Poenostavljen strojno-opisni jezik

Andrej Trost, Andrej Žemva

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

Simplified hardware-description language

In digital circuit design education we use standard hardware description language VHDL for building circuit models and prototyping implementation with programmable devices. The students are learning basic modeling concepts, the language syntax and tool design flow at the same time. We propose to use a simplified hardware-description language when introducing digital design to electrical engineering students. In the paper we present the proposed language with our web design tool and experience of including the language to the laboratory practice of several digital design courses.

1 Uvod

Strojno-opisne jezike HDL (*angl. Hardware Description Languages*) uporabljamo za računalniško načrtovanje sodobnih digitalnih vezij. Digitalna elektronska vezja sestavljajo logična vrata in pomnilni elementi (npr. registri), ki jih opišemo z elektronsko shemo ali Boolovimi enačbami. Sodobna digitalna vezja in sistemi so tako kompleksni, da je shematsko načrtovanje neučinkovito. Model vezja v jeziku HDL je narejen na bolj abstraktni ravni in omogoča hitrejši postopek razvoja. Najbolj razširjena in standardizirana jezika za opis digitalnih vezij na ravni registrov sta VHDL in Verilog [1]. V raziskav-i [2] navajajo uporabo HDL pri več kot polovici študijskih predmetov s področja digitalnih vezij. Prototip načrtanega digitalnega vezja naredimo s programirljivimi vezji FPGA z orodji proizvajalcev (npr. Intel Quartus, Xilinx Vivado), ki podpirajo standardne strojno-opisne jezike.

Ko uvajamo študente v praktično delo s strojno-opisnim jezikom in računalniškimi orodji za načrtovanje vezij je glavni izziv obvladovanje zahtevnosti. Pri laboratorijskih vajah izgubimo precej časa s slovničnimi pravili jezika in z učenjem uporabe pripadajočih orodij. Na področju načrtovanja digitalnih vezij imamo precej manjšo izbiro jezikov in prosto dostopnih orodij, kot na področju programiranja. Na Fakulteti za elektrotehniko uporabljamo v študijskem procesu predvsem strojno-opisni jezik VHDL. Aktualna standardna različica jezika je IEEE 1076-2008 [3] ali na kratko VHDL 2008, ki pa še ni v celoti podprta v razvojnih orodjih [4]. Najbolj razširjena je različica VHDL'93 (IEEE 1076-1993). Slovnica jezika VHDL je nastala po vzoru programskega jezika ADA in je precej drugačna od slovnic modernih programskih jezikov. Študenti elektrotehnike, ki niso vešči modeliranja v različnih jezikih, imajo zato težave pri usvajanju slovnice jezika VHDL.

Strojno-opisni jezik Verilog je bolj podoben programskemu jeziku C, vendar še vedno zahteva poznavanje konceptov modeliranja digitalnega vezja s stavki, ki se pri simulaciji vezja izvajajo sočasno. Pretvorba algoritma v opis vezja zahteva drugačen pristop kot pretvorba v programsko opremo, zato na videz znana slovnična pravila jezika Verilog ne predstavljajo velike prednosti pri učenju načrtovanja vezij.

Zaradi naraščajoče kompleksnosti digitalnih sistemov prihaja do razkoraka med zahtevami in učinkovitostjo modeliranja v jeziku HDL, ki ga rešujemo z modeliranjem na še bolj abstraktni ravni. Orodja za visokonivojsko sintezo vezja (*angl. High-Level Synthesis*) omogočajo pretvorbo algoritma v obliki programske kode v vezje [5]. Uporaba teh orodij zahteva razumevanje postopka modeliranja in sinteze vezij na ravni registrov, kjer pa je najbolj smiselno uporabiti enega izmed standardnih jezikov.

V prispevku bomo opisali poenostavljen strojno-opisni jezik za hitro uvajanje študentov v modeliranje digitalnih vezij. Razvili smo orodje [6], ki omogoča opis in simulacijo vezij v spletnem brskalniku ter avtomatsko pretvorbo v jezik VHDL. V naslednjem poglavju predstavljamo glavne izzive pri poučevanju načrtovanja digitalnih vezij in razloge za uvedbo poenostavljenega jezika. V tretjem poglavju opisujemo delo s poenostavljenim jezikom, v četrtem pa izkušnje uporabe spletnega orodja na laboratorijskih vajah.

2 Poučevanje načrtovanja digitalnih vezij

Načrtovanje digitalnega vezja je običajno le ena izmed tematik v študijskem predmetu s področja digitalne tehnike. Pri osnovnih predmetih v nižjih letnikih se ukvarjamo z logičnimi vrati in gradniki, njihovo analizo ter načrtovanjem preprostih vezij. Načrtovanje oz. sinteza vezja spada med zahtevnejše kognitivne učne cilje. Študenti se praktično seznanijo s tehnikami načrtovanja na laboratorijskih vajah. Vaje s področja digitalnih vezij razdelimo glede na način dela na:

1. spoznavanje orodij in načina opisa vezja
2. sistematično načrtovanje določene vrste vezja
3. reševanje sestavljenih nalog oz. projektov

Na laboratorijskih vajah uporabljamo vse tri načine dela. V nižjih letnikih je večji poudarek na spoznavanju poteka načrtovanja vezja in orodij. Posamezne vrste vezij, kot so seštevalniki, dekodirniki in sekvenčni stroji načrtujemo sistematično, po postopku, ki ga razložimo na predavanjih. Za zaključek vaj pa rešujemo sestavljeno nalogo. V višjih letnikih na vajah utrdimo

predhodno znanje in od študentov pričakujemo samostojno izvedbo projekta.

2.1 Opis modela vezja

Praktično spoznavanje računalniškega načrtovanja vezij v jeziku HDL je vključeno v uvodni sklop vaj. Modelirni jezik opisuje digitalno vezje v treh oblikah: strukturni, opisu pretoka podatkov in opisu obnašanja. Strukturni opis je najbolj podoben risanju sheme. Določiti moramo medsebojne povezave že pripravljenih gradnikov oz. komponent vezja. Shematske povezave modeliramo s signali, ki so lahko eno-bitni ali pa več-bitni vektorji.

Opis pretoka podatkov modelira funkcije gradnikov vezja, ki predstavljajo kombinacijsko logiko. Funkcije opišemo z operatorji v prireditvenih stavkih. Pri tem načinu modeliranja uporabljamo sočasne stavke, katerih vrstni red ni pomemben. Z aritmetičnimi in logičnimi operatorji nad vektorskimi signali opišemo kombinacijsko logiko na zelo kompakten način.

Opis obnašanja vezja je nabolj abstraktna raven opisa vezja v jeziku HDL. Delovanje vezja opisuje stavčni blok pri katerem je vrstni red zapisa pomemben in omogoča uporabo vejitev (npr. pogojne staveke). Če v modelu vezja upoštevamo določene omejitve, lahko programska oprema iz opisa obnašanja vezja avtomatsko sintetizira vezje na ravni registrov. Opis obnašanja vezja omogoča, da se načrtovalec vezja ukvarja z algoritmom in ne s strukturami za logično izvedbo algoritma.

2.2 Model vezja v jeziku VHDL

Strojno-opisni jezik VHDL se je najprej razvil kot jezik za modeliranje in simulacijo splošnih digitalnih vezij. Kasneje so naredili orodja za avtomatsko sintezo vezja, pri katerih smo omejeni na del sintakse jezika. Nekateri konstrukti (npr. zakasnitve, izpisi, datoteke) so namenjeni le za simulacijo. Sintaksa, ki je primerna za sintezo vezja, obsega vse tri oblike modeliranja. Opis vezja na grobo razdelimo na:

- deklaracije knjižnic,
- opis vmesnika in
- opis delovanja vezja, ki vsebuje:
  - deklaracije signalov,
  - opis povezav komponent,
  - sočasne stavke in
  - procesne stavke

Opis povezav komponent (*angl. port map*) predstavlja strukturni opis. Sočasni stavki modelirajo kombinacijske komponente vezja, procesni stavki pa obsegajo sekvenčne in opis obnašanja vezja. Jezik VHDL ima dolgovezno (*angl. verbose*) slovnico, kjer je potrebno poznati veliko ključnih besed in pravil za opis modela vezja. Poučevanje modeliranja vezja v jeziku VHDL temelji na vzorcih oz. primerih opisa osnovnih kombinacijskih in sekvenčnih vezij. Slika 1 prikazuje opis podatkovnega flip-flopa.

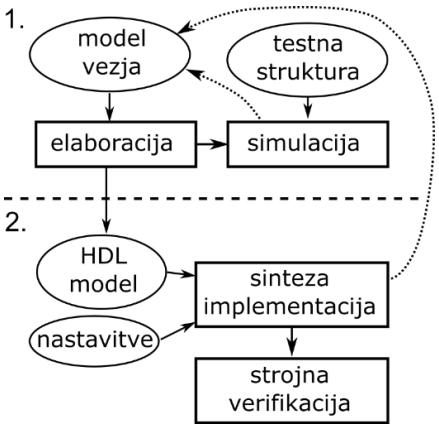
```
1  library IEEE;                                          knjižnice
2  use IEEE.std_logic_1164.all;
3
4  entity flipflop is
5   port (                                                vmesnik
6     clk, d : in std_logic;
7     q : out std_logic );
8  end flipflop;
9
10 architecture RTL of flipflop is
11 begin
12  process(clk)                                          procesni
13  begin                                                 stavek
14   if rising_edge(clk) then
15    q <= d;
16   end if;
17  end process;
18 end RTL;
```

Slika 1: Opis flip-flopa v jeziku VHDL.

Delovanje najbolj osnovnih vezij določimo v le nekaj stavkih, ki pa potrebujejo dolgovezen opis vmesnika in knjižnic. V prireditvenih stavkih veljajo striktna pravila glede ujemanja podatkovnih tipov signalov in konstant. Lastnosti jezika VHDL povzročajo težave neizkušenim študentom pri reševanju nalog na laboratorijskih vajah. Z uvedbo poenostavljenega strojno-opisnega jezika želimo izboljšati praktično izkušnjo modeliranja vezij.

2.3 Računalniška orodja za načrtovanje vezij

Slika 2 prikazuje tipičen potek načrtovanja in prototipne izvedbe digitalnih vezij, ki ga razdelimo na dva dela: modeliranje in simulacijo modela ter tehnološko preslikavo in verifikacijo.



Slika 2: Načrtovanje digitalnih vezij z računalniškimi orodji za modeliranje in simulacijo (1.) in tehnološko preslikavo (2.).

Model vezja načrtujemo v obliki sheme ali strojno-opisnega jezika in ga preverjamo s simulacijo. Računalniško orodje najprej naredi elaboracijo modela in javi morebitne sintaktične napake. Za pripravo simulacije potrebujemo testno strukturo, ki določa potek vhodnih signalov. V tem delu je zelo pomemben uporabniški vmesnik orodij, ki lahko precej olajša učenje in pospeši proces snovanja vezij.

Tehnološko preslikavo izvajajo orodja proizvajalcev programirljivih vezij. Model vezja v strojno-opisnem jeziku in uporabniške nastavitve so vhodni podatki za

avtomatsko sintezo in implementacijo. Implementirano vezje naložimo na razvojno ploščo s programirljivim vezjem, na kateri izvajamo strojno preizkušanje oz. strojno verifikacijo.

Potek načrtovanja zahteva poznavanje različnih orodij, npr. simulatorja ModelSim v prvem delu in orodja Quartus za končno implementacijo. Od proizvajalcev vezij FPGA dobimo brezplačne različice orodij, ki so dokaj zahtevne za delo, zato se le posamezni študenti odločajo za njihovo uporabo na domačem računalniku. Odprtih orodij je malo; za jezik VHDL je na voljo le simulator GHDL [7]. Za izvedbo simulacije v GHDL potrebujemo testno strukturo, ki zahteva še dodatno znanje strojno-opisnega jezika in poseben program za pregled rezultatov simulacije. Z nekaj znanja programiranja lahko naredimo interaktivno simulacijo [8], ni pa na voljo uporabniku prijaznih orodij.

Na Fakulteti za elektrotehniko smo razvili spletno orodje v katerem razvijamo vezje v poenostavljenem strojno-opisnem jeziku, ki modelira osnovne koncepte jezika VHDL [9]. Orodje omogoča preprosto izvedbo simulacije z grafičnim uporabniškim vmesnikom. Model vezja in nastavitve simulatorja se avtomatsko prevedejo v lepo oblikovano kodo jezika VHDL.

3 Poenostavljen strojno-opisni jezik

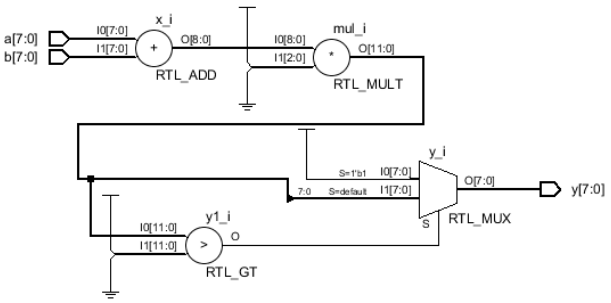
Poenostavitve modelirnega jezika smo pripravili na podlagi izkušenj uporabe jezika VHDL v pedagoškem procesu. Na vajah uporabljamo le del konstruktov jezika in omejen nabor podatkovnih tipov, pri dodatnih poenostavitvah jezika pa želimo skrajšati dolgovezno slovnico. Poenostavitve obsegajo:

- deklaracije vmesnika, arhitekture in knjižnic,
- deklaracije signalov in numeričnih vektorjev,
- intuitivna pravila za prireditvene stavke,
- privzeto pretvarjanje podatkovnih tipov in
- opis obnašanja vezja z različnimi vrstami prireditev in pogojnimi stavki namesto procesov.

Poenostavljen strojno-opisni jezik bomo predstavili na modelu majhnega kombinacijskega vezja z dvema 8-bitnima vhodoma in enim izhodom. Vezje naredi vsoto in skaliranje vhodnih podatkov s konstanto. Izhod naj gre v nasičenje, če je izračunan rezultat izven območja 8-bitnih vrednosti. Funkcionalnost vezja opisuje psevdokoda:

```
mul = (a + b)*5;
if (mul>255) mul = 255;
```

Shematski opis vezja sestavljajo kombinacijski seštevalnik, množilnik, primerjalnik in izbiralnik, kot prikazuje slika 3. Izhod seštevalnika je 9-bitno (8:0), množilnika pa 12-bitno vodilo. Širino vodila določimo glede na pričakovan razpon vrednosti, ki se izračunajo v posameznih operacijah. Če bi uporabili premalo bitov, bi dobili napačne rezultate zaradi preliva, pri prevelikem številu bitov pa imamo preveliko porabo kombinacijske logike.



Slika 3: Shema kombinacijskega vezja za vsoto in skaliranje podatkov z nasičenjem.

Vodila modeliramo s signali, ki jim določimo podatkovni tip. V jeziku VHDL najpogosteje uporabljamo eno-bitne signale deklarirane kot *std_logic* vrednosti ali več-bitne vektorje tipa *std_logic_vector*. Računske operacije s celimi števili omogoča vektorski podatkovni tip *unsigned* ali *signed*.

Podatkovni tip signalov v poenostavljenem jeziku določa črka **u** (unsigned) ali **s** (signed) in število bitov, ki je med 1 in 64. Priključke in signale vezja deklariramo v tabeli, kot prikazuje slika 4. Vhodnim priključkom določimo način **in**, izhodnim pa **out**.

Name	Mode	Type
a, b	in	u8
mul		u12
y	out	u8
x		u9

```
x = a + b
mul = x * 5
if (mul>255) y = 255
else y = mul
```

Slika 4: Definiranje signalov in opis kombinacijskega vezja v poenostavljenem strojno-opisnem jeziku.

Opis vezja v poenostavljenem jeziku vsebuje le prireditve in pogojne stavke. Izvajajo se paralelno (sočasno), ker predstavljajo opis delov vezja. Za opis vezja moramo poznati le nekaj osnovnih pravil.

- Blok stavkov je zaporedje stavkov, ki so vsak v svoji vrstici ali pa ločeni s podpičjem.
- Za pogojem stavka **if** oz. za delom **else** sledi en stavek v isti vrstici ali pa blok stavkov med zavitima oklepajema, npr. { stavek1; stavek2; stavek3 }.
- Znotraj bloka lahko le enkrat priredimo vrednost posameznemu signalu.
- Vrstni red prireditev različnim signalom znotraj bloka ni pomemben.

Tabela 1: Orodja in jeziki za načrtovanje digitalnih vezij na laboratorijskih vajah.

Predmet	**Stopnja**	**Orodja**	**Opis vezja**
Programirljivi digitalni sistemi	BVS, 1 letnik	Intel Quartus in **spletno orodje**	shema diagram stanj **poenostavljen jezik**
Načrtovanje digitalnih elektronskih sistemov	BVS, 3 letnik, izbirni	Intel Quartus in **spletno orodje**	**poenostavljen jezik** VHDL
Digitalni elektronski sistemi	BUN, 2 letnik	Xilinx Vivado ali Intel Quartus **spletno orodje**	**poenostavljen jezik** VHDL
Preizkušanje elektronskih vezij	BMAG, 1 letnik, izbirni	Modelsim	Verilog
Digitalna integrirana vezja in sistemi	BMAG, 2 letnik	Xilinx Vivado in SDK	blokovna shema z IP jedri VHDL ali Verilog in C

Spletno orodje za načrtovanje vezij pretvori poenostavljen opis vezja v lepo oblikovan model v jeziku VHDL, kot prikazuje slika 5. Orodje poskrbi za ustrezno sintakso prevedenih stavkov z upoštevanjem pravil pretvarjanja podatkovnih tipov in velikosti vodil. Koda poenostavljenega jezika je precej krajša od opisa v jeziku VHDL – v povprečju ima za učne primere vezij štirikrat manj vrstic.

```
11  architecture RTL of sat is
12    signal x : unsigned(8 downto 0);
13    signal mul : unsigned(11 downto 0);
14  begin
15   x <= resize(a,9) + resize(b,9);
16   mul <= x * to_unsigned(5, 3);
17
18  process(all)
19   begin
20     if mul > 255 then
21        y <= to_unsigned(255, 8);
22     else
23        y <= resize(mul,8);
24     end if;
25  end process;
26
27  end RTL;
```

Slika 5: Avtomatsko generiran opis vezja v jeziku VHDL.

4 Načrtovanje vezij na laboratorijskih vajah

Spletno orodje s poenostavljenim strojno opisnim jezikom smo vključili na laboratorijske vaje pri treh predmetih, kot prikazuje tabela 1. V prvem letniku Bolonjskega visokošolskega študija (BVS) smo prej uporabljali le shematsko načrtovanje in risanje diagrama stanj, ker bi bilo uvajanje sintakse jezika VHDL prezahtevno. Poenostavljen jezik nam sedaj omogoča pripravo nekaj vaj in uvajanje v modeliranje HDL.

V tretjem letniku imamo izbirni predmet, pri katerem načrtujemo digitalni sistem od osnovnih vezij do videoigre [10]. Zaradi izbirnosti dobimo študente z zelo različnim predznanjem modeliranja vezij. S poenostavljenim jezikom naredimo v laboratoriju več primerov in hitreje uvedemo študente, ki se s strojno-opisnim jezikom prvič srečujejo. Spletno orodje spodbuja samostojno delo in vajo na domačem računalniku. Ob koncu vaj smo opravili kratko anketo in ugotovili:

- polovici študentov dela preglavice zahtevna sintaksa jezika VHDL,
- v poenostavljenem jeziku se bolj ukvarjajo s pomenom kode in manj s sintakso,
- s spletnim orodjem so uspeli rešiti več nalog, kot bi jih samo z jezikom VHDL.

Kot pomanjkljivosti novega pristopa so študenti izpostavili: nekaj napak v izhodni kodi, premalo uporabnih stavkov in prevelika razlika v primerjavi z jezikom VHDL.

Pri predmetu v drugem letniku univerzitetnega študija obravnavamo na predavanjih VHDL vzporedno s poenostavljenim jezikom. Na laboratorijskih vajah uporabljamo jezik in spletno orodje za učenje primerov vezij ter razvoj komponent sestavljene naloge. Včasih smo veliko časa porabili za razlago sintakse jezika in odpravljanje napak v kodi. Pri izdelavi modela vezja v poenostavljenem jeziku pa je večji poudarek na osnovnih konceptih modeliranja digitalnega vezja.

5 Zaključek

Računalniška orodja se razvijajo skupaj s tehnologijo in jeziki za načrtovanje vezij so kompleksni, ker so narejeni za splošen opis vezij in testnih struktur. Zahtevna sintaksa jezika je posledica razvoja jezika, ki mora biti združljiv s prvotnim standardom.

Predstavili smo poenostavljen strojno-opisni jezik, ki ga uporabljamo pri delu s študenti. Poenostavitev omogoča, da se več ukvarjamo s koncepti modeliranja kot pa s sintakso jezika. Sintaksa se spreminja, koncepti modeliranja na nivoju registrov pa so univerzalni in so eden izmed učnih ciljev praktičnega laboratorijskega dela.

Literatura

[1] Steve Golson and Leah Clark. Language Wars in the 21st Century: Verilog versus VHDL – Revisited. In Synopsys Users Group (SNUG), 2016

[2] H. A. Ochoa, M. V. Shirvaikar, A Survey of Digital Systems Curriculum and Pedagogy in Electrical and Computer Engineering Programs, ASEE Gulf-Southwest Section Annual Conference, Austin, 2018

[3] IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-2008, IEEE Computer Society, New York, 2009

[4] Vivado Design Suite User Guide - Synthesis, Xilinx, UG901, junij 2019

[5] W. Meeus, K. Van Beeck, T. Goedeme, J. Meel, D. Stroobandt, An overview of today's high-level synthesis tools, Design Automation for Embedded Systems, 16(3), 2012, str. 31-51

[6] A. Trost, A. Žemva, Online VHDL Generator and Analysis Tool, Mediterranean Conference on Embedded Computing MECO 2019, str. 238-241

[7] T. Gringold, GHDL, 2017, http://ghdl.free.fr/

[8] M. Strubel, Using GHDL for interactive simulation under Linux, FPGA related, 2011 https://www.fpgarelated.com/showarticle/20.php

[9] SHDL, https://lniv.fe.uni-lj.si/shdl/

[10] A. Trost, A. Žemva, Načrtovanje digitalnih sistemov: od števca do videoigre, Zbornik Elektrotehniške in računalniške konference ERK 2014, Portorož, 2014, str. 173-176