Appendix E

A Source Code of Simulation Program for the Proposed Coherent Scheme

```
/*
        This is a program for simulating the 2 stable point close-loop acquisition. The program has an
output in the file's name
        " t2coclo.txt".
*/

#include <math.h>
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

const float Tc = 0.000001;// chip rate = 1 MHz
float Y,S_signal_1,S_signal_2,Power,SNR,initialY;
int Hf;
int far *alpha;
int loop1,J,JJ,n0,sum_1,sum_2,initialJ;
int SX[9] = {1,1,1,1,1,1,1,1,1}, SR[9] = {1,1,1,1,1,1,1,1,1};
long KK, K = 511;
float KVCC,Town,Townhat,sum_34;
float far *HH;

void get_J_Y(float T1,float T2);//change the phase difference to be in term of J and Y
void forward_shift(int sh[9]);// function for shifting the shift-regester in the forward direction
void reward_shift(int sh[9]);// function for shifting the shift-regester in the reward direction
void equal_shift(int sh1[9],int sh2[9]); // function for setting sh1 by sh2
void print_shift(int sh[9]);//function for printting the value in the shift-regester
float randn(float qq);// generate noise
void initial_Hf(void);
void VCC_loop(void);//VCC loop in the receiver
void gen_alpha(void);//generate the alpha signal
void clearH(float far *H);
void set_shift(int sh1[9]);
void main()
{
        FILE  *looptime;
        float threshold,factor;

        int declare = 0,Trial,Kp,KKK,miss,false_alarm;
        long int time_1,time_acquisition,square,var;
        alpha = (int far *) farmalloc(511*sizeof(int));
        HH = (float far *) farmalloc(10000*sizeof(float));
        Power = 1;
        printf("Enter the value of integral lenght = ");
        scanf("%d",&n0);
        printf("Enter the value of SNR = ");
        scanf("%f",&SNR);
        printf("Enter the value of penalty time = ");
        scanf("%d",&Kp);
```

```
        printf("Enter the value of trial = ");
        scanf("%d",&Trial);
        printf("Enter the factor of threshold = ");
        scanf("%f",&factor);
        printf("Enter the filter 's size = ");
        scanf("%d",&Hf);
        threshold = (float)Tc*sqrt(2*Power)*n0/factor;
        time_acquisition = 0;
        miss = 0;
        false_alarm = 0;
        square = 0;
        KVCC = 1/(sqrt(2*Power)*1.0*n0);
        printf("program is running");
        gen_alpha();

for(KKK = 1;KKK <= Trial; KKK++)
{

        time_1 = 0;
        declare = 0;
        Townhat = 0;
        sum_34 = 0;
        clearH(HH);
        KK = 0;
        set_shift(SX);
        Town = (KKK%511 + 0.5)*Tc;
        get_J_Y(Townhat,Town);
        initialJ = J;
        initialY = Y;
        printf("J = %d",J);
        //initial_Hf();
        while (declare == 0)
        {
                get_J_Y(Townhat,Town);
                //printf("\nJ = %d ",J);
                //printf("\nTownhat = %f  Town  = %f",Townhat,Town);
                sum_1 = 0;
                sum_2 = 0;
                equal_shift(SR,SX);

                if(J <= 0)
                        {
                                for(loop1 = 0; loop1 < (-1)*J; loop1++)reward_shift(SR);
                        }
                else
                        {
                                for(loop1 = 0; loop1 < J; loop1++)forward_shift(SR);
                        }
        /***************branch 1 ************************/
                if(Y >= 0)
                {
                for (loop1 = 0; loop1 < n0 ; ++loop1)
                        {  reward_shift(SR);
                                sum_2 = sum_2 + SX[8]*SR[8];
                                forward_shift(SR);
                                sum_1 = sum_1 + SX[8]*SR[8];
                                forward_shift(SX);
```

```
                                forward_shift(SR);
                        }
                }
                else
                {
                                for (loop1 = 0; loop1 < n0 ; ++loop1)
                        {  forward_shift(SR);
                                sum_2 = sum_2 + SX[8]*SR[8];
                                reward_shift(SR);
                                sum_1 = sum_1 + SX[8]*SR[8];
                                forward_shift(SX);
                                forward_shift(SR);
                        }
                }
for(loop1 =0; loop1 <n0; loop1++)reward_shift(SX);
time_1 = time_1 + n0;
S_signal_1 =  sqrt(2*Power)*Tc*(sum_1*(1-fabs(Y)) + sum_2*fabs(Y)) + Tc*randn((double)
(2*Power*n0/SNR));
//printf("\nSignal = %f\nThreshold = %f", S_signal, threshold);
//getch();

/*****************branch 2**********************/
                Townhat = Townhat - 255.5*Tc;
                get_J_Y(Townhat,Town);
                Townhat = Townhat + 255.5*Tc;
                equal_shift(SR,SX);

                if(J <= 0)
                        {
                                for(loop1 = 0; loop1 < (-1)*J; loop1++)reward_shift(SR);
                        }
                else
                        {
                                for(loop1 = 0; loop1 < J; loop1++)forward_shift(SR);
                        }
        sum_1 = 0;
        sum_2 = 0;
        if(Y >= 0)
                {
                for (loop1 = 0; loop1 < n0 ; ++loop1)
                        {  reward_shift(SR);
                                sum_2 = sum_2 + SX[8]*SR[8];
                                forward_shift(SR);
                                sum_1 = sum_1 + SX[8]*SR[8];
                                forward_shift(SX);
                                forward_shift(SR);
                        }
                }
                else
                {
                                for (loop1 = 0; loop1 < n0 ; ++loop1)
                        {  forward_shift(SR);
                                sum_2 = sum_2 + SX[8]*SR[8];
                                reward_shift(SR);
                                sum_1 = sum_1 + SX[8]*SR[8];
                                forward_shift(SX);
                                forward_shift(SR);
```

```
                }
        }

for(loop1 =0; loop1 <n0; loop1++)reward_shift(SX);
S_signal_2 =  sqrt(2*Power)*Tc*(sum_1*(1-fabs(Y)) + sum_2*fabs(Y)) + Tc*randn((double)
(2*Power*n0/SNR));


/**************check threshold ************/
        get_J_Y(Townhat,Town);
        JJ = J%511;
        if ((S_signal_1 >S_signal_2) && (S_signal_1 >=threshold))
                {
                if ( J%511 == 0)
                        {
                                        declare = 1;
                                        printf("\nhi 1    time = %ld",time_1);

                        }
                else
                        {
                                        time_1 = time_1 + n0*Kp;
                                        printf("\nfalse alarm1");
                                        false_alarm++;

                        }
                }
        else if((S_signal_2 >S_signal_1) && (S_signal_2 >=threshold))
                {
                if ( (fabs(JJ+Y)>255)&&(fabs(JJ+Y)<256) )
                        {
                                        declare = 1;
                                        printf("\nha 2  time = %ld",time_1);

                        }
                else
                        {
                                        time_1 = time_1 + n0*Kp;
                                        printf("\nfalse alarm2   " );
                                        false_alarm++;
                        }
                }
        else
                {
                if (J%511 == 0 ||(fabs(JJ+Y)>255)&&(fabs(JJ+Y)<256))
                        {
                                printf("\nmiss detection");
                                miss++;
                        };
                }
        VCC_loop();
 }//end while
square = square + ((long)time_1/1000)*((long)time_1/1000);
time_acquisition = time_acquisition + ((long)time_1/Trial);

}//end for
var =  (long)square/Trial - (long)(time_acquisition/1000)*(time_acquisition/1000);
```

```
printf("\nacquisition time = %ld",time_acquisition);
printf("\nsquare =%ld",square);
printf("\nvariance = %ld",var);
printf("\nmiss detection = %d",miss);
printf("\nfalse alarm = %d",false_alarm);



/***************write to file  ************/

        if((looptime = fopen("t_x2clo.txt","w")) == NULL)
        {
        printf("Error ya");
        exit(1);
        }

 fprintf(looptime,"The result of 2 stable point close loop acquisition");
 fprintf(looptime,"\nThe filter size = %d",Hf);
 fprintf(looptime,"\nmean time = %ld\nvar = %ld\nfalse alarm = %d\nmiss detection =
%d",time_acquisition,var,false_alarm,miss);
 fprintf(looptime,"\nthreshold value = %f",threshold);

        fclose(looptime);

}//end main



void get_J_Y(float T1,float T2)
{
float dif_phase = 0;
int sign;
                dif_phase = fabs(T1 - T2);
                if (T1 -T2 >= 0)sign = 1;
                else sign = -1;
                J = dif_phase/Tc;
                Y = (float)fmod(dif_phase,Tc)/Tc;
                if (Y >= 0.5)
                {
                 Y = Y - 1;
                 J = J + 1;
                }
                Y = (float)Y*sign;
                J = (int)J*sign;
}



void equal_shift(int sh1[9],int sh2[9])
{
 int i;
 for(i = 0; i < 9 ; i++)
                sh1[i] = sh2[i];
}
void print_shift(int sh[9])
{
```

```
        int i;
        printf("\n");
        for(i = 0; i < 9;i++) printf("%d ",sh[i]);
}

void forward_shift(int sh[9])
{
        int doomy,i;
        doomy = (-1)*sh[8]*sh[3];
        for(i = 8;i>=1;--i)
                sh[i]=sh[i - 1];
        sh[0]=doomy;
}
void reward_shift(int sh[9])
{
        int doomy,i;
        doomy = (-1)*sh[4]*sh[0];
        for(i = 1;i<=8;i++)
                sh[i-1]=sh[i];
        sh[8]=doomy;
}

float randn(float qq)
{

        float PI,B,XIN,R,revalue;
        PI = 4.0*atan(1.0);
        B  = (float)(( rand() % 1000 ) / 1000.)*2*PI;
        XIN = (float)(( rand() % 1000 ) / 1000.);
        if(qq < 0.0)
        {
                printf("qq = %f", qq);
        //      getch();
        }
        R = sqrt(2.0*qq*log(1.0/(1.0 - XIN)));
        revalue = R*cos(B);
        return(revalue);
}

void initial_Hf(void)
{
                int sum_3,sum_4,sum_5,sum_6,BB,i;
                float zz,doomy,LLL;
                for(i = 0; i < Hf; i++) reward_shift(SX);
                for (i = 0; i < Hf; i++)
                {
                        get_J_Y(Townhat,Town);
                        sum_3 = 0;
                        sum_4 = 0;
                        sum_5 = *(alpha+((20*K + i+J -Hf)%K));
                        sum_6 = *(alpha+((20*K + 1+i+ J - Hf)%K));
                        sum_3 =  (0.5+Y)*SX[8]*sum_5;
                        sum_4 =  (0.5-Y)*SX[8]*sum_6;
                        forward_shift(SX);
                        zz = (float)(sum_5*sum_5*(0.5+Y) + sum_6*sum_6*(0.5-Y))/SNR;
                //      printf("\nzz = %f sum_5 = %d  sum_6 = %d  KK =
%d",zz,sum_5,sum_6,KK);
```

```
                    LLL = randn(zz);
                    *(HH+i)= sqrt(2*Power)*Tc*((sum_3 + sum_4) + LLL)/Hf;
                    sum_34 = sum_34 + (*(HH+i));
                }

}

void VCC_loop(void)
{
                int sum_3,sum_4,sum_5,sum_6,BB,i;
                float zz,doomy,LLL;
                for (i = 0; i < n0; i++)
                {
                        get_J_Y(Townhat,Town);
                        sum_3 = 0;
                        sum_4 = 0;
                        sum_5 = *(alpha+((K + KK+J)%K));
                        sum_6 = *(alpha+((K + 1+KK+J)%K));
                        sum_3 =  (0.5+Y)*SX[8]*sum_5;
                        sum_4 =  (0.5-Y)*SX[8]*sum_6;
                        forward_shift(SX);
                        BB = KK%Hf;
                        doomy = *(HH+BB);
                        zz = (float)(sum_5*sum_5*(0.5+Y) + sum_6*sum_6*(0.5-Y))/SNR;
                        //printf("\nzz = %f sum_5 = %d  sum_6 = %d  KK =
%d",zz,sum_5,sum_6,KK);
                        LLL = randn(zz);
                        *(HH+BB)= sqrt(2*Power)*Tc*((sum_3 + sum_4) + LLL)/Hf;
                        sum_34 = sum_34 + (*(HH+BB)) - doomy;
                        Townhat = Townhat + sum_34*KVCC;
                        KK++;
                        //printf("\nk = %ld  J = %d  doomy = %e",KK,J,doomy);
                }

}



void gen_alpha(void)
{
int shb[9] = {1,1,1,1,1,1,1,1,1},i,mm;
for (i = 0;i<511;i++)
        {
         *(alpha+i) =  0;
         set_shift(shb);
         for(mm = 0 ; mm < i; mm++)forward_shift(shb);
         forward_shift(shb);
         for(mm = 1; mm <= 127; mm++)
                {
                        *(alpha+i) = *(alpha + i) + shb[8];
                        forward_shift(shb);
                }
         for(mm = 1; mm <= 127; mm++)
                {
                        *(alpha+i) = (*(alpha+i))- shb[8];
                        forward_shift(shb);
```

```
                }
        forward_shift(shb);
        for(mm = 1; mm <= 127; mm++)
                {
                        *(alpha+i) = (*(alpha+i))+ shb[8];
                        forward_shift(shb);
                }
        for(mm = 1; mm <= 127; mm++)
                {
                        *(alpha+i) = (*(alpha+i))- shb[8];
                        forward_shift(shb);
                }
       printf("\nalpha = %d",*(alpha + i));
       }

}

void  clearH(float far *H)
{
 int i;
 for ( i = 0 ; i < 10000; i++)
        *(H+i) = 0.0 ;
}
void set_shift(int sh1[9])
{
        int i;
        for(i = 0; i < 9 ; i++)
                sh1[i] = 1;
}
```