UN MINI CORRELATORE

PER PC A 16 LAGS REALI

G. Tuccari
S. Buttaccio

Rapporto Tecnico IRA n. 220/96

1. Introduzione.

E' stato realizzato un sistema in grado di calcolare la funzione di cross-correlazione e di auto-correlazione in real time di segnali con banda massima di 5 MHz. Tale funzione viene stimata in parallelo per 16 differenti ritardi temporali relativi tra i due segnali. Lo sviluppo di questo strumento e' nato dalla esigenza di poter valutare tali funzione per segnali presenti in diversi punti della catena di acquisizione VLBI nella stazione di Noto per ragioni e di testing e di caratterizzazione. Il sistema opera attraverso un PC ed e` costituito da una scheda installata sul bus a 16-bit del PC e dal relativo software di gestione. Questo documento descrive la funzionalita' della parte hardware e di quella software e alcune applicazioni utilizzate a Noto.

2. L'hardware.

La scheda di correlazione e' realizzata in wire wrapping e presenta il pettine per bus a 16-bit tipico per Personal Computer IBM compatibili. L'intera funzionalita` logica viene realizzata da due chip FPGA e pochi altri componenti. In allegato sono presenti gli schemi elettrici relativi a:

- la scheda di correlazione per PC;
- il chip Xilinx XC2064 U1;
- il chip Xilinx XC3064 U2;

Il chip U1 e' un dispositivo della famiglia Xilinx 2000 XC2064 la cui funzione e' molteplice. Questo infatti gestisce la comunicazione tra il bus del PC e il chip di correlazione, effettua il campionamento dei due segnali da correlare con frequenze di clock 10 - 5 - 4 MHz o attraverso un clock esterno. Tali segnali di clock vengono generati a partire da un quarzo da 20 Mhz attraverso una catena di divisione presente all'interno del chip. I segnali da correlare devono giungere ai pin di ingresso del chip squadrati a livelli TTL, mentre come detto il campionamento viene realizzato dal chip stesso. Il chip XC2064 include anche un contatore che consente di generare gli indirizzi per una eprom. I dati memorizzati su questa memoria consentono il testing del chip di correlazione ed eventuali simulazioni. Il chip U2 realizza il calcolo della funzione di correlazione dei due segnali provenienti da U1. E' possibile inserire un delay aggiuntivo nel percorso del segnale X per un massimo di tre cicli di clock, mentre il segnale Y puo' essere ritardato fino ad un massimo di 10 cicli di clock. Cio` e` utile per traslare temporalmente la funzione di correlazione e consente quindi di centrare il massimo della funzione nello spazio dei ritardi, compensando cosi' eventuali ritardi che agiascono su uno dei due segnali. Il chip di correlazione e' descritto in dettaglio in un documento dedicato. Per le applicazioni oggetto di questo rapporto tecnico il chip viene usato in modalita` reale.

Ne descriviamo brevemente il funzionameto. I segnali X e Y ritardati delle quantita` descritte, programmabili atraverso un registro interno, vengono inviati alla catena di shift registers da cui sono prelevati e motiplicati. L'esito di tale operazione viene cumulato in caso di coincidenza da parte di un contatore. Il chip in generale puo` essere suddiviso in modo da operare in modalita` complessa e in tal caso uno dei due segnali in ingresso puo` venir moltiplicato per le componenti seno e coseno di una funzione a tre livelli allo scopo di calcolare la funzione di correlazione complessa per lo stop delle frange di interferenza. La nostra applicazione, come detto, comporta operazioni a baseline zero e tali opzioni non vengono utilizzate. Il cumulo della verifica delle coincidenze avviene per il periodo massimo concesso a 256 cicli di clock e al termine di tale intervallo di tempo i risultati di integrazione sono accumulati in parallelo per 16 differenti lags. A tal punto viene emessa una interruzione sul bus del PC. I risultati vengono depositati in appositi buffers che verranno letti successivamente e i contatori azzerati per consentire un nuovo ciclo di integrazione. I dati vengono presentati sul bus come multipli interi di 16 ovvero i quattro bit meno significativi sono fissi a 0. Vengono lette 8 parole da 16-bit che rappresentano il contenuto di due lags ciascuna.

3. Software di gestione

Il chip di correlazione richiede che il contenuto dei suoi registri venga letto ogni intervallo di integrazione che e' pari a

$T_{leu} = 2^8 * T_s$ con T_s tempo di campionamento

Nel caso di T_s=250 ns l'intervallo di tempo entro cui e` necessario leggere i dati e` pari a 64μs. E` quindi necessario operare con un PC in grado di effettuare 8 letture in un tempo inferiore a questo. Un sistema relativamente obsoleto come un 80286 e` in grado di eseguire queste operazioni, ma volendo usare una banda piu` grande e` indispensabile adoperare un PC piu` veloce.

Il software opera a livello basso attraverso la gestione di una interruzione che recupera il contenuto delle integrazioni. Vengono riportati in allegato i listings per le seguenti routines:

INT.ASM

INT.FOR

SOM.FOR

OUTFILE.FOR

DIFF.FOR

La routine assembler INT.ASM si occupa di recuperare i dati provenienti dal chip di correlazione in seguito alla richiesta della interruzione hardware. Include l'handler della interruzione e la gestione di questa. Il programma INT.FOR produce un file con i dati provenienti dal chip integrati ulteriormente di 256 parole. L'integrazione totale e' quindi pari a 2^{16} bit e nel caso di campionamento a 4 MHz il tempo di integrazione e' pari a circa 16.4 ms. Il programma non potendo gestire ulteriori integrazioni su variabili a16-bit produce i risultati della integrazione su file. Una ulteriore fase di integrazione viene realizzata dal programma

SOM.FOR che fornisce anche i risultati delle integrazioni per lag in maniera grafica. Il programma OUTFILE.FOR produce diversi file contenenti i coefficienti di correlazione, le parti pari e dispari della funzione di correlazione. DIFF.FOR effettua una rimozione di fondo nei casi in cui cio` sia utile.

4. Note applicative sull'uso della scheda di cross-correlazione.

Vogliamo qui ricordare alcuni elementi che risultano utili nelle valutazioni che seguono. La funzione di cross-correlazione puo' essere descritta come

$$R(n\Delta t)=\frac{1}{K}\sum_{m=0}^{m=K-1}\chi_1(t_0+m\Delta t)\cdot\chi_2(t_0+(m+n)\Delta t), \qquad \text{n=0,1,...,N-1}$$

con K numero dei prodotti mediati e Δt incremento del ritardo temporale (solitamente uguale all'intervallo di campionamento). Tale funzione e` in generale asimmetrica rispetto al delay cosi` che il crosspower spectrum di questa e` descritto da una funzione complessa. Se abbiamo a disposizione R(-n$\Delta$t) ovvero i termini della funzione per intervalli relativi negativi, e` possibile scrivere la funzione di cross-correlazione nelle sue componenti pari e dispari

$$R^{even}(n\Delta t)=[R(n\Delta t)+R(-n\Delta t)]/2$$

$$R^{odd}(n\Delta t)=[R(n\Delta t)-R(-n\Delta t)]/2$$

Consideriamo le parti reale e immaginaria della trasformata di Fourier che dopo la normalizzazione sono

$$\mathrm{Re}(P_j)=\frac{1}{N}\left[R_0{}^{even}+2\sum_{n=1}^{N-1}R_n{}^{even}\cos\frac{\pi\cdot n\cdot j}{N}\right]$$

$$\mathrm{Im}(P_j) = \frac{2}{N}\left[\sum_{n=1}^{N-1} R_n^{odd} \sin\frac{\pi \cdot n \cdot j}{N}\right]$$

La trasformata coseno della parte pari della funzione di cross-correlazione fornisce lo spettro di potenza del parametro U di Stokes della polarizzazione, mentre la trasformata seno della parte dispari fornisce lo spettro del parametro V. In interferometria le parti pari e dispari della funzione di cross-correlazione corrispondono alle componenti seno e coseno delle frange di interferenza. E' quindi possibile realizzare delle stime strumentali di utilita' nel controllo della catena di acquisizione VLBI.

A Noto il sistema sviluppato consente di valutare il corretto funzionamento di alcune sezioni della acquisizione dati. Cio' e' particolarmente utile ad esempio allorquando si voglia verificare la funzionalita' dei Base Band Converter operando con ricevitori sprovvisti di segnale phase cal iniettato. In tal caso infatti e' possibile evidenziare la correlazione o meno della IF in una determinata porzione di banda. La stabilita' di fase scarsa in una delle sezioni della catena infatti rende impossibile la coerenza e quindi un eccesso di correlazione. La funzione di correlazione continua R_c e` legata alla funzione di correlazione ad 1-bit dalla relazione di Van Vleck

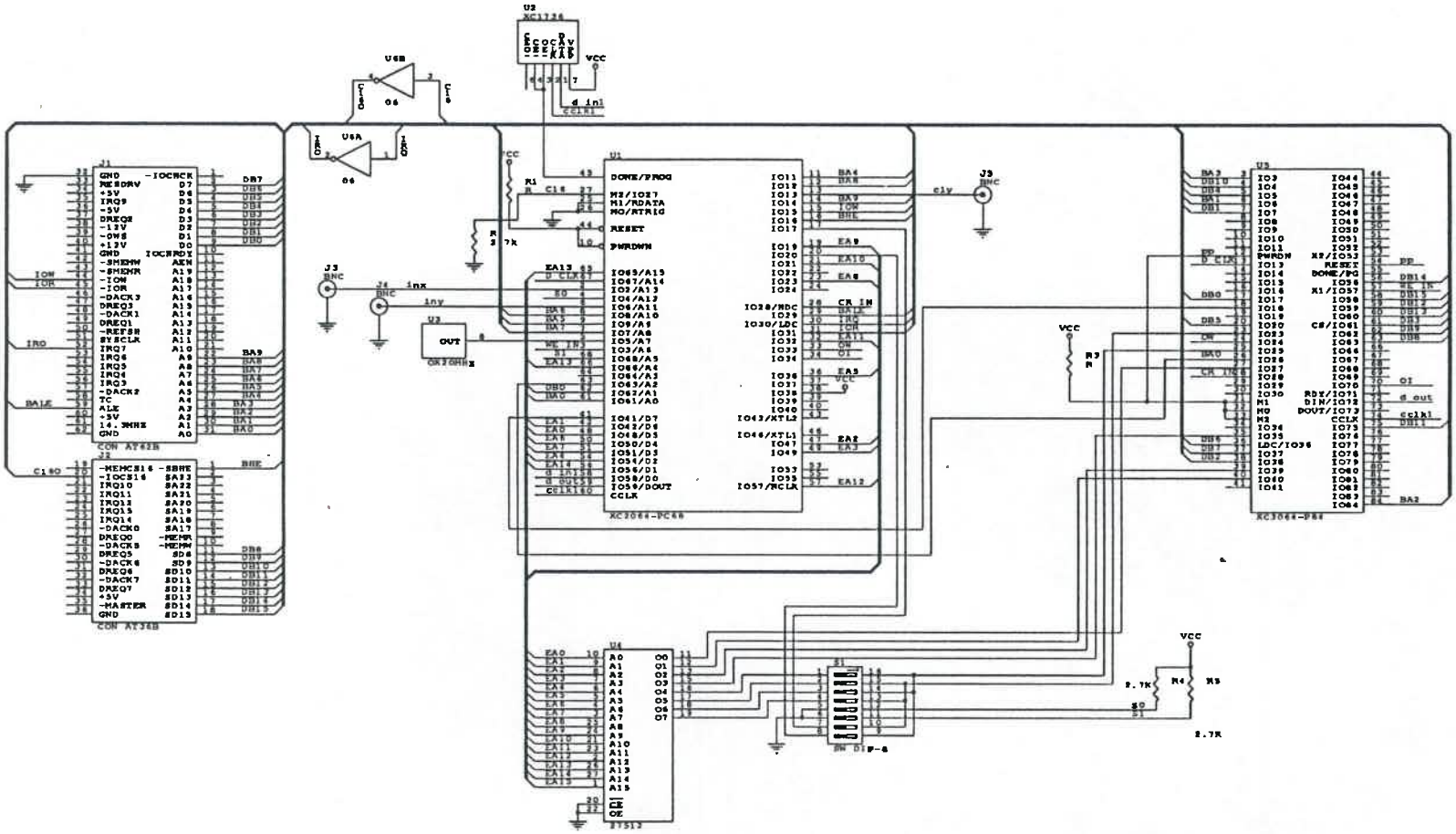
$$R_c = \sin\left(\frac{1}{2}\cdot\pi\cdot R_{1bit}\right) \quad \text{dove}$$

$$R_{1bit} = 2\frac{K_c}{K} - 1,$$

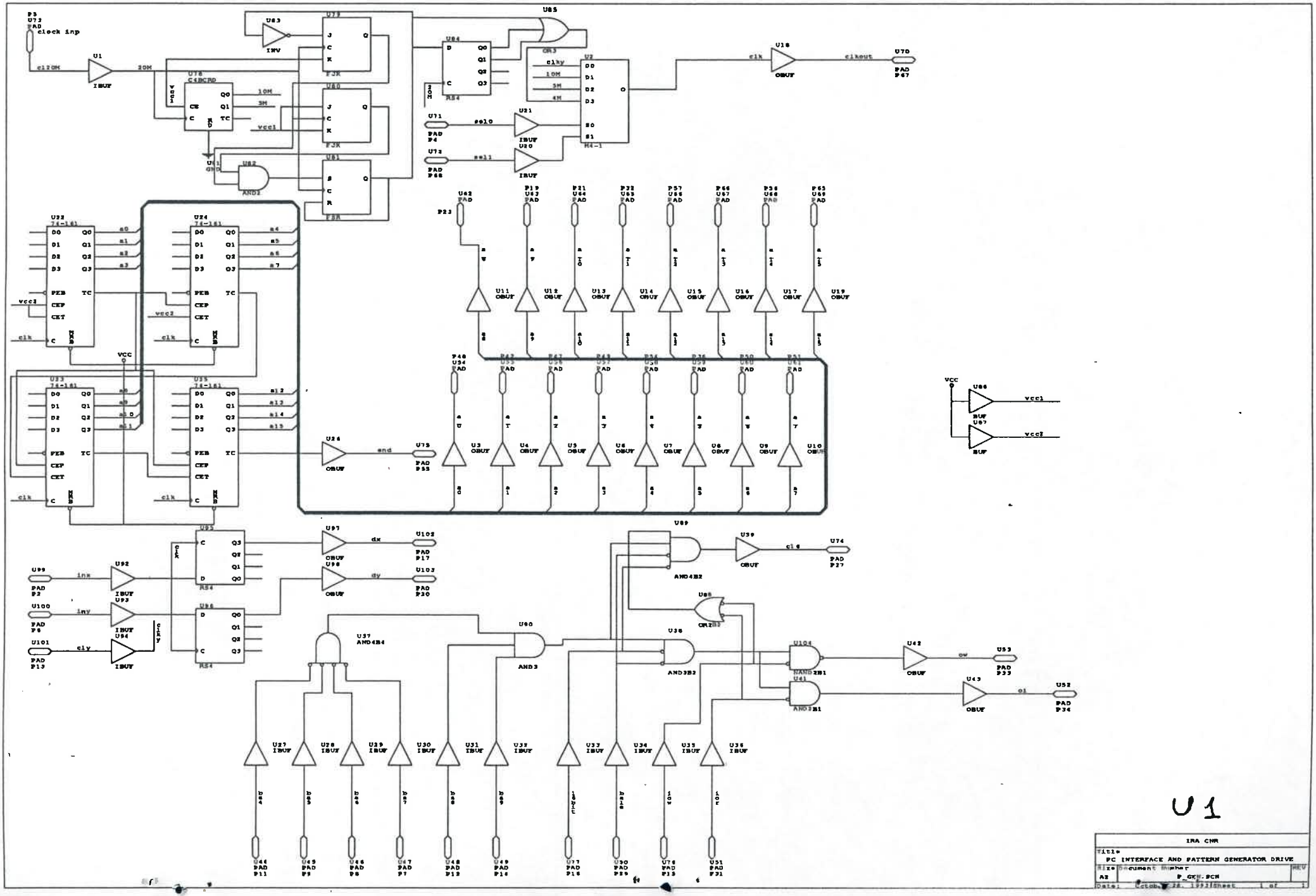
con K_c numero di coincidenze e K numero totale di campioni, e` il coefficiente di cross-correlazione per dati campionati ad 1-bit. Ricordiamo come sia dimostrato che nel caso di un segnale assimilabile a rumore a distribuzione gaussiana e` possibile esasperare il campionamento fino a valutare solo il segno. Cio' comporta una riduzione nella sensibilita` del

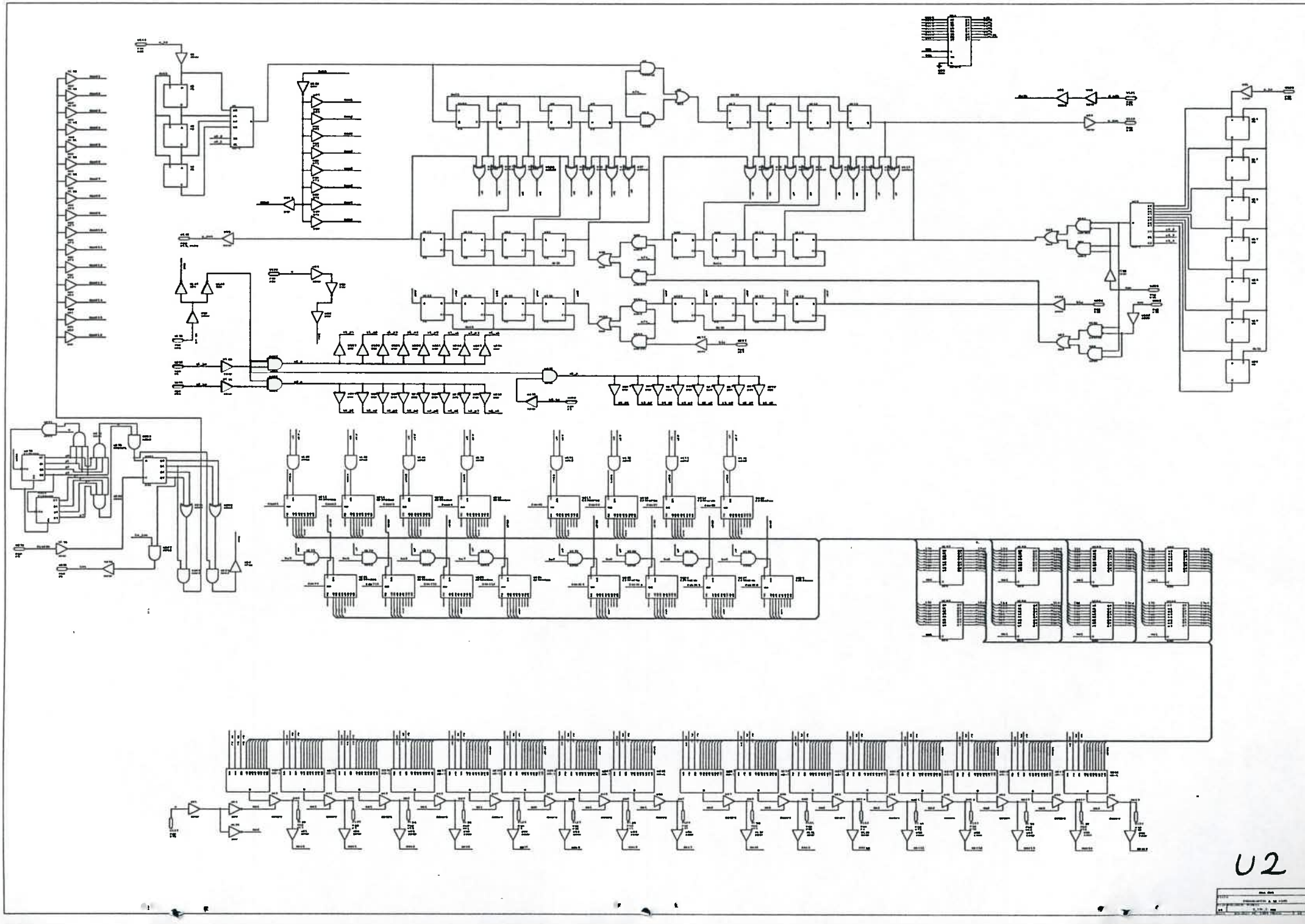
sistema rispetto al campionamento a molti livelli o del continuo che risulta pari al 64% rispetto a quest'ultimo.

La scheda di correlazione puo' essere usata per stimare l'entita` di cross-talk del sistema di acquisizione e in particolare di cross-polarizzazione del ricevitore. La correlazione a parita` di frequenza delle IF provenienti dalle due polarizzazioni Left e Right dei ricevitori consente infatti di stimare con buona approssimazione il contributo comune ai due segnali.



Title	IRA CNR	
	PC CORRELATOR BOARD	
Size: A2	Document Number: CR_TEST.SCH	REV:
Date:		Sheet 1 of 1


clock inp
clk20M
20M
10M
5M
vcc1
vcc2
clk
clkout
sel0
sel1
end
dx
dy
inx
iny
clr
ow
oi
VCC
IRA CNR
PC INTERFACE AND PATTERN GENERATOR DRIVE
P_GEN.SCH
October 28, 1993
U1



U2

PROGRAM INT.ASM

```
        .model small
        .data
        PUBLIC vector,maskold,flag,ferma,nread,modo,co1,co2,co3,co4,co5,co6
        PUBLIC co7,co8,co1a,co2a,co3a,co4a,co5a,co6a,co7a,co8a

nread   dw      ?       ; int da servire   (si setta da prog)
modo    dw      ?       ; modo corr.       (       "       )

vector  dd      ?
maskold db      ?
flag    dw      0h      ; int serviti
ferma   db      0h
co1  dw     0h
co2  dw     0h
co3  dw     0h
co4  dw     0h
co5  dw     0h
co6  dw     0h
co7  dw     0h
co8  dw     0h
co1a  dw     0h
co2a  dw     0h
co3a  dw     0h
co4a  dw     0h
co5a  dw     0h
co6a  dw     0h
co7a  dw     0h
co8a  dw     0h

        .code
        PUBLIC start,hand,res,setta

start   proc FAR
        mov ax,@data
        mov ds,ax

        mov ax,350fh
        int 21h
        mov WORD PTR vector[2],es
        mov WORD PTR vector[0],bx
        ; legge il vettore di int. attuale e lo mette in vector

        push ds
        mov ax,cs
        mov ds,ax
        mov dx,OFFSET hand
        mov ax,250fh
        int 21h
        pop ds
        ; setta il vettore di int. con l'addr. della proc. hand

        in al,21h
        mov BYTE PTR maskold,al
        jmp $+2                 ;legge la int. mask attuale

;       and al,07fh
```

```
;       out21h,al    ;abilita anche int 7

        mov al,20h
        out 20h,al
        mov al,67h
        jmp $+2
        out 20h,al   ;reset per int 7 sul gestore

        mov al,07fh
        out 21h,al   ;abilita solo int 7

        mov dx,300h
        mov ax,64
        or ax,modo
        out dx,ax  ; abilita int sulla scheda

        ret
start   ENDP

hand  PROC FAR      ; int handler
        cli

        inc flag            ; service

        mov dx,300h
        in ax,dx
        mov cx,ax
        and cx,00ffh
        mov al,ah
        and ax,00ffh
        add co1,ax
        add co1a,cx

        mov dx,302h
        in ax,dx
        mov cx,ax
        and cx,00ffh
        mov al,ah
        and ax,00ffh
        add co2,ax
        add co2a,cx

        mov dx,304h
        in ax,dx
        mov cx,ax
        and cx,00ffh
        mov al,ah
        and ax,00ffh
        add co3,ax
        add co3a,cx

        mov dx,306h
        in ax,dx
        mov cx,ax
        and cx,00ffh
        mov al,ah
        and ax,00ffh
        add co4,ax
```

```
        add co4a,cx

        mov dx,308h
        in ax,dx
        mov cx,ax
        and cx,00ffh
        mov al,ah
        and ax,00ffh
        add co5,ax
        add co5a,cx

        mov dx,30ah
        in ax,dx
        mov cx,ax
        and cx,00ffh
        mov al,ah
        and ax,00ffh
        add co6,ax
        add co6a,cx

        mov dx,30ch
        in ax,dx
        mov cx,ax
        and cx,00ffh
        mov al,ah
        and ax,00ffh
        add co7,ax
        add co7a,cx

        mov dx,30eh
        in ax,dx
        mov cx,ax
        and cx,00ffh
        mov al,ah
        and ax,00ffh
        add co8,ax
        add co8a,cx         ; end service

        mov ax,nread
        cmp flag,ax
        jnz conti

        mov dx,300h
        mov ax,modo
        out dx,ax           ;inibisce int scheda
        mov ferma,1h
        sti
        iret

conti:  mov al,20h
        out 20h,al
        mov dx,2f7h
        out dx,al
        sti
        iret
hand  ENDP

res     proc far
```

```
        push ds
        push es
        cli

        mov al,BYTE PTR maskold
        out 21h,al          ; restore old int mask

        push ds
        lds dx,vector
        mov ax,250fh
        int 21h
        pop ds              ; restore old int vector

        sti
        pop es
        pop ds
        ret

res     ENDP

setta   PROC FAR

        push ds
        push es
        mov dx,300h
        mov ax,modo
        out dx,ax
        pop es
        pop es
        ret

setta   ENDP
        END
```

PROGRAM INT.FOR

```
      integer*2 nread[EXTERN]
      integer*2 modo[EXTERN]
      integer*2 flag[EXTERN]
      integer*2 co1[EXTERN]
      integer*2 co2[EXTERN]
      integer*2 co3[EXTERN]
      integer*2 co4[EXTERN]
      integer*2 co5[EXTERN]
      integer*2 co6[EXTERN]
      integer*2 co7[EXTERN]
      integer*2 co8[EXTERN]
      integer*2 co1a[EXTERN]
      integer*2 co2a[EXTERN]
      integer*2 co3a[EXTERN]
      integer*2 co4a[EXTERN]
      integer*2 co5a[EXTERN]
      integer*2 co6a[EXTERN]
      integer*2 co7a[EXTERN]
      integer*2 co8a[EXTERN]
      integer*1 ferma[EXTERN]
      jjj=0
33     flag=0
c      write (*,*) ' modo ?'
c      read (*,*) modo
c      write (*,*) ' numero di letture ?'
c      read (*,*) nread
      modo=33
      nread=255
      call setta
      call start
c      do 10 i=0,1000
c      j=j+1
c      j=i-1
10     continue
      if (ferma.ne.1) goto 10
      call res
      write (*,100) co1,co1a,co2,co2a,co3,co3a,co4,co4a,co5,co5a,
     + co6,co6a,co7,co7a,co8,co8a
100    format(1x,16(z,1x))
      if(jjj.lt.2000) then
        jjj=jjj+1
        ferma=0
        flag=0
        co1=0
        co1a=0
        co2=0
        co2a=0
        co3=0
        co3a=0
        co4=0
        co4a=0
        co5=0
        co5a=0
        co6=0
        co6a=0
        co7=0
```

```
      co7a=0
      co8=0
      co8a=0
      goto 33
    endif
    end
```

```
PROGRAM SOM.FOR      integer*4 letti(16)
      integer*4 isomme(16)
      real maxx,minn,somme(16),jfatt,jm,ssom
      character*75 linea
      character*1 dot
      dot='.'
      open(10,file='dati.dat',status='old')
      do 100 i=1,2000
      read(10,'(16(z4,1x))') letti
      do 200 j=1,16
200     isomme(j)=isomme(j)+letti(j)
100     continue
      do 300 i=1,16
      somme(i)=real(isomme(i))
      somme(i)=abs((2*somme(i)/131072000)-1)*100
      ssom=ssom+somme(i)
300     write(*,'(1x,f6.2)') somme(i)
      maxx=max(somme(1),somme(2),somme(3),somme(4),somme(5),somme(6),
     + somme(7),somme(8),somme(9),somme(10),somme(11),somme(12),
     + somme(13),somme(14),somme(15),somme(16))
      minn=min(somme(1),somme(2),somme(3),somme(4),somme(5),somme(6),
     + somme(7),somme(8),somme(9),somme(10),somme(11),somme(12),
     + somme(13),somme(14),somme(15),somme(16))
      jfatt=maxx/60
      write (*,*)
      write (*,444) maxx,minn,ssom
444     format (f6.2,' MAX',2x,f6.2,' MIN',2x,f6.2,' SUM')
      write(*,*)
      jm=0
      ii=0
      linea='.'
15      if(jm.lt.somme(1)) then
          write(linea((ii+1):),'(a)') dot
          ii=ii+1
          jm=jm+jfatt
          goto 15
        endif
      write (*,*) ' 0',linea
      jm=0
      ii=0
      linea='.'
16      if(jm.lt.somme(2)) then
          write(linea((ii+1):),'(a)') dot
          ii=ii+1
          jm=jm+jfatt
          goto 16
        endif
      write (*,*) ' 1',linea
      jm=0
      ii=0
      linea='.'
17      if(jm.lt.somme(3)) then
          write(linea((ii+1):),'(a)') dot
          ii=ii+1
          jm=jm+jfatt
          goto 17
        endif
      write (*,*) ' 2',linea
```

```
      jm=0
      ii=0
      linea='.'
18     if(jm.lt.somme(4)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 18
      endif
      write (*,*) ' 3',linea
      jm=0
      ii=0
      linea='.'
19     if(jm.lt.somme(5)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 19
      endif
      write (*,*) ' 4',linea
      jm=0
      ii=0
      linea='.'
20     if(jm.lt.somme(6)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 20
      endif
      write (*,*) ' 5',linea
      jm=0
      ii=0
      linea='.'
21     if(jm.lt.somme(7)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 21
      endif
      write (*,*) ' 6',linea
      jm=0
      ii=0
      linea='.'
22     if(jm.lt.somme(8)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 22
      endif
      write (*,*) ' 7',linea
      jm=0
      ii=0
      linea='.'
23     if(jm.lt.somme(9)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 23
```

```
      endif
      write (*,*) ' 8',linea
      jm=0
      ii=0
      linea='.'
24    if(jm.lt.somme(10)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 24
      endif
      write (*,*) ' 9',linea
      jm=0
      ii=0
      linea='.'
25    if(jm.lt.somme(11)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 25
      endif
      write (*,*) ' A',linea
      jm=0
      ii=0
      linea='.'
26    if(jm.lt.somme(12)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 26
      endif
      write (*,*) ' B',linea
      jm=0
      ii=0
      linea='.'
27    if(jm.lt.somme(13)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 27
      endif
      write (*,*) ' C',linea
      jm=0
      ii=0
      linea='.'
28    if(jm.lt.somme(14)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
         jm=jm+jfatt
         goto 28
      endif
      write (*,*) ' D',linea
      jm=0
      ii=0
      linea='.'
29    if(jm.lt.somme(15)) then
         write(linea((ii+1):),'(a)') dot
         ii=ii+1
```

```
          jm=jm+jfatt
          goto 29
        endif
        write (*,*) ' E',linea
        jm=0
        ii=0
        linea='.'
30      if(jm.lt.somme(16)) then
          write(linea((ii+1):),'(a)') dot
          ii=ii+1
          jm=jm+jfatt
          goto 30
        endif
        write (*,*) ' F',linea
1200    end
```

PROGRAM OUTFILE.FOR

```
      real*4 f(16),reven(8),rodd(8),amp(8),ph(8)
      integer*4 g,letti(16)
      integer*4 somme(16)
      character*75 linea
      character*1 dot
      dot='.'
      open(10,file='dati.dat',status='old')
      open(2,file='xcorr.dat')
      open(4,file='xeven.dat')
      open(5,file='xodd.dat')
      open(6,file='xamp.dat')
      open(7,file='xph.dat')
      do 100 i=1,2000
      read(10,'(16(z4,1x))') letti
      do 200 j=1,16
200     somme(j)=somme(j)+letti(j)
100     continue
      do 300 i=1,16
300     f(i)=real(somme(i))
      do 400 i=1,16
      f(i)=(2*f(i)/131072000)-1
400     write(2,*) f(i)
      write(2,*)
      do 500 i=0,7
      reven(i+1)=(f(9+i)+f(9-i))/2
500     rodd(i+1) =(f(9+i)-f(9-i))/2
      do 10 i=1,8
10     write(4,*) reven(i)
      do 20 i=1,8
20     write(5,*) rodd(i)
      do 30 i=1,8
      amp(i)=sqrt((reven(i)**2)+(rodd(i)**2))
30     write(6,*) amp(i)
      do 40 i=1,8
      ph(i)=acos(reven(i)/amp(i))*180/3.1415
40     write(7,*) ph(i)
      end
```

PROGRAM DIFF.FOR

```
      real*4 f1(16),f2(16),dif(16),reven(8),rodd(8),amp(8),ph(8)
      open(1,file='xcorrok.dat')
      open(2,file='xcorrko.dat')
      open(4,file='xeven.dat')
      open(5,file='xodd.dat')
      open(6,file='xamp.dat')
      open(7,file='xph.dat')
      do 300 i=1,16
      read(1,*) f1(i)
      read(2,*) f2(i)
      dif(i)=f1(i)-f2(i)
      sum=sum+dif(i)
300    write(*,*) dif(i)
      write(*,*)
      write(*,*) sum*100
      do 500 i=0,7
      reven(i+1)=(f1(9+i)+f1(9-i))/2
500    rodd(i+1) =(f1(9+i)-f1(9-i))/2
      do 10 i=1,8
10     write(4,*) reven(i)
      do 20 i=1,8
20     write(5,*) rodd(i)
      do 30 i=1,8
      amp(i)=sqrt((reven(i)**2)+(rodd(i)**2))
30     write(6,*) amp(i)
      do 40 i=1,8
      ph(i)=acos(reven(i)/amp(i))*180/3.1415
40     write(7,*) ph(i)
      end
```