Digitalni signalni generator na napravi STEMlab

Matija Mavsar, Andrej Trost

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

Digital signal generator on the STEMlab device

STEMlab is a low-cost programmable logic platform intended to replace many expensive laboratory measurement and control instruments. The STEMlab produced by Red Pitaya includes 2-channel oscilloscope and signal generator application. The open source software enables design of custom laboratory instruments. Our contribution is extension of the digital signal generator to an optional number of channels with arithmetic operations between the channels and the input signals. We implemented the changes at the levels of hardware description language, API interface and SCPI server. The extended signal generator can be used for digital signal processing experiments with 125MS/s RF outputs.

1 Uvod

Signalni generator je merilna oprema, ki jo skupaj z osciloskopom potrebujemo pri eksperimentiranju in razvoju elektronike. Eksperimentiranje z elektroniko je v izobraževalnih ustanovah del uvajanja v področja znanosti in tehnologije (ang. STEM - *Science, Technology, Engineering and Mathematics*). Namesto dragih merilnih instrumentov je podjetje Red Pitaya razvilo napravo STEMlab, ki deluje kot dvokanalni osciloskop in signalni generator [1]. Naprava ima odprto programsko kodo, ki omogoča razvoj različnih merilnih aplikacij [2][3].

Naprava STEMlab-14 ima dva vhodna in dva izhodna kanala s frekvenco zajemanja oziroma generiranja signalov 125 MHz, ločljivost signalov pa je 14-bitna. Na plošči je tudi razširitveni priključek z digitalnimi in analognimi vhodi ter izhodi. Jedro naprave je sistem na čipu (SoC) iz družine Xilinx Zynq-7000 [4] s programirljivim poljem logičnih vrat (ang. FPGA - *Field-Programmable Gate Array*), dvojedrnim procesorjem in bralno-pisalni pomnilnikom (RAM). Na procesorju teče operacijski sistem Linux s spletnimi aplikacijami za upravljanje naprave preko omrežja. Priključitev na omrežje je mogoča z Ethernet kablom ali z Wi-Fi USB priključkom.

V članku predstavljamo nadgradnjo signalnega generatorja ASG (ang. *Arbitrary Signal Generator*), pri katerem smo želeli povečati število kanalov, ki jih lahko uporabljamo za generacijo signalov, in omogočiti dodatne operacije med njimi. Naš cilj je bil razvoj strojnega opisa modula ASG s parametrično nastavljivim številom kanalov in ločljivostjo izhodov. Poleg logike generatorja signalov so potrebni tudi programski vmesniki za nastavljanje parametrov med delovanjem generatorja in izvajanje operacij med izbranimi kanali.

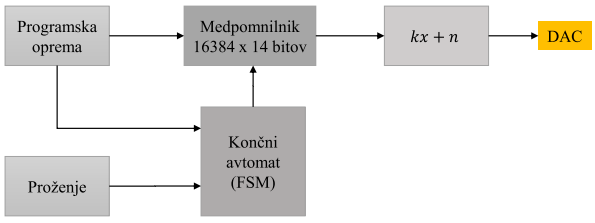
2 Signalni generator

Tabela 1 opisuje lastnosti obstoječega signalnega generatorja na napravi STEMlab. V osnovni izvedbi sta na voljo dva kanala z možnostjo nastavljanja oblike, amplitude, frekvence in enosmerne komponente signala.

Tabela 1: Lastnosti signalnega generatorja

Lastnost	Vrednost
Vzorčna frekvenca	125 MHz
Ločljivost DAC	14 bit
Pomnilnik vzorcev	16384
Napetostno območje	$\pm$ 1 V
Najvišja strmina napetosti	200 V / µs
Izhodna impedanca	50 Ω

Napravo STEMlab poganja sistem na čipu z zmogljivo programirljivo logiko in procesnim sistemom. Digitalna sinteza signala se izvaja v logiki programirljivega vezja, programska oprema pa nudi vmesnik za nastavitve parametrov izhoda in določanje oblike periodičnega signala. Slika 1 prikazuje glavne gradnike za izvedbo enega kanala.



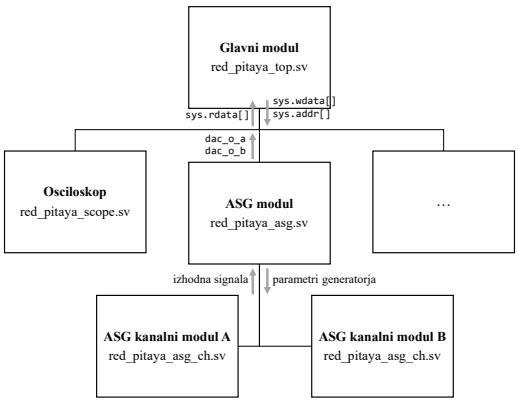
Slika 1: Gradniki enega kanala signalnega generatorja

Vsak kanal signalnega generatorja ima svoj končni avtomat (FSM), ki skrbi za periodično branje vzorcev iz medpomnilnika. Programska oprema ob inicializaciji zapiše obliko signala v medpomnilnik, med delovanjem pa nastavlja frekvenco, način proženja in vrednosti amplitude ter enosmerne komponente ($kx + n$). Izhod ob-

stoječega generatorja je neposredno povezan na digitalno-analogni pretvornik (DAC).

2.1 Modul signalnega generatorja

Signalni generator je na vezju FPGA realiziran s strojno opisnim jezikom SystemVerilog kot eden izmed 8 modulov oziroma IP (ang. *Intellectual Property*) komponent, vsaka pa ima na voljo 4 MB naslovnega prostora [1]. Uporaba že vnaprej načrtovanih in sintetiziranih IP komponent omogoča lažjo integracijo v projekte in tako zmanjša njihovo kompleksnost [5]. Hierarhijo modulov prikazuje slika 2.



Slika 2: Hierarhija modulov v jeziku Verilog

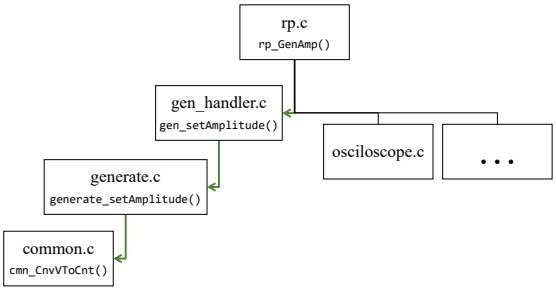
Spreminjanje parametrov generatorja poteka s pošiljanjem vrednosti na ustrezne naslove s pomočjo jezikov C (vmesnik API) ali Python (SCPI strežnik). Pri tem FPGA logika prebere naslove za pisanje in podatke zapiše v ustrezne registre. S procesorjem je povezana preko vmesnika AXI (ang. *Advances eXtensible Interface*), ki sestoji iz naslovnih in podatkovnih vodil za pisanje ter branje [6]. Na napravi je v FPGA logiki narejen pretvornik iz kompleksnega AXI vodila v enostavnejše vodilo *sys*. Nanj so v glavnem modulu priključene komponente programirljive logike (slika 2) in tako predstavlja sistemsko vodilo, ki omogoča zapisovanje v registre.

V ASG modulu je koda, potrebna za zapisovanje parametrov generatorja v registre in branje iz njih. Pri pisanju se glede na naslov, prejet preko naslovnega vodila, v ustrezen register zapiše trenutna vrednost na podatkovnem vodilu za pisanje. Pri branju registrov pa se vrednosti glede na naslov pošljejo na podatkovno vodilo za branje. V modulu je za vsak kanal generatorja definiran nov ASG kanalni podmodul, v katerega so posredovani parametri generatorja, ki skrbi za zajemanje vrednosti iz medpomnilnika in pošiljanje teh vrednosti na digitalno-analogni pretvornik. Vrednosti v medpomnilnik zapišemo s pomočjo programske opreme (prek signalov v ASG modulu), ki tudi nastavlja krmilni avtomat za branje vrednosti, način proženja in linearno transformacijo izhodov.

2.2 Nastavljanje parametrov signalnega generatorja

API vmesnik je napisan v jeziku C, sestavljajo pa ga različne datoteke, namenjene delovanju komponent programirljive logike. V glavni datoteki *rp.c* so združene funkcije, ki posredno upravljajo s parametri vseh komponent.

Nastavljanje parametrov signalnega generatorja je opisano v datoteki *generate.c*, kamor se preko glavnih funkcij iz *rp.c* posredujejo parametri. Ti se nato združijo v skupno strukturo in se s pomočjo kazalcev usmerijo na ustrezne naslove, ki jih FPGA logika dekodira in vrednosti zapiše v ustrezne registre. Na podoben način se v medpomnilnik zapišejo tudi vrednosti želenih izhodnih signalov, vendar s pomočjo drugih kazalcev. Potek klicanja funkcij za nastavitev amplitude je prikazan na sliki 3.



Slika 3: Potek klicanja funkcij za nastavitev amplitude

Spreminjanje parametrov ASG lahko poteka tudi z jezikom Python preko SCPI strežnika. Pri tem na ciljno napravo pošiljamo ukaze v obliki znakovnih nizov, po prejetju pa se kliče pripadajoča funkcija z želenimi parametri. Izvorna koda je napisana v jeziku C in se nanaša na kodo za API. Za delovanje generatorja je tako pomembna datoteka *generate.c*, v kateri so funkcije, ki ob prejetju določenega znakovnega niza kličejo ustrezno funkcijo iz vmesnika API. V datoteki *scpi-commands.c* pa je koda, ki določa, katera funkcija se bo klicala ob posameznem prejetem znakovnem nizu.

3 Večkanalni signalni generator

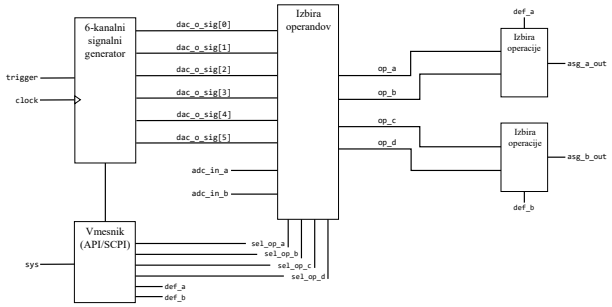
Blokovna shema večkanalnega signalnega generatorja je prikazana na sliki 4. Na sredini je prikazan blok z osmimi vhodnimi signali (šest iz signalnega generatorja in dva iz analognih vhodov naprave STEMlab), spodaj pa so štirje parametri, ki določajo, katere signale želimo izbrati kot operande. Vsak par operandov gre na blok procesnega modula, kjer vhodni parameter (*def_a* ali *def_b*) določa operacijo, ki se bo na paru operandov izvedla. Rezultata se nato pošljeta v glavni modul in na analogna izhoda naprave.

3.1 Logično vezje

Opis večkanalnega generatorja temelji na nadgradnji posameznih modulov referenčne kode v jeziku Verilog, ki je dostopna na GitHub repozitoriju Red Pitaye.

3.1.1 Glavni modul

V glavnem modulu smo dodali le kodo, ki s pomočjo pogojnega operatorja (ang. *ternary operator*) določa, ali naj se na vhod modula za osciloskop pošljeta izhodna



Slika 4: Blokovna shema večkanalnega generatorja

signala generatorja ali vrednosti iz analognih vhodov naprave.

3.1.2 Modul ASG

Pri opisu modula ASG smo v začetku deklaracije definirali parametra za nastavljanje števila kanalov in bitov ter pomožni parameter za maksimalno število bitov, dodali pa smo tudi dva 14-bitna vhoda za signala z analognih vhodov naprave, posredovana iz glavnega modula, in signal *send_scope* za pošiljanje izhodnih signalov na vhod osciloskopa. Nato smo deklarirali matriko vseh kanalov generatorja, signale, ki predstavljajo operande, in nove parametre iz tabele 2. Dodali smo stavke za izbiro operandov izmed vseh kanalov generatorja in analognih vhodov, ter spremenili način proženja kanalov, tako da se vsi lahko ponastavijo naenkrat.

Tabela 2: Registri novih parametrov

Parameter	Št. bitov	Opis
sel_op_a	3	izberi operand A
sel_op_b	3	izberi operand B
sel_op_c	3	izberi operand C
sel_op_d	3	izberi operand D
def_a	3	operacija med A in B
def_b	3	operacija med C in D
send_scope	1	pošiljanje na vhod
calib_amp_a	14	kalibracija prvega izhoda
calib_dc_a	14	
calib_amp_b	14	kalibracija drugega izhoda
calib_dc_b	14	
r_trig	$4 \cdot CN$	proženje kanalov
r_rst	CN	ponastavitev števcev

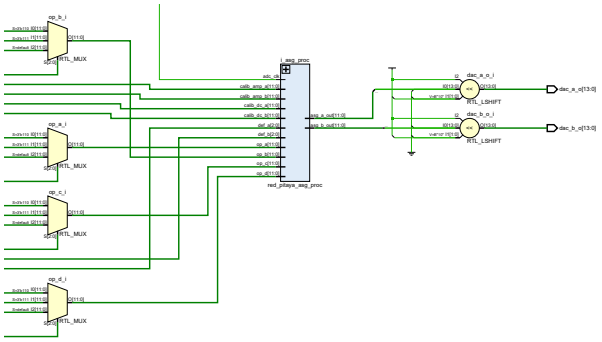
Kodi za zapisovanje in branje registrov smo dodali še nove parametre generatorja ter celotno kodo prilagodili za poljubno število kanalov in bitov.

3.1.3 Procesni modul

Za izvajanje aritmetičnih operacij med izbranimi operandi smo ustvarili nov procesni modul z dvema izhodoma (rezultata operacij), v katerega posredujemo parametre iz ASG modula (parametre za število kanalov, bitov, kalibracijo in izbiro operacij) in signale operandov. Deklarirali smo tudi vmesne signale za izvajanje operacij med operandi.

V *always* bloku je opisano računanje z operandi glede na vrednost parametra za izbiro operacije. Na rezultatu se nato upoštevajo kalibracijski parametri in decimalna mesta (pomik v desno). Končna signala sta zatem posredovana v glavni modul in na DA pretvornik.

Slika 5 prikazuje blokovno shemo procesnega modula, ki smo jo sintetizirali s programom Xilinx Vivado. Vezje sestavljajo multiplekserji za izbiro operandov, ki se skupaj z ostalimi parametri posredujejo v procesni modul, iz katerega izhajata dva izhoda, ki se pomakneta za ustrezno število bitov.



Slika 5: Shema sintetiziranega vezja s procesnim modulom

3.2 Aplikacijski vmesnik

Za uporabo nadgrajenega logičnega vezja signalnega generatorja smo naredili spremembe in napisali nove funkcije v vmesniku API.

- V datoteki *rp.h* smo dodali konstanto za število kanalov generatorja, podatkovni tip za izbiro kanala ter deklaracije novih funkcij.
- V *rp.c* smo definirali nove funkcije, ki se sklicujejo na funkcije iz datoteke *generate.c* z želenimi parametri.
- V *generate.h* smo dodali konstante (npr. začetni naslov za zapis novih parametrov v registre), definirali strukturo za spremenljivke novih parametrov (tabela 2) in deklarirali nove funkcije.
- V *generate.c* smo kodo posplošili za poljubno število kanalov, spremenili način proženja in kalibracije izhodnih signalov ter dodali še pošiljanje novih parametrov v registre FPGA logike.
- V datotekah *gen_handler.c* in *gen_handler.h* se je koda nekoliko poenostavila, saj smo namesto ločenega opisa delovanja kanal uporabili le indeks želenega kanala. Dodali smo funkcijo za ponastavitev kanalov in definirali nov parameter, ki ima v binarnem zapisu enice na mestu kanalov, ki jih želimo ponastaviti.

3.3 Strežnik SCPI

Delovanje signalnega generatorja se lahko omogoči tudi preko SCPI strežnika s pomočjo pošiljanja ukaznih nizov z jezikom Python. V nadaljevanju so opisani koraki za nadgradnjo obstoječe kode strežnika.

- V datoteki *generate.c* SCPI strežnika smo definirali funkcije za klicanje novih funkcij istoimenske datoteke API vmesnika. Vhodni parameter je znakovni niz iz katerega razberejo vrednosti in kličejo ustrezne funkcije API.
- V datoteki *scpi-commands.h* smo na seznam vseh obstoječih znakovnih ukaznih nizov dodali nize, ki pripadajo novim funkcijam. Tako se ob pošiljanju nekega niza z jezikom Python izvede ustrezna C funkcija.

4 Rezultati

Tabela 3 prikazuje zasedenost vezja FPGA po sintezi modula večkanalnega signalnega generatorja v programu Vivado. Sintetizirali smo eno- do osemkanalne generatorje s 14 bitno ločljivostjo.

Tabela 3: Zasedenost originalnega in nadgrajenega FPGA vezja v odstotkih glede na število kanalov

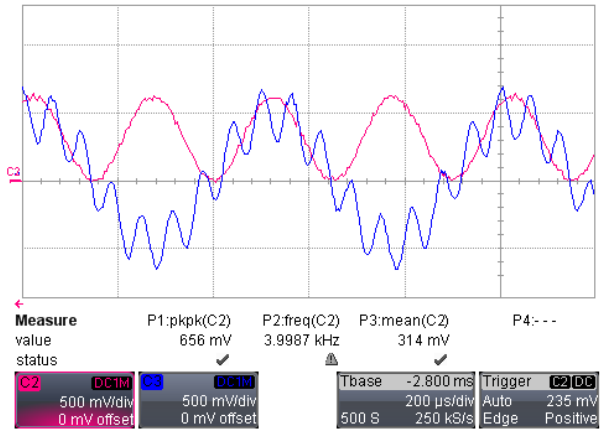
Tip enote	Orig.	Nadgrajen generator za različna št. kanalov							
		1	2	3	4	5	6	7	8
LUT – logika	5.18	4.28	6.91	10.10	12.72	15.19	18.07	20.46	23.30
Registri – flip-flop	2.61	1.76	3.00	4.27	5.50	6.78	8.01	9.15	10.37
F7 MUX	0.14	0.00	0.07	0.38	0.40	0.48	1.06	1.07	1.06
BRAM	23.33	11.67	23.33	35.00	46.67	58.33	70.00	81.67	93.33
DSP	2.50	6.25	7.50	8.75	10.00	11.25	12.50	12.50	12.50

Razvidno je naraščanje uporabljenih enot s številom kanalov generatorja. Nadgrajen generator za dva kanala porabi v primerjavi z originalnim nekoliko več LUT in registrov, trikratno število DSP-jev, enako število BRAM enot in pol manj F7 MUX.

Pri sintezi celotnega sistema se pri 6 kanalih in več uporabijo tudi LUT v obliki pomnilnika. Pri 6 kanalih je uporabljenih 0.02 %, pri 7 kanalih pa kar 85.35 % enot, saj pri 6 kanalih začne zmanjkovati BRAM enot.

4.1 Preizkus na napravi STEMlab

Slika 6 prikazuje oscilogram izhodnih signalov izmerjen na izhodu naprave STEMlab pri 4 kanalnem signalnem generatorju z nastavljenimi vhodi amplitude 0.8 V, 0.5 V in 0.2 V in frekvence 2 kHz ter 16 kHz.



Slika 6: Meritev izhodnega signala. Moder signal prikazuje vsoto dveh izhodov, pri čemer ima eden nižjo amplitudo in višjo frekvenco, rdeč signal pa je kvadrat sinusnega signala.

5 Zaključek

V pripevku smo opisali postopek nadgradnje strojne in programske opreme merilne in krmilne naprave STEMlab na primeru razvoja večkanalnega signalnega generatorja. Predstavili smo postopek nadgradnje kode na strojnem (Verilog) in programskem nivoju (C in Python), ki je podrobneje opisan v [7].

Z nadgradnjo signalnega generatorja smo uspešno omogočili izbiro poljubnega števila kanalov generatorja, dodali možnosti izbire vhodnih kanalov za operande in omogočili spreminjanje števila bitov, s katerim so signali opisani. Signalni generator smo preizkusili na napravi STEMlab z različnimi nastavitvami ter ugotovili, da je njegovo delovanje v skladu z želenimi lastnostmi.

Nadgradnja generatorja je lahko uporabna npr. pri modulaciji in obdelavi signalov ter nasploh pri raziskovanju oziroma eksperimentiranju, kjer potrebujemo nadzor nad več signali in sprotno izvajanje aritmetičnih operacij v realnem času.

Razvoj strojne in programske opreme za digitalno obdelavo signalov v sistemu SoC, ki je osrednji del naprave STEMlab zahteva specifična znanja iz obeh področij in poznavanje tehnologije. Možna rešitev za tiste uporabnike naprave, ki nimajo specifičnih znanj, je sinteza vezja iz programske kode in avtomatsko sestavljanje sistema iz ključnih komponent IP [8]. Večkanalni digitalni generator signalov je ena izmed takšnih komponent in ga bomo v nadalnjem delu uporabili pri razvoju računalniških orodij za eksperimentiranje z napravo STEMlab.

Literatura

[1] Red Pitaya. "Red Pitaya documentation", 2016. [Online]. Dosegljivo: http://rpdocs.readthedocs.io/en/latest/. [Dostopano: 7. 1. 2018].

[2] M. Ossmann: Red Pitaya – not just a USB scope module, Elektor, December 2014, str. 38–42.

[3] M. Cimerman, Z. Topčagić et al.: Laboratorijski merilni sistem na osnovi naprave Red Pitaya, zbornik ERK 2014, 22. - 24. september 2014, Portorož, Slovenija, zv. B, str. 215.

[4] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, R. W. Stewart: The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc, Strathclyde Academic Media, UK, 2014.

[5] T. Thomas, "Technology for IP reuse and portability", *IEEE Design & Test of Computers*, vol. 16, št. 4, str. 7-13, 1999.

[6] Xilinx, "AXI reference guide", podatkovni list, marec 2011 [revidirano: januar 2012].

[7] M. Mavsar: Večkanalni signalni generator na merilni napravi STEMlab, Magistrsko delo, FE, Ljubljana 2018.

[8] A. Trost, A. Žemva: Pipeline circuit synthesis from Python code, 6th Mediterranean Conference on Embedded Computing (MECO), 11 - 15 junij, 2017, Bar, Črna Gora, str. 320-323.