SGP MEMO NO. 1

NATIONAL RADIO ASTRONOMY OBSERVATORY
SOCORRO, NEW MEXICO
VERY LARGE ARRAY PROJECT

VLA COMPUTER MEMORANDUM NO. 137

A DIGITAL MAP-MAKING SCHEME

B. G. Clark

July 1977

We have been vigorously investigating the application of optical devices to VLA data handling problems for the last two years. It appears that even after this time that path is still dimly illuminated and fraught with snares. Before embarking on it, we should have a look at other paths; this memo comprises such a look at a "moderate size" digital alternative. Other digital alternatives are a "large system" approach, typified by Staran, at over twice the cost of this system and correspondingly more powerful, or to simply reduce our expectations to the level that our currently on-order PDP 11/70-array processor system could handle them.

The approach of this memo is based on the equipment of Figure 1. For purposes of this memo, the array processors are taken to be FPS120B processors with fast memory, and the computers to be modest PDP 11/34's. This memo proceeds in the following sections: I. Performance of this engine on the cannonical problem of a 2048x2048x5 transform, with some consideration of what is happening inside the system, II. Remarks on extensions to more and to less severe problems, III. A growth plan for the system, IV. Specifications and cost estimates.

I. SYSTEM PERFORMANCE

The cannonical case has been taken to be a 2048x2048x5 transform.

For convenience I have assumed that input convolution and map-making proceed synchronously, which may not be the case (convolution will require more time on the center of the 5 w slices, because that is where most of the data are). In this case, each w slice is handled in about 45 seconds, resulting in a final output map every 3.5 minutes.

The system shows two minicomputers. The host computer has the dual functions of control and input data buffer. This will not be a severe load, because the input data set is notably smaller than the data found elsewhere; instead of data reduction we have data expansion. In the cannonical 2048x2048x5 case, the five maps contain 2.1×10^7 complex data points, whereas the input data set contains only $351 \times 4320 = 1.5 \times 10^6$ complex numbers and a similar number of u,v coordinate pairs for a 12 hour observation; the data are expanded by a factor of 7.

The form of input convolution assumed in n x n square area in the output plane is affected by each input point and that the convolving fraction is a separable function of u and v. There are K convolving functions, depending on the truncated portions of u and v. Specifically, the convolution is equivalent to the following Fortran program.

```
      DIMENSION F(K,N)
      COMPLEX UVPLAN (N,LROW), DATUM, T(N)
10    CALL INPUT (DATUM,U,V)
C     SELECT CONVOLUTION FUNCTION
C     ASSUMED SEPARABLE F(u,v) = F(u)*F(v)
      IUCONV = MOD(U,K)
      IUTRUN = U/K
      IVCONV = MOD(V,K)
      IVTRUN = V/K
      IF(IVTRUN.NE.IPREV)GO TO 40
```

```
      DO    20    I=1,N
20    T(I) = DATUM*COMPLX ( F(IUCONV,I),0)
      DO    30    I=1,N
      DO    30    J=1,N
30    UVPLAN (I,IUTRUN+J)
          = UVPLAN (I,IUTRUN+J) + T(J) * F(IVCONV,I)
      GO TO 10
40    CONTINUE
      IPREV = IVTRUN
C     OUPUT ROW 1 AND PROMOTE OTHER ROWS ETC.
```

The inmost loop requires for each complex number two multiplies and two adds, along with some indexing operations. This process can probably run at near memory speed - two fetches and two stores require 1 1/3 microsecond. With indexing, loop control, etc. the inmost loop will run at about 2 microseconds per cycle (which handles one complex number). If our goal is to process all 1.5×10^6 complex points in 220 seconds, we are limited to ~75 executions per data point, or a 8 x 8 input convolution. If as discussed below, we have to do each point twice, this drops to 6 x 6. An 8 x 8 Gaussian convolution seems to reduce aliased sources to near zero by 1.3 field semidiameters from field center.

The row transform proceeds very rapidly - a 2048 complex FFT runs in about 12 ms, so that the entire map is transformed in about 25 seconds. This will be slowed down by up to an additional 12 seconds due to the interference of the I/O operations with the computation's memory accesses. It looks, then, like an allowance of 45 seconds for a 2 K x 2 K complex map is reasonable.

The transposing memory would have special interfaces which would cause the map to be stored in rows and recovered in columns. Its normal mode of operation would be read/write, so that while it is retrieving a word for AP-3 it would also be storing a word from AP-2

into that location, so that these two AP's would be running absolutely synchronously, and would have hardware interlocks to compel this synchrony. Probably a similar arrangement will be necessary between AP-1 and AP-2.

Operation in AP-3 is like that in AP-2 with two exceptions. The first is that the output to the map merging computer will proceed much more slowly, so that buffer management must be done more carefully. The second is that the phase factors for the third dimension of the 3 dimensional transform would be applied (the third dimension is, of course, a classical FT, since there is only one output point) so that the map merging computer could complete the transform in the third dimension by a simple add, allowing it to complete this task in less time than the AP's take to do their FFT's. This phasing should add about 3-5 seconds to the processing time of AP-3.

The utilization of the Hermitian property in 3 dimensions is not entirely straightforward. To minimize both the size of the transposing memory and any mass-store data transfers, it seems to me necessary to handle the w dimension last. There are then apparently only two, rather unattractive options: 1) use the Hermitian property at the end, in which case the intermediate u,v planes are not Hermitian, and the transposing memory must consist of 4 million <u>complex</u> words. 2) Use the Hermitian property at the beginning to construct u,v planes of size 1K x 2K, and which transformed, result in a 2K x 2K real map, which seems to require access to both +w and -w data to construct, and therefore doubles the convolution time. The second option is the one discussed here, though we shall continue to look for a better algorithm, as well as considering combining short time slices as one would do for an optical processor.

The map merging computer handles the final step of the 3rd dimension of the Fourier transform, and in addition handles formatting and communication with the rest of the computer system. It should have available about 300 MBytes of disk for storage of the intermediate

maps for a very large transform, say 4096x4096x10.

II. OTHER PROBLEMS

For less severe problems, the system speeds up nicely. For instance, a 1024x1024x4 transform could be stored directly in the transposing memory and AP-3 would supply the final output map in 45 seconds. For smaller maps yet, one would have a hardware mode that used only one corner of the transposing memory. Note though, that for small maps, doing any input convolution (except for pillbox averaging) becomes a severe limitation, since this time is proportional to the amount of input data and independent of map size. At some point it becomes more profitable to make a larger map and discard the outer portion (equivalent to a sinc convolution) than to convolve the input data.

For more severe problems, 4K x 4K or 8K x 8K maps, the machine slows down very much. For instance, a 4K x 4K transform requires four passes through the machine, each handling a quarter of the array, and each running a little less than a minute. An amusing sidelight is that it looks like the way to do the job is to have AP-2, after doing a 4096 transform, simply discard 3/4 of the data; it is faster and easier to regenerate these 13 million numbers from the original 1.5 million input points via a FFT than it is to write them on disk and recover them.

An 8K x 8K transform similarly requires 16 passes, and therefore takes about a quarter of an hour for each two dimensional map.

III. A GROWTH PLAN

One of the principal strengths of a digital system is that it may grow gracefully as our needs and our understanding of the system increase. The primitive part of the system is already on order - an FPS AP120B array processor (already received) and a PDP 11/70 host computer. This system has not been configured into

the final system - its usefulness as a map manipulation system will probably occupy it too much of the time. However, it is available to use as part of the system, and will be successively rescued from the system as the increasing work load is taken over by purchasing an additional component of the final system. At each of these junctures, we can review where we stand, and, in light of the up-to-date experience, decide whether we are going in the right direction. My suggested development plan is given below.

1977 We shall make the 11/70, AP120B system operational. It's capacity is, approximately, a 2K x 2K map in 15 minutes.

1978 Purchase of AP-2 and the new Host, plus the disks of the map merging computer. This will be only slightly less capable than the 11/70 system by itself. Also this year we should procure the transposing memory and design its special purpose interfaces. This, with the use of the 11/70 system, should give us the final throughput rates, but without the input convolution.

1979 Purchase of AP-1. This gives us the entire system.

1980 Purchase of AP-3 and map merging computer frees the 11/70 system for map manipulation and analysis.

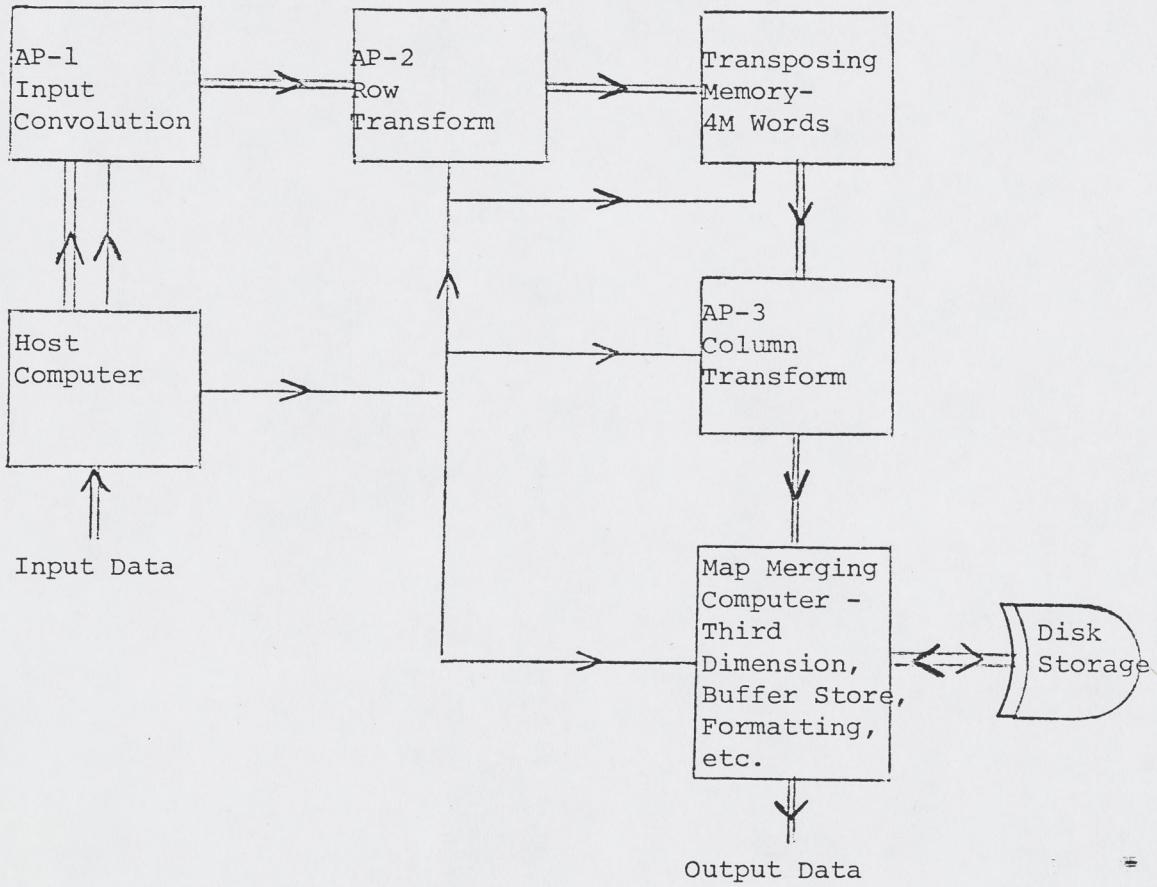
IV. COST ESTIMATES

Host Computer		
CPU, Memory, & Options	$26K	
Cartridge Disks	18K	
Cabinets, Cables, & other hardware	8K	$52K
AP-1		
CPU, Interfaces, Program Memory Options	$56K	
Table Memory	6K	
Main Memory	86K	$148K

AP-2 and AP-3		
CPU, Interfaces, Program Memory, Options	$56K	
Table Memory	3K	
Main Memory	34K	
	94	$188K
Map Merging Computer		
CPU, Memory, and Options	$26K	
Pack Disks	92K	
Cabinets, Cables, and Other	8K	$126K
Transposing Memory		
4 M Words x 24 Bits	$220K	
Special Interfaces - Parts	12K	$232K
TOTAL		$746K

In addition, the sorting system needs to be 3-stage, rather than the one or two stages needed for the optical processor. This is an increase in cost of ~$120K, so for price comparisons, this system should be figured as about $870K, vs the optical processor at $1200K.

Software costs will be high for both the digital and optical system, but, I think, comparable, with perhaps a slight edge to the digital system. The inconvenience of working in multiple CPU's is slightly more than offset by the particularly messy interfaces to a complicated analog device, and the necessity to support several processor modes to work around its accuracy problems.



= control information

= main data flow

AP = Array Processor

FIGURE 1: BLOCK DIAGRAM