# 4. Načrtovanje logičnega in sekvenčnega vodenja

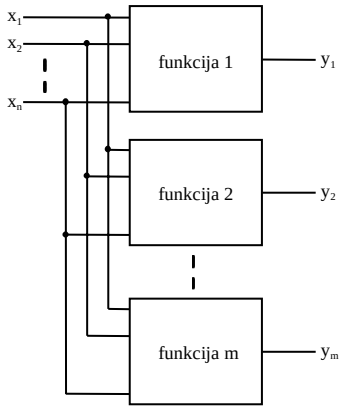
- Dve vrsti logičnih krmilij:
  - **Kombinacijska krmilja**
    - stanje vhodnih signalov se neposredno preslika v stanje izhodnih signalov
    - takšno krmilje ne vsebuje pomnilnih elementov
  - **Sekvenčna krmilja**
    - stanje izhodov ni odvisno le od stanja vhodnih signalov, temveč tudi od stanja notranjih pomnilnih elementov
    - ista kombinacija vhodnih stanj se lahko preslika v različne izhodne kombinacije

## 4.1 Kombinacijska krmilja

- Realizirana z osnovnimi logičnimi funkcijami
- Preklopne funkcije
  - vhodi so preklopne ali logične spremenljivke, ki lahko zavzamejo dve vrednosti, 0 ali 1
  - tudi rezultat lahko zavzame le ti dve vrednosti
- Interpretacija logičnih spremenljivk
  - delovni kontakt releja: x = 0 - kontakt je razklenjen (ne prevaja), x = 1 - kontakt je sklenjen (prevaja)
  - delovni kontakt tipke: x = 0 - tipka ni pritisnjena, x = 1 - tipka je pritisnjena
  - mirovni kontakt tipke: x = 0 - tipka je pritisnjena, x = 1 - tipka ni pritisnjena

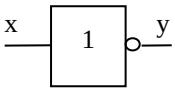
## Zgradba kombinacijskega krmilja

- Takšno krmilje sestavlja več preklopnih funkcij
- Če znamo preklopno funkcijo izraziti z osnovnimi logičnimi operacijami, znamo funkcijo tudi realizirati
- Pravilnostna tabela
  - vrednosti preklopne funkcije pri vseh možnih kombinacijah vrednosti vhodnih logičnih spremenljivk



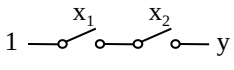

## Osnovne logične operacije

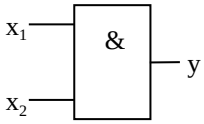
- Negacija




| x | y |
|---|---|
| 0 | 1 |
| 1 | 0 |

- Logični in (AND)



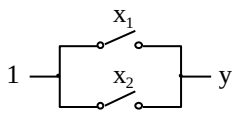

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

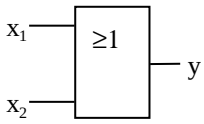
'maskiranje'

| | |
|---|---|
| vhodni register | 11011000 |
| maska | 01101101 |
| izhod | 01001000 |

## Osnovne logične operacije /2

- Logični ali (OR)



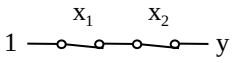

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

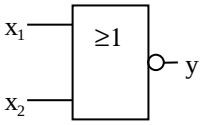
'maskiranje'

| | |
|---|---|
| vhodni register | 11011000 |
| maska | 01101101 |
| izhod | 11111101 |

## Osnovne logične operacije /3

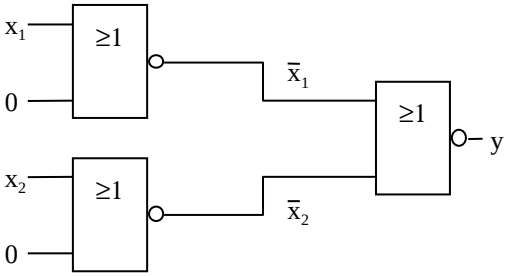
- Negiran ali (NOR)




| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

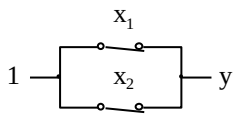
Realizacija AND z NOR

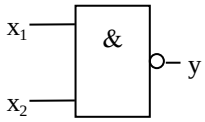



| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## Osnovne logične operacije */4*

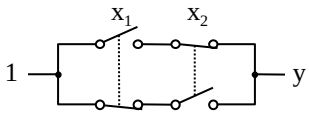
- Negiran in (NAND)

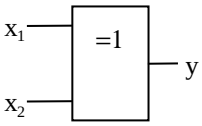



| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- Ekskluzivni ali (XOR)




| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## Boolova algebra (preklopna algebra)

- Dve vrednosti spremenljivk: 0, 1
- Operacije
  - disjunkcija (ali, OR): +, $\vee$;
    $x + y$, $x \vee y$
  - konjunkcija (in, AND): $\cdot$, $\wedge$, &;
    $x \cdot y$, $x \wedge y$, $x \,\&\, y$
  - negacija (ne, NOT): $\overline{\ }$, $\neg$;
    $\overline{x}$, $\neg x$
- Zapis preklopnih funkcij z Boolovo algebro
  $f_1=\overline{x}\cdot z+\overline{y}\cdot z+\overline{x}\cdot y$, $f_2=\overline{x}\cdot\overline{z}+\overline{y}\cdot\overline{z}+x\cdot y\cdot z$

## Pravila Boolove algebre

- Aksiomi (postulati)

  p1. $x+0 = x$

  p1' $x \cdot 1 = x$

  p2. $x+y = y+x$

  p2' $x \cdot y = y \cdot x$

  p3. $x+(y \cdot z) = x+y \cdot z = (x+y) \cdot (x+z)$

  p3' $x \cdot (y+z) = x \cdot y+x \cdot z$

  p4. $x+\overline{x} = 1$

  p4' $x \cdot \overline{x} = 0$

## Pravila Boolove algebre /2

- Teoremi

  $x+1 = 1$

  $x+1 = (x+1) \cdot 1 = (x+1) \cdot (x+\overline{x}) =$

  $= x+(1 \cdot \overline{x}) = x+\overline{x} = 1$

  $x+x = x$ $\qquad$ $x \cdot x = x$

  $\overline{\overline{x}} = x$ $\qquad$ $x \cdot 0 = 0$

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

  $\overline{x \cdot y} = \overline{x}+\overline{y}$ $\qquad$ (de Morganovi pravili)

## Kanonične oblike Boolovih funkcij

- Minterm n spremenljivk:

  Boolov produkt teh n spremenljivk, kjer se vsaka spremenljivka lahko pojavi v resnični ali negirani obliki
  npr. A, B; mintermi: AB, $\overline{A}$B, A$\overline{B}$, $\overline{A}\overline{B}$
- Maksterm n spremenljivk:

  Boolova vsota teh n spremenljivk, kjer se vsaka spremenljivka lahko pojavi v resnični ali negirani obliki
  npr. A, B; makstermi: A+B, $\overline{A}$+B, A+$\overline{B}$, $\overline{A}$+$\overline{B}$
- Normalna ali kanonična oblika:

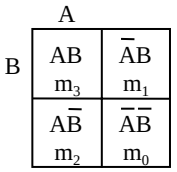
  vsota mintermov $m_i$ ali produkt makstermov $M_i$

## Predstavitev z Veitchevim diagramom

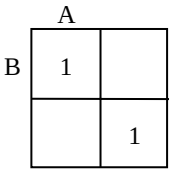
- Veitchev diagram
  - mreža kvadratov, ki predstavljajo minterme
  - razporejeni so tako, da se sosednja minterma razlikujeta le v eni spremenljivki
  - na rob diagrama napišemo spremenljivke, ki se v vseh mintermih ustreznega stolpca ali vrstice pojavijo v svoji resnični obliki
- Predstavitev Boolove funkcije
  - funkcijo zapišemo v kanonični obliki
  - v kvadratke diagrama, ki predstavljajo minterme funkcije, vpišemo enice

## Veitchevi diagrami

- Za dve spremenljivki



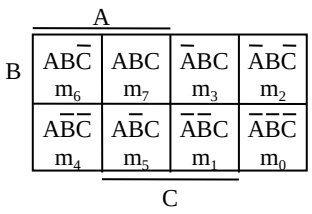

npr.



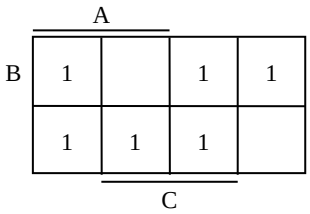

$f = AB+\bar{A}\bar{B}$

- Za tri spremenljivke

$f = AB\bar{C}+A\bar{B}C+A\bar{B}\bar{C}+\bar{A}BC+\bar{A}B\bar{C}+\bar{A}\bar{B}C$

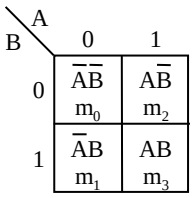


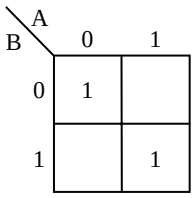

npr.



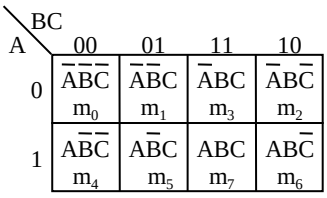

## Karnaugh-jev diagram

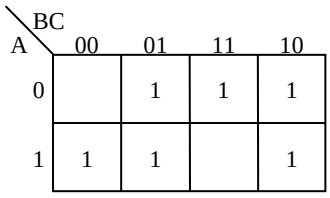
- K-diagram za dve spremenljivki




- K-diagram za tri spremenljivke



## Poenostavljanje

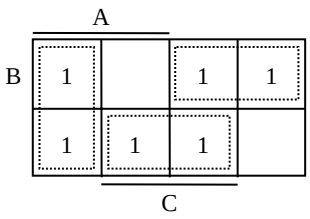
- Enolična pravilnostna tabela - različni algebrajski zapisi
- Zapise lahko poenostavljamo
  - s pravili Boolove algebre
    - zahteva nekaj spretnosti
    - ne vemo, če je dobljeni zapis minimalen
  - s sistematično minimizacijo
    - grafične metode (do 5 vhodnih spremenljivk)
    - tabelarične metode - poljubno število vh. spremenljivk
      najbolj znana je metoda Quine McCluskey
      zanjo obstajajo računalniški algoritmi

## Poenostavljanje z Veitchevimi diagrami

- Osnova - sosednost kvadratov
  - sosednja kvadrata se razlikujeta le v eni spremenljivki
  - kot sosednje štejemo tudi kvadrate na skrajnih robovih neke vrstice ali stolpca
  - če je v dveh sosednjih kvadratih enica, ju lahko združimo
  - združitev dveh sosednjih kvadratov eliminira spremenljivko, ki je pri enem negirana, pri drugem pa ne:

$$x \cdot y + \overline{x} \cdot y = (x + \overline{x}) \cdot y = 1 \cdot y = y$$

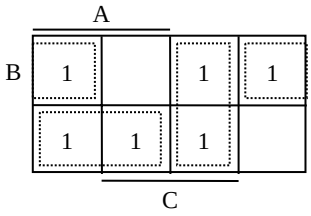
# Primeri poenostavljanja




$f = AB\bar{C}+A\bar{B}C+A\bar{B}\bar{C}+\bar{A}BC+\bar{A}B\bar{C}+\bar{A}\bar{B}C$

$f = A\bar{C}+\bar{B}C+\bar{A}B$

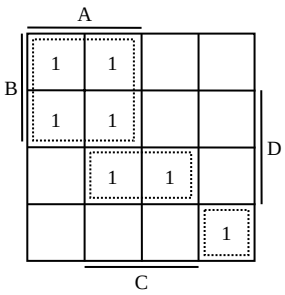
Ali na drug način:




$f = B\bar{C}+A\bar{B}+\bar{A}C$

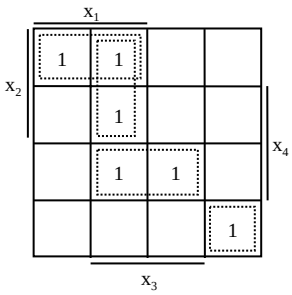
# Primeri poenostavljanja /2

a)



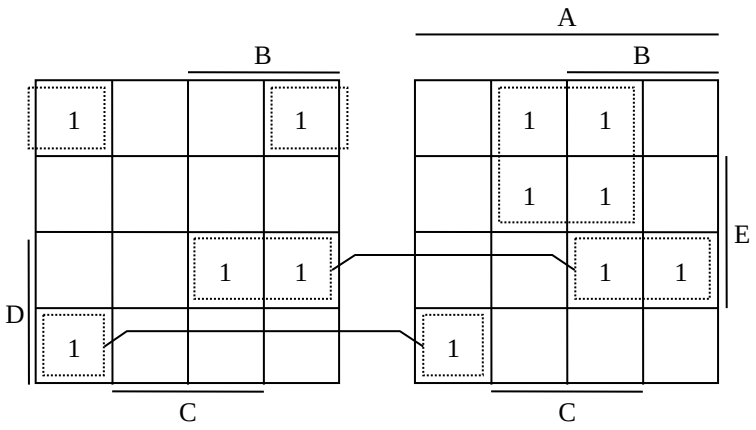

$f = AB+\bar{B}CD+\bar{A}\bar{B}C\bar{D}$

b)




$f = x_1x_2\bar{x}_4+x_1x_2x_3+\bar{x}_2x_3x_4+\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$

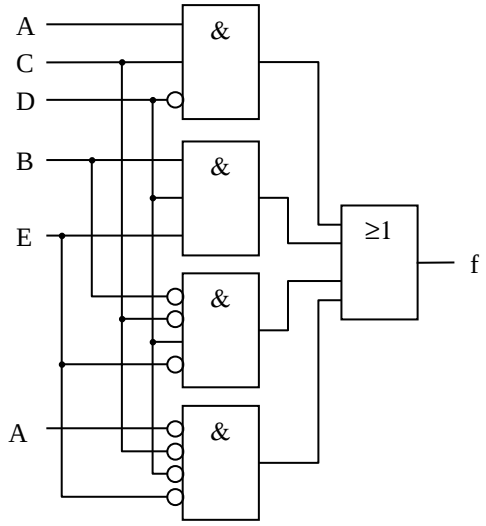
## Primeri poenostavljanja /3



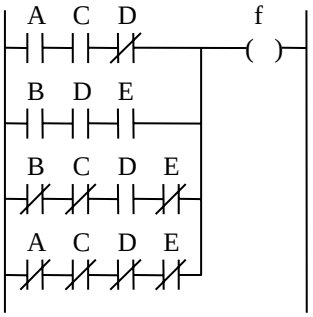

$f = AC\bar{D}+BDE+\bar{B}\bar{C}D\bar{E}+\bar{A}CD\bar{E}$

## Realizacija preklopne funkcije v PLK

FBD:

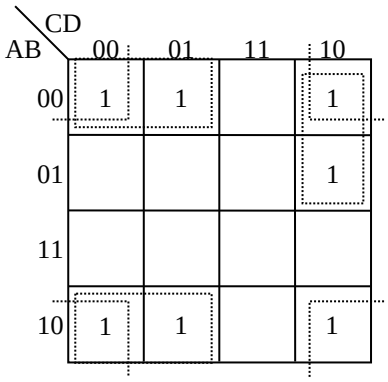



LD:



## Poenostavljanje s Karnaughovimi diagrami

- Postopek je enak kot pri Veitchevih diagramih
  - iščemo sosednje kvadrate
- Primer:

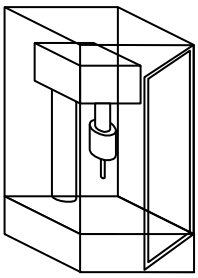



$$f = \overline{A}\overline{B}\overline{C} + \overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\overline{C}$$

$$f = \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + A\overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D$$

$$f = \overline{B}\overline{C} + \overline{B}\overline{D} + \overline{A}C\overline{D}$$

## Primer - vrtalni stroj



Besedni opis

Če je deluje stroj v avtomatskem načinu, se motor vrtalnika vrti le tedaj, ko so vrata zaprta. Če deluje stroj v ročnem načinu, se motor vrti tedaj, ko so vrata zaprta, ali pa če so vrata odprta in je vrtalnik v zgornji legi.

Spremenljivke

$x_1$ - ročni/avtomatski način (0 - avtomatsko, 1 - ročno)
$x_2$ - vrata zaprta
$x_3$ - vrtalnik v zgornji legi
$y$ - pogon motorja

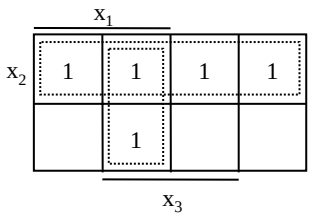
## Primer /2

Direktna rešitev: $y = \bar{x}_1 x_2 + x_1(x_2 + \bar{x}_2 x_3)$

Pravilnostna tabela

| $x_1$ | $x_2$ | $x_3$ | y |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

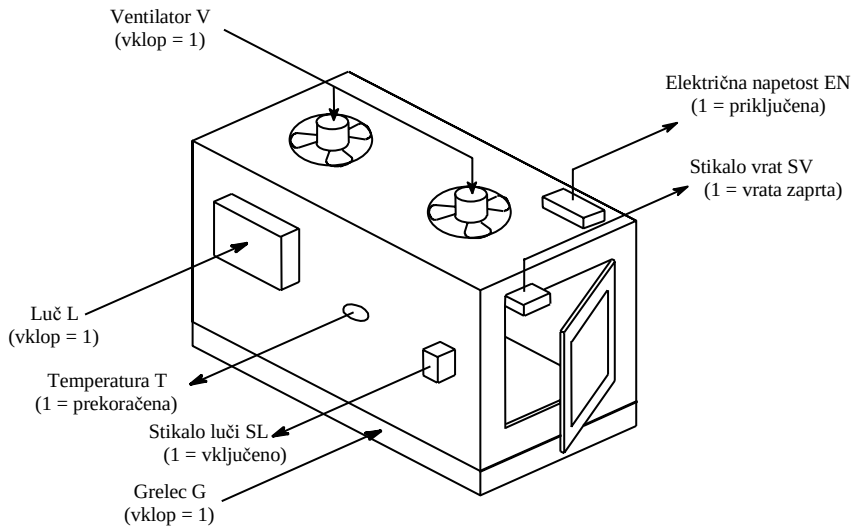
Poenostavljanje




Poenostavljena rešitev:

$$y = x_2 + x_1 x_3$$

## Primer - laboratorijska peč



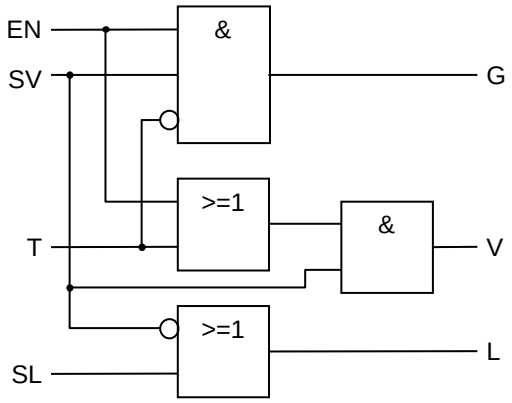
## Primer /2

- Besedni opis
  - Grelec (G) je vključen, ko je aktivirano močnostno stikalo (EN) in so vrata zaprta (SV) ter je temperatura (T) pod mejo.
  - Ventilator (V) je vključen, ko je vključen grelec (G), ali ko je temperatura (T) nad mejo in so vrata (SV) zaprta.
  - Luč (L) je prižgana, ko je vključeno stikalo luči (SL), ali če so vrata odprta.

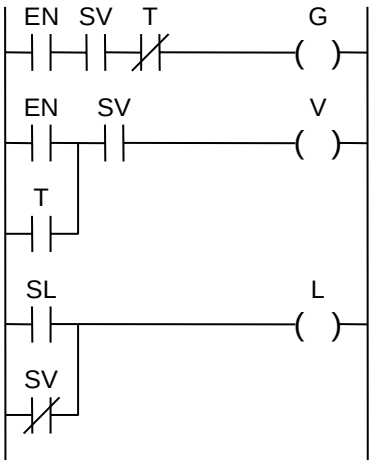
## Primer /3

Izvedba v PLK

FBD:




LD:



### Načrtovanje kombinacijskih krmilij

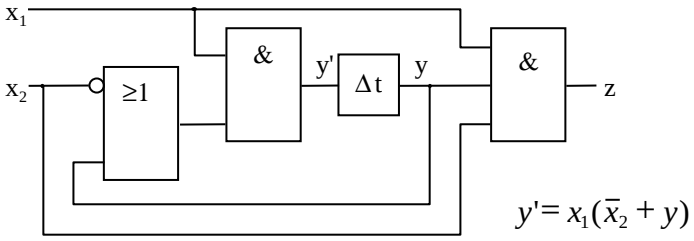
- Besedni opis
- Za vsak izhod krmilja
  - ugotovimo, kateri vhodi nanj vplivajo
  - zapišemo pravilnostno tabelo
    - če v tabeli pride do protislovij, je potrebna realizacija s sekvenčnim krmiljem
  - zapišemo preklopno funkcijo
    - poenostavimo zapis, če je potrebno
  - realiziramo preklopno funkcijo v PLK

## 4.2 Sekvenčna krmilja

- Izhodi so odvisni od vhodov in notranjih stanj
- Notranji pomnilnik z lastnimi stanji
  - krmilja ne moremo opisati z vhodno/izhodno pravilnostno tabelo
  - v pravilnostni tabeli nastopijo protislovja - ista kombinacija vhodov da različne izhode
- Dve skupini sekvenčnih krmilij
  - prosto delujoča krmilja - na vhodu se lahko pojavi poljubna kombinacija v poljubnem zaporedju
  - koračno delujoča krmilja - kombinacije vhodov se vedno pojavljajo v določenem zaporedju

## Primer sekvenčnega krmilja

- Izhod krmilja mora imeti vrednost 1, če sta oba vhoda enaka 1, vendar le, če se je najprej postavil na 1 prvi vhod in šele nato drugi




$$y' = x_1(\bar{x}_2 + y)$$
$$z = x_1 x_2 y$$

## Zgradba sekvenčnega krmilja

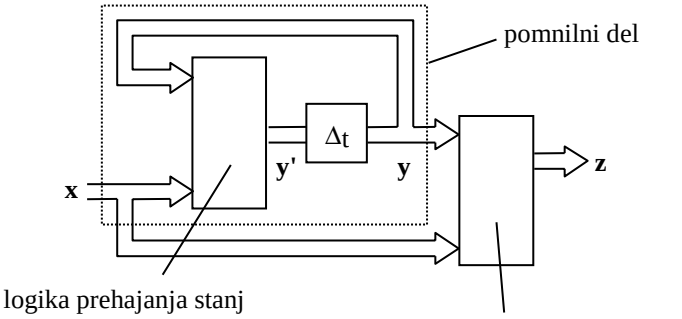
- Vezje, ki predstavlja realizacijo krmilja, lahko razdelimo v dva dela:
  - pomnilni del
  - izhodna kombinacijska logika
- Pripadajoči enačbi:
  - enačba prehajanja stanj

$$y_\lambda' = f_\lambda(x_1, x_2, ..., x_n; y_1, y_2, ..., y_l); \quad \lambda = 1,2,\ldots,l$$

  - izhodna enačba

$$z_\mu = g_\mu(x_1, x_2, ..., x_n; y_1, y_2, ..., y_l); \quad \mu = 1,2,\ldots,m$$

## Zgradba sekvenčnega krmilja /2




$$y_\lambda' = f_\lambda(x_1, x_2, ..., x_n; y_1, y_2, ..., y_l); \quad \lambda = 1,2,\ldots,l$$

$$z_\mu = g_\mu(x_1, x_2, ..., x_n; y_1, y_2, ..., y_l); \quad \mu = 1,2,\ldots,m$$

# 4.2.1 Koračna krmilja

- Večina procesov v industriji ima izrazito sekvenčno naravo
  - postopki z natanko določenim zaporedjem operacij oz. korakov
- Med izvajanjem operacije
  - se lahko spremenijo le tisti vhodi, ki so s to operacijo povezani
  - ostali vhodi nas medtem ne zanimajo

## Koračna krmilja /2

- Znatno manjše število stanj in prehodov med njimi
  - ko se izvaja določen korak, upoštevamo le del vhodov in izhodov, ki so s tem korakom povezani
- Postopek načrtovanja
  - v primerjavi s prosto delujočimi krmilji enostavnejši
  - dva osnovna pristopa
    - “dogodkovni” pristop
    - pristop s stanji

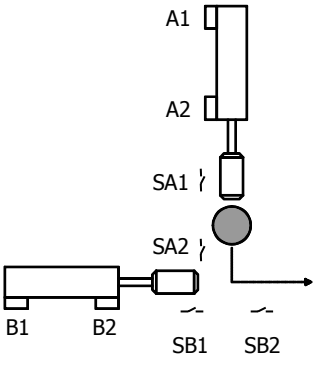
## Načrtovanje koračnih krmilij

- “Dogodkovni” pristop
  - stanje izhodov spreminjamo ob nastopu dogodkov
    - v splošnem moramo detektirati **prehode na vhodih** namesto stanja vhodov; le izjemoma zadošča samo stanje
    - izhode postavljamo s SET/RESET
- Pristop s stanji
  - notranja stanja v programu spreminjamo glede na kombinacijo **trenutnih stanj in vhodov**; izhodi so odvisni od notranjih stanj
    - pogoji za spremembo stanja morajo vedno vključevati trenutno stanje; stanja postavljamo zadržano (SET/RESET)
    - izhodi so odvisni od kombinacije notranjih stanj in jih ne postavljamo zadržano (v LD navadna tuljava namesto S/R)

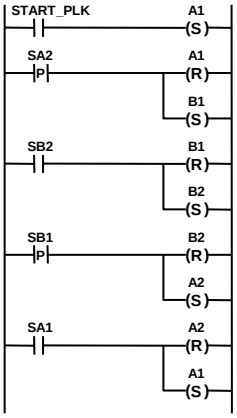
## Načrtovanje koračnih krmilij /2

- Dogodkovni pristop
  - \+ manjši in enostavnejši program krmilnika
  - – težja vrnitev v začetno stanje (npr. po zaustavitvi)
  - – težje odkrivanje napak
    - postavljanje in brisanje stanj je razpršeno po programu
    - prehajanje stanj se prepleta s postavljanjem izhodov
- Pristop s stanji
  - \+ manjša občutljivost na šumne signale in motnje
    - prehod v novo stanje se izvede le ob določenih pogojih; na ostale spremembe krmilje ne odgovarja
  - \+ enostavnejše odkrivanje napak, lažje popravljanje
  - – obsežnejši program

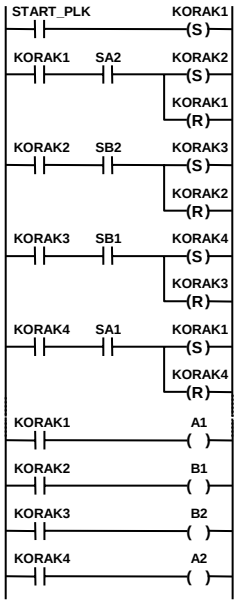
## Primer:
### potiskanje obdelovanca z dvema pnevmatskima cilindroma




Dogodkovni pristop




Pristop s stanji

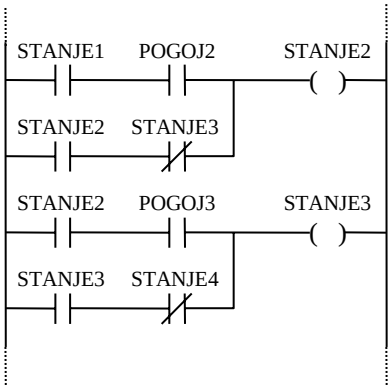


## Izvedba koračnih krmilij z lestvičnimi diagrami - pristop s stanji

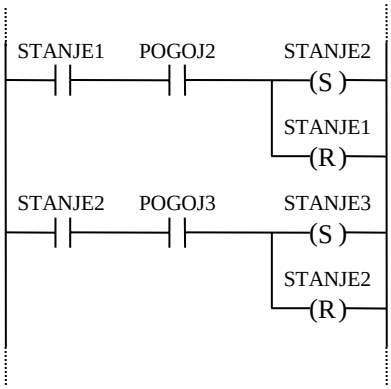
- Narišemo diagram prehajanja stanj
- Vsakemu stanju v diagramu priredimo notranje stanje – logično spremenljivko oz. marker
- Za vsako notranje stanje poiščemo pogoje za vstop in izstop iz stanja
- V pogoje za vstop vključimo aktivnost predhodnega stanja in v pogoje za izstop aktivnost naslednjega stanja
- Z notranjimi stanji sestavimo pogoje za aktivnost izhodnih signalov

## Realizacija prehajanja stanj

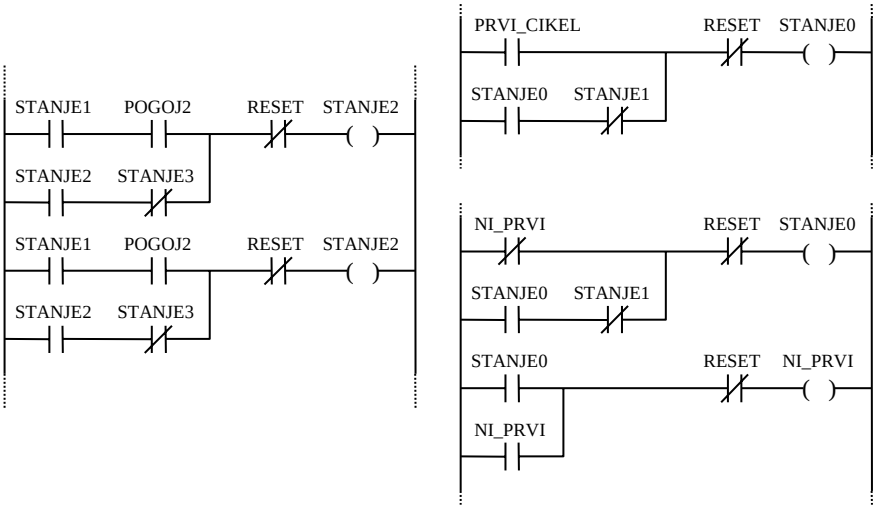
1. način




2. način



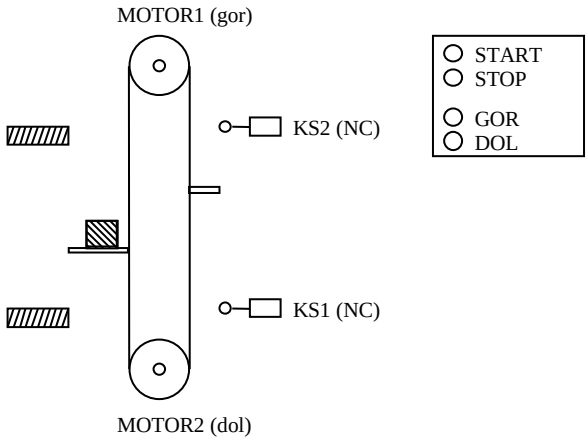
## Resetiranje sekvence, inicializacija



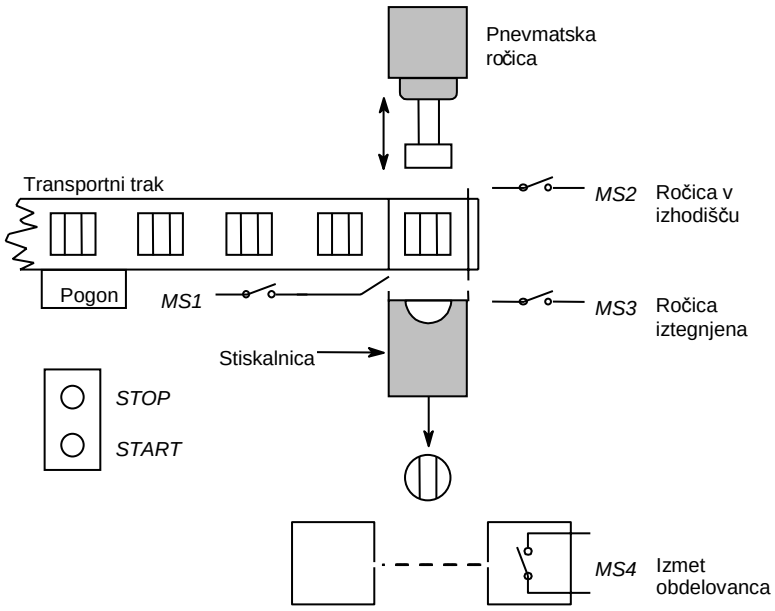

## Priporočena struktura lestvičnega diagrama

1. Načini delovanja in osnovne funkcije
   - postavitev v začetno stanje (inicializacija)
   - pogoji za omogočanje in reset programa
   - preklop ročno/avtomatsko
2. Osnovno zaporedje operacij
   - prehajanje stanj
3. Postavljanje izhodov (aktuatorjev)
   - izhodi so aktivni glede na stanja (točka 2.) in način delovanja (točka 1.)
4. Signalizacija

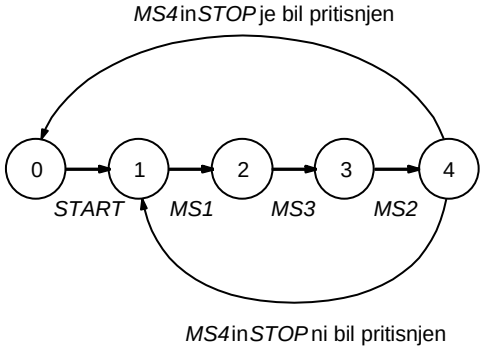
## Primer: tovorno dvigalo




## Primer: stroj za preoblikovanje



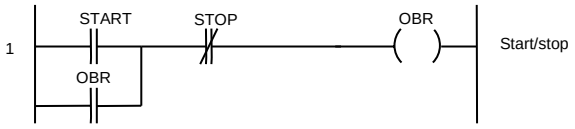
## Diagram prehajanja stanj in prireditvena tabela




| Signal/spremenlj. | Naslov PLK |
|---|---|
| *START* | %I0 |
| *MS1* | %I1 |
| *MS2* | %I2 |
| *MS3* | %I3 |
| *MS4* | %I4 |
| *STOP* | %I5 |
| *ST_0 (Osn. st.)* | %M100 |
| *ST_1 (Stanje 1)* | %M101 |
| *ST_2 (Stanje 2)* | %M102 |
| *ST_3 (Stanje 3)* | %M103 |
| *ST_4 (Stanje 4)* | %M104 |
| *Transportni trak* | %Q0 |
| *Ročica potiska* | %Q1 |
| *Ročica se vrača* | %Q2 |
| *Stiskalnica* | %Q3 |
| *OBR (Obratovanje)* | %M1 |

## Lestvični diagram

a) Vklop obratovanja

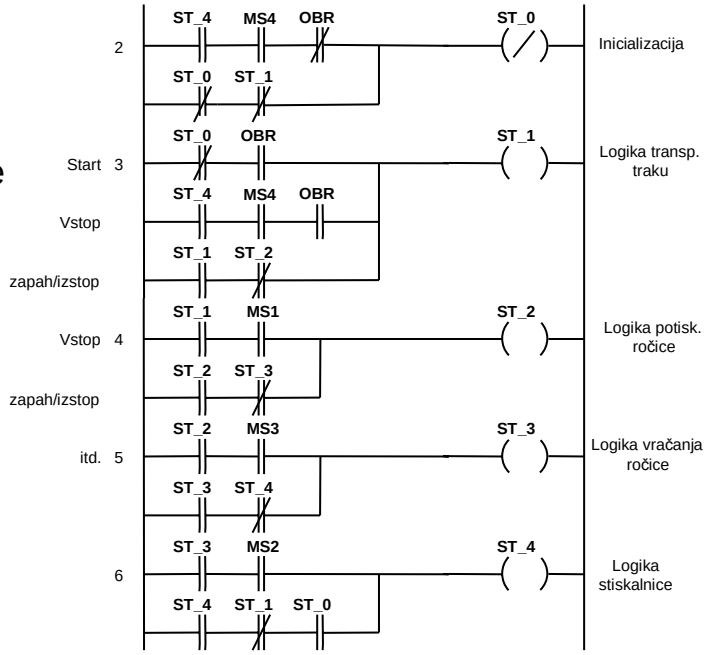



$$OBR = START \cdot \overline{STOP} + OBR \cdot \overline{STOP} = (START + OBR) \cdot \overline{STOP}$$

STANJE VSTOP ZAPAH/IZSTOP

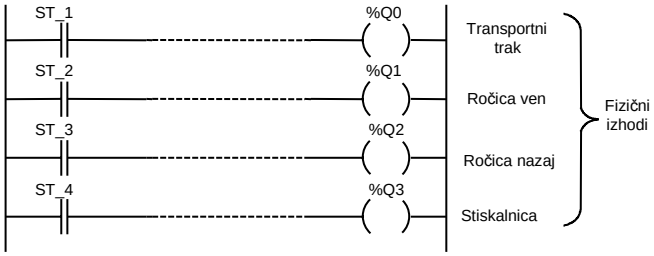
# Lestvični diagram

b) Prehajanje stanj




# Lestvični diagram

c) Postavljanje izhodov




d) Signalizacija

Krmiljenje prikazov, zaščita, alarmiranje

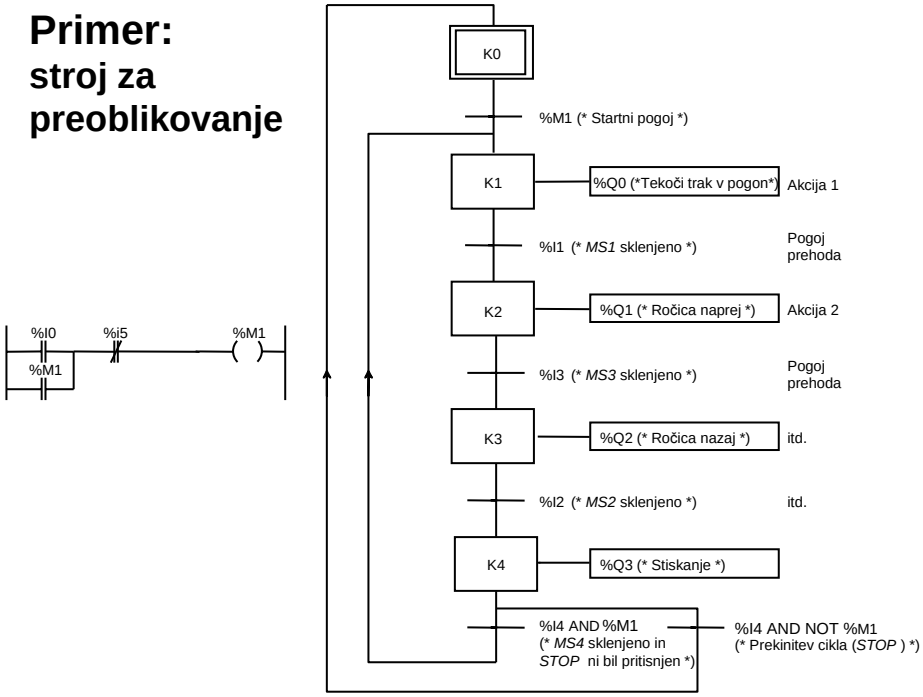
## Izvedba koračnih krmilij s sekvenčnim funkcijskim diagramom (SFC)

- Zaporedje prehajanja stanj določa sama struktura SFC
- Akcije, prehodni pogoji
  - programiramo v ostalih jezikih, npr. LD ali FBD
- Signalizacija, branje tipk ipd.:
  - program v LD ali FBD, ki se izvaja vzporedno
  - oba programa (SFC in LD) povežemo z istim opravilom (TASK)
  - podatke izmenjujeta preko globalnih spremenljivk

## Izvedba koračnih krmilij s SFC

- Kompleksnost realnih problemov
  - en sam SFC bi bil preobsežen
- Običajno več diagramov za različna opravila
- Vklapljanje opravil lahko izvaja spet sekvenčni funkcijski diagram
  - koraki tega SFC predstavljajo glavna stanja sistema
  - akcije v tem diagramu predstavljajo pogoje za izvajanje posameznih opravil – podrejeni SFC
  - problematično resetiranje
  - težaven ponoven zagon po odpravi napake

**Primer:**
**stroj za preoblikovanje**




## 4.2.2 Prosto delujoča krmilja

- Na vhodu krmilja se lahko pojavljajo poljubne kombinacije vrednosti vhodnih signalov v poljubnem zaporedju
- Pri načrtovanju izhajamo iz tabele stanj
- Huffmanov postopek
  - prilagojen za relejska in elektronska krmilja
  - rezultat je logično vezje z minimalnim številom pomnilnih elementov
  - ne minimizira kompleksnosti kombinacijskega dela krmillja
- Modificirana različica Huffmanovega postopka
  - prilagojena za načrtovanje krmilij, ki so realizirana s PLK

## Zapis s Karnaughovim diagramom

- Ločen zapis enačbe stanj in izhodne enačbe
- Tabela stanj za primer $y' = x_1(\bar{x}_2 + y)$

| y \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

(y' =)

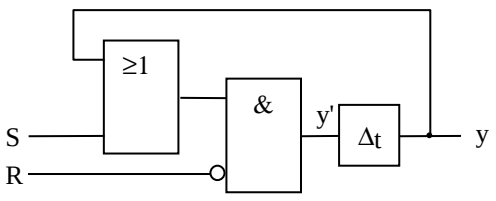
- Izhodna tabela za primer $z = x_1 x_2 y$

| y \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |

(z =)

## Primer pomnilne celice

1.)



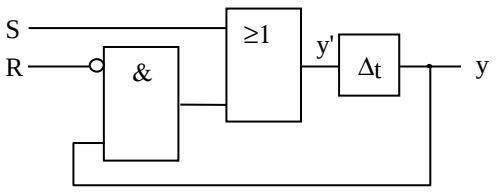

| y \ SR | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |

(y' =)

$y' = \bar{R}(S + y)$

$z = y$

2.)




| y \ SR | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |

(y' =)

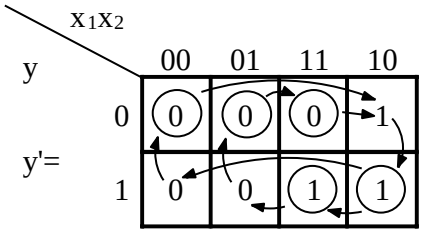
$y' = S + \bar{R}y\,;\; z = y$

## Stabilna in nestabilna stanja

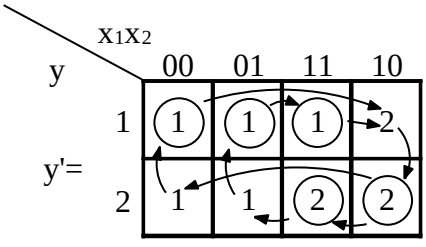
- y je zakasnjen glede na y', vendar že po kratkem času postane y enak y'
- Nestabilna stanja
  - polja v diagramu, kjer y ob robu ni enak vrednosti y' v polju, predstavljajo nestabilna stanja
  - aktivna le kratek čas
- Stabilna stanja
  - polja v diagramu, kjer je y ob robu enak vrednosti y' v polju, predstavljajo stabilna stanja
  - aktivna do spremembe stanja vhodov

## Označevanje stabilnih stanj

- S stanji pomnilnih elementov

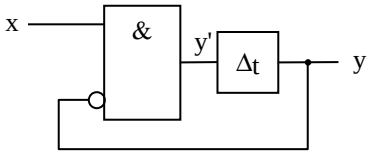


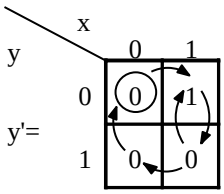

- S simboličnimi oznakami



## Neželeni pojavi

- Oscilacije pomnilnega elementa
  - če lahko sklenemo zanko s prehodi med nestabilnimi stanji
- Primer: oscilator



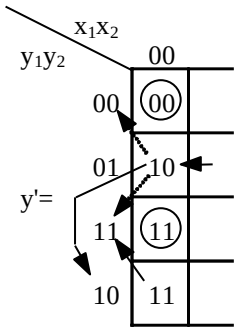




le eno stabilno stanje: (x;y)=(0;0)
čim x postane 1, se pričneta izmenjevati nestabilni stanji
(1;0) in (0;1)

## Neželeni pojavi /2

- Tekmovanje med prehodi
  - enaka sprememba vhodov lahko vodi v različna stanja
  - vzrok so razlike v hitrosti preklopa elementov
  - do tega pojava pride le, če je v krmilju več notranjih pomnilnih stanj
- Ugotavljanje iz tabele stanj
  - preverimo prehode iz nestabilnih v stabilna stanja
  - iščemo stolpce, kjer je pri samodejnem prehodu potrebna sprememba več kot ene spremenljivke
  - problem nastopi, če je v takšnem stolpcu več stabilnih stanj

## Primer tekmovanja




- Krmilje z dvema vhodoma in dvema notranjima stanjema
- Celotno stanje: ($x_1x_2$;$y_1y_2$)
- Pričnemo opazovati pri (00;01)
- Prehodi:
  (00;01)->(00;10)->(00;11)
  ali
  (00;01)-> (00;11)
  ali
  (00;01)->(00;00) !!!

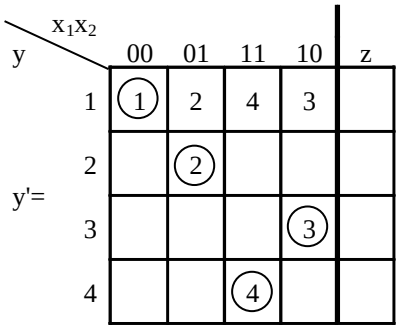
## Huffmanov postopek načrtovanja

- Vhodni podatek
  - besedni opis želenega delovanja krmilja
- Prvi korak je sestavljanje osnovne tabele prehajanja stanj
  - izberemo eno od poznanih stabilnih stanj (običajno je to začetno stanje)
  - v tabelo vpisujemo prehode iz tega stanja
  - za vsako novo stabilno stanje tabeli dodamo eno vrstico
  - ponovimo za vsa stabilna stanja
  - dodamo postavitev izhodov

## Primer načrtovanja

- Besedni opis
  - Krmilje ima dva vhoda in en izhod, ki se postavi na ena, če se najprej postavi na ena drugi vhod, nato pa še prvi (drugi ob tem ostane na 1). Možne so tudi sočasne spremembe obeh vhodov.
  - Krmilje torej 'prepoznava' sekvenco (00,01,11)
- Sestavljanje osnovne tabele prehajanja
  - pričnemo pri stanju vhodov 00, to stanje vnesemo v tabelo kot stanje 1
  - obravnavamo tri možne prehode:
    (00)->(01), (00)->(10) in (00)->(11)

## Primer načrtovanja /2




- Dobljena stanja 2, 3 in 4 so nestabilna, vsakemu pa priredimo enako oštevilčeno stabilno stanje v novi vrstici in istem stolpcu
- Nadaljujemo z drugo vrstico ter po potrebi dodajamo nova stanja
- Če v krmilju kateri od prehodov ni možen, je ustrezno polje v tabeli prazno

## Primer načrtovanja /3

- Končna osnovna tabela prehajanja
- V danem primeru so možni vsi prehodi, zato so izpolnjena vsa polja
- Takšna tabela vsebuje več informacije, kot je potrebno za samo delovanje krmilja
- Tabelo lahko skrčimo

| y | $x_1x_2$ = 00 | 01 | 11 | 10 | z |
|---|---|---|---|---|---|
| 1 | (1) | 2 | 4 | 3 | 0 |
| 2 | 1 | (2) | 5 | 3 | 0 |
| 3 | 1 | 6 | 4 | (3) | 0 |
| 4 | 1 | 6 | (4) | 3 | 0 |
| 5 | 1 | 6 | (5) | 3 | 1 |
| 6 | 1 | (6) | 4 | 3 | 0 |

y'=

## Poenostavljanje osnovne tabele prehajanja stanj

- Tabelo poenostavljamo z združevanjem vrstic
- Pravila:
  1. Dve ali več vrstic lahko združimo, če imata v istih stolpcih iste oznake stanj.
  2. Če je v eni vrstici neko stanje nestabilno, v drugi pa stabilno, potem tudi v združeno vrstico vpišemo stabilno stanje.
  3. Če v polju ni oznake stanja, polje je torej redundantno, lahko temu polju pripišemo poljubno stanje.

## Primer poenostavljanja

- Primer dveh vrstic

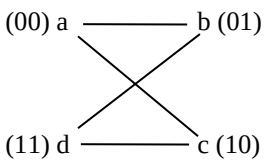
| | | | |
|---|---|---|---|
| (3) | - | 6 | 8 |
| 3 | (5) | 6 | - |
| (3) | (5) | 6 | 8 |

- Tabela iz obravnavanega primera
- a, b, c, d so reducirana stanja

| y \ $x_1x_2$ | 00 | 01 | 11 | 10 | prvotne vrstice |
|---|---|---|---|---|---|
| a | (1) | 2 | 4 | 3 | 1 |
| b | 1 | (2) | 5 | 3 | 2 |
| c | 1 | (6) | (4) | (3) | 3, 4, 6 |
| d | 1 | 6 | (5) | 3 | 5 |

## Kodiranje reduciranih stanj

- n stanj lahko kodiramo z log2(n) bitov
  - paziti moramo na neželene pojave (oscilacije, tekmovanja med prehodi)
  - kodiranje poskušamo izvesti tako, da se pri prehodih iz nestabilnih v stabilna stanja vedno spremeni le en bit
- Prehodni diagram:
  - rišemo le povezave, ki ustrezajo prehodom v stolpcih z več kot enim stabilnim stanjem



## Kodirana tabela prehajanja

| y \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| a 00 | (1) | 2 | 4 | 3 |
| b 01 | 1 | (2) | 5 | 3 |
| d 11 | 1 | 6 | (5) | 3 |
| c 10 | 1 | (6) | (4) | (3) |

$y_1'y_2'=$

| $y_1y_2$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | (00) | 01 | 10 | 10 |
| 01 | 00 | (01) | 11 | 10 |
| 11 | 00 | 10 | (11) | 10 |
| 10 | 00 | (10) | (10) | (10) |

## Enačbe prehajanja stanj

$y_1'=$

| $y_1y_2$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 1 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | 0 | 1 | 1 | 1 |

$y_2'=$

| $y_1y_2$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

$$y_1' = x_1 + x_2 y_1$$

$$y_2' = \bar{x}_1 x_2 \bar{y}_1 + x_1 x_2 y_2$$

## Izhodna funkcija

- Izhajamo iz osnovne tabele prehajanja
  - za vsako stabilno stanje je postavitev izhodov definirana
  - potrebno določiti še izhode v nestabilnih stanjih
  - skušamo dobiti čimbolj enostavno izhodno funkcijo, vendar brez neželenih pulzov pri prehodu nestabilnih stanj

z=

| $y_1y_2$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

$$z = x_1 x_2 y_1 y_2$$

## Izhodna funkcija /2

z=

| $y_1y_2$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

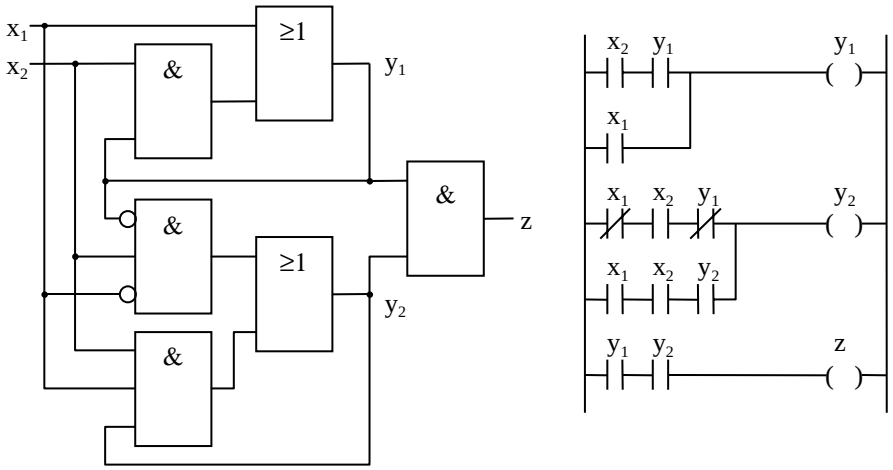
$$z = y_1 y_2 \quad (\checkmark)$$

z=

| $y_1y_2$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

$$z = x_1 y_2 \quad (\times)$$

## Realizacija v PLK




## Postopek

1. Iz besednega opisa sestavimo osnovno tabelo prehajanja.
2. Tabelo prehajanja poenostavimo z združevanjem vrstic.
3. Kodiramo stanja v poenostavljeni tabeli prehajanja. Pri tem se izogibamo prehodom, pri katerih bi se hkrati spremenil več kot en bit.
4. Iz kodirane prehodne tabele sestavimo K-diagrame in zapišemo enačbe prehajanja notranjih pomnilnih stanj.
5. Sestavimo izhodno tabelo in s pomočjo K-diagrama zapišemo izhodno funkcijo za vsak izhod krmilja. Paziti moramo na morebitne pulze na izhodih pri prehodu preko nestabilnih stanj.
6. Na podlagi zapisanih preklopnih funkcij narišemo preklopno vezje ali lestvični diagram.

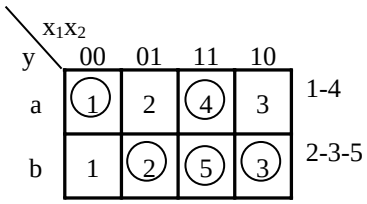
## Primer: dvoročni vklop

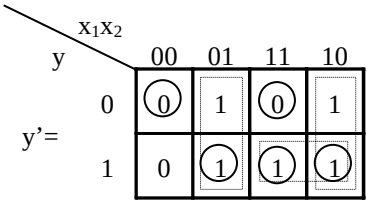
- Besedni opis:
  - Krmilje zahteva, da hkrati pritisnemo dve tipki (z vsako roko eno), preden se prične neka avtomatizirana akcija. S tem preprečimo, da bi npr. stiskalnica delavcu stisnila roko.
  - Krmilje mora odreagirati na vhodno sekvenco (00,11), izhod se torej postavi na 1 le, če se 'hkrati' vključita oba vhoda. Če se najprej vključi en vhod in nato drugi, mora ostati izhod krmilja na vrednosti 0.

## Osnovna tabela prehajanja

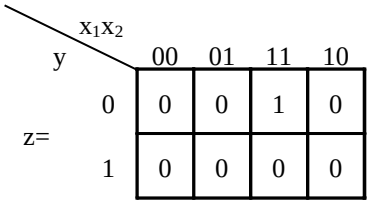
| y \ $x_1x_2$ | 00 | 01 | 11 | 10 | z | |
|---|---|---|---|---|---|---|
| 1 | (1) | 2 | 4 | 3 | 0 | oba vhoda izključena |
| 2 | 1 | (2) | 5 | 3 | 0 | drugi vhod vključen, prvi izključen |
| 3 | 1 | 2 | 5 | (3) | 0 | prvi vhod vključen, drugi izključen |
| 4 | 1 | 2 | (4) | 3 | 1 | oba vhoda vključena hkrati |
| 5 | 1 | 2 | (5) | 3 | 0 | oba vhoda vključena drug za drugim |

## Okrajšana in kodirana tabela prehajanja ter izhodna tabela



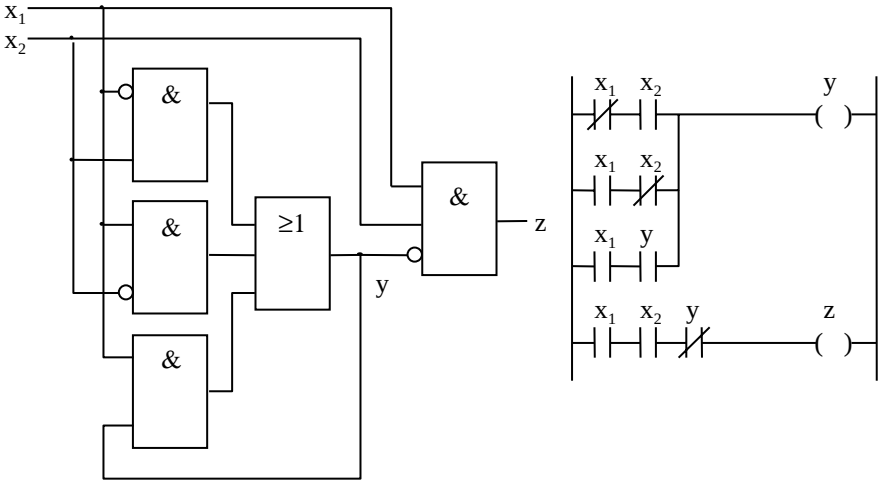




$$y' = \bar{x}_1 x_2 + x_1 \bar{x}_2 + x_1 y$$




$$z = x_1 x_2 \bar{y}$$

## Realizacija v PLK



## Slabosti Huffmanovega postopka

- Postopek je bil zasnovan za potrebe načrtovanja asinhronskih sekvenčnih vezij
- Slabosti izvirajo iz
  - narave problemov sekvenčnega vodenja
  - razlik v delovanju elektronskega vezja in logičnega krmilnika
- Problem kompleksnosti krmilij
  - do pet vhodov in pet izhodov
  - število možnih stanj narašča eksponencialno s številom vhodov -> nepregledna tabela stanj

## Slabosti /2

- Krmilja v industriji imajo običajno več vhodov in izhodov
  - razgraditev na podsisteme
  - krmilja podsistemov dodatno razgradimo na osnovne operacije
- Način delovanja logičnih krmilnikov
  - delovanje je lahko odvisno od vrstnega reda prečk v lestvičnem diagramu
  - npr. rešitev prvega primera (prepoznavanje sekvence (00), (01), (11)) ni več pravilna, če zamenjamo 2. in 3. prečko - dodaten pulz na izhodu ((01;01) -> (10;10))

## Poenostavljen Huffmanov postopek

- Ne minimizira števila pomnilnih elementov
- Odpadejo problemi s kodiranjem stanj in preprečevanjem neželenih prehodov
- Primeren za načrtovanje krmilij, ki jih implementiramo v PLK
- Nekoliko večja poraba pomnilnika in daljši čas izvajanja programskega cikla, vendar ne bistveno
- Rezultirajoči program je laže razumljiv
  - lažje iskanje napak
  - enostavnejše vzdrževanje programa

## Poenostavljen Huffmanov postopek /2

- Prvi del postopka ostaja nespremenjen
  - izhajamo iz osnovne tabele prehajanja
- Celotni postopek:
  1. Na podlagi besednega opisa sestavimo osnovno tabelo prehajanja.
  2. Osnovno tabelo prehajanja poenostavimo z združevanjem vrstic.
  3. Vsaki vrstici okrajšane tabele priredimo eno notranje stanje. Tako dobljena stanja se medsebojno izključujejo, vedno je aktivno le eno stanje.

## Poenostavljen Huffmanov postopek /3

4. Poiščemo relacije med vpeljanimi notranjimi stanji in izhodi krmilja. Pri tem si pomagamo z osnovno tabelo prehajanja, v kateri so definirane vrednosti izhodov pri vsakem stabilnem stanju.
5. Poiščemo logične pogoje za postavljanje notranjih stanj na 1 (funkcije SET):
   - poiščemo vse prehode iz ostalih vrstic tabele v stabilna stanja v vrstici, ki jo obravnavamo,
   - postavitev vhodnih signalov in stanje notranjih spremenljivk v izvornem stanju takšnega prehoda sestavimo v logični pogoj za prehod,
   - logični ALI vseh tako dobljenih pogojev je pogoj za postavitev notranjega stanja, ki pripada obravnavani vrstici.

## Poenostavljen Huffmanov postopek /4

6. Poiščemo logične pogoje za postavljanje notranjih stanj na 0 (funkcije RESET):
   - poiščemo vse prehode iz nestabilnih stanj obravnavane vrstice v stabilna stanja ostalih vrstic v tabeli,
   - iz stabilnih stanj, kjer se ti prehodi končujejo, podobno kot v prejšnjem koraku sestavimo logične pogoje,
   - logični ALI vseh teh pogojev je pogoj za reset notranjega stanja, ki pripada obravnavani vrstici.

- Postopek ilustrirajmo na že obdelanem primeru
  - želimo načrtati krmilje, ki prepoznava sekvenco (00,01,11)

## Osnovna tabela prehajanja

- Pričnemo pri stanju vhodov 00, to stanje vnesemo v tabelo kot stanje 1
- Obravnavamo tri možne prehode:
  (00)->(01), (00)->(10) in (00)->(11)
- Dobljena stanja 2, 3 in 4 so nestabilna, vsakemu priredimo novo vrstico ...

y'=

| y \ $x_1x_2$ | 00 | 01 | 11 | 10 | z |
|---|---|---|---|---|---|
| 1 | (1) | 2 | 4 | 3 | 0 |
| 2 | 1 | (2) | 5 | 3 | 0 |
| 3 | 1 | 6 | 4 | (3) | 0 |
| 4 | 1 | 6 | (4) | 3 | 0 |
| 5 | 1 | 6 | (5) | 3 | 1 |
| 6 | 1 | (6) | 4 | 3 | 0 |

## Okrajšana tabela in notranja stanja

- Vsaki vrstici okrajšane tabele prehajanja priredimo notranje stanje
- Določimo relacijo notranjih stanj z izhodom

| y \ $x_1x_2$ | 00 | 01 | 11 | 10 | notranje stanje | z |
|---|---|---|---|---|---|---|
| a | (1) | 2 | 4 | 3 | $y_1$ | 0 |
| b | 1 | (2) | 5 | 3 | $y_2$ | 0 |
| c | 1 | (6) | (4) | (3) | $y_3$ | 0 |
| d | 1 | 6 | (5) | 3 | $y_4$ | 1 |

$z = y_4$

## Funkcije za postavitev stanja

- Prva vrstica tabele:
  - edino stabilno stanje je stanje 1
  - obravnavamo prehode v pripadajočem stolpcu
  - logični pogoj za vstop v stanje 1:

$$S_1 = \bar{x}_1\bar{x}_2y_2 + \bar{x}_1\bar{x}_2y_3 + \bar{x}_1\bar{x}_2y_4 = \bar{x}_1\bar{x}_2(y_2 + y_3 + y_4)$$

  - ker vsi prehodi v tem stolpcu vodijo v isto stanje, lahko funkcijo za vstop v stanje 1 poenostavimo:

$$S_1 = \bar{x}_1\bar{x}_2$$

## Funkcije za postavitev stanja /2

- Druga vrstica tabele:
  - edino stabilno stanje je stanje 2
  - edini prehod v to stanje je iz prve vrstice, zato:

$$S_2 = \bar{x}_1x_2y_1$$

- Tretja vrstica tabele:
  - tri stabilna stanja (6, 4, 3)
  - prehodi: v stanje 6 iz 4. vrstice, v stanje 4 iz 1. vrstice in v stanje 3 iz poljubne vrstice:

$$S_3 = \bar{x}_1x_2y_4 + x_1x_2y_1 + x_1\bar{x}_2 = \bar{x}_1x_2y_4 + x_1(y_1 + \bar{x}_2)$$

## Funkcije za postavitev stanja /3

- Četrta vrstica tabele
  - edino stabilno stanje je stanje 5
  - edini prehod v to stanje je iz druge vrstice, zato:

$$S_4 = x_1 x_2 y_2$$

- Podobno določamo funkcije za reset oziroma izstop iz stanja
  - iščemo prehode iz obravnavane vrstice v stabilna stanja

## Funkcije za reset stanja

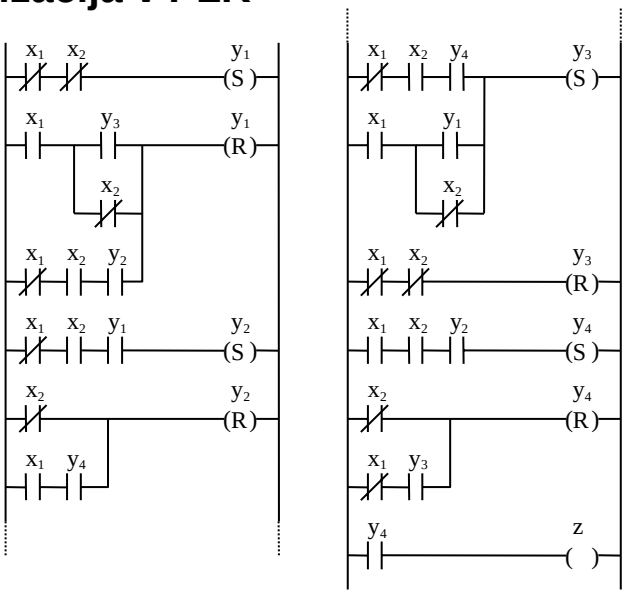
- Prva vrstica tabele:
  - trije prehodi iz te vrstice v stabilna stanja - v stanje 2 (druga vrstica), v stanje 4 (tretja vrstica) in v stanje 3 (tretja vrstica):

$$R_1 = \bar{x}_1 x_2 y_2 + x_1 x_2 y_3 + x_1 \bar{x}_2 = \bar{x}_1 x_2 y_2 + x_1 (y_3 + \bar{x}_2)$$

- Podobno dobimo:

$$R_2 = \bar{x}_1 \bar{x}_2 + x_1 x_2 y_4 + x_1 \bar{x}_2 = \bar{x}_2 + x_1 x_2 y_4 = \bar{x}_2 + x_1 y_4$$
$$R_3 = \bar{x}_1 \bar{x}_2$$
$$R_4 = \bar{x}_1 \bar{x}_2 + \bar{x}_1 x_2 y_3 + x_1 \bar{x}_2 = \bar{x}_2 + \bar{x}_1 x_2 y_3 = \bar{x}_2 + \bar{x}_1 y_3$$

**Realizacija v PLK**

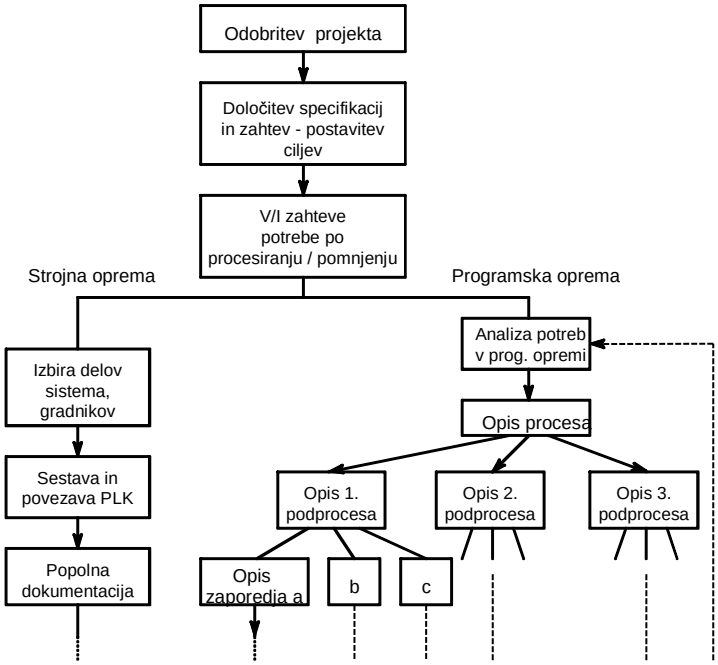


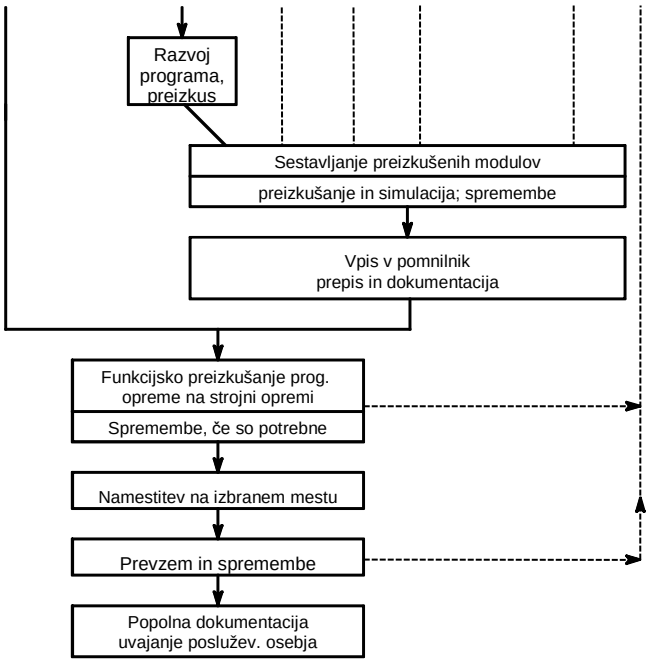

# 4.3 Planiranje projekta sekvenčnega vodenja

- Razdelitev procesa na podsisteme in pripadajoča opravila
- Specifikacija podsistemov in opravil
  - definiramo tako delovanje kot elemente vodenja
  - električni, mehanski in logični vhodi ter izhodi za vsako opravilo
  - medsebojne odvisnosti in potrebni usklajevalni mehanizmi
- Definicija varnostnih zahtev
  - vezja za izklop v sili, blokade ipd. - izven krmilnika!

# Planiranje projekta sekvenčnega vodenja /2

- Definicija operaterskega vmesnika
  - prikazovalniki, signalne lučke, zvočni signali
  - tipke, stikala
- Konfiguracija sistema PLK
  - tip CPE
  - število in tip vhodno izhodnih modulov
  - konfiguracija vhodov in izhodov






## Dokumentacija

- Splošen opis
  - glavne karakteristike uporabe:
  - datum, ime, tip, število izvedenih nalog, število podprogramov itd.;
- Arhitektura strojne opreme
  - število ohišij, tip V/I modulov,
  - konfiguracija pomnilnika,
  - povezave V/I modulov;
- Konfiguracija programa
  - tip in perioda uporabljenih opravil,
  - število podprogramov;

## Dokumentacija /2

- Konfiguracija sekvenčnega funkcijskega diagrama
  - število korakov, makrokorakov;
- Konfiguracija funkcijskih blokov
  - podaja parametre: časovnikov, števcev, registrov, drugih blokov;
- Konstante
- Izpis programa
  - v tekstovnem ali grafičnem jeziku s komentarji.