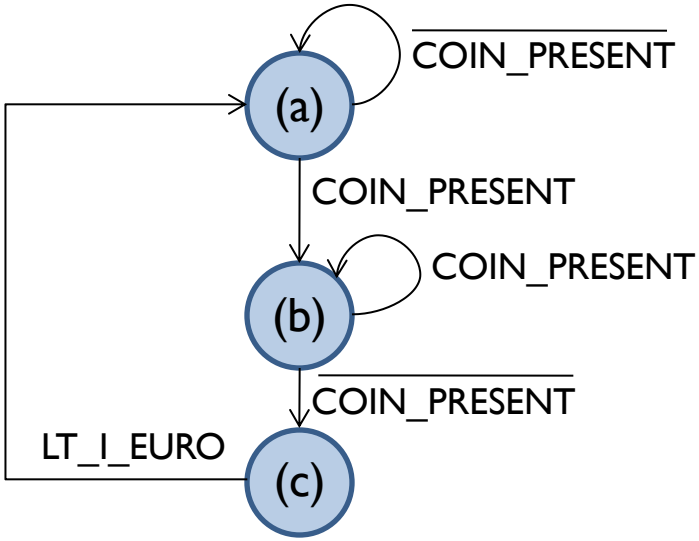
IE1205 Digital Design:

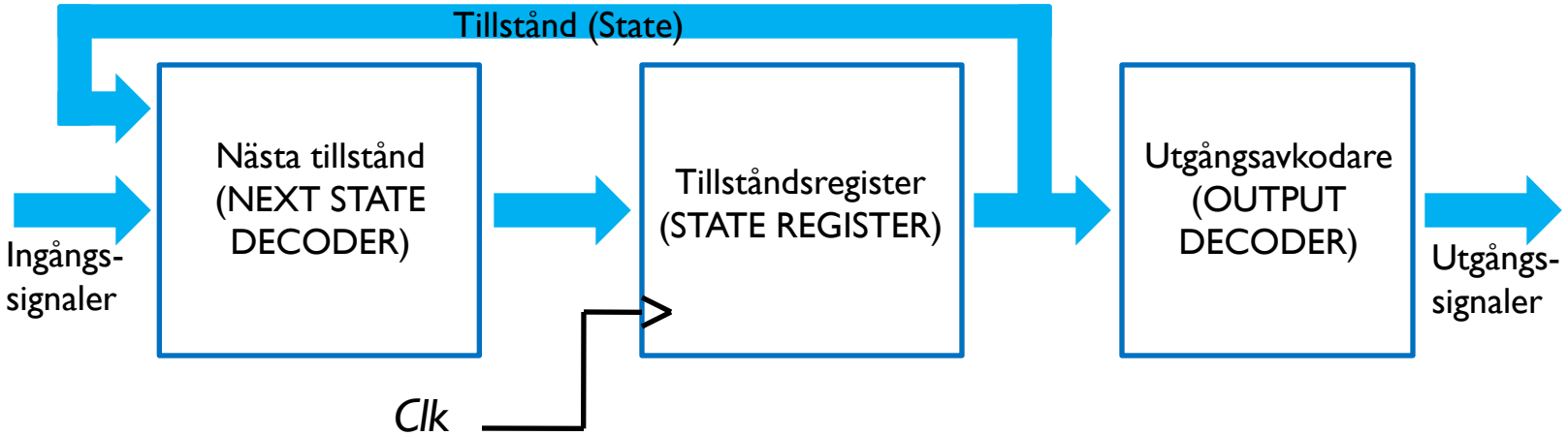
F13: Asynkrona Sekvensnät (Del 2)

Rep. Tillståndsmaskiner



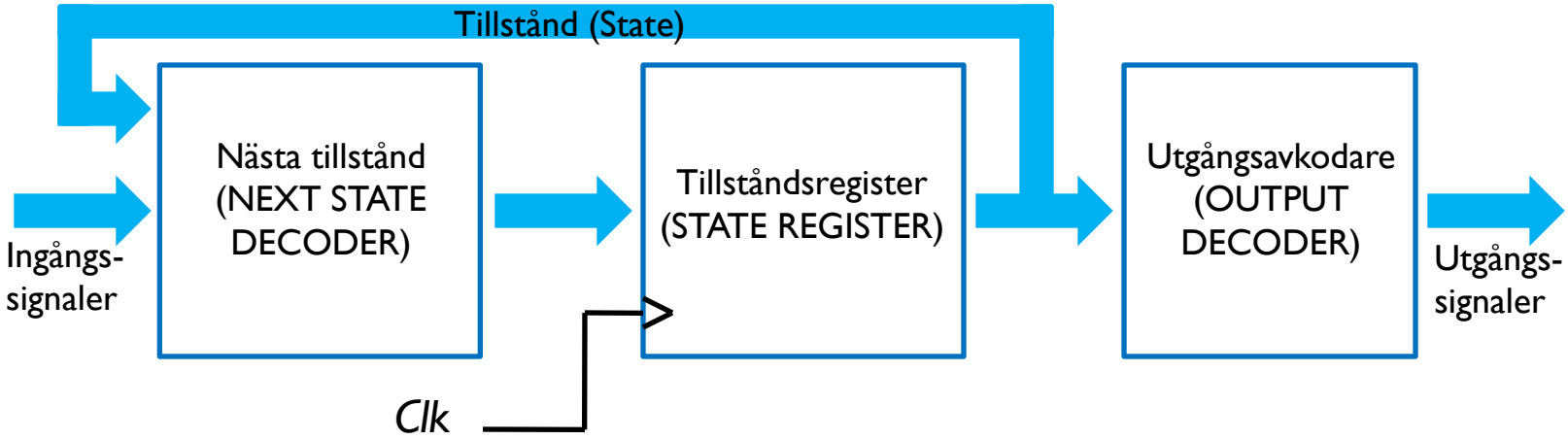
Exempel tillståndsmaskin

- Tillståndsmaskiner styr sekvenser av händelser.
- Övergångar mellan tillstånd styrs av instignaler och nuvarande tillstånd
- Motsvarar ”program” implementerat i digital hårdvara

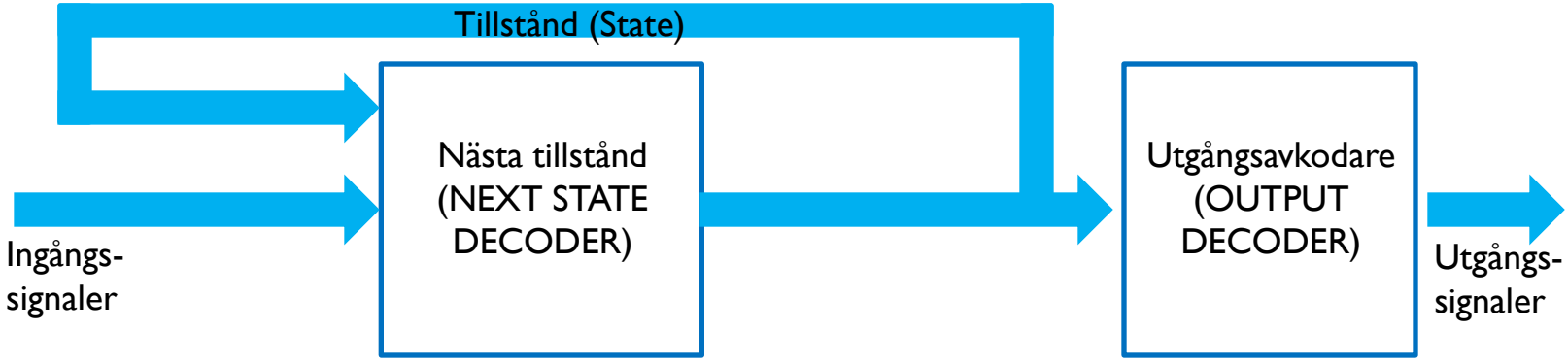


Synkront sekvensnät (Moore automat)

Rep. Asynkrona Sekvensnät

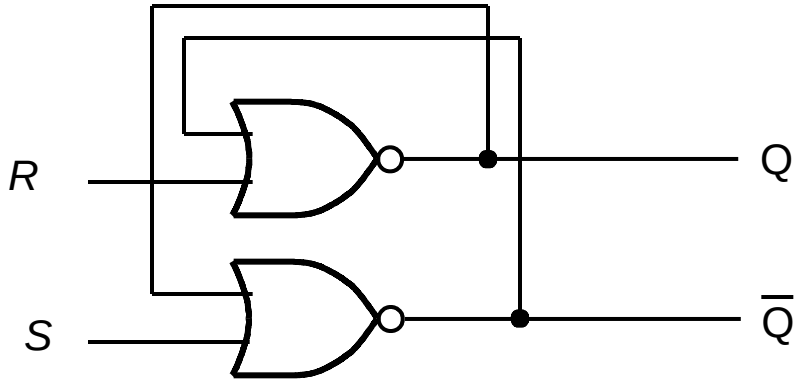


Synkront sekvensnät. Tillståndet låses i ett klockat tillståndsregister.

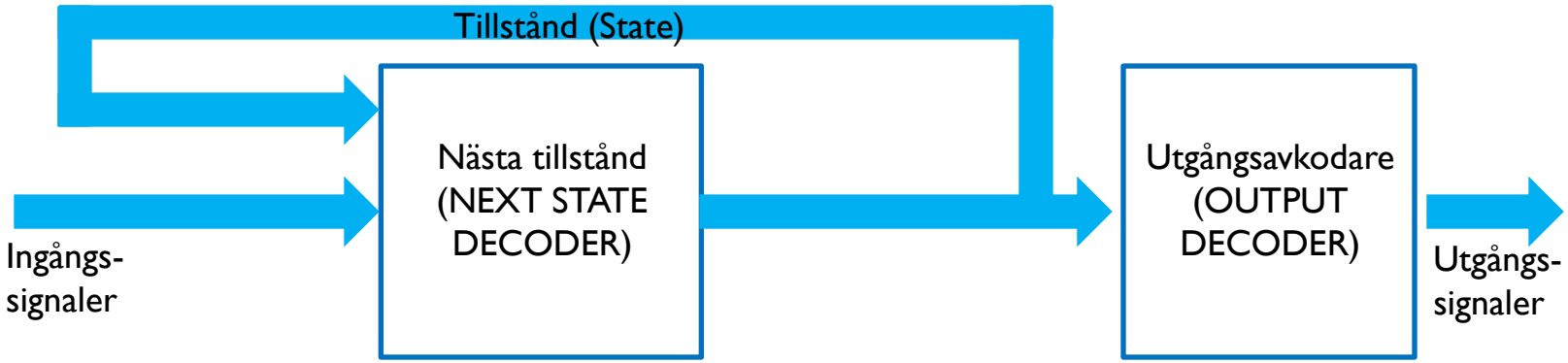


Asynkront sekvensnät. Tillståndet återkopplas direkt.

Rep. Asynkrona Sekvensnät



Exempel: SR vippa kan analyseras som ett asynkront sekvensnät



Asynkront sekvensnät. Tillståndet återkopplas direkt.

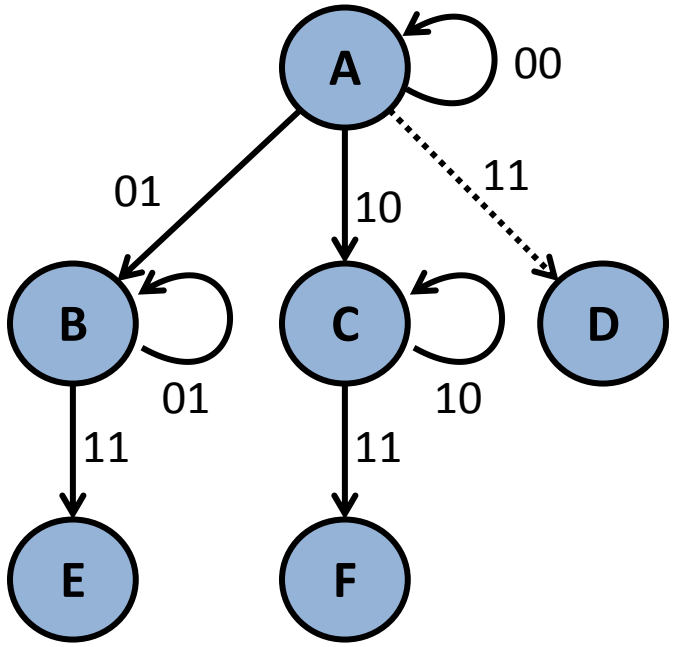
Gyllene regeln

Rep. Asynkrona Sekvensnät

Pga osäkerhet i fördröjningar antar vi att endast en insignal och tillståndsvariabel ändras i taget.

(Hamming distance = 1)

Om flera tillstånd ändrades samtidigt skulle vi kunna få osäkra tillstånds övergångar (Race condition, Hazard)



Exempel: Försöker vi gå från tillstånd A till tillstånd D (insignal ändras från 00 ->11) och signalerna inter är EXAKT synkroniserade blir insignalen (00->01->11) eller (00->10->11) hamnar vi antingnen i tillstånd E eller F. I en verklig implementering är inga signaler exakt synkroniserade, vi kan alltså inte garantera att vi kommer till tillsånd D.

Rep. Asynkrona Sekvensnät

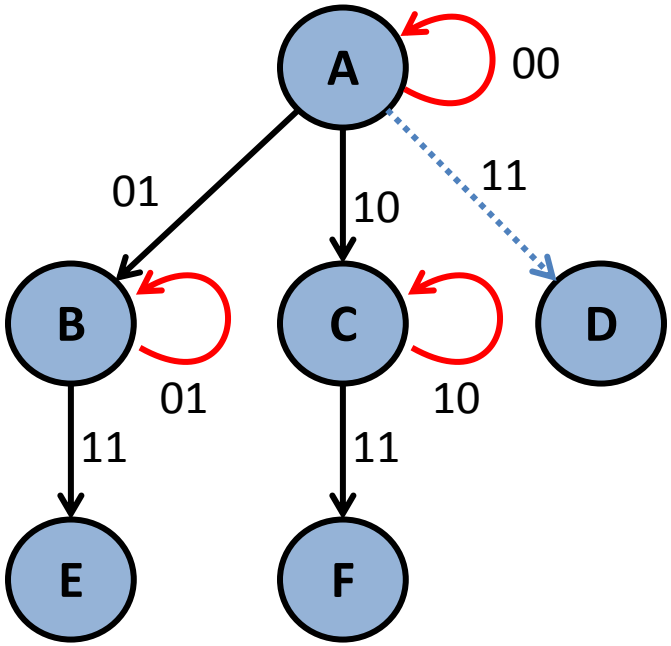
Samma sak på tabellform:

Vi kan inte garantera att vi kan komma till detta tillstånd

Pres state	Next State			
	X=00	01	10	11
A	A	B	C	-
B	x	B	-	E
C	x	-	C	F

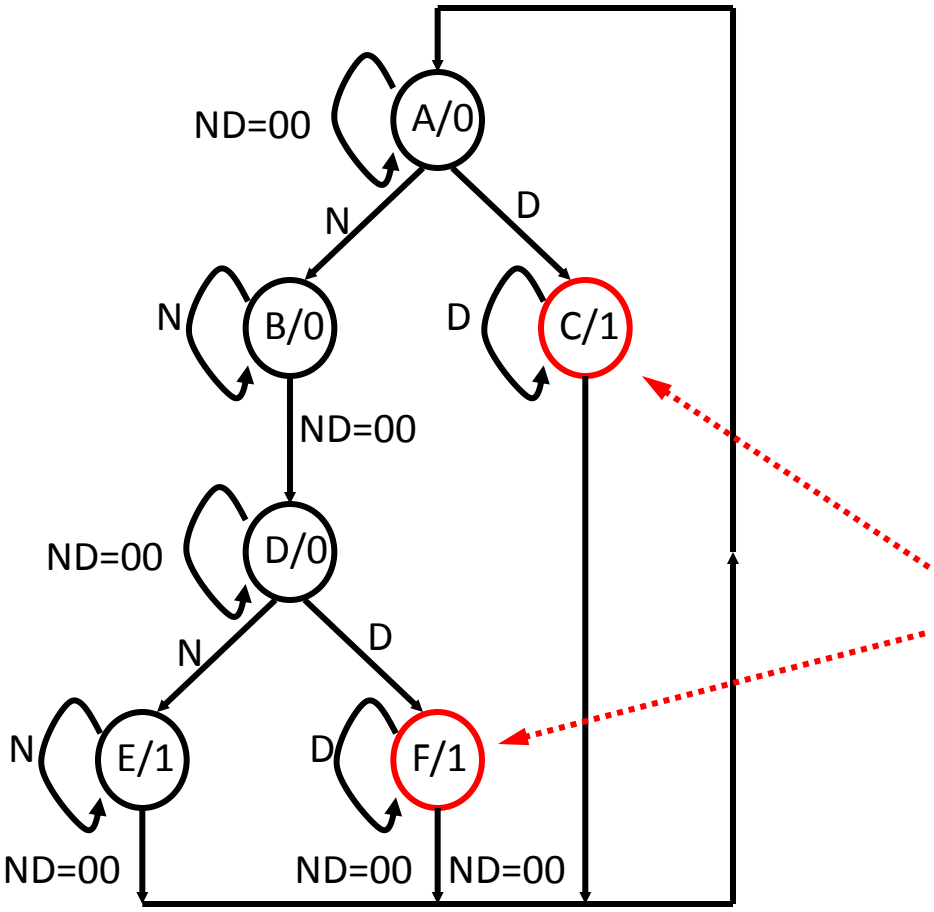
Stabila tillstånd

x ej definierade av tillståndsgrafen (don't care)



Exempel: Försöker vi gå från tillstånd A till tillstånd D (insignal ändras från 00 ->11) och signalerna inter är EXAKT synkroniserade blir insignalen (00->01->11) eller (00->10->11) hamnar vi antingnen i tillstånd E eller F. I en verklig implementering är inga signaler exakt synkroniserade, vi kan alltså inte garantera att vi kommer till tillsånd D.

Rep. Tillståndsminimering



Två tillstånd kan vara ekvivalenta och minimeras.

Minimera för att förenkla tillståndsmaskinen

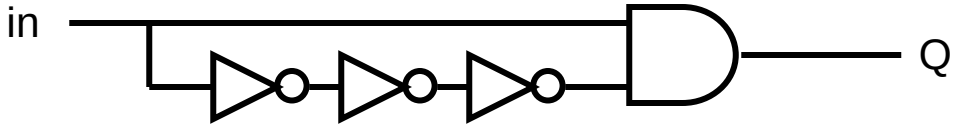
Ex: Tillstånd C och F kan minimeras, eftersom godis skall matas ut om en dime matas in i automaten oberoende av tidigare tillstånd

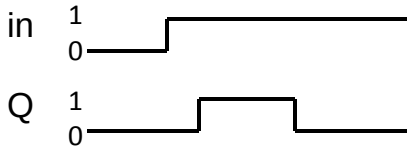
Hasard: Spikar på signalvärden

- När man konstruerar asynkrona kretsar så kan det händer att man få spikar (glitches) på signalvärden
- Detta beror på att olika signalvägar har olika fördröjningstider
- Fenomenet kallas för *hasard* (hazard) och kan elimineras med noggrann konstruktion

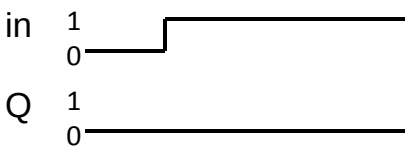
Snabbfråga

Vilket tidsdigaram motsvarar bäst den signal som genereras av följande grindnät vid stigande flank?

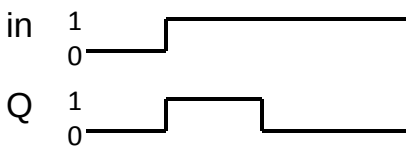




Alt: A



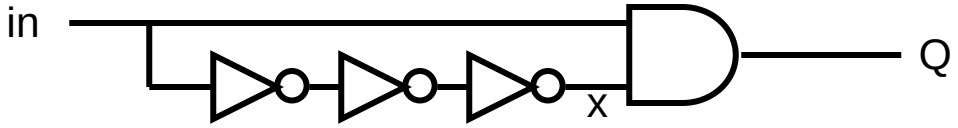
Alt: B

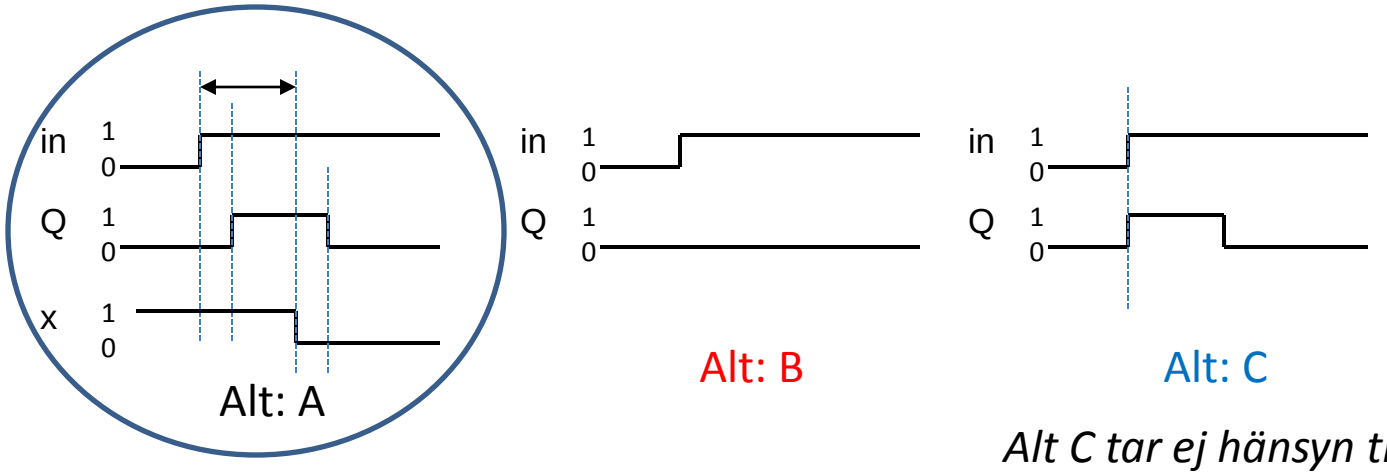


Alt: C

Snabbfråga

Vilket tidsdigaram motsvarar bäst den signal som genereras av följande grindnät vid stigande flank?

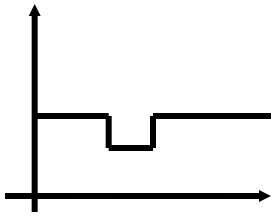




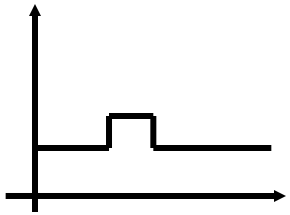
PGA fördröjning i inverterare blir båda ingångarna till AND grinden 1 ett kort tag

Alt C tar ej hänsyn till fördröjning i AND grind

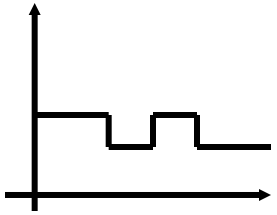
Olika typer av Hasard



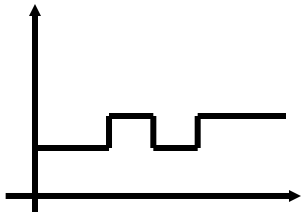
Statisk 1 → 1



Statisk 0 → 0

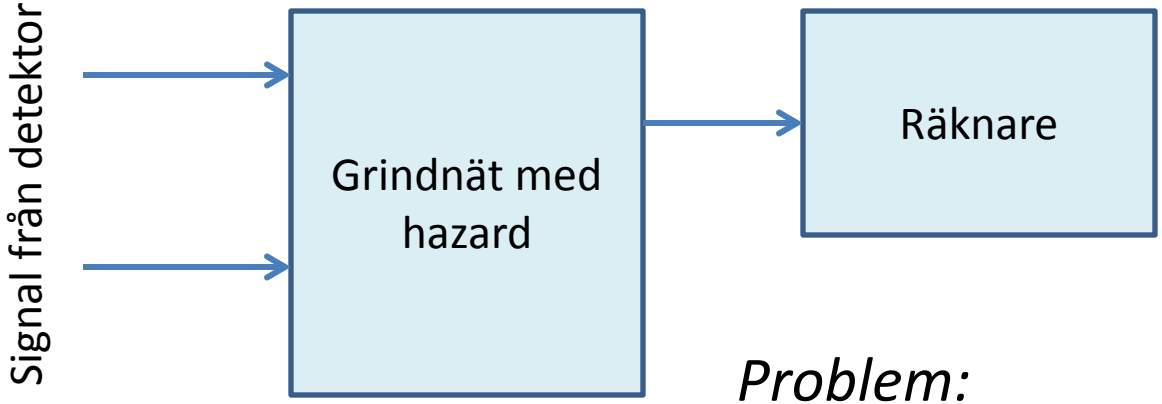


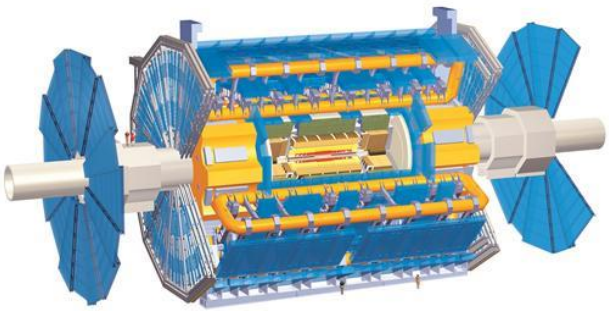
Dynamisk 1 → 0



Dynamisk 0 → 1

Exempel på möjliga problem



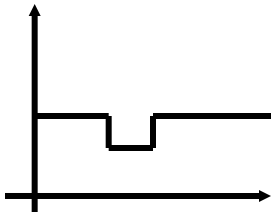


ATLAS particle detector

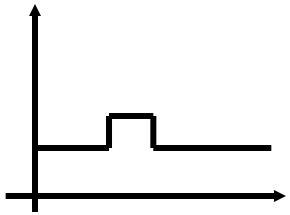
Problem:
Om grindnätet genererar spikar kommer räknaren visa felaktigt antal detekterade partiklar

I askynkrona sekvensnät måste vi undvika hazard

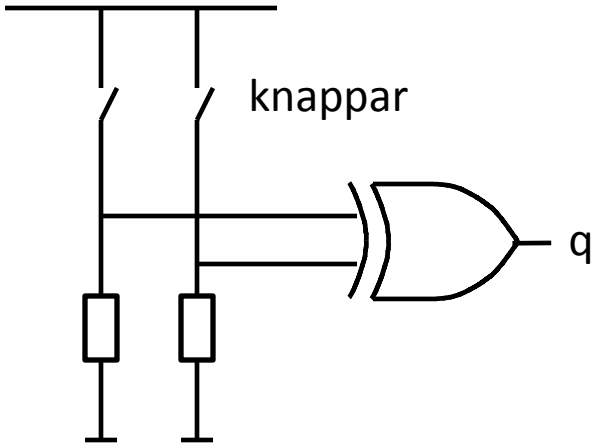
Statisk Hasard



Statisk 1 ⟶ 1



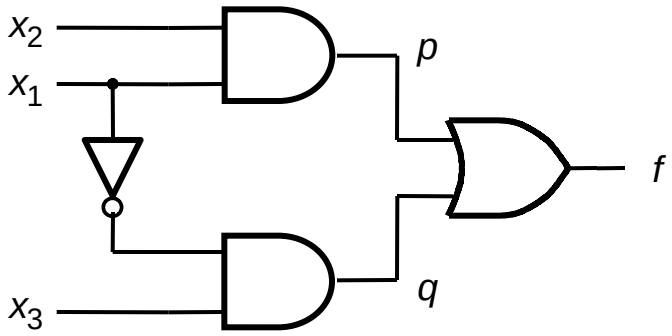
Statisk 0 ⟶ 0



Exempel:
Trycker vi ned
båda knapparna
genereras alltid en kort
puls på ugången eftersom
vi inte kan trycka ned knapparna
EXAKT samtidigt

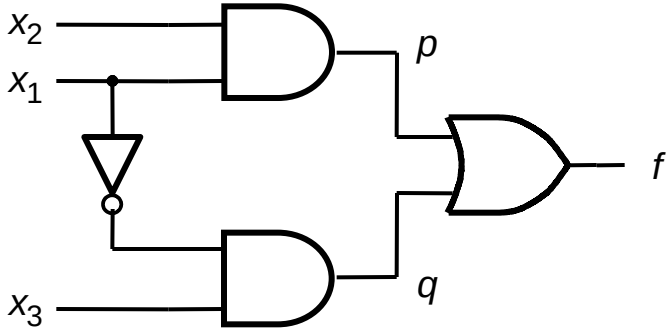
Exempel Statisk Hasard

- Hasard kan uppträda vid nedanstående krest om vid övergången av $x_3x_2x_1$ från 111 => 110

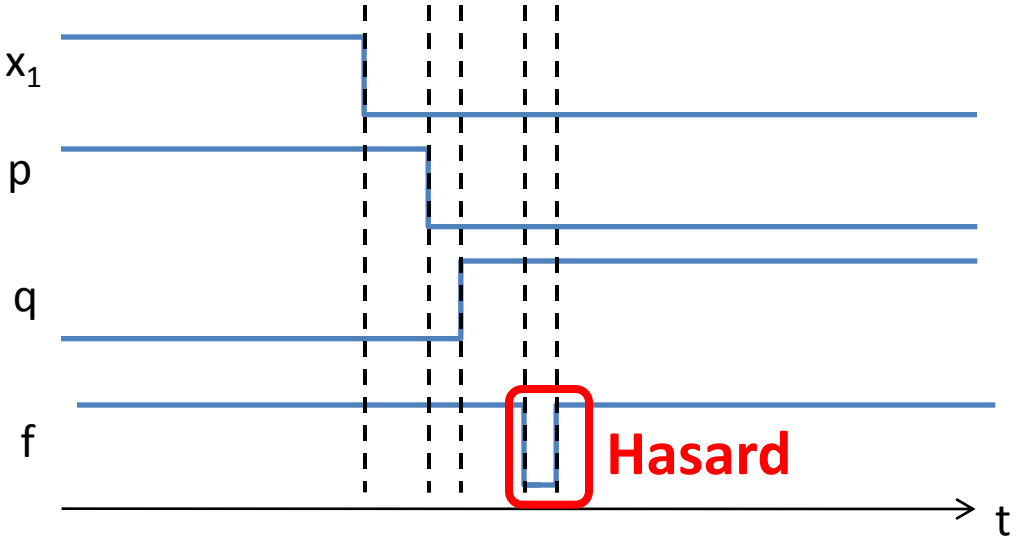


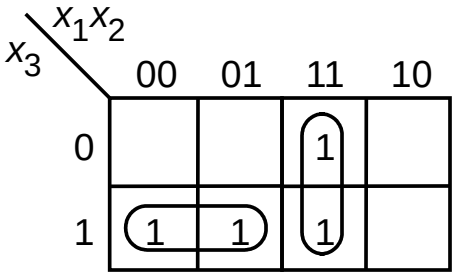
$$f = x_1x_2 + \bar{x}_1x_3$$

Tidsdiagram



$$f = x_1x_2 + \bar{x}_1x_3$$

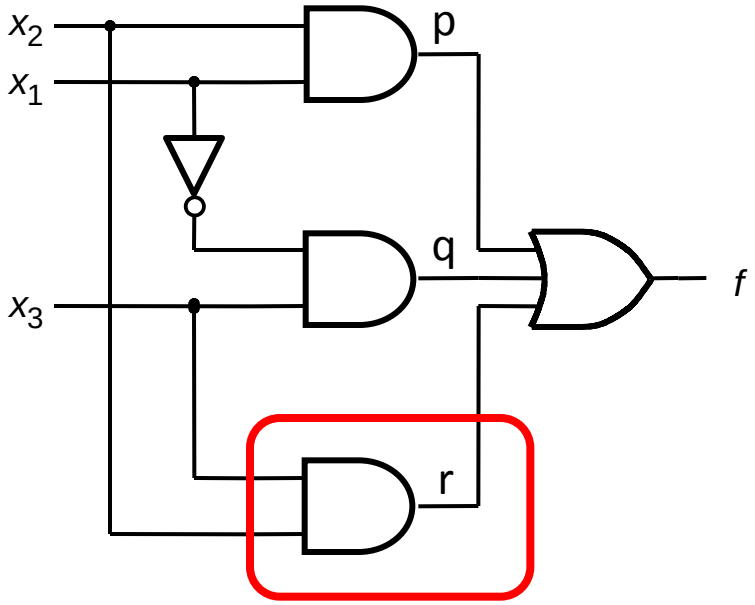


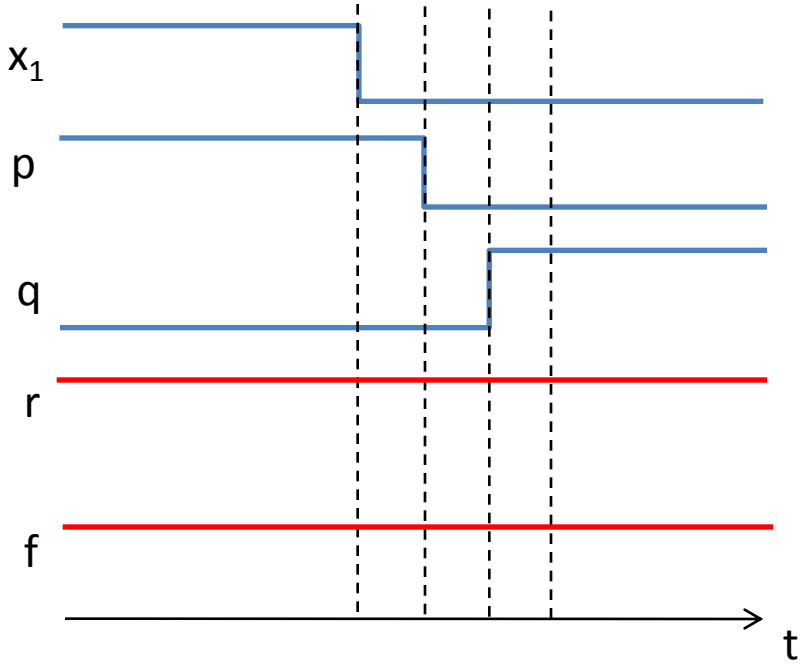


Hur undviker man statisk Hasard?

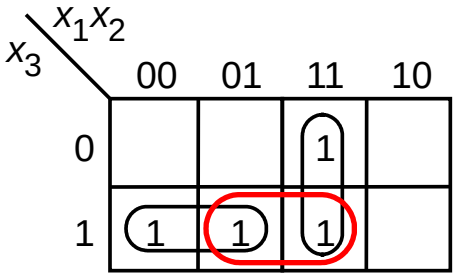
- Möjligheten för statisk hasard finns om två intill-liggande 1:or inte är täckta med en egen produkter i Sum of Products
- Därmed kan man ta bort risken för statisk hazard genom att lägga till en inringningar så att alla intill-liggande 1:or är täckta med en egen inringning

Hasardfri krets (SOP)





Hasard-Cover



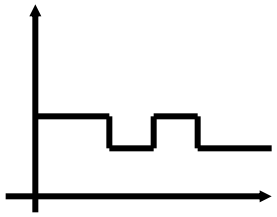
$$f = x_1x_2 + \bar{x}_1x_3 + x_2x_3$$

Statisk Hasard i en POS-circuit

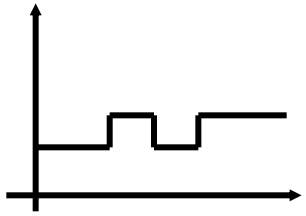
- Har man en Product of Sums (POS)-implementering så måste man se till att man alla bredvidliggande 0:or är täckta av en egen produktterm

Dynamisk Hasard

- En dynamisk hasard orsaker flera spikar på utgången
- En dynamisk hasard orsakas av kretsens struktur



Dynamisk 1 → 0



Dynamisk 0 → 1

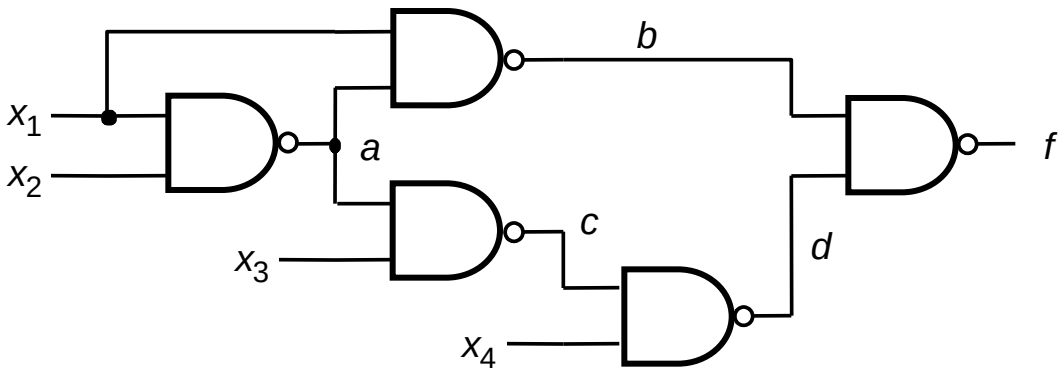
Exempel: Dynamisk Hasard

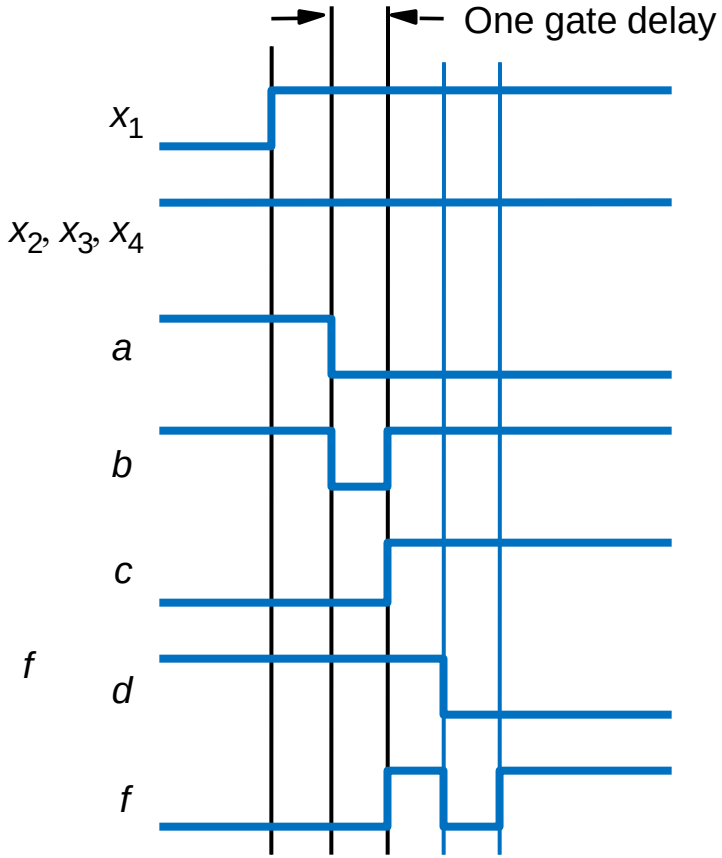
- Följande ekvation osaker ingen hasard om man implementera det som AND-OR-struktur

$$f = x_1\bar{x}_2 + \bar{x}_3x_4 + x_1x_4$$

Exempel: Dynamisk Hasard

- Men implementerar man ekvationen med följande flernivåslogik, så uppträder dynamsik hasard



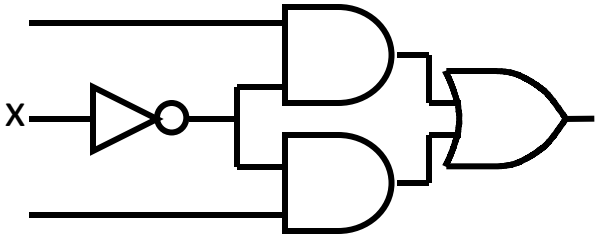


Hur kan man undvika Dynamisk Hasard?

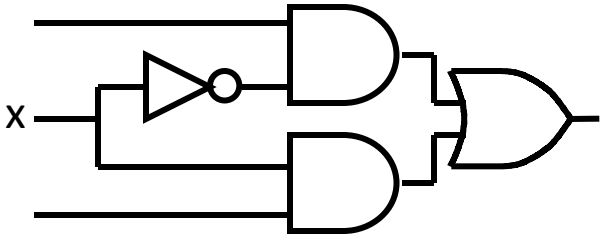
- Dynamisk hasard kan undvikas med två-nivås-logik
- Man måste vid minimering se till att kretsen är fri från statisk hasard, då finns det inte heller en dynamisk hasard!

Snabbfråga

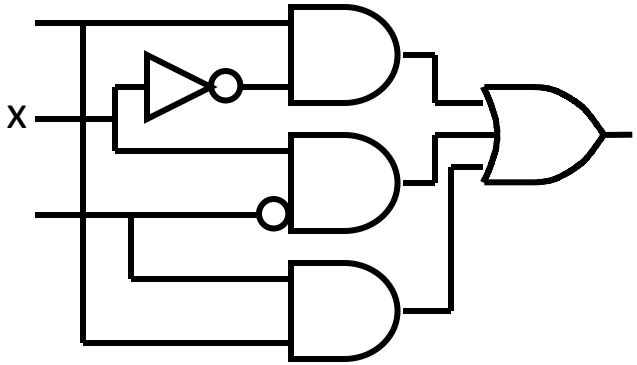
Vilket/vilka av följande grindnät kan ge upphov till hazard då x ändras ?



Alt: B



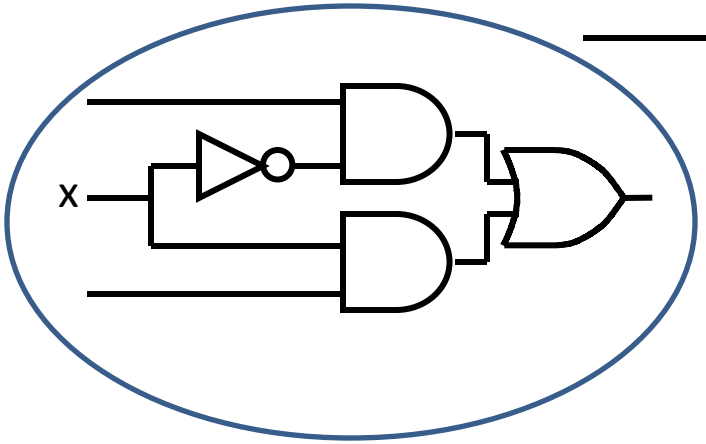
Alt: A



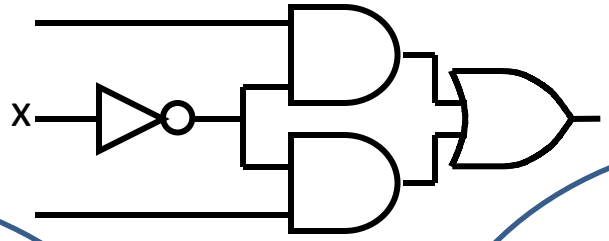
Alt: C

Snabbfråga

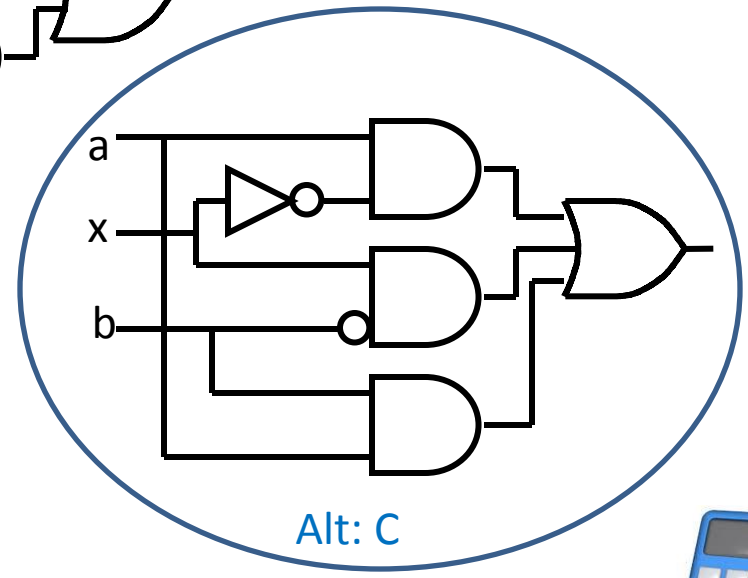
Vilket/vilka av följande grindnät kan ge upphov till hazard då x ändras ?



Alt: A



Alt: B



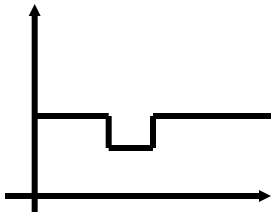
Alt: C

Risk för hazard då a=1 och b=0
Den extra grinden täcker
inte detta fall (utan a=1 och b=1)

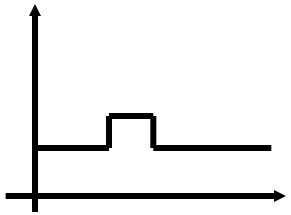
När ska man tänker på hasard?

- I ett *asynkront sekvensnät* måste avkodaren för nästa-tillstånd vara hasardfri!
  - Annars kan man hamnar i ett inkorrekt tillstånd
- För *kombinatoriska kretsar* är hasard inte viktigt eftersom utgången kommer efter ett kort tag stabiliserar sig
- I ett *synkront sekvensnät* är hasard inget problem, så länge man respekterar setup- och hold-tider (under dessa tider får hasard inte uppträda!)

Undvik Hasard!

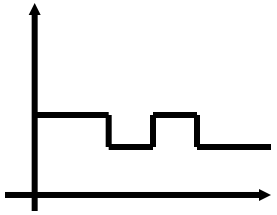


Statisk 1 ⟶ 1

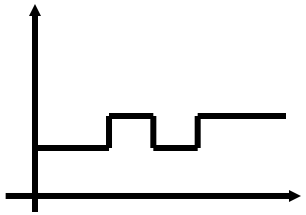


Statisk 0 ⟶ 0

Statisk hasard orsakas av utelämnade prim-implikanter



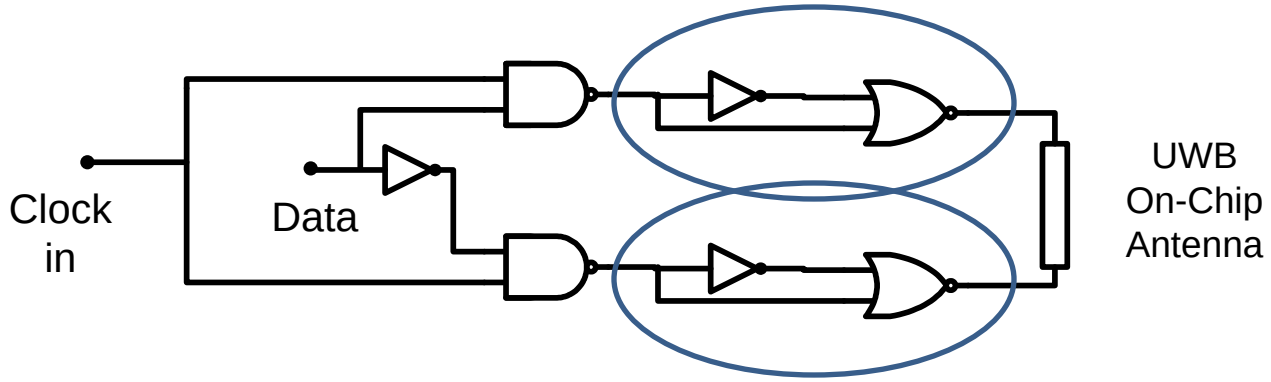
Dynamisk 1 ⟶ 0



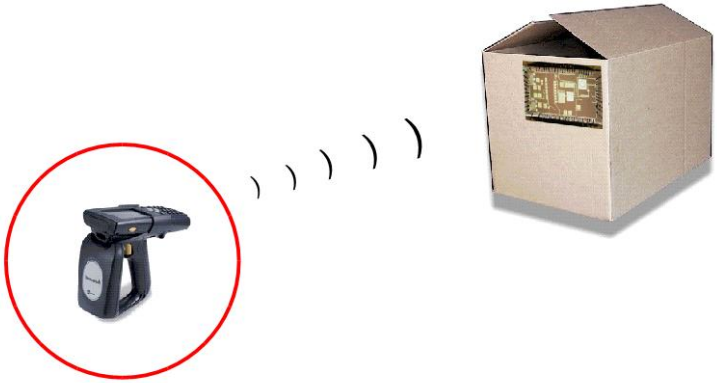
Dynamisk 0 ⟶ 1

Dynamisk hasard kan uppstå när man implementera kretsar med flernivåslogik. Två- nivåslogikkretsar som är fria från statisk hasard är också fria från dynamisk hasard.

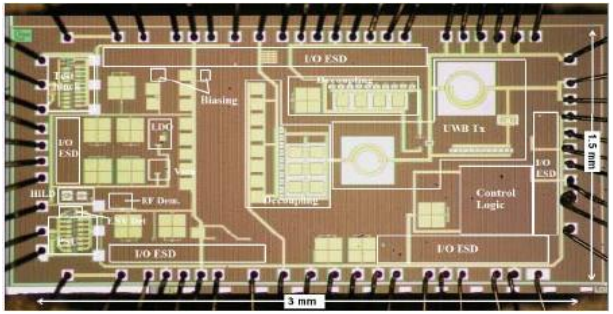
Exempel på önskvärd hazard



Pulse generators



Exempel: Kommunikation med Pulsad Ultra Wideband



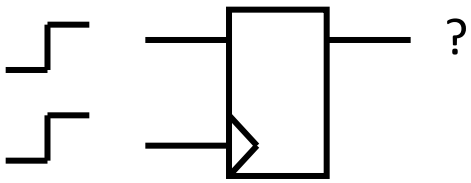
Chipfoto

Spikar på utgången i ett asynkront sekvensnät

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

Man kan få utgångsspikar I ett asynkront sekvensnät när man byter från ett stabilt tillstånd till ett annat genom att passerar flera instabila tillstånd (Fenomenet är ingen hasard!).

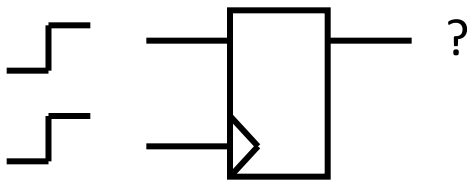
Metastabilitet



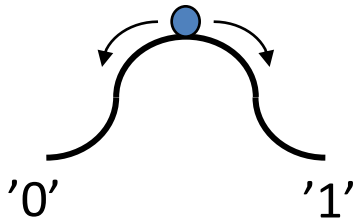
Om Clk och D slår om samtidigt, vilket värde får då Q?

Metastabilitet (forts.)

- Denna instabilitet varar tills transistorerna i återkopplingen behagar gå åt ena eller andra hållet – men det kan ta tid, och tiden beror bla på hur nära $V_{DD}/2$ som låsningen skedde.
- Man kan likna situationen vid en boll som ligger på toppen en kulle eller en penna som balanserar på sin spets. Minsta störning kommer att få bollen eller pennan att falla åt ena eller andra hållet.

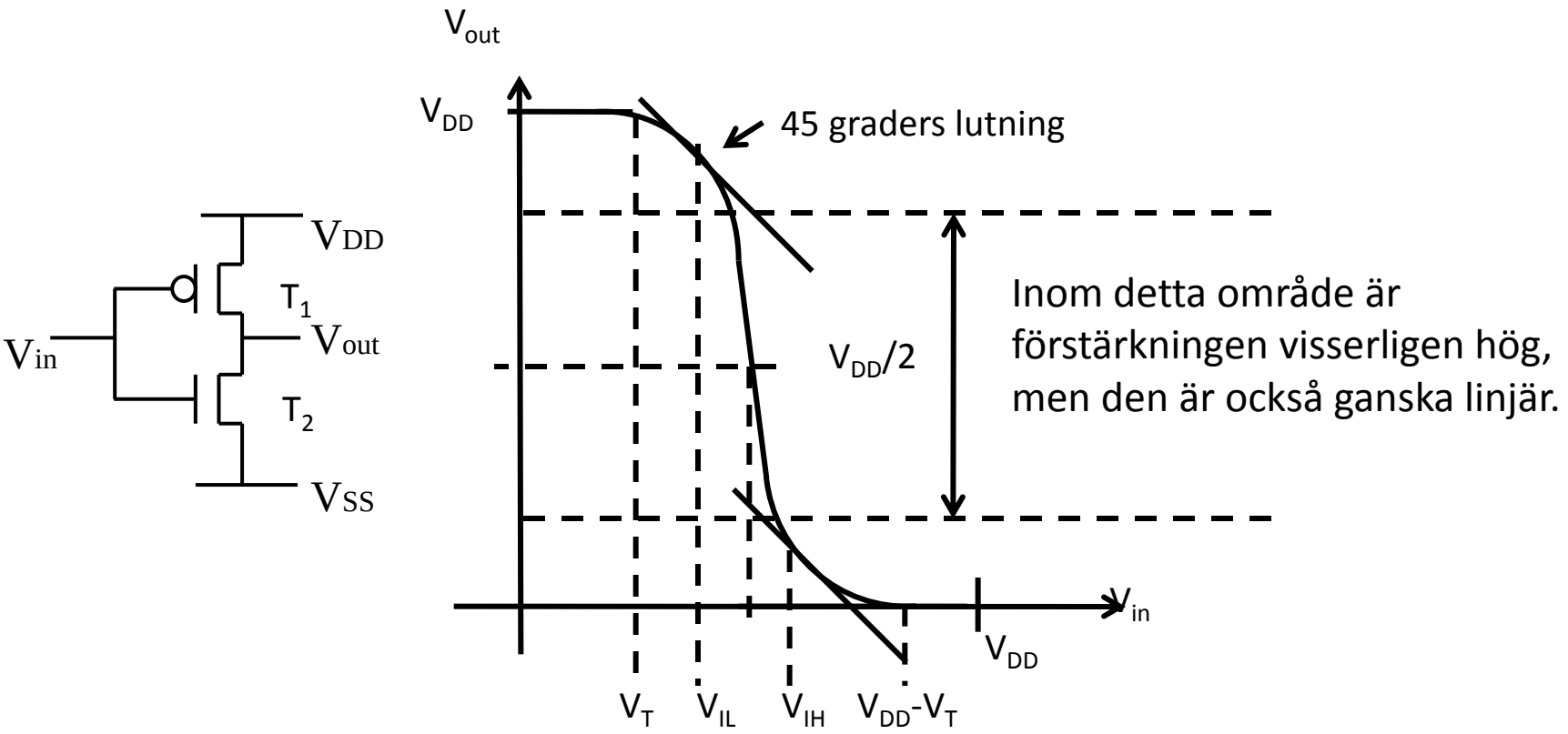


Om Clk och D switchar samtidigt, vilket värde får då Q?



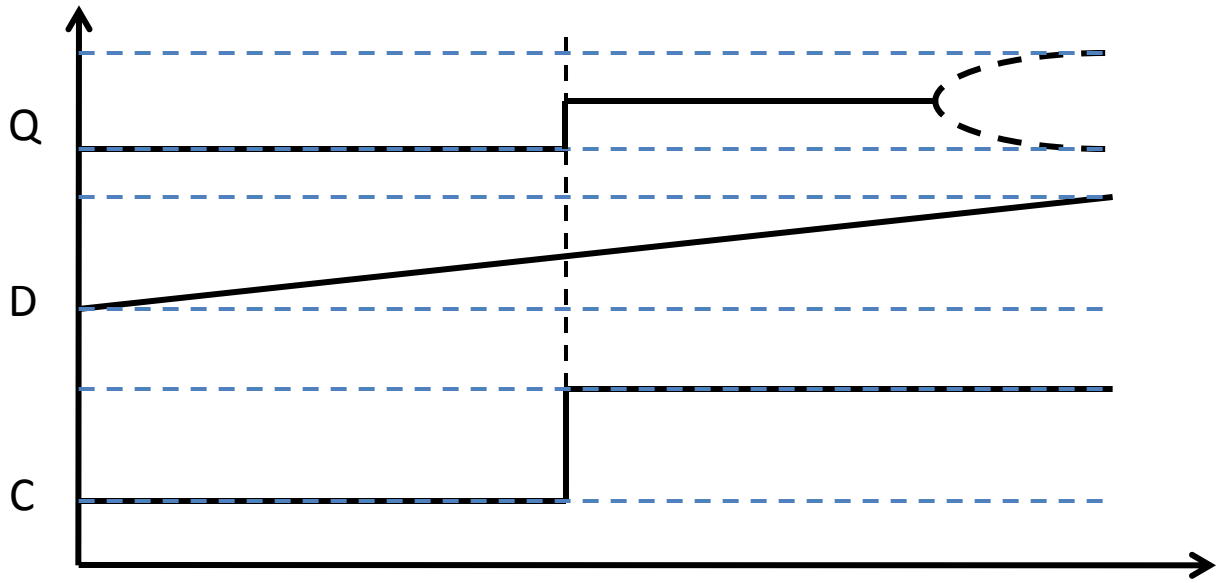
På vilken sida kommer bollen att trilla ner?

Inverterarens överföringsfunktion



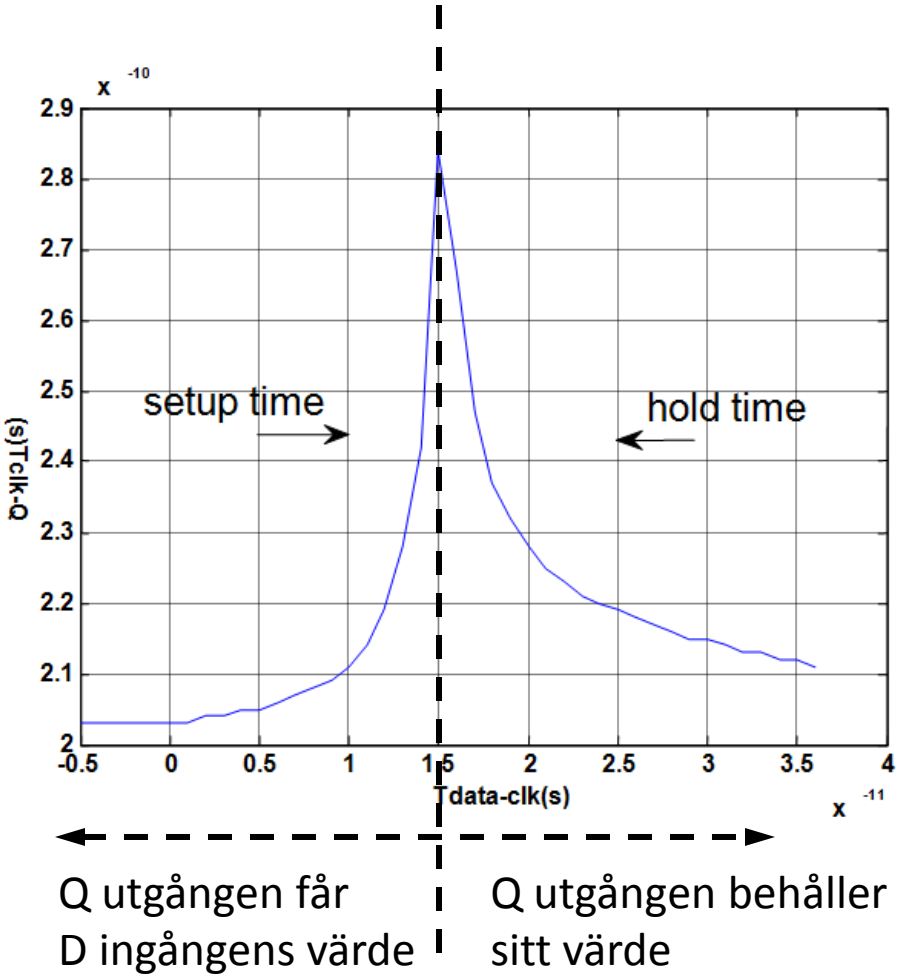
För att förstå metastabilitet krävs att grindar analyseras som analoga byggblock

Om metastabilitet...



För att förstå vad metastabilitet innebär så kan vi tänka oss att insignalen D till latchen är väldigt belastad och därmed slår om mycket långsamt i förhållande till klockan. Antag vidare att klock-signalen C slår om precis när D är vid $V_{DD}/2$. Då låser sig latchen vid det spänningsvärde som råkar finnas på D. Efter en tid slår latchen om till antingen ’1’ eller ’0’.

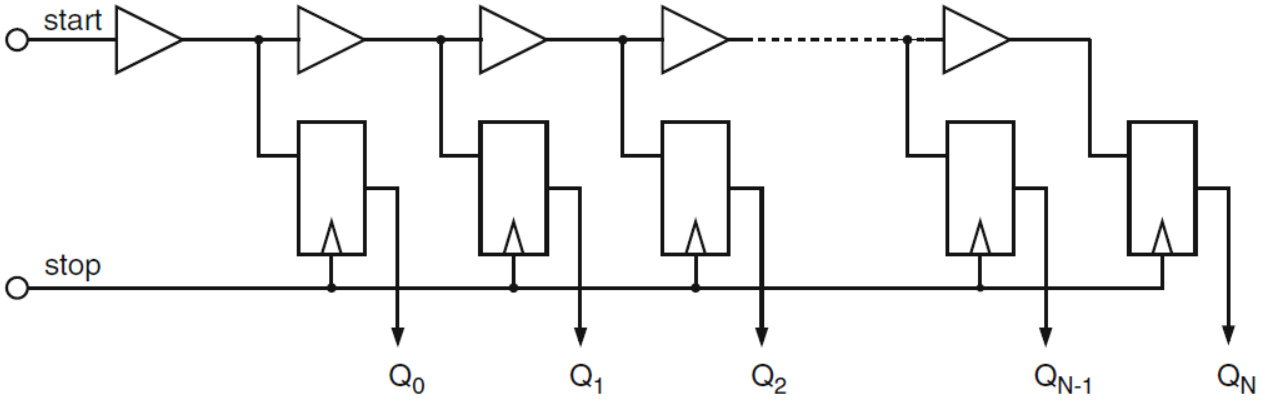
Exempel Setup and Hold time

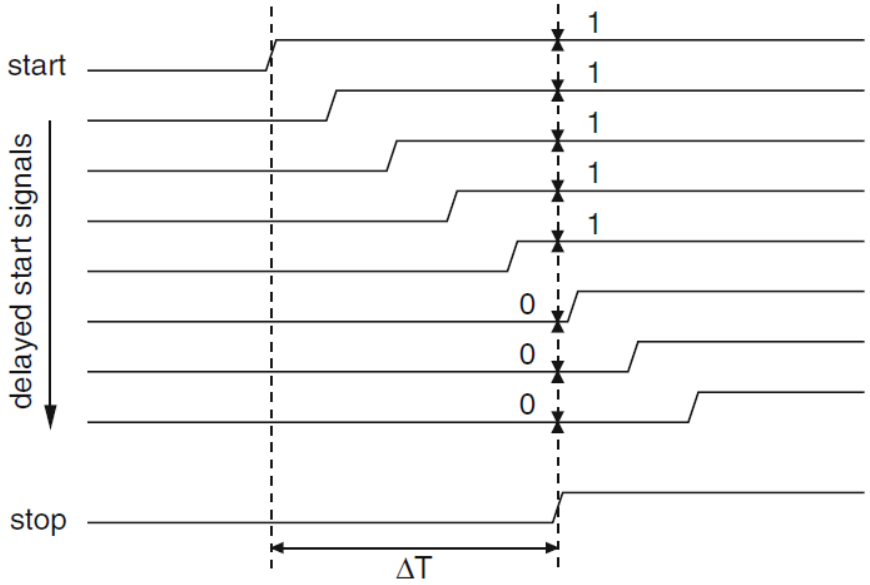


Utgången stabiliseras
alltid till ett logiskt värde,
men det tar längre
tid om setup och hold time
är litet.

På grund av process variationer
är det också osäkert exakt
vid vilken tidpunkt övergången
mellan setup och hold sker.

Exempel: Time to Digital converter



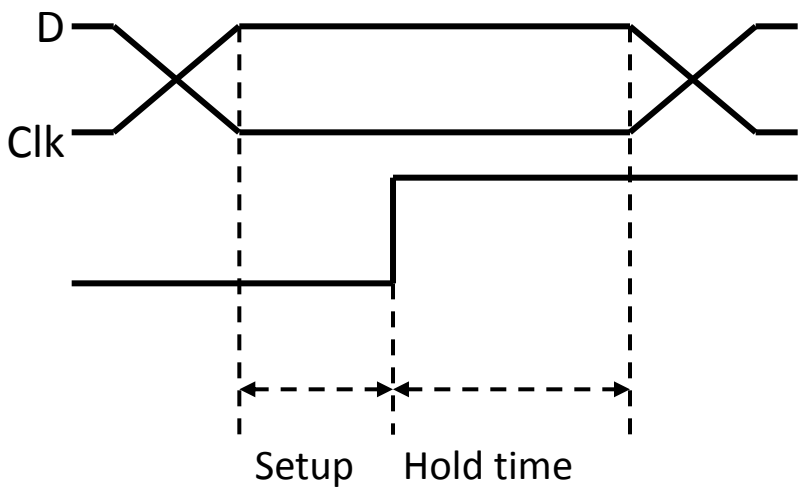


Exempel vågform

Används för att detektera tidsskillnad mellan två signaler, tex vid frekvenslåsning och i avståndsmätning

Setup and Hold time (= metastabilitets-skydd)

- För att undvika samtidigt omslag/switchning, så måste setup and hold times garanteras:



Setup time är den tid D måste vara stabil innan Clk ändrar värde
Hold time är den tid D måste vara stabil efter Clk har ändrat värde

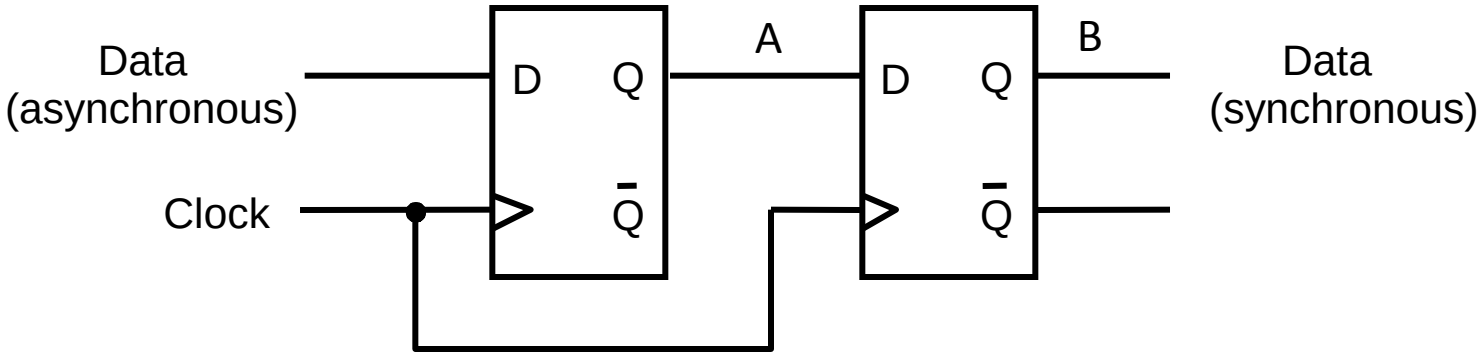
Om Setup and Hold times är uppfyllda, så kommer vippan (Flip-flop) att garanterat bete sig snällt/deterministiskt!

Asynkrona ingångar

- Dessvärre kan vi inte alltid garantera att en ingång är stabil under hela setup- och hold-tiden
- Antag att du kopplar in en tryckknapp på D-ingången av en vippa
  - Användaren kan trycker knappen när som helst, även under setup- och hold-tiden!
  - Risken är att vippan hamnar i ett metastabilt tillstånd!

Synkronisering av asynkrona utgångar

- För att synkronisera asynkrona ingångar användar man en extra vippa på ingången
- Den första vippans utgång (A) kan hamnar i ett metastabilt läge
- Men om klockperioden är tillräckligt lång, så kommer den att stabiliseras innan nästa klockflank, så att B inte hamnar I ett metastabilt läge!

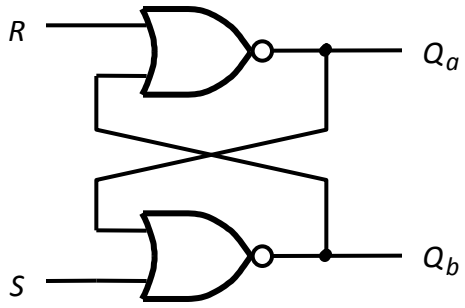


Exempel – förbättrad SR-latch

- **Konstruktion av set-dominant SR-latch**
- **Specifikation**
  - Konstruktionen är en speciell typ av SR-latch (det finns ***inte*** ett förbjudet läge 11)
    - Om *S* och *R* är 1 så går latchen i SET-läge (*Q* = 1)
    - Latchen kan först gå till RESET-läge om
      1. både *S* och *R* först sätts till *S*=0 och *R*=0 (*Q* = 1)
      2. *R* aktiveras (*S* = 0, *R* = 1) $\rightarrow$ *Q* = 0 !

Källa: "Fletcher: Engineering Approach to Digital Design", Prentice-Hall, 1980. Exempel 10.5 (pp 670).

Repris: SR-latch



(a) Circuit

S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

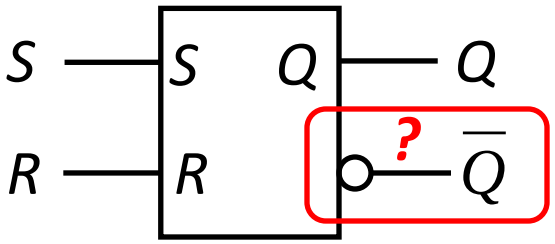
(b) Truth table

Förbjuden insignal S=R=1

$Q_a \neq \overline{Q}_b$

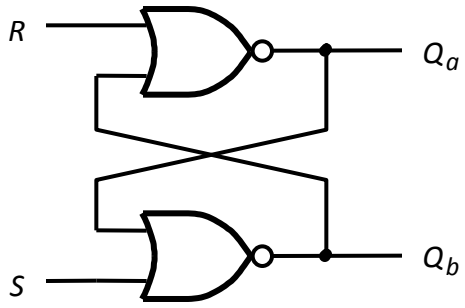
Så länge man **undviker** insignalen $S = R = 1$ (= förbjudet tillstånd) kommer utgångarna Q_a och Q_b att vara varandras inverser. Man kan då använda symbolen till höger.

SR-Latch



Tar man signaler från låskretsar finns det således alltid inverser att tillgå!

Mer problem med SR-latchen



(a) Circuit

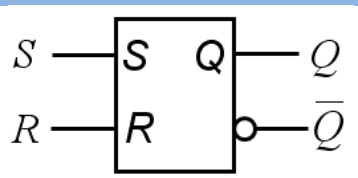
S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

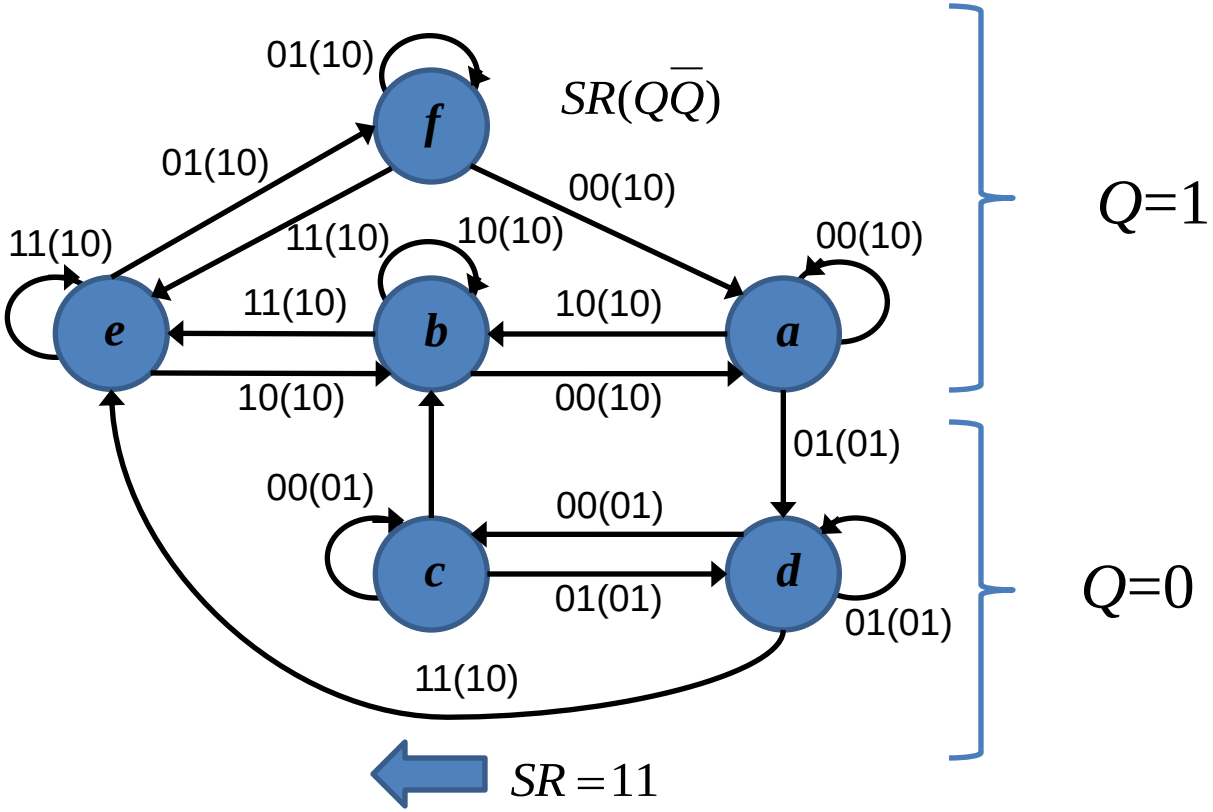
(b) Truth table

Om man går från $SR = 11$ till $SR = 00$ är det en **dubbeländring** som krävs av insignalerna. Därför hamnar vi antingen i $Q = 0$ eller i $Q = 1$ ingen kan veta!

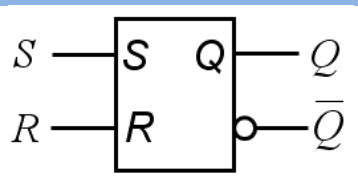
- Detta är ytterligare ett skäl till att förbjuda $SR = 11$.

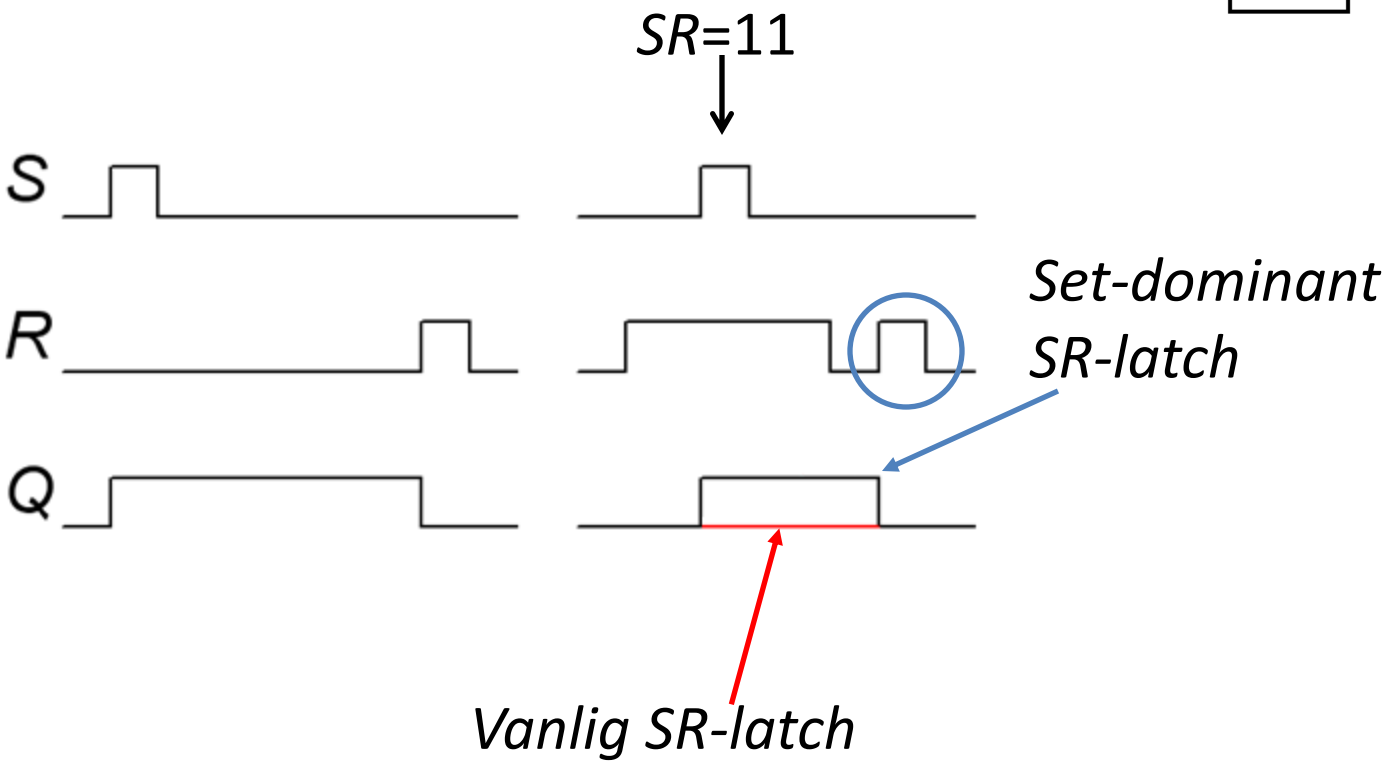
SET-dominant SR-latch



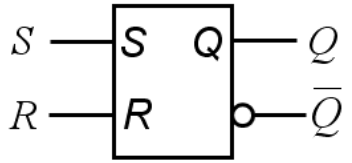


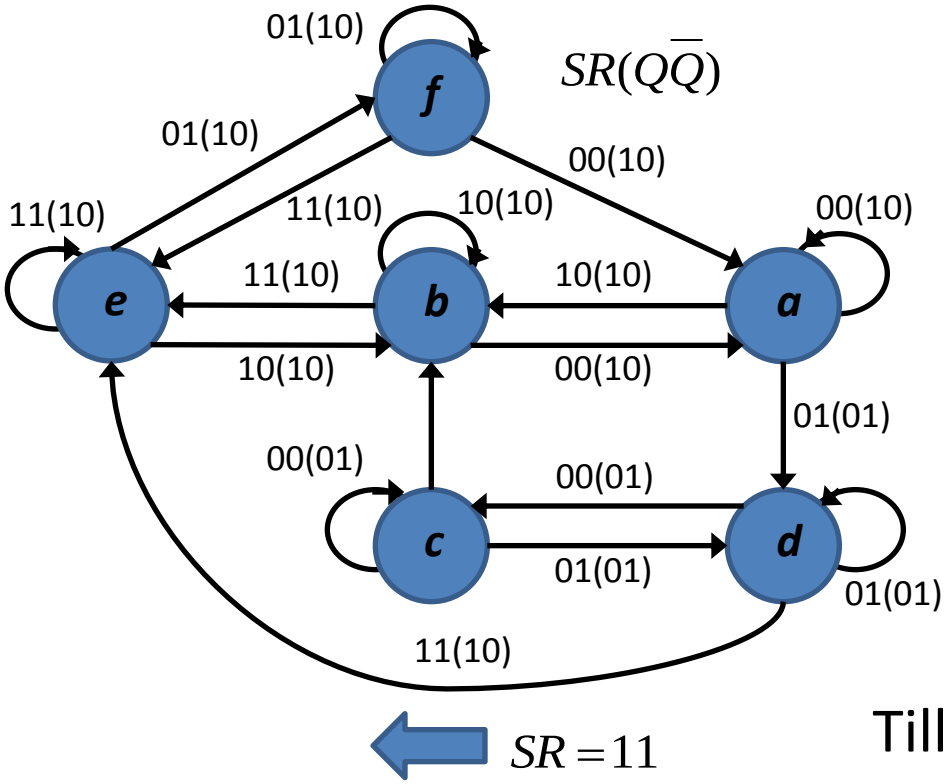
Önskat beteende





SET-dominant SR-latch



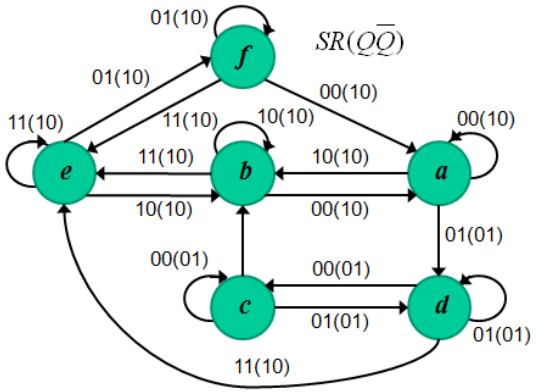


$SR(Q\overline{Q})$

	00	01	11	10	Q	$\overline{Q}$
a	(a)	d	–	b	1	0
b	a	–	e	(b)	1	0
c	(c)	d	–	b	0	1
d	c	(d)	e	–	0	1
e	–	f	(e)	b	1	0
f	a	(f)	e	–	1	0

Tillståndet ***e*** tar ”hand om” fallet $SR = 11$

Kompatibilitet



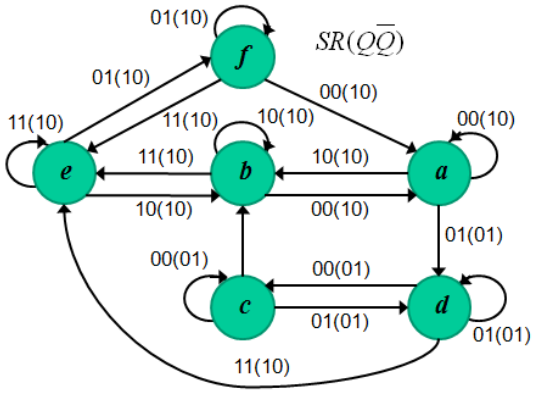
a (10) **b** (10) **c** (01) **d** (01) **e** (10) **f** (10)

$SR(Q\bar{Q})$

	00	01	11	10	Q	$\bar{Q}$
a	(a)	d	–	b	1	0
b	a	–	e	(b)	1	0
c	(c)	d	–	b	0	1
d	c	(d)	e	–	0	1
e	–	f	(e)	b	1	0
f	a	(f)	e	–	1	0

***Det finns inga* ekvivalenta *tillstånd, finns det några Moore-* kompatibla *tillstånd* … ?**

Kompatibilitet



$SR(Q\overline{Q})$

	00	01	11	10	Q	$\overline{Q}$
a	(a)	d	–	b	1	0
b	a	–	e	(b)	1	0
c	(c)	d	–	b	0	1
d	c	(d)	e	–	0	1
e	–	f	(e)	b	1	0
f	a	(f)	e	–	1	0

a (10) ***b*** (10) ***c*** (01) ***d*** (01) ***e*** (10) ***f*** (10)

Många valmöjligheter …

a(10): **a**d-b
b(10): a-e**b**

b(10): a-e**b** f(10):
a**f**e-

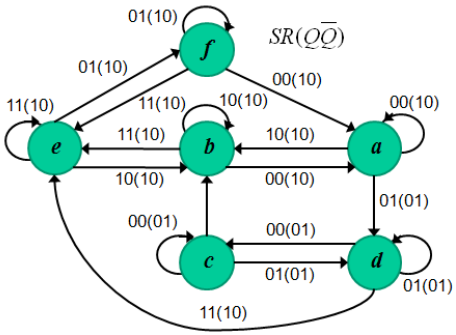
b(10): a-**e**b
e(10): -f**e**b

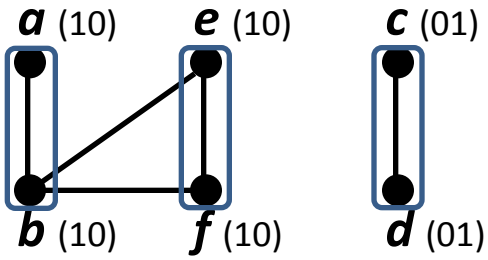
e(10): -f**e**b f(10):
a**f**e-

c(01): **c**d-b
d(01): c**d**e-

Kompatibilitetsgraf

Många valmöjligheter …





$SR(Q\bar{Q})$

	00	01	11	10	Q	$\bar{Q}$
a	(a)	d	–	b	1	0
b	a	–	e	(b)	1	0
c	(c)	d	–	b	0	1
d	c	(d)	e	–	0	1
e	–	f	(e)	b	1	0
f	a	(f)	e	–	1	0

Nya beteckningar
a (*ab*), *e* (*ef*), *c* (*cd*)
Tre tillstånd kräver
två tillståndsvariabler
Y_2 och Y_1

$SR(Q\bar{Q})$

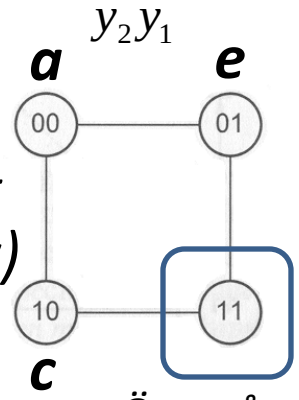
	00	01	11	10	Q	$\bar{Q}$
a	(a)	c	e	(a)	1	0
c	(c)	(c)	e	a	0	1
e	a	(e)	(e)	a	1	0
–	–	–	–	–	–	–

Reducerad flödestabell

Tillståndskodning

$SR(Q\bar{Q})$	00	01	11	10	Q	$\bar{Q}$
a	a	c	e	a	1	0
c	c	c	e	a	0	1
e	a	e	e	a	1	0
–	–	–	–	–	–	–

Vald tillstånds-kod (Gray)



Övergångs-tillstånd

Utgångsavkodning

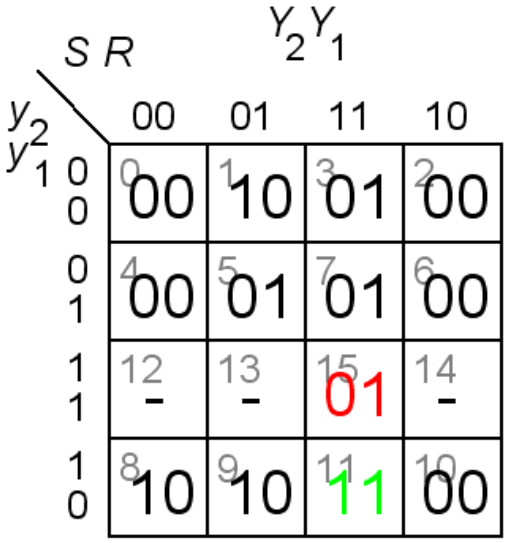
$$a, e \quad Q = 1$$
$$c \quad Q = 0$$
$$Q = \bar{y}_2$$

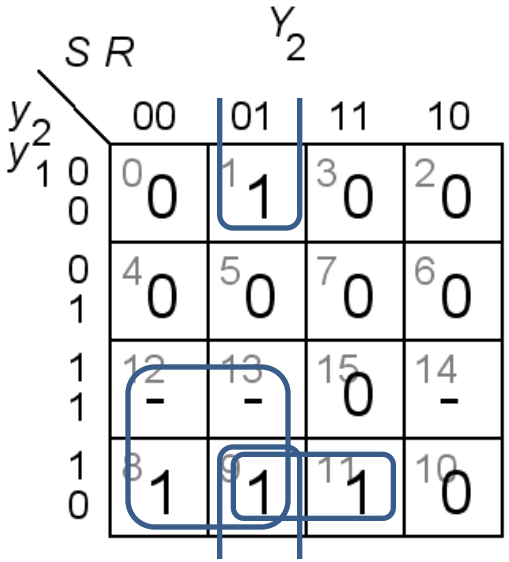
y_2y_1 \ SR	00	01→11	11	10
00	0 a	1 c	3 e	2 a
01	4 a	5 e	7 e	6 a
11	12 –	13 –	15 – ?	14 –
10	8 c	9 c	11 e	10 a

y_2y_1 \ SR	00	01	11	10
00	0 00	1 10	3 01	2 00
01	4 00	5 01	7 01	6 00
11	12 –	13 –	15 01	14 –
10	8 10	9 10→11	11	10 00

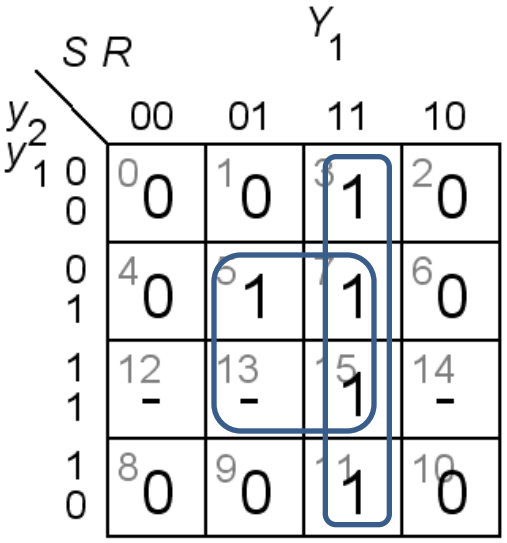
Från ***c*** till ***e*** krävs det en ”dubbeländring” av Y_2Y_1 detta ändras med hjälp av **övergångstillståndet** till två ”enkeländringar”

Karnaughdiagram





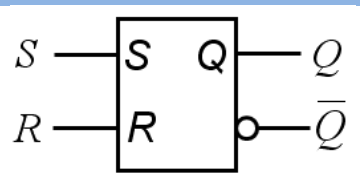
$$Y_2 = \bar{S}y_2 + \bar{S}R\bar{y}_1 + Ry_2\bar{y}_1$$



$$Y_1 = Ry_1 + SR$$

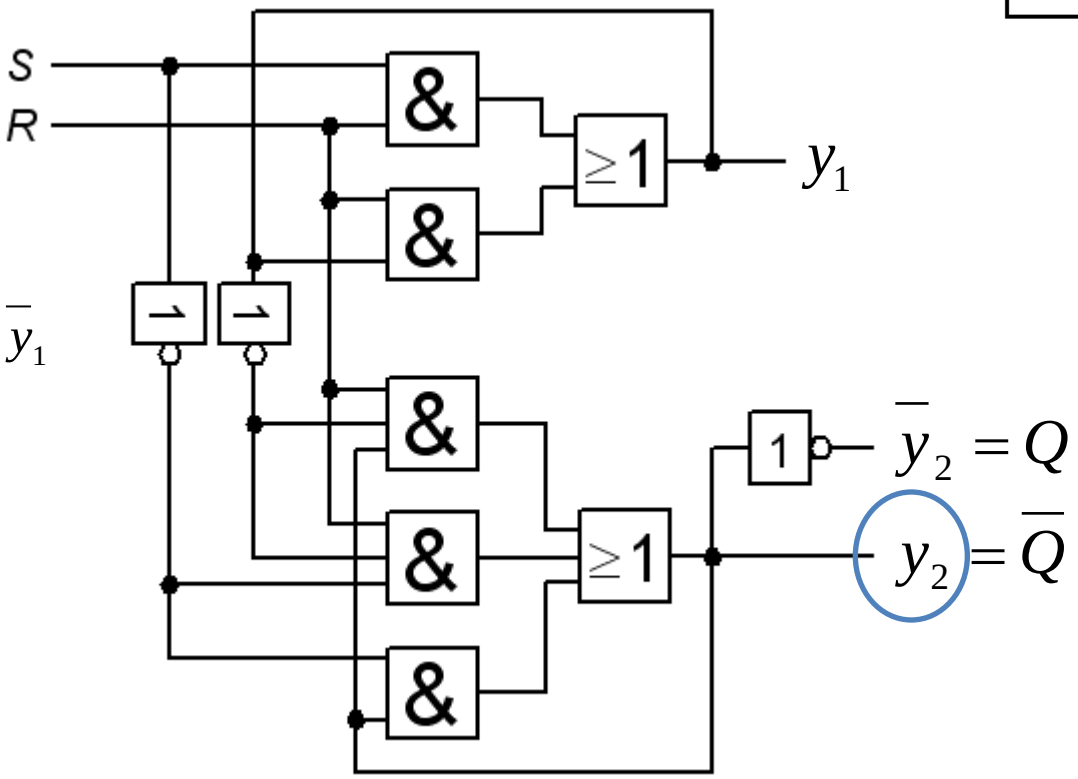
Hasardfria nät direkt!

Krets-schema



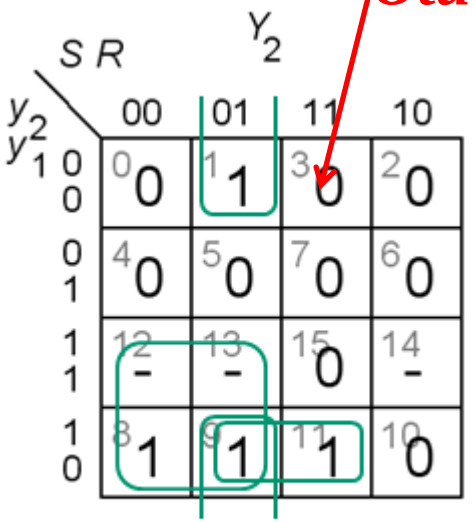
$$Y_2 = \overline{S}y_2 + \overline{S}R\overline{y}_1 + Ry_2\overline{y}_1$$

$$Y_1 = Ry_1 + SR$$



Här har vi vår ”idiotsäkra” SR-låskrets!

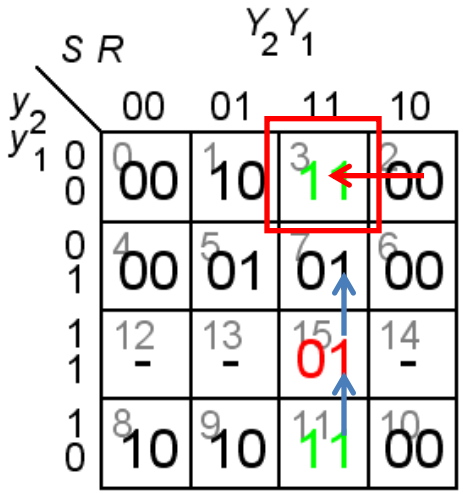
Otur!



$$Y_2 = \overline{S}y_2 + \overline{S}R\overline{y}_1 + Ry_2\overline{y}_1$$

En annan lösning?

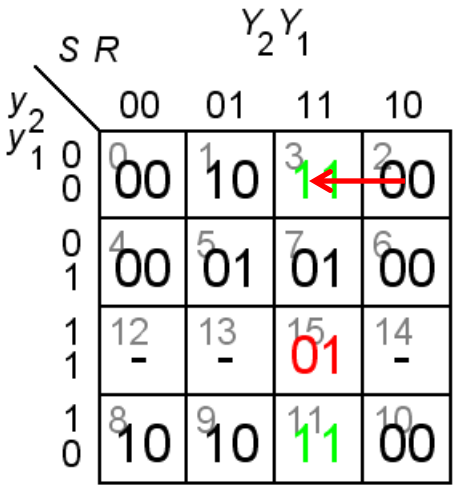
Förutom att lösa problemet med dubbeländringen, som vi redan löst, vill vi få så enkla nät som möjligt!

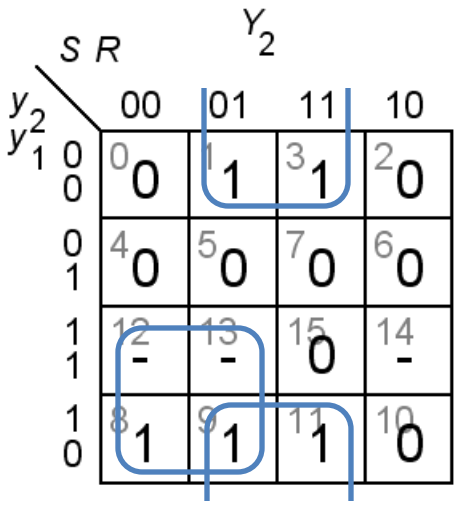


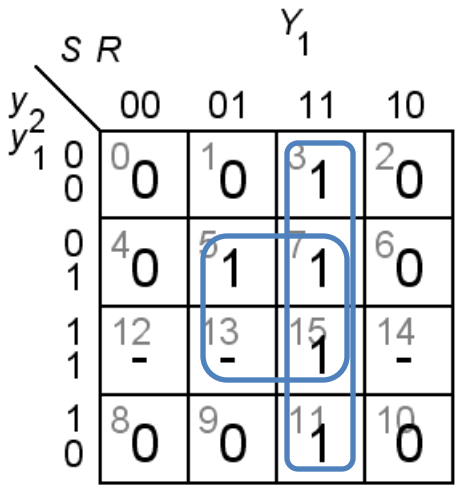
- Vad händer om vi skriver 11 som instabilt tillstånd i ruta 3 (på ren spekulation att detta kommer att ge oss ett enklare nät)?

Från 00 i ruta 2 till 11 i ruta 3 är en *ofarlig* dubbeländring. Blir det 01 hamnar man stabilt i **01**, blir det 10 går man till 11 och till sist stabilt till **01**.

Nya Karnaughdiagram







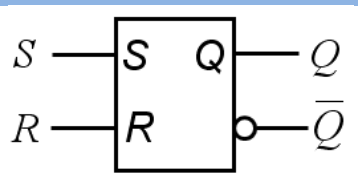
$$Y_2 = \overline{S} y_2 + R \overline{y}_1$$

$$Y_1 = R y_1 + SR$$

- Från 00 till 11 är en *ofarlig* dubbeländring som till sist alltid leder till 01

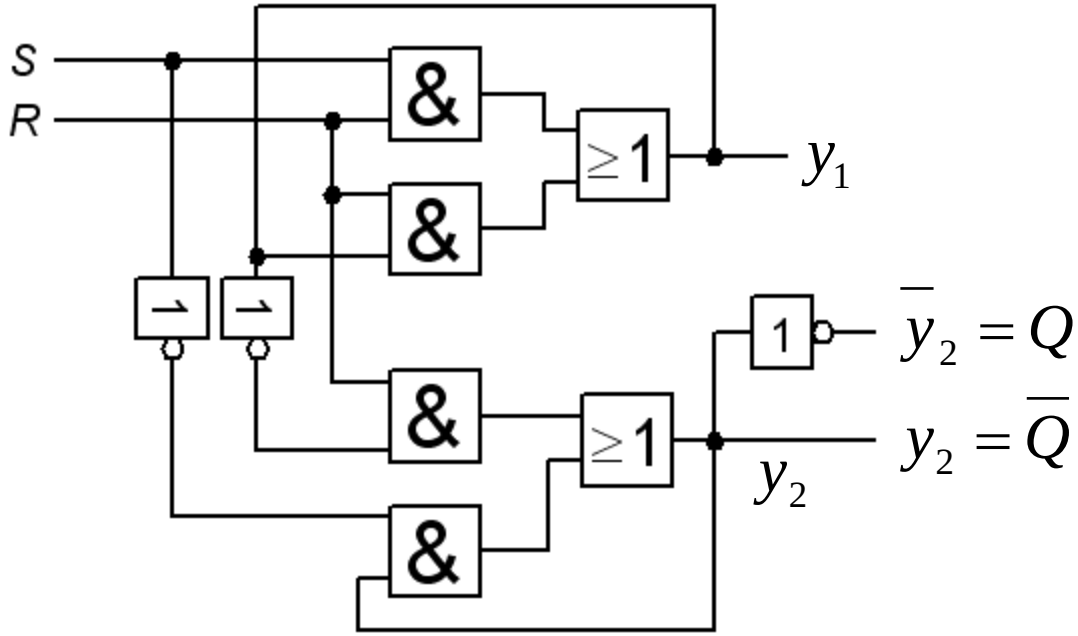
Enklare nät! Vi introducerade en icke-kritisk Hasard och det gav oss större hoptagningar och ett enklare nät!

Idiotsäker *och* kompakt



$$Y_2 = \overline{S} y_2 + R \overline{y}_1$$

$$Y_1 = R y_1 + SR$$



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

Sammanfattning

- Race conditions and Hazards
- Static and dynamic hazards
- Metastability