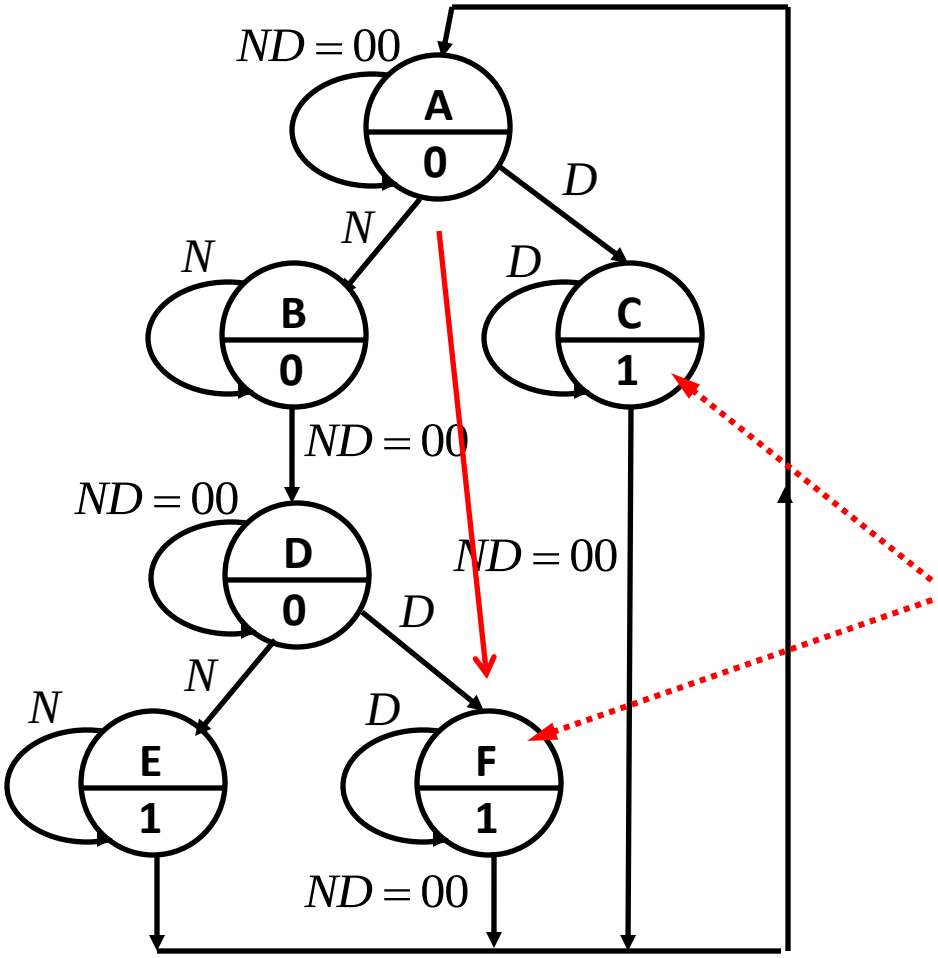
- ***Inga ”dubbeländringar” av insignalerna!***
- ***Två mynt går inte att stoppa i samtidigt!***

Pres state	Next State X=00	01	10	11	Q
A	(A)	B	C	-	0
B	D	(B)	-	-	0
C	A	-	(C)	-	1
D	(D)	E	F	-	0
E	A	(E)	-	-	1
F	A	-	(F)	-	1

(X = ND, Q = z)

En flödestabell som endast innehåller ett stabilt tillstånd per rad kallas för en ***primitiv flödestabell***.

Tillståndsminimering



Tillståndsminimering innebär att **två** tillstånd kan vara ekvivalenta, och i så fall ersättas av **ett** tillstånd för att förenkla tillståndsdiagrammet, och nätet.

Man kan lätt inse att tillstånd **C** och **F** kommer att kunna ersättas med **ett** tillstånd eftersom godis *alltid* ska matas ut efter en Dime oavsett tidigare tillstånd.

Bilda/minimera ekvivalensgrupper

1. **Bilda ekvivalensgrupper. För att vara i samma grupp ska följande gäller:**
    - Utgångar måste ha samma värde
    - Stabila tillstånd måste finnas på samma plats (kolumn)
    - Don’t cares för next state måste finnas på samma plats (kolumn)
2. **Minimera ekvivalensgrupperna (state reduction).**

Ekvivalensgrupper

Pres state	Next State X=00	01	10	11	Q
A	(A)	B	C	-	0
B	D	(B)	-	-	0
C	A	-	(C)	-	1
D	(D)	E	F	-	0
E	A	(E)	-	-	1
F	A	-	(F)	-	1

(*X = ND*, *Q = z*)

Tillstånden delas i block efter utsignal.
ABD har utsignal **0**, **CEF** har utsignal **1**.
P_1 = (ABD)(CEF)

Stabila tillstånd måste finnas för samma insignal (kolumn), don't care måste finnas för samma kolumn.

AD har stabilt tillstånd för 00. **B** har stabilt för 01. **CF** har stabilt tillstånd för 10. **E** har stabilt för 01. **AD** och **CF** har don't care för motsvarande insignaler.

P_2 = (AD)(B)(CF)(E)

Slå ihop ekvivalensgrupper

*Två rader kan ”slås ihop” om det **inte** innebär någon **konflikt** för deras efterföljartillstånd*

Pres state	Next State X=00	01	10	11	Q
A	(A)	B	C	-	0
B	D	(B)	-	-	0
C	A	-	(C)	-	1
D	(D)	E	F	-	0
E	A	(E)	-	-	1
F	A	-	(F)	-	1

(X = ND, Q = z)

P_2=(AD)(B)(CF)(E)
P_3=(A)(D)(B)(C)(E)
P_4=P_3.

Raderna **C** och **F** kan slås ihop med ny samlingsbeteckning **C**, medan **A** och **D** som har efterföljare i olika grupper *inte* kan slås ihop.

$CF_{00} \rightarrow (A)$
$CF_{01} \rightarrow (--)$
$CF_{10} \rightarrow (CF)$
$CF_{11} \rightarrow (--)$

$AD_{00} \rightarrow (AD)$
$AD_{01} \rightarrow (B)(E)$
$AD_{10} \rightarrow (CF)$
$AD_{11} \rightarrow (--)$

Resulterande flödestabell

Pres state	Next State X=00	01	10	11	Q
A	(A)	B	C	-	0
B	D	(B)	-	-	0
C	A	-	(C)	-	1
D	(D)	E	C	-	0
E	A	(E)	-	-	1

Kompatibilitetsgrupper

3. Bilda sammanslagningsdiagram ***antingen*** för **Mealy** eller **Moore**
4. Slå ihop kompatibla tillstånd i grupper. Minimera samtidigt antalet grupper. Varje tillstånd får endast ingå i en grupp.
5. Konstruera den reducerade flödestabellen genom att slå samman raderna i de valda grupperna
6. Repetera steg 3-5 för att se om fler minimeringar kan göras

Sammanslagningsregler

- **Två tillstånd är ”kompatibla” och kan slås ihop om följande gäller**
  1. åtminstone ***ett*** av följande villkor gäller för alla ingångskombinationer
     - både S_i och S_j har samma följdtillstånd, eller
     - både S_i och S_j är stabila, eller
     - följdtillståndet av S_i eller S_j eller båda är ospecifierade
  2. Sedan gäller följande om man vill konstruera en Moore-kompatibel automat
     - både S_i och S_j har samma **utgångsvärde** (gäller ju inte när man konstruerar en Mealy-kompatibel automat)

Sammanslagningsdiagram

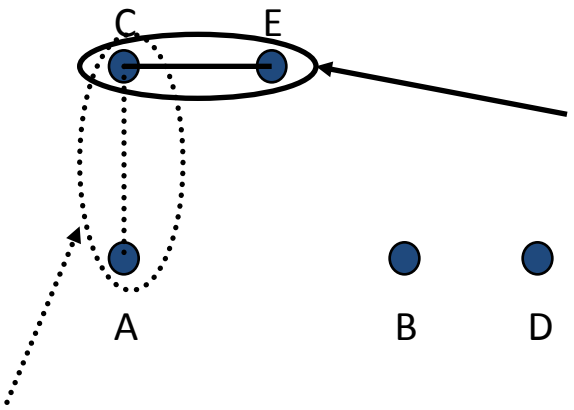
Resulterande flödestabell

Pres state	Next State X=00	01	10	11	Q
→ A	(A)	B	C	-	0
B	D	(B)	-	-	0
→ C	A	-	(C)	-	1
D	(D)	E	C	-	0
→ E	A	(E)	-	-	1

Varje rad blir en punkt i kompatibilitetsgrafen.

- *När det finns flera möjligheter …*

Kompatibilitetsgraf



Moore-kompatibla

C(1):	A–**C**–
E(1):	A**E**––

Mealy-kompatibla: I tillstånd **A** (X = 00) är utgången 0, i tillstånd **C** är utgången 1

C(1):	A–**C**–
A(0):	**A**BC–

Ett illustrativt exempel (BV 9.8)

Primitiv flödestabell

Pres state	Next State X=00	01	10	11	Q
A	(A)	F	C	-	0
B	A	(B)	-	H	1
C	G	-	(C)	D	0
D	-	F	-	(D)	1
E	G	-	(E)	D	1
F	-	(F)	-	K	0
G	(G)	B	J	-	0
H	-	L	E	(H)	1
J	G	-	(J)	-	0
K	-	B	E	(K)	1
L	A	(L)	-	K	1

Ekvivalensklasser

Samma utsignal, samma position för stabilt tillstånd och för don't care tillstånd (AG) (BL) (HK)

$\mathbf{P_1}$= (AG)(BL)(C)(D)(E)(F)(HK)(J)

Följdtillstånd:

$AG_{00} \rightarrow (AG)$ $AG_{01} \rightarrow (F)(B)$
$AG_{10} \rightarrow (C)(J)$ $AG_{11} \rightarrow (-)$

A, G är *inte* ekvivalenta

$BL_{00} \rightarrow (A)$ $BL_{01} \rightarrow (BL)$
$BL_{10} \rightarrow (--)$ $BL_{11} \rightarrow (HK)$

$HK_{00} \rightarrow (--)$ $HK_{01} \rightarrow (BL)$
$HK_{10} \rightarrow (E)$ $HK_{11} \rightarrow (HK)$

$\mathbf{P_2}$= (A)(G)(BL)(C)(D)(E)(F)(HK)(J) $\mathbf{P_3}$=P_2

Ett illustrativt exempel (BV 9.8)

Primitiv flödestabell

Pres state	Next State X=00	01	10	11	Q
A	(A)	F	C	-	0
B	A	(B)	-	H	1
C	G	-	(C)	D	0
D	-	F	-	(D)	1
E	G	-	(E)	D	1
F	-	(F)	-	K	0
G	(G)	B	J	-	0
H	-	L	E	(H)	1
J	G	-	(J)	-	0
K	-	B	E	(K)	1
L	A	(L)	-	K	1

Ekvivalens-klasser

$\mathbf{P_1}$= (AG)(BL)(C)(D)(E)(F)(HK)(J)
$\mathbf{P_2}$= (A)(G)(**BL**)(C)(D)(E)(F)(**HK**)(J)
$\mathbf{P_3}$=P_2

B för (**BL**)
H för (**HK**)

Inga ospecifierade tillstånd har ***ännu*** *utnyttjats!*

Reducerad flödestabell

Pres state	Next State X=00	01	10	11	Q
A	(A)	F	C	-	0
B	A	(B)	-	H	1
C	G	-	(C)	D	0
D	-	F	-	(D)	1
E	G	-	(E)	D	1
F	-	(F)	-	H	0
G	(G)	B	J	-	0
H	-	B	E	(H)	1
J	G	-	(J)	-	0

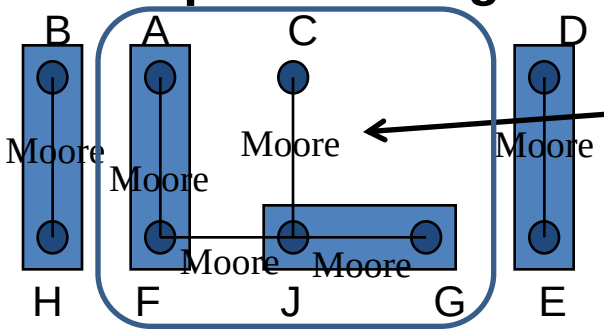
Ett illustrativt exempel …

Reducerad flödestabell

Pres state	Next State X=00	01	10	11	Q
A	(A)	F	C	-	0
B	A	(B)	-	H	1
C	G	-	(C)	D	0
D	-	F	-	(D)	1
E	G	-	(E)	D	1
F	-	(F)	-	H	0
G	(G)	B	J	-	0
H	-	B	E	(H)	1
J	G	-	(J)	-	0

Nya beteckningar **B** (**BH**), **A** (**AF**), **G** (**JG**), **D** (**DE**)

Kompatibilitets-graf



Olika val är tänkbara

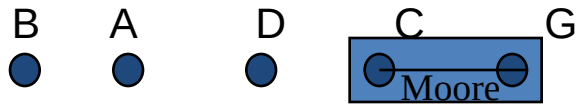
Pres state	Next State X=00	01	10	11	Q
A	A	A	C	B	0
B	A	B	D	B	1
C	G	-	C	D	0
D	G	A	D	D	1
G	G	B	G	-	0

Ett illustrativt exempel …

Mer reducerad flödestabell

Pres state	Next State X=00	01	10	11	Q
A	Ⓐ	Ⓐ	C	B	0
B	A	Ⓑ	D	Ⓑ	1
C	G	-	Ⓒ	D	0
D	G	A	Ⓓ	Ⓓ	1
G	Ⓖ	B	Ⓖ	-	0

Kompatibilitets-graf



Slutlig flödestabell

Pres state	Next State X=00	01	10	11	Q
A	Ⓐ	Ⓐ	C	B	0
B	A	Ⓑ	D	Ⓑ	1
C	Ⓒ	B	Ⓒ	D	0
D	C	A	Ⓓ	Ⓓ	1

Ny beteckning **C** för (**CG**)

Nu har alla ospecificerade tillstånd utnyttjats!

Sammanfattning

- Asynkrona tillståndsmaskiner
  - Bygger på analys av återkopplade kombinatoriska nät
  - Alla vippor och latchar är asynkrona tillståndsmaskiner
- En liknande teori som för synkrona tillståndsmaskiner kan appliceras
  - Bara en ingång eller tillståndsvariabel kan ändras åt gången!
  - Man får även ta hänsyn till kapplöpningsproblem