ELEKTRONIKA II - VAJE

12.3.2022

Ponovimo pravila Boolove algebre

- Zamenljivost: $A \cdot B = B \cdot A$, $A + B = B + A$
- Združljivost: $A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$

  $A + B + C = (A + B) + C = \dots$

- Prioriteta: Najprej oklepaji ( ) → NOT → AND → OR

Par primerov:

i) $A + 1 = 1$ $\quad A + \bar{A} = 1$ $\quad A \cdot A = A$

$A \cdot 1 = A$ $\quad A \cdot \bar{A} = 0$

$A \cdot \bar{B} + \bar{A} \cdot B = A \oplus B$ → exclusive OR (XOR)

DeMorgan: $\overline{A \cdot B} = \bar{A} + \bar{B}$ in $\overline{A + B} = \bar{A} \cdot \bar{B}$

ii) $A + A \cdot B = A(\underbrace{1 + B}_{1}) = A \cdot 1 = A$

iii) $(A+B)(A+C) = \overbrace{A \cdot A}^{A} + B \cdot A + A \cdot C + B \cdot C = A(\overbrace{1 + B + C}^{1}) + B \cdot C =$

$= A + B \cdot C$

iv) $\underline{A + \bar{A} \cdot B} = A + \overline{\overline{\bar{A} \cdot B}} \overset{\text{DeM.}}{=} A + \overline{(\bar{\bar{A}} + \bar{B})} = A + \overline{(A + \bar{B})} = A + \overline{\overline{(A + \bar{B})}} =$

$= A + \overline{\bar{A} \cdot (\bar{A} + \bar{\bar{B}})} \overset{\text{DeM.}}{=} A + \overline{\bar{A} \cdot (A + B)} = A + \overline{\underbrace{A \cdot A}_{0} + \bar{A} \cdot B} = A + \overline{\bar{A} \cdot \bar{B}} = A + \bar{\bar{A}} + \bar{\bar{B}} = \underline{A + B}$

To je ZELO uporabna identiteta! Do nje lahko pridemo tudi na lepši način:

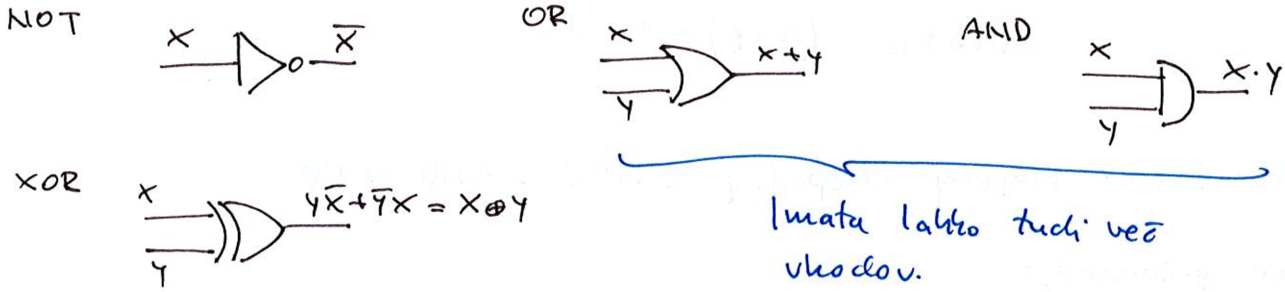
$\underline{A + \bar{A} \cdot B} = \underbrace{A(1 + B)}_{\text{Efektivno množimo z } 1 = 1 + B} + \bar{A} \cdot B = A + AB + \bar{A} \cdot B = A + B(\underbrace{A + \bar{A}}_{1}) = \underline{A + B}$

$A\cdot B\cdot C + \bar{A} + A\bar{B}C = AC(B+\bar{B}) + \bar{A} = AC + \bar{A} = \bar{A} + C$ (Uporabimo prejšnjo identiteto)

$(A\cdot\bar{B}(C+BD) + \bar{A}\cdot\bar{B})\cdot C = (A\bar{B}C + A\bar{B}BD + \bar{A}\bar{B})C = A\bar{B}C + \bar{A}\bar{B}C =$

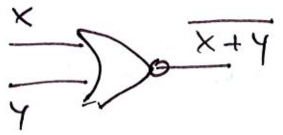
$= C\bar{B}(A+\bar{A}) = CB$

Logične operacije izvajamo z logičnimi vrati. Logično...

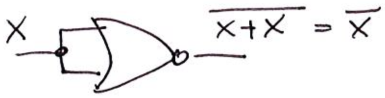


Imata lahko tudi več vhodov.

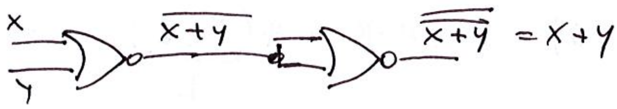
Sestavimo prve tri osnovne operacije le z uporabo vrat NOR:



a) NOT $X \to \bar{X}$

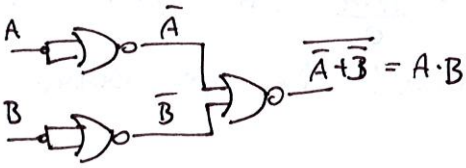


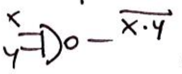
b) OR Kaj je OR drugega kot $\overline{NOR}$?



c) AND Želimo dobiti $A\cdot B$, lahko pa poskusimo dobiti $\overline{A\cdot B}$ in potem negirati, kar znamo.

$\overline{A\cdot B} = \overline{A}+\overline{B}$ (To znamo) $\longrightarrow A\cdot B = \overline{\bar{A}+\bar{B}}$



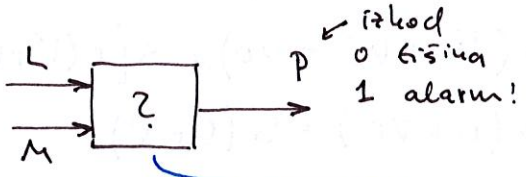
Na podoben način lahko napravimo vse iz vrat NAND  . Izkaže se, da vrata NOR in NAND vsaka zase predstavljajo kompleten sistem logičnih funkcij, dočim to NE drži za OR in AND, ki vedno potrebujejo še NOT vrata.

9.3.2022

Napravimo vezje, ki opozori, če smo v avtu pustili prižgano luč

Vhodni parametri: L luči 0 ugasnjene, 1 prižgane

M motor 0 ugasnjen, 1 teče

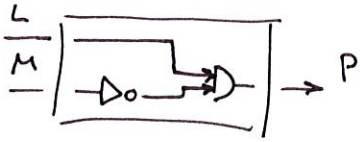


Napišemo tabelo možnih vhodov in želenih izhodov:

L	M	P
0	0	0
0	1	0
1	0	1
1	1	0

← Želimo, da alarm piska takrat, ko je ključ prižgana luč in motor ugasnjen.

$P = L \cdot \overline{M}$

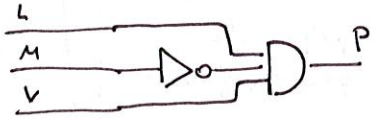


Lahko dodamo še parameter V, ki pove, če so vrata odprta ali zaprta V 1 odprta, 0 zaprta

Nova tabela

L M V	P
0 0 0	
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	1
1 1 0	
1 1 1	

$P = L\overline{M}V$



Vezje, ki upravlja z domačim alarmom

Na vratih in oknih imamo stikala, ki prevajajo, če so odprta ali zaprta. Imamo tudi glavno stikalo, ki vklaplja in izklaplja stikalo.

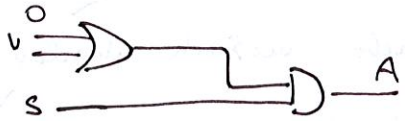
V ... 1 odprto, 0 zaprto
O ... 1 odprto, 0 zaprto

Tabela

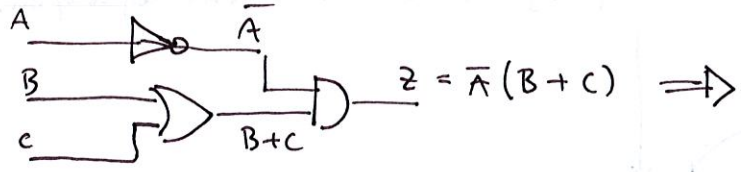
Želimo, da alarm tuli, ko je gl. stikalo vklop. in so odprta vratca ali okna

V	O	S	A
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$A = \bar{V}OS + V\bar{O}S + VOS$$
$$= S(\bar{V}O + V\bar{O} + VO) = S[O(\bar{V}+V) + V\bar{O}]$$
$$= S(O + V\bar{O}) = S(O+V)$$



Iz vezja zapišimo tabelo: zdaj gremo v obratni smeri



A	B	C	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

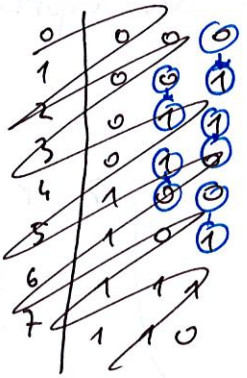
Grayeva koda

Klasično binarno štejemo takole (primer za 3 bite)

0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Težava tu je v tem, da kar nekajkrat obrnemo več kot le bit, kar lahko pripelje do hudih napak pri majhnih vhodnih napakah

Pri Grayevi kodi to napako odpravimo (ozir. težavo) tako, da vedno spremenimo le en bit. Ta je vedno skrajno desni, a tako, da ne ponavljamo stanj.



0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0

Uporabnost poznavanja Grayeve kode se bo kmalu pokazala.

Karnaughove mape

Začnemo z relativno kompleksno tabelo

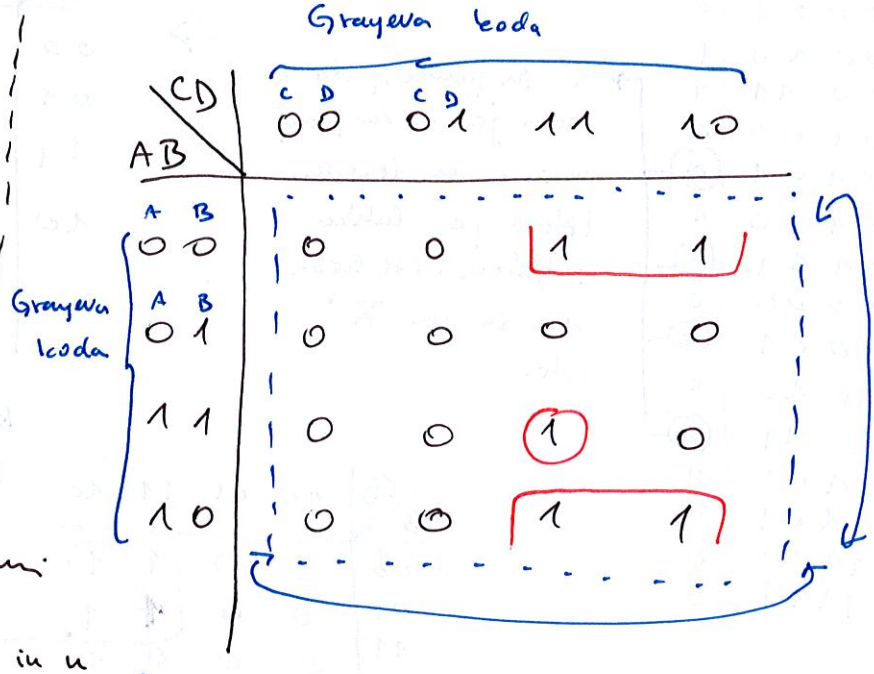
A	B	C	D	R
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Začnemo klasično:

$\bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + A\bar{B}C\bar{D} + A\bar{B}CD + ABCD =$

= kar lahko poenostavljamo v nedogled

Lahko se pa naučimo novega načina z uporabo Grayeve kode.

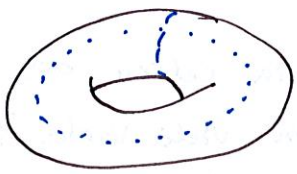


V tabeli tedaj poiščemo otočke enic z dimenzijami

$2^m \times 2^n$, kjer sta m in n nenegativni celi števili.

} Ta pogoj je KLJUČEN!!

Tabela se nadaljuje cikljično preko meja - lahko bi jo zvili v torus.



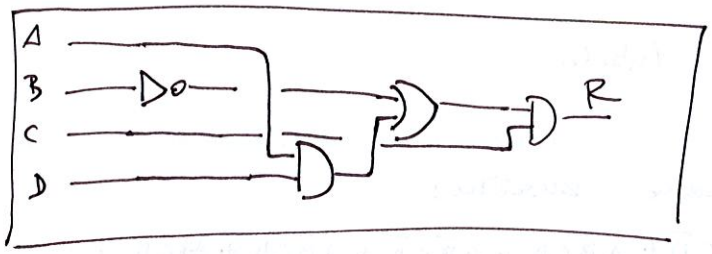
Seveda lahko izberemo različne otočke. Rešitve so po poenostavljenju enake oz. ekvivalentne.

Mi se lotimo tako

○ : ABCD → ~~$\bar{B}\cdot C$~~

□ : Znotraj otočka se spremenita bita B in D, kar pomeni, da ju lahko ignoriramo:

$\Rightarrow \quad R = ABCD + \overline{B}\cdot C = C\left(ABD + \overline{B}\right) = C\left((AD)\cdot B + \overline{B}\right) = C\left(AD + \overline{B}\right)$



Še malo Karnaughja

16.3.2022

A	B	C	D	z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	X
1	0	0	0	0
1	0	0	1	X
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

"X" tu pomeni, da nam je vseeno, kaj dobimo na izhodu. Izhod je lahko poljuben. Ni treba, da so vsi "X" isti.

⇒

AB \ CD	00	01	11	10
00	0	0	1	1
01	0	X	X	1
11	0	0	0	0
10	0	X	X	0

Ko malo pogledamo, vidimo, da se nam najbolj izplača zapolniti tabelo tako

AB \ CD	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	0	0

Na ta način dobimo zelo hitro rešitev.

$\Rightarrow z = \overline{A}\cdot C$

Zakaj delujejo Karnaughejeve mape?

Tu je bistvena uporaba Grayeve kode, kjer se vsaka celica od sosednjih razlikuje le za enega od bitov. Če bi vzeli vsako "1" posebej (ne bi združevali v območje), bi dobili rezultat z vsotami tipa:

$$\ldots + ABCD + ABC\overline{D} + \ldots = \ldots + ABC(D + \overline{D}) + \ldots = \ldots + ABC + \ldots$$

S Karnaughejevo mapo grafično lažje in hitreje pridemo do istega rezultata.

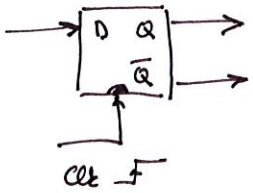
Zakaj $2^m \times 2^n$ celice? Ker nova ~~vsaka~~ bit, ki se spremeni znotraj območja, nastopati s svojo negacijo, da se lahko pokrajša v 1.

Flip flop

Flip flop je v osnovi 1-bitna spominska celica. Sestavimo ga lahko iz logičnih vrat, kar je pa pomembno je naslednje:

Ponavadi sta v uporabi dve verziji

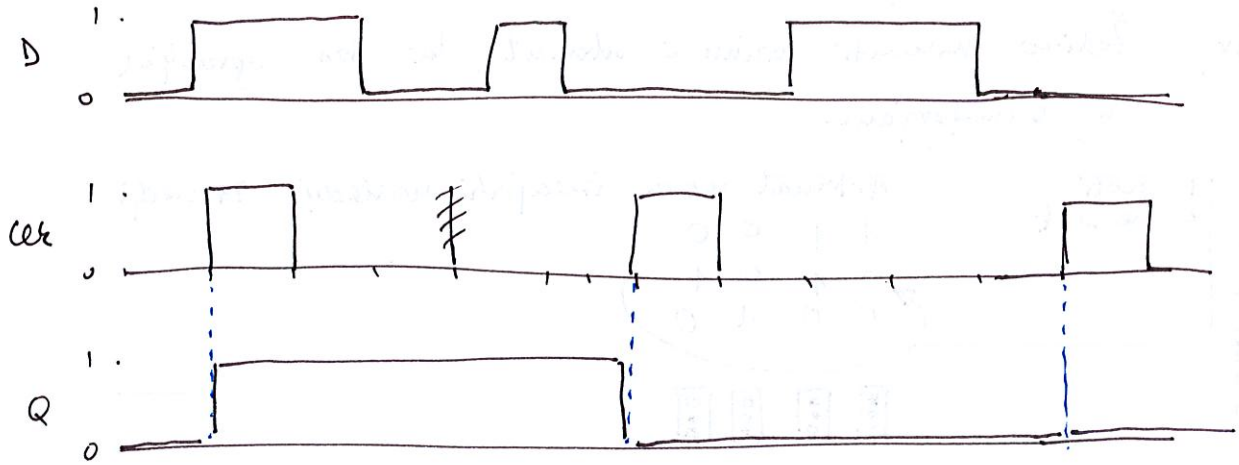
① D-FF



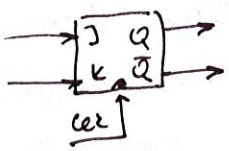
D	Q^t
0	0
1	1
ni clk	Q ← ohranja prejšnje stanje

← po prehodu ure

Za lažje razumevanje:



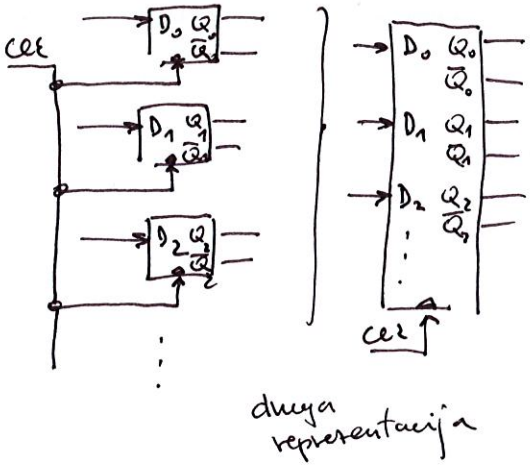
② JK - flip flop



J	K	Q^t	
0	0	Q	← Pri J=K=0 drži prejšnje stanje
0	1	0	← oz. kot D-FF
1	0	1	
1	1	$\bar{Q}$	← preklaplja med 0 in 1 ni vsake clk.

Iz flip-flopov sestavimo registre = večbitne spominske celice

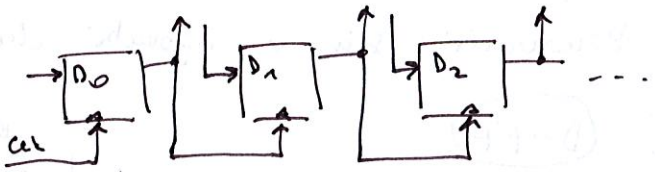
Sinhroni register



druga reprezentacija

Asinhroni

Kot primer



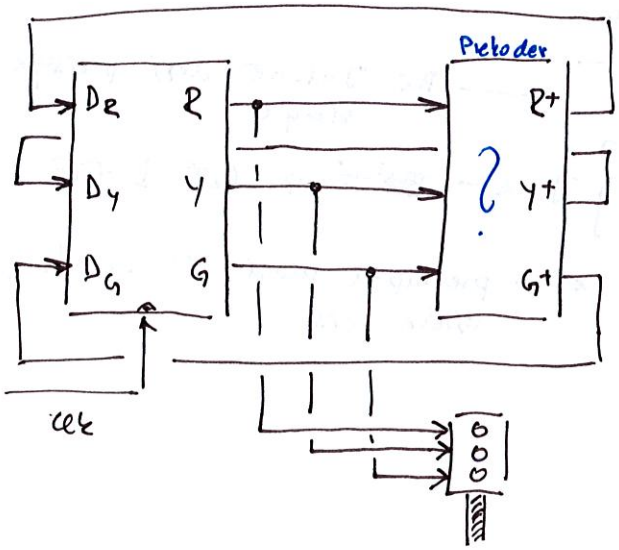
Z uporabo registrov in logike tvorimo avtomate - preprosta verja ki opravljajo specifične naloge.

Semafor Želimo narediti sinhroni avtomat, ki bo upravljal s semaforjem.

Avtomat mora izvajati naslednje zaporedje:

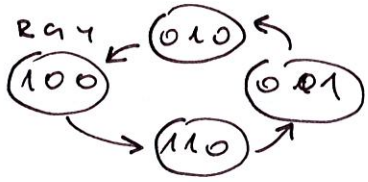
1	1	0	0
0	1	0	1
0	0	1	0

Takšne naloge se klasično lotimo z registrom in preklopljem, ki izračuna naslednje stanje, ki se mora prepisati v register. Ker delamo s 3-biti, vzamemo 3-bitni register

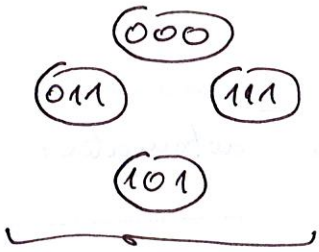


← To je celotna shema vezja. Glavna naloga je najti pravi preklopl.

Začnemo tako, da si pomagamo z diagramom stanj



Po teh stanjih mora krožiti naš avtomat.



Tod so še preostala možna stanja, ki se ne smejo zgoditi. Če že pride do kakšnega od njih, se mora v končnem številu stanj zapeljati v osnovno zanko.

Če imamo srečo, nam lahko že rešitev osnovne zanke poskrbi za "prepredanja stanja". Ker pa to ne velja vujno, kar sami določimo, kaj se mora z njimi zgoditi.

Daleč najlažje je vsako od teh stanj poslati v **100**. To je tudi najvarnejše s stališča prometne varnosti. Po ~~vsaki~~ kakršnikoli napaki naj semafor kaže rdečo luč, da se promet ustavi in umiri. Možne so seveda tudi druge rešitve.

S tem se lotimo tabele

	R	Y	G	R⁺	Y⁺	G⁺
	0	0	0	1	0	0
⚀	0	0	1	0	1	0
⚁	0	1	0	1	0	0
	0	1	1	1	0	0
⚂	1	0	0	1	1	0
	1	0	1	1	0	0
⚃	1	1	0	0	0	1
	1	1	1	1	0	0

Rešimo za vsako barvo posebej:

(R) Poskusimo rešiti za $\overline{R}$

$$\overline{R^+} = \overline{R}\,\overline{Y}\,G + R\,Y\,\overline{G}$$

$$R^+ = \overline{\overline{R}\,\overline{Y}\,G + R\,Y\,\overline{G}} = \overline{\overline{R}\,\overline{Y}\,G} \cdot \overline{R\,Y\,\overline{G}} =$$

~~$= (\overline{\overline{R}\,\overline{Y}} + \overline{G})(\overline{R} +$~~

$$= (R + Y + \overline{G})(\overline{R} + \overline{Y} + G) =$$

$$= R\overline{R} + R\overline{Y} + RG + Y\overline{R} + Y\overline{Y} + YG + \overline{G}\overline{R} + \overline{G}\overline{Y} + \overline{G}G =$$

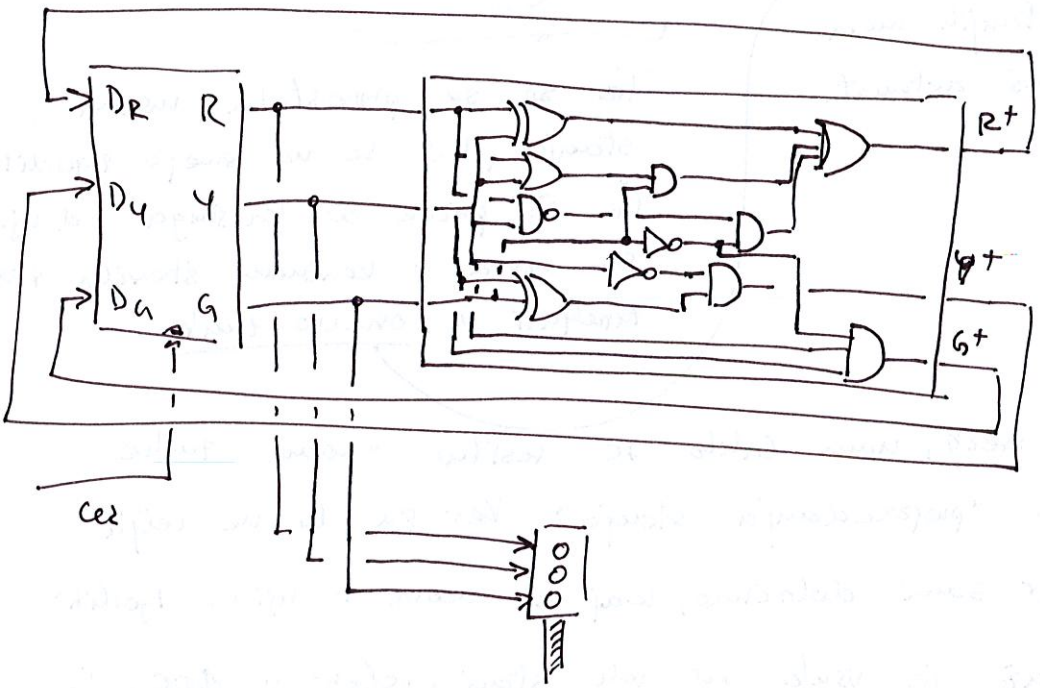
$$= R \oplus Y + G(R+Y) + \overline{G}(\overline{R}+\overline{Y})$$

$$= R \oplus Y + G(R+Y) + \overline{G}(\overline{RY})$$

(Y) $Y^+ = \overline{R}\,\overline{Y}G + R\overline{Y}\,\overline{G} = \overline{Y}(\overline{R} \oplus G)$

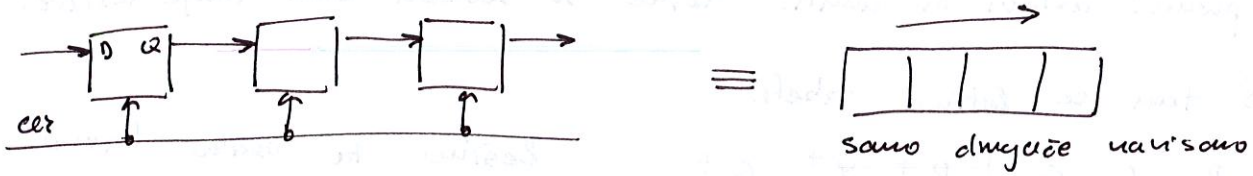
(G) $G^+ = RY\overline{G}$

Končna shema avtomata:

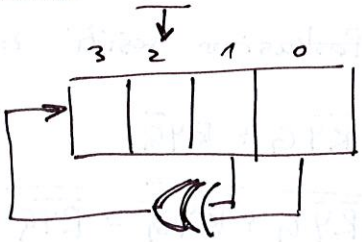


Linear feedback shift register LSFR

23.3.2022

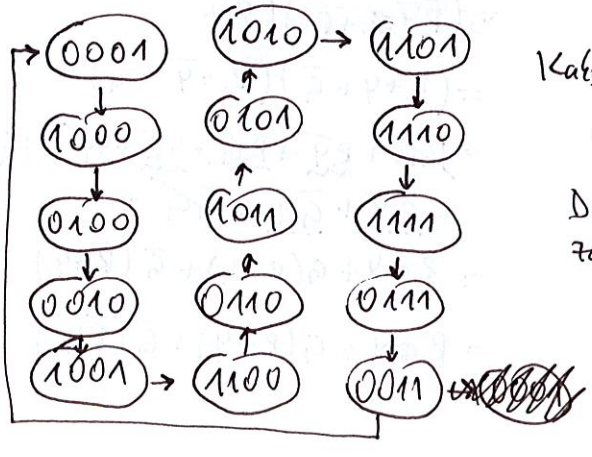


Naprimer tak avtomat in pogledamo kaj počne



Pogledamo diagram stanj. Začnemo s stanjem (0000)

(0000) Vidimo, da se to stanje ohranja. Pa začnimo z 0001



Kakšne so te vrednosti v desetiškem sistemu?

1, 8, 4, 2, 9, 12, 6, 11, 5, 10, 13, 14, 15, 7, 3

Dobimo (psevdo)naključno zaporedje - Lahko bi ga zapisali kot

$$B^{(3)}_{n+1} = B^{(1)}_n \oplus B^{(0)}_n$$

Na takšen princip delujejo velike generatorji psevdonaključnih števil.

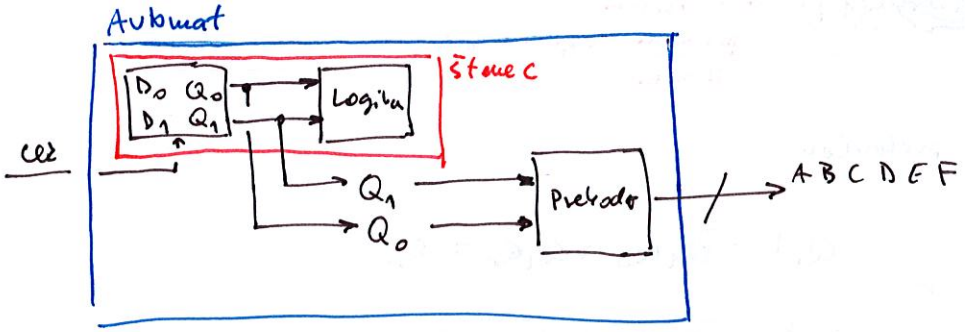
-x-

Sestavi sinhroni avtomat, ki izvaja naslednje zaporedje stanj

A	B	C	D	E	F
0	0	1	0	1	0
0	1	1	1	0	1
1	0	0	1	1	0
1	1	1	1	0	1

Naslednje smo delali tako: Napisali diagram stanj (001010) -> (011101) -> ... in nato zapisali tabelo prehodov. Tu imamo 6-bitov, kar pomeni 2^6 možnih stanj in bi nam vzelo veliko časa, da bi potrebeli za preposredana stanja.

Po premisleku opazimo, da krožimo le preko štirih stanj, kar nas na idejo, da avtomat sestavimo iz 2-bitnega (sinhronega) števca in prekodirja.



Globa shema avtomata

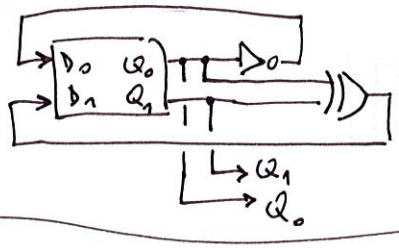
Torej se lotimo najprej števca:

$Q_1 Q_0$	$Q_1^+ Q_0^+$
0 0	0 1
0 1	1 0
1 0	1 1
1 1	0 0

Tako števec dela 0, 1, 2, 3, 0, 1, 2, 3, 0 ...

$$\Rightarrow \quad Q_1^+ = \bar{Q}_1 Q_0 + Q_1 \bar{Q}_0 = Q_1 \oplus Q_0$$

$$Q_0^+ = \bar{Q}_1 \bar{Q}_0 + Q_1 \bar{Q}_0 = \bar{Q}_0$$

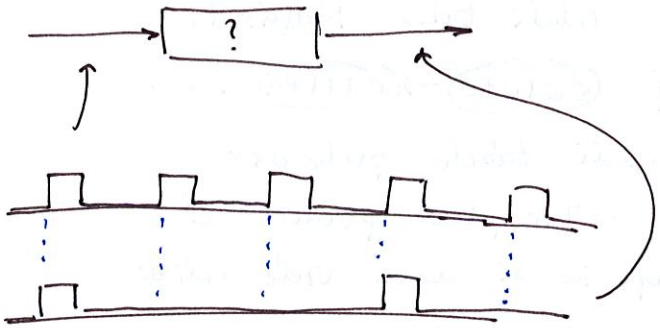


Kako pa iz Q_1 in Q_0 dobimo A B C D E F?

Q_1	Q_0	A	B	C	D	E	F
0	0	0	0	1	0	1	0
0	1	0	1	1	1	0	1
1	0	1	0	0	1	1	0
1	1	1	1	1	1	0	1

- $A = Q_1 \bar{Q}_0 + Q_1 Q_0 = Q_1$
- $B = \bar{Q}_0 \bar{Q}_1 + Q_0 Q_1 = Q_0$
- $\bar{C} = Q_1 \bar{Q}_0 \Rightarrow C = \overline{Q_1 \bar{Q}_0} = \bar{Q}_1 + Q_0$
- $\bar{D} = \bar{Q}_1 \bar{Q}_0 \Rightarrow D = \overline{\bar{Q}_1 \bar{Q}_0} = Q_1 + Q_0$
- $E = \bar{B} = \bar{Q}_0 \quad F = B = Q_0$

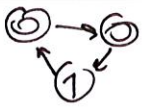
Stroj, ki deli frekvenco s 5



Kako se lotimo tega? Iščemo tesitev, ki ho ponavljala zaporedje

0,0,1,0,0,1,0,0,1,0,...

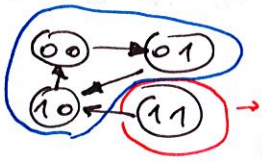
pod diktatom zunanjega ure signala CL.



⇒ Če vzame 2-bitni register in možna stanja

Q_1	Q_0
0	0
0	1
1	0
1	1

→ Vidimo, da se nam iskano zaporedje skriva že nekje. Diagram vseh možnih stanj bi bil potem:



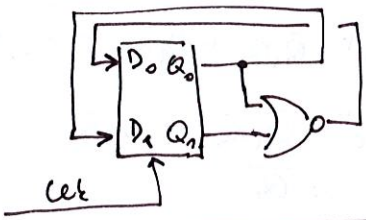
→ V našem primeru "prepovedano" stanje.

Nadaljujemo s tabelo prehodov

Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	1
0	1	1	0
1	0	0	0
1	1	1	0

$$\Rightarrow \quad Q_1^+ = \overline{Q}_1 Q_0 + Q_1 \overline{Q}_0 = Q_0$$

$$Q_0^+ = \overline{Q}_1 \overline{Q}_0 = \overline{Q_1 + Q_0}$$

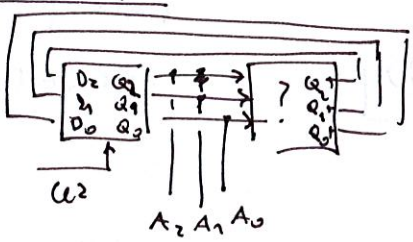


Še en sinhron avtomat za vajo

30.3.2022

A_2	A_1	A_0	~~Q_2^+~~	Q_1^+	Q_0^+
0	0	0		0	1
0	1	0		1	0
1	0	0		1	0
1	1	0		0	1
0	1	1		0	0
0	0	1		1	0
1	0	1		1	1
1	1	1		0	0

(označimo tabelo za Q^+)



~~$Q_2^+ = Q_2 Q_1 Q_0$~~

- Q_2^+

$Q_2 Q_1$ \ Q_0	0	1
0 0	0	(1)
0 1	(1)	0
1 1	0	0
1 0	(1)	(1)

$Q_2^+ = \bar{Q}_1 Q_0 + \bar{Q}_2 Q_1 \bar{Q}_0 + Q_2 \bar{Q}_1 \bar{Q}_0 = \bar{Q}_1 (Q_0 + Q_2 \bar{Q}_0) + \bar{Q}_2 Q_1 \bar{Q}_0$

~~$= \bar{Q}_1 Q_0 + \bar{Q}_0 (\bar{Q}_2 Q_1 + Q_2 \bar{Q}_1)$~~ $= \bar{Q}_1 (Q_0 + Q_2) + \bar{Q}_2 Q_1 \bar{Q}_0$

~~$= \bar{Q}_1 Q_0 + \bar{Q}_1 Q_2 + \bar{Q}_2 Q_1 \bar{Q}_0$~~

- Q_1^+

$Q_2 Q_1$ \ Q_0	0	1
0 0	(1)	0
0 1	0	0
1 1	(1)	0
1 0	(1)	(1)

$Q_1^+ = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + Q_2 Q_1 \bar{Q}_0 + Q_2 \bar{Q}_1 = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + Q_2 (Q_1 \bar{Q}_0 + Q_1)$

$= \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + Q_2 (\bar{Q}_0 + Q_1) = \overline{Q_2 + Q_1 + \bar{Q}_0} + Q_2 (Q_0 + Q_1)$

- Q_0^+

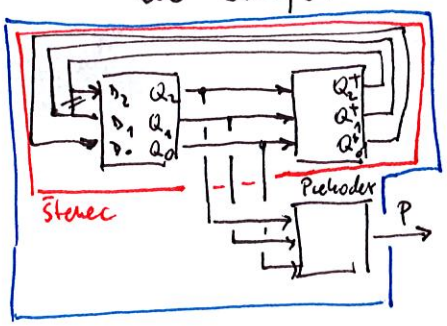
$Q_2 Q_1$ \ Q_0	0	1
0 0	0	1
0 1	0	1
1 0	1	0
1 1	0	1

... tesite sami ...

Avtomat, ki periodično ponavlja pet stanj

$P = \circlearrowright 1, 1, 0, 0, 1$

Ideja: Števec, ki steje od 0 do 4 in vsakemu številu pripiše eno stanje.

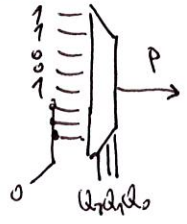


Ker moramo šteti vsaj do 4 (skozi pet vrednosti), bomo potrebovali 3-bite.

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	
0	0	0	0	0	1	} od 0 do 4
0	0	1	0	1	0	
0	1	0	0	1	1	
0	1	1	1	0	0	
1	0	0	0	0	0	
1	0	1	0	0	0	} tega ne potrebujemo
1	1	0	0	0	0	
1	1	1	0	0	0	

- $Q_2^+ = \bar{Q}_2 Q_1 Q_0$ ~~$+ Q_2 \bar{Q}_1 \bar{Q}_0$~~
- $Q_1^+ = \bar{Q}_2 \bar{Q}_1 Q_0 + \bar{Q}_2 Q_1 \bar{Q}_0$
  $= \bar{Q}_2 (\bar{Q}_1 Q_0 + Q_1 \bar{Q}_0)$
  $= \bar{Q}_2 (Q_1 \oplus Q_0)$
- $Q_0^+ = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_2 Q_1 \bar{Q}_0$
  $= \bar{Q}_2 \bar{Q}_0 (\bar{Q}_1 + Q_1) =$
  $= \bar{Q}_2 \bar{Q}_0 = \overline{Q_2 + Q_1}$

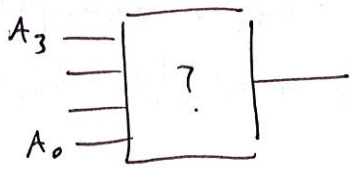
Za prehode uporabimo kar multiplekser



Lahko bi ga sestavili tudi klasično. Kakor želimo...

Preveri, če je 4-bitno število deljivo s 3

6.4.2022



	A_3	A_2	A_1	A_0	
0	0	0	0	0	1
	0	0	0	1	0
	0	0	1	0	0
	0	0	1	1	1
	0	1	0	0	0
	0	1	0	1	0
	0	1	1	0	1
	0	1	1	1	0
	1	0	0	0	0
	1	0	0	1	1
	1	0	1	0	0
	1	0	1	1	0
	1	1	0	0	1
	1	1	0	1	0
	1	1	1	0	0
15	1	1	1	1	1

→ Nič je seveda deljivo s tri

A_3A_2 \ A_1A_0	00	01	11	10
00	1	0	1	0
01	0	0	0	1
11	1	0	1	0
10	0	1	0	0

Vidimo, da nam Karnaugh ne pomaga.

$\bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0 + \bar{A}_3\bar{A}_2A_1A_0 + \bar{A}_3A_2\bar{A}_1\bar{A}_0 +$

$+ A_3\bar{A}_2\bar{A}_1A_0 + \bar{A}_3\bar{A}_2A_1A_0 + A_3A_2A_1A_0 =$

= poenostavljamo do osnovnih ...

Tu mnes lahko, da sem na kakšno nalogo pozabil. Boste že, saj smo vajetno samo ponavljali.

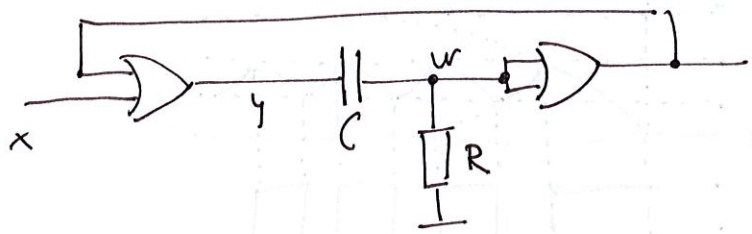
~~2.4.22~~ ~~4.4.2022~~

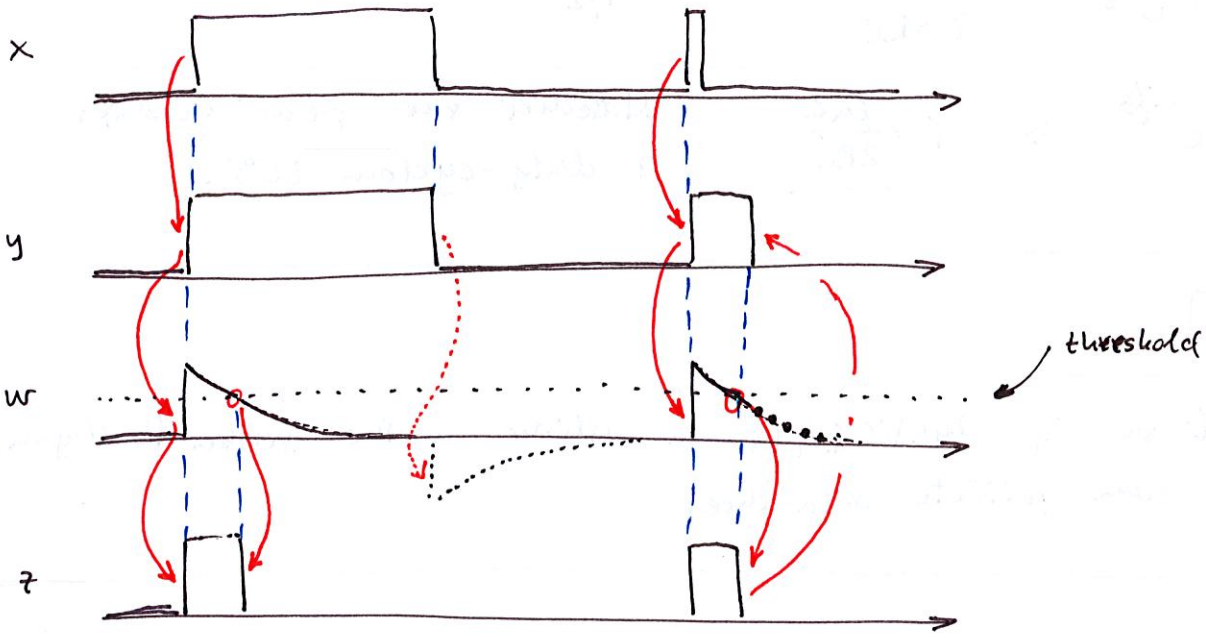
PRAZEN PROSTOR

UNIVIBRATOR

13.4.2022

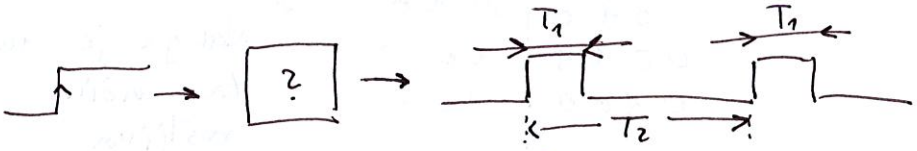
Kaj pocne naslednje vezje?

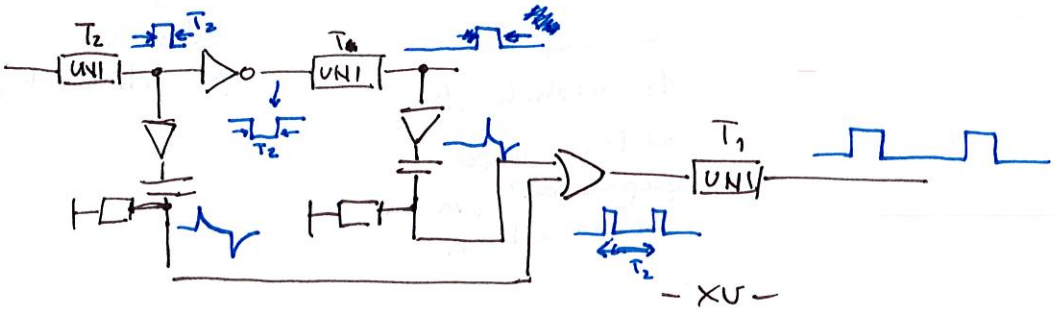




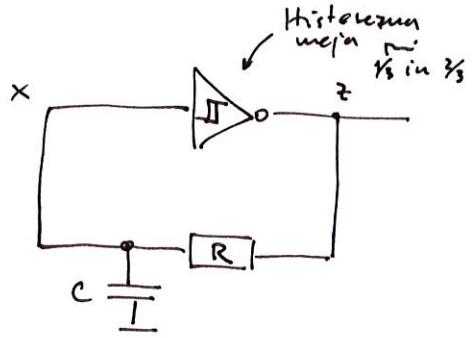
Univibrator nam na vhodni pulz poljubne dolžine t vedno enako dolgim in visokim pulzom. Dolžina izhodnega pulza je povezana oz. sorazmerna časovni konstanti $\tau = RC$. Če namesto upora uporabimo potenciometer, lahko spreminjamo izhodno dolžino. Uporabnih vrednosti takšnega stvojilca je veliko.

Vezje, ki na izhodni pulz odgovari z dvema enakima.



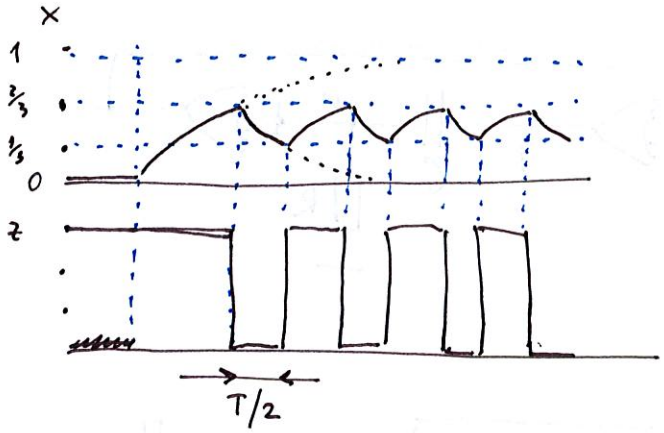


URA



Začnemo s praznim kondenzatorjem

$\Rightarrow x = 0$



Dolžina

$$\frac{U}{3} = \frac{2}{3} U e^{-\frac{T/2}{\tau}}$$

$\tau = RC$

$$\frac{1}{2} = e^{-\frac{T/2}{\tau}} \Rightarrow T = \frac{\ln 2}{2RC}$$

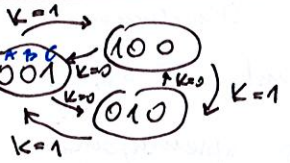
Navedli smo pulzni generator z duty-cyclom 50%.

20.4.2022

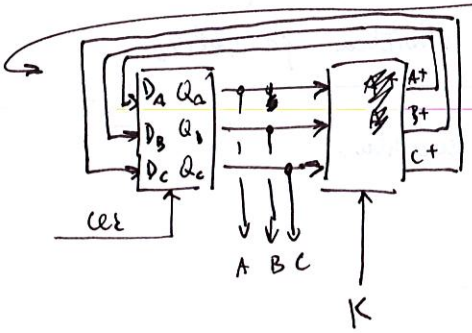
Pogledali smo čip 74LVC1G98 in različne načine uporabe. Predlagam, da si sami poiščete data-sheet.

4.5.2022

Tribitni sinhroni avtomat, ki izvaja naslednje zaporedje stanj: (s kontrolnim bitom K)



Klasično lahko problem bomo rešili tabelo, (register + prehodna + krmilni vhod)



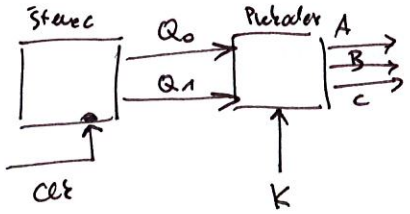
A	B	C	K	A^+	B^+	C^+
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	1	0	0
⋮						

16 možnih stanj, od tega veliko prepovedanih, za kar je treba poskrbeti

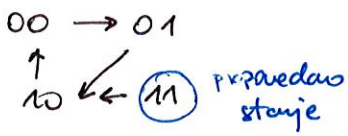
$\Rightarrow$ vesolo na delo

Naloga je na ta način rešljiva, lahko se pa lotimo drugače

Alternativa 2-bitni sinhroni števec, saj imamo le tri izhodna stanja, preko katerih krožimo



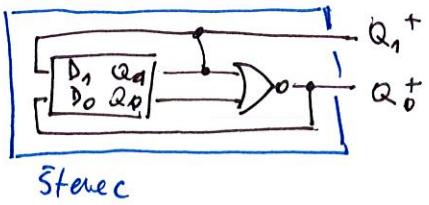
Števec vaj šteje tako



Q_1 Q_0	Q_1^+ Q_0^+
0 0	0 1
0 1	1 0
1 0	0 0
1 1	1 0

$Q_0^+ = \overline{Q_1 + Q_0}$

$Q_1^+ = Q_0$

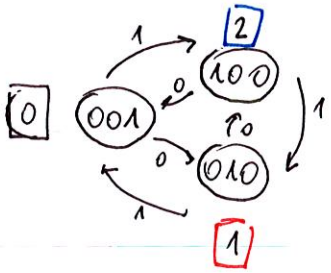


Zastranitev

Hitro nas zagrabi, da bi števec sestavili iz treh FF vezanih tako

in bi s K obrnili smer štetja. A potor, hic sunt dracones! Zahtevano sinhroni avtomat, ta pa ne bi bil.

Tu naredimo konceptualni premislek.



$Q_1 Q_0$ K	A B C
0 0 0	0 1 0
0 0 1	1 0 0
0 1 0	1 0 0
0 1 1	0 0 1
1 0 0	0 0 1
1 0 1	0 1 0

$C = \overline{Q_1} Q_0 K + Q_1 \overline{Q_0}\,\overline{K}$

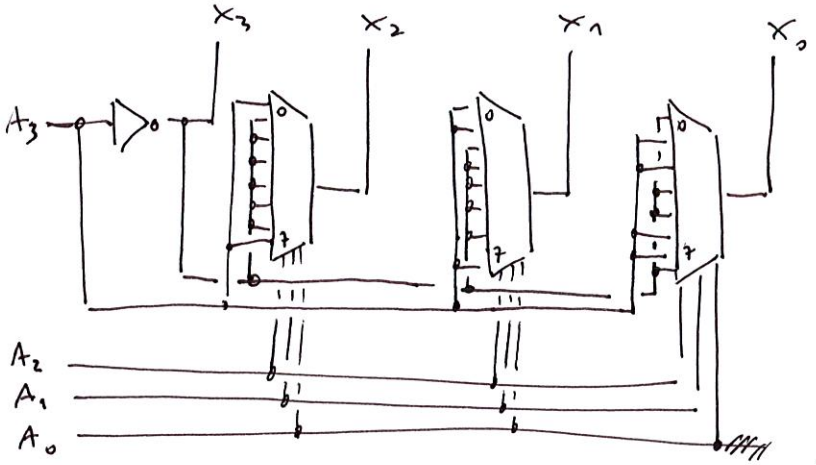
$B = \overline{Q_0}\left(\overline{Q_1} K + Q_1 K_1\right)$

$A = \overline{Q_1}\left(\overline{Q_0} K + Q_0 \overline{K}\right)$

$= \overline{Q_1}\left(Q_0 \oplus K\right)$

Naslednje vaje:

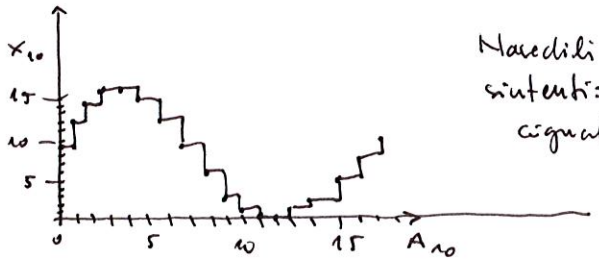
11.5.2022



Na vhod pripeljemo izhod 4-bitnega sinhronega števca, ki šteje kot

0, 1, 2, ..., 13, 14, 15, 0, 1, 2, ...

$A_3 A_2 A_1 A_0$	$X_3 X_2 X_1 X_0$	
0 0 0 0	1 0 0 1	9
0 0 0 1	1 1 0 0	12
0 0 1 0	1 1 1 0	14
0 0 1 1	1 1 1 1	15
0 1 0 0	1 1 1 1	15
0 1 0 1	1 1 1 0	14
0 1 1 0	1 1 0 0	12
0 1 1 1	1 0 0 1	9
1 0 0 0	0 1 1 0	6
1 0 0 1	0 0 1 1	3
1 0 1 0	0 0 0 1	1
1 0 1 1	0 0 0 0	0
1 1 0 0	0 0 0 1	1
1 1 0 1	0 0 1 1	3
1 1 1 0	0 1 1 0	6
15 1 1 1 1		



Naredili smo sintentizator sinusnega signala.