Univerza v Ljubljani

Fakulteta za elektrotehniko

Gal Nadrag

Asinhrona sekvenčna digitalna vezja

Magistrsko delo

Magistrski študijski program druge stopnje Elektrotehnika

Mentor: Aleksander Sešek

Ljubljana, 2023

Zahvala

Hvala Sandi, car si. Karin hvala za smehe orehe. Dru?ina, hvala de ste mi težil ko se mi ni dal. Pa še faks je tud dost kul. Pa folk na šihtu

Povzetek

V zadnjem desetletju smo bili priča prenehanju povečevanja taktnih frekvenc posameznih procesorskih jeder. A hkrati število tranzistorjev še vedno narašča. Trenutni trend je povečevanje števila jeder, vendar nekaterih problemov ni mogoče paralelizirati, zato ta pristop ne pomaga.

V tem delu predstavljamo metodo za načrtovanje asinhronih digitalnih vezij. Ta vezja ne potrebujejo taktnega signala in so hitrostno omejena le z omejitvami vrat in povezav. Predstavljamo tudi metodo za implementacijo takšnih vezij v FPGA. Implementacije osnovnih vezji doežejo frekvenco 35MHz kar je jakoma počasi. Tako pride.

Ključne besede: Asinhrona logika, Digitalna logika, FPGA

Abstract

In the last decade, we have witnessed the cessation of the increase in clock frequency of individual processor cores. At the same time, the number of transistors continues to grow. The current trend is to increase the number of cores, but some problems cannot be parallelized, so this approach does not help.

In this work, we present a method for designing asynchronous digital circuits. These circuits do not require a clock signal and are speed-limited only by gate and connection constraints. We also present a method for implementing such circuits in FPGA.

Key words: Asynchronous logic, Digital logic, FPGA

Vsebina

Seznam slik

Seznam tabel

Seznam uporabljenih simbolov in kratic

V pričujočem zaključnem delu so uporabljene naslednje veličine in simboli:

Kratica	**Pomen**
FPGA	Field programmable gate array
LEDR	Level-Encoded Dual-Rail

Veličina / oznaka		**Enota**	
Ime	Simbol	Ime	Simbol
čas	t	sekunda	s
frekvenca	f	Hertz	Hz
napetost	U	Volt	V
tok	I	Amper	A
kapacitivnost	C	Farad	F

1 Uvod

Digitalna vezja uporabljamo za obdelavo podatkov. Osnovna digitalna vezja glede na vhodne podatke naredijo izhodne podatke, torej nimajo spomina. Taka vezja imenujemo **kombinatorična** vezja. Z kombinatoričnimi vezji lahko izračunamo kar koli želimo a imamo težavo:

Velikost: velikost kombinatoričnega vezja je v splošnem proporcionalna kvadratu števila vhodov in proporcionalna številu izhodov. Ker želimo obdelovati gigabajte podatkov in več je to jasno nesprejemljivo.

Rešitev je, da kose vezja uporabimo večkrat, torej da naredimo povratno vezavo. Izhod takih vezji ni več odvisen le od vhodov, temveč tudi od stanja povratne zanke, torej imajo taka vezja spomin, imenujejo se **sekvenčna** vezja.

Ampak tu se pojavi nova težava. Predstavljajmo si seštevalnik, katerega izhod povežemo na enega izmed vhodov, Na drugi vhod damo konstanto 1. Želeli bi si, da vezje šteje 1,2,3... Ampak to se ne zgodi, namesto tega na izhodu dobimo kaotične signale. Razlog je, da je v vezju ogromno število povratnih zank, vendar med seboj niso sinhronizirane. Zato zanke počasi zlezejo iz faze in na izhodu dobimo šum.

Nujno je torej vpeljati mehanizem s katerim periodično sinhroniziramo povratne zanke. To storimo s spominskimi elementi, ki jih vstavimo v povratno vezavo, rečemo jim **registri**. Registri ne smejo prepustiti naprej novih podatkov, dokler niso stari podatki popolnoma obdelani. Efektivno ponovno sinhronizirajo faze vseh povratnih zank.

Torej moramo izvedeti kdaj so podatki na vhodu registra obdelani, in podatki

trenutno v registru obdelani. Takrat lahko register sprejme nove podatke.

V splošnem imamo dva načina na katera lahko storimo:

-Globalna sinhronizacija(Urin takt): V tej shemi uporabimo predpostavko, da obdelva podatkov v **vseh** vezjih med dvema zaporednima registroma ne traja dlje od neke konstante. Vse registre nato naenktrat sinhroniziramo z to periodo.

-Lokalna sinhronizacija V tej shemi sinhroniziramo vse povratne zanke na en signal. Zagotoviti moramo, da ima ta signal **najdaljšo** periodo izmed vseh povratnih zank.

2 Teoretične osnove in pregled literature

2.1 Teorija asinhronega prenosa in obdelave podatkov

2.1.1 Osnove / Hitrostna neodvisnost

Naredimo model asinhronih vezji, da bolje razumemo kako delujejo. Predstavljajmo si vezje vrat, povezanih med seboj. Vsaka vrata imajo vhodne in izhodne signale. Ko vhod vrat diktira spremembo izhoda postanejo vrata aktivna. Po neznanem časovnem zamiku se spremeni tudi izhod. Zanimivo je, kar se zgodi na izhodu vrat. Vrata ki imajo na vhodu izhodini signal naše originalnih vrat imajo tri opcije. Lahko sprememba vhoda ne spremeni izhoda. Lahko sprememba vhoda ne povzroči spremembe izhoda. Lahko pa so bila vrata aktivna, a zaradi spremembe izhoda sedaj niso več aktivna.

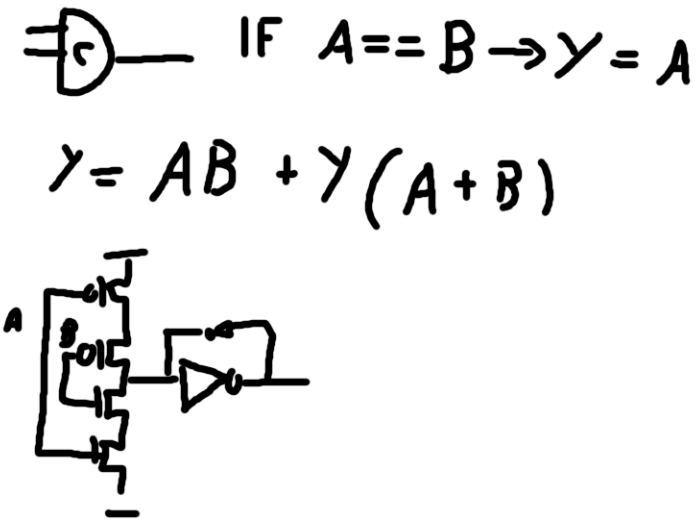
Vezja, kjer se zadnji kriterij nikoli ne zgodi so hitrostno neodvisna.

Moramo poskrbeti da si signali sledijo v urejenem zaporedju.

Poglejmo katere predpostavke uporabljamo:

Hitrostno neodvisna vezja imajo neznane zakasnitve v vratih in nimajo zakasnitev v žicah. To žal ni zelo realno.

Vezja, ki so neodvisna na zakasnitve imajo neznane zakasnitve v žicah in vratih, tu ne predpostavimo nič, ampak takih vezji je zelo malo. Citat



Slika 2.1:

Srednja pot je kvazi zakasnitvena neodvisnost, kjer imamo določene predpostavke, o zakasnitvah v žicah.

2.1.2 Osnovna vezja

Če želimo načrtovati asinhrona vezja je torej potrebno sinhronizirati vse signale z najpočasnejšim. Poglejmo si kako lahko to dosežemo.

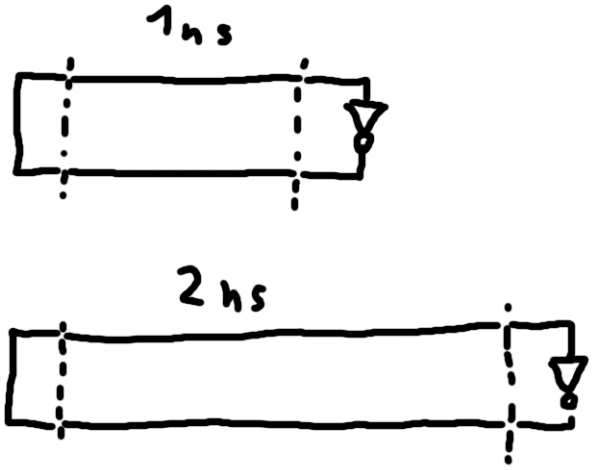
2.1.2.1 Muller C element

Muller C element je osnovni element sinhronizacije dveh signalov. Ima dva vhoda, izhod se spremeni, ko sta oba vhoda enaka.

Poglejmo si kako tak element sinhronizira dva signala:

Imamo dve povratni zanki, vsaka z različno zakasnitvjo, kateri želimo sinhronizirati.

Če ustvarimo prek C elementa skupno povratno zanko, mora hirejši vedno



Slika 2.2:

čakati počasnejšega, torej sta sinhronizirana.

Trivialno lahko sinhroniziramo poljubno število povratnih zank.

To lahko razumemo kot vzporedno vezavo oscilatorjev, kjer je skupna perioda, perioda najdaljše povratne zanke. Lahko pa vežemo oscilatorje tudi vzporedno na sledeči način:

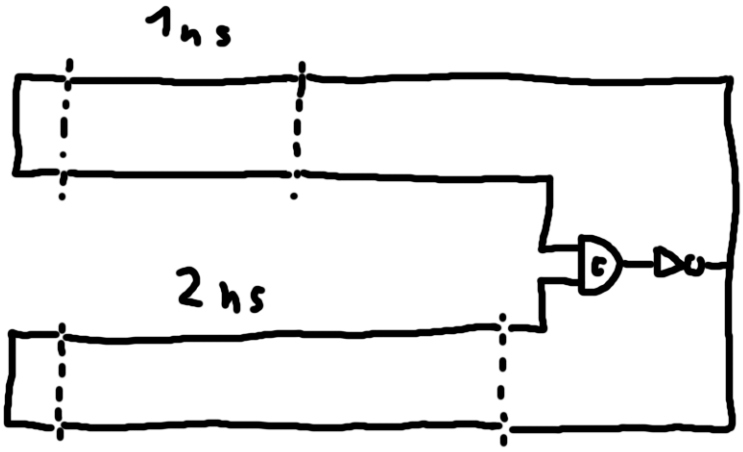
Tu povratne zanke prepletemo med seboj. Vsako povratno zanko sprožijo njeni sosedje. V tem primeru prva zanka sproži drugo. Ko druga konča ponovno sporži prvo.

Tak vzorec lahko nadaljujemo poljubno dolgo.

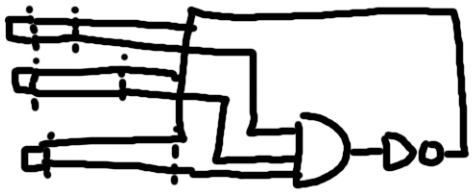
2.1.2.2 Cevovodi

Linija iz knjige: A register may input and store a new data token from its predecessor if its successor has input and stored the data token that the register was pre- viously holding.

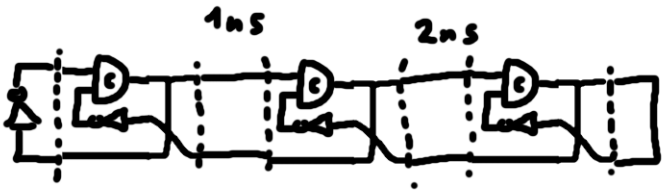
Če pogledamo dano vezje lahko vidimo, da deluje kot 1-biten FIFO. Podatki



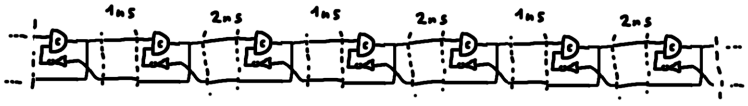
Slika 2.3:



Slika 2.4:



Slika 2.5:



Slika 2.6:

se nalagajo noter in črpajo ven. Temu konstruktu se reče Mulerjev cevovod, in je osnova vseh asinhronih vezji.

Lahko naredimo tudi obroče, Ali več sklenjenih obročov...

Osnovna celica cevovoda je..., če pogledamo logične funkcije: ..., To lahko izvedemo tudi na druge načine.

2.1.2.3 Podatkovno procesiranje

Sedaj imamo cevovode, želimo med stopnjami cevovoda procesirati podatke. Dva načina:

- vzporedno imamo default logiko - Bundlede data
- Vzporedni sklopljeni cevovodi

Vzporedno logika, timing assumptions, delays kako prožit latche, samo pol jih ima naenkrat noter data, isto kot v sinhronih vezjih.

Vzporedni cevovdi, kako sklopit, kako procesirat podatke ect...

2.2 Protokoli

Zgornja načina plus 2/4 phase.

- 2phase hitrejsi, ampak rabimo feedback da vemo kdaj smo done
- 4phase preprostejši več komunikacije

Kartezični produkt:

2.2.1 4-Phase dual rail

Karl Fant, dve žici na bit ect

2.2.2 4-bundled data

lepa logika, ampak potrebujemo zakasnitve (lahko predvidimo zakasnitve)

2.2.3 2-Phase dual rail

Ta magisterska

2.2.4 2-bundled data

Micropipelines ect.

2.3 Podatkovni potek

Podatki se pretakajo po asinhronih vezjih kot valovi. v eno smer gre data, v drugo gre ACK. Vedno moramo kontrolirati število podatkovnih valov v vezju ect. To lahko gledamo kot graf po katerem se pretekajo tokeni.

Osnova igre ect:

Drugače za 2 phase kot za 4 phase:

2.3.1 2phase

Najmanj 2 v ciklu poljubno število, inicializacija. Vrsta grafa, za to

2.3.2 4phase

Najmanj 3 v ciklu sodo število, inicializacija. Vrsta grafa, za to

3 Metodologija

3.1 Arhitektura FPGA

FPGA so vezja za implementacijo specializirane logike.

Narejena so iz množice spominskih in kombinatoričnih celic. Kombinatorične celice so mali kosi spomina, s katerimi definiramo pravilnostno tabelo naše funkcije. Spominske celice so navadni DFF. Pomembna stvar so povezave med celicami, ki so konfigurabilne. To pomeni, da mi določimo pravilnostne tabele in povezave med njimi.

Obstajajo tudi prenosne verige, reset in clk linije in vgrajeni spomin, množilniki CPU ect.

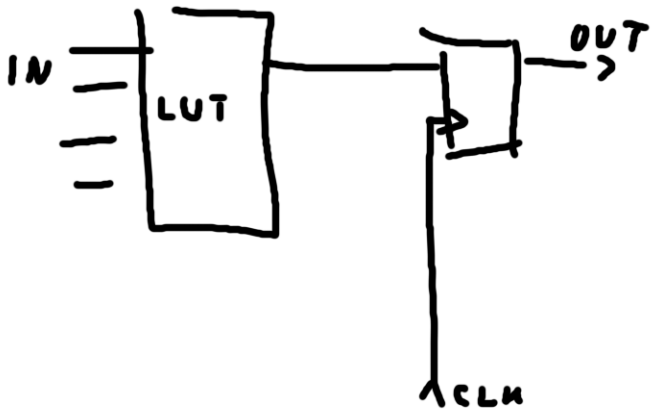
Ima možnost povratne vezave, tako da lahko gledamo kot max preprost turing machine???, na sploh lohk iz teh blokov zelo lahko delamo vezja po kopitu register,logika,register...ect

3.1.1 FPGA Celice

FPGA Celica zgleda pribljižno takole:

3.1.1.1 LUT

Lut je mehn košček spomina. Notr ma par bitou, direk implemenira tabelo



Slika 3.1:

3.1.1.2 FF

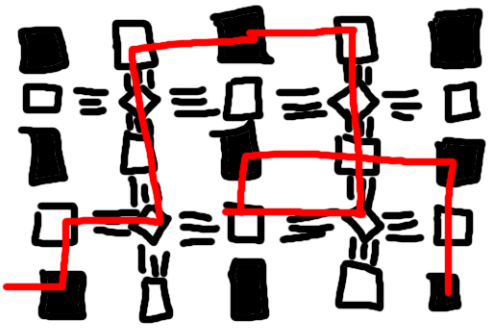
FF je preprost FF, ima puno option bitov ect

3.1.2 FPGA povezave

Povezave so večino površine, grejo med celicami, so programibilne, torej imajo puno tranzistorjev, torej zakasnitve...

3.2 Verilog

Jezik za opis procesov, uporablja se za programiranje FPGA



Slika 3.2:

3.3 Implementacija

3.3.1 4-Phase dual rail

Za impelementacijo tega protokola potrebujemo posplošene C elemente, Logika je narejena iz spominskih celic, data/ null waves ect

3.3.2 2-Phase dual rail

Podobno, kot pri 4 Phase ampak uporabljamo robove, potrebujemo več tranzistorjev za logiko, saj potrebujemo povratno informacijo, od izhodov

3.3.3 Implementacija cellic

3.3.3.1 Primitivi

Imamo 5 Primitivov, iz katerih sestavimo ostalo funkcionalnost vezja:

C muller C, razne implementacije, rezultati P za TP C splošno, drevesna struktura ect

3.3.3.2 Funkcionalne celice

injektorji pričnejo delovanje cikla cmpl det preveri če je stanje validno adder sestevalnik mem reg spominska celica

3.3.3.3 Vezja

FIB fibbonacci counter, 1,2,3,5,8,13 toy example predstavi inicalizacijo

GCD, ciklična komputacija, preprostejša inicializacija

RISCV čim bolj preprost processor, proof of concept

4 Rezultati

4.1 PIPELINE

Frekvenca, zanesljivost, meritve, ect

4.2 CNT

Frekvenca, zanesljivost, meritve, ect

4.3 FIB

Frekvenca, zanesljivost, meritve, ect

4.4 GCD

Zamik, zanesljivost

4.5 RISCV

Bomo vidl

5 Zaključek

Načrtali smo Asinhrona vezja in jih implementirali v FPGA. Kljub Suboptimalni arhitekturi FPGA smo dosegli delujoče rezultate in ogrodje za nadalnje delo. Hitrosti vezji so nižje kot bi upali.

Literatura

[1] T. Oetiker, H. Partl, I. Hyna in E. Schlegl, *Ne najkrajši uvod v LaTeX 2ε, The not so short introduction to LaTeX 2ε*. Elektronska verzija dostopna na http://www-lp.fmf.uni-lj.si/plestenjak/vaje/latex/lshort.pdf, 2006. Bor Plestenjak, Slovenski prevod in priredba.

Dodatek

A Urejanje dokumentov z orodjem LaTeX

Korak 1 Avtor kreira tekstovno datoteko s končnico *.tex*, ki vsebuje tekst in ukaze za oblikovanje teksta (oziroma se uporabi izvorna koda te predloge). Dober uvod v delo z ukazi LaTeX so spletna navodila [1]. Za pisanje je lahko uporabljen katerikoli tekstovni urejevalnik. Priporočamo uporabo urejevalnikov WinEdt[1] ali TeXstudio[2], ki sta namenski orodji z integriranimi ikonami za posamezne korake. Urejevalnika vsebujeta tudi slovar slovenskih besed[3] za sprotno preverjanje in deljenje besed. Na spletu so na voljo tudi hitri priročniki z LaTeX ukazi[4] in spletni urejevalniki[5].

Korak 2 Prevajanje izvorne datoteke s prevajalnikom MiKTeX. Možnost direktnega prevajanja v PDF dokument (ikona LaTeX) pri prvem prevajanju ustvari tudi listo citatov in sklicevanj (datoteka *.aux*).

Korak 2.1 [6] Zagon BibTeX prevajanja (ikona *Bib*), ki na osnovi *.aux* datoteke in podatkov iz baze referenc, ustvari oblikovan spisek referenc (datoteka *.bbl*) glede na izbran stil citiranja (datoteka *.bst*).

Korak 2.2 [7] Ponovno prevajanje s prevajalnikom MiKTeX, ki v glavni dokument vključi oblikovane reference iz datoteke *.bbl*.

[1] Dostopno na: http://www.winedt.org
[2] Dostopno na: http://texstudio.sourceforge.net/
[3] Dostopno na: http://www.winedt.org/Dict
[4] Primer dostopen na: https://en.wikibooks.org/wiki/Category:Book:LaTeX
[5] Primer dostopen na: https://overleaf.com
[6] Potrebno samo pri navajanju virov s pomočjo orodja BibTeX
[7] Potrebno samo pri navajanju virov s pomočjo orodja BibTeX

Korak 3 Ponovno prevajanje s prevajalnikom MiKTeX, ki poveže spisek referenc z navedki v tekstu.

Korak 4a Pretvorba oblikovanega dokumenta v format PostScript in nato izvoz v obliki PDF dokumenta:

- ikona DVI-PS - pretvorba v datoteko *.ps*
- Ogled PostScript datoteke s programom GhostView
- Pretvorba v PDF dokument: GhostView: File/Convert/pdfwrite, pri čimer je potrebno izbrati parametre za format PDF/A glede na spletna navodila[8].

V tem primeru morajo biti vse vključene slike v formatu PostScript. V tem načinu je možna tudi uporaba orodja PSfrag, ki omogoča zamenjavo tekstovnih elementov na originalni sliki s poljubnim tekstom ali enačbo.

Korak 4b Pretvorba oblikovanega dokumenta neposredno v PDF format. Ikona PDFTexify. V tem primeru so vključene slike lahko le v formatu PDF, PNG, JPEG ali GIF. Format izhodnega dokumenta PDF/A, ki je zahtevan za oddajo v Repozitorij UL, je podprt v tej LaTeX predlogi[9]. Alternativna možnost je generiranje standardne PDF datoteke in poznejša pretvorba v format PDF/A. To je možno narediti z enim izmed plačljivih programov (npr. Adobe Acrobat Professional, Nitro Pro) ali s spletnimi pretvorniki[10]. Pri slednjih je potrebno biti pozoren na morebitne vodne žige ali napake, ki lahko nastanejo pri pretvorbi.

[8]Dostopno na: http://svn.ghostscript.com/ghostscript/trunk/gs/doc/Ps2pdf.htm#PDFA
[9]Dodatne informacije: https://www.mathstat.dal.ca/~selinger/pdfa/
[10]Primer: https://docupub.com/pdfconvert/

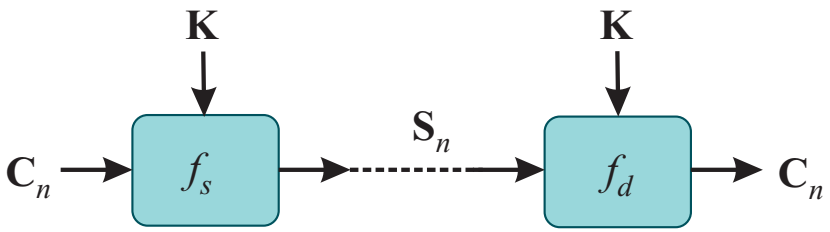
B Vključevanje slik v LaTeX dokumente

Slike vključujemo z ukazom \includegraphics v okolju figure. EPS slike morajo biti shranjene brez glave z bitno sliko za predogled. Dodaten paket PSfrag omogoča zamenjavo (prepis) napisov na vektorski sliki. Za uporabo je potrebna vključitev orodja z ukazom \usepackage{psfrag}. Primer LaTeX kode za zamenjavo napisa *test* na sliki z LaTeX simbolom ϵ [μ] je:

```
\begin{figure}[h]
\centering
\psfrag{test}[B1][B1][1][0]{$\epsilon \;[\mu]$}
\includegraphics[width=0.75\columnwidth]{primer_vektorske_slike.eps}
\caption{\label{oznaka_slike} Primer slike.}
\end{figure}
```

Za vključitev vektorske slike je možno uporabiti tudi makro \epsslika, ki je vključen v stil za predlogo. Prvi parameter v makroju \epsslika je podnaslov, drugi pa je ime datoteke s sliko brez končnice (privzeta končnica je .eps) in hkrati tudi oznaka za sklicevanje na sliko. Pri uporabi tega ukaza morajo biti datoteke EPS v korenu delovnega direktorija. Za vključevanje slika ne sme imeti glave z bitno sliko za predogled. Pri stilu je za vključevanje slik potrebno izbrati ustrezno opcijo pdftex ali pctex, glede na to katero distribucijo LaTeX prevajalnika se uporablja za prevajanje. Primer je prikazan na sliki B.1.

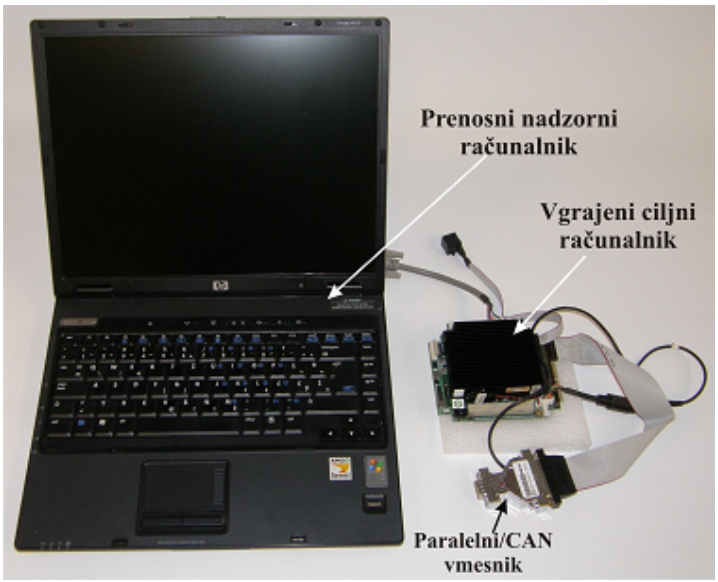
Za vključevanje bitnih slik je v predlogi na voljo makro \jpgslika. Prvi parameter v makroju \jpgslika je podnaslov, drugi pa je ime datoteke s sliko (privzeta končnica je .jpg) in hkrati tudi oznaka za sklicevanje na sliko. Pri uporabi tega



Slika B.1: Primer vektorske slike EPS.

ukaza morajo biti slikovne datoteke v korenu delovnega direktorija. Slike so v tekst vključene v originalni velikosti.

Slika B.2 predstavlja primer vključitve bitne slike JPG formata velikosti 9,4 x 7,6 cm.



Slika B.2: Primer vključitve bitne slike: sistem vodenja.

C Namestitev programskih orodij za urejanje LaTeX dokumentov

C.1 Sistemi Windows

Korak 1 Namestitev paketa MiKTeX, ki je prevajalnik za dokumente, napisane v okolju LaTeX. Datoteke so dostopne na spletu: `http://miktex.org/`

Korak 2 Namestitev tekstovnega urejevalnika. Nekaj priporočenih:

- TeXstudio (`http://texstudio.sourceforge.net/`)
- Texmaker (`http://www.xm1math.net/texmaker/`)
- Geany (`http://www.geany.org/`)
- WinEdt (`http://www.winedt.com/`)

Korak 3 Namestitev ogledovalnika PostScript dokumentov:

- modul GhostScript
- modul GhostView

Datoteke so dostopne na: `www.cs.wisc.edu/~ghost/`

Pri prvem prevajanju se lahko zgodi, da MiKTeX poskusi namestiti potrebne LaTeX knjižnice (pakete), kar mu je treba dovoliti, sicer predloga ne more delovati. Zaradi prenašanja in nameščanja prvo prevajanje lahko spodleti, kar je napaka paketa MiKTeX.

C.2 Sistemi Linux

Na sistemih Linux je potrebno za uporabo predloge namestiti naslednje pakete:

- `texlive-lang-european`
- `texlive-lang-greek`
- `texlive-latex-extra`

Imena paketov so povzeta po znani Linux distribuciji Ubuntu. V drugih distribucijah se enaki paketi lahko imenujejo tudi malo drugače. Za urejanje dokumentov se priporočajo naslednji tekstovni urejevalniki:

- TeXstudio (paket `texstudio`)
- Texmaker (paket `texmaker`)
- Geany (paket `geany`)