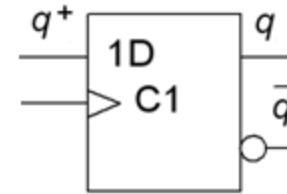
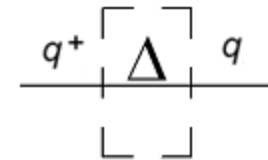


Repetition delay-element

Synkront sekvensnät
Klockad vippa



Asynkront sekvensnät
ett konstgrepp:
Delay-element



Andra beteckningar: Y och y

Gyllene regeln

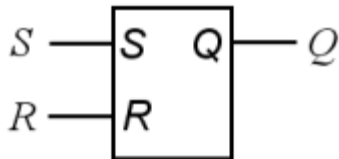


Excitationstabelle

Den asynkrona kodade tillståndstabellen kallas för **Excitationstabelle**

De stabila tillstånden

(de med next state = present state) ”ringas in”



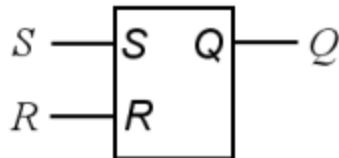
Present state y	Nextstate			
	$SR = 00$	01	10	11
	Y	Y	Y	Y
0	$\textcircled{0}$	$\textcircled{0}$	1	$\textcircled{0}$
1	$\textcircled{1}$	0	$\textcircled{1}$	0

$$Y = y$$

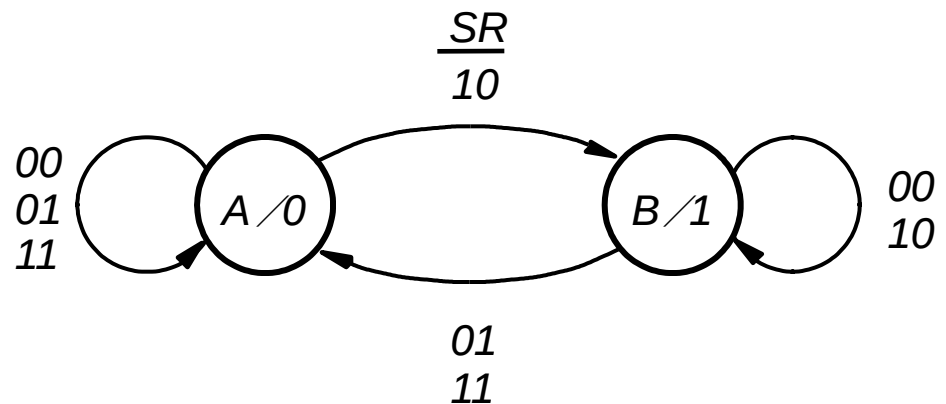
Flödestabell och Tillståndsdigram

Den asynkrona
okodade
tillståndstabellen
kallas för

Flödestabell



Present state	Next state				Output Q
	SR = 00	01	10	11	
A	$\textcircled{A}$	$\textcircled{A}$	B	$\textcircled{A}$	0
B	$\textcircled{B}$	A	$\textcircled{B}$	A	1



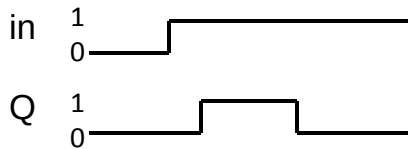
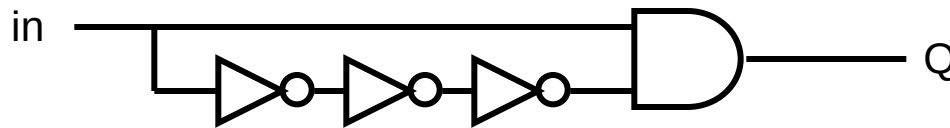
William Sandqvist william@kth.se

Hasard ”spikar”

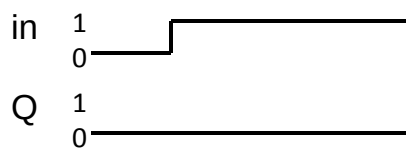
- När man konstruerar asynkrona kretsar så kan det händer att man får spikar (glitches) på signalvärden
- Detta beror på att olika signalvägar har olika fördröjningstider
- Fenomenet kallas för *hasard* (hazard) och kan elimineras med noggrann konstruktion

Snabbfråga

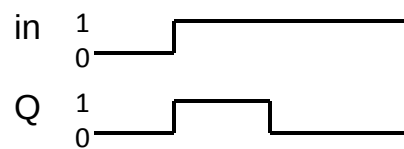
Vilket tidsdiagram motsvarar bäst den signal som genereras av följande grindnät vid stigande flank?



Alt: A



Alt: B

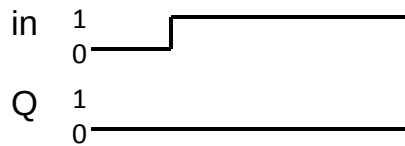
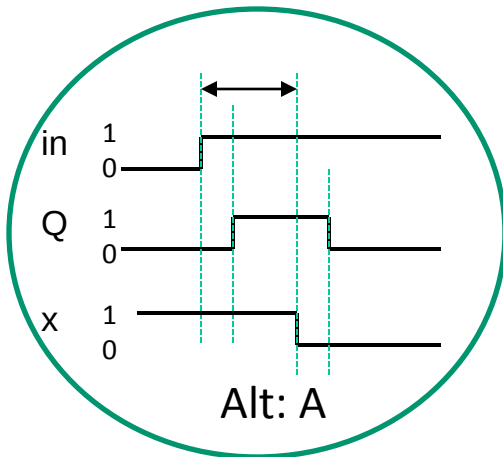
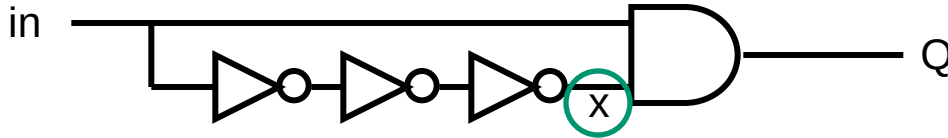


Alt: C

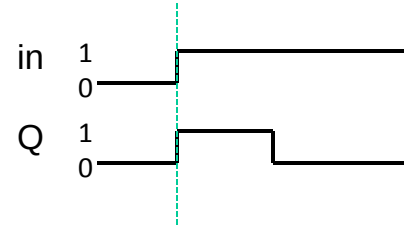


Snabbfråga

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



Alt: B



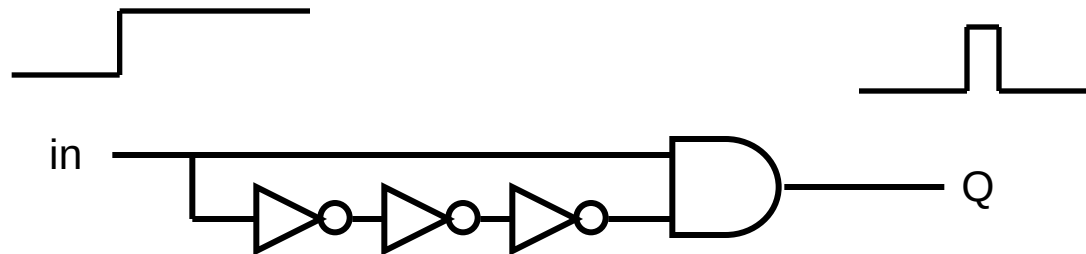
Alt: C

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

Alt C tar ej hänsyn till fördröjning i AND grind

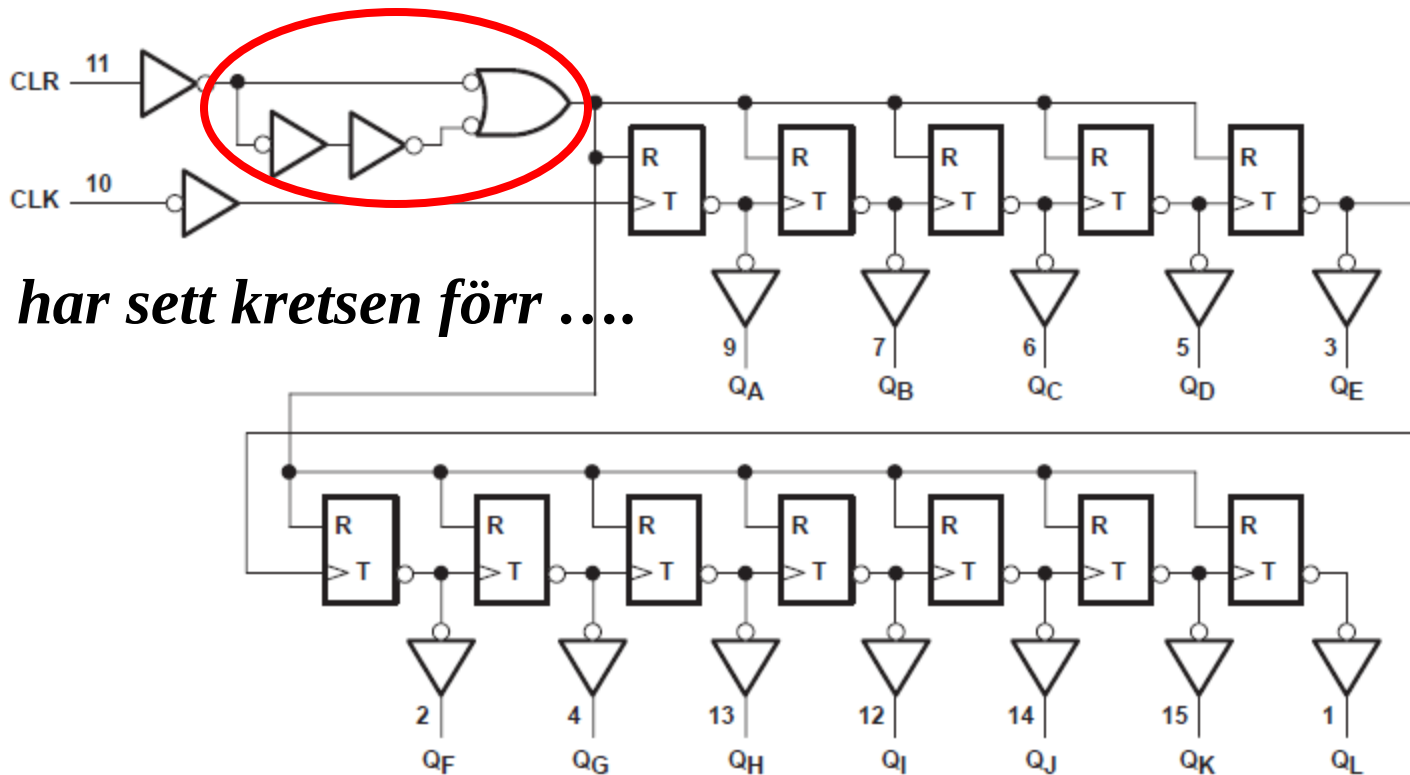


(Kort 0-ställningspuls)



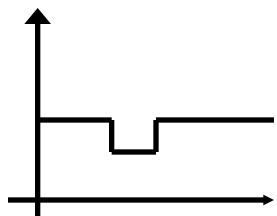
Kretsen används ibland för att generera en kort 0-ställningspuls.

(Kort 0-ställningspuls)

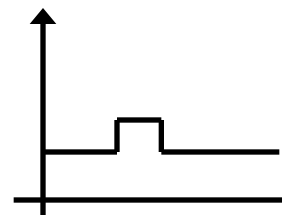


Du har sett kretsen förr

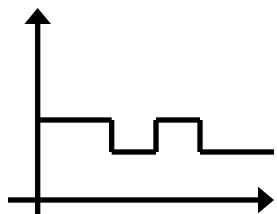
Olika typer av Hasard



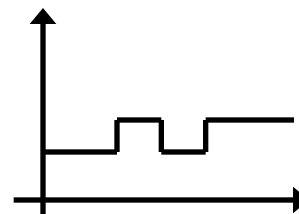
Statisk 1 $\rightarrow$ 1



Statisk 0 $\rightarrow$ 0

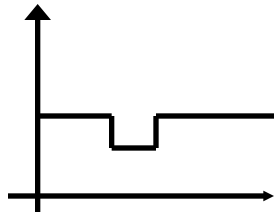


Dynamisk 1 $\rightarrow$ 0

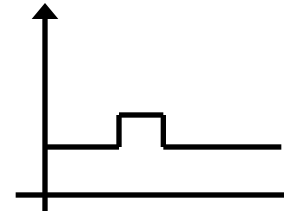


Dynamisk 0 $\rightarrow$ 1

Statisk Hasard



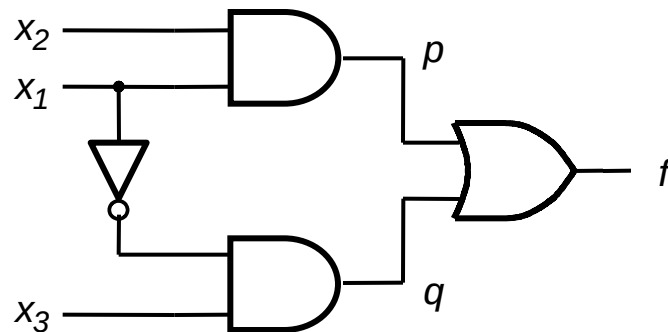
Statisk 1 $\rightarrow 1$



Statisk 0 $\rightarrow 0$

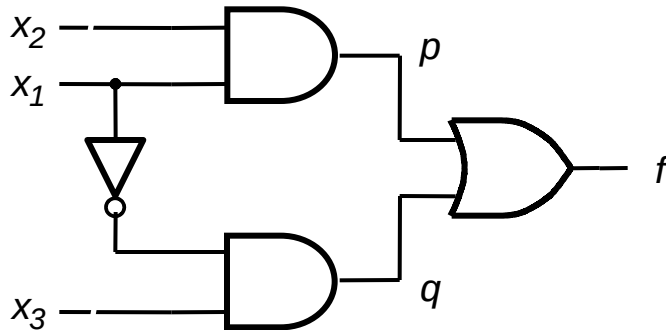
Exempel, Statisk Hasard

- Hasard kan uppträda vid nedanstående krets vid övergången av $x_3x_2x_1$ från **111** $\leftrightarrow$ **110**

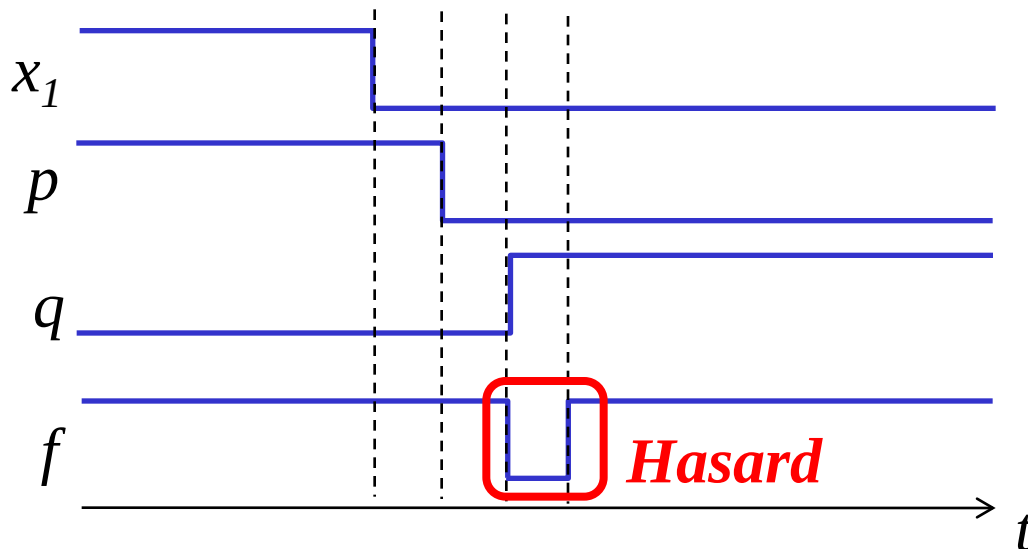


$$f = x_1x_2 + \bar{x}_1x_3$$

Tidsdiagrammet

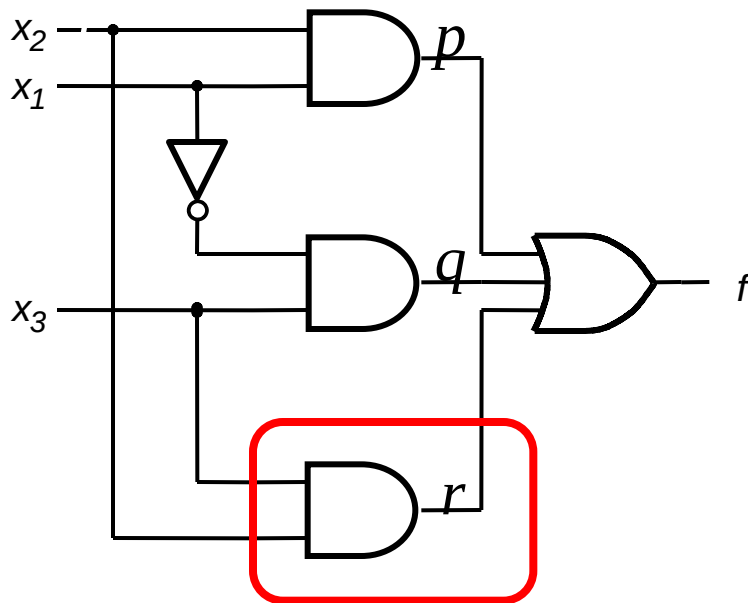


$$f = x_1x_2 + \bar{x}_1x_3$$



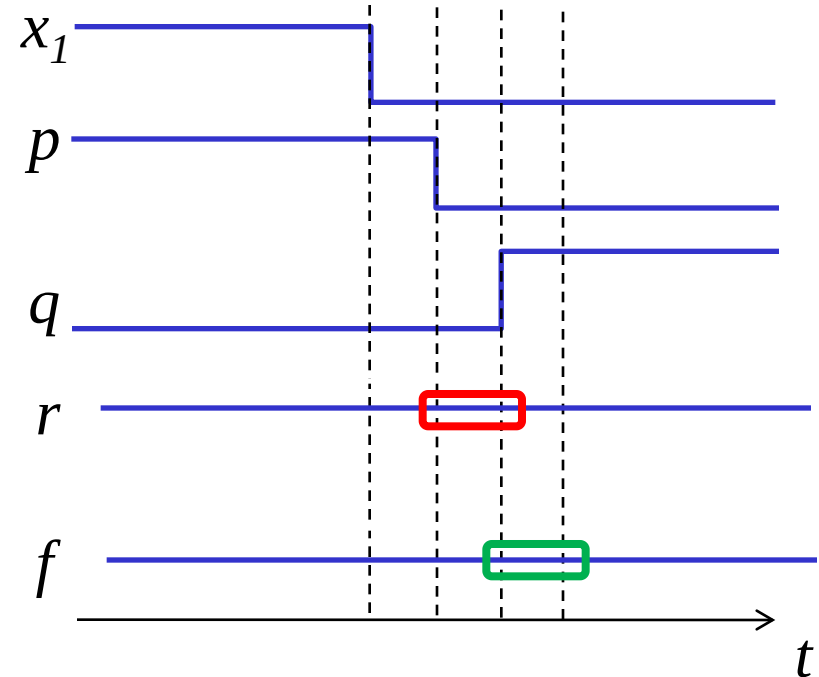
x_1x_2		x_3			
		00	01	11	10
0				1	
1		1	1	1	

Hasardfri krets



Hasard-Cover

x_1x_2		x_3			
		00	01	11	10
0				1	
1	1	1	1		



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

Hur undviker man statisk hasard?

- Möjligheten för statisk hasard finns om två intilliggande 1:or inte är täckta med en egen produktterm vid SOP
- Därmed kan man ta bort risken för statisk hazard genom att lägga till inringningar så att **alla intilliggande** 1:or är täckta med en egen inringning

Ex. Hasardfria hoptagningar

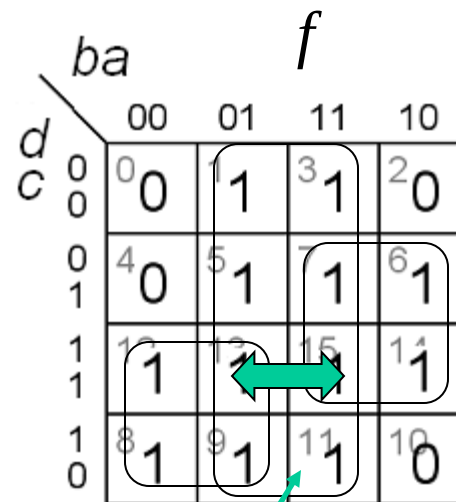
Räcker dessa hoptagningar för hasardfrihet?

		f			
		ba	00	01	11
d c	0	0	1	1	0
	0	0	1	1	1
	1	1	1	1	1
	1	1	1	1	0

$$f = a + d\bar{b} + cb$$

Ex. Hasardfria hoptagningar

Räcker dessa hoptagningar för hasardfrihet?



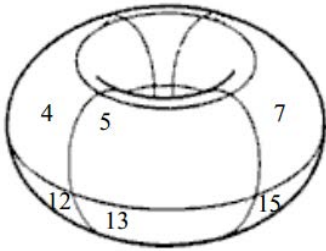
		<i>ba</i>		<i>f</i>		
		00	01	11	10	
<i>d</i>	<i>c</i>	0	0	1	1	0
		0	1	1	1	1
1	1	1	1	1	1	
1	0	1	1	1	0	

$$f = a + d\bar{b} + cb$$

Hasard cover

Ex. Hasardfria hoptagningar

Räcker dessa hoptagningar för hasardfrihet?



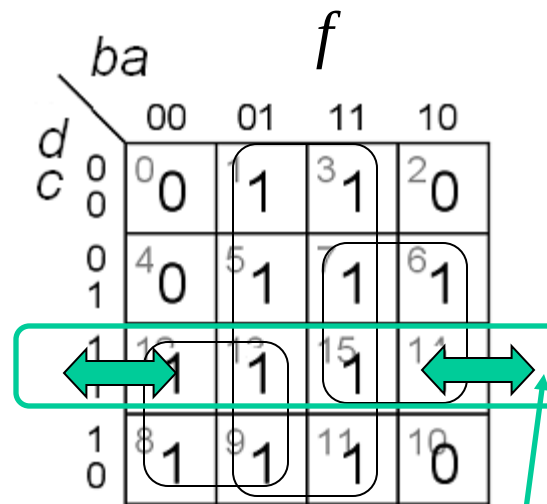
*Karnaugh-
diagrammet är
en doughnut!*

		f							
		ba	00	01	11	10			
d	0	0	0	1	3	1	2	0	
	0	0	4	0	5	1	7	6	1
c	1 <td>1</td> <td>12</td> <td>1</td> <td>13</td> <td>1</td> <td>15</td> <td>14</td> <td>1</td>	1	12	1	13	1	15	14	1
	1 <td>0</td> <td>8</td> <td>1</td> <td>9</td> <td>1</td> <td>11</td> <td>10</td> <td>0</td>	0	8	1	9	1	11	10	0

$$f = a + d\bar{b} + cb$$

Ex. Hasardfria hoptagningar

Räcker dessa hoptagningar för hasardfrihet?



		<i>ba</i>		<i>f</i>				
		00	01	11	10			
<i>d</i>	<i>c</i>	0	0	1	3	1	2	0
		0	4	5	1	7	6	1
1	1	10	13	15	14			
1	0	8	9	11	10			

$$f = a + d\bar{b} + cb + dc$$

Ex. Hasardfria hoptagningar

Lätt att missa!

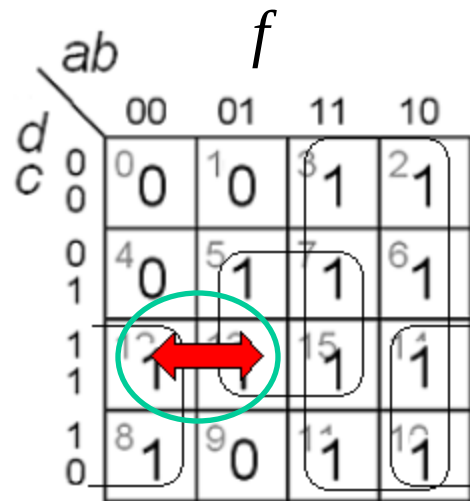
		f			
		ba			
		00	01	11	10
d	0	0	1	3 1	2 0
	0	0	1	1	0
c	0	4 0	5 1	7 1	6 1
	1	0	1	1	1
	1	10 1	12 1	15 1	14 1
	1	1	1	1	1
	1	8 1	9 1	11 1	10 0
	0	1	1	1	0

$$f = a + d\bar{b} + cb + dc$$

$\uparrow$ $\uparrow$
Hasard cover

Ex. Hasardfria hoptagningar

Med annan variabelordning missar man inte!



		ab			
		00	01	11	10
d	0	0	1	1	1
	0	0	0	1	1
c	0	0	1	1	1
	1	1	1	1	1
	1	1	1	1	1
	0	1	0	1	1

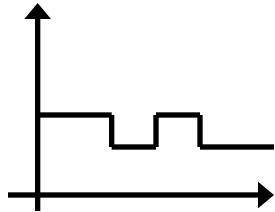
$$f = a + d\bar{b} + cb + \textcircled{dc}$$

Statisk hasard vid POS?

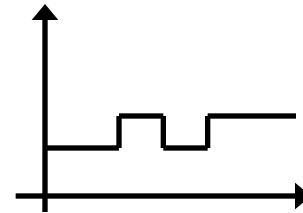
- Har man en POS-implementering så måste man se till att alla bredvidliggande 0:or är täckta av en egen summaterm

Dynamisk Hasard?

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



Dynamisk 1 $\rightarrow$ 0



Dynamisk 0 $\rightarrow$ 1

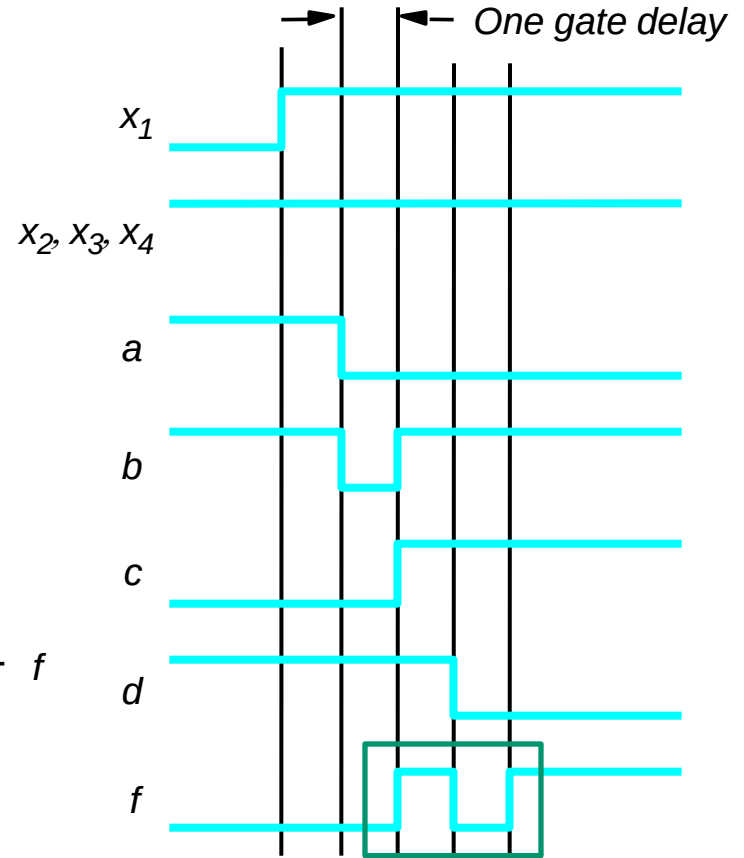
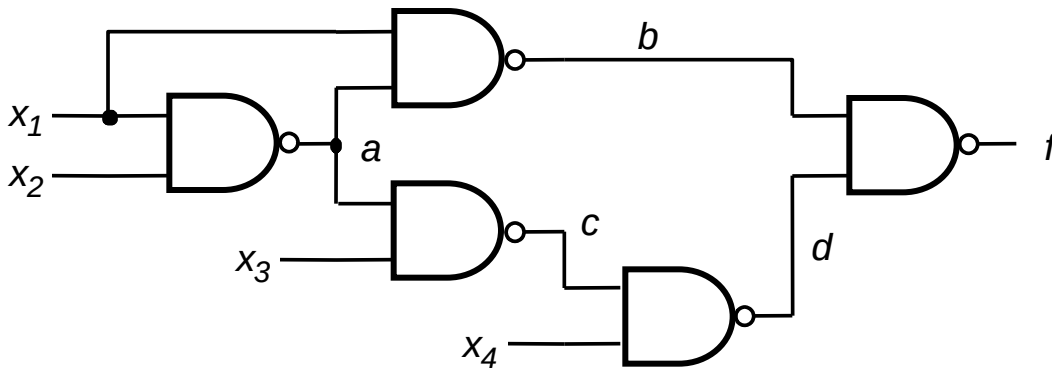
Exempel, Dynamisk Hasard

- Följande ekvation osakar ingen hasard om man implementerar den som en 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 dynamisk hasard

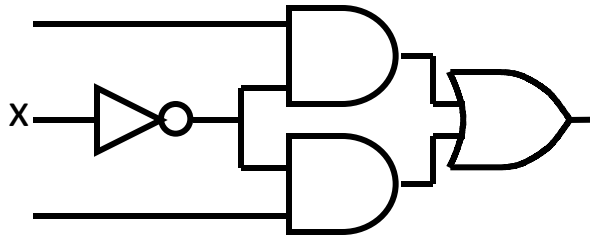


Hur undviks Dynamisk Hasard?

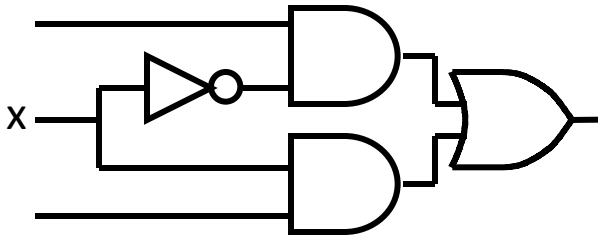
- Dynamisk hasard kan undvikas med **två-nivå-logik**
- Ser man till att en två-nivå krets är fri från statisk hasard, så finns det *inte* heller någon dynamisk hasard!

Snabbfråga

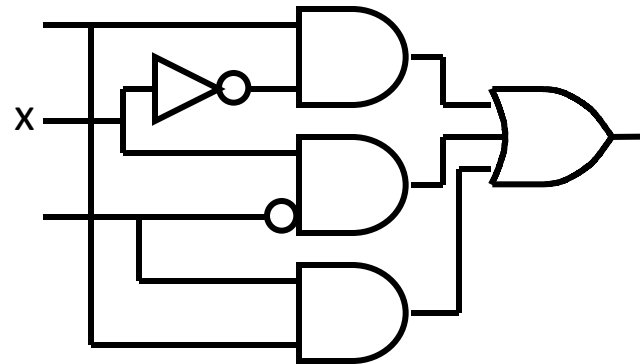
Vilket/vilka av följande grindnät kan ge upphov till hazard då x ändras ?



Alt: B



Alt: A

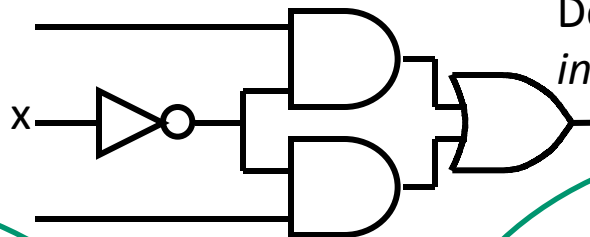


Alt: C

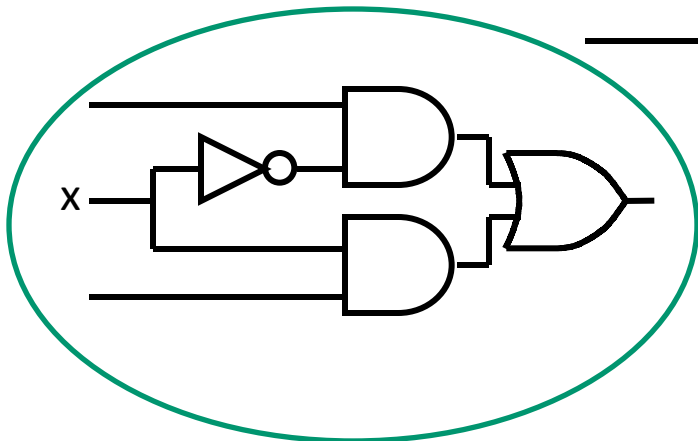


Snabbfråga

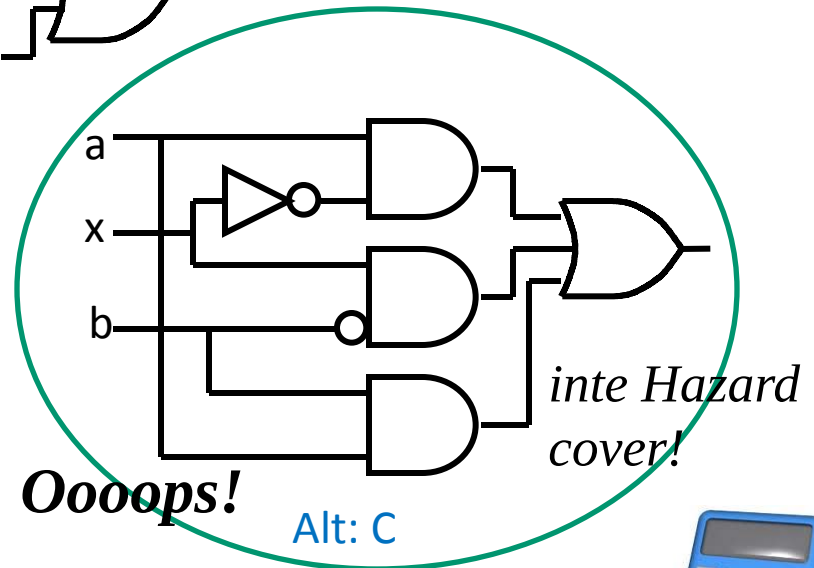
Vilket/vilka av följande grindnät kan ge upphov till hazard då x ändras ?



Alt: B

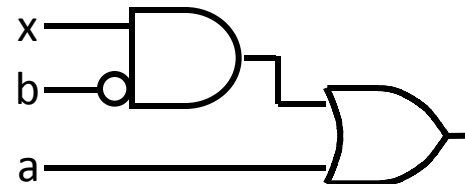
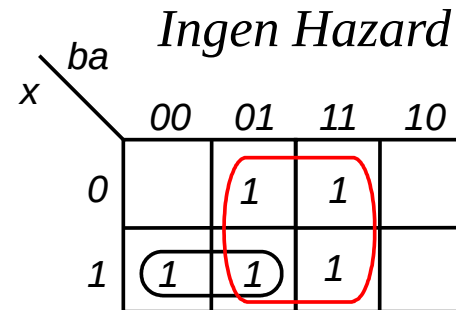
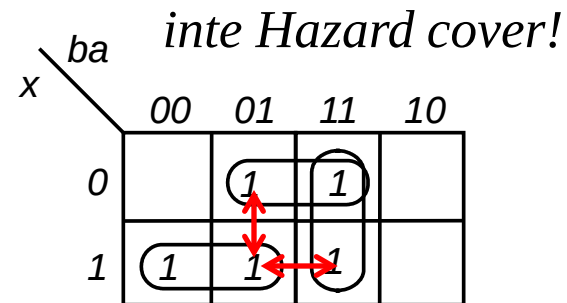
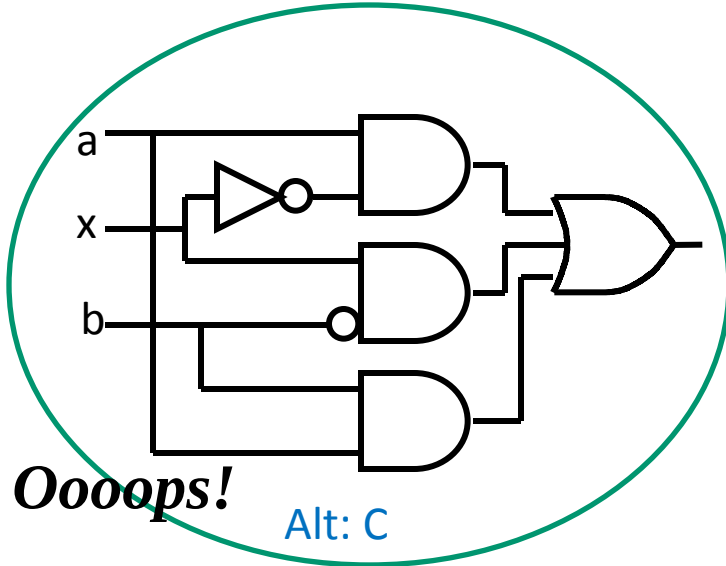


Alt: A



Snabbfråga

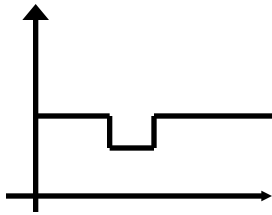
Risk för hazard då $a=1$ och $b=0$
Den extra grinden täcker
inte detta fall (utan $a=1$ och $b=1$)



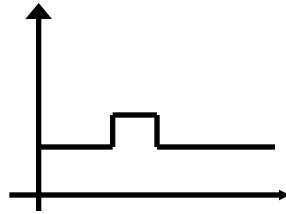
När behöver man ta hänsyn till Hasard?

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

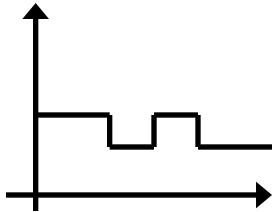
Undvik Hasard



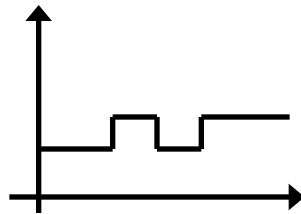
Statisk 1 $\rightarrow$ 1



Statisk 0 $\rightarrow$ 0



Dynamisk 1 $\rightarrow$ 0

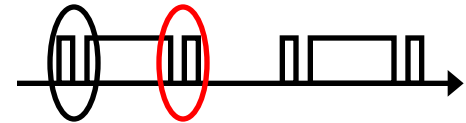


Dynamisk 0 $\rightarrow$ 1

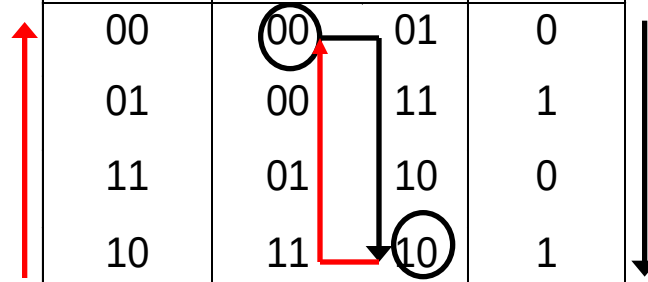
Statisk hasard orsakas av utelämnade primimplikanter

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

Utgångsspikar i asynkrona sekvensnät



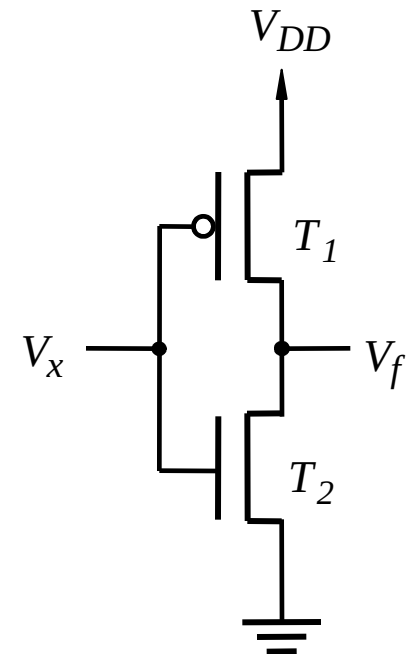
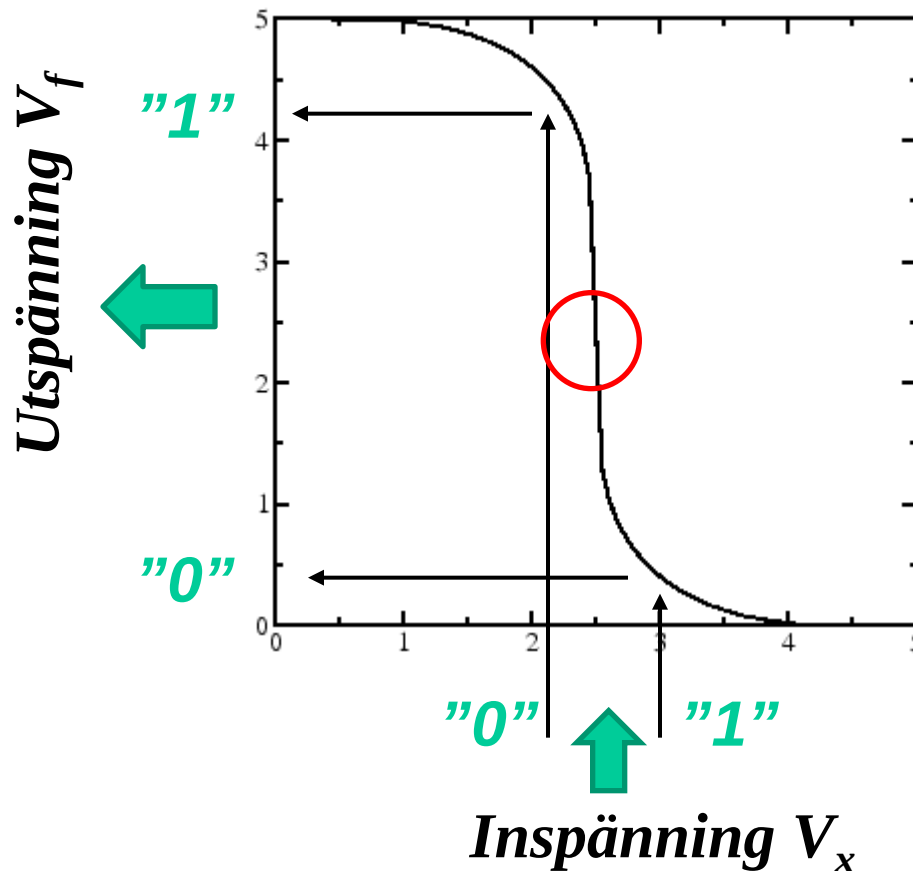
Pres state y_2y_1	Next State		Q
	X=0	1	
	Y_2Y_1		
00	00	01	0
01	00	11	1
11	01	10	0
10	11	10	1



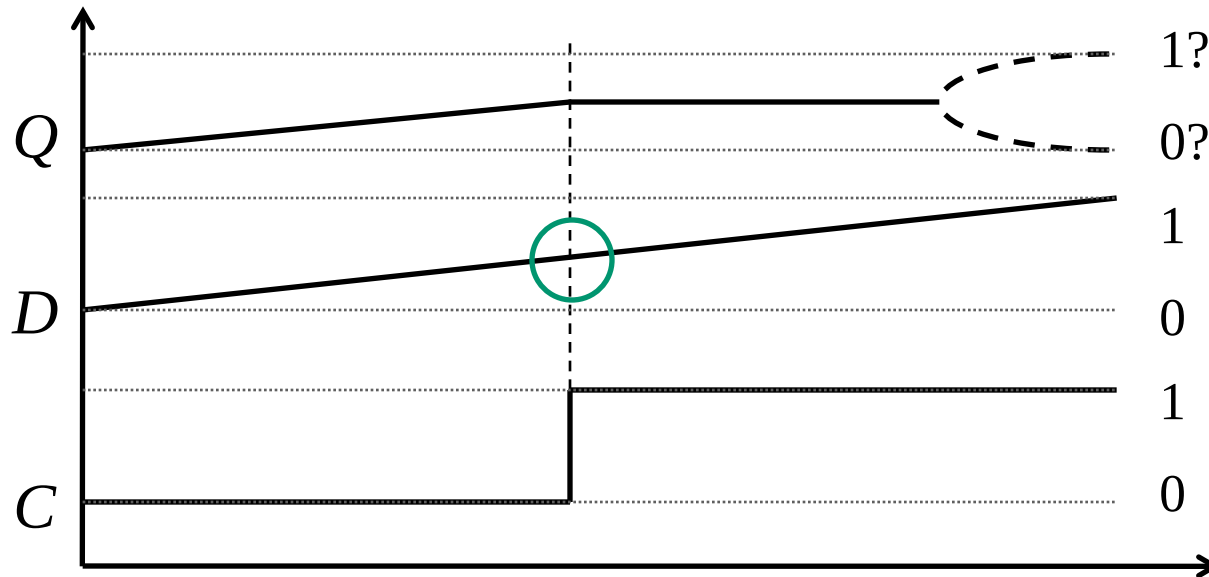
Man kan få utgångsspikar i ett asynkront sekvensnät när man byter från ett stabilt tillstånd till ett annat genom att passerar flera instabila tillstånd (Fenomenet är *ingen* hasard!).

Metastabilitet

CMOS-kretsens överföringsfunktion (ex. inverterare)



Om metastabilitet



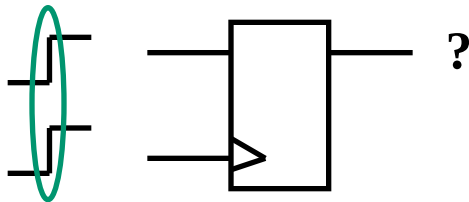
För att förstå vad metastabilitet innebär så kan vi tänka oss att insignalen D till en latch är väldigt belastad och därmed ändrar sig mycket långsamt i förhållande till klockan. Antag vidare att klock-signalen C slår om precis när D är vid $V_{DD}/2$.

Då låser sig latchen vid det spänningsvärde som råkar finnas på D . Efter en tid slår latchen om till antingen '1' eller '0'.

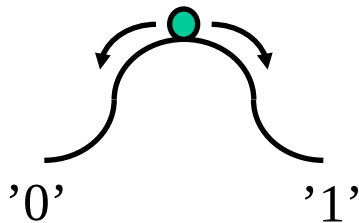
Om metastabilitet ...

Denna instabilitet varar tills transistorerna i återkopplingen behagar gå åt ena eller andra hållet – men det kan ta tid, och tiden beror 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.



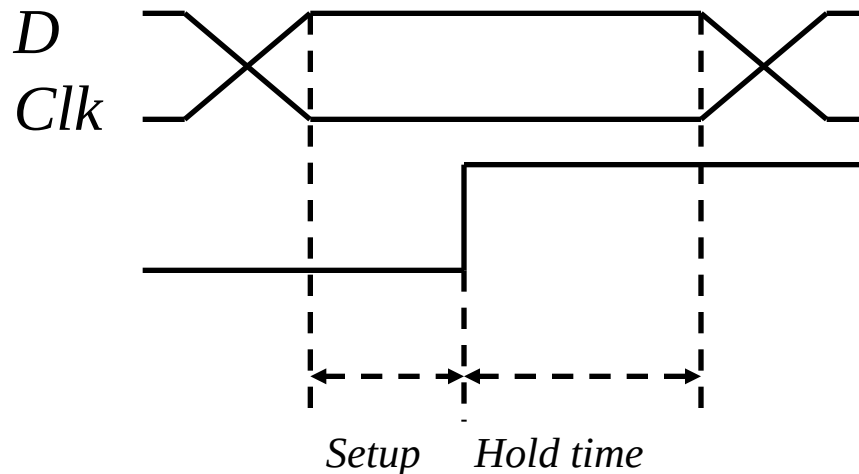
Om *Clk* och *D* switchar samtidigt, vilket värde får då *Q*?



På vilken sida kommer bollen att trilla ner?

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

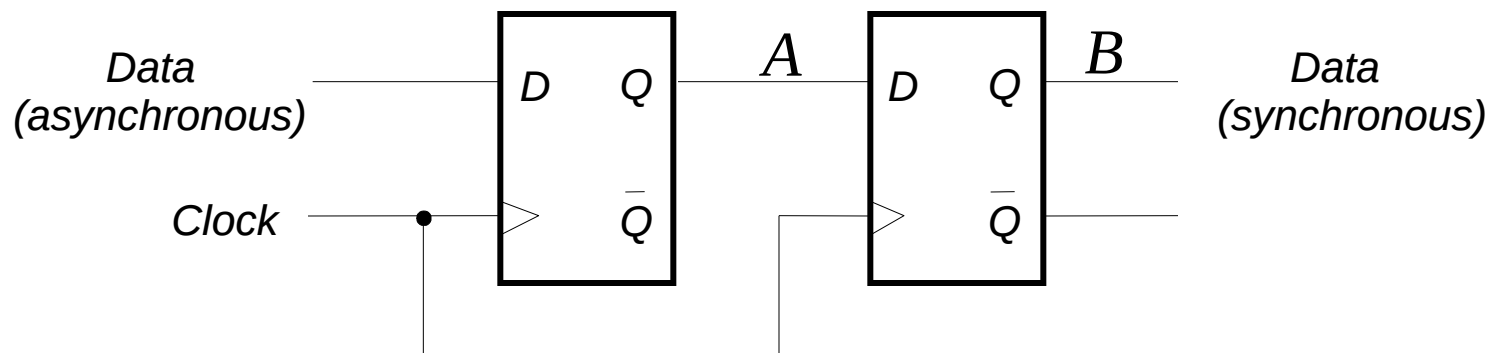
Om Setup and Hold time's är uppfyllda, så kommer vippan (Flip-flop) att garanterat bete sig snällt/deterministiskt!

Asynkrona insignaler?

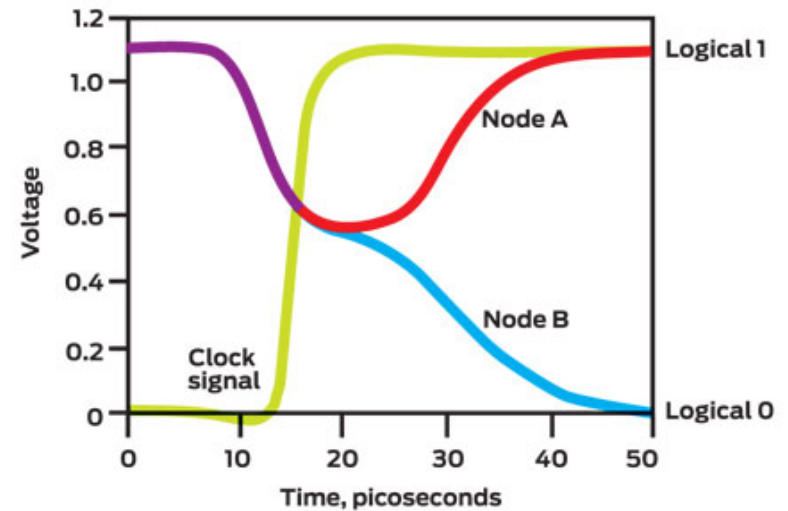
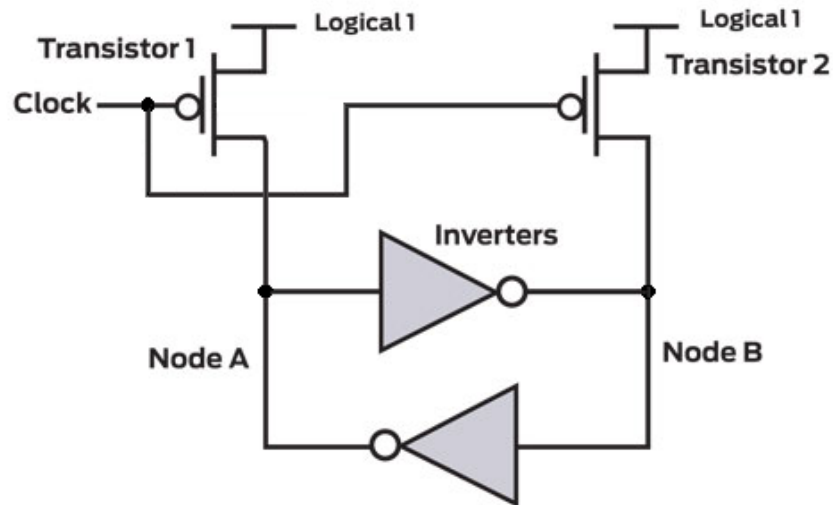
- Dessvärre kan vi inte alltid *garantera* att en ingång är stabil under hela setup- och hold-tiden
- Antag att du kopplar in en tryckknapp på D-ingången av en vipa
 - Användaren kan trycker knappen ***när som helst***, även under setup- och hold-tiden!
 - Risken är att vippan hamnar i ett metastabilt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

Synkronisering av signaler

- För att synkronisera asynkrona ingångar använder man en extra vippa på ingången
- Den första vippans utgång (A) kan hamna 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!



(Slumptal med metastabilitet?)

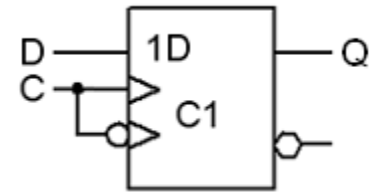


*Intelprocessorer " singlar slant" med följande krets. Innan klockpulsen blir "1" är både node A och node B logiskt "1". När klockpulsen kommer hamnar båda inverterarna i det metastabila tillståndet och **slumpen** avgör sedan vilket tillstånd inverterarna slutgiltigen hamnar i.*

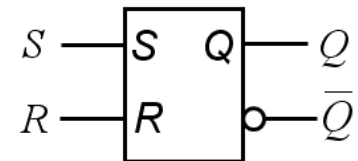
William Sandqvist william@kth.se

Avancerade byggelement

De asynkrona vipporna och låskretsarna används som säkra byggelement vid digital design. Nya byggelement utvecklas hela tiden.



- Vid övningen kommer vi att konstruera en dubbel-flankvippa – en vipptyp som kan komma att ge framtidens datorkretsar högre (dubblerade) prestanda



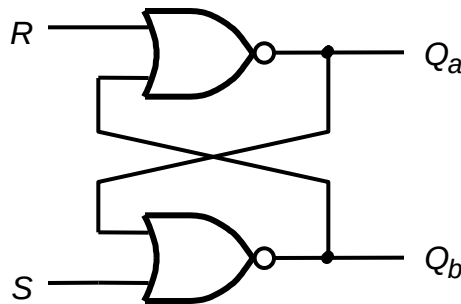
- Vid föreläsningen förfinar vi nu den enkla SR-låskretsen

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

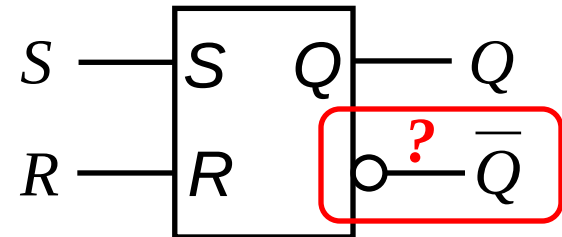
S	R	Q _a	Q _b
0	0	0/1	1/0 (no change)
0	1	0	1
1	0	1	0
1	1	0	0

(b) Truth table

*Förbjuden
insignal $S=R=1$
 $Q_a \neq \bar{Q}_b$*

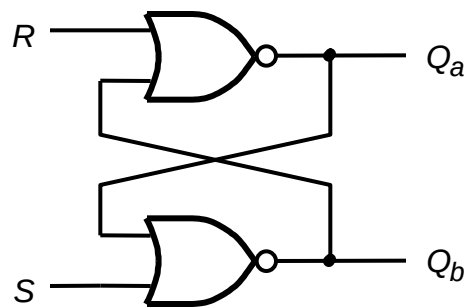
Så länge man **undviker** insignalen $S = R = 1$ (= förbjudet tillstånd) kommer utgångarna Q_a och Q_b att vara varandras inverser. Man kan då använda symbolen till höger.

SR-Latch



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

Mer problem med SR-latchen



(a) Circuit

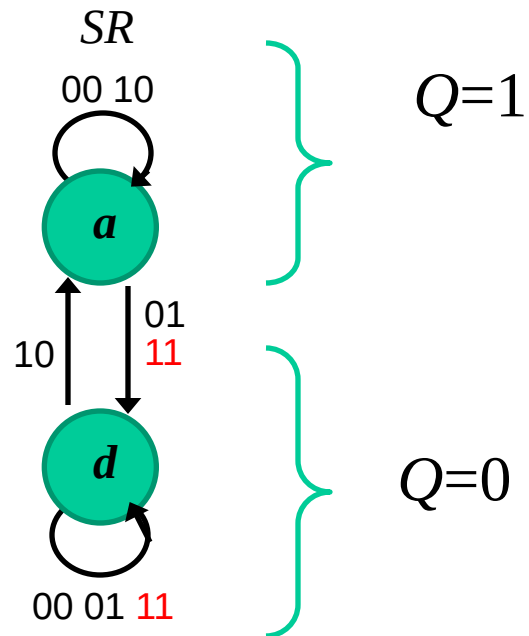
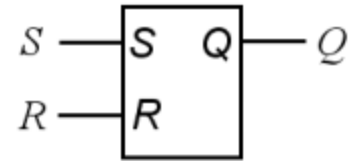
S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

(b) Truth table

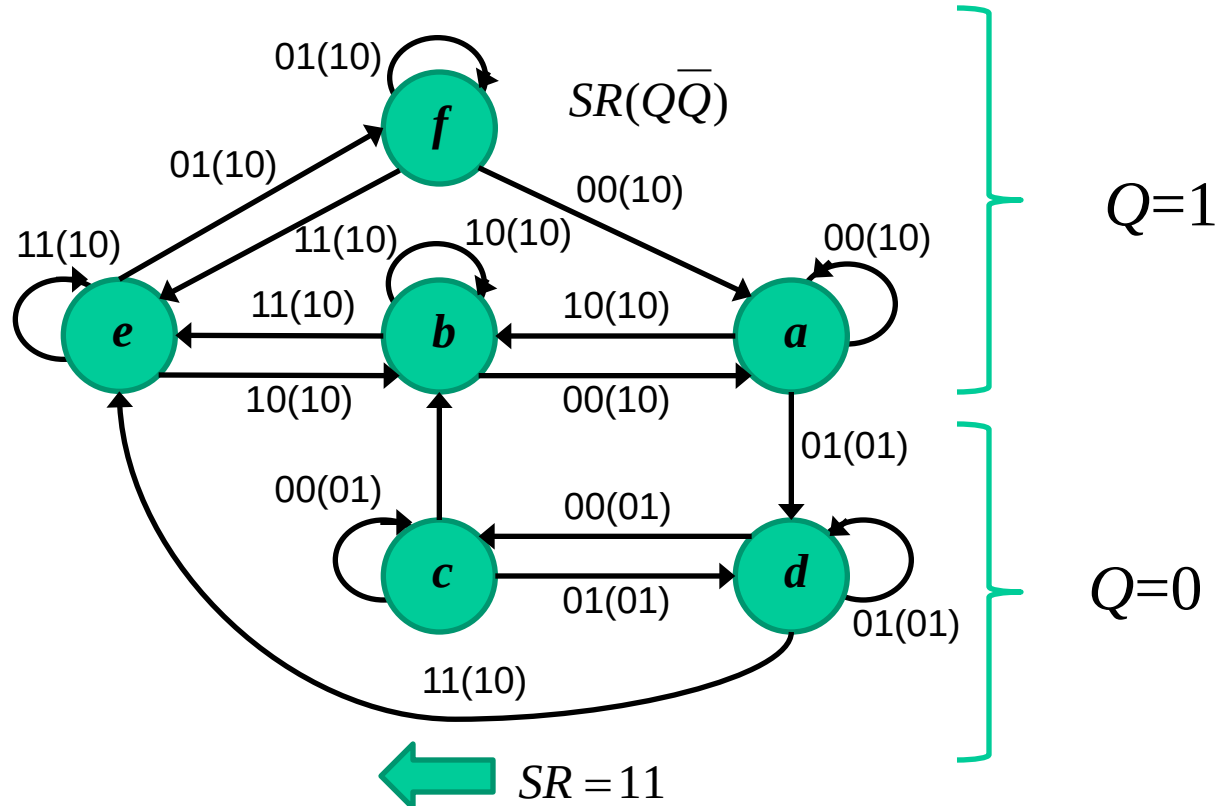
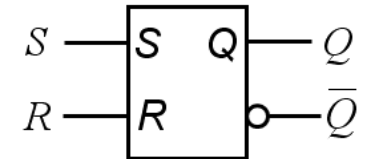
Om man vill gå från $SR = 11$ till $SR = 00$ är det en **dubbeländring** av signalerna. Därför hamnar vi antingen i $Q = 0$ eller i $Q = 1$ ingen kan veta!

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

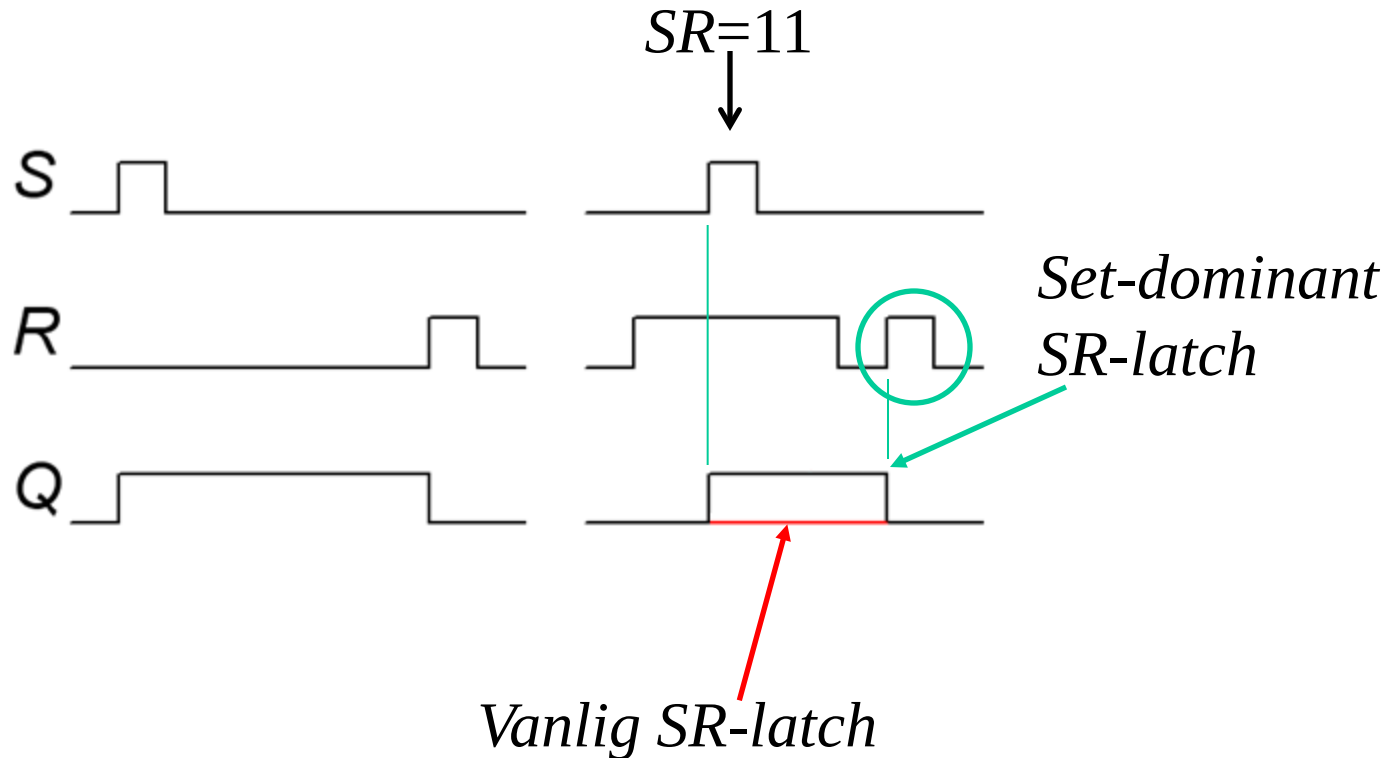
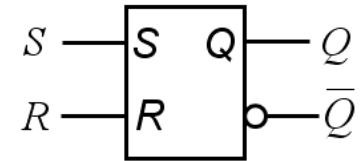
SR-latch



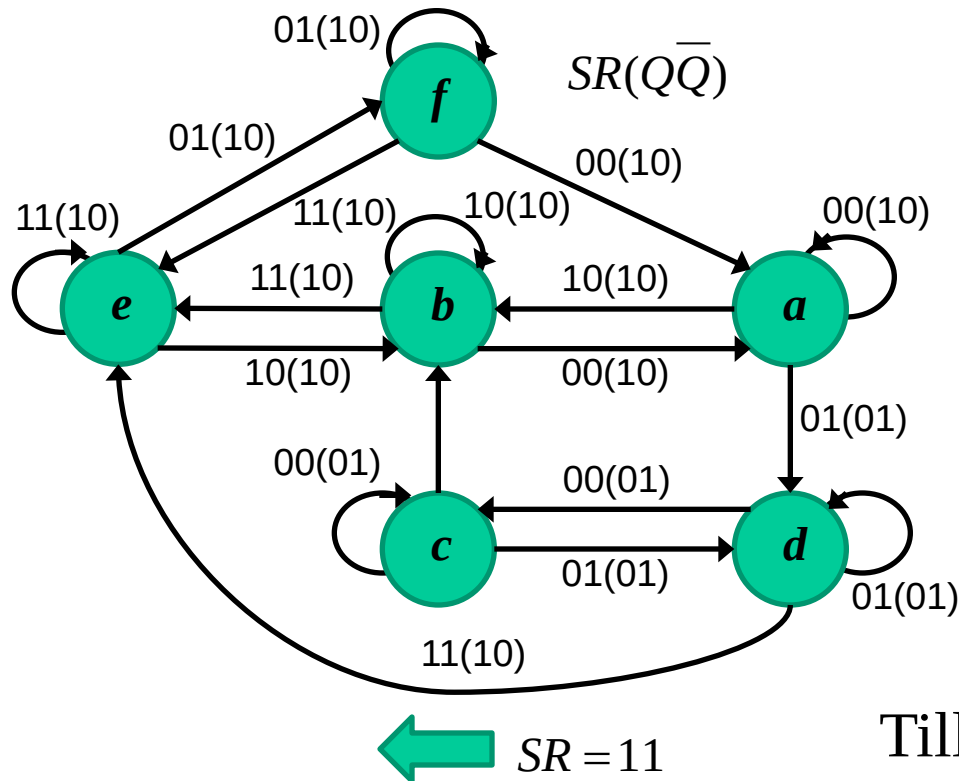
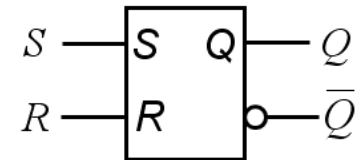
SET-dominant SR-latch



Önskat beteende



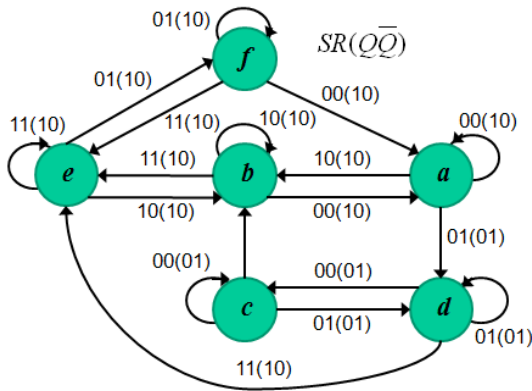
SET-dominant SR-latch



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

Tillståndet e tar "hand om"
fallet $SR = 11$

Kompatibilitet



a (10)

b (10)

c (01)

d (01)

e (10)

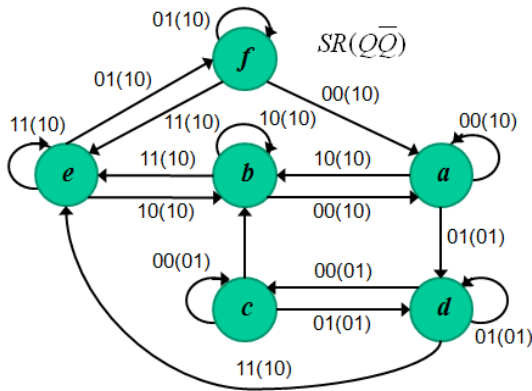
f (10)

$SR(Q\bar{Q})$

	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



$SR(Q\bar{Q})$

	00	01	11	10	Q	$\overline{Q}$
a	$\textcircled{a}$	d	$-$	b	1	0
b	a	$-$	e	$\textcircled{b}$	1	0
c	$\textcircled{c}$	d	$-$	b	0	1
d	c	$\textcircled{d}$	e	$-$	0	1
e	$-$	f	$\textcircled{e}$	b	1	0
f	a	$\textcircled{f}$	e	$-$	1	0

a (10)



b (10)



c (01)



d (01)



e (10)



f (10)



Många valmöjligheter ...

a(10) : **ad** - b

b(10) : a - **eb**

b(10) : a - **eb**

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

b(10) : a - **eb**

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

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

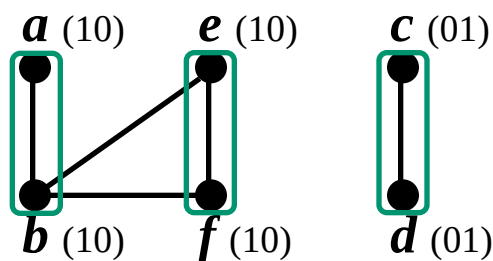
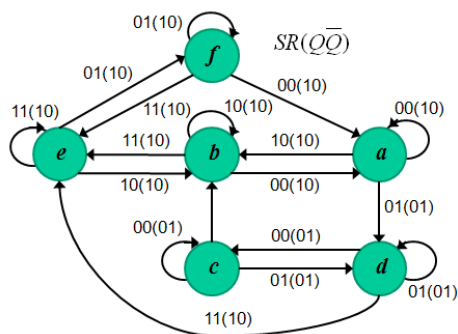
f(10) : a **f** e -

c(01) : **cd** - b

d(01) : **cde** -

Kompatibilitetsgraf

Många valmöjligheter ...



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

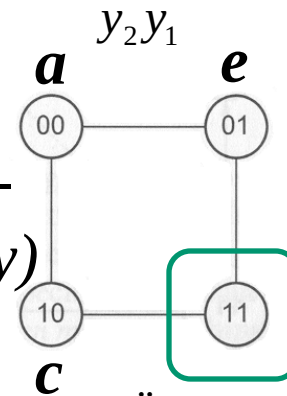
Reducerad flödestabell

Tillståndskodning

Utgångsavkodning

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

Vald
tillstånds-
kod (Gray)



Övergångs-tillstånd

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

	$S R$		$Y_2 Y_1$			
			00	01	11	10
$y_2 \backslash y_1$	0	1				
a :	0	0	$\textcircled{a}$	1 c	3 e	2 $\textcircled{a}$
e :	0	1	4 a	5 $\textcircled{e}$	7 $\textcircled{e}$	6 a
	1	1	12 —	13 —	15 ?	14 —
c :	1	0	8 $\textcircled{c}$	9 $\textcircled{c}$	11 e	10 a

	$S R$		$Y_2 Y_1$			
			00	01	11	10
$y_2 \backslash y_1$	0	1				
0	0	0	00	1 10	3 01	2 00
0	1	0	4 00	5 01	7 01	6 00
1	1	1	12 —	13 —	15 01	14 —
1	0	0	8 10	9 10	11 11	10 00

Från c till e krävs det en ”dubbeländring” av $Y_2 Y_1$ detta ändras med hjälp av **övergångstillståndet** till två ”enkeländringar”

Karnaughdiagram

SR		$Y_2 Y_1$			
		00	01	11	10
y_2	0	0	1	3	2
	0	00	10	01	00
y_1	0	4	5	7	6
	1	00	01	01	00
y_2	1	12	13	15	14
	1	-	-	01	-
y_1	1	8	9	11	10
	0	10	10	11	00

SR		Y_2			
		00	01	11	10
y_2	0	0	1	3	2
	0	0	1	0	0
y_1	0	4	5	7	6
	1	0	0	0	0
y_2	1	12	13	15	14
	1	-	-	0	-
y_1	1	8	9	11	10
	0	1	1	1	0

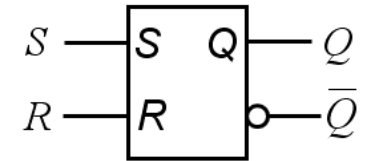
SR		Y_1			
		00	01	11	10
y_2	0	0	1	3	2
	0	0	0	1	0
y_1	0	4	5	7	6
	1	0	1	1	0
y_2	1	12	13	15	14
	1	-	-	1	-
y_1	1	8	9	11	10
	0	0	0	1	0

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

$$Y_1 = Ry_1 + SR$$

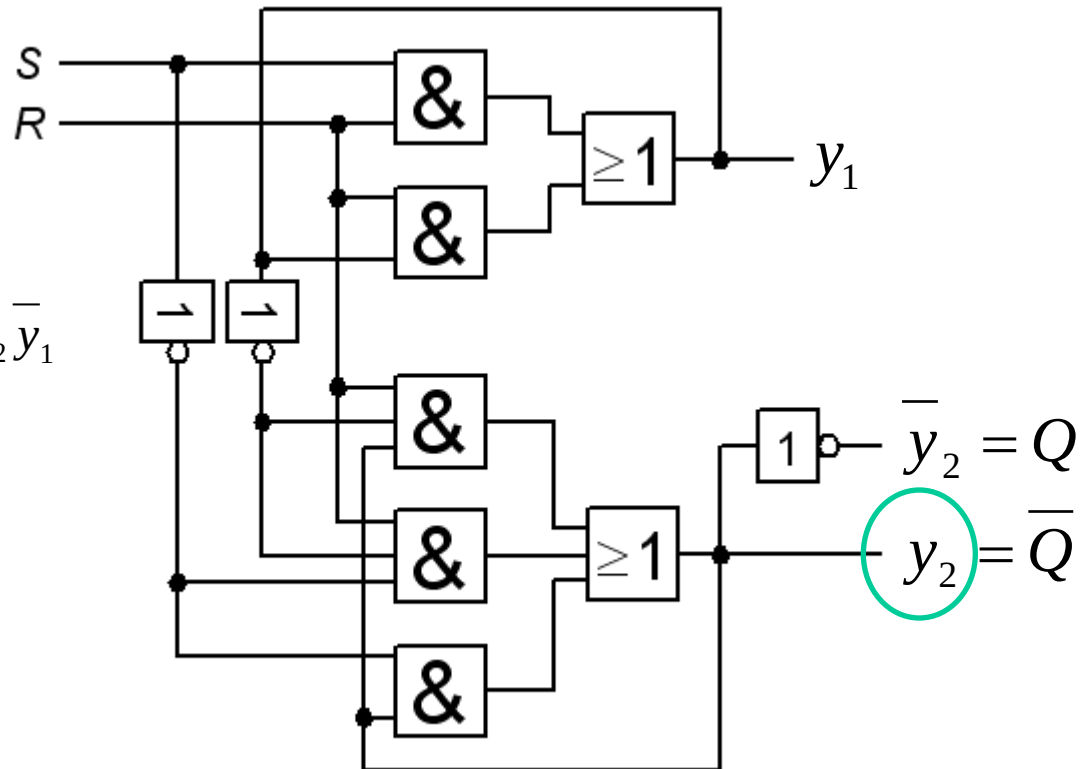
Hasardfria nät direkt!

Krets-schema



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

$$Y_1 = Ry_1 + SR$$



Här har vi vår "idiotsäkra" SR-låskrets!

Otur!

S R		Y ₂			
		00	01	11	10
y ₂	y ₁ 0	0 0	1 1	3 0	2 0
	0 1	4 0	5 0	7 0	6 0
1	1 1	12 -	13 -	15 0	14 -
	1 0	8 1	9 1	11 1	10 0

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

En annan lösning?

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

		$Y_2 Y_1$			
		00	01	11	10
y_2	0	0 00	1 10	3 10	2 00
	1	4 00	5 01	6 01	7 00
y_1	1	12 -	13 -	15 01	14 -
	0	8 10	9 10	11 11	10 00

- Vad händer om vi skriver 11 (i stället för 01) som instabilt tillstånd i ruta 3, på ren spekulaton 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

S R		$Y_2 Y_1$			
		00	01	11	10
y_2	y_1				
0	0	0	1	3	2
0	0	00	10	1	00
0	1	4	5	7	6
0	1	00	01	01	00
1	1	12	13	15	14
1	1	-	-	01	-
1	0	8	9	11	10
1	0	10	10	11	00

S R		Y_2			
		00	01	11	10
y_2	y_1				
0	0	0	1	3	2
0	0	0	1	1	0
0	1	4	5	7	6
0	1	0	0	0	0
1	1	12	13	15	14
1	1	-	-	0	-
1	0	8	9	11	10
1	0	1	1	1	0

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

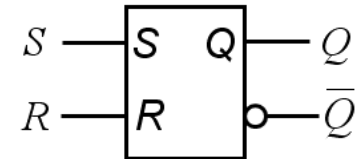
S R		Y_1			
		00	01	11	10
y_2	y_1				
0	0	0	1	3	2
0	0	0	0	1	0
0	1	4	5	7	6
0	1	0	1	1	0
1	1	12	13	15	14
1	1	-	-	1	-
1	0	8	9	11	10
1	0	0	0	1	0

$$Y_1 = Ry_1 + SR$$

- Från 00 till 11 innebär en *ofarlig* dubbeländring av tillståndsvariablerna som till sist alltid leder till stabilt 01.

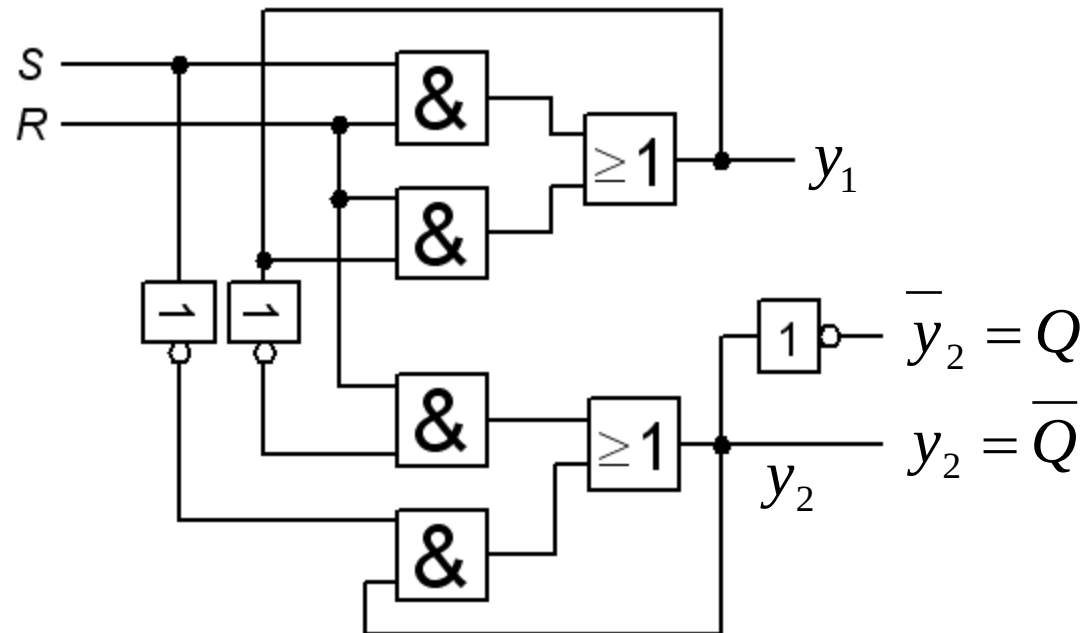
Enklare nät! Vi introducerade en icke-kritisk Hasard och det gav oss större hoptagningar och ett enklare nät!

Idiotsäker *och* kompakt



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

$$Y_1 = Ry_1 + SR$$



Asynkrona nät är byggstenar

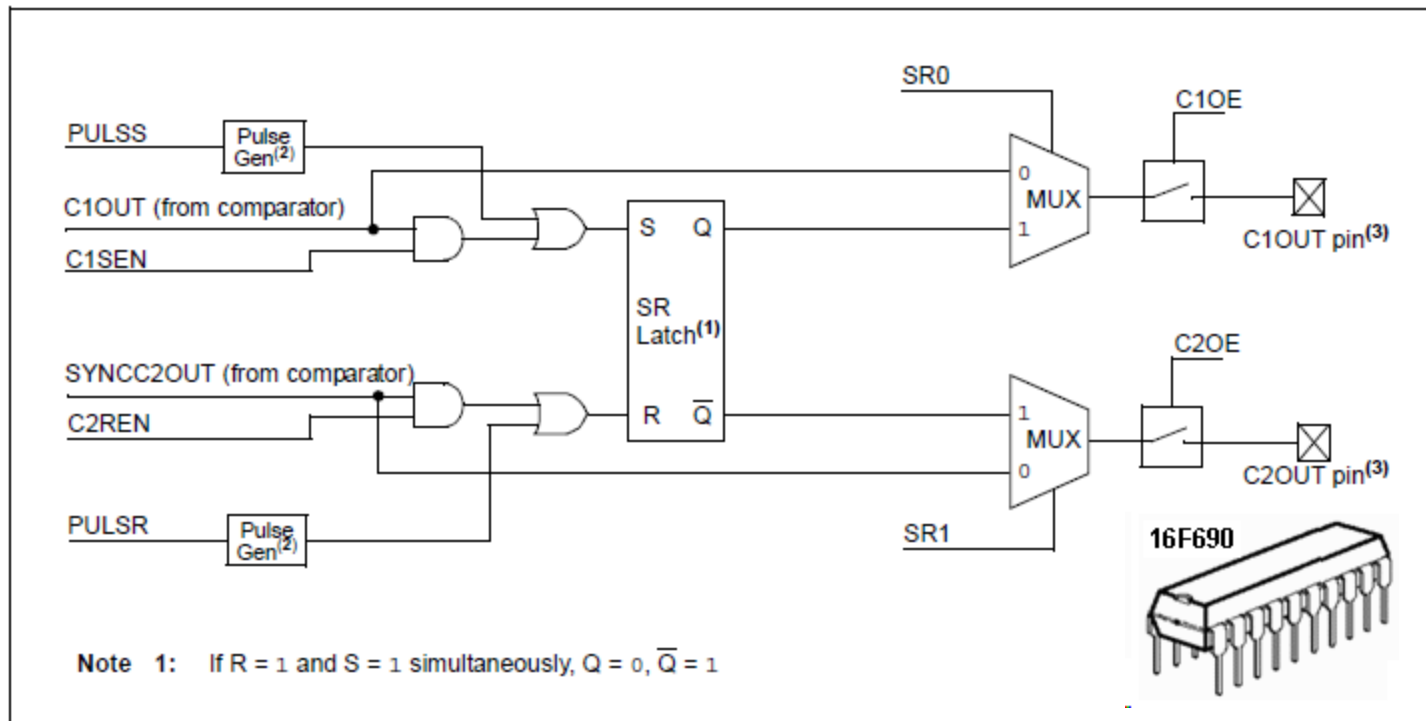
Eftersom de asynkrona sekvensnäten används som byggstenar vid all annan Digital Design är det vanligt att stor möda har lagts på att göra dem så optimala som möjligt.

De används oftare i tusental i en konstruktion än styckvis. Varje ingående grind ”kostar” och räknas!

Används dominanta SR-latchar?

PIC16F690 IO-enhet, SR-latch

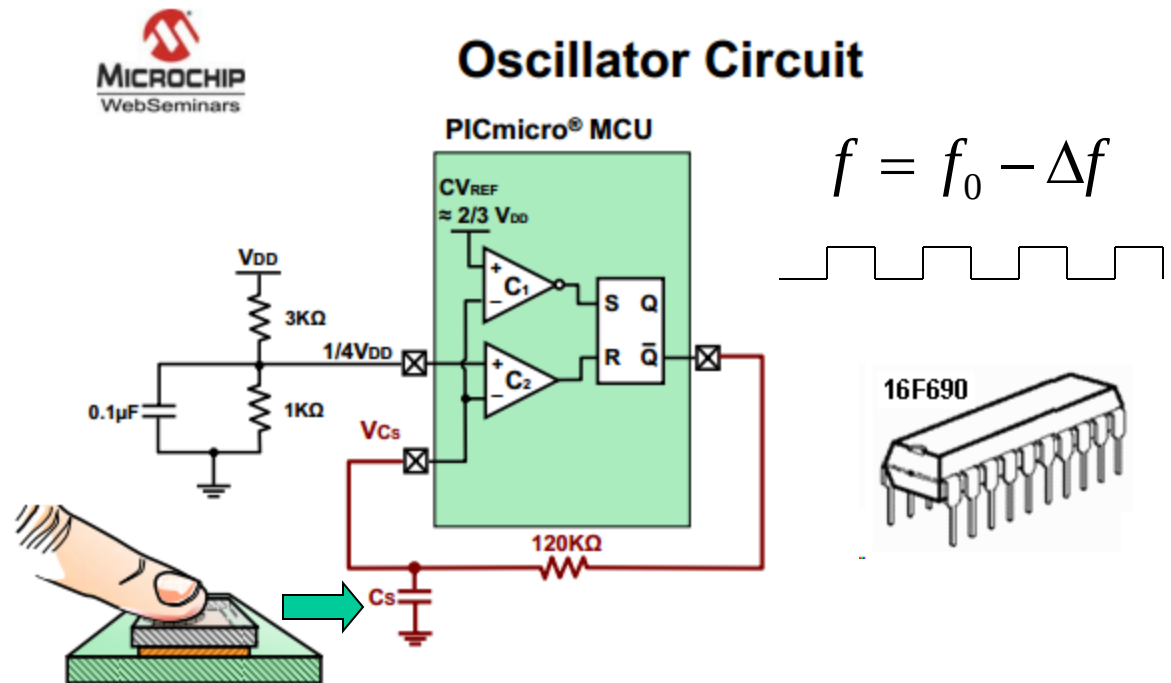
FIGURE 8-7: SR LATCH SIMPLIFIED BLOCK DIAGRAM



Låskretsen är **RESET-dominant** SR 11 $\rightarrow Q = 0 \quad \bar{Q} = 1$

PIC16F690 IO-enhet, SR-latch

Touch control. Användningsområdet för SR-latchen är en kapacitivt styrd oscillator. Den ändrar frekvens vid en "touch" med fingret.



William Sandqvist william@kth.se