SOFTVERSKA IMPLEMENTACIJA DEKODOVANJA KASKADNIH KODOVA VITERBIJEVIM ALGORITMOM

Žarko Vitomir, *Aerodrom Beograd*
Predrag Ivaniš, *Elektrotehnički fakultet u Beogradu*

***Sadržaj* – Poznato je da kaskadni kodovi omogućuju znatno sniženje zaostale vjerovatnoće greške za male vrijednosti odnosa signal/šum, što je teško postići jednim zaštitnim kodom. U ovom radu je razvijen fleksibilni simulator komunikacionog sistema i njegovim korišćenjem je izvršeno određivanje performansi kaskadne veze Hemingovog i konvolucionog koda. Dekodovanje konvolucionog koda izvršeno korišćenjem Viterbijevog algoritma. Razvijena aplikacija može poslužiti i kao nastavno sredstvo: Viterbijev algoritam se grafički izlaže korak po korak, prikazujući određivanje metrike i dekodovanje odgovarajućih bita.**

***Ključne reči* - Kaskadni kodovi, konvolucioni kodovi, softverska implementacija, Viterbijev algoritam**

I. UVOD

U ovom radu opisana je softverska implementacija dekodera jedne klase kaskadnih kodova, nastalih povezivanjem konvolucionog i Hemingovog blok koda. Konvolucioni kodovi pokazuju "dobre" osobine po pitanju zaštite informacija, što je omogućeno unosom statističke zavisnosti između kodnih riječi koje se šalju, tj. povećanjem memorije kodera uz zadržavanje istog kodnog količnika. Ovaj dobitak je plaćen usložnjavanjem procesa dekodovanja, koje se u ovom radu obavlja pomoću Viterbijevog algoritma [1]. Međutim, ni konvolucioni kodovi nisu omogućili pouzdan prenos za slučaj malog odnosa signal/šum u kanalu. Performanse se mogu dodatno poboljšati njihovim kaskadnim vezivanjem sa blok kodovima. U ovom radu u tu svrhu je upotrebljen Hemingov blok kod [2], koji nema značajne korekcione sposobnosti, ali je upotrebljen kao nastavno sredstvo radi ilustracije značaja kaskadne veze.

Aplikacija koja je razvijena omogućuje softversku simulaciju prethodno opisane kaskadne veze. Realizacija je izvršena prema blok šemi sistema sa stanovišta Teorije informacija [3], pa je navedeni primjer samo jedan od scenarija koji se mogu analizirati. Omogućeno je kaskadno povezivanje velikog broja blokova i određivanje performansi u zavisnosti od unetih parametara. Pored konvolucionih kodova sa Viterbijevim dekoderom i Hemingovog blok koda, dodati su i izvor (sa ili bez memorije [4,5]), statistički koder zasnovan na Hafmenovom algoritmu [4] i binarni diskretni kanal [4,5] ili Gilbert-Eliotov kanal sa memorijom [4]. Omogućeno je da kroz nekoliko jednostavnih koraka korisnik izvrši pripremu komunikacionog sistema za simulaciju, unese parametre i sačuva dobijene rezultate.

Ž. Vitomir, Aerodrom Beograd, Beograd, Srbija (e-mail: *zvitomir@yahoo.com*).

P. Ivaniš, Elektrotehnički fakultet, Beograd, Srbija (e-mail: *predrag.ivanis@etf.rs*).

Takođe je dodata i funkcionalnost koja omogućuje da se aplikacija primjeni i u nastavne svrhe: Viterbijev algoritam dekodovanja se izlaže korak po korak omogućujući korisniku da stekne uvid u proces odabiranja, brisanja grana duž trelisa i određivanja poslate sekvence.

II. SIMULACIONI MODEL

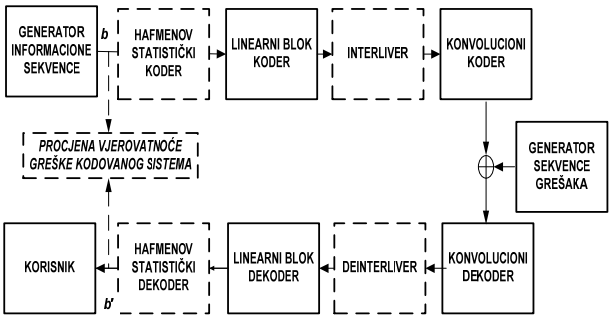
Metode Monte Karlo simulacije [6] predstavljaju alat kojima je moguće izvršiti preciznu procjenu parametara nekog sistema. U stvari, to je skup simulacionih postupaka koji za cilj imaju određivanje veličina koje opisuju ponašanje sistema i to u slučaju kada je poznato kako se ponašaju komponente koje čine sistem, ali ponašanje sistema kao cjeline nije dovoljno ispitano.

Nakon generisanja informacione sekvence i sekvence grešaka kojom je opisan diskretni kanal, određivanje performansi primjenjenih zaštitnih kodova izvodi se na osnovu postupka opisanog u [4] i prikazanog na Slici 1. Monte Karlo simulacija u ovom slučaju počinje tako što se se iz generisane informacione sekvence b izdvoji potreban broj bita za formiranje kodne riječi linearnog blok koda, nakon čega se dobijena sekvenca uvodi u konvolucioni koder. Tako dobijena sekvenca se vodi na ulaz diskretnog kanala i na nju se superponira vektor greške iste dužine (sabiranjem po modulu 2). Na prijemu, riječ sa izlaza diskretnog kanala se vodi na zaštitni dekoder. Na njegovom izlazu dobija se estimirana informaciona sekvenca b'. U cilju određivanja vjerovatnoće zaostale greške, vrši se poređenje informacione sekvence na ulazu u zaštitni koder, i primljene sekvence na izlazu zaštitnog dekodera.

Zaostala ili rezidualna vjerovatnoća greške se može izračunati pomoću izraza

$$\hat{P}_e = sum(b \oplus b')/N, \qquad (1)$$

gdje je operator $\oplus$ sabiranje po modulu 2 a N-dužina poslate i primljene sekvence bita.



Slika 1: Blok šema simulacionog modela

Prikazani komunikacioni sistem ima u osnovi dvije vrste komponenti: determinističke (koder izvora i koder kanala) i nedeterminističke (izvor i diskretni kanal). Kod determinističkih komponenti, preslikavanje ulaznih promjenljivih u izlazne je dato po nekom zakonu i nepromjenljivo je. Za razliku od njih, nedeterminističke komponente nemaju strogo definisano ponašanje, već je njihov rad probabilistički i modeluje se pomoću sekvence simbola (izvor), odnosno sekvence grešaka (kanal). Modelovanje ovih sekvenci je takvo da se poštuju stacionarne osobine realnog izvora/kanala, pri čemu za svaki model, jedna sekvenca treba da bude pripadnik odgovarajućeg ansambla sekveci koji opisuju realni izvor/kanal.

Jasno je da dužina sekvenci direktno utiče na preciznost procenjenih vrijednosti zaostale vjerovatnoće greške. Zato je potrebno odrediti veličinu uzorka za Monte Karlo simulaciju pa da procjena vjerovatnoće greške s dovoljno velikom vjerovatnoćom bude unutar intervala (y^-, y^+). Ovaj problem je kompletno riješen u članku Jeruhima [6]. Ukoliko je sa $\hat{P}_e$ označena procjenjena zaostala vjerovatnoća greške, potreban broj prenetih simbola je

$$N = \eta \cdot (1/\hat{P}_e) \qquad (2)$$

pri čemu vrijednost η određuje interval pouzdanosti, za zadati nivo pouzdanosti, $1-\alpha$. U ovom radu, uzete su vrijednosti $\eta = 100$ i $1-\alpha = 0.9$, pa su dobijeni rezultati sa vjerovatnoćom od 90% u intervalu $(0.85\hat{P}_e, 1.2\hat{P}_e)$ što je dobar kompromis između potrebnog vremena za simulaciju i procjenjenog odstupanja od dobijene vrijednosti.

III. Opis Aplikacije

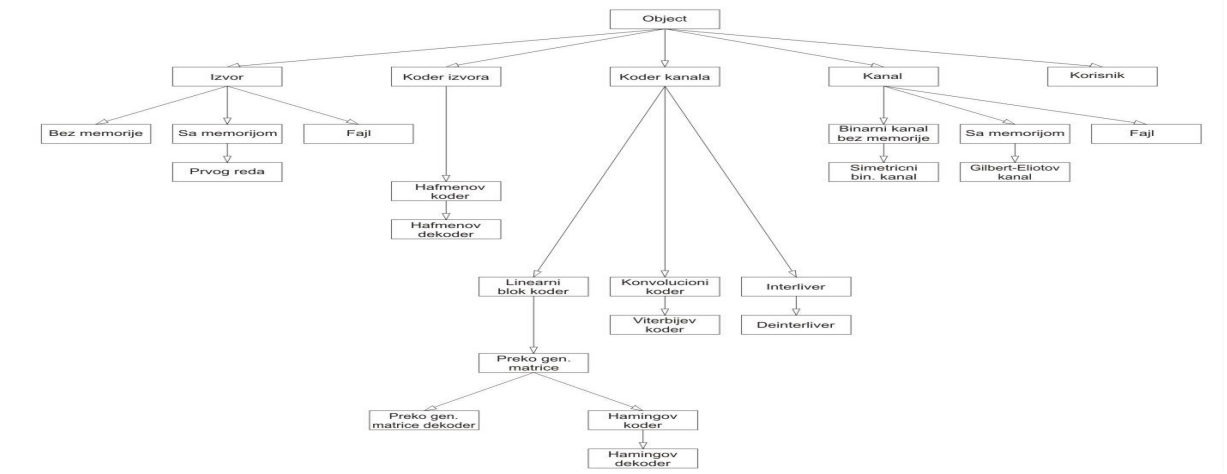
Aplikacija je razvijena u programskom jeziku kompanije Borland: CodeGear™ C++Builder®Version 11.0.2709.7128 Copyright © 2007 [7]. Ovaj programski jezik je objektno orijentisan sa vizuelnim okruženjem koje omogućuje lako kreiranje Windows aplikacija i to pod programskim jezikom „C++“ [8,9].

Polazeći od činjenice da je jedan blok osnovna jedinica, razvijena je klasa Object, koja predstavlja njegovu apstrakciju. Prema terminima jezika C++, definisana klasa je apstraktna, sa deklarisanim virtuelnim metodima koje svaka klasa „potomak“ treba da redefiniše prema svojoj ulozi. Najznačajnija metoda ove klase - Tick(), nije virtuelna ali zato određuje redosled i način pozivanja virtuelnih metoda. Upravo je ona zadužena za prihvatanje simbola, njihovu obradu shodno funkciji bloka i prosleđivanje simbola narednom bloku za definisani scenario komunikacije. Prikaz svih potomaka klase Object dat je na slici 3.

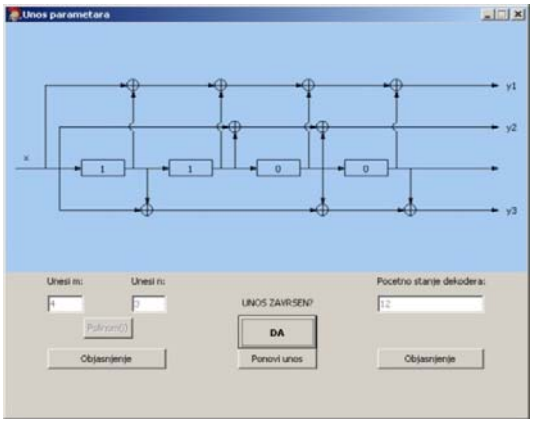
Klasa Clock je sledeća klasa po važnosti. Ona objedinjuje sve blokove (entitete) u jedan povezan i uređen sistem. Osim toga, vrši pokretanje simulacije za određen broj simbola koje treba prenijeti. Specifično rukovođenje i obrada podataka potomaka klase Object je uslovilo i specifičan način za mjerenje vjerovatnoće greške u sistemu, bilo u kanalu, bilo rezidualne vjerovatnoće greške. Ovo su samo osnovne funkcije koje izvršava klasa Clock.

Međutim, kompletan proces simulacije prikriven je za korisnika. Izvršavanje simulacije je u pozadini, pomoću „niti“, klase koja je pod direktnim nadzorom operativnog sistema, sa mogućnošću definisanja određenog koda koji može da se izvršava i da se konkurentski izbori sa ostalim procesima za potrebno vrijeme procesora. Vizuelni dio aplikacije je urađen zahvaljujući prikazu preko formi, interakcijom sa korisnikom preko odgovarajućih dijaloga za unos kao i za obavještenja.

Korišćenje aplikacije je veoma jednostavno i u nekoliko koraka sistem se može pripremiti za simulaciju. Po otvaranju, kreira se novi projekat ili se otvori već postojeći. Sledeći korak je definisanje komunikacionog scenarija koji se želi simulirati. Glavni meni nudi više opcija za jednu komponentu (izvor, statistički koder, koder kanala, kanal), i pored izvora i statističkog kodera, može ih se dodati više iako su istog tipa. Po odabiru, komponenta se pojavljuje na radnoj površini u skladu sa organizacijom sistema. Svaka komponenta iz glavnog menija, ima definisanog „parnjaka“, tj. odgovarajuću komponentu na prijemnoj strani, čija je funkcija inverzna onoj na predaji.



Slika 3: Prikaz izvedenih klasa iz klase *Object*



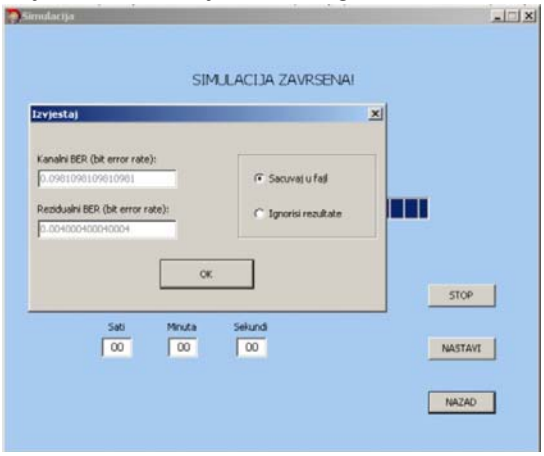
Slika 4: Prozor za unos parametara konvolucionog koda

Pomoću opcije iskačućeg menija, svaka od komponenti definiše nekoliko operacija koje se mogu izvršiti nad njom. Osnovna operacija je unos parametara kojom se definiše njen rad. Ova procedura se razlikuje od komponente do komponente, može da bude veoma jednostavna i da se izvede pomoću nekoliko uzastopnih poziva dijaloga za unos podataka (za celobrojni, realni i tekstualni tip), a može da bude i dovoljno komplikovana da se koristi poseban prozor za to. Jedna od karakteristika razvijene aplikacije je udobnost, zapravo veoma softificiran metod kojim korisnik bez mnogo dvoumljenja zna šta treba da uradi. Kao primjer, na slici 4. dat je prikaz prozora za unos parametara kod konvolucionog kodera. Ukoliko korisniku nije poznata funkcija polja za unos podataka, pritiskom na dugme „Objasnjenje“ se razrešavaju nedoumice. Po unosu parametara, iscrtava se šema kodera sa odgovarajućim brojem ulaza, izlaza, memorijskih registara, kao i prikazom za dobijanje pojedinog izlaza preko sabirača.

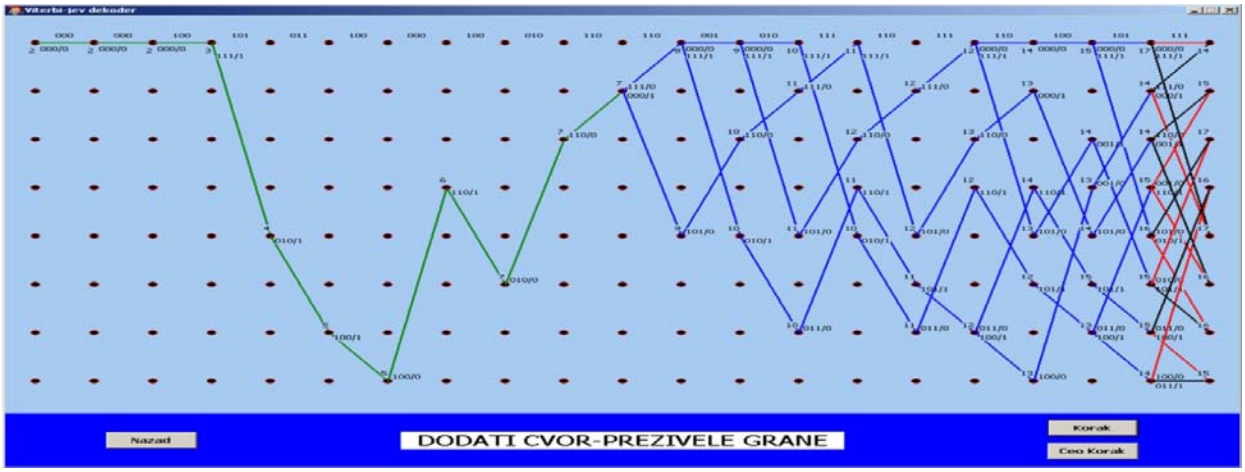
Sledeća operacija je mogućnost internog prikaza komponente. Pod internim prikazom se smatra uvid u vrijednost važnijih parametara klase koja definiše komponentu i njeno ponašanje po prijemu simbola, kroz korake, dok se ne proslijedi izlazu. Trenutno nije definisana za sve komponente komunikacionog sistema, već isključivo za konvolucioni koder i njegovu funkciju na predaji kao i na prijemu (Viterbijev algoritam dekodovanja).

Opcija postaje dostupna po definisanju parametara komponente i ne zavisi od ostatka sistema. Sasvim je korektno dodati samo jednu komponentu na radnu površinu, zadati parametare i uću u interni prikaz. Zahvaljujući ovoj mogućnosti, aplikacija se može primjeniti i u nastavne svrhe u cilju razumjevanja rada komponenti komunikacionog sistema. Za Viterbijev algoritam dekodovanja, prikazan je trelis u trenutku dodavanja novog čvora i izbora preživelih grana, slika 5. Poslednja opcija iskačućeg menija je brisanje komponente kao posledica loše izabranog scenarija komunikacije.

Nakon unetih parametara, pokreće se simulacija za izvestan broj simbola koje treba prenijeti. Ovaj broj se može definisati, ali je daleko pouzdanije ostaviti aplikaciji da automatski odredi rezultate sa definisanom pouzdanošću (zavisi od rezidualne vjerovatnoće greške u kanalu). Aplikacija ne dozvoljava unos nekorektnih vrijednosti, bilo da je to dodavanje komponenti, zadavanje parametara, pokretanje simulacije a postoji i mogućnost otklanjanja nekih logičkih grešaka. Po regularnom završetku simulacije, otvara se prozor sa izvještajem, slika 6. Prikazan je rezultat za jednu tačku. Takođe, postoji način kojim se definiše pokretanje simulacije za više tačaka, pri čemu se zadaju početna i krajnja vrijednost odnosa signal/šum u kanalu ili odgovarajući interval za vjerovatnoću greške.



Slika 6: Prikaz rezultata simulacije



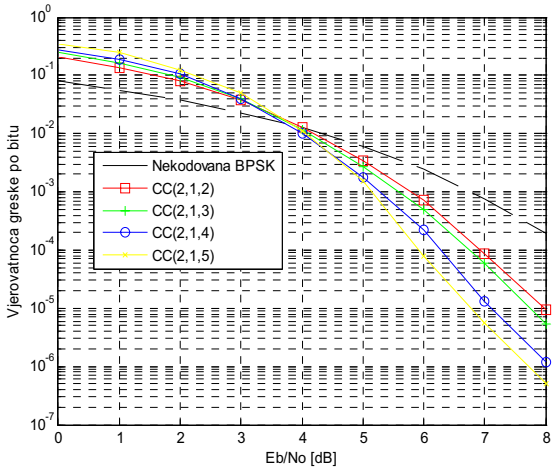
Slika 5: Interni prikaz Viterbijevog postupka dekodovanja

IV. NUMERIČKI REZULTATI

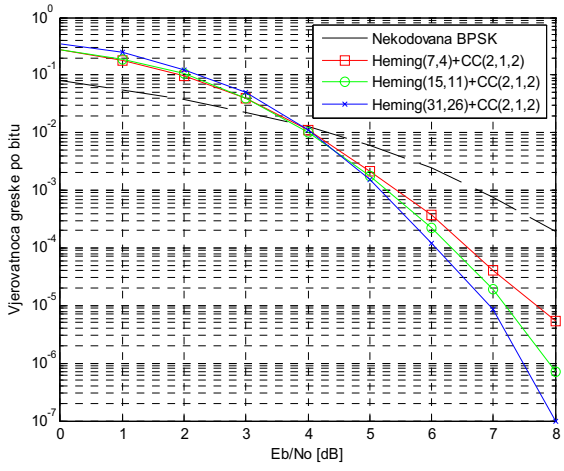
Koristeći se aplikacijom, procjenjena je rezidualna vjerovatnoća greške u funkciji odnosa energije po bitu i spektralne gustine snage šuma. Posmatrana su dva komunikaciona scenarija, a kod oba su korišćeni binarni izvor bez memorije (sa jednakim vjerovatnoćama pojavljivanja simbola) i binarni simetrični kanal. Kod prvog komunikacionog scenarija, izvršeno je poređenje konvolucionih kodova za konstantne vrijednosti ulaza i izlaza (k+1, n+2), i za promjenljivu veličinu registra kodera. Rezultati su prikazani na slici 7, u funkciji odnosa energije informacionog bita prema snazi šuma, prema jednačini

$$\left[E_b / N_0\right]_{info} = \left[E_b / N_0\right]_{kanal} - 10 \cdot \log(1/R) \qquad (3)$$

Dobijeni rezultati su očekivani, jer veće m znači i manju rezidualnu vjerovatnoću greške pri istom odnosu E_b / N_0 u kanalu. Tako je za $m = 2$ i zaostalu vjerovatnoću greške od $P_{e,rez} = 10^{-4}$, kodni dobitak u odnosu na nekodovan slučaj oko $1.25\,dB$. Primjena koda CC(2,1,2) omogućuje da se i pri manjem odnosu signal/šum u kanalu mogu ostvariti identične performanse po pitanju vjerovatnoće zaostale greške u odnosu na nekodovan sistem.



Slika 7: Grafički prikaz rezultata, prvi scenario



Slika 8: Grafički prikaz rezultata, drugi scenario

Drugi scenario predstavlja kaskadnu vezu Hemingovog i konvolucionog (2,1,2) kodera, pri čemu su mijenjani parametri Hemingovog kodera: (7,4), (15,11), (31,26). Prikaz dobijenih rezultata je na slici 8. Ostvareni kodni dobitak za rezidualnu vjerovatnoću greške od $P_{e,rez} = 10^{-4}$, u odnosu na nekodovani slučaj je za krivu Heming(7,4)+CC(2,1,2) oko $1.7\,dB$.

V. ZAKLJUČAK

U ovom radu je opisana razvijena aplikacija koja omogućava procjenu performansi komunikacionog sistema Monte Karlo simulacijom, za zahtjevani nivo pouzdanosti. Pored toga, dodata je funkcionalnost vizuelnog prikaza strukture kodera i procesa dekodovanja korišćenjem Viterbijevog algoritma. Pomoću opisane aplikacije, procjenjene su performanse sistema sa primjenjenom kaskadnom vezom konvolucionog i Hemingovogm koda.

Pokazano je da se pomoću kaskadne veze ostvaruju bolje performanse po pitanju korekcije grešaka i kodnog dobitka u odnosu na upotrebu samo jednog koda, ali na račun veće složenosti sistema i kašnjenja. Odluka o primjenjenom scenariju je stvar kompromisa koji se moraju napraviti.

LITERATURA

[1] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Trans. Inform. Theory*, Vol. IT-13 (1967), pp. 260-269

[2] R. W. Hamming, "Error Detecting and Error Correcting Codes", *Bell Sys. Tech J.*, Vol. 29 (1950), pp. 147-160

[3] C. E. Shannon, "A Mathematical Theory of Communication", *Bell Sys. Tech. J.*, Vol. 27 (1948), pp. 379-423, 623-656

[4] D. Drajić, P. Ivaniš, Uvod u teoriju informacija i kodovanje, Akademska misao, Beograd, 2009.

[5] S. Lin, D. J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice Hall, 2004.

[6] M. C. Jeruchim, "Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communications Systems", *IEEE J. Sel. Areas Comm*., Vol. 2, 1984, pp. 153-170

[7] Web strana http://www.yevol.com/en/bcb/

[8] L. Kraus, "Programski jezik C", Akademska misao, Beograd, 2004

[9] D. Milićev, "Objektno orijentisano programiranje na jeziku C++", Mikro knjiga, Beograd, 1995

ABSTRACT

In this paper, the flexible simulator for the communication system is developed, and using it, comparing performance between cascade decay of Hamming and convolutional code with performance of single code is done. In the convolutional code decoding, the Viterbi algorithm is applied. Beside that, developed application can be useful as a teaching method: Viterbi algorithm is showing graphically step by step, moving trough the trellis, measuring metrics and decoding corresponding bits.

SOFTWARE IMPLEMENTATION OF CONCATENATED CODES DECODING USING VITERBI ALGORITHM

Žarko Vitomir, Faculty of Electrical Engineering Belgrade
Predrag Ivaniš, Faculty of Electrical Engineering Belgrade