FPGA implementacija sum-product algoritma za dekodovanje LDPC kodova

Srđan S. Brkić, Dajana M. Lazarević, Predrag N. Ivaniš
Elektrotehnički fakultet, Univerziteta u Beogradu
Beograd, Republika Srbija
brka05@gmail.com, radovic.dajana@gmail.com, predrag.ivanis@etf.rs

Sadržaj—**U ovom radu predstavljen je FPGA (eng. *Field Programmable Gate Array*) dizajn algoritma *sum-product* koji se koristi za dekodovanje LDPC (eng. *Low Dencity Parity Check*) kodova. Predloženi dizajn minimizira broj taktnih intervala potrebnih za proces dekodovanja, tako da se kodne reči mogu dekodovati u roku od nekoliko mikrosekundi, što omogućava ostvarivanje protoka od više desetina Mb/s.**

Ključne riječi-FPGA; Low Density Parity Check; sum-product algoritam;

I. UVOD

U razgranatoj oblasti formiranja zaštitnih kodova trenutno najbolje performanse postižu takozvani kodovi sa malom gusinom provera parnosti (eng. *Low Density Parity Check* - LDPC), koji su se približili Šenonovoj granici na samo deo decibela [1], [2]. Iako su metode njihove konstrukcije i dalje predmet brojnih istraživanja, posebna pažnja se posvećuje razvijanju novih algoritama dekodovanja kao i efikasnim realizacijama dekodera LDPC kodova.

Algoritam dekodovanja pod nazivom *sum-product* predstavlja najbolji poznati (eng. *state-of-the-art*) algoritam dekodovanja LDPC kodova [3]. Algoritam koristi Tanerovu grafovsku reprezentaciju LDPC kodova [4] i omogućava razmenu informacija između čvorova grafa u nizu iteracija na osnovu kojih proces dekodovanja treba da rekonstruiše poslatu informacionu poruku. Dobre performanse algoritma praćene su visokim nivoom kompleksnosti, pa se u literaturi mogu naći i različite aproksimacije originalnog algoritma koje pojednostavljuju implementaciju [5], [6]. Aproksimacije više ili manje degradiraju performanse algoritma, pa je u ovom radu implementiran originalni *sum-product* algoritam prilagođen hardverskoj implementaciji.

FPGA (eng. *Field Programmable Gate Array*) tehnologija omogućava efikasnu implementaciju kompleksnih algoritama uz veliki nivo paralelizma operacija [7]. Tako se i *sum-product* dekoder može efikasno implementirati na FPGA čipu. Takođe, analizu performansi pojedinih LDPC kodova nepraktično je obavljati softverski i formiranje FPGA dizajna omogućava značajno brže ispitivanje performansi kodova većih dužina kodnih reči.

U ovom radu predstavljen je FPGA dizajn *sum-product* dekodera LDPC kodova kod koga se sve poruke koje razmenjuju čvorovi grafa prenose istovremeno. Ovakva paralelna arhitektura omogućava kraće vreme dekodovanja od drugih tipova kao što su serijska ili hibridna arhitektura.

U odeljku II opisan je originalni *sum-product* algoritam, kao i modifikacija koja čuva nivo preciznosti, ali algoritam prilagođava za hardversku implementaciju. U odeljku III predstavljen je FPGA dizajn algoritma i ispitano kakve performanse postiže na primeru nekoliko različitih LDPC kodova. U poglavlju IV data su neka zaključna razmatranja.

II. SUM-PRODUCT ALGORITAM

A. Sum-product u LLR domenu

Sum-product algoritam dekodovanja koristi Tanerovu grafovsku strukturu, kojom se opisuje kontrolna matrica LDPC koda i pretpostavlja da u jednoj iteraciji čvorovi međusobno šalju poruke pomoću kojih se vrši dekodovanje. Postoje dva pristupa pri opisu *sum-product* algoritma i to: putem prenosa verovatnoća pojavljivanja pojedinog bita u informacionoj poruci, ili putem prenosa tzv. LLR (eng. *Log-Likelihood Ratio*) vrednosti. Opis u LLR domenu je lakši za realizaciju, pa će u nastavku ukratko biti predstavljan algoritam čija je praktična implementacija tema ovog rada.

Neka se posmatra binarni LDPC kod u oznaci (N,K), koji se može opisati kontrolnom matricom **H** dimenzija MxN. Matrica **H** je male gustine tj. većina njenih elemenata je jednaka 0. Dimenzije kontrolne matrice otkrivaju da u grafovskom prikazu posmatrani kod ima N simbolskih čvorova i M kontrolnih čvorova. Svaki simbolski čvor i povezan je sa p_i (i=1,2,...,N) kontrolnih čvorova, dok je svaki kontrolni čvor j povezan sa q_j (j=1,2,...,M) simbolskih čvorova. Ako se dekoduje neregularan kod vrednosti p_i (i=1,2,...,N) i q_j (j=1,2,...,M) nisu konstante, dok je kod regularnih kodova svaki simbolski čvor je povezan sa istim brojem kontrolnih čvorova i obratno, svaki kontrolni čvor je povezana sa istim brojem simbolskih čvorova.

Neka se sa $A(n)$ označi skup kontrolnih čvorova povezanih sa simbolskim čvorom n, tj. skup pozicija jedinica u n-toj koloni kontrolne matrice **H**, dok se sa $B(m)$ može označiti skup simbolskih čvorova koji učestvuju m-toj jednačini provere parnosti, tj. skup pozicija jedinica u m-toj koloni matrice **H**. Dalje se može definisati i skup $B(m)\backslash n$ koji predstavlja skup $B(m)$ iz koga je izbačen n-ti simbolski čvor. Slično, skup $A(n)\backslash m$ predstavlja skup $A(n)$ bez pozicije m-tog kontrolnog čvora.

Dodatno se mogu definisati i poruke kojima pojedini čvorovi međusobno komuniciraju. Tako $q_{n\to m(x)}$, $x\in\{0,1\}$, označava poruku koju simbolski čvor n šalje kontrolnom čvoru m koja sadrži njegovu procenu o vrednosti n-tog simbola (da li je simbol ima vrednost 1 ili 0). Slično, $r_{m\to n(x)}$, $x\in\{0,1\}$, označava poruku koju m-ti kontrolni čvor šalje n-tom simbolskom čvoru, koja sadrži procenu m-tog kontrolnog čvora o vrednosti n-tog simbola, dobijenu na osnovu jednačina provera parnosti u kojima učestvuje m-ti kontrolni čvor. Na kraju, primljena kodna reč je označena sa $\mathbf{y}=[y_1,y_2,...,y_N]$ i ona odgovara poslatoj kodnoj reči $\mathbf{u}=[u_1,u_2,...,u_N]$.

LLR količnik binarne vrednosti slučajne promenljive U se može definisati kao

$$L(U)=\log\frac{P(U=0)}{P(U=1)}, \tag{1}$$

gde $P(U=x)$ označava verovatnoću da slučajna promenljiva U ima vrednost x. Dalje se mogu definisati i LLR vrednosti poruka koje razmenjuju čvorovi kao

$$\begin{aligned}\lambda_{n\to m}(u_n)&=\log\left(q_{n\to m}(0)/q_{n\to m}(1)\right),\\ \Lambda_{m\to n}(u_n)&=\log\left(r_{m\to n}(0)/r_{m\to n}(1)\right).\end{aligned} \tag{2}$$

U nastavku odeljka će ukratko biti predstavljen opis *sum-product* algoritam preuzet iz rada [8].

Inicijalizacija. Svakom simbolskom čvoru n dodeljuje se aposteriori LLR vrednost

$$L(u_n)=\log\left(P(u_n=0\,|\,y_n)/P(u_n=1\,|\,y_n)\right). \tag{3}$$

U slučaju da su ulazni simboli jednako verovatni i da u kanalu deluje samo beli Gausov šum važi $L(u_n)=2y_n/\sigma^2$, gde je σ^2 varijansa šuma. Za svaku poziciju (m,n) za koju je $H_{m,n}=1$ se inicijalno podrazumeva

$$\begin{aligned}\lambda_{n\to m}(u_n)&=L(u_n),\\ \Lambda_{m\to n}(u_n)&=0.\end{aligned} \tag{4}$$

Korak (i) *(ažuriranje kontrolnih čvorova)*. Za svako m i svako $n\in A(m)$ računa se vrednost koju kontrolni čvor šalje izabranom simbolskom čvoru na sledeći način

$$\Lambda_{m\to n}(u_n)=2\tanh^{-1}\left\{\prod_{n'\in A(m)\backslash n}\tanh\left[\lambda_{n'\to m}(u_{n'})/2\right]\right\}. \tag{5}$$

Korak (ii) (*ažuriranje simbolskih čvorova)*. Za svako n i svako $m\in B(n)$ računa se vrednost koju simbolski čvor šalje izabranom kontrolnom čvoru na sledeći način

$$\lambda_{n\to m}(u_n)=L(u_n)+\sum_{m'\in B(n)\backslash m}\Lambda_{m'\to n}(u_n). \tag{6}$$

Takođe, u toku svake iteracije mogu se izračunati i meke odluke o svakom poslatom bitu n (n=1,2,...,N) kao

$$\lambda_n(u_n)=L(u_n)+\sum_{m\in B(n)}\Lambda_{m\to n}(u_n). \tag{7}$$

Korak (iii) *(donošenje odluke)*. Konačna procena poruke $\hat{\mathbf{u}}=[\hat{u}_1,\hat{u}_2,...,\hat{u}_N,]$ dobija se kvantizacijom mekih odluka $\lambda_n(u_n)$ prema pravilu: $\hat{u}_n=0$ ako je $\lambda_n(u_n)\geq 0$, u suprotnom ako je $\lambda_n(u_n)<0$, smatra se da je $\hat{u}_n=1$.

Koraci *(i)* i *(ii)* se ponavljaju u unapred izabranom broju iteracija nakon čega se pristupa donešenju odluke. Opciono moguće je i nakon svake iteracije proveriti da su zadovoljene sve provere parnosti, tj. ako je $\hat{\mathbf{u}}\mathbf{H}^T=0$, proces dekodovanja se prekida i smatra se da su sve greške ispravljene.

B. Modifikacija sum-product algoritma

Očigledno je da su operacije ažuriranja kontrolnih čvorova najkompleksnije i teško ih je realizovati u hardveru u originalnoj formi. U nastavku će biti opisana modifikacija ažuriranja kontrolnih čvorova preuzeta iz rada [8].

Za svaki kontrolni čvor m, može definisati pomoćna promenljiva S_m na sledeći način

$$S_m=\sum_{i=1}^{q_m}\oplus u_{n_i}. \tag{8}$$

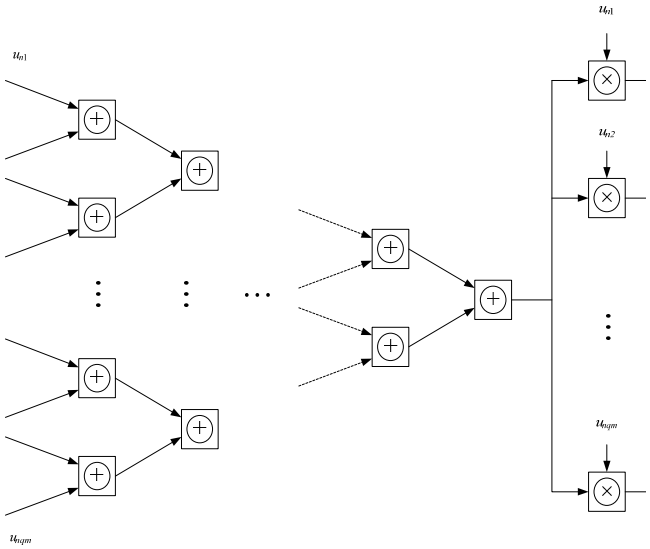
LLR vrednost promenljive S_m se za svaki čvor može izračunati koristeći topologiju stabla prikazanu na slici 3.1. Tako se proces računanja LLR vrednosti ekskluzivnog sabiranja više slučajnih promenljivih može svesti na računanje funkcija od dve promenljive $L(U\oplus V)$, čija će efikasna implementacija biti opisana u nastavku. Koristeći Jakobijev algoritam može se doći do sledećeg

$$L(U\oplus V)=\operatorname{sign}\left[L(U)\right]\operatorname{sign}\left[L(V)\right]\cdot\min\left[\left|L(U)\right|,\left|L(V)\right|\right]+\log\left[1+e^{-\left|L(U)+L(V)\right|}\right]-\log\left[1+e^{-\left|L(U)-L(V)\right|}\right], \quad (9)$$

gde se logaritmi $\log\left[1+e^{-\left|L(U)+L(V)\right|}\right]$ i $\log\left[1+e^{-\left|L(U)-L(V)\right|}\right]$ računaju koristeći skup aproksimativnih funkcija koje su date u tabeli I.

TABELA I. APROKSIMACIJA FUNKCIJE $f(x)=\log\left[1+e^{-|x|}\right]$.

$\|x\|$	$\log\left[1+e^{-\|x\|}\right]$	$\|x\|$	$\log\left[1+e^{-\|x\|}\right]$
[0,0.5)	$-\|x\|*2^{-1}+0.7$	[2.2,3.2)	$-\|x\|*2^{-4}+0.2375$
[05,1.6)	$-\|x\|*2^{-2}+0.575$	[3.2,4.4)	$-\|x\|*2^{-5}+0.1375$
[1.6,2.2)	$-\|x\|*2^{-3}+0.375$	[4.4,+∞)	0



Slika 1. Konfiguracija za ažuriranje kontrolnih čvorova.

Nakon određivanja LLR vrednosti promenljive S_m moguće je izračunati i vrednosti $\Lambda_{m\to n_i}\left(u_{n_i}\right)$ na sledeći način

$$\Lambda_{m\to n_i}\left(u_{n_i}\right)=L\left(\lambda_{n_i\to m}\left(u_{n_i}\right)\otimes L(S_m)\right)=\log\left|e^{\lambda_{n_i\to m}\left(u_{n_i}\right)+L(S_m)}-1\right|-\log\left|e^{\lambda_{n_i\to m}\left(u_{n_i}\right)-L(S_m)}-1\right|-L(S_m), \quad (10)$$

izrazi oblika $h(x)=\log\left|e^{x}-1\right|$ mogu se efikasno računati koristeći linerane aproksimativne funkcije date u tabeli II.

TABELA II. APROKSIMACIJA FUNKCIJE $h(x)=\log\left|e^{x}-1\right|$.

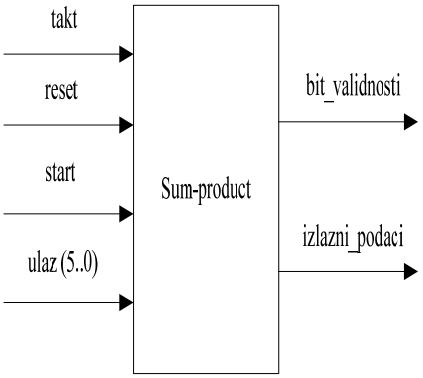
x	$\log\left\|e^{x}-1\right\|$	x	$\log\left\|e^{x}-1\right\|$
[-∞,-3)	0	[0,0.15)	$2^{4}x$ -4.0
[-3,-0.68)	$-2^{-2}x$ -0. 75	[0.15,0.4)	$2^{2}x$ -2.2
[-0.68,-0.27)	$-2x$ -1.94	[0.4,1.3)	$2x$ -1.4
[-0.27,0)	$-2^{3}x$ -3.56	[1.3,+∞)	x-0.1

III. FPGA DIZAJN SUM-PRODUCT ALGORITMA

A. Opis FPGA dizajna

U ovom odeljku opisan je FPGA dizajn *sum-poduct* algoritma realizovan u VHDL (eng. *VHSIC Hardware Description Language*) programskom jeziku. Predstavljeni dizajn je generički i nije prilagođen posebnom LDPC kodu, ali je robustan i učitavanjem željene kontrolne matrice lako se formira dekoder bilo kod LDPC koda.

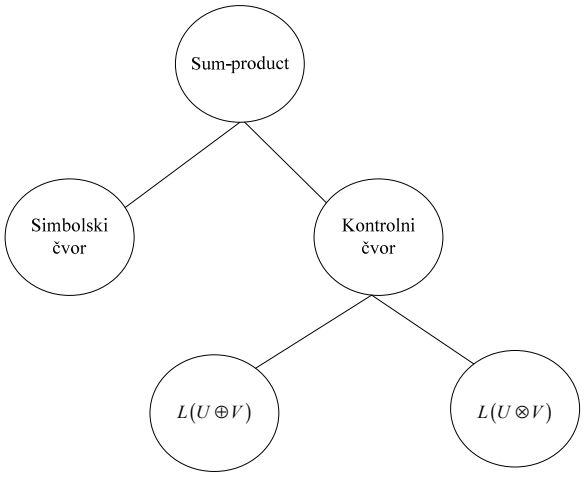
Na slici 2 prikazani su ulazni i izlazni signali iz dizajna *sum-product* dekodera. Dekoder je na svom ulazu povezan sa generatorom takta i nekim kontrolnim entitetom koji upravlja njegovim radom aktivirajući početak rada dekodera (signal *start*), njegov asinhroni reset (signal *reset*) kao i protok ulaznih poruka (magistrala *ulaz*). Ulazne vrednosti su, prema inicijalnom koraku algoritma, početne LLR vrednosti $L(u_n)$ n=1,…,N, koje se dekoderu prosleđuju serijski u 6-btinom zapisu komplementa dvojke. Izlaznim signalom *bit_validnosti* signalizira se završetak dekodovanja ulazne poruke, dok se tvrde odluke o poslatim bitima prosleđuju signalom *izlazni_podaci*.



Slika 2. Blok dijagram najvišeg nivoa dizajna.

Kako se može videti na slici 3, u najviši nivo dizajna (blok Sum-product) instancirane su komponente kontrolni čvor i simbolski čvor. Izabrana je arhitektura u kojoj se hardverski mapiraju svi čvorovi Tanerovog biparitnog grafa, tako da se optimizuje broj taktnih ciklusa izvršavanja procesa dekodovanja, na račun veće potrošnje resursa.

Tada je broj simbolskih čvorova koje je potrebno mapirati jednak broju kolona kontrolne matrice **H**, dok broj vrsta matrice **H** određuje broj kontrolnih čvorova u dizajnu. Najviši nivo dizajna takođe sadrži i logiku kojom se ostvaruju veze između pojedinih čvorova, kao i sekvencijalnu kontrolnu logiku koja upravlja tajmingom kompletnog dizajna. Veze koje se uspostavljaju između čvorova su dvosmerne, pa je potrebno osmisliti efikasan način koji garantuje jednoznačno povezivanje. Čvorovi grafa se povezuju na osnovu vektora veza koji sadrži informacije o preslikavanju pojedinih veza ka kontrolnim čvorovima u veze ka simbolskim čvorovima. Tako ako se na *i*-toj poziciji u vektoru veza nalazi broj *n*, *i*-ta i *n*-ta veza se ostvaruju između istih čvorova samo u suprotnim smerovima. Naravno, potrebno je pamtiti i broj veza koji ostvaruje svaki čvor grafa. Ako je kod regularan tada svi čvorovi jednog tipa (simbolski ili kontrolni) ostvaruju isti broj veza, što dodatno pojednostavljuje mapiranje. Vektor veza se ekstrahuje iz kontrolne matrice i zahtevna značajno manje resursa od slučaja kada bi se memorisala kompletna kontrolna matrica.



Slika 3. Različiti nivoi dizajna.

Aktiviranjem signala *start* dekoder započinje prosleđivanje inicijalnih LLR vrednosti svim simbolskim čvorovima. Kada se učita i poslednja LLR vrednost svi simbolski čvorovi primljene vrednosti sinhrono prosleđuju kontrolnim čvorovima, prema inicijalnom koraku *sum-product* algoritma. Nakon toga prelaze u stanje u kome se ne obavljaju nikakve operacije, već samo čeka kraj obrade u kontrolnim čvorovima. Broj taktova koji simbolski čvorovi provode u ovom neaktivnom stanju zavisi od vremena izvršavanja operacija u "najsporijem" kontrolnom čvoru. Kada svi simbolski čvorovi prime informacije od kontrolnih čvorova, opet sinhrono vrše ažuriranje LLR vrednosti i njihovo slanje ka kontrolnim čvorovima. U slučaju da je dostignut predviđen broj iteracija algoritma dekodovanja vrši se odlučivanje i od tekućih LLR vrednosti dobijaju se binarne procene poslatih bita. Sve poruke koje razmenjuju simbolski i kontrolni čvorovi su 6-obitne, što donekle smanjuje performanse dekodera, ali omogućava značajno jednostavniju realizaciju.

Kontrolni čvorovi su realizovani koristeći modifikaciju prikazanu u prethodnom odeljku. Svaki kontrolni čvor sadrži komponente koje obavljaju funkcije $L(U \oplus V)$ i $L(U \otimes V)$, realizovane korišćenjem lineranih aproksimativnih funkcija datih u tabelama 1 i 2. Ove funkcije prilagođene su hardverskoj realizaciji i sadrže operacije sabiranja, oduzimanja, kao i množenja i deljenja sa brojevima stepena dvojke. Množenje, odnosno deljenje sa brojem 2^n, u binarnom formatu predstavlja jednostavno pomeranje decimalne tačke za *n* mesta udesno, odnosno ulevo. Tako se relativno jednostavno mogu realizovati sve potrebne funkcije. Kako se to može primetiti na slici 1, broj komponenti koje obavljaju operaciju $L(U \otimes V)$, jednak je broju veza koje ostvaruje posmatrani čvor dok se primećuje eksponencijalna zavisnost između broja ulaza u čvor i broja operacija $L(U \oplus V)$ koje treba mapirati.

B. Analiza performansi dizajna

U ovom odelju predstavljene su performanse dizajna *sum-product* dekodera za nekoliko različitih LDPC kodova, ako se dizajn implementira na FPGA čipu kompanije *Xilinx* iz serije *Virtex 6* sa oznakom XC6VLX240T. Ovaj čip je sastavni deo razvojne platforme ML605 istog proizvođača [9].

Razmatrane su performanse dekodera nekoliko različitih LDPC kodova kratkih i srednjih dužina kodnih reči. Takođe, posmatrani kodovi pripadaju i različitim klasama tako da je kod u oznaci EG(·,·) generisan na bazi Euklidske geometrije, G(·,·) je Galagerov kod, dok oznaka QC(·,·) odgovara kvazi-cikličnim LDPC kodovima. Svi posmatrani kodovi su regularni i njihovi parametri prikazani su u tabeli III. Tako na primer, kvazi-ciklični kod QC(129,86) ima 129 simbolskih čvorova i 86 kontrolnih čvorova, koji su međusobno povezani sa 258 bidirekcionih veza.

TABELA III. PARAMETRI LDPC KODOVA.

Oznaka koda	**Parametri koda**		
	Dimenzije matrice H	**Težina vrsta**	**Težina kolone**
EG(15,7)	15×15	4	4
G(20,15)	15×20	4	3
QC(44,22)	22×44	4	2
QC(129,86)	86×129	3	2
QC(155,93)	93×155	5	3
QC(186,155)	155×186	6	5

Zauzeće resursa FPGA čipa *sum-product* dekodera izabranih LDPC kodova prikazano je u tabeli IV izraženo preko ukupnog broja logičkih elemenata čipa potrebnih za implementaciju, broja lukap tabela (LUT) i memorijskih resursa. Uočava se da broj čvorova grafa i broj veza koji treba uspostaviti direktno utiču na zauzeće resursa čipa. Tako, dekoder koda sa dužinom kodne reči od 15 bita zauzima samo 6% ukupnih logičkih jedinica čipa, dok dekoder koda dužine 129 bita zahteva 31% logičkih resursa čipa.

Primetno je i ograničenje ovog tipa potpuno paralelizovane arhitekture dekodera i *sum-product* dekoderi veoma dugih kodova (kodne reči od više hiljada bita) teško se mogu realizovani na ovaj način. Realizacija dekodera ne zahteva velike memorijske resurse i kao što je već napomenuto treba čuvati samo informacije o vektoru veza. Postignute su i zadovoljavajuće vrednosti maksimalne radne frekvencije koja se sporo smanjuje sa povećanjem veličine dekodera.

Veliki utrošak resursa čipa nadoknađuje se brzinom rada dekodera. Ovakav tip arhitekture minimizira broj taktnih intervala (ciklusa) potrebnih za dekodovanje i on se može izračunati na osnovu sledeće formule

$$N_{cik} = N + iter \cdot \left(\log_2 m + 5\right) + 2, \qquad (11)$$

gde je N dužina kodne reči koda, *iter* broj iteracija algoritma dekodovanja, a m najmanji broj oblika 2^n (n=1,2,...) ne veći od maksimalnog broja veza kontrolnog čvora. Dalje se može definisati i maksimalni protok hardverske implementacije kao mera broja dekodovanih bita u jedinici vremena na sledeći način

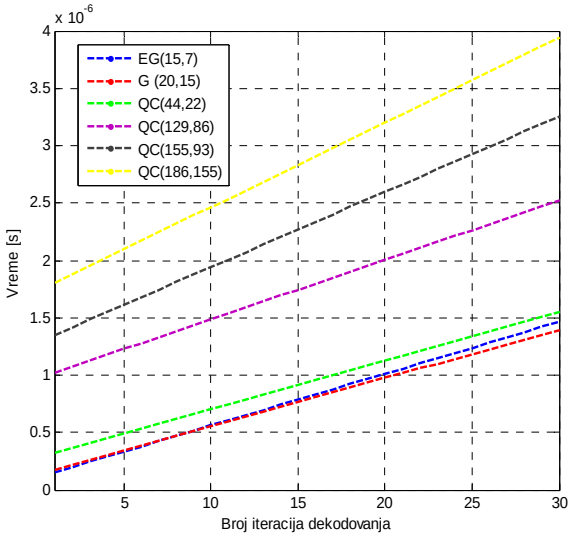
$$C_{\max} = \frac{N \cdot f_{\max}}{N_{cik}}, \qquad (12)$$

gde je sa $f_{\max}$ označena maksimalna radna frekvencija čipa.

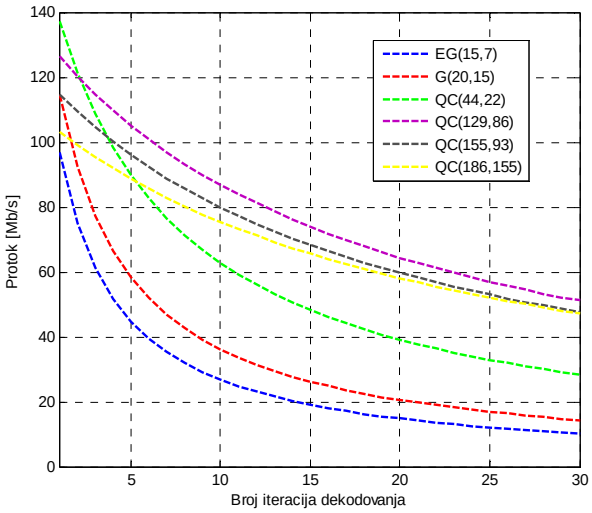
Na slikama 4 i 5 prikazane su mere performansi realizacije *sum-product* dekodera. Slika 4 sadrži prikaz zavisnosti vremena izvršavanja procesa dekodovanja od broja iteracija dekodera. Primećuje se da je vreme dekodovanja linearna funkcija broja iteracija sa približno istim nagibom za sve posmatrane kodove. Vreme dekodovanja za posmatrani opseg broja iteracija se kreće u granicama od 0,2 μs (1 iteracija EG(15,7) kod) do 4 μs (30 iteracija QC(186,155)), a male varijacije su posledica različite dužine kodnih reči posmatranih kodova.

TABELA IV. ZAUZEĆE RESURSA ČIPA VIRTEX 6 XC6VLX240T.

Oznaka koda	Parametri FPGA čipa			
	Broj log. elem.	Broj LUT	RAM [kb]	Max. frekv. [MHz]
EG(15,7)	2308(6%)	7089(4%)	0,6	155
G(20,15)	2401(6%)	7424(4%)	0,6	166
QC(44,22)	3601(9%)	11067(7%)	0,88	165
QC(129,86)	11832(31%)	34104(22%)	2,58	135
QC(155,93)	24862(65%)	71410(47%)	4,65	122
QC(186,155)	36207(96%)	125791(83%)	9,3	108



Slika 4. Zavisnost vremena dekodovanja od broja iteracija *sum-product* algoritma.



Slika 5. Zavisnost ostvarenog protoka dizajna od broja iteracija *sum-product* algoritma.

Na slici 5 prikazan je protok koji ostvaruje realizacija računata na osnovu formule (12). Na ostvareni protok vreme dekodovanja i dužina kodne reči koda imaju suprotan uticaj, pa se ne može unapred tvrditi da će dekoderi dužih kodova ostvariti veći protok. Tako, kod QC (186,155) neće ostvarivati najveći protok i na primer za slučaj da se koristi 10 iteracija približno se ostvaruje protok od 75 Mb/s, dok će za isti broj iteracija kod QC(129,86) ostvari protok od 87 Mb/s. Ipak, veoma kratki kodovi ostvaruju najniže protoke jer mala dužina kodne reči ima dominantan uticaj na protok dekodera.

IV. ZAKLJUČAK

Predstavljena hardverska implementacija *sum-product* algoritma minimizira broj taktnih intervala FPGA čipa potrebnih za dekodovanje LDPC kodova. Hardverskim mapiranjem čvorova Tanerovog grafa obezbeđuje se visok nivo paralelizma i skraćuje vreme potrebno za dekodovanje kodnih reči u odnosu na druge tipove arhitekture dekodera. Tako je na ovaj način proces dekodovanja moguće izvršiti u roku od nekoliko μs i ostvariti protok od više desetina Mb/s. Predložena arhitektura je zahtevna po pitanju resursa čipa koji su potrebni za njenu implementaciju, pa je *sum-product* dekodere veoma dugih kodova teško realizovati. Zbog toga, korišćenje predstavljene arhitekture u sistemima sa dugim okvirom za podatke, kao što je *Ethernet* standard, nije praktično. Neki drugi sistemi koji koriste kraće kodove mogu efikasno implementirati paralelnu arhitekturu prezentovanu u ovom radu.

ZAHVALNICA

Rad je finansiran sredstvima Ministarstva za nauku i tehnološki razvoj Republike Srbije, projekat TR 32028 - "Napredne tehnike za efikasno korišćenje spektra u bežičnim sistemima".

LITERATURA

[1] D. Drajić, P. Ivaniš, *Uvod u teoriju informacija i kodovanje*, 3. prošireno izd. Akademska misao, Beograd, 2009.

[2] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[3] S. Lin, D. J. Costello, Jr., *Error Control Coding – Fundamentals and Applications*, 2nd. edition, Prentice Hall, Upper Saddle River, N. J., 2004.

[4] R. M. Tanner,"A Recursive Approach to Low Complexity Codes",*IEEE Transactions on Information Theory*, vol. it-27, no.5, Sept.1981.

[5] Z. Zhang, V. Anantharam, M. Wainwright, B. Nikolić, „An Efficient 10GBASE-T Ethernet LDPC Decoder Design With Low Error Floors", *IEEE Journal of Solid-state Circuits*, vol. 45, no. 4, pp. 843-855, April 2010.

[6] J. Zhao,F. Zarkershvari, A. Banihashemi, „On Implementation of Min-Sum Algoritham and Its Modification for Decoding Low-Dencity-Parity-Check (LDPC) Codes", *IEEE Transactions on Communications*, vol 53, no. 4, pp. 549-554 April 2005.

[7] M. Petrović, A. Smiljanić, *Programiranje Alterinih FPGA čipova*, *Akademska misao*, Beograd, 2008.

[8] X. Y. Hu, E. Eleftheriou, D. M. Arnold, A. Dholakia, „Efficient Implementation of Sum-Product Algorithm for Decoding LDPC Codes", *in Proc. of GLOBECOM 2001*, vol. 2, pp. 1036-1036E, 2001.

[9] Xilinx Inc, *FPGA family data sheet*, (pristupljeno: januar 2013) http://www.xilinx.com/support/documentation/data_sheets/ds610.pdf

ABSTRACT

In this paper we present an FPGA (Field Programmable Gate Array) design of the sum-product algorithm that is used for decoding LDPC (Low Dencity Parity Check) codes. The proposed design minimizes the number of clock interval required for the decoding process, thus the codewords can be decoded within a few μs, achieving a decoding throughput of tens of Mb/s.

FPGA IMPLEMENTATION OF SUM-PRODUCT ALGORITHM FOR DECODING LDPC CODES

Srdjan S. Brkic, Dajana M. Lazarevic, Predrag N. Ivanis