THE AP 120 B ARRAY PROCESSOR

by

Walter Wagner
KRUPP ATLAS-ELEKTRONIK
2800 Bremen, Germany

ABSTRACT The Floating Point Systems array processor AP 120 B is briefly presented. An overview of the architecture is given. The most important features and the use of the AP software are described.

THE AP 120 B ARRAY PROCESSOR

For the past two years, a Floating Point Systems AP 120 B array processor has been used by the KRUPP ATLAS-ELEKTRONIK research group for processing sonar signals. The processor is connected to an HP 2100 minicomputer as a host computer. Normally, analogue data are collected on tape recorders during sea trials. These data have to be processed in the laboratory. Because there is only a small research group, every scientist is obliged to develop his own signal processing program. As a consequence the system must be easy to handle.

The acquisition of the array processor has produced the following main improvements for the signal processing facilities of our sonar research group:

1. A lot of programming time has been saved because a library consisting of more than 200 routines was delivered with the system and could be used easily by everybody.

2. Execution time of signal processing programs was reduced by a factor of about 60 to 80 depending on the program, by the use of simple FORTRAN programming techniques.

3. With some more programming effort, special processing problems could be solved in real time. A good example of this is beamforming in real time. In this way a speed-up factor of several hundred can be achieved.

A short overview of the array processor is given in the following. For detailed information please refer to the FPS manuals.

The AP 120 B is a full floating point array processor, synchronous and pipelined. Once the data have been converted to floating point numbers, all calculations are done in this format and normally no further consideration of overflow - underflow is necessary. For output purposes, conversion is possible to any integer format up to 32 bit.

Fig. 1 shows a block diagram of the processor's hardware with its various memories, address calculators and the two arithmetic units, the floating point adder and the floating point multiplier.

The system has a 6 MHz clock corresponding to 167 ns cycle time. The main data memory is available in two versions

a slow version with 333 ns cycle time

and

a fast version with 167 ns cycle time.

The following table gives an example of the calculation speed for both memories:

	fast memory	slow memory
1024 points real FFT	2.7	4.2 ms
vector multiply	0.8	1.3 us/point
vector add	0.8	1.3 us/point
vector multiply, multiply and add	1.5	2.3 us/point

The fastest vector multiplication between data of two different memories takes 500 ns/point.

The normal programmer does not need to know the structure of the processor in detail. He is not aware of the different types of memory such as the table memory, the program source memory, the main data memory and the scratch pad memories.

The only thing he should not forget is the size of the main data memory.

Fig. 2 shows an example of the mathematical library. Its documentation is self-explanatory.

DISCUSSION

R. Seynaeve Do you frequently use chaining and Assembler?

W.G. Wagner We did not buy the Vector Function chainer, I have only mentioned this feature. I sometimes use chaining myself; this is very easily accomplished by simple assembler coding. We do not use Assembler coding very often, because most of our requirements are met by the given library functions. Assembler coding is used only for special real-time applications.

D.V. Crowe Is anyone using the Floating Point Systems Fortran compiler?

W.G. Wagner No! As far as I know not in Europe, but it is in use in the U.S.

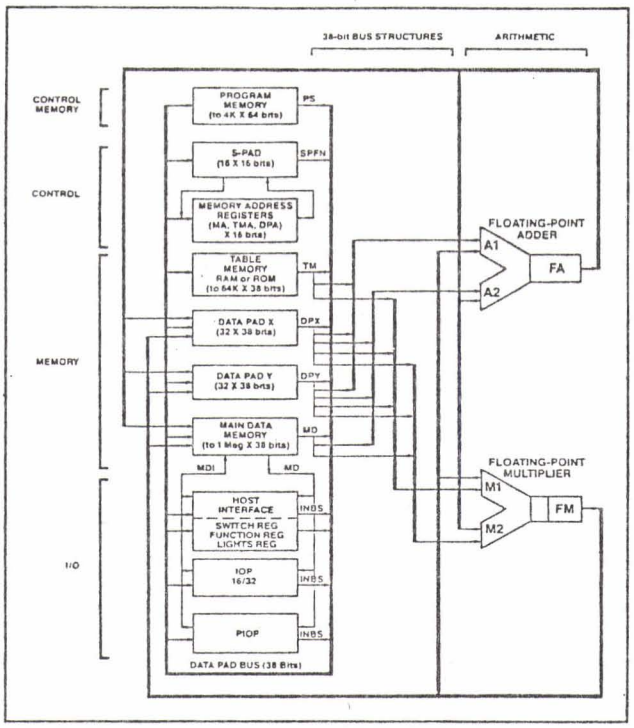

FIG. 1 FUNCTIONAL BLOCK DIAGRAM

```
**********                                              **********
*        *                                              *        *
*  VMUL  *          ---- VECTOR MULTIPLY ----           *  VMUL  *
*        *                                              *        *
**********                                              **********

PURPOSE:        To multiply the elements of two vectors.

FORTRAN CALL:   CALL VMUL(A,I,B,J,C,K,N)

PARAMETERS:     A = Source vector base address
                I = A address increment
                B = Source vector base address
                J = B address increment
                C = Destination vector base address
                K = C address increment
                N = Element count

FORMULA:        C(mK) = A(mI) * B(mJ)   for m=0 to N-1

DESCRIPTION:    Multiplies N elements of the vector beginning at address
                A by N elements of the vector beginning at address B,
                and stores results in the vector beginning at
                address C.

EXAMPLE:        CALL VMUL(0,1,200,1,400,1,100)
                Stores into AP120B main data memory locations
                400,401,....,498,499 the product of the numbers
                in locations 0 and 200, 1 and 201, .... ,
                98 and 298, 99 and 299.

EXECUTION       BEST    TYPICAL    WORST      SETUP(us)
TIME/LOOP:      0.5 (B)   0.8        1.0        2.7    (167 ns memory)
 (us)           1.0 (A)   1.3        1.5        1.2    (333 ns memory)

PROGRAM SIZE:   17                                     (167 ns memory)
 (AP words)     11                                     (333 ns memory)

-------------------------------------------------------------------------

APAL CALL:      JSR VMUL
SCRATCH:        SP(0,2,4,14,15),DPX(0,1),DPY(0),       (167 ns memory)
                  FA,FM,MD,TM
                SP(0,2,4,6),DPX(0),FA,FM,MD            (333 ns memory)
EXTERNALS:      SPFLT                                  (167 ns memory)
                None                                   (333 ns memory)
```

FIG. 2 MATHEMATICAL LIBRARY - EXAMPLE