Načrtovanje strojne in programske opreme merilne plošče STEMlab

Andrej Trost, Andrej Žemva

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

Design of HW and SW components on STEMlab measurement board

STEMlab measurement board from Red Pitaya contains System-on-Chip (SoC) and fast analog inputs and outputs. It is used as programmable data acquisition and signal processing system for customized digital measurement and processing solutions. In the paper, we present our approach to design custom logic and applications for SoC on the STEMlab board. Development process requires substantial knowledge of digital design for programmable logic and low-level software applications. An approach to reuse existing open-source code and minimize design complexity is presented together with applications of the board in digital design education.

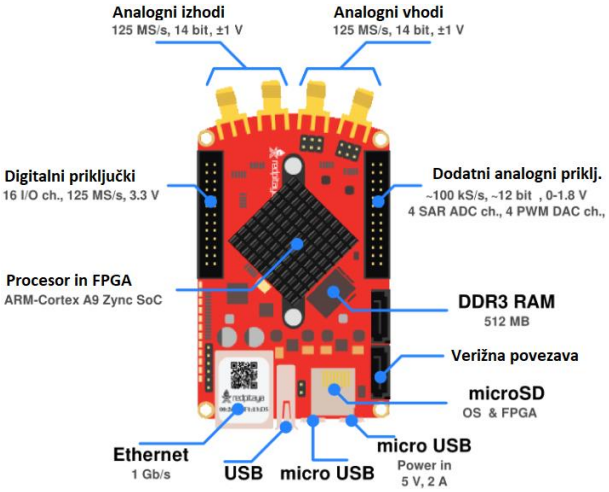
1 Uvod

Specializirane instrumente za merjenje in obdelavo elektronskih signalov lahko nadomestimo z napravami za zajem in računalniško obdelavo podatkov [1]. Proizvajalci programirljivih vezij so razvili sisteme na čipu SoC (ang. System-on-Chip), ki vključujejo zmogljive mikrokrmilnike in vezja FPGA (ang. Field Programmable Gate Array). Sistemi na čipu omogočajo v povezavi z analogno-digitalnimi (AD in DA) pretvorniki izdelavo razvojnih plošč za zajem in obdelavo hitrih signalov [2].

Merilna razvojna plošča STEMlab podjetja Red Pitaya vsebuje SoC iz družine Xilinx Zynq [3] z dvojedrnim procesorjem in zmogljivo logiko v vezju FPGA. Procesorski del deluje kot vgrajen računalnik na katerem se izvaja operacijski sistem Linux in spletni strežnik Nginx, ki omogoča nadzor aplikacij s spletnim brskalnikom. Za obdelavo hitrih analognih signalov ima dva vhodna in dva izhodna SMA priključka z AD oz. DA pretvorniki zmogljivosti 125 MS/s (slika 1). STEMlab ponuja aplikacije osciloskop, signalni generator, spektralni, omrežni in logični analizator, Bodejev analizator ter merilnik LCR. Dokumentacija vsebuje primere izvorne kode in navodila za izdelavo lastnih aplikacij [4]. S spremembo programirljive logike in programske opreme razvijemo namensko napravo za zajem, generiranje in obdelavo signalov. Obdelavo hitrih signalov izvajamo v vezju FPGA, procesor pa poskrbi za krmiljenje, naknadno obdelavo in uporabniški vmesnik.

Primer nadgradnje obstoječih aplikacij je večkanalni signalni generator, pri katerem izvajamo sestavljanje oz. modulacijo signala iz več digitalnih generatorjev v strojni opremi [5]. V logiki vezja FPGA učinkovito naredimo digitalna sita [6-7]. V literaturi najdemo tudi zahtevnejše aplikacije za STEMlab, npr. zaklenjen ojačevalnik (angl. Lock-in) [8] in pulzni radar osnovan na programsko določenem radiu (angl. Software Defined Radio) [9].

Poleg analognih ima STEMlab tudi digitalne priključke v povezavi s SoC in namenskimi aplikacijami. Prispevek [10] predstavlja merilnik zelo hitrih časovnih dogodkov TDC (ang. Time-to-Digital Converter) z ločljivostjo pod 20 ps.



Slika 1: Strojna oprema merilne plošče STEMlab.

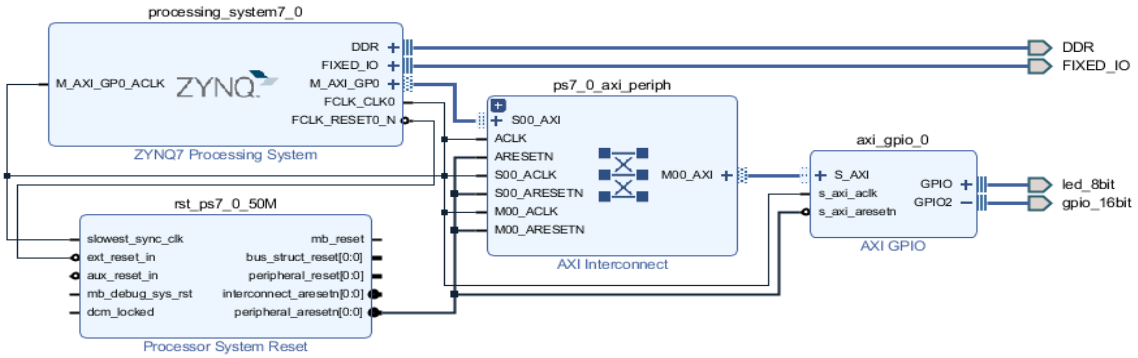
Merilna plošča STEMlab temelji na odprtokodni programski opremi in prosto dostopnih programih, zato je primerna za razvoj lastnih merilnih naprav. SoC zahtevajo dobro poznavanje načrtovanja strojne in programske opreme. Posamezne postopke je mogoče avtomatizirati, npr. v [11] opisujejo okvir za integracijo razvojnega procesa programske opreme razvojne plošče, ki vključuje prevajanje jedra sistema Linux.

V članku predstavljamo potek načrtovanja strojne in programske opreme merilne plošče STEMlab, ki je prilagojen za učne namene.

2 Razvoj digitalnega sistema

Računalniška orodja za razvoj namenske programske in strojne opreme na Red Pitayi razdelimo na štiri sklope:

- urejevalniki in uporabniški vmesniki za pisanje kode in modelov vezja (npr. Notepad++),
- orodja za prenos kode iz spleta (git) in komunikacijo z razvojno ploščo (npr. Bitwise SSH client),
- prevajalniki programske kode (gcc) in
- orodje za implementacijo programirljive logike (Vivado).



Slika 2: Blokovni načrt vezja za preizkus digitalnih priključkov.

Vsa orodja so na voljo v brezplačni različici. Nekatera so vključena v obstoječo programsko opremo, npr. gcc prevajalnik jezika C na razvojni plošči STEMlab, druga pa je potrebno namestiti. Najzahtevnejše je orodje za prevajanje logičnega vezja, ki je vezano na proizvajalca vezij.

Program Vivado je osnovno orodje za prevajanje načrtov logičnega vezja v naprave FPGA proizvajalca AMD-Xilinx. Družino SoC Zynq podpirajo vse različice orodja Vivado, vendar projekti med njimi niso vedno prenosljivi. Vezja v katerih imamo povezavo logike in procesorskega sistema uporabljajo knjižnične komponente, ki se med različicami programskega orodja razlikujejo. Na spletnem repozitoriju Red Pitaye [12] so shranjene skripte za Vivado 2020.1, naši projekti [13] pa uporabljajo tudi nekoliko starejši Vivado 2019.

2.1 Vezje za preizkus digitalne periferije

Ob izdelavi novega projekta v Vivadu določimo ciljno vezje ali razvojno ploščo. Če izberemo razvojno ploščo, imamo nastavljene električne lastnosti perifernih komponent, kar precej olajša začetni opis projekta. Datoteke plošče Red Pitaya se nahajajo v mapi `brd` na repozitoriju [12]. To mapo najprej prenesemo na lokalni računalnik, kjer je nameščen Vivado:

```
C:\Xilinx\Vivado\2019.1\data\boards\board_files\
```

Vivado vključuje shematski urejevalnik blokovnih načrtov (angl. Block Design) s katerim hitro naredimo testni projekt za preizkus LED in digitalnih priključkov. Na načrt vključimo poleg procesorskega dela (Zynq Processing System) še paralelni vmesnik AXI GPIO, ki mu nastavimo izbrane periferne enote, čarovniku programa pa prepustimo izdelavo vezne logike (AXI Interconnect), povezavo periferije s procesorjem in komponento za resetiranje vodila. Z urejevalnikom blokovnega načrta je shema (slika 2) hitro pripravljena in jo je potrebno le še prevesti. Povezovalna komponenta določi območje naslovov paralelnega vmesnika, konkretni naslov pa dobimo z odmikom, ki je odvisen od posamezne komponente. Izhod LED je na naslovu `0x41200000`, drugi priključek (GPIO) pa na naslovih z odmikom 8 za podatkovne bite in 12 za nastavitev smeri.

Nastavitveno datoteko (*.bit) prenesemo na razvojno ploščo po varni mrežni povezavi s programom Bitwise SSH client. Program omogoča tudi terminalsko povezavo z operacijskim sistemom Linux na razvojni plošči. V terminalu naložimo novo vsebino vezja FPGA in z monitorjem nastavljamo periferne registre.

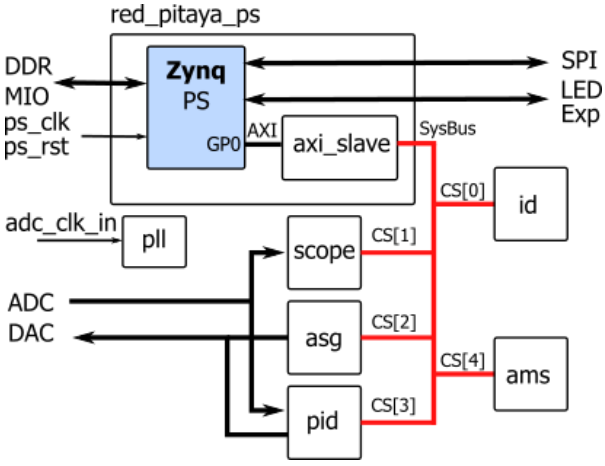
```
cat fpga_wrapper.bit > /dev/xdevcfg
monitor 0x41200000 0x03
monitor 0x41200008
```

Prvi ukaz prenese datoteko v gonilnik za nastavitev FPGA, drugi vklopi zadnji dve svetleči diodi, tretji pa prebere in izpiše stanje GPIO, ki so privzeto nastavljeni kot vhodi.

2.2 Vezje z AD in DA vmesnikom

Vključitev AD in DA pretvornika zahteva precej več logike zaradi specifik krmiljenja teh gradnikov. Komponente za krmiljenje pretvornikov niso na voljo v knjižnici orodja Vivado. Red Pitaya v repozitoriju ponuja projekte, ki jih uporabljajo njihove aplikacije (npr. *v0.94*, *classic*) in lahko služijo kot osnova za lasten projekt.

Ko prenesemo datoteke iz repozitorija rekonstruiramo projekt z izvedbo skript (Makefile), ki zahtevajo lokalno nameščen Linux in točno določeno različico orodja Vivado. S skriptami naredimo celoten projekt in izvedemo prevajanje oz. implementacijo vezja v skriptnem načinu orodja Vivado. Za operacijski sistem Windows smo na podlagi projekta iz repozitorija pripravili paket in primere vezij za obdelavo signalov [13]. Pripravljeni projekti omogočajo delo z grafičnim vmesnikom in različico Vivado 2019 ali 2020.



Slika 3: Blokovni diagram vezja *classic* z logiko osciloskopa (scope), generatorja signalov (asg, ams) in regulatorja (pid).

Paket vsebuje vse datoteke projekta *classic* z logiko dvokanalnega osciloskopa, signalnega genera-torja in regulatorja PID. Pripravili smo skripto s katero v orodju Vivado rekonstruiramo projekt in blokovni načrt vezja, ki je prikazano na sliki 3. Datoteke z opisom logike so narejene v jeziku System Verilog oz. Verilog.

Povezava logike s procesorskim delom je narejena v blokovnem načrtu, ki pa je namesto na zgornji ravni hierarhije skrit v komponenti `red_pitaya_ps`. V tej komponenti je tudi vmesnik med vodilom AXI in sistemskim vodilom (SysBus) na katerega so priklopljene vse periferne enote vezja, ki komunicirajo s procesorjem. Vmesnik razdeli periferni naslovni prostor na bloke dodeljene posameznim modulom (tabela 1).

Tabela 1: Razdelitev pomnilnika med periferne enote [14]

signal	naslov bloka	enota (komponenta)
CS[0]	0x40000000	Housekeeping (id)
CS[1]	0x40100000	Oscilloscope (scope)
CS[2]	0x40200000	Arbitrary sig. generator (asg)
CS[3]	0x40300000	PID controller (pid)
CS[4]	0x40400000	Analog mixed signals (ams)

Enota *Housekeping* v datoteki `red_pitaya_id` vsebuje osnovne nastavitvene registre, v enoti *Oscilloscope* pa je logika za proženje in zajem podatkov iz AD pretvornikov. Vzorčeni signali se prenašajo kot 14-bitne predznačene vrednosti z uro frekvence 125 MHz iz zunanjega vira `adc_clk_in`. Na to uro so povezane vse enote na signalni poti, vključno s komunikacijo na sistemskem vodilu. Komponenta `pll` vsebuje fazno sklenjeno zanko za generiranje fazno zamaknjenih in frekvenčno podvojenih signalov za krmiljenje AD in DA pretvornikov.

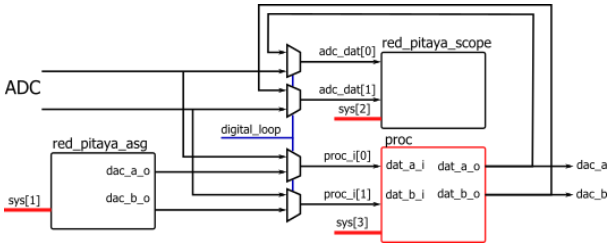
Tabela 2 prikazuje zasedenost logike FPGA s posameznimi enotami, ki smo jo ugotavljali s postopnim izključevanjem in implementiranjem vezja. Največ zasedeta osciloskop in signalni generator, kjer je uporabljen tudi blokovni pomnilnik za vzorce signalov.

Tabela 2: Zasedenost FPGA po modulih in za celoten projekt

enota	LUT	FF	DSP	BRAM
`ams`	219	338	0	0
`pid`	657	542	12	0
`asg`	1339	1229	2	14
`scope`	1759	1663	12	14
classic	4640	4712	26	28

Podatki iz AD pretvornika so vezani na osciloskop in `pid`, izhodi iz signalnega generatorja in `pid` pa so sešteti in vezani na DA pretvornik. Regulator `pid` je manj uporabna enota, zato jo nadomestimo s komponento za obdelavo signalov, ki je povezana kot prikazuje slika 4.

Priključitev komponente `proc` na sistemsko vodilo, na vhode iz AD pretvornika in izhode na DA pretvornik omogoča razvoj poljubne logike za obdelavo signalov. Če v vezju ohranimo signalni generator, ga lahko uporabimo za generiranje testnih signalov pri testiranju



Slika 4: Komponenta za obdelavo signalov (proc) je vključena med signalni generator in osciloskop.

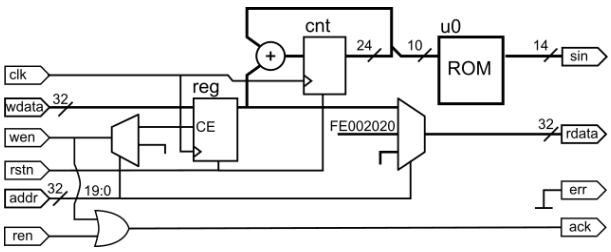
delovanja nove komponente na razvojni plošči, vgrajen osciloskop pa za pregled rezultatov obdelave signalov. V glavni datoteki projekta `red_pitaya_top` dodamo ustrezne povezave, nova komponenta pa je lahko opisana v jezikih (System)Verilog ali VHDL.

2.3 Učni primeri

Na spletni strani Laboratorija za načrtovanje integriranih vezij [13] so kratka navodila in nekaj učnih primerov načrtovanja vezij v orodju Vivado in programskih aplikacij.

Osnovni primer komponente za obdelavo signalov je vezje za skaliranje in povprečevanje vrednosti iz AD pretvornika. Faktor skaliranja zapišemo po sistemskem vodilu v register kot binarno vrednost s fiksno decimalko. Množenje lahko privede do preliva vrednosti, zato dodamo logiko za nasičenje na zgornji in spodnji meji območja DA pretvornika.

Drugi primer je numerično krmiljen oscilator. Vezje generira sinusni izhodni signal z nastavljivo frekvenco. Komponenta obdelave signalov vsebuje števec (cnt), ki mu vrednost inkrementa določamo z vpisom v register (reg) iz sistemskega vodila, kot prikazuje slika 5. Najvišji biti števca predstavljajo fazo s katero naslavljamo pomnilnik z vzorci sinusnega signala. Z dodatnimi registri in logiko dodamo še nastavljanje amplitude in odmika izhodnega signala.



Slika 5: Gradniki numerično krmiljenega oscilatorja in sistemski vmesnik za nastavljanje frekvence.

Generator signalov najprej preizkusimo s simulacijo, nato pa vključimo v projekt *classic* namesto komponente `pid`. Po prevajanju preizkusimo delovanje na razvojni plošči z nastavljanjem registrov v terminalu in vgrajenim osciloskopom. Izbiralnik in razdeljevalnik na sliki 5 vključujeta dekodiranje odmika naslovov za dostop do registra. Odmik naj bo 0x50 ali več, kar je izven območja registrov komponente `pid` [14], da ne bomo imeli konflikta z obstoječimi aplikacijami.

Tretji učni primer je komponenta s pomnilnikom za vzorčenje in shranjevanje signalov. Vsebuje sekvenčni stroj s štirimi stanji: iz začetnega izvede procesor pomik v stanje čakanja na prožilni pogoj (prehod vhodnega signala čez vrednost 0), nato pa določa naslove za vpis vzorcev v pomnilnik. Vsebino pomnilnika beremo zaporedno z uporabo števca, ki se poveča vsakokrat, ko dostopamo do perifernega registra za branje vsebine.

3 Programska oprema

Osnovne nastavitve parametrov in branje registrov izvajamo na plošči kar z uporabo terminala in programa monitor. Namensko programsko opremo, ki dostopa do perifernih enot za obdelavo signalov, razvijemo v jeziku C in prevedemo kar na sami razvojni plošči. Za dostop do perifernih registrov v Linuxu odpremo gonilnik `/dev/mem` in uporabimo funkcijo preslikave pomnilniških naslovov (`mmap`). Operacijski sistem na plošči podpira tudi izvajanje kode v jeziku Python.

Obstoječe aplikacije na merilni plošči uporabljajo za dostop do perifernih enot namenski gonilnik (`rp`). Gonilnik definira podatkovne strukture in funkcije za nastavljanje in branje vrednosti iz perifernih enot projekta *classic*. Postopek za posodobitev gonilnika:

- iz repozitorija Red Pitaya prenesemo izvorno kodo (mapa `rp-api`),
- dodamo funkcije (`rp.c` in `rp.h`),
- prevedemo gonilnik, premaknemo statično in dinamično knjižnico v `/opt/redpitaya/lib` ter nastavimo pot `LD_LIBRARY_PATH`

Ko odpremo spletno aplikacijo (npr. osciloskop), bo sistem naložil ustrezno nastavitveno datoteko v FPGA in programski krmilnik, ki odpre povezavo WebSocket. Krmilnik omogoča prenos podatkov v obliki JSON do aplikacije na spletnem brskalniku.

Če želimo uporabiti spletno aplikacijo za testiranje namenskega vezja, jo moramo povezati z nastavitveno datoteko pridobljeno iz orodja Vivado. Napisali smo skripto (`nastavi.sh`) s katero naredimo v terminalu ustrezne nastavitve [13]. Paziti moramo, da izvedemo skripto pred aplikacijo, med izvajanjem aplikacije pa ne smemo programirati vezja FPGA.

Posodobljeni gonilnik vsebuje funkcije za inicializacijo, branje in pisanje v naslovnem prostoru enote za obdelavo signalov:

```
void rp_ProcInit();
void rp_ProcRelease();

int rp_GetProcReg(uint32_t *val, int n);
int rp_SetProcReg(uint32_t val, int n);
```

4 Zaključek

Razvojna plošča STEMlab je uporabna platforma za prototipno načrtovanje sistemov za zajem in obdelavo digitalnih signalov. Odprtokodne rešitve omogočajo ponovno uporabo že obstoječih modelov in kode, zato se lahko posvetimo le specifikam načrtovanega sistema. Vgrajeno logiko osciloskopa in obstoječe aplikacije izkoristimo za testiranje in razhroščevanje vezij za obdelavo signalov na merilni plošči.

Proces razvoja strojne in programske opreme na SoC zahteva veliko različnih znanj. Za hitrejše uvajanje in začetek dela smo pripravili in objavili na spletnih straneh učne primere. Primeri predstavljajo zasnovo tipičnih vezij s področja digitalne obdelave signalov, ki jih lahko nadgrajujemo v zahtevnejša vezja. Pomembna je tudi ustrezna programska podpora.

V nadaljevanju dela želimo pripraviti tudi podporo za namenske spletne aplikacije in oddaljen standarden dostop do instrumentov (SCIPI), ki bi olajšal razvoj programske opreme.

Zahvala

Delo je bilo sofinancirano iz programa ARRS P2-0415 in R-534B.

Literatura

[1] M. Abdallah and O. Elkeelany, "A Survey on Data Acquisition Systems DAQ," 2009 International Conference on Computing, Engineering and Information, 2009, str. 240-243

[2] M. P. Patel, A. Ganatra, R. J. Nayak and J. B. Chavda, "Survey on developing data acquisition system using ZYNQ architecture," 2017 International Conference on Intelligent Sustainable Systems (ICISS), 2017, pp. 507-511, doi: 10.1109/ISS1.2017.8389463.

[3] XILINX all Programmable SoC with Hardware and Software Programmability, AMD Xilinx, (2022), https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[4] Welcome to the Red Pitaya documentation, Red Pitaya d.o.o., (2021), https://redpitaya.readthedocs.io/en/latest/

[5] Matija Mavsar, Večkanalni signalni generator na merilni napravi STEMlab, magistrsko delo, Univerza v Ljubljani, FE, 2018

[6] Jernej Kokalj, Generično vzporedno in sekvenčno digitalno sito na merilni napravi STEMlab, magistrsko delo, UL, FE, 2018

[7] L. H. Arnaldi, "Implementation of a Polyphase Filter Bank Channelizer on a Zynq FPGA," 2020 Argentine Conference on Electronics (CAE), 2020, str. 57-62,

[8] M. A. Luda et al., Compact embedded device for lock-in measurements and experiment active control, The Review of scientific instruments 90 2 (2019): 023106.

[9] Chaudhry Mendívil, Haris & Castillo, Cristiam & Verastegui, Joaquin. A Bi-static Pulsed Radar System Prototype Based on the Red Pitaya Development Platform, 2017

[10] M. Adamič, A. Trost, A Fast High-Resolution Time-to-Digital Converter Implemented in a Zynq 7010 SoC, 2019 Austrochip Workshop on Microelectronics, Dunaj, 2019, str. 29-34

[11] A. Rigonia, G. Manduchia, A. Luchettaa, C. Taliercioa, T. Schröderb, A framework for the integration of the development process of Linux FPGA System on Chip devices, Fusion Engineering and Design, Vol. 128, marec 2018, str. 122-125

[12] GitHub – RedPitaya, (2022), https://github.com/RedPitaya/RedPitaya-FPGA

[13] A. Trost, LNIV: Red Pitaya Projects, (2022), https://lniv.fe.uni-lj.si/redpitaya/

[14] Red Pitaya documentation: Register map (v0.94), (2022), https://redpitaya.readthedocs.io/en/latest/developerGuide/software/build/fpga/regset_common.html