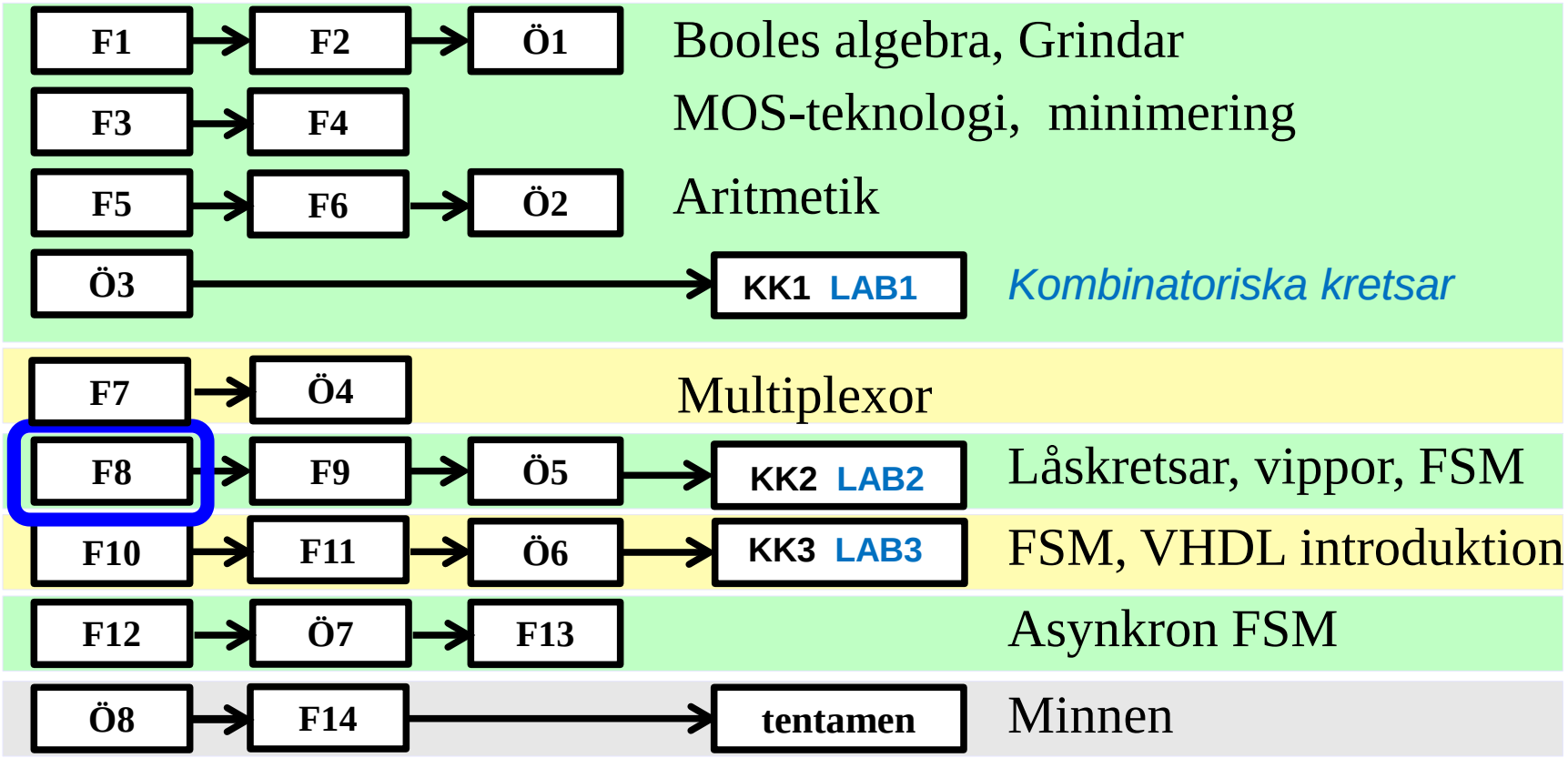
Digital Design IE1204

F8 Vippor och låskretsar, räknare

william@kth.se

IE1204 Digital Design



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

William Sandqvist william@kth.se

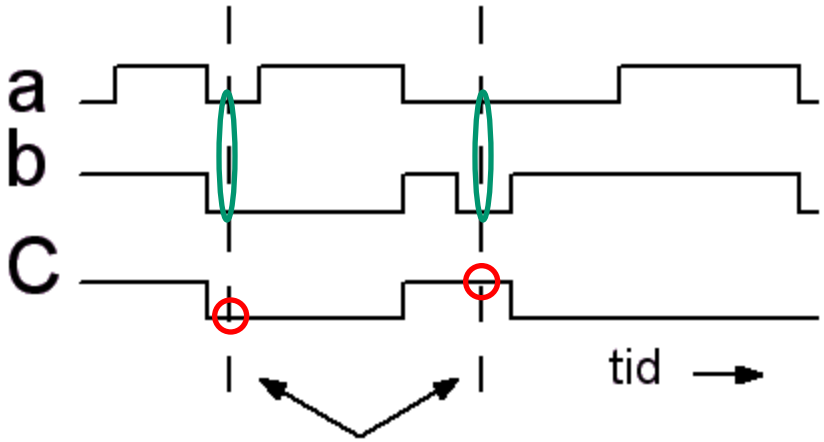
Detta har hänt i kursen …

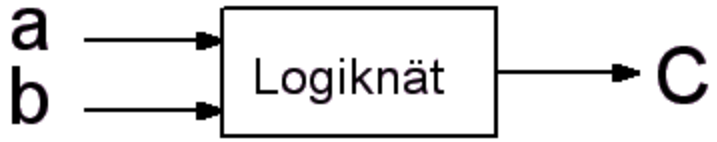
Decimala, hexadecimala, oktala och binära talsystemen
AND OR NOT EXOR EXNOR Sanningstabell, mintermer Maxtermer PS-form
Booles algebra SP-form deMorgans lag Bubbelgrindar Fullständig logik
NAND NOR CMOS grindar, standardkretsar Minimering med Karnaugh-diagram 2, 3, 4, 5, 6 variabler
Registeraritmetik tvåkomplementrepresentation av binära tal
Additionskretsar Multiplikationskrets Divisionskrets

Multiplexorer och Shannon dekomposition Dekoder/Demultiplexor Enkoder
Prioritetsenkoder Kodomvandlare
VHDL introduktion

William Sandqvist william@kth.se

Sekvensnät





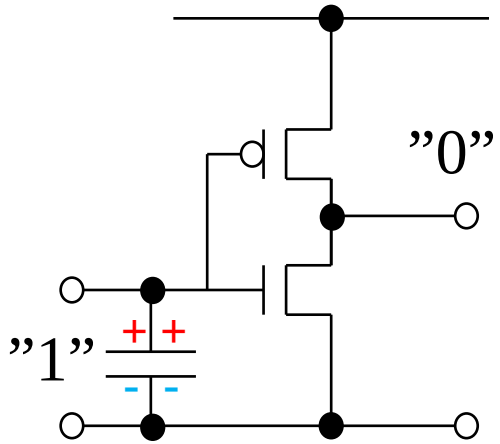
Om en och samma insignal kan ge upphov till olika utsignal, är logiknätet ett sekvensnät.
Det måste då ha ett ***inre minne*** som gör att utsignalen påverkas av både nuvarande och föregående insignaler!

William Sandqvist william@kth.se

Hur minns hårdvara?

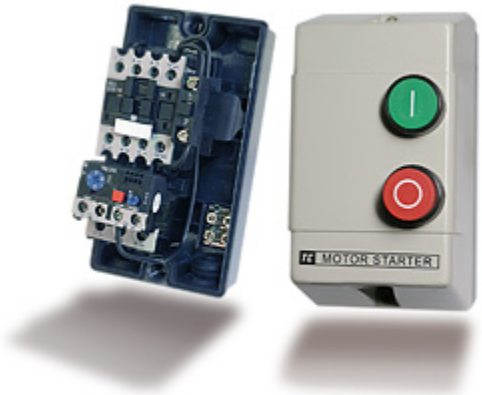
- För att minnas någonting, så måste vi på något sätt hålla kvar informationen.

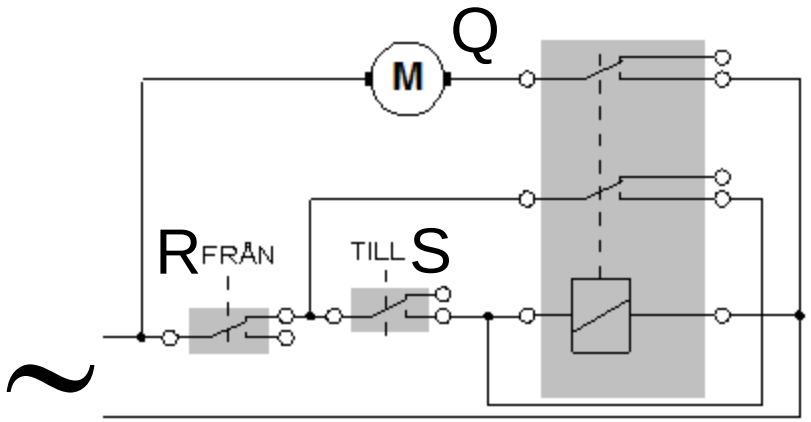
- Ett sätt är att lagra information i form av en laddning på en kapacitans (DRAM).

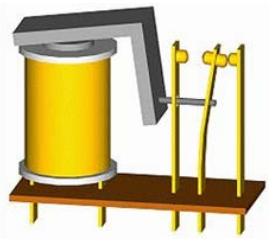


Det finns andra möjligheter …

(Motorskydd)







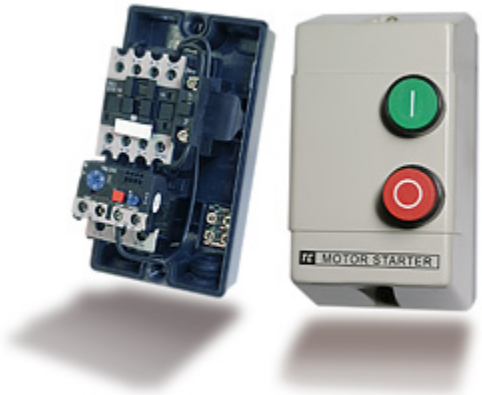
Relä

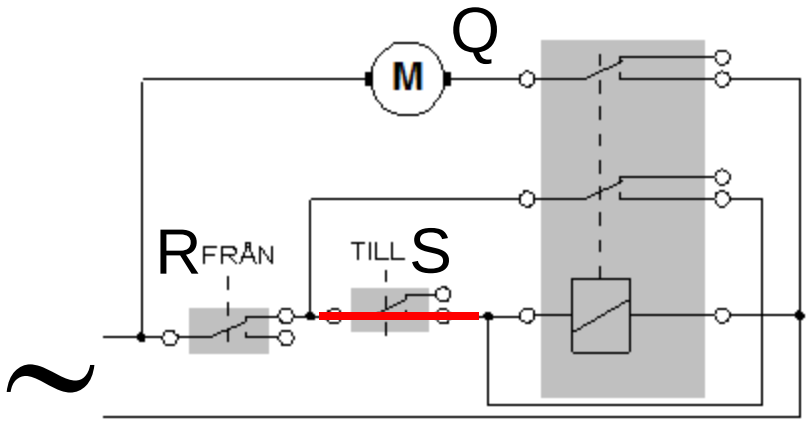
Ett **motorskydd** är ett relä med en **självhållning**s kontakt.

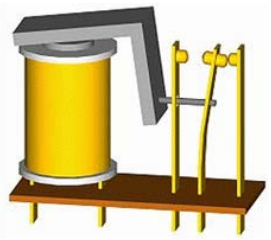
- Man behöver bara trycka kort för att motorn ska starta.
- Blir det strömavbrott så startar *inte* motorn plötsligt av sig själv när strömmen kommer tillbaks – en bra säkerhetsdetalj.
- Lamporna i lokalen däremot tänds direkt – också det bra.

William Sandqvist william@kth.se

(Motorskydd)







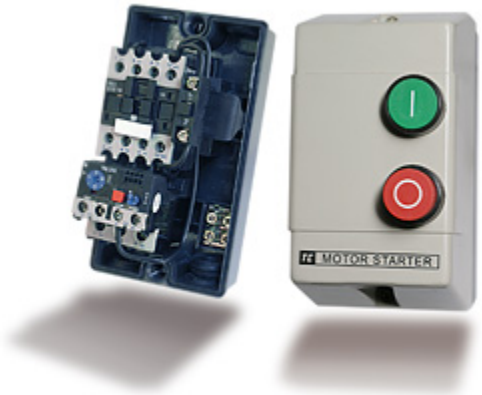
Relä

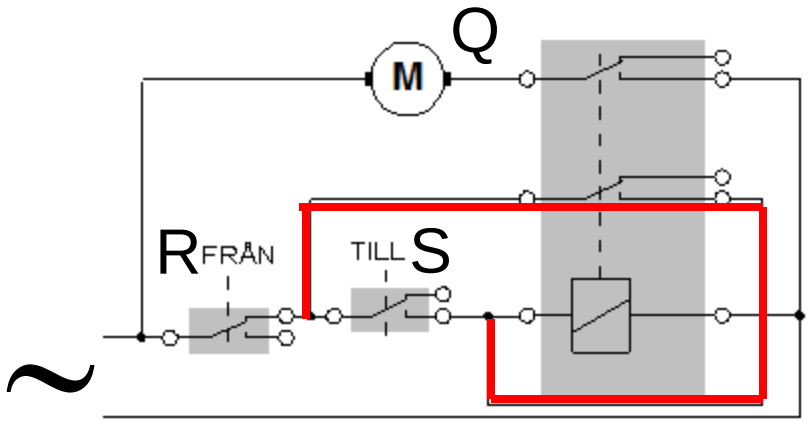
Ett **motorskydd** är ett relä med en **självhållning**s kontakt.

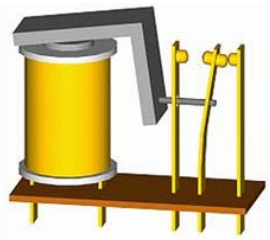
- Man behöver bara trycka kort för att motorn ska starta.
- Blir det strömavbrott så startar *inte* motorn plötsligt av sig själv när strömmen kommer tillbaks – en bra säkerhetsdetalj.
- Lamporna i lokalen däremot tänds direkt – också det bra.

William Sandqvist william@kth.se

(Motorskydd)







Relä

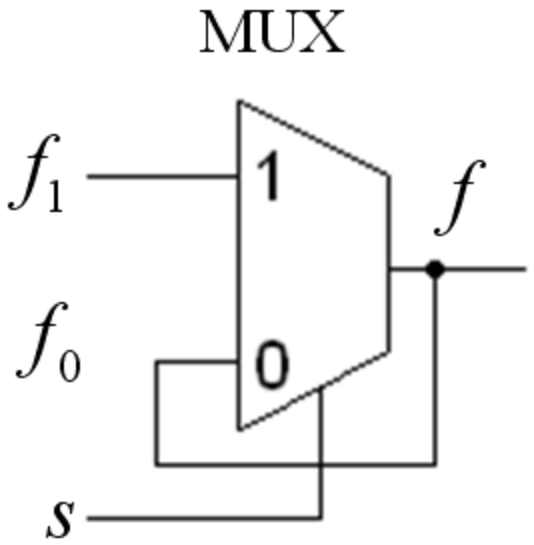
Ett **motorskydd** är ett relä med en **självhållnings**kontakt.

- Man behöver bara trycka kort för att motorn ska starta.
- Blir det strömavbrott så startar *inte* motorn plötsligt av sig själv när strömmen kommer tillbaks – en bra säkerhetsdetalj.
- Lamporna i lokalen däremot tänds direkt – också det bra.

William Sandqvist william@kth.se

”Självhållning”

s	f_1	f_0	f
0	–	f_0	f_0
1	f_1	–	f_1



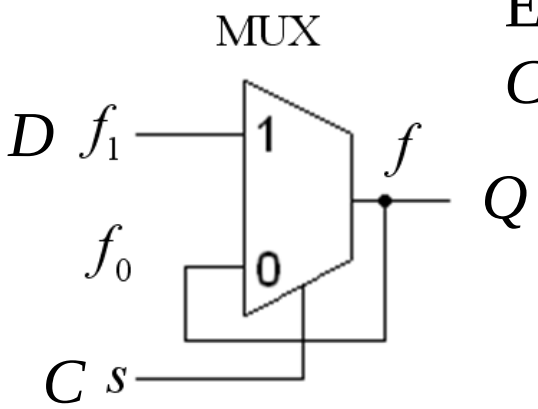
Om $s = 1$ följer utgången f ingången f_1. När s blir $s = 0$ så ”låser sig” kretsen till det värde f hade i ögonblicket ***före*** övergången till $s = 0$.

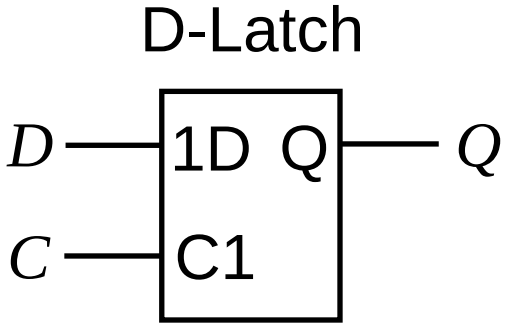
$$s = follow / \overline{latch}$$

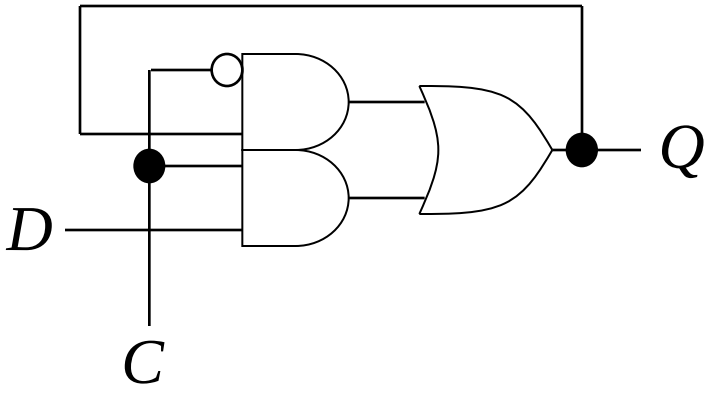
William Sandqvist william@kth.se

D-Latch

En D-latch är en ”återkopplad” MUX. När $C = 0$ låses värdet. (Latch = låsklyka).



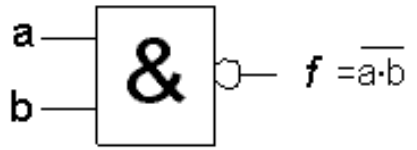
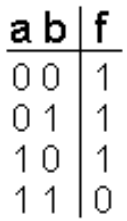
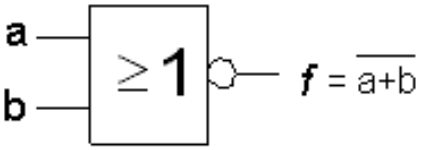




C follow / $\overline{latch}$	D	Q
0	–	M latch
1	D	D follow

William Sandqvist william@kth.se

NOR och NAND ”låsande insignal”

Benämning	Grindfunktion
NAND	 
NOR	 

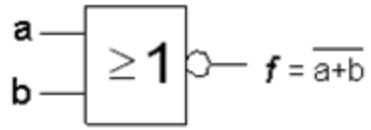
Regel …

NAND. Om **någon** ingång är **”0”**, så är utgången ”1” oavsett värdet på den andra ingången!

NOR. Om **någon** ingång är **”1”**, så är utgången ”0” oavsett värdet på den andra ingången!

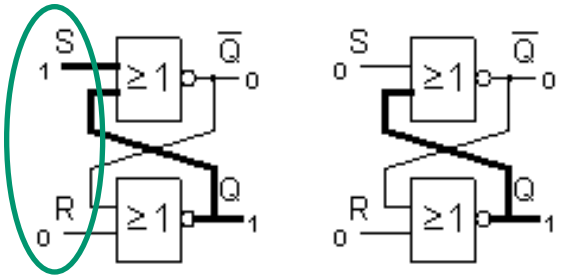
William Sandqvist william@kth.se

SR-latchen med NOR-grindar

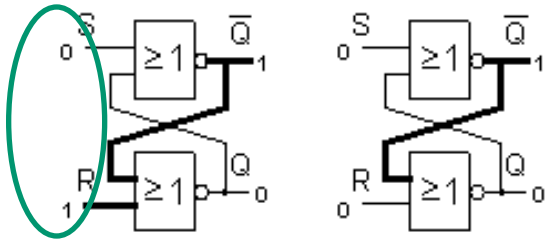


a	b	f
0	0	1
0	1	0
1	0	0
1	1	0

Q=1



Q=0

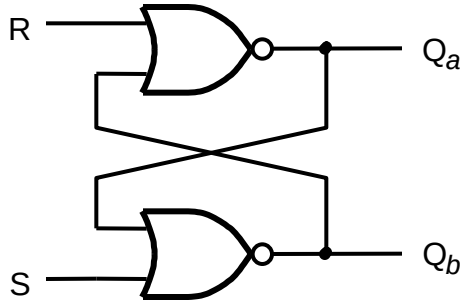


För en NOR-grind är ”1” en ”låsande” insignal – om någon ingång är ”1” har det ingen betydelse vad någon annan ingång har för värde – utgången blir då alltid ”0”.

Det räcker därför med en **kort puls ”1”** på S för att kretsen ska hålla Q = 1. En kort puls ”1” på R ger Q = 0.

William Sandqvist william@kth.se

SR-latch



(a) Circuit

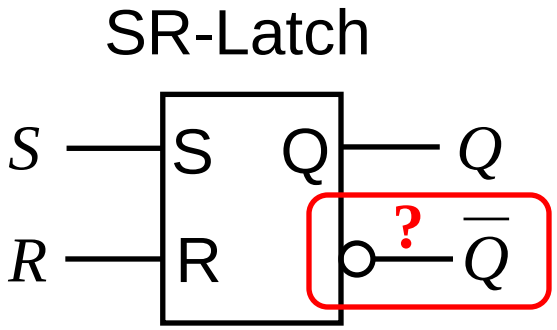
S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

(b) Truth table

Förbjuden insignal S=R=1

$Q_a \neq \overline{Q}_b$

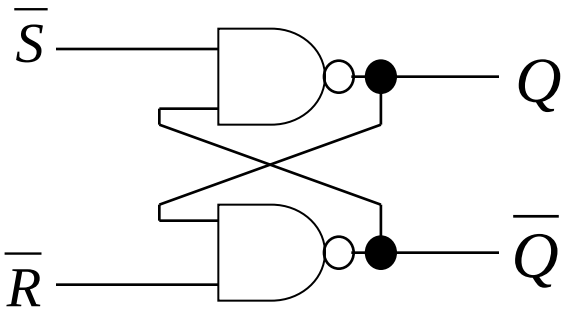
Så länge man **undviker** insignalen $S = R = 1$ (= förbjudet tillstånd) kommer utgångarna Q_a och Q_b att vara varandras inverser. Man kan då använda symbolen till höger.

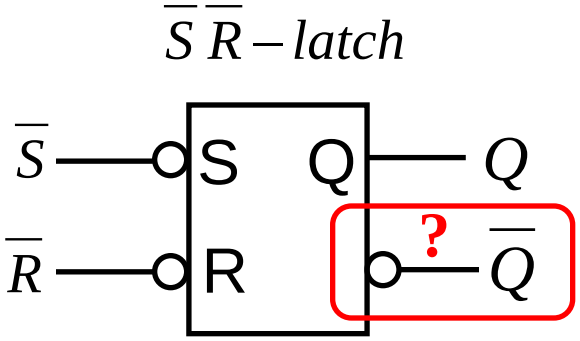


Tar man signaler från låskretsar finns det således alltid inverser att tillgå!

William Sandqvist william@kth.se

$\overline{S}\overline{R}$-latch med NAND-grindar





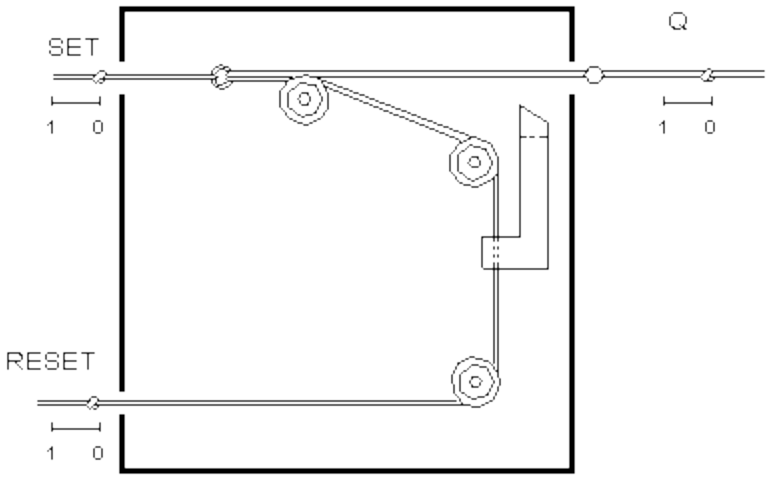
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	M	M

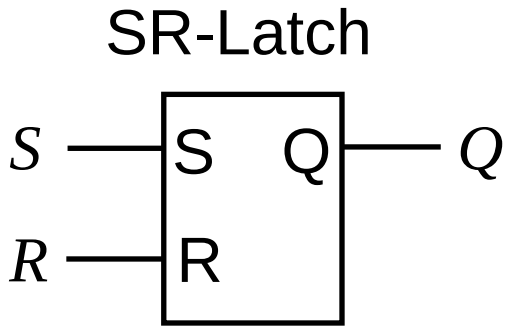
För NAND-grindar är ”0” en låsande insignal som tvingar grindens utgång till ”1”.

Latch med NAND-grindar har **aktiv låga** SET och RESET ingångar. De får ***inte*** vara ”0” samtidigt.

William Sandqvist william@kth.se

SR-Latch

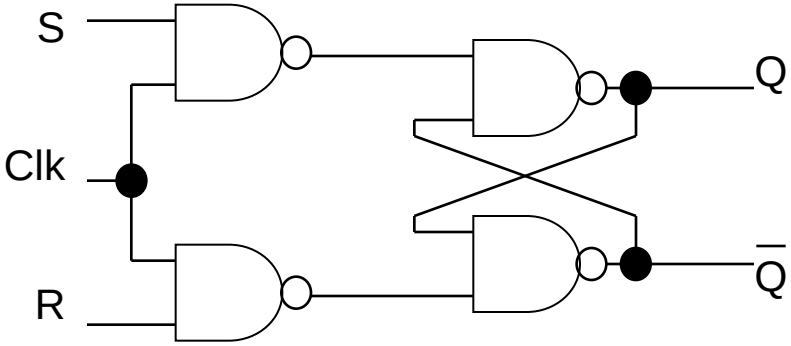




Till vänster har vi en SR-latch i repteknik – ett April-skämt från Scientific American! Inte heller här får man dra i SET och RESET repen samtidigt.

(Gated SR-Latch)

Med två extra grindar och en klocksignal *Clk* kan man styra **när** låskretsen ska få påverkas av insignalerna *S* och *R*. När *Clk* = 0 finns ***ingen påverkan***, då kan ju till och med $S = R = 1$ tillåtas.



Clk	S	R	Q	$\overline{Q}$
1	0	0	M	M
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1
0	-	-	M	M

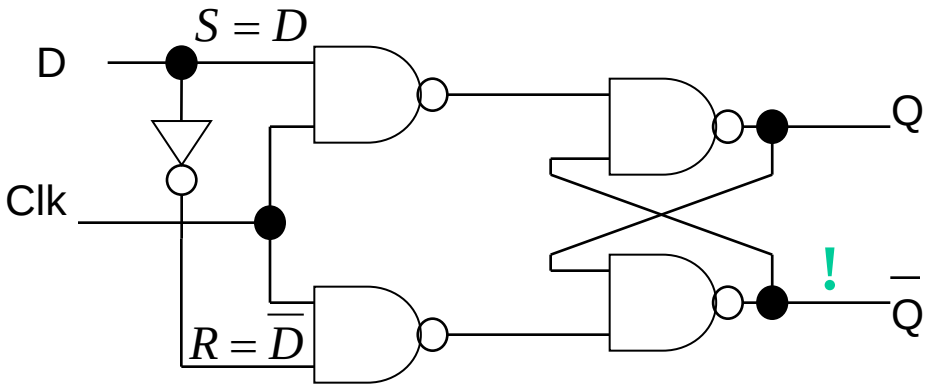
Förbjudet tillstånd

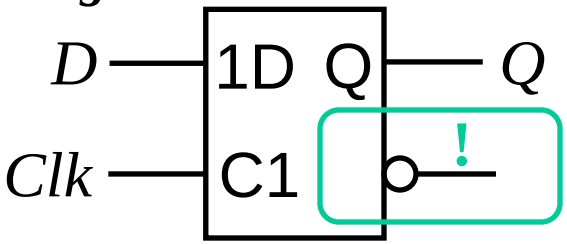
William Sandqvist william@kth.se

D-latch

En ändå bättre lösning på problemet med det ”förbjudna tillståndet” är D-latchen. Med en ***inverterare*** säkerställer man att *S* och *R* helt enkelt ***alltid har olika värden***!

Låskretsens utgång följer D-ingången när *Clk* = 1 för att låsa värdet när *Clk* = 0. Denna låskrets har samma funktion som den återkopplade MUX-kretsen. Skillnaden ligger i att denna krets har ***snabbare*** återkoppling. Dessutom får vi också tillgång till en ***inverterad utsignal***.



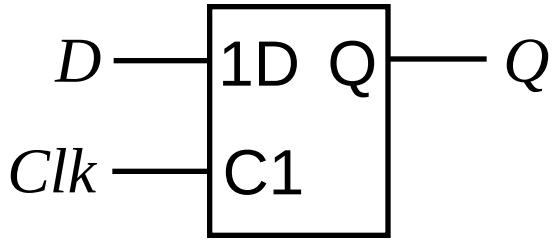


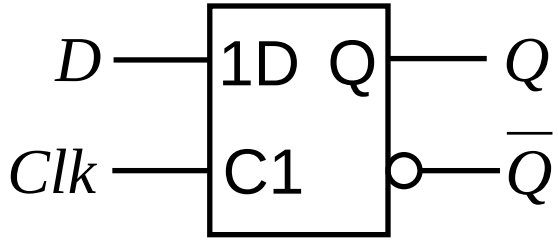
Clk	D	Q	$\overline{Q}$
1	0	0	1
1	1	1	0
0	-	M	M

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

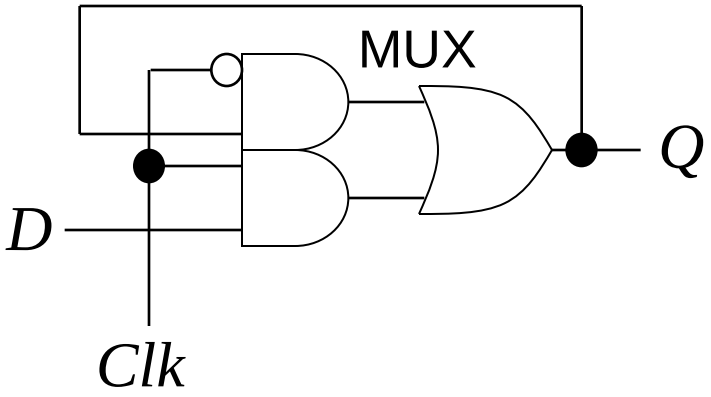
William Sandqvist william@kth.se

Två olika D-latchar

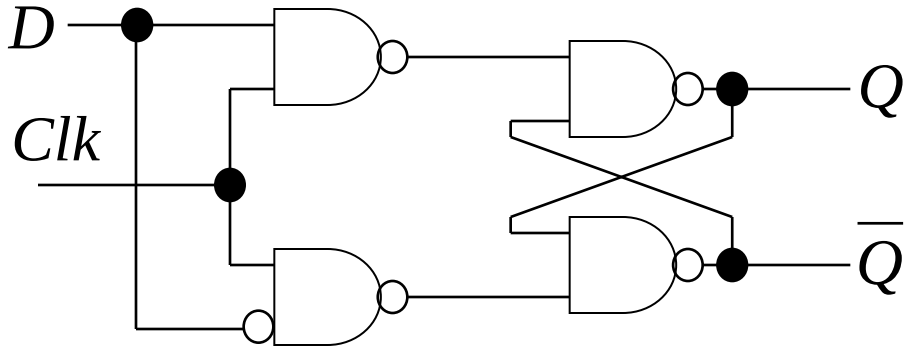




Lång återkoppling (~4T)

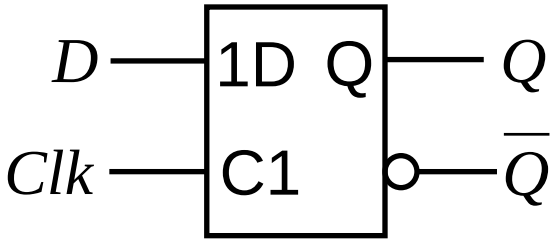


Kort återkoppling (~1T)

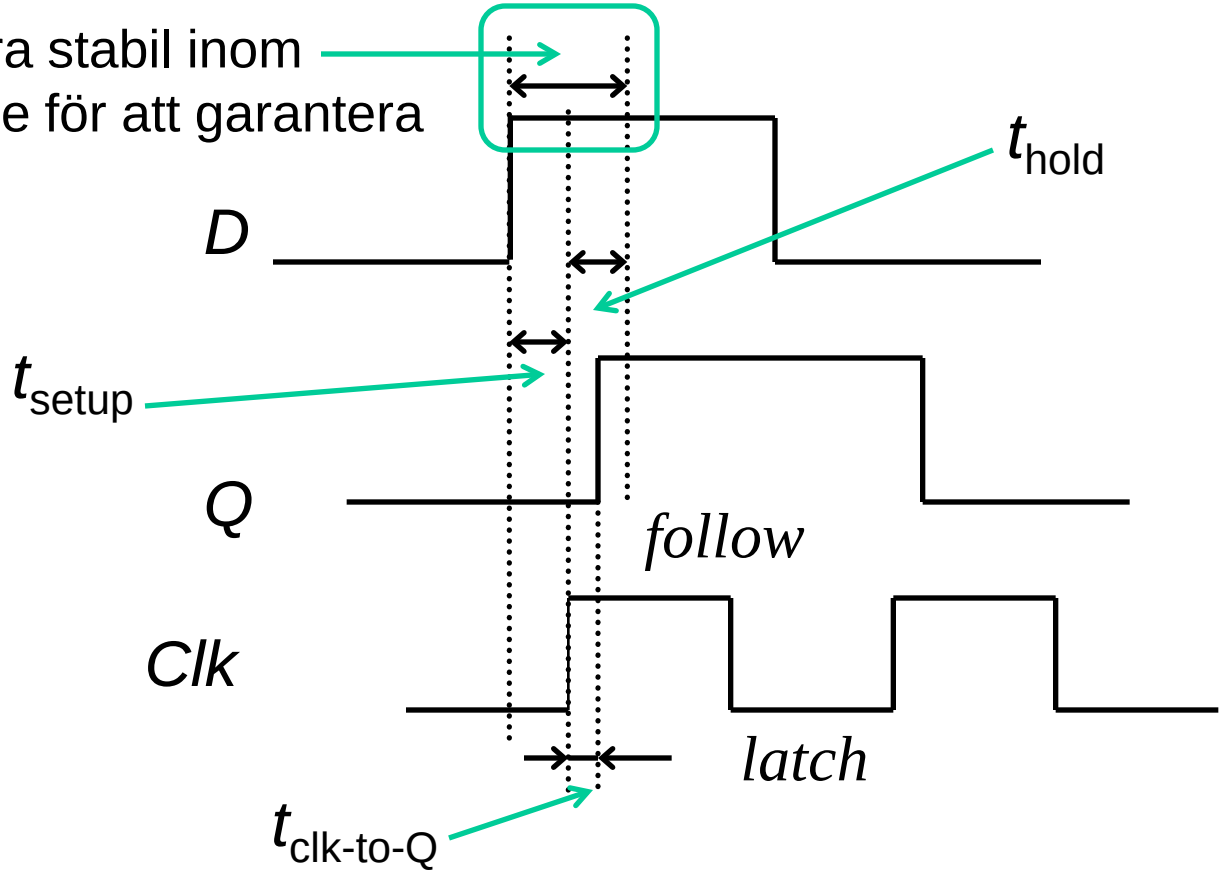


William Sandqvist william@kth.se

Setup- & Hold-time

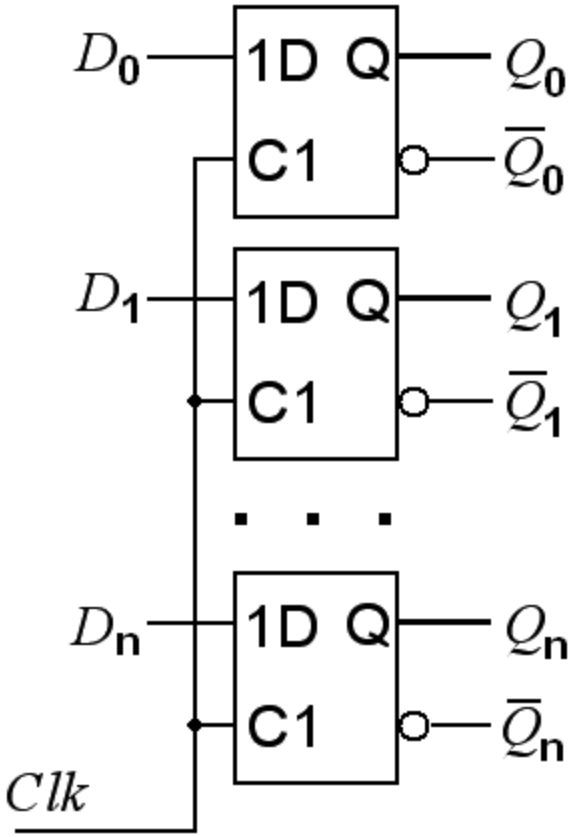


D måste vara stabil inom detta område för att garantera funktionen



William Sandqvist william@kth.se

Register – inverterade utsignaler

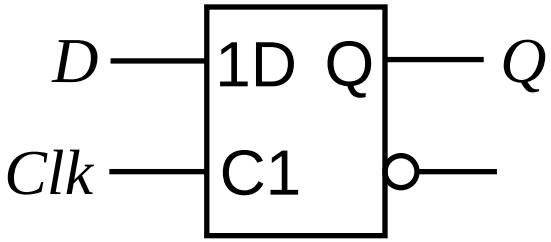


Ett vanligt sätt att konstruera digitala kretsar är att signaler tas via register (= en samling låskretsar eller vippor) till de kombinatoriska nätens ingångar.
D-låskretsar har ”automatiskt” en inversutgång.

Det är därför vi i räkneexemplen oftast utgått ifrån att inverterade signaler finns att tillgå.

William Sandqvist william@kth.se

Varannan gång?



Hur gör man en **sekvenskrets** som ”byter” värde 1/0 efter varje klockpuls, *Clk* ?

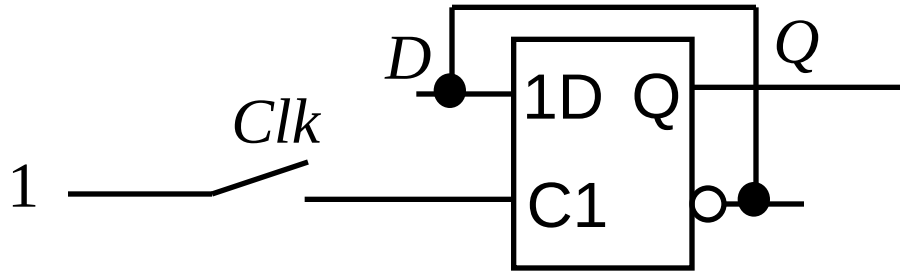
- Kretsen behöver ”minnas” sitt förra värde *Q*
- och ändra detta till $Q = D = \overline{Q}$.

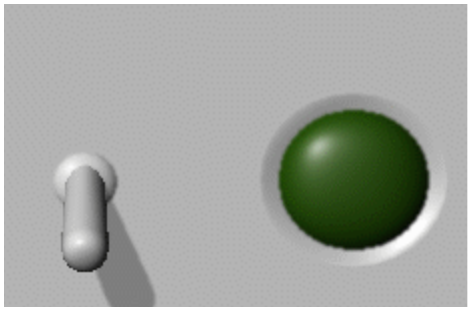
Låskretsen har ju både ”minne” och en inversutgång – skulle inte den kunna användas?

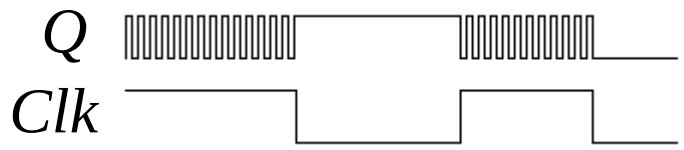
Går ej med enkel låskrets …

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

$$D = \overline{Q} \quad Q = D$$



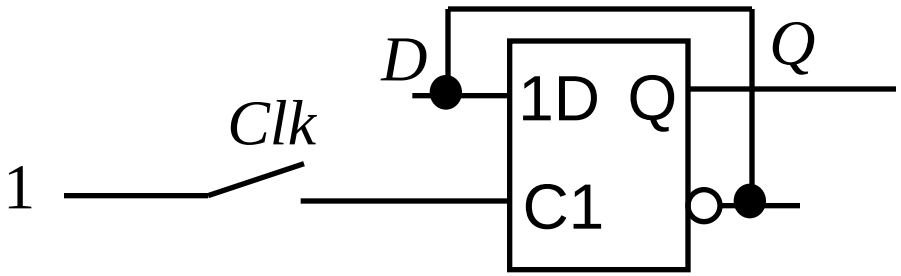


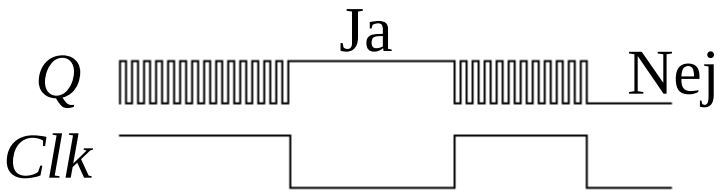


- När *Clk* = 1 följer utsignalen insignalen – därför byter utgången 1/0 så fort som möjligt! Kretsen blir en oscillator!
- När sedan *Clk* = 0 behåller utsignalen sitt värde 1/0 allt efter vad den senast stod på. (= Slumpgenerator?)

William Sandqvist william@kth.se

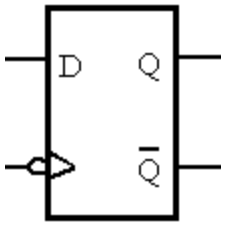
Voteringshjälp i riksdagen?

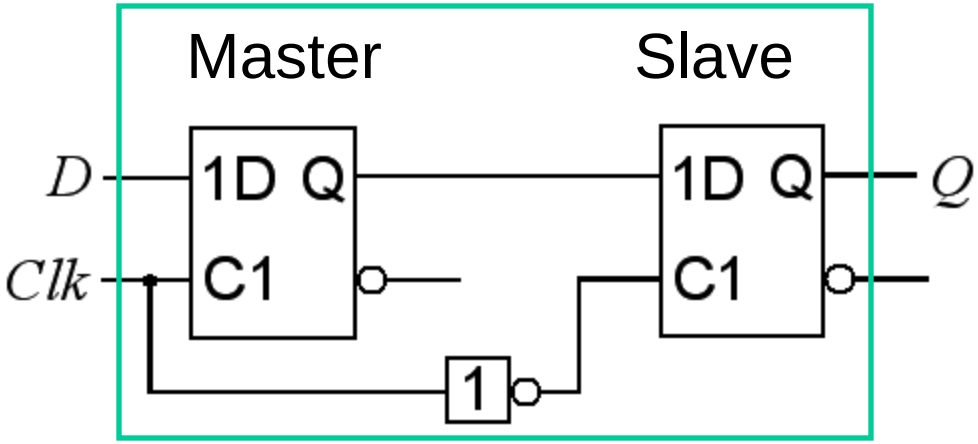




William Sandqvist william@kth.se

Klockade vippor
Master-Slave vippan

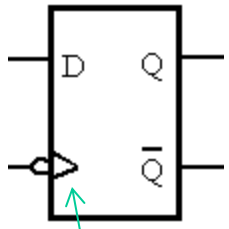




Problemet är att den **enkla låskretsen** är öppen för förändring ända fram tills den ska låsa sitt värde.
Lösningen är den **klockade vippan** som består av flera låskretsar. En låskrets tar emot nya data (Master) medan en annan har kvar det gamla datat (Slave).

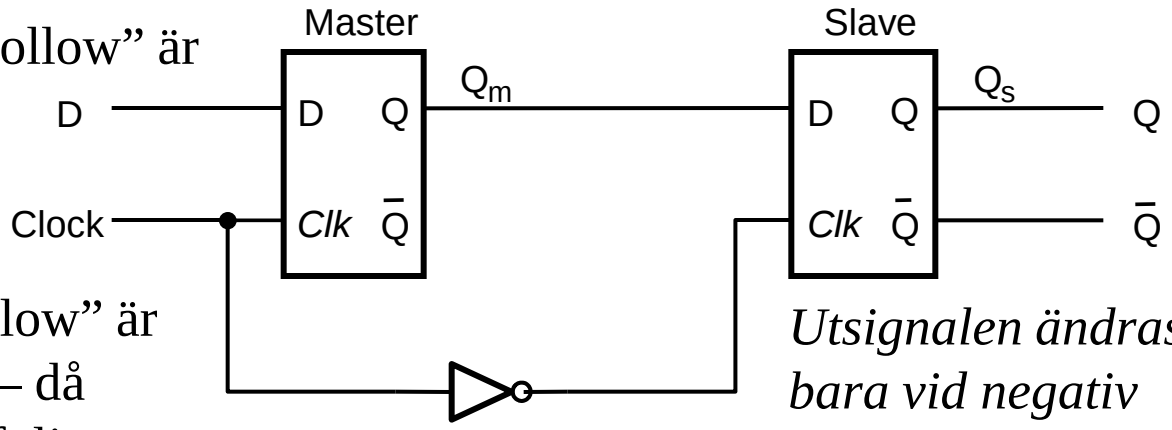
William Sandqvist william@kth.se

Tidsdiagram Master-Slave

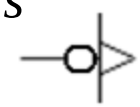


När **Master** gör ”follow” är **Slave** ”latched”.

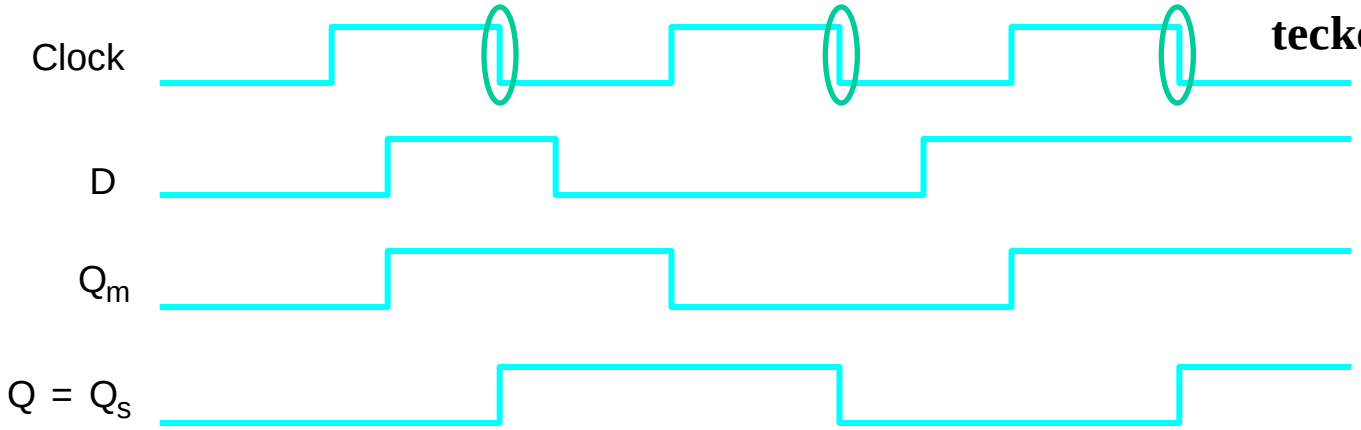
När **Slave** gör ”follow” är **Master** ”latched” – då finns det inget att följa.



Utsignalen ändras bara vid negativ klockflank

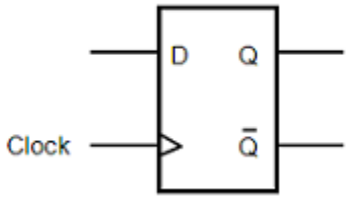


Flanktriggnings-tecken

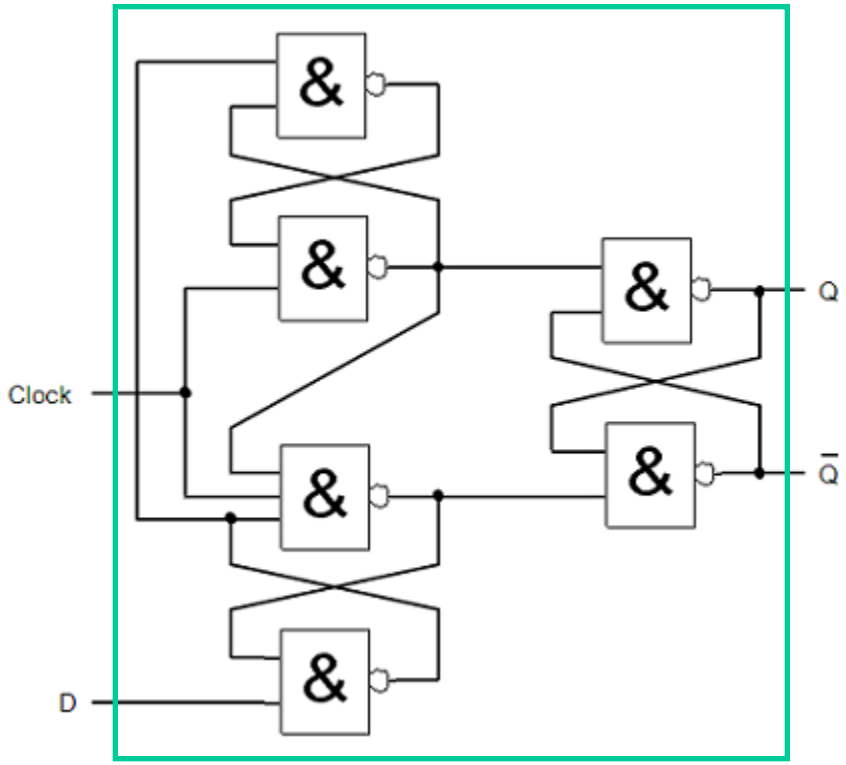


William Sandqvist william@kth.se

Flanktriggad D-vippa

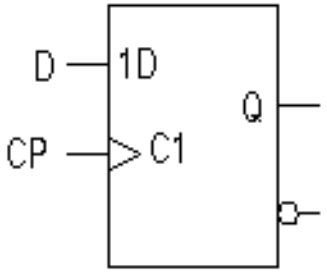


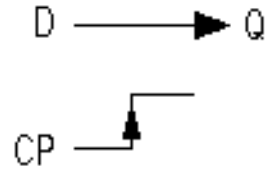
En annan flanktriggad vippa består av **tre** låskretsar. Datavärdet ”kopieras” till utgången **precis** när klocksignalen går från $0 \rightarrow 1$.



Positiv flank $0 \rightarrow 1$

Negativ flank $1 \rightarrow 0$





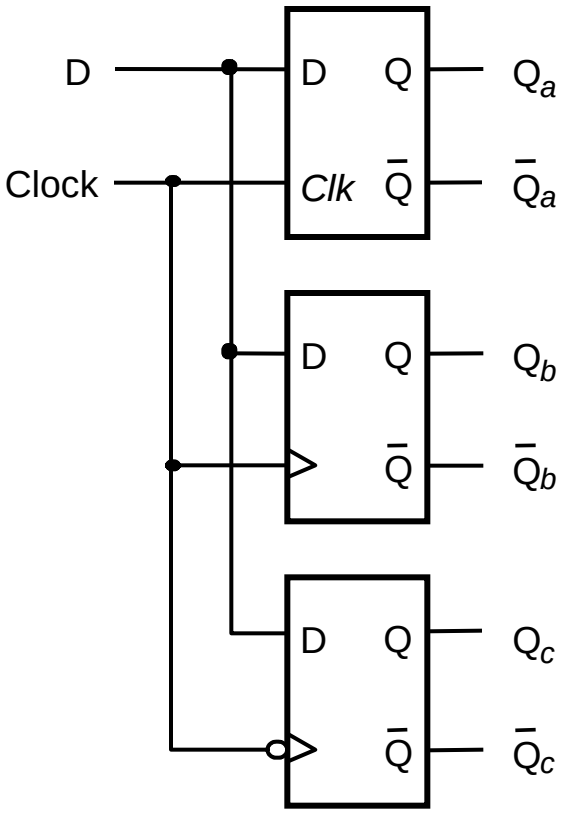
William Sandqvist william@kth.se

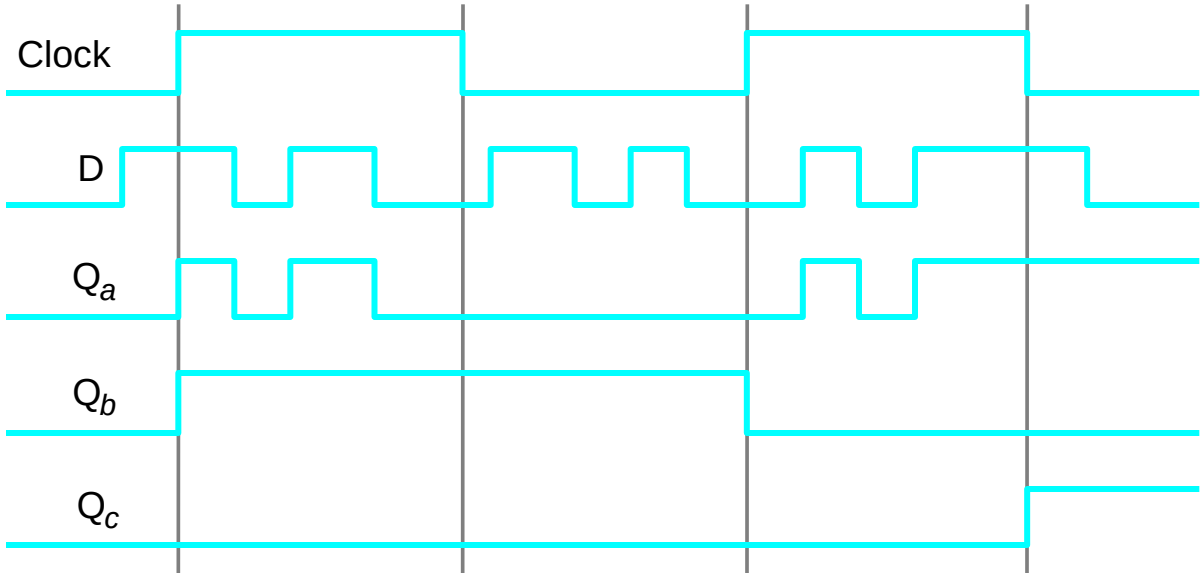
Låskrets eller Vippa?

a) Låskrets – follow/$\overline{\text{latch}}$

b) Positivt flanktriggad vippa

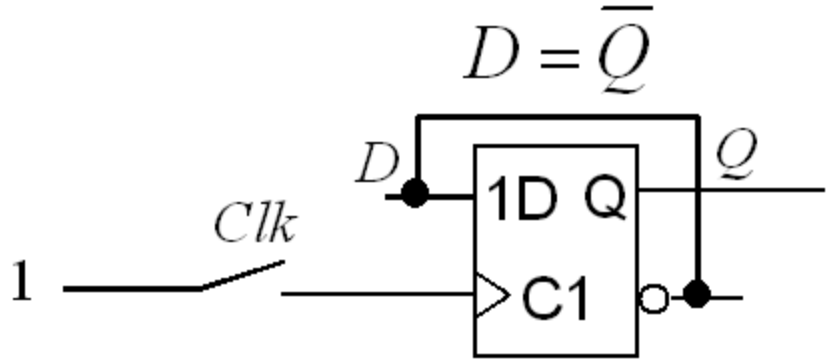
c) Negativt flanktriggad vippa

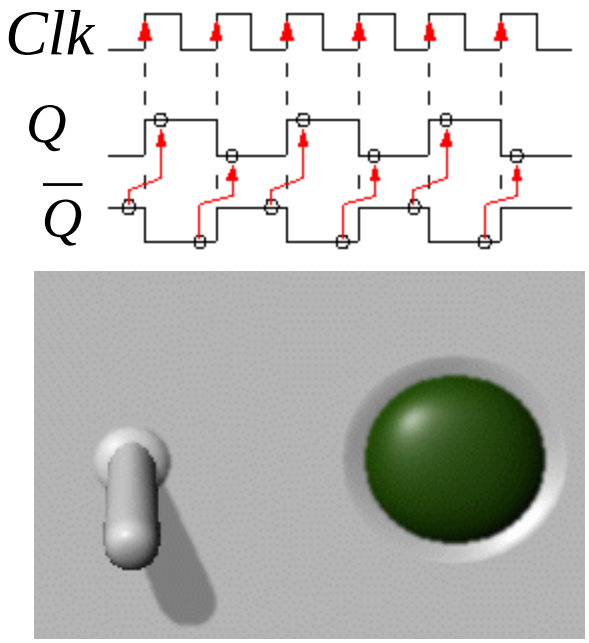




William Sandqvist william@kth.se

Varannan gång?





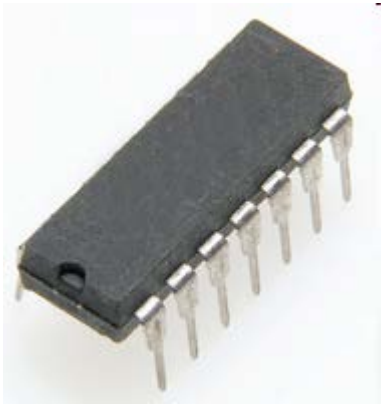
Nu fungerar ”varannan gång” precis som tänkt!

Till sekvensnät använder man i allmänhet flanktriggade vippor som minneselement!

William Sandqvist william@kth.se

Varannan gång med Impulsrelä On-Off-On-Off …

Impulsrelä
Pris: 300:-

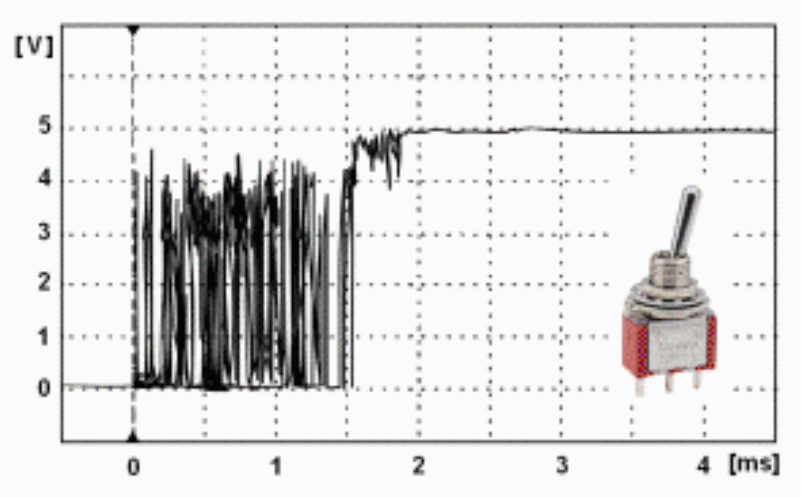


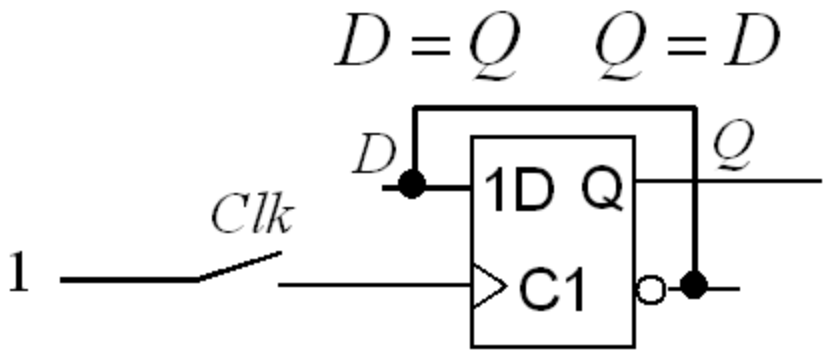
7474 (2st D-vippor)
Pris: 5:- styck

William Sandqvist william@kth.se

(Kontaktstuds)

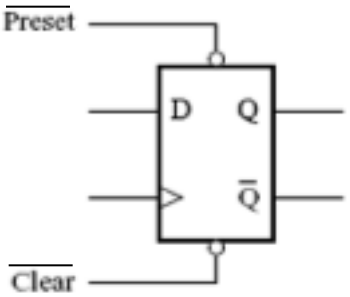
Det kan finnas ett annat hot mot ”varannan gång” kretsen, och det är att mekaniska kontakter studsar! Detta får Du pröva på vid laborationen …

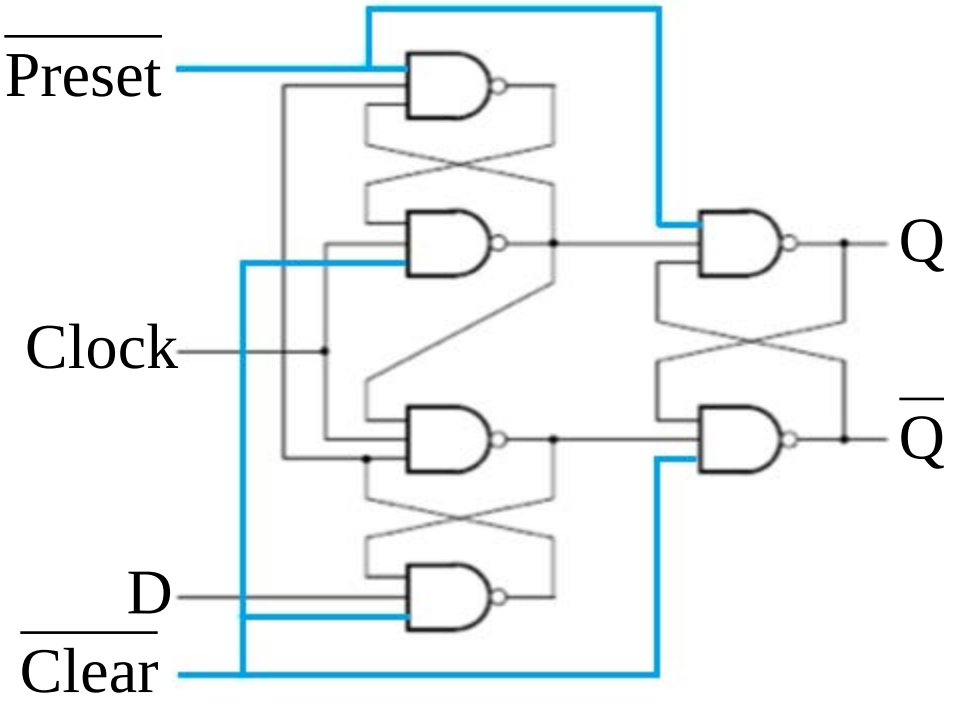




William Sandqvist william@kth.se

Clear och Preset





D-vippan innehåller tre låskretsar. Signalerna **Preset** och **Clear** går direkt till låskretsarna och kan ”låsa” dessa oberoende av klockpulsen. Preset och Clear är aktivt låga.

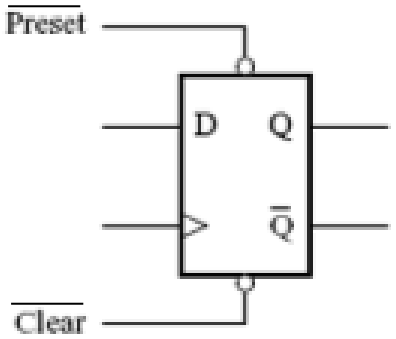
Preset = 0 tvingar Q = 1, medan Clear = 0 tvingar Q = 0.
Preset = Clear = 1 tillåter vippan att fungera som avsett.

William Sandqvist william@kth.se

Reset-knappen

De flesta digitala system behöver kunna startas i ett känt tillstånd. Det kan innebära att en del vippor ska vara ”1” medan andra ska vara ”0”. En resetfunktion kan därför behöva anslutas till antingen **Preset** eller **Clear** på de ingående vipporna.

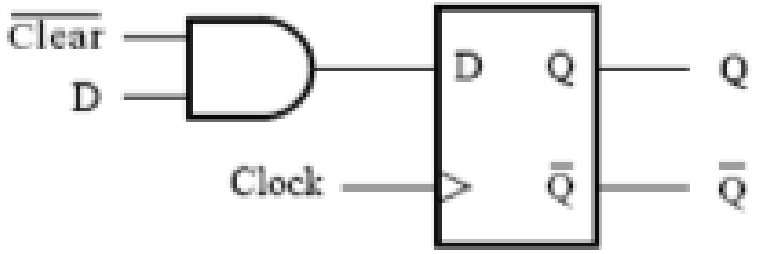
Preset och **Clear** är asynkrona ingångar – vipporna ändrar sig omedelbart oavsett klockpuls.



William Sandqvist william@kth.se

Synkron Reset

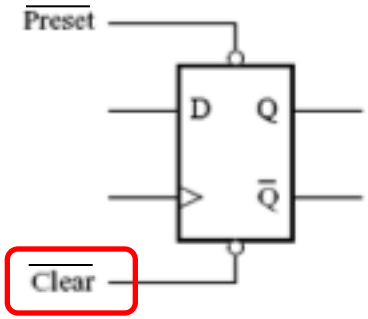
Saknar vipporna Preset- och Clear- ingångar, kan reset implementeras med extra logik. Synkron reset orsakar att vippan går till läge 0 vid *nästa* klockflank.

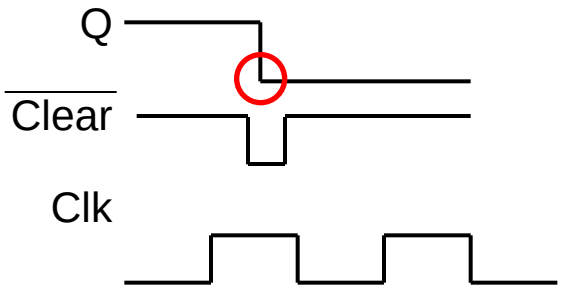


William Sandqvist william@kth.se

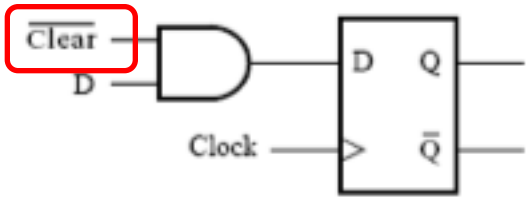
Asynkron/Synkron Reset

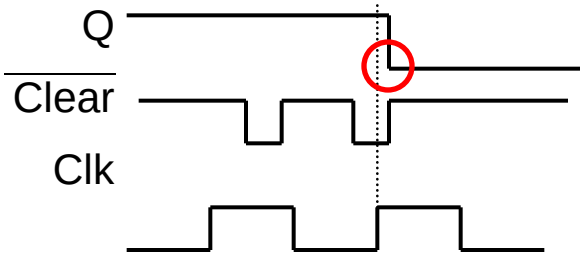
Asynkron reset





Synkron reset



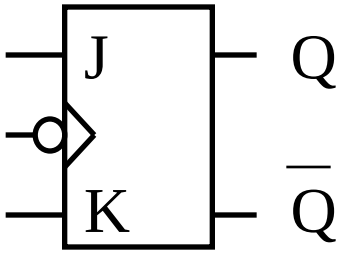


William Sandqvist william@kth.se

Andra vanliga typer av vippor

JK-vippa

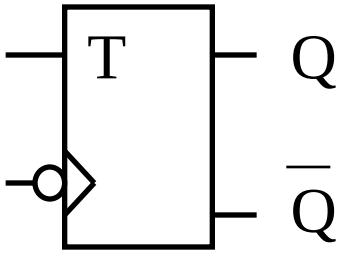
(JK-vippan är en SR-vippa med ”toggle” i stället för förbjudet tillstånd)



Clk	J	K	Q	$\overline{Q}$
↓	0	0	M	M
↓	0	1	0	1
↓	1	0	1	0
↓	1	1	Toggle	Toggle

T-vippa (T=Toggle)

(T-vippan är speciellt lämplig för ”räknare”)

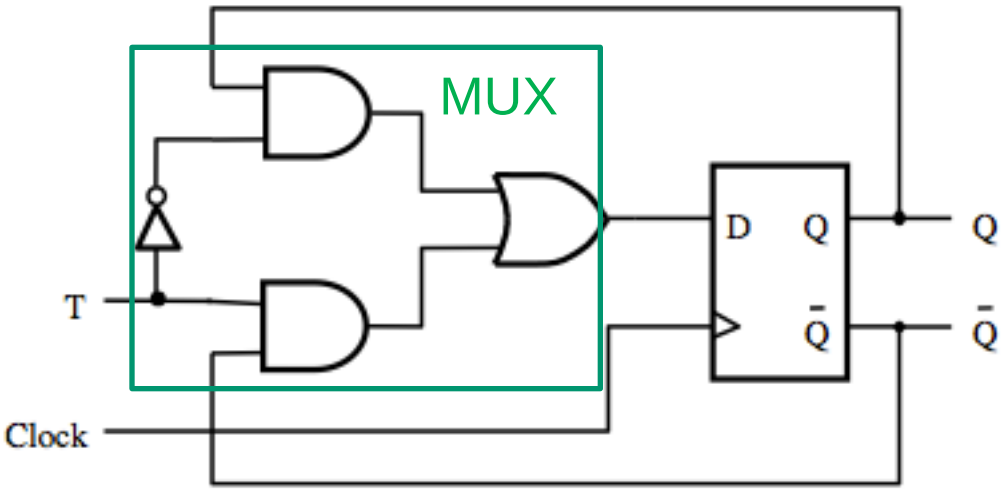


Clk	T	Q	$\overline{Q}$
↓	0	M	M
↓	1	Toggle	Toggle

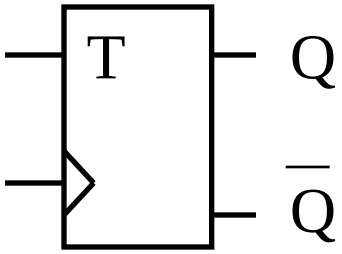
William Sandqvist william@kth.se

Konstruera T-vippa med D-vippa

$D = Q$ *hold*



$D = \overline{Q}$ *toggle*



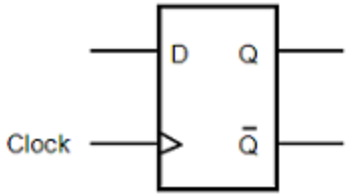
William Sandqvist william@kth.se

Timing analysis

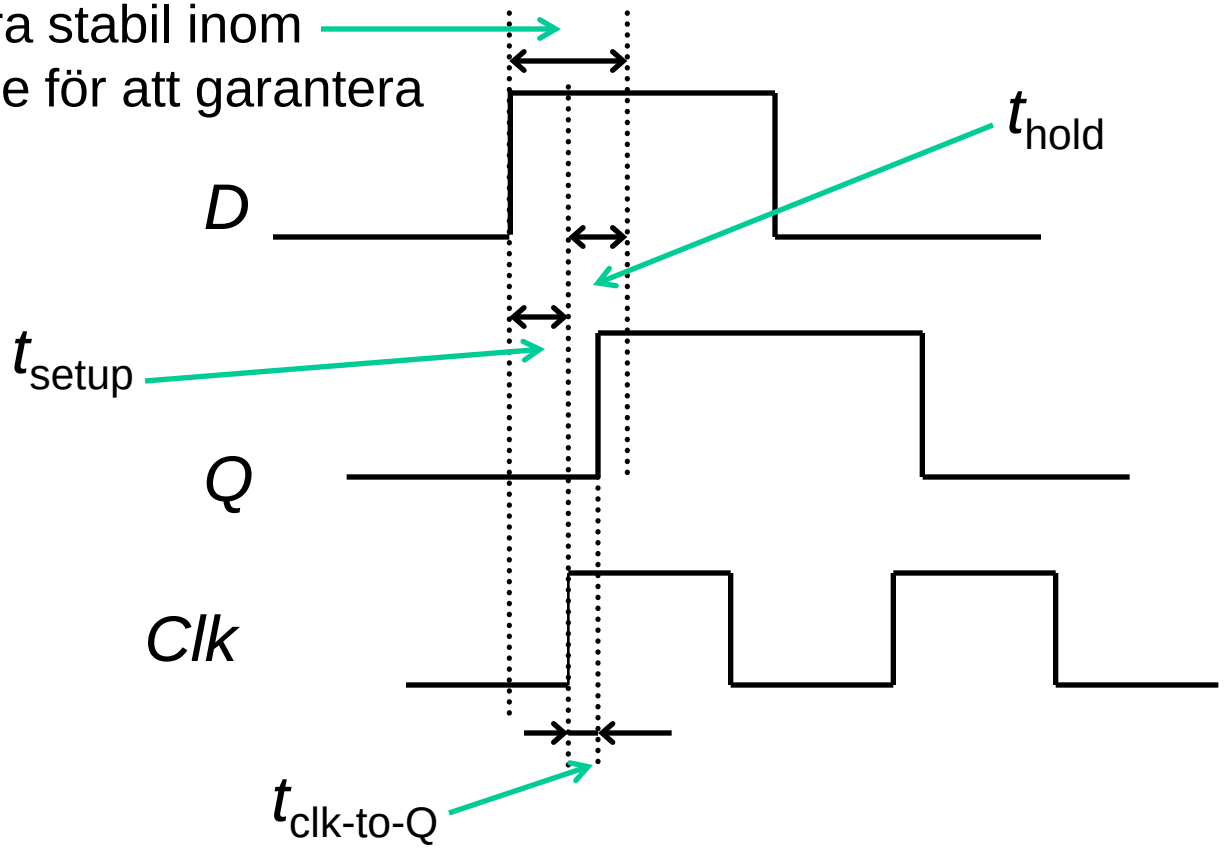
Det är möjligt att kunna bestämma den maximala frekvensen i en sekvensiell krets genom att ha information om

- Grindfördröjningar t_{logic}
- Setup-tid t_{su} för vippan
- Hold-tid t_{h} för vippan
- Clock-to-utgång t_{cQ} tiden

Setup- & Hold-time



D måste vara stabil inom detta område för att garantera funktionen



William Sandqvist william@kth.se

Vad är den maximala frekvensen?

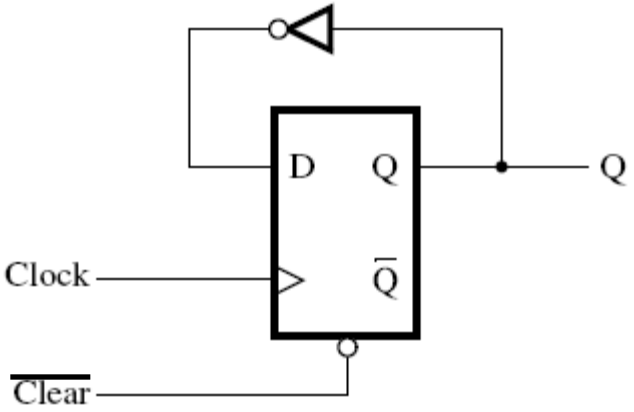
- Grindfördröjningar

  $t_{logic} = t_{NOT} = 1.1$ ns
- Setup-tid

  $t_{su} = 0.6$ ns
- Hold-tid

  $t_h = 0.4$ ns
- Clock-to-output

  $t_{cQ} = 1.0$ ns



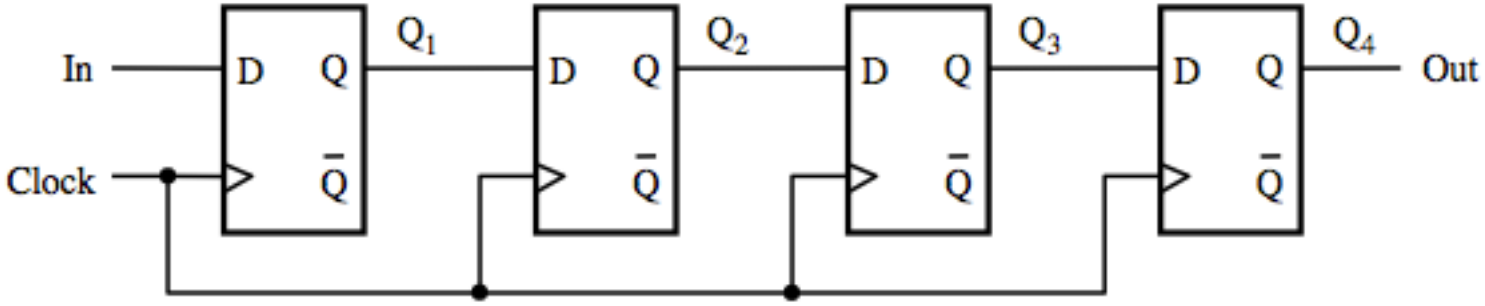
0.6 0.4 < 1.0 1.1

$T = t_{su} + \max(t_h, t_{cQ}) + t_{logic} = 2.7$ ns

$f = 1/T = 370$ MHz

William Sandqvist william@kth.se

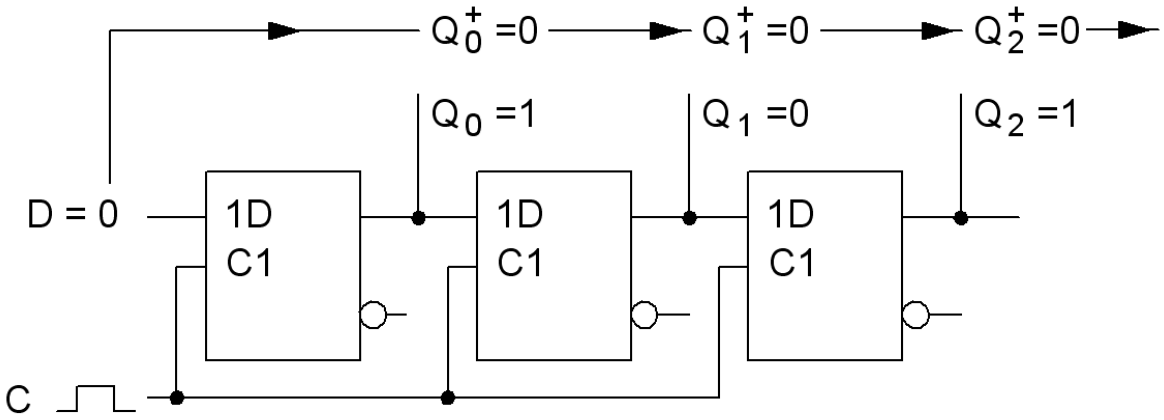
Shiftregister



- En **shiftregister** innehåller flera vippor
  För varje klockcykel skiftar man in ett värde från vänster till höger
- Många konstruktioner använder **shiftregister** och värden Q_4, …, Q_1 som ingångsvärden till andra komponenter

Går inte med låskretsar …

Det går inte att bygga ett **shiftregister** med låskretsar.



När C = 1 *follow* så ”rinner” datat igenom alla låskretsarna …

Vanliga typer av shiftregister

- Parallel-In/Parallel-Out (**PIPO**)
- Parallel-In/Serial-Out (**PISO**)
- Serial-In/Parallel-Out (**SIPO**)
- Serial-In/Serial-Out (**SISO**)

- Användningsområden
  - Köer, tex First-In/First-Out (**FIFO**)
  - Mönsterigenkänning (eng. Pattern recognizers)

William Sandqvist william@kth.se

Räknare

En räknare är en speciell typ av sekvensnät som registrerar antalet inkommande klockpulser. Registreringen sker efter någon kod, oftast **binärkod**. Efter ett visst antal pulser tar räknarens tillstånd slut och den börjar om från början igen.

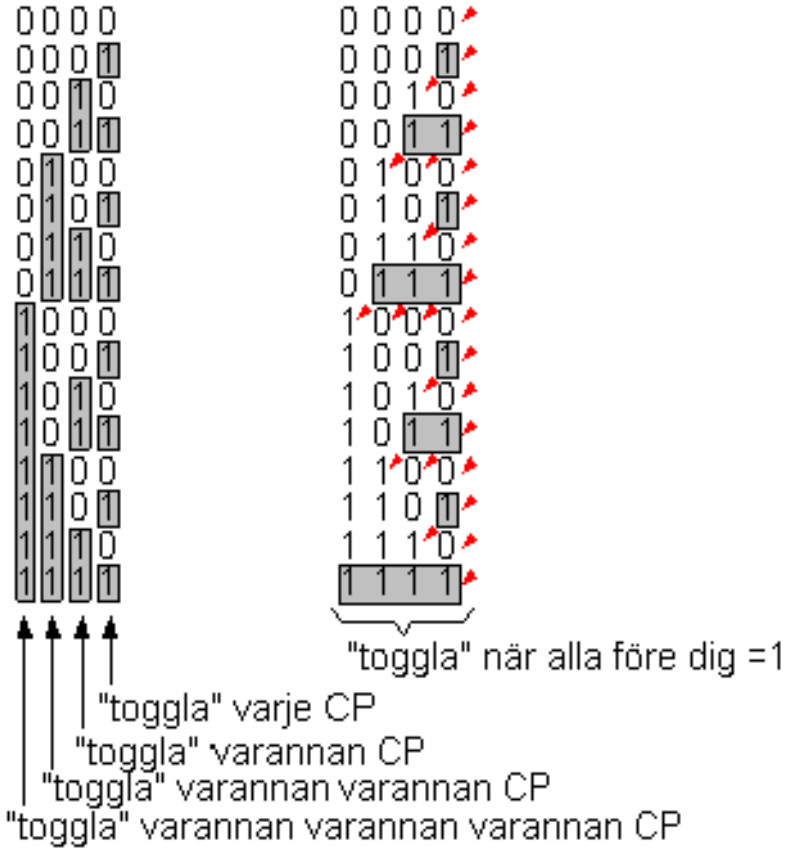
Man talar om räknarens **modul** (dvs. hur många tillstånd räknecykeln innehåller).

Räknaren behöver inte ha någon insignal utom klockpulserna (som då kan ses som insignalen).
Ett sådant sekvensnät kallas för **autonomt**.

William Sandqvist william@kth.se

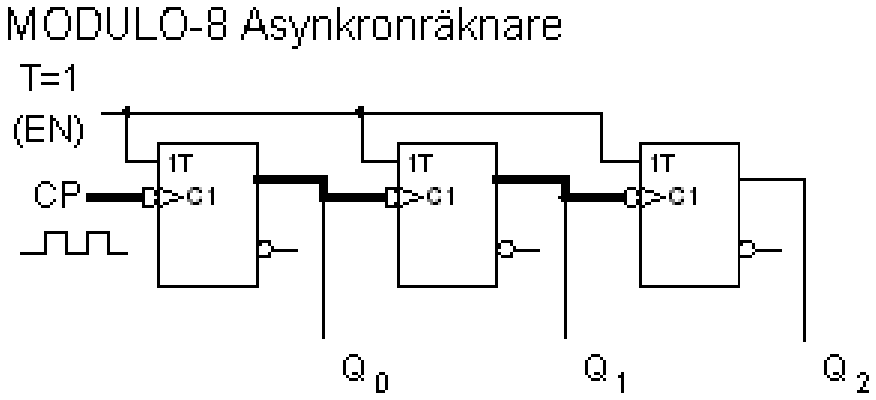
Binärkodens räkne-egenskaper

Det finns två olika "regler" för att konstruera binärkoden ur mindre signifikanta bitar. Ex. med binärkoden 0 ... 15.



William Sandqvist william@kth.se

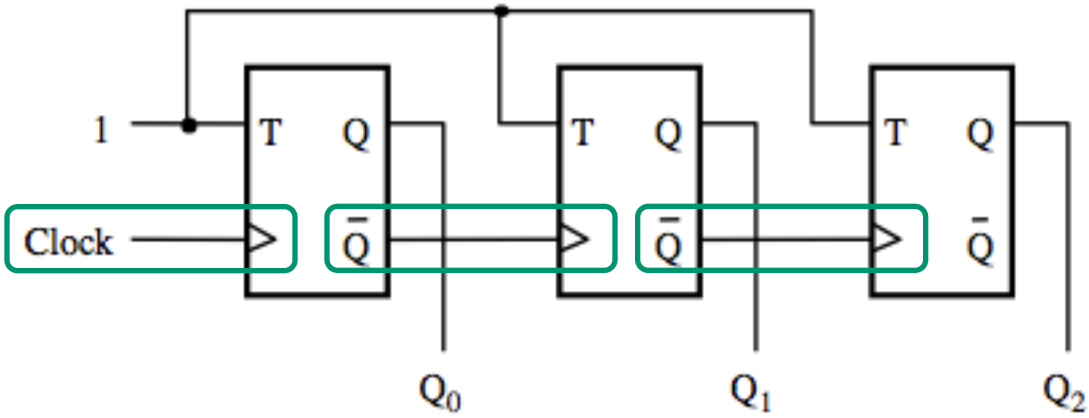
Toggla varannan gång …



varannan, varannan varannan, varannan varannan varannan …

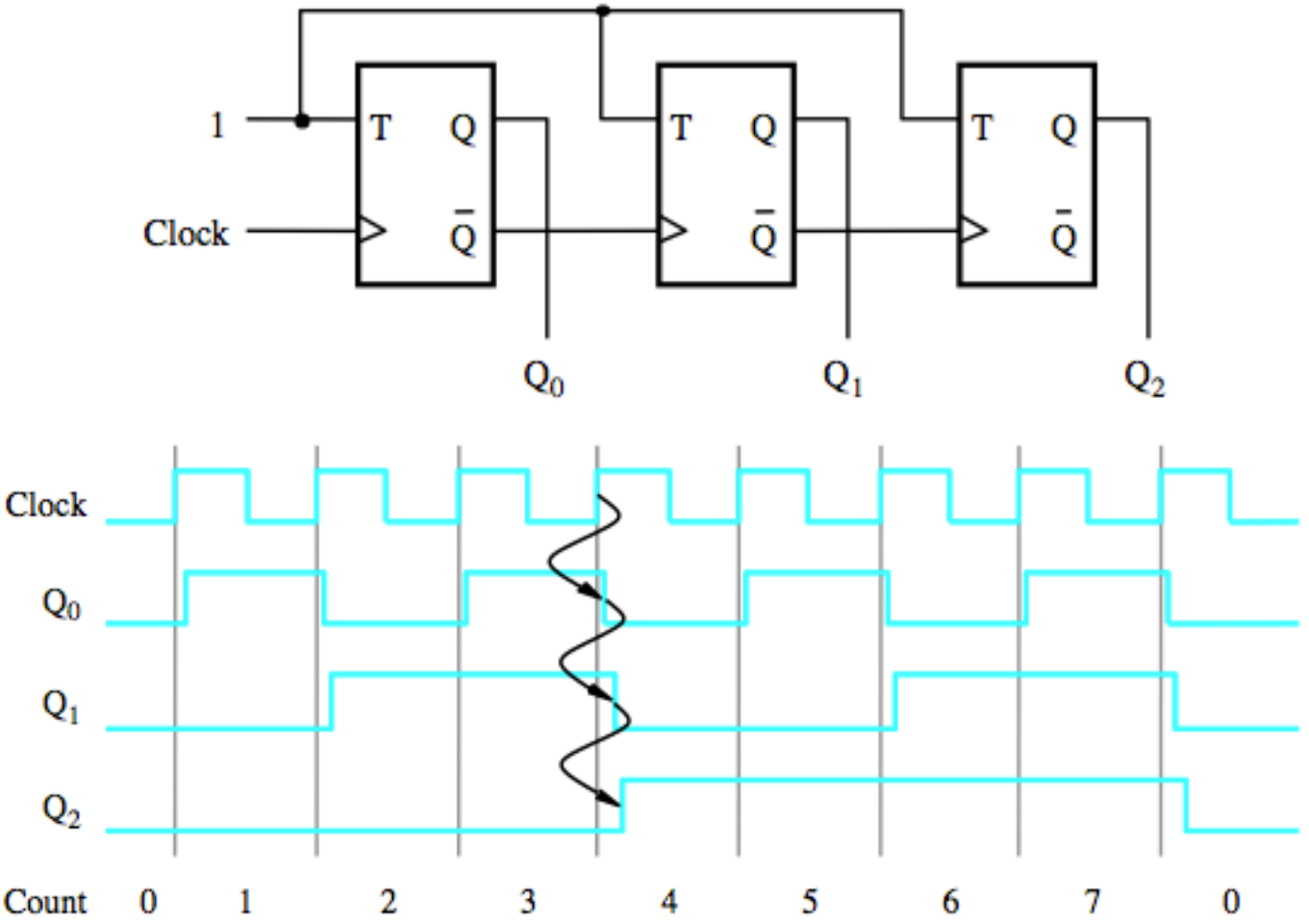
Räknaren är uppbygd av T-vippor, de har alla T=1 och "togglar" därför vid varje klockpuls. Den första vippan Q_0 "togglar" för varje klockpuls. Vippan därefter Q_1 klockas av den första vippan. Den kommer därför bara att "toggla" för varannan klockpuls. Den tredje vippan Q_2 kommer "toggla" för varannan varannan klockpuls.
Enligt binärtabellen kommer räknaren därför att räkna i binärkod.
($Q_2Q_1Q_0$: 000 001 010 011 100 101 110 111 000 ...).

Hur räknar den här räknaren?



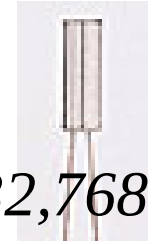
William Sandqvist william@kth.se

Asynkron räknare

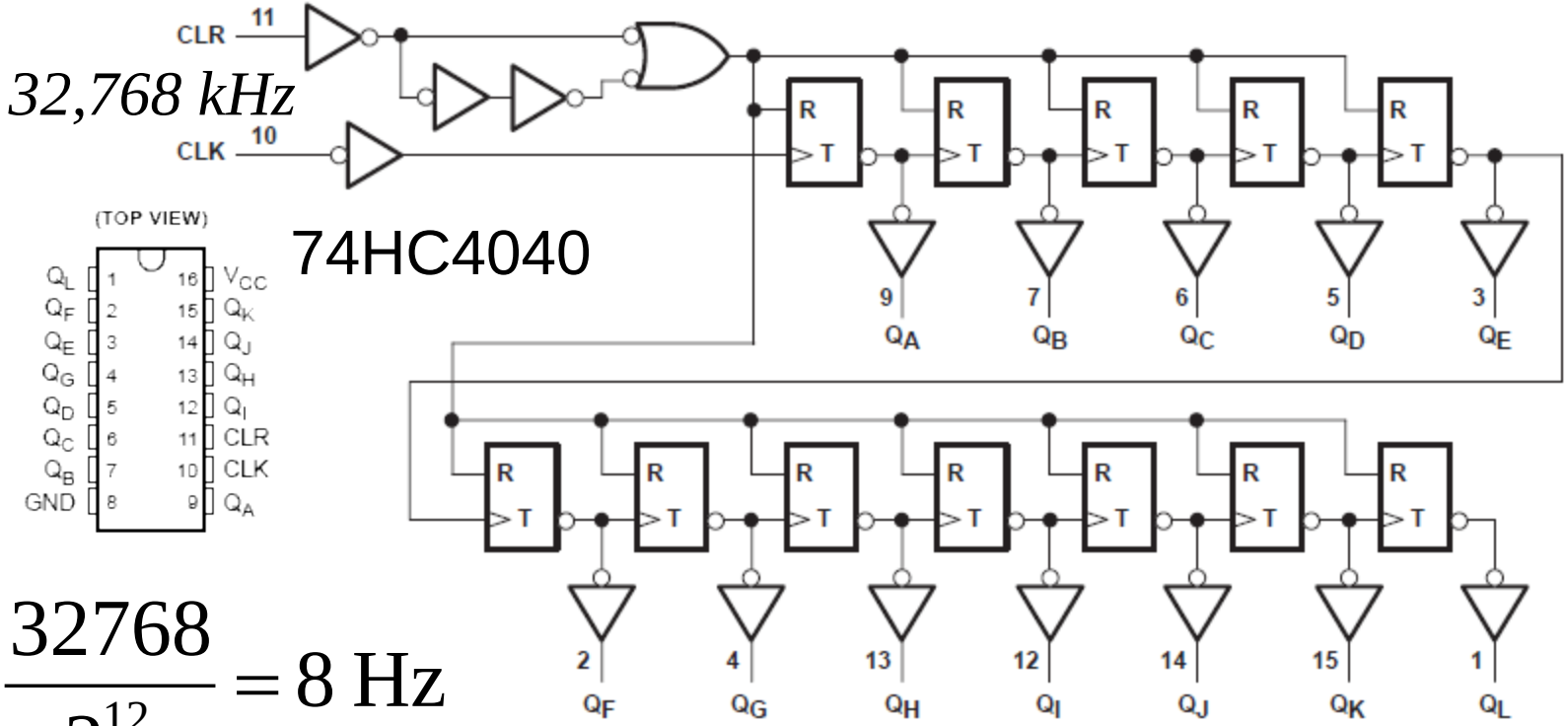


William Sandqvist william@kth.se

En räknarkrets



32,768 kHz



32,768 kHz

74HC4040

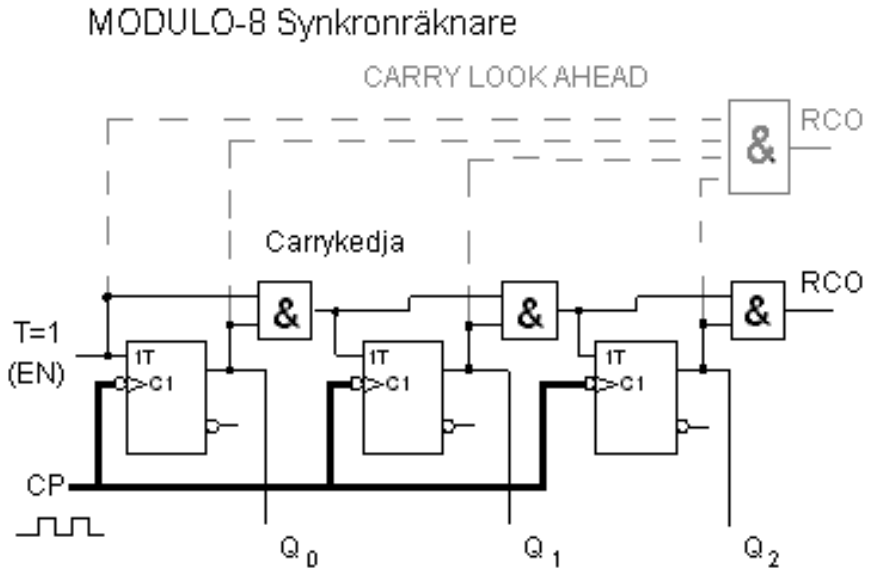
$$\frac{32768}{2^{12}} = 8\ \text{Hz}$$

8 Hz

Hur man får 1 sekund får Du räkna ut själv …

William Sandqvist william@kth.se

Toggla om alla före dig är 1…



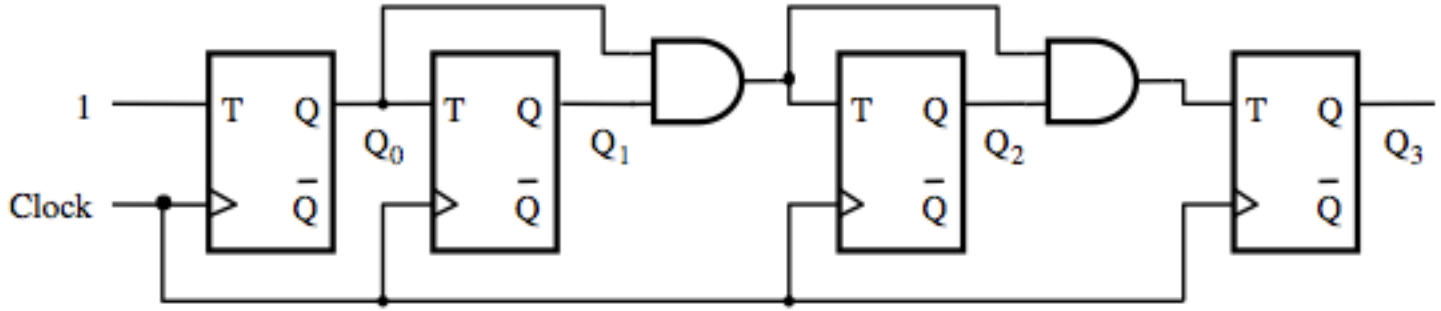
En snabbare räknare kan man få om man tar fram Carry parallellt (jfr. med adderaren).

Vill man utöka räknaren sker det med en vippa och en AND-grind per steg.

Klockpulserna går direkt till alla vippor och därför slår de om samtidigt. Vilka vippor som ska slå om eller ej styrs med T-ingångarna. Den första vippan har T=1 och den slår om för varje klockpuls. En viss vippa ska slå om när alla vippor som är före den står på "1". Det villkoret får man från AND-grindarna i den sk. Carrykedjan och det är dessa som styr T-ingångarna.

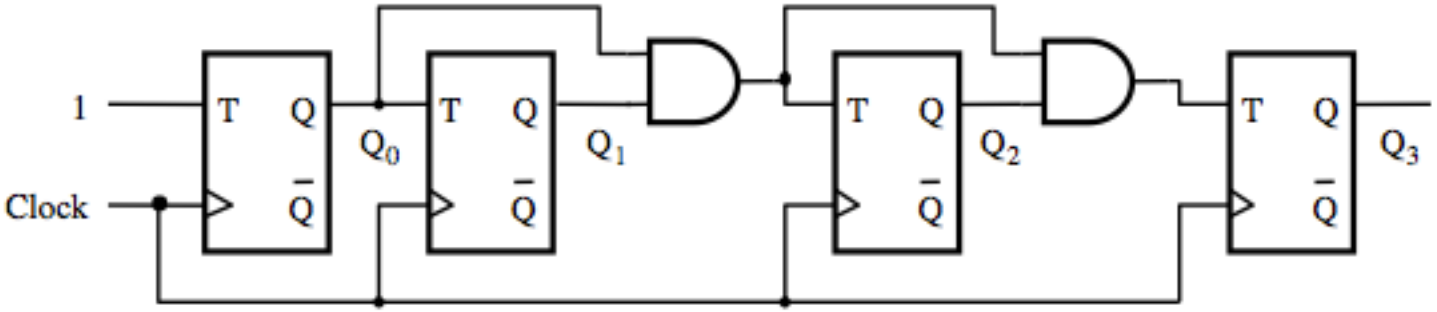
Synkron räknare

I en ***synkron*** räknare är vippornas klockingångar kopplade till **samma klocksignal**

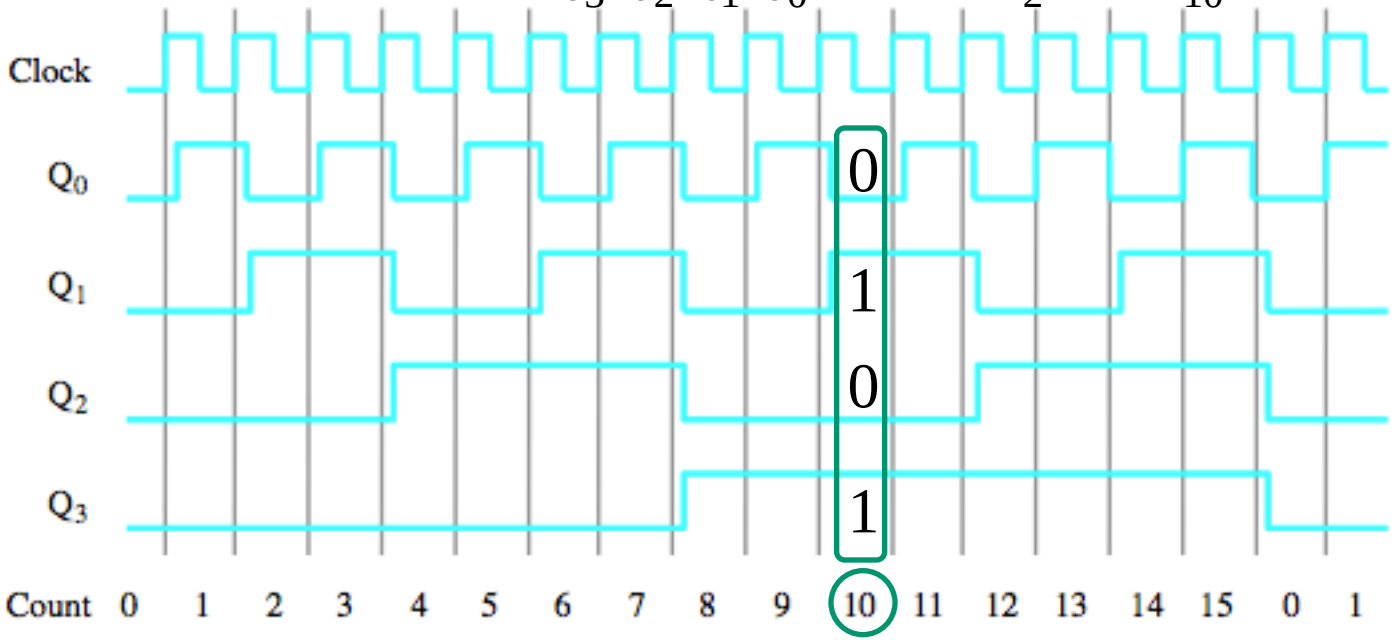


Hur räknar den här räknaren?

Synkron räknare

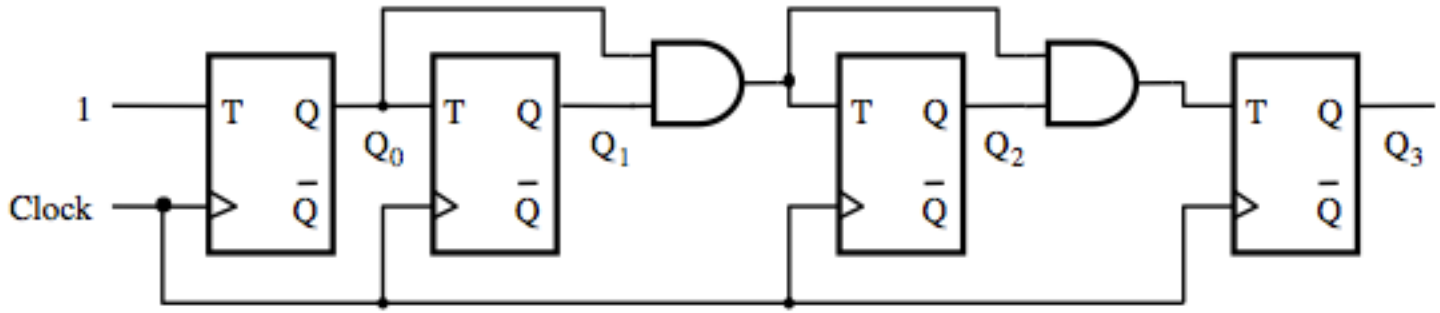


$$Q_3Q_2Q_1Q_0 = 1010_2 = 10_{10}$$



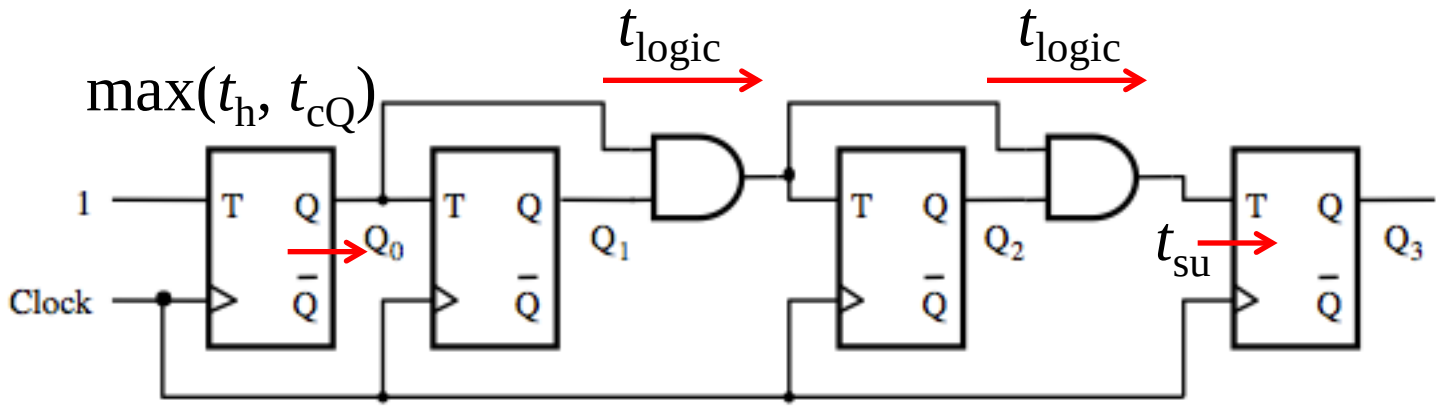
William Sandqvist william@kth.se

Maximal räknefrekvens?



William Sandqvist william@kth.se

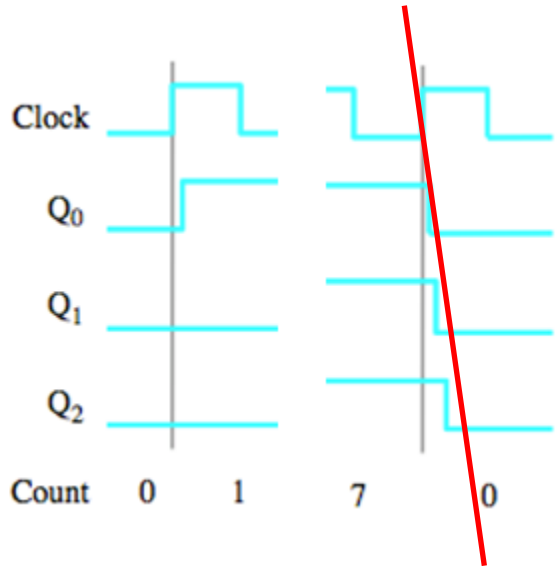
Maximal räknefrekvens?



Den kritiska vägen bestämmer den maximala frekvensen!
Här är den längsta kombinatoriska vägen från Q_0 via två AND-grindar till ingången av vippan som beräknar Q_3
t_{logic} motsvarar alltså fördröjningen av två AND-grindar.

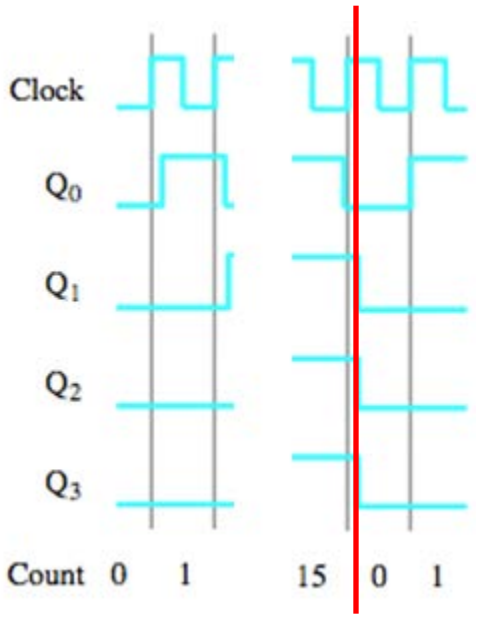
Asynkron eller Synkron räknare

Asynkron räknare



Utgångssignalerna fördröjs mer och mer för varje räknarsteg

Synkron räknare



Utgångssignalerna har samma fördröjning

William Sandqvist william@kth.se

William Sandqvist william@kth.se

VHDL för vippor och låskretsar

Programmerbar logik har inbyggda vippor.

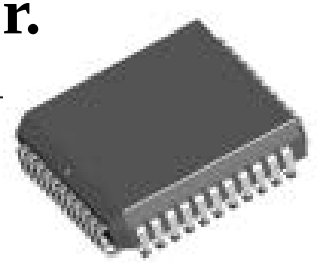
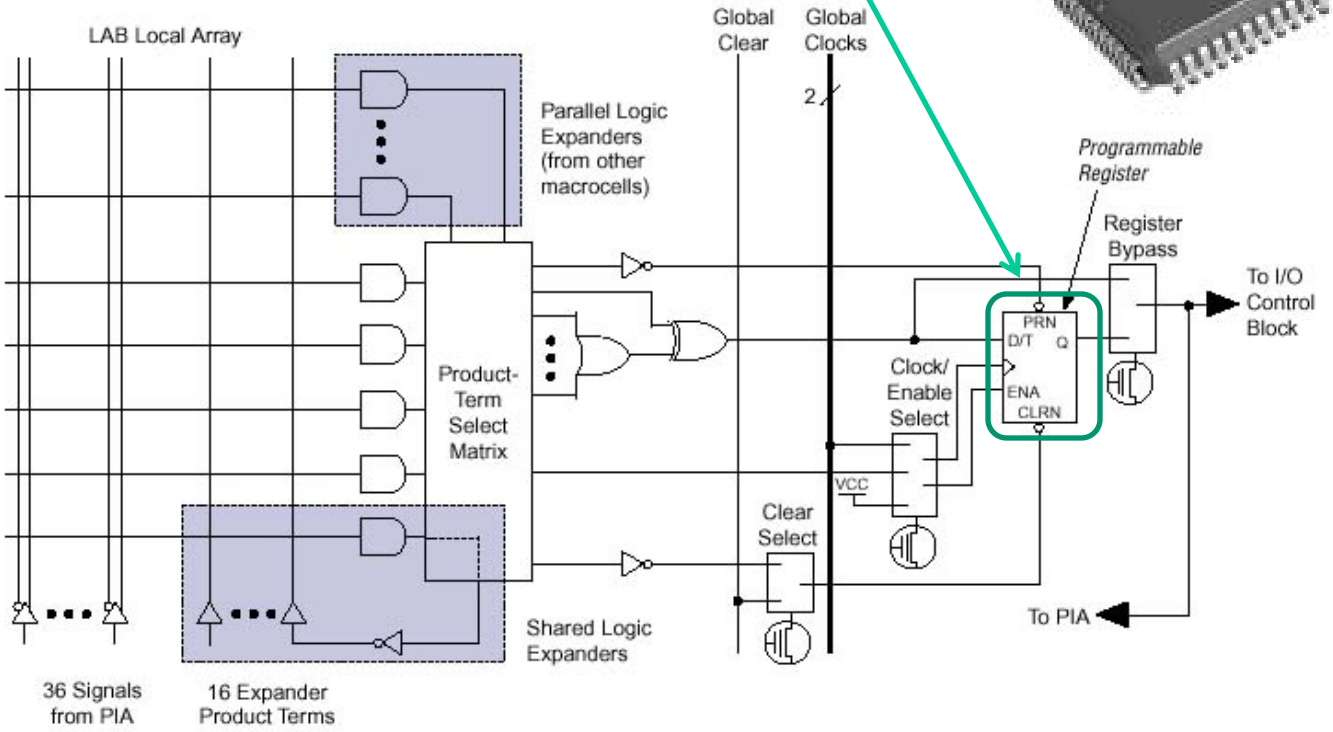


Figure 2. MAX 3000A Macrocell



VHDL för vippor och låskretsar

Programmerbar logik har inbyggda vippor.
Hur skriver man VHDL-kod som ”talar om” för kompilatorn att man vill använda dom?

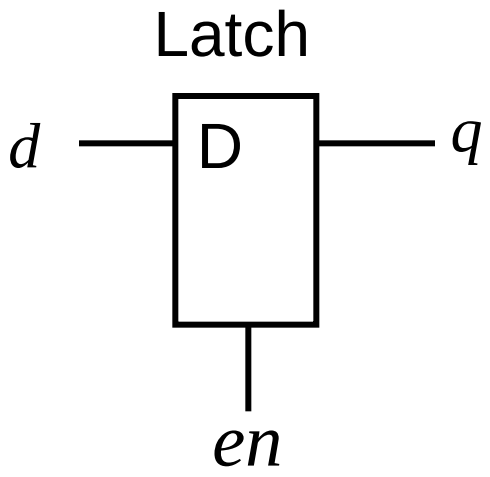
William Sandqvist william@kth.se

En D-latch i VHDL

```
ENTITY D_Latch IS
    PORT(en : IN std_logic;
         d : IN std_logic;
         q : OUT std_logic);
END ENTITY D_Latch;

ARCHITECTURE RTL OF D_Latch IS
BEGIN
    PROCESS(en, d)
    BEGIN
        IF en = '1' THEN
            q <= d;
        END IF;
    END PROCESS;
END ARCHITECTURE RTL;
```

Inget else? →



Enable	D	Q
0	-	M
1	D	D

William Sandqvist william@kth.se

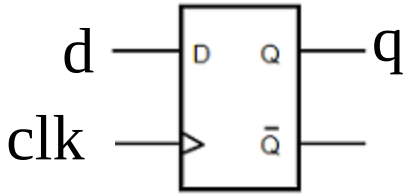
Latch som process

```
PROCESS(en, d)
    BEGIN
        IF en = '1' THEN
            q <= d;
        END IF;
END PROCESS;
```

Latchar anses generellt vara dåliga ur syntes-synpunkt eftersom de inte alltid är testbara (pga. asynkrona återkopplingar).

Därför undviker man latchar. (Programmerbar logik har inbyggda vippor med asynkron Preset och Clear som man kan använda).

Vippa som process



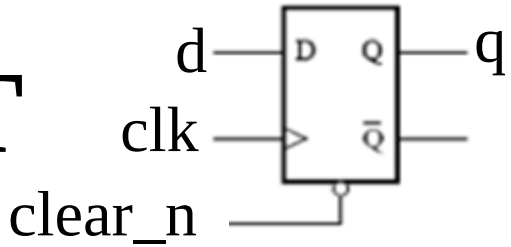

```
PROCESS(clk)
    BEGIN
       IF rising_edge(clk) THEN
          q <= d;
       END IF;
END PROCESS;
```

Endast en flank är tillåten per process

I stället för funktionen ”`rising_edge(clk)`” kan man skriva ”`clk'event and clk=1`”

Kompilatorn kommer att ”förstå” att detta är en vippa och använder någon av de inbyggda vipporna för att implementera processen.

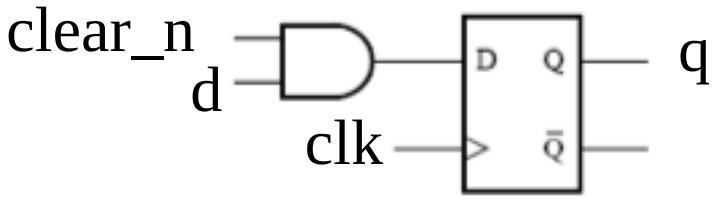
Med asynkron RESET



Clear oberoende av clk

```
PROCESS(clk, clear_n)
    BEGIN
        IF clear_n = '0' THEN
            q <= '0';
        ELSE IF rising_edge(clk) THEN
            q <= d;
        END IF;
END PROCESS;
```

Med synkron RESET




```
PROCESS(clk)
    BEGIN
       IF rising_edge(clk) THEN
       IF clear_n = '0' THEN
          q <= '0';
       ELSE
          q <= d;
       END IF;
END PROCESS;
```

Räknare och andra sekvenskretsar

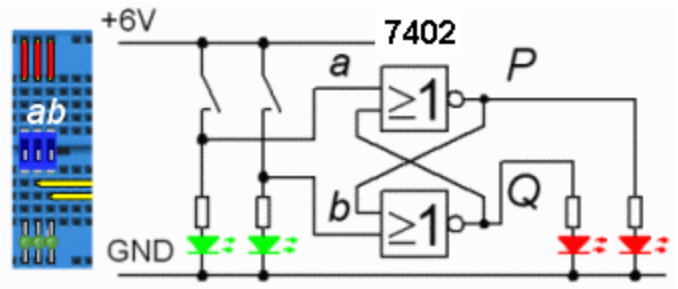
Vad gör den här ”räknaren” ?

```
bcd:
PROCESS(clk)
    BEGIN
       IF rising_edge(clk) THEN
          IF (count = 9) THEN
             count <= 0;
          ELSE
             count <= count+1;
          END IF;
       END IF;
END PROCESS;
```

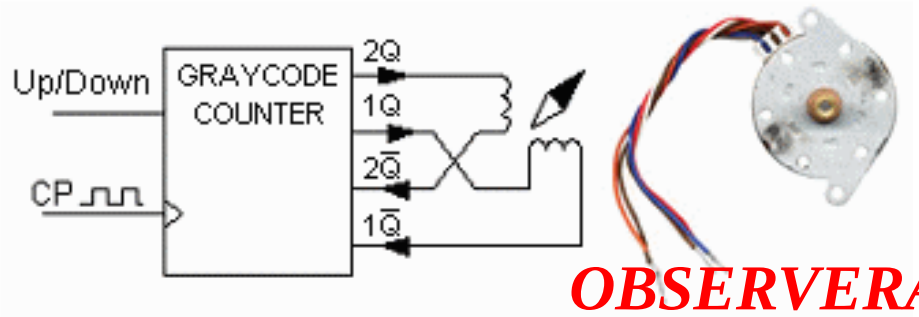
William Sandqvist william@kth.se

LAB Sekvenskretsar

Låskretsar …



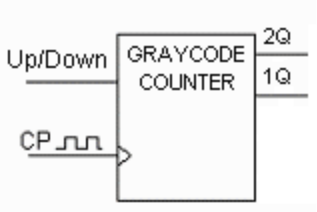
Graykodsräknare som stegmotorcontroller …



OBSERVERA! Mycket förberedelser inför LAB.

William Sandqvist william@kth.se

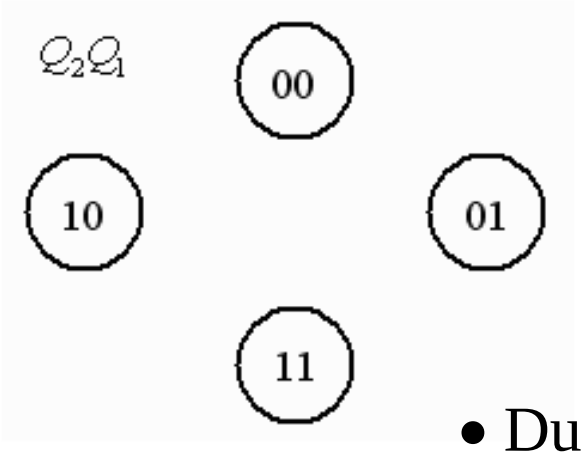
LAB Sekvens kretsar



Control signal	Counter mode
x=1	Up: Q_2Q_1= 00, 01, 11, 10, 00, ...
x=0	Down: Q_2Q_1= 00, 10, 11, 01, 00, ...

$$Q_2^+Q_1^+ = f(x, Q_2, Q_1)$$

Nästa tillstånd $Q_2^+Q_1^+$ är funktion av nuvarande tillstånd Q_2Q_1 och insignal ***x***



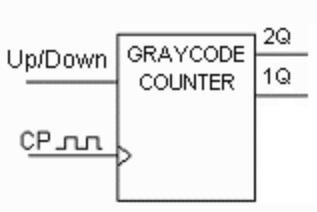
Present state $\xrightarrow{x = 0}$ Next state

Pilar mellan tillstånden visar övergångsvilkor

- Du skall rita ett tillståndsdiagram

William Sandqvist william@kth.se

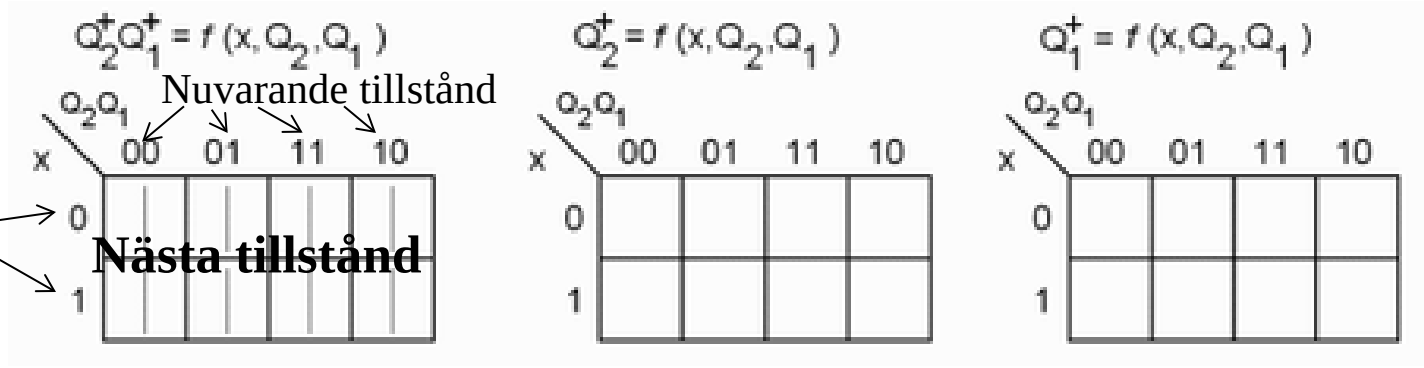
LAB Sekvens kretsar



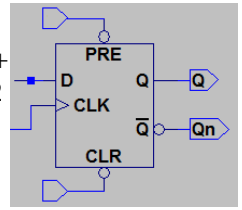
Control signal	Counter mode
x=1	Up: Q_2Q_1= 00, 01, 11, 10, 00, ...
x=0	Down: Q_2Q_1= 00, 10, 11, 01, 00, ...

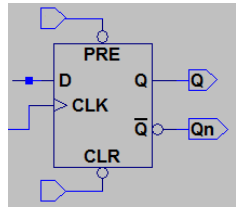
$$Q_2^+Q_1^+ = f(x, Q_2, Q_1)$$

Vänstra talen i tabellen Högra talen i tabellen



Tillståndstabell i Karnaugh diagram stil

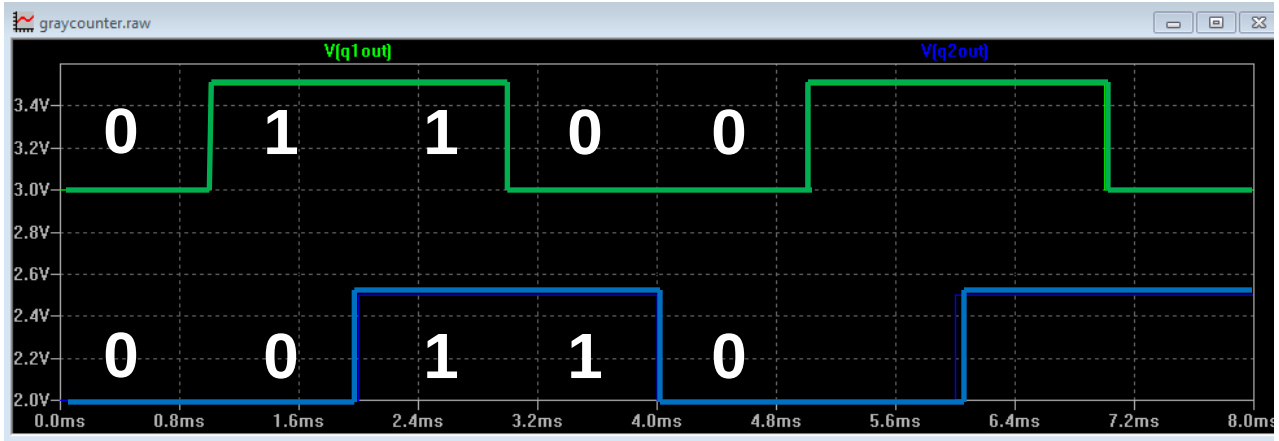


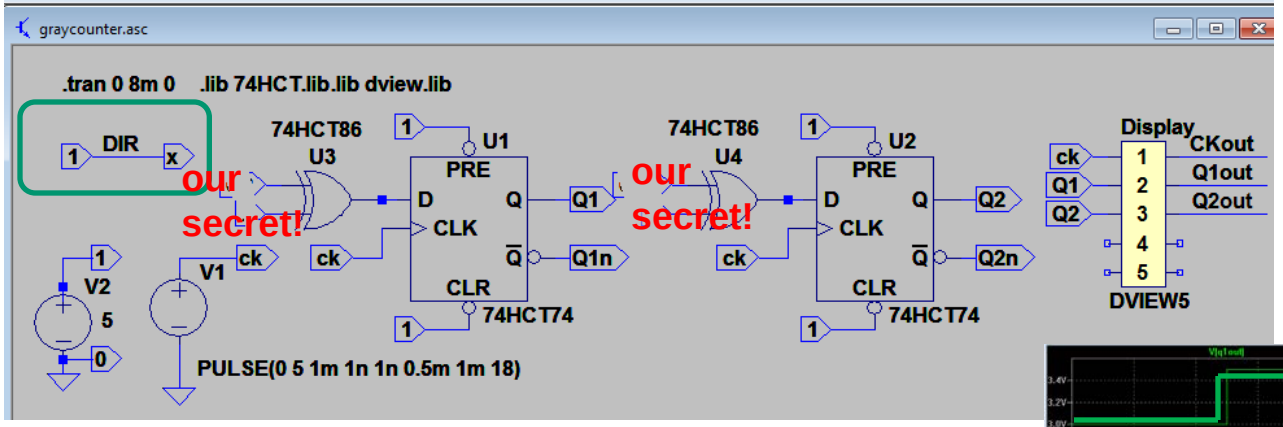


Ger oss logikfunktionerna till vipporna!

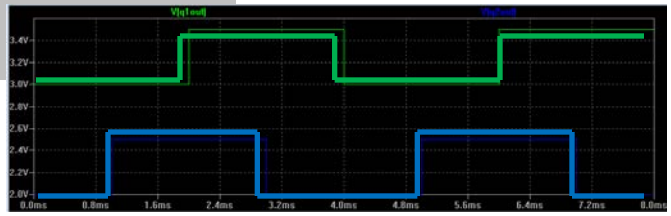
William Sandqvist william@kth.se

Simulera Gray räknaren





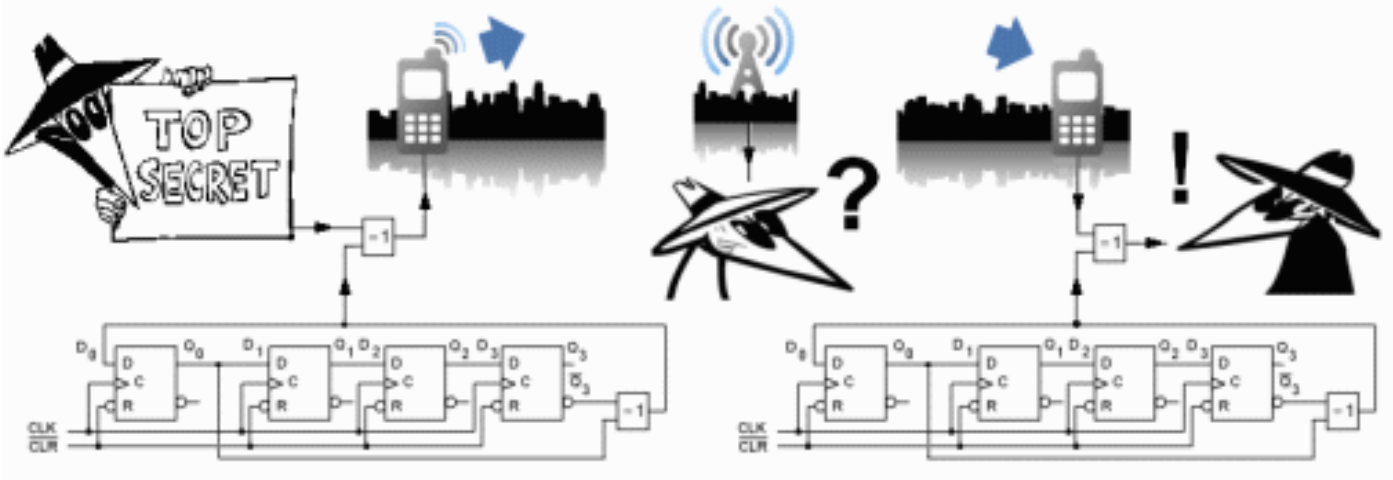
eller



William Sandqvist william@kth.se

LAB Sekvenskretsar

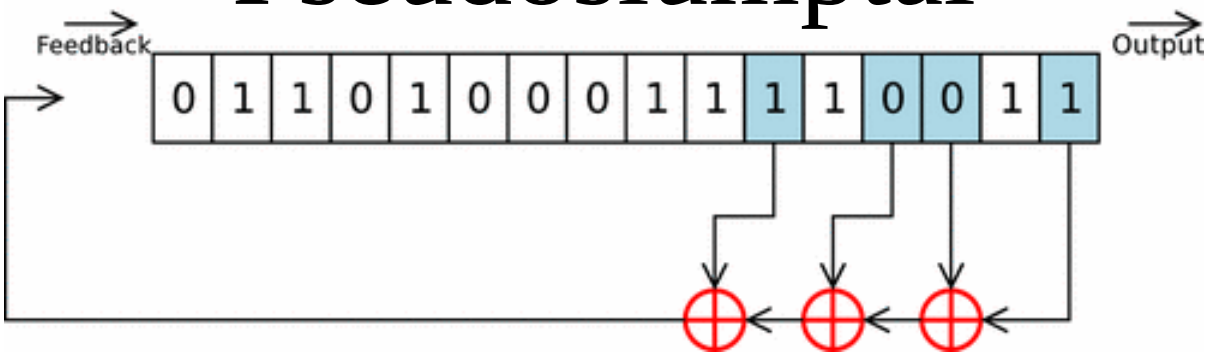
Shiftregisterräknare som genererar pseudoslumptal …



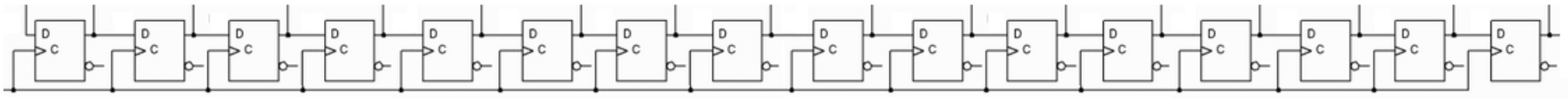
PRBS-sekvenser (pseudoslumptal) används tex. för att kryptera dataöverföringen vid GSM-telefoni och vid Bluetooth.
Ett annat användningsområde är för att bygga in "självtest-förmåga" i större digitala chip.

William Sandqvist william@kth.se

Pseudoslumptal

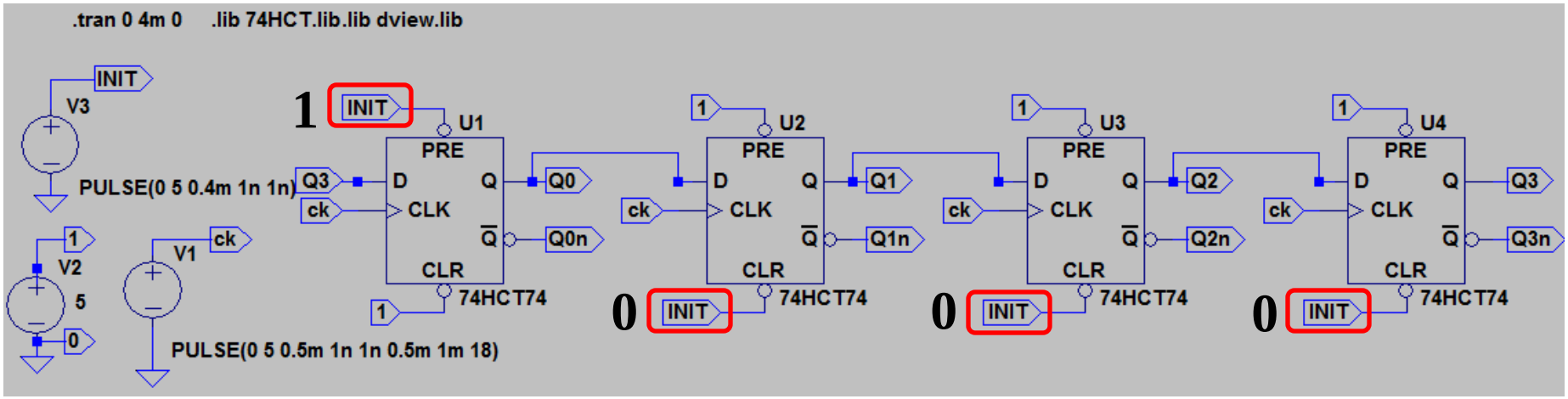


Exempel på ett återkopplat skiftregister med 16 vippor. I figuren sker "avtapp-ning" från vippa 0,2, 3, och 5.
Skiftregistrets ingång matas med EXOR-funktionen av dessa bitar. Denna "avtappning" ger en *maximalt* lång talsekvens som upprepas efter 65535 ggr.
Om alla vipporna i skiftregistret är “0” så stannar sekvensen, så den kombinationen måste undvikas!



William Sandqvist william@kth.se

Simulatorns vippor kan 0-ställas eller 1-ställas individuellt

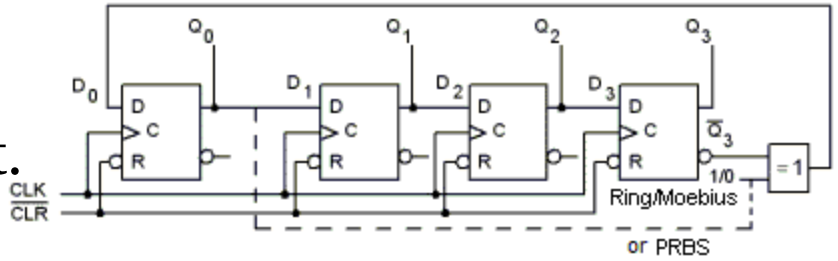


Den verkliga världens vippor som är djupt inbäddade inuti chippen kan inte påverkas på detta sätt. De måste testas annorlunda.

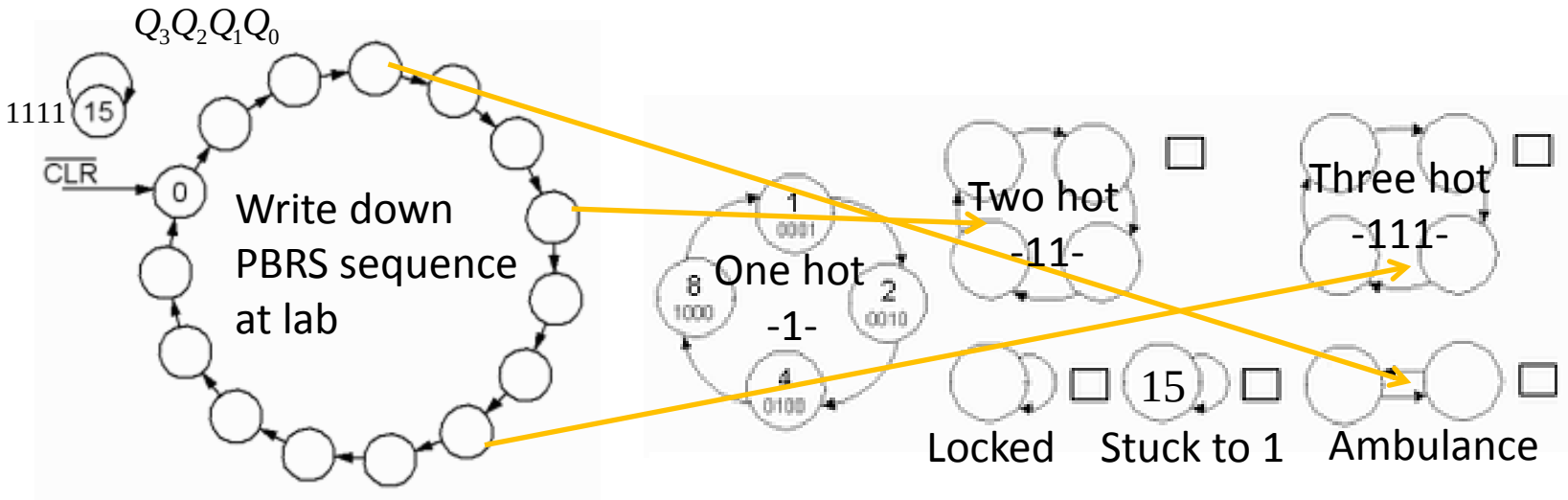
- Denna simulering startar med **1000**.

PRBS för att testa inbyggda kretsar

Alla vippor inuti 75175 kretsen 0-ställs samtidigt. De kan inte påverkas individuellt.



- Använd **PRBS** för att generera startvärden för de olika ringräknar cyklerna!



William Sandqvist william@kth.se

William Sandqvist william@kth.se