AUTORI : Bojan Dimitrijević, Zorica Nikolić i Nenad Milošević
Katedra za telekomunikacije,
Elektronski fakultet ,
Univerzitet u Nišu

NAZIV : **Novi PSK prijemnik realizovan primenom tehnologije softverskog radija**

TIP TEHNIČKOG REŠENJA : M85 – prototip

ZAVRŠENO: 2010.

OBLAST : Telekomunikacije

KORISNICI: Elektronski fakultet u Nišu

PROBLEM:

Projektni zadatak je bio napraviti prototip novopredloženog PSK prijemnika na bazi USRP platforme uz korišćenje namenski realizovanog uprevljačkog softvera. N*ovim PSK prijemnikom* rešava se problem degradacije performansi u prisustvu značajnog frekvencijskog ofseta učestanosti lokalno generisanog nosioca u odnosu na učestanost nosioca prijemnog signala.

STANJE REŠENOSTI PROBLEMA:

U cilju rešavanja problema degradacije performansi koherentnog PSK prijemnika kada je prisutan značajan frekvencijski ofset učestanosti lokalno generisanog nosioca koristi se estimacija koeficijenta kanala. U tom slučaju može da se javlja novi problem a to je propagacija greške koji dodatno degradira performanse. *Novi prijemnik PSK signala* pripada ovoj klasi prijemnika.
Ukoliko povećanje snage na predaji ne predstavlja problem to se korišćenjem diferencijalne modulacije pri prenosu mogu postići iste performanse uz povećanje snage predajnika od 3 dB. Takođe se u praksi sreće i koričćenje Multiple Bit Differential Detection (MBDD). Nedostatak ovog rešenja je pre svega što pri većim frekvencijskim ofsetima dolazi do značajnog pada performansi.

Predloženo tehničko rešenje ima značajno bolje performanse pri prijemu PSK signala kada postoji veće odstupanje lokalno generisane učestanosti nosioca od učestanosti nosioca dolazećeg signala u poređenju sa koherentnim PSK prijemnikom a pri istoj snazi predajnika.

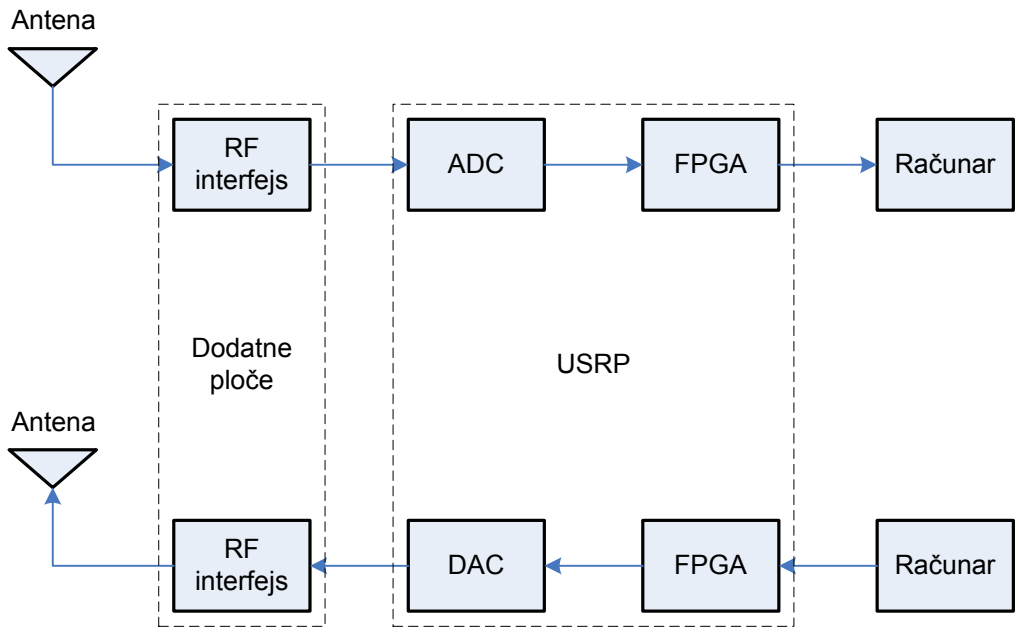
NOVI PSK PRIJEMNIK REALIZOVAN PRIMENOM TEHNOLOGIJE SOFTVERSKOG RADIJA

Realizacija novog PSK prijemnika se sastoji iz dva dela. Prvi deo predstavlja USRP hardver i namenski realizovan upravljački softver. Između ostalog u ovom delu se vrši konverzija primljenog signala u osnovni opseg. Drugi deo je realizovan kao softver za procesiranje signala u osnovnom opsegu koji radi pod LINUX-om na RS računaru. On predstavlja *Novi PSK prijemnik realizovan primenom tehnologije softverskog radija*.

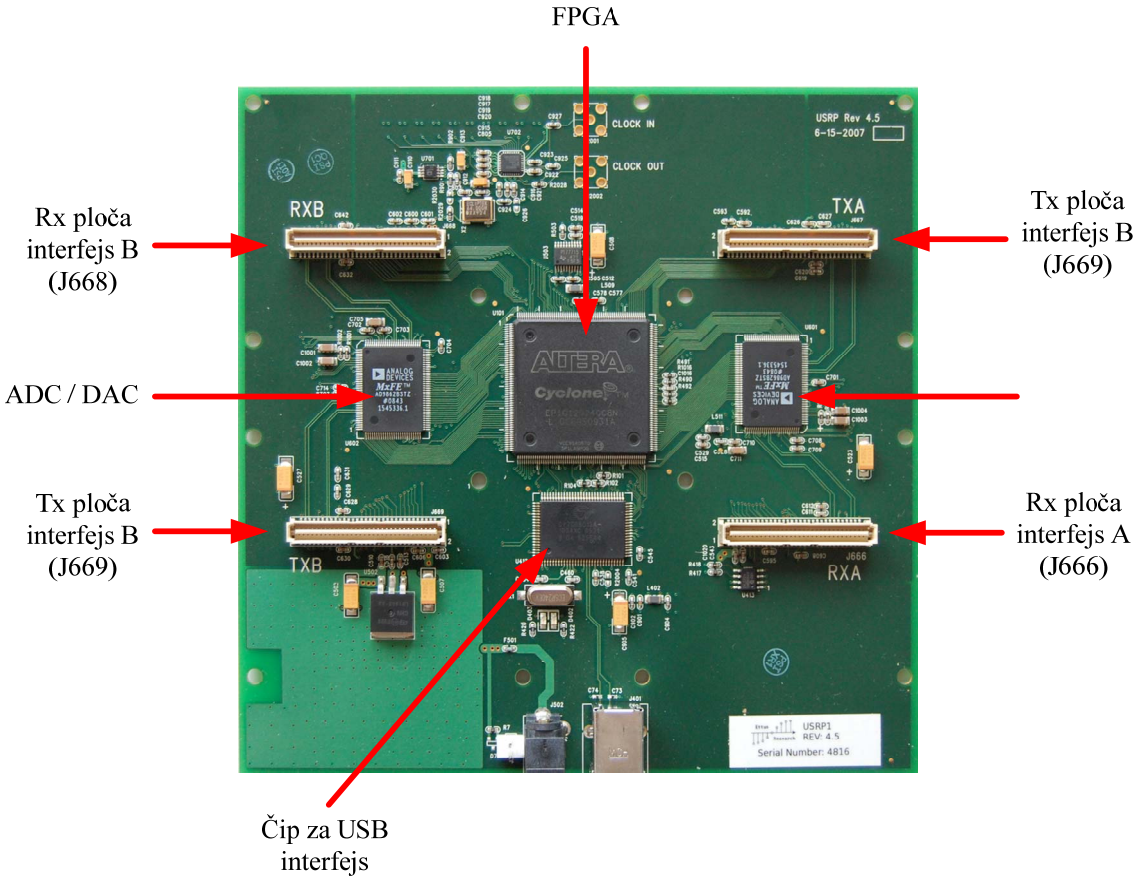
U daljem tekstu će najpre biti opisan korišćeni USRP hardver a zatim će biti izloženi osnovni principi novog *PSK* prijemnika i biće data konkretna realizacija u vidu softvera. Na kraju će biti upoređene performanse *Novog PSK prijemnik*a sa performansama trenutno aktuelnog prijema *PSK* signala koji koristi *PLL* petlju.

USRP (Universal Software Radio Peripheral)

Na slici 1 je prikazano kako koncept softverskog radija može da se realizuje korišćenjem USRP-a i dodatnih ploča. USRP obezbeđuje ADC/DAC i FPGA funkcionalnosti, dok različite dodatne ploče obezbeđuju translaciju spektra sa željenog frekvencijskog opsega u osnovni ili MF opseg. Fotografija USRP-a je prikazana na slici 2.



Slika 1. Blok šema sistema na principu softverskog radija realizovanog USRP-om



Slika 2. USRP matična ploča

USRP interfejsi za dodatne ploče su na slici 2 obeleženi sa *J66X*.

USRP i dodatne ploče su razvijene u okviru otvorenog projekta što znači da su kompletna dokumentacija i fajlovi korišćeni tokom razvoja javno dostupni. [2]

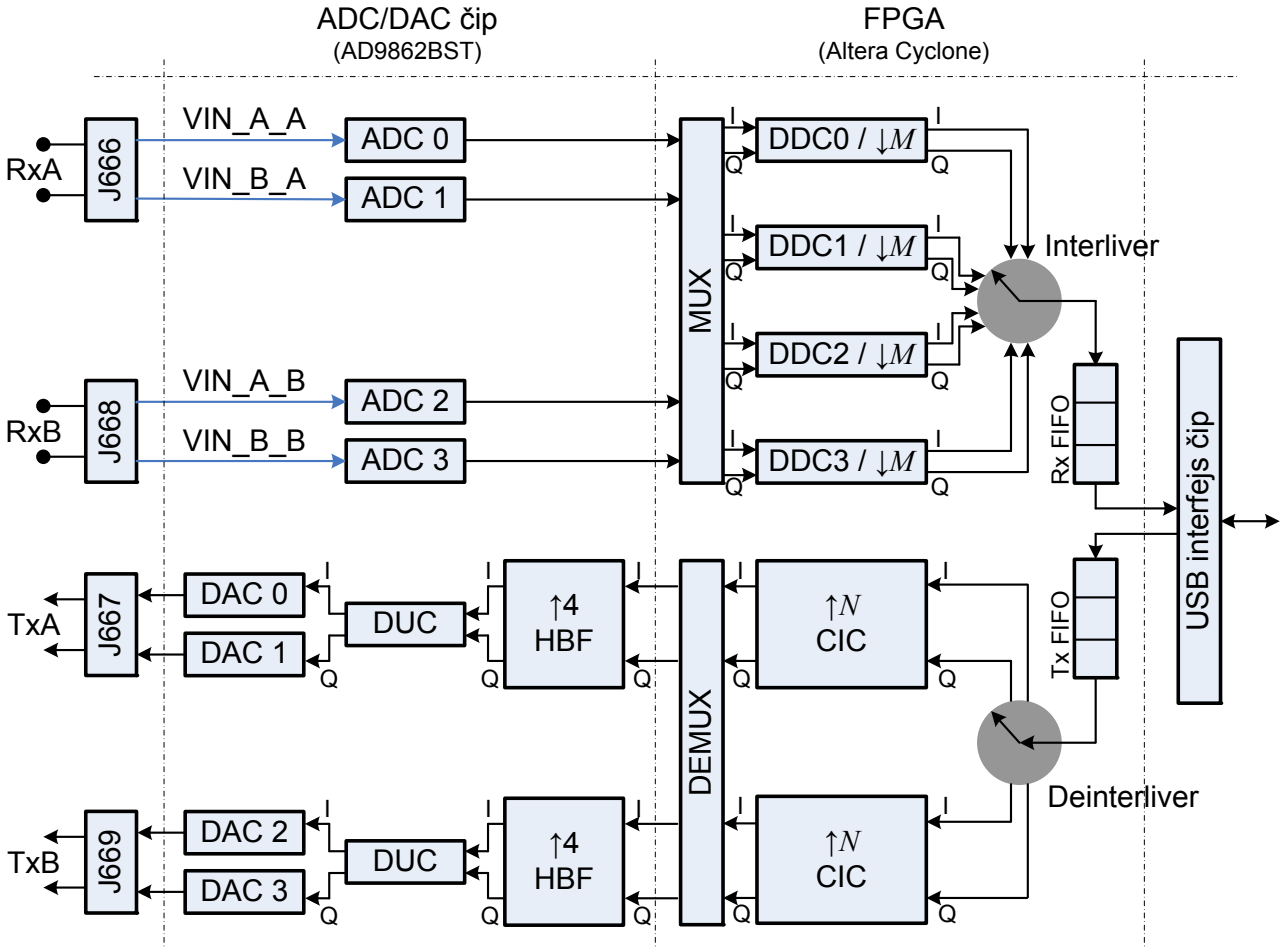
Osnovna ideja koja leži iza USRP-a je da se sva obrada signala vezana sa specifične talasne oblike, kao što je modulacija i demodulacija, obavlja u procesoru računara, a da se sve opšte operacije, za koje je potrebna velika brzina, kao što su digitalna konverzija naviše i naniže, decimacija i interpolacija obavljaju na FPGA. [1] Na slikama 3-5 su prikazane putanje signala u prijemnom i predajnom delu USRP-a.

Prijemni deo USRP-a

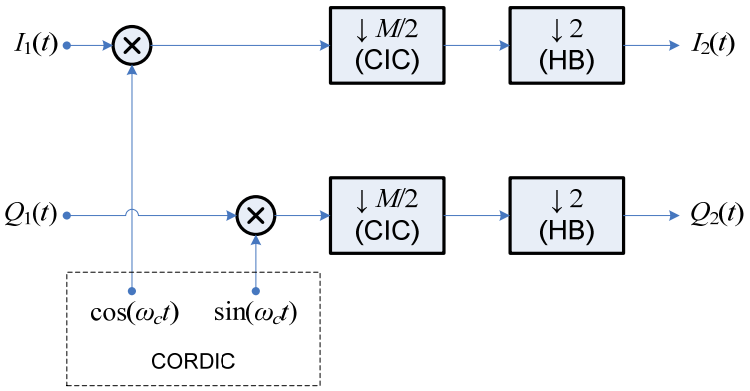
USRP ima dva slota koji služe za priključenje prijemnih ploča. Ovi slotovi su označeni sa *RxA* i *RxB*, a odgovarajući interfejsi su *J666* i *J668*. Svaki od interfejsa prihvata dva signala iz dodatne ploče sa realnim naponskim nivoima. Ovi signali su označeni sa *VIN_A_X* i *VIN_B_X*, gde se *X* menja sa *A* ili *B*, u zavisnosti od toga kom slotu signali pripadaju (*RxA* ili *RxB*). S obzirom da su oba slota identična, ovi signali će u daljem teksu biti označeni sa *VIN_A* i *VIN_B*, osim u slučaki kad ih je potrebno u potpunosti označiti.
Analogni signali *VIN_A* i *VIN_B* se vode na dva dovojena AD konvertora. Signali se odmeravaju frekvencijom od 64 MS/s, a svaki odmerak ima 12 bita. Signali se iz AD konvertora zatim šalju u FPGA na dalju obradu. Nakon ulaska u FPGA signali se multiplekserom (MUX) prenose do odgovarajućeg digitalnog konvertora naniže (DDC).
DDC je u osnovi kompleksni mikser. On na ulazima očekuje signale u fazi i kvadraturi. Korisnik određuje da li će *VIN_A_A, VIN_B_A, VIN_A_B, VIN_B_B*, ili sve nule da se proslede do porta u

fazi ili kvadraturi svakog od četiri DDC-a. Svaki DDC prebacuje svoj ulazni signal u osnovni opseg. Nakon konverzije naniže, signal se decimira faktorom koji korisnik odredi. Decimacija se odvija u dve faze. Pod pretpostavkom da je korisnik izabrao faktor decimacije *M*, signal se prvo decimira za faktor *M*/2, korišćenjem CIC (cascaded integrator-comb) filtra. Poslednja decimacija sa 2 se obavlja HB (half-band) filtrom. DDC i decimacija su prikazani na slici 4.

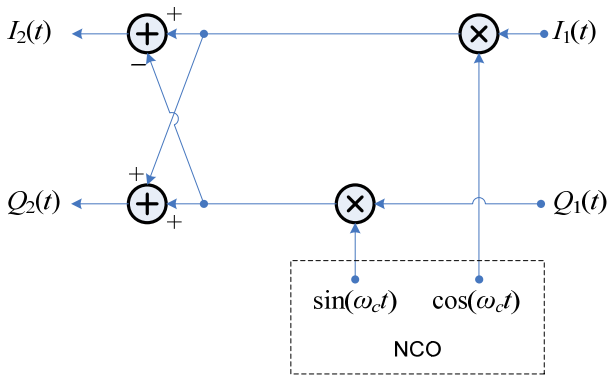
Posle decimatora signal se vodi na FIFO bafer. Odavde se odmerci šalju u računar posredstvom USB 2.0 interfejsa. Na slici 3 su prikazana četiri DDC/decimacija kola. Međutim, trenutno su implmentirana samo dva. To znači da korisnik može da spcificira dva kanala i prima podatke iz oba *RxA* i *RxB*. Svaki kompleksni odmerak (u fazi i kvadraturi) se šalje korišćenjem 32 bita (16 bita za signal u fazi i 16 bita za signal u kvadraturi). Čip koji je zadužen za prenos podataka preko USB 2.0 interfejsa može da podrži maksimalnu brzinu prenosa od 32 megabajta u sekundi. Ovaj parametar ograničava propusni opseg signala koji se prenosi ka- i iz računara.



Slika 3. Prijemni i predajni deo USRP-a



Slika 4. Stepen za decimaciju i digitalnu konverziju naniže



Slika 5. Stepen za digitalnu konverziju naviše

Predajni deo USRP-a

Predajni deo USRP-a radi vrlo slično prijemnom. Prvo se podaci iz računara smeštaju u predajni FIFO bafer na USRP-u. Iz ovog bafera podaci se šalju na interpolator *X*. Svaki kompleksni odmerak je dužine 32 bita, kao u prijemnom delu. Ako je korisnik podesio faktor interpolacije *L*, ulazni podaci se prvo interpoliraju faktorom *L*/4 pomoću CIC filtra.
Izlaz interpolatora se šalje do demultipleksera (DEMUX). DEMUX je manje komplikovan od MUX-a u prijemniku. Ovde se izlazi u fazi i kvadraturi svakog CIC interpolatora šalju na ulaze u fazi i kvadraturi jednog od DAC čipova na ploči. Korisnik bira koji DAC čip prima signale svakog od CIC interpolatora.
Unutar DAC-a, kompleksni signal se interpolira za faktor 4, korišćenjem HB filtra. Ovim se završava zadata interpolacija. Nakon interpolatora, signal se vodi do digitalnog konvertora naviše (DUC). U ovom trenutku signal ne mora da bude modulisan na frekvenciju nosioca. Dodatna RF ploča može da dalje konvertuje signal naviše.
Signali u fazi i kvadraturi na izlazu DUC-a se šalju kao 14-bitni odmerci do individualnih DA konvertora. DA konvertori rade brzinom od 128 MS/s. Ovi analogni signali se šalju iz AD9862 do jednog od interfejsa *J667* ili *J666*, koji predstavljaju slotove *TxA* i *TxB*.

Dodatne ploče

U tabeli 1 su prikazane dodatne ploče koje su trenutno razvijene. Osnovne ploče nemaju nikakvo podešavanje ili pojačanje i, u osnovi, predstavljaju interfejse eksternih komponenti sa USRP-om. Sve druge ploče imaju podešavanje i pojačanje.

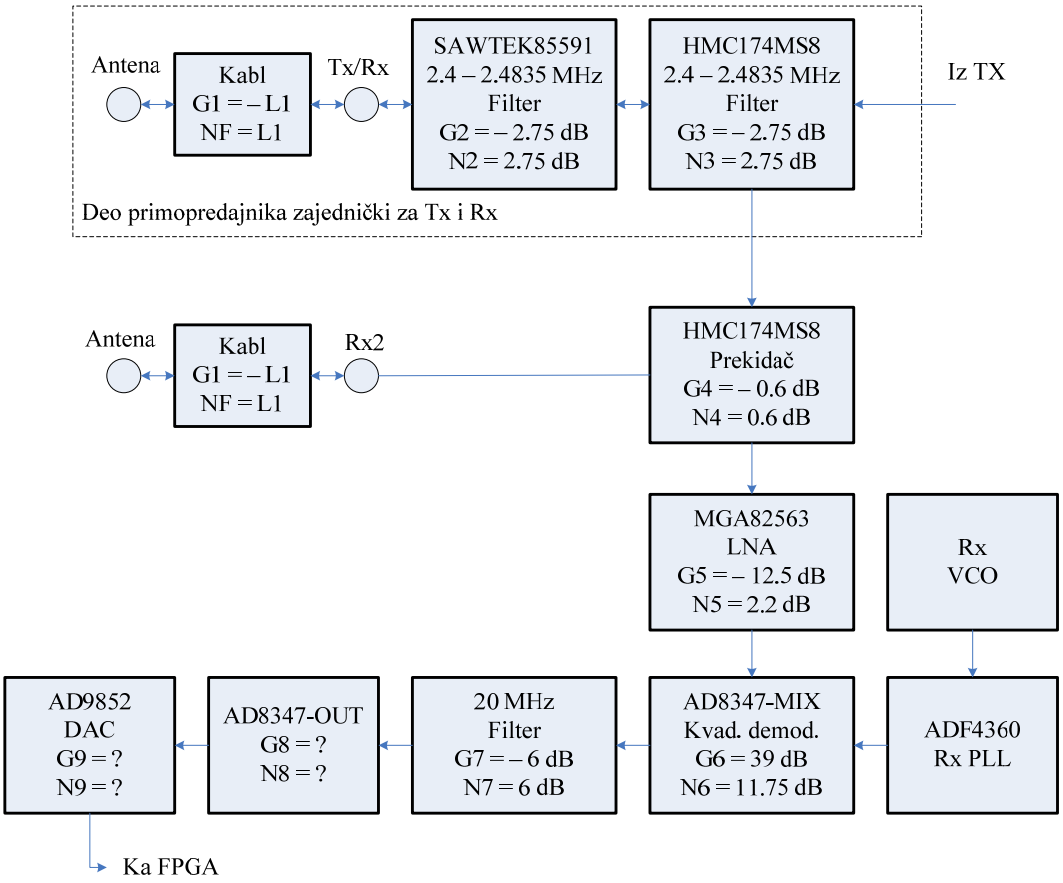
Tabela 1. Dodatne ploče za USRP

Ime	**Funkcionalnost**	**Frekvencijski opseg (MHz)**
BasicRx	Prijemnik	2 – 300+
BasicTx	Predajnik	2 – 200
LFRX	Prijemnik	0 – 30
LFTX	Predajnik	0 – 30
TVRX	Prijemnik	50 – 870
DBSRX2	Prijemnik	800 – 2400
RFX400	Primopredajnik	400 – 500
RFX900	Primopredajnik	800 – 1000
RFX1200	Primopredajnik	1150 – 1450
RFX1800	Primopredajnik	1500 – 2100
RFX2200	Primopredajnik	2000 – 2400
RFX2400	Primopredajnik	2300 – 2900
XCVR2450	Primopredajnik	2400 – 2500 4900 – 5850
WBX	Primopredajnik	50 – 2200

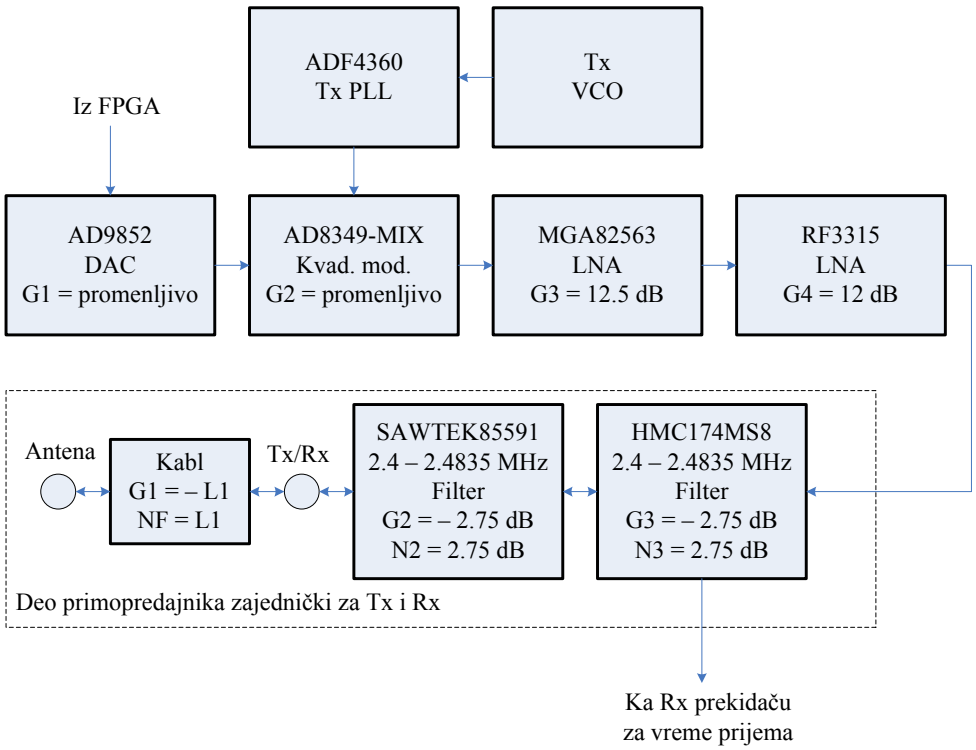
Pri realizaciji ovog tehničkog rešewa korišćena je *RFX2400* 2.4 GHz primopredajna ploča i opisana je u daljem tekstu.

RFX2400 sadrži dva osnovna čipa i to AD8347 kvadraturni demodulator i AD8349 kvadraturni modulator. Oba ova čipa vrše direktnu konverziju, što znači da mogu da prebacuju signal iz osnovnog opsega u 2.4 GHz i obratno, bez međufrekventnih stepena. Signali lokalnog oscilatora, koji se dovode na kvadraturni demodulator AD8347, se sintetišu pomoću PLL-a, u okviru koga se koristi naponski kontrolisan oscilator (VCO). Kolo ADF4360 se koristi za implementaciju ove petlje. Ima tipično vreme hvatanja frekvencije od 250 μs i može da ima samo diskretne vrednosti frekvencije.

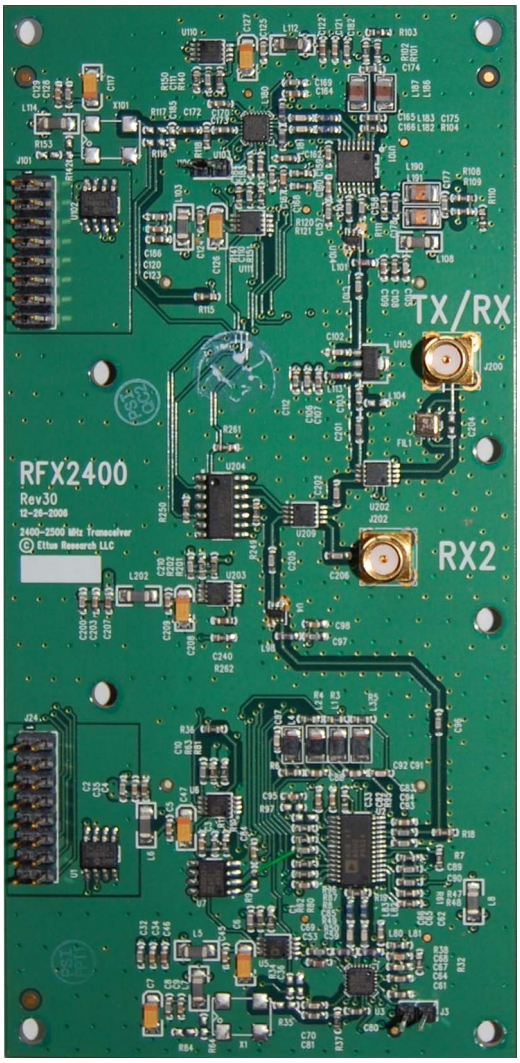
Na slikama 6 i 7 su prikazane blok šeme prijemnog i predajnog dela RFX2400, dok je na slici 8 prikazana fotografija ove ploče.



Slika 6. Prijemni deo RFX2400 ploče



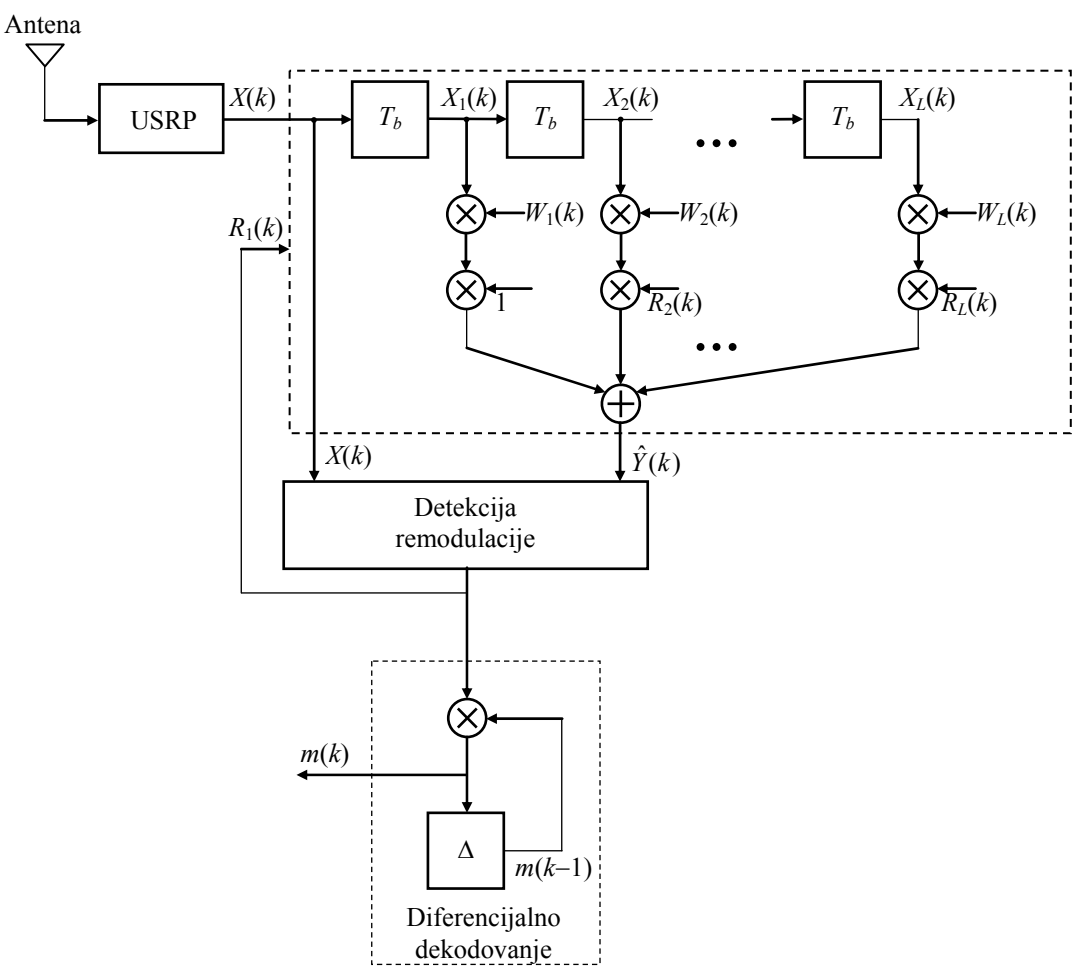
Slika 7. Predajni deo RFX2400 ploče



Slika 8. Fotografija RFX2400 ploče

PRINCIPI RADA *NOVOG PSK PRIJEMNIKA*[4]

Blok šema *Novog PSK prijemnika* realizovanog tehnologijom softverskog radija prikazana je na slici 9.



Slika 9

Kompleksni signal na izlzu USRP-a može se predstaviti na sledeći način:

$$X(k) = X_I(k) + jX_Q(k)$$

gde je k diskretno vreme .

Procesiranje signala $X(k)$ najpre se vrši u strukturi koja je na blok šemi označena sa M-ATF. Ova struktura je kompleksni jednostrani transverzalni filter dužine L koji, pored težinskih koeficijenata $W_l(k), l = 1...L$, sadrži i remodulacione koeficijente $R_l \in \{+1,-1\}, l = 1,...,L$. Rad M-ATF structure može se opisati pomoću algoritma koji ima sledeće korake:

1. Predpostavlja se da remodulacioni koeficijent $R_1(k)$ ima vrednost 1

2. Vrši se preliminarna estimacija ulaznog signala:

$$\hat{Y}(k) = \frac{1}{L}\left(1 \cdot X(k-1) \cdot W_1(k) + \sum_{l=2}^{L} R_l(k) \cdot X(k-l) \cdot W_l(k)\right)$$

3. U okviru bloka *Detekcija remodulacije* ,vrši se estimacija koeficijenta $R_1(k)$ na sledeći način:

$$R_1(k) = \begin{cases} 1, & \left|X(k) - \hat{Y}(k)\right|^2 \le \left|X(k) + \hat{Y}(k)\right|^2 \\ -1, & \left|X(k) - \hat{Y}(k)\right|^2 > \left|X(k) + \hat{Y}(k)\right|^2 \end{cases}$$

4. Korišćenjem estimiranog koeficijenta $R_1(k)$ vrši se estimacija ulaznog signala:

$$Y(k) = \hat{Y}(k) \cdot R_1(k)$$

Na taj način, vrši se korekcija moguće greške učinjene u koraku 1.

5. Signal greške kod LMS algoritma je

$$E(k) = X(k) - Y(k)$$

6. Pored coeficijenata remodulacije M-ATF takođe sadrži i težinske koeficijente $W_l(k), l = 1...L$ čije se podečavanje vrši LMS algoritmom [6]:

$$W_l(k+1) = W_l(k) + \frac{\mu E(k)[X(k-l)R_l(k)]^*}{\frac{1}{L}\sum_{l=1}^{L}\left(X(k-l)\right)^2}$$

gde je μ koeficijent konvergencije.

7. U narednom intervalu semplovanja remodulacioni koeficijenti se računaju sledećom rekurzivnom relacijom:

$$R_l(k+1) = R_{l-1}(k) \cdot R_1(k), \quad l = 2,...,L$$

Težinski koeficijent $R_1(k)$ predstavlja detektovani informacioni signal pri diferencijalnoj modulaciji. Kako je naš zadatak da detektujeno informacioni signal koherentne fazne modulacije (*m(k)*) to se konačno dobija:

$$m(k) = m(k-1) \cdot R_1(k)$$

Izvorni kod softvera *Novog PSK prijemnika*

```
//=========================================================================
===
//    definition of class PSKReceiver
//=========================================================================
===
#include <math.h>
class PSKReceiver
{
      public:


      int Ready();
            void GetOut(int& OutI);
```

Funkcija za preuzimanje izlaznog signala iz prijemnika

Funkcije za podešavanje parametara prijemnika

```
            void SetL1(int L1);
            void SetFLLA(double FLLA);
            void SetFLLK(double FLLK);
            void SetDLLA(double DLLA);
            void SetDLLK(double DLLK);
            void SetAntennaNumber(int AntennaNumber);
            void SetDefault();
            void SetFrequency(double Frequency);
            void SetC1(double C1);
            void SetIn(int Antenna, double InI,double InQ);
            void SetSamplingPeriod(double SamplingPeriod);
            void SetOversampling(int Oversampling);
            void Init();
            void Step();
```

Postavljanje svih parametara na podrazumevane vrednosti

Inicijalizacija prijemnika pre početka rada

Funkcija koja se mora pozivati za svaki diskretni vremenski korak. U okviru nje se vrši kompletno procesiranje

```
      private:
```

Interni parametri koje koristi klasa PSKReceiver

```
            double itsSamplingPeriod;
            int itsOut;
            double itsXI[201],itsXQ[201];
            double itsWI[201],itsWQ[201];
            double itsRXI[201],itsRXQ[201];
            double itsInI[20],itsInQ[20];
            double itsSInI[20][20],itsSInQ[20][20];
            double itsOSInI[20],itsOSInQ[20];
            double itsFrequency;
            double itsFLLK,itsFLLA;
            double itsC1;
            double      itsPhase,itsTeta;
            double itsDLLK,itsDLLA;
            double itsDLLP;
            int itsL1;
            double itsCounter;
            int itsAntennaNumber;
            int itsOversampling;
```

```
};

inline int PSKReceiver::Ready()
{
    int aCounter=(int)(itsCounter);
      if (aCounter>0)
      {
            if ((aCounter%itsOversampling)==0) return 1;
            else return 0;
      }
      else return 0;
}

inline void PSKReceiver::GetOut(int& Out)
{
      Out=itsOut;
}

inline void PSKReceiver::SetL1(int L1)
{
      itsL1=L1;
}

inline void PSKReceiver::SetFLLK(double FLLK)
{
      itsFLLK=FLLK;
}

inline void PSKReceiver::SetFLLA(double FLLA)
{
      itsFLLA=FLLA;
}

inline void PSKReceiver::SetDLLK(double DLLK)
{
      itsDLLK=DLLK;
}

inline void PSKReceiver::SetDLLA(double DLLA)
{
      itsDLLA=DLLA;
}

inline void PSKReceiver::SetAntennaNumber(int AntennaNumber)
{
    itsAntennaNumber=AntennaNumber;
}

inline void PSKReceiver::SetDefault()
{
      itsL1=4;
      itsC1=0.01;
      itsFLLK=1;
      itsFLLA=0.001;
      itsDLLK=0.0001;
      itsDLLA=0.0001;
      itsFrequency=0;
      itsPhase=0;
      itsSamplingPeriod=5e-10;
      itsOversampling=8;
      itsAntennaNumber=1;
}
```

```
inline void PSKReceiver::SetFrequency(double Frequency)
{
      itsFrequency=Frequency;
}

inline void PSKReceiver::SetC1(double C1)
{
      itsC1=C1;
}

inline void PSKReceiver::SetIn(int Antenna, double InI,double InQ)
{

      itsInI[Antenna]=InI;
      itsInQ[Antenna]=InQ;
}

inline void PSKReceiver::SetSamplingPeriod(double SamplingPeriod)
{
      itsSamplingPeriod=SamplingPeriod;
}

inline void PSKReceiver::SetOversampling(int Oversampling)
{
      itsOversampling=Oversampling;
}

inline void PSKReceiver::Init()
{
      int i,ii;
      itsCounter=1000;
      for (i=0;i<201;i++)
      {
            itsXI[i]=0;
            itsXQ[i]=0;
            itsWI[i]=1;
            itsWQ[i]=0;
            itsRXI[i]=0;
            itsRXQ[i]=0;
      }
      itsOut=1;
      itsTeta=0;
      itsDLLP=0;
      for (i=0;i<20;i++)
      {
        for (ii=0;ii<20;ii++)
        {
            itsSInI[i][ii]=0;
            itsSInQ[i][ii]=0;
        }
        itsInI[i]=0;
        itsInQ[i]=0;
        itsOSInI[i]=0;
        itsOSInQ[i]=0;
      }
}



inline void PSKReceiver::Step()
{// BPSK Receiver
```

```
    int aCounter,i,j,k;
      double tI,tQ,taI,taQ,tbI,tbQ;
      double ta,tb;
      double tp,tmI,tmQ;


////////////////////////////////////////////////
      itsPhase+=(6.28318530717959*itsFrequency*itsSamplingPeriod);
// FLL Off
    itsPhase+=(itsTeta/itsOversampling);
//
// DLL ON
    itsCounter+=(itsDLLP/itsOversampling);
    aCounter=(int)(itsCounter);
//
    tI=cos(itsPhase);
    tQ=sin(itsPhase);
    for (i=0;i<itsAntennaNumber;i++)
    {
        j=aCounter%(itsOversampling*2);
        itsSInI[i][j]=(itsInI[i]*tI+itsInQ[i]*tQ);
        itsSInQ[i][j]=(itsInQ[i]*tI-itsInI[i]*tQ);
    }
      if ((aCounter%itsOversampling)==0)
      {
          for (i=0;i<itsAntennaNumber;i++)
          {
            taI=0;
            taQ=0;
            tbI=0;
            tbQ=0;
              for (k=1;k<(1+itsOversampling);k++)
              {
                j=(aCounter-k)%(itsOversampling*2);
                if (k>(itsOversampling/2))
                {
                    taI+=itsSInI[i][j];
                    taQ+=itsSInQ[i][j];
                }
                else
                {
                    tbI+=itsSInI[i][j];
                    tbQ+=itsSInQ[i][j];
                }
              }
              itsOSInI[i]=taI+tbI;
              itsOSInQ[i]=taQ+tbQ;


            ta=atan2((taI*tbQ-taQ*tbI),(taI*tbI+taQ*tbQ));
            itsTeta=(itsTeta*(1-itsFLLA))+(ta*itsFLLA*itsFLLK);// II-order-
FLL

            taI=taI*taI+taQ*taQ;
            tbI=tbI*tbI+tbQ*tbQ;
            if ((taI+tbI)>1e-20) ta=(taI-tbI)/(taI+tbI);
            else ta=0;
            itsDLLP=(itsDLLP*(1-itsDLLA))+(ta*itsDLLA*itsDLLK);// II-order-
DLL
          }
    }
////////////////////////////////////////////////
```

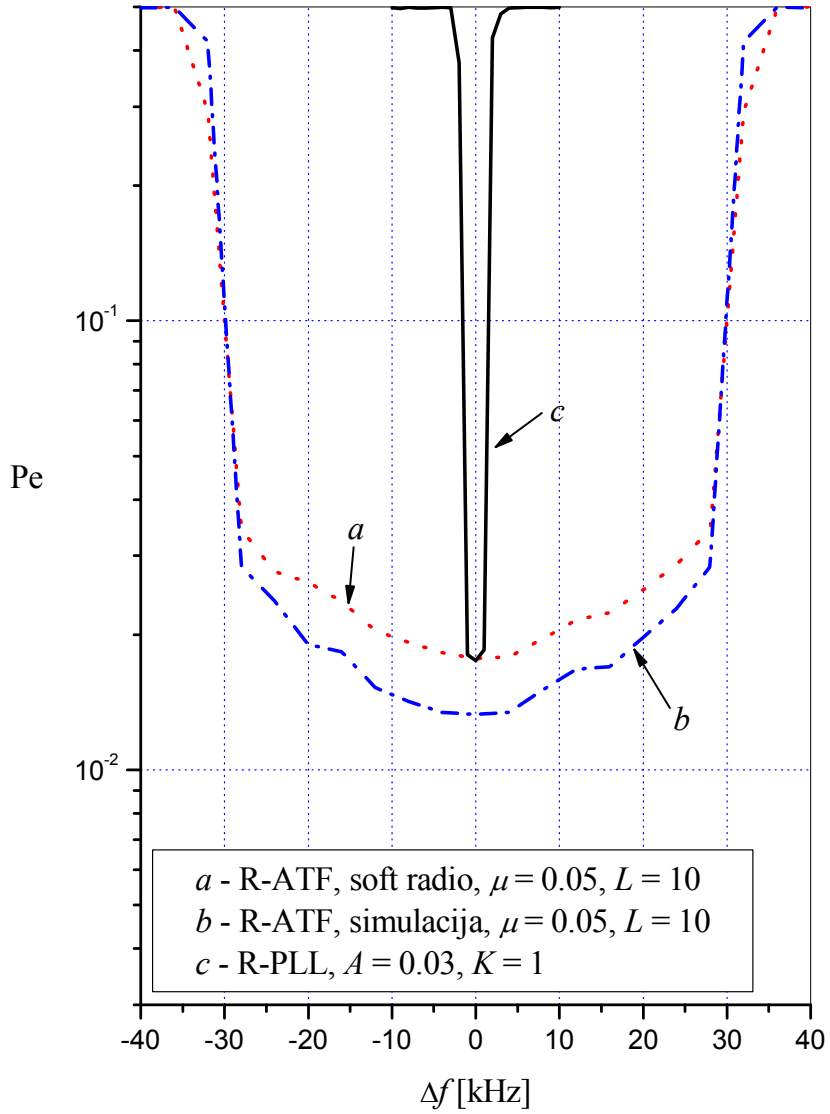
```
      if ((aCounter%itsOversampling)==0)
      {
        for (i=200;i>0;i--)
        {
            itsXI[i]=itsXI[i-1];
            itsXQ[i]=itsXQ[i-1];
            itsRXI[i]=itsRXI[i-1];
            itsRXQ[i]=itsRXQ[i-1];
        }

////////////////////////////////////////////////////

        itsXI[0]=0;
        itsXQ[0]=0;
          for (i=0;i<itsAntennaNumber;i++)
          {
            itsXI[0]+=itsOSInI[i];
            itsXQ[0]+=itsOSInQ[i];
          }


////////////////////////////////////////////////////

        itsRXI[1]=1;
        itsRXQ[1]=0;
        tI=0;
        tQ=0;
        for (i=1;i<itsL1+1;i++)
        {
            taI=itsXI[i]*itsRXI[i]-itsXQ[i]*itsRXQ[i];
            taQ=itsXI[i]*itsRXQ[i]+itsXQ[i]*itsRXI[i];
            tI+=taI*itsWI[i]-taQ*itsWQ[i];
            tQ+=taI*itsWQ[i]+taQ*itsWI[i];
        }
        tI/=itsL1;
        tQ/=itsL1;

        taI=itsXI[0]-tI;
        taQ=itsXQ[0]-tQ;
        tbI=itsXI[0]+tI;
        tbQ=itsXQ[0]+tQ;

        ta=taI*taI+taQ*taQ;
        tb=tbI*tbI+tbQ*tbQ;
        if (ta<=tb)
        {
            itsRXI[1]=1;
            itsRXQ[1]=0;
            if (itsOut>0)
                itsOut=1;
            else
                itsOut=0;
        }
        else
        {
            for (i=1;i<itsL1+1;i++)
            {
                itsRXI[i]=-itsRXI[i];
                itsRXQ[i]=0;
            }
            tI=-tI;
            tQ=-tQ;
            if (itsOut>0)
```

Predviđena mogućnost implementacije kombinovanja signala sa više antena. Trenutno je realizovano sabiranje signala bez kofaziranja.

```
                itsOut=0;
            else
                itsOut=1;
        }

        tmI=itsXI[0]-tI;
        tmQ=itsXQ[0]-tQ;
        tmI*=itsC1;
        tmQ*=itsC1;
        tp=1e-10;
        for (i=1;i<itsL1+1;i++)
        {
            tp+=(itsXI[i]*itsXI[i]+itsXQ[i]*itsXQ[i]);
        }
        tp/=itsL1;
        for (i=1;i<itsL1+1;i++)
        {
            taI=itsXI[i]*itsRXI[i]-itsXQ[i]*itsRXQ[i];
            taQ=itsXI[i]*itsRXQ[i]+itsXQ[i]*itsRXI[i];
            itsWI[i]+=((tmI*taI+tmQ*taQ)/tp);
            itsWQ[i]+=((tmQ*taI-tmI*taQ)/tp);
        }



////////////////////////////////////////////////////////////////////////////
///


        if ((aCounter%1000)==0)
        {
            tp=0;                           //debug point
        }
        if ((aCounter%10000)==0)
        {
            tp=0;                           //debug point
        }
//          itsTeta=(itsWQ[1]*itsA);
      // I-order-FLL OK
//        itsTeta=(itsTeta*(1-itsB))+((itsWQ[1]*itsA)*itsB);          // II-
order-FLL OK
      }
      itsCounter++;
}
```

Verifikacija rezultata

Na slici 10 prikazana je zavisnost verovatnoće greške u funkciji ofseta učestanosti referentnog nosioca Δf pri odnosu signal/šum 4 dB na izlazu iz USRP-a. Grafik ilustruje približno poklapanje simulacionih karakteristika novopredloženog PSK prijemnika (kriva *b*) sa karakteristikama prijemnika realizovanog tehnologijom softverskog radija (kriva *a*). Na tom istom grafiku je i simulaciona karakteristika PSK prijemnika kada on za ekstrakciju nosioca koristi PLL(kriva c). Očigledan je doprinos novopredloženog PSK prijemnika u proširenju dozvoljenog ofseta učestanosti referentnog nosioca Δf u poređenju sa PSK prijemnikom kod koga se ekstrakcija nosioca vrši PLL-om .



Slika 10.Verovatnoća greške u funkciji frekvencijskog ofseta učestanosti nosioca

(a) Novi PSK prijemnik - realizovan tehnologijom softverskog radija
(b) Novi PSK prijemnik – simulacioni rezultati
(c) PSK prijemnik koji za ekstrakciju nosioca koristi PLL

LITERATURA

[1] Matt Ettus, USRP User's and Developer's Guide, Ettus Research LLC, Online

[2] Matt Ettus, "Ettus Research LLC", http://www.ettus.com

[3] S. Glisic, Z. Nikolic, B. Dimitrijevic, "Adaptive self-reconfigurable interference suppression schemes for CDMA," *IEEE Transactions on Communications*, vol.47, no.4, pp.598-607, Apr. 1999.

[4] B.Dimitrijević, Z. Nikolić and N. Milošević, " BPSK Receiver Based on Adaptive Structure with Remodulation**",** rad prihvaćen za publikovanje