R.K.MALIK

(Correlator Group)

(Ref: rkm9/MACSIM/sept92)

SOFTWARE SIMULATOR FOR THE ASIC

(*MAC MODE*)

INTRODUCTION: In order to validate the ASIC based hardware, it is necessary to predict the output for a given input to the ASIC. To achieve this, a software simulator has been written which simulates the flow of data through the ASIC and performs Gate-Level Arithmetic operations explicitly. I won't go into the details of the arithmetic which has been described in [1].

DESCRIPTION: The subroutine attached alongwith has been used for testing the operation of the ASIC in the MAC mode. Initially, some test patterns supplied by Chuck from NRAO were used to validate the software. Once the hardware became ready, it was found to produce same results as predicted by the simulator.

One has to give the following inputs to the subroutine (*"/home/rakesh/test/mac.c"* :

ramr[] : Array of the bit pattern of the Real Part of the Accumulated result

rami[] : Array of the bit pattern of the Imaginary Part of the Accumulated result

sumexp[] : Array of the bit pattern of the Exponent Part of the Accumulated result

rr[] : Array of the bit pattern of the Real Part of the number entering the NORMAL PORT of the ASIC.

ri[] : Array of the bit pattern of the Imaginary Part of the number entering the NORMAL PORT of the ASIC.

re[] : Array of the bit pattern of the Exponent Part of the number entering the NORMAL PORT of the ASIC.

tr[] : Array of the bit pattern of the Real Part of the number entering the TWIDDLE PORT of the ASIC.

ti[] : Array of the bit pattern of the Imaginary Part of the number entering the TWIDDLE PORT of the ASIC.

te[] : Array of the bit pattern of the Exponent Part of the number entering the TWIDDLE PORT of the ASIC.

The subroutine returns the accumulated result in the arrays ramr[], rami[] and sumexp[].

In order to achieve clarity in the program and make it possible for others also to understand it, each bit has been represented as one character and also the symbols used bear names very similar to the ones in the ASIC Documents. All the computations have been performed explicitly as they occur in the ASIC and therefore the user can, by suitably inserting print statements in the program, can monitor the fate of the data as it

passes through different arithmetic blocks like Adders, Shifters etc etc.

A sample program *" /home/rakesh/test/macmain.c"* is also attached alongwith to serve as a guide for using the MAC Simulator Subroutine. This program reads two numbers from the file *"in.dat"*, accumulated result from the file *"ram.dat"* and performs accumulations as many times as specified while running the program. The resulting bit-patterns are written back in *"ram.dat"*. This is however a very restrictive example and with a little imagination, the subroutine *"mac.c"* can be made to do much more as per the requirements.

COMMENTS: Another stage of ADDERS and SHIFTERS has been added for simulating FFT also. Vivek is debugging it right now and soon it will be available.

ACKNOWLEDGEMENTS: Thanks to Chuck@NRAO for supplying some test patterns and clarifying some of the aspects of the ASIC.

REFERENCES:

[1] R.K.Malik, V.M Tatke : *Digital Arithmetic Inside the FX-ASIC"*,rkm7/FXARITH/july92

```
#include <stdio.h>
#include <math.h>
void mac(ramr,rami,sumexp,rr,ri,re,tr,ti,te)

int ramr[],rami[],sumexp[],rr[],ri[],re[],tr[],ti[],te[];

{
        int i,l;
        int dum,dum1,dum2, r1r2sign,ili2sign,r1i2sign,i1r2sign;
        int r1r2mag,ili2mag,r1i2mag,i1r2mag;
        int r1r2[12],ili2[12],r1i2[12],i1r2[12];
        int r1mag,r2mag,i1mag,i2mag;
        int realmag,immag,real[12],im[12];
        int xmult[7],xmultmag,sumexpmag,c4,c5,shift,sum5;
        int rexp,texp,xexpmag,sign;
        int sm[7];
        int ARmag,AImag,AR[16],AI[16],ACR,WYI;
        int ramrmag,ramimag;
        int kk1,mp;
        int FLAGR,FLAGI;

        ri[6]=ri[6]^1;

        r1r2sign = rr[6]^tr[4];
        ili2sign = (ri[6]^1)^ti[4];
        r1i2sign = rr[6]^ti[4];
        i1r2sign = ri[6]^tr[4];
        r2mag=0;
        i2mag=0;

                for(i=0;i<4;i++)
                {
                r2mag= r2mag + tr[i]*(int)pow(2.0,(double)i);

                i2mag= i2mag + ti[i]*(int)pow(2.0,(double)i);

                }


       r1mag=0;
       i1mag=0;


       for(i=0;i<6;i++)
       {
       r1mag= r1mag + rr[i]*(int)pow(2.0,(double)i);
       i1mag= i1mag + ri[i]*(int)pow(2.0,(double)i);
                }



                r1r2mag = r1mag * r2mag;
                ili2mag = i1mag * i2mag;
                r1i2mag = r1mag * i2mag;
                i1r2mag = r2mag * i1mag;

       dum = 1;
       for(i=0;i<10;i++)
       {
       i1i2[i] = (i1i2mag>>i)&dum;
       r1i2[i] = (r1i2mag>>i)&dum;
       i1r2[i] = (i1r2mag>>i)&dum;
       }
       r1r2[10]=r1r2sign;
       ili2[10]=ili2sign;
       r1i2[10]=r1i2sign;
       i1r2[10]=i1r2sign;

       for(i=0;i<10;i++)
       {
       dum=r1r2sign & 1;
                        r1r2[i] = r1r2[i]^dum ;
                        dum=ili2sign & 1;
                        ili2[i] = ili2[i]^dum ;
                dum=r1i2sign & 1;
                r1i2[i] = r1i2[i]^dum ;
                dum=i1r2sign & 1;
                i1r2[i] = i1r2[i]^dum ;
                }

                dum = 1;

                r1r2[10] = r1r2sign;

                ili2[10] = ili2sign;
                r1i2[10] = r1i2sign;
                i1r2[10] = i1r2sign;


                r1r2[11]=r1r2[10];
                ili2[11]=ili2[10];
                r1i2[11]=r1i2[10];
                i1r2[11]=i1r2[10];

                r1r2mag = 0;
                ili2mag = 0;
                r1i2mag = 0;
                i1r2mag = 0;

                for(i=0;i<12;i++)
                {
                r1r2mag = r1r2mag + r1r2[i]*(int)pow(2.0,(double)i)
                ili2mag = ili2mag + ili2[i]*(int)pow(2.0,(double)i)
                r1i2mag = r1i2mag + r1i2[i]*(int)pow(2.0,(double)i)
                i1r2mag = i1r2mag + i1r2[i]*(int)pow(2.0,(double)i)
                }

                realmag = r1r2mag + i1i2mag;
                dum1=realmag;
                dum=1;
                dum2=12;
                dum1=dum1>>dum2;
                dum1=dum1 & dum;
                realmag=realmag + dum1;

                immag = r1i2mag + i1r2mag;
                dum1=immag;
                dum=1;
```

```
        dum1=dum1 & dum;
immag=immag + dum1;

dum=1;
for(i=0;i<12;i++)
{
real[i] = (realmag>>i) & dum;
im[i] = (immag>>i) & dum;
}


rexp=0;
        texp=0;
        for(i=0;i<4;i++)
        {
        rexp=rexp + re[i]*(int)pow(2.0,(double)i);
                texp=texp + te[i]*(int)pow(2.0,(double)i);
        }

xexpmag=rexp + texp;

dum=1;
for(i=0;i<5;i++)
{
xmult[i]=(xexpmag>>i) & dum;
}

xmult[5] = 0;
xmult[6] = 0;
sumexp[6]=sumexp[5];  /* Sign Extention */
        sumexpmag = 0;   /*        ..........  */
        for(i=0;i<7;i++)
        {
        sumexpmag = sumexpmag + sumexp[i]*(int)pow(2.0,(double
        }

        shift=xexpmag + sumexpmag; /* 2's COMP Addition */
        dum=1;
for(i=0;i<7;i++)
{
sm[i] = (shift>>i)&dum;
}


dum=1;
dum1=5;
dum1=shift>>dum1;
c4=dum1 & dum;
        sum5=sumexp[5]^c4 ;



        c5=c4 & sumexp[5];
        sign= sm[6];

for(i=0;i<6;i++)
{
sm[i] = sm[i]^sign; /* 1's COMP */
```

```
{
shift = shift + sm[i]*(int)pow(2.0,(double)i); /* Mag of SH
}


if(sign==1 & shift!=0)  /* Negative Shifter */
{
dum=ramr[14]; /* Store the Sign of the Mantissa */
dum1=rami[14];

if(shift>16)
{
for(i=0;i<15;i++)
                        {
                        ramr[i]=0;
                        rami[i]=0;
        }
        }
        if(shift<15)
        {
        for(i=0;i<(15-shift);i++)
        {
        ramr[i]=ramr[i+shift];
        rami[i]=rami[i+shift];
        }
        for(i=0;i<shift;i++)
        {
        ramr[14-i]=dum;
        rami[14-i]=dum1;
        }
        }

        if(shift<17 & shift>14)
        {
        for(i=0;i<15;i++)
        {
        ramr[i] = dum;



        rami[i] = dum1;
        }
        }
        }

        dum=real[11];
        dum1=im[11];

        for(i=3;i<15;i++)



        {
        AR[i]=real[i-3];
        AI[i]=im[i-3];
        }
        for(i=0;i<3;i++)
        {
        AR[i]=dum;
```

```
        dum=AR[14]; /* Store the Sign of the Mantissa */
dum1=AI[14];

if(shift>15)
{
for(i=0;i<15;i++)
{
AR[i]=0;
AI[i]=0;
}
}
if(shift<15)
        {
        for(i=0;i<(15-shift);i++)
        {
        AR[i]=AR[i+shift];
                AI[i]=AI[i+shift];
                }
                for(i=0;i<shift;i++)
{
AR[14-i]=dum;
AI[14-i]=dum1;
}
}


if(shift<16 & shift>14)
{
for(i=0;i<15;i++)
{
                AR[i] = dum;
                AI[i] = dum1;
                }
        }
        }



ramr[15]=ramr[14];    /*Sign Extention */
rami[15]=rami[14];
AR[15]=AR[14];
AI[15]=AI[14];

FLAGR=1;
FLAGI=1;
for(l=0;l<16;l++)
{
if(FLAGR!=0)
        {



                if((AR[l]^ramr[l])!=1)
                FLAGR=0;
                }

if(FLAGI!=0)
{
if((AI[l]^rami[l])!=1)
```

```
        ramrmag=0;
        ramimag=0;

        ARmag=0;
        AImag=0;
        for(i=0;i<16;i++)
        {

        ramrmag=ramrmag + ramr[i]*(int)pow(2.0,(double)i);
        ramimag=ramimag + rami[i]*(int)pow(2.0,(double)i);

        ARmag  =ARmag + AR[i]*(int)pow(2.0,(double)i);
                        AImag  =AImag + AI[i]*(int)pow(2.0,(
                }
                ACR=ramrmag + ARmag;
                WYI=ramimag + AImag;
                dum=1;
                dum1=ACR;
                dum2=16;

                dum1=dum1>>dum2;
                dum1=dum1&dum;
                ACR=ACR+dum1;

                dum=1;
                dum1=WYI;
                dum2=16;

                dum1=dum1>>dum2;
                dum1=dum1&dum;
                WYI=WYI+dum1;

/* ..................................................*/
                dum=1;
                for(i=0;i<16;i++)
                {
                AR[i]=(ACR>>i) & dum;
                AI[i]=(WYI>>i) & dum;
                }

                if(FLAGR!=0)
                {
                for(l=0;l<16;l++)
                AR[l]=0;
                }
                if(FLAGI!=0)
                {
                for(l=0;l<16;l++)
                AI[l]=0;
                }




                dum=AR[14]^AR[15];
                dum1=AI[14]^AI[15];
                dum=dum^1;
                dum1=dum1^1;
```

```
                rami[i]=AI[i+shift];
        }


        dum=1;
        for(i=0;i<6;i++)
        {
        xmult[i] = xmult[i]^dum; /* COMP */
        }

        xmultmag = 0;
        for(i=0;i<6;i++)
                {
                xmultmag=xmultmag + xmult[i]*(int)pow(2.0,(double)i);
                }

                xmultmag = xmultmag+1;

                for(i=0;i<6;i++)
        {
        xmult[i]=(xmultmag>>i) & dum;
        }

        if(sign==1)
        {
        for(i=0;i<6;i++)
        {
        sumexp[i]=xmult[i];
        }
        }

                sumexpmag=0;
                for(i=0;i<6;i++)
                {
                sumexpmag=sumexpmag + sumexp[i]*(int)pow(2.0,(double)i
                }
                sumexpmag=sumexpmag+shift;

        dum=1;
        for(i=0;i<6;i++)
        sumexp[i]=(sumexpmag>>i)&dum;

}

/* ..... */
```

```
/* ... /home/rakesh/test/macmain.c ..... */
#include <stdio.h>
#include <math.h>

main()
{
        int i,niter,mp;
        int rr[7],ri[7],re[4],tr[5],ti[5],te[4];
        int ramr[16],rami[16],sumexp[7];

        FILE *fd1, *fd2, *fd3, *fd4, *fd5;
        fd3=fopen("/dev/tty","w");
        fd4=fopen("res.dat","w");
        fd5=fopen("/dev/tty","r");
        fprintf(fd3,"No of Accumulations = ?\n");
        fscanf(fd5,"%d",&niter);

        for(mp=0;mp<niter;mp++)
        {

fd1=fopen("in.dat","r");
fd2=fopen("ram.dat","r");
        for(i=0;i<15;i++)
        {
        fscanf(fd2,"%d",&ramr[14-i]) ;
        }

        for(i=0;i<15;i++)
        {
        fscanf(fd2,"%d",&rami[14-i]) ;
        }

                for(i=0;i<6;i++)
                {
                        fscanf(fd2,"%d",&sumexp[5-i]);
                }

                fclose(fd2);
                tr[0] = 0;
                ti[0] = 0;
        for(i=0;i<3;i++)
        {
        rr[i]=0;
        ri[i]=0;
        }

        for(i=0;i<4;i++)
        fscanf(fd1,"%d",&re[3-i]);

        for(i=0;i<4;i++)
                fscanf(fd1,"%d",&ri[6-i]);

                for(i=0;i<4;i++)
                fscanf(fd1,"%d",&rr[6-i]);

                for(i=0;i<4;i++)
                        fscanf(fd1,"%d",&te[3-i]);

                for(i=0;i<4;i++)
        fscanf(fd1,"%d",&ti[4-i]);
        fscanf(fd1,"%d",&tr[4-i]);


mac(ramr,rami,sumexp,rr,ri,re,tr,ti,te) ;


        fd2=fopen("ram.dat","w");
        for(i=0;i<15;i++)
        {
        fprintf(fd2,"%d ",ramr[14-i]);
        }


                fprintf(fd2,"\n");
                for(i=0;i<15;i++)
                {

                fprintf(fd2,"%d ",rami[14-i]);
                }

                fprintf(fd2,"\n");
                for(i=0;i<6;i++)
                {
                fprintf(fd2,"%d ",sumexp[5-i]);
                }
                fclose(fd1);
                fclose(fd2);
        }

                printf("\n\n");
                printf("RESULTS (MSB.....
                fprintf(fd4,"\t RAMA\t");
                printf("\t RAMA\t");

                fprintf(fd4,"\t\t\tRAMB\n");
                printf("\t\t\tRAMB\n");
                for(i=0;i<3;i++)
                {
                fprintf(fd4,"%d",sumexp[2-i]);
                printf("%d",sumexp[2-i]);
                }
                fprintf(fd4," ");
                printf(" ");
                for(i=0;i<15;i++)
                {
                fprintf(fd4,"%d",ramr[14-i]);
                printf("%d",ramr[14-i]);

                if((i==2)|(i==6)|(i==10))
                {
                fprintf(fd4," ");
                printf(" ");
                }
                }
                fprintf(fd4,"\t");
                printf("\t");

                for(i=0;i<3;i++)
                {
                fprintf(fd4,"%d",sumexp[5-i]);
```

```
                }
                fprintf(fd4," ");
                printf(" ");
                for(i=0;i<15;i++)
                {
                fprintf(fd4,"%d",rami[14-i]);
                printf("%d",rami[14-i]);

                if((i==2)||(i==6)||(i==10))
                {
                fprintf(fd4,"   ");
                printf("   ");
                        }
                        }
                        fprintf(fd4,"\n\n");
                        printf("\n\n");
}
```