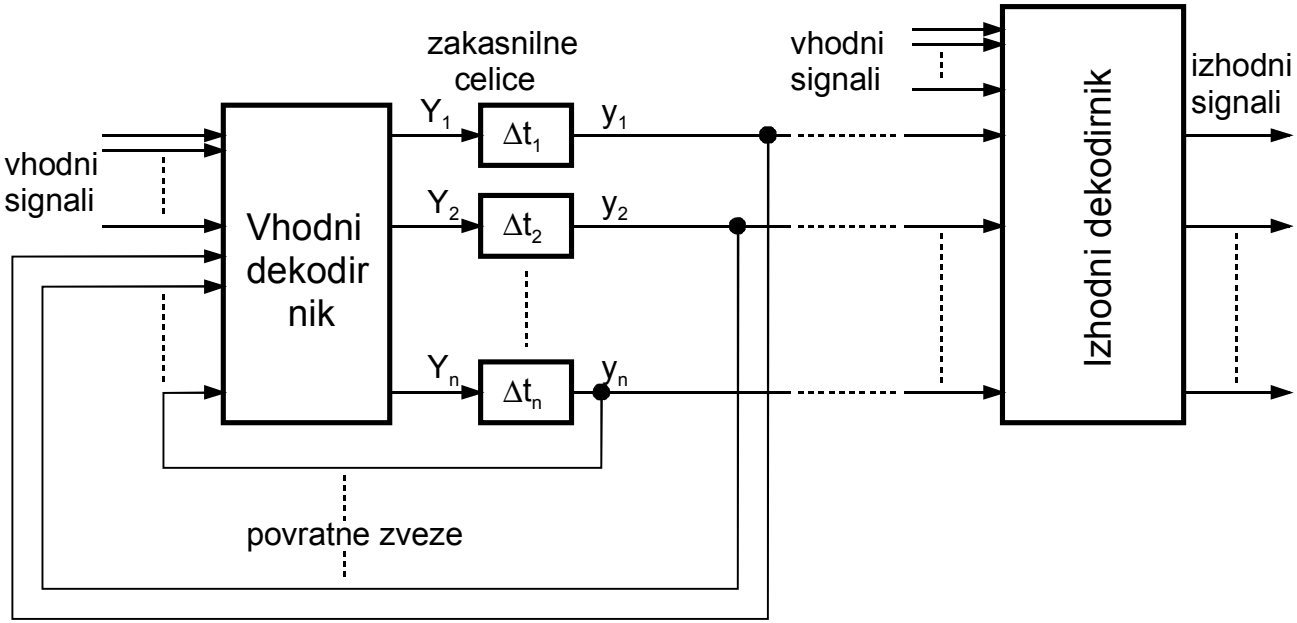
10 ASINHRONSKA SEKVENČNA VEZJA

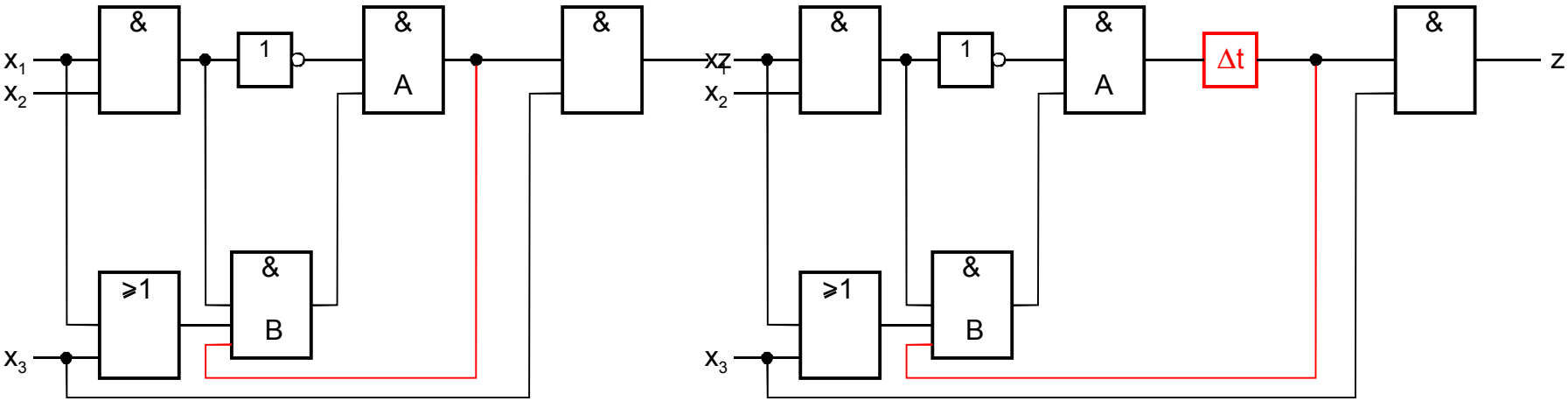
10.1 Model asinhronskega sekvenčnega vezja



$$z_j(t) = \lambda_j\left[x_i(t), y_p(t)\right]$$

$$i = 1,2,...,I; \quad p=1,2,...,n$$

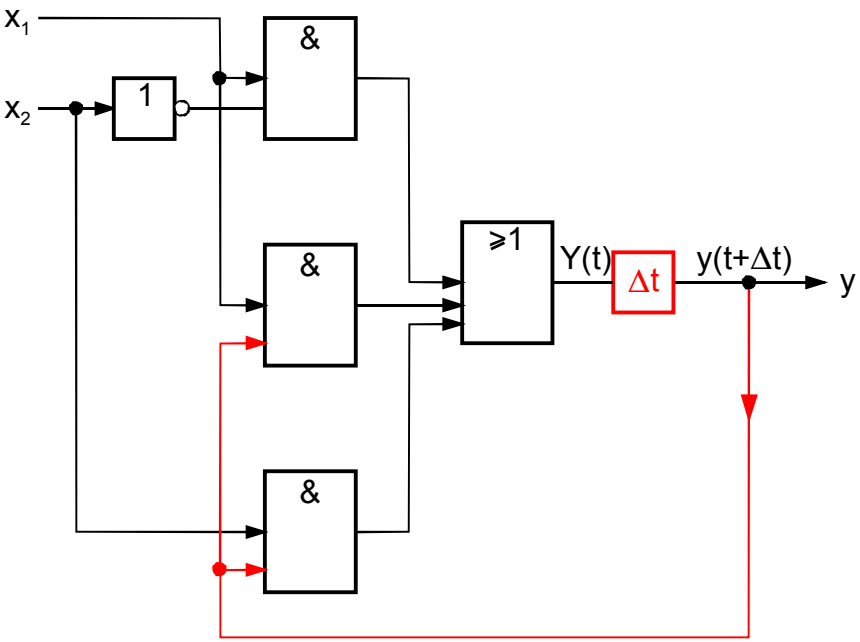
$$Y_k(t) = f_k\left[x_i(t), y_p(t+\Delta t)\right] \quad i = 1,2,...,I; \quad p=1,2,...,n$$



10. 2 Analiza asinhronskih sekvenčnih vezij

- vzbujalna tabela (vzbujalni diagram)
- diagram prehajanja stanj – osnovnih ali združenih
- tabela prehajanja stanj – osnovnih ali združenih

10. 2. 1 Osnovna oblika asinhronskega vezja (Fundamental mode)



$Y(t) = y(t + \Delta t)$.

ali pa

$Y(t) \neq y(t + \Delta t)$.

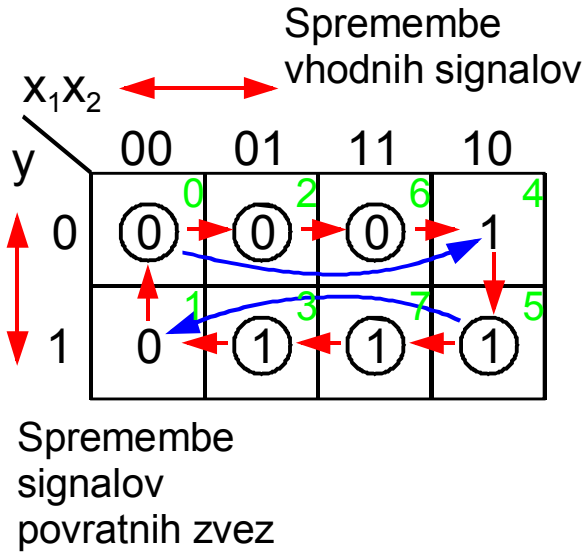
Y(t) notranje stanje ali vzbujalna spremenljivka

y(t+ Δ t) pa signal povratne zveze ali sekundarna vhodna spremenljivka.

$Y(t) = f [x_1 (t), x_2(t) , y(t+ \Delta t)]$

Za opazovano vezje je to:

$Y(t) = x_1(t)\overline{x}_2(t)+ x_1(t)y(t+ \Delta t) + x_2(t)y(t+ \Delta t)$



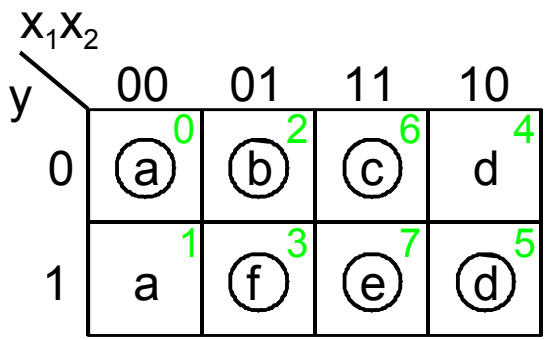
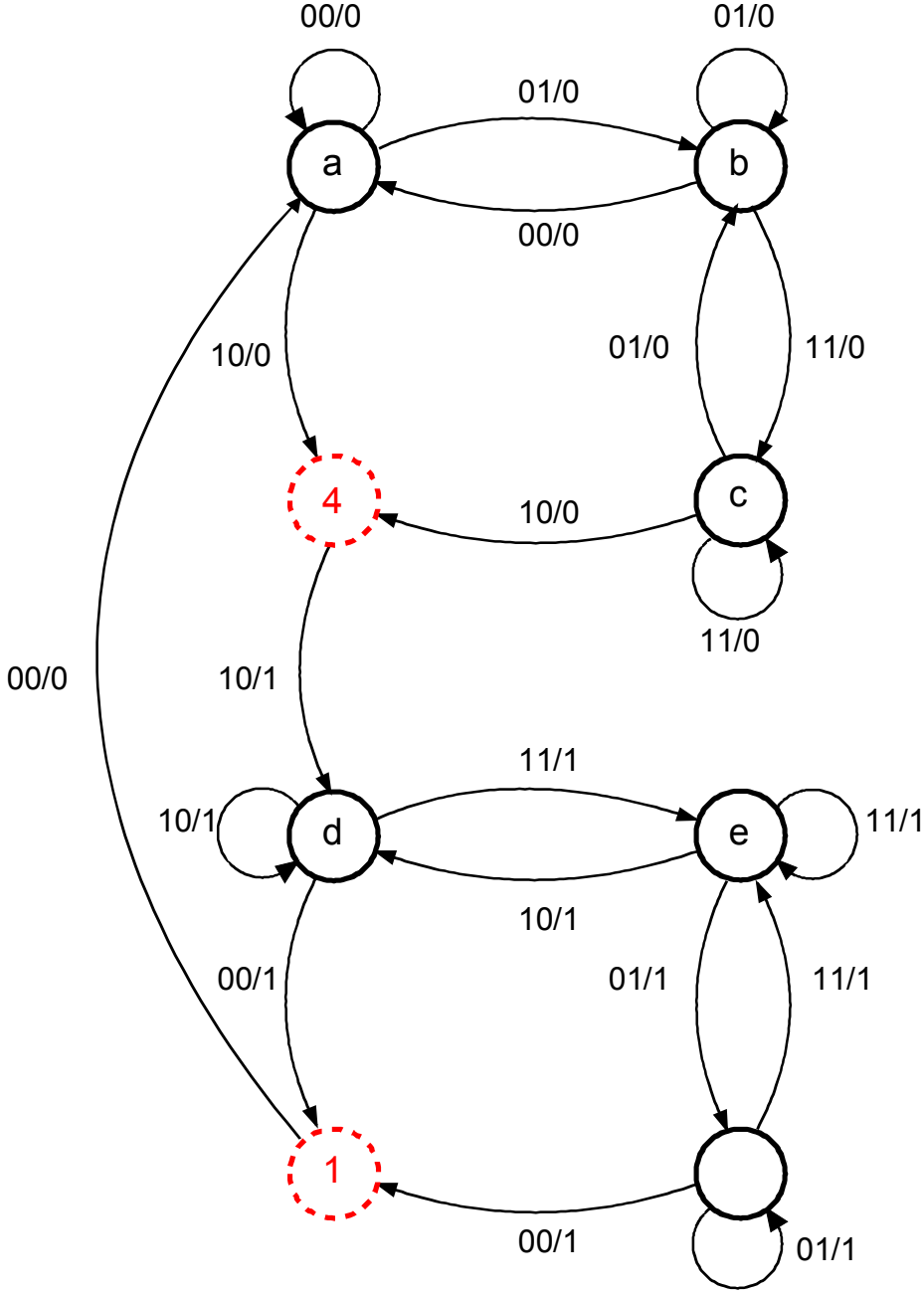
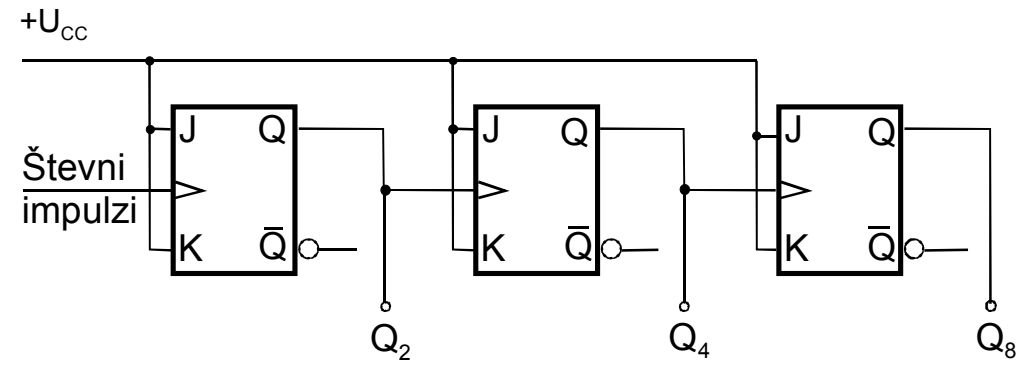
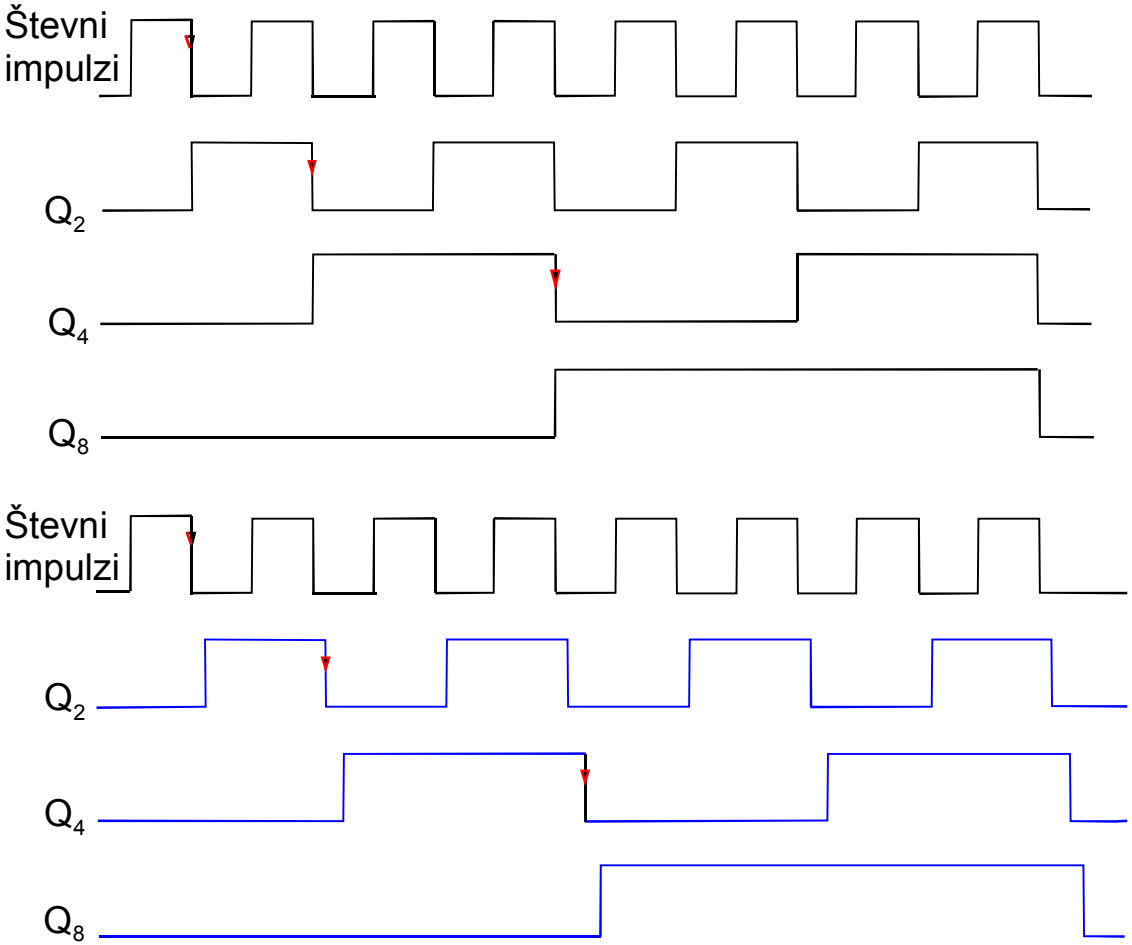


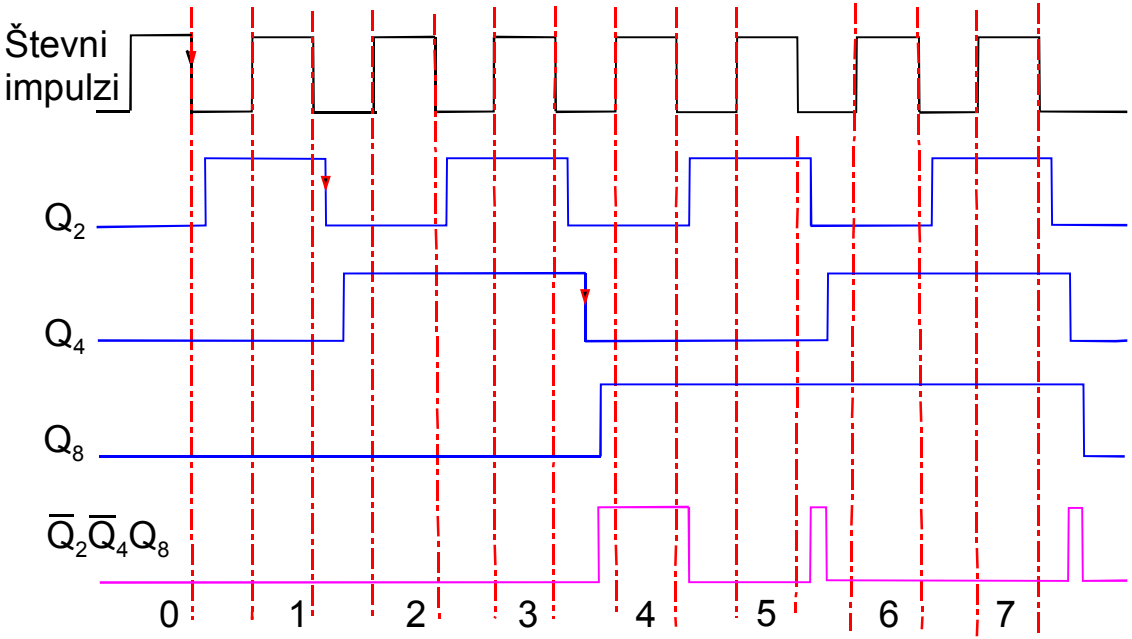
Diagram prehajanja osnovnih (elementarnih) stanj asinhronskega vezja

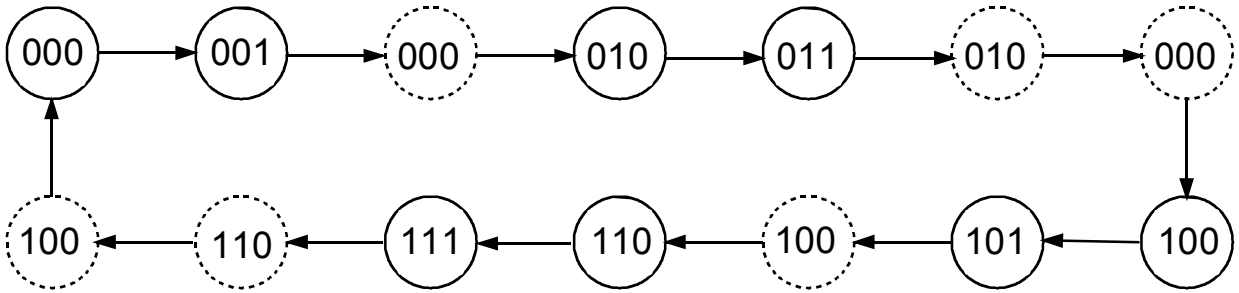


10.2.2 Analiza ripple števca







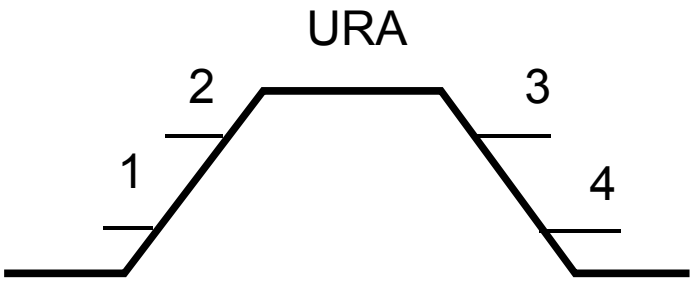


$$t_z = \sum_{i=1}^{n} t_i, \qquad t_z = n\Delta t,$$

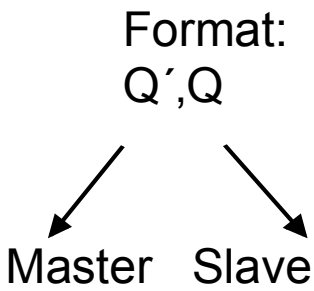
$$\Delta t = 10ns; \quad n = 8; \quad t_z = 80ns$$

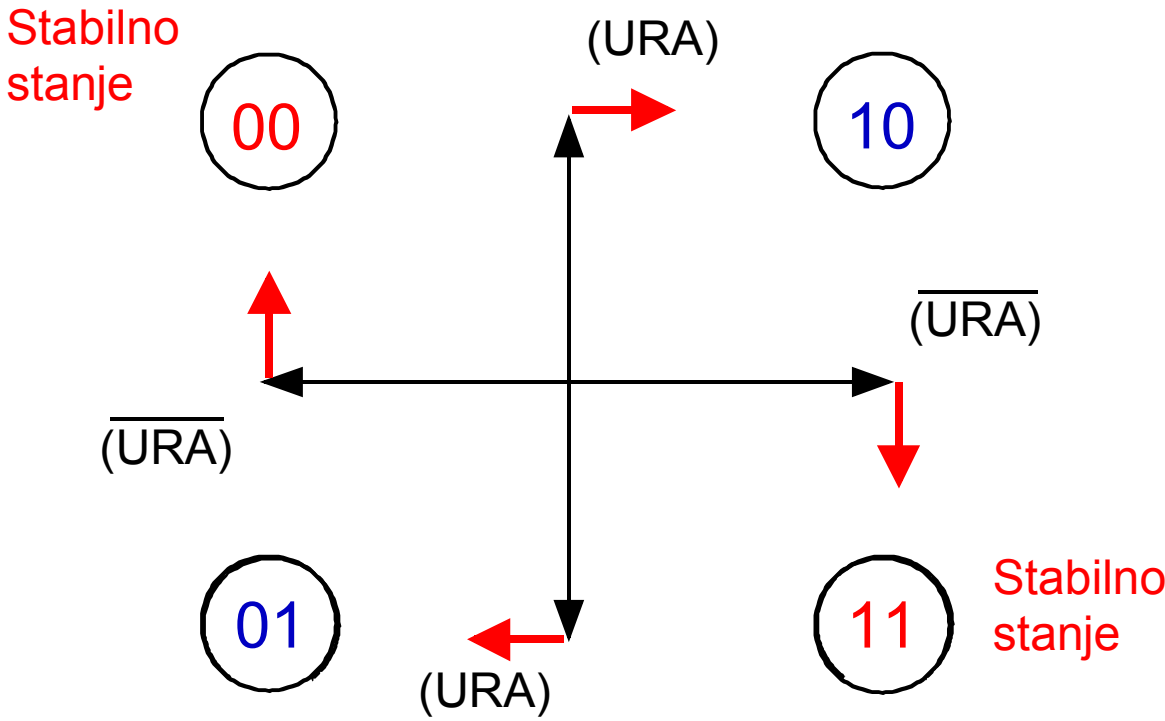
$$f_{max} = \frac{1}{t_z} = \frac{1}{80ns} = 12{,}5MHz$$

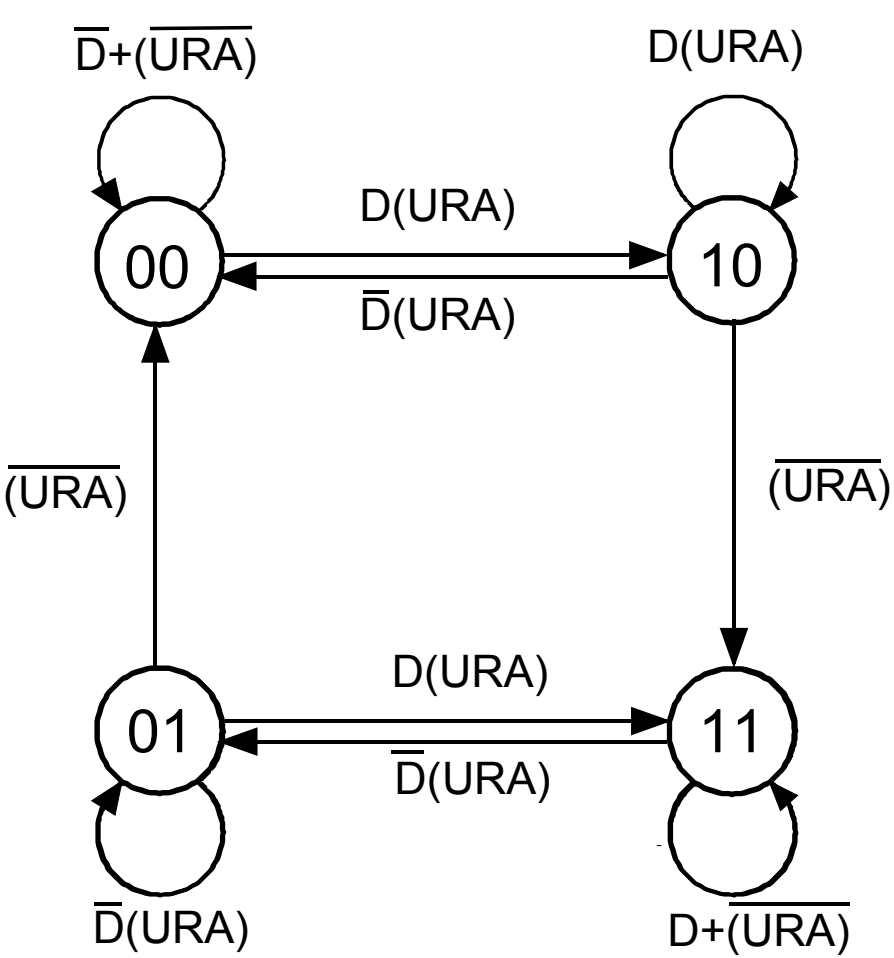
10.2.3 Asinhronska analiza spominskih celic s predpomnjenjem

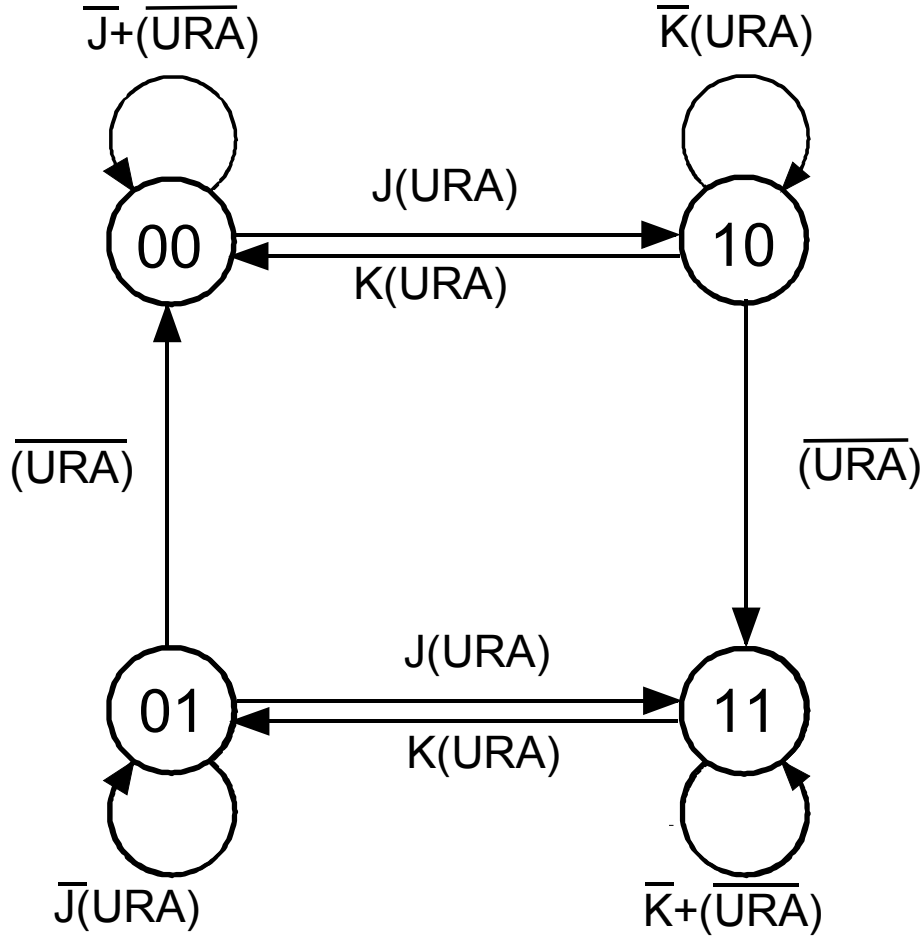


1: Zapora slave celice
2: Vpis v master celico
3: Zapora master celice
4: Vpis v slave celico






$\overline{D}+(\overline{URA})$
D(URA)
00
10
D(URA)
$\overline{D}$(URA)
$(\overline{URA})$
$(\overline{URA})$
01
11
D(URA)
$\overline{D}$(URA)
$\overline{D}$(URA)
D+$(\overline{URA})$


$\overline{J}+(\overline{URA})$
$\overline{K}$(URA)
00
10
J(URA)
K(URA)
$(\overline{URA})$
$(\overline{URA})$
01
11
J(URA)
K(URA)
$\overline{J}$(URA)
$\overline{K}+(\overline{URA})$

10. 3 Asinhronski sekvenčni avtomati

10. 3. 1 Uvod in osnovne karakteristike

Kadar je širina “urinih” impulzov primerljiva z zakasnitvami posameznih sklopov vezja moramo delovanje vezja obravnavati v luči asinhronske analize. Urin impulz, če obstoja, postane tako neodvisen vhod.

V veliki večini primerov ni vhodnih sprememb, ki bi se dogajale ob ekvidistantnih časovnih intervalih.

Spremembe stanj bodo torej nastopale v spremenljivih diskretnih časih naključno razmaknjenih med seboj.

Da bomo lahko ta vezja podredili dvovrednostni logiki, bomo morali privzeti določene predpostavke in postaviti nekatere neizbežne omejitve.

Ena izmed teh je Huffmanov pogoj !

Še več – omejitev moramo postaviti tudi na frekvenco ponavljanja vhodnih sprememb.

Spremembe vhodov se smejo dogajati samo v časovnih intervalih v katerih se celotno vezje zadržuje v stabilnem stanju, povzročenem s prejšnjo spremembo na vhodu.

Jezik avtomata, kot sredstvo za interpretacijo algoritma, mora biti metodološko prirejen sintezi in analizi teh vezij podobno, kot smo imeli pri sinhronskih.

Kljub temu pa nastopajo nekatere pomembne razlike. Ena teh je:

Totalno stanje avtomata – kombinacija vhodnega in notranjega stanja.

Tu je potrebno strogo ločiti med totalnim, notranjim in vhodnim stanjem.

Za vsako totalno stanje tabela prehajanja stanj določa naslednje notranje stanje in izhodno stanje v obliki kombinacije izhodnih spremenljivk.

Množico totalnih stanj bomo označevali takole:

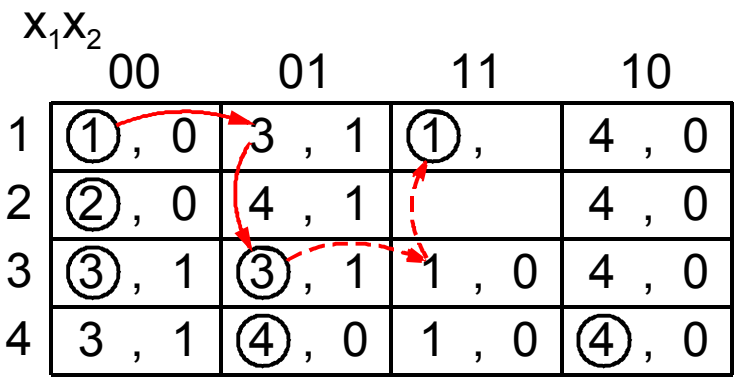
$$\mathbf{T}^A = \{\mathbf{s}_{1T}, \mathbf{s}_{2T}, \ldots, \mathbf{s}_{nT}\}$$

pri čemer je na primer: $\mathbf{s}_{lT} = \mathbf{s}_k^i = (\mathbf{s}_k, \mathbf{x}_i)$

kombinacija notranjega stanja $\mathbf{s}_k \in \mathbf{S}$ in vhodnega stanja $\mathbf{x}_i \in \mathbf{X}$

Analogno vpeljimo tudi oznake izhodnih stanj: $\mathbf{z}_{jT} = \mathbf{z}_j^i \in \mathbf{Z}$

x_1x_2	00	01	11	10
1	①, 0	3 , 1	①,	4 , 0
2	②, 0	4 , 1		4 , 0
3	③, 1	③, 1	1 , 0	4 , 0
4	3 , 1	④, 0	1 , 0	④, 0



Tako zamišljen in formuliran način delovanja imenujemo že omenjeno prehajanje osnovnih stanj ali osnovni način delovanja (Fundamental mode)

- za vsako vhodno vzbujanje obstoja najmanj eno stabilno stanje,
- nova sprememba vhodnega stanja se lahko zgodi šele potem, ko je avtomat dosegel stabilno stanje, povzročeno s prejšnjo spremembo vhodnega stanja,
- izmed izhodnih stanj so pomembna in uporabna samo tista stanja, ki pripadajo stabilnim notranjim stanjem.

Tabela posamičnih izhodnih prehodov (PIP tabela):

x	0	1
1	①, 00	2 , 00
2	②, 11	3 , 01
3	4 , 10	③, 01
4	④, 10	3 , 01

a)

x	0	1
1	①, 00	3 , 00
2	②, 11	3 , 01
3	4 , 10	③, 01
4	④, 10	3 , 01

b)

x: 0 → 1

Vendar, če to ne povzroča nobenih posledic, lahko obe tabeli prehajanja stanj jemljemo kot ekvivalentni, glede na to, da se pri obeh za enake vhodne sekvence generirajo na izhodu iste sekvence.

V primeru, da po PIP tabeli vsako prehajanje pelje direktno v stabilno stanje, pravimo, da asinhronski avtomat deluje v normalnem načinu.

Vsako PIP tabelo lahko transformiramo v tabelo normalnega načina delovanja, če je le trajanje spremembe izhoda nepomembno.

VIP tabela

x	0	1
1	2 , 10	① , 11
2	3 , 00	1 , 10
3	③ , 01	2 , 01

a)

x_1x_2	$\mathbf{x}_1$ 00	$\mathbf{x}_2$ 01	$\mathbf{x}_3$ 11	$\mathbf{x}_4$ 10
1	① , 00	2 , 10	① , 11	-,-
2	3 , 11	② , 10	3 , 00	-,-
3	③ , 11	4 , 01	4 , 01	-,-
4	1 , 00	④ , 01	2 , 10	-,-

b)

Če totalni stanji (2,$\mathbf{x}_2$) ali (4,$\mathbf{x}_2$) vzbudimo z $\mathbf{x}_3$, avtomat ne more več priti v stabilno stanje!

Za razliko od primera a) tu ni mogoče osnovno delovanje, zato gornja tabela podaja “nefundamentalni mode”, za katerega so značilne oscilacije v enem ali več stolpcih.

Iz teh primerov zaključimo, da vse tabele, ki podajajo normalen način, podajajo hkrati tudi osnovnega, toda tabele osnovnega načina ne podajajo vedno tudi normalnega.

x_1x_2	00	01	11	10
1	①, 0	3 , 1	①,	4 , 0
2	②, 0	4 , 1		4 , 0
3	③, 1	③, 1	1 , 0	4 , 0
4	3 , 1	④, 0	1 , 0	④, 0

	x_1	x_2	x_3	x_4	x_5
1	①, S	1 , 0	3 , 1	1 , 0	4 , 0
2	②, S	2 , 0	4 , 1	-,-	4 , 0
3	③, S	3 , 1	3 , 1	1 , 0	4 , 0
4	④, S	3 , 1	4 , 0	1 , 0	4 , 0

Stolpci $\mathbf{x}_2$ do $\mathbf{x}_4$ ustrezajo stolpcem leve tabele. Vsak stolpec $\mathbf{x}_i$, kjer je $i \neq 1$ predstavlja impulzni vhod. Stolpec $\mathbf{x}_1$ pa predstavlja odsotnost vzbujanja.

Obnašanje avtomata pri spremembi vzbujanja od impulza $\mathbf{x}_i$ na impulz $\mathbf{x}_k$ lahko določimo z opazovanjem prehodov iz $\mathbf{x}_i$ na $\mathbf{x}_j$ in iz $\mathbf{x}_j$ na $\mathbf{x}_k$.

10. 3. 2 Metode sinteze in minimizacije sinhronskih avtomatov

10. 3. 2. 1 Primitivna tabela prehajanja stanj

Najbolj občutljiva faza sinteze se nanaša na pravilno formuliranje, oziroma na natančno jezikovno interpretacijo delovanja avtomata. Nabor vseh funkcionalnih karakteristik in definicij končnega algoritma je večinsko osnovan na intuiciji, zato nobena metodologija sinteze ne more v nadaljnjih korakih sinteze izločiti netočnosti ali nepavilnosti algoritma.

Izhodiščna osnova za natančno metodološko sintezo je že omenjena tabela prehajanja stanj.

Določitev te tabele prehajanja stanj se izvrši na osnovi verbalnega, matričnega opisa ali pa grafičnega v obliki diagrama prehajanja stanj.

Diagram prehajanja nam močno olajša kontrolo in morebitno potrebno korekcijo funkcionalnih karakteristik, ki svojo kanonično obliko dobijo prav z tabelo prehajanja stanj.

Pretvorba jezikovnih izrazov v tabelo prehajanja stanj je v principu enaka, kot pri sinhronskih avtomatih.

Kljub temu pa določene posebnosti, ki nastopajo pri asinhroskih avtomatih zahtevajo nekoliko več pazljivosti, kar se odraža tudi pri strukturi te tabele prehajanja stanj.

Začetno obliko te tabele imenujemo primitivna tabela prehajanja stanj – tudi tabela prehajanja osnovnih stanj; skladno z osnovnim načinom delovanja asinhronskega avtomata.

Izhajamo namreč iz predpostavke, da za vsako vhodno vzbujanje obstoja stabilno stanje v avtomatu, kateremu je pridruženo odgovarjajoče izhodno stanje. Če ta predpostavka nebi veljala, potem celoten poskus, da z avtomatom realiziramo vnaprej definirane in programirane pretvorbe informacije postane absurden – avtomat nebi nujno sploh moral priti v stabilno stanje.

S tem zavestno vnašamo redundanco, saj tako vsakemu stabilnemu (totalnemu) stanju dodelimo svojo vrstico v tabeli (primitivni) prehajanja stanj.

Vendar moramo še enkrat poudariti, da to redundantnost uvajamo zaradi maksimalne zanesljivosti glede točnosti in vernosti algoritemskega prikaza.

Z metodami reduciranja stanj nato zmanjšamo število stanj na minimalno potrebno, čeprav so te metode pogosto težavne in dogotrajne.

Da je postopek zahteven se bomo prepričali na primeru nekoliko bolj zapletenega števca. Izhajali bomo torej iz verbalnega opisa algoritma.

Števec naj ima dva vhoda x_1 in x_2. Vhodno stanje $x_1 = 0$ in $x_2 = 0$ da vzbujanje za brisanje števca, ki ga pripelje v odgovarjajoče stabilno stanje in pripadajoče izhodno stanje 00.

Dovoljeno je, da se za vsako sukcesivno vzbujanje spremeni samo eno stanje.

Števec mora povečati svojo vrednost za ena vsakič, kadar se na vhodu pojavi vzbujanje $x_1,x_2 = 01$ in za dve vsakič, ko je vzbujanje $x_1,x_2 = 10$.

Vzbujanje $x_1,x_2 = 11$ pa ne sme sprementi vsebine števca.

Kombinacije izhodnih spremenljivk z_1 in z_2 naj prestavljajo trenutno stanje števca.

Ta verbalni opis je potrebno sedaj pretvoriti v primitivno tabelo prehajanja stanj.

X	**x_1**	**x_2**	**x_3**	**x_4**
S	**x_1x_2=00**	**x_1x_2=01**	**x_1x_2=11**	**x_1x_2=10**
1	**1** , 00	2	- ; -	3
2	1	**2** , 01	4	- , -
3	1	- , -	5	**3** , 10
4	- , -	6	**4** , 01	7
5	- , -	8	**5** , 10	9
6	1	**6** , 10	5	- , -
7	1	- , -	10	**7** , 11
8	1	**8** , 11	10	- , -
9	1	- , -	11	**9** , 00
10	- , -	13	**10** , 11	12
11	- , -	2	**11** , 00	3
12	1	- , -	4	**12** , 01
13	1	**13** , 00	11	- , -

Začetno stanje je 1 z izhodom 00

Na vhodu se lahko pojavljajo vzbujanja:

01, 10, 11 po katerem koli vrstnem redu.

V splošnem primeru se notranje stanje $\mathbf{s}_1$ spremeni v neko drugo, ki mu dodelimo simbol $\mathbf{s}_j$.

Pomembno je, da je to stanje stabilno in da ne podaljšuje časa prehajanja v stabilno stanje oziroma naredi trajno prehajanje.

Omeniti je potrebno, da vhodno vzbujanje ne povzroči nujno tudi spremembo notranjega stanja, čeprav to v tej fazi obravnave še ni razvidno.

Omeniti je potrebno zato, ker je to pogoj za dodatno reduciranje notranjih stanj in daje problemu minimizacije asinhronskih avtomatov novo dimenzijo v primerjavi s sinhronskimi.

Stolpec 11 je v prvi vrstici ne specificiran, ker smo zahtevali, da števec tega vzbujanja ne registrira. To vzbujanje bi se lahko zapomnilo samo kot neko novo notranje stanje.

Vidimo pa tudi, da sta se hkrati spremenile dve vhodne spremenljivke, kar pomeni, da toleriramo nefundamentalni način delovanja; toda v tem vezju to želimo!

Pri stabilnem stanju 2 v stolpcu 01 lahko nastopi vzbujanje $x_1x_2 = 00$ ali 11, kar povzroči resetiranje števca ali pa prehod v novo stabilno stanje 4 z izhodom $z_1z_2 = 01$

Za vzbujanje 10 tu nimamo specificiranega naslednjega stanja in zopet zato ne, ker nastopa dvojna sprememba vhoda.

Redukcija primitivne tabele zajema, poleg že znanih metod iskanja ekvivalentnih razredov stanj, tudi postopek spajanja in nekatere posebnosti, ki so vezane na specifični način delovanja.

10. 3. 2. 2 Reduciranje primitivne tabele prehajanja stanj.

Reduciranje primitivne tabele vključuje v sebi zmanjšanje števila notranjih stanj asinhronskega avtomata.

Spomnimo se, da je stabilno stanje definirano z vhodnim in notranjim stanjem.

Zato ni razloga, da ista koda nebi bila pridružena različnim notranjim stanjem v primitivni tabeli.

To pomeni, da se lahko v isti vrstici nahajata dve oziroma več (totalnih) stabilnih stanj, do katerih spremembe prihaja izključno zaradi sprememb vhodnih stanj.

S tem avtomatično zmanjšujemo število vrstic primitivne tabele.

Zavedati pa se moramo, da s tem zmanjšujemo tudi število sekundarnih vhodnih spremenljivk s katerimi kodiramo notranja stanja.

Kasneje bomo spoznali, da za spajanje ni neobhodno potrebna identičnost izhodnih stanj.

Dalje, omejitev, ki se nanaša na posamično spremembo stanja vhodnih spremenljivk, pogosto omogoča reduciranje tabele prehajanja stanj.

To možnost se lahko enostavno modelira z nespecificiranim stanjem v primitivni tabeli.

Toda, če v tabeli obstojajo vrstice z dvema ali več stabilnimi stanji, se lahko zgodi, da za nekatere stolpce znotraj te vrstice ni več možno skupno sekundarno vzbujanje.

Opazujmo naslednjo tabelo prehajanja

x_1x_2	00	01	11	10
1	①, 0	2, 0	-, -	3, 0
2	1, 0	②, 0	②, 0	4, 1
3	4, 1	2, 0	③, 1	③, 0
4	④, 1	2, 0	2, 0	4, 1

x_1x_2	00	01	11	10
1	①, 0	2, 0	-, -	3, 0
2	1, 0	②, 0	2´, 0	-, -
2´	-, -	2, 0	②´, 0	4, 1
3	4´, 1	-, -	3´, 1	③, 0
3´	-, -	2, 0	③´, 1	3, 0
4	4´, 1	-, -	2´, 0	④, 1
4´	④´, 1	2, 0	-, -	4, 1

x_1x_2	00	01	11	10
A	Ⓐ, 0	Ⓐ, 0	B, 0	C, 0
B	Ⓑ, 1	A, 0	Ⓑ, 0	Ⓑ, 1
C	B, 1	A, 0	Ⓒ, 1	Ⓒ, 0

A: (1,2)

B: (2´, 4, 4´)

C: (3, 3´)

In pripadajočo primitivno tabelo prehajanja

x_1x_2	00	01	11	10
1	①, 0	2, 0	-, -	3, 0
2	1, 0	②, 0	2´, 0	-, -
2´	-, -	2, 0	②´, 0	4, 1
3	4´, 1	-, -	3´, 1	③, 0
3´	-, -	2, 0	③´, 1	3, 0
4	4´, 1	-, -	2´, 0	④, 1
4´	④´, 1	2, 0	-, -	4, 1

Razumljivo je, da vsako PIP tabelo lahko pretvorimo nazaj primitivno tabelo prehajanja stanj, ki je izhodiščna v postopku minimizacije.

V tabelah prehajanja v katerih obstoja večkratno prehajanje izhodnih spremenljivk (VIP) mora biti izhod, ki je pridružen nestabilnemu stanju, enak izhodu, ki je pridružen stabilnemu stanju in to na začetku ali na koncu prehoda.

Če isti izhod pridružimo nestabilnim stanjem in končnemu stanju to samo skrajša čas prehajanja.

Torej, če je neko stanje nestabilno v posameznem stolpcu, je lahko njegovo naslednje stanje in pridružen izhod enako stabilnemu stanju in njegovemu pridruženemu izhodu v tem stolpcu.

Namreč, če je s_j nestabilno stanje v stolpcu $\mathbf{x}_i$, a $\mathbf{s}_k$ stabilno, ki ga bo to nestabilno stanje doseglo, potem lahko izenačimo:

$$\mathbf{s}_k = s_j^i = \delta(s_j, \mathbf{x}_i) \quad \text{in} \quad \mathbf{z}_k^i = z_j^i = \lambda(s_j, \mathbf{x}_i) = \lambda(\mathbf{s}_k, \mathbf{x}_i)$$

S tem postopkom pretvarjamo PIP tabelo v tabelo prehajanja za normalni način delovanja.

Izrek :

Če je par s_k, s_l neke PIP tabele “M” kompatibilen, potem je ta isti par kompatibilen tudi v tabeli M´ za normalni način delovanja, ki je nastala na osnovi normalizacije tabele M.

Dokaz:

Naj bosta $\mathbf{s}_m$ in $\mathbf{s}_n$ stabilni stanji v katera pridemo iz stanj s_k oziroma s_l z vzbujanjem $\mathbf{x}_i$.

Ker sta s_k in s_l kompatibilni v M morata biti kompatibilni tudi stanji $\mathbf{s}_m$ in $\mathbf{s}_n$ v M´.

To pomeni: $\delta(s_k, \mathbf{x}_i)=s_k^{\mathbf{i}} = \mathbf{s}_m$ in $\delta(s_l, \mathbf{x}_i) = s_l^i = \mathbf{s}_n$.

Stanji s_k in s_l bosta kompatibilni v M´, če bosta izhodno kompatibilni.

S konstrukcijo M´dobimo:

$$z_k^i = \mathbf{z}_m^i \quad \text{in} \quad z_l^i = \mathbf{z}_n^i$$

Velja pa :

$$\mathbf{z}_m^i = \mathbf{z}_n^i$$

ker sta stanji $\mathbf{s}_m$, in $\mathbf{s}_n$ kompatibilni.

Zaradi tega je:

$$z_k^i = z_l^i$$

kar pomeni, da sta stanji s_i in s_j kompatibilni tudi v M´.

Normalizacija torej ne ruši obstoječih kompatibilnih parov, lahko pa ustvari nove.

Oglejmo si primer avtomata, ki nima kompatibilnih parov stanj.

x_1x_2	00	01	11	10
1	①, 0	2, 0	2, 0	①, 0
2	4, 1	3, 0	②, 1	②, 1
3	1, 0	③, 0	4, 1	-, -
4	④, 1	3, 1	④, 1	④, 1

x_1x_2	00	01	11	10
1	①, 0	3, 0	2, 1	①, 0
2	4, 1	3, 0	②, 1	②, 1
3	1, 0	③, 0	4, 1	-, -
4	④, 1	3, 0	④, 1	④, 1

x_1x_2	00	01	11	10
1	①, 0	①, 0	2, 1	①, 0
2	②, 1	①, 0	②, 1	②, 1

Te postopke je potrebno nekoliko predrugačiti pri VIP tabelah.

VIP tabele je mogoče interpretirati na dva načina.

Trajanje vsakega izhodnega stanja je lahko proporcionalno številu zaporednih prehodnih stanj v času katerih se generirajo posamična izhodna stanja. V tem primeru je izhod časovno odvisen.

V osnovnem načinu delovanja asinhronskega vezja, ki ga je mogoče opisati z VIP tabelo, predpostavljamo da vezje starta iz stabilnega stanja in da se vhodne spremembe dogajajo šele, ko pride vezje v odgovarjajoče novo stabilno stanje.

Rezultat tega je, da moramo za neko totalno nestabilno stanje analizirati polja (vrstice) samo enega vhodnega stolpca za posamezno prehajanje.

To lahko generira pare, ki pod predpostavko sinhronskega delovanja vezja niso bili kompatibilni, tu pa postanejo kompatibilni.

Pojasnimo to s primerom:

X	x_1	x_2
1	①, 0	2 , 1
2	②, 1	3 , 0
3	③, 0	③, 1
4	④, 0	3 , 0
5	⑤, 0	4 , 1

10. 3. 2. 3 Spajanje vrstic

	000	001	010	100	$z_1z_2z_3$
1	①	2	4	6	000
2	3	②	-	-	001
3	③	2	4	6	001
4	5	-	④	-	010
5	⑤	2	4	6	010
6	7	-	-	⑥	100
7	⑦	2	4	6	100

	00	01	11	10	z
α	4	①	2	3	0
β	4	1	②	3	1

	00	01	11	10
(α,β)	4	① 0	② 1	3

Pri spajanju posameznih vrstic se moramo držati naslednjih splošnih pravil:

Dve ali več vrstic lahko spojimo samo pod pogojem, da imajo v istih stolpcih identična stanja (stabilna ali nestabilna) ne glede na njihove izhode.

Katerokoli stanje, ki je obkroženo ali neobkroženo se lahko spoji z redundatnim stanjem, če s tem nista kršeni ti dve pravili v ostalih stolpcih vseh spajanih vrstic.

x_1x_2				Z
①	2	-	4	00
-	2	3	④	00

→ spojena vrstica

x_1x_2				Z
①	2	3	④	00

x_1x_2				Z
①	2	-	3	10
1	②	-	-	11

→ spojena vrstica

x_1x_2			
①^{10}	②^{11}	-	3

x_1x_2				Z
①	2	3	-	101
1	②	-	-	011
-	2	③	4	100

→ spojena vrstica

x_1x_2			
①^{101}	②^{011}	③^{100}	4

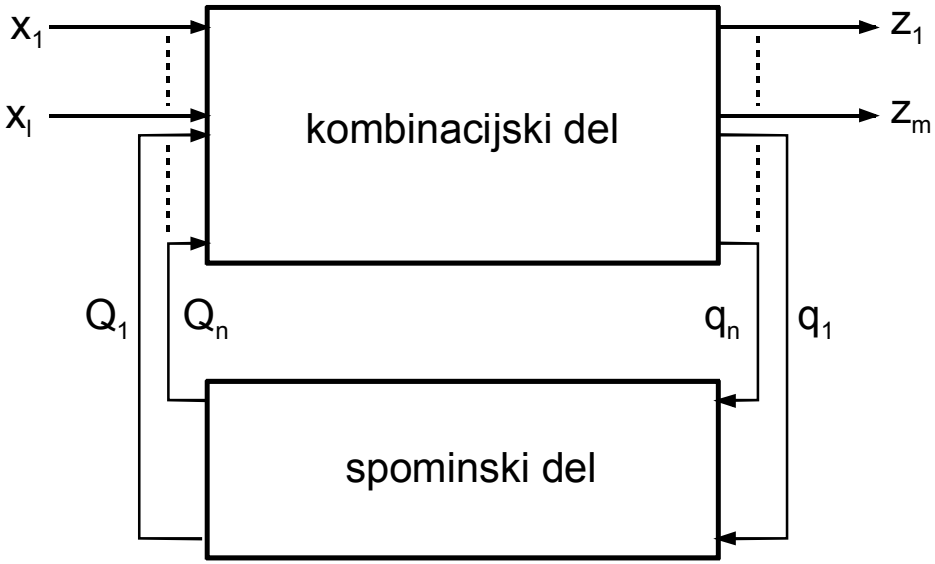
x_1x_2				**Z**
①	2	4	-	00
1	2	④	5	01
1	3	4	⑤	11

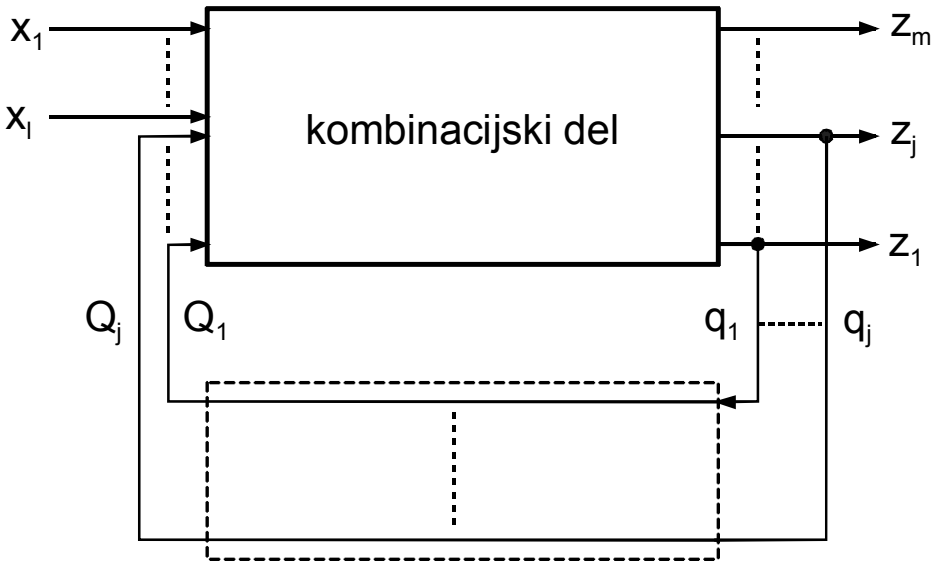
spojena vrstica

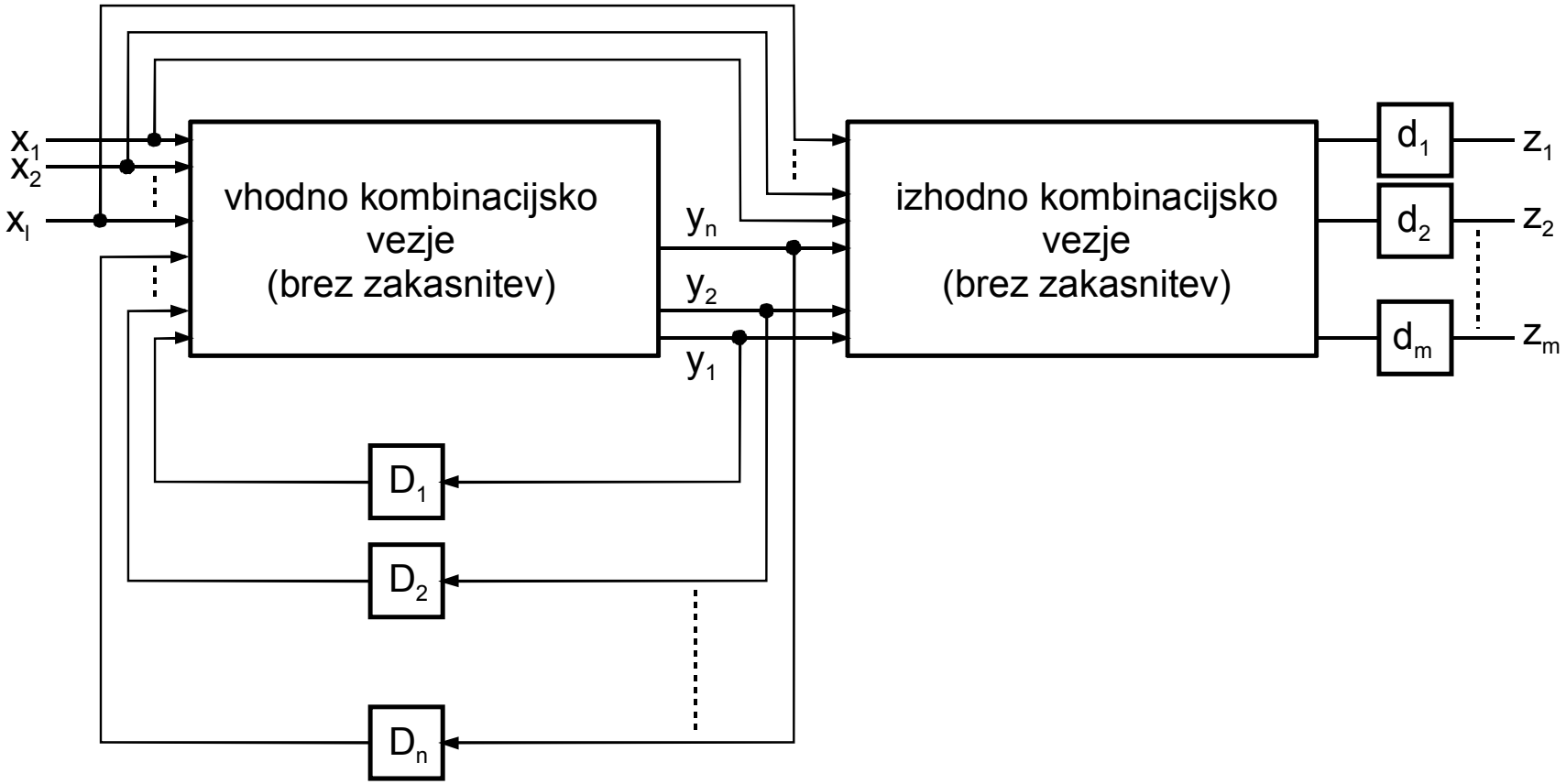
x_1x_2			
①^{00}	2	④^{01}	5
1	3	4	⑤^{11}

- Vsak par simbolov (lahko tudi več), ki ni obkrožen v stolpcu moramo nadomestiti z istim neobkroženim simbolom v spojeni vrstici
- Če je eden izmed simbolov obkrožen, ga je potrebno nadomestiti z istim obkroženim simbolom v spojeni vrstici
- Če se v nekem stolpcu nahajata obkrožen ali neobkrožen simbol in redundatno stanje, moramo v spojeni vrstici v tem stolpcu vnesti obkrožen simbol.
- Če so v istem stolpcu vseh vrstic, ki se spajajo sama redundatna stanja, potem je v tem stolpcu spojene vrstice tudi redundatno stanje.
- Izhode v spojeni vrstici lahko izpustimo, če se spajajo vrstice z različnimi izhodi.
- Izhode v spojeni vrstici lahko izpustimo, če se spajajo vrstice z različnimi izhodi. To informacijo še vedno lahko dobimo iz primitivne tabela prehajanja stanj. priporočljivo pa je , da jih pripišemo k stabilnim (totalnim stanjem) znotraj odgovarjajočega polja spojene vrstice.

10. 4 Zakasnitve v asinhronskih vezjih







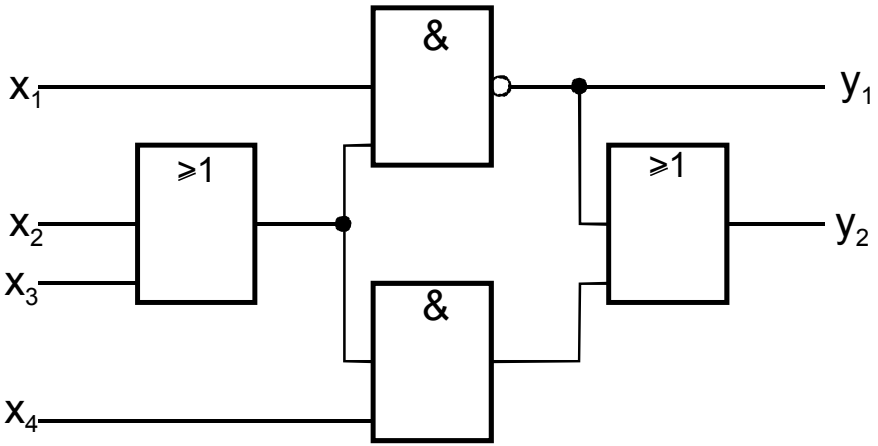
$$z_j(t+\Delta t_{j0}) = \Phi\left[x_1(t), x_2(t), \ldots, x_l(t); y_1(t+\Delta t_1), y_2(t+\Delta t_2), \ldots, y_n(t+\Delta t_n)\right]$$

Δt_{j0} - časovni interval v katerem se stabilizira j-ti izhod

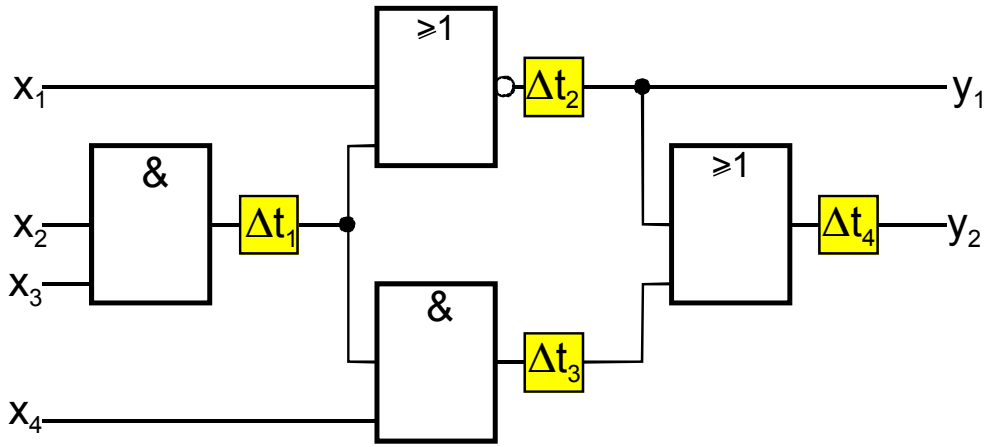
Končno izhodno stanje vektorja $\mathbf{z_1}$ lahko opazujemo šele po Δt^1_{min} za vzbujanjem $\mathbf{x_i}$, pri čemer mora veljati:

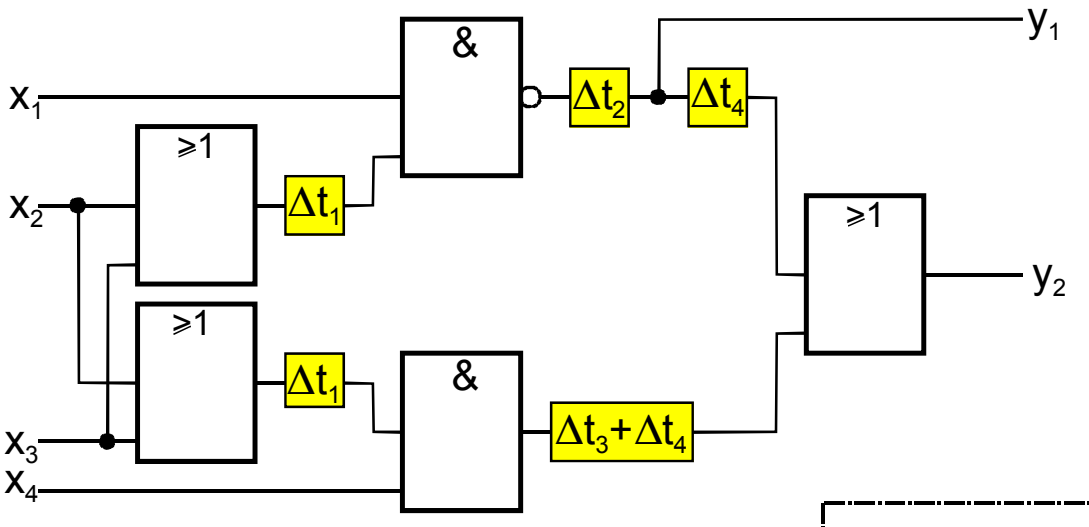
$$\Delta t^1_{min} \geq (\Delta t^0)_{jmax}$$

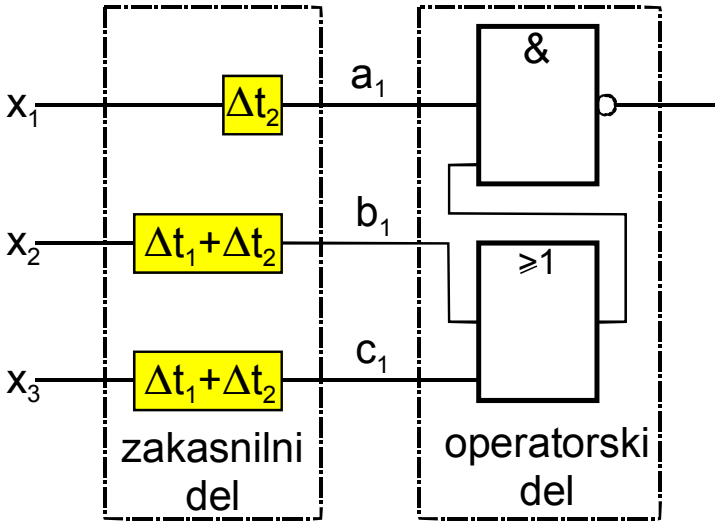
Primer kocentriranja zakasnitev:

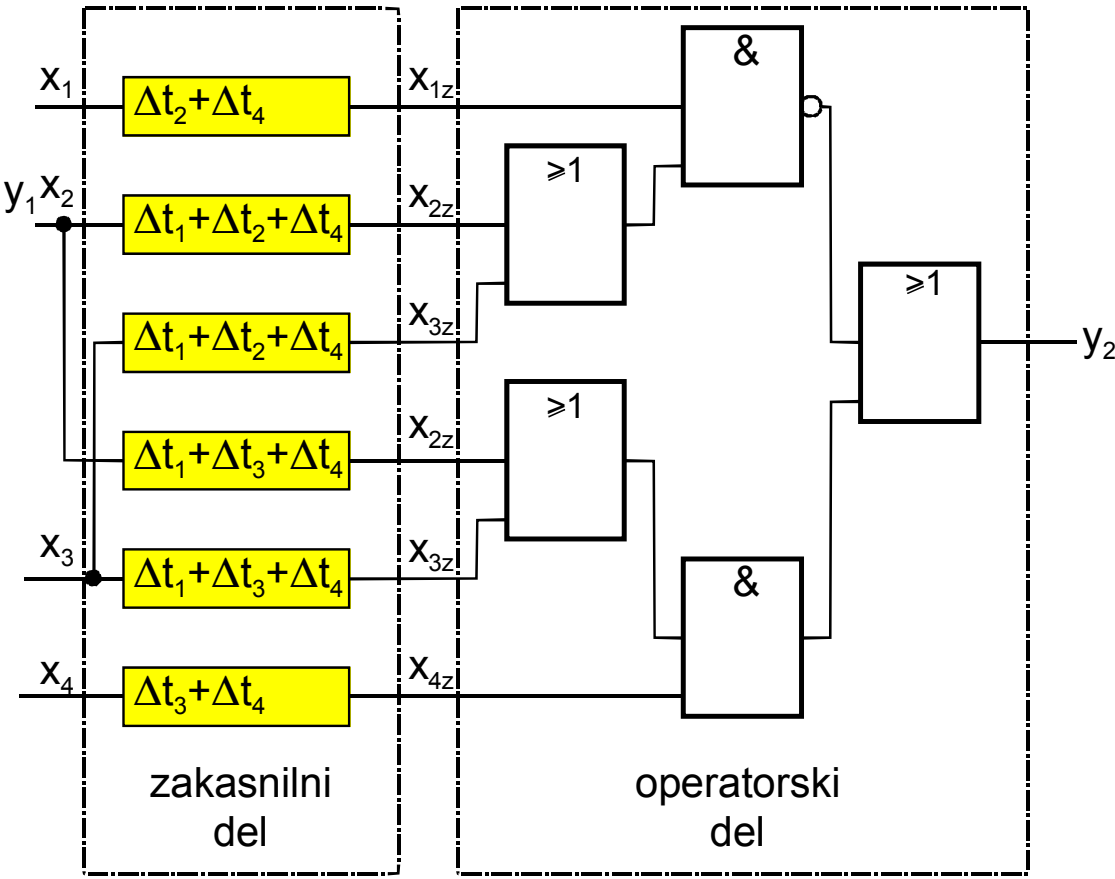


V tem vezju nastopajo sledeče zakasnitve:



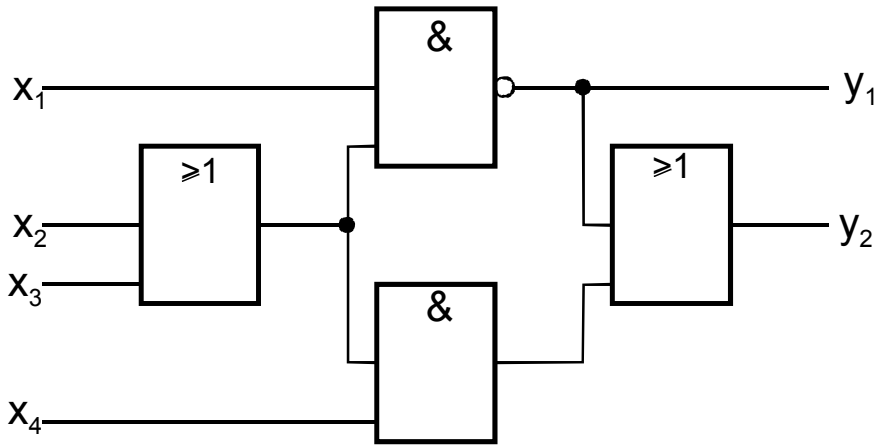

x1
x2
x3
x4
&
≥1
≥1
≥1
Δt1
Δt1
Δt2
Δt4
Δt3+Δt4
y1
y2


x1
x2
x3
Δt2
Δt1+Δt2
Δt1+Δt2
a1
b1
c1
&
≥1
zakasnilni del
operatorski del
y1


x1
x2
x3
x4
Δt2+Δt4
Δt1+Δt2+Δt4
Δt1+Δt2+Δt4
Δt1+Δt3+Δt4
Δt1+Δt3+Δt4
Δt3+Δt4
x1z
x2z
x3z
x2z
x3z
x4z
&
≥1
≥1
&
zakasnilni del
operatorski del
y2

$$\Delta t_1 = \Delta t_2 = \Delta t_3 = \Delta t_4 = \Delta t_0$$

Operatorski del določimo iz osnovnega vezja, ki ga tako obravnavamo idealno:



$$y_1 = \overline{x}_1 + \overline{x}_2\overline{x}_3$$

$$y_2 = y_1 + x_2x_4 + x_3x_4$$

Spremenljivke x_1, x_2, x_3, x_4 so elementi vhodnega vektorja:

$$\mathbf{x} = \{x_1, x_2, x_3, x_4\}$$

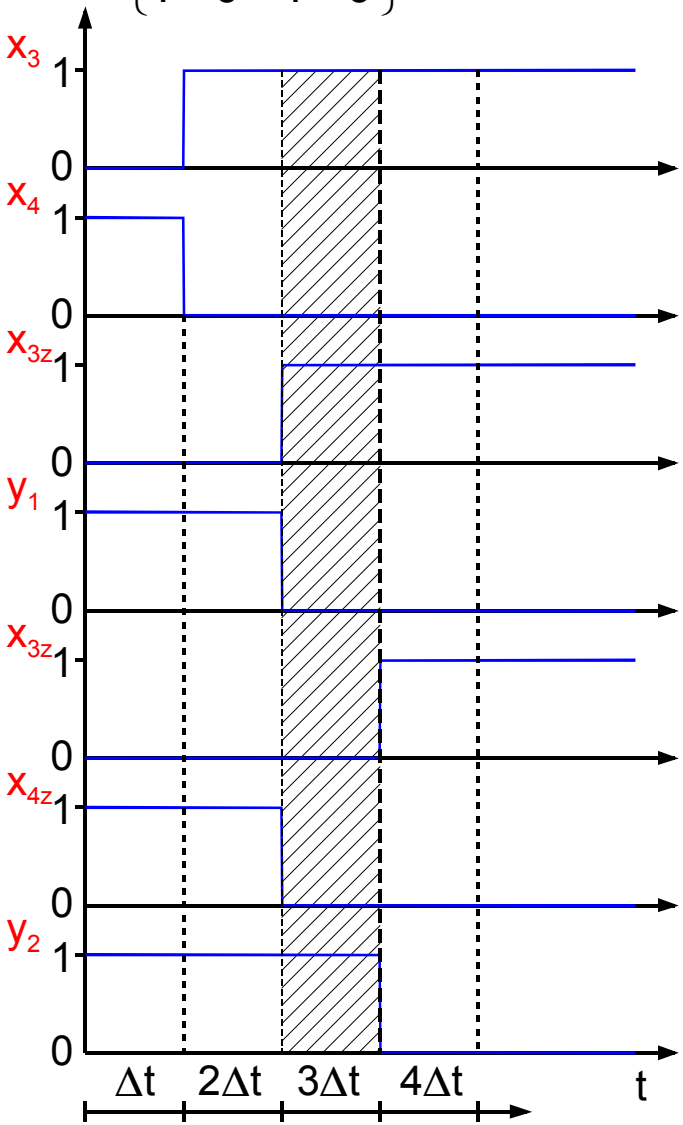
Vektor **x** lahko zavzame 16 (2^4) različnih vrednosti (v obliki urejenih kompozicij vhodnih stanj x_1, x_2, x_3, x_4), kar predstavlja 16 različnih vhodnih dogodkov.

$$\mathbf{x_0} \triangleq \begin{Bmatrix} \overline{x}_1 & \overline{x}_2 & \overline{x}_3 & \overline{x}_4 \\ 0 & 0 & 0 & 0 \end{Bmatrix}$$

Iz celotne množice: $\mathbf{X} = \{\mathbf{x_0}\ \mathbf{x_1}, \ldots, \mathbf{x_{15}}\}$ opazujmo kompoziciji predstavljeni z vektorjema: $\mathbf{x_9}$ in $\mathbf{x_{10}}$.

$$\mathbf{x}^9 = \begin{Bmatrix} x_1 & \overline{x}_2 & \overline{x}_3 & \overline{x}_4 \\ 1 & 0 & 0 & 1 \end{Bmatrix};\ y_1(\mathbf{x}_9) = 1,\ \ y_1(\mathbf{x}_{10}) = 0$$

$$\mathbf{x}^{10} = \begin{Bmatrix} x_1 & \overline{x}_2 & x_3 & \overline{x}_4 \\ 1 & 0 & 1 & 0 \end{Bmatrix};\ y_2(\mathbf{x}_9) = 1,\ \ y_2(\mathbf{x}_{10}) = 0$$



Pomembno je opozoriti, da je v času 3Δt izhod v “nedefiniranem” stanju.

V tem času je namreč $y_1 = 0$; $y_2 = 1$.

10. 5 Kodiranje notranjih stanj

Kodiranje notranjih stanj (KNS) je eno izmed najbolj kritičnih opravil v postopku sinteze asinhronskega avtomata.

Poudariti je potrebno, da se ta problem v temelju razlikuje od kodiranja pri sinhronskih vezjih.

Pri sinhronskih je bilo dovolj, da smo zadovoljili enačbo: k = ldr; oziroma $r = 2^k$.

Pri asinhronskih pa nastopi nova dimenzija kodiranja, od katere ni odvisna samo kompleksnost vezja, temveč tudi zanesljivost delovanja.

{s} \ {x}	x = 0	x = 1
1	①	2
2	3	②
3	③	4
4	1	④

Koda A

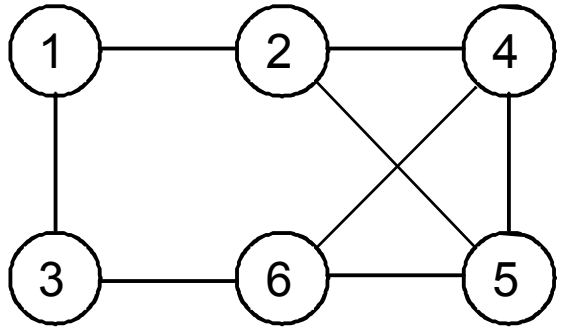
y_1	y_2
0	0
0	1
1	0
1	1

Koda B

y_1	y_2
0	0
0	1
1	1
1	0

Pri kodiranju notranjih stanj nam je lahko v veliko pomoč graf sosednih prehodov.

X / S	x_1x_2			
	00	01	11	10
1	①	2	3	①
2	4	②	②	-
3	1	③	③	-
4	④	5	④	-
5	6	⑤	2	-
6	⑥	3	4	⑥



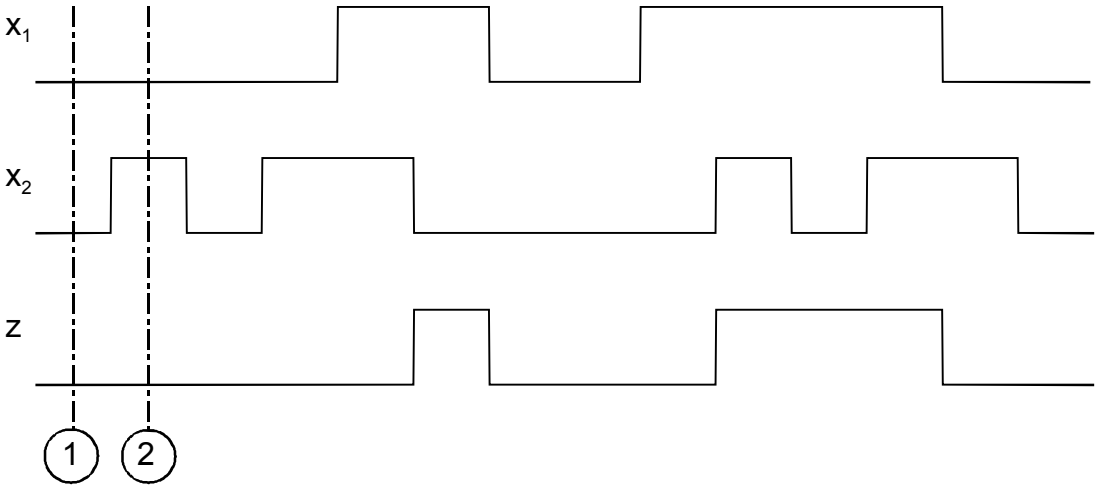
Stanja 2, 4, 5 morajo biti medsebojno sosedna!

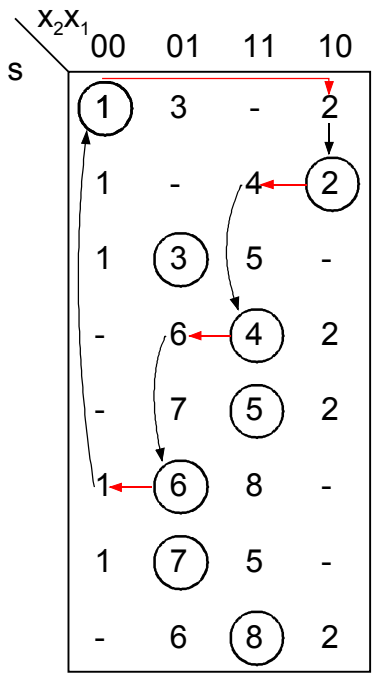
Isto velja za stanja 4, 5 in 6!

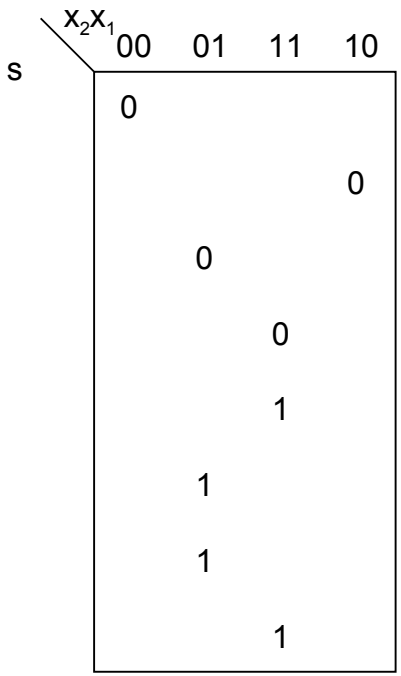
Zahteva je potrebna zato, da se lahko pri vseh označenih prehodih istočasno spreminja le posamična sekundarna vhodna spremenljivka.

Pri treh sosednih stanjih pa je nemogoče binarno tako kodirati, da bi se vsak par razlikoval samo za en bit.

Primer sinteze asinhronskega avtomata:







Določitev maksimalnih kompatibilnosti:

(12), (13), (24), (5678)

(13), (24), (5678)

q		$\Delta^1 q$				z			
		00	01	11	10	00	01	11	10
(1,3)	a	a	a	c	b	0	0	-	0
(2,4)	b	a	c	b	b	0	-	0	0
(5,6,7,8)	c	a	c	c	b	-	1	1	-

Kodiranje notranjih stanj:

Obstojajo prehodi med vsemi tremi stanji: a – b, a – c in b – c !

Potrebujemo torej dve notranji spremenljivki: y_1 in y_2.

y_2y_1	q	Δ^1q			
		00	01	11	10
00	(a)	(a)	(a)	c	b
01	(b)	a	c	(b)	(b)
11	(c)	a	(c)	(c)	b
10	(c)	a	(c)	(c)	b

y_2y_1	q	Δ^1q			
		00	01	11	10
00	(a)	(a)	(a)	c	b
01	(b)	a	c	(b)	(b)
11	(c)	a	(c)	(c)	b
10	c	a	-	c	-

Če izberemo drugi način kodiranja dobimo za Y_1 in Y_2 naslednji Karnaughjev diagram:

y_2y_1: \ x_2x_1:	00	01	11	10
00	(00)	(00)	10	11
01	00	11	(01)	(01)
11	00	(11)	(11)	01
10	00	-	11	-

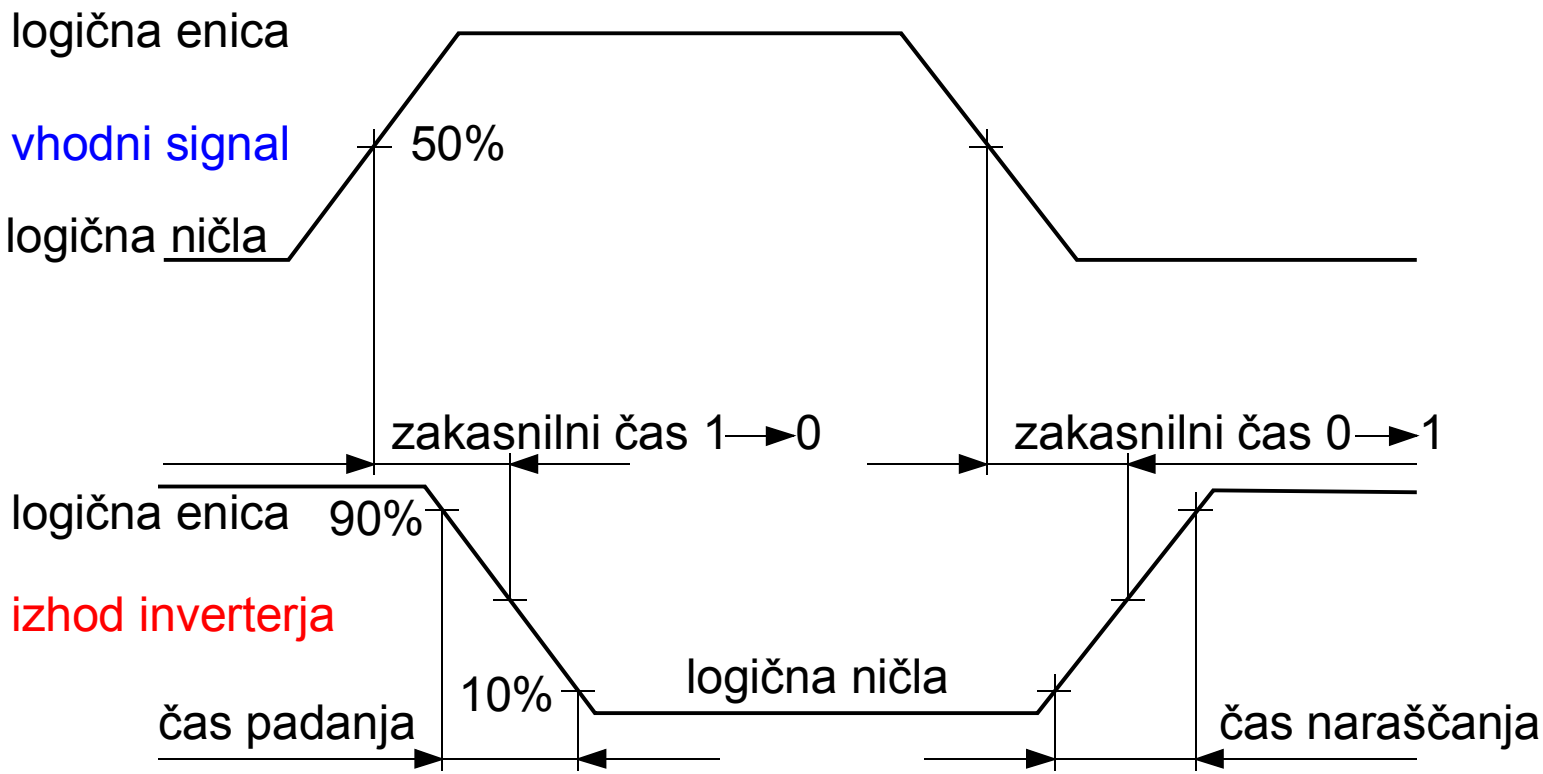
Za izhodno funkcijo pa:

y_2y_1: \ x_2x_1:	00	01	11	10
00	0	0		0
01	0		0	0
11		1	1	
10				

10. 6 Hazardni prehodi

10. 6. 1 Hazardni prehodi v kombinacijskih vezjih

V fizikalnem svetu vedno nastopajo zakasnitve med trenutkom, ko se spremeni vhod v nek element in trenutkom, ko se pojavi ustrezna sprememba tudi na izhodu.



Zakasnitev lahko izvira tudi iz povezav med elementi, če so te daljše ali hitrosti velike

Čas prehodnega pojava ni vedno le preprost seštevek posameznih zakasnilnih časov

Obstoja lahko več vzporednih poti, zato se prehodni pojav lahko odvija po več poteh hkrati.

Vsi logični izrazi, funkcijske odvisnosti, izjavnostne tabele in podobno veljajo le za stacionarno stanje!!!

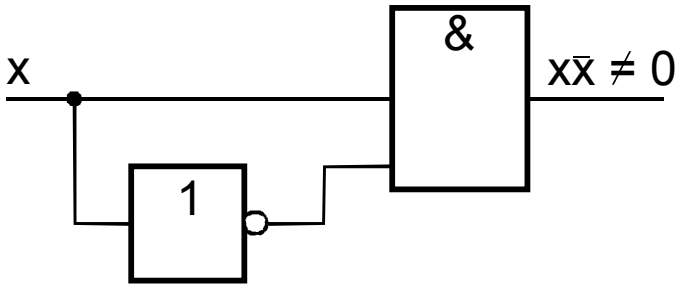
Zakasnitve elementov se lahko močno razlikujejo tudi znotraj iste družine

Samo logična analiza in sinteza ne zadoščata več, potrebna je tudi analiza prehodnih pojavov!

Definicija hazarda:

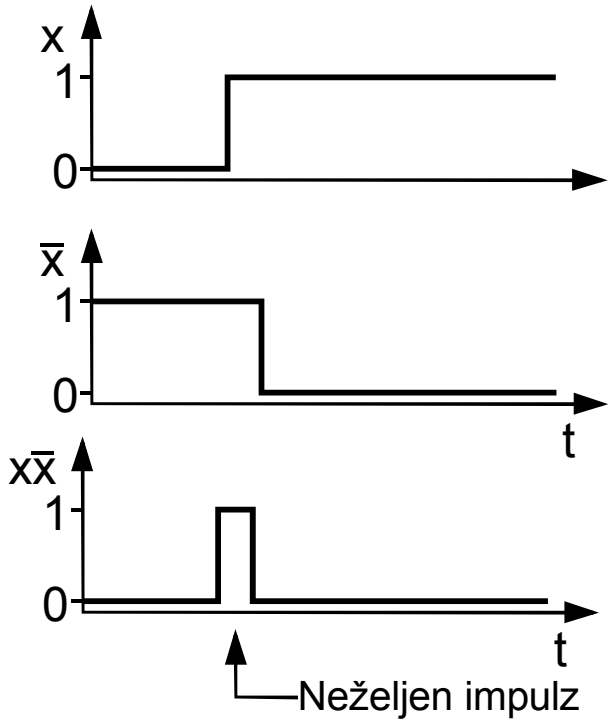
Hazard je dejansko ali potencialno nepravilno delovanje vezja zaradi prisotnosti zakasnitev v vezju

Da do hazarda pride zelo hitro lahko pokažemo na zelo enostavnem primeru invertiranja neodvisne spremenljivke:



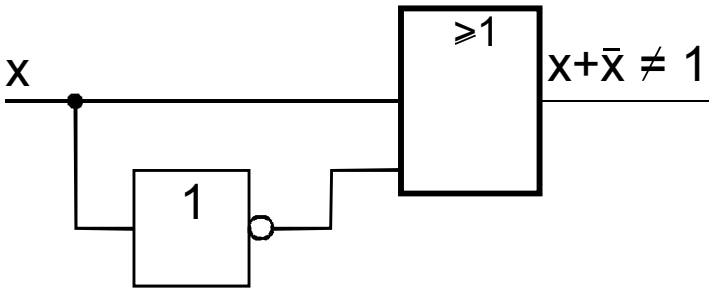
V času prehodnega pojava tako ne veljata niti postulata Boole-ove algebre p_4 in p_4'

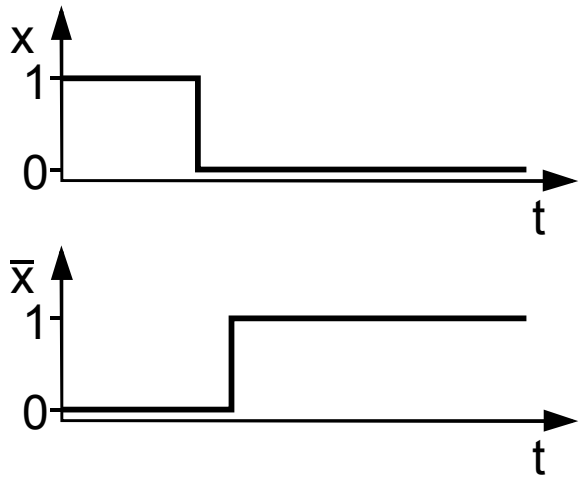
$$x\bar{x} = 0$$



In tudi ne $x + \bar{x} = 1$

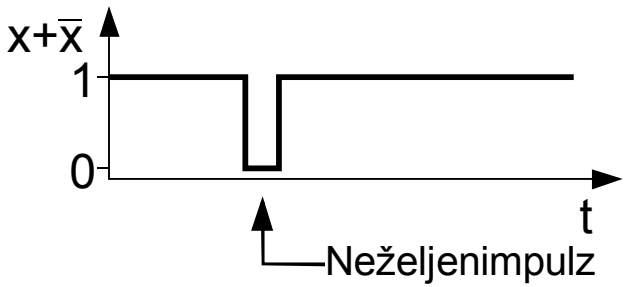
Naslednje vezje bi moralo imeti enico na izhodu ne glede na vhod





Vendar temu ni tako!

Zato je potrebno še enkrat poudariti, da logični zapisi Boole-ovih funkcij ne zajamejo časovnega dogajanja.



Umestno pa je vprašanje ali hazarden prehod pomeni nepravilno delovanje

Odgovor na to vprašanje je da in ne!!! Odvisen je od funkcije, ki jo vezje opravlja.

Podrobnejše opazovanje teh pojavov nam odkrije, da obstoja več vrst hazarda:

Statični hazard – dva prehoda brez logične spremembe izhoda

Glede na stacionarno stanje nastopata:

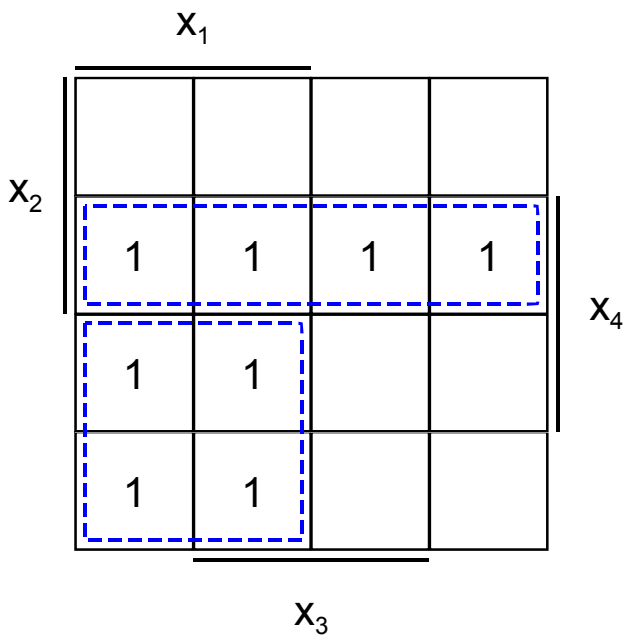
Statični hazard enice

Statični hazard ničle

Katerega koli od teh hazardov lahko povzroči sprememba ene same vhodne spremenljivke ali pa več spremenljivk hkrati.

$f = x_1\bar{x}_2 + x_2x_4$

$x_1 = x_4 = 1$ je $f = 1$ ne glede na x_2



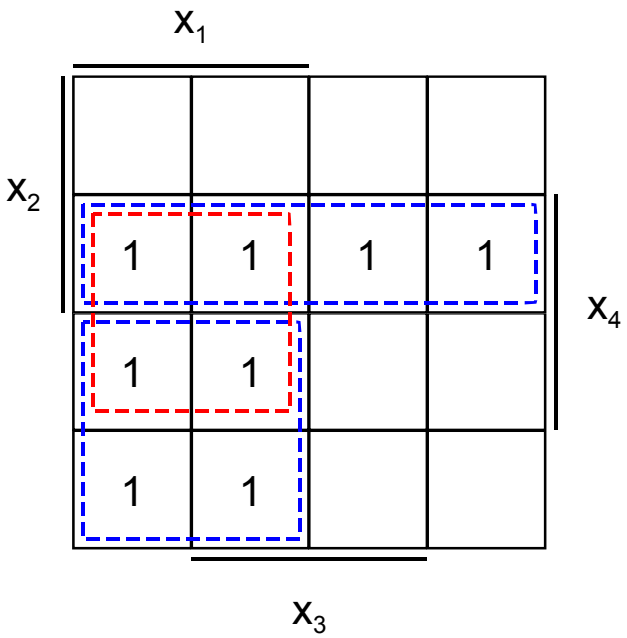
V prehodnem času nastopa:

$x_2 + \bar{x}_2 \neq 1$

Za odpravljanje statičnega hazarda imamo dve možnosti

- logično odpravljanje
- odpravljanje z dodajanjem zakasnitev

Logično odpravljanje hazarda bi v prejšnjem primeru pomenilo dodajanje nove konjunkcije:



V splošnem torej velja, da minimizacija ustvarja potencialne možnosti različnih hazardov.

Da jih preprečimo moramo narediti vse konjunkcije oz. minterme medsebojno odvisne.

To pa je neka nova minimalna oblika – tista, ki ne dovoljuje hazardnih prehodov!!

V literaturi lahko najdemo metode za takšno minimizacijo – Mc Cluskey.

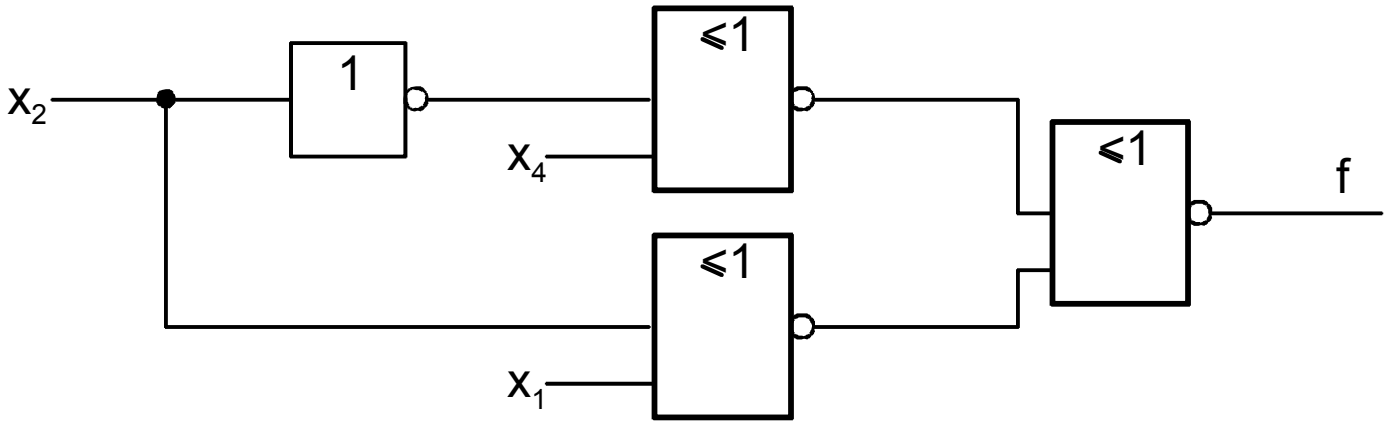
Vendar je potrebno opozoriti, da vse te metode temeljijo na disjunktivni obliki, ta pa v primeru uporabe NOR elementov ni optimalna.

$$f = x_1\overline{x}_2 + x_2x_4 + x_1x_4$$

Pripadajoča konjunktivna oblika je:

$$f = (\overline{x}_2 + x_4)(x_1 + x_2)$$

Realizacija z NOR pa naslednja:



Iz analize simboličnega diagrama lahko ugotovimo, da bo sedaj mogoč hazardni prehod pri stanju vhoda $x_1 = x_4 = 0$.

Prehod x_2 iz nič na ena bo povzročil kratkotrajno enico na izhodu.

V tem primeru gre za hazardni prehod pri ničli !!

V nasprotju s statičnim hazardom, pri katerem se stacionarno stanje izhoda ne spreminja, se pri dinamičnem stacionarno stanje zamenja.

Pri tej vrsti hazarda nastopajo zato najmanj tri spremembe izhoda.

Sprememb je lahko tudi več, vendar njihovo število je vedno liho.

Sproži ga lahko sprememba ene same spremenljivke, ali pa sočasna sprememba večih vhodnih spremenljivk.

Posledica takšnih hazardnih prehodov je lahko sprožitev ali preklop v vezju, ki je nanj priključeno.

Posebej nevarno je če med tema vezjema obstoja povratna zveza.

Poglobljena analiza dogajanja med dinamičnimi hazardnimi prehodi nas pripelje do spoznanja, da so za nastanek dinamičnega hazarda potrebne najmanj tri različne poti z različnim zakasnilnim časom za isto vhodno spremenljivko.

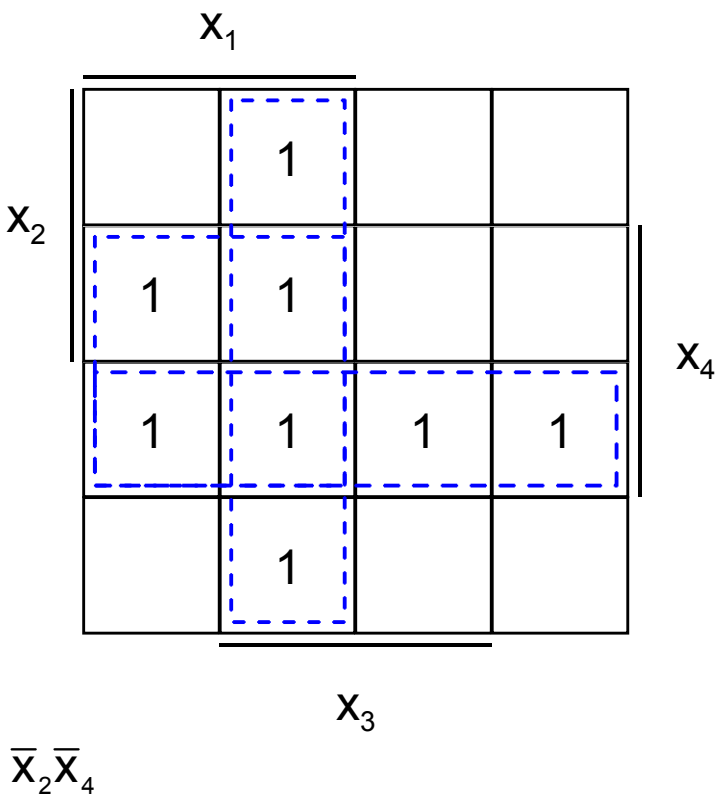
Dinamični hazard je v splošnem posledica:

- konjunktivne oblike realizacije
- predolgih povezav pri ekstremno hitrih vezjih

Za dinamični hazard so še zlasti občutljive NOR/NAND izvedbe – inverter na izhodu.

Zato si dogajanje oglejmo kar na obravnavanem primeru funkcije.

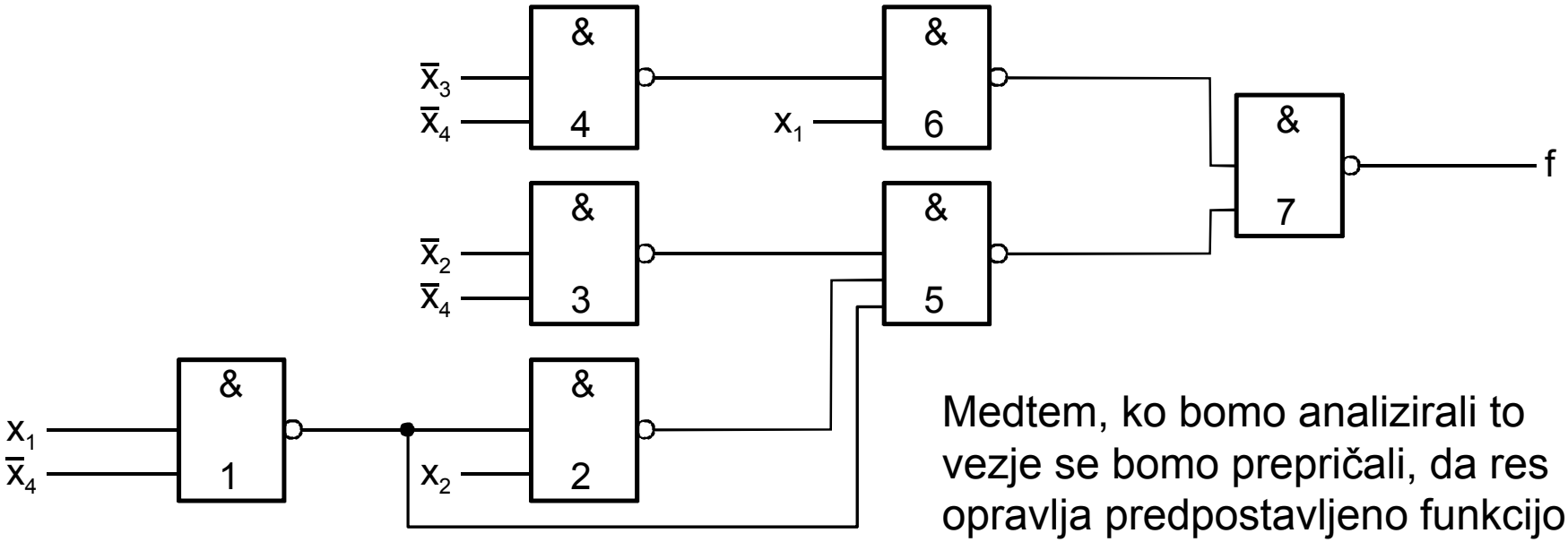
$$f = \overline{x}_2 x_4 + x_1 x_3 + x_1 x_4$$



To konjunkcijo “želimo” izkoristiti tudi pri tej funkciji

Takšna realizacija sicer ni dobra, se pa v praksi podobne izvedbe, ki so potem viri dinamičnega hazarda rade pojavljajo.

Zato si je vredno primer podrobno analizirati !



Medtem, ko bomo analizirali to vezje se bomo prepričali, da res opravlja predpostavljeno funkcijo

$$f = \overline{\left\{\overline{\left[\overline{(x_1\bar{x}_4)x_2}\right]\left[\overline{(x_1\bar{x}_4)}\right]\left[\overline{(\bar{x}_2\bar{x}_4)}\right]}\right\}\left\{\left[\overline{(\bar{x}_3\bar{x}_4)x_1}\right]\right\}}$$

$$f = \left[\overline{\overline{(x_1\bar{x}_4)x_2}}\right]\left[\overline{(x_1\bar{x}_4)}\right]\left[\overline{(\bar{x}_2\bar{x}_4)}\right] + \left[\overline{\overline{(\bar{x}_3\bar{x}_4)x_1}}\right]$$

$$f = \left[(x_1\bar{x}_4) + \bar{x}_2\right]\ \left[\bar{x}_1 + x_4\right]\ \left[x_2 + x_4\right] + x_1x_4 + x_1x_3$$

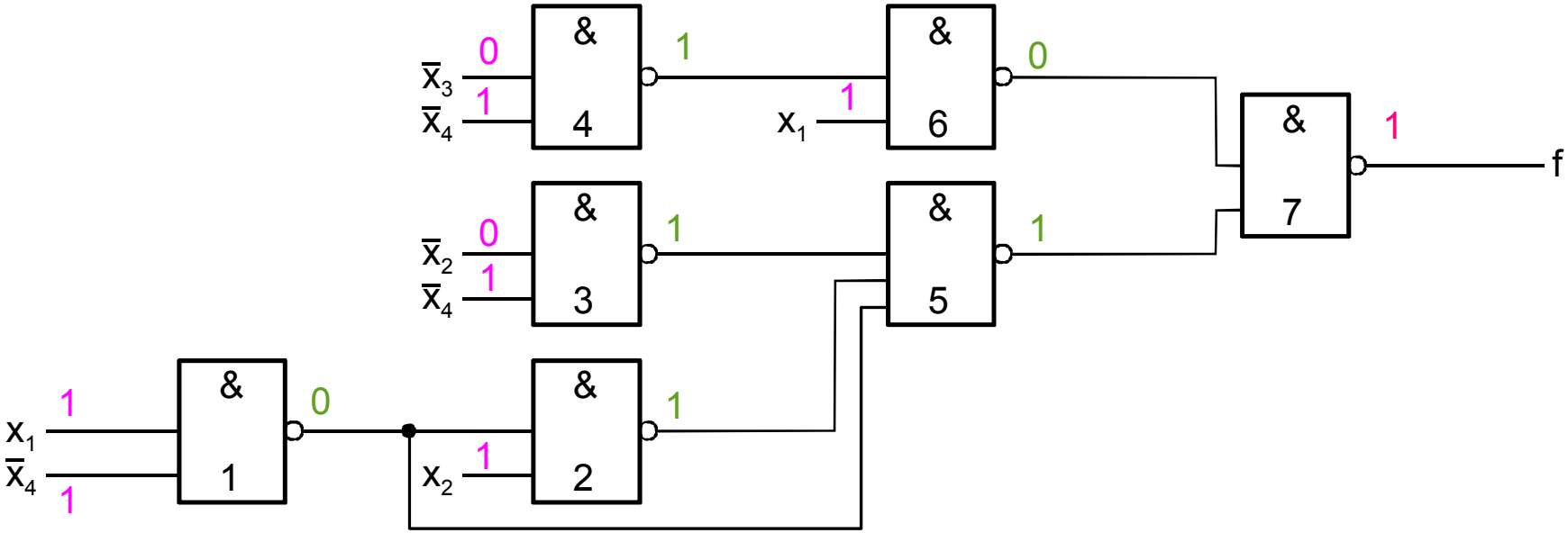
$$f = (x_1\bar{x}_4x_4 + x_1\bar{x}_1x_4 + \bar{x}_2x_4 + \bar{x}_1\bar{x}_2)(x_2 + x_4) + x_1x_3 + x_1x_4$$

$$f = \bar{x}_2x_4 + x_2\bar{x}_2x_4 + \bar{x}_1\bar{x}_2x_4 + \bar{x}_1\bar{x}_2x_2 + x_1x_4 + x_1x_3$$

Če to še dalje poenostavimo pridemo do izhodiščne oblike:

$$f = \bar{x}_2x_4 + x_1x_3 + x_1x_4$$

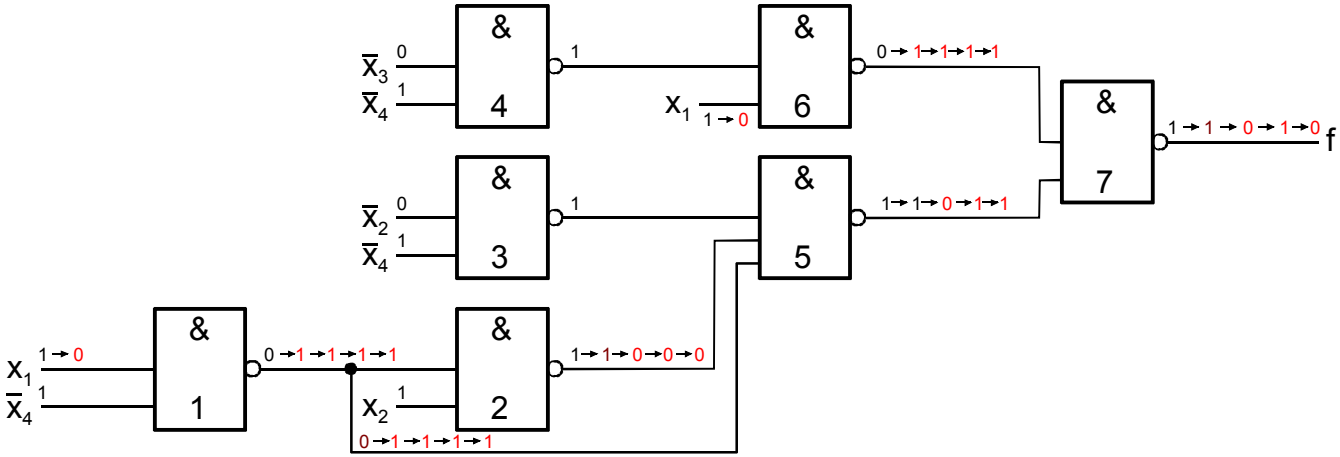
Predpostavimo stacionarno stanje: $x_1 = x_2 = x_3 = 1$ in $x_4 = 0$

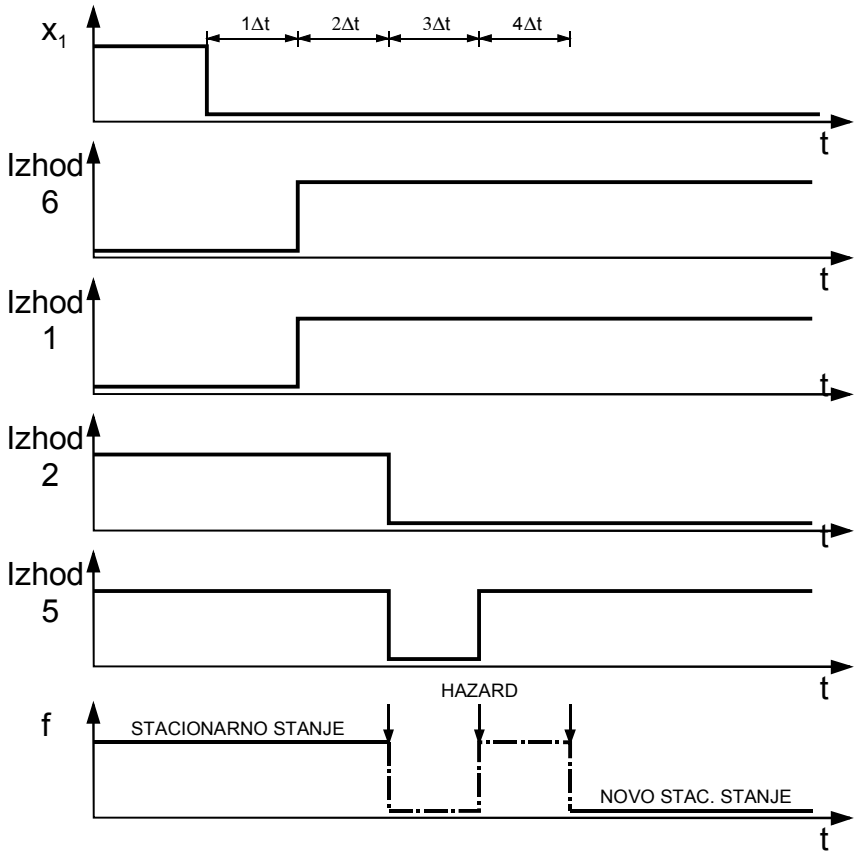


V naslednjem trenutku naj se x_1 spremeni iz 1 v 0 in tam ostane

V analizi uporabimo tipične logične lastnosti uporabljenih elementov

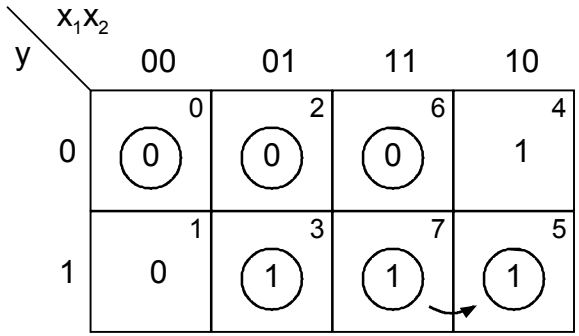
Pri NAND je to pojav “0” na vhodu, ki spremeni izhod iz 0 v 1


&
4
&
6
&
7
&
3
&
5
&
1
&
2
$\overline{x}_3$
$\overline{x}_4$
$\overline{x}_2$
x_1
x_2
0 → 1 → 1 → 1 → 1
1 → 1 → 0 → 1 → 0
f
1 → 1 → 0 → 1 → 1
1 → 0
0 → 1 → 1 → 1 → 1
1 → 1 → 0 → 0 → 0


x_1
1Δt
2Δt
3Δt
4Δt
t
Izhod 6
Izhod 1
Izhod 2
Izhod 5
HAZARD
f
STACIONARNO STANJE
NOVO STAC. STANJE

10. 6. 2 Hazard v asinhronskih vezjih

$$Y = x_1\bar{x}_2 + x_2y$$



Δt – zakasnitev posameznega elementa

Začetno stanje:

$x_1 = 1$

$x_2 = 1$

$y = 1$

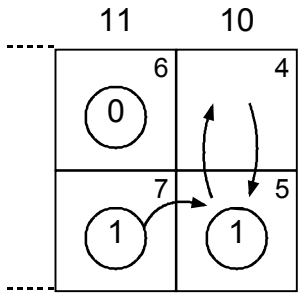
Predpostavimo: $t = 0^-$ x_1x_2y : 1,1,1 – minterm 7 v vzbujalni tabeli

Zato : $Y = x_1\bar{x}_2 + x_2y = 1 \bullet 0 + 1 \bullet 1 = 1$

Potem: $t = 0 – x_2$ preide iz $1 \rightarrow 0$ – pomik na minterm 5

Potem: $t = \Delta t_1 – Y = 1 \cdot 0 + 0 \cdot 1 = 0$ – HAZARD - preskok na 4

Potem: $t = \Delta t_1 + \Delta t_2 – Y = 1 \cdot 1 + 0 \cdot 1 = 1$

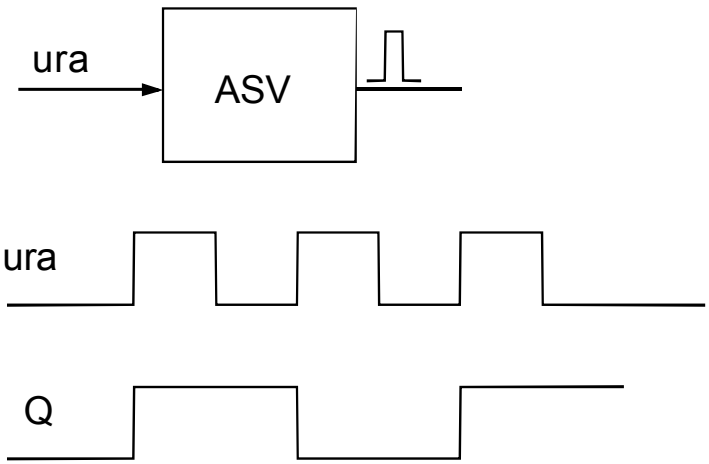


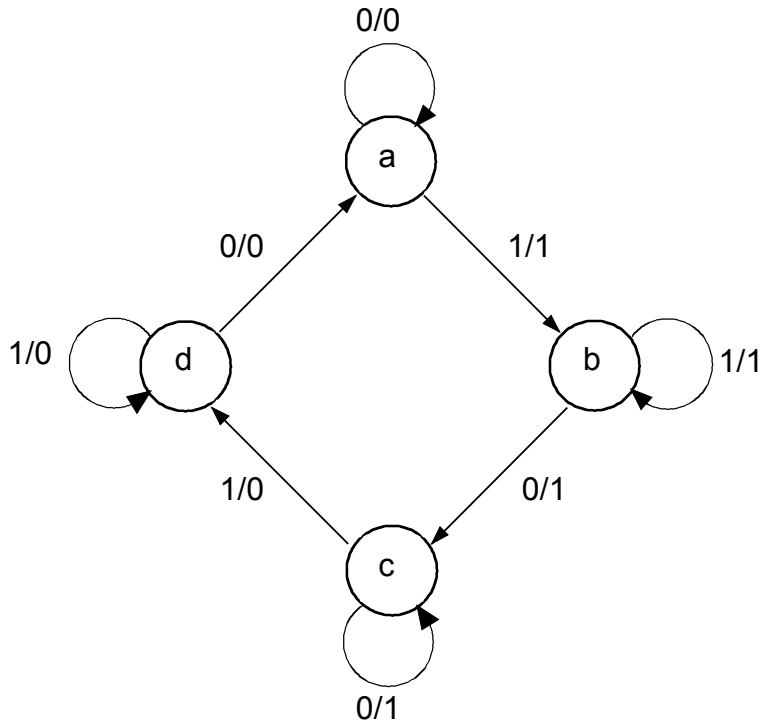
Bistveni ali vgrajeni hazard (Essential hazard)

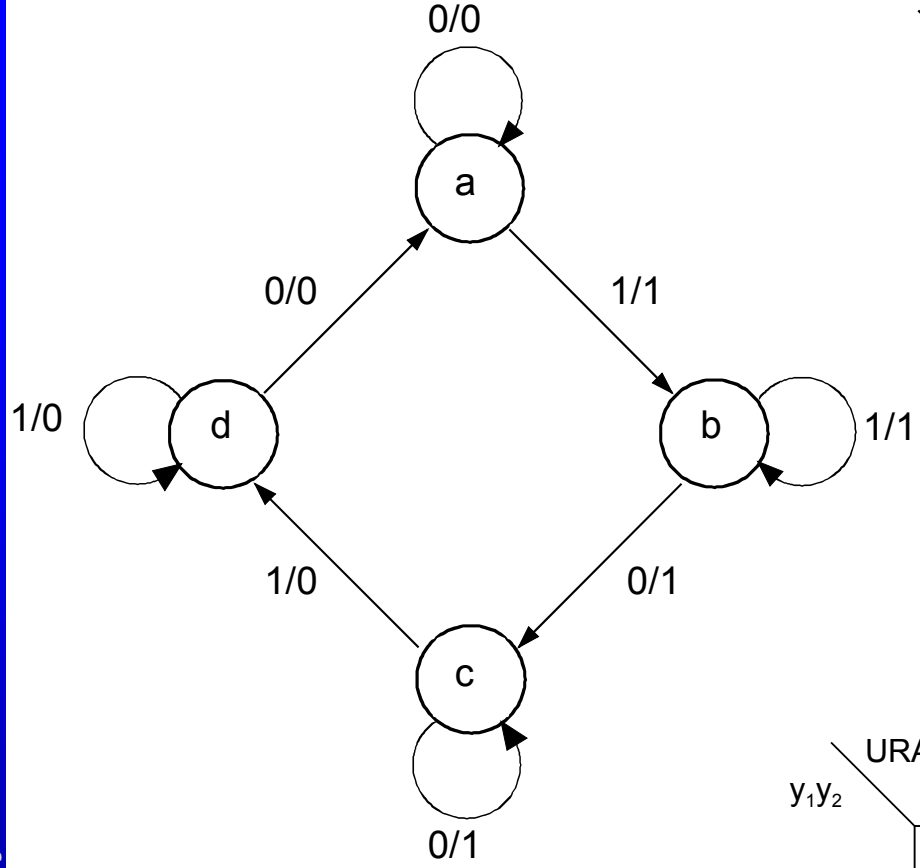
- Pojavlja se le v vezjih z dvema ali več povratnimi zvezami
- Če obstoja je vezan na oboje; to je na zakasnitve in na vhodne specifikacije o delovanju vezja (design specifications)

To pomeni, da bodo določene zahteve za delovanje sekvenčnega vezja že potencialno vsebovale možnost za pojav vgrajenega hazarda.

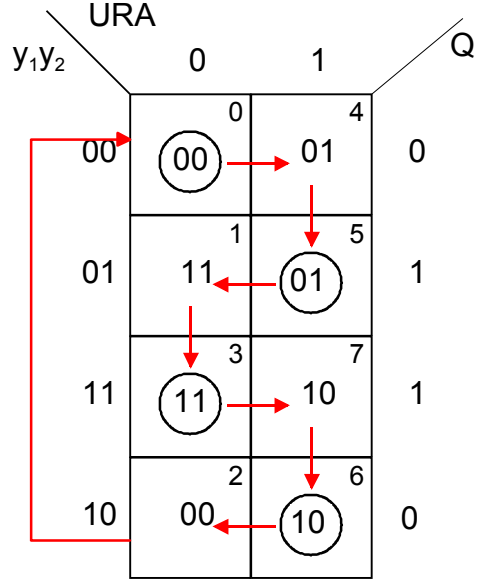
Tipična vezja, ki vsebujejo možnost vgrajenega hazarda so vezja z lastnostmi klecnega stikala (Toggle switch)



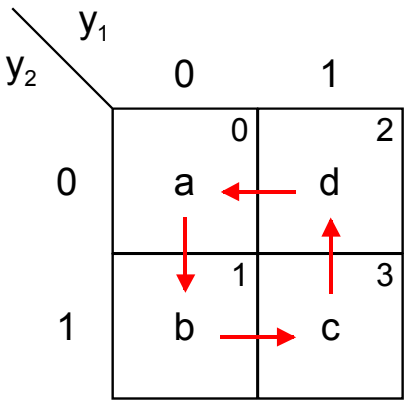


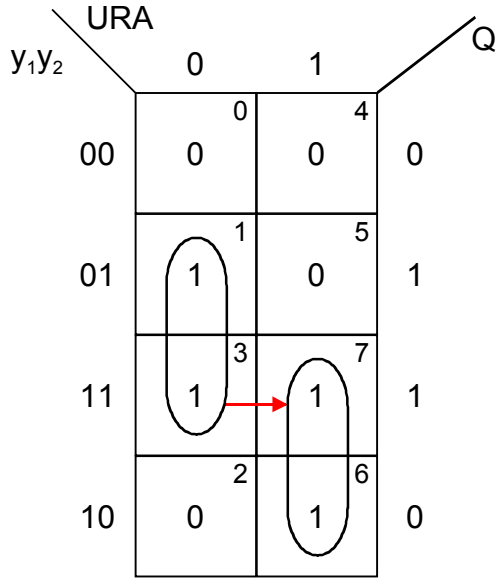


URA	0	1	Q
a	b		0
c	b		1
c	d		1
a	d		0

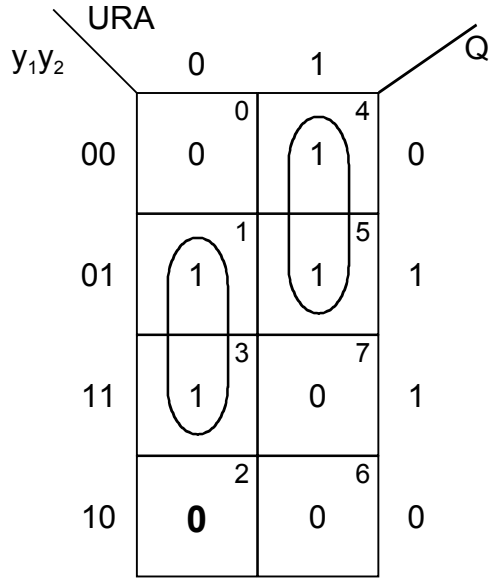


Določitev vzbujalnih funkcij:





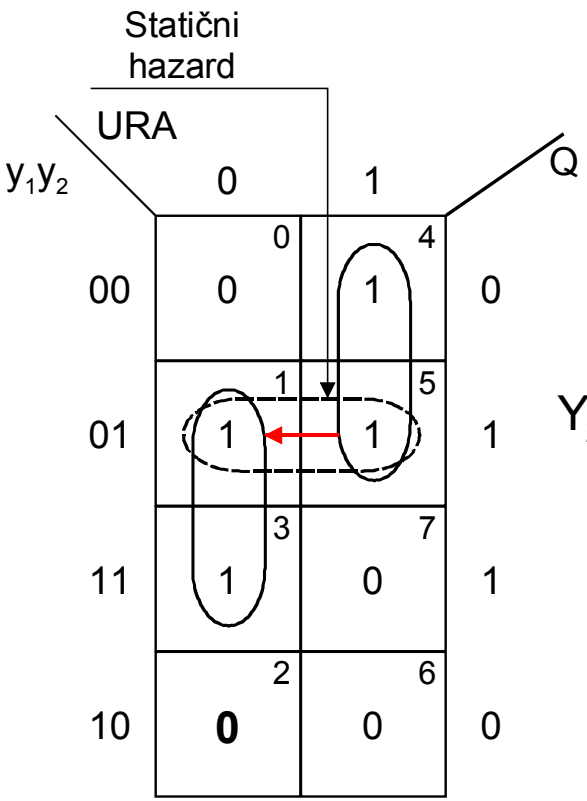
$$Y_1 = \overline{URA}y_2 + URAy_1$$



$$Y_2 = \overline{URA}y_2 + URA\overline{y}_1$$

Statični hazardni prehod nastopi pri prehodu iz b v c: $Y_2 = \overline{URA}y_2 + URA\bar{y}_1$

URA = 0 – y_2 pade za trenutek na nič, potem pa se vrne na ena.



Analiza vgrajenega hazarda pokaže, da pri enakih zakasnitvah elementov:

$\Delta t_1 = \Delta t_2 = \Delta t_3 = ... = \Delta t_7 = \Delta t$ vezje deluje brez napake.

Bistven ali vgrajeni hazard se pokaže šele, če Δt_1 povečamo na $7\Delta t$. Takrat pride do tega, da se vezje pri prehodu: **a**→b→**b**→d→**d** sploh ne ustavi v stanju **b,** ki je stabilno stanje vezja, temveč zaradi velike zakasnitve y_1 nadaljuje svoj cikel.

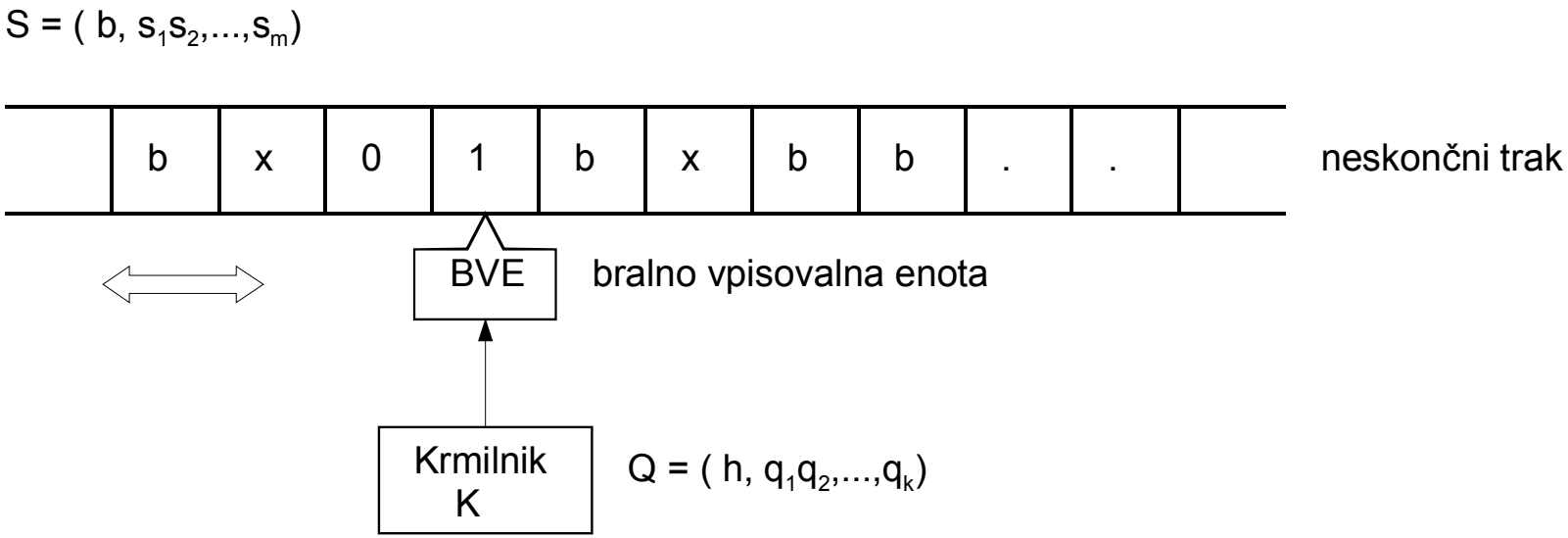
10. 7 Turingov stroj

Gre za teoretični model računalnika, na katerem je mogoče razčleniti logične operacije, ki vodijo k rešitvi nekega problema.

Zaporedju ukazov, ki povzročijo te logične operacije, pravimo algoritem.

Za analizo algoritmov potrebujemo formalizem, ki je dovolj preprost za razčlenitev, pa vendarle dovolj splošen, da lahko opiše vse procese, ki bi jih lahko še imenovali algoritem.

1. pomnilnik, ki lahko sprejema, shranjuje in oddaja podatke;
2. enoto, ki zmore vpisovati v pomnilnik in iz njega tudi brati;
3. krmilno enoto, ki enolično določi posamezne faze celotnega procesa.



Krmilnik upravlja Turingov stroj z eno izmed končnega števila peterk, ki predstavljajo program.

Vsaka peterka ima naslednje elemente:

1. sedanje stanje
2. vhodni simbol (branje)
3. izhodni simbol (pisanje)
4. naslednje stanje
5. premik bralno-pisalne enote.

Sedanje stanje in simbol nad bralno-pisalno enoto določujeta nadaljnje obnašanje stroja.

Peterke Turingovega stroja bodo torej imele naslednjo obliko:

qi, sj, sk, ql, M

Za opis delovanja potrebujemo:

1. Končno abecedo S na traku, vključno s praznim prostorom b

$S = (b, s_1, s_2 \ldots s_m)$

2. Končno število notranjih stanj Q, vključno s stanjem stoj, h.

$Q = (h, q1, q2 \ldots qk)$

3. Program, določen s končno množico peterk v obliki

qi sj sk ql M,

kjer so qi, q1 Q, sj, sk S, M pa je L (premik na levo) ali R (premik na desno).

Nemogoče je, da bi obstajali dve različni peterki:

$q_{i1}\, s_{j1}\, s_{k1}\, q_{l1}\, M_1$ ter $q_{i2}\, s_{j2}\, s_{k2}\, q_{l2}\, M_2$

pri

$q_{i1} = q_{i2}$ in $s_{j1} = s_{j2}$

4. Trak naj ima končno število ne-praznih kvadratov, ki vsebujejo črke abecede S.

5. Začetno stanje, ki je element množice Q.

6. Začetni položaj bralno pisalne enote.

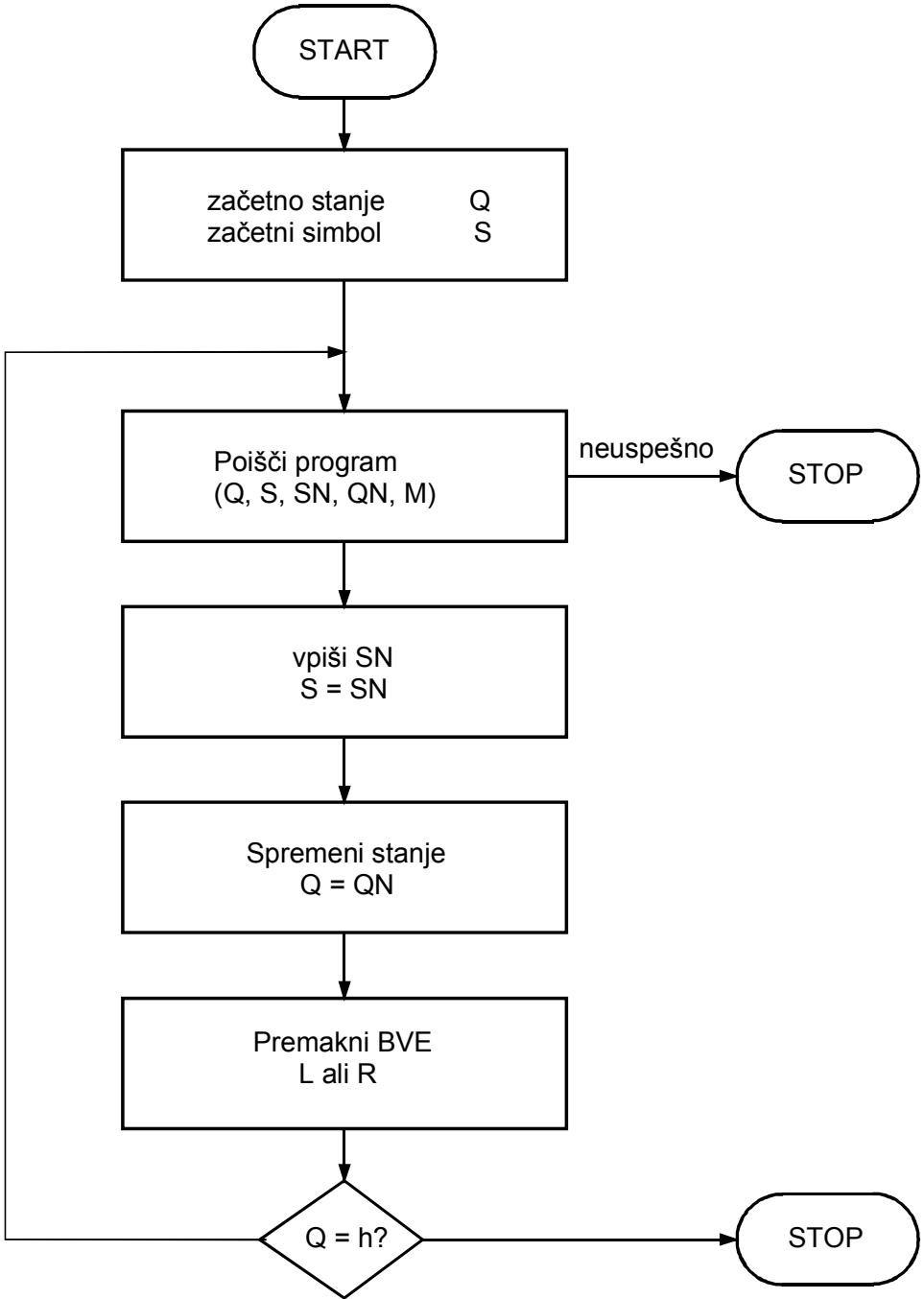
Vsak ukaz lahko razdelimo na tri korake:

1. sprememba vzorčnega simbola na traku

2. sprememba stanja

3. premik bralno-pisalne enoto na levo ali desno.

Stroj se ustavi, ko naleti na stanje h. Rezultat računanja je tedaj predstavljen z vsebino, vpisano na traku.


START
začetno stanje Q
začetni simbol S
Poišči program
(Q, S, SN, QN, M)
neuspešno
STOP
vpiši SN
S = SN
Spremeni stanje
Q = QN
Premakni BVE
L ali R
Q = h?
STOP

Določanje parnosti enic na traku

P – parno število enic

N – neparno število enic

Podatki stroja so:

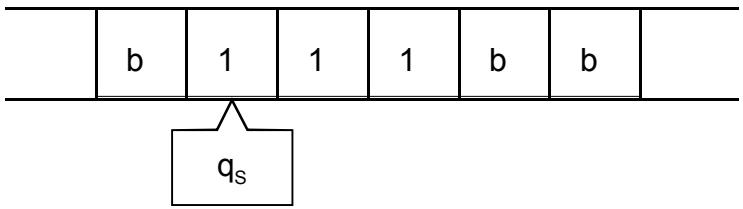
S = (b, 1, N, P)

Q = (h, ql, qs)

ql - liho, qs - sodo število

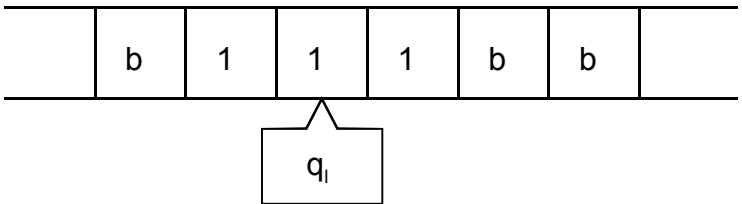
Program: (ql 1 1 qs R, ql b N h R, qs 1 1 ql R, qs b P h R).

1. qs naj bo začetno stanje
2. glava BVE je na najbolj levi strani
3. na traku je vpisano zaporedje enic, ki ga zaključujejo znaki b



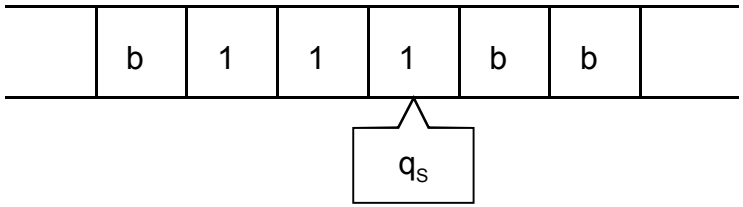
začetno stanje

Stroj začne teči s peterko (qs 1 1, ql R)



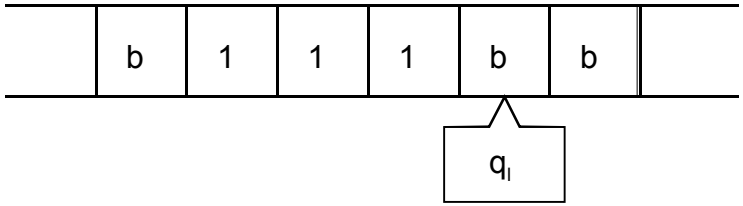
po izvršitvi q_S 11 q_l R

Nadaljuje s peterko (ql 1 1 qs R)



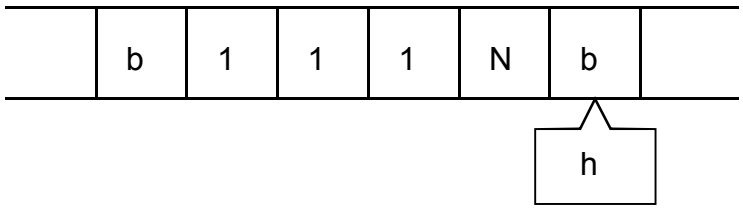
po izvršitvi q_l 11 q_S R

V tretjem premiku naleti na znak b



po izvršitvi q_S 11 q_l R

Zato steče peterka (ql b N h R), ki natisne N in stroj se ustavi.



po izvršitvi q_l bN h R