PREKLOPNA VEZJA
izpiski za ustni izpit

Šolsko leto	2008 / 2009
Izvajalec	Stanislav Reberšek
Avtor dokumenta	Matic Golob
Sodelavci	Uroš Pirnat

UREJANJE DOKUMENTA

VERZIJA	01	**REVIZIJA**	02
DATUM	29. 3. 2009		
ZADNJI POPRAVLJAL	/		
PREGLEDAL	**/**		

OPOMBE

POPRAVKI

29. 3. 2009 – B&W, razteg čez A4, kontrast

PV

1. ŠTEVILSKI SISTEMI IN KODE

Številski sistemi

Binarni sistem:

$$X = a_m 2^m + a_{m-1} 2^{m-1} + \dots + a_1 2^1 + a_0 + a_{-1} 2^{-1} + \dots a_{-m} 2^{-m} = \sum_{i=-m}^{m} a_i 2^i$$

↳ vrednost števila v desetiškem sistemu

$a_i \in \{0, 1\}$

Heksadecimalni sistem:

$$X = \sum_{i=-m}^{m} a_i 16^i \qquad a_i \in \{0,1,\dots 9, A, \dots F\}$$

↳ vred. št. v (10) ● sist.

Pretvarjanje iz desetiškega v sistem z osnovo q:
- št. razdelimo na celi del in ulomčni del
- pretvorba celega dela:

N = naravno št. z osnovo 10

$\frac{N}{q} = S_0 + \frac{a_0}{q} \Rightarrow a_0$

$\frac{S_0}{q} = S_1 + \frac{a_1}{q} \Rightarrow a_1$

S_1 ← celi del; $\frac{a_1}{q}$ → ostanek

dokler ne delimo vseh koeficientov (do $S_m = 0$)

Primer: $N_{10} = 41$, $q = 2$

$\frac{41}{2} = 20 + \frac{1}{2}$ $a_0 = 1$

$\frac{20}{2} = 10 + \frac{0}{2}$ $a_1 = 0$

$\frac{10}{2} = 5 + \frac{0}{2}$ $a_3 = 0$

$\frac{5}{2} = 2 + \frac{1}{2}$ $a_4 = 1$

$\frac{2}{2} = 1 + \frac{0}{2}$ $a_5 = 0$

$\frac{1}{2} = 0 + \frac{1}{2}$ $a_6 = 1$

$N_2 = 101001$

- pretvorba ulomčnega dela:

naj bo $N < 1$

$N \cdot q = a_{-1} + S_{-1}$

$S_{-1} \cdot q = a_{-2} + S_{-2}$

$a_{-m} \in \{0,1\}$

do $S_{-m} = 0$

Primer: $N_{10} = 0{,}25$ $q = 2$

$0{,}25 \cdot 2 = 0 + 0{,}5$ $a_{-1} = 0$

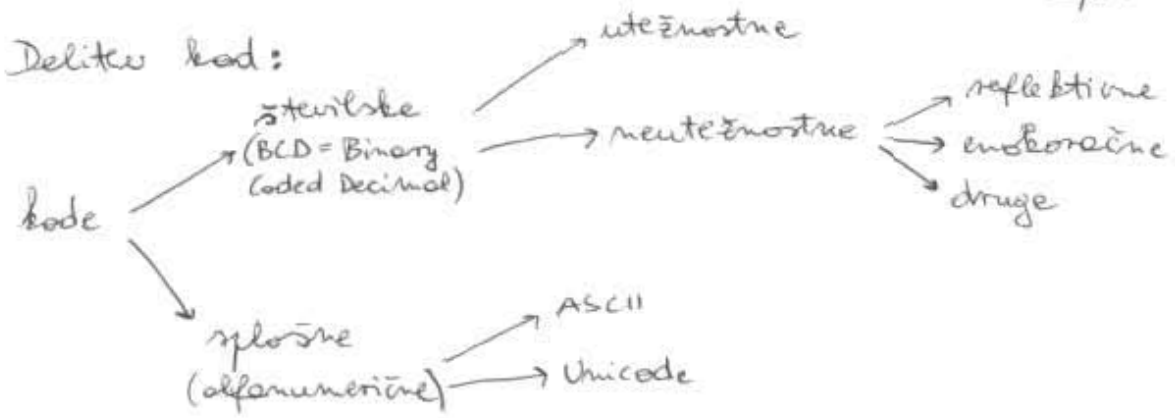
$0{,}5 \cdot 2 = 1 + 0$ $a_{-2} = 1$

$N_2 = 0{,}01$

Kode in kodiranje

Kodiranje = priredjanje števil, črk in drugih znakov v obliko, ki je primerna za zapis v kinomen zapisu

Delitev kod:



utežnostne:
- naravna BCD koda - preprosto kodiranje, primerna za prenos št. podatkov, ne omogoča odkrivanja napak
- liščinarna BCD koda - premik enic ⇒ ugodna za štetje
  - pomost enic omogoča preprosto odkrivanje napak

neutežnostne:
- reflektivna BCD koda - zrcalna (preprosto določanje komplementa)
- Grayeva endekoračna - spremeni se samo po 1 bit naenkrat (za kodiranje stiriških, osmiških, ... števil je tudi reflektivna)

2. BOOLEOVA ALGEBRA

Boolova spr. ima vrednost 0(false), 1(true)

Aksiomi in teoremi

- aksiomi (postulati) = pravila, ki se med seboj ne izključujejo in jih privzamemo brez preverjanja
- teoremi = izpeljani iz postulatov

▨ postulati (Huntingtonovi – eden od možnih sistemov, ki omogoča postavitev pravil za sistem operacij $\{+, \cdot, \bar{}\}$

P1: $x+0=x$ } nevtralnost
P1': $x \cdot 1 = x$ }

P2: $x+\bar{x}=1$ } komplementarnost
P2': $x \cdot \bar{x} = 0$ }

P3: $x+y=y+x$ } komutativnost
P3': $x \cdot y = y \cdot x$ }

P4: $(x+y)+z = x+(y+z)$ } asociativnost
P4': $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ }

P5: $(x+y) \cdot z = x \cdot z + y \cdot z$ } distributivnost
P5': $x + y \cdot z = (x+z) \cdot (y+z)$ }

- tri načini dokazovanja teoremov
  - iz aksiomov in že dokazanih teoremov
  - s pravilnostno tabelo
  - z Vennovimi diagrami

- teoremi:

T1: $x+1=1$
T2: $x+x=x$
T3: $x \cdot x = x$
T4: $x \cdot 0 = 0$
T5: $\bar{\bar{x}} = x$

T6: $x + x \cdot y = x$
T7: $x \cdot (x+y) = x$
T8: $(x+\bar{y}) \cdot y = x \cdot y$
T9: $x \cdot \bar{y} + y = x + y$
T10: $x + y + \bar{x} = 1$
T11: $\bar{x} \cdot \bar{y} \cdot x = 0$
T12: $\overline{x+y} = \bar{x} \cdot \bar{y}$ } de Morganova teorema
T13: $\overline{x \cdot y} = \bar{x} + \bar{y}$ }

▨ Dokaz de Morganovega teorema $\overline{x+y} = \bar{x} \cdot \bar{y}$

Najprej dokažemo, da velja:
$(x+y) \cdot \bar{x} \cdot \bar{y} = 0$
$(x+y) + \bar{x} \cdot \bar{y} = 1$

Dokaz 1. enačbe:

$\xrightarrow{PS}$ $(x+y)\cdot\bar{x}\cdot\bar{y} = \overbrace{x\cdot\bar{x}\cdot\bar{y}}^{0} + \overbrace{y\bar{x}\bar{y}}^{0} = 0$

Dokaz 2. en.:

$\xrightarrow{PS'}$ $(x+y)+\bar{x}\cdot\bar{y} = \overbrace{(x+y+\bar{x})}^{1}\cdot\overbrace{(x+y+\bar{y})}^{1} = 1$

Uvedemo novi sprem. $A = x+y$ in $B = \bar{x}\cdot\bar{y}$ ter velja:

$$A\cdot B = 0 \quad \text{in} \quad A+B = 1$$

Kadar za sprem. A in B veljata zgornji dve enačbi, pravimo, da sta si komplementarni. Za komplem. sprem. pa velja:

$$A = \bar{B} \quad \text{oz.} \quad \bar{A} = B$$

Sledi

$$\overline{x+y} = \bar{x}\cdot\bar{y}$$

3. PREKLOPNE FJE IN ELEMENTI

Mintern in maksterm

- mintern fje $f(x_1, x_2, \ldots x_n)$: konjunkcija vseh sprem. fje, v kateri vsaka sprem. nastopa enkrat, bodisi v osnovni (nenegirani) obliki, ali v neg. obliki

  minterni fje $f(x,y)$: $m_3 = xy$, $m_2 = x\bar{y}$, $m_1 = \bar{x}y$, $m_0 = \bar{x}\bar{y}$
- maksterm fje $f(x_1, \ldots x_n)$: disjunkcija vseh sprem. fje, v kateri vsaka sprem. nastopa enkrat

  maksterni fje $f(x,y)$: $M_3 = x+y$, $M_2 = x+\bar{y}$, $M_1 = \bar{x}+y$, $M_0 = \bar{x}+\bar{y}$

☐ Kaj je negacija maksterma?

Negacija maksterma je ustrezen mintern in obratno

$$\overline{M_i} = m_{2^n-1-i} \qquad \overline{m_i} = M_{2^n-1-i}$$

n = št. spremenljivk

Popolni normalni obliki zapisa preklopne fje

PKNO: produkt maksternov

$$f(x_1, x_2, \dots x_m) = \prod_{i=0}^{2^m-1} (\alpha_i + M_i) \quad \alpha_i \in \{0, 1\}$$

PDNO: vsota minternov

$$f(x_1, x_2, \dots x_m) = \sum_{i=0}^{2^m-1} \beta_i m_i \quad \beta \in \{0, 1\}$$

Izpelji obrazec za pretvorbo iz PKNO v PDNO

PKNO: $f = \prod_{i=0}^{2^m-1} (f_i + M_i)$

$$\overline{f} = \prod_{i=0}^{2^m-1} (\overline{f_i} + M_i) = (\overline{f_0} + M_0)(\overline{f_1} + M_1)(\overline{f_2} + M_2) \cdots$$

$$\overline{\overline{f}} = f = \overline{(\overline{f_0} + M_0)(\overline{f_1} + M_1)(\overline{f_2} + M_2) \cdots} = (\overline{\overline{f_0} + M_0}) + (\overline{\overline{f_1} + M_1}) + (\overline{\overline{f_2} + M_2}) + \dots$$

$$f = f_0 \overline{M_0} + f_1 \overline{M_1} + f_2 \overline{M_2} + \dots$$

$$f = \sum_{i=0}^{2^m-1} f_i \overline{M_i} = \sum_{i=0}^{2^m-1} f_i m_{2^m-1-i}$$

$\overline{M_i} = m_{2^m-1-i}$

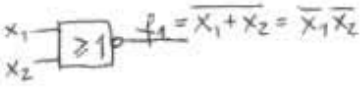
- PDNO in PKNO lahko zapišemo iz prav tabele, Veitchevega diag., Karnaughjevega diag.
  ↕
  Grayeva koda

Preklopne fje dveh spremenljivk (operatorji)

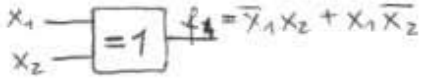
x_1	x_2	f_1 ↓	f_2 ⇐	f_3 ⇒	f_4 ⊕	f_5 ↑	f_6 ·	f_7 ≡	f_8 →	f_9 ←	f_{10} +
0	0	1	0	0	0	1	0	1	1	1	0
0	1	0	1	0	1	1	0	0	1	0	1
1	0	0	0	1	1	1	0	0	0	1	1
1	1	0	0	0	0	0	1	1	1	1	1

negirani implikaciji — implikaciji

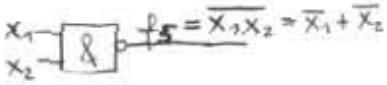
NOR(↓):



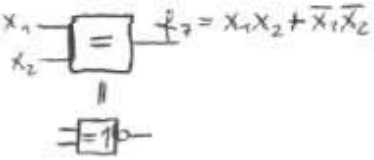
XOR(⊕) - vsota po modulu 2:



NAND(↑, |):



NXOR, XNOR, EQU(≡):



Funkcijsko polni sistemi

- sistem je funk. poln, če omogoča zapis poljubne preklopne fje
- funkcijsko polni sistemi:
  - $\{+, \cdot, ^-\}$ - elementarni FPS
  - $\{+, ^-\}$ $\overline{xy} = \overline{x} + \overline{y} \Rightarrow xy = \overline{\overline{x} + \overline{y}}$
  - $\{\cdot, ^-\}$ $\overline{x+y} = \overline{x}\,\overline{y} \Rightarrow x+y = \overline{\overline{x}\,\overline{y}}$

  } konjunkcijo nadomestimo s komb. disjunkcije in negacije, ter obratno
  - $\{\uparrow\}$ - Shefferjev FSP
  - $\{\downarrow\}$ - Pierceov FPS
  - $\{\equiv, +, 0\}, \{\oplus, \cdot, 1\}$

☑ Izpelji relacijo med Shefferjem in Pierceom + poenostavi, da dobiš fjo realizirano na enem oz. dveh nivojih.

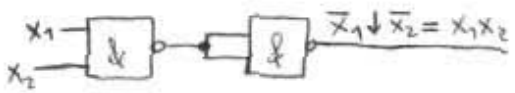
$x_1 \uparrow x_2 = \overline{x_1 x_2} = \overline{x}_1 + \overline{x}_2$

$\overline{x_1 \uparrow x_2} = \overline{\overline{x}_1 + \overline{x}_2} = \overline{x}_1 \downarrow \overline{x}_2$

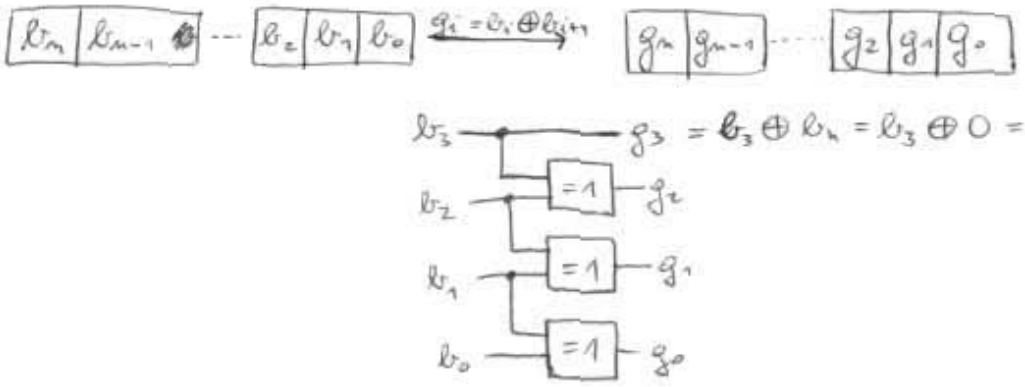
$x_1 \downarrow x_2 = \overline{x_1 + x_2} = \overline{x}_1 \overline{x}_2$

$\overline{x_1 \downarrow x_2} = \overline{\overline{x}_1 \overline{x}_2} = \overline{x}_1 \uparrow \overline{x}_2$

Negacija Sheff. operatorja je Pierceov operator negiranih spremenljivk in obratno.



Pretvorba linearne kode v Grayevo kodo



Razredi preklopnih fij – 5 zapstih razredov

- fje, ki ohranjajo ničlo (R_0): $f(0,0,\dots 0)=0$
- fje, — 1. — enico (R_1): $f(1,1,\dots 1)=1$
- sebi dualne fje (R_S): $f(\bar{x}_1,\bar{x}_2,\dots\bar{x}_n)=f(x_1,x_2,\dots x_n)$
- linearne fje (R_L):
  $f(x_1,x_2,\dots,x_n)=a_0\oplus a_1x_1\oplus a_2x_2\oplus\dots\oplus a_nx_n,\ a_i\in\{0,1\}$
- pozitivno monotone fje (R_M):
  $x_1\le y_1,\ x_2\le y_2,\dots,x_n\le y_n \Rightarrow f(x_1,x_2,\dots x_n)\le f(y_1,y_2,\dots y_n)$

- nabor fij $\{f_1,f_2,\dots,f_n\}$ je **FRS**, če nobeden od petih zapstih razredov ne vsebuje vseh fij nabora

Prevedba operatorjev iz $\{+,\cdot,^-\}$ v $\{\uparrow\}$

za prevedbo iz DNO uporabimo naslednji formuli (teorema):

$$x_1x_2\dots x_k + y_1y_2\dots y_m + z_1z_2\dots z_n =$$
$$= \overline{\bar{x}_1\bar{x}_2\dots\bar{x}_k + \bar{y}_1\bar{y}_2\dots\bar{y}_m + \bar{z}_1\bar{z}_2\dots\bar{z}_n} =$$
$$= \overline{\bar{x}_1\bar{x}_2\dots\bar{x}_k}\cdot\overline{\bar{y}_1\bar{y}_2\dots\bar{y}_m}\cdot\overline{\bar{z}_1\bar{z}_2\dots\bar{z}_n} =$$
$$= \uparrow(\overline{\bar{x}_1\bar{x}_2\dots\bar{x}_k},\ \overline{\bar{y}_1\bar{y}_2\dots\bar{y}_m},\ \overline{\bar{z}_1\bar{z}_2\dots\bar{z}_n}) =$$
$$= \uparrow\big(\uparrow(x_1,x_2,\dots x_k),\uparrow(y_1,y_2,\dots y_m),\uparrow(z_1,z_2,\dots z_n)\big)$$

$$\bar{x} = x\uparrow x$$

Simetrične fje

Simetrične fje.

Lokalno sim. fje glede na x_i in x_j:

$f(x_1,\dots,x_i,\dots,x_j,\dots,x_n) = f(x_1,\dots,x_j,\dots,x_i,\dots x_n) \Rightarrow x_i \sim x_j$

ali $f(x_1,\dots,x_i,\dots,x_j,\dots x_n) = f(x_1,\dots,\overline{x}_j,\dots,\overline{x}_i,\dots x_n) \Rightarrow x_i \sim \overline{x}_j$

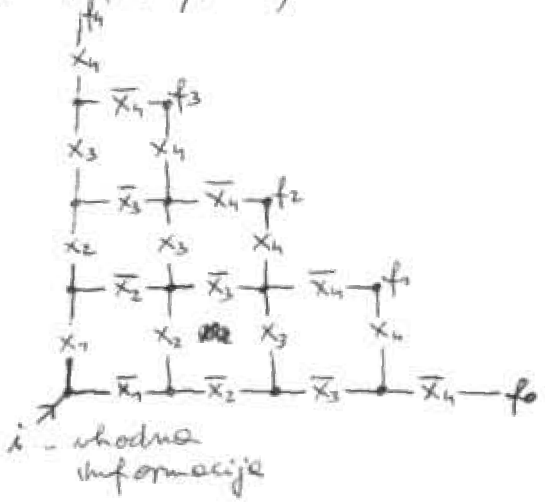
Globalno sim fje: simetrične glede na vsak par $\{x_i,x_j\}$,
t.j. za vsak par $\{x_i,x_j\}$ velja $x_i \sim x_j$ ali pa $x_i \sim \overline{x}_j$

zapis: $S_{a,b,\dots}(x_1,x_2,\dots x_n)$ – fja, ki v PDNO vsebuje vse mintherme z a negiranimi spremenljivkami, z b negiranimi spremenljivkami, ...

Realizacija simetričnih fij s Caldwellovim vezjem.

Značilnost simetričnih fij je v tem, da imajo zelo obsežno MDNO, zato je realizacija obsežna. Med najbolj uspešne rešitve spada realizacija v relejski tehniki s Caldwellovim vezjem.

$n = 4$ (4 spremenljivke)

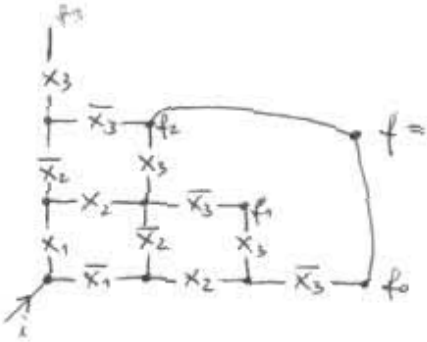


n-1)

8

9

Primer: Realizacija $f_{0,2}(x_1, \overline{x}_2, x_3)$



Pragovne fje in elementi

Pragovne fje.

Preklopna fja $f(x_1, x_2, \ldots x_n)$ je ~~prag~~ pragovna, če obstaja množica celih števil $\{w_1, w_2, w_3, \ldots w_n\}$ in takšno celo št. P, da velja: (uteži)

$$f(x_1, \ldots x_n) = \begin{cases} 0, \text{ če } \sum_{i=1}^{n} w_i x_i < P \\ 1, \text{ če } \sum_{i=1}^{n} w_i x_i \geq P \end{cases}$$

P - prag (hiperravnina, ki loči množico točk z vredn. 0

Pragovni element je skupina logičnih vrat, katere izhod f je pragovna fja vhodov $\{x_1, x_2, \ldots x_n\}$

Potreben pogoj za pragovnost: fja mora biti v MDNO enotijna (ne vsebuje hkrati negirane in nenegirane sprem.)

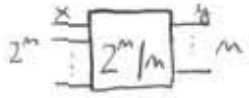
Ortogonalna fja - definicija.

Funkcija $f(x_1, x_2, \dots x_m) = k_1 + k_2 + \dots + k_i + \dots + k_j + \dots k_q$ je ortogonalna, če za vsak par (ki, kj) velja $k_i k_j = 0$ pri $i \neq j$. (k_i je minterm)
Vsaka PDNO fja je ortogonalna. Minimizirana disj. fja pa je ortogonalna samo, če je bila vsaka enica v Veitchevem diag. uporabljena samo 1x.

4. KOMBINACIJSKA VEZJA

Kodirniki

Kodirnik



izhod kodirnika je koda (šestnajstiška, BCD, ...) naslova vhoda, na katerem je nenegiran bit (enica)

Vhodom ponavadi določimo prioriteto (v primeru, da se na dveh vhodih naenkrat pojavi nenegiran bit upoštevamo tistega z višjo prioriteto

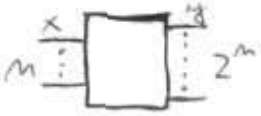
4/2, 8/3, 10/4 - BCD kodirnik (desetiški digit pretvori v šestnajstni zapis)

$y_0 =$ } iz tabele
$y_1 =$ }

10/4 → 0-9

Dekodirniki

Dekodirnik



izhod dekodirnika ima neneg. bit (1) na mestu, katerega naslov ustreza vrednosti kode na vhodu

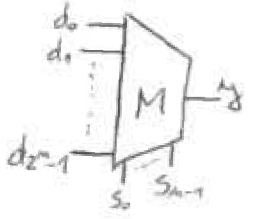
Prav. tabela je popolna (nima nedoločenih bitov), zato tokrat ni potrebno postavljanje prioritete

2/4 3/8 ~~10/4~~ 4/10 - iz BCD kode v des. digit

$y_0 =$
$y_1 =$
$y_2 =$
$y_3 =$
} tabela

Multipleksorji

MUX ima 2^n podatkovnih (d) vhodov in n izbirnih (s) adresnih vhodov ter en izhod. Na izhodu je vrednost tistega podatkovnega vhoda, katerega naslov je napisan na izbirnih vhodih.



$$y = \bar{s}_0\bar{s}_1 \dots \bar{s}_{n-1} d_0 + \bar{s}_0 \bar{s}_1 \dots \bar{s}_{n-2} s_{n-1} d_1 + \dots + s_0 \dots s_{n-1} d_{2^n-1} =$$

$$= m_0 d_0 + m_1 d_1 + \dots + m_{2^n-1} d_{2^n-1}$$

$$y = \underline{m} \Sigma\Pi \underline{d}$$

$\underline{m}$ ... vektor mintermov s vhodov

$\Sigma\Pi$ ... vsota produktov (skalarni produkt)

$\underline{d}$ ... vektor d vhodov

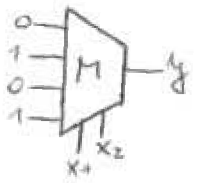
Trivialna realizacija preklopne fje z MUX-om.

$y = f(x_1, x_2, \dots x_n)$

Uporabimo MUX $2^n/n, 2^n-1$

Na adresne (izbirne) vhode pripeljemo spremenljivke $x_1, \dots x_n$, na podatkovne pa konstanti 0 in 1 v/n sicer jih prepišemo iz izhodnega stolpca pravilnostne tabele.

x_1	x_2	y
0	0	0
0	1	1
1	0	0
1	1	1



- Netrivialna realizacija:

Uporabimo MUX z manjšim št. n vhodov kot je spren.; na izbirne vhode pripeljemo nekaj spren., na podatkovne pa preostale spren., njihove fje in konstante

Realizacija skalarne fje z multipleksorji z enim n vhodom.

$y = f(x_1, x_2, \dots x_n)$

Fjo realiziramo kaskadno v več nivojih: Eno od spem. priključimo na izbirni vhod na podatkovne pa konst., katero od ostalih spem. in fjo ostalih spem. To fjo ostalih spem. dobimo kot izhod nekega drugega MUX. Zadevo ponavljamo, dokler nimamo na vhodih MUX-ov le konst. in spem.

- Vektorski MUX

k - 2^n vhodnih skalarnih MUX-ov z skupnimi s vhodi

$y_0 = m_0 d_{0,0} + m_1 d_{0,1} + \dots + m_{2^n-1} d_{0,2^n-1}$

$y_1 = m_0 d_{1,0} + m_1 d_{1,1} + \dots + m_{2^n-1} d_{1,2^n-1}$

$\vdots$

$y_k = m_0 d_{k,0} + m_1 d_{k,1} + \dots + m_{2^n-1} d_{k,2^n-1}$

$\underline{y} = \underline{m} \Sigma\!\!\!| \underline{D}$ → matrika podatkovnih vhodov

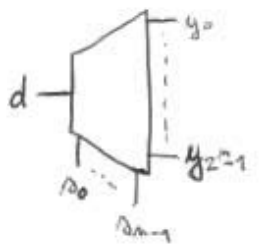
$\underline{y}$ ↓ vektor izhodov; $\underline{m}$ ↓ vektor mintermov s vhodov

Demultipleksorji

Demultipleksor

DEMUX ima en podatkovni (d) vhod, n izbirnih oz. adresnih (s) vhodov ter 2^n izhodov.

Vrednost podatkovnega vhoda se prenese na tisti izhod, katerega naslov je zapisan na izbirnih vhodih.



$y_0 = \bar{s}_0 \bar{s}_1 \dots \bar{s}_{n-1} d = m_0 d$

$y_1 = \bar{s}_0 \bar{s}_1 \dots \bar{s}_{n-2} s_{n-1} d = m_1 d$

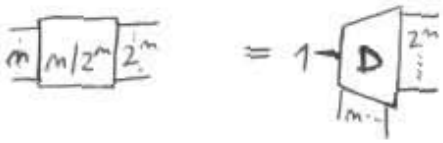
$\vdots$

$y_2 = s_0 s_1 \dots s_{n-1} d = m_{2^n-1} d$

$\underline{y} = \underline{m}^T \Sigma\!\!\!| d$ (podat. vhod; $\underline{m}^T$ transponirani vektor mintermov izbirnih vhodov; $\underline{y}$ ↓ vektor izhodov)

Povezava med demultipleksorjem in dekodirnikom

Če 2^m-izhodnemu DEMUX-u na podatkovni vhod vežemo enico, dobimo dekodirnik $m/2^m$



- več enakih DEMUX-ov s skupnimi izbirnimi vhodi lahko obravnavamo kot vektorski DEMUX

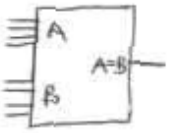
$k \times 2^m$ izhodni vektorski DEMUX

$$\underline{Y} = \underline{m}^T \Sigma k \, \underline{d}$$

matrika izhodov

Primerjalniki (komparatorji)

- n-bitni primerjalnik enakosti ima dva n-bitna (vektorska) vhoda in en izhod, ki pove ali sta števili na obeh vhodih enaki (preverja istoležne bite)

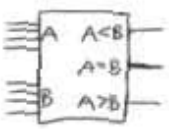


$$(A=B) = \overline{a_0 \oplus b_0} \cdot \overline{a_1 \oplus b_1} \cdot \overline{a_2 \oplus b_2} \cdot \overline{a_3 \oplus b_3}$$

NXOR 4

Primer: 4 bitni primerjalnik enakosti

- n-bitni primerjalnik velikosti ima 2 n-bitna vhoda in tri izhode, ki odražajo razmerje velikosti števil na vhodih



$$(A<B) = \overline{a_3} b_3 + \overline{(a_3 \oplus b_3)} \, \overline{a_2} b_2 + \overline{(a_3 \oplus b_3)} \, \overline{(a_2 \oplus b_2)} \, \overline{a_1} b_1 + \overline{(a_3 \oplus b_3)} \, \overline{(a_2 \oplus b_2)} \, \overline{(a_1 \oplus b_1)} \, \overline{a_0} b_0$$

$$(A=B) = \ldots$$

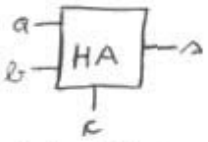
$$(A>B) = \overline{(A<B)} \, \overline{(A=B)}$$

Začne primerjati najpomembnejša bita - v primeru, da sta enaka nadaljuje pri naslednjih dveh, itd.

Primer: 4 bitni primerjalnik velikosti

Seštevalniki

Polovični seštevalnik, realizacija z NAND

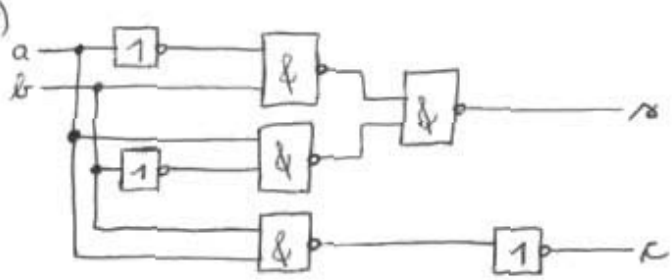


$c = ab$
$s = a \oplus b$

c ... carry bit (prenos) → na višji bit
s ... sum (vsota po modulu 2)

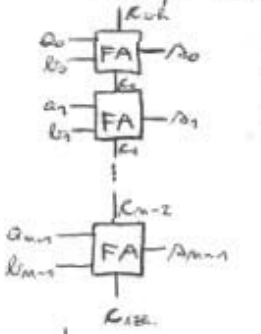
polovični 1-bitni seštevalnik (half adder)

$s = a \oplus b = \bar{a}b + a\bar{b} = \overline{\overline{\bar{a}b} + \overline{a\bar{b}}} = \overline{\bar{a}b} \cdot \overline{a\bar{b}}$



Realizacija polnega seštevalnika z NAND

Polni seštevalnik (full adder) upošteva še morebitni prenos z nižjega bita (c_{i-1})
S kaskadno vezavo FA dobimo n-bitni serijski (zaporedni) seštevalnik (ripple adder)



$c_i = a_i b_i + (a_i \oplus b_i) c_{i-1}$
$s_i = a_i \oplus b_i \oplus c_{i-1}$

$c_i = a_i b_i + \bar{a}_i b_i c_{i-1} + a_i \bar{b}_i c_{i-1} = a_i b_i + (a_i \oplus b_i) c_{i-1}$

s_i:

a_i \ b_i	0	1
0	0	1
1	1	0

c_{i-1}

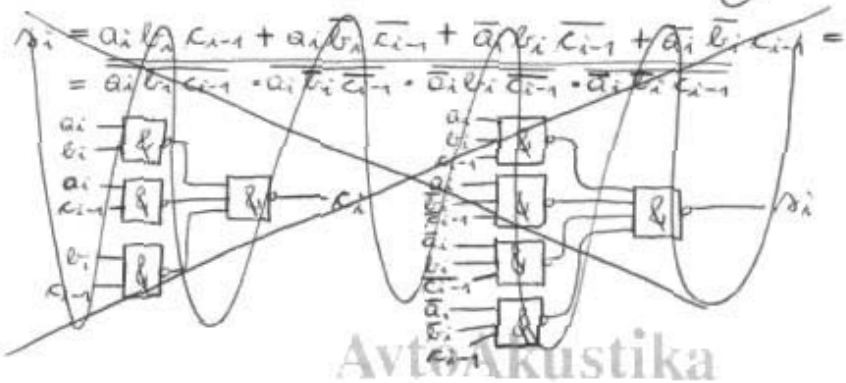
c_i:

a_i \ b_i	0	1
0		1
1	1	1

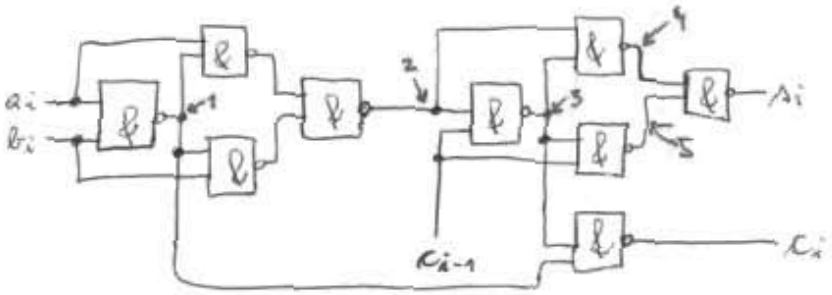
c_{i-1}

↳ n-bitni serijski seštevalnik

$s_i = a_i b_i c_{i-1} + a_i \bar{b}_i \bar{c}_{i-1} + \bar{a}_i b_i \bar{c}_{i-1} + \bar{a}_i \bar{b}_i c_{i-1} =$
$= \overline{a_i b_i c_{i-1}} \cdot \overline{a_i \bar{b}_i \bar{c}_{i-1}} \cdot \overline{\bar{a}_i b_i \bar{c}_{i-1}} \cdot \overline{\bar{a}_i \bar{b}_i c_{i-1}}$



Realizacija z NAND:



1: $\overline{a_i b_i}$

2: $a_i \overline{b_i} + \overline{a_i} b_i = a_i \oplus b_i$

3: $\overline{(a_i \oplus b_i) \cdot c_{i-1}}$

$c_i = \overline{\overline{a_i b_i} \cdot \overline{(a_i \oplus b_i) \cdot c_{i-1}}} = a_i b_i + (a_i \oplus b_i) c_{i-1}$

4: $\overline{(a_i \oplus b_i) \cdot \overline{(a_i \oplus b_i) \cdot c_{i-1}}}$

5: $\overline{\overline{(a_i \oplus b_i) \cdot c_{i-1}} \cdot c_{i-1}}$

$s_i = 4 \uparrow 5 = (a_i \oplus b_i) \cdot \overline{(a_i \oplus b_i) \cdot c_{i-1}} + c_{i-1} \cdot \overline{(a_i \oplus b_i) \cdot c_{i-1}} =$

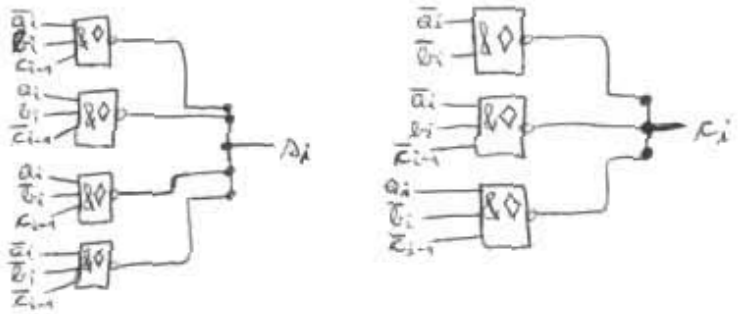
$= (a_i \oplus b_i) \cdot \left(\overline{a_i \oplus b_i} + \overline{c_{i-1}}\right) + c_{i-1} \cdot \left(\overline{a_i \oplus b_i} + \overline{c_{i-1}}\right) =$

$= (a_i \oplus b_i) \cdot \overline{c_{i-1}} + c_{i-1} \cdot \left(\overline{a_i \oplus b_i}\right) = a_i \oplus b_i \oplus c_{i-1}$

Realizacija popolnega seštevalnika z NAND-WIRED-AND – prednosti in slabosti. (= AND-OR-INVERT)

$s_i = a_i \oplus b_i \oplus c_{i-1} = a_i \overline{b_i}\, \overline{c_{i-1}} + \overline{a_i}\, \overline{b_i}\, c_{i-1} + \overline{a_i}\, b_i\, \overline{c_{i-1}} + a_i b_i c_{i-1}$

$c_i = a_i b_i + (a_i \oplus b_i) c_{i-1} = a_i b_i + a_i \overline{b_i} c_{i-1} + \overline{a_i} b_i c_{i-1}$



Posamezne sprem. iz MDNO negiramo in peljemo na NAND-WIRED
Ker imamo pri omenjenem vezju izhode povezane, pridelimo na času, ki ga signal porabi za potovanje skozi vrata (manjši Φ skupni delay vezja).

- paralelni (vzporedni) seštevalnik (carry look-ahead adder) vse prenose izračuna samo iz vhodnih podatkov: c_{vh}, $G_i = a_i b_i$, $P_i = a_i \oplus b_i$

Zakasnitev je precej manjša kot pri serijskem sešteva, kjer vrednosti $s_0, s_1, \dots s_{n-1}$ na izhode prihajajo ena za drugo, pri paralelnem pa naenkrat.

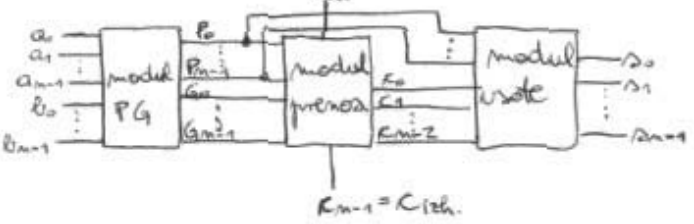
Serijski sešt. → delay je linearna fja števila bitov - $2n$
Paralelni sešt. → delay je neodvisen od št. bitov

Izpelji enačbe za prenose pri paralelnem seštevalniku s PG modulom.

$$c_0 = G_0 + c_{vh} P_0$$
$$c_1 = G_1 + c_0 P_1 = G_1 + G_0 P_1 + c_{vh} P_0 P_1$$
$$c_2 = G_2 + c_1 P_2 = G_2 + G_1 P_2 + G_0 P_1 P_2 + c_{vh} P_0 P_1 P_2$$
$$\vdots$$
$$\boxed{c_k = G_k + \sum_{j=0}^{k-1} \left(G_j \prod_{i=j+1}^{k} P_i \right) + c_{vh} \prod_{i=0}^{k} P_i}$$

$$s_0 = P_0 \oplus c_{vh}$$
$$s_1 = P_1 \oplus c_0$$
$$s_2 = P_2 \oplus c_1$$
$$\vdots$$
$$s_k = P_k \oplus c_{k-1}$$

- paralelni seštevalnik tako zgradimo iz treh delov:
  - modul PG – določi vse P_i in G_i
  - modul prenosov – določi vse c_i
  - modul vsote – določi vse s_i



} zgradba n-bitnega paralelnega seštevalnika

Zakasnilni čas je za n-bitni seštevalnik 8 enot (3 na PG + + 2 na prenos. + 3 na vsoti) skupno število elementov NAND pa je $\frac{n(n+23)}{2}$.

- za odštevanje uporabimo binarni zapis števil s predznakom (1 - negativno, 0 - pozit.), število $-a$ pa je dvojiški komplement števila a, ki ga dobimo tako, da v binarnem zapisu števila a negiramo vsak bit posebej (mislimo z zanemarjanjem enicami in dvatno) in rezultatu prištejemo 1

- $b-a$ je b plus dvoj. kompl. št. a, morebitni prenos z višjega bita zanemarimo; če dejanska vrednost $(b-a)$ ne preseže območje n-bitnega zapisa, je rezultat pravilen

Primer: $(7-3)$ v 4-bitnem zapisu

$7_{10} = 0111$
$3 = 0011$
$-3 = 1100$
$+ \quad 1$
1101

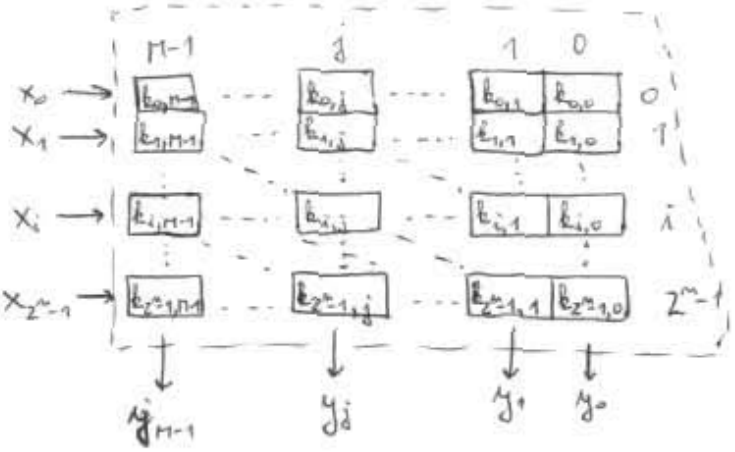
$(7-3) =$
0111
$+ 1101$
0100

5. PROGRAMIRLJIVE ODLOČITVENE STRUKTURE

Programirljivost

- povezave v notranjosti vezja določa uporabnik:
  - programirljivi bralni pomnilnik PROM (enkrat električno zapisljiv; povezave v vezju so varovalke, ki jih z visoko nap. prežgemo in izločimo ustrezne minterme)
  - izbrisljivi (erasable) PROM = EPROM (povezave so MOS tranzistorji, visoka nap. → neprevodna plast, izbris z ultravijolično svetlobo)
  - električno izbrisljivi PROM = EEPROM (brisanje z nasprotno polariteto od tiste pri pisanju, po bytih
  - EEPROM s hitrim izbrisom (FLASH) (pisanje po 512 byte-ov naenkrat → hitrejše
  - PLA in PAL

Pomnilna mreža



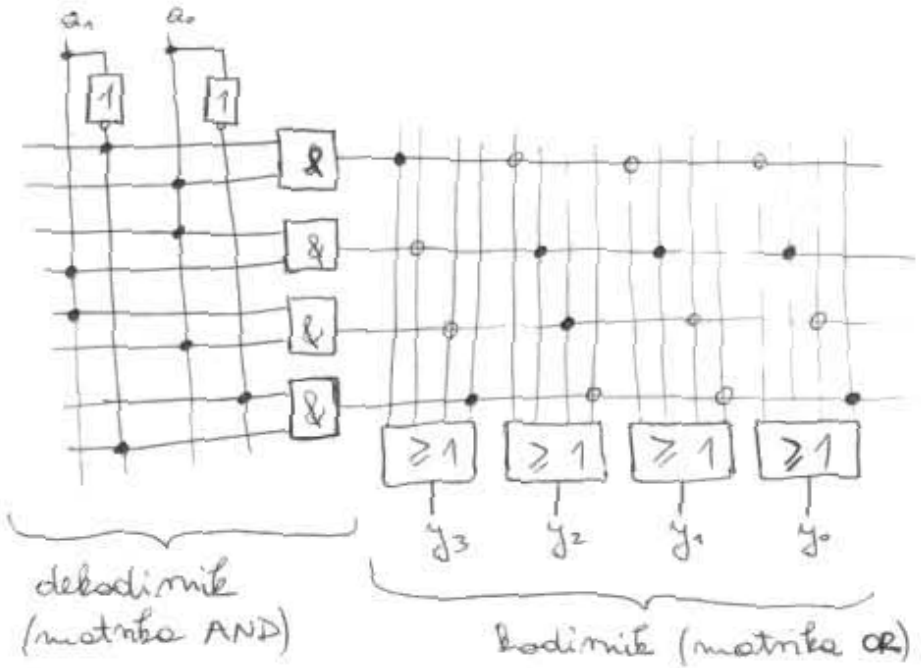
- struktura, ki v notranjosti hrani urejeno 2^n terico M bitnih besed, na vhod pripeljemo 2^n bitni podatek, na izhodu dobimo M bitni podatek

- vhodno izhodne povezave mreže podaja izraz:

$$y_j = \sum_{i=0}^{N-1} x_i k_{ij} \quad \text{oz. matrično} \quad \underline{y} = \underline{x} \Sigma \& \underline{K}$$

- le eden od vhodov x_i ima lahko vrednost 1
- dobimo splošni, programirljivi kodirnik, kjer izhodni vektor ni eno od standardnih kod za število i, temveč poljubno zaporedje M bitov, ki jih vprogramiramo v i-to vrstico mreže

PROM



$y_3(a_1,a_0) = \overline{a}_1 a_0 + a_1 \overline{a}_0$
$y_2 =$ ⋮
$y_1 =$ ⋮
$y_0 =$ ⋮

- povezave v matriki AND so stalne (niso programirljive)
- povezave v matriki OR so program.
- območnost dekodirnika eksponentno narašča s številom vhodnih spremenljivk. ⇒ PROM-i so veliki, dragi, počasni

Programirljiva logična mreža - PLA

- programirljiv je tudi dekodirni del pomnilnika, t.j. matrika AND → bolj fleksibilno vezje

Programirljivo logično polje - PAL

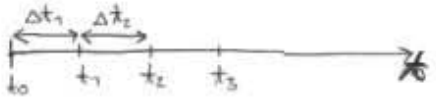
- programirljiva le matrika AND, matrika OR je nespremenljiva → deluje hitreje kot PLA, a je manj fleksibilno

6. ČASOVNO ODVISNE PREKLOPNE FUNKCIJE IN POMNENJE V PREKLOPNIH VEZJIH

6.1 Neodvisna časovna preklopna spremenljivka

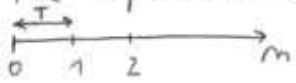
6.1.1 Vpeljava časa v preklopne spremenljivke

Ekvidistantna delitev časa:



$\Delta t_1 = \Delta t_2 = \ldots = \Delta t = T$

- čas razsekamo na enake intervale – rečemo, da se spremenljivke dogajajo le ob 0, 1, 2, ... n, (n+1)



$t = n\Delta t = nT$ nam bo ponenil sedanji čas

6.1.2 Zapis neodvisne časovne spremenljivke

a) s funkcijsko odvisnostjo

$x(t) = x(n\Delta t) = x(nT) \equiv x(n) \equiv x[n]$

b) z operatorjem

$$\Delta^n x = \begin{cases} x\,; & \text{če je } t \in (n\Delta t) \\ 0\,; & \text{če je } t \notin (n\Delta t) \end{cases}$$

$\Delta^n x$ - neodvisna časovna spremenljivka

Časovni operator dodamo k funkcijsko polnemu sistemu (FPSPF). Izhodiščne opazovanje je sedanji čas:

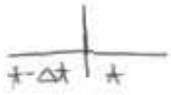
$\Delta^0 x = x$ - izhodiščno opazovanje

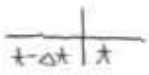
$\Delta^n(\Delta^m x) = \Delta^{n+m} x$ - pomik v desno po časovni osi (prih. čas)

$\Delta^n(x_1 x_2) = \Delta^n x_1 \Delta^n x_2$ - razširjenost nastopa spremenljivk

$\Delta^{\bar{n}} x = \bar{\Delta}^n x = \Delta^n \bar{x}$ - komplement čas. spremenljivke

6.1.3 Prehod med statičnimi signali in fronte





Def. fje fronte (prednje in zadnje) neodvisne časovne spremenljivke:

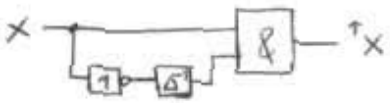
$$\uparrow x = (\Delta^{-1}\bar{x})\,x \qquad x^{\downarrow} = (\Delta^{-1}x)\,\bar{x}$$

Njuna medsebojna povezava:

$$\uparrow x = (\bar{x})^{\downarrow} \qquad x^{\downarrow} = \uparrow(\bar{x})$$

Prožilno vezje na pozitivno fronto. Kako bi realiziral vezje za generiranje impulza (Diracovega) in zakaj to kdaj potrebuješ.

$$\uparrow x = (\Delta^{-1}\bar{x}) \cdot x$$



↳ zakasnitev (delay) za eno enoto

Fje fronte rabimo za CLOCK (za vsa sinhrona vezja).

6.2 Časovna preklopna fja

6.2.1 Odvodi časovnih operacij

$$\uparrow(x_1x_2) = \uparrow x_1x_2 + x_1\uparrow x_2 + \uparrow x_1\uparrow x_2$$

$$\uparrow(x_1+x_2) = \uparrow x_1\bar{x}_2 + \bar{x}_1\uparrow x_2 + \uparrow x_1\uparrow x_2$$

$$(x_1x_2)^{\downarrow} = x_1^{\downarrow}x_2 + x_1x_2^{\downarrow} + x_1^{\downarrow}x_2^{\downarrow}$$

$$(x_1+x_2)^{\downarrow} = x_1^{\downarrow}\bar{x}_2 + \bar{x}_1x_2^{\downarrow} + x_1^{\downarrow}x_2^{\downarrow}$$

$$\uparrow(\bar{x}) = x^{\downarrow};\quad (\bar{x})^{\downarrow} = \uparrow x$$

odvode si lahko ogledamo na konkretnem primeru in zapišemo pravila

6.2.2 Def. časovne preklopne fje

Definicijo časovnega operatorja prenesemo na splošno preklopno fjo:

$$f(\Delta^n x_1, \Delta^n x_2, \dots \Delta^n x_k) = \Delta^n f(x_1, x_2, \dots x_k)$$

6.2.3. Podobnost časovno odvisnih in neodvisnih preklopnih fij

$$f(x_1, x_2, \dots x_k) \Rightarrow f(x_1, x_2, \dots x_k, t)$$

$$f(x_1, x_2, \dots, x_k, t) = \sum_{i=0}^{k} \Delta^i f_i(x_1, x_2, \dots, x_k)$$

Δ_k^i podoben m_i

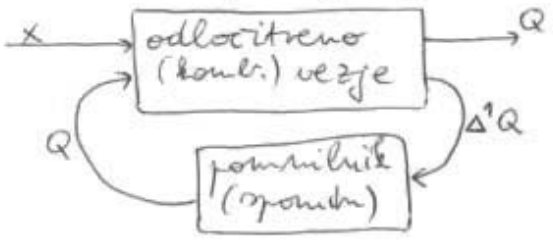
$$\sum_i \Delta^i = 1 \qquad \sum_i m_i = 1$$

$$\Delta^i \Delta^j = 0 \qquad m_i m_j = 0;\ i \neq j$$

6.3. Pomnjenje in spominski elementi

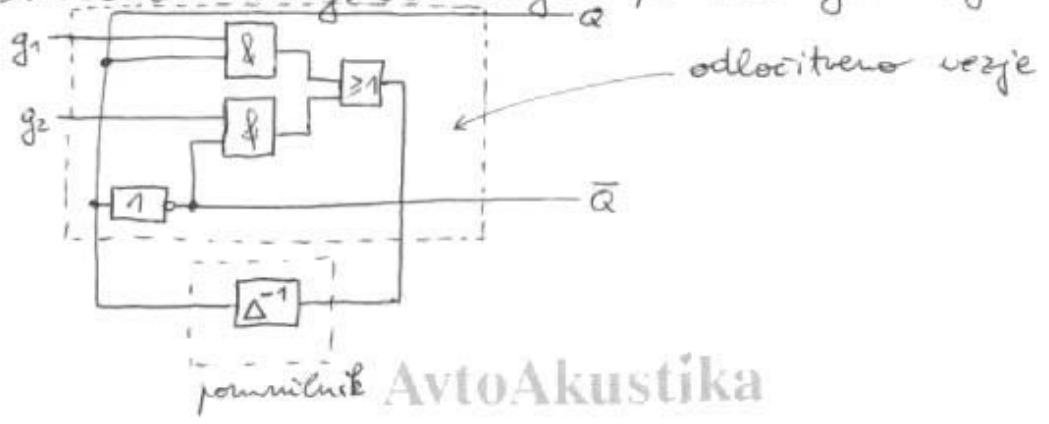
6.3.1. Osnovna spominska celica

Princip pomnjenja v prekl. vezjih:



$Q = Q(nT) = Q(n)$
$\Delta^1 Q = Q(nT+T) = Q(n+1)$

Shematični diag. osnovnega spominskega vezja:



6.3.2 Karakteristična enačba in kmiljenje pomnilniških celic

VII Osnovna pomnilniška enačba

Splošni vhodni fji sta:

$g_1 = g_1(x_1, x_2, \dots x_2) = g_1(\vec{x})$; $\vec{x} = (x_1, x_2, \dots, x_2)$

$g_2 = g_2($ — " — $) = g_2(\vec{x})$; — " —

g_1 pove kaj bo na izhodu, če je $Q=1$

g_2 pove — " — $Q=0$

$\boxed{\Delta^1 Q = g_1 Q + g_2 \bar{Q}}$ osnovna enačba pomnjenja - karakteristična fja

$Q = \Delta^{-1}(g_1 Q + g_2 \bar{Q})$ kar. fja vezja osnovne pomnilniške celice

Kmiljenje osnovne pomnilniške celice izvedemo na sledeče načine:

- D - zakasnitev (delay)
- T - prožnje (triggering)
- R - pogojno brisanje (reset)
- S - pogojno postavljanje (set)
- K - brezpogojno brisanje celice
- J - — " — postavljanje - " -

6.3.3 Vhodne fje pomnilniških celic

g_1, g_2 - zunanji fji - sta neodvisni

R, S - vhodni fji RS pomnilne celice; to je njeno kmiljenje

Napišemo tabelo

g_1	g_2	Q	$\Delta^1 Q$	R	S

in [redacted] za naše vezje izračunamo ($\Delta^1 Q = g_1 Q + g_2 \bar{Q}$).

Dobimo sledeči fji:

$R(g_1, g_2, Q) = \bar{g}_1 Q$

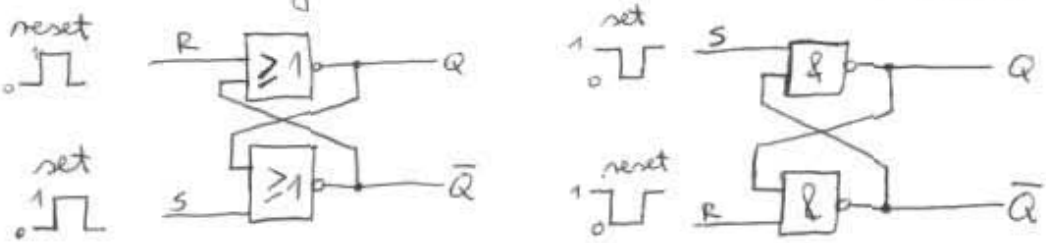
$S(g_1, g_2, Q) = g_2 \bar{Q}$

Podobno bi lahko realizacijo izvedli tudi z drugimi celicami

6.4 Izvedbe pomnilnih celic

6.4.1 Asinhronska pomnilna celica - RS zatič

Realizacija z dvemi NOR ali NAND vrati:



Preklopni oz. izjavnostni tabeli:

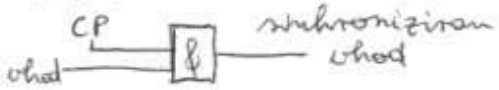
S	R	Q	$\bar{Q}$	opombe
0	0	0	1	po S=0; R=1
0	0	1	0	po S=1; R=0
0	1	0	1	
1	0	1	0	
1	1	0	0	prep. stanje

S	R	Q	$\bar{Q}$	opombe
1	1	0	1	po S=1; R=0
1	1	1	0	po S=0; R=1
0	1	1	0	
1	0	0	1	
0	0	1	1	prepov. stanje

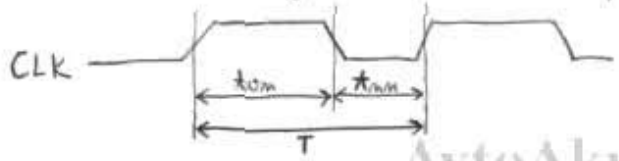
6.4.2 Sinhronizacija in sinhronizirana celica RS

▨ Ura - CLOCK (kje, v bakšnih vezjih)

Doseči želimo, da se stanje pomnilne celice spremeni le ob 0, 1, ..., n, (n+1) - v flip flop vpeljemo sinhronizacijski impulz (CP), ki določa te čase. Na ta način iz asinhronskih vezij preidemo na sinhronska.



a) urin signal aktiven pri visokem nivoju napetosti:



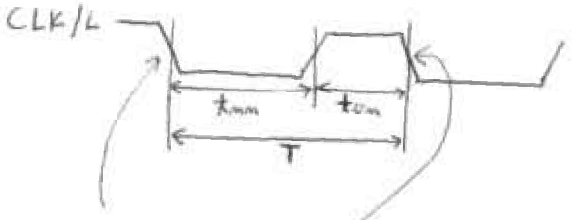
T ≡ čas periode

$\frac{1}{T}$ ≡ frekvenca

$\frac{t_{vm}}{T}$ ≡ faktor polnjenja

spremenile stanj nastopijo ob pozitivni fronti ↑

b) urin signal aktiven pri nizkem nivoju napetosti:



$\frac{t_{nn}}{T}$ = faktor polnjenja

spremenbe stanj nastopijo tu (↓)

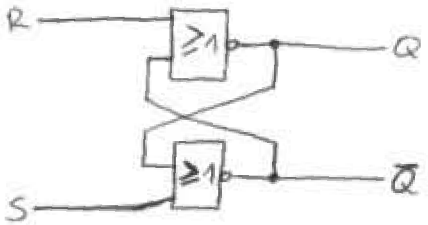
- Karakteristična enačba in karakteristična tabela:

$\Delta^1 Q = \bar{R}Q + S$; pogoj $RS = 0$

R	S	$\Delta^1 Q$
0	0	Q
0	1	1
1	0	0
1	1	* neddločeno

zaradi

RS celica - * realizacija in izpeljava karakteristične enačbe.



$\Delta^1 Q = \overline{R + \bar{Q}} = \overline{R + \overline{S+Q}} = \bar{R} \cdot \overline{(S+Q)} = \bar{R}S + \bar{R}Q$

ker mora biti izpolnjen pogoj $RS = 0$ lahko pišemo:

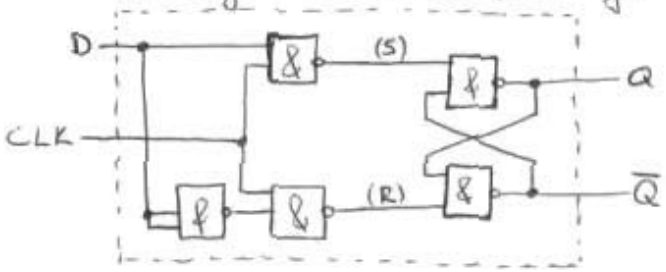
$\Delta^1 Q = \bar{R}S + \bar{R}Q + RS = S(\bar{R} + R) + \bar{R}Q = \bar{R}Q + S$

- vzbujalna (ekscitacijska) tabela za RS:

Q	$\Delta^1 Q$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

6.4.3 Sinhronizirana zatična pomnilniška celica – D zatič

Dobimo jo z modifikacijo RS celice:

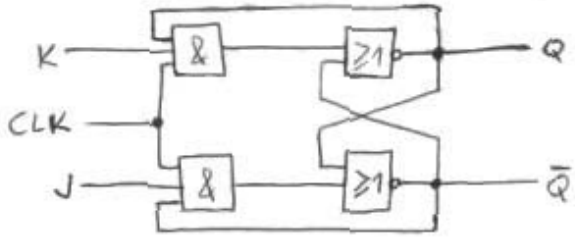


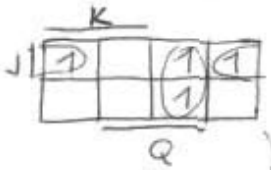
- karakteristična enačba in tabela:

$\Delta^1 Q = D$

Q	D	$\Delta^1 Q$
0	0	0
0	1	1
1	0	0
1	1	1

6.4.4 Sinhronizirana pomnilniška celica JK





Karakteristična enačba oz. izhodna fja:

$$\Delta^1 Q = \overline{KQ + \bar{Q}} = \overline{KQ + J\bar{Q} + Q} = \overline{KQ + J\bar{Q} \cdot \bar{Q}} = \overline{KQ + (\bar{J} + Q)\bar{Q}} =$$
$$= \overline{KQ + J\bar{Q}} = \overline{KQ} \bullet \overline{J\bar{Q}} = (\bar{K} + \bar{Q}) \cdot (J + Q) = \bar{K}(J + Q) + \bar{Q}(J + Q) =$$
$$= \bar{K}J + \bar{K}Q + J\bar{Q} = KQ + J\bar{Q}$$

- Karakteristična tabela:

J	K	$\Delta^1 Q$
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

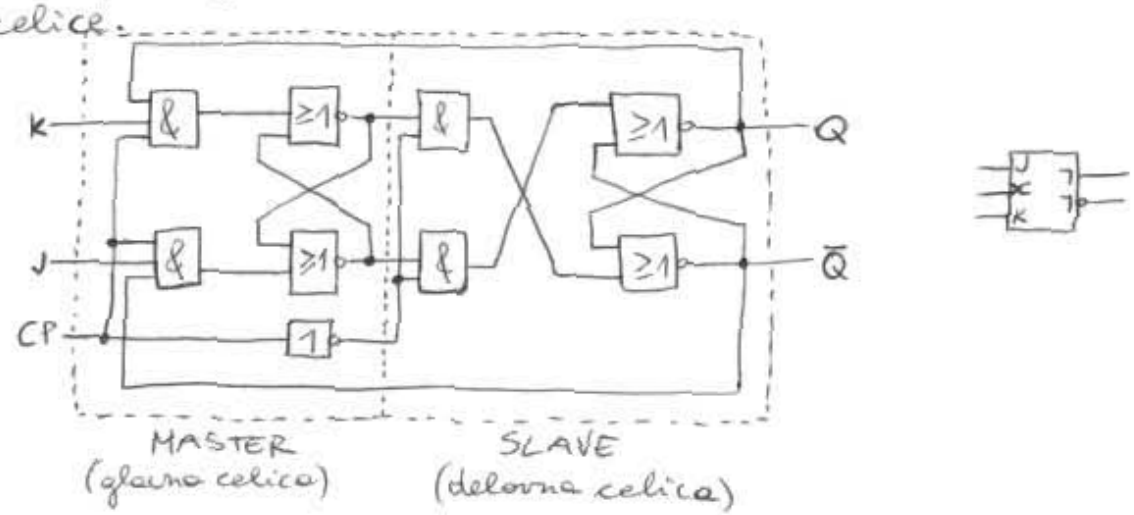
→ J=1, K=1 povzroči negacijo prejšnjega stanja

- Vzbujalna tabela

Q	$\Delta^1 Q$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Zakaj se uporablja Master-slave celica?

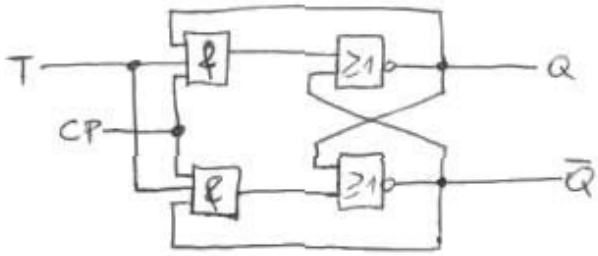
Master-slave celica ali celica s predpomnjenjem se uporablja kot sledilno vezje JK spominske celice:



Glavna celica sprejme informacije ob $\uparrow CP$, jih shrani, ter odda delovni celici ob prvi možni fronti $CP^{\downarrow}$. Z Master-slave se izognemo osciliranju med "časom polnjenja", do katerega pride, ko sta J=1 in K=1 pri navadni JK celici.

6.4.5 Spominska celica T

Dobimo jo iz JK, če vhoda zvežemo skupaj:



- Karakt. en. in tabela:

$$\Delta^1 Q = T\bar{Q} + \bar{T}Q$$

T	$\Delta^1 Q$
0	Q
1	$\bar{Q}$

- vzbujalna tabela:

Q	$\Delta^1 Q$	T
0	0	0
0	1	1
1	0	1
1	1	0

Izpelji enačbo za T celico z g parametri in jo realiziraj.

Splošna pomnilna enačba: $\Delta^1 Q = g_1 Q + g_2 \bar{Q}$

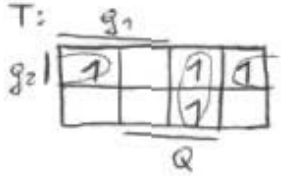
T - triggering - prožanje

Celica pri 0 ohranja, pri 1 po spremeni stanje.

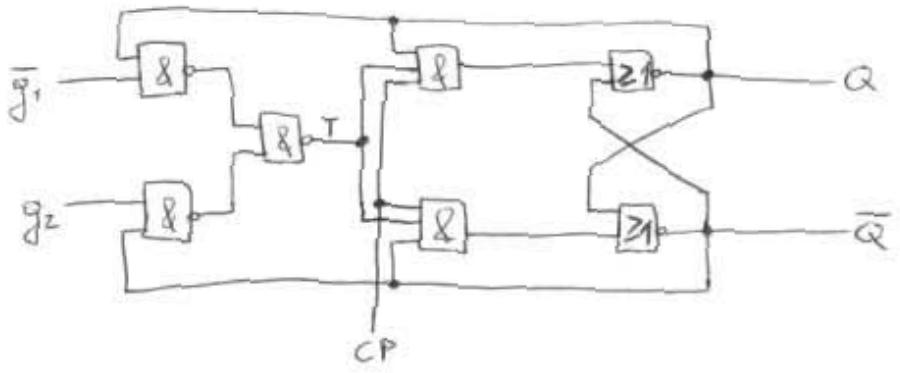
⇓ iz teh podatkov sest. vzbujalno tabelo

g_1	g_2	Q	$\Delta^1 Q$	T
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	1
1	0	0	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	1	0

T:



$T = \bar{g}_1 Q + g_2 \bar{Q}$



pri pogoju $g_1 g_2 = 0$ velja: $T = \bar{g}_1 = g_2$

Razlike med JK in RS (zakasnitve, nestabilna stanja).

JK in RS celici sta si precej podobni, obe se pri kombinacijah 00, 01, 10 enako odzivata. Razlika je pri kombinaciji 11, pri kateri ima RS celica prepovedano stanje, saj bi se v tem primeru na izhodu pojavilo $Q = \bar{Q}$, kar pa ne more biti res. Celica JK pa ima stanje 11 dovoljeno a se vseeno pojavi določen problem: oba vhoda sta 1, torej gre ob CP signal skozi tista AND vrata ki imajo tudi drug vhod 1 (Q ali $\bar{Q}$); vrednost te enice se prenese naprej na ustrezen NOR, izhod mora zaradi tega spremeniti stanje; če bi bil CP še bar 1 bi bila vsaj druga AND vrata aktivirana in f.f. bi spet spremenil stanje, itd. → tudi pri CLK: ne sme biti predolg (zadeva pri JK rešljiva s predpomnjenjem)

Prepovedana stanja v pomnilniških celicah

Prepovedano stanje nastopi pri nesinhroniziranih celicah, kjer bi bilo $Q = \bar{Q}$, kar pa ni mogoče, saj dobimo dve stimulaciji, ki zahtevata nasprotni akciji. Pri takšni situaciji bi se vezje znašlo v nestabilnem stanju, katero bi preslo v eno ali drugo stabilno stanje. Ker ne vemo katero stanje bi to bilo, te vhodne kombinacije ne dopuščamo. Pri sinhroniziranih celicah takemu stanju pravimo nedoločeno stanje (lahko je 0 ali 1)

RS ff realiziran z NOR : 1,1 } primera prepoved.
— " — NAND : 0,0 } stanj

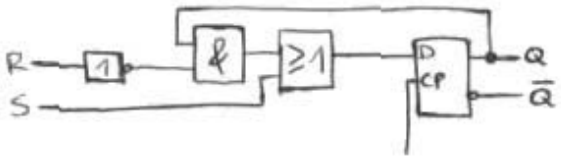
6.5 Spominske celice z dinamičnim prožnjem

- dinamično prožena spominska celica D
- — " — JK
- dinamično prožena zatična celica z dodatnim vhodom - E

6.6 Pretvorbe spominskih celic

Vsako izmed dosedanjih spominskih celic je z dodatnim kombiniranjem mogoče pretvoriti v neko drugo.

- $\Delta^1 Q = S + \bar{R}Q = D \qquad D = S + \bar{R}Q$



- $\Delta^1 Q = \bar{K}Q + J\bar{Q} = D$
- $\Delta^1 Q = S + T\bar{Q} + \bar{R}\bar{T}Q = D$

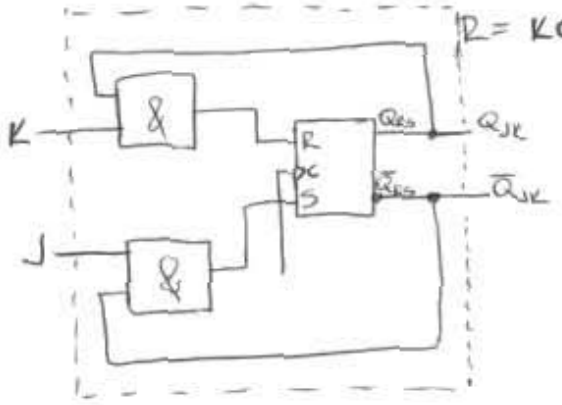
- pretvorba RS v JK

$\Delta^1 Q_{RS} = S + \bar{R} Q_{RS}$ $\quad RS = 0$ $\quad \Delta^1 Q_{JK} = \bar{K} Q_{JK} + J \bar{Q}_{JK}$

$Q = Q_{RS} = Q_{JK} \Rightarrow \quad S + \bar{R} Q = \bar{K} Q + J \bar{Q} = \Delta^1 Q$

⇓ tabela ⇓

$R = KQ; \quad S = J\bar{Q}$

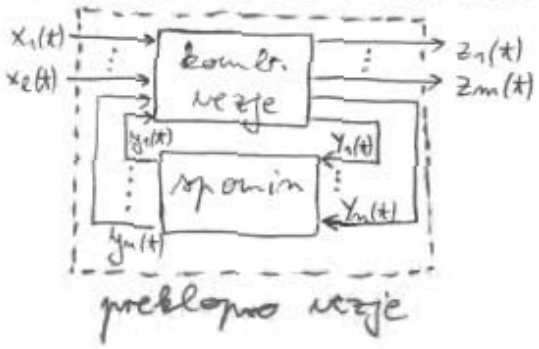


6.7. Komercialne izvedbe spominskih celic

- spominska celica s predpomnjenjem JK
- spominska celica D

7. KONČNI AVTOMATI

7.1 Osnovne karakteristike

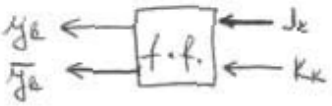


$x_1(t), x_2(t), \ldots x_k(t)$ – vhodne spren.
$y_1(t), y_2(t), \ldots y_n(t)$ – spremenljivke stanja (sekundarne vhodne spren.)
$Y_1(t), Y_2(t), \ldots Y_n(t)$ – spren. nasl. stanja (vzbujalne spren.)
$z_1(t), z_2(t), \ldots z_m(t)$ – izhodne spren.

7.2 Sinhronski sekvenčni avtomati

7.2.1 Modeli in vrste spomina

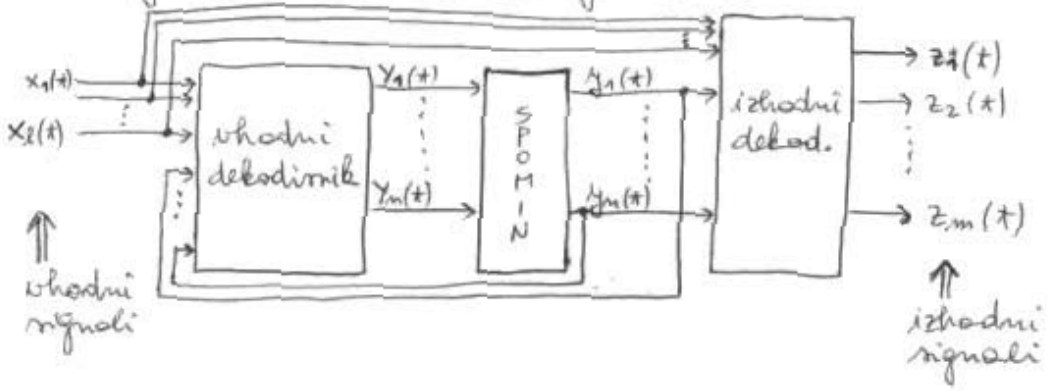
Vrste spomina.



$$\Delta^{\wedge} y_k = \bar{K}_k y_k + J_k \bar{y}_k$$

$$\Delta^{\wedge} y_k = Y_k$$

Serijski model sekvenčnega avtomata:



7.2.2. Definicija vhodov, stanj in izhodov

$\vec{x} = (x_1, x_2, \ldots, x_i, \ldots x_\ell)$; $x_i = \{0, 1\}$

$p = 2^\ell$ - število možnih vhodnih črk ali stanj

$\vec{X} = (\vec{x}_1, \vec{x}_2, \ldots, x_p)$ - vhodna abeceda avtomata

$\vec{z} = (z_1, z_2, \ldots, z_m)$; $q = 2^m$ - št. možnih izhodnih črk ali stanj

$\vec{Z} = (\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_q)$; - izhodna abeceda avtomata

$\vec{s} = (s_1, s_2, \ldots s_n)$; $r = 2^n$; - št. možnih notranjih črk ali stanj

$\vec{S} = (\vec{s}_1, \vec{s}_2, \ldots \vec{s}_r)$; - notranja abeceda avtomata

7.2.3. Def. determinističnega avtomata

$\Delta^1 \vec{s} = \delta[\vec{s}, \vec{x}]$; $\vec{z} = \lambda[\vec{s}, \vec{x}]$

$\delta : \vec{X} \times \vec{S} \to \vec{S}$

$\lambda : \vec{X} \times \vec{S} \to \vec{Z}$ pri Mealyevem

$\lambda : \vec{S} \to \vec{Z}$ pri Moore-ovem

$\vec{X} \times \vec{S}$ - kartezični produkt $(\vec{x}_i, \vec{s}_j)$

$\delta : (\vec{x}_i, \vec{s}_j) \to \vec{s}_k \in \vec{S}$

Def.:

$\vec{A} = \langle \vec{X}, \vec{S}, \vec{Z}, \delta, \lambda \rangle$

7.2.4 Jezikovni opis avtomata

Standardna jezika:
- tabela stanj
- diag. preh. stanj

Tabela preh. stanj:

p - stolpcev - po enega za vsako vhodno stanje in

r - vrstic za r mogočih sedanjih stanj avtomata

Pari (x_i, s_j) določajo izhod oz. nasl. stanje v katerega bo prešel avtomat.

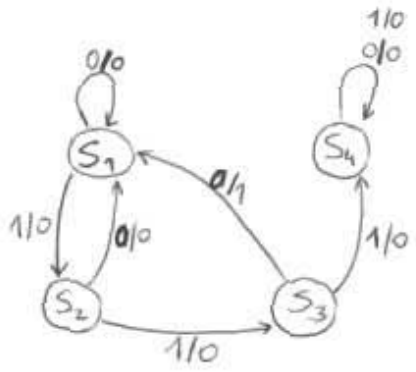
$\ell = \log_2 p$

$n = \log_2 r$

Diag. preh. stanj:

r - vozlišč, iz vsakega vozlišča izhaja p - vej

S	$\Delta^1 S, z$	
	$x_1 = 0$	$x_2 = 1$
S_1	$S_1, 0$	$S_2, 0$
S_2	$S_1, 0$	$S_3, 0$
S_3	$S_1, 1$	$S_4, 0$
S_4	$S_4, 0$	$S_4, 0$



7.3 Vrste končnih avtomatov in 7.4 Način podajanja končnih avtomatov

☒ Funkcija avtomata - Mealy-ev in Moore-ov, razlike, koliko različnih izhodov lahko pri Mealy-evem avtomatu dobiš v enem urinem ciklu, če ga prožiš na negativno fronto.

Mealy: $A = \langle X, S, Z, \delta, \lambda \rangle$ $\quad \Delta^1 s = \delta(s, x)$ $\quad z = \lambda(s, x)$

$\delta: X \times S \to S$

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

Moore: $A = \langle X, S, Z, \delta, \lambda \rangle$ $\quad \Delta^1 s = \delta(s, x)$ $\quad z = \lambda(s)$

$\delta: X \times S \to S$

$\lambda: S \to Z$

Izhodi Mealy-evega avtomata so odvisni od stanja in vhodnih spremenljivk, z razliko od Moorovega, pri katerem so izhodi odvisni samo od stanj. Izhodi Mealy-evega avt. prehitevajo Moore-ovega za pol takta - izhodi Mealy-a se spremenijo sočasno s prehodom v stanje, medtem ko se pri Moore-u spremenijo šele, ko je avt. že v stanju.

7.5 Ekvivalentnost stanj in minimizacija spomina

Stanji s_i in s_j avtomata P sta ekvivalentni samo tedaj, kadar da avtomat za vse možne vhodne sekvence enake izhodne sekvence ne glede na to katero stanje je začetno - s_i ali s_j.

Definicija 1:

Naj bosta P in Q dva popolnoma določena avtomata, ki delirata isto množico možnih vhodnih sekvenc poljubne dolžine.

Naj bo $x_1, x_2, \ldots x_\ell$ ta sekvenca. Stanji $p \in P$ in $q \in Q$ sta ekvivalentni samo, ko je za katerokoli vhodno sekvenco izpolnjen pogoj:

$$\lambda_P(p, x_1, x_2, \ldots, x_\ell) = \lambda_Q(q, x_1, x_2, \ldots x_\ell)$$

Definicija 2:

Dva digitalna avtomata P in Q sta ekvivalentna $P \equiv Q$, če obstoja za vsako stanje $p \in P$ stanje $q \in Q$ tako, da velja $p \equiv q$ in obratno.

$s_1 \equiv s_1$ - zakon refleksivnosti

$s_1 \equiv s_2$, $s_2 \equiv s_1$ - simetričnost

$s_1 \equiv s_2$, $s_2 \equiv s_3 \rightarrow s_1 \equiv s_3$ tranzitivnost

⇓

stanja avtomata razdelimo na skupine ločljivih razredov ekvivalentnosti → minimiziran avtomat bo imel toliko stanj, kolikor je teh razredov v začetnem avtomatu

Teorem: Naj bodo stanja avtomata A razdeljena na ločljive razrede. Z $s_1 \stackrel{\Delta}{=} s_2$ označimo, da sta stanji s_1 in s_2 znotraj istega razreda.

Razdelitev je sest. iz parov ekvivalentnosti, ki vsebujejo ekvivalentna stanja samo takrat, ko za katerikoli par stanj: $s_1 \triangleq s_2$ in isti izhod x_i velja:

1. $\lambda(s_1, x_i) = \lambda(s_2, x_i)$
2. $\delta(s_1, x_i) \triangleq \delta(s_2, x_i)$

} izhodi dveh stanj morajo biti enaki, naslednja st. teh stanj pa morajo biti v istih razredih

Metoda, ki je osnovana na tem se imenuje Huffman – Mealy-ev algoritem, sest. je iz več korakov, bistvena pa sta naslednja:

- zgrajujemo razdvojene množice ekvivalentnih stanj s pomočjo relacije: $\lambda(s_1, x_i) = \lambda(s_2, x_i)$
- nadaljujemo izgradnjo razdvoj. množic s testom naslednjih stanj po relaciji: $\delta(s_1, x_i) \triangleq \delta(s_2, x_i)$

vrstni red

Koraka po potrebi ponavljamo.

7.6 Pretvarjanje avtomatov

7.6.1 Pretvorba Mealy-evega avtomata v Moore-ovega

Moore-ov avt. ima lahko le en izhod na stanje, zato vsako stanje, ki ima v Mealy-u več izhodov, razdelimo na toliko podstanj kolikor je teh izhodov.
V kolikor gre pri Mealy-u iz stanja s_i, ki ima j izhodov ob vhodu x avtomat v neko drugo stanje, gre pri istem vzbujanju v to drugo stanje pri Moore-u avtomat iz vseh j podstanj stanja s_i. To drugo stanje v katerega avtomat preide pa je nedvoumno določeno z izhodom.

(s_{iME}, x_j) – stanje Mealy-evega avt.
x_n – vhodna črka za naslednje stanje v Moore-ovem avt.
$S_{MO}((s_{iME}, x_j), x_n) = (\delta_{ME}(s_{iME}, x_j), x_n)$; $s_i = s_0$
$\delta_{MO}(s_{oMO}, x_n) = (s_{oME}, x_n) = s_{oinMO}$; $s_{oMO} = s_{oME}$

Stanju $S_{ME}(s_{iME}, x_j)$ ustreza stanje s_{ijMO}.

$\lambda_{MO}(s_{ij}) = \lambda_{ME}(s_i, x_j)$; $s_{ij} \neq s_0$

$\lambda_{MO}(s_0) = M$; $s_{0ME} = s_{0MO}$

7.6.2 Pretvorba Mooreovega končnega avtomata v Mealy-evega

Združimo lahko stanja, ki imajo enaka naslednja stanja, neglede na izhod teh naslednjih stanj - tako dobimo novo Mealy-evo stanje s_{ME}.

$\delta_{ME}(s_{ME}, x) = \delta_{MO}(s_{MO}, x)$; $s_{ME} = s_{MO}$

$\lambda_{ME}(s_{ME}, x) = \lambda_{MO}(\delta_{MO}(s_{MO}, x))$

8. ANALIZA IN SINTEZA SINHRONSKIH SEKVENČNIH VEZIJ

8.1 Analiza sinh. sekvenčnih vezij

Analiziramo jih tako, da napišemo tabelo ali diagram za časovno zaporedje vhodov, izhodov in notranjih stanj.

- tabela prehajanja stanj

	nasl. st.		sedanji
sed. st.	vhod x_1 ..	x_m	izhod
			x_1 .. x_n
.	. .	.	.
i	. .	.	.
.	. .	.	.

m-f. f: 2^m stanj
n-vhodov 2^n nasl. stanj

- diagram stanj
- enačba stanj oz. aplikacijska ali spominska enačba je izraz, ki določi kakšna mora biti funkcija, ki določa sedanje stanje, da bo nasl. stanje 1. ($\Delta^1 Q = \ldots$)
- izhodne enačbe - definirajo sek. vezje

8.2 Uvod v sintezo

Sinteza poteka v več zaporednih korakih

1.) Na osnovi besednega opisa algoritma delovanja razvijemo tabelo stanj oz. diagram stanj avtomata.
2.) Opravimo minimizacijo stanj avtomata.
3.) Kodiramo stanja in izberemo tip spominskih elementov.
4.) Določimo tabelo prehajanja stanj in izhodno tabelo.
5.) Določimo vzbujalne in izhodne fje.
6.) Izdelamo shematični načrt avtomata.

Pri točki 3.) je potrebno omeniti, da veljajo določena priporočila glede uporabnosti spominskih celic, kot na primer:

- JK: splošna uporaba
- RS, D: kjer moramo prenašati oz. pomikati podatke (pomični registri, binarni števci)

8.2.1 Sinteza na osnovi jezikovnega opisa

8.2.2. Sinteza vezja iz diagrama stanj

8.2.3. Sinteza vezja iz časovnega diagrama

8.2.4. Sinteza mesopolno opredeljenega avtomata

9. SEKVENČNE PREKLOPNE STRUKTURE

9.1. Števci

Doseg štetja je dan s št. možnih stanj od začetnega stanja dalje
0, 1, 2, ..., m-1, 0, 1, 2, ...
$m \leq 2^n - 1$

Delilniki frekvence: 1 vhod in 1 izhod - izhod se spremeni vsakih d vhodnih impulzov

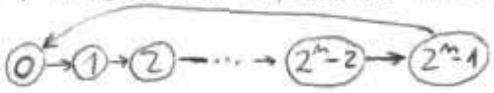
S števcem lahko zgradimo delilnik frekvence in obratno.

Osnovne fje števcev:
1.) pretvorba števila vhodnih impulzov v predpisano kodo in pomnjenje te kode
2.) generiranje predpisanih kodnih sekvenc
3.) ustvarjanje natančnih časovnih zakasnitev

Delitev:
- namembnost - enostavni in večnamenski števci
- št. izhodnih bitov (stopnja števca)
- št. predpisanih stanj po katerih krožijo, kar splošno označujemo kot modul štetja
- kodna sekvenca, ki jo generirajo
- načinu obratovanja (sinhronski ali asinhr.)

9.1.1. Binarni števci

9.1.1.1 Enostaven binarni števec

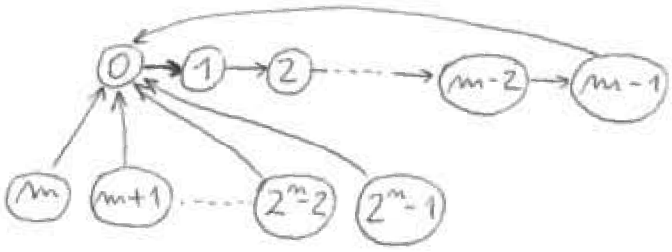


9.1.1.2 Enostaven binarni števec z vključenim prenosom
9.1.1.3 Binarni števec s sinhroniziranim vpisom in sinhr. brisanjem
- krmilni vhodi

9.1.1.4 Binarni števec s paralelnim asinhronim vpisom
9.1.1.5 Binarni števec s sinhronim vpisom in asinhronim brisanjem
9.1.1.6 Vzvratni in dvosmerni števci

9.1.2 Števci po modulu "m"

$0, 1, 2, \ldots, m-1, 0, 1, 2, \ldots$ $\quad m \leq 2^n - 1$



Izolirano stanje števca

Izolirano stanje števca je stanje, ki ni znotraj delovnega cikla. Do izoliranega stanja (oz. stanj) lahko pride, ko je $m < 2^n - 1$. Vezje realiziramo tako, da se števec iz izoliranega stanja vrne v začetno stanje.

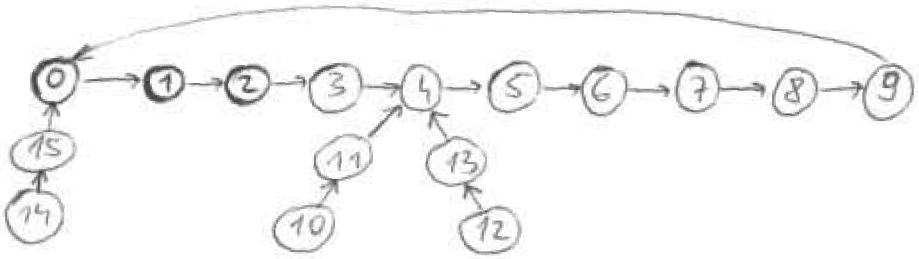
9.1.2.1 Števec po modulu 7

9.1.2.2 Števec po modulu 10 - nestandardni

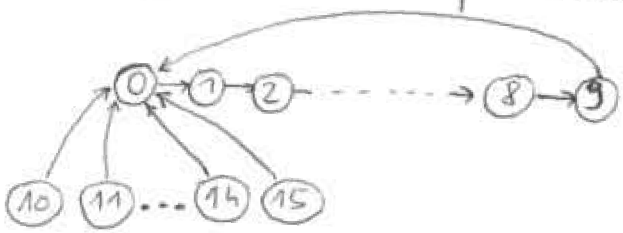
9.1.2.3 Števci s spremenljivim modulom modulu 10

9.1.3. Dekadni števci

9.1.3.1 Dekodni Johnsonov števec

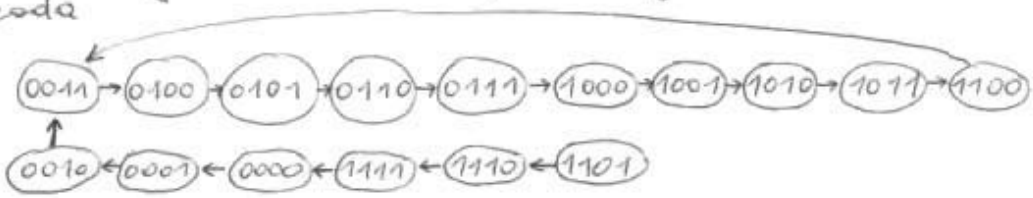


9.1.3.2 Dekadni števec po modulu 10



9.1.3.3 Dekadni števec s kodo IN-3 (EXCESS-3)

- IN-3 koda je za 3 mesta premaknjena klasična BCD koda



9.1.4 Specialni števci

9.1.4.1 Števci z Grayevo kodo

- Grayeva enokoračna koda

9.1.4.2 Krožni števci

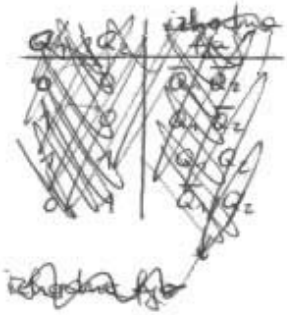
- krožni števec z vsiljeno enico
- samovzbudni krožni števec
- števci s prepletanjem

$D_3 = (Q_1 + Q_3)Q_2$

☒ Razlika med krožnim števcem in števcem s prepletanjem.

Standardni krožni števec potrebuje n pomnilnih celic za štetje po modulu n, števec s prepletanjem pa potrebuje n/2 spom. celic za štetje po modulu n.

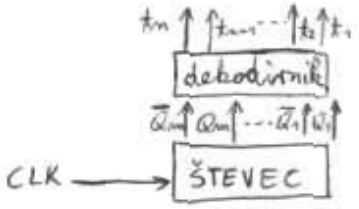
Q_8	Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	izhod
0	0	0	0	0	0	0	1	t_1
0	0	0	0	0	0	1	0	t_2
0	0	0	0	0	1	0	0	t_3
0	0	0	0	1	0	0	0	t_4
0	0	0	1	0	0	0	0	t_5
0	0	1	0	0	0	0	0	t_6
0	1	0	0	0	0	0	0	t_7
1	0	0	0	0	0	0	0	t_8



Q_1	Q_2	Q_3	Q_4	izh. fja
0	0	0	0	$\overline{Q_4}\,\overline{Q_1}$
1	0	0	0	$Q_1\overline{Q_2}$
1	1	0	0	$Q_2\overline{Q_3}$
1	1	1	0	$Q_3\overline{Q_4}$
1	1	1	1	$Q_4 Q_1$
0	1	1	1	$\overline{Q_1} Q_2$
0	0	1	1	$\overline{Q_2} Q_3$
0	0	0	1	$\overline{Q_3} Q_4$

9.1.4.3 Števci s spremenljivo sekvenco stanja

9.1.5 Štetno dekodirna vezja



9.1.5.1 Štetno dekodirno vezje z dekodirnikom 3/6

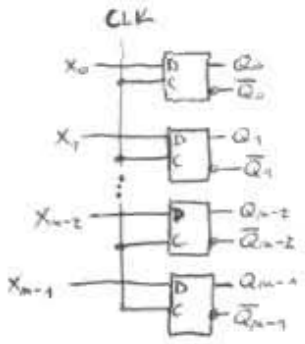
9.1.5.2 Štetno dekodirno vezje s standardnim dekodirnikom 3/8

9.1.5.3 Štetno dekodirno vezje s števcem v Grayevi kodi

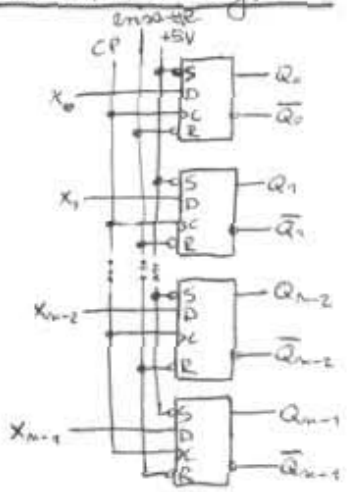
9.1.5.3 Štetno dekodirno vezje zasnovano s števcem s prepletanjem

9.2 Registri

9.2.1 Enostavni pomnilniški registri



$Q_i = \Delta^{-1} x_i$



$Q_i = \overline{R}(\Delta^{-1} x_i)$

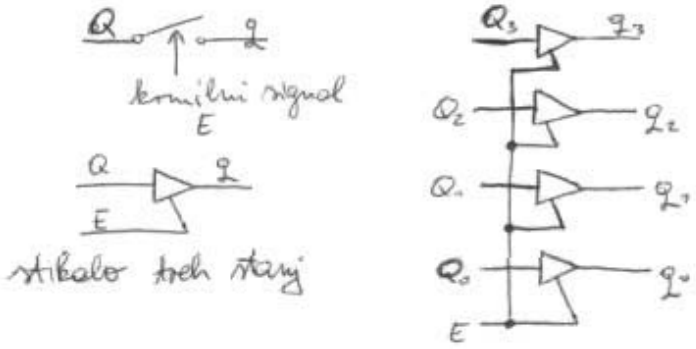
$Q_i = \overline{R}(\Delta^{-1} x_i + \Delta^{-1} Q_i)$

9.22. Naslovni registri

9.2.3 Pomnilniški registri z zatiči

9.2.4 Pomnilniški register s skupnim krmiljenjem

9.2.4. Pomnilniški register s stikali treh stanj



9.2.5 Premikalni registri

9.2.6 Integrirani premikalni registri

9.2.7 Serijsko paralelni premikalni register

9.2.8 Dvosmerni serijsko paralelni premikalni register

$$\Delta^1 Q_i = P_L Q_{i+1} + P_D Q_{i-1} + \bar{P}_D \bar{P}_L Q_i$$

9.2.9. Univerzalni premikalni register

- paralelni zapis
- serijski pomik desno
- — " — levo

Enačbe premikalnega registra (L, D, x)

a) Paralelni zapis
če je $P=1, S_R=0, S_L=0$ se podatki vhodov x_0-x_3 prenesejo v A_0-A_3

b) Pomik desno
če postavimo $S_R=1, P=0, S_L=0$ se podatki v registru pomikajo desno

c) pomik levo
če je $S_L = 1$, $P = 0$, $S_R = 0$ gredo podatki proti levi

$$\Delta Q = P x_i + S_R Q_{i+1} + S_L Q_{i-1}$$

9.3 Sekvenčna aritmetična vezja

9.3.1 Seštevalniki

glej nazaj!

9.3.2 Primerjalniki

$A = a_n, a_{n-1}, \ldots\ldots a_0$
$B = b_n, b_{n-1}, \ldots\ldots b_0$

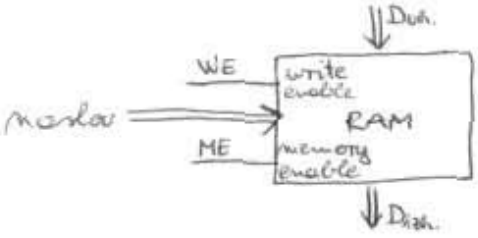
	$A > B$	$A = B$
Vsebina A večja	✓	
Vsebina B večja		
Vsebini enaki		✓

glej nazaj!

9.4 Pomnilniška sekvenčna vezja

9.4.1 Pomnilnik z naključnim dostopom – RAM pomnilnik

Glavna značilnost RAM pomnilnika je možnost spreminjanja vsebine med delovanjem, to je v času uporabe pomnilnika.



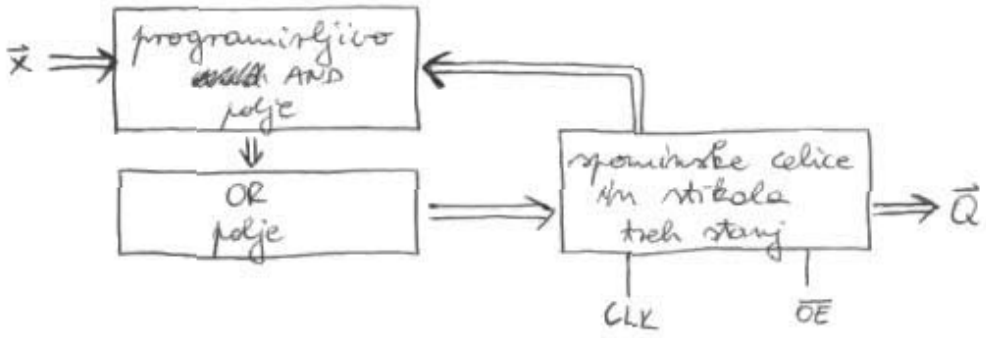
ME	WE	operacija	izhod	
0	0	zadrži	ničdeč	
0	1	zadrži	ničdeč	
1	0	beri	zvezan	naslovna beseda na izhodu
1	1	piši	ničdeč	naslovna beseda zapisana

- Spremljajoči registri:
  - pomnilniško adresni register MAR → preprečuje težave pri prehodnih pojavih ob vpisu podatkovnih besed
  - podatkovni register MDR
- bitne ravnine RAM-a

9.5 PROGRAMIRLJIVA SEKVENČNA VEZJA

9.5.1 PAL arhitektura z dodatnimi spominskimi celicami

- programirljivi vhodno/izhodni priključki
- programirljiva polanteta izhodnih signalov
- vgrajeni XOR elementi med AND-OR poljem in spominskimi celicami



9.5.2 Programirljiva vezja z makro celicami

Za razliko od prejšnjih, ki so večinoma narejena v TTL tehnologiji so to vezja najpogosteje narejena v CMOS tehnologiji EEPROM vezij.

Glavni sklopi so spom. celice in MUX, ki jih je z CMOS zelo enostavno realizirati

Prednosti (zaradi EEPROM tehnologije in MUX):
- enostavno preprogramiranje
- večja kompleksnost, ker odpade okenček za brisanje

9.5.2.1 Makro celica firme ALTERA

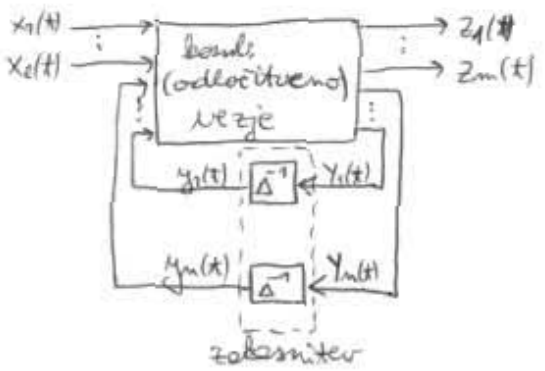
MUX-i omogočajo neodvisno programiranje izhodnih linij povratnih zvez in urinih impulzov za vsako celico posebej.
S-vhodi MUX-ov so kmiljeni z izhodi EPROM-a.
Izhodni MUX: izhod sekvenčen ⇔ izhod kombinacijski
MUX v povratni zvezi: vhod v AND polje priključen na ff ⇔ priključen na zunanji signal

9.5.2.2 Makro celica firme Xilinx

10. ASINHRONSKA SEKVENČNA VEZJA

10.1 Modeli asinhronskih sekvenčnih vezij

10.1.1 Kanalski model asinhronskega sekvenčnega vezja



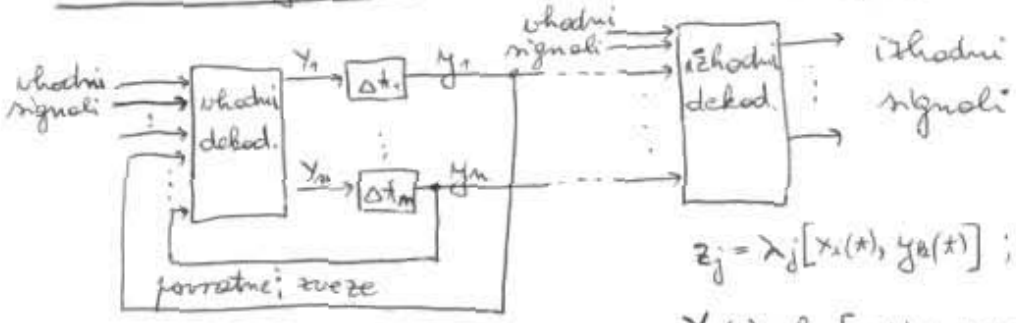
$\vec{x} = (x_1(t), x_2(t), \dots x_i(t), \dots, x_l(t))$ - primarni vhodni vektor

$\vec{y}$ - sekundarni vhodni vektor

$\vec{z}$ - vektor izhodnih spr.

$\vec{Y}$ - vektor vzbujalnih spr.

10.1.2 Serijski model asinhr. sekv. vezja



$z_j = \lambda_j [x_i(t), y_k(t)]$; $j = 1, 2, \dots, m$

$Y_k(t) = f_k [x_i(t), y_k(t + \Delta t)]$

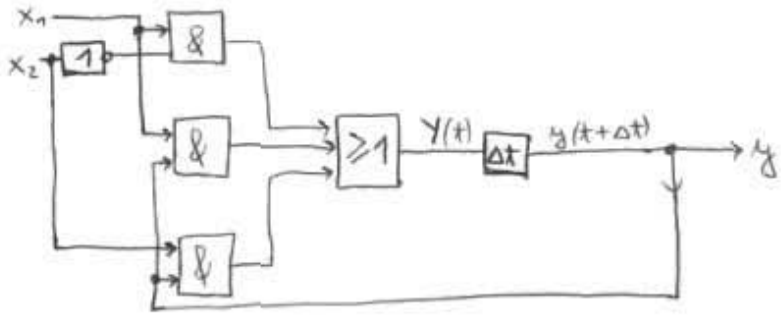
$i = 1, 2, \dots, l$; $k = 1, 2, \dots, n$

10.2 Analiza asinhr. sekv. vezij

Označanje asinhr. sekv. vezij lahko podamo na več načinov:

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

10.2.1 Osnovna oblika asinhr. vezja (Fundamental mode)



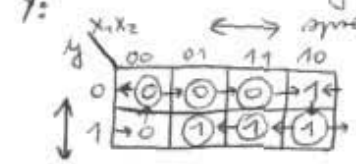
$Y(t) = y(t+\Delta t)$
ali pa
$Y(t) \neq y(t+\Delta t)$

Če velja prvi pogoj je vezje v stabilnem stanju. Za drugi pogoj pa pravimo, da je vezje na prehodu iz enega stabilnega stanja v drugo. To stanje je torej nestabilno.

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

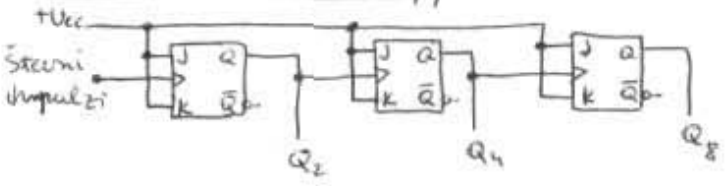
Karnough:

Y:



① - stabilno
1 - nestabilno

10.2.2 Analiza ripple števca



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

$\Delta t = 10ns, n = 8, t_z = 80ns$

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

10.3 Asinhronski sekv. avtomati

10.3.1 Uvod in osnovne karakteristike

Kadar je "širina" vrinih impulzov primerljiva z zakasnitvami posameznih sklopov vezja, moramo delovanje vezja obravnavati z asinhronsko analizo.
Vrhi impulz, če obstaja, postane tako neodviseh vhod.
Spremembe stanj so nekvidistantne
Privzeti moramo določene predpostavke in omejitve (npr. Huffmanov pogoj!).
Spremembe vhodov se smejo dogajati samo v časovnih intervalih v katerih se celotno vezje zadržuje v stabilnem stanju, povzročenem s prejšnjo spremembo na vhodu.

10.3.1.1 Jezik avtomata in tabela prehajanja osnovnih stanj

Jezik avtomata mora biti metodološko prirejen sintezi in analizi teh vezij, ko podobno kot smo imeli pri sinhr.
Kljub temu pa nastopajo nekatere pomembne razlike, ena teh je:
Totalno stanje avtomata - kombinacija vhodnega in notranjega stanja.
Množica totalnih stanj:

$T^{\wedge} = \{ s_1^t, s_2^t, s_3^t, \ldots s_n^t \}$ pri čemer je na primer:

$s_i^t = s_j^k = S(s_j, x^k)$ kombinacija notranjega stanja $s_i \in S$ in vhodnega stanja $x^k \in X$

Analogno vpeljemo tudi oznake izhodnih stanj:

$z_m^n \in Z$

10.3.1.2 Tabela posamičnih izhodnih prehodov – PIP tabela

Asinhronski avtomat deluje v normalnem načinu, če po PIP tabeli vsako prehajanje pelje direktno v stabilno stanje

10.3.1.3 Tabela večkratnih izhodnih prehodov – VIP tabela

Če avtomat v določenem stanju vzbudimo z določenim vhodnim stanjem, avtomat ne more več preiti v stabilno stanje.

Vse tabele, ki podajajo normalni način delovanja, podajajo hkrati tudi osnovni način delovanja, obratno ne velja vedno.

10.3.1.4 Impulzno vzbujanje asinhr. avtomatov

10.3.2 Metode minimizacije asinh. avtomatov

10.3.2.1 Primitivna tabela prehajanja stanj

Na osnovi verbalnega, matričnega opisa, ali pa grafičnega opisa v obliki diagrama prehajanja stanj naredimo začetno obliko tabele, ki se imenuje primitivna tabela preh. stanj ali tabela preh. osnovnih stanj.
Izhajamo iz predpostavke, da za vsako vhodno vzbujanje obstaja stabilno stanje v avtomatu. Vsakemu stabilnemu (totalnemu) stanju dodelimo svojo vrstico v tabeli – s tem vnašamo redundanco, a vnos je potreben zaradi maksimalne zanesljivosti glede točnosti in pristnosti algoritemskega prikaza.

10.3.2.2 Reduciranje primitivne tabele prehajanja stanj

- že znane metode iskanja ekvivalentnih razredov stanj
- postopek spajanja
- posebnosti, ki so vezane na specifičen način delovanja

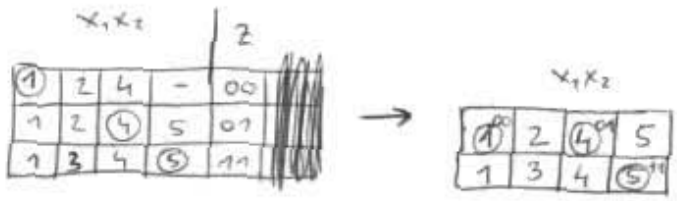
V isti vrstici se lahko nahaja več stabilnih (totalnih) stanj, do katerih sprememba prihaja izključno zaradi sprememb vhodnih stanj.

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.

10.3.2.3 Spajanje vrstic

Dve ali več vrstic lahko spojimo samo v primeru, da imata/imajo v istoležnih 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 ni kršeno pravilo identičnosti v ostalih stolpcih vseh spojenih vrstic



10.4 Kodiranje notranjih stanj in njihovo prehajanje

10.4.1 Kodiranje notranjih stanj

- se razlikuje od kodiranja pri navh. vezjih
- od kodir. je odvisna tudi zanesljivost delovanja
- graf sosednih prehodov ⇔ medsebojno sosednja stanja

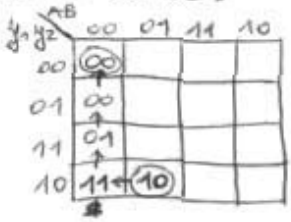
10.4.2 Prehajanje stanj v asinhr. avtomatih

- ciklično prehajanje notranjih stanj
- sočasno — " —

Tipi prehodov stanj v asinhronskih avtomatih

- ciklično prehajanje
- sočasno — " —

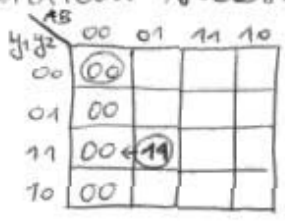
Pri cikličnem prehajanju stanj se istočasno spreminja samo ena sekundarna spremenljivka ob. stanje povratne zveze:



koda "11" pove, naj gre v vrstico 11

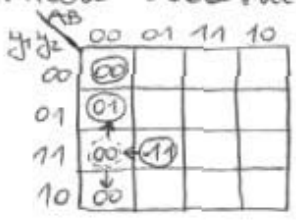
Pri sočasnem prehajanju nastopata dve možnosti tega prehajanja - kritično in nekritično sočasno prehajanje.

Nekritičen sočasni prehod notranjega stanja:



obe y sta v stanju prehajanja

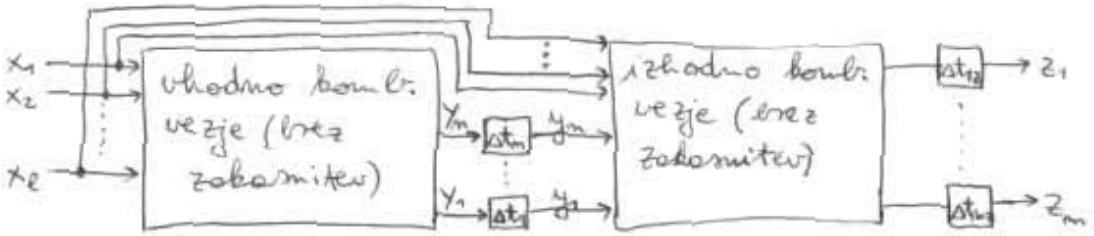
Kritičen sočasni prehod notr. stanja:



napačen prehod

pravilen prehod

10.5 Zakasnitve v asinhronskih vezjih



Končno stanje vektorja $\vec{z}^1$ lahko opazujemo šele po $(\Delta t_1)_{min}$ za vzbujanjem $\vec{x}^m_1$, pri čemer mora veljati:

$$(\Delta t_1)_{min} \geq (\Delta t_j^\circ); \quad j = 1, 2, \ldots, m$$

V zakasnitve je lahko vključen tudi čas preleta po povezavah.

10.6 Hazardni prehodi

10.6.1 Hazardni prehodi v kombinacijskih vezjih

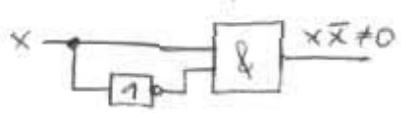
Vedno nastopajo zakasnitve med trenutkom, ko se spremeni vhod v nek element in trenutkom, ko se pojavi ustrezna sprememba tudi na izhodu.

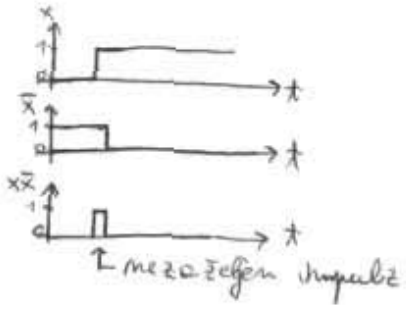
Hazard (v kombi. vezjih)

Definicija:

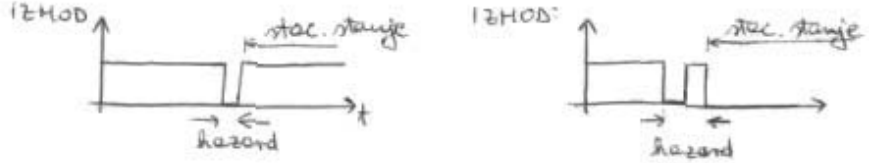
Hazard je dejansko ali potencialno nepravilno delovanje vezja zaradi zakasnitev v vezju.

Enostaven primer:





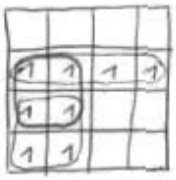
Obstaja več vrst hazarda:
- statični hazard je enkratna sprememba izhodnega signala prehodnega značaja, ko ne pričakujemo spremembe na izhodu; nastopi v primeru, ko bi morala izhodna vrednost ostati enaka, kljub spremembi vhodov
- dinamični hazard: do njega pride kadar pričakujemo spremembo izhoda, ta pa se spremeni 2x - najprej pride do želene spremembe, nato pa se pojavi še začasni prehodni pojav z napačno vrednostjo



a) statični b) dinamični

Statični in dinamični hazard je mogoče odpraviti z isto metodo dodajanja novih elementov - t.j. konjunkcij v disjunktivni obliki.

Veitch:



- redundantni člen

↳ vsi minitermi morajo biti medsebojno odvisni

Statični hazard lahko odpravimo tudi z dodajanjem zakasnitev.

Dinamični hazard je v splošnem posledica:
- konjunktivne oblike realizacije fje
- predolgih povezav pri ekstremno hitrih vezjih
* se posebej je občutljiv na NOR/NAND izvedbe - inverter na izhodu

10.6.2 Hazard v asinh. vezjih

Hazard (asinh. vezja)

Zaradi povratne(-ih) zveze(k) predstavlja še večji problem pri načrtovanju in sami izvedbi.

Bistven ali vgrajen hazard (Essential hazard):
- pojavlja se le v vezjih z dvema ali več povratnimi zvezami
- če obstoja je vezan na ure, t.j. na zakasnitve in na vhodne specifikacije o delovanju vezja

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

Bistveni ali vgrajeni hazard se pokaže šele, če ena zakasnitev v vezju v primerjavi z ostalimi močno povečamo.

Razlika med sinhr. in asinhronskim avtomatom (vloga ure), kako se razlika vidi iz diagrama prehajanja stanj.

Sinhronski avtomat potrebuje za delovanje ura - spremembyanje vhodnih spren. je ekvidistantno, asinhron. avtomat ure ne potrebuje, odreagira takoj, ko se spremeni vhodna spren. - deluje hitreje.

V diagramu preh. stanj asinhr. avtomat prepoznamo po prehodih med stanji, ki jih definirajo vhodne spremenljivke - od teh pa se lahko naenkrat spremeni le ena (enokoračna koda), enako velja za sekundarne vhodne spremenljivke.