IP-VLBI サンプラー評価試験と結果（その２）

近藤哲朗
2000 年 10 月 13 日

２０００年9月１２日から１３日にかけて、鹿島３４ｍアンテナと２６ｍアンテナを使用して、IP-VLBI サンプラーボードの評価を行ったのでその結果について報告する。なお今回は 4MHz サンプリングまでの評価を行うことができた。

1　IP-VLBI サンプラー評価観測

図 1.1 に評価試験観測のセットアップを示す。２６ｍアンテナの IF 信号は光ファイバで３４m庁舎まで伝送した後、ビデオ信号に変換した。３４m庁舎では３４ｍアンテナおよび２６ｍアンテナのビデオ信号をプログラマブル・ローパスフィルターに通した後、IP-VLBI サンプラーボードに入力した。サンプラーボードのサンプリン

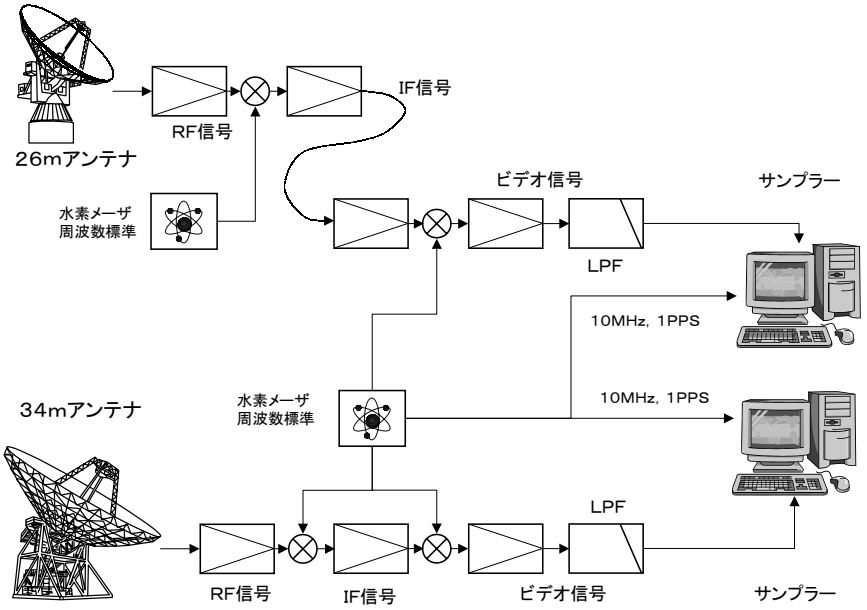

図 1.1. IP-VLBI サンプラーボード評価試験観測セットアップ。

グ周波数は最高 4MHz まで変えて、１ビットサンプリングによるデータ収集を行った。サンプラーにはそれぞれ周波数標準 (３４m水素メーザ）の 10MHz と 1PPS 信号を接続した。それぞれのサンプラーボードで収集したデータをオフライン処理することにより、ボードの性能評価および相関処理プログラムの速度評価を行った。当日の気象状況、観測局情報、観測スケジュールを表 1.1、1.2、1.3 に示す。３４ｍアンテナ側水素メーザのレートをずらしているのは、２６ｍアンテナ間の短基線において確実にフリンジ回転を起こすことにより、ＰＣＡＬ信号同志の相関を避けるためである（３４m－２６mアンテナといった短基線ではフリンジ周波数が非常に低いため、相互相関処理において PCAL 信号同士の相関が非常に強く検出されてしまい、本来のフリンジ検出が困難と

なる)。また 4MHz サンプリング時のＬＰＦの設定が 1.59MHz となっているが、これは使用したプログラマブルフィルターの上限周波数の制限のため、2MHz とすることができなかったためである。なお、この観測は測地ＶＬＢＩ実験 (JPNTI4）に相乗りして行った。

表 1.1. IP-VLBI サンプラーボード評価観測時の気象データ

日時 (JST)	天候	気温（℃）	気圧（hPa）	湿度（%）
2000/9/12 14:30	晴れ	30.4	1006.8	80
2000/9/13 08:10	くもり	22.9	1012.6	97
2000/9/13 13:25	くもり	22.9	1012.0	98

表 1.2. 観測局情報

観測局	鹿島 26m アンテナ－鹿島 34m アンテナ
受信周波数（ベースバンド）	8499.99 MHz
ＰＣＡＬ信号	ON
３４m水素メーザーレートオフセット	-2.1×10^{-9} s/s
サンプリング周波数	40kHz – 4MHz
A/D 分解能	1 ビット

表 1.3. 観測スケジュール

時刻（ＵＴ）	電波源	サンプリング周波数	LPF(MHz)	vlbi1 26m データファイル名	vlbi 2 34m データファイル名
256D					
05:46:40-05:47:50	3C273B	4MHz	1.59	26m001.dat	34m001.dat
05:53:30-05:54:40	4C39.25	4MHz	1.59	26m002.dat	34m002.dat
ここまでＫ３ビデコンモニター出力					
以下Ｋ３ビデコン背面出力使用					
23:05:40-23:06:50	3C273B	4MHz	1.59	26m003.dat	34m003.dat
23:11:00-23:12:10	0749+540	4MHz	1.59	26m004.dat	34m004.dat
257D					
01:48:50-01:50:00	3C273B	1MHz	0.5	26m005.dat	34m005.dat
02:35:00-02:36:10	1334-127	500kHz	0.25	26m006.dat	34m006.dat
02:57:10-02:58:20	4C39.25	40kHz	0.02	26m007.dat	34m007.dat
04:31:10-04:32:20	3C273B	2MHz	1.0	26m008.dat	34m008.dat
以下共通雑音入力					
07:18:00-07:18:10	COMMON	4MHz	1.59	4mchk-1.dat	4mchk-2.datt
07:20:00-07:20:10	COMMON	2MHz	1.0	2mchk-1.dat	2mchk-2.datt
07:22:00-07:22:10	COMMON	1MHz	0.5	1mchk-1.dat	1mchk-2.datt

Ｋ３ビデコンのモニター出力は、サンプラー入力としてはレベルが低すぎたため、Ｋ３背面出力を使用したデータの処理を行った。図 1.2、1.3 にビデコンモニター出力信号および背面出力信号を示す。

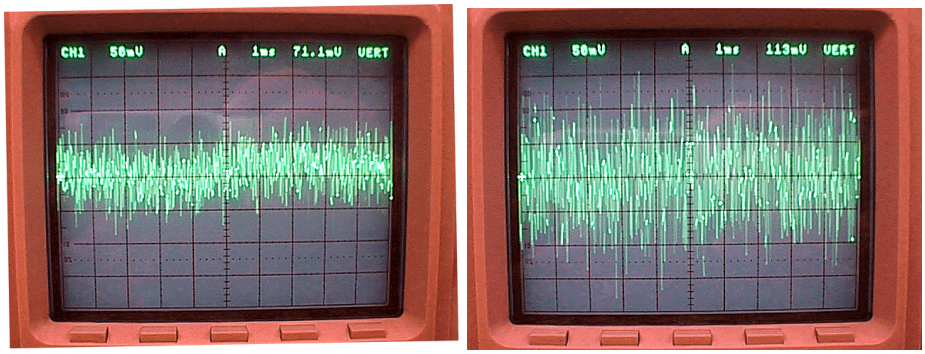

図 1.2. K３ビデコンモニター出力（1.59MHz のＬＰＦ出力）信号の様子。左が２６ｍアンテナからの信号、右が３４ｍアンテナからの信号である。それぞれ 50mV/div。

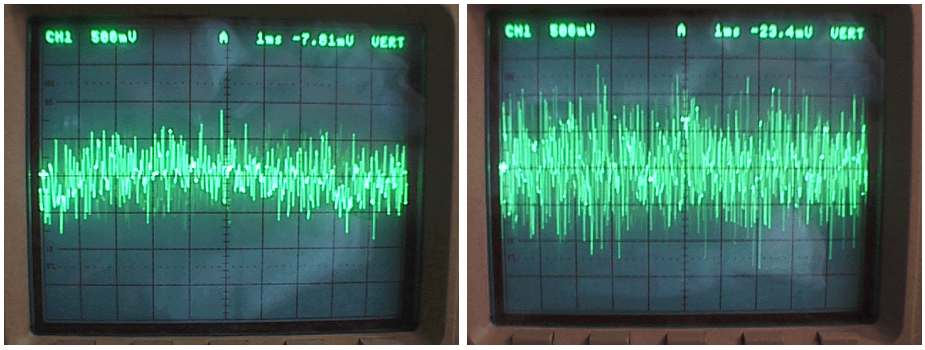

図 1.3. K３ビデコン背面出力（1.59MHz のＬＰＦ出力）信号の様子。左が２６ｍアンテナからの信号、右が３４ｍアンテナからの信号である。それぞれ 500mV/div。

2　相関処理手法

相関処理プログラムは PV-WAVE 言語を使用して開発したものであり、処理速度向上を目指して以下の３通りの手法の比較を行っている。いずれも、時系列での処理を行う相関処理であり、ハードウェアで行う相関処理（図 2.1）と同じ原理に従ってプログラミングしている。相関ラグ指標の定義は図 2.2 に従っている。

手法Ａ： １ビットサンプリングデータを１ビット毎に分解するのに、１ビット毎の演算を行って抽出する。フリンジストッピングは近似なしの関数値を使用する。（PV-WAVE での演算の場合、近似を行わない方が高速である。）原理図を図 2.3 に示す。

PV-WAVE プログラム例

```
xdat=intarr(numbyte*8)
 patn=[1B,2B,4B,8B,16B,32B,64B,128B]
 ipos=long(0)

 for n=long(0),numbyte-1 do begin
      wb=ia(n)
      ;print,wb
      for i3=0,7 do begin
         kk=wb and patn(i3)
         if kk ne 0 then xdat(ipos)=1 else xdat(ipos)=-1
         ipos=ipos+1
```

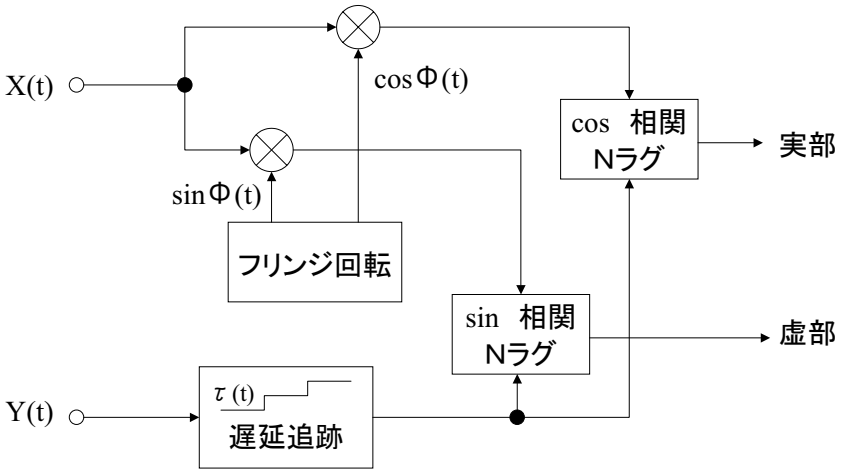

図 2.1. 相関処理の原理。

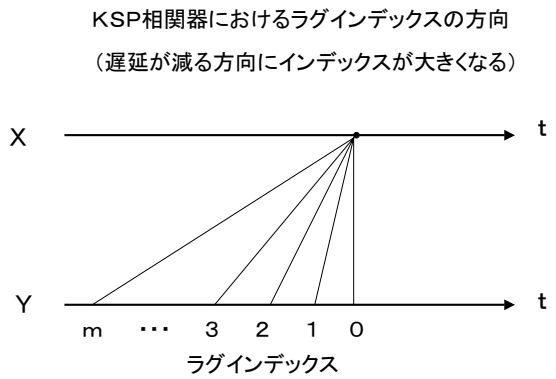

図 2.2. 相関処理の原理。

```
      endfor
endfor
```

手法Ｂ： １ビットサンプリングデータを１ビット毎に分解するのに、バイト単位で、テーブルを参照し、一度に８ビットの分解を行う。フリンジストッピングは近似なしの関数値を使用する。原理図を図 2.4 に示す。

PV-WAVE プログラム例

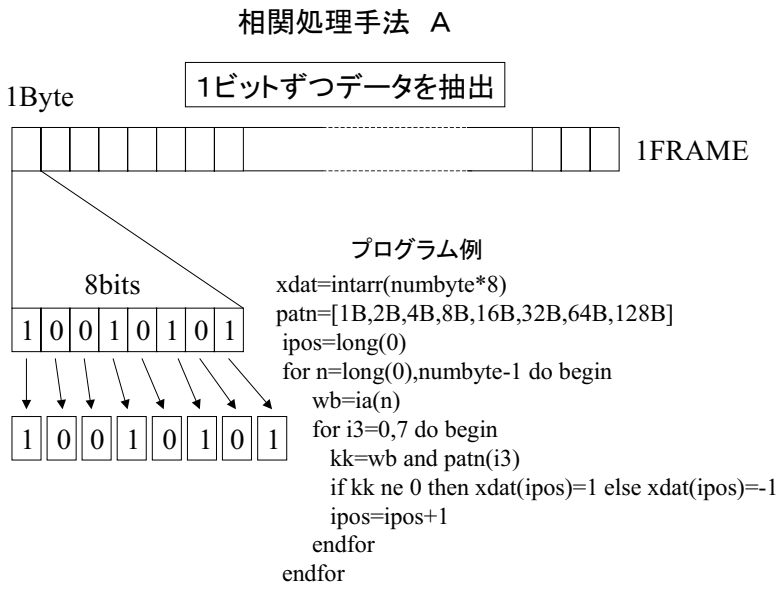

図 2.3. 手法Aの原理図。

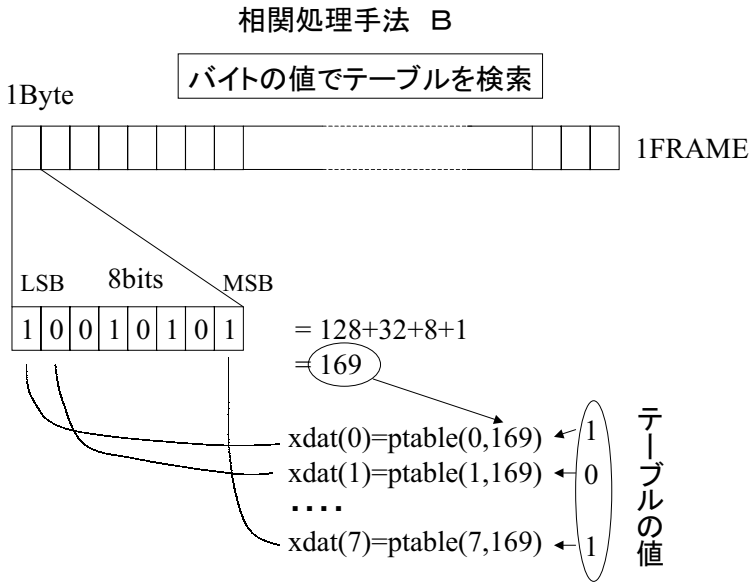

図 2.4. 手法Bの原理図。

メインでのテーブルセット

```
patn=[1B,2B,4B,8B,16B,32B,64B,128B]
ptabl=intarr(8,256)
ipos=long(0)
for ii=0,255 do begin
```

```
        wb=byte(ii)
        for i3=0,7 do begin
           kk=wb and patn(i3)
           if kk ne 0 then ptabl(i3,ii)=1 else ptabl(i3,ii)=-1
        endfor
  endfor
```

以下サブルーチン内

```
  xdat=intarr(numbyte*8)
   ia=bytarr(numbyte)
   ierr=-1
   readu,luid,ia
   if eof(luid) then goto, exithere
   ierr=0

   kp=indgen(numbyte)
   xdat=ptabl(*,ia(kp))

   exithere:
```

手法C： 1ビットサンプリングデータを1ビット毎に分解せずに、バイト単位の相互相関テーブルを参照して、相関積分まで行ってしまう。フリンジストッピングは3レベル近似を使用する。(1,0 の演算結果のテーブルを使用するため、3レベル近似または2レベル近似の使用が不可欠である。) 原理図を図 2.5 に示す。

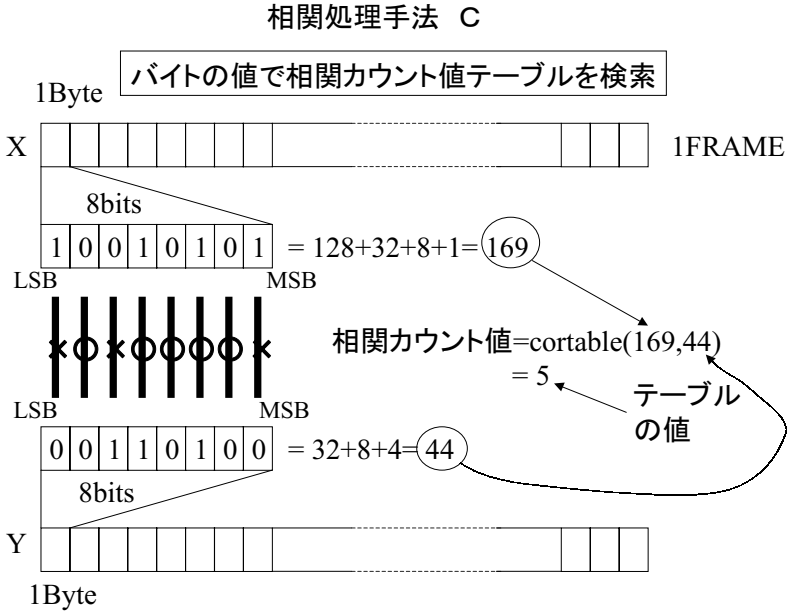

図 2.5. 手法Cの原理図。

PV-WAVE プログラム例

メインでの相関テーブルセット

```
   cortabl=bytarr(256,256)
   for i=0,255 do begin
     a=byte(i)
     for j=0,255 do begin
        b=byte(j)
        c=(a xor b)
        bsum=0
        for k=0,7 do begin
           if (c and 1B) eq 0 then bsum=bsum+1
           c=ishft(c,-1)
        endfor
        cortabl(i,j)=bsum
     endfor
   endfor
```

サブルーチン例

```
Pro xroscor_b,lagbyte,xdat1,xdat2,fcos,fsin,numbyte,$
        cortabl,xycsum,xyssum,totcntc,totcnts
       indexc1=where(fcos lt 0.0,numc1)
       indexc3=where(fcos ne 0.0,numc3)
       indexs1=where(fsin lt 0.0,nums1)
       indexs3=where(fsin ne 0.0,nums3)
       xdatc=xdat1
       if numc1 gt 0 then xdatc(indexc1)=(not xdatc(indexc1)) ; correspond to xdatc*(-1)
       xdats=xdat1
       if nums1 gt 0 then xdats(indexs1)=(not xdats(indexs1))
       ;
       ydat0=shift(xdat2,-lagbyte/2)  ; initial offset

       if numc3 gt 0 then xc=xdatc(indexc3)
       if nums3 gt 0 then xs=xdats(indexs3)
       for k=1,8 do begin   ; bit shift 8 times
         ydat1=ydat0
         for m=1,lagbyte do begin
             lag=(m-1)*8+k-1
               if numc3 gt 0 then xycsum(lag)=xycsum(lag) $
                                              +total(cortabl(xc,ydat1(indexc3)))
               if nums3 gt 0 then xyssum(lag)=xyssum(lag) $
                                              +total(cortabl(xs,ydat1(indexs3)))
             ydat1=shift(ydat1,1)
         endfor

         ydat2=(ydat0 and 128B)   ; 1B should be used to assure byte expression
         ;                          ;  Pick up LSB
         ydat2=shift(ydat2,1)    ;  Shift array component right ward
         ydat2=ishft(ydat2,-7)    ;  Shift MSB to LSB
         ydat0=ishft(ydat0,1)     ;  1bit shift leftward
         ydat0=ydat0 or ydat2
       endfor
```

```
      totcntc=totcntc+numc3*8.0
      totcnts=totcnts+nums3*8.0
    end
```

手法A，Bは１ビットデータへの分解の方法が異なるだけである。手法CはA，Bとは本質的に異なる手法であり、１ビット毎へのデータの分解を行わず、バイト単位で相関積分結果まで得てしまうため、高速化が期待できる。高速化のためフリンジストッピングもバイト単位で行う。

図 2.6,2.7,2.8 にそれぞれ、１，２，４MHｚサンプリングデータに対して得られたフリンジを示すが図に示されるようにすべてのサンプリング周波数においても、手法によらず良好なフリンジが検出された。なお、電波源は 3C273B で、積分時間は５秒である。また相関係数には何の補正も行っていない。

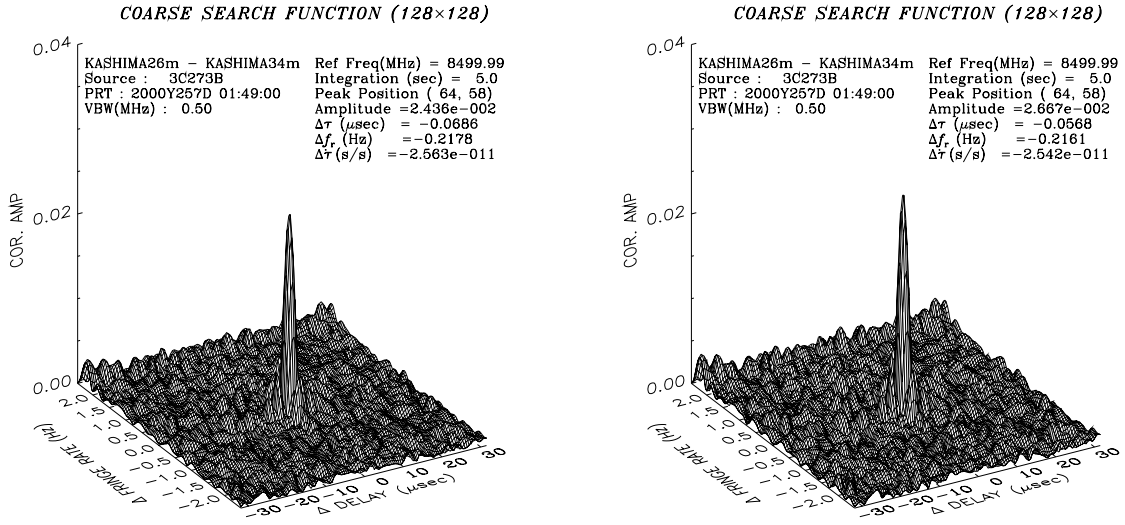

図 2.6. １MHｚサンプリング時のフリンジ。電波源は 3C273B。左が手法B、右が手法C。

共通信号を入力した場合の相関強度の違いを調べた結果を表 2.1 に示す。フリンジサーチプログラムで相関係数の最大をサーチして求めた相関強度である。この場合、フリンジ回転はないので、手法による相関係数の違いは発生しない。

表 2.1. 共通信号入力時の相関係数

サンプリング周波数	相関係数
1 MHz	0.9447
2 MHz	0.8996
4 MHz	0.8705

サンプリング周波数が高くなるほど相関強度が減少し、４MHｚサンプリング時には約１３%の現象となっている。サンプリング周波数を変化させた場合、ローパスフィルター遮断周波数のみ変化させている。つまり入力信号の帯域幅はサンプリング周波数が高くなるほど広くなり、つまり電圧の振幅は大きくなるため、ここで見られる減少は入力信号レベル不足によるとは考えにくい。サンプリング信号のジッターが疑われる。

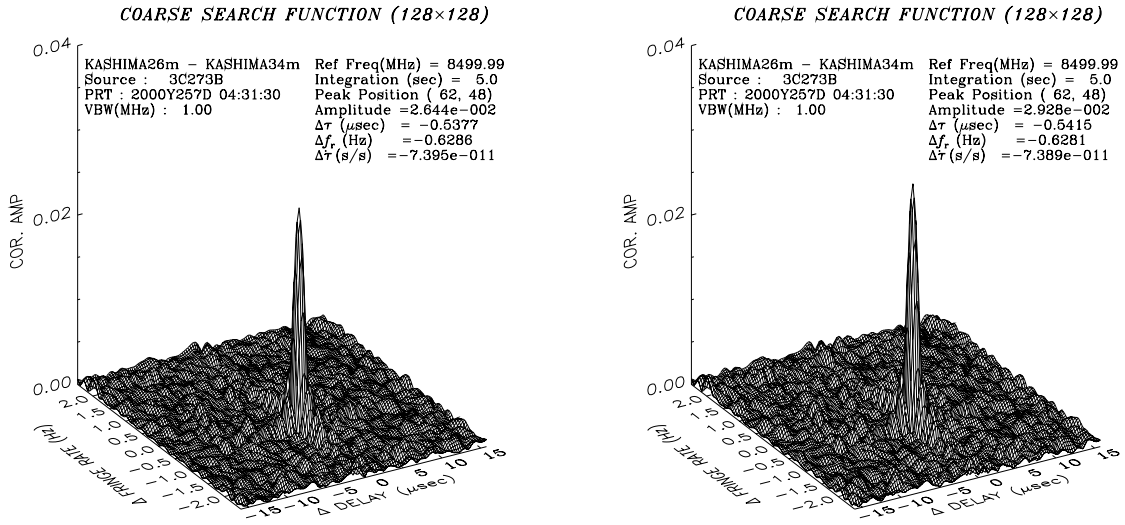

図 2.7. 2MHzサンプリング時のフリンジ。電波源は 3C273B。左が手法B、右が手法C。

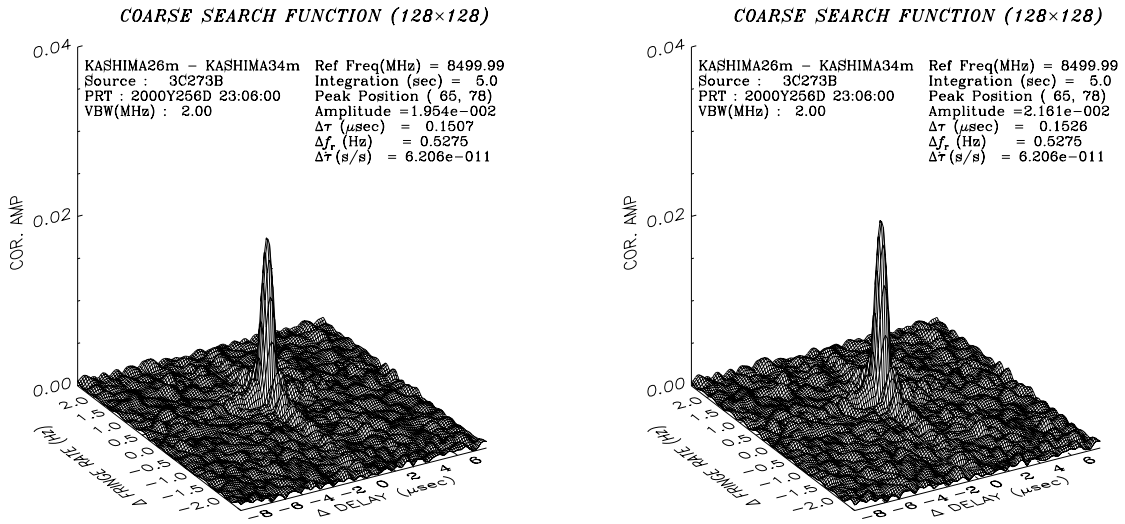

図 2.8. 4MHzサンプリング時のフリンジ。電波源は 3C273B。左が手法B、右が手法C。

3 処理時間の比較

手法の違いによる処理速度の違いを比較した。処理に使用しているプロセッサーは Pentium II 300MHz、使用言語は PV-WAVE である。PV-WAVE は配列計算を得意とする言語であるがインタープリタ言語である。相関処理にはＦＦＴ法もあるが、ラグ数が小さい場合、ここで使用している直接相関法の方が有利と考えられる（まだ未比較ではある）。特に、手法Ｃは直接相関法に特化した手法である。

手法および相関処理ラグ数等の条件を変えて、１秒積分に要する時間を比較した。１秒積分に要する時間が１秒以上の場合、実時間処理が行えないことを意味する。表 3.1 および図 3.1 に手法の違いによる処理速度の違いの比較結果を示す。

表 3.1. 手法の違いによる処理時間の比較

手法	ラグ数	サンプリング周波数 (MHz)	PP(秒)	処理単位（ビット数）	１秒積分に要する時間（秒）
A	64	4	1	20000	201.6
B	64	4	1	40000	170.6
B	64	4	1	20000	125.4
B	64	4	0.2	40000	165.4
B	64	4	0.2	20000	127.0
C	64	4	0.2	40000	18.2
C	64	4	0.2	20000	20.0
C	16	4	0.2	20000	6.6
C	8	4	0.2	20000	5.6

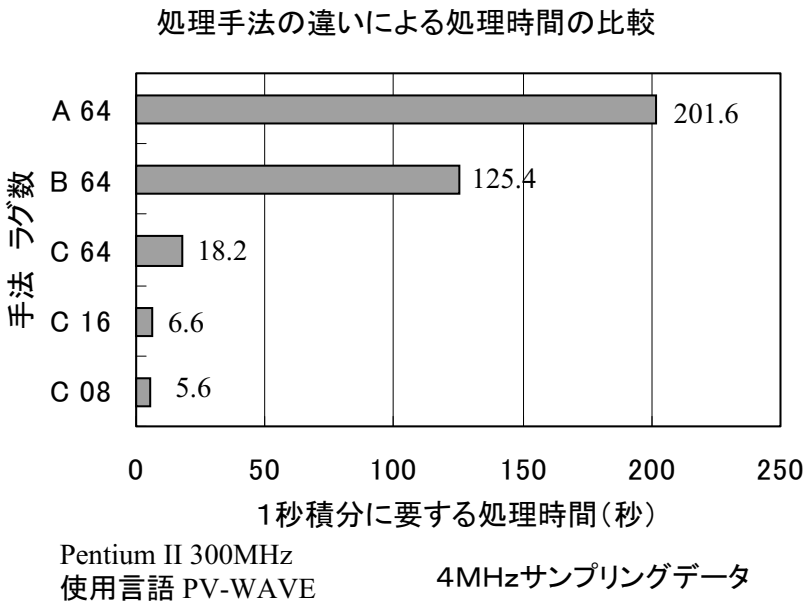

図 3.1. 処理手法の違いによる処理時間の比較

手法Ｂにおいては処理単位を 20000 ビットから 40000 ビットとした場合、処理時間が増えるがこれは、ビットを分解したデータを格納する配列の引数が通常の２バイト整数の範囲を超えて４バイト整数となるためのオーバーヘッドの発生によると思われる。手法Ｃにおいては、40000 ビットの方が高速となるが、これはバイト単位で処理するため引数の範囲が２バイト整数で収まるためである。ともかく、手法Ｃにより手法Ａに比べて１０倍以上の高速化が達成された。しかしながら、８ラグ相関においても１秒積分に６秒程度かかっているが、PV-WAVE ではなくＣへの移植で更なる高速化は期待できる。

手法Ｃを使用し、サンプリング周波数、相関ラグ数の違いによる処理速度の比較を行った結果を表 3.2 および図 3.2 に示す。PP は 0.2 秒、処理単位は 40000 ビットとした。

表 3.2. サンプリング周波数および相関ラグ数の違いによる処理時間の比較。手法C、PP は 0.2 秒、処理単位は 40000 ビット。

ラグ数	サンプリング 周波数 (MHz)	1 秒積分に 要する時間（秒）
64	1	5.0
32	1	2.6
16	1	1.6
8	1	1.2
64	2	9.2
32	2	5.2
16	2	3.4
8	2	2.2
64	4	18.4
32	4	12.6
16	4	6.6
8	4	4.8

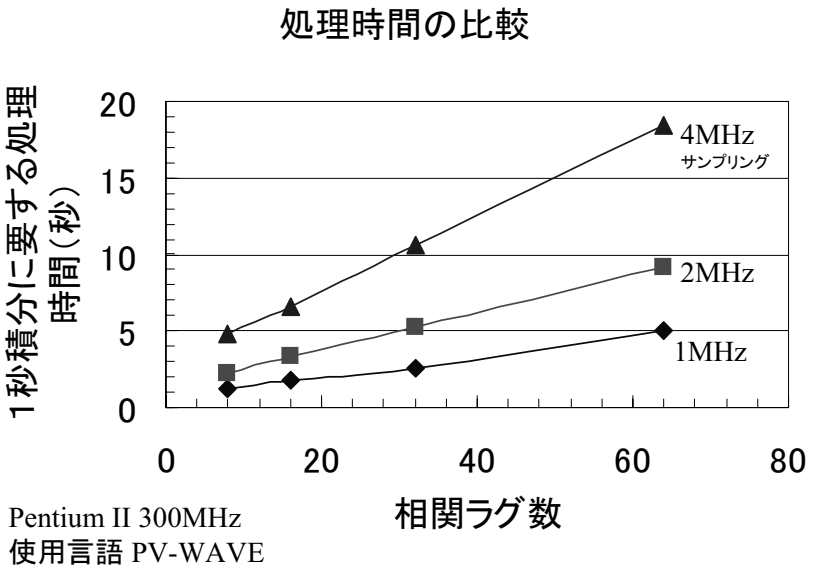

図 3.2. サンプリング周波数および相関ラグ数の違いによる処理時間の比較。手法C、PP は 0.2 秒、処理単位は 40000 ビット。