Digitalna vezja
UL, FRI

Vaja 11, Avtomati

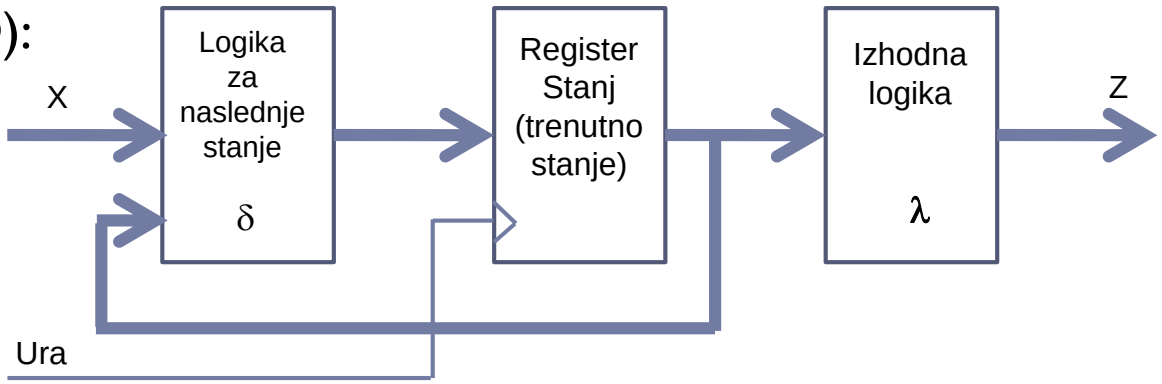
Avtomat – Končni stroj stanj

FSM (Finite state machine): $A = \{X, S, Z, \delta, \lambda\}$

Moorov avtomat (MO):

$\delta: X \times S \rightarrow S$

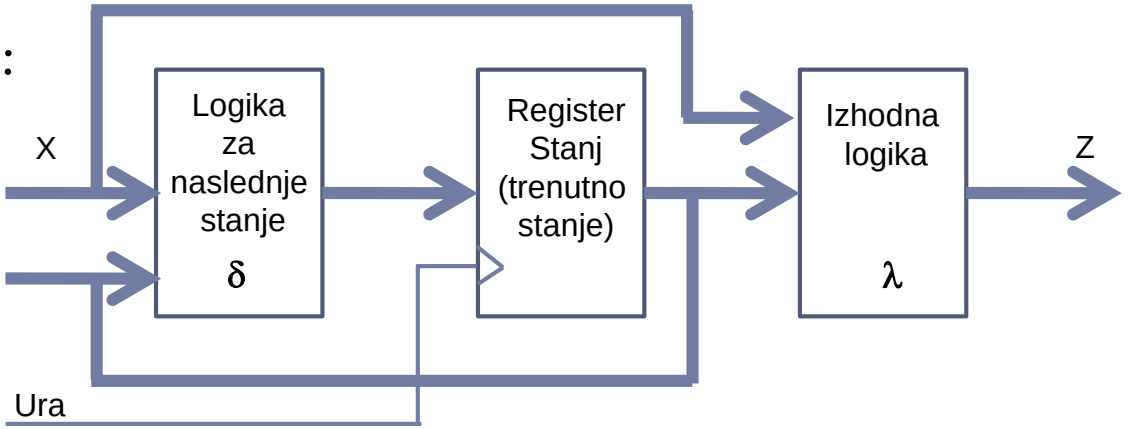
$\lambda: S \rightarrow Z$



Mealyjev avtomat (ME):

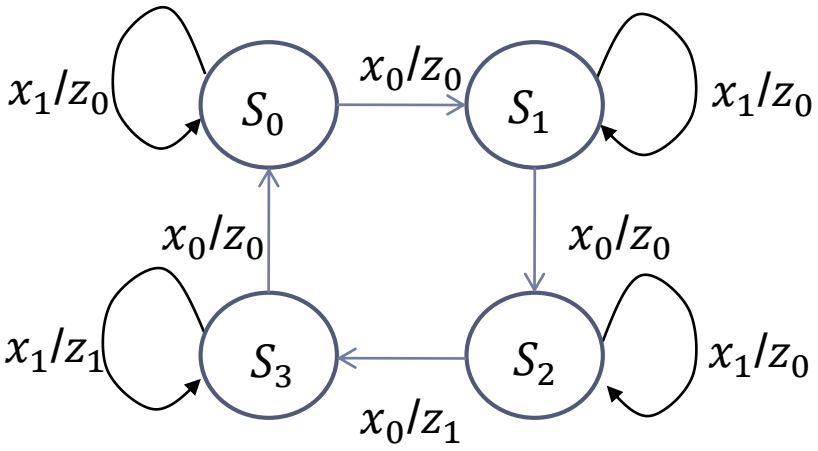
$\delta: X \times S \rightarrow S$

$\lambda: X \times S \rightarrow Z$



Primer 1: Pretvorba med avtomati: Mealy → Moore

1) Diagram prehajanja stanj - Mealy



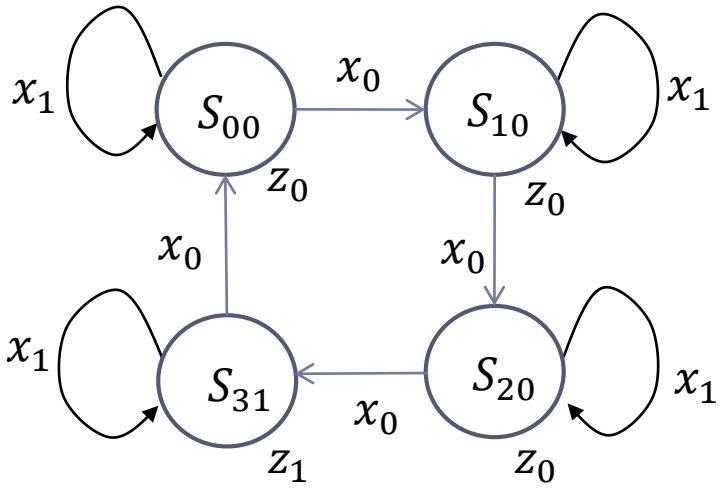
2) Tabela prehajanja stanj - Mealy

	S_0	S_1	S_2	S_3
x_0	S_1/z_0	S_2/z_0	S_3/z_1	S_0/z_0
x_1	S_0/z_0	S_1/z_0	S_2/z_0	S_3/z_1

3) Tabela prehajanja stanj - Moore

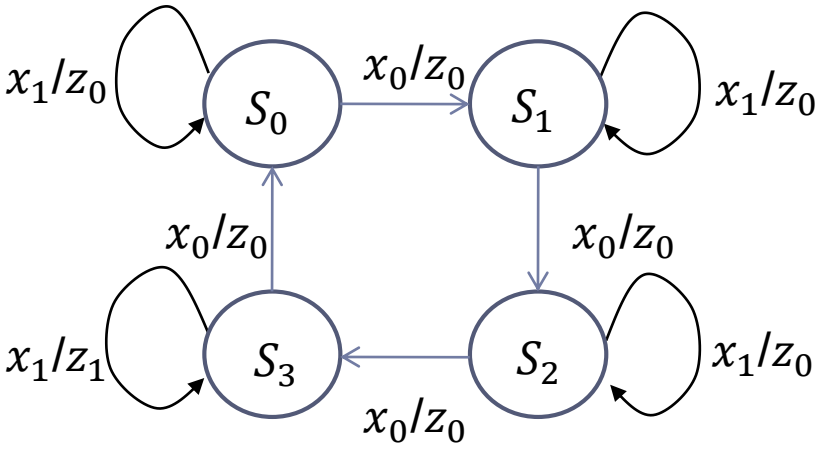
	z_0	z_0	z_0	z_1
	S_{00}	S_{10}	S_{20}	S_{31}
x_0	S_{10}	S_{20}	S_{31}	S_{00}
x_1	S_{00}	S_{10}	S_{20}	S_{31}

4) Diagram prehajanja stanj - Moore



Primer 2: Pretvorba med avtomati: Mealy → Moore

1) Diagram prehajanja stanj - Mealy



2) Tabela prehajanja stanj - Mealy

	S_0	S_1	S_2	S_3
x_0	S_1/z_0	S_2/z_0	S_3/z_0	S_0/z_0
x_1	S_0/z_0	S_1/z_0	S_2/z_0	S_3/z_1

Naslednja stanja Mealyjevega avtomata:

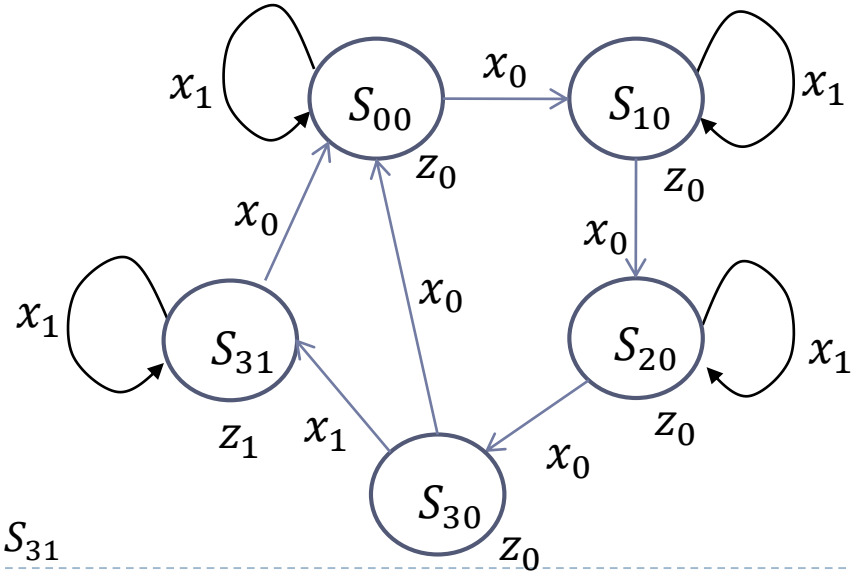
S_0/z_0,S_1/z_0, S_2/z_0, S_3/z_0, S_3/z_1,

določajo Moorov avtomat: S_{00}, S_{10}, S_{20}, S_{30}, S_{31}

3) Tabela prehajanja stanj - Moore

	z_0	z_0	z_0	z_0	z_1
	S_{00}	S_{10}	S_{20}	S_{30}	S_{31}
x_0	S_{10}	S_{20}	S_{30}	S_{00}	S_{00}
x_1	S_{00}	S_{10}	S_{20}	S_{31}	S_{31}

4) Diagram prehajanja stanj - Moore



Pretvorba Moore → Mealy

Vzamemo Moorov avtomat iz prejšnje pretvorbe:

1) Tabela prehajanja stanj - Moore

	z_0	z_0	z_0	z_0	z_1
	S_{00}	S_{10}	S_{20}	S_{30}	S_{31}
x_0	S_{10}	S_{20}	S_{30}	S_{00}	S_{00}
x_1	S_{00}	S_{10}	S_{20}	S_{31}	S_{31}

2) Tabela prehajanja stanj - Mealy

	S_{00}	S_{10}	S_{20}	S_{30}	S_{31}
x_0	S_{10}/z_0	S_{20}/z_0	S_{30}/z_0	S_{00}/z_0	S_{00}/z_0
x_1	S_{00}/z_0	S_{10}/z_0	S_{20}/z_0	S_{31}/z_1	S_{31}/z_1

3) Končna pretvorba - Mealy

	S_0	S_1	S_2	S_3
x_0	S_1/z_0	S_2/z_0	S_3/z_0	S_0/z_0
x_1	S_0/z_0	S_1/z_0	S_2/z_0	S_3/z_1

Rešitev je Mealyjev avtomat iz prejšnje pretvorbe.

Enaki prehodi stanj, enaki izhodi pri posamezni vhodni črki
→ združimo stanji $S_{30} = S_{31}$ in vsa stanja preimenujemo v :

$S_{00} \rightarrow S_0, S_{10} \rightarrow S_1, S_{20} \rightarrow S_2, S_{30} = S_{31} \rightarrow S_3$

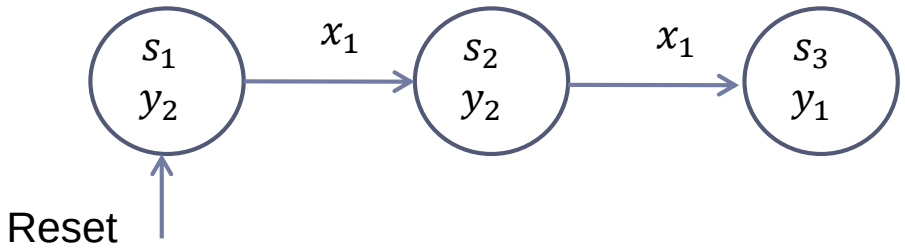
Primeri avtomatov

N1: Razpoznavanje niza dveh črk 'x_1x_1'

Moore-ov avtomat je podan z vhodno množico $X = (x_1, x_2)$ in izhodno množico $Y = (y_1, y_2)$. Izhod naj bo y_1, če je na vhodu zaporedje x_1x_1, sicer pa je izhod y_2.

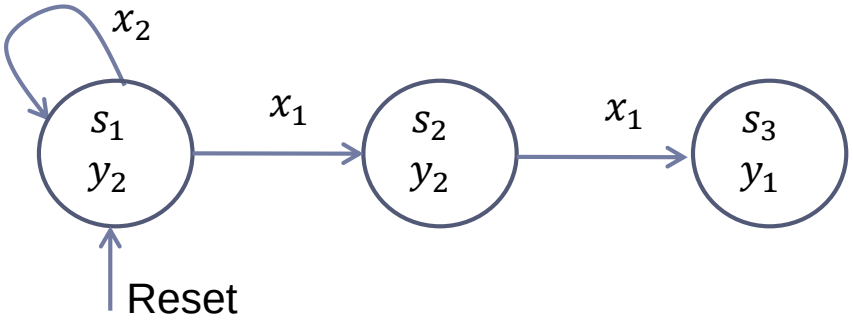
- Definirajte diagram prehajanja stanj (DPS).
- Primer zaporedja vhodov in izhodov:
  - Vhod: $x_1x_2x_2x_1x_1x_1x_2x_1x_2x_1x_1x_1x_1x_1 \ldots$
  - Izhod: $y_2y_2y_2y_2y_1y_1y_2y_2y_2y_2y_1y_1y_1y_1 \ldots$

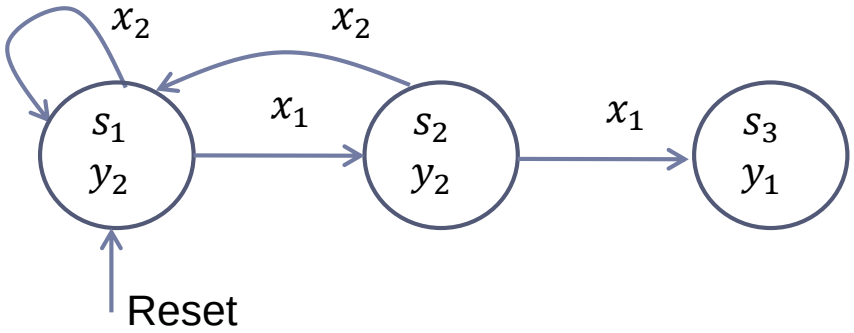
1. Začetno stanje označimo z s_1 in določimo izhod y_2. Temu sledi zaporedje stanj z izhodom y_2 in prehodov, ki vodijo v stanje z izhodom y_1.

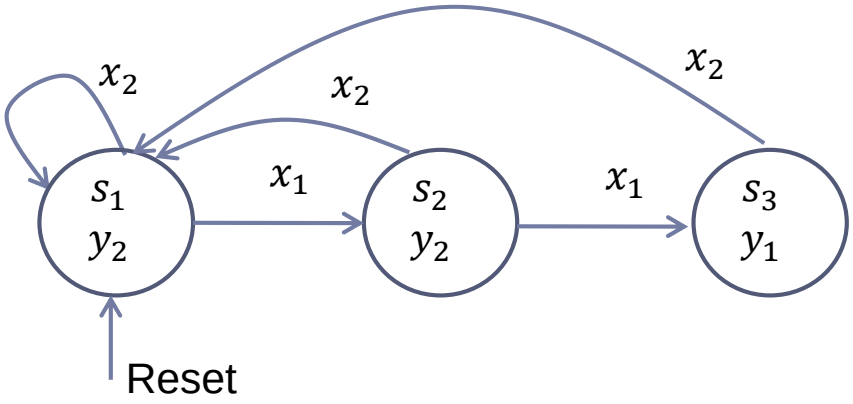


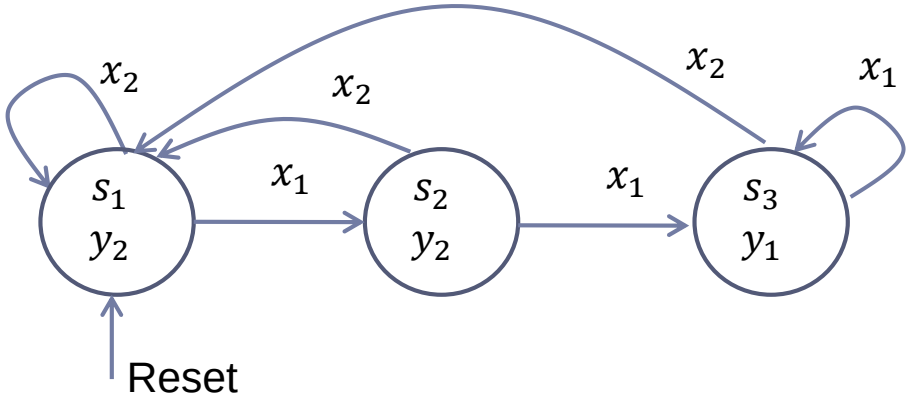
2. Za vsa stanja v nadaljevanju določimo:

- prehode iz stanj s_1, s_2, s_3 ob pojavu vhoda x_2 in
- prehod iz s_3 ob pojavu vhoda x_1.



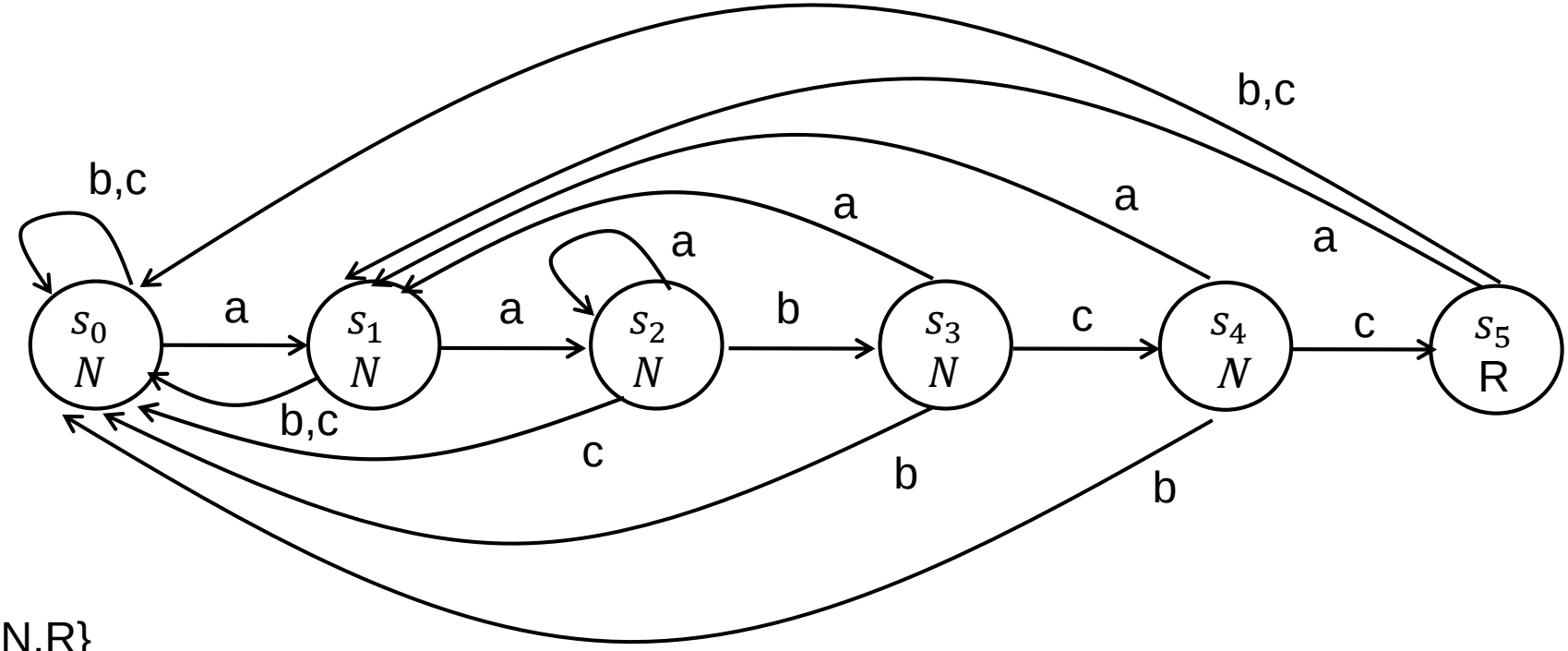






N2: Razpoznavanje niza *'aabcc'*

Podana je vhodna abeceda X = {a, b, c}. Avtomat daje na izhodu črko R, če je v zaporedju vhodov iz množice X razpoznan niz ***aabcc*** in vrne na izhodu črko N v vseh ostalih primerih. Začetno stanje je s_0, vsa nadaljnja stanja pa morajo biti označena z naraščajočimi vrednostmi indeksov od 1 naprej. Zapišite množico stanj S in množico izhodov Y za Moorov avtomat in narišite diagram prehajanja stanj.



Y = {N,R}

S = {s_0, s_1, s_2, s_3, s_4, s_5}

Izvedba avtomatov

N3: Izvedba avtomata za razpoznavanje niza črk ‘baa‘

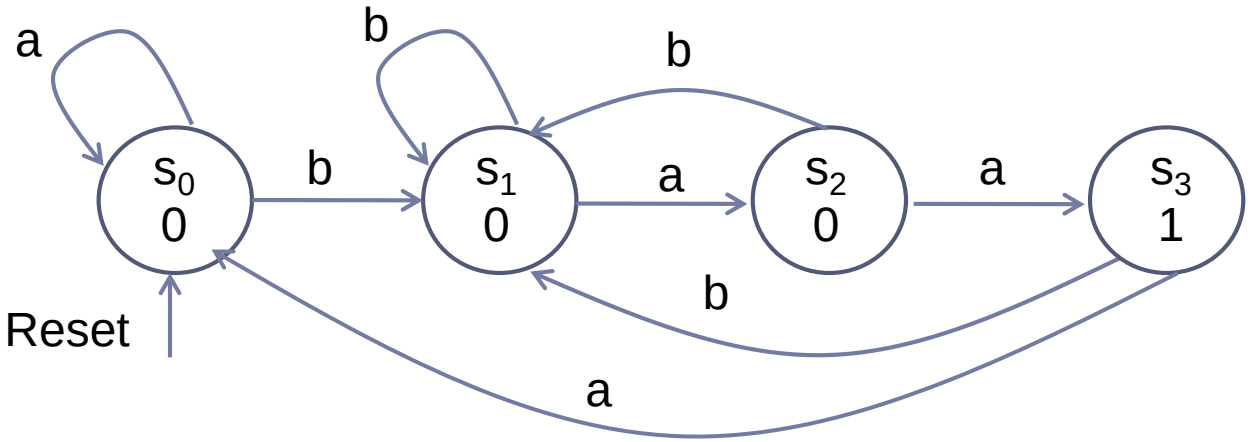
- Realizirajte Mooreov avtomat za razpoznavanje niza črk s podano vhodno množico X={a,b} in izhodno množico Y={0,1}.
- Izhod y je enak 1, če smo v vhodnem zaporedju črk na tipkovnici razpoznali niz ‘baa‘ (prva črka je b), sicer je izhod y enak 0.
- Za vpis niza črk v logisimu uporabimo tipkovnico, na izhodu tipkovnice je potrebno vsako črko kodirati oz. pretvoriti v binarno vrednost (a=1,b=0) in jo posredovati v vhodni pomikalni register (VH). Izhod iz VH je označen kot vhod avtomata x={0,1}, izhod y={0,1} je rezultat razpoznavanja in se shranjuje v izhodni pomikalni register (IZ).
- Vezje ima na vhodu urin signal Clk za sinhronski vpis podatkov v registre in mora imeti omogočeno brisanje vseh modulov s signalom Reset.

Primer razpoznavanja niza za 8 podanih črk:

Tipkovnica: aabbaaba (→ smer branja vhodnih črk)

IZ (Y): 00000100

1.) Diagram prehajanja stanj (DPS)



- Kodiranje

vhodna množica (X)

Črka	x
a	1
b	0

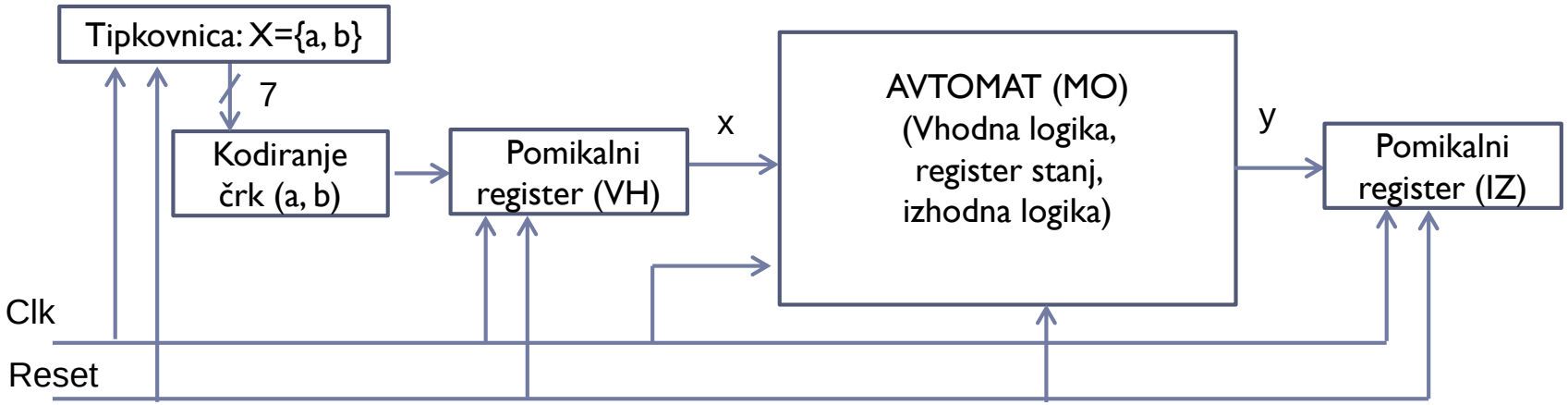
množica stanj (TS)

Stanje	Q_1	Q_0
s_0	0	0
s_1	0	1
s_2	1	0
s_3	1	1

izhod y za TS

Stanje	y
s_0	0
s_1	0
s_2	0
s_3	1

2.) **Blok shema** vezja z Mooreovim avtomatom



3.) Gradniki

- Tipkovnica in pomikalni registri.
- Za register stanj uporabite JK pomnilne celice, kjer je sta vhoda povezana (j=k).
- Za vhodno in izhodno logiko uporabite gradnike, ki omogočajo najbolj enostavno vezje, če imate na voljo logična vrata (NOT, AND, OR, NAND, NOR) in multiplekserje (MUX 2/1, MUX 4/1).

4.) Vhodna logika, register stanj, izhodna logika

- Binarna aplikacijska tabela prehajanja stanj, izhod y in krmilni signali za JK pomnilni celici ($j_1 = k_1$ in $j_0 = k_0$).

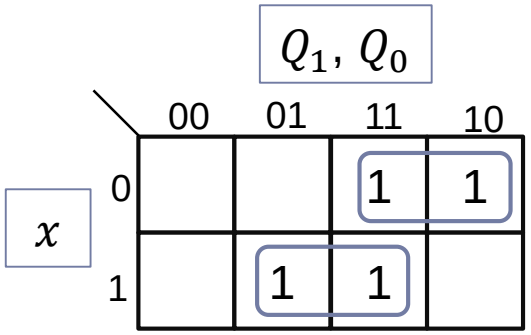
x	$Q_1(t)$	$Q_0(t)$	$Q_1(t)$	$Q_0(t)$	y	$j_1 = k_1$	$j_0 = k_0$
0	0	0	0	I	0	0	I
0	0	I	0	I	0	0	0
0	I	0	0	I	0	I	I
0	I	I	0	I	I	I	0
I	0	0	0	0	0	0	0
I	0	I	I	0	0	I	I
I	I	0	I	I	0	0	I
I	I	I	0	0	I	I	I

Vzbujanje pomnilne celice $j_1 = k_1$

Vzbujanje pomnilne celice $j_0 = k_0$

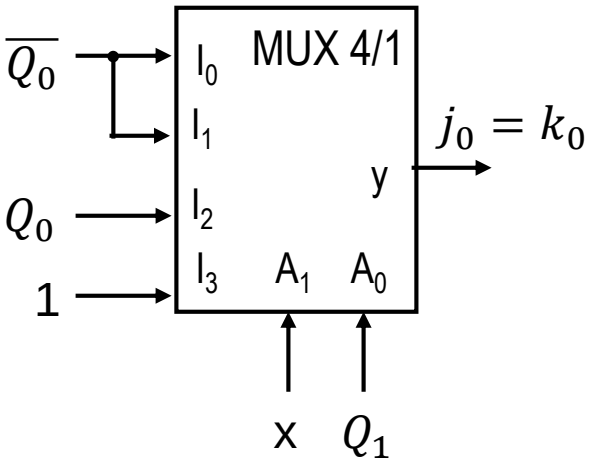
- Vhodna logika za določanje naslednjega stanja:
  - Krmilna funkcija $j_1 = k_1$ je realizirana z logičnimi vrati NAND
  - Krmilna funkcija $j_0 = k_0$ je realizirana z MUX 4/1
- Izhodna logika: y je realizirana z logičnimi vrati AND

$j_1 = k_1$

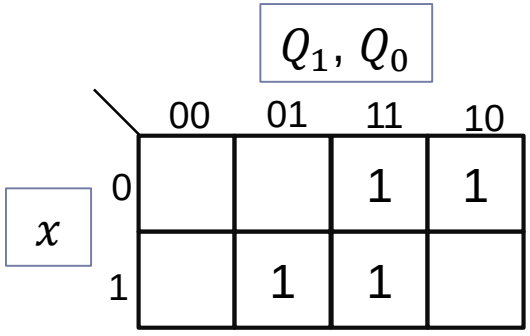


$$j_1 = k_1 = \bar{x}.Q_1 \vee x.Q_0$$
$$= (\bar{x} \uparrow Q_1) \uparrow (x \uparrow Q_0)$$

$j_0 = k_0$



y



$$y = Q_1.Q_0$$

5.) Implementacija

- Pretvorba črk (7-bitna koda ASCII) v binarno vrednost 0 in 1:
  - črka a = 1100001 = 61_h,
  - črka b = 1100010 = 62_h

Pomembno:

Predpostavljamo, da sta na tipkovnici uporabljeni samo črki podane abecede X={a,b}, vhod x = 1 za črko ‘a‘ in vhod x = 0 za črko ‘b‘.

- Psevdokoda pretvorbe, kjer preverjamo ali je na izhodu tipkovnice črka a:

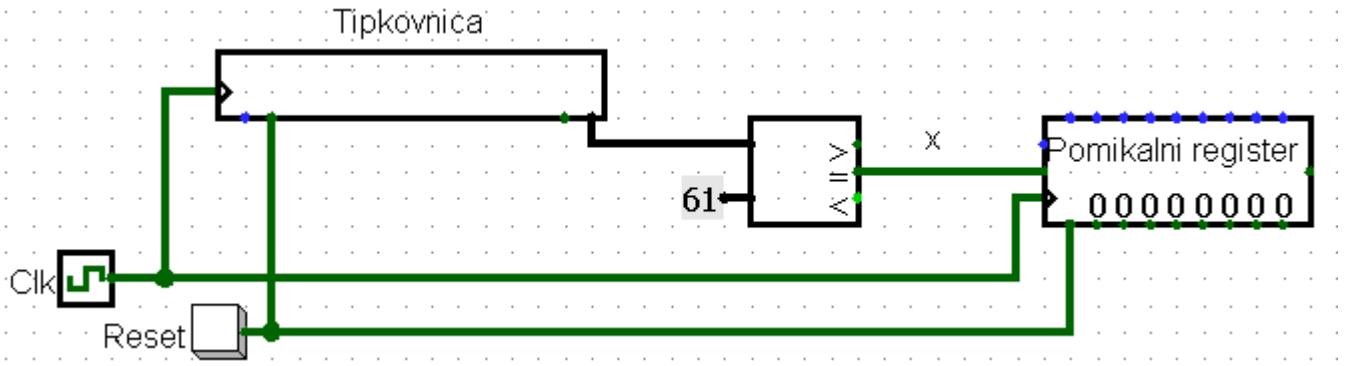
if ASCII (a) == 61_h

x=1

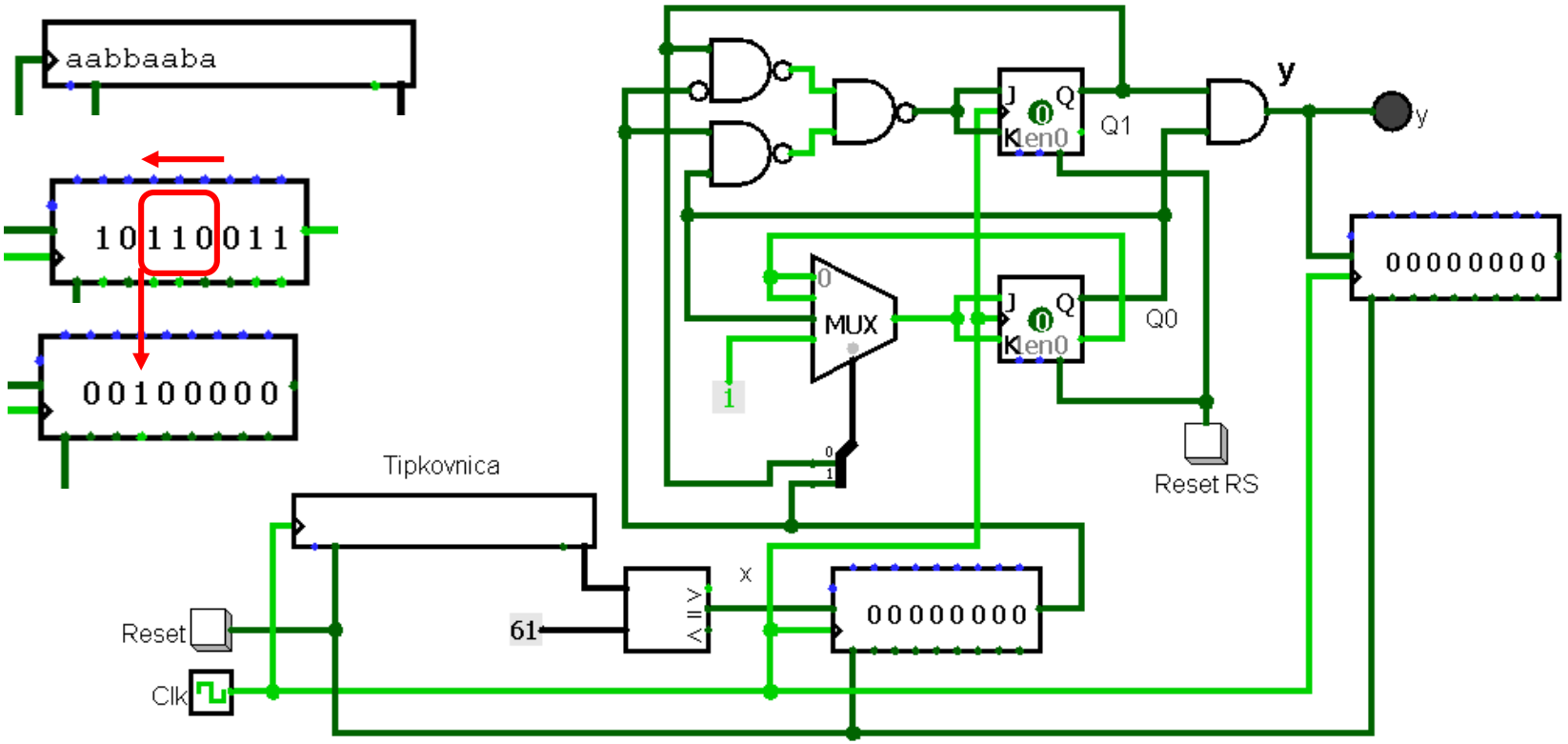
else

x=0

Logično vezje: primerjalnik ima na izhodu 1

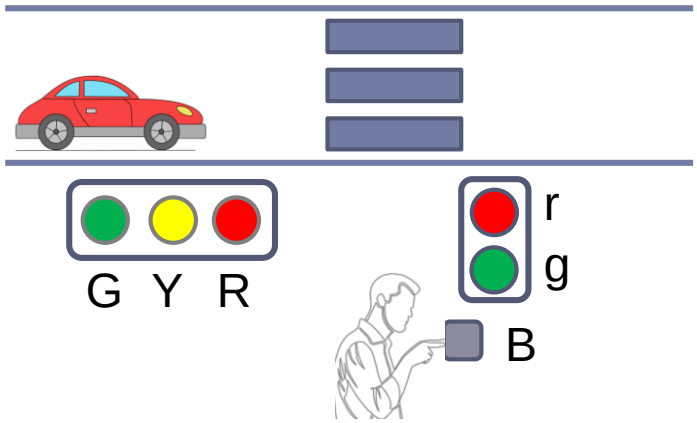


- Realizacija v logisimu – vhodni niz je shranjen v pomikalnem registru.
- Kodiranje znakov – Rešitev a)
- Tipkovnica: aab**baa**ba



N4: Semafor

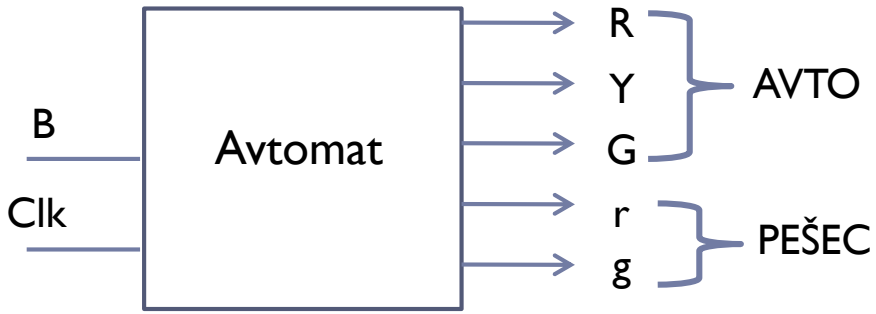
- Realizirajte krmilnik za semafor, ki je nameščen na cesti ob prehodu za pešce, kot je prikazano na sliki. Luči so označene z R-rdeča, Y-rumena, G-zelena za avtomobile in r-rdeča, g-zelena za pešce.
- Pešec bo s pritiskom na gumb B sprožil postopek za spremembo signalizacije tako, da bo lahko prečkal cesto.



- Delovanje semaforja definirajte tako, da izpustite kombinacijo, ko imata oba semaforja rdečo luč in pri določanju stanj za kombinacije prižganih luči pazite na varnost pešcev.
- Poenostavitev, ki jo lahko upoštevate, je enak časovni interval prižganih luči za avtomobile in za pešca.

Opis naloge

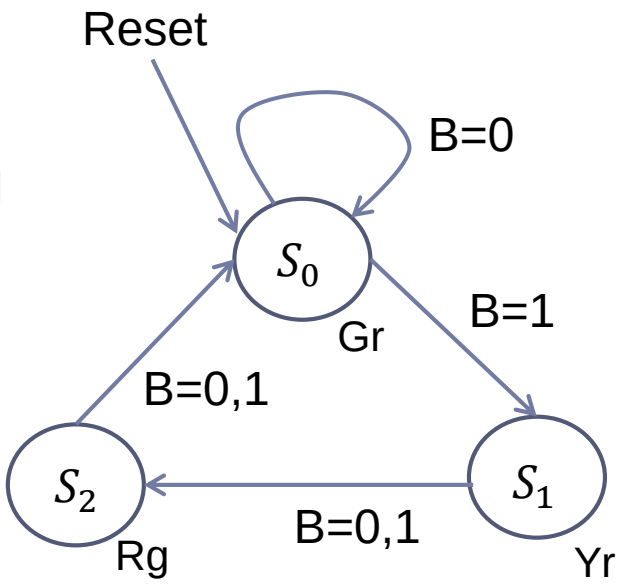
- Vhodni signal: gumb B
- Izhodni signali:
  - Avto: R – rdeča luč, Y – rumena luč, G – zelena luč
  - Pešec: r – rdeča luč, g – zelena luč



- Naloge:
  - Narišite diagram prehajanja stanj
  - Definirajte binarno aplikacijsko tabelo.
  - Pri realizaciji registra stanj uporabite JK pomnilni celici.
  - Za vhodno logiko uporabite logična vrata NAND in MUX 4/1.
  - Izhodno logiko za prižiganje luči realizirajte z logičnimi vrati AND, OR, NOT.

Vhod B in stanja avtomata

- Vhod B = 1 – tipka je pritisnjena
- Stanja z izhodi, ki so varna za pešca poimenujmo kot:
  - Gr – izhodi so zelena luč za avto, rdeča luč za pešca
  - Yr – izhodi so rumena luč za avto, rdeča luč za pešca
  - Rg – izhodi so rdeča luč za avto, zelena luč za pešca
- Diagram prehajanja stanj:
  - Začetno stanje je Gr,
  - B =1 povzroči prehod iz Gr v stanje Yr.
  - Če je B=1, ko smo v stanjih Yr ali Rg, se ne sme prekiniti regularen zaključek prehodov stanj, ki je bil sprožen ob prvem pritisku na tipko.
  - B=0 se v Gr stanje ohranja, v stanjih Yr in Rg pa zagotovi regularen zaključek prehodov v začetno stanje Gr.



- Kodiranje stanj

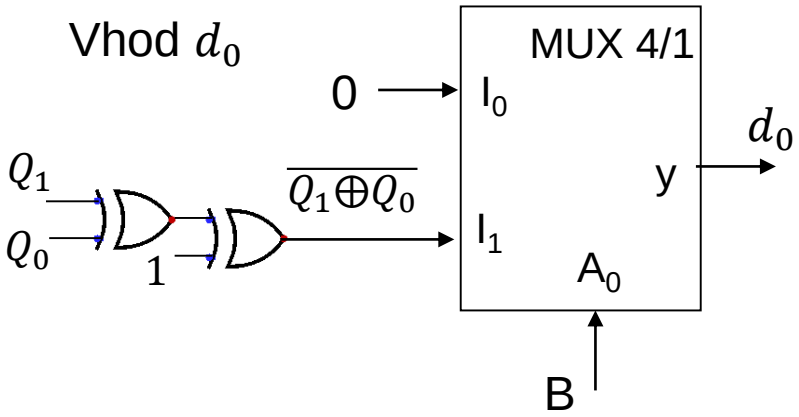
	Q_1	Q_0
S_0	0	0
S_1	0	I
S_2	I	0

- Binarna aplikacijska tabela in **krmiljenje registra stanj** z D pomnilnimi celicami

B	$Q_1(t)$	$Q_0(t)$	d_1 $Q_1(t+1)$	d_0 $Q_0(t+1)$
0	0	0	0	0
0	0	I	I	0
0	I	0	0	0
0	I	I	?	?
I	0	0	0	I
I	0	I	I	0
I	I	0	0	0
I	I	I	?	?

Vhod $d_1 = Q_0$

Vhod d_0



- **Izhodna logika** – krmiljenje luči semaforja, realizacija z logičnimi vrati

stanje (t)			semafor avto			semafor pešec	
	Q_1	Q_0	R	Y	G	r	g
S_0	0	0	0	0	1	1	0
S_1	0	1	0	1	0	1	0
S_2	1	0	1	0	0	0	1
	1	1	?	?	?=0/1	?	?

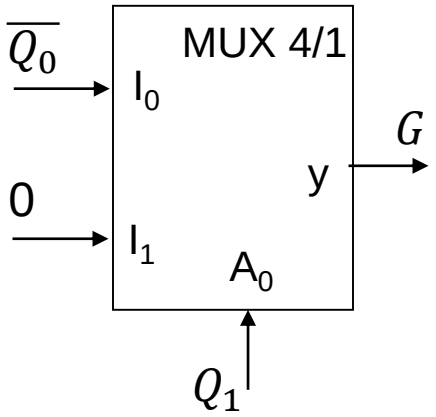
R = Q_1

Y = Q_0

G = $\overline{Q_1}.\overline{Q_0}$ ali G = $\overline{Q_1 \oplus Q_0} = Q_1 \oplus Q_0 \oplus 1$

r = $\overline{Q_1}$

g = Q_1



- Realizacija v logisimu