Konstruktionsmetodik för sekvenskretsar

Digitalteknik – Föreläsning 7

Mattias Krysander

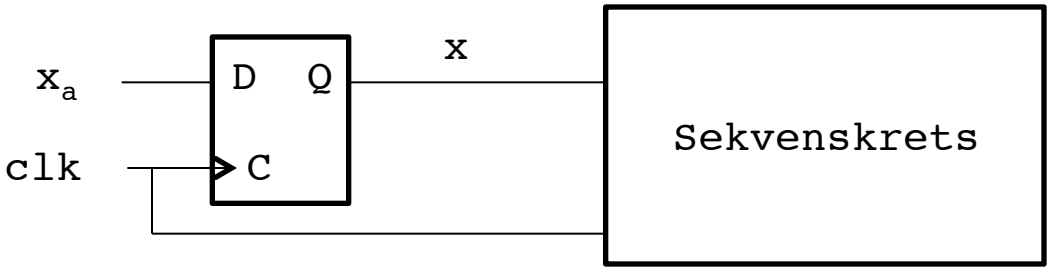
Institutionen för systemteknik

Dagens föreläsning

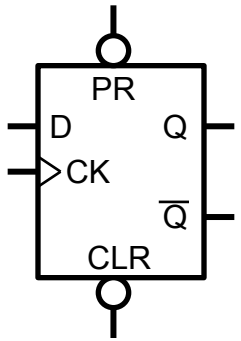
- Inför laboration 2
  - Synkronisering av insignaler
  - Asynkrona ingångar på vippor
  - Initiering/nollställning
  - Programmerbara kretsar och VHDL
- Från problemformulering till tillståndsdiagram

Synkronisera asynkrona insignaler

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



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

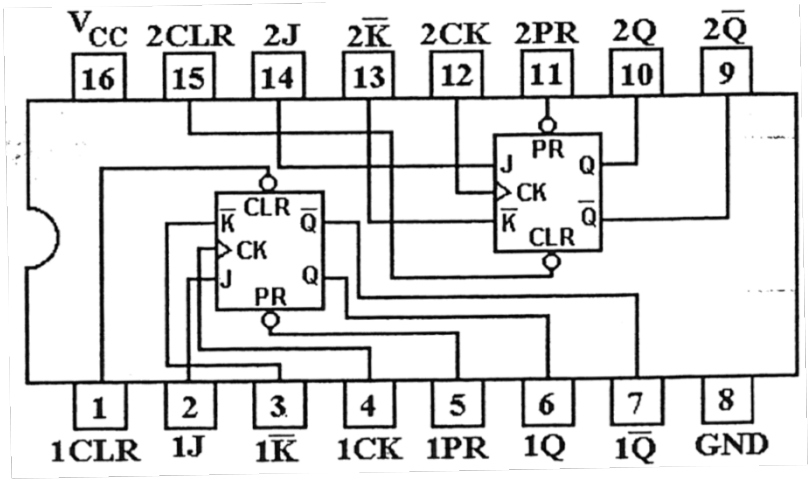
  CLR = 0 => Q = 0

  PR = 0 => Q = 1

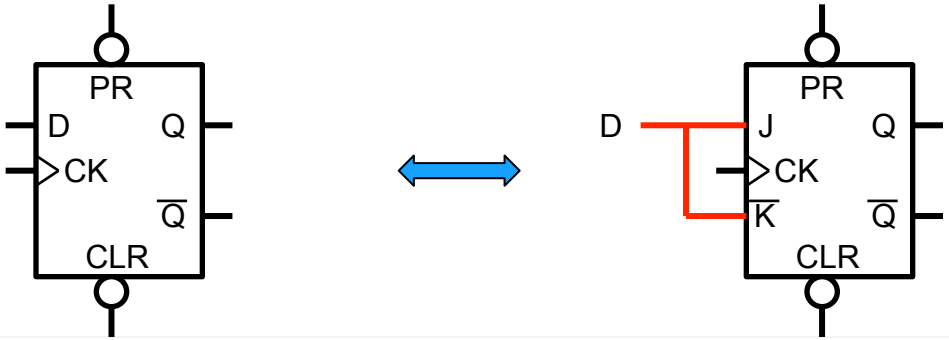
Lab-vippan

- Positivt flanktriggad J$\overline{K}$-vippa med asynkron clear och preset.

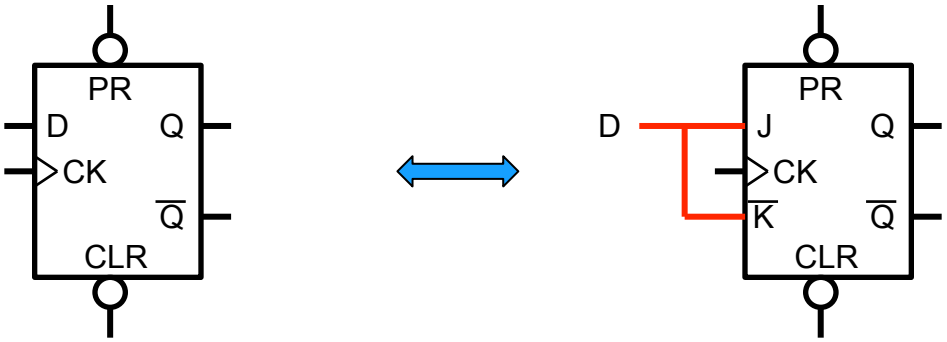
Inputs					Outputs	
PR	CLR	CLK	J	$\overline{K}$	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↑	L	L	L	H
H	H	↑	H	L	Toggle	
H	H	↑	L	H	Q_0	$\overline{Q}_0$
H	H	↑	H	H	H	L
H	H	L	X	X	Q_0	$\overline{Q}_0$



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



Lab-vippan

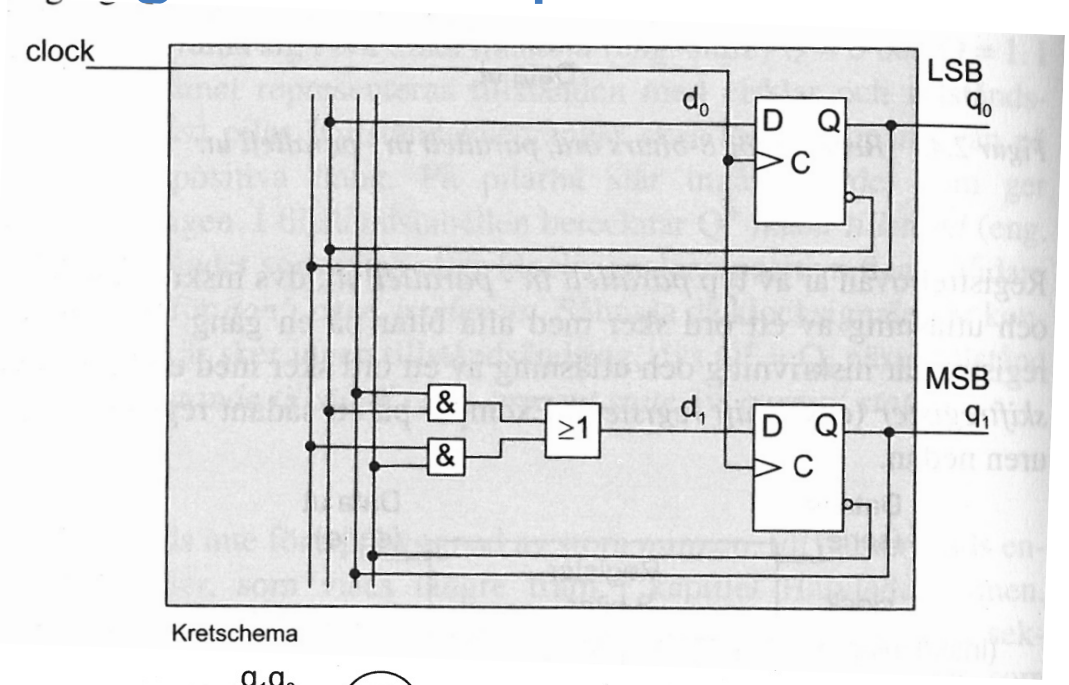


- Asynkrona ingångar får endast användas för att ställa in starttillståndet.
- I övriga fall skall CLR = 1, PR = 1.
- Klocksignalen får inte ledas genom grindar innan den skickas in till vippor, räknare, etc ty det ger en asynkron krets.

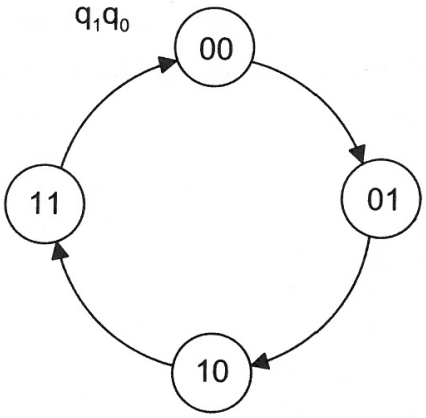
Nollställning ett exempel

Autonom tvåbitsräknare ska förses med nollställning

- Asynkront
- Synkront



Kretschema



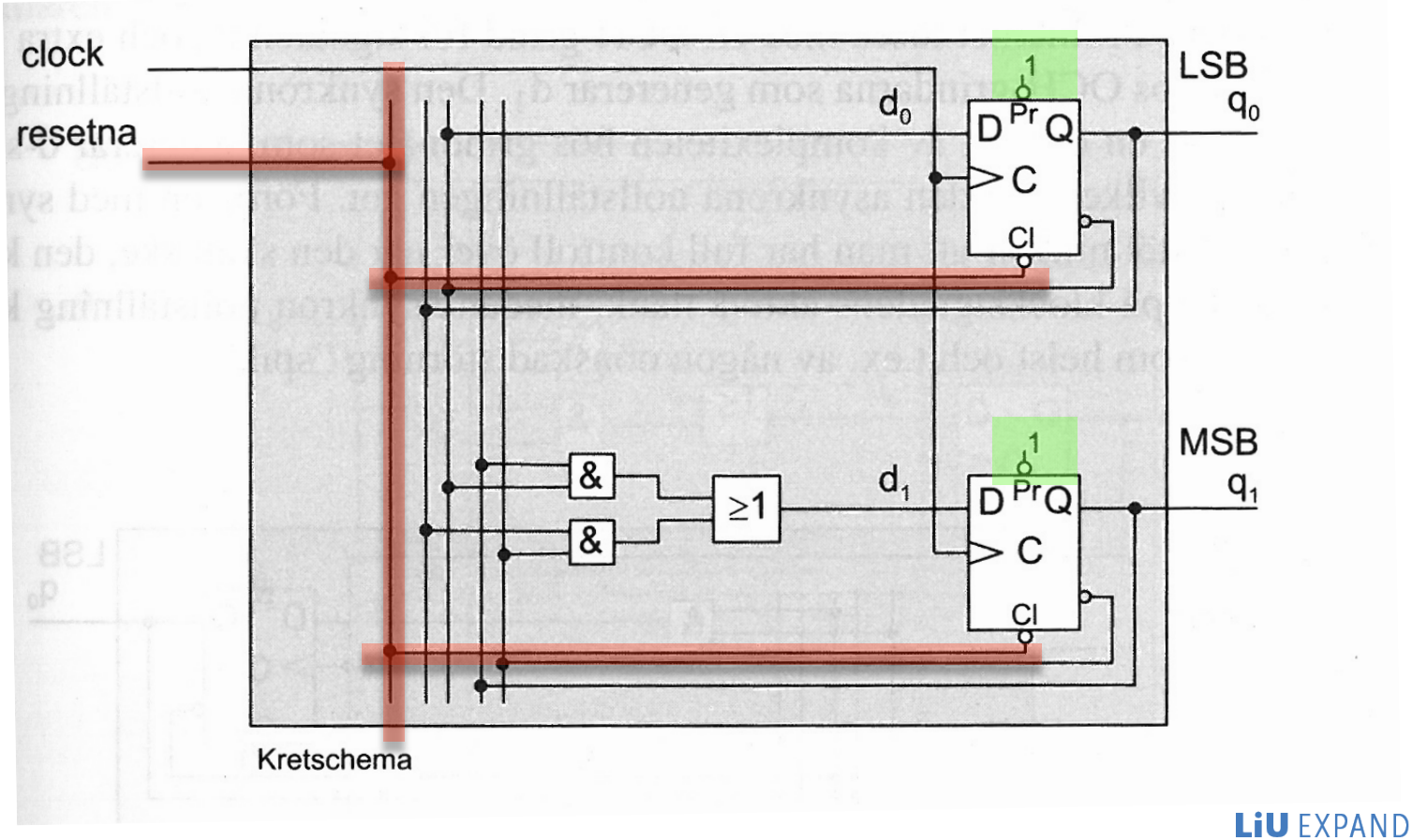
Tillståndsdiagram

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

Tillståndstabell

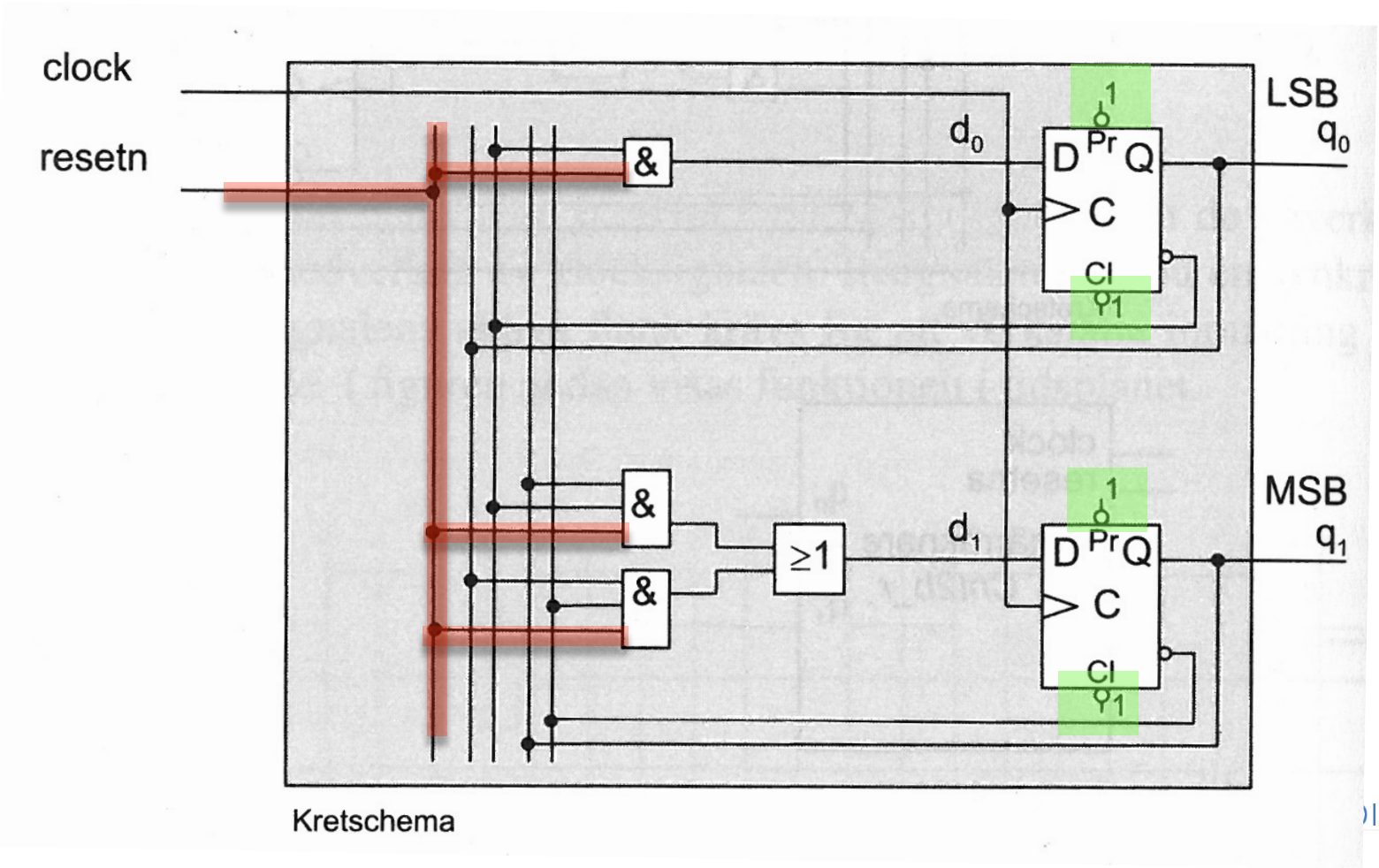
Asynkron nollställning

- Nollställning sker med resetna = 0, utan medverkan av klockan.



Synkron nollställning

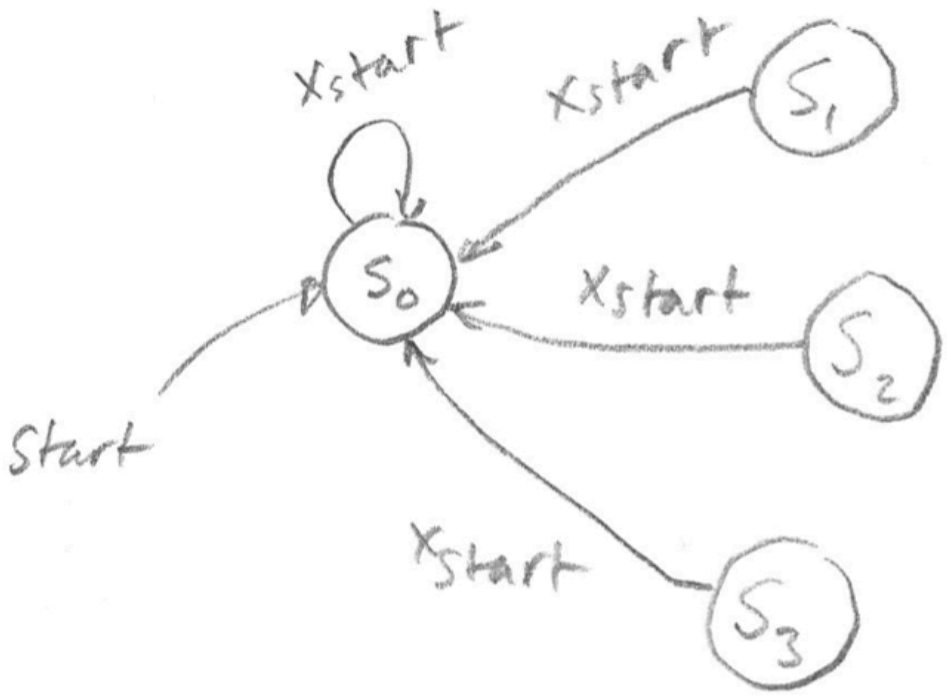
- Synkron nollställning aktiverad med resetn = 0



Kretschema

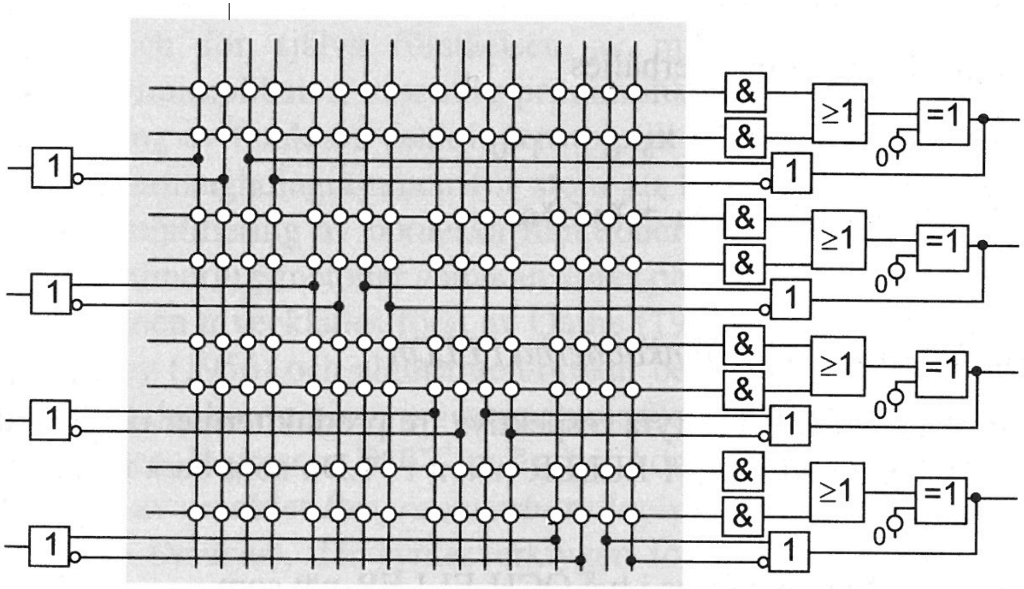
Synkron nollställning

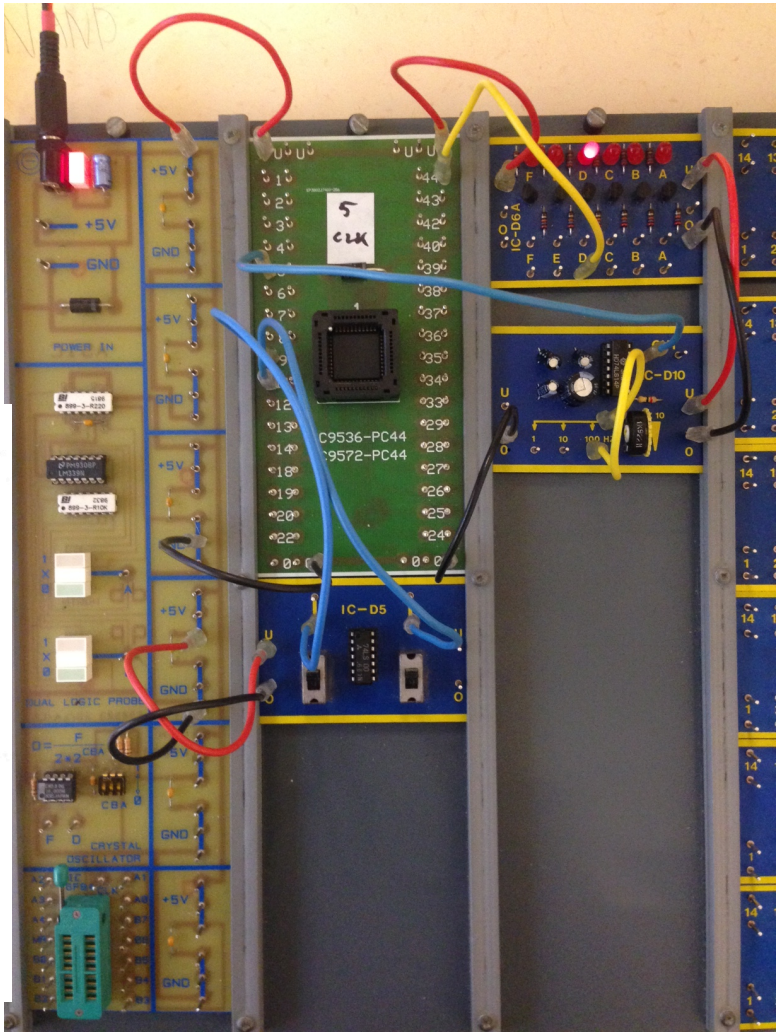
- En insignalkombination sätter nästa tillstånd till starttillståndet oberoende av nuvarande tillstånd.
  - Hanteras därmed som vilken insignal som helst



Programmerbar logik

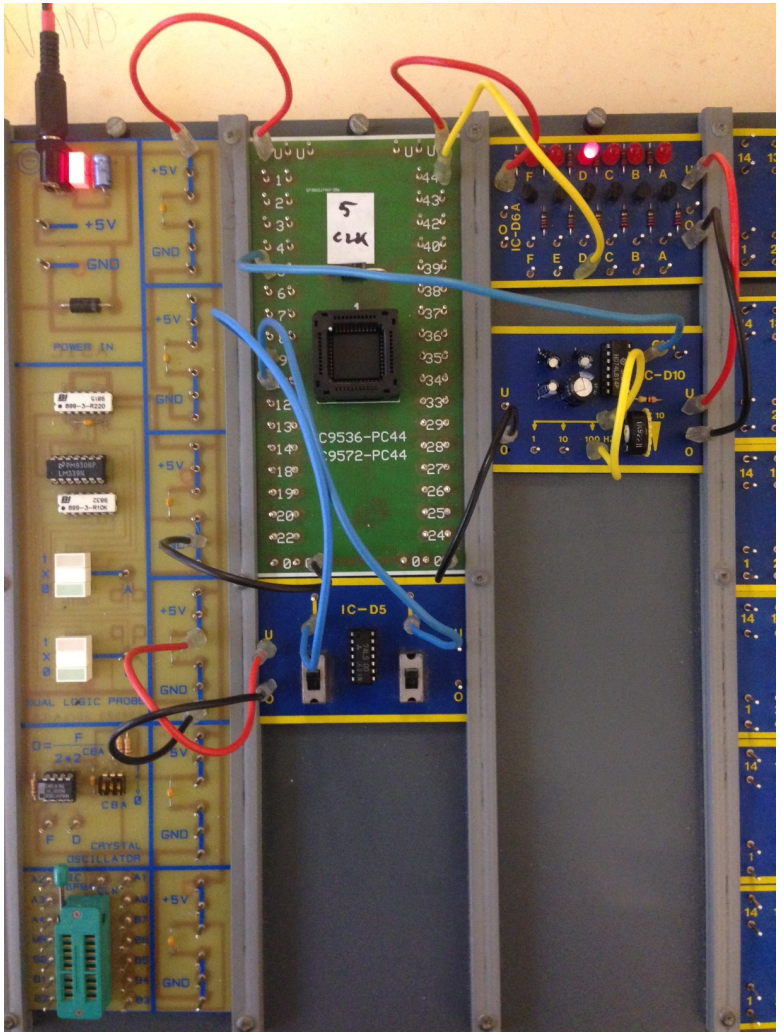
- Complex programmable logic device (CPLD)
- Fix komponentuppsättning
- Programmerar vilka kopplingar som ska finnas





Konstruktion med CPLD

- Skriv beskrivning i VHDL
  (**V**ery high speed integrated circuit **H**ardware **D**escription **L**anguage)
- 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

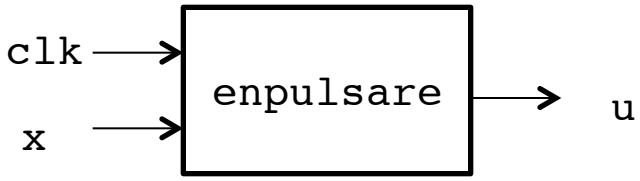


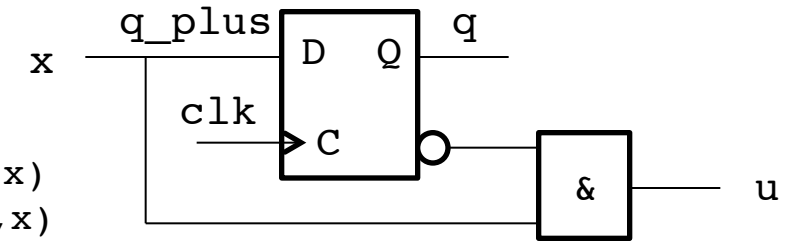
VHDL-exempel - enpulsaren

```
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;                -- q+ = f(q,x)
  u <= (not q) and x;         -- u  = g(q,x)
end ekvationer;
```

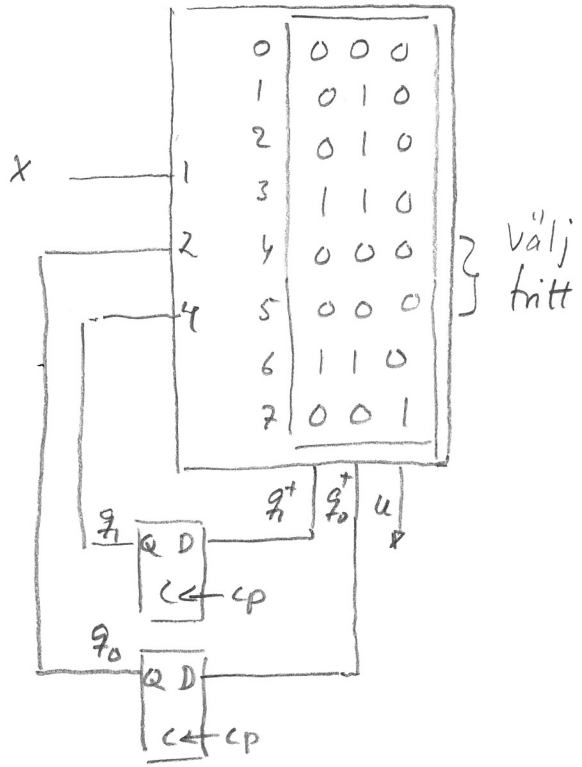




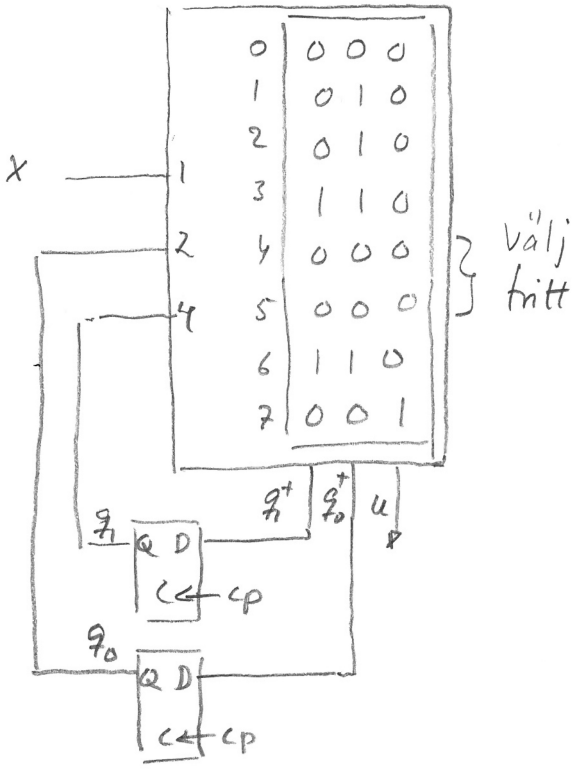
Detektion av var tredje etta i VHDL

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

adress — utsignal



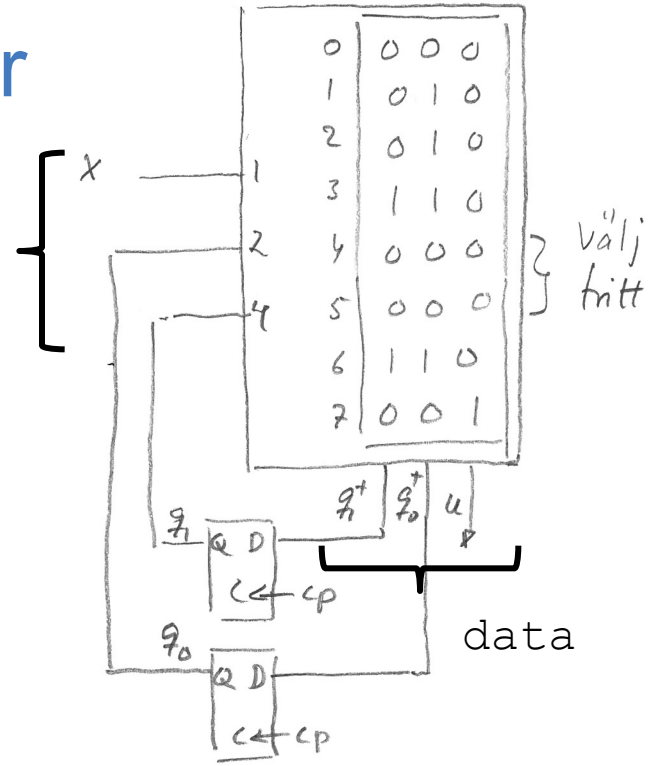
Gränssnitt mot omgivningen




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

Interna signaler

adress



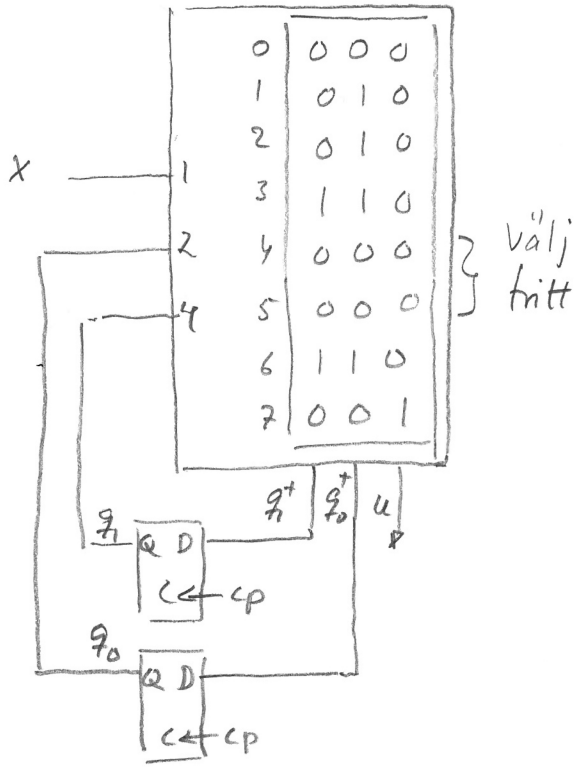

```
adress = (q1,q0,x)
data   = (q1_plus,q0_plus,u)
```

```
architecture behavior of skrets is
  signal q0, q0_plus: std_logic;
  signal q1, q1_plus: std_logic;
  signal adress : 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;
```

Vippor

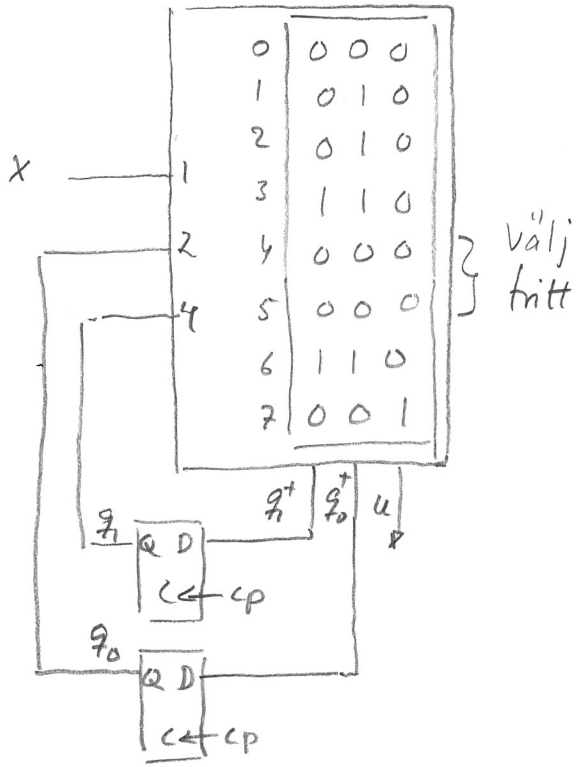
```
-- vippor
process(clk)
begin
  if rising_edge(clk) then
      q0 <= q0_plus;
      q1 <= q1_plus;
  end if;
end process;
```



Minnet

```
-- ROM
adress <= q1 & q0 & x; -- concatenering
with adress 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);
```

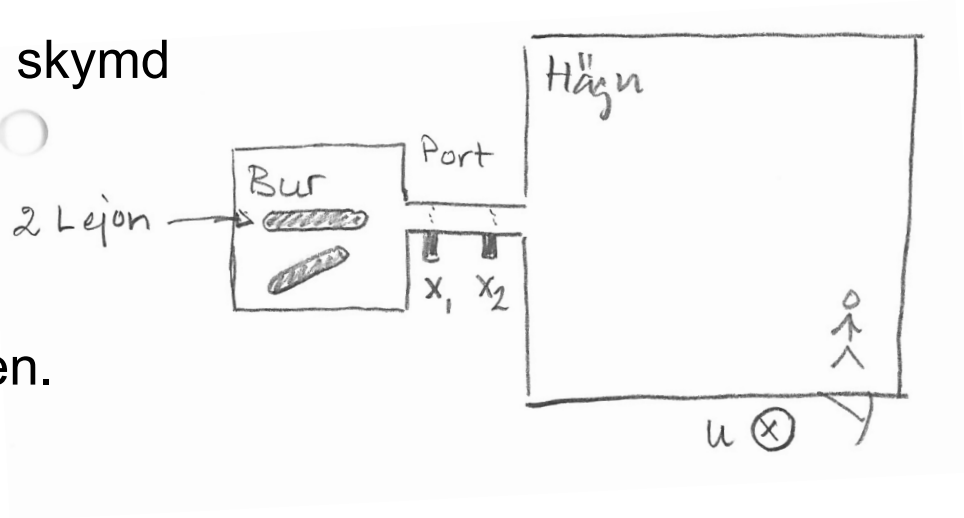


Dagens föreläsning

- 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: $x_i = \begin{cases} 1 & \text{fotocellen skymd} \\ 0 & \text{annars} \end{cases}$
- Lampa: $u = \begin{cases} 0 & \text{släkt} \\ 1 & \text{tänd} \end{cases}$
- Vid start är båda lejonen i buren.
- Lejonen:
  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

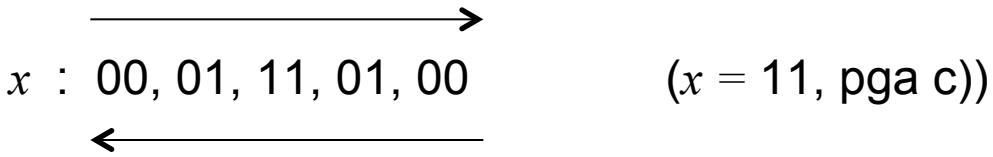
- Porten tom: $x = (x_1, x_2) = 00$
- Lejon i port:

  a) => ett lejon

  b) => rör sig genom porten

  Fall

  - Lejon ut ur bur:



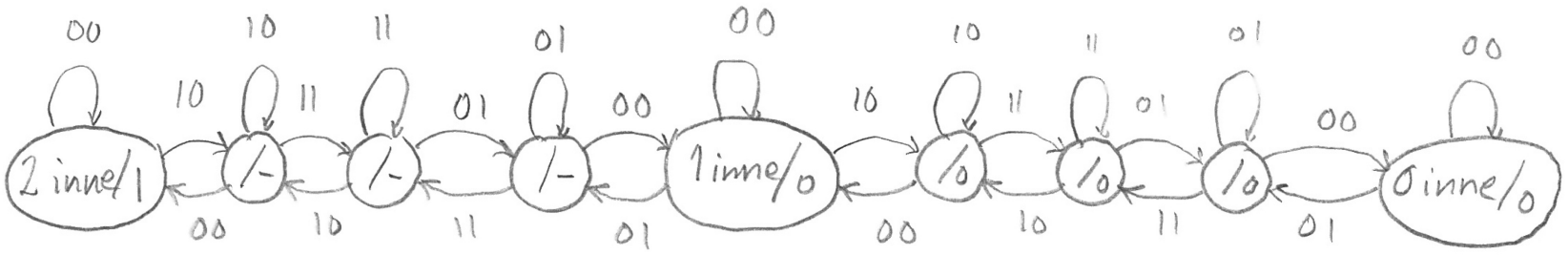
  - Lejon in i bur:

  d) => Varje insignalkombination upprepas

  => 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.

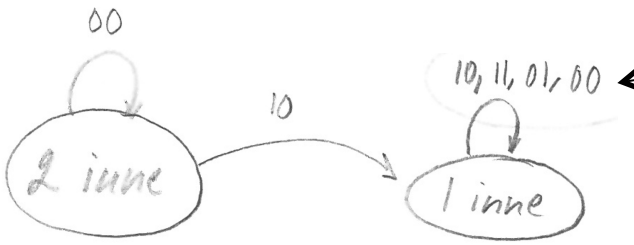


- Onödigt många tillstånd: 9 st
- Det finns algoritmer som minimerar antalet tillstånd.
- Hanterar även att lejon backar i porten

  => Onödigt komplicerat tillståndsdiagram

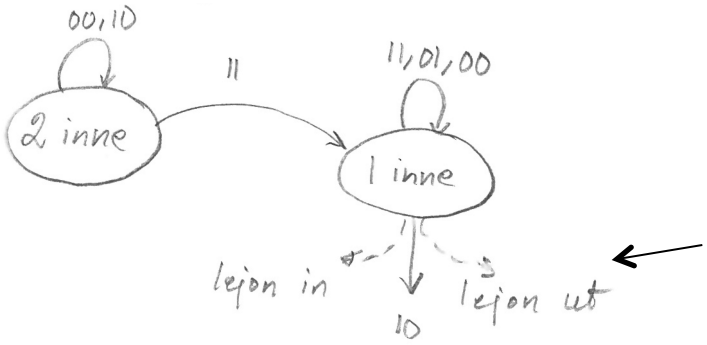
Lösning med få tillstånd

- 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



alla insignalkombinationer => fast i tillståndet

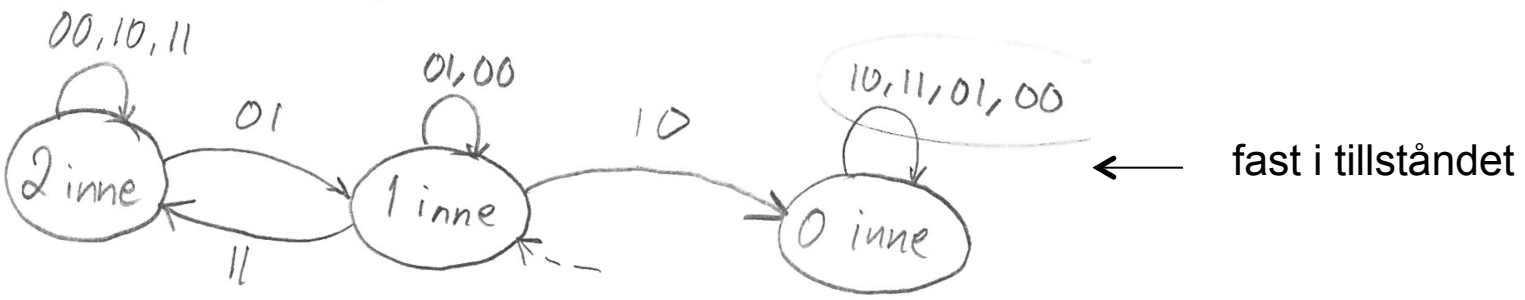
- Detektera utpassage med 11



Går ej att skilja på fallen

Lejonbur fortsättning …

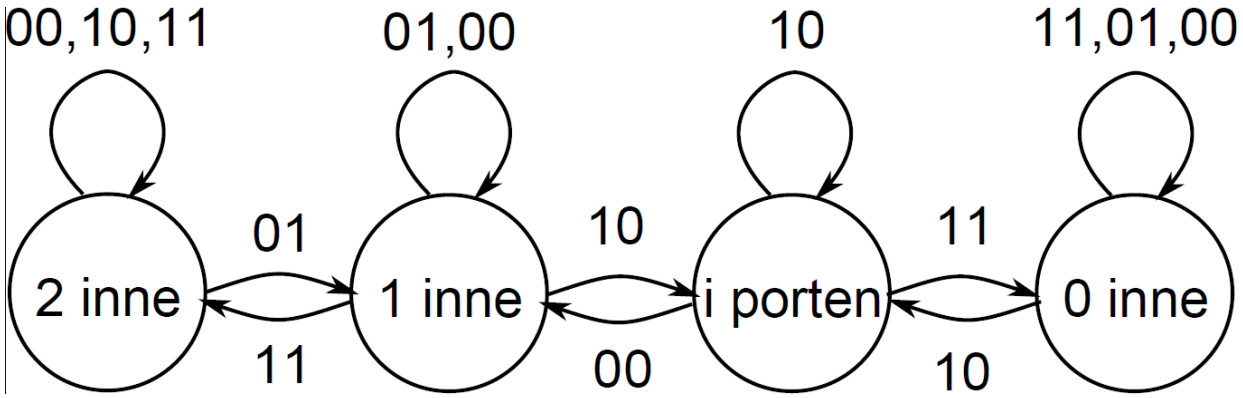
- Detektera utpassage med 01



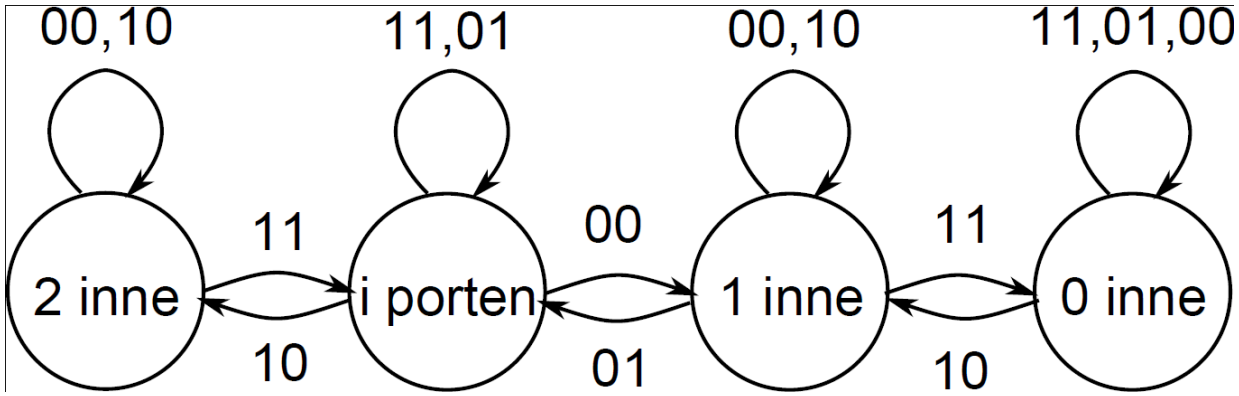
- 10, 11 kvar
  - 11 påträffas först vid inpassage
  - 10 påträffas först vid utpassage
- Vilka insignaler för sträckad tillståndsövergång är tänkbar?
  - 01: funkar ej ty vi hamnar i “2 inne”
  - 11: funkar ej ty vi hamnar i “2 inne”
  - 10: funkar ej ty vi hamnar i “0 inne”
  - 00: OK, men nytt tillstånd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

Tillståndsdiagram för lejonburen

- Tillståndsdiagram framtaget på föreläsningen:



- Alternativt tillståndsdiagram:



Linköpings universitet

expanding reality

www.liu.se