SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4239

SKLOPOVSKA IZVEDBA ROBUSNE ESTIMACIJE OČEKIVANJA I VARIJANCE

Ivan Laznibat

Zagreb, lipanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 13. ožujka 2015.

ZAVRŠNI ZADATAK br. 4239

Pristupnik: **Ivan Laznibat (0036468165)**
Studij: Elektrotehnika i informacijska tehnologija
Modul: Elektroničko i računalno inženjerstvo

Zadatak: **Sklopovska izvedba robusne estimacija očekivanja i varijance**

Opis zadatka:

U okviru završnog rada potrebno je istražiti robusne postupke estimacije očekivanja i varijance ulaznog procesa iz njegovih realizacija s konačnim brojem uzoraka. Postupak treba biti neosjetljiv na iznimke u slučaju kada su ulazni uzorci željene distribucije pomiješani s uzorcima neželjene distribucije koja generira iznimke. Posebnu pažnju posvetiti statistici uređenih nizova (engl. order statistics) i njenoj mogućoj primjeni. Izraditi referenti model u Matlabu i verificirati točnost rada na sintetičkim primjerima. Razviti ponašajne (engl. behavioral) modele implementacije algoritma u VHDL jeziku, te istražiti mogućnosti sklopovske izvedbe korištenjem FPGA sklopova. Verificirati točnost rada usporedbom s referentnim modelom u Matlabu i diskutirati mogućnosti primjene takvog estimatora na vremenske serije.

Zadatak uručen pristupniku: 13. ožujka 2015.
Rok za predaju rada: 12. lipnja 2015.

Mentor:
Prof. dr. sc. Davor Petrinović

Djelovođa:
Izv. prof. dr. sc. Dražen Jurišić

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Mladen Vučić

Sadržaj

Uvod

Svaka analiza signala se temelji na skupu uzoraka koji se obrađuju. Točnost tih uzoraka ima posljedicu na valjanost rezultata, a budući da nerijetko u stvarnim situacijama postoje kontaminirani uzorci koje nije jednostavno uočit i ukloniti, bitno je razviti statističke metode koje će i u tim situacijama pružati dobre rezultate. To je bila jedna od glavnih motivacija u razvoju robusne statistike, odnosno proučavanje metoda kako umanjiti utjecaj uzoraka čije se vrijednosti previše ističu (eng. outliears) jer zbog njih u praksi dolazi do nevjerodostojnih statističkih analiza. Robusna estimacija očekivanja i varijance ima mnoge primjene u praksi (obrada signala i informacija, telekomunikacije, financije, razvoj znanosti,...) jer skupovi uzoraka koji se trebaju analizirati često sadrže „stršeće“ vrijednosti koje ako ih se ne izuzme iz analize mogu dati pogresne rezultate.

1. Metode u robusnoj statistici

.

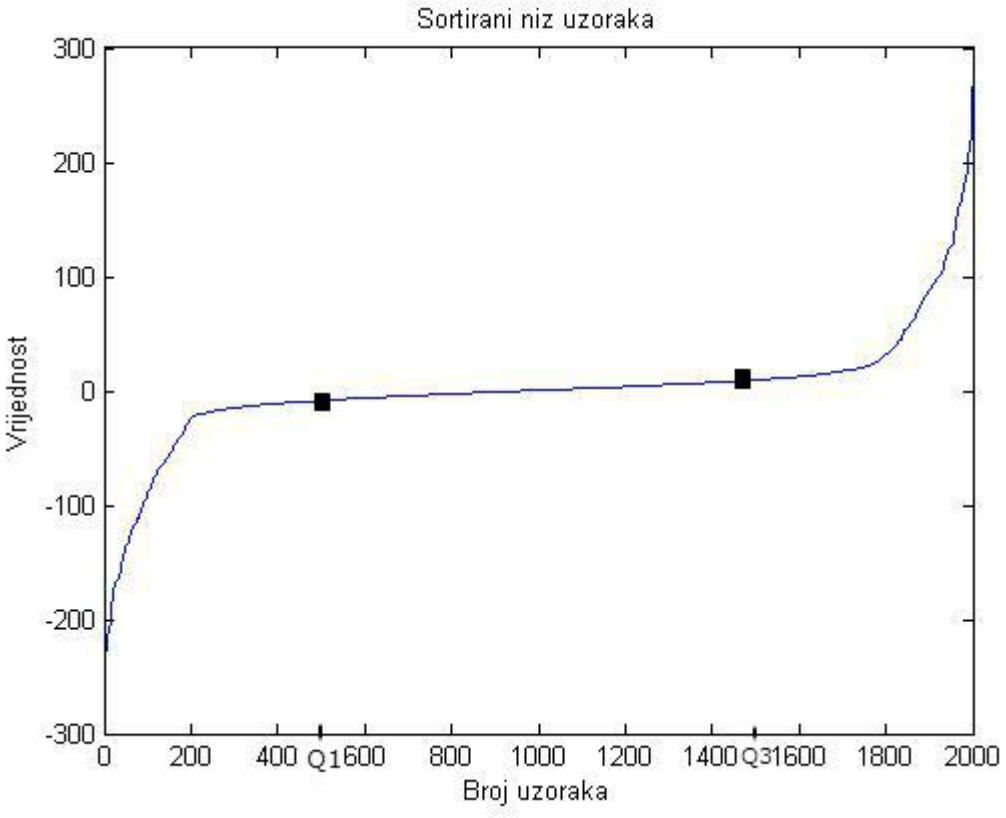
Kako bi se riješio problem utjecaja stršećih vrijednosti (eng. outliers) na rezultat razvijene su nove statističke metode i procjenitelji koji ne podliježu utjecaju stršećih vrijednosti. To su robusne metode koje pružaju dobre rezultate čak i ako je veća količina uzoraka kontaminirana. Slijedi nekoliko primjera dobrih i loših metoda u robusnoj statistici:

Medijan predstavlja broj koji dijeli uzorke na dvije polovice. Medijan konačnog niza brojeva se može pronaći sortiranjem uzoraka i biranjem središnjeg uzorka. Ukoliko je paran broj uzoraka, medijan je obično definiran kao aritmetička sredina dva srednja broja Za razliku od aritmetičke sredine medijan ne ovisi o uzorcima koji su mnogo veći ili manji od srednje vrijednosti, pa je time dobar alat u robusnoj statistici.

Interkvartilni raspon (IQR) predstavlja mjeru statističke disperzije, a jednak je razlici između gornjeg (Q_3) i donjeg kvartila (Q_1). Kvarlili predstavljaju vrijednosti tri uzorka koji dijele skup podataka u četiri ekvivalentne grupe. Prvi kvartil Q_1 predstavlja vrijednost srednjeg uzorka izmedu izmedu najmanjeg broja i medijana, dok treći kvartil Q_3 predstavlja vrijednost srednjeg uzorka izmedu medijana i najvećeg uzorka. Drugi kvartil Q_2 je medijan. Na slici 1. je prikazan 1. i 3. kvartil sortiranog skupa N uzoraka (N=2000).

IQR se koristi u robusnoj statistici kao mjera koliko se uzorci rasprostiru oko centralne vrijednosti. IQR je proporcionalan rasprostiranju uzoraka. Zajedno s drugim mjerilima kao sto su medijan i totalni raspon daje cjelovitu sliku o rasprostiranju podataka oko medijana.

$$IQR = Q_3 - Q_1$$



. Slika 1.

Osim za izračun robusne varijance, IQR se moze koristiti za pronalaženje „stršećih“vrijednosti. Vrijedi da su „stršeći“ uzorci svi oni koji su manji od $Q_1 - 1.5 * IQR$ ili veći od $Q_3 + 1.5 * IQR$.

Medijan apsolutne devijacije (MAD) je mjera statističke disperzije. Ova mjera je precizna statistička metoda za analizu uzoraka zbog veće otpornosti na udaljene uzorke. Najbolji način za shvatiti MAD je razumjeti metodu njegovog izračuna. Na početku je potrebno izračunati medijan skupa uzoraka. Nakon toga treba odrediti koliko je svaka vrijednost uzorka udaljena od medijana. Bez obzira je li vrijednost manja ili veća od medijana, ona se izražava kao apsolutna vrijednost. Posljedni korak je pronalazak medijana od skupa novih vrijednosti.

$$MAD = medijan\left(\left| X - medijan\left(X\right) \right|\right)$$

X - skup uzoraka

Robusna estimacija očekivanja može se izraziti kao medijan skupa uzoraka.

Robusna estimacija varijance predstavlja matematičku vrijednost odstupanja skupa uzoraka od medijana, ali uz zanemarivanje onih koji se previše ističu. Za izračun se može koristiti na nekoliko načina:

- Pomoću interkvartilnog raspona (IQR):

$$\delta_2 = \left(20 \div 27\right) * IQR$$

- Pomoću medijana apsolutne devijacije (MAD):

$$\delta_2 = 1.4826 * MAD$$

Aritmetička sredina:

$$\mathrm{A} = \frac{1}{N}\sum_{i=1}^{N} a_i$$

Standardna devijacija je mjera u statistici kojom se izražava mjera disperzije skupa podataka. Mala standardna devijacija ukazuje da uzorci imaju tendenciju biti blizu srednjoj vrijednosti, dok velika standardna devijacija ukazuje da se uzorci rasprostiru na širom opsegu vrijednosti.

$$\delta = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - A\right)^2}$$

N- broj uzoraka

A- aritmetička sredina

Aritmetička sredina i standardna devijacija spadaju u loše metode u robusnoj statistici jer na njih snažno utječu vrijednosti uzoraka koji su mnogo veći ili manji od većine vrijednosti.

.

2. Izvedba u C-u

2.1. Općenito o algoritmima za pronalazak k-tog najmanjeg broja

Za izračun robusne estimacije očekivanja i varijance potrebno je izraditi algoritam koji može u razumno kratkom vremenu pronaći točno određene uzorke. Estimacija varijance biti će izračunata na dva načina: pomoću interkvartilnog raspona (IQR) i uz medijan apsolutne devijacije (MAD), te će se usrednjavanjem ta dva rezultata dobiti preciznija vrijednost varijance. Estimacija očekivanja jednaka je vrijednosti medijana (Q_2) skupa uzoraka.

Konceptualno najjednostavnije, ovaj se problem može riješiti sortiranjem skupa uzoraka od najmanjeg do najveceg i uzimanjem onih uzoraka koji su potrebni za izračun. Budući da je za izračun potrebno samo četiri uzorka (Q_1, medijan, Q_3, MAD) nije efikasno sortirati cijeli skup. Bolje rješenje je koristiti određeni algoritam koji će u skupu uzoraka pronaći samo one koji su potrebni. U računalnoj znanosti, takvi algoritmi se nazivaju „selection“ algoritmi. Oni pronalaze k-ti najmanji broj u polju uzoraka. U toj kategoriji najpoznatiji je „quickselect“ algoritam. Ovaj algoritam osmislio je Tony Hoare, a u literaturi se moze pronaći i pod nazivom „Hoare's selection algorithm“. Iako „Quickselect“ algoritam ima vremensku složenost u najgorem slucaju O(n^2), dobrim izborom pivota može se postići prosječna složenost O(n). Upravo zbog toga „quickselect“ algoritam i njegove inačice danas se najviše koriste u praksi.

2.2 Medijan od medijana

Općenito vrijedi: najjednostavniji slučaj je pronalaženje najmanje ili najveće vrijednosti (jedno spremanje vrijednosti), dok je najkompleksnije pronalaženje medijana (nužno je n/2 spremanja vrijednosti). Upravo iz tog razloga razvijene su posebne inačice „quickselect“ algoritama čiji je zadatak brzo pronalaženje medijana. Jedan od najpoznatijih iz te skupine naziva se medijan od medijana koji se uz neke preinake proučava se u ovom radu. Medijan od medijana je rekurzivni algoritam, najgore vremenske složenosti O(n) za traženje medijana. Algoritam je objašnjen u nastavku.

Osim glavne (eng. main) funkcije za analizu algoritma, korištene su ove funkcije:

```
void sortiraj(float* polje,int n);
float median_of_medians(float* polje,int k,int n);
```

Funkcija za sortiranje je jednostavan „ selection sort” najgore vremenske slozenosti O(n^2). Budući da je broj uzoraka n=5 (osim za posljednju iteraciju kada može biti najviše 10 uzoraka), nije potrebno implementirati kvalitetan algoritam jer se ne bi dobilo značajnije ubrzanje programa.

```
void sortiraj(float* polje,int n){
    int i,j;
    float pom;
    for(i=0;i<n;i++){
            pom=*(polje+i);
            for(j=i+1;j<n;j++)
                if(pom>*(polje+j)){
                    *(polje+i)=*(polje+j);
                    *(polje+j)=pom;
                    pom=*(polje+i);
                }
        }
}
```

Deklaracija funkcije algoritma Medijana od medijana:

```
float median_of_medians(float* polje,int k,int n);
```

1. Tablica varijabli u funkciji median_of_medians

# define N	definirana vrijednost duljine polja
float * polje	pokazivač na prvi član ulaznog polja
int n	veličina ulaznog polja
int k	pozicija k-tog člana ulaznog polja
float M	vrijednost k-tog člana polja (rezultat)
float potpolje[N][5]	polje koje u svakom retku ima 5 članova
float x[N]	polje u koje se spremaju vrijednosti medijana potpolja[N][5]
float polje_m[N]	polje u koje se spremaju vrijednosti ulaznog polja koja su manja od M
int n_m	veličina polja: polje_m[N]
float polje_v[N]	polje u koje se spremaju vrijednosti ulaznog polja koja su veća od M
int n_v	veličina polja: polje_v[N]
int i,j	pomoćne varijable

Princip rada algoritma Medijan od medijana može se prikazati u nekoliko koraka.

1. Jednodimenzionalno ulazno polje se dijeli u potpolja od pet članova, U C-u se to realizira pretvaranjem ulaznog polja u dvodimenzionalno polje koje će u svakom retku imati 5 članova.

```
for(i=0;i<n/5;i++)
        for(j=0;j<5;j++)
            potpolje[i][j]=*(polje+i*5+j);
```

2. Za svako potpolje od 5 članova potrebno je izračunati medijan (k=2) što se postiže rekurzivnim pozivom funkcije.

```
for(i=0;i<n/5;i++)
        x[i]=median_of_medians(&potpolje[i][5],2,5);
```

3. Ovaj dio koda osigurava povratak iz rekurzije ukoliko je duljina polja manja od 11. Vrši se sortiranje polja i vraća se vrijednost k-tog elementa.

```
if(n<=10){
        sortiraj(polje,n);
        return polje[k];
    }
```

4. Sljedeći korak je pronalazak medijana od skupa medijana potpolja iz tocke 2.

```
M=median_of_medians(&x[0],n/10,n/5);
```

5. Nakon što je u prethodnom koraku izračunat medijan od medijana M on se uspoređuje sa svim elementima ulaznog polja, te ovisno o tome li jesu li veći ili manji, pune dva nova polja.

```
for(i=0;i<n;i++){
        if(polje[i]<M){
            polje_m[n_m]=polje[i];
            n_m++;
        }
        else if(polje[i]>M){
            polje_v[n_v]=polje[i];
            n_v++;
        }
    }
```

6. Ovisno o traženom broju k i veličini manjeg polja (n_m), rekurzivna funkcija se ponavlja na neki od sljedećih načina:

```
if (k<n_m)
        return median_of_medians(polje_m,k,n_m);
    else if (k > (n_m+1))
    return median_of_medians(polje_v,k-n_m-1,n_v);
    else
        return M;
```

Algoritam Medijan od medijana zbog svoje dobre O-notacije predstavlja kvalitetan „selection" algoritam za implementaciju u C-u, ali zbog rekurzivnosti potrebna je velika memorija, što je njegov nedostatak i nije dobro (jednostavno) rješenje za sklopovsku realizaciju. Upravo radi toga u ovom radu je analiziran još jedan „selection" algoritam.

2.3 Wirthov algoritam za brzo pronalaženje medijana

Wirthov algoritam osmislio je švedski računalni znanstvenik Niklaus Wirth, po kome je algoritam i dobio ime. Algoritam ne pokušava sortirati cijelo ulazno polje, već ga pregledava sve dok nije siguran da je pronašao k-ti najmanji element. Njegova najveća prednost je to što nije rekurzivan (dobro rješenje za sklopovovsku implementaciju), a cijena za to je da se inicijalno ulazno polje mora kopirati jer se modificira tijekom izvođenja algoritma.

Tablica 2. Opis varijabli u funkciji k_ti_najmanji

float * polje	pokazivač na prvi član ulaznog polja
int n	veličina ulaznog polja
int k	pozicija k-tog člana ulaznog polja
float x	vrijednost k-tog elementa polja
int i,j,l,m	pomoćne varijable

Kod Wirthovog algoritma (funkcija k_ti_najmanji):

```
float k_ti_najmanji(float *polje, int k, int n)
{
    int i=0,j=0,l,m ;
    float  x ;
    l=0 ;
      m=n-1 ;
```

```
    while (l<m) {
        x=polje[k] ;
        i=l ;
        j=m ;
        do{
            while (polje[i]<x)
                        i++ ;
            while (x<polje[j])
                        j-- ;
            if (i<j){
                zamijeni(&polje[i],&polje[j]) ;
                i++ ;
                        j-- ;
            }
            else if(i==j){
                i++;
                j--;
            }
        }
            while (i<=j) ;
        if (j<k)
                  l=i ;
        if (k<i)
                  m=j ;
    }
    return polje[k] ;
}
```

Unutar algoritma se koristi funkcija za zamjenu vrijednosti dvaju članova polja.

```
void zamijeni(float *x,float *y){
        float pom;
        pom=*x;
        *x=*y;
        *y=pom;
}
```

2.4 Usporedba dobivenih rezultata u C-u s Matlabom

2.4.1. Rezultat u C-u

U ovom poglavlju će se usporediti rezultati izračuna robusne estimacije očekivanja i varijance u C-u s programom u Matlabom za isto ulazno polje. U glavnoj (eng. main) funkciji u C pozivamo jedan od algoritama za pronaženje k-tog najmanjeg elementa iz prethodnog poglavlja 2.3. Budući da se radi sklopovska implementacija u ovom primjeru odabran je nerekurzivni Wirthov algoritam. Unutar glavne funkcije u C-u pozivaju se ove funkcije:

```
float k_ti_najmanji(float *a, int n, int k);
void inicijalizacija_polja(float *polje);
```

Pozivom funkcije inicijalizacija_polja(float *polje) inicijalizara se ulazno polje u glavnoj funkciji. U ovom primjeru inicijalizirat će se polje od 100 uzoraka (N=100). Pozivom „selection“ algoritma računa se medijan, prvi kvartil (Q_1) i treći kvartil (Q_3).

```
M=k_ti_najmanji(&P[0],N/2-1,N);
Q1=k_ti_najmanji(&P[0],N/4-1,N);
Q3=k_ti_najmanji(&P[0],N*3/4-1,N);
```

Robusna estimacija očekivanja jednaka je medijanu M. Varijanca se može izračunati kao umnožak interkvartilne razlike (IQR= Q_3 - Q_1) i koeficijenta 20/27.

```
printf("Ocekivanje E=%f\n",M);
Var1=(Q3-Q1)*20/27;
```

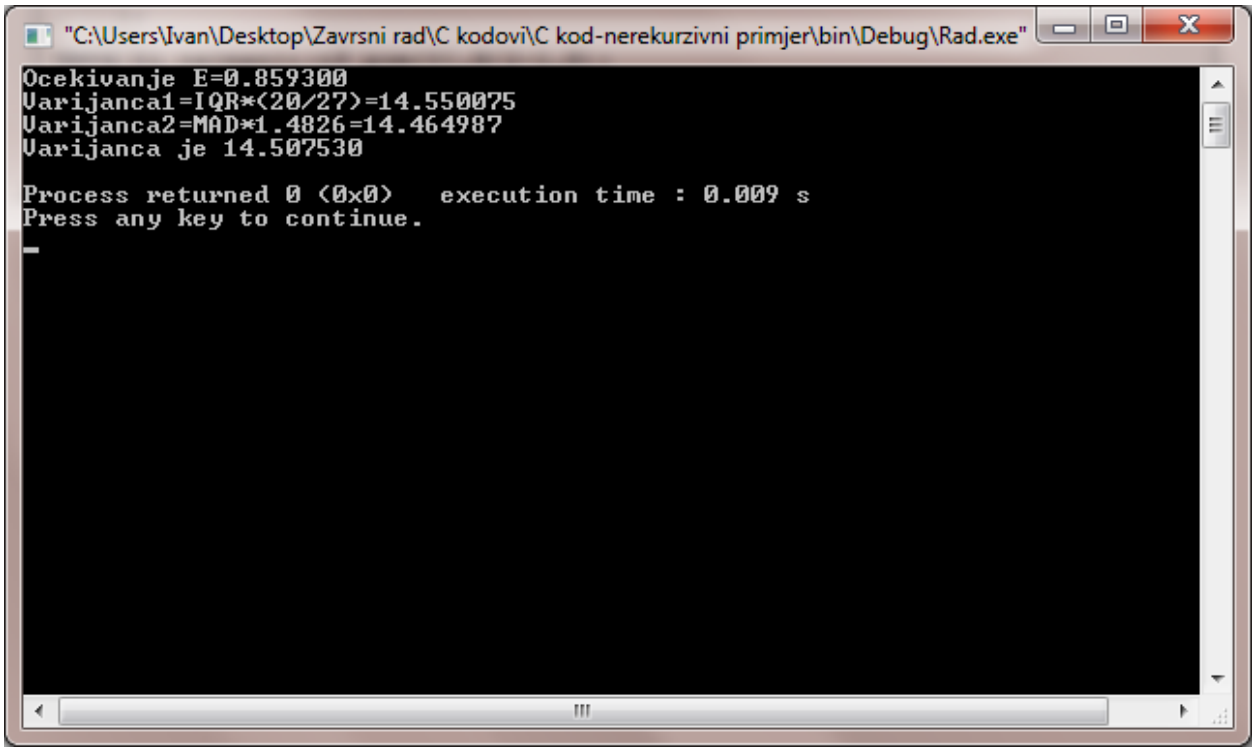
Drugi način kako se robusna estimacija varijance može izračunat je umnožak MAD-a i koeficijenta 1.4826. Dio koda za izračun MAD-a je u nastavku:

```
for(i=0;i<N;i++){
        if(P[i]>0)
            P_pom[i]=P[i]-M;
        else
            P_pom[i]=-P[i]-M;
    };
MM=k_ti_najmanji(&P_pom[0],N/2-1,N);
Var2=MM*1.4826;
```

Aritmetičkom sredinom se dobiva jos precizniji rezultat varijance.

```
Var=(Var1+Var2)/2;
printf("Varijanca je %f\n",Var);
```

Slika 2. Rezultat primjera u C-u

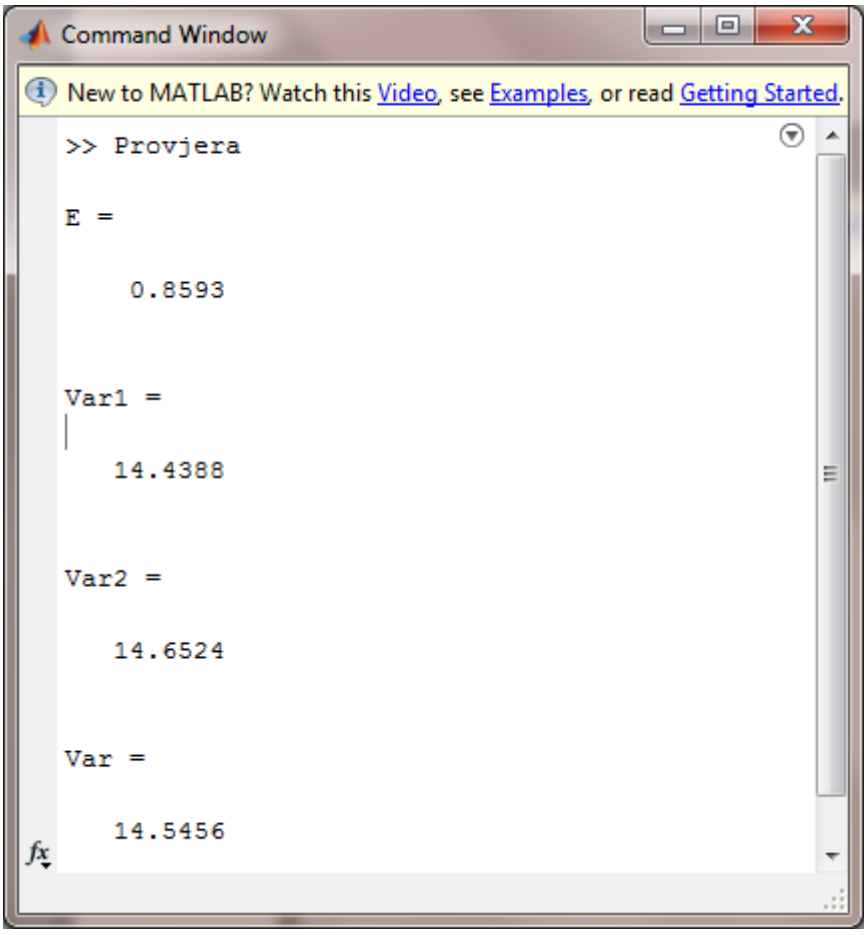


2.4.2 Rješenje u Matlabu

U Matlabu se dobiju gotovo iste vrijednosti za isto ulazno polje.

```
load('Polje.mat');
sort_polje=sort(polje);              % Sortiranje uzoraka
E=sort_polje(round(N/2))             % Ocekivanje
Var1=iqr(polje)*20/27;               %Varijacija preko IQR
Var2=mad(polje,1)*1.4826;            %Varijacija preko MAD
Var=(Var1+Var2)/2                    % usrednjavanje varijance
```

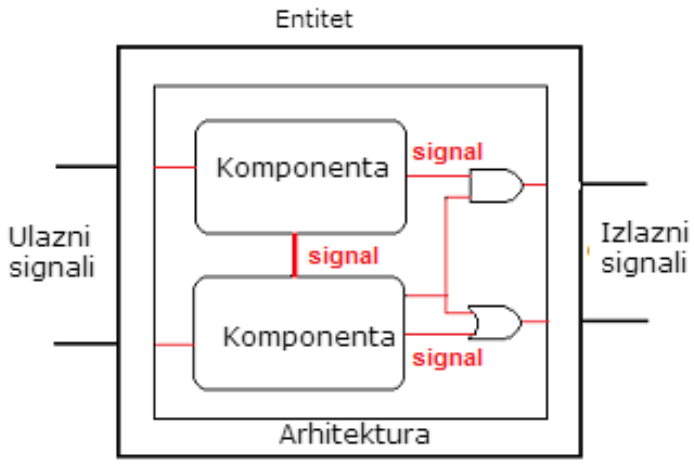
Slika 3. Rezultat u Matlabu



3. Ponašajni model (Behavioral model) u VHDL-u

3.1. Općenito o VHDL-u

VHDL (VHSIC Hardware Description Language) je uz Verilog najpopularniji jezik za opis digitalnih i mješovitih sustava kao što su FPGA i ostali integrirani krugovi. Osim toga ima primjenu kao jezik u paralelnom programiranju (eng. parallel programming language). Početak VHDL-a je 1981. godina kao projekt (VHSIC) američkog Ministarstva obrane (DoD) zbog potrebe za standardiziranim opisom sklopova, dokumentacije i verifikacija. Glavna prednost VHDL-a je to što omogućuje da se opiše ponašanje sustava i da se on verificira u simulatoru prije nego sto se dizajn ugradi u pravi sklop. Osim toga omogućuje opis složenijih struktura sklopa, dijeljenjem u jednostavnije. Opis sklopa u VHDL-u se sastoji od dva dijela (slika 4.) : sučelja i tijela opisa arhitekture (eng. architecture). Sučelje se opisuje u prvom dijelu i naziva se entitet (eng. entity). Unutar njega opisujemo vrste signala koji mogu biti ulazni/izlazni/dvosmjerni te tip podatka (opis izgleda sklopa prema vani). Broj ulaza (izlaza) mora biti konstantan. Unutar opisa arhitekture nalazi se model koji opisuje ponašanje sklopa. Ona se sastoji od deklaracije internih signala, procedura i funkcija.



slika 4. Opis sklopa u VHDL-u

Logika unutar arhitekture može biti kombinacijska (redoslijed naredbi nije bitan) ili slijedna. Slijedna logika je dopuštena unutar procesne petlje (eng. precess). Ako se blokovi programskog koda ponavljaju korisno je napraviti funkciju ili proceduru kako bi se pojednostavio kod. One moraju biti definirane unutar deklaracijskog dijela arhitekture ili procesa. Razvoj digitalnog sklopa u VHDL-u može se podijeliti u dva dijela: razvoj ponašajnog modela (eng. behavioral model) i fizički ostvariv model (eng. register transfer level model). Ponašajni model se rijetko može izravno implementirati jer se ne vodi računa o stvarnom sklopovlju. Za razliku od njega fizički ostvariv (RTL) model ima nizak stupanj apstrakcije, budući da je primarna namjena implementacija. Kod pisanja RTL modela moraju se uzeti u obzir konkretni zahtjevi koje postavlja tehnologija. U ovom radu naglasak je na razvoju ponašajnog modela robusne estimacije očekivanja i varijance.

3.2 Ponasajni model robusne estimacije ocekivanja i varijance u VHDL-u

Ponašajni model se piše na visokom stupnju apstrakcije tako da ima sličnosti s implementacijom u C-u. Najvažniji dijelovi koda će biti opisani u nastavku.

Ulazni podaci (realni brojevi) se čitaju preko tekstualne datoteke. U VHDL-u nema jedinstvene naredbe za čitanje iz datoteke, već se čita redak po redak naredbom readline i onda se naredbom read vrijednost sprema u zadano realno polje. Naredba endfile provjerava je li datoteka otvorena. U slučaju da je endfile:=1 prekida se s čitanjem iz datoteke.

```
for n in 1 to 100 loop
      if (not endfile(infile)) then
            readline(infile, inline);
            read(inline, ulazno_polje(n));
            kopija_ul_polja(n):=ulazno_polje(n);
            ulazno_polje_sig(n) <=ulazno_polje(n); -- citanje ulaznog
polja i spremanje u signal
      else
            endoffile <='1';
      end if;
end loop;
```

Slično vrijedi i za pisanje u tekstualnu datoteku. Ako je uvjet za pisanje u datoteku zadovoljen, mogu se izvršavati naredbe. Prvo se, naredbom write upisuje vrijednost varijable u redak, a zatim naredbom writeline taj se redak upisuje u tekstualnu datoteku.

```
if(endoffile='0') then
      write(outline, var_MAD, right, 12, 8);
      writeline(outfile, outline);
else
      null;
end if;
```

Unutar deklaracije arhitekture definirane su dvije procedure: za pronalazak k-tog najmanjeg broja i za pronalazak MAD-a. Procedura za pronalazak k-tog najmanjeg broja ima dosta sličnosti s implementacijom Wirthovog algoritma u C-u. Najveća razlika predstavlja deklaracija procedure gdje se deklariraju ulazni i izlazni tipovi podataka.

```
procedure k_ti_najmanji(variable Ulazni_uzorci:in realni_ulaz;
                                    constant k:in integer;
                                    constant n:in integer;
                                    variable rje:out real) is

variable i:integer:=0;
variable j:integer:=0;
variable x:real;
variable l,m:integer;
variable pomocna:real;
variable ulazno_pom_polje:realni_ulaz;
begin

      for n in 1 to 100 loop
            ulazno_pom_polje(n):=ulazni_uzorci(n);
      end loop;
      l:=1;
      m:=100;
      while (l<m) loop
            x:=ulazno_pom_polje(k);
            i:=L;
            j:=m;
            while(i<=j)loop
                  while(ulazno_pom_polje(i)<x)loop
                        i:=i+1;
                  end loop;
                  while(x<ulazno_pom_polje(j)) loop
                        j:=j-1;
                  end loop;
                  if(i<=j) then
```

```
                  if(i<=j) then
                        --zamjena brojeva u polju
                        pomocna:=ulazno_pom_polje(i);
                        ulazno_pom_polje(i):=ulazno_pom_polje(j);
                        ulazno_pom_polje(j):=pomocna;
                        i:=i+1;
                        j:=j-1;

                  end if;
            end loop;
            if(j<k) then
                  L:=i;
            end if;
            if(k<i) then
                  m:=j;
            end if;
   end loop;
      rje:=x;
end k_ti_najmanji;
```

U procedure za izračun MAD-a deklarirano je ulazno-izlazno polje, medijan i izlazna realna varijabla. Unutar for petlje apsolutnoj vrijednosti uzoraka oduzima se vrijednost medijana, te se tako promjenjenom polju traži medijan preko poziva proceduri k-ti najmanji broj.

```
procedure pronalazak_MAD(variable ulaz:inout realni_ulaz;
                                                variable M:in real;
                                                variable rje:out real)is


begin
      for n in 1 to 100 loop
            if(ulaz(n)<0.0)then
                  ulaz(n):=ulaz(n)*(-1.0);
            end if;
            ulaz(n):=ulaz(n)-M;
      end loop;
      k_ti_najmanji(ulaz,50,100,rje); --poziv procedure unutar procedure
end pronalazak_MAD;
```

Unutar procesa se izračunavaju očekivanje i varijanca, pa se pozivaju procedure.

```
k_ti_najmanji(ulazno_polje,50,100,Medijan);
Ocekivanje<=Medijan;
```

```
k_ti_najmanji(ulazno_polje,25,100,Q1);
k_ti_najmanji(ulazno_polje,75,100,Q3);
IQR:=Q3-Q1;
var_iqr:=IQR * 0.740741;
Varijanca_IQR<=var_iqr;
```

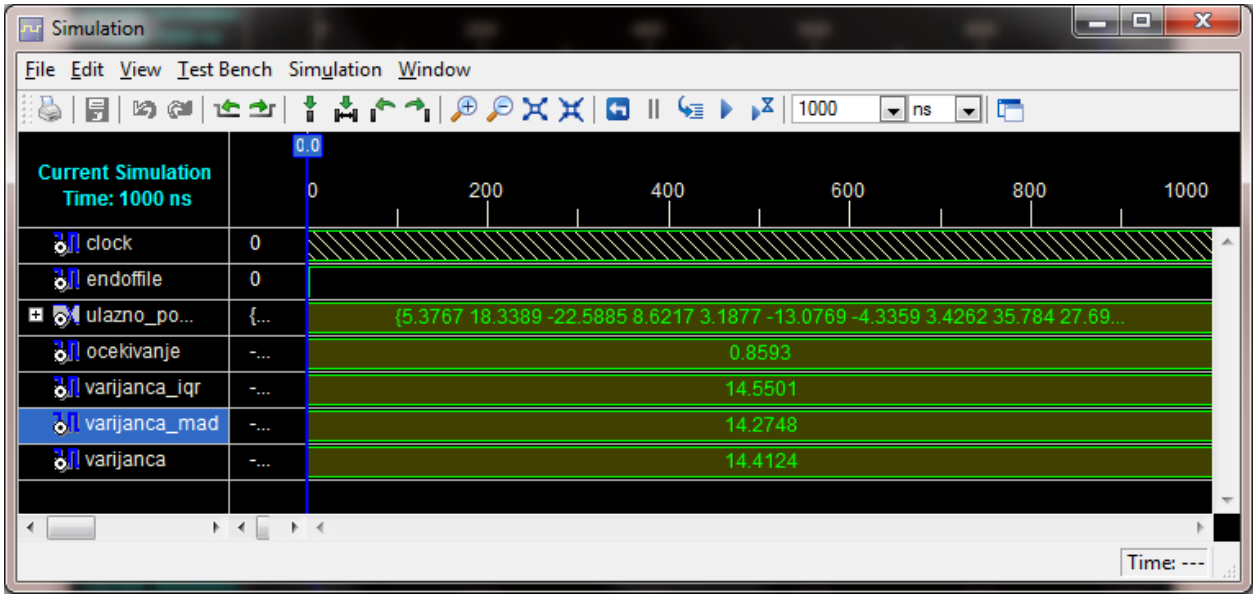
Kod računanja MAD-a važno je pričekati da se signal očekivanja promjeni, jer je to siguran znak da je tada i medijan izračunat. Ukoliko bi se MAD računao u isto vrijeme kao medijan rezultat sigurno ne bi bio točan.

```
wait until ocekivanje'event;
pronalazak_MAD(kopija_ul_polja,medijan,MAD);
Var_MAD:=MAD*1.4826;
Varijanca_MAD<=Var_MAD;
```

Usrednjavanjem rezultata varijance pomoću IQR-a i MAD-a dobiva se jos precizniji rezultat.

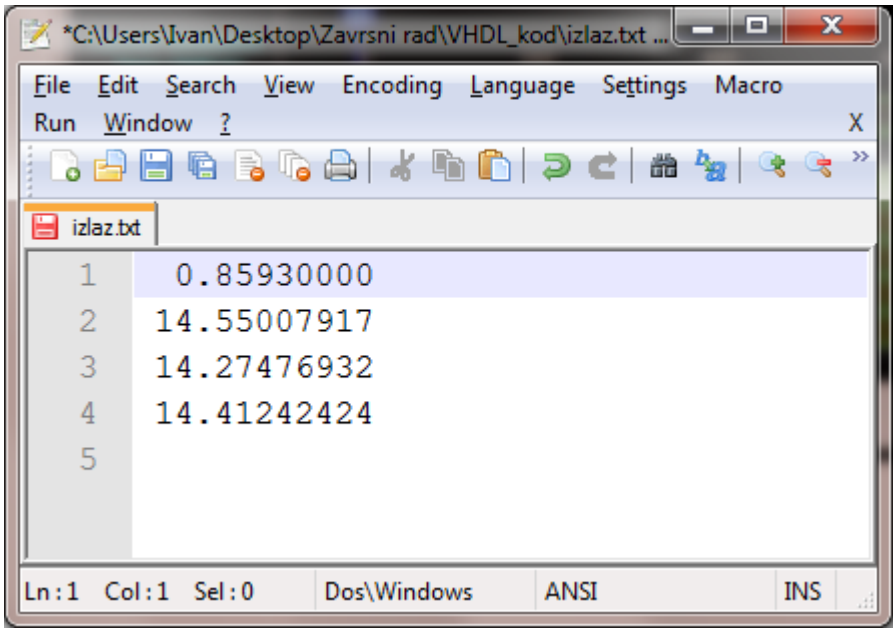
```
var:=(var_iqr+var_mad)/2.0;
Varijanca<=var;
```

Izvođenje ponašajnog modela pokazalo je zanemarive razlike u rezultatu u odnosu na realizaciju u C-u ili Matlabu.



Slika 5.

Ispis rezultata u izlaznu datoteku:



Slika 6.

Zaključak

U ovom radu proučavani su algoritmi koji služe za efikasno računanje robusne estimacije očekivanja i varijance, te razvoj ponašajnog modela (behavioral model) u VHDL-u. Za izračun robusne estimacije očekivanja i varijance potrebno je pronaći četiri uzorka u skupu N uzoraka. Za brzo pronalaženje, točno određenih uzoraka koji su potrebni za izračun robusne estimacije očekivanja i varijance, koriste se algoritmi koji pronalaze k-ti najmanji broj. Proučavana su dva algoritma: rekurzivni Medijan od medijana i Wirthov algoritam koji je jednostavnije sklopovski implementirati. Rezultati ponašajnog modela u VHDL-u su uspoređivani s rezultatima u C-u i Matlabu.

Potpis

Literatura

[1] http://en.wikipedia.org/wiki/Robust_statistics

[2] http://en.wikipedia.org/wiki/Interquartile_range

[3] http://en.wikipedia.org/wiki/Median_absolute_deviation

[4] Devillard, Nicolas: Fast median search: an ANSI C implementation, http://ndevilla.free.fr/median/median/index.html

[5] http://www.ics.uci.edu/~eppstein/161/960130.html

[6] Vučić, M., Molnar, G., Alati za razvoj digitalnih sustava – materijali za predavanja, Zagreb, 2009.

[7] Mlinarić,Hrvoje, Ugradbeni računalni sustavi – materijali za predavanja, Zagreb

Sažetak

Cilj ovog rada je usporedba ponašajnog (eng. behavioral) modela robusne estimacije očekivanja i varijance s programskim kodom u C-u i Matlabu. Robusna estimacija očekivanje jednaka je medijanu, dok se estimacija varijance određuje na dva načina: pomoću interkvartilnog raspona (IQR) i pomoću apsolutne devijacije medijana (MAD), te se rezultati usrednjuju kako bi estimacija bila što kvalitetnija. Četiri potrebna uzorka se pronalaze korištenjem algoritama za pronalazak k-tog najmanjeg elementa.

Ključne riječi

Robusna statistika, medijan, IQR, MAD, Medijan od medijana, Widhtov algoritam za brzi pronalazak medijana

Abstract

The aim of this project was realization of VHDL behavioral model of robust expectation and variance estimation and comparison to C programming code and model in Matlab. Estimation of robust expectation is equal to median, while robust variance is determineted with two methods: using interquatile range (IQR) or using median absolute deviation (MAD). Final result of robust variance estimation is arithmetic mean of these two methods. To find four needed samples, selection algoritm is used.

Keywords

Robust statistics, median, interquatile range (IQR), median absolute deviation (MAD), Median of medians, Fast median search (Wirth's method)

Dodatak A

Priložen je kod algoritma Medijan od medijana u C-u:

```
float median_of_medians(float* polje,int k,int n){
    int i,j;
    int n_m=0,n_v=0;
    float M;
    float potpolje[100][5];
    float x[100];
    float polje_m[100],polje_v[100];

     if(n<=10){
        sortiraj(polje,n);
        return polje[k];
     }
     for(i=0;i<n/5;i++)
        for(j=0;j<5;j++)
            potpolje[i][j]=*(polje+i*5+j);

     for(i=0;i<n/5;i
        x[i]=median_of_medians(&potpolje[i][5],2,5);

     M=median_of_medians(&x[0],n/10,n/5);

     for(i=0;i<n;i++){
         if(polje[i]<M){
             polje_m[n_m]=polje[i];
             n_m++;
        }
        else if(polje[i]>M){
            polje_v[n_v]=polje[i];
            n_v++;
        }
    }
```

```
    if (k<n_m)
        return median_of_medians(polje_m,k,n_m);
    else if (k > (n_m+1))
        return median_of_medians(polje_v,k-n_m-1,n_v);
    else
        return M;
}
```

Dodatak B

Priložen je kod za izračun robusne estimacije očekivanja i varijance u VHDL-u (ponašajni model):

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
library std;
use std.textio.all;


entity Robusna_estimacija_ocekivanja_i_varijance is
end Robusna_estimacija_ocekivanja_i_varijance;


architecture Behavioral of Robusna_estimacija_ocekivanja_i_varijance is
type realni_ulaz is array(1 to 100) of real;   -- definicija tipa ulaznog polja
signal clock,endoffile : bit := '0';
signal ulazno_polje_sig:realni_ulaz;
signal Ocekivanje:real;
signal Varijanca_IQR:real;
signal Varijanca_MAD:real;
signal Varijanca:real;
```

```
-- precedura za pronalazak k-tog najmanjeg uzorka
procedure k_ti_najmanji(variable Ulazni_uzorci:in realni_ulaz;
                                                    constant k:in integer;
                                                    constant n:in integer;
                                                    variable rje:out real) is

variable i:integer:=0;
variable j:integer:=0;
variable x:real;
variable l,m:integer;
variable pomocna:real;
variable ulazno_pom_polje:realni_ulaz;
begin

        for n in 1 to 100 loop
                ulazno_pom_polje(n):=ulazni_uzorci(n);
        end loop;
        l:=1;
        m:=100;
        while (l<m) loop
                x:=ulazno_pom_polje(k);
                i:=L;
                j:=m;
                while(i<=j)loop
                        while(ulazno_pom_polje(i)<x)loop
                                i:=i+1;
                        end loop;
                        while(x<ulazno_pom_polje(j)) loop
                                j:=j-1;
                        end loop;
                        if(i<=j) then
                                --zamjena brojeva u polju
                                pomocna:=ulazno_pom_polje(i);
                                ulazno_pom_polje(i):=ulazno_pom_polje(j);
                                ulazno_pom_polje(j):=pomocna;
                                i:=i+1;
```

```
                                j:=j-1;

                        end if;
                end loop;
                if(j<k) then
                        L:=i;
                end if;
                if(k<i) then
                        m:=j;
                end if;
  end loop;

  rje:=x;
end k_ti_najmanji;


--Procedura za pronalazak MAD-a
procedure pronalazak_MAD(variable ulaz:inout realni_ulaz;
                                variable M:in real;
                                  variable rje:out real) is

begin
        for n in 1 to 100 loop
                if(ulaz(n)<0.0)then
                        ulaz(n):=ulaz(n)*(-1.0);
                end if;
                ulaz(n):=ulaz(n)-M;
        end loop;
        k_ti_najmanji(ulaz,50,100,rje);         --poziv procedure unutar procedure


end pronalazak_MAD;
```

```
begin
clock <= not (clock) after 1 ns;    --clock with time period 2 ns

process
   file   infile    : text is in  "ulaz.txt";  --Ulazna datoteka
   file   outfile  : text is out "izlaz.txt";      -- Izlazna datoteka
   variable  inline    : line;
   variable  outline  : line;
   variable ulazno_polje:realni_ulaz;     --deklaracija varijable ulaznog polja
            variable kopija_ul_polja:realni_ulaz;       --pomocno polje za izracunat MAD
            variable Medijan:real;
            variable MAD:real;
            variable Q1,Q3,IQR:real;                                                    -- Prvi i treci kvartil
            variable Var,Var_iqr,Var_mad:real;

begin
wait until clock = '1' and clock'event;
-- citanje ulazne datoteke
for n in 1 to 100 loop
         if (not endfile(infile)) then
                  readline(infile, inline);
                  read(inline, ulazno_polje(n));
                  kopija_ul_polja(n):=ulazno_polje(n);
                  ulazno_polje_sig(n) <=ulazno_polje(n); -- citanje ulaznog polja i spremanje u signal

         else
                  endoffile <='1';
         end if;
end loop;

--Ocekivanje i upis u datoteku
k_ti_najmanji(ulazno_polje,50,100,Medijan);
Ocekivanje<=Medijan;              --spremanje u signal radi ocitanja u simulatoru
if(endoffile='0') then
         write(outline, medijan, right, 12, 8);
         writeline(outfile, outline);
```

```
else
        null;
end if;
--Varijanca pomocu IQR-a i upis u datoteku
k_ti_najmanji(ulazno_polje,25,100,Q1);
k_ti_najmanji(ulazno_polje,75,100,Q3);
IQR:=Q3-Q1;
var_iqr:=IQR * 0.740741;
Varijanca_IQR<=var_iqr;                 --spremanje u signal radi ocitanja u simulatoru
if(endoffile='0') then
        write(outline, var_IQR, right, 12, 8);
        writeline(outfile, outline);
else
        null;
end if;
--varijanca pomocu MAD-a i upis u datoteku
wait until ocekivanje'event;   --kad se signal ocekivanja promjeni,sigurno je tada i medijan poznat
pronalazak_MAD(kopija_ul_polja,medijan,MAD);
Var_MAD:=MAD*1.4826;
Varijanca_MAD<=Var_MAD;                 --spremanje u signal radi ocitanja u simulatoru
if(endoffile='0') then
        write(outline, var_MAD, right, 12, 8);
        writeline(outfile, outline);
else
        null;
end if;
--Aritmeticka sredina var_iqr i var_mad i upis u datoteku
var:=(var_iqr+var_mad)/2.0;
Varijanca<=var;                         --spremanje u signal radi ocitanja u simulatoru
if(endoffile='0')then
        write(outline, var, right, 12, 8);
        writeline(outfile, outline);
else
        null;
end if;
 end process;
end Behavioral;
```