FPGA realizacija CIC filtra optimizovana po pitanju resursa

Marko Nikolić, Zoran Jakšić, *Institut „Mihajlo Pupin“*

***Sadržaj* — U ovom radu je predstavljen jedan način realizacije CIC filtra u FPGA tehnologiji koji je optimizovan po pitanju resursa. Na početku rada je pokazano da se dužina registara u CIC filtru može redukovati, a da pri tom izlazni signal ostane korektan. U drugom delu rada predstavljeno je konkretno rešenje FPGA realizacije. Rad se završava objašnjenjem funkcionalne verifikacije sistema, diskutuju se dobijeni rezultati, resursi potrebni za implementaciju za konkretno izabrani FPGA, kao i poređenje sa postojećim rešenjima.**

***Ključne reči* — CIC filtar, FPGA, decimacija, interpolacija.**

I. UVOD

Digitalni češljasti filtri predstavljaju značajnu klasu filtara koji se koriste u sistemima za konverziju frekvencije odabiranja. Oni najčešće nalaze primenu u slučaju velikih faktora decimacije ili interpolacije, jer tada klasični FIR filtri moraju imati jako veliku dužinu, koja eksponencijalno raste sa porastom faktora konverzije, dok se IIR filtri ne mogu praktično realizovati zbog efekata konačne dužine reči.

Ova klasa digitalnih filtara se odlikuje malom računskom složenošću, malom potrošnjom i velikom ekonomičnošću. Zbog toga, digitalni češljasti filtri mogu da rade i na vrlo visokim frekvencijama odabiranja. Kod višestepene decimacije oni se nalaze u prvom stepenu, dok se u slučaju višestepene interpolacije nalaze u poslednjem stepenu.

Istraživanja u ovoj oblasti su poslednjih godina intenzivirana zbog značaja ove grupe filtara u oblasti radio sistema, modema za kablovske distributivne sisteme, satelitskih prijemnika i radarskih sistema.

Najpoznatiji predstavnik digitalnih češljastih filtara je kaskadni integrator-češalj filtar (CIC - eng. Cascaded Integrator-Comb Filter), koji se realizuje bez množača, pri čemu je broj sabiranja minimizovan.

II. CIC FILTAR

CIC filtar[1][2] se realizuje kaskadnom vezom integratora i češljastog filtra (eng. *comb filter*).

Istraživanja prikazana u ovom radu su podržana od strane Ministarstva za nauku i tehnologiju Republike Srbije.

M. V. Nikolić, Institut Mihajlo Pupin, Volgina 15, 11060 Beograd, Srbija; (e-mail: marko.nikolic@institutepupin.com).

Z. M. Jakšić, Institut Mihajlo Pupin, Volgina 15, 11060 Beograd, Srbija; (e-mail: zoran.jaksic@institutepupin.com).

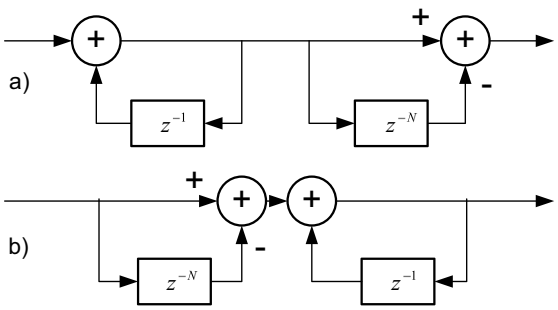
On ima identičnu funkciju prenosa kao pravougaoni FIR filtar, s tim što se, za razliku od njega, realizuje rekurzivnom realizacionom strukturom. Ulazno-izlazna relacija koja opisuje pravougaoni filtar data je sledećom relacijom.

$$y[n]=\sum_{k=0}^{N-1}x[n-k] \tag{1}$$

Po jednom izlaznom odbirku potrebno je N – 1 operacija sabiranja i N operacija pomeranja, gde je N dužina filtra. Broj operacija sabiranja se može značajno smanjiti ako se iskoristi sledeći identitet

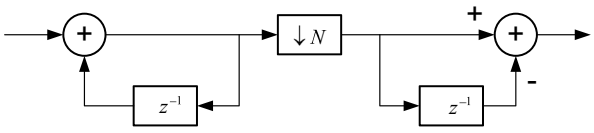
$$y[n]=y[n-1]+x[n]-x[n-N] \tag{2}$$

Poslednja jednačina daje ulazno-izlaznu relaciju CIC filtra, čija je blok-šema prikazana na Sl.1. Redosled integratora i češljastog filtra može biti proizvoljan, jer oba filtra pripadaju linearnim vremenski invarijantnim sistemima.



Sl. 1. Blok šema CIC filtra: a) na ulazu CIC filtra je integrator, b) na ulazu CIC filtra je češljasti filtar

Hogenauer je prvi predložio primenu ovih filtara za konverziju frekvencije odabiranja [2]. On je primenio kaskadne ekvivalencije na CIC filtar, čime je broj pomeranja u nerekurzivnom delu filtra smanjen na 1, tj. češljasti filtar je sveden na diferencijator. Decimacioni i interpolacioni CIC filtar se razlikuju po redosledu integratora i diferencijatora. Na Sl. 2. je prikazana Hogenauerova efikasna realizacija decimacionog CIC filtra. Treba napomenuti da se kaskadne ekvivalencije mogu primeniti i na CIC filtar sa više sekcija.



Sl. 2. Hogenauerov decimacioni filtar

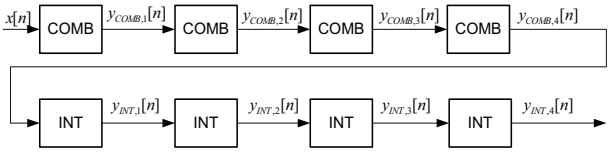
Nula i pol CIC filtra u tački $z = 1$, koji se međusobno poništavaju, razdvojeni su u dva filtra, što nije slučaj kod pravougaonog filtra. Ovo razdvajanje nule i pola ima velike posledice na funkcionisanje CIC filtra. Naime, integrator je nestabilan filtar, a kako su u realnom slučaju dužine registara ograničene, neminovno dolazi do prekoračenja u integratorskim sekcijama. Uprkos prekoračenju opsega u integratorskim sekcijama, može se dobiti korektan signal na izlazu [3]. Potreban uslov za to je da registri u integratorskim sekcijama i sekcijama češljastih filtara budu dovoljne dužine i da se koristi aritmetika u komplementu dvojke [3][4].

Potrebna dužina registra filtra da ne bi došlo do prekoračenja može izračunati prema sledećoj jednačini:

$$B_{ACCUM} = B_{IN} + ceil\ \ K \log_2 N \qquad (3)$$

gde je B_{IN} ulazna dužina reči, K red filtra, a N dužina svake sekcije.

U daljem tekstu posmatraćemo primer četvorostepenog interpolacionog filtra koji je prikazan na Sl. 3.



Sl. 3. Blok-šema četvorostepenog interpolacionog CIC filtra

Dužina reči svakog posebnog češljastog filtra (COMB) i integratora (INT) može se smanjiti, a da na izlazu kompletnog filtra ne dođe do prekoračenja. Ovaj postupak zbog svoje obimnosti nije prikazan u ovom radu, a može se pronaći u literaturi [3][5]. U Tabeli 1 prikazano je kako se dužina registra svakog filtra povećava od ulaza ka izlazu, zato što svaka naredna sekcija unosi pojačanje. Dužina svake sekcije je $N = 20$.

TABELA 1: PORAST BROJA BITA ZA INTERPOLACIONI CIC FILTAR U ODNOSU NA ULAZ

Filtar	***Porast broja bita***
Češljasti filtar 1	1
Češljasti filtar 2	2
Češljasti filtar 3	3
Češljasti filtar 4	4
Integrator 1	9
Integrator 2	10
Integrator 3	14
Integrator 4	18

Slično tome, dužina registra kod decimacionog filtra za svaki stepen se može optimizovati. U takvoj implementaciji se dužina registra svakog narednog filtra smanjuje. Za četvorostepeni CIC filtar, kod koga je dužina svake sekcije $N = 20$, to je ilustrovano Tabelom 2. Skraćivanje dužine registara kod decimacionog CIC filtra je ekvivalentno dodavanju šuma, a dužina svakog registra se izračunava tako da doprinos ovog šuma bude manji od 1 LSB-a na izlazu filtra [3].

TABELA 2: PORAST BROJA BITA ZA DECIMACIONI CIC FILTAR U ODNOSU NA IZLAZ

Filtar	***Porast broja bita***
Integrator 1	18
Integrator 2	14
Integrator 3	11
Integrator 4	8
Češljasti filtar 1	7
Češljasti filtar 2	6
Češljasti filtar 3	5
Češljasti filtar 4	4

III. NAČIN IMPLEMENTACIJE, ALATI, VERIFIKACIJA REŠENJA

Zbog fleksibilnosti koju poseduje, odlučeno je da se za praktično ispitivanje resursa potrebnih za realizaciju CIC filtra koristi FPGA tehnologija.

Najpre su u Matlab-u napisane posebne funkcije koje u zavisnosti od parametara filtra (ulazna dužina reči, red, i dužina sekcije) računaju potrebnu dužinu registra svakog posebnog filtra integratora i diferencijatora (COMB) [6].

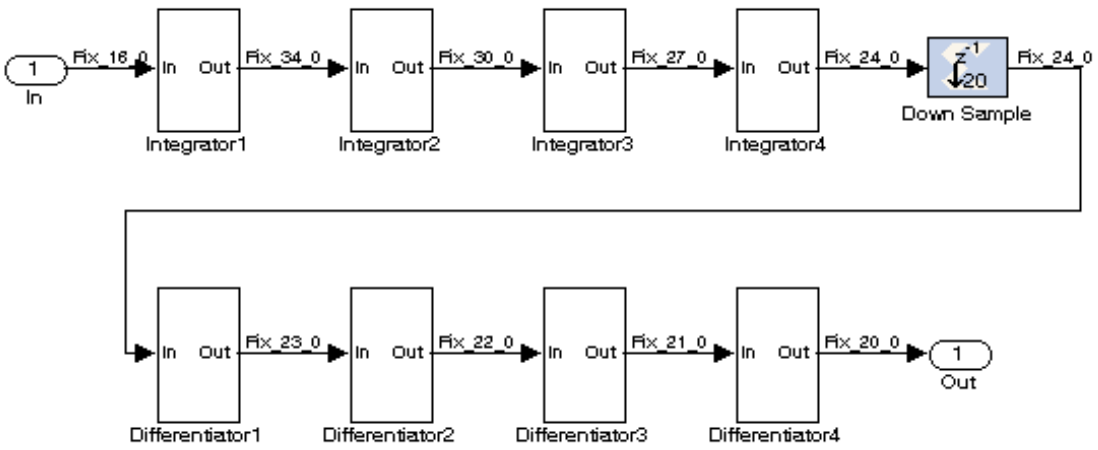
Za implementaciju filtra u FPGA korišćen je razvojni alat System Generator [7] firme Xilinx, čiji je Spartan XC6SLX45 [8] korišćen za praktičnu realizaciju, tj. ispitivanje potrebnih resursa za realizaciju.

System Generator predstavlja skup posebnih biblioteka dizajniranih od strane Xilinx-a, koji se kroz Simulink integrišu u Matlab. Ovo omogućava jednostavnije kreiranje modela kao i funkcionalnu verifikaciju dizajna korišćenjem Matlab programskog jezika.

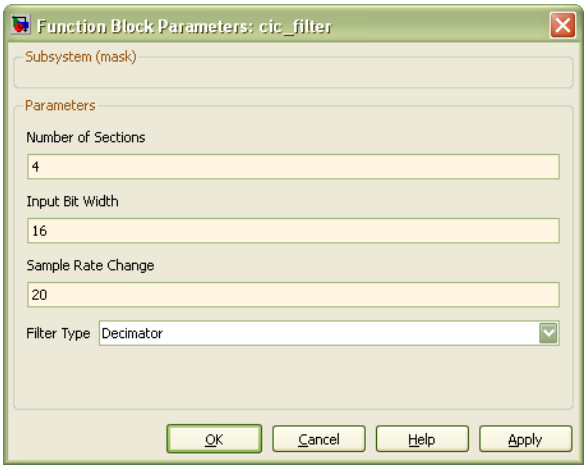
Nakon obavljene funkcionalne verifikacije, okruženje omogućava jednostavno generisanje sintetizabilnog VHDL koda koji se dalje može instancirati kao poseban modul u nekom drugom projektu. Ovakav način implementacije je mnogo jednostavniji od direktnog pisanja složenog VHDL koda koje zahteva znatno više vremena. Posebna pogodnost je jednostavna integracija sa Matlab programskim okruženjem koji je praktično standard kada je oblast digitalne obrade signala u pitanju. Ovakav pristup generalno omogućava jednostavnije poređenje rezultata dobijenih simulacijom FPGA modela sa rezultatima dobijenim simulacijom prethodno razvijenog algoritma.

Za ovaj rad je posebno značajno što se u bibliotekama System Generator-a može pronaći realizacija CIC filtra čije su dužine registara konstantne. Ova realizacija je kasnije iskorišćena za poređenje resursa potrebnih za implementaciju na konkretnom FPGA kada se koristi konstantna (maksimalna) i promenljiva (redukovana) dužina registara. Na taj način se najbolje može uočiti ušteda resursa u FPGA kada se za implementaciju CIC filtra koristi postupak smanjivanja dužine registara.

Na Sl. 4. je prikazan System Generator model dizajniranog CIC filtra, dok je na Sl. 5. prikazan interfejs za njegovu konfiguraciju. U konkretnom primeru realizovan je decimacioni CIC filtar sa 4 sekcije, sa faktorom decimacije 20 i dužinom ulazne reči 16 bita.



Sl. 4. System generator model dizajniranog CIC filtra sa 4 sekcije, faktorom decimacije 20 i dužinom ulazne reči 16 bita



Sl. 5. Prikaz interfejsa za konfiguraciju CIC filtra

Matlab Simulink model je realizovan po ugledu na Xilinx-ovo rešenje. Sam konfiguracioni interfejs modula je prilično razumljiv i fleksibilan za korišćenje. Od korisnika se jednostavno zahteva da unese parametre filtra, nakon čega posebna funkcija za inicijalizaciju jednostavno generiše model. Od parametara je moguće menjati: broj sekcija (*Number of Sections*), dužinu reči ulaznog signala (*Input Bit Width*), faktor decimacije/interpolacije (*Sample Rate Change*) i tip filtra (*Filter Type*).

Sistem je vrlo lako i brzo verifikovan korišćenjem Xilinx-ove realizacije CIC filtra na taj način što je za isti ulazni signal poređen izlaz postojećeg (Xilinx-ovog) i predloženog rešenja.

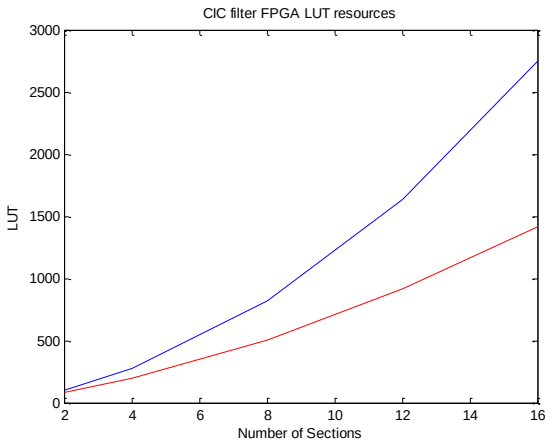
IV. Rezultati. Resursi potrebni za implementaciju

U ovoj sekciji su prikazani rezultati, koji se odnose na potrebne resurse za realizaciju CIC filtra sa redukovanom dužinom registara u FPGA tehnologiji. Takođe je kvantifikovano i prikazano smanjenje resursa u odnosu na slučaj kada CIC filtar ima konstantnu (maksimalnu) dužinu registara. Resursi su snimani za različite parametre filtra. Za procenu resursa korišćen je alat Resource Estimator (dolazi u paketu sa System Generatorom) koji brzo i efikasno procenjuje resurse čipa potrebne za implementaciju System Generator modela.

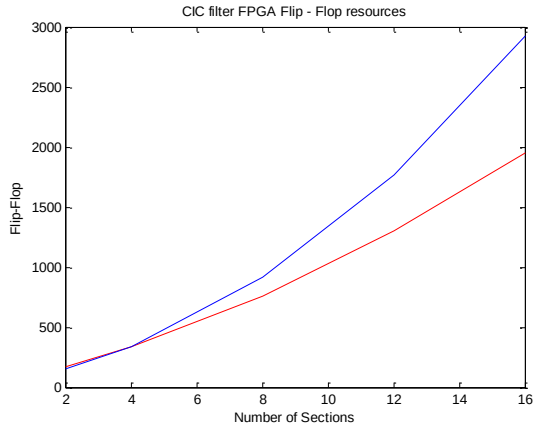
Za ispitivanje hardverskih resursa potrebnih za implementaciju korišćen je FPGA Spartan 6 familije XC6SLX45. Iako raspolaže različitim tipovima hardverskih resursa, koji uključuju posebne memorijske blokove na čipu, množače i još neke dodatne resurse, zbog svoje jednostavnosti za implementaciju CIC filtra korišćeni su samo flip-flop-ovi kao i look-up tabele (LUT).

U nastavku je prikazan broj potrebnih LUT-a i flip-flop-ova za implementaciju CIC filtra u zavisnosti od broja sekcija i faktora decimacije. Plavom bojom označeni su resursi potrebni za implementaciju već postojećeg rešenja koje u System Generatoru daje Xilinx, a crvenom bojom prikazuju se resursi potrebni za implementaciju rešenja koje je predloženo u ovom radu.

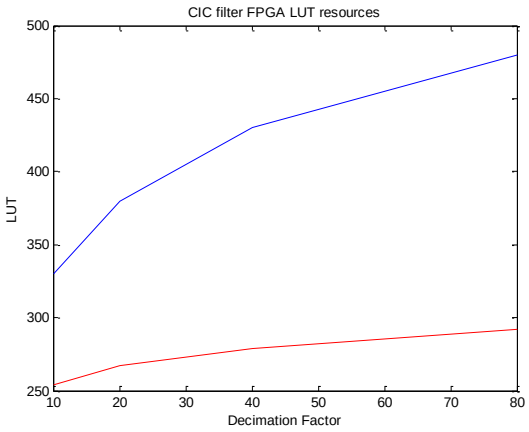
Najpre su na Sl. 6. i Sl. 7. prikazani resursi snimljeni za slučaj implementacije CIC filtara sa različitim brojem sekcija, kod kojih je dužina ulazne reči 16 bita, a decimacioni faktor 20. Zatim su snimljeni resursi za CIC filtre sa različitim decimacionim faktorom, dok je dužina ulazne reči 16 bita, a broj sekcija je fiksiran na 5 (Sl. 8. i Sl. 9.).



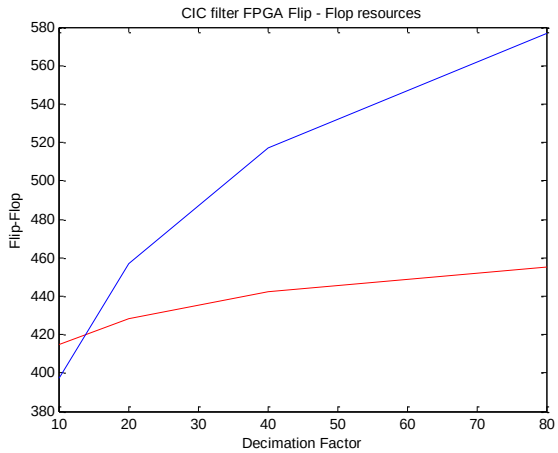
Sl. 6. Prikaz broja potrebnih LUT-a za implementaciju CIC filtra u zavisnosti od broja sekcija; plavo – postojeće rešenje, crveno – predloženo rešenje



Sl. 7. Prikaz broja potrebnih flip-flop-ova za implementaciju CIC filtra u zavisnosti od broja sekcija; plavo – postojeće rešenje, crveno – predloženo rešenje



Sl. 8. Prikaz broja potrebnih LUT-a za implementaciju CIC filtra u zavisnosti od decimacionog faktora; plavo – postojeće rešenje, crveno – predloženo rešenje



Sl. 9. Prikaz broja potrebnih flip-flop-ova za implementaciju CIC filtra u zavisnosti od decimacionog faktora; plavo – postojeće rešenje, crveno – predloženo rešenje

V. ZAKLJUČAK

U radu je predstavljena jedna efikasna realizacija CIC filtra u FPGA tehnologiji. Cilj rada je bio da se, pre svega, predstavi način na koji se CIC filtri mogu implementirati u FPGA, a onda i da se pokaže kako se postojeća rešenja mogu optimizovati. Značaj predložene realizacije upravo se ogleda u smanjenju resursa potrebnih za implementaciju decimacionog i interpolacionog CIC filtra kada se koristi promenljiva dužina registra u odnosu na postojeće rešenje filtra za čiju je realizaciju korišćena konstantna (maksimalna) dužina registra. Ova ušteda u resursima posebno je izražena za veće faktore decimacije ili interpolacije i veći broj sekcija CIC filtra.

LITERATURA

[1] L. D. Milić, Multirate Filtering for Digital Signal Processing: MATLAB Applications, Ed. Information Science Reference, 2009, ch. 4, 5, and 11.

[2] E. B. Hogenauer: ”An economical class of digital filters for decimation and interpolation”, IEEE Trans. Acoustics, Speech, and Signal Processing, 29(2), 155-162 , 1981

[3] F. J. Harris, Multirate Signal Processing for Communication Systems, Prentice Hall PTR, 2004, ch. 11

[4] M. V. Popović: Digitalna obrada signala, Drugo izdanje, Nauka, Beograd 1997.

[5] M. V. Nikolić: „Realizacija digitalnih češljastih filtara u aritmetici sa fiksnom tačkom", rad 5.3, Telfor 2009.

[6] M. V. Nikolić, Metode projektovanja kaskadnih integrator-češalj digitalnih filtara. Magistarska teza, Elektrotehnički fakultet Univerziteta u Beogradu, Oktobar 2010.

[7] System Generator for DSP, release 10.1, March, 2010, www.xilinx.com

[8] Spartan 6 family overview, DS160 (v 1.5) August 2, 2010, Advanced product specification www.xilinx.com

ABSTRACT

This paper presents one example of resource optimized FPGA implementation of CIC filter. At the beggining of the paper we present how the registers’ length can be reduced in Hogenauer’s implementation of interpolation or decimation CIC filter. In the second part of the paper one specific solution of FPGA implementation is shown. System Generator and Matlab are used for implementation and resource estimation. At the end, functional verification of the system and resources needed for implementation for specific FPGA have been presented, as well as resource comparison with the existing solutions.

FPGA IMPLEMENTATION OF RESOURCE OPTIMIZED CIC FILTER

Marko Nikolić, Zoran Jakšić