Digitala system EITA15

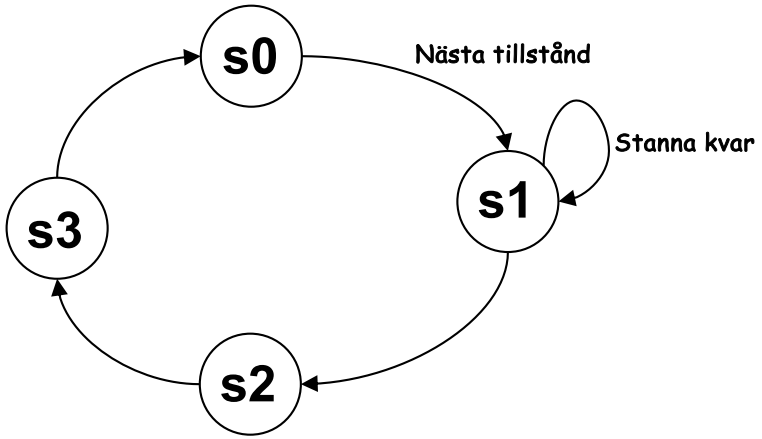
Elektro- och informationsteknik

sekvensnät

Sekvensnät

- En klass av kretsar med minnesfunktion
- För att kunna beskriv utsignalerna krävs kännedom om systemets tillstånd och beteende.
- Tillståndsdiagram, tillståndstabell , klocksignal och register
- D-vippor
- Sekvensmodellerna Moore och Mealy.
- Synkrona och asynkrona sekvensnät
- Finite State Machines (FSM)

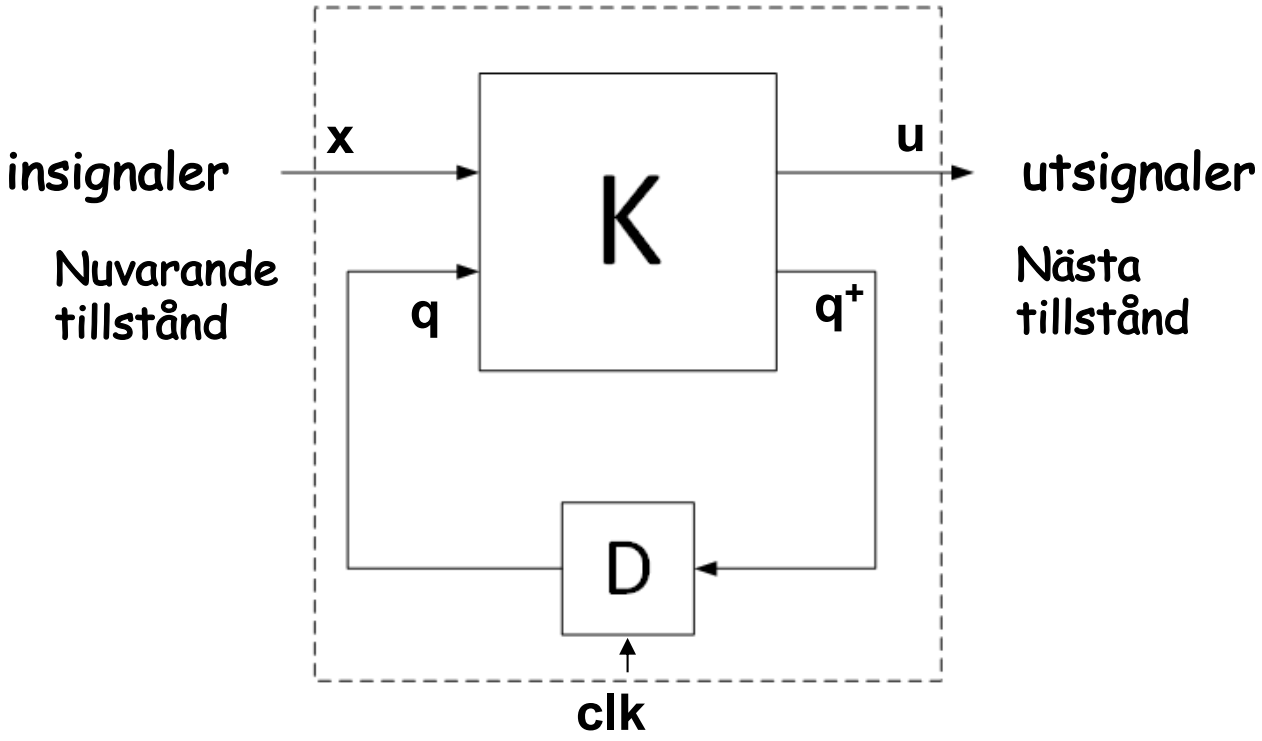
Tillståndsdiagram



Realisering av sekvenskretsar :

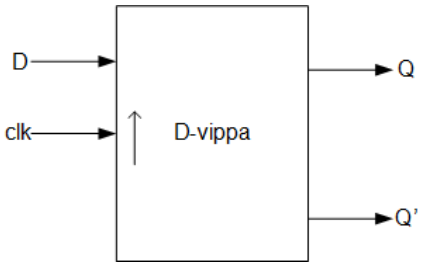
- börjar med att **beskriva beteendet med tillstånd**
- Tillstånd Sx betecknas med en cirkel
- Börja med ett **starttillstånd**
- Gå till nästa tillstånd eller stanna kvar **beroende på insignalen**
- Generera **utsignaler i tillstånden (Moore)**
- Generera **utsignaler vid övergångarna (Mealy)**

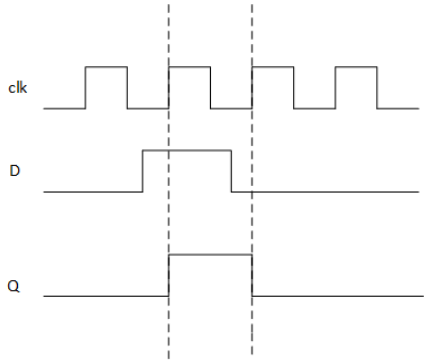
Beteendet av tillståndsdiagrammet kan realiseras av en sekvenskrets



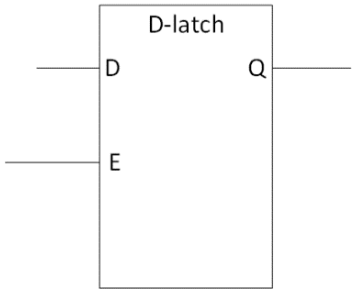
D = D-vippan, realiserar minnet av systemet (tillstånd)
K = Kombinatoriska nätet (Booleska funktioner)

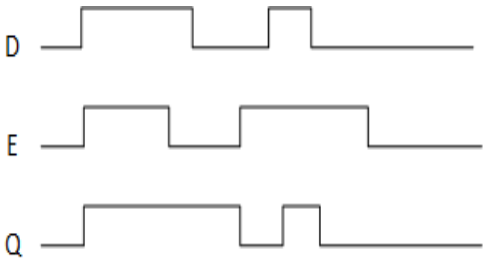
D-vippa vs D-latch





clk	D	Q	Q'
L	X	Q	Q'
clk	L	L	H
clk	H	H	L





E	D	Q
L	X	Q
H	L	L
H	H	H

Tillståndstabell

- Innehåller ingen ny information med avseende på tillståndsgrafen

Nuvarande tillstånd	Nästa tillstånd		Utsignal
	Insignal x		u
	0	1	
start	start	nästa	0
nästa	nästa	nästa2	0
nästa1	nästa2	start	0
nästa2	nästa1	start	1

Tillståndskodning

Kodning (0, 1)

	q_1q_0
start	0 0
Nästa	0 1
Nästa1	1 0
Nästa2	1 1

Nuvarande tillstånd	Nästa tillstånd		Utsignal
	Insignal x		u
	0	1	
00	00	01	0
01	01	11	0
10	11	00	0
11	10	00	1

Ta fram booleska uttryck för q_1^+ , q_0^+ och u (enligt tidigare)

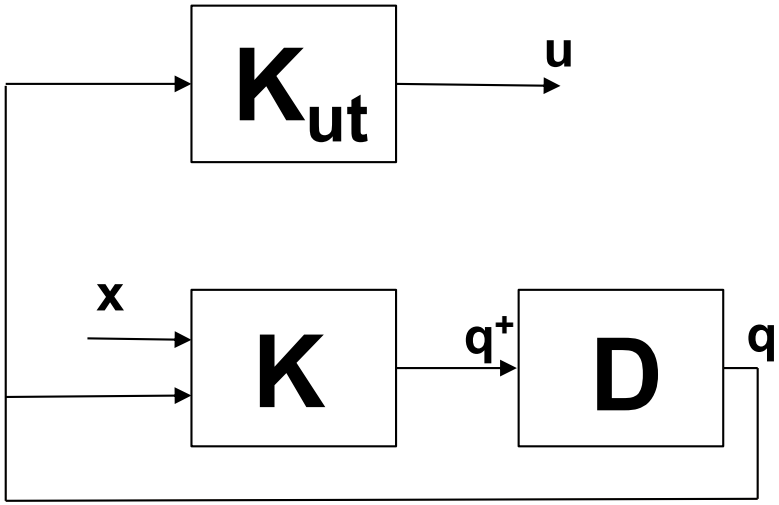
Gray kodning och one-hot, andra typer av tillståndskodning

Definitioner

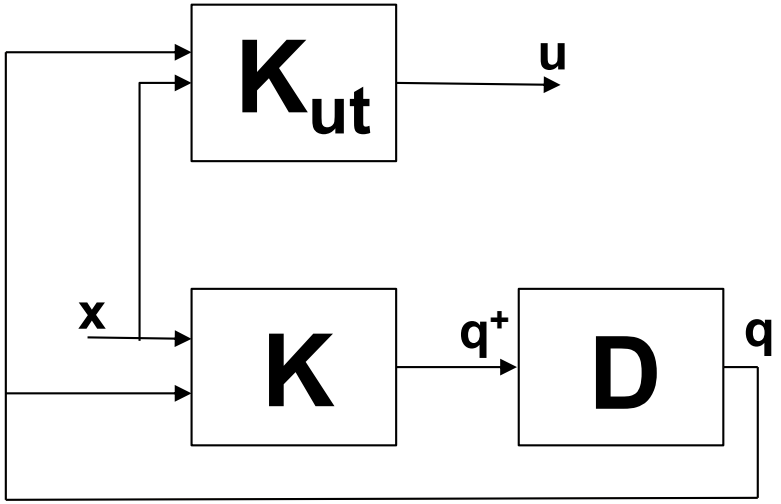
- En tillståndsmaskin har insignal **x**, utsignal **u**, ett nuvarande tillstånd **q** och nästa tillstånd q^+
- En **Mealymaskin** är en tillståndsmaskin där tillstånd och utsignal beror på både nuvarande tillstånd **q** och insignal **x**
- En **Mooremaskin** är en tillståndsmaskin där utsignalen beror enbart på nuvarande tillstånd **q**

Mealy eller Moore

Moore



Mealy



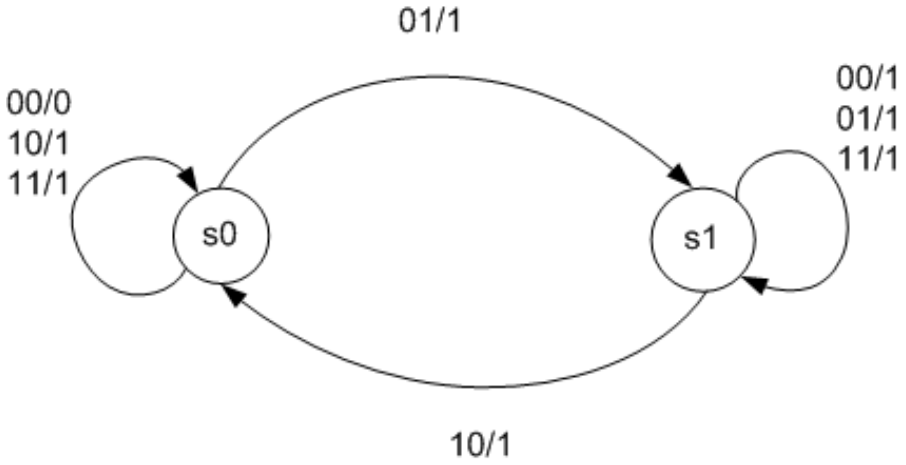
Utsignal:

- beror endast av nuvarande tillstånd.
- ändrar i nästa tillstånd

Utsignalen :

- beror både av nuvarande tillståndet och insignalen
- ändrar vid övergång till nästa tillstånd

Lejonbur (igen)

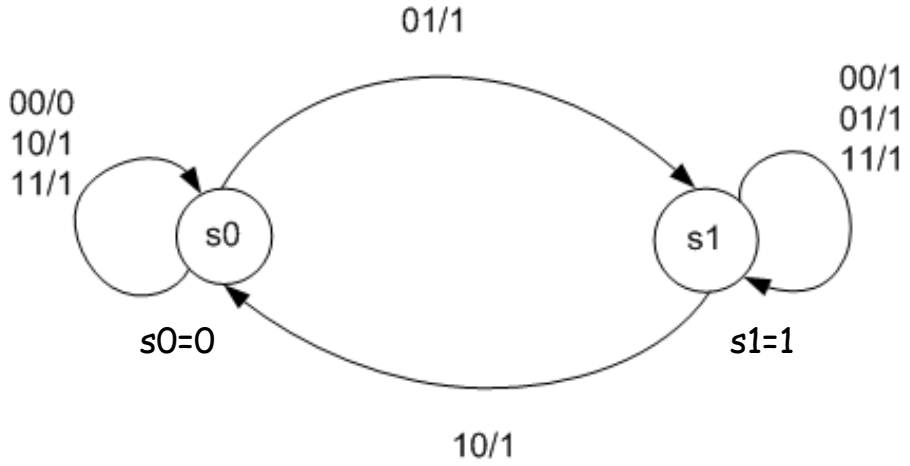


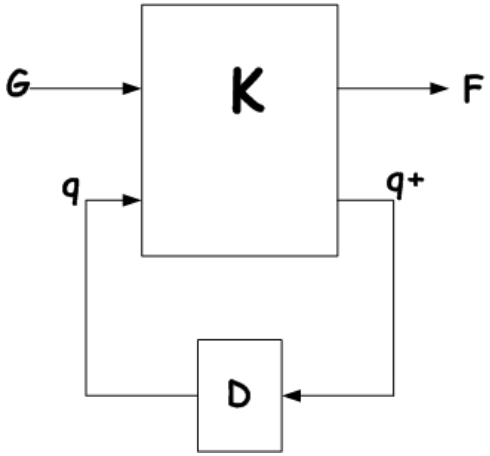
Mealy

00 -> 10 -> 11 -> 01 -> 00 ; ute

00 -> 01 -> 11 -> 10 -> 00 ; inne

Lejonbur (igen)

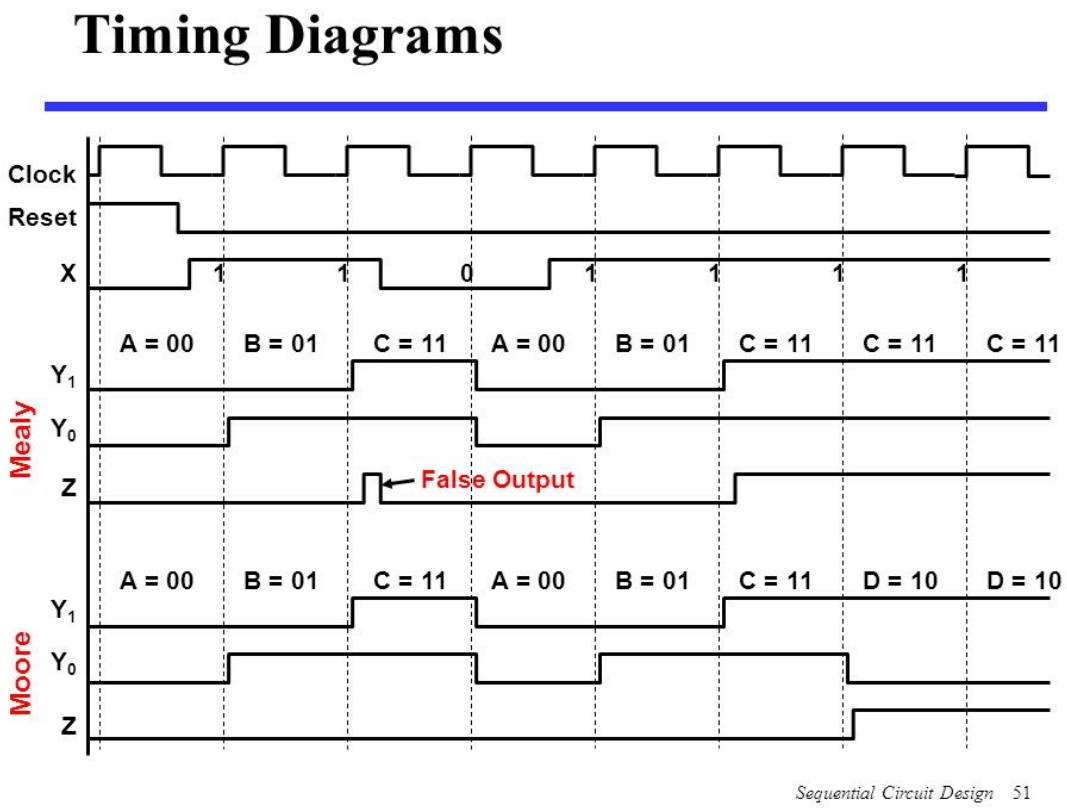




q nuvarande tillstånd
q+ nästa tillstånd
F fara
g1 och g2 givare

g1	g2	q	q+	F
0	0	0	0	0
0	1	0	1	1
1	0	0	0	1
1	1	0	0	1
0	0	1	1	1
0	1	1	1	1
1	0	1	0	1
1	1	1	1	1

Mealy vs Moore



Mealy har färre tillstånd
Moore är säkrare, utgång ändras vid klockflank
Mealy snabbare

Mealy -> Moore

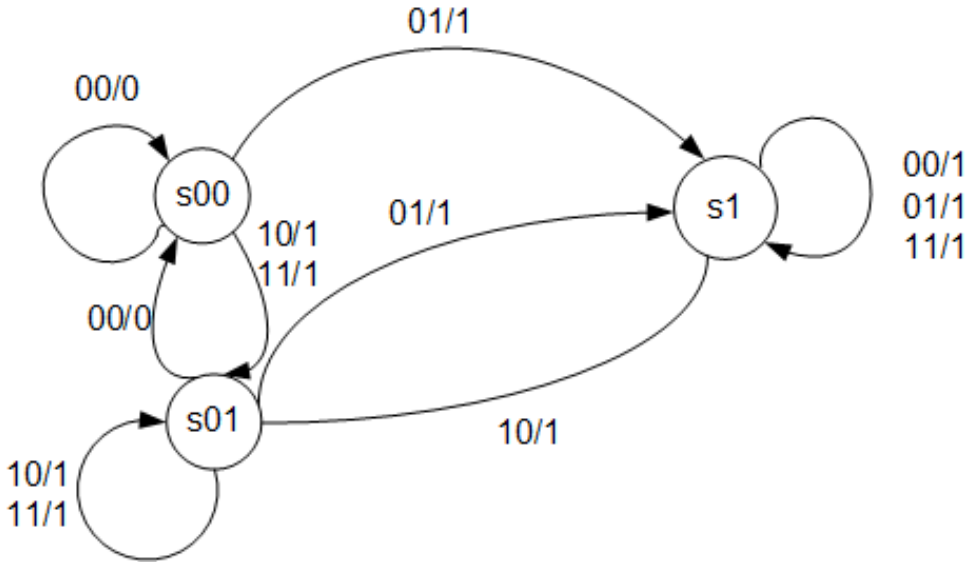
- Alla Mealymaskiner kan översättas till Mooremaskiner om vi accepterar att:

- Utsignaler är fördröjda ett steg.
- Fler tillstånds behövs.

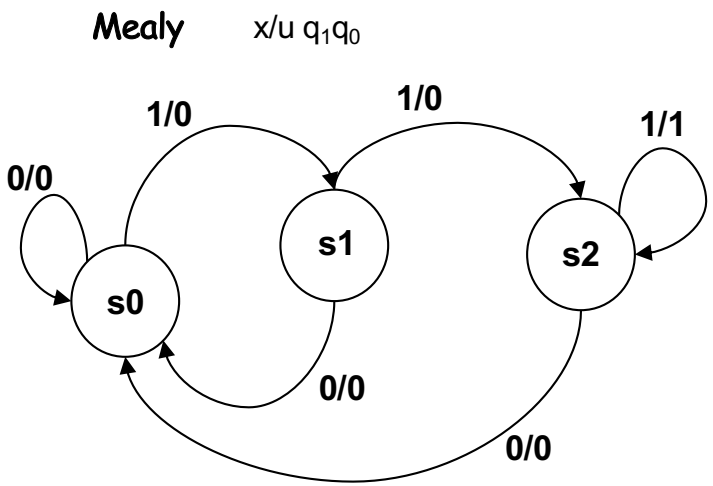
- Med andra ord,

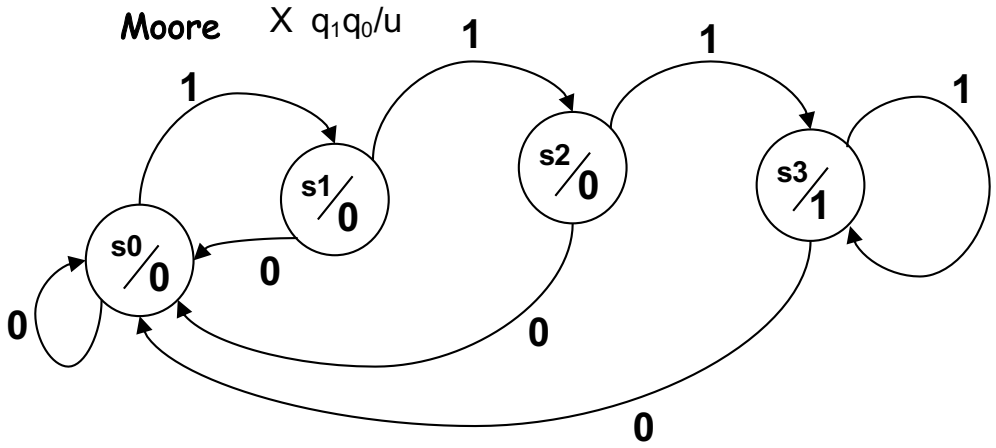
- Dela upp tillstånd så att alla ingående bågar har samma utsignal.
- Flytta utsignal från båge till tillståndet bågen pekar.

Mealy -> Moore exempel



Exempel: detektera minst 3 ettor.

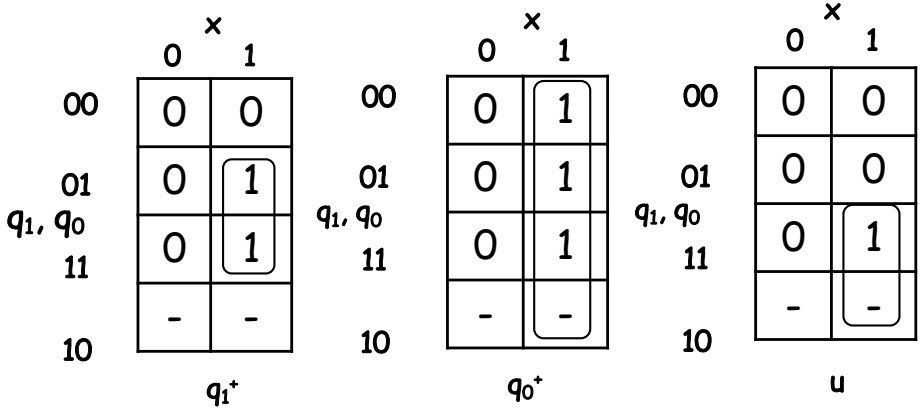




Nuvarande tillstånd	Nästa tillstånd / u	
	Insignal x	
	0	1
00 s0	00 s0/0	01 s1/0
01 s1	00 s0/0	11 s2/0
11 s2	00 s0/0	11 s2/1
--	--	--

Nuvarande tillstånd	Nästa tillstånd		Utsignal
	Insignal x		u
	0	1	
00 s0	00 s0	01 s1	0
01 s1	00 s0	11 s2	0
11 s2	00 s0	10 s3	0
10 s3	00 s0	10 s3	1

x	q_1	q_0	q_1^+	q_0^+	u
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	-	-	-
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	-	-	-
1	1	1	1	1	1



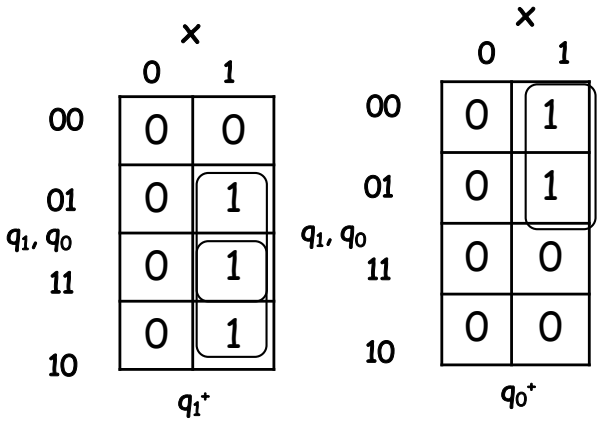
$q_1^+ = x\, q_0$

$q_0^+ = x$

$u = x\, q_1$

x	q_1	q_0	q_1^+	q_0^+
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

	u
00	0
01	0
11	0
10	1



$q_1^+ = x\, q_0' + x\, q_1$

$q_0^+ = x\, q_1'$

$u = q_1\, q_0'$

Tillståndskodning

- One_hot; Kodorden innehåller endast en etta, detta medför att antalet bitar i kodordet blir = antalet tillstånd.
- Binär kodning; Kodorden kodas enligt BCD-kodning 000,001..
- Gray kodning; Kodorden kodas enligt Gray kodning. Endast en bit får ändras mellan varje kodord. Ex. 00,01,11,10
- Tanke; försök hitta en kodning där så få variabler ändras mellan de olika tillstånden.

VHDL Beskrivning av D-vippa

```
----------------------------------
-- D-vippa
----------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity d_vippa is
   Port (
          d_in, clock : in STD_LOGIC;
          q_ut        : out STD_LOGIC
         );
end d_vippa;

architecture Behavioral of d_vippa is

begin
   process(clock)
   begin
      if rising_edge(clock) then
         q_ut <= d_in;
      end if;
   end process;

end Behavioral;
```

annat sätt

```
.
.
process (clock)
begin
   if (clock'event and clock = '1') then
      q_ut <= d_in;
   end if;
end process;
.
.
```


VHDL Beskrivning av D-latch

```
-------------------------------------
-- D-latch
-------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity D_Latch is
port (
          enable, d : in std_logic;
          q              : out std_logic
        );
end entity;

architecture Behavioral of D_latch is
begin
    process (enable)
    begin
        if enable = '1' then
            q <= d;
        end if;
     end process;
end Behavioral;
```


Asynkron vs synkron

Asynkron reset

```
.
.
process (clk, reset)
begin
    if (reset='0') then
        reg <= '0';
    elsif (clk'event and clk = '1') then
        reg_q <= input;
     end if;
end process;
.
.
```

Synkron reset

```
.
.
process(clk)
begin
    if (clk'event and clk = '1') then
        if (reset='0') then
            reg <= '0';
        else
            reg_q <= input;
        end if;
     end if;
end process;
.
.
```


Asynkrona sekvenskretsar och signaler

- Ett asynkront sekvensnät är ett sekvensnät som saknar yttre synkroniseringssignaler och minneselement
- Byt ut d-elementet i synkront sekvensnät mot fördröjningselement
- I ett asynkront sekvensnät kan varje insignalsförändring orsaka byte av tillstånd

För asynkron sekvenskrets gäller

- **Asynkront realiserbart**: stabila tillstånd för alla insignalskombinationer
- **Kapplöpningsfritt**: koda tillstånden så att endast en signal åt gången får styra tillståndsändingar
- **Hasardfritt**: booleska funktionerna realiseras med samtliga primimplikatorer