Prenos pristopov uporabljanih pri razvoju programske opreme v EDA okolja

Mitja Nemec

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenija
E-pošta: mitja.nemec@fe.uni-lj.si

Software development practices in EDA environment

Abstract. *This paper presents how the processes which are well known from software development practices are slowly penetrating into design of printed circuit boards. The biggest impacts come from source code management tools which form a foundation for more advanced tools and services. Open source also has an effect by exposing EDA environment to developers with different background than electrical engineering which generates fresh and completely unheard of solutions.*

1 Uvod

Orodja za računalniško podprto načrtovanje električnih vezij (Electronic Design Automation – EDA oziroma Electronic Computer-Aided Design - ECAD) se v grobem delijo na orodja za:

- načrtovanje integriranih vezij
- načrtovanje tiskanih vezij

V tem članku se bomo osredotočili na orodja za načrtovanje tiskanih vezij in osnovna funkcionalnost ki jih ta orodja omogočajo je zajem električne sheme in postavitev elementov na vezje ter avtomatsko ali ročno ustvarjanje bakrenih povezav na podlagi električne sheme. Za načrtano tiskano vezje se nato generira datoteke, ki jih proizvajalci rabijo za njegovo izdelavo (gerber, drill, …). V večini orodja omogočajo tudi izdelavo 3D modela tiskanega vezja skupaj s komponentami kar omogoča integracijo z mehaničnimi CAD programi.

Orodja tudi omogočajo vzpostavitev in vzdrževanje knjižnic s simboli in odtisi komponent, naprednejša orodja pa lahko knjižnico simbolov povežejo s podatkovno bazo kar omogoči dodatne funkcionalnosti (sledenje zalogam, pregled dobavljivosti, izdelava stroškovnika, …).

Razvoj EDA orodij se je začel v osemdesetih letih prejšnjega stoletja, ki so tekom desetletji prerasla v zelo zmogljiva a kompleksne orodja [1]. Smer razvoja diktirajo njihovi uporabniki, saj podjetja, ki jih razvijajo, lahko dobijo prednost na trgu le, če se njihovim uporabnikom nove funkcionalnosti, ki jih podjetje ponudi, zdijo vredne investicije. Tako lahko v grobem trdimo, da inženirji elektrotehnike v večjem delu usmerjajo razvoj EDA orodij.

V zadnjem desetletju pa je opazen napredek teh orodij tudi na področjih ki so bliže računalništvu in informatiki in v nadaljevanju si bomo pogledali v katero smer lahko pričakujemo, da se bodo ta orodja razvijala v prihodnosti.

Pri odprtokodnih EDA orodjih je smer razvoja javno objavljena [2] in tudi uporabniki teh programov velikokrat objavijo na kak način jih uporabljajo [3]. Tako smo lahko pri odprtokodnih EDA orodij tudi dobro seznanjeni z samim procesom načrtovanja električnih vezij.

Zaradi zaprte narave zaprtokodnih EDA orodij bistveno težje sodimo o smeri njihovega razvoja in tudi uporabniki zaprtokodnih orodij ne razkrivajo na kak način jih uporabljajo. Tako je proces načrtovanja vezij pri katerih se uporablja zaprtokodna EDA orodja javnosti neznan.

Tako velja podati opozorilo, da je podana raziskava, zaradi teh dejstev, zelo verjetno pristranska.

2 Vpliv odprte kode

V devetdesetih letih prejšnjega stoletja so se pojavila prva odprtokodna EDA orodja. Pri odprtokodnih EDA okoljih pa pri razvoju sodelujejo tako inženirji elektrotehnike, ki poznajo tudi področje programiranja, kot tudi inženirji računalništva, ki poznajo področje elektronike. In prav slednji so domači s pristopi s katerimi se inženirji elektrotehnike tipično nikoli ne srečajo.

Odprte specifikacije in predvsem odprt in dokumentiran format zapisa posameznih datotek (shema, …) omogoča, da se lahko kdorkoli loti razvoja samostojnih aplikacij, ki se povežejo z EDA orodjem in tako razširijo nabor funkcionalnosti.

Tako se v/okoli odprtokodnih EDA okoljih pogosto pojavijo rešitve, ki izvirajo s področja računalništva in informatike. Nekatere od njih so uporabne in jih kasneje povzamejo tudi druga EDA orodja.

3 Preneseni pristopi

3.1 Vpeljava standardov

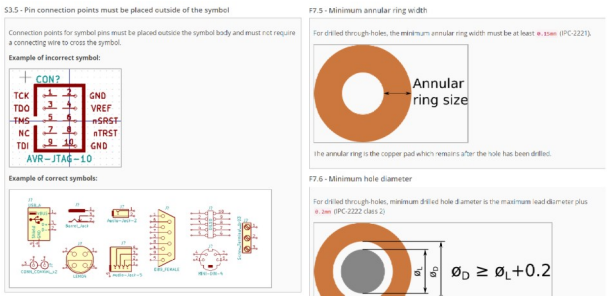
Pri razvoju programske opreme je pregled kode [4]–[6] del procesa. In za učinkovitost pregledov kode je treba kodo pisati na podlagi standardov, ki določajo slog, obliko in nabor funkcionalnosti jezika ki se smejo oziroma ki se ne smejo uporabljati [7], [8].

Na področju načrtovanja je ERC (electrical rules check) funkcionalnost na voljo že nekaj časa vendar pa le ta preverja samo električno smiselnost sheme (kratek stik med dvema napajalnima linijama, manjkajoče povezava …). Praktično enako stanje je na področju načrtovanja električnih vezij. DRC (design rules check) funkcionalnost preverja ali je vezje primerno za

izdelavo (dovolj velik razmik med bakrenimi povezavami, širina bakrenih povezav, …). Nekoliko bolje je na področju izdelave odtisov komponent. V zadnjih dveh desetletjih sta se dodobra uveljavila standarda IPC7351 in IPC-2222, ki nekoliko bolj podrobno opisujeta poleg električnih tudi ostale slogovne parametre odtisa (ime, dokumentacijski sloji, …). Z leti sta se standarda tako dopolnila, da danes obstaja že več orodij s katerimi lahko programsko generiramo odtise v celoti in ti ustrezajo standardu.

V grobem pa dlje od napotkov oziroma namigov [9], [10] še nismo prišli. Razlogov za to je verjetno več, eden izmed njih pa je zagotovo dejstvo, da je tako risanje sheme kot samega vezja v osnovi ročno grafično opravilo in računalnik tudi danes težko preverja ali je grafika po standardu ali ne. In če se grafika ne more avtomatsko generirati ali pa vsaj avtomatizirano preverjati, potem se je takih standardov težko držati. Vendar se tudi na tem področju kažejo premiki.

Eden od primerov je standard za vnos simbolov in odtisov v knjižnico [11]. Ta standard precej podrobno predpisuje kako naj bo izgled simbola (kje naj bodo napajalne nožice, kje naj bodo signalne nožice, kje naj bo tekst, kako velik naj bo tekst, poimenovanje nožic, …) in kakšen naj bo odtis komponente (lokacija nožice št. 1, rotacija komponente, velikost in položaj teksta, kateri tekst na katerem sloju, …). Ker se precej teh pravil lahko preverja z računalnikom se lahko to avtomatizira, kar pripomore k splošni sprejetosti standarda.



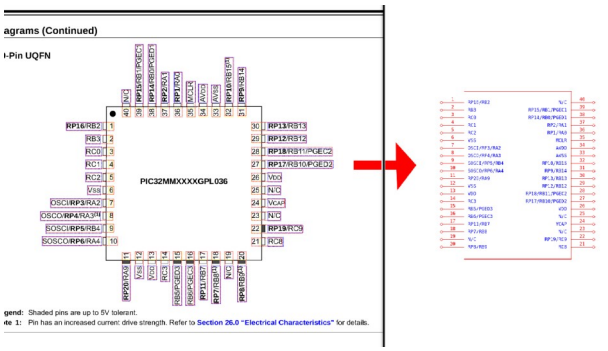
Slika 1: Izsek iz standarda KLC (KiCad Library Convention)

3.2 Programsko generiranje simbolov, odtisov, 3D modelov

Težnja po avtomatizaciji ponavljajočih procesov je v samem bistvu računalništva in informatike. To se najbolj kaže pri vseh orodjih ki se še vedno pojavljajo in so v pomoč pri ustvarjanju simbolov, odtisov in 3D modelov komponent.

Prvi primer je orodje, s katerim programsko iz PDF datoteke, ki opisuje komponento, generiramo simbol zapisan v formatu, ki ga uporablja EDA orodje [12]. Dodatno je mogoče nastaviti stil kakšen naj bo generiran simbol.

Drugi primer, je orodje s katerim lahko generiramo odtis induktivnosti [13]. Orodje omogoča da določimo število slojev, število ovojev in dimenzije odtisa ter nam generira odtis, s FEM orodjem izračuna induktivnost in upornost pri različnih frekvencah ter po potrebi tudi iterativno prilagodi dimenzije odtisa, da dosežemo želeno induktivnost.



Slika 2: Pretvorba simbola iz PDF datoteke v zapis, ki ga uporablja EDA orodje

Pri 3D modelih je osnova za avtomatizacijo CadQuerry knjižnica [14]. Z njeno pomočjo lahko s skriptami zapisanimi z jezikom python generiramo množico 3D modelov [15].

Poleg očitnega prihranka na času, ki ga omogoča avtomatizacija pri izdelavi simbolov, odtisov in 3D modelov, je dodatna prednost tudi zmanjšana verjetnost za napake zaradi človeškega faktorja.

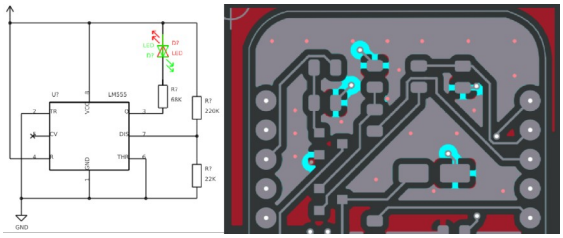
3.3 Pretvorbe med različnimi zapisi

Odprti formati zapisov datotek EDA okolji omogočajo pretvorbo med različnimi podatkovnimi tipi, ki tipično niso direktno podprta. To omogoča direktno povezavo različnih orodij [16], [17] brez uporabe vmesnih bolj standardnih zapisov (.step, dwg, .dxf, ...). Hkrati pa take pretvorbe omogočajo dostop do tehnologije izdelave tiskanih vezij področjem, ki temu sicer niso izpostavljena (umetnost, ...) [18].

3.4 Orodja za sledenje spremembam

Osnovna funkcionalnost, ki jo orodja za nadzor kode (SCM – source control management) nudijo je shranjevanje in sledenje različnim verzijam kode. Le to postane še bolj uporabno z orodij, ki omogočajo primerjavo dveh verzij (diff). Na področju računalništva imajo ta orodja že dolgo zgodovino in v tem času so šla čez več evolucijskih korakov. V zadnjem času orodje »git«, ki omogoča distribuiran nadzor kode, postaja de-facto standard na tem področju.

Ker orodja za nazor kode lahko delujejo nad datotekami v katerem koli zapisu so se pojavila različna orodja, ki omogočajo primerjavo različnih verzij [19]–[22].

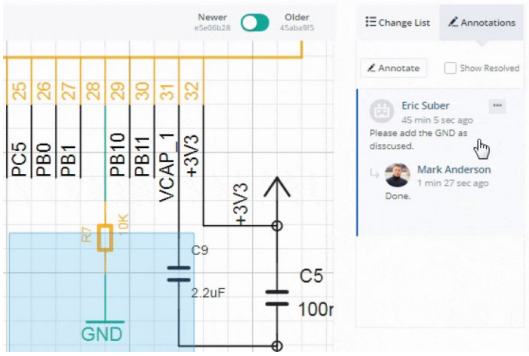


Slika 3: Grafični prikaz razlik na shemi na tiskanem vezju

3.5 Orodja in storitve za kolaboracijo

Podpora orodjem za nadzor kode pa je omogočila tudi razvoj novih storitev. Tako so se tudi za načrtovanje tiskanih vezij pojavile storitve, ki nudijo gostovanje

repozitorijev kamor se shranjujejo različne verzije. Poleg osnovnega gostovanja, te storitve podpirajo tudi pregled nad repozitorijem preko brskalnika in funkcionalnosti, ki omogočajo podpirajo sodelovanje (»pull request, veje, komentarji sprememb, …)[23], [24]. S »Pull request«-om sprožimo prošnjo za pregled in vključitev sprememb v repozitorij. Ko jih eden ali več sodelavcev pregleda in potrdi kot ustrezne, se spremembe dejansko vključijo v repozitorij. Pri tem procesu je v pomoč namenska spletna stran, na kateri lahko avtor in ostali diskutirajo o predlaganih spremembah. (slika 4).



Slika 4: Primer komentarja na spremembo sheme

Koncept vej pa je najbolj uporaben pri načrtovanju vezja. Pri zahtevnejših vezjih lahko preizkusimo več možnih razporeditev komponent ter poteka povezav in pri tem vsako možnost shranimo v svojo vejo. Tako lahko hitro prikličemo različne verzije in jih tudi med seboj primerjamo.

Nekatera EDA orodja podpirajo tudi hrambo knjižnic v repozitorijih SCM orodij. Le to omogoča lažje sodelovanje preko procesa, ki ga ta orodja podpirajo. V LRTME se pri EDA programu KiCad poslužujemo tega pristopa. Tako ima vsak študent in zaposleni svoj lokalen repozitorij s knjižnicami gradnikov. V njega lahko uporabnik vnaša nove gradnike kot tudi vnaša spremembe iz glavnega repozitorija. V glavni repozitorij pa se spremembe vnaša samo preko »pull request« mehanizma. Tako so v glavnem repozitoriju prisotni samo gradniki, ki so bili pregledani in potrjeni.

Pomembno področje kolaboracije je tudi med ECAD in MCAD orodij. V večini primerov to poteka preko datotek zapisanih v standardnih formatih (.step, idx) [25], vendar pa je ta kolaboracija zelo osnovna in v večini primerov enosmerna (iz ECAD v MCAD). Za nadgrajene možnosti ECAD-MCAD kolaboracije (dvosmerna kolaboracija, …) se pojavljajo povezave ECAD in MCAD okolji [26], ki pa so pri zaprtokodnih rešitvah tipično omejene na točno določene programe (tipično so proizvajalci lastniško povezani).

3.6 Vpeljava CI/CD pristopov

Z razmahom SCM orodij in storitev, ki temeljijo na njih, so se vzpostavili tudi CI/CD (continuous integration / continuous delivery) procesi v razvoju programske opreme[27]. Pri tem procesu se ob vsakem zapisu v repozitorij, ki se nahaja na strežniku požene avtomatiziran proces v katerem se lahko preverijo zadnje spremembe in po potrebi tudi pošljejo v uporabo.

Prvi tak primer v EDA okoljih je avtomatsko testiranje novih simbolov in odtisov v knjižnici ki je shranjena v repozitoriju. Tu se vsak simbol ali odtis preveri ali ustreza predpisanemu standardu (KLC). V kolikor avtomatski test potrdi skladnost se o tem obvesti upravljalce knjižnice, ki ob dodatnem ročnem pregledu potrdijo in simbol ali odtis se doda v knjižnico.

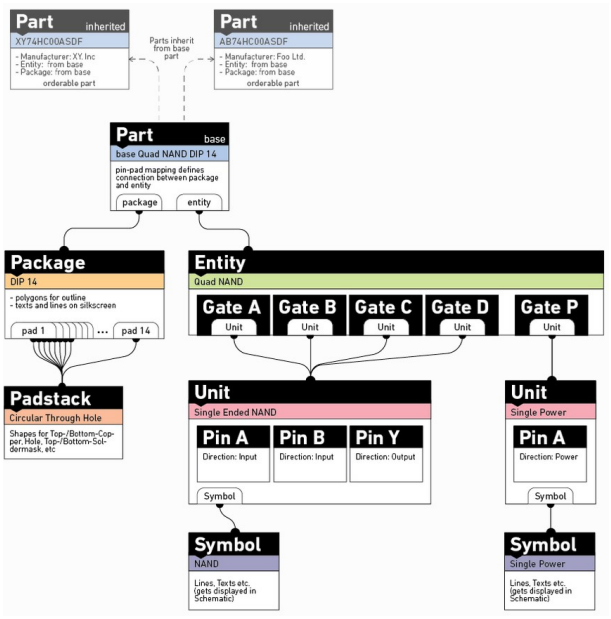
Drug primer je avtomatski test prekrivanj v 3D prostoru. Ko se zadnja verzija tiskanega vezja pošlje v repozitorij se požene test, ki preveri ali med 3D modeli komponent in ohišja prihaja do prekrivanja.

V pripravi so tudi rešitve pri katerih bi se zadnja verzija tiskanega vezja avtomatsko preverila z PDN (power delivery network) analizo, ki vključuje FEM analizo vendar pa je za avtomatiziranje tega procesa treba že na nivoju sheme vnesti dovolj specifikacij.

4 Popolnoma novi pristopi

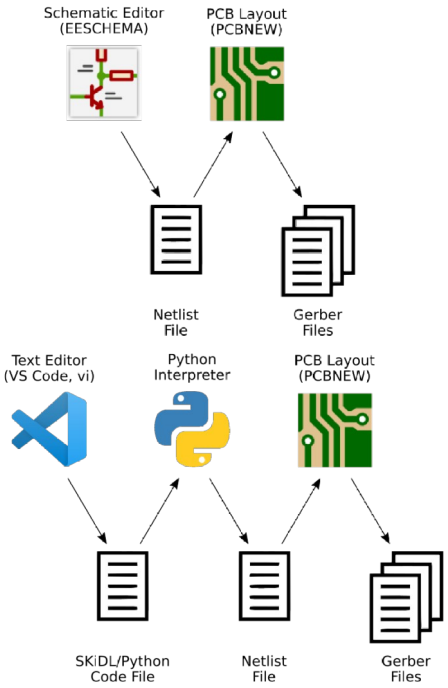
Do sedaj omenjeni pristopi predstavljajo bolj ali manj inkrementalne razvoj EDA orodij. Pojavljajo pa se tudi nekoliko bolj nekonvencionalne rešitve.

Prvi tak primer predstavlja EDA orodje [28], ki popolnoma na novo zastavi upravljanje z osnovnimi gradniki. V večini obstoječih EDA orodjih prav upravljanje z osnovni gradniki (simboli, odtisi, 3D modeli) predstavlja večino dela. Na podlagi izkušenj iz vseh teh orodij predlagan način gradnike še nekoliko detajlno razbije (slika 5). Pri simbolih večina EDA okolji podpira simbole ki so sestavljeni iz več podsimbolov. Vendar pa podsimbolov ne moremo ponovno uporabiti pri različnih simbolih. V tem primeru pa so posamezni gradniki (simbol, podsimbol, nožica, ...) neodvisni. Tako lahko na primer različni simboli uporabijo isti podsimbol (e.g. 7400 in 742G00 uporabljata isti podsimbol za NAND vrata).



Slika 5: Primer gradnikov in povezav med njimi

Še bolj eksotična rešitev pa je projekt »SKiDL« [29]. Tu avtorji predlagajo zamenjavo za prvi korak načrtovanja vezja to je vnos sheme. Namesto grafične sheme predlagajo da se medsebojne povezave med elementi vpišejo v obliki kode. Prednosti takega vnosa so očitne: bistveno bistveno lažje pregledovanje sprememb med različnimi verzijami in lažja ponovna uporaba saj se pogosto uporabljene povezave (od najbolj enostavnih kot je na primer napetostni delilnik do kompleksnih kot je na primer večkanalni DC/DC pretvornik z vso potrebno periferijo) bistveno lažje modularizirajo.



Slika 6: Klasičen proces ravzoja tiskanega vezja (zgoraj) in proces kjer se povezave med elementi zapiše s kodo (spodaj)

5 Zaključek

Kot smo lahko videli razvoj EDA orodij poteka v več smereh, dejstvo pa je, da bodo procesi, ki so dobro znani iz področja razvoja programske opreme, prodrli tudi v proces načrtovanja električnih vezij.

Pričujoči članek zagotovo ni pokril vseh področji na katerih lahko pričakujem napredek pri razvoju EDA orodij vendar pa z omejenimi informacijami, ki so na voljo, bolj celovite slike ni možno vzpostaviti.

Zagotovo pa lahko trdimo, da ko eno izmed EDA orodij vpelje nov pristop, ostala EDA orodja prej ali slej vpeljejo primerljivo (ali enako) funkcionalnost.

Zahvala

Delo je bilo sofinancirano iz programa ARRS »Pretvorniki električne energije in regulirani pogoni« P2-0258 (B).

Literatura

[1] "Electronic design automation," *Wikipedia*. Mar. 23, 2020, Accessed: Jul. 20, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Electronic_design_automation&oldid=946896501.

[2] "KiCad Roadmap · GitLab," *GitLab*. https://gitlab.com/groups/kicad/-/roadmap (accessed Jul. 20, 2020).

[3] "KiCon 2019 Talks," *KiCon 2019*. https://kicad-kicon.com/talks/ (accessed Jul. 20, 2020).

[4] T. Berling and T. Thelin, "A case study of reading techniques in a software company," in *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04.*, Aug. 2004, pp. 229–238, doi: 10.1109/ISESE.2004.1334910.

[5] K. E. Wiegers, *Peer reviews in software: A practical guide*. Addison-Wesley Boston, 2002.

[6] A. F. Ackerman, L. S. Buchwald, and F. H. Lewski, "Software inspections: an effective verification process," *IEEE Softw.*, vol. 6, no. 3, pp. 31–36, May 1989, doi: 10.1109/52.28121.

[7] "MISRA C," *Wikipedia*. Jun. 25, 2020, Accessed: Jul. 20, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=MISRA_C&oldid=964441704.

[8] "C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems." SEI CERT, 2016.

[9] D. L. Jones, "PCB Design Tutorial." 2004.

[10] T. J. Sobering, "Guidelines for Drawing Schematics," p. 13, 2008.

[11] "KiCad Library Convention." https://kicad-pcb.org/libraries/klc/ (accessed Jul. 20, 2020).

[12] *uConfig*. Robotips, 2020.

[13] in3otd, *spiki*. 2020.

[14] *cadquery*. CadQuery, 2020.

[15] easyw, *kicad 3d models in freecad*. 2020.

[16] T. Lepoix, *Qucs-RFlayout*. 2020.

[17] Jānis, *pcbmodelgen*. 2020.

[18] B. P. Iyok, *svg2shenzhen*. 2020.

[19] "AllSpice: Hardware Development Tools for Electrical Engineers," *AllSpice Diff Tool*. https://www.allspice.io (accessed Jul. 20, 2020).

[20] "Eeshow." https://neo900.org/stuff/eeshow/ (accessed Jul. 20, 2020).

[21] John, *KiCad-Diff*. 2020.

[22] J.-N. Avila, *plotkicadsch*. https://github.com/jnavila/plotkicadsch, 2020.

[23] "CADLAB.io | Visual collaboration and version control platform for your PCB." https://cadlab.io/ (accessed Jul. 20, 2020).

[24] "InventHub." https://inventhub.io/ (accessed Jul. 20, 2020).

[25] "ECAD/MCAD Collaboration with IDX Standard | prostep ivip." https://www.ecad-mcad.org/ (accessed Jul. 20, 2020).

[26] easyw, *kicadStepUpMod*. 2020.

[27] "CI/CD," *Wikipedia*. Jul. 13, 2020, Accessed: Jul. 20, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=CI/CD&oldid=967493732.

[28] "horizon-eda.org." https://horizon-eda.org/ (accessed Jul. 20, 2020).

[29] XESS Corporation, "SKiDL," *SKiDL*. https://xesscorp.github.io/skidl/docs/_site/ (accessed Jul. 20, 2020).