Načrtovanje digitalnih sistemov: od števca do videoigre

Andrej Trost, Andrej Žemva

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

Digital Systems Design: from Counter to Video Game

Engineers working in digital electronics industry require design skills beyond basic logic and programing theory. The students of electronics should understand the operation and interaction of the digital system components and be able to develop a complete digital system on the circuit and software level.

In the paper we present laboratory practice for a Digital electronic systems design course covering from design of a simple counter to a complete video game. We are using programmable devices FPGA for prototyping implementation of the system. The system consists of a custom CPU core and logic components described in VHDL language and tested on the prototyping boards. In the midterm the students present their own version of a CPU. The final project is a videogame utilizing the designed CPU and a custom video hardware. We provide web based tools for remote testing and software development in order to enable the students finalizing their assignments at home.

1 Uvod

Razvoj digitalnih elektronskih sistemov zahteva poznavanje širokega področja elektrotehnike, od elektronskih vezij, digitalnih komponent do programiranja procesorjev. Med študijem vsa ta področja pokrivajo specifični predmeti in od študentov se pričakuje sinteza vsega specifičnega znanja za uspešen prenos v prakso. V članku predstavljamo izvedbo laboratorijskih vaj in projekta, pri katerem dobi študent vpogled v celoten proces načrtovanja digitalnih sistemov.

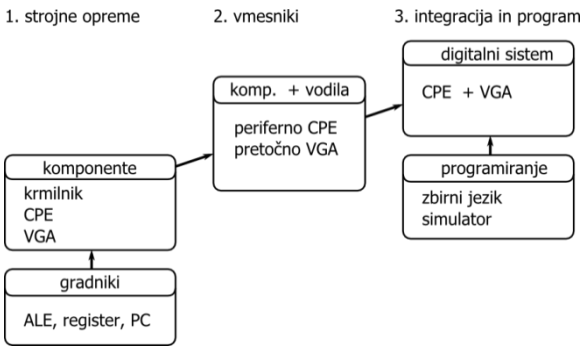
Podoben koncept so predstavili na področju računalništva s predmetom, ki obravnava zgradbo strojne opreme, izdelavo prevajalnikov in preprostega operacijskega sistema ter aplikacije [1, 2]. Predmet podaja poenostavljeno teorijo pomembnejših konceptov iz računalništva. Vaje potekajo na simulacijskih orodjih, ki omogočajo eksperimentiranje na zelo različnih nivojih abstrakcije.

Z vidika elektrotehnike so digitalni elektronski sistemi obravnavani kot nadgradnja digitalnih struktur ali pa kot specifični mikroprocesorski oz. vgrajeni sistemi. Za praktično učenje razvoja digitalnih sistemov uporabljamo mikroprocesorske in programirljive razvojne sisteme ter simulacijska orodja [3]. S programirljivimi vezji v tehnologiji FPGA lahko naredimo laboratorijske vaje, ki vključujejo tako osnovne digitalne strukture kot tudi celotne sisteme na integriranem vezju [4]. Razvoj lastnega procesorja v programirljivem vezju omogoča eksperimentiranje z naprednimi tehnikami načrtovanja digitalnih vezij [5]. Programirljivi razvojni sistemi se zato pogosto uporabljajo pri poučevanju digitalnih struktur in pripravi laboratorijskih vaj [6].

V prispevku bomo predstavili zasnovo in učne cilje laboratorijskih vaj na visokošolskem študiju, ki smo jih poimenovali: "Načrtovanje digitalnih sistemov: od števca do videoigre" [7]. Opisali bomo vsebino posameznih vaj, rezultate študentskih projektov ter izkušnje in ugotovitve pri izvedbi vaj v zadnjih letih.

2 Načrtovanje digitalnih sistemov

Cilj laboratorijskega projekta je izdelava digitalnega sistema v obliki videoigre. Preprosto videoigro je mogoče opisati kot kompleksno digitalno vezje na nivoju registrov v jeziku VHDL in preizkusiti na razvojnem sistemu z vezjem FPGA. Prvi korak je generator videosignala (npr. za VGA signal), nato dodamo prikaz sličic in algoritem igre. Takšna zasnova sistema ni optimalna, saj potrebujemo za generator video signala hitro in enostavno logiko, za krmiljenje igre pa kompleksen, vendar počasnejši algoritem. Če načrtujemo le na nivoju registrov in logike, bo hitro zmanjkalo virov in bo rešitev daleč od optimalne. Boljši sistem naredimo z uporabo procesorja in namenske strojne opreme, kar pa zahteva drugačen proces načrtovanja (slika 1).



Slika 1: Proces načrtovanja digitalne videoigre

Učni cilj laboratorijskih vaj je praktično delo, ki povezuje osnovna področja izdelave digitalnih sistemov:

- razvoj logičnih struktur in komponent,
- razvoj vmesnikov, ki povezujejo komponente,
- integracija komponent v strojno opremo sistema in
- izdelava programske opreme.

Obravnavana področja so zelo obsežna, zato je potrebno nivo obravnave prilagoditi, da ne bo prezahteven ali preveč poenostavljen. Od študentov pričakujemo predznanje logičnih gradnikov in osnov jezika VHDL. Za izbiro procesorja imamo na voljo strojno ali programsko procesorsko jedro. Strojna jedra so bolj zmogljiva, vendar so vezana na konkretno družino integriranih vezij, zato smo se odločili za centralno procesno enoto (CPE) narejeno v programirljivi logiki. Izbrali smo preprosto 12-bitno CPE [5], ki jo naredimo v sklopu prvega dela laboratorijskih vaj. Testiranje CPE poteka s kratko strojno kodo zapisano s konstantami v jeziku VHDL, za lažji razvoj aplikacije pa smo pripravili spletno stran z zbirniškim prevajalnikom in simulatorjem. Študenti imajo na voljo tudi strežnik za testiranje CPE na strojni opremi iz domačega računalnika [8]. Razvoj in nadgradnja osnovne CPE je prvi mejnik v laboratorijskih vajah. Študenti napišejo poročilo (miniprojekt), na osnovi katerega dobimo prvo oceno o tem, kako sledijo poteku vaj in po potrebi naredimo dodatno razlago ter prilagodimo zahtevnost končnega projekta.

Izdelava videoigre zahteva razvoj komponent za prikaz gibljivih slik na računalniškem monitorju in povezavo s CPE. Na tem mestu obravnavamo različne vmesnike in izberemo ustrezno vodilo glede na zahteve prenosa podatkov. V okviru skupnih vaj naredimo celoten sistem za premikanje ene gibljive sličice, nato pa mora vsak narediti in zagovarjati projekt v katerega je vključenih več sličic in kompleksnejši nadzor.

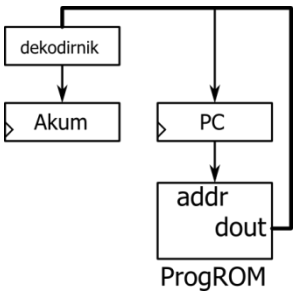
3 Vsebina laboratorijskih vaj

Teme posameznih laboratorijskih vaj podaja tabela 1. Na začetku obravnavamo osnovne operacije nad vektorskimi signali in naredimo model aritmetično logične enote (ALE). Druga vaja obravnava osnovna sekvenčna vezja – pomikalni register in programski števec. V prvih vajah posvetimo nekaj časa spoznavanju razvojnih orodij, načina opisa in simulacije vezja.

Pri tretji vaji naredimo mikroprogramirano krmilno vezje na nivoju registrov (RTL, angl. Register Transfer Level). Vezje je sestavljeno iz programskega pomnilnika s programskim števcem, dekodirnika in akumulatorja, kot prikazuje slika 2. Vezje pozna dve vrsti mikroukazov: nalaganje vrednosti v akumulator in skočni ukaz, ki naloži novo vrednost v programski števec. Vezje preizkusimo na razvojnih ploščah, tako da vežemo izhod akumulatorja na LED, uro pa pripeljemo preko delilnika frekvence. Zaporedje vrednosti v akumulatorju se spreminja v skladu s programom. Mikroprogramiran krmilnik predstavlja zasnovo CPE.

Tabela 1: potek laboratorijskih vaj

Vaja	teorija	naloga
1.	operacije	ALE
2.	sekvenčna vezja	pomikalni register, števec
3.	vezja RTL	mikroprogramiran krmilnik
4.	model CPE	CPE z osnovnimi ukazi in testnim programom
5.	vodilo, struktura	nadgradnja CPE z V/I
6.	miniprojekt	CPE z lastnimi ukazi
7.	standard VGA	prikaz slike na monitorju
8.	točkovna grafika	prikaz grafike
9.	struktura sistema	grafični sistem s CPE
10.	projekt	izdelava videoigre



Slika 2: Zgradba mikroprogramiranega krmilnika

3.1 Razvoj namenskega procesorja

Mikroprocesor za učne namene vsebuje le en podatkovni register (akumulator) in majhen nabor ukazov, ki se izvedejo v dveh ciklih. V prvem ciklu iz pomnilnika zajamemo ukaz in absolutni naslov operanda, v drugem pa izvedemo ukaz.

Začetni model CPE pozna le šest osnovnih ukazov:

- **lda** naloži operand iz pomnilnika v akumulator,
- **sta** prenese vrednost akumulatorja v pomnilnik,
- **add** prišteje operand k akumulatorju,
- **sub** odšteje operand od akumulatorja,
- **jmp** naredi skok na absolutni naslov in
- **jze** naredi skok kadar je v akumulatorju vrednost nič.

Opis CPE je nadgradnja mikroprogramiranega krmilnika, kateremu dodamo ALE z zastavicami, register za programsko kodo, logiko za določanje pomnilniškega naslova in krmilnika, kot prikazuje slika 3. Krmilnik določa stanje CPE: *zajemi* ali *izvedi*.

V stanju *zajemi* določa naslov pomnilnika programski števec, ukaz pa se prenese v register za programsko kodo. V stanju *izvedi* določa naslov pomnilnika operand in ukaz se izvede. Vsi ukazi so 12-bitni: zgornji 4 biti določajo kodo ukaza, spodnjih 8 pa

naslov operanda. Pomnilnik vsebuje največ 256 ukazov in podatkov, kar zadošča za učni namen.

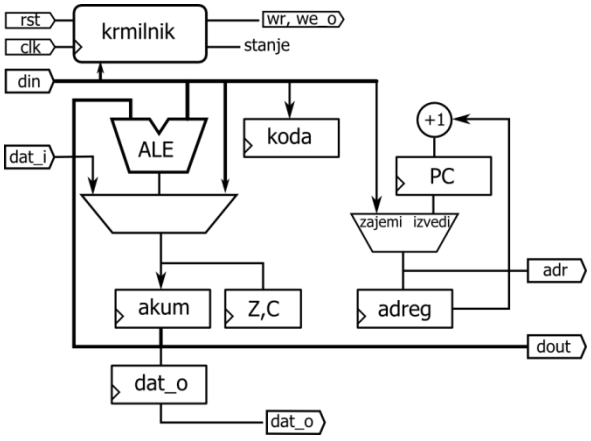
V jeziku VHDL definiramo ukaze kot 4-bitne simbolične konstante. Konstante uporabljamo v opisu procesorja in programa v pomnilniku. Prvi simulacijski preizkus naredimo s kratkim testnim programom, ki naloži akumulator in v zanki odšteva vrednost do 0, potem pa vse skupaj ponavlja:

```
signal ram : memory := (
  lda & x"05",
  sub & x"06",
  sta & x"07",
  jze & x"00",
  jmp & x"01",
  x"005",
  x"001",
  x"000"
);
```

Procesorju v naslednji vaji dodamo vhodno/izhodni (V/I) vmesnik in ukaze za prenos podatkov na zunanje vodilo. Slika 3 prikazuje zgradbo CPE z glavnimi enotami in zunanjimi signali:

- ura (clk) in reset (rst),
- povezave na pomnilnik (adr, din, dout, wr) in
- V/I vodilo (adr, dat_i, dat_o, we_o).

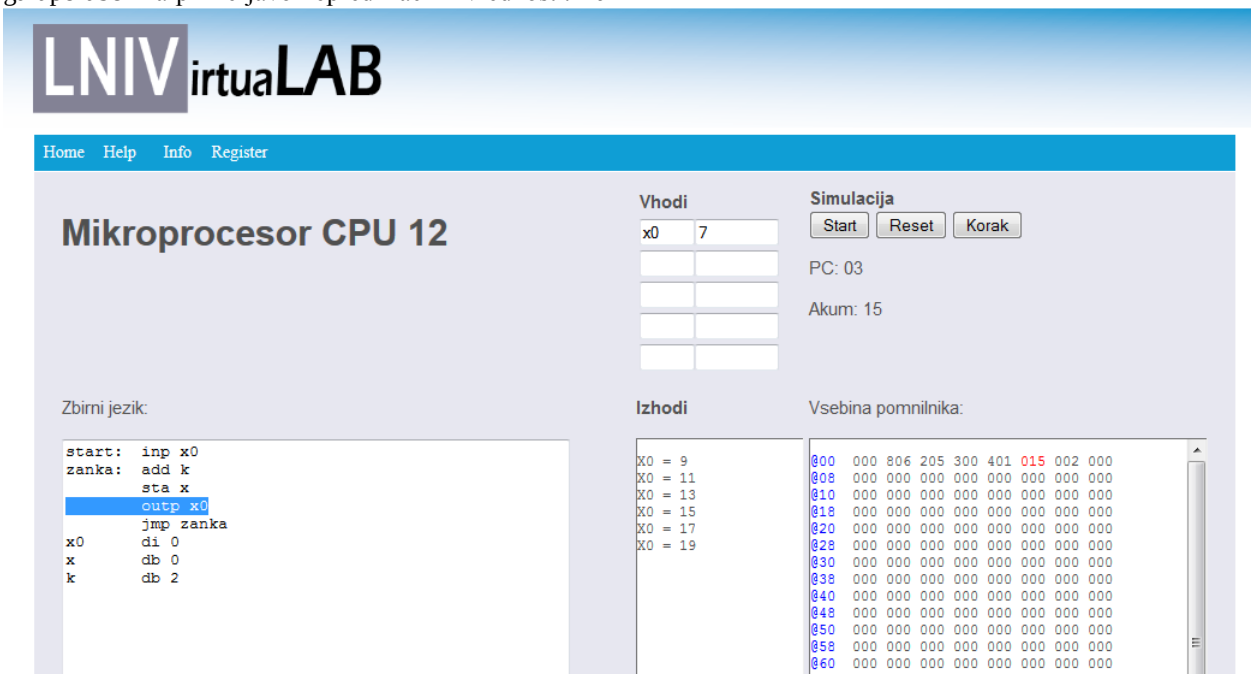
Na osnovi začetnega modela CPE študenti naredijo miniprojekt, v katerem nadgradijo procesor z novimi ukazi in enotami. Obstoječi aritmetično-logični enoti na primer dodajo logične operacije (**and**, **or**, **not**, **xor**) ali operacije pomika (**shl**, **shr**). Z dodatnim registrom, ki se imenuje skladovni kazalec, nadgradijo CPE z ukazi za delo s skladom (**push**, **pop**) ali s podprogrami (**call**, **ret**). Nabor pogojnih skokov razširijo z uporabo dodatnih zastavic na izhodu ALE. Z zastavico prenosa (C) izvedejo pogojni skok **jcs** (skok, kadar je C=1), ki ga uporaben za primerjavo nepredznačnih vrednosti. Za primerjavo predznačenih vrednosti pa je potrebno dodati še zastavici predznaka (N) in preliva (V).



Slika 3: Zgradba CPE z V/I vmesnikom

Pripravili smo spletni simulator za preizkušanje CPE v osnovni in nadgrajeni izvedbi. Simulator je napisan v jeziku Javascript in deluje v kateremkoli spletnem brskalniku. Simulator najprej prevede vhodno zbirniško kodo z oznakami za skočne ukaze in simboli pomnilniških in V/I spremenljivk v obliko z absolutnimi naslovi, ki jo lahko vstavimo v VHDL. Spremenljivke deklariramo z zbirniškimi direktivami: **db** – deklaracija pomnilniške besede in **dio** – deklaracije V/I besede.

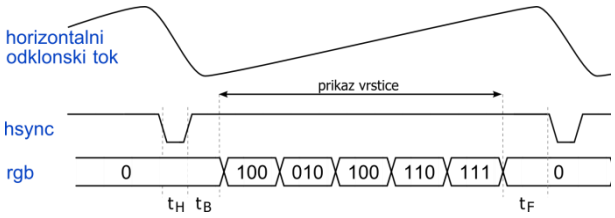
Ob resetu pokaže simulator vsebino pomnilnika in registrov, nato izvajamo simulacijo po korakih in sproti opazujemo rezultate. Med simulacijo vpisujemo v okenca vrednosti na vhodih V/I vmesnika, simulator pa prikazuje spreminjanje izhodov. S simulatorjem študentom olajšamo razumevanje delovanja CPE in omogočimo primerjavo delovanja s simulacijo njihovega vezja v jeziku VHDL.



Slika 4: Spletni simulator mikroprocesorja

3.2 Prikaz grafike

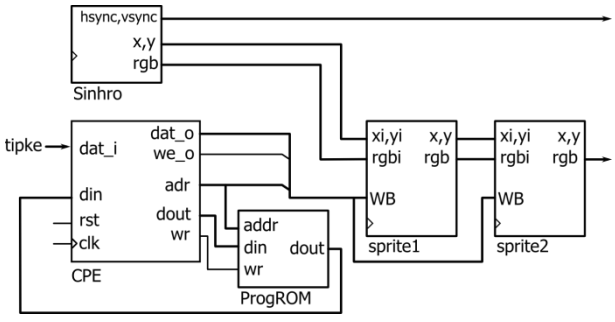
Prikazovanje grafike na računalniškem monitorju usklajuje sinhronizacijska komponenta, ki s števci generira koordinate in sinhronizacijske signale. Slika 5 prikazuje potek horizontalne sinhronizacije v času katere se izriše ena vrstica slike. Standard VGA določa frekvenco preleta vrstice in osveževanja slike ter vse časovne parametre sinhronizacijskih signalov (t_H, t_B, t_F).



Slika 5: Časovni potek signalov za izris vrstice na monitorju

Videoigro naredimo s pomočjo grafike sličic (angl. sprite). Sličice so definirane v pomnilniku vezja FPGA in določajo izhodno barvo (rgb), ko se prikazovana točka nahaja znotraj okvirja sličice. V okviru vaj naredimo komponente za prikaz sličic (sprite1, sprite2) in jih povežemo v digitalni sistem, kot prikazuje slika 6. Komponente imajo vodilo (WB) za povezavo z izhodnim vmesnikom CPE, tako da program v CPE določa parametre prikazovanja (npr. koordinate sličice na zaslonu).

Grafični sistem je zasnova tako, da ga je mogoče enostavno nadgraditi z več sličicami, logiko igre pa izvaja algoritem v procesorju. Vsak študent mora za končni projekt laboratorijskih vaj narediti in zagovarjati svojo izvedbo videoigre.



Slika 6: Zgradba vezja videoigre

4 Rezultati in zaključek

Predstavili smo zasnovo laboratorijskih vaj s področja digitalnih elektronskih sistemov. Na vajah ponudimo študentom zanimivo temo pri izdelavi končnega projekta in pregled nad procesom snovanja strojne in nizkonivojske programske opreme digitalnega sistema.

Za prototipno izvedbo videoigre potrebujemo razvojno ploščo z vezjem FPGA in izhodom za računalniški monitor, ki ga lahko naredimo tudi kot enostaven dodatni modul na poljubni razvojni plošči. Ker uporabljamo lastno CPE vsebina vaj ni odvisna od ciljne tehnologije in proizvajalca orodij. Vaje izvajamo že nekaj let in ugotavljamo, da izvedba videoigre po lastni zamisli motivira študente, ki na koncu pripravijo zahtevnejše projekte. Primeri narejenih projektov so:

- "Pacman", kjer premikamo figuro, ki jo lovi duhec,
- "Labirint", pri kateri iščemo izhod iz labirinta,
- strelska igra "Space Invaders" in
- "Pong", pri kateri odbijamo žogico.

Prvotno smo imeli naloge bolj natančno določene (npr. avtomat za parkirišče, serijski vmesnik, digitalna štoparica) in večina študentov se je odločala za lažjo nalogo. Vrsto nalog je možno reševati le s poznavanjem določenega koncepta ali abstrakcije digitalnega sistema (npr. kot končni avtomat). Na vajah skupaj zgradimo enostaven, vendar heterogen sistem, ki vsebuje sinhrona vezja, pomnilnik, procesor in vodila. S tem ponudimo študentom zasnovo, ki jo je mogoče nadgraditi na različnih mestih in utrditi poznavanje različnih konceptov.

V prihodnosti nameravamo uporabiti sistem za oddaljeno testiranje CPE na strojni opremi [8] in morda tudi oddaljeno testiranje narejenih videoiger.

Literatura

[1] S. Schocken, N. Nisan, M. Armoni, “A synthesis course in hardware architecture, compilers, and software engineering”, *SIGCSE Bull.* 41, 1 (March 2009), 443-447.

[2] From NAND to Tetris, *http://www.nand2tetris.org/*, julij 2014

[3] A. Trost, A. Žemva, “Teaching design of video processing circuits”, International Journal of Electrical Engineering Education, 49(2), str. 170-178, 2012

[4] Vainio, O.; Salminen, E.; Takala, J., "Teaching digital systems using a unified FPGA platform," Electronics Conference (BEC) 2010, str.137-140, Talin, 2010

[5] A. Trost, A. Žemva, Razvoj namenskih mikroprocesorjev za vezja FPGA, *Elektrotehniški vestnik*, 79(1-2), str. 55-60, 2012

[6] D. Hanna and R. E. Haskell, “Learning Digital Systems Design in VHDL by Example in a Junior Course,” Proceedings of the ASEE North Central Section Conference, Charleston, West Virginia, March 2007.

[7] Modul B: Integrirana vezja in Načrtovanje digitalnih elektronskih sistemov, *http://lniv.etris.org/*, julij 2014

[8] K. Saksida, A. Trost, Remote Laboratory for Testing Processor Cores in FPGA Device, MIPRO 2014, str. 178-183, Opatija, 2014