

Konstruktionsmetodik för sekvenskretsar

Föreläsning 7

Digitalteknik, TSEA22

Mattias Krysanter

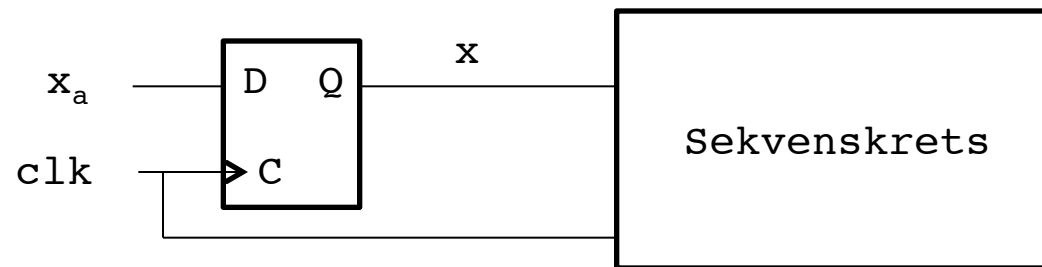
Institutionen för systemteknik

Dagens föreläsning

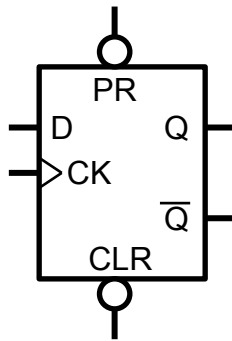
- Inför laboration 2
 - Synkronisering av insignaler
 - Asynkrona ingångar på vippor
 - Initiering/nollställning
 - Programmerbara kretsar och VHDL
- Från problemformulering till tillståndsdigram

Synkronisera asynkrona insignaler

- Asynkron insignaler x_a kommer från
 - Brytare
 - Sensorer
- Använd synkroniseringsvipa:



D-vippa med asynkrona ingångar



PR	CLR	CK	D	Q ⁺
1	0	X	X	0
0	1	X	X	1
1	1	0	X	Q
1	1	1	X	Q
1	1	↑	0	0
1	1	↑	1	1

Asynkrona ingångar:
Clear (CLR), Preset (PR) aktivt låg

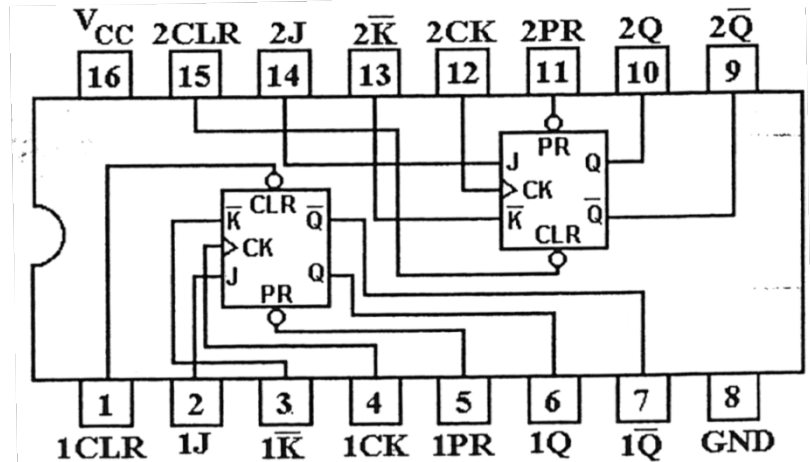
$$\text{CLR} = 0 \Rightarrow Q = 0$$

$$\text{PR} = 0 \Rightarrow Q = 1$$

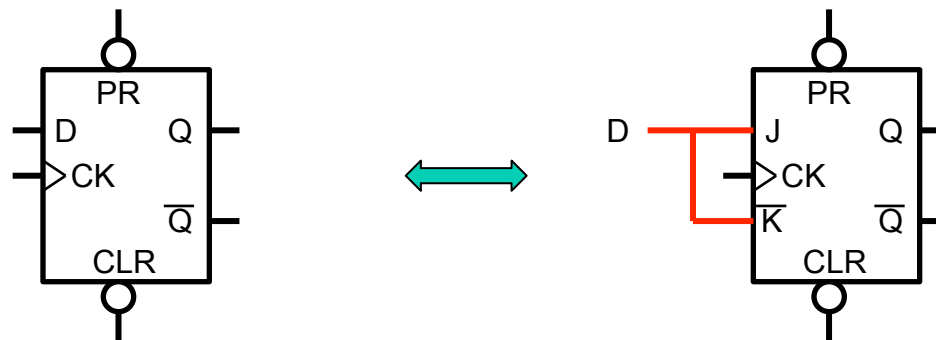
Lab-vippan

- Positivt flanktriggad J $\bar{K}$ -vippa med asynkron clear och preset.
- $J = 1$ sätter Q till 1, $\bar{K} = 0$ reserar Q till 0.

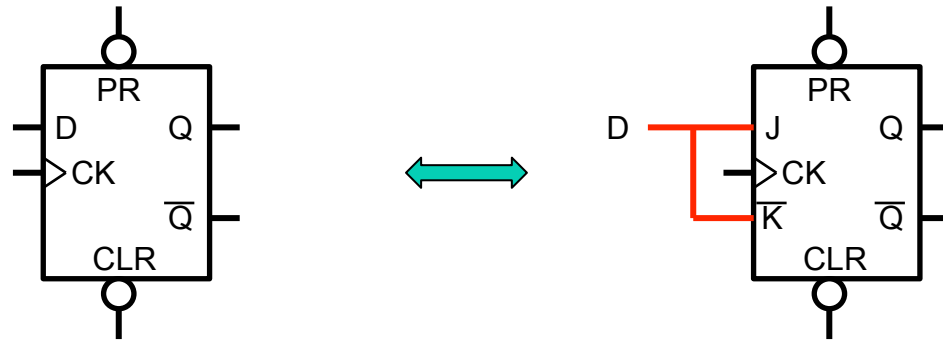
Inputs					Outputs	
PR	CLR	CLK	J	$\bar{K}$	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	$\uparrow$	L	L	L	H
H	H	$\uparrow$	L	L	Toggle	
H	H	$\uparrow$	L	L		
H	H	$\uparrow$	H	H	Q_0	$\bar{Q}_0$
H	H	$\uparrow$	H	H	H	L
H	H	L	X	X	Q_0	$\bar{Q}_0$



- Koppla ihop J och $\bar{K}$ ingången för att få en D-vippa.



Lab-vippan

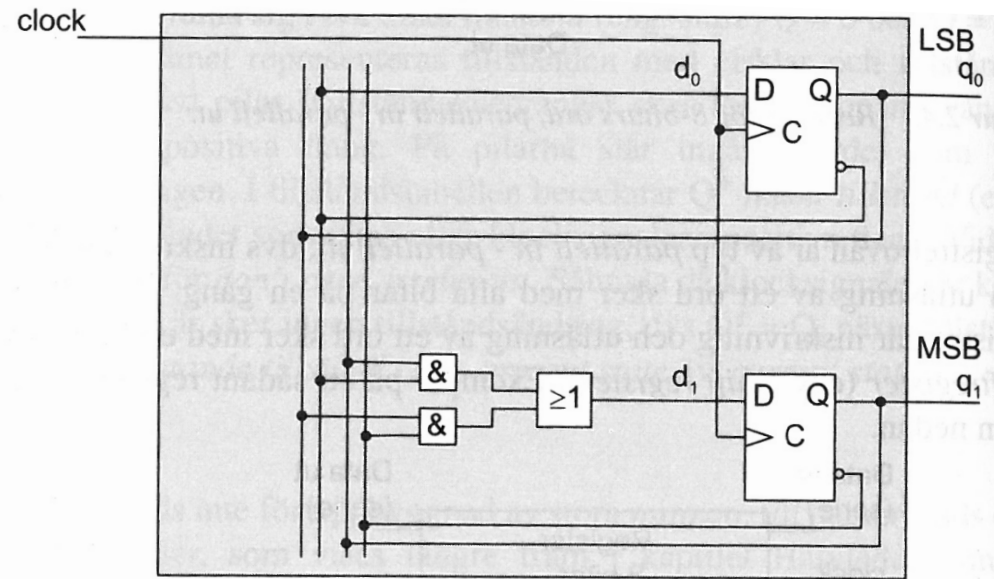


- Asynkrona ingångar får endast användas för att ställa in starttillståndet.
- I övriga fall skall $\text{CLR} = 1$, $\text{PR} = 1$.
- Klocksignalen får inte ledas genom grindar innan den skickas in till vippor, räknare, etc ty det ger en asynkron krets.

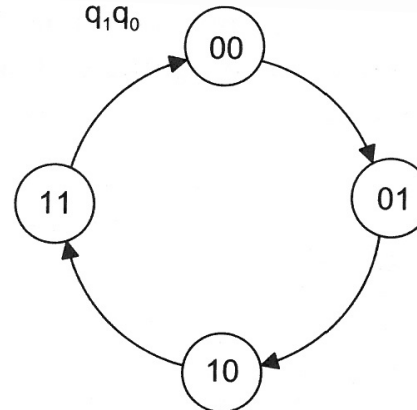
Nollställning ett exempel

Autonom 2-bitsräknare
ska förses med
nollställning

- Asynkront
- Synkront



Kretschema
 $q_1 q_0$



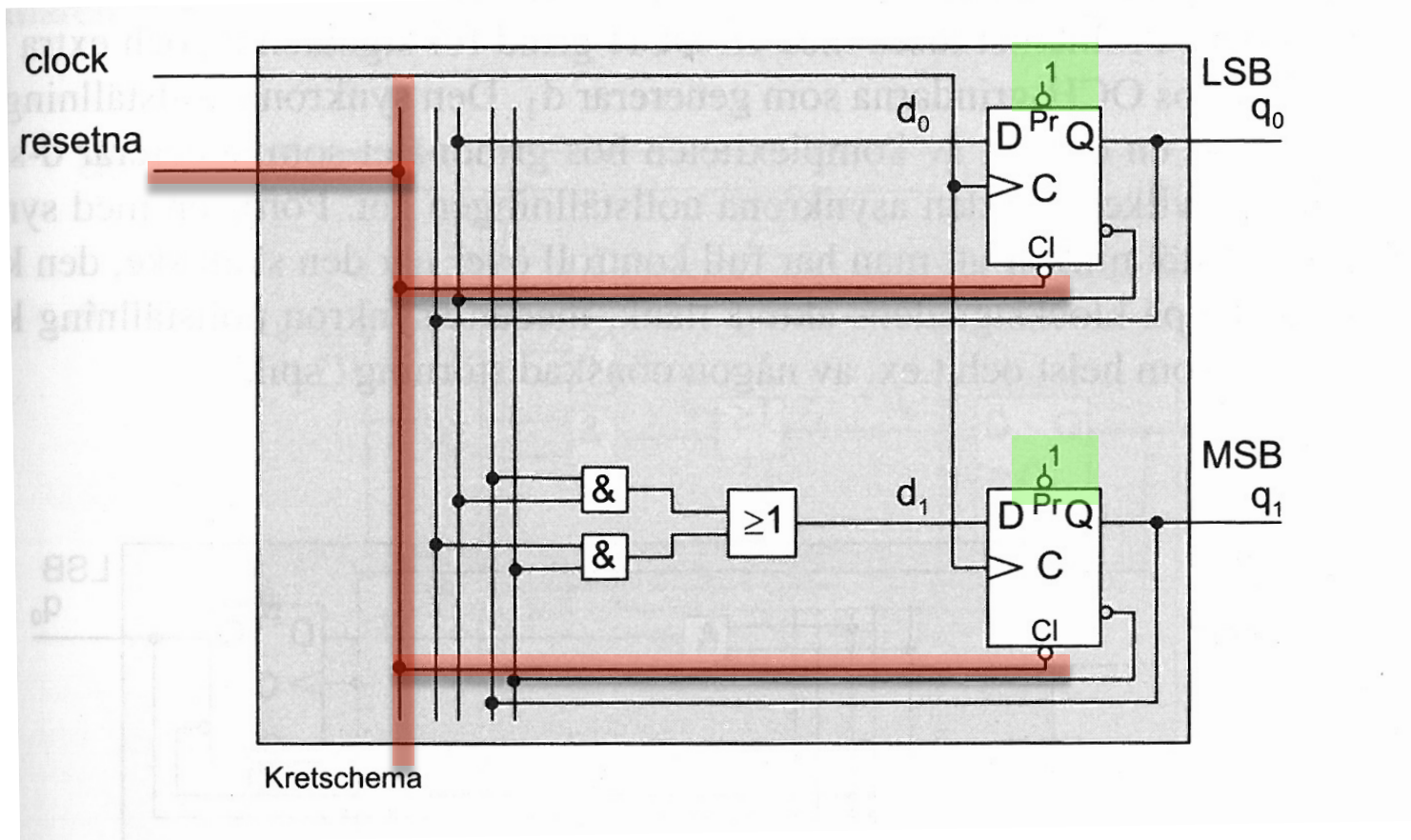
Tillståndsdigram

Nuvarande tillstånd	Nästa tillstånd
00	01
01	10
10	11
11	00

Tillståndstabell

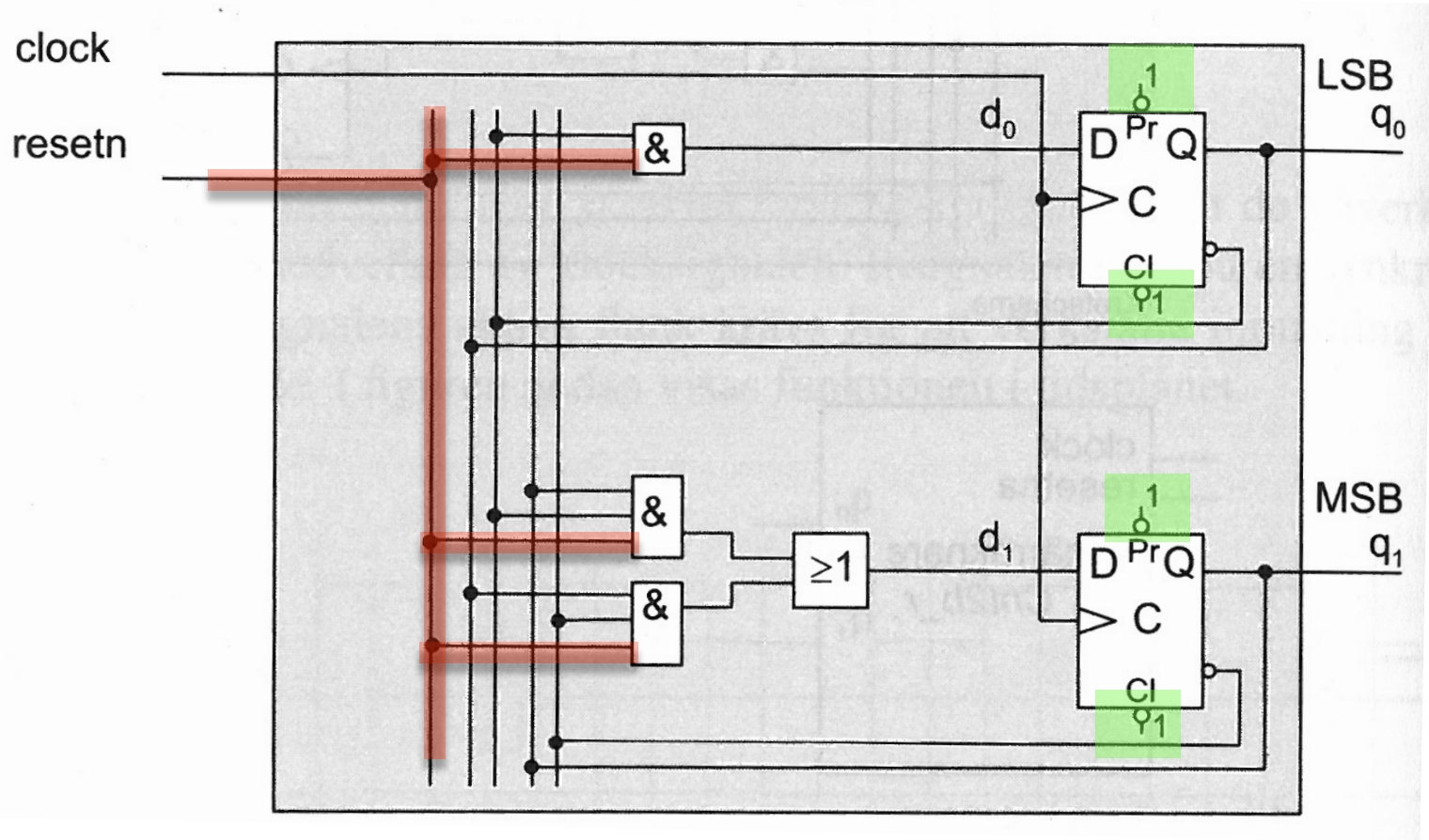
Asynkron nollställning

- Nollställning sker med $\text{resetna} = 0$, utan medverkan av klockan.



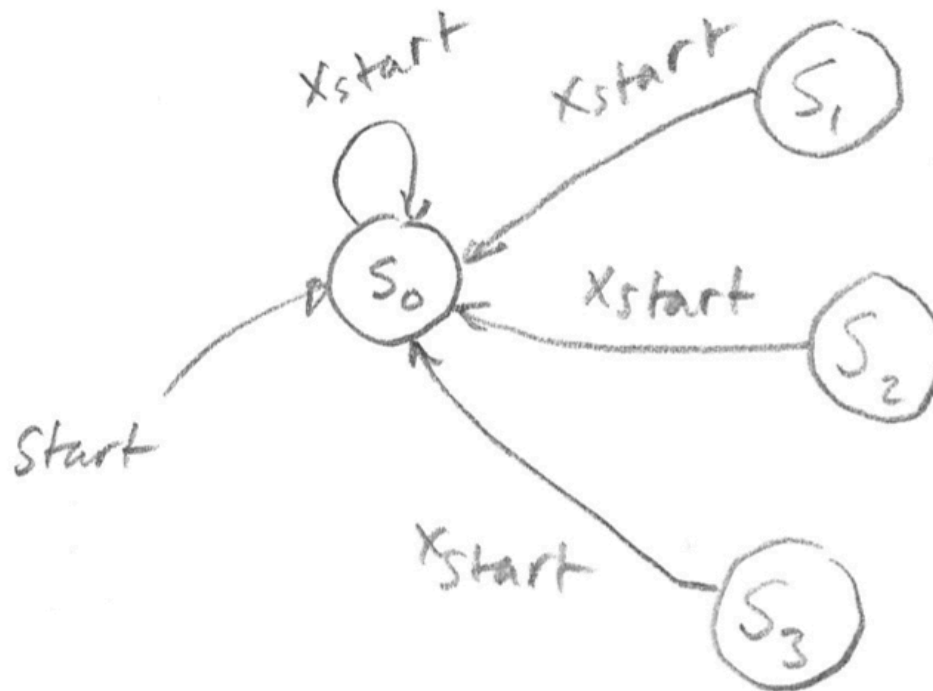
Synkron nollställning

- Synkron nollställning aktiverad med $\text{resetn} = 0$



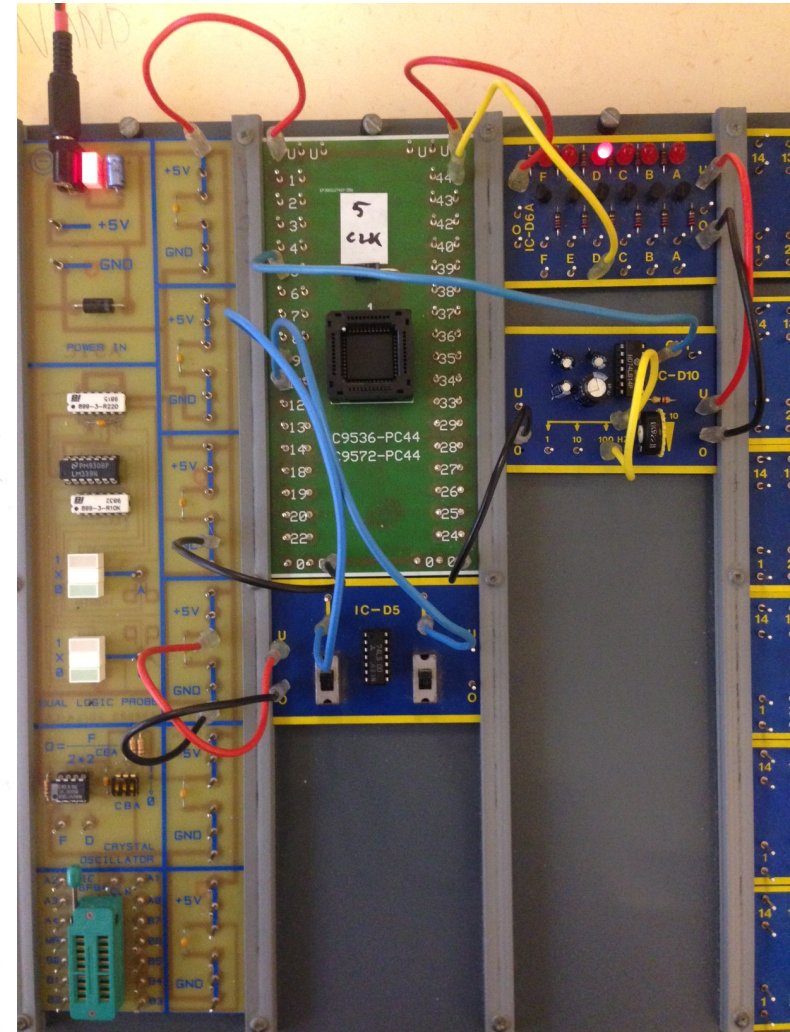
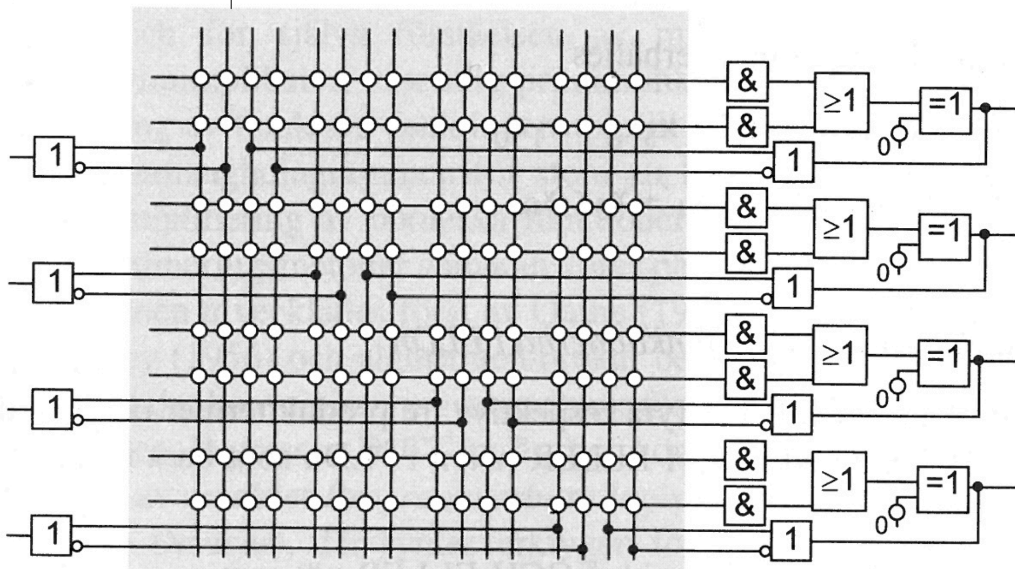
Synkron nollställning

- En insignalkombination sätter nästa tillstånd till starttillståndet oberoende av nuvarande tillstånd.
 - Hanteras därmed som vilken insignal som helst



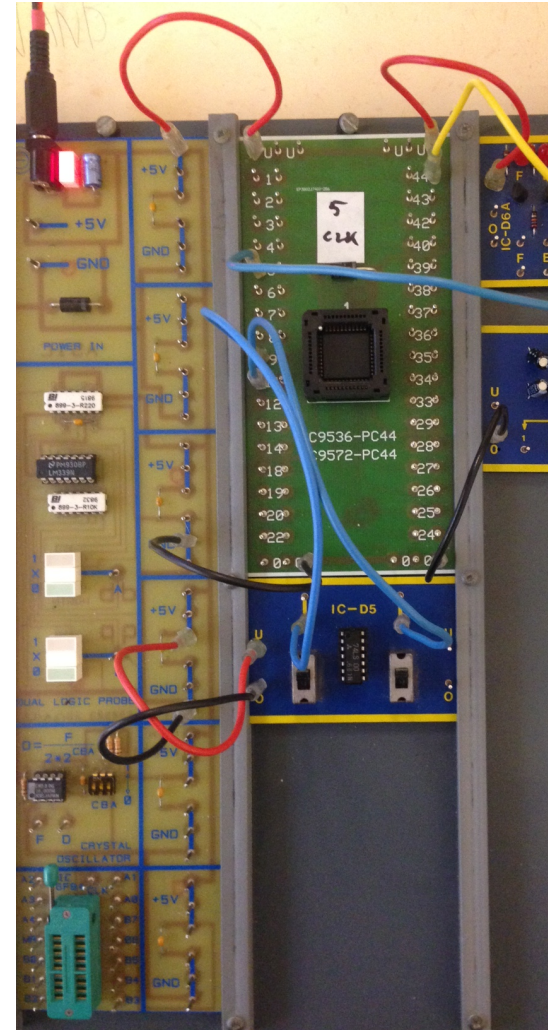
Programmerbar logik

- Complex programmable logic device (CPLD)
- Fix komponentuppsättning
- Programmerar vilka kopplingar som ska finnas



Konstruktion med CPLD

- Skriv beskrivning i VHDL
(Very high speed integrated circuit **H**ardware **D**escription Language)
- Syntes
 - passar in (optimerar) kretsen på den aktuella CPLD:n
 - bestämmer vilka in och utgångar som kommer att användas
- Programmering
 - speciell mjuk- och hård-vara används för att programmera CPLD:n



VHDL-exempel - enpulsaren

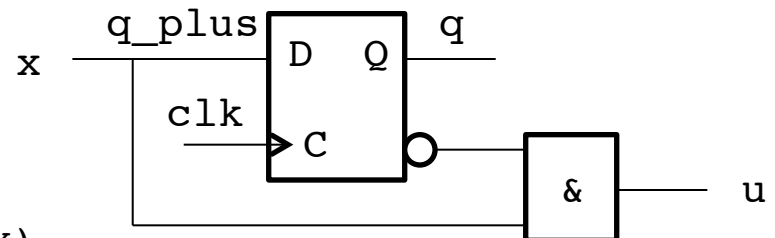
```

library ieee;
use ieee.std_logic_1164.all;

entity enpulsare is
  port(clk, x : in  std_logic;
        u : out std_logic);
end enpulsare;

architecture ekvationer of enpulsare is
  signal q, q_plus : std_logic;
begin
  process(clk)
  begin
    if rising_edge(clk) then
      q <= q_plus;
    end if;
  end process;
  q_plus <= x;
  u <= (not q) and x;
end ekvationer;

```



-- $q^+ = f(q, x)$
 -- $u = g(q, x)$

VHDL-exempel - enpulsaren

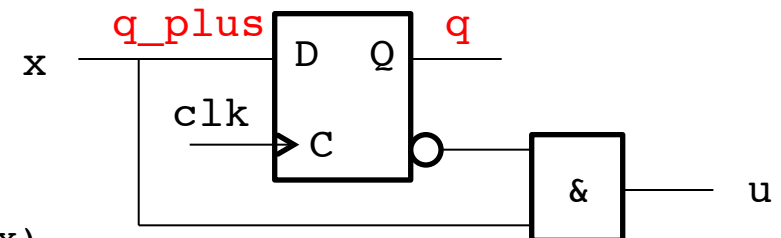
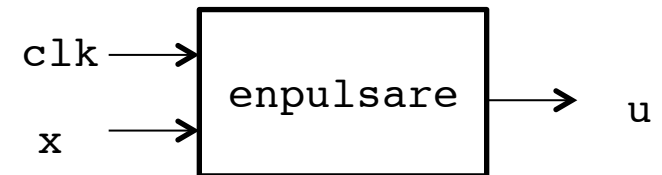
```

library ieee;
use ieee.std_logic_1164.all;

entity enpulsare is
  port(clk, x : in  std_logic;
        u : out std_logic);
end enpulsare;

architecture ekvationer of enpulsare is
  signal q, q_plus : std_logic;
begin
  process(clk)
  begin
    if rising_edge(clk) then
      q <= q_plus;
    end if;
  end process;
  q_plus <= x;
  u <= (not q) and x;
end ekvationer;

```



$q^+ = f(q, x)$
 $u = g(q, x)$

VHDL-exempel - enpulsaren

```

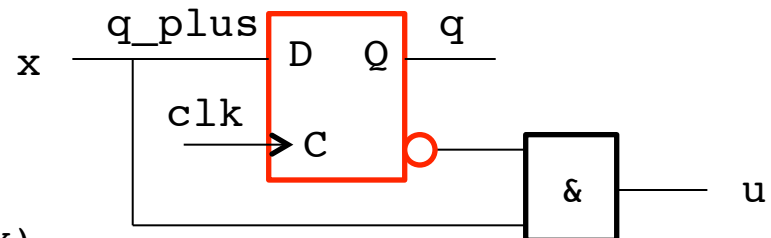
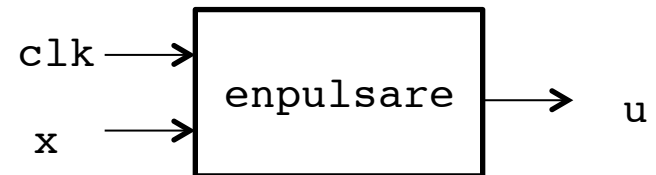
library ieee;
use ieee.std_logic_1164.all;

entity enpulsare is
    port(clk, x : in  std_logic;
          u : out std_logic);
end enpulsare;

architecture ekvationer of enpulsare is
    signal q, q_plus : std_logic;
begin
    process(clk)
    begin
        if rising_edge(clk) then
            q <= q_plus;
        end if;
    end process;
    q_plus <= x;
    u <= (not q) and x;
end ekvationer;

```

-- $q^+ = f(q, x)$
 -- $u = g(q, x)$



VHDL-exempel - enpulsaren

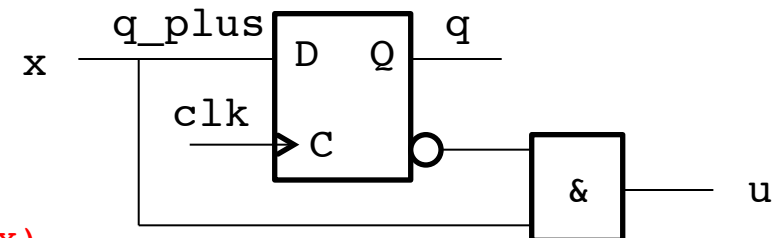
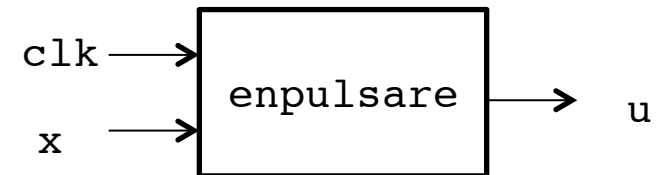
```

library ieee;
use ieee.std_logic_1164.all;

entity enpulsare is
  port(clk, x : in  std_logic;
        u : out std_logic);
end enpulsare;

architecture ekvationer of enpulsare is
  signal q, q_plus : std_logic;
begin
  process(clk)
  begin
    if rising_edge(clk) then
      q <= q_plus;
    end if;
  end process;
  q_plus <= x;                                -- q+ = f(q,x)
  u <= (not q) and x;                          -- u = g(q,x)
end ekvationer;

```



VHDL-exempel - enpulsaren

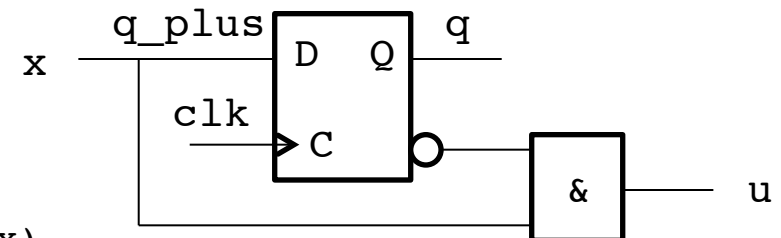
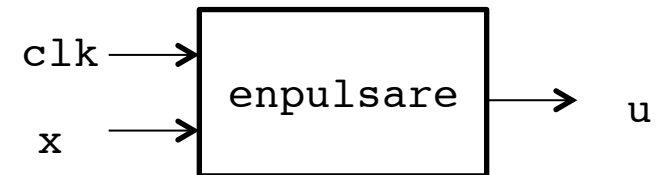
```

library ieee;
use ieee.std_logic_1164.all;

entity enpulsare is
    port(clk, x : in  std_logic;
          u : out std_logic);
end enpulsare;

architecture ekvationer of enpulsare is
    signal q, q_plus : std_logic;
begin
    process(clk)
    begin
        if rising_edge(clk) then
            q <= q_plus;
        end if;
    end process;
    q_plus <= x;
    u <= (not q) and x;
end ekvationer;

```



```

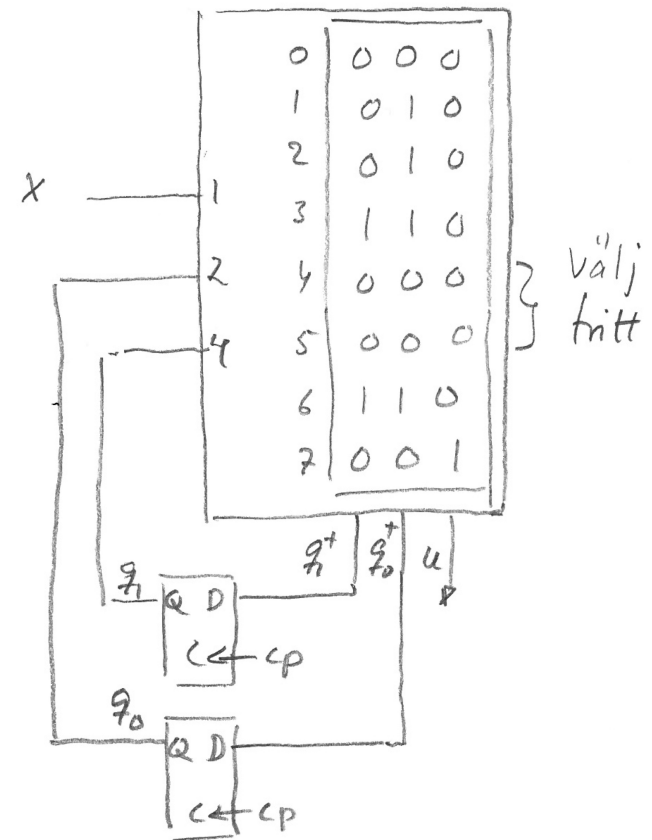
-- q+ = f(q,x)
-- u = g(q,x)

```

Detektion av var tredje etta i VHDL

	4	2	1			
	q_1	q_0	x		q_1^+	q_0^+
	u				u	
0	0	0	0		0	0
1	0	0	1		0	1
2	0	1	0		0	1
3	0	1	1		1	1
4	1	0	0		-	-
5	1	0	1		-	-
6	1	1	0		1	1
7	1	1	1		0	0

adress utsignal

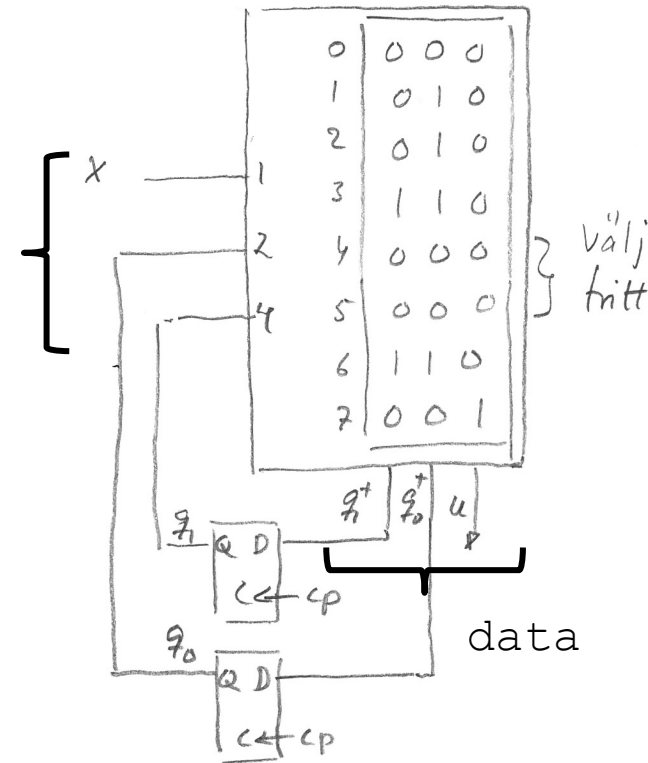


Interna signaler

```
address = (q1, q0, x)
data    = (q1_plus, q0_plus, u)
```

```
architecture behavior of skrets is
  signal q0, q0_plus: std_logic;
  signal q1, q1_plus: std_logic;
  signal address : std_logic_vector(2 downto 0);
  signal data    : std_logic_vector(2 downto 0);
begin
  -- beskrivning av kretsens beteende
  -- se kommande 2 oh:ar
end behavior;
```

address



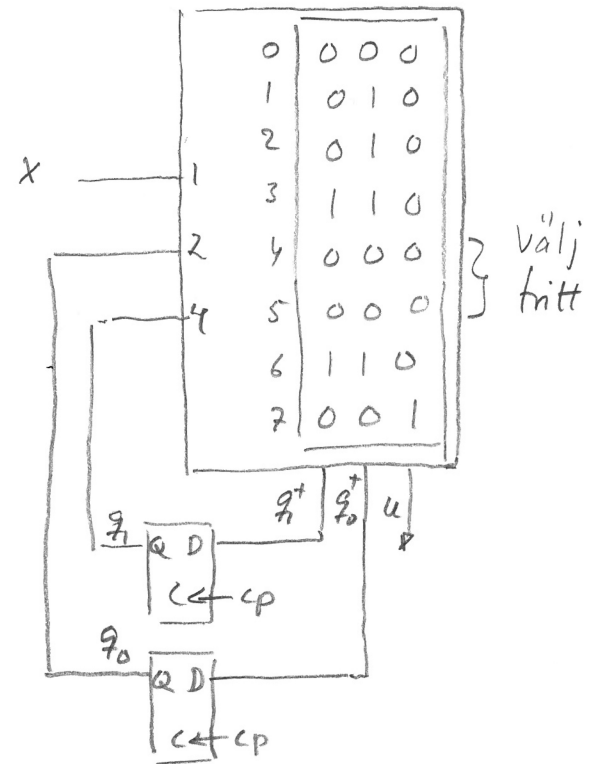
data

Vippor

```

-- vippor
process (clk)
begin
    if rising_edge (clk) then
        q0 <= q0_plus;
        q1 <= q1_plus;
    end if;
end process;

```



Minnet

-- ROM

address <= q1 & q0 & x; -- concatenering

with address select

```
data <= "000" when "000",
        "010" when "001",
        "010" when "010",
        "110" when "011",
        "000" when "100",
        "000" when "101",
        "110" when "110",
        "001" when "111",
        "---" when others;
```

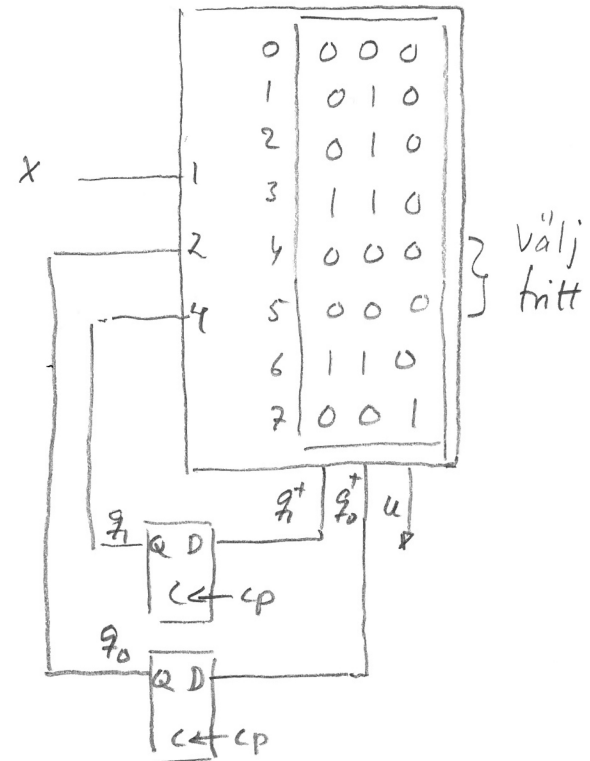
-- nästa tillstånd

q1_plus <= data(2); -- indexera vektor

q0_plus <= data(1);

-- utsignal

u <= data(0);

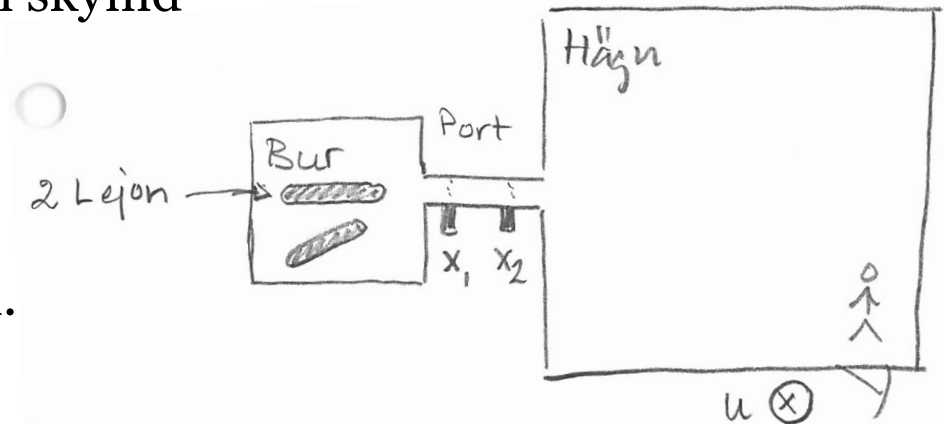


Dagens föreläsning

- Inför laboration 2
 - Synkronisering av insignaler
 - Asynkrona ingångar på vippor
 - Initiering/nollställning
 - Programmerbara kretsar och VHDL
- Från problemformulering till tillståndsdiagram

Lejonburen

- Lampa skall lysa när hägnet är tomt.
- Fotoceller: $x_i = \begin{cases} 1 & \text{1 fotocellen skymd} \\ 0 & \text{annars} \end{cases}$
- Lampa: $u = \begin{cases} 0 & \text{släkt} \\ 1 & \text{tänd} \end{cases}$
- Vid start är båda lejonerna i buren.
- Lejonerna:
 - a) Max ett lejon i porten
 - b) Kan ej vända/backa i porten.
 - c) Är längre än avståndet mellan x_1 och x_2 .
 - d) Rör sig långsamt i förhållande till klockfrekvensen.



Senarier

- Porten tom: $x = (x_1, x_2) = 00$
- Lejon i port:
 - a) $\Rightarrow$ ett lejon
 - b) $\Rightarrow$ rör sig genom porten

Fall

– Lejon ut ur bur:

$\xrightarrow{\hspace{1.5cm}}$
 $x : 00, 01, 11, 01, 00$
 $\xleftarrow{\hspace{1.5cm}}$

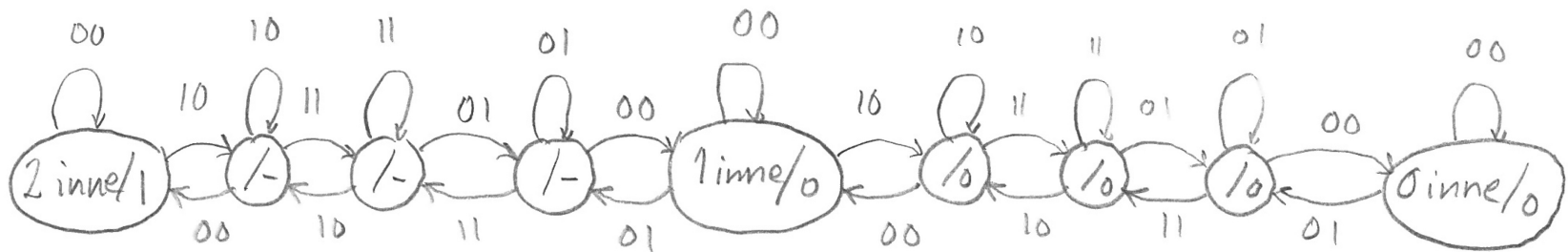
($x = 11$, pga c))

– Lejon in i bur:

- d) $\Rightarrow$ Varje insignalkombination upprepas
- $\Rightarrow$ Vi kan vänta en klockpuls med att tända lampan, dvs Moore-krets är okej.

Enkel lösning

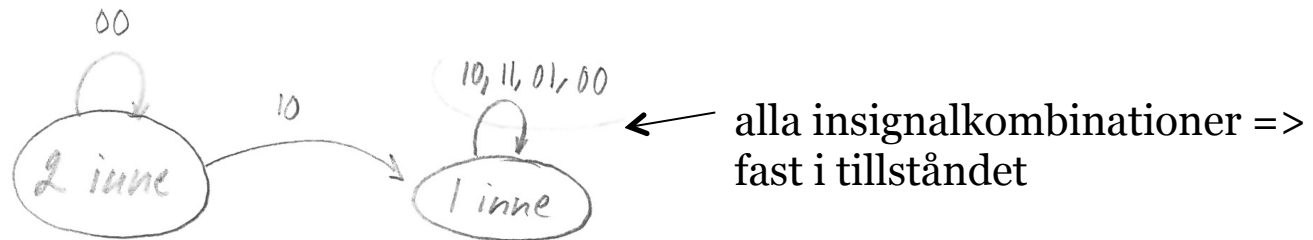
- Låt tillståndet “i inne” beteckna att i lejon är i buren.



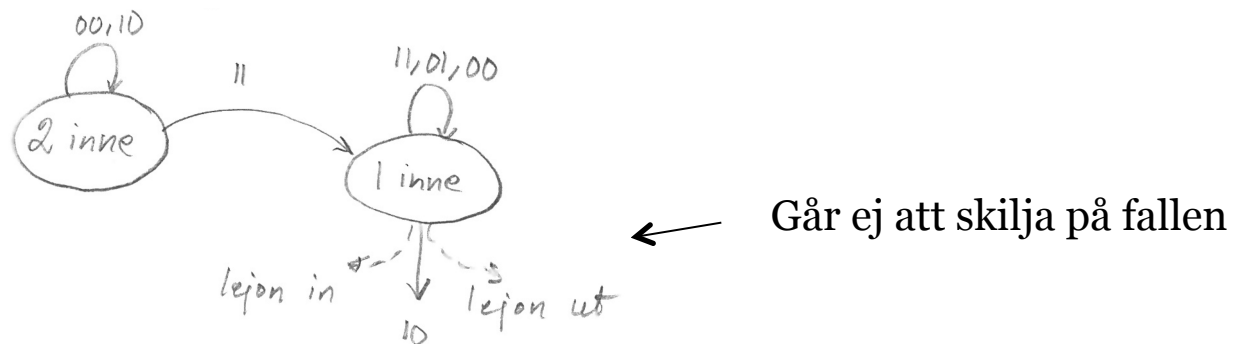
- Onödigt många tillstånd: 9 st
 - Det finns algoritmer som minimerar antalet tillstånd.
 - Hanterar även att lejon backar i porten
- => Onödigt komplicerat tillståndsdigram

Lösning med få tillstånd

- Vi behöver ett tillstånd då lampan ska lysa ($u = 1$). I tillståndet “2 inne” lyser lampan i övriga tillstånd är den släkt.
- Detektera utpassage med 10



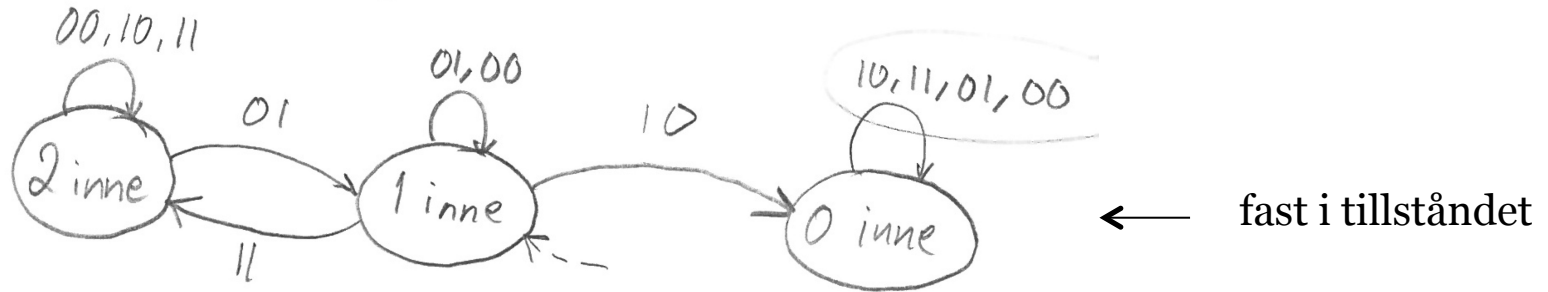
- Detektera utpassage med 11



Lejonbur fortsättning ...

28

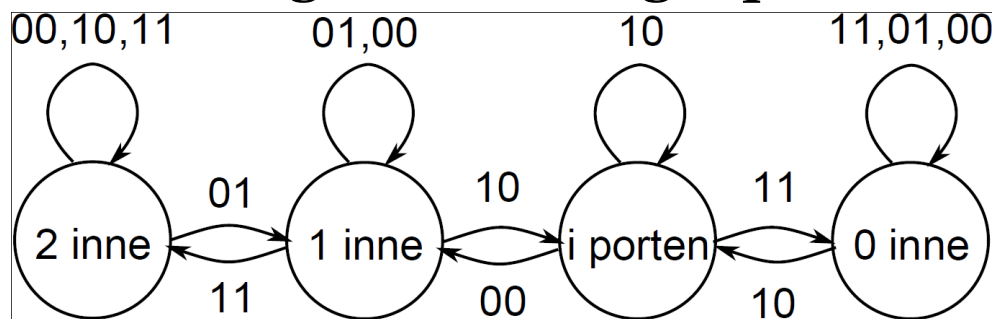
- Detektera utpassage med 01



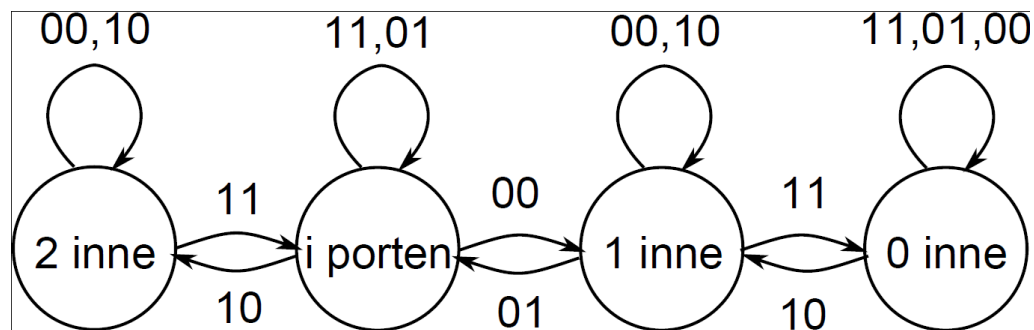
- 10, 11 kvar
 - 11 påträffas först vid inpassage
 - 10 påträffas först vid utpassage
- Vilka insignaler för streckad tillståndsövergång är tänkbar?
 - 01: funkar ej ty vi hamnar i "2 inne"
 - 11: funkar ej ty vi hamnar i "2 inne"
 - 10: funkar ej ty vi hamnar i "0 inne"
 - 00: OK, men nytt tillstånd måste införas: "i porten"

Tillståndsdigram för lejonburen

- Tillståndsdigram framtaget på föreläsningen:



- Alternativt tillståndsdigram:



Digitalteknik

Mattias Krysender

www.liu.se